Massively Parallel Electromagnetic Transient Simulation of Large Power Systems

by

Zhiyin Zhou

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Energy Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Electromagnetic transient (EMT) simulation, which is widely utilized in power system planning and design, is one of the most complex power system studies that requires detailed modeling of the study system including all frequency-dependent and nonlinear effects. Large-scale EMT simulation is becoming commonplace due to the increasing growth and interconnection of power grids, and the need to study the impact of system events of the wide area network. To cope with enormous computational burden, the massively parallel architecture of the graphics processing unit (GPU) is exploited in this work for large-scale EMT simulation. A fine-grained network decomposition, called shattering network decomposition, is proposed to divide the power system network exploiting its topological and physical characteristics into linear and nonlinear networks, which adapt to the unique features of the GPU-based massive thread computing system. Large-scale systems, up to 240,000 nodes, with typical components, including synchronous machines, transformers, transmission lines and nonlinear elements, are tested and compared with mainstream simulation software to verify the accuracy and demonstrate the speed-up improvement with respect to sequential computation.

Power electronic devices are widely utilized in modern power grid, especially for AC/DC converters in HVDC systems. The proposed fine-grained decomposition algorithm can also be applied in the simulation of multiple levels modular multilevel converter (MMC) consisting of Insulated Gate Bipolar Transistors (IGBTs) based on linear and nonlinear switch models, which effectively enhances the simulation performance and extends the system scale by parallelizing the calculation and maintaining the convergence during the computation.

Preface

This thesis is an original work by Zhiyin Zhou. As described in the following, several chapters in this thesis have been published or submitted as journal articles. My supervisor, Dr. Venkata Dinavahi has provided me with instructive comments and corrections during the research and the manuscript composition.

Chapter 3 of this thesis has been published as: Z. Zhou and V. Dinavahi, "Parallel massive-thread electromagnetic transient simulation on GPU," *IEEE Trans. Power Del.,* vol. 29, no. 3, pp. 1045-1053, June 2014.

Chapter 4 of this thesis has been accepted for publication (PETSJ-00058-2016.R2): Z. Zhou and V. Dinavahi, "Fine-grained network decomposition for massively parallel electromagnetic transient simulation of large power systems," *IEEE Power and Energy Technology System Journal,* vol. 4, no. 3, pp. 51-64, Sept. 2017.

Chapter 5 of this thesis has been accepted for publication (TPEL-REG-2017-02-0298.R1): S. Yan, Z. Zhou and V. Dinavahi, "Large-scale nonlinear device-Level power electronic circuit simulation on massively parallel graphics processing architectures," *IEEE Transaction on Power Electronics,* pp. 1-19, July 2017.

Chapter 6 of this thesis has been published as: V. Jalili-Marandi, Z. Zhou and V. Dinavahi, "Large-scale transient stability simulation of electrical power systems on parallel GPUs," *IEEE Trans. Parallel and Distrib. Syst.,* vol. 23, no. 7, pp. 1255-1266, July 2012.

To my parents and my families,

for their support as always.

Acknowledgements

I would like to express my sincere thanks to my supervisor *Dr. Venkata Dinavahi* for his full support, encouragement, and guidance for the years throughout my research at the University of Alberta. His insightful guidance, passion and enthusiasm for the research has been an invaluable motivation for my life.

It is an honor for me to extend my gratitude to my Ph.D. committee members for reviewing my thesis and providing invaluable comments. Special thanks go to my colleagues and friends at the RTX-Lab: *Yuan Chen*, *Jiadai Liu* and *Shenhao Yan*.

Finally, financial help from NSERC, the University of Alberta, Government of the Province of Alberta for my living in Edmonton during these years is greatly appreciated.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ATP	Alternative Transients Program
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CS	Control System
CB	Control Block
CN	Connecting Network
DRAM	Dynamic Random-Access Memory
EMT	ElectroMagnetic Transients
EMTP	ElectroMagnetic Transients Program
FFT	Fast Fourier Transform
IGBT	Insulated Gate Bipolar Transistor
GPC	Graphic Processing Cluster
GPGPU	General Purpose Computing on GPU
GPU	Graphic Processing Unit
GUI	Graphical User Interface
HVDC	High-Voltage, Direct Current
LU	Lower Upper
LS	Linear Subsystem
LB	Linear Block
MMC	Modular Multilevel Converter
NLS	NonLinear Subsystem
NLB	NonLinear Block
OS	Operation System
PCIe	Peripheral Component Interconnect Express
SIMD	Single Instruction Multiple Data

SIMT	Single Instruction Multiple Thread
SM	Streaming Multiprocessor
ULM	Universal Line Module
UMM	Universal Machine Module

1 Introduction

Computer simulation plays an essential role in modern power system design and analysis. The electromagnetic transient program (EMTP) [1], which analyzes the temporary electromagnetic phenomena in both off-line and real-time [2], such as changes of voltage, current and flux in a short time slice induced by switching, surges, faults, lightning strike or any other disturbances in the network [3], moreover, is also indispensable for the planning, construction and operation of realistic electrical generation, transmission and distribution facilities in power systems, which is one of the most complicated large-scale network. Due to the versatility of EMT simulation tools, various power devices and components, including generators, transformers, transmission lines, loads, power electronic devices and control systems, are modeled using linear and nonlinear detailed models, which are calculated by discretized numerical methods such as Trapezoidal rule, and iterative solutions such as the Newton-Raphson method. Not only steady-state but also dynamic phenomena of the power system can be presented and studied with EMT simulation tools using the above models [4].

Along with the extending size of power system scale and the consequent increase in computational burden, the computing capability of CPU-based single thread EMT sequential program lags the requirement of computational power for large-scale power system simulation. Since the clock speed is reaching the ceiling and the chip power dissipation is constrained by fabrication, the computing ability of the single CPU core is closing to saturation. Therefore, an evolution of high performance computing is looming ahead, especially for the EMT simulation of power system.

1.1 EMTP on CPU

For the large-scale power system EMT simulation, off-line method is studied other than real-time mode in this work since the real-time simulation normally focuses on a specific study zone of the system; however, the off-line simulation will cover all buses and components over the system. Mainstream off-line EMT tools, such as ATP [5], PSCAD/EMTDC® [6], EMTP-RV® [7] and etc., developed over several decades, are routinely used in the power and energy industries, to study transient and dynamic phenomena over a wide frequency range [8]. These simulation tools comprise of extensive libraries of power system equipment models to accommodate most physical phenomena of practical significance.

The typical components modeled in EMT simulation are as follows,

- For 3-phase synchronous machines, universal machine model [9] can represent it with multiple windings on rotor d-axis and q-axis. The conventional mass-shaft system representation for mechanical part is equivalent to an electrical circuit modeling the mechanical part, which can be solved by the same discretization as electrical part.
- The transformer can be modeled by the admittance-based model [10] with nonlinear magnetic saturation representation by connecting an additional nonlinear inductance.
- The transmission line modeled by the universal line model [11] can be calculated in time-domain instead of phase-domain with frequency dependence. However, the numerical convolution has to be involved, which absorbs large amount of computational power.
- The linear passive loads, such as resistor, capacitor and inductor, can be modeled as a unified combination of a admittance and a history current source in parallel.
- The nonlinear relationship between valuables in the nonlinear component can be expressed as analytical equations or piecewise functions, which are solved by Newton-Raphson iterative method.
- The HVDC AC/DC converter, containing the modular multilevel converter (MMC) structure, is made of a series of submodules, which contains high frequency swit-

ching power electronic devices, such as IGBT and Diode, modeled using physics-based or behavior models.

- Since the control systems for normal power circuit and high speed power electronic devices are in different sample rates, interfaces are required to synchronize the computation.

Due to the detailed, high-order and nonlinear models applied, the performance of mainstream EMT tools is bound by the sequential programing based on the CPU, running on a closely saturated clock frequency, when the network size becomes large. Meanwhile, the sparsity of the system matrix is increasing rapidly along with the system scale, which burdens the computational efficiency seriously. The traditional way to handle this situation is to create a large-scale sparse system matrix describing the physical electrical network, and then use a mathematical method for its solution [12].

1.2 EMTP on GPU

The graphics processing unit (GPU) is originally designed to accelerate the computation for video and graphic processing which require a great amount of lighting, triangle and rendering calculation. Thus, it has a large number of processing units working similarly, which are independent of the CPU. Based on the massive cores architecture of GPU, it brings supercomputing for the masses, which has been described by many developers and researchers in various compute, analysis and simulation fields [13–15]. The application of high performance computing technology, such as multi-core CPU and many-core GPU, for solving large-scale power system problems, especially for dynamic and transient analysis, is on the rise.

There have already been contributions made in transient stability, dynamic state estimation, electromagnetic transient simulation, and power flow calculation fields [16–25]. For EMT simulation, while there have been efforts to develop massively parallel models for linear passive elements and frequency-dependent elements, a comprehensive treatment of transients in nonlinear elements using a fully iterative solution for large systems is a desired and yet to be achieved objective. As a compute-intensive job, the electromagnetic transient simulation in electrical system is also an excellent problem for GPU application [24], though there are only linear and basic models applied without detailed and frequency-dependent features included. Moreover, In order to increase the parallelism of

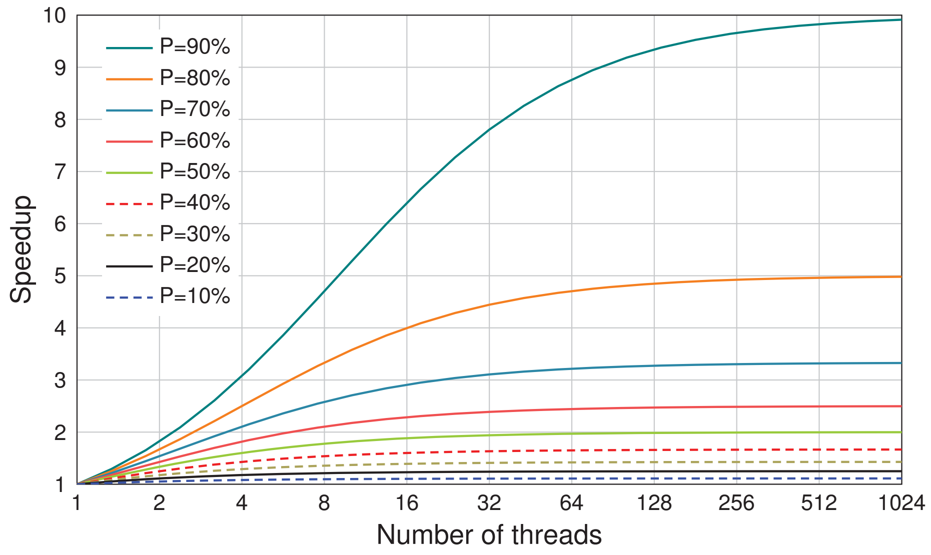


Figure 1.1: Amdahl's law.

the calculation, duplicated circuit structure and precalculated matrix inverse are applied, which seriously impacts the utilization of the designed GPU-based EMT simulator.

1.3 Parallel Performance

In order to approximate the performance in parallel computation, there are two basic criteria giving the theoretical speedup in latency of the execution. One is Amdahl's law and the other is Gustafson-Barsis's law, which address the problem in different scopes.

1.3.1 Amdahl's law

Amdahl's law, based on the assumption of a fixed workload, gives the formula in (1.1) as follows,

$$S = \frac{1}{(1-P)+P/N} \quad (P \in [0,1]), \tag{1.1}$$

where S is the execution speedup, P is the parallel portion of the whole computation, and N is the number of threads [26]. According to the above equation (1.1), the maximum performance that a parallel system can achieve is bound by the portion of parallel part in the whole task. No matter how many computing resources N are obtained, the max speedup S has a limitation respecting the parallel portion P. In the plotted Fig. 1.1, the speedup can never reach 2 when the portion rate is lower than 50%, even if the number of

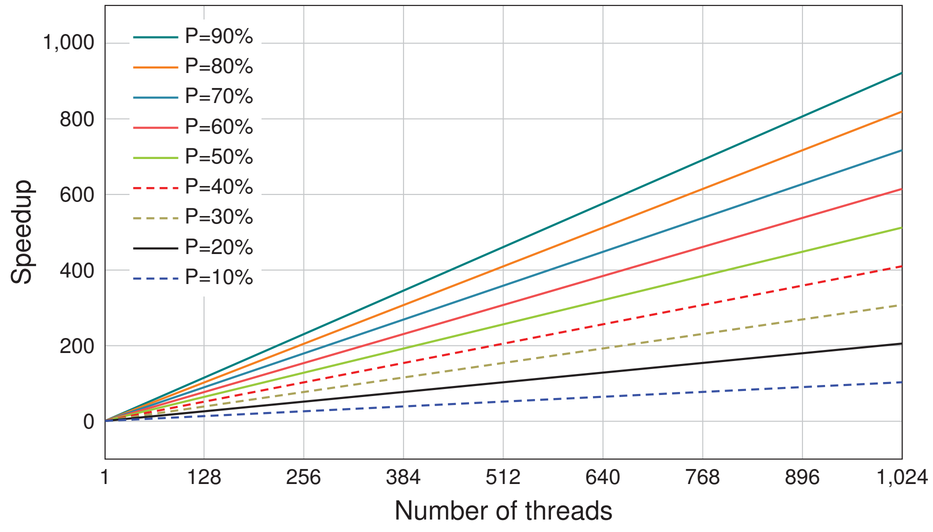


Figure 1.2: Gustafson-Barsis's law.

threads is given as many as 1024. Considering the unlimited resource,

$$N \to \infty, \qquad P/N \to 0, \tag{1.2}$$

the theoretical max performance is obtained as,

$$S_{max} = \frac{1}{1-P} \quad (P \in [0,1]), \tag{1.3}$$

which shows that the maximum speedup of a fixed size problem is decided by the parallelism of the problem itself instead of the hardware resources of parallel computing system. Therefore, increasing the parallel proportion is the key to optimize a fixed size problem in parallel computing.

1.3.2 Gustafson-Barsis's law

Different from Amdahl's law, in Gustafson-Barsis's law, the workload of problem is assumed to keep increasing to fulfill the existing computing resource, which is given as,

$$S = (1-P) + N * P \quad (P \in [0,1]), \tag{1.4}$$

where S is the execution speedup, P is the parallel portion of the whole computation, and N is the number of threads [27]. With above formula (1.4), the overall performance of the parallel computing system will continue climbing, only if enough computing resources are offered, whereas, the parallel proportion of the problem only influences the difficulty of reaching the higher speedup. For example, to obtain 200 times acceleration, it needs 256 threads when the portion rate is 80%, however, it requires 1000 threads when the portion rate is only 20%. Therefore, the performance enhancement of a parallel system is unlimited when the computation acquires enough computing resources for an open problem.

For large-scale parallel EMT simulation, the system parallelism can be improved by dividing the large system into smaller subsystems as possible, thus more computing resources can be involved in the computation, calculating all those subsystems simultaneously to increase the maximum possible speedup according to Amdahl's law. On the other hand, the optimization of parallelism has a limitation: in that case, the whole workload, that is the power system scale, can be increased to promote the utilization of computing resource following Gustafson-Barsis's law. Therefore, by complying with both laws, the performance of parallel EMT simulation will obtain the substantial growth via fine-grained system decomposition and system scale extension.

1.4 Motivation for this work

The single instruction multiple thread (SIMT), which is derived from single instruction multiple data (SIMD), provides the ability of branching, whereas the GPU has lightweight cores and coalescent memory. Therefore, the irregularity and unpredictability of the sparse construction seriously sabotages the SIMT execution and access efficiency of GPU cores compared with a multi-core CPU-based sparse algorithm. Transferring the physical circuit problem to the mathematical equations solved by a sparse algorithm is complicated to implement on GPU because of numerous random data access; and extends the number of iterations since Newton's iterative method is sensitive to initial values.

An alternative solution for SIMT execution is to partition the large system into similar small parts, which increases the rate of parallelism and the speed of convergence of solution for nonlinear systems using Newton iterations [28]. Exploiting the interface and boundary type sharing between subsystems, domain decomposition methods were proposed for parallel processing, such as Schur complement [29], additive Schwarz and multiplicative Schwarz method [30], which solve the system iteratively. Waveform relaxation methods, similarly, were developed for solving large sparse linear and nonlinear numerical systems iteratively [31]. In addition to the various pure mathematical methods, a large circuit can also be broken down into small parts based on its topology and physical characteristics. Diakoptics, introduced by G. Kron [32], showed that the method of system tearing can be a combination of equations and topology, and was further developed and systematized as a method to decompose electrical circuits [33,34]. Taking into account the interconnection among subsystems, the system matrix exhibits a special bordered block diagonal pattern after transformation and reorganization, which enables parallelism for the system solution.

This work proposes a fine-grained decomposition method for sparse linear and nonlinear networks for large-scale EMT simulation implemented on a GPU-based parallel computing system. Considering the EMT simulation with detailed, nonlinear component models and the particular features of GPU-based computing, the large-scale sparse linear system, (which requires numerous random access and lowers data processing density) and the wide ranging nonlinear solution, (which is slow to converge and expensive to synchronize among threads) are still the main challenges of this work. Therefore, the partitioning method employed is much fine-grained than usual and is named *shattering decomposition* in this work. The compensation network decomposition method, evolving from diakoptics, is utilized to solve the partitioned linear system in parallel without iteration . The Jacobian domain decomposition method is proposed to decouple the nonlinear system to be solved by Newton-Raphson method iteratively.

1.5 Research Objectives

The objectives to be accomplished in developing the GPU-based massively parallel program for EMT simulation are listed as follows:

- To develop the GPU-based parallel EMT algorithm for the synchronous machine based on the universal machine model. The mechanical part is equivalent to a electrical circuit, which can also be represented as EMT power components' models.
- To develop the massively parallel algorithm of admittance-based model on the GPU for transformer, and the magnetic saturation effect is represented by a nonlinear inductance.
- To develop the massively parallel algorithm of universal line model for transmission line with frequency dependence. The parallel numerical convocation is implement due to the time-domain modeling.
- To design the massively parallel unified linear passive element model on GPU for common linear power system components, such as resisters, capacitors, inductors and breakers.

- To partition the large power network by shattering decomposition, which includes two levels. The first level is coarse-grained and the second level is fine-grained.
- In the fine-grained level, to use compensation network decomposition for linear network and Jacobian domain decomposition for the Jacobian matrix in Newton-Raphson method.
- To design the massively parallel matrix decomposition algorithm for LU factorization.
- To design the massively parallel linear system solver based on LU factorization with forward-backward substitution for the decomposed power network.
- To design the massively parallel Newton-Raphson method for the solution of nonlinear components, such as lightening arrestor and nonlinear inductors.
- To design the decomposition scheme for the Jacobian matrix of MMC AC/DC converter based on Jacobian domain decomposition.
- To solve the MMC AC/DC converter based on the nonlinear device-level and linear behavior-based models in massively parallel, using proposed fine-grained decomposition for nonlinear and linear modeling respectively.
- The performance of the GPU-based massively parallel EMT simulation is evaluated for accuracy, computational efficiency and scalability, using a series of test power systems in different scales. The results and execution times are compared with the CPU-based EMT simulation using the same method and the mainstream commercial EMT simulation tools.

1.6 Thesis Outline

The thesis consists of six chapters. Other chapters are organized as follows:

- Chapter 2 introduces the general features of the GPU-Based computing system, including background, structure and technology; and the developing environment, CUDA architecture. This chapter also describes the important characteristics of electrical power systems which will influence the performance of parallel simulation.

- Chapter 3 describes the typical components modeling, such as synchronous machine, transformer, transmission line, loads and control system with linear and nonlinear features. Trapezoidal rule is applied to discretize the differential and integral equations; and then, the discretized equations are solved to find the network solutions.

- Chapter 4 gives the methods of shattering decomposition for linear and nonlinear systems, which consist two levels: the first level is coarse-grained one based on propagation delay; and the second level is fine-grained one based on circuit topology analysis.

- Chapter 5 demonstrates the application of the fine-grained decomposition on the MMC AC/DC converter, which is a strongly coupled power electronic network based on nonlinear and linear modeling.

- Chapter 6 explains the algorithms and implementations of the entire EMT simulation on a multi-GPU computing system.

- Chapter 7 includes four case studies, which show, compare and analyze the experimental results for various large-scale test systems.

- Chapter 8 gives the conclusions and future work.

2 Computing System and Electrical Network

Equipped with thousands of cores and possessing massive parallelism, the GPU has already shown its power in parallel computing because of its native massive processing cores, high memory bandwidth and outstanding floating point capability, so that it becomes the natural heart of massively parallel computing system. As the cradle of GPGPU, NVIDIA® developed generations of GPU architecture from Tesla, Fermi, Kepler, Maxwell to Pascal. Pascal is the most advantageous architecture released by NVIDIA® in 2016, which obtained far more improvement than its predecessor in performance and efficiency [35]. Meanwhile, the software developing environment, named the compute unified device architecture (CUDA™), was also evolved along with the hardware architecture to support the novel features, of which the compute capability indicating the version of supported features in parallel computation is updated to 6.x up to now [36]. In this work, Pascal architecture GPU with CUDA compute capability version 6.x is utilized to implement the massively parallel EMT simulation.

Different from the conventional CPU-based computing, GPU-based computing has some special requirements of the target, such as data flow, memory arrangement and task division, which associates with the natural characteristics of electrical power network. Taking advantage of these characteristics can effectively help deploy the massive parallelism and improve the efficiency.

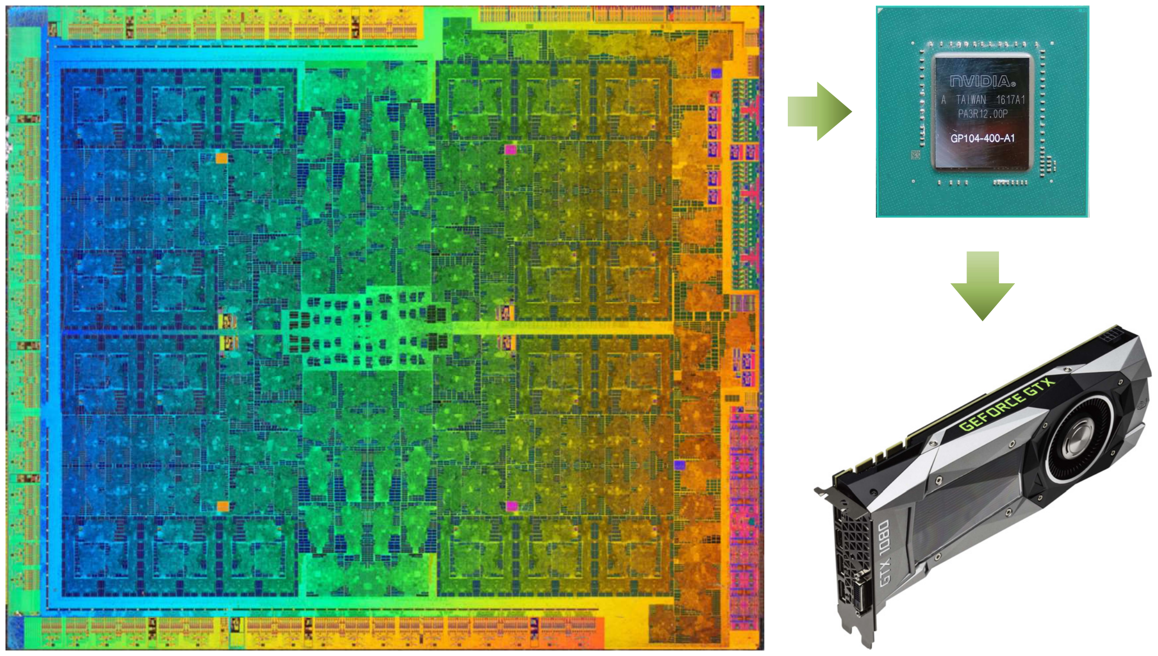


Figure 2.1: Die, chip and device for NVIDIA® GP104 GPU.

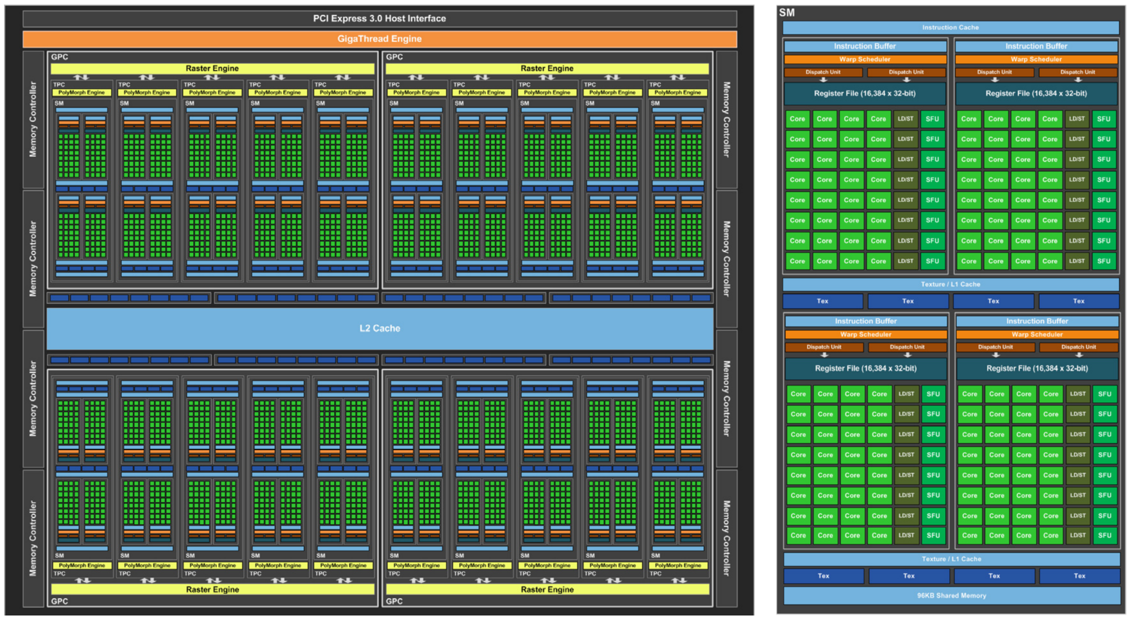
2.1 GPU-based computing system

2.1.1 GP104 GPU

The NVIDIA® GP104 GPU is based on Pascal architecture, which is comprised of 7.2 billion transistors on 314 mm^2 die using 16nm fin field-effect transistor (FinFET) manufacturing process that provides higher performance and improves power efficiency at the same.

Table 2.1: Specs comparison of NVIDIA's GPUs

GPU	Kepler GK104	Maxwell GM204	Pascal GP104
GPU clock (MHz)	1006	1226	1607
SMs	8	16	20
Cores	1536	2048	2560
Memory (GB)	4	4	8
Memory clock (GT/s)	6	7	10
Bandwidth (GB/s	192	224	320
GFLOPS	3090	4612	8228
Transistors (billion)	3.54	5.2	7.2
Die size (mm^2)	294	398	314
Fabrication (nm)	28	28	16
TDP (W)	195	165	180



(a) GPU (b) SM

Figure 2.2: Block diagram of the GP104 GPU [37].

Fig. 2.1 shows the die image, chip package and the card device of the GP104 GPU. It features 20 streaming multiprocessors (SMs) containing 2560 cores, 8 32-bit memory controllers (256-bit total) providing 320 GB/s memory bandwidth, and 8 GB memory. The GPU cores run at 1607 MHz, and the DDR5 memory offers 10 GT/s data transfer rate. The Table 2.1 provides the comparison of GP104 versus its predecessors, GM204 and GK104. The block diagram of the GP104 GPU is shown in Fig. 2.2(a). There are 4 graphics processing clusters (GPCs), and each GPC has 5 SMs. As shown in Fig. 2.2(b), each SM contains 128 cores, 256 KB register, 96 KB shared memory, 48 KB L1 cache, and 1 warp scheduler which manage the execution of 32 threads in group [37].

2.1.2 CUDA Abstraction

CUDA is a parallel computing software platform, besides C++ Accelerated Massive Parallelism (C++ AMP), open computing language (OpenCL$^{\text{TM}}$) and etc., introduced by NVIDIA$^{®}$ to access the parallel resources of the GPU. It offers application programming interfaces (APIs), libraries and compiler for software developers to use GPU for general purpose processing (GPGPU). With CUDA runtime, the GPU architectures are abstracted into CUDA specs, which decide how to map the parallel requests to hardware entities when the GPGPU program is being developed. It provides a unified interface for CUDA supported GPU according to the compute capability version regardless of the different details

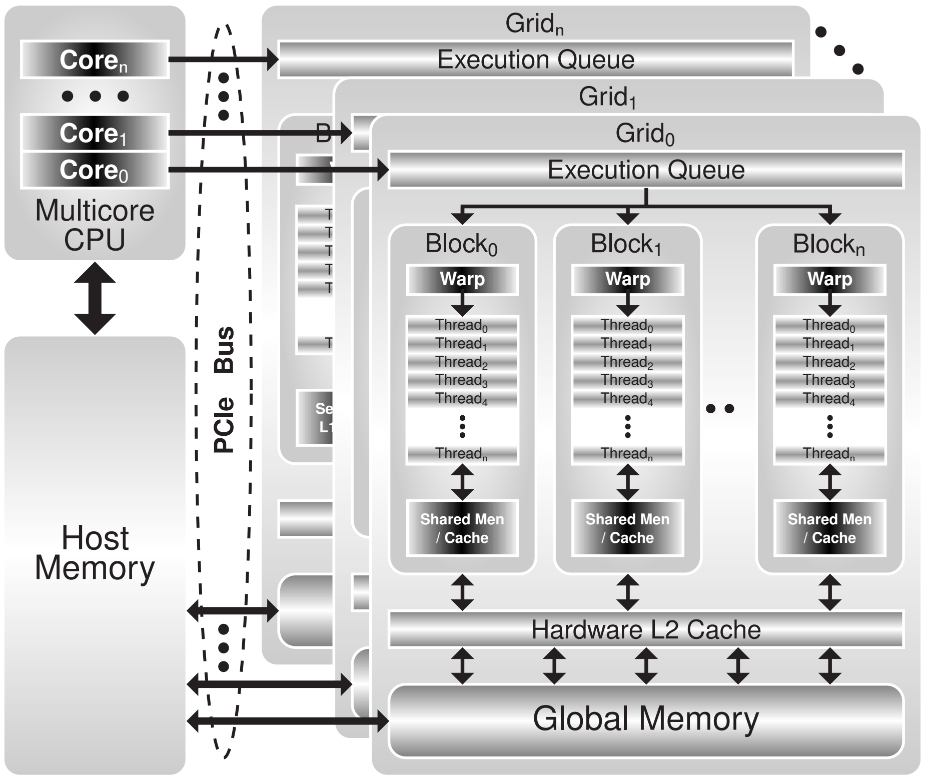


Figure 2.3: CUDA abstraction of device (GPU) and host (CPU)

of each generation device. Thus, the programmer can only focus on their algorithm design and need not concern themselves about the GPU hardware too much. Based on the CUDA attraction, the whole parallel computing platform including CUP and GPU is described as a host-device system, as shown in Fig. 2.3. When developing parallel application with

Table 2.2: Specs of CUDA Compute Capability 6.1

Device	**GeForce GTX 1080**
Total amount of global memory	8192 MB
CUDA Cores	2560
Total amount of constant memory	64 KB
Total amount of shared memory per block	48 KB
Total number of registers per block	64 K
Warp size	32
Maximum number of threads per block	1024
Max dimension size of a thread block (x,y,z)	(1024, 1024, 64)
Max dimension size of a grid size (x,y,z)	(2 G, 64 K, 64 K)
Concurrent copy and kernel execution	Yes, with 2 copy engines

Table 2.3: Memory bandwidth

Type	Bandwidth
Host to Device	6GB/s[*]
Global Memory	226GB/s
Shared Memory	2.6TB/s
Cores	5TB/s

CUDA, the programmer follows the standard of the CUDA capability given by the NVIDIA driver, such as 6.1 used in this work, which defines thread hierarchy and memory organization, as listed in Table 2.2. Since each generation GPU hardware is binded with specific CUDA version, the configuration of parallel computing resource including threads and memory running on GPU is based on the CUDA capability version.

Different from the heavyweight cores in multi-core CPU whose threads are almost independent workers, the threads in SIMT GPU are numerous but lightweight. Thus, the performance of GPU-based computation depends to a great extent on the workload distribution and resource utilization. The C-extended functions in CUDA, called *kernel*, runs on GPU, *device* side, in parallel by different *threads* controlled by CPU, *host* side [36]. All *threads* are grouped into *blocks* and then *grids*. Each GPU device presents itself as a *grid*, in which there are up to 32 active *blocks* [35]. The *threads* in a block are grouped by warps. There are up to 4 active warps per block. Although a block maximally supports 1024 threads, only up to 32 threads in one warp can run simultaneously. The initial data in *device* are copied from *host* through PCIe bus, and the results also have to be transfered back to host via PCIe bus again, which causes serial delay.

There are 3 major types of memory in CUDA abstraction: global memory, which is large and can be accessed by both *host* and *device*; shared memory, which is small, can be accessed by all *threads* in a *block*, and is even faster than global one; registers, which is limited, can only be accessed by each *thread*, and is the fastest one. Although the global memories have high bandwidth, the data exchange channel, PCIe bus, between *host* and *device* is slow; thus avoiding those transfers unless they are absolutely necessary is vital for computational efficiency. Table 2.3 lists the typical bandwidth of major memory types in CUDA.

Besides that the compiler is extended to the industry-standard programming languages including C, C++ and Fortran for general programmers, CUDA platform offers the

[*]The PCIe interface is reduced to $\times 8$ instead of $\times 16$ due to multiple PCIE devices

interfaces to other computing platforms, including OpenCL, DirectCompute OpenGL and C++ AMP. In addition, CUDA is supported by various languages, such as Python, Perl, Java, Ruby, MATLAB and etc., as a third part plug-in. CUDA toolkits also comes with the following libraries as listed in Table 2.4. Developers can choose some of libraries on-demand to simplify their programming.

Table 2.4: CUDA Libraries

Library	Description
CUBLAS	CUDA Basic Linear Algebra Subroutines
CUDART	CUDA RunTime
CUFFT	CUDA Fast Fourier Transform library
CUSOLVER	CUDA based collection of dense and sparse direct solvers
CUSPARSE	CUDA Sparse Matrix

CUDA C/C++, which is used in this work, extends C by define a C-like function called kernel to invoke parallel execution on the GPU by deploying the execution configuration for dimensions of thread, block and grid before the kernel is called. The configuration information can be retrieved inside the function by the Built-in Variables, including **gridDim**, **blockIdx**, **blockDim** and **threadIdx**.

There are different types of functions classified by type qualifiers in CUDA, such as __device__, __global__ and __host__.

The __device__ qualifier declared function is

- executed on the device,
- callable from the device only.

The __global__ qualifier declared function is

- executed on the device,
- callable from the host or device,
- eeturned void type,
- specified execution configuration,
- asynchronous execution.

The __host__ qualifier declared function is

- executed on the host,
- callable from the host only.

According above criterion, a __global__ function cannot be __host__.

Similarly, the variables are also classified by type qualifiers in CUDA, such as __device__, __constant__ and __shared__.

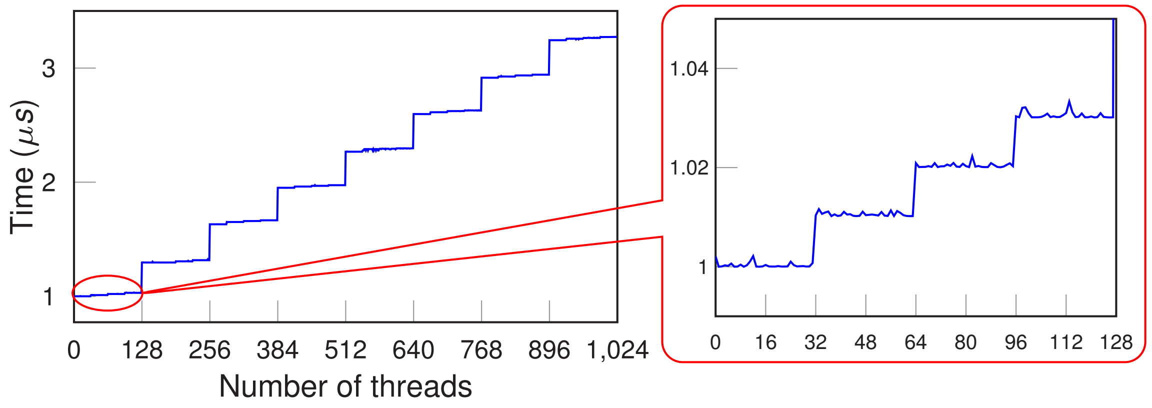


Figure 2.4: CUDA compute performance related to the number of threads in one CUDA Block.

The __device__ qualifier declared valuable is

- located in global memory on the device,
- accessible from all the threads within the grid.

The __constant__ qualifier declared function is

- located in constant memory space on the device,
- accessible from all the threads within the grid.

The __shared__ qualifier declared function is

- located in the shared memory space of a thread block,
- accessible from all the threads within the block.

__device__ and __constant__ valuables have the lifetime of an application, while __shared__ valuable has the lifetime of the block. [36]

2.1.3 Performance Tuning

As shown in Fig. 2.4, the first inner level step characteristic (zoomed-in balloon) shows 32 threads working in parallel, while the second outer level step characteristic shows 4 active warps in parallel to make up the total 128 executing threads. Therefore, lowering occupation in one block as well as raising some number of blocks with the same total number of threads is an optimal way to increase efficiency. In each block, there is 48KB of *shared memory* which is roughly 10x faster and has 100x lower latency than uncached *global memory*, whereas each thread has up to 255 registers running at the same speed as the cores. The overwhelming performance improvement was shown with avoiding and optimizing communication for parallel numerical linear algebra algorithms in various supercomputing platforms including GPU [38]. Making a full play of this critical resource

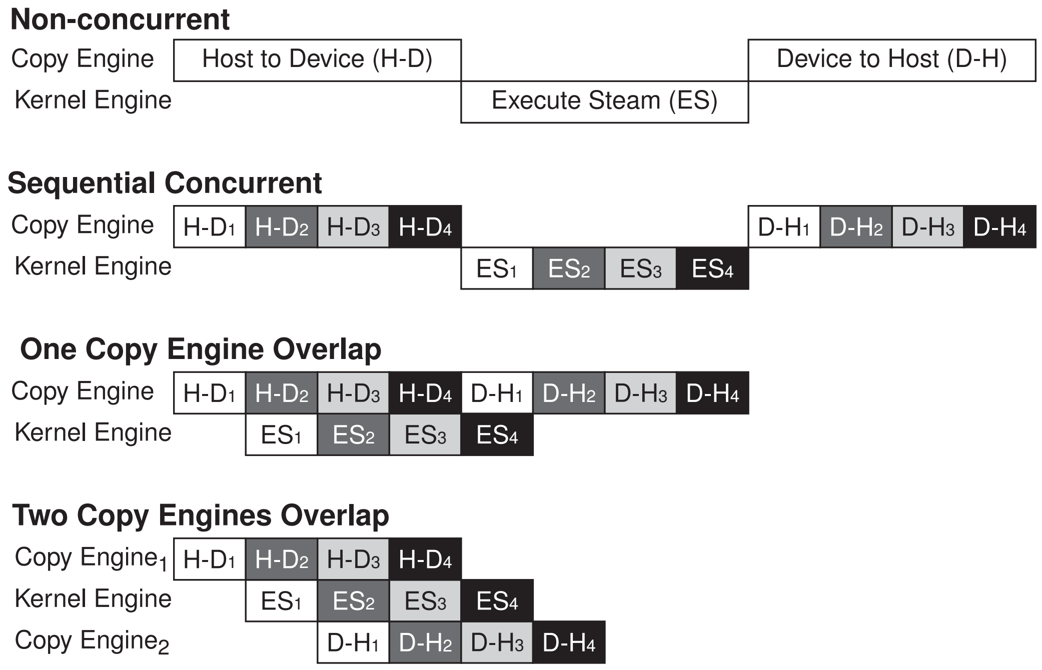


Figure 2.5: Concurrent execution overlap for data transfer delay.

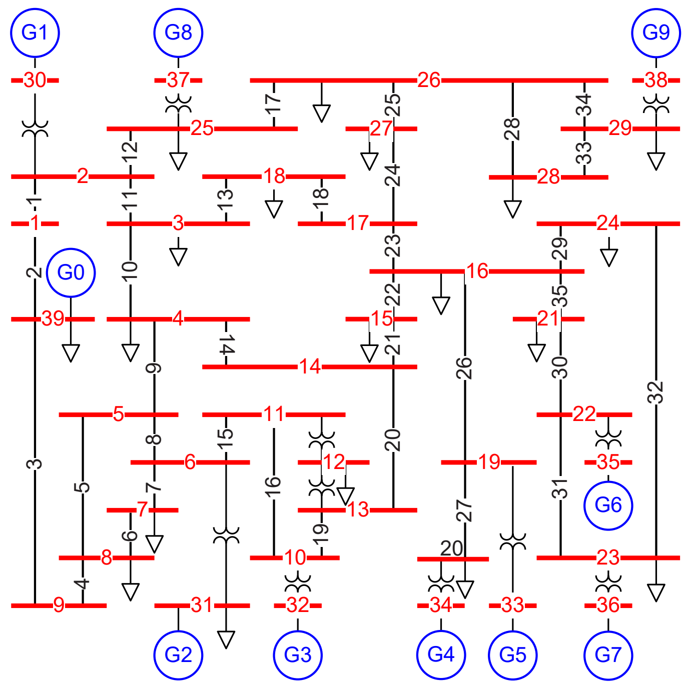
can significantly increase the efficiency of computation, which requires the user to scale the problem perfectly and unroll the *for* loops in a particular way [39].

In addition to one task being accelerated by many threads, concurrent execution is also supported on GPU. As shown in Fig. 2.5, the typical non-concurrent kernel is in 3 steps with one copy engine GPU:

- Copy data from host to device first by *Copy Engine*;
- Execute in the default Stream by *Kernel Engine*;
- Copy results back to host from device by *Copy Engine*.

The calculation can also be finished by multiple streams. In sequential concurrent, the performance is the same as Non-concurrent; however, different streams can run in overlap, thus, the calculation time can be completely hidden with one copy engine. Furthermore, the maximum performance can be reached using the hardware with two copy engines, where most of the time of data transfer is covered, and the device memory limitation is effectively relieved since the runtime device memory usage is divided by multiple streams. According to CUDA compute capability specs, version 6.1 supports concurrent copy and kernel execution with 2 copy engines.

2.2 Electrical Power Network


(a) 39-bus network

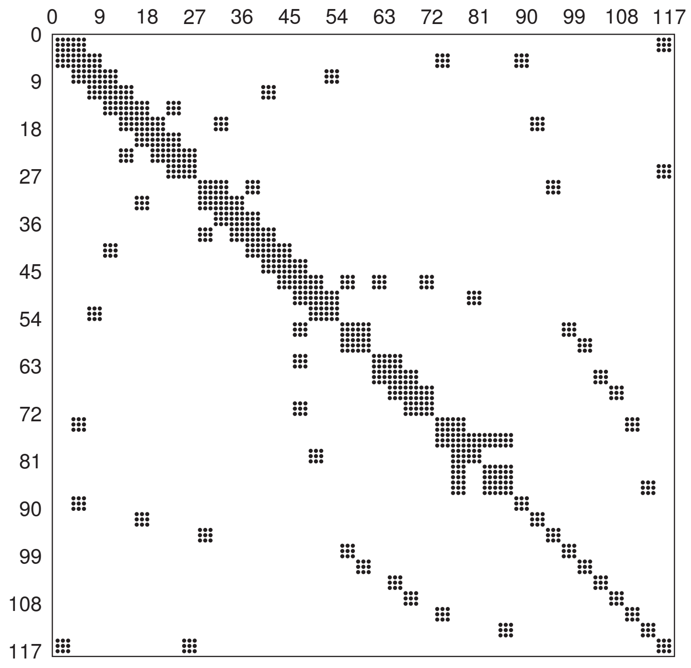

(b) Admittance matrix ($\boldsymbol{Y}$)

Figure 2.6: Sparsity of IEEE 39-bus power system.

Similar to most electrical circuits, the electrical power network transmitting energy from end to end is a sparse system, where each element (component) only links with few other elements (component) nearby. Fig. 2.6 shows the sparsity pattern of IEEE 39-bus power system, where each dot represents link between two nodes. There are 117 nodes in total since all 39 buses are 3-phase. In order to avoid dealing with this sparsity during the computation, especially when the scale of network is considerably large, the fine-grained decomposition is applied during the simulation. Although the ideal target is to tear the system into component level pieces, such as bus-by-bus, for EMT simulation such a partition scheme would cause extra computing effort normally, such as data communication and connection networks. The simulation performance relating to the scale of the subsystem has a step effect due to the warp execution of CUDA, which means it has the same performance within every 32-thread (1-warp) enlargement, and similar performance within every 4-warp increasing step. Therefore, sparsity can be ignored in each warp, and the scale of the subsystems can be manipulated to meet the maximum size of the warp.

Thinking of partitioning and reorganizing the power network, there are two types of components are considered for decoupling the network. One type has non-negligible single propagation delay from one node to other node like transmission line. When the delay is larger than the EMT simulation time-step, the subnetworks linked by this type of component are natively decoupled for time discretized numerical method. The other type is at the border of subnetworks since a large power network is comprised by connecting a series of small subnetworks. If the calculation of this type of border components can be separated from the overall network solution, the solutions of subnetworks are decoupled in numerical computation and can run in parallel. The fine-grained decomposition to handle this problem is detailed in Chapter 4.

2.3 Summary

In this chapter, the main features about the computing system and electrical power network relating to the massively parallel EMT simulation are introduced. The Pascal architecture GPU, GP104, ships 7.2 billion transistors with 16 nm fabrication to offer 2560 cores grouped into 20 SMs running under 1607 MHz and carrying 8 GB memory, which shows much more computational power than its predecessors. Meanwhile, the CUDA compute capability version 6.1 demonstrates a substantial and consolidated spec for GPU-based massively parallel EMT simulation. The important characteristics of the computing system, including massive cores, SIMT execution, memory bandwidth, CUDA abstract and concurrent engines; and features of electrical power network, such as sparse structure and interconnection topology, will be considered and utilized to implement and optimize the GPU-based massively parallel EMT simulation.

3 Electromagnetic Transient Modeling

The proposed GPU-based EMT massively parallel simulation includes typical electrical power devices and components to build up a realistic-size power system. In order to present the simulation performance on GPU-based massively parallelism, they are modeled in detailed, frequency-dependent or lump with linear and nonlinear features. Because power electronic devices contain high-frequency switching characteristics, they are discussed separately in Charter 5.

In modern electrical networks, the classical components include synchronous machines with control systems, transformers, transmission lines, and linear and nonlinear passive elements. Although the purpose of this work is to show the compute acceleration of fine-grained parallel EMT nonlinear simulation, detailed models are used to realize the computing power of the GPU. The basic theory of the electromagnetic transient program is to discretize the differential and integral equations in electrical circuits by Trapezoidal rule; and then to solve them repeatedly to find the numerical time-domain solutions, such as voltages and currents [8].

3.1 Synchronous Machine with Control System

The universal machine model provides a unified mathematical framework to represent various types of rotating machines including synchronous, asynchronous and DC machine [9]. As shown in Fig. 3.1, the electrical part of the synchronous machine includes 3 stator armature windings $\{a, b, c\}$; one field winding f and up to 2 damper windings $\{D_1, D_2\}$ on

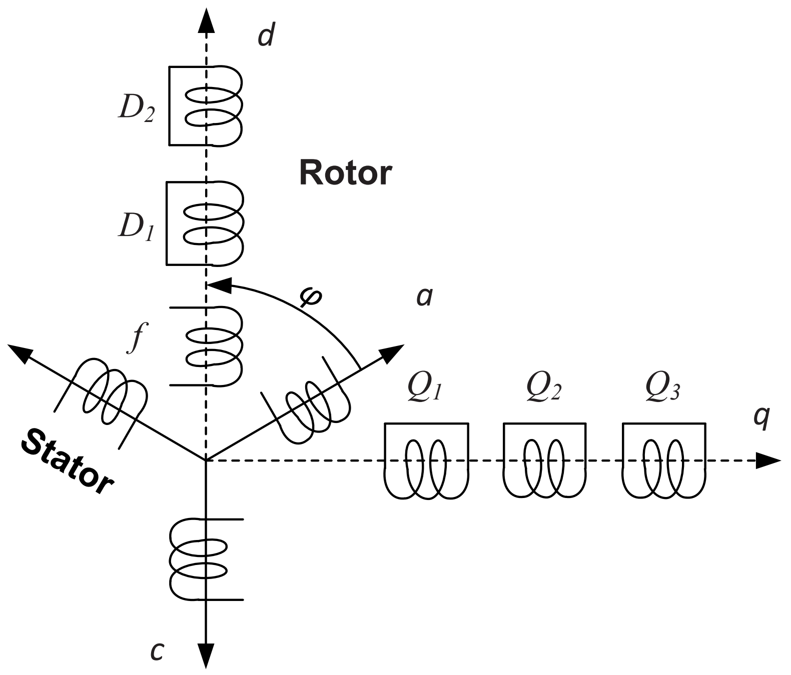


Figure 3.1: Electrical side model of synchronous machine.

the rotor direct d-axis; and up to 3 damper windings $\{Q_1, Q_2, Q_3\}$ on the rotor quadrature q-axis. The discretized winding equations after *dq0* conversion are described as

$$\boldsymbol{v}_{dq0}(t) = -\boldsymbol{R}\boldsymbol{i}_{dq0}(t) - \frac{2}{\Delta t}\boldsymbol{\lambda}_{dq0}(t) + \boldsymbol{u}(t) + \boldsymbol{V}_h, \tag{3.1}$$

where $\boldsymbol{R}$ is the winding resistance, $\boldsymbol{\lambda}_{dq0}$ are the flux linkages, $\boldsymbol{u}$ are speed voltages and Δt is the simulation time-step. where $\boldsymbol{R}$ is the winding resistance matrix, and the flux linkage $\boldsymbol{\lambda}_{dq0}$ is given as

$$\boldsymbol{\lambda}_{dq0}(t) = \boldsymbol{L}\boldsymbol{i}_{dq0}(t), \tag{3.2}$$

where $\boldsymbol{L}$ is the winding leakage inductance matrix given as

$$\boldsymbol{L} = \begin{bmatrix} L_d & 0 & 0 & M_{df} & M_{dD_1} & M_{dD_2} & 0 & 0 & 0 \\ 0 & L_q & 0 & 0 & 0 & 0 & M_{qQ_1} & M_{qQ_2} & M_{qQ_3} \\ 0 & 0 & L_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ M_{df} & 0 & 0 & L_f & M_{fD_1} & M_{fD_2} & 0 & 0 & 0 \\ M_{dD_1} & 0 & 0 & M_{fD_1} & L_{D_1} & M_{D_1D_2} & 0 & 0 & 0 \\ M_{dD_2} & 0 & 0 & M_{fD_2} & M_{D_1D_2} & L_{D_2} & 0 & 0 & 0 \\ 0 & M_{qQ_1} & 0 & 0 & 0 & 0 & L_{Q_1} & M_{Q_1Q_2} & M_{Q_1Q_3} \\ 0 & M_{qQ_2} & 0 & 0 & 0 & 0 & M_{Q_1Q_2} & L_{Q_2} & M_{Q_2Q_3} \\ 0 & M_{qQ_3} & 0 & 0 & 0 & 0 & M_{Q_1Q_3} & M_{Q_2Q_3} & L_{Q_3} \end{bmatrix} \tag{3.3}$$

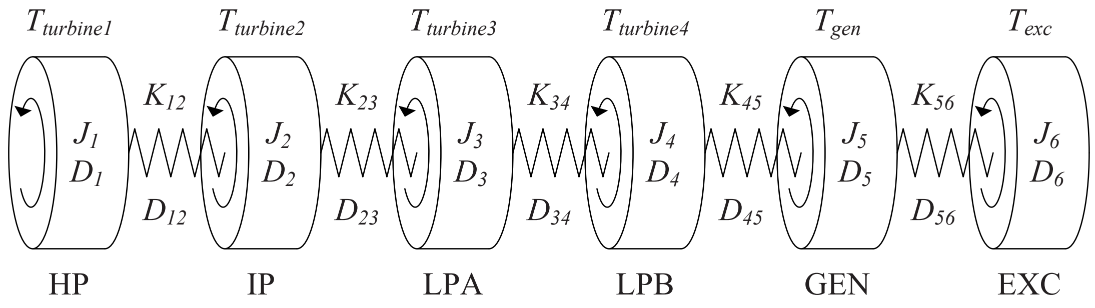


Figure 3.2: Mechanical side model of synchronous machine.

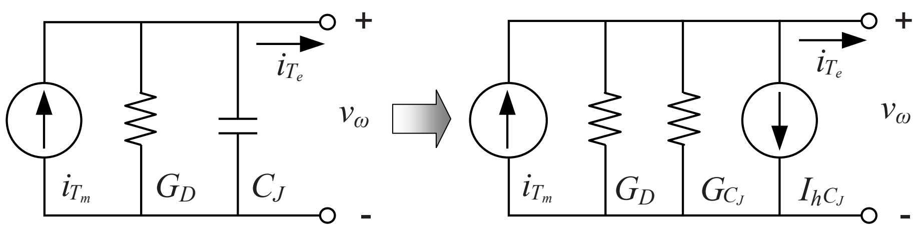


Figure 3.3: Electrical equivalent of mechanical side model.

with L and M standing for the self and mutual inductances respectively. In (3.1), the vectors of voltages $\boldsymbol{v}_{dq0}$, currents $\boldsymbol{i}_{dq0}$ and speed voltages $\boldsymbol{u}$ of the windings are expressed as

$$\begin{aligned}
\boldsymbol{v}_{dq0} &= [\ v_d\ ,\ v_q\ ,\ v_0,\ v_f,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ],\\
\boldsymbol{i}_{dq0} &= [\ i_d\ ,\ i_q\ ,\ i_0,\ i_f,\ i_{D_1},\ i_{D_2},\ i_{Q_1},\ i_{Q_2},\ i_{Q_3}],\\
\boldsymbol{u} &= [\ -\omega\lambda_q,\ \omega\lambda_d,\ 0,\ 0,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ];
\end{aligned}$$

the winding resistance matrix $\boldsymbol{R}$ is a diagonal matrix, given as

$$\boldsymbol{R} = diag[R_d, R_q, R_0, R_f, R_{D_1}, R_{D_2}, R_{Q_1}, R_{Q_2}, R_{Q_3}];$$

and the history voltages $\boldsymbol{V}_h$ in (3.1) are given as,

$$\boldsymbol{V}_h(t-\Delta t) = -\boldsymbol{v}_{dq0}(t-\Delta t) - \boldsymbol{R}\boldsymbol{i}_{dq0}(t-\Delta t) + \frac{2}{\Delta t}\boldsymbol{\lambda}_{dq0}(t-\Delta t) + \boldsymbol{u}(t-\Delta t). \tag{3.4}$$

The mass-shaft mechanical side model shown in Fig. 3.2 is represented as a linear electrical equivalent, as shown in Fig. 3.3, thus the speed-torque equation (3.5) is updated to voltage-current equation (3.6)

$$J\frac{d\omega}{dt} + D\omega + K\int \omega dt = T_{turbine} - T_{gen/exc}, \tag{3.5}$$

$$i_{T_m} = C_J\frac{\mathrm{d}v_\omega}{\mathrm{d}t} + G_D v_\omega + i_{T_e}, \tag{3.6}$$

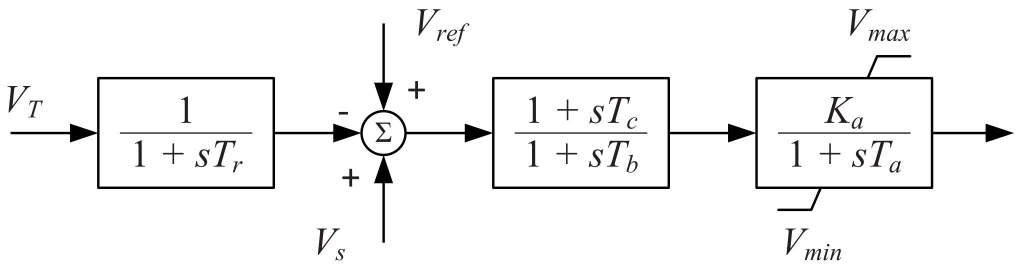


Figure 3.4: Excitation control system.

where i_{T_m}, C_J, G_D and v_ω represent the equivalent mechanical torque, inertia, damping and rotor speed respectively [40]. The equivalent electromagnetic torque i_{T_e}, which interfaces to the electrical part, can be represented by the flux linkages $\boldsymbol{\lambda}_{dq0}$ and the machine currents $\boldsymbol{i}_{dq0}$ as,

$$i_{T_e} = \lambda_d i_q - \lambda_q i_d. \tag{3.7}$$

Then the time discretization for linear passive elements is applied to (3.6), to obtain:

$$v_\omega(t) = \frac{i_{T_m}(t) - i_{T_e}(t) - I_{hC_J}(t-\Delta t)}{G_D + G_{C_J}}, \tag{3.8}$$

which merges the solution of mechanical part to that of the electrical part. The equivalent conductance G_{C_J} is given as,

$$G_{C_J} = 2C_J/\Delta t. \tag{3.9}$$

and the history term I_{hC_J} is given as,

$$I_{hC_J}(t-\Delta t) = -I_{hC_J}(t-2\Delta t) - 2G_{C_J}v_\omega(t-\Delta t). \tag{3.10}$$

3.2 Control System Interface

Since the model of excitation control system, shown in Fig. 3.4, is different from that of the electrical network, whose equations are difficult to be merged into the nodal analysis equations in EMT simulation, they will be solved separately. On the other hand, the sample time (T_s), used in control system is much larger than the time step:

$$T_s \gg \Delta t. \tag{3.11}$$

Therefore, there is an interface between the two subsystems, as shown in Fig. 3.5. After the machine network is solved, the solutions including voltages and currents are sent to

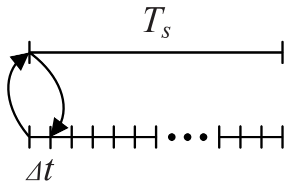


Figure 3.5: Excitation control system.

the excitation system, and then its solutions are calculated, which are in turn sent back to the former in the next time-step of EMT computation. Since the excitation system appears as almost static to the EMT network, the error introduced by the time delay inserted is usually very small.

3.3 Transformer with Magnetic Saturation

The linear part of the transformer is modeled by admittance based model [10]. The n-winding transformer for EMT simulation with winding resistance $\boldsymbol{R}$ and leakage inductance $\boldsymbol{L}$ is represented as,

$$\boldsymbol{v} = \boldsymbol{R}\boldsymbol{i} + \boldsymbol{L}\frac{\mathrm{d}\boldsymbol{i}}{\mathrm{d}t}, \tag{3.12}$$

where $\boldsymbol{R}$ and $\boldsymbol{L}$ are $n \times n$ matrices, given as

$$\boldsymbol{R} = diag\{R_1 \quad R_2 \quad \dots \quad R_n\}, \tag{3.13}$$

$$\boldsymbol{L} = \begin{bmatrix} L_{11} & L_{12} & \dots & L_{1n} \\ L_{21} & L_{22} & \dots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \dots & L_{nn} \end{bmatrix}, \tag{3.14}$$

which are obtained from open and short circuit tests. In case of the three phase 2-winding transformer, $\boldsymbol{R}$ and $\boldsymbol{L}$ are given as

$$\boldsymbol{R}_{12}^{abc} = \begin{bmatrix} R_1^a & & & & & \\ & R_1^b & & & & \\ & & R_1^c & & & \\ & & & R_2^a & & \\ & & & & R_2^b & \\ & & & & & R_2^c \end{bmatrix}, \boldsymbol{L}_{12}^{abc} = \begin{bmatrix} L_{11}^{aa} & L_{11}^{ab} & L_{11}^{ac} & L_{12}^{aa} & & \\ L_{11}^{ab} & L_{11}^{bb} & L_{11}^{bc} & & L_{12}^{bb} & \\ L_{11}^{ac} & L_{11}^{bc} & L_{11}^{cc} & & & L_{12}^{cc} \\ L_{21}^{aa} & & & L_{22}^{aa} & L_{22}^{ab} & L_{22}^{ac} \\ & L_{21}^{bb} & & L_{22}^{ab} & L_{22}^{bb} & L_{22}^{bc} \\ & & L_{21}^{cc} & L_{22}^{ac} & L_{22}^{bc} & L_{22}^{cc} \end{bmatrix}. \tag{3.15}$$

Applying trapezoidal rule, (3.12) is rewritten as,

$$\boldsymbol{i}(t) = \boldsymbol{G}_{eq}\boldsymbol{v}(t) + \boldsymbol{I}_h(t - \Delta t), \tag{3.16}$$

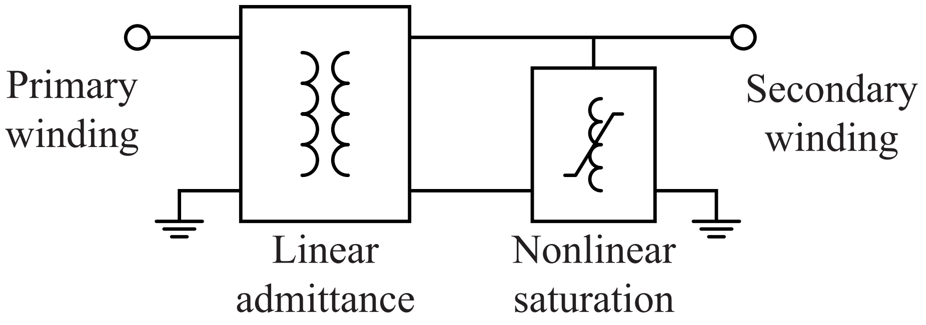


Figure 3.6: Admittance based transformer model with saturation.

where the equivalent admittance is given as,

$$\boldsymbol{G}_{eq} = \frac{\Delta t}{2}[\mathbf{I} + \frac{\Delta t}{2}\boldsymbol{L}^{-1}\boldsymbol{R}]^{-1}\boldsymbol{L}^{-1}. \tag{3.17}$$

and the history currents are expressed as,

$$\begin{aligned}\boldsymbol{I}_h(t-\Delta t) &= \boldsymbol{G}_{eq}\boldsymbol{v}(t-\Delta t)\\ &+ [\mathbf{I} + \frac{\Delta t}{2}\boldsymbol{L}^{-1}\boldsymbol{R}]^{-1}[\mathbf{I} - \frac{\Delta t}{2}\boldsymbol{L}^{-1}\boldsymbol{R}]\boldsymbol{i}(t-\Delta t)\\ &= \boldsymbol{G}_{eq}\boldsymbol{v}(t-\Delta t) + (\mathbf{I} - 2\boldsymbol{G}_{eq}\boldsymbol{R})\boldsymbol{i}(t-\Delta t),\end{aligned} \tag{3.18}$$

where $\mathbf{I}$ is the identity matrix. Expressed in term of $\boldsymbol{v}(t-\Delta t)$ and $\boldsymbol{I}_h(t-2\Delta t)$, (3.18) is updated as,

$$\begin{aligned}\boldsymbol{I}_h(t-\Delta t) &= 2(\boldsymbol{G}_{eq} - \boldsymbol{G}_{eq}\boldsymbol{R}\boldsymbol{G}_{eq})\boldsymbol{v}(t-\Delta t)\\ &+ (\mathbf{I} - 2\boldsymbol{G}_{eq}\boldsymbol{R})\boldsymbol{I}_h(t-2\Delta t).\end{aligned} \tag{3.19}$$

The saturation effect is represented by the extra nonlinear inductance connected to the secondary winding, which is joined to the linear inductance matrix, as shown in Fig. 3.6. The nonlinear inductance taking care of the saturation effect contains the nonlinear relation between flux λ and branch current i, given as,

$$g(\lambda, i) = 0. \tag{3.20}$$

On the other hand, the flux λ is the integral of node voltage v over a time-step,

$$\lambda(t) = \lambda(t-\Delta t) + \int_{t-\Delta t}^{t} v(u)\mathrm{d}u. \tag{3.21}$$

Discretizing by Trapezoidal rule, we get,

$$\lambda(t) = \frac{\Delta t}{2}v(t) + \Lambda_h, \tag{3.22}$$

where the history term Λ_h is defined as,

$$\Lambda_h(t-\Delta t) = \lambda((t-\Delta t) + \frac{\Delta t}{2}v(t-\Delta t). \tag{3.23}$$

Thus, the relation found by substituting λ in (3.20) with (3.22) between voltage and current of the nonlinear inductor is expressed as,

$$g(\frac{\Delta t}{2}v(t) + \Lambda_h, i) = 0. \tag{3.24}$$

Since the nonlinear inductance is also connected to the linear network, the node equation is given as,

$$G_N v = i_0 - i, \tag{3.25}$$

where G_N is the Norton admittance of the node which includes all admittance connects the node, and i_0 is the sum of all currents, except the current from the nonlinear component, injecting to the node. i_0 can be know or unknown, if there are multiple nonlinear components connecting to the node, the number of unknown valuables and nonlinear equations are added at the same time. Substituting i in (3.24) with

$$i = G_N v - i_0, \tag{3.26}$$

from (3.25), the nonlinear function for Newton-Raphson method is obtained as,

$$F \equiv g(\frac{\Delta t}{2}v(t) + \Lambda_h, G_N v - i_0) = 0, \tag{3.27}$$

And the Jacobian value is the partial derivative of F respect to v, as

$$J = \frac{\partial F}{\partial v} = \frac{\partial g(\frac{\Delta t}{2}v(t) + \Lambda_h, G_N v - i_0)}{\partial v}. \tag{3.28}$$

With F and J, the iterative method is applied to find the node voltage when the algorithm reaches converge.

3.4 Transmission Line

The universal line model (ULM) [11] is used for one of the most important components in electrical circuit, transmission line, which can represent both symmetrical and asymmetrical overhead lines and cables since it is constituted in the phase-domain directly. As shown in Fig. 3.7, the long line is modeled by two separate circuits with admittance $\boldsymbol{G}$ and history

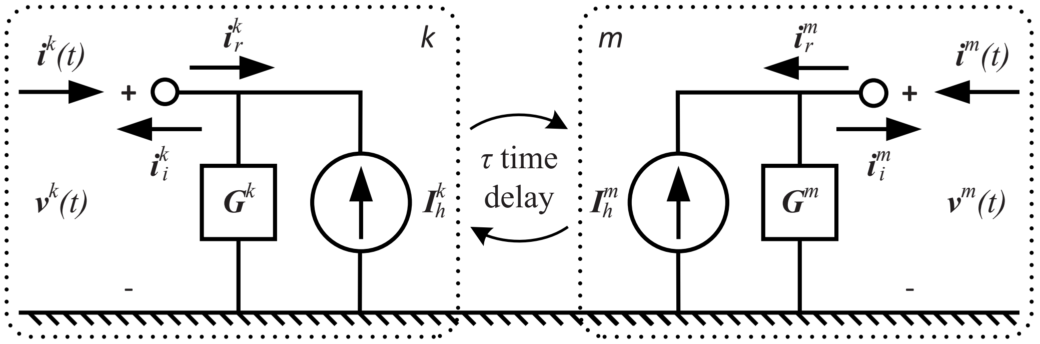


Figure 3.7: Frequency-dependent transmission line model.

current source $\boldsymbol{I}_h$, sending end 'k' and receiving end 'm', linked by traveling waves. The KCL equation is expressed as,

$$\boldsymbol{i}_k(t) = \boldsymbol{G}_Y \boldsymbol{v}_k(t) - \boldsymbol{I}_{hk}, \tag{3.29a}$$

$$\boldsymbol{i}_m(t) = \boldsymbol{G}_Y \boldsymbol{v}_m(t) - \boldsymbol{I}_{hm}, \tag{3.29b}$$

where the history currents $\boldsymbol{I}_h$ at the two ends of the line are expressed as

$$\boldsymbol{I}_{hk} = \boldsymbol{Y} * \boldsymbol{v}_k(t) - 2\boldsymbol{i}_{ik}, \tag{3.30a}$$

$$\boldsymbol{I}_{hm} = \boldsymbol{Y} * \boldsymbol{v}_m(t) - 2\boldsymbol{i}_{im}, \tag{3.30b}$$

and the incident currents $\boldsymbol{i}_i$ can be expressed by remote reflected currents $\boldsymbol{i}_r$ as,

$$\boldsymbol{i}_{ik} = \boldsymbol{H} * \boldsymbol{i}_{rm}(t-\tau), \tag{3.31a}$$

$$\boldsymbol{i}_{im} = \boldsymbol{H} * \boldsymbol{i}_{rk}(t-\tau), \tag{3.31b}$$

where $\boldsymbol{Y}$ and $\boldsymbol{H}$ are the characteristic admittance and the propagation matrices respectively, which are approximated by finite-order rational functions using the vector fitting (VF) method [41]. In the phase domain, the general entry of $\boldsymbol{Y}$ is given as,

$$\boldsymbol{Y}_{(i,j)}(s) = \sum_{m=1}^{N_p} \frac{\boldsymbol{r}_{Y(i,j)}(m)}{s - \boldsymbol{p}_Y(m)} + \boldsymbol{d}(i,j), \tag{3.32}$$

where $\boldsymbol{r}_Y$, $\boldsymbol{p}_Y$, $\boldsymbol{d}$ and N_p are the residues, poles, proportional terms, and the number of poles of $\boldsymbol{Y}$ respectively. Similarly, the general entry of $\boldsymbol{H}$ after fitting in multiple modes is given as,

$$\boldsymbol{H}_{(i,j)}(s) = \sum_{k=1}^{N_g} \left[\sum_{n=1}^{\boldsymbol{N}_p(k)} \frac{\boldsymbol{r}_{H}{}_{(i,j)}^{(k)}(n)}{s - \boldsymbol{p}_H{}^{(k)}(n)} \right] e^{-s\boldsymbol{\tau}(k)}, \tag{3.33}$$

where N_g is the number of modes; $\boldsymbol{N}_p(k)$ and $\boldsymbol{\tau}(k)$ are numbers of poles and time delays for fitting the k^{th} mode; and $\boldsymbol{r}_H{}^{(k)}$ and $\boldsymbol{p}_H{}^{(k)}$ are residues and poles for the k^{th} mode. In (3.30) and (3.31), the '$*$' denotes numerical complex matrix-vector convolution, which is a trade-off between the simplification of model complexity and computational resources. The admittance in (3.29) is given as,

$$\boldsymbol{G}_Y = (\frac{\Delta t}{2})/(1 - \boldsymbol{p}_Y \frac{\Delta t}{2})\boldsymbol{r}_Y + \boldsymbol{d}, \tag{3.34}$$

where $\boldsymbol{r}_Y$, $\boldsymbol{p}_Y$ and $\boldsymbol{d}$ are the residues, poles and proportional terms of the vector fitting respectively. The numerical convolution $\boldsymbol{Y} * \boldsymbol{v}(t)$ in (3.30) is defined as

$$\boldsymbol{Y} * \boldsymbol{v}(t) = \boldsymbol{c}_Y \boldsymbol{x}_Y(t), \tag{3.35}$$

where the coefficients $\boldsymbol{c}_Y$ are given as

$$\boldsymbol{c}_Y = \boldsymbol{r}_Y(\boldsymbol{\alpha}_Y + 1)\boldsymbol{\lambda}_Y, \tag{3.36}$$

and the state variables $\boldsymbol{x}_Y$ are defined as

$$\boldsymbol{x}_Y(t) = \boldsymbol{\alpha}_Y \boldsymbol{x}_Y(t - \Delta t) + \boldsymbol{v}(t - \Delta t) \tag{3.37}$$

with the coefficients $\boldsymbol{\alpha}_Y$ expressed as

$$\boldsymbol{\alpha}_Y = (1 + \boldsymbol{p}_Y \frac{\Delta t}{2})/(1 - \boldsymbol{p}_Y \frac{\Delta t}{2}). \tag{3.38}$$

Similarly, the numerical convolution $\boldsymbol{H} * \boldsymbol{i}_r(t - \tau)$ in (3.31) is defined as

$$\boldsymbol{H} * \boldsymbol{i}_r(t - \tau) = \boldsymbol{c}_H \boldsymbol{x}_H(t) + \boldsymbol{G}_H \boldsymbol{i}_r(t - \tau), \tag{3.39}$$

where the coefficients $\boldsymbol{c}_H$ are given as

$$\boldsymbol{c}_H = \boldsymbol{r}_H(\boldsymbol{\alpha}_H + 1)\boldsymbol{\lambda}_H; \tag{3.40}$$

and the state variables $\boldsymbol{x}_H(t)$ are defined as

$$\boldsymbol{x}_H(t) = \boldsymbol{\alpha}_H \boldsymbol{x}_H(t - \Delta t) + \boldsymbol{i}_r(t - \tau - \Delta t), \tag{3.41}$$

with the coefficients $\boldsymbol{\alpha}_H$ expressed as

$$\boldsymbol{\alpha}_H = (1 + \boldsymbol{p}_H \frac{\Delta t}{2})/(1 - \boldsymbol{p}_H \frac{\Delta t}{2}). \tag{3.42}$$

The propagation matrix $\boldsymbol{G}_H$ in (3.39) is given as

$$\boldsymbol{G}_H = \sum_{1}^{N_g} \boldsymbol{r}_Y \boldsymbol{\lambda}_Y. \tag{3.43}$$

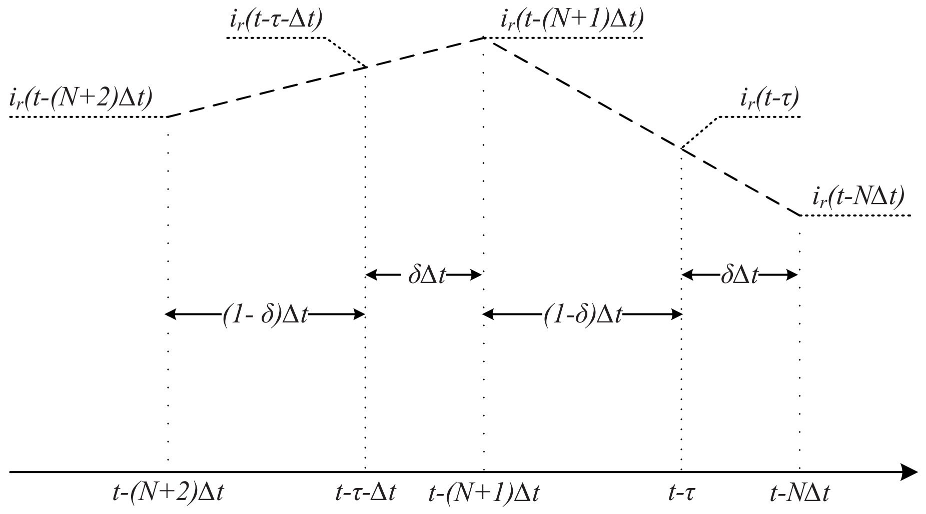


Figure 3.8: Linear interpolation.

In (3.31), there is a propagation time delay $'\tau'$ between k and m ends. Since the wave traveling time τ is not an integral multiple of the time step Δt normally, linear interpolation is used to approximate the reflected current $\boldsymbol{i}_r(t-\tau)$ in (3.39) and $\boldsymbol{i}_r(t-\tau-\Delta t)$ in (3.41). As shown in Fig. 3.8 the mean values, $i_r(t-\tau-\Delta t)$ and $i_r(t-\tau)$ are approximated by the linear interpolation as

$$i_r(t-\tau-\Delta t) = (1-\delta)i_r(t-(N+1)\Delta t) + \delta i_r(t-(N+2)\Delta t), \tag{3.44a}$$

$$i_r(t-\tau) = (1-\delta)i_r(t-N\Delta t) + \delta i_r(t-(N+1)\Delta t), \tag{3.44b}$$

where δ is the residue of τ divided by Δt, given as

$$\delta = \tau - \left[\frac{\tau}{\Delta t}\right]. \tag{3.45}$$

3.5 Linear Passive Components

The linear passive components, such as resistor (R), inductor (L) and capacitor (C) have linear current-voltage relations expressed as,

$$v_R(t) = Ri(t), \tag{3.46a}$$

$$v_L(t) = L\frac{di(t)}{dt}, \tag{3.46b}$$

$$v_C(t) = \frac{1}{C}\int i(t)dt. \tag{3.46c}$$

Discretized by trapezoidal rule, we get

$$\begin{aligned} Ri(t) &= v_R(t), && (3.47a)\\ \frac{2L}{\Delta t}i(t) &= v_L(t) + V_h^L(t-\Delta t), && (3.47b)\\ \frac{\Delta t}{2C}i(t) &= v_C(t) + V_h^C(t-\Delta t), && (3.47c) \end{aligned}$$

where $V_h^L(t-\Delta t)$ and $V_h^C(t-\Delta t)$ are the history voltages which are updated by previous results in every time-step, given as,

$$\begin{aligned} V_h^L(t-\Delta t) &= \quad v_L(t-\Delta t) - \frac{2L}{\Delta t}i(t-\Delta t), && (3.48a)\\ V_h^C(t-\Delta t) &= -v_C(t-\Delta t) - \frac{\Delta t}{2C}i(t-\Delta t). && (3.48b) \end{aligned}$$

Considering that a general linear component has all R, L, C characters, 3 equations in (3.47) are combined as,

$$R_{eq}i(t) = v(t) + V_h, \tag{3.49}$$

where V_h is the total history voltage, defined as,

$$V_h(t-\Delta t) = V_h^L(t-\Delta t) + V_h^C(t-\Delta t), \tag{3.50}$$

and R_{eq} is the equivalent resistance, defined as,

$$R_{eq} = R + \frac{2L}{\Delta t} + \frac{\Delta t}{2C} = R_{eq}^R + R_{eq}^L + R_{eq}^C. \tag{3.51}$$

Denoting the equivalent admittance G_{eq} as,

$$G_{eq} = \frac{1}{R_{eq}}, \tag{3.52}$$

and the history current I_h as,

$$I_h(t-\Delta t) = G_{eq}V_h(t-\Delta t), \tag{3.53}$$

the KCL format equation of general linear passive component is given as,

$$i(t) = G_{eq}v(t) + I_h(t-\Delta t), \tag{3.54}$$

which is used for network node analysis method.

As shown in Fig. 3.9, the linear passive component, including resistor (R), inductor (L) and capacitor (C) elements, can be represented as a unified circuit with equivalent admittance (G_{eq}), that can be calculated by the equivalent resistances (R_{eq}) from each elements,

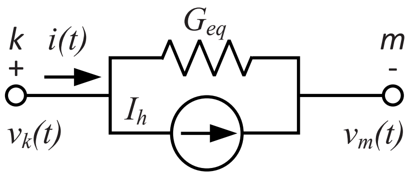


Figure 3.9: Linear passive component.

and a virtual history current source (I_h), that can be updated by the history voltages (V_h) from each elements respectively. In this case, the processing flow and equation pattern for all linear passive elements are the same and similar, that is important and helpful to the massively parallel computation on the GPU based on SIMT. Thus, the differential equations of L and C are discretized in time. Applying the network nodal analysis method to create the circuit equation

$$\boldsymbol{Y}\boldsymbol{v} = \boldsymbol{i}, \tag{3.55}$$

the numerical solutions of the electrical circuit can be obtained by solving above linear system (3.55).

3.6 Nonlinear Components

Besides linear passive elements, the nonlinear components, generally represented in Fig. 3.10, also appear in the EMT simulation often to obtain more accuracy results, such as nonlinear resistor, inductor, capacitor, surge arrestor and so on, for which the voltage and current relations are defined by a nonlinear function,

$$f(v, i) = 0, \tag{3.56}$$

instead of linear Ohm's law. Thus, a Newton type iteration method is involved to find their solutions. Theoretically, the nonlinear solution will cover the whole network including the linear and nonlinear components even if there are only few nonlinear ones since the system is nonlinear. However, applying the iterative solution to the allover network would cost too much computation resource, bring convergence problem and reduce simulation efficient. Therefore, a fine-grained decomposition method for linear network is proposed in this work to trim the size of nonlinear subsystem. When the nonlinear subsystems is decoupled with linear subsystems, only the linear components directly connecting to the nonlinear component are included in the Newton iterations. Further more, when the nonlinear subsystem has multiple nonlinear components strongly coupled each other, the

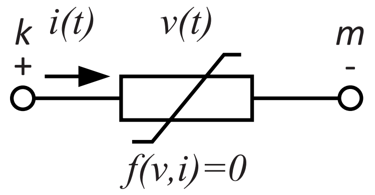


Figure 3.10: Nonlinear passive component.

extended Jacobian matrix also seriously lowers the simulation efficiency, weakens the convergence and challenges the capability of computing system. In order to reduce the size of Jacobian matrix and optimize the convergence, a decomposition method for Jacobian matrix based on domain decomposition is proposed as well. The proposed fine-grained network decomposition for both linear and nonlinear solution is detailed in Chapter 4, and an application of its usage is described in Chapter 5.

Considering the nonlinear component, the nonlinear Newton function f can be found combining with linear equation and nonlinear equation. For the latter one, there are two methods normally used to describe the nonlinearity, which are analytical function and piecewise function. If it is easy to find the derivative of the nonlinear mathematical equation, such as the elementary function, the analytical function can be applied directly; however, if some nonlinearity is described by the special function and difficult to obtain the derivative, the piecewise function is an effective replacement, which can reduce the computational complication and maintain relevant precise by controlling the number of interpolation points. And then, the derivative of f respecting to the unknown voltages v, given as

$$f' = \frac{df}{dv}, \tag{3.57}$$

which is updated in every iteration, the next approximate solutions $v^{(n+1)}$ can be solved as:

$$v^{(n+1)} = v^{(n)} - \frac{f(v^{(n)})}{f'(v^{(n)})}. \tag{3.58}$$

The iteration is repeated until $v^{(n+1)}$ and $f(v^{(n+1)})$ is converged.

$$|v^{(n+1)} - v^{(n)}| < \epsilon, \qquad |f(v^{(n+1)})| < \epsilon, \tag{3.59}$$

where ϵ is the tolerance of error. In view of the stability of simulation, a boundary of maximum iteration is preset to prevent the deadlock of the program; and a reasonable tolerance of error also significantly impacts the results.

3.7 Summary

The models of typical components in power system are described in this chapter, which are the universal machine model for synchronous machine; the lumped model for transformer and nonlinear inductance for its magnetic saturation; the universal line model for transmission line; the unified linear passive element model for linear loads; and the Newton-Raphson method for nonlinear components. From their theoretical formulas, the massively parallel algorithms can be elaborated. Since the modeling covers from linear to nonlinear, lumped to detailed, and frequency-independent to frequency-dependent, the parallel implementation can be extended to most other components in power system. Especially, an extra component with strong coupled structure and high-frequency switching function, power electronics devices such as IGBT, will be demonstrated separately in Chapter 5.

4 Shattering Network Decomposition

According to the characteristics of GPU-based computing system mentioned in Chapter 2, it can offer extraordinary computation power with its enormous cores and massive bandwidth. However, the practical effectiveness of computation is determined by the utilization of the computing resource on the GPU when the problem is solving. Based on the special execution model of the GPU, SIMT, which is different form the traditional multi-core CPU parallel computing system, the problem that can be effectively accelerated on the GPU must have not only numerous independent tasks, but also similar procedure of all these tasks. In other words, dividing the computation into decoupled branches is not enough; moreover, all those subroutines must have similar data structure and processing flow. In addition, all data for each task should be scheduled, ordered and aligned.

In order to increase the degree of parallelism, the large system will be partitioned into small enough independent divisions, which can be fully accessed by the massive threads of the GPU. Two levels of decomposition, coarse-grained and fine-grained, are proposed to decompose the large-scale electrical power circuit. As shown in Fig. 4.1, the original circuit is divided into *linear subsystems* (LSs), *nonlinear subsystems* (NLSs), and *control systems* (CSs) in the first-level decomposition by propagation delay based partitioning. In linear subsystem (LS), there are only linear components, which can be represented by linear matrix system and solved by a linear solver such as Gaussian elimination directly or Jacobi method iteratively. In nonlinear subsystem (NLS), on the other hand, it contains not only nonlinear components but also linear components connecting to the same nodes,

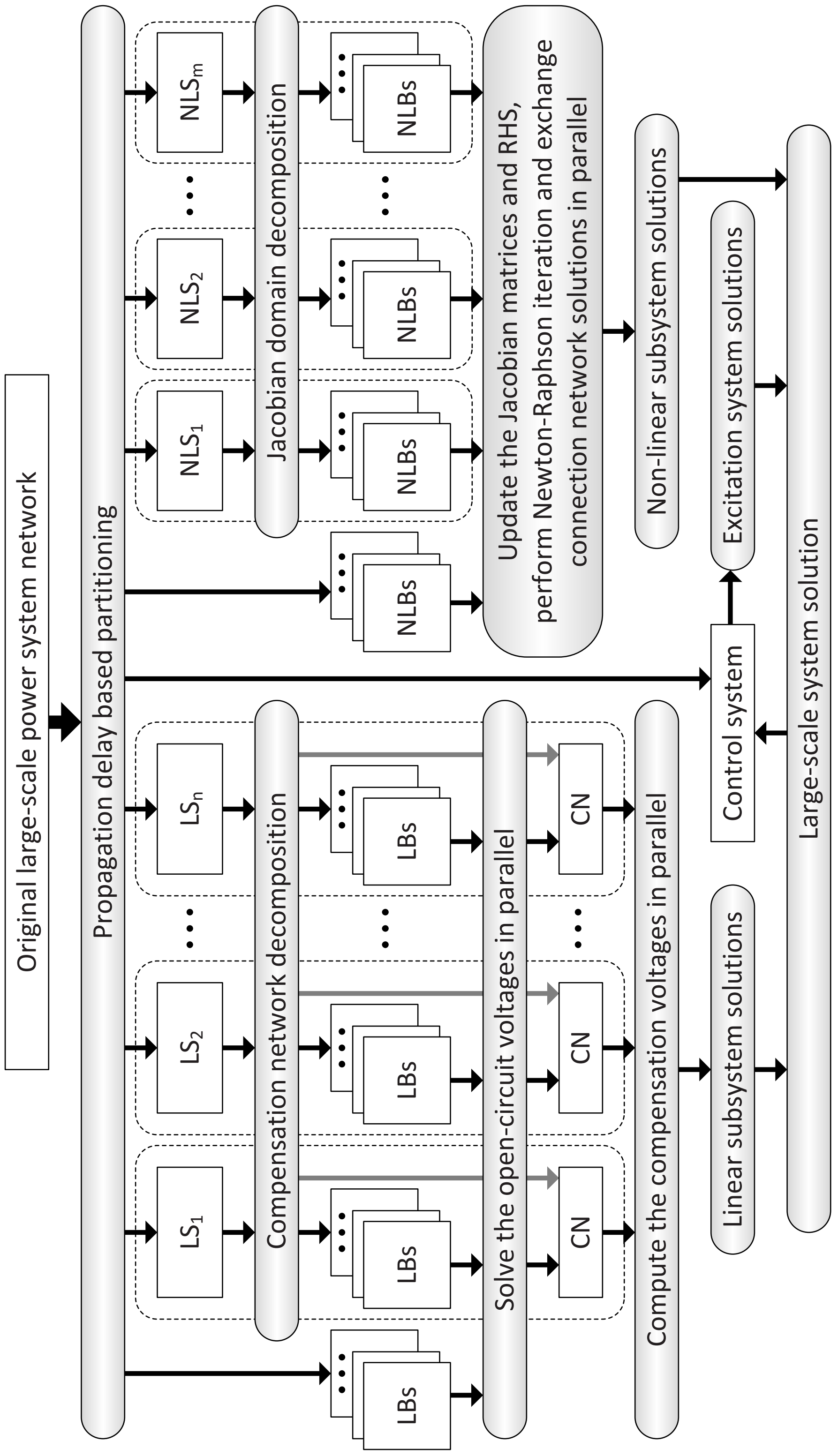

Original large-scale power system network
Propagation delay based partitioning
LS_1
LS_2
LS_n
NLS_1
NLS_2
NLS_m
Compensation network decomposition
Jacobian domain decomposition
LBs
NLBs
CN
Solve the open-circuit voltages in parallel
Compute the compensation voltages in parallel
Update the Jacobian matrices and RHS, perform Newton-Raphson iteration and exchange connection network solutions in parallel
Linear subsystem solutions
Non-linear subsystem solutions
Excitation system solutions
Control system
Large-scale system solution

Figure 4.1: Shattering network decomposition.

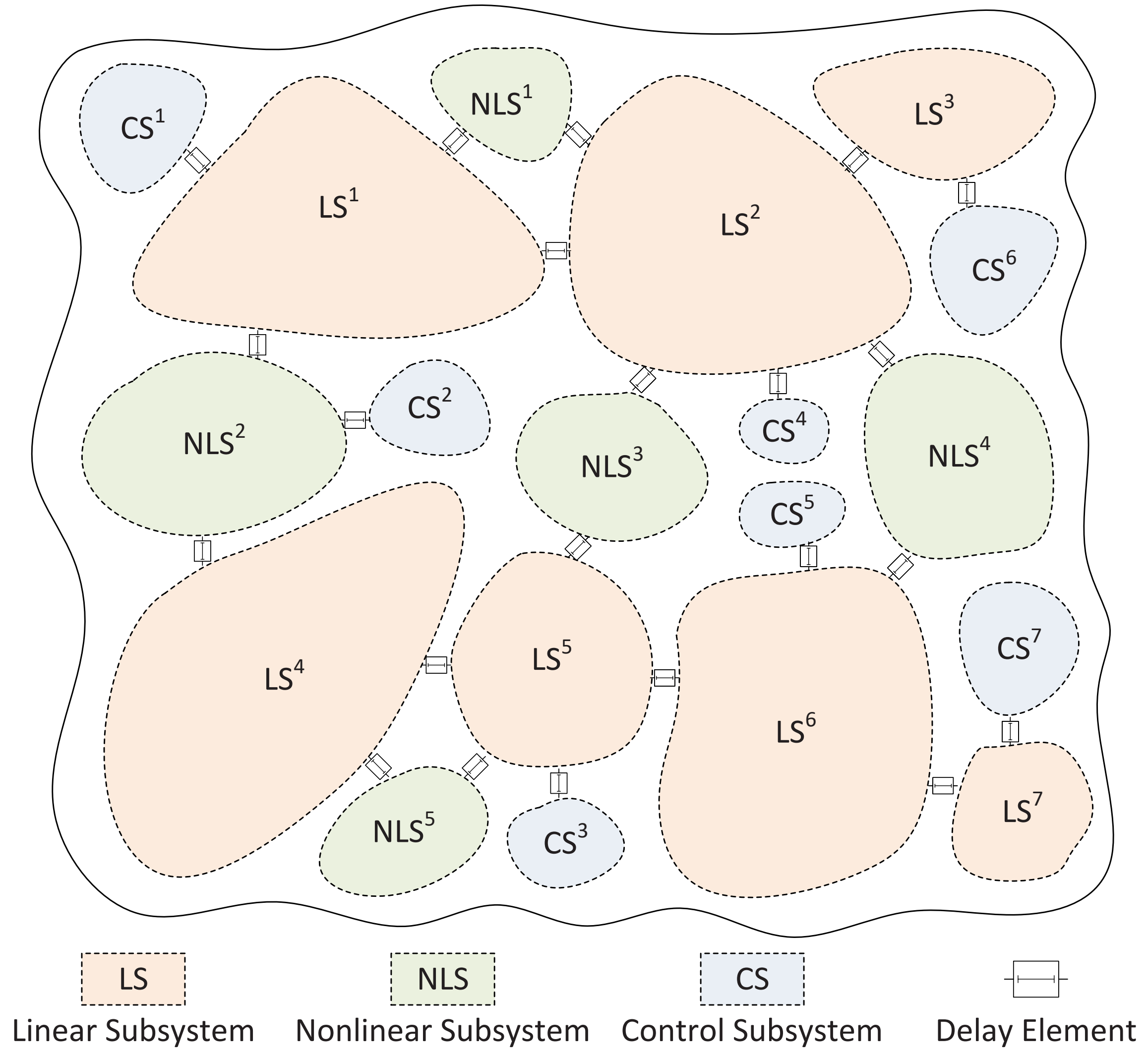


Figure 4.2: Propagation delay based partition.

which cannot be solved by linear solving methods and has to be solved by nonlinear solving method such as Newton-Raphson iterative method. In other words, NLS may not be a pure nonlinear subsystem. The control system (CS) only contains the control logic with different sample rate from the main EMTP simulation. Fig. 4.2 shows an example of the first level decomposition based on propagation delay, which partitions the large network into LSs, NLSs and CSs in coarse-grained along the border of delay elements' connection. In the second-level, the fine-grained decomposition methods are different for linear and nonlinear subsystems due to the solution methods. The linear subsystems are partitioned into *linear blocks* (LB) and *connecting network* (CN) by compensation network decomposition, through which the linear solutions can be obtained by solving the open-circuit

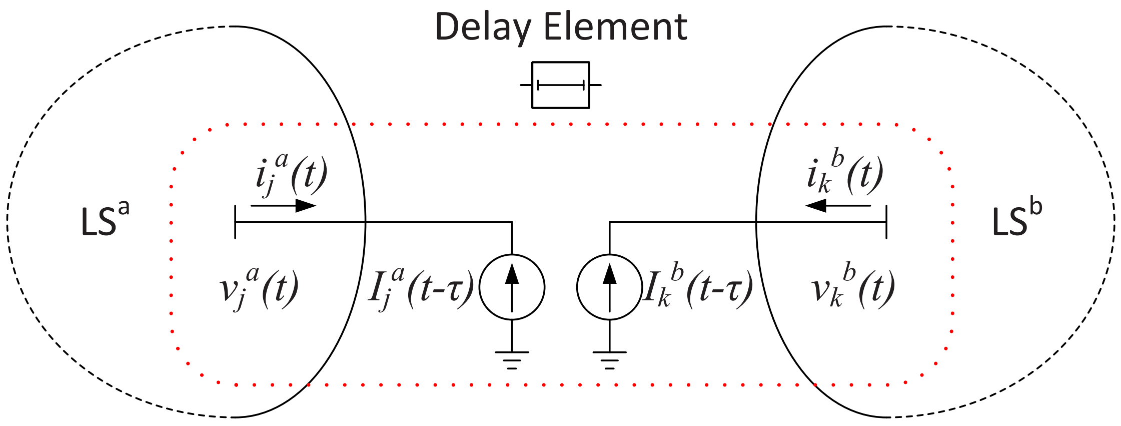


Figure 4.3: Delay element equivalent circuit.

voltages and compensation voltages in parallel. For nonlinear subsystems, the nonlinear components are detached from the network into *nonlinear blocks* (NLB) by Jacobian domain decomposition computed independently in parallel, and then the temporary results are interfaced to the rest of the connecting network to find the temporary solutions. Iterations of the above two-step computations are repeated until the solutions of the subsystems converge. All linear and nonlinear subsystems solutions are integrated to obtain the final solutions of the large-scale system. When the sample step arrives, the integrated solutions are sent to the *control block* (CB) to obtain the excitation signals which are fed back to the main circuit for the EMT computation of next time-step.

4.1 First-level decomposition (Coarse-grained)

Observing the procedure of modeling and discretization, some components, such as transmission lines and control systems, have time delays inside or for connecting to external components, which provide the native characteristics to decouple the network based on propagation delay [42]. Since the network variables, such as voltages and currents, of the nodes between those delay elements are asynchronous at the same time step, the subsystems divided by them can be calculated separately. In the parallel computing platform, this independence of computation can be utilized for designing the parallel algorithm of EMT simulation.

The equivalent circuit after delay based decomposition is shown in Fig. 4.3, where the delay element is represented as two current sources connecting to the nodes on both sides respectively, and τ is the delay time of signal propagation between two nodes linked by the delay element. The injected current $I_j^a(t-\tau)$ and $I_k^b(t-\tau)$ are history values calculated previously, given as

$$I_j^a(t-\tau) = f^a(v_k^b(t-\tau), i_k^b(t-\tau)), \tag{4.1}$$

$$I_k^b(t-\tau) = f^b(v_j^a(t-\tau), i_j^a(t-\tau)), \tag{4.2}$$

which are the functions, f^a and f^b, relating to the previous values by τ delay in opposite peers. For example, in ULM of transmission line, the propagation delay τ in (3.31) is given as,

$$\tau = (\beta/\omega)l = \sqrt{LC}l, \tag{4.3}$$

where β is the phase constant, ω is the angular frequency, L is the intrinsic inductance, C is the intrinsic capacitance and l is the length of the transmission line. Similarly the control system can also be decoupled by T_s time delay. Since the transmission line is one of the essential components, the large network is partitioned into small subsystems, which are classified into linear subsystems decomposed into linear blocks and connecting networks, and nonlinear subsystems decomposed into nonlinear blocks.

In the GPU architecture, shared memory is much faster than other memory types, so it is used to store the initial and intermediate data when the task is dispatched to the CUDA block, and only the result is sent back to global memory. On the other hand, registers of CUDA threads have more bandwidth than shared memory during arithmetic operations. The way to maximize the usage of registers, reaching the best performance, is to unroll the *for* loops in CUDA threads because the registers cannot be indexed and optimized by GPU automatically, which multiplies the workload of each CUDA thread to reduce communication, overlap memory access and increases occupation [43]. Since the general electrical network is 3-phase, the number of operations is multiples of 3 normally. Considering the number of threads per CUDA warp, the amount of shared memory per CUDA block and the number of registers per CUDA thread, which are the fundamental criterions of determining the subsystem partition granularity, the node limitation of each network block is set to 12 (4×3-phase buses) and the unroll factors can be 2, 3 or their multiple. Therefore, the subsystems with less than 12 nodes can be sent to GPU directly as network blocks.

4.2 Second-level decomposition (Fine-grained)

After the large network is partitioned into subsystems in fine-grained, the ones which are relevantly small with less than 12 buses are grouped into linear blocks (LBs) and nonlinear

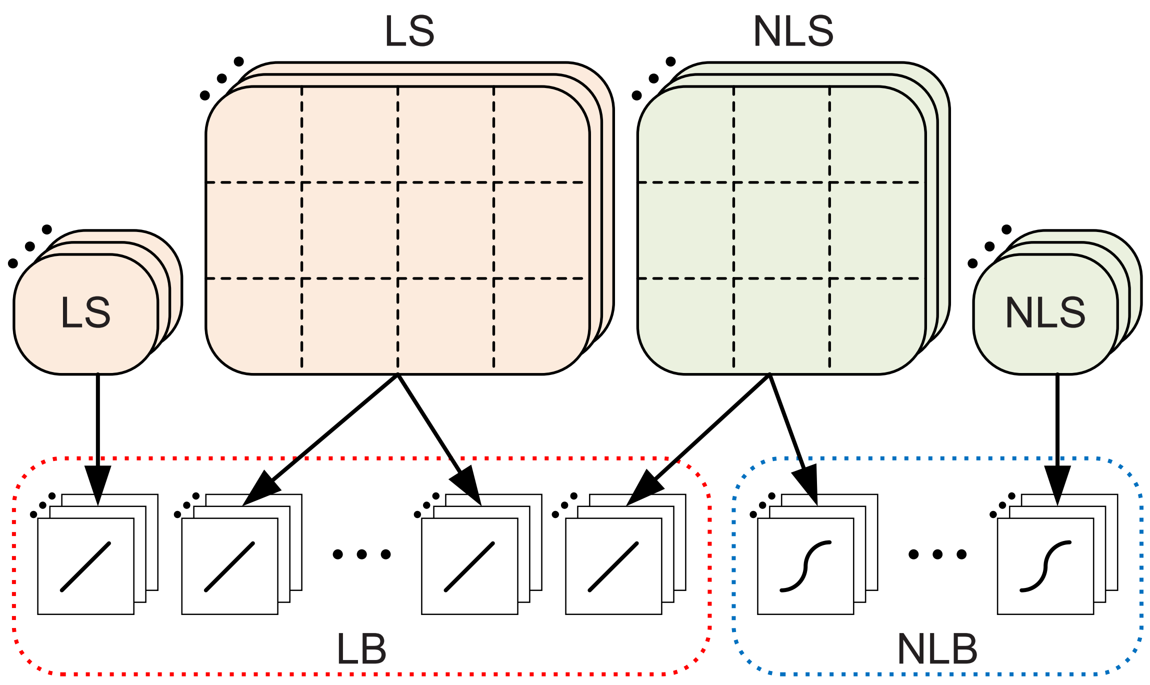


Figure 4.4: Fine-grained partition for LSs and NLSs.

blocks (NLBs), as shown in Fig. 4.4. In contrast, the subsystems with more than 12 buses will be divided into smaller blocks by fine-grained methods. Since linear subsystems (LSs) are made of linear components, they can only be divided into linear blocks (LBs); while nonlinear subsystems (NLSs) contain both linear and nonlinear components, they produce not only nonlinear blocks (NLBs) but also linear blocks (LBs) in the second-level partition. According to the difference of topological characteristics and solution methods of linear and nonlinear networks, there are two fine-grained decomposition methods proposed: compensation network decomposition for linear subsystems, and Jacobian domain decomposition for nonlinear subsystems, which will be detailed in following sections respectively.

4.2.1 Compensation network decomposition

Considering an N-node linear subsystem deriving by first-level decomposition, its linear system equations can be obtained using nodal analysis method, given as

$$\boldsymbol{Y}\boldsymbol{v} = \boldsymbol{i}, \tag{4.4}$$

where the admittance matrix $\boldsymbol{Y}$ is sparse and its dimension is N by N. In order to increasing the parallelism of linear solution, the fine-grained decomposition method, compensation network decomposition, for coupled linear system is applied by which the linear

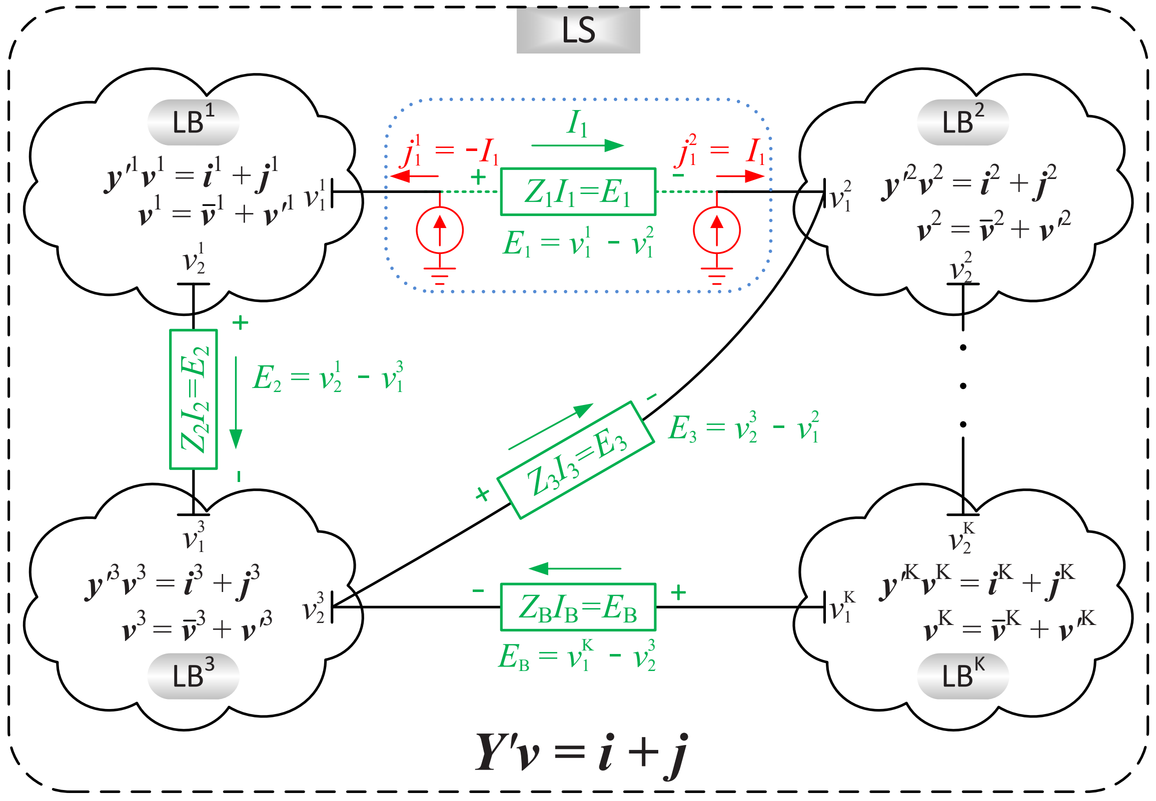


Figure 4.5: Compensation network decomposition for LS.

subsystem (LS) can be divided into small linear blocks (LBs) and solved in parallel without iteration.

As shown in Fig. 4.5, LS can be partitioned into K linear blocks (LBs) by breaking down B links, on which there is a linear component with nonzero impedance connecting linear blocks via its both nodes. If the impedance is zero, those nodes connected can be considered as the same one, and then the other link will be chosen for the decomposition. Since there are B branch currents go through the links, defined as

$$\boldsymbol{I} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_B \end{bmatrix}, \tag{4.5}$$

equivalent node currents are injected to the relevant nodes when the linear components on the links are removed, which are named as $\boldsymbol{j}$. Since the B links are broken down, the LS network is divided into K LBs, in which there are n_k $(k \in 1, 2, \cdots, K)$ nodes for each LB.

Thus, the total number of nodes in LS is given as

$$N = \sum_{k=1}^{K} n_k \tag{4.6}$$

After remapping the node indices by LB to make them continuous for each LB, the linear system (4.4) can be rewritten as:

$$\begin{array}{ccccccc} \boldsymbol{Y}' & \cdot & \boldsymbol{v} & = & \boldsymbol{i} & + & \boldsymbol{j} \\ \begin{bmatrix} \boldsymbol{y}'^1 & & & \\ & \boldsymbol{y}'^2 & & \\ & & \ddots & \\ & & & \boldsymbol{y}'^K \end{bmatrix} & & \begin{bmatrix} \boldsymbol{v}^1 \\ \boldsymbol{v}^2 \\ \vdots \\ \boldsymbol{v}^K \end{bmatrix} & = & \begin{bmatrix} \boldsymbol{i}^1 \\ \boldsymbol{i}^2 \\ \vdots \\ \boldsymbol{i}^K \end{bmatrix} & + & \begin{bmatrix} \boldsymbol{j}^1 \\ \boldsymbol{j}^2 \\ \vdots \\ \boldsymbol{j}^K \end{bmatrix} \end{array}, \tag{4.7}$$

where large system $\boldsymbol{Y}$ is decomposed into $\boldsymbol{Y}'$, which contains K decoupled small systems $\boldsymbol{y}'^k$ $(k \in 1, 2, \cdots, K)$ for every LBs. Similarly, the node voltages $\boldsymbol{v}$, node currents $\boldsymbol{i}$ and additional equivalent injection currents $\boldsymbol{j}$ are also regrouped according to the decoupled LBs. The linear system matrix of kth LB is defined as

$$\boldsymbol{y}'^k = \begin{bmatrix} y_{11}^k & y_{12}^k & \cdots & y_{1n_k}^k \\ y_{21}^k & y_{22}^k & \cdots & y_{2n_k}^k \\ \vdots & \vdots & \ddots & \vdots \\ y_{n_k1}^k & y_{n_k2}^k & \cdots & y_{n_kn_k}^k \end{bmatrix}, \tag{4.8}$$

whose dimension is n_k by n_k. Similarly, the node voltages of kth LB are given as

$$\boldsymbol{v}^k = \begin{bmatrix} v_1^k \\ v_2^k \\ \vdots \\ v_{n_k}^k \end{bmatrix}; \tag{4.9}$$

the kth node currents are

$$\boldsymbol{i}^k = \begin{bmatrix} i_1^k \\ i_2^k \\ \vdots \\ i_{n_k}^k \end{bmatrix}; \tag{4.10}$$

and the kth injection currents are

$$\boldsymbol{j}^k = \begin{bmatrix} j_1^k \\ j_2^k \\ \vdots \\ j_{n_k}^k \end{bmatrix}. \tag{4.11}$$

Solving for (4.7), we get

$$\boldsymbol{v} = (\boldsymbol{Y}')^{-1}\boldsymbol{i} + (\boldsymbol{Y}')^{-1}\boldsymbol{j}. \tag{4.12}$$

From (4.12), the linear system solution v can be considered as a combination of two parts: one is for open circuits defined as,

$$\overline{\boldsymbol{v}} = (\boldsymbol{Y}')^{-1}\boldsymbol{i}, \tag{4.13}$$

the other is for injection compensation, given as

$$\boldsymbol{v}' = (\boldsymbol{Y}')^{-1}\boldsymbol{j}. \tag{4.14}$$

Since the decoupled large linear system matrix, $\boldsymbol{Y}'$, is in block diagonal format and the node currents, $\boldsymbol{i}$, are also grouped by LBs, the open circuit solution, $\overline{\boldsymbol{v}}$, can be solved with additional block-level parallelism on the GPU avoiding most zero elements in the sparse matrix structure.

Since the injection currents, $\boldsymbol{j}$, are caused by the branch currents, $\boldsymbol{I}$, of the omitted links, they can be expressed as

$$\begin{array}{ccccc} \boldsymbol{j} & = & \boldsymbol{c} & \cdot & \boldsymbol{I}^{\dagger} \\ \begin{bmatrix} \boldsymbol{j}^1 \\ \boldsymbol{j}^2 \\ \vdots \\ \boldsymbol{j}^K \end{bmatrix} & = & \begin{bmatrix} \boldsymbol{c}^1 \\ \boldsymbol{c}^2 \\ \vdots \\ \boldsymbol{c}^K \end{bmatrix} & & \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_B \end{bmatrix} \end{array}, \tag{4.15}$$

where the dimension of the coordination matrix, **c**, is $N \times B$ according to (4.6). The structure of the coordination matrix for kth BL is given as

$$\boldsymbol{c}^k = \begin{bmatrix} c^k_{11} & c^k_{12} & \cdots & c^k_{1B} \\ c^k_{21} & c^k_{22} & \cdots & c^k_{2B} \\ \vdots & \vdots & \ddots & \vdots \\ c^k_{n_k 1} & c^k_{n_k 2} & \cdots & c^k_{n_k B} \end{bmatrix}. \tag{4.16}$$

The coordination matrix indicates the connecting relations between skipped branch currents and injection node currents, which only maintains one value of sign function among -1, 1 and 0. All nodes of the linear subsystem have themselves coordination values. For common nodes, since the connection at these nodes are not changed during the decomposition, there is no additional equivalent injection currents involved, so that the coordination

[†]The superscripts present indexes

values are set to '0'. If the link connected to a node is removed for the decomposition, an additional equivalent injection current is taken account, therefore, the coordination value is '1' or '-1' depending on the direction of branch current, defined as

$$c_{nb} = \begin{cases} 1 & \text{if the same direction,} \\ 0 & \text{if no removed link,} \\ -1 & \text{if the opposite direction.} \end{cases} \qquad n \in (1, 2, \cdots, N), b \in (1, 2, \cdots, B) \tag{4.17}$$

Taking the circuit network in Fig. 4.5 for example, the coordination matrix, $\boldsymbol{c}$, is given as

$$\boldsymbol{c} = \left[\begin{array}{ccccc} -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ \hdashline 1 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & * & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ \hdashline 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ \hdashline & & \vdots & & \\ \hdashline 0 & 0 & 0 & \cdots & -1 \\ 0 & 0 & 0 & * & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{array}\right], \tag{4.18}$$

where the row indexes are grouped by the order of linear blocks. Similarly, the potential difference, $\boldsymbol{E}$, across the connecting linear components can be represented by the node voltages, $\boldsymbol{v}$, as

$$\begin{array}{ccccc} \boldsymbol{E} & = & \boldsymbol{d} & \cdot & \boldsymbol{v} \\ \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_B \end{bmatrix} & = & \begin{bmatrix} \boldsymbol{d}^1 & \boldsymbol{d}^2 & \cdots & \boldsymbol{d}^K \end{bmatrix} & & \begin{bmatrix} \boldsymbol{v}^1 \\ \boldsymbol{v}^2 \\ \vdots \\ \boldsymbol{v}^K \end{bmatrix} \end{array}, \tag{4.19}$$

where the elements of transformation matrix, $\boldsymbol{d}$, are also the values of sign function, '-1', '0' or '1', depending on the connection of the omitted linear components. The dimension of $\boldsymbol{d}$ is $B \times N$, and the structure of kth matrix is given as

$$\boldsymbol{d}^k = \begin{bmatrix} d_{11}^k & d_{12}^k & \cdots & d_{1B}^k \\ d_{21}^k & d_{22}^k & \cdots & d_{2B}^k \\ \vdots & \vdots & \ddots & \vdots \\ d_{n_k 1}^k & d_{n_k 2}^k & \cdots & d_{n_k B}^k \end{bmatrix}. \tag{4.20}$$

For example, the equations of potential differences in Fig. 4.5 are given as

$$\begin{cases} E_1 = v_1^1 - v_1^2, \\ E_2 = v_2^1 - v_1^3, \\ E_3 = v_2^3 - v_1^2, \\ \qquad \vdots \\ E_B = v_1^K - v_2^3, \end{cases} \tag{4.21}$$

which can be rewritten to matrix mode, where the $\boldsymbol{d}$ is obtained as

$$\boldsymbol{d} = \left[\begin{array}{cccc|cccc|cccc|c|cccc} 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & * & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & * & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & -1 & \cdots & 0 & & 1 & 0 & \cdots & 0 \end{array}\right]. \tag{4.22}$$

Therefore, the transformation matrix in (4.19) can be derived from the coordination matrix in (4.15), defined as

$$\boldsymbol{d} = -\boldsymbol{c}^T, \tag{4.23}$$

when defining the direction of branch currants as same as the potential difference of connecting linear components. Then, (4.19) is expressed as

$$\boldsymbol{E} = -\boldsymbol{c}^T \boldsymbol{v}. \tag{4.24}$$

Applying Ohm's Law to the connecting components, we obtain

$$\begin{array}{ccccc} \boldsymbol{Z} & \cdot & \boldsymbol{I} & = & \boldsymbol{E} \\ \begin{bmatrix} Z_1 & & & \\ & Z_2 & & \\ & & \ddots & \\ & & & Z_B \end{bmatrix} & & \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_B \end{bmatrix} & = & \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_B \end{bmatrix} \end{array}, \tag{4.25}$$

where the impedance matrix, $\boldsymbol{Z}$, is a B-order diagonal matrix. Combining (4.24) and (4.25) gives:

$$\boldsymbol{Z} \cdot \boldsymbol{I} = -\boldsymbol{c}^T \boldsymbol{v}, \tag{4.26}$$

then, substituting (4.12) to (4.26) gives:

$$\boldsymbol{Z}\cdot\boldsymbol{I} = -\boldsymbol{c}^T(\overline{\boldsymbol{v}} + \boldsymbol{v}'), \tag{4.27}$$

and substituting (4.14) to (4.27) gets:

$$\boldsymbol{Z}\cdot\boldsymbol{I} = -\boldsymbol{c}^T(\overline{\boldsymbol{v}} + (\boldsymbol{Y}')^{-1}\boldsymbol{j}). \tag{4.28}$$

Replacing the injecting currents $\boldsymbol{j}$ with (4.15) gives:

$$\boldsymbol{Z}\boldsymbol{I} = -\boldsymbol{c}^T(\overline{\boldsymbol{v}} + (\boldsymbol{Y}')^{-1}\boldsymbol{c}\boldsymbol{I}). \tag{4.29}$$

Define an equivalent impedance matrix as

$$\boldsymbol{Z}' = \boldsymbol{Z} + \boldsymbol{c}^T(\boldsymbol{Y}')^{-1}\boldsymbol{c}, \tag{4.30}$$

and reformatting (4.29), we get:

$$\boldsymbol{Z}'\boldsymbol{I} = -\boldsymbol{c}^T\overline{\boldsymbol{v}}. \tag{4.31}$$

then, the branch currents $\boldsymbol{I}$ can be represented as

$$\boldsymbol{I} = -(\boldsymbol{Z}')^{-1}\boldsymbol{c}^T\overline{\boldsymbol{v}}. \tag{4.32}$$

Since the computation complexity of (4.32) is determined by the order of $\boldsymbol{Z}'$, which is B, the same as that of $\boldsymbol{Z}$, partitioning a network with fewer connection links results in less computational load. With (4.32), the compensation solutions $\boldsymbol{v}'$ can be obtained as

$$\boldsymbol{v}' = -(\boldsymbol{Y}')^{-1}\boldsymbol{c}(\boldsymbol{Z}')^{-1}\boldsymbol{c}^T\overline{\boldsymbol{v}}. \tag{4.33}$$

Thus, the final solutions can be written as:

$$\boldsymbol{v} = (1 - (\boldsymbol{Y}')^{-1}\boldsymbol{c}(\boldsymbol{Z}')^{-1}\boldsymbol{c}^T)\overline{\boldsymbol{v}}. \tag{4.34}$$

In (4.34), all variables including $\overline{\boldsymbol{v}}$, $\boldsymbol{Y}'$, $\boldsymbol{c}$ and $\boldsymbol{Z}'$ are organized based on LBs and can be calculated independently and concurrently. Since the coordination matrix, $\boldsymbol{c}$, have few nonzero elements, column-wise compression can be used to increase the matrix storage efficiency; and similarly, row-wise compression can be used for $\boldsymbol{c}^T$.

The compensation network decomposition method partitions the linear subsystem into fine-grained linear blocks, and the sparse admittance matrices are decoupled into small size linear systems which have much more density. Therefore, the GPU-based dense direct

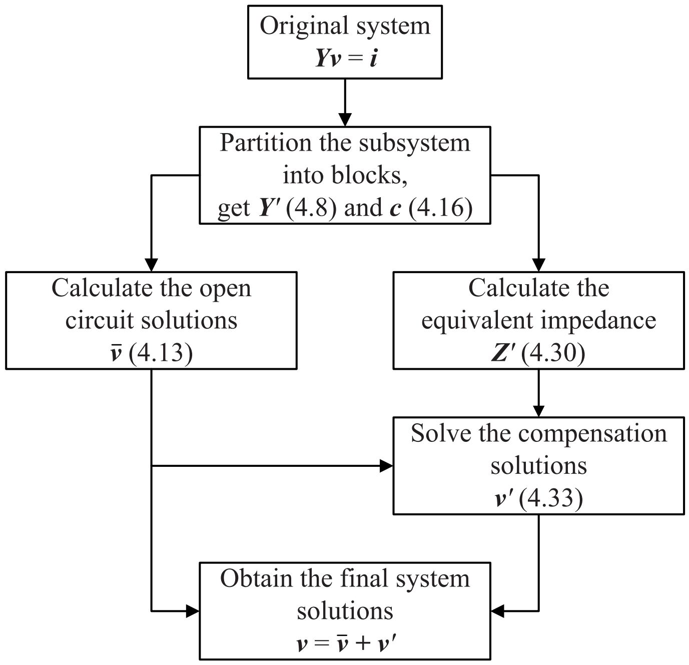
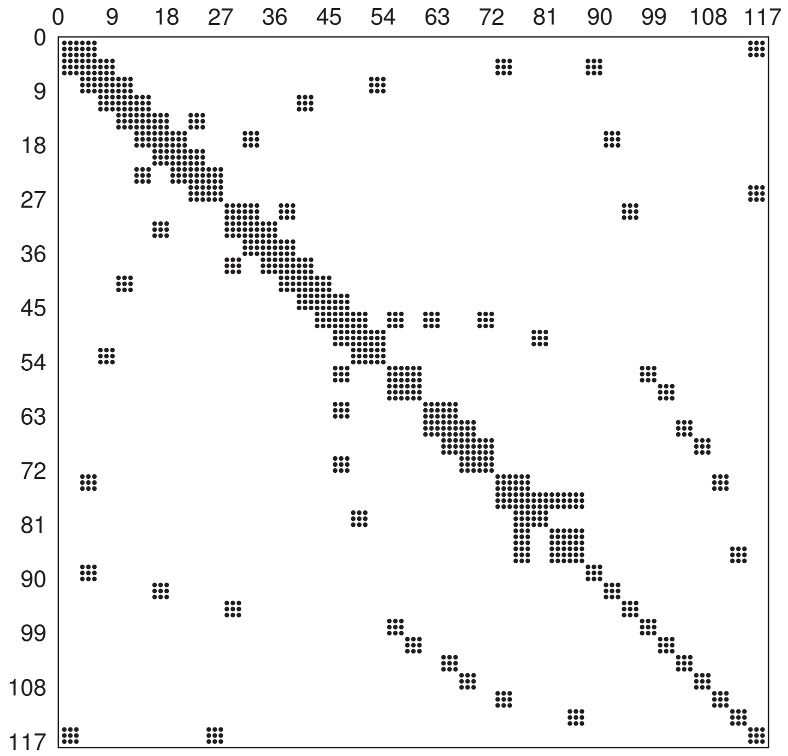

Original system
$Yv = i$
Partition the subsystem into blocks, get Y' (4.8) and c (4.16)
Calculate the open circuit solutions $\bar{v}$ (4.13)
Calculate the equivalent impedance Z' (4.30)
Solve the compensation solutions v' (4.33)
Obtain the final system solutions $v = \bar{v} + v'$

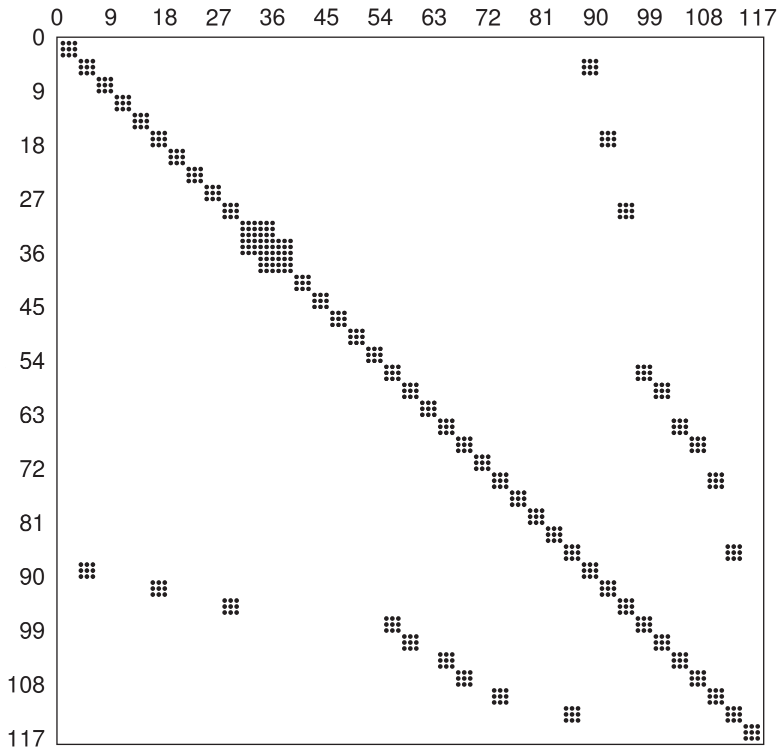
Figure 4.6: Flowchart of compensation network decomposition.

linear solution algorithms, including matrix LU factorization and inverse, can be implemented to solve the decomposed linear subsystems in parallel without iterations. The working flow is shown in Fig. 4.6

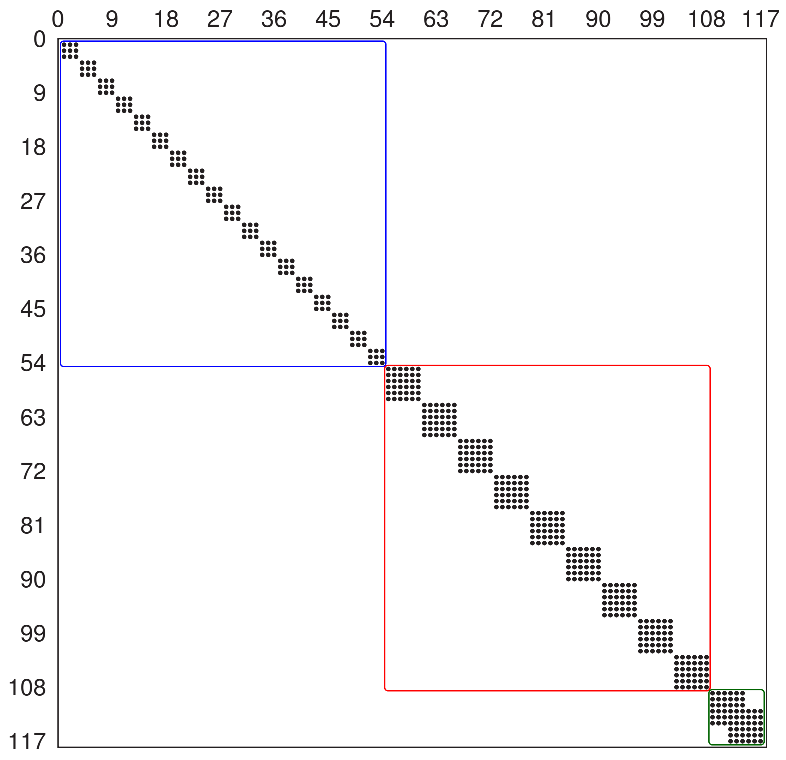
An example of transforming the Y matrix of 39-bus system is shown in Fig. 4.7. The original admittance matrix shown in Fig. 4.7(a) with random index has a irregularly sparse pattern since the node numbers are created by user or system randomly, whose elements are coupled and sparse. After decomposition, the structure is much more concise as shown in Fig. 4.7(b), but the shape is still not good enough. Applying the block node adjustment and reordering the blocks, a perfect block diagram shape is shown in 4.7(c) adapting to the requirement of GPU-based SMIT computation, which can be grouped by the dimension of the blocks. Finally, in order to increase the efficiency of memory, most of zero area are trimmed and only the elements of blocks are saved with just few zeros according to the order of the block dimensions. When the admittances connecting to some nodes are changed during computation, only the block related to the relevant nodes need to be updated, and the LU and inverse operation are only limited within those blocks. Even if the topology is changed due, the difference is only propagating inside the block or linking few blocks as well, without spreading the whole system.


0
9
18
27
36
45
54
63
72
81
90
99
108
117

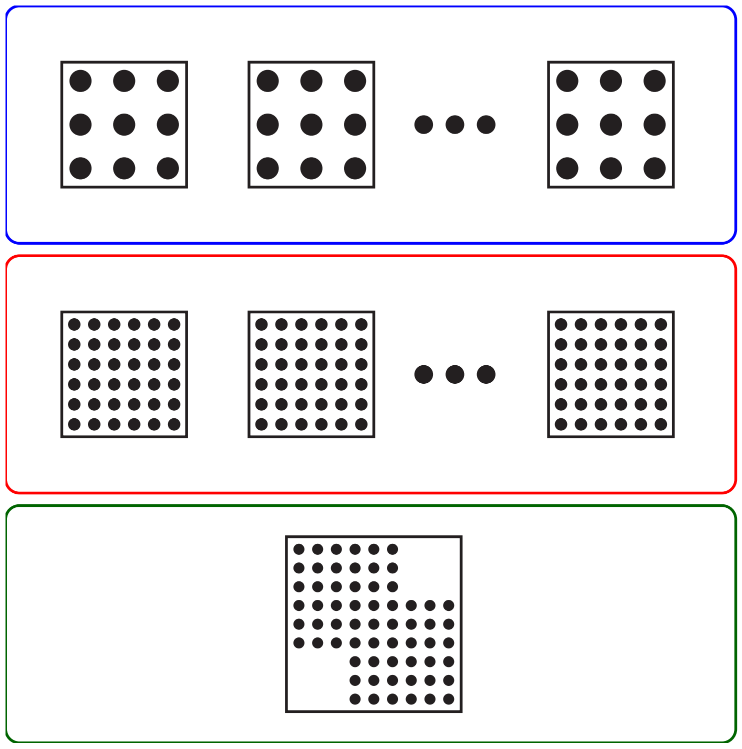
(a)


0
9
18
27
36
45
54
63
72
81
90
99
108
117

(b)



(c)



(d)

Figure 4.7: $\boldsymbol{Y}$ matrix transformation.

4.2.2 Jacobian domain decomposition

For the solutions of nonlinear subsystems, Newton-Raphson iteration (NRI) method is applied. If there are multiple nonlinear components in the subsystem, all components in the subsystem including linear ones will be involved in the NRI theoretically. Thus, the size of the Jacobian matrix in NRI will extended to large according to the number of components since extra component has additional variables and equations. However, the large sparse Jacobian matrix will impact the solution convergence and the application of parallelism. Therefore, Jacobian domain decomposition method is proposed to decouple the nonlinear elements into nonlinear blocks.

As shown in Fig. 4.8, for a nonlinear subsystem (NLS) with M groups of nonlinear components, it can be decomposed into M nonlinear blocks (NLBs). Since the pattern of Jacobian matrix created in the NRI is decided by the topology of NLS, the partitioned subsystem will derive a block diagonal format for the Jacobian matrix, by which the Newton function can be solved in parallel on GPU-based computing system. In the divided nonlinear subsystem (NLS), each nonlinear block (NLB) has l_m $(m \in 1, 2, \cdots, M)$ links connecting to the other NLBs. The variables of those connection, node voltages ($\boldsymbol{\nu}$) and currents ($\boldsymbol{\iota}$), are defined as

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\nu}^1 \\ \boldsymbol{\nu}^2 \\ \vdots \\ \boldsymbol{\nu}^M \end{bmatrix}, \quad \boldsymbol{\iota} = \begin{bmatrix} \boldsymbol{\iota}^1 \\ \boldsymbol{\iota}^2 \\ \vdots \\ \boldsymbol{\iota}^M \end{bmatrix}, \tag{4.35}$$

where the connection voltages and currents for mth NLB is given as

$$\boldsymbol{\nu}^m = \begin{bmatrix} \nu_1^m \\ \nu_2^m \\ \vdots \\ \nu_{l_m}^m \end{bmatrix}. \tag{4.36}$$

Denoting $\boldsymbol{\chi}^m$ as all internal variables of the mth block, which may be any internal voltages, currents, fluxes and etc. bound with nonlinear relations inside the NLB^m, the nonlinear equations can be expressed as

$$\boldsymbol{f}^m(\boldsymbol{\nu}^m, \boldsymbol{\iota}^m, \boldsymbol{\chi}^m) = 0, \tag{4.37}$$

where $\boldsymbol{f}^m$ represent the nonlinear relations of NLB^m.

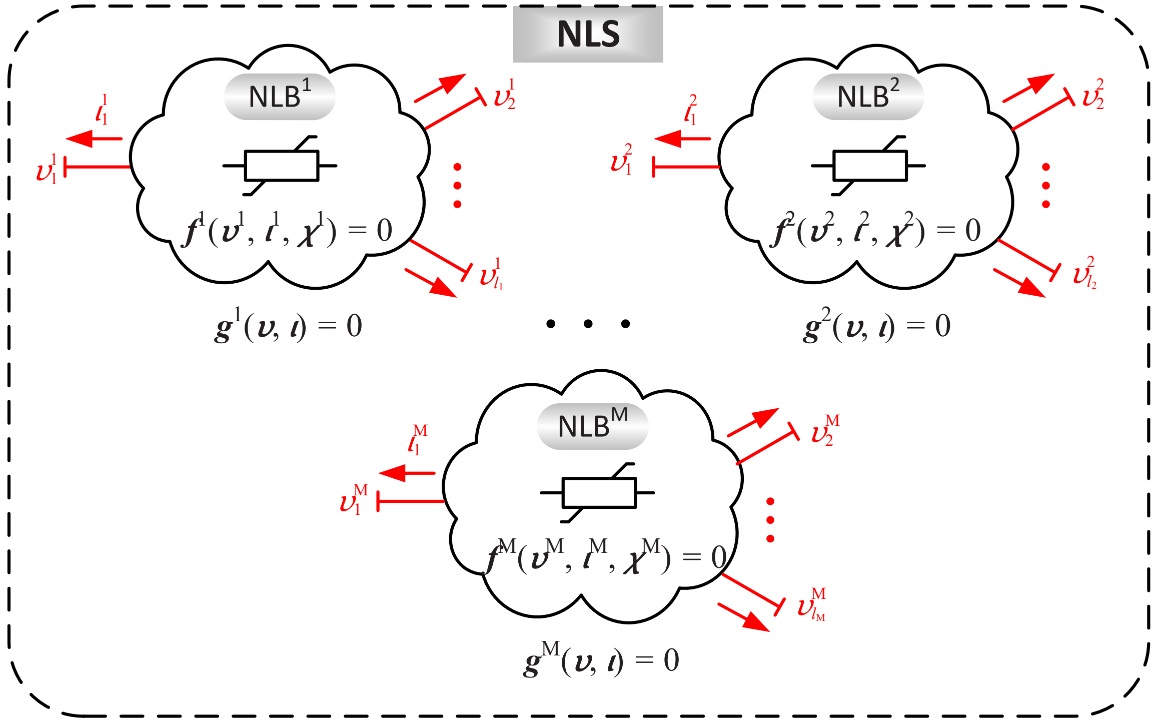


Figure 4.8: Jacobian domain decomposition for NLS.

On the other hand, since the relations between NLBs are linear, the linkages of the M nonlinear blocks in the subsystem can be described by a set of linear equations, which are denoted $\boldsymbol{g}^m$, given as

$$\begin{cases} \boldsymbol{g}^1(\boldsymbol{\nu}, \boldsymbol{\iota}) = 0 \\ \boldsymbol{g}^2(\boldsymbol{\nu}, \boldsymbol{\iota}) = 0 \\ \quad\vdots \\ \boldsymbol{g}^M(\boldsymbol{\nu}, \boldsymbol{\iota}) = 0 \end{cases} \tag{4.38}$$

For the mth block, $\boldsymbol{\nu}$ and $\boldsymbol{\iota}$ can also be rewritten by $\boldsymbol{\nu}^m$, $\boldsymbol{\iota}^m$ and their complementary $\overline{\boldsymbol{\nu}}^m$, $\overline{\boldsymbol{\iota}}^m$, which defined as,

$$\boldsymbol{\nu} = [\boldsymbol{\nu}^m, \overline{\boldsymbol{\nu}}^m] \quad (\overline{\boldsymbol{\nu}}^m = [\boldsymbol{\nu}^1, \cdots, \boldsymbol{\nu}^{m-1}, \boldsymbol{\nu}^{m+1}, \cdots, \boldsymbol{\nu}^M]), \tag{4.39a}$$

$$\boldsymbol{\iota} = [\boldsymbol{\iota}^m, \overline{\boldsymbol{\iota}}^m] \quad (\overline{\boldsymbol{\iota}}^m = [\boldsymbol{\iota}^1, \cdots, \boldsymbol{\iota}^{m-1}, \boldsymbol{\iota}^{m+1}, \cdots, \boldsymbol{\iota}^M]). \tag{4.39b}$$

Since (4.38) is linear, $\boldsymbol{g}^m$ can be reformatted as $\overline{\boldsymbol{g}}^m$ for the node currents $\boldsymbol{\iota}^m$ after substituting (4.39) into (4.38), represented as

$$\boldsymbol{\iota}^m = \overline{\boldsymbol{g}}^m(\boldsymbol{\nu}^m, \overline{\boldsymbol{\nu}}^m, \overline{\boldsymbol{\iota}}^m). \tag{4.40}$$

Substituting (4.40) into (4.37), the nonlinear equations are updated as

$$\boldsymbol{f}^m(\boldsymbol{\nu}^m, \overline{\boldsymbol{g}}^m(\boldsymbol{\nu}^m, \overline{\boldsymbol{\nu}}^m, \overline{\boldsymbol{\iota}}^m), \boldsymbol{\chi}^m) = 0. \tag{4.41}$$

Denoting $\overline{\boldsymbol{f}}^m$ for the reorganized nonlinear equations gives

$$\overline{\boldsymbol{f}}^m(\boldsymbol{\nu}^m, \boldsymbol{\chi}^m, \overline{\boldsymbol{\nu}}^m, \overline{\boldsymbol{\iota}}^m) = 0. \tag{4.42}$$

where the complementary variables $\overline{\boldsymbol{\nu}}^m, \overline{\boldsymbol{\iota}}^m$ can be trimmed by using previous values calculated in other NLBs during Newton iteration, thus, the simplified nonlinear equations are obtained as

$$\overline{\boldsymbol{f}}^m(\boldsymbol{\nu}^m, \boldsymbol{\chi}^m) = 0. \tag{4.43}$$

The Jacobian matrix for the mth NLB can be calculated by the partial derivative of (4.43) for $\boldsymbol{\nu}^m$ and $\boldsymbol{\chi}^m$, given as

$$\boldsymbol{J}^{m(n+1)} = \frac{\partial \overline{\boldsymbol{f}}^m(\boldsymbol{\nu}^{m(n)}, \boldsymbol{\chi}^{m(n)})}{\partial [\boldsymbol{\nu}^m, \boldsymbol{\chi}^m]}. \tag{4.44}$$

Since the Jacobian matrix for each nonlinear block only relates to the variables of itself and does not couple with others, the large Jacobian matrix of NLS is partitioned into M small blocks, which can be processed in parallel on GPU. Therefore, the Newton equations of NLB^m are given as

$$\boldsymbol{J}^m \Delta[\boldsymbol{\nu}^m, \boldsymbol{\chi}^m] = \overline{\boldsymbol{f}}^m. \tag{4.45}$$

From solving (4.45), all connection voltages, $\boldsymbol{\nu}$, can found by combined the solution of each nonlinear block with (4.39). And then, the connection currents, $\boldsymbol{\iota}$, can also be obtained by (4.38) and (4.37) for the next iteration. In contrast to the traditional NR method, the connection node voltages and currents of all blocks are interchanged with others to update the Jacobian matrix for the next iteration according to domain decomposition theory [44, 45]. The final solution for the whole nonlinear subsystem can be found when all nonlinear blocks get converged.

Comparing to the original Newton-Raphson iteration method, although the rate of convergence of decomposed Newton iteration will be slightly impacted by the data exchange between nonlinear blocks, the computational speed will be significantly enhanced by parallel computation, especially when system size becomes large. Obviously, the number of connections among nonlinear blocks is the point influencing the convergence, therefore, choosing a path with less connections to partition a large system into blocks is always an optimization policy. When the Jacobian matrix is near ill-conditioned due to the high-frequent solution and transient, the decomposition can limit the scope of computation, reduce the accumulation of calculation error and increase the convergence of simulation when the scale of nonlinear system becomes large, which will be shown on the application of simulation MMC AC/DC converter based on nonlinear device-level model in Chapter 5.

4.3 Summary

In this chapter, the multi-level shattering network decomposition is introduced, of which the first level is coarse-grained based on propagation delay, and the second level is fine-grained for linear and nonlinear solution respectively. The purpose of the decomposition is to divide the large system into subsystems relatively small, which can optimally fit to the SIMT execution features and release the computational power of GPU as possible. In the fine-grained level, the compensation network decomposition is proposed for linear subsystem, which partitions the linear subsystem into linear blocks and connection network. The solution of linear subsystem is obtained by solving linear blocks in parallel and compensate the results with the solution of connection network. The Jacobian domain decomposition decouples the Jacobian matrix based on domain decomposition method to accelerate the solution of Newton equations, which parallelizes the calculation and enhances the convergence.

5 Power Electronic Circuit Decomposition

Different from conventional components in power electrical system, power electronic devices, such as diode, MOSFET and IGBT, which are widely used in renewable energy, smart grid and modern transport, have intensive nonlinearity and high-frequency transient [46–48]. Growing with the complicate structure of power electronic circuit, the challenge for device-level simulation tool is not only the complexity of modeling, but also the increasing circuit size. Mainstream commercial simulation tools, such as SaberRD®, Orcad®, PSIM® and etc., provide plenty of support for device-level simulation [49–53] with various models. However, the single-thread execution or limited coarse-grained parallelism hampers the computational performance of above simulators seriously. Especially in system-level simulation, the model of power electronic device is downgraded to linear and behavior level, which covers important information in EMT simulation. The GPU executing in SIMT is a candidate, that can effectively enhances the simulation performance with its massive cores and data transfer bandwidth, as long as the simulation procedure of power electronic system can be decoupled into fine-grained level. As a typical voltage source converter in HVDC grid, the modular multi-level converter (MMC) has a strongly coupled relation among its power electronic devices, such as IGBTs, in its complicate structure. When a device-level nonlinear detailed model is applied, the situation is getting worse. It is hard to break down the relation of devices in MMC by the circuit structure. Fortunately, the shattering decomposition proposed in this work shows itself to be of great benefit to alleviate the contradiction by decoupling the solution method, so that

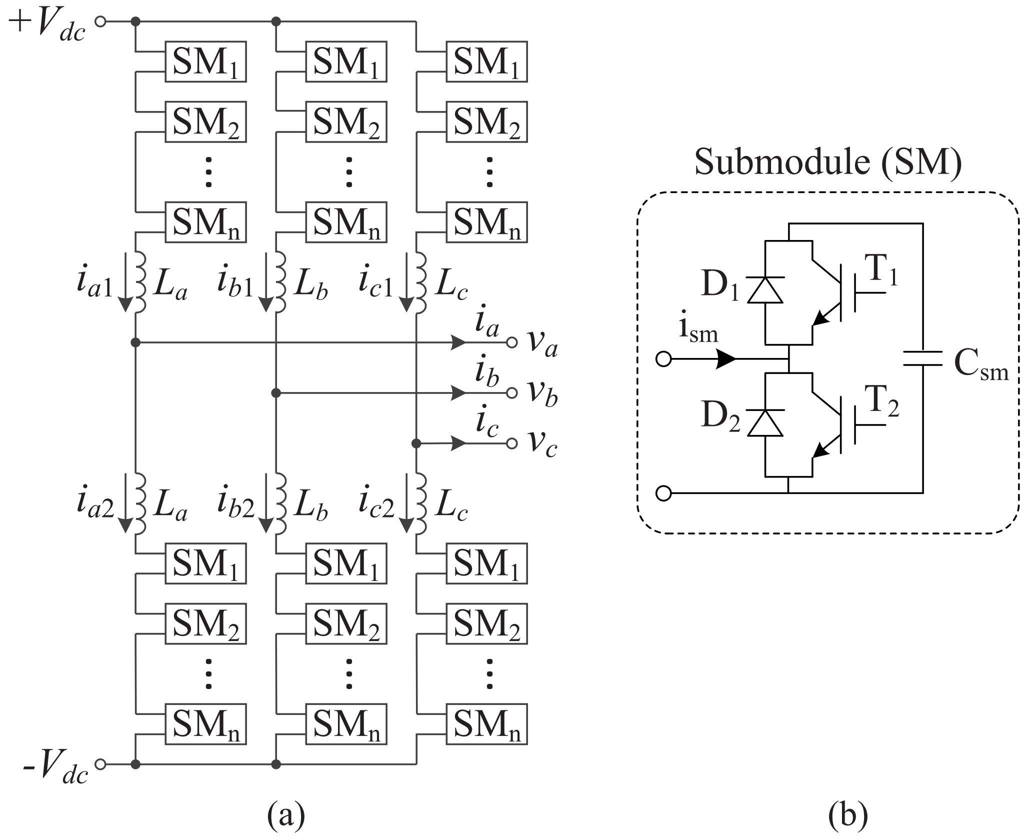


Figure 5.1: MMC circuit structure.

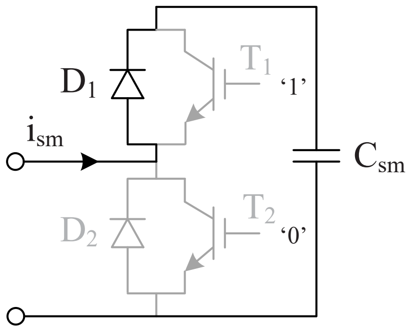
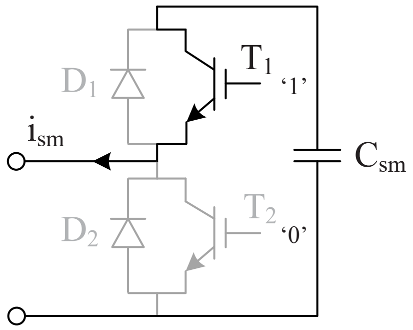
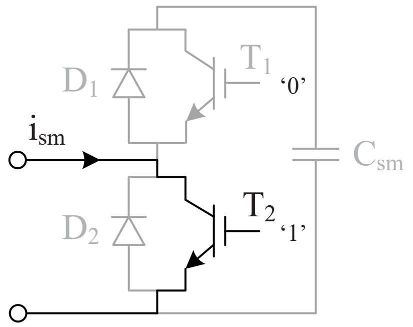
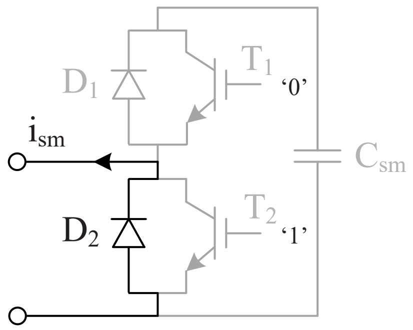
the EMT simulation of power electronic circuit can be accelerated by GPU-based massively parallel computing platform.

5.1 Modular Multi-Level Converter (MMC) and Control Strategy

5.1.1 Circuit Structure

Popularly applied in HVDC systems, the 3-phase modular multi-level converter (MMC) circuit with cascade structure, as shown in 5.1(a), consists of a series of half-bridge sub-modules (SMs) in each arm. Fig. 5.1(b) shows the internal devices and connection of an SM, which contains two IGBT-Diode units paralleling with a capacitor, C_{SM} which stores the energy transferring between AC and DC. Based on the gate signal combination and current direction of each IGBT, SMs have different operating states. As listed in Table 5.1, once T_1 gate has on signal and T_2 gate has off signal, C_{SM} will be charged and discharged according to the direction of i_{SM}; and when T_1 gate has off signal and T_2 gate has on signal, C_{SM} will be uncharged regardless the current direction. Especially, when the gate signal

Table 5.1: Capacitor states of SM

T_1	T_2	i_{SM}	Capacitor State	SM Operation
1	0	>0	Charging	
1	0	<0	Discharging	
0	1	>0	Bypass	
0	1	<0	Bypass	

combination is ’00’, the SM is blocked; and the gate signal combination ’11’ will cause the short circuit of the capacitor. Thus, neither of those two signal combinations is used in normal operation.

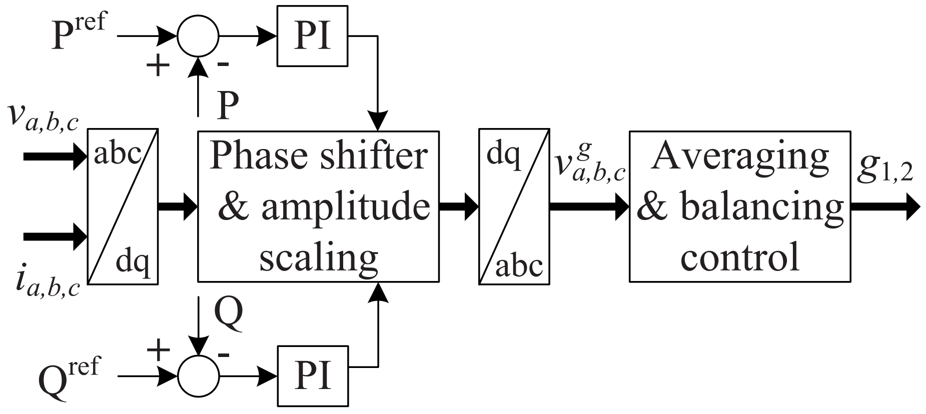


Figure 5.2: MMC control strategy.

The MMC circuit has been modeled by different types of simulation models including equivalent circuit base model, arm switching function and average value model [58–61]. Hefner's physics-based IGBT model is brought up as the first complete analytical model available for circuit simulator, which is implemented in SaberRD® [55]. Based on Hefner's model, IGBT is described as the combination of nonlinear current sources and capacitors, that extends the circuit structure by introducing extra internal nodes. [56] Due to the most detailed information inside every device and the nanosecond level time-step, the physics-based model requires so much computational requirement that it is uncommon in simulation [62]. However, the application of GPU with fined-grained decomposition makes it feasible in both accuracy and speed.

5.1.2 Control strategy

The control strategies of MMC circuit include the active and reactive power control, modulation signal phase shifter and amplitude scaling, and capacitor voltage averaging and balancing control [63–65]. The outer-layer in Fig. 5.2 is the active and reactive power control. Comparing the instance power and the reference power, the difference of active and reactive power is inputted to PI controllers. Combining the 3-phase voltages and currents, phase shifter and amplitude scaling module creates the 3-phase reference voltages for modulation. The outputs of the capacitor voltage averaging and balancing control module are the gate signals, g_1 and g_2, used to control the IGBTs, T_1 and T_2, in the SM respectively. Since the gate signals are necessary for switching in each time-step, the sample rate for the control logic of MMC is the same with the time-step of the network solution.

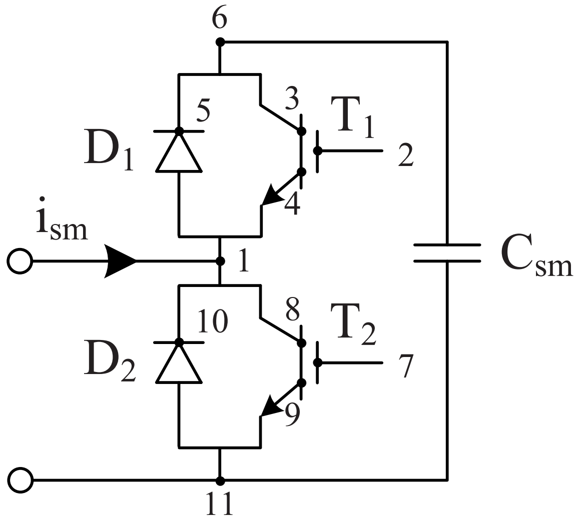


Figure 5.3: Node order of SM.

5.2 Decomposition for Nonlinear Modeling MMC

Since the MMC circuit is difficult to be partitioned from its physical structure due to the strongly coupled serial SM connection, the solution method of nonlinear modeling MMC is considered for the decomposition. As the basic solving method for nonlinear system is Newton-Raphson iterative method, by which to solve the nonlinear system is converted to solve the same order linear system multiple times to find the closest approximate root, the proposed Jacobian domain decomposition method can be applied to break down the relevance during Newton iteration.

Without loss of generality, the MMC circuit modeled by the Hefner's physics-based IGBT model [55,56], accompanying nonlinear power diode model [57], is taken as an example, which are detailed in Appendix A. Since the diode has 3 nodes including one internal node, the conductance matrix of the linerized device, G^{Diode}, is 3×3 according to (A.16), denoted as

$$\boldsymbol{G}^{Diode} = \begin{bmatrix} G^D_{11} & G^D_{12} & G^D_{13} \\ G^D_{21} & G^D_{22} & G^D_{23} \\ G^D_{31} & G^D_{32} & G^D_{33} \end{bmatrix}. \tag{5.1}$$

And the IGBT is modeled in 5 nodes equivalent circuit, as shown in Fig. A.3, with a 5×5 conductance matrix from (A.41) after linearization, denoted as

$$\boldsymbol{G}^{IGBT} = \begin{bmatrix} G^I_{11} & G^I_{12} & G^I_{13} & G^I_{14} & G^I_{15} \\ G^I_{21} & G^I_{22} & G^I_{23} & G^I_{24} & G^I_{25} \\ G^I_{31} & G^I_{32} & G^I_{33} & G^I_{34} & G^I_{35} \\ G^I_{41} & G^I_{42} & G^I_{43} & G^I_{44} & G^I_{45} \\ G^I_{51} & G^I_{52} & G^I_{53} & G^I_{54} & G^I_{55} \end{bmatrix}. \tag{5.2}$$

The node order of SM, which combines two Diode-IGBT units, is shown in Fig. 5.3, where there are 11 nodes in total. Since diodes and IGBTs are connected in SM, the node IDs of

Table 5.2: Node mapping of Diodes and IGBTs in SM

	1	2	3	4	5	6	7	8	9	10	11
D_1	1				2	3					
D_2	3									2	1
T_1	1	2	3	4		5					
T_2	5						2	3	4		1

new conductance matrix of SM must be remapped according to the structure of SM, which is listed in Table. 5.2.

The linear system with Jacobian matrix before decomposition in Newton method is given as,

$$\boldsymbol{J}^{o(n)}_{SM} \cdot \Delta\boldsymbol{v}^{(n+1)} = -\boldsymbol{F}^{o(n)}, \tag{5.3}$$

where n denotes the previous step and $n+1$ represents the next step. The topology of Jacobian matrix of a SM applying Hefner's physics-based IGBT model is given as,

$$\boldsymbol{J}^{o}_{SM} = \begin{bmatrix} J_{1,1} & J_{1,2} & J_{1,3} & J_{1,4} & J_{1,5} & J_{1,6} & J_{1,7} & J_{1,8} & J_{1,9} & J_{1,10} & J_{1,11} \\ J_{2,1} & J_{2,2} & J_{2,3} & J_{2,4} & J_{2,5} & J_{2,6} & & & & & \\ J_{3,1} & J_{3,2} & J_{3,3} & J_{3,4} & J_{3,5} & J_{3,6} & & & & & \\ J_{4,1} & J_{4,2} & J_{4,3} & J_{4,4} & J_{4,5} & J_{4,6} & & & & & \\ J_{5,1} & J_{5,2} & J_{5,3} & J_{5,4} & J_{5,5} & J_{5,6} & & & & & \\ J_{6,1} & J_{6,2} & J_{6,3} & J_{6,4} & J_{6,5} & J_{6,6} & & & & & \boxed{J_{6,11}} \\ J_{7,1} & & & & & & J_{7,7} & J_{7,8} & J_{7,9} & J_{7,10} & J_{7,11} \\ J_{8,1} & & & & & & J_{8,7} & J_{8,8} & J_{8,9} & J_{8,10} & J_{8,11} \\ J_{9,1} & & & & & & J_{9,7} & J_{9,8} & J_{9,9} & J_{9,10} & J_{9,11} \\ J_{10,1} & & & & & & J_{10,7} & J_{10,8} & J_{10,9} & J_{10,10} & J_{10,11} \\ J_{11,1} & & & & & \boxed{J_{11,6}} & J_{11,7} & J_{11,8} & J_{11,9} & J_{11,10} & J_{11,11} \end{bmatrix} \tag{5.4}$$

according to the node mapping of SM [66, 67]. Since the idea of shattering decomposition is based on the topology of the system, only the pattern of Jacobian matrix are concerned. Jacobian matrix of one SM is 11 by 11, partial sparse and close to block diagonal except for few elements. It can be noticed that the matrix will be in border block diagonal if elements, $J_{6,11}$ and $J_{11,6}$ marked in (5.4), are eliminated.

5.2.1 Matrix trim based on relaxation domain decomposition

Original Jacobian matrix J^o_{SM} can be reshaped by eliminating two elements, $J_{6,11}$ and $J_{11,6}$, using the relaxation algorithm. Their influence to the linear system can be approximated as known values using $J_6 \Delta v^n_6$ and $J_{11} \Delta v^n_{11}$ and inject to the RHS. After the transformation, the linear system (5.3) is updated to

$$\boldsymbol{J}^{(n)}_{SM} \cdot \Delta \boldsymbol{v}^{(n+1)} = -\boldsymbol{F}^{(n)}, \tag{5.5}$$

where the updated Jacobian matrix of one SM is

$$\boldsymbol{J}_{SM} = \begin{bmatrix} J_{1,1} & J_{1,2} & J_{1,3} & J_{1,4} & J_{1,5} & J_{1,6} & J_{1,7} & J_{1,8} & J_{1,9} & J_{1,10} & J_{1,11} \\ J_{2,1} & J_{2,2} & J_{2,3} & J_{2,4} & J_{2,5} & J_{2,6} & & & & & \\ J_{3,1} & J_{3,2} & J_{3,3} & J_{3,4} & G_{3,5} & J_{3,6} & & & & & \\ J_{4,1} & J_{4,2} & J_{4,3} & J_{4,4} & G_{4,5} & J_{4,6} & & & & & \\ J_{5,1} & J_{5,2} & J_{5,3} & J_{5,4} & G_{5,5} & J_{5,6} & & & & & \\ J_{6,1} & J_{6,2} & J_{6,3} & J_{6,4} & G_{6,5} & J_{6,6} & & & & & \\ J_{7,1} & & & & & & J_{7,7} & J_{7,8} & J_{7,9} & J_{7,10} & J_{7,11} \\ J_{8,1} & & & & & & J_{8,7} & J_{8,8} & J_{8,9} & J_{8,10} & J_{8,11} \\ J_{9,1} & & & & & & J_{9,7} & J_{9,8} & J_{9,9} & J_{9,10} & J_{9,11} \\ J_{10,1} & & & & & & J_{10,7} & J_{10,8} & J_{10,9} & J_{10,10} & J_{10,11} \\ J_{11,1} & & & & & & J_{11,7} & J_{11,8} & J_{11,9} & J_{11,10} & J_{11,11} \end{bmatrix}, \tag{5.6}$$

and the RHS vector $\boldsymbol{F}^{o(n)}$ is the updated to $\boldsymbol{F}^{(n)}$, given as

$$\boldsymbol{F}^{(n)} = \boldsymbol{F}^{o(n)} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ J_{11,6} \Delta v^{(n)}_6 \\ 0 \\ 0 \\ 0 \\ 0 \\ J_{6,11} \Delta v^{(n)}_{11} \end{bmatrix}, \tag{5.7}$$

whose 6th and 11th elements are merged by $J_{11,6}\Delta v_6^n$ and $J_{6,11}\Delta v_{11}^n$ with known values in previous iteration respectively. Thus, the original Jacobian matrix can be divided into multiple blocks so that the simulation of each Diode-IGBT unit can be computed independently for parallelism.

In normal relaxation domain decomposition algorithm, there should be an extra outer Gauss-Jacobi loop to converge the solution. In this case, however, the outer loop can be skipped since the relaxation approximation is inside Newton iteration. When the solution of Newton method gets converged,

$$\Delta \boldsymbol{v}^{n+1} = \Delta \boldsymbol{v}^{n} \to 0. \tag{5.8}$$

The solution satisfies (5.5) must also satisfies the original linear system (5.3), that guarantees the solution of nonlinear system is converged. Thus, an updated Jacobian matrix in border block diagonal is obtained, which can be partitioned into smaller blocks to apply the GPU-based parallel solving algorithm.

5.2.2 Partial LU Decomposition for MMC

For the MMC circuit with K SMs, the cascades structure is created by connecting node 11 of an SM to node 1 of the next one. Therefore, the pattern of Jacobian matrix J_{MMC} is shown in Fig. 5.4(a), where the 5×5 middle matrices are named as $\boldsymbol{A}_{2k-1}$ and $\boldsymbol{A}_{2k}$; the 1×5 horizontal border vectors are denoted as $\boldsymbol{c}_{2k-1}$ and $\boldsymbol{c}_{2k}$; the 5×1 vertical border vectors are named as $\boldsymbol{d}_{2k-1}$ and $\boldsymbol{d}_{2k}$; and the overlap element is denoted as e_k $(k = 1, 2, \cdots, K)$. In order to decompose the Newton iteration equation (5.5) and solve it avoiding its sparsity, the last elements of border vectors, $\boldsymbol{c}$ and $\boldsymbol{d}$, which are $J_{1,11}$ and $J_{11,1}$ in (5.6), need to be trimmed by relaxation method, as shown in Fig. 5.4(b), where the $\boldsymbol{c}_{2k}$ and $\boldsymbol{d}_{2k}$ are updated as $\boldsymbol{c}'_{2k}$ and $\boldsymbol{d}'_{2k}$ by trimming the last elements. Thus, the Newton iteration equation of MMC, given as

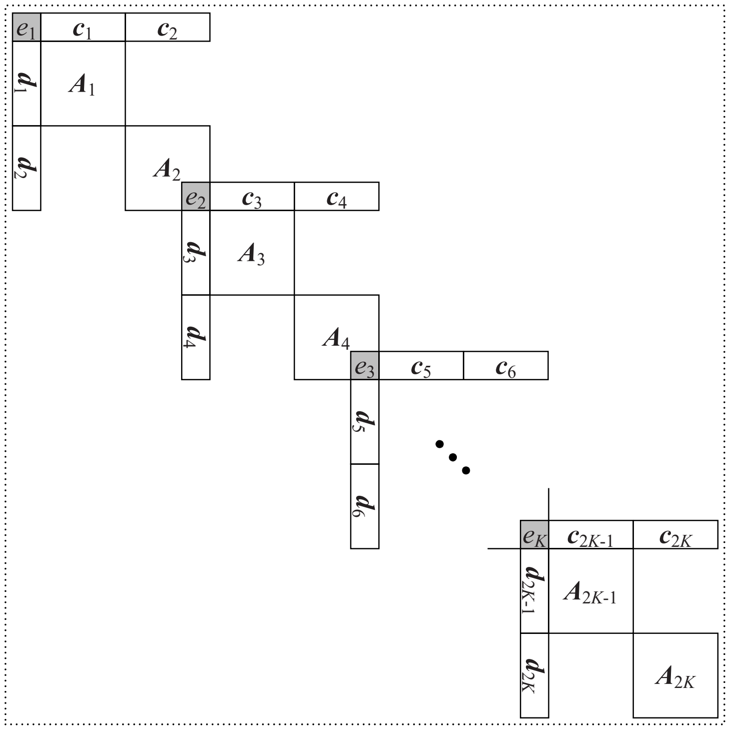
$$\boldsymbol{J}_{MMC}^{(n)} \cdot \Delta \boldsymbol{v}^{(n+1)} = -\boldsymbol{F}^{(n)}, \tag{5.9}$$

is updated as

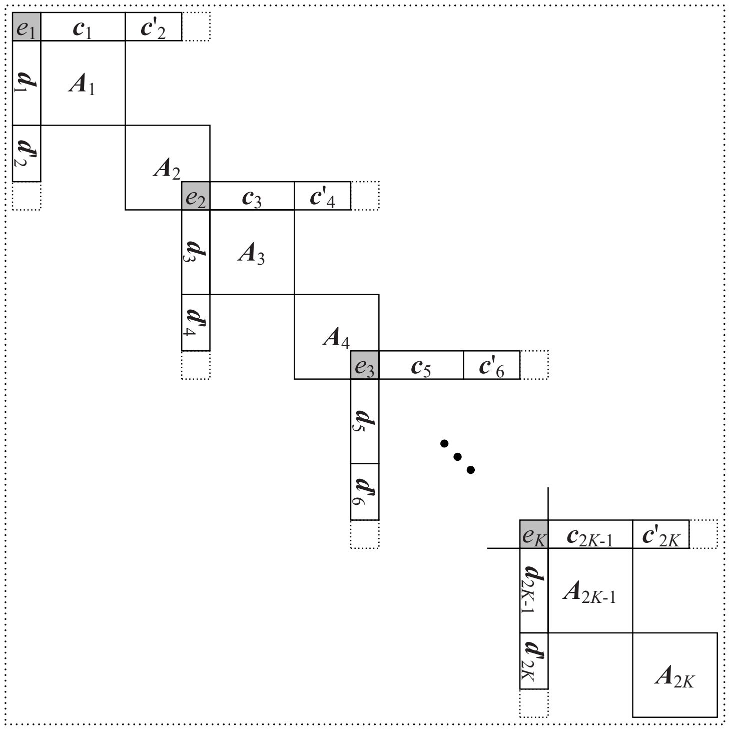
$$\boldsymbol{J}_{MMC}^{*(n)} \cdot \Delta \boldsymbol{v}^{(n+1)} = -\boldsymbol{F}^{*(n)}, \tag{5.10}$$

where the trimmed elements in LHS Jacobian matrix, $\boldsymbol{J}_{MMC}$, are merged into RHS vector, $\boldsymbol{F}$, with known values in previous iteration to obtain the updated RHS vector, $\boldsymbol{F}^*$, of which The kth block RHS vector is given as

$$\begin{bmatrix} \boldsymbol{F}^*_{2k-1} \\ \boldsymbol{F}^*_{2k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{2k-1} \\ \boldsymbol{F}_{2k} \end{bmatrix} + \left[(d^5_{2k})\Delta v_1^{(n)}, 0, 0, 0, 0, 0, 0, 0, 0, (c^5_{2k+2})\Delta v_{10}^{(n)} \right]^T. \tag{5.11}$$



(a) Jacobian matrix J_{MMC} of MMC circuit



(a) Updated Jacobian matrix J^{*}_{MMC} with relaxation

Figure 5.4: MMC Jacobian matrix trim.

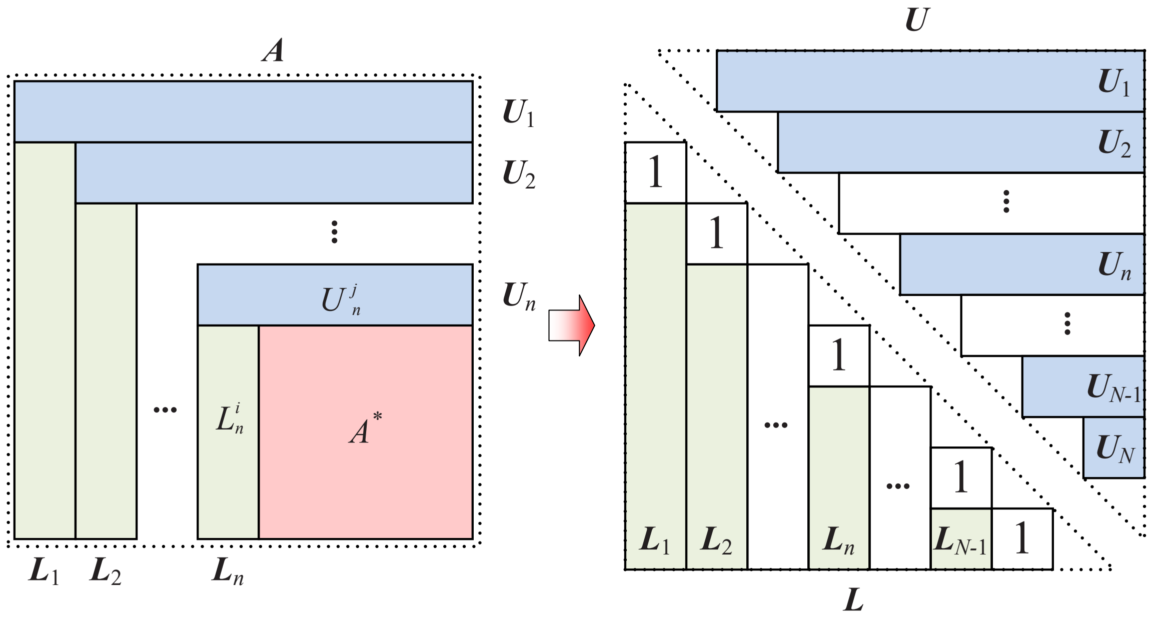


Figure 5.5: LU factorization.

After reshaping the Jacobian matrix in (5.10) by relaxation method, the LU factorization of the reshaped matrix can be processed in block-level parallel since the computation is decoupled according to the structure of $\boldsymbol{J}^*_{MMC}$.

Firstly, for all A matrices in $\boldsymbol{J}^*_{MMC}$, the LU factorization is processed first, we get

$$\boldsymbol{L}_l \cdot \boldsymbol{U}_l = \boldsymbol{A}_l \qquad (l = 1, 2, \cdots, 2K). \tag{5.12}$$

As shown in Fig. 5.5, the column vectors $\boldsymbol{L}_n$ and $\boldsymbol{u}_n$ are calculated in order of 1 to N for an $N \times N$ matrix $\boldsymbol{A}$. The i element of $\boldsymbol{L}_n$ is calculated as

$$L_n^i = \frac{A_{in}}{A_{nn}} \qquad i \in [n+1, N]; \tag{5.13}$$

the j element of $\boldsymbol{u}_n$ is given as

$$U_n^i = A_{nj} \qquad j \in [n, N]; \tag{5.14}$$

and the elements of residue matrix $\boldsymbol{A}^*$ is updated as

$$A_{ij}^i = A_{ij} - L_n^i U_n^i \qquad i, j \in [n+1, N]. \tag{5.15}$$

Combining all $\boldsymbol{L}_n$ column vectors and setting diagonal elements to '1' obtain the lower triangle matrix $\boldsymbol{L}$ of $\boldsymbol{A}$; similarly, the upper triangle matrix is composed of all $\boldsymbol{U}_n$ row vectors.

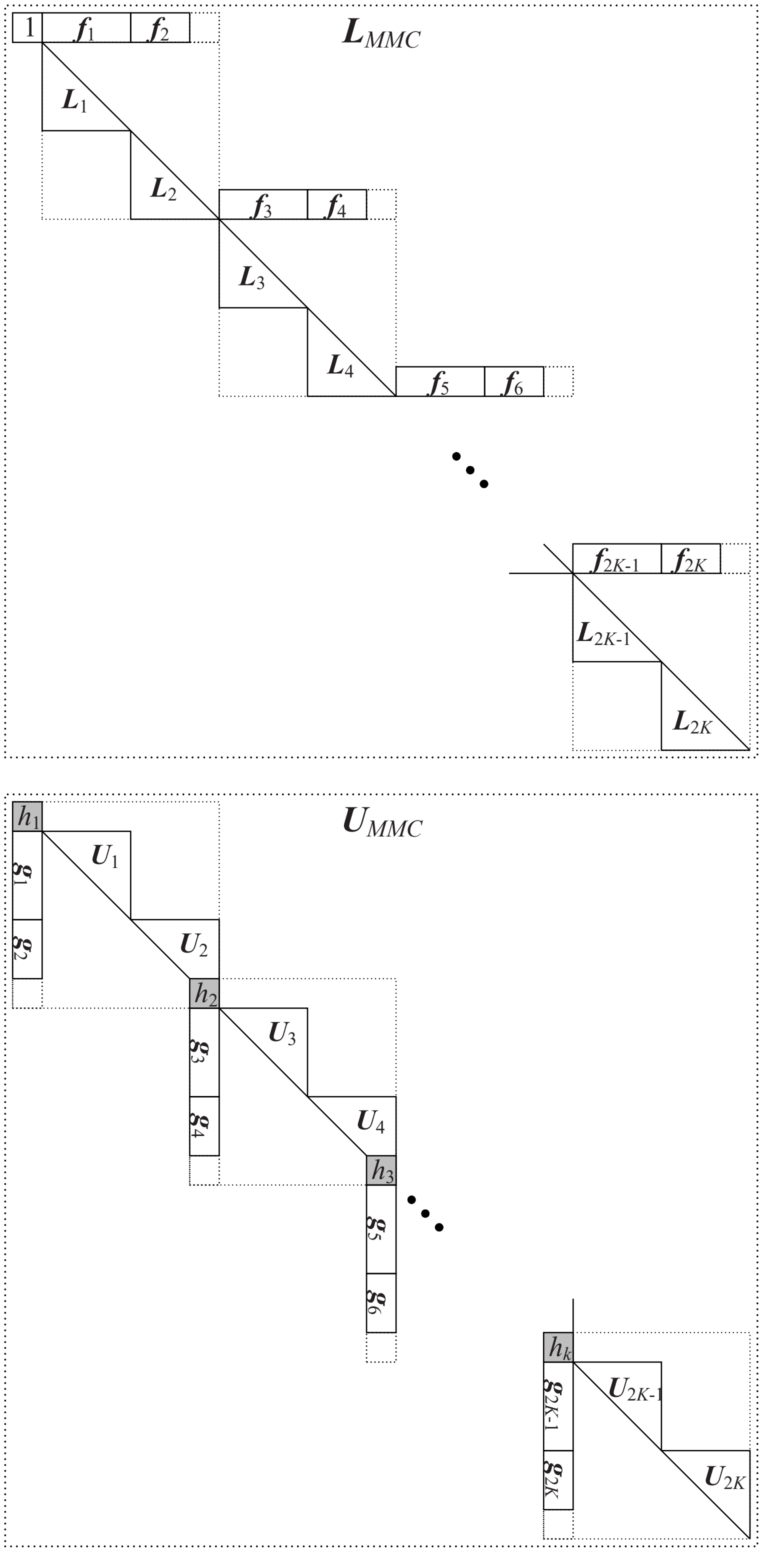


Figure 5.6: Partial LU decomposition for $\boldsymbol{J}^*_{MMC}$.

And then, the border vectors, $\boldsymbol{f}_j$ and $\boldsymbol{g}_j$ in Fig. 5.6, can be found according the relations in (5.16) and (5.17)

$$\boldsymbol{f}_l \cdot \boldsymbol{U}_l = \boldsymbol{c}_l \tag{5.16}$$

$$\boldsymbol{L}_l \cdot \boldsymbol{g}_l = \boldsymbol{d}_l \tag{5.17}$$

Since $\boldsymbol{f}_l$ and $\boldsymbol{c}_l$ are row vectors, and $\boldsymbol{U}_l$ is Upper triangular matrix, (5.16) gives

$$[f_l^1 \quad f_l^2 \quad \cdots \quad f_l^N] \begin{bmatrix} U_l^{11} & U_l^{12} & \cdots & U_l^{1N} \\ & U_l^{22} & \cdots & U_l^{2N} \\ & & \ddots & \vdots \\ & & & U_l^{NN} \end{bmatrix} = [c_l^1 \quad c_l^2 \quad \cdots \quad c_l^N]. \tag{5.18}$$

Therefore, $\boldsymbol{f}_l$ can be solved by forward substitution, given as.

$$f_l^n = \frac{c_l^n}{U_l^{nn}} - \sum_{i=1}^{n-1} f_l^i U_l^{in} \qquad n \in [1, N]. \tag{5.19}$$

where N is the size of $\boldsymbol{A}_l$. Similarly, (5.17) is expressed as

$$\begin{bmatrix} 1 & & & \\ L_l^{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ L_l^{N1} & L_l^{N2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} g_l^1 \\ g_l^2 \\ \vdots \\ g_l^N \end{bmatrix} = \begin{bmatrix} d_l^1 \\ d_l^2 \\ \vdots \\ d_l^N \end{bmatrix}, \tag{5.20}$$

with column vectors, $\boldsymbol{g}_l$ and $\boldsymbol{d}_l$, and lower triangle matrix, $\boldsymbol{L}_l$. Thus, $\boldsymbol{g}_l$ can also be solved by forward substitution, given as

$$g_l^n = d_l^n - \sum_{i=1}^{n-1} g_l^i L_l^{ni} \qquad n \in [1, N]. \tag{5.21}$$

Last, the elements at the connecting node, h_k $(k = 1, 2, \cdots, K)$ in Fig. 5.6, are calculated with e_i in Fig. 5.4, as

$$h_k = e_k - \boldsymbol{f}_{2k-1} \cdot \boldsymbol{g}_{2k-1} - \boldsymbol{f}_{2k} \cdot \boldsymbol{g}_{2k}, \tag{5.22}$$

which is expressed as

$$h_k = e_k - \begin{bmatrix} f_{2k-1}^1 & f_{2k-1}^2 & \cdots & f_{2k-1}^N \end{bmatrix} \begin{bmatrix} g_{2k-1}^1 \\ g_{2k-1}^2 \\ \vdots \\ g_{2k-1}^N \end{bmatrix} - \begin{bmatrix} f_{2k}^1 & f_{2k}^2 & \cdots & f_{2k}^N \end{bmatrix} \begin{bmatrix} g_{2k}^1 \\ g_{2k}^2 \\ \vdots \\ g_{2k}^N \end{bmatrix}, \tag{5.23}$$

where N is the size of vectors.

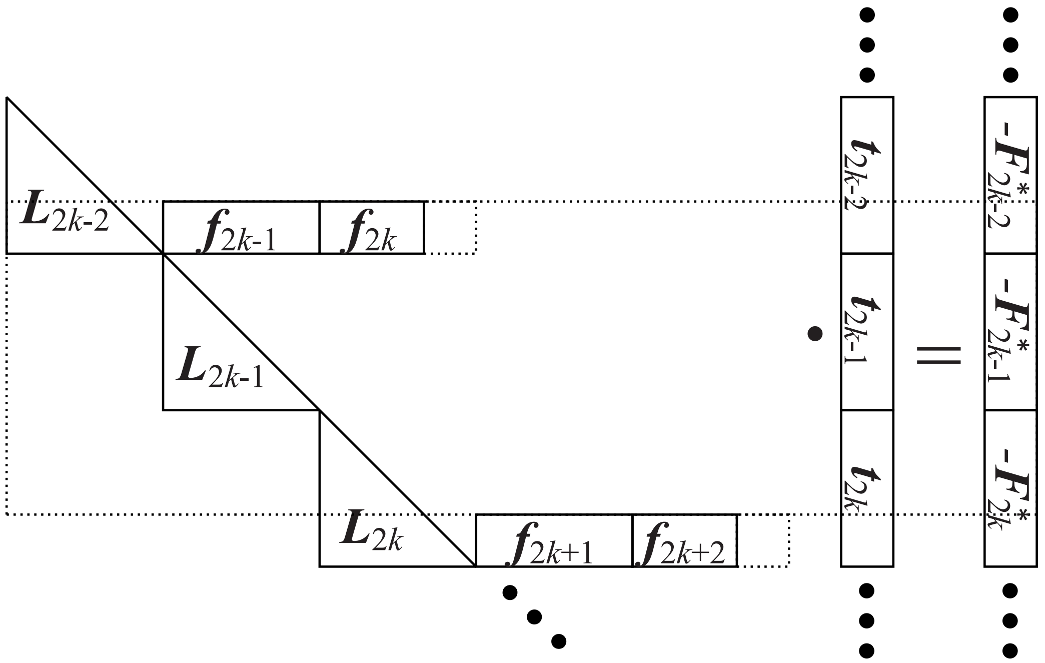


Figure 5.7: Blocked forward substitution.

So far, the Jacobian matrix, $\boldsymbol{J}^*_{MMC}$, is factorized into lower and upper matrices by the proposed partial LU decomposition, which can be computed in parallel since the decoupled structure of the matrix, including the LU factorization of $\boldsymbol{A}$; calculating border vectors, $\boldsymbol{f}$ and $\boldsymbol{g}$; and updating connecting node elements $\boldsymbol{h}$.

5.2.3 Blocked Forward and Backward Substitution

After obtaining the semi lower and upper triangular matrices utilizing partial LU factorization, we obtain the updated linear system of (5.10) as

$$\boldsymbol{L}^{(n)}_{MMC}\boldsymbol{U}^{(n)}_{MMC} \cdot \Delta\boldsymbol{v}^{(n+1)} = -\boldsymbol{F}^{*(n)}. \tag{5.24}$$

Defining

$$\boldsymbol{U}^{(n)}_{MMC} \cdot \Delta\boldsymbol{v}^{(n+1)} = \boldsymbol{t}, \tag{5.25}$$

where $\boldsymbol{t}$ are a temporary intermediate variables. Substituted by (5.25), (5.24) can be rewritten as

$$\boldsymbol{L}^{(n)}_{MMC} \cdot \boldsymbol{t} = -\boldsymbol{F}^{*(n)}. \tag{5.26}$$

Since $L^{(n)}_{MMC}$ is lower diagonal, (5.26) can be solved by blocked forward substitution, as shown in Fig. 5.7. Taking the kth block for example, $\boldsymbol{t}_{2k-1}$ can be solved directly by forward

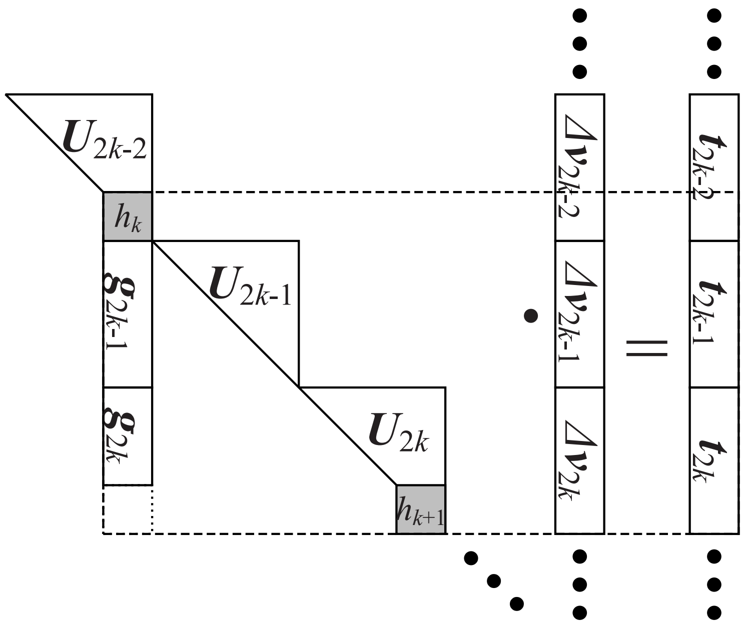


Figure 5.8: Blocked backward substitution.

substitution as

$$t^n_{2k-1} = -F^{*n}_{2k-1} - \sum_{i=1}^{n-1} t^i_{2k-1} L^{ni}_{2k-1} \qquad n \in [1, N], \tag{5.27}$$

where N is the order of L_{2k-1}. Meanwhile, $\boldsymbol{t}_{2k}$ except the last element, t^N_{2k}, can also be calculated by forward substitution as

$$t^n_{2k} = -F^{*n}_{2k} - \sum_{i=1}^{n-1} t^i_{2k} L^{ni}_{2k} \qquad n \in [1, N-1]. \tag{5.28}$$

The major part of $\boldsymbol{t}_{2k-2}$ is solved similarly. Afterward, the last element of $\boldsymbol{t}_{2k-2}$, t^N_{2k-2}, can be found with the solved results of $\boldsymbol{t}_{2k-2}$, $\boldsymbol{t}_{2k-1}$ and $\boldsymbol{t}_{2k}$ as

$$t^N_{2k-2} = -F^{*N}_{2k-2} - \sum_{i=1}^{N-1} t^i_{2k-2} L^{Ni}_{2k-2} - \sum_{i=1}^{N} t^i_{2k-1} f^i_{2k-1} - \sum_{i=1}^{N-1} t^i_{2k} f^i_{2k}. \tag{5.29}$$

At the same time, the last element of $\boldsymbol{t}_{2k}$ can also be calculated using the same method, given as

$$t^N_{2k} = -F^{*N}_{2k} - \sum_{i=1}^{N-1} t^i_{2k} L^{Ni}_{2k} - \sum_{i=1}^{N} t^i_{2k+1} f^i_{2k+1} - \sum_{i=1}^{N-1} t^i_{2k+2} f^i_{2k+2}. \tag{5.30}$$

Since all blocks are decoupled, $\boldsymbol{t}$ is solved by the blocked forward substitution in parallel. With solved $\boldsymbol{t}$, $\boldsymbol{v}^{(n+1)}$ in (5.25) can be found by blocked backward substitution, as shown in Fig. 5.8. The voltage different at connecting nodes, is calculated first as follows

$$\Delta v_{2k-2}^{N} = \frac{t_{2k-2}^{N}}{h_k} \qquad k \in [1, K], \tag{5.31}$$

and

$$\Delta v_{2k}^{N} = \frac{t_{2k}^{N}}{h_{k+1}} \qquad k \in [1, K], \tag{5.32}$$

where N is the order of $\boldsymbol{U}$. Then $\Delta \boldsymbol{v}_{2k-1}$ can be solved by backward substitution, given as

$$\Delta v_{2k-1}^{n} = \frac{t_{2k-1}^{n} - \Delta v_{2k-2}^{N} g_{2k-1}^{n} - \sum_{i=n+1}^{N} \Delta v_{2k-1}^{i} U_{2k-1}^{ni}}{U_{2k-1}^{nn}} \qquad n \in [N, 1]. \tag{5.33}$$

Simultaneously, the rest elements of $\Delta \boldsymbol{v}_{2k}$ can also calculated as

$$\Delta v_{2k}^{n} = \frac{t_{2k}^{n} - \Delta v_{2k-2}^{N} g_{2k}^{n} - \Delta v_{2k-2}^{N} U_{2k}^{nN} - \sum_{i=n+1}^{N} \Delta v_{2k}^{i} U_{2k}^{ni}}{U_{2k}^{nn}} \qquad n \in [N-1, 1]. \tag{5.34}$$

Finally, the results of (5.10) are solved in parallel. When the newton iteration of (5.10) is converged, the solution of nonlinear system, (5.9), gets converged as well.

In the MMC circuit, the size of Jacobian matrix grows with the number of output voltage levels. Instead of solving a system containing a $(10k+1) \times (10k+1)$ large Jacobian matrix, the 5×5 block perfectly accommodates the parallel scheme of GPU with its limited shared memory to reduce the data transmission cost.

5.3 Decomposition for Linear Modeling MMC

In system-level MMC circuit simulation, the linear behavior-based SM models are commonly adopted [68], where the IGBT-diode unit is represented as functional switching control resistor [69], as shown in Fig. 5.9, given as,

$$r_1 = \begin{cases} r_{on} & \text{if} \quad (g_1 = 1) \\ r_{off} & \text{if} \quad (g_1 = 0) \end{cases} \tag{5.35a}$$

$$r_2 = \begin{cases} r_{on} & \text{if} \quad (g_2 = 1) \\ r_{off} & \text{if} \quad (g_2 = 0) \end{cases} \tag{5.35b}$$

The capacitor, C_{SM}, in each SM is discretized into an equivalent resistor r_c, given as,

$$r_c = \frac{2\Delta t}{C}, \tag{5.36}$$

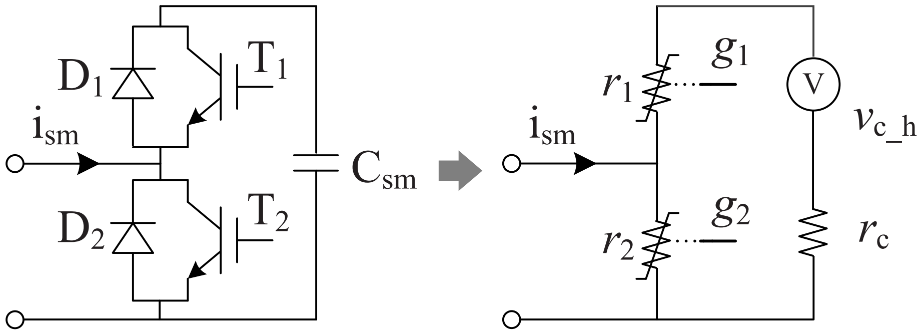


Figure 5.9: Linear behavior SM model based on functional switching.

in series with an equivalent history voltage source $v_{c\_h}(t-\Delta t)$, given as,

$$v_{c\_h}(t-\Delta t) = 2r_c i_c(t-\Delta t) - v_{c\_h}(t-2\Delta t), \tag{5.37}$$

by trapezoidal rule. The r_1 and r_2, are decided by the gate signals, g_1 and g_2, which are generated by the control logic, as shown in Fig. 5.2. In this way, each SM's Thevenin equivalent circuit in Fig. 5.9 contains an equivalent resistor r_{SM} and a history voltage source v_{SM}, given as

$$r_{SM} = \frac{r_2(r_1+r_c)}{r_1+r_2+r_c}, \tag{5.38}$$

$$v_{SM} = \frac{r_2}{r_1+r_2+r_c} v_{c\_h}(t-\Delta t). \tag{5.39}$$

Thus, each arm of MMC containing n SMs in Fig. 5.1 is represented by a voltage source and a resistor as

$$v_{arm} = \sum_{i=1}^{n} v_{SM}^i, \tag{5.40}$$

$$r_{arm} = \sum_{i=1}^{n} r_{SM}^i. \tag{5.41}$$

The arm current can be calculated by above arm equivalent model with (5.40) and (5.41) as

$$i_{SM} = \frac{v_{arm}}{r_{arm}}. \tag{5.42}$$

Since each SM's input current is the same as arm current, the node voltage inside each SM can be updated independently. Therefore, the solving process for each SM is natively decoupled, and the solution of MMC are computed in parallel with massive cores.

5.4 Summary

In this chapter, the fine-grained decomposition method is applied for the simulation of MMC AC/DC converter, which is modeled by both nonlinear physics-based model and linear behavior-based models. When Newton iteration method is applied to the nonlinear system, the Jacobian matrix of K-SM MMC circuit is strongly coupled. The proposed fine-grained decomposition method deriving from relaxation domain decomposition decouples the Jacobian matrix into border block diagram formation to adapting to the SIMT execution model of the GPU-based massively parallel computation. For the linear behavior-based model, each SM is represented as a node of functional switching controlled Thevenin equivalent resistor and voltage source. After the arm current is obtained, all nodes along the MMC arm can be solved independently, which satisfies the SIMT execution model of the GPU-based parallelism. Therefore, the proposed fine-grained decomposition method transform the MMC structure to a decoupled data topology effectively, which can fully release the computational power of GPU-based massively parallel computing platform on EMT simulation of power electronic system.

6 Massively Parallel Implementation on GPUs

The proposed fine-grained EMT simulation is implemented on a CPU-GPU heterogeneous platform, whose execution method is shown in Fig. 6.1. Considering the 64-bit double precision performance, which is used across the simulation, two Pascal microarchitecture (GP104) NVIDIA® GPUs are mounted into the Intel® Xeon® E-2620 server with 32 GB memory running Windows 7 Enterprise 64-bit OS.

6.1 EMT Simulation Implementation on GPUs

As shown in Fig. 6.2, the simulation starts with loading the netlist including the network connections and parameters, from which the topology of the network can be analyzed to find the boundaries of propagation delay for the first-level decomposition, such as transmission lines and control systems. The large network is then divided into subsystems by

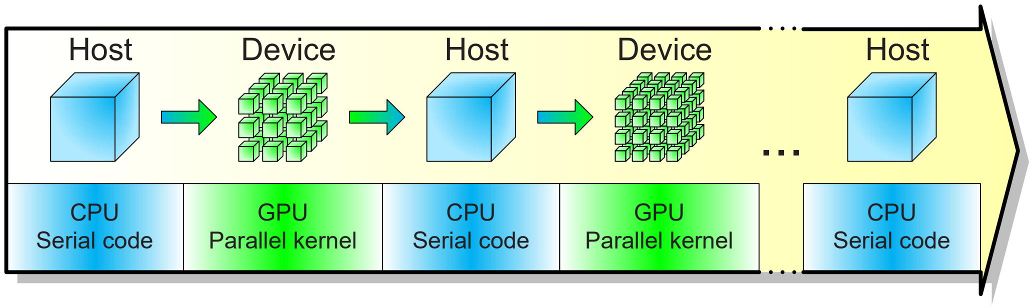


Figure 6.1: Heterogeneous CPU-GPU execution.

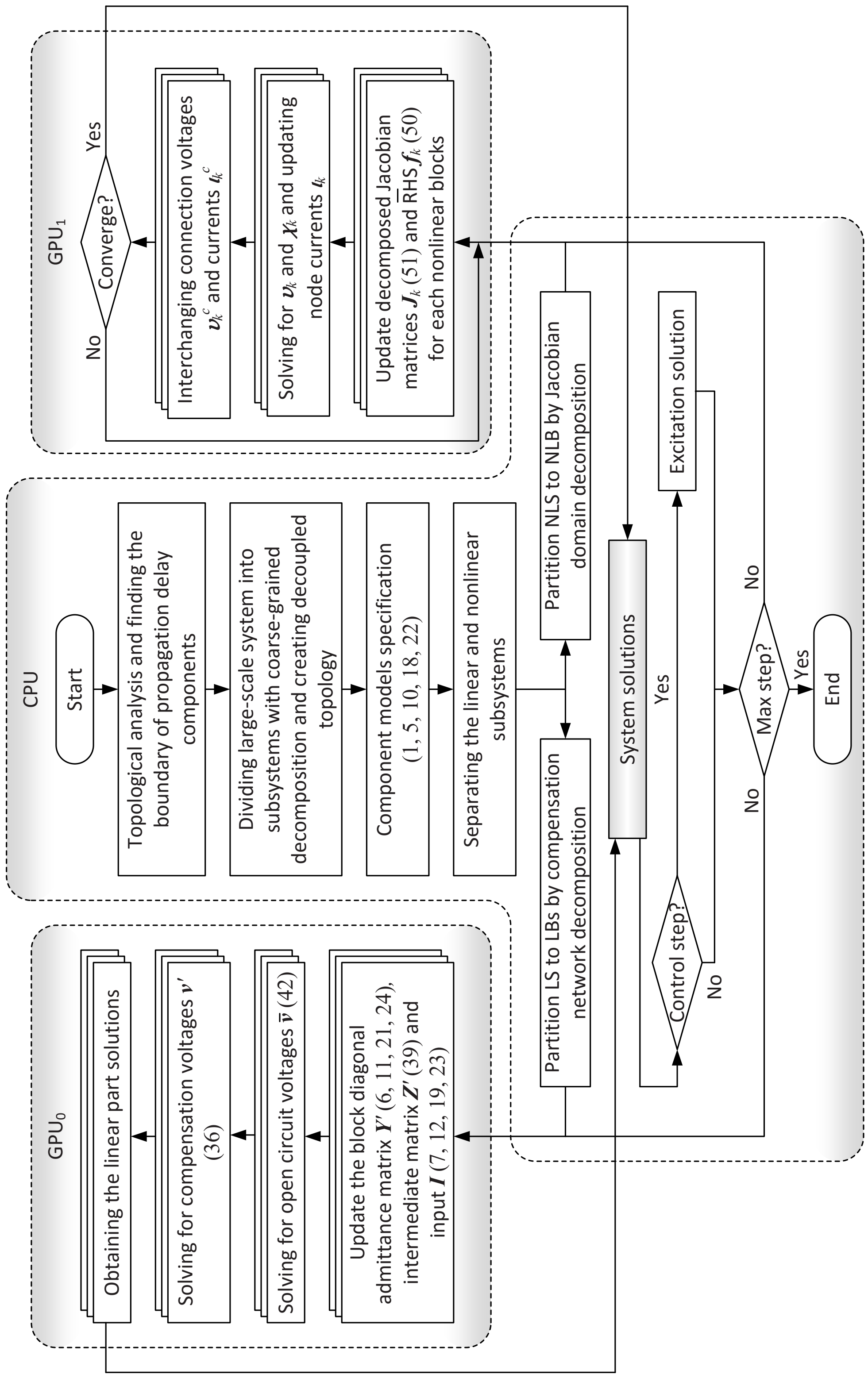

CPU
Start
Topological analysis and finding the boundary of propagation delay components
Dividing large-scale system into subsystems with coarse-grained decomposition and creating decoupled topology
Component models specification (1, 5, 10, 18, 22)
Separating the linear and nonlinear subsystems
Partition NLS to NLB by Jacobian domain decomposition
Partition LS to LBs by compensation network decomposition
System solutions
Excitation solution
Control step?
Max step?
Yes
No
End
GPU$_1$
Update decomposed Jacobian matrices $\boldsymbol{J}_k$ (51) and RHS $\boldsymbol{f}_k$ (50) for each nonlinear blocks
Solving for $\boldsymbol{v}_k$ and $\boldsymbol{\chi}_k$ and updating node currents $\boldsymbol{\iota}_k$
Interchanging connection voltages $\boldsymbol{v}_k^c$ and currents $\boldsymbol{\iota}_k^c$
Converge?
GPU$_0$
Update the block diagonal admittance matrix $\boldsymbol{Y}'$ (6, 11, 21, 24), intermediate matrix $\boldsymbol{Z}'$ (39) and input $\boldsymbol{I}$ (7, 12, 19, 23)
Solving for open circuit voltages $\bar{\boldsymbol{v}}$ (42)
Solving for compensation voltages $\boldsymbol{v}'$ (36)
Obtaining the linear part solutions

Figure 6.2: Fine-grained EMT simulation work flow.

coarse-grained decomposition, and the bus node system is also rebuilt according to the new topology. After separating linear and nonlinear subsystems, they are partitioned into small linear blocks and nonlinear blocks with fine-grained decomposition methods as described in Section 4.2. The bus node numbers have to be remapped again to guarantee the admittance, and the resulting Jacobian matrices will be block diagonal. At this time, all the detailed component models including frequency-dependent line model are specified, the data structures on both host (CPU) and device (GPU) are determined and all necessary data are transferred from the host to the devices when the entire simulation process is branched. One GPU is responsible for linear blocks and the other takes charge of nonlinear blocks. Every component model listed in Section 3 is represented by a parallel module, consisting a set of CUDA kernels, as well as solution methods, such as matrix operators, linear and nonlinear solvers [23]. According to the communication avoiding parallel theory in numerical linear algebra [70], in order to increase the register utilization inside each thread and minimize the data exchange between memories, the kernels are designed in small scale for limited register resource, the *for* loops are unrolled to make more data cached and individual thread work load is increased to extend the data lifetime inside the thread. Therefore, the per thread throughput is amplified, while the device occupancy per kernel is lowered, which can be compensated by kernel concurrent execution.

6.1.1 Linear Side

Algorithm 1 Linear parallel solution

```
for each LB do
    if any LB of $\boldsymbol{Y}'$ (3.9, 3.17, 3.34, 3.52) is updated then
        Invert $\boldsymbol{y}'_k$ for the updated LB
        update $\boldsymbol{Z}'$ (4.30)
    for each 3-phase bus node do
        update $\boldsymbol{i}$ with history terms (3.10, 3.18, 3.30, 3.53)
        solve the open circuit solution $\overline{\boldsymbol{v}}$ (4.13)
        calculate the compensation voltages $\boldsymbol{v}'$ (4.33)
    $\boldsymbol{v} \leftarrow \overline{\boldsymbol{v}} + \boldsymbol{v}'$
```

The component modules are applied to compose the admittance matrix $\boldsymbol{Y}'$ and initial inputs $\boldsymbol{i}$ in (4.7). Due to the independence of component modeling, all classified modules can run in parallel on the GPU. Since the optimized sparse admittance matrix $\boldsymbol{Y}'$ is decoupled, the open circuit linear solutions $\overline{\boldsymbol{v}}$ for all blocks run in parallel as well using (4.13). After all compensation voltages $\boldsymbol{v}'$ are solved by (4.33) at the same time, the solutions of

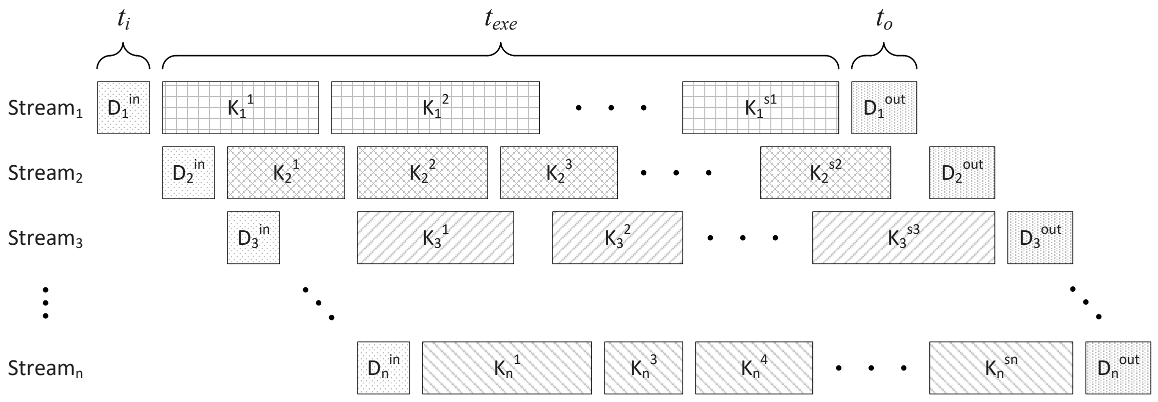


Figure 6.3: Concurrent execution with multiple streams.

linear blocks are found, which are then integrated to the solutions of the large system.

6.1.2 Nonlinear Side

Algorithm 2 Nonlinear parallel solution

```
repeat
    for each NLB do
        update RHS (4.43)
        update Jacobian matrix $\boldsymbol{J}$ (4.44)
        for each external bus node and internal node do
            solve for $\Delta\boldsymbol{\nu}_k$ and $\Delta\boldsymbol{\chi}_k$
            $\boldsymbol{\nu}_k^{(n+1)} \leftarrow \boldsymbol{\nu}_k^{(n)} + \Delta\boldsymbol{\nu}$
            $\boldsymbol{\chi}_k^{(n+1)} \leftarrow \boldsymbol{\chi}_k^{(n)} + \Delta\boldsymbol{\chi}$
        calculate currents $\boldsymbol{\iota}$ (4.40)
        update $\boldsymbol{\nu}^c$ and $\boldsymbol{\iota}^c$ by interchange
until
    $\mid \boldsymbol{\nu}^{(n+1)} - \boldsymbol{\nu}^{(n)} \mid < \epsilon$ and $\mid \boldsymbol{f}^{(n+1)} - \boldsymbol{f}^{(n)} \mid < \epsilon$
```

Since all nonlinear components are decoupled by Jacobian domain decomposition, the NR iterations are processed for all nonlinear blocks independently. The Jacobian matrices are composed by (4.44) with the updated nonlinear equations $\overline{\boldsymbol{f}}$ created during decomposition. When the next step voltages $\boldsymbol{\nu}^{m(n+1)}$ for each blocks are solved, the node currents $\boldsymbol{\iota}^{m(n+1)}$ can be obtained by the updated linkage functions $\overline{\boldsymbol{g}}$ in (4.40). They are interchanged among the blocks to update the connection voltages and currents for the next iteration. The parallel nonlinear solvers are synchronized when all solver loops are converged, and the results are sent to the host side as the other part of the system solution.

Combining the linear and nonlinear parts solutions, the large network system solutions for one time-step are found. Before approaching the next time-step, the synchronization of control system and EMT network is checked on the CPU, and if 'Yes', the control system solution will be calculated for the next EMT time-step. In order to parallelize the tasks with various algorithms that cannot be contained in a same kernel with different blocks, and cover the data transfer time between host and device, multiple streams are used to group independent kernels. As shown in Fig. 6.3, the dependent kernels, such as a set of kernels for a component module, are assigned to the same stream, which are executed in serial, while the kernels in different streams are independent without any data or procedure interference. Firstly, the data for each stream are copied from host to device costing t_i; then the set of kernels belonging to the stream are executed in t_{exe}; lastly, the results of the stream execution are copied back to host consuming t_o. If the following conditions for different hardware properties are satisfied,

$$t_{exe} > \begin{cases} max(t_i, t_o) & (two\ copy\ engine) \\ t_i + t_o & (one\ copy\ engines) \end{cases}, \tag{6.1}$$

The data transfer cost can be effectively covered only if $t_{exe} > (t_i + t_o)$, which is scheduled delicately. In addition, the execution of streams can also be concurrent when the GPU hardware still has enough resources available, which increases the overall occupation of GPU since the kernel are designed with low occupancy.

6.2 System Diagram

After system decomposition and discretization, the data structure, components modules, solution algorithms and input data are organized into an integrated system, as shown in Fig. 6.4. The Netlist and initial data carrying network topological information and components parameters are input into the parallel simulation system; then all component modules are created based on the EMT modeling for linear components, nonlinear components, transmission lines, power transformers, synchronous machines, power electronic device and control system; according to above modules of components, the solution algorithms are involved, such as time-domain discretization, matrix LU decomposition, forward/backward substitution, Newton-Raphson iteration and connecting network compensation, to compute all variables inside the EMT simulation. Between every time-steps, the results and intermediate values are exchanged with those of components modules and updated to solution algorithms. The time loop of the EMT simulation keeps iterating until the maximum simulation time is reached. Finally, the time-domain solutions of the power electrical system are obtained by collecting all results of each time-step.

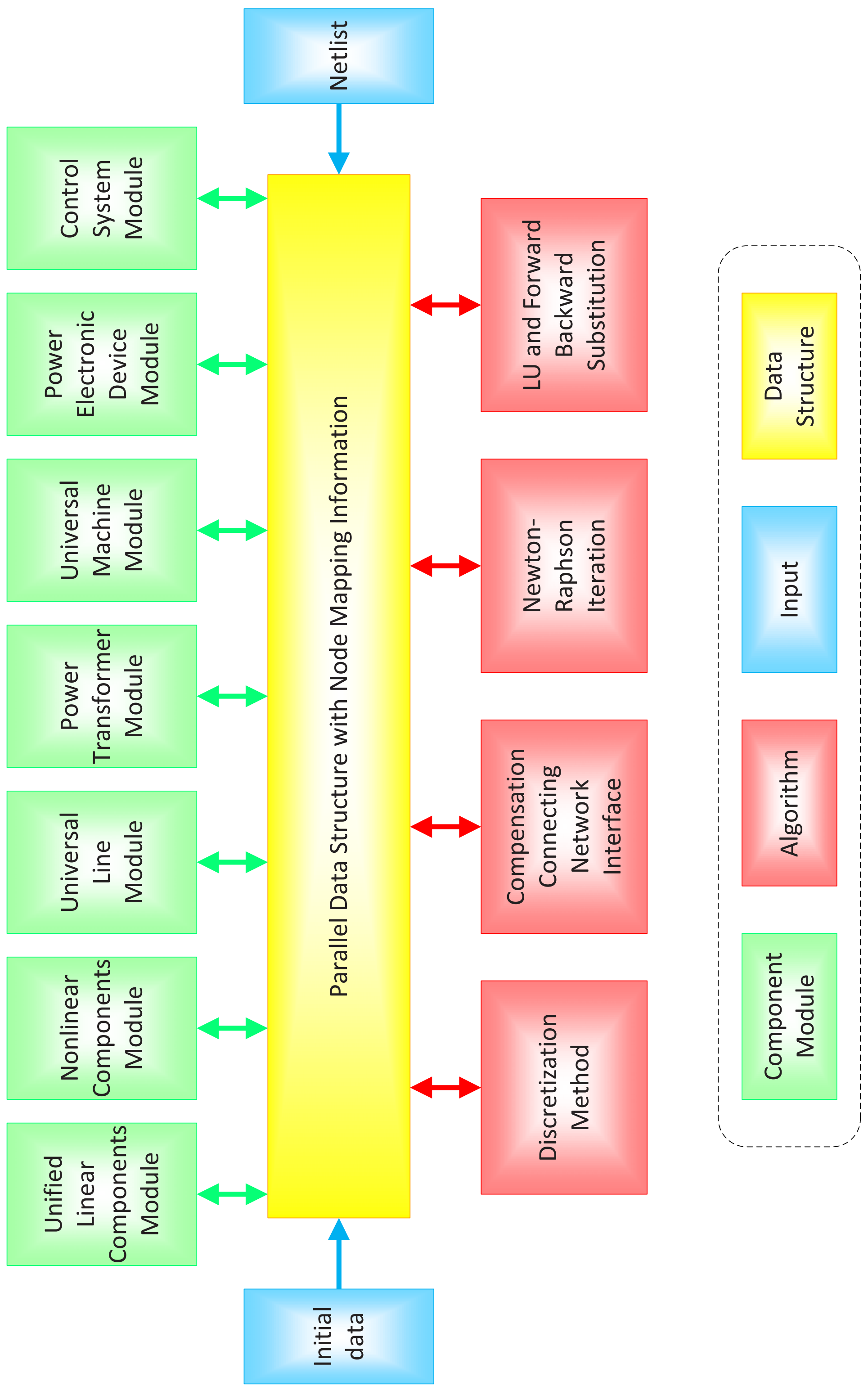

Netlist
Initial data
Parallel Data Structure with Node Mapping Information
Control System Module
Power Electronic Device Module
Universal Machine Module
Power Transformer Module
Universal Line Module
Nonlinear Components Module
Unified Linear Components Module
LU and Forward Backward Substitution
Newton-Raphson Iteration
Compensation Connecting Network Interface
Discretization Method
Data Structure
Input
Algorithm
Component Module

Figure 6.4: System diagram of massively parallel EMT simulator

6.3 Matrix LU Factorization and Inverse

In order to solve the linear system obtained by the node analysis method in EMT simulation after discretization, the LU factorization are applied to decompose the admittance matrix to Lower and Upper triangle matrices, in this work. The linear system built up by node analysis method is given as

$$\boldsymbol{Y}\boldsymbol{v} = \boldsymbol{i} \tag{6.2}$$

Applying LU factorization to $\boldsymbol{Y}$, we get

$$\boldsymbol{L}\boldsymbol{U}\boldsymbol{v} = \boldsymbol{i} \tag{6.3}$$

Define

$$\boldsymbol{U}\boldsymbol{v} = \boldsymbol{x} \tag{6.4}$$

and substitute to (6.3) can get

$$\boldsymbol{L}\boldsymbol{x} = \boldsymbol{i}. \tag{6.5}$$

Since $\boldsymbol{L}$ is a lower triangle matrix, the solution $\boldsymbol{x}$ can be get by forward substitution from top to bottom. After $\boldsymbol{x}$ is solved, the solution of the linear system, $\boldsymbol{v}$, can be found by backward substitution from bottom to top in (6.4) since $\boldsymbol{U}$ is an upper triangle matrix. Since the power electrical system is partitioned by shattering decomposition method, the admittance matrix created by the decomposed system is definitely decoupled into block diagram structure. The LU factorization for the whole large matrix is converted to the factorization for each small block, which can be processed in parallel on the GPU-based computing system.

Considering there are only few blocks of the admittance matrix influenced by the switching occurring in the power system since it is decoupled, the admittance matrix is relevant stable. Therefore, the linear system solution, $\boldsymbol{v}$, can also be obtained by the inversed matrix multiplying with the RHS currents, $\boldsymbol{i}$, more efficiently than doing forward-backward substitution every time. From the definition of matrix inverse,

$$\boldsymbol{Y}\boldsymbol{Y}^{-1} = \mathbf{I}, \tag{6.6}$$

where $\mathbf{I}$ is the identity matrix, $\boldsymbol{Y}^{-1}$ can be considered as a combination of the vector solution of the linear systems as follows,

$$\boldsymbol{Y}\boldsymbol{y}'_k = \mathbf{i}_k, \qquad k = 0, 1, \ldots, n \tag{6.7}$$

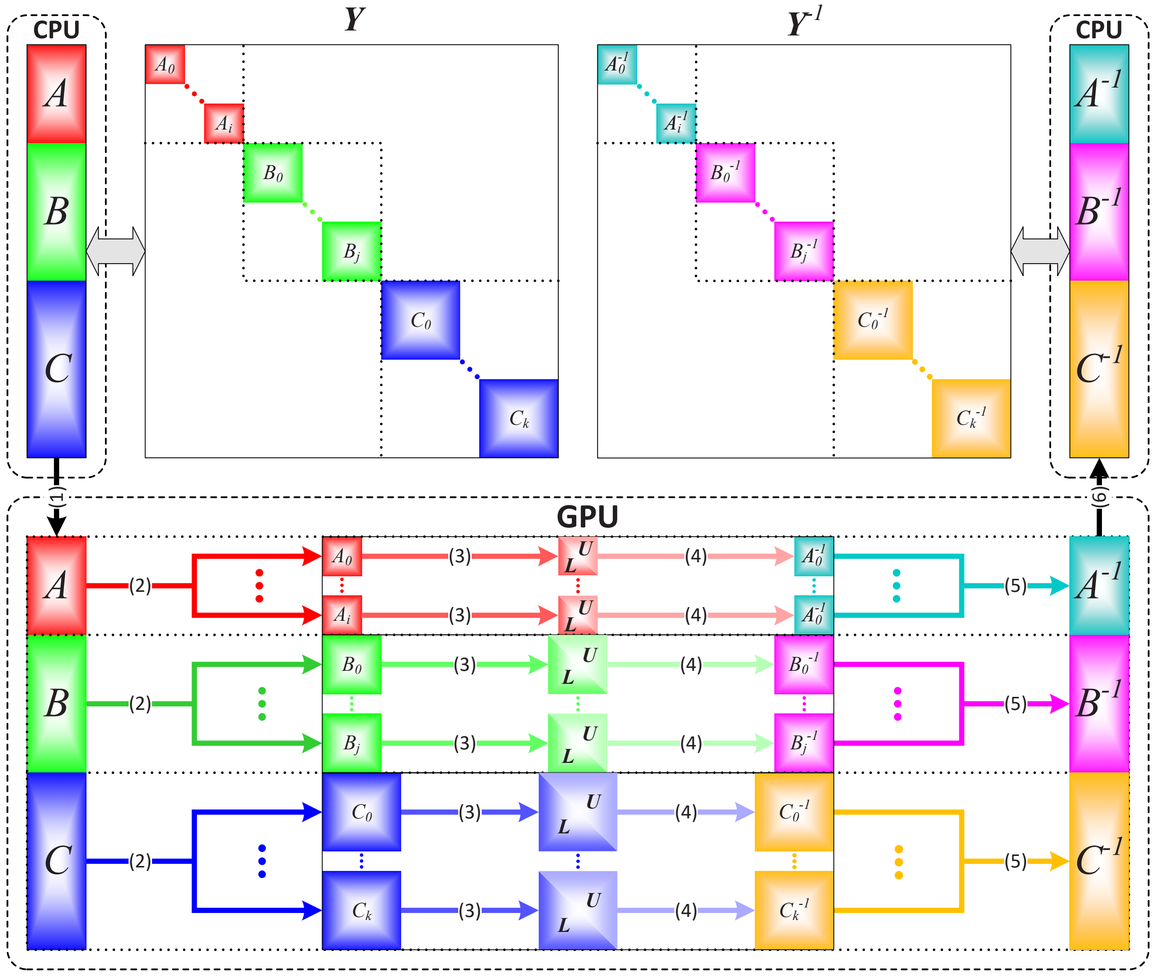


Figure 6.5: Massively parallel matrix inverse based on LU Factorization.

where n is the dimension of the linear system, $\boldsymbol{y}'_k$ are the column vectors of $\boldsymbol{Y}^{-1}$ and $\mathbf{i}_k$ are the column vectors of $\mathbf{I}$. Since $\boldsymbol{Y}$ is factorized into $\boldsymbol{LU}$, and $\mathbf{i}_k$ is independent with each other, the column vectors $\boldsymbol{y}'_k$ can be solved by forward-backward substitution in parallel and combined into $\boldsymbol{Y}^{-1}$ finally. The working flow of massively parallel matrix inverse based on LU factorization is shown in Fig. 6.5. The $\boldsymbol{Y}$ matrix is partitioned into small blocks which is grouped according to the dimensions, for instance groups A, B and C.

- In step (1), all grouped data are copy from host side (CPU) to device side (GPU).
- All matrix block are extracted from the groups and assigned to different CUDA blocks in the light of dimensions in step (2).
- In step (3), The LU factorization for each block is processed in parallel.
- The inverse of each block is computed with massively threads in step (4).

- All data of inverse matrices which have the same size with the original matrices are regrouped into a large data block containing groups A^{-1}, B^{-1} and C^{-1} in step (5).
- And in step (6), all grouped data of inversed matrices are copy back from device side to host side, which can be extracted back into the inversed admittance matrix, $\boldsymbol{Y}^{-1}$, with block diagram structure.

After the admittance matrix is inversed and stored, the solution of the linear system can be obtained by matrix-vector multiplication, which can be processed in high-rate parallel as well. If a switching happens, only the block related to that switch need to be updated, and the other blocks are remained.

6.4 Nonlinear System Newton-Raphson Method

To find the solution of the nonlinear system,

$$\boldsymbol{F}(\boldsymbol{v}) = 0, \tag{6.8}$$

consisting k unknown variables, the Newton equations with Jacobian matrix $\boldsymbol{J}_F(\boldsymbol{v})$ gives as follows,

$$\boldsymbol{J}_F(\boldsymbol{v}^n)\Delta\boldsymbol{v}^{n+1} = -\boldsymbol{F}(\boldsymbol{v}^n), \tag{6.9}$$

where $\Delta\boldsymbol{v}^{n+1}$ is given as

$$\Delta\boldsymbol{v}^{n+1} = \boldsymbol{v}^{n+1} - \boldsymbol{v}^n. \tag{6.10}$$

Jacobian matrix $\boldsymbol{J}_F(\boldsymbol{v})$ is a $k \times k$ matrix of the first-order partial derivatives of $\boldsymbol{F}$ respecting the unknown variables $\boldsymbol{v}$, given as

$$\boldsymbol{J}_F = \frac{d\boldsymbol{F}}{d\boldsymbol{v}} = \begin{bmatrix} \frac{\partial F_1}{\partial v_1} & \frac{\partial F_1}{\partial v_2} & \cdots & \frac{\partial F_1}{\partial v_k} \\ \frac{\partial F_2}{\partial v_1} & \frac{\partial F_2}{\partial v_2} & \cdots & \frac{\partial F_2}{\partial v_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_k}{\partial v_1} & \frac{\partial F_k}{\partial v_2} & \cdots & \frac{\partial F_k}{\partial v_k} \end{bmatrix}. \tag{6.11}$$

Therefore, to find the root of a nonlinear system is converted to solving a linear system multiple times. Since the Jacobian matrix is normally updated in every iteration, using matrix inverse methodis less efficient than LU or Gaussian elimination with substitution to

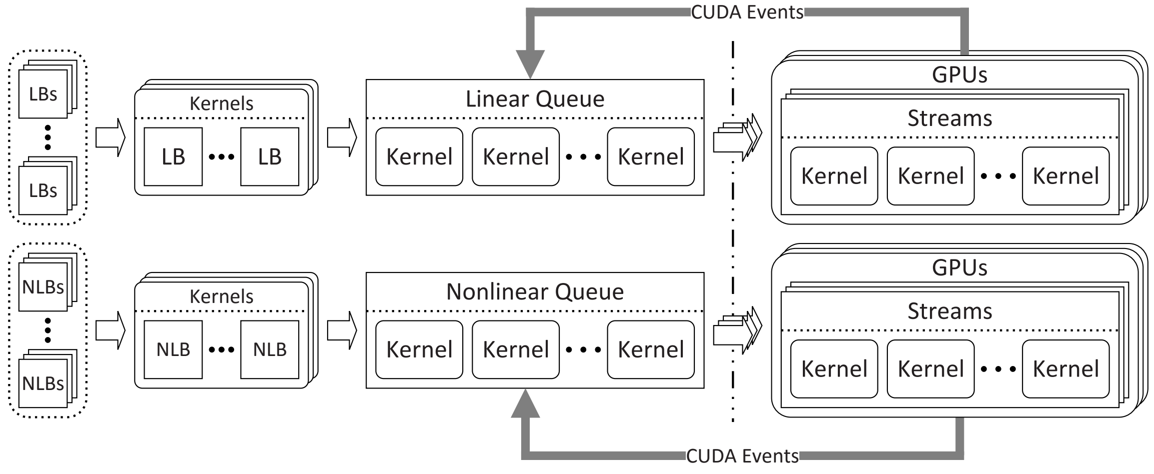


Figure 6.6: Mechanism of computational load balancing and event synchronization.

find the solution. After $\Delta \boldsymbol{v}^{n+1}$ is solved, $\boldsymbol{v}^{n+1}$ can be found with (6.10), by which Jacobian matrix can also be updated. The solving process is repeated till the solution difference, $||\Delta \boldsymbol{v}^{n+1}||$, is converged.

6.5 Balance and Synchronization

Since the large scale system has already been decomposed into LBs and NLBs of similar size relevantly small, the computing tasks can be assigned to each GPU evenly with a round-robin scheme with the task queues [71], if more than two GPUs are present on the simulation platform, as shown in Fig. 6.6. There are several criteria that are followed during the workload distribution:

- linear and nonlinear subsystems are processed in different groups of GPUs separately;
- all blocks belonging to one subsystem are assigned to the same GPU due to data interchange inside the subsystem;
- linear blocks with the same size can be grouped in multiple CUDA kernels and apportioned to different CUDA streams;
- nonlinear blocks with the same components can be grouped in multiple CUDA kernels and apportioned to different CUDA streams;
- CUDA kernels inside the queue are synchronized by CUDA events.

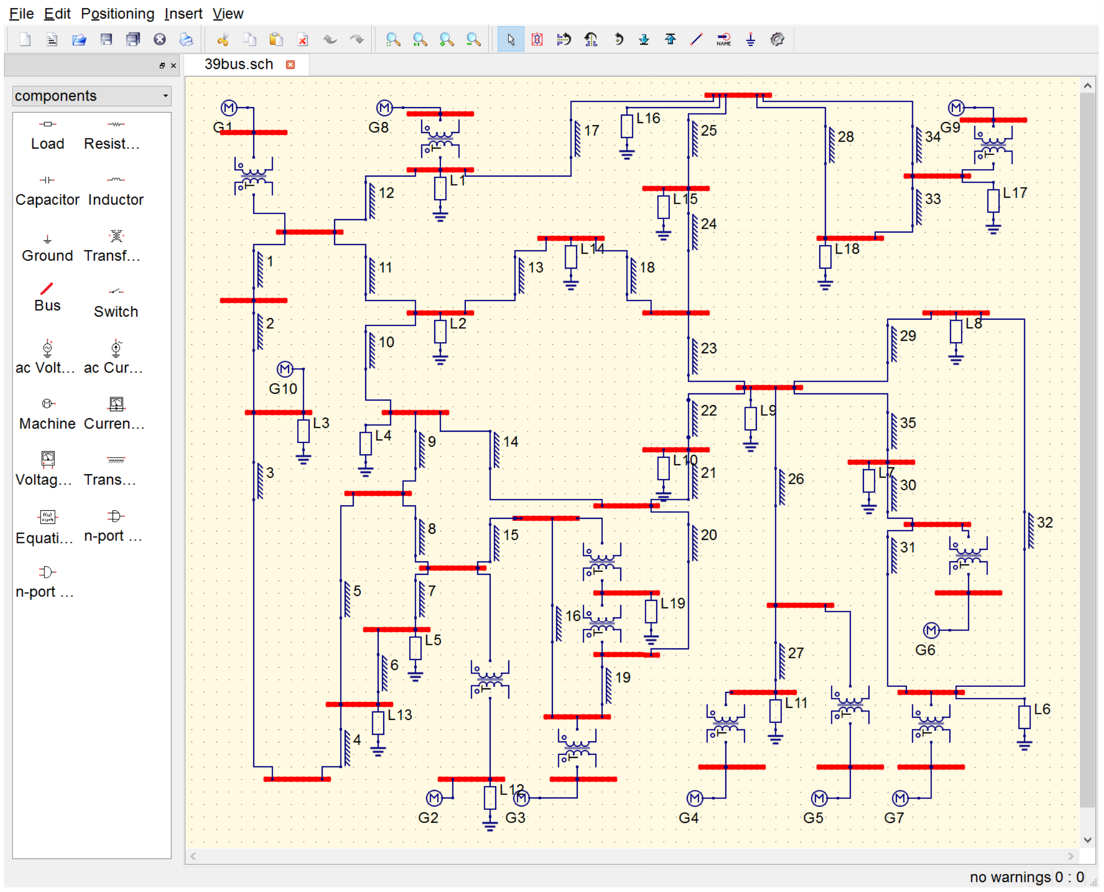


Figure 6.7: GUI for the fine-grained GPU-based EMT simulator.

6.6 GUI

A graphical user interface (GUI) prototype was developed for the GPU-based fine-grained EMT simulation tool, which provides basic functions for network construction and parameter configuration, as shown in Fig. 6.7. The power system diagram created by users is transformed to a Netlist file, which contain the information of bus nodes, connecting relations, parameter values, the number of components, modeling arguments and so on. When the massively parallel EMT simulation engineer accesses the Netlist file, all information of the electrical power network is parsed and extracted, whose results are saved into database for the network topology analysis to create the decomposition rules. According to the decomposition information, the kernel data structures of computing engineering, such as data structures for admittance matrix and Jacobian matrix are built up and assigned to relevant computing units. Finally, the large-scale EMT simulation is processed following the SIMT execution model on the GPU-based massively parallel computing platform.

6.7 Summary

In this chapter, the implementation of massively parallel EMT simulation on GPUs is introduced, including computing platform environment, hardware and software configuration, simulation work flow, multi-stream concurrent execution, linear solution, nonlinear solution, synchronization mechanism and GUI.

7 Simulation Case Studies

In order to show the accuracy of transients, and the acceleration for the proposed GPU-based parallel EMT simulator, four test cases are utilized.

- In the first test case, various transient behaviors are presented, and the simulation results are validated by the EMT software ATP and EMTP-RV®.
- In the second test case, the accelerating performance of GPU, whose execution times on various system scales are compared to those of EMTP-RV®, is shown and analyzed by running the EMT simulation on the extended large-scale power systems.
- In the third test case, the single-phase and 3-phase physics-based MMC circuits are simulated and compared to those of SaberRD®.
- In the last test case, the 3-phase behavior-based MMC based AC/DC converter is simulated and compared with different submodule levels.

Table 7.1: Test system specification

CPU	Intel Xeon E5-2620
Main memory	32GB
GPU	GeForce GTX 1080 (Pascal) $\times$ 2
Video memory	8GB $\times$ 2 (16GB)
OS	Windows 7 Enterprise, 64-bit

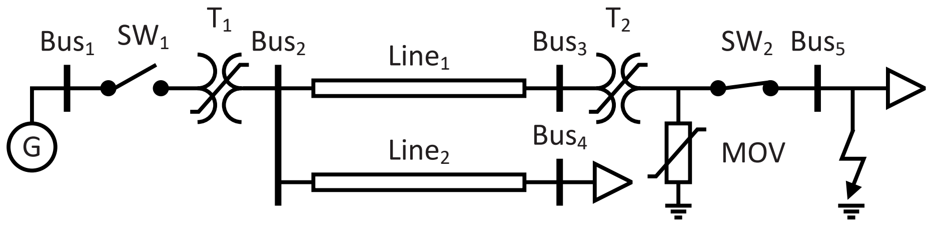


Figure 7.1: Single-line diagram for Case Study A.

The hardware and software environment of the test system is listed in Table 7.1, and the parameters for the test cases are given in the Appendix B.

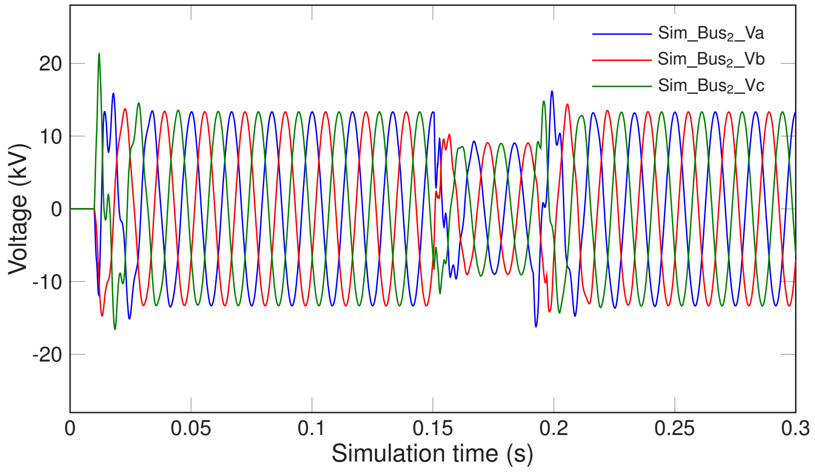
7.1 Case Study A

The synchronous machine (SM), two transformers (T_1, T_2) and the arrester (MOV) are nonlinear components in the test system, as shown in Fig. 7.1. The first switch (SW_1) closes at 0.01s, the ground fault happens at 0.15s, and then the second switch (SW_2) opens at 0.19s to clear the fault. The total simulation time is 0.3s with $20\mu s$ time-step. All results of GPU-based EMT simulation are compared with those of EMTP-RV® and ATP.

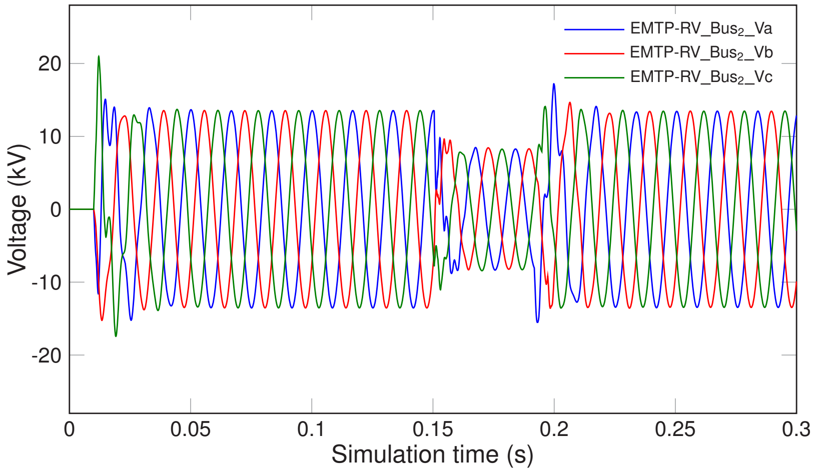
The 3-phase voltages at Bus_2 are shown in Fig. 7.2, which are the output voltages of the step up transformer, T_1.; the 3-phase currents through Bus_2 are shown in Fig. 7.3, which are the currents through the transformer, T_1; the 3-phase voltages at Bus_3 are shown in Fig. 7.4, which are the waveforms after transmission and the input of step down transformer, T_2t the power angle and electromagnetic torque waveforms of the synchronous machine, G, are shown in Fig. 7.5; and the active and reactive power of the case study A are shown in Fig. 7.6. Additionally, the phase a voltage of Bus_2, phase a current of Bus_2 and power angle are compared in overlapped waveforms in Fig. 7.7 with the simulation results of GPU-based EMTP, EMTP-RV® and ATP.

When the switches activate and fault happens in the circuit of case study A, the power electromagnetic transients are clearly demonstrated by the proposed GPU-based parallel simulation in the waveforms of voltages, currents, power angle, electromagnetic torque, active power and reactive power, which illustrate good agreement with the results from EMTP-RV® and ATP.

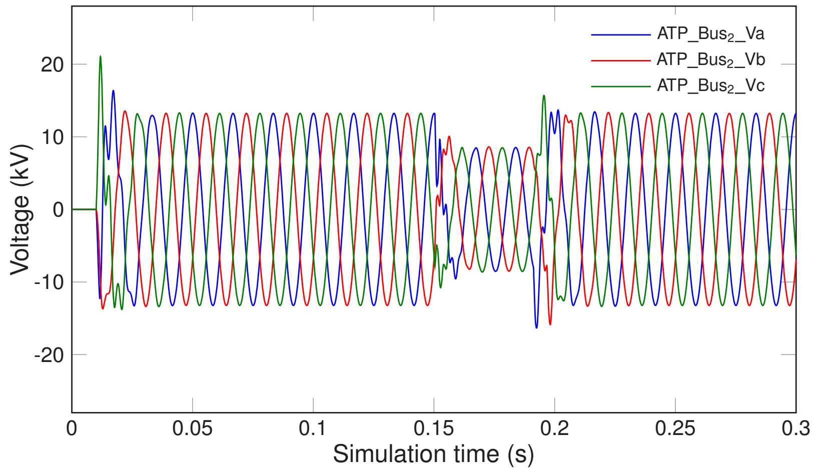
Although the different synchronous machine model (SM type 58) and transmission line model (Line type JMarti) are used in ATP other than GPU-based parallel simulation and EMTP-RV®, the results are nevertheless close enough to represent designed transient phenomena. Due to more sophisticated models applied, there are more details on the



(a) Bus$_2$ Voltages from GPU-based simulation

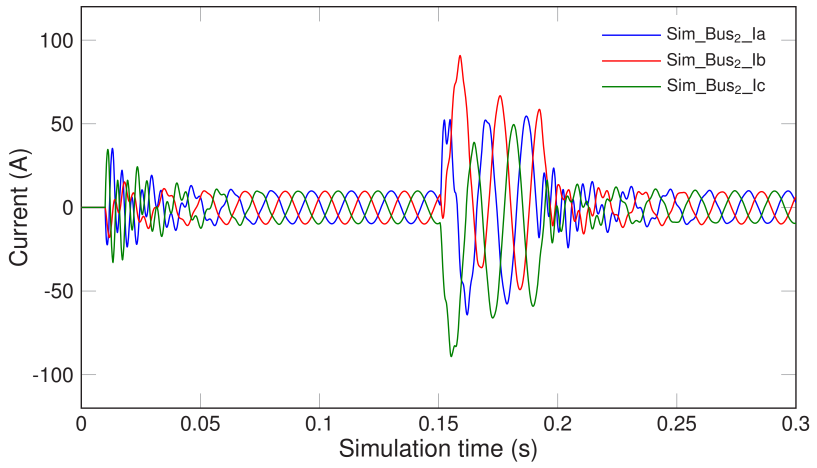


(b) Bus$_2$ Voltages from EMTP-RV®

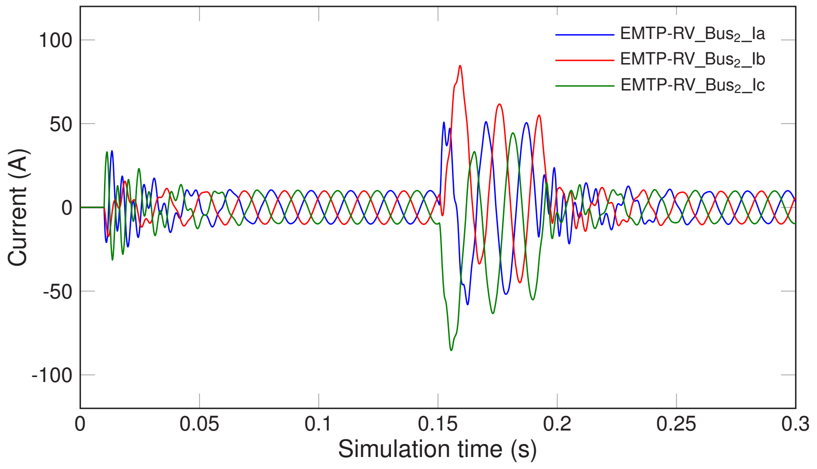


(c) Bus$_2$ Voltages from ATP

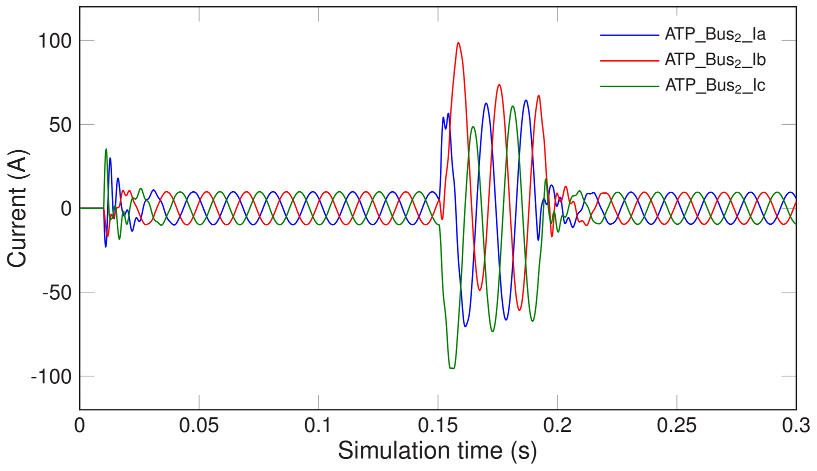
Figure 7.2: 3-phase voltages comparison at Bus$_2$ of Case Study A



(a) Bus$_2$ currents from GPU-based simulation

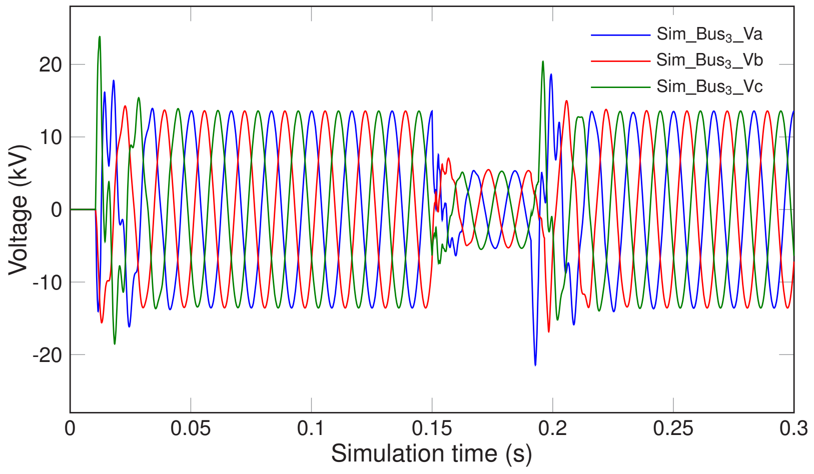


(b) Bus$_2$ currents from EMTP-RV®

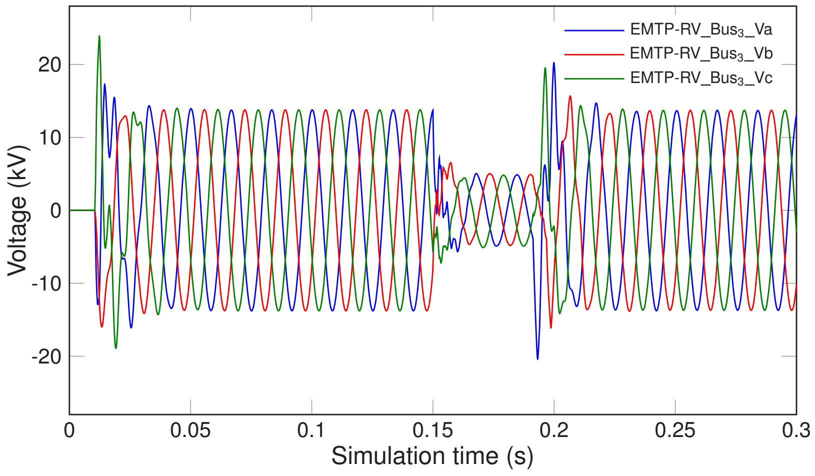


(c) Bus$_2$ currents from ATP

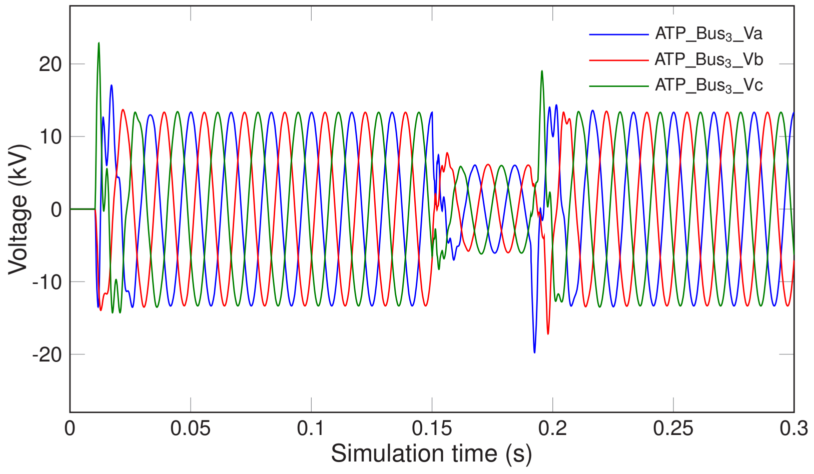
Figure 7.3: 3-phase currents comparison through Bus$_2$ of Case Study A



(a) Bus$_3$ Voltages from GPU-based simulation

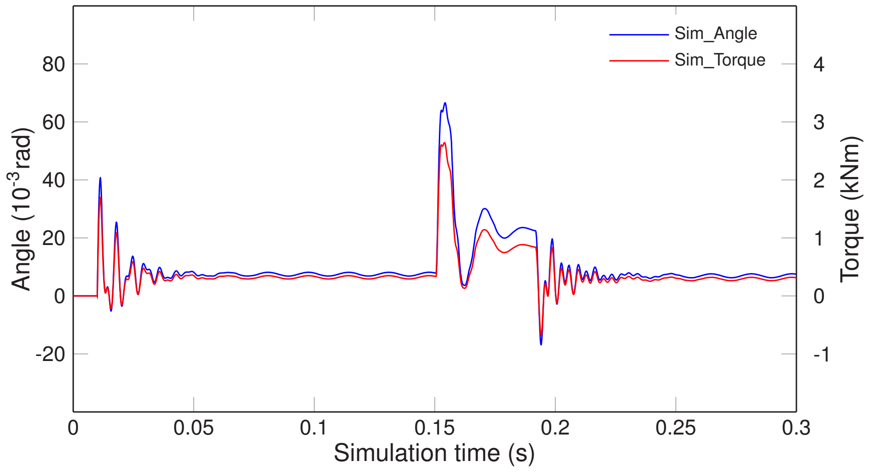


(b) Bus$_3$ Voltages from EMTP-RV®

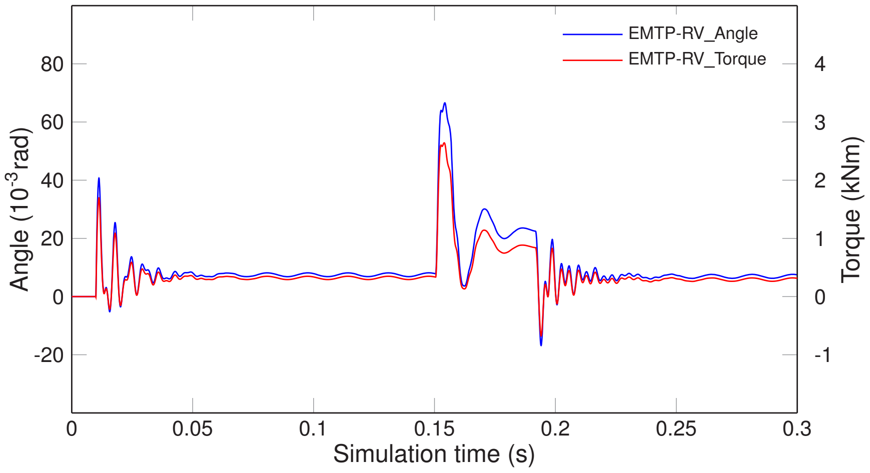


(c) Bus$_3$ Voltages from ATP

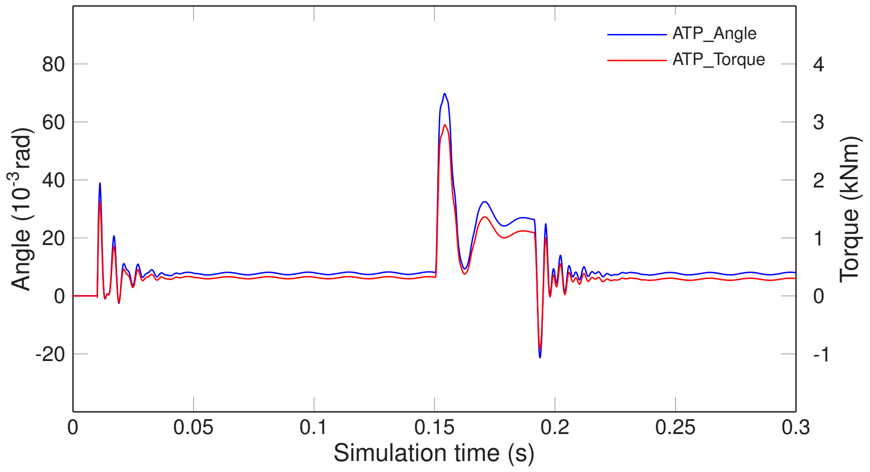
Figure 7.4: 3-phase voltages comparison at Bus$_3$ of Case Study A.



(a) Angle and torque from GPU-based simulation

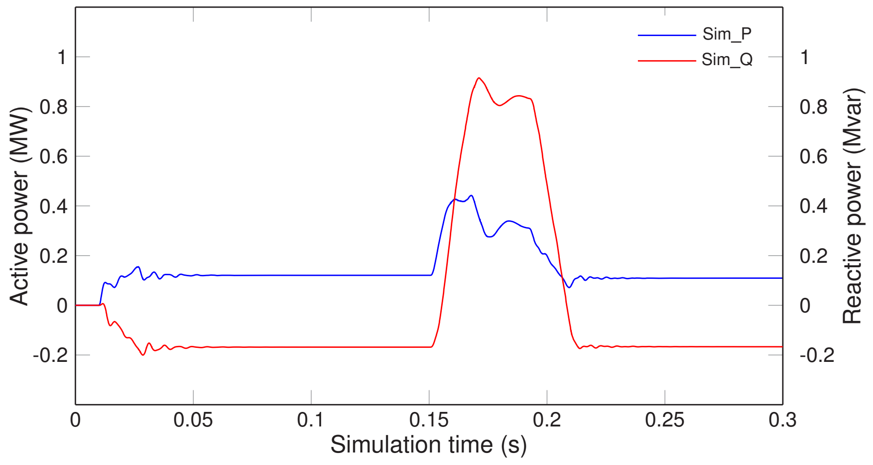


(b) Angle and torque from EMTP-RV®

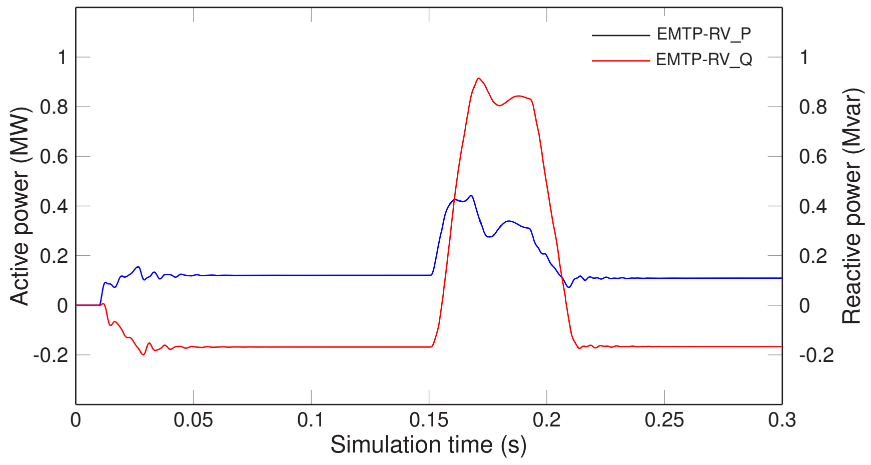


(c) Angle and torque from ATP

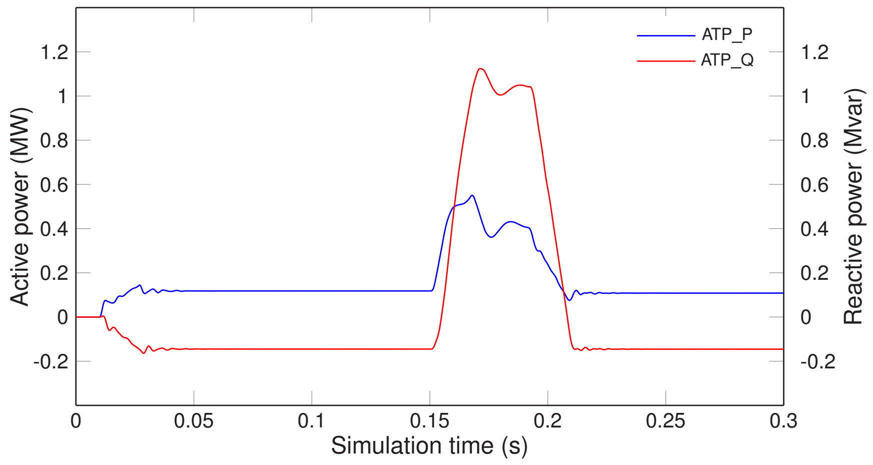
Figure 7.5: Synchronous machine angle and torque of Case Study A.



(a) P and Q from GPU-based simulation

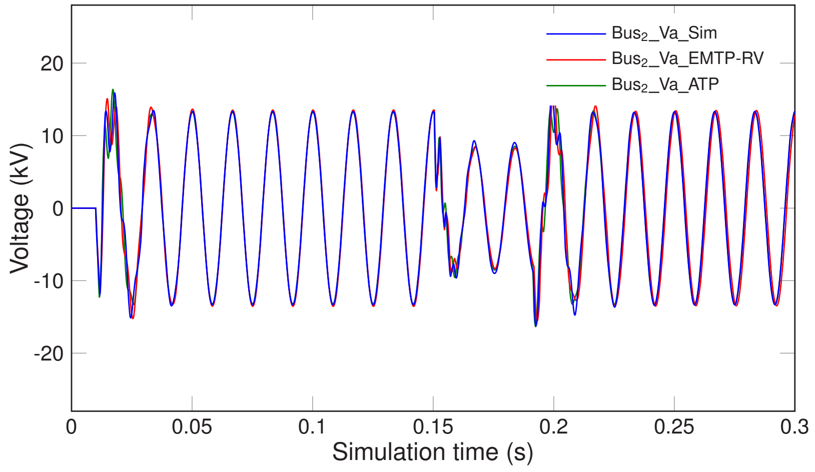


(b) P and Q from EMTP-RV®

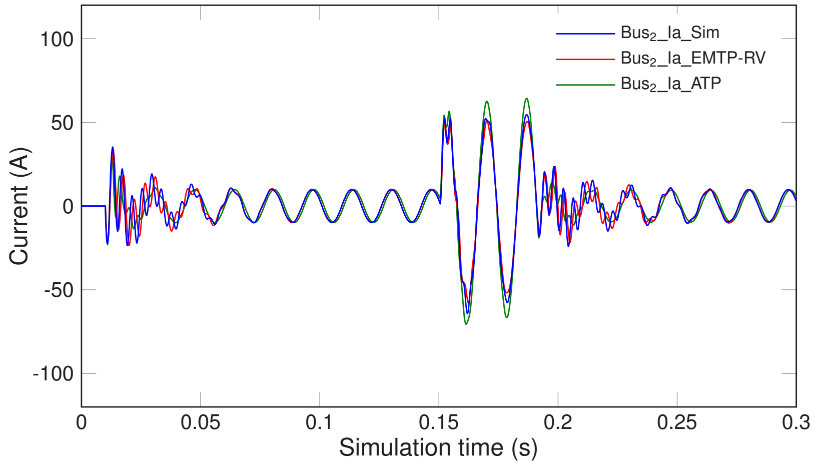


(c) P and Q from ATP

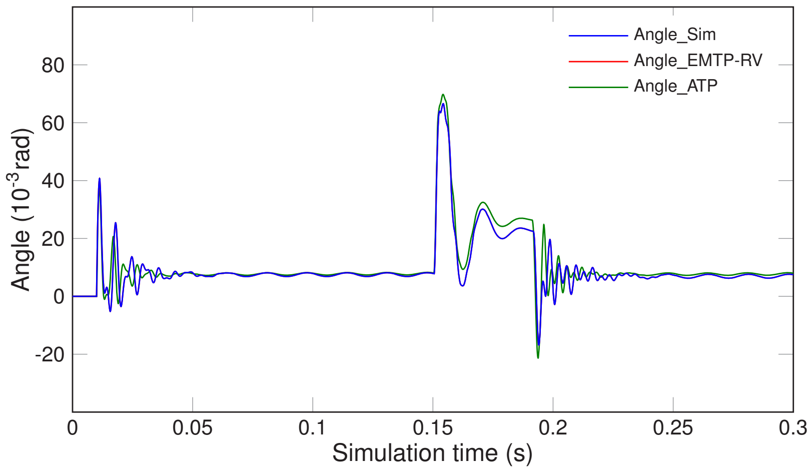
Figure 7.6: Active power and reactive power of Case Study A.



(a) Bus$_2$ phase a overlapped voltages



(b) Bus$_2$ phase a overlapped currents



(c) Bus$_1$ overlapped Angles

Figure 7.7: Comparison of overlapped waveforms

transient waveforms from the GPU simulation.

7.2 Case Study B

In order to show the acceleration of GPU based EMT simulation, large-scale power systems are built, which are based on the IEEE 39-bus network as shown in Fig. 7.8. Considering the interconnection is a path of power grid growth, the large-scale networks are obtained by duplicating the Scale 1 system and interconnecting by transmission lines. As shown in Table 7.2, the test systems are extended up to 3×79872 (239616) buses. All networks are decomposed into LBs, NLBs and CBs after fine-grained decomposition in the unified patterns. For instance, the 39-bus network is divided into 28 LBs, 21 NLBs and 10 CBs, as shown in Fig. 7.9. The simulation is based on CPU, 1-GPU and 2-GPU computational systems from 0 to 100ms with 20μs time-step respectively, using double precision and 64-bit operation system. All test cases are extended sufficiently long to suppress the deviation of the software timer, which starts after reading the circuit net-list and parameters, including network decomposition, memory copy, component model calculation, linear/nonlinear solution, node voltage/current update, result output and transmission delay.

The scaled test networks are given in Table 7.2, including network size, bus number and partition. The execution time for each network is listed in order of network size and

Table 7.2: Comparison of execution time for various networks among CPU, single-GPU and multi-GPU for simulation duration $100ms$ with time-step $20\mu s$

Scale	3ϕ	Blocks			Execution time (s)				Speedup		
	buses	LBs	NLBs	CBs	EMTP-RV®	CPU	1-GPU	2-GPU	CPU	1-GPU	2-GPU
1	39	28	21	10	1.24	1.38	1.16	0.93	0.89	1.06	1.33
2	78	56	42	20	2.71	2.87	1.52	1.36	0.94	1.78	1.99
4	156	112	84	40	5.54	5.53	2.03	1.77	1.00	2.73	3.14
8	312	224	168	80	11.94	11.22	2.96	2.37	1.06	4.04	5.03
16	624	448	336	160	26.50	23.12	4.63	3.35	1.15	5.73	7.91
32	1248	896	672	320	60.65	48.16	7.98	5.19	1.26	7.60	11.68
64	2496	1792	1344	640	142.31	100.11	14.42	8.72	1.42	9.87	16.31
128	4992	3584	2688	1280	323.26	210.76	28.36	15.83	1.53	11.40	20.42
256	9984	7168	5376	2560	705.92	439.26	55.47	30.23	1.61	12.73	23.35
512	19968	14336	10752	5120	1513.35	892.45	109.29	57.78	1.70	13.85	26.19
1024	39936	28672	21504	10240	3314.24	1863.66	225.17	116.86	1.78	14.72	28.36
2048	79872	57344	43008	20480	7033.61	3796.37	454.48	234.37	1.85	15.48	30.01

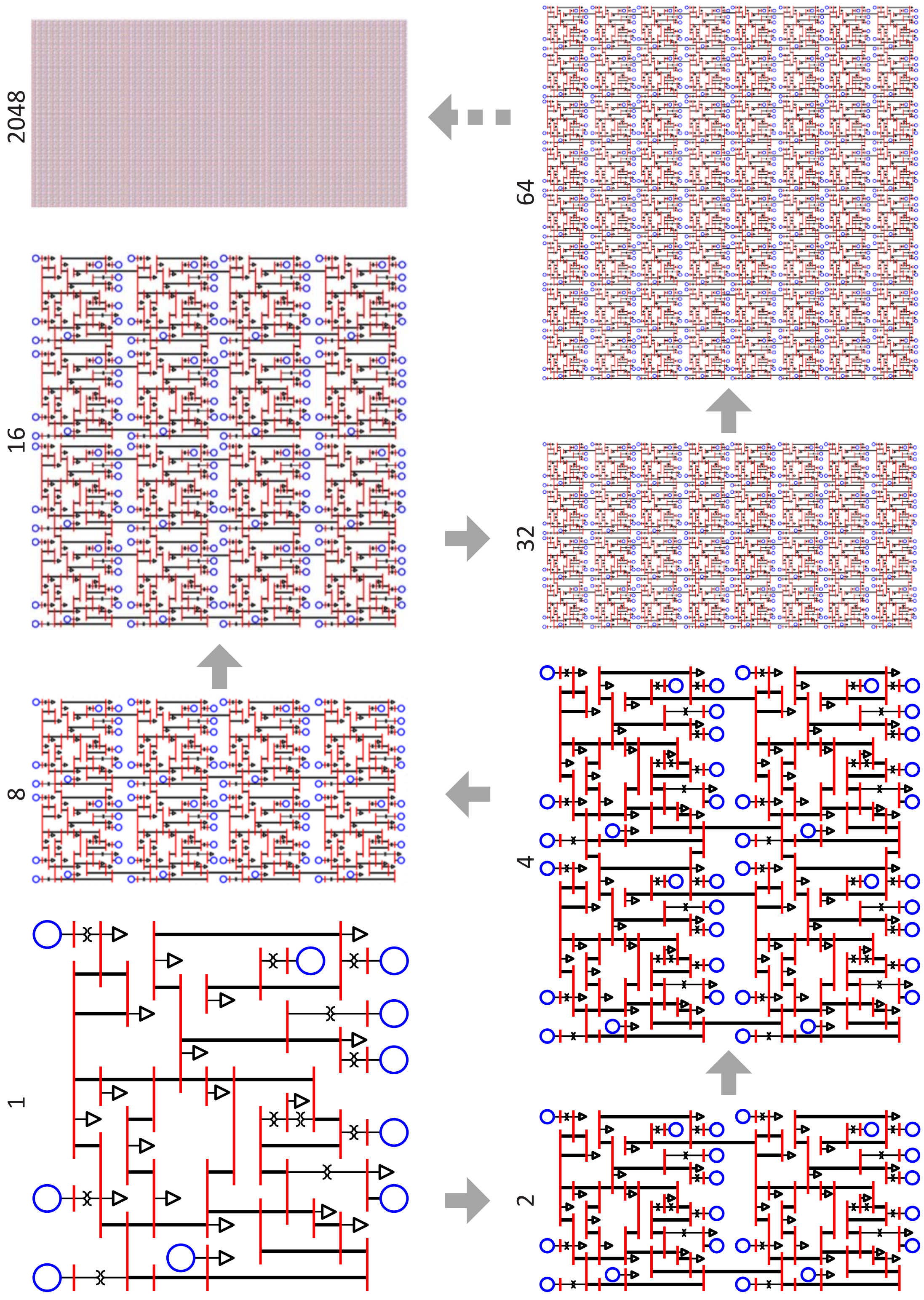


Figure 7.8: System scale extension for case study B.

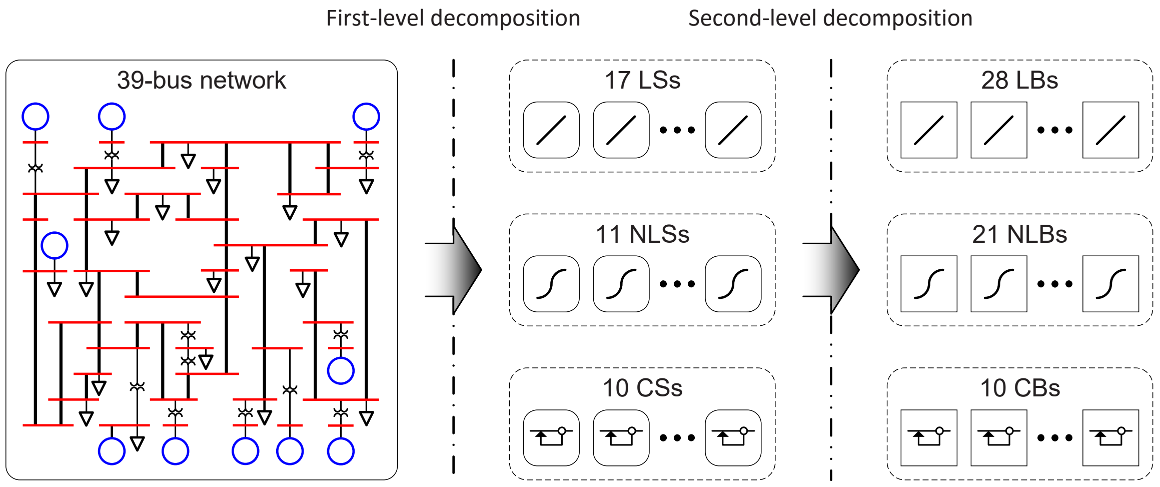


Figure 7.9: Fine-grained decomposition for the 39-bus system.

Table 7.3: Average execution time (ms) for one time-step

Scale	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}
EMTP-RV®	0.25	0.54	1.11	2.39	5.30	12.13	28.46	64.65	141.18	302.67	662.85	1406.72
CPU	0.28	0.57	1.11	2.24	4.62	9.63	20.02	42.15	87.85	178.49	372.73	759.27
1-GPU	0.23	0.30	0.41	0.59	0.93	1.60	2.88	5.67	11.09	21.86	45.03	90.90
2-GPU	0.19	0.27	0.35	0.47	0.67	1.04	1.74	3.17	6.05	11.56	23.37	46.87

categorized by the type of computing systems as well as the speedup referred to the performance on CPU. In the plotted Fig. 7.10 along the 1-GPU speedup curve, the speedup increases slowly when network size is small (lower than 4 scales) since the GPU cannot be fed enough workload; for the network scale from 4 to 32, the acceleration climbs fast, showing the computational power of the GPU is released by fetching a greater amount of data; when the network size is more than 32, the performance approaches a constant since the computational capability of GPU closes to saturation. In the case of 2-GPU system, the trend of speedup increase is similar to the 1-GPU case except that the saturation point is put off because of the doubled computational capability. Due to the nonlinear relationship of the execution time to the system scale, the bar diagrams of execution times for various system scales are zoomed using a logarithmic log axis to obtain a detailed view. Additionally, it can be noticed that the performance of CPU is also enhanced by the proposed decomposition method since the divided circuit blocks simplify the sparse data structure to dense one along with the increasing system scale, thus the systems can be solved by dense solver and avoid the extra cost of dealing with the sparse structure, such as non-zero elements analysis, which is involved in every solution. In that case, the computation

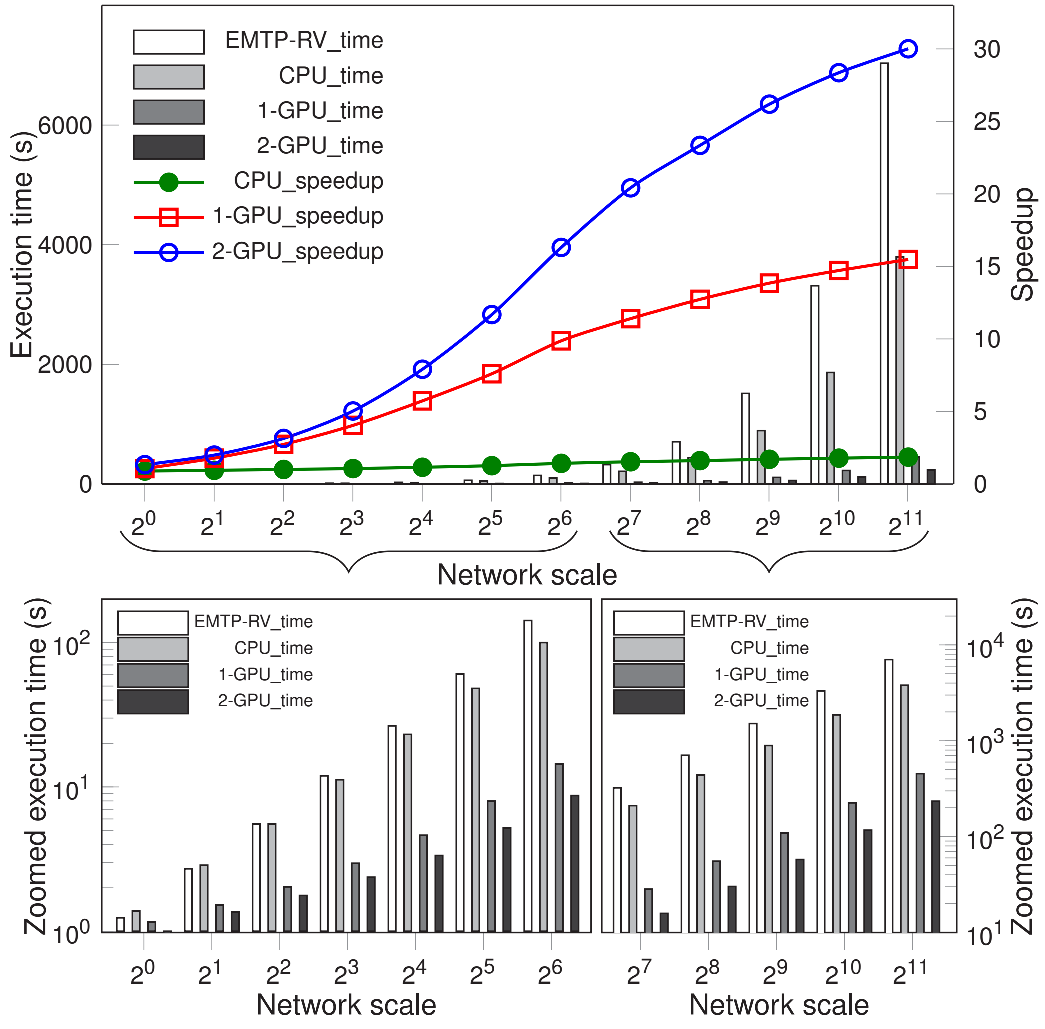


Figure 7.10: Execution time and speedup for varying scales of test networks on CPU, 1 GPU and 2 GPUs based programs compared to EMTP-RV®.

load is almost linearly related to the system scale comparing with the nonlinear traditional sparse solver.

Owing to the shattering network decomposition, the computation load can be well-distributed to each compute device so that the overall performance of a computing system is decided by the number of processors, following the Gustafson-Barsis' law [27]. The average execution time for one time-step is listed in Table 7.3 due to the different convergent speed of each time-step. When the network scale is up to 2^{11} (2048), which is close to the memory limitation of the computing system, the 2-GPU system doubles the performance of the 1-GPU system and attains 30 times faster than EMTP-RV®.

7.3 Case Study C

The performance of massively parallel nonlinear solver based on Newton-Raphson method with fine-grained decomposition is evaluated by solving the nonlinear system of MMC circuit modeling in Hefner's physics-based model. Multiple nonlinear systems with various order derived from difference level MMC circuits are tested. The execution time and convergence are compared among single-thread CPU code, massive-thread GPU code and SaberRD® since the model is included in SaberRD®.

For a l-level MMC system, there are $2(l-1)$ half-bridge SMs used, of which the size of Jacobian matrix $\boldsymbol{J}_{MMC}$ is $(20l-19)\times(20l-19)$ based on Hefner's IGBT model. a typical Jacobian matrix of an IGBT is given with following element values,

$$\begin{bmatrix} 1.17722\times10^{-19} & -1.2410\times10^{-9} & -1.0000\times10^{-12} & -5.3015\times10^{-10} & 0 \\ -1.2410\times10^{-9} & 3.9958\times10^{-9} & 0 & -2.7558\times10^{-9} & 0 \\ 0 & 0 & 0.5454 & 0 & -0.5454 \\ -5.3015\times10^{-10} & -2.7548\times10^{-9} & 0 & 4.3443\times10^{-9} & -1.0593\times10^{-9} \\ -2\times10^{-12} & 0 & -0.5454 & -1.0583\times10^{-9} & 0.5454 \end{bmatrix}.$$

The challenge of solving the high order nonlinear system is the ill conditioning of the Jacobian matrix since the nonlinear solution is obtained by solving multiple linear systems. The sensitivity of a linear system can be measured by the condition number of the square nonsingular matrix A, which is defined as:

$$cond(A) = ||A||\cdot||A^{-1}||. \tag{7.1}$$

If the condition number is close to 1, it means the matrix is well-conditioned, whereas if the condition number is large, the matrix is deemed as ill-conditioned. Unfortunately, due to the variety and difference of the conductance caused by the switching character of the power electric devices, the Jacobian matrices, $\boldsymbol{J}_{MMC}$, of the MMC circuits are very ill-conditioned, whose condition number are more than 1×10^{7} normally. The ill conditioning of Jacobian matrix influences the solution accuracy of the Newton equation; and then the solution impacts the convergence of Newton iteration.

The solution times of various nonlinear systems are list in Table 7.4 with the error tolerance, $\epsilon = 1\times10^{-6}$, and the maximum iteration, $Iter_{max} = 10$. For SaberRD®, the Newton iteration for nonlinear solution cannot converge when the level of MMC reaches 6 (the order of the Jacobian matrix is 101), due to the ill-conditioned linear system of Newton method. The CPU program uses the same fine-grained decomposition algorithm with

Table 7.4: SaberRD®, CPU and GPU solution times of nonlinear systems.

N_L	O_J	Solution time (ms)			Speedup
		SaberRD®	CPU	GPU	
2	21	1.06	1.13	1.42	0.80
3	41	2.00	2.22	1.63	1.36
4	61	2.98	3.20	1.75	1.83
5	81	4.04	4.28	1.80	2.37
6	101	-	5.35	1.88	2.84
7	121	-	6.61	2.03	3.26
8	141	-	7.82	2.15	3.64
9	161	-	9.14	2.24	4.09
10	181	-	10.70	2.31	4.63
11	201	-	12.40	2.42	5.11

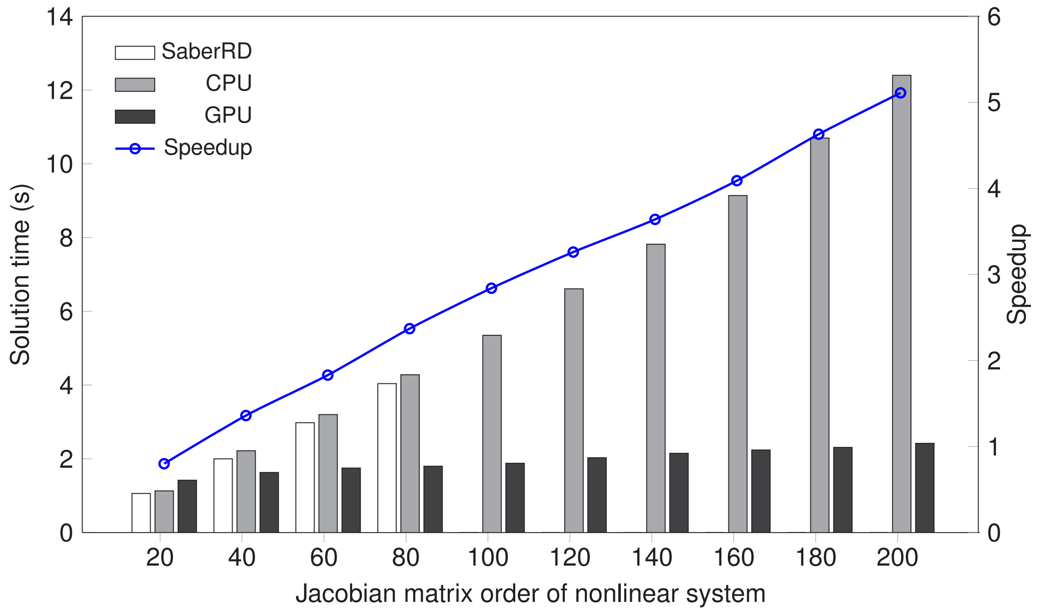


Figure 7.11: Nonlinear system solution time and speedup comparison.

GPU program but runs in single thread. Therefore the convergence of CPU and GPU programs are similar, which both solve the nonlinear system up to 11 levels MMC (201 by 201 Jacobian matrix); however, even for the decomposed system, the Newton iteration cannot converge when the level of MMC is higher than 11. Over 5 time speedup is gained from the advantage of massively parallel computing. The nonlinear system solution times respecting to the order of Jacobian matrices are compared by bar graph and the speedup trend is plotted in Fig. 7.11.

Table 7.5: Condition number of Jacobian matrix during Newton iteration

Number of iteration	1	2	3	4	5
cond($\boldsymbol{J}_{MMC}$)	2.3813×10^{17}	1.6154×10^{17}	5.7653×10^{16}	9.2175×10^{16}	7.1517×10^{17}

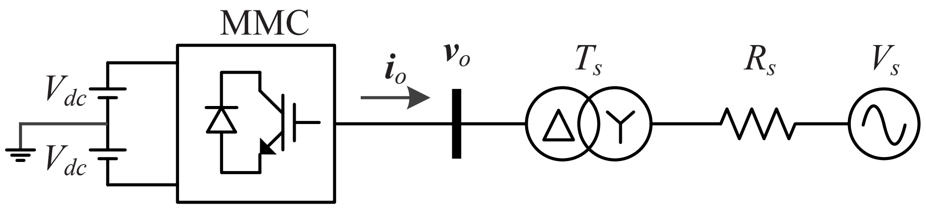


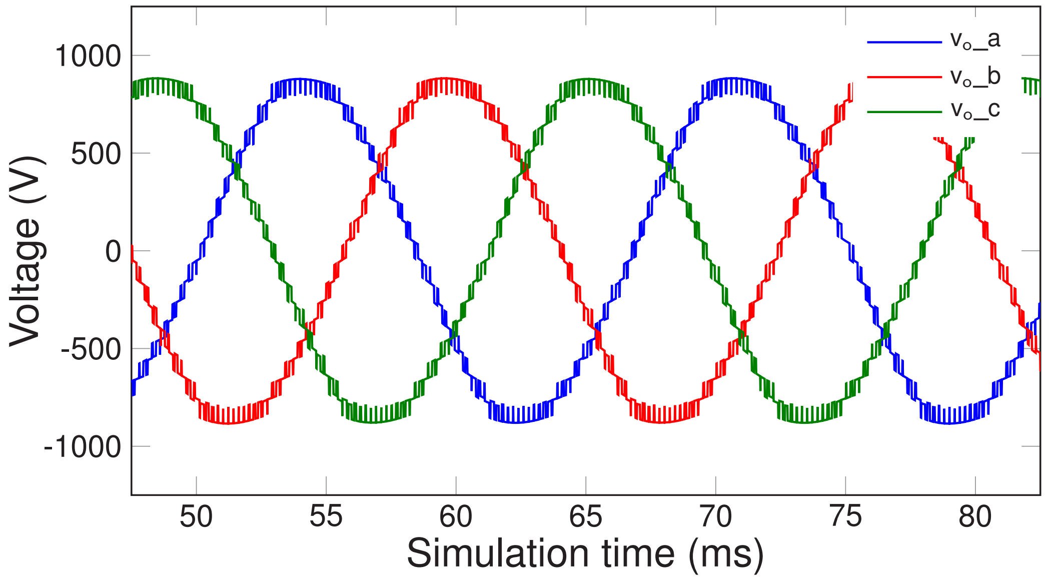
Figure 7.12: Single-line diagram for Case Study D.

Table. 7.5 lists the condition numbers of $\boldsymbol{J}_{MMC}$ during 5 steps of Newton iteration. The condition numbers of the Jacobian matrix are quite large. Therefore, the solution of the linear system in Newton method is very sensitive to errors, and the Newton iteration is very difficult to converge due to the inaccurate results from the linear system. The situation gets even worse along with the increasing levels of MMC circuit since the order of the Jacobian matrix of nonlinear system grows as well.

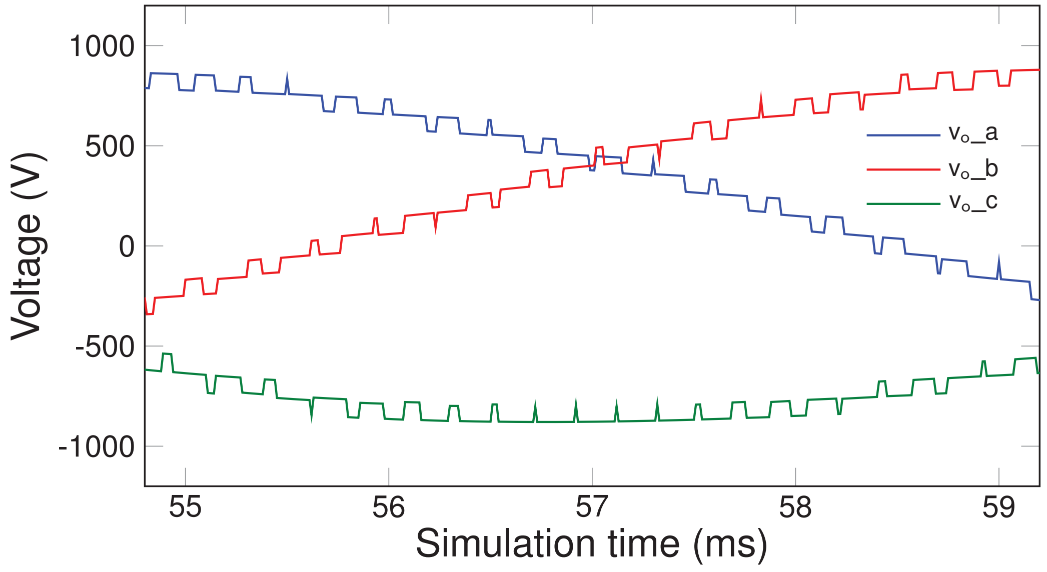
7.4 Case Study D

The AC/DC converter based on MMC, as shown in Fig. 7.12, is used to evaluate the power electronic type of switching in GPU-based EMT simulation. Due to the proposed fine-grained decomposition algorithm, all 6 arms in 3-phase MMC are decoupled and each SM in one arm is processed by one thread. The waveforms in Fig. 7.15 show the EMT simulation results of the converter with 8 SMs per arm (17-level) MMC with 10μs time-step. The 3-phase output voltages of MMC are shown in Fig. 7.15(a) and zoomed in Fig. 7.15(d) between 56ms to 59ms; the capacitor voltages of upper and lower arms of SMs in MMC are shown in Fig. 7.15(b) and the waveforms inside the marked area on upper arm curves are zoomed in Fig. 7.15(e) between 53.2ms and 54.8ms; 3-phase output currents are shown in Fig. 7.15(c); active and reactive power control results are shown in Fig. 7.15(f), which correctly follow the reference P and Q signals.

The performance of GPU-based massively parallel EMT algorithm with the proposed fine-grained decomposition is compared to CPU-based simulation by varying the number of SMs per arm in MMC. The execution times from CPU, 1-GPU and 2-GPU based simulation of 3-phase MMC converter with a 10μs time-step during 0.5s simulation are listed and



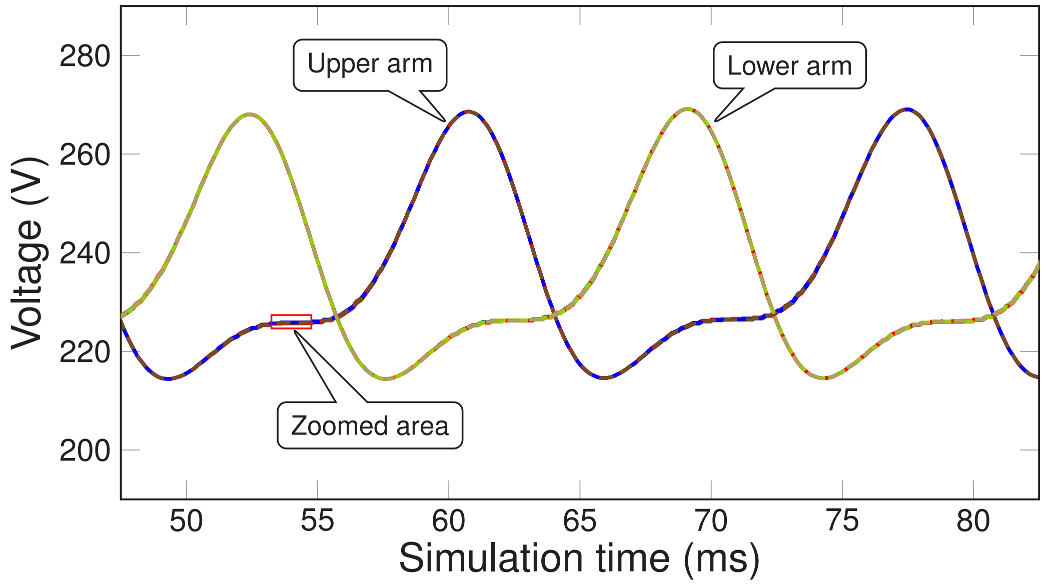
(a) 3-phase output Voltages



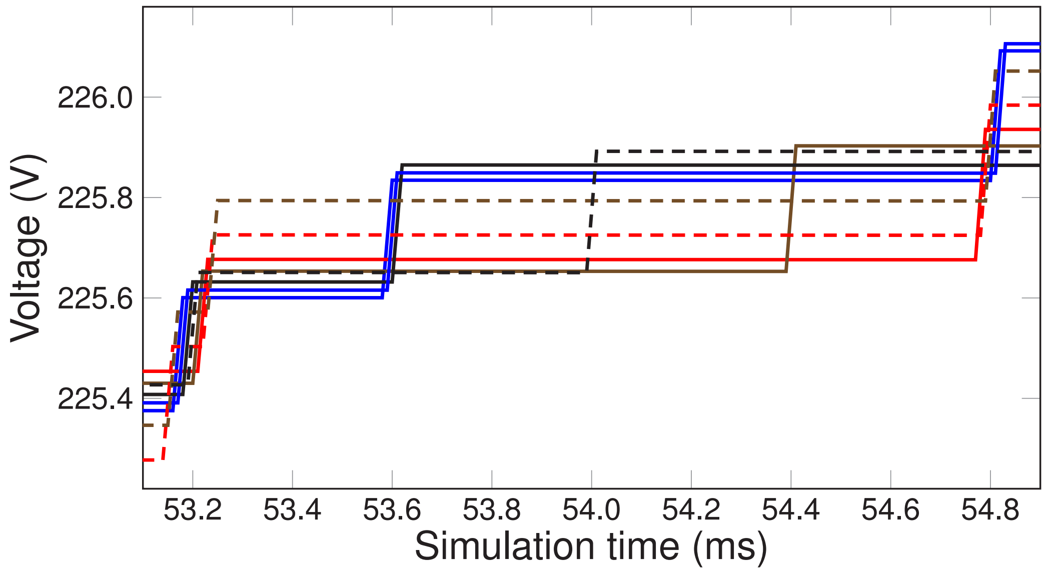
(b) Zoomed output voltages of MMC

Figure 7.13: 3-phase output Voltages of 17-level MMC from GPU-based simulation

compared in Table 7.6 from 8 SMs per arm (17-level) to 1024 SMs per arm (2049-level) in MMC. In Fig. 7.16, the bar graph shows the comparison of execution among various computational platforms and curves illustrate that the speedup keeps increasing along with the number of SMs in MMC, which is close to 51 times for 1-GPU platform and reaches 64 times for 2-GPU platform compared to the single thread CPU simulation. It is obvious that the execution time is almost doubled when the number of SMs in MMC is doubled on CPU-based simulation; however, it grows much slower for GPU-based simulation. Since



(a) SM capacitor voltages



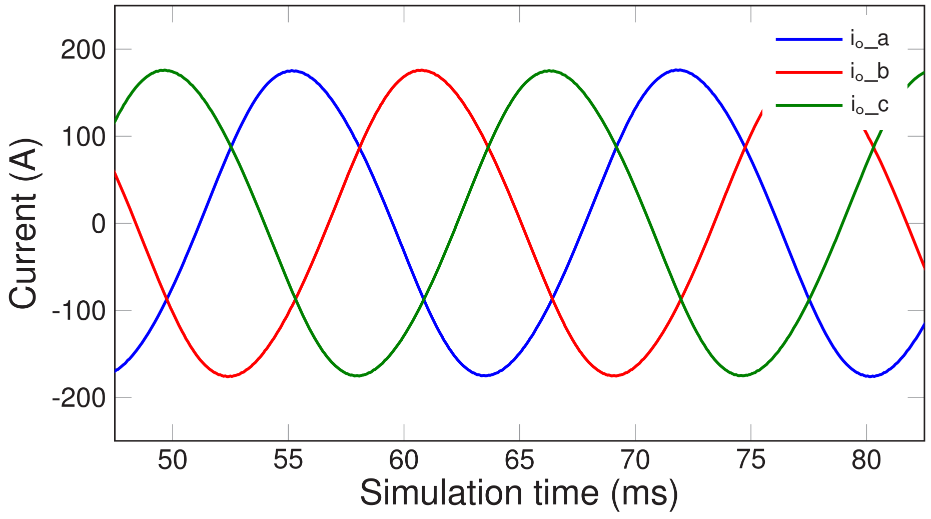
(b) Zoomed SM capacitor voltages of MMC

Figure 7.14: SM capacitor voltages of 17-level MMC from GPU-based simulation

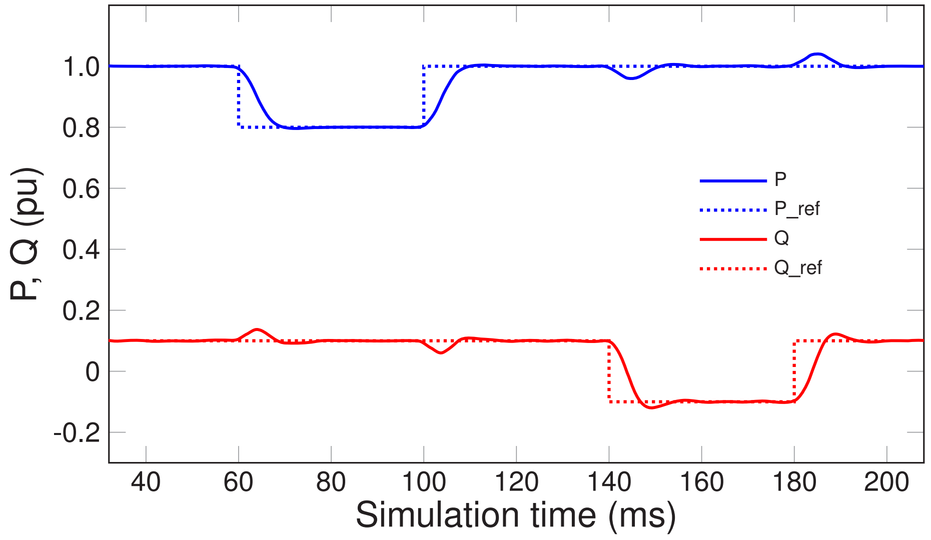
the increase of speedup is close to linear, the computational complexity order of the EMT simulation is reduced effectively by the massively parallel computation on the GPUs.

7.5 Summary

In this chapter, four test cases are studied to demonstrate the accuracy and performance of the proposed GPU-based parallel EMT simulator. The transients caused by system energi-



(a) 3-phase Output currents



(b) Active and reactive power control

Figure 7.15: 3-phase output currents, P and Q of 17-level MMC from GPU-based simulation.

zation and ground fault are verified with mainstream commercial EMT simulation tools, including EMTP-RV® and ATP, which shows a reasonable agreement. The large-scale power systems made of 39-bus system extend the number of buses up to 23.9k, of which the execution times and acceleration are compared among those of EMTP-RV®, CPU, 1-GPU and 2-GPU EMT simulation. For the power electronic circuit, the performance of the nonlinear solver based on fine-grained decomposition is tested and compared with SaberRD® and CPU program, which shows better convergence and acceleration benefitting from the massively parallel computing on GPU. The MMC circuit with linear behavior-based mo-

Table 7.6: CPU and GPU execution times of 3-phase AC/DC converter for 0.5s duration with $10\mu s$ time-step.

N_{SM}	N_{SM}	Execution time (s)			Speedup	
per arm	total	CPU	1-GPU	2-GPU	1-GPU	2-GPU
8	48	9.77	1.98	1.91	4.93	5.12
16	72	17.75	2.18	2.02	8.14	8.79
32	144	34.83	2.61	2.27	13.34	15.34
64	288	68.89	4.16	3.12	17.30	22.08
128	768	131.72	5.92	4.87	23.03	27.05
256	1536	254.95	8.64	6.96	29.51	36.63
512	3072	485.58	12.87	9.55	37.73	50.85
1024	6144	902.69	17.53	14.03	51.49	64.34

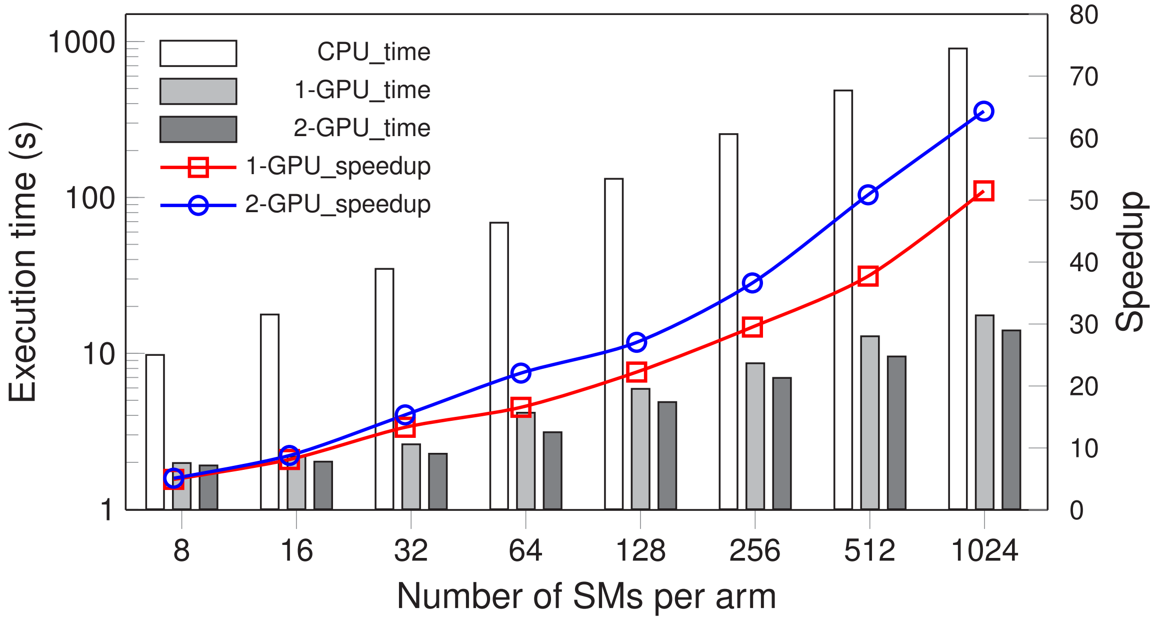


Figure 7.16: Execution time and speedup of simulation for varying levels of MMC on CPU, 1 GPU and 2 GPUs based programs.

deling are implemented on GPU-based massively parallel computing platform. Multiple-level MMC circuits, up to 1024, are tested to show accuracy and performance in comparison to single-thread CPU program.

8 Conclusions and Future Works

Electromagnetic transient simulation of large-scale nonlinear power grids is computationally demanding, and it is therefore imperative to accelerate the simulation which is used to conduct a wide variety of studies by electric utilities. The shattering network decomposition methods proposed in the thesis partitions the power electrical circuit, decouples the linear and nonlinear solution, and accelerates the power electronic converter circuit simulation working with high frequency switching, which enables the transient simulation to take advantage of massively parallel computing platform conveniently and unleashes the computational power of GPUs. All the component models employed are detailed and the solution is fully iterative. Along with the increasing scale, fully decomposed simulation can be easily deployed on to multi-GPU computing system and implement the massively parallel algorithms, so that the maximum performance of simulation can be obtained according Gustafson-Barsis' law.

8.1 Contributions of Thesis

- The fundamental procedure framework and data structure for the electromagnetic transient simulation of power systems are designed in this work, which adapt to the attributes of the SIMT parallel execution model of the modern massive-core processor such as GPU.

- Nonlinear massively parallel modules are developed for synchronous machine, transformer with magnetic saturation and nonlinear load, including both piecewise and analytical nonlinear expressions.

- Linear massively parallel modules are developed for transmission line and linear passive components (R, L, C) with frequency-dependent detailed model and unified lump model.
- Control logic is included in the massively parallel EMT simulation system and interfaced to the main routine of components models computation.
- The power electronic circuit, MMC, are decoupled in fine-grained to adapt to the massively parallel solving algorithm for both nonlinear and linear modeling with proposed shattering decomposition method.
- The high frequency switching power electronic devices are implemented on the GPU-based massively parallel computing platform with linear behavior-based model.
- The multi-level shattering decomposition method is proposed by analyzing the topology of the circuit network, including the coarse-grained level based on propagation delay and the fine-grained level based on system diakoptics, to accommodate the SIMT massively parallelism.
- For the fine-grained system partition, compensation network decomposition and Jacobian domain decomposition are proposed for linear and nonlinear solution in massively parallel computing platform respectively.
- The massively parallel linear system solution is developed based on SIMT data and execution scheme. In addition to the basic algebra operations, the linear solver includes matrix LU decomposition, forward/backward substitution and matrix inverse.
- The massively parallel nonlinear system solution is developed based on Newton-Raphson iteration algorithm according to the SIMT data and execution method.
- A prototype of GUI is designed to offer a friendly interface between user and parallel computing engine.

8.2 Directions for Future Work

- Although the typical components are implemented, there are vast components with various models in power electrical network, which can be implemented step by step to build up a complete EMT simulation environment on the massively parallelism.

- HVDC transmission grid is important to modern energy system, which can be simulated and extended with both behavior-based and physics-based models.
- Other than LU decomposition, various parallel linear system solving methods can be developed and implemented on GPU-based massively parallel computing system to enhance the accuracy and performance of EMT simulation.
- The shattering decomposition proposed in this work can be used on other types of parallel platforms potentially, such as multi-core CPU and distributed parallel system, since network partition is one of the keys to parallel computation.

Bibliography

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

[2] M. D. Omar Faruque et al., “Real-time simulation technologies for power systems design, testing, and analysis,” *IEEE Power Energy Technol. Syst. J.,* vol. 2, no. 2, pp. 63-73, Jun. 2015.

[3] H. W. Dommel and W. S. Meyer, “Computation of electromagnetic transients,” *Proc. IEEE,* vol. 62, no. 7, pp. 983-993, Apr. 1974.

[4] J. Mahseredjian, V. Dinavahi, J. A. Martinez “Simulation tools for electromagnetic transients in power systems: overview and challenges,” *IEEE Trans. on Power Delivery,* vol. 24, no. 3, pp. 1657-1669, July 2009.

[5] CAN/AM EMTP Users Group, *Alternative Transients Program (ATP) Rule Book,* 2000.

[6] Manitoba HVDC Research Centre, *EMTDC User's Guide,* Winnipeg, MB, Canada, 2003.

[7] EMTP-RV website: www.emtp-software.com

[8] H. W. Dommel, *EMTP Theory Book,* Bonneville Power Administration,1984.

[9] H. K. Lauw and W. Scott Meyer, “Universal machine modeling for the representation of rotating electric machinery in an electromagnetic transients program,” *IEEE Trans. Power App. Syst.,* vol. 101, no. 6, pp. 1342-1350, Jun. 1982.

[10] 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 App. Syst.,* vol. PAS-101, no. 6, pp. 1369?1378, Jun. 1982.

[11] 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, July 1999.

[12] J. W. Demmel, J. R. Gilbert, and X. S. Li "An asynchronous parallel supernodal algorithm for sparse gaussian elimination," *SIAM J. Matrix Analysis and Applications,* vol. 20, no. 4, pp. 915 - 952, 1999.

[13] E. Lindholm, J. Nickolls, S. Oberman and J. Montrym, "NVIDIA Tesla: A Unified Graphics and Computing Architecture," *IEEE Micro,* vol. 28, no. 2, pp. 39-55, Mar.-Apr. 2008.

[14] D. Blythe, "Rise of the graphics processor," *Proc. IEEE,* vol. 96, no. 5, pp. 761-778, May 2008.

[15] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, "GPU computing," *Proc. IEEE,* vol. 96, no. 5, pp. 879-899, May 2008.

[16] A. Gopal, D. Niebur and S. Venkatasubramanian, "DC power flow based contingency analysis using graphics processing units," *IEEE Power Tech.,* pp. 731-736, Lausanne, 2007.

[17] V. Jalili-Marandi and V. Dinavahi, "Large-scale transient stability simulation on graphics processing units," *IEEE Power & Energy Society General Meeting,* pp. 1-6, Calgary, AB, 2009.

[18] N. Garcia, "Parallel power flow solutions using a biconjugate gradient algorithm and a Newton method: a GPU-based approach," *IEEE PES General Meeting,* pp. 1-4, Minneapolis, MN, 2010

[19] V. Jalili-Marandi and V. Dinavahi, "SIMD-Based large-scale transient stability simulation on the graphics processing unit," *IEEE Trans. Power Syst.,* vol. 25, no. 3, pp. 1589-1599, Aug. 2010.

[20] V. Jalili-Marandi, Z. Zhou and V. Dinavahi, "Large-scale transient stability simulation of electrical power systems on parallel GPUs," *IEEE Trans. Parallel Distrib. Syst.,* vol. 23, no. 7, pp. 1255-1266, Jul. 2012.

[21] H. Karimipour and V. Dinavahi, "Extended Kalman filter-Based parallel dynamic state estimation," *IEEE Trans. Smart Grid,* vol. 6, no. 3, pp. 1539-1549, May 2015.

[22] H. Karimipour and V. Dinavahi, “Parallel relaxation-based joint dynamic state estimation of large-scale power systems,” *IET Gener. Transm. Distrib.,* vol. 10, no. 2, pp. 452-459, 2 4 2016.

[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] J. K. Debnath, A. M. Gole and W. K. Fung, “Graphics-processing-unit-based acceleration of electromagnetic transients simulation,” *IEEE Trans. Power Del.,* vol. 31, no. 5, pp. 2036-2044, Oct. 2016.

[25] V. Roberge, M. Tarbouchi and F. Okou, “Parallel power flow on graphics processing units for concurrent evaluation of many networks,” *IEEE Trans. Smart Grid,* vol. PP, no. 99, pp. 1-10, 2015.

[26] G. M. Amdahl, “Validity of the single-processor approach to achieving large scale computing capabilities,” *Proc. AFIPS Spring Joint Computer Conf.,* vol. 30, pp. 483-485, Atlantic City, N.J., Apr. 18-20, 1967.

[27] John L. Gustafson , “Reevaluating Amdahl’s law,” *Communications of the ACM,* vol. 31, pp. 532-533, 1988.

[28] N. Fröhlich, B. M. Riess, U. A. Wever and Q. Zheng, “A new approach for parallel simulation of VLSI circuits on a transistor level,” *IEEE Trans. Circuits Syst. I: Fundamental Theory and Applications,* vol. 45, no. 6, pp. 601-613, Jun. 1998

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

[30] H. Karimipour and V. Dinavahi, “Parallel domain-decomposition-dased distributed state estimation for large-scale power systems,” *IEEE Trans. Ind. Appl.,* vol. 52, no. 2, pp. 1265-1269, March-April 2016.

[31] Y. Liu and Q. Jiang, “Two-stage parallel waveform relaxation method for large-scale power system transient stability simulation,” *IEEE Trans. Power Syst.,* vol. 31, no. 1, pp. 153-162, Jan. 2016.

[32] G. Kron, *Diakoptics: The Piecewise Solution of Large Scale Systems,* London: MacDonald, 1963.

[33] H. H. Happ, "Diakoptics and piecewise methods," *IEEE Trans. Power App. Syst.,* vol. PAS-89, no. 7, pp. 1373-1382, Sept. 1970.

[34] D. Montenegro, G. A. Ramos and S. Bacha, "Multilevel a-diakoptics for the dynamic power-flow simulation of hybrid power distribution systems," *IEEE Trans. Ind. Informat.* vol. 12, no. 1, pp. 267-276, Feb. 2016.

[35] NVIDIA® Corp., *Whitepaper NVIDIA Tesla P100,* 2016.

[36] NVIDIA® Corp., *CUDA C Programming Guide,* Sept. 2015.

[37] NVIDIA® Corp., *Whitepaper NVIDIA GeForce GTX 1080,* 2016.

[38] G. Ballard, J. Demmel, O. Holtz and O. Schwartz, "Minimizing communication in linear algebra," *SIAM J. Matrix Analysis and Applications,* vol. 32, no. 3, pp. 866 - 901, Sept. 2011.

[39] V. Volkov and J. W. Demmel, "Benchmarking GPUs to tune dense linear algebra," *SC2008 Proc. ACM/IEEE Conf. on Supercomputing,* pp. 1-11, Austin, TX, USA, 15-21 Nov. 2008.

[40] H. K. Lauw, "Interfacing for universal multi-machine system modeling in an electromagnetic transients program," *IEEE Trans. Power App. Syst.,* vol. 104, no. 9, pp. 2367-2373, Sept. 1985.

[41] 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.

[42] S. Priyadarshi, C. S. Saunders, N. M. Kriplani, H. Demircioglu, W. R Davis, P. D. Franzon and M. B. Steer, "Parallel transient simulation of multiphysics circuits using delay-based partitioning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.,* vol. 31, NO. 10, pp. 1522 - 1535, Oct. 2012.

[43] G.S. Murthy, M. Ravishankar, M. M. Baskaran and P. Sadayappan, "Optimal loop unrolling for GPGPU programs," in *Proc. IEEE Int. Symp. on Parallel Distrib. Process.,* pp.1-11, 19-23 Apr. 2010

[44] J.S Przemieniecki, "Matrix structural analysis of substructures," *AIAA J.,* 1 (1963) 138-147.

[45] H.A. Schwarz, *Gesammelte Mathematiche Abhandlungen,* vol. 2, Springer, Berlin, 1890, pp. 133?143.

[46] A. Ammous et al., "Electrothermal modeling of IGBTs: application to short-circuit conditions," *IEEE Trans. Power Electron.,* vol. 15, no. 4, pp. 778-790, Jul 2000.

[47] W. Zhou, X. Zhong and K. Sheng, "High temperature stability and the performance degradation of SiC MOSFETs," *IEEE Trans. Power Electron.,* vol. 29, no. 5, pp. 2329-2337, May 2014.

[48] A. Subbiah and O. Wasynczuk, "Computationally efficient simulation of high-frequency transients in power electronic circuits," *IEEE Trans. Power Electron.,* vol. 31, no. 9, pp. 6351-6361, Sept. 2016.

[49] Synopsys Inc., "Saber Industry Standard for Multi-Domain and Mixed-signal Simulation," 2007.

[50] L. Johnson, "Saber-MATLAB Integrations: Enabling Virtual HW/SW Co-Verification," Synopsys, Inc., June 2004

[51] "Analog and mixed signal simulation," Cadence Design Systems, Inc.

[52] O. A. Ahmed and J. A. M. Bleijs, "Pspice and simulink co-simulation for high efficiency DC-DC converter using SLPS interface software," *5th IET International Conference on Power Electronics, Machines and Drives (PEMD 2010),* Brighton, UK, pp. 1-6. 2010.

[53] M. Pahlevaninezhad, P. Das, G. Moschopoulos and P. Jain, "Sensorless control of a boost PFC AC/DC converter with a very fast transient response," *2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC),* Long Beach, CA, USA, 2013, pp. 356-360.

[54] Synopsys, Inc. "Saber® Accelerates Robust Design," vol.5, June 2007

[55] A. R. Hefner and D. M. Diebolt, "An experimentally verifed IGBT model implemented in the Saber circuit simulator," *IEEE Trans. Power Electron.,* vol. 9, no. 5, pp. 532-542, Sept. 1994.

[56] A. R. Hefner, "Modeling buffer layer IGBTs for circuit simulation", *IEEE Trans. Power Electron.,* vol. 10, no. 2, pp. 111-123, Mar. 1995.

[57] C. L. Ma and P. O. Lauritzen, "A simple power diode model with forward and reverse recovery," *IEEE Trans. Power Electron.,* vol. 8, no. 4, pp. 342-346, 1993.

[58] S. Debnath, J. Qin, B. Bahrani, M. Saeedifard and P. Barbosa, "Operation, control, and applications of the modular multilevel converter: a review, " *IEEE Trans. Power Electron.,* vol.30, no. 1, pp. 37-53, Jan. 2015.

[59] D. C. Ludois and G. Venkataramanan, "Simplified terminal behavioral model for a modular multilevel converter," *IEEE Trans. on Power Electron.,* vol.29, no. 4, pp. 1622-1631, Apr. 2014.

[60] H. Peng, M. Hagiwara and H. Akagi, "Modeling and analysis of switching-ripple voltage on the DC link between a diode rectifier and a modular multilevel cascade inverter (MMCI)," *IEEE Trans. on Power Electron.,* vol.28, no. 1, pp. 75-84, Jan. 2013.

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

[62] H. Saad et al., "Dynamic averaged and simplified models for MMC-Based HVDC transmission systems", *IEEE Trans. on Power Del.,* vol. 28, no. 3, pp. 1723-1730, Apr. 2013.

[63] M. Hagiwara and H. Akagi, "Control and experiment of pulsewidth-modulated modular multilevel converters," *IEEE Trans. on Power Electron.,* vol.24, no. 7, pp. 1737-1746, July 2009.

[64] H. Akagi, S. Inoue and T. Yoshii, "Control and performance of a transformerless cascade PWM STATCOM with star configuration," *IEEE Trans. on Ind. Appl.,* vol. 43, no. 4, pp. 1041-1049, July-Aug. 2007.

[65] G. P. Adam, O. Anaya-Lara, G. M. Burt, D. Telford, B. W. Williams and J. R. Mcdonald, "Modular multilevel inverter: pulse width modulation and capacitor balancing technique," *IET Power Electronics,* vol. 3, no. 5, pp. 702-715, Sept. 2010.

[66] W. Wang, Z. Shen and V. Dinavahi, "Physics-based device-level power electronic circuit hardware emulation on FPGA," *IEEE Transactions on Industrial Informatics,* vol. 10, no. 4, pp. 2166-2179, Nov. 2014.

[67] S. Yan; Z. Zhou; V. Dinavahi, "Large-scale nonlinear device-level power electronic circuit simulation on massively parallel graphics processing architectures," *IEEE Transactions on Power Electronics,* vol.PP, no.99, pp.1-1., 2017

[68] Ó. Jiménez, Ó. Luca, I. Urriza, L. A. Barragan, D. Navarro and V. Dinavahi "Implementation of an FPGA-based online hardware-in-the-Loop emulator using high-level synthesis tools for resonant power converters applied to induction heating appliances," *IEEE Trans. on Ind. Electron.,* vol. 62, no. 4, pp. 2206-2214, April 2015.

[69] A. Myaing, M. O. Faruque, V. Dinavahi, C. Dufour, "Comparison of insulated gate bipolar transistor models for FPGA-based real-time simulation of electric drives and application guideline," *IET Power Electronics,* vol. 5, no. 3, pp. 293-303, March 2012.

[70] E. Solomonik and J. Demmel, "Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms," *EECS Technical Report,* EECS-2011-10, UC Berkeley, Feb. 2011.

[71] L. Chen, O. Villa, S. Krishnamoorthy and G. R. Gao, "Dynamic load balancing on single- and multi-GPU systems," *Proc. IEEE Int. Symp. Parallel Distrib. Process.,* Atlanta, GA, 2010, pp. 1-12.

A Appendix

A.1 Nonlinear Power Diode

A.1.1 Model Formulation

Detailed device-level modeling of power diodes covers a wide range of circuit operating conditions since it includes equations for drift and diffusion of electrons and holes. However, different from conventional detailed models which are too complicated to simulate, this paper employs a simplified physics-based model containing p-i-n structure suitable for power diode operating condition of high-voltage and fast switching [57].

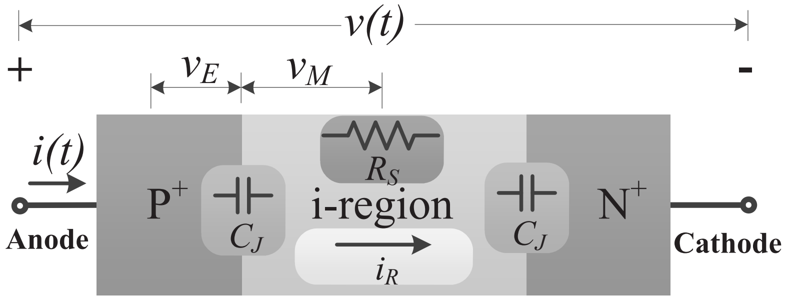
The physical structure of a power diode is shown in Fig. A.1(b). The reverse recovery happens when turning off a forward conducting diode rapidly as described by the following equations:

$$i_R(t) = \frac{q_E(t) - q_M(t)}{T_M}, \tag{A.1}$$

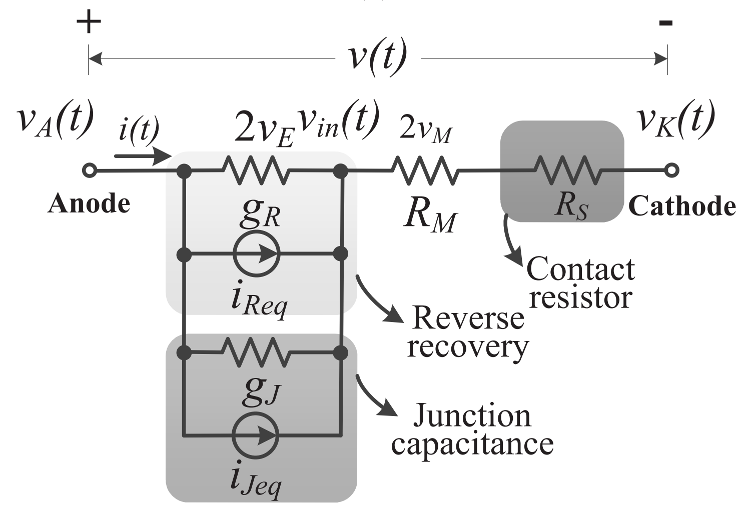
$$0 = \frac{dq_M(t)}{dt} + \frac{q_M(t)}{\tau} - \frac{q_E(t) - q_M(t)}{T_M}, \tag{A.2}$$

$$q_E(t) = I_S\tau(e^{\frac{v_E(t)}{V_T}} - 1), \tag{A.3}$$

where $i_R(t)$ is the diffusion current in i-region, $q_E(t)$ represents charge variable in the junction area, $q_M(t)$ represents charge variable in the middle of i-region, T_M is the diffusion transit time across i-region, τ is the lifetime of recombination, I_S is the diode saturation current constant, v_E is the junction voltage, and V_T is the thermal voltage constant.



(a)



(b)

Figure A.1: (a) Physical structure of power diode, (b) Discretized and linearized equivalent circuit of power diode.

The voltage drop across i-region $v_M(t)$ is described as

$$v_M(t) = \frac{V_T T_M i(t)}{q_M(t)}. \tag{A.4}$$

The voltage across diode, $v(t)$, is expressed as

$$v(t) = 2v_M(t) + 2v_E(t) + R_S\left[i_E(t) + \frac{dq_J(t)}{dt}\right], \tag{A.5}$$

where R_S is the contact resistance presented as an internal resistance and the charge of junction capacitance in the capacitance C_J is given as

$$q_J(t) = \int C_J(t)d(2v_E). \tag{A.6}$$

The expression of junction capacitance $C_J(t)$ is given as follows:

$$C_J(t)=\begin{cases} \dfrac{C_{J0}}{(1-\dfrac{2v_E(t)}{\phi_B})^m} & v_E<\dfrac{\phi_B}{4} \\ \left[\dfrac{2^{m+2}mv_E(t)}{\phi_B}-(m-1)2^m\right]C_{J0} & v_E\geq\dfrac{\phi_B}{4} \end{cases}, \tag{A.7}$$

where C_{J0} is the zero-biased junction capacitance, ϕ_B is the built-in potential and m is the junction grading coefficient.

A.1.2 Model Discretization and Linearization

After discretization by Trapezoidal rule, the differential term $\frac{dq_M(t)}{dt}$ in (A.2) is expressed as

$$q_M=\frac{\Delta t\cdot q_E(t)}{2T_M(1+\frac{k_1\Delta t}{2})}+\frac{q_{\text{hist}}(t-\Delta t)}{1+\frac{k_1\Delta t}{2}}, \tag{A.8}$$

where the history term is given as

$$q_{\text{hist}}(t-\Delta t)=\frac{\Delta t}{2T_M}q_E(t-\Delta t)-\frac{k_1\Delta t}{2}q_M(t-\Delta t). \tag{A.9}$$

The equivalent reverse recovery current i_{Req}, as shown in Fig. A.1(c), is given as

$$i_{\text{Req}}=k_2I_S\tau(e^{\frac{v_E(t)}{v_T}}-1)-\frac{q_{\text{hist}}(t-\Delta t)}{T_M(1+\frac{k_1\Delta t}{2})}-2v_E(t)g_R, \tag{A.10}$$

where g_R is the dynamic conductance defined as

$$g_R=\frac{1}{2v_T}k_2I_S\tau e^{\frac{v_E(t)}{V_T}}. \tag{A.11}$$

Similarly, the equivalent junction capacitance current i_{Jeq} is obtained as

$$i_{J\text{eq}}=i_J(t)-2v_E(t)g_J, \tag{A.12}$$

where the equivalent junction conductance g_J is given as

$$g_J=\frac{2}{\Delta t}C_J(t). \tag{A.13}$$

The discretized and linearized system (Diode-DLE) shown in Fig. A.1(c) can be obtained as follows:

$$\boldsymbol{G}^{\text{Diode}}\cdot\boldsymbol{V}^{\text{Diode}}=\boldsymbol{I}_{\text{eq}}^{\text{Diode}}, \tag{A.14}$$

where

$$\boldsymbol{G}^{\text{Diode}} = \begin{bmatrix} g_R+g_J & -g_R-g_J & 0 \\ -g_R-g_J & g_R+g_J+\frac{1}{R_M+R_S} & -\frac{1}{R_M+R_S} \\ 0 & -\frac{1}{R_M+R_S} & \frac{1}{R_M+R_S} \end{bmatrix}, \tag{A.15}$$

$$\boldsymbol{V}^{\text{Diode}} = [v_A, \quad v_{in}, \quad v_K]^T, \tag{A.16}$$

$$\boldsymbol{I}_{\text{eq}}^{\text{Diode}} = [-i_{R\text{eq}}-i_{J\text{eq}}, \quad i_{R\text{eq}}+i_{J\text{eq}} \quad ,0]^T. \tag{A.17}$$

Applying Newton-Raphson method, the next iterate values $\boldsymbol{V}^{\text{Diode}^{(n+1)}}$ can be updated by previous nth iterate values until the solution is converged.

A.2 Nonlinear Physics-based IGBT

A.2.1 Model Formulation

Based on the Hefner's physics-based model [56], the IGBT is described as the combination of a bipolar transistor and a MOSFET. Since these internal devices are differently structured from standard microelectronic devices, a regional approach is adopted to identify the phenomenological circuit of IGBT as shown in Fig. A.2(a). An analog equivalent circuit, shown in Fig. A.2(b), makes it possible to implement the model in circuit simulators by replacing the BJT with base and collector current sources and MOSFET with a current source, which represents the currents between each of the terminals and internal nodes in terms of nonlinear functions.

A.2.1.1 Current Sources

The steady-state collector current i_{css} of BJT is formulated as

$$i_{css} = \frac{i_T}{1+b} + \frac{4bD_pQ}{(1+b)W^2}, \tag{A.18}$$

where b is the ambipolar mobility ratio, D_p is the hole diffusivity, W is the quasi-neutral base width, Q is the instantaneous excess-carrier base charge and the anode current i_T is shown as follows:

$$i_T = \frac{v_{ae}}{r_b}. \tag{A.19}$$

The base resistance r_b in (A.19) is expressed as

$$r_b = \begin{cases} \frac{W}{q\mu_n AN_B} & v_{eb} \leq 0 \\ \frac{W}{q\mu_{\text{eff}} An_{\text{eff}}} & v_{eb} > 0 \end{cases}, \tag{A.20}$$

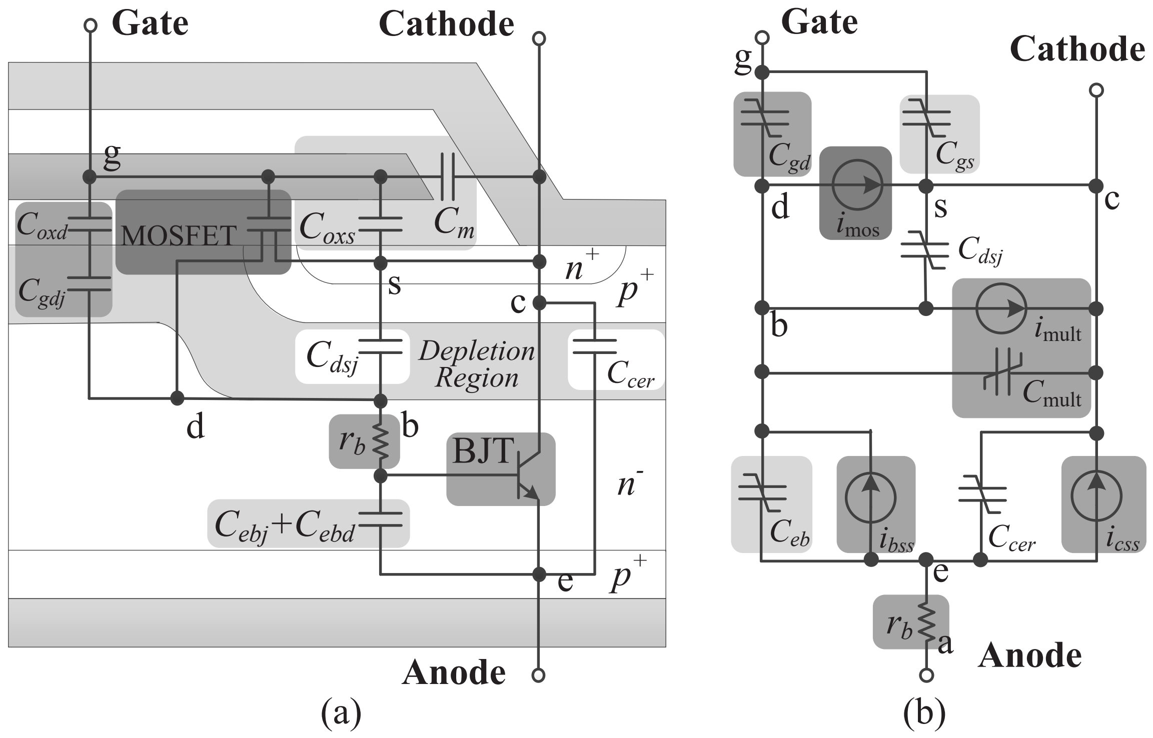


Figure A.2: (a) Phenomenological structure of IGBT, (b) Analog quivalent circuit of IGBT.

where μ_n and μ_{eff} stand for electron mobility and effective mobility, n_{eff} is the effective doping concentration, q is the electron charge, N_B is the base doping concentration and A is the device active area. The steady-state base current i_{bss} is caused by the decay of excess base charge of recombination in the base and electron injection in the emitter, and is expressed as

$$i_{bss} = \frac{Q}{\tau_{HL}} + \frac{4Q^2 N_{scl}^2 i_{sne}}{Q_B^2 n_i^2}, \tag{A.21}$$

where τ_{HL} is the base high-level lifetime, N_{scl} is the collector-base space concentration, i_{sne} is the emitter electron saturation current, n_i is the intrinsic carrier concentration and Q_B represents the background mobile carrier base charge. The MOSFET channel current i_{mos} is expressed as

$$i_{\text{mos}} = \begin{cases} 0 & v_{gs} < v_T \\ K_p(v_{gs} - v_T)v_{ds} - \frac{K_p v_{ds}^2}{2} & v_{ds} \leq v_{gs} - v_T \\ \frac{K_p(v_{gs}-v_T)^2}{2} & v_{ds} > v_{gs} - v_T \end{cases}, \tag{A.22}$$

where K_p is the MOSFET transconductance parameter, v_{gs} is the gain-source voltage and v_T is the MOSFET channel threshold voltage [?]. In addition, due to thermal generation in the depletion region and carrier multiplication which is a key factor to determine the avalanche breakdown voltage and the leakage current, the avalanche multiplication current i_{mult}, shown in Fig. A.2(b) is given as

$$i_{\text{mult}} = (M-1)(i_{\text{mos}} + i_{css} + i_{c\_cer}) + M i_{\text{gen}}, \tag{A.23}$$

where M stands for the avalanche multiplication factor.

A.2.1.2 Charges and Capacitances

The gate-source capacitance C_{gs} in the analog model is a constant, and its charge Q_{gs} is given as

$$Q_{gs} = C_{gs} v_{gs}. \tag{A.24}$$

The gate-drain capacitance C$_{gd}$ is expressed as

$$C_{gd} = \begin{cases} C_{oxd} & v_{ds} \leq v_{gs} - v_{Td} \\ \frac{C_{gdj} C_{oxd}}{C_{gdj} + C_{oxd}} & v_{ds} > v_{gs} - v_{Td} \end{cases}, \tag{A.25}$$

where v_{Td} is the gate-drain overlap depletion threshold voltage, C_{oxd} is the gate-drain capacitance. The gate-drain overlap depletion capacitance C_{gdj} is given as

$$C_{gdj} = \frac{A_{gd} \epsilon_{si}}{W_{gdj}}, \tag{A.26}$$

where A_{gd} is the gate-drain overlap area, ϵ_{si} is the silicon dielectric constant and W_{gdj} is the gate-drain overlap depletion width. The charge of C_{gd} has the expression as

$$Q_{gd} = \begin{cases} C_{oxd} v_{dg} & v_{ds} \leq v_{gs} - v_{Td} \\ \\ \frac{q N_B \epsilon_{si} A_{gd}^2}{C_{oxd}} [\frac{C_{oxd} W_{gdj}}{\epsilon_{si} A_{gd}} - & \\ \quad ln(1 + \frac{C_{oxd} W_{gdj}}{\epsilon_{si} A_{gd}})] - & v_{ds} > v_{gs} - v_{Td} \\ \quad\quad C_{oxd} v_{Td} & \end{cases}. \tag{A.27}$$

Similarly, the drain-source depletion capacitance C_{dsj}, related to the active area $(A - A_{gd})$ and drain-source depletion width W_{dsj}, is given as

$$C_{dsj} = \frac{(A - A_{gd}) \epsilon_{si}}{W_{dsj}}, \tag{A.28}$$

and its charge Q_{ds} is expressed as

$$Q_{ds} = A_{ds} \sqrt{2 \epsilon_{si} (v_{ds} + 0.6) q N_{scl}}. \tag{A.29}$$

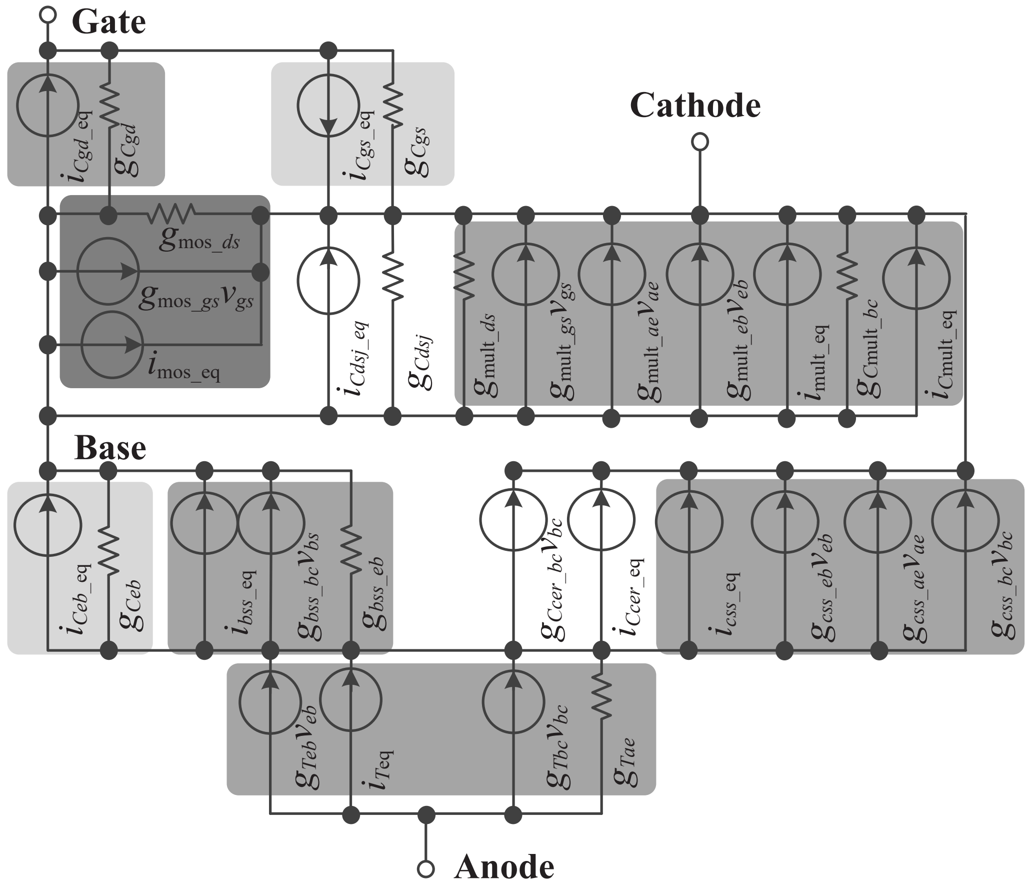


Figure A.3: Discretized and linearized equivalent circuit of IGBT (IGBT-DLE).

The emitter-base capacitance C$_{eb}$ is solved from $\frac{\partial Q_{eb}}{\partial V_{eb}}$ as

$$C_{eb} = -\frac{qN_B\epsilon_{si}A^2}{Q - Q_{bi}}. \tag{A.30}$$

The collector-emitter redistribution capacitance C$_{cer}$ is solved from the ambipolar diffusion equation as

$$C_{cer} = \frac{QC_{bcj}}{3Q_B}, \tag{A.31}$$

where Q$_B$ is the background mobile carrier base charge and C$_{bcj}$ is the base-collector depletion capacitance. The carrier multiplication charge and capacitance relating to C_{cer} are given as

$$Q_{\text{mult}} = (M-1)Q_{ce} \qquad \text{and} \qquad C_{\text{mult}} = (M-1)C_{cer}. \tag{A.32}$$

A.2.2 Model Discretization and Linearization

After applying the Newton-Raphson method on four current sources and the conductivity-modulated base resistance r_b, the analog equivalent circuit model (IGBT-AE) shown in Fig. A.2(b), containing five nonlinear and time varying elements, is transferred into discretized and linearized equivalent circuits (IGBT-DLE), as shown in Fig. A.3. The iterative equations of i_{mos}, i_T, i_{css}, i_{bss}, i_{mult} for the $(n+1)^{th}$ iteration are obtained as follows,

$$\begin{aligned}
i_{\text{mos}}^{n+1} =& i_{\text{mos\_eq}}^{n} + g_{\text{mos\_}gs}^{n} v_{gs}^{n+1} + g_{\text{mos\_}ds}^{n} v_{ds}^{n+1}, && \text{(A.33)}\\
i_{T}^{n+1} =& i_{T\text{eq}}^{n} + g_{Tae}^{n} v_{ae}^{n+1} + g_{Tbc}^{n} v_{bc}^{n+1} + g_{Teb}^{n} v_{eb}^{n+1}, && \text{(A.34)}\\
i_{css}^{n+1} =& i_{css\text{\_eq}}^{n} + g_{css\_bc}^{n} v_{bc}^{n+1} + g_{css\_ae}^{n} v_{ae}^{n+1} \\
&+ g_{css\_eb}^{n} v_{eb}^{n+1}, && \text{(A.35)}\\
i_{bss}^{n+1} =& i_{bss\text{\_eq}}^{n} + i_{bss\_eb}^{n} v_{eb}^{n+1} + g_{bss\_bc}^{n} v_{bc}^{n+1}, && \text{(A.36)}\\
i_{\text{mult}}^{n+1} =& i_{\text{mult\_eq}}^{n} + g_{\text{mult\_}ds}^{n} v_{ds}^{n+1} + g_{\text{mult\_}ds}^{n} v_{ds}^{n+1} \\
&+ g_{\text{mult\_}ae}^{n} v_{ae}^{n+1} + g_{\text{mult\_}eb}^{n} v_{eb}^{n+1}. && \text{(A.37)}
\end{aligned}$$

Applying KCL to the nodes Gate, Collector, Base and Emitter, results in the following nodal equation:

$$\boldsymbol{G}^{\text{IGBT}} \cdot \boldsymbol{V}^{\text{IGBT}} = \boldsymbol{I}_{\text{eq}}^{\text{IGBT}}, \tag{A.38}$$

where

$$\boldsymbol{V}^{\text{IGBT}} = [v_c \quad v_g \quad v_a \quad v_d \quad v_e]^T, \tag{A.39}$$

$\boldsymbol{I}_{\text{eq}}^{\text{IGBT}}$ is given in (A.40) and the 5×5 conductance matrix, $\boldsymbol{G}^{\text{IGBT}}$, is given in (A.41).

$$\boldsymbol{I}_{\text{eq}}^{\text{IGBT}} = \begin{bmatrix} i_{c\text{\_eq}} \\ i_{g\text{\_eq}} \\ i_{a\text{\_eq}} \\ i_{d\text{\_eq}} \\ i_{e\text{\_eq}} \end{bmatrix} = \begin{bmatrix} i_{\text{mult\_eq}} + i_{C\text{mult\_eq}} + i_{css\text{\_eq}} + i_{C\_cer\text{\_eq}} + i_{Cgs\text{\_eq}} + i_{Cdsj\text{\_eq}} + i_{\text{mos\_eq}} \\ -i_{Cgs\text{\_eq}} + i_{Cdg\text{\_eq}} \\ -i_{T\text{eq}} \\ i_{C\_ceb\text{\_eq}} + i_{bss\text{\_eq}} - i_{\text{mos\_eq}} - i_{Cdg\text{\_eq}} - i_{Cdsj\text{\_eq}} - i_{\text{mult\_eq}} - i_{C\text{mult\_eq}} \\ i_{css\text{\_eq}} - i_{bss\text{\_eq}} - i_{C\_cer\text{\_eq}} - i_{C\_ceb\text{\_eq}} + i_{T\text{eq}} \end{bmatrix} \tag{A.40}$$

$$\boldsymbol{G}^{\text{IGBT}} = \begin{bmatrix}
\begin{array}{c} g_{\text{mult\_}ds}+g_{\text{mult\_}gs} \\ +g_{\text{mult\_}bc}+g_{css\_bc} \\ +g_{C\_cer\_bc}+g_{Cgs} \\ +g_{Cdsj}+g_{\text{mos\_}ds} \\ +g_{\text{mos\_}gs} \end{array} & \begin{array}{c} -g_{\text{mult\_}gs}-g_{Cgs} \\ -g_{\text{mos\_}gs} \end{array} & -g_{\text{mult\_}ae}-g_{css\_ae} & \begin{array}{c} -g_{\text{mult\_}ds}-g_{C\text{mult\_}bc} \\ -g_{css\_bc}+g_{css\_eb} \\ -g_{C\_cer\_bc}-g_{dsj} \\ -g_{\text{mos\_}ds}+g_{\text{mult\_}eb} \end{array} & \begin{array}{c} -g_{\text{mult\_}ae}-g_{css\_ae} \\ +g_{css\_eb}-g_{\text{mult\_}eb} \end{array} \\
-g_{Cgs} & g_{Cgs}+g_{Cdg} & 0 & -g_{Cdg} & 0 \\
-g_{Tbc} & 0 & g_{Tae} & g_{Tbc}-g_{Teb} & -g_{Tae}+g_{Teb} \\
\begin{array}{c} g_{bss\_bc}-g_{\text{mos\_}gs} \\ -g_{\text{mos\_}ds}-g_{C\_dsj} \\ -g_{\text{mult\_}ds}-g_{\text{mult\_}gs} \\ -g_{C\text{mult\_}bc} \end{array} & \begin{array}{c} g_{\text{mos\_}gs}-g_{Cdg} \\ +g_{\text{mult\_}gs} \end{array} & g_{\text{mult\_}ae} & \begin{array}{c} g_{Ceb}-g_{bss\_bc} \\ +g_{bss\_eb}+g_{\text{mos\_}ds} \\ +g_{Cdg}+g_{Cdsj} \\ +g_{\text{mult\_}ds}-g_{\text{mult\_}eb} \\ -g_{C\text{mult\_}bc} \end{array} & \begin{array}{c} -g_{Ceb}+g_{bss\_eb} \\ -g_{\text{mult\_}ae}+g_{\text{mult\_}eb} \end{array} \\
\begin{array}{c} -g_{css\_bc}-g_{C\_cer\_bc} \\ -g_{bss\_bc}+g_{Tbc} \end{array} & 0 & g_{css\_ae}-g_{Tae} & \begin{array}{c} g_{css\_bc}-g_{css\_eb} \\ +g_{C\_cer\_bc}-g_{bss\_eb} \\ +g_{bss\_bc}-g_{Ceb} \\ -g_{Tbc}+g_{Teb} \end{array} & \begin{array}{c} -g_{css\_ae}+g_{css\_eb}+ \\ g_{bss\_eb}+g_{Ceb}+ \\ g_{Tae}-g_{Teb} \end{array}
\end{bmatrix} \tag{A.41}$$

Applying Newton-Raphson method to solve the nonlinear matrix equation (A.38), $\Delta\boldsymbol{V}^{\text{IGBT}}$ is obtained to update $\boldsymbol{V}^{\text{IGBT}}$ iteratively. Therefore, the iterative equation is given as

$$\boldsymbol{G}^{\text{IGBT}(n)}\Delta\boldsymbol{V}^{\text{IGBT}(n+1)} = -\boldsymbol{I}^{\text{IGBT}(n)}, \tag{A.42}$$

where

$$\boldsymbol{I}^{\text{IGBT}(n)} = \boldsymbol{G}^{\text{IGBT}(n)}\boldsymbol{V}^{\text{IGBT}(n)} - \boldsymbol{I}_{\text{eq}}^{\text{IGBT}(n)}. \tag{A.43}$$

B
Appendix

B.1 Parameters for Case Study A

The parameters for Case Study A are as follows:

1. Synchronous machine parameters: 10MVA, 3.5kV, Y-connected, field current: 5A, 2 poles, 60Hz, moment of inertia: 4Mkg·m^2/rad and damping: 50kg·m/s/rad.

2. Transmission line parameters: Line$_1$: three conductors, resistance: 0.0583/km, diameter: 3.105cm, line length: 50km; Line$_2$: three conductors, resistance: 0.0583/km, diameter: 3.105cm, line length: 150km. Line geometry: flat horizontal phase spacing; horizontal distance between adjacent phases = 4.87m; vertical distance: phases *a* to ground, *c* to ground = 30m, phase *b* to ground = 28m, and shield wire to tower arm = 6m.

3. Transformer parameters: T$_1$: 10MVA, 3.5kV/22kV, X$_{\text{leakage}}$ = 6.89e^{-3}pu, Y-Y connection; T$_2$: 10MVA, 22kV/3.5kV, X$_{\text{leakage}}$ = 0.192pu and Y-Y connection.

4. Arrester parameters: V$_{\text{ref}}$ = 5kV, multiplier(p) = 1.1 and exponent(q) = 26.

5. Loads parameters: R = 1kΩ and L = $1mH$.

B.2 Parameters for Case Study B

The parameters for the Scale-1 39-bus test system of Case Study B are as follows:

1. Synchronous machine parameters: 1000MVA, 22kV, Y-connected, field current: 2494A, 2 poles, 60Hz, moment of inertia: 4Mkg·m^2/rad and damping: 6780kg·m/s/rad.

2. ULM transmission line parameters: Line 1 - 35: three conductors, resistance: 0.0583/km, diameter: 3.105cm, line length: 50km (Line 5, 6, 7, 8, 15, 16, 18, 19, 23, 27, 29, 30, 31, 35), 150km (Line 2, 3, 4, 9, 10, 11, 13, 14, 20, 21, 22, 24, 25, 26, 32, 33) and 500km (Line 1, 12, 17, 28, 34). Line geometry: flat horizontal phase spacing; horizontal distance between adjacent phases = 4.87m; vertical distance: phases *a* to ground, *c* to ground = 30m, phase *b* to ground = 28m, and shield wire to tower arm = 6m.

3. Transformer parameters: 1000MVA, 22V/220kV, X$_{\text{leakage}}$ = 9.24e^{-3}pu and Y-Y connection;

4. Loads parameters: R = 1kΩ and L = $1mH$.

B.3 Parameters for Case Study D

The parameters for MMC of Case Study D are as follows:

1. MMC parameters: DC voltage V_{dc}=1kV, arm inductance L_a=L_b=L_c=150mH, SM capacitance C_m=4mF, carrier frequency f_c=2500Hz, system frequency f_s=60Hz.

2. AC Source parameters: transformer reactance X_T=1.88Ω, V_s=1kV, R_s=5Ω.