A Study of Solution Process Parallelization for an EMT Analysis Program Using OpenMP

Rikido Yonezawa and Taku Noda

Abstract--**Owing to high demand for electromagnetic transient (EMT) simulations of power systems, faster simulations are desired for EMT analysis programs since the circuits simulated are becoming increasingly large and complex. In this paper, we describe solution process parallelization for an EMT analysis program executed on a general-purpose PC with a multi-core processor using OpenMP. If the number of subsystems is greater than the number of available cores, an appropriate task-scheduling algorithm for efficient load balancing is required. Accordingly, we discuss the effect of task-scheduling algorithms on the computation time through a verification using static and dynamic scheduling algorithms. Finally, it is shown that parallel processing of the solution process with six threads using OpenMP in this study achieves over three times faster computation than the conventional sequential computation.**

Keywords: **Electromagnetic transient analysis, Parallel processing, Multi-core processor, OpenMP and Task-scheduling algorithm.**

I. Introduction

ELECTROMAGNETIC transient (EMT) simulations have been used for studies of conventional phenomena such as overvoltages, inrush currents, ferroresonance, and abnormal oscillations. Nowadays, studies of power systems including power-electronics converters such as dc transmission, back-to-back system interconnection (frequency conversion between 50 and 60 Hz), and power conditioning systems for renewable energies are also carried out using EMT simulations since these studies require waveform-level calculations to take into account switchings in power-electronics converters. Owing to the high demand for EMT simulations of power systems, faster simulations are desired for EMT analysis programs since the circuits simulated are becoming increasingly large.

The calculation process in EMTP-based programs consists of the following three main processes: a formulation process to formulate the circuit equations for the circuit to be simulated; a solution process to solve the circuit equations using LU decomposition and forward-backward substitution; and an updating process to update the internal states of the circuit components. The total computation time of the simulation includes the computation time not only of these three processes but also that required to solve the control system and write the simulation results to a file. Although the computation time of each process depends on the type of circuit to be simulated, a large proportion of the computation time is spent in the solution process. Thus, a fast algorithm for the solution process is important for faster simulations.

The following two approaches have been used to accelerate the solution process of EMT analysis programs: sparse matrix methods [1] for the efficient computation of a large sparse matrix given by the circuit equations and parallel processing of the solution process using many computers or multi-core processors. In EMT simulations of a power system, the propagation delay of transmission lines in the power system allows a large power system to be decoupled into several independent subsystems without the use of complicated techniques [2], [3]. Therefore, it is technically straightforward to apply the latter approach to an EMT analysis program, and much work has been carried out using various hardware architectures [4]-[10]. In addition, several studies using multi-core processors installed in general-purpose PCs have been reported owing to their high penetration in recent years [11], [12].

In this paper, we describe solution process parallelization for an EMT analysis program executed on a general-purpose PC with a multi-core processor using OpenMP [13], which is one of the shared-memory parallel programming interfaces. To be more precise, the circuit to be simulated is partitioned into subcircuits by distributed parameter transmission lines, and the solution processes of the circuit equations formulated for each subcircuit are solved in parallel. If the number of subsystems is greater than the number of available cores executing the solution processes, an appropriate task-scheduling algorithm for efficient load balancing is required. Accordingly, we discuss the effect of task-scheduling algorithms on the computation time through a verification using static and dynamic scheduling algorithms. The parallelization is implemented in the EMT analysis program XTAP (eXpandable Transient Analysis Program) [14], [15], and the performance of the parallelization is validated using IEEJ's West-10 benchmark power system model [16], [17], called the West-10 Benchmark System. As a result of the validation, it has been demonstrated that the computation time using a static scheduling algorithm is less than that using a dynamic scheduling algorithm. Moreover, it has been shown that parallel processing of the solution process with six threads using OpenMP in this study achieves over three times faster computation than the conventional sequential computation.

II. Flowchart of Calculation Process in EMT analysis Programs

Fig. 1 shows a flowchart of the EMT analysis program with

R. Yonezawa and T. Noda are with Power Quality Group, Energy Innovation Center, CRIEPI (Central Research Institute of Electric Power Industry), 2-6-1, Nagasaka, Yokosuka, Kanagawa 240-0196, Japan (e-mail: {yonezawa, takunoda}@criepi.denken.or.jp).

Paper submitted to the International Conference on Power Systems Transients (IPST2017) in Seoul, Republic of Korea June 26-29, 2017

solution process parallelization used in this study. As is clear from Fig. 1, the solution process is parallelized but the other processes are executed sequentially. Actually, parallel execution is allowed for the updating process because the internal states of the circuit components can be updated independently. However, the main purpose of this study is to discuss the effect of task-scheduling algorithms on the computation time when the solution process is parallelized. For this reason, this updating process (and other processes except for the solution process) is executed sequentially. When the transmission lines in the circuit to be simulated are represented by the distributed parameter line model, the propagation delay of the line model can completely decouple the circuit into several independent subcircuits as long as the delay is longer than the simulation time step [2]. In this case, the solution processes of each subcircuit can be executed in parallel. There is no communication overhead except for that of the synchronization processes at the beginning and end of the parallel region because of the complete independence of the solution processes of each subcircuit.

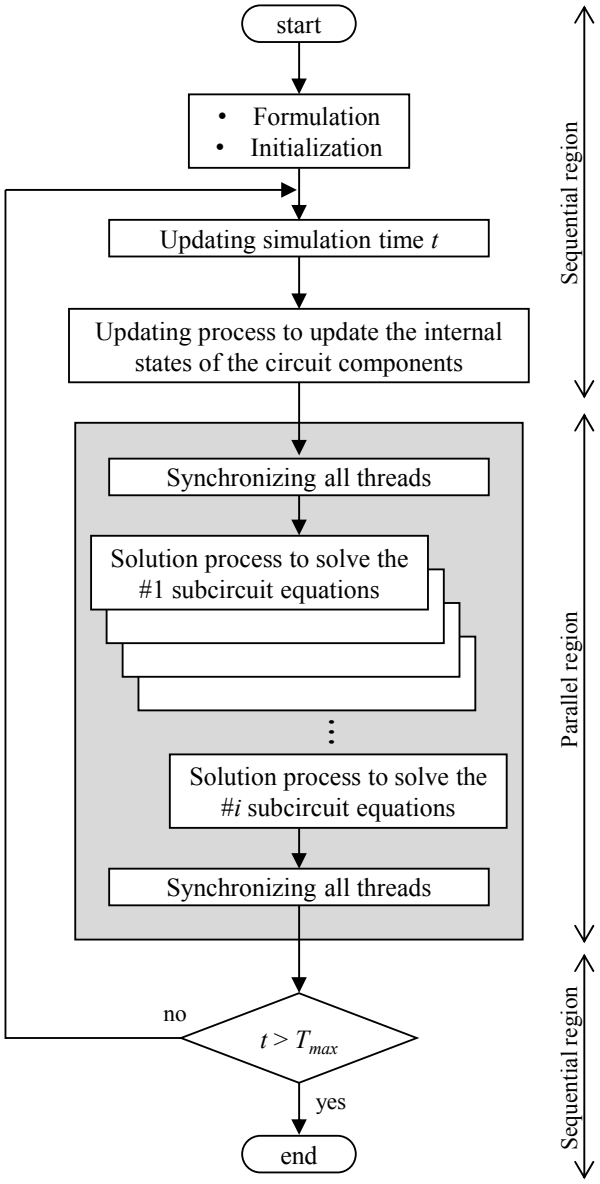

Fig. 1. The flowchart of the calculation process in the EMT analysis program with solution process parallelization in this study.

III. Task-scheduling Algorithms

In this paper, the parallel EMT analysis program shown in Fig. 1 is implemented using OpenMP, which is a standard application programming interface (API) for multi-platform shared-memory parallel programming in C/C++ and Fortran [13]. OpenMP uses the fork-join model of parallel execution. The program begins as a single process called the master thread. The master thread executes computation tasks sequentially until a parallel region is encountered. The master thread then creates multiple slave threads, and all the threads execute the computation tasks allocated to them in parallel. After all threads have completed the tasks in the parallel region, they synchronize and terminate, leaving only the master thread, and it continues to execute the subsequent tasks sequentially. Creating and terminating the threads require a certain amount of CPU usage, which becomes an overhead that leads to slower execution of the program.

We assume that the number of available threads in a simulation environment is m and that n subcircuits are created as a result of decoupling the circuit to be simulated by all the transmission lines in the circuit. If n is smaller than m, the computation tasks of the solution process for each subcircuit can be allocated to the threads one by one. However, if n is greater than m, an appropriate task-scheduling algorithm is required for efficient load balancing.

Task-scheduling algorithms can be divided into two classes: static scheduling algorithms and dynamic scheduling algorithms. In static scheduling algorithms, all the computation tasks are allocated to threads before the processes of the parallel region begin. In dynamic scheduling algorithms, the computation tasks are allocated to threads dynamically as the threads request them in the parallel region. Generally, a static scheduling algorithm is preferable in cases where the load of computation tasks is almost the same. By contrast, if the load of computation tasks is variable and/or cannot be estimated, a dynamic scheduling algorithm is the better choice [18]. It appears that the load of the computation tasks required by the solution process of a subcircuit can be roughly predicted from the order or the number of nonzero elements of the coefficient matrix given by the circuit equations. However, it is difficult to estimate it accurately because a sparse matrix method requires not only essential operations with LU decomposition and forward/backward substitution but also computation to find the nonzero elements. Thus, static scheduling algorithms using the approximate load of computation tasks and dynamic scheduling algorithms are compared.

A. Static scheduling algorithms

For static scheduling algorithms, it is important to estimate the load of a computation task accurately and allocate tasks to

threads optimally. If the coefficient matrix given by the circuit equations is dense, the number of operations in the solution process is $O(N^3)$, where N is the order of the matrix. In the context of power system analysis, the matrix is sparse. Thus, a sparse matrix method is usually used and the number of operations in the solution process is much less than $O(N^3)$. Fig. 2 shows the relationship between the number of operations (addition and multiplication) and N, and the number of nonzero elements NZ using the matrices for 44 test cases of various EMT simulations. The figure indicates that the number of operations is almost proportional to $N^{1.25}$ and $NZ^{1.5}$. Here, the numbers calculated by these two formulas are assumed as the load of computation tasks in the solution process of the circuit.

A task-scheduling problem is known to be NP-complete, and a method that can find a suboptimal solution quickly is often utilized [19]. To obtain an appropriate task-scheduling solution, the following heuristic algorithm, called the best-fitting decreasing (BFD) method [19], is used in this study.

i) Sort the computation tasks in decreasing order using the load of each computation task.
ii) Allocate one task to every thread.
iii) Allocate a task to the thread with smallest load of computation task.
iv) Repeat iii) until all the tasks are allocated.

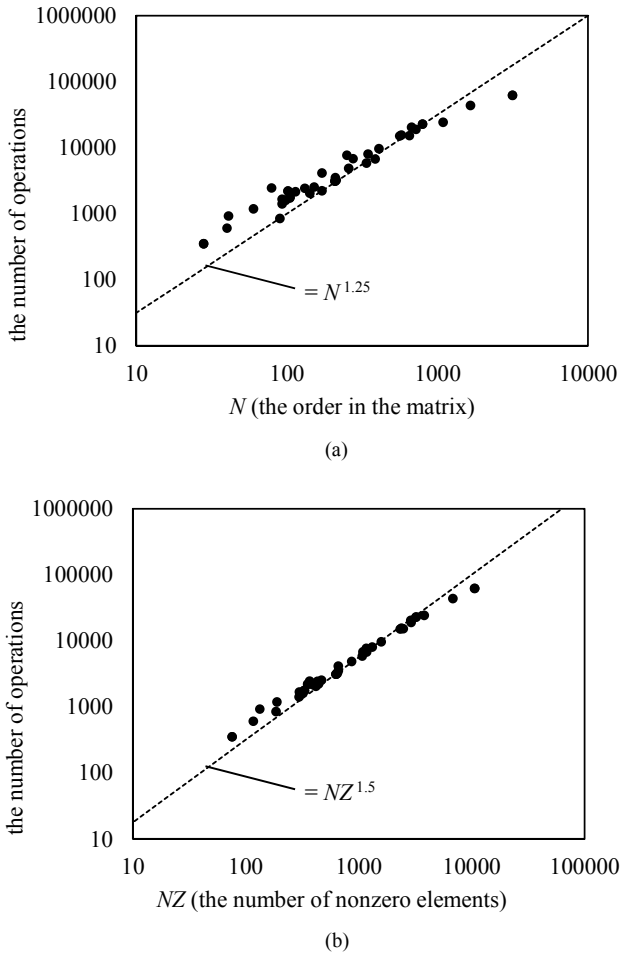

Fig. 2. The relationship between (a) the number of operations and N, and (b) the number of nonzero element NZ using the matrices for 44 test cases.

B. Dynamic scheduling algorithms

A dynamic scheduling algorithm does not necessarily estimate the load of a computation task. In the implementation of a dynamic scheduling algorithm, exclusive access control must be installed to prevent threads from accessing a shared resource for dynamic allocation to threads. List 1 shows the dynamic scheduling algorithm implemented using a critical construct [13] that realizes exclusive access control in this study. In list 1, lines 4-16 are executed in parallel except for lines 10-13, which are executed by only one thread at a time by the critical construct and the tasks are allocated to the threads in these lines.

In OpenMP, it is also possible to realize dynamic scheduling easily by attaching a "dynamic" clause to the work-sharing construct to control how iterations are assigned to threads. List 2 shows examples of codes using the dynamic clause. In this study, the above two dynamic scheduling implementations are validated.

IV. Validation

In this section, the effect of parallelization of the solution process is verified using the task-scheduling algorithms discussed in the previous section. The parallelization is implemented in the EMT analysis program XTAP. The computer used for verification has one Intel Core i7 3930K (3.2 Hz, 6 core) CPU and 32 GB memory. The operating system of this computer is Windows 7 Ultimate Edition. Intel

List 1 The dynamic scheduling algorithm that implemented using the critical construct.

```
 1 ItrNum = 0;
 2 #pragma omp parallel shared(ItrNum)
 3 {
 4   bool eflag = false;
 5   int i;
 6   while ( !eflag )
 7   {
 8     #pragma omp critical
 9     {
10       i = ItrNum;
11       ItrNum++;
12       if (ItrNum > TaskNum)
13         eflag = true;
14     }
15     if (!eflag)
16       Solve(i);  // Solve #i subcircuit.
17   }
18 }
```

List 2 The dynamic scheduling algorithm that implemented using the dynamic clause of the work-sharing construct.

```
1 #pragma omp parallel for schedule(dynamic, 1)
2 for (int i = 0; i < TaskNum; i++)
3   Solve(i);  // Solve #i subcircuit.
```

Hyper-Threading technology [20] and Intel Turbo Boost technology [21], which are implemented in Intel CPU and enhance CPU performance and efficiency, are disabled in this verification.

The parallel EMT analysis program described above is tested on the EMT model of IEEJ's West-10 Benchmark System. This system, which is shown in Fig. 3, approximates the 60 Hz power system in Japan with ten generators. Reference [16] should be consulted for details. The development of the EMT model of the system was reported in [17]. The model represents all three phases. In this model, the transmission lines are represented by the constant-parameter line model [2], the generators are represented by the *d*- and *q*-axes equivalent circuits interfaced with terminals in the *abc* frame using dependent sources, the transformer models are represented by the basic transformer model (delta-star connection), and the load model consists of three-phase star-connected series RL components.

XTAP uses the sparse tableau approach [22] for the formulation process and Fig. 4 shows the coefficient matrix given by the circuit equations obtained by the EMT model of the West-10 Benchmark System using the sparse tableau approach. As a result of decoupling the circuit into several independent subcircuits using all the transmission lines, the number of subcircuits is 17. Table I shows the order *N* and the number of nonzero elements *NZ* in the coefficient matrix of each subcircuit.

Varying the number of available threads from 1 to 6, the EMT model is calculated by the parallel EMT analysis program with the four types of task-scheduling algorithm described above. Fig. 5 shows the ratio between the computation times required for the conventional sequential program and the parallel program in each scheduling algorithm. This graph shows the following:

i) The two static scheduling algorithms are faster than the two dynamic algorithms. It is considered that the overhead required for task allocation is negligible compared with the computation time required for the solution process, and the overhead of the dynamic allocation is larger than that of the static allocation. There are no significant differences between the two methods in the static scheduling algorithms, and between the two methods in the dynamic scheduling algorithms.

ii) The parallel program with only one thread is 1.25 times faster than the conventional sequential program. This is because the load of computation tasks of the solution process is not proportional to the order *N* of the coefficient matrix, and it is considered that LU decomposition and backward/forward substitution on one large matrix require more computational load than computing them separately for the partial matrices.

iii) In this study, the parallel EMT analysis program with six threads is up to 3.5 times faster than the conventional sequential program.

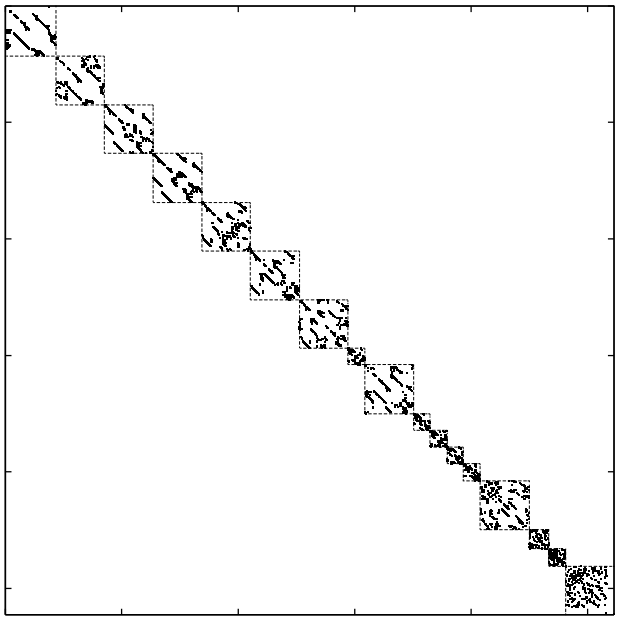

Fig. 4. The coefficient matrix given by the circuit equations obtained by the West-10 Benchmark System using the sparse tableau approach.

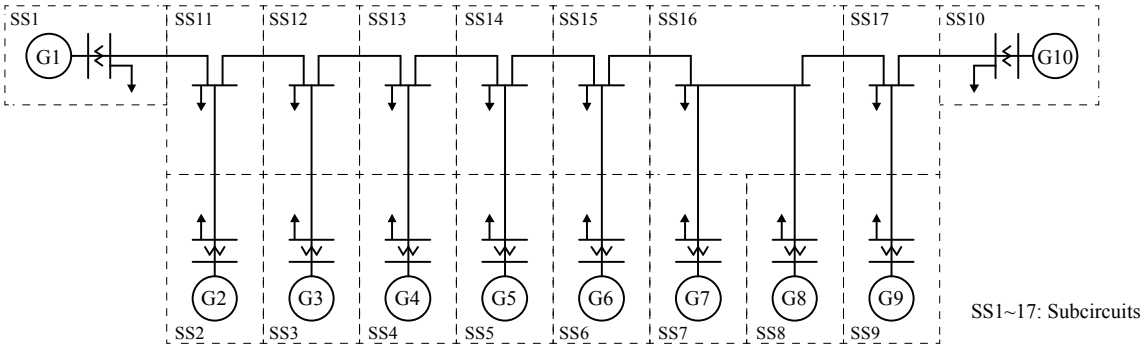

Fig. 3. One-line diagram of the West-10 Benchmark System (IEEJ's West-10 benchmark power system model).

TABLE I
THE ORDER *N* AND THE NUMBER OF NONZERO ELEMENTS *NZ* OF THE COEFFICIENT MATRIX OF EACH SUBCIRCUIT.

No. of Subcircuit	the order *N*	the number of nonzero elements *NZ*
SS1	215	541
SS2	209	526
SS3	209	526
SS4	209	526
SS5	209	526
SS6	209	526
SS7	209	526
SS8	209	526
SS9	209	526
SS10	209	526
SS11	72	257
SS12	72	257
SS13	72	257
SS14	72	257
SS15	72	257
SS16	84	317
SS17	72	257

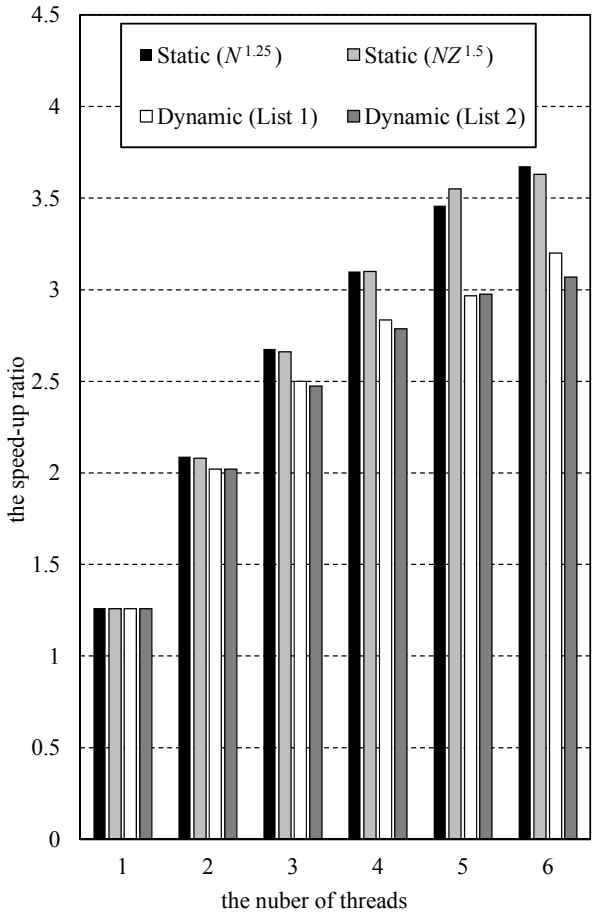

Fig. 5. The ratio between the computing time required in the conventional sequential program and the parallel program in each scheduling algorithm.

V. CONCLUSIONS

Owing to the fast simulation of an EMT analysis program executed on a general-purpose PC with a multi-core processor, the circuit to be simulated was partitioned into subcircuits by distributed parameter transmission lines, and the solution processes of the circuit equations given by the subcircuits were solved in parallel using OpenMP, which is one of the shared-memory parallel programing interfaces. We discussed the effect of task-scheduling algorithms on the computation time through a verification using static and dynamic scheduling algorithms. The parallelization was implemented in the EMT analysis program XTAP, and the performance of the parallelization was validated using the EMT model of the West-10 Benchmark System. As a result of the validation, it was demonstrated that the computation time using the static scheduling algorithm was less than that using the dynamic scheduling algorithm and that the parallel EMT analysis program with six threads was up to 3.5 times faster than the conventional sequential program.

In this paper, we have only validated the performance of the parallelization using the EMT model of the West-10 Benchmark System, which is simple and not a realistic power system. Thus, we will validate the performance using practical power system in the future.

VI. ACKNOWLEDGEMENT

The authors would like to express their sincere thanks to the engineers from Hokkaido Electric Power Co., Inc., Tohoku Electric Power Co., Inc., Chubu Electric Power Co., Inc., Hokuriku Electric Power Company, The Kansai Electric Power Company, Inc., The Chugoku Electric Power Co., Inc., Shikoku Electric Power Co., Inc., Kyushu Electric Power Co., Inc., Electric Power Development Co., Ltd., and The Okinawa Electric Power Co., Inc., for valuable discussions.

VII. REFERENCES

[1] W. F. Tinney, J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," in *Proc. IEEE*, vol. 55, pp. 1801-1809, 1967.

[2] H. W. Dommel, "Digital computer solution of electromagnetic transients in single-and multiphase networks," *IEEE Trans. Power Apparatus and Systems*, vol. PAS-88, issue 4, April 1969.

[3] R. J. Thomas, J. S. Thorp, S. Linke, C. Pottle, C. R. Strohman, "The Cornell university kettering energy systems laboratory," *IEEE Trans. Power Apparatus and Systems*, vol. PAS-104, no. 9, Sep. 1985.

[4] D. M. Falcao, E. Kaszkurewicz, H. L. S. Almeida, "Application of parallel processing techniques to the simulation of power system electromagnetic transients," *IEEE Trans. Power Systems*, vol. 8, no. 1, Feb. 1993.

[5] J. R. Marti, L. R. Linares, "Real-time EMTP-based transients simulation," *IEEE Trans. Power Systems*, vol. 9, no. 3, Aug. 1994.

[6] R. Kuffel, J. Giesbrecht, T. Maguire, R. P. Wierckx, P. Mclaren, "RTDS – A fully digital power system simulator operating in real time," in *Proc. IEEE WESCANEX 95*, pp. 300-305, May 1995.

[7] J. A. Hollman, J. R. Marti, “Real time network simulation with PC-cluster,” *IEEE trans. Power Systems*, vol. 18, no. 2, May 2003.
[8] C Larose, S Guerette, F Guay, A Nolet, T Yamamoto, H Enomoto, Y Kono, Y Hasegawa, H Taoka, “A fully digital real-time power system simulator based on PC-cluster,” *Elsevier Mathematics and Computers in Simulation*, vol. 63, issue 3-5, pp. 151–159, Nov. 2003.
[9] R. Singh, A. M. Gole, P. Graham, J. C. Muller, R. Jayasinghe, B. Jayasekera, D. Muthumuni, “Using local grid and multi-core computing in electromagnetic transients simulation,” in *Proc. IPST 2013*, Vancouver, Canada, July 2013.
[10] Z. Zhou, V. Dinavahi, “Parallel massive-thread electromagnetic transient simulation on GPU,” *IEEE Trans. Power Delivery*, vol. 29, no. 3, June 2014.
[11] H. M. Barros, A. Castro, R. N. Fontoura Filho, M. Groetaers dos Santos, C. F. Teodosio Soares, “Efficient application of parallel processing with standard tools for electromagnetic transients simulation,” in *Proc. IPST 2013*, Vancouver, Canada, July 2013.
[12] S. Montplaisir-Goncalves, J. Mahseredjian, O. Saad, X. Legrand, A. El-Akoum, “A semaphore-based parallelization of networks for electromagnetic transients,” in *Proc. IPST 2015*, Cavtat, Croatia, June 2015.
[13] OpenMP. *OpenMP Application Program Interface* [Online]. Available: http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf
[14] T. Noda, K. Takenaka and T. Inoue, “Numerical integration by the 2-stage diagonally implicit Runge-Kutta method for electromagnetic transient simulations,” *IEEE Trans. Power Delivery*, vol. 24, no. 1, pp. 390-399, Jan. 2009.
[15] T. Noda, T. Kikuma, “A robust and efficient iterative scheme for the EMT simulations of nonlinear circuits,” *IEEE Trans. Power Delivery*, vol. 26, no. 2, pp. 1030-1038, April 2011.
[16] Committee for Standardization of Benchmark Power System Models, “Standard benchmark power system models,” *IEEJ Technical Report*, no. 754, 1999 (title translated into English by the authors).
[17] T. Noda, H. Takizawa, T. Nakajima, “A study of electromagnetic transient simulations using IEEJ’s West-10 benchmark power system model,” *Elsevier Electric Power Systems Research*, no. 138, pp. 195-201, 2016.
[18] C. Breshears, *The Art of Concurrency*, O’Reilly Media, Inc., Sebastopol, CA, USA 2009.
[19] R. Blikberg and T. Sorevik, “Load balancing and OpenMP implementation of nested parallelism,” *Elsevier Parallel Computing*, no. 31, pp. 984-998, 2005.
[20] Intel Hyper-Threading Technology. Intel web site [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html
[21] Intel Turbo Boost Technology. Intel web site [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html
[22] G. D. Hachtel, R. K. Brayton and F. G. Gustavson, “The sparse tableau approach to network analysis and design,” *IEEE Trans. Circuit Theory*, vol. CT-18, no. 1, pp. 101-113, Jan. 1971.