Parallelization of MMC Detailed Equivalent Model

A. Stepanov, J. Mahseredjian, H. Saad, U. Karaagac

***Abstract*—This paper proposes a method to parallelize the computations of the detailed equivalent model of modular multilevel converter (MMC) on multicore CPUs in offline simulations of electromagnetic transients (EMTs). Each arm of the converter is implemented as a DLL independently from the main solver and is interfaced with it using standard procedures. It is also proposed to parallelize the capacitor balancing algorithm using a similar approach. Depending on the simulated system, the proposed method allows to accelerate simulations by five times without affecting accuracy of the results. Results also demonstrate that parallelization of the capacitor balancing algorithm plays an important role in improving simulation speed and can have a larger impact than the parallelization of electrical circuit equations.**

***Keywords*: detailed equivalent model, parallelization, modular multilevel converter, simulations.**

I. Introduction

Modular multilevel converter (MMC) is a power electronic converter that is used in many modern HVDC transmission projects, Fig. 1. It has several significant advantages, including easy scalability to high voltage levels, smooth AC voltage waveform, and relatively low losses, all due to its modular structure and lower switching frequency. The MMC generates AC voltages by inserting the appropriate number of submodules (SMs), which are essentially capacitors with quasi-constant voltage, each of which represents one level of the resulting voltage waveform [1].

It is essential to perform electromagnetic transient (EMT) simulations to ensure safe and reliable operation of HVDC systems. To do so, accurate time-domain models of various equipment are required. Owing to the structural complexity of MMCs, numerous EMT models have been developed, some of the most used ones are [2-4]:

- The detailed model (DM), that represents IGBTs in each SM using a piecewise linear v-i characteristic.
- The detailed equivalent model (DEM), that represents IGBTs as two-value resistances.
- The arm equivalent model, that aggregates all SMs in each arm into a single equivalent circuit.
- The average value model, that represents all SMs in whole converter as a single capacitor.

While detailed models provide high accuracy, they are typically much slower, which restricts their application. Efforts have been put forward to improve the speed of simulations involving MMCs in various ways [4-6]. However, parallelization of MMC model computations has not been widely researched [7].

Parallelization in EMT simulations has been applied through network decoupling by transmission lines [8, 9], co-simulation methods [10], parallel implementation of LU factorization [11]. CPU or FPGA implementations have been used to accelerate the simulations of power electronic devices. Real-time applications are investigated in [7, 12].

This paper proposes a method to parallelize the computations of the DEM on multicore CPUs in offline simulations. Among the conventional MMC models, the DEM is a good candidate for parallelization because:

- It has relatively high computational burden, which offers potentially significant time-gains.
- The nonlinearities in the DEM are not as complex as in the DM, which allows for a relatively easier implementation.
- The DEM is often implemented independently from the main EMT solver, which results in easier interfacing.
- There are at least six DEM blocks per MMC.
- The DEM requires a capacitor balancing algorithm (CBA) block, which can also be parallelized.

Unlike FPGAs, multicore CPUs are often available in modern day PCs and laptops. This allows to prioritize CPU parallelization over other types due to the larger availability of the appropriate hardware.

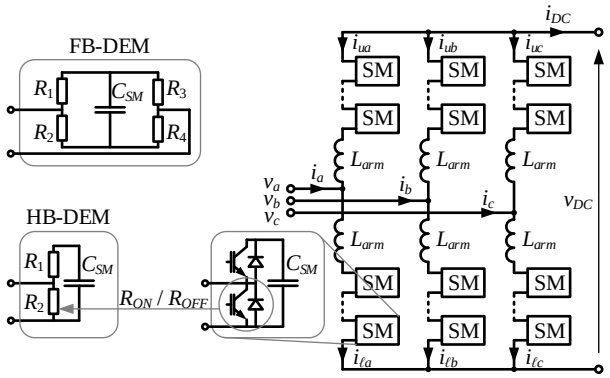

Fig. 1. Structure of the detailed equivalent model of MMC.

II. Procedure Overview

The EMT simulation software considered in this chapter is EMTP [13], which is based on the request-participation interface with the constituting modules. The EMTP computational core (CC) sends requests to each model one by

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada as part of the industrial chair "Multi time-frame simulation of transients for large scale power systems.".

A. Stepanov and J. Mahseredjian are with the Department of Electrical Engineering, Polytechnique Montréal, Montreal, QC, Canada (e-mail of corresponding author: anton.stepanov@polymtl.ca, jeanm@polymtl.ca).

H. Saad is with Réseau de Transport d'Electricité, Paris, France (e-mail: hani.saad@rte-france.com).

U. Karaagac is with with the Department of Electrical Engineering, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: ulas.karaagac@polyu.edu.hk)

Paper submitted to the International Conference on Power Systems Transients (IPST2021) in Belo Horizonte, Brazil June 6-10, 2021.

one and they reply with the results of their internal calculations. The main requests of CC at a single time-point computation with nonlinear elements such as the DEM are as follows:

- Request admittances and history currents from all DEMs.
- Solve the main network equations (MNEs) matrix.
- Repeat two previous steps in case of insufficient precision.
- Request capacitor voltages for control system equations.
- Solve control system equations.
- Go to next time-point.

In the default sequential execution case, the internal model calculations are performed one at a time for each arm when the CC requests them. The idea researched in this paper is that internal computations of all DEM blocks are performed at once in parallel at the beginning of the current time-point and then the solicited DEMs only retrieve the results when requested by CC, as shown in Fig. 2 (EMT-core is CC). The CBA parallelization is performed in a similar manner with the difference being that it only participates in the control system equations part, but not in the MNE.

There are similarities between the proposed approach and the one used in [7]. However, this paper deals with offline simulations and focuses on the simulation time reduction and evaluation of various contributing factors. Besides, the proposed parallelization approach is applicable to computers with any number of cores and to circuits with any number of DEM and CBA blocks and any number of SMs.

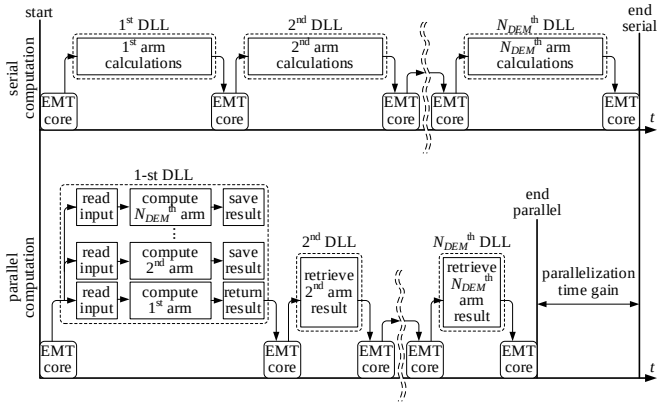

Fig. 2. Overview of the proposed parallelization scheme.

III. DEM PARALLELIZATION

Each DEM arm is interfaced with the CC by using its Norton equivalent circuit. To calculate it, the DEM requires the IGBT control signals and the input voltage [2, 4]. The Norton equivalents values supplied by the DEM DLLs to the CC depend on the control and history values from the previous time-point, so do not require iterations unless in blocked mode.

Internal DEM actions are divided in the following groups:

- Reading input data (IGBT commands, input arm voltage). This is reading from memory so can be parallelized.
- Internal computations (updating history currents, calculating the Norton equivalent, updating capacitor voltage and current values). They only require local variables so can also be parallelized.
- Output data exchange (provide the Norton equivalent and SM voltages to the CC). This has to be performed in series since calls to internal CC functions are used, that cannot be guaranteed to be thread-safe.

The DEM is implemented as a DLL using Fortran-2015 and parallelization is performed using OpenMP provided by the Intel Fortran compiler. Each DEM DLL executes the algorithm shown in Fig. 3. To be able to study the impact of the number of threads on the acceleration, the total number of DEM DLLs N_{DEM} is divided into several groups. The DLLs within the same group are launched in parallel. Each DLL group contains $n_{threads}$ blocks and requires as many threads. Each DLL first checks if it is the first in the group and if so, launches all group's computations in parallel. To avoid multiple computations of the same data, the first DLL raises a special flag ("first DLL in the group") that indicates that the calculations of this group have been performed. The DLLs also check if they are the last one in the group to reset the "first DLL in the group" flag that has been raised by the first DLL in the group so that at the next time-point the first DLL could be correctly assigned.

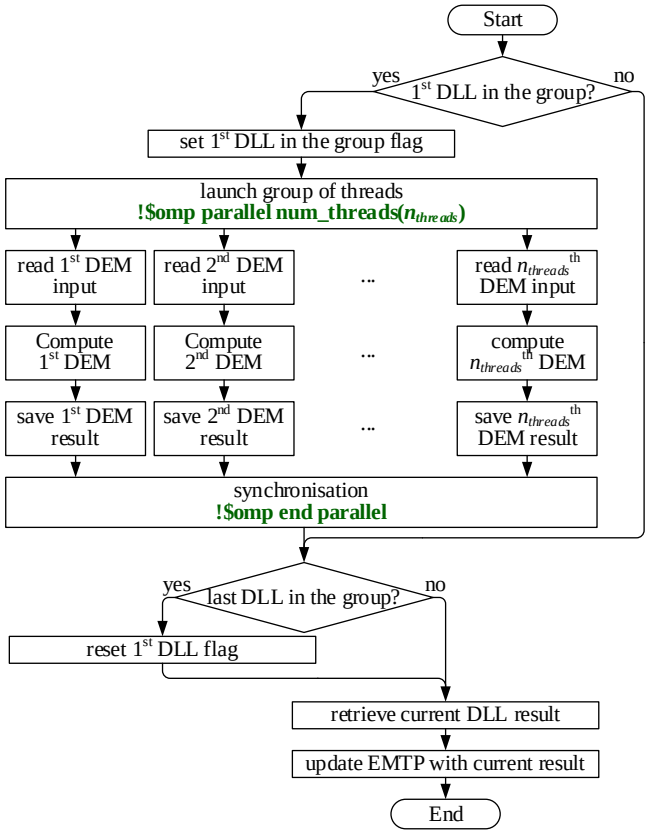

Fig. 3. Implemented DEM parallelization algorithm.

IV. CBA PARALLELIZATION

The CBA control block, i.e. the block that keeps all SM voltages close to each other, deals only with control signals. As with the DEM, all internal functions of the CBA are divided as follows:

- Reading input data (SM voltages, arm current). It can be performed in parallel because it is reading from memory.
- Internal computations (sorting SM voltages, selecting the SMs to insert and to bypass). Only local variables are required so this can also be parallelized.

• Output data exchange (IGBT gating signals). Performed in series since it uses calls to the internal functions of the CC that cannot be guaranteed to be thread-safe.

The CBA is active during normal operation and only provides blocking signals during the blocked mode. The same implementation as in Fig. 3 is used with the only difference being that the DEMs are replaced by the CBAs.

V. Performance Evaluation

The goal of the proposed DEM parallelization scheme is to reduce the time needed to perform simulations of MMC-HVDC systems. Therefore, the time gains must be evaluated. They are evaluated using the acceleration factor, which is found as

$$f_{acc} = t_s / t_p \tag{1}$$

where t_p is the computational time with parallelization and t_s is the computational time with the default single-threaded sequential implementation of the DEM.

In the first approximation, the single-threaded and the multithreaded computing times can be represented by the following formulas

$$t_s = T_0 + T_{DEM} \; N_{DEM} \tag{2}$$

$$t_p = T_0 + T_{DEM} \; N_{DEM} / n_{threads} \tag{3}$$

where T_0 is the time necessary to perform all sequential computations such as the resolution of the MNE, line and machine models, etc.; T_{DEM} is the time to compute one arm.

Given that the number of DEM blocks in each group $n_{threads}$ is between 1 and N_{DEM}, f_{acc} is also limited:

$$\min\left(f_{acc}\right) = 1 \tag{4}$$

$$\max\left(f_{acc}\right) = \frac{T_0 + T_{DEM} \; N_{DEM}}{T_0 + T_{DEM}} \tag{5}$$

If the time required to compute the DEM is large compared to the rest of the computations (i.e. $T_{DEM} \gg T_0$), the maximal acceleration factor will be equal to the total number of DEM blocks in the simulated electrical circuit N_{DEM}.

It should be noted that the threads are created at the beginning of each time-step and suppressed at the end of the time-step and thread management and affinity is performed by the operating system. Since (4) and (5) do not consider the time required to create and manage multiple threads, the actual acceleration factors will be smaller than the theoretical limit.

VI. Simulation Results

The parallelization tests are performed in EMTP on a point-to-point MMC-HVDC link with two MMC stations shown in Fig. 4 (except subsection VI.A). There are 12 DEM blocks, yielding the maximum of 12 threads.

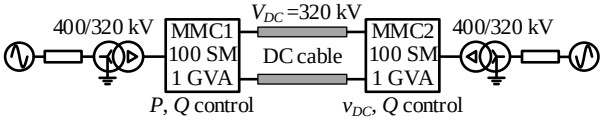

Fig. 4. Simulated MMC-HVDC transmission.

A. Validation

The tests in this subsection validate the implementation, accuracy and demonstrate dependencies between computing times and the number of DEM blocks.

In this and the following sections, the execution times are measured as the average of multiple runs using internal clock of the simulation software. This allows to consider the effects of parallelization on the simulation as a whole.

1) Parallelization validation

A simple electrical circuit shown in Fig. 5 is used for validation (the circuit does not represent a realistic HVDC system so the values of currents and resistances are selected arbitrarily). It contains a DC current source and 12 DEM blocks with the corresponding CBAs, that exchange gating signals ($\boldsymbol{S}_j$) and capacitor voltages ($\boldsymbol{V}_{\boldsymbol{SM}\,j}$) between themselves. Half-bridge SMs are used and all of them are inserted. Simulation time is 1 s and the time-step is 5 µs. The tests are performed on a computer with 4 physical and 8 logical cores. The number of SMs per arm is varied.

Computing times depending on the number of threads and SMs are shown in Table I and Fig. 6. The dependence on the number of SMs is linear for a given number of threads, which validates (2) and (3). Extrapolating computing times to $N_{SM} = 0$ yields the value of T_0. With parallelization, T_0 is relatively higher because some additional time is spent on thread creation and management. No clearly identifiable dependency of T_0 on $n_{threads}$ has been observed.

For further analysis, the computing times in Table I are adjusted for T_0, the value of T_0 is subtracted from the corresponding computing times for a given value of $n_{threads}$. The adjusted computing times are plotted in Fig. 7 as a function of the number of DEM blocks computed in series

$$n_{serial} = N_{DEM} / n_{threads} \tag{6}$$

The results show linear dependency, which confirms the implemented parallelization scheme. However, it can be observed that even after the adjustment the extrapolation does not yield zero computing time for $n_{serial} = 0$. This remaining time is due to the actions performed in series for each arm, as explained in sections III and IV. It is possible to make another observation: the time necessary to compute the DEM is proportional to the number of SMs, which also confirms the correctness of the implementation.

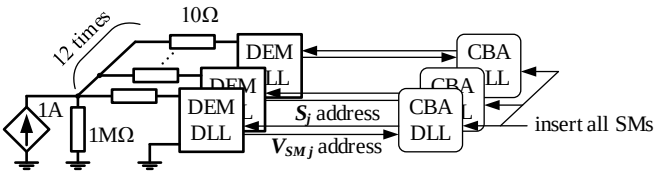

Fig. 5. Circuit to validate the implemented parallelization.

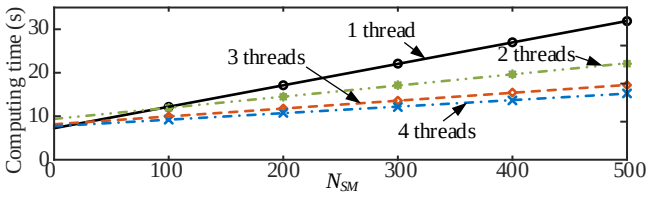

Fig. 6. Computing times with different number of threads.

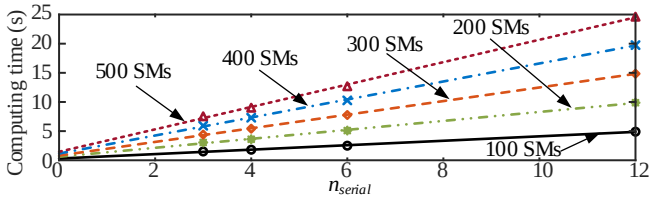

Fig. 7. Adjusted computing times.

TABLE I
COMPUTING TIMES WITH DIFFERENT NUMBER OF THREADS (S)

Number of SMs	1 thread	2 threads	3 threads	4 threads
0 (extrapolation)	7.25	9.39	8.15	7.70
100	12.15	11.89	9.96	9.21
200	17.13	14.49	11.76	10.75
300	22.10	17.15	13.57	12.11
400	27.00	19.66	15.45	13.63
500	31.88	22.08	17.16	15.24

2) Power reference step

In this test, the waveforms obtained with the single-thread simulation are considered as references. The plot lines used for waveforms in figures of this subsection: 1 thread – solid black, 6 threads – dash-dotted blue, 12 threads – dashed orange. The power reference step from 100% to 50% of the nominal power transfer of 1 GW is applied at 0.5 s. No visible difference exists between the waveforms of the DC voltage (Fig. 8) or the total capacitor voltage (Fig. 9).

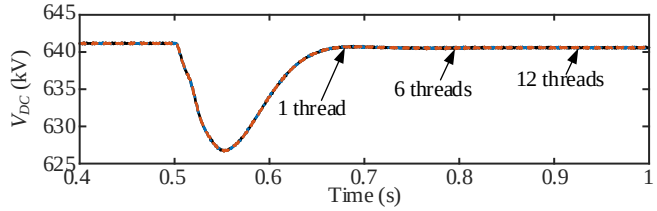

Fig. 8. DC voltage with different number of threads.

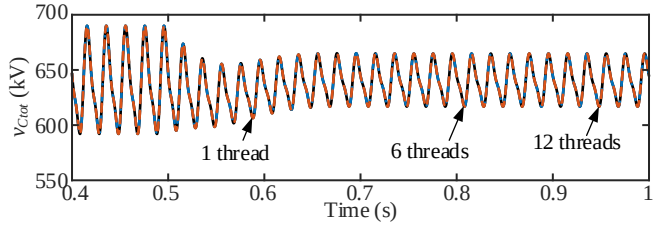

Fig. 9. Total capacitor voltage in phase A upper arm at MMC1.

3) DC fault

The DC fault is applied at 0.5 s in the middle of the DC link. The blocking signal is activated at MMC1 at 0.50044 s, which causes all half-bridge SMs (for example, the 5th SM in Fig. 10) to stop conducting immediately. Full-bridge SMs (for example, the 305th SM in Fig. 11) are able to pass the current in the negative direction for a brief moment. It is clear that the waveforms obtained with parallelization match closely the single-threaded case irrespective of the number of threads.

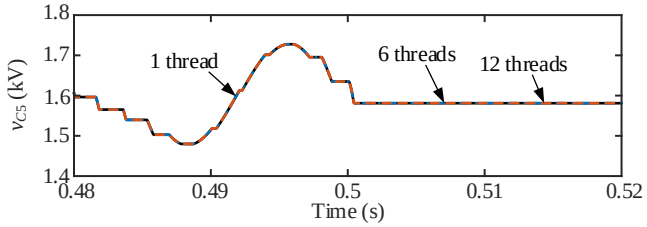

Fig. 10. Voltage of the 5th SM in phase A upper arm at MMC1.

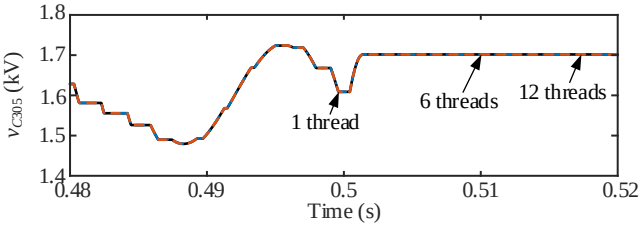

Fig. 11. Voltage of the 305th SM in phase A upper arm at MMC1.

4) Error analysis

Theoretically, parallelization should have no effect on the precision of the results since the computed equations are identical (the single-threaded reference waveforms are obtained using the same implementation of the DLL but each arm is put in its own group, as per Fig. 3, so no groups of threads are launched). However, in reality, small relative difference in the order of 10^{-12} to 10^{-10} is perceivable in all signals. This can be attributed to the potential differences in the compiled binary code and to the different order of calculations, which leads to rounding errors in the least significant digits when representing variables using formats with finite levels of precision (double-precision floating-point format).

B. Time gains

Two computers are used: a Lenovo T560 laptop with two physical and four logical cores and a Lenovo P910 Thinkstation with 24 physical cores and 48 logical cores. The first configuration demonstrates behavior under the restricted number of cores and the second one demonstrates the behavior of the system when the number of cores is higher than the maximum number of threads. In all performed cases, the system is simulated for one second in normal operating mode.

Since different computers are used, acceleration factors will be used for comparisons. This allows to eliminate the differences that appear due to different clock rates, processors architectures, and thread management on different computers.

1) Restricted number of cores

The computing times of the single-thread simulation are taken as the reference. The simulations are performed with two different types of CBA: permutation-based and voltage sorting-based. The former acts on one SM at a time whereas the latter sorts all SMs and selects the most appropriate ones [14]. Acceleration factors depending on the number of SMs are shown in Table II and Table III whereas Fig. 12 and Fig. 13 show the dependence graphically.

The effort required to create multiple threads and manage them in parallel with a low number of cores has significant effect on simulation speed: with a relatively low number of SMs the gains are below 1 and even as the computational burden increases with the number of SMs (until $N_{SM} = 500$), parallelization results in insignificant acceleration.

The acceleration tends to a saturation limit as the number of SMs increases. This limit represents the proportion of the computations performed in parallel to the computations that still have to be performed in series.

It can also be observed that the acceleration factor is not proportional to the number of threads: when the computational

burden of the DEM and the CBA is small compared to the rest of the network (which is the case with the permutation-based CBA), the smaller number of threads results in a better acceleration. As the computational burden of the MMC becomes dominant (with the voltage sorting-based CBA), the parallelization yields better acceleration factors.

When the number of threads is higher than the number of cores, some threads must wait until others finish their execution, which causes additional time losses. Such effects can be observed when the number of threads is higher than the number of physical cores (see the curves for two and four threads in Fig. 12 and Fig. 13, two have better acceleration factors more often than four).

TABLE II
ACCELERATION FACTORS WITH PERMUTATION CBA

Number of SMs	2 threads	4 threads	6 threads	12 threads
100	1.03	0.91	0.52	0.59
200	1.1	1	0.61	0.7
300	1.13	1.06	0.68	0.78
400	1.16	1.07	0.76	0.85
500	1.15	1.11	0.8	0.9

TABLE III
ACCELERATION FACTORS WITH VOLTAGE SORTING CBA

Number of SMs	2 threads	4 threads	6 threads	12 threads
100	1.06	0.98	0.53	0.61
200	1.18	1.17	0.78	0.87
300	1.26	1.3	1.04	1.13
400	1.29	1.37	1.19	1.28
500	1.3	1.4	1.27	1.36

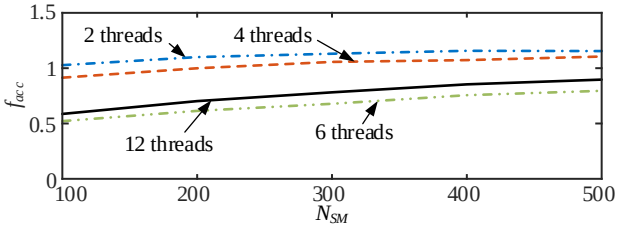

Fig. 12. Acceleration factors with permutation-based CBA.

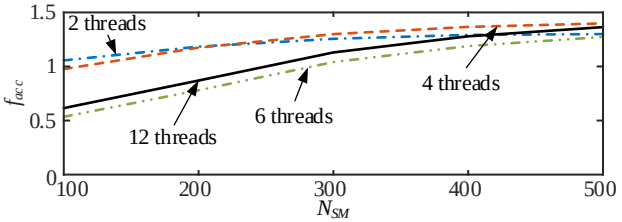

Fig. 13. Acceleration factors with voltage sorting-based CBA.

2) Unrestricted number of cores

Computing times of the single-threaded simulation are taken as reference. Simulations are performed with two different CBAs: permutation-based and voltage sorting-based [14]. Acceleration factors are shown in Table IV, and Fig. 14.

With high number of cores, acceleration factors are higher. As in the case with a relatively small number of cores, better acceleration is achieved when the computational burden of the MMC is high relatively to the rest of the simulated design. This is the case with the voltage sorting-based CBA, where the acceleration factor reaches four.

TABLE IV
ACCELERATION FACTORS WITH HIGH NUMBER OF CORES

Number of SMs	Permutation CBA		Voltage sorting CBA	
	6 threads	12 threads	6 threads	12 threads
100	1.21	1.36	1.32	1.5
200	1.43	1.57	1.83	2.28
300	1.56	1.74	2.38	2.97
400	1.62	1.84	2.64	3.53
500	1.68	1.93	2.9	4.03

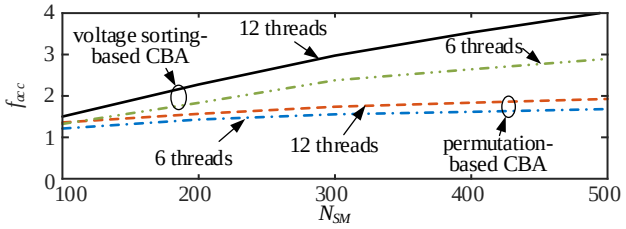

Fig. 14. Acceleration factors with high number of cores.

3) Two MMC-HVDC links

Two MMC-HVDC links are simulated in this test case. The maximum number of threads is 24. This is used to demonstrate the impact of the number of threads on the acceleration factor. Voltage sorting CBA is used and 400 SMs with 320 full bridge SMs. Total simulation time is 1 s and the time-step is 5 µs. Performance of the memory pointer implementation of the DEM is also compared. Memory pointer implementation refers to the information exchange between the DEM and CBA blocks directly in the memory without soliciting the CC (details are given in [5]).

Results are shown in Fig. 15 and Table V (first row indicates the reference time for each case, relatively to which acceleration factors are computed). With the voltage sorting-based CBAs, the system does not exhibit saturation, which means that the computational load of this CBA is high compared to the rest of the simulated design. With the permutation-based CBA the acceleration factor tends to a saturation when the number of parallel threads is above six.

With six threads, a small dip can be seen in all waveforms. This can be attributed to the hardware implementation of multicore processors. In all tests, the memory pointer implementation improves acceleration.

TABLE V
ACCELERATION FACTORS DEPENDING ON THE NUMBER OF THREADS

Thread count	Permutation CBA		Voltage sorting CBA	
	Default	Memory pointer	Default	Memory pointer
1 (base time)	289.4 s	232.2 s	764.2 s	730.8 s
4	1.5	1.7	2.32	2.6
6	1.59	1.84	2.59	2.97
8	1.64	2.08	3.06	3.77
12	1.67	2.14	3.36	4.3
24	1.72	2.33	3.84	5.51

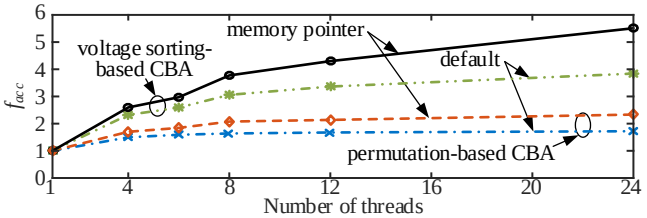

Fig. 15. Acceleration factors with two HVDC links.

4) DEM-only parallelization

In some cases, individual blocks of the control system might not be available for parallelization. This is the case when the whole control system is provided by a manufacturer in a form of a black box due to the confidentiality. In such cases, only the DEM can be parallelized.

The same two MMC-HVDC links as in subsection VI.B.3 are used to demonstrate the effects of DEM-only parallelization. The results are shown in Table VI and Fig. 16.

When the CBA requires a considerable amount of computations, which is the case of the voltage sorting-based CBA, the DEM parallelization has a negligible effect, the acceleration factors are close to one. With the permutation-based CBA, the acceleration factors are closer to 1.3. The saturation limit of f_{acc} is reached with four threads.

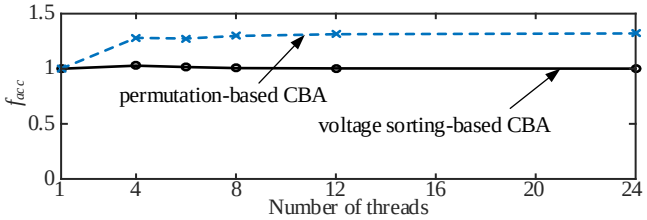

Fig. 16. Acceleration factors with DEM-only parallelization.

TABLE VI
ACCELERATION FACTORS IN CASE OF DEM-ONLY PARALLELIZATION

Thread count	2 threads	4 threads
1 (base time)	289.4 s	756.5 s
4	1.28	1.03
6	1.27	1.02
8	1.3	1.01
12	1.31	1
24	1.32	1

VII. CONCLUSION

The proposed arm-level parallelization method for the detailed equivalent model of MMC, which also includes the CBA parallelization, allows to significantly accelerate offline EMT simulations. The acceleration factor increases as the number of SMs increases, owing to the increase in the computational burden of the parallelizable computations compared to the rest of the simulated design. If this method is used in a large network that is solved in parallel using decoupling lines, significant acceleration gains are expected.

When a large number of cores is available, the acceleration factor increases with the number of threads until it reaches a saturation limit. This limit indicates that the time spent to perform the DEM and CBA computations has become negligible compared to the rest of the design. However, when the number of available cores is limited, parallelization can increase the computing time due to additional efforts required to create and manage multiple threads.

The CBA has an important role in providing high acceleration factors, due to significant computational burden. The highest acceleration factors are achievable when the CBA has a lot of internal calculations and they are parallelized.

DEM-only parallelization can result in some acceleration if its computational effort is comparable to the rest of the simulated circuit. With the permutation-based CBA, the simulations can be accelerated by about 30%.

VIII. ACKNOWLEDGMENT

A. Stepanov would like to acknowledge the support of Vanier Canada Graduate Scholarship in his research.

IX. REFERENCES

[1] A. Lesnicar and R. Marquardt, "An innovative modular multilevel converter topology suitable for a wide power range," in *2003 IEEE Bologna PowerTech - Conference Proceedings*, Bologna, Italy, 2003, vol. 3: IEEE Computer Society, pp. 272-277, doi: 10.1109/ptc.2003.1304403.

[2] H. Saad, S. Dennetiere, J. Mahseredjian, P. Delarue, X. Guillaud *et al.*, "Modular Multilevel Converter Models for Electromagnetic Transients," (in English), *IEEE Trans. on Power Delivery*, vol. 29, no. 3, pp. 1481-1489, June 2014, doi: 10.1109/tpwrd.2013.2285633.

[3] H. Saad, J. Peralta, S. Dennetiere, J. Mahseredjian, J. Jatskevich *et al.*, "Dynamic Averaged and Simplified Models for MMC-Based HVDC Transmission Systems," (in English), *IEEE Trans. on Power Delivery*, vol. 28, no. 3, pp. 1723-1730, July 2013, doi: 10.1109/tpwrd.2013.2251912.

[4] U. N. Gnanarathna, A. M. Gole, and R. P. Jayasinghe, "Efficient Modeling of Modular Multilevel HVDC Converters (MMC) on Electromagnetic Transient Simulation Programs," (in English), *IEEE Transactions on Power Delivery*, vol. 26, no. 1, pp. 316-324, Jan 2011, doi: 10.1109/tpwrd.2010.2060737.

[5] A. Stepanov, J. Mahseredjian, U. Karaagac, and H. Saad, "Adaptive Modular Multilevel Converter Model for Electromagnetic Transient Simulations," *IEEE Transactions on Power Delivery*, pp. 1-1, 2020, doi: 10.1109/TPWRD.2020.2993502.

[6] S. Yu, S. Zhang, Y. Wei, Y. Zhu, and Y. Sun, "Efficient and accurate hybrid model of modular multilevel converters for large MTDC systems," *IET Generation, Transmission & Distribution*, vol. 12, no. 7, pp. 1565-1572, 2017.

[7] H. Saad, T. Ould-Bachir, J. Mahseredjian, C. Dufour, S. Dennetiere, and S. Nguefeu, "Real-Time Simulation of MMCs Using CPU and FPGA," *IEEE Transactions on Power Electronics*, vol. 30, no. 1, pp. 259-67, 2015, doi: 10.1109/tpel.2013.2282600.

[8] D. M. Falcao, E. Kaszkurewicz, and H. L. S. Almeida, "Application of parallel processing techniques to the simulation of power system electromagnetic transients," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 90-96, 1993, doi: 10.1109/59.221253.

[9] A. Abusalah, O. Saad, J. Mahseredjian, U. Karaagac, and I. Kocar, "Accelerated Sparse Matrix-Based Computation of Electromagnetic Transients," *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 13-21, 2020, doi: 10.1109/OAJPE.2019.2952776.

[10] M. Cai, J. Mahseredjian, U. Karaagac, A. El-Akoum, and X. Fu, "Functional Mock-Up Interface Based Parallel Multistep Approach With Signal Correction for Electromagnetic Transients Simulations," *IEEE Transactions on Power Systems*, vol. 34, no. 3, pp. 2482-2484, 2019, doi: 10.1109/TPWRS.2019.2902740.

[11] F. Cong and Y. Tao, "Sparse LU Factorization with Partial Pivoting on Distributed Memory Machines," in *Supercomputing '96:Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*, 1-1 Jan. 1996 1996, pp. 31-31, doi: 10.1109/SUPERC.1996.183533.

[12] M. Matar and R. Iravani, "FPGA Implementation of the Power Electronic Converter Model for Real-Time Simulation of Electromagnetic Transients," *IEEE Transactions on Power Delivery*, vol. 25, no. 2, pp. 852-860, 2010, doi: 10.1109/TPWRD.2009.2033603.

[13] J. Mahseredjian, S. Dennetière, L. Dubé, B. Khodabakhchian, and L. Gérin-Lajoie, "On a new approach for the simulation of transients in power systems," (in English), *Electric Power Systems Research*, vol. 77, no. 11, pp. 1514-1520, Sep. 2007, doi: 10.1016/j.epsr.2006.08.027.

[14] A. Stepanov, H. Saad, U. Karaagac, and J. Mahseredjian, "Initialization of Modular Multilevel Converter Models for the Simulation of Electromagnetic Transients," *IEEE Transactions on Power Delivery*, vol. 34, no. 1, pp. 290-300, 2018.