A Semaphore-based Parallelization of Networks for Electromagnetic Transients

S. Montplaisir-Gonçalves, J. Mahseredjian, O. Saad, X. Legrand, A. El-Akoum

Abstract-- **This paper presents a parallel processing tool used to link multiple instances of the electromagnetic transients program EMTP (RV version) through one or more transmission lines or cables. The intrinsic propagation delay of transmission lines allows to naturally decouple a large network into several separate subnetworks. The subnetworks can then be solved in parallel. A co-simulation process is established, and each instance of EMTP running on a separate core solves its own subnetwork and shares data through linking transmission lines at every time-step. The resulting parallelization reduces computing times for the total simulation process. Initialization issues are also addressed and solved for parallel computations. The presented approach is compliant with the Functional Mock-up Interface (FMI) co-simulation standard and the co-simulation process (in parallel) is programmed with low-level synchronization primitives known as semaphores.**

Keywords: **Electromagnetic transients, EMTP, Co-Simulation, Parallelization, Multi-core architecture, FMI.**

I. INTRODUCTION

The complexity of modern power systems is continuously increasing. The integration of renewable energies, HVDC systems and FACTS devices requires extensive analysis methods for system design, protection, optimization and detection of abnormal/unstable operating conditions. The methodology used in the simulation of electromagnetic transients (EMTs) is widely accepted in the industry. EMT-type simulation tools, such as EMTP, are extensively used to achieve highly accurate computations with circuit based power system models. The main disadvantage with current EMT-type off-line simulation tools is computing time for increasingly large and complex networks. The higher computing time costs are incurred in the solution of large linear algebra systems, updating and refactoring matrices due to switches and nonlinear functions and model equations. Although sparse matrix solution methods are currently used in EMTP and model codes are optimized, the implemented sequential solution approach cannot benefit from the parallel architectures of modern computers.

The emergence of multi-core computers is an important paradigm shift in the world of electrical network simulation, and must be addressed for faster and more efficient system studies. This paper contributes towards such research for the computation of EMTs. The presented tool is based on network decoupling through distributed parameter transmission lines (or cables). Such line models have an intrinsic time delay [1] that allows to accurately subdivide a given network into subnetworks and solve the subnetwork equations concurrently. The new tool links the separate subnetworks, establishing a co-simulation process where each instance of EMTP running on a separate core solves its own subnetwork. At every time-step, data is shared through the linking transmission lines. The resulting parallelization allows to reduce computing time for the total simulation process. The presented interfacing of data is compliant with the Functional Mock-up Interface (FMI) co-simulation standard and the parallel co-simulation process is programmed with low-level synchronization primitives known as semaphores.

The presented tool is implemented in EMTP [2]. Recently, other electromagnetic transient programs have developed similar tools for parallel simulations, as detailed in [3] and [4]. These programs use the same network decoupling method, but implement data sharing between processes by different means than what is shown in this paper. Moreover, the parallel simulation tool in EMTP incorporates three additional features. First, precise initialization of the subnetworks is possible using the steady-state solution of the complete network. Second, the tool allows multiple transmission lines between master-slave co-simulation couples. Finally, the tool proposed in this paper is tested with practical electrical networks and contributes realistic benchmarks for the related field of research. Another distinctive aspect is the fact that the proposed method is based on a sparse matrix solver and the iterative Newton method is used for the inclusion of nonlinear models.

Although it is conceptually feasible to apply network tearing at arbitrary locations (connected through transmission lines or cables), the computing times will depend on the sizes and contents of the resulting subnetworks. This aspect is analyzed in this paper using practical system benchmarks. Other aspects, such as communication delays between decoupled networks, are also discussed.

II. DESIGN OF THE SIMULATION TOOL

A. *Parallel co-simulation using FMI and semaphores*

A co-simulation tool between EMTP and MATLAB was initially developed in [5] with the objective to run EMTP simulations with separate control systems (e.g. governor and

S. Montplaisir-Gonçalves and J. Mahseredjian are with École Polytechnique de Montréal, Université de Montréal, Montréal (Québec), Canada, H3T 1J4 (e-mail of corresponding author: Sara.Montplaisir-Goncalves@polymtl.ca).
O. Saad is with Hydro-Québec (IREQ), Varennes (Québec), Canada, J3X 1S1.
X. Legrand and A. El-Akoum are with EDF – Électricité de France R&D, 1 avenue du Général de Gaulle, 92141 Clamart CEDEX, France.

Paper submitted to the International Conference on Power Systems Transients (IPST2015) in Cavtat, Croatia June 15-18, 2015.

voltage regulation of synchronous machines) programmed and computed in MATLAB. The tool presented in this paper follows this work. A parallel co-simulation environment is established using the Functional Mock-up Interface (FMI) standard. In this standard it is possible to interface two or more simulation tools in a co-simulation environment using a master-slave communication method [6]. The FMI standard also allows to automatically extend the presented co-simulation setup to include other simulation packages.

The overall architecture of the tool is presented in Fig. 1 for two concurrent subnetworks. The DLL (Dynamic Link Library) functionality in EMTP [8] is used to program the interfacing transmission line models. The Master and Slave DLLs are programmed to communicate with a predefined protocol of requests from the main EMTP solver. These DLLs are set to communicate through a second layer of FMI functions (see FMI Links in Fig. 1). This standard is supported by the European project *Modelisar* aiming to regulate sharing formats in co-simulation environments. More specifically, it defines the exact composition of the co-simulation bus, and all the objects and variables needed for the sharing functions. The names and definitions of these functions are also detailed. It is those sharing functions that were coded in the *FMI Link Master* and *FMI Link Slave* DLLs shown in Fig. 1. The parallel simulation tool for EMTP was implemented to be compliant with this standard in order to simplify communication with other compliant simulation programs or co-simulation platforms, if the need arises.
The tool offers a variety of co-simulation modes, depending on the user needs. The synchronization schematic used to run multiple EMTP instances in parallel is shown in Fig. 2. It illustrates the communication procedure between the master and slaves.

The co-simulation method is implemented with low-level synchronization primitives known as semaphores [7]. They are widely used in computer programming to solve concurrency problems where data is shared between multiple programs or threads. They are used both as locks and condition variables to protect and manage read/write operations on shared variables. Synchronization with semaphores is achieved with the *Release* and *Wait* pre-built functions that can be used by any process or thread. The EMTP master and slave instances manage one semaphore each, as shown in Fig. 2. The red arrow represents the master DLL releasing its semaphore for the slave. The master finishes its current time-step, and then passively waits for the slave semaphore to be released. When the slave EMTP instance finishes its current time step, it releases its semaphore as represented by the blue arrow. An additional semaphore is used by the master in the same manner for initialization procedures (not shown). During the time interval between acquiring and releasing a semaphore, data can be exchanged through the shared memory buffer (co-simulation bus) as represented with thick gray arrows. An arrow pointing towards the master or slave represents a reading operation, while an arrow pointing towards the bus represents a writing operation.

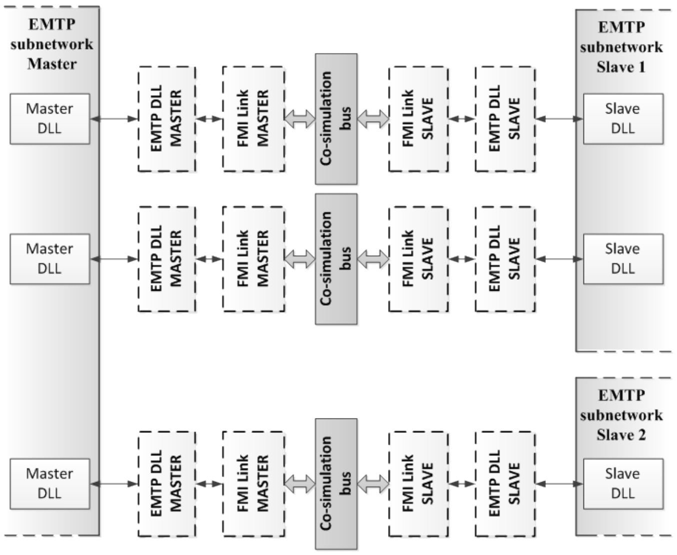


Fig. 1. Interfacing of three subnetworks via DLLs and multiple co-simulation buses.

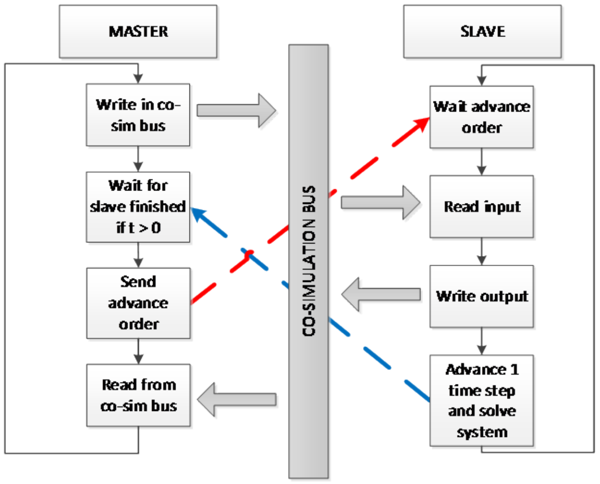


Fig. 2. Communication scheme between master and slave instances of EMTP (single slave example).

B. *Steady-state initialization*

In EMTP it is possible to start the network solution from an unbalanced and multiphase load-flow solution followed by a steady-state solution. The load-flow solution is based on load-flow constraints and requires iterations, whereas the linear steady-state solution uses lumped models for all components. The time-domain solution starts from the steady-state phasors to enter steady-state immediately.

In the above initialization method it is required to solve the complete network in a single EMTP instance since the network equations cannot be decoupled in load-flow and steady-state solutions. Once the steady-state solution is found for the complete network, the voltage phasors at each end of each transmission line are stored and transmitted to the independent subnetwork solutions. In this approach all concurrently simulated subnetworks can calculate their own independent steady-solutions and initialize their time-domain counterparts in parallel. The load-flow solution is found only once and only for the complete network.

III. TEST CASES

The parallel processing tool described above is tested on practical system benchmarks. Three benchmarks are described in the following paragraphs. The computing time gains are based on the CPU time of the same system being simulated on a single core. The computers used for testing had 4 cores on the Windows 7 environment. The numerical integration time-step is $50\mu s$ for all test cases.

It is noted that the simulation results when using multiple cores (parallelization) are identical to those obtained on a single core.

For all test cases, EMTP uses a fully iterative solver [2] (Newton method). Such a solver is essential for solution accuracy. During its iterations, the solver must continuously update its sparse matrix and refactor it. Although the number of iterations is minimized by optimizing convergence speed, this numerical phase creates significant overhead. When the network is solved on separate cores, the iterative process is applied on smaller subnetworks (smaller matrices) and consequently also contributes to improving performance. The subnetworks may require more or less iterations while affecting only their own system of equations.

A. Test case Network-1A

The Network-1A benchmark shown in Fig. 3 is based on the IEEE-39 bus system. It is the EMT version [9] of the original IEEE-39 benchmark. The long transmission lines are modeled using the distributed parameter line model with frequency independent parameters (CP-line). All lines are continuously transposed. Some short lines are modeled using pi-sections. All load and generator transformers are modeled using three independent units with magnetization data. The synchronous machines are using the dq0 modeling approach, with single mass data and include related automatic voltage regulators and governors (AVR/GOV). For all loads the model includes voltage and frequency dependency [10]. The load models are implemented using block diagrams. Such a setup is also suitable for performing electromechanical transient analysis in EMTP.

The total number of circuit nodes is 478 and the main system of network equations is 730×730 with 3082 non-zeros. There is a total of 1333 circuit devices. It is recalled that EMTP uses modified-augmented-nodal analysis [2] with a sparse matrix solver. The representation of control systems (all block diagrams) requires 5443 devices.

The setup is to simulate a 3-phase fault on bus B3 with line tripping.

The parallel simulation of this system is using 3 co-simulation buses with one Master and one Slave. The co-simulation buses are shown in Fig. 3. This setup allows to achieve an equal sharing of computational load between two cores and optimize for communication overhead. The observed gain in simulation is 1.2 times (faster) or 16.7%. The communication overhead is 3.9%. The average number of iterations per time-point in this case is 1.34.

B. Test case Network-1B

The Network-1B (see Fig. 4), is similar to Network-1A, except that now onshore wind generation is added on buses B1, B2 and B25 to replace the synchronous generators on buses B2 and B25. The configuration the onshore wind park Onshore_WP2 connected to buses B2, B25 and to Onshore_WP1 is shown in Fig. 4b.

The Onshore_WP1 subnetwork contains 4 separate wind parks, two of Type-III and two of Type IV, for a total generation of 255 MW. The Onshore-WP2 subnetwork contains 7 separate wind parks for a total generation of 600 MW. Each wind park is aggregated and modeled with its equivalent collector network connected to a 315/34.5 kV transformer. Average value models are used for wind generators.

In this test case, there is a total of 1554 circuit nodes and the main system of network equations uses a sparse matrix of 2588×2588 with 9288 non-zeros. There is a total of 2336 electric devices and 24364 control system blocks.

As shown in Fig. 4, for this network, the parallel simulation setup is using 5 co-simulation buses with one Master and two Slaves. This setup allows to share a quasi-equal effort on wind turbines between two cores, whereas the third core being loaded with synchronous machine and load model equations. The computational gain is 1.8 times (faster) or 44%. The communication overhead is 2.2%. The average number of iterations per time-point now increases to 3, due to the presence of wind generators. In this case the fault is applied on the interconnection point between Onshore_WP1 and Onshore_WP2.

C. Test case Network-2A

The third case Network-2A used in this paper is based on a realistic power system [11] case for studying power system transients. The top view of this case is presented in Fig. 5.

The number of circuit nodes is 469, the main system of network equations has a size of 691 with 3535 non-zeros. There is a total of 596 circuit devices and 2054 control blocks.

All synchronous generators are modeled with exciter and governor controls, include saturation data and single mass representation. The generator subcircuits include all running individual group units for a total of 35 generators. The loads are now represented using constant impedances in time-domain.

The simulation studies a 3-phase fault with line tripping.

As shown in Fig. 5, this test case is using 8 co-simulation buses for one Master and three Slaves. This configuration results into a computational gain of 1.14 times (faster) or 12.6%. The communication overhead is 24.5% and this is mainly due to the large number of co-simulation buses and their share in the overall process. The average number of iterations per time-point is 1.

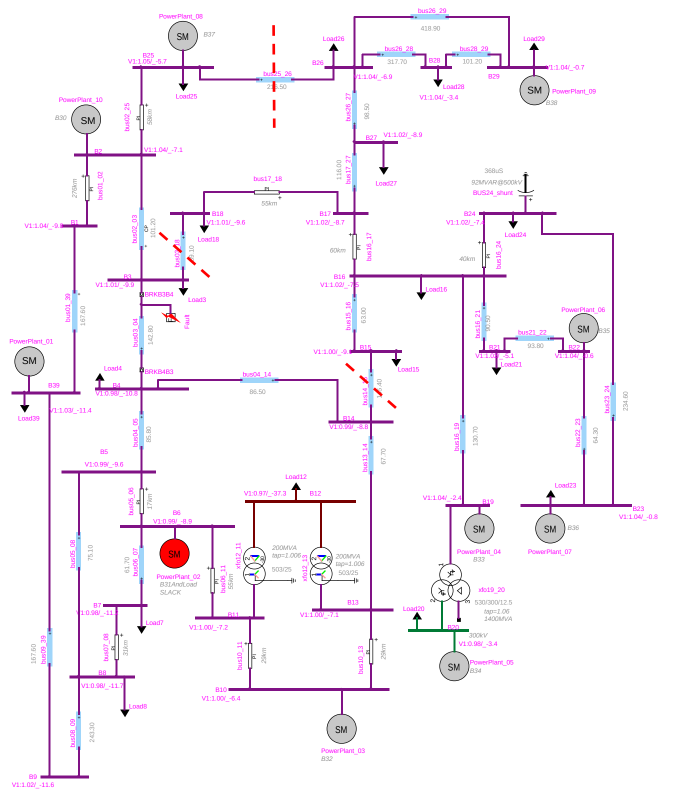


Fig. 3. IEEE 39 bus test system, Network-1A, dotted red lines indicate the co-simulation buses.

D. Test case Network-2B

This test case is similar to Network-2A, except that the number of synchronous generators is reduced to 33 and onshore wind generation is added between the buses ADAPA and OSMAN. The total wind generation is 322.5 MW and modeling details are similar to Network-1B. There are three Type-III and two Type-IV wind farms.

The network details are as follows: 964 circuit nodes, 1542 equations with 6364 non-zeros in the sparse matrix, 1284 circuit devices and 10420 control blocks.

The studied fault is at the same location as Network-2A.

This case was parallelized using one Master and one Slave and 3 co-simulation buses. The main computational bottleneck being the wind farms, the achieved computational gain is only 1.03 times (faster) or 3%. It was not possible to schedule the wind farms onto a separate core due to short line sections (pi-circuit model). The communication overhead is 2.02%. The average number of iterations per time-point is now 3 due to wind generators.

E. Test case Network-2C

This test case is also based on Network-2A, but, in addition to the inclusion of the on-shore wind park of Network-2B, an off-shore wind park of a total capacity of 1057.5 MW is added as shown in Fig. 6.

The off-shore wind parks are connected to the shore through a multiterminal HVDC-MMC system (see Fig. 7). The MMC converters are modeled using average value representation [12][13] and each arm contains 400 submodules.

The network setup details are: 1480 circuit nodes, 2425 equations with 9359 non-zeros in the sparse matrix, 1997 circuit devices and 17931 control blocks.

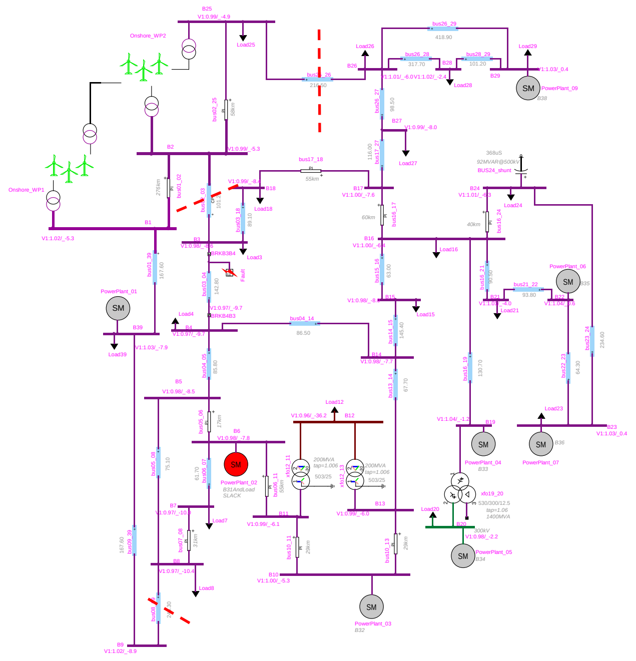
As shown in Fig. 6, this case is using 3 co-simulation buses for one Master and one Slave. By the fact that off-shore and on-shore wind farms are solved on separate CPUs this test case achieve a significant gain of 1.9 times (faster) or 47.5%. The communication overhead is 1.92%. The average number of iterations per time-point remains at 3.

F. Analysis

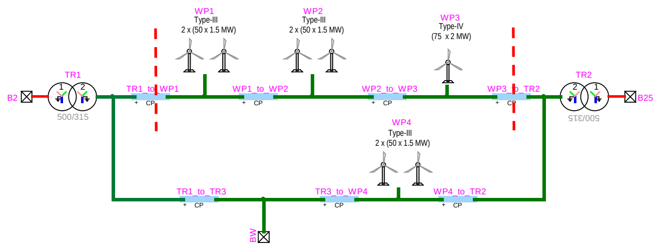
It is apparent from the above test cases that the most significant computational gains in the presented co-simulation (parallelization) approach are obtained when parallelizing computationally demanding models, such as wind farms. The parallelization of such models is optimized due to reduced communication overhead.

It has been observed, as expected, that using multiple subnetworks is not necessarily faster. More parallel instances may result into more co-simulation buses, and therefore increased communication overhead. This overhead has been estimated to vary from 65% for the worst case, to 1.9% for the best case.

As in other parallelization methods, computational load sharing between subnetwork cores remains an essential aspect and further research must allow to develop automatic optimization methods based on test case contents and topology.



a) main network



b) subnetwork Onshore_WP2

Fig. 4. IEEE 39 bus test system with wind generation, Network-1B, dotted red lines indicate the co-simulation buses.

IV. CONCLUSION

This paper presented a parallel processing approach for the simulation of power system transients. The new tool innovates by developing a simulation environment based on the master-slave Functional Mock-up Interface co-simulation standard and by programming the co-simulation process with low-level synchronization primitives known as semaphores. Moreover, the delivered environment for the computation of electromagnetic transients (EMTP) is distinctively based and tested on a sparse matrix solver with a fully iterative process for nonlinear models. Starting parallel simulations from a full load-flow solution is also achieved.

This paper also contributes benchmarks for testing parallel computation methods with EMT-type simulation tools. The most significant computational performance gains are achieved by the parallelization of computationally demanding models, such as wind farms, and by minimizing relative communication delays through co-simulation buses.

Further research should concentrate on the optimal loading of parallel CPU cores by automatic calculation of network tearing locations for maximizing performance.

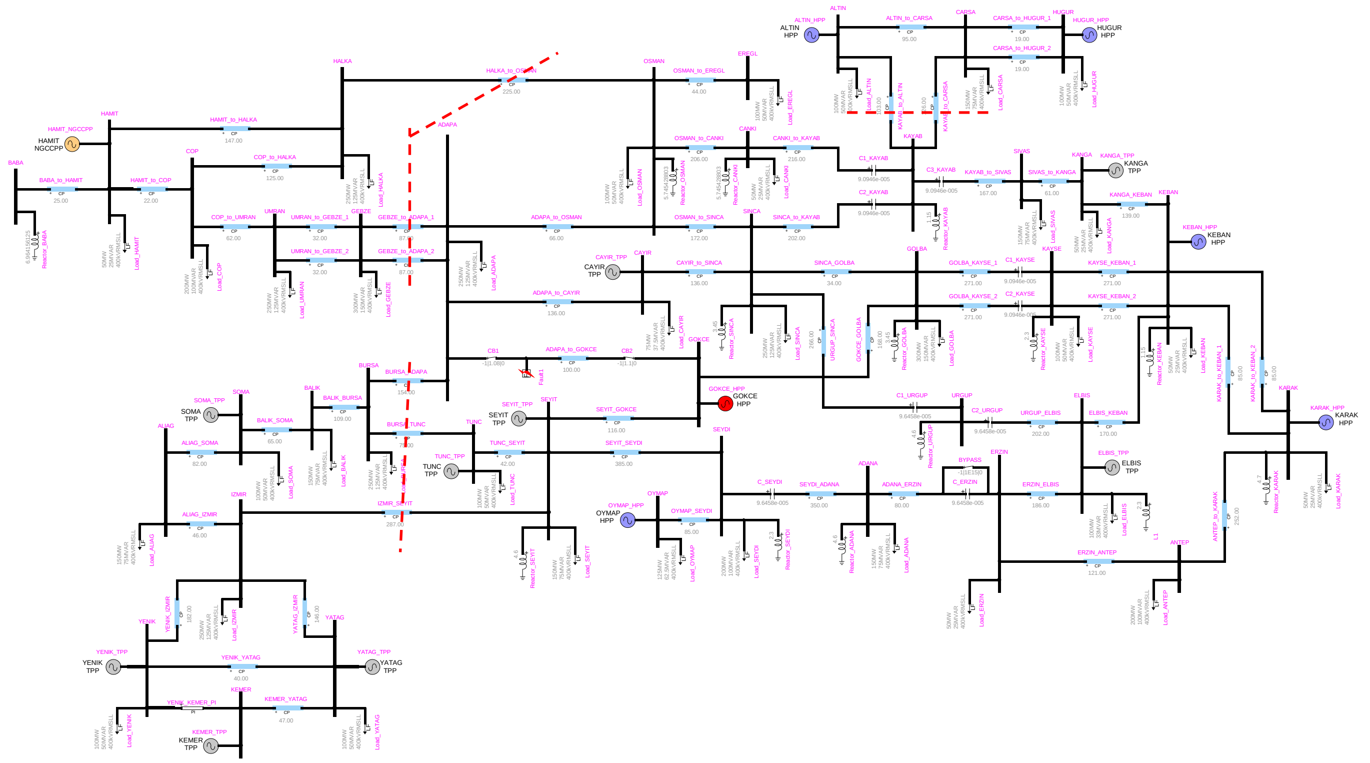


Fig. 5. Top level view of large scale system benchmark, Network-2A, dotted red lines indicate the co-simulation buses.

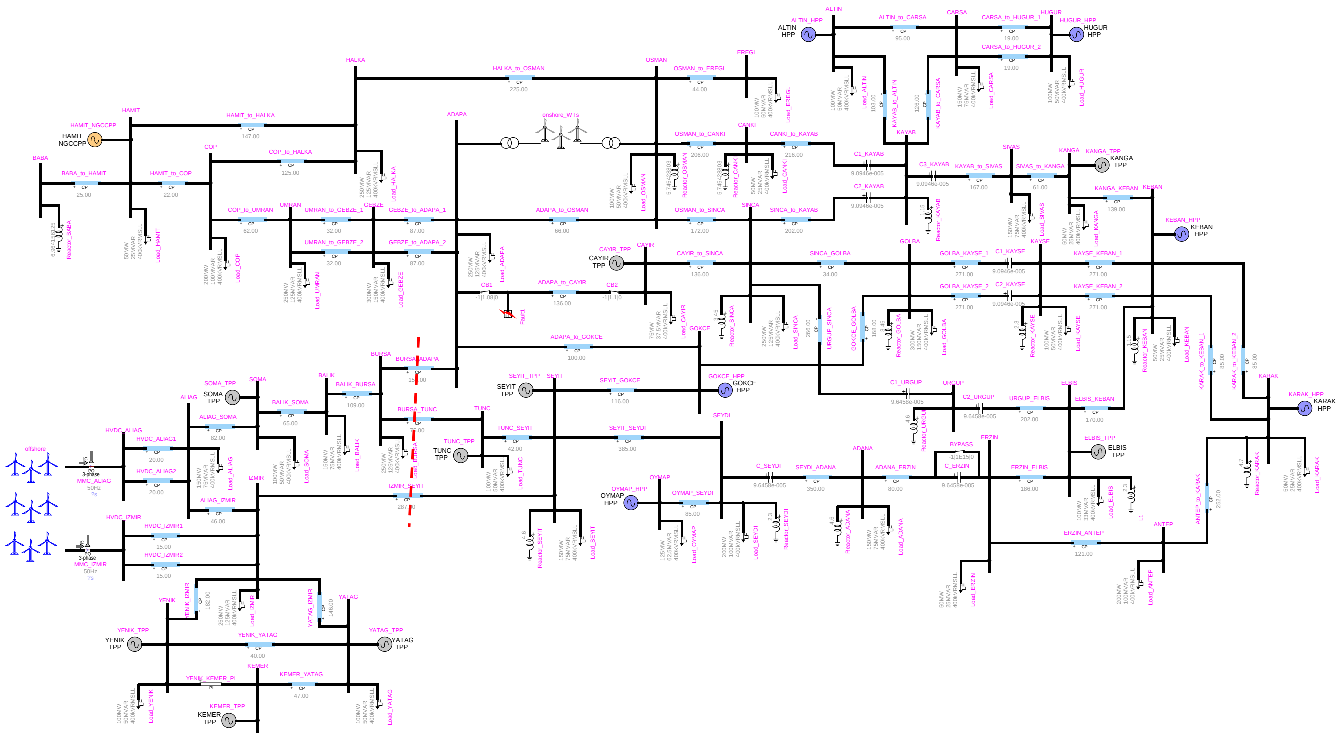


Fig. 6. Top level view of large scale system benchmark, Network-2C, dotted red line indicates the co-simulation bus.

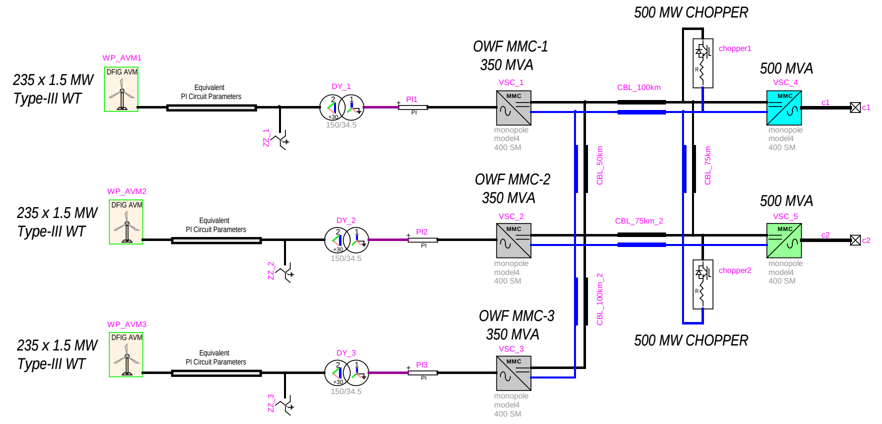


Fig. 7. Off-shore wind parks with HVDC-MMC transmission in Fig. 6.

V. References

[1] D. J. Tylavsky et al., "Parallel processing in power systems computation," *IEEE Transactions on Power Systems*, vol. 7, no. 2, 1992, pp.629-638.

[2] J. Mahseredjian, S. Dennetière, L. Dubé, B. Khodabakhchian and L. Gérin-Lajoie, "On a new approach for the simulation of transients in power systems," *Electric Power Systems Research*, vol. 77, no. 11, 2007, pp.1514-1520.

[3] R. Singh, A. M. Gole, P. Graham, J. C. Muller, R. Jayasinghe, B. Jayasekera and D. Muthumuni, "Using Local Grid and Multi-core Computing in Electromagnetic Transients Simulation," in *International conference on Power System Transients*, 2013.

[4] H. M. Barros, A. Castro, R. N. Fontoura Filho, M. Groetaers dos Santos, C. F. Teodósio Soares, "Efficient Application of Parallel Processing with Standard Tools for Electromagnetic Transients Simulation," in *International conference on Power System Transients*, 2013.

[5] X. Legrand, "Outil de co-Simulation pour EMTP-RV, présentation et application à une étude EMTP/Simulink," EDF R&D, Clamart, France, Internal technical report, H-R26-2012-00810-FR, 2012.

[6] MODELISAR Consortium, "Functional Mock-up Interface for Co-Simulation," *ITEA 2*, 07006, 2010.

[7] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, "Semaphores," in *Operating Systems: Three Easy Pieces*, Arpaci-Dusseau Books, 2014, pp.1-18.

[8] J. Mahseredjian, "DLL programming in EMTP," *EMTP-EMTPWorks help files*, 2012, pp.1-24.

[9] L. Gérin-Lajoie, J. Mahseredjian, "IEEE-39 benchmark for electromagnetic transients in EMTP-RV," 2014, available on request.

[10] IEEE Task Force on Load Representation for Dynamic Performance, "Load representation for dynamic performance analysis," *IEEE Trans. on Power Systems*, Vol. 8, No. 2, May 1993.

[11] U. Karaagac, "Realistic test system for simulating power system transients in EMTP," 2014, available on request.

[12] J. Peralta, H. Saad, S. Dennetière, J. Mahseredjian and S. Nguefeu, "Detailed and Averaged Models for a 401-level MMC-HVDC system," *IEEE Trans. on Power Delivery*, vol. 27, no. 3, July 2012, pp. 1501-1508.

[13] H. Saad, S. Dennetière, J. Mahseredjian, P. Delarue, X. Guillaud, J. Peralta, S. Nguefeu, "Modular Multilevel Converter Models for Electromagnetic Transients," *IEEE Trans. on Power Delivery*, vol. 29, no. 3, pp.1481-1489, June 2014.