A Novel Co-Simulation Concept using Interprocess Communication in Shared Memory

Christian Scheibe, Anatoli Semerow, Jasmin Menke, Piergiovanni La Seta
Siemens AG, Energy Management DG PTI, Erlangen, Germany
christian.scheibe@siemens.com

Alexander Raab, Gert Mehlmann, Matthias Luther
FAU, Institute of Electrical Energy Systems, Erlangen, Germany
alexander.raab@fau.de

***Abstract*—The paper describes a novel co-simulation concept for power system analysis using interprocess communication via shared memory. This innovative approach enables manufacturers, utilities and other parties a continuous and efficient use of simulation models within the phases of preliminary investigations, planning, design and operation. It paves the way for the practical implementation of the concept of a digital twin. Besides the technical specification of the process communication, the realization of process synchronization and the model interfaces are comprehensively discussed. Performance topics are practically treated. Finally, as an application of the novel concept, a successfully implemented study in a power system simulation tool is demonstrated in order to prove its practical feasibility.**

***Index Terms*—Co-Simulation, Interprocess Communication, Shared Memory, Power System Simulation, Digital Twin.**

I. Introduction

Recent developments and future trends in electric power systems worldwide involve a rapidly growing complexity of asset models for different types of investigations and jeopardize the consistency of the basis for evaluation in power system planning, equipment design and operation. It implies a substantial need for an efficient and consistent use of power system models, whereby this challenge is faced by the concept of a digital twin [1]. The main objective of this approach is to avoid unnecessary expenses as well as to prevent data discrepancies in the development and operation process.

Such a typical use case arises at integration projects of new assets in existing power systems. Within the different implementation phases of such a technical project, its common practice that manufacturers, utilities and other parties manage several models in different simulations environments.

Power system models under investigation contain controls, physical behavior as well as different conditions regarding the dimensions of time and probabilities. Employed network models may completely meet the requirements of one party while being insufficient for the others. Control algorithms are designed to run on a target hardware such as microprocessors. Within integration projects, the functional structure is then usually migrated to a required type of a model into utilities environment which takes physics, the control and protection sphere into account. This conversion procedure is used to be completely manual. The entire control structure needs to be analyzed and remodeled to enable a feasibility for the target environment. These laborious steps need to be repeated for each simulation tool in use within an integration project. Have been converted, these control and equipment models enable the responsible utilities performing feasibility studies related to stationary and dynamic behavior of a system. However, even a minor parameter change may cause a need for an update of the entire model chain for physics and controls. Obviously, that fact causes enormous efficiency and time losses.

Fundamentally, there are four simulation techniques [2]. The *Classic Simulation* with one model and one solver, used for instance for power flow or control system studies [3], is the most common method. For a combination of a mathematical model and its appropriate control system, as wind turbines [4] or offshore wind farms [5], *Hybrid Simulation* can be utilized. This uses models of different representations with a common solver. In order to reduce the demand of computing capacity for large power system models, it is advisable to separate relevant and non-relevant model parts in an EMT or rather RMS frame. The *Parallel Simulation* divides the entire model and runs it in parallel using multiple solvers. In [6] this approach is applied to an HVDC model for stability studies using EMT and RMS partitions in PSS$^{\circledR}$NETOMAC [7]. The disadvantage of these three methods is the limitation to one simulation tool and can be coped by the *Co-Simulation*.

A *Co-Simulation* based approach utilizing the shared memory of computers is presented in that paper. It describes according to Fig. 1 the communication and synchronization between processes and the model interfaces in detail.

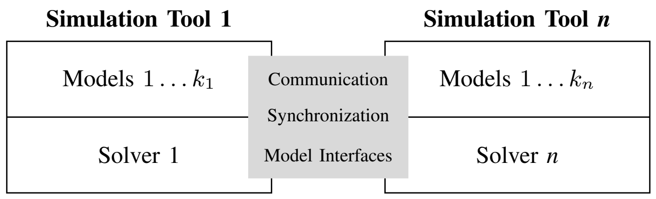

Figure 1: Fundamental concept of Co-Simulation

The figure indicates that it is possible to use a variable number of models and solvers, which allows a coupling of various simulation tools. In [8] a concept is illustrated as a potential solution for the challenges pointed out above. It does

978-1-7281-1981-6/19/$31.00 ©2019 IEEE

not only simplify the controller development process but also eliminates the need to convert the model from one simulation tool to another. The re-utilization of the models within their native environments entails great advantages with respect to effort and time within power system integration projects. The main components of such a co-simulation approach include communication, synchronization and the model interface.

Some further solutions exist for this challenge. A concept of co-simulation utilizing the IEC61850 standard can be found in [9]. Another architecture for power system communication with sockets for interprocess communication is presented in [10] while the smart grid simulation API MOSAIK makes use of syntactic and semantic layers [11]. A further approach, the Functional Mock-up Interface (FMI) as a master-slave concept, is introduced in [12]. In contrast to the existing solutions the presented approach particularly focuses on the use of manufacturer controllers in hybrid power system models with very high requirements to the calculation performance. The paper comprehensively describes the concept of the developed solution and its implementation in the state-of-the-art simulation tool PSS®NETOMAC.

II. Proposed Concept for Co-Simulation

The coordination between different simulation tools needs a common interface that has to be designed and developed comprising the requirements to be:

- Lean: producing a small overhead so that there is no significant increase of simulation time
- Universal: being usable by as many tools as possible
- Robust: controlling simulation iterations if needed

The fundamental basis of this interface concept is the approach of interprocess communication (IPC). This enables individual threads and processes to share information and data as well as to synchronize with each other. Some investigated options of IPC are, but not limited to:

- Sockets
- Pipes and Named Pipes
- Shared Memory

Sockets enable interprocess communication using a server-client scheme. While they are easy to access – due to the existence of an application programming interface (API) – for many applications and languages, they also provide a certain amount of data overhead. This is due to the fact that the operating system provides the entire framework for the data exchange and the process synchronization. Even though, only a single socket instance is needed for the communication since sockets are uni-directional as shown in Fig. 2.

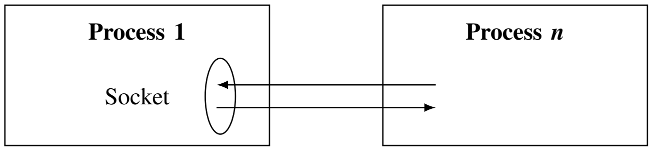

Figure 2: Concept of IPC by Sockets

Also a widely used concept of IPC is the utilization of pipes as depicted in Fig. 3. These are, as already with Sockets, provided by an API originated from the operating system. Pipes allow for end-to-end data transfer in a single direction. Their API also enables handling of the communication and synchronization between multiple processes.

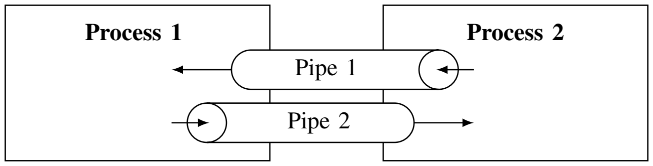

Figure 3: Concept of IPC by Pipes

Both Sockets and Pipes require memory to be at least copied from one point to another in order to be accessible. Although their API provides ease of use but overhead as well. These effects limit their performance to a certain level. Since the major requirement of performance needs an efficient communication in order to avoid bottlenecks, the Shared Memory alternative needs to be investigated.

The proposed solution is based on Shared Memory (SHM) according to Fig. 4. This concept has a single address in memory to be used by different processes, quasi at the same time. The Shared Memory API does usually not provide a method for synchronization of processes itself. However, the approach minimizes memory access since only one memory section is used by multiple processes simultaneously. Hereby, no synchronization mechanism is provided to inform a process that new data is available and has to be done separately.

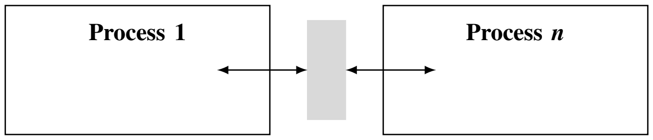

Figure 4: Concept of IPC by Shared Memory

A. Interprocess Communication Scheme

The communication between processes can be specified up to a single bit for each of the two participants. This is possible due to the rudimentary data format being transferred (the data points are floating point figures as elements of arrays) and the ease of using only two partners. For its high performance and the relatively simple communication scheme, the Shared Memory approach is chosen in that research project.

An introduction of a ”ready”-bit becomes necessary since any non-atomic (simultaneous read and/or write) memory operation performed by the two processes may result in an undefined state. This may produce wrong values used in a simulation time step. This bit signalizes the other process whether Shared Memory is being accessed or not, effectively limiting Shared Memory access to only one process at any point of time. Thus, read and write operations are ensured to be atomic. For Sockets and Pipes this issue is already solved within their API. For the Shared Memory concept the utilization of (named) event handles is chosen. The event handles allow processes to wait for a specified event while also falling into an inactive state. They provide a good performance while minimizing overhead from operating system.

With that synchronization tool at hand, the communication scheme itself can be developed. The process, which accesses a certain Shared Memory location as first, automatically creates this scheme and is designated to be the "master" process. After the set-up of the Shared Memory section and the completion of event handles, the process waits for the connection of a communication partner. The second process, that finds the already available Shared Memory sector, automatically connects to that and is now referred as the "slave" process. Now that both partners are connected to the same Shared Memory address, the simulation tools may initialize the simulation process until the inputs of the Shared Memory blocks are accessed. At this point of time, the communication loop which is triggered at every major time step of the simulation tool, is launched.

The communication loop controls the access of the communication partners to the Shared Memory section. Initially, the master provides its inputs (from the simulation tool) to the Shared Memory instance. This one then signalizes to the slave process that new data has been provided and waits in turn for it to finish its loop. The slave process begins at this point with writing its inputs to the Shared Memory instance. After that, the masters data have been read and the corresponding events are set. Finally, the master continues reading slaves data, setting the handles and the loop to be finished. This com loop routine is executed at each simulation time step. Finally, when the last time step has been calculated by the simulation tools, a final communication loop is executed and the shutdown routines free the Shared Memory section. The proposed communication process including initialization, communication loop and shutdown is pointed out in Fig. 5.

A generalized interface has been developed based on the communication scheme described above. The main objective of the interface is to efficiently couple different simulation tools at calculation procedures. The core module provides the data exchange, synchronization and communication itself. This modular integration part can be reused and adapted for any simulation tool. The adaptation module connects the Shared Memory interface with the simulation tools external code interface. The concept of separation of core and integration modules provides compatibility throughout all simulation tools served. Any change in the core module does not affect the integration. Only re-testing and compilation is needed. The core module provides the basic functionality by three functions:

- Connection and setup of the IPC
- Communication loop of the IPC
- Shutdown of the IPC

These IPC functionalities may be used by executing wrapper functions within the specific simulation tools. The connect and shutdown functions are each executed once at the beginning and at the end of a simulation process. The loop is executed each time a data transmission is intended.

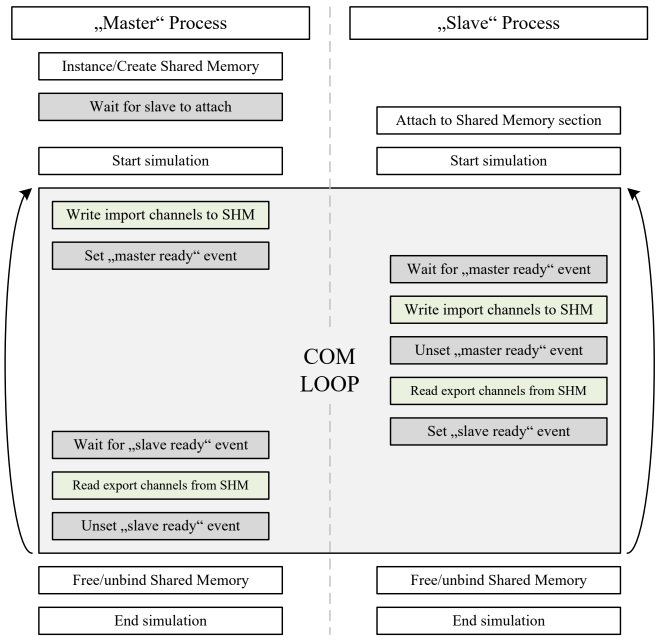
„Master“ Process
„Slave“ Process
Instance/Create Shared Memory
Wait for slave to attach
Attach to Shared Memory section
Start simulation
Start simulation
Write import channels to SHM
Set „master ready“ event
Wait for „master ready“ event
Write import channels to SHM
Unset „master ready“ event
Read export channels from SHM
Set „slave ready“ event
COM LOOP
Wait for „slave ready“ event
Read export channels from SHM
Unset „slave ready“ event
Free/unbind Shared Memory
Free/unbind Shared Memory
End simulation
End simulation

Figure 5: Communication process using Shared Memory

The usability of co-simulation highly depends on the performance of its interfaces. If the interface slows down the simulation tool significantly, the co-simulation itself may be in question. Therefore, the Shared Memory approach can prove to be competitive, making co-simulation easy to use and performant at the same time.

B. Model Interfaces

As two of three requirements in Fig. 1 are handled by the IPC scheme, the third one is treated in the following. The design of the model interfaces depends on the intended purpose of the IPC. For control structures containing a simple data transfer with numerical values there may be concepts of hierarchical or decentralized instances as shown in Fig. 6.

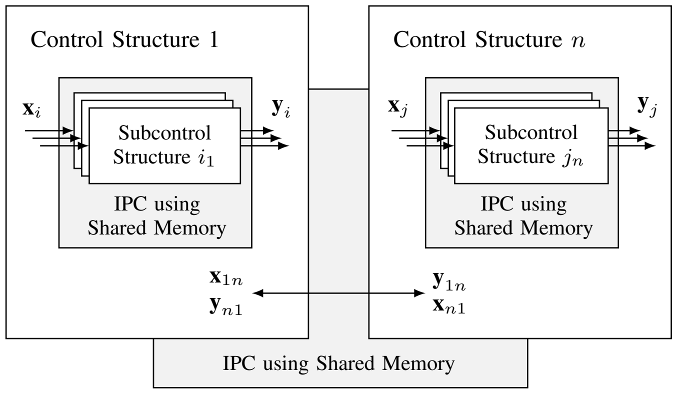

Figure 6: Model interface for control structures

While the Shared Memory module provides synchronization on a per-timestep-basis, the simulation tool needs to transfer this into the simulation time step. For instance, if the objective is to couple a controller development tool running at a time step of 1 msec with an electrical network simulation tool running at 100 μsec, proper measures need to be taken in order to ensure that the signals are provided at identical time steps. This might be done by re-sampling the corresponding signal with measures like interpolation and (low-pass) filtering.

For the co-simulation of network models, which of course may each have attached controller models processes, the model interfaces are more sophisticated and have to be primarily defined with regard to the simulation types and secondarily with regard to the connection type. While parallel RMS simulations may be connected exchanging the symmetrical components data, combined RMS-EMT simulation types require signal processing in form of space vector composition (SVC) and decomposition (SVD) in order to translate electrical signals instantaneous values to and from fundamental frequency vectors of the symmetrical components as shown in Fig. 7.

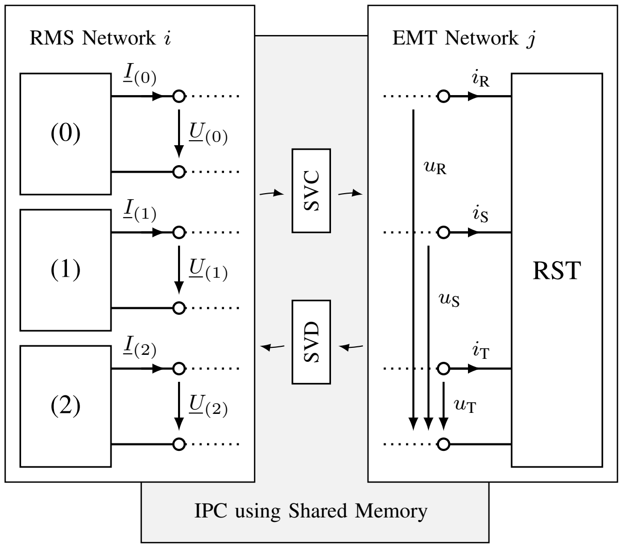

Figure 7: RMS-EMT simulation interface

For the second dimension of network connection types there are mainly the two concepts of exchanging the data between a Thevenin and a Norton equivalent (Fig. 8) or rather between two Thevenin equivalents each representing the counterpart within a co-simulation instance.

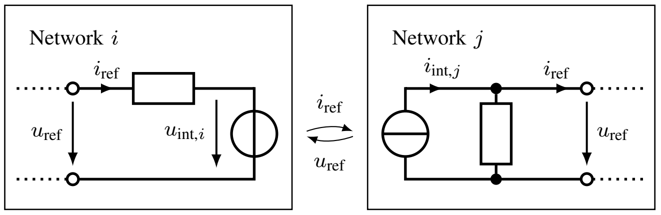

Figure 8: Thevenin-Norton network interface

III. Shared Memory Performance Considerations

A performance test for the Shared Memory approach was conducted in order to identify sweet spots and to answer the question whether multiple Shared Memory instances should be preferred against a large single instance.

The measurements were taken on a Fujitsu Celsius H type Notebook utilizing a four core Intel CPU of the 4th generation with 16 GB of memory. Only the time of the data transfer itself was measured, not the time for the simulation. A mean over 100 data transfers was used to get comparable values. The amount of data transferred translates as follows:

- 3 Bytes of status words
- 8 Bytes per datapoint to be transferred in each direction

For the performance measurements, a data point translates in 19 Bytes and 5,000,000 to about 76.3 MB per time step.

The diagram shown in Fig. 9 indicates the relative time for the amount of data transferred in parallel. The time for a single transfer is normalized to the first value. It mainly shows that the time itself scales linear with the amount of data points. Also, multiplying the base amount of data points by the factor of ten only increases time by about 6 %.

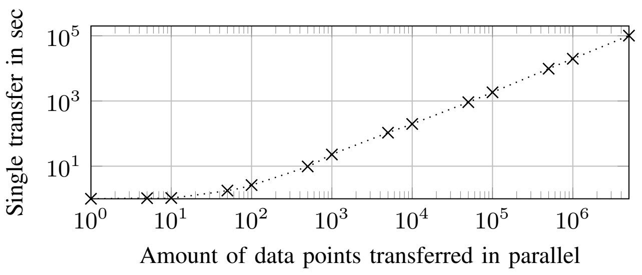

Figure 9: Relative time for the data transfer

Conclusively, the Shared Memory interface indicates a high performance as well as good scalability. Increasing the bandwidth is the better option than instancing multiple Shared Memory sections simultaneously and shall be applied whenever possible. This results from the overhead of calling the function multiple times instead of a single time as well as the (minimum but existent) overhead of the status words. Also, from a hardware point of view reading from and writing to a single range of memory is more performant than doing the same on multiple, fragmented memory elements.

IV. Feasibility Study using PSS®NETOMAC

Any commercial simulation software has its own specifications for integration of foreign program code. The investigated simulation tool PSS®NETOMAC utilizes among others the DLL interface (so-called ESE-interface) according to IEC61400-27-1F. Using this method, a DLL is compiled according to the defined structure whose functions can be called externally. Although, this interface initially intended to support wind turbine models in different simulation tools, it can serve a lot of varios applications as for instance the integration of the Shared Memory data exchange module.

Utilizing the ESE-interface, the corresponding functions are bound in the following way:

- Model_Initialize() sets up the Shared Memory sections
- Model_Outputs() calls the communication loop and handles the simulation data exchange
- Model_Terminate() handles the shutdown sequence

One objective of the Shared Memory module is a minimum impact on the simulation performance, namely it should not noticeably increase simulation times. Investigations show that using the above mentioned computer a simulation time step in a simple PSS®NETOMAC model takes about $5-20\,\mu$sec. The Shared Memory module creates an overhead and increases the effort. On the same system, the Shared Memory interaction takes about $10-25\,\mu$sec, if two identical PSS®NETOMAC instances communicate with each other. Within this time frame the co-process does its calculation.

If the model complexity is increased, simulation times increase as well. The slower one of the processes dominates the needed step time. Due to synchronization the time needed is practically identical to the processes. Hence, the more populated a simulation model becomes, the less part does the time effort of the Shared Memory module play a role, since the communication time needed keeps nearly the same.

The feasibility study with a simple power system model is performed by a co-simulation of RMS-EMT simulation type utilizing one Thevenin-Norton (RMS-EMT) connection point. Both network parts are each provided by an own PSS®NETOMAC instance. Values from the EMT network are processed within the SVD using Park transformation and low-pass filter according to [13] without affecting the phase with respect to the initial signal. Due to different times steps of the solvers a linear interpolation is used within the SVC, transforming RMS to EMT signals.

Fig. 10 and 11 show the voltage and current of phase R or rather of positive sequence at the connection point of the two co-simulated networks. The results show signals for a 3phase short-circuit in the EMT part with a duration of 0.5 sec.

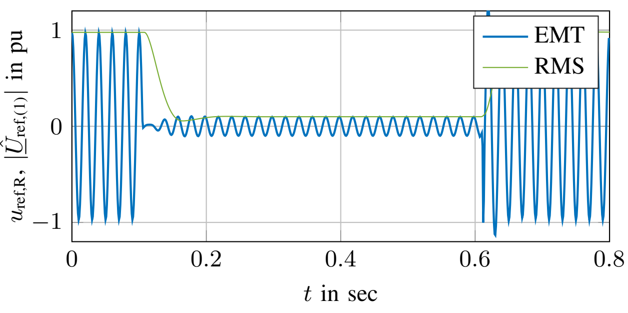

Figure 10: Voltage results in feasibility study

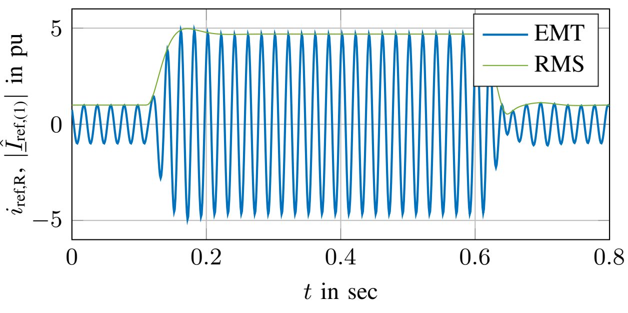

Figure 11: Currents results in feasibility study

In a first step, the results verify the feasibility of the presented co-simulation concept using Shared Memory and applying an RMS-EMT simulation type for a simple power system model. The signals are exchanged between two instances and are accurately considered in the each counter part network, so that system characteristics are inherited.

V. Conclusion

A performant interprocess communication interface utilizing Shared Memory is developed in this project. The paper describes the concept of the communication, synchronization and the model interfaces in detail. A demonstrated performance test indicates a high efficiency with a low overhead for data exchange as well as good scalability. The feasibility of the concept is successfully proven for the simulation tool PSS®NETOMAC. In future works it is intended to extend the feasibility tests to other simulation tools and types in order to pave the way for the practical implementation of the concept of a digital twin. Further developments include the investigation of possible convergence and initial condition issues.

References

[1] C. Brosinsky, D. Westermann and R. Krebs, Recent and Prospective Developments in Power System Control Centers: Adapting the Digital Twin Technology for Application in Power System Control Centers, In *2018 IEEE Int. Energy Conf. (ENERGYCON)*, Limassol, Cyprus, 2018.

[2] F. Schloegl, S. Rohjans, S. Lehnhoff, J. Velasquez, C. Steinbrink, P. Palensky, Towards a classification scheme for co-simulation approaches in energy systems, In *2015 Int. Symposium on Smart Electric Distr. Sys. and Techn. (EDST)*, pages 516–521, Vienna, Austria, 2015.

[3] R. D. Zimmerman, C. E. Murillo-Sanchez and R. J. Thomas, MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education, *IEEE Trans. on Pow. Sys.*, 26(1):12–19, Feb 2011.

[4] A. Tapia, G. Tapia, J. X. Ostolaza and J. R. Saenz, Modeling and Control of a Wind Turbine Driven Doubly Fed Induction Generator, *IEEE Trans. on Energy Conv.*, 18(2):194–204, June 2003.

[5] C. Meyer, M. Hoing, A. Peterson and R. W. De Doncker, Control and Design of DC Grids for Offshore Wind Farms, *IEEE Trans. on Industry Appl.*, 43(6):1475–1482, Nov 2007.

[6] G. Deiml, C. Hahn, W. Winter and M. Luther, A Novel Dynamic Model for Multiterminal HVDC Systems based on Self-Commutated Full- and Half-Bridge Multilevel Voltage Sourced Converters, In *2014 16th European Conf. on Power Electron. and Appl.*, pages 1–13, Lappeenranta, Finland, 2014.

[7] O. Ruhle, Modern Power System Analysis Tools, In *2011 EPU-CRIS Int. Conf. on Science and Technology*, Hanoi, Vietnam, 2011.

[8] P. Palensky, A. van der Meer, C. Lopez, A. Joseph and K. Pan, Applied Cosimulation of Intelligent Power Systems: Implementing Hybrid Simulators for Complex Power Systems, *IEEE Industrial Electronics Magazine*, 11(2):6–21, June 2017.

[9] M. Manbachi, A. Sadu, H. Farhangi, A. Monti, A. Palizban, F. Ponci and S. Arzanpour, Real-Time Co-Simulation Platform for Smart Grid Volt-VAR Optimization Using IEC 61850, *IEEE Transactions on Industrial Informatics*, 12(4):1392–1402, Aug 2016.

[10] M. Mirz, L. Razik, J. Dinkelbach and et al., A Cosimulation Architecture for Power System, Communication, and Market in the Smart Grid, *Complexity*, 2018:12, Aug 2018.

[11] S. Schütte, S. Scherfke and M. Sonnenschein, MOSAIK - SMART GRID SIMULATION API: Toward a semantic based standard for interchanging smart grid simulations, pages 14–24, Porto, Portugal, April 2012.

[12] J. Bastian, Ch. Clau, S. Wolf and P. Schneider, Master for Co-Simulation Using FMI, Dresden, Germany, 2011.

[13] F. Plumier, P. Aristidou, C. Geuzaine and T. Van Cutsem, Co-Simulation of Electromagnetic Transients and Phasor Models: A Relaxation Approach, *IEEE Trans. on Power Del.*, 31(5):2360–2369, Oct 2016.