Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

MODELING AND REAL-TIME SIMULATION OF MICROGRID COMPONENTS USING SYSTEMC-AMS

Rahul Bhadani
Gabor Karsai

Institute for Software Integrated Systems
Vanderbilt University
1025 16th Ave S, Suite 102
Nashville, TN 37212, USA

Hao Tu
Srdjan Lukic

Electrical and Computer Engineering
North Carolina State University
890 Oval Drive, 3114 Engineering Building II
Raleigh, NC 27606, USA

ABSTRACT

Microgrids are localized power systems that can function independently or alongside the main grid. They consist of interconnected generators, energy storage, and loads that can be managed locally. Using SystemC-AMS, we demonstrate how microgrid components, including solar panels and converters, can be accurately modeled and simulated, along with their interactions. Real-time simulations are crucial for understanding microgrid behavior and optimizing components. This approach facilitates seamless integration with hardware prototypes and automation systems, supporting various development stages. Our study presents a best-case scenario for real-time simulation, assuming each loop takes less time than the simulation time step, with fallback to the previous value if data isn't received in time. This article introduces the first known real-time simulation strategy using SystemC-AMS, enabling the real-time simulation of microgrid components and integration with external devices. The implementation adopts a model-based design approach, creating increasingly complex systems with grid components and controllers.

1 INTRODUCTION

Microgrid is a small-scale grid consisting of various loads and distributed energy resources designed and installed to serve a small community such as a neighborhood, university campus, or office space. Microgrids can work in two forms: islanded and grid-connected (i.e. can interoperate with the main grid). There are many complex tasks related to the control and management of microgrids that requires precise modeling and analysis of microgrid plants and associated controllers before moving to real implementation. Such tasks include power flow control, synchronization, energy management, stability, etc. Microgrid control is often implemented using a hierarchical architecture (Guerrero et al. 2011), which comprises primary control, secondary control, and tertiary control. The primary control involves local control to stabilize the microgrid, while the secondary control provides frequency and voltage regulation. The tertiary control is based on the economics of the electricity market. The three hierarchical control levels are usually executed at different speeds and time granularities while being coordinated through the exchange of information between them. Traditionally, researchers and practitioners have utilized offline and as-fast-as-possible (AFAP) simulations for implementing and studying microgrid control. AFAP simulations are inexpensive and easy to implement and analyze. However, they do not accurately represent the discrete nature of a controller's computer implementation that interacts with the continuous-time nature of the physical world. AFAP simulators lack in providing capabilities for transient response and may not be suitable for interfacing with external hardware for hardware-in-the-loop simulation. In this paper, we address these shortcomings of microgrid simulation using SystemC-AMS and ZeroMQ library and present a use case of secondary control for a DC microgrid.

1.1 Background and Literature Review

Usually, simulation tools such as PLECS (Asadi and Eguchi 2019), Simulink, OpenDSS (Montenegro et al. 2012), or GridLab-D (Chassin et al. 2008) can be used for microgrid modeling and control. Such modeling and simulation tools can be used for prototyping control and grid management algorithms. However, PLECS, Simulink, and GridLab-D are not meant for real-time simulation. OpenDSS can be used for real-time simulation but its use cases are limited to static or steady-state simulation such as power flow and are not suitable for dynamic simulation such as electromagnetic transient response. A real-time simulation tool for transient response and dynamic situations such as fast-changing voltage fluctuations is required for determining the most suitable value of control parameters and deciding on the appropriate grid component for installation. OPAL-RT (Bian et al. 2015) is a real-time simulator developed by a Canadian company for power systems simulation. However, it is a hardware-in-the-loop simulator, requiring specialized hardware in addition to being prohibitively expensive.

When designing a controller for a microgrid, the controller parameters should be selected based on the discrete nature of the computer implementation, the dynamic behavior of microgrid components, and the finite latency of communication systems. A real-time simulator should not only operate in a steady state but should also be capable of simulating transient responses and rapidly changing fluctuations of signals such as voltage and current. Moreover, a real-time simulator must ensure timing properties with acceptable variability, which can vary depending on the application. While OPAL-RT is capable of simulating microgrid components and primary or secondary controllers to study transient responses in real-time, we are looking for a solution that does not require specialized hardware and can utilize simpler models.

In this paper, we use the SystemC and SystemC-AMS libraries to develop a real-time simulator that employs the SystemC-AMS model of computations, such as Time Data Flow (TDF) and Electrical Linear Network (ELN), to generate models of grid components and implement primary and secondary controllers. Initially designed for functional and architectural-level modeling and simulation of mixed-signal subsystems and hardware/software subsystems, SystemC-AMS (P1666.1 - SystemC AMS Extensions Working Group 2016) allows for the interaction of these components. To address the needs of industries like telecommunications, automotive, and semiconductors, SystemC-AMS extensions were developed to provide a uniform and standardized methodology for modeling heterogeneous AMS/HW/SW systems. These extensions are built on top of the SystemC language standard and define additional language constructs that introduce new execution semantics and system-level modeling methodologies to design and verify mixed-signal systems. The models of computation supported by the SystemC-AMS extensions include Timed Data Flow (TDF), Linear Signal Flow (LSF), and Electrical Linear Networks (ELN). TDF introduces discrete-time modeling and simulation without the overhead of the dynamic scheduling imposed by the discrete-event kernel of SystemC. LSF supports the modeling of continuous-time behavior by offering a consistent set of primitive modules, and ELN supports the modeling of electrical networks by instantiating predefined linear network primitives such as resistors or capacitors.

We present a method for simulating microgrid components and controllers in real time using SystemC-AMS and ZeroMQ. For real-time simulation of microgrid components, we use TDF and ELN models of computations (MoC). TDF MoC is suitable for modeling transfer-function or state-space models of the plant or to implement a computing algorithm. ELN MoC is suitable for at-detailed circuit-level implementation. This paper presents an example of a DC microgrid and relevant voltage controller that uses ELN MoC respectively. By combining various models of computations of SystemC-AMS, the publisher-subscriber communication paradigm from ZeroMQ (Hintjens 2013), and the high-precision Chrono C++ library for scheduling, our method functions as a real-time simulator. Our approach to creating microgrid components employs a model-based design that relies on the COSIDE tool (Einwich et al. 2022). With model-based design, components can be reused to create models for complex microgrids. In this paper, we only target soft real-time, i.e., missing a few deadlines doesn't have any safety-critical implications.

2 SYSTEMC-AMS FOR MODELING & SIMULATION

SystemC is a design and verification library written in C++ that facilitates the design and verification of hardware systems (Panda 2001), (Swan 2001). SystemC allows for the creation of models of hardware components and systems that can be simulated at various levels of abstraction. It has gained popularity in electronic design and automation (EDA). SystemC-AMS (SystemC for Analog/Mixed-Signal) was created to extend the capabilities of SystemC to include modeling and simulation of analog and mixed-signal components (Vachoux et al. 2004). SystemC-AMS provides support for continuous and discrete-time signal processing.

We model microgrid components, such as a DC-DC converter, inverter, or a circuit representation of a solar panel, using ELN MoC in SystemC-AMS. ELN MoC is suitable for continuous-time modeling. On the other hand, we can model transfer-function or state-space-based systems using TDF MoC, which is suitable for designing and implementing digital control algorithms. SystemC-AMS, in contrast to traditional circuit design tools such as SPICE (Rashid 2017), abstracts low-level details that significantly aid in the incremental design and analysis of a circuit component.

Discrete-time simulation in SystemC-AMS is managed using TDF MoC. A TDF model can comprise a number of connected TDF modules that form a TDF cluster, internally represented as a directed graph. A TDF module is described by a set of input and output ports, a time-step for each module and its ports, delay, and time-offset. In the case of a feedback system, an algebraic loop is encountered. To break an algebraic loop, a port delay must be introduced into the system. ELN MoC is used to implement the circuit-level system in the continuous domain using a set of predefined primitives such as resistors, capacitors, inductors, controlled voltage sources, and so on. ELN models are always structural models and are used for implementing continuous dynamic and conservative behavior. To interact with a TDF module, special converter ports must be used to translate data between TDF and ELN MoCs.

SystemC-AMS simulation is conducted in a sequential fashion after the SystemC kernel generates a directed graph of the connected modules. The scheduling of the execution is determined automatically by the underlying kernel. Equations involving various modules are solved by a dedicated linear differential-algebraic equation solver. The behavior of discrete-time models is defined in a member function called `processing` of a TDF module. Continuous-time models may be implemented in discrete time as a Laplace transform or a state-space model. On the other hand, an ELN module is instantiated as a child module of a SystemC parent module using the `SC_MODULE` macro.

SystemC-AMS has been used for modeling and simulating production systems (Fraccaroli and Vinco 2022), electric vehicle energy management systems (Chen et al. 2019), electronic systems (Alekhin 2020), thermal simulations (Chen et al. 2016), and MEMS (Vernay et al. 2015), among others. However, all of these works have used standalone SystemC-AMS simulations with no communication components, and they employed the AFAP simulation strategy. Contrary to the previous approaches, our work presented in this paper incorporates a communication component to enable real-time simulation, and it can be used as a digital twin for microgrid components and their control. Such simulations can be used for distributed control. However, in this paper, we only focus on single-process control.

3 DESCRIPTION OF THE REAL-TIME SIMULATOR

A general design of a microgrid is presented in Figure 1 that consists of a plant model and several controllers exchanging data through some network interfaces. To model microgrid components, we use SystemC-AMS for its capability to model and simulate analog/mixed-signal systems, including hardware-software systems and control algorithms. Specifically, we use the TDF MoC to model a grid component using the transfer function or state-space-based model. The TDF MoC also allows for the implementation of control algorithms and other computing procedures. Additionally, we use the ELN MoC to implement detailed circuit-based components for a grid component.

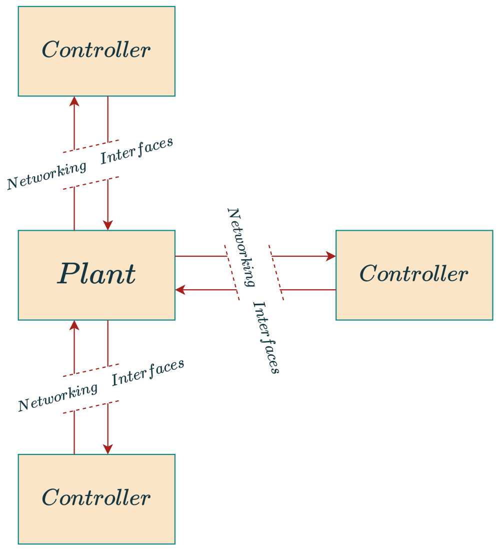


Figure 1: A general design of a microgrid using software-in-the-loop simulation with the plants and controller exchanging data through communication interfaces.

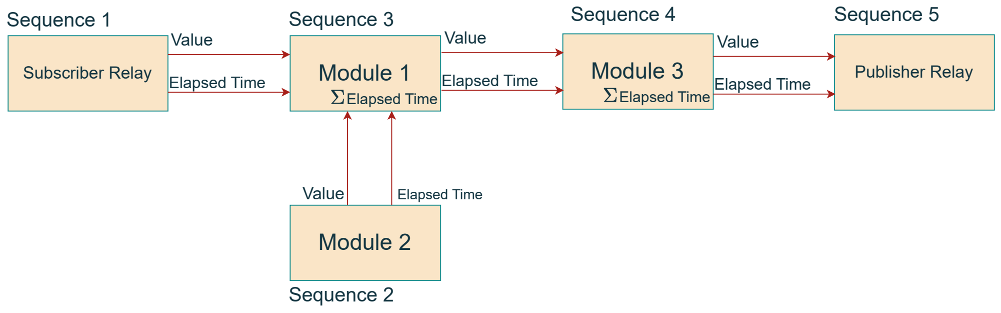


Figure 2: Flow of information and cumulative elapsed time of computation in a SystemC-AMS implementation of a system for real-time execution.

Our approach to implementing microgrid components involves utilizing model-based design with the aid of a visual drag-and-drop software tool called COSIDE (Einwich et al. 2022). COSIDE facilitates a modular approach to creating TDF modules by generating skeleton code that programmers can use to implement their algorithms. These TDF modules can then be used as library blocks for creating complex systems. Additionally, COSIDE provides the ability to create ELN schematics using primitives such as resistors, inductors, capacitors, and other circuit components. The schematics are saved as XML files that are used to generate SystemC `SC_MODULE` classes. Schematics can be further used as blocks for creating more complex systems in combination with TDF modules.

Components of microgrids communicate with each other through network interfaces. For the work presented in this paper, we use ZeroMQ (Hintjens 2013) for communication interfaces. We use ZeroMQ's publisher-subscriber mechanism for exchanging data between components of the microgrid. However, as SystemC and its extension SystemC-AMS are designed to simulate as fast as possible (AFAP), they cannot work with ZeroMQ off-the-shelf. The theoretical data rate of communication interfaces in AFAP simulators exceeds the data rate that ZeroMQ is capable of handling. AFAP simulation can generate data at frequencies of 10000 Hz or higher, which is beyond the physical limitations of both the ZeroMQ APIs and the hardware that utilizes them. Therefore, we require scheduling for message delivery and handling to ensure that messages are processed in a timely fashion to imitate real-time behavior.

To implement real-time behavior, we create a custom TDF module that uses ZeroMQ APIs to create a communication interface. ZeroMQ relay modules corresponding to ZeroMQ publishers and subscribers are also created as TDF modules. The subscriber relay module subscribes to a ZeroMQ topic on a specific

TCP port, reads messages, and writes them to an output port of the relay module. The subscriber relay module outputs the elapsed time for each simulation step and passes it on to the next module in the system. Elapsed time is defined as the time taken to read a message from a ZeroMQ subscription until it is available at the output port of the relay module, as shown in Figure 2. The subsequent TDF module in the system takes the input value of the output from the ZeroMQ subscriber relay module and the elapsed time. The following TDF module receives input from the previous TDF module along with the cumulative elapsed time, performs some computation, and outputs the specified value to the output port along with the cumulative elapsed time. The C++ Chrono library (C++ References 2022) is used to measure the time elapsed while performing a computation. The final module in the system is the ZeroMQ relay block that publishes the information taken as input from the previous modules. The relay publisher module computes the total wall-clock time elapsed in the given SystemC simulation step and sleeps using the Chrono C++ API for the amount of time until the specified wall-clock time-step has passed for the given SystemC simulation step. The idea is depicted in Figure 2. The cumulative elapsed time calculation exploits the fact that SystemC execution is sequential and the SystemC execution kernel constructs a scheduling graph for the sequential execution.

In addition, we execute the SystemC simulation on Ubuntu with a PREEMPT_RT kernel patch (Arthur et al. 2007) (Reghenzani et al. 2019) and run the generated executable corresponding to the plant and controller on separate cores in a multicore CPU with the highest priority. To provide the most reliable deterministic simulation result, we disable hyperthreading. We also set the highest scheduling priority for SystemC-AMS processes. Before instantiating the SystemC-AMS simulation, we further disable memory swapping and lock the memory using a system call `mlockall`. Locking memory prevents any page fault during simulation execution and provides a greater deterministic behavior.

4 CASE STUDY: DC MICROGRID CONTROL

We use SystemC-AMS to simulate a DC microgrid in real time. We stabilize the DC microgrid with droop control as the primary control, and later, we also include the secondary control. A single-bus DC microgrid can be modeled by a representative system consisting of a DC-DC converter, an inductor, a capacitor, and a load (Tu et al. 2023) (refer to Figure 3). The equations that describe the DC microgrid can be written as in Equation (1).

$$\begin{aligned} V_{\text{ref}}(t) - V(t) &= L\frac{di}{dt} \\ i(t) - \frac{V(t)}{R} &= C\frac{dV}{dt} \end{aligned} \tag{1}$$

where V is the capacitor voltage; i is the inductor current; V_0 and V_{ref} are the converter's output voltage and reference voltage, respectively. In Equation (1), it is assumed that the output voltage can track the reference voltage accurately thanks to the converter's inner control loops, i.e., $V_0 = V_{\text{ref}}$.

The reference voltage V_{ref} is generated by a primary controller, which is droop control as follows:

$$V_{\text{ref}}(t) = V_n - K \cdot i(t)$$

where V_n is microgrid's nominal voltage; K is the droop gain. With droop control serving as the primary controller, the reference voltage deviates from the nominal voltage when the output current is not zero.

To maintain the voltage for a droop-controlled DC microgrid at the nominal value, a secondary controller can be used, which retunes the nominal voltage V_n with a correction term $\delta(t)$. For the use case presented in this article, the secondary controller, along with the primary controller, is governed by the following

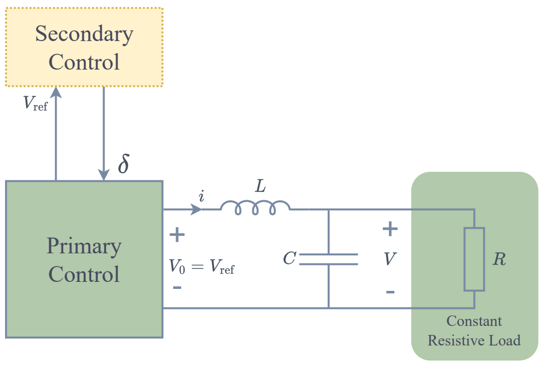


Figure 3: A simplified microgrid model with a constant resistive load. The Droop controller is used as the primary controller that determines the reference voltage based on a nominal voltage and output current. Optionally, a secondary controller can be used to adjust the nominal voltage of the primary controller to regulate the microgrid voltage with an additional correction term $\delta(t)$.

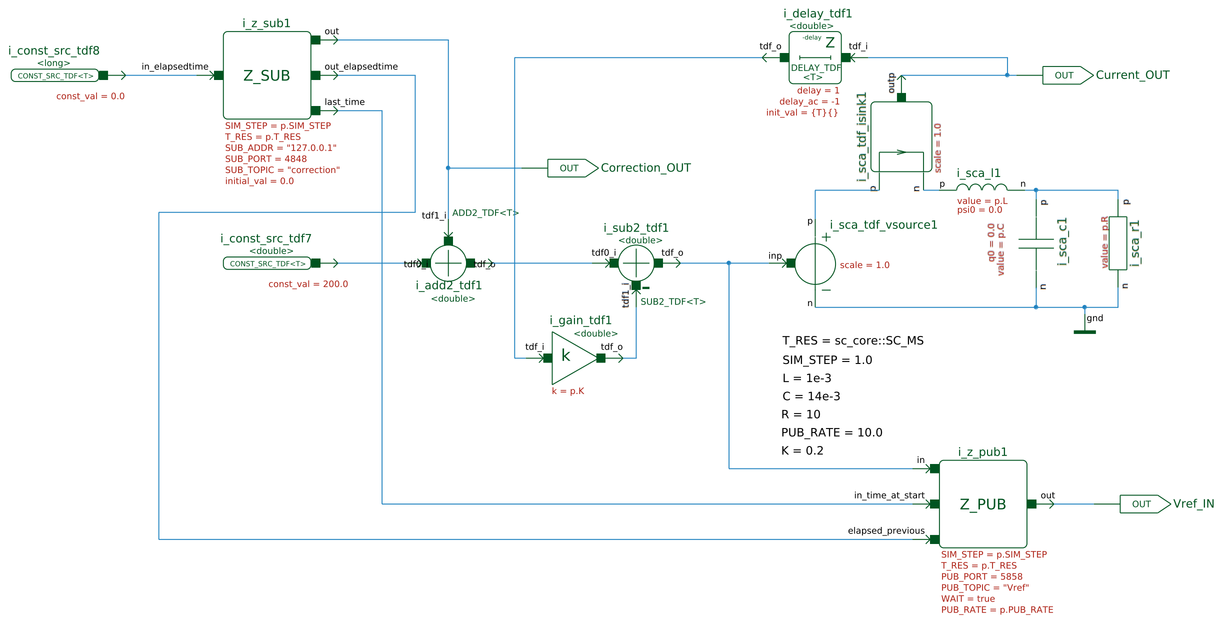


Figure 4: SystemC-AMS schematic of DC grid with primary controller along with ZeroMQ communication interfaces. Data is fetched from the secondary controller `Z_SUB` relay block, and the timestamp of fetching the data is logged, which is then propagated to the publishing relay block `Z_PUB`. The publishing relay block uses the captured timestamp to decide how long to wait to induce real-time behavior. Additionally, we have a delay block to break the algebraic loop.

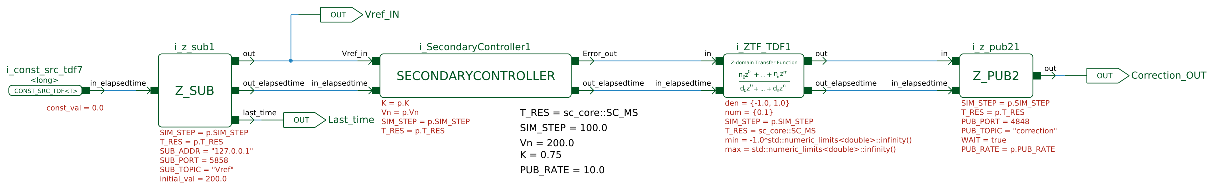


Figure 5: SystemC-AMS schematic of the secondary controller. The block following the relay block `Z_SUB` calculates the difference between the nominal voltage and the reference voltage, which is then multiplied by the secondary constant k_s. It is then integrated using a TDF block that implements integration as a z-transform. The publisher relay block `Z_PUB2` receives the required correction from the integrator block, along with the cumulative time elapsed for the given simulation step, and adds wait time to the scheduler to induce real-time behavior.

equation:

$$V_n'(t) = \delta(t) + V_n$$
$$V_{\text{ref}}(t) = V_n'(t) - K \cdot i(t)$$
$$\delta(t) = \int k_s(V_n - V_{\text{ref}}(t))dt$$

where $\delta(t)$ is the correction term for the nominal voltage and k_s is the secondary control gain. Secondary controllers are generally executed at a frequency of the order of $1-100$ Hz (Tu et al. 2020) which is suitable for real-time execution with SystemC-AMS.

4.1 Reference Implementation for Comparison

To validate our method, we implemented the plant model along with the secondary controller in PLECS (Asadi and Eguchi 2019). PLECS is an AFAP simulator with no communication component involved, which is ideal for studying ideal scenarios. An equivalent circuit of a DC microgrid along with the primary controller is shown in Figure 6. The PLECS implementation contains an algebraic loop when the DC microgrid is implemented along with the primary controller. To break the algebraic loop, we use a pulse delay block as shown in Figure 6, which models the measurement and control delay that exist in real converters. Furthermore, as we are interested in real-time simulation, we use discrete-time solvers for the simulation. The simulation time step is 1 ms. As the secondary controller is executed at 10 Hz, the sample time of the discrete integrator is 100 ms. The result of the PLECS simulation ran for 60 s is shown in Figure 7a. We display the values of signals up to 0.1 s on a linear scale while the remaining portion of the signal is on a logarithmic scale to zoom in on the initial dynamics of the system.

4.2 Implementation details for Real-time Simulation

We use a modular, model-based design approach to create a real-time simulation of a DC microgrid using SystemC-AMS and the COSIDE tool. We implement a DC grid plant using the ELN MoC from SystemC-AMS, with the reference voltage as input and the current through the inductor as output, using ZeroMQ subscriber and publisher, respectively. In our SystemC-AMS implementation, we simulate the primary controller and DC grid together in a plant while another program is used to simulate the secondary controller. Figure 4 shows a schematic of the plant. Similar to the PLECS implementation, we added a unit-delay block to model the measurement and control delay while breaking the algebraic loop in the model. We conducted a simulation of the plant at a time-step of 1 ms. The secondary controller is implemented in two steps, with calculation $k_s(V_n - V_{\text{ref}}(t))$ in one TDF module, followed by its integration using a Z-transform implemented in the subsequent TDF module. Figure 5 shows a schematic of the secondary controller.

The simulation of the plant was conducted at a time step of 1 ms, while the secondary controller was executed at a time step of 100 ms. The plant uses the previous value of the error signal until a new value is received by the plant from the secondary controller. From the simulation traces of the plant, we see the secondary controller regulate the voltage to 200 V as shown in Figure 7a. Since the secondary controller is executed at a slower rate than the plant, we observe that the correction term changes less gradually than the rest of the signal. In this case, the plant simulation retains the previous correction value until it receives the new value from the secondary controller. This behavior is consistent with real-world implementations where the secondary controller operates at a lower frequency. Additionally, the voltage and current resemble the ones obtained from the PLECS simulation. We calculated the root mean square (RMS) error of the signals to compare the SystemC simulation with the PLECS simulation. The RMS error between two timeseries signals is given by:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(V_{\text{ref}_i} - V_{\text{test}_i})^2}$$

where V_{ref} and V_{test} are the reference and test signals, respectively, and n is the total number of samples in the signal.

The RMS error for the current measured between these two simulations was 0.163 A, while the RMS error for the reference voltage was 0.051 V, and the RMS error for the correction voltage applied was 0.04 V. For further comparative analysis, we looked at the settling time of the reference voltage with a settling threshold of 99% and tolerance of 0.1%. The settling time of the reference voltage in the case of SystemC-AMS simulations varied from 0.88 s to 1.084 s for all ten instances of the simulation reported in this paper, while in the case of the PLECS simulation, it was 0.9 s.

4.3 Real-time Simulation: Delayed Start of the Controller

While the results of the SystemC-AMS implementation closely resemble those of the PLECS implementation, the advantage of the real-time SystemC-AMS implementation is useful in situations when the plant will execute initially in an open-loop manner and later receives information from the controller. In such a case, the controller is expected to stabilize the plant if designed properly. Our real-time SystemC-AMS implementation enables real-time simulation, where plant simulation can be conducted standalone, and the controller can interact with the plant in the future.

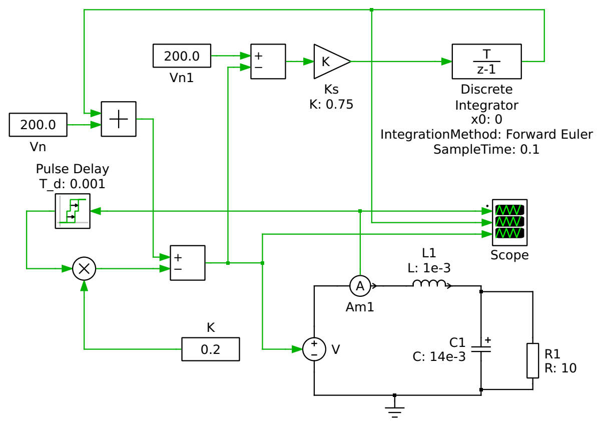
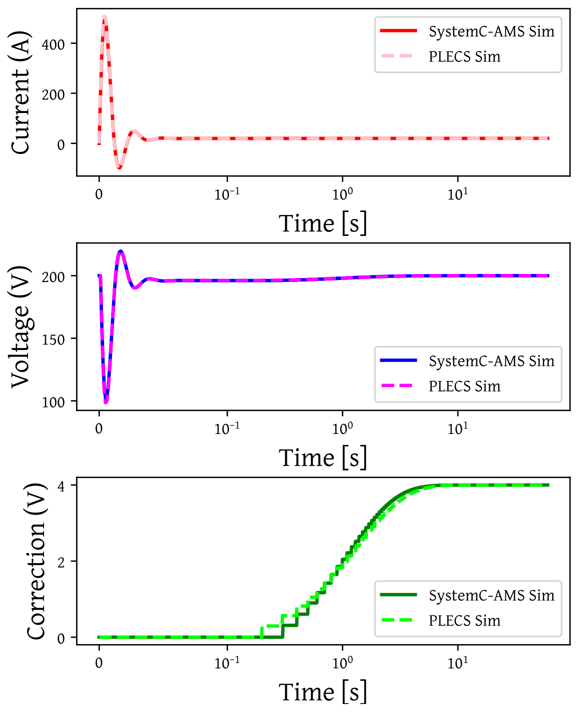


Figure 6: PLECS implementation of a simplified microgrid model along with primary and secondary controller for regulating its voltage.

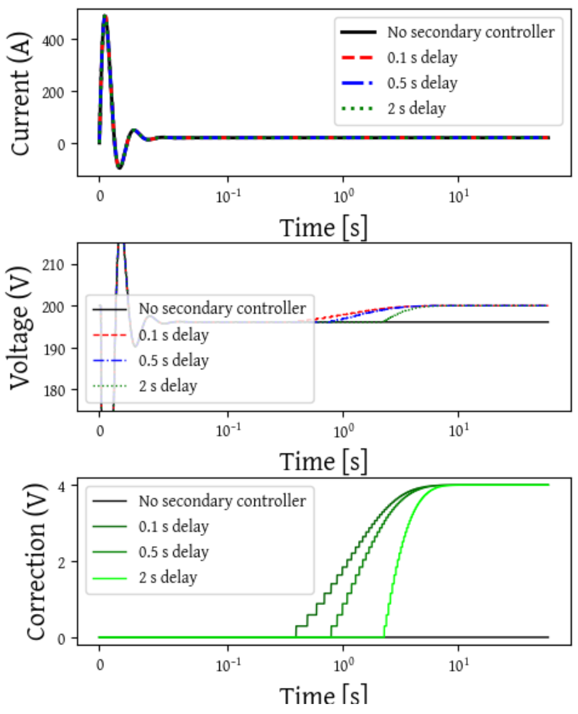
For the examples presented earlier, in the absence of the secondary controller, the reference voltage doesn't reach the specified nominal voltage of 200 V. At the same time, when the secondary controller is executed with a delayed start, settling time is achieved later. With a delayed start of 0.1 s, the settling time for the reference voltage is 0.986 s, while with a delayed start of 0.5 s, the settling time is 1.383 s, and with the delayed start of 2.0s, the settling time comes out to be 2.869 s. The graph in Figure7b displays the logged behavior. The secondary controller does not impact the current through the inductor.

5 EVALUATION OF REAL-TIME PERFORMANCE

We assessed the simulation for determinism or repeatability, timeliness, and accuracy for real-time behavior. To assess the determinism of the simulation, we executed the simulation ten times and then calculated the root mean square error of signals such as voltage measured. An RMS value closer to zero indicates that the two simulations are similar. In the case of the DC microgrid simulation, we measured the root-mean-square error of voltage and current from every possible pair of simulations and plotted it as a heatmap. The heatmap showing the RMS error of voltage and current is shown in Figure 8a and Figure 8b.

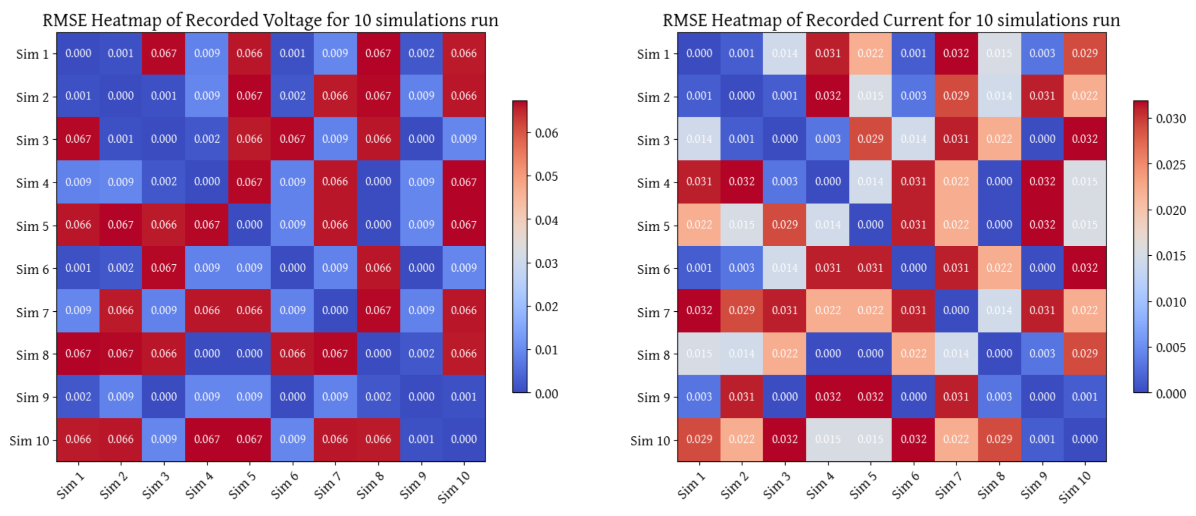


(a)



(b)

Figure 7: **Left:** Traces of current, voltage, and error from simulation of DC microgrid with the primary controller as a plant and the secondary controller to stabilize the voltage. To simulate the real-world scenario, the secondary controller publishes the correction term at a lower frequency than the plant simulation. Since the plant simulation operates at a higher frequency than the secondary controller, it continues to use the previous correction term until it receives a new correction value from the subscriber relay block. This results in a discrete step change in the correction plot. **Right:** Traces of current, voltage, and error from simulation of DC microgrid with the primary controller as a plant in case of SystemC-AMS simulation. Various cases of the secondary controller execution are presented, with the first being a total absence of it. We only show a portion of the voltage plot when the voltage starts stabilizing for each of the cases of 0.1 s, 0.5 s, and 2.0 s delay. The use of the secondary controller doesn't impact the current passing through the inductor.



(a) (b)

Figure 8: **Left:** Heatmap of the reference voltage RMS error for all pairs of simulation where secondary controller regulates the voltage of DC microgrid in conjugation with the droop controller. **Right:** Heatmap of the current RMS error for all pairs of simulation where secondary controller regulates the voltage of DC microgrid in conjugation with the droop controller.

5.1 Timeliness

For the assessment of real-time simulation, we are interested in the publication of messages in a timely fashion by the ZeroMQ node. To assess the timelines of publication, we log the wall-clock time stamp of reference voltage messages at the time they are published by ZeroMQ and calculate the time difference of successive time stamps for timeliness. A histogram of the time difference between each successive publication is shown in Figure 9. As the time-step of the simulation of the DC microgrid plant was set at 1 ms, for an ideal real-time system, the mean time difference is expected to be at 0.001 s, and the standard deviation should be 0. From Figure 9, we see that the median is 1.01259 ms, the mean is 1.01728 ms, and the standard deviation is 0.08753 ms.

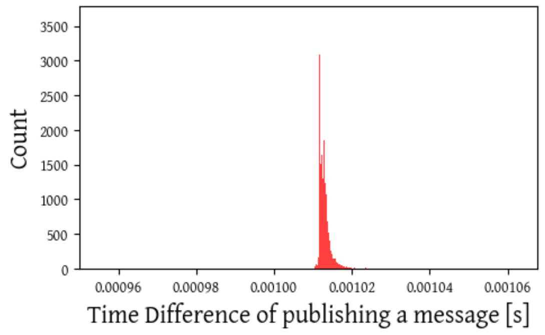


Figure 9: Histogram of time differences of publication time-stamp of timeliness of publication of messages.

6 CONCLUSION

In this paper, we present a real-time simulation method for modeling and simulating microgrids and their control using SystemC-AMS. The simulation incorporates a communication component to enable real-time simulation. We use the ZeroMQ C++ library to exchange messages between the plant simulation and the controller simulation. The SystemC-AMS real-time simulation strategy can be used as a digital twin for microgrids that can be used with hardware prototypes in hardware-in-the-loop experiments to refine control algorithms. In our future work, we plan to demonstrate the capabilities of real-time simulation using SystemC-AMS in conjunction with hardware components to regulate grid signals, such as voltage and/or current, under various conditions.

ACKNOWLEDGEMENT

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001580. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

REFERENCES

Alekhin, V. 2020. “Designing Electronic Systems Using SystemC and SystemC–AMS”. *Russian Technological Journal* 8(4):79–95.

Arthur, S., C. Emde, and N. Mc Guire. 2007. “Assessment of the Realtime Preemption Patches (RT-Preempt) and their Impact on the General Purpose Performance of the System”. In *Proceedings of the 9th Real-Time Linux Workshop*.

Asadi, F., and K. Eguchi. 2019. *Simulation of Power Electronics Converters Using PLECS®*. Academic Press.

Bian, D., M. Kuzlu, M. Pipattanasomporn, S. Rahman, and Y. Wu. 2015. “Real-time Co-simulation Platform using OPAL-RT and OPNET for Analyzing Smart Grid Performance”. In *2015 IEEE Power & Energy Society General Meeting*, 1–5. IEEE.

C++ References 2022. “Chrono C++ API”.

Chassin, D. P., K. Schneider, and C. Gerkensmeyer. 2008. “GridLAB-D: An Open-source Power Systems Modeling and Simulation Environment”. In *2008 IEEE/PES Transmission and Distribution Conference and Exposition*, 1–5. IEEE.

Chen, Y., D. Baek, J. Kim, S. Di Cataldo, N. Chang, E. Macii, S. Vinco, and M. Poncino. 2019. “A Systemc-AMS Framework for the Design and Simulation of Energy Management in Electric Vehicles”. *IEEE Access* 7:25779–25791.

Chen, Y., S. Vinco, E. Macii, and M. Poncino. 2016. “Fast Thermal Simulation using SystemC-AMS”. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, 427–432.

Einwich, K., T. Hartung, and T. Arndt. 2022. “COSIDE System Level Design Environment”. *COSEDA Technologies GmbH*.

Fraccaroli, E., and S. Vinco. 2022. “Modeling Cyber-Physical Production Systems with SystemC-AMS”. *IEEE Transactions on Computers* 72(7):2039–2051.

Guerrero, J. M., J. C. Vasquez, J. Matas, L. G. de Vicuna, and M. Castilla. 2011. “Hierarchical Control of Droop-Controlled AC and DC Microgrids—A General Approach Toward Standardization”. *IEEE Transactions on Industrial Electronics* 58(1):158–172.

Hintjens, P. 2013. *ZeroMQ: Messaging for Many Applications*. O’Reilly Media, Inc.

Montenegro, D., M. Hernandez, and G. Ramos. 2012. “Real Time OpenDSS Framework for Distribution Systems Simulation and Analysis”. In *2012 Sixth IEEE/PES Transmission and Distribution: Latin America Conference and Exposition (T&D-LA)*, 1–5. IEEE.

P1666.1 - SystemC AMS Extensions Working Group 2016. “IEEE Standard for Standard SystemC(R) Analog/Mixed-Signal Extensions Language Reference Manual”. *IEEE Standards Association* P1666.1.

Panda, P. R. 2001. “SystemC: A Modeling Platform Supporting Multiple Design Abstractions”. In *Proceedings of the 14th International Symposium on Systems Synthesis*, 75–80.

Rashid, M. H. 2017. *Spice for Power Electronics and Electric Power*. CRC press.

Reghenzani, F., G. Massari, and W. Fornaciari. 2019. "The Real-time Linux Kernel: A Survey on PREEMPT_RT". *ACM Computing Surveys (CSUR)* 52(1):1–36.

Swan, S. 2001. "An Introduction to System Level Modeling in SystemC 2.0". *Cadence Design Systems, Inc., draft report*.

Tu, H., Y. Du, H. Yu, A. Dubey, S. Lukic, and G. Karsai. 2020. "Resilient Information Architecture Platform for the Smart Grid: A Novel Open-Source Platform for Microgrid Control". *IEEE Transactions on Industrial Electronics* 67(11):9393–9404.

Tu, H., H. Yu, and S. Lukic. 2023. "Impact of Virtual Inertia on DC Grid Stability with Constant Power Loads". *IEEE Transactions on Power Electronics* 38(5):5693–5699.

Vachoux, A., C. Grimm, and K. Einwich. 2004. "Towards Analog and Mixed-signal SOC Design with SystemC-AMS". In *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*, 97–102. IEEE.

Vernay, B., A. Krust, G. Schröpfer, F. Pêcheux, and M.-M. Louerat. 2015. "SystemC-AMS Simulation of a Biaxial Accelerometer Based on MEMS Model Order Reduction". In *2015 Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, 1–6. IEEE.

AUTHOR BIOGRAPHIES

RAHUL BHADANI is a post-doctoral research fellow at Institute for Software Integrated Systems, Vanderbilt University. He obtained his BE in Information Technology from Bengal Engineering & Science University, Shibpur, India in the Spring of 2012, and his MS, and Ph.D. in Electrical and Computer Engineering from the University of Arizona in the Spring of 2017 and 2022 respectively. His research interests include model-based design, and real-time modeling and simulation of cyber-physical systems such as microgrids and connected & autonomous vehicles. His email address is rahul.bhadani@vanderbilt.edu.

HAO TU received his bachelor's, master's, and Ph.D. degrees from Xi'an Jiaotong University, China in 2012, RWTH Aachen University, Germany in 2015, and North Carolina State University, USA in 2020, respectively, all in electrical engineering. He is currently a Research Assistant Professor at the National Science Foundation Future Renewable Electric Energy Delivery and Management (FREEDM) Systems Engineering Research Center, headquartered at North Carolina State University. His research interests include control of power electronics converters, energy storage systems, microgrids, and machine learning. He is the recipient of the Second Prize Best Paper Award in IEEE Transactions on Transportation Electrification 2021. His email address is htu@ncsu.edu.

SRDJAN LUKIC received the Ph.D. degree in electrical engineering from Illinois Institute of Technology, Chicago, IL, USA, in 2008, and is currently a Professor and University Faculty Scholar with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA. He serves as the Deputy Director of the National Science Foundation Future Renewable Electric Energy Delivery and Management (FREEDM) Systems Engineering Research Center, headquartered at North Carolina State University. His current research interests include the design, and control of power electronic converters and electromagnetic energy conversion with application to microgrids, wireless power transfer, energy storage systems, and electric automotive systems. He was a Distinguished Lecturer with the IEEE Vehicular Technology Society from 2011 to 2015. His email address is smlukic@ncsu.edu.

GABOR KARSAI is a professor of electrical and computer engineering and computer science at Vanderbilt University and senior research scientist at ISIS. His research interests include model-integrated computing (MIC), design automation for model-driven development processes, automatic program synthesis, and the application of MIC in various government and industrial projects. Karsai received a Ph.D. in electrical engineering from Vanderbilt University. He's a senior member of the IEEE Computer Society. His email address is gabor.karsai@vanderbilt.edu.