BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 69(3), 2021, Article number: e137057
DOI: 10.24425/bpasts.2021.137057

IEC 61850 interface for real time power system simulation

Karol KUREK*, Łukasz NOGAL, Ryszard KOWALIK, and Marcin JANUSZEWSKI

Faculty of Electrical Engineering, Warsaw University of Technology, Pl. Politechniki 1, 00-661 Warszawa, Poland

Abstract. Real time simulators of IEC 61850 compliant protection devices can be implemented without their analogue part, reducing costs and increasing versatility. Implementation of Sampled Values (SV) and GOOSE interfaces to Matlab/Simulink allows for interaction with protection relays in closed loop during power system simulation. Properly configured and synchronized Linux system with Real Time (RT) patch, can be used as a low latency run time environment for Matlab/Simulink generated model. The number of overruns during model execution using proposed SV and GOOSE interfaces with 50 µs step size is minimal. The paper discusses the implementation details and time synchronization methods of IEC 61850 real time simulator implemented in Matlab/Simulink that is built on top of run time environment shown in authors preliminary works and is the further development of them. Correct operation of the proposed solution is evaluated during the hardware-in-the-loop testing of ABB REL670 relay.

Key words: implementation in real time; power system; protection relays testing; computer simulation; IEC 61850.

1. Introduction

Digital Real Time Simulation (DRTS) of a power system aims to model its behavior during transient states and corresponding current and voltage waveforms in time domain from specific nodes [1]. The simulation is done mainly in software by solving the equations of modeled power system. When part of the simulated system is replaced by real hardware, the Hardware in the Loop (HIL) simulation is done. The inclusion of additional hardware under test is done by pulling measurement signals from the simulation to the device. Binary signals introduced from the device to the online simulation, allows the tested device to affect simulation state.

Real time simulators have been evolving since 1991, when first commercial solution was introduced. Nowadays hardware architecture used varies from specialized extension boards formed into racks, capable of parallel processing to nodes formed on the basis of multipurpose Intel CPUs [1]. Simulation based on co-processing in FPGA based circuits, enables very small simulation steps, which is especially vital while simulating power electronics [2]. FPGA can be used to build the whole simulator or only the latency critical part of the simulation [3], which allows to achieve simulation steps as low as 5 µs. The lowest single simulation step during power system simulation, was noted in [4] by the simulator based solely on FPGA circuits – 24 ns. Another successful implementation in literature involves Raspberry PI platform [5] being an affordable alternative.

One of the most common simulation software is Matlab/Simulink Environment, that is used in one of the commercially available real time simulators. Other alternatives are Powerfactory and Labview based implementation [6] and proprietary RSCAD. Real time simulators are mostly run on Windows or Linux operating systems with some implementations using parts based on proprietary solutions.

Traditionally real time simulators use analog measurements and binary signals to interface hardware under test. With the advancements in the field of digital substations, IEC 61850 interfaces used to generate SV data stream and GOOSE datagrams are also becoming more common [7]. Commercially available simulators have successful, closed source implementations of such interfaces, evaluated in [8] and [9]. Such implementation can be part of a simulation or be exported to the external circuits dedicated to this task [8]. FPGA hardware seems to be ideal for the task of IEC 61850 traffic processing, because of minimum latency on both input and output. Such design is introduced in [10] and allows the generation of up to 16 concurrent SV streams.

2. Related work

Implementations of Matlab/Simulink IEC 61850 interfaces available commercially are closed software with no reports in the literature. Implementation of SV interfaces to Matlab/Simulink known from literature, is presented [11, 12]. The implementations lack GOOSE implementation and time synchronization in simulation software that allows closed loop testing.

Matlab/Simulink is one of the most common simulation environments with the capability to execute its models in real time, therefore it was chosen to implement the interface to SV IEC 61850–9-2 LE compliant stream as a s-function block. To allow the testing of protection in closed loop, GOOSE interface is implemented as well. Implementation is done in C libiec61850 open-source library in Linux-RT environment. Simulation program is compiled used Simulink-Coder and modified ert-linux target [13]. Time synchronization technique of

*e-mail: karol.kurek@ien.pw.edu.pl

Manuscript submitted 2020-11-04, revised 2021-01-20, initially accepted for publication 2021-02-22, published in June 2021

real time simulation, based on operating system IEEE 1588 v2 Precision Time Protocol (PTP) implementation is proposed. The resulting IEC 61850 interface is used in the simulation of high voltage power line that allows for hardware-in-the-loop testing of REL 670 over current protection.

3. IEC 61850 Interface implementation

To allow two-way interaction between simulated power system and hardware under test, measurements in the form of SV measurement stream and binary signals in the form of GOOSE messages need to be introduced into Matlab/Simulink environment. The mechanisms that allow such integration are Matlab s-functions. S-functions are the programming interface that allows to introduce user written functions into Simulink environment in the form of custom blocks. User written code is compiled using Matlab mex compiler and can be then imported into the simulation as a new function block, that has inputs, outputs and internal configuration parameters.

Being a programming interface means that Simulink delivers a set of functions that are dedicated to performing specific tasks at specific moments during execution of s-function block that is imported into the simulation model. Simulation process of a s-function broken down to corresponding C interface functions is shown in Fig. 1.

The first stage of the execution of s-function is the initialization in mdlInitializeSizes and mdlInitializSampleTimes C

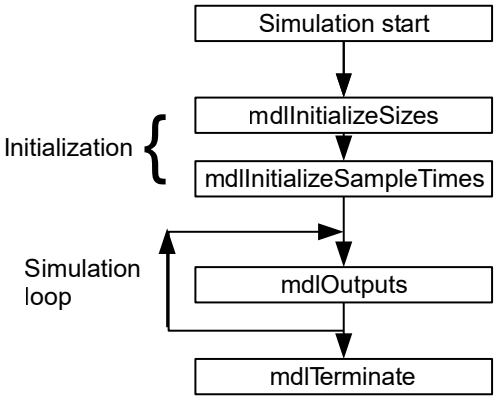

Fig. 1. Simulation process of s-function block

functions. Mentioned functions are executed once at the beginning of a simulation. During this stage, function block is added to the simulation model. Input and output signal count, sampling frequency used in a model, block configuration parameters, execution order and memory size for additional variables are determined.

After the initialization stage, the model is executed at the defined simulation step. When S-functions are processed, its C function mdlOutputs is executed at every simulation step. The function allows to interact with the rest of the model by its C language input arguments and return value. Data handed over

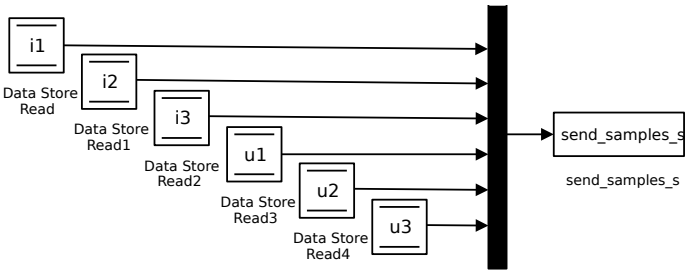

Fig. 2. SV generation simulink block

to these functions and its return values at C code level are seen as input and output signals inside the Simulink model.

SV generation block implemented as a s-function is presented in Fig. 2. The purpose of this block is to generate SV sample stream with measurement data from selected power system point and send it to protection relay under test. Block has 6 signals connected to its input port – 3 phase currents and voltages that are pulled from simulated power system model from chosen measuring points. Those signals represent the samples meant to be sent in an SV frame. Before being connected to input port, sample values have to be conditioned to match data format specified in IEC 61850-9-2 LE – measurements are sent as primary currents and voltages expressed as 1mA and 10 mV respectively.

SV generation block allows the user to set basic parameters for SV publisher in the form of block configuration parameters, processed in a mdlInitializeSizes function. These configuration parameters are network interface name, SVID, destination multicast address, AppID, Vlan ID and Vlan Priority. At every simulation step, during the execution of the corresponding mdlOutputs function, the block gathers samples from selected measuring points. Using these samples and block configuration parameters, SV frame is formed and sent through Ethernet interface.

In this approach, a single SV frame is sent during the execution of SV generation block mdlOutputs function. This forces the simulation step to be intact with SV stream frequency, defined in IEC 61850–9-2LE implementation guidelines – 80 samples per period. Therefore, when using simulation steps that are different than expected SV generation frequency (e.g., 4 kHz for 50 Hz power system) rate transition block from Simulink standard library has to be introduced.

GOOSE subscriber implemented as a s-function block is shown in Fig. 3. Main purpose of the block is to receive data from protection relay – trip signals, that are then connected to circuit breaker inside the simulated power system model. The block gets the data from received GOOSE messages network and sets the corresponding state of its output signals. Goose block Output signals are interpreted as phase trip commands of IED under test. Internal configuration parameters of GOOSE subscriber function block are: AppID, network interface name, source multicast address, dataset, GoCB, host IP and port, variable count, variable offset. These parameters are used during the initialization phase inside mdlInitializeSizes to properly set up and start the GOOSE receiver thread.

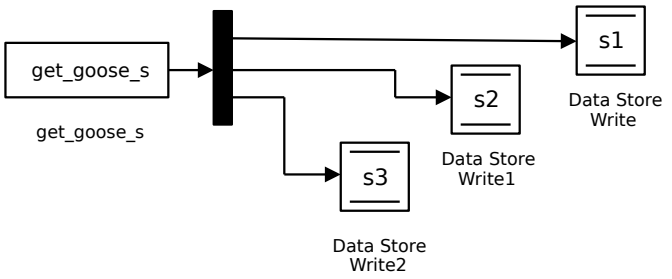

Fig. 3. GOOSE input simulink block

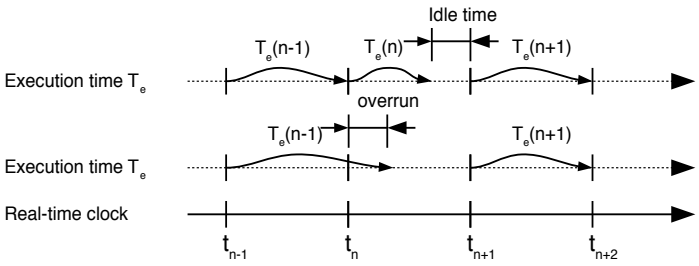

Fig. 5. Real time simulation cases

The approach to process received GOOSE messages is different than in the case of SV generation block. A GOOSE receiver thread is started in parallel to the simulation, whose task is to continuously listen for GOOSE messages on network interface. Therefore, GOOSE reception and processing is independent of simulation step. However, changes of variables received in GOOSE messages are rewritten to GOOSE subscriber function block output signals, once every simulation step inside corresponding mdlOutputs function.

Trip signals received from GOOSE function block, can be used to control the circuit breaker (CB) inside of a simulation model and therefore change the simulation state during its execution – this enables hardware-in-the-loop simulation. CB logic connected directly to variables read from GOOSE datagram is presented in Fig. 4 and allows to latch the CB opening signal after the change of controlling GOOSE variables to zero.

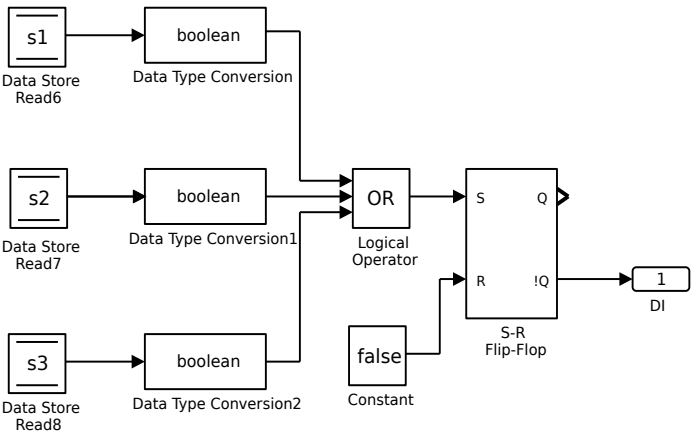

Fig. 4. Circuit breaker logic controlled by GOOSE variables

4. Real time operation

Carrying out the simulation in real time, means the discrete solving of model equations should happen at definite time on par with real world clock. When solver does finish before the deadline being a consequence of sampling frequency, it goes idle and waits for the next simulation step. The time constraint is met. When all the work including solving equations and setting hardware, exceeds the simulation step, the overrun happens. Single overruns are acceptable, but if they happen often the simulation is considered inaccurate and non real time. Both described situations of real and non-real time simulations are presented in Fig. 5.

To run power system model in real time, Simulink Coder is used. It is used to compile the model to single executable binary that can be executed in real time in many operating systems defined as Simulink Coder targets. Default Simulink Coder target dedicated for Linux is Generic Real-Time Target. Unfortunately, the target generated code is unable to run in real time on modern Linux kernels and in turn results in code executing at full speed. Michal Sojka wrote and published ert-linux, alternative target for Simulink Coder [13] that works well for recent Linux kernels.

Solution of IEC 61850 compliant real time tester proposed in this paper is based on runtime environment that was a subject of authors preliminary works shown in the paper [14]. Power system model containing SV and GOOSE blocks has to be run in real time with the smallest possible latency to keep the number of overruns low.

Linux operating system that is used to run Simulink Coder prepared executable, has to be patched with PREEMPT_RT patch and further adjusted to achieve the lowest system latency possible. Linux kernel without latency optimalizations may not be capable to run real time power system simulations with simulation step of 50 µs. The sources of latency and corresponding mitigation techniques are described and tested in detail in [14]. The following list summarizes the methods of latency optimalizations in Linux operating system, that can be used to achieve the CPU scheduler microsecond worst case latency of 4 µs which is sufficient for most real time power system simulation cases.

Logical CPUs
The CPU feature that allows to address two logical CPUs on one physical core should be disabled, because of introduced additional latency.

Frequency scaling
Frequency scaling allows for CPU power conservation at the expense of time-consuming switching and therefore should be disabled at hardware level, system wide or in user space.

Real time throttling
By default, Linux real time processes utilize 95% of CPU time to avoid system locking. This behavior can be changed to the

exclusive use of the CPU by writing "–1" value to /proc/sys/kernel/sched_rt_runtime_usfile lo-cated in procfs.

Core isolation
Single core should be designated to run Simulink Coder executable. Selected core should be isolated from: new user space processes scheduled by the system, hardware interrupt services, kernel processes.

Processor sleep states
Modern x86 processors implements C states which allows the CPU to conserve energy during idle states. Switching between C states introduces latency and should be disabled at hardware level or system wide (processor.max_cstate=0 can be written to kernel command line at boot time).

5. Time synchronization

Testing IEC 61850 protection systems, synchronization to external clock signal is mandatory. Precision Time Protocol (PTPv2) defined in IEEE 1588 and adopted by IEC 61850 standard allows the time synchronization through common Ethernet link. There are two software packages available for PTP support in Linux: ptpd and linuxptp. Coupled with Network Interface Controller (NIC) supporting hardware time stamping, the latter allows for sub microsecond time synchronization. Tests carried out in [14] on two mutually synchronized Linux systems, shows synchronization accuracy lower than 100 ns when using PTP mapping directly to network layer 2 (Power profile). This allows to synchronize the simulator with protocol commonly used in IEC 61850 protection systems with accuracy conforming to requirements defined in T5 class of IEC 61850 part 5.

Linuxptp software synchronizes NIC and system clock to external master clock. To achieve synchronous sampling of Simulink model executing in real time, additional modification has to be applied to Simulink coder target. The target describes how model should be executed on the target platform. Code of ert-linux target, can be inspected and modified to use system clock being the subject of linuxptp time adjustments due to openness of the code.

Nanosleep function family, being a Linux Posix Timer interface offers two main methods of time keeping in user space: monotonic (CLOCK_MONOTONIC) and real time (CLOCK_REALTIME). First one measures time from some unspecified point (usually boot time of the system) and the second offers the timestamp describing current date and time. Monotonic method is a right choice when measuring relative time in a single application. Real time method uses absolute values of time and reflects time adjustments from Linux time services which allows for time measurement between multiple autonomous systems. For the Posix Timer Linux implementation to offer nanosecond precision, should be used with Time Stamp Counter CPU register (TSC) as a time source.

Synchronous sampling modification for ert-linux target is based on the change from monotonic to real time method used in clock_nanosleep function and measuring the simulation time as absolute value. Modification has to be applied to tlc files included in ert-linux. Furthermore, to keep the SV stream in sync, the zero sample (smpCnt = 0) should always be generated in the start of a second. Therefore, the resulting application should always wait and start the simulation on next full second. The idea of time correction during model execution is shown in Fig. 6.

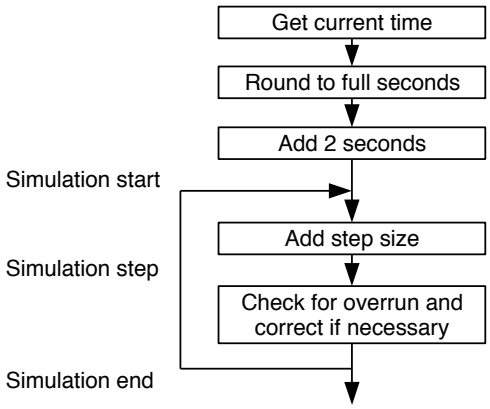

Fig. 6. Time synchronization process during the simulation

Timestamps generated from the real time method are a subject of time correction by PTP process. This ensures the simulation is always run synchronously to the external clock. The use of this property can also be used to synchronize multiple simulation nodes of bigger simulation system.

6. Evaluation

Evaluation of proposed IEC 61850 compliant real time simulator was done in two steps. Firstly, its real time performance during the execution in Linux-RT environment is evaluated and then it is used to test ABB REL670 relay in closed loop.

The number of overruns that happen executing a model of given complexity is considered a measure of real-time simulator performance. For the sake of evaluation, the model is run on following hardware/software configuration.

- AMD 3-core CPU 4 Ghz,
- Linux 4.19.59-rt24,
- High resolution timers, TSC time source,
- GCC 5.5.0.

Table 1 shows the results of overrun tests for different configurations of model and operating systems. The same tests

Table 1
Number of overruns for different system/model configurations

System configuration	Single Line		Extended model	
	50 µs	250 µs	50 µs	250 µs
Non optimized	–	6	–	15
Latency optimized	0	0	115	0

were run on stock RT kernel and latency optimized kernel (optimizations and their effects are described in [14] and briefly summarized in Chapter 4). Two power system models were tested: single line model that consists of a single line and 2 sources and extended model with 6 lines, 4 sources and 2 loads. Models were equipped with one SV and GOOSE interface. Every test was repeated with step sizes 50 and 250 µs. Single test lasted for 5 minutes.

Non optimized system was unable to finish the tests with step size 50 µs due too many overruns. Latency optimized system allows the execution of all test configurations. Overruns are occasional and happen only in case of extended model and time step size 50 µs. Introduction of additional SV interfaces, that created up to 16 streams, resulted in no impact on measured number of overruns. The results suggest, IEC 61850 interface can be successfully used in the simulation of more complex power systems and to generate concurrent SV streams from multiple simulation nodes while still maintaining low number of overruns.

Final test involves testing of ABB REL670 over current function in closed loop, using the proposed simulator architecture. Laboratory setup is shown in Fig. 7.

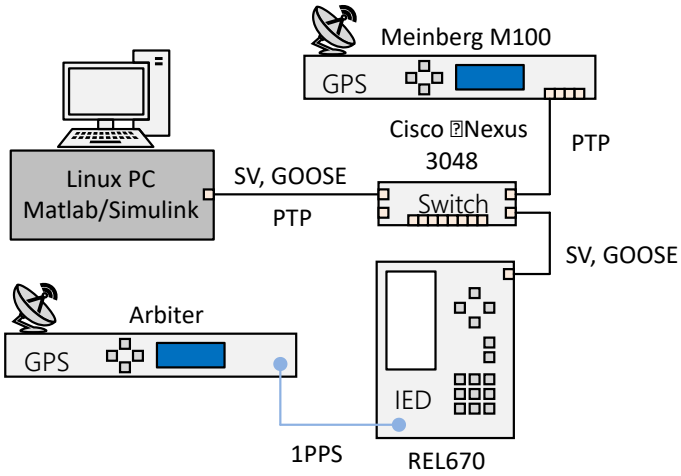

Fig. 7. Laboratory setup during ABB REL670 testing

Process data including SV, GOOSE and PTP protocol are sent through Ethernet network using Cisco Nexus switch acting as a PTP boundary clock. Meinberg M1000 is acting as PTP master clock. Simulation model is run on Linux PC synchronized to the Cisco switch. REL670 relay because of hardware limitations, was synchronized to different GPS clock 1PPS signal. REL670 implements over current function: IEC Definite time characteristic, threshold at 180% of nominal current, zero-time delay. Separate trip signals from 3 phases were sent in GOOSE message back to the simulator. Relay was tested during phase A-B short circuit in the middle of the line. Figure 8 shows the phase A current waveform during the fault in free run.

After the configuration of ABB670 relay in terms of over-current protection, SV and GOOSE configuration it was

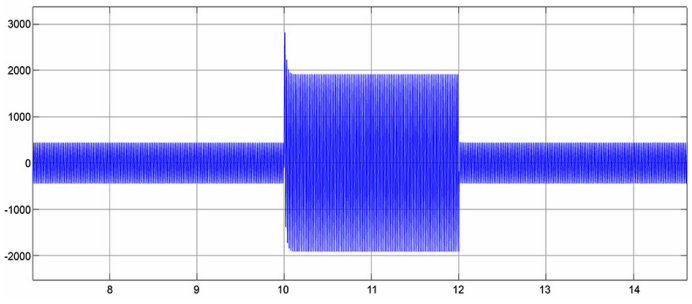

Fig. 8. Current in phase A during the free run of simulation

connected to the common process network. PCM 600 software was used to adjust the IEC 61850 settings to the ones of implemented function blocks. SV function block delivered SV data to the protection device, which was confirmed by reading device measurements. After the actual start of simulation, over current function tripped which generated GOOSE message. Simulators GOOSE function block extracted variable values of corresponding trip signals and assigned them to the output port of the block. Tripping of over-current resulted in the change of state in the simulated power system and resulted in the clearance of simulated phase to phase fault. During the simulated fault, disturbance recording was triggered on the relay side (Fig. 9). Analysis of presented signals shows proper operation of not only over-current function but also time synchronization. Calculated tripping time of the relay was 11.6 ms, which is a reasonable value for this device. Loss of synchronization signal at any end, results in random drift between recorded measurement signals and change of binary states.

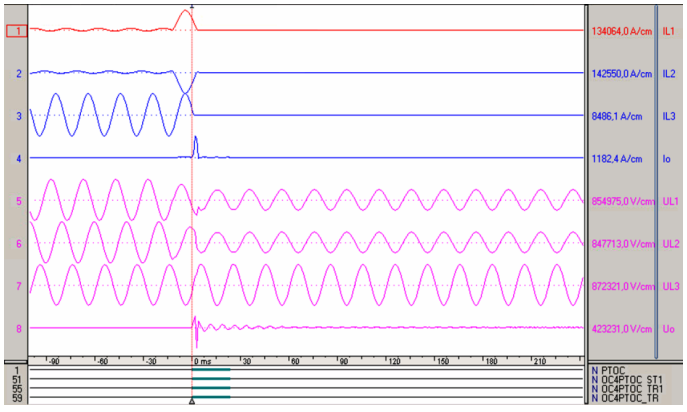

Fig. 9. Disturbance recording obtained during the testing of ABB REL670 over current function

7. Conclusions

Protection relays using SV and GOOSE messages can be evaluated using real time simulators implemented mainly in software. This allows to build real time simulators on the basis of Matlab/Simulink software and Linux-RT. The limitation of such tester is the complexity of simulated power system relative

to processing power of the CPU. Properly time synchronized Matlab/Simulink power system simulation extended with SV and GOOSE blocks can be successfully used to evaluate protection relays in closed loop. Linux RT environment offers acceptable levels of latency and time synchronization accuracy for power system simulation purposes. In order to run more complex simulation models with lower time step it is recommended to optimize the operating system for latency by disabling logical CPUs, frequency scaling, real time throttling, processor sleep state and maintaining core isolation for the real time simulation. Simulation run on latency optimized Linux shows occasional overruns while running with time step size 50 µs and no overruns on step size 250 µs (SV generation frequency) which is sufficient for IEC 61850-9-2 LE stream generation. GOOSE interface allows for hardware-in-the-loop testing – change of simulation state on the basis of GOOSE variables. Tested system behaved similarly regardless of number of SV nodes present in the model – inclusion of 16 SV streams interfaces, resulted in no impact on performance. Linux PTP system service can be used to constantly adjusts the simulation clock to external sources which allows for proper time synchronization when testing protection systems and synchronization of different simulation nodes.

The proposed solution can be used to extend Matlab/Simulink power system models with real world interfaces to IEC 61850 compliant protection devices using open source components. This can be achieved using regular PC hardware with minimal costs and almost no wiring. The downside of the solution is its limitation to protection schemes using digital process data.

References

[1] M.D.O. Faruque *et al.*, "Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis," *IEEE Power Energy Technol. Syst. J.* 2(2), 63–73 (2015).

[2] S. Piróg, R. Stala, and Ł. Stawiarski, "Power electronic converter for photovoltaic systems with the use of FPGA-based real-time modeling of single phase grid-connected systems," *Bull. Pol. Acad. Sci. Tech. Sci.* 57(4), 345–354 (2009).

[3] C. Yang, Y. Xue, X. Zhang, Y. Zhang, and Y. Chen, "Real-Time FPGA-RTDS Co-Simulator for Power Systems," *IEEE Access* 6, 44917–44926 (2018)

[4] M. Matar and R. Iravani, "The Reconfigurable-Hardware Real-Time and Faster-Than-Real-Time Simulator for the Analysis of Electromagnetic Transients in Power Systems," *IEEE Trans. Power Deliv.* 28(2), 619–627 (2013).

[5] M.E. Hernandez, G.A. Ramos, M. Lwin, P. Siratarnsophon, and S. Santoso, "Embedded Real-Time Simulation Platform for Power Distribution Systems," *IEEE Access* 6, 6243–6256 (2018).

[6] D.A.M. Montaña, D.F.C. Rodriguez, D.I.C. Rey, and G. Ramos, "Hardware and Software Integration as a Realist SCADA Environment to Test Protective Relaying Control," *IEEE Trans. Indust. Appl.* 54(2), 1208–1217 (2018).

[7] C. Dufour and J. Bélanger, "On the Use of Real-Time Simulation Technology in Smart Grid Research and Development", *IEEE Trans. Indust. Appl.* 50(6), 3963–3970 (2014).

[8] M. Shoaib and L. Vanfretti, "Performance evaluation of protection functions for IEC 61850-9-2 process bus using real-time hardware-in-the-loop simulation approach," in *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013),* 2013, pp. 1–4.

[9] M.S. Almas, R. Leelaruji, and L. Vanfretti, "Over-current relay model implementation for real time simulation amp; Hardware-in-the-Loop (HIL) validation," in *IECON 2012 – 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 4789–4796.

[10] D.R. Gurusinghe, S. Kariyawasam, and D.S. Ouellette, "Testing of IEC 61850 sampled values based digital substation automation systems," *J. Eng.* 2018(15), 807–811 (2018).

[11] Y. Wu, N. Honeth, L. Nordström, and Z. Shi, "Software MU based IED functional test platform", *2015 IEEE Power Energy Society General Meeting*, 2015, pp. 1–5.

[12] N. Honeth, Z.A. Khurram, P. Zhao, and L. Nordström, "Development of the IEC 61850-9-2 software merging unit IED test and training platform," in *2013 IEEE Grenoble Conference*, 2013, pp. 1–6.

[13] M. Sojka, "On generating Linux applications from Simulink." [Online]. Available: https://rtime.felk.cvut.cz/~sojka/blog/on-generating-linux-applications-from-simulink/

[14] K. Kurek, M. Januszewski, R. Kowalik, and Ł. Nogal, "Implementation of IEC 61850 Power Protection Tester in Linux Environment", *Bull. Pol. Acad. Sci. Tech. Sci.* 68(4), 689–696 (2020)