BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 68, No. 5, 2020
DOI: 10.24425/bpasts.2020.134169

Testing power system protections utilizing hardware-in-the-loop simulations on real-time Linux

M. KRAKOWSKI* and Ł. NOGAL

Faculty of Electrical Engineering, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland

Abstract. The complexity of power system phenomena challenges power system protection testing to obtain the required adequacy of the testing environment. Hardware-in-the-loop simulation in real-time substantially increases testing capabilities. However, there is still the question of the availability of commercial solutions. To address the challenges, a new hardware-in-the loop system has been designed and implemented utilizing the easily available Matlab/Simulink environment and Linux RT Preempt OS. The custom software part prepared for the presented system is based on the Matlab/Simulink s-function mechanism, Embedded Coder toolbox and Advantech biodaq library as the interface for the utilized I/O cards. The simulator's real-time performance limits on Linux RT Preempt have been verified, and it was shown that its performance is sufficient to conduct successful tests of protection relays. Consequently, a simple power system protection relay testing example is provided, including a discussion of results. Finally, it has been proven that the presented system can be utilized as a simpler and more accessible hardware-in-the-loop testing alternative to commercial simulators.

Key words: hardware-in-the-loop simulation, testing, real time systems, power systems protection.

1. Introduction

Testing the power system protection and control devices is not a trivial procedure. Due to the complexity of electromagnetic and electromechanical phenomena, corresponding to the various types of faults, it is very challenging to reproduce them with regular testing equipment. Considering that it is often impossible to arrange field testing on a real network, digital real-time simulation testing methods have evolved from scaled-down physical models and analog equivalents of the power system.

The theoretical basics of real-time digital simulation were formulated along with the electromagnetic transient program by Hermann W. Dommel in the late 60s [1]. But it took almost 2 more decades to design and release the first commercial solution – RTDS [2].

Since that time, a couple of other commercial and academic simulators have been released. Among those, it is worth mentioning the Opal RT and Typhoon [3, 4] solutions. The Opal RT simulator is based on x86 multi-core hardware cooperating with FPGA and RT-Linux or QNX operating system. Typhoon is purely based on FPGA hardware. There are also other academic FPGA-based solutions dedicated to high speed real-time simulations utilized mainly for power electronics applications [5, 6]. The popularization of real-time digital simulations over the years has resulted in their various applications in the design and testing of power system protection algorithms [7–15] and in the other research areas [16–18].

On the other hand, over the last couple of years, the Matlab/Simulink environment has been supporting real-time digital simulations by offering the Simulink Desktop Real-Time toolbox [19], preceded by the Real-Time Windows Target library. The toolbox is dedicated to Windows and macOS and provides a dedicated kernel that runs the real-time application. Its application for power system protection testing has already been verified in some academic solutions [20]. However, it should be noted that the dedicated co-kernel approach limits the toolbox's flexibility in adding customized components. Additionally, there is no support for GNU/Linux and other Unix-based operating systems.

Despite the presence of various real-time digital simulators on the market, most available solutions are relatively expensive, which limits their wide application in industry and academia.

The idea of designing a new real-time digital simulator was driven by 2 main factors. The first factor was to prove that it is possible to design a relatively cheap system that would be easily available, at least to the people in the academic world, but in the end, also to industry. The second factor was to verify the applicability of the Linux RT-Preempt in real-time simulations. Though some kind of RT-Linux is also utilized in the Opal-RT simulator [3], there is no detailed benchmarking of that application, so the aim was to thoroughly verify performance with different operating system setups.

The structure of the paper is organized as follows. Section 2 describes the design of the proposed simulator system. Benchmarking of the real-time simulator's performance on Linux RT-Preempt is provided in Section 3. Finally, in Section 4, a simple power system protection relay testing example is provided. including discussion of results.

*e-mail: marcin.krakowsk.dokt@pw.edu.pl

Manuscript submitted 2019-12-19, revised 2020-03-30, initially accepted for publication 2020-04-19, published in October 2020

2. Description of the proposed hardware-in-the-loop system

The proposed hardware-in-the-loop system was built on an industrial PC x86 architecture-based platform with a GNU/Linux operating system kernel with the RT-Preempt real-time patch. An overview of the system applied in the described work is depicted in Fig. 1. The simulation models used in the proposed system were prepared in the Matlab/Simulink environment with the Simscape Power Systems toolbox for modeling power system domain elements. As the native Matlab/Simulink environment is not exactly dedicated to real-time applications, the Embedded Coder toolbox was utilized to generate the executable application optimized for real-time performance from the model.

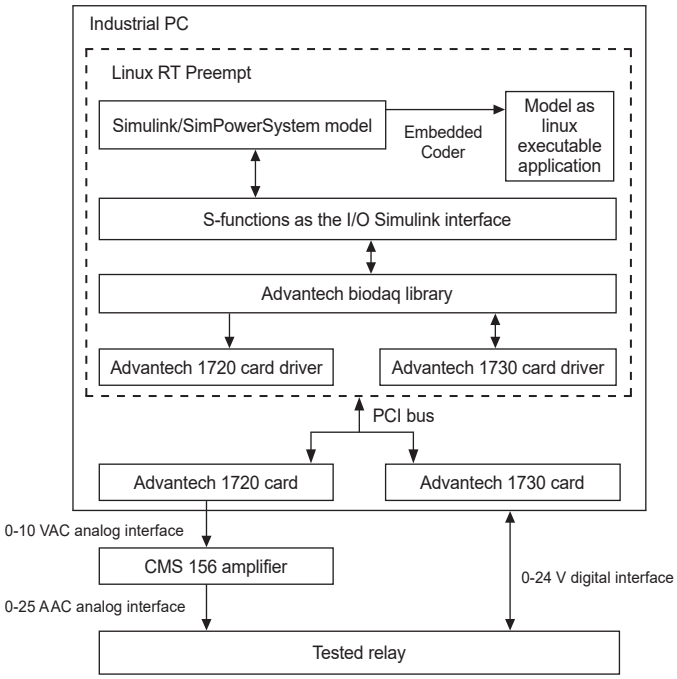

Fig. 1. Overview of the proposed hardware-in-the-loop simulation system

The system input/output interface for physical cards was implemented using the Simulink s-functions mechanism. Direct handling of the PCI 1720 and 1730 cards was done with the biodaq library provided by Advantech. The analog output PCI 1720 card has a 0–10 V output voltage range, which is not adjusted to the protection relays' signal range of current and voltage analog inputs. Therefore, in the described system, a dedicated Omicron CMS 156 amplifier was used to provide signal level adjustment.

2.1. Hardware. The hardware basis of the proposed system is an Elmatic SIGMA 4020 industrial PC. The motivation for this choice was that 5 PCI slots are installed in the PC's mainboard. The PCI standard was important, because the system utilizes Advantech PCI-based I/O cards – PCI 1720 and PCI 1730. Spare PCI slots allow for scalability of the system in the future development.

There were 2 main reasons for choosing Advantech I/O cards – a reasonably low price and included Linux drivers, along with a high-level C++ biodaq library for I/O handling. The PCI 1720 analog output card provides 4 outputs with signals up to 10 V or up to 20 mA. A PCI 1730 digital input/output card provides 16 isolated digital inputs with up to 30 V input range and 16 isolated digital outputs with up to 40 V output range.

2.2 Software. The custom software part prepared for the presented system is based on the Matlab/Simulink s-function mechanism, Embedded Coder toolbox and Advantech biodaq library as the interface for the utilized I/O cards. The s-function was implemented in C++ to be aligned with the C++ API provided by the biodaq library. One of the important aspects of the implementation was to set a proper s-function block sample time, which should correspond to real-time performance requirements. In this case, it was inheritance of the sample time from the s-function caller inside the simulation. The forbidden s-function sample time variant is continuous sample time, which produces indeterminism that is unacceptable for a real-time solution.

Another important subject related to the software part of the presented simulator is the system target file for the code generation module. The utilized Embedded Coder toolbox, within its default target file – ert.tlc – does not provide any application handling mechanisms dedicated to real-time Linux performance. This means that the generated main application file does not have an application step time corresponding to the simulation step pre-set in the Simulink options. As a result, the application works with a nondeterministic time step, which is as fast as the Linux scheduler's execution of the application. Additionally, in the default target file, no overrun diagnostics are provided, which is crucial for verification of real-time simulation performance [21]. By default, the toolbox's designers assumed that the user would modify the system target file according to specific target requirements. In consequence, to adjust the code generated by the Embedded Coder to the real-time simulation system's performance, either a new system target file should be created, or some appropriate customized target file should be utilized with the necessary modifications.

In the presented system, the second of the aforementioned variants was applied with the help of the Linux Embedded Coder's system target file [22]. The applied target file smoothly solves step execution cycle problems and provides the required mechanism for overrun verification and reporting. Further target file modifications were provided to enhance real-time performance following Linux RT Preempt guidelines [23]. The enhancements are as follows:

- **Real-time scheduling and priorities**;
- **Memory locking.** Locks all pages mapped into the address space of the calling process preventing that memory from being paged to the swap area;

- **Limiting processor power saving states and state transitions.** To prevent the system from entering power-saving states and provide the fastest time out of the idle state, the kernel was booted with processor.max_cstate=1 and idle=poll options;
- **Disabling X window server and network interfaces.**

3. Performance verification

The main criterion in the system performance verification was to check the number of overruns of the simulation step and the operating system setup's influence on that number. For the system's performance benchmarking, 2 simulation models were prepared. The first model was kept very simple and includes only the analog output interface. The second model is dedicated not only to performance verification, but also to protection of relays testing, and hence is much more complicated and includes both analog output and digital input interfaces.

3.1. Simplified model-based performance verification. The simplified model utilized in the first part of performance verification is depicted in Fig. 2. The model contains only a discrete Sine Wave source, analog_output simulator s-function interface and Scope used for signal comparison. The Sine Wave source feeds the analog output with a 0.5 V 50 Hz sinusoidal signal.

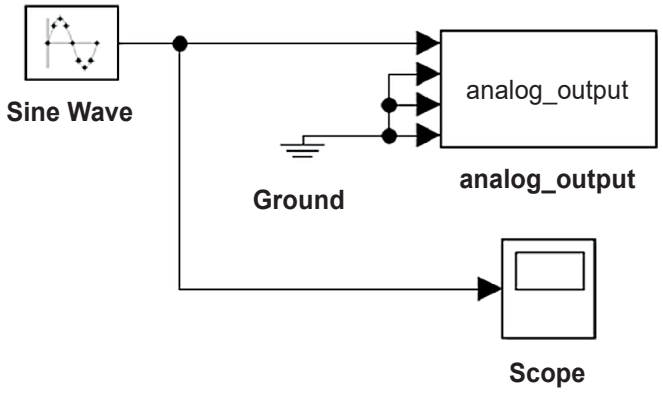

Fig. 2. Simplified model for system performance verification

The utilized verification method runs the executable application generated by the embedded coder for 40 s and logs all overruns in the proper error.log file. Tests were performed with a real-time FIFO scheduling policy and with 100 µs and 50 µs model time steps. The results for the described test cases are presented in Tables 1 and 2, respectively.

The results reveal that the crucial factor influencing the system's performance is operating system "tuning", which means disabling all unnecessary processes and services that might distort real-time system operation (in the presented case, graphical and network interfaces). This particular action eliminates the overrun (for 100 µs time step) or limits it to a very small number (for 50 µs time step). Further enhancements like real-time prior-

Table 1
Performance verification with simplified model and 100 µs time step

Test case		Test iteration	Number of overruns
1	Priority –3, X window server and Network interface enabled, memory locked	1	3
		2	26
		3	2
2	Priority –3, X window server and Network interface disabled, memory locked	1	0
		2	0
		3	0
3	Priority –50, X window and Network interface disabled, memory locked	1	0
		2	0
		3	0
4	Priority –99, X window and Network interface disabled, memory locked	1	0
		2	0
		3	0
5	Priority –99, X window and Network interface disabled, memory unlocked	1	0
		2	0
		3	0
6	Priority –99, X window and Network interface disabled, memory locked, limiting power saving states	1	0
		2	0
		3	0

Table 2
Performance verification with simplified model and 50 µs time step

Test case		Test iteration	Number of overruns
1	Priority –3, X window server and Network interface enabled, memory locked	1	120
		2	115
		3	250
2	Priority –3, X window server and Network interface disabled, memory locked	1	0
		2	2
		3	10
3	Priority –50, X window and Network interface disabled, memory locked	1	13
		2	4
		3	5
4	Priority –99, X window and Network interface disabled, memory locked	1	6
		2	0
		3	8
5	Priority –99, X window and Network interface disabled, memory unlocked	1	4
		2	3
		3	2
6	Priority –99, X window and Network interface disabled, memory locked, limiting power saving states	1	0
		2	0
		3	0

ities also improve the performance, which can be seen for 50 µs time step. However, in the performed tests, this effect is not so significant. It is more visible when changing from –3 to –50 than from –50 to –99. Memory locking is always important in real-time applications, but its effect might not be highlighted in the obtained results, as memory swapping is a non-deterministic process. Limiting the power saving states also yields notable improvement by eliminating the overrun phenomenon in 50 µs time step tests.

3.2. Full model-based performance verification. The full model utilized in the second part of performance verification is depicted in Fig. 3. The model represents a part of the 15 kV, 50 Hz distribution network containing one incomer and two feeders.

The feeders are modeled by 2 Three-Phase PI Section Line elements in each feeder. This setup represents changing fault location with the Three-Phase element connected between 2 PI sections. The applied line parameters are: 0.23 Ω/km positive sequence reactance, 0.38 Ω/km zero sequence reactance, 0.17 Ω/km positive sequence resistance and 0.2 Ω/km zero sequence resistance. The length of Feeder 1 is 30 km and of Feeder 2–5 km. The active power values of feeder loads are 5 MW and 0.8 MW, respectively. Additionally, both feeders are equipped with circuit breaker models. However, CB 1 is controlled by the digital_input element, whereas CB 2 is permanently closed in the described application.

Feeder 1 currents are measured, and phase 1 current is further utilized in protection relays testing, thus it is scaled and fed to the channel 1 of the analog_output block. Scaling consists of two parts, where the first corresponds to the CT 1/2000 turn ratio, and the second corresponds to Omicron CMS 156 amplifier amplification adjustment, which is 5A/V. Hence the voltage output value will additionally be divided by 5 before amplification.

The disturbance simulated in the model occurs in feeder 1 after 5 s of the simulation, and it is a solid three-phase fault. The applied fault location of feeder 1 is at 20% of the line's length.

The modeled network is also grounded via a Petersen coil. However, during the performed tests, the actual network detuning is irrelevant due to the three-phase character of the simulated fault.

The applied verification method is the same as for the simplified model. The results are presented in Tables 3 and 4. Comparison with the simplified model's results from Tables 1 and 2

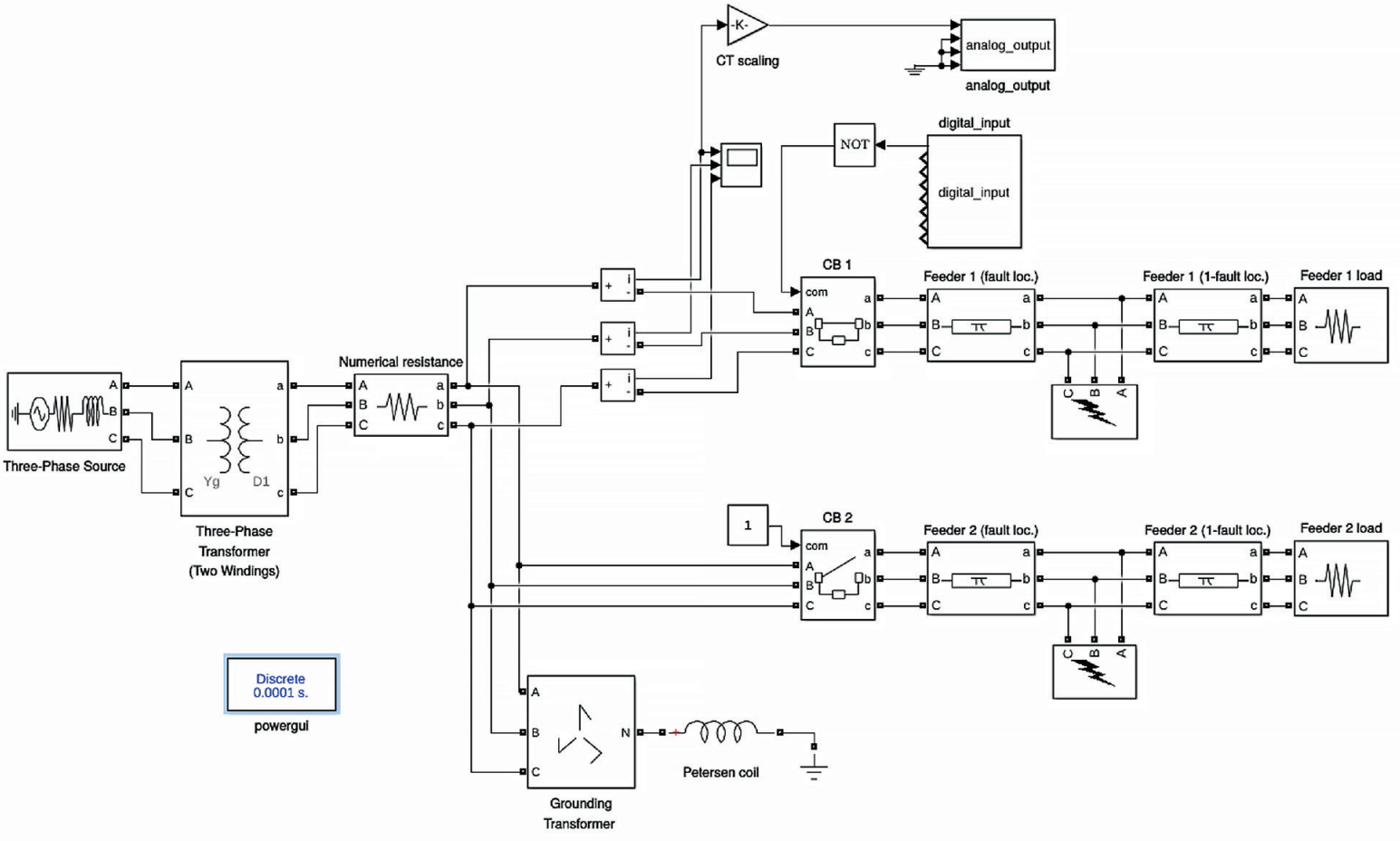

Fig. 3. Full model for system performance verification and for the protection relays testing

Table 3
Performance verification with full model and 100 µs time step

Test case		Test iteration	Number of overruns
1	Priority –3, X window server and Network interface enabled, memory locked	1	48
		2	3
		3	3
2	Priority –3, X window server and Network interface disabled, memory locked	1	3
		2	3
		3	3
3	Priority –50, X window and Network interface disabled, memory locked	1	3
		2	4
		3	3
4	Priority –99, X window and Network interface disabled, memory locked	1	3
		2	3
		3	3
5	Priority –99, X window and Network interface disabled, memory unlocked	1	3
		2	3
		3	3
6	Priority –99, X window and Network interface disabled, memory locked, limiting power saving states	1	0
		2	2
		3	0

Table 4
Performance verification with full model and 50 µs time step

Test case		Test iteration	Number of overruns
1	Priority –3, X window server and Network interface enabled, memory locked	1	1061
		2	1199
		3	1053
2	Priority –3, X window server and Network interface disabled, memory locked	1	38
		2	19
		3	52
3	Priority –50, X window and Network interface disabled, memory locked	1	45
		2	26
		3	69
4	Priority –99, X window and Network interface disabled, memory locked	1	45
		2	40
		3	56
5	Priority –99, X window and Network interface disabled, memory unlocked	1	60
		2	43
		3	61
6	Priority –99, X window and Network interface disabled, memory locked, limiting power saving states	1	3
		2	3
		3	3

highlights how the model's complexity influences the overall system performance. Even with the higher 100 µs time step, a small number of overruns occurs. With the 50 µs time step, the number of overruns increases significantly, especially when the graphical and network interfaces are enabled. The most optimal test case with –99 task priority still results in overruns within the range of 45–56. In this case, the variant with limited power saving states yields much improvement by practically eliminating overruns for the 100 µs time step and highly limiting their number for the 50 µs time step. The conclusion is that in a larger, more complex model, it is even more crucial to limit transitions to the processor power saving states from the perspective of real-time performance.

4. Protection relays testing

As it was mentioned in the previous paragraph, the full model depicted in Fig. 3 is also applied for verification of simple protection relays testing capabilities with the presented system. The tests were performed on the ABB REF615 feeder protection and control relay. As the idea was to make it as simple as possible, the PHIPTOC instantaneous overcurrent relay function was used. The function start value is set to 1.0 xIn, which corresponds to 2000 A. The number of start phases is set to 1, as only one analog output of the simulator is utilized in tests. There are two different operating delay time settings, 50 and 80 ms, which were applied for verification to have some reference with different setups. The diagram of the connection between the simulator system and tested relay is depicted in Fig. 4.

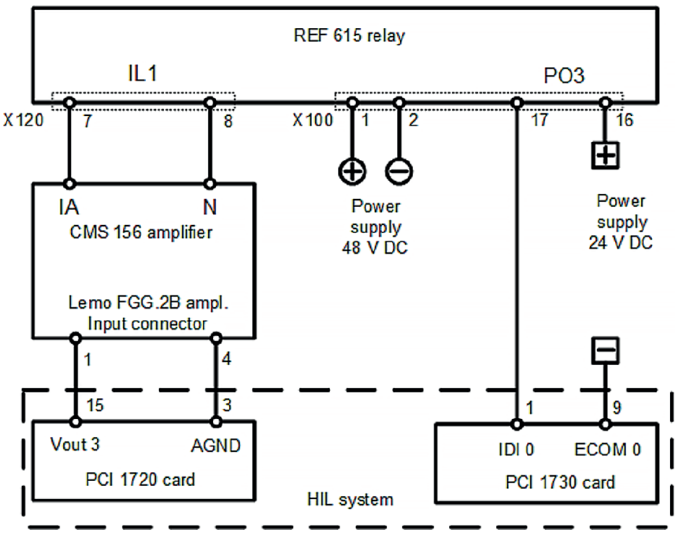

Fig. 4. Connection diagram for protection relay testing

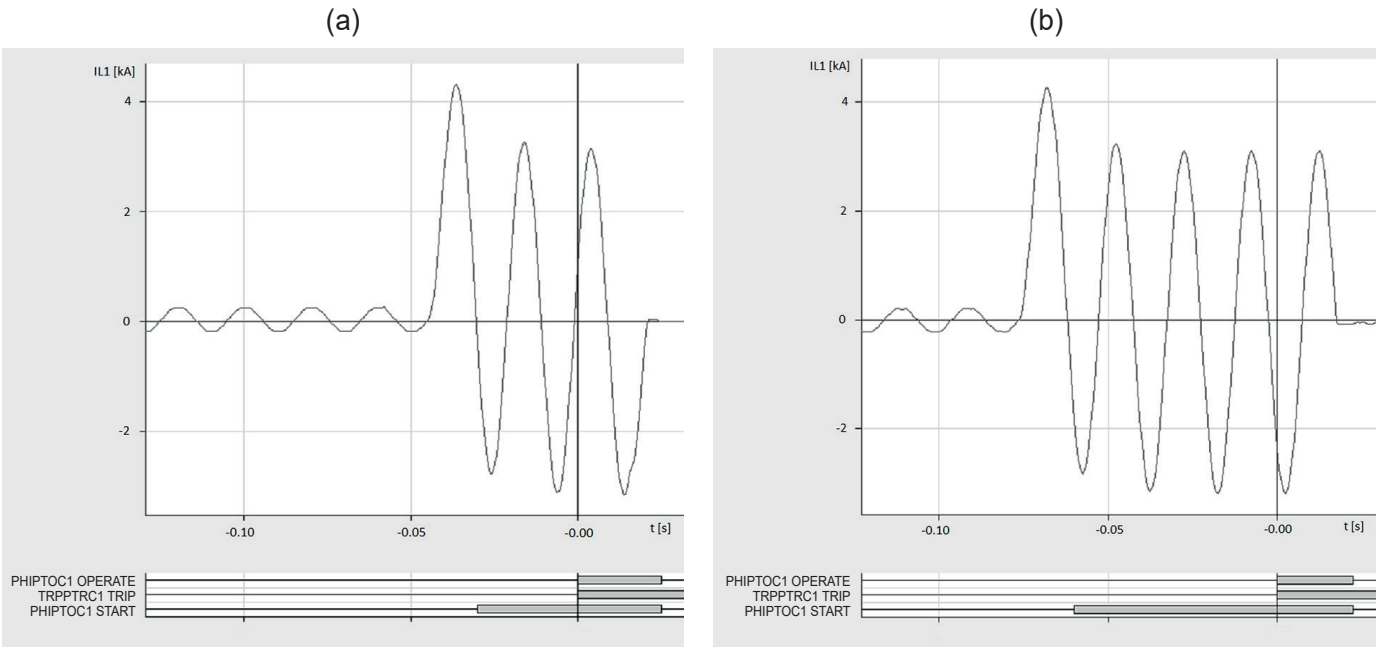

Fig. 5. Exemplary results of the REF 615 instantaneous overcurrent function tests utilizing the proposed HIL system for 50 (a) and 80 (b) ms operating delay time settings

The exemplary results of the protection relay tests, taken from the REF 615 disturbance recorder's Comtrade files, are depicted in Fig. 5. The oscillograms prove that the simulator system works as expected in the case of overcurrent relays testing.

Operate signals (PHIPTOC1 OPERATE) appear 30 and 50 ms after the start signal (PHIPTOC1 START) in the case of 50 and 80 ms operating delay time settings, respectively. However, it should be noted that the setting includes 20 ms of the function's operating delay (corresponding to the full cycle window of the digital filter). Breaker tripping signal (TRPPTRC1 TRIP) is triggered directly by the operate signal. After the operate signal, it takes roughly one cycle before the current is tripped. In this case, normal circuit breaker opening time is not simulated, but the reason for that additional delay comes from the binary outputs' operating time (less than 15 ms) and from the Simulink circuit breaker model that simulates the arc extinction process by opening the breaker device when its current passes through 0.

The symmetrical fault current measured from Comtrade files was approximately equal to 2230 A RMS, which corresponded to the result from Simulink and was obviously higher than the 2000 A function start value.

5. Conclusion

In this paper, A real-time hardware-in-the-loop simulation system based on Linux, dedicated for power system protection testing, is presented.

The tests and analysis that were conducted proved that the simulator demonstrates satisfactory performance for protection relays testing within the 50-100 µs time step range. However, there is still some small space for further real-time performance optimization. Decreasing the achievable time step in the case of the power system protection area does not make much sense unless power electronics performance and traveling wave phenomena are considered. But for these applications, FPGA hardware seems to be the best solution.

It was also shown with A simple overcurrent function test example that the simulator can successfully operate with A physical protection relay in A closed loop, providing valid and consistent test results.

The presented system can be utilized as A simpler and more accessible hardware-in-the-loop testing alternative to commercial simulators.

References

[1] H.W. Dommel, "Digital Computer Solution of Electromagnetic Transients in Single- and Multiphase Networks", *IEEE Trans. Power ASyst.* 88(4), 388–399 (1969).

[2] P.G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A Real Time Digital Simulator for Testing Relays", *IEEE Trans. Power Deliv.* 7(1), 207–213 1(992).

[3] C. Dufour and J. Belanger, "A PC-Based Real-Time Parallel Simulator of Electric Systems and Drives", *Parallel Comput. Electr. Eng. 2004. PARELEC 2004. Int. Conf.* pp. 105–113, 2004.

[4] D. Majstorovic, I. Celanovic, N.D. Teslic, N. Celanovic, and V.A. Katic, "Ultralow-Latency Hardware-in-the-Loop Platform for Rapid Validation of Power Electronics Designs", *IEEE Trans. Industrial Electronics* 58(10), 4708–4716 (2011).

[5] Z. Wang F. Zeng, P. Li, C. Wang, X. Fu, and J. Wu, "Kernel Solver Design of FPGA-Based Real-Time Simulator for Active Distribution Networks", *IEEE Access* 6, 29146–29157 (2018).

[6] 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).

[7] V.A. Papaspiliotopoulos, G.N. Korres, V. A. Kleftakis, and N.D. Hatziargyriou, "Hardware-In-the-Loop Design and Optimal Setting of Adaptive Protection Schemes for Distribution Systems with Distributed Generation", *IEEE Trans. Power Deliv.* 32(1), 393–400 (2017).

[8] A.S. Makhzani, M. Zarghami, B. Falahati, and M. Vaziri, "Hardware-in-the-loop testing of protection relays in distribution feeders with high penetration of DGs", *2017 North Am. Power Symp. NAPS 2017*, 2017.

[9] M.S. Almas and L. Vanfretti, "Methodologies for Power Protection Relay Testing: From Conventional to Real-Time Hardware-in-the-Loop (HIL) Simulation Approaches", in *International Conference on Power Systems Transients (IPST2013),* Vancouver, Canada*,* 2013.

[10] M.S. Almas, R. Leelaruji, and L. Vanfretti, "Over-current relay model implementation for real time simulation & Hardware-in-the-Loop (HIL) validation", *IECON Proc. Industrial Electron. Conf.*, pp.4789–4796, 2012.

[11] F. Coffele, C. Booth, and A. Dyśko, "An Adaptive Overcurrent Protection Scheme for Distribution Networks", *IEEE Trans. Power Deliv.* 30(2), 561–568 (2015).

[12] J. Jia, G. Yang, A.H. Nielsen, and P. Roenne-Hansen, "Hardware-in-the-loop tests on distance protection considering VSC fault-ride-through control strategies", *J. Eng.* 2018(15), 824–829 (2018).

[13] D.T. Dantas, E.L. Pellini, and G. M. Junior, "Energy and reactive power differential protection hardware-in-the-loop validation for transformer application", *J. Eng.* 2018(15), 1160–1164 (2018).

[14] Z.Y. Xu, S. Member, Z.P. Su, J.H. Zhang, A. Wen, and Q.X. Yang, "An Interphase Distance Relaying Algorithm for Series-Compensated Transmission Lines", *IEEE Trans. Power Deliv.* 29(2), 834–841 (2014).

[15] IEEE PES Task Force on Real-Time Simulation of Power and Energy Systems, "Applications of Real-Time Simulation Technologies in Power and Energy Systems", *IEEE Power Energy Technol. Syst. J.* 2(3), 103–115 (2015).

[16] M. Baszynski, "Low cost, high accuracy real-time simulation used for rapid prototyping and testing control algorithms on example of BLDC motor", *Archives of Electrical Engineering* 65(3), 463–479 (2016).

[17] 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. Ac.: Tech.* 57(4), 345–354, 2009.

[18] Y. Li., B. Zhang, and X. Xu, "Decoupling control for permanent magnet in-wheel motor using internal model control based on back-propagation neural network inverse system", *Bull. Pol. Ac.: Tech.* 66(6), 961–972, 2018.

[19] Simulink Desktop Realtime Toolbox. The MathWorks, Inc., Natick, MA, USA. [Online]. Available: https://www.mathworks.com/products/simulink-desktop-real-time.html

[20] A. Smolarczyk, R. Kowalik, and E. Bartosiewicz, "Closed-loop testing method for protective relays with use of MATLAB/Simulink software", *12th IET International Conference on Developments in Power System Protection (DPSP 2014)*, Copenhagen, Denmark, pp. 1–6, 2014.

[21] IEEE PES Task Force on Real-Time Simulation of Power and Energy Systems, "Real-Time Simulation Technologies for Power Systems Design , Testing , and Analysis", *IEEE Power Energy Technol. Syst. J.* 2(2), 63–73, 2015.

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

[23] The Linux Foundation, "HOWTO build A simple RT application", [Online]. Available: https://wiki.linuxfoundation.org/real-time/documentation/howto/applications/application_base.