Renewable Energy, Environment, and Power Quality Journal (REE&PQJ)
ISSN:3020-531X
Volume No.2, July 2024

Optimizing the utilisation of different computational hardware types for real time simulation

D. Kiss, A. Balogh Dr. I. Varjasi Dr.

[1] Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics – Műegyetem rkp. 3., H-1111 Budapest, (Hungary)
Phone number:+36205500468

Abstract. Development of power electronics devices have never been so widespread among different industries than today. Efficient power conversion, precise and efficient control of electrified actuator systems, and of course electric powertrains and electrical power generation is a challenge everywhere. The opportunities provided by Wide-bandgap semiconductor devices fuelling the innovation even further. The paper focuses on the potential utilisation of various computing hardware's for the purpose of real-time hardware-in-the-loop simulation to support power electronics development processes. FPGA devices can provide fast execution, with specific models deployed in their fabric, but compromising flexibility, both in hardware and in numerical precision, whereas ARM based CPU cores can execute different models more easily but compromising execution speed. This paper proposes an approach which embodies the benefits of both worlds, utilizing the different hardware platforms in one system.

Key words. Power electronics, testing, HIL, electrical drivetrain

1. Introduction

The control of modern power converters is a continuous challenge. The task is to design devices with optimal utilization, minimum cost, but with better and better efficiency and power quality. The development of the underlying hardware architecture enables higher and higher switching frequencies, leading towards more efficient, more compact devices, and better signal quality.[1] Devices based on classic semiconductor devices, like silicon MOSFETS and IGBTs are tend to operate in the range of 2 – 15 kHz, whereas modern, wide-bandgap devices, utilising the potential of Silicon Carbide (SiC) and Gallium Nitride (GaN) semiconductors in the range of 20 kHz – 50 kHz.[2] The benefit of the higher switching frequency is clearly visible in the size of the passive components required to the power circuits, like capacitors and inductors, resulting in much better packaging efficiency, thus more compact devices. Reducing the volume and weight of the power converters, thus improving the gravimetric and volumetric efficiency of the system is especially important in case of aerospace applications like autonomous drones, or VTOL vehicles, which significance are rising rapidly.

The challenge of designing, testing and validating control loops for these converters is significant. To maintain the control and keep the device operating within the limits provided by various standards, the control software needs to be designed and validated with care. During the development off-line simulations can be used to size the system and the components appropriately. However with the increasing switching frequencies, off-line simulation are starting to become very resource intensive and time consuming, with run-times several magnitudes slower than real-time. The low-level control software interacting with the switching elements and regulating the current waveforms can only be validated real- time, since in the end, usually an embedded microcontroller or FPGA will be responsible for the control. There are several reasons to use a Hardware-in-the-Loop (HIL) system during the development:

- HIL devices are usually flexible, the simulated hardware can be changed, as the development progressing
- HIL is available earlier and can be reused in different projects
- Failure of the control software cannot lead to the destruction of the hardware
- HIL devices are not requiring laboratory equipment, high-voltage or high-power to operate

HIL simulators however still can be a scarce resource for developers. Since the switching frequencies of a range of 20-50 kHz, and the low time constant of the power electronics devices, real-time requirements are very strict in these applications. [3] If the modelled switching frequency (f_{sw}) is 20 kHz, then to achieve a desirable 0.1% Pulse Width Modulation (PWM) signal resolution in simulation, the sample time of the model needs to be at least 50 ns to detect the change in duty cycle

with the desired resolution. 50 ns timing is not sustainable with microprocessors and microcontrollers, neither with PC setups; hence such models are usually compiled to execute on FPGAs. FPGAs ability to implement custom hardware, tailor made to the model enable such critical timing. The limitation usually is the cost, since large capacity FPGA devices (such as the Kintex or Virtex family) are coming with a significant price tag and affordable devices are limited in terms of model complexity.[4][5] A potential solution could be to divide the model in multiple partitions, and only use the FPGA for the most time sensitive components. The rest could be executed on less special hardware. The ZYNQ hardware platform offers a composition of ARM cores and an FPGA fabric, which could be utilized to test the aforementioned approach.

This paper suggests an implementation of such HiL simulator, based on a low cost, commercially available computing platform. The main computational hardware element is an AMD (formerly XILINX) XC7Z020-1CLG400C SoC incorporating a Dual-core ARM Cortex-A9 processor and an Artix-7 FPGA with 85.000 programmable logic cells.

In the second chapter the paper introduces the simulated power system and investigates the option of deployment the various parts of the simulation. Chapter 3 explains how the model were implemented in MATLAB/Simulink, Chapter 4 evaluates the results of the on-line simulation, and finally, chapter 5 concludes the paper and suggest ways to move forward.

2. The example system and partitioning

To demonstrate the concept the usual setup of a PMSM drivetrain is proposed. The system will contain a DC power source or energy storage, a 3-phase, 2-level inverter, and a 3-phase PMSM machine, with an incremental encoder. The DC source will be a battery, with 60 V of nominal voltage, which will be modelled with an ideal voltage source (U_{DC}). The voltage is chosen to be compatible with an available low-voltage experiment drive system, enabling further comparison measurements with real-life conditions. A resistive pre-charge system is modelled for realistic startup. For the sake of simplicity, the inverter will not model the details of semiconductor devices, only implement them as ideal switches. Finally, the machine will be modelled with a $R - L - U_{EMF}$ model, implemented in x-y coordinate system. [6][7]

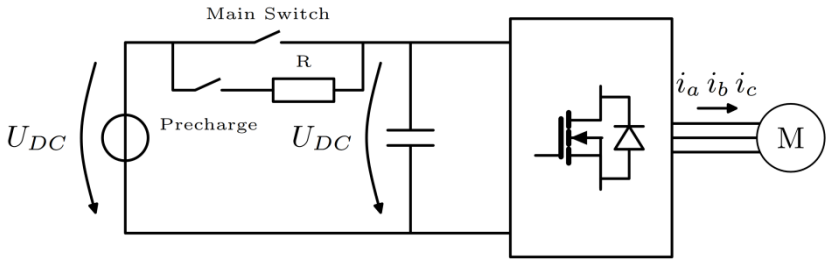

Figure 1. The schematic diagram of the modelled system

In this model, the most time critical pars are the power circuit of the inverter, and the inductance of the electrical machine. Hence, the parts which needed to be implemented in the FPGA:

- Sampling the PWM signals which are controlling the switches within the inverter.
- The voltage on the inductance of the machine.
- Motor current and flux calculation.
- Emulation of the position encoder.
- Driving the Σ/Δ digital - analog converters to generate the feedback signals

In summary the motor current related calculations and the I/O handling is implemented within the FPGA.

On the ARM side, the rest of the model, which is not time critical can be implemented. These parts are either having lower electrical time constant (DC voltage), or linked to a mechanical system (torque, Back-EMF, etc.), with usually even slower time constants:

- DC system voltage
- Motor torque calculation, mechanical loads
- Motor position calculation
- Thermal models

Regarding timing, the FPGA subsystem is running with a 10 MHz clock speed, which leads to a 100 ns sample time. The parts modelled on the ARM core, will be running in an interrupt triggered by the FPGA core with 100 kHz, leading to a 10 us sample time, 100 times slower.

First, the theoretical feasibility of the timing constraints should be assessed. The bottleneck of the system is the calculation of the Back-EMF voltage. U_{EMF} is used to calculate the motor currents in every FPGA cycle, however it is only updated in

every 10 us, when the ARM task is running. In the behaviour of the system this phenomenon introduces an error which behaviour could be decoupled into a static and a dynamic component.

The static component is due to the sample time of the ARM subsystem. The U_{EMF} signal will have a 100 kHz noise addition as a simulation artefact since it will be composed from 10 us long steps. To estimate the error, the average difference between the actual U_{EMF} and the sampled signal should be calculated. De difference of the two signal will have a Vs dimension, so it is a flux error on the inductance of the motor

$$\Psi_{err} = \widehat{U}_{EMF} \cdot \omega \cdot \frac{t_{sample}^2}{8} \qquad (1)$$

$$\Psi_{nom} = L_{relEMF} \cdot \frac{\widehat{U}_{nom}}{\omega} \qquad (3) \qquad L = \frac{L_{rel} \cdot \frac{U_{nom}}{I_{nom}}}{\omega} \qquad (2)$$

$$\frac{\Psi_{err}}{\Psi_{nom}} = \widehat{U}_{EMF} \cdot \omega \cdot \frac{t_{sample}^2}{8} = 4.5\ nVs$$

Compared to the nominal flux of 52,6 mVs, the resulted value is marginal, therefore it is safe to conclude the error will not lead to issues with the simulation.

The dynamic part of the error is due to the changing nature of the U_{EMF} vector, both in position, and in magnitude. The U_{EMF} value is calculated as follows:

$$\begin{aligned} u_x(t) &= R_s \cdot i_x(t) - \omega \cdot \Psi \cdot \sin(\alpha) \\ u_y(t) &= R_s \cdot i_y(t) + \omega \cdot \Psi \cdot \cos(\alpha) \end{aligned} \qquad (2)$$

In the equations above, the rotor position is an input parameter. The position during constant rotor speeds can be easily predicted in the next sampling period. However, when the torque is changing rapidly, the prediction will have an error.

As it is shown in eq. 3-4 motor speed is calculated from the motor torque, which is calculated from the motor current, so the current signals need to be fed to the ARM subsystem.

$$m = \frac{3}{2} \cdot i_q \cdot \Psi \qquad (3)$$

$$m - m_t = \theta \frac{d\omega}{dt}, \qquad \omega = \frac{d\alpha}{dt} \qquad (4)$$

To avoid under sampling caused by the much lower sampling rate of the ARM subsystem, the transferred current value is averaged with a 10 us window, also introducing latency in the system.

We can conclude, the speed error could be only caused by the torque error, and the torque error is caused by a potential current error, which can be caused by inaccurate information about U_{EMF} . The use cases in which this type of error could be most apparent is either accelerating or decelerating the machine, since these operating modes are embodying high torque and high $^{d\omega}/_{dt}$ .With the example edge-case, the torque error during startup with a step to nominal voltage can be as high as 20% only after 10 µs, if we assume a $\omega = 1\ kHz$ fundamental frequency, and 30% relative inductance.

$$\Delta i = \frac{U_{nom}}{L} \cdot t_{step} = I_{nom} \cdot \frac{\omega \cdot t_{step}}{L_{rel}} = 1.2\ A\ (\sim 20\%) \qquad (5)$$

Such high $^{dI}/_{dt}$ is not sustainable during normal operation since it would cause overcurrent in very short time, nominal current would be reached in 50 µs. The original acceptance criteria were defined as a speed error, so the effects on the rotor speed should be investigated. To estimate the rotor speed error, the motor inertia should be known. In our case of estimation the motor startup time (t_{start}) seem to be more useful. t_{start} is the time until which the machine with nominal current reaches the nominal speed. The higher the inertia, the slower the derivative of the rotor speed is, so to have a worst-case assumption, let's assume t_{start} = 100 ms. This value is on the extreme end, even with free-running high-speed machines.

$$\frac{\Delta\omega}{\omega_{nom}} = \frac{t_{step}}{T_{start}} = 0.02\ \% \qquad (6)$$

It can be seen, with these condition, the 20% current error, during the time until which overcurrent is not reached (30 us) will only cause 0.02% speed error, which is insignificant. Therefore, it can be concluded the modelling approach is adequate.

3. System Implementation in Simulink

The system for the demonstration was realized with the following parameters:

Table 1. Machine parameters

	Value	Unit
$\boldsymbol{U_{DC}}$	60	V
$\boldsymbol{pp}$ – pole pair number	2	-
$\boldsymbol{L_{rel}}$ – relative inductance	30	%
$\boldsymbol{R_{rel}}$ - relative resistance	5	%
$\boldsymbol{L}$ - inductance	28	mH
$\boldsymbol{R}$ - resistance	290	mΩ
$\boldsymbol{\Psi}$ – flux linkage	52,6	mVs
$\boldsymbol{n_nom}$ – nominal speed	6000	1/s

To maintain control over the system, a Texas Instruments TMS320F069 microcontroller will be utilized. The applied algorithm is a standard cascaded FOC control loop in a cascaded speed control. For initial tests, the motor torque calculation was disabled to examine the behaviour in a fixed position.

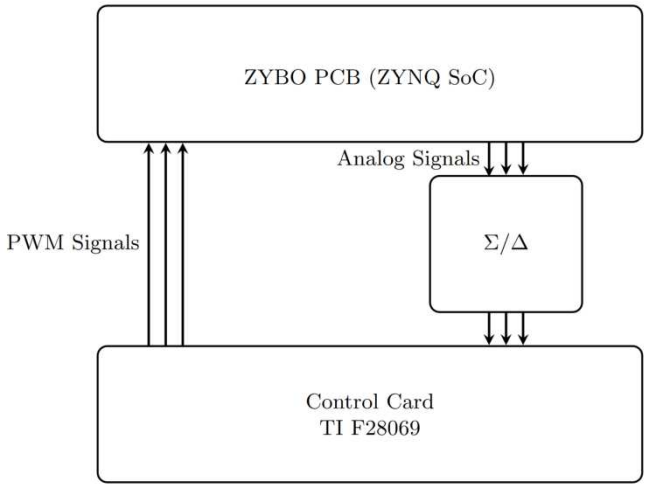

Figure 2. Block diagram of the real-time simulator hardware[8]

The model is implemented in Simulink for both subsystems. Since MATLAB/Simulink is capable to generate code wether in C or in VHDL, the same model structures can be used, and the partitioning can be changed on demand. [9][10]

To model the synchronous machine, the following equations were implemented:

To emulate the behaviour of the half-bridges, the logic described with eq. 7 is implemented. The model determines the output voltage based on the PWM signal state and the direction of the current. It does not include any timing delay, transition dynamics, neither the emulation of the losses.

$$\begin{matrix} U_{inv} = 0\ V \mid if\ PWM_H = 0, or\ PWM_L = 1\ \&\ I_{inv} \\ > 0 \\ U_{inv} = U_{DC} \mid if\ PWM_H = 1, or\ PWM_L = 0\ \&\ I_{inv} \\ < 0 \end{matrix} \qquad (7)$$

The motor are modelled with a linear equivalent model, in x-y coordinate system, described by eq. 8.

$$i_{x,y}(t) = L \cdot \left(U_{inv_{x,y}}(t) - U_{EMF_{x,y}}(t)\right) dt \qquad (8)$$

Please note, that for efficient VHDL implementation, the model is used with fix-point arithmetic.

The rest of the model -calculating the torque and the Back-EMF- in the ARM partition is implementing the following equations:

$$\begin{aligned} i_d &= i_x \cdot \cos(\alpha) + i_y \cdot \sin(\alpha) \\ i_q &= i_y \cdot \cos(\alpha) - i_x \cdot \sin(\alpha) \end{aligned} \qquad (9)$$

$$\begin{aligned} m &= \frac{3}{2} \cdot i_q \cdot \Psi \\ m - m_t &= \theta \frac{d\omega}{dt} \\ \omega &= \int \theta \cdot (m - m_t) dt \end{aligned} \qquad (10)$$

$$\begin{aligned} u_x(t) &= R_s \cdot i_x(t) - \omega \cdot \Psi \cdot \sin(\alpha) \\ u_y(t) &= R_s \cdot i_y(t) + \omega \cdot \Psi \cdot \cos(\alpha) \end{aligned} \qquad (11)$$

In exchange of the slower cycle times, on the ARM side, the signals of the model can be used as simple floating-point numbers.

Another benefit of using Simulink as a code generation tool, is the model can be verified before implementing it on the real-time hardware.

4. On-line simulation and performance evaluation

After code generation and integration within the framework developed on the ZYNQ platform, real-time validation and verification of the model can be executed.

To evaluate the performance and the usability of the system it should be investigated if the models are behaving as expected electrically and mechanically. It should be also proved; the different sample times of the ARM and the FPGA subsystem are not causing unexpected errors during the simulation. [11][12]

First, the behaviour of the electrical circuit was tested. With a fixed mechanical axle, a step change is introduced to the reference of iq signal, from 1 A to 6 A. The results are demonstrated on Figure 3.

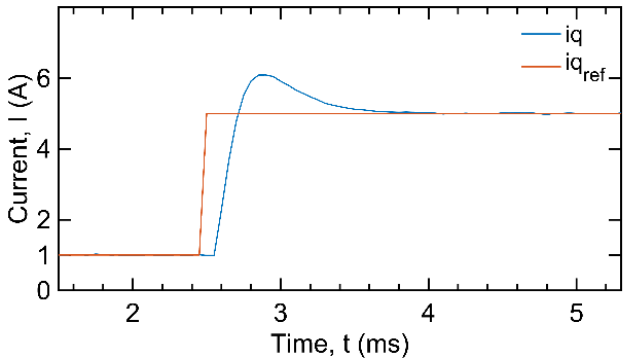

Figure 3. Current step sampled by the control hardware

As it is observable, the current signal behaves as expected: with a little overshoot, it follows the reference value. Here, the current ripple is not observable, this is due to the synchronous sampling of the current signal by the control microcontroller, which means a sample time of 5 kHz. The same step is visible on Figure 4, sampled by the FPGA, therefore the sample time is 10 MHz. Due to the higher resolution, the current ripple is very well discoverable.

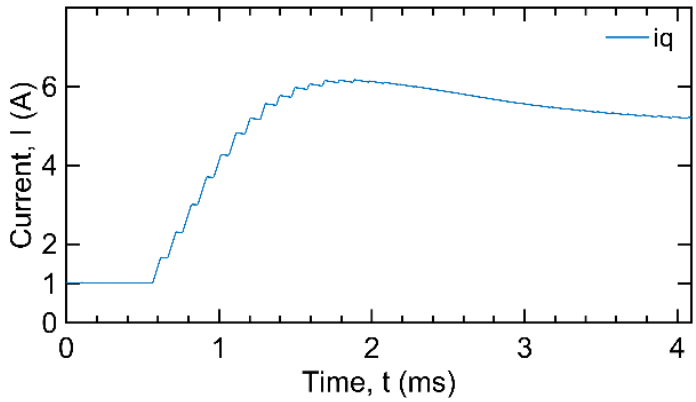

Figure 4. Current step sampled by the FPGA

After the current signals with fixed rotor angle are verified, the functionality is proved in normal speed control mode with an acceleration to a step change from 0 to 500 rpm. The test results are presented on Figure 5. In this case, the sampling was also done by the control hardware microcontroller.

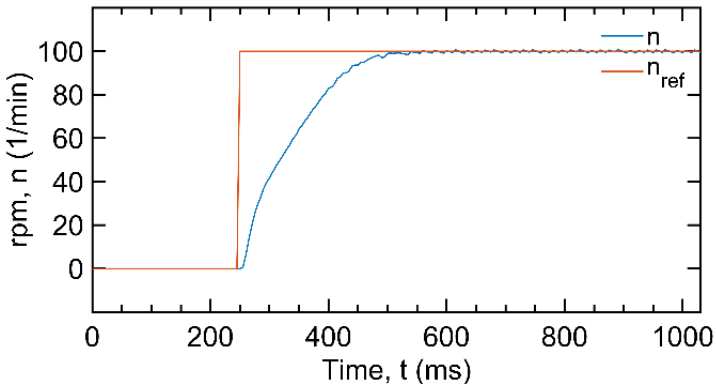

Figure 5. Speed step sampled by the control hardware

Finally the potential introduced errors by the sample time difference between the ARM and FPGA subsystem is examined. As it was disclosed in Chapter 2, the root cause of the potential issues is expected to be the Back EMF voltage generated in the machine. U_{EMF} is calculated in the ARM subsystem, which is only running in every 10 us, however the signal value is used to calculate the current waveform. The sample time difference will only cause an issue if U_{EMF} is changing, which occurs when the angular velocity of the machine is changing, therefore the current waveforms during acceleration are examined on Figure 6.

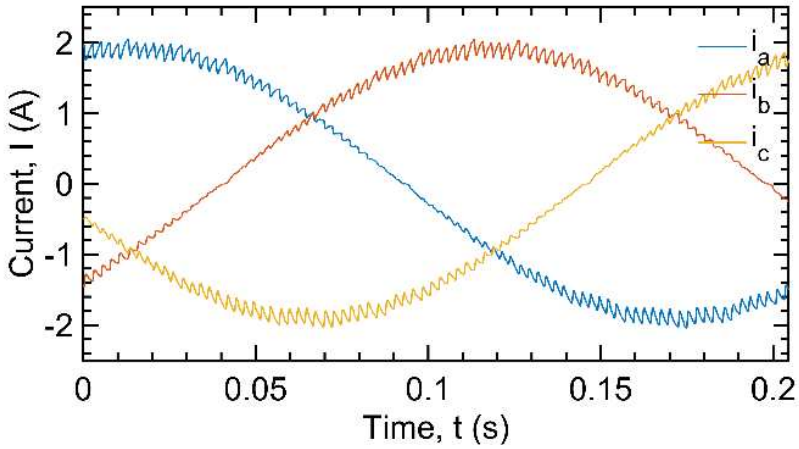

Figure 6. Current waveforms sampled by the FPGA

The current ripple is looking like expected, without any additional 100 kHz noise added. An artefact can be visible with roughly 1/6th of the frequency of f_{SW}. Since there is no correlation between the ARM sampling rate and this frequency, it's safe to conclude that they are not related. The root cause of this phenomenon is the poor resolution of the interpretation of the PWM signal.

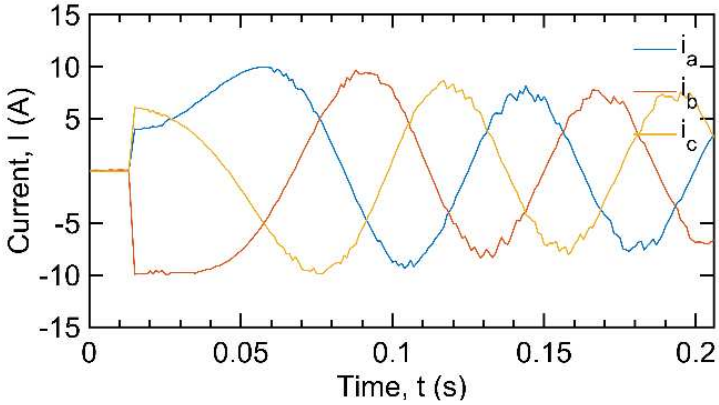

Figure 7. Current waveforms sampled by the control hardware

On Figure 7. A startup from standstill is visible as it was measured by the control hardware microcontroller. Again, the general behaviour is meeting the expectations, with the additional noise around the peak values, which is likely due to the aforementioned PWM sampling rate.

5. Conclusion

As it was shown, the model partitioning is possible between the FPGA and the ARM subsystem, and it can work without losing accuracy and modelling fidelity. The benefits from this approach are in one hand to enable using more cost-effective hardware for simulation purposes, and also to enable engineers to update the mechanical model more quickly and adjust it to the needs of the controller. Reconfiguring the FPGA could take more than 10 minutes, or even hours in extreme cases, since the synthesis and the implementation of the Verilog code is very time consuming, whereas rebuilding the C code only takes a few 10 seconds, enabling much faster iteration loops. Further investigation should be done, how the free FPGA resources should be utilized, either even faster sampling rate should be achieved, or the model should be extended with non-linear functionalities.

Acknowledgement

This research was supported by the Ministry of Culture and Innovation and the National Research, Development and Innovation Office within the Cooperative Technologies National Laboratory of Hungary (Grant No. 2022- 2.1.1-NL-2022-00012).

References

[1] T. Schulte, A. Kiffe, and F. Puschmann. "HIL Simulation of Power Electronics and Electric Drives for Automotive Applications". In: Electronics ETF 16.2 (2012), pp. 130–135.

[2] J. Biela, M. Schweizer, S. Waffler, and J. W. Kolar, "SiC versus Si–evaluation of potentials for performance improvement of inverter and DC-DC converter systems by SiC power semiconductors," IEEE Transactions on Industrial Electronics, vol. 58, pp. 2872–2882, Jul. 2011.

[3] T. Kökényesi and I. Varjasi. "Hardware-in-the-Loop Simulation of Power Converters Using FPGA Devices". In: Proceedings of International Conference on Innovative Technologies. Bratislava, Slovakia, (2011), pp. 294–297.

[4] J. J. Rodríguez-Andina, M. D. Valdés-Peña, and M. J. Moure. "Advanced Features and Industrial Applications of FPGAs – A Review". In: IEEE Transactions on Indus- trial Informatics 11.4 (2015), pp. 853–864.

[5] C. Dufour, S. Cense, V. Jalili-Marandi, and J. Bélanger. "Review of state-of-the- art solver solutions for HIL simulation of power systems, power electronic and motor drives". In: European Conference on Power Electronics and Applications (EPE). Lille, France, (2013), pp. 1–12.

[6] S. R. S. Raihan and N. A. Rahim. "Comparative Analysis of Three-Phase AC-DC Con- verters Using HIL-Simulation". In: Journal of Power Electronics 13.1 (2013), pp. 104– 112.

[7] E. A. Grunditz and T. Thiringer. Modelling and scaling procedure of a vehicle electric drive system. http://publications.lib.chalmers.se/records/fulltext/25348 8/local_253488.pdf. Chalmers Publication Library (CPL), 2017.

[8] S. Sarkar and A. Choubey. "FPGA implementation of all-digital adaptive delta sigma modulator with enhanced SQNR and dynamic range". In: IEEE International Confer- ence on Electronics, Computing and Communication Technologies. Bangalore, India, 2015, pp. 1–6.

[9] A. Bergmann. “Benefits and drawbacks of model-based design”. In: KMUTNB Inter- national Journal of Applied Science and Technology 7.3 (2014), pp. 15–19.

[10] Z. Sütő, T. Debreceni, T. Kökényesi, A. Futó, and I. Varjasi. “Matlab/Simulink Gen- erated FPGA Based Real-time HIL Simulator and DSP Controller: A Case Study”. In: Renewable Energy & Power Quality Journal 1.12 (2014), pp. 431–436. doi: 10. 24084%2Frepqj12.357.

[11] S. R. S. Raihan and N. A. Rahim, “Comparative analysis of three-phase AC-DC converters using HIL-simulation,” *Journal of Power Electronics*, vol. 13, pp. 104–112, Jan. 2013.

[12] T. Kokenyesi and I. Varjasi, “Validation method for hardware-in-the-loop simulation models,” in Proceedings of the 9th EUROSIM Congress on Modelling and Simulation (Eurosim’16), Oulu, Finland, Sep. 2016, pp. 647–652.