A Design Method of an Embedded Real-Time Simulator for Electric Drives using Low-Cost System-on-Chip Platform

Aravinda Perera[1], Roy Nilsen[1], Thomas Haugan[1], Kjell Ljøkelsøy[2]

[1] Norwegian University of Science and Technology, Norway
[2] SINTEF Energy Research, Norway

Corresponding author: Aravinda Perera, aravinda.perera@ntnu.no

Abstract

This paper presents a modular and easily reusable Zynq System-on-Chip (SoC) based Embedded Real-Time Simulator (ERTS) aimed for rapid prototyping of electric drives. The power hardware components of the drive including the voltage source converters (VSC) is programmed in the field programmable gate array (FPGA) fabric of the SoC to achieve real-time emulation. The control algorithms of the electric motor drive are programmed in the on-chip processor which can be used to drive either the physical- or emulated- hardware. The ERTS is scaled in the per-unit system to enhance reusability irrespective of the hardware ratings. The architectures and schematics of different partitions of the ERTS are illustrated. The simulator is demonstrated using a position—sensorless, interior permanent magnet synchronous machine (IPMSM) drive and compared against offline simulation for performance.

1 Introduction

In the process of developing electric motor drive systems, the personal computer (PC) based offline simulation-methods are extensively used, in which, the common practice is to run the control system in discrete- and the power components in continuous- mode to emulate a physical motor drive as close as possible. The type of the solver and the size of the simulation time step (*h*) of these simulation environments determine the stability and the precision of the simulation. When the switching frequency increases in the kilohertz scale, *h* needs to be shrunk below microsecond-level in order to precisely capture the switching transients [1]. One of the main challenges with smaller *h* in the PC-based simulations is that the execution time becomes excessively elongated particularly when simulating computationally intensive systems. In [2], it is reported that the offline simulation can consume beyond 20000 more time than a real-time simulator when power electronic applications are concerned.

The digital real-time simulation (DRTS) technologies [3] that can solve the model equations taking a time-step which is equal to the real-world clock, are able to yield real-time results emulating the physical systems. SoC-based emulation, which falls into the ERTS-category, is one such method that exploits the inherent parallelism in its FPGA fabric in computing the dynamic mathematical models. Implementation of all or parts of the components of the motor drive in the SoC-platform makes hardware-in-the-loop (HIL) scenarios possible in order to evaluate all or certain components of the physical system nondestructively and cost-effectively. In the same time, owing to the on-chip processor(s) in the SoC that can execute the actual control software, the software-in-the-loop (SIL) also becomes possible to validate control software while not having access to the actual hardware or experimental setup. In this manner, ERTS can dramatically reduce the time-to-market and the development costs of electric drives. Having both FPGA and processors in the same package along with other hardware resources makes SoCs an easily customizable yet, compact tool with high data-fidelity [4]. The cost of a modern SoC is just a fraction of the commercially available DRTS technologies and the space requirement, too, is negligible compared to its counterparts, which allow a MW-scale motor drive system be emulated in a pocket-size digital electronic card.

Consequently, ERTS is emerging to be an essential tool in the electric drive development process, although their implementation details are not prevalent. An approach to realize the electric drives in the FPGA, including the control system, are elaborated in [5] and [6]. Authors in [7], and [8] employ an SoC to implement a doubly-fed induction generator and its control either fully hardware or fully software and evaluate the pros and cons. In [9], an ERTS is developed for modular multilevel converter and control in which, the plant is emulated in the processor. Zynq SoC has been employed only to implement the motor drive control algorithm in [10] in which the FPGA is used for Pulse-Width Modulation (PWM) and interrupt generation. A comprehensive ERTS investigation is available in [1], although its application is not focusing electric drives. An IPMSM drive is simulated in [11], where two discrete FPGA and DSP chips have been employed instead of a SoC.

This paper aims to present a design method of a modularized, scalable, reconfigurable, and easily programmable ERTS for electric drives using a state-of-the-art SoC. The proposed simulator effectively exploits the processor system and FPGA of the SoC to emulate a range of industrial electric drives. The ERTS is demonstrated by implementing a sensorless IPMSM drive. In addition to the ERTS-architecture and implementation, the real-time results are compared with offline simulations for the numerical precision and execution time.

2 Application: Sensorless IPMSM Drive System

2.1 IPMSM Dynamic Model

The voltage model and current model of the electrical machine is in stator co-ordinates, when given in the per-unit (pu) system:

$$\underline{u}_s^s = r_s \cdot \underline{i}_s^s + \frac{1}{\omega_n}\frac{d\underline{\psi}_s^s}{dt}; \qquad \underline{\psi}_s^s = \mathbf{x}_s^s(\vartheta) \cdot \underline{i}_s^s + \underline{\psi}_m^s \tag{1}$$

Here, ω_n is the nominal rotational frequency. The superscript and subscript denote the reference frame and the location of the quantity (*s*-stator, *r*-rotor, *m*-magnet) respectively. When the currents are chosen as the state variables, (1) becomes as follows in the rotor coordinates:

$$\underline{u}_s^r = r_s \cdot \underline{i}_s^r + \frac{\mathbf{x}_s^r}{\omega_n} \cdot \frac{d\underline{\mathrm{i}}_s^r}{dt} + \mathbf{j} \cdot n \cdot \mathbf{x}_s^r \cdot \underline{\mathrm{i}}_s^r + \mathbf{j} \cdot n \cdot \underline{\psi}_m^r \tag{2}$$

Here ϑ is the electrical angle of the mechanical position $p^*\vartheta_{\text{mech}}$, where p is the number of pole pairs. Electrical speed is denoted by n. The rotor-oriented inductance matrix becomes:

$$\mathbf{x}_s^r = \begin{bmatrix} x_d & 0 \\ 0 & x_q \end{bmatrix},\ \mathbf{j} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},\ \underline{i}_s^r = [i_d \quad i_q]^T,\ \underline{\psi}_m^r = [\psi_m \quad 0]^T \tag{3}$$

2.2 Position and Speed Estimation Model

The position estimation adopts the Active Flux Observer presented in [12]. Accordingly, a quantity called 'active flux' ($\underline{\psi}^s{}_T$) is defined as follows.

$$\hat{\underline{\psi}}_T^s = \omega_n \cdot \int (\underline{u}_s^s - \hat{r}_s \underline{i}_s^s + \underline{u}_{comp}^s) \cdot dt - \hat{x}_q \cdot \underline{i}_s^s = \hat{\underline{\psi}}^s{}_{s,u} - \hat{x}_q \cdot \underline{i}_s^s \tag{4}$$

In this active flux observer structure, the current model and voltage model are employed as the reference and adaptive model respectively. Thus, the reference model is given as follows:

$$\hat{\underline{\psi}}^r{}_{s,i} = \mathbf{x}_s^r \cdot \underline{i}_s^r + \underline{\psi}_m^r \tag{5}$$

The adaptive model is given as follows:

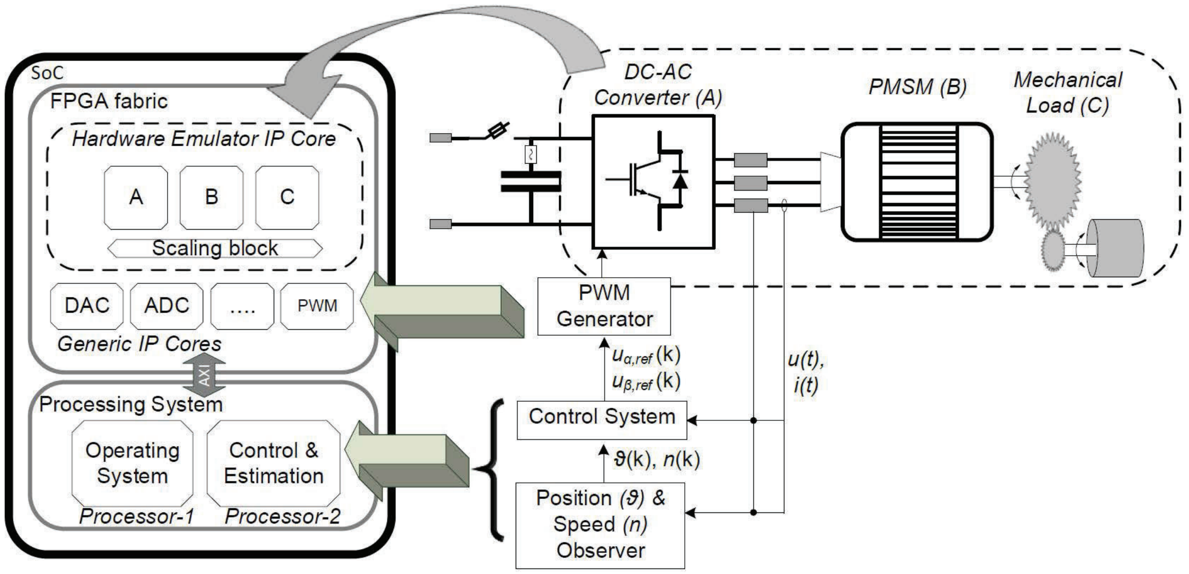

Fig. 1: Overview of the proposed Embedded Real-time Simulator

$$\underline{\hat{\psi}}^{s}_{s,u} = \omega_n \cdot \int (\underline{u}^{s}_{s} - \hat{r}_s \cdot \underline{i}^{s}_{s} + \underline{u}^{s}_{comp}) \cdot dt \tag{6}$$

From which, the error, $\underline{\varepsilon}_{s,o}$ is calculated and attempted to eliminate with the aid of a proportional-integral (PI) compensator.

$$\underline{\varepsilon}_{s,o} = \underline{\hat{\psi}}^{s}_{s,i} - \underline{\hat{\psi}}^{s}_{s,u} \tag{7}$$

Subsequently, by using (8), the estimated -rotor position and -speed can be calculated.

$$\begin{aligned} \vartheta &= \operatorname{atan2}\left(\frac{\hat{\psi}^{s}_{T,\beta}[k]}{\hat{\psi}^{s}_{T,\alpha}[k]}\right) \\ n &= \frac{\hat{\psi}^{s}_{T,\alpha}[k-1]\cdot\hat{\psi}^{s}_{T,\beta}[k] - \hat{\psi}^{s}_{T,\beta}[k-1]\cdot\hat{\psi}^{s}_{T,\alpha}[k]}{T_{samp}\cdot\left(\left(\hat{\psi}^{s}_{T,\alpha}[k]\right)^2 + \left(\hat{\psi}^{s}_{T,\beta}[k]\right)^2\right)} \end{aligned} \tag{8}$$

2.3 Drive Control System

The maximum torque per ampere (MTPA) strategy is applied in the drive control system. The general control block diagram of the IPMSM drive is given in the Fig. 2. The measured voltages and currents are fed into transformation blocks that use the estimated rotor position in their operations. The reference-currents are calculated based on (9). The output voltage can be either directly measured from the output of the drive or estimated from the measured dc-link voltage.

$$\begin{aligned} i_d^* &= \frac{\dfrac{\psi_m}{3} - \sqrt[3]{\left(\dfrac{\psi_m}{3}\right)^3 + \dfrac{\left(x_q - x_d\right)^2 \cdot \tau_{eref}^2}{3\cdot\psi_m}}}{x_q - x_d} \\ i_q^* &= \frac{\tau_{eref}}{\psi_m - \left(x_q - x_d\right)\cdot i_d} \end{aligned} \tag{9}$$

3 ERTS Overview and Architecture

3.1 Hardware

PicoZed7030 from Avnet is used as the System-on-Module (SoM) that contains the Zynq 7030 SoC. This SoC houses two units of ARM Cortex - A9 processors and an FPGA-section. An application specific carrier-board is used access the communication physical layers and input/output of the SoM. The carrier-board also contains a high-speed analog to digital converter (ADC) and digital to analog converter (DAC) -chips. See Fig. 10 for overview of the hardware.

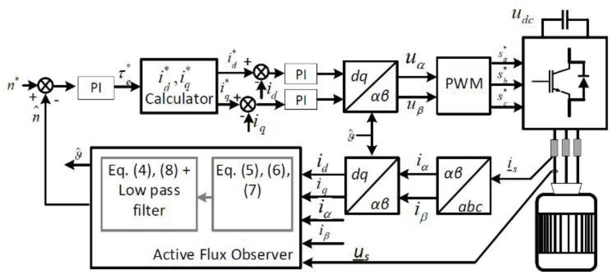

Fig. 2: Block diagram of position-sensorless, field oriented controlled IPMSM Drive

3.2 ERTS Architecture

The proposed ERTS, illustrated in the Fig. 1., contains three main components; 1) Application software program 2) Generic Intellectual Property-(IP) Cores (GIPC) 3) the Hardware Emulator, (HWE). This three-part ERTS allows individual development of each part and test with use of already validated remaining parts. Also, such partitioning maximizes the reusability of the common components, ensures uniformity across simulations, minimizes design and simulation failures, and help diagnose the faults or design errors rapidly. Also, by retaining much of the ERTS modularized, the proposed simulator can rapidly be modified to emulate different motor-types and designs, converters, or mechanical loads.

The proposed simulator is also designed using the per-unit system which makes its main components common across machines, loads and converters irrespective of their ratings.

3.3 Application Software

This part contains the per-unit scaling, reference frame transformations, MTPA and flux-weakening strategies, field-oriented control algorithms, reference voltage-vector calculations, the motor flux models, and active flux observer algorithms given in the (1) through (8), among other. These are programmed in the processor of the SoC, with the aim of achieving iterative programmability, ease of expandability and performance tuning with the help of a high-level programming language.

Application software program is modularized into 4 different layers as given in the Fig. 3, where states and references are cascaded only between the neighboring layers. To ensure modularity, and structure, C++ programming language is chosen

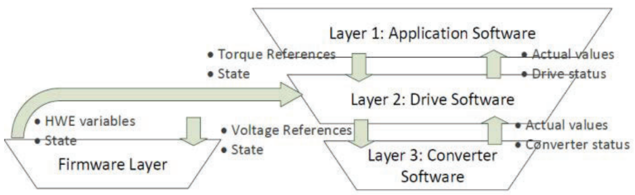

Fig. 3: Modularized architecture of the ERTS Application Software

aiming to exploit its object-oriented programming capabilities.

3.4 Generic IP Cores (GIPC)

This part of the ERTS contains a set of building blocks programmed on the FPGA fabric of the SoC and applicable for a wide range of electric drives and other power electronic applications. These include PWM and interrupt generation IP core, ADC-receiver, DAC, digital filters, and the fastest protection schemes. These IP cores are accessible to both the physical power hardware as well as the hardware emulation.

3.5 Hardware Emulator IP Core

This is the entity in the FPGA fabric that contains the digital replication of the power hardware of the electric drive. The emulator encompasses two units of 2-level VSCs to facilitate multiphase machines or separately excited machines, the rotating machine model, the mechanical load model, and the digital replication of the analog front end, known as the scaling block in this context. The emulator is designed in the per-unit system and it is parameterizable from the application software, thus this IP Core can remain unchanged when simulating different electric drives. The scaling block ensures that the generic IP cores will not see a difference between the physical hardware and the emulator when exchange of variables.

3.5.1 Programming Language

The Simulink library, Xilinx System Generator (XSG), a vendor specific schematic approach is chosen to program the hardware emulator in the FPGA fabric, which will eventually convert the schematics to a preferred hardware descriptive language and also generate drivers for the IP cores. The basic combinatory and sequential building blocks offered by the System Generator library within the Matlab/Simulink environment ease the FPGA programming of rather complex models [6]. Also, since the FPGA program is already in the Matlab environment with XSG, the program can be fairly quickly ported to other digital real-time simulators as such as OPAL-RT.

3.5.2 Numerical Representation

The AXI-interface in the SoC is 32-bit, therefore the ERTS word-length is kept at 32-bits for the convenience of data exchange with the on-chip processor. When real-time simulation is concerned, the 32-bit fixed-point representation guarantees higher numerical precision and demand less internal resources over the 32-bit floating point representation [6], [8], [13].

The per-unit scaling of the emulator further enhances the numerical precision such that its rated voltages and currents become 1 unit, thus the range is minimized to maximize the number of precision bits of the 32-bit fixed-point format. Applied precision is 32.28 which means, 28 bits are retained for precision having a resolution of 3.725×10^{-9}. While leaving the most significant bit to represent the sign, the chosen precision offers a range between +7.99 to -8 which is sufficient when per-unit system is concerned.

3.6 ERTS Clock Settings

Three different clock cycles, as tabulated in the Table 1, are utilized to represent the physical motor drive system as close as possible in the ERTS. The subcomponents that require fastest processing speeds like the GIPC and converter block in the emulator take advantage of the FPGA clock. The processor-interrupt that is twice the speed of the PWM sets the processor interrupt cycle, which is sufficient for the application software. Solver-clock is used in the solver of the discretized rotating machine and mechanical load models in the hardware emulation. The ratio between the T_{step} to the mechanical time constant determines the stability of the discrete system, which is preferred to be as small as possible. T_{step} will also determine the integration intervals, thus the shorter integration intervals will yield more precise results. Instead of applying the fast FPGA-clock, a much slower solver clock is used to slow down the processing speed of the rotating machine- and the mechanical load- realizations in the FPGA fabric with the aim of emulating their real-world sluggishness in the digital hardware. Such processing speed reduction without compromising the numerical precision and the stability will help reduce the power consumption and heat generation from the FPGA of the SoC.

Table 1: Clock settings in the proposed ERTS

Clock	Frequency	Time step
FPGA Clock	100 MHz	T_{FPGA} = 10 ns
Processor interrupt cycle	8 kHz	T_{samp} = 125 μs
Solver clock	1 MHz	T_{step} = 1 μs

3.7 Discretization

In codifying the emulation models in the FPGA fabric, Euler method (10) is applied as this method is sufficient in the precision.

$$s = \frac{z-1}{T_{step}} \tag{10}$$

When control compensation schemes are concerned, it is mainly the proportional-integral (PI) compensation is applied, and they are implemented in the processor software using the trapezoidal method as given below, due to the ease of implementation and reasonable performance.

$$s = \frac{2}{T_{samp}}\left(\frac{1+z^{-1}}{1-z^{-1}}\right) \tag{11}$$

4 Hardware Emulator Development

4.1 Overview

The overview of the Hardware Emulator IP core is as illustrated in Fig. 4. Apart from the motor model, its remaining subcomponents can be ported across different AC- or DC- motor types. The different colors of the arrows interpret where the respective data is carried to/from: black arrows to/from the processor; gray arrows within the same IP Core; purple to/from the other IP cores. The blue arrowed data can be viewed from an oscilloscope using the DAC. The development of main subcomponents is unveiled subsequently where some of the subcircuits are omitted to simplify the illustration. Pre/post -transformation blocks in the Fig. 4 are the digital implementation of Park- and Clarke- transformations which are not illustrated due to space restrictions.

4.2 Converter Implementation

The voltage source converters are modelled using the switching function model [14] and its schematic is presented in the Fig. 5. Accordingly, the gate signals are supplied from the modulator IP core which resides outside the hardware emulator IP core, yet inside the FPGA fabric.

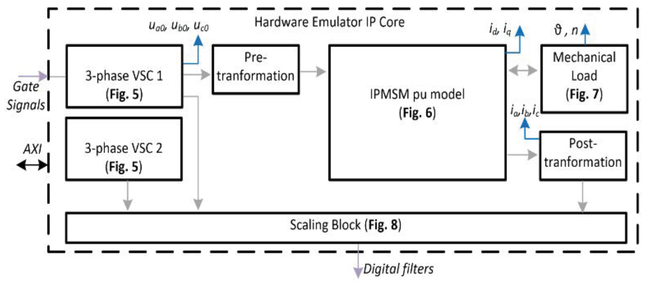

Fig. 4: Overview of the Hardware Emulation IP Core

The converter logic is as follows: during the deadtime, if the current is positive, choose the ground DC-rail. In the same instance, if the current is negative, choose the positive DC-rail. During the other times, choose the gate signal command.

4.3 IPMSM Implementation

The schematic is found in the Fig. 6 in which the motor model assumes sinusoidal flux distribution in the stator, no cross-coupling between the d- and q- axes, thus no secondary saliencies apart from the rotor-geometric saliency.

IPMSM model, when discretized, becomes:

$$\begin{aligned} i_d[k+1] &= i_d[k]\cdot\left[1 - T_{step}\cdot\frac{\omega_n\cdot r_s}{x_d}\right] + \\ &\quad T_{step}\cdot\left(\frac{\omega_n}{x_d}\right)\cdot\left[u_d[k] + n[k]\cdot x_q\cdot i_q[k]\right] \\ i_q[k+1] &= i_q[k]\cdot\left[1 - T_{step}\cdot\frac{\omega_n\cdot r_s}{x_q}\right] + \\ &\quad T_{step}\cdot\left(\frac{\omega_n}{x_q}\right)\cdot\left[u_q[k] - n[k]\cdot x_d\cdot i_d[k] - n[k]\cdot\psi_m\right] \end{aligned} \tag{12}$$

The electrical parameters and constants are passed from the application software program during the initialization of simulator. In principle, any of the intermediate- or boundary- digital signals can be passed to the processor or DAC to be monitored using an oscilloscope.

4.4 Mechanical Load Implementation

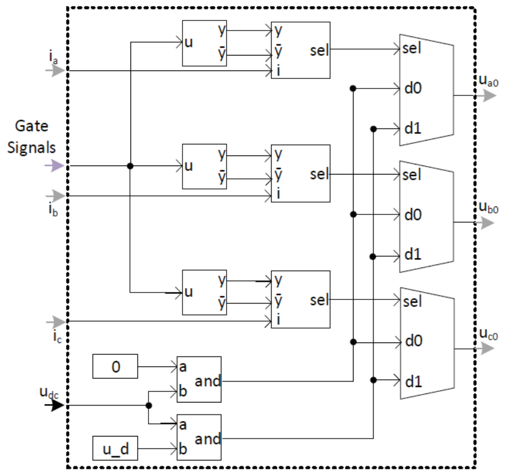

Fig. 5: Schematic of voltage source converter implementation within the hardware emulator IP core

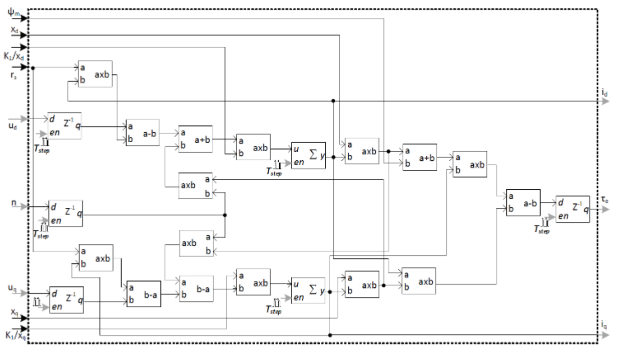

Fig. 6: Schematic of the IPMSM per-unit model implementation within the hardware emulator IP core

The mechanical load is modelled as follows and it is illustrated in the Fig. 7

$$n[k+1] = n[k] + \frac{T_{step}}{T_m} \cdot \left[\tau_e[k] - \tau_L[k]\right] \qquad (13)$$

$$\tau_L = k_n \cdot sign(n) \cdot n^2 + \tau_{L,extern} \qquad T_m = \frac{J \cdot \Omega^2}{S_n}$$

4.5 Scaling Block

The scaling block as in the Fig. 8 ensures the currents and voltages from the emulator passed to the GIPC (for digital filtering and multiplexing) contain the identical volts-per-bit and amperes-per-bit resolution of those obtained from the physical hardware. Owing to this identicality, neither the GIPC nor the application software will notice a difference between the hardware emulator and physical hardware.

With respect to the gains of each blocks given in the Fig. 8, one can find the gains for the scaling block using (14) which is common for both voltages and currents. Since the emulator is designed in per-unit scale, but the GIPC is scaled for SI values,

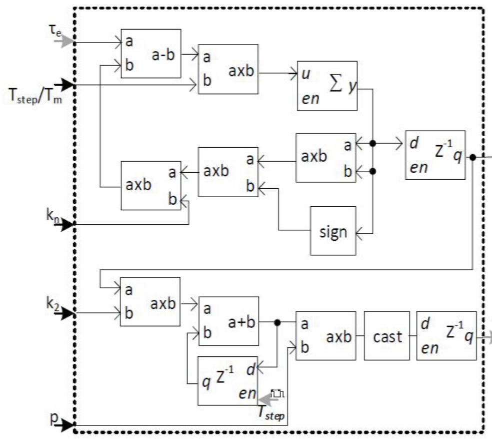

Fig. 7: Schematic of mechanical load model implementation within the hardware emulator IP

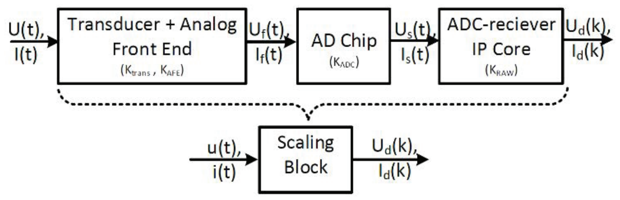

Fig. 8. Scaling block within the emulator IP Core

the *variable$_{pu}$* i.e. *u(t), i(t)* must be multiplied by the base values (K_{base}) in addition as seen in the formula.

$$variable\_per\_bit = variable_{pu} \cdot \frac{K_{AFE} \cdot K_{base}}{K_{trans} \cdot K_{RAW}} \qquad (14)$$

4.6 Development Procedure

The Matlab development environment can be exploited not only to design the hardware emulation models, but also to identify the satisfactory fixed-point precisions, ranges and test the closed-loop performance using the floating-point drive controller. As shown in the Fig. 9, both the Xilinx System Generator model as well as the Simulink models can be run in parallel to identify the numerical errors, thus finetune the fixed-point precisions. Once the models are ready for synthesis, they can be exported to the Vivado integrated development environment where the final synthesis and bitstream generation is performed. Table 2 presents the internal FPGA-resources utilized by the hardware emulator. The bitstream is then linked to the Xilinx Software Development Kit (SDK) for target-oriented application software and firmware development.

Table 2: Resource utilization of implemented emulator

Resource	Used	% of the Total
Slice Look-Up Tables	4937	6.3%
Slice registers	5634	3.6%
Block RAMs	0	0% (total 265)
Block DSPs	0	0% (total 400)

5 Real-Time Simulation

The proposed ERTS is experimented against the offline simulation (Matlab/Simulink R2019a) model for the IPMSM drive with use of simulation data tabulated in the Table 3. The offline simulation model utilizes the built-in ode23tb solver with variable time step. The current controllers have been tuned with use of the same control parameters in both the simulators.

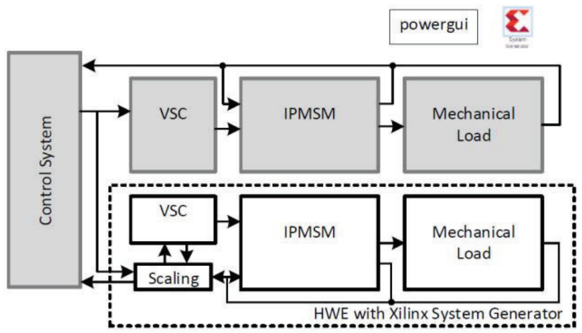

Fig. 9: IPMSM drive co-simulation: Simulink floating point double precision model (gray blocks) in parallel with XSG fixed-point, 32.28-precision model (white blocks) using common floating-point control logic.

Table 3. Electric drive data for simulations

	Value	**Unit**
Nominal voltage, U_N	220	V
Nominal current, I_N	51	A
Rated frequency, f_N	35	Hz
Pole pairs, p	1	-
Nominal speed, N	2100	rpm
Motor parameter vector $[\psi_m\ x_d\ x_q\ r_s]^T$	$[0.66\ 0.4\ 1\ 0.009]^T$	pu
Switching frequency, f_{sw}	4	kHz

Fig. 10 presents the experimentation setup of the ERTS. The PC is used for programming/debugging while the oscilloscope is used to monitor high bandwidth real-time variables. In general, the variables of the ERTS can be exported to plot in the Matlab environment as it is seen in the next section.

5.1 ERTS Performance

The ERTS performance is analyzed with respect to the execution time of a given simulation run and the numerical precision. For a 1-second simulation run of the IPMSM drive, the ERTS consumed just the run-time of 1 second while, the offline method took ~550 seconds which means a remarkable time saving with the ERTS. The numerical precision of the ERTS is evaluated using the root-mean-square (rms) error with reference to the offline simulation variable (with double precision floating-point). Fig. 11 presents the motor d- and q-axes currents processed through a moving average filter in both simulators, when the reference torque receives a step command from 0 to 0.8 pu. The corresponding transient errors are plotted in the Fig. 12. For the same scenario, the steady-state rms errors for d- and q- axes currents were calculated to be 7.16x10^{-4} and 3.67x10^{-4} respectively in per-unit. Fig. 13 presents the corresponding torque-step and response. Fig. 14 illustrates the rotor position from the sensorless algorithm and the motor model in the hardware emulation, using the DAC-features of the ERTS.

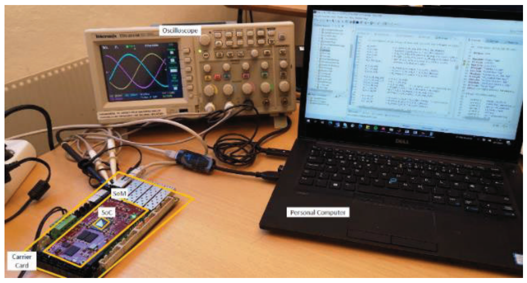

Fig. 10: Embedded Real-Time Simulator Test Setup

5.2 ERTS Limitations

Some of the notable limitations of the proposed ERTS with respect to a physical electric drive: the VSC is modeled using the switching function model in the hardware emulator, which may not accurately represent the converter during the regenerative modes [5]. The emulated motor model, too, does not entail the secondary

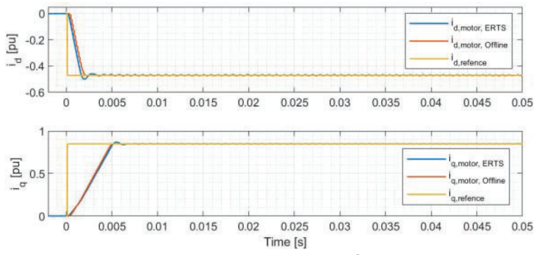

Fig. 11: d- and q- axes currents of the motor and their respective references in using the two simulators

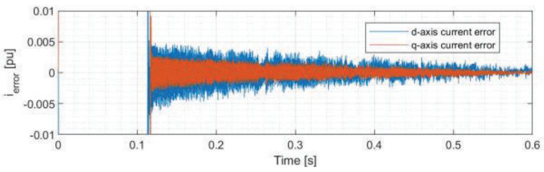

Fig. 12: Transient current -errors of the real-time simulator with respect to the offline simulator

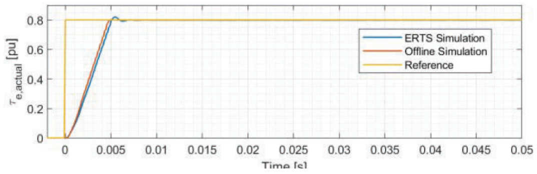

Fig. 13: Actual torque -response from the ERTS and the offline simulator for a step reference change

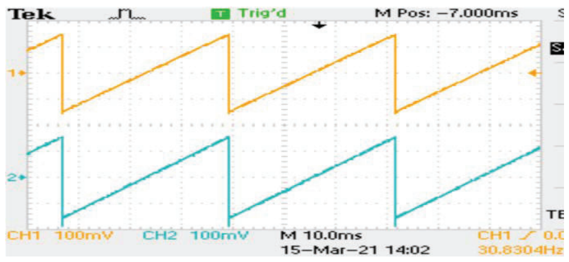

Fig. 14: Rotor position from: the hardware emulation (blue), the estimation using the observer (yellow)

saliencies nor the saturation effects which limits its performance precision in certain operating ranges.

6 Conclusion

An inexpensive, modular, portable, and rapidly reusable Embedded Real-Time Simulator targeted for electric motor drives has been presented in this paper. Both the software and hardware architectures of the simulator and the detailed design method of the hardware emulation have been discussed. The internal resource utilization of the FPGA is kept at a minimum in designing the real-time emulation. The overall simulation is significantly accelerated in the given sensorless control application at the expense of negligible numerical error. The simulator sets the basis to expand its model-behavior as close as possible to the real-world systems and to include diagnosis and prognosis of faults in the drives to develop towards a real-time digital twin.

7 References:

[1] M. Dagbagi, A. Hemdani, L. Idkhajine, M. W. Naouar, E. Monmasson, and I. Slama-Belkhodja, "ADC-Based Embedded Real-Time Simulator of a Power Converter Implemented in a Low-Cost FPGA: Application to a Fault-Tolerant Control of a Grid-Connected Voltage-Source Rectifier," *IEEE Trans. Ind. Electron.*, vol. 63, no. 2, pp. 1179–1190, 2016.

[2] A. Sanchez, A. De Castro, and J. Garrido, "A comparison of simulation and hardware-in-the-loop alternatives for digital control of power converters," *IEEE Trans. Ind. Informatics*, vol. 8, no. 3, pp. 491–500, 2012.

[3] M. D. Omar Faruque *et al.*, "Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis," *IEEE Power Energy Technol. Syst. J.*, vol. 2, no. 2, pp. 63–73, 2015.

[4] L. Crockett, R. Elliot, M. Enderwitz, and R. Stewart, *The Zynq Book*, 1st ed. Glasgow: Strathclyde Academic Media, 2014.

[5] G. G. Parma and V. Dinavahi, "Real-time digital hardware simulation of power electronics and drives," *IEEE Trans. Power Deliv.*, vol. 22, no. 2, pp. 1235–1246, 2007.

[6] N. Roshandel Tavana and V. Dinavahi, "A General Framework for FPGA-Based Real-Time Emulation of Electrical Machines for HIL Applications," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2041–2053, 2015.

[7] D. Tormo, L. Idkhajine, E. Monmasson, and R. Blasco-Gimenez, "Evaluation of SoC-based embedded Real-Time simulators for electromechanical systems," *IECON Proc. (Industrial Electron. Conf.*, pp. 4772–4777, 2016.

[8] D. Tormo, L. Idkhajine, E. Monmasson, V.-A. Ricardo, and R. Blasco-Gimenez, "Embedded real-time simulators for electromechanical and power electronic systems using system-on-chip devices," *Math. Comput. Simul.*, no. 158, pp. 326–343, 2019.

[9] M. Ricco, M. Gheorghe, L. Mathe, and R. Teodorescu, "System-on-Chip Implementation of Embedded Real-Time Simulator for Modular Multilevel Converters," in *ECCE 2017- IEEE Energy Conversion Congress and Exposition, Proceedings*, 2017, pp. 1500–1505.

[10] D. Mohammadi, L. Daoud, N. Rafla, and S. Ahmed-Zaid, "Zynq-based SoC implementation of an induction machine control algorithm," *Midwest Symp. Circuits Syst.*, vol. 0, no. October, pp. 16–19, 2016.

[11] E. D. M. Fernandes, D. R. Huller, A. C. Oliveira, and W. R. N. Santos, "Real-time simulator of interior permanent-magnet synchronous motor based on FPGA devices," in *Brazilian Power Electronic Conference (COBEP)*, 2017, pp. 11–16.

[12] M. C. Paicu, I. Boldea, G.-D. Andreescu, and F. Blaabjerg, "Very low speed performance of active flux based sensorless control: interior permanent magnet synchronous motor vector control versus direct torque and flux control," *IET Electr. Power Appl.*, vol. 3, no. 6, p. 551, 2009.

[13] A. Sanchez, E. Todorovich, and A. de Castro, "Exploring the limits of floating-point resolution for hardware-in-the-loop implemented with fpgas," *Electron.*, vol. 7, no. 10, pp. 1–12, 2018.

[14] H. Jin, "Behavior-Mode Simulation of Power Electronic Circuits," *IEEE Trans. POWER Electron.*, vol. 12, no. 3, pp. 1149–1154, 1997.