FPGA-Based Real-Time Simulation of Sensorless Control of PMSM Drive at Standstill

A. Darba, F. De Belie, T. Vyncke and J. Melkebeek, *Senior Member, IEEE*

***Abstract*—This paper presents a real-time simulation of a sensorless control method for IPMSM drives based on Park Transformation with a reference frame fixed to the rotor and assuming a sinusoidal flux (sinusoidal back-emf). The objective of this paper is to Hardware In Loop (HIL) evaluation of a sensorless position estimation of the Permanent Magnet Synchronous Motor (PMSM) drive at standstill as well as the current controller. An anti-windup method is integrated within the controller to ensure a good dynamic performance during transients. Test voltages vectors are injected in such a manner the current samples are not affected. An asymmetric Pulse Width Modulation (PWM) allows to apply these test vectors each PWM period.**

Motor model and controller are implemented on a Field-Programmable Gate Array (FPGA) evaluation board for Hardware In Loop testing of motor drive and controllers including the sensorless control method. Programming is done in Simulink by using a blockset called Xilinx System Generator (XSG), without any low level VHDL coding.

The paper explains various aspects of the design of the motor model in fixed-point representation, as well as actual simulation validations against a standard PMSM model built in Simulink.

The motor model, current controller, anti-windup compensator and sensorless position estimators are implemented inside a Xilinx Spartan 3E XC3S500E board. The PMSM drive model runs with 25 ns time steps (50 MHz FPGA card) which calculates the states every 2.5 us and holds them to the next calculation moment.

***Index Terms*—permanent-magnet Synchronous Motor (PMSM), field-programmable gate arrays (FPGAs), Real-time Simulation, Sensorless Control**

I. Introduction

The trend in modern drives is to replace noise sensitive and less reliable sensors with computational algorithms that use current or voltage measurements. These current and voltage parameters are indeed already available for the control of the drive or the protection of the inverter. Moreover, instead of actual voltage measurements reference values can sometimes be used. Removing the position sensor yields remarkable savings in production, installation and maintenance and improves the reliability. As variable speed drives are supplied by PWM, the voltage pulses can be modified easily and the resultant current ripple can be used to estimate the rotor position. In most power converters, the phase currents are controlled in discrete-time and sampled at a fixed frequency [1]–[4]. By injecting high-frequency test voltages, these current samples can be distorted, making it necessary to lower the controller bandwidth and consequently the dynamic behaviour of the current controller may deteriorate.

A. Darba "Email: araz.darba@ugent.be", F. De Belie "Email: frederik.debelie@ugent.be" and J. Melkebeek "Email: jan.melkebeek@ugent.be" are with the Department of Electrical Energy, Systems and Automation of Ghent University, St-Pietersnieuwstraat 41, B-9000 Gent, Belgium.

The competitive market of these days requires a minimum time of design-to-market. The design for controllers and estimators for high efficiency motor drives is more and more done in parallel with the design and prototyping of the inverter and the motor. Achieving these goals can be supported by using real-time models of the motor instead of using a prototype motor to test the controllers. The main advantages of this method are:

- modifying parameters in an easy and flexible manner
- accelerating production time of a design by designing the controllers and estimators in parallel with motor
- testing the simulated motor drive under extreme condition which would damage the expensive prototype motor
- doing experimental tests requires direct measurements and sensors and instruments that would be complex and noise sensitive

To implement an hardware in loop platform the hardware that one needs must be fast enough (in our case we update all control strategies and motor model each 2.5 us). Moreover trying to reach the optimum point of cost and technology pushes us to use a device that can be able of doing many parallel calculations at the same time. DSPs are fast but it's necessary to do all the calculations sequentially. If someone wants to build a real time emulator it's not impossible with DSP but it needs a DSP with very fast clock. programmable devices, such as programmable logic arrays (PLAs), have been available since the 1970s but their application were limited. More possibilities are offered by the Field-Programmable Gate Array (FPGA) concept, introduced by Xilinx' cofounder Freeman in 1984 [5]. A FPGA is a matrix of configurable logic blocks (CLBs), linked to each other by an interconnection network which is entirely reprogrammable. In the recent years more powerful and less expensive hardware in FPGA's has been developed. Also advances in programming softwares (e.g. Xilinx' System Generator) allowed the design to become more and more independent of hardware description language. Model Based Predictive Control (MBPC) is one of the areas which FPGA's are playing an important role to prove this methods to be true in reality. Parallelism and reuse of resources features of FPGA's provide the demand of massive amount of calculations in short period of time [6], [7]. For industrial control systems FPGA's are quite interesting because of the ever-increasing level of expected performance while they at the same time reduce the development costs [8]–[14] . Some new applications like parameter estimation and senslorless drive control require FPGA performance [15], [16].

Compared to DSP's, FPGA's offer many advantages: computational speeds are much faster by using parallel computational structures (e.g. in [17]). The freedom to

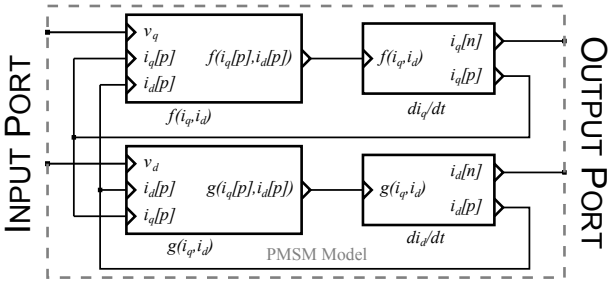

Fig. 1. Schematic view of FPGA implementation of discrete-time PMSM model

customize the individual signal and coefficient formats can be exploited for overcoming the numerical difficulties caused by finite-word-length effects [9], [13]. Which means, the possibility to obtain a higher precision of data formats at critical points in the calculation and a smaller precision at other points which is fixed in DSP's.

II. Motor Model

The continuous-time mathematical equations of the IPMSM in the synchronous qd-reference frame are:

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t} i_q = \frac{1}{L_q} v_q - \frac{R}{L_q} i_q - \omega \frac{L_d}{L_q} i_d \\ \frac{\mathrm{d}}{\mathrm{d}t} i_d = \frac{1}{L_d} v_d - \frac{R}{L_d} i_d + \omega \frac{L_q}{L_d} i_q \end{cases} \tag{1}$$

For our purpose a discrete-time model is required. The system of continuous time differential equations (1) are coupled, making it time-consuming to compute the solution in each PWM-period analytically. This system demonstrates our technique for the FPGA model development. In (1), the left-hand side is the differential term and the right-hand side is denoted as a term which defines how two differential term are related together. Functions f and g are defined as follows:

$$\begin{cases} f\left(i_q, i_d\right) = \frac{1}{L_q} v_q - \frac{R}{L_q} i_q - \omega \frac{L_d}{L_q} i_d \\ g\left(i_q, i_d\right) = \frac{1}{L_d} v_d - \frac{R}{L_d} i_d + \omega \frac{L_q}{L_d} i_q \end{cases} \tag{2}$$

Defining functions f and g in (2) which they have both of the variables i_q, i_d has an advantage: this makes delineation between the differential term and the intermediate clear for our discrete-time integration method. Each ODE in the continuous time domain must be mapped to the discrete-time domain. By using the forward-Euler integration, the following discrete-time representation can be obtained where n is the iteration step:

$$\begin{cases} i_q\left[n+1\right] = i_q\left[n\right] + \Delta t \left(\frac{1}{L_q} v_q - \frac{R}{L_q} i_q\left[n\right] - \omega \frac{L_d}{L_q} i_d\left[n\right] \right) \\ i_d\left[n+1\right] = i_d\left[n\right] + \Delta t \left(\frac{1}{L_d} v_d - \frac{R}{L_d} i_d\left[n\right] + \omega \frac{L_q}{L_d} i_q\left[n\right] \right) \end{cases} \tag{3}$$

These equations are not optimized with respect to the processing time in an FPGA where each operation should be carried out in the shortest time possible. In addition, parallelism is a main feature of an FPGA. The arithmetic operations should be reordered to reach the parallelism based on each operator latency. Considering $\frac{\mathrm{d}}{\mathrm{d}t} i_q = f\left(i_q, i_d\right)$ and $\frac{\mathrm{d}}{\mathrm{d}t} i_d = g\left(i_q, i_d\right)$ and replacing (2) in equations (3) results in:

$$\begin{cases} i_q\left[n+1\right] = i_q\left[n\right] + \Delta t.f\left(i_q\left[n\right], i_d\left[n\right]\right) \\ i_d\left[n+1\right] = i_d\left[n\right] + \Delta t.g\left(i_q\left[n\right], i_d\left[n\right]\right) \end{cases} \tag{4}$$

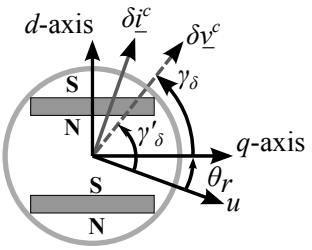

Fig. 2. Supplied voltage deviation $\delta \underline{v}^c$ and current response $\delta \underline{i}^c$ in the qd-reference frame fixed to the rotor.

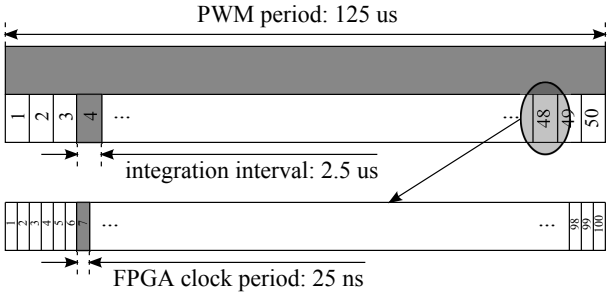

Fig. 3. Comparison of PWM period vs. integration and FPGA clock period

Equation (4) can be easily implemented in an FPGA considering parallelism and operator latency, either using low-level languages (e.g. VHDL) or in a high-level of abstraction (System Generator in Simulink). A simplified block diagram is illustrated in Fig. 1 where $[n]$ is the count of the outer and slower clock and $[p]$ refers to the count of the inner and faster clock. The fast (inner) clock is the actual clock cycle of the FPGA core 20 ns and slow (outer) clock cycle is 2.5 us which is 100 times slower than inner clock. Motor model calculates the output current of motor 50 times in each PWM period. Fig. 3 illustrates different clock cycles.

III. Rotor Position Estimation

In this section, the mathematical basics of the sensorless control scheme which are digitally implemented in FPGA is explained. It is assumed that the inductances L_q, L_d in the real-time implementations are invariable. In this paper, in order to estimate the rotor position at low speed and standstill condition a measurement of high frequency voltage injection current response is used. High frequency voltages are called test signals and can be pulsed or sinusoidal [3]. These test signals are added to the output of the current controller. By sampling the current ripple, the rotor position can be estimated due to the dependence of current response to the supply voltage deviations on the rotor position. Test signals are added in such a way a zero stator flux deviation at the current sampling moments is obtained. An asymmetric PWM strategy is considered to implement these test signals each PSM period.

As shown in [1] the current response caused by the voltage test pulses can be approximated as follows:

$$\delta i_q^c = \frac{\tau}{L_q} \delta v_q^c \quad , \quad \delta i_d^c = \frac{\tau}{L_d} \delta v_d^c \tag{5}$$

where δi_q^c, δi_d^c are the resulting motor current variations and δv_q^c, δv_d^c are qd-voltage deviations from steady-state voltage. To estimate the rotor position θ_r, an auxiliary angle γ_δ is used which is the angle of voltage vector $\delta v^c = \sqrt{\delta v_q^c + \delta v_d^c}$

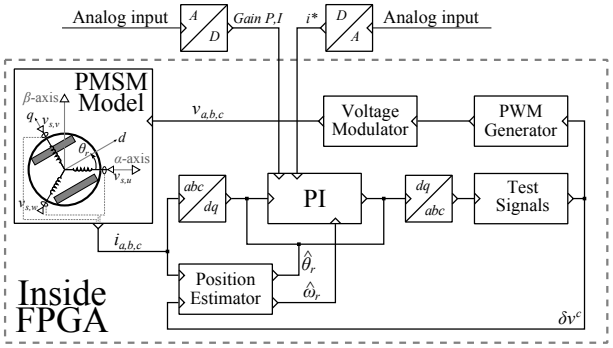

Fig. 4. Schematic overview of emulator.

with respect to the q-axis as is shown in Fig. 2. The current responses in the xy-coordinate system can be calculated by:

$$\begin{cases} \delta i_x^c & = \tau \delta v^c \left(\frac{L_q + L_d}{2L_q L_d} - \frac{L_q - L_d}{2L_q L_d} \cos\left(2\gamma_\delta\right) \right) \\ \delta i_y^c & = \tau \delta v^c \frac{L_q - L_d}{2L_q L_d} \sin\left(2\gamma_\delta\right) \end{cases} \tag{6}$$

Using equations (6) the qd-inductances (Lq, Ld) are required to calculate δi_x^c, δi_y^c. However, there exists a parameterless method to calculate δi_x^c, δi_y^c [4]. In this method two sequential high frequency test voltages must be applied for each rotor position estimation. The current responses $\delta \underline{i}_{xy,I}^c \left(\delta i_{x,I}^c, \delta i_{y,I}^c\right)$ of the first test vector correspond to equations (6). The current responses $\delta \underline{v}_{II}^c$ of the second test vector correspond to equations (7). Hereby the modulus of the second voltage test vector is equal to the first one but its angle is displaced by $\Delta\gamma_\delta$. a second current response is obtained:

$$\begin{cases} \delta i_{x,II}^c & = \tau \delta v^c \left(\frac{L_q + L_d}{2L_q L_d} - \frac{L_q - L_d}{2L_q L_d} \cos\left(2\gamma_\delta + 2\Delta\gamma_\delta\right) \right) \\ \delta i_{y,II}^c & = \tau \delta v^c \frac{L_q - L_d}{2L_q L_d} \sin\left(2\gamma_\delta + 2\Delta\gamma_\delta\right) \end{cases} \tag{7}$$

To estimate the auxiliary angle γ_δ in a parameterless way, the current responses are then subtracted. From this difference, using (6) and (7) the angle γ_δ can be estimated by:

$$\hat{\gamma_\delta} = \frac{1}{2} \arctan\left(\frac{\delta i_{y,I}^c - \delta i_{y,II}^c}{\delta i_{x,II}^c - \delta i_{x,I}^c} - \frac{2\Delta\gamma_\delta + \pi}{4} \right) \tag{8}$$

IV. Digital Current Controller

The current components are controlled in order to track torque or flux reference inputs. Fig. 4 shows the schematic view of the drive system emulator, including position estimator, current controller and motor model. The rotor position θ_r is used to transform the currents measured in the abc frame to the qd-reference frame fixed to the rotor. The rotor speed ω_r is required to remove the mutual interaction between the perpendicular magnetic axes as a result of the rotor speed induced voltage. A 3-phase voltage modulator is used to generate the voltage pulses for the supply of the PMSM. The PWM generator block provides the gate driver logic commands for the switches. To obtain approximately average values of the currents, the samples are taken in the middle of the PWM period. During the test periods additional measurements will be necessary. This results in a variable sampling frequency of phase current measurement. With this variable sampling frequency digital current controller then will be difficult to tune and implement. However in this paper, a discrete-time current controller with a fixed sampling frequency, even during a test period, is used. To obtain this goal, an asymmetric PWM strategy is needed. This strategy is described in detail in [3]. In this method, the duty ratio can be changed each half PWM-period. As the current is sampled each PWM-period, the controller output is updated each PWM-period. The currents deviations δi^c used in the position estimator for the purpose of the sensorless control are measured by sampling the phase currents at the start and end of each half PWM-period during a test period.

V. Implementation Methodology and Results

Continuous-time simulations can be done in the Matlab/Simulink software. However, in order to use the results for an FPGA, we need a completely discrete low-level software. System Generator is an add-on top-level software which is added to Simulink. In that way, a user can do all simulations in a fully discrete-time environment using System Generator blocks and user defined modules which is very similar to the normal Simulink blocks. After doing all simulations in System Generator environment it's possible to compile the top-level Netlist file directly to a low-level bitstream code and this code can be used to program the FPGA.

A. Methodology

To implement a whole simulation on a FPGA board we have to use some techniques such as reusing the resources considering parallelism and optimization of the modules in terms of performance with the help of the Algorithm Architecture Adequation [18] which is called A^3. A^3 methodology is developed in order to help the designer obtain an efficient implementation. This implementation satisfies real-time constraints and minimizes the architecture size of complex algorithms and simplifies the implementation task from the specification to the final prototype.

For a digital control algorithm which is used once every PWM-period (half PWM-period) e.g. 125 us (62.5 us) we use a FPGA system period 25 ns. The model of the motor uses a discrete-time integrator implemented only by adders delays and one register for each d and q component. The sampling time for integrator is set to 2.5 us which is 50 times smaller than PWM period. To obtain correct results, specially in the feedback path, timing is important in the implementation of the discrete integrator. Also delays should be chosen carefully in the core of the integrator.

As mentioned before, the control algorithm requires the motor output once in each PWM-period or half of it. In order to obtain a more efficient implementation the motor model block is enabled once each 2.5 us (equal to the integrator sampling time) which is 125 times slower than maximum FPGA clock frequency and 100 times slower than our working frequency. By using this integration method the motor model needs less than one period of each FPGA clock cycle (working with Spartan 3E XC3S500E board results in 12.76 ns calculation time) to calculate its output values. This means we have enough time to do more complicated calculations such as current control with anti-windup and position estimation in the rest of the period. If we don't use a motor model inside the FPGA and if we connect it to a real motor, then we should consider 2-3 us for the AD conversion process time. This is approximately 200 times longer than the time needed for model calculation.

TABLE I
CLASSIFICATION OF DIFFERENT MODULE LEVELS IN FPGA PROGRAMMING

Level	Module Name
3	Full Algorithms (Current, Torque, Speed Control, ...)
2	Regulation (PI, PID, Hysteresis Controllers, ...) Modulation (PMM, SVPWM,...) Estimation (PLL, Torque, Flux, Position Estimators, ...) Vector Operators (abc-to-dq, abc-to-$\alpha\beta$, ...)
1	Basic Operators (Registers, Multiplexers, ...) Arithmetic Operators (Adder, Multiplier, Sine-Cosine, ...)

TABLE II
THEORETICALLY NECESSARY RESOURCES FOR USE OF ONE INDIVIDUAL FOR EACH PARK AND INVERSE PARK TRANSFORMATION

Resources needed for one individual Park Transformation block Execution time: 6 ns				
Slices: 1280	FFs: 436	LUTs: 2121	Emb. Mults: 16	TBUFs: 1
Resources needed for one individual Inverse Park Transformation block Execution time: 17.45 ns				
Slices: 2645	FFs: 612	LUTs: 4993	Emb. Mults: 16	TBUFs: 1
Total resources needed for doing all transformations Percentage of needed resources for Xilinx Spartan 3E XC3S500E				
Slices: 13055 281%	FFs: 3580 38%	LUTs: 23463 252%	Emb. Mults: 112 560%	TBUFs: 2 24%

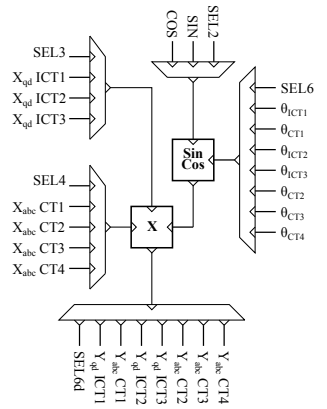

Fig. 5. Signal flow for Park and Inverse Park Transformations using one block

B. Reusing of Resources

The reusability feature is applicable to the motor model as well as current controller by using the same set of modules and blocks to do the specified job for both the d and q components. This is a result of the very small calculation time in the integration part of the motor model and in the current controller block. One of the most important features of the FPGA boards which is used in this study is parallelism. The simulation model includes six second level modules which are: Motor Model, Current Controller, Gate drive, PWM Modulator, Current measurement and Position Estimator. All these modules can work together in a parallel manner using separate resources of the FPGA. The classification of modules in three levels is shown in Table I.

The most demanding part of this simulation, in resource as well as time, is the calculation of Park and inverse Park Transformations. There are four Park and Three Inverse Park transformation blocks in our simulations and for each of them it was necessary to have the resources as mentioned in Table II.

A method to overcome this huge amount of calculations and required resources is to reuse a common modules for calculating sine-cosine for all Park and inverse Park Transformation. Moreover we can asign a single block to do all multiplications. Applying this method requires special attention to the timing as shown in Fig. 5. CORDIC is an acronym for COordinate Rotation Digital Computer [19]. CORDIC is only based on adders/subtractors and shifters for computing a wide range of trigonometric, hyperbolic, linear and logarithmic functions. In this paper we used CORDIC functions to calculate Sin, Cos, which are used in Park and inverse Park Transformations and ATAN2 (ArcTangens) to estimate the position. The best choice for having accurate position estimation and acceptable output for Park and inverse Park transformations block were 11 and 7 CORDIC steps for ATAN2 and Sinus/Cosine functions respectively. One iteration of sine-cosine calculations procedure for all Park and inverse Park transformations as well as their timing process, is explained in the following paragraph. There are four Park and three inverse Park transformation blocks in our simulation program. These transformation blocks are distinguished by CT1,...,CT4 and ICT1,...,ICT3 for Park and inverse Park transformations respectively. X_{abc}, X_{qd}, are the input quantities of the blocks and Y_{abc}, Y_{qd} are output quantities, with respect to the Park or inverse Park transformation. Each individual transformation block needs a selector signal to enable it at specific moment after its inputs are generated by prior blocks. Five general selection signals are defined and each one drives a multiplexer. This means that each block generates its output continuously but its output chosen by a multiplexer in specific moments. The selection signals are as follows:

SEL2 sets CORDIC sine-cosine block to calculate sine or cosine of the input signal and SEL2 changes between 0, 1.

SEL3 enables one of the inverse Park transformation blocks at their specified working period and SEL3$= 0, 1, 2$.

SEL4 enables one of the Park transformation blocks at their specified working period and SEL4$= 0, \ldots, 3$.

SEL6 is a signal which controls a multiplexer to select one input of CORDIC sine-cosine block each time. These inputs are synchronized to work with SEL3, SEL4 in order to calculate sine and cosine of the signals within Park and inverse Park transformations. This means, the CORDIC sine-cosine block does its job for each transformation block by changing value of the SEL6 signal and SEL6$= 0, \ldots, 6$.

SEL6d drives a multiplexer to specify the output of the multiplier which gives an output related to one of the transformation blocks each time. This means, output of the multiplier changes between outputs of the each transformation block by changing value of the SEL6d signal and SEL6d$= 0, \ldots, 6$.

For example, the execution of CT1 in simulations follows in time after the ICT1. To do this, first by setting selector signal SEL6 to 1 the θ_{CT1} signal which belongs to CT1 is written in the input of the CORDIC sine-cosine calculation block. Meanwhile SEL4, SEL3 are set to 0, 2 respectively. Which means the inputs of the multiplier are set to $X_{abc}CT1$, $X_{qd}ICT3$ and by setting SEL2 to 0 only $X_{qd}ICT3$ go through the multiplier and makes the required output for ICT3 block that was started before. SEL2 is set to 0 which means CORDIC sine-cosine

TABLE III
NECESSARY RESOURCES FOR REUSE OF ONE BLOCK FOR ALL PARK AND INVERSE PARK TRANSFORMATION

Resources needed for doing all transformations by reusing of blocks Execution time: 16 ns Percentage of needed resources for Xilinx Spartan 3E XC3S500E				
Slices: 791 16%	FFs: 904 9%	LUTs: 1543 16%	Emb. Mults: 16 80%	TBUFs: 1 4%

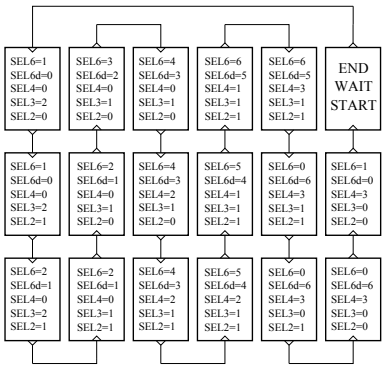

Fig. 6. Finite State Machine representation for Park and Inverse Park Transformations using one block

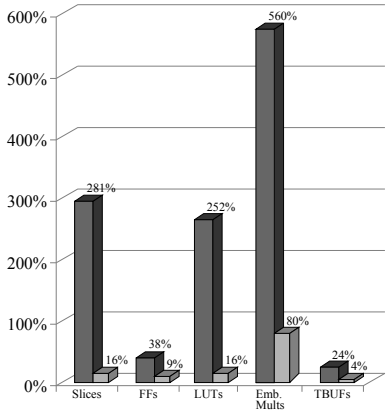

Fig. 7. (Dark gray bars) Total resources needed for doing each transformation in one individual block, (Gray bars) Total resources needed for doing all transformations in one block

calculation block should do sine operation. At this moment the output of the multiplier block which was set to $X_{qd}ICT1$ one period before, remains constant by using enable signal SEL6d and gives the calculation output for ICT1. This time sequence calculates first part of CT1. In the next step cosine part of this transformation will be calculated and input of the multiplier will connect to $X_{abc}CT1$ and the output will ready one moment later. A FSM graph shows working sequence for calculating one period of all transformations in one block illustrated in Fig. 6. By using this method needed resources will be as mentioned in Table III and a comparison of used resources shown in Fig. 7.

VI. FINE TUNING OF PI CONTROLLERS

As an example of possible uses for HIL we introduce the fine tuning of the proportional and integral gains of the PI current controllers. The current reference set-point and fine tuning of the integrator gains of PI-controller are realised using a rotary encoder available in the Spartan 3E FPGA board. By using this encoder, the designer can tune each controller

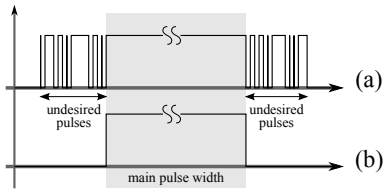

Fig. 8. (a) Rotary encoder actual output pulse (worth-case), (b) Filtered output

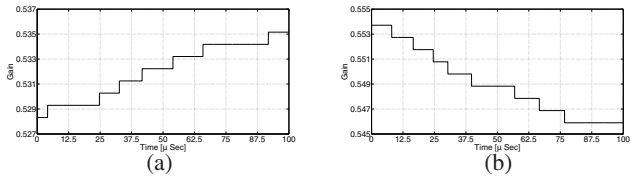

Fig. 9. Rotary encoder filtered and limited measured output (a) increasing gain (b) decreasing gain

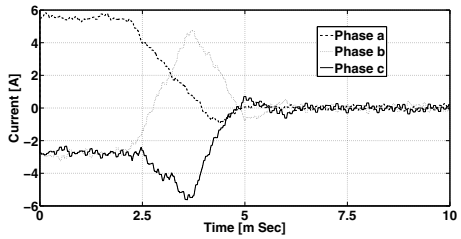

Fig. 10. Simulation results for startup abc currents in System Generator

gain and measure the output of the PI-controller to find the appropriate values for every normal and extreme working conditions. To remove undesired pulses of the mechanical rotary encoder a digital logic low-pass filter is designed as shown in Fig. 8. The figure shows in gray the main and desirable part of the output pulse of the encoder. In Fig. 8(a) an output signal which includes a mixture of very short pulses with different widths and spikes is illustrated which is perfectly filtered by the logic filter. Measurements show that in each rotation only one or two of these undesired pulses occur. The working sequence of the encoder starts with filtering two output pulses of the encoder and then the direction is calculated comparing two pulses. At the end of the program, pulses of each direction are counted and limited to the upper and lower values of the gain. Fig. 9 shows how the gain values change inside the FPGA. It's very easy to achieve smaller or larger step sizes to make tuning more accurate. The step size is chosen as 0.01 in this implementation.

The fine tuning of the PI-controller gains is done by using the proposed method. The following proportional and integral values have been selected for each of the q-axis and d-axis currents:

$$G_{qI} = 0.55,\ G_{qP} = 0.47,\ G_{dI} = 0.28,\ G_{dP} = 0.23$$

VII. SIMULATION RESULTS FROM SYSTEM GENERATOR AND MEASURED RESULTS FROM FPGA

The simulation results for rotor position estimation using System Generator at zero speed are plotted in Figs. 10 and 11. Fig. 10 shows the 3-phase currents of the motor which is converging very fast to zero after start-up transients and Fig. 11 shows the estimated position of the rotor which

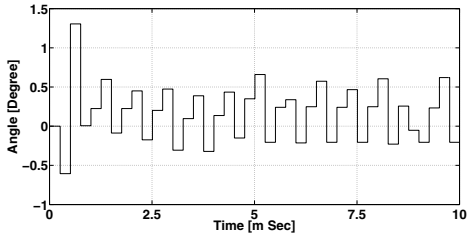

Fig. 11. Simulation results for estimated rotor-position in System Generator

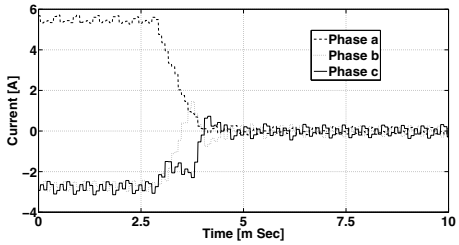

Fig. 12. Measurements for abc currents

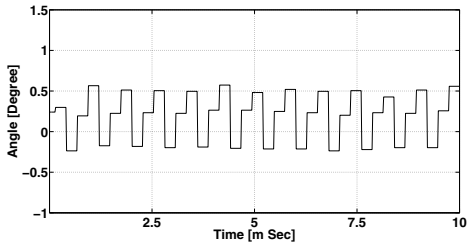

Fig. 13. Measurements for estimated rotor-position

should be zero and it has maximum 0.5 degree error which is negligible. These results confirm the accuracy of sensorless control method, motor model and controller from the start-up transients to the steady-state. Measurements from the FPGA board are provided by Chipscope Pro Analyzer software which measures the internal physical signals of a FPGA and uploads it to the computer. The Figs. 12 and 13 show the on-line measured steady-state currents and rotor-position estimation. These figures are similar to the steady-state part of Figs. 10 and 11 respectively and confirm that the measured states of hardware in loop platform are in agree with simulation results.

VIII. CONCLUSION

In this paper, a position estimator as well as a current controller and interior PMSM model are discussed and simulated in a completely discrete-time environment For this purpose, the program is uploaded to a FPGA to obtain online measurements. The FPGA platform can easily be used for testing all control methods in online mode. It is also useful to study the effects of parameter changes in all parts of the drive system and this on a cheap Spartan 3E XC3S500E FPGA evaluation board which can bought for below 160$. It's also possible to replace the motor model with a real one and verify the performance of digital control units in their final design stage such as current controller, position estimation. Therefore it provides a powerful tool for students as well as designers.

IX. ACKNOWLEDGEMENTS

The authors wish to thank the federal program Interuniversity Attraction Pole (IUAP) financed by the Belgian Science Policy

REFERENCES

[1] F. De Belie, P. Sergeant and J. Melkebeek "A Sensorless Drive by Applying Test Pulses Without Affecting the Average-Current Samples" *IEEE Trans. Power Electron.*, Vol. 24 (4), pp. 875-888, April 2010

[2] L. Peretti, M. Zigliotto "FPGA-Based Voltage Measurements in AC Drives" *IEEE Int. Conf.*, Electrical Machines (ICEM), pp.1-6, 6-8 Sep. 2010

[3] F. De Belie, J. Melkebeek "Seamless Integration of a Low-Speed Position Estimator for IPMSM in a Current-Controlled Voltage-Source Inverter" *IEEE Int. Conf.*, Sensorless Control for Electrical Drives (SLED), pp.50-55, 9-10 July 2010

[4] F. De Belie, T. Vyncke and J. Melkebeek "Parameterless Rotor Position Estimation in a Direct-Torque Controlled Salient-Pole PMSM without Using Additional Test Signals" *IEEE Int. Conf*, XIX International Conference on Electrical Machines (ICEM), pp.1-6, 6-8 Sept. 2010

[5] J.J. Rodriguez-Andina, M.J. Moure, and M.D. Valdes, "Features, Design Tools, and Application Domains of FPGAs," *IEEE Trans. Ind. Electron.*, Vol. 54, No. 4, Aug. 2007

[6] T. J. Vyncke, S. Thielemans, M. Jacxsens, and J. A. Melkebeek, "Analysis of some design choices in model based predictive control of flying-capacitor inverters," *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 31, no. 2, p. 619-635, Feb. 2012

[7] S. Thielemans, T. J. Vyncke, and J. A. Melkebeek, "Weight factor selection for model based predictive control of a four-level flying-capacitor inverter," *IET Power Electron.*, vol 5, no. 3, p. 323-333, March 2012

[8] L. Idkhajine, E. Monmasson, M.W. Naouar, A. Prata, and K. Boullaga, "Fully Integrated FPGA-Based Controller for Synchronous Motor Drive," *IEEE Trans. Ind. Electron.*, Vol. 56, No. 10, Oct. 2009

[9] Z. Fang, J.E. Carletta, and R.J. Veillette, "A Methodology for FPGA-Based Control Implemention," *IEEE Trans. Cont. Sys. Tech.*, Vol. 13, No. 6, Nov. 2005

[10] C. Dufour, J. Blanger, V. Lapointe, S. Abourida, "Real-Time Simulation on FPGA of a Permanent Magnet Synchronous Machine Drive Using a Finite-Element Based Model," *IEEE Int. Symp.*, Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), pp.19-25, 11-13 June 2008

[11] X. Lin-Shi, F. Morel, A.M. Llor, B. Allard, and J.M. Rtif, "Implementation of Hybrid Control for Motor Drives," *IEEE Trans. on Ind. Electron.*, Vol. 54, No. 4, Aug. 2007

[12] C. Dufour, S. Abourida, J. Belanger, "Real-Time Simulation of Permanent Magnet Motor Drive on FPGA Chip for High-Bandwidth Controller Tests and Validation," *IEEE Int. Symp.*, Industrial Electronics, Vol.3, No., pp.2591-2596, 9-13 July 2006

[13] E. Monmasson, and M.N. Cirstea, "FPGA Design Methodology for Industrial Control Systems-A Review," *IEEE Trans. Ind. Electron.*, Vol. 54, No. 4, Aug. 2007

[14] M.W. Naouar, E. Monmasson, A.A. Naassani, I. Slama-Belkhodja, and N. Patin, "FPGA-Based Current Controllers for AC Machine Drives-A Review," *IEEE Trans. Ind. Electron.*, VOL. 54, NO. 4, Aug 2007

[15] M. Boussak and K. Jarray, "A high-performance sensorless indirect stator flux orientation control of induction motor drive," *IEEE Trans. Ind. Electron.*, Vol. 53, No. 1, pp. 41-49, Feb. 2006

[16] E.D. Mitronikas and A.N. Safacas, "An improved sensorless vector control method for an induction motor drive," *IEEE Trans. Ind. Electron.*, Vol. 52, No. 6, pp. 1660-1668, Dec. 2005

[17] F.W. Krach, B.P. Frackelton, J.E. Carletta, and R.J. Veillette, "FPGAbased implementation of digital control for a magnetic bearing," *Proc. Amer. Control Conf.*, Denver, CO, pp. 1080-1085, Jun. 2003

[18] Y. Sorel, "Massively parallel computing systems with real time constraints, the algorithm architecture adequation methodology," *In Proc.*, Proc. of the Massively Parallel Computing Systems, May 1994

[19] R. Andraka, "A survey of CORDIC algorithms for FPGAs," *ACM Proc.*, International symposium on Field programmable gate arrays (ACM/SIGDA), pp. 191-200, Monterey CA, Feb. 22-24, 1998