

FPGA implementation of Induction Motor Vector Control using Xilinx System Generator

Jean-Gabriel Mailloux, Stéphane Simard, Rachid Beguenane
Groupe ERMETIS, Département des Sciences appliquées
Université du Québec à Chicoutimi,
555, boulevard de l'université, G7H 5B1 (QC), Canada

Abstract: - Mechatronic systems are complex interdisciplinary products where many parts are present that overlap each other; electronic control board, electromechanical machines, and power drives. With FPGAs, their control can run faster as multiple operations are executed in parallel. A simulation tool that facilitates the direct translation into hardware of mechatronic control algorithms is desirable. The Xilinx System Generator (XSG), a high-level tool for designing high-performance DSP systems under Simulink, has emerged as an excellent tool for such purpose. In this paper such tool is exercised to implement the conventional vector control algorithm for AC drives, which exhibit a good level of complexity.

Key-Words: - FPGA, System Generator, Induction Motor, Vector Control

1 Introduction

The control of mechatronic systems, a complex interdisciplinary product, can run faster with FPGAs as multiple operations are executed in parallel. It is then highly desirable to have a simulation tool that can easily make the direct translation into hardware of control algorithms with no-knowledge of any Hardware Description Language (HDL). The Xilinx System Generator (XSG), a high-level tool for designing high-performance DSP systems under Simulink environment, has emerged as an excellent tool advisable for such purpose. In fact the XSG takes the graphical algorithmic approach and extends it to FPGA development by using dedicated Simulink Blockset [1]. The knowledge of HDL, necessary to program FPGA devices, is then not required for DSP designers who are not familiar with. This is because the XSG tool can automatically translate the algorithm into FPGA resource and loaded onto the FPGA device. The XSG tool has been used in some areas, such as to cite few; developing hardware-based computer vision algorithms from a system level approach [2], designing a reconfigurable video encryption [3], implementing a wide variety of neural models with the performance of custom analogue circuits or computer clusters [4].

The present work is aiming to show the usefulness of using XSG to prototype complex control algorithms such as the vector control, a well known control strategy for AC drives, particularly those based on induction motors [5]. The authors of paper [6] have already modelled the widely used DTC algorithm where they have used two FPGA boards with the control algorithm implemented on the master board, while the other board controls the power inverter.

The authors of [7] have presented a number of papers on FPGA prototyping of a vector control system using XSG. They mainly developed new IP modules specifically dedicated for vector control design. Moreover, no experimental results were reported to illustrate the profiles of the controlled signals as the design has been tested only in open loop without coupling the vector control design with a virtual induction motor drive.

In this paper the vector control, considered as a special field of digital signal processing exhibiting a complex modularity, is designed using XSG version 8.1. The design will be tested in closed loop with the electromechanical drive being modelled with SimPowerSystems blockset. Section 2 presents briefly the vector control algorithm in order to derive the

corresponding scheme from which the algorithm is implemented using XSG, which is the focal point of section 3. Section 4 discusses the implementation analysis and provides some simulation results while section 5 sketches few conclusions.

2 Vector Control Principle

Vector control is the first method which makes it possible to artificially give certain linearity to the torque control of the induction motor. Precise induction motor control requires the independent control of the stator current components producing the required magnetizing flux and the electromechanical torque, in the same way as with the DC motor.

There are several variants of vector control, among them the rotor flux orientation control scheme as shown in Fig. 1. This is derived from the electromechanical model of the induction motor in the Park reference frame (d, q), known as Park domain. The mathematical description and the corresponding vector control expressions can be found in [8].

3 Vector Control Modeling with XSG

Initially, an algorithm is designed and simulated at the system level with the floating-point Simulink blocksets. A hardware representation for FPGA implementation is then derived using XSG, a plug-in tool to the Simulink modelling software. The later provides a bit-accurate model of FPGA circuits and automatically generates a synthesizable VHDL code for implementation in Xilinx FPGAs. It contains common parameterizable blocks that enable bit-true and cycle-true modelling. For vector control modelling, the blocks used are mostly multipliers, adders, MACs, etc. Some complex arithmetic operators, such as square-root and division used in rotor flux estimation, are embedded from the Xilinx IP core generator. The important feature of the XSG tool is a quick evaluation of the algorithm response when modifying data width and coefficient width of the used variables and parameters. The design and simulation process of the vector control algorithm consists of the following steps:

1. Start by coding separately each module of the vector control algorithm using Xilinx blocks, shown in Fig. 1. An example of PI controller, duplicated 4 times, is illustrated in Fig. 2. This particular instance of vector control requires an external start signal to execute, meaning that the MACs used in the four PIs are also driven by an enable signal so as not to overflow.

2. Once completed, use the Xilinx Gateway blocks to automatically convert floating-point numbers from the Simulink environment into the fixed-point numbers for the XSG environment. There are Gateway-in and Gateway-out blocks to define respectively the inputs and outputs of the vector control fully designed with Xilinx blocks (Fig. 3).

3. Import pre-existing hardware implementations that may not exist in the base XSG blockset. For example, the CORDIC square-root function from the Xilinx IP core generator is useful for rotor flux estimation (Fig. 4). The recent versions of XSG offer these math functions in their Xilinx Reference Blockset.

4. Depending on the bit precision desired for each module and for the system as a whole, the XSG blocks are optimized through the setup of their number of bits and binary point position. Any method may be used to deduce the optimal values, but a tool such as GAPP [9] proves very helpful in pinpointing the best numbers.

5. Interface the XSG design with any other Simulink resource for simulation, analysis or other purposes. For example, to simulate the vector control in closed loop, Embedded MATLAB functions are used to emulate the PWM firing and gating signals generation while the SimPowerSystems blockset is used to model the power drive and induction motor. The simulated induction motor used in this paper has the following parameters nominal power of 2238 VA, 220 V^{rms}, 5.87 A^{rms}, 2 pairs of poles and 60 Hz frequency.

6. Verify the results using stimuli inputs, for reference speed and load torque, and scopes or

other visual outputs to evaluate the algorithm response.

4 Analysis and Simulation Results

In order to validate the XSG vector control algorithm, an example of one particular test is depicted. Under rotor flux rated value and a slight load, the modelled drive starts from standstill, and then smoothly accelerates to an arbitrary speed of 286 rpm, slows down, reverses, accelerates to -286 rpm, slows down again to standstill and goes back up to 573 rpm. After the first 7 seconds, a very similar speed profile is conducted for another 7 seconds for nominal load. Speed response of the vector control algorithm is shown in Fig. 4, and the corresponding rotor flux profile is illustrated in Fig. 5. As we can see, the speed is well controlled while the rotor flux is kept regulated to its rated value of 1 Wb, which is the objective of vector control.

Once the vector control is tested, hardware blocks can be synthesized and downloaded into a desirable FPGA device. Table 1 shows the implementation results for each developed module when targeting the Virtex-4 device and using an overall precision of 16 bits. We can notice the rotor flux estimator module is the most resource intensive as it requires the use of one square-root and two division operations, all of which are implemented with the CORDIC Xilinx IP core.

5 Conclusion

An FPGA implementation of the conventional vector control algorithm for AC drives has been presented using the XSG tool under Simulink environment. The XSG is very practical since knowledge of any HDL is unnecessary; and it renders algorithm prototyping more accessible and less laborious as it permits an automatic extraction of bit-file targeting commercial FPGA devices.

An equivalent ASIC design could then be easily derived for mass production. The design has been validated by integrating the vector control designed solely with XSG blocks into a closed loop where the PWM firing and gating signal generation, power drive, and induction

motor are modelled using standard Simulink blocks and the SimPowerSystems blockset.

Table 1: Vector Control Modules Characteristics (16 bits precision)

Module Name	Resources		
	Slices	FFs	LUTs
Decoupling	965	995	1759
PIs (All 4 PIs)	703	463	1143
Park Transform	666	1124	1268
Inverse Park	624	1064	1206
Clarke Transform	102	67	148
Rotor Flux Estimator	3066	4763	4505
ω estimator	299	267	459
Total Estimated Resources	6425	8743	10488

A test has been conducted when the complete AC drive runs under a particular speed profile and physical resources for the FPGA have been estimated.

The massive available number of XSG blocks and dedicated libraries for implementing advanced control algorithms makes it a highly suitable environment for designing and simulating the control systems in modern FPGAs with the advantage of being close to real hardware.

Acknowledgment

This research is funded by a grant from the National Sciences and Engineering Research Council of Canada (NSERC). CMC Microsystems provided development tools and support through the System-on-Chip Research Network (SOCRN) program.

References:

- [1] "System Generator for DSP," Xilinx Inc. 2006, 10, February 2006.
http://www.xilinx.com/ise/optional_prod/system_generator.htm

[2] Ana Toledo Moreo, et al, Experiences on developing computer vision hardware algorithms using Xilinx system generator, *Microprocessors and Microsystems*, vol. 29, Issues 8-9, 1 November 2005, pp. 411-419.

[3] Daniel Denning, et al, Using System Generator To Design A Reconfigurable Video Encryption System, *13th International Conference on Field Programmable Logic and Applications*, Sept.1-3, 2003, Lisbon – Portugal.

[4] Randall K Weinstein and Robert H Lee, Architectures for high-performance FPGA implementations of neural models, *Journal of Neural Engineering*, 3 (2006) 21–34.

[5] Ramon Blasco Giménez., High performance Sensorless Vector Control Induction Motor Drives, *PhD Thesis*, University of Nottingham, UK, December 1995.

[6] Francesco Ricci and Hoang Le-Huy, Modeling and simulation of FPGA-based variable-speed drives using Simulink, *Mathematics and Computers in Simulation*, vol. 63, Issues 3-5, 17 Nov. 2003, pp. 183-195.

[7] József Vásárhelyi, et al, FPGA Implementation Vector Control of Tandem Converter Fed Induction Machine, *6th International Symposium of Hungarian Researchers on Computational Intelligence*, Nov. 18-19, 2005, Budapest, Magyar.

[8] Rachid Beguenane, et al, Towards the System-on-Chip Realization of a Sensorless Vector Controller with Microsecond-order Computation Time, *19th IEEE Canadian Conference on Electrical and Computer Engineering*, Ottawa, Canada, May 10, 2006.

[9] Florent de Dinechin and Christoph Lauter, Assisted verification of elementary functions using Gappa, *SAC'06*, 2006, Dijon, France.

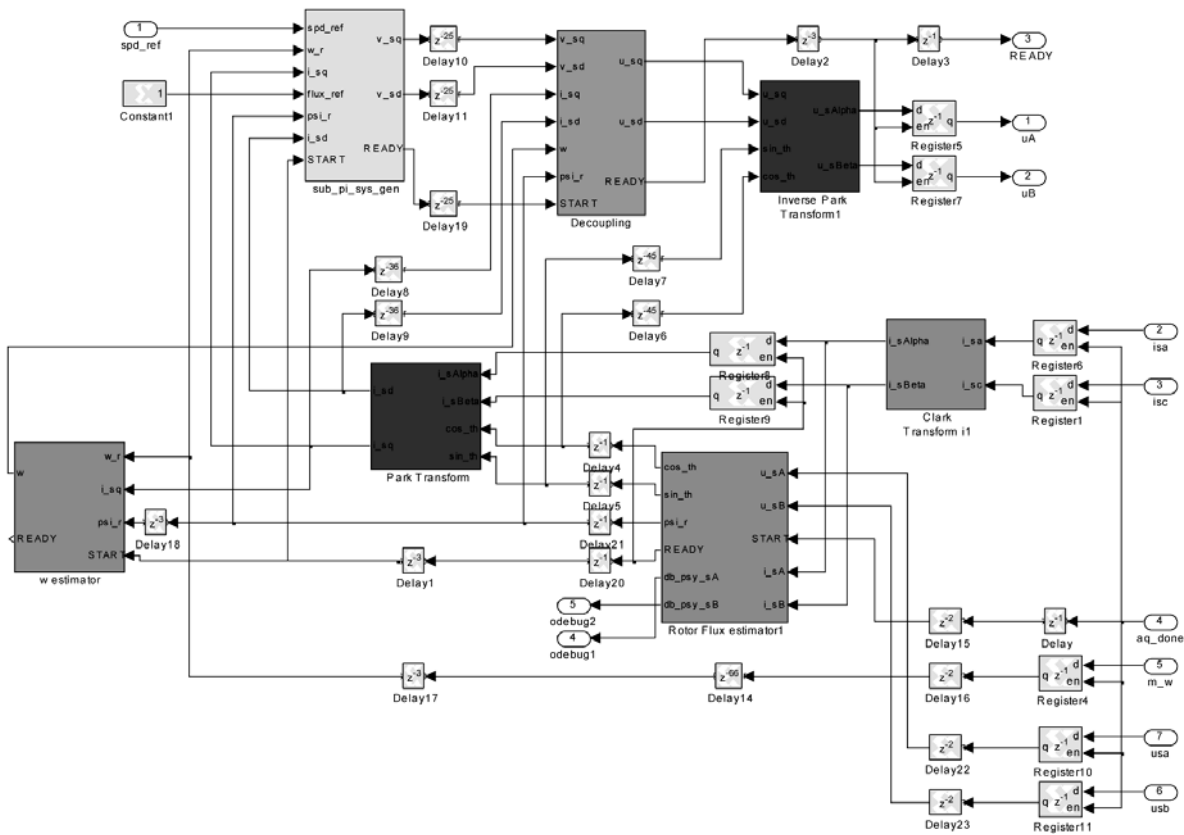


Figure 1. Vector control XSG modules

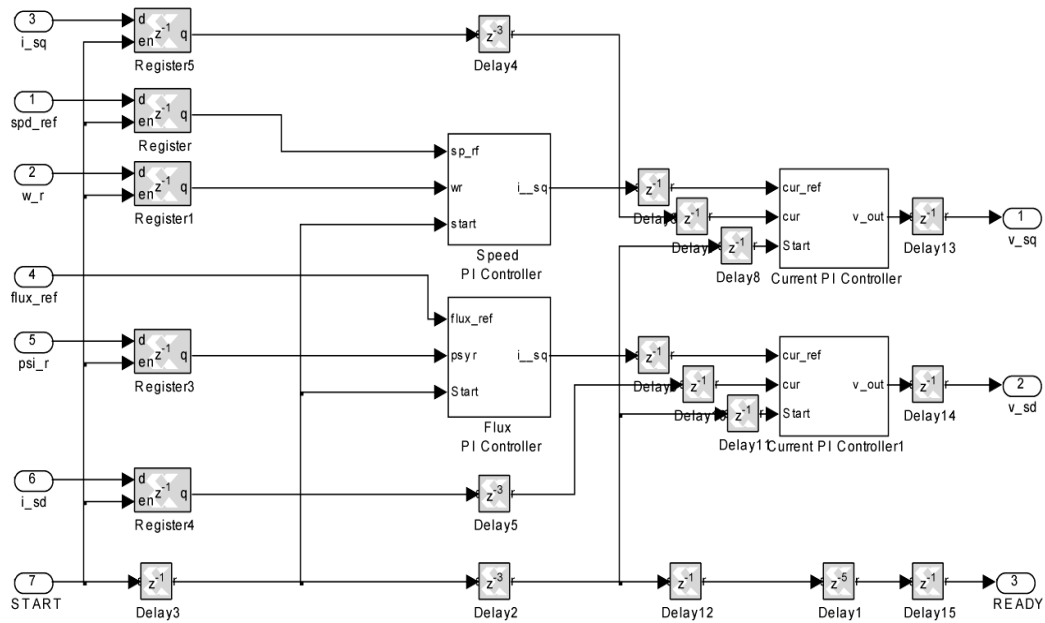


Figure 2. Four PI instances using XSG blocks and a start signal

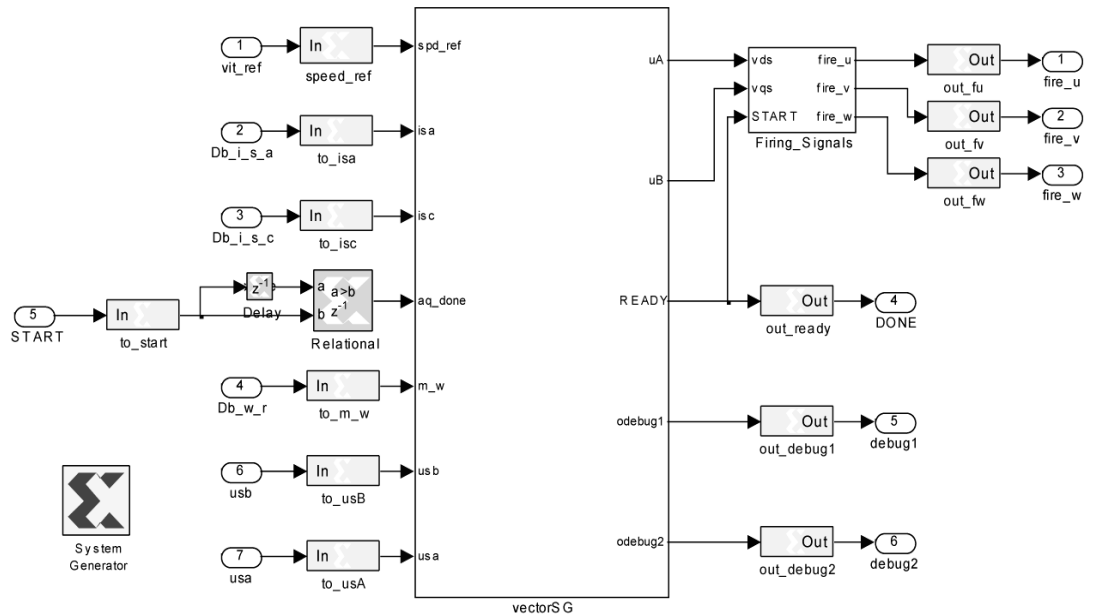


Figure 3. An interface of gateway-in and gateway-out blocks

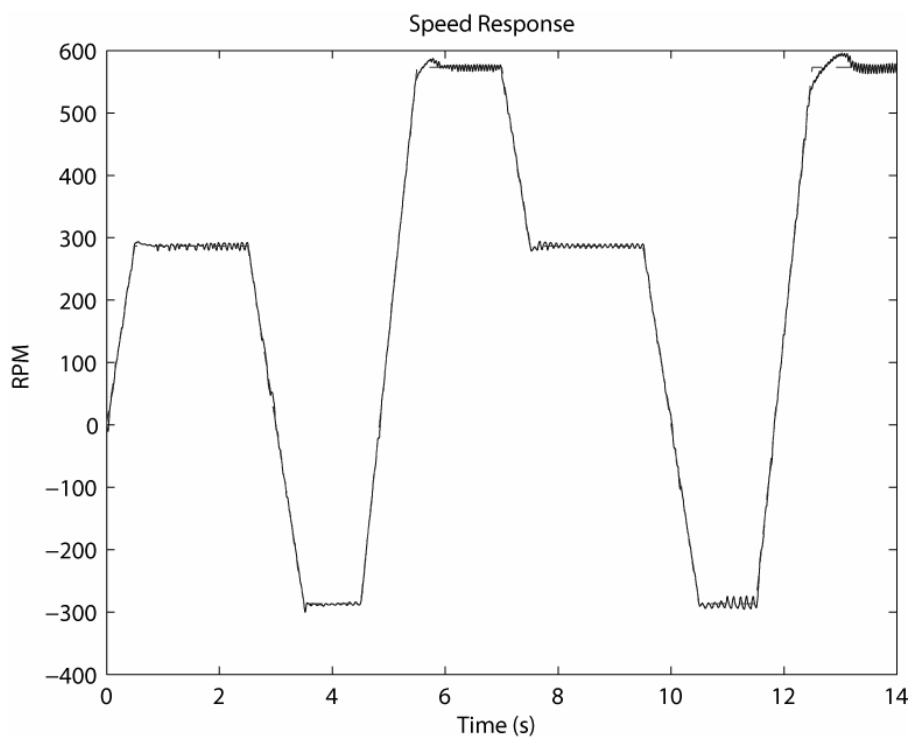


Figure 4. Speed response of the vector control in closed-loop

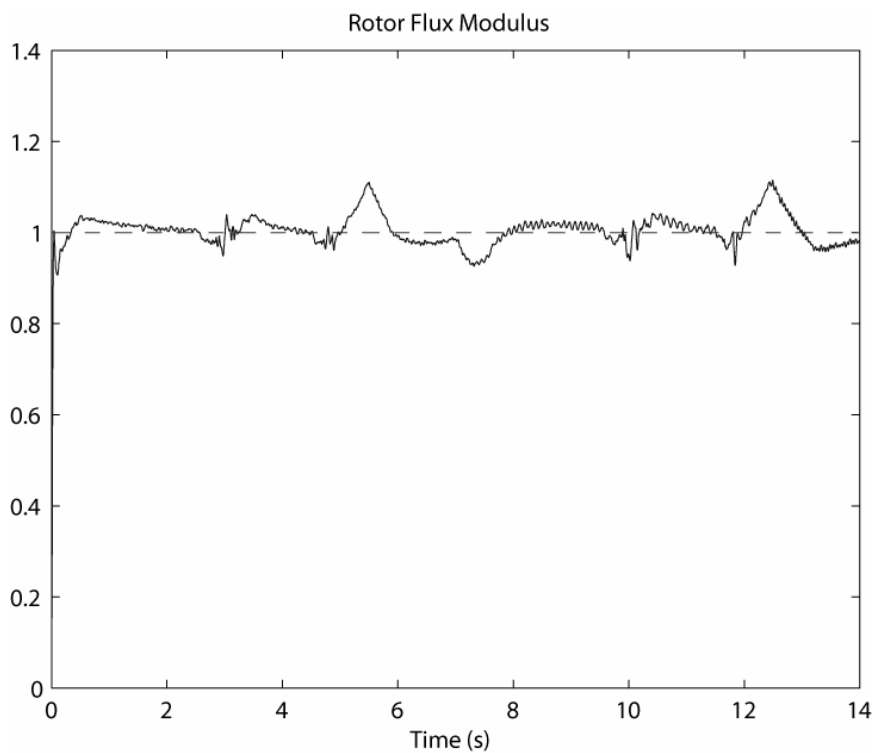


Figure 5. Rotor flux modulus in closed-loop