Design, Validation and FPGA Implementation of a Brushless DC Motor Speed Controller

Bogdan Alecsa
Department of Automatic Control and Applied Informatics
Technical University "Gheorghe Asachi"
Iasi, Romania
balecsa@ac.tuiasi.ro

Alexandru Onea
Department of Automatic Control and Applied Informatics
Technical University "Gheorghe Asachi"
Iasi, Romania
aonea@ac.tuiasi.ro

Abstract— **The brushless DC (BLDC) motors have become the common choice in low power, high speed, high accuracy applications and this imposed the need for efficient and low cost motor control drives. In this paper, a design method for a digital controller implemented in a low cost field programmable gate array (FPGA) is presented. The design is done by schematic capture and Simulink block diagram capture. The developed design is validated in a modular fashion by Simulink-HDL (hardware description language) co-simulation and experimental results are provided. The main contribution is the presented controller design method for FPGA implementation, with special emphasis on simulation validation and FPGA debugging tools usage for signals extraction and analysis.**

Keywords-FPGA, BLDC, speed control, System Generator, HDL co-simulation.

I. INTRODUCTION

Industrial electronics, home appliances, automotive and robotics applications are relying more and more on the use of brushless motors, due to their higher efficiency and lower maintenance compared to conventional brushed DC motors. This has imposed the need for low cost brushless motor control drives.

The programmable logic devices (PLD) have been a widely used mean of prototyping BLDC driving circuits, because they offer the possibility of defining custom hardware and modifying it in several iterations until the desired functionality is obtained. However, the control algorithm is not easily implemented in a PLD, and a processor is needed to perform the computations. Usually, a digital signal processor (DSP) is employed.

Modern embedded devices include a DSP and powerful peripherals into a so called digital signal controller (DSC), which is basically a microcontroller with DSP processing capabilities. DSCs targeting motor control are commonly offered.

Although these devices are indeed very powerful, they do not have the total flexibility of PLDs. And the top in PLD technology is represented by field programmable gate arrays (FPGAs), which offer the possibility of defining efficient custom hardware and powerful processing capabilities. In fact, due to the inherent parallelism in FPGA architecture, they can perform complex computations faster than DSPs [1].

This is the reason FPGAs are presenting interest in automatic control: because of the highly parallel structure and the embedded dedicated hardware, FPGAs can be used to implement very fast and complex control algorithms [2].

Even low cost FPGAs, like the Spartan-3E XC3S500E, which was used in the experiments, provide embedded hardware multipliers 18x18 bits wide [3], making them very efficient in implementing custom multiply-add or multiply-accumulate based computing engines.

Corroborating the hardware possibilities of FPGAs with the high abstraction level of the modern design tools, it is possible to implement easily control systems for fast processes. Moreover, it is possible to validate the control system using powerful simulation tools, using complex models for the process and bit and cycle accurate models for the FPGA hardware [2].

This paper presents the design of a brushless DC motor speed controller implemented inside a low cost FPGA, emphasizing the validation and verification methods. Combined schematic capture and Simulink system level design were employed, and Simulink/ModelSim co-simulation was used for validation.

II. BRUSHLESS DC MOTOR DRIVING

This section gives a short overview of the BLDC motor driving, in order to ease the explaining of the controller design.

A BLDC motor is a type of synchronous motor: the magnetic field generated by the stator and the magnetic field generated by the rotor rotate at the same frequency. The BLDC motor uses a permanent magnet rotor, and an electro-magnetic stator, usually built in a three-phase Y configuration [4].

In a BLDC, the back electromotive force (EMF) induced per phase of the motor winding is constant for 120° and changes linearly with rotor angle before and after the constant part. In order to obtain constant output power, current is driven through the motor winding during the flat portion of the back EMF waveform [5].

This work was partially supported by the National Centre for Programs Management from Romania under the research grant SICONA – 12100/2008.

The most commonly used driver topology for three phase BLDC in low power applications is the three-phase inverter bridge [5], employing 6 MOSFETs.

Unlike in a brushed DC motor, in a BLDC motor the current commutation must be controlled electronically. To rotate the BLDC motor, the stator windings should be supplied in sequence [5]. Each phase is energized only during the 120° when the back EMF is constant. There is a commutation event for every 60°. The controller must therefore have information about the rotor position, in order to activate the switches. Usually, this information is obtained from 3 Hall sensors spaced 120° apart. These sensors give 3 bit codified information about the rotor position.

The BLDC motor speed can be controlled using only digital electronics by varying the duty cycle of a pulse width modulated (PWM) signal injected to the inverter bridge transistors.

The motor speed can be determined from the Hall effect sensors state change frequency, but the resolution of this method is very low. A more precise optical position encoder is usually used for speed feedback.

III. CONTROLLER DESIGN

In this section, only a short description of the design is made, in order to present the validation procedure in the next section. A detailed description of the design is given in [6].

Fig. 1 presents the block diagram of the controller. The control system is totally digital. It interfaces to the motor through a three phase inverter bridge and by three Hall effect sensors and a quadrature encoded pulse (QEP) optical tachometer. The speed regulator is a discrete proportional-integral (PI) type [6].

The design is synchronous, driven by a 50MHz clock signal. The logic blocks were implemented by schematic capture. Only the PI regulator was designed in Matlab Simulink, using Xilinx System Generator for DSP, by a methodology similar to that presented in [7], and then integrated with the rest of the design.

In the following paragraphs, each part of the controller is shortly explained.

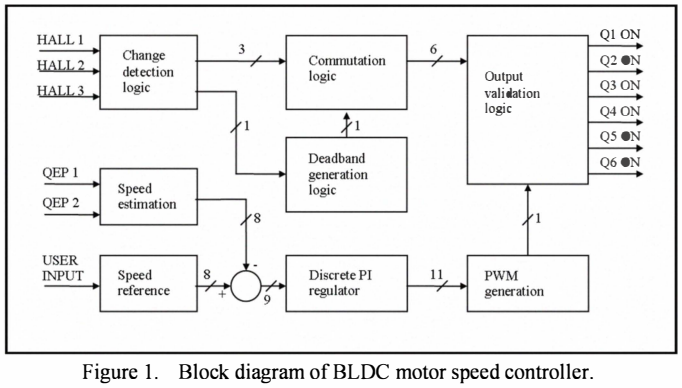

Figure 1. Block diagram of BLDC motor speed controller.

A. The logic modules

The change detection logic synchronizes the Hall sensors signals to the system clock signal and generates a one clock cycle pulse when a change in the sensors state in detected. This way, it signals the rotor position has changed.

The dead band generation logic inserts a delay between the switch off command of one transistor and the switch on command of the next transistor in sequence on the same side (low side or high side) of the inverter bridge. This prevents overcurrents in the motor windings caused by a short moment when both the deactivated transistor and the activated one are in conduction. The "dead time" must be long enough to cover the transistor switch on/switch off time. For the experimental setup used in this case, a dead time of at least 0.5μs was required.

The commutation logic uses the synchronized Hall sensors signals to derive 6 signals, each representing the state of one of the 6 transistors in the inverter bridge. Additionally to the 3 Hall sensors signals, a validation input from the dead band generation logic is used to insert the commutation delay. The commutation logic is implemented by 6 read only memory (ROM) cells of 16 bits each, this being the optimal choice for the 4-input look-up table (LUT) based architecture of the FPGA.

The PWM generation logic provides a PWM signal of fixed frequency, the duty cycle being set by an 11 bit input. For simplicity, the frequency of the PWM signal is set at generation time as 24.425 kHz, but it can be made configurable.

The output validation logic passes through the PWM signal to the inverter bridge transistors according to the inputs it gets from the commutation logic. It can be configured for hard chopping (both low side and high side transistors are injected the PWM signal) or soft chopping (only the low side transistors are chopped).

The speed estimation logic counts the QEP signals edges in a sample period, according the constant elapsed time (CET) measuring principle. For a better resolution of the speed estimation, both the rising and falling edges of the two QEP signals are counted. The counter value is saved to a register at every sampling period.

B. The discrete PI regulator

As already mentioned, the PI regulator was designed in Simulink using System Generator blocks and was validated by simulation.

The motor modeling and regulator tuning are not in the scope of this paper, but after these steps, the PI parameters resulted as:

$$K_I = 1704.003\,, \tag{1}$$

$$K_P = 12.8604\,. \tag{2}$$

The sampling time for the control algorithm was chosen as:

$$T_s = 1ms. \tag{3}$$

Using the bilinear transform, the following equation is obtained for the discrete PI regulator (a detailed description of the method through which it is obtained can be found in [6]):

$$u[k] = (\frac{K_I T_s}{2} + K_P)\varepsilon[k] + (\frac{K_I T_s}{2} - K_P)\varepsilon[k-1] + u[k-1], \tag{4}$$

where $u[k]$ is the regulator output and $\varepsilon[k]$ is the speed error, which must be computed numerically as the difference between the speed reference and the estimated speed.

It is obvious that (4) corresponds to an infinite impulse response (IIR) system, and so it can be implemented as a digital filter.

Using the System Generator for DSP software from Xilinx, the digital IIR filter was designed in Simulink and simulated together with a motor model. The simulation used a 1ms sample time for the inputs.

The filter coefficients corresponding to the $\varepsilon[k]$ and $\varepsilon[k-1]$ terms in (4), computed with the PI parameters (1) and (2) and the sample period in (3) are:

$$b_0 = 13.7124\,, \tag{5}$$

$$b_1 = -12.0084\,. \tag{6}$$

They can be observed in Fig. 2, having values within the quantization error limit of the chosen fixed point representation: 18 bits width, with 13 bits for the fractional part. The 18 bits precision was employed to make use of the 18x18 embedded hardware multipliers of the target FPGA.

Full precision was used for the multiply and add operations, avoiding the possibility of overflow. The $u[k-1]$ value was limited to 28 bits for the add operation, and the output of the algorithm was limited to a 11 bit unsigned representation, because the PWM circuitry uses a 11 bit value to set the duty cycle. The limitations were done with additional hardware for numeric saturation in case of overflow, which is an option of the System Generator blocks.

While this regulator design method, also employed in [7], has the advantage of a fast simulation in Simulink, together with models of the process and double precision floating point regulator models, thus validating the design, it has two major disadvantages when integrated with the rest of the design, which is made by schematic capture or HDL description. Firstly, it needs a clock signal given by the sample period. This must be generated in the top level design, violating the principle of a synchronous design driven by only one clock signal. Secondly, the propagation delay through the filter computation logic, although ensured to be smaller than the filter clock period, may be longer than the system clock period, and must be accounted for in the modules using the filter output. In the case, the PWM module must ensure it uses a valid command from the PI regulator, by reading it with a delay with regard to the computation start (the filter clock rising edge).

So, after the validation of the regulator design, it was modified to use the 50MHz system clock.

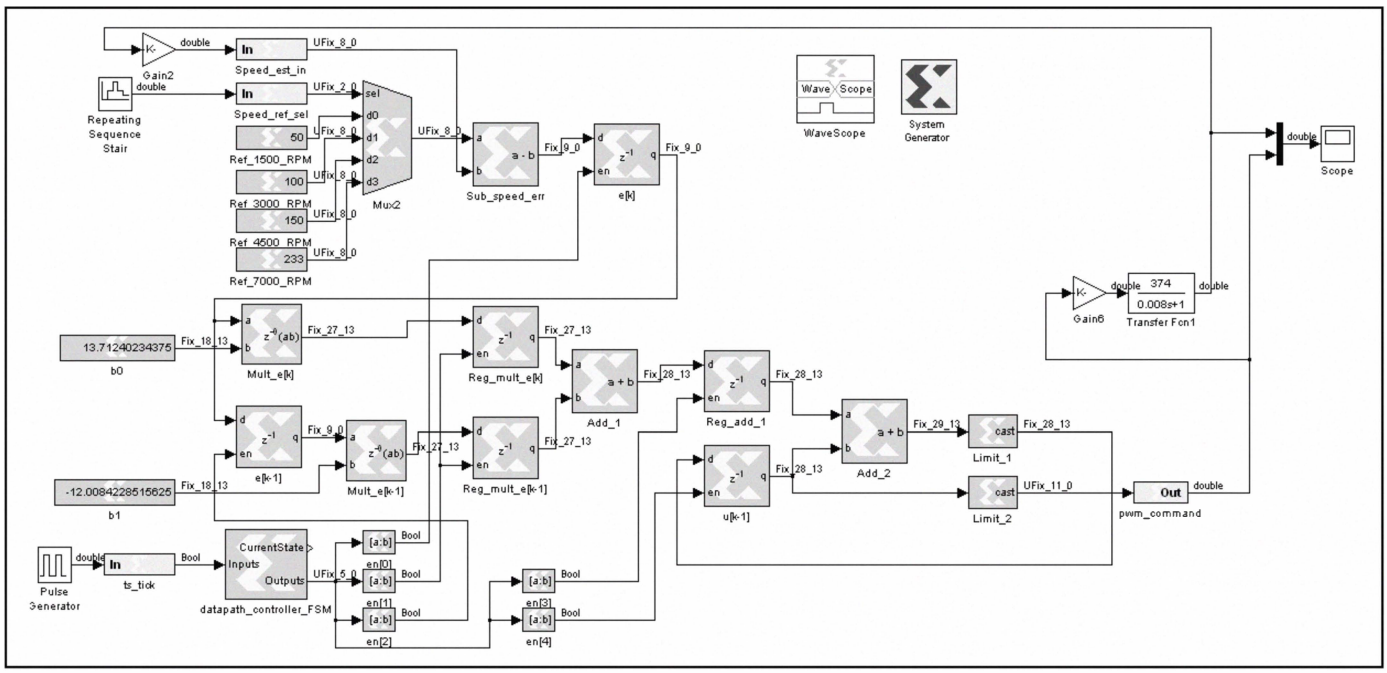

Figure 2. The PI regulator modified to use the system clock signal

For this purpose, registers with enable input were inserted in the filter computation path (data path) and a Moore type finite state machine (FSM) was designed to sequence the operations in the data path (a data path controller). The FSM has one input, the sample time signal (a one clock cycle pulse), and several outputs, one for each stage of the data path. The FSM remains in an idle state until the input activates, and then goes from one state to another without checking any input conditions. In the idle state, all outputs are disabled. In each of the other states, only one output is enabled. After all FSM states were sequenced, it returns to the idle state.

Fig. 2 presents the modified PI regulator, with additional logic to select the reference speed by using user input and to compute the speed error.

IV. FPGA IMPLEMENTATION AND EXPERIMENTAL RESULTS

The logic modules described in the previous section were implemented by schematic capture. At synthesis, a HDL description using only primitives is derived, which can be simulated for functional verification.

HDL-Simulink co-simulation [8] was used for verification: a Simulink model of the motor equipped with Hall sensors and tachometer was simulated together with imported HDL modules. The vectors created by Simulink were used by ModelSim to simulate the HDL modules, and the outputs of the HDL modules were imported back in Simulink. This method allows an automated test vector generation, with vectors probable to appear in the real environment.

After verification by simulation, the design was implemented inside the FPGA device and verified using signal extraction by the ChipScope virtual logic analyzer [9]. This tool allows the capture of internal FPGA signals with a chosen rate and triggered by a certain condition, and analysis of the captured waveform on a host computer. Depending on the capture rate, better precision or wider signal viewing can be obtained.

Fig. 3 presents a ChipScope capture of the Hall sensors signals and transistors command signals (active high) on the running design. Because the capture period is 10µs, the inserted dead time can not be observed, but it was verified with a shorter capture period.

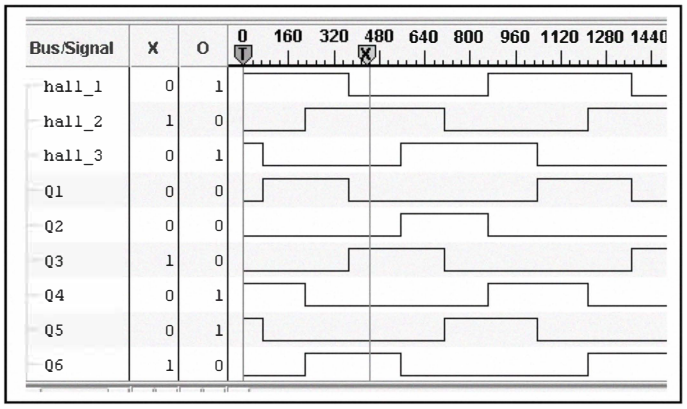

Figure 3. The Hall sensors signals and transistors command.

The waveforms can be exported as ASCII data from ChipScope and imported as numeric vectors in Matlab for further analysis. This way, the performance of the controller behavior can be analyzed by viewing the measured speed variation at a reference change. Fig. 4 presents a Matlab graphic representation of the PWM duty cycle value (dark) and measured speed value (light) at a step change in the speed reference, from 1500 to 3000 RPM (corresponding to 50 to 100 QEP pulses per ms).

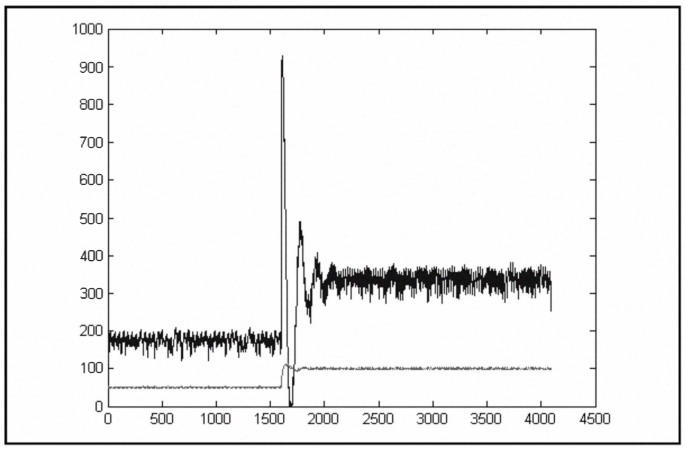

Figure 4. PWM duty cycle and speed variation at a step in reference speed

V. CONCLUSION

This paper describes a method to implement a digital BLDC motor speed controller inside an FPGA device. The controller design is a classical PI type, but the implementation method, using System Generator and Simulink, is the novelty. HDL-Simulink co-simulation was used for validation and the ChipScope virtual logic analyzer was used for verification. The design proved to work as expected.

REFERENCES

[1] C. Maxfield, FPGAs: World Class Designs, Newnes Elsevier, 2009.

[2] E. Monmasson, M. N. Cirstea, "FPGA design methodology for industrial control systems - a review", IEEE Transactions on Industrial Electronics, vol. 54, no. 4, August 2007.

[3] "Spartan-3E FPGA family: complete datasheet", Xilinx, 2008.

[4] A. Emadi, Handbook of Automotive Power Electronics and Motor Drives, CRC Press, 2005.

[5] A. Sathyan, N. Milivojevic, Y.-J. Lee, M. Krishnamurthy, A. Emadi, "An FPGA-based novel digital PWM control scheme for BLDC motor drives", IEEE Transactions on Industrial Electronics, vol. 56, no. 8, August 2009.

[6] B. Alecsa, A. Onea, "An FPGA Implementation of a Brushless DC Motor Speed Controller", Proceedings of SIITME 2010, September 2010.

[7] B. Alecsa, A. Onea, "An FPGA implementation of the time domain deadbeat algorithm for control applications", Proceedings of 2009 Norchip, Trondheim, Norway, November 2009.

[8] Astraloa A., Lazaro J., Bidarte U., Jimenez J., Zuloaga A., "FPGA technology for multi-axis control systems", Mechatronics, vol. 19, no. 2, March 2009.

[9] O. Oltu, P. L. Milea, A. Simion, "Testing of Digital Circuitry Using Xilinx Chipscope Logic Analyzer", Proceedings of CAS 2005 International Semiconductor Conference, 2005.