

A Methodology to Design Custom Hardware Digital Controllers for Switching Power Converters

A. de Castro, T. Riesgo, O. García, J. Uceda
 Universidad Politécnica de Madrid
 E.T.S.I. Industriales, División de Ingeniería Electrónica
 C/ José Gutiérrez Abascal, 2. 28006 Madrid, Spain.
 Email: {acastro; teresa; oscar; uceda}@die.etsii.upm.es

Abstract—Digital controllers are becoming an important alternative to the traditional analog solutions employed for controlling switching power converters. Among digital controllers, microprocessor-based implementations (such as DSPs) are the most usual ones. However, custom hardware implementations (FPGAs and ASICs) are obtaining special attention for their advantages in some applications, derived from their concurrency and high processing speed. These solutions are still less usual because of two main drawbacks: lack of knowledge by the designers and no methodology to follow. This work focuses on the second point, proposing a methodology to design custom hardware controllers for switching power converters.

I. INTRODUCTION

Switching power converters have been traditionally controlled by analog solutions. However, digital controllers are attracting a growing interest in this field [1]. Advantages such as more complex control algorithms, reprogrammability, reduced design time or lower components count are causing the migration from analog to digital controllers, while the price gap between both solutions is narrowing.

Most digital controllers are based on microprocessors or similar devices (i.e. DSPs -Digital Signal Processors-), but custom hardware (FPGA -Field Programmable Gate Array- and ASIC -Application Specific Integrated Circuit-) controllers are becoming attractive for switching power converters [2]-[3]. Their concurrency (all functionality is executed in parallel) allows controlling many switches or including more modules into the controller, while the superior processing speed is suitable for high switching frequencies and bandwidth [4]. In fact, some DSP controllers also include an FPGA for special features, like generating multiple driving signals [5]-[6]. However, as custom hardware devices can be in charge of the complete controller, the proposed solution is to integrate it in a single custom hardware device [7].

A typical drawback of custom hardware solutions has been price, but nowadays an FPGA for this application can be almost as cheap as a DSP (2,5 \$ for a *Spartan-XL* FPGA from *Xilinx*TM and 2 \$ for a *TMS320C24x* DSP from *Texas Instrument*TM), and an ASIC can be even cheaper for high-volume production. In fact, the main drawbacks of using

custom hardware for these controllers are that their use is less spread than that of microprocessors, and that there is no clear methodology to design them. This work proposes a complete methodology to design custom hardware controllers, proving its validity with a prototype.

II. METHODOLOGY

The proposed methodology defines the necessary steps to design a digital controller for a switching power converter. This methodology is thought for custom hardware implementations, that is, digital controllers based on FPGAs (programmable hardware devices) or ASICs (non-programmable but high-performance devices). The methodology is valid for the controller of any converter, independently of its topology or application, although an example is used in order to clarify it (see below).

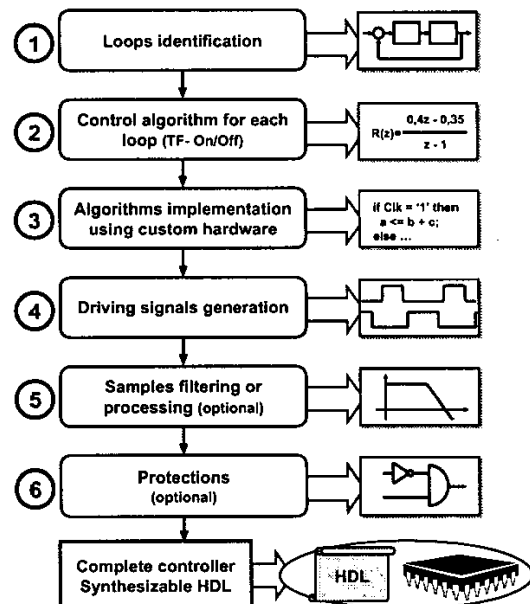


Fig. 1. Methodology steps.

The methodology is summarized in Fig. 1, showing six steps for obtaining the final HDL (hardware description language) description of the controller ready for synthesis. The first three steps would correspond with some analog design flow in a microprocessor-based methodology,

changing the implementation from hardware to software. However, the next three steps (4 to 6) are specific of a custom hardware design, as they refer to problems that are usually solved separately from the main control algorithm. Furthermore, steps 5 and 6 are optional, as they include improvements not completely necessary for the controller. However, these improvements represent noteworthy advantages that demand few resources using custom hardware, so they are recommended. Each step is explained in further detail in the points below.

- *Example:* in order to clarify the methodology, an example of its application is used along the paper. The selected example is the design of the controller of a multiphase buck DC/DC converter (see Fig. 2), which has been built as a prototype and successfully controlled by the controller designed using the methodology. However, the methodology is generic and applicable to the controller of any power converter.

The chosen prototype is intended for connecting the power buses in cars with two batteries (42 and 14 V). A bidirectional 8-phase, 400W, buck converter has been constructed [7], implementing the controller in an XCV200E VirtexE Xilinx™ FPGA (the complete controller occupies 9,829 equivalent gates). Multiphase converters are a good example for showing digital controller advantages, because current loops can be avoided thanks to the driving signals precision.

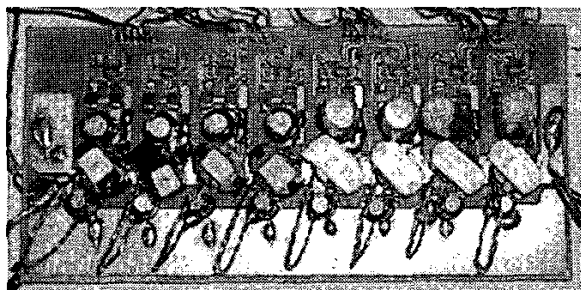
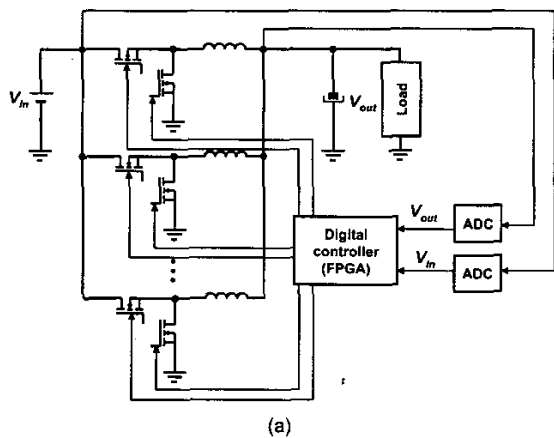


Fig. 2. Multiphase buck converter with 8 phases. Topology (a) and prototype (b).

1. Loops identification

The first thing to do is to identify the signals to be controlled, that is, the control loops. The purpose of the converter indicates which these signals are, such as output voltage in a DC-DC converter or input current and output voltage in an AC-DC converter with power factor correction. It is important to distinguish between voltage and current loops, as they are faced in a different way in the next step.

- *Example:* in the case of the multiphase buck converter, the output voltage is the signal to be controlled. Therefore, a voltage loop is needed. Besides, current distribution among the phases is a concern, so usually a current loop is needed for each phase (8 loops in the example). However, using a custom hardware digital controller, current loops can be avoided [7]. This is an important advantage of custom hardware controllers in multiphase converters, simplifying considerably the controller structure.

2. Control algorithm for each loop

The second step is to design a control algorithm for each loop. This methodology distinguishes between voltage and current loops for this purpose. The reason is not their nature, but the ripple they usually present. Voltages usually have a low switching ripple, so only one sample is used per switching cycle (additional samples would add few new information). In this case, Transfer Functions (TF) algorithms, such as linear controllers, or similar algorithms are proposed. Their common characteristic is that they calculate a new duty cycle every switching cycle, which is imposed in the following cycle, therefore introducing a one cycle delay (see Fig. 3). This is also the usual procedure for microprocessor-based controllers.

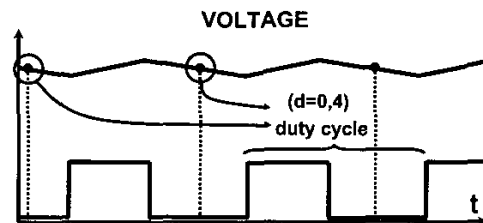


Fig. 3. Control algorithms for voltage loops.

However, most currents have a high switching ripple, so taking more than one sample per cycle do add significant information. Microprocessor-based controllers usually do not take more than one sample per cycle because they can not process all that information at the necessary speed. However, custom hardware high processing speed allows using more than one sample per cycle. In this way, the driving signal can be generated *during* the switching cycle using an On/Off algorithm (see Fig. 4), therefore achieving higher precision and bandwidth. The proposed On/Off algorithms use simple operations, such as the digital charge control proposed in [8], but executed at very high frequencies (above 100 times the switching frequency).

They are based on integrating the current samples until certain value is reached. The integration is accomplished by an adder and an accumulator register. The accumulator is compared to an objective value, opening the switch when this value is reached. Only custom hardware implementations are suitable for these algorithms, as all the operations must be executed at very high frequencies (once per sample) and in parallel. However, the operations are quite simple and demand few resources.

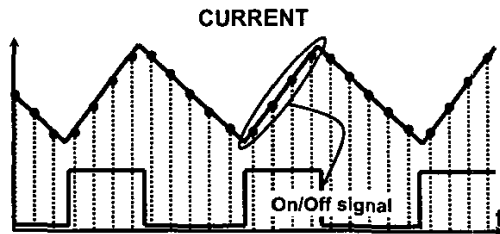


Fig. 4. Control algorithms for current loops.

In the case of TF algorithms, there are a number of methods to design the controller. An analog TF of the controller can be calculated and then discretized by different transformations (backward euler, bilinear, step invariant, pole/zero match). Other option is to use a direct digital approach (discrete TF). This leads to better results [9], but even in this case there are different choices: frequency response, Truxal method (direct synthesis) or root-locus design, which is the best choice for custom hardware digital controllers [10]. Another important issue to take into account is that the controller is usually intended for a range of conditions.

• *Example:* a PI controller was designed for the output voltage loop using root-locus. The controller must work between 5% and 100% of the nominal load, so its design was done accordingly. Different controllers can be designed for each load, but their behavior under variable load has to be considered. A trade-off between dynamics and relative stability must be reached. In this case, a controller designed for 25% nominal load was the final choice. The TF of the buck converter for 25% nominal load is:

$$G(z) = \frac{V_{out}(z)}{d(z)} = \frac{0.7038}{z - 0.9908} \quad (1)$$

Using this TF and the root-locus technique, the following PI controller was designed:

$$R(z) = \frac{d(z)}{E(z)} = \frac{0.062988 \cdot z - 0.0625}{z - 1} = \frac{(2^{-4} + 2^{-11}) \cdot z - 2^{-4}}{z - 1} \quad (2)$$

As it can be seen, its coefficients have been adjusted to powers of two (see next step).

3. Algorithms implementation using custom hardware

Once the control algorithms have been designed, they have to be implemented using custom hardware resources.

In order to achieve higher flexibility and easier implementation of complex algorithms and to reduce implementation time, a HDL (such as VHDL or Verilog) is used. Furthermore, technology independence is achieved in this way, enabling implementation with different FPGAs or even ASICs.

In the case of TF algorithms, adders and multipliers are necessary. The higher the order of the TF, the higher the number of these elements. However, if the TF coefficients are adjusted to powers of two (see equation (2)), the multipliers are substituted by shifters and adders, which are much simpler. In fact, shifters can be implemented using the appropriate connections, with no logic. If only one power of two is used per coefficient, the multiplier is substituted by a shifter, so no logic is necessary for that multiplication. If two powers of two are used for a coefficient, the multiplier is substituted by two shifters (no logic) and an adder. When three powers of two are used, two adders are necessary, etc. The maximum and mean error caused by the approximation of a random coefficient to a number powers of two has been studied in [10]. These results are synthesized in Table I. In almost all cases, 2 to 4 powers of two are enough for each coefficient. Synthesis of TFs using this technique is considerably simplified, usually achieving a reduction of 1/2 in area and 1/1.75 in the critical path.

TABLE I.
COEFFICIENT APPROXIMATION ERRORS USING POWERS OF TWO.

Number of powers of two	Maximum error	Mean error
1	33.33%	16.99%
2	14.29%	5.17%
3	6.67%	1.67%
4	3.23%	0.55%

In the case of On/Off algorithms, few resources are needed as they are based on integrating some signal (adder and accumulator register) until it reaches a reference (comparator). However, these operations must be executed at a frequency much higher than the switching frequency. This is only possible using custom hardware because of its concurrency and processing speed.

• *Example:* the algorithm to be implemented, which is basically constituted by the TF shown in equation (2), was translated into VHDL. As its coefficients were adjusted to powers of two, the complete voltage loop algorithm implementation occupied only 1,298 equivalent gates (615 gates for the TF).

4. Driving signals generation

The previous steps must be accomplished for both microprocessor and custom hardware implementations. However, driving signals generation is faced in a different way for each implementation. Microprocessor solutions need an external DPWM (Digital Pulse Width Modulator). Many DSPs include a DPWM module inside the device, but anyway it is managed as an external module because it must

be used *as is*, without changing its structure or functionality. Furthermore, usually there is only one DPWM per device. If the converter has more than one switch, additional hardware devices are needed for generating the driving signals.

Custom hardware implementations face driving signals generation like any other part of the controller. In fact, in the case of On/Off algorithms the driving signal is directly the result of the algorithm. Even using TFs, the DPWM is designed inside the device using the same kind of resources used for the control loops (see Fig. 5). Concurrency is the key feature for this, allowing the generation of multiple driving signals. Furthermore, custom hardware DPWMs achieve a higher precision because of their processing speed.

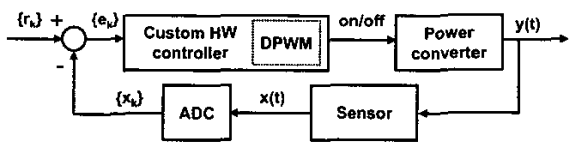


Fig. 5. Control scheme using custom hardware.

• *Example:* the high accuracy achieved in the driving signals (errors below 1 ns) when using custom hardware allows passive current sharing in multiphase converters, that is, elimination of the current loops [11]. Besides, each driving signal is shifted from the previous one (interleaving technique) in order to decrease ripple, EMI and to increase dynamic response. This can be done through adders and comparators (equivalent to shifted saw-tooth signals) or through shift-registers (equivalent to introducing a delay), as shown in Fig. 6. The first method (adders, Fig. 6 (a)) generates a structure proportional to the number of phases. However, the second method (shift-registers, Fig. 6 (b)) does not depend on the number of phases, as including more phases just needs extracting more signals from the already available shift-register. On the other hand, this method strongly depends on the duty cycle resolution, as the length of the shift-register is proportional to it. Therefore, the shift-register method needs fewer resources for low duty cycle resolution and many phases. Table II shows the boundary between both methods in terms of necessary resources. Further details can be found in [10].

TABLE II.
MINIMUM NUMBER OF PHASES FOR OBTAINING FEWER RESOURCES USING THE SHIFT-REGISTER METHOD.

Duty cycle resolution	Boundary between methods
128	6 phases
256	8 phases
512	16 phases

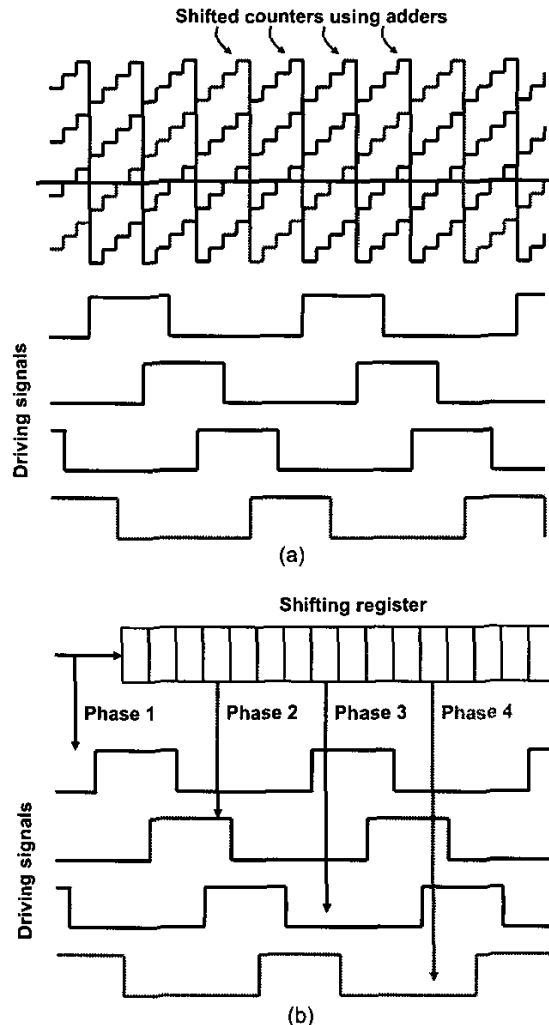


Fig. 6. Custom hardware shifters for interleaving technique. Adders (a) and shift-registers (b).

5. Samples filtering or processing (optional)

Digital controllers need A/D converters (ADCs) in order to measure analog signals. ADCs introduce some noise that can produce duty cycle variations. Therefore, it is desirable to filter the ADC samples in order to reduce this effect, but filtering introduces unwanted delay. The proposed solution is to take samples at a frequency higher than necessary and filtering them. This is possible thanks to the custom hardware high processing speed. As a result, the global sample and filtering delay is equal to or even smaller than the delay obtained when sampling at the switching frequency, but with decreased noise. Using custom hardware, filtering can be accomplished in parallel to the main control functions without affecting their performance.

• *Example:* sampling at the switching frequency without filtering, noise caused up to 7 different duty cycles once steady-state was achieved (from a total of 256 possible

solutions). Sampling 16 times faster but using a low-pass filter to achieve one noise-free sample per cycle, only 2 duty cycles were detected in steady-state.

Apart from low-pass filters, some very simple filters in terms of hardware resources can be used. For example, the intermediate value filter compares the three last samples rejecting to highest and lowest values among them. This filter only needs three registers and three comparators (see Fig. 7), but it is very useful for avoiding samples affected strongly by noise, that would otherwise change significantly the control algorithm behavior.

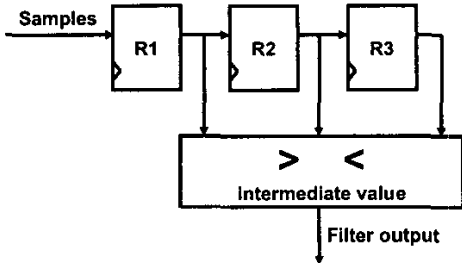


Fig. 7. Hardware structure of the intermediate value filter.

6. Protections (optional)

The final methodology step is to introduce some protections in the controller. Using a digital controller, no additional components are necessary for the protections, so they can be added without hardware changes. Furthermore, as a HDL description is employed, adding a protection is as simple as including an “if error then protection” code line. Custom hardware concurrency allows including as many protections as wanted without affecting the controller performance.

- Example: some protections implemented in the example are input or output over-voltage and maximum duty cycle.

III. DESIGN VALIDATION

The final validation of a controller is experimental results. However, in order to reduce the trial and error cycle, closed-loop simulation should be accomplished. The sooner an error is detected, the lower the effort to correct it. That is why two different simulations are considered in the methodology. These simulations are intended for three different points distributed along the methodology, as shown in Fig. 8. Below, each kind of simulation is explained in further detail.

1. Algorithmic simulation

After the second methodology step, all the control algorithms are designed, so they can now be evaluated. This is not a very exhaustive test, but just a way of making sure that the chosen algorithms are theoretically right. Using a simple converter model, an algorithmic simulation (i.e. using *Matlab*TM) can be done. This is just an approximation, as no implementation information is used, but the

algorithms can be tested in order to detect possible errors as soon as possible.

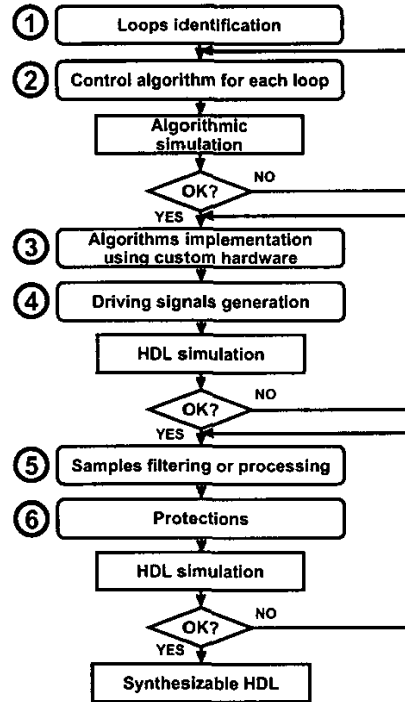


Fig. 8. Design validation included in the methodology.

2. HDL simulation

Simulating a HDL description of the controller with the power converter is a problem, as the complete system is a mixed analog-digital one (see Fig. 9). Few mixed-signal simulators are suitable for this. Most of the available choices are intended for microprocessor-based solutions, such as simulating C or C++ code with an electric model of the converter altogether (e.g. *PSIM*TM). A possible solution would be creating a C code model of the controller. However, it is desirable to use the same HDL controller model for both synthesis and simulation. In this way, simulation and experimental results are as similar as possible, at least regarding the controller behavior which is the objective of this simulation.

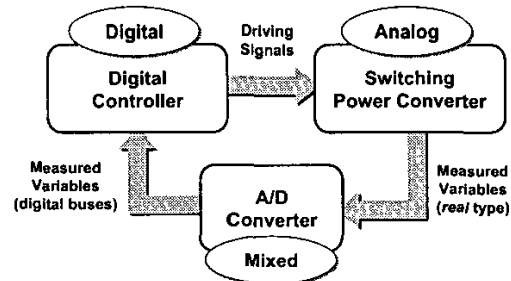


Fig. 9. Complete system for closed-loop simulation.

The proposed solution is to develop simple VHDL or VHDL-AMS (VHDL Analog and Mixed Signal extensions) models of the power converter and ADC for obtaining a closed-loop simulation [12] (Verilog models are also possible if the chosen HDL is this other one). These models can be quite simple, considering only ideal components, because their objective is evaluating the controller, not the converter. However, as the same HDL description of the controller is used for both synthesis and simulation, a high accuracy is achieved in closed-loop simulation. Even clock by clock controller behavior is observable. Furthermore, as the converters models are quite simple, closed-loop simulation runs faster than those obtained by alternate methods. In Fig. 10, a simulation of the multiphase converter used for the prototype is shown. The total current and each phase current are displayed during a start-up with different initial values for each phase current.

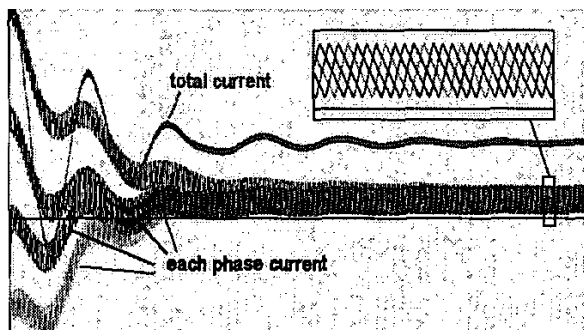


Fig. 10. Closed-loop simulation using a VHDL model of the converter. Total and phase currents.

HDL simulation can be first accomplished after the fourth step of the methodology, when the main controller functionality is fulfilled. In this point, the controller already generates the driving signals, which are necessary for closed-loop simulation as they are the inputs of the power converter model. HDL simulation can be used again after the sixth and final step, when the complete controller is designed, to test the entire design as final validation and prior to experimental results.

IV. CONCLUSIONS

This work presents a methodology to design digital controllers for switching power converters. The most significant feature is the use of *custom hardware* (FPGA or ASIC) for implementing the controller, which is especially suitable for this application due to its *concurrency* (any number of functions can be executed in parallel) and *processing speed* (especially important for high switching frequencies). The proposed methodology defines the necessary steps to design a custom hardware controller.

The methodology has been proved designing the controller of an 8-phase buck converter. Using custom hardware, a high number of phases can be controlled while

keeping a driving signal generation accuracy that enables passive current sharing (no current loop). The methodology has made possible a quick but rigorous controller design, being applicable to the design of any other power converter controller.

REFERENCES

- [1] L. Guo, J. Y. Hung, R. M. W. Nelms, "PID Controller Modifications to Improve Steady-State Performance of Digital Controllers for Buck and Boost Converters", *Applied Power Electronics Conference and Exposition (APEC)*, vol. 1, pp. 381-388, March 2000.
- [2] B. J. Patella, A. Prodic, A. Zirger, D. Maksimovic, "High-Frequency Digital Controller IC for DC-DC Converters", *IEEE Trans. on Power Electronics*, vol. 18, n. 1, pp. 438-446, Jan. 2003.
- [3] A. V. Peterchev, J. Xiao, S. R. Sanders, "Architecture and IC Implementation of a Digital VRM Controller", *IEEE Trans. on Power Electronics*, vol. 18, n. 1, pp. 356-364, Jan. 2003.
- [4] J. van den Keybus, B. Bolsens, K. de Brabandere, J. Driesen, R. Behnans, "DSP and FPGA Based Platform for Rapid Prototyping of Power Electronic Converters and its Application to a Sampled-Data Three-Phase Dual-Band Hysteresis Current Controller", *Power Electronics Specialists Conference (PESC)*, vol. 4, pp. 1722-1727, June 2002.
- [5] S. Chen, G. Joos, "Symmetrical SVPWM Pattern Generator using Field Programmable Gate Array Implementation", *Applied Power Electronics Conference and Exposition (APEC)*, vol. 2, pp. 1004-1010, March 2002.
- [6] E. S. Kim, T. J. Kim, Y. B. Byun, T. G. Koo, Y. H. Kim, "High Power Full Bridge DC-DC Converter using Digital-to-Phase-Shift PWM Circuit", *Power Electronics Specialists Conference (PESC)*, vol. 1, pp. 221-225, June 2001.
- [7] A. de Castro, P. Zumel, O. Garcia, T. Riesgo, "Digital Control in Multi-Phase DC-DC Converters", *EPE Journal*, vol. 13, n. 2, pp. 21-27, May 2003.
- [8] A. de Castro, P. Zumel, O. Garcia, T. Riesgo, J. Uceda, "Concurrent and Simple Digital Controller of an AC-DC Converter with Power Factor Correction based on an FPGA", *IEEE Trans. on Power Electronics*, vol. 18, n. 1, pp. 334-343, Jan. 2003.
- [9] Y. Duan, H. Jin, "Digital Controller Design for Switchmode Power Converters", *Applied Power Electronics Conference and Exposition (APEC)*, vol. 2, pp. 967-973, March 1999.
- [10] A. de Castro, "Custom Hardware Digital Control for Switching Power Converters", *Ph. D. dissertation*, Univ. Politécnica de Madrid, Spain, 2004.
- [11] O. Garcia, P. Zumel, A. de Castro, J. A. Cobos, J. Uceda, "An Automotive 16 Phases DC-DC Converter", *Power Electronics Specialists Conference (PESC)*, June 2004.
- [12] A. de Castro, T. Riesgo, O. Garcia, R. Prieto, "Comparing VHDL and VHDL-AMS for Modelling and Simulation of Power Converters with Digital Control", *Design of Circuits and Integrated Systems conference (DCIS)*, pp. 292-297, Nov. 2003.