



**Repositorio Institucional de la Universidad Autónoma de Madrid**

<https://repositorio.uam.es>

Esta es la **versión de autor** del artículo publicado en:

This is an **author produced version** of a paper published in:

IEEE Transactions on Industrial Informatics 8.3 (2012): 491 – 500

**DOI:** <http://dx.doi.org/10.1109/TII.2012.2192281>

Copyright: © 2012 IEEE

El acceso a la versión del editor puede requerir la suscripción del recurso  
Access to the published version may require subscription

# A comparison of simulation and hardware-in-the-loop alternatives for digital control of power converters

Alberto Sanchez, Angel de Castro, *Member, IEEE*, and Javier Garrido, *Member, IEEE*

**Abstract**—Debugging digital controllers for power converters can be a problem because there are both digital and analog components. This paper focuses on debugging digital controllers to be implemented in FPGAs or ASICs, which are designed in hardware description languages. Four methods are proposed and described. All of them allow simulation, and two methods also allow emulation — synthesizing the model of the converter to run the complete closed loop system in actual hardware. The first method consists in using a mixed analog and digital simulator. This is the easiest alternative for the designer, but simulation time can be a problem, specially for long simulations like those necessary in power factor correction or when the controller is very complex, for example with embedded processors. The alternative is to use pure digital models, generating a digital model of the power converter. Three methods are proposed: *real* type, *float* type and fixed point models (in the latter case including hand-coded and automatic-coded descriptions). *Float* and fixed point models are synthesizable, so emulation is possible, achieving speedups over 20,000. The results obtained with each method are presented, highlighting the advantages and disadvantages of each one. Apart from that, an analysis of the necessary resolution in the variables is presented, being the main conclusion that 32-bit floating point is not enough for medium and high switching frequencies.

**Index Terms**—Digital control, switching converters, field programmable gate arrays, debugging, simulation, emulation.

## I. INTRODUCTION

THE importance of debugging controllers for power converters is out of doubt. Testing a controller in actual hardware without previous simulation can result in material damages, if not bodily injuries. Recently, there has been an important growth of digital control of switching mode power supplies [1]–[5]. The debugging process for digital controllers is more complex because it is a mixed analog and digital system. This paper focuses on debugging digital controllers designed in a hardware description language (HDL), which is the common choice when they will be implemented in FPGAs (Field Programmable Gate Arrays) [6]–[10] or ASICs (Application Specific Integrated Circuits) [2], [11]. VHDL (Very high speed integrated circuit Hardware Description Language) [12] is used in the experimental results, but most conclusions can be also applied to Verilog.

Manuscript received September 19, 2011. Accepted for publication March 6, 2012. Copyright (c) 2012 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

Authors are with the HCTLab research group, Universidad Autonoma de Madrid, Madrid, Spain (e-mail: (alberto.sanchezgonzalez, angel.decastro, javier.garrido@uam.es)

The first debugging step is usually accomplished while designing the transfer function of the regulator in a control tool, such as Matlab. This is not complex, because all the parts (controller and power converter) are modeled in the same tool. However, once the controller is translated into synthesizable VHDL, it is necessary to debug it again. There are multiple reasons for simulating the VHDL implementation of the controller: possible wrong codification, checking specific implementation details of the controller not easily modeled in Matlab, such as fixed-point implementation, non-linear parts of the controller, pipeline or any other RTL (Register Transfer Level) issues, etc. So the objective is to simulate a VHDL description of the controller together with the power converter. The question is how to make this mixed signal simulation.

Simulation time is a main issue. It is important to notice that the objective at this point is not a very accurate simulation of the power converter in order to check its losses or to see the effect of parasitic components. That is a different problem in which a simulation of a few switching cycles can be enough, and the controller does not need to be in its final implementation. Our objective is to check the controller, not the power converter, and its final HDL implementation should be used in the simulation. This problem is not new, and multiple solutions have been proposed. One of the first approaches was proposed in [13]. Four alternative models with different levels of accuracy were compared, two of them using the HDL model of the controller. There are not many simulation tools that allow mixed signal models including VHDL, so there have been proposals using two simulators [14]: one for the analog part and other for the digital blocks. However, specific links between the simulators must be created. Other possibility is to make a HDL model of the power converter. The advantage of this method is obtaining faster simulations, but the obvious disadvantage is that the power converter model must be designed by hand. In [15], different models of the converter in Spice, VHDL-AMS (an analog and mixed signal extension of VHDL) and VHDL were compared for simulation, being the VHDL model the fastest one.

However, even these simulations may not be fast enough in some applications. For example, in power factor correction (PFC) the voltage loop needs simulations of hundreds of ms. Or maybe the controller has a very complex model, such as those using hardware-software techniques that have an embedded processor [16]. When simulation is too long, a solution is using Hardware-in-the-Loop (HIL) techniques. The idea is that a model of the plant is implemented in

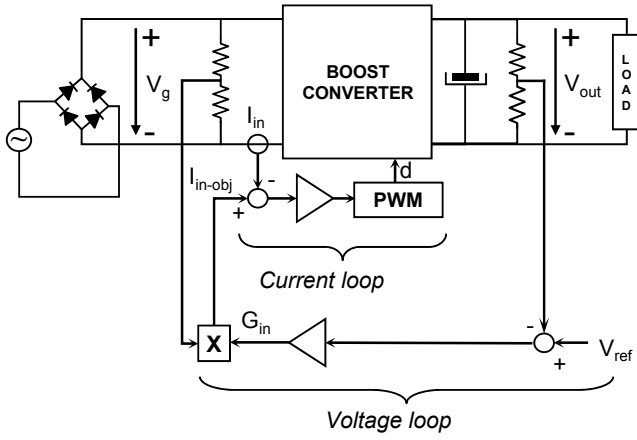


Fig. 1. PFC technique with a boost converter

digital hardware (a computer, a microprocessor or an FPGA) to emulate all the closed loop system in hardware. The first proposals used computers, but the integration step is usually in the order of hundreds of  $\mu s$ , so they are only usable in low switching frequencies applications. In [17] real time computer techniques were used to decrease the integration step to 50  $\mu s$ . In order to drastically reduce the integration step (tens or hundreds of ns), FPGAs can be used. There have been HIL implementations in FPGA, like [18]–[22]. In all these cases, low switching frequency converters were modeled using fixed point numerical representation. Fixed point obtains the best synthesis results, but increases design time. In fact, in [19], [20] they use a Matlab model that is automatically translated to VHDL. It would be easier for the designer to use floating point, but it has not been synthesizable until recent times. In [23], the use of the VHDL2008 *float\_pkg* package is proposed for HIL. In their case, the controller uses an embedded microprocessor (microBlaze), so simulations would be too long. A problem of the *float\_pkg* package is that, by the moment, is not supported by all synthesis tools.

In previous HIL proposals, only low switching frequency converters were modeled (below 10  $kHz$ ). In this paper, a 100  $kHz$  converter is used and new resolution problems arise, such as that 32-bit floating point variables do not have enough resolution. Apart from that, simulations and HIL emulations are compared using different models of the power converter in terms of simulation time and necessary resources, making a comparison of different possibilities for debugging a HDL controller. The rest of the paper is organized as follows. Section II defines the application used as example, the different models that are compared and the equations used in the models. Section III deals with implementation details, focusing on fixed point implementation, which is more difficult for the designer. Section IV presents the results, comparing the four proposed methods. Finally, conclusions are given in section V.

## II. VERIFICATION OF DIGITAL CONTROLLERS

### A. Application example

This paper presents the whole simulation process to verify digital controllers for boost converters using PFC (Fig. 1).

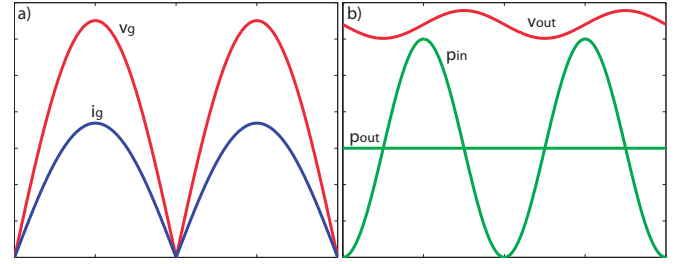


Fig. 2. a) Input current and input voltage in a PFC converter. b) Input power, output voltage and output power in a PFC converter

TABLE I  
BOOST CONVERTER PARAMETERS

Parameter	Value
$f_{sw}$	100 $kHz$
$L$	5 $mH$
$C$	100 $\mu F$
$P$	300 $W$
$V_{out}$	400 $V$

TABLE II  
REGULATORS OF THE PFC CONTROLLER

Regulator	Transfer function	Samp. period	Bandwidth	Settling time
Current	$\frac{0.5z - 0.4844}{z - 1}$	10 $\mu s$	6.330 $kHz$	472 $\mu s$
Voltage	$\frac{3.052 \cdot 10^{-5}z - 1.526 \cdot 10^{-5}}{z - 1}$	10 $ms$	6.71 $Hz$	109 $ms$
FPGA $f_{CLK} = 100 MHz$				

Regulators for PFC allow to control the output voltage ( $v_{out}$ ) of the converter, while the input current ( $i_g$ ) is proportional to the input voltage ( $v_g$ ) in order to reduce the harmonics. Therefore, there are two loops in the regulator: current and voltage loop. The former compares the input current to a reference, which is the multiplication of the input voltage and the equivalent input conductance ( $g_{in}$ ), and outputs the duty cycle of the PWM signal that must be driven to the switching MOSFET. The latter compares the output voltage to a voltage reference which is an input of the regulator, usually constant. This loop outputs the equivalent input conductance ( $g_{in}$ ), which is an input of the current loop. Fig.2 shows the evolution of input and output voltages, input current, and input and output powers. While the input power is variable because it is the multiplication of two sinusoidal waves ( $v_g$  and  $i_g$ ), the output power is more or less constant. Therefore, even in steady state, there is an unavoidable ripple in the output voltage, because the input power changes at twice the frequency of the ac mains.

The selected parameters of the boost converter used for experimental results are shown in table I. The transfer functions of the plants related to both loops are described in the literature [24]. The regulators to control the plants have been implemented in an FPGA with a clock frequency of 100  $MHz$ . The objective of this paper is not to propose new controllers for PFC, but to show how to simulate them. Therefore, classical and simple PID regulators have been designed for both loops as table II describes. Both regulators

must be simulated before testing the system using a prototype. The simulation must handle a high-frequency loop in order to check the dynamics of the input current, but it must be also long enough to check the evolution of the output voltage. In our design, the clock period of the FPGA is  $10\text{ ns}$ , the settling time of the current loop is  $472\text{ }\mu\text{s}$ , and  $109\text{ ms}$  in the voltage loop. For this reason, usual simulations must handle hundreds of milliseconds (which is equivalent to tens of millions of clock cycles), or even a few seconds.

### B. Simulation possibilities

As stated before, the problem is how to simulate the final VHDL controller together with a model of the power converter. There are several simulation possibilities to check the operation of the regulator. Mixed analog and digital simulators, such as Questa and SystemVision of Mentor Graphics, can handle simultaneously analog circuits and VHDL code, and allow modeling easily losses and electrical parasitics. However, there are few mixed AD simulators and the simulation time is very long.

Another approach is simulating the whole system in VHDL, modeling the plant in VHDL — the boost converter in our case. Whereas the regulator is natively implemented in synthesizable VHDL, the plant may be described in non-synthesizable VHDL. There are three main possibilities to model the plant in VHDL:

1) *Real* type. The plant can be modeled with the signal type called *real*, which is a floating point numeric type that is supported by simulators but cannot be implemented in hardware.

2) *Float* type. Using this type, which is implemented in the VHDL2008 *float\_pkg* package [25], the plant can be described with floating point signals and it can also be implemented in hardware using certain synthesis tools, allowing emulation but consuming many hardware resources. Both floating point simulation strategies allow small design time as it will be shown in section III.

3) *Fixed point*. This notation involves longer design time, but allows the system to be emulated in hardware using less resources than floating point emulation and allowing higher emulation frequency. The higher design time of this type of simulation is because the designer must take into account the format of every fixed point signal. Nevertheless, it is important to notice that the plant is usually modeled just once, whereas the regulator is frequently changed during the testing stage. Although fixed-point models are more complex to design, there are software tools that automatically translate high-level codes into synthesizable code. For example, an m-code of Matlab/Simulink can be translated into synthesizable code for Xilinx and Altera FPGAs using the software tools called System Generator and DSP Builder, respectively. This automatic code does not achieve the same results in speed and area compared to the code designed by hand, but the design time is much shorter.

In the following subsection, the model of a boost converter is described using VHDL with different numeric notation: fixed point, *float*, and *real* types.

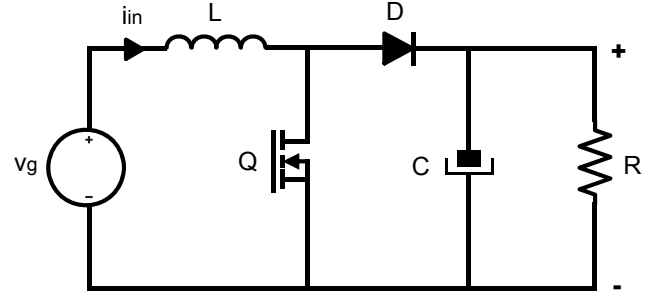


Fig. 3. Boost converter topology

### C. Model of the plant

The topology of a boost converter is shown in Fig.3. The proposed model is the simplest one, using fixed time step and therefore allowing synthesizable implementations. The model needs to calculate the output voltage ( $v_{out}$ ) and input current ( $i_L$ ) every time step, taking into account the status of the switch. The input inductor voltage is defined by (1):

$$v_L = L \cdot \frac{di_L}{dt} \quad (1)$$

Converting (1) in a difference equation, the input current for each time step  $k$  is defined in (2):

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot v_L \quad (2)$$

Likewise, the output capacitor current is defined by (3):

$$i_C = C \cdot \frac{dv_{out}}{dt} \quad (3)$$

and converting the previous to a difference equation, the output voltage for each time step  $k$  is defined by (4):

$$v_{out}(k) = v_{out}(k-1) + \frac{\Delta t}{C} \cdot i_C \quad (4)$$

$\Delta t$  is the time step of the calculus of the state variables, which is equal to the clock period ( $10\text{ ns}$ ) in our case, so  $\frac{\Delta t}{L}$  and  $\frac{\Delta t}{C}$  are constants.  $i_C$  is the current through the capacitor, which is determined by the output load.  $i_C$  is  $-i_R$  when the switch is closed, and  $i_L - i_R$  if the switch is open.  $i_R = \frac{v_{out}}{R}$  can be used if a resistive load is present, but the proposed model lets  $i_R$  as an independent variable, so any load can be modeled. When the switch is open, the input current ( $i_L$ ) can be positive so the diode does conduct (called CCM or Continuous Current Mode), or can be zero so the diode does not conduct (called DCM or Discontinuous Current Mode). Thus, there are three possibilities (closed switch, open switch in CCM or open switch in DCM) which are described respectively in (5), (6) and (7):

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot v_g \\ v_{out}(k) &= v_{out}(k-1) - \frac{\Delta t}{C} \cdot i_R \end{aligned} \quad (5)$$

TABLE III  
SIGNED QX.Y SIGNAL FORMATS

Signal	Number of bits	Format	Scale	Equivalent range (3 decimal places)	Resolution
$v_g$	13	9.3	-	$\pm 511.875 \text{ V}$	$0.125 \text{ V}$
$v_{out}$	13	9.3	-	$\pm 511.875 \text{ A}$	$0.125 \text{ V}$
$i_R^*$	13	22.-10	$\frac{\Delta t}{L}$	$\pm 8.387 \text{ A}$	$2.048 \cdot 10^{-3} \text{ A}$
$v_{out}^*$	34	43.-10	$\frac{\Delta t}{L} \frac{\Delta t}{C}$	$\pm 1,759.219 \text{ V}$	$2.048 \cdot 10^{-7} \text{ V}$
$v_{out}Sat^*$	18	43.-26	$\frac{\Delta t}{L} \frac{\Delta t}{C}$	$\pm 1,759.205 \text{ V}$	$0.013 \text{ V}$
$i_L^*$	26	22.3	$\frac{\Delta t}{L}$	$\pm 8.389 \text{ A}$	$2.500 \cdot 10^{-7} \text{ A}$
$i_LSat^*$	18	22.-5	$\frac{\Delta t}{L}$	$\pm 8.389 \text{ A}$	$6.400 \cdot 10^{-5} \text{ A}$

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot (v_g - v_{out}) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \quad (6)$$

$$\begin{aligned} i_L(k) &= 0 \\ v_{out}(k) &= v_{out}(k-1) - \frac{\Delta t}{C} \cdot i_R \end{aligned} \quad (7)$$

One of these three equation sets must be calculated each time step, involving two multiplications.

In the hand-coded fixed point model of the plant, some transformations are used in order to speed up the calculus of the equations. Instead of calculating  $i_L$  and  $v_{out}$ ,  $i_L^*$  and  $v_{out}^*$  are calculated using the transformations shown in (8) and (9):

$$i_L^* = \frac{L}{\Delta t} \cdot i_L \quad (8)$$

$$v_{out}^* = \frac{C}{\Delta t} \cdot v_{out} \quad (9)$$

Consequently, applying these transformations to the fixed point model, (10) and (11) are used:

$$i_L^*(k) = i_L^*(k-1) + v_L \quad (10)$$

$$v_{out}^*(k) = v_{out}^*(k-1) + i_C \quad (11)$$

These equations do not use multiplications, so the maximum working frequency is higher and they use less hardware resources. Again,  $v_L$  and  $i_C$  depend on the switch and conduction mode, so the equations to be implemented in hardware are:

$$\begin{aligned} i_L^*(k) &= i_L^*(k-1) + v_g \\ v_{out}^*(k) &= v_{out}^*(k-1) - i_R \end{aligned} \quad (12)$$

$$\begin{aligned} i_L^*(k) &= i_L^*(k-1) + v_g - v_{out} \\ v_{out}^*(k) &= v_{out}^*(k-1) + i_L - i_R \end{aligned} \quad (13)$$

$$\begin{aligned} i_L^*(k) &= 0 \\ v_{out}^*(k) &= v_{out}^*(k-1) - i_R \end{aligned} \quad (14)$$

### III. IMPLEMENTATION

This section describes the implementation process of every model that has been proposed: mixed AD, *real type*, *float type* and fixed-point. Mixed analog and digital simulation is the simplest method for the designer. This type of simulators allows to draw a circuit using the *drag-and-drop* method with components such as capacitors, inductors, ADCs, etc. Usually, these drag-and-drop components are VHDL-AMS (an extension of VHDL for analog and mixed-signal) models, but that is transparent for the designer. Besides, the simulators handle VHDL entities which in our case is the regulator to be checked. The implementation and simulation of this system is not complex but the simulation time is very long.

The boost model based on *real* signals is implemented using the difference equations (5), (6) and (7) presented in the previous section. This model is also simple, using four multiplexers and two multipliers apart from several adders and registers, but must be coded by hand. This model cannot be implemented in hardware because *real* type is not synthesizable, but it can be simulated in any VHDL simulator.

Using the *float* type, the model can be simulated and also synthesized. This model is almost identical to the *real* type model, but using the *float* type which is defined in the package *float\_pkg* of the VHDL-2008 Support Library. This package provides the floating point notation described in the standard IEEE 754, but is not very extended and only few synthesizers can handle it by the moment. In our case, we have used Synplify Premier of Synopsys. The package provides floating type signals of 32 and 64 bits, but in order to reduce the required hardware resources, which is one of the main disadvantages of the *float* model, 32 bit signals have been used.

Finally, the model can be implemented using fixed point notation. The first possibility is to create a model of the boost converter in Matlab/Simulink using m-code files and then translate these high-level files into synthesizable HDL code. For this purpose, System Generator can be used if the code will be implemented in Xilinx FPGAs and DSP Builder if Altera FPGAs will be used. This high-level model can implement directly the equations (5), (6) and (7), so the design time is almost as small as in the previous cases. The designer should only specify the number of bits of the accumulators of  $v_{out}$  and  $i_L$ , so the code translator can optimize the area of the

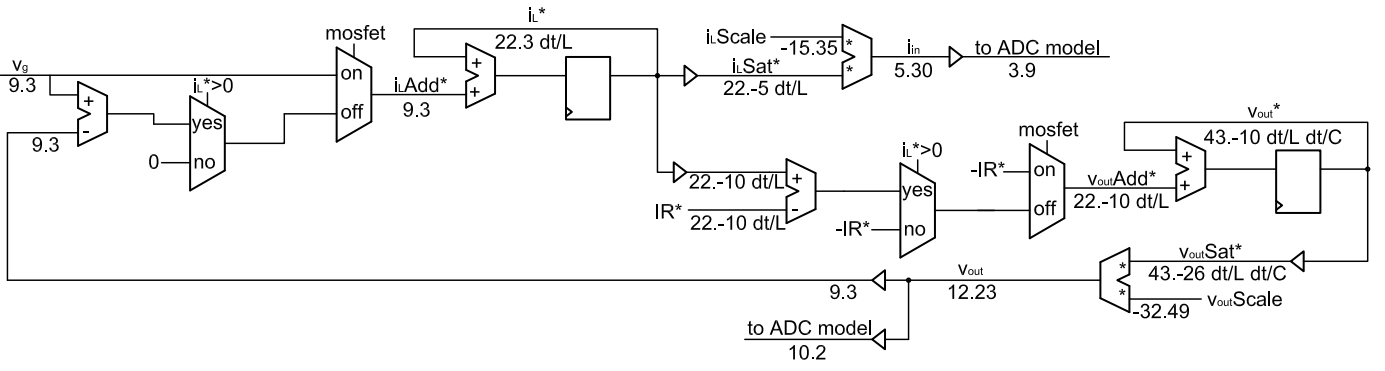


Fig. 4. Schematic of the implemented circuit

code. Nonetheless, the results of area and speed of automatic code can be improved if the fixed-point model is coded by hand, but increasing the design time.

The fixed point model coded by hand uses the difference equations (12), (13) and (14). Fixed point signals have been implemented using QX.Y notation. A QX.Y signal has X bits in the integer part and Y bits in the decimal part. As 2's complement is used, there is one extra (MSB, most significant bit) sign bit. For instance, a Q9.3 signal has 1+9+3 bits. Table III shows the format, scale and equivalent range of the internal signals of the QX.Y boost model. To translate the value of a QX.Y signal, its value must be multiplied by the scale of the signal and by  $2^{-Y}$ . Some signals, like  $v_{out}$ , do not have any scale, so they only need to be multiplied by  $2^{-Y}$  to get their value in volts or amperes. For instance, the value "0100000000001" in  $v_{out}$  represents 256.125 V. However,  $i_R$  has the same number of bits but its QX.Y format is different and it also has a scale. The scale is a constant that has to be multiplied by the stored value before obtaining the final value in volts or amperes.  $\Delta t$  is the integration step, which is the inverse of the FPGA clock frequency (10 ns), and  $L$  and  $C$  are shown in table I. These scales are due to the transformations shown in (8) and (9) used for simplifying the resulting hardware of the model. Therefore, the same value "0100000000001" for  $i_R$  represents 4.196352 A. Finally, table III also gives the range of each signal and its resolution, i.e. the LSB (least significant bit) value.

Fig.4 shows the schematic of the implemented model in QX.Y. The left part represents the hardware needed to calculate the input current.  $i_LAdd^*$  is the quantity to be added to the previous value each clock cycle, which is chosen with two multiplexers depending on the switch status and conduction mode. This quantity is in volts, so its direct addition to the previous current value implies the transformation shown in (8). Therefore,  $i_L^*$  is in a different scale. This internal variable uses 26 bits for avoiding resolution problems. This input current, together with the load current  $i_R^*$ , are the inputs to the hardware that calculates the output voltage (lower right part of the figure). However, not all the 26 bits are used, but the 13 MSB, using the same range and scale of  $i_R^*$  so they can be directly subtracted. Again, two multiplexers choose the value  $v_{outAdd}^*$  to be added to the previous voltage sample. Adding a current to the previous output voltage sample implies

a new transformation, as shown in (9). Therefore,  $v_{out}^*$  has a double scale. The loop is closed because the output voltage is needed for calculating the input current. However, the transformations must be undone before subtracting the output voltage, with a double scale, to the input voltage, with no scale. This is done multiplying by the scale.  $v_{out}^*$  is represented with 34 bits for avoiding resolution problems, but hardware multipliers in Spartan-3 FPGAs only use 18 bits, so only the 18 MSB ( $v_{outSat}^*$ ) are used in the multiplication. After the multiplication by the scale, the output voltage is in volts, with no scale. The final step is truncating it to 13 bits, as in the model input  $v_g$ , because they must be subtracted. Finally, the input current is also multiplied by its scale, also using 18 bits, to get the current in amperes, which is an output of the model. However, this second multiplier (upper right part of the figure) is not in the critical path because its output is not feedback to the model, so the maximum frequency is not affected by it. The VHDL model of this schematic, together with the VHDL models using *real* and *float* types, and the rest of necessary files for closed loop simulation or emulation can be downloaded from [26].

The model has two outputs,  $v_{out}$  and  $i_{in}$ , which are sent to the ADC models, and three inputs: *mosfet* (on or off state of the switch),  $v_g$  and  $i_R^*$ . *mosfet* is the output of the controller, and  $i_R^*$  is left as an independent input, so any load can be modeled. However,  $v_g$ , which is the rectified AC mains, will be in most cases an always positive sinusoidal wave. In order to simplify the emulation of the whole system, it is pre-calculated in 1,000 time steps, so in every switching cycle the value of  $v_g$  is loaded from a BRAM (Block RAM) of the FPGA. This memory can be used not only for the hand-coded fixed point model but also with the automatic fixed point and floating point models using *float* type.

The HIL approach requires the extraction of emulation data to be traced. One possibility is to output the desired data to FPGA pins so they can be read by an external digital analyzer or sent to digital-to-analog converters. Another approach is to add a VHDL digital analyzer which sends the data through the programmer cable. In our case, we have used a Xilinx ChipScope module, which is a soft-core digital analyzer. The simplest topology of this analyzer is an ILA (Integrated Logic Analyzer) which traces the desired internal signals, and an ICON (Integrated CONTroller) which is the interface between

the ILA component and the JTAG programmer cable. After a trigger, the ILA module stores a predefined number of samples of the desired signals in the internal BRAMs of the FPGA. Accordingly, the ICON module reads these data and sends them to the PC for visualization and debugging. In our case, the values of the input current and the output voltage are stored in the BRAMs, with 11 bits each one. When the trigger is on, 16,384 samples are taken, one every switching cycle. With this configuration, the information of 16 rectified line cycles can be extracted. The ChipScope module uses 23 BRAMs to store these data. The chosen FPGA has 24 BRAMs, but one is reserved for generating the input voltage.

#### IV. RESULTS

A comparison of all the simulation and emulation approaches has been accomplished. The mixed analog and digital simulation has been implemented with SystemVision of Mentor Graphics. The *real* model has been simulated with Modelsim 6.5b of Mentor Graphics. And finally, the *float* and fixed point models have been both simulated with Modelsim and implemented in a Xilinx FPGA.

The first comparison criterion is if the model can be only be simulated or both simulated and emulated, which is synthesizing the model and implementing it in an FPGA for debugging the whole closed-loop system inside the FPGA (HIL). The main advantage of emulation is that it is much faster than simulation, as will be shown below. The mixed model and the *real* type model can only be simulated, while the *float* type and fixed point models can be both simulated and emulated, which is the main advantage of these models. However, regarding design effort, the mixed model is the easiest because a graphic schematic is enough, with no code typing for the power converter model. The *real* and *float* models do need code typing, but they are direct translations from the difference equations without worrying about data widths or resolution. The System Generator model uses a direct translation from the difference equations but also needs to know the format (number of bits) of the input and output signals of the boost model, so the regulator can handle them, and the accumulators representing the state variables. Finally, the QX.Y model (hand-coded) is the hardest one, because the designer must also worry about all data widths and resolution. However, it must be highlighted that all these methods are proposed for debugging a VHDL controller, which will be almost for sure a QX.Y model in order to make it work at the objective clock frequency, 100 MHz in our case. Therefore, the design effort is not so high because the designer is already familiar with QX.Y models.

A critical comparison criterion is simulation time. We have to take into account that the FPGA clock is 10 ns, but the settling time of the voltage loop is 109 ms, so millions of clock cycles are necessary. Table IV shows the time results of simulation and emulation of the different models when simulating 200 ms. This would be a basic simulation, but if multiple load steps need to be simulated, it can easily go into seconds. Although the HIL systems (*float* and fixed point) have been designed to be implemented in hardware,

TABLE IV  
TIME RESULTS OF A SIMULATION OF 200 MS

System	Simulation/Emulation	Time	Speedup
Mixed simulation	Simulation	2h 13' 21" 751 ms	Reference
"Real" type	Simulation	2' 14" 646 ms	59.4x
"Float" type	Simulation	2h 5' 14" 438 ms	1.1x
"Float" type	Emulation	3" 228 ms	2,478.9x
System Generator	Simulation	14' 45" 264 ms	9.0x
System Generator	Emulation	501 ms	15,971.6x
QX.Y	Simulation	2' 24" 871 ms	55.2x
QX.Y	Emulation	277 ms	28,887.2x

TABLE V  
FPGA (XILINX XC3S1000) RESOURCES USED BY THE DESIGN

System	Max freq	4 input LUTs	FFs	Mult 18x18	BRAMs (16 kB)
Boost model: QX.Y (XST synthesizer)	68.747 MHz	170	60	2	0
Boost model: QX.Y (Synplify synthesizer)	61.584 MHz	380	79	0	0
Boost model: System Generator (XST synthesizer)	43.273 MHz	409	72	3	0
Boost model: floating point (Synplify synthesizer)	6.103 MHz	7,355	76	0	0
HIL: QX.Y (XST synthesizer)	68.781 MHz	447	361	4	1
HIL: QX.Y (Synplify synthesizer)	56.497 MHz	658	358	1	1
HIL: System Generator (XST synthesizer)	42.073 MHz	755	391	5	1
HIL: floating point (Synplify synthesizer)	6.085 MHz	9,332	392	1	1
HIL w/ CS: QX.Y (XST synthesizer)	72.202 MHz	814	665	4	24
HIL w/ CS: QX.Y (Synplify synthesizer)	55.121 MHz	1,036	662	1	24
HIL w/ CS: System Generator (XST synthesizer)	39.861 MHz	1,122	697	5	24
HIL w/ CS: floating point (Synplify synthesizer)	6.196 MHz	9,412	685	1	24

simulations have been also performed. Simulation times have been measured in a 2.33 GHz Intel Core 2 Duo E6550 with 4 GB of RAM. Emulation times are extracted from the maximum clock frequency of each model, taking into account that real time is reached at 100 MHz. Speedups are related to mixed signal simulation. Emulations are much faster than simulations, obtaining a speed-up of 28,887.2x using QX.Y and about ten times slower using *float* type. However, the speedup of *float* emulation is enough (a few seconds for each emulation) and the model is much simpler than QX.Y. The main advantages of QX.Y are area and resolution, as shown below. Comparing simulation times, not emulations, *real* type and QX.Y are more than 50 times faster than mixed signal simulation, involving a few minutes instead of hours. *float* type simulation is almost as slow as mixed signal because the floating point hardware that is simulated is very complex, so it makes no sense to use *float* type for simulation.

Both emulation systems must be synthesized, and their synthesis results are very different. As the XST synthesizer of Xilinx ISE 12.3 cannot compile the *float\_pkg*, Synplify Premier E2011 of Synopsys has been used. However, the system which uses QX.Y notation has been implemented both with Synplify and XST, which gives better results in this case and is included in the Xilinx ISE tool. Table V presents the synthesis results of the emulation systems after implementation in a Xilinx XC3S1000 FPGA, which is a

low cost FPGA. The table shows the results in area and speed. Three different synthesis have been carried out. 1) Only the boost model, because that is the part that changes from case to case, as the controller is the same in all cases. 2) The whole HIL system, which includes the boost model and the controller (composed of two simple PID regulators), but without including debugging hardware. This is the minimum configuration for closed loop emulation. 3) The whole HIL system and a ChipScope module for debugging. As the table shows, the fixed point models need much fewer hardware resources than the *float* model, and the maximum frequency is about ten times faster. The reason is that floating point adders and multipliers are much more complex than fixed point ones. As a conclusion, if area is an important restriction, fixed point models would be the preferred option. Regarding both fixed point models (QX.Y, which is hand-coded, and the code automatically generated by System Generator), it can be seen that the hand-coded implementation is quite smaller (about half size only for the boost model), and its maximum frequency is about 50% higher. Therefore, there is a trade-off between design effort (higher for hand-coded fixed point) and necessary resources (higher for floating point), with the fixed point model created by System Generator as a point in between. Apart from that, the maximum frequency of the QX.Y system using XST is about 30% greater than using Synplify. The reason is that XST synthesizer uses more dedicated multipliers (MULT18x18) instead of implementing them with LUTs. This is because Synplify uses LUTs for multiplications by a constant, because the multiplier can be somewhat simplified in this particular case, but the results are not as good as a dedicated multiplier. Finally, a comment about maximum working frequencies. As it can be seen, they are almost the same from the boost model to the complete system with ChipScope, indicating that the critical path is in the boost model. In some cases, adding additional resources (such as ChipScope) result in a slightly higher maximum frequency, which is in principle contradictory. The reason is the pseudo-random part of the place and route algorithm, which can produce slightly different results in the maximum working frequency even if the critical path does not change.

A comment about FPGA clock frequency is necessary. It was said that the clock frequency was 100 MHz, and  $\Delta t$  (the integration step) is therefore 10 ns. However, the emulation models do not reach 100 MHz, which means that they will not run at real time. However, the final implementation of the controller, not the boost model, will run at 100 MHz, so 10 ns is also the duty cycle resolution.

Another very important comparison criterion is accuracy. If the simulation results are not the correct ones, no speedup can compensate for that. An experiment has been performed using the whole system in closed loop for power factor correction. All the models have been used in this experiment and a prototype has been designed and built for comparison purposes. In this experiment the output of the voltage loop ( $g_{in}$ ) has been extracted. Steady state  $G_{in}$  has been selected because it is affected both by the calculus of the input current and the output voltage of the model, so it allows to use a single parameter to test the accuracy of the whole system. If the

TABLE VI  
ACCURACY OF THE MODEL - PFC CONVERTER

System	Simulation Emulation	$G_{in}$	$G_{in}$ error related to ideal $G_{in}$
Ideal $G_{in}$ without losses		0.00567108	
Experimental results		0.00564575	-0.45%
Mixed simulation	Simulation	0.00576782	+1.71%
"Real" type	Simulation	0.00565338	-0.31%
32-bit "Float" type	Sim/Emulation	0.00512314	-9.66%
QX.Y	Sim/Emulation	0.00564957	-0.38%
System Generator	Sim/Emulation	0.00565338	-0.31%
Results taken in steady state with $V_{out}$ reference set to 400 V			

output voltage calculation has inaccuracies, the voltage loop will modify  $g_{in}$  to compensate the error. But if the error comes from the input current, there will be a power unbalance that will be also compensated modifying  $g_{in}$ . Therefore, the steady state  $G_{in}$  value is a good parameter for checking inaccuracies in any part of the model.

Table VI shows  $G_{in}$  values in steady state, comparing them to ideal  $G_{in}$ . This can be calculated as  $G_{in} = \frac{P}{V_g^2}$ , representing a model without losses. The mixed simulation is not an ideal model, but includes some parasitic elements. That is why  $G_{in}$  in this model is 1.7% higher to compensate the losses. The rest of the models do not include losses, except of course the experimental results. As we are trying to check if the implementation of the different models affect accuracy, the comparison is made with the ideal case because these other models do not include losses. The *real* type, which uses floating point of double precision (64 bits), achieves the most accurate value of  $G_{in}$  (the error is 0.31%). The same result is achieved with the System Generator model, which includes enough bits to store the variables of the boost converter without losing precision in the internal calculations. The QX.Y model has almost the same accuracy (0.38%), but not exactly the same because it uses less bits to store internal values. The number of bits has been chosen so almost no precision is lost, but keeping the hardware implementation as simple as possible.

The  $G_{in}$  value of the prototype should be a bit higher due to the electrical losses, but it is lower, due to measurement inaccuracies. For instance, if the gain of the ADCs is not exactly equal to the calculated gain, the  $G_{in}$  parameter of the regulator diverges upward or downward. Therefore, in this case the experimental  $G_{in}$  is 0.45% lower than the expected input conductance due to these measurement inaccuracies. An important conclusion is that the accuracy of the previously presented models is even higher than the inherent measurement errors that will appear in real conditions. Therefore, the *real* and fixed point models have enough accuracy.

However, results show that the *float* type presents an error of 9.66%, much higher than fixed point. The reason is that *float* type uses 32-bit signals, which do not have enough resolution to store the incremental values of  $v_{out}$  and  $i_{in}$  in this case. For instance, typical incremental values of  $v_{out}$  are around  $7.5 \cdot 10^{-5}$  V (equation (5)), while  $v_{out}$  is around 400 V. The *float* type uses 24 bits for the mantissa: a fixed '1' and 23 additional bits, while the QX.Y model uses 34 bits for  $v_{out}^*$  and 26 for  $i_L^*$  (table III). For a  $v_{out}$  value around 400 V,

the MSB in floating point is  $2^8$ , so the LSB is  $2^{-15}$ , i.e.  $3.05 \cdot 10^{-5}$ . This is in the same order of magnitude of the incremental value. For instance,  $7.5 \cdot 10^{-5}$  would have to be rounded to twice the LSB, i.e.  $6.1 \cdot 10^{-5}$ . That is why there is an error in  $G_{in}$  of about 10%. There are two solutions to this problem. One would be to use float signals of 64 bits, but the resulting hardware would be enormous and only high-end FPGAs could be used. The other solution is to increase the parameter  $\Delta t$  so the incremental values are greater. However, this decreases the accuracy of the system as  $\Delta t$  increases, because it is the integration step. In our case, using  $\Delta t$  equal to 10 ns and 100 kHz as the switching frequency, the duty cycle resolution is 0.1%. If  $\Delta t$  was increased to 100 ns, the duty cycle resolution would drop to 1%. The conclusion is that *float32* is not appropriate for hardware-in-the-loop emulation of medium or high switching frequency power converters, in which  $\Delta t$  must be small. In previous examples of the state of the art, lower frequencies were used and precision of floating point variables with 32 bits was not an issue. However, if HIL is going to be used in higher frequency applications, resolution must be studied with care.

It can be surprising that the QX.Y model achieves good accuracy when some of its signals are represented with 13 bits. For instance,  $v_g$  is represented with only 13 bits. However, this signal will be sampled by an ADC in the final system, so 13 bits including the sign bit is enough, representing a 12-bit ADC. The resolution problem comes when two very different values are added. This is true for the difference equations. As said before, values of  $v_{out}$  around 400 V must be added to incremental values each integration step that can be around  $7.5 \cdot 10^{-5}$  V. Therefore, a high number of bits is used for these variables,  $v_{out}^*$  and  $i_L^*$  in our case. However, the inputs and outputs of the model use fewer bits. This is an advantage of the fixed point models, that only use the necessary bits for each variable, decreasing the hardware resources.

Another experiment to check accuracy is the simulation of a load step. In this way, we can check the dynamic behavior of each model compared to the real behavior in the prototype. Figure 5 shows the behavior of  $v_{out}$  when a step in the load from 136 to 296 W takes place after the system has previously achieved steady state. As it can be seen, the most similar response to the prototype is the mixed-simulation. This is expected, because the mixed-simulation is the only one that includes electrical losses and other non-idealities of the system. The *real* and QX.Y models have a very similar response between them because both have enough accuracy but they do not model electrical losses. Therefore, the output voltage in these models has a somewhat lower dumping than the real case. Finally, the *float* model presents the worst dynamic behavior. As it was explained before, the *float32* model has not enough resolution to model high-frequency systems. The steady state is very similar in all cases because it is a close-loop simulation. However, dynamic responses do differ depending on the accuracy of the model. The System Generator model has also been tested in this experiment, and its dynamic response is almost identical to the *real* model. Because of this, and for the sake of clarity, the waveform of the System Generator model has not been included in this

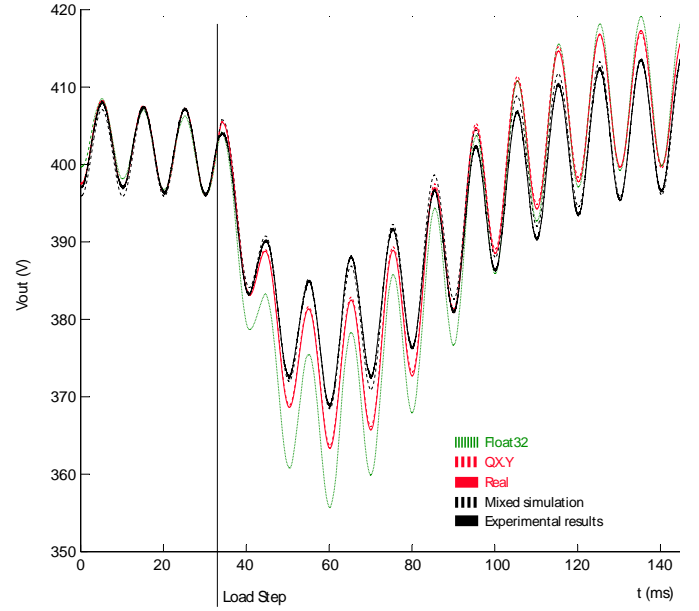


Fig. 5. Comparison of the proposed systems after load step from 1176  $\Omega$  to 540  $\Omega$  ( $v_{outRef} = 400$  V)

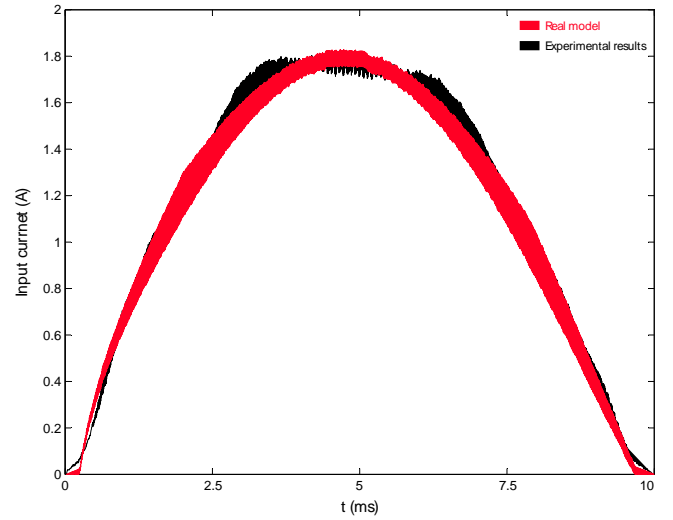


Fig. 6. Waveform of the input current at steady state in the prototype and the *real* model with the optimal regulator

figure.

Finally, another experiment has been accomplished in order to estimate the accuracy of the models. Figures 6 and 7 show the waveforms of the input current at steady state in the prototype and the *real* model simulation (the fixed point models have almost identical results to the *real* model which are not shown for the sake of clarity). The experimental waveforms present higher noise, but both experimental and simulated waveforms have very similar behavior. In Fig. 6 the converter is in nominal situation and the current regulator is the optimal one. In this case, both the experimental and simulated waveforms have the peak of the input current located about 0.5 ms before the ideal point, which would have been in the middle point, at 5 ms. The power factor of both experiments has been extracted: 0.9967 in the prototype and 0.9964 in the

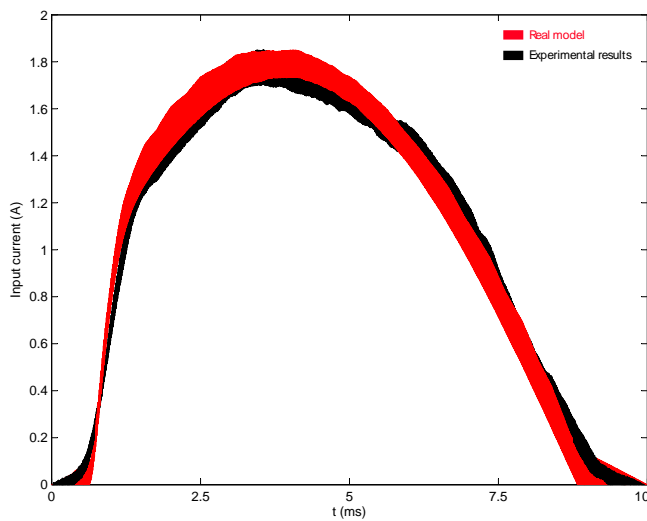


Fig. 7. Waveform of the input current at steady state in the prototype and the *real* model when the regulator has a quarter of the optimal gain

*real* model, which are almost identical. Fig. 7 shows the same experiment, but with a non-ideal current regulator, which has one quarter of the optimal gain. The input current is clearly non-ideal in this case, but the simulation waveform is almost identical to the experimental one. The conclusion is that the proposed simulation techniques are suitable to evaluate the performance of the regulator before being implemented in a real prototype.

## V. CONCLUSIONS

This paper has presented different alternatives for debugging digital controllers to be implemented in FPGAs or ASICs, which are designed in hardware description languages like VHDL. The main difficulty is simulating the VHDL controller together with the power converter, which is analog. The first possibility is a mixed analog and digital simulator. This is the easiest alternative for the designer, but simulation time can be a problem for those cases in which long simulations are necessary, like PFC, or the model of the controller is very complex, like embedded microprocessors. The alternative is to generate a digital model of the power converter. In this way, it is possible to obtain pure digital and faster simulations or even emulations: synthesizing the model of the converter to run the complete closed loop system in digital hardware (HIL). Three digital models have been presented. Using the *real* type, the model is non synthesizable but the design is straight forward, without worrying about resolution, and the simulation runs more than 50 times faster. If this speedup is not enough, emulation is necessary and the model must be synthesized. The *float* type model is almost identical to the *real* one, but can be synthesized. This emulation obtains a speedup over 2,000, which should be enough in all cases. However, its simulation is slow, not all synthesis tools support it, it needs many hardware resources and, most important, it has accuracy problems, specially when high switching frequencies are involved. All these problems are solved by a fixed point model. Its main disadvantage is its high design effort, at least

when the model is hand-coded. The design effort of fixed point models can be diminished using automatic generation of the VHDL code from a high-level model, like System Generator. However, the design effort of the model should not be a problem, because in most cases it will be a model similar to the controller, usually also designed in fixed point. Fixed point emulation speedup is over 20,000 for emulation and also over 50 for simulation. A comparison between all these models and a real prototype has been accomplished, and results demonstrate that the simulation of the models is accurate enough to show the dynamics and the steady state variables of the converter except for the *float32* model when high switching frequencies are involved. Therefore, the final decision of which model should be used depends on a trade-off between design effort, simulation/emulation speedup and available hardware resources.

## REFERENCES

- [1] B. Patella, A. Prodic, A. Zirger, and D. Maksimovic, "High-frequency digital PWM controller IC for DC-DC converters," *Power Electronics, IEEE Transactions on*, vol. 18, no. 1, pp. 438–446, Jan 2003.
- [2] A. Peterchev, J. Xiao, and S. Sanders, "Architecture and IC implementation of a digital VRM controller," *Power Electronics, IEEE Transactions on*, vol. 18, no. 1, pp. 356–364, Jan. 2003.
- [3] T. D. Nguyen, J. Hobraiche, N. Patin, G. Friedrich, and J. Vilain, "A direct digital technique implementation of general discontinuous pulse width modulation strategy," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 9, pp. 4445–4454, Sept. 2011.
- [4] E. Vidal-Idiarte, C. Carrejo, J. Calvente, and L. Martínez-Salamero, "Two-loop digital sliding mode control of DC-DC power converters based on predictive interpolation," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 6, pp. 2491–2501, June 2011.
- [5] M. Kazmierkowski, M. Jasinski, and G. Wrona, "DSP-based control of grid-connected power converters operating under grid distortions," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 2, pp. 204–211, May 2011.
- [6] E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, and M. Naouar, "FPGAs in industrial control applications," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 2, pp. 224–243, May 2011.
- [7] E. Monmasson, L. Idkhajine, and M. Naouar, "FPGA-based controllers," *Industrial Electronics Magazine, IEEE*, vol. 5, no. 1, pp. 14–26, March 2011.
- [8] J. Rodriguez-Andina, M. Moure, and M. Valdes, "Features, design tools, and application domains of FPGAs," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1810–1823, Aug. 2007.
- [9] J. Alvarez, O. Lopez, F. Freijedo, and J. Doval-Gandoy, "Digital parameterizable VHDL module for multilevel multiphase space vector PWM," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 9, pp. 3946–3957, Sept. 2011.
- [10] F. Taeed, Z. Salam, and S. Ayob, "FPGA implementation of a single-input fuzzy logic controller for boost converter with the absence of an external analog-to-digital converter," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 2, pp. 1208–1217, Feb. 2012.
- [11] Z. Lukic, N. Rahman, and A. Prodic, "Multibit sigma-delta PWM digital controller IC for DC-DC converters operating at switching frequencies beyond 10 MHz," *Power Electronics, IEEE Transactions on*, vol. 22, no. 5, pp. 1693–1707, Sept. 2007.
- [12] F. Azcondo, A. de Castro, and C. Branas, "Course on digital electronics oriented to describing systems in VHDL," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 10, pp. 3308–3316, Oct. 2010.
- [13] A. Prodic and D. Maksimovic, "Mixed-signal simulation of digitally controlled switching converters," in *Computers in Power Electronics, 2002. Proceedings. 2002 IEEE Workshop on*, June 2002, pp. 100–105.
- [14] P. Zumel, M. Garcia-Valderas, A. Lazaro, C. Lopez-Ongil, and A. Barrodo, "Co-simulation PSIM-ModelSim oriented to digitally controlled switching power converters," in *Control and Modeling for Power Electronics (COMPEL), 2010 IEEE 12th Workshop on*, June 2010, pp. 1–7.
- [15] L. Barragan, I. Urriza, D. Navarro, J. Artigas, J. Acero, and J. Burdío, "Comparing simulation alternatives of FPGA-based controllers for switching converters," in *IEEE International Symposium on Industrial Electronics (ISIE)*, June 2007, pp. 419–424.

- [16] O. Lucia, L. Barragan, J. Burdio, O. Jimenez, D. Navarro, and I. Urriza, "A versatile power electronics test-bench architecture applied to domestic induction heating," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 3, pp. 998–1007, March 2011.
- [17] B. Lu, X. Wu, H. Figueroa, and A. Monti, "A low-cost real-time hardware-in-the-loop testing approach of power electronics controls," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 2, pp. 919–931, April 2007.
- [18] M. Matar and R. Irvani, "FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients," *Power Delivery, IEEE Transactions on*, vol. 25, no. 2, pp. 852–860, April 2010.
- [19] G. Parma and V. Dinavahi, "Real-time digital hardware simulation of power electronics and drives," *Power Delivery, IEEE Transactions on*, vol. 22, no. 2, pp. 1235–1246, April 2007.
- [20] A. Myaing and V. Dinavahi, "FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 1, pp. 358–368, Jan. 2011.
- [21] S. Karimi, P. Poure, and S. Saadate, "An hil-based reconfigurable platform for design, implementation, and verification of electrical system digital controllers," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 4, pp. 1226–1236, April 2010.
- [22] Y. Chen and V. Dinavahi, "Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 2, pp. 1300–1309, Feb. 2012.
- [23] O. Lucia, I. Urriza, L. Barragan, D. Navarro, O. Jimenez, and J. Burdio, "Real-time FPGA-based hardware-in-the-loop simulation test bench applied to multiple-output power converters," *Industry Applications, IEEE Transactions on*, vol. 47, no. 2, pp. 853–860, March-April 2011.
- [24] R. Erickson and D. Maksimovic, *Fundamentals of power electronics*. Kluwer Academic, 2001.
- [25] [Online]. Available: <http://www.eda.org/fphdl/>
- [26] [Online]. Available: <http://www.hctlab.com/hil/emulationFiles.rar>



**Javier Garrido** (M'97) was born in Madrid, Spain, in 1954. He received the B.Sc. degree in 1974, the M.Sc. degree in 1976 and the Ph.D. degree in 1984 in Physics from the Universidad Autonoma de Madrid, UAM (Spain). Since 1992 he has been participated, in the implementation of the Computer Science (1992) and Telecommunication (2002) engineering studies at the Polytechnic School (EPS-UAM), and he got his current position as Full Professor at 2010. From his incorporation to the EPS, he has extended his research interests to topics

related with HW/SW applications on embedded systems (microcontrollers, microprocessors, FPGAs and SOC devices) as platforms for wireless sensor nets (WSN) or robotic sensor agents (RSA). In 2003 he co-founded the HCTLab group, (Human Computer Technology Laboratory) and now he is its director. Dr. Garrido has been participated in several R&D projects and has published more than 40 articles in peer-review journals and 60 papers in archived conference proceedings.



**Alberto Sanchez** was born in Madrid, Spain, in 1986. He received the M.Sc. degree in computer science and telecommunication engineering from the Universidad Autonoma de Madrid, Spain, in 2010, where he is currently working toward the Ph.D. degree in the Technology for Electronics and Communications Department. His research interests include digital control of switching mode power supplies and wireless sensor networks with mobile nodes.



**Angel de Castro** (M'08) was born in Madrid, Spain, in 1975. He received the M.Sc. and the Ph.D. degrees in electrical engineering from the Universidad Politecnica de Madrid, Madrid, Spain, in 1999 and 2004, respectively. He has been an Associate Professor in the Universidad Autonoma de Madrid since 2010, and as Assistant Professor from 2006 to 2010. Previously, he was an Assistant Professor in the Universidad Politecnica de Madrid since 2003. His research interests include digital control of switching mode power supplies, field

programmable gate arrays and mobile nodes in wireless sensor networks.