FPGA Implementation of a PWM for a Three-Phase DC–AC Multilevel Active-Clamped Converter

Emili Lupon, Sergio Busquets-Monge, Senior Member, IEEE, and Joan Nicolas-Apruzzese, Student Member, IEEE

Abstract—With the aim to implement a suitable controller for a three-phase dc–ac multilevel active-clamped converter to enable its use in practice, and as a first step toward a full closed-loop converter control implementation into a single field-programmable gate array (FPGA) device, this paper presents the structure and features of an FPGA implementation of an appropriate pulsewidth modulation (PWM) strategy. The selected PWM strategy guarantees dc-link capacitor voltage balance in every switching cycle, and covers both the undermodulation and overmodulation regions. A flexible implementation is conceived, allowing the variation of important operating parameters, such as the modulation index and switching frequency, through a simple user interface. The key aspects to achieve an efficient and robust FPGA implementation are discussed. Experimental results in a four-level converter prototype controlled with an Altera Cyclone III device under different operating conditions match fairly well with the expected results obtained through simulation, thus verifying the accurate performance of the FPGA-based modulator.

Index Terms—Field-programmable gate array (FPGA), multilevel active-clamped (MAC) converter, pulsewidth modulation (PWM).

I. INTRODUCTION

POWER ELECTRONICS converter technology evolves toward multilevel topologies with higher number of switching devices, higher switching frequencies to obtain higher power density and better dynamic performance, and increased control complexity. One of such topologies is the recently proposed multilevel active-clamped (MAC) topology [1], shown in Fig. 1 (a) for a four-level case, which is built upon a single controlled switching semiconductor device with an antiparallel diode. The functional model of the converter leg is the same as for a diode-clamped converter: a single-pole four-throw switch that connects the output leg terminal to one of the four input leg terminals through the application of the corresponding switching state. Fig. 2 presents these switching states for a four-level converter leg. The uncircled switches are off-state devices. The circled switches are on-state devices. The solid-line circled switches connect the output terminal to the desired input terminal and conduct the output terminal current \( (i_o) \). The dotted-line-circled switches do not conduct any significant current and simply clamp the blocking voltage of the off-state devices to the voltage across adjacent input terminals \( (V) \). Compared to a multilevel diode-clamped converter, which presents a lower controlled switch count, the active-clamped topology’s advantages are [1] lower conduction loss, better distribution of switching losses, device blocking voltage always equal to one dc-link capacitor voltage, and increased fault-tolerance capacity. Motor drives, in particular the traction inverter of electric vehicles, is one of the applications where the MAC topology could bring benefits. For this purpose, three legs can be connected to a common dc-link to obtain a four-level three-phase active-clamped converter shown in Fig. 1(b). However, in order to take full advantage, in practice, of the topology benefits, a robust and efficient controller has to be developed at a reasonable cost and with reduced complexity. This is the final goal of the present study.

Power converter control has been progressively migrating from the analog to the digital domain [2]. With the improvement of digital control devices and control techniques, in many applications, the inherent drawbacks of digital control (i.e., finite resolution leading to a reduced accuracy and the need of sampling and processing time leading to delays) have been compensated by the following advantages: 1) increased control complexity; 2) flexibility; 3) repeatability; 4) reliability; 5) versatility; and 6) expandability.

Typical power converter digital controllers are based on microprocessors (μPs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs), or a combination of them. μPs and DSPs contain a central processing unit (CPU, fixed hardware) in charge of sequentially processing a set of instructions. They are, therefore, known to be software-programmable. Microcontrollers and digital signal controllers embed the μP or DSP core, memory, and input/output peripherals (including PWM units) in the same chip. The programming can be performed with high-level algorithmic languages familiar to most designers.

FPGAs [3] basically contain an array of logic elements or cells (each cell containing look-up tables and registers), whose interconnections are decided by the contents of a random-access memory (RAM). The FPGA design, typically using hardware description languages, decides the final circuit structure to achieve a desired functionality, and it is, therefore, said to be hardware-programmable. Besides a large number of configurable input/output blocks, FPGAs also typically contain additional optimized RAM blocks and multipliers to increase the performance of the device. Logic, adders, subtractors, comparators, multipliers, and special functions are easy to implement in these typical FPGAs. Special functions are implemented using look-up tables in RAM blocks. However,

Manuscript received July 04, 2013; revised November 12, 2013 and January 27, 2014; accepted February 19, 2014. Date of publication March 03, 2014; date of current version May 02, 2014. This work was supported by the Ministerio de Ciencia e Innovación, Spain, under Grant CSD2009-00046. Paper no. TII-13-0438.

The authors are with the Department of Electronic Engineering, Technical University of Catalonia, 08028 Barcelona, Spain (e-mail: emili.lupon@upc.edu; sergio.busquets@upc.edu; joan.nicolas@upc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2014.2309483

RAM size increases exponentially with the resolution of the input variables. Multipliers and dividers by powers of two are immediate. Nevertheless, general dividers are more complex. The hardware programmability of FPGAs allows both the concurrent or parallel processing of data and pipelining, which may dramatically reduce the required processing time and increase the number of computations per switching cycle, compared to μPs and DSPs. Additionally, in cases where the implementation of the desired control benefits from the use of a CPU, this can be embedded within the FPGA through the programming of the corresponding circuit. FPGAs have been employed in the control of a number of different topologies and applications [4]–[46], but specially for the control of multilevel converters [23]–[46] and the implementation of PWMs [4]–[9], [24], [26]–[43], [46], predictive controls [10], [44], [45], and fault-tolerant controls [14]–[16].

Due to wide device availability and familiarity with software programming, μP and DSP were the preferred designer’s choice in the first digital control implementations. As control requirements became more stringent from the viewpoint of processing time and number of PWM outputs, many designers opted for a combination of a μP or DSP with an FPGA [13], [23], [41]. The sequential processor is in charge of the high-level control tasks, while the FPGA typically implements the PWM strategy and subsequent switch control signal generation. A critical reason to combine two devices is that current microcontrollers do not contain enough PWM units to control multilevel converters. This is not the case of FPGAs, which contain a large number of possible output pins that can be controlled precisely to generate the switch control signals.

Due to the FPGA’s inherent advantages and expected future improvements, the authors foresee a progressive trend toward a full closed-loop converter control implementation within a single FPGA. In addition to performance improvements, this also saves costs and avoids the cumbersome synchronization and communication between the μP or DSP and the FPGA. Therefore, a full FPGA-based implementation of a three-phase de–ac MAC converter controller is concluded to be the best choice. The authors’ final goal is to implement a closed-loop controller with relevant voltages and currents sampled as close as possible to the beginning of the switching cycle where these values are applied. As a first step, in this paper, the controller is implemented in open loop with the same philosophy: updating the PWM variables as close as possible to the beginning of the switching cycle. The full description of the remaining closed-loop control implementation will be presented in a complementary future publication.

The paper is organized as follows. Section II reviews the PWM strategy and presents a suitable algorithm from which the FPGA
implementation can be discussed. Section III explains the implementation structure and relevant design details to obtain an efficient and robust controller. In Section IV, simulation and experimental results are compared to verify the good performance of the designed modulator under different operating conditions. Section V presents a preliminary discussion of the full closed-loop control implementation. Finally, Section VI presents the conclusions.

II. PWM STRATEGY

The PWM presented in [47] and extended to the overmodulation region in [48] is selected to operate the converter in Fig. 1(b). This PWM, originally defined applying the virtual-vector concept, guarantees the dc-link capacitor voltage balance in every switching cycle and all operating conditions, as long as the phase currents are approximately constant over the switching cycle and their addition is equal to zero. This allows minimizing the size of the dc-link capacitors, leading to a high converter power density. The PWM assumes that the switching frequency \( f_s = 1/t_s \), where \( t_s \) is the switching or sampling period) is much larger than the fundamental frequency. This PWM allows modulation index values \( m \in [0, 1.1027] \), where \( m = v_{\text{dc},1} / v_{\text{dc}} \) and \( v_{\text{dc},1} \) is the peak value of the fundamental component of the line-to-line converter voltage. Therefore, it covers both the undermodulation (UM) and overmodulation (OM) ranges up to six-step operation. The overmodulation region is further divided into two subregions (OMI and OMII) with different types of applied reference vector trajectories [48].

This PWM can be defined with the following algorithm. The desired instantaneous values of the three converter ac-side voltages define a desired rotating reference vector in the \( \alpha - \beta \) plane of the converter space vector diagram (SVD). The desired reference vector-normalized length and angle (with reference to axis \( \alpha \)) are designated as \( m^* \) and \( \theta^* \), respectively. In the UM region, this desired reference vector can be directly synthesized with the available converter space vectors. However, in the OM region, this vector cannot be always synthesized because it sometimes falls outside the SVD outer hexagon. Therefore, an alternative viable reference vector trajectory inside the SVD outer hexagon is defined so as to produce the same fundamental three-phase output voltages, despite the introduction of low-order harmonics. Let us designate the length and angle of the applied reference vector as \( m \) and \( \theta \). Due to the sixfold symmetry of the SVD, this analysis can be performed in the first sextant of the SVD considering the first-sextant-equivalent reference vectors with angles \( \theta^* \) and \( \theta_1 \). The values of \( m \) and \( \theta_1 \) are determined with (1), (2), and (3). These equations have been extracted from [48] [(7) in Appendix A], and rewritten here to facilitate the PWM implementation. Variables \( \text{sex}t_i \) contain the sextant number in which the reference vector is located. The number assigned to each sextant of the SVD is different for each leg ( \( \text{sex}t_a, \text{sex}t_b, \text{sex}t_c \) and \( \text{sex}t_c \)) to explicitly state the three-phase symmetry of the SVD, which will help in simplifying the PWM implementation. Variable \( \text{mode} \) contains two bits to identify the operating mode ( \( \text{mode} = 01 \): UM, \( \text{mode} = 00 \): OMI, \( \text{mode} = 10 \): OMII). Variable \( hbc \) is a special parameter that regulates the capacitor voltage balance control margin in the overmodulation region [48].

\[
\begin{align*}
 m^*_{\text{max}I} &= 3 \cdot \ln(3) / \pi \\
 m^*_{\text{max}II} &= (2 / \pi) \cdot \sqrt{3} \\
 \text{sext}_a &= 1 + \text{floor}(0^* / (\pi / 3)) \\
 \text{sext}_b &= \text{mod}(\text{sext}_a + 4, 6) \\
 \text{sext}_c &= \text{mod}(\text{sext}_a + 2, 6) \\
 \theta^*_1 &= \text{mod}(\theta^*, \pi / 3)
\end{align*}
\]

\[\text{if}(m^* > hbc \cdot m^*_{\text{max}II}) \{\]
\[m = hbc \cdot m^*_{\text{max}II}\]
\[\} \text{end}
\]
\[m = m^* \]
\[\theta_1 = \theta^*_1 \]
\[\text{if}(hbc < m^* \leq hbc \cdot m^*_{\text{max}II})\{
\text{mode} = 00; \\
\theta_{\text{lim}} = (\pi / 6) \cdot (1 - (m^* - hbc) / (hbc \cdot m^*_{\text{max}II} - hbc)) \\
k_{m1} = hbc / \sin(\theta_{\text{lim}} + \pi / 3) \\
k_{m2} = 1 / \sin(\theta^*_1 + \pi / 3) \\
\text{if}((\theta_{\text{lim}} < \theta^*_1) \leq \pi / 3 - \theta_{\text{lim}})\{
\text{mode} = 01; \\
m = hbc \cdot k_{m2} \}
\} \text{elseif}(hbc \cdot m^*_{\text{max}II} < m^* \leq hbc \cdot m^*_{\text{max}II})\{
\text{mode} = 10; \\
\theta_{\text{lim}} = \pi / 6 \cdot (m^* - hbc \cdot m^*_{\text{max}II}) / (hbc \cdot m^*_{\text{max}II} - hbc \cdot m^*_{\text{max}II}) \\
k_{m1} = hbc \cdot 2 / \sqrt{3} \\
k_{m2} = 1 / \sin(\theta^*_1 + \pi / 3) \\
\text{if}(\theta^*_1 < \theta_{\text{lim}})\{
\text{mode} = 01; \\
m = k_{m1} \\
\theta_1 = 0 \}
\} \text{elseif}(\theta^*_1 \geq \pi / 3 - \theta_{\text{lim}})\{
\text{mode} = 01; \\
m = k_{m1} \\
\theta_1 = \pi / 3 \}
\} \text{else}\{
\text{mode} = 01; \\
m = hbc \cdot k_{m2} \}
\} \text{end}
\]

Next, the value of auxiliary variables \( t_a, t_b, t_\gamma, \) and \( t_\delta \) are computed as

\[
\begin{align*}
t_a &= m \cdot k_{t_a} \cdot t_s / 2, \\
k_{t_a} &= \cos(\theta_1 + \pi / 6) \\
t_\beta &= m \cdot k_{t_\beta} \cdot t_s / 2, \\
k_{t_\beta} &= \cos(\theta_1 - \pi / 6) \\
t_\gamma &= t_\beta - t_a \\
t_\delta &= (1 - t_\beta) / 2.
\end{align*}
\]
These auxiliary variables will be used to define the final dwell times of connection to the dc-link terminals $t_1$, $t_2$, $t_3$, and $t_4$, with reference to Fig. 3. But previously, they need to be modified, so that final dwell times fall into a specified value range, as will be commented later.

$$[t_{\text{on}}, t_{\text{off}}, t_{\gamma}, t_{3\text{m}1}, t_{3\text{m}2}] = f(t_\alpha, t_\beta, t_{\gamma}, t_\delta).$$

Finally, the leg $x \in \{a, b, c\}$ dwell times of connection of the leg output terminal to each input terminal are determined with the simple assignments as indicated in Table I.

![Table I: Leg Dwell-Times Computation](image)

Table I: Leg Dwell-Times Computation

<table>
<thead>
<tr>
<th>sext</th>
<th>Leg x dwell times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$t_1 = 0$</td>
</tr>
<tr>
<td>2</td>
<td>$t_1 = t_{\text{on}}$</td>
</tr>
<tr>
<td>3</td>
<td>$t_1 = t_{\text{on}}$</td>
</tr>
<tr>
<td>4</td>
<td>$t_1 = t_{\text{on}}$</td>
</tr>
<tr>
<td>5</td>
<td>$t_1 = t_{\text{on}}$</td>
</tr>
<tr>
<td>6</td>
<td>$t_1 = 0$</td>
</tr>
</tbody>
</table>

Fig. 3. Distribution of the leg output terminal connections over a switching cycle and corresponding switch control signals (high level: switch on; low level: switch off).

![Figure 3](image)

![Figure 4](image)

Fig. 4. Leg “a” duty-ratio pattern over a line cycle ($kbc = 0.98$). (a) $m^* = 0.25$. (b) $m^* = 0.75$. (c) $m^* = 1.0$. (d) $m^* = 1.05$. (the one that turns on first or the one that turns off last, depending on the current polarity).

In this paper, the correction in (5) is made, so that final dwell times verify

$$t_2, t_3 \in [2 \cdot t_\delta, t_\delta/2]$$
$$t_1, t_4 \in [0, [2 \cdot t_\delta, t_\delta/2 - 4 \cdot t_\delta]].$$

It is forced that $t_2, t_3 \geq 2 \cdot t_\delta$ to avoid direct transitions of the leg output terminal connection between nonadjacent input.
whenever they PWM P modules. The applied variable value and is defined in the clk50M, used in most of the relevant processes. Module_, module_, generates the clock signal, and units over a whole switching cycle. parameter values. This reduces to one simple parameter, de the converter and to reset or_.

III. PWM IMPLEMENTATION

The previous PWM strategy is implemented into an Altera Cyclone III EP3C16F484C6 FPGA device with a 50-MHz oscillator, mounted on a DE0 development and education board from Terasic. The device architecture structure and functionality have been fully described in very-high-speed integrated-circuit hardware description language (VHDL).

The simulation has been performed with ModelSim software from Mentor Graphics. The synthesis has been performed with Quartus II software from Altera. Simulation results have been compared with the results obtained from MATLAB-Simulink simulations. The main PWM parameters are set to be user-configurable through 10 slide switches (SdS) to introduce the parameter code and value, and one pushbutton switch (PBS) to validate the data, while the converter is in the off state. Table II lists these parameters, their value range and resolution, from which a 158 Hz–625 kHz range of possible switching frequencies can be easily derived. This allows thoroughly testing the implemented modulator performance under a wide range of operating conditions and power hardware prototypes. Two additional PBS are used to turn-on/off the converter and to reset the system.

A. Basic Design Aspects

Variable values are represented in fixed-point format to reduce hardware needs. The advantages of a floating point representation, such as wider variable value ranges, are not of special interest in the considered application, and it would lead to a more complex hardware implementation. The number of bits of every variable is selected to guarantee the desired accuracy and to optimize the use of FPGA resources. Identification of symmetries, value offset, and value scaling are applied whenever possible to minimize the number of bits required to obtain a specified resolution.

All operations in the algorithm of Section II are easy to implement, except for divisions and transcendental functions. Expression (1) contains simple constants that can be precalculated. Expression (2) highlights the 60-degree symmetry of the algorithm. The divisions and trigonometric functions in (3) require a detailed analysis to obtain the simplest implementation. Fig. 5 deploys the pattern for \( k_{m2} \), \( k_{t_0} \), and \( k_{t_3} \), from which 30-degree symmetries, proper offset values, and proper scaling values can be identified.

A basic time unit equal to \( t_d \) is defined. A clock signal with period \( t_d \) is generated to control most of the FPGA processing. All time variables are expressed as an integer number of this basic unit or power-of-two multiples of this unit. 16-Bit counters are necessary to count \( t_d \) units over a whole switching cycle.

A basic angle unit (au) is defined, so that each switching cycle corresponds to an integer number of au for every combination of \( f_s \), \( t_d \), \( t_b \), and \( t_c \) parameter values. This reduces to one simple addition modulus a sextant, the calculation of the next switching cycle angle. On the other hand, to maximize the resolution for a given number of bits, one sextant of a line cycle must correspond to a number of au slightly lower than a power of two. It is finally set that one sextant contains 1 875 000 au. Thus, all angles lower than 60 degrees can be represented with 21 bits.

B. Controller Structure and Module Description

The designed controller structure, applying concurrent processing as much as possible, is depicted in Fig. 6. It is composed of a set of blocks or modules, classified as auxiliary \((A_1 - A_6)\) and main \((M_1 - M_11)\) modules. The applied variable value offset and scaling to optimize the implementation is not explicitly shown in the diagram.

From the 50-MHz system clock signal clk50M, having a period \( t_{clk} = 20 \text{ ns} \), module \( A_1 \) generates the clock signal clktd, with period \( t_d \), used in most of the relevant processes. Module \( A_2 \) applies simple logic to generate a synchronous system reset signal rst from the state of the corresponding PBS. Module \( A_3 \) samples at low frequency the value of the signals generated by the other two PBS, and \( A_4 \) generates a pulse of length \( t_d \) in the output signal whenever a rising edge occurs in the corresponding input signal. Finally, modules \( A_5 \) and \( A_6 \) are in charge of generating the required signals to show in three seven-segment displays the code and stored value [more significant hexadecimal digit (MSD) and less significant hexadecimal digit (LSD)] of a given system parameter, determined by the position of the SdS.

Module \( M_1 \) updates the registered value of a given system parameter, defined by the 10 SdS, whenever the data validation PBS is pressed. The values are expressed and stored in the units indicated within brackets in the block output signals. Parameter
limits and coherence are checked. Before proceeding to store a new value, it is first verified that

\[ f_s [5 \text{ Hz}] \cdot t_s [4t_b] \cdot t_b [2t_d] \cdot t_d [t_{ck}] \leq 50 000 \quad (7) \]

to guarantee a minimum of 25 switching cycles per line cycle. It is also verified that \( n^* \leq hbc \cdot n^* \text{add} \). On the other hand, two additional parameters are calculated as

\[ t_s [8t_d] = t_s [4t_b] \cdot t_b [2t_d] \]

\[ \Delta \theta^* [\text{au}] = 9 \cdot f_s [5 \text{ Hz}] \cdot t_s [4t_b] \cdot t_b [2t_d] \cdot t_d [t_{ck}], \quad (8) \]

where \( \Delta \theta^* \) is the increment of \( \theta^* \) corresponding to one switching cycle.

Module \( M_3 \) updates the value of \( \theta_1 \) according to \( \Delta \theta^* \) as close as possible to the end of the switching cycle, and generates a set of signals (\( \text{cntts}_s.0..\text{cntts}_s.4 \)) to control the timing of the PWM variable value updates, as indicated in Fig. 7.

Modules \( M_3 \) and \( M_4 \) are read-only memories (ROM) plus a little logic that determine the operating mode (UM, OM1, or OMII), the characteristic angle \( \theta_{\text{in}} \) in the OM (crossover or holding angle), and parameters \( k_{m1} \) and \( k_{m2} \) defining the applied instantaneous modulation index in the OM. \( M_3 \) outputs are constant for constant \( m^* \) and \( hbc \), while the \( M_4 \) output changes every switching cycle. \( M_3 \) requires a capacity of 512 × 32 bits, and it is implemented with two 9-Kb memory blocks. \( M_4 \) requires a capacity of 4096 × 9 bits, and it is implemented with four 9-Kb memory blocks.

---

**Fig. 6.** FPGA-controller global structure.

**Fig. 7.** Timing of events before the beginning of a new switching cycle.

**Fig. 8.** Leg switch-control-signal generator module structure.
Based on the values of \( mode \) and \( \theta_{lim} \), \( M_5 \) selects the proper values of \( \phi_1 \) among the three possible. After this, and with the aid of multipliers and ROM modules \( M_6 \) and \( M_7 \), the value of \( t_\phi \) and \( t_\tau \), expressed in \( t_d \) units with 15 bits, are obtained. Through simple algebraic operations, \( M_8 \) computes \( t_\phi \) and \( t_\tau \). Then, \( M_9 \) modifies this set of time values so as to verify (6). The criteria followed to perform this modification are to maintain, whenever possible, the produced switching-cycle-averaged leg output-terminal voltage \( V_{\text{avg}} \), and to reduce the number of switching transitions. A robust modification requires a detailed analysis of the multiple possible cases and generates two possible modified values of \( t_\phi \): for sextants 2 and 5 in Table I and for the rest.

\( M_9 \) requires a capacity of 8192 × 12 bits and is implemented with twelve 9-Kb memory blocks. Considering offsets and symmetries (see Fig. 5), \( M_7 \) requires a capacity of 4096 × 9 bits, and it is implemented with four 9-Kb memory blocks.

From the above-mentioned information, \( M_{10} \) is in charge of generating the 12 switch control signals for one leg. Fig. 8 presents the internal block diagram of \( M_{10} \). Module \( M_{10,A} \) is in charge of updating and storing the \( t_1, t_2, t_3, \) and \( t_4 \) values for the next switching cycle (refer to Fig. 3) at the time defined in Fig. 7, and according to Table I. Two more bits are generated \( t_1,0 \) and \( t_4,0 \), which indicate whether \( t_1 \) and \( t_4 \) are equal to zero, respectively. These values are updated as indicated in Fig. 7. All this information is fed to \( M_{10,B} \), which contains a state machine in charge of determining the leg switching state, presented in Fig. 9.

The name of the 27 states in Fig. 9 is defined with reference to the switching state \( s_{21} \) of connection of the leg output terminal to the \( j \)th input terminal. The state is further identified by the output values of the 16 independent switch control signals \( s_{n1} − (s_{p11} = s_{p12}) − s_{p13} − s_{n21} − s_{n22} − s_{p21} − s_{p22} − (s_{n31} = s_{n32}) − s_{n33} − s_{p31} \) (in red) and two additional bits (black) to differentiate states with the same switch control signal values. The transition from one state to the next is in general governed by the time elapsed according to Fig. 3. It is interesting to note that the transition from one switching cycle to the next must be handled properly in accordance to the \( t_4,0 \) value of the old and new switching cycles (there are four different transitions). Similarly, in the middle of the switching cycle, the transition from \( ss2 \) to \( ss1 \) and vice versa are skipped when \( t_1,0 = 1 \). The module output \( \text{switching.state} \) defined by the 10 red bits is then fed to \( M_{10,C} \), which generates the 12 switch control signals and masks the values of some switch control signals during the global system turn-\( \text{ON} \) and turn-\( \text{OFF} \) transitions, to avoid a possible switch overvoltage during these transients [49].

The global \( \text{ON} - \text{OFF} \) state of the system is controlled by the state machine of \( M_{11} \), as presented in Fig. 10. The red bits in each state define the \( \text{system.state} \) output of \( M_{11} \). They represent, from left to right and with reference to Fig. 1(a), the masking bit for the switches in pole 3 (this bit is also the \( t_4 \) counter enable in \( M_{10,B} \)), the masking bit for pole 2, the masking bit for pole 1, and the...
counter cntts enable in $M_2$. In the transition from OFF to ON states, first pole 1 is enabled, then pole 2 is enabled, and finally pole 3 is enabled. This sequence of events is reversed in the transition from ON to OFF.

C. Consumed FPGA Resources

A special effort has been made in the design to save FPGA resources. Table III indicates the resources used to synthesize the previous controller, as reported by Quartus II software. These resources represent roughly around 20% of the total FPGA resources, leaving enough space to implement a full closed-loop converter controller, including the closed-loop control of the dc-link capacitor voltages and three-phase currents. A reduced number of resources will be necessary if a fixed value of Table II parameters is assumed, which is typically the case in practical applications.

IV. Simulation and Experimental Results

The previous FPGA-based control has been tested with a three-phase dc–ac converter configuration feeding a three-phase series R–L load, as shown in Fig. 1(b). The converter prototype with the DE0 board on top, containing the Altera Cyclone III EP3C16F484C6 FPGA, is presented in Fig. 11.

Fig. 12 presents the experimental switch control signals and output voltage of one leg, which follows the pattern described in Fig. 3. Note that the analog $v_C$ voltage waveform is delayed with reference to the digital switch control signals due to gate driver and measurement delays. Since the converter is nonideal and it is operating in open-loop, there is some unbalance among the capacitor voltages that could be easily corrected with the introduction of a closed-loop control [50]. The operating principle of such closed-loop control can be summarized as follows. From the sensed dc-link capacitor voltages $v_{C1}$, $v_{C2}$, and $v_{C3}$, the value of the capacitor voltage unbalance is determined. This unbalance is then corrected through a slight modification of the open-loop leg duty-ratio pattern as shown in Fig. 4, which produces a switching-cycle average current through the inner dc-link points [$i_2$ and $i_3$ in Fig. 1(b)] different from zero, as needed to correct the unbalance.

Figs. 13 and 14 present simulation and experimental results at different modulation indexes, covering the UM and OM ranges. MATLAB-Simulink is used to perform the simulations. The obtained experimental results fairly match those from simulation, verifying the accurate performance of the programmed controller.

V. Closed-Loop Control Implementation

This section presents a preliminary description of the future work on a full closed-loop control implementation. Fig. 15 presents the intended control structure for a three-phase
permanent magnet synchronous motor drive. Variables $\omega$ and $\phi$ correspond to the measured rotor angular speed and position, respectively, provided in digital form by a resolver and associated circuitry. Variables $i_d$ and $i_q$ are the direct and quadrature components of the three-phase currents. Variables $d^*_q$ and $q^*_d$ are the direct and quadrature components of the normalized reference vector required by the modulator. Command values are designated with an asterisk superscript. The control inputs are
VI. CONCLUSION

A full FPGA-based controller implementation is concluded to be the best choice to enable the practical use of MAC converters and take full advantage of their potential benefits at a reasonable cost. Conventional PWM strategies lead to the collapse of some dc-link capacitor voltages under a wide range of operating conditions for the case of converters with a number of levels higher than three. A singular and complete PWM strategy capable of guaranteeing the dc-link capacitor voltage balance in every switching cycle for all operating conditions has been selected to operate a four-level three-phase dc–ac active-clamped converter featuring 36 switching devices with independent gate control signals. This PWM strategy has been efficiently and robustly implemented into a medium performance FPGA, consuming only a limited amount of resources. The remaining FPGA resources can be employed to implement typical closed-loop controls for different applications with a high dynamic performance, and also dynamically reconfiguring the control in runtime, which can be useful, for instance, in implementing fault-tolerant controls [51].

REFERENCES


Emili Lupon received the M.S. and Ph.D. degrees in electrical engineering from the Universitat Politecnica de Catalunya (UPC), Barcelona, Spain, in 1975 and 1983, respectively. Since 1986, he has been an Associate Professor with the Department of Electronic Engineering, UPC. His research interests include design, test, and verification of digital and mixed-signal circuits; digital design based on FPGAs and/or microprocessors; and error correction codes.

Sergio Busquets-Monge (S’99–M’06–SM’11) was born in Barcelona, Spain. He received the M.S. degree in electrical engineering from the Technical University of Catalonia, Barcelona, Spain, in 1999; the M.S. degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2001; and the Ph.D. degree in electrical engineering from the Technical University of Catalonia, in 2006. From 2001 to 2002, he was with Crown Audio, Inc., Elkhart, IN, USA. He is currently an Associate Professor with the Department of Electronic Engineering, Technical University of Catalonia. His research interests include multilevel conversion and converter integration.

Joan Nicolas-Apruzzese (S’11) was born in Maracaibo, Venezuela. He received the M.S. and Ph.D. degrees in electrical engineering from the Technical University of Catalonia, Barcelona, Spain, in 2008 and 2013, respectively. Since 2008, he has been a Researcher in the Power Electronics Research Centre, Technical University of Catalonia. His research interests include power multilevel converters applied to electric vehicles, and photovoltaic- and wind-energy systems.