Abstract—The multilevel multiphase technology combines the benefits of multilevel converters and multiphase machines. Recently, a new multilevel multiphase space vector pulsewidth modulation algorithm that is valid for any number of levels and phases was developed. In this paper, a generic digital VHDL module for such an algorithm is presented. This module is parameterizable in relation to the number of levels, the number of phases, and the number of bits of fractional part of the reference vector. It is also technology independent, reusable, and modular. This circuit has been tested by implementation in a field-programmable gate array, with the goal of a balance of speed and area optimization. It was tested by using a five-level five-phase inverter feeding an induction motor.

Index Terms—Field-programmable gate array (FPGA), multilevel converter, multiphase machine, space vector pulsewidth modulation (SVPWM), VHDL.

I. INTRODUCTION

MULTIPHASE electrical machines have many advantages when compared with their three-phase counterparts, such as improved reliability, increased fault tolerance, higher efficiency, and lower torque pulsations [1]. They have been found to be ideally suited for direct drives in marine propulsion applications [2]–[4] and for drive systems in safety-critical applications, such as the more-electric aircraft [5]–[9]. Other recent applications include electric vehicle propulsion [10]–[13] and locomotive traction [14], [15].

Multilevel converters have been extensively studied because they can manage high output voltages with voltage-limited devices, and they combine high efficiency with low harmonic distortion [16]. Recent industrial applications of multilevel inverters include induction high-power motor drives [17]–[19], hybrid electric vehicles [20], regenerative rectifiers [21], power conversion for renewable energy sources [22], [23], static synchronous compensators [24], [25], and fault-tolerant systems [26].

Multilevel multiphase variable-speed motor drives inherit the benefits of both technologies, with the drawback of an increased complexity of the modulation process because of the large number of devices that must be controlled [27]. Carrier-based pulsewidth modulation (PWM) is a simple modulation technique that individually handles each leg of the converter. Therefore, its application multiphase converter is rather straightforward [28], [29]. The space vector PWM (SVPWM) simultaneously deals with all phases of the converter, and consequently, the extension of the three-phase SVPWM techniques to multiphase converters is more involved. In the last years, many multiphase SVPWM algorithms have been developed [30]–[42]. Most of them are devoted to two-level converters, and they make use of the multiple dq space concept [30]–[38]. The SVPWM techniques in [30] and [31] for five-phase converters, in [32] for seven-phase converters, and in [33] and [34] for nine-phase converters can only deal with the sinusoidal output voltage. The SVPWM algorithms presented in [35] and [36] allow us to add small amplitude harmonics to the output voltage. The extension of the sinusoidal output SVPWM algorithms to the multifrequency output was addressed in [37]. In [38], a general modulation algorithm for converters with an odd number of phases is presented. The algorithms in [39]–[41] can also deal with the multifrequency output, but they make use of the multidimensional approach introduced in [43]. The algorithm in [39] can be applied to two-level converters with any number of phases. Nevertheless, it is very time consuming, and its implementation using lookup tables (LUTs) requires a big amount of memory [44]. The SVPWM problem for the multilevel converters is addressed in [40] and [41], where two algorithms for converters with any number of levels and phases are presented. Both algorithms can handle the thousands of space vectors available in multilevel multiphase [45] converters with low computation and low memory requirements, which makes them very suitable for online algorithm implementation in a field-programmable gate array (FPGA).

FPGAs stand out among the integrated circuits that allow the implementation of specific applications because they combine high performance, relatively low cost, enough flexibility, and short development process due to field programmability. Moreover, most of the FPGAs in the market are reprogrammable, which allows updating the implemented circuit with bug fixes and new enhancements [46], [47]. In the field of variable-speed motor drives, many experiences with FPGAs have been reported. The control for different permanent-magnet synchronous motors is implemented in an FPGA in [48]–[50]. In [51], an FPGA is used to implement a field-oriented controller based on a neural network for an induction motor drive. In [52], a direct-torque control for a sensorless induction motor...
is implemented in an FPGA. A review of the FPGA-based current controllers for ac machine drives is presented in [53]. In [54], a digital PWM controller scheme for a brushless dc motor drive implemented in an FPGA is presented.

The FPGA implementation of the SVPWM algorithms started with the two-level three-phase implementation realized in [55]. In [56]–[59], different multilevel SVPWM algorithms for three-phase converters were implemented in an FPGA. The two-level multistage SVPWM algorithm presented in [60] and the multilevel multistage SVPWM algorithms presented in [40] and [41] are verified using an FPGA. In [40] and [41], the circuits that were used to verify the modulation algorithms were specifically designed for a five-level five-phase converter, proving that they can be implemented online in a low-cost device.

This paper presents a generic VHDL module for the multilevel multiphase SVPWM algorithm in [40]. The main contribution of this paper with respect to [40] is to detail a completely new and optimized VHDL implementation of the SVPWM technique. This implementation is parameterizable in relation to the number of levels, the number of phases, and the number of bits of the fractional part of the reference vector. The influence of the value of those parameters in the used FPGA resources, in the converter switching frequency, and in the resolution of the algorithm is discussed in this paper. A modular design, in which each step of the algorithm is carried out by an independent submodule, was developed. The SVPWM algorithm steps were analyzed to determine which submodules can run in parallel and which submodules must run sequentially to obtain a good area–speed tradeoff. All submodules were described using only the standard VHDL to obtain a technology-independent circuit. Since the SVPWM algorithm with switching state redundancy in [41] is based on [40], some of the circuits described in this paper can also be reused to implement such algorithm.

This paper is organized as follows. The basics and the steps of the multilevel multiphase SVPWM algorithm in [40] are summarized in Section II. Section III describes the implementation in VHDL of the modulation algorithm. In Section IV, the implementation is verified by the simulation, and the circuit is tested in the laboratory by means of a five-level five-phase inverter. Section V presents the conclusion of this paper.

II. MULTILEVEL MULTIPHASE SVPWM ALGORITHM

In multiphase converters, the SVPWM is a multidimensional problem where the vector selection can be directly carried out in a multidimensional space [43]. In [40], the modulation problem of a P-phase converter is formulated in a P-dimensional space, and it is solved for multilevel topologies in which the output level of every phase is an integer multiple of a fixed voltage step $V_{dc}$. Flying capacitor, diode-clamped, cascaded full-bridge, and hybrid converters are included in such topologies [61]. Since the switching states of any power converter topology stay at discrete states, the SVPWM technique in [40] is used to synthesize a reference voltage vector $\mathbf{v}_r = [v^1_r, v^2_r, \ldots, v^P_r]^T$ by means of a sequence of space vectors during each modulation cycle. Each space vector $\mathbf{v}_{sj} = [v^1_{sj}, v^2_{sj}, \ldots, v^P_{sj}]^T$ must be applied during an interval $t_j$ in accordance with the following modulation law:

$$\mathbf{v}_r = \sum_{j=1}^{P+1} \mathbf{v}_{sj} t_j \sum_{j=1}^{P+1} t_j = 1. \quad (1)$$

The aforementioned modulation problem is solved in [40] by means of a mathematical algorithm, with the block diagram shown in Fig. 1. This algorithm, which is based on a displacement plus a two-level multiphase SVPWM algorithm, has the following steps.

1) Decompose the normalized reference $\mathbf{v}_r$ into the sum of its integer part $\mathbf{v}_i = [v^1_i, v^2_i, \ldots, v^P_i]^T$ and its fractional part $\mathbf{v}_f = [v^1_f, v^2_f, \ldots, v^P_f]^T$ by means of

$$\mathbf{v}_i = \text{int}(\mathbf{v}_r) \quad (2)$$

$$\mathbf{v}_f = \mathbf{v}_r - \mathbf{v}_i. \quad (3)$$

2) Calculate the permutation matrix $\mathbf{P}$ that sorts the vector $\mathbf{v}_f$ in descending order in accordance with

$$\mathbf{P} \begin{bmatrix} 1 \\ \mathbf{v}_f \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{\hat{v}}_f \end{bmatrix} \quad (4)$$

where $\mathbf{\hat{v}}_f = [\hat{v}^1_f, \hat{v}^2_f, \ldots, \hat{v}^P_f]^T$ is the sorted vector in which

$$1 > \hat{v}^1_f \geq \cdots \geq \hat{v}^{k-1}_f \geq \hat{v}^k_f \geq \cdots \geq \hat{v}^P_f \geq 0. \quad (5)$$

\footnote{All of the details of the mathematical justification of the modulation algorithm can be found in [40].}
III. PARAMETERIZABLE SVPWM MODULATOR

A. Design Considerations

In order to make a generic parameterizable circuit with a modular design, balanced in terms of speed and area and described with independence of the implementation technology, the following decisions were adopted.

1) Parameterizable Circuit: All of the components designed for the modulation circuit are parameterizable through the use of a VHDL package and the use of VHDL generics, which provides great flexibility. The user only needs to edit such package to select a new set of values for the parameters and to resynthesize the modulation circuit to obtain a new implementation because the values of the additional modulator parameters are automatically calculated.

2) Modular Design: Every component of the SVPWM circuit is designed to be a generic and parameterizable module, which makes them reusable. For example, there is only one VHDL file which describes a counter. This module is parameterized in a different way to make each distinct counter needed as part of the modulation circuit. The internal block diagram (structure) of the modulation algorithm described in [40] has been followed. Each of the six steps, which have been described in Section II, is carried out by an independent circuit.

3) Speed Optimization: The calculation time, which must be less than the converter switching period, depends on the actual values of the modulator parameters. The whole modulator circuit is synchronous, i.e., every component uses the same global clock signal. Therefore, the calculation time can be calculated by taking into account the clock frequency and the number of clock cycles needed by the modulation algorithm. In order to get an implementation tradeoff between speed and area, some operations are simultaneously executed, and some of them are sequentially executed.

4) Area Optimization: The partial results of the modulation algorithm are stored in distributed random-access memory (RAM). Such memory is based on the FPGA LUTs, which are actually tiny static RAMs (SRAMs). By using RAMs instead of registers, the required logical resources are drastically reduced. Nevertheless, it increases the calculation time of the circuit because of the sequential access to the data in the RAM.

5) Technology-Independent Description: All circuits are described entirely in VHDL, with no instantiations to any specific manufacturer components. Such a description allows the implementation of the modulation circuit in any FPGA, application-specific integrated circuit (ASIC), or equivalent digital integrated circuit.

B. Development of the SVPWM Circuit

The parameters of the modulation circuit are the number of phases of system \( P \), the number of levels of converter \( N \), and the number of bits of the fractional part of the reference vector \( Q \). All of these parameters are defined by the user in the part of the `svpwm_constants.vhd` VHDL package shown in Table I. All of the inner parameters of the circuit, such as the number of bits that is used to represent the levels of converter \( I \), the number of bits \( K \) that is used to represent the number of phases, and the number of bits that is used to
Third step is realized by the circuit, the second step is realized by the described in Section II. The first step is realized by the modulation circuit, which correspond with the algorithm steps in this package.

\[
V_d \quad \text{realized by the } t \quad \text{in Section II.}
\]

\[
V_r \quad \text{is realized by the } V_{d}\text{ extr}. \quad \text{Corresp. to steps 1 and 2.}
\]

\[
V_f \quad \text{calc} \quad \text{with the internal circuits properly connected. It is composed}
\]

\[
\text{of the following three main circuits: } V_{r}\text{ dec, } V_{d}\text{ extr, and } V_{s}\text{ calc.}
\]

Fig. 2. Flow chart of the multilevel multiphase SVPWM module.

represent the number of phases plus one \( J \), are also calculated in this package.

The flow chart in Fig. 2 shows the different operations of the modulation circuit, which correspond with the algorithm steps described in Section II. The first step is realized by the \( V_{r}\text{ dec} \) circuit, the second step is realized by the \( V_{f}\text{ sort} \) circuit, the third step is realized by the \( D\text{ calc} \) circuit, the fourth step is realized by the \( V_{d}\text{ extr} \) circuit, the fifth step is realized by the \( t\text{ calc} \) circuit, and the sixth step is realized by the \( V_{s}\text{ calc} \) circuit. Since steps 2–5 correspond to an inner two-level multiphase SVPWM algorithm, the circuits corresponding to those steps have been gathered in the \( 2P\text{_SVPWM} \) block. This block is parameterizable in relation to the number of phases \( P \) and the number of bits of the fractional part of the reference vector \( Q \), and it can be used as a stand-alone with the two-level multiphase converters. The whole multilevel multiphase SVPWM circuit is the \( NP\text{_SVPWM} \) block that groups all circuits.

The flow chart in Fig. 2 shows that step 6 requires the data previously calculated in step 4, step 4 requires the data from step 3, and so on until step 1. Thus, circuits \( V_{r}\text{ dec}, V_{f}\text{ sort, } D\text{ calc, } V_{d}\text{ extr, and } V_{s}\text{ calc} \) must sequentially run, as shown in Fig. 3. Step 5 only requires the data calculated in step 2. Consequently, circuit \( t\text{ calc} \) can start after \( V_{f}\text{ sort} \) and can run in parallel with circuits \( D\text{ calc, } V_{d}\text{ extr, and } V_{s}\text{ calc} \).

Fig. 4 shows the hierarchical view of the \( NP\text{_SVPWM} \) module, with the internal circuits properly connected. It is composed of the following three main circuits: \( V_{r}\text{ dec, } 2P\text{_SVPWM, and } V_{s}\text{ calc} \). The operation of these circuits is sequential. The \( 2P\text{_SVPWM} \) circuit uses the values calculated by the \( V_{r}\text{ dec} \) circuit. Thus, it starts its operation once the \( V_{r}\text{ dec} \) circuit ends its operations. Similarly, the \( V_{s}\text{ calc} \) circuit uses the values calculated by the \( 2P\text{_SVPWM} \) circuit, so it starts its operation once \( 2P\text{_SVPWM} \) has ended its operation. To concatenate the sequential operation of the circuits, each one has two synchronizing signals (Init and End). Usually, the End signal of one circuit is connected to the Init signal of the following circuit. It means that the internal circuits are inactive most of the time. This consideration, joined to the use of synchronous circuits that avoid the propagation of unwanted glitches, reduces the FPGA dynamic power consumption due to the SVPWM process. The \( 2P\text{_SVPWM} \) block includes the following circuits: \( V_{f}\text{ sort, } D\text{ calc, } V_{d}\text{ extr, and } t\text{ calc} \). The \( t\text{ calc} \) circuit runs in parallel with the \( D\text{ calc} \) and \( V_{d}\text{ extr} \) circuits. Therefore, the End signal of the \( V_{f}\text{ sort} \) circuit is connected to the Init signals of the \( D\text{ calc} \) and \( t\text{ calc} \) circuits. The End signal of the \( 2P\text{_SVPWM} \) circuit is connected only when \( V_{d}\text{ extr} \) ends its operations, because this circuit always ends the calculations after \( t\text{ calc} \), as shown in Table III.

All of the internal circuits in Fig. 4 have the basic structure shown in Fig. 5. They are controlled by a synchronous finite-state machine (FSM) that receives the Init order, controls the processing unit, and generates the End signal. The processing unit takes the input data \( \text{In}(A_{in}) \) by means of an addressing signal \( A_{in} \), and it makes the calculations and stores the results in a dual-port RAM \( \text{Out}(A_{out}) \). The first addressing port of the RAM is a write-only port that is internally used to store the calculation results, and the second addressing port \( A_{out} \) is a read-only port that can be accessed by external circuits. Dual-port RAMs are usual components physically included in most of the modern FPGAs, and of course, they can be included in any digital ASIC. Nevertheless, the SVPWM circuit is always implementable due to the VHDL code, which has been designed to be independent from any manufacturer libraries. However, if dual-port RAMs are not physically available, the implementation would use more hardware resources.

C. Description of the Internal Circuits

1) \( V_{r}\text{ dec} \): This circuit decomposes the normalized reference into its integer part \( v_i \) and its fractional part \( v_f \) according to (2) and (3). Therefore, it splits the \( I + Q \) bits of each component of the reference vector in two parts: the first \( I \) bits
Fig. 4. Hierarchical view of the multilevel multiphase SVPWM circuit. (a) NP\textsubscript{SVPWM} circuit. (b) 2P\textsubscript{SVPWM} circuit.

Fig. 5. Basic structure of the internal circuits.

corresponding to the integer part and the last $Q$ bits corresponding to the fractional part. The circuit reads the components of the reference vector $Vr(kr)$ by means of the addressing signal $kr$. The resulting vectors $v_i$ and $v_f$ are separately stored in the dual-port RAMs $V_i(ki)$ and $V_f(kf)$ that can be addressed by means of $ki$ and $kf$, respectively. Each memory position stores one vector component. The $Vr$\textsubscript{dec} circuit is composed of one counter, two dual-port RAMs, and one FSM.

2) $Vf$\textsubscript{sort}: This circuit sorts the components of the fractional part of the reference vector $v_f$ in descending order, according to (5), using the bubble sort algorithm. This sorting algorithm starts at the beginning of the vector. It compares the first two components (phases), and if the first one is lower than the second one, it swaps them. It continues to do this for each pair of adjacent components to the end of the vector. It then starts again with the first two components, repeating until no
swaps have occurred on the last pass. The sorted vector is stored in the RAM \( V_0(k_0) \), and the final position of each component at the end of the sort process, with respect to its initial position, is stored in the RAM \( \text{id}(k_i \text{d}) \). Both RAMs have two read ports and one write port. The read ports are used by the processing unit to access the dual-port RAM and to easily swap the vector components with the help of an auxiliary register. The \( \text{Vf}\_\text{sort} \) circuit is composed of two counters, five multiplexers, two registers, one magnitude comparator, two dual-port RAMs, and one FSM.

3) \( \text{D}\_\text{calc} \): This circuit calculates matrix \( D \) in accordance with (7). Matrix \( P^T \) reversely rearranges the rows of matrix \( D \) with respect to the sort process of the fractional part. For example, if the third component of vector \( \nu_f \) finally takes the fifth position, then the fifth row of matrix \( D \) will finally have the value of the third row of the upper triangular matrix \( D \). Consequently, this circuit rearranges the rows of matrix \( D \) with the information stored in \( \text{id}(k_i \text{d}) \). Each row of the resulting matrix \( D \) is stored in a memory position of the dual-port RAM \( D(\text{row}) \). The first row of matrices \( D \) and \( D \), which is useful in algorithm demonstration, was not taken into account in the implementation because this row is always constant, and it is not needed in extracting the displaced switching vectors \( \nu_d \). The \( \text{D}\_\text{calc} \) circuit is composed of one counter, one register, one dual-port RAM, and one FSM.

4) \( \text{Vd}\_\text{extr} \): This circuit extracts the displaced switching vector sequence \( \{\nu_{dj}\} \) from the coefficient matrix \( D \) according to (8). The values of the components \( \nu_{dj}^k \) of each vector are stored bit by bit in the 2-D RAM \( \text{Vd}(k_\text{d}, j_\text{d}) \). This simplifies the later calculations of the switching vector sequence \( \{\nu_{sj}\} \). A parameterizable 2-D dual-port RAM, with row and column addresses, was defined in VHDL. The parameters are the row size, the column size, and the number of bits of each memory position. Physically, the memory is implemented as a conventional RAM in which the actual address is obtained by combining both the row and column addresses by the equation address = row_address × 2\(^{\text{column_size}}\) + column_address. The column address \( j_\text{d} \) is used to access the \( j \)th vector of the displaced vector sequence \( \{\nu_{sj}\} \), and the row address \( k_\text{d} \) is used to access the \( k \)th component \( \nu_{dj}^k \) of that vector. The \( \text{Vd}\_\text{extr} \) circuit is composed of two counters, one register, one dual-port RAM, and one FSM.

5) \( \text{t}\_\text{calc} \): This circuit calculates the dwell times \( t_j \) from the sorted vector \( \nu_f \) in accordance with (9). The result is stored in the dual-port RAM \( t(j) \). The \( \text{t}\_\text{calc} \) circuit is composed of one counter, two registers that are used to store the minuend and the subtrahend, one subtractor, one dual-port RAM, and one FSM. The minuend register is initialized to a value of one to ease the calculation of the first switching time \( t_1 = 1 - \nu_f^1 \).

6) \( \text{Vs}\_\text{calc} \): This circuit calculates the final switching vector sequence \( \{\nu_{sj}\} \) in accordance with (10). The results are stored component by component in the 2-D dual-port RAM \( \text{Vs}(k_\text{r}, j_\text{s}) \). Therefore, to access the component \( \nu_{sj}^k \) of the final switching sequence, it is necessary to provide the row address \( k_\text{r} \) and column address \( j_\text{s} \), as with the \( \{\nu_{dj}\} \) sequence in the \( \text{Vd}\_\text{extr} \) circuit. The \( \text{Vs}\_\text{calc} \) circuit is composed of two counters, one adder, one dual-port RAM, and one FSM.

All signals, except \( \text{Vr}(k_r) \), \( t(j) \), and \( \text{Vf}(k_f) \), are integer numbers represented with the number of bits shown in Fig. 4. These three signals are real numbers that are represented in the fixed-point format. Signal \( \text{Vr}(k_r) \) is represented in two’s complement using \( I \) bits for the integer part and \( Q \) bits for the fractional part. Signal \( \text{Vf}(k_f) \) is a positive real number in the range \([0, 1)\), with all \( Q \) bits representing the fractional part. Signal \( t(j) \) is a real number in the range \([0, 1)\), using one bit to represent the integer part and \( Q \) bits to represent the fractional part.

The size of the main basic components used by the internal circuits is detailed in Table II. It depends basically on the number of phases, directly through parameter \( P \), and indirectly through variables \( J \) and \( K \). The number of bits of the reference vector \( Q \) and the number of levels through variable \( I \) have a lower influence in the circuit size.

Table III shows the number of clock cycles \( n_{cik} \) used by the internal circuits. Related to this table, it is necessary to highlight the following issues.

1) The number of clock cycles \( n_2 \) used by the \( \text{Vf}\_\text{sort} \) circuit is not constant. It depends on the initial sorting of the numbers.
2) The value \( 2i \) in every circuit is due to two initial clock cycles, which simultaneously elapse for all circuits. Therefore, both cycles do not accumulate to the total delay of the whole circuit.

<table>
<thead>
<tr>
<th>Step</th>
<th>Circuit</th>
<th>Counters ( \text{bits} )</th>
<th>Registers ( \text{bits} )</th>
<th>Multiplexers ( \text{channels/\text{bits}} )</th>
<th>Adders ( \text{bits} )</th>
<th>Comparators ( \text{bits} )</th>
<th>dual port RAMs ( \text{size} \times \text{bits} )</th>
<th>FSMs ( \text{states} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \text{Vr}_\text{dec} )</td>
<td>( 1 \times (K + 1) )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>( 1 \times (2^K \times I) )</td>
<td>( 1 \times (7) )</td>
</tr>
<tr>
<td>2</td>
<td>( \text{Vf}_\text{sort} )</td>
<td>( 2 \times (K + 1) )</td>
<td>( 1 \times (K) )</td>
<td>( 3 \times (2/K) )</td>
<td>( 1 \times (Q) )</td>
<td>( 1 \times (2^K \times K) )</td>
<td>( 1 \times (6) )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( \text{D}_\text{calc} )</td>
<td>( 1 \times (K) )</td>
<td>( 1 \times (J) )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>( 1 \times (2^K \times (P + 1)) )</td>
<td>( 1 \times (7) )</td>
</tr>
<tr>
<td>4</td>
<td>( \text{Vd}_\text{extr} )</td>
<td>( 1 \times (K) )</td>
<td>( 1 \times (P + 1) )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>( 1 \times ((2^K \times 2^f) \times 1) )</td>
<td>( 1 \times (10) )</td>
</tr>
<tr>
<td>5</td>
<td>( \text{t}_\text{calc} )</td>
<td>( 1 \times (J) )</td>
<td>( 2 \times (Q + 1) )</td>
<td>-</td>
<td>( 1 \times (Q) )</td>
<td>-</td>
<td>( 1 \times (2^f \times (Q + 1)) )</td>
<td>( 1 \times (14) )</td>
</tr>
<tr>
<td>6</td>
<td>( \text{Vs}_\text{calc} )</td>
<td>( 1 \times (K + 1) )</td>
<td>( 1 \times (J + 1) )</td>
<td>-</td>
<td>-</td>
<td>( 1 \times (I) )</td>
<td>( 1 \times ((2^K \times 2^f) \times I) )</td>
<td>( 1 \times (9) )</td>
</tr>
</tbody>
</table>
of two LUTs that are 16 b (16 configurable logic block cells called as slices, composed mainly belongs to the Spartan 3 family from Xilinx, has basic enumerated in FPGA XC3S200-FT256-4. This FPGA, which the second smallest FPGA of the family, which has 1920 slices.

3) The number of clock cycles $n_{15}$ elapsed by the $t_{\text{calc}}$ circuit has not been added to the total number of clock cycles elapsed by the 2P_SVPWM and NP_SVPWM circuits, i.e., $n'_{\text{clk}} = n_2 + n_3 + n_4$ and $n_{\text{clk}} = n_1 + n_2 + n_3 + n_4 + n_6$. It is because $t_{\text{calc}}$ simultaneously runs with D_calc and Vd_extr, and the sum of the number of clock cycles used by these two circuits is always greater than the number of clock cycles used by the $t_{\text{calc}}$ circuit ($n_3 + n_4 > n_5$). Fig. 3 shows the number of cycles elapsed by the circuits in the particular case $P = 3$.

Since the circuit is synchronous, the calculation time can be calculated by taking into account the clock frequency and the number of clock cycles needed by the modulation algorithm. It is also important to remark that only the number of phases $P$ has influence on the circuit speed.

D. Implementation Results

The parameterizable NP_SVPWM circuit has been implemented in FPGA XC3S200-FT256-4. This FPGA, which belongs to the Spartan 3 family from Xilinx, has basic configurable logic block cells called as slices, composed mainly of two LUTs that are 16 b (16 × 1) and two flip-flops. This is the second smallest FPGA of the family, which has 1920 slices (thus, 3840 LUTs and 3840 slice flip-flops).

Table IV shows the resources used by the NP_SVPWM circuit, with different parameter values, obtained through the Foundation ISE 10.1 tool. The parameters that have a major influence are the number of phases $P$ and the number of bits $Q$. The number of levels $N$ affects in a lesser extent, and if its increment does not require more bits for its representation, it does not affect the amount of resources used at all. For instance, in the five-phase case, the resources used by the circuit for five and seven levels are the same. In the same case, with nine levels, the resources used increase a little because of the extra bit required to represent the number nine. Any other FPGA with the same SRAM technology from other manufacturer will give similar implementation results. As an example, the implementation of the NP_SVPWM circuit with $N = 5$, $P = 5$, and $Q = 9$ in FPGA EP2C5F256C6, which is the smallest FPGA of the Cyclone II family from Altera, uses 242 logic elements (out of 4608), 136 flip-flops (out of 4608), and 768 memory bits (out of 119808).

Table V shows the maximum converter switching frequency $f_s$ that can be achieved with the NP_SVPWM circuit. The value of $f_s$ is

\[
 f_s \leq \frac{1}{n_{\text{clk}} \tau_{\text{clk}}} \quad (11)
\]

where the number of clock cycles $n_{\text{clk}}$ is obtained from Table III and the minimum applicable clock period $\tau_{\text{clk}}$ is extracted from the implementation tool report. The maximum switching frequency of the modulator circuit in Table V is, in all cases, higher than the typical switching frequencies used with the multilevel power converters. The implementation results, i.e., the resources used and the clock delay, can slightly vary from one implementation to another, with no circuit modifications due to the non deterministic nature of the place and route algorithms.

The time resolution of the modulation circuit is given by \( \delta = \max(\tau_{\text{clk}}, \tau_m) \), where $\tau_{\text{clk}}$ is the clock period applied to the circuit and $\tau_m$ is the mathematical resolution of the algorithm. The value of $\tau_m$ can be calculated from the converter switching frequency and the parameter $Q$ as

\[
 \tau_m = \frac{1}{2^Q f_s} \quad (12)
\]

IV. Verification

The NP_SVPWM circuit was tested by simulation and in the laboratory with a five-level five-phase converter. Nine bits were considered to represent the fractional part of the reference vector. Therefore, the svpwm_constants.vhd package is configured with $N = 5$, $P = 5$, and $Q = 9$, as shown in Table I.

A. Simulation Results

Figs. 6 and 7 show the simulation results of the multilevel multiphase SVPWM circuit. All of them have been obtained
through the Modelsim simulation tool from Mentor Graphics in its XE (Xilinx Edition) version.

Fig. 6 shows the post place and route simulation corresponding to one switching cycle. The values of the reference vector $v_r$, the switching vectors $v_{s,j}$, and the dwell times $t_j$ are included in binary and real formats to facilitate the circuit analysis. The values in real format have been obtained through the VHDL test bench because the real format is not synthesizable in VHDL. In Fig. 6(a), the activation of the $Init$ signal indicates the beginning of the calculations. After that, the reference vector component values are read by the $Vr_{dec}$ circuit. The reference values $v_{r0}^c = 1.43$, $v_{r1}^c = 1.13$, $v_{r2}^c = -0.73$, $v_{r3}^c = -1.58$, and $v_{r4}^c = -0.25$ match those of the example in [40]. Fig. 6(b) shows the final values of the final switching vector component values ($Vs$ and $Vs_{int}$) and their corresponding switching times ($t$ and $t_{real}$) once the modulator calculations have ended. The activation of the $End$ signal, which indicates the end of the calculations, is not shown in the figure.

The results match the theoretical results obtained in the example in [40]

$$v_{s1} = [1, 1, -1, -2, -1]^T \quad t_1 = 0.25$$
$$v_{s2} = [1, 1, -1, -2, 0]^T \quad t_2 = 0.32$$
$$v_{s3} = [2, 1, -1, -2, 0]^T \quad t_3 = 0.01$$
$$v_{s4} = [2, 1, -1, -1, 0]^T \quad t_4 = 0.15$$
$$v_{s5} = [2, 1, 0, -1, 0]^T \quad t_5 = 0.14$$
$$v_{s6} = [2, 2, 0, -1, 0]^T \quad t_6 = 0.13.$$  \hspace{1cm} (13)

Fig. 7 shows the functional simulation of the SVPWM circuit when the reference is a balanced five-phase sinusoidal wave. In this figure, $Vr(a)$ stands for the reference voltage of phase $a$ $v_r^a$. The reference for the other four phases $Vr(b)$, $Vr(c)$, $Vr(d)$, and $Vr(e)$, not shown in the figure, is equal to $Vr(a)$ with the appropriate phase offset. Traces $Vo(a)$ and $Vo(b)$ are the simulated filtered output voltages of phases $a$ and $b$, respectively. Those signals have been calculated through the VHDL test bench as $v_o^k = \sum_{j=1}^{6} v_s^k t_j$. The trace $Vo(a)$ is equal to the reference $Vr(a)$ delayed with one sampling cycle. $Vo(b)$ is equal to $Vo(a)$ shifted $2\pi/5$ rad. The rest of the output signals, not shown in Fig. 7, is also correct. Therefore, the SVPWM modulator has been properly described in VHDL.

All simulations in this section have also been carried out with the Altera edition of the Modelsim simulation tool for FPGA EP2C5F256C6, providing the same results.

B. Experimental Results

The $NP$ SVPWM circuit was tested by using the experimental setup shown in Fig. 8, which includes a dSPACE platform, an FPGA board, an inverter, and a load. The dSPACE
DS1103 PPC Controller Board provides the reference vectors to the modulation circuit. The SVPWM circuit has been implemented in an S3 board from Digilent Inc., which includes the XC3S200FT256-4 FPGA considered in Section III-D. Two auxiliary boards have been built to adapt the FPGA signals (3.3 V) to the dSPACE control board and the optical link that sends the trigger signals to the inverter (both 5 V). The 74LVC4245A integrated circuit was used to shift the voltage levels of the trigger signals generated by the FPGA control the 40 transistors of the five-level five-phase cascaded full-bridge inverter shown in Fig. 9, which has 3125 different switching states [45]. The dc source voltage of each full-bridge cell is 60 V. Therefore, the inverter voltage step \( V_{dlc} \) is also 60 V. The load is a five-phase distributed–concentrated winding induction motor of four poles in parallel with a five-phase \( RL \) circuit. This motor was specifically built for the tests by rewinding the stator phases on the 30 stator slots of one 1-kW three-phase motor. Each phase of the \( RL \) circuit is an \( L = 15 \) mH series connected with \( R = 300 \) Ohm.

In order to test the new VHDL module, two specific auxiliary VHDL modules (dSPACE_com and Trigger) were developed to interface the \( NP_{SVPWM} \) circuit with the dSPACE control board and the power converter, respectively. Their connection inside the FPGA is shown in Fig. 10.

1) dSPACE_com: This circuit receives the normalized reference vector \( \mathbf{v}_r \) given by the dSPACE control system and stores its components in a dual-port RAM that is read by the \( NP_{SVPWM} \) circuit. The dSPACE_com circuit uses a two-phase communication protocol (validation–acknowledge) specifically designed for this application. It is composed of various synchronizing flip-flops, one edge detector, one timer, one counter, one dual-port RAM, and one FSM.

2) Trigger: This circuit reads the dual-port RAMs \( V_s(ks,js) \) and \( t(j) \) that belong to the \( NP_{SVPWM} \) circuit to calculate the trigger signals \( T \) for the power transistors. Since the relationship between the transistor trigger signals and the converter output levels is different for each multilevel converter topology [61], this circuit is specifically designed for a five-level cascaded full-bridge inverter. It is composed of the following four subcircuits: \( \text{Switch}_\text{timer}, \text{Change}_\text{instant}, \text{Switch}_\text{state}_\text{sequencer}, \) and \( \text{Trigger}_\text{signals}. \)

\( \text{Switch}_\text{timer} \) calculates the current time within the switching period by counting the pulses of the \( \text{clk} \) signal. The \( \text{Change}_\text{instant} \) circuit calculates the absolute time instants in which the converter must change the switching state. The \( \text{Switch}_\text{state}_\text{sequencer} \) circuit compares the absolute time instants with the current time provided by \( \text{Switch}_\text{timer} \) to determine the current switching vector, i.e., the current output level of every phase. The \( \text{Trigger}_\text{signals} \) circuit translates the current output level into trigger signals. Additionally, it inserts the dead times by delaying the rising edges of the complementary trigger signals. The Trigger circuit is composed of \( 5 + 8P \) counters, \( 3 + 17P \) registers, \( 2P + 2 \) dual-port RAMs, \( 1 + 9P \) multiplexers, two comparators, \( 4P \) combinational functions, and one FSM.

As shown in Table V, the modulation circuit can deal with a switching frequency of up to 362 kHz. Nevertheless, a switching frequency of 10 kHz was used in the experimental test because the dSPACE platform is slower (it limits the switching frequency to 10.4 kHz). In this case, from (12), the mathematical resolution of the algorithm is \( \tau_m = 195 \) ns, which is longer than the FPGA clock period \( \tau_{\text{clk}} = 20 \) ns. Thus, the time resolution of the circuit is also \( \delta = 195 \) ns. Figs. 11 and 12 show the experimental measurements obtained when a purely sinusoidal voltage reference is considered. In Fig. 11, channel 1 shows the leg voltage of phase \( a \), channel 2 is the same signal after being filtered, channel 3 shows the current drawn by phase \( a \) of the motor, and channel 4 is the voltage across the resistor of the \( RL \) load in parallel with the motor. The harmonic content of the output voltage is depicted in Table VI. The amplitude of the low-order harmonics is below 1% of the fundamental, and the total harmonic distortion of the
Fig. 10. Control.

Fig. 11. Voltages and current of phase $a$.

Fig. 12. Trajectories of the output voltage vector in the $dq$ planes.

Table VI

<table>
<thead>
<tr>
<th>Harmonic Content of the Output Leg Voltage Waveform</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>100%</td>
</tr>
</tbody>
</table>

The voltage waveform is 2.44%. The current drawn by the motor (CH3) presents a nonsinusoidal shape that is caused by the load characteristic of the motor operated in open loop (probably by mechanical ringing). The current drawn by the RL load, which has the same shape as that of the resistor voltage (CH4), does not show a low harmonic distortion as expected. Fig. 12 shows the trajectories of the output voltage vector in the $dq$ planes. The gray trace corresponds to the output voltage, and the black trace corresponds to the filtered output voltage. The later trajectory is a circle in the $d_1$–$q_1$ plane and a point in the $d_2$–$q_2$ plane that corresponds to a pure sinusoidal output, which proves that the system properly operates.

The NP_SVPWM module was tested with a converter with externally supported dc sources, which alleviates the voltage capacitor balancing issue present in a number of applications. The SVPWM algorithm in [40] is a simple modulation technique that generates an output phase-to-neutral voltage that is equal to the reference phase-to-neutral voltage. Consequently, this multilevel multiphase algorithm does not handle joint-phase redundancy, and it cannot integrate a general balancing capacitor technique. Nevertheless, in multilevel topologies with per-phase redundancy [61], such as flying capacitor and cascaded full-bridge converters, the Trigger_signals subcircuit in the Trigger auxiliary circuit can be designed to select the appropriate trigger signals from the switching vectors to the balance voltage capacitors. If the neutral voltage is not a constraint, then the SVPWM algorithm in [41], which handles joint-phase redundancy, is a more appropriate modulation technique because it has a degree of freedom that can be used for voltage capacitor balancing. Additionally, the NP_SVPWM module requires balanced dc voltages. In the case of unbalanced dc voltages, the output voltage distortion can be reduced with the specific multilevel multiphase feedforward SVPWM technique in [62].

V. Conclusion

A generic digital parameterizable VHDL module for the recent multilevel multiphase SVPWM algorithm has been presented. The parameters of the module are the number of phases, the number of levels of the converter, and the number of bits of the fractional part of the reference vector.

The new module has a modular design. It is technology independent as it was described using the standard VHDL sentences. Thus, although it was optimized for the common SRAM FPGAs, it can be synthesized with programmable devices of any manufacturer or even ASICs. The hierarchical structure, the logical resources, and the calculation time of the circuit are detailed throughout this paper. The resources used basically depend on the number of phases of the inverter and the number of bits used to represent the fractional part of the reference
vector. The number of clock cycles only depends on the number of phases. Thus, the maximum modulation frequency that is achievable with the circuit mainly depends on the number of phases.

The digital parameterizable VHDL module has been implemented in a low-cost SRAM FPGA, and it has been tested with a five-level five-phase voltage source inverter.

REFERENCES


