Abstract
In this work, a simplified startup method for AC synchronous machines is solved by means of a simplified algorithm based only in feeding the AC synchronous machine with stator voltages with a ramp frequency. The implementation of the algorithm along with the Space Vector Pulse Width Modulation technique is carried out by means of a Field Programmable Gate Array (FPGA) device, where Very High Speed Integrated Circuit, Hardware Description Language (VHDL) modules are custom designed. Experimental results were carried out for the ventilation of a greenhouse. The fan drive is turned on according to an existing control strategy designed for an optimal ventilation of the greenhouse. The designed algorithm with custom VHDL modules yields to a great performance for the fan drive in terms of smooth transient currents. The proposed algorithm was also implemented with an embedded microprocessor or softcore in an FPGA device for comparison purposes, yielding to same results. But in the case of optimal utilization of FPGA resources and the speed of execution of the algorithm, the VHDL modules custom design showed the best results.

Keywords: Startup, AC synchronous motors, Field programmable gate arrays, Fan drive

1. Introduction

Electrical drives [20], [29] can be classified into two groups: Alternate Current (AC) an Direct Current (DC) machine drives. DC machine drives are very popular due to a high torque, high efficiency, and are controlled by adjusting the magnitude of the supply voltage. Some drawbacks of these motors are the service requirements and the relative high costs. On the other hand, three-phase AC machines can be classified in asynchronous, synchronous and variable reluctance machines. The asynchronous or induction motor (IM) is a popular machine widely used in industrial drive systems [3], [6], due to their simple mechanical construction, low service requirements, and lower costs with respect to DC motors. Synchronous machines are classified in sinusoidal, brushless and reluctance machines, where the most popular are the sinusoidal which is characterized by a sinusoidal Back Electro-Motive Force (BEMF) and the brushless that is characterized by a trapezoidal BEMF and by using permanent magnets [23]. The sinusoidal motor is classified depending on the manner the magnetic field is provided: wound field motors and Permanent Magnet Motors (PMM) [7], [5], [22], where PMM are at the same time classified in surface mounted PMM and interior mounted PMM. In particular, PMM have attracted the attention of control engineers practitioners due to its properties as high efficiency, minimal maintenance requirements and high torque-inertia ratio, therefore they are widely used in household appliances and in several industrial fields [18] with applications as fans, pumps and compressor type loads.

The startup of three-phase AC motors by directly being connected to a line voltage is not the best way due to large transient current values. Hence, there are several methods for the startup of electric motors, where common methods are [9]:

- **Direct start method**, connects the motor directly to the line voltage but can only be applied to asynchronous motors;
- **Reduced voltage starting**, also known as soft start is commonly carried out with thyristors where...
the firing angle is linearly incremented resulting in a voltage ramp that can go from zero to full voltage.

- **Soft start with control**, consists in adding to the soft start method a current limiter by means of a control loop.

The direct start is a common method used in industrial process but the energy consumption can not be satisfactory due to large transient current values. The soft start method reduces the acceleration torque produced by the motor, as a consequence more time is required to move the load. Soft start with control can significantly reduce large transient current values by adding complexity to the startup algorithm. The direct start method when applied to an AC synchronous motor may be accompanied with a temporary reverse rotation or may lead to a starting failure. The AC synchronous motor can be accelerated from standstill to a certain speed by applying a ramp frequency signal into the stator voltages in a similar fashion as in the well known scalar \( V/f \) control method [1]. The drawback of the \( V/f \) control technique is that requires of an extra circuitry and a control logic for the regulation of the voltage magnitude.

In general, in supplying voltages to AC motors (generated in open-loop or closed-loop), a three-phase inverter is required and a digital device capable of generating the Space Vector Pulse Width Modulation (SVPWM) technique. The SVPWM technique is becoming a popular modulation technique for three-phase AC motors due to the fact that it reduces the total harmonic distortion [28]. Nowadays, digital microprocessor technology [27] make feasible the implementation of SVPWM techniques for the digital generation of approximated sinusoidal signals. Digital devices as microprocessors or Digital Signal Processors (DSP) [26] are software-based devices where algorithms are executed in sequence. Since instructions are interpreted line by line, intrinsic time delays can lead to a poor performance. Also, computational resources are reduced with SVPWM algorithms due to the fact that some resources are devoted to periodic timing events such as sampling, SVPWM algorithm calculation and gating signals generation, making difficult to incorporate other algorithms. An alternative approach is the use of Field Programmable Gate Arrays (FPGA) [10], [13], for implementing the SVPWM algorithm. The advantages of these devices are due to the fact that it is a hardware-based technique where some set of instructions can be executed in parallel, rapid prototyping, simpler hardware and software designs, high speed performance and reutilization of blocks for future designs.

Due to its features, an FPGA device is considered as an appropriate solution that can handle the computational burden of a SVPWM algorithm. Recent researches regarding the implementation of the SVPWM algorithm in FPGA devices have been reported in the literature. In [12], multilevel SVPWM algorithms are analyzed and implemented into an FPGA by means of Matlab® and Xilinx® System Generator for Simulink, where optimization of algorithms is not performed. A shifted SVPWM method to improve the operation of dc-link resonant inverters is partially implemented in a DSP and a FPGA device in [16]. In [14] a variable common mode rejection pulse width modulation algorithm for a neutral point clamped inverter is implemented in an FPGA. Such implementation is not described, only the illustration of the main RTL schematic diagram is provided, making it difficult in that way to reproduce the reported results. The reader can find a complete survey of FPGA design methodology for industrial control systems as presented in [13], for the control of AC machine drives in [15], for the tuning of control systems in [8], [18], and for control of industrial applications in [21], [17], [19].

One of the industrial applications where the assisted startup of AC electric drives is necessary, is with fan drive systems employed in greenhouses. In warm regions and during the day, an excessive accumulation of heat inside greenhouses is registered. This particular problem was solved in [2] by reducing the heat inside a greenhouse in an efficient way, consisting in solving an energy optimization problem which was stated in the following manner: How to maintain the inner temperature of a greenhouse under an established bound and, at the same time, minimize the time lapse that the fan drives are turned on, and consequently reduce the consumption of energy? Although in [2] the time lapse that the fan drives are turned on is minimized, the startup of these drives are carried out with the direct start method, resulting in large transient currents each time the motor is turned on.

Hence, based on all of the above exposed, the objectives of this work are to present a simple startup method design for the broad class of AC synchronous machines, relying on supply voltages with a ramp-
shape frequency. Also to present the implementation design of the supply voltages along with the SVPWM algorithm in a FPGA with a Very High Speed Integrated Circuit, Hardware Description Language (VHDL) custom design. Finally, to present the real-time results when the synchronous machine is used as a fan drive for the heat reduction in a greenhouse.

The rest of this work is of the following form: Section 2 deals with the startup algorithm for AC synchronous machines. Section 3 shows the FPGA implementation of the proposed startup algorithm along with the SVPWM algorithm. Experimental results are shown in Section 4 and finally some comments conclude the work in Section 5.

2. Startup of AC synchronous machines

In particular, synchronous PMM have a three-phase stator winding and a rotor with surface or interior mounted permanent magnets. Surface mounted PMMs are for low velocity applications and interior mounted PMM for higher velocities. In the case of wound field synchronous motors, the magnetic field is provided by a field winding in the rotor where a mechanical commutator is necessary as in DC motors. In any case, the induced torque of AC synchronous machines is as follows [4]:

\[ \tau_{\text{ind}} = kB_R \times B_S \]

where \( B_R \) and \( B_S \) are the rotor and stator magnetic fields respectively. At standstill, when the stator is connected to a three-phase power supply of frequency \( f \), the magnetic field of the stator \( B_S \) starts to rotate meanwhile the magnetic field of the rotor \( B_R \) is stationary. Fig. 1 illustrates the interactions between \( B_R \) and \( B_S \) in a cycle of duration \( 1/f \). It can be appreciated that the average induced torque in one cycle is zero, where the motor is characterized by severe vibrations followed by excessive heating.

Therefore if at initial time, the frequency fed to the stator is slowly increased, then the average torque will be \( |\tau_{\text{ind}}| > 0 \) and the motor will start to accelerate in each cycle, until a nominal velocity is reached.

Synchronous machines rotate to a synchronous velocity, i.e., the rotor velocity is uniquely related to the supply frequency provided in the stator voltages by the following relation [4]:

\[ \omega_s = \frac{2\pi f_c}{p} \] \hspace{1cm} (1)

where \( \omega_s \) is the synchronous velocity of the rotor, \( f_c \) is the supply frequency in the stator voltages, and \( p \) is the number of pole pairs. Based on this fact, a time-varying frequency is proposed as follows:

\[ f(t) = \begin{cases} f_m t + f_b & t_s \geq t \geq 0 \\ f_c & t > t_s \end{cases} \] \hspace{1cm} (2)

Figure 1: Magnetic field alignments during one electrical cycle of duration \( 1/f \). a) At time \( t = 0 \) s the induced torque is \( \tau_{\text{ind}} = 0 \). b) At time \( t = 1/(4f) \) s the induced torque is \( \tau_{\text{ind}} > 0 \). c) At time \( t = 1/(2f) \) s the induced torque is \( \tau_{\text{ind}} = 0 \). d) At time \( t = 3/(4f) \) s the induced torque is \( \tau_{\text{ind}} < 0 \). e) At time \( t = 1/f \) s the induced torque is \( \tau_{\text{ind}} = 0 \).

where \( f_{m} \) is the slope of the frequency ramp, \( f_{b} \) is the initial frequency value that is chosen close to zero, so the motor is not fed with zero frequency at an initial stage, \( f_{c} \) is a constant frequency value, and \( t_{s} \) is the time instant where the frequency changes from ramp to a constant value. Based on (1), the corresponding mechanical rotor position results of the following form:

\[ \theta_m(t) = \begin{cases} \frac{2\pi f_m t^2}{2p} + \frac{2\pi f_b t}{p} & t_s \geq t \geq 0 \\ \frac{2\pi f_c t}{p} & t > t_s \end{cases} \] \hspace{1cm} (3)

For determining the rotor velocity \( \omega_m(t) \), let us take the time derivative of (3)

\[ \frac{d\theta_m(t)}{dt} = \omega_m(t) = \begin{cases} \frac{4\pi f_m t}{2p} + \frac{2\pi f_b}{p} & t_s \geq t \geq 0 \\ \frac{2\pi f_c}{p} & t > t_s \end{cases} \] \hspace{1cm} (4)

It is clear that the rotor velocity will be a ramp shape signal for \( t_s \geq t \geq 0 \), and will keep constant for \( t > t_s \). In order to match this constant value to the final value of the ramp, we propose the following relation

\[ \frac{2\pi f_c}{p} = \frac{4\pi f_m t_s}{p} + \frac{2\pi f_b}{p} \]

and by solving for \( f_c \) results in:

\[ f_c = 2f_m t_s + f_b \]
This frequency value is greater than \( f(t_s) = f_m t_s + f_b \) in (3), i.e., frequency \( f(t) \) will have the form shown in Fig. 2 a), but yielding to a desired rotor velocity.

**Remark 1.** If \( f_c \) is chosen as the frequency value just before the switching in (3), i.e., \( f(t_s) = f_m t_s + f_b \), then the rotor velocity profile will switch from a ramp rotor velocity to a constant lower value as shown in Fig. 2 b).

Figure 2: Rotor velocity profiles. a) With \( f_c = 2f_m t_s + t_b \). b) With \( f_c = f_m t_s + t_b \).

Now, we summarize the startup algorithm for PMM:

1. Define the time instant \( t_s \) where the frequency switches from ramp to constant, and the initial frequency value \( f_b \) close to zero.
2. Define the constant value for the rotor velocity \( \omega_m(t_s) \).
3. Calculate the slope frequency as \( f_m = (P\omega_m(t_s) - 2\pi f_b)/(2\pi t_s) \).
4. Calculate the constant frequency as \( f_c = 2f_m t_s + f_b \).
5. Form the frequency signal \( f(t) \) in (3) with the calculated values.
6. Generate the stator voltages with this frequency in the frame fixed to the stator known as \((\alpha, \beta)\) reference frame [11]

\[
\begin{align*}
    u_\alpha &= V_m \cos(2\pi f(t)t), \\
    u_\beta &= V_m \sin(2\pi f(t)t).
\end{align*}
\]

7. Feed the stator voltages to the SVPWM algorithm (see Appendix A) as shown in Fig. 3.

**Remark 2.** If the motor is started with a given load torque, the synchronous motor will develop the right torque to raise the rotor velocity to the synchronous one. Even if the load torque suffers an increment, the rotor velocity will initially slow down, but eventually the developed torque will increase, so the motor rotates again to the synchronous velocity [4]. This is an attractive feature of AC synchronous machines.

**Remark 3.** Although asynchronous motors do not present the startup problem with the direct start method as the synchronous ones, the assisted startup of asynchronous motors is recommended. The proposed method can be applied to asynchronous motors by considering the following facts: The rotor velocity of asynchronous motors is \( \omega_m = (1-s)\omega_{sync} \), where \( \omega_m \) is the rotor velocity, \( s \) is the motor slip that depends on the rated power and the percentage of the applied rated load torque of the motor, \( \omega_{sync} \) is the synchronous velocity of the magnetic field and is defined as \( \omega_{sync} = 4\pi f_c / p \), with \( p \) as the number of pole pairs, and \( f_c \) as the electrical frequency of the supply voltages that can be proposed as in (3).

3. **FPGA implementation**

The startup algorithm for synchronous PMM is implemented in FPGAs with a VHDL custom design. The programmed frequency signal was proposed as follows:

\[
    f(t) = \begin{cases} 
        2.39t + 2.8 & 10s \geq t \geq 0s \\
        50.6 & t > 10s,
    \end{cases}
\]

for a desired steady-state rotor velocity of 28 rad/s. The designed modules for the startup algorithm are developed in VHDL and are implemented in a Virtex-5® FPGA platform from Xilinx® [24]. The employed platform is a ML506 board [25] that incorporates the above mentioned FPGA. The modules are developed by means of ISE® Design Suite from Xilinx and for embedded ROM memory blocks, CORE Generator® System is used.

In general, the modules for the proposed algorithm must agree with the following steps (see Appendix A):

1. Establish the time periods for the ramp and constant frequencies used in the input voltages.
2. Determine the angle and sector that correspond for voltage vector \( u = (u_\alpha, u_\beta)^T \).
3. Calculate time values for $T_0$, $T_1$ and $T_2$.
4. Determine the time patterns for the active vectors according to the 7-segment algorithm.
5. Generate the output signals $sw_1, \ldots, sw_6$.

In order to accomplish the latter points, four main modules are proposed as illustrated in Fig. 4. The clock module is the one that provides the necessary commutation times for developing the SVPWM algorithm (Appendix A), so, this module is directly connected to the internal clock of the evaluation board. The counter module generates the angle for the sinusoidal input voltages according to the ramp and constant frequencies. The angle and sector for a given voltage vector $u$ are also determined in this module. The time values module calculates values for $T_0$, $T_1$ and $T_2$ in each commutation interval according to the specific position of vector $u$. Finally, the output stage module determines the time intervals for implementing the 7-segment algorithm and generates the output signals $sw_1, \ldots, sw_6$. Each module will be detailed in the following lines.

3.1. Clock module

Due to the synchronization of all functions, base time generation is a very important issue for the SVPWM algorithm. Hence, the clock module is the one that provides the switching times that are needed for the SVPWM algorithm. This module is directly interconnected to the 100MHz clock signal of the ML506 platform. Based on this signal, clock signals of 1ms and 1µs are derived; for that reason two counters are implemented. In the first counter with a base time period of 10ns, the 1µs output is complemented each 50 cycles. Using the 1µs output signal as the switching signal for the second counter, each 500 cycles the 1ms output signal is then complemented.

3.2. Counter module

This module generates the proper rotation of vector $u$ by determining the corresponding angle $\theta$, it also provides the sector information of the actual angle inside the hexagon. This module generates the $\theta$ angle based on the frequency profile described in Section 2, by means of the relation $\theta = \omega t = 2\pi ft$. A 17-bit resolution is required for its representation, 9 bits are used for the integer part and 8 bits for the fractional one. Actually, the generation of $\theta$ is precalculated and saved in a memory. Thus in the ramp frequency interval, a variable named ‘index’ is used for indexing a memory containing the calculated values for $\theta$, where a modulo operation with $2\pi$ results in several sweeps of the hexagon by vector $u$ (from 0° to 360°). As the ‘index’ variable is incremented, the time interval for sweeping the hexagon is reduced, until the constant frequency operation mode is reached. The counter module also determines the time interval for the ramp frequency used in the startup process and the change to a constant frequency for the steady-state operation of the electric motor. A variable ‘stable’ was used in this module for indicating the ramp frequency (stable=0) or constant frequency operation (stable=1). Fig. 5 shows a graphical depiction of both processes.

3.3. Time values module

This module is the one that calculates the switching time $T_0$ (A.8), $T_1$ (A.6) and $T_2$ (A.7) with specific arithmetic units. Assigning $T = 1$ms, $V_m = 180$V and $V_{DC} = 270$, constant $\tau$ in (A.6) and (A.7) results
in the value of $8.6602 \times 10^{-4}$s and it is represented with a 16-bit resolution, meanwhile the sine function is represented with a 18-bit resolution. The bit size for the switching times $T_0$, $T_1$, and $T_2$ are shown in Table 1.

<table>
<thead>
<tr>
<th>Sw. time</th>
<th>Range ($I_{\text{min}}, I_{\text{max}}$)</th>
<th>bit size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0$</td>
<td>$(9.93 \times 10^{-4}, 9.94 \times 10^{-4})$</td>
<td>32 bits</td>
</tr>
<tr>
<td>$T_1$</td>
<td>$(1.7924 \times 10^{-11}, 5.55 \times 10^{-6})$</td>
<td>32 bits</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$(0.55 \times 10^{-6})$</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

Table 1: Bit size and value range for switching time.

A block diagram of the time values module is shown in Fig. 7. The block named $S_{60}$ performs the arithmetic operation $\text{SECTOR} \times 60^\circ$. The output of this block is implemented with a 9-bit resolution in order to represent the largest angle of $360^\circ$. Then, the blocks designated as $m$ and $n$ perform the argument of the trigonometric functions in (A.6) and (A.7) respectively. These arguments are used for the parallel calculation of the sine function in the blocks designated as $\sin_m$ and $\sin_n$. The input to these blocks is a 18-bit resolution signal where 1 bit is for the sign, 9 bits are for the integer part and 8 bits are for the fractional part. The output is also in a 18-bit resolution where 1 bit is for the sign, 1 bit for the integer part and 16 bits for the fractional part. Here, the sine function is implemented in an embedded memory. As suggested by the name of block $\text{Calc}_T1\_T2$, this calculates the time values for $T_1$ and $T_2$ by multiplying the trigonometric functions by constant $\tau$ as in (A.6) and (A.7) respectively. The response of this block depends on the latency of the blocks $\sin_m$ and $\sin_n$ due to the use of an embedded memory. The outputs of this block have a 32-bits resolution. Finally, the block named $\text{Calc}_T0$ calculates the time value $T_0$ as in (A.8).

### 3.4. Output stage module

The output stage block determines the logic switching sequence signals that are directly connected to the three-phase inverter. The input signals to this block are $T_0$, $T_1$, $T_2$ and the sector information. The synchronization signals are the 1$\mu$s and 1$ms$ signals generated by the clock module. Fig. 8 shows a block diagram of this module.

The mod_block is the one that realizes the time division for $T_0$, $T_1$ and $T_2$ in order to obtain the time intervals $T_0/4$, $T_1/2$ and $T_2/2$. The division is implemented with shift registers due to the fact that the time values are divided by $2^n$, then the resulting bits are rearranged. The counter block is a binary counter that counts 1000 cycles of the $1\mu$s clock signal that allows to form the 1$ms$ base signal for the SVPWM. The block gen_inter determines the time intervals of each segment according to the 7-segment algorithm [28] as shown in Fig. 9.

These time intervals are determined as follows: $l = T_0/4$, $m = l + T_1/2$, $n = m + T_2$, $o = 1ms - 0$, $p = 1ms - m$ and $q = 1ms - l$. Then these values are compared with the output of the counter block (variable “ent”) as illustrated in Fig. 10.

The output of this block is the variable “sec” which is assigned an address value for a table available in the Tables block. The sequence “sec” includes a security
Figure 9: Time intervals \( l, m, n, o, p, q \) measured from \( KT \) where \( K \in \mathbb{N}, T = 0.001s \).

Figure 10: Flow diagram of the comparator block.

4. Experimental results

In this section one verifies the performance of the proposed design for the startup of a fan drive (syn-

Table 2: Digital output signals for Sector I, where the upper part corresponds to \((sw_1, sw_3, sw_5)\) and the lower one to \(sw_2, sw_4, sw_6\).

<table>
<thead>
<tr>
<th>Upper</th>
<th>000</th>
<th>100</th>
<th>110</th>
<th>111</th>
<th>110</th>
<th>100</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_0 )</td>
<td>( T_0 )</td>
<td>( T_1 )</td>
<td>( T_2 )</td>
<td>( T_0 )</td>
<td>( T_2 )</td>
<td>( T_1 )</td>
<td>( T_0 )</td>
</tr>
<tr>
<td>( I_0 )</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td>111</td>
<td>011</td>
<td>001</td>
<td>000</td>
<td>001</td>
<td>011</td>
<td>111</td>
</tr>
</tbody>
</table>

Table 3: Digital output signals for Sector II, where the upper part corresponds to \((sw_1, sw_3, sw_5)\) and the lower one to \(sw_2, sw_4, sw_6\).

<table>
<thead>
<tr>
<th>Upper</th>
<th>000</th>
<th>110</th>
<th>010</th>
<th>111</th>
<th>010</th>
<th>110</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_0 )</td>
<td>( T_0 )</td>
<td>( T_1 )</td>
<td>( T_2 )</td>
<td>( T_0 )</td>
<td>( T_2 )</td>
<td>( T_1 )</td>
<td>( T_0 )</td>
</tr>
<tr>
<td>( I_0 )</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td>111</td>
<td>001</td>
<td>101</td>
<td>000</td>
<td>101</td>
<td>001</td>
<td>111</td>
</tr>
</tbody>
</table>

Table 4: Digital output signals for Sector III, where the upper part corresponds to \((sw_1, sw_3, sw_5)\) and the lower one to \(sw_2, sw_4, sw_6\).

<table>
<thead>
<tr>
<th>Upper</th>
<th>000</th>
<th>010</th>
<th>011</th>
<th>111</th>
<th>011</th>
<th>010</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_0 )</td>
<td>( T_0 )</td>
<td>( T_1 )</td>
<td>( T_2 )</td>
<td>( T_0 )</td>
<td>( T_2 )</td>
<td>( T_1 )</td>
<td>( T_0 )</td>
</tr>
<tr>
<td>( I_0 )</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td>111</td>
<td>101</td>
<td>100</td>
<td>000</td>
<td>100</td>
<td>101</td>
<td>111</td>
</tr>
</tbody>
</table>
chronous PMM drive) for the ventilation of a greenhouse. Fig. 12 shows the exterior of the greenhouse. Also, the performance of the VHDL modular design in a customized fashion is verified. Regarding the FPGA implementation, a comparative study is carried out with a softcore. It is worth mentioning that softcores are becoming a handy solution for FPGA based implementations.

4.1. Experimentation with custom design VHDL modules

The implementation in the reconfigurable FPGA device is a Virtex-5® (XC5VSX50T–1FF1136) platform model ML506 from Xilinx®[24] that is shown in Fig. 13.

Fig. 14 illustrates the Register Transfer Level (RTL) implementation of the SVPWM algorithm where inputs and outputs are shown. Table 8 shows the logic input combinations for “rst” and “start” and the resulting action in the embedded system.

When the SVPWM module is synthesized, one obtains the corresponding occupation in the recon-figurable FPGA device Virtex-5® from Xilinx® as shown in Table 9. It can be noted the advantage of a custom design of VHDL modules due to a reduced utilization of the FPGA resources. Leaving in this way plenty of space for implementing other algo-

Table 5: Digital output signals for Sector IV, where the upper part corresponds to \((sw_1, sw_3, sw_5)\) and the lower one to \(sw_2, sw_4, sw_6\).

<table>
<thead>
<tr>
<th>Upper</th>
<th>000</th>
<th>011</th>
<th>001</th>
<th>111</th>
<th>001</th>
<th>011</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_0)</td>
<td>(S_1)</td>
<td>(S_2)</td>
<td>(S_7)</td>
<td>(S_2)</td>
<td>(S_1)</td>
<td>(S_0)</td>
<td></td>
</tr>
<tr>
<td>(I_0)</td>
<td>(I_1)</td>
<td>(I_2)</td>
<td>(I_0)</td>
<td>(I_2)</td>
<td>(I_1)</td>
<td>(I_0)</td>
<td></td>
</tr>
<tr>
<td>(T_0)</td>
<td>(T_1)</td>
<td>(T_2)</td>
<td>(T_0)</td>
<td>(T_2)</td>
<td>(T_1)</td>
<td>(T_0)</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td>111</td>
<td>110</td>
<td>000</td>
<td>110</td>
<td>110</td>
<td>111</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Digital output signals for Sector V, where the upper part corresponds to \((sw_1, sw_3, sw_5)\) and the lower one to \(sw_2, sw_4, sw_6\).

<table>
<thead>
<tr>
<th>Upper</th>
<th>000</th>
<th>011</th>
<th>001</th>
<th>111</th>
<th>001</th>
<th>011</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_0)</td>
<td>(S_1)</td>
<td>(S_2)</td>
<td>(S_7)</td>
<td>(S_2)</td>
<td>(S_1)</td>
<td>(S_0)</td>
<td></td>
</tr>
<tr>
<td>(I_0)</td>
<td>(I_1)</td>
<td>(I_2)</td>
<td>(I_0)</td>
<td>(I_2)</td>
<td>(I_1)</td>
<td>(I_0)</td>
<td></td>
</tr>
<tr>
<td>(T_0)</td>
<td>(T_1)</td>
<td>(T_2)</td>
<td>(T_0)</td>
<td>(T_2)</td>
<td>(T_1)</td>
<td>(T_0)</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td>111</td>
<td>110</td>
<td>010</td>
<td>000</td>
<td>010</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

Table 7: Digital output signals for Sector VI, where the upper part corresponds to \((sw_1, sw_3, sw_5)\) and the lower one to \(sw_2, sw_4, sw_6\).

<table>
<thead>
<tr>
<th>Upper</th>
<th>000</th>
<th>101</th>
<th>100</th>
<th>111</th>
<th>100</th>
<th>101</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_0)</td>
<td>(S_1)</td>
<td>(S_2)</td>
<td>(S_7)</td>
<td>(S_2)</td>
<td>(S_1)</td>
<td>(S_0)</td>
<td></td>
</tr>
<tr>
<td>(I_0)</td>
<td>(I_1)</td>
<td>(I_2)</td>
<td>(I_0)</td>
<td>(I_2)</td>
<td>(I_1)</td>
<td>(I_0)</td>
<td></td>
</tr>
<tr>
<td>(T_0)</td>
<td>(T_1)</td>
<td>(T_2)</td>
<td>(T_0)</td>
<td>(T_2)</td>
<td>(T_1)</td>
<td>(T_0)</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td>111</td>
<td>010</td>
<td>011</td>
<td>000</td>
<td>011</td>
<td>010</td>
<td>111</td>
</tr>
</tbody>
</table>

Table 8: Logic input combinations for “rst” and “start” and the resulting action.

<table>
<thead>
<tr>
<th>start</th>
<th>rst</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Halt</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Reset, Halt</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Start</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Reset</td>
</tr>
</tbody>
</table>

Figure 12: Greenhouse.

Figure 13: Development platform ML506.
of phase signals with a particular value of 28.09 Hz as proposed in Section 3. Fig. 22 shows the introduced dead time to protect transistors. Here, it can be observed that the upper transistor (signal 1) is activated 1.5 $\mu$s after lower transistor is turned off.

Once the good performance of the proposed embedded algorithm was verified, the main tests were carried out with a high power inverter connected to a synchronous PMM that is used as a fan drive in a greenhouse. For that, a three-phase voltage source supplies a three-phase variable transformer (VARIAC), whose knob allows the regulation of the voltage amplitude. These voltages are fed into a power module (Semikron® IGBT Power Electronics Teaching System), which incorporates a three-phase rectifier and inverter, so that a regulated DC voltage is available for the three-phase inverter. The six digital signals provided by the development platform for switching the IGBT transistors of the inverter are converted

<table>
<thead>
<tr>
<th>Logic elem.</th>
<th>Used</th>
<th>Available</th>
<th>Util. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td>125</td>
<td>32,640</td>
<td>0.38</td>
</tr>
<tr>
<td>LUTs</td>
<td>541</td>
<td>32,640</td>
<td>1.6</td>
</tr>
<tr>
<td>Slices</td>
<td>202</td>
<td>8,160</td>
<td>2</td>
</tr>
<tr>
<td>IOBs</td>
<td>9</td>
<td>480</td>
<td>1.8</td>
</tr>
<tr>
<td>BlockRAM/FIFOs</td>
<td>2</td>
<td>132</td>
<td>1.5</td>
</tr>
<tr>
<td>Memory(KB)</td>
<td>72</td>
<td>4752</td>
<td>1.5</td>
</tr>
<tr>
<td>BUFG/CTRLs</td>
<td>3</td>
<td>32</td>
<td>9</td>
</tr>
<tr>
<td>DSP48Es</td>
<td>6</td>
<td>288</td>
<td>2</td>
</tr>
<tr>
<td>Fanout</td>
<td>3.16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9: Resource occupation.

Figures 15–20 illustrate the registered measurements by sectors that corresponds to those obtained in digital simulations and to the logic sequences presented in Tables 2–7.

Captured phase voltages are illustrated in Fig. 21 where can be appreciated the 120° of phase difference between voltages and the registered frequency...
to a CMOS level of 15 V. This voltage level is the required one for switching on the IGBTs. The three-phase inverter and VARIAC are shown in Fig. 23. The mechanical rotor position is provided by an encoder and the stator phase currents are measured by Hall-type sensors (LEM® HX 10-P). Such signals are acquired by a dedicated encoder connector and by analog to digital converters available in an external DSP board. It is worth mentioning that the DSP board is only used for monitoring variables of the fan drive.

The control algorithm for the optimal ventilation of the greenhouse [2] is running in the Labview software from National Instruments. Initially, the control algorithm puts the FPGA in halt mode according to Table 8. When the control algorithm determines that is time to turn on the drive fan, the FPGA is started.

The obtained rotor velocity profile for the fan drive is shown in Fig. 24, where can be observed an acceleration ramp free of fluctuations in the first 10 s due to the programmed startup ramp in frequency for the stator voltages. After $t > 10$ s the frequency for the stator voltages is kept constant and the rotor velocity remains constant as well. The $\alpha$ and $\beta$ current components are shown in Fig. 25, it can be observed a smooth transient response without peaks and that remains almost constant in amplitude for all time. This type of response is favorable for the power electronics and the fan drive itself.

4.2. Comparison with a softcore implementation

A novel development that is gaining popularity among electronics engineers are softcores or embedded microprocessors in general purpose FPGA devices. Some advantages of softcores are:

1. flexible implementation of the algorithm due to the fact that it is programmed in C/C++ language,
2. required resources for implementing the soft-core can be selected in order to optimize the FPGA space.

On the other hand, disadvantages are:

1. as any other microprocessor, softcores have a fixed execution time for instructions that cannot be optimized,
2. some softcores have an IP that implies an additional cost for the whole setup.

Hence, the proposed algorithm is also implemented in a NIOS® II microprocessor from Altera® (that has an IP) in order to make comparisons with the proposed implementation of the algorithm. The NIOS II® microprocessor is a general purpose flexible core chip with a RISC architecture. This system is equivalent to a microcontroller, which includes a microprocessor and a combination of peripherals and memory in a single chip.

After synthesizing the NIOS II processor (see Appendix B), the occupation in the platform Cyclone IV FPGA (EP4CE2-2F17C6N) DE0-nano board is shown in Table 10. Although the Virtex-5 FPGA differs from the Cyclone IV in terms of available resources, it is evident that by Tables 9 and 10 that a custom design consumes less resources from an FPGA leaving plenty of space for additional algorithms. Another advantage of a custom design over a softcore is a faster execution of the algorithm. As an example of this fact, Fig. 26 illustrates time values calculated for $T_0$.

**Remark 4.** Due to a lack of space, real time results are omitted, but it is worth mentioning that similar results were obtained to those in a VHDL modules custom design, reducing the advantages of the latter only to implementation issues.
5. Conclusion

The rotor velocity of AC synchronous machines show a direct relation with the the supply electrical frequency. Thanks to this relation, we have explicitly designed a frequency profile based on a desired rotor velocity for the startup of this class of motors. This is a difficult task for asynchronous machines since the rotor velocity depends on the load torque through the slip of the machine, hence requiring of a closed-loop control system. Although the proposed startup algorithm is simple, must be implemented along with the complex SVPWM algorithm, where an FPGA device was an appropriate solution for the implementation of both algorithms. With the custom design for the implementation of the complete algorithm in a Virtex-5® FPGA platform, the complete algorithm is executed faster, when compared with a softcore embedded in a FPGA, moreover, plenty of space if left in the FPGA that can be occupied for additional algorithms. For the performance demonstration of the startup algorithm implemented in a FPGA device, a PMM is used as a fan drive for the ventilation of a greenhouse. The startup of the motor was characterized by a smooth acceleration ramp in the rotor velocity, and smooth transient currents without large peak values. Some issues still remain as the integration of the control algorithm [2] that determines the optimal turn on time for the fan drive.

6. References


Appendix A. SVPWM Algorithm

Fig. A.27 shows a schematic diagram of the three-phase inverter connected to the stator of an electric motor.

![Schematic Diagram](image)

Figure A.27: Schematic diagram of a three-phase inverter.

In Fig. A.27, the DC source voltage, $V_{DC}$ is represented with two voltage sources of the same value just for analysis purposes. The transistors are idealized and represented with switches. Switching signals $sw_2$, $sw_4$ and $sw_6$ depend on $sw_1$, $sw_3$ and $sw_5$ respectively:

$\begin{align*}
sw_2 &= 1 - sw_1 \\
sw_4 &= 1 - sw_3 \\
sw_6 &= 1 - sw_5
\end{align*}$

(A.1)
with \( sw_i \in \{0,1\}, \ i = 1, \ldots, 6 \). The equations presented in (A.1) are such that the upper switches \( (sw_1, sw_3, sw_5) \) when set to off, then the lower switches \( (sw_2, sw_4, sw_6) \) are on, and vice versa. Now, fictitious voltages with respect to node “o” are determined as follows:

\[
\begin{align*}
V_{ao} &= V_{DC} \left( -\frac{1}{2} + sw_1 \right) \\
V_{bo} &= V_{DC} \left( -\frac{1}{2} + sw_3 \right) \\
V_{co} &= V_{DC} \left( -\frac{1}{2} + sw_5 \right),
\end{align*}
\]

the line to line voltages are:

\[
\begin{align*}
V_{ab} &= V_{ao} - V_{bo} = V_{DC} (sw_1 - sw_3) \\
V_{bc} &= V_{bo} - V_{co} = V_{DC} (sw_3 - sw_5) \\
V_{ca} &= V_{co} - V_{ao} = V_{DC} (sw_5 - sw_1), \quad (A.2)
\end{align*}
\]

meanwhile, phase voltages with respect to node “n” result as follows:

\[
\begin{align*}
V_{a} &= -\frac{1}{3} (V_{ca} - V_{ab}) \\
V_{b} &= -\frac{1}{3} (V_{ab} - V_{bc}) \\
V_{c} &= -\frac{1}{3} (V_{bc} - V_{ca}). \quad (A.3)
\end{align*}
\]

Substituting (A.2) in (A.3) yields to an explicit relation between phase voltages and the upper switch states:

\[
\begin{bmatrix}
V_{a} \\
V_{b} \\
V_{c}
\end{bmatrix}
= -\frac{1}{3} V_{DC} \begin{bmatrix}
-2 & 1 & 1 \\
1 & -2 & 1 \\
1 & 1 & -2
\end{bmatrix}
\begin{bmatrix}
sw_1 \\
sw_3 \\
sw_5
\end{bmatrix}. \quad (A.4)
\]

Table A.11 shows a logic switching sequence for generating sinusoidal signals at the inverter output.

<table>
<thead>
<tr>
<th>Step</th>
<th>( sw_1 )</th>
<th>( sw_3 )</th>
<th>( sw_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table A.11: Logic switching sequence for generating three–phase sinusoidal voltages.

**Remark 5.** The logic switching sequence can start at any step and must be cycled in order to reproduce three–phase voltages continuously over time.

Since the upper switch states \( (sw_1, sw_3, sw_5) \) can be represented by means of three bits, there are a total of eight possible steps. The lacking steps are step 0 \((0,0,0)\) and 7 \((1,1,1)\), and both represent a no output voltage state. As an example of use of equation (A.4) let us apply step 1

\[
\frac{1}{3} V_{DC} \begin{bmatrix}
1 \\
-2 \\
1
\end{bmatrix}
= -\frac{1}{3} V_{DC} \begin{bmatrix}
-2 & 1 & 1 \\
1 & -2 & 1 \\
1 & 1 & -2
\end{bmatrix}
\begin{bmatrix}
0 \\
1 \\
0
\end{bmatrix},
\]

that is, \( V_a = V_{DC}/3, V_b = -2V_{DC}/3 \) and \( V_c = V_{DC}/3 \).

When all steps of Table A.11 are applied then the graphical result of Fig. A.28 is obtained. It is clear that three–phase voltages in \((a, b, c)\) are produced in each step. These voltages can be transformed to \((\alpha, \beta)\) reference frame by means of Clark transformation. Then, the transformed voltages can be represented in rectangular or polar coordinates as shown in Table A.12. By plotting the \((\alpha, \beta)\) voltages in polar coordinates (note that these voltages have the same magnitude and phase shift between them is 60°), an hexagon is constructed as the one shown in Fig. A.29 where two close vectors form a sector.

The hexagon is used for determining the corresponding switching signals for a given pair of voltages in the \((\alpha, \beta)\) reference frame. The voltage components \( u_\alpha \) and \( u_\beta \) are located in the hexagon, then its magnitude \( u_m \) and angle \( \theta \) are determined by the use of
Table A.12: Stator voltages in natural variables \((a, b, c)\), and in the stator fixed reference frame \((\alpha, \beta)\) where rectangular and polar representations are presented when applying the logic sequence of Table A.11.

| Step | \(V_a\) | \(V_b\) | \(V_c\) | \(V_\alpha\) | \(V_\beta\) | \(|V|\) | \(\angle V\) |
|------|---------|---------|---------|-------------|-------------|--------|--------|
| 1    | \(\frac{1}{3}V_{DC}\) | \(-\frac{2}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(-\frac{1}{\sqrt{3}}V_{DC}\) | \(\frac{2}{3}V_{DC}\) | 0°     |
| 2    | \(\frac{1}{3}V_{DC}\) | \(-\frac{2}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(-\frac{1}{\sqrt{3}}V_{DC}\) | \(\frac{2}{3}V_{DC}\) | 60°    |
| 3    | \(\frac{1}{3}V_{DC}\) | \(-\frac{2}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(-\frac{1}{\sqrt{3}}V_{DC}\) | \(\frac{2}{3}V_{DC}\) | 120°   |
| 4    | \(\frac{1}{3}V_{DC}\) | \(-\frac{2}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(-\frac{1}{\sqrt{3}}V_{DC}\) | \(\frac{2}{3}V_{DC}\) | 180°   |
| 5    | \(\frac{1}{3}V_{DC}\) | \(-\frac{2}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(-\frac{1}{\sqrt{3}}V_{DC}\) | \(\frac{2}{3}V_{DC}\) | 240°   |
| 6    | \(\frac{1}{3}V_{DC}\) | \(-\frac{2}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(\frac{1}{3}V_{DC}\) | \(-\frac{1}{\sqrt{3}}V_{DC}\) | \(\frac{2}{3}V_{DC}\) | 300°   |

Fig. A.30 and the following formulas:

\[
\begin{align*}
\mu_m &= \sqrt{u_\alpha^2 + u_\beta^2} \\
\theta &= \arctan \left( \frac{u_\beta}{u_\alpha} \right).
\end{align*}
\number{A.5}

Now we will suppose that the magnitude of the control components \(u_m\) is inside of an arbitrary Sector \(SEC\) \((SEC = 1, 2, 3, 4, 5, 6)\), which is between steps \(S_S\) and \(S_{S+1}\) (with \(S_7 = S_1\), in this case \(S_7\) must not be confused with the state represented with \((1, 1, 1)\) that resides at the origin of the hexagon), then, the projection of \(u_m\) on \(S_S\) and \(S_{S+1}\) will be denoted by \(u_1\) and \(u_2\) respectively; that along with the magnitude \(u\) form the triangle shown in Fig. A.31.

The angle \(\theta_S\) in Fig. A.31 corresponds with the angle inside sector \(SEC\) and is calculated as \(\theta_S = (1 - SEC)60^\circ + \theta\). Now, applying the law of sines in Fig. A.31 one can find \(u_1\) and \(u_2\) of the following form:

\[
\begin{align*}
\frac{\sin (120^\circ)}{u_m} &= \frac{\sin (60^\circ - \theta_S)}{u_1} \iff u_1 &= \frac{2u}{\sqrt{3}} \sin (60^\circ - \theta_S) \\
\frac{\sin (120^\circ)}{u_m} &= \frac{\sin (\theta_S)}{u_2} \iff u_2 &= \frac{2u}{\sqrt{3}} \sin (\theta_S).
\end{align*}
\number{A.6}
\number{A.7}

The values for \(u_1\) and \(u_2\) are proportional to time values \(T_1\) and \(T_2\) respectively; values that the inverter must be on the state \(S_S\) and \(S_{S+1}\) respectively again:

\[
\frac{u_1}{|S_S|} = \frac{T_1}{T}, \quad \frac{u_2}{|S_{S+1}|} = \frac{T_2}{T}
\]

where \(|S_S| = |S_{S+1}| = (2/3)V_{DC}\) and \(T\) is the sampling period. Therefore one determines \(T_1\) and \(T_2\) of the following form:

\[
\begin{align*}
T_1 &= \tau \sin (60^\circ - \theta_S) = \tau \sin (SEC60^\circ - \theta) \\
T_2 &= \tau \sin (\theta_S) = \tau \sin ((1 - SEC)60^\circ + \theta)
\end{align*}
\number{A.6}
\number{A.7}
where \( \tau = \sqrt{3}T_{um}/V_{DC} \). The sampling period is fixed as sampling period of the three–phase inverter \( T_{PWM} \), i.e., \( T = T_{PWM} \). If \( T_1 + T_2 < T \) then the period \( T \) is complemented with dead time intervals denoted by \( T_0 \) that corresponds to the steps \( S_0 \) or \( S_7 \), therefore

\[
T_0 = T - T_1 - T_2. \tag{A.8}
\]

Up to this point, the determination of active vectors and their corresponding time for being active is the same for several SVPWM strategies as the 3-segment, 5-segment and 7-segment algorithms. As already mentioned, the 7-segment algorithm yields to a good performance of the inverter in terms of harmonic distortion [28]. These strategies differ from each other by the choice of the null vector, that in particular for the 7-segment algorithm the null vector is alternated in each cycle in which the sequence is reversed after each null vector.

Appendix B. Implementation method with a softcore

The NIOS II microprocessor is a general purpose softcore based on a RISC architecture developed by ALTERA. The NIOS II microprocessor consists of the NIOS II core, a set of peripherals, internal memory, and interfaces for external memory, all of this in a single chip. Fig. B.32 shows a block diagram of the NIOS II architecture.

The main characteristics of the NIOS II processor are:

- 32 bits instruction set and address
- 32 general purpose registers
- Support for 32 external interrupts

The employed platform is a DE0-nano Cyclone® IV (EP4CE-22F17C6N) FPGA, and for software designing, a Quartus® II is used since it includes the Qsys design tool that allows to build the NIOS II microprocessor with required resources. For programming of the algorithm, the NIOS II Software Build Tools for Eclipse are used for editing, building and reviewing the programming code in C/C++. Fig. B.33 shows a flow diagram for the implementation of the softcore in the Altera® FPGA platform.

Due to the sequential nature of the proposed algorithm, it is programmed in C, where its corresponding flow diagram is shown in Fig. B.34. Constant \( k_{ramp} \) represents the number of samples for making the ramp frequency. When \( k > k_{ramp} \) the frequency remains constant. For both cases, the SVPWM subroutine consists of the five basic steps outlined in Section 3.
Implementation

C/C++ Libraries for peripheral controllers and ALTERA hardware layer

Development tool program (NIOS II Software Build Tools for Eclipse)

NIOS II peripherals, instructions and custom logic

Define and generate Qsys system

Add Qsys system in Quartus II project

Pin assignment, required time values and other constants

Build designed hardware for the platform

Download FPGA design to the platform

C/++ Libraries for peripheral controllers and ALTERA hardware layer

Development tool program (NIOS II Software Build Tools for Eclipse)

NIOS II peripherals, instructions and custom logic

Define and generate Qsys system

Add Qsys system in Quartus II project

Pin assignment, required time values and other constants

Build designed hardware for the platform

Download FPGA design to the platform

Figure B.33: Flow diagram for implementation of the embedded microprocessor in C/C++.