FPGA-Based Space Vector PWM with Artificial Neural Networks

J. Osorio[1], P. Ponce[2] and A. Molina[3]
[1]Research Department Tecnologico de Monterrey-ITESM, Mexico D.F., Mexico
[2]Director of Master in Science, Tecnologico de Monterrey-ITESM, Mexico D.F., Mexico
[3]Vice Chancellor for Research, Tecnologico de Monterrey-ITESM, Mexico D.F., Mexico
Phone (00521) 5483-2020 Ext. 2782 E-mail: joycer.damian@gmail.com, pedro.ponce@itesm.mx, armolina@itesm.mx

Abstract — **This article presents the improvement of a PWM technique, called Space Vector PWM (SVPWM), using an Artificial Neural Network (ANN) to minimize the mathematic complexity involved with the SVPWM. The latter is a pulse-width modulation technique that is wide implemented to control AC electric motors. The results obtained from this research work will be used for further implementation of artificial intelligence techniques to control electric vehicle powertrains. Matlab is implemented for the ANN design and Labview for the FPGA programming and implementation.**

Keywords — **SVPWM, ANN, Artificial Intelligence, AC Motors, FPGA.**

I. Introduction

Pulse-width modulation techniques are widely used to control DC and AC motors. From the latter the asynchronous motors, also known as induction motors, have an important place in industry applications. The first prototype was developed and presented by Nikola Tesla in 1882 for which he received the patent in 1888 [1]. The induction motor is basically conformed by the rotor, that can be squirrel cage or slip ring, and the stator, where the inductor windings are located.

The induction motor operation is based on rotatory magnetic fields interaction, that is, an AC current feeds the stator windings generating a rotatory magnetic field in the stator which induces currents in the rotor windings, hence it is generated a rotatory magnetic field in the rotor that spins at a slower speed than the stator magnetic field. Thus, there is a relative speed between stator and rotor field, called slip speed. Therefore, to control an induction motor must be controlled the AC energy source frequency and amplitude in order to define the motor speed and torque. This is possible thanks to PWM techniques that allow controlling both the current frequency and amplitude.

In this research work, it is presented a SVPWM which is improved by means of ANN. Nevertheless, several research works had been proposed regarding PWM techniques, thus it is interesting to analyze those works related to this topic.

Hong Hee Lee et al [2] proposed a SVPWM with different sub-nets, one for overmodulation I, another sub-net for overmodulation II (zones out of the linear modulation), and another sub-net was implemented for the linear modulation mode. The ANNs designed in [2] were implemented in a FPGA of Xilinx with acceptable results.

Other interesting research works related to SVPWM were proposed by Pongiannan and Yadaiah [3], and Tzou et al [4]. In [3] the authors proposed a SVPWM implemented in a FPGA in order to improve execution speed and relieve the controller from the time consuming computational task of PWM signal generation. In [4] was implemented a SVPWM in two RAM based FPGA XC4003 and XC4010, and to this was incorporated a DSP in order to provide a solution for high-performance ac-drivers. Several research works related to the area of control and power electronics [5–7] demonstrate how researches are working on improving PWM techniques and implementing better hardware tools.

In this paper is designed one ANN to performance the switching time process involved in SVPWM, this will be explained in detail later. Previous to the design of the ANN, it was developed a SVPWM based on [8], [9] in order to adapt the calculation to the FPGA.

II. Space Vector PWM

The SVPWM is a technique implemented to control electric motors; specifically this technique used the inverter (i.e. the component that sends the current signals to the motor) as a voltage source (VSI).

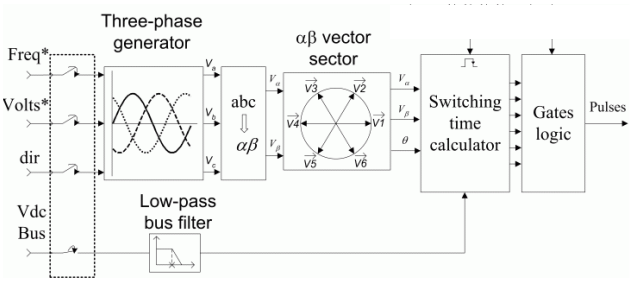

Fig. 1. SVPWM operation

Fig. 1 shows how the SVPWM works and the way the operations are carried out. It is possible to note that the main inputs are: reference voltage (*Volts* or* V_{ref}), frequency (f_{ref}) and DC bus voltage (V_{dc}). Then, with the frequency and amplitude (V_{ref}) three voltage signals with 120° phase between each other are generated, these are Va, Vb and Vc.

After, the Park's transform (represented in Fig. 1 as abc->αβ) is executed in order to obtain the voltage signals in a orthogonal framework. The Park's transform is achieved with:

$$\begin{bmatrix} V_\alpha \\ V_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}$$

With the voltage in an orthogonal framework is possible to identify the components of the reference voltage into the following sectors.

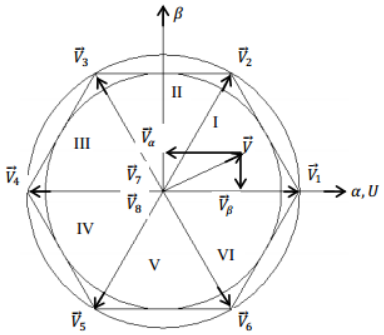

Fig. 2. Inverter states representation

Fig. 2 depicts the inverter states that are defined in Table 1 with their values, for each row it corresponds an inverter state.

Table 1
Inverter state

a	b	c	v_a	v_b	v_c	v_{ab}	v_{bc}	v_{ca}
0	0	0	0	0	0	0	0	0
1	0	0	2/3	−1/3	−1/3	1	0	−1
1	1	0	1/3	1/3	−2/3	0	1	−1
0	1	0	−1/3	2/3	−1/3	−1	1	0
0	1	1	−2/3	1/3	1/3	−1	0	1
0	0	1	−1/3	−1/3	2/3	0	−1	1
1	0	1	1/3	−2/3	1/3	1	−1	0
1	1	1	0	0	0	0	0	0

The SVPWM is the control logic of the inverter, that is, it defines the on or off states (i.e. 1 or 0 digital signals) for the transistors inverter and the times for each state. Therefore, the SVPWM calculates the switching time with:

$$[T_1\ T_2]^T = T_{pwm}[V_x\ \ V_{x\pm60}]^{-1} V_{ref}$$

Where V_{ref} is the reference voltage and $[V_x\ \ V_{x\pm60}]$ corresponds to the adjacent voltage vectors, this is possible to note from Fig. 2 where each vector is spaced 60° (e.g. V_1 and V_2). The time T_{pwm} defines the duty cycle of the SVPWM, that means the time in which is executed each sector of the SVPWM. However, T_1 and T_2 are the times for the active vectors (V_1 to V_6), but from Fig. 2 is possible to note that V_7 and V_8 are in the origin, and for these vectors there is a component of time that defines the symmetry of the pulses, this is:

$$T_0 = T_{pwm} - T_1 - T_2$$

Fig. 3 presents the time for each inverter state in sector 1 (i.e. the sector between 0° and 60°). This is the type of output obtained in the block called Gate logic in Fig. 1

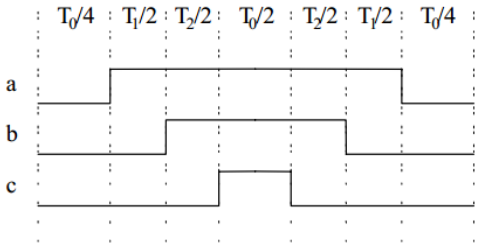

Fig. 3. Times and pulses for each phase

However, the calculations presented are for linear modulation mode, that is, modulation under the circle defined within the hexagon of Fig. 2. There are two other modulation modes that may occur during operation:

- Linear modulation: $\frac{|V_{ref}|}{V_{dc}} \leq \frac{1}{\sqrt{3}}$, within the hexagon.
- Overmodulation mode I: $\frac{1}{\sqrt{3}} < \frac{|V_{ref}|}{V_{dc}} \leq \frac{2}{3}$, in boundary of the circle.
- Overmodulation mode II: $\frac{2}{3} < \frac{|V_{ref}|}{V_{dc}}$, beyond the circle.

The parameter $\frac{|V_{ref}|}{V_{dc}}$ is called modulation index. For these operation modes, there are some methods that can be applied in order to maintain the correct SVPWM operation. In this research work it is used the method proposed in [9]. Therefore, the following voltage components are computed first:

$$\begin{cases} V_\alpha^* = \frac{|\vec{V}_\alpha|}{\frac{2}{3}V_{DC}} \\ V_\beta^* = \frac{|\vec{V}_\beta|}{\frac{2}{3}V_{DC}} \end{cases}$$

Following the calculation proposed in [9] Table 2 and Table 3 depict the time calculation for linear modulation and overmodulation modes respectively.

III. SVPWM and ANN-SVPWM Development

In section II all the calculation involved with the SVPWM was defined, it was also explained the modulation modes. Now, in this section the implementation of the SVPWM in the FPGA is presented, and then it will be

explained how the ANN-SVPWM is designed and implemented.

Table 2.
Switching times in linear modulation

Sector 1 and IV	
T_{U_off}	$\frac{T_s}{4}\left(1-V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{U_on}	$\frac{T_s}{4}\left(3+V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{V_off}	$\frac{T_s}{2}\left(1+V_\alpha^*-\sqrt{3}V_\beta^*\right)$
T_{V_on}	$\frac{T_s}{4}\left(3-V_\alpha^*+\sqrt{3}V_\beta^*\right)$
T_{W_off}	$\frac{T_s}{4}\left(1+V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{W_on}	$\frac{T_s}{4}\left(3-V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
Sector II and V	
T_{U_off}	$\frac{T_s}{4}\left(1-2V_\alpha^*\right)$
T_{U_on}	$\frac{T_s}{4}\left(3+2V_\alpha^*\right)$
T_{V_off}	$\frac{T_s}{4}\left(1-\frac{2V_\beta^*}{\sqrt{3}}\right)$
T_{V_on}	$\frac{T_s}{4}\left(3+\frac{2V_\beta^*}{\sqrt{3}}\right)$
T_{W_off}	$\frac{T_s}{4}\left(1+\frac{2V_\beta^*}{\sqrt{3}}\right)$
T_{W_on}	$\frac{T_s}{4}\left(3-\frac{2V_\beta^*}{\sqrt{3}}\right)$
Sector III and VI	
T_{U_off}	$\frac{T_s}{4}\left(1-V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{U_on}	$\frac{T_s}{4}\left(3+V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{V_off}	$\frac{T_s}{4}\left(1+V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{V_on}	$\frac{T_s}{4}\left(3-V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{W_off}	$\frac{T_s}{4}\left(1+V_\alpha^*+\sqrt{3}V_\beta^*\right)$
T_{W_on}	$\frac{T_s}{4}\left(3-V_\alpha^*-\sqrt{3}V_\beta^*\right)$

A. SVPWM

The SVPWM is implemented with Labview in the FPGA, the implementation is presented in Fig. 4.

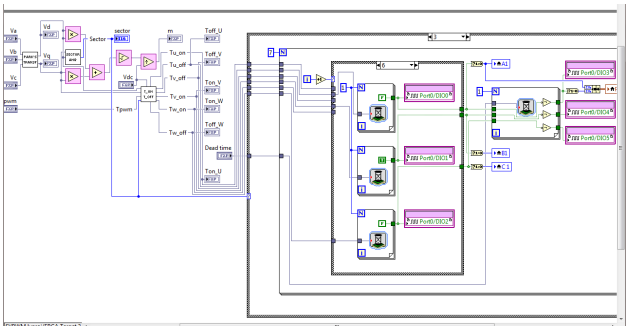

Fig. 4. SVPWM in the Labview FPGA

In the figure above the design in Labview is depicted for the FPGA of SVPWM and gives an idea of the programming environment Labview offers. The SVPWM execution steps are:

1. Define V_{ref}, f_{ref}, V_{dc}
2. Define Va, Vb and Vc,
3. Execute the Park's transform
4. Define the angle of the reference voltage vector in a α-β framework
5. Identify if there is overmodulation or not
6. Calculate switching times

With the SVPWM programed in the FPGA, samples of V_{ref}, f_{ref} and V_{DC} are gathered as the inputs for the ANN, the ANN target is the switching times.

Table 3.
Switching time for overmodulation I and II

Sector 1	
T_{U_off}	0
T_{U_on}	T_s
T_{V_off}	$\frac{T_s}{2}\left(1-\frac{2}{\sqrt{3}}V_\beta^*\right)$
T_{V_on}	$\frac{T_s}{2}\left(1+\frac{2}{\sqrt{3}}V_\beta^*\right)$
T_{W_off}	T_s
T_{W_on}	0
Sector II	
T_{U_off}	$\frac{T_s}{2}\left(1-V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{U_on}	$\frac{T_s}{4}\left(1+V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{V_off}	0
T_{V_on}	T_s
T_{W_off}	T_s
T_{W_on}	0
Sector III	
T_{U_off}	T_s
T_{U_on}	0
T_{V_off}	0
T_{V_on}	T_s
T_{W_off}	$\frac{T_s}{2}\left(1+V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{W_on}	$\frac{T_s}{2}\left(1-V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
Sector IV	
T_{U_off}	T_s
T_{U_on}	0
T_{V_off}	$\frac{T_s}{2}\left(1+V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{V_on}	$\frac{T_s}{2}\left(1-V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{W_off}	0
T_{W_on}	T_s
Sector V	
T_{U_off}	$\frac{T_s}{2}\left(1-V_\alpha^*+\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{U_on}	$\frac{T_s}{2}\left(1+V_\alpha^*-\frac{V_\beta^*}{\sqrt{3}}\right)$
T_{V_off}	T_s
T_{V_on}	0
T_{W_off}	0
T_{W_on}	T_s
Sector VI	
T_{U_off}	0
T_{U_on}	T_s
T_{V_off}	T_s
T_{V_on}	0

T_{W_off}	$\frac{T_s}{2}\left(1+\frac{2}{\sqrt{3}}V_\beta^*\right)$
T_{W_on}	$\frac{T_s}{2}\left(1-\frac{2}{\sqrt{3}}V_\beta^*\right)$

B. ANN-SVPWM

Before designing the ANN-SVPWM, it is important to understand how an ANN works.

ANNs work as a biological neural network, that is, they simulate the learning process based on a training process. The form of an artificial neuron is depicted in Fig. 5

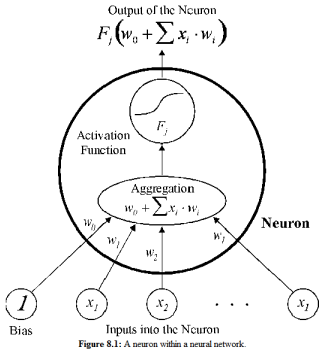

Fig. 5. Artificial neuron

Each neuron has an activation function that evaluates the summatory of inputs multiplied by its respective weight, and the result of this process is the output of each neuron. For a net the configuration has the form depicted in Fig. 6

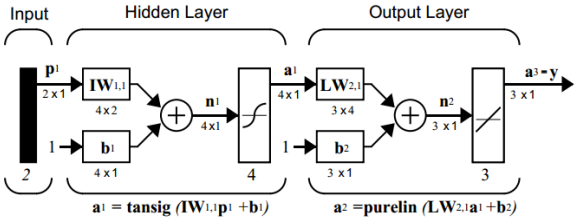

Fig. 6. ANN structure

In Fig. 6 a feedforward network structure is presented, which is the same network type that will be used for the ANN-SVPWM. In the example structure depicted in Fig. 6, the hidden layer has a tansig activation function, and the output pureline activation function with 4 and 3 neurons respectively.

An important aspect of an ANN is the structure selection and configuration, since there are different training processes that can be used in order to design an ANN. Therefore, depending on the application and complexity of the process to control or simulate, an ANN structure is selected, as well as the activation functions and learning process.

As it was mentioned in this research work the inputs and targets for the ANN will be:

$$\text{Input}=[V_{ref}, f_{ref}, V_{dc}]$$

Target=[Toff_U, Ton_U, Toff_V, Ton_V, Toff_W, Ton_W]

The target is the switching time for each phase, depicted in Table 2 and Table 3. For the design of the ANN, it was implemented Matlab and its toolbox "nntool"[10].

The structure proposed for the ANN is depicted in Fig. 7 with 4 inputs, 20 hidden neurons and 6 output neurons. This ANN was trained with the Levenberg-Marquardt method. This method was selected because it is a quasi-Newton algorithm that optimizes the training time, since it does not require the calculation of the second derivative of the Hessian matrix [11]. Also, this training method was selected since excellent results are obtained for high nonlinear and least square problems.

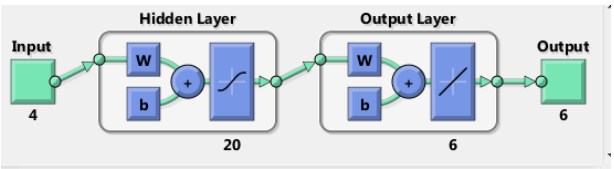

Fig. 7. ANN proposed

The performance and fitting are depicted in Fig. 8 and Fig. 9. These figures represent the error achieved during the training process and the way it fits the ANN output with the target. An ANN must have a good fitting with the target values but the outputs cannot be equal to the targets; in other words, an ANN cannot be designed to achieve an error equal to zero, because an ANN needs a tolerance to achieve generalization in order to obtain good results with different input values.

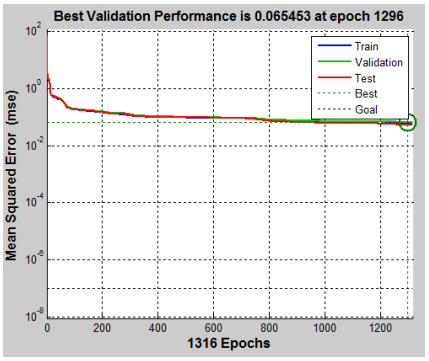

Fig. 8. ANN performance

The ANN designed is now implemented in the FPGA with Labview.

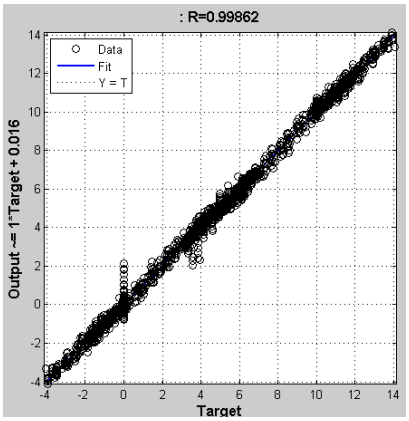

Fig. 9. ANN fitting

Next is presented some results obtained with Labview for the SVPWM and ANN-SVPWM

IV. Results

The SVPWM and ANN-SVPWM were tested with a V_{ref} of 400 and 300 V to a f_{ref} of 70Hz..

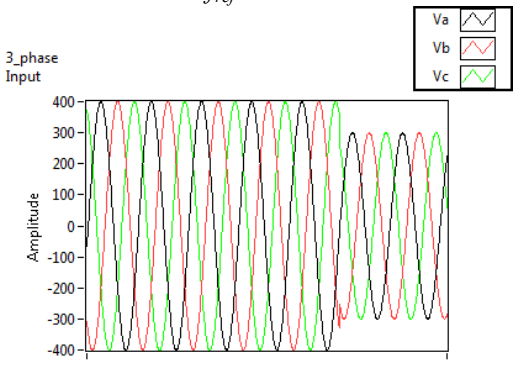

Fig. 10. Reference voltage for the SVPWM

With the V_{ref} defined it is obtained the modulation index depicted in Fig. 11. Based on the definitions of linear and overmodulation modes is possible to note that the modulation index obtained in Fig. 11 corresponds to overmodulation mode II.

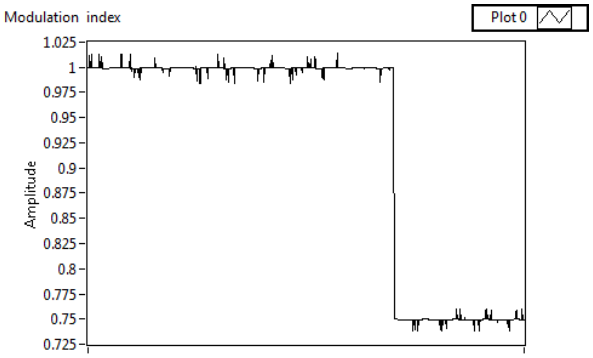

Fig. 11. Modulation index for the SVPWM

The SVPWM output is depicted in Fig. 12

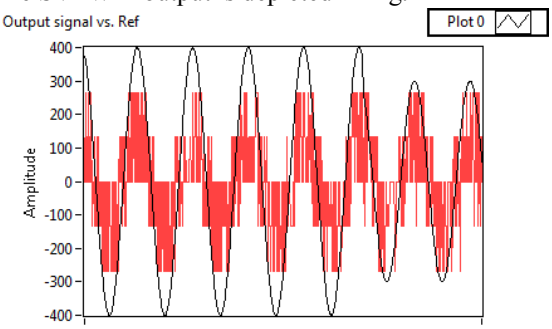

Fig. 12. Pulses with the SVPWM

The inputs used for the SVPWM are now used for the ANN-SVPWM.

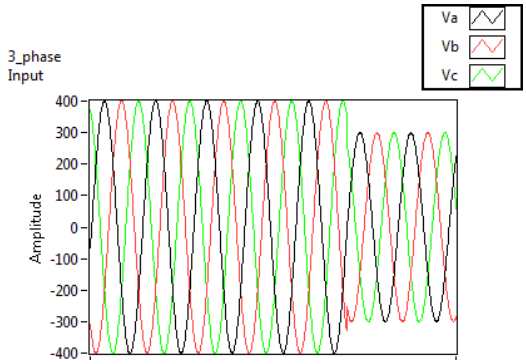

Fig. 13. Reference voltaje for the ANN-SVPWM

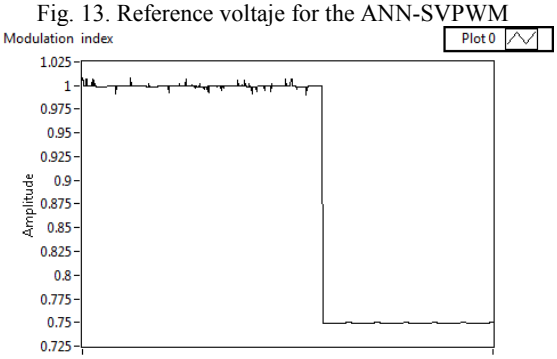

Fig. 14. Modulation index for the ANN-SVPWM

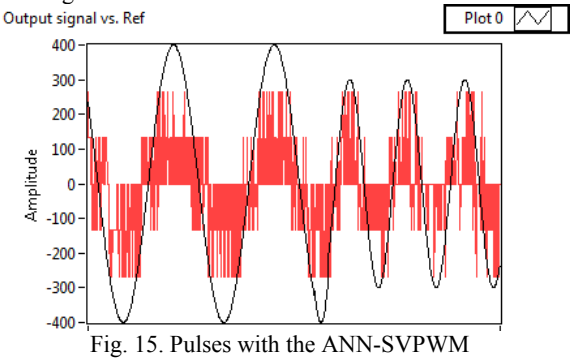

Fig. 15. Pulses with the ANN-SVPWM

Fig. 14 and Fig. 15 depict the results obtained with the ANN-SVPWM. It is possible to note from Fig. 12 and Fig. 15 that with overmodulation II the outputs obtained

maintain the voltage amplitude and current forms. It is important to remark that the outputs presented are for the top U phase, and the dead time between top and bottom part of one leg of the inverter is the 1 ms. The latter is quite important in order to avoid a short circuit.

Another interesting comparison is possible to obtain by analyzing the result obtained in Fig. 15 and comparing it with the analysis established in [12], since both result are quite similar for overmodulation mode operation.

V. Conclusion

In this research work a SVPWM and ANN-SVPWM were implemented with Labview in a FPGA. The mathematic complexity involved with the SVPWM could be reduced significantly with the design of the ANN-SVPWM, because from the sixth step defined for the SVPWM, it was minimized to one ANN trained to receive inputs and to calculate switching times, just by means of simple multiplications and additions. This can be synthesized as three steps for the ANN-SVPWM, those are:

1. Define V_{ref}, f_{ref}, V_{dc}
2. Identify if there is overmodulation or not
3. Calculate switching times

The tools used to achieve this work were Matlab and Labview, which are powerful tools for control design. This work presented all the calculations implicated with a SVPWM and ANN-SVPWM, and how it can be adapted to work with FPGA. The design of an ANN is an iterative process, and based on the application and non-linearity implicated it is selected the ANN structure. Therefore, from the results obtained from the ANN-SVPWM is possible to note that a good ANN design was achieved.

Future works will be implementing these results in the simulation environment of ADVISOR and real implementation with electric vehicles powertrain.

References

[1] N. Tesla, "U.S. Patent 381,968 Nikola Tesla," 1882.

[2] N. T. Lee, Hong Hee; Dzung, Phan Quoc; Phuong, Le Minh; Khoa, Le Dinh; Dan Vu, "The Space Vector PWM for Voltage Source Inverters Using," in *International Forum on Strategic Technology*, 2010, pp. 1-6.

[3] R. K. Pongiannan and N. Yadaiah, "FPGA based Space Vector PWM Control IC for Three Phase Induction Motor Drive," in *IEEE ICIT*, 2006, pp. 2061-2066.

[4] Y.-yu Tzou, H.-jean Hsu, and T.-sung Kuo, "FPGA-Based SVPWM Control IC for 3-Phase PWM Inverters," in *IEEE IECON*, 1996, pp. 138-143.

[5] J. Holtz, W. Lotzkat, and A. M. Khambadkone, "On Continuous Control of PWM Inverters in the Overmodulation Range Including the Six-Step Mode," *IEEE Transactions on Power Electronics*, vol. 8, no. 4, pp. 546-553, 1993.

[6] S. Bolognani and M. Zigliotto, "Novel Digital Continuous Control of SVM Inverters in the Overmodulation Range," *IEEE Transactions on Industry Applications*, vol. 33, no. 2, pp. 525-530, 1997.

[7] H. Hu, W. Yao, Z. Lu, and S. Member, "Design and Implementation of Three-Level Space Vector PWM IP Core for FPGAs," *IEEE Transactions on Power Electronics*, vol. 22, no. 6, pp. 2234-2244, 2007.

[8] Z. Zhou and T. Li, "Design of a Universal Space Vector PWM Controller Based on FPGA," in *APEC*, 2004, vol. 00, no. C, pp. 1698-1702.

[9] Y. Wang and U. Schaefer, "Real Time Simulation of a FPGA Based Space Vector PWM Controller," in *SPEEDAM*, 2010, no. 1, pp. 833-838.

[10] H. Demuth, *Neural Network Toolbox ™ 6 User ' s Guide*. 2009.

[11] H. Gavin, "The Levenberg-Marquardt method fornonlinear least squares curve-fitting problems." pp. 1-15, 2011.

[12] J. O. P. Pinto, S. Member, B. K. Bose, and L. Fellow, "A Neural-Network-Based Space-Vector PWM Controller for Voltage-Fed Inverter Induction Motor Drive," *IEEE Transactions on Industry Applications*, vol. 36, no. 6, pp. 1628-1636, 2000.