International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 7, July - 2013

VHDL Implementation Of PWM Technique For Generation Of Switching Pulses

Veena Walimbe — PG Student
N. R. Bhasme — Associate Professor

Department of Electrical Engineering, Government College of Engineering, Aurangabad.

Abstract:
The design and implementation of a Variable-Voltage Variable-Frequency (VVVF) Controller based on Sinusoidal Pulse Width Modulation (SPWM) Technique for a 1 and / or 3 Phase Induction Motor using VHDL. Variable-Voltage Variable-Frequency (VVVF) technique is used extensively in the industry as it provides the accuracy required at minimal cost. Voltage/ frequency (v/f) controlled motors fall under the category of Variable Voltage Variable Frequency (VVVF) drives. To maintain maximum torque for a given working condition, the flux in the machine must be maintained constant. The ratio of Voltage to frequency must be held constant. For Variable Voltage Variabe Frequency (VVVF) drives, there is a need to control the fundamental voltage of the inverter if its frequency (and therefore the frequency of the induction motor), need to be varied. To vary the fundamental component of the inverter, the Modulation Index of the carrier signal has to be changed. The speed at rated supply frequency is normally used as the base speed. At frequencies below the base speed, the supply magnitude needs to be reduced so as to maintain a constant Volt/Hertz. The VHDL based controller is used to generate SPWM pulses based on the frequency input, that are used to control the inverter. The VVVF output of the inverter can be used as supply to a three phase induction motor and thereby speed of the motor can be controlled.

Keywords: PWM Technique, Speed Control

I. Introduction

The Motor Control industry is a strong aggressive sector. Each industry to remain competitive, must reduce costs but also has to answer to power consumption reduction and EMI radiation reduction issues imposed by governments and power plant lobbies. The results of these constraining factors are the need of enhanced algorithms. PSoc technology allows achieving both, a high level of performance as well as a system cost reduction. The AC induction motor is the workhorse of industrial and residential motor applications due to its simple construction and durability. These motors have no brushes to wear out or magnets to add to the cost. The rotor assembly is a simple steel cage. ACIM's are designed to operate at a constant input voltage and frequency; we can effectively control an ACIM in an close loop speed application if the frequency of the motor input voltage is varied. If the motor is not mechanically overloaded, the motor will operate at a speed that is roughly proportional to the input frequency. As you decrease the frequency of the drive voltage, you also need to decrease the amplitude by a proportional amount. Otherwise, the motor will consume excessive current at low input frequencies. This control method is called "Volts-Hertz control". In practice, a custom Volts-Hertz profile is developed that ensures the motor operates correctly at any speed setting. This profile can take the form of a look-up table or can be calculated during run time. Often, a slope variable is used in the application that defines a linear relationship between drive frequency and voltage at any operating point. The Volts-Hertz control method can be used in conjunction with speed and current sensors to operate the motor in a closed loop fashion. The part is targeted toward applications in both industrial and home appliance industries, such as washing machines, compressors, air conditioning units, pumps & industrial drives. Controlling the speed of induction motors has ever since been an important topic of research. The control methodologies have evolved from electromechanical switching to high speed digital controllers using DSP and FPGA [1]. Of late, Pulse-Width Modulation techniques have been the subject of intensive research; as PWM controlled power electronic devices find increasing applications in many new industrial processes involving more stringent performance specifications [2]. This is particularly true in case of high performance drive systems, uninterruptible power supply and programmable AC power sources. Since PWM inverters play an important role in each of these applications, the whole system is dependent on the algorithm controlling the PWM inverter [3]. In recent years, Field Programmable Gate Arrays have drawn much attention due to its short design cycle, low cost and high flexibility in terms of programmability. The Field Programmable Gate Arrays (FPGAs) offer significant advantages over microprocessors and DSPs for high performance, low volume applications, particularly for applications that can exploit customized bit-widths and massive instruction-level parallelism. The innovative development of FPGAs whose configuration could be re-programmed an unlimited number of times spurred the invention of a new field in which many different hardware algorithms could execute, in turn, on single device, just as many different software algorithms can run on a conventional processor [4]. When comparing the dynamic performance, control capabilities and concurrency in PWM controlled Power Converters, FPGA based digital techniques are better than DSPs [4] [5]. The FPGA controller produces the SPWM pulses which are at a voltage of 3.3V. The voltage shifting from 3.3V to 12V requires a voltage-level shifting circuit. The voltage level shifted pulses will be fed to the three phase inverter to convert the DC supply to a three phase supply which is in turn fed to the induction motor. The variation in the duty cycle and the number of SPWM pulses determines the amplitude and frequency of the inverter. This can be achieved by varying either the amplitude or frequency of the carrier signal (triangular wave).

II. PWM Technique

A model for the controller was designed using VHDL. The VHDL code developed to generate a three phase sinusoidal pulse width modulated signal is divided into seven entities, namely: Interface Module, Oscillator, Addition of 120^0, Amplitude Module, PWM Module, Top Level Module and Clock Divider Module. The inputs to the program as a whole are an 8 bit multiplexed data signal, a 2 bit selection signal and an enable signal. The outputs are three pulse width modulated sinusoidal signals; each signal being phase shifted 120^0 with respect to the previous signal in a cyclic manner. The block diagram of the system is shown is shown in figure 1.

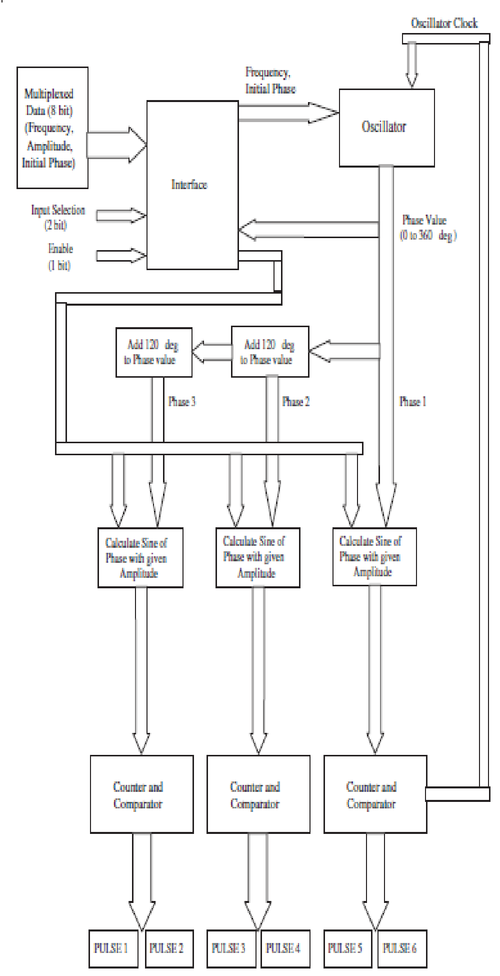
Oscillator Clock
Multiplexed Data (8 bit) (Frequency, Amplitude, Initial Phase)
Frequency, Initial Phase
Oscillator
Interface
Input Selection (2 bit)
Enable (1 bit)
Phase Value (0 to 360 deg)
Add 120 deg to Phase value
Add 120 deg to Phase value
Phase 3
Phase 2
Phase 1
Calculate Sine of Phase with given Amplitude
Calculate Sine of Phase with given Amplitude
Calculate Sine of Phase with given Amplitude
Counter and Comparator
Counter and Comparator
Counter and Comparator
PULSE 1
PULSE 2
PULSE 3
PULSE 4
PULSE 5
PULSE 6

Figure 1: Block Diagram

A. Interface module

The purpose of this entity is to interface the FPGA with the inputs. The input to the FPGA is an 8 bit multiplexed data signal, a 2 bit selection signal and an enable signal. The data signal is demultiplexed with the help of the selection signal when the enable signal is high. The following rules are used to demultiplex the data signal: If the selection signal is ’00’, the data lines represent amplitude. If the selection signal is ’01’, the data lines represent frequency. If the selection signal is ’10’, the data lines represent the 8 lower bits of the phase. If the selection signal is ’11’, the data lines represent the 2 higher bits of the phase. Clearly, signals ’10’ and ’11’ can be used to set an initial phase value. If the enable signal is low, the frequency and amplitude values remain the same until enable is made high again. The entity has another input, a 10 bit *phasein*. When enable is low, the phase output is the same as *phasein*. Thus, the outputs of this entity are: Frequency (8 bits), Amplitude (8 bits) and Phase (10 bits).

B. Amplitude lookup table

This entity is used to calculate the value of modulation index from the value of frequency input to the FPGA. It generates a Modulation Index of 1 for frequencies of 50 Hz and above. For frequencies below 50 Hz, it varies linearly. The modulation index is a value between 0 and 1, and hence can be represented by cos(φ). This is done to reduce the FPGA resources used by changing a multiplication to an addition. This is described in detail in the Amplitude Module. The output of the look-up table is an 8 bit value which represents 'φ'.

C. Oscillator

This entity is used to generate phase values increasing from 0◦ to 360◦, and then reset back to zero, resulting in a shape resembling a sawtooth with a minimum value of 0◦ and maximum value of 360◦. The entity takes the frequency, initial phase and a clock as inputs. The whole entity is basically a D-flip flop. The value of phase is initialised to a given value using the data signal and selection signals '10' and '11', or to zero by default. The Phase value here is 10 bits wide with '0000000000' representing 0◦ and '1111111111' representing 360◦. The 8 bit frequency value determines the phase increment that is the value of phase is incremented by the frequency value every clock cycle. When the phase value reaches 360◦, it is reset to zero, thus effectively producing a sawtooth. Clearly, if the value of frequency input is larger, the phase value reaches 360◦ faster and thus produces a saw tooth of higher frequency. Note that the adder here adds an eight bit frequency value to a 10 bit phase value. Thus, the output of this entity is a ten bit phase value which increases and decreases in the shape of a sawtooth. It has to be noted that the clock signal for the oscillator should have a frequency 1/512 times that of the frequency of clock signal used in the counter-comparator, since the counter counts from 0 to 511 for each pulse of the PWM output.

D. Addition of 120◦

This entity takes a ten bit phase value as input, increments it by 120◦ (i.e. '0101010101'). The output is also a ten bit phase value. Thus when a phase value, say $P1$, is incremented by 120◦ once to get $P2$ and then $P2$ is incremented by 120◦ to get $P3$; three values of phase are obtained that can be arranged in a cyclic manner resembling that of a three phase system.

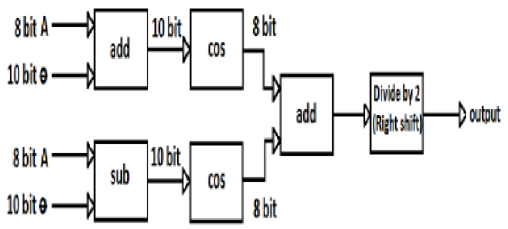

Figure 2: Amplitude Module

E. Amplitude module

This entity is used to calculate the value of the sine of the phase taking into account the value of the amplitude input. It takes as inputs 10 bit phase, 8 bit amplitude and a clock signal, and gives an 8 bit value as output. The value of Acos(θ), should be calculated where 'A' is the amplitude and 'θ' is the phase. The value of the cosine can be computed using a cosine lookup table. However, using a multiplier to multiply the value of amplitude would use too much of the FPGA's resources. Thus an alternate method is suggested. The amplitude is coded as an angle from zero to 90◦, where '00000000' represents zero and '11111111' represents 90◦. The cosine of this angle can be used to represent amplitude. Thus cos(φ)*cos(θ) has to be computed, where φ is the amplitude angle. Using simple trigonometric transformations, this becomes [cos(θ+φ) + cos(θ-φ)]/2, and thus avoiding the need to use a multiplier. The schematic of the amplitude module is shown in figure 2. The lookup table is generated for cos(θ) where θ is the ten bit phase value converted to an angle. 127 represent decimal 1, and is added so that all resulting values are positive. For example, if the phase value is '0101010101' (ie. 120◦) the output of the lookup table is 63, ie. '00111111', when the actual value of cosine is -0.5. The lookup table utilises eight most significant bits of the phase to give an eight bit cosine output. Only half of the phase values (from 0 to 127) need to have entries in the table because of the symmetry of the cosine curve. The lookup table has to be used six times, two times for each phase. In an ordinary case, the lookup table would be instantiated six times, resulting in the unnecessary use of excess resources. The lookup table has to be instantiated only once for the most efficient code. This is done in the following way. A counter counting from 0 to 11 is made. When the value of the counter is 0, the value of (A+Θ) for the first phase is taken as input for the table. When it is 1, the output of the table is taken as cos(A+Θ). When it is 2, the value of (A-Θ) for the first phase is taken as input for the table. When it is 3, the output of the table is taken as cos(A-Θ).

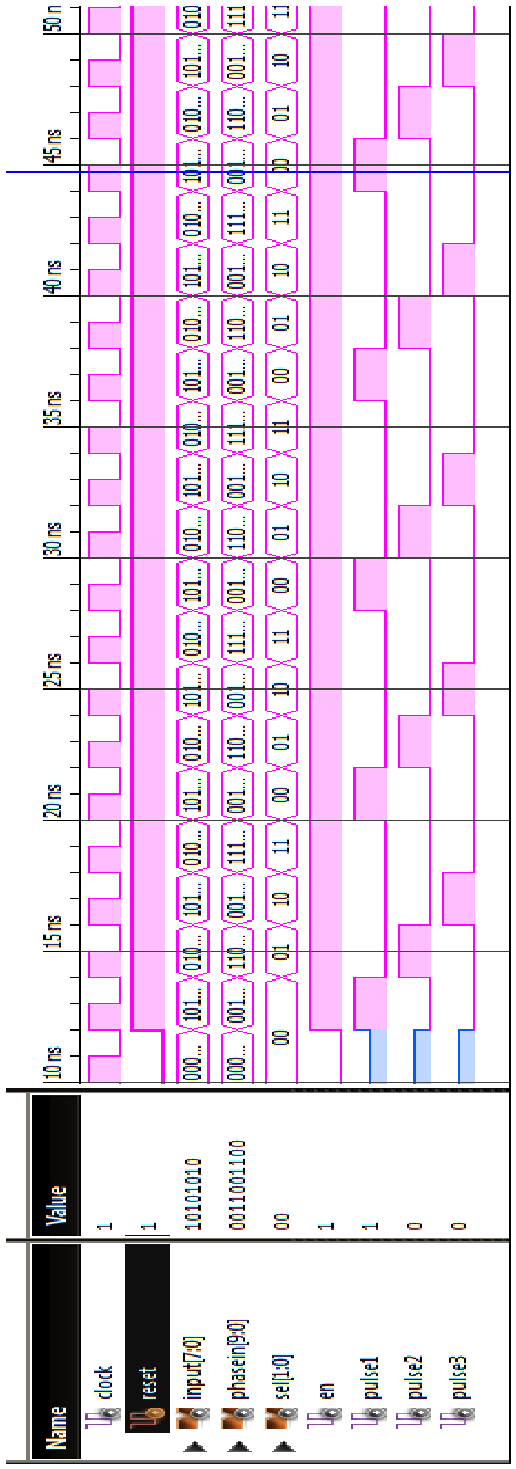

Figure 3 : Waveforms

F. PWM module

This entity is a counter-comparator pair used to generate the sinusoidal PWM output from a cosine input given by the Amplitude Module. Normally, the counter would count from 0 to 255 for an 8 bit input, but here, to have both half wave and quarter wave symmetry, the counter counts from zero to 512. Such symmetry greatly reduces inverter harmonics. The comparator outputs low if the value of the count is between (256-input) and (256+input), and high otherwise, thus resulting in both half wave and quarter wave symmetry. This entity also generates the clock for the oscillator which clearly has to have a frequency 1/512 times the clock used for the counter comparator pair. From the three SPWM signals, six gate driving pulses have to be generated. The pulses for one of the branches of the inverter have to be phase shifted by 180◦ and those for two adjacent branches have to be phase shifted 120◦. Thus, to generate the two out of-phase pulses from a single pulse, the following steps are done: 1. Zero crossing is detected, 2. the pulses representing the positive half cycle of the Sinusoid form the first pulse are generated. 3. The pulses representing the negative half cycle are inverted to form the second pulse are generated. 4. The process is repeated for the other two phases. The final pulses generated for two the IGBTs in any one of the branches will be as shown in figure 3.

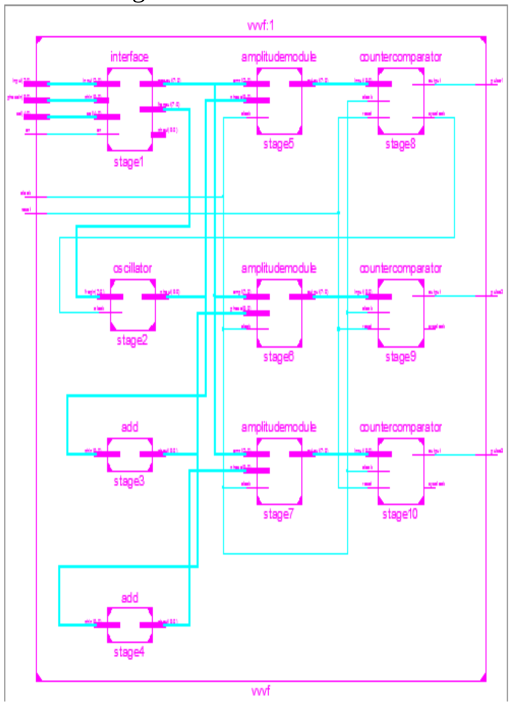

Figure 4: Top Module with Entities

G. Top level module

This is the top-level entity in which copies of all the other entities are instantiated. In other words, this is the entity that takes the inputs to the FPGA, viz. The 8 bit data signal, the two bit selection signal, the enable signal and the clock, and gives the three PWM sinusoids as outputs, using instances of each of the above mentioned entities shown in figure 4.

III. Results and Conclusion

The open loop control scheme for a three phase inverter is implemented using VHDL. The versatility in VHDL programming makes the designer to implement an efficient controller in it. The most important factor in support of using FPGA based designs is that it can be started from scratch and the design can be improved along the way by continuously testing and improving the code. Furthermore, using a digital controller make the system less susceptible to noise, temperature and other environmental factors. More significantly, the controller size and complexity is considerably reduced. The reconfigurable feature of FPGAs makes it more flexible. In conclusion, the VVVF controller is successfully implemented using FPGA and the experimental results show that the controller enables the inverter to produce a proper supply voltage which is fed to drive the three phase induction motor.

References

[1] R.Nandhakumar, S. Jeevananthan P. Dananjayan, *Design and Implementation of an FPGA-Based High Performance ASIC for Open Loop PWM Inverter*, IICPE, 2006, pp.349-354.

[2] Arulmozhiyal, R. Baskaran, K. Devarajan, N. Kanagaraj, *Space Vector Pulse Width Modulation Based Induction Motor Speed Control Using FPGA*, ICETET, 16-18 Dec 2009, pp. 242-247.

[3] Nitish Patel, Udaya Madawala, *A Bit stream based scalar control of an Induction Motor*, IECON, 2008, pp. 1071-1076.

[4] A. Fratta, G. Griffero, and S. Nieddu (2004), *Comparative analysis among DSP and FPGA-based control capabilities in PWM power converters* in Proceedings of the 30th Annual Conference of the IEEE Industrial Electronics Society (IECON .04). Novemeber: 257.262.

[5] A De Castro, A., P. Zumel, O. Garcia, T. Riesgo, and J. Uceda (2003), *Concurrent and simple digital controller of an AC/DC converter with power factor correction based on an FPGA,*IEEE Transactions on Power Electronics. 18(1 Part2): 334343.

[6] Thida Win, Nang Sabai, and Hnin Nandar Maung, *Analysis of Variable Frequency Three Phase Induction Motor Drive* , World Academy of Science, Engineering and Technology 42 2008.

[7] Silver Ott, Indrek Roasto, Dmitri Vinnikov, *Comparison of pulse width modulation methods for a quasi impedance source inverter*, 10th International Symposium „Topical Problems in the Field of Electrical and Power Engineering“ Pärnu, Estonia, January 10-15, 2011