The Islamic University – Gaza

Deanery of Graduate Studies

Faculty of Engineering

Electrical Engineering

الجامعة الإسلامية – غزة

عمادة الدراسات العليا

كلية الهندسة

الهندسة الكهربائية

Fuzzy Logic Speed Controllers Using FPGA Technique for Three-Phase Induction Motor Drives

Moayed N. EL Mobaied

Advisor

Dr. Basil Hamed

This Thesis is submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in

Electrical Engineering

1429هـ – 2008م

بسم الله الرحمن الرحيم

"يرفع الله الذين امنوا منكم والذين أوتوا العلم درجات والله بما تعملون خبير"

صدق الله العظيم

المجادلة – آية ١١

Dedication

To the memory of my beloved brother, the martyr, Ashraf.

To my parents, big brother, sisters, wife, and lovely kids:

Wafiqa and Ashraf.

Moayed Naser El-Mobaied

Acknowledgments

I wish to express my deepest gratitude to my advisor, Dr. Basil Hamed, for his professional assistance, support, advice and guidance throughout my thesis, and to my discussion committee, Dr. Hatem El Aydi and Dr. Assad Abu Jasser for their acceptance to discuss my thesis.

I would also like to extend my gratitude to my family for providing all the preconditions necessary to complete my studies, also for keeping me in their prayers.

Abstract

Fuzzy Logic Speed Controllers Using FPGA Technique for Three-Phase Induction Motor Drives

By

Moayed N. EL Mobaied

The design of intelligent control systems has become an area of intense research interest. The development of an effective methodology for the design of such control systems undoubtedly requires the synthesis of many concepts from artificial intelligence. A promising direction in the design of intelligent systems involves the use of Fuzzy logic control to discover the abilities of intelligent control systems in utilizing experience via rule-based knowledge.

The most commonly used controller in the industry field is the proportional-plus-integral-plus-derivative (PID) controller, which requires a mathematical model of the system. Fuzzy logic controller (FLC) provides an alternative to PID controller, especially when the available system models are inexact or unavailable. Also rapid advances in digital technologies have given designers the option of implementing controllers using Field Programmable Gate Array (FPGA) which depends on parallel programming. This method has many advantages over classical microprocessors. In this research, a novel Fuzzy-PID controller, which is fabricated on modern FPGA card (Spartan-3A, Xilinx Company, 2007) is proposed to implement a prototype of a speed controller for three-phase induction motor (squirrel cage type) as a an example for complex model systems.

The proposed controller and the pulse width modulator PWM inverter algorithm which have been built in FPGA appeared fast speed response and good stability in controlling the three-phase induction motor.

For comparison purpose, another two widely used controllers "PID and Fuzzy" have been implemented in the same FPGA card to examine the performance of the proposed system. These controllers have been tested using Matlab/Simulink program under speed and load

variation conditions. The results show that the Fuzzy-PID is the best controller between them.

In this thesis, the fuzzy logic control demonstrates good performance. Furthermore, fuzzy logic offers the advantage of faster design, and emulation of human control strategies. Also fuzzy control work well for high-order and nonlinear and shows the efficiency over the PID controller.

ملخص البحث

تصميم متحكم ضبابي بتقنية ال FPGA للتحكم بسرعة المحرك الحثي ذو ثلاثة أوجه

إعداد

مـــــــؤيــد نصر المبــيــض

إن تصميم الأنظمة الذكية اصبح من الأبحاث المهمة في عالم التحكم الرقمي والموجودة على أولويات اهتمام الباحثين في هذا المجال.من أجل تطوير تصاميم متميزة في هذا المجال فاننا بلا شك نحتاج الى تطوير مهارات في مجالات الأنظمة الذكية. يعتبر المتحكم بي أي دي من أكثر المتحكمات شيوعا في مجال الصناعة ،وهو يحتاج الى المعادلة الرياضية المكافئة للنظام ، ولقد أثبتت المتحكمات المنطقية الضبابية تفوقا على هذه الانظمة، خصوصا عندما تكون المعادلات المكافئة للأنظمة غير دقيقة أو لا يمكن استنتاجها.اضافة الى ذلك لقد قدم التطور التكنولوجي الرقمي للمصممين خيار تطبيق المتحكمات داخل ال FPGA والتي تعتمد على أنظمة البرمجة المتوازية والتي أظهرت العديد من الايجابيات على المتحككمات المبنية داخل المعالجات الدقيقة.

في هذا البحث تم تقديم فكرة مطورةحديثة في هذا المجال وهي اعداد متحكم مهجن يشمل ايجابيات كل من ال بي أي دي وايجابيات المتحكم الضبابي حيث تم تجميع هذا المتحكم وبرمجته داخل وحدة ال FPGA الحديثة سبارتن- أ-٣ المقدمة من شركة زيلينكس ، حيث استخدم هذا المتحكم الحديث بغرض التحكم بسرعة موتور ثلاثي الأوجه من نوع قفص السنجاب لكي يكون مثالا عن الأنظمة التي تكون فيها المعادلات المكافئة معقدة وغير خطية.

المتحكم المقترح وبرنامج الانفرتر المعتمد على نظام التحكم بعرض الموجة ،و اللذان تم بناؤهما داخل ال FPGA أظهرا استجابة سريعة لطلبات السرعة وايضا أظهرا استقرارية عالية في اداء الموتور ثلاثي الأوجه المستخدم في هذا البحث.

لغرض المقارنة تم تصميم متحكمين اخرين وهما المتحكم الضبابي والمتحكم ال بي أي دي كل على حدا ،حيث تم تطبيقهما على نفس لوحة ال FPGA لفحص الكفاءة للمتحكم المقدم.

لقد تم فحص هذه المحكمات الثلاثة باستخدام برنامج المحاكاة الماتلاب /سيميولينك تحت ظرف تغيير السرعة وتغيير الحمل المطبق على الجزء الدوار، نتائج المقارنة أظهرت تفوق المتحكم الهجين المقدم على المتحكمين الأخرين.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1 Introduction

Three phase Induction motors are the most common motors used in industrial motion control systems. Low-cost, simple and rugged design and low maintenance are the main advantages of induction motors. Due to that, these motors are often called the workhorse of the motion industry. Squirrel cage is the widely type of induction motors used in industry [1].

Generally, most of the industrial applications which contain induction motor need to vary their speed. However, induction motors can only run at their rated speed when they are connected directly to the main power supply. This is the reason why variable speed drives are needed to vary the rotor speed of an induction motor. The most popular algorithm for the control of a three-phase induction motor is the V/f control approach. Open-loop control is sometimes used in motor speed control system. However, open–loop AC motor speed control requires a precise speed profile to operate the motor from stand still to full speed. Speed error may arise due to load changes or external disturbances. To overcome these shortages, closed-loop control is frequently utilized in AC motor speed control system. Designing a robust controller will ensure the system to remains stable and keeps its required speed even the loads are applied or external disturbances occur [2].

The applications of induction motors in high accurate drives require more advanced control techniques so that those nonlinear and strongly coupled induction motors whose parameters are time-variant can be effectively controlled. At present, Proportional-Integral-Derivative "PID" controller, due to its simplicity, stability, and robustness, is a type of controller that is most widely applied [3]. However, it is difficult to design when the accurate model of plant is unknown. For induction motors, factors such as unknown load characteristic and parameter variation influence seriously the controlling effect of speed controller.

Therefore, some advanced control techniques such as variable structure control and

adaptive control etc. have been presented. However, the design of these controllers is based on known system model parameters and if the parameters can't be achieved, it is very troublesome in designing these controllers too [4].

Nowadays, many people paid a lot of attention to the application research of artificial intelligence in the AC drives. Fuzzy control dose not strictly need any mathematical model of the plant. It is based on plant operator experience, and it is very easy to apply. Fuzzy control gives robust performance for a linear or nonlinear plant with parameter variation. Fuzzy logic controller provides an alternative to PID controller since it is a good tool for the control of systems that are difficult in modeling [5]. Hardware implementation of the controller can be achieved in a number of ways to create new products. The most popular method of implementing fuzzy controller is using a general-purpose microprocessor or microcontroller. Generally, an 8-bit microprocessor can handle most of the necessary computations. Microprocessor based controllers are more economical, but often face difficulties in dealing with control systems that require high processing and input/output handling speeds. Rapid advances in digital technologies have given designers the option of implementing a controller on a variety of Programmable Logic Device (PLD), Field Programmable Gate Array (FPGA), etc. FPGA is suitable for fast implementation controller and can be programmed to do any type of digital functions. There are three main advantages of an FPGA over a microprocessor chip for controller designing:

- An FPGA has the ability to operate faster than a microprocessor chip.
- The new FPGAs that are on the market will support hardware that is upwards of one million gates, which increase program capacity.
- Because of the flexibility of the FPGA, additional functionality and user interface controls can be incorporated into the FPGA minimizing the requirement for additional external components.

FPGAs are programmed using Very High Speed Integrated Circuit hardware description language (VHDL) and a download cable connected to a host computer. Once they are programmed, they can be disconnected from the computer, and it will be running as stand alone device. The FPGAs can be programmed while they run, because they can be reprogrammed in the order of microseconds. This short time means that the system will not even sense that the chip was reprogrammed [6]. Applications of FPGAs include industrial motor drivers, real time systems, digital signal processing, aerospace and

defense systems, medical imaging, computer vision, speech recognition, cryptography, computer hardware emulation and a growing range of other areas.

1.2 Thesis Motivation and Objectives

1.2.1 Motivation

Nowadays, fuzzy logic controllers have an efficient performance over the traditional controller researches especially in nonlinear and complex model systems. FPGA is a new key technology used in modern control hardware implementation. Modem manufactures began to apply these technologies in their applications instead of the traditional ones, due to the low cost and widely features available in these controllers. Thus motivated me to investigate this topic.

1.2.2 Objectives

The main objective of this thesis is to build three different types of controllers (PID, Fuzzy, and Fuzzy-PID) which have been constructed on a FPGA card to be used as speed controllers for three-phase induction motor (squirrel cage type).

The specific objectives include:

- Improving FPGA knowledge.
- Improving VHDL programming skills.
- Designing PID controller skills.
- Designing Fuzzy controller skills.
- Building a variable-speed driver for three phase induction motor.

1.3 Literature review

Manny studies for three-phase induction motor speed controller appeared using the general PI technique. Volcanjk and Jezernik presented a novel design method using this technique in 1994 (IEEE) [7]. Fuzzy controller began in this field due to the drawbacks of the previous methods. Zidani, Benbouzid, M.E.H., and Diallo presented a Fuzzy efficient-optimization controller for induction motor drives in 2000 (IEEE) [8]. Shi, Chan, and Wong, presented a novel hybrid fuzzy-PI two-stage controller for an induction motor drive in 2001 (IEEE) [9]**.** Yong, Han, Kim, and Chang-Goo Lee presented sensorless vector control of induction motor using improved self-tuning fuzzy PID controller in 2003

(IEEE)[10]. The above papers present good performance controllers but without load variation conditions, and all of them had been constructed using microprocessors technique, due to the fast technology growing, FPGA technique appeared in the speed controller of the motor. Lin, Wang, and Huang, presented FPGA-based fuzzy sliding-mode control for a linear induction motor drive in 2005 (IEEE) [11],in this paper they implement only the fuzzy controller in FPGA and gets acceptable results. Priya, Kumar, and Renganarayanan, presented a FPGA based Fuzzy logic controller for dc electrical vehicle at Singapore in 2005 [12]. This paper was focused on dc motor and get a novel performance. Zhang, Li, and Collins, presented a digital Anti-Windup PI Controllers for variable-speed motor drives Using FPGA and Stochastic Theory in 2006 (IEEE) [13] but not deals with fuzzy approaches. It's memorable to note that the above mentioned papers have been selected only to describe the survey progress, but in real, there are many other papers have been written in the same field.

Due to the survey on the available resources, a Fuzzy-PID controller in FPGA for three-phase induction motor speed controller hasn't been presented yet.

1.4 Contribution

In this thesis three different types of controllers (PID, Fuzzy, and Fuzzy-PID) have been constructed in FPGA card which are used as a speed controller for three-phase induction motor (squirrel cage type). These controllers have been tested using Matlab/Simulink program under different speed and load variation conditions. The comparison result shows that the Fuzzy-PID controller is the best between them.

The novel approach, which is proposed in this thesis is: Design and practical implementation of a Fuzzy-PID controller using modern FPGA card (Spartan-3A, Xilinx Company, 2007) for speed control of a three-phase induction motor (squirrel cage type) as an application.

1.5 Outline of the thesis

The thesis is organized into six chapters. Chapter 2 handles some basic principles of three-phase induction motor. Chapter 3 focuses on Fuzzy logic sets. Chapter 4 deals with FPGA and VHDL software implementation. Chapter 5 presents the design of the three types of controllers, also the simulation and results are included. The last chapter concludes the design and the implementation and proposes some future work.

CHAPTER 2

Three-Phase Induction Motor

2.1 Introduction

AC induction motors are the most common motors used in industrial motion control systems, as well as in main powered home appliances. Hence, they are often called the workhorse of the motion industry. Induction motors are more rugged, require less maintenance, and are less expensive than dc machines of equal kilowatt and speed ratings [14].

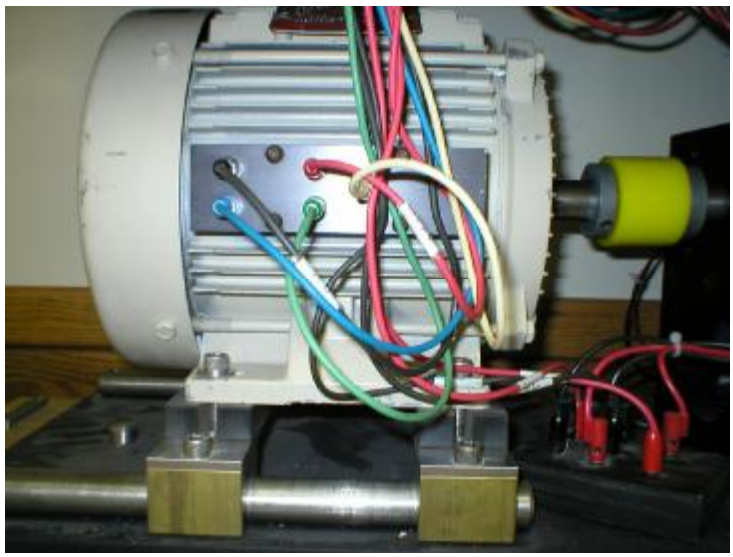


Figure 2.1: Three-phase induction motor.

Induction motors are constructed both for single-phase and three-phase operation. Three-phase induction motors are widely used for industrial applications such as in lifts, pumps, exhaust fans, grinding and filling machines, etc. Where as single-phase induction motors are used mainly for domestic-electrical appliance such as fans, refrigerators, washing machines, exhaust pumps, etc. Various types of AC induction motors are available in the market. Different motors are suitable for different applications. Although AC induction motors are easier to design than DC motors, the speed and the torque control in various types of AC induction motors require a greater understanding of the design and the characteristics of these motors [1].

2.2 Basic construction and operation principle

Like most motors, an AC induction motor has a fixed outer portion, called the stator and a rotor that spins inside with a carefully engineered air gap between the two. Virtually all electrical motors use magnetic field rotation to spin their rotors. A three-phase AC induction motor is the only type where the rotating magnetic field is created naturally in the stator because of the nature of the supply. DC motors depend either on mechanical or electronic commutation to create rotating magnetic fields. A single-phase AC induction motor depends on extra electrical components to produce this rotating magnetic field. Two sets of electromagnets are formed inside any motor. In an AC induction motor, one set of electromagnets is formed in the stator because of the AC supply connected to the stator windings. The alternating nature of the supply voltage induces an Electromagnetic Force (EMF) in the rotor (just like the voltage is induced in the transformer secondary) as per Lenz's law, thus generating another set of electromagnets; hence the name – induction motor. Interaction between the magnetic field of these electromagnets generates twisting force, or torque. As a result, the motor rotates in the direction of the resultant torque [1].

2.2.1 Induction motor construction

A 3-phase induction motor is shown in Figure 2.2 has two main parts:

a. Stator.

b. Rotor.

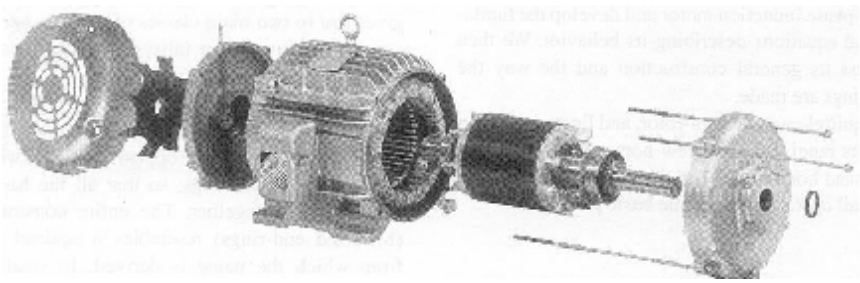


Figure 2.2: Induction motor construction.

2.2.1.a Stator

The stator is constructed from several thin laminations of aluminum or cast iron. They are punched and clamped together to form a hollow cylinder (stator core) with slots as shown in Figure 2.3. These slots contain coils of insulated wires. Each grouping of coils, together with the core it surrounds, forms an electromagnet (a pair of poles) on the application of AC supply. The number of poles of an AC induction motor depends on the internal connection of the stator windings. The stator windings are connected directly to the power

source. Internally they are connected in such a way that on applying AC supply, a rotating magnetic field is created [1].

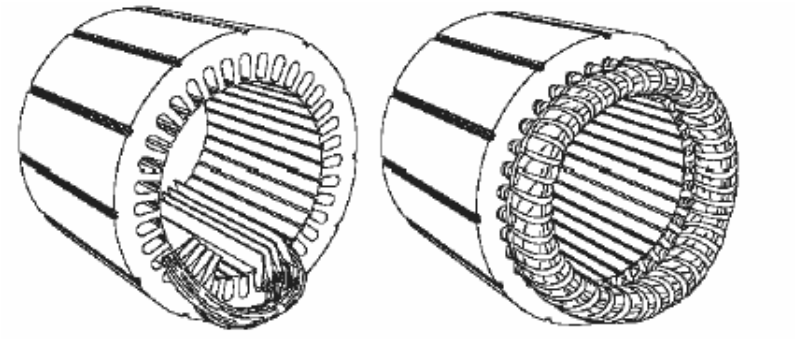


Figure 2.3: Typical stator.

2.2.1.b Rotor

The rotor is made up of several thin steel laminations with evenly spaced bars, which are made up of aluminum or copper. Induction motors are classified in two categories based on the construction of the rotor: squirrel cage motors and slip ring motors, but the stator part is the same in both motors.

Squirrel cage motor represents about 90% of induction motors. That is due to the simplest and rugged construction of this motors type. The rotor consists of cylindrical laminated core with axially placed parallel slots for carrying the conductors. Each slot carries copper, aluminum, or alloy bar. If the slots are semi closed, then these bars are inserted from the ends. These rotor bars are permanently short-circuited both ends by means of the end rings, as shown Figure 2.4 [1]. This total assembly resembles the look of a squirrel cage, which gives the rotor its name as shown in Figure 2.5.

The rotor slots are not exactly parallel to the shaft. Instead, they are given a skew for two main reasons. The first reason is to make the motor run quietly by reducing magnetic hum and to decrease slot harmonics. The second reason is to help reduce the locking tendency of the rotor. The rotor teeth tend to remain locked under the stator teeth due to direct magnetic attraction between the two. This happens when the number of stator teeth is equal to the number of rotor teeth [14]. The rotor is mounted on the shaft using bearings on each end; one end of the shaft is normally kept longer than the other for driving the load. Some motors may have an accessory shaft on the non-driving end for mounting speed or position sensing devices. Between the stator and the rotor, there exists an air gap, through which due to induction, the energy is transferred from the stator to the rotor. The generated torque forces the rotor and then the load to rotate [15].

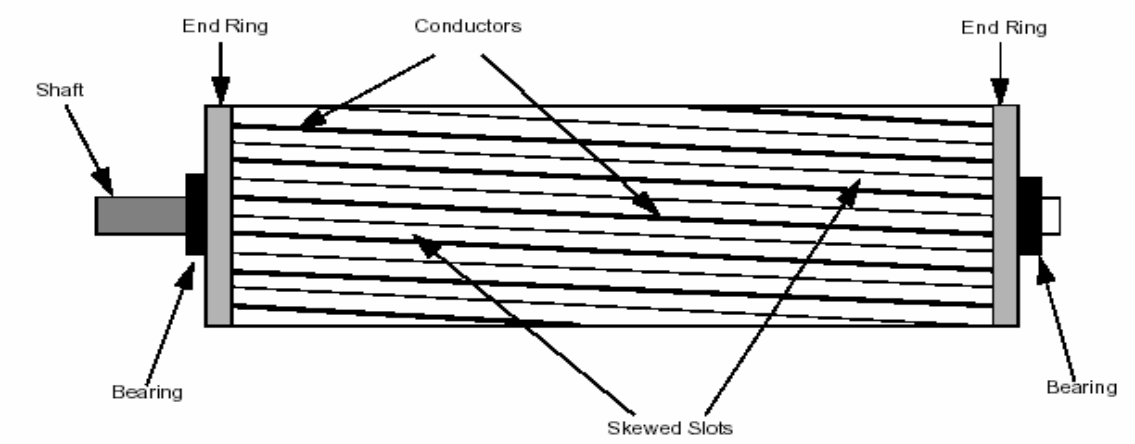


Figure 2.4: Squirrel-cage construction.

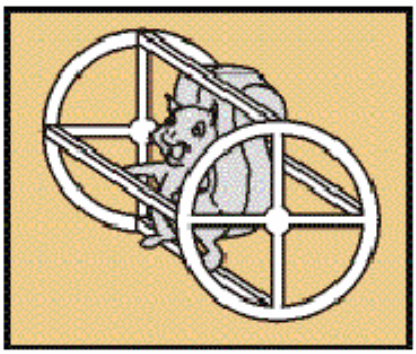


Figure 2.5: Squirrel-cage.

A wound rotor shown in Figure 2.6 has a 3-phase winding, similar to the stator winding. The rotor winding terminals are connected to three slip rings which turn with the rotor. The slip rings/brushes allow external resistors to be connected in series with the winding.

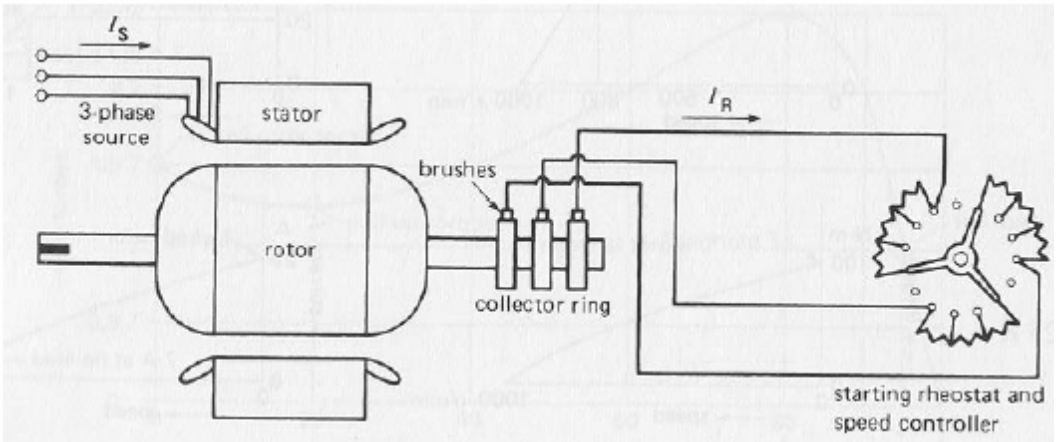


Figure 2.6: Wound rotor.

The external resistor can be used to boost the starting torque of the motor and change the speed-torque characteristic. When running under normal conditions, the slip rings are short circuited, using an external metal collar, which is pushed along the shaft to connect the rings.

So, in normal conditions, the slip ring motor functions like a squirrel cage motor. The downside of the slip ring motor is that slip rings and brush assemblies need regular maintenance, which is a cost not applicable to the standard cage motor. This type of motor is used in applications for driving variable torque variable speed loads, such as in printing presses, compressors, conveyer belts, hoists and elevators [15].

2.2.2 Squirrel-cage induction motor operation principle

Due to the above comparison between the two kinds in this thesis, the squirrel-cage type has been used. To get familiar with the operation of this kind of motors, we can consider the series of conductors (length L) whose extremities are shorted by bars A and B. A permanent magnet moves at a speed v, so that its magnetic field sweeps across the conductors [15].

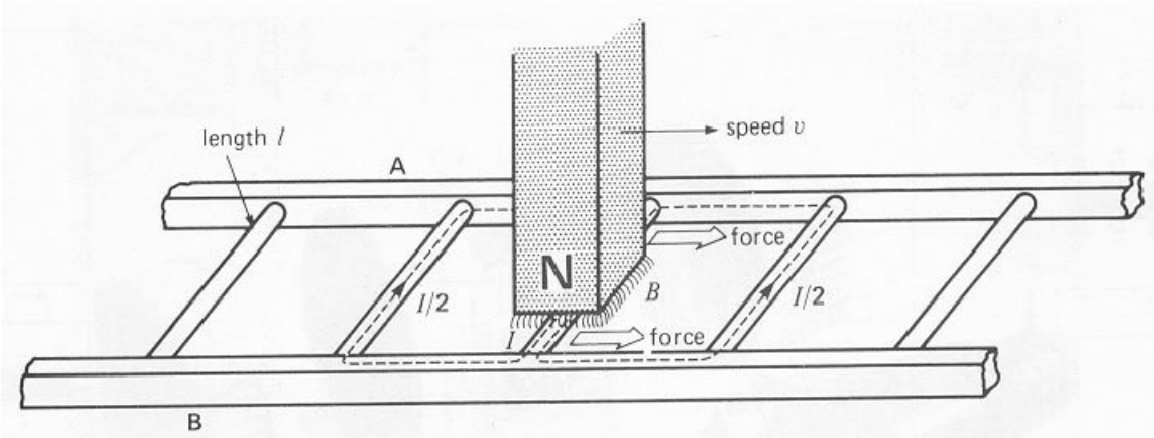


Figure 2.7: Series of conductors of length l.

The following sequence of events takes place:

- A voltage $E = B*L*V$ is induced in each conductor while it is being cut by the flux (Faraday's Law).
  where B is the magnetic field, V is the linear speed, and B,I,V are mutually perpendicular.
- The induced voltage produces currents which circulate in a loop around the conductors (through the bars).
- Since the current-carrying conductors lie in a magnetic field, they experience a mechanical force (Lorentz force).
- The force always acts in a direction to drag the conductor along with the magnetic field.

If the above ladder has been enclosed upon itself to form a squirrel cage, and place it in a rotating magnetic field, then a rotor for induction motor will be constructed establish as shown in Figure 2.8 [1].

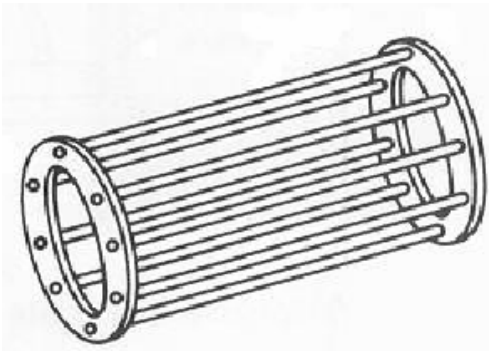


Figure 2.8: Squirrel-cage shape.

2.3 Rotating magnetic field

There are two kinds of rotating magnetic field which are rotating in the induction motor:

2.3.1 Magnetic field in the stator

The stator represents the stationary part of the motor which consists of a group of individual electro-magnets arranged in such a way that they form a hollow cylinder, with one pole of each magnet facing toward the center of the group. The term 'stator' is derived from the word stationary. The rotor represents the rotating part of the motor, which consists of a group of electro-magnets arranged around a cylinder, with the poles facing toward the stator poles. The rotor is located inside the stator and is mounted on the motor's shaft. The term 'rotor' is derived from the word rotating. The magnetic interaction between the stator and rotor will rotate the motor shaft. This rotation will occur because unlike magnetic poles attract each other and like poles repel. If the polarity of the stator poles is changed in such a way that their combined magnetic field rotates, then the rotor will follow and rotate with the magnetic field of the stator.

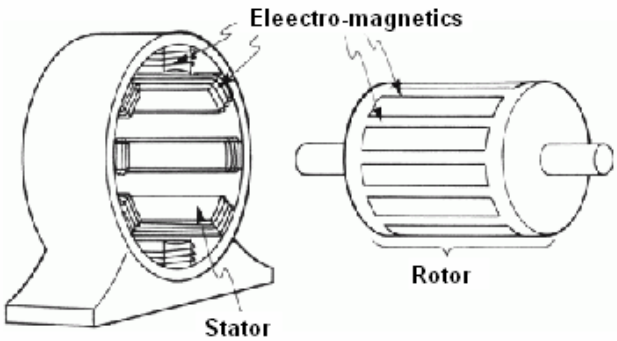


Figure 2.9: Stator and Rotor.

To be familiar with fundamental of the rotating magnetic fields it can be assumed that the stator has six magnetic poles and the rotor has two poles [15].

At time 1, stator poles A-1 and C-2 are north poles and the opposite poles, A-2 and C-1, are south poles. The S-pole of the rotor is attracted by the two N-poles of the stator and the N-pole of the rotor is attracted by the two south poles of the stator.

At time 2, the polarity of the stator poles is changed so that now C-2 and B-1 are N-poles and C-1 and B-2 are S-poles. The rotor then is forced to rotate 60 degrees to line up with the stator poles as shown in Figure 2.10.

At time 3, B-1 and A-2 are N-poles.

At time 4, A-2 and C-1 are N-poles.

As each change is made, the poles of the rotor are attracted by the opposite poles on the stator. Thus, as the magnetic field of the stator rotates, the rotor is forced to rotate with it.

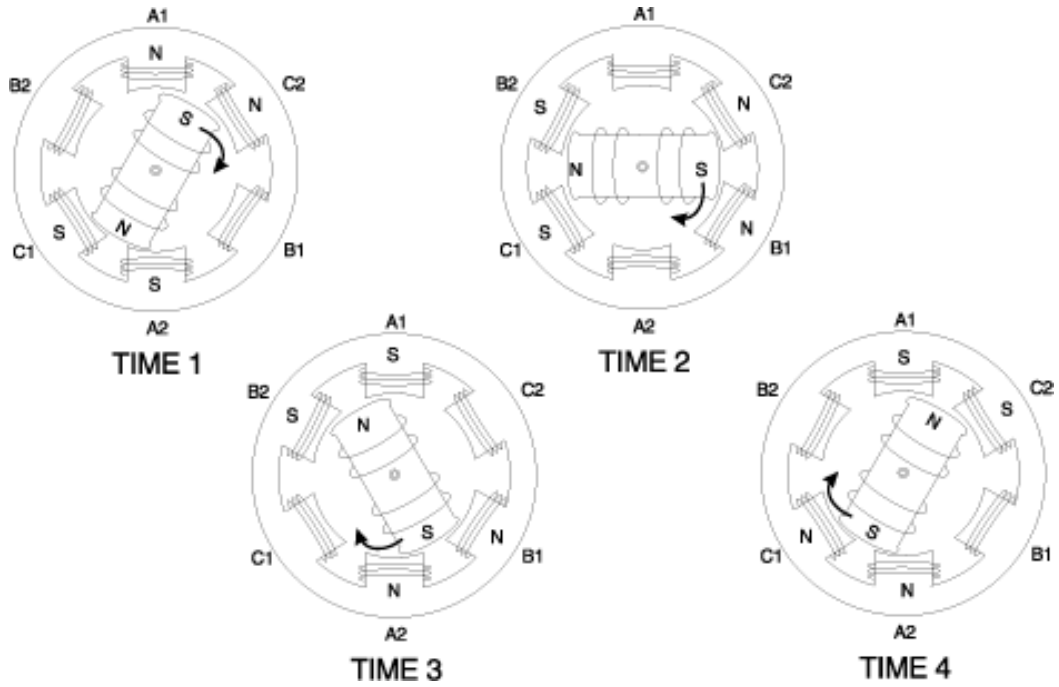


Figure 2.10: Rotating magnetic field.

To produce a rotating magnetic field in the stator of a three-phase AC motor, we need to direct couple the 3 phase power supply to the stator terminal. Each phase of the three-phase power supply is connected to opposite poles and the associated coils are wound in the same direction. The polarities of the poles are determined by the direction of the current flow through the coil. Therefore, if two opposite stator electro-magnets are wound in the same direction, the polarity of the facing poles must be opposite. Therefore, when pole A1 is N, pole A2 is S. When pole B1 is N, B2 is S and so forth.

The windings A_N, B_N, C_N are mechanically spaced at 120°from each other. AC currents I_a, I_b and I_c will flow in the windings, but will be displaced in time. Each winding produces its own EMF, which creates a flux across the hollow interior of the stator. The three fluxes combine to produce a magnetic field that rotates at the same frequency as the supply [3].

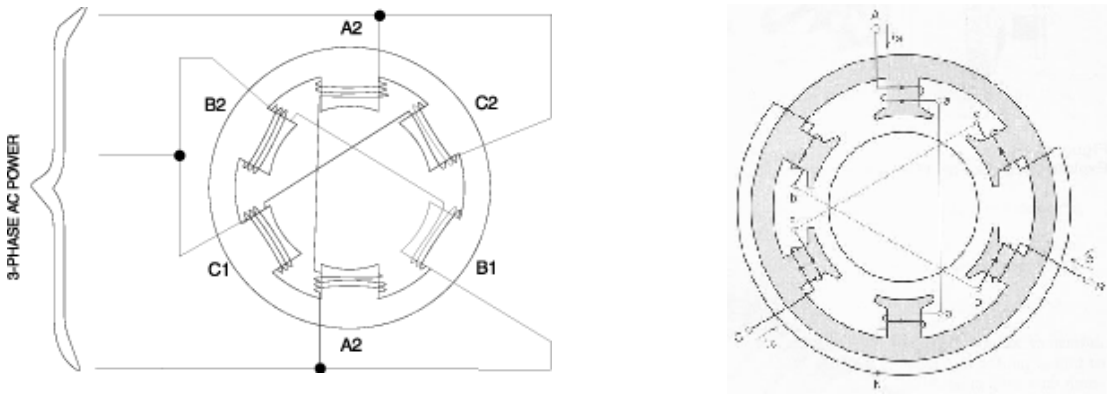


Figure 2.11: Six poles stator connections.

The phase current waveforms follow each other in the sequence A-B-C. This produces a clockwise rotating magnetic field. If we interchange any two of the lines connected to the stator, the new phase sequence will be A-C-B. This will produce a counterclockwise rotating field, reversing the motor direction. In practice, induction motors have internal diameters that are smooth, instead of having salient poles. Also, instead of a single coil per pole, many coils are lodged in adjacent slots. The staggered coils are connected in series to form a phase group. Spreading the coil in this manner creates a sinusoidal flux distribution per pole, which improves performance and makes the motor less noisy.

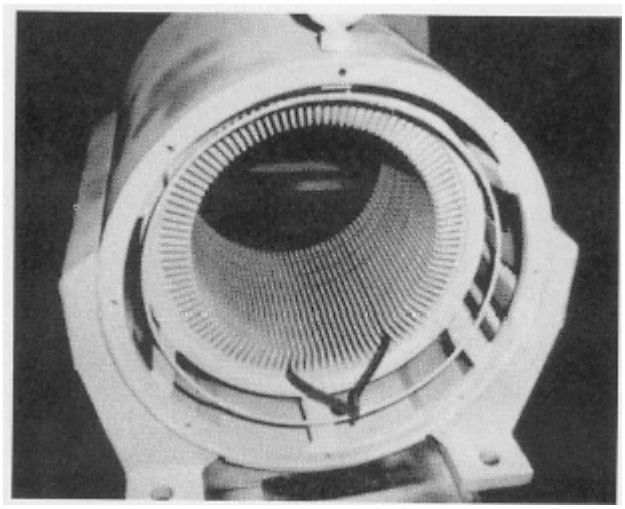


Figure 2.12: Six poles stator construction.

The rotating speed of the revolving flux can be reduced by increasing the number of poles (in multiples of two). In a four-pole stator, the phase groups span an angle of 90°. In a six-pole stator, the phase groups span an angle of 60° and so on. Figure 2.13 shows how the rotating magnetic field is produced. At time1, the current flow in the phase "A" poles is positive making A-1 N and pole A-2 is S. The current flow in the phase "C" poles is negative, making C-2 a N-pole and C-1 is S. There is no current flow in phase "B", so these poles are not magnetized. At time 2, the phases have shifted 60 degrees, making poles C-2 and B-1 both N and C-1 and B-2 both S. Thus, as the phases shift their current flow, the resultant N and S poles move clockwise around the stator, producing a rotating magnetic field. The rotor acts like a bar magnet, being pulled along by the rotating magnetic field [15].

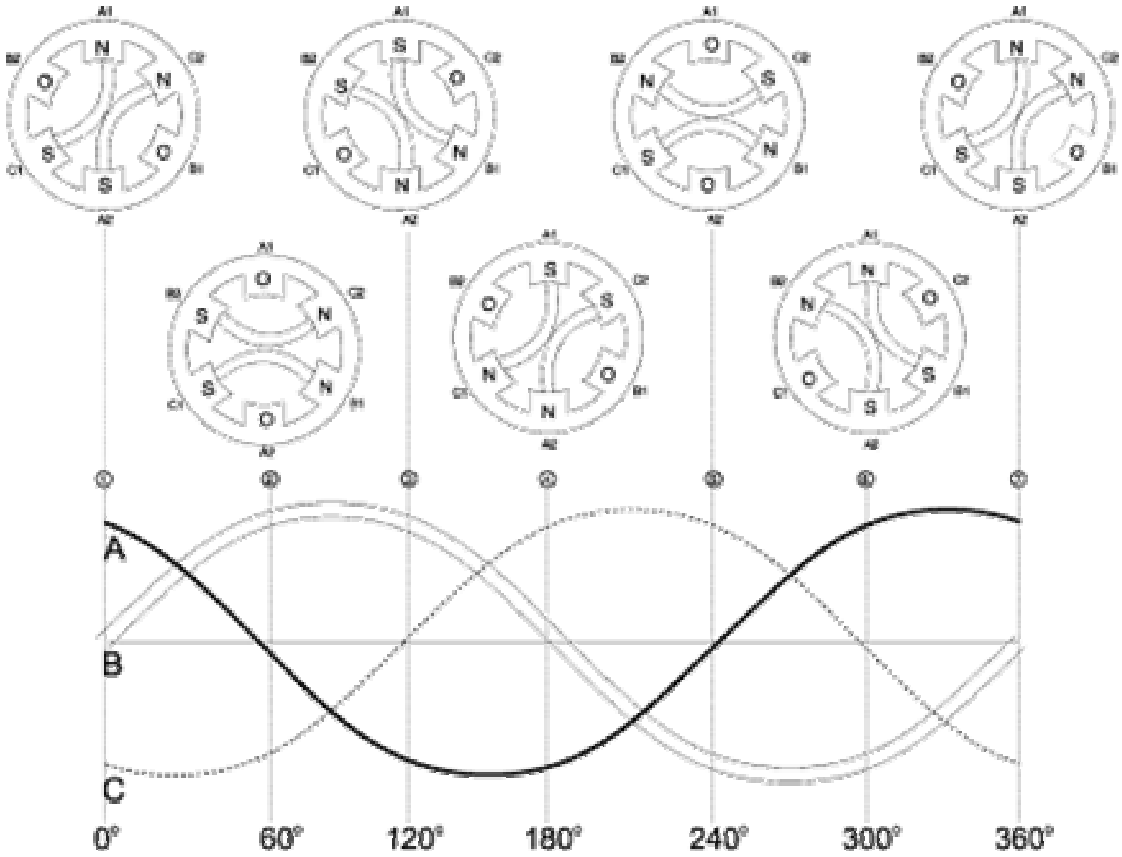


Figure 2.13: Magnetic field Stages.

2.3.2 Magnetic field in the rotor

There is no external power supply for the rotor part. It is a natural phenomena which occurs when a conductor which contains aluminum bars shown in Figure 2.14, is moved through an existing magnetic field or when a magnetic field is moved past a conductor. In either case, the relative motion of the two causes an electric current to flow in the

conductor. This is referred to as "induced" current flow. In other words, in an induction motor the current flow in the rotor is not caused by any direct connection of the conductors to a voltage source, but rather by the influence of the rotor conductors cutting across the lines of flux produced by the stator magnetic fields. The induced current which is produced in the rotor results in a magnetic field around the rotor conductors as shown in Figure 2.15. This magnetic field around each rotor conductor will cause each rotor conductor to act like the permanent magnet. As the magnetic field of the stator rotates, due to the effect of the three-phase AC power supply, the induced magnetic field of the rotor will be attracted and will follow the rotation. The rotor is connected to the motor shaft, so the shaft will rotate and drive the connection load [15].

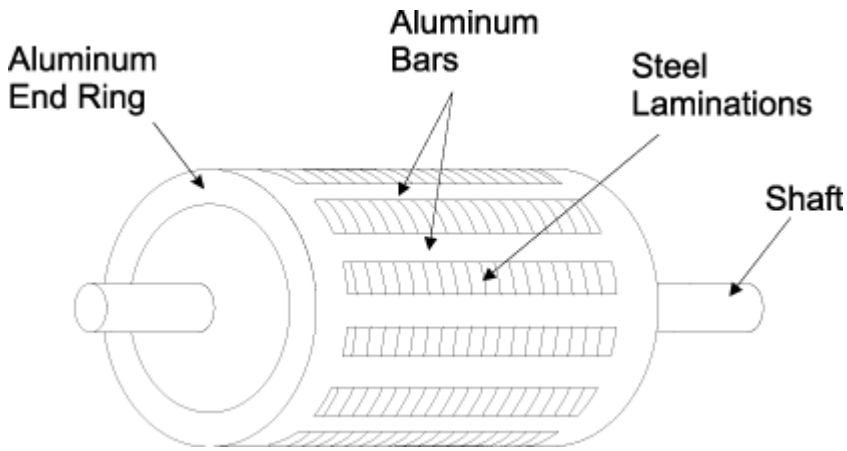


Figure 2.14: Construction of an AC induction motor's rotor.

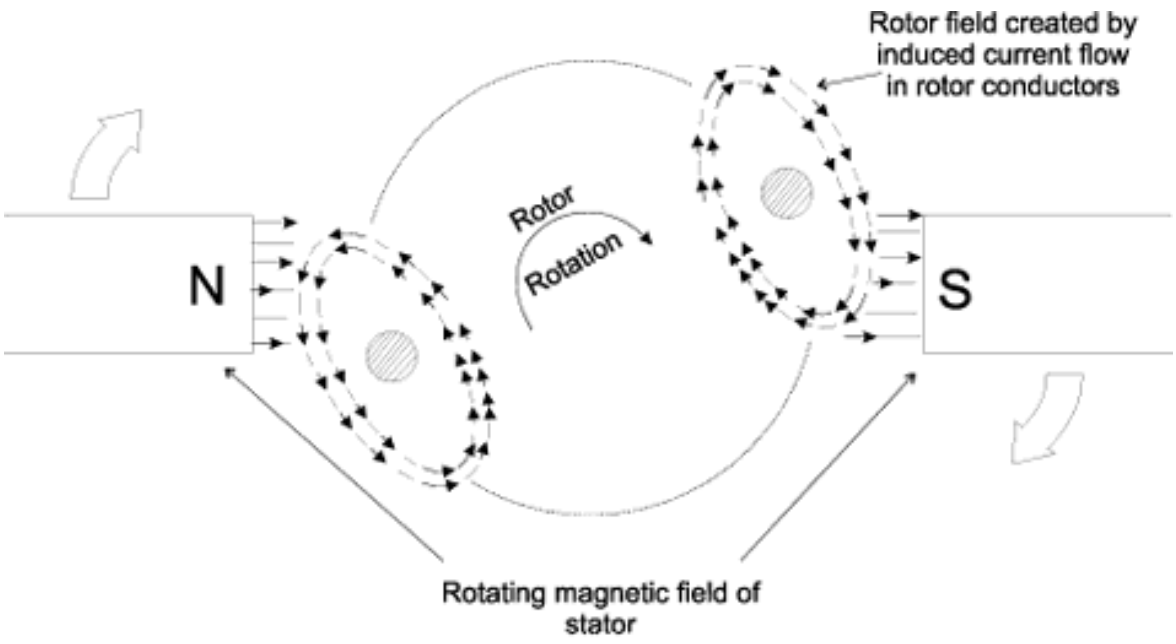


Figure 2.15: Voltage induced in the rotor.

2.4 Speed of an induction motor

The magnetic field created in the stator rotates at a synchronous speed (N_s) which can be obtained from the following formula:

$$N_s = 120 * \frac{f}{p} \tag{2.1}$$

Where

N_s : The synchronous speed of the stator magnetic field in RPM.

P : The number of poles on the stator.

F : The supply frequency in Hertz.

The magnetic field produced in the rotor because of the induced voltage is alternating in nature. To reduce the relative speed, with respect to the stator, the rotor starts running in the same direction as that of the stator flux and tries to catch up with the rotating flux. However, in practice, the rotor never succeeds in "catching up" to the stator field. The rotor runs slower than the speed of the stator field. This speed is called the Base Speed (N_b). The difference between N_S and N_b is called the slip. The slip varies with the load. An increase in load will cause the rotor to slow down or increase slip. A decrease in load will cause the rotor to speed up or decrease slip. The slip is expressed as a percentage and can be determined with the following formula [1]:

$$\%slip = \frac{N_s - N_b}{N_s} * 100 \tag{2.2}$$

2.5 Torque equation governing motor operation

The motor load system can be described by a fundamental torque equation.

$$T - T_l = J * \frac{dW_m}{dt} + W_m * \frac{dJ}{dt} \tag{2.3}$$

Where:

T : The instantaneous value of the developed motor torque (N.m).

T_l : The instantaneous value of the load torque (N.m).

W_m: The instantaneous angular velocity of the motor shaft (rad/sec).

J : The moment of inertia of the motor load system ($kg.m^2$)

For drives with constant inertia, $(dJ/dt) = 0$. Therefore, the equation would be:

$$T = T_l + J * \frac{dW_m}{dt} \tag{2.4}$$

This shows that the torque developed by the motor is counter balanced by a load torque, T_l and a dynamic torque, $J\frac{dw_m}{dt}$. The torque component, $J\frac{dw_m}{dt}$, is called the dynamic torque because it is present only during the transient operations. The drive accelerates or decelerates depending on whether T is greater or less than T_l. During acceleration, the motor should supply not only the load torque, but an additional torque component $J\frac{dw_m}{dt}$, in order to overcome the drive inertia. In drives with large inertia, such as electric trains, the motor torque must exceed the load torque by a large amount in order to get adequate acceleration. In drives requiring fast transient response, the motor torque should be maintained at the highest value and the motor load system should be designed with the lowest possible inertia. The energy associated with the dynamic torque, $J\frac{dw_m}{dt}$, is stored in the form of kinetic energy (KE) given by, $J\frac{w_m^2}{2}$. During deceleration, the dynamic torque, $J\frac{dw_m}{dt}$, has a negative sign. Therefore, it assists the motor developed torque T and maintains the drive motion by extracting energy from the stored kinetic energy. To summarize, in order to get steady state rotation of the motor, the torque developed by the motor (T) should always be equal to the torque requirement of the load (T_l). The torque-speed curve of the typical three-phase induction motor is shown in Figure 2.16 [1].

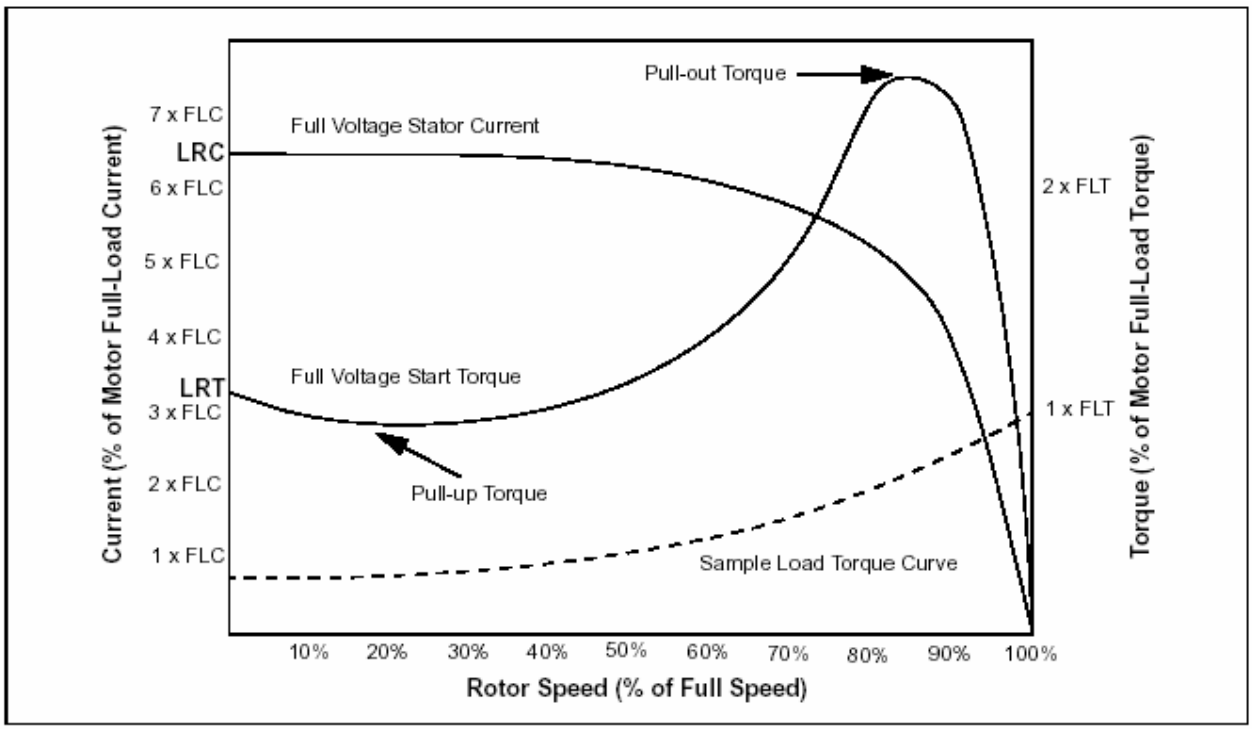


Figure 2.16: Typical torque-speed curve of 3-phase AC induction motor.

2.5.1 Starting characteristic

Induction motors, at rest, appear just like a short circuited transformer and if connected to the full supply voltage, draw a very high current known as the "Locked Rotor Current." They also produce torque which is known as the "Locked Rotor Torque". The Locked Rotor Torque (LRT) and the Locked Rotor Current (LRC) are function of the terminal voltage of the motor and the motor design. As the motor accelerates, both the torque and the current will tend to alter with rotor speed if the voltage is maintained constant. The starting current of a motor with a fixed voltage will drop very slowly as the motor accelerates and will only begin to fall significantly when the motor has reached at least 80% of the full speed. The actual curves for the induction motors can vary considerably between designs but the general trend is for a high current until the motor has almost reached full speed. The LRC of a motor can range from 500% of Full-Load Current (FLCu) to as high as 1400% of FLCu. Typically, good motors fall in the range of 550% to 750% of FLCu. The starting torque of an induction motor starting with a fixed voltage will drop a little to the minimum torque, known as the pull-up torque, as the motor accelerates and then rises to a maximum torque, known as the breakdown or pull-out torque, at almost full speed and then drop to zero at the synchronous speed. The curve of the start torque against the rotor speed is dependant on the terminal voltage and the rotor design. The LRT of an induction motor can vary from as low as 60% of FLT to as high as 350% of FLT. The pull-up torque can be as low as 40% of FLT and the breakdown torque can be as high as 350% of FLT. Typically, LRTs for medium to large motors are in the order of 120% of FLT to 280% of FLT [1].

2.5.2 Running characteristic

Once the motor is up to speed, it operates at a low slip, at a speed determined by the number of the stator poles. Typically, the full-load slip for the squirrel cage induction motor is less than 5%. The actual full-load slip of a particular motor is dependant on the motor design. The typical base speed of the four pole induction motor varies between 1420 and 1480 RPM at 50 Hz, while the synchronous speed is 1500 RPM at 50 Hz. The current drawn by the induction motor has two components: reactive component (magnetizing current) and active component (working current). The magnetizing current is independent of the load but is dependant on the design of the stator and the stator voltage. The actual magnetizing current of the induction motor can vary, from as low as 20% of FLCu for the large two pole machine, to as high as 60% for the small eight pole machine.

The working current of the motor is directly proportional to the load. The tendency for the large machines and high-speed machines is to exhibit a low magnetizing current, while for the low-speed machines and small machines the tendency is to exhibit a high magnetizing current. A typical medium sized four pole machine has a magnetizing current of about 33% of FLC. A low magnetizing current indicates a low iron loss, while a high magnetizing current indicates an increase in iron loss and a resultant reduction in the operating efficiency. Typically, the operating efficiency of the induction motor is highest at 3/4 load and varies from less than 60% for small low-speed motors to greater than 92% for large high-speed motors. The operating PF and efficiencies are generally quoted on the motor data sheets. As seen in the speed-torque characteristics, torque is highly nonlinear as the speed varies. In many applications, the speed needs to be varied, which makes the torque vary [1].

2.6 V/F control theory

The most widely way to control the motor's shaft speed is to vary the supply frequency with constant voltage over frequency percent. If the induction motor is supplied with its rated voltage and frequency, the flux produced will be at the optimum design value. As it can be seen in the speed-torque characteristics shown in Figure 2.16, the induction motor draws the rated current and delivers the rated torque at the base speed. When the load is increased (over-rated load), while running at base speed, the speed drops and the slip increases. As has been presented in the earlier section, the motor can take up to 2.5 times the rated torque with around 20% drop in the speed. Any further increase of load on the shaft can stall the motor. Reducing the supply frequency below the rated value whilst maintaining the rated supply voltage will cause an increase in motor flux. If the frequency is increased above its rated value, the flux and hence torque will decrease. Figure 2.17 illustrates the effect on the motor torque/speed characteristic as the operating frequency is varied with constant supply voltage applied [16].

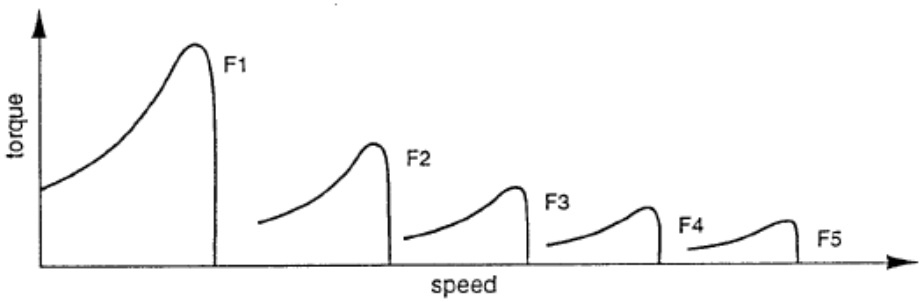


Figure 2.17: Torque/speed characteristic curve due to variable frequency and constant voltage.

The torque developed by the motor is directly proportional to the magnetic field produced by the stator. So, the voltage applied to the stator is directly proportional to the product of stator flux and angular velocity. This makes the flux produced by the stator proportional to the ratio of applied voltage and frequency of supply. By varying the frequency, the speed of the motor can be varied. Therefore, by varying the voltage and frequency at the same ratio, flux and hence, the torque can be kept constant throughout the speed range.

$$\begin{aligned} &Stator\,Voltage(V)\ a\ \ [Stator\,Flux\,(\Phi)]*[Angular\,velocity(w)] \\ &V\ \ a\ \ \Phi * 2pf \\ &\Phi\ \ a\ \ \text{V/F} \end{aligned} \tag{2.5}$$

With a variable frequency of the supply, the motor impedance will change with the frequency, influencing also the magnetic flux produced at the motor air gap. Since the torque produced in the motor is a function of the air gap magnetic flux, it is necessary to compensate for the motor impedance change by changing the voltage applied to the motor to maintain a constant flux over the operating frequency range.. This is achieved by increasing the motor voltage, Vs, at the lower frequencies and is termed 'voltage boost'. Figure 218 illustrates the effect of voltage boost on the motor torque/speed characteristic.

.

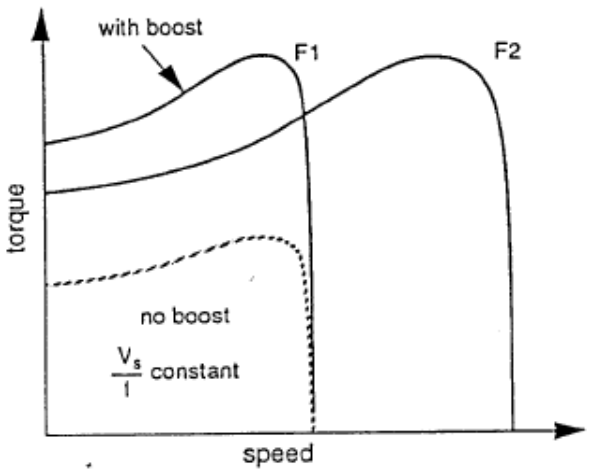


Figure 2.18: Torque/speed characteristic with constant V/F ratio.

Figure 2.19 shows the relation between the voltage and torque versus frequency. Which demonstrates that the voltage and frequency being increased up to the base speed. At base speed, the voltage and frequency reach the rated values as listed in the nameplate. It's possible to drive the motor beyond base speed by increasing the frequency further.

However, the voltage applied cannot be increased beyond the rated voltage. Therefore, only the frequency can be increased, which results in the field weakening and the torque available being reduced. Above base speed, the factors governing torque become complex,

since friction and windage losses increase significantly at higher speeds. Hence, the torque curve becomes nonlinear with respect to speed or frequency [17].

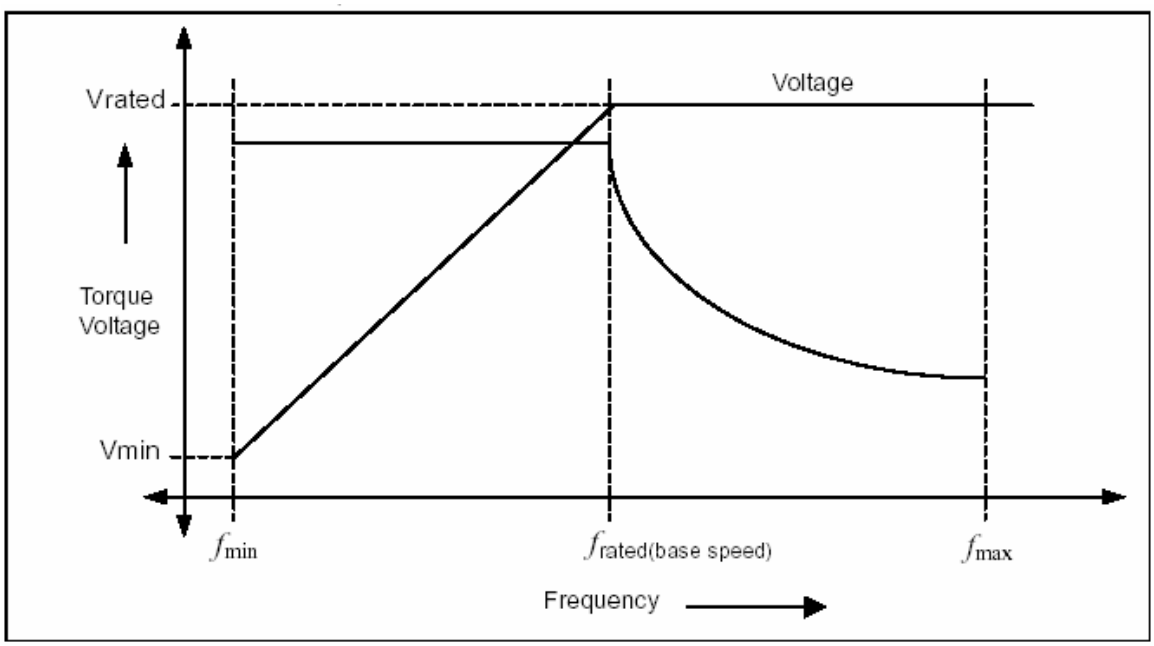


Figure 2.19: Frequency -Torque characteristics with V/F Control.

2.7 Equivalent circuit of the induction motor

The polyphase induction motor can be assumed as a polyphase transformer since the EMF produced by the rotor currents is rotating at synchronous speed relative to the stator winding, it induces a source frequency voltage just as in a normal transformer. Figure 2.20 shows the equivalent circuit of the induction motor [18].

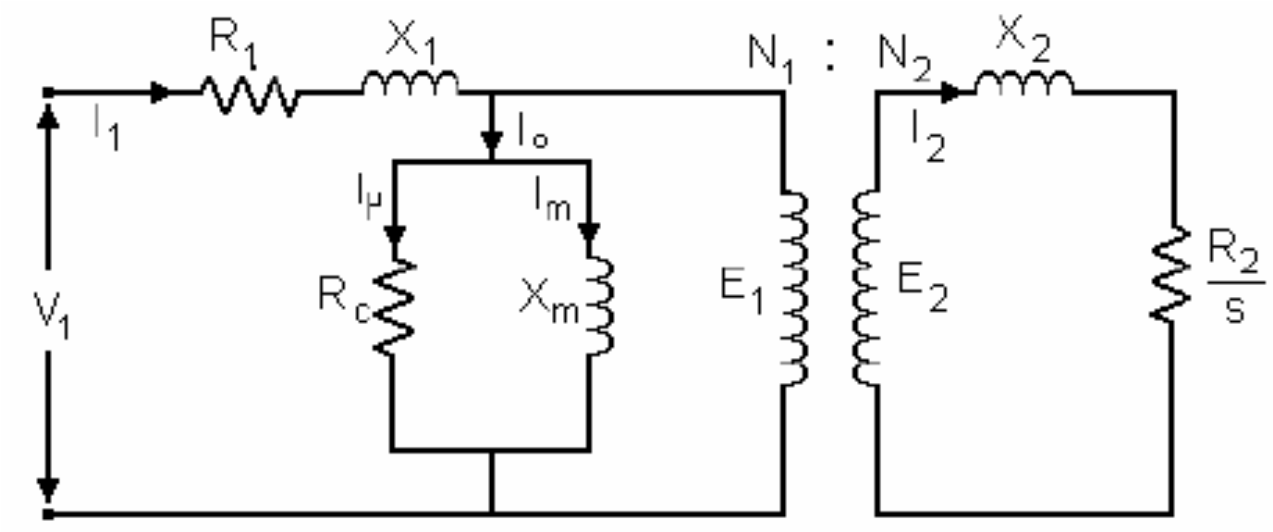


Figure 2.20: General equivalent circuit of the induction motor.

Where

$v_1 = applied\ voltage$
$R_1 = stator\ resistor$
$L_1 = stator\ \ leakage\ induc\tan ce$
$X_1 = 2pfL_1 = leakage\ reac\tan ce$
$R_2 = rotor\ winding\ resis\tan ce$
$L_b = rotor\ leakage\ induc\tan ce\ at\ s = 1$
$X_b = 2pfL_b = rotor\ leakage\ reac\tan ce\ at\ s = 1$
$X2 = 2psfL_b = sX_b = rotot\ reactacne\ at\ any\ s$
$X_m = magnetizing\ reac\tan ce$
$R_c = core\ resis\tan ce$
$N_1 = stator\ turns\ per\ phase$
$N_2 = rotor\ turns\ per\ phase$
$K_{w1} = stator\ winding\ factor$
$K_{w2} = rotor\ winding\ factor$
$f_m = \max imum\ phase\ flux$
$E_1 = 4.44fN_1K_{w1}f_m = stator\ emf$
$E_b = 4.44fN_2K_{w2}f_m = rotor\ emf\ at\ s = 1$
$E_r = sE_b = rotor\ emf\ at\ any\ s$
$I_2 = rotor\ winding\ current$
$I_1 = stator\ current$
$I_m = core - loss\ current$
$I_m = magnetiztion\ current$

$$I_o = I_m + I_m = excitation\ current$$

The secondary parameters and variables of the model can be referred to the primary. For convenience, this is done by adding primes to the symbols for the secondary resistance, leakage reactance, current, and voltage as shown in Figure 2.21.

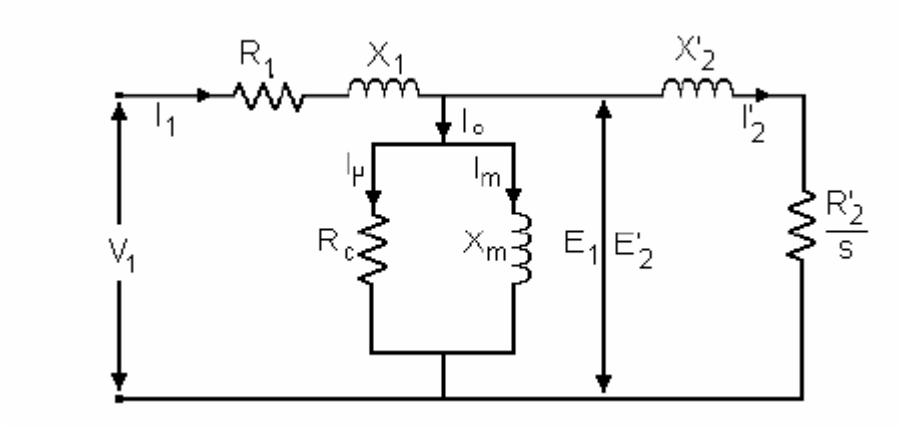


Figure 2.21: Parameter of rotor referred to the primary

Where:

$$E_2' = \left(N_1/N_2\right)E_2$$
$$\frac{R_2'}{S} = \frac{R_2}{S}\left(N_1/N_2\right)^2$$
$$X_2' = X_2\left(N_1/N_2\right)^2$$
$$I_2' = I_2\left(N2/N1\right)$$

Then we can rewrite the term $\frac{R_2'}{S}$ into the following:

$$\frac{R_2'}{S} = R_2' + R_2'\left(\frac{1-S}{S}\right)$$

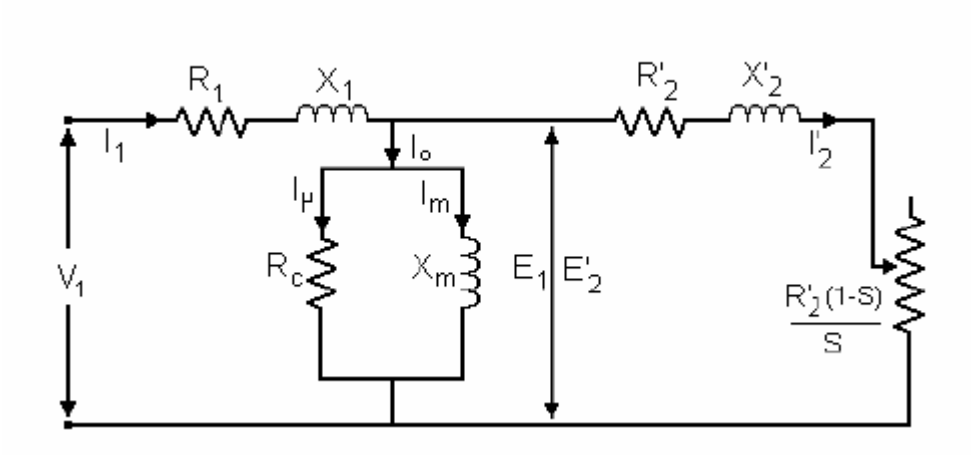


Figure 2.22: Equivalent circuit of induction motor.

In the induction machine the stator core loss exists as long as the machine is connected to a source, the rotor core loss exists when the slip is any value other than zero, and the windage and friction losses exist when the machine rotates. Since the windage and friction losses are significantly greater than others, it is common to group all of them as rotational losses and remove R_m from the circuit model. This is convenient since the no-load power is the sum of these lenses other than the small copper losses, and normally it is not possible to separate the core loss from the mechanical loss. Naturally this introduce a small error in computed values of gap power, but this error is quite insignificant .When this is done, the equivalent circuit appears as shown in Figure 2.23.

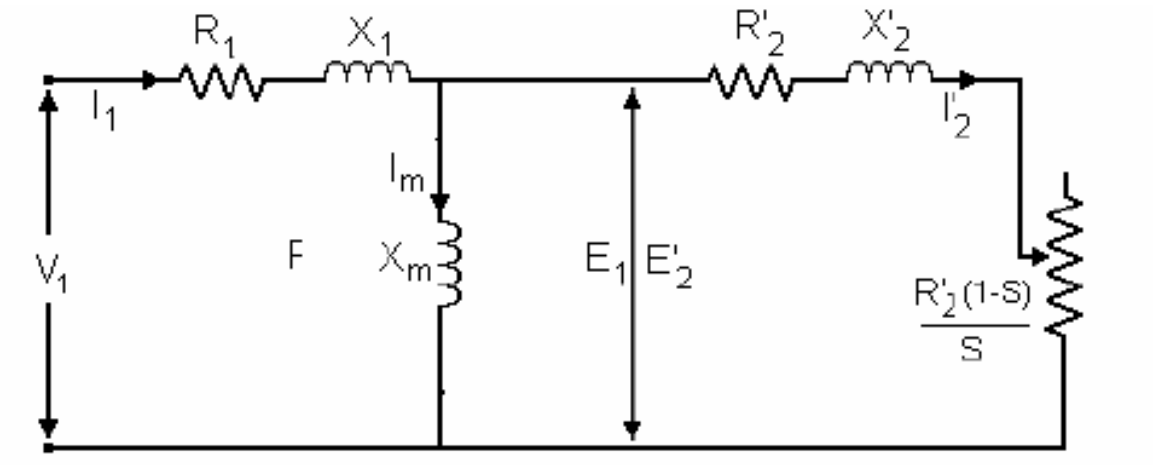


Figure 2.23: Approximate Equivalent circuit of induction motor.

In order to get the transfer function of the induction motor, the values of each of the resistors and inductors given in Figure 2.23 must be obtained [18].

Three separate tests can be performed to evaluate the circuit parameters:

DC resistance Test:

This dc resistance test applied to the stator terminal to measured and provides an estimate of the primary resistance R_1. This test is performed by setting ω_e equal to zero. This causes all the impedances caused by inductances to be zero at steady state according to

$$\text{Steady State Impedance} = Z = j*\omega_e*L$$

R_1 can then be evaluated by the equation below.

$$R_1(dc) = Vdc / Idc$$

The above value will be right if the test applied on one phase, but if the phase motor connected star or delta then to get the resistance per phase the getting value must be scaled as following:

If the motor connect in Star form then the value must divided by two.

If the motor connect in Delta form then the value must multiplied by 1.5.

Blocked-rotor Test:

The blocked rotor test is done by running the motor at zero speed ($\omega_r = 0$). The rotor is blocked to prevent rotation and balanced voltages are applied to the stator terminals at a frequency of 25 percent of the rated frequency at a voltage where the rated current is achieved. Current, voltage and power are measured at the motor input. This test will let us obtain the rotor resistance, R_2, and the leakage inductances of the rotor and stator, L_2 and L_1. Figure 2.2 below shows the equivalent blocked rotor circuit.

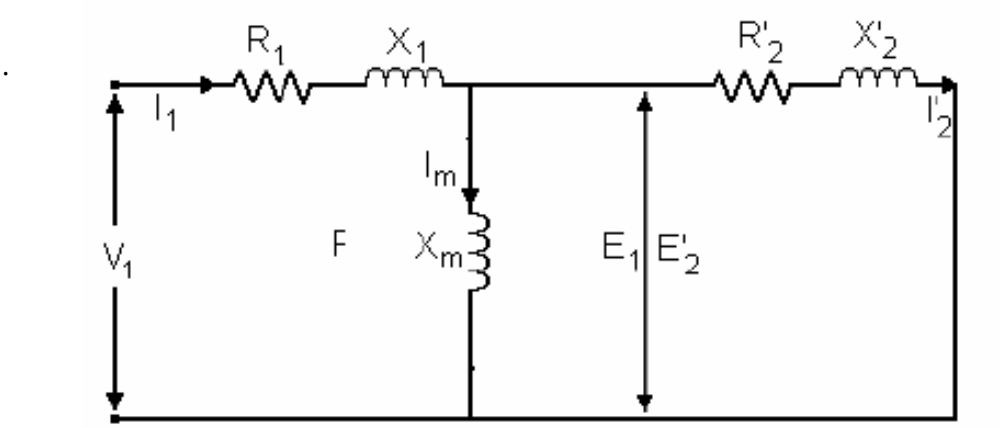


Figure2.24: Blocked-rotor test equivalent circuit of induction motor.

Because I_2 is much greater than the exciting current I_m, we can neglect the magnetizing branch, and assuming that $X_1 = X_2'$, the equivalent impedance of the circuit can be represented as below.

$$Zeq = V_1 / I_1 = R_1 + R_2' + 2X_1$$

And we can get the value of R_1 from the previous dc resistance test.

No-load test

This test will let us find the value of the mutual inductance, Lm then we can get X_m=J*W*L_m. During the No Load Test, we can assume that $\omega_e = \omega_r$. This corresponds to a slip of zero. A slip of zero also means that right loop of the circuit is now an open circuit as shown in Figure 2.25. The equivalent impedance of the circuit is then:

$$Z_{eq} = R_1 + X_1 + X_m$$

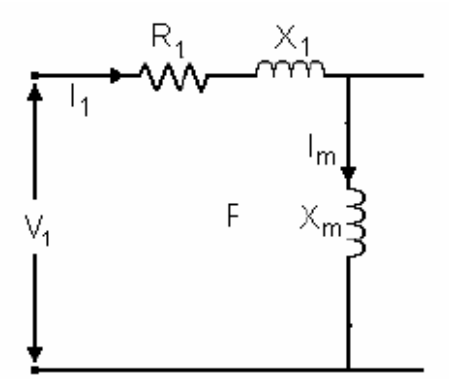


Figure 2.25: No-load test equivalent circuit of induction motor.

CHAPTER 3

Fuzzy Logic Control

3.1 Fuzzy logic history

The concept of fuzzy Logic (FL) was conceived by Lotfi Zadeh, a professor at the University of California at Berkley, who was published the first paper on fuzzy set theory in early 1960's [19], which was presented not as a control methodology, but as a way of processing data.

This approach to set theory was not applied to control systems until the 70's due to insufficient small-computer capability prior to that time. Professor Zadeh reasoned that people do not require precise, numerical information input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement [20].

However, during its early years, it was met with a lot of criticisms, some of which are from Prof. Zadeh's colleagues themselves. Rudolph E. Kalman had this to say in 1972: "I would like to comment briefly on Prof. Zadeh's presentation. His proposals could be severely, ferociously, even brutally criticized from a technical point of view. This would be out of place here. But a blunt question remains: Is Prof. Zadeh presenting important ideas or is he indulging in wishful thinking? No doubt Prof. Zadeh's enthusiasm for fuzziness has been reinforced by the prevailing climate in the U.S.A one of unprecedented permissiveness. 'Fuzzification' is a kind of scientific permissiveness; it tends to result in socially appealing slogans unaccompanied by the discipline of hard scientific work and patient observation." [21].

Similarly, his esteemed and brilliant colleague Prof. William, stated the following in 1975: "Fuzzy theory is wrong, wrong, and pernicious. I cannot think of any problem that could not be solved better by ordinary logic. What Zadeh is saying is the same sort of things: Technology got us into this mess and now it can't get us out. Well, technology did not get us into this mess. Greed and weakness and ambivalence got us into this mess. What we need is more logical thinking, not less. The danger of fuzzy theory is that it will encourage the sort of imprecise thinking that has brought us so much trouble.".

Unfortunately, U.S. manufacturers have not been so quick to deal with this technology while the Europeans and Japanese have been aggressively building real products around it. [21]. In 1974, Mamdani published the first paper for fuzzy applications [22]. Mamdani method was proposed as an attempt to control a real application in steam engine. The fuzzy inference system proposed by Mamdani, known as the Mamdani model in fuzzy system literature.

In 1985, Takagi and Sugeno published the paper of fuzzy systems [23]. The fuzzy inference system proposed by Takagi and Sugeno, known as the T-S model in fuzzy system literature.

There are several advantages of using fuzzy control over classical control methods. As Lotfi Zadeh, who is considered the father of fuzzy logic, once remarked: "In almost every case you can build the same product without fuzzy logic, but fuzzy is faster and cheaper." [19]. Japanese were the first to use fuzzy logic in application in 1980's. Japanese and Korean companies are using fuzzy logic to enhance things like computers, air conditioners, automobile parts, cameras, televisions, washing machines, and robotics. In 1994 Japan exported products using fuzzy logic totaling 35 billion dollar. Today, many publications discuss the theoretical background of fuzzy logic, its history, and how to program fuzzy logic algorithms.

3.2 Fuzzy logic

FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems [24]. It can be implemented in hardware, software, or a combination of both. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noise, or missing input information.

FL's approach to control problems mimics how a person would make decisions, only much faster FL incorporates a simple, rule-based IF X AND Y THEN Z approach to a solving control problem rather than attempting to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system. In other words fuzzy logic is used in system control and analysis design, because it shortens the time for engineering development and sometimes, in the case of highly complex systems, is the only way to solve the problem [25].

Before illustrating the mechanisms which make fuzzy logic machines work, it is important to realize what fuzzy logic actually is. Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth values between "completely true" and "completely false". As its name suggests, it is the logic underlying modes of reasoning which are approximate rather than exact. The importance of fuzzy logic derives from the fact that most modes of human reasoning and especially common sense reasoning are approximate in nature.

The essential characteristics of fuzzy logic as founded by Lotfi Zadeh are as follows:

- In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning.
- In fuzzy logic everything is a matter of degree.
- Any logical system can be Fuzzified.
- In fuzzy logic, knowledge is interpreted as a collection of elastic or, equivalently, fuzzy constraint on a collection of variables.
- Inference is viewed as a process of propagation of elastic constraints.

3.3 Fuzzy sets

General definition of a set is that a set is a collection of objects distinct and perfectly specified. A part of a set is a subset. For example, let E is a finite referential set:

$$E = \{a, b, c, d, e, f\}$$

It can form a crisp subset of E, for example:

$$A = \{b,d,f\}$$

If we present it in the other form:

A =	0	1	0	1	0	1
	a	*b*	*c*	*d*	*e*	*f*

In the classical set theory one element can either belong to a set, or not. This property can be represented by a degree of membership. In the case shown before, the element f belongs to A, and its degree of membership is 1.

The element c doesn't belong to A and its membership is 0. We can form a function which represents this property:

$$mA(x)=\begin{cases}1 & if \quad x \in A \\ 0 & if \quad x \notin A\end{cases} \tag{3.1}$$

This concept is basic in the classical set theory. The main concept of fuzzy theory is a notion of fuzzy set. Fuzzy set is an extension of crisp set. Zadeh was giving the following definition:

A fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between zero and one [25].

After Dr.Zadeh fuzzy set,, many authors found different ways of denoting fuzzy sets [26]. Zimmermann writes:

A fuzzy set is denoted by an ordered set of pairs, the first element of which denotes the element (x) and the second (μA(x)) the degree of membership:

$$A = \{(x, m_A(x)) \mid x \in X\} \tag{3.2}$$

where μ_A takes values in the interval [0,1].

One of the biggest differences between crisp and fuzzy sets is that the former always have unique membership functions, whereas every fuzzy set has an infinite number of membership functions that can represent it. From the above definitions follows that one possible fuzzy subset of the referential E is:

B =	0.8	0.4	0	0.2	1	1
	a	*b*	*c*	*d*	*e*	*f*

It means that the element *a* belongs to B with a value of 0.8, element *b* with 0.4 etc. This value has different names in the literature. The mostly used are: membership value, degree of membership, degree of compatibility, degree of truth, grade of membership, level of membership etc.

3.4 Membership function

Every fuzzy set can be represented by its membership function. The shape of membership function depends on the application and can be monotonic, triangular, trapezoidal or bell-shaped as shown in Figure 3.1.

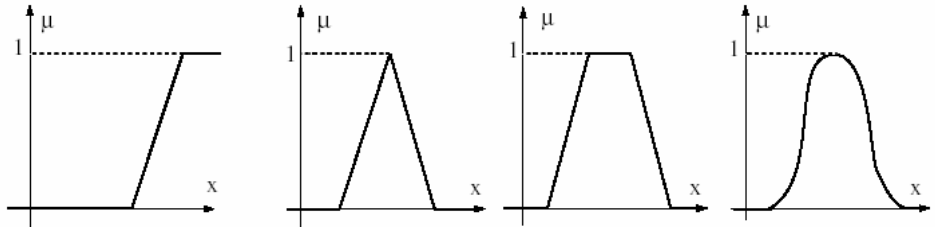


Figure 3.1: Different shapes of membership functions: monotonic, triangular, trapezoidal and bell-shaped.

For example to represent the property: positive small of the linguistic variable: temperature shown in Figure 3.2. If the measured temperature in one system is x, then the level of membership of x in the fuzzy set positive small is given by μ(x) and it is 0.8. We can say that the level of truth for the proposition:" The temperature x is positive small" is 0.8 or 80%.

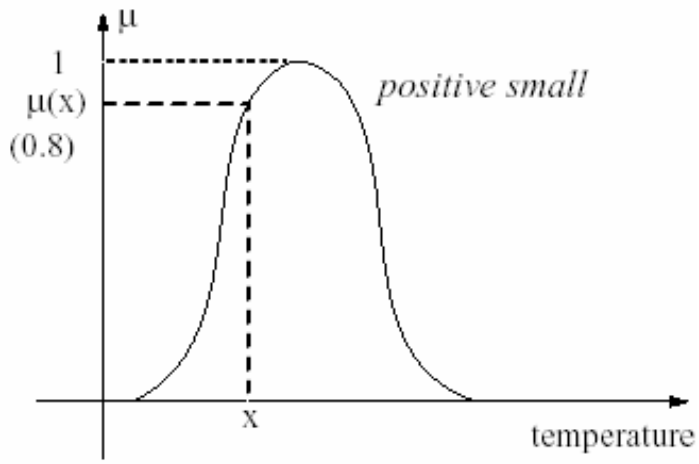


Figure 3.2: membership function example (positive small temperature).

One of the first steps in every fuzzy application is to define the universe of discourse (dynamic range) for every linguistic variable. The set of terms: T(temperature) can be characterized as fuzzy sets whose membership functions are shown in Figure 3.3. Every fuzzy set in a universe of discourse represents one linguistic value or label.

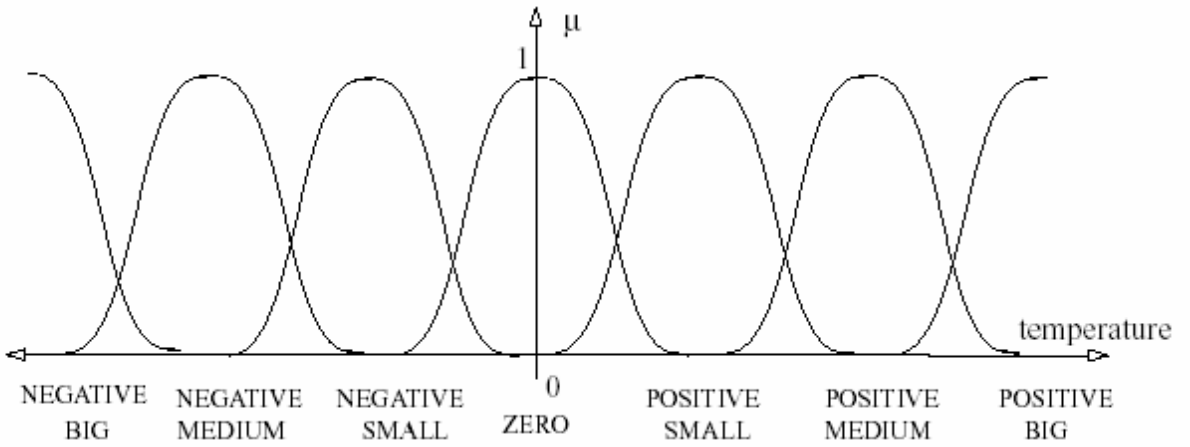


Figure 3.3: Universe of discourse for linguistic variable: temperature.

3.5 Operations with fuzzy sets

The most important operators in classical set theory with ordinary (crisp) sets are complement, intersection, union. These operations are defined in fuzzy logic via membership functions. Moreover, fuzzy set theory offers the vast range of operations on fuzzy sets that don't exist in the classical theory [28].

3. 5. 1 Complement

Complementation in fuzzy set theory corresponds to the complementation in classical set theory. For example, the element belongs to the fuzzy subset B with a level 0.6. It means that it does not belong to B with a level 0.4. Mathematically the membership values in a complement subset $\overline{B}$ are.

$$\mu_{\overline{B}}(x) = not\ (\mu\ B(x)) = 1 - B(x). \tag{3.3}$$

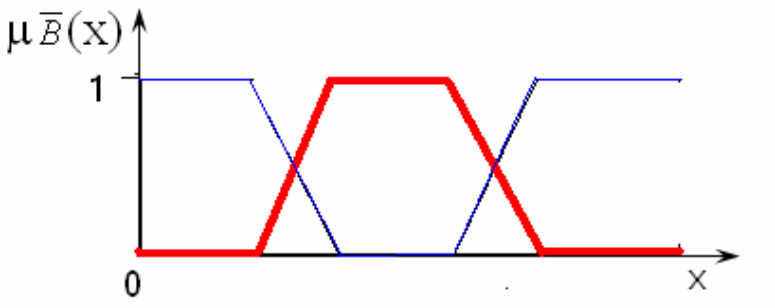


Figure 3.4: Complement of fuzzy sets B.

3. 5. 2 Intersection or triangular norms

For the intersection of fuzzy sets, Zadeh [28] suggested the min operator and the algebraic product. Following Zadeh's idea a lot of researchers proposed various operators for this operation [26]. Let A and B be two fuzzy sets in U universe of discourse, with membership functions μA and μB respectively. The most important intersection operators are:

- min operator

$$\mu A(x)\ and\ \mu B(x) = \min\ \{\ \mu A(x), \mu B(x)\} \tag{3.4}$$

- algebraic product

$$\mu A(x)\ and\ \mu B(x) = \mu A(x)^*\ \mu B(x) \tag{3.5}$$

- bounded product

$$\mu A(x)\ and\ \mu B(x) = \max\ (0, \mu A(x) + \mu B(x) - 1) \tag{3.6}$$

- drastic product

$$m_A\ and\ m_B = \begin{cases} x & if \quad y = 1 \\ y & if \quad x = 1 \\ 0 & if \quad x, y < 1 \end{cases} \tag{3.7}$$

- Einstein product

$$m_A \, and \, m_B = \frac{m_A(x) m_B(x)}{2 - [m_A(x) - m_B(x) - m_A(x) m_B(x)]} \tag{3.8}$$

- Hamacher product

$$m_A \, and \, m_B = \frac{m_A(x) m_B(x)}{m_A(x) + m_B(x) - m_A(x) m_B(x)]} \tag{3.9}$$

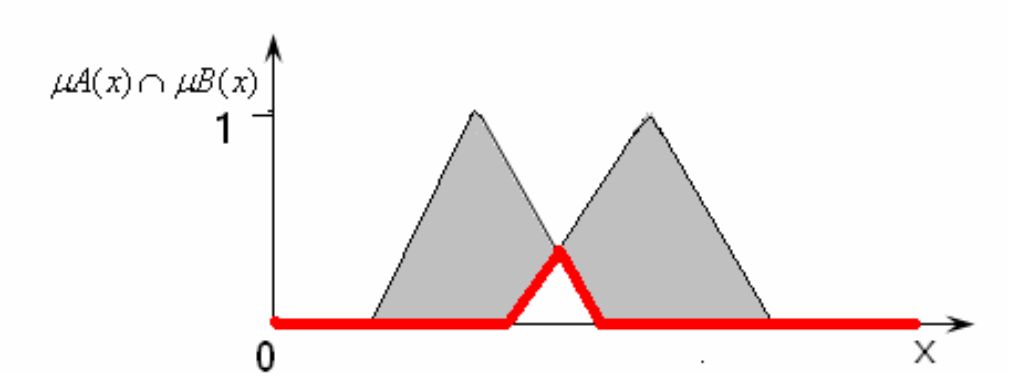


Figure 3.5: Intersection of fuzzy sets A and B(most used).

3. 5.3 Union triangular

For the union of two fuzzy sets, the most used in the literature are:

- max operator

μA(x) *or* μB(x) = max {μA(x), μB(x)} (3.10)

- algebraic sum

μA(x) *or* μB(x) = μA(x) μB(x)- μA(x)* μB(x) (3.11)

- bounded sum

μA(x) *or* μB(x) = min { 1, μA(x)+μB(x) } (3.12)

- drastic sum

$$m_A \, or \, m_B = \begin{cases} x & if \quad y = 0 \\ y & if \quad x = 0 \\ 1 & if \quad x, y > 0 \end{cases} \tag{3.13}$$

- Einstein sum

$$m_A \, or \, m_B = \frac{m_A(x) + m_B(x)}{1 + m_A(x) m_B(x)} \tag{3.14}$$

- Hamacher sum

$$m_A \, or \, m_B = \frac{m_A(x) + m_B(x) - 2m_A(x) m_B(x)}{1 - m_A(x) m_B(x)} \tag{3.15}$$

- Disjoint sum

$$\mu A(x) \; or \; \mu B(x) = \max \{\min (\mu A(x), 1 - \mu B(x)), \min(1 - \mu A(x), \mu B(x)) \} \tag{3.16}$$

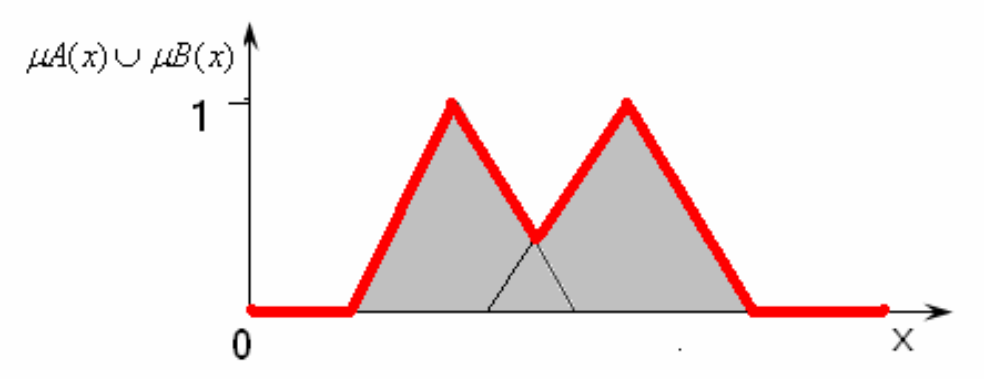


Figure 3.6: Union Intersection of fuzzy sets A and B(most used).

3.6 Notion of linguistic rule

The principal idea of fuzzy logic systems is to express the human knowledge in the form of linguistic if-then rules. Every rule has two parts:

• Antecedent part (premise), expressed by if... and

• Consequent part, expressed by: then...

The antecedent part is the description of the state of the system, and the consequent is the action that the operator who controls the system must take. There are several forms of if-then rules. The general is:

If (a set of conditions is satisfied) then (a set of consequences can be inferred).

Example: If the temperature is high, then the pressure is small.

The general form of this rule is:

Rule: If x is A, then y is B.

Temperature (x) and pressure (y) are linguistic variables. x represents the state of the system, and y is control variable and represents the action of the operator. High (A) and small (B) are linguistic values or labels characterized by appropriate membership

functions of fuzzy sets. They are defined in the universe of discourse of the linguistic variables x and y.

Takagi and Sugeno [26] proposed the form which has the fuzzy sets only in the premise part of the rule, and the consequent part is described by a non-fuzzy equation of the input variable.

Example: If velocity is high, then force is $k*(velocity)^2$

Another form of this rule is:

Rule: If x is A, then y is $k*x^2$.

or more general,

Rule: If x is A, then y is f(x)

3.7 General structure of fuzzy logic control "FLC" system

Every fuzzy system is composed of four principal blocks (Figure 3.7):

1. Knowledge base (rules and parameters for membership functions).

2. Decision making unit (inference operations on the rules).

3. Fuzzification interface (transformation of the crisp inputs into degrees of match with linguistic variables).

4. Defuzzification interface (transformation of the fuzzy result of the inference into a crisp output).

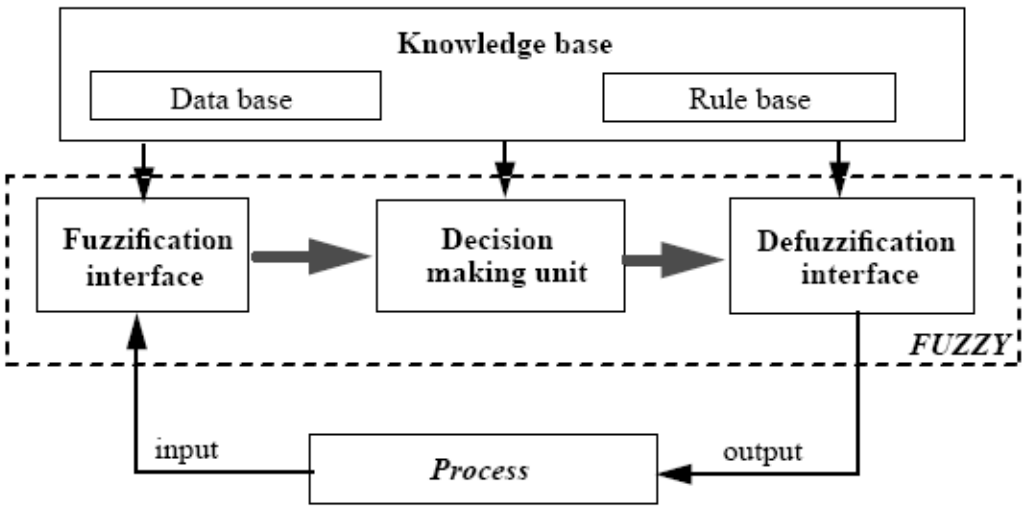


Figure 3.7: General structure of fuzzy inference system.

3.7.1 Knowledge base

We can use four modes of derivation of fuzzy control rules. These four modes are not mutually exclusive, and it is necessary to combine them to obtain an effective system.

- Expert experience and control engineering knowledge: operating manual and questionnaire.
- Based on operators' control actions: observation of human controller's actions in terms of input-output operating data.
- Based on the fuzzy model of a process: linguistic description of the dynamic characteristics of a process.
- Based on learning: ability to modify control rules such as self-organizing controller.

The number of base rules depends on the number of membership in the fuzzy set of the inputs. For example if the system contains one input with fuzzy set contains 4 memberships then there are 4 base rules. If the system has two inputs and one of them contains five membership in its fuzzy set, and the other contains three memberships then the total base rules will equal to 5*3=15 base rule. The efficiency of the system will proportionally depend on the number of membership, but here the system is going more complex to implement. So there are many studies in genetic algorithm which try to minimizing the base rules and hence to simplify the system calculations.

3.7.2 Procedure of fuzzy inference

There are a lot of inference methods which deals with fuzzy inference like : Mamdani method, Larsen method, Tsukamoto method, and the Sugeno style inference, or to be more complete, Takagi-Sugeno_Kang (TSK) method. The most important and widely used in fuzzy controllers are the Mamdani and Takagi-Sugeno methods.

3.7.2.a Mamdani method

Which is the most commonly used fuzzy inference technique. In 1974, Professor Ebrahim Mamdani of London University built one of the first fuzzy systems to control a steam engine and boiler combination. He applied a set of fuzzy rules supplied by experienced human operators. The Mamdani-style fuzzy inference process is performed in four steps [23]:

- Fuzzification of the input variables,.
- Rule evaluation.
- Aggregation of the rule outputs.
- Defuzzification.

To illustrate the fuzzy inference let's examine a simple two-input one-output problem that includes three rules:

Rule(1)…. IF X is A_3 OR Y is B_1 THEN z is C_1

Rule(2)…. IF X is A_2 AND Y is B_2 THEN z is C_2

Rule(3)…. IF X is A_1 THEN z is C_3

Step 1: Fuzzification

The first step in the application of fuzzy reasoning is a Fuzzification of inputs in the controller, which is to take the crisp inputs, x1 and y1, and determine the degree to which these inputs belong to each of the appropriate fuzzy sets. It means that to every crisp value of input we attribute a set of degrees of membership (mj, j=1,n) to fuzzy sets defined in the universe of discourse for that input.

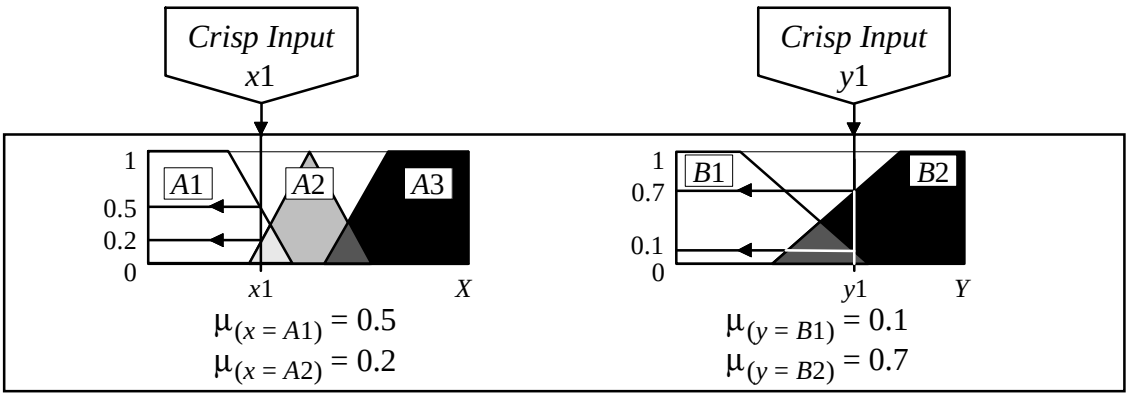


Figure 3.8: Fuzzification stage

Step 2: Rule evaluation

The second step is to take the Fuzzified inputs, $\mu(x=A1) = 0.5$, $\mu(x=A2) = 0.2$, $\mu(y=B1) = 0.1$ and $\mu(y=B2) = 0.7$, and apply them to the antecedents of the fuzzy rules. If a given fuzzy rule has multiple antecedents, the fuzzy operator (AND or OR) is used to obtain a single number that represents the result of the antecedent evaluation. This number (the truth value) is then applied to the consequent membership function. To evaluate the disjunction of the rule antecedents, we use the OR fuzzy operation. As shown Operations with fuzzy sets the most used approach for the union is to get the maximum:

$$\mu A \cup B(x) = max\,[\mu A(x), \mu B(x)] \tag{3.17}$$

Similarly, in order to evaluate the conjunction of the rule antecedents, we apply the AND fuzzy operation intersection which used minimum approach:

$$\mu A \cap B(x) = min\,[\mu A(x), \mu B(x)] \tag{3.18}$$

the rule evaluations are clearly appears in Figure 3.9.

The most common method of correlating the rule consequent with the truth value of the rule antecedent is to cut the consequent membership function at the level of the antecedent truth. This method is called clipping. Since the top of the membership function is sliced, the clipped fuzzy set loses some information. However, clipping is still often preferred because it involves less complex and faster mathematics, and generates an aggregated output surface that is easier to Defuzzify.

While clipping is a frequently used method, scaling offers a better approach for preserving the original shape of the fuzzy set. The original membership function of the rule consequent is adjusted by multiplying all its membership degrees by the truth value of the rule antecedent. This method shown in Figure 3.10, which generally loses less information, can be very useful in fuzzy expert systems

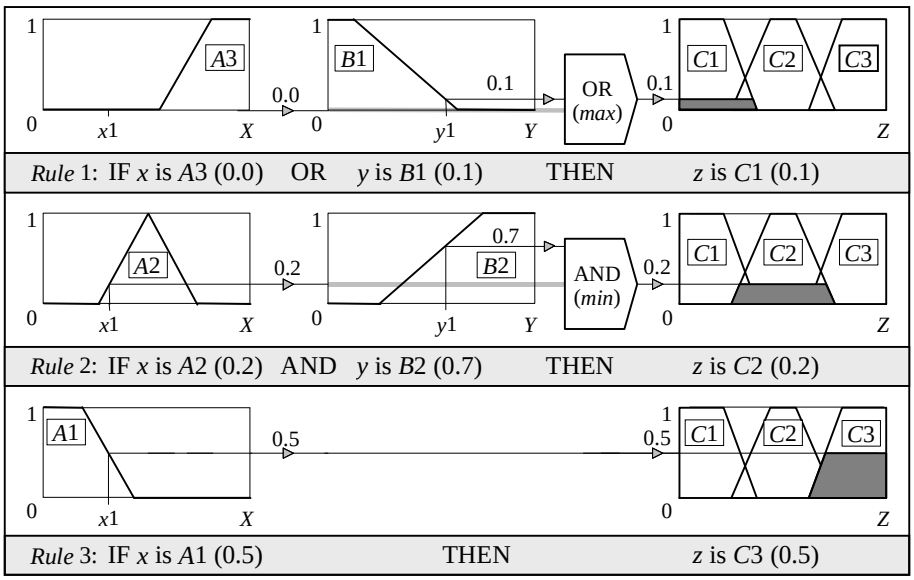


Figure 3.9: Rule evaluation in Mamdani method

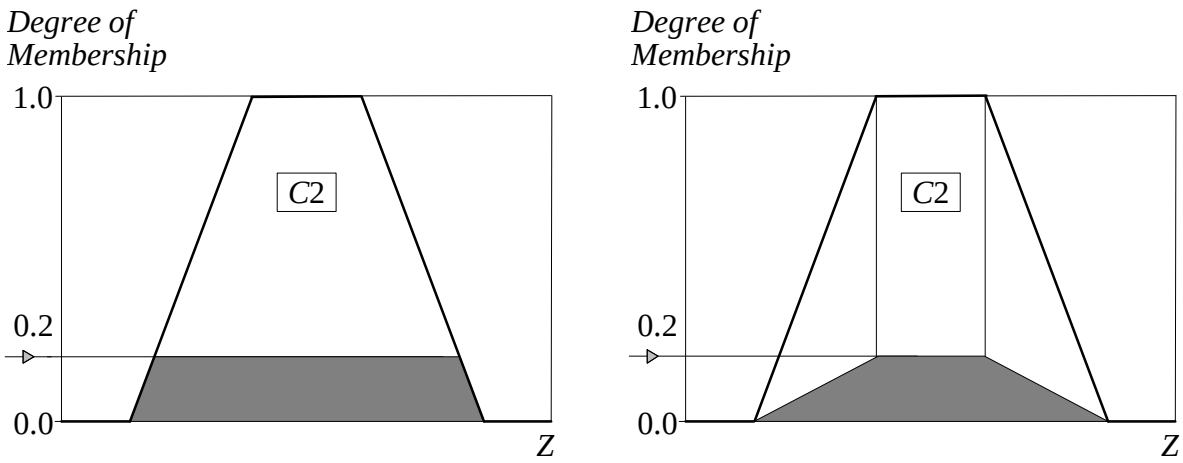


Figure 3.10: Clipping and scaling stage.

Step 3: Aggregation of the rule outputs

Aggregation is the process of unification of the outputs of all rules. We take the membership functions of all rule consequents previously clipped or scaled and combine them into a single fuzzy set. The input of the aggregation process is the list of clipped or scaled consequent membership functions, and the output is one fuzzy set for each output variable.

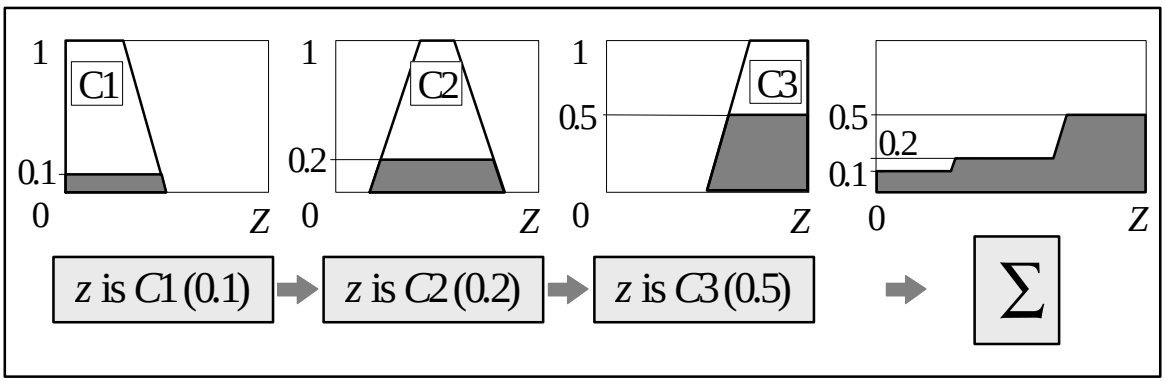


Figure 3.11: Aggregation stage in Mamdani method.

Step 4: Defuzzification

The last step in the fuzzy inference process is Defuzzification. Fuzziness helps us to evaluate the rules, but the final output of a fuzzy system has to be a crisp number. The input for the Defuzzification process is the aggregate output fuzzy set and the output is a single number. There are several methods for the Defuzzification, proposed in the literature. Here are four of them [27].

- **The center of gravity method**

This widely used method generates a center of gravity (or center of area) also called centroid technique of the resulting fuzzy set of a control action. If we discretize the universe it is:

$$Z = \frac{\sum_{i=1}^{n} r_i Z_i}{\sum_{i=1}^{n} Z_i} \tag{3.19}$$

Where n is the number of quantization levels, r_i is the amount of control output at the quantization level i and Z_i represents its membership value.

- **The mean of maximum method**

The mean of maxima method generates a crisp control action by averaging the support values which their membership values reach the maximum. In the case of discrete universe:

$$Z = \sum_{i=1}^{l} \frac{Z_i}{l} \tag{3.20}$$

Where *l* is the number of the quantized *r* values which reach their maximum memberships.

- **Tsukamoto's method**

If monotonic membership functions are used, then the crisp control action can be calculated as follows:

$$Z = \frac{\sum_{i=1}^{n} W_i Z_i}{\sum_{i=1}^{n} W_i} \tag{3.21}$$

Where *n* is the number of rules with firing strength *wi* is grater than zero and *zi* is the amount of control action recommended by the rule *i*.

- **The weighted average method**

This method is used when the fuzzy control rules are the functions of their inputs.

In general, the consequent part of the rule is:

$z = f(x,y)$ If W_i is the firing strength of the rule *i*, then the crisp value is given by:

$$Z = \frac{\sum_{i=1}^{n} W_i f(x_i, y_i)}{\sum_{i=1}^{n} W_i} \tag{3.22}$$

where *n* is the number of firing rules.

The most popular method of Defuzzification is the Centroid technique. It finds the point where a vertical line would slice the aggregate set into two equal masses. Mathematically this centre of gravity (COG) can be expressed as:

$$COG = \frac{\int_a^b m_A(x) x dx}{\int_a^b m_A(x) dx} \tag{3.23}$$

A reasonable estimate can be obtained by calculating it over a sample of points.

$$COG = \frac{(0+10+20)\times 0.1+(30+40+50+60)\times 0.2+(70+80+90+100)\times 0.5}{0.1+0.1+0.1+0.2+0.2+0.2+0.2+0.5+0.5+0.5+0.5} = 67.4$$

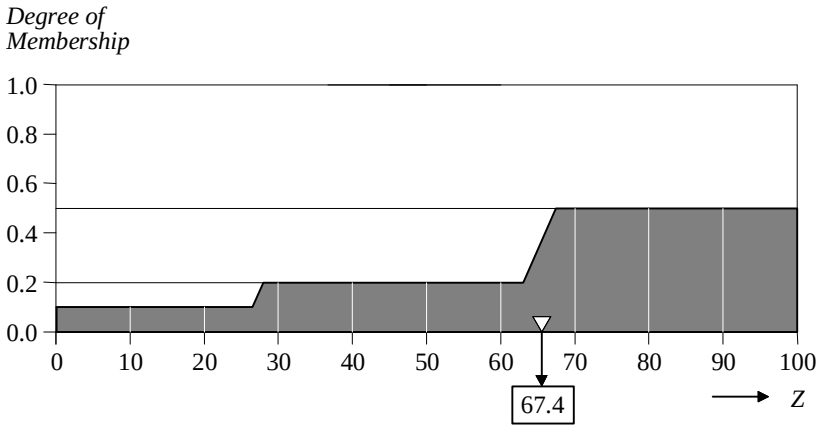


Figure 3.12: COG approach in Defuzzification stage.

Mamdani-style inference, as we have just seen, requires us to find the centroid of a two-dimensional shape by integrating across a continuously varying function. In general, this process is not computationally efficient [27].

3.7.2.b Sugeno method

Since Mamdani's pioneering work [28] on fuzzy control motivated by zadeh's approach to inexact [29], there have been numerous studies on fuzzy reasoning [30,31]. Most fuzzy controllers have been designed, based on human operator experience and/or control engineer knowledge. It is; however, often the case that an operator cannot tell linguistically what kind of action he takes in a particular situation. In this respect, it is quite useful to provide a method of modeling the control actions using numerical data [32]. In 1985 Takagi-Sugeno-Kang suggested to use a single spike, a singleton, as the membership function of the rule consequent, and they suggested another approach that using equation consequent in place off singleton consequent. A singleton, or more precisely a fuzzy singleton, is a fuzzy set with a membership function that is unity at a single particular point on the universe of discourse and zero everywhere else. Sugeno-style fuzzy inference is very similar to the Mamdani method. Sugeno changed only a rule consequent. Instead of a fuzzy set, he used a mathematical function of the input variable. The format of the Sugeno-style fuzzy rule is

IF X is A AND Y is B THEN Z is $f(x, y)$ (3.24)

where X, Y and Z are linguistic variables; A and B are fuzzy sets on universe of discourses X and Y, respectively; and $f(x, y)$ is a mathematical function.

The most commonly used zero-order Sugeno fuzzy model applies fuzzy rules in the following form:

IF X is *A* AND *Y* is *B* THEN Z is k (3.25)

Where *k* is a constant.

In this case, the output of each fuzzy rule is constant. All consequent membership functions are represented by singleton spikes.

The following Figures (3.13,14,15) illustrate the idea for TSK which like Mamadni steps.

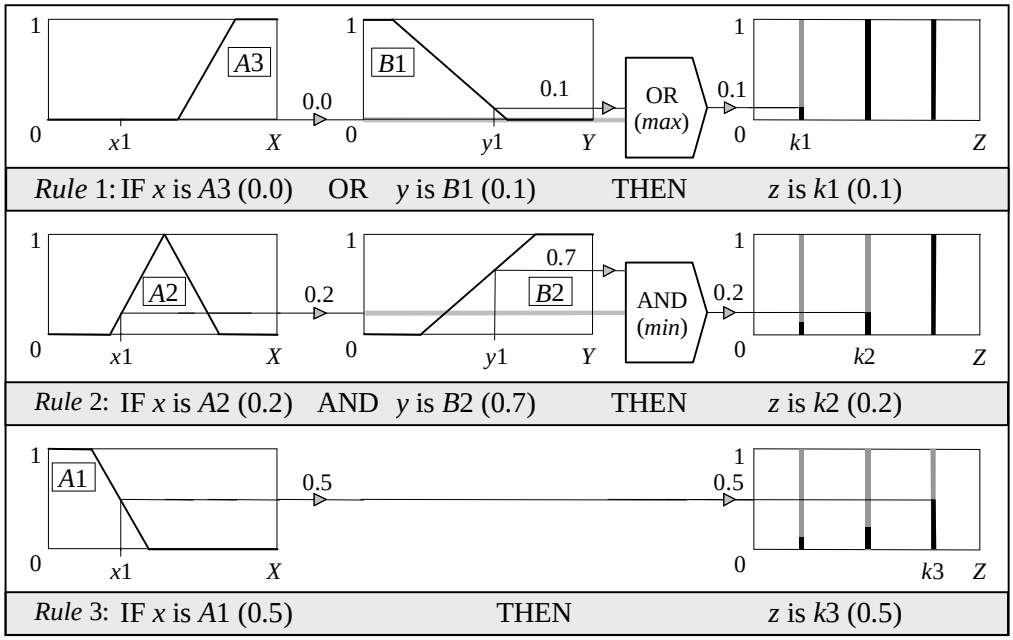


Figure 3.13: Rule evaluation stage in TSK method.

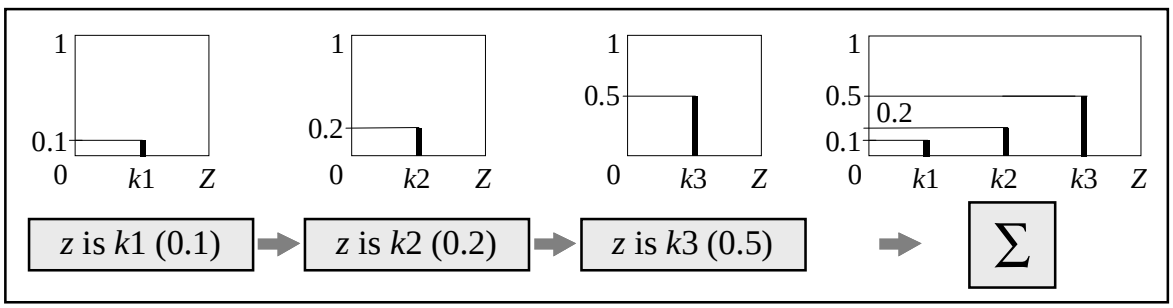


Figure 3.14: Aggregation stage in TSK method.

For Defuzzification stage its better to use Weighted Average method (WA)

$$WA = \frac{m(k1)\times k1 + m(k2)\times k2 + m(k3)\times k3}{m(k1) + m(k2) + m(k3)} = \frac{0.1\times 20 + 0.2\times 50 + 0.5\times 80}{0.1 + 0.2 + 0.5} = 65$$

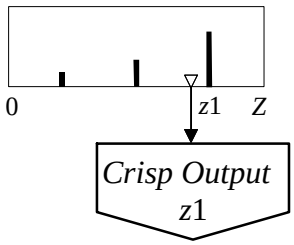


Figure 3.15: (WA) method in Defuzzification stage.

The overall fuzzy logic controller "FLC" appear in Figure 3.16

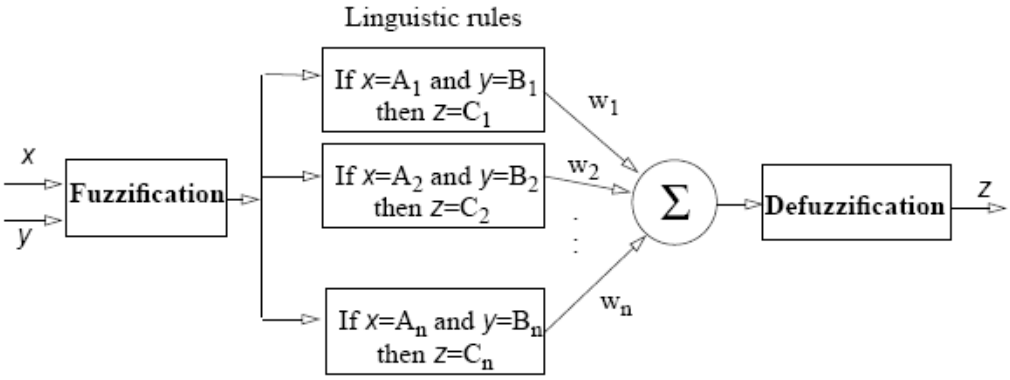


Figure 3.16: General structure of fuzzy logic control part of the system.

3.7.2.c How to make a decision Mamdani or Sugeno?

- Mamdani method is widely accepted for capturing expert knowledge. It allows us to describe the expertise in more intuitive, more human-like manner. However, Mamdani-type fuzzy inference entails a substantial computational burden.
- On the other hand, Sugeno method is computationally effective and works well with optimization and adaptive techniques, which makes it very attractive in control problems, particularly for dynamic nonlinear systems.

CHAPTER 4

Field Programmable Gate Arrays (FPGAs)

4.1 Introduction

FPGAs stand for Field Programmable Gate Arrays are one type of programmable logic devices (PLDs). They are an integrated circuit that can be configured by the user in order to implement digital logic functions of varying complexities. FPGAs can be very effectively used for control purposes in processes demanding very high loop cycle time. One of the fundamental advantage of FPGA over DSP or other microprocessors is the freedom of programming parallelism. Since different parts of FPGA can be configured to perform independent functions simultaneously, its performance is just not tied to clock rate as in DSPs. This fact enables FPGA‘s to score over general purpose computing chips in the digital control systems implementation [32].

4.2 PLDs history

By the late 1970s, standard logic devices like AND, OR, NAND, and others basic gates with printed circuit boards loaded with them were the rage techniques in electronics design. Then a novel idea has been appeared which provided designers with the ability to implement different interconnections in a bigger device. This would allow designers to integrate many standard logic devices into one part. To offer the ultimate in design flexibility, Ron Cline from Signetics™ (which was later purchased by Philips and then eventually Xilinx) came up with the idea of the programmable logic device.

A programmable logic device or PLD is an electronic component used to build reconfigurable digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture, and before the PLD can be used in a circuit it must be programmed. The first type of PLD family which had been appeared in the market is called programmable logic array (PLA). PLAs have two programmable planes as illustrated in Figure 4.1. These two planes provided any combination of “AND” and “OR” gates, as well as sharing of AND terms across multiple ORs. This architecture

was very flexible, but at the time wafer geometries of 10 µm made the input-to-output delay (or propagation delay time T_{pd}) high, which made the devices relatively slow [33].

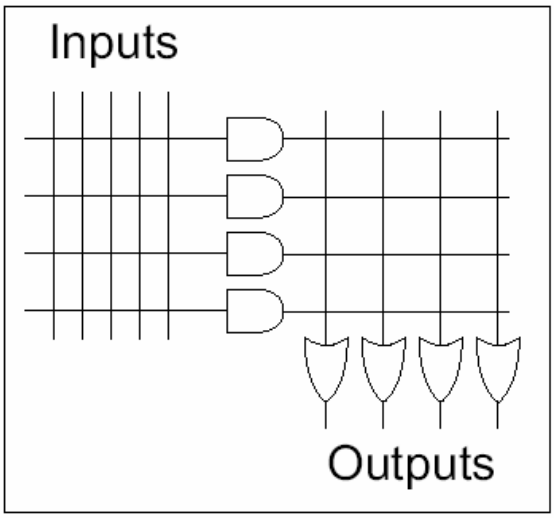


Figure 4.1: PLA constructions.

After fabrication issues applied to PLA, it was modified to become the programmable array logic (PAL). This new architecture shown in Figure 4.2 differed from in that one of the programmable planes (OR array) was fixed. PAL architecture also had the added benefit of faster T_{pd} and less complex software, but without the flexibility of the PLA structure. This category of PLD devices is often called Simple PLD or SPLD.

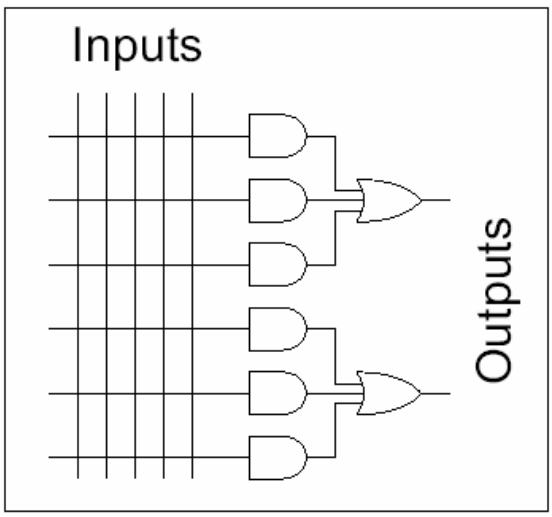


Figure 4.2: PAL constructions.

The architecture had a mesh of horizontal and vertical interconnect tracks. At each junction was a fuse. With the aid of software tools, designers could select which junctions would not be connected by "blowing" all unwanted fuses. This process was done by a device programmer. Input pins were connected to the vertical interconnect. The horizontal tracks were connected to AND-OR gates, also called "product terms". These in turn

connected to dedicated flip-flops, whose outputs were connected to output pins. SPLDs provided as much as 50 times more gates in a single package than discrete logic devices. PLD technology has moved on from the early days with companies such as Xilinx producing ultra-low-power CMOS devices based on flash memory technology. Flash PLDs provide the ability to program the devices time and time again, electrically programming and erasing the device [34,35].

Complex Programmable Logic Devices (CPLD) which shown in Figure 4.3 are another way to extend the density of the simple PLDs. The concept is to have a few PLD blocks or macrocells on a single device with general purpose interconnect in between. Simple logic paths can be implemented within a single block. More sophisticated logic will require multiple blocks and use the general purpose interconnect in between to make these connections.

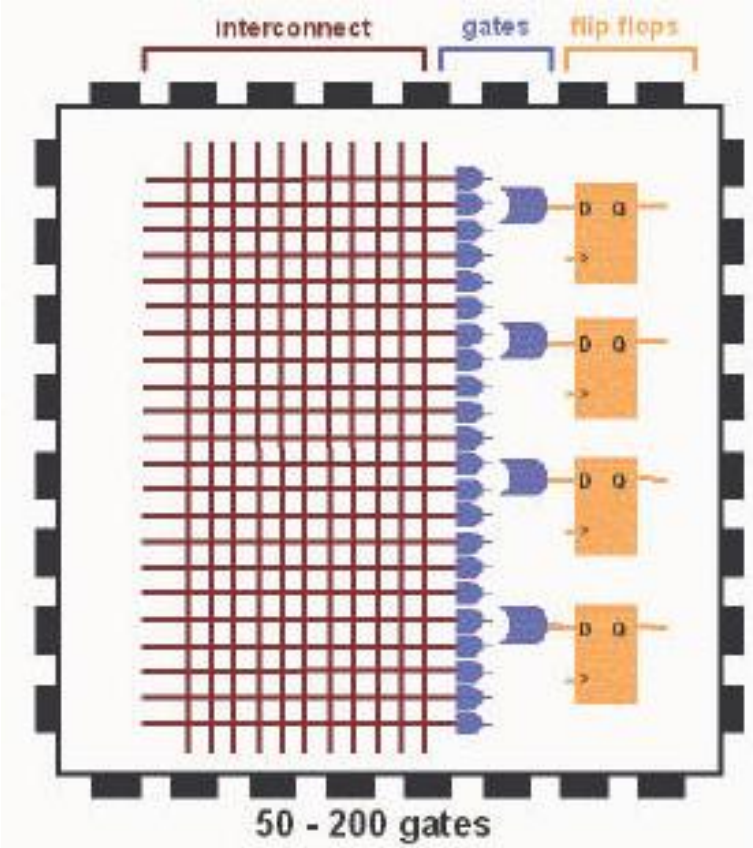


Figure 4.3: CPLD architecture.

In 1985, Xilinx company introduced a completely new idea. The concept was to combine the user control and time to market of PLDs with the densities and cost benefits of gate arrays. A lot of customers liked it, and the FPGA was born. The term is most commonly applied to electronic devices which contain an array of identical logic elements which can be configured using a programming procedure to replicate any particular logic circuit.

4.3 FPGAs construction

A normal FPGA device is usually contained within a single silicon package which may also house some form of memory elements [36]. As shown in Figure 4.4 the logic element has a programmable Look-Up Table and a register (flip-flop).

The LUT can perform any logic function on the available inputs to produce a single logic output. The final output is either this new value or the previous value (stored in the flip-flop), although the logic element may have more than the four inputs.

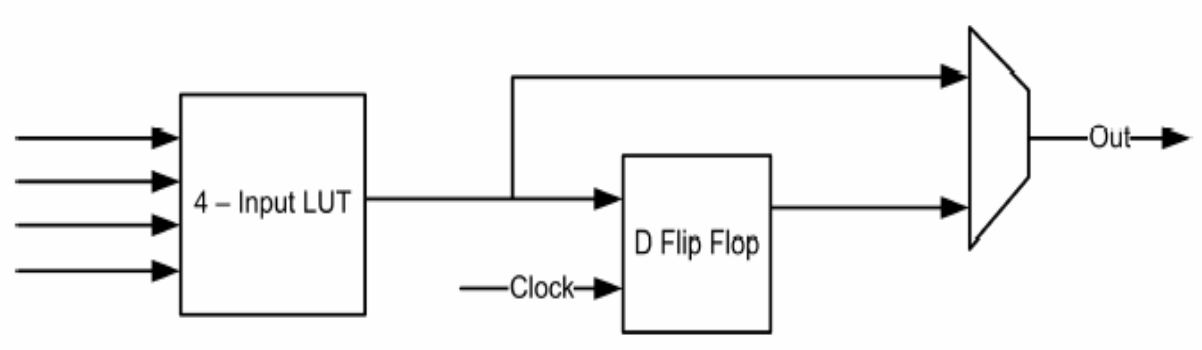


Figure 4.4: FPGA logic element

Figure 4.5 demonstrates the way in which logic blocks are laid out to form an array inside the FPGA. Different Manufactures will have slightly different configurations but all use some type of programmable switch matrix at the crossing point of the logic block interconnection lines.

By having programmable switches and programmable logic elements, the system can be configured to mimic any combination of logic functions as long as the overall design can be fitted into the available number of logic elements and switches.

There are two basic types of FPGAs: SRAM-based reprogrammable and One-time programmable (OTP). These two types of FPGAs differ in the implementation of the logic element and the mechanism used to make connections in the device.

The dominant type of FPGA is SRAM-based and can be reprogrammed by the user as often as the user chooses. In fact, an SRAM FPGA is reprogrammed every time it is powered-up.

That's why you need a serial programmable read only memory (SPROM) or system memory with every SRAM FPGA. One-time programmable (OTP) FPGAs use anti-fuses to make permanent connections in the chip and so do not require a SPROM or other means to download the program to the FPGA. However, every time you make a design change, you must throw away the chip.

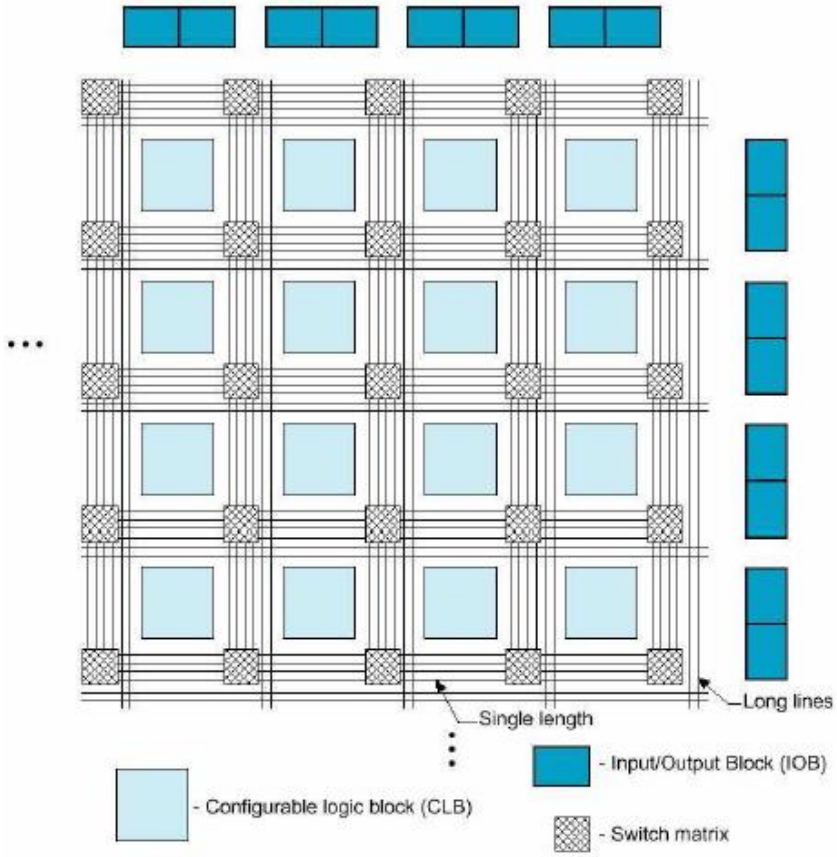


Figure 4.5: Structure of a Xilinx FPGA standard.

4.4 ASIC vs FPGA as a design choice

In comparison with PLDs, there is another kind of un-programmable digital devices called application specific integrated circuit (ASIC). An ASIC is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. By carefully tuning each ASIC to a given job, the computer designer can produce a smaller, cheaper, faster chip that consumes less power than a programmable processor. A graphics chip for a personal computer (PC), for instance, can draw lines or paint pictures on the screen 10 or 100 times as quickly as a general-purpose central processing unit can. The ASIC must be fabricated on a manufacturing line, a process that takes several months, before it can be used or even tested. ASIC have some significant advantages because they are designed for a particular purpose they are very fast, efficient circuitry, and lower cost for high volume production. The disadvantages that it takes time for the ASIC vendor to manufacture and test the parts and the process of designing, testing and setting up fabrication facilities for the production of an ASIC is generally very expensive. In situations where the market for a certain device is large and reprogrammability is not needed, this high non-recurring

engineering (NRE) cost can be countered by the smaller, faster and cheaper end product that ASIC technology produces. Strictly speaking, a working FPGA device programmed with a hardware image is in fact a type of ASIC, however the FPGA's reprogrammability makes it very different as a design choice. The advantages and disadvantages of both types of technology are summed up in Table 4.1.

Table 4.1: ASIC and FPGA comparison.

	ASIC	FPGA
NRE Cost	High	Low
Unit Cost	Less Expensive	More Expensive
Speed	Faster	Slower
Re-Configurable in the field	No	Yes
Footprint	Smaller	Larger
Time to Market	Longer	Shorter
Cost of Debugging	High	Low

4.5 FPGAs design advantages

FPGAs enable ease of design, lower development costs, and more product revenue for money, and the opportunity to speed products to market.

- **Ease of Design:** FPGAs offer the simplest way to implement a design. Once a design has been described, it is simply use software development tools to optimize, fit, and simulate the design.

- **Lower Development Costs:** FPGAs offer very low development costs. Because its reprogrammablity, it's easily and very inexpensively to change designs. This allows optimizing designs and continuing to add new features to enhance products.

- **More Product Revenue:** FPGAs offer very short development cycles, which means the product will getting market quicker and begin generating revenue sooner. Again due to its reprogrammablity, products can be easily modified. This in turn allows easily introducing additional features and quickly generating new revenue.

- **Reduced Board Area:** FPGAs offer a high level of integration (that is, a large number of system gates per area) and are available in very small form factor packages. This provides the perfect solution for designers whose products which must fit into small enclosures or who have a limited amount of circuit board space to implement the logic design [36].

4.6 Parallel processing

Of the many advantages that FPGA devices offer, the ability to allow for customized parallel processing is perhaps the most beneficial of all. Since a designer no longer needs to rely on a ASIC vendor to provide him with the processing tools that he needs, the designer can quickly create his own processing blocks in hardware. This is hugely beneficial for computationally strenuous tasks, and since the cost of trial and error designing is now essentially nil, the designer is free to experiment with different processing configurations to fine tune his system. The basic concept is demonstrated in Figure 4.6. The Figure shows how an FPGA can be used to quadruple the speed of digital processing using existing software to defined DSP core, along with a control core. For example, let us assume that the original DSP core can perform a given operation in 4 clock cycles. This gives an output of 1/4 = 0.25 operations per clock cycle. In the configuration at Figure 4.6, the control core switches the data input and outputs in a rotational fashion thereby allowing a new input value to be applied to the inputs of a different DSP core every clock cycle. Thus the DSP blocks will operate in parallel on a single sequential stream of incoming data. Our resulting performance is 4/4 = 1 Operations per clock cycle [37].

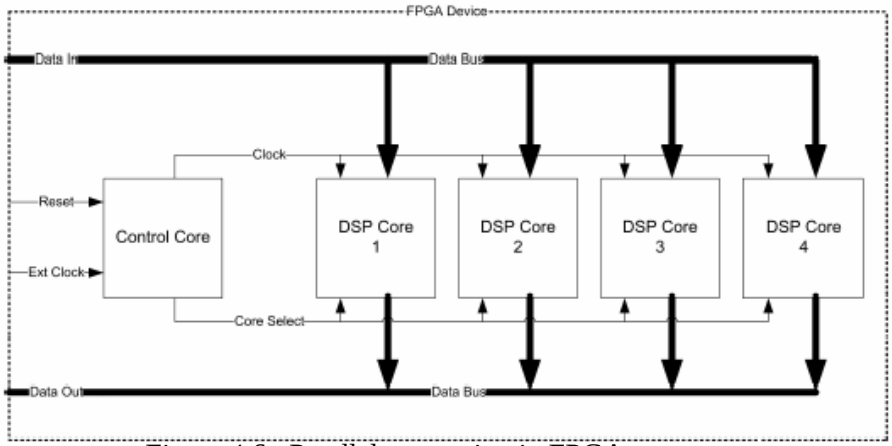


Figure 4.6: Parallel processing in FPGA.

4.7 Xilinx design software package

Xilinx offers complete electronic design tools that enable the implementation of designs in Xilinx PLDs. These development solutions combine powerful technology with a flexible, easy-to-use graphical interface regardless of any experience level. The availability of products such as WebPACK ISE software has made it much easier to design with programmable logic. Designs can be described easily and quickly using a description language such as ABEL, VHDL, Verilog™, or with a schematic capture package.

Schematic capture is the traditional method that designers have used to specify gate arrays and programmable logic devices. It is a graphical tool that allows the user to specify the exact gates required and how they connected. There are four basic steps to using schematic capture:

1. After selecting a specific schematic capture tool and device library, begin building the circuit by loading the desired gates from the selected library. It can be used any combination of needed gates that you need.

2. Connecting the gates together using nets or wires.

3. Add and label the input and output buffers. These will define the I/O package pins for the device.

4. Generate a netlist which is a text equivalent of the circuit. It is generated by design tools such as a schematic capture program. The netlist is a compact way for other programs to understand what gates are in the circuit, how they are connected, and the names of the I/O pins.

Figure 4.7 describe an illustrative example, the netlist reflects the actual syntax of the circuit in the schematic. There is one line for each of the components and one line for each of the nets. The computer assigns names to components (G1 to G4) and to the nets (N1 to N8). When implementing this design, it will have input package pins A, B, C, and D, and output pins Q, R, and S.

The above example is obviously very simplistic. Let's describe a more realistic design of 10,000 equivalent gates. The typical schematic page contains about 200 gates, contained with soft macros.

Therefore, it would require 50 schematic pages to create a 10,000-gate design! Each page needs to go through all the steps mentioned previously which are: adding components, interconnecting the gates, adding I/Os, and generating a netlist.

This is rather time-consuming, especially if it is required to have a 20,000, 50,000, or even larger design. Another inherent problem with using schematic capture is the difficulty in migrating between vendors and technologies.

If initially we create 10,000 gates design with FPGA vendor X and then want to migrate to a gate array, then it will required to modify every one of those 50 pages using the gate array vendor's component library.

.

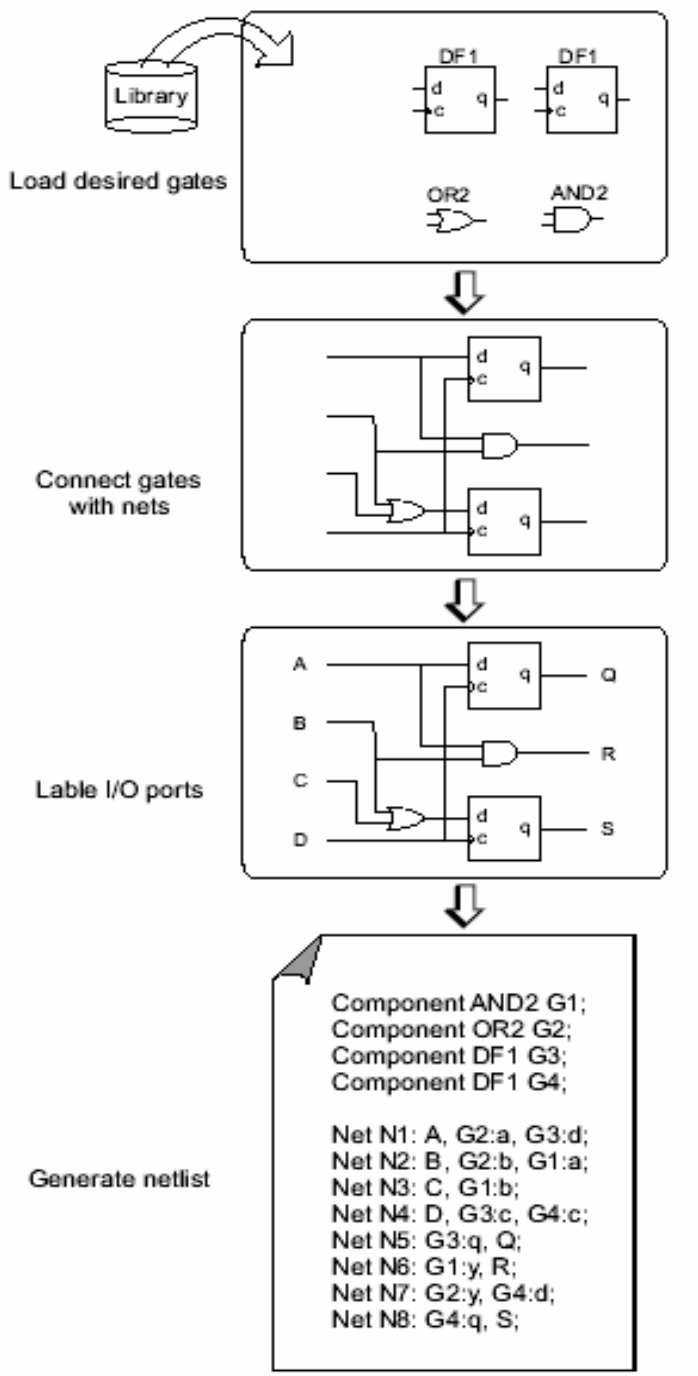


Figure 4.7: Design flow in schematic method.

There has to be a better way, and, of course, there is. It is called high-level design (HLD), behavioral, or hardware description language (HDL). For our purposes, these three terms are essentially the same thing. The idea is to use a high-level language to describe the circuit in a text file rather than a graphical low-level gate description. There are many programming language illustrate HDL and the mostly famous is VHDL (Very High Speed Integrated Circuits) [36]. As an example, let's consider the design work needed specifying a 16 x 16 multiplier with schematic capture or an VHDL file. A multiplier is a regular but

complex arrangement of adders and registers that requires quite a few gates. This example has two 16-bit inputs (A and B) and a 32-bit product output (Y = A x B) for a total of 64 I/Os. This circuit requires approximately 6,000 equivalent gates. In the schematic implementation, the required gates would have to be loaded, positioned on the page, and interconnected, with I/O buffers added. That would be about three days of work. The VHDL implementation shown in Figure 4.8 , which is also 6,000 gates, requires eight lines of text and can be done in three minutes. This file contains all the information necessary to define the 16 x 16 multiplier. In addition to the tremendous time savings, the VHDL method is completely vendor-independent. This opens up tremendous design possibilities for engineers. To create a 32 x 32 multiplier, it could simply modify the work which already done for the smaller multiplier. For the schematic approach, this would entail making three copies of the 30 pages, then figuring out where to edit the 90 pages so that they addressed the larger bus widths. This would probably require four hours of graphical editing.

16 × 16 Multiplier

```
 1  entity MULT is
 2    Port (A: in std_logic_vector(15 downto 0);
 3          B: in std_logic_vector(15 downto 0);
 4          Y: out std_logic_vector(31 downto 0));
 5  end MULT;
 6
 7  architecture Behavioral of MULT is
 8  begin
 9    Y<=A*B;
10  end Behavioral;
```

32 × 32 Multiplier

```
 1  entity MULT is
 2    Port (A: in std_logic_vector(31 downto 0);
 3          B: in std_logic_vector(31 downto 0);
 4          Y: out std_logic_vector(63 downto 0));
 5  end MULT;
 6
 7  architecture Behavioral of MULT is
 8  begin
 9    Y<=A*B;
10  end Behavioral;
```

Figure 4.8: Multiplier code in VHDL.

Figure 4.9 illustrate the design flow of the multiplication example in schematic and VHDL approaches. Once we have specified the design in a behavioral description we can convert It into gates using the process of synthesis.

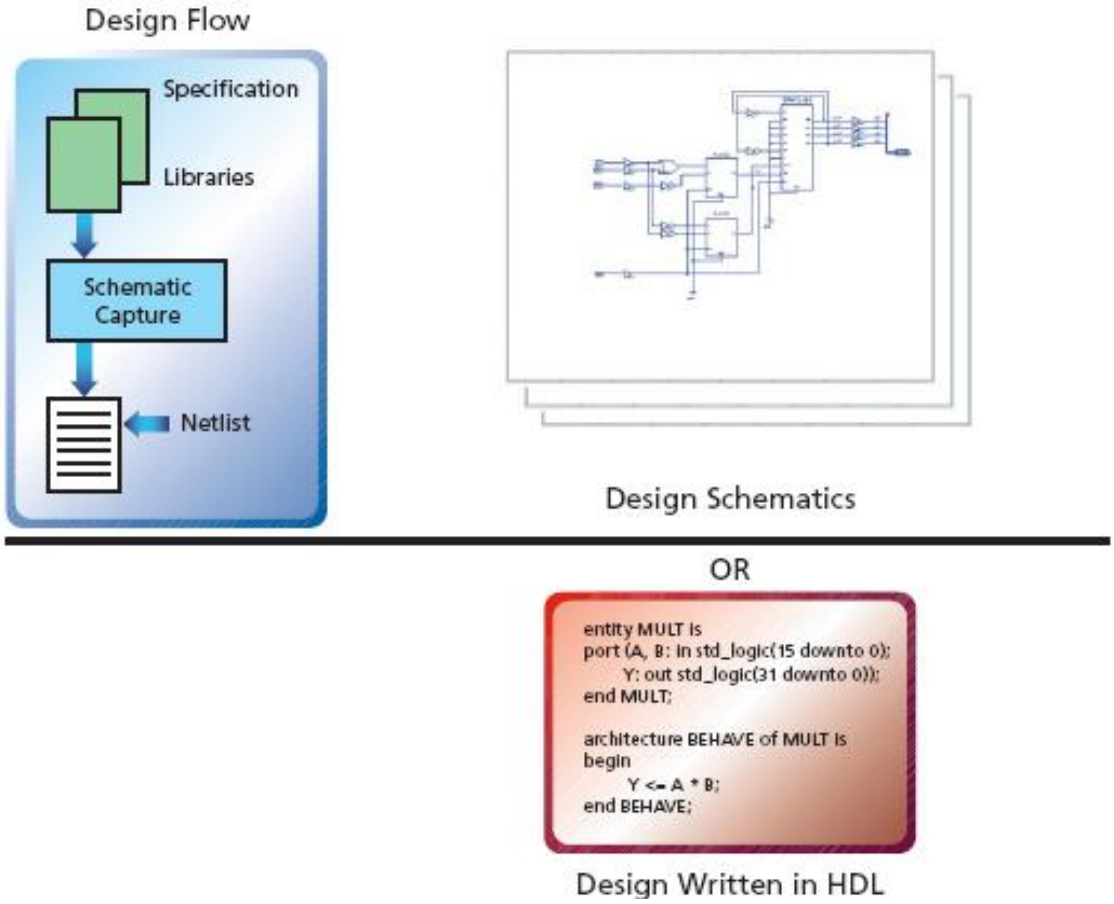


Figure 4.9: Design flow in both schematic and VHDL approaches.

The next step from schematics or synthesis approaches is the verification done by using a simulator, which is a software program that confirms the functionality or timing of a circuit. The last step is the implementation, which contains five stages:

- Translate: This Stage translates comprise various programs used to import the design netlist and prepare it for layout.

- Fitting: Meaning to "fit" the design to the target device.

- Place: Is the process of selecting specific modules, or logic blocks, in the FPGAs where design gates will reside.
- Route: As the name implies, is the physical routing of the interconnection between the logic blocks.

Most vendors provide automatic place and route tools so that we don't have to worry about the intricate details of the device architecture.

- Download program to FPGA chip: using USB cable or special programmer device depending on the kind of the FPGA chip and its board.

Figure 4.10 illustrates the overall procedure to implement the design using FPGAs .

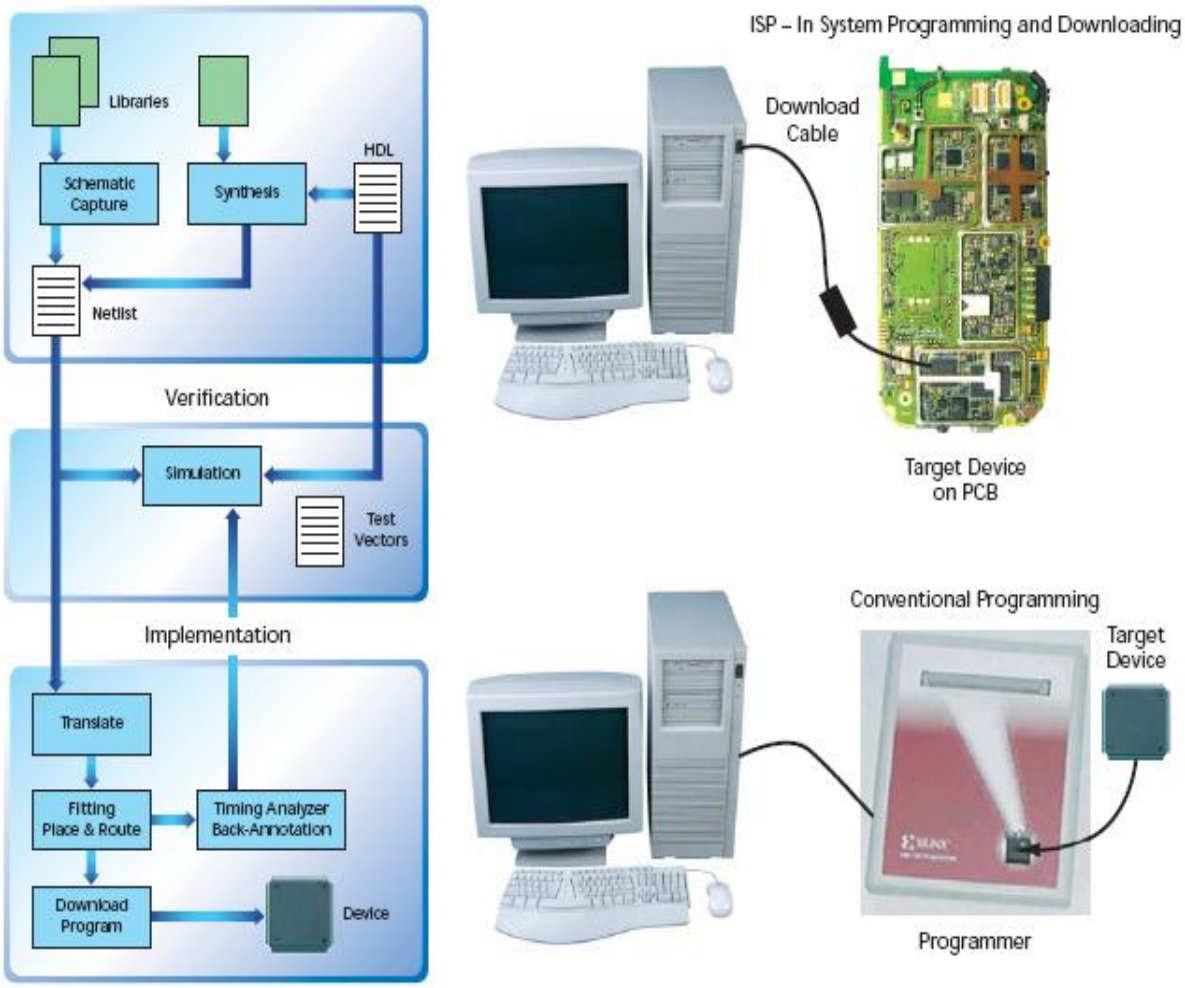


Figure 4.10: Overall process to program FPGAs.

4.8 Spartan-3A Starter Kit Board user guide

Xilinx Spartan-3A FPGAs (Figure 4.11) are ideal for low-cost, high-volume applications and are targeted as replacements for fixed-logic gate arrays. The combination of low cost and some improvements features makes it an ideal replacement for ASICs.

Depending on that and after features checking for the needs of my project in this thesis, I decided to select this board. Also it's memorable to mention that this board is very modern due to its manufacturing date (February 15, 2007) [37].

Figure 4.11 Spartan-3A Starter Kit Board.

This family of cards Spartan-3 FPGA has many applications in our real life, for example, in a car multimedia system shown in Figure 4.12 could absorb many system functions, including embedded IP cores, custom system interfaces, DSP, and logic.

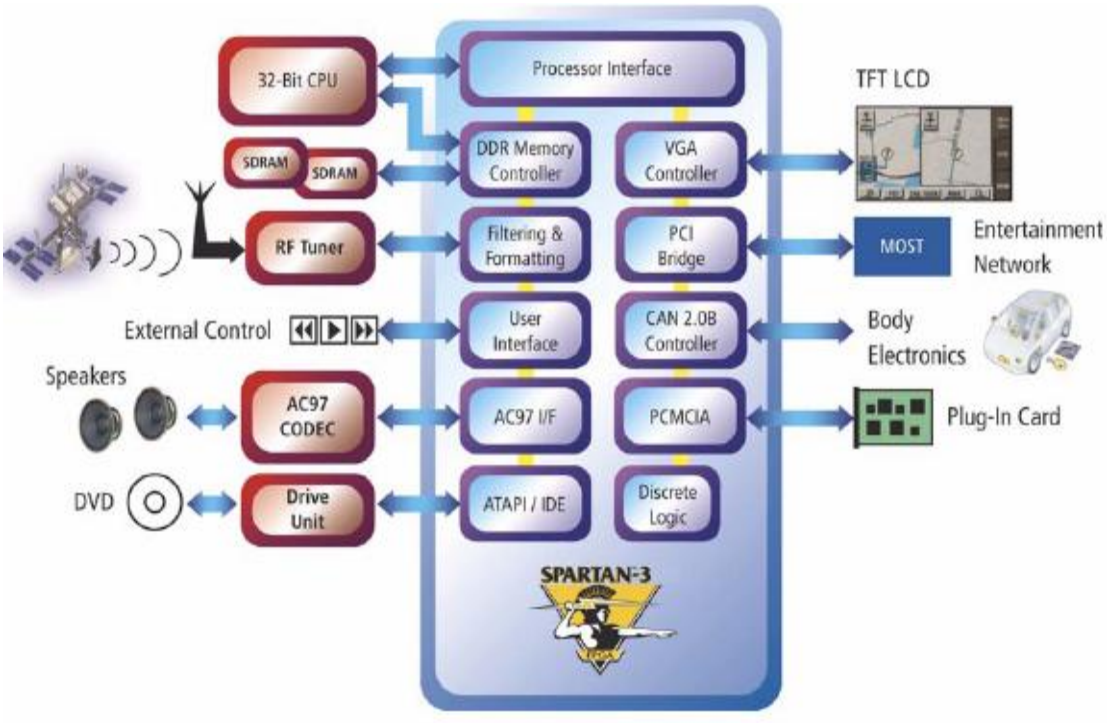


Figure 4.12 Spartan -3 in car applications.

The key features of the Spartan-3A Starter Kit are available at the Appendix A, but here are some of them:

- Xilinx 700K-gate XC3S700A Spartan-3A FPGA .
- 16 Mbits of SPI serial Flash
- . Two-line, 16-character LCD screen.
- PS/2 mouse or keyboard port
- VGA display port
- 10/100 Ethernet
- Two nine-pin RS-232 ports (DTE- and DCE-style)
- On-board USB-based programming solution
- FPGA download/debug
- 50 MHz clock oscillator
- High-speed differential I/O connectors
- Receiver: Five data channels plus clock
- Transmitter: Five data channels plus clock
- Supports multiple differential I/O standards
- Supports up to 24 single-ended I/O
- Uses widely available 34-conductor cables
- Four-output, Digital-to-Analog Converter (DAC)
- Two-input, Analog-to-Digital Converter (ADC) with programmable-gain
- pre-amplifier
- Stereo audio jack using digital I/O pins
- Rotary-encoder with push-button shaft
- Eight discrete LEDs
- Four slide switches
- Four push-button switches

4.9 PicoBlaze processor

Due to the complexity of some programming application in FPGA like video display , sound processing, network application, lcd etc. Xilinx Company provide The PicoBlaze™ microcontroller which is a compact, capable, and cost-effective fully embedded 8-bit RISC microcontroller core optimized for the Spartan™-3, and some others FPGA

families. The PicoBlaze software microcontroller provides cost-efficient microcontroller-based control and simple data processing. Designing with the PicoBlaze means instantiating it in the design using VHDL. An external assembler is then used to generate the code that is downloaded to the block RAM. One of the outputs of the assembler is a VHDL file that defines the BRAM and its contents. This is added as a source to the project and that's all there is to it. The PicoBlaze microcontroller is optimized for efficiency and low deployment cost. It occupies just 96 FPGA slices, or about 2% of an XC3S700 FPGA. In typical implementations, a single FPGA block RAM stores up to 1024 program instructions, which are automatically loaded during FPGA configuration. Even with such resource efficiency, the PicoBlaze microcontroller performs a respectable 44 to 100 million instructions per second (MIPS) depending on the target FPGA family and speed grade. The PicoBlaze microcontroller core is totally embedded within the target FPGA and requires no external resources. The PicoBlaze microcontroller is extremely flexible. The basic functionality is easily extended and enhanced by connecting additional FPGA logic to the microcontroller's input and output ports [38].

As shown in the block diagram in Figure 4.13, the PicoBlaze microcontroller supports the following features:

- 16 byte-wide general-purpose data registers.
- 1K instructions of programmable on-chip program store, automatically loaded during FPGA configuration.
- Byte-wide Arithmetic Logic Unit (ALU) with CARRY and ZERO indicator flags.
- 64-byte internal scratchpad RAM.
- 256 input and 256 output ports for easy expansion and enhancement.
- Automatic 31-location CALL/RETURN stack.
- Predictable performance, always two clock cycles per instruction, up to 200 MHz or 100 MIPS in a Virtex-II Pro FPGA.

- Fast interrupt response; worst-case 5 clock cycles.

- Optimized for Xilinx Spartan-3, Virtex-II, and Virtex-II Pro FPGA architectures—just 96 slices and 0.5 to 1 block RAM.

- Assembler, instruction-set simulator support.

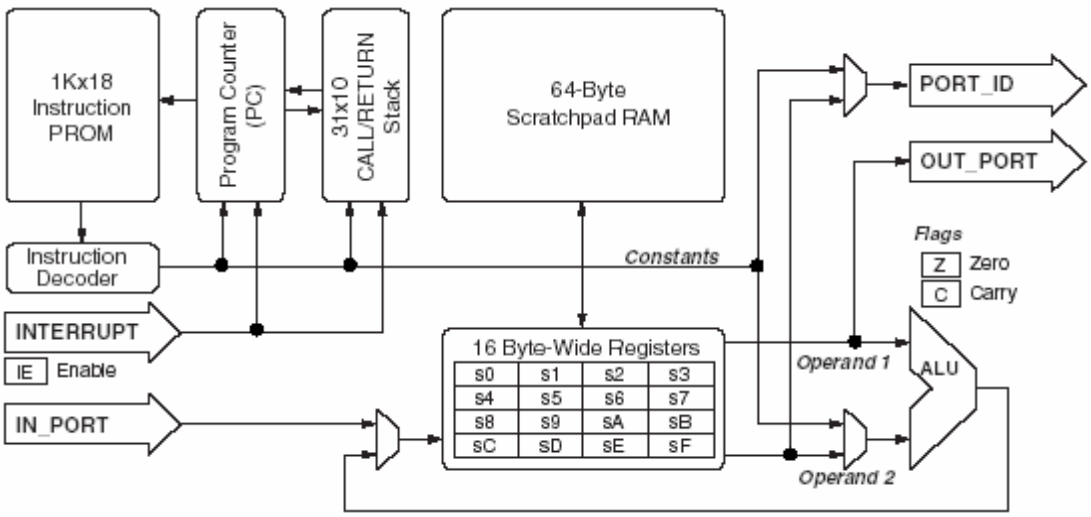


Figure 4.13 PicoBlaze architecture.

CHAPTER 5

Designing Controllers Using FPGA for Three-phase Induction Motor

5.1 Overall system design and implementation

This chapter presents the process used to design three kinds of widely used controllers (PID, Fuzzy, and Fuzzy-PID) for the three phase induction motor using FPGA technique. The block diagram explains the concept of this system is shown in Figure 5.1.

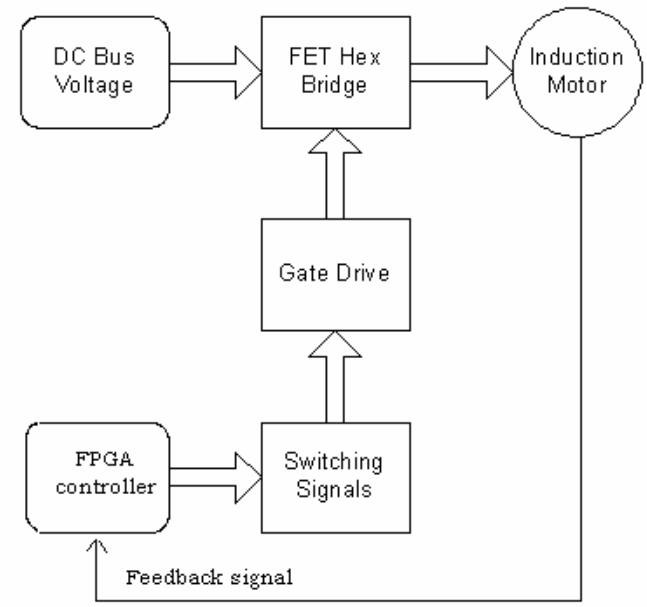


Figure 5.1: Block diagram for the over all control system.

5.2 Three-phase induction motor inverter

When power is supplied to an induction motor at the recommended specifications, it runs at its rated speed. However, many applications need variable speed operations. For example, a three-phase induction motor in the elevator must use different speeds for each order needs to travel between floors. Historically, mechanical gear systems were used to obtain variable speed. Recently, electronic power and control systems have matured to allow these components to be used for motor control in place of mechanical gears. These electronics not only control the motor's speed, but can

improve the motor's dynamic and steady state characteristics [39]. In addition, electronics can reduce the system's average power consumption and noise generation of the motor. While there are different methods for control, Variable Voltage Variable Frequency (VVVF) or V/*f* is the most common method of speed control. As has been mentioned in chapter 2 it is needed to change the frequency of the input power supply of the motor for getting variable speeds, and at same time the percentage V/f must be constant to keep the delivered torque to the load constant during speed varying . This means that the motor can no longer be supplied directly from the AC power line if there are needs to change its speed.

5.2.1 Hardware construction of the inverter

If the incoming AC supply were converted to DC using a rectifier as shown in Figure 5-2, it could then use some kind of switching power inverter to generate an oscillation of needed frequency.

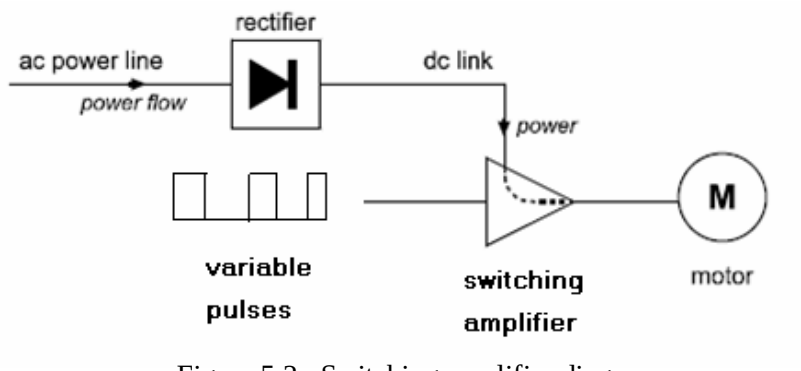


Figure 5.2: Switching amplifier diagram.

The rectifier circuit can be one-phase or three-phase bridge rectifier as shown in Figure (5.3-5.4) According to the kind of the input power supply used in the system.

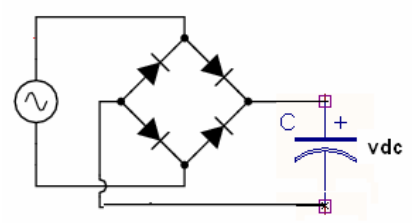


Figure 5.3: One-phase rectifier bridge with ripple capacitor.

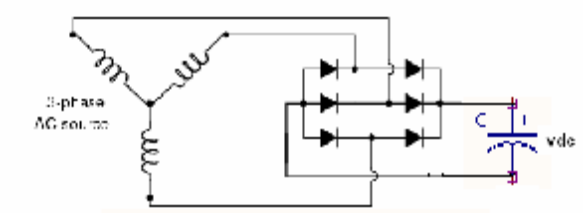


Figure 5.4: Three-phase rectifier bridge with ripple capacitor.

The input and output for each rectifier is shown in Figure 5.5 and Figure 5.6

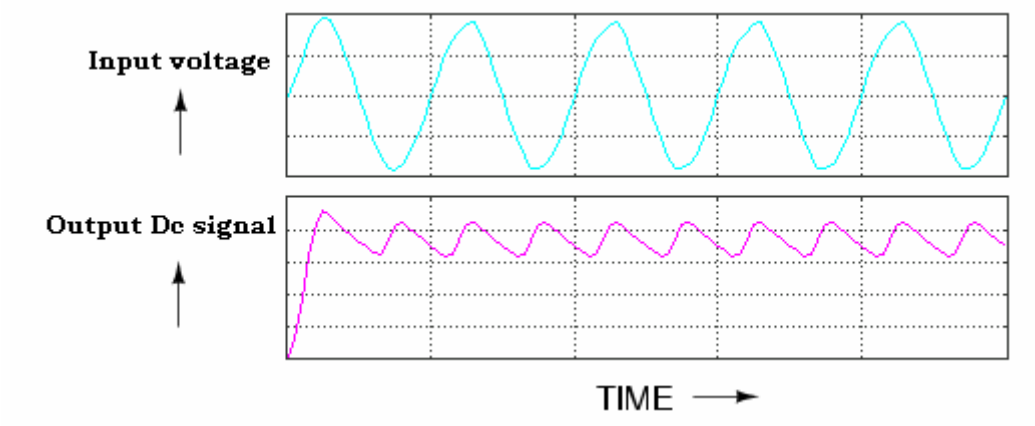


Figure 5.5: Input and output signal for the one phase bridge rectifier.

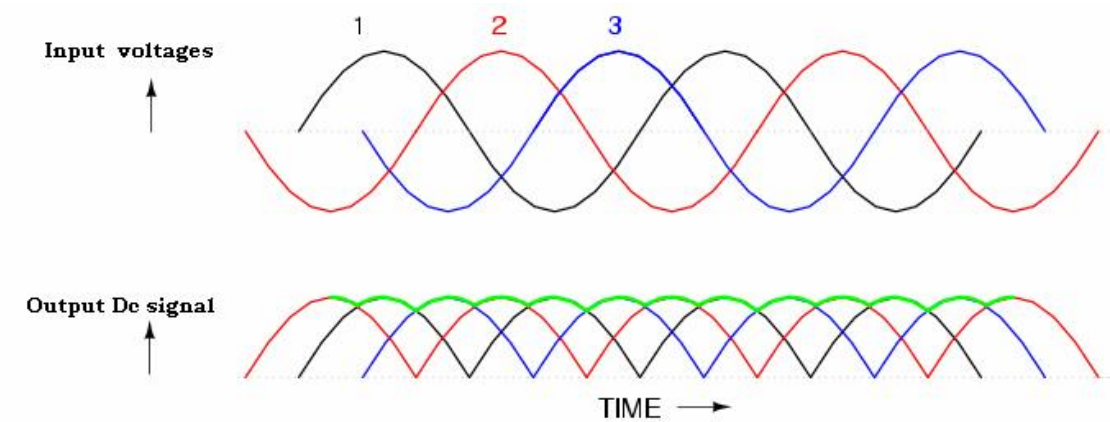


Figure 5.6: Input and output signal for the three phase bridge rectifier.

For a given load, a larger capacitor will reduce ripple but will cost more and will create higher peak currents in the supply feeding it. In Figure 5.7, the voltage waveform of capacitors is depicted to calculate corresponding capacitance value.

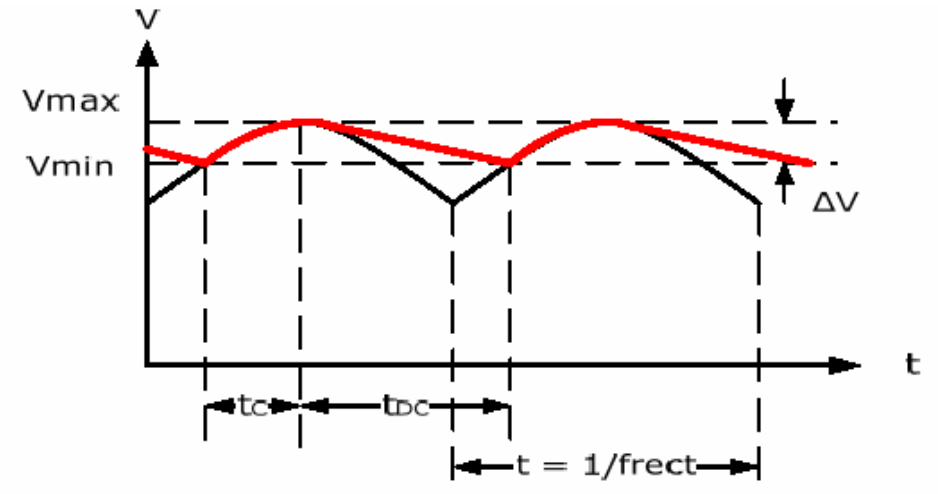


Figure 5.7: Ripple factor for the full wave rectifier.

Electrolytic capacitors are used to smooth the dc bus voltage. Its capacitance can be found from the formula:

$$C_{\min} = \frac{2P_{in}}{(V_{\max}^2 - V_{\min}^2) f_{rect}} \quad (5.1)$$

Where P_{in} is the load power in watts, *f*rect is the ripple frequency, *V*max is the maximum DC voltage and *V* min is the minimum dc voltage [31]. In practical realization, if a one phase 230V AC input is connected to the input of the rectifier. The peak voltage value of input is as follows:

$V_{\max} = \sqrt{2}Vl = 325V$

Assuming $V_{\min} = 96\%V_{\max} = 312V;$

$p_{in} = 1.5*746 = 1119W$; $frec = 2*50 = 100Hz$

$$C_{\min} = \frac{2P_{in}}{(V_{\max}^2 - V_{\min}^2) f_{rect}} = \frac{2*1117.5}{(325^2 - 312^2)*100} = 2700mF$$

So the capacitor which been selected was equal to 3300 mF which is the nearest value available in the local market. A voltage source power inverter is used to convert the DC bus to the required AC voltages and frequency. In summary, the power section consists of a power rectifier, filter capacitor, and power inverter. The motor is connected to the inverter as shown in Figure 5.8. The power inverter has 6 switches that are controlled in order to generate an AC output from the DC input. Pulse width modulation PWM signals generated from the FPGA controller will control these 6 switches. The phase voltage is determined by the duty cycle of the PWM signals. In time, a maximum of three switches will be on, either one upper and two lower switches, or two upper and one lower switch. When the switches are on, current flows from the DC bus to the motor winding. Because the motor windings are highly inductive in nature, they hold electric energy in the form of current. This current needs to be dissipated while switches are off. Diodes connected across the switches give a path for the current to dissipate when the switches are off. These diodes are also called freewheeling diodes. Upper and lower switches of the same limb should not be switched on at the same time. This will prevent the DC bus supply from being shorted. A dead time is given between switching off the upper switch and switching on the lower switch and vice versa, as will be explained in next section.

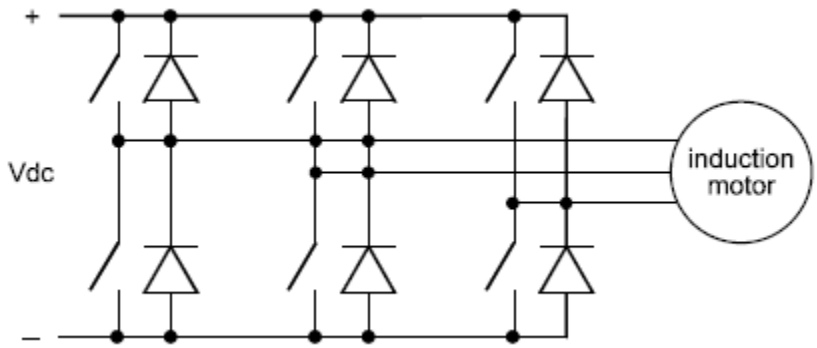


Figure 5.8: Six switches and free-wheeling diodes used in the inverter.

Up to switching powers of a few kW MOSFET (Metal Oxide Silicon Field-Effect Transistors) are often used as switching devices. The MOSFET can be switched on and off by a low level signal requiring virtually no current, is robust and has excellent conduction when 'switched on'. For powers up to about 10 kW, IGBTs (Insulated Gate Bipolar Transistors) are the economical power control devices shown in Figure 5.9. For higher powers the only available devices are thyristors [39].

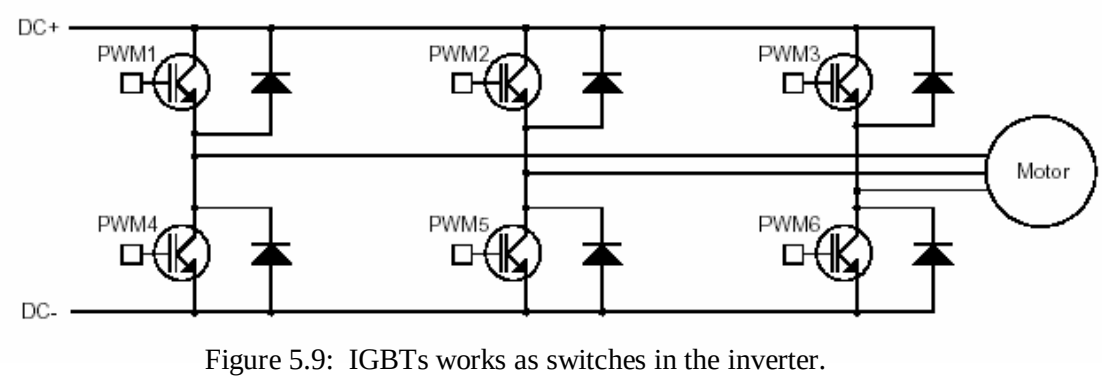


Figure 5.9: IGBTs works as switches in the inverter.

Depending on the motor power size and the available chips in our local market, the IXDH20N IGBT has been selected which its datasheet available in Appendix B. The practical IGBTs circuit which was implemented in this thesis are shown in Figure 5.10.

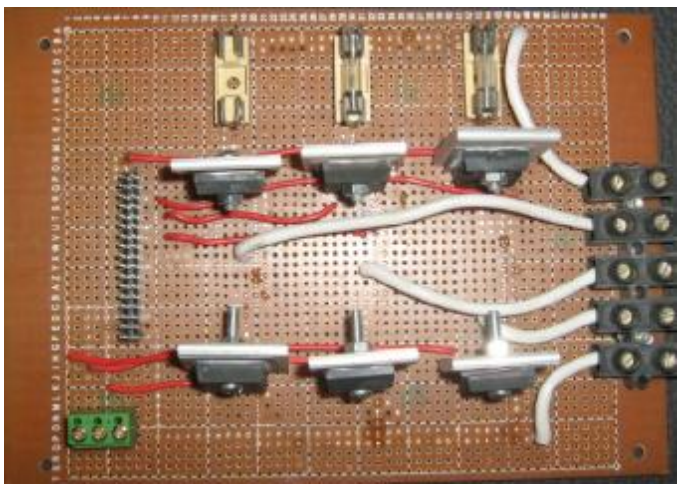


Figure 5.10: Practical IGBTs circuit.

The signal coming from the FPGA controller can't feed directly to the IGBT gate pin due to the weakness of its voltage (3.3v). So it's required to build a power driver circuit to provide the required voltage. This driver shown in Figure 5.11 include four transformers, one for each upper three IGBT and common one for the other three lower IGBT due to its common source pins. Another two transformers are added to this board to provide power to auxiliary electrical units like the relay and the feedback sensor. In practice, its better to use the positive –negative driver, which send 10 volt to the gate of IGBT in case "ON" and -10 volt in case "OFF" to insure that the IGBT clears its internal capacitor which is connected between gate and source as shown in Figure 5.12 .

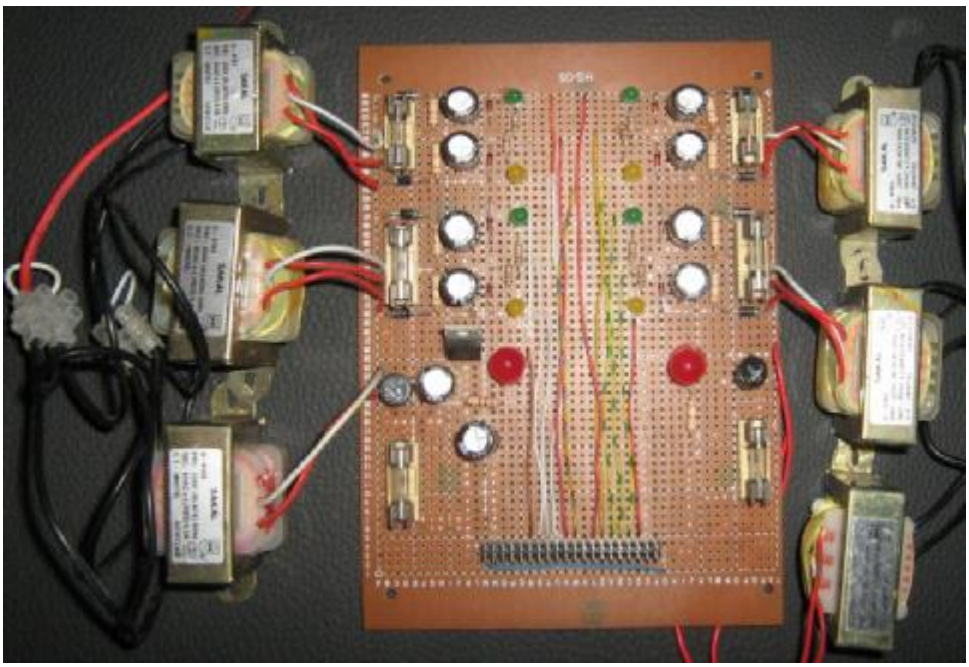


Figure 5.11: Power driver circuit to provide the required voltage for the overall system.

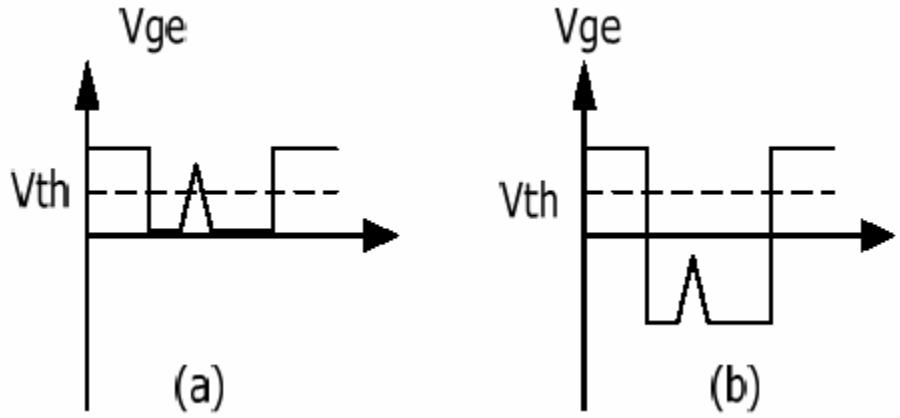


Figure 5.12: Driver signal for the IGBT.

Figure 5.13 shows the circuit diagram for a leg, which includes high and low sides of IGBT modules. The over all driver circuit contains three identical legs. The driver chip which is selected here is HCPL-3120 and its datasheet available in Appendix C. Figure 5.14 shows the real circuit containing another protection IC's to provide the complete isolation between the FPGA card and high Voltage stage.

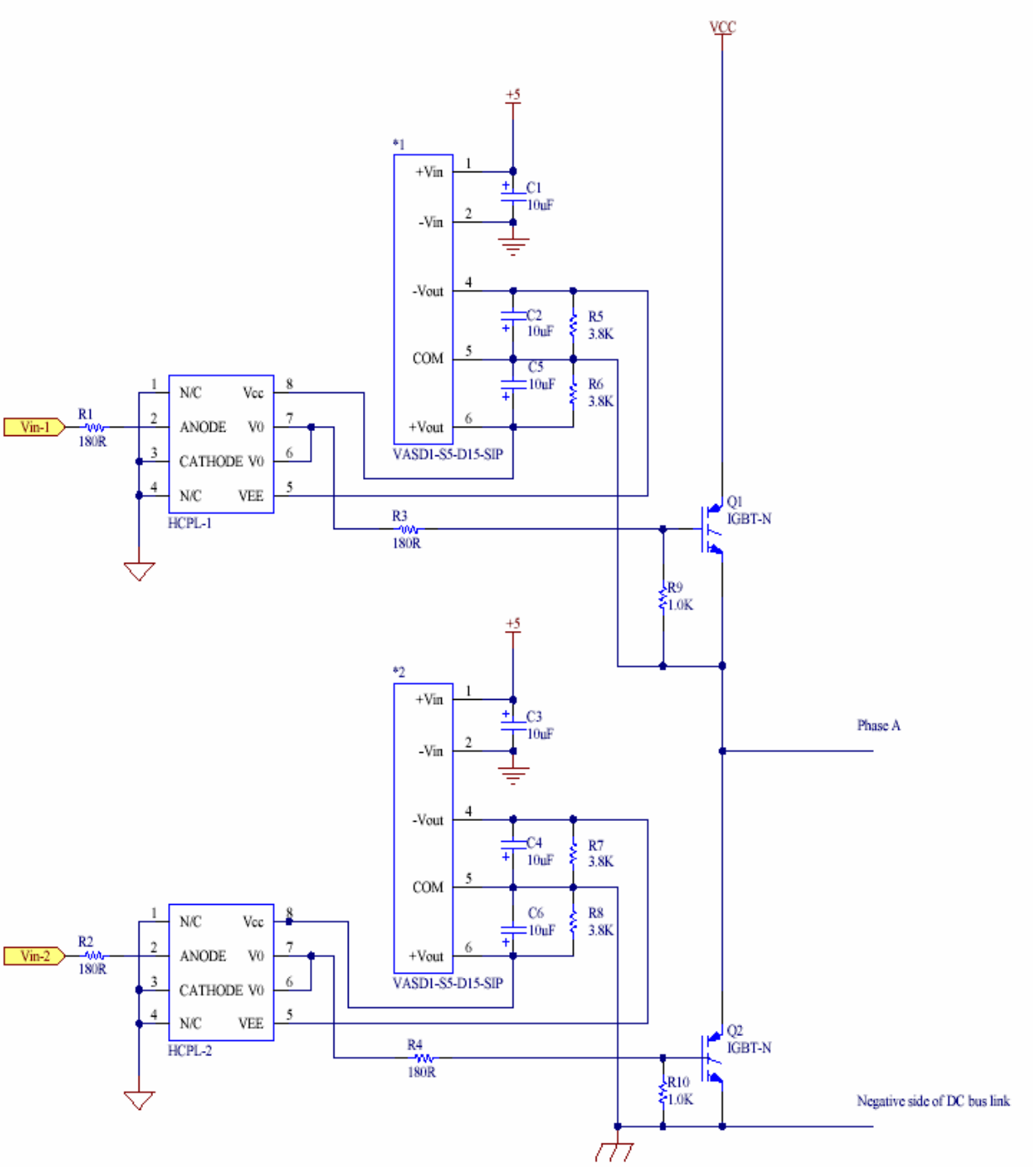


Figure 5.13: Driver signal for the IGBT.

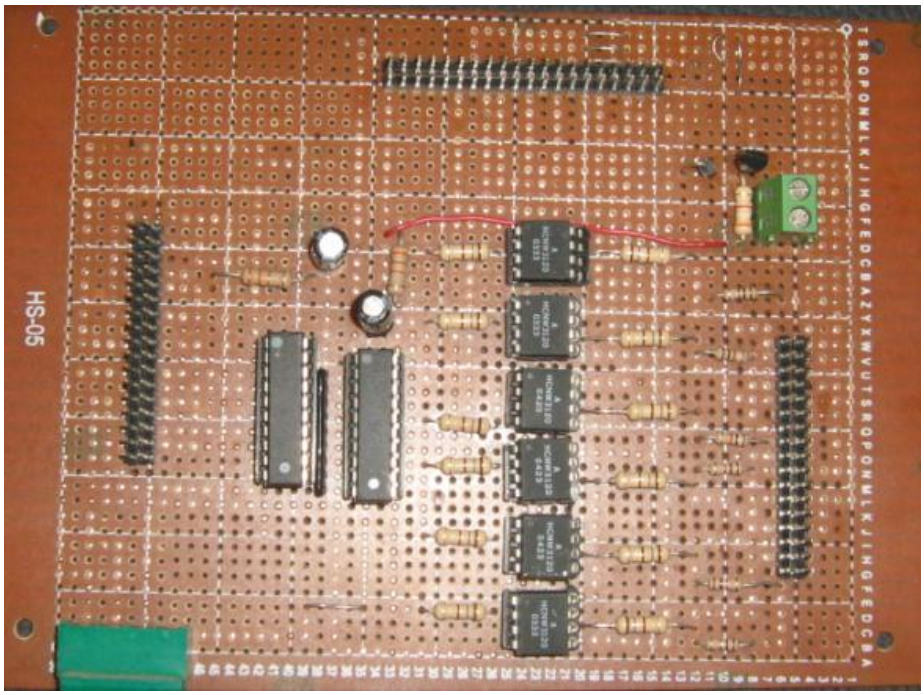


Figure 5.14: Practical IGBT driver circuit.

5.2.2 Software implementation of the inverter

As has been motioned in chapter 4, the FPGA can be programmed using VHDL language, which is selected as the main language in programming the process and needs to generate the outputs PWM signals. The main idea here is to generate a three PWM signals for the upper three IGBTs and the inverse of them towards the other lower three IGBTs with some modification as will be illustrated in the next section. The resultant AC frequency of these signals will vary from 0 to 200 HZ range depending on the value of specified register called the plant speed register. Then this register will be used as speed input to the plant in the controller design (i.e. the three phase motor plant has only one input register to specify its speed, and the output of this block will generate the required six PWM channels where its values will depend on the input speed) as shown in Figure 5.15.

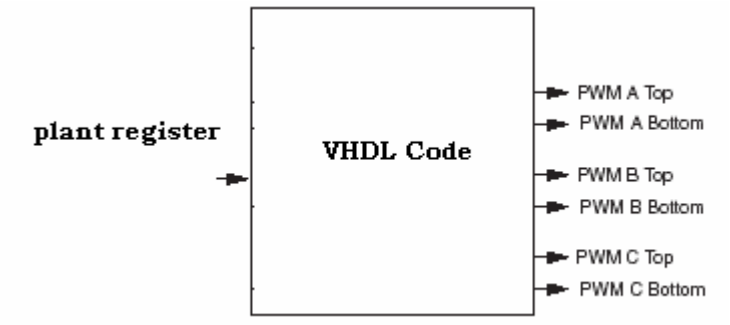


Figure 5.15: Input output for the Driver software.

5.2.2.a Pulse width modulators

To generate the upper three phase PWM signals of the bridge inverter needed to run the induction motor we construct a single up-down counter unit, and three comparator units. For example we consider a five bit counter counts from 0 – 31, with 0 being the minimum voltage available at the power inverter and 31 being the maximum voltage available at the power inverter. Now, assuming that the inverter voltage ranges from 0V – 400V, each time the counter increments by 1, the inverter voltage will jump by 1/32 of the maximum voltage, or12.5V. The counter counts from 0 to its maximum value, then from its maximum value to 0, as seen in Figure 5.16 [40].

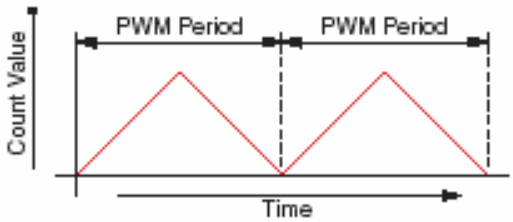


Figure 5.16: Up down counter used in PWM generation.

The frequency of the PWM signal is equal to the frequency of the counter and is typically in the 16 – 20 KHz range (outside of the audible range). A fine line must be drawn between audible noise, and power loss that occurs as the frequency of the PWM period is increased. Another limiting factor of the PWM period frequency is the switching frequency of the power semiconductors used by the inverter. These devices have physical switching frequency limitations, which they will fail to operate over it. One compare unit is used for each of the three phases, the compare units compare the count value from the counter unit with a user specified value derived from the sine wave lookup tables. Each compare unit has two PWM outputs, one is asserted when the count value is ≥ the user specified value, and the other is asserted when the count value is < the user specified value [33]. The relationship between the compared values and the PWM Top and Bottom outputs of the compare unit is shown in Figure 5.17.

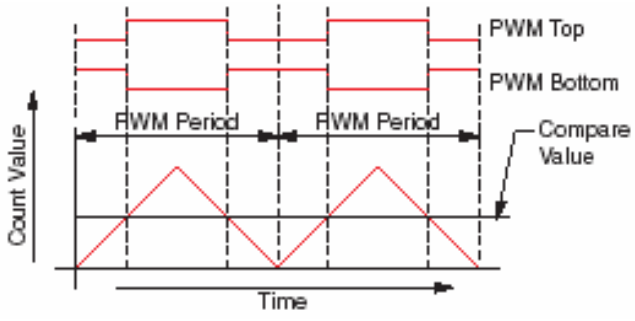


Figure 5.17: Top and bottom PWM signals.

This type of up-down counter/compare unit combination generates pulses center aligned within the PWM period. This type of center aligned PWM which is shown in Figure 5.18 has advantages over edge aligned PWM Figure 5.19 because the outputs signals in this method will not run the IGBT's together, at the beginning of every period, as they would do with edge aligned PWM. This can help reduce noise on the inverter power lines; thus, increasing motor power efficiency.

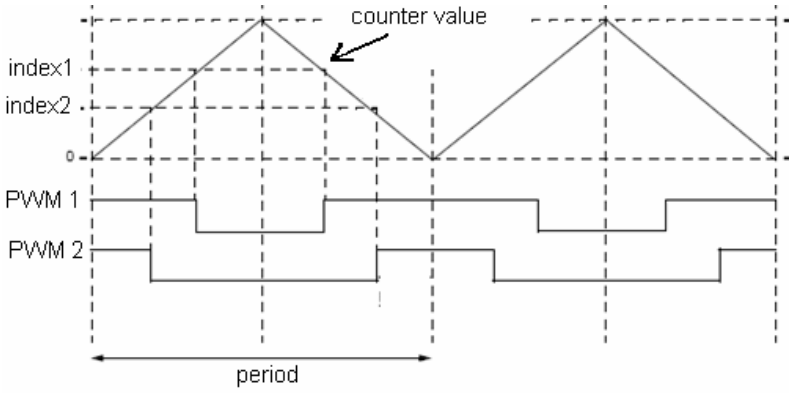


Figure 5.18: Center aligned method in PWM generation.

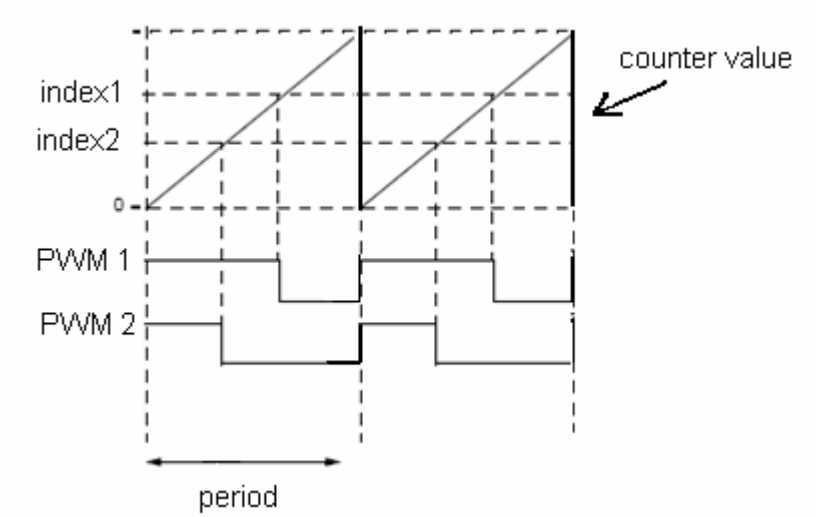


Figure 5.19: Edge aligned method in PWM generation.

In order to control the inverter efficiently and without damaging the power semiconductors, it is important to consider the time it takes for the power semiconductors to switch-on and off. Figure 5.20 shows that the PWM bottom signal is the inverse of the PWM top signal; this is fine in theory, but in the real world the power semiconductors do not switch immediately. There could be a period of time where both the top and bottom power semiconductors of a phase are on at the same time causing a direct short to ground. To alleviate this problem, a dead-band is inserted between the turning off of the PWM bottom signal and the turning on of the

PWM top signal, and between the turning off of the PWM top signal and the turning on of the PWM bottom signal. Thus, both PWM outputs remain off to allow the power semiconductors time to switch.

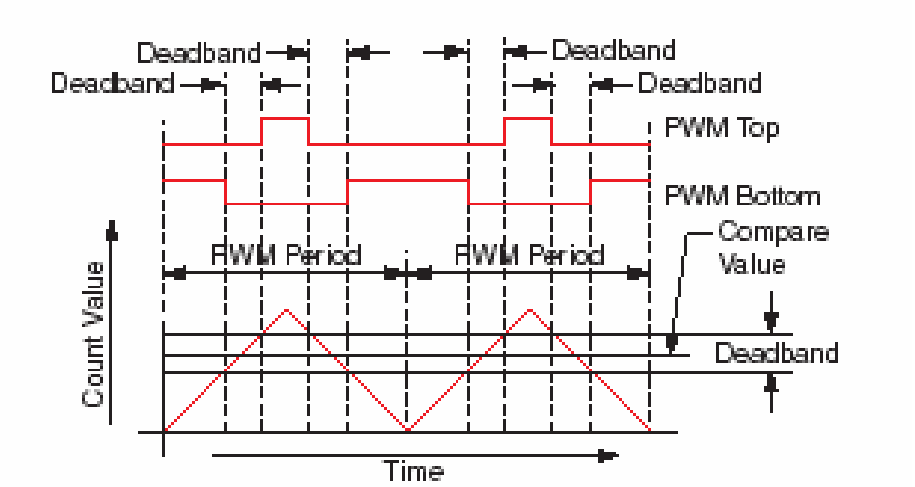


Figure 5.20: Dead-band region.

With dead-time inserted, the PWM top output is asserted when the count value is ≥ user specified value + (deadband/2) and the PWM bottom output is asserted when the count value is < user specified value – (deadband/2). I used a $2\,m$ second as dead-band time in this thesis [40].

5.2.2.b Sine wave generation

There are many methods approved to control the frequency varying of the sine wave signal. Here, sine table skipping method was selected due to its simplicity and efficiently in programming using VHDL code. It's required to create a sine lookup table; the values contained in this lookup table are created by a special program called sinegen which is available at Xilinx website [33]. The sinegen program generates a table containing 360 degree of the sine wave and has 256 entries, with each entry being 16-bits wide as shown in Figure 5.21. The required three sine waves (each 120 degree out of phase from the others) are created by using three separate indices into the lookup table, as shown in Figure 5.22. It is important to note that the values which appear in the lookup table represent a sine wave with its maximum voltage. In others words, this table should change depending on the required frequency to keep the ration voltage/frequency always constant, for example at low frequency the peak value for the sine wave will be very small reaching about "X000a" which approximately equal to 5 volt, and at maximum frequency the peak value will equal to maximum "Xffff" which approximately equal to 325v.

```
constant sine : memory := (
   X"8000", X"8324", X"8647", X"896a", X"8c8b", X"8fab", X"92c7", X"95e1",
   X"98f8", X"9c0b", X"9f19", X"a223", X"a527", X"a826", X"ab1e", X"ae10",
   X"b0fb", X"b3de", X"b6b9", X"b98c", X"bc56", X"bf16", X"c1cd", X"c47a",
   X"c71c", X"c9b3", X"cc3f", X"cebf", X"d133", X"d39a", X"d5f4", X"d842",
   X"da81", X"dcb3", X"ded6", X"e0eb", X"e2f1", X"e4e7", X"e6ce", X"e8a5",
   X"ea6c", X"ec23", X"edc9", X"ef5e", X"f0e1", X"f254", X"f3b5", X"f503",
   X"f640", X"f76b", X"f883", X"f989", X"fa7c", X"fb5c", X"fc29", X"fce2",
   X"fd89", X"fe1c", X"fe9c", X"ff08", X"ff61", X"ffa6", X"ffd7", X"fff5",
   X"ffff", X"fff5", X"ffd7", X"ffa6", X"ff61", X"ff08", X"fe9c", X"fe1c",
   X"fd89", X"fce2", X"fc29", X"fb5c", X"fa7c", X"f989", X"f883", X"f76b",
   X"f640", X"f503", X"f3b5", X"f254", X"f0e1", X"ef5e", X"edc9", X"ec23",
   X"ea6c", X"e8a5", X"e6ce", X"e4e7", X"e2f1", X"e0eb", X"ded6", X"dcb3",
   X"da81", X"d842", X"d5f4", X"d39a", X"d133", X"cebf", X"cc3f", X"c9b3",
   X"c71c", X"c47a", X"c1cd", X"bf16", X"bc56", X"b98c", X"b6b9", X"b3de",
   X"b0fb", X"ae10", X"ab1e", X"a826", X"a527", X"a223", X"9f19", X"9c0b",
   X"98f8", X"95e1", X"92c7", X"8fab", X"8c8b", X"896a", X"8647", X"8324",
   X"8000", X"7cdb", X"79b8", X"7695", X"7374", X"7054", X"6d38", X"6a1e",
   X"6707", X"63f4", X"60e6", X"5ddc", X"5ad8", X"57d9", X"54e1", X"51ef",
   X"4f04", X"4c21", X"4946", X"4673", X"43a9", X"40e9", X"3e32", X"3b85",
   X"38e3", X"364c", X"33c0", X"3140", X"2ecc", X"2c65", X"2a0b", X"27bd",
   X"257e", X"234c", X"2129", X"1f14", X"1d0e", X"1b18", X"1931", X"175a",
   X"1593", X"13dc", X"1236", X"10a1", X"0f1e", X"0dab", X"0c4a", X"0afc",
   X"09bf", X"0894", X"077c", X"0676", X"0583", X"04a3", X"03d6", X"031d",
   X"0276", X"01e3", X"0163", X"00f7", X"009e", X"0059", X"0028", X"000a",
   X"0001", X"000a", X"0028", X"0059", X"009e", X"00f7", X"0163", X"01e3",
   X"0276", X"031d", X"03d6", X"04a3", X"0583", X"0676", X"077c", X"0894",
   X"09bf", X"0afc", X"0c4a", X"0dab", X"0f1e", X"10a1", X"1236", X"13dc",
   X"1593", X"175a", X"1931", X"1b18", X"1d0e", X"1f14", X"2129", X"234c",
   X"257e", X"27bd", X"2a0b", X"2c65", X"2ecc", X"3140", X"33c0", X"364c",
   X"38e3", X"3b85", X"3e32", X"40e9", X"43a9", X"4673", X"4946", X"4c21",
   X"4f04", X"51ef", X"54e1", X"57d9", X"5ad8", X"5ddc", X"60e6", X"63f4",
   X"6707", X"6a1e", X"6d38", X"7054", X"7374", X"7695", X"79b8", X"7cdb"
);
```

Figure 5.21: Sine Look-up table to generate Vmax= 325V.

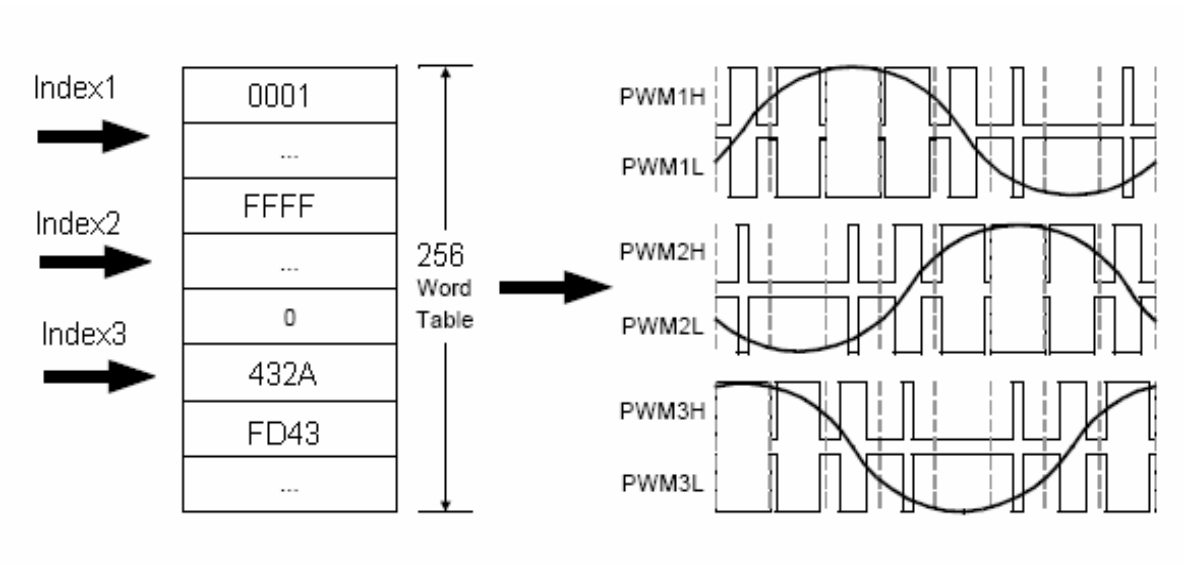


Figure 5.22: Indexes for the Sine look-up table.

The sine wave generation depending on the integer based sine lookup table does not have to be mean perfect as shown in Figure 5.23.

However distortion of the sine wave begins as soon as it enters the digital domain.

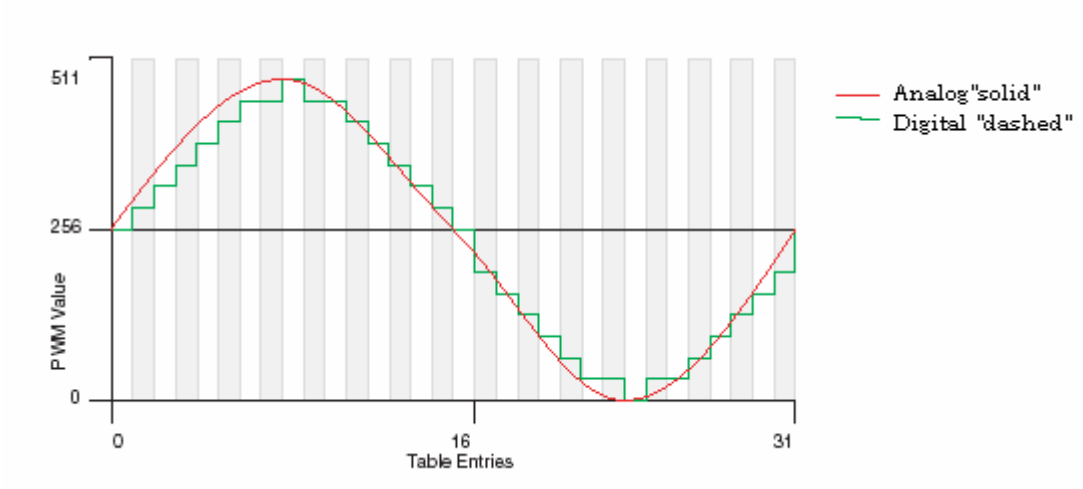


Figure 5.23: Analog and digital 32 entries signal for sine wave.

5.2.2.c Sine table skipping method

This method of sine wave generation employs a static 256 entry lookup table that contains 360° of a sine wave. By reading values from the table in sequence and passing them to the PWMs, it is possible to generate a sinusoidal output. Figure 5.23 depicts a 32-entry sine lookup table with 9-bit PWM values. The solid line (continuous one) represents the analog sine wave, while and the green line (discrete one) is the digital representation of this sine wave. The frequency of the sine wave can be increased by skipping entries in the sine lookup table. If for example, the lookup table entries shaded grey in Figure 5.23 were skipped, the resulting sine wave as shown in Figure 5.24 would be twice the frequency of that which is displayed in Figure 5.23. Hence, the term Sine Table Skipping refers to the skipping of entries in the sine lookup table to increase the frequency of the sine wave [40].

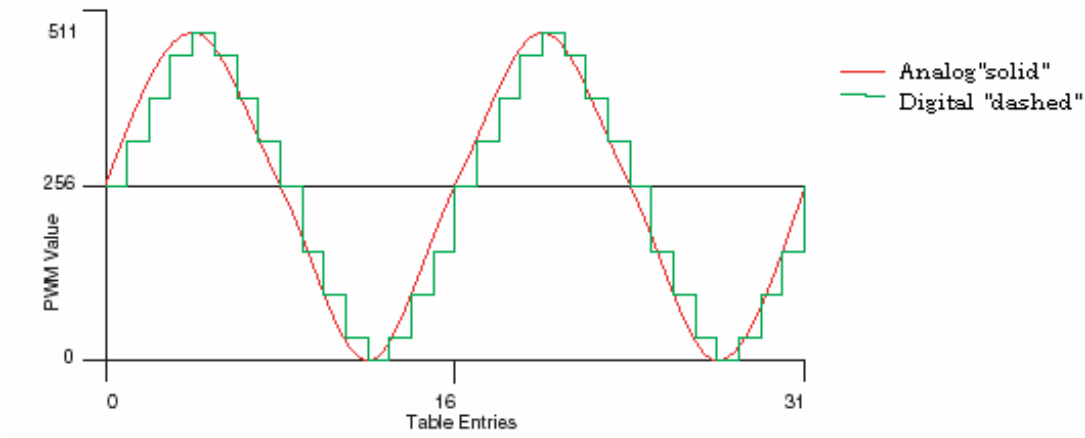


Figure 5.24: Sine table skipping method to increase/decrease the frequency.

This method of sine wave generation has two major drawbacks. First, the minimum sine wave frequency is based upon the number of entries in the sine lookup table and the PWM period. With a PWM period of 16 KHz and a 256 entry sine lookup table, the minimum sine wave frequency would be 62.5 Hz (16000/256=62.5). Secondly, the maximum frequency of the sine wave is also limited by the size of the sine table, as we skip more and more entries in the lookup table to increase the sine wave frequency; the sine wave becomes more and more distorted, until it no longer resembles a sine wave at all. As this distortion increases, it is possible that the peaks of the sine waves are clipped, reducing the utilization of the DC bus voltage; because of the wrap-around nature of the access to the sine lookup tables, this could cause unwanted harmonics to be generated in the windings of the motor. A better solution to both of these problems is to use a larger sine lookup table. This would allow for a lower minimum sine wave frequency, and a less distorted maximum sine wave frequency. However, large lookup tables may be undesirable due to the amount of space required within the FPGA device to hold them. An intermediate solution is to use a 256 entry lookup table with linear interpolation. A 16-bit value is used to address the entries in the table; the upper eight bits are used to index the table and the lower eight bits are used to interpolate one of 256 points between the index value and the next index value. This gives the impression that the sine lookup table actually has 65536 entries and not 256 entries. Figure 5.25 depicts the equation used to perform the linear interpolation of the 256 entry sine lookup table [40].

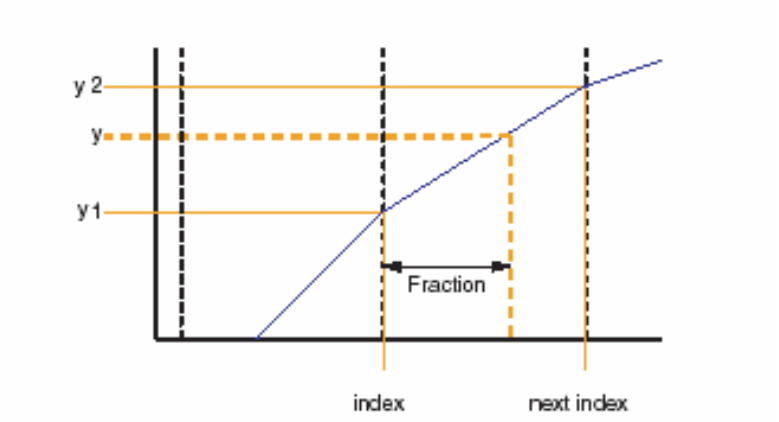


Figure 5.25: Interpolation method.

And the Y value will equal to the following equation:

$$Y = \left(\frac{(Y_2 - Y_1) * Fraction}{2^{Fraction\ Bits}} \right) + Y_1 \qquad (5.2)$$

Although the interpolated sine wave in Figure 5.26 continues to appear distorted, the majority of the step increments seen in the digitized sine wave have been smoothed out.

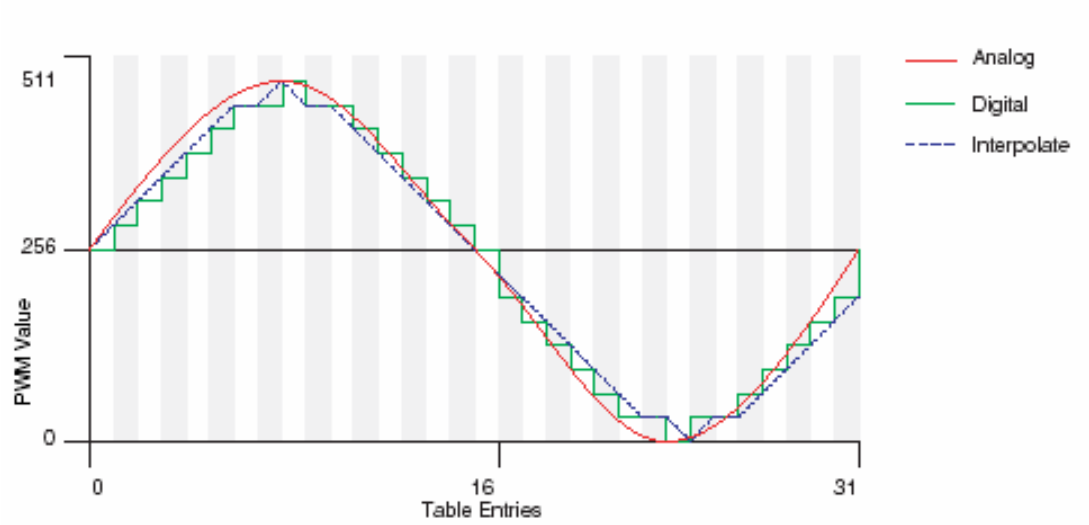


Figure 5.26: Interpolation method applied to sine table skip..

Hence, the block diagram for the plant driver is shown in Figure 5.27. After that the design is ready for the controller part, since there is a unique input to the plant which is 8-bit or 16-bit register called plant speed register, and six digital PWM outputs will feed the driver circuit. The clock "clk" input is a 50 MHZ clock which provides the time base for the software. In VHDL programming, this block diagram called the top level of the inverter module, and its ready to connect to the others blocks in the overall design to implement the general top level block diagram.

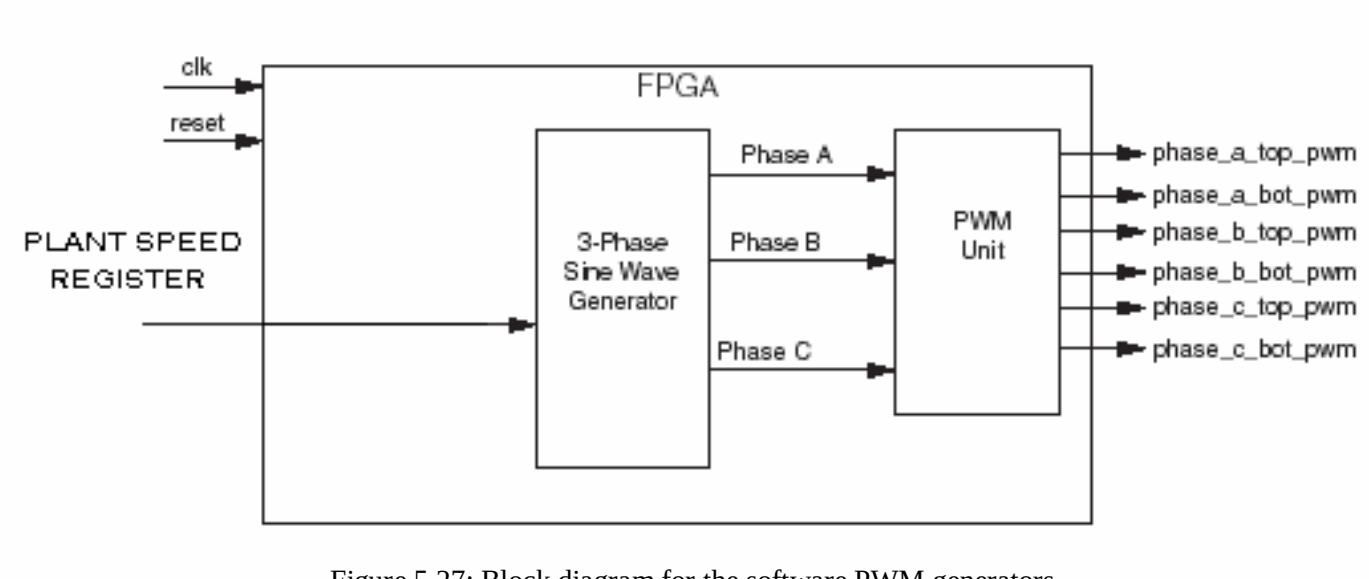


Figure 5.27: Block diagram for the software PWM generators.

5.3 Controllers design

The previous section focused on building the PWM inverter, where a value is stored the plant speed register and generates a three PWM signals required to run the motor at the speed specified in this register; hence, the controller blocks design will contain the PWM inverter as shown in Figure 5.28.

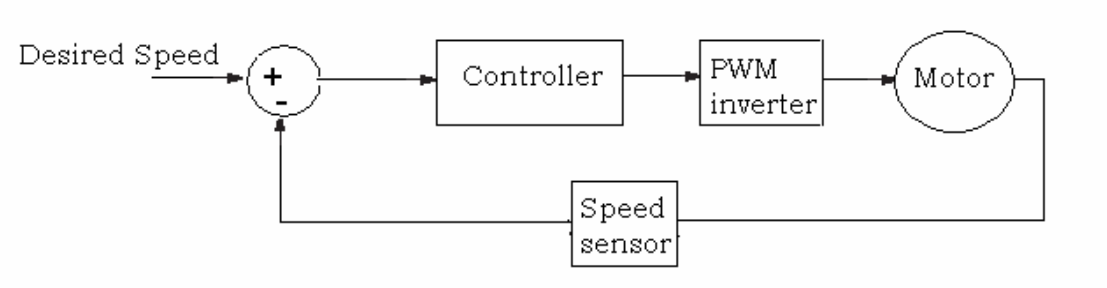


Figure 5.28: General block diagram for the induction motor controller.

Three different controllers were developed in this thesis which are: PID controller, Fuzzy controller, and Fuzzy-PID controller. Finally a comparison will be made among these controllers to find the best performer. To implement the PID controller for the three phase induction motor, it's required to get the model of the motor, as shown in the next subsection.

5.3.1 Model of the induction motor

Three phase induction motor squirrel cage has been selected, which manufactured by Feedback Company " machine Laboratory 64-501 code" shown in Figure 5.29.

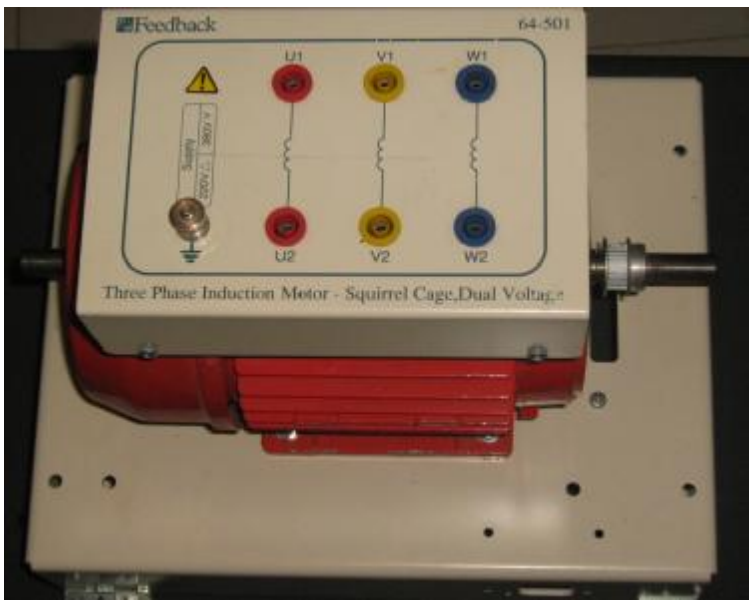


Figure 5.29: Three-phase induction motor squirrel cage "64-501 Feedback company".

This motor has the following specifications:

- Two poles induction motor.
- 250W continuous operation.
- Rotates at up to 2980 rev/min at 50 Hz.
- 380/415 V three phase A.C at 50/60 Hz for star connection and 220/240 V Three phase A.C at 50/60 Hz for delta connection.

In chapter 2, it was explained the technique and the theory needed to get the model of the three phase induction motor shown in Figure 2.23, and extract its parameters via the three practical tests.

The three phase induction motor parameters have been gotten in the lab after the implementation of the tests and the results was as the following:

- **DC resistance Test**

Rs=1.4 Ω

Due to skin effect, the AC resistance value is closer to the practical value than the DC value, so the following approach can be applied:

R1(ac)=(1---1.3)* Rdc =1.2*1.4 = 1.68 Ω

- **BLocked-rotor Test**

$R_2' =$ .56 Ω $\qquad L1 = L2' = 0.0073H$

- **No-load test**

Lm =0.089H

Now that all of the parameters have been found, the motor function can be obtained. Unlike many other motors (i.e. DC Motor), the induction motor is a nonlinear machine.

Thus, to build a robust controller we need a nonlinear designing method which depends on many complex mathematically approaches. However, the nonlinear controller is not being covered in this thesis.

Therefore a ready simulink module depending on linearization technique has been selected as shown in Figure 5.30 which was developed by Professor Mahmoud Riaz. This module needs to fill with the pervious calculated parameters of the induction motor.

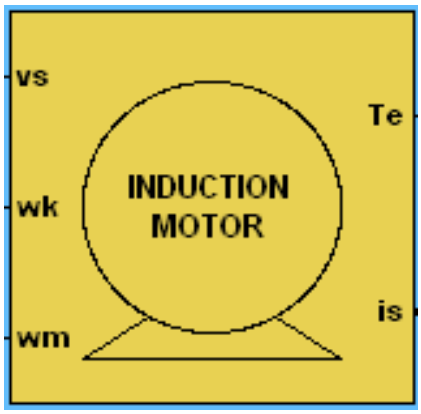


Figure 5.30: Simulink module for induction motor.

Where W_k is the input speed, W_m is the actual speed, and T_e is the torque.

5.3.2 Speed sensor

There are many types for speed sensor. The most widely used is the digital pulses optical encoder, where the output frequency of these pulses is proportional to the speed of the rotor as shown in Figure 5.31. In Practice, dc motor was used with optical digital pulses sensor for this purpose. The dc motor has been mechanically coupled to the rotor of the induction motor as shown in Figure 5.32.

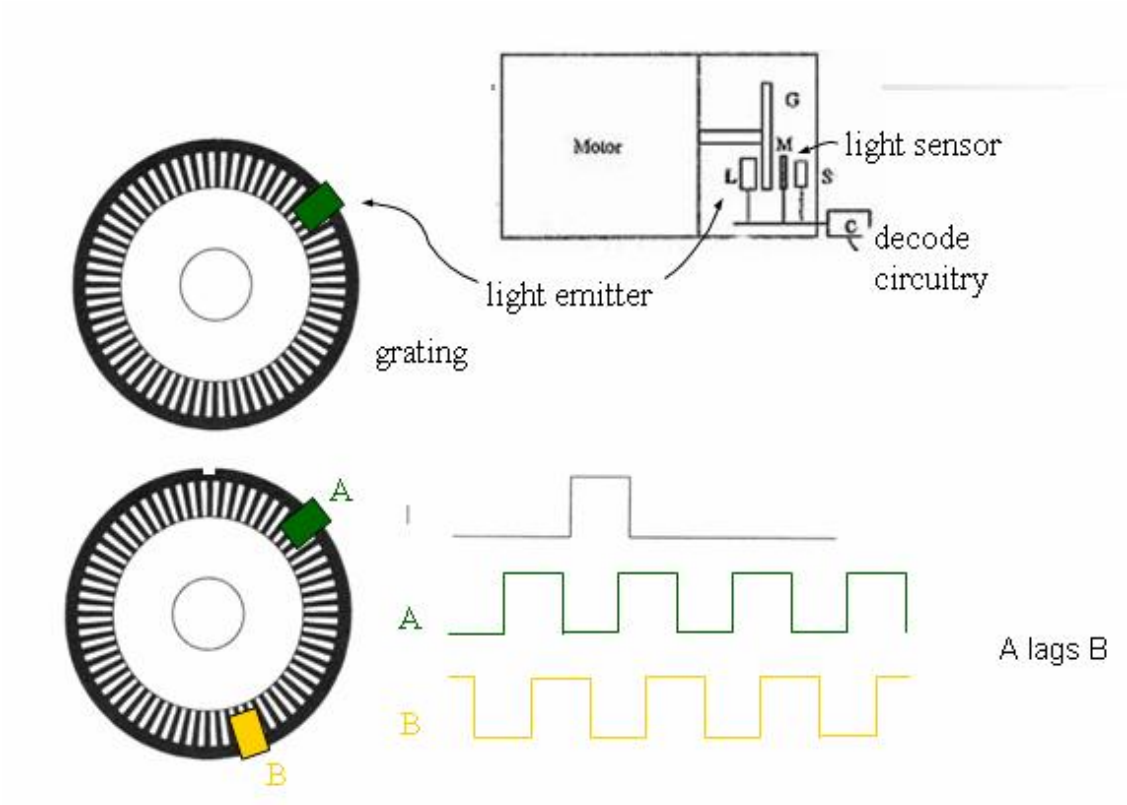


Figure 5.31: Dc motor with pulse optical encoder.

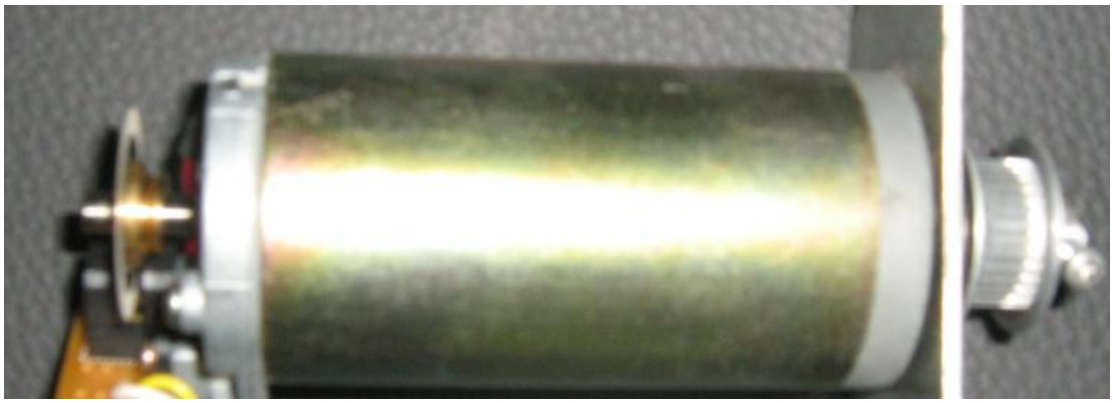


Figure 5.32: Pulse optical encoder.

5.3.3 PID controller

Between many of control design techniques, the PID controller is the most widely used. Over 85% of all dynamic controllers are of the PID variety. PID stands for proportional, integral and differential control. There are many types of the PID controller; here the mostly general and flexible form is selected which is called parallel form as shown in Figure 5.33.

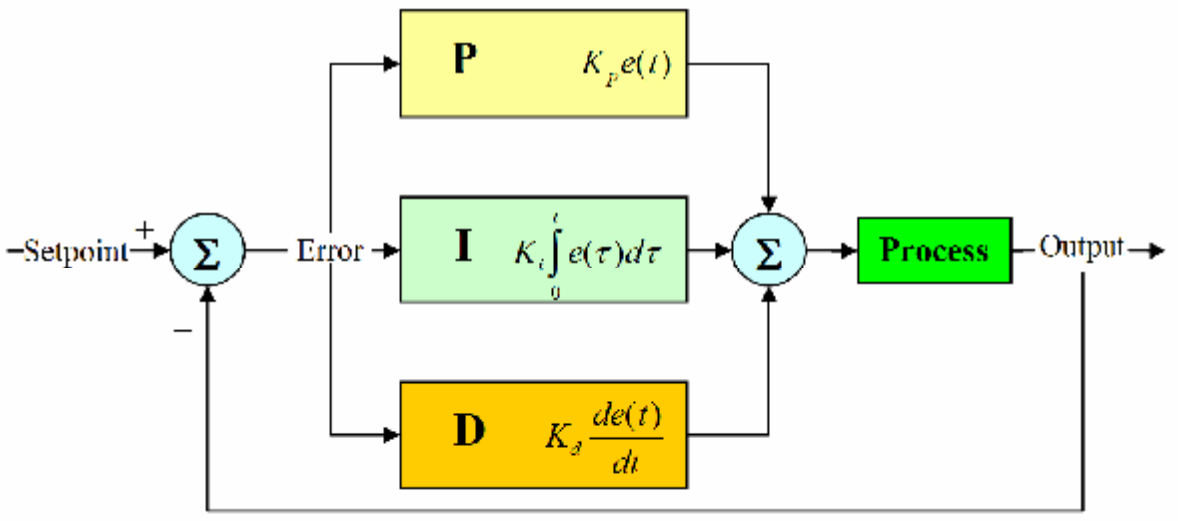


Figure 5.33: General block diagram for PID controller.

5.3.3.a PID parameters

In this thesis, it's required to build a digital PID controller in the FPGA. To do that firstly we need to design an analog PID controller and use same resultant parameters in designing process of the digital one using Tustin approach. The equation which represents the PID controller is:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \qquad (5.3)$$

The effects of each parameter on the step response of the system is illustrated in Table 5.1

Table 5.1: Effects of P, I, and D on the step response.

Effects of *increasing* parameters				
Parameter	**Rise Time**	**Overshoot**	**Settling Time**	**S.S. Error**
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Decrease	Decrease	Decrease	None

Tuning method

There are many methods to get the best values of K_P ,K_I, and K_d. One approach is to use a technique which was developed in the 1950's but has stood the time and is still used today. This is known as the Ziegler Nichols tuning method [41].

Ziegler Nichols tuning method

First, it's needed to get the closed loop transfer function of the system, and then apply the following procedures:

- Select proportional control alone.
- Increase the value of the proportional gain until the point of instability is reached (sustained oscillations), the critical value of gain, Kc, is reached.
- Measure the period of oscillation Pc.

Once the values for Kc and Pc are obtained, the PID parameters can be calculated, according to the design specifications, from Table 5.2.

Table 5.2: Ziegler-Nichols method.

Ziegler-Nichols method			
Control type	K_p	K_i	K_d
P	0.5 K_c	-	-
PI	0.45 K_c	$1.2K_p/P_c$	-
PID	0.6.K_c	$2K_P/P_c$	$K_pP_c/8$

When implementing the Ziegler-Nichols method to the plant, the resultant parameters were K_P= 15, K_I= 7, and K_d=2. Practically due to the approximate transfer function,

these values may need some manual adaption when the system is implemented. The next step is to get the digital form of the PID controller appears in Equation 5.4.

$$C(n) = KpE(n) + KiTs\sum_{0}^{N} E(n) + Kd[E(n) - E(n-1)]/Ts \quad (5.4)$$

A general rule of thumb in control design is to sample at least 4 to 20 times the rise time of the system response, so the sampling time will be chosen to 4.1 millisecond. Hence, the output of the PID controller will feed to the plant speed register of the PWM inverter.

5.3.3.b PID algorithm implementation in VHDL

In FPGA, two PID algorithms were implemented, the first one was written in VHDL language, and the second implemented using PicoBlaze processor which was presented in Chapter 4. The main difficulties faced in programming the two algorithms were in programming the mathematics operations like product and division. The program in FPGA card was developed to use the LCD unit for displaying the values of the K_P, K_I, and K_d as shown in Figure (5.34-35). Figure 5.36 shows the flowchart of the PID algorithm.

Figure 5.34: PID parameters on FPGA card.

Figure 5.35: PID parameters on LCD of the FPGA card.

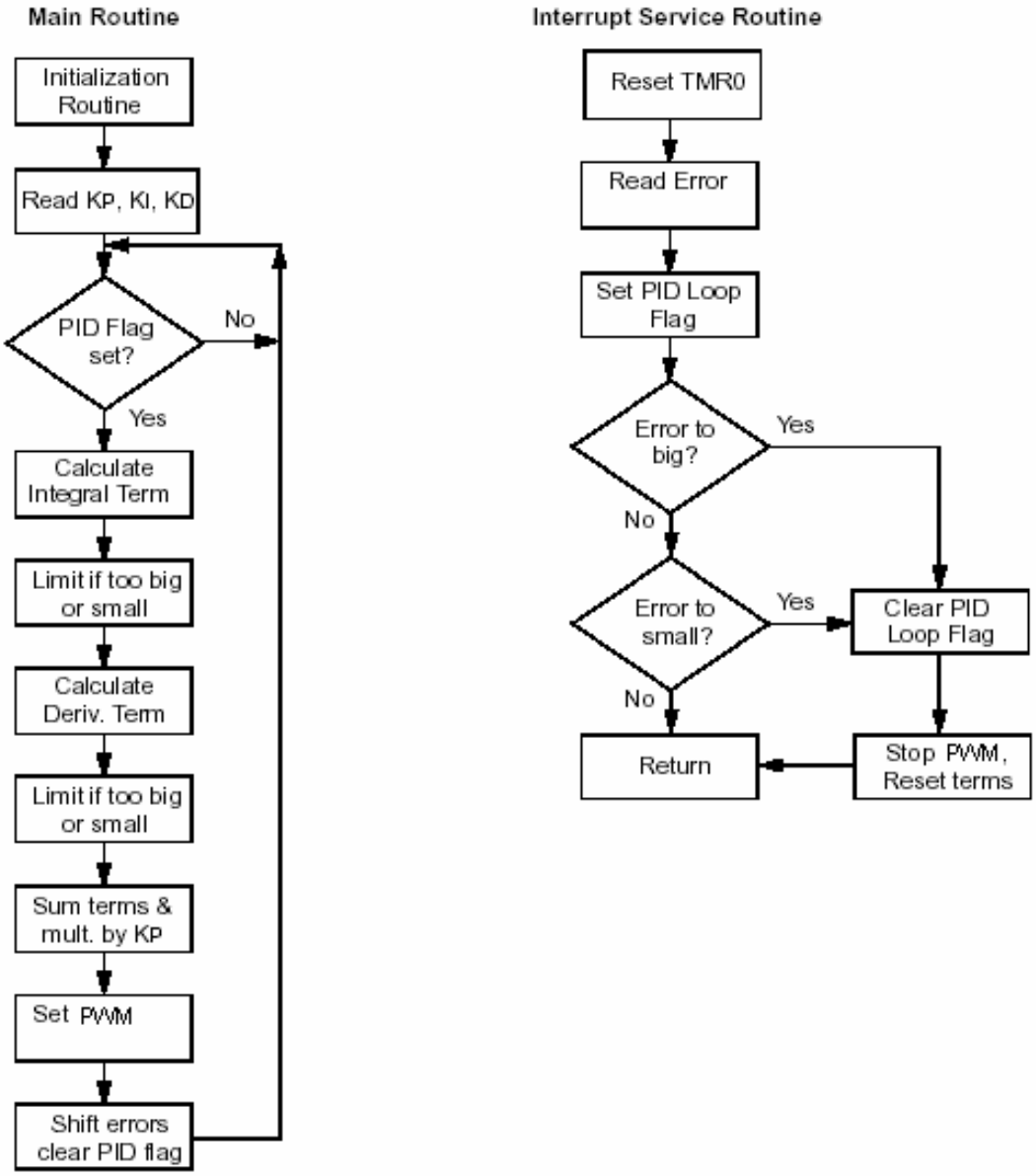


Figure 5.36: PID algorithm flow chart.

5.3.3.c PID controller simulation in Matlab

Figure 5.37 illustrates the Simulink block diagram for the PID speed controller for three phase induction motor.

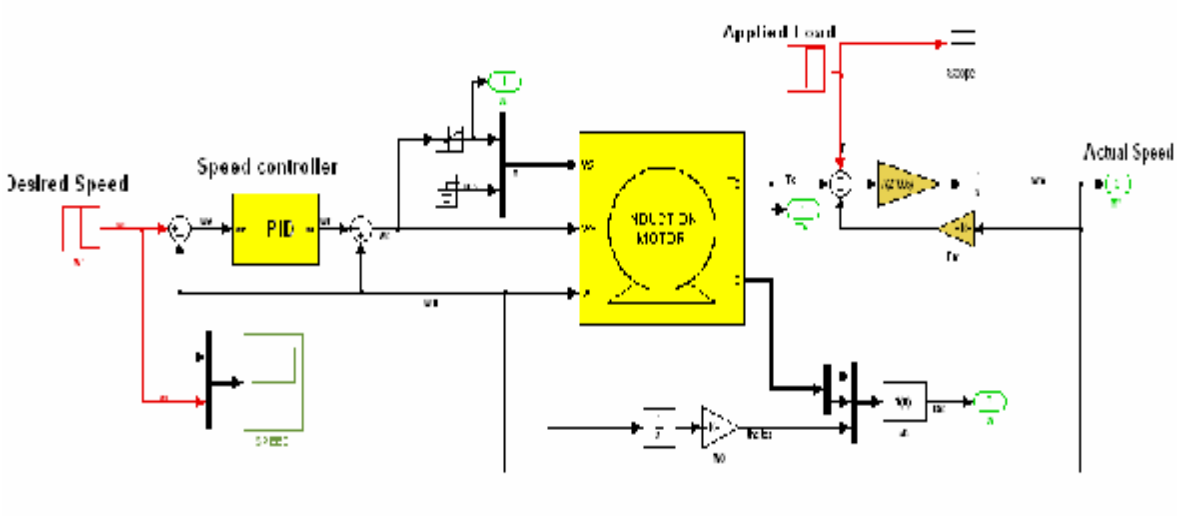


Figure 5.37: PID speed Controller.

The controller has been tested using Simulink motor module in Matlab by applying full load on the rotor after 3.5 second from starting.

I also changed the desired input speed from the rating speed to fifty percentages from this rate after 4 seconds from the starting. The results are shown in Figures 5.38.

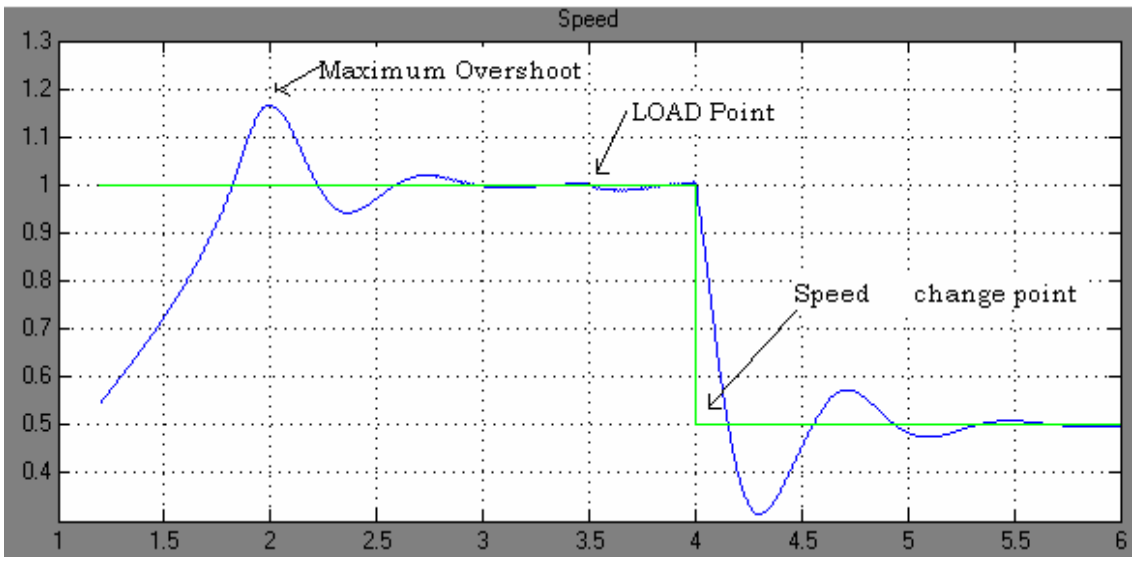


Figure 5.38: PID controller step response with load and speed variation.

5.3.4 Fuzzy logic controller "FLC"

5.3.4.a FLC design

FLC has been constructed Using VHDL and embedded PicoBalze processor in FPGA. The block diagram for the FLC for the three phase induction motor is shown in Figure 5.39.

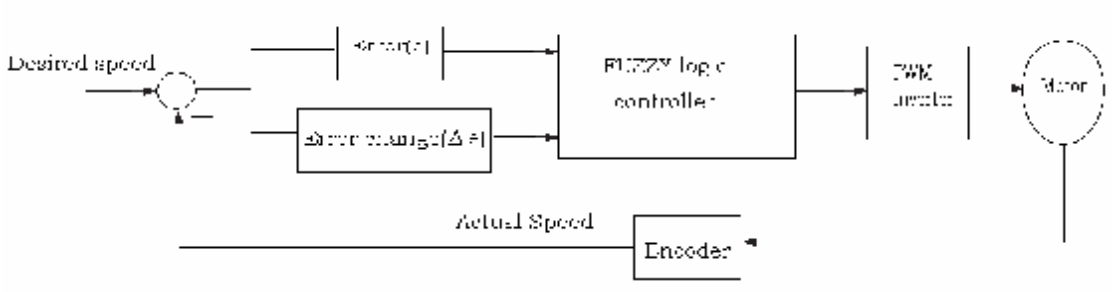


Figure 5.39: FLC controller for the three-phase induction motor.

FLC has two inputs which are: Error and the Error change, and one output feeding to the plant speed register of the PWM inverter. Figure 5.40 illustrates the method used in reaching the desired speed value. For example, at stage A the Error is positive (desired speed –actual speed) and the Change Error (Error – last Error) is negative which meaning that the response is going in the right direction; hence, the FLC will go forward in this direction. Using the same criteria at stage B, the Error is negative and CE is bigger negative; hence, the response is going in wrong direction so FLC will change its direction to enter Stage C, until reaching the desired speed.

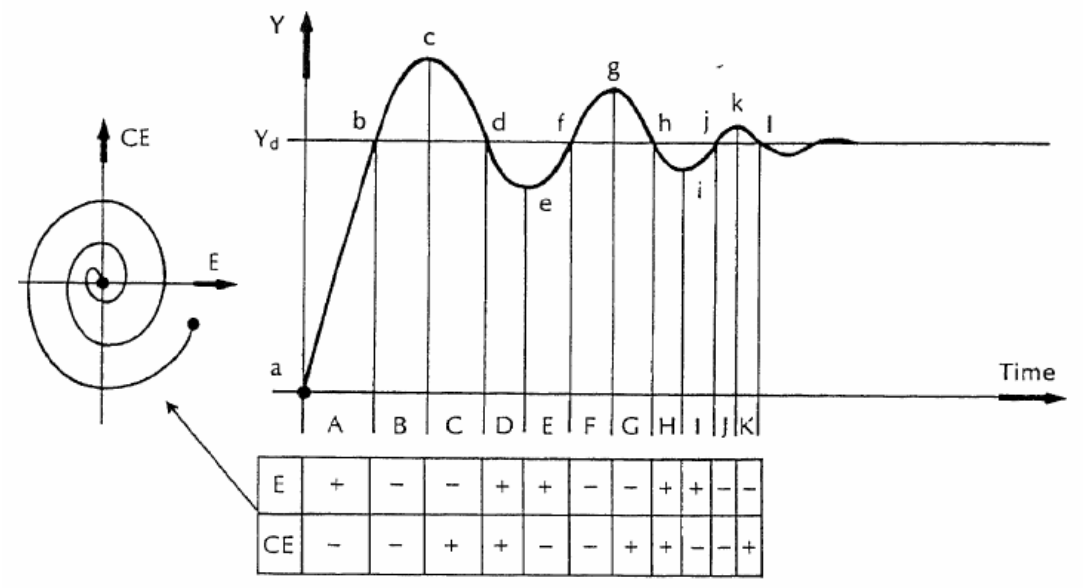


Figure 5.40: Error and error change approach in FLC.

As has been presented in Chapter 3, there are two widely used approaches in FLC implementation: Mamdani and Sugeno. In this thesis, Mamdani approach in FPGA has been used to implement FLC for the three phase induction motor. FLC contains three basic parts: Fuzzification, Base rule, and Defuzzification.

- **Fuzzification**

Figure 5.41 illustrates the fuzzy set of the Error input which contains 7 Triangular memberships

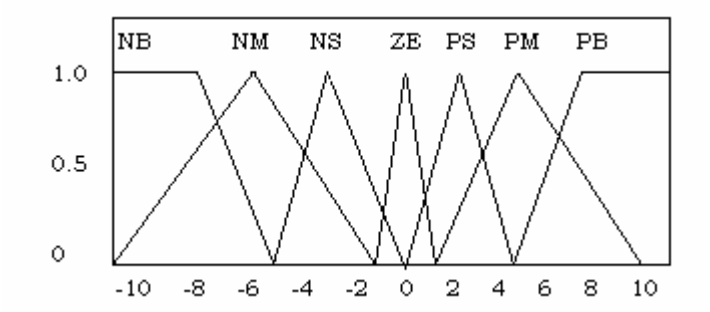


Figure 5.41: Error fuzzy set of FLC.

Figure 5.42 illustrates the fuzzy set of the Change Error input which contains 7 Triangular memberships.

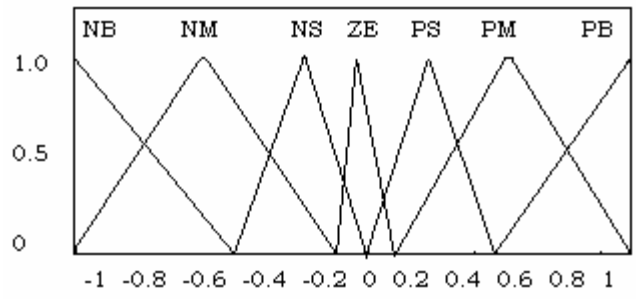


Figure 5.42: Change error fuzzy set of FLC.

Figure 5.43 illustrates the fuzzy set of the output which contains 7 Triangular memberships.

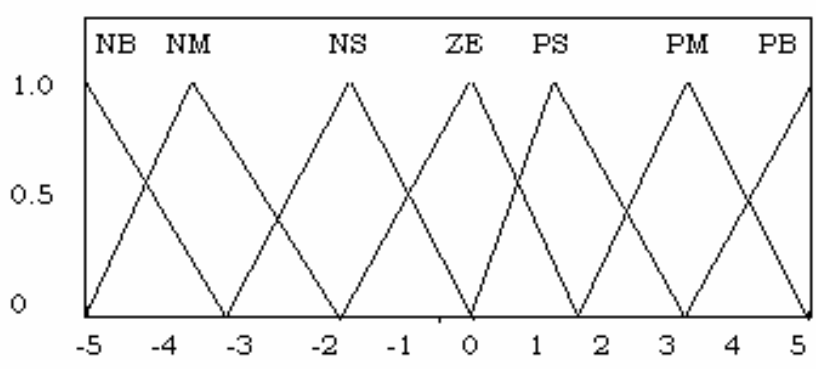


Figure 5.43: Fuzzy set of FLC output entering to plant speed register.

- **Control rule base**

Table 5.3 illustrates the knowledge base defining the rules for the desired relationship between the input and output variables in terms of the membership functions. The control rules are represented as a set of:

IF Error is ... and Change Error is ... THEN the output will … .

The control rules are evaluated by an inference mechanism.

The overall rules are located in Appendix D.

Table 5.3: Control rule base for fuzzy controller.

E / CE	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZE
NM	NB	NB	NM	NM	NS	ZE	PS
NZ	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PB	PB
PB	ZE	PS	PM	PB	PB	PB	PB

Figure 5.44 shown the surface of the base rules using in FLC.

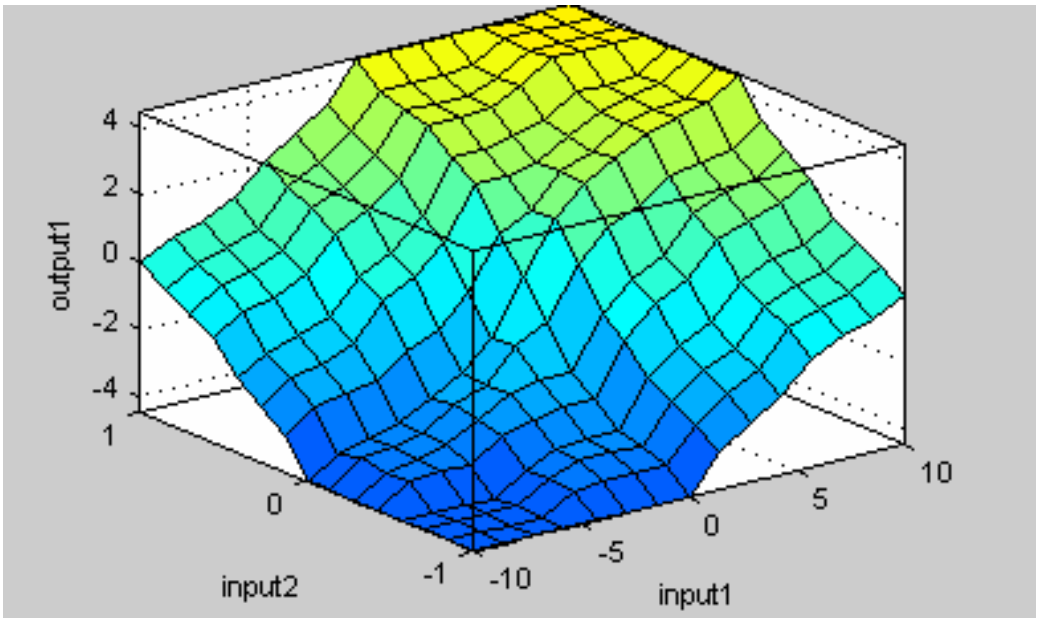


Figure 5.44: Rule surface of FLC.

- **Defuzzification**

As illustrated in Chapter 3, the center of gravity "centroid" method is widely used in Mamdani approach which has been selected in this thesis.

5.3.4.b Fuzzy logic controller simulation in Matlab

Figure 5.45 illustrates the Simulink block diagram for the Fuzzy speed controller for three phase induction motor.

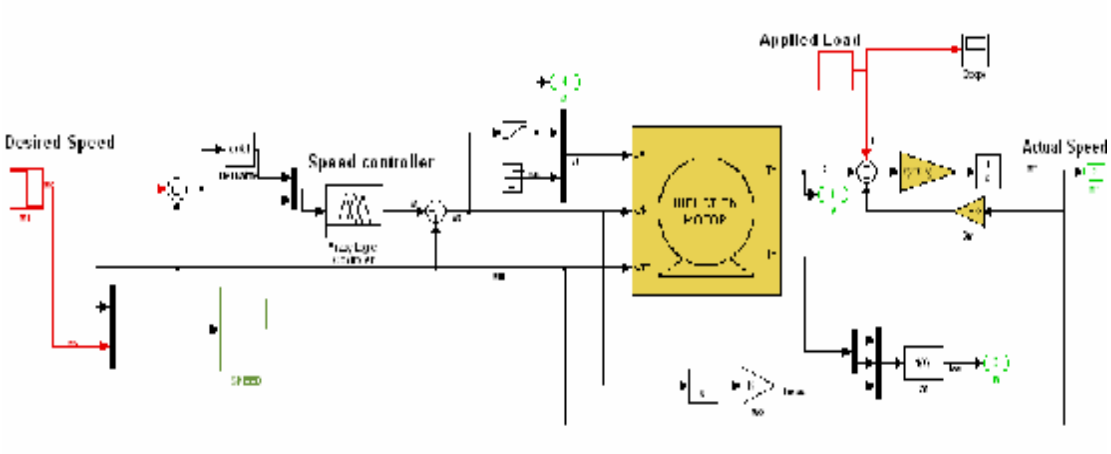


Figure 5.45: Fuzzy logic speed controller.

The controller has been tested using Simulink motor module in Matlab by applying full load on the rotor after 3.5 second from starting. I also changed the desired input speed from the rating speed to fifty percentages from this rate after 4 seconds from the starting, as shown in Figures 5.46.

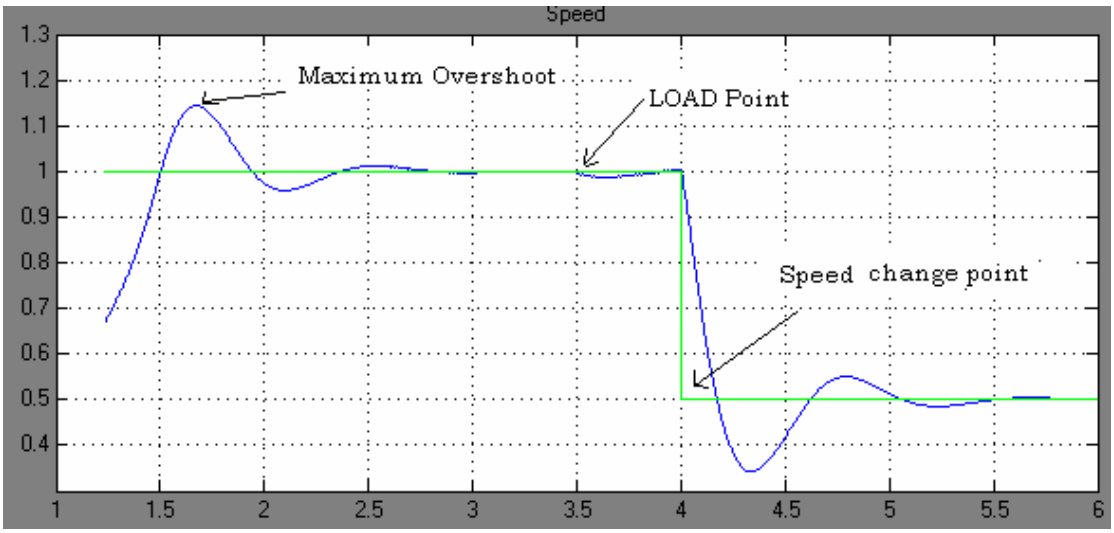


Figure 5.46: Fuzzy controller step response with load and speed variation.

5.3.5 Fuzzy-PID logic controller

5.3.5.a Fuzzy-PID design

Here, another approach which depends on mixing the PID controller with Fuzzy controller in FPGA using VHDL and embedded PicoBlalze processor. Hence, the value of the PID parameters will be evaluated using the Fuzzy controller. Figure 5.47 shows the complete design of this system. Figure 5.48 shows the fuzzy sets for the output: K_p, K_i and K_d, practically there are three different fuzzy sets for these parameters. .For inputs fuzzy sets the same Error and Change Error presented in the previous section has been used.

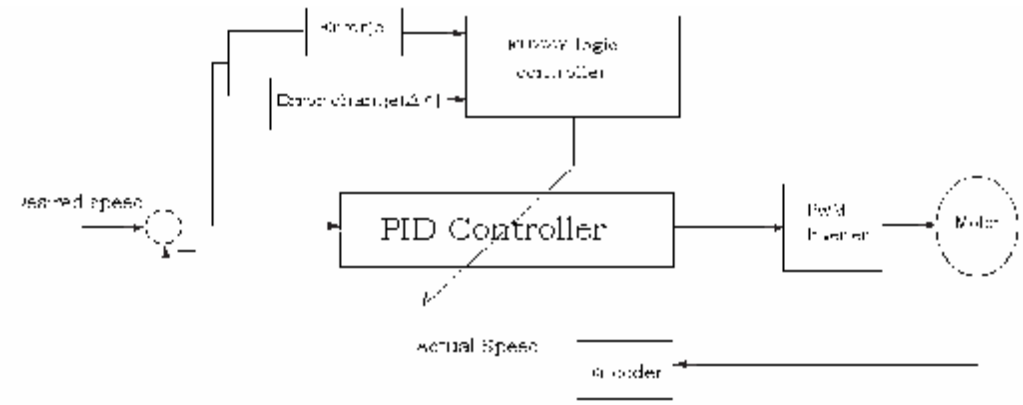


Figure 5.47: Fuzzy-PID controller for the induction motor.

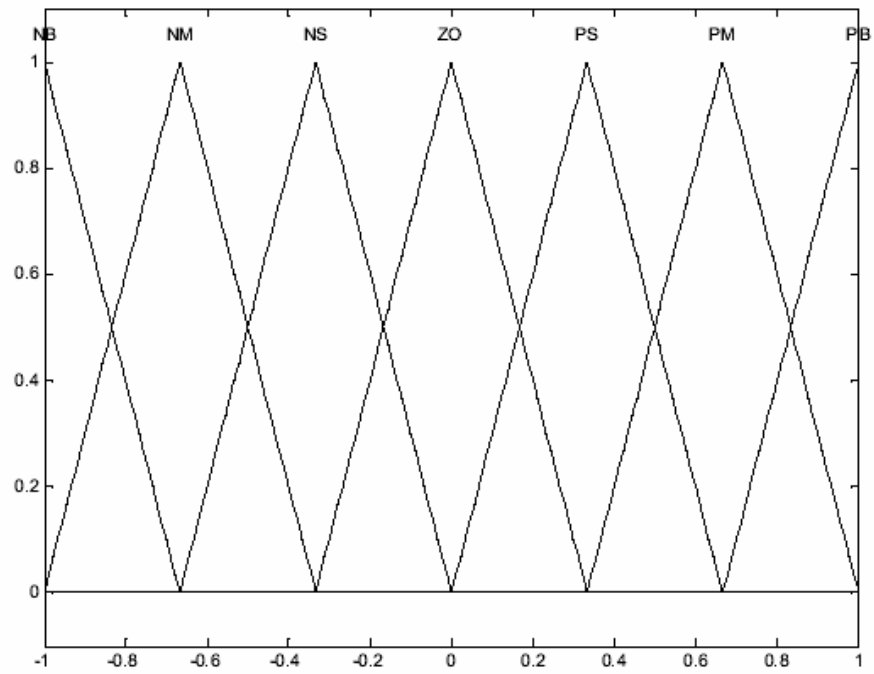


Figure 5.48: Fuzzy sets for Kp, Ki, and Kd with different memberships boundaries.

Table 5.4 illustrates the base Rules of Fuzzy-PID controller. Practically there are three different tables for these parameters. The complete rules are available in Appendix E.

Table 5.4: Control rule base for Fuzzy-PID controller.

E CE	Nb	NM	NS	ZE	PS	PM	PB
PB	ZE	PS	PM	PB	PB	PB	PB
PM	NS	ZE	PS	PM	PB	PB	PB
PS	NM	NS	ZE	PS	PM	PB	PB
ZE	NB	NM	NS	ZE	PS	PM	PB
NS	NB	NB	NM	NS	ZE	PS	PM
NM	NB	NB	NB	NM	NS	ZE	PS
NB	NB	NB	NB	NB	NM	NS	ZE

5.3.5.b Fuzzy-PID controller simulation in Matlab

Figure 5.49 illustrate the Simulink block diagram for the Fuzzy-PID speed controller for three phase induction motor.

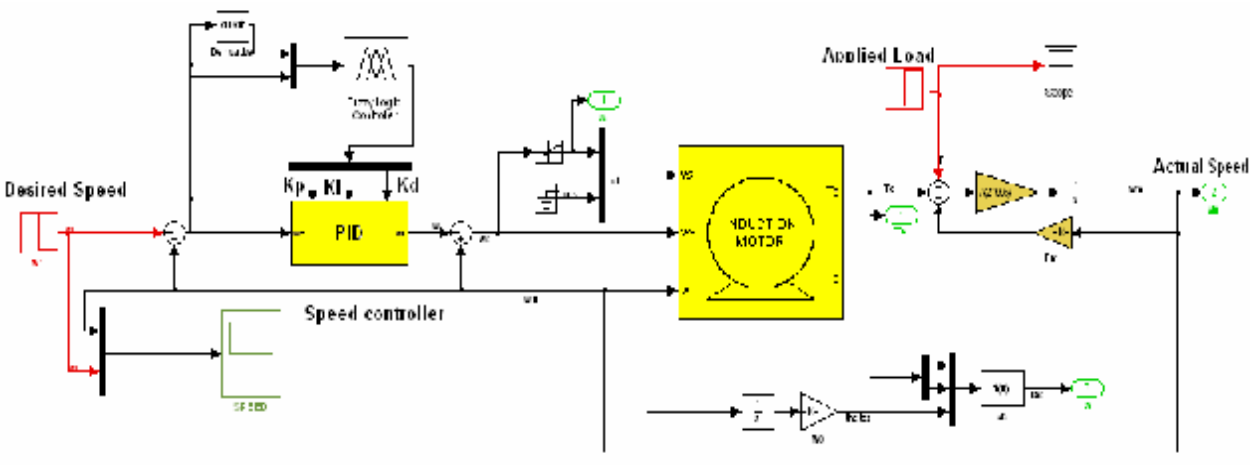


Figure 5.49: Fuzzy-PID Logic speed Controller.

The controller has been tested using Simulink motor module in Matlab by applying full load on the rotor after 3.5 second from starting.

I also changed the desired input speed from the rating speed to fifty percentage from this rate after 4 seconds from the starting, as shown in Figures 5.50.

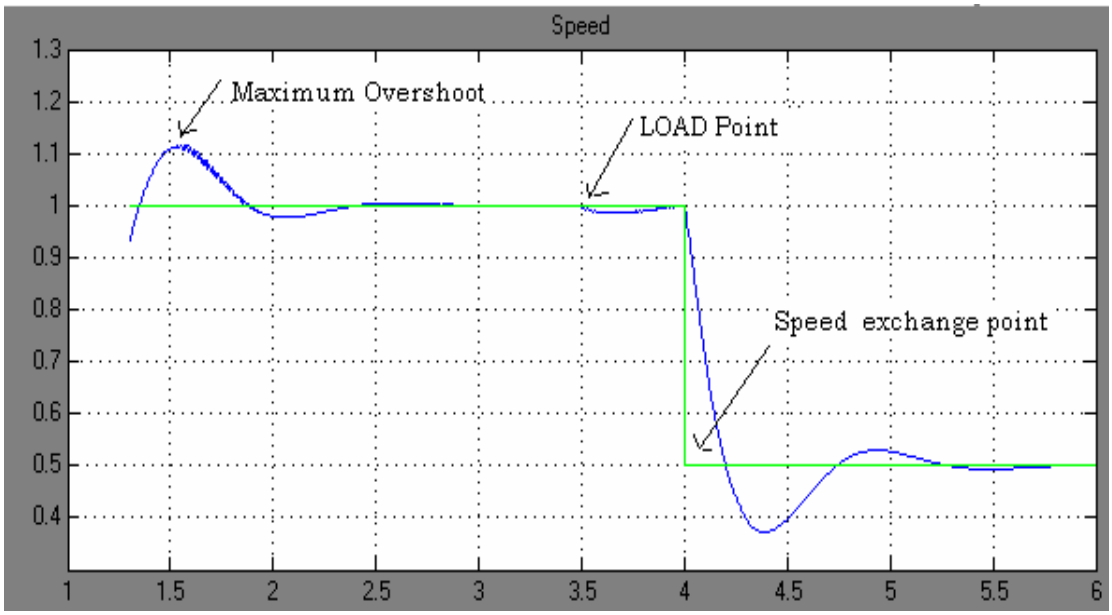


Figure 5.50: Fuzzy-PID controller step response with load and speed variation.

5.4 Results comparison

Table 5.5 explains controllers comparison, hence from table results we find that the Fuzzy-PID controller is the best between these controllers.

Table 5.5: Controllers comparison.

Controller	System O.S	System S.T	S.T after speed 2	O.S after speed 2	O.S after load applied
PID	**13%**	**2.5S**	**2.25S**	**30%**	**2%**
Fuzzy	**11%**	**2. 0s**	**2.1S**	**23%**	**2%**
Fuzzy-PID	**9%**	**1.5s**	**1.7S**	**13%**	**2%**

In the lab I used the Feedback machine which appears in Figure 5.51 to implement the load in N.M using dynamometer device shown in Figure 5.52.

Tachometer device shown in Figure 5.53 used to scale the real speed value.

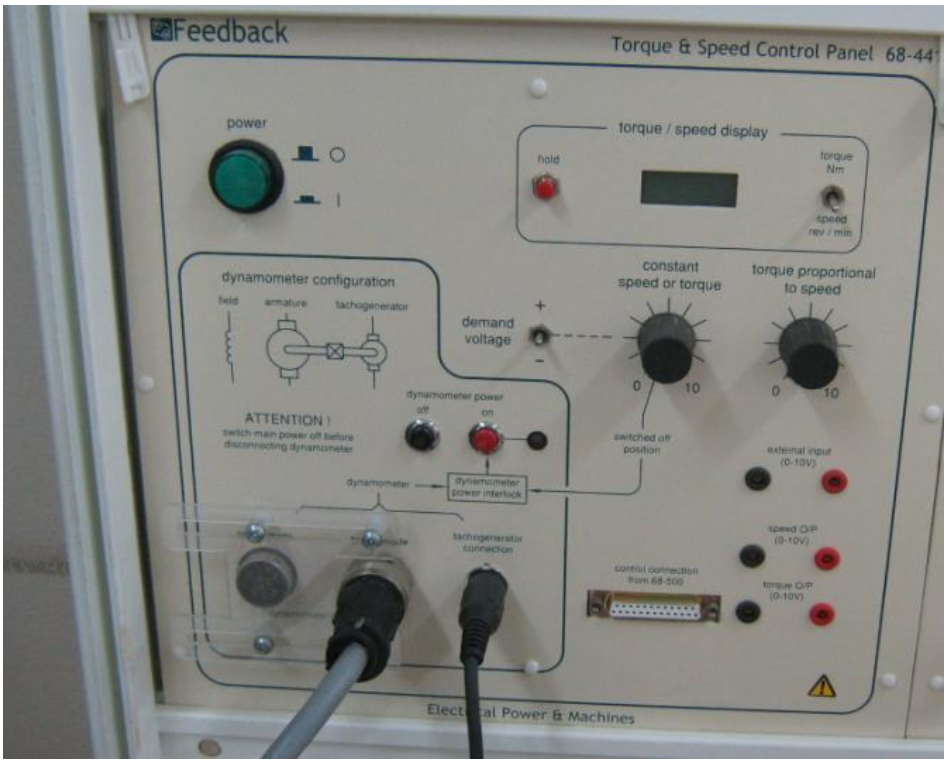


Figure 5.51: Feedback torque unit.

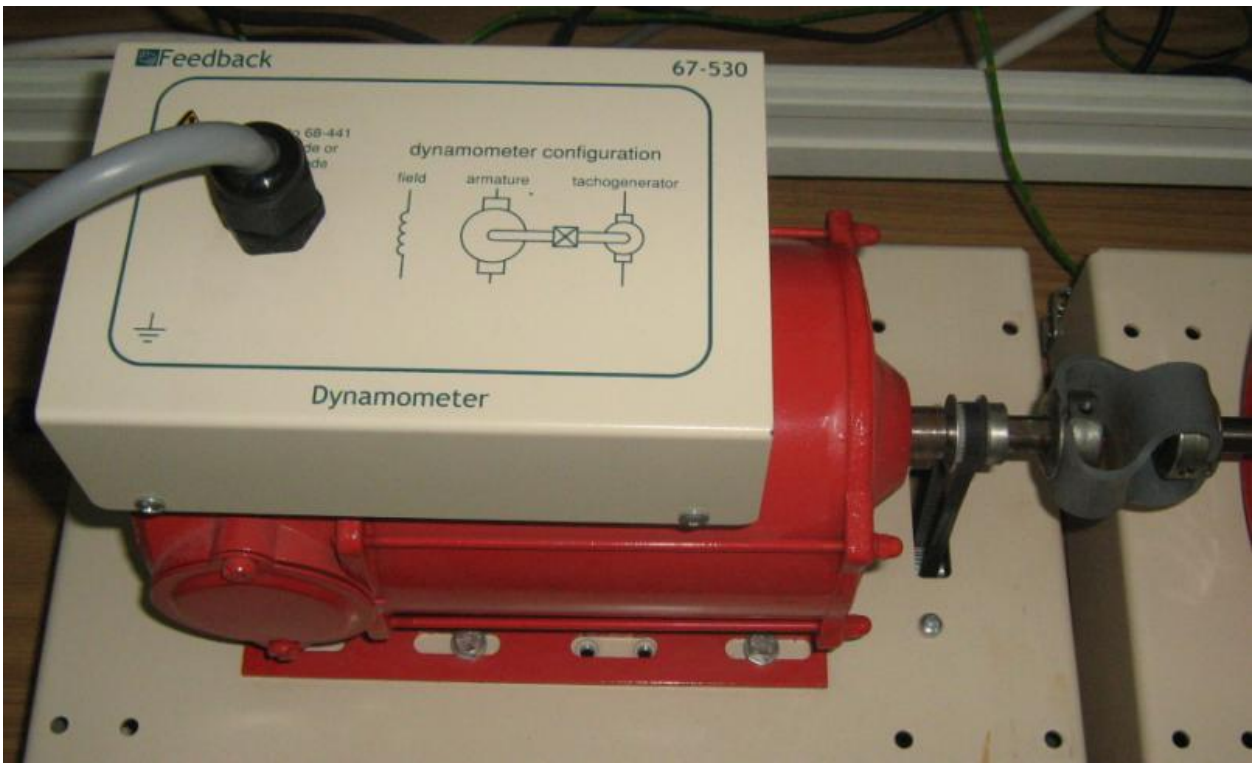


Figure 5.52: Dynamometer coupled with induction motor.

Figure 5.53: Tachometer for speed scaling.

For real time testing I used the digital storage oscilloscope shown in Figure 5.54

Figure 5.54: Storage oscilloscope.

Figure 5.55 shown the over all system.

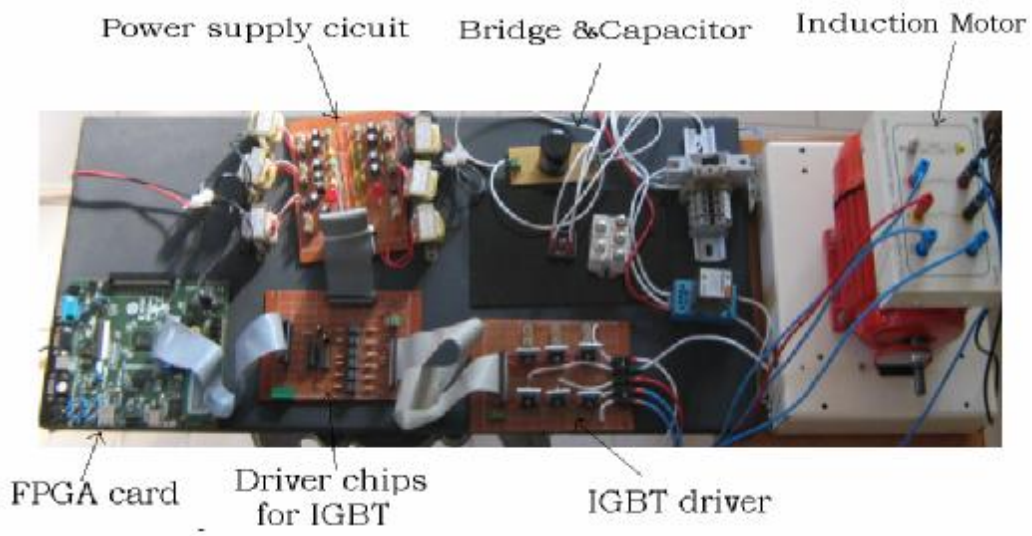


Figure 5.56: Overall system.

5.5 Modelsim results

PWM inverter software was tested using Modelsim program. Each pairs of top and bottom PWM can be mixed together using ac approach as shown in Figure 5.56.

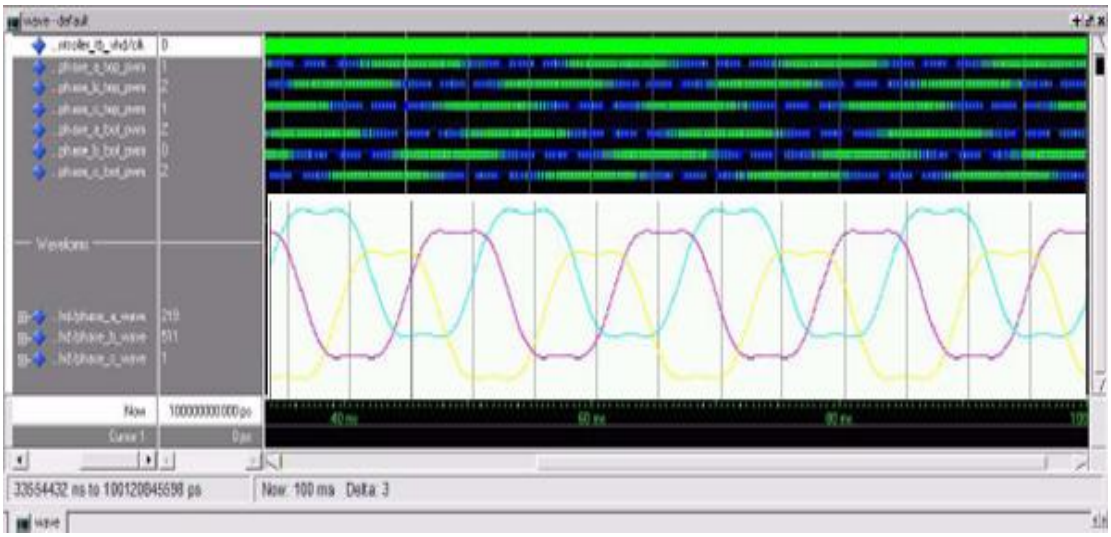


Figure 5.56: Three-phase simulation using Modelsim.

5.6 VHDL and PicoBalze software

The over all software contains two main parts:

VHDL algorithm: For controllers blocks and for user interface units available on the FPGA card like rotating switch needed to control with the speed of the induction motor and sliding switches to select between the controllers and to specify the K_p, K_i and K_d parameter in PID controller.

PicoBlaze algorithm which contains the LCD program and some auxiliary functions built for controllers.

CHAPTER 6

Conclusions and Future Research

1.1 Conclusions

In recent years, fuzzy logic control has been suggested as an alternative approach to conventional process control techniques.

Since the first paper on fuzzy sets was published, fuzzy logic control has attracted great attention from both the academic and industrial communities. Much progress has been made in successfully applying FLC in industrial control systems. FLC techniques represent means of both collecting human knowledge and expertise and dealing with uncertainties in the process of control. They have a number of advantages. Although fuzzy logic control is not the solution for all problems, it can play an important role in making automated processes more intelligent.

In this research, fuzzy logic controller FLC has been selected to control the speed of three-phase induction motor (squirrel cage type) due to its advantages over the traditional PID controller. This wide-spreading of FLC's appears because these kind of controllers have been a very effective techniques for complicated and imprecise processes for which either no mathematical model exists or the mathematical model is severely nonlinear as induction motor.

Modern FPGA card (Spartan 3-A) was implemented in Xilinx company in 2007, which represents headquarter in the FPGA manufacturing companies, have been chosen in this research as a digital controllers board builder.

In this research, three different types of controllers (PID, Fuzzy, and Fuzzy-PID) were constructed in FPGA card, which was used as a speed controller for three-phase induction motor (squirrel cage type). These controllers have been tested using Matlab/Simulink program under speed and load variation conditions. The comparison results show that the Fuzzy-PID controller was the best of them.

Xilinx Company provided very attractive package software "ISE9.1 and Modelsim" for programming purpose, this software was used to implement the three different

controllers, and to generate the six PWM pulses feeding to the induction motor drivers boards.

In this research, the algorithm for the three-phase induction motor inverter was constructed in the FPGA card, also the hardware equipments needed for this driver was constructed which contains: IGBTs board, IGBT driver chip, and the power supply board.

1.2 Future research

In this thesis, Mamdani method was used to implement the fuzzy control rules, it can be replaces by Sugeno approach and compare it with Mamdani. Also a good area of research is using optimization method to reduce the rules of the controller such as using Genetic Algorithm with fuzzy controllers. Genetic algorithms are search algorithm based on natural genetics. There are used in the control algorithm to tune the membership functions so that the inexact reasoning characteristics of the FLC are sufficient to control a system that requires precise control actions. On of the more important areas in control is stability; more work can be done on the stability of the fuzzy controller.

Bibliography

[1] Microchip Company, AC Induction Motor Fundamentals, technical data, AN887, 2003.

[2] M. Lai, C. Chang, and W. Chiou "Design of fuzzy logic controller for an induction motor speed drive," IEEE, Vol.113, No.13, 1997, 1071 – 1076.

[3] C. Shanmei, W. Shuyun, and D. Zhengeheng " Fuzzy speed controller of induction motors," IEEE, Vol.145, No.15, 2004, 747- 750

[4] X. Fu and J. Dong, ‘‘ The decoupling-variable structure adjustable-speed control system for induction motor,” IPEC’93. Singapore, V1.2, pp466- 471,1993.

[5] Fayez F. M. El-Sousy and Maged N. F. Nashed " Robust fuzzy logic current and speed controllers for field-oriented induction motor drive," The Korean Institute of Power Electronics (KIPE), Journal of Power Electronics (JPE), Vol. 3, No. 2, pp. 115-123, April, 2003.

[6] Tipsuwanpom, Runghimmawan., T Runghim, T. Intajag, and S. Krongratana "Fuzzy logic PID controller based on FPGA for process control," IEEE, Vol.2 , No.11, 2004, 1495- 1500.

[7] Volcanjk, and Jezernik" Induction motor control with PI-load estimator**,**" IEEE, Vol.3 , No. 2 , 1994, 1097-1100.

[8] Zidani, Benbouzid, and Diallo " Fuzzy efficient-optimization controller for induction motor drives," IEEE, Vol.20, No. 10, 2000,43-44.

[9] Shi, Chan, and Wong "A novel hybrid fuzzy/PI two-stage controller for an induction motor drive," IEEE, Vol.1, No.1, 2001, 415 – 421.

[10] W.Yong Han, S. Kim, and C. Lee " Sensorless vector control of induction motor using improved self-tuning fuzzy PID controller," IEEE, Vol.3, No.1, 2003. 3112-3117.

[11] Lin, Wang, and Huang " FPGA-based fuzzy sliding-mode control for a linear induction motor drive," IEEE,Vol.152, No. 5, 2005, 1137 – 1148.

[12] T.V.S.Urmila Priya , K.Udaya Kumar and S.Renganarayanan ,present in 2005 “FPGA based fuzzy logic controller for dc electric vehicle," S.Poorani Journal of The Institution of Engineers, Singapore Vol. 45 Issue 5 2005.

[13] Zhang and Collins " Digital anti-windup PI controllers for variable-speed motor drives using FPGA and stochastic theory," IEEE**,** Vol.21, No.5, 2006, 1496 – 1501.

[14] S.K Bhattacharya, Electrical Machines, 2th Edition, McGraw-Hill, 1998.

[15] Wildi, T., Electrical Machines, Drives and Power Systems, 5th Edition, Prentice-Hall, 2002.

[16] Feedback Company, Variable Frequency Drive Trainer, Technical data, 66-003, 2006.

[17] Microchip Company, Speed Control of 3-Phase Induction Motor Using PIC18 Microcontrollers, technical data, AN843, 2002.

[18] Bhag S. Guru & Huseyin R. Hizirglu, Electric Machinery and Transformers, Oxford University Press, 3th Edition , 2001.

[19] L. A. Zadeh, "Fuzzy sets," Information and control, vol. 8, pp. 338-353, 1965.

[20] K. M. Passion and S. Yurkovich, Fuzzy Control .Addison-Wesley,1998.

[21] F. Mcneill and E. Thro, Fuzzy Logic: a practical Approach. AP Professional, 1994

[22] L.X. Wang, A Course in Fuzzy Systems and Controls. Englewood Cliffs: Prentice-Hall, 1997.

[23] E H Mamdani, "Application of Fuzzy algorithms for the control of a dynamic plant" IEE vol. 121, pp. 1585-1588, 1974.

[24] E. H. Mamdani and S. Assilian, "An experiment with in linguistic synthesis with a fuzzy logic controller," International journal of Man-Machine studies, vol. 7, pp. 1-13, 1975.

[25] L. A. Zadeh, "Fuzzy logic = computing with words," IEEE Transactions on Fuzzy Systems, vol. 4, pp. 103–111, May 1996.

[26] H.-J. Zimmermann, Fuzzy Sets Theory - and Its Applications, Kluwer Academic Publishers, 1990.

[27] T. Takagi and M. Sugeno, Derivation of fuzzy control rules from human operator's control actions, Proc. of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis, p. 55-60, July 1983

[28] E.H. .Mamdani "Application of Fuzzy algorithms for control of simple dynamic plant," IEEE, Vol.121,no.12,1976,1585-1588.

[29] L.A. Zadeh, "Fuzzy logic and approximate reasoning," syntheses, Vol.30, 1975, 407-428.

[30] J. F. Baldwin,"Amodel of FUZZY reasoning through multi-valved logic and set theory," International journal of Man-Machine Studies, vol 11,1979, 351-380.

[31] Michio Sugeno,"Anew approach to design of Fuzzy controller," P.P wany ed.,Advance in Fuzzy Sets, Possibility Theory, and Application,Plenum Press,1983.

[32] T. Takagi and M.Sugeno,"Fuzzy identification of systems and its applications to modeling and control," IEEE Transactions on systems, Man and Cybernetics, vol.Smc-15, no, 1985.

[33] R. Jastrzebski, O. Pyhonen, A. Napieralski, "FPGA based platform for real-time testing of fast induction motor controllers", Proceedings of the 11th International Conference on Mixed Design (MIXDES 2004), Szczecin, Poland, June24-26, 2004, pp. 491-496.

[34] R. Jastrzebski, O. Laakkonen, K. Rauma, J. Luukko, H. Saren, O. Pyhonen, "Real-Time Emulation of Induction Motor in FPGA using Floating Point Representation", Proceedings of the IASTED International Conference on Applied Simulation and Modelling, June 28-30, Rhodes, Greece, pp. 226-231.

[35] J. Birkner et al, "A very-high-speed field-programmable gate array using metal-tometal antifuse programmable elements," Microelectronics Journal, v. 23, pp. 561-568.

[36] Xilinx Company, Programmable Logic Design, Technical data, Logic_handbook 2006.

[37] Xilinx Company, Spartan-3A starter kit –Board User Guide,Technical data, UG 330, 2007.

[38] Xilinx Company, PicoBlaze, Technical data, Kcpsm3,2003.

[39] Ying-Yu Tzou and Hau-Jean Hsu, "FPGA realization of Space-Vector PWM control IC for three-phase pwm inverters", IEEE transactions on power electronics, Vol. 12, No. 6,Novemeber 1997.

[40] Xilinx Company, Logic-Based AC Induction Motor Controller, Technical data, xapp448,2005.

[41] B. Kuo, *Automatic Control Systems*, Englewood Cliffs: Prentice Hall, 1995.

[42] P. Asheaden " The designer's guide to VHDL , Morgan Kaufmann," 1998.

Appendices

1.1 Appendix A

The key features of the Spartan-3A Starter Kit

- The key features of the Spartan-3A Starter Kit board are:
- Xilinx 700K-gate XC3S700A Spartan-3A FPGA in the Pb-free 484-ball BGA package (FGG484)
- 4 Mbit Xilinx Platform Flash configuration PROM
- 64 MByte (512 Mbit) of DDR2 SDRAM, 32Mx16 data interface
- 4 MByte (32 Mbit) of parallel NOR Flash
- FPGA configuration storage
- MicroBlaze code storage/shadowing
- x8 or x16 data interface after configuration
- 16 Mbits of SPI serial Flash
- Choose either the STMicroelectronics or the Atmel DataFlash serial architectures
- FPGA configuration storage
- MicroBlaze code shadowing
- Two-line, 16-character LCD screen
- PS/2 mouse or keyboard port
- VGA display port
- 10/100 Ethernet PHY (requires Ethernet MAC in FPGA)
- Two nine-pin RS-232 ports (DTE- and DCE-style)
- On-board USB-based programming solution
- FPGA download/debug
- SPI serial Flash in-system direct programming
- 50 MHz clock oscillator
- 8-pin DIP socket for second oscillator
- SMA connector for clock inputs or outputs
- 100-pin Hirose FX2 expansion connector with up to 43 FPGA user I/Os
- Compatible with Digilent FX2 add-on cards
- High-speed differential I/O connectors
- Receiver: Five data channels plus clock
- Transmitter: Six data channels or five data channels plus clock
- Supports multiple differential I/O standards, including LVDS, RSDS, mini-LVDS
- Also supports up to 24 single-ended I/O
- Uses widely available 34-conductor cables
- Three six-pin expansion connectors for Digilent Peripheral Modules
- Four-output, SPI-based Digital-to-Analog Converter (DAC)
- Two-input, SPI-based Analog-to-Digital Converter (ADC) with programmable-gain pre-amplifier
- Stereo audio jack using digital I/O pins
- ChipScope™ SoftTouch debugging port
- Rotary-encoder with push-button shaft
- Eight discrete LEDs
- Four slide switches
- Four push-button switches

1.2 Appendix B

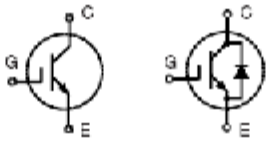
IXYS

High Voltage IGBT with optional Diode

IXDH 20N120
IXDH 20N120 D1

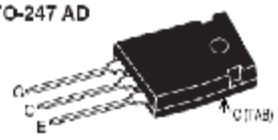
V_{CES} = 1200 V
I_{C25} = 38 A
$V_{CE(sat)\ typ}$ = 2.4 V

Short Circuit SOA Capability
Square RBSOA



IXDH 20N120 IXDH 20N120 D1

TO-247 AD



G = Gate, C = Collector, E = Emitter, TAB = Collector

Symbol	Conditions		Maximum Ratings	
V_{CES}	T_J = 25°C to 150°C		1200	V
V_{CGR}	T_J = 25°C to 150°C; R_{GE} = 20 kΩ		1200	V
V_{GES}	Continuous		±20	V
V_{GEM}	Transient		±30	V
I_{C25}	T_C = 25°C		38	A
I_{C90}	T_C = 90°C		25	A
I_{CM}	T_C = 90°C, t_p = 1 ms		50	A
RBSOA	V_{GE} = ±15 V, T_{VJ} = 125°C, R_G = 82 Ω Clamped inductive load, L = 30 µH		I_{CM} = 35 $V_{CEK} < V_{CES}$	A
t_{SC} (SCSOA)	V_{GE} = ±15 V, V_{CE} = V_{CES}, T_J = 125°C R_G = 82 Ω, non repetitive		10	µs
P_C	T_C = 25°C	IGBT Diode	200 75	W W
T_J			-55 ... +150	°C
T_{stg}			-40< ... +150	°C
Maximum lead temperature for soldering 1.6 mm (0.062 in.) from case for 10 s			300	°C
M_d	Mounting torque		0.8 - 1.2	Nm
Weight			6	g

Symbol	Conditions		Characteristic Values (T_J = 25°C, unless otherwise specified) min.	typ.	max.	
$V_{(BR)CES}$	V_{GE} = 0 V		1200			V
$V_{GE(th)}$	I_C = 0.6 mA, V_{CE} = V_{GE}		4.5		6.5	V
I_{CES}	V_{CE} = V_{CES}	T_J = 25°C T_J = 125°C		2	1	mA mA
I_{GES}	V_{CE} = 0 V, V_{GE} = ±20 V				± 500	nA
$V_{CE(sat)}$	I_C = 20 A, V_{GE} = 15 V			2.4	3	V

Features

- NPT IGBT technology
- low saturation voltage
- low switching losses
- square RBSOA, no latch up
- high short circuit capability
- positive temperature coefficient for easy paralleling
- MOS input, voltage controlled
- optional ultra fast diode
- International standard package

Advantages

- Space savings
- High power density

Typical Applications

- AC motor speed control
- DC servo and robot drives
- DC choppers
- Uninterruptible power supplies (UPS)
- Switch-mode and resonant-mode power supplies

1.3 Appendix C

2.0 Amp Output Current IGBT Gate Drive Optocoupler

Technical Data

HCPL-3120
HCPL-J312
HCNW3120

Features

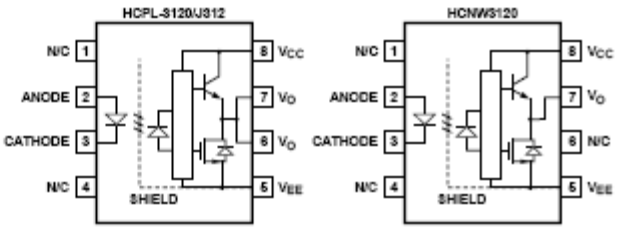
- **2.0 A Minimum Peak Output Current**
- **15 kV/μs Minimum Common Mode Rejection (CMR) at V_{CM} = 1500 V**
- **0.5 V Maximum Low Level Output Voltage (V_{OL})** Eliminates Need for Negative Gate Drive
- **I_{CC} = 5 mA Maximum Supply Current**
- **Under Voltage Lock-Out Protection (UVLO) with Hysteresis**
- **Wide Operating V_{CC} Range: 15 to 30 Volts**
- **500 ns Maximum Switching Speeds**
- **Industrial Temperature Range:** -40°C to 100°C
- **Safety Approval**
  UL Recognized
  2500 Vrms for 1 min. for HCPL-3120
  3750 Vrms for 1 min. for HCPL-J312
  5000 Vrms for 1 min. for HCNW3120

CSA Approval
VDE 0884 Approved
V_{IORM} = 630 Vpeak for HCPL-3120 (Option 060)
V_{IORM} = 891 Vpeak for HCPL-J312
V_{IORM} = 1414 Vpeak for HCNW3120
BSI Certified (HCNW3120 only) (Pending)

Applications

- **IGBT/MOSFET Gate Drive**
- **AC/Brushless DC Motor Drives**
- **Industrial Inverters**
- **Switch Mode Power Supplies**

Functional Diagram



TRUTH TABLE

LED	V_{CC} - V_{EE} "POSITIVE GOING" (i.e., TURN-ON)	V_{CC} - V_{EE} "NEGATIVE GOING" (i.e., TURN-OFF)	V_O
OFF	0 - 30 V	0 - 30 V	LOW
ON	0 - 11 V	0 - 9.5 V	LOW
ON	11 - 13.5 V	9.5 - 12 V	TRANSITION
ON	13.5 - 30 V	12 - 30 V	HIGH

1.4 Appendix D

Control rule base for FUZZY controller

1	-	IF	E	Is	NB	And	CE	Is	NB	Then	Output	NB
2	-	IF	E	Is	NB	And	CE	Is	NM	Then	Output	NB
3	-	IF	E	Is	NB	And	CE	Is	NZ	Then	Output	NB
4	-	IF	E	Is	NB	And	CE	Is	ZE	Then	Output	NB
5	-	IF	E	Is	NB	And	CE	Is	PS	Then	Output	NM
6	-	IF	E	Is	NB	And	CE	Is	PM	Then	Output	NS
7	-	IF	E	Is	NB	And	CE	Is	PB	Then	Output	ZE
8	-	IF	E	Is	NM	And	CE	Is	NB	Then	Output	NB
9	-	IF	E	Is	NM	And	CE	Is	NM	Then	Output	NB
10	-	IF	E	Is	NM	And	CE	Is	NZ	Then	Output	NM
11	-	IF	E	Is	NM	And	CE	Is	ZE	Then	Output	NM
12	-	IF	E	Is	NM	And	CE	Is	PS	Then	Output	NS
13	-	IF	E	Is	NM	And	CE	Is	PM	Then	Output	ZE
14	-	IF	E	Is	NM	And	CE	Is	PB	Then	Output	PS
15	-	IF	E	Is	NS	And	CE	Is	NB	Then	Output	NB
16	-	IF	E	Is	NS	And	CE	Is	NM	Then	Output	NM
17	-	IF	E	Is	NS	And	CE	Is	NZ	Then	Output	NS
18	-	IF	E	Is	NS	And	CE	Is	ZE	Then	Output	NS
19	-	IF	E	Is	NS	And	CE	Is	PS	Then	Output	ZE
20	-	IF	E	Is	NS	And	CE	Is	PM	Then	Output	PS
21	-	IF	E	Is	NS	And	CE	Is	PB	Then	Output	PM
22	-	IF	E	Is	ZE	And	CE	Is	NB	Then	Output	NB
23	-	IF	E	Is	ZE	And	CE	Is	NM	Then	Output	NM
24	-	IF	E	Is	ZE	And	CE	Is	NZ	Then	Output	NS
25	-	IF	E	Is	ZE	And	CE	Is	ZE	Then	Output	ZE
26	-	IF	E	Is	ZE	And	CE	Is	PS	Then	Output	PS
27	-	IF	E	Is	ZE	And	CE	Is	PM	Then	Output	PM
28	-	IF	E	Is	ZE	And	CE	Is	PB	Then	Output	PB
29	-	IF	E	Is	PS	And	CE	Is	NB	Then	Output	NM
30	-	IF	E	Is	PS	And	CE	Is	NM	Then	Output	NS
31	-	IF	E	Is	PS	And	CE	Is	NZ	Then	Output	ZE
32	-	IF	E	Is	PS	And	CE	Is	ZE	Then	Output	PS
33	-	IF	E	Is	PS	And	CE	Is	PS	Then	Output	PS
34	-	IF	E	Is	PS	And	CE	Is	PM	Then	Output	PM
35	-	IF	E	Is	PS	And	CE	Is	PB	Then	Output	PB
36	-	IF	E	Is	PM	And	CE	Is	NB	Then	Output	NS
37	-	IF	E	Is	PM	And	CE	Is	NM	Then	Output	ZE
38	-	IF	E	Is	PM	And	CE	Is	NZ	Then	Output	PS
39	-	IF	E	Is	PM	And	CE	Is	ZE	Then	Output	PM
40	-	IF	E	Is	PM	And	CE	Is	PS	Then	Output	PM
41	-	IF	E	Is	PM	And	CE	Is	PM	Then	Output	PB
42	-	IF	E	Is	PM	And	CE	Is	PB	Then	Output	PB
43	-	IF	E	Is	PB	And	CE	Is	NB	Then	Output	ZE
44	-	IF	E	Is	PB	And	CE	Is	NM	Then	Output	PS
45	-	IF	E	Is	PB	And	CE	Is	NZ	Then	Output	PM
46	-	IF	E	Is	PB	And	CE	Is	ZE	Then	Output	PB
47	-	IF	E	Is	PB	And	CE	Is	PS	Then	Output	PB
48	-	IF	E	Is	PB	And	CE	Is	PM	Then	Output	PB
49	-	IF	E	Is	PB	And	CE	Is	PB	Then	Output	PB

1.5 Appendix E

Control rule base for FUZZY controller

1	-	IF	E	Is	PB	And	CE	Is	PB	Then	Output	ZO
2	-	IF	E	Is	PB	And	CE	Is	PM	Then	Output	NS
3	-	IF	E	Is	PB	And	CE	Is	PS	Then	Output	NM
4	-	IF	E	Is	PB	And	CE	Is	ZE	Then	Output	NB
5	-	IF	E	Is	PB	And	CE	Is	NS	Then	Output	NB
6	-	IF	E	Is	PB	And	CE	Is	NM	Then	Output	NB
7	-	IF	E	Is	PB	And	CE	Is	NB	Then	Output	NB
8	-	IF	E	Is	NM	And	CE	Is	PB	Then	Output	PS
9	-	IF	E	Is	NM	And	CE	Is	PM	Then	Output	ZO
10	-	IF	E	Is	NM	And	CE	Is	PS	Then	Output	NS
11	-	IF	E	Is	NM	And	CE	Is	ZE	Then	Output	NM
12	-	IF	E	Is	NM	And	CE	Is	NS	Then	Output	NB
13	-	IF	E	Is	NM	And	CE	Is	NM	Then	Output	NB
14	-	IF	E	Is	NM	And	CE	Is	NB	Then	Output	NB
15	-	IF	E	Is	NS	And	CE	Is	PB	Then	Output	PM
16	-	IF	E	Is	NS	And	CE	Is	PM	Then	Output	PS
17	-	IF	E	Is	NS	And	CE	Is	PS	Then	Output	ZO
18	-	IF	E	Is	NS	And	CE	Is	ZE	Then	Output	NS
19	-	IF	E	Is	NS	And	CE	Is	NS	Then	Output	NM
20	-	IF	E	Is	NS	And	CE	Is	NM	Then	Output	NB
21	-	IF	E	Is	NS	And	CE	Is	NB	Then	Output	NB
22	-	IF	E	Is	ZE	And	CE	Is	PB	Then	Output	PB
23	-	IF	E	Is	ZE	And	CE	Is	PM	Then	Output	PM
24	-	IF	E	Is	ZE	And	CE	Is	PS	Then	Output	PS
25	-	IF	E	Is	ZE	And	CE	Is	ZE	Then	Output	ZO
26	-	IF	E	Is	ZE	And	CE	Is	NS	Then	Output	NS
27	-	IF	E	Is	ZE	And	CE	Is	NM	Then	Output	NM
28	-	IF	E	Is	ZE	And	CE	Is	NB	Then	Output	NB
29	-	IF	E	Is	PS	And	CE	Is	PB	Then	Output	PB
30	-	IF	E	Is	PS	And	CE	Is	PM	Then	Output	PB
31	-	IF	E	Is	PS	And	CE	Is	PS	Then	Output	PM
32	-	IF	E	Is	PS	And	CE	Is	ZE	Then	Output	PS
33	-	IF	E	Is	PS	And	CE	Is	NS	Then	Output	ZO
34	-	IF	E	Is	PS	And	CE	Is	NM	Then	Output	NS
35	-	IF	E	Is	PS	And	CE	Is	NB	Then	Output	NM
36	-	IF	E	Is	PM	And	CE	Is	PB	Then	Output	PB
37	-	IF	E	Is	PM	And	CE	Is	PM	Then	Output	PB
38	-	IF	E	Is	PM	And	CE	Is	PS	Then	Output	PB
39	-	IF	E	Is	PM	And	CE	Is	ZE	Then	Output	PM
40	-	IF	E	Is	PM	And	CE	Is	NS	Then	Output	PS
41	-	IF	E	Is	PM	And	CE	Is	NM	Then	Output	ZO
42	-	IF	E	Is	PM	And	CE	Is	NB	Then	Output	NS
43	-	IF	E	Is	PB	And	CE	Is	PB	Then	Output	PB
44	-	IF	E	Is	PB	And	CE	Is	PM	Then	Output	PB
45	-	IF	E	Is	PB	And	CE	Is	PS	Then	Output	PB
46	-	IF	E	Is	PB	And	CE	Is	ZE	Then	Output	PB
47	-	IF	E	Is	PB	And	CE	Is	NS	Then	Output	PM
48	-	IF	E	Is	PB	And	CE	Is	NM	Then	Output	PS
49	-	IF	E	Is	PB	And	CE	Is	NB	Then	Output	ZO