HARDWARE IMPLEMENTATION OF V/f CONTROL OF INDUCTION MOTOR USING HSRPEC

DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF
MASTER OF TECHNOLOGY
IN
CONTROL AND INSTRUMENTATION

Submitted by:

Animesh Kumar Patra
2K17/C&I/03

Under the supervision of
Prof. Madhusudan Singh

DEPARTMENT OF ELECTRICAL ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

2019

DEPARTMENT OF ELECTRICAL ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I, Animesh Kumar Patra, Roll No. 2K17/C&I/03 student of M.Tech. Control and Instrumentation (C&I), hereby declare that the dissertation/project titled **"HARDWARE IMPLEMENTATION OF V/f CONTROL OF INDUCTION MOTOR USING HSRPEC"** under the supervision of Prof. Madhusudan Singh of Electrical Engineering Department, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology has not been submitted elsewhere for the award of any Degree.

.

Place: Delhi **(ANIMESH KUMAR PATRA)**

Date:

Prof. Madhusudan Singh
(SUPERVISOR)
Professor
Department of Electrical Engineering
Delhi Technological University

ACKNOWLEDGEMENT

I am highly grateful to the Department of Electrical Engineering, Delhi Technological University (DTU) for providing this opportunity to carry out this project work.

The constant guidance and encouragement received from my supervisor Prof. Madhusudan Singh of Department of Electrical Engineering, DTU, has been of great help in carrying my present work and is acknowledged with reverential thanks.

I would like to thank Mr. Suryakant Shukla (PhD scholar), Mrs. Pallavi Verma (PhD scholar), Mr. Avdhesh Kumar and Mr. Harsh Chaturvedi (M.tech Student) for their guidance and continuous support in completion of this project work.

I would like to express a deep sense of gratitude and thanks to Prof. Rachna Garg for allowing me to work in Smart Grid laboratory to carry out this project work.

Finally, I would like to expresses gratitude to all faculty members of Electrical Engineering Department, DTU for their intellectual support in my M.tech study at DTU.

ANIMESH KUMAR PATRA
2K17/C&I/03
M. Tech. (Control and Instrumentation)
Delhi Technological University

ABSTRACT

FPGAs is an Integrated Circuit designed to be configured by a user or designer. They are Integrated Circuits which can be field programmed even after manufacturing. They are basically used to design specialized circuit and test them so that the ICs can be manufactured for bulk use by manufacturers. The idea is to develop reconfigurable environment rather than conventional embedded environment.

The basic aim of this project is to learn the FPGAs and develop controller for implementation of V/f control of induction motor using FPGA(HSRPEC) which is designed by CDAC. The processor used is ALTERA Cyclone III EP3C25E144C8N. The designing of system is carried out on Quartus platform and the coding is developed in NIOS platform.

For analysis of induction motor drives two different frequencies are considered to change the V/f ratio, and varying loads are used to compare the motor torque. A tacho generator is used to monitor the speed of motor for close loop operation. The output of speed sensor is fed as feedback to the ADC channel of HSRPEC. The actual speed is compared to the reference speed and error is used for the generation of PWM. This PWM signal are used to control the inverter which drives a 3-phase induction motor at desired speed.

Simulation of power Electronics systems enable prior understanding of the system behavior before real time model are developed. Simulation study includes

1. Their offline simulation on digital computer.
2. Real time simulation.

These simulation study help for the design and performance evaluation of controller for the Power Electronic systems.

The real time simulation allows in to perform Hardware in Loop (HIL) simulation to test physical controllers which reduces the cost of field testing. A High Speed Reconfigurable Power Electronics Controller (HSRPEC), which can replace conventional DSP based controller design for real time control and monitoring of Power Electronics system is being used to implement controller for high performance induction motor drive.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

DTC	Direct Torque Control
FOC	Field Oriented Control
MPC	Model Predictive Control
MMF	Magneto Motive Force
EMF	Electro Motive Force
IM	Induction Motor
PI	Proportional-Integral
AC	Alternating Current
DC	Direct current
EV	Electric Vehicles
FPGA	Field Programmable Gate Arrays
ASIC	Application Specific Integrated Circuit
PLD	Programmable Logic Design
HSRPEC	High Speed Re-Configurable Power Electronics Controller
CAD	Computer Aided Design
HDL	Hardware Description Languages
DSP	Digital Signal Processors

PWM	Pulse Width Modulation
CLB	Configurable Logic Block
SOPC	System on Programmable Chip
PE	Power Electronic
IPs	Intellectual Properties
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
PFC	Power Factor Correction
PIO	Parallel Input Output
IRQ	Interrupt Request
IDE	Integrated Development Environment
ISA	Instruction Set Architecture
DMIPS	Dhrystone Million Instruction per Second
VSI	Voltage Source Inverter

LIST OF SYMBOLS

T_d	Developed Torque
T_l	Load Torque
ω_m	Angular frequency of Induction Motor
ω_{slip}	Slip angular frequency
i_a	Armature Current
i_{abc}	Three phase current vector
$i_{\alpha\beta\gamma}$	Two phase stationary reference frame current vector
i_{dq0}	Two phase rotating reference frame current vector
Rs	Stator Resistance
Rr	Rotor Resistance
Lr	Rotor self-Inductance
Ls	Stator self-Inductance
Lm	Mutual Inductance of Induction Motor
i_q	Q axis current (Rotating frame of reference)
i_d	D axis current (Rotating frame of reference)

i_α	α axis current (Stationary frame of reference)
i_β	β axis current (Stationary frame of reference)
Vq	Q axis voltage (Rotating frame of reference)
Vd	D axis voltage (Rotating frame of reference)
$V\alpha$	α axis voltage (Stationary frame of reference)
$V\beta$	β axis voltage (Stationary frame of reference)
λ	Lambda (tuning parameter)
φ_s	Total stator flux
φ_r	Rotor flux
φ_m	Magnetizing component of flux

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION

Induction Motors (IM) are widely used in all industrial applications because of their low cost, rugged and robust performance. The performance of Induction Motor is very promising in variable speed applications. Conventionally DC motor were extensively used for variable speed application due to their easy and flexible speed control. The torque and speed in DC motor are decoupled and can be controlled independently.
In DC machines MMF axis is developed at 90˚ electrical to the main field axis. The electromagnetic torque is directly proportional to the product of armature current and field flux. The field flux is proportional to the field current due to orthogonal orientation between armature MMF and field MMF, and is unaffected by the armature current. Thus, the torque in a separately excited DC machine is directly proportional to the armature current, with a constant value of field flux. Direct control of armature current in DC motor allows direct control of torque and quick response however, in Induction Motor the flux and torque components of stator current are mutually coupled, so the control of Induction Motor is more complex. The DC motor is not economical as it has the additional cost of maintenance of brushes, brush holders, commutators etc. Now a days Induction Motor are used in place of the DC motor. To decouple the flux and torque component of stator current in Induction Motor vector control is a popular method for Induction Motor [25].

Induction Motor control can be classified broadly into two parts:
1) Scalar control of Induction Motor
2) Vector control of Induction Motor
These control schemes can be further divided, Fig.1 shows the various control schemes for Induction Motor.

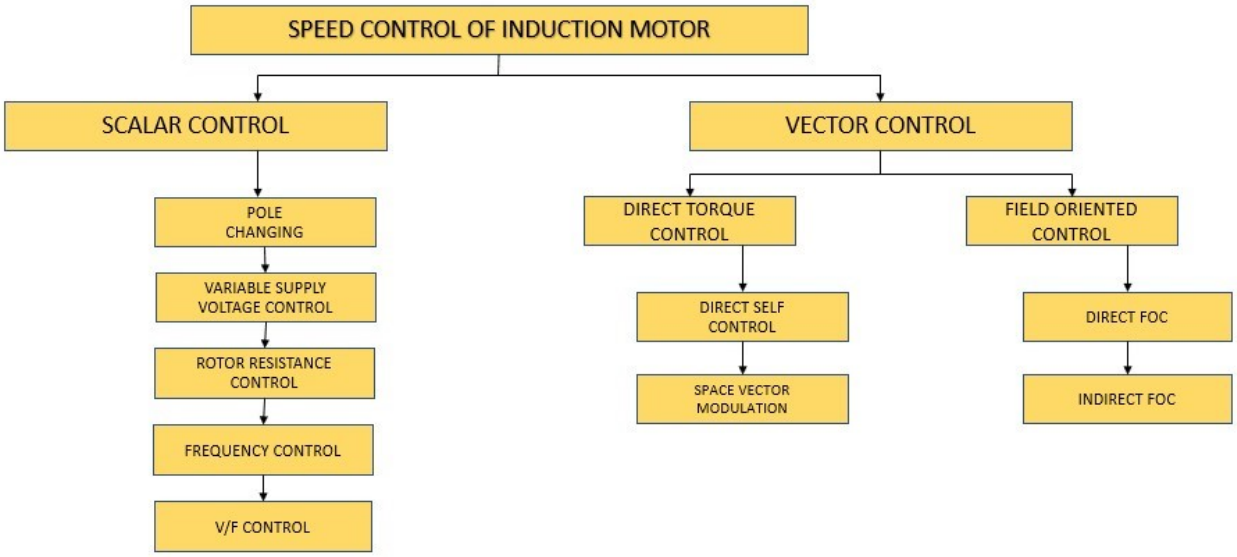


Fig.1. Classification of control methods used for speed control in Induction Motor

1.2. SPEED CONTROL OF INDUCTION MOTOR

An Induction Motor is practically a constant speed motor, that means, change in speed of the motor is relatively small, for the entire loading range. Speed of a DC motor can be easily varied with good efficiency, but in case of Induction Motors, speed reduction is accompanied by a corresponding loss of efficiency and poor power factor. As induction motors are widely being used, their speed control may be required in many applications. Various induction motor speed control techniques are explained below.

1.2.1. Scalar control of Induction Motor

Various methods for scalar control of Induction Motor are [23].
1) Pole changing method
2) Variable voltage control
3) Rotor resistance control
4) Variable frequency control
5) V/f control

1.2.1.1. Pole changing method

Speed control of squirrel cage Induction Motors can be achieved through variable pole stator winding in squirrel cage type Induction Motor. The number of rotor poles is equal to the stator winding poles and hence change in speed of the Induction Motor is possible.

1.2.1.2. Variable voltage control

By varying the voltage, the speed can be controlled. The torque of IM is directly proportional to square of the voltage, so until the required torque is not achieved the voltage is varied. This method is used where the load torque reduces with speed for example in fan loads. This method can be used for speed control only for below synchronous speeds, because if the voltage is increased beyond the rated voltage then the insulation may breakdown.

1.2.1.3. Rotor resistance control

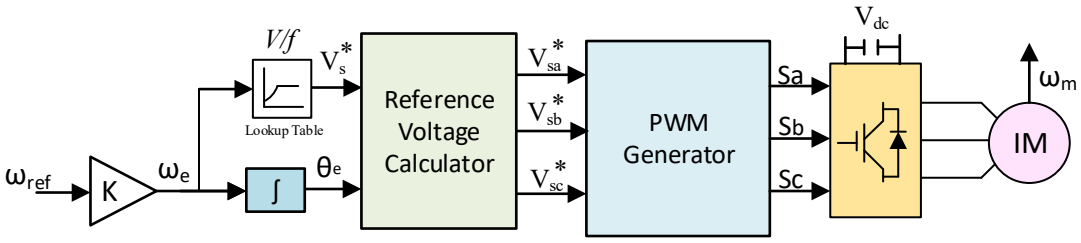
This method is used to control wound rotor type Induction Motor. Variable resistances can be connected through slip rings to the rotor circuit of Induction Motor. The maximum torque is independent of the rotor resistance, but the slip at which the maximum torque occurs depend upon the resistance. As the rotor resistance increases, slip increases and hence the slip at which maximum torque is changed. However due to the additional rotor resistance rotor current loss is increased and efficiency of motor reduces.

1.2.1.4. Variable frequency control

Variable Frequency Control is a technique used to control the speed of an Induction Motor. It is possible to control the speed of Induction Motor by varying the frequency. If the supply frequency is changed, induced EMF changes so as to maintain the constant air gap flux.

1.2.1.5. V/f speed control

The synchronous speed of Induction motor is directly proportiona to supply frequency and also induced emf in stator winding. So, if supply frequency is varied, the supply voltage magnitude is also changed accordingly to maintain the same air gap flux. This method is efficient, but has a limitation as for implementation of this control variable supply frequency is needed. Fig.2 here shows the block diagram of open and close loop control of V/f speed control method.



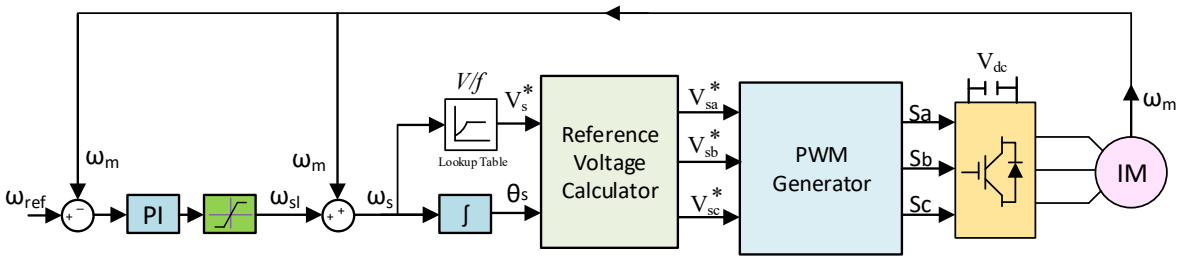


Fig.2. Block diagram of V/F speed control

1.3. VECTOR CONTROL OF INDUCTION MOTOR

The method vector control of Induction Motor was initially developed by Blaschke. In vector control two quadrature components of three phase currents are synthesized, one of which is responsible for flux component which in turn effect the speed and the second component is responsible for the torque [10,16].
The transformation methods which transform the three phase quantities into two phase quadrature quantities are:
1) Clarke Transform
2) Park Transform

1.3.1. Clarke Transformation

Clarke transformation is shown in Fig.3. It is used to transform the three phase quantities into stationary two-phase quantities which are orthogonal to each other. Clarke transformation is described by eq. (1.1).

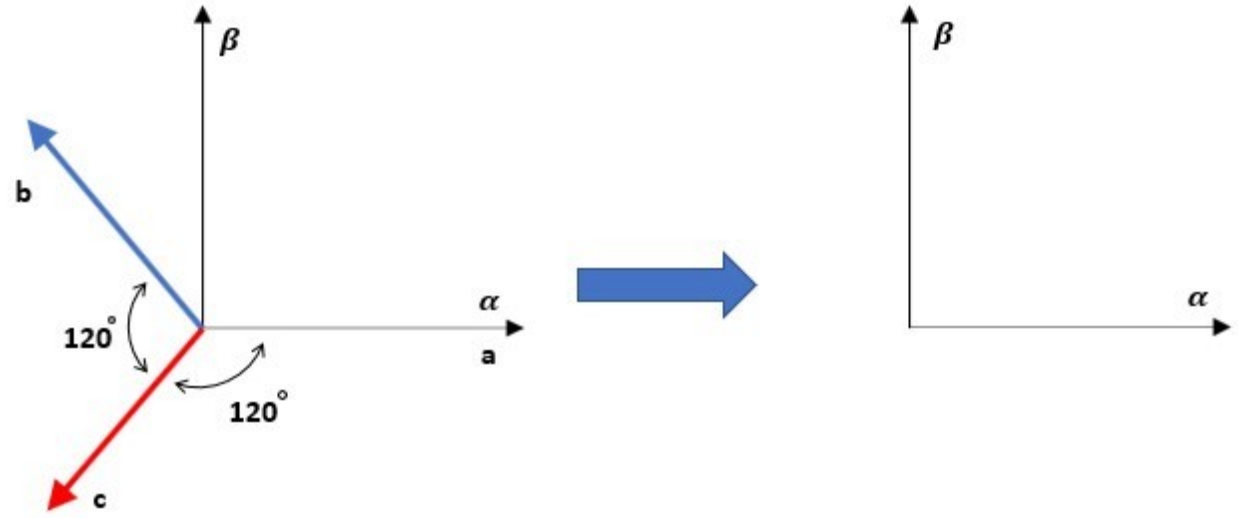


Fig.3. Clarke Transformation

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_\gamma \end{bmatrix} = \begin{bmatrix} 1 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} . \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \quad (1.1)$$

where $i_a\ i_b\ i_c$ are three phase quantities and $i_\alpha\ i_\beta\ i_\gamma$ are transformed quantities and i_γ is zero as vector sum of three phase quantities displaced equally and of same magnitude is zero, so $i_\alpha\ i_\beta$ are given by eq. (1.2) and (1.3).

$$i_\alpha = \frac{2}{3}(i_a - \frac{1}{2} i_b - \frac{1}{2} i_c) \quad (1.2)$$

$$i_\beta = (i_b - i_c) \quad (1.3)$$

Also, inverse Clarke transformation is used to transform the stationary two-phase orthogonal components to three phase balanced components which is given by eq. (1.4).

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} & 1 \\ \frac{-1}{2} & \frac{-\sqrt{3}}{2} & 1 \end{bmatrix} . \begin{bmatrix} i_\alpha \\ i_\beta \\ i_\gamma \end{bmatrix} \quad (1.4)$$

1.3.2. Park Transformation

By Clarke transformation the 3 phase quantities are changed to stationary two phase quantities. Park transformation is used to transform the stationary frame reference to rotating frame reference and it is described as eq. (1.5).

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} . \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \quad (1.5)$$

where i_{dq} is the rotating reference frame quantities, $i_{\alpha\beta}$ is the stationary reference frame quantities and θ is the rotor angle that has to be known for calculating the park transformation matrix. Fig.4 shows the geometrical interpretation of park transformation.

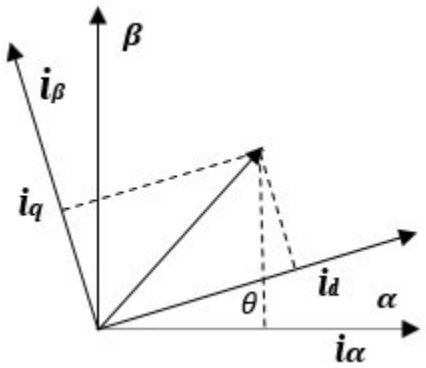


Fig.4. Park Transformation

The inverse park transformation is given by eq. (1.6)

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{1.6}$$

1.4. DIRECT TORQUE CONTROL SCHEME

In Direct Torque Control (DTC) the flux and torque are estimated. The flux is estimated by integrating back emf, torque is then estimated by cross product of flux and stator quadrature currents. These estimated values are then compared to the preset reference values and if the deviation is high then the switching of power electronic device is done such that these values are brought back to their references. The reference values of torque and flux are obtained by reference speed. Reference speed and measured speed are compared and then with a PI regulator reference torque is obtained. Also, the reference value of flux can be calculated through flux table. Fig.5 shows block diagram of Direct Torque Control scheme [5,11,27].

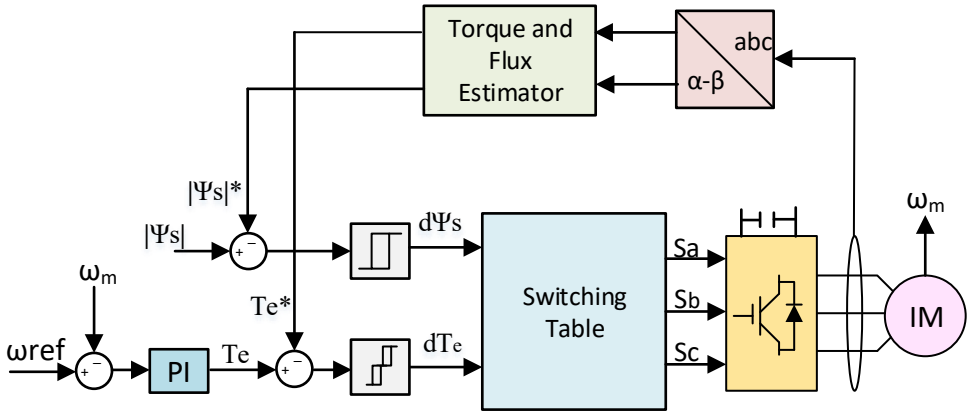


Fig.5. Block diagram for Direct Torque Control scheme

In Fig.5 shows the basic DTC where Sa, Sb and Sc are the switching instants. The 3-phase Voltage Source Inverter (VSI) has three arms and six switches. The speed of Induction Motors is controlled through generation of different switching patterns. Three switches from different legs are ON and the remaining are OFF, and accordingly switching voltage vectors appears on the terminals of the 3 phase Induction Motor, which effects the performance of Induction Motor.

1.5. FIELD ORIENTED CONTROL SCHEME

The three phase Induction Motor is controlled as a DC motor in the field-oriented control. The 3-phase current and voltage are measured and then transformed into two phase voltage and current using Clarke and Park transformation. The reference value of flux producing component of stator current i_d is calculated by applying a PI control which is fed with the error in speed [8,11,27,5]. Another PI controller output gives the torque and from this value i_q is calculated by the relation given in eq. (1.7).

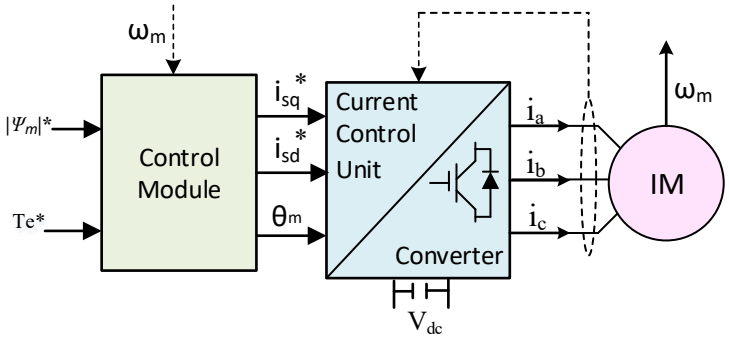
$$i_q = \frac{2}{3}\frac{2}{\mathrm{p}}\frac{L_r}{L_m}\frac{T_e}{\varphi} \tag{1.7}$$

In above eq. L_r is the rotor inductance, L_m is the mutual inductance, T_e is the electrical torque and φ represents the calculated flux.

From reference flux i_d is calculated which is given in eq. (1.8).

$$i_d = \frac{\varphi}{L_m} \tag{1.8}$$

Now these two-phase values are converted into 3 phase currents and then a hysteresis current controller is fed with the error between the reference and measured values of three phase currents. Output of hysteresis current controller provides the switching for inverter from which the Induction Motor is controlled. Fig. 6 shows the block diagram of Field Oriented Control (FOC).



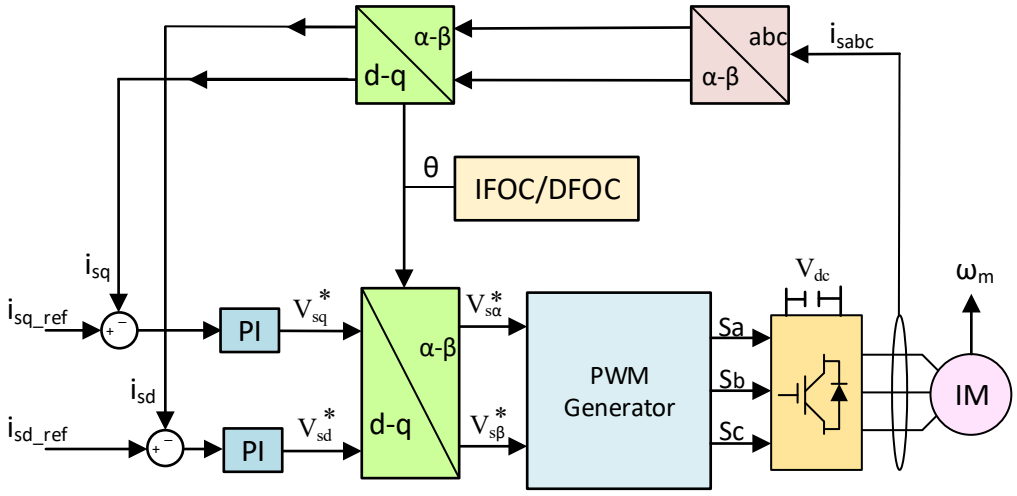


Fig.6. Block diagram for Direct Field Oriented Control

The Field Oriented Control (FOC) is further subdivided into two methods i.e. Direct FOC and Indirect FOC. Fig. 6 is showing the Direct FOC as the measured speed is directly sensed through the tacho-generator and then compared to the reference. Due to the need of the tacho-generator for Direct FOC, this approach becomes bulkier and costly. To remove tacho-generator, Indirect FOC is implemented in which the speed is also estimated from terminal voltage and currents and the only component which have to be measured from the motor terminals are only 3 phase voltages and currents.

1.6. ESTIMATION OF FLUX IN VECTOR CONTROL

In Vector Control (Direct Torque Control and Indirect Field Oriented Control), the estimation of flux and rotor angle θ are necessary for implementation of vector control drive on three phase Induction Motor. Estimation part of any controlling scheme is very important. In these two schemes also the estimation of rotor flux is done. The two methods for estimating the rotor flux are:

1) Current model-based estimation of rotor flux
2) Voltage model-based estimation of rotor flux

In current model the estimation is done by using the rotor parameters i.e. rotor resistance, rotor inductance, and mutual inductance along with the terminal voltages and currents, because of this the current model is not reliable because the rotor parameters change with the performing conditions of the Induction Motor. On the other hand, the voltage model estimation for flux is done by using only one parameter of Induction Motor i.e. stator resistance along with the terminal voltages and currents. The stator resistance remains almost constant during transient and steady state operation of Induction Motor with very wide range of speed.

In voltage model the flux is basically estimated by integrating the back emf of the motor. The back emf of the motor is calculated by eq. (1.9) and the estimated rotor flux can be obtained from eq. (1.10)

$$e = \mathrm{v} - i.R_s \qquad (1.9)$$

$$\varphi = \int e = \int (\mathrm{v} - i.R_s) \qquad (1.10)$$

where v and i represents terminal voltage and current respectively and Rs represents the stator resistance.

1.7. OBJECTIVE OF THE PRESENT WORK

The main focus of the present work is at:

1) To develop mathematical model of V/f control of three phase Induction Motor in MATLAB/Simulink and analyze the performance of the Induction Motor drive through off line simulation.
2) Real time simulation of V/f control in three phase Induction Motor through High Speed Reconfigurable Power Electronics Controller (HSRPEC) developed by C DAC.
3) To analyze the V/f control technique and compare them in both off line and online simulation.

1.8. OUTLINE OF THE THESIS

There are total six chapters in this dissertation work.

1) Chapter 1 presents a brief introduction on conventional speed control methods of Induction Motor and the concept of Field Oriented Control to obtain DC motor like control in Induction Motor.
2) Chapter 2 presents the mathematical modelling of Induction Motor. This is used to model the Induction Motor in MATLAB/Simulink.
3) Chapter 3 gives the complete overview of High Speed Reconfigurable Power Electronics Controller used for online simulation of V/f control of Induction Motor, the design environment used in the controller, the hardware design tools and the advantage of FPGA over conventional DSP controller
4) Chapter 4 discusses the V/f control technique used for the speed control of Induction Motor. It discusses about the relation between torque, speed and flux.
5) Chapter 5 discusses the modelling of Induction Motor using both offline and online along with their results and discussion. It discusses about the processor designing and code writing platform for HSRPEC. The MATLAB/Simulink models are presented and online simulation is also described along with the output.
6) Chapter 6 gives a brief conclusion of the work presented in the dissertation and the future scope of this dissertation.

CHAPTER 2

MATHEMATICAL MODELLING OF INDUCTION MOTOR DRIVES

2.0. INTRODUCTION

The dynamic simulation is one of the main steps in validating the design process of the motor drive system, eliminating the errors and designing mistakes in the construction and testing of prototype. The dynamic model of the Induction Motor in direct, quadrature, and zero-sequence axes can be obtained from fundamental equations of transformation. In the arbitrary reference frame, the dynamic analysis of the symmetrical induction machines has been intensively used as a conventional simulation startegy from which any specific mode of operation could be created. In modelling the induction machine using dq0 axis transformation MATLAB/Simulink has an edge over other machine simulators.

2.1. INDUCTION MOTOR MODELLING FOR VECTOR CONTROL

Induction Motor vector control is used for high performance Induction Motors. For Induction Motor, vector control is used to achieve the performance equivalent to DC motor. Two components of stator current one is accountable for torque and other is responsible for flux are determined through transformation of three phase current into two phase current.
The current which is proportional to torque, and the flux φ which is orthogonal to torque producing component are obtained [12,26,27].

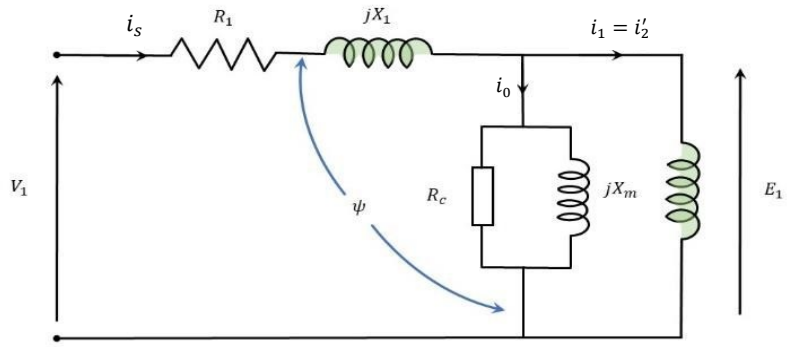


Fig.7. Diagram showing the direct field-oriented control

In Fig. 7, φ represents the total flux component and i_0 represents the rotor flux producing component and i_2 represents the torque producing component. Fig.8 represents the phasor diagram representing the total flux component i_s, magnetizing current i_m, and torque producing component

i_r. The reference for phasor diagram in Fig. 8 is taken as φ_m, i_φ is the component in the direction of pure inductance and i_r is in the direction of resistance and inductance.

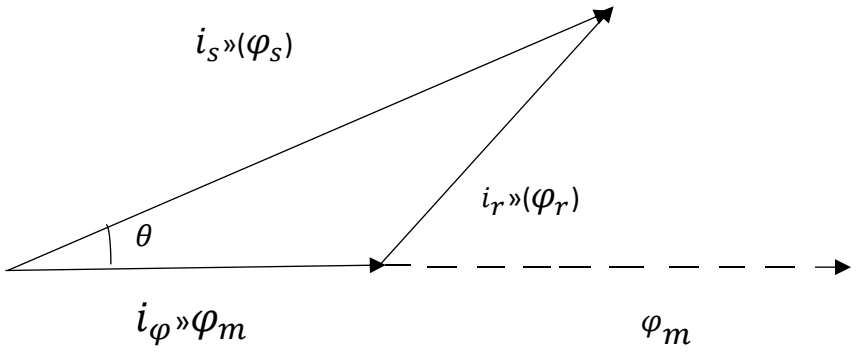


Fig.8. Phasor diagram for Induction Motor

The EMF induced is the n times the rate of change of flux, so voltage across R_r/s is given by eq. (2.1).

$$v_{\frac{R_r}{s}} = \frac{d\varphi_r}{dt} = j.\,\omega.\,\varphi_r.\cos\omega t \qquad (2.1)$$

The total component i_s has two components, and accordingly the dynamic model is developed. Now if the current 'i_s' is to be resolved along φ_r and perpendicular to it, the position of φ_r under all dynamic condition must be known. This is the basic concept of Vector Control of Induction Motor.

Now by Clarke transformation three phase quantities (abc) are changed to αβ component which is the stationary frame of reference given by eq. (2.2) and (2.3).

$$i_s = i_\alpha + j.\,i_\beta \qquad (2.2)$$

$$v_s = v_\alpha + j.\,v_\beta \qquad (2.3)$$

α-axis is always placed at the phase 'a' axis. Eq. (2.2) and (2.3) can be represented by (2.4) and (2.5). Now in steady state these phases rotate that is ϕ rotate and angles are fixed between voltage and currents and in dynamic model ϕ and magnitude both varies.

$$i_s = |i_s|.\,e^{j\omega t} \qquad (2.4)$$

$$v_s = |v_s|.\,e^{j\omega t} \qquad (2.5)$$

Fig.9 shows phasor for stationary frame of reference. The stator and rotor equations are given by eq. (2.6) and (2.7).

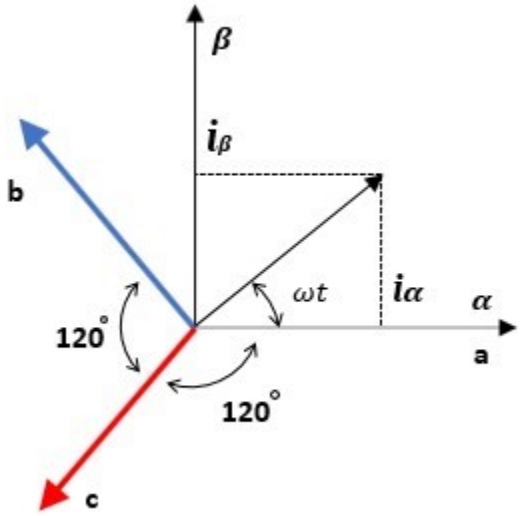


Fig.9. Phasor diagram for αβ frame of reference

For stator

$$v_s = R_s.\, i_s + L_{ss}.\frac{\mathrm{d}i_s}{\mathrm{dt}} + M.\frac{\mathrm{d}i_r}{\mathrm{dt}} \qquad (2.6)$$

For squirrel cage rotor (rotor is shorted in IM so net voltage sum across the mesh is equal to zero)

$$0 = R_r.\, i_r + L_{rr}.\frac{\mathrm{d}i_r}{\mathrm{dt}} + M.\frac{\mathrm{d}i_s}{\mathrm{dt}} \qquad (2.7)$$

To find instantaneous position of rotor flux for converting $\alpha\beta$ into dq reference frame i.e. synchronous frame of reference. So, if we know the instantaneous position of ϕ_r i.e. the rotor flux, then we can position dq along ϕ_r.

Fig. 10 shows the shifting of $\alpha\beta$ reference frame to dq reference frame, here θ shows the rotor angle and ϕ_1 and ϕ_2 shows angle with i_s and v_s with respect to stationary reference frame axis α.

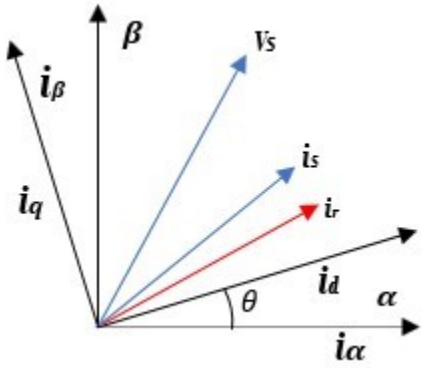


Fig.10. Phasor diagram for dq frame of reference

Now for transforming voltages and currents from stationary frame of reference to synchronous frame of reference eq. 2.8, 2.9, 2.10 and 2.11 are used [24, 30, 31].

$$v_{sd} = |v_s|.\cos(\phi 1 - \theta) \qquad (2.8)$$

$$v_{sq} = |v_s|.\sin(\phi 1 - \theta) \quad (2.9)$$
$$i_{sd} = |i_s|.\cos(\phi 2 - \theta) \quad (2.10)$$
$$i_{sq} = |i_s|.\sin(\phi 2 - \theta) \quad (2.11)$$

Now the final dynamic equations for stator and rotor with dq frame of reference are given in eq. 2.12, 2.13, 2.14 and 2.15.

For stator d axis voltage eq.

$$v_{sd} = R_s.\, i_{sd} + L_{ss}.\frac{\mathrm{d}i_{sd}}{\mathrm{dt}} - L_{ss}.\,\omega\theta.\, i_{sq} + M.\frac{\mathrm{d}i_{rd}}{\mathrm{dt}} - M.\,\omega\theta.\, i_{rq} \quad (2.12)$$

For stator q axis voltage eq.

$$v_{sq} = R_s.\, i_{sq} + L_{ss}.\frac{\mathrm{d}i_{sq}}{\mathrm{dt}} - L_{ss}.\,\omega\theta.\, i_{sd} + M.\frac{\mathrm{d}i_{rq}}{\mathrm{dt}} - M.\,\omega\theta.\, i_{rd} \quad (2.13)$$

For rotor d axis voltage eq.

$$0 = R_r.\, i_{rd} + L_{rr}.\frac{\mathrm{d}i_{rd}}{\mathrm{dt}} - L_{rr}.(\omega\theta - \omega r).\, i_{rq} + M.\frac{\mathrm{d}i_{sd}}{\mathrm{dt}} - M.(\omega\theta - \omega r).\, i_{sq} \quad (2.14)$$

For rotor q axis voltage eq.

$$0 = R_r.\, i_{rq} + L_{rr}.\frac{\mathrm{d}i_{rq}}{\mathrm{dt}} - L_{rr}.(\omega\theta - \omega r).\, i_{rd} + M.\frac{\mathrm{d}i_{sq}}{\mathrm{dt}} - M.(\omega\theta - \omega r).\, i_{sd} \quad (2.15)$$

In above equations

Stator inductance = L_{ss} = M+L_s, where M is mutual inductance and $L_s$is stator self-inductance

Rotor inductance = L_{rr} = M+L_r, where M is mutual inductance and L_r is rotor self-inductance

ω_θ = Angular frequency with which dq axis is rotating

ω_r = Angular frequency of rotor

v_{sd} and v_{sq} = Stator voltages correspond to d and q axis

i_{sd} and i_{sq} are stator currents correspond to d and q axis

i_{rd} and i_{rq} are rotor currents correspond to d and q axis

Rr and Rs are rotor and stator resistance respectively

Now based on the above equations the equivalent dynamic model is shown in Fig. 11 and Fig. 12.

d axis equivalent circuit model

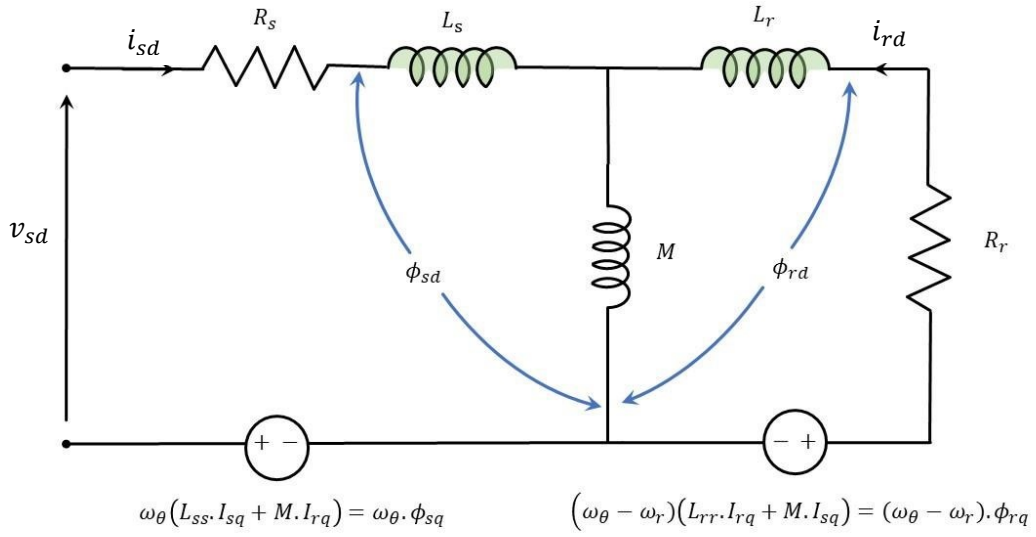


Fig.11. Equivalent dynamic model representing eq. 2.12

q axis equivalent circuit model

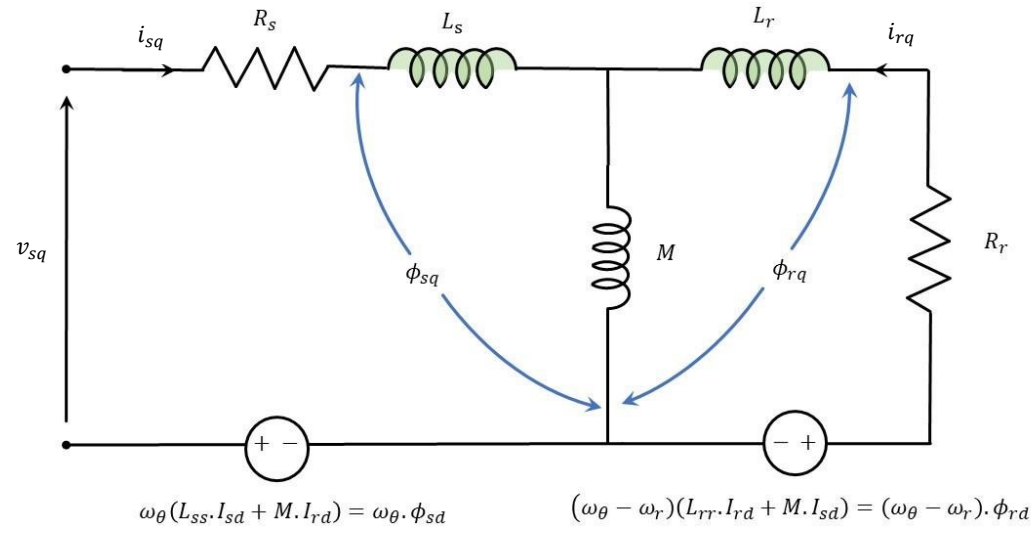


Fig.12. Equivalent dynamic model representing eq. 2.13

In Fig.11 and Fig. 12 ϕ_{sq} , ϕ_{rq}, ϕ_{sd}, ϕ_{rd} are given below:

$$\phi_{sq} = \omega_{\theta}(L_{ss}\, i_{sq} + M.\, i_{rq}) \quad (2.16)$$

$$\phi_{rq} = (\omega_{\theta} - \omega_r).(L_{rr}\, i_{rq} + M.\, i_{sq}) \quad (2.17)$$

$$\phi_{sd} = \omega_{\theta}\,(L_{ss}\, i_{sd} + M.\, i_{rd}) \quad (2.18)$$

$$\phi_{rd} = (\omega_{\theta} - \omega_r).(L_{rr}\, i_{rd} + M.\, i_{sd}\,) \quad (2.19)$$

Here dq axis is rotating with speed ω_{θ}, so when ω_{θ} is zero then it represents the stationary reference frame and when ω_{θ} is equal to synchronous speed then it represents the synchronous reference frame.

So, all the above equations can be represented in matrix form and given by eq. 2.20.

$$\begin{bmatrix} V_{ds} \\ V_{qs} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_{ss} + L_{ss}.D & -\omega_s.L_{ss} & M.D & -M.\omega_s \\ L_{ss}.\omega_s & R_{ss} + L_{ss}.D & M.\omega_s & M.D \\ M.D & -(\omega_s - \omega_r).M & R_r + L_{rr}.D & -L_{rr}.(\omega_s - \omega_r) \\ -M.(\omega_s - \omega_r) & M.D & L_{rr}.(\omega_s - \omega_r) & R_{rr} + L_{rr}.D \end{bmatrix} \cdot \begin{bmatrix} I_{ds} \\ I_{qs} \\ I_{dr} \\ I_{qr} \end{bmatrix} \quad (2.20)$$

2.2. CONCLUSION

The following chapter discusses the mathematical modelling of the Induction Motor. These equations help in dynamic modelling of Induction Motor in MATLAB/Simulink. Thus, the dynamic model developed is used to test the Induction Motor for various purposes, here it will be used for V/f speed control.

CHAPTER 3

HIGH SPEED RECONFIGURABLE POWER ELECTRONICS CONTROLLER

3.0. INTRODUCTION

Now a days, most of Induction Motors are driven by Voltage Source Inverter (VSI). These inverters consist of power semiconductor devices such as MOSFETs and IGBTs. PWM techniques are being developed to control VSIs for producing desired voltage and optimum converters harmonics. To meet the switching requirements of VSIs micro controllers, DSP and FPGA have been used for control of Induction Motor. At the core of every power electronics system lies a controller for meeting the system requirements. These controllers are often realized in off-the-shelf microcontrollers or DSPs which are stiff and limited in terms of resources and are at the risk of becoming obsolete. Using other on-board peripherals and PLDs, these constraints are conventionally. This method is lopsided by higher power consumption, component density and cost. ASICs were considered as the alternatives for general purpose controllers for long time. However, they proved to be a costlier substitute as any change in the system will require re-design and re-fabrication which is expensive as well as time consuming, taking into account the complexity of recent embedded design considerations. An alternative is provided in the form of a High Speed Re-Configurable Power Electronics Controller (HSRPEC) developed in an FPGA using SOPC methodology. Reconfigurable architectures are those that can be dynamically configured and reused for varying applications. Reusable components including CPU come in the form of Intellectual Property (IP) cores as soft, hard or firm cores. Soft cores generally offer maximum flexibility, enabling many core parameters to be altered prior to synthesis, while hard cores offer little or no flexibility [32]. For designing High Speed Re-Configurable Power Electronics Controller, some of the typical power electronics applications such as converters and inverters were analyzed so as to identify the requirements of control algorithms and modules like PWM, PI, UART, Phase Transformations etc. Since their inception in the mid-1980s, Field Programmable Gate Arrays (FPGAs) were a popular choice for reconfigurable prototyping and production of products in small to moderate quantities [32, 33].System design in FPGA chips is typically done using CAD (Computer Aided Design) software which supports tools for integrating IP cores of processors and peripherals including memory and also provide application software development environment. Modern CAD tools support design entry using several techniques. As the complexity of the circuit increases, the only practical choice is the Hardware Description Languages (HDLs). HDLs support circuit description using high-level language constructs. The CAD tool automatically handles the Low-level execution information, so that the designer can concentrate on the design functionality. The required modules are developed in HDL as generic IP cores. These cores are added on to the IP library of the CAD tool so that they can be placed as add-on peripherals to the generic CPU soft core IP which is readily available. The drivers for integration of these IPs to the CPU core and custom functions for using them in the application software are also developed. The System Architecture, Design Methodology, Implementation and Application Development are discussed in further sections.

3.1. SOPC BASED POWER ELECTRONICS CONTROLLER ARCHITECTURE

Conventional designs of Power Electronic control are based on microcontrollers and Digital Signal Processors (DSPs) [34, 35]. Microcontrollers market is so dynamic that for a designer, overheads are high to keep pace with changing technology. Traditional DSPs are well suited for intensive algorithmic applications, but in terms of clock rate efficiency and the sequential nature of their internal design are limited. Typically, three or four clock cycles are needed per arithmetic logic unit (ALU) operation which is again limited by wait states requirement of external memories. Also limited resources such as PWM channels and communication interface controllers offer other challenges. In the past several solutions have been suggested, including the use of multiple-core DSPs within a device or multiple DSPs on a board; however, such system often considerably increases the costs significantly and simply shift the problem to another arena.

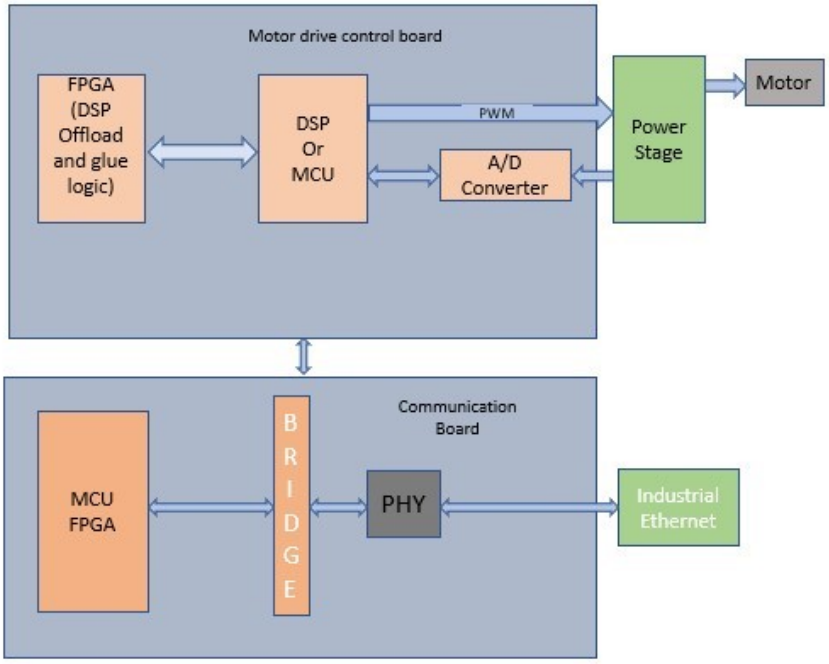


Fig.13. Block diagram of PE controller system implemented in Conventional Method

A solution to the growing issues of DSP implementations came with the introduction of FPGA technology enabling re-configurability. Unlike traditional DSPs, FPGAs have massively parallel structures containing a standardized set of arrays of configurable logic blocks (CLBs), memory, DSP slices and other components such as hardware multipliers. Besides soft processor cores, several other IP cores including memory controllers, interrupt controllers, Ethernet controllers, UARTs, timers, buses, and others are readily available for direct use in a highly optimized form within the FPGA. Designer can additionally create specific IP cores and place them along with in-built cores. All components of a computer system can be placed on a single FPGA using these modules and on-chip memory. This concept is known as a System on Programmable Chip (SOPC). The 'system builder' tools available with CAD software help in realizing this.

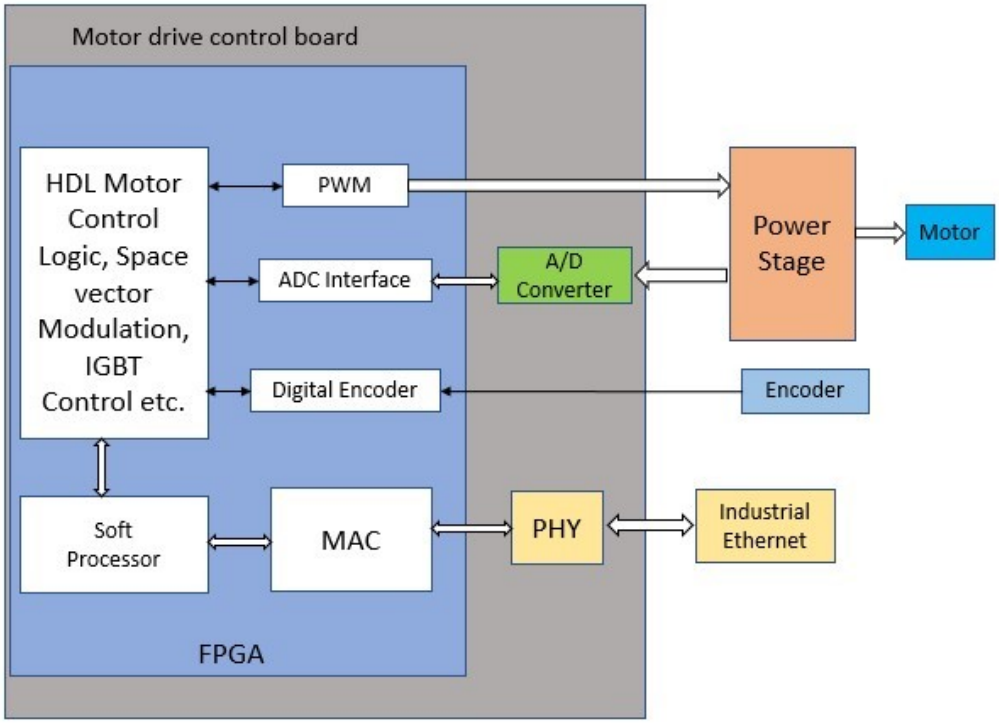


Fig.14. Block diagram of PE controller system implemented in SOPC Method

Any change in application will result mainly in changes in IP configuration inside FPGA whereas changes at board level hardware will be very limited. Unlike the DSP environment where the application software is highly linked to the hardware, the IP cores in FPGA environment is realized in HDLs, making it easy to port between different variants or even between FPGAs from different vendors. Thus, reliability and reusability of the design is more and a better time to market can be achieved [35, 36, 37].

3.2. HSRPEC USES AND ADVANTAGES

The "SOPC Based Power Electronics Controller" project is conceived with the idea of developing a reconfigurable controller architecture to replace the standard integrated micro controller/Digital Signal Processor based controller design, for Power Electronics system real-time control and monitoring. The basic controller building blocks for such a system will be incorporated in hardware and software into a programmable system like FPGA, which shall be appropriately reconfigured for specific application like drive control, power quality control, solar power conditioners, DC-DC converter systems etc.
In this system, the user can implement a controller for a specific application with minimal efforts than required in conventional processor hardware and its software development requirement. The control optimized functions specially developed for Power electronic (PE) Applications (custom made IPs) will help in faster controller development, reduce the processor obsolesce risk and satisfies minimum time to market criteria. The SOPC methodology created for PE applications discusses the concept of using control blocks as hardware IPs (designed in HDL as custom IPs),

which is faster and more reusable than its equivalent software implementation. These IPs can either be used as peripheral of a soft/ hard CPU core or as standalone IPs. Using the custom functions/instructions created, the IPs with soft/hard CPU cores can be invoked from the application software project, which can easily be understood by the PE system developer.

The FPGA offers various advantages such as:

1. Reconfigurable hardware for various PE applications
2. Reduced processor obsolescence risk
3. FPGA independent design
4. Faster performance
5. Faster concept to system design
6. Generic board design handles a variety of PE applications

Main uses and domain

Domain: SOPC based Hardware IP Module Development

Potential Application Areas
As a Digital Controller for
1. Solar MPPT Applications
2. High Frequency Inverter
3. High Performance AC Drive
4. DC Drive
5. UPS
6. DC-DC Converters
7. Multi-level Inverters
8. Interleaved DC-DC Converters
9. Cyclo Converters
10. Matrix Converters
11. Smart metering
12. Power Factor Correction (PFC)
13. Power Quality Applications
14. Controller with Custom/standard Communication Protocol

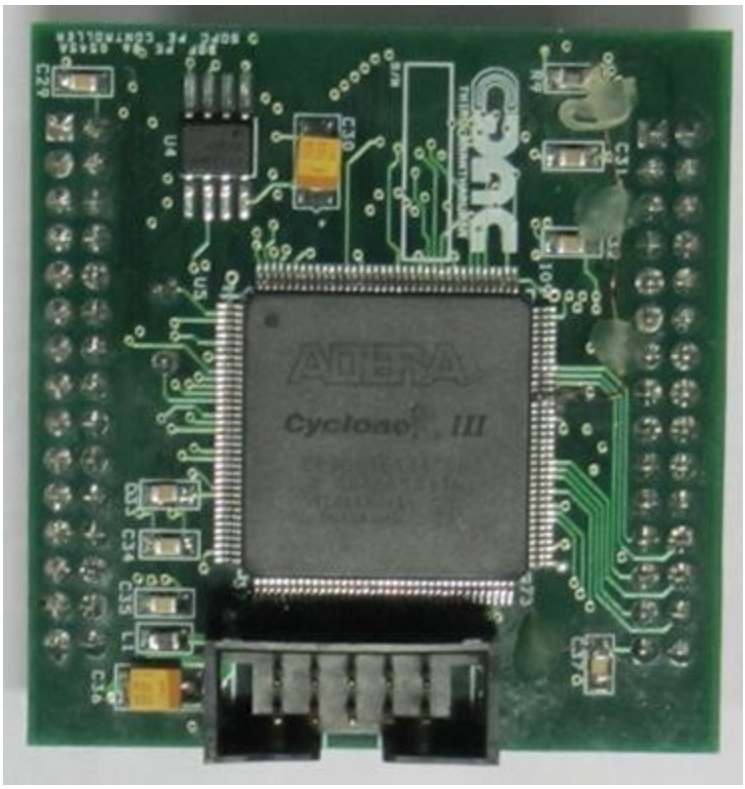


Fig.15. Controller Card

3.3. DESIGN METHODOLOGY

The methodology, as shown in the Fig. 16, is a two stage process. After analyzing the system requirements, the controller hardware is created with a soft processor IP core [39] at its center. The processor should have the important feature of integrating custom hardware or peripherals inside the processor and if necessary, to include custom instruction to support that hardware. The system requirements are primarily met by integrating off-the-shelf IP cores with the soft processor. IP cores that are not available in the system builder software is created in HDL and added on to the library for integration with the processor core inside an FPGA. Only the top-level entity of HDL code needs to be changed depending on the soft-core processor. The controller software is developed during the second stage in a software development environment for the soft processor. A hardware information file generated in stage 1 using the system builder tool provides the necessary information regarding the hardware to the software environment. Off-the-shelf hardware modules have built-in driver files for the software interface. For the custom developed IP cores, the onus is on designer to create these driver files.

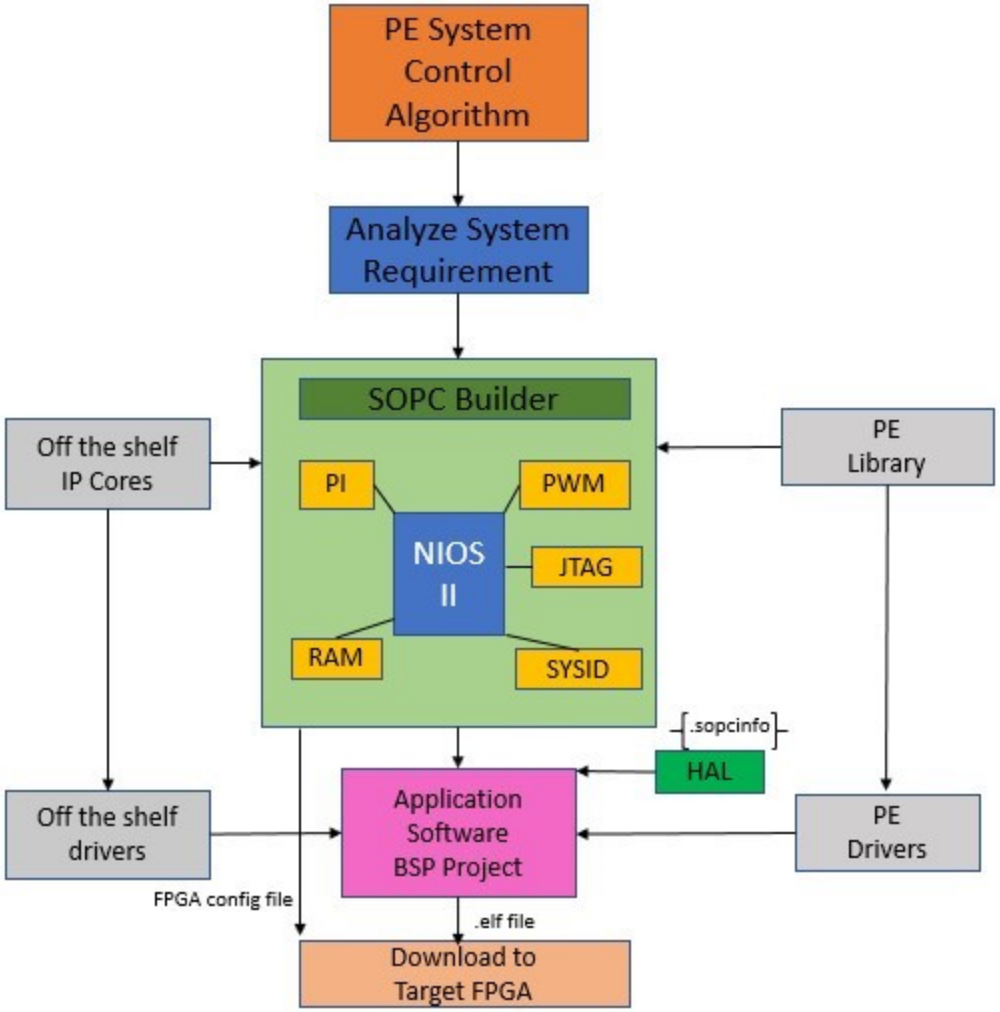


Fig.16. SOPC Design Methodology for Power Electronic Controller Development

3.4. DEVELOPMENT ENVIRONMENT

Development environment will be based on the following systems

1. Altera QUARTUS II IDE with QSYS (SOPC Builder) from ALTERA
2. NIOS II Embedded Suit for System Software Development
3. Model Sim for SOPC Simulation
4. PC with Windows /Linux

There are two major steps in the design flow

a. Hardware project –

1. For configuring the CPU core and PE specific IPs
2. As ready to use IPs designed in HDLs
3. PE Specific IPs illustrated here are designed by CDAC
4. Drag and drop approach

b. Software Project –

1. To design, debug, compile and down load the application software
2. 'C' or assembly-based environment
3. User can use custom instructions / functions to invoke the IPs

The Altera based SOPC design will be used in the current system as the IP cores especially the processor IP (NIOS II) by Altera is widely accepted in industry than Xilinx's Micro blaze processor IP. Moreover, the Altera SOPC design can be eventually implemented as an ASIC if required, for bulk application.

3.4.1. Hardware Design Tools

a. SOPC Builder/QSYS

SOPC /QSYS Builder is a powerful system development tool that works under the QUARTUS II FPGA project, which enables to define and generate a complete System-On-a-Programmable-Chip (SOPC) in much less time than using traditional, manual integration techniques. SOPC Builder is a general-purpose tool for creating systems that may or may not contain a processor and may include a soft processor other than the Nios II processor. SOPC Builder automatically generates the interconnect logic to integrate the components in the hardware system, once the SOPC Builder is ready with all IPs required for the application.

b. Architecture of SOPC Builder Systems

The building blocks for creating a SOPC builder system the SOPC builder modules are required. The SOPC Builder modules use Avalon interfaces for the physical connection of components, such as memory-mapped, streaming and IRQ. Integrate custom logic inside or outside the SOPC Builder system. Custom logic outside of the SOPC Builder system is connected to the SOPC Builder system through a Parallel Input Output (PIO) interface.
SOPC Builder ready-to-use components, includes:

1. Microprocessors, such as the NiosoII processor
2. Microcontroller peripherals, such as a Scatter-Gather DMA Controller, on chip memory etc.
3. Serial communication interfaces, such as a UART and a Serial Peripheral Interface (SPI).

4. General purpose I/O
5. Communications peripherals, such as a 10/100/1000 Ethernet MAC, PCI Express
6. Interfaces to off-chip devices
7. Power electronic IPs designed by CDAC can be imported to the builder easily

The primary outputs of SOPC Builder are the following file types:

1. SOPC Builder Design File (.sopc)—contains the hardware contents of the SOPC Builder system.
2. SOPC Information File (.sopcinfo)—contains a human-readable description of the contents of the .sopc file. The Nios II EDS uses the. sopcinfo file to compile software for the target hardware.
3. Hardware description language (HDL) files—are the hardware design files that describe the SOPC Builder system. The Quartus II software uses the HDL files to compile the overall FPGA design into an SRAM Object File (.sof) or Programmer Object File (.pof).

3.5. CONCLUSION

The following chapter discusses in detail about the advantages of FPGA over conventional DSP along with the specification of HSRPEC.
The design methodology tells about the various steps involved in the creation of a model in FPGA (HSRPEC) for various power electronics application.
The development environment discusses about the hardware and software tools required for the design methodology.

CHAPTER 4

V/f CONTROL OF INDUCTION MOTOR

4.0. INTRODUCTION

The Induction Motor drive system controlled by V/f is widely used in various industrial fields. However, its applicability is restricted, due to limited speed variation. The V/f control provides large torque in low speed range but in high speed range the torque is compromised. In view of this fact the coordination of V/f and vector control is also explored for wide range speed control of Induction Motor

The Induction Motor speed is governed in the V/f control by changing the stator voltage and frequency magnitude in such a manner that the airgap flux always stays at the required steady-state value. Sometime this system is called scalar control because it focuses exclusively on the steady-state dynamics.

4.1. CHARACTERISTICS OF INDUCTION MOTOR IN CONVENTIONAL V/f CONTROLLED DRIVE

Fig. 17 shows the equivalent circuit of Induction Motor, when Rs and Rr are stator and rotor winding resistances. L_m is magnetizing inductance. i_1, i_m and i_2 are stator, magnetizing and rotor current respectively.

In this equivalent circuit the terminal voltage is given as

$$v_s = -j\omega_r L_m i_m \tag{4.1}$$

Where ω_r is electrical angular speed of rotor

If the Induction Motor is operating in linear magnetic region, L_m is constant. Then the equation can be written as:

$$i_m = \frac{v_s}{2\pi f L_m} \tag{4.2}$$

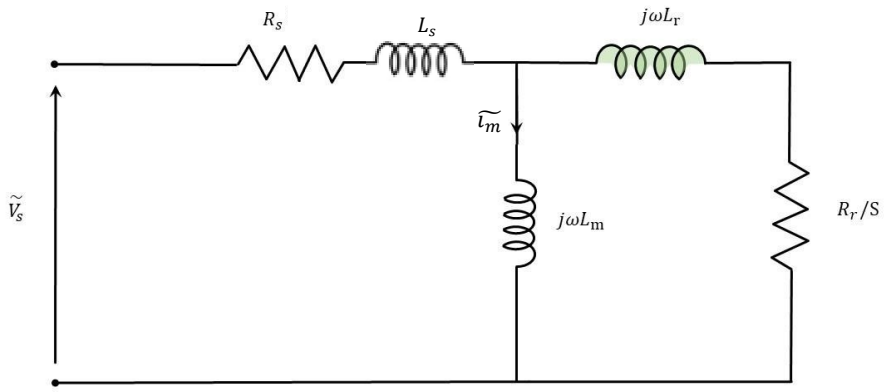


Fig. 17. Simplified steady state equivalent circuit of induction motor

From eq. (4.2), it follows that if the V/f proportion stays constant for any change in any given frequency then flux remains constant and the torque becomes independent of the supply frequency. In order to keep flux constant, the Vs/f ratio at the different speed would also be constant. As the speed increases, the stator voltages must, therefore, be proportionally increased in order to maintain the ratio of Vs/f constant. However, due to slip as a function of motor load, the frequency (or synchronous speed) is not the actual speed. The slip is very small at no load torque and the speed is almost equal to the synchronous speed [40, 41]. Thus, with the presence of the load torque, the simple open-loop Vs/f system cannot accurately regulate the speed. With the speed measurement, the slip compensation can be simply added in the system.

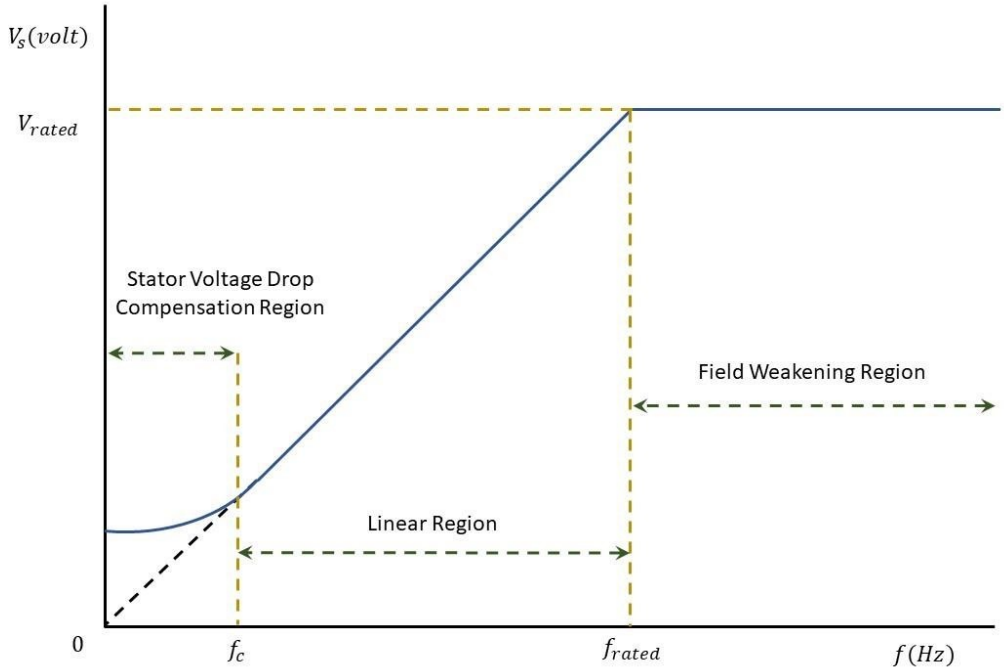


Fig.18. Stator voltage versus Frequency profile under V/f control

The ratio of stator voltage to frequency is usually based on the rated values of these variables. Fig.18 shows the typical V/f profile. Basically, in the V/f profile there are three speed ranges as follows:

1. A voltage is required at 0-fc Hz, so the voltage drop across the stator resistance cannot be ignored and must be compensated for by increasing the Vs. So, the V/f profile is not linear. From the steady- state equivalent circuit with Rs not equal to 0, the cutoff frequency (fc) and the appropriate stator voltages may be analytically calculated.
2. At fc-frated Hz, it follows the constant V/f relationship. The slope actually represents the air gap flux quantity as seen in eq. (4.2).
3. The constant Vs/f ratio cannot be satisfied at higher frated Hz, because the stator voltages would be limited at the rated value in order to prevent insulation break down at stator windings. The resulting air gap flux would therefore be decreased, and this will inevitably result in decreasing the developed torque correspondingly. Usually this region is called a "field weakening region". In order to prevent this, the constant V/f principle is also violated at such frequencies.

In the speed torque characteristics of Induction Motor the maximum torque is independent of the rotor resistance. However, the slip at which the maximum torque occurs varies with rotor resistance. When the rotor resistance is increased, so is the slip for maximum torque, and the stable operating slip range of the motor increases. Typical characteristics of an Induction Motor for different values of rotor resistances are shown in Fig. 19. The figure illustrates that the starting torque can be increased by increasing the rotor resistance. The maximum torque occurs at starting if the rotor resistance is increased to a value.

If the stator resistance is neglected the rotor resistance needs to be increased to a value equal to the rotor leakage resistance. If the rotor resistance is increased beyond this value the starting torque decreases. The breakdown torque occurs at slips greater than one (in the braking

region). The starting current decreases and the starting power factor is better at increased values of rotor resistances. The full load slip changes, facilitating speed control in a limited range when the rotor resistance is varied. However, efficiency is impaired at high rotor resistances due to increased losses.

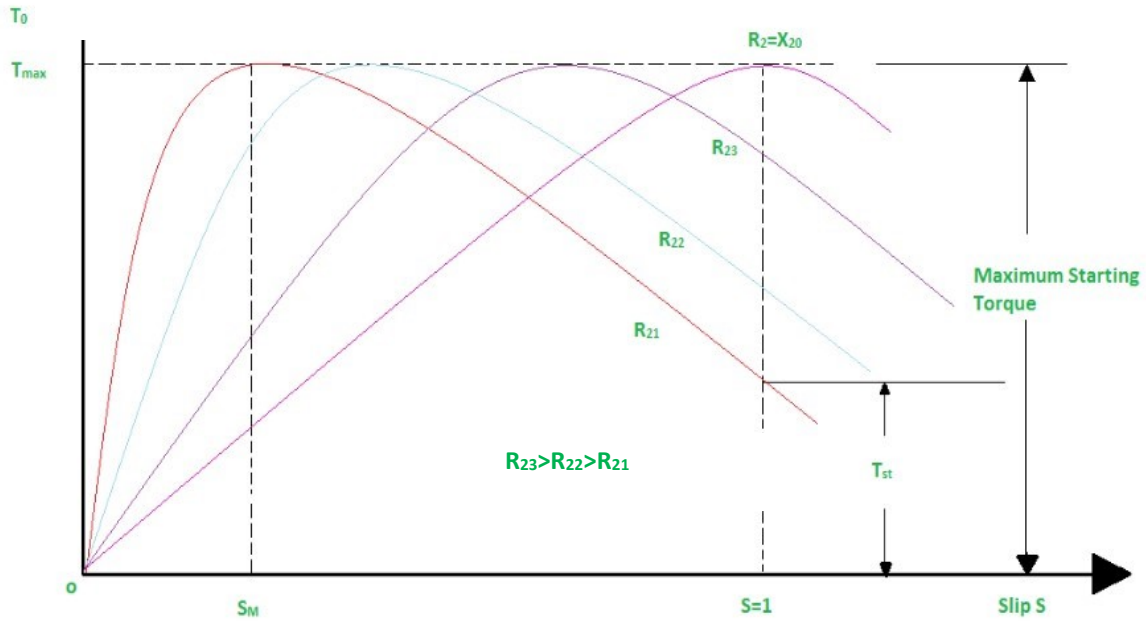


Fig. 19. Torque Versus Slip Speed of an Induction Motor

Based on the constant V/f principle, the speed of an AC induction motor can be controlled by both open loop and closed loop. Open-loop speed control is used when speed response precision is not a concern, for example in HVAC (heating, ventilation and air conditioning), fan or blower applications. In this case, the supply frequency is determined on the basis of the desired speed and the assumption that the motor will roughly follow its synchronous speed. The error in speed resulted from slip of the motor is considered acceptable.

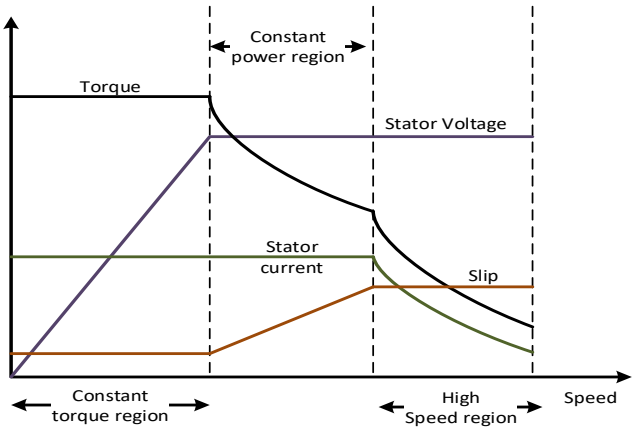


Fig. 20. Standard Torque Versus Slip Speed of an Induction Motor

4.2. CONCLUSION

This chapter discusses about:

1. The basics of V/f control technique.
2. Characteristics of Induction Motor in conventional V/f controlled drive and the speed torque characteristic.

Thus, the speed drops with increase in load torque. Also, due to significant voltage drop in stator resistance R_s in low speed range breakdown torque is reduced.

CHAPTER 5

SIMULATION AND EXPERIMENTAL IMPLEMENTATION OF V/F CONTROL OF THREE PHASE INDUCTION MOTOR USING HSRPEC

5.0. INTRODUCTION

The mathematical modelling of Induction Motor is developed in MATLAB/Simulink, and the online simulation is carried on using HSRPEC. These models are used to analyze the behavior of Induction Motor for V/f control in open as well as close loop operation. Open loop V/f Control is implemented by varying the supply frequency and terminal voltage such that the V/f ratio remains the same, the flux produced by the stator remained constant. As a result, the maximum torque of the motor remained constant across the speed range.
Closed-loop V/f Control uses a Proportional and Integral Controller to process the error between the actual rotor speed and reference speed and uses this to vary the supply frequency. The Voltage Source Inverter (VSI) varies the magnitude of the terminal voltage accordingly so that the V/f ratio remained the same so that the maximum torque remains constant across the speed range. Hence, the motor is fully utilized and successful speed control is achieved.

5.1. MODELLING OF SYSTEM USING HSRPEC

The Induction Motor velocity is governed by the adjustable magnitude of stator voltages and frequency in the V/f control so that the airgap flux is always constant at the required value at the stable state. In this implementation, the ADC channel is used to input the motor speed that is incorporated for angular positioning [29, 44, 40].

$$\theta = \int \omega_{ref}\, dt \tag{5.1}$$

θ based look-up tables for sin(θ) and cos(θ) for a cycle are evaluated. The values of sin(θ) and cos(θ) are obtained from the look-up tables on the basis of θ evaluated from speed, which correspond to alpha and beta. The phase voltage is calculated using the eq. (5.2).

$$v_{ph} = v_0 + k.\, \omega_{ref} \tag{5.2}$$

Where v_0 is stator voltage drop, k=V/ω, ω_{ref} is the electrical rotor speed. v_{ph} values are provided as input to alpha_beta _to_abc IP which generates PWM references corresponding to v_a, v_b and v_c voltages which is to be applied to the inverter bridge. The pulses of PWM vary the supply frequency and the voltage of the phase is varied to keep the flux constant. Fig. 21 shows the block schematic of V/f control of the 3-phase AC induction motor.

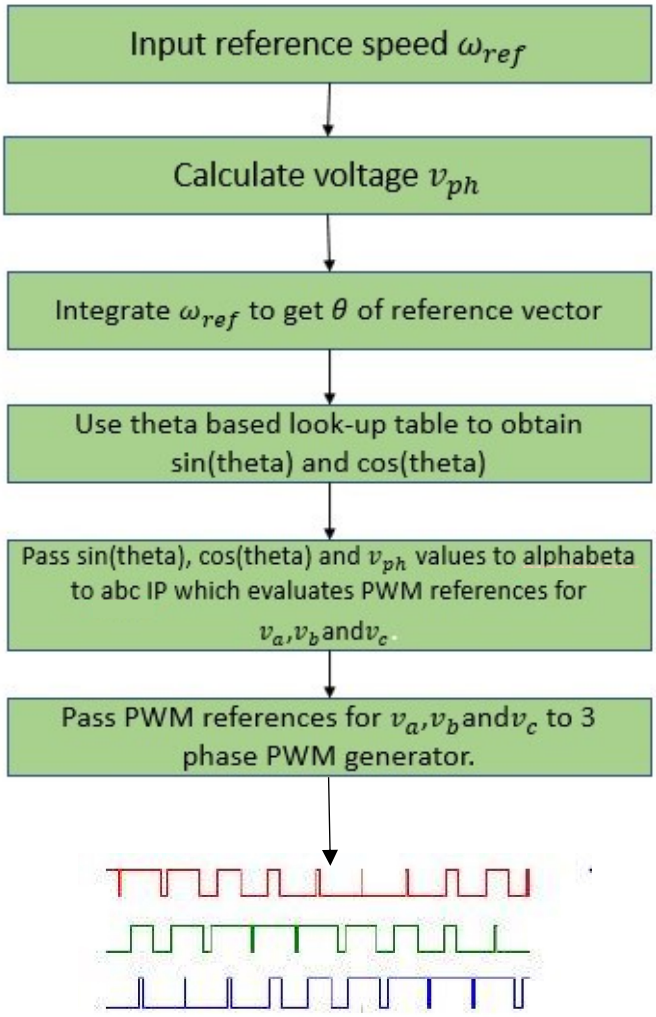


Fig.21. Flowchart of V/f control of 3-phase AC induction motor

The processor of HSRPEC is designed on Quartus platform and the model of the processor has been shown in Fig. 22.

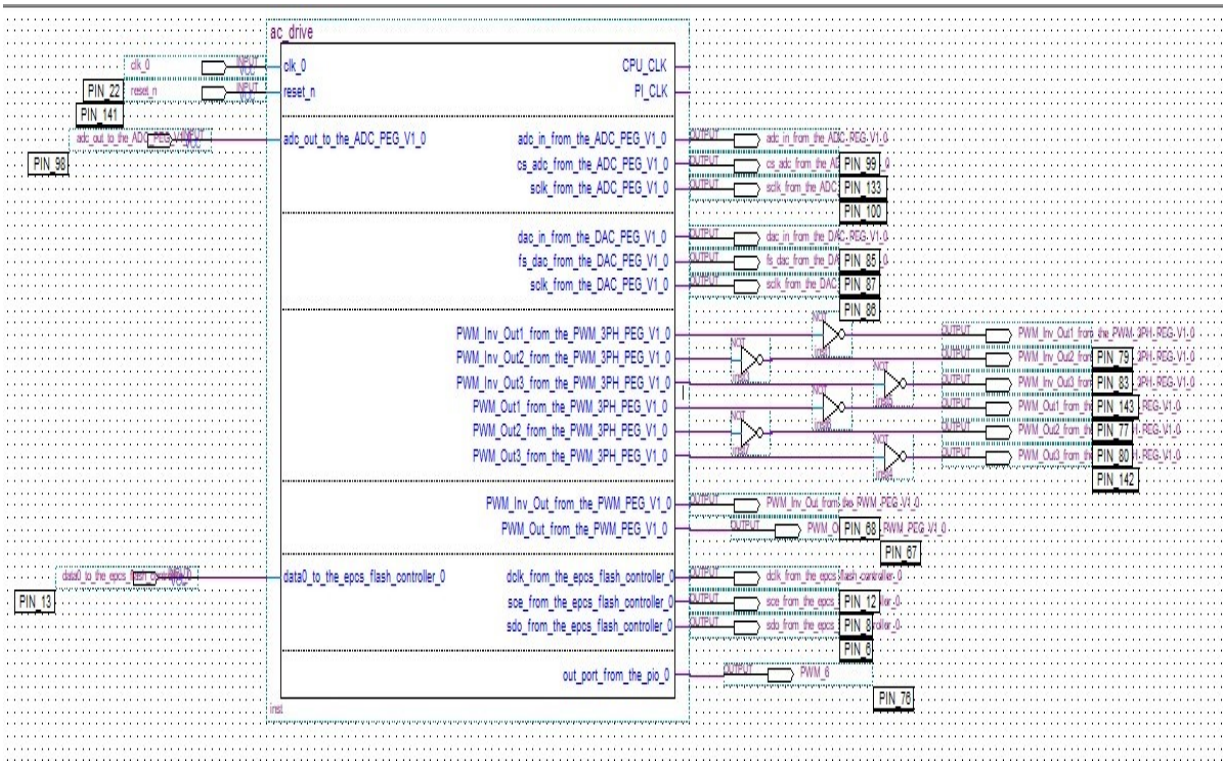


Fig.22. Power Electronic Processor designed for V/f control of 3-phase induction motor

The three phase PWM generator IP (PWM_PEG_V1) was used to generate the three phase PWM. This PWM allows the user to input the carrier wave frequency as well as modulating wave frequency, which enables the user to generate PWM the PWM of required frequency. The three phase PWM IP generates only three PWM so three not gates were used to get six pulses for the inverter leg.

The voltage feedback from the converter is digitized using Analog to Digital Converter (AD7328) in the SOPC-PEC peripheral interface card which is a serial ADC. The serial output from ADC is received by the Serial ADC Controller IP (ADC_PEG_V1) confirming SPI standard. The serial ADC controller IP is used to specify the pins to be used for the interfacing of analog data to the controller. Using this IP, the user can define which port to be used for the inputting of analog data to the controller, the controller then processes the data and converts it to the digital form to be used in the coding.

For debugging purpose, a serial DAC is provided in the SOPC-PEC peripheral interface card which can be configured using the Serial DAC Controller IP (DAC_PEG_V1).

The control loop consists of PI controller IP (PI_PEG_V1). The error signal is the input to the PI controller which in turn generates the reference signal for the PWM generator. The PWM pulses from PWM generator IP act as switching pulses for the IGBT.

The common flash interface controller core (CFI controller) connects SOPC Builder system to external flash memory to which the application program is loaded. The .jam file (FPGA Configuration File) is loaded in to the external serial flash memory using EPCS device controller core. The tri state bridge IP drives the CFI flash memory signals.
The JTAG Interface IP enables communication between SOPC builder system and JTAG host. The system ID core provides SOPC builder system with a unique identifier. Nios II processor systems use the system ID core to verify that an executable program was compiled targeting the actual hardware image configured in the target FPGA. If the expected ID in the executable does not match the system ID core in the FPGA, it is possible that the software will not execute correctly.

5.2. MODELLING OF SYSTEM USING MATLAB/SIMULINK

The PWM generation and speed control of induction motor is implemented using MATLAB/Simulink and has been shown in Fig. 23 [52]. A 3 HP, 415V, 50Hz, 1430 RPM Induction Motor is considered for the simulation study with the parameters given in Table 1.

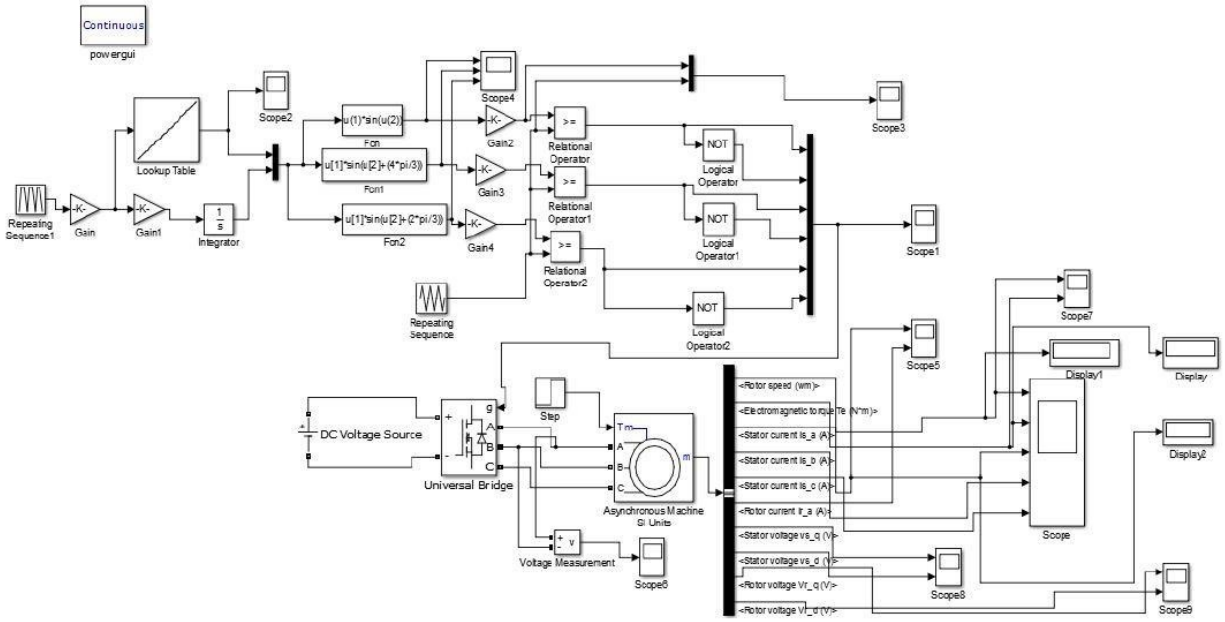


Fig.23. Open loop Simulation of V/f control of Induction Motor

Parameters	Values
Rs	6.347 ohm
Ls	0.0093242 henry
Rr	4.609 ohm
Lr	0.003412 henry
Lm	0.2144 henry
No. of poles	4

Table 1. Induction Motor parameters

Fig. 24 shows the PWM generation technique for the inverter.

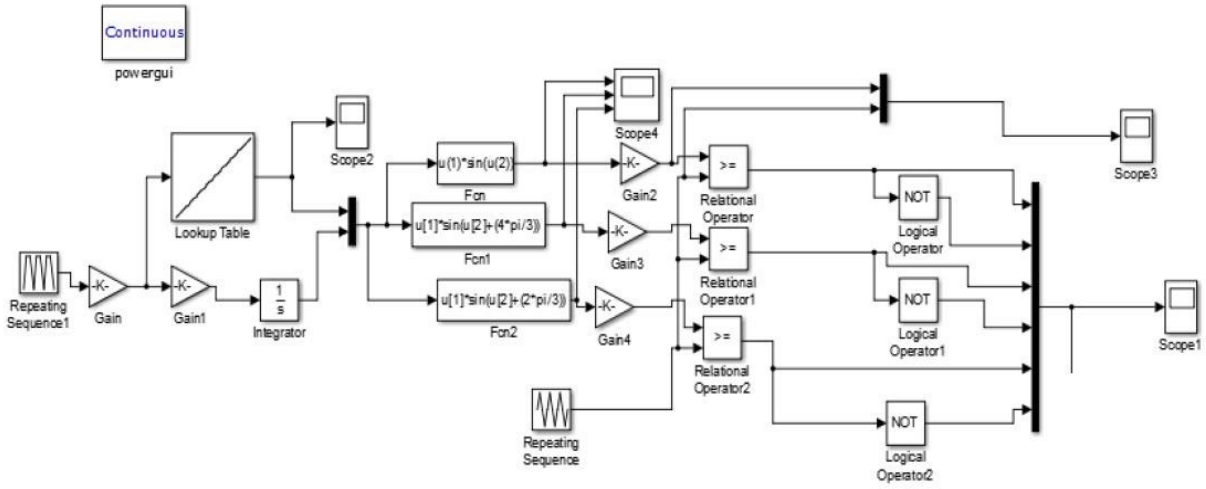


Fig.24. PWM generation technique for V/f control

The switches in the voltage source inverter can be turned on and off as required. If the top switch is turned on and off only once in each cycle, a square wave waveform result. However, if turned on several times in a cycle an improved harmonic profile may be achieved. Generation of the desired output voltage is achieved by comparing the desired reference waveform (modulating signal) with a high-frequency triangular 'carrier' wave. This technique is called Simple Pulse Width Modulation (SPWM) technique.

The angular position θ was obtained by integrating the reference speed. This θ was used to generate three phase output by passing through three function block which generates the modulating signal. This signal was then compared with a triangular waveform resulting in generation of three phase PWM. The triangular wave was generated using repeating sequence block. These three phase PWMs are passed through three NOT gate to generate six pulse PWM.

The six pulse PWM was applied to the gate of inverter, which results in switching of inverter. The three phase Voltage Source Inverter then generates three phase output which was applied to the Induction Motor. Output torque, angular speed, output voltage and output current are analyzed for varying frequencies using the scopes.

Frequency was changed thus; the voltage was also changed to keep the V/f ratio constant. This keeps the output torque nearly constant.

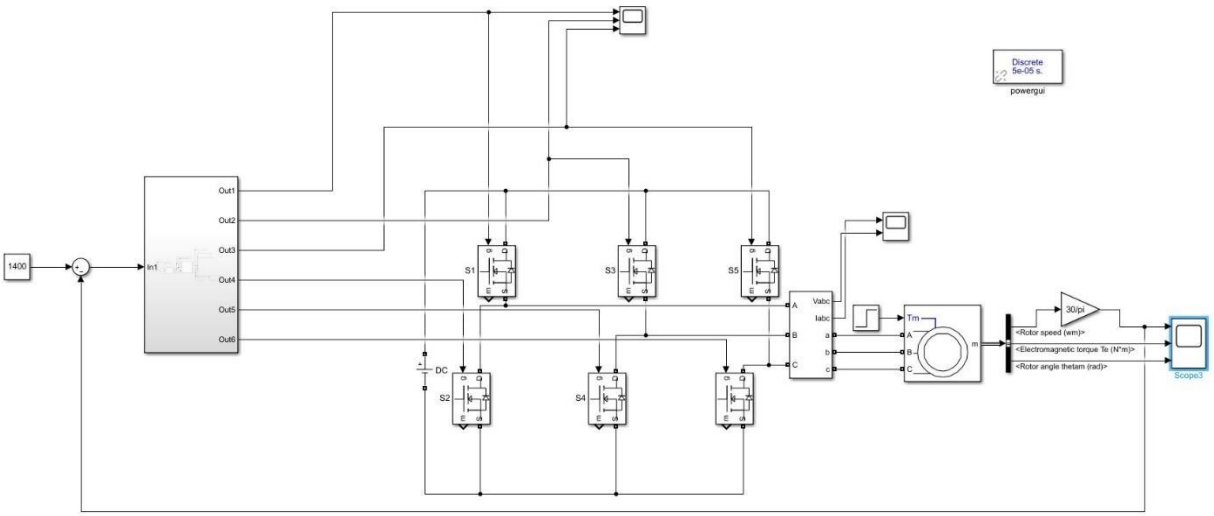


Fig.25. Close loop simulation of V/f control of Induction Motor

Fig. 25 shows the close loop V/f control. The PWM generation technique is same as used in the open loop simulation. Here, the angular speed was taken as feedback and compared with reference speed to calculate the error. This error was passed through PI controller which then controls the flux by keeping the V/f ratio constant. Thus, the output torque remains constant.

5.3. HARDWARE SETUP FOR V/F CONTROL OF INDUCTION MOTOR

The V/f control of three phase Induction Motor is implemented using HSRPEC and developed experimental setup was used to analyze the performance of drives. Fig. 26 shows a photograph of the experimental setup.

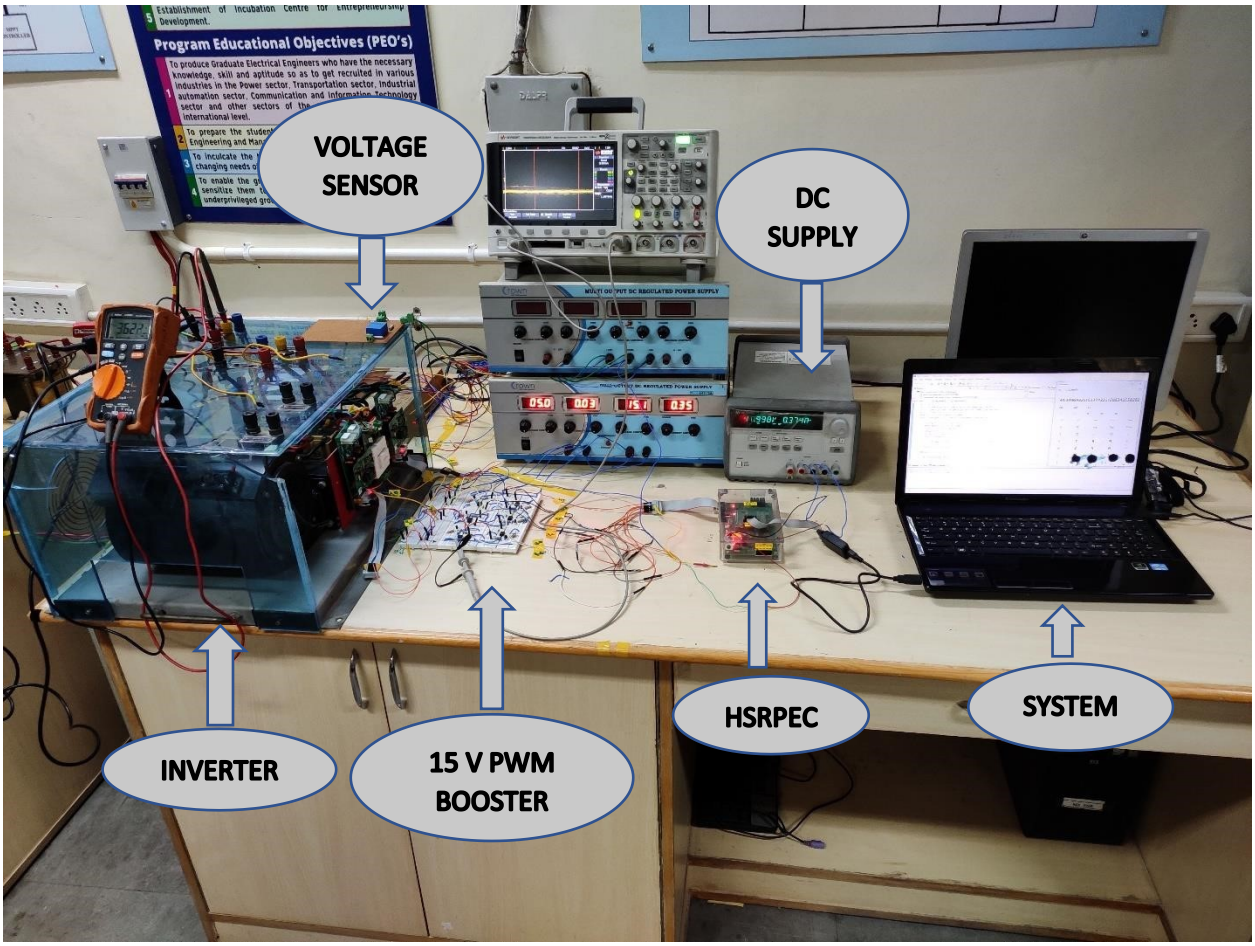


Fig.26. Setup for V/f control of induction motor

The V/f control algorithm is implemented in HSRPEC through necessary code developed in NIOS II platform. The required PWM control signal generated through HSRPEC is fed through a driver to the Voltage Source Inverter (VSI), which drive the Induction Motor. The Induction Motor is coupled with a DC generator for loading effect.

Fig.27. Complete setup for V/f control of induction motor

5.4. RESULTS AND DISCUSSION

The simulation study and experimental set-up for V/f control of three phase Induction Motor in open-loop and close-loop system is implemented using HSREPC controller.

5.4.1. SIMULATION RESULTS

5.4.1.1. OUTPUT OF OPEN LOOP SIMULATION

The simulation result for open loop V/f control of induction motor is shown in Fig. 28 and Fig. 29 for 50 Hz and 40 Hz respectively. The PWM signals generated are fed to the inverter as gate pulses which helps in switching action of inverter, thus the output from inverter is 3-phase output.

In open loop operation the speed is set at rated motor speed i.e. 1400 RPM. The angular speed (ω_e), torque (Te), output voltage (v_{abc}) and output current (i_{abc}) are plotted for two different frequencies of 50Hz and 40 Hz respectively.

The angular speed (ω_e) and torque (Te) are becoming constant after the time period of t = 0.5 seconds. Correspondingly the output current (i_{abc}) and output voltages (v_{abc}) are becoming stable after t = 0.5 seconds. The angular speed is becoming stable at around 1400 RPM and the torque is setting down to 3 Nm after 0.5 seconds.

At t = 1 second the load was applied; the output torque increased and the speed decreased to 1300 RPM correspondingly the current is also increased at t = 1 second. At t = 2 second another load applied increases the torque and decreases the speed further to 1280 RPM.

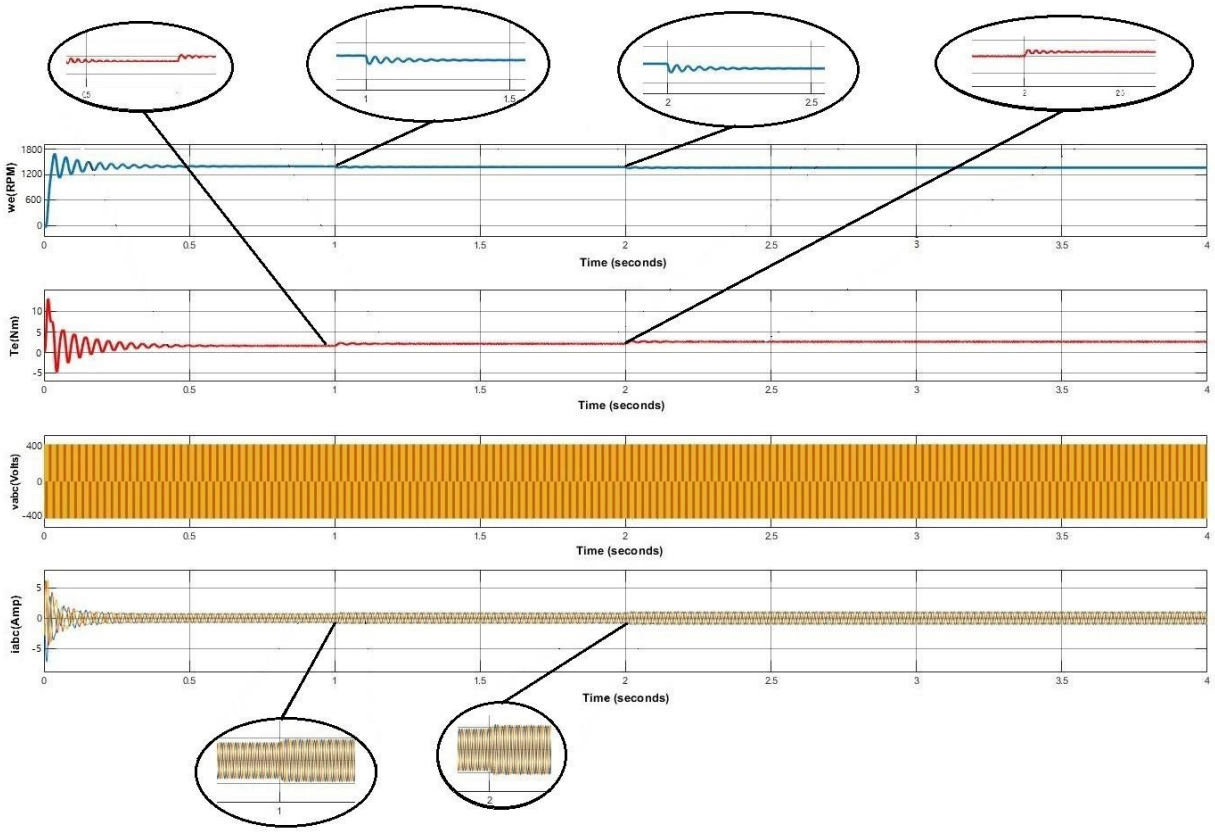


Fig.28. Angular speed (ω_e), torque (Te), output voltage (v_{abc}) and output current (i_{abc}) for open loop control at 50 Hz

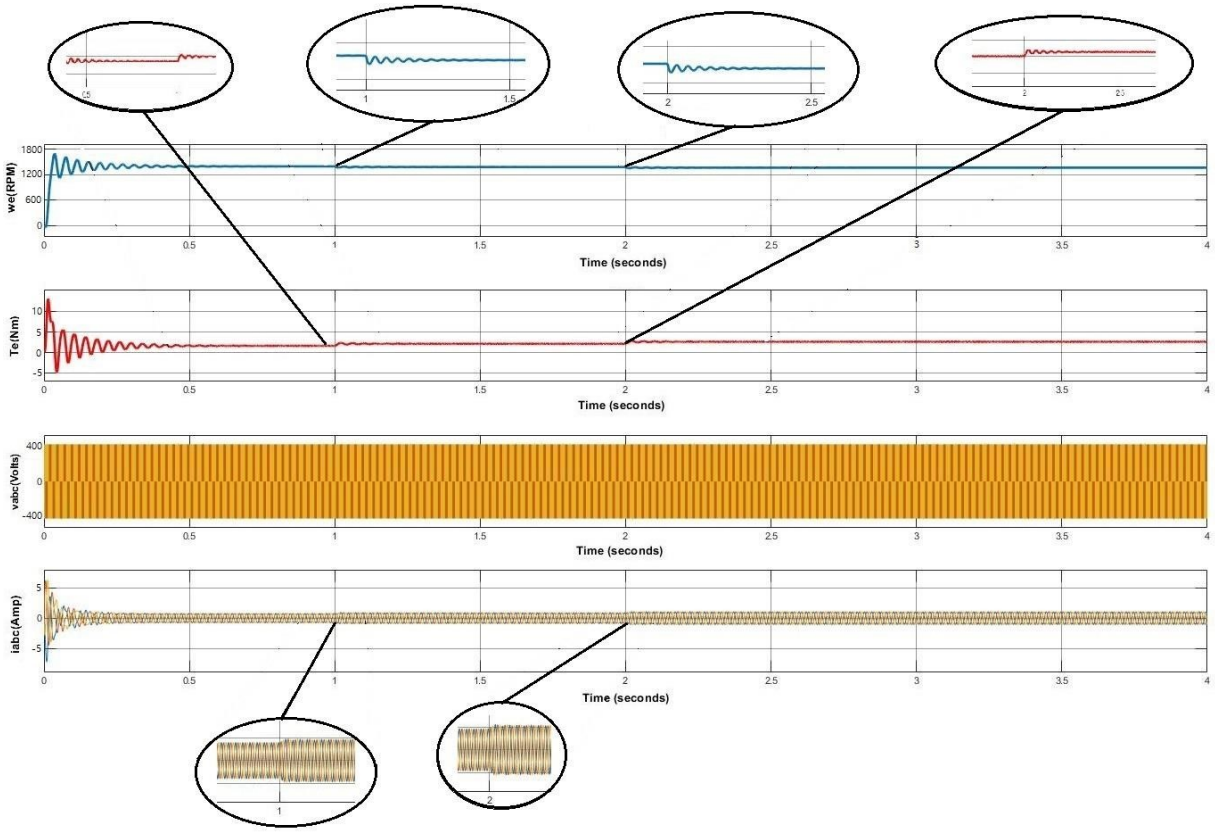


Fig.29. Angular speed (ω_e), torque (Te), output voltage (v_{abc}) and output current (i_{abc}) for open loop control at 40 Hz

The angular speed (ω_e) and torque (Te) are becoming constant after the time period of t = 0.5 seconds. Correspondingly the output current (i_{abc}) and output voltages (v_{abc}) are becoming stable after t = 0.5 seconds. The angular speed is becoming stable at around 1300 RPM and the torque is setting down to 3 Nm after 0.5 seconds.

Thus, the speed is decreasing at 40 Hz to 1300 RPM which was 1400 RPM at 50Hz but the torque is remaining nearly constant in both the case satisfying the V/f control criterion.

On application of load at t = 1 second and t = 2 second correspondingly the torque increased and the speed decreased to 1250 RPM.

5.4.1.2. OUTPUT OF CLOSE LOOP SIMULATION

The simulation result for close loop V/f control of induction motor is shown in below Fig. 30 and Fig. 31 for 50Hz and 40Hz respectively. The PWM are fed to the inverter as gate pulses which helps in switching action of inverter, thus the output from inverter is 3-phase output.

In close loop operation the speed is set at rated motor speed i.e. 1400 RPM. The rotor speed is fed back and the control operation tries to obtain the set speed by comparing with rotor speed.

The angular speed (ω_e) and torque (Te) are becoming constant after the time period of t = 1.5 seconds. Correspondingly the output current (i_{abc}) and output voltages (v_{abc}) are becoming stable after t = 1.5 seconds. The angular speed is becoming stable at around 1400 RPM and the torque is setting down to 3 Nm after 1.5 seconds.

At t = 1.5 second the load was applied; the output torque increased and the speed decreased to 1390 RPM correspondingly the current is also increased at t = 1.5 second. At t = 3 second another load applied increased the torque and decreased the speed further to 1380 RPM.

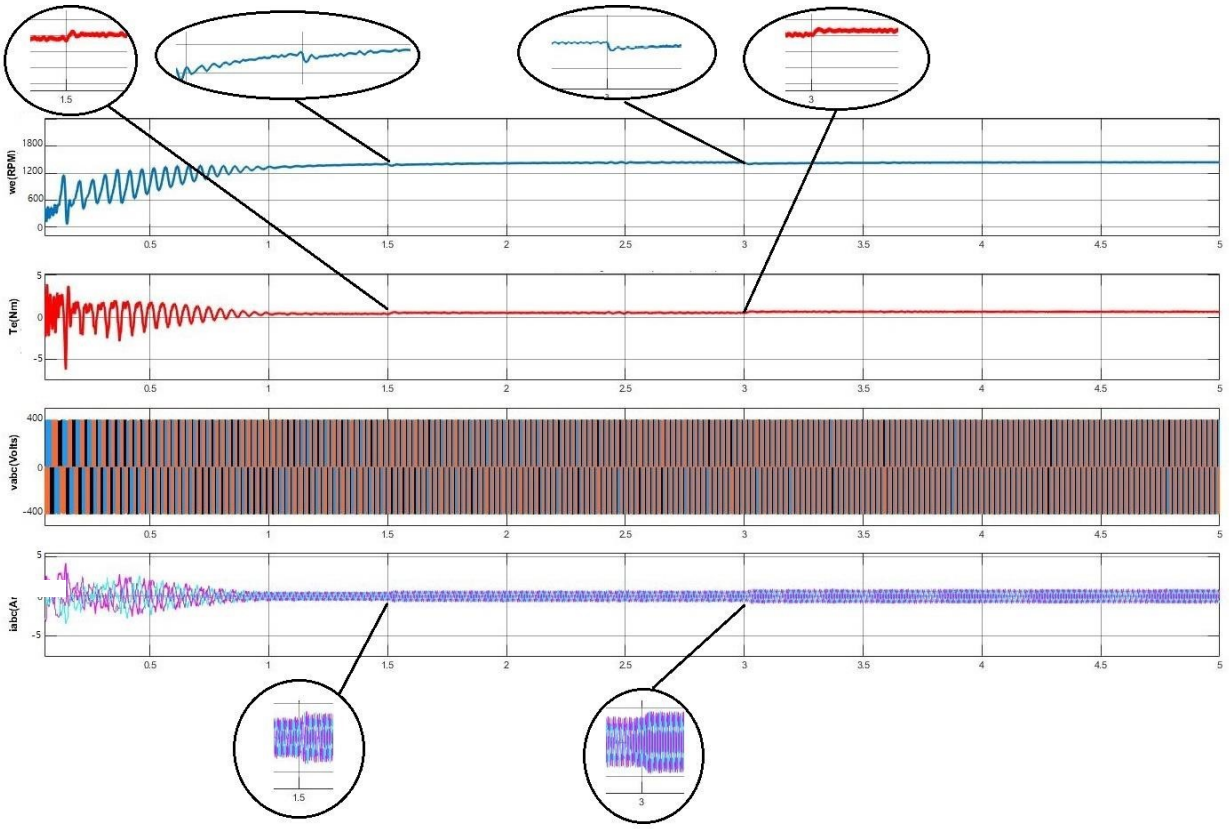


Fig.30. Angular speed (ω_e), torque (Te), output voltage (v_{abc}) and output current (i_{abc}) for close loop control at 50 Hz

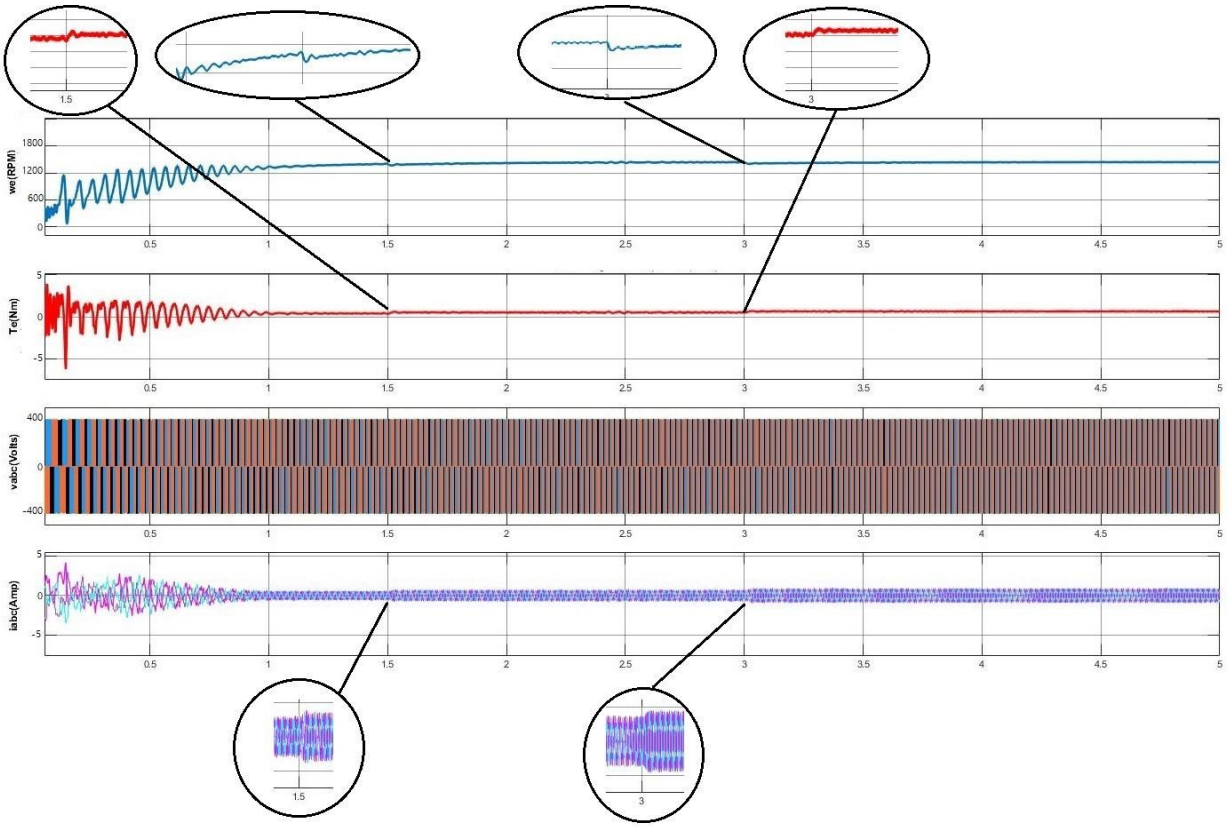


Fig.31. Angular speed (ω_e), torque (Te), output voltage (v_{abc}) and output current (i_{abc}) for close loop control at 40 Hz

The angular speed (ω_e) and torque (Te) are becoming constant after the time period of t = 1.5 seconds. Correspondingly the output current (i_{abc}) and output voltages (v_{abc}) are becoming stable after t = 1.5 seconds. The angular speed is becoming stable at around 1370 RPM and the torque is setting down to 3 Nm after 1.5 seconds.

On application of load at t = 1.5 second and t = 3 second correspondingly the torque increased and the speed decreased to 1340 RPM.

For the close loop operation, the output torque is remaining nearly constant for the two different values of frequencies, also the speed variation is also less as compared to open loop operation.

5.4.2. HARDWARE SETUP RESULTS

The relation between power and torque is given by:

$$P = \omega . Te \quad (5.1)$$

Where:

P is the output power

ω is the angular speed

Te is the output torque

The horse power and torque relation is given as:

$$H.P = (Te*RPM)/5252 \quad (5.2)$$

We also know that

$$1H.P = 746 \text{ watts} \quad (5.3)$$

Thus eq. 5.2 can be written as:

$$P = (Te *RPM*746)/ (5252) \quad (5.4)$$

The DC generator has an efficiency of 80%. Thus, the output power of DC generator is 80% of what the motor is producing. So, to calculate torque of the motor efficiency of 80% should be included.

5.4.2.1. Open Loop V/f Control at 50 Hz

S. no.	Output Power(W)	I (A.C)	I (D.C)	V (A.C)	V (D.C)	RPM	Output Torque (Nm)
1	No load	0.50	0.00	210	100	1410	No load
2	434.7	2.25	3.45	210	126	1320	2.6725
3	484	2.63	4.00	210	121	1288	3.3062
4	497.2	3.16	4.40	210	113	1230	3.5562

Table 2. Output for Open loop V/f control at 50 Hz

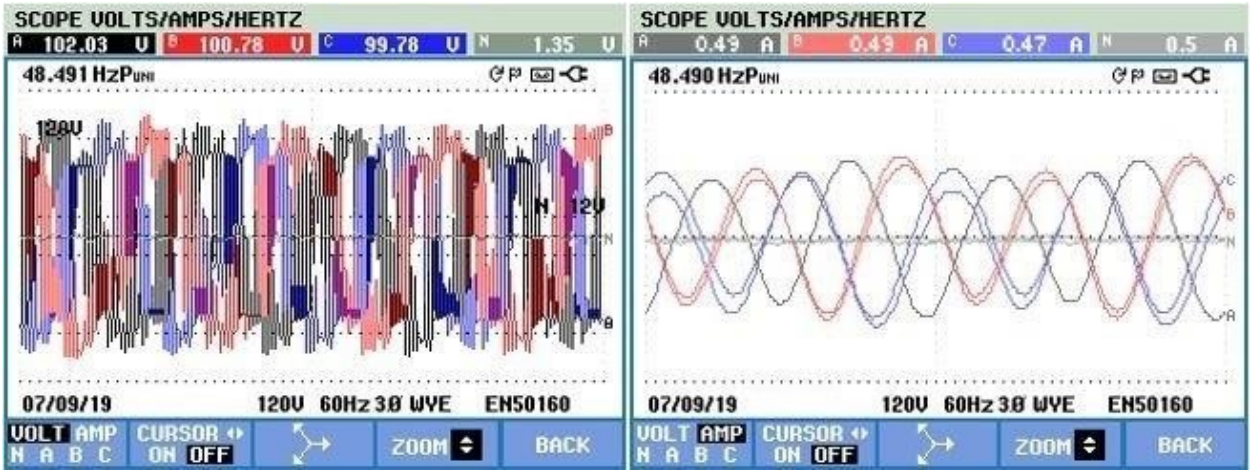


Fig.32. No load Output Voltage and Current at 50Hz

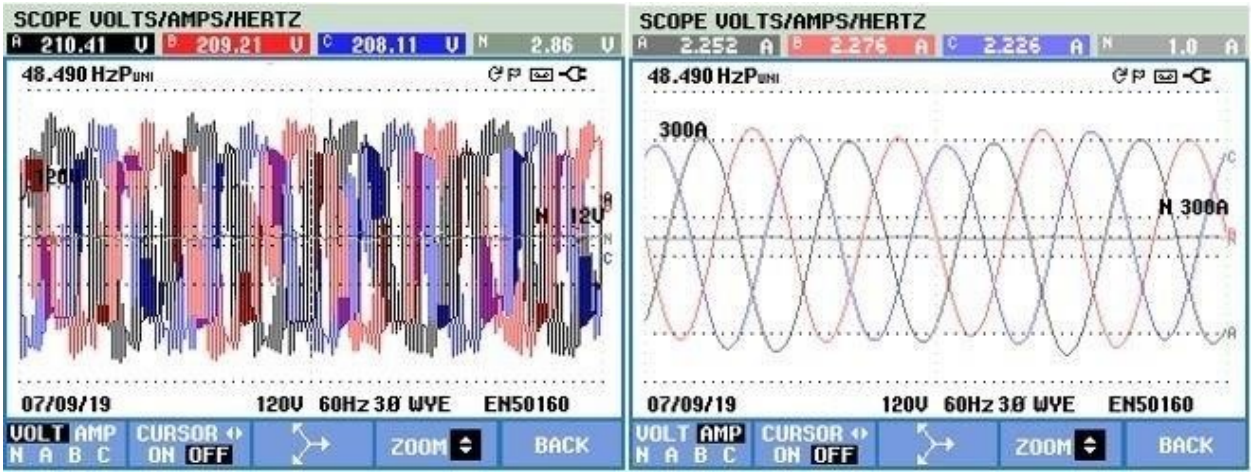


Fig.33. Output Voltage and Current at Load 1 at 50Hz

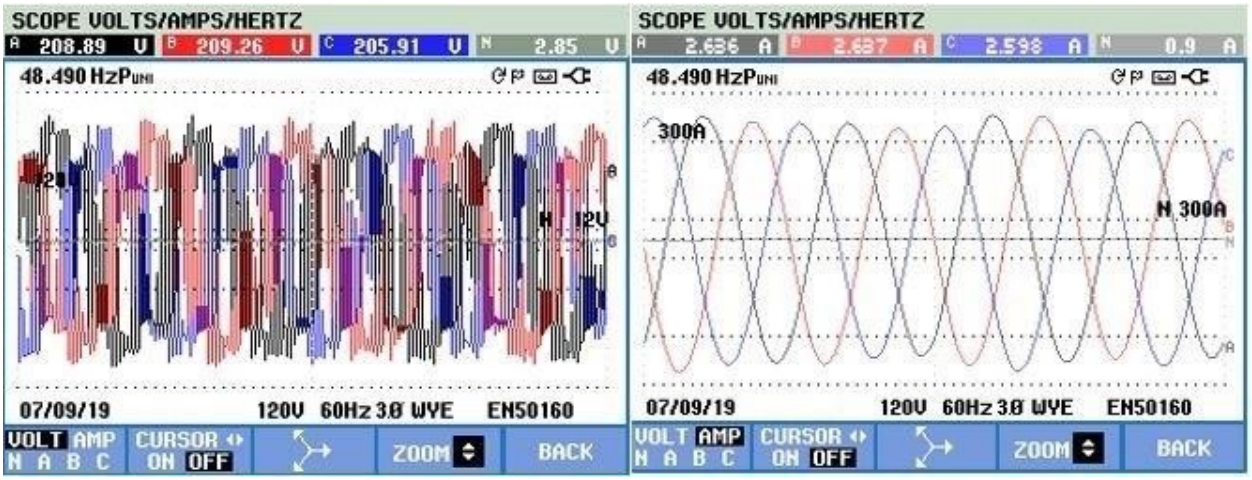


Fig.34. Output Voltage and Current at Load 2 at 50Hz

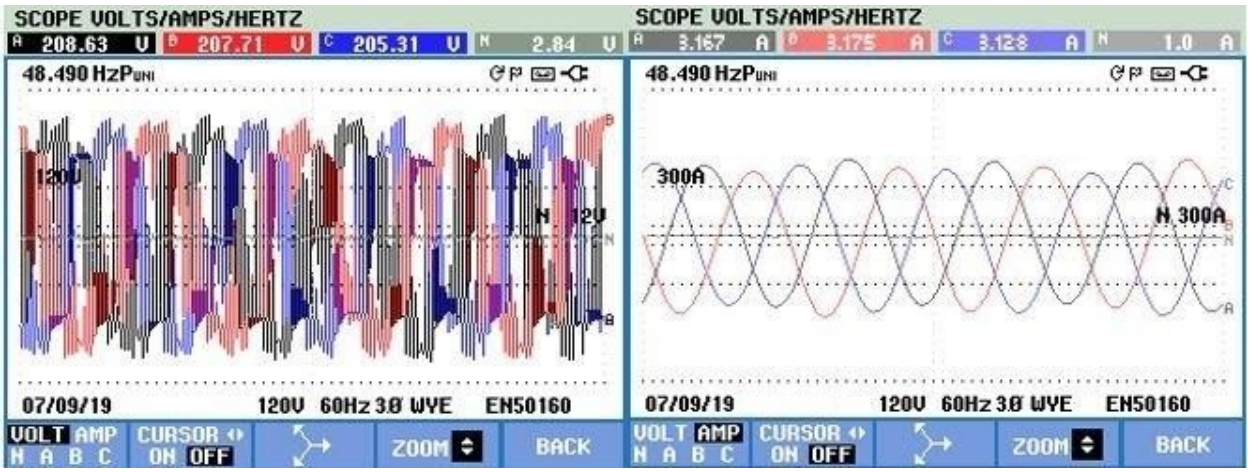


Fig.35. Output Voltage and Current at Load 3 at 50Hz

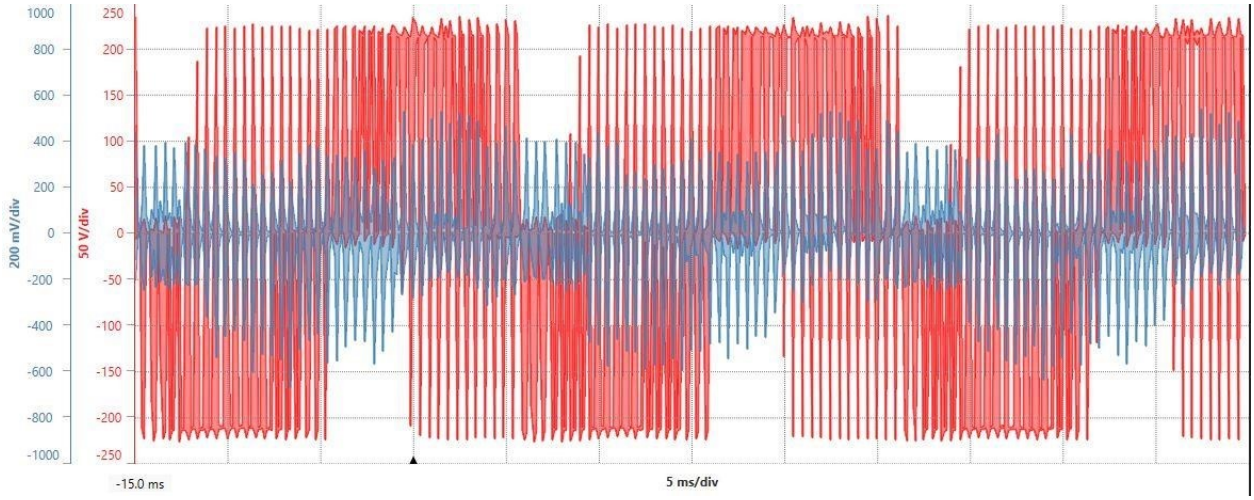


Fig.36. Single phase Output Voltage and Current Waveform at 50 Hz

5.4.2.2 Open Loop V/f Control at 40 Hz

S. no.	Output Power(W)	I (A.C)	I (D.C)	V (A.C)	V (D.C)	RPM	Output Torque (Nm)
1	No load	0.30	0.00	210	96	1350	No load
2	390.103	2.25	3.33	210	117	1290	2.6612
3	463.49	2.63	4.06	210	114	1256	3.2475
4	470.61	3.16	4.278	210	110	1200	3.4512

Table 3. Output for Open loop V/f control at 40 Hz

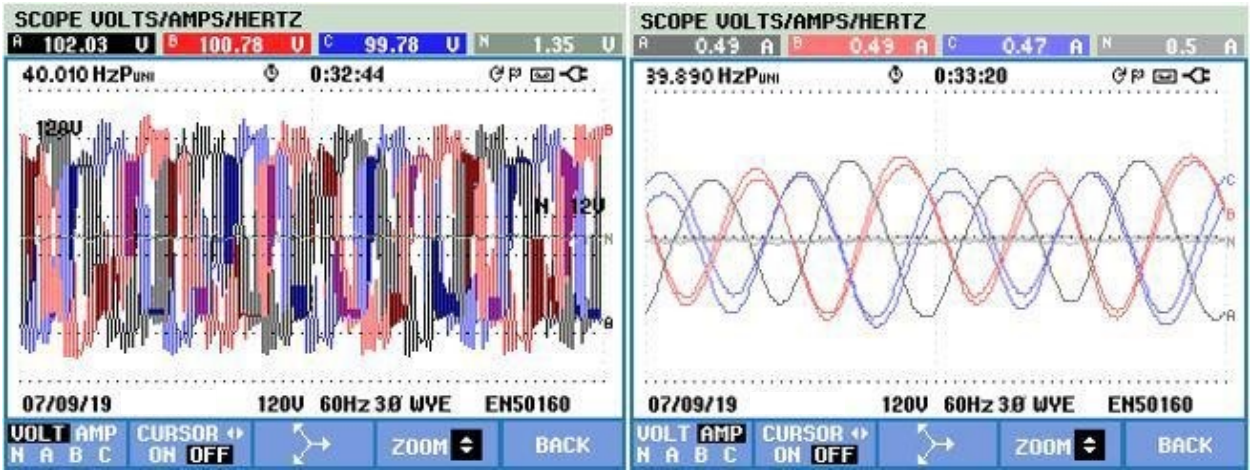


Fig.37. Output No Load Voltage and Current at 40Hz

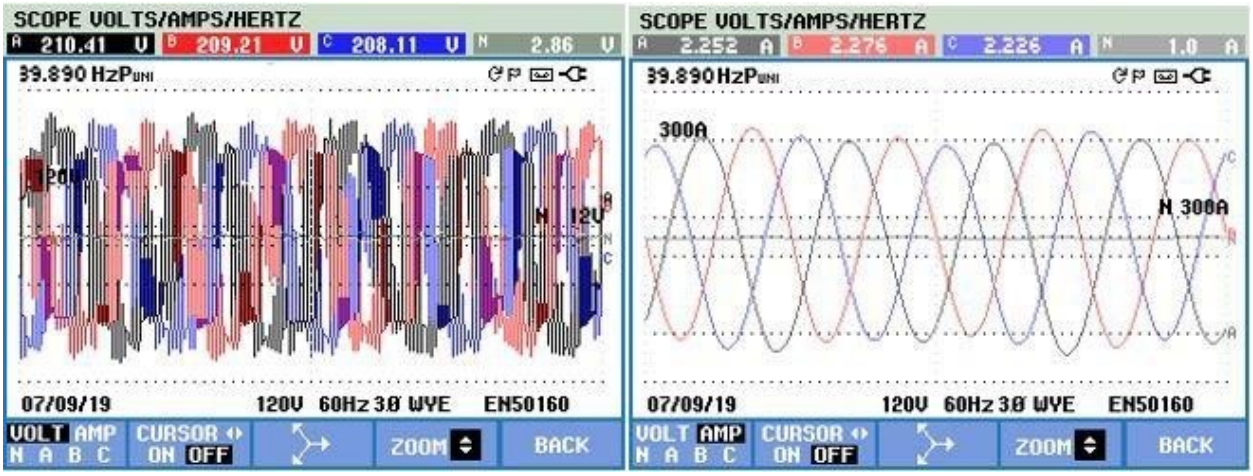


Fig.38. Output Voltage and Current at Load 1 at 40Hz

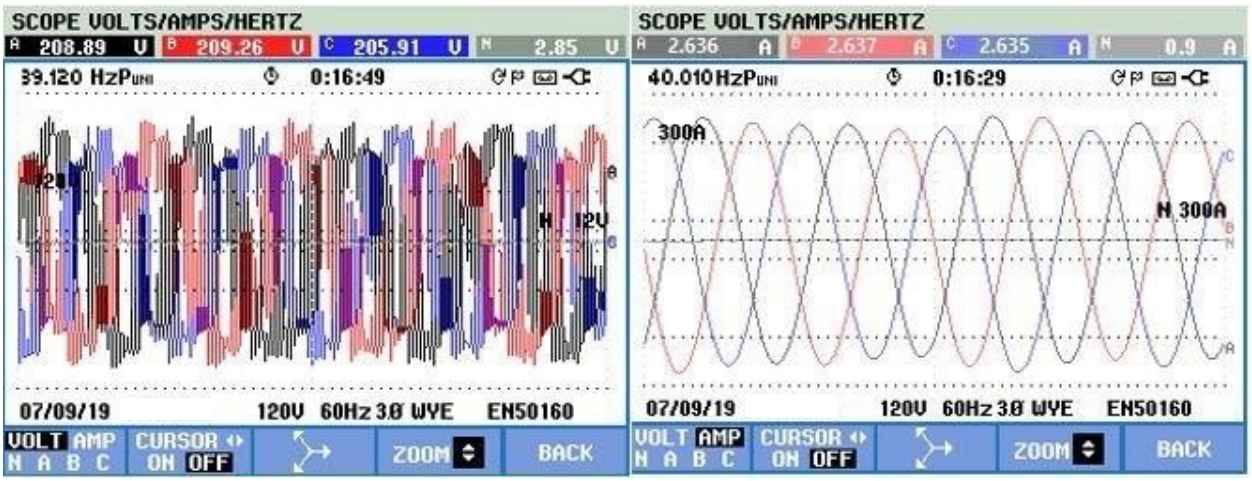


Fig.39. Output Voltage and Current at Load 2 at 40Hz

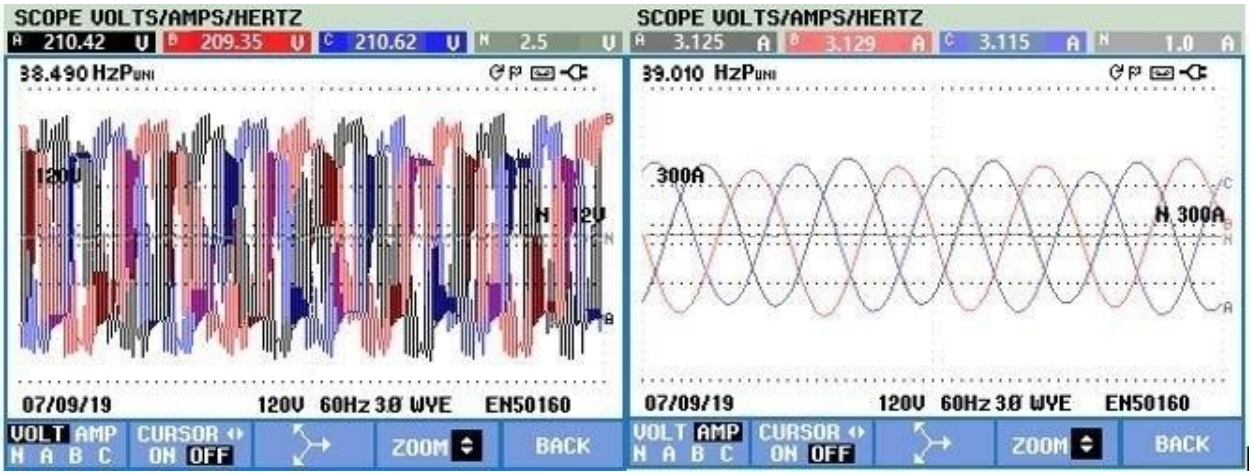


Fig.40. Output Voltage and Current at Load 3 at 40Hz

The load used here was resistive load which was connected to the armature of DC generator, the field of DC generator was connected to the DC supply of 300V. The resistive load was switched on one by one and the readings are taken for 50Hz and 40Hz respectively. Fig. 32 to Fig. 40 shows the output with varying loads, and the Table 2 and Table 3 shows the output for load voltage, load current, motor A.C voltage, motor A.C current, power and torque calculated.

Thus, on comparing Table 2 and Table 3 it can be observed that the output torque for varying load is nearly equal for two different frequency of 50Hz and 40Hz. It is because the V/f ratio is constant which makes the flux constant.

Also, on comparing the simulation output and hardware setup output it is observed that the output torque is nearly 3.25 Nm, with the angular speed nearly 1400 RPM for no load which drops to 1350 RPM for 40Hz.

5.4.2.3. Close Loop V/f Control at 50 Hz

S. no.	Output Power(W)	I (A.C)	I (D.C)	V (A.C)	V (D.C)	RPM	Output Torque (Nm)
1	No load	0.50	0.00	210	100	1410	No load
2	399.74	2.252	3.20	210	124.90	1390	2.5330
3	472.71	2.636	3.93	210	120.00	1380	3.0144
4	477.63	3.167	4.20	210	113.68	1365	3.0793

Table 4. Output for Close loop V/f control at 50 Hz

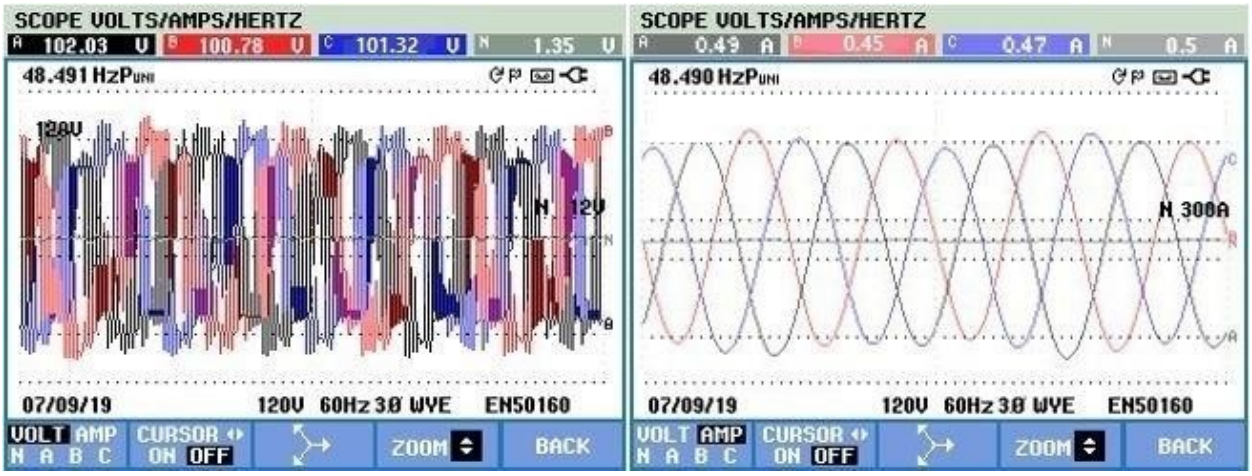


Fig.41. Output No load Output Voltage and Current at 50Hz

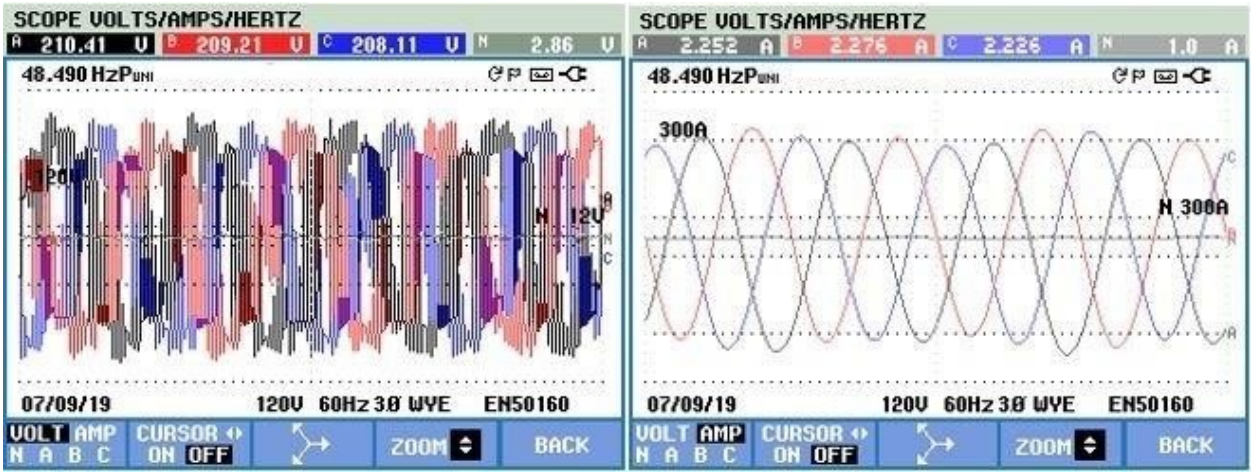


Fig.42. Output Voltage and Current at Load 1 at 50Hz

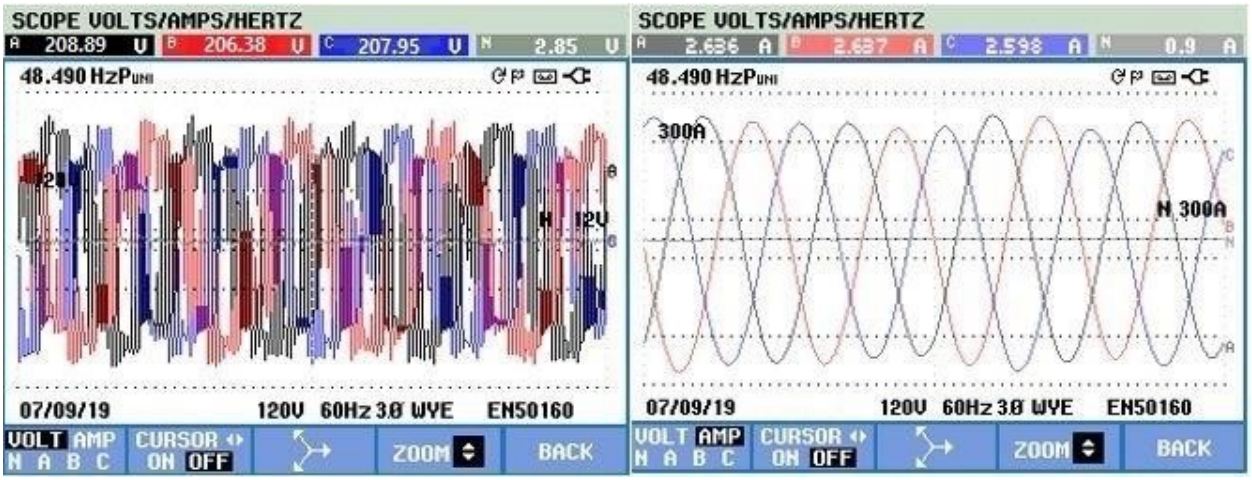


Fig.43. Output Voltage and Current at Load 2 at 50Hz

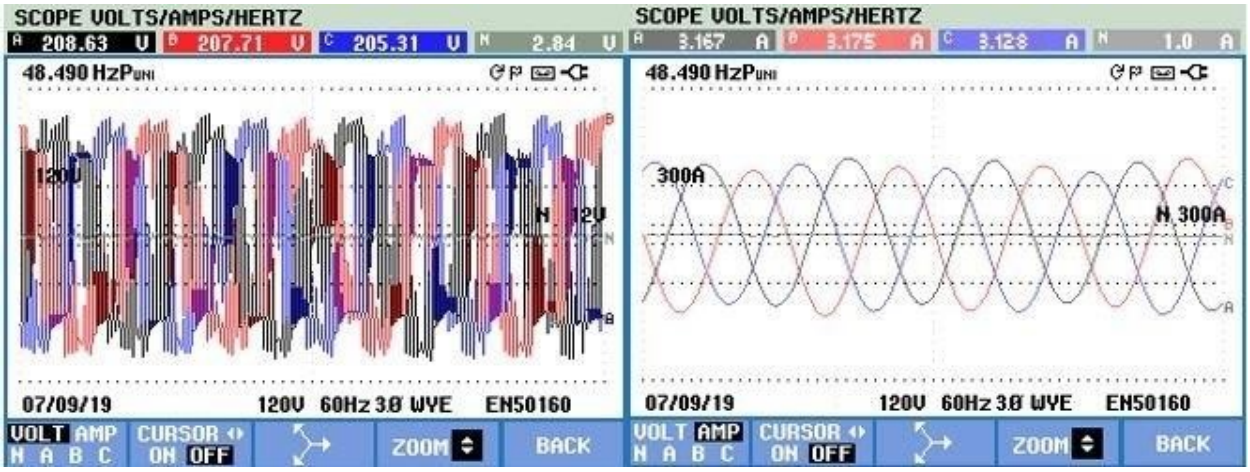


Fig.44. Output Voltage and Current at Load 3 at 50Hz

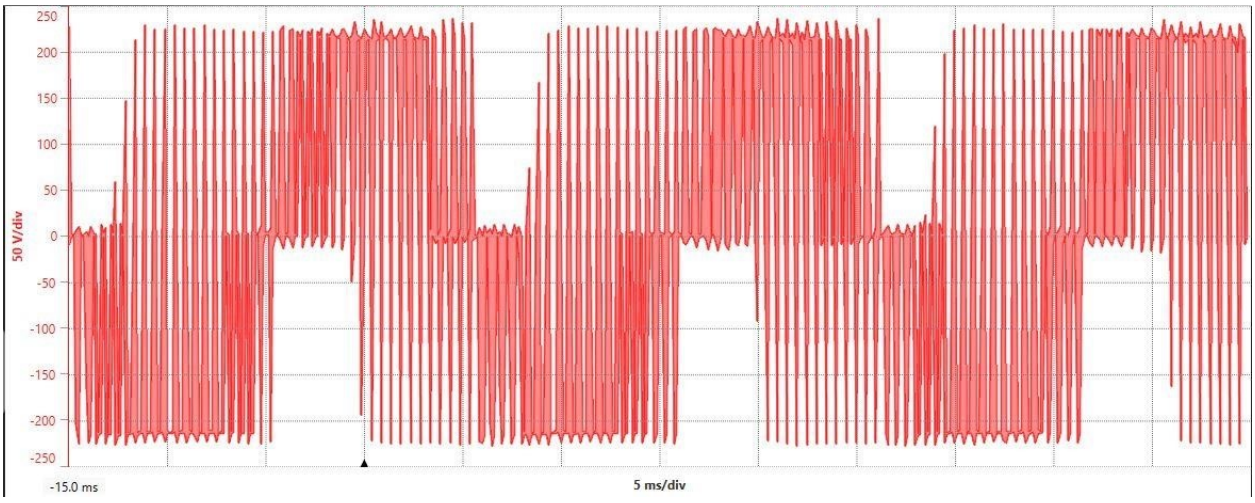


Fig.45. Output Single phase voltage at 50 Hz

5.4.2.4. Close loop V/f control at 40 Hz

S. no.	Output Power(W)	I (A.C)	I (D.C)	V (A.C)	V (D.C)	RPM	Output Torque (Nm)
1	No load	0.50	0.00	210	100	1370	No load
2	399.36	2.252	3.19	210	125.19	1354	2.5956
3	473.06	2.636	3.92	210	120.68	1346	3.0929
4	476.63	3.167	4.20	210	113.48	1338	3.1348

Table 5. Output for Close loop V/f control at 40 Hz

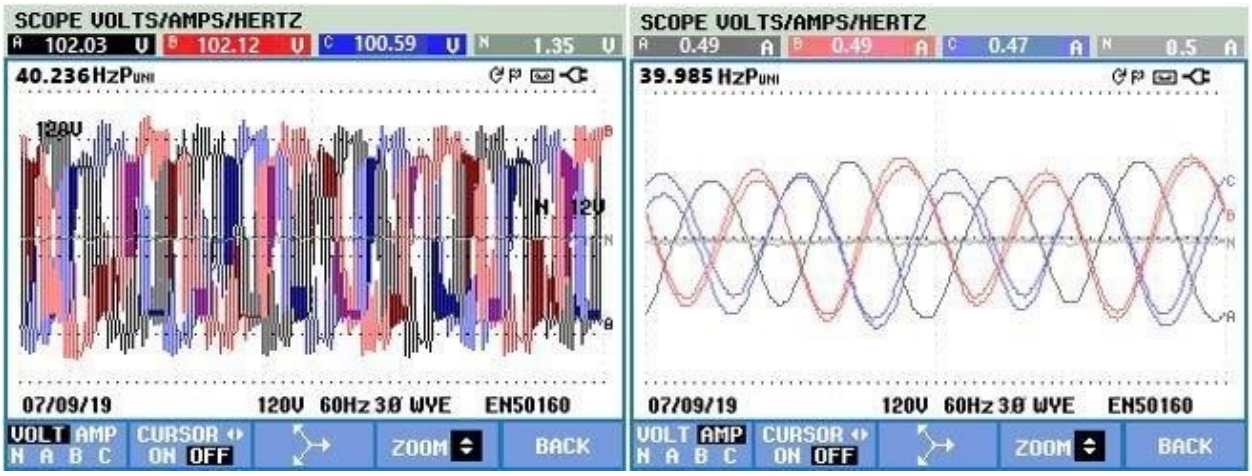


Fig.46. Output No load Output Voltage and Current at 40 Hz

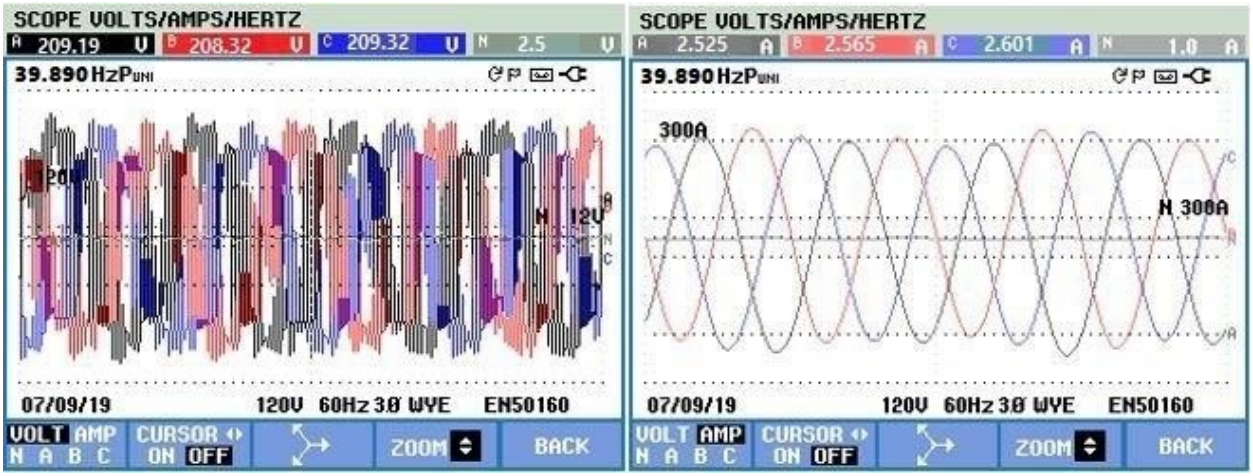


Fig.47. Output Voltage and Current at Load 1 at 40 Hz

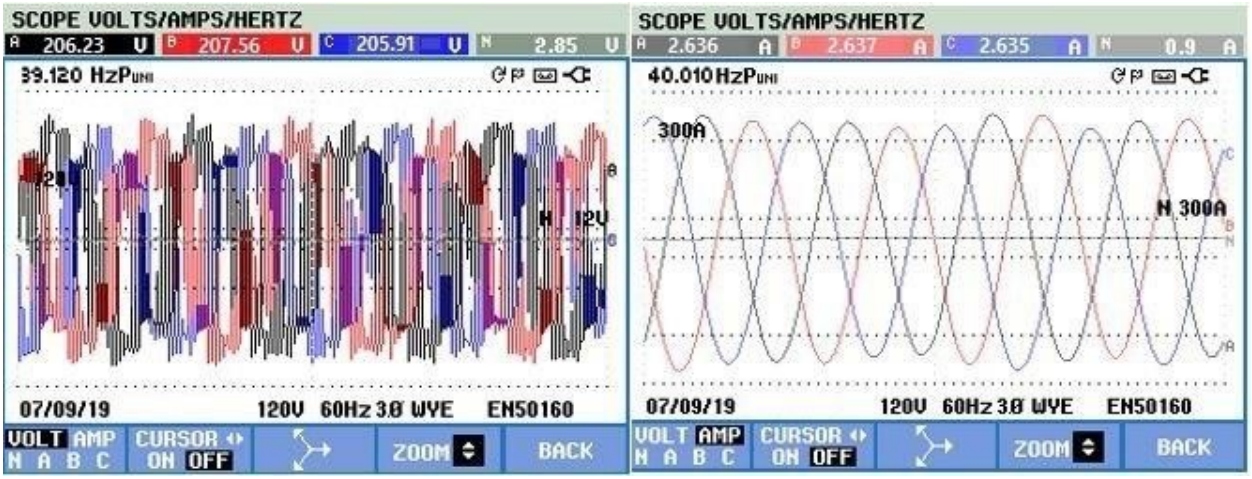


Fig.48. Output Voltage and Current at Load 2 at 40 Hz

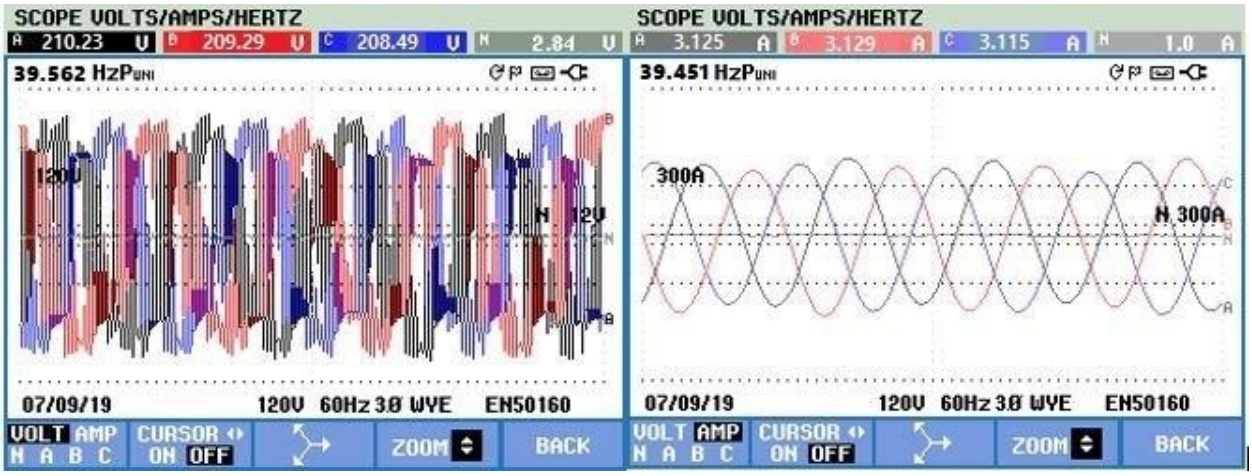


Fig.49. Output Voltage and Current at Load 3

The feedback of speed was taken using a tacho generator and this was passed through voltage sensor to be fed to the FPGA controller. The close loop operation was then done by the controller and the output has been plotted in Fig. 45 to Fig. 53. Table 4 and Table 5 shows the output voltage and current along with the calculation of output power and torque.

Thus, on comparing Table 4 and Table 5 it is observed that the output torque for varying load is nearly equal for two different frequency of 50Hz and 40Hz. It is because the V/f ratio is constant which makes the flux constant.

Also, on comparing the simulation output and hardware setup output it can be observed that the output torque is nearly 3 Nm.

It is also seen that the output obtained through closed loop control is closer to each other as compared to the open loop operation.

5.5. CONCLUSION

The offline and online simulation performed for V/f control if Induction Motor using MATLAB/Simulink and HSRPEC gives nearly same results, thus verifying the results. The output torque (Te) remains constant even on changing the frequency because the V/f ratio remains constant keeping the flux constant. For the close loop operation, the decrease in speed is small compared to the open loop operation.

CHAPTER 6
CONCLUSIONS AND FUTURE SCOPE OF WORK

6.1. CONCLUSION

The offline simulation of V/f control of Induction Motor is implemented in MATLAB/Simulink for both open as well as close loop operation of the Induction Motor.

In open loop application the operation of the Induction Motor at two different frequencies have been demonstrated and analyzed. It was observed that the output torque remains nearly constant for varying frequency as the V/f ratio is kept constant by varying the voltage in proportion to frequency.

In close loop application also the output torque remains constant for varying frequency as the V/f ratio is constant which is described by calculating the error between actual speed and reference speed and varying the frequency accordingly.

The online simulation using HSRPEC shows similar results for both open as well as close loop operation, where the torque was found to remain constant for varying operating frequencies.

Thus, the results of offline simulation are in consonance with the online simulation results verifying the V/f control of Induction Motor.

6.2. FUTURE SCOPE OF WORK

Scalar control of induction motor has been shown using V/f control technique on FPGA. Further real time implementation of vector control and Direct Torque control of three phase Induction Motor can be demonstrated. Also, the FPGA board can be used in various power electronic application like voltage mode control of boost converter, DC drives, MPPT applications, cyclo converters etc. FPGA can also be used in power system application like power factor correction, power quality application, smart metering and many more.

REFERENCES

[1] Nik Rumzi Nik Idris, Member, IEEE, and Abdul Halim Mohamed Yatim, IEEE, "An Improved Stator Flux Estimation in Steady-State Operation for Direct Torque Control of Induction Machines", IEEE Transactions on Industry Applications, Vol. 38, No. 1, January/February 2002, pp. 110-116

[2] Mohamad Koteich, "Flux estimation algorithms for electric drives: a comparative study", Inter-national Conference on Renewable Energies for Developing countries (REDEC 2016), Jul 2016, Zouk Mosbeh, Lebanon. REDEC 2016.

[3] Kevin D. Hurst, Member, IEEE, Thomas G. Habetler, Senior Member, IEEE, Giovanni Griva, and Francesco Profumo, Senior Member, IEEE, "Zero-Speed Tacholess IM Torque Control: Simply a Matter of Stator Voltage Integration", IEEE Transactions on Industry Applications, Vol. 34, No. 4, July/August 1998, pp. 790-795

[4] D. Seyoum, F. Rahman and C. Grantham, "An Improved Flux Estimation in Induction Machine for Control Application", unpublished.

[5] B. Srinu Naik, "Comparison of Direct and Indirect Vector Control of Induction Motor", International Journal of New Technologies in Science and Engineering Vol. 1, Issue. 1, Jan. 2014.67

[6] José Rodríguez, Senior Member, IEEE, Jorge Pontt, Senior Member, IEEE, César A. Silva, Member, IEEE, Pablo Correa, Pablo Lezana, Member, IEEE, Patricio Cortés, Student Member, IEEE, and Ulrich Ammann, "Predictive Current Control of a Voltage Source Inverter", IEEE Transactions on Industrial Electronics, Vol. 54, No. 1, February 2007, pp. 495-503.

[7] Sergio Vazquez, Jose I. Leon, Leopoldo G. Franquelo, Jose RodrÍguez, Hector A. Young, Abraham Marquez, and Pericle z\Zanchetta, "Model Predictive Control A Review of Its Applications in Power Electronics", IEEE industrial electronics magazine, march 2014.

[8] Yongchang Zhang, Member, IEEE, and Haitao Yang, Student Member, IEEE, "Model-Predictive Flux Control of Induction Motor Drives with Switching Instant Optimization", IEEE Transactions on Energy Conversion, Vol. 30, No. 3, September 2015, pp. 1113-1122.

[9] S. Alireza Davari, "Predictive Direct Angle Control of Induction Motor", IEEE Transactions on Industrial Electronics, Vol. 63, No. 8, August 2016, pp. 5276-5284.

[10] Ass. Prof. Yasser G. Dessouky and Eng. Mona Moussa, "Vector Controlled-Induction Motor Drive: Operation and Analysis", unpublished

[11] B. K. Bose and N. R. Patel, "A programmable cascaded low-pass filter based flux synthesis for a stator flux-oriented vector-controlled induction motor drive," IEEE Trans. Ind. Electron., vol. 44, pp. 140–143, Feb.1997.

[12] I. Takahashi and T. Noguchi, "A new quick response and high efficiency control strategy of an induction motor," IEEE Trans. Ind. Applicat., vol. 22, no. 5, pp. 820–827, 1986.

[13] X. Xu, R. Doncker, and D. W. Novotny, "A stator flux oriented induction machine drive," in IEEE PESC Conf. Rec., 1988, pp. 870–876.

[14] H. Tajima and Y. Hori, "Speed sensorless field oriented control of the induction machine," in IEEE IAS Conf. Rec., 1991, pp. 385–391.

[15] R. Wu and G. R. Slemon, "A permanent magnet motor drive without a shaft sensors," IEEE Transaction on Industry Applications, Vol. 27, No. 5, 1991, pp. 1005-1011.

[16] Xingi Xu, Rik De Doncker and Donald W. Novotny "A stator flux oriented induction machine drive," PESC 1988, pp. 870-876.

[17] T. A. Lipo and K. C. Chang, "A new approach to flux and torque sensing in induction machines," IEEE Trans. Ind. Applicat., vol. IA-22, pp. 731–737, July/Aug. 1986.

[18] Jean-Marie Retif, Xuefang Lin-Shi, Florent Morel, "Predictive Current Control for an Induction Motor", PESC, Jun 2008, Rhodes, Greece. pp.3463-3468, 2008.

[19] G. Victor Raj, K.Venkataramana and V.Chaitanya, "Predictive Vector Control Based Induction Motor Drive using Fuzzy Controller", International Journal of Science, Engineering and Technology Research (IJSETR), Volume 4, Issue 11, November 2015.

[20] Yongchang Zhang, Member, IEEE, and Haitao Yang, Student Member, IEEE, "Two-Vector-Based Model Predictive Torque Control without Weighting Factors for Induction Motor Drives", IEEE Transactions On Power Electronics, Vol. 31, No. 2, February 2016, pp. 1381-1390.69

[21] Fan, Sheng Wen, Xin Yong Zhao, Zheng Xi Li, and Hu Zhang. "A Flux Observer of Voltage Current Model Based on PI Regulator"Advanced Materials Research, 2011.

[22] [Online]. Available: https://en.wikipedia.org/wiki/Model_predictive_control

[23] P.L. Jansen. "A self-tuning, closed-loop flux observer for sensor less torque control of standard induction machines", Proceedings of PESC 95 - Power Electronics Specialist Conference PESC-95, 1995.

[24] Choudhury, Abhijit, and Kishore Chatterjee. "Modified stator flux estimation-based direct torque-controlled induction motor drive with constant switching frequency operation"International Journal of Power Electronics, 2012.

[25] R. Krishnan," Vector Control of Induction Motor Drives" in Electric Motor Drives modeling, analysis and control, 2nd ed, New Jersey, prentice hall ,2001, ch.8, pp. 411-513

[26] Bimal.K.Bose, "Control and Estimation of Induction Motor Drives" in Modern Power Electronics and AC Drives, 2002,ch.8, pp.333-439

[27] Jose Rodriguez and Patricio Cortes, "Model Predictive Control" in Predictive Control of Power Converters and Electrical Drives,1st ed, Wiley,20012, ch.3, pp 31-38.

[28] Jose Rodriguez and Patricio Cortes, "Predictive Control of a Three-Phase Inverter" in Predictive Control of Power Converters and Electrical Drives,1st ed, Wiley,20012, ch.4, pp 43-63.

[29] A. k. Devangan, N. Chakraborty, S. Shukla "PWM based closed loop speed control of DC motor" International conference of Engineering Trends and Technology (IJETT) 2012, pp 110-112.

[30] K. L. Shi, T. F. Chan, Y. K. Wong and S. L. HO, "Modeling and simulation of the three phase induction motor Using SIMULINK," Int.J. Elect. Engg. Educ., 1999, pp. 163–172.

[31] P. C. Krause, C. H. Thomas, "Simulation of Symmetrical Induction Machinery," IEEE Trans on Power Apparatus and Systems 1965, pp 1038-1053.

[32] Armando Astarloa, Unai Bidarte, Jesus Lazaro, Jon Andreu, and Jose Luis Martın, Configurable-System-on Programmable-Chip for Power Electronics Control Applications, International Conference on Reconfigurable Computing and FPGAs, pp. 169 – 174, 2008

[33] Henrik Svensson, Reconfigurable Architectures for Embedded Systems, Lund University, Sweden, 2008.

[34] H. L. Huy, Microprocessors and Digital ICs for Motion Control, Proceedings of the IEEE, 1994, pp. 1140-1163

[35] G. J. van Heerden and H du T. Mouton, Design of a DSP based controller for power electronics applications, 6th Africon Conference in Africa, vol.2,pp. 743 - 747, 2002 [5] K. Sugahara, S. Oida, T. Yokoyama, High performance FPGA controller for digital control of power electronics applications, IEEE 6th International Power Electronics and Motion Control Conference, pp. 1425 - 1429, 2009.

[36] K.Sudha, L Kathir, Implementation of FPGA Based Controller for Induction Motor Drives, International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC), 2013

[37] E. Monmasson, L. Idkhajine, I. Bahri, M-W- Naouar and L. Charaabi, Design methodology and FPGA-based controllers for Power Electronics and drive applications, Pacific-Asia Conference on Circuits, Communications and System, pp. 360-363, 2009

[38] Jason G. Tong, Ian D. L. Anderson and Mohammed A. S. Khalid, Soft-Core Processors for Embedded Systems, International Conference on Microelectronics, 2006

[39] Ding Zheng, Yichen Wang and Zou Xueyi, The methods of FPGA software verification, IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2011

[40] S. Oliveirada and S. Marcelo, "Scalar control of an induction motor using a neural sensor less technique," Elsevier Electric Power System Research, 2014, pp 322- 320.

[41] Abbondanti, "Method of flux control in induction motors driven by variable frequency, variable voltage supplies", IEEE/IAS Intl. Semi. Power Conv. Conf., 1977 pp. 177- 184.

[42] Alfredo,Thomas A. Lipo and Donald W. Novotny, "A New Inductio Motor V/f Control Method Capable of High-Performance Regulation at Low Speeds" IEEE Trans. Industry Applications, Vol. 34, No. 4 July/ August 1998.

[43] B. Biswas, S. Das, P. Purkait, M. S. Mandal and D. Mitra," Current Harmonic Analysis of Inverter-Fed Induction Motor Drive System under Fault Conditions", International Conference of Engineers and Computer Scientists 2009 Vol II IMECS 2009

[44] Ku. Trupti Deoram Tembhekar "A constant v/f open loop and closed loop speed control of a three phase induction motor drive"

[45] Masayuki Morimoto, Kiyotaka Sumito, Shinji Sato, Katsumi Oshitani, Shigeru Okuma" High efficiency, unity power factor VVVF drive system of an induction motor" IEEE transactions on power electronics. Vol 6. No.3.July 1991.

[46] Ned Mohan, Tore M. Undelad, William P. Robbins, 1995, "Power Electronics Converters, Applications and Design", Wiley, New York.

[47] Rodolfo Echavada, Sergio Horta, Marc0 Oliver," A Three phase motor drive using IGBT"s and Constant V/F speed control with slip regulation", 0- 7803-3071-4/95 1995 IEEE

[48] Rashid M.H, Power Electronics-Circuits, Devices and Applications", third edition Printice HallIndia,2001.

[49] Thida Win, Nang Sabai, and Hnin Nandar Maung" Analysis of variable frequency three phase induction motor drive". World academy of science, Engineering and technology 2008.

[50] M.S. Aspalli, Vinaya Kumar, P V Hunagund, Development and Analysis of Variable Frequency Three Phase Induction Motor Drive, IJ-ETA-ETS, July 10-Dec 10, Vol.3, Issue2: PP 189-195.

[51] M.S.Aspalli, Veerendra.D, P.V.Hunagund, A New generation VLSI approach for V/F control of Three-Phase Induction Motor. Proceedings of the International Conference on VLSI, Communication and Instrumentation, April7th-9th, 2011, Kottayam, India.

[52] M Harsha Vardhan Reddy and V. Jegathesan, "Open loop V/f Control of Induction Motor based on hybrid PWM with Reduced Torque Ripple", ICETECT 2011, Karunya University.

APPENDIX I

C CODE FOR INITIATION

```
#include "sys/alt_stdio.h"
#include "system.h"
#include <stdio.h>
#include <stdlib.h>
#include <alt_types.h>
#include "ADC_PEG_V1.h"
#include "DAC_PEG_V1_regs.h"
#include "PI_PEG_V1_regs.h"
#include "PWM_PEG_V1_regs.h"
#include "PWM_3PH_PEG_V1_regs.h"
#include "TIMER_PEG_V1_regs.h"
#include "ALPHA_BETA_TO_ABC_PEG_V1.h"
#include "reg_values.h"
#include <sys/alt_irq.h>

extern void ISR(alt_u32);
static void ISR_legacy_irqhandler (void* , alt_u32 );

alt_u8 init_hw_ips()
{
      alt_u8 result=0;

            /*-------------------------------------------------------------
---------------
             *Initializing PWM for 3ph Inverter
             *-------------------------------------------------------------
---------------
             */

      init_three_ph_pwm(PWM_3PH_PEG_V1_0_BASE,pwm_max_cnt_inverter,pwm_dead_c
nt_inverter);//base,max_cnt,dead_cnt

      ctrl_reg_three_ph_pwm(PWM_3PH_PEG_V1_0_BASE,pwm_ctrl_reg_inverter);//Ba
se,Ctrl reg
            /*-------------------------------------------------------------
---------------
             *Initializing PWM for Boost Converter
             *-------------------------------------------------------------
---------------
             */

      init_pwm(PWM_PEG_V1_0_BASE,pwm_max_cnt_boost,pwm_dead_cnt_boost);//base
,max_cnt,dead_cnt
          ctrl_reg_pwm(PWM_PEG_V1_0_BASE,pwm_ctrl_reg_boost);//Base,Ctrl reg
          /*-------------------------------------------------------------
-------------
           *Initializing ADC
           *-------------------------------------------------------------
-------------
           */
```

```
          enable_adc(ADC_PEG_V1_0_BASE,adc_ctrlr_ctrl_reg);//base,Control
          adc_in(ADC_PEG_V1_0_BASE,adc_ran_reg_1);  //base, data_in_ADC
          adc_in(ADC_PEG_V1_0_BASE,adc_ran_reg_2);  //base, data_in_ADC
          adc_in(ADC_PEG_V1_0_BASE,adc_seq_reg);  //base, data_in_ADC
          adc_in(ADC_PEG_V1_0_BASE,adc_ctrl_reg);   //base, data_in_ADC
          adc_in(ADC_PEG_V1_0_BASE,0x0000);   //base, data_in_ADC

          /*enable_adc(ADC_PEG_V1_0_BASE,adc_ctrlr_ctrl_reg);//base,Control
          adc_in(ADC_PEG_V1_0_BASE,adc_ctrl_reg);   //base, data_in_ADC
          adc_in(ADC_PEG_V1_0_BASE,adc_seq_reg);  //base, data_in_ADC
          adc_in(ADC_PEG_V1_0_BASE,adc_ran_reg_1);  //base, data_in_ADC
          adc_in(ADC_PEG_V1_0_BASE,adc_ran_reg_2);  //base, data_in_ADC
          */
           /*-------------------------------------------------------------------
--------------
           *Initializing PI Controller
           *-------------------------------------------------------------------
-------------
           */

      init_pi(PI_PEG_V1_0_BASE,pi_Kp,pi_Ki,0x00000000,0x00000000);//Base,Kp,K
i,anti_windup_max, anti_windup_min registers
            enable_pi(PI_PEG_V1_0_BASE,pi_ctrl_reg);//Enable PI
            /*-------------------------------------------------------------------
---------------
             *Initializing DAC
             *-------------------------------------------------------------------
---------------
             */
            enable_dac(DAC_PEG_V1_0_BASE,dac_ctrlr_ctrl_reg);//base,Control
            dac_in(DAC_PEG_V1_0_BASE,dac_ctrl_0_reg);//base, data_in_DAC
            /*-------------------------------------------------------------------
---------------
             *Initializing TIMER
             *-------------------------------------------------------------------
---------------
             */
            hw_prd_timer(TIMER_PEG_V1_0_BASE,TIMER_PRD);

alt_irq_register(TIMER_PEG_V1_0_IRQ,NULL,ISR_legacy_irqhandler);//register ir
of PWM
          hw_ctrl_reg_timer(TIMER_PEG_V1_0_BASE,0x01);

          init_alphabeta_abc(ALPHA_BETA_TO_ABC_PEG_V1_0_BASE,0x0001);

          return(result);
}

static void ISR_legacy_irqhandler (void* context, alt_u32 id)//Interrupt
handler
{
      alt_irq_disable(id);
      ISR(id);
      alt_irq_enable(id);
}
```

C CODE FOR REGISTER DEFINITION

```
#ifndef REG_VALUES_H_
#define REG_VALUES_H_
/*-------------------------------------------------------------------------
---
* Register values for PWM (boost Converter)
* Fcarrier = 10kHz
*-------------------------------------------------------------------------
--
*/
#define pwm_max_cnt_boost 0x2710
#define pwm_dead_cnt_boost 0x000
#define pwm_ctrl_reg_boost 0x000F
/*-------------------------------------------------------------------------
---
* Register values for PWM (Inverter)
* Fcarrier = 2kHz
*-------------------------------------------------------------------------
--
*/
#define pwm_max_cnt_inverter 0xC350
#define pwm_dead_cnt_inverter 0x190
#define pwm_ctrl_reg_inverter 0x000F
/*-------------------------------------------------------------------------
---
* Register values for PI Controller
*-------------------------------------------------------------------------
--
*/
#define pi_Kp 0xA
#define pi_Ki 0x0000
#define pi_ctrl_reg 0x0001

/*-------------------------------------------------------------------------
---
* Register values for Serial SPI ADC Controller
*-------------------------------------------------------------------------
--
*/
#define adc_ctrlr_ctrl_reg 0x0001
#define adc_ctrl_reg 0x8834
#define adc_seq_reg 0xFC00
#define adc_ran_reg_1 0xBF80
#define adc_ran_reg_2 0xC000

/*#define adc_ctrlr_ctrl_reg 0x0001
#define adc_ctrl_reg 0x8030
#define adc_seq_reg 0xE000
#define adc_ran_reg_1 0xBFE0
#define adc_ran_reg_2 0xDFE0
```

```
*/
/*-------------------------------------------------------------------------
---
* Register values for Serial SPI DAC Controller
*-------------------------------------------------------------------------
--
*/
#define dac_ctrlr_ctrl_reg 0x0001
#define dac_preset_reg 0x000C
#define dac_ctrl_0_reg 0x8006
#define dac_ctrl_1_reg 0x9000

#define TIMER_PRD 0x384

/*-------------------------------------------------------------------------
---
* Voltages, Currents and Scale values
*-------------------------------------------------------------------------
--
*/
#define v_dc_bus_init 60
#define voltage_scale  89.964

#endif /* REG_VALUES_H_ */
```

C CODE FOR V/F CONTROL

```
#include "sys/alt_stdio.h"
#include "system.h"
#include <stdio.h>
#include <stdlib.h>
#include <alt_types.h>
#include "altera_avalon_pio_regs.h"
#include "ADC_PEG_V1.h"
#include "DAC_PEG_V1_regs.h"
#include "PI_PEG_V1_regs.h"
#include "PWM_PEG_V1_regs.h"
#include "PWM_3PH_PEG_V1_regs.h"
#include "TIMER_PEG_V1_regs.h"
#include "ALPHA_BETA_TO_ABC_PEG_V1.h"
#include "ALPHA_BETA_TO_ABC_PEG_V1_regs.h"
#include "reg_values.h"
#include <sys/alt_irq.h>
```

```
/*------------------------------------------------------------------------
---
* Function Definitions
*-------------------------------------------------------------------------
--
*/
void ISR(alt_u32);
alt_u8 init_hw_ips();
/*------------------------------------------------------------------------
---
*
*-------------------------------------------------------------------------
--
*/
/*------------------------------------------------------------------------
---
* Global Variables
*-------------------------------------------------------------------------
--
*/
alt_u16 xyz=0;
alt_u16 abc=0;
alt_16 pi_out[50];
alt_u16 pi_out1[3000];
alt_u16 pi_out2[3000];
alt_u16 pi_out3[3000];
alt_u16 adc_in_ch1[2000];
alt_16 table[50];
alt_u16 Vph;
```

```
alt_u16 theta_new;
alt_u16 theta_old;
alt_u16 pwm_a_ph_ref;
alt_u16 pwm_b_ph_ref;
alt_u16 pwm_c_ph_ref;
alt_u16 vref_hex,v_dc_init_hex;
alt_u32 count=0;
alt_u32 count2=0;
alt_u16 pwm_input=0;
alt_u8 interrupt_flag = 0;
alt_16
sine_table[2000]={0,6,12,19,25,32,38,45,51,57,64,70,77,83,90,96,102,109,115,1
22,128,134,141,147,154,160,167,173,179,186,192,199,205…986,991,997,1003,1008,
1014,1019,1025,1030,1036,1042,1047,1053,1058,1064,1069,1075,1080,1085,1091,10
96,1102,1107,1113,1118,1123,1129,1134,1139,1145,1150,1155,1161,1166,1171,1177
,1182,1187,1192,1197,1203…1999,2000,2001,2003,2004,2005,2007,2008,2009,2010,2
011,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022,2023,2024,2025,2026,202
7,2028,2029,2030…2000,1999,1997….1003,997,991,986,980,974,969,963,957,952,946
,940,935,929,923,917,912,906,900,894,888,883,877,871,865,859,854,848,842,836,
830,824,818,812,807,801,795,789,783,777,771,765,759,753,747,741,735,729,723,7
17,711,705,699,693,687,681,675,669,663,656,650,644,638,632,626,620,614,608,60
1,595,589,583,577,571,564,558,552,546,540,533,527,521,515,509,502,496,490,484
,477,471,465,459,452,446,440,433,427,421…0,-6,-12,-19,-25,-32,-38,-45,-51,-
57,-64,-70,-77,-83,-90,-96,-102,-109,-115,-122,-128…-1003,-1008,-1014,-1019,-
1025,-1030,-1036,-1042,-1047….-2000,-2001,-2003,-2004,-2005,-2007,-2008,-
2009,-2010,-2011,-2013,-2014,-2015,-2016,-2017,-2018…-2046,-2045,-2045,-
2045,-2045,-2045,-2044,-2044,-2044,-2043,-2043,-2042,-2042,-2042,-2041,-
2041,-2040,-2040,-2039,-2039,-2038,-2037,-2037,-2036,-2036,-2035,-2034,-
2033,-2033,-2032,-2031….-1003,-997,-991,-986,-980,-974,-969,-963,-957,-952,-
946,-940,-935,-929,-923,-917,-912,-906,-900,-894,-888,-883,-877,-871,-865,-
859,-854,-848,-842,-836,-830,-824,-818,-812,-807,-801,-795,-789,-783,-777,-
771,-765,-759,-753,-747,-741,-735,-729,-723,-717,-711,-705,-699,-693,-687,-
681,-675,-669,-663,-656,-650,-644,-638,-632,-626,-620,-614,-608,-601,-595,-
589,-583,-577,-571,-564,-558,-552,-546,-540,-533,-527,-521,-515,-509,-502,-
496,-490,-484,-477,-471,-465,-459,-452,-446,-440,-433,-427,-421,-415,-408,-
402,-396,-389,-383,-377,-370,-364,-358,-351,-345,-339,-332,-326,-320,-313,-
307,-301,-294,-288,-282,-275,-269,-262,-256,-250,-243,-237,-231,-224,-218,-
211,-205,-199,-192,-186,-179,-173,-167,-160,-154,-147,-141,-134,-128,-122,-
115,-109,-102,-96,-90,-83,-77,-70,-64,-57,-51,-45,-38,-32,-25,-19,-12,-6};
alt_16
cos_table[2000]={2047,2046,2046,2046,2046,2046,2046,2046,2046,2046,2045,2045,
2045,2045,2045,2044,2044,2044,2043,2043,2042,2042,2042,2041,2041,2040,2040,20
39,2039,2038,2037,2037,2036,2036,2035,2034,2033,2033,2032,2031,2030…2000,1999
,1997,1996,1994,1993,1991,1990…1900,1898,1896,1893,1891,1888,1886,1883,1881,1
878,1876,1873,1870,1868,1865….1500,1496,1492,1487,1483,1478,1474,1470,1465,14
61,1456,1451,1447,1442,1438,1433,1429,1424,1419,1415,1410,1405,1401….1003,997
,991,986,980,974,969,963,957,952,946,940….134,128,122,115,109,102,96,90,83,77
,70,64,57,51,45,38,32,25,19,12,6,0,-6,-12,-19,-25……-402,-408,-415,-421,-427,-
433,-440,-446,-452,-459,-465,-471,-477,-484,-490,-496,-502……801,-807,-812,-
818,-824,-830,-836,-842,-848,-854,-859,-865,-871,-877,-883,-888,-894,-900,-
906,-912,-917,-923,-929,-935,-940,-946,-952,-957,-963,-969,-974,-980,-986,-
991,-997…… -1500,-1505,-1509,-1514,-1518,-1522,-1526,-1531,-1535,-1539,-
1543,-1548,-1552,-1556,-1560,-1564,-1569,-1573,-1577,-1581,-1585,-1589,-
1593,-1597,-1600….-2000,-2001,-2003,-2004,-2005,-2007,-2008,-2009,-2010,-
2011,-2013,-2014,-2015,-2016,-2017,-2018,-2019,-2020,-2021,-2022,-2023,-
2024,-2025,-2026,-2027,-2028,-2029,-2030,-2030,-2031,-2032,-2033,-2033,-
2034,-2035,-2036,-2036,-2037,-2037,-2038,-2039,-2039,-2040,-2040,-2041,-
```

```
2041,-2042,-2042,-2042,-2043,-2043,-2044,-2044,-2044,-2045,-2045,-2045,-
2045,-2045,-2046,-2046,-2046,-2046,-2046,-2046,-2046,-2046,-2047,-
2046,-2046,-2046,-2046,-2046,-2046,-2046,-2046,-2046,-2045,-2045……-2000,-
1999,-1997,-1996,-1994,-1993,-1991,-1990,-1988,-1987,-1985,-1984,-1982,-
1981,-1979,-1977,-1976,-1974,-1972,-1971,-1969,-1967,-1965,-1963,-1962,-
1960,-1958,-1956,-1954,-1952,-1950,-1948,-1946,-1944,-1942,-1940,-1938,-
1936,-1934,-1932,-1930,-1928,-1925,-1923,-1921,-1919,-1917,-1914,-1912,-
1910,-1907,-1905,-1903,-1900……..-1003,-997,-991,-986,-980,-974,-969,-963,-
957,-952,-946,-940,-935,-929,-923,-917,-912,-906,-
900…0,6,12,19,25,32,38,45,51,57,64,70,77,83,90,96,102……997,1003,1008,1014,101
9,1025,1030,1036,1042,1047,1053,1058,1064,1069,1075,1080,1085,1091,1096,1102,
1107,1113,1118,1123,1129,1134,1139,1145,1150,1155,1161,1166,1171,1177,1182,11
87,1192,1197,1203….1500,1505,1509,1514,1518,1522,1526,1531,1535,1539,1543,15
48,1552,1556,1560,1564,1569,1573,1577,1581,1585,1589,1593,1597,1601,1605,1609
,1613,1617,1621,1625,1629,1633,1636,1640,1644,1648,1652,1656,1659,1663,1667,1
671,1674,1678,1682,1685,1689,1693,1696,1700,1703……2000,2001,2003,2004,2005,20
07,2008,2009,2010,2011,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022,2023
,2024,2025,2026,2027,2028,2029,2030,2030,2031,2032,2033,2033,2034,2035,2036,2
036,2037,2037,2038,2039,2039,2040,2040,2041,2041,2042,2042,2042,2043,2043,204
4,2044,2044,2045,2045,2045,2045,2045,2046,2046,2046,2046,2046,2046,2046,2046,
2046};
int main()
{

      float voltage ;
      alt_u16 dc_bus_voltage;
      alt_u16 w_set;
      alt_u32 m_fact;
      alt_u32 speed_fact;
      alt_u16 del_speed;
      alt_u32 dac_data;
      alt_16 error;
      alt_u16 incr=0;
      alt_u16 del_v,limit;
      alt_u8 result=0;
      alt_u16 Vph;
      alt_u16 pwm_a_ph_ref=0,pwm_b_ph_ref=0, pwm_c_ph_ref=0;
      alt_16 pwm_a_ph_ref_1=0,pwm_b_ph_ref_1=0, pwm_c_ph_ref_1=0;
      alt_u32 test_val;
      static alt_u8 soft_start_drive=0;
      static unsigned int index1=0,index2=666,index3=1333;

          voltage = 60 * voltage_scale;
          vref_hex = (alt_u16)voltage ;
          v_dc_init_hex = vref_hex;
          theta_old = 0;
            alt_putstr("[Application: AC_DRIVE Initialization]\n");
            if(0 == result)
            {
                  alt_putstr("[Application: Initialized the System]\n");
            }
            result = init_hw_ips();

        while(1)
        {
            if(0==interrupt_flag)
            {
```

```
                interrupt_flag = 1;
                IOWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE,0x1);
                dc_bus_voltage = adc_read_output(ADC_PEG_V1_0_BASE,1);
                dc_bus_voltage = dc_bus_voltage << 3;

                if(count <100000)
                {
                      error = (alt_16)(vref_hex - dc_bus_voltage);
                      if(error<0)
                      {
                            error=0;
                      }
                      if(error<=0x383)
                      {
                            pi_in(PI_PEG_V1_0_BASE,error);
                            pwm_input = pi_read_output(PI_PEG_V1_0_BASE);
                      }
                      else
                      {
                            pwm_input =0x2328;
                      }
                      m_fact = count*0x14B9;
                      del_v =m_fact>>16;
                      vref_hex = 0x1515 + del_v;
                      count = count + 1;
                }
                else
                {
                /*-------------------------------------------------------
----------------------
                 *Normal Operation
                 *Maximum Error Voltage = 10V =0x383
                 *Maximum Duty = 90 % =0x2328
                 *-------------------------------------------------------
----------------------
                 */
                      error = (alt_16)(vref_hex - dc_bus_voltage);
                      if(error<0)
                      {
                            error=0;
                      }
                      if(0x383>=error)
                      {
                            pi_in(PI_PEG_V1_0_BASE,error);
                          pwm_input = pi_read_output(PI_PEG_V1_0_BASE);
                      }
                      else
                      {
                            pwm_input =0x2328;
                      }
                      if(0==soft_start_drive)
                      {
                            w_set = adc_read_output(ADC_PEG_V1_0_BASE,0);
                            w_set =0x7FF;
                            w_set =(w_set *0xD51)>>12;
                            if(w_set<0x82) // 10 rad/s
                            {
```

```
                              w_set=0x82;
                        }
                  }
                  if(count2<100000)
                  {
                   del_speed = (w_set-0x82)*0xA;
                   speed_fact = del_speed *count2;
                   speed_fact = speed_fact >>20;
                   w_set= 0x82 + speed_fact;
                   count2 ++;
                   soft_start_drive =1;
                  }
                  else
                  {
                        w_set = adc_read_output(ADC_PEG_V1_0_BASE,0);
                        w_set =(w_set *0xD51)>>12;
                        w_set = 0x7FF;
                        if(w_set<0x82)
                        {
                              w_set=0x82;
                        }
                  }
                        //Vph= 0x681+0x19*((0x732*w_set)>>12);
                    Vph = 0x664+0xA6*((0x4c*w_set)>>12);
                    //Vph=0xC350;
                  theta_new = (w_set*0x10*incr)>>16;

                  if(theta_new >= 2000)
                        {
                        theta_new=0;
                        incr =0 ;
                        }

                        incr++;
                  //IOWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE,0x1);

      alphabeta_abc_in(ALPHA_BETA_TO_ABC_PEG_V1_0_BASE,sine_table[theta_new],
cos_table[theta_new],Vph);

alphabeta_abc_read(ALPHA_BETA_TO_ABC_PEG_V1_0_BASE,&pwm_a_ph_ref,&pwm_b_ph_re
f,&pwm_c_ph_ref);
                  //IOWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE,0x0);

ref_three_ph_pwm(PWM_3PH_PEG_V1_0_BASE,pwm_a_ph_ref,pwm_c_ph_ref,pwm_b_ph_ref
);
                pwm_a_ph_ref = (pwm_a_ph_ref *0x114E)>>16;
                  dac_data = pwm_a_ph_ref & 0x0FF0;
                  dac_data = dac_data | 0x1000;    //The 4th digit
shows Channel ID ranging from 0 to 7
                  dac_in(DAC_PEG_V1_0_BASE,dac_data);    // Taking
120ns for execution in 100 MHz Clk

                  pwm_b_ph_ref = (pwm_b_ph_ref *0x114E)>>16;
                  dac_data = pwm_b_ph_ref & 0x0FF0;
                  dac_data = dac_data | 0x2000;    //The 4th digit
shows Channel ID ranging from 0 to 7
                  dac_in(DAC_PEG_V1_0_BASE,dac_data);
```

```
                        pwm_c_ph_ref = (pwm_c_ph_ref *0x114E)>>16;
                        dac_data = pwm_c_ph_ref & 0x0FF0;
                        dac_data = dac_data | 0x3000;    //The 4th digit
shows Channel ID ranging from 0 to 7
                        dac_in(DAC_PEG_V1_0_BASE,dac_data);
                  }
                  /*-------------------------------------------------------------
---------------------
                   *V/F
                   *-------------------------------------------------------------
---------------------
                   */

                        ref_pwm(PWM_PEG_V1_0_BASE,pwm_input);
                        IOWR_ALTERA_AVALON_PIO_DATA(PIO_0_BASE,0x0);

            }

        }
  /* Event loop never exits. */
  return 0;
}
/*-----------------------------------------------------------------------------
---
 *PWM ISR
 *-----------------------------------------------------------------------------
---
*/
void ISR(alt_u32 id)
{
      if(id==5)
      {
            hw_timer_interrupt_ack(TIMER_PEG_V1_0_BASE);
            interrupt_flag =0;
      }
}
```

APPENDIX II

SPECIFICATIONS OF HSRPEC

1. CONTROLLER CARD: Fixed design hardware for variety of PE control applications
2. FPGA: Cyclone III EP3C25E144C8N (24,624 LEs)
3. On chip memory: 64 Kbytes (inside FPGA)
4. Flash memory: 2 MB
5. Digital I/Os: 55 Nos. (3.3-V LVTTL)
6. Host Interface: JTAG
7. Supply voltage: 3.3 V
8. INTERFACE CARD: Interface Card interfaces the controller to the Power Electronic systems
9. Analog Input: +10V (ADC 13-bit 8 Channel, SPI interface)
10. Analog Output: +3.3V (DAC 12-bit 8 channel, SPI interface)
11. Digital Input: +5 V 2 Nos.
12. Digital Output: +5 V 12 Nos.
13. Supply voltage: 12 V

Features and Technical Specifications

1. Soft processor integrated reconfigurable PE Controller
2. Application dependent processor design
3. An exclusive Power Electronics specific IPs Library (PWM generator, PI, ADC, DAC Controller, SPI, UART etc.)
4. Custom made instructions/functions for PE IP Library
5. Explores the idea of Hardware parallel processing than software pipelining
6. Reduced software/coding overhead

NIOS II Processor

1. Full 32-bit instruction set, data path, and address space
2. 32 general-purpose registers
3. 32 external interrupt sources
4. Single-instruction 32 x32 multiply and divide producing a 32-bit result.
5. Dedicated instructions for computing 64-bit and 128-bit products of multiplication.
6. Single-instruction barrel shifter
7. Access to a variety of on-chip peripherals, and interfaces to off-chip memories and peripherals
8. Single-instruction barrel shifter
9. Hardware-assisted debug module enabling processor start, stop, step and trace under Integrated Development Environment (IDE) control.
10. Software development environment based on the GNU C/C++ tool chain and Eclipse IDE.

11. Instruction Set Architecture (ISA) compatible across all Nios II processor systems.
12. Performance beyond 150 DMIPS
13. Variable clock rate depends on FPGA and application