FPGA IMPLEMENTATION OF INDIRECT VECTOR CONTROL OF AN INDUCTION MACHINE

A Project Report

submitted by

M MOHAMED ZUBAIR

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
&
MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2013

CERTIFICATE

This is to certify that the project work titled **FPGA IMPLEMENTATION OF INDIRECT VECTOR CONTROL OF AN INDUCTION MACHINE**, submitted by **M MOHAMED ZUBAIR**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology** and **Master of Technology**, is a bona fide record of the project work done by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Krishna Vasudevan
Project Guide
Professor
Dept. of Electrical Engineering
IIT-Madras

Place: Chennai - 600 036

Date: 26th June 2013

ACKNOWLEDGEMENTS

I wish to express my gratitude to Dr. Krishna Vasudevan, Professor, Electrical Engineering Department for his invaluable guidance, during the course of the project. He has been a great mentor and teacher for me. I learnt a lot from the challenging work he gave me. By listening to his classes, I developed an interest in Power Electronics and Motors.

I would also like to thank Dr. Kamalesh Hatua, Assistant Professor, Electrical Engineering Department for guiding and giving valuable inputs for the completion of my project, during the last few months. I gained much knowledge from the discussions I had with them.

I would like to thank my lab-mates, from my batch and, the seniors for helping me out, when I was stuck with the hardware work.

I thank the Institute for giving me an ideal environment to nurture my interest in science and technology. I was fortunate to have the company of my good friends, without whom it would not have been possible to spend such an enjoyable and memorable five years at IIT.

Finally, I am indebted to my mother, father and grandparents for their support, encouragement and guidance throughout. I thank my brother for making my life colourful.

ABSTRACT

KEYWORDS: Vector control, FPGA, Simulink

AC motor drives are becoming increasingly ubiquitous, owing to advancements in technology. Their applications are in locomotives, wind power, rolling mills and so on. This is mostly due to advanced control schemes for the drive, and they are getting more feasible nowadays. Indirect rotor flux oriented control is one such scheme, wherein the angle and magnitude of the rotor flux phasor is controlled to vary the torque. This gives a dynamic behaviour like the Separately Excited DC motor. In this project, an attempt is made to implement this methodology in hardware to control a 1 HP 3-phase induction motor. Simulation of the system in Simulink is done to analyze the functioning and performance. The theory is validated by the results, and provides a structure for the forward work in hardware.

Of the current digital platforms available, the Field Programmable Gate Array (FPGA) is selected for the task. It can easily execute the complex signal processing, owing to its parallel architecture. National Instrument's CompactRIO consists of a Xilinx Virtex-5 FPGA and data-acquisition units. This is an ideal candidate as it meets the requirements of taking analog feedback signals. The circuit design is modularized, in that the functional units are individually built and tested. Finally, the consolidated circuit is configured on the FPGA. The project covers the design of the Sine wave generation, SPWM scheme, Speed detection and current feedback processing. The experiments conducted on the drive, at each stage are discussed and the results, interpreted.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Field Oriented Control

There are various control schemes for induction motor drives. Scalar control techniques involve the control of magnitudes of the variables such as voltage, current, frequency and slip. The control hardware for these schemes is minimum, but the response is sluggish. This is due to the inherent coupling in the induction machine. The torque and flux are functions of voltage, current and frequency. In comparison, the separately excited DC motor has only two parameters, namely the armature current i_a and field current i_f ($\propto \psi_f$). The magnetic field due to these two currents are perpendicular to each other. The torque expression is:

$$T_e = K_t i_a i_f \tag{1.1}$$

The flux and torque are controlled independently by i_f and i_a, respectively using two control loops. Hence, the dynamic response is faster. Vector or field oriented control replicates this behaviour in an induction machine. The voltage, current and flux equations are written in a synchronously rotating reference frame, and with a particular alignment (referred to as the $d^e - q^e$ frame). Then, the stator current space vectors appear as in the following figure:

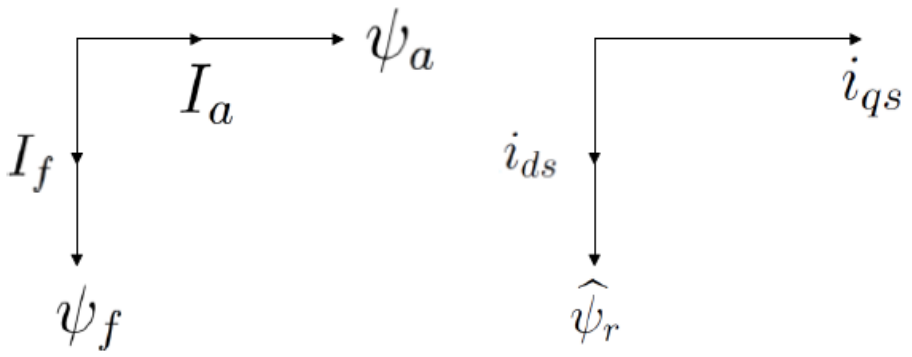


Figure 1.1: Current and flux space vectors in DC motor and vector controlled IM

The torque expression reduces to:

$$T_e = K_t' i_{ds} i_{qs} \tag{1.2}$$

i_{ds} and i_{qs} are the components of the stator current on the d^e and q^e axes, respectively. The analogy with the DC machine can be easily seen. The flux linking the rotor ψ_r can be controlled using i_{ds} and maintained constant. The torque can be varied independently using i_{qs}. The advantages of vector control are:

- Decoupled control of flux and torque
- Better dynamic response
- Full torque capability at low speeds
- Four quadrant operation

The d^e axis can be aligned along rotor flux, stator flux or air gap flux. The rotor oriented method gives a natural decoupling control and is used in this project. Depending on the manner in which the ψ_r angle, θ_e is calculated, there are two methods:

1. **Direct vector control**: Stator voltages and currents are taken as feedback. The flux vector is estimated using Voltage model or current model.
2. **Indirect vector control**: Speed (ω_r) feedback is taken. Stator currents are sensed and the slip speed (ω_{sl}) is found in a feed-forward manner. The synchronous speed is the sum of (ω_r) and (ω_{sl}). This method is followed in this project as, in some cases measuring voltage may not be feasible (e.g. if armature lead is long)

1.2 Signal Processing Component

Field oriented control involves complex mathematical calculations, as will be seen in the next chapter. The feedback signal processing, feed-forward estimation and axes transformation place a high demand on the control hardware. An important requirement is that all this processing, should not impact the dynamics of the drive performance, that is the processing is ideally expected to be continuous. As far as the machine is concerned, the corresponding switching voltage is asserted on its terminals, in a continuous manner. The electromechanical response of the AC drive such as motor currents and speed, depend only on the motor and inverter parameters. Any transition in the inverter switch configuration (S_1 to S_6)occurs at the switching frequency (f_{sw})(usually in the range of $5kHz$ to $20kHz$). Hence, the vector control algorithm should continuously execute at a frequency higher than f_{sw} and update the states of S_1 to S_6, every $1/f_{sw}\ secs$.

Usually, the hardware for motor drives are implemented using analog circuitry or Digital Signal Processors. Analog circuits are extensively used for DC drives, but they cannot be used to realize complex mathematical functions, owing to complexity and unreliability. Hence, for decades, DC motors were used for precision control applications, despite the induction motor having the ruggedness for industrial applications. With the advent of modern DSPs and microcomputers, AC drives are being increasingly deployed in areas such as factory process flow, electric vehicles, deep sea mining, etc. The cost factor and time to market have improved.

1.3 FPGA-An Overview

Field programmable gate array or FPGA is a reconfigurable digital circuit, fabricated on Silicon. It consists of a matrix of 'Configurable logic blocks' (CLBs), with programmable interconnects. Thus, the FPGA is generic and can be customized by downloading a binary bitstream, which configures these interconnects. The programming is done using VHDL or Verilog Hardware Description language. The fundamental elements on an FPGA chip are as follows:

1. *Slice*: A CLB consists of 2-4 slices. The slice constitutes the logic part and consists of the following digital circuits:
   - Look-up table (LUT): This stores a fixed output (1 bit) for a combination of inputs. All combinational logic is implemented internally on LUTs. For example, a 4-input NAND gate is realized by storing its truth table in a 4-input LUT.
   - Flip-flops: This is the sequential element. It stores 1-bit and is triggered by the clock.
   - Multiplexer (MUX): It chooses between several inputs, depending on control bit and gives the output.
2. *Interconnect resources*
3. *Registers*: They are a bunch of flip-flops, with an input, output, reset and clock. They serve several functions such as storing values between iterations, etc.
4. *BlockRAM*: They are blocks of memory, embedded in the chip and store large amounts of data. For example, a non-linear waveform can be stored in it.
5. *Clock*: The FPGA clock is a single bit, which toggles at a very high frequency ($2\ to\ 5MHz$). Clocks running at a lower frequency can be derived from this master clock.
6. *Input/Output blocks*: They are at the periphery of the chip and used to communicate with the external environment.
7. *Multipliers*: This is an added resource. Multiplication using LUTs can be quite resource expensive. Hence, some FPGAs which are intended to be used for DSP applications, are provided with dedicated multipliers.

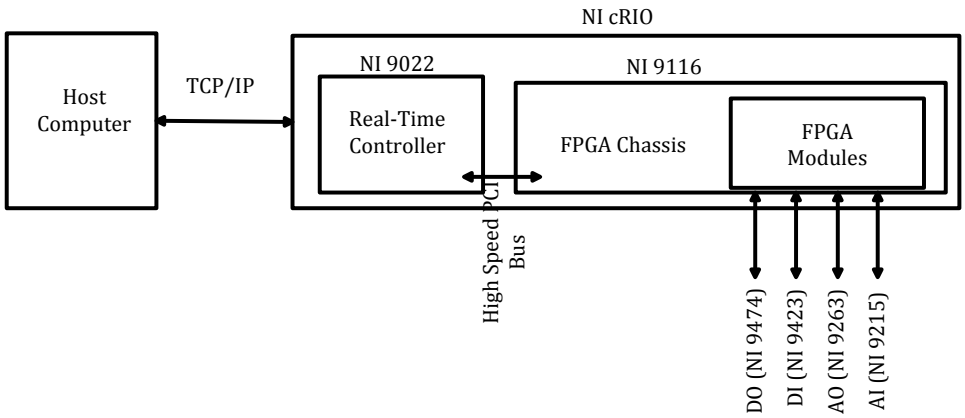
The suitability of FPGA for Vector control application can be appreciated by looking at the following features:

1. *Reconfigurability*: In this respect, it differs from Application Specific IC (ASIC), whose functionality is frozen during manufacturing. The same FPGA can be used for motors with different parameters or for a different control scheme.
2. *Parallelism*:In Digital Signal Processors (DSPs), a program for computing the control algorithm is executed in Arithmetic & Logic Unit (which has limited resources). Results at each stage are available in a sequential manner. Hence,one iteration of a complex program takes a long time to execute. The architecture of FPGA allows all the loops to execute in parallel. For example, the axes transformation, slip estimation and PI controllers can all execute simultaneously and the next switching sequence will be calculated within $T_{sw}\ secs$.
3. *High Speed & Resource sharing*: If the circuit is fast enough, the same hardware can be shared by multiple operations. The 4 PI compensators execute only once in $T_{sw}\ secs$. Hence, the same piece of circuit can be re-used 4 times with varying values of k_i and k_p. As an extension, one FPGA can run two AC drives, in sequence, if time permits.

4. *Cost-effective* : Recent advances in VLSI have reduced the cost of FPGA chip, whilst increasing the logic density (CLBs per unit area). The number of inputs and outputs far exceed that of DSPs.

1.4 NI cRIO - An Overview

National Instrument's CompactRIO is a reconfigurable real-time signal processing engine with data acquisition capabilities. It is extremely rugged and is used in industrial control systems. The cRIO is equipped with a Real-time controller, FPGA and ethernet port for interfacing with a host computer. At each level, a program loop executes at different clock rates. The cRIO can be programmed, graphically using NI LabView. The graphic design is called a Virtual Instrument or VI. The whole system is depicted in the following diagram and explained below.



crio.pdf

Figure 1.2: cRIO modules

1. *Host Computer*: This is the highest level and used for programming the two lower blocks. It provides a user interface for viewing the data acquisition. But, real-time programs do not run at this level.

2. *Real time controller*: It consists of a real-time processor, Random Access Memory (DRAM) and a non-volatile memory. It is used for running applications such as control, data-logging and communicating with the FPGA (via PCI bus)and the host (via TCP/IP). It is programmed using LabView Real-Time Module.

3. *FPGA Chassis*: This is lowest level in the system, which directly interfaces with the signals being acquired. The process runs at a very high sampling rate and hence, used for time-critical programs. It is programmed using LabView FPGA

module. There is no need for VHDL coding. The compiler on the host generates the bit-stream to be dumped on the FPGA.

4. *Signal Input/Output Modules*: C series modules are used for acquiring data such as temperature, current, pressure, etc. In this project, there is requirement only for voltage sensing and output. Hence, Analog input (AI) (consists of ADC), analog output (AO) (consists of DAC), digital input (DI) and digital output (DO)modules are mounted on the chassis.

The names and description of the cRIO modules used in this project are shown in Table 1.1 . The blocks of the vector control algorithm are completely,implemented on

Table 1.1: cRIO components

Module	Description	Slot no.
NI 9022	Real-Time Controller:533 MHz Freescale processor,256 MB DRAM, 2 GB Storage	-
NI 9116	8-Slot chassis, Xilinx Virtex-5 LX85 FPGA	-
NI 9215	4-Channel, $\pm$10 V, 16-Bit Simultaneous Analog Input Module	1-3
NI 9263	4-Channel, $\pm$10 V, 16-Bit Analog Voltage Output Module	4
NI 9423	8-Channel Sinking Digital Input Modules	5,7
NI 9474	8-Channel Current sourcing, Digital Output Modules	6

the FPGA, using the FPGA interface mode. The real-time controller plays no role in the process, except for facilitating communication with the user interface on the host computer. Hence, it is important to evaluate the features of this FPGA. The Xilinx Virtex-5 LX85 is a very powerful FPGA with advanced architecture and additional capabilities. It employs a clock of a very high frequency of $40\ MHz$, which is more than sufficient for this project. 6-input LUTs allow more complex logic to be implemented in the same slice, compared to 4-input LUTs of other FPGAs. DSP48E slice is a 25 bit X 18 bit multiplier (again an extension over the previous FPGA family). This offloads the task of the normal logic elements and performs most of the DSP functions, such as Multiply-and-Accumulate, wide-bus comparison, multiplexing, adder chains and barrel shifting. The signals being operated on are either Fixed Point or Integer data-type.The exact number of resources available are in Table 5.1. It is important to keep track of these numbers, for optimizing as the digital circuitry is designed. The correlation between the logic implemented and resources used can be inferred.

1.5 Scope of Project

The goal of the project is to evaluate the feasibility of implementing the vector control algorithm on an FPGA. The project starts with a simulation study of Vector controlled drive. This gives a preliminary view of the hardware units to be built. The design process is modularized. Each unit, such as PWM, axes transformation, etc. are built and tested separately. The project extends till running the motor using Constant Volts/Hertz Scheme, acquiring feedback signals and processing them.

1.6 Organization of Report

The theory and derivation behind the vector control scheme is discussed in Chapter 2. The motor is modelled in the $d^e - q^e$ reference frame.

The computer simulation study of the drive is done in Chapter 3. The results obtained speed waveforms are analyzed from a theoretical perspective.

Chapter 4 lists and describes in detail the work done on the FPGA and the experiments conducted on the motor. The design considerations for the digital circuitry are discussed. Analog interface for the feedback signals (speed and current) are presented. The Oscilloscope plots of Motor voltage, current and intermediate FPGA signals are analyzed.

CHAPTER 2

INDIRECT VECTOR CONTROL-THEORY

2.1 Machine model

The following notations are used in the derivation and throughout the report:

Table 2.1: Machine parameters and variables

Notation	Quantity
$d^e - q^e$	Synchronously rotating frame
$d^s - q^s$	Stationary reference frame
i^s_{ds}	d^s - axis stator current
i^s_{qs}	q^s - axis stator current
i_{ds}	d^e - axis stator current
i_{qs}	q^e - axis stator current
i_{dr}	d^e - axis rotor current
i_{qr}	q^e - axis rotor current
i_a, i_b, i_c	Stator phase currents
v_{qr}	q^s - axis rotor voltage
v_{dr}	d^s - axis rotor voltage
ψ_{dr}	d^s - axis Rotor flux linkage
ψ_{qr}	q^s - axis Rotor flux linkage
$\widehat{\psi_r}$	Total Rotor flux linkage
$\omega_r, \omega_{sl}, \omega_e$	Rotor(electrical)speed, slip frequency, synchronous speed
θ_r, θ_{sl}	Rotor(electrical)angle, slip angle
θ_e	Angle of $d_e - q_e$ frame
L_r	Rotor inductance
L_s	Stator inductance
L_m	Magnetizing inductance
R_r	Rotor resistance
T_e	Electrical torque
T_l	Load torque
ω_m	Rotor mechanical speed
P	No.of poles
J	Moment of inertia

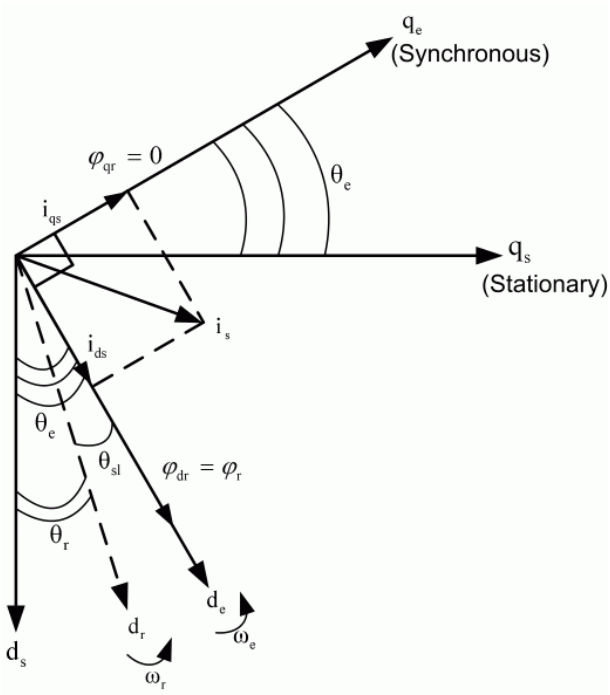


Figure 2.1: Phasor diagram of indirect vector control

The above figure shows the current and flux space vectors, with respect to the three frames, namely, $d_r - q_r$, $d_s - q_s$ and $d_e - q_e$. The rotor position is aligned along d_r axis. The stator current phasor rotates at w_e. Thus, an emf of frequency, w_{sl} is induced in the rotor bars, which in turn produces a current at w_{sl}. The magnetizing flux, stator and rotor flux linkage are all rotating at w_e. The $d_e - q_e$ frame has to chosen such that rotor flux,$\widehat{\psi_r}$ is along d_e axis. This ensures the decoupling of the flux and torque components of the stator current.

$$\theta_e = \int \omega_e dt = \int (\omega_r + \omega_{sl}) dt = \theta_r + \theta_{sl} \tag{2.1}$$

Since this is a squirrel cage motor, the rotor voltages are zero. The rotor circuit

equations in the $d_e - q_e$ frame are:

$$\frac{d\psi_{dr}}{dt} + R_r i_{dr} - (\omega_e - \omega_r)\psi_{qr} = v_{dr} = 0 \tag{2.2}$$

$$\frac{d\psi_{qr}}{dt} + R_r i_{qr} - (\omega_e - \omega_r)\psi_{dr} = v_{qr} = 0 \tag{2.3}$$

$$\psi_{dr} = L_r i_{dr} + L_m i_{ds} \tag{2.4}$$

$$\psi_{qr} = L_r i_{qr} + L_m i_{qs} \tag{2.5}$$

To eliminate the rotor currents and introduce the stator currents, the rotor flux linkage equations are substituted in the above two equations, to get

$$\frac{d\psi_{dr}}{dt} + \frac{R_r}{L_r}\psi_{dr} - \frac{L_m}{L_r}R_r i_{ds} - w_{sl}\psi_{qr} = 0 \tag{2.6}$$

$$\frac{d\psi_{qr}}{dt} + \frac{R_r}{L_r}\psi_{qr} - \frac{L_m}{L_r}R_r i_{qs} - w_{sl}\psi_{dr} = 0 \tag{2.7}$$

If decoupling is achieved, $\psi_{qr} = 0$. Then substituting in the above equations,

$$\frac{L_r}{R_r}\frac{d\widehat{\psi}_r}{dt} + \widehat{\psi}_r = L_m i_{ds} \tag{2.8}$$

$$w_{sl} = \frac{L_m R_r}{\widehat{\psi}_r L_r} i_{qs} \tag{2.9}$$

It can be seen that the rotor flux is related to i_{ds} by first order dynamics. The expression for electrical torque in this frame is:

$$T_e = \frac{3}{2}\frac{P}{2}\frac{L_m}{L_r}\widehat{\psi}_r \times i_{qs} \tag{2.10}$$

The mechanical side equation is:

$$T_e - T_l = \frac{2}{P} J \frac{dw_r}{dt} \tag{2.11}$$

The preceding four equations completely describe the electrical and mechanical behaviour of the induction motor. The control loop is built using these equations. The $a-b-c$ to $d_e - q_e$ axes transformation will be discussed in the implementation section.

2.2 Control scheme

The implementation of the Indirect vector control with speed command is explained below.

1. A speed encoder is used to measure the rotor mechanical speed. Two of the phase currents are sensed and provided as feedback.

2. The stator currents are transformed from abc to dq (synchronously rotating) reference frame. This block needs $\cos\theta_e$ and $\sin\theta_e$.

3. The difference between the reference rotor speed, ω_r^* and actual speed,ω_r is taken and given to a PI (Proportional-Integral) controller. This will give T_e^*, which is in turn proportional to i_{qs}^*, as shown in equation 2.10.

4. The reference rotor flux, ψ_r^* is maintained constant. The value of i_{ds}^* is calculated using the transfer function from equation 2.8.

5. The desired slip frequency, ω_{sl}^* is generated using the value of i_{qs}^* and equation 2.9, in a feed-forward manner. The actual value of $\widehat{\psi_r}$ (estimated using i_{ds})is used.

6. ω_r and ω_{sl}^* are added to get the synchronous frequency,ω_e, which is integrated to obtain flux field angle θ_e.

7. As Voltage Source Inverter(VSI) is being used, the switching pulses should be generated using stator voltages. Reference stator currents, i_a^*, i_b^*, i_c^* cannot be given as commands. Hence,the errors in the dq frame stator currents is fed into PI compensators, which are expected to give reference stator voltages, v_{ds}^* and v_{qs}^*. But, there is a cross-coupling term as shown in the following stator circuit equations in the $d_e - q_e$ reference frame:

$$v_{ds} = R_s i_{ds} + \frac{d}{dt}\psi_{ds} - \omega_e \psi_{qs} \tag{2.12}$$

$$v_{qs} = R_s i_{qs} + \frac{d}{dt}\psi_{qs} + \omega_e \psi_{ds} \tag{2.13}$$

8. The stator flux linkages, $\psi_d s$ and $\psi_q s$ are estimated using the stator voltages. The decoupling terms, $-\omega_e\psi_{qs}$ and $\omega_e\psi_{ds}$ are added to to get the voltage command values, v_{ds}^* and v_{qs}^*. They are transformed to the abc frame.

9. v_a^*, v_b^*, v_c^* are given as inputs for the Sine PWM block to get the switching pulses, S_1 through S_6 for the inverter.

CHAPTER 3

SIMULATION AND RESULTS

3.1 Implementation

A simulation of the system being designed is carried out to evaluate the system, fix control variables (PI) and anticipate any problems, while building the hardware. The tool used is Simulink. A variable step solver, ode23tb is used for a continuous simulation. Other parameters are maximum step size of $5 \times 10^{-5} sec$ and relative tolerance of 10^{-4}. The implementation is done as follows:

1. A 1 HP induction machine with the resistance and inductance values found in Section 4.1 is used. Motor speed and stator currents are taken as feedback.

2. dq to abc transform and abc to dq transform units are built, with input as the synchronous angle, θ_e, which in tur is obtained from rotor speed and estimated slip frequency.

3. The reference rotor flux is a constant, 0.92 Wb. The actual value of the flux is obtained with i^e_{ds} by an observer (Eqn: 2.8). There is a PI loop to reduce this error, which gives the reference value of i^*_{ds}. $k_I = 10, k_P = 1$.

4. The difference between set speed and actual speed is compensated with PI, and gives i^*_{qs}, as output. $k_I = 10, k_P = .02, k_I = 1$This is used in slip estimation (2.9). The actual value of ψ_r is used. As it is zero initially, a reference value of 0.1 is given then.

5. Rated current is 1.8 A (rms). Hence, limit on the d-q frame currents is taken as 1.5 times the peak. Thus a limit of (-4, +4) A is put on the outputs of the two PI units. This ensures that a heavy current, which may damage the motor windings does not flow. The values of k_P and k_I are tuned by optimizing on the overshoot and settling time of the speed response.

6. The reference d-q currents give i^*_a, i^*_b, i^*_c. These currents are applied to the motor terminals using Controlled Current Sources.

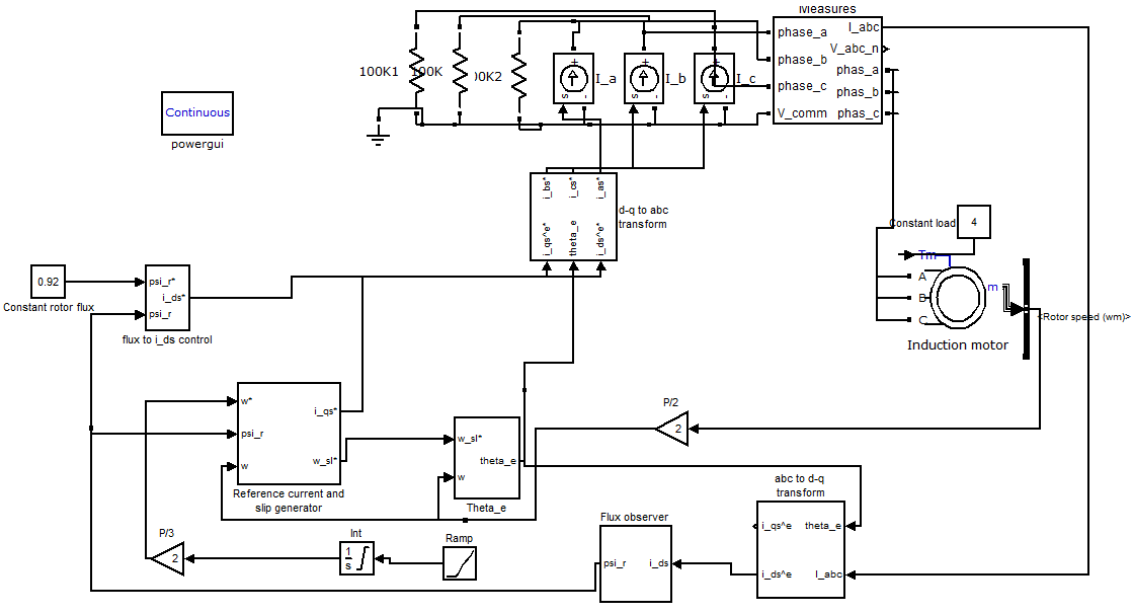


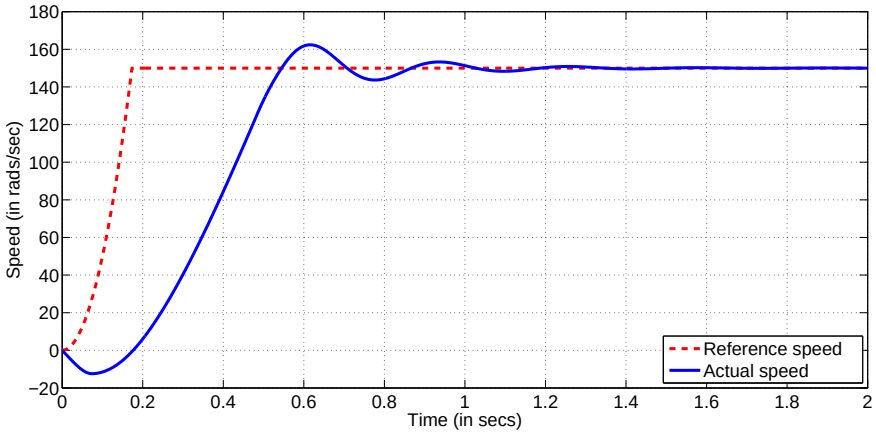
Figure 3.1: Simulink Block diagram

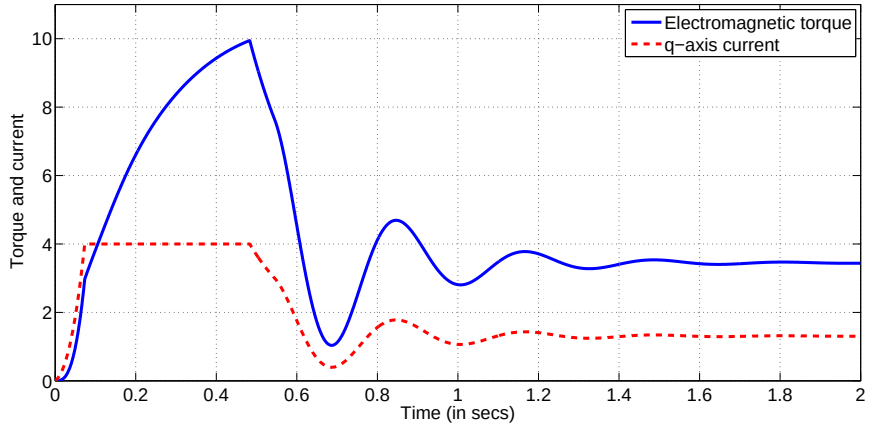
3.2 Discussion of results

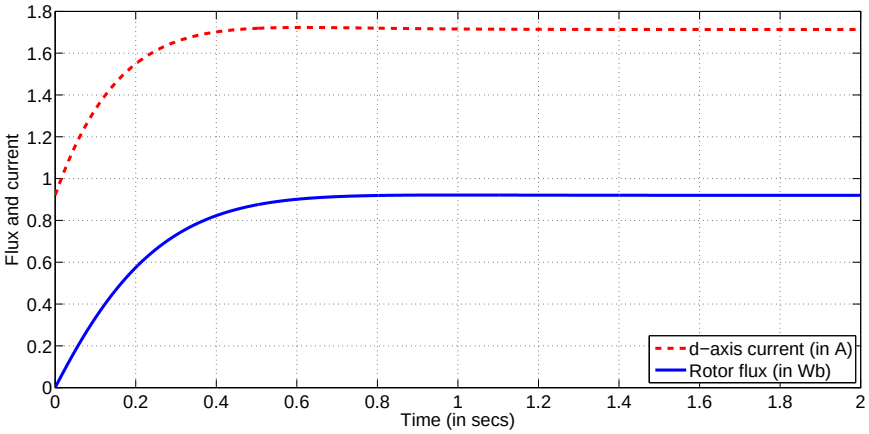
Initially, the drive is tested by increasing the ω_{ref} slowly, till it reaches 145 rad/s. The motor is on a load of 5 Nm .The plots are given in Fig. 3.1. The flux is established by its control loop, with no overshoot. i^e_{ds} is proportional to the flux once it reaches steady state. The torque and i^e_{qs} curves also follow each other closely, except for initial 0.8 secs, when the rated flux was not yet developed in the air gap. After that decoupling is achieved as seen in the flux and torque plots. The changes in the i_{dq} plots can be linked to the PI values. The abrupt change in i^*_d is due to mismatch in flux. The i^*_q is saturated at its maximum value, during the first cycle, due to the strong PI loop. Then speed increases linearly and hits ω_{ref}, when T^*_e responds immediately to dampen the speed oscillations within 3 cycles. The overshoot is also within 10%. The RYB phase currents start off at a low frequency, as commanded ω_e is small (motor starting). The amplitude is large till 0.8 secs as the flux has to built up and there is an initial torque to be overcome. Once i^*_{ds} stabilizes, it can be observed that the amplitude of i_{ph} is merely dependent on the torque component, i^*_{qs}.

In Fig. 3.2, the motor is on constant load of 5 Nm, but the speed reference keeps changing (stepped waveform). The motor initially rotates in the opposite direction, due to external load. The time for which ω_r is in linear region depends on the abrupt change in reference. After the crossover, the speed and motor mechanical loop is fast enough (note steep change in T_e at 3.3 secs) to ensure not much overshoot occurs. All oscillations die down in 4-5 cycles. But greater the step in ω_{ref}, longer is the settling time. At 3 sec, there is braking on the motor. i_{qs} is observed to remain unperturbed throughout(system is decoupled). So, in the dq frame, flux phasor is stationary, while $\overline{I_s}$ sweeps past it in the other direction to generate negative torque.

The speed reference is constant, but load torque is varied in Figures 3.3. The speed responds to this change first, then the electromagnetic torque compensates this error. At 3 sec, the speed overshoots as , the external load is aiding in its direction. This serves as the signal for T_e to bring the speed back to ω_{ref}. After that instant, the motor is being braked by the drive. In steady state , T_e is offset from the load, T_l, by a small amount. This is due to the motor friction, $B\omega_m$. The flux is constant, thus giving a DC motor like behaviour.







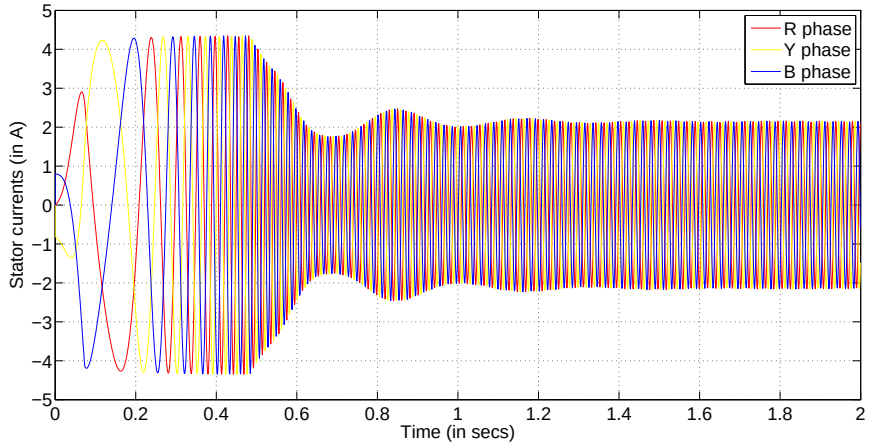


Table 3.1: Gradual start characteristics

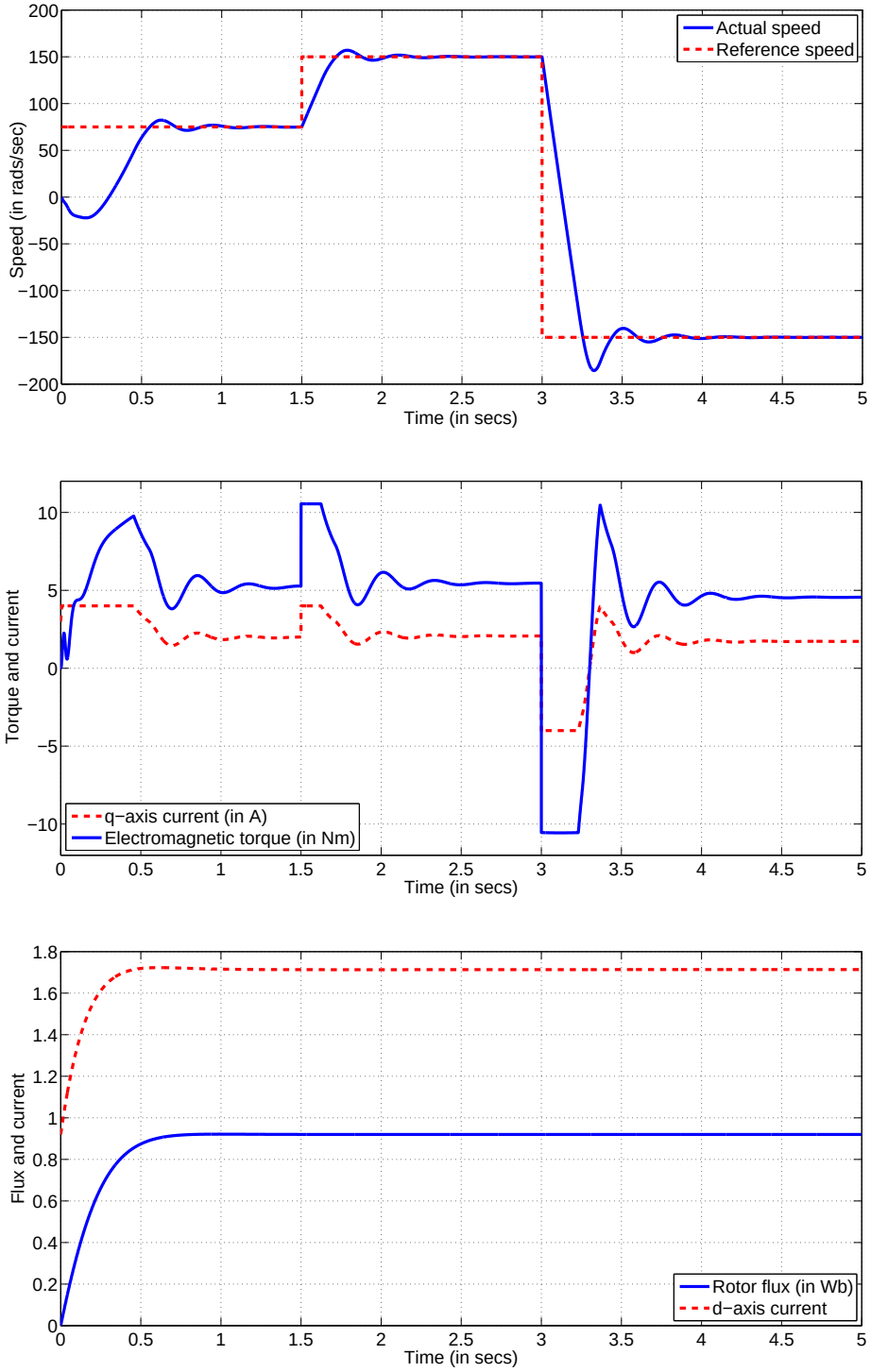


Table 3.2: Constant load and Varying speed reference

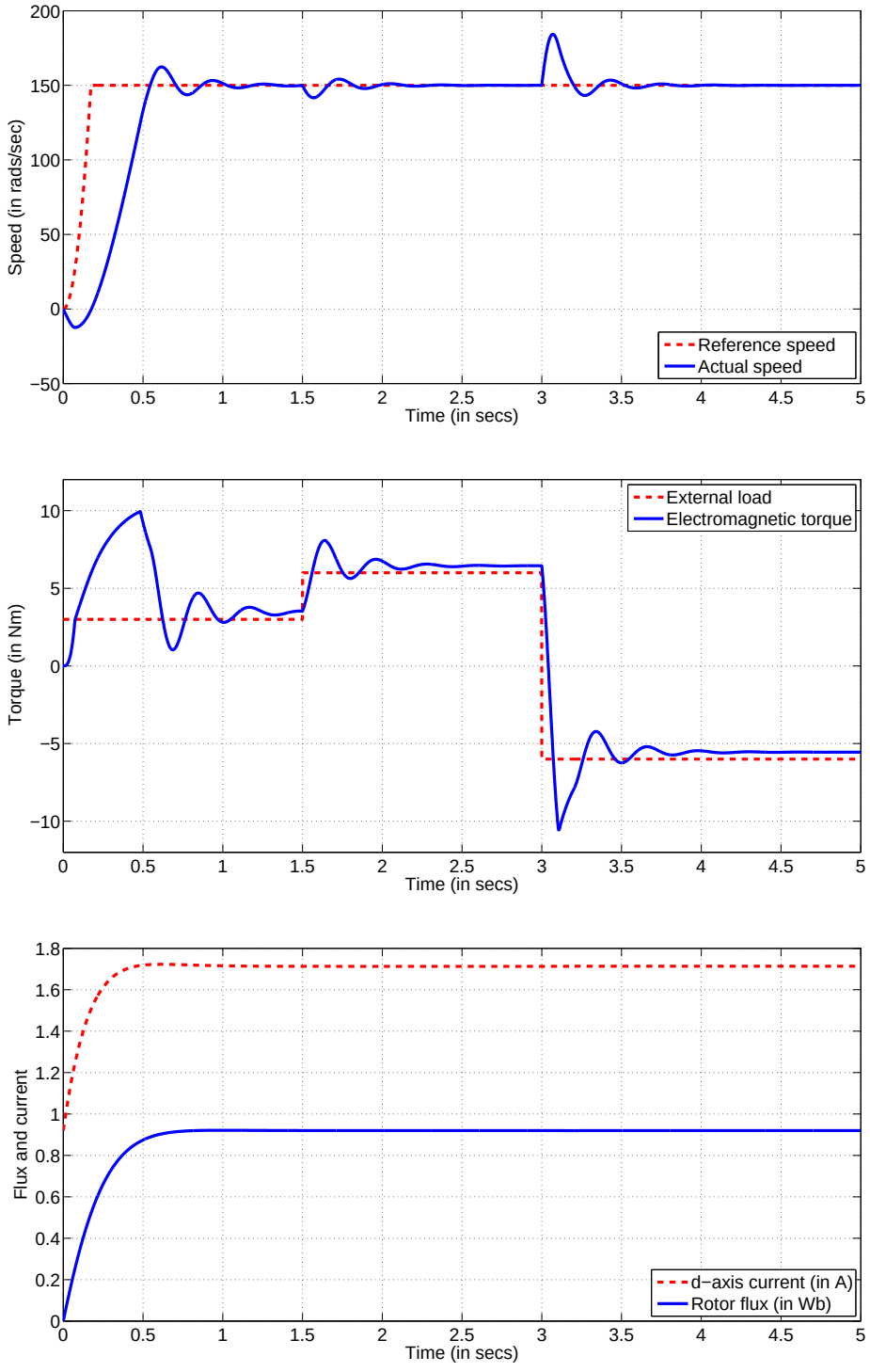


Table 3.3: Constant speed reference and Varying load

CHAPTER 4

HARDWARE IMPLEMENTATION AND RESULTS

4.1 Experiment for finding motor parameters

A 4-pole, $50Hz$ squirrel cage induction motor, with the windings connected in star configuration is used. The rotor shaft is fitted with a drum and belt arrangement.A constant load torque can be applied by tightening the belt. The nameplate details of the motor are:

Table 4.1: Rated values

Quantity	Rating
Output Power	$0.75\ kW$
Line voltage	$415V$
Phase current	$1.8A$
Power factor,$cos(\phi)$	0.81
Rotor mechanical speed	$1395\ rpm$

The control scheme depends a lot on the motor parameters such as rotor resistance, moment of inertia, etc. Hence, these values have to be calculated prior to the design. The short circuit (blocked rotor) test and open circuit (no load) test were conducted to derive this data. The line-to-line voltage, V_{ab} and phase current, I_a, are captured on the oscilloscope using probes. Their magnitude and phase lag/lead is noted. Only rms values are used in the calculations. The impedances are all referenced to the stator. The following per-phase equivalent circuit model of an Induction machine is used.

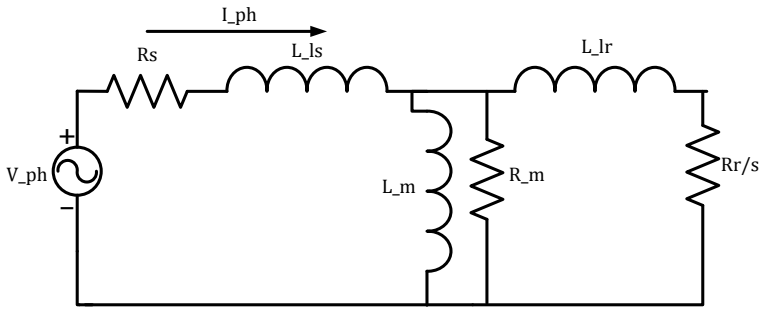


Figure 4.1: Equivalent circuit of induction motor

4.1.1 No load test

The belt is loosened so that there is zero load torque acting on the motor. There are only frictional losses. A high voltage(near the rated value), is given to the motor, via the auto-transformer. So,the motor rotates at close to synchronous frequency, and slip s is approximately zero.In figure 4.1, the rotor branch consisting of leakage inductance, L_{lr} and equivalent resistance, R_r/s presents a huge impedance, and is assumed to be open. Now, the circuit consists of a series combination of stator impedance, $R_s + j\omega L_{ls}$ and magnetizing branch, $L_m||R_m$. The stator impedance is a small value, compared to L_m and R_m, and is neglected. The net impedance of the circuit is $L_m||R_m$. The plots observed on the oscilloscope are:

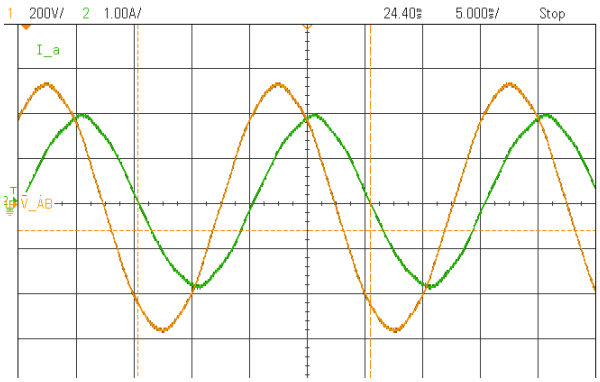
Measured values: $V_{ab}(rms) = 380.61V, I_a(rms) = 1.31A, Phase - lag = 54°$

$$\overline{V_a} = \frac{\overline{V_{ab}}}{\sqrt{3}} \angle 30° = 219.75V \angle 0° \tag{4.1}$$

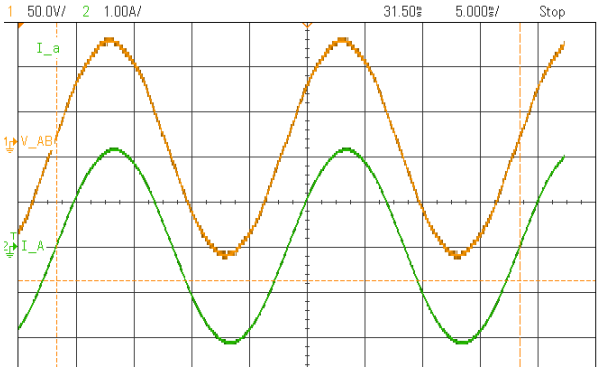
$$Z = \frac{V_a}{I_a} = 167.75 \angle 84° \tag{4.2}$$

$$\Rightarrow \frac{\omega LR}{R + j\omega L} = 167.75 \angle -6° \tag{4.3}$$

Solving, we get: $R_m = 1604.87\ \Omega$ and $L_m = 536.9\ mH$



Open circuit test
Scale - x:5 ms/div, V:200 V/div, I:1 A/div



Short circuit test
Scale - x:5 ms/div, V:50 V/div, I:1 A/div

4.1.2 Blocked rotor test

The shaft of the motor is held at rest by tightening the belt. Slowly, the voltage applied is increased till rated current flows. As motor is at rest, slip = 1. The magnetizing and rotor branches are in parallel. In comparison, $R_r + j\omega L_{lr}$ is much smaller than $R_m + j\omega L_m$. Hence, not much current flows through the latter. The circuit in Figure 4.1 reduces to series combination of R_s, L_{ls}, L_{lr} and R_r. The oscilloscope plots are:

Measured values: $V_{ab}(rms) = 81.41V, I_a(rms) = 1.518A, Phase\ lag = 10.8°$

$$\overline{V_a} = 47\angle 0° \tag{4.4}$$

$$\overline{I_a} = 1.518 \angle -40.8° \tag{4.5}$$

$$\Rightarrow R + j\omega L = 30.97\ \angle 40.8° \tag{4.6}$$

Solving, we get: $R_s + R_r = 17.75\ \Omega$ and $L_{lr} + L_{ls} + = 48.77\ mH$.

The rotor and stator leakage inductances are assumed to be equal.

$\Rightarrow L_{lr} = L_{ls} = 24.38\ mH$

The star connection of the the three motor windings is removed. A DC voltage is applied across the two terminals of one of the windings. The current is measured. By this method, R_s is found to be $13.5\ \Omega$. So, $R_r = 4.25\ \Omega$.

4.2 Experimental Setup

The power-side circuit, used is same throughout the project. A schematic is provided in the following figure.

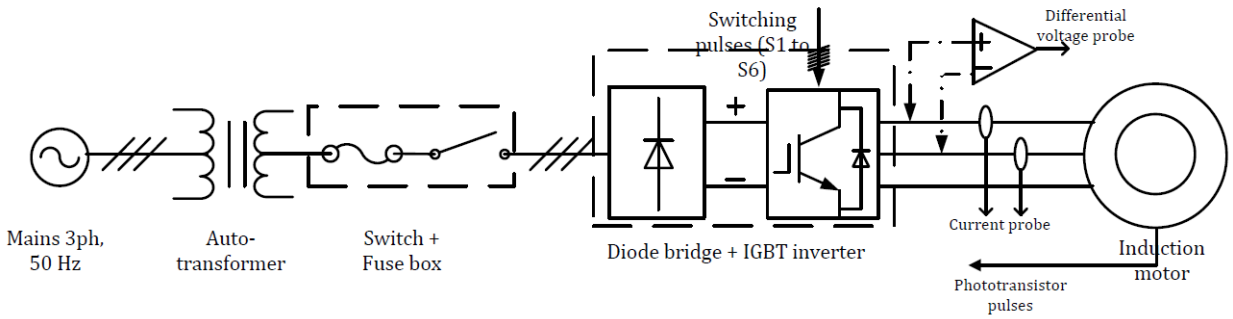


Figure 4.2: Work-bench

The primary of an auto-transformer is connected to 230 V, 3-phase AC mains. The secondary is connected to the Semikron power electronics kit through a single-pole-single-throw switch and fuse box. The Semikron kit consists of a 3 ϕ diode bridge rectifier, which can give upto 600 V DC. This is the input for a 3 ϕ IGBT based controlled inverter. The output AC voltages, v_{an}, v_{bn} and v_{cn} are asserted on the motor terminals (v_n is the $-ve$ of the DC bus).

The inverter switches are controlled by a gate driver card, which is driven using 6 digital signals of levels 15 V/ 0 V. The driver ensures an interlocking time of 3 μs between the signal rising edge and IGBT gate rising edge. So, there is essentially an inbuilt dead-time of 3 μs. The DO module, NI 9474 drives the PWM signals. A V_{sup} is given and output signal is between $D0\ 0...5$ and COM terminals. After switching off, the potential on the pin takes some time to decay, due to parasitic capacitance. To minimize this, a pull-down resistance of $47\ \Omega/5\ W$ is put between each of the 6 DO's and COM/GND. Six of the wires in a sheath cable provide connection between the FPGA and inverter driver. A differential voltage probe measures the line-to-line voltage. A current probe measures one of the phase currents. Both are displayed on the oscilloscope.

4.3 Sine PWM module

4.3.1 Scheme

The Pulse-width modulation sequence is calculated, mathematically to emulate saw-tooth - sine wave comparison. This is an alternative to the conventional method of using explicit comparator. Refer to the following figures.

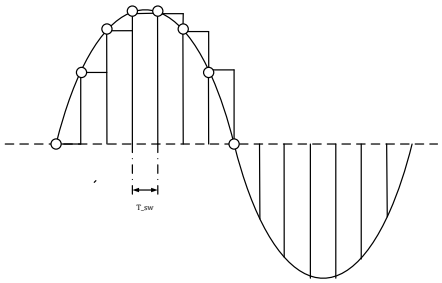


Figure 4.3: Sine-wave : Sample and hold

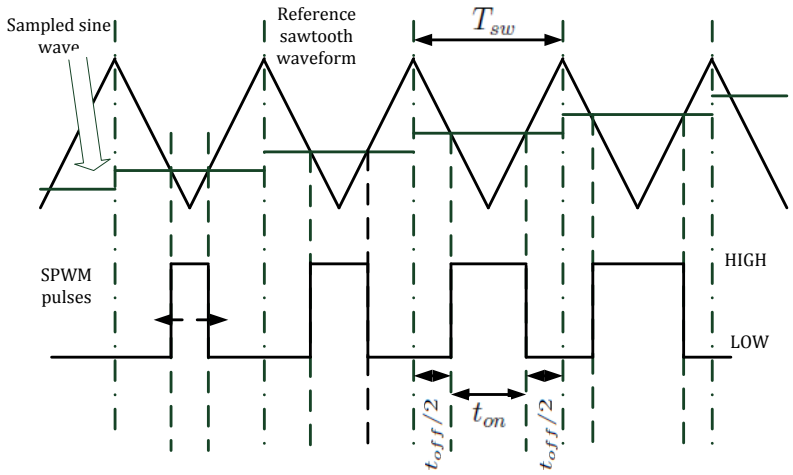


Figure 4.4: PWM

The incoming modulating sine wave of frequency, f_{sig} is sampled at the switching frequency, f_{sw}. A step like function is obtained. Note that,the peak of sawtooth pulses is used as the reference, to start a new FPGA loop. In Figure 4.4, a reference sawtooth waveform at f_{sw} is compared with the relatively slower sampled sine wave (added with

an offset of +1). The PWM pattern, obtained can be deduced using similar triangles. V_m and V_{peak} are the amplitudes of the sine and sawtooth waves, respectively.

$$x[n] = \frac{V_m}{2}[1 + \sin(\omega t)]_{nT_{sw}} \tag{4.7}$$

$$\frac{t_{on}}{T_{sw}} = \frac{x[n]}{V_{peak}} \tag{4.8}$$

$$\Rightarrow d = 0.5(1 + m_a[\sin(\omega t)]_{nT_{sw}}) \tag{4.9}$$

$$t_{on} = dT_{sw}\ and\ t_{off} = (1-d)T_{sw} \tag{4.10}$$

Here, d is the duty ratio and m_a is the amplitude modulation index.

In this PWM, the centre-to-centre of ON time is fixed. Both the rising and falling edges vary, as the OFF time is split into two halves on either side of the ON time. This was used, because, if the rising edges of S_a, S_b, S_c are synchronized, all the six switch gating pulses change state simultaneously.But the IGBTs don't switch at the same time, owing to the current flowing in the inductive load. Hence, there is a lag between the switching of two pole voltages.

$$v_{ab} = v_{an} - v_{bn} \tag{4.11}$$

This results in the line voltage getting a spurious spike and distorting. This is observed in the following plot. Hence, this trial scheme was modified to get the above one.

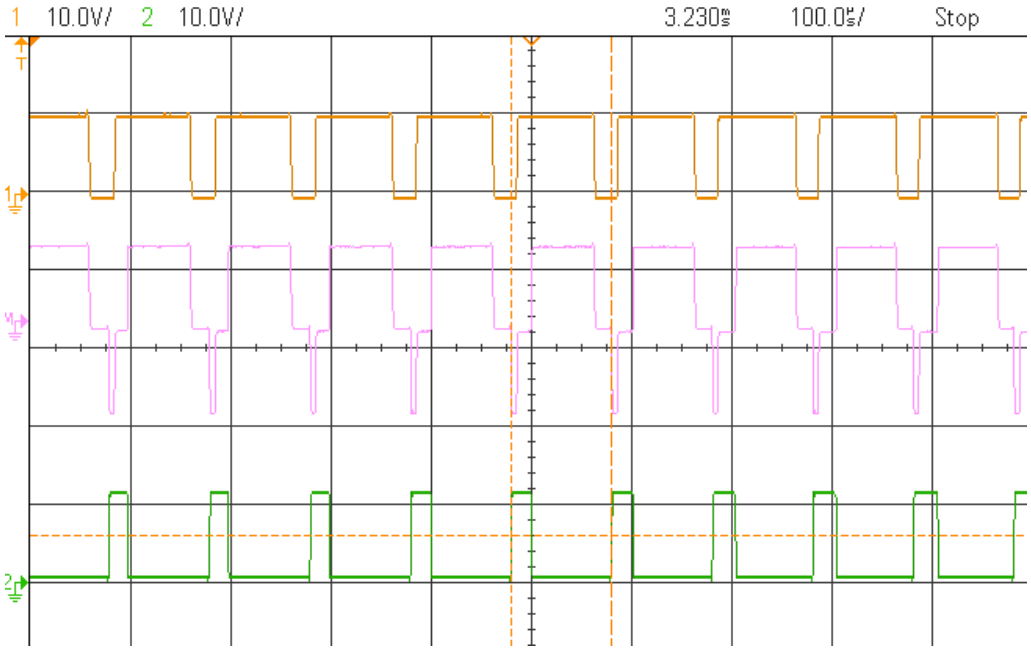


Figure 4.5: M: Spike in line voltage; 1,2 : Pole voltages (Scale - V:10 V/div , x: 100 μs/div)

4.3.2 FPGA block

The input signals are $\frac{T_s}{2}$ and $\frac{T_s}{2} * m_a * \overline{v_a^*[n]}$. The values of t_{off} and t_{on} are calculated and input to the wait timers. The flat frame structure in LabView makes the pieces of code to execute simultaneously. 8-bit μs timers are used, which wait at a particular state of the $DO0$ and $DO1$ pins (for S_a, leg A of inverter). There is provision for dead-time and switching off the PWM (All pins LOW). The loop time is T_s, which is usually 100 μs ($\leftrightarrow$ 10 kHz). Three such VI's are implemented for the 3 phases. They are housed in the external interface loop (Fig 4.8). It's inputs are:$m_a, T_s, \frac{1}{V_{peak}}$, and the command modulating voltage: $v_{an}^*, v_{bn}^*, v_{cn}^*$. V_{peak} is used for scaling the reference within the range: $(-1, 1)$. The multiplication, $k_{pwm} * v^*$ for the 3 values is multiplexed and done in a separate higher frequency (2.5 MHz) loop (Fig. 4.7) .

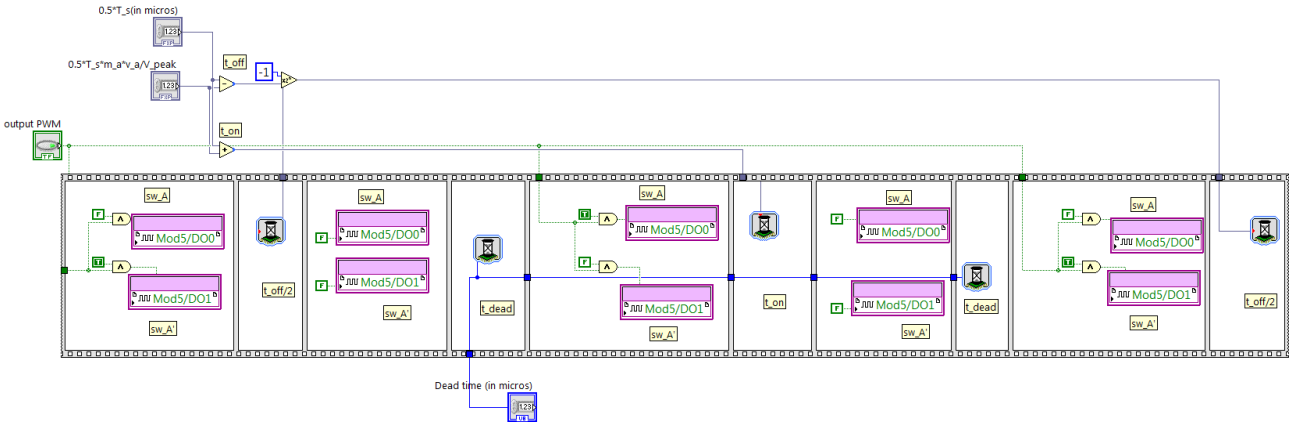


Figure 4.6: PWM module

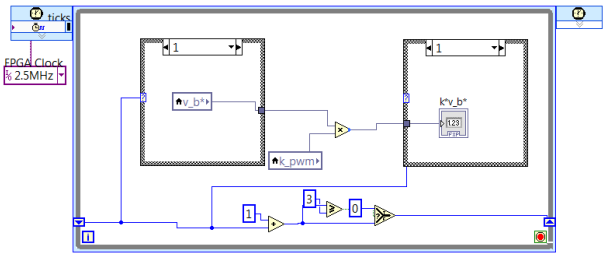


Figure 4.7: Multiplication:Resource sharing

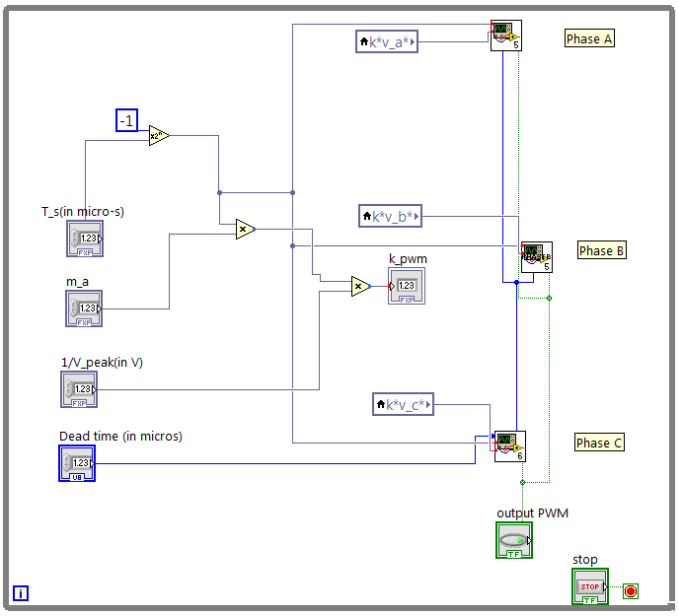


Figure 4.8: 3 phase PWM

4.3.3 Results

The SPWM scheme was first tested by applying to a star connected resistance load box ($50\ \Omega\ to\ 100\ \Omega$). The line voltage was unipolar, oscillating between 0 and $+V_{dc}$ for half of T_{sig}, and) and $-V_{dc}$, for the other half. The phase current was a 6-stepped waveform, a result of the characteristic phase voltage of any star connected load. The FPGA signals for few switching instances are plotted. They are expected to be a replica of the pole voltages. Their difference gives the shape of the line voltage.

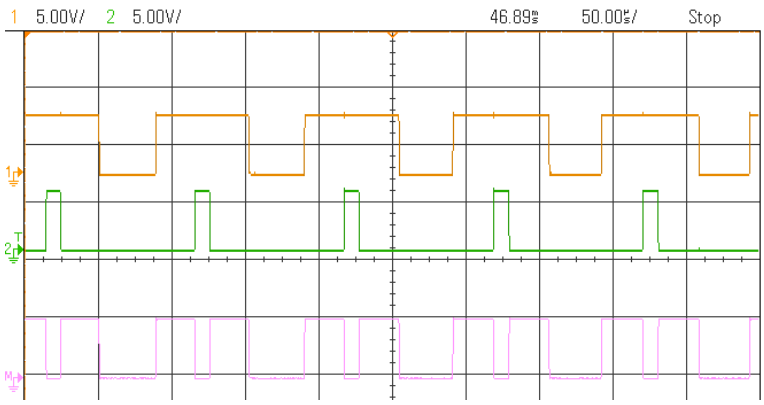


Figure 4.9: Gate pulses from FPGA (Scale - x:50 μs/div, y: 5 V/div)

The induction motor, when driven directly from the transformer, started to rotate at $24\,Vrms$ of line voltage. It needs sufficient torque to overcome the friction and windage losses. The inverter is connected to the motor.Switching Conditions for inverter: $f_{sig} = 50\ Hz, f_s = 10\ kHz, m_a = 0.8$ and motor is on no load. The DC bus voltage is slowly increased till the motor starts to rotate. The waveforms observed at that instant are:

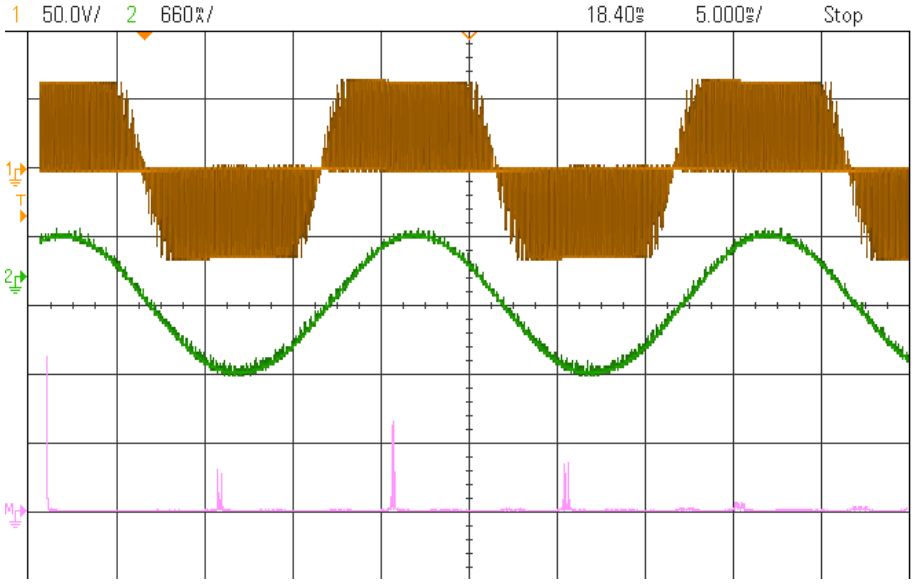


Figure 4.10: 1) V_{ab} 2) I_a M) FFT of V_{ab} (Scale - $x : 10\ kHz, y : 10\ V$)

Observations:

- Minimum V_{dc} for motor to rotate is 60V. The line voltage is unipolar. This is because, the t_{ON} of Phase A subsumes the t_{ON} of Phase B, for $\frac{T_s}{2}$, and vice versa for rest of T_s.
- The current waveform is sinusoidal at $50\ Hz$ and amplitude is 660 mA. The inductive motor load does not respond to the switching components ($I = \frac{V}{j\omega L}$) and gives a filtering effect.
- The FFT spectrum shows the fundamental at $50\ Hz$ and its rms amplitude is $24\ V$. The harmonics at multiples of f_s, $10\ kHz$ can be seen. Fig 4.9 shows that line voltage switches two times in one switching period. Hence, $20\ kHz$ component is higher than $10\ kHz$ one.

The table below, gives the observations for varying DC bus voltage. The speed was measured using a speed module, which is discussed later. The PWM is linear modulation region.

Table 4.2: Motor response to SPWM

V_{dc} (Volts)	V_{line} (fund) $Vrms$	I_{ph} (ampl) mA	Motor speed rpm
60.6	25	712.5	290
65	26.7	750	416
71.25	30	669	959
82.5	35.2	475	1271
88	37	450	1318
95	41	406	1361
102.5	43.5	394	1387

4.4 Sine wave generator

3 sine waves are generated internally in the FPGA for using in constant Volts/Hz control. The sine waves are phase shifted by $0°, 120°$ and $240°$ and given as the reference voltage signals. The frequency is made tunable in the range : $[5, 50]Hz$ and with a resolution of 1. This is implemented as follows:

1. A fixed waveform can be stored in the BRAM of the FPGA (called as LUT in LabView). This does not consume any logic resources. A sine wave digitized with 3000 samples/period and 2^{16} (16-bit integer) quantization levels is divided into three equal segments, each $\frac{2\pi}{3}$ rads. 3 sets of 1000 samples each are stored in 3 LUTs.

2. There is a counter (say, A) running which scales 0 to 999. This serves as the address for all the LUTs. After reaching 1000, a counter B (takes values 0,1,2)is incremented. The following figure shows the logic. Ptr_a corresponds to v_a^*, and so on. Depending on the counter B value, a pointer is assigned to a particular LUT.(Inner case structure in Fig. 4.12) This is shown in Thus, each pointer reads from LUT 1, LUT 2 and LUT 3,in a cyclic order. Hence, v_a^*, v_b^*, v_c^* take all values of the sine wave, but shifted by 1000 samples from each other.

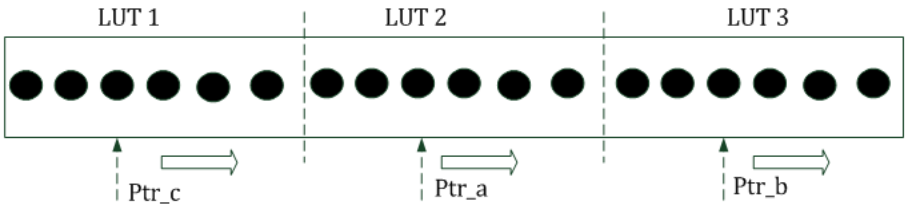


Figure 4.11: Pointers and the LUTs

3. The Single Cycle Timed loop (SCTL) runs at $40\ MHz$. But the increment of counter A, happens every T_{loop} no. of CLK cycles, which is controlled to vary output sine frequency. For a certain f_{sig}, the pointer should read 3000 samples in $T_{sig\ s}$.Hence,

$$T_{loop} = \frac{1}{3000} \times T_{sig} \times f_{CLK} = \frac{4}{3} \times 10^4 \times \frac{1}{f_{sig}} \tag{4.12}$$

   This is also implemented as an LUT of $f_{freq}\ vs\ T_{loop}$. Hence $[5, 50] \leftrightarrow [2667, 267]$.

4. The Out put sine wave can be scaled down to get the required value. For example, to make it fall within the range, $[-1, 1]$, $v^*\ >>\ 15$ will suffice. This is just a bit-shifting operation and consumes minimal resources.

Thus a sine wave of high accuracy and resolution in amplitude as well as frequency is obtained.

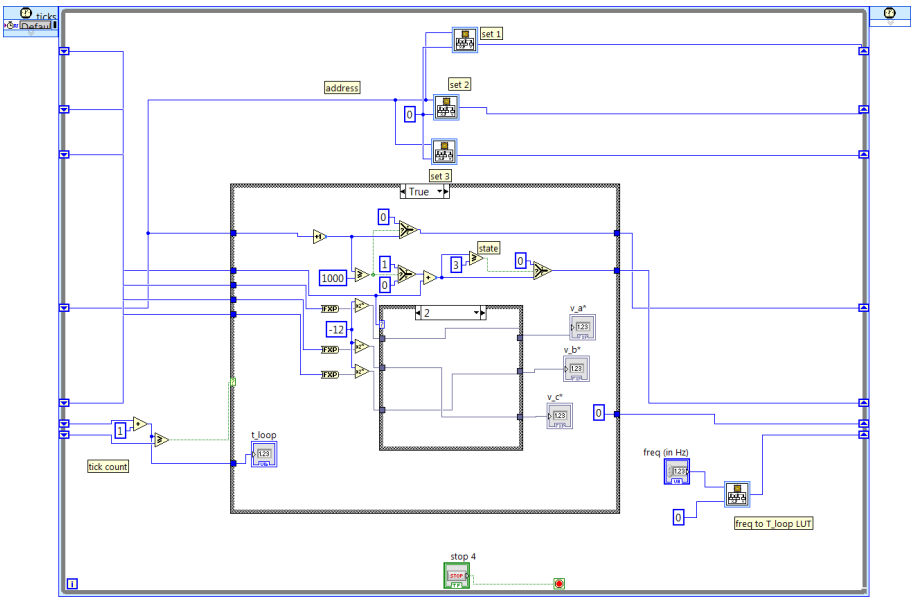


Figure 4.12: Internal Sine Generator VI

4.5 Speed detector

4.5.1 Signal Conditioning circuit

The motor drum consists of 14 slots. An infra-red emitting diode and photo-transistor, H21A1 is fitted to the body, such that as the motor rotates, the slots interrupt the light. This results in a digital pulse, signal at the transistor (B_1), from which speed information can be deduced. The four terminals come out through a RS-232 cable.The resistance, R_d is chosen, so that constant current, $I_f = 20\ mA$ flows through the diode. The other values are: $R_l = 3\ k\Omega, R_1 = R_2 = 20\ k\Omega, R_3 = 4.7\ k\Omega$. The B_1 gives a pulse with rising and falling times in the range of $25 - 100\ \mu s$, which is a high value. The DI of NI 9423 requires a digital signal of levels: 15 V/ 0 V. To make the transition faster and change the levels, one more npn transistor, B_2 is used. The DI is of the sinking type, that is, it draws around $4\ mA$ in ON state. If connected to collector of B1, there is drop comparable to 15 V across R_3. Hence a second pnp transistor, B_3 is put at the final stage, which connects the DI directly to supply rail, when ON. The transitions are observed to be faster and the FPGA detects the levels properly. When the motor is running at constant speed, the time period of the pulses is T_{slot}. The speed of motor, in rpm is given by:

$$\omega_m = 60 \times \frac{1}{14 \times T_{slot}} \tag{4.13}$$

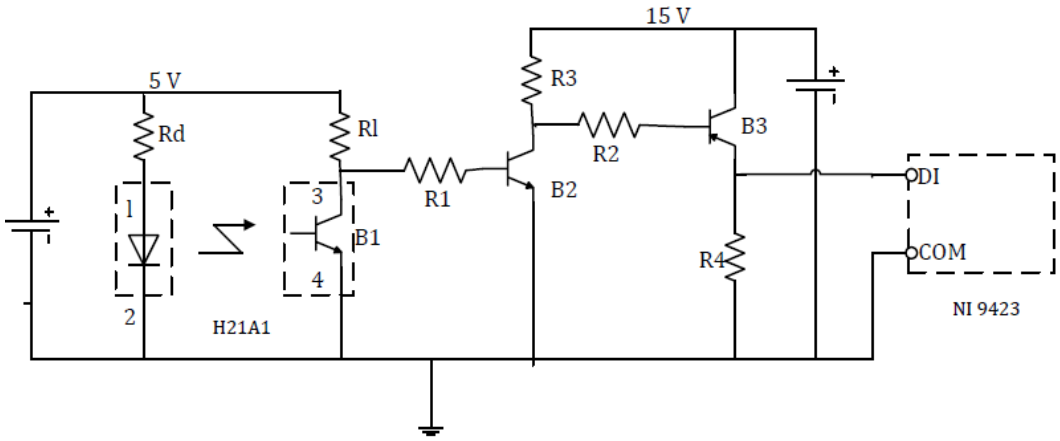


Figure 4.13: Interface circuit for H21A1

4.5.2 Time averaged detector

The number of rising edges for a fixed observation period (t_{upd}) is recorded, using a counter. This is directly proportional to the speed. The edge detect = $a[k]\ AND\ a'[k-1]$. The free msec-running timer, gives t_{curr}, and initial time-stamp is from when the count starts. When $t_{curr} - t_{init} = t_{upd}$, the value of edge count (n_{edge}) is output.

$$\omega_m(rpm) = 60 \times \frac{n_{edge}}{14} \times \frac{1}{t_{upd}} \tag{4.14}$$

Thus, we see that the resolution in $\omega_m \propto \frac{1}{t_{upd}}$, for a fixed number of slots. Here, for $t_{upd} = 1\ sec$, resolution is 4.286 rpm. In motor speed control loops, value of speed is needed every $\frac{1}{10}^{th}$ of the mechanical time constant, $\frac{J}{B}$. With this sensor, there is a trade-off between accuracy and speed of the control loop. This is an inherent problem which can be remedied by increasing the number slots/rotation.

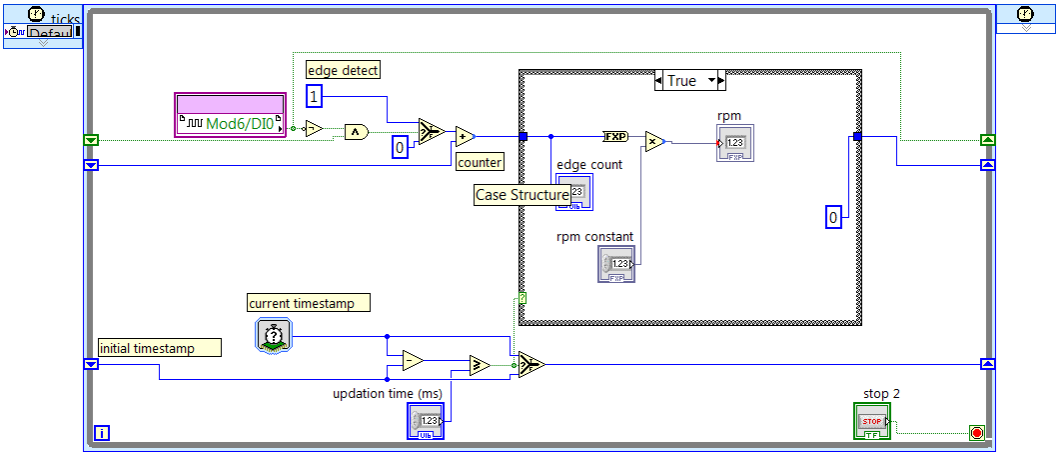


Figure 4.14: Time Averaged rpm VI

The motor voltage was increased from the transformer till motor rotates at 1380 rpm and the speed is sensed. The calculated rpm value in the FPGA is output through Analog Output onto the oscilloscope. It can be seen that the updation time is 500 ms and the value changes in fixed discrete steps.

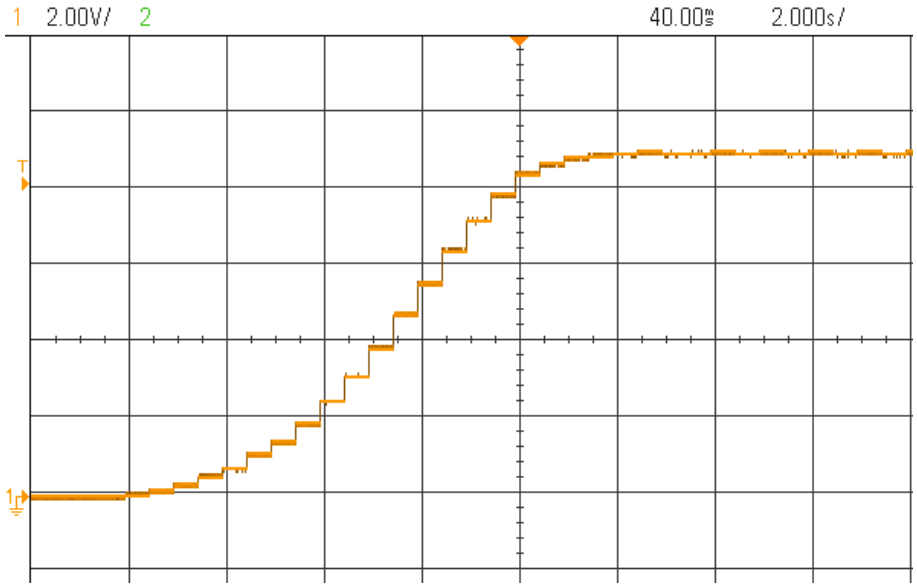


Figure 4.15: Plot of motor accelerating: Testing the speed module (Scale - x: 2 sec/div, y: 2 V/div)

4.5.3 Speed by Period measurement

An alternative to above scheme is to measure the value of T_{slot} . This is directly proportional to the period of 1 rotation. The time between two consecutive rising edges is measured using a 16-bit μs timer. The challenge lies in doing the division, as this operation is very resource expensive in an FPGA. Another option is to implement a LUT, as the dividend is known. But, still there is a problem. As seen in following graph, the ranges are:

$$\omega_m[100, 1800](in\ rpm) \leftrightarrow T_s : [2381, 42857](in\ \mu s) \tag{4.15}$$

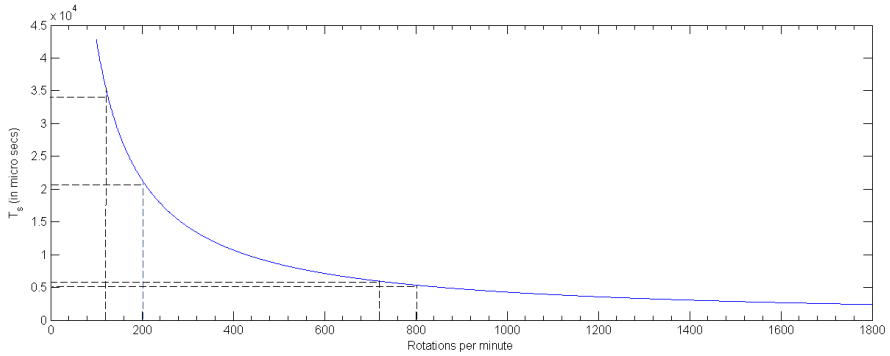


Figure 4.16: T_s vs rpm plot : equation 4.13

The forward mapping has time taken by 1 slot as input and storing such enormous data is not only infeasible, but also unnecessary. Hence, an iterative reverse mapping strategy is adopted. Let ΔT_s denote difference in T_s for every 1 rpm change in ω_m. From the graph, it can be seen that $\Delta T_s \propto \frac{1}{\omega_m^2}$. Hence, a wider range of T_s maps to 1 rpm at lower speeds. This can be used, by storing equispaced speed information in the LUT. 1700 values of T_s (16-bit unsigned integer) are stored, corresponding to the address, which is speed. The upper and lower limits are taken care off. The following algorithm is executed in each FPGA loop ($25\ ns$):

1. Initialize $\omega'_m = 100$. $T'_s = f(\omega'_m)$
2. Compare with actual recorded value of slot time and get error: $e = T_s - T'_s$
3. If $e < 0$, increment: $\omega'_m[k] = \omega'_m[k-1] + 1$; If $e > 0$, decrement.
4. When the desired value of $\omega_m \leftrightarrow T_s$ is reached, there will be a change in the sign bit of error. When $e[k].e[k-1] < 0$, stop the process and output the value of ω_m

This process works via successive approximation of the LUT output, to meet the input requirements. It is depicted in figure 4.17. The crossed is the actual value, but settles at the following value in the table. It will converge to actual value in maximum $1700 \times 25\ ns$, which is well within the time at which T_s is updated (max $1000\ \mu s$). This gives a degree of instantaneity to the speed output. The maximum absolute error is ± 1 rpm. The division module is tested separately and found to give the correct values.

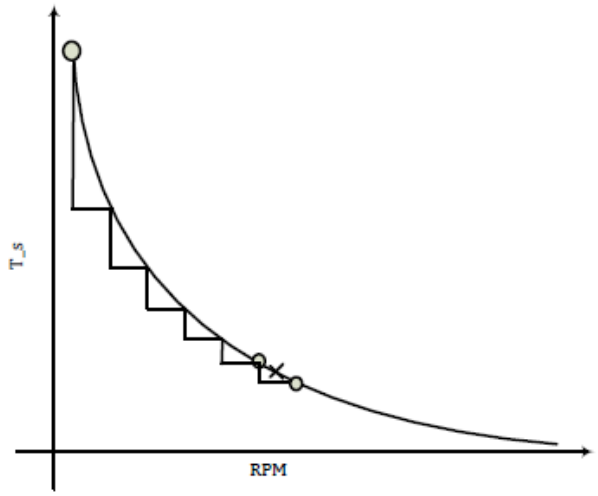


Figure 4.17: Successive approximation type division

There is a flaw in functioning of the complete unit. The problem is in the period recording. It saturates to its maximum value. Thus, a noisy speed profile is measured by the module in fig. 4.18.

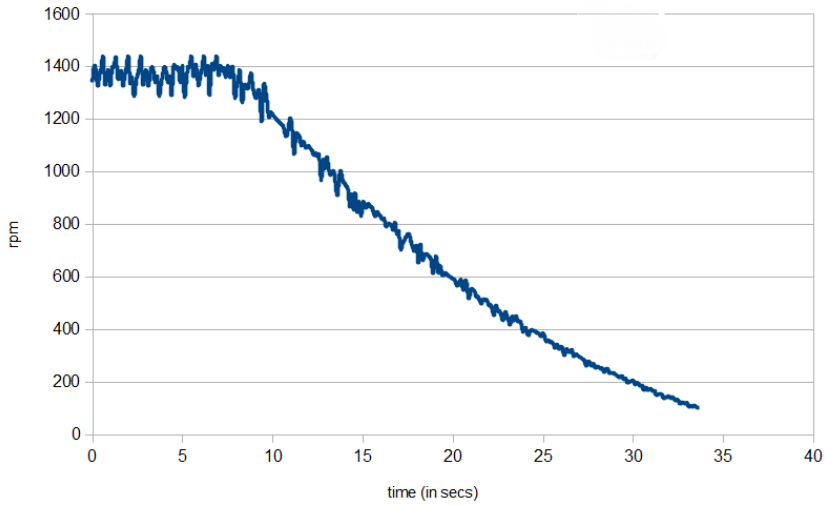


Figure 4.18: Motor decelerating: Testing speed module

4.6 Open loop Constant Volts/Hertz control

A common and simple scalar control scheme is V/f control. This is implemented here, to evaluate the SPWM scheme and to get the feedback signals. In low slip region, the stator flux, $\psi_s = \frac{V_s}{\omega_e}$. If flux is maintained at a constant, the torque can be varied by varying stator voltage, V_s. This gives a faster transient response than constant frequency, voltage control. Moreover, the air gap flux does not saturate as in low supply frequency and rated voltage. Thus, the commanded frequency and f^*_{sig} and modulation index, m^*_a are increased proportionately. As $m_a = 0.8$ corresponds to 50 Hz, the linearity constant is $\frac{0.8}{50}$. f^*_{sig} is the primary control variable. The steady state response of the drive system is tabulated below. The parameters are: $f_{sw} = 10\ kHz, V_{dc} = 102.5\ V$

Table 4.3: V/f: Steady state characteristics

$Freq$ Hz	V_{ph} (Vrms)	I_{ph} (mA) rms	Speed (rpm)	Slip
5	1.35	122	0	1
10	2.6	214	0	1
15	4.36	245	261	0.42
20	6.12	247	453	0.245
25	8.12	258	616	0.179
30	10.08	262	783	0.13
35	11.84	272	929	0.115
40	13.31	274	1083	0.0975
45	15.31	287	1225	0.0925
50	17.35	292	1378	0.081

Observations from the table:

1. The phase voltage increases in proportion to the frequency.
2. Initially, most of the stator voltage drops across the stator resistance, and the flux has not attained rated value. Hence, torque and speed is low.

The DC bus voltage is maintained at 400 V. The sine frequency is ramped up from

5 to 50 Hz in steps of 1 Hz, over a time span of 9 secs (Fig. 4.19). The motor speed vs time characteristics is plotted. (Fig. 4.20)

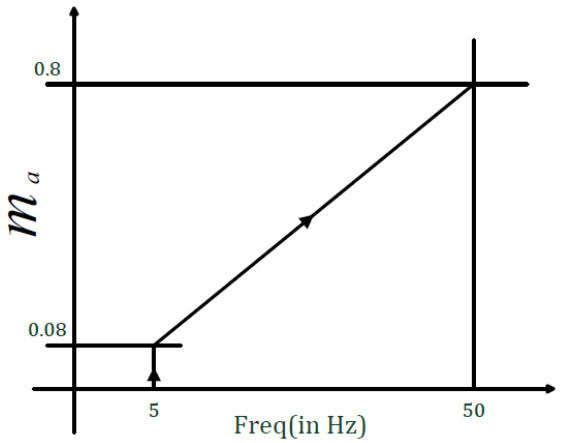


Figure 4.19: V/f control: Ramp

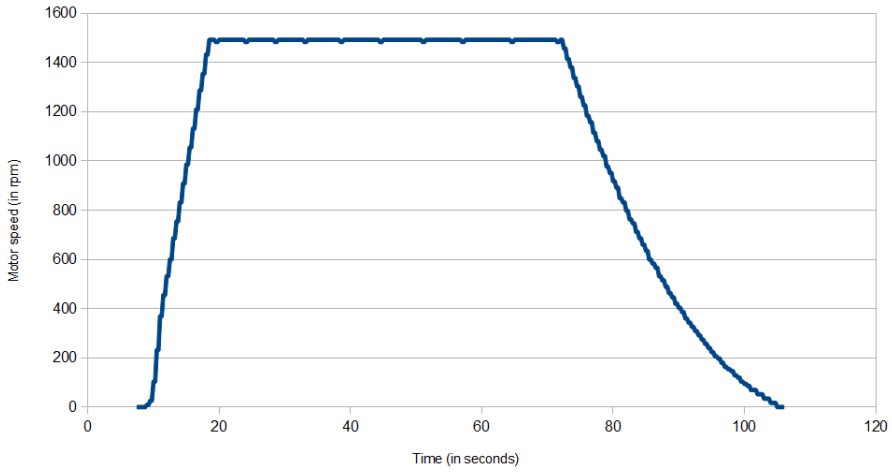


Figure 4.20: V/f control: Transient speed profile

The start-up speed follows a linear curve, thus closely tracking the command, ω_e^*. This shows that the rated value of flux has been attained much earlier than previous case, as DC bus voltage is higher. The steady state speed is close to the synchronous speed of 1500 rpm. The PWM is switched off abruptly. Now the curve is characteristic of first order dynamics (exponential decay), determined by the motor mechanical constant $\frac{J}{B}$. The following plot gives the waveforms at $m_a = 0.8$.

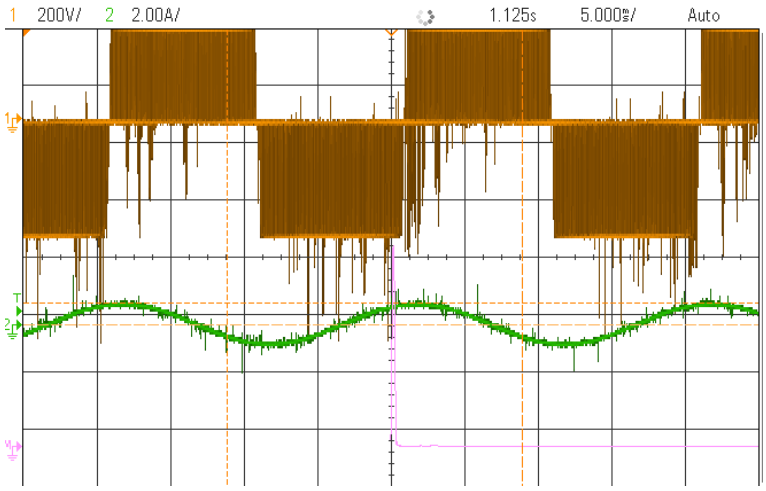


Figure 4.21: V/f control: (Scale of x: 5 ms/div) 1)V_{ab} (200 V/div) 2)I_a (2 A/div)

4.7 Current sensing and Transformation

4.7.1 Signal conditioning circuit

The phase currents are taken as feedback for estimation and control. Two of the values, i_a and i_b are sufficient, as motor is star connected. The sensing is done using LA 55-P Current transducer. It gives a conversion ratio, $I_S : I_P$ of 1:2000, and the measuring resistance, R_M can be maximum 300 Ω. The range of primary current for the motor is 0 to 2 A, with sufficient resolution. The signal is input to the FPGA, through the Analog Input module, NI 9215. The 16-bit ADC has a input range of [-10,10] V. Thus for accurate current sensing, the sensor signal should be amplified and filtered. For this, 2 sets of the circuit given in Fig. 4.22 were built.

3 turns of the motor terminal wire is wound on LA 55-P. The secondary current, passes through R_M. The potential drop across it is $V_i - V_{gnd}$. There may be magnetic field interference in the transducer and ground noise in the breadboard(denoted by V_{gnd}). Hence, a general purpose OPAMP $\mu A741$ is used in differential configuration, which removes the effect of common mode noise. The differential mode signal consists of the desired signal and some noise, which can be filtered using RC. The transfer function

for the circuit in Fig. 4.22 is:

$$V_o(s) = (V_i(s) - V_{gnd}(s))\frac{R_2}{R_1}\frac{1}{1+sR_2C} \tag{4.16}$$

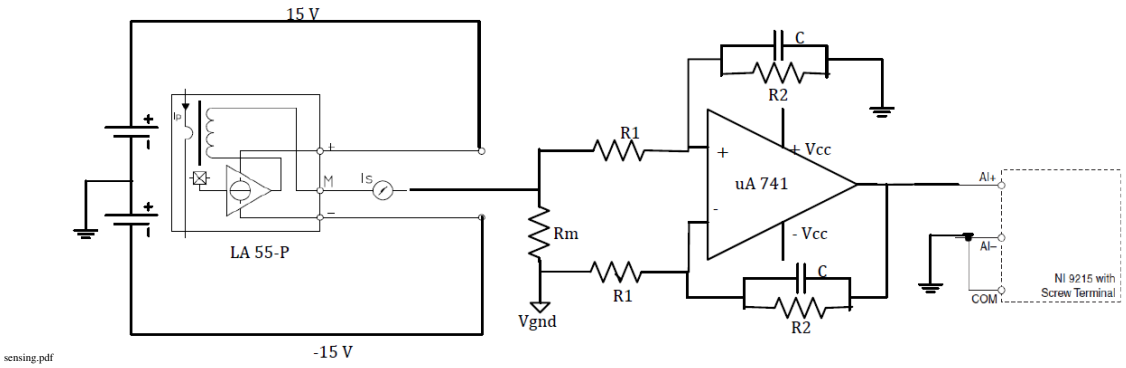


Figure 4.22: Current transducer + Interface circuitry

The values chosen are : $R_1 = 2.5\ k\Omega, R_2 = 14.5\ k\Omega, R_M = 300\ k\Omega, C = 10\ nF$. This gives a DC closed loop gain of 5.8 and filter corner frequency, $\frac{1}{R_2C}$ of 6.9 kHz. The output signal was observed on the oscilloscope and found to have the same shape and phase as that from a current sensor. It was noise-free and gave a net scaling of 6 V/A. This is the per-unitized value stored in the FPGA, that is $1A \leftrightarrow 6V$.

4.7.2 Unit Vector generator

The value of synchronous angle, θ_e is needed for the transform. In discrete domain, the integration of ω_e is implemented as repeated addition on a shift register.

$$\theta_e = \int \omega_e dt \Rightarrow \omega_e = \frac{d\theta_e}{dt} \tag{4.17}$$

$$\omega_e[k] = \frac{\theta_e[k] - \theta_e[k-1]}{T_s} \tag{4.18}$$

$$\theta_e[k] = \theta_e[k+1] + \omega_e[k].T_s \tag{4.19}$$

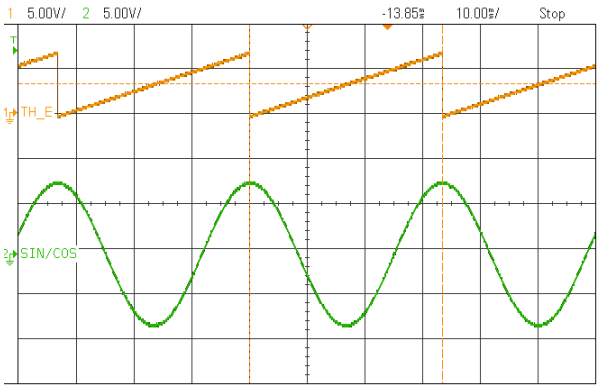
This unit is put in the PWM loop, which runs at $10\ kHz$. Hence, $T_s = 100\ \mu s$. Unit of ω_e: rpm, T_s : μs. These are very small units. So, to reduce the size of multiplier,adders

and registers, the product is scaled by 13 bits ($\omega_e[k] >> 10, T_s >> 3$). In the FPGA, the angle is stored as (per-unitized): $[0, 7324) \leftrightarrow [0, 2\pi)$. After hitting 7324, it wraps around to 0.

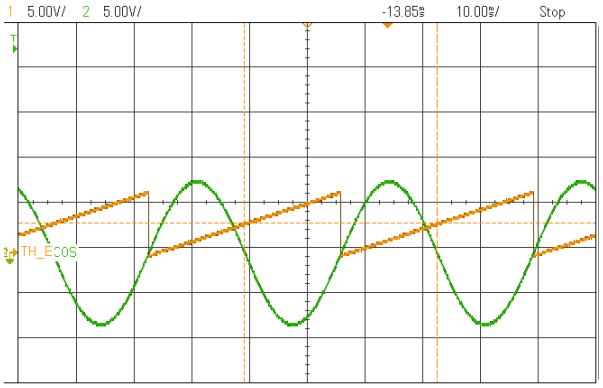
$\sin(\theta_e)$ and $\sin(\theta_e)$ are generated by sharing one LUT of 7324 elements. A sine waveform (quantized to 16 bits) with index(i) as θ_e is stored in the LUT. Generating sine is trivial. Cosine is created by the following piecewise function, $i = f(\theta_e)$ (let $\frac{7324}{4} = \Delta$):

$$\theta_e \in [0, 3\Delta - 1] : i = \theta_e + \Delta \leftrightarrow \cos\theta_e = \sin(\theta_e + \frac{\pi}{2}) \tag{4.20}$$

$$\theta_e \in [3\Delta, 4\Delta - 1] : i = \theta_e - 3\Delta \leftrightarrow \cos\theta_e = \sin(\theta_e - \frac{3\pi}{2}) \tag{4.21}$$



Cosine
Scale - x: 10 ms/div, y: 5 V/div



Sine
Scale - x: 10 ms/div, y: 5 V/div

This unit was tested and found to function properly. In fig. 4.7.2, the ω_e was set as 1800 rpm. The frequency of the output wave is 30 Hz. The signal is glitch-free and has sufficient accuracy.

4.7.3 Axes Transformation

Non-power invariant transformation is used for changing the currents from stator abc frame to stator dq frame. The equations are:

$$i^s_{qs} = i_{as} \tag{4.22}$$

$$i^s_{ds} = \frac{1}{\sqrt{3}}(i_{cs} - i_{bs}) \tag{4.23}$$

$$= -\frac{1}{\sqrt{3}}(i_{as} + 2i_{bs}) \tag{4.24}$$

In binary, $\frac{1}{\sqrt{3}} = 0.1001001111$. Thus the above multiplication is implemented as a series of bit shifts and additions.

$$n*[2^{-1}+2^{-4}+2^{-6}-2^{-10}] = (n >> 1)+(n >> 4)+(n >> 6)-(n >> 10) \quad (4.25)$$

The unit is tested by running the motor using the SPWM and sensing the currents. The following oscilloscope plots give i_a $(= i_d^s)$ and i_q^s. It can be seen that d-component leads q-component by $90°$ and both are of the same amplitude.

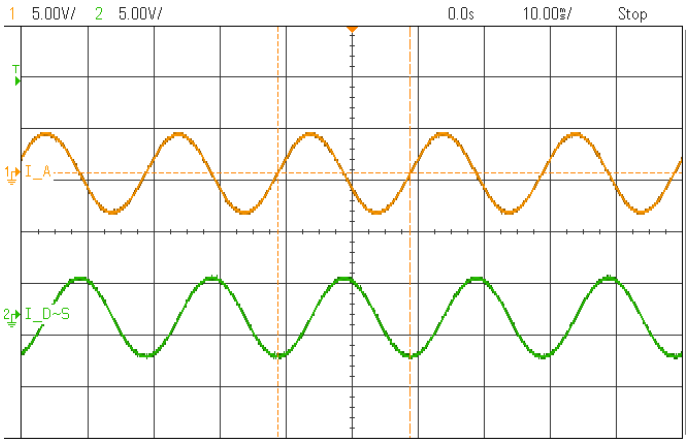


Figure 4.23: Stator frame d-q axes currents (Scale - x: 10 ms/div, y: 5 V/div)

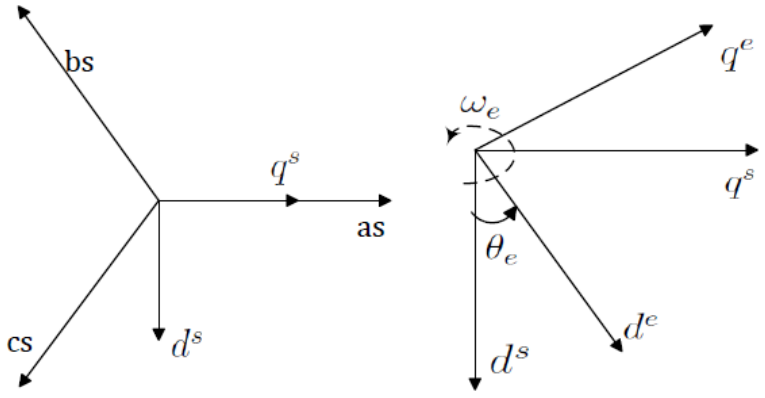


Figure 4.24: abc to dq frame

The stator reference frame, $d^s - q^s$ and synchronous reference frame, $d^e - q^e$ are depicted in fig. 4.24. If stator current phasor is represented as $\bar{i}$, the equations are:

$$\overline{i^s} = i_q^s - j i_d^s \; and \; \overline{i^e} = i_q^e - j i_d^e \tag{4.26}$$

$$\overline{i^s} = -j I_m e^{j\theta_e} \tag{4.27}$$

$$\overline{i^e} = j\overline{i^s} e^{-j\omega_e t} \tag{4.28}$$

$$\Rightarrow \overline{i^e} = I_m e^{-j(\theta_e - \omega_e t)} = I_m e^{j\theta m} \tag{4.29}$$

$$i_q^e = I_m \cos(\theta_m) \; and \; i_d^e = I_m \sin(\theta_m) \tag{4.30}$$

Here, the angle, θ_m is the initial phase lag between the rotating axis and phasor. Thus, the dq current components appear as DC quantities in the rotating frame. This transformation is described in the following equations:

$$i_d^e = i_q^s \cos(\omega_e t) + i_d^s \sin(\omega_e t) \tag{4.31}$$

$$i_q^e = i_d^s \cos(\omega_e t) - i_q^s \sin(\omega_e t) \tag{4.32}$$

The following 4 FPGA circuits run in parallel, to implement the complete function, described till here in this section.

Table 4.4: FPGA Loops for Axes transform

No.	Loop frequency	Function of block
1	$10\ kHz$	Integration to get angle ($\omega_e \rightarrow \theta_e$)
2	$2.5\ MHz$	abc to stator dq frame / Summing up the products to give i_d^e, i_q^e
3	$2.5MHz$	Angle to Sine/Cosine conversion
4	$40MHz$	The 4 multiplications in eqns 4.31 and 4.32

A proper timing analysis is required for evaluating the robustness of this system. Block 4 executes 16 times in the time, that blocks 2 and 3 run once. Block 4 gets its input from block 2 and 3. Thus, the inputs can be expected to remain stable for the outputs to be generated properly. The following diagram explains it.

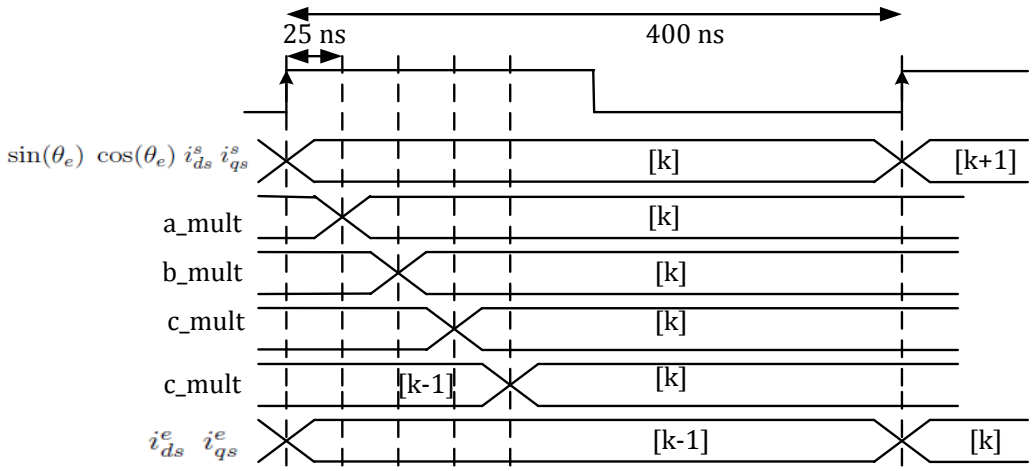


Figure 4.25: Transform calculation: Timing diagram

As shown in the figure, the value of synchronous dq frame currents corresponds to the angle and currents at the previous instant (400 ns before). But, then the vector scheme needs these values (samples) only once in $100\ \mu s$. So a lag of $400ns$ will not introduce much error in the sampled value.

The results given by the transformation are plotted below. The phase current is sinusoid at $50\ Hz$. The value of synchronous speed was tuned to get DC values for the rotating frame currents. It was close to DC for $\omega_e = 3030\ rpm$. But, there is a small AC ripple of $50\ Hz$, riding on it. The q-axis current is 0, suggesting that the d^e axis has been aligned to the stator current phasor $\overline{I_s}$. Note, that in vector control we attempt to align the d^e axis along the rotor flux phasor, $\widehat{\psi_r}$. Here, it is an arbitrary frame. For different values of stator current, it was verified that the transform preserves the magnitude:

$$\sqrt{(i^e_{qs})^2 + (i^e_{ds})^2} = |\overline{I_s}| = Amplitude\ (i_a) \tag{4.33}$$

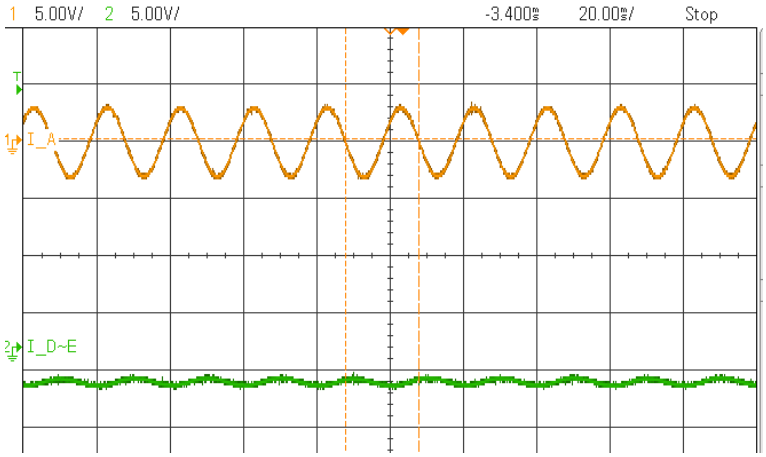


Figure 4.26: d-axis component of stator current (Scale - x: 20 ms/div, y: 5 V/div)

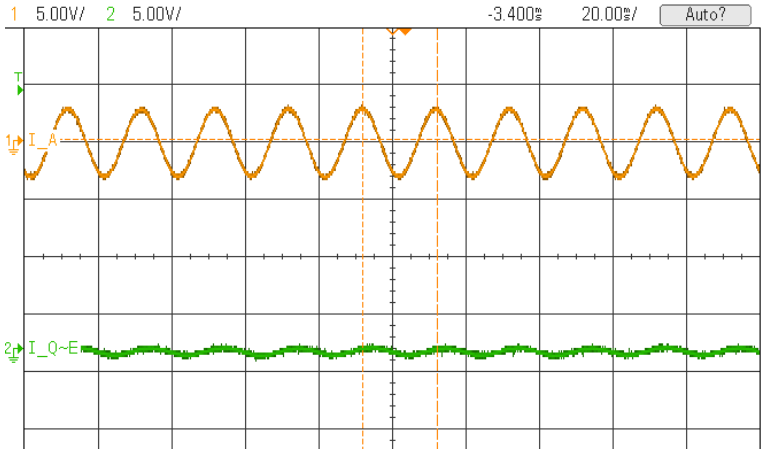


Figure 4.27: q-axis component of stator current (Scale - x: 20 ms/div, y: 5 V/div)

CHAPTER 5

CONCLUSION

In this project, the speed control of an induction motor using vector control was studied. The motor was modelled in synchronous dq frame, and the equations governing the dynamics were derived. This formed as a basis for building the control and estimation scheme, to run the simulations. The results show that the decoupling between the torque and flux occurs, when the motor starts. Later, the decoupling is preserved. Any variations in speed or load corresponds to only the torque component responding. The PI constants were chosen to give acceptable overshoot and settling time.

5.1 Hardware design

The new PWM scheme gives a satisfactory performance. There is no audible noise due to the switching components. The FFT spectrum shows the highest peak at the fundamental and the 2nd harmonic ($2 * f_s$) is higher than the 1st harmonic. The sine wave block generates the reference waveform for frequencies in the range : $[5, 50]\ Hz$, with sufficient time accuracy and amplitude resolution. Two methods of getting the motor speed, from the photo-transistor pulses were designed. They were found to give the correct rpm values, when compared with a tachometer. There is a contention between updating time and accuracy, in the two methods. The motor is soft-started using the Open loop constant volts/Hertz control. The speed profile shows a linear increase with input frequency (f_{sig}). The analog interface circuits for current and speed signals were built, to take them into the FPGA. The unit vector rotating at synchronous speed was generated and the transformation form abc to dq frame was implemented. While the motor was operating, the feedback signals were taken, processed and observed on oscilloscope. The dq axis currents were DC values.

FPGA served as a good platform for the design. The units used in the design of the final FPGA hardware are:

1. Sine Generator
2. 3 phase PWM
3. Time averaged speed detector
4. Constant Volts/Hertz
5. Current axes transformation

An analysis of the Xilinx log report gives an idea about the efficiency of the circuit design. As the application does not demand time accuracy to the level of few ns, the timing parameters such as critical path delay, setup/hold time are not presented here. The consolidated resource usage is tabulated below:

Table 5.1: Final Device Utilization

Component	Quantity	Total available	Percentage
Slice LUTs	6536	51,840	12%
Slice LUT Flip-flop pairs	8206	51,840	15%
Slice Registers	4674	51,840	9%
Total Slices	3107	12960	12%
Block RAM	8 (234 kb)	96 (3456 kb)	6 %
DSP48E Multiplier	9	48	18 %

5.2 Future scope of work

The feedforward estimation of the slip frequency and PI controllers (for flux and speed loop) have to be built. As a Voltage Source inverter is being used, the reference voltages should be generated after eliminating the cross-coupling terms. The design in FPGA can be made economical by resource sharing. Before closing the loop, the speed signal should be with sufficient accuracy and speed. Hence, the number of slots on the motor periphery has to be increased.

REFERENCES

[1] Bimal K. Bose *Modern Power Electronics and AC drives*, Eastern economy edition,2002.

[2] R. Krishnan. *Electric Motor drives: Modelling, Analysis and Control*, Prentice Hall,Third Edition, 2002.

[3] Semikron Power Electronics Teaching Kit: Application Note

[4] Operating Instructions for NI cRIO 9022, cRIO 9116, 9215, 9263, 9423, 9474

[5] Getting Started with CompactRIO and LabView

[6] LabView FPGA Course manual

[7] H21A1-Phototransistor and LA 55-P - Current Transducer datasheets

[8] Drawing Tool: Microsoft Visio 2007