POLITECNICO DI MILANO
Facolt`a di Ingegneria dell'Informazione
Master of Science in Automation and Control Engineering

Induction Motor Controlling using FPGA-based platform in a parallel processing environment using Field Oriented Control with analytical analysis over Microcontroller-based solutions

Author:

- SeiedMilad Mohammadi, Mart. 841126

Supervisors:

- Prof. Francesco Castelli Dezza
- Eng. Mattia Rossi

Academic Year 2017-2018

Acknowledgments

This thesis is the outcome of my works done in Politecnico di Milano, in Electrical Motors and Drives laboratory.

First of all, I would like to thank my supervisors Prof. Francesco Castelli Dezza and his Assistant Mattia Rossi for their kind and strong support all over the period of the thesis development and for all the useful and great recommendations they gave me whenever I needed them.

Special thanks to my family and my friends who continuously supported me for finalizing this thesis.

Table of Contents

List of Figures:

List of Tables:

Abstract

Field Oriented Control (FOC) is one of the most known methods for controlling 3-phase motors specially AC motors or Induction Motors (IM or ACIM) inside the industry of motor controlling. Comparing to classical control methods for IMs like V/Hz or other conventional algorithms, FOC can be considered as a much more complex solution with high demands of processing power an in return a very high quality control and performance, that's why despite of being available since long time ago, just recently by improvements in electronics components and rapid growth of processing power of controllers the FOC became a feasible method to be implemented on various types of IM to be a competitive alternate to DC drives.

By improvement of digital processing power of microprocessors or more precisely microcontrollers they became dominant solution of the designs for lots of projects and controllers. Microcontrollers are generally designed with a sequential architecture, meaning that tasks are run in order or they can be interrupted by other tasks, and the environment that microcontrollers are programmed are also based on sequential design like C programming language, which is the core part of programming of almost all the microcontrollers in the market.

During the recent years, by increase of the fame of FPGAs, now they found the chance to be challenged and tested for designing FOC in parallel processing manner. Actually, the main difference between FPGA-based solutions versus the Microcontroller-based solution is that the architecture of the FPGA is based on parallel processing, and there are possibilities to run different tasks in concurrent way without the necessity of braking or interrupting them to be able to do other tasks. This can make a huge difference specially on the freedom of the design and possibility of testing much more different algorithms in FPGA-based solution environments regarding the performance evaluations and reliable control system design.

In this thesis the main intention is firstly to develop an FPGA-based solution for controlling AC motors using FOC in a concurrent manner by actually adapting a previously made microcontroller-based hardware to the new design for the parallel processing control with FPGA-based hardware. Then this will be followed by an interface design between the FPGA-based solution to switching module and finally by making the software and testing the FOC in the real environment. The results will be compared to the simulations and finally an analytical analysis will be given for the comparison of FOC design in FPGA-based environments versus Microcontroller-based environments.

Chapter 1:

Introduction:

The goal of this chapter is firstly introducing the thesis topic in brief and talking about challenges which should be overcome, then there would be an introduction to the literature of the topic with explanations of how the modeling are done. Finally, there will be a discussion over the architecture of the project which is used.

1-1 Thesis Topic description:

The main topic of Thesis is controlling of AC motor using FPGA based platform by designing and customizing the interfaces and the invertor board using Field Oriented Control (FOC) methodology.

In nutshell, since FPGA is rather a new product in electronic world firstly introduced at late 1980s, the usage and commercialization for motor controlling industry is remained with lots of space to be discovered and tested.

Currently microcontrollers are leading mostly the motor controlling markets specially for cost effective products for obvious reasons which one of the most important ones is the huge price gap between a simple microcontroller-based platform versus a FPGA based platform capable of controlling a motor.

There are quite lots of reasons for this difference in the prices which can be categorized simply as following:

1- FPGAs are not solely capable of performing a motor control task and they must be embedded inside systems with other parts like DSPs, ADC units and so on. So, this naturally effects drastically the price while the microcontrollers are mostly featured with dozens of capabilities in stand-alone manner with no need to external components.

2- Currently due to huge demand of market for cost effectiveness of solutions, the microcontrollers have much more software and libraries available to meet the needs of crowds, while for FPGA based products the support and variety of solutions is incomparable and most of the solutions are propriety with license requirements.

3- Since FPGA based platforms are sensitive in terms of risk of failure or damage, the circuits designed for them demands much more protection comparing with microcontrollers, so this leads automatically more sophisticated electronics and subsequently higher price.

But up to now one may ask what is the advantage of FPGA versus microcontrollers? In the next section the answer will be elaborated.

1-2 why to use FPGA over microcontrollers

To answer this question, we need to go a little bit in details of comparing the architecture of conventional microcontrollers versus FPGA based solutions. The main differences are categorized in two parts of parallel processing and the determinist task execution in terms of timings.

1-2-1 Parallel processing

Unlike processors and microcontrollers, FPGAs are truly parallel in nature, so different processing actions can be done completely in parallel with other tasks without braking or interrupting other parts unless required. In FPGA each independent processing task is assigned to a dedicated section of the chip and can function autonomously without any influence from other logic blocks. As a result, the performance of one part of the application is not affected when you add more processing units as long as there are enough resources inside the FPGA by other parts.

This is totally different in microcontrollers and due to the architecture of conventional microcontrollers their design is based on the interrupts and prioritization of tasks, so eventually the interrupts with higher priority can brake all the other executive tasks with lower interrupt priority which is totally different with FPGAs.

1-2-2 Deterministic execution

The other main important difference is the determinism in timings, meaning that in FPGA it's completely possible to execute a task within a fixed time frame all the time, so in this case one makes sure that the whole processing or sampling time remains the same all over the execution of the tasks. In microcontrollers considering the interrupting behavior this ability is highly fragile, since some main interrupts in the core have the highest priorities and they can break almost any functionality which eliminates a perfect time determinism.

The two mentioned features are crucial when you are dealing with sensitive and demanding algorithms in sense of processing and efficiency like FOC, in another word these features will enable you for example to make sure that at each time instance each executed action is based on exactly the most recent sample taken exactly on requested time.

In The last chapter we will talk more in detail about these differences and how they can shape the main differences between motor Controlling using Microcontrollers versus FPGA's in FOC controlling of Induction Motors.

Chapter 2

2. Literature Introduction and Modeling

Introduction:

In this chapter we are going to explain the dynamics of the Induction machines in general, including the equivalent circuits and the modeling we are going to use for the control.

2-1 Induction Machine with squirrel cage Rotor:

Figure 2-1 shows the cross section of an induction machine with the squirrel-cage rotor. The squirrel cage is made by some set of conductors (shown in red), which are short circuited at both ends by a conductive ring. In Figure 2- 1 a three phase two-layer winding is placed in stator of a four-pole machine (Rik De Doncker, 2011).

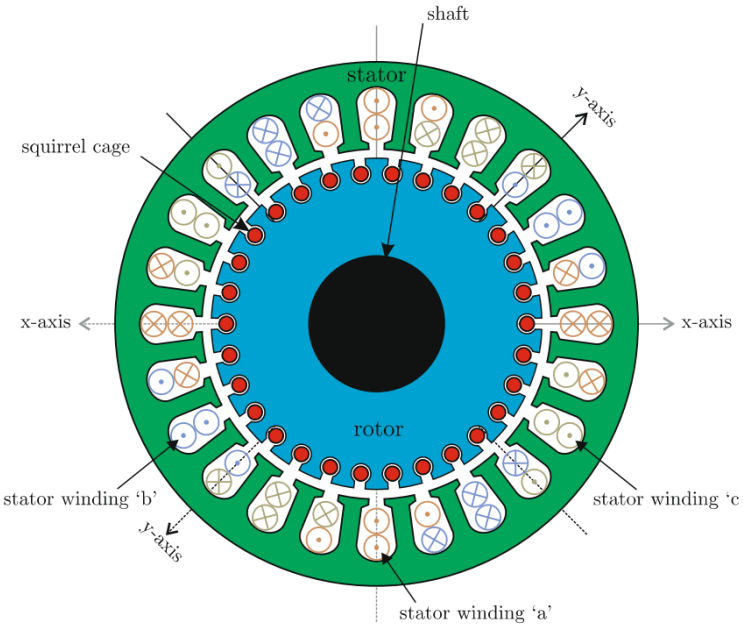

Figure 2- 1: Induction Machine internal Structure (Rik De Doncker, 2011)

A rotating field created by the stator winding is penetrating the rotor. If the rotor is rotating asynchronously to the stator field (which means it is rotating at a different speed), alternating currents are induced in the squirrel cage. These currents, together with the stator field, are responsible for the torque production of the machine. Therefore, asynchronous machines are also known as induction machines.

2-2 Dynamic model of an induction machine

The AC machine models, necessary for the development of control systems, are obtained using prerequisite mathematics in relation to what is called the unified theory of electrical machines. For modeling the induction motors, we will consider the structure of a three-phase induction machine. Suppose we have a three-phase winding either in the stator and in the rotor (consisting of three windings physically displaced by 120 mechanical degrees), If the electromagnetic structure is equipped with only two poles, and by considering the hypothesis that the iron permeability is very high with respect to air.

Suppose that the machine is rotating at a speed Ω_m, the behavior can be studied using the mutually coupled circuits theory (Dezza, 2017), which brings a system of equations like (2.1):

$$v_k = R_k . i_k + p\psi_k \quad (2.1)$$

where *k* refers to one of the six windings, R_k is the resistance of the k-th winding, *p* is the time derivative operator and ψ_k is the flux linked with the k-th winding. Now we can consider the relationship between fluxes and currents as equation (2.2):

$$\psi_k = \sum L_{ki}(\theta_m) . i_i \quad (2.2)$$

where the mutual inductance $L_{ki}(\theta_m)$ in equation 2.2 is a function of the mechanical angle (θ_m) between the magnetic axes of the rotor and the stator. Now by adding the mechanical energy balance equation as (2.3) we will have:

$$T_e - T_r = J . p\Omega_m \quad (2.3)$$

Now by considering a reference frame fixed with the stator (stationary frame, superscript "s") and by applying the space phasor formula to the three stator equations a new machine is obtained, with only two stator windings, fixed with the stator itself and orthogonal to each other, passed through two currents, which are the real and imaginary parts of the stator current's space phasor, we will have:

$$\overline{v_s^s} = R_s . \overline{i_s^s} + p\overline{\psi_s^s} \quad (2.4)$$

Similarly, for the rotor quantities, referred to a reference frame fixed with the rotor (a frame fixed with the rotor, superscript "r") by applying the space phasor formula to the rotor windings we obtain two rotor equations, fixed with the rotor and orthogonal to each other as below:

$$\overline{v_r^r} = R_r . \overline{i_r^r} + p\overline{\psi_r^r} \quad (2.5)$$

But the two reference frames are not the same, so there is no way to compare the two quantities here, it's necessary to have a unique reference frame from which to look at the variables. There are many possibilities which among them, a reference frame fixed with the stator (stationary reference frame) is chosen. Recalling the expression of a space phasor as a function of the reference frame as in equation (2.6):

$$p\left(e^{-j\theta_m}\right) = -j\dot{\theta}_m . e^{-j\theta_m} \quad (2.6)$$

Now using equation (2.6) we can have the rotor quantities "seen" in a stator reference frame as:

$$\overline{v_r^s} = R_r.\overline{i_r^s} + p\overline{\psi_r^s} - j\dot{\theta}_m\,\overline{\psi_r^s} \quad (2.7)$$

where -θ_m is the angle between the stationary frame and the rotor winding. In these conditions, the two magnetic axes of the stator (orthogonal to each other) are always facing the corresponding magnetic axis of the rotor, for which the mutual couplings are no longer dependent on the position between the rotor and the stator (θ_m). The relationship fluxes/currents is then expressed as equations (2.8) and equation (2.9):

$$\overline{\psi_s^s} = L_s.\overline{i_s^s} + M.\overline{i_r^s} \quad (2.8)$$

$$\overline{\psi_r^s} = L_r.\overline{i_r^s} + M.\overline{i_s^s} \quad (2.9)$$

This relationship is independent of the reference frame used. The important result is that it is the same for the rotor and the stator (θ_m). This model is known as 5 parameter dynamic model of asynchronous machine. (R_s, R_r, L_s, L_r, M)

For the torque expression we use the energy balance strategy. The total instantaneous power entering the system from the stator terminals is the sum of the incoming power from the terminals of the sα winding and from sβ winding terminals. This sum can be represented using the space phasors approach as the real part of the product between the voltage and the conjugate of the current, the same reasoning for the rotor terminals can be applied.

$$v_{s\alpha}i_{s\alpha} + v_{s\beta}i_{s\beta} + v_{r\alpha}i_{r\alpha} + v_{r\beta}i_{r\beta} = Re\left(\overline{v_s^s}\underline{i_s^s}\right) + Re\left(\overline{v_r^s}\underline{i_r^s}\right) = R_s.i_s^{s^2} + Re\left(p\overline{\psi}_s^s.\underline{i_s^s}\right) + R_r i_r^{s^2} + Re\left(p\overline{\psi}_r^s.\underline{i_r^s}\right) + Re(-j\dot{\theta}_m\overline{\psi}_r^s\underline{i_r^s}) \quad (2.10)$$

The total input power is a combination of dissipation into heat by the Joule effect (terms of Ri^2) and the parts used to change the magnetic energy stored in the inductors which will be transformed into mechanical power. In effect, the following terms in equation (2.11) represent precisely the variation of internal energy of the four equivalent inductors (two on the rotor and two on the stator considering the machine isotropic):

$$Re\left(p\overline{\psi}_s^s.\underline{i_s^s}\right) + Re\left(p\overline{\psi}_r^s.\underline{i_r^s}\right) = i_{s\alpha}p\psi_{s\alpha} + i_{s\beta}p\psi_{s\beta} + i_{r\alpha}p\psi_{r\alpha} + i_{r\beta}p\psi_{r\beta} \quad (2.11)$$

So the mechanical power can be expressed using equation (2.12) as below:

$$P_m = Re\left(-j\dot{\theta}_m\overline{\psi}_r^s\underline{i_r^s}\right) = Im\left(\dot{\theta}_m\overline{\psi}_r^s\underline{i_r^s}\right) = \dot{\theta}_m Im(\overline{\psi}_r^s\underline{i_r^s}) \quad (2.12)$$

We also recall the equation (2.13) for the relationship between mechanical power and electrically generated torque,

$$P_m = \Omega_m T_e \quad (2.13)$$

And we also point out that

$\dot{\theta}_m = n_p \Omega_m$(2.14)

Which n_p stands for the number pole pairs at the IM, so finally using equation (2.14) and (2.12) it results:

$T_e = n_p \, Im(\overline{\psi}_r^s \underline{i_r^s})$ (2.15)

If we consider now a new generic reference frame *"dq"*, rotated by θ_s with respect to the stator (Figure 2- 2). The dynamic model of the induction machine, in this new frame, is:

$\overline{v_s} = R_s . \overline{i_s} + p\overline{\psi_s} + j\dot{\theta}_s \overline{\psi}_s$ (2.16)

$\overline{v_r} = R_r . \overline{i_r} + p\overline{\psi_r} + j\dot{\theta}_r \overline{\psi}_r$ (2.17)

$T_e - T_r = J . p\Omega_m$ (2.18)

$\theta_r = \theta_s - \theta_m$(2.19)

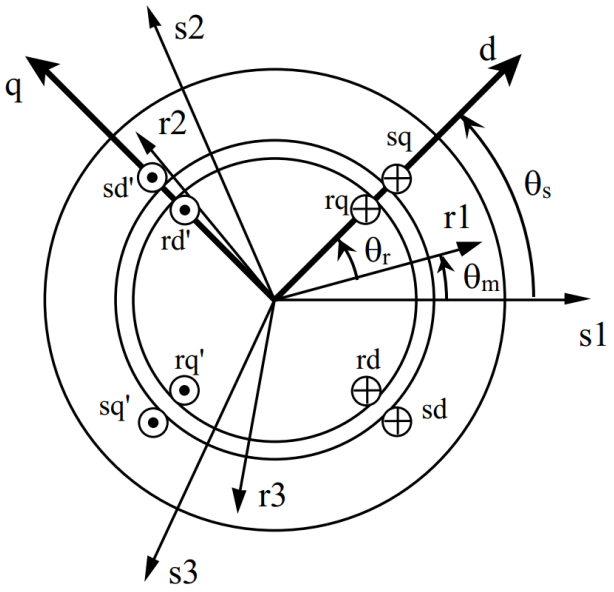

Figure 2- 2: Induction machine model with dq axes (Dezza, 2017)

A final observation is necessary to show how the use of space phasors and the adoption of a single reference frame for the stator to the rotor has eliminated the dependence of the mutual inductances by the angle θ_m between stator and rotor. In this way the parameters needed to describe the machine are constant and identical to those in the classical theory (4 or 5 parameter modeling).

2-3 Equivalent circuit of an induction machine:

Substituting the relationship fluxes/currents in the stator and rotor dynamic equations on a generic axes, but maintaining unaltered the motional terms we obtain equations (2.20) and (2.21) as following (Dezza, 2017):

$$\overline{v_s} = R_s.\overline{\imath_s} + L_s p\overline{i_s} + Mp\overline{i_r} + j\dot{\theta}_s\overline{\psi}_s \quad (2.20)$$

$$\overline{v_r} = R_r.\overline{\imath_r} + L_r p\overline{i_r} + Mp\overline{i_s} + j\dot{\theta}_r\overline{\psi}_r \quad (2.21)$$

By adding and subtracting Mpi_s in the (2.20) and in the (2.21) equation we obtain (2.22) and (2.23) subsequently:

$$\overline{v_s} = R_s.\overline{\imath_s} + L_s p\overline{i_s} + Mp\overline{i_r} + j\dot{\theta}_s\overline{\psi}_s + Mp\overline{i_s} - Mp\overline{i_s} = R_s.\overline{\imath_s} + (L_s - M)p\overline{i_s} + Mp\left(\overline{i_s} + \overline{i_r}\right) + j\dot{\theta}_s\overline{\psi}_s \quad (2.22)$$

$$\overline{v_r} = R_r.\overline{\imath_r} + L_r p\overline{i_r} + Mp\overline{i_s} + j\dot{\theta}_r\overline{\psi}_r + Mp\overline{i_r} - Mp\overline{i_r} = R_r.\overline{\imath_r} + (L_r - M)p\overline{i_r} + Mp\left(\overline{i_s} + \overline{i_r}\right) + j\dot{\theta}_r\overline{\psi}_r \quad (2.23)$$

The equations of (2.23) and (2.22) lead to an equivalent circuit of Induction machine as below in Figure 2- 3:

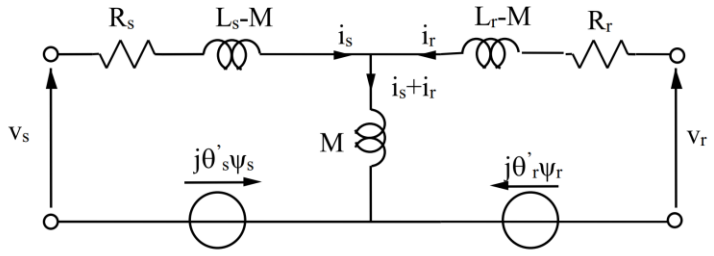

Figure 2- 3: Dynamic equivalent circuit of Induction Machine (Dezza, 2017)

In the case in which the stator voltage corresponds to a symmetrical three-phase system, the corresponding spatial phasor, obtained by applying the classic formula of equation (2.24):

$$\overline{v_s} = \sqrt{\frac{2}{3}}[v_1(t) + \overline{\alpha}.v_2(t) + \overline{\alpha^2}.v_3(t)] \quad (2.24)$$

Which equation (2.24) in fixed axes condition will be equal to equation (2.25) as below:

$$\overline{v_s} = V.e^{j\omega t} \quad (2.25)$$

Which in equation (2.25) the V is equal to the RMS value of line to line voltage of the stator. At steady state, all the quantities assume the same shape as the input voltage which leads to equations (2.26) and (2.27) as following:

$\overline{i_s} = I.e^{j(\omega t+\varphi_i)}$ (2.26)

$\overline{\psi}_s = \Psi e^{j(\omega t+\varphi_\Psi)}$ (2.27)

And the same relationships hold for rotor quantities as well. If we suppose that the rotor is short circuited ($v_r = 0$) and by introducing the slip "x" as in equation (2.28) below:

$x = \frac{\omega-\omega_m}{\omega}$ (2.28)

and recalling that on fixed axes ($\theta_s = 0$), it follows that $\dot{\theta_r} = -\omega_m$ and the rotor equation, at steady state takes the following form as equation (2.29):

$0 = R_r.\overline{i_r^s} + j\omega\overline{\psi}_r^s - j\omega_m\overline{\psi}_r^s = R_r.\overline{i_r^s} + jx\omega\overline{\psi}_r^s$ (2.29)

where the derivative of the phasor (rotating) is jω times the phasor itself. In Figure 2- 4 the equivalent circuit in stationary frame with rotor short circuited is shown:

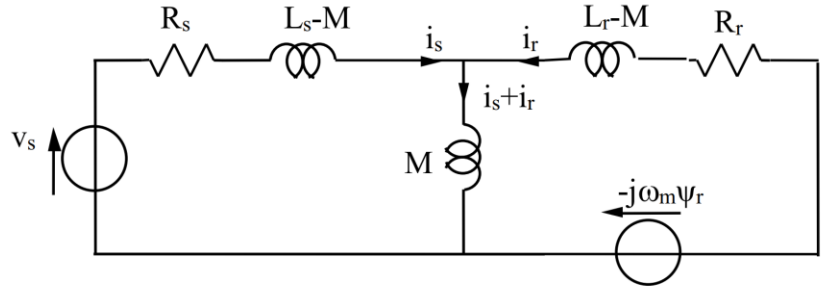

Figure 2- 4: Dynamic equivalent circuit in stationary frame with rotor short circuited (Dezza, 2017)

In equation (2.29) by dividing by x and multiplying by (1-x) we will have equation (2.30) as below:

$0 = \frac{1-x}{x}R_r.\overline{i_r^s} + j(1-x)\omega\overline{\psi}_r^s = \frac{1-x}{x}R_r.\overline{i_r^s} + j\omega_m\overline{\psi}_r^s$(2.30)

This means that the motional term (rotor side) is equivalent to a suitable value of a resistance, which is in series with the resistance of the rotor itself. In this way, the equivalent circuit scheme of the induction machine is obtained (classical theory) and can be shown as Figure 2- 5 below:

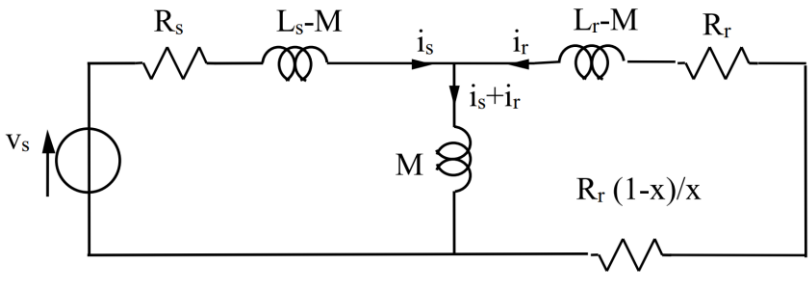

Figure 2- 5: Steady state equivalent circuit, classical theory (Dezza, 2017)

It can be mentioned that the classical theory is the special case of the dynamic theory.

2-4 Four parameters dynamic model

Suppose you connect to the rotor terminals an ideal transformer with transformation ratio k, maintaining the same quantities at the rotor terminals. The rotor quantities, at the left of the transformer, will be linked to the original variables by the relationship according to equation (2.31) as following (Dezza, 2017):

$$\frac{\overline{v_r}}{\overline{v_r'}} = \frac{\overline{\Psi_r}}{\overline{\Psi_r'}} = \frac{\overline{i_r'}}{\overline{i_r}} = \frac{1}{k} \text{ (2.31)}$$

So the dynamic equations using (2.31) takes the form of equation (2.32) and (2.33) as below:

$$\overline{v_s} = R_s.\overline{i_s} + L_s p\overline{i_s} + Mpk\overline{i_r'} + j\dot{\theta}_s\overline{\psi}_s \text{ (2.32)}$$

$$\frac{\overline{v_r'}}{k} = R_r.k\overline{i_r'} + L_r pk\overline{i_r'} + Mp\overline{i_s} + j\dot{\theta}_r\frac{\overline{\psi}_r'}{k} \text{ (2.33)}$$

Therefore, by a little simplification we will end up in (2.34) and (2.35) respectively:

$$\overline{v_s} = R_s.\overline{i_s} + L_s p\overline{i_s} + kMp\overline{i_r'} + j\dot{\theta}_s\overline{\psi}_s \text{ (2.34)}$$

$$\overline{v_r'} = k^2R_r.\overline{i_r'} + k^2L_r p\overline{i_r'} + kMp\overline{i_s} + j\dot{\theta}_r\overline{\psi}_r' \text{ (2.35)}$$

So, the mentioned equations of (2.35) and (2.34) will lead us to the equivalent circuit of as Figure 2- 6 below:

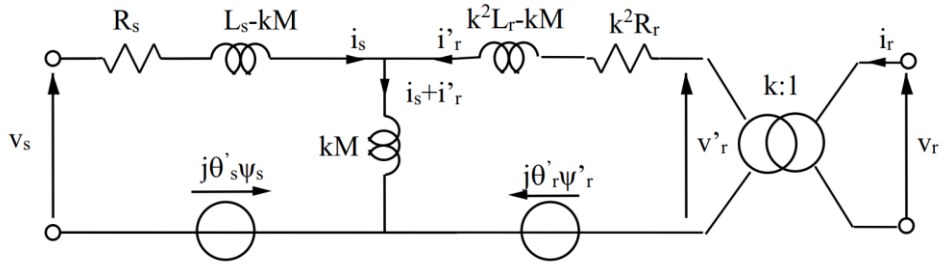

Figure 2- 6: Dynamic Circuit model of IM with addition of an ideal transformer (Dezza, 2017)

We choose k so as to cancel the series inductance of the rotor side, so we can write equation (2.36) as below:

$$k^2L_r - kM = 0 \rightarrow k = \frac{M}{L_r} \text{ (2.36)}$$

So, we obtain the four parameters equivalent circuit by introducing L_{ks} as the inductance of the short circuit according to equation (2.37) as below:

$$L_{ks} = L_s - \frac{M^2}{L_r} \text{ (2.37)}$$

Now by applying the equation (2.36) into the circuit introduced in Figure 2- 6 the circuit takes the form of Figure 2- 7 as below:

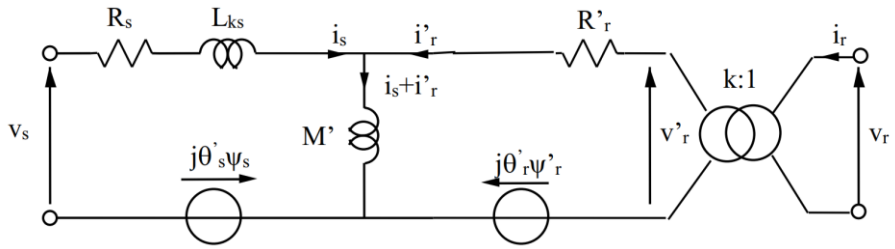

Figure 2- 7: The 4-parameter circuit taking $k = \frac{M}{L_r}$ (Dezza, 2017)

Finally, we can write the dynamic model of induction machine in four parameters format using equations of (2.38), (2.39), (2.40) and (2.41) as following:

$$\overline{v_s} = R_s.\overline{\imath_s} + p\overline{\psi}_s + j\dot{\theta}_s\overline{\psi}_s \quad (2.38)$$

$$\overline{v_r'} = \acute{R}_r.\overline{\imath_r'} + p\overline{\psi}_r' + j\dot{\theta}_r\overline{\psi}_r' \quad (2.39)$$

$$\overline{\psi}_s = L_{ks}\overline{\imath_s} + \acute{M}\left(\overline{i_s} + \overline{i_r'}\right) = L_{ks}\overline{i_s} + \overline{\psi}_r' \quad (2.40)$$

$$\overline{\psi}_r' = M(\overline{i_s} + \overline{i_r'}) \quad (2.41)$$

Eventually the torque takes the expression of equation (2.42) as below:

$$T_e = n_p.Im\left(\overline{\psi}_r\,\underline{i_r}\right) = n_p.Im\left(\overline{\psi}_r'\,\underline{\imath_r'}\right) \quad (2.42)$$

Thanks to rotor flux equation of (2.41) we can rewrite equation (2.42) as equation (2.43) like below:

$$T_e = n_p.Im\left(\overline{\psi}_r'\,\underline{\imath_r'}\right) = n_p.Im\left(\overline{\psi}_r'\,(\frac{\overline{\psi_r'}}{\acute{M}} - \underline{i_s})\right) = -n_p.Im\left(\overline{\psi}_r'\,\underline{i_s}\right)\right) = n_p.Im\left(\overline{i_s}\underline{\Psi_r'}\right) \quad (2.43)$$

On the other hand, thanks to the expression for the stator flux indicated in equation 2.40 we can write equation 2.44 as below:

$$T_e = n_p.Im\left(\overline{\psi}_r'\,\underline{\imath_r'}\right) = n_p.Im\left(\overline{i_s}\underline{\Psi_r'}\right) = n_p.Im\left(\overline{i_s}\underline{\Psi_s}\right) = n_p.\frac{1}{L_{ks}}Im(\overline{\Psi_s}\underline{\Psi_r'}) \quad (2.44)$$

The voltage v_r' is linked to the real voltage v_r, but since normally the rotor of an induction machine is short-circuited, this voltage is zero. On the other hand, the rotor current is not measurable in the case in which the rotor is constructed by a squirrel cage structure and therefore it is not necessary to know the transformation ratio k here.

From these considerations, it results that it is not necessary to know the transformation ratio k, because of no use. It is important to remember that the introduction in an ideal transformer has enabled us to introduce a degree of freedom, while maintaining the equivalence to the external effects. This degree of freedom was used, in this case, to cancel the series inductance of rotor side.

Usually when a 4-parameter model is used, the superscript ' is not shown. It should be recalled that the two rotor resistors (4-parameter and 5-parameter) are different as the two inductances M are different. Finally, Figure 2- 8 shows the circuit of the 4-parameter dynamic model of induction machine as we drove it out.

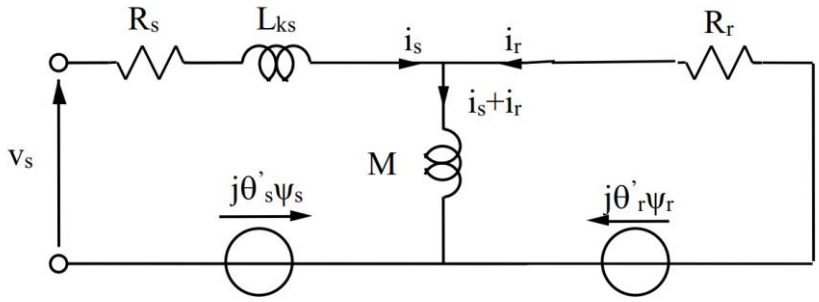

Figure 2- 8: 4-parameter equivalent circuit of induction machine with a shorted rotor (Dezza, 2017)

Chapter 3

3.Theory of vector control of Induction Motors

Introduction:

In this chapter the intention is to talk about the theory of vector control of Induction machines using the modeling introduced in previous chapter, the whole goal is to investigate how we can move from models to control theories and how we shape the Field Oriented control for Induction Motors.

3-1- Introduction to vector control:

In a practical point of view the superiority of the solution based on induction motors, from the control point of view the DC machine is still a point of reference. It can be said that the focus of the theory of AC motor control is to try to recreate the same electromechanical situation of DC machine, by introducing the control of a fictitious machine, equivalent to the original one through appropriate mathematical transformations. The strong point of the DC machine is the presence of a completely decoupled situation, from a magnetic point of view, so that you can define with precision two main things as followings:

1- a polar axis in the same direction of the magnetization flux, generated by the excitation current
2- an interporal axis on which the armature current acts.

The particularly favorable situation is the fact that we can identify two well-defined electrical ports each capable of acting on only one of the two terms that define the electromagnetic torque. In fact we have a first port, which can be called direct or *"d"*, on which the excitation current acts and a second port - quadrature or *"q"* - on which the armature current operates. The fact that the two ports must work on two magnetic circuits at 90 degrees to each other, to ensures complete decoupling between the two actions.

Now consider a machine supplied by alternating current, whether induction or permanent magnet synchronous. In this case we are dealing with three terminals single "power port", it follows that, in this case, direct and quadrature actions are not "naturally" decoupled. The fundamental purpose of vector control is to make a mathematical transformation on the variables of the machine to highlight two electrical ports on the "fictitious machine" such that each of them acts on a decoupled magnetic circuit exactly as in the DC machine. It results the following axes for us:

- **Direct axis:** fictitious circuit for the creation of the field. This circuit will be powered by a suitable current in the case of an induction machine to create the field in the gap, and will be an open circuit in the case of the machine with permanent magnets as the field is already present by the action of the magnets

- **Quadrature axis:** a fictitious circuit in which a current is proportional to the desired torque. This circuit therefore plays the same role played by the armature circuit of the DC machines.

The practical result is that we get a complete decoupling of the dynamics of the slow transient (ie those related to the magnetization of the machine) by the fast transient (ie, those related to the electromechanical torque control). Therefore, it is possible to obtain, for the faster dynamics, the maximum bandwidth allowed by the machine, as having only limited by the time constant of the electric circuit seen by the converter. With this in mind, we can find a logical scheme that can be considered valid for vector control in general, regardless of the type of machine with the following features:

- the real electrical quantities are measured
- the real electrical quantities of the machines are transformed in the fictitious machine quantities
- the regulation, based on the theory of the DC machine, is applied on the fictitious variables
- the fictitious commands, to be applied to the machine, are converted into the appropriate commands to the real machine

Then, it is easy to show what are the limitations and practical problems of this approach:

- retrieve the information needed to determine the state of the "fictitious machine"; that determines the conditions to be imposed on current and voltage of the real machine to turn them into the state variables of the "fictitious machine"

- determine the strategies to "map" the conditions of the "fictitious machine" on the real machine

3-2 Vector control of an induction machine

The vector control of an induction machine, as mentioned above, is based on a suitable choice of reference axes used by the controller so that a component of the stator current space phasor acts only on the flux, while the other one on the electromagnetic torque. In this way, the induction machine is controlled like a DC machine where we act separately on the excitation and the armature current. This approach of regulation has been known for ages, but its practical use has been achieved only in recent years due to the development of digital control systems and of power semiconductor switches. To illustrate the principle of field-oriented control, the induction machine dynamic model must be recalled.

Consider the dynamic equation of the induction machine, referred to a rotating reference frame *"d"* and *"q"* as in equations of (3.1), (3.2) and (3.3) as following (Dezza, 2017):

$$\overline{v_s} = R_s.\overline{\imath_s} + p\overline{\psi}_s + j\dot{\theta}_s\overline{\psi}_s \quad (3.1)$$

$$\overline{v_r} = 0 = R_r.\overline{\imath_r} + p\overline{\psi}_r + j\dot{\theta}_r\overline{\psi}_r \quad (3.2)$$

$$p\omega_m = \frac{n_p(T_e - T_r)}{J} \quad (3.3)$$

Where:

$T_e = np \cdot \mathrm{Im}(\overline{i_s}\underline{\Psi_s})$ electromagnetic torque,

v_s = stator voltage;

v_r = rotor voltage;

i_s = stator current;

i_r = rotor current;

Ψ_s = stator flux linkage;

Ψ_r= rotor flux linkage;

$\dot{\theta}_s$=ωs = speed of the rotating reference frame with respect to the stationary frame (fixed with stator);

ω_m = mechanical speed (electrical degree per second);

$\dot{\theta}_r$= ω_r = ω_s -ω_m = speed of the rotating reference frame with respect to the frame fixed with rotor;

R_s, R_r = stator and rotor resistances;

T_r = load torque;

n_p = number of pole pairs;

J = moment of inertia.
It also worth mentioning again the relationship between fluxes and currents in four parameters model as equations (3.4) and (3.5) below:

$$\overline{\Psi_s} = L_{ks}\overline{i_s} + \overline{\Psi_r} \text{ (3.4)}$$

$$\overline{\Psi_r} = M(\overline{i_s} + \overline{i_r}) \text{(3.5)}$$

So, from equation (3.5) we can drive equation (3.6) as below:

$$\overline{i_r} = \frac{\overline{\Psi_r}}{M} - \overline{i_s} \text{ (3.6)}$$

Where:

L_{ks} = short circuit stator inductance

M= magnetizing inductance

By substituting equations of (3.6), (3.5), (3.4) into equations (3.2), (3.3), (3.1) we can get system of equations expressed in the electrical state variable i_s and Ψ_r :

$$\overline{v_s} = R_s.\overline{i_s} + L_{ks}\, p\overline{i_s} + p\overline{\Psi_r} + j\dot{\theta}_s L_{ks}\overline{i_s} + j\dot{\theta}_s\overline{\Psi_r} \text{ (3.7)}$$

$$0 = \frac{R_r}{M}\overline{\Psi_r} - R_r.\overline{i_s} + p\overline{\Psi_r} + j\dot{\theta}_r\overline{\Psi_r} \text{ (3.8)}$$

where as you can see in equation (3.8) the rotor voltages are zero, since this is an induction machine that has a short-circuited rotor.

From the rotor equation in equation (3.8) it results that:

$$p\overline{\Psi_r} = R_r.\overline{i_s} - \frac{R_r}{M}\overline{\Psi_r} - j\dot{\theta}_r\overline{\Psi_r} \quad \text{(3.9)}$$

By substituting equation (3.9) into (3.7) we will have equation (3.10) as below:

$$\overline{v_s} = R_s.\overline{i_s} + L_{ks}\, p\overline{i_s} + \left(R_r.\overline{i_s} - \frac{R_r}{M}\overline{\Psi_r} - j\dot{\theta}_r\overline{\Psi_r}\right) + j\dot{\theta}_s L_{ks}\overline{i_s} + j\dot{\theta}_s\overline{\Psi_r} \text{ (3.10)}$$

By putting $\dot{\theta}_r = \dot{\theta}_s - \dot{\theta}_m$ in equation (3.10) we will have:

$$\overline{v_s} = (R_s + R_r).\overline{i_s} + L_{ks}\, p\overline{i_s} - \frac{R_r}{M}\overline{\Psi_r} + j\dot{\theta}_s L_{ks}\overline{i_s} + j\dot{\theta}_m\overline{\Psi_r} \text{ (3.11)}$$

Introducing the term $R_{ks} = R_s + R_r$ we will have equation (3.11) in form of equation (3.12) as below:

$$\overline{v_s} = R_{ks}.\overline{i_s} + L_{ks}\, p\overline{i_s} - \frac{R_r}{M}\overline{\Psi_r} + j\dot{\theta}_s L_{ks}\overline{i_s} + j\dot{\theta}_m\overline{\Psi_r} \quad (3.12)$$

By substituting equations of (3.4) and (3.5) inside the expression of electromechanical torque we will end up in equation (3.13) as below:

$$T_e = np \cdot \mathrm{Im}(\overline{i_s}\underline{\Psi_s}) \quad (3.13)$$

At this point we choose the axes *"d"* and *"q"* so that the direction of the *"d"* has to be always coincident with the rotor flux space phasor; with this choice the quadrature component of the rotor flux is always zero which is as equalities of (3.14) and (3.15) as below:

$$\Psi_{rd} = \Psi_r \quad (3.14)$$

$$\Psi_{rq} = 0 \quad (3.15)$$

Therefore, the equation (3.12) will take the form of equation (3.16) and (3.17) on *"d"* and *"q"* axes as below:

$$v_{sd} = R_{ks}.i_{sd} + L_{ks}\, pi_{sd} - \frac{R_r}{M}\Psi_r - \dot{\theta}_s L_{ks} i_{sq} \quad (3.16)$$

$$v_{sq} = R_{ks}.i_{sq} + L_{ks}\, pi_{sq} + \dot{\theta}_m\Psi_r + \dot{\theta}_s L_{ks} i_{sd} \quad (3.17)$$

$$p\Psi_r = R_r\, i_{sd} - \frac{R_r}{M}\,\Psi_r \quad (3.18)$$

$$0 = R_r\, i_{sq} - \dot{\theta}_r.\Psi_r \quad (3.19)$$

$$p\omega_m = \frac{n_p}{J}\,(T_e - T_r) \quad (3.20)$$

$$T_e = n_p.\Psi_r.i_{sq} \quad (3.21)$$

Where $\dot{\theta}_s$ and $\dot{\theta}_r$ means respectively the speed of the reference frame (fixed with the rotor flux) with respect to the stator and the rotor winding.

All the equations from (3.16) and (3.17) in canonical form take the following shapes respectively like equations (3.22) and (3.23):

$$pi_{sd} = \frac{1}{L_{ks}}\left[v_{sd} - R_{ks}.i_{sd} + \frac{R_r}{M}\Psi_r - \dot{\theta}_s L_{ks} i_{sq}\right] \quad (3.22)$$

$$pi_{sq} = \frac{1}{L_{ks}}\left[v_{sq} - R_{ks}.i_{sq} - \dot{\theta}_m\Psi_r - \dot{\theta}_s L_{ks} i_{sd}\right] \quad (3.23)$$

Looking at the equations (3.18) and (3.21) we see that the two components of the stator current act separately on the rotor flux and torque, because the flux depends only on the component i_{sd}while i_{sq}component acts only on the torque.

The behavior of the induction machine controlled by field orientation is thus like that of a DC machine, in this analogy, the direct component of stator current assumes the role of the excitation current and the quadrature component of the armature current. Of course, while in DC machine the two currents flow in two distinct windings, in this case the i_{sd} and i_{sq} are the components along the axes *"d"* and *"q"* of a unique single-phase system of currents.

The transition from one to the other system is obtained through the formula of the space phasor. The decoupling, obtained between the effects of the two components of the stator current, can simplify the control of the mechanical variables of the drive. In fact, if the rotor flux is kept constant with the aim to exploit well the iron, the torque is directly proportional to the quadrature component of stator current i_{sq} whose reference value can be directly derived from the desired value of the torque. A change of the torque value, however, may be obtained by a variation of flux acting on the i_{sd}, but given the presence of high time constant with which i_{sd} influences the flux ($\frac{M}{R_r}$), which is higher than the time constant between stator current and stator voltage ($\frac{L_{ks}}{R_{ks}}$), this mode is not suitable for a rapid adjustment of the torque T_e.

The outline of the induction machine in a rotor flux reference frame is represented in Figure 3- 1 as below:

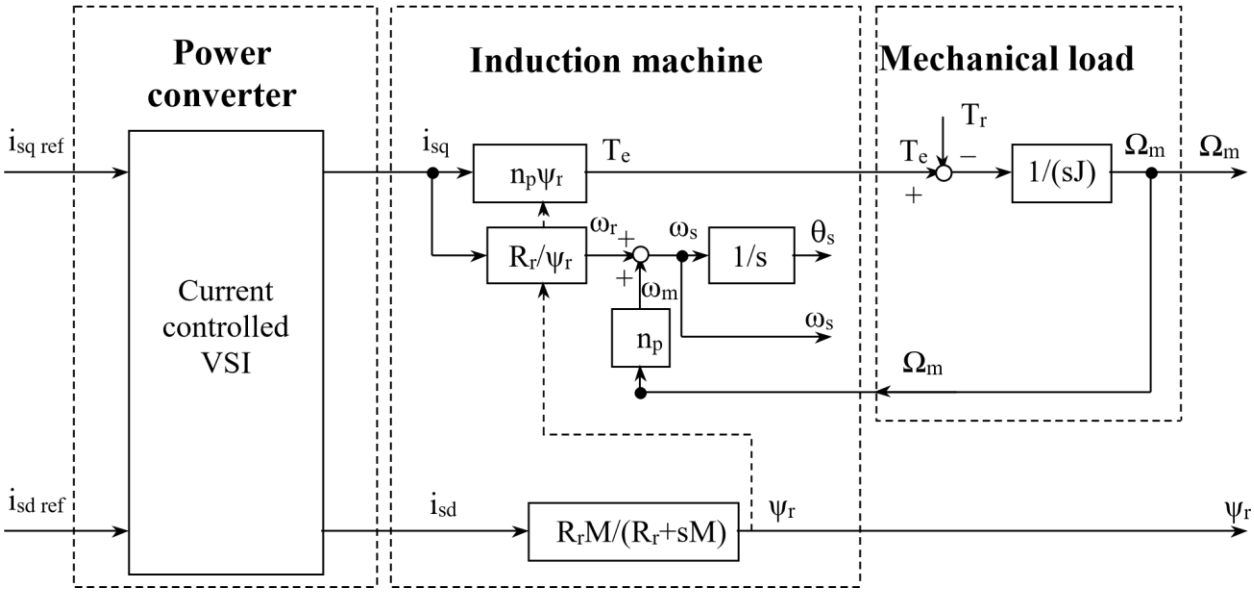

Figure 3- 1 : Diagram of the induction machine in a rotor flux reference frame, considering the stator currents as inputs (Dezza, 2017)

Finally, the schematic diagram of the drive control system is then expressed in Figure 3- 2 as following:

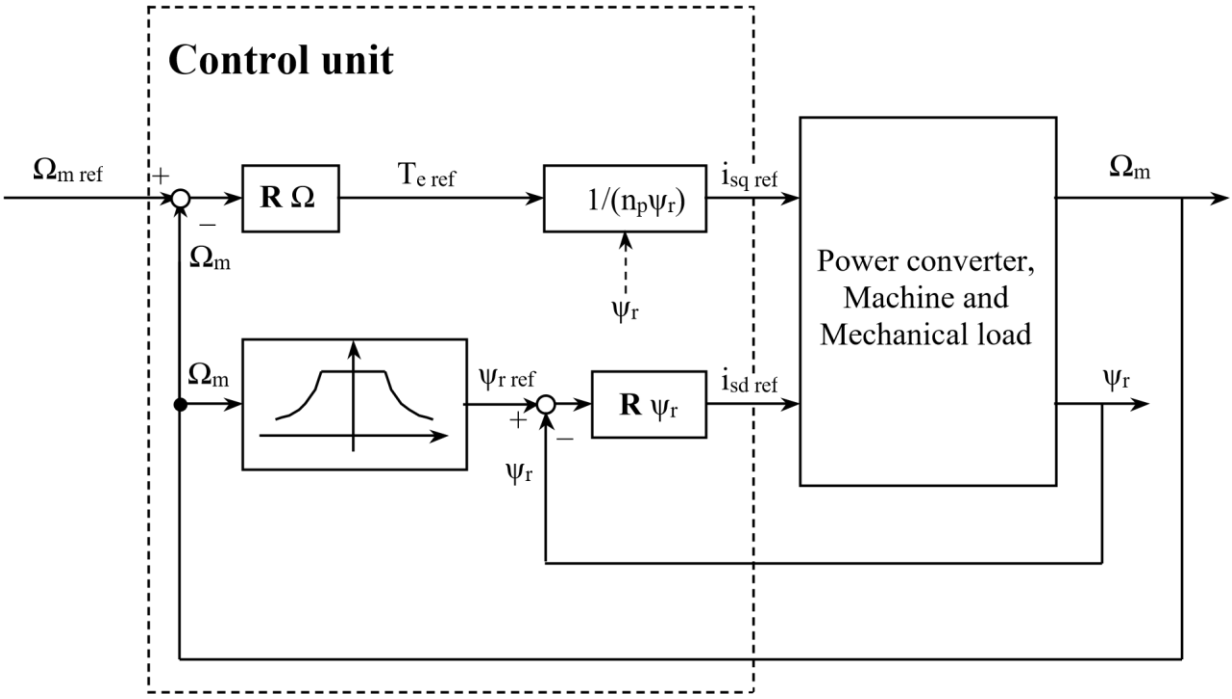

Figure 3- 2: Diagram of the control of the Induction Machine (Dezza, 2017)

In the Figure 3- 2 the block "**R** Ω" controls the mechanical speed and determines the desired value of torque while the block "**R** ψ_r" acts on the component i_{sd} so that the rotor flux is equal to its reference value Ψ_{r-ref}.

At low speeds, where the drive must provide high torque regardless of speed itself, the reference value of the flux is kept constant and equal to the maximum allowed by the magnetic circuit of the machine. At high speeds, in which the drive must work at maximum power, the reference value of flux follows the trend dictated by the operating regions. The implementation of field-oriented control can be performed according to different approaches (direct or indirect control, voltage or current controlled), but in any case, the obtained decoupling enables the synthesis of the control system of mechanical variables in a way completely independent.

At this point there are clear advantages of this type of control as below:

- direct access to the flux and the torque in an independent manner allowing field weakening action and torque and current control;
- decoupling is active both in the transient and in steady state;
- at steady state the control system works with constant quantities which makes the control less sensitive to unavoidable delays or phase displacement on the signals.

Given these advantages, the field-oriented control has some serious hurdles to overcome, mainly the acquisition of a flux signal, frequency independent, which gives module and position of the space phasor of the rotor flux. Secondly, there is a certain computational complexity mainly due to the necessary transformations of the variables which makes Field Oriented Control (FOC) challenging in terms of implementation and efficiency.

3-3 Simulation Model of the Induction Motor

After modeling the induction motor mathematically, at this stage we can prepare a modeling of the induction motor inside various types of software which here we chose MATLAB Simulink software to meet our needs. The Simulink model of the Motor can be seen in

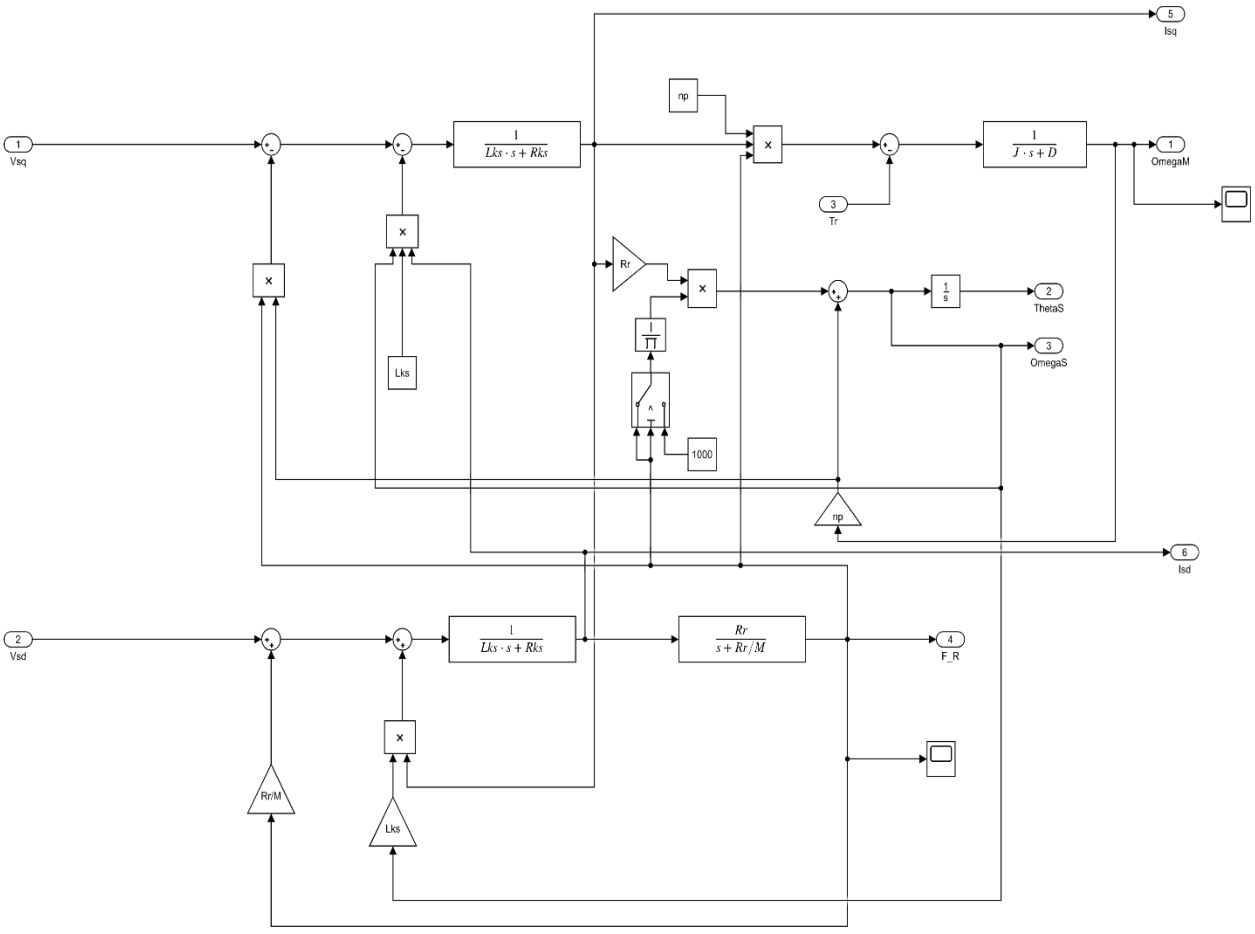

Figure 3- 3: MATLAB Simulink model of induction motor using vector control modeling

Chapter 4

4. Controller to inverter interface

Introduction:

In this section the intention is to speak about the architecture of the hardware designed for being able to proceed with testing and FOC evaluation on the real Motor of choice. In this part we will firstly introduce the main parts which are used and then we will discuss about each one in details and we end the chapter with the costume interface hardware designed in this thesis.

4-1 FPGA based controller- sbRIO-9606:

In this thesis the controller used is from a family of FPGA based platforms from National instrument company[1]. This platform is designed in a way that the users thanks to LABVIEW software and Automatic VHDL code generator and compiler can program the device with graphical user interface and block diagram design so that they don't need to write personally the VHDL codes for every single simple action.

This platform enables engineers to go faster with prototyping and testing since they can concentrate on high level tasks like controller design rather than writing a tedious code for turning on and off one pin for generating a PWM as an instance.

In Figure 4- 1 there is a view of sbRIO-9606 which is compatible with Labview software and used in this thesis as the main core of controlling.

[1] www.ni.com

Figure 4- 1: sbRIO-9606 FPGA based platform form National Instrument Company

Using this platform Labview will automatically convert the blocks made inside itself to VHDL codes thanks to Xillinx[2] libraries already built inside Labview. So, the converted codes will be compiled and synthesized over the FPGA based platform and the user will be able to test the functionalities immediately afterwards.

The other nice feature of Labview is enabling the users to graphically track their signals and feedbacks. This is a very critical feature which enormously helps the users to minimize the timing spent for debugging or tuning their controllers and devices. Also, users can capture any signal they want, and they can show them instantly or buffer them for future usages.

In Figure 4- 2 you can see a sample code done on Induction Motors using V/Hz method by National Instruments as an example, you can see graphical user interface of LABVIEW environmental and some features which it provides for developers.

[2] www.xillinx.com

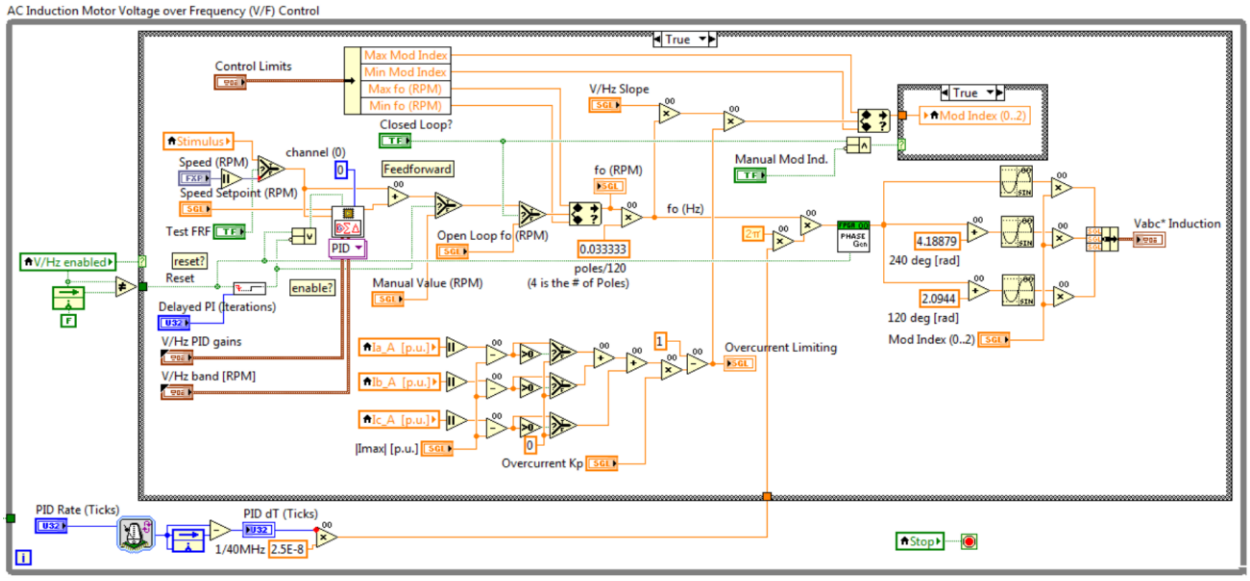

Figure 4- 2: A look inside Labview coding environment of FPGA based platforms

4-2 General Purpose Inverter Controller- NI GPIC 9683

To be able to use sbRIO-9606 for motor controlling purposes we needed to add another add-on card which is known as General purpose inverter controller or technically as NI GPIC 9683 I/O. this mezzanine card enables the sbRIO-9606 users to send and receive signals to inverters safely and properly, meaning that by using GPIC 9683 the users can send signals with higher voltage levels as well as reading signals with higher voltage levels coming from inverter or switching boards with also some insulations and safety provided inside GPIC.

As a matter of fact, The NI 9683 is a multiple analog I/O and digital I/O board for any NI Single-Board RIO device. You can connect all inputs and outputs to NI Single-Board RIO controller boards through the RIO Mezzanine Card (RMC) connector.

The NI 9683 provides connections for 16 simultaneous analog input channels with isolated ground reference; eight scanned analog input channels; eight analog output channels, all with ±30 V overvoltage protection; 28 simultaneously sampled sourcing digital input channels; 14 push-pull half-bridge digital output channels; 24 sinking digital output channels; four relay control digital output channels; and 32 LVTTL digital I/O channels.

In Figure 4- 3 there is a view of the GPIC 9683:

Figure 4- 3: Top view of GPIC 9683 mezzanine board

GPIC 9683 can be easily mounted on the back of sbRIO 9606 like Figure 4- 4 as below:

Figure 4- 4: Stacked sbRIO 9606 on GPIC 9683 mezzanine card

Now after stacking the GPIC board on sbRIO 9606 we are ready to introduce our inverter board and the methods of connection and signaling between these structures.

4-3 Inverter Board MC1H 3-phase:

From the beginning of the thesis we've decided to use a specific type of 3-phase inverter board for our switching purposes and connection to the IM from Microchip[3] Company. The inverter we chose is MC1H which is a general purpose 3-phase switching package for three phase systems, in Figure 4- 5 there is view of MC1H connected to a generic microcontroller-based control board.

Figure 4- 5 : MC1H 3-Phase inverter connected to a microcontroller-based Platform

As you can see in Figure 4- 5 the only way to access the MC1H boar is through a female D-sub 37 connector.

[3] www.microchip.com

As a matter of fact, The Microchip MC1H 3-Phase High Voltage Power Module is intended to aid the user in the rapid evaluation and development of a wide variety of motor control applications using the dsPIC microcontroller. The design of the system includes Microchip analog components, as well as a PIC microcontroller used to provide isolated voltage feedback.

The rated continuous output current from the inverter is 2.5A (RMS). This allows up to approximately 0.8 kVA output when running from a 208V to 230V single-phase input voltage in a maximum 30°C (85F) ambient temperature environment. Thus, the system is ideally suited to running a standard 3-Phase Induction Motor of up to 0.55 kW (0.75 HP) rating or an industrial servomotor of slightly higher rating. The power module can drive other types of motors and electrical loads that do not exceed the maximum power limit and are predominantly inductive. Furthermore, single-phase loads can be driven using 1 or 2 of the inverter outputs. The unit is capable of operating from any AC voltage up to a maximum of 265V. Operation at voltages beneath 208V requires that the output power is reduced owing to inverter output and AC input stage current limits.

As it mentioned above the MC1H provides us the possibility to run and control the AC induction machines up the mentioned power limits. Since MC1H is basically designed for specific type of microcontroller-based platforms, we had to design our own interface for connecting our FPGA-based platform to MC1H, to do so, we made our own printed circuit board (PCB) for interfacing GPIC 9683 board to MC1H because there was no possibility to connect directly MC1H to GPIC9683 with direct connection and they were not designed for each other.

4-4 GPIC 9683 Interface to MC1H inverter:

To pass through this phase firstly we have identified what features of MC1H are critical so that we can make sure that our interface board handles them correctly. Generally speaking for controlling and induction machine in field-oriented control manner we need the following possibilities:

1. possibility of controlling 3 different half bridges so that there would be the possibility of controlling 3 phases separately.
2. Being able to read the value of Current floating through each phase of the motor
3. Being able to measure the voltage of each phase

For the first feature mentioned above there were 6 different inputs for sending high-side and low-side control signals to half-bridges inside the MC1H. So, we needed to make sure that we are properly connecting these signals to appropriate outputs from GPIC with proper voltage levels.

According to GPIC reference designthe half-bridge outputs from NI 9683 were used which are designed for sinking and sourcing PWM signals from +5 to +30 volts (Instruments N.). Based on the capacitive load the maximum switching frequency of the half-bridge outputs is recommended as Figure 4- 6 as below:

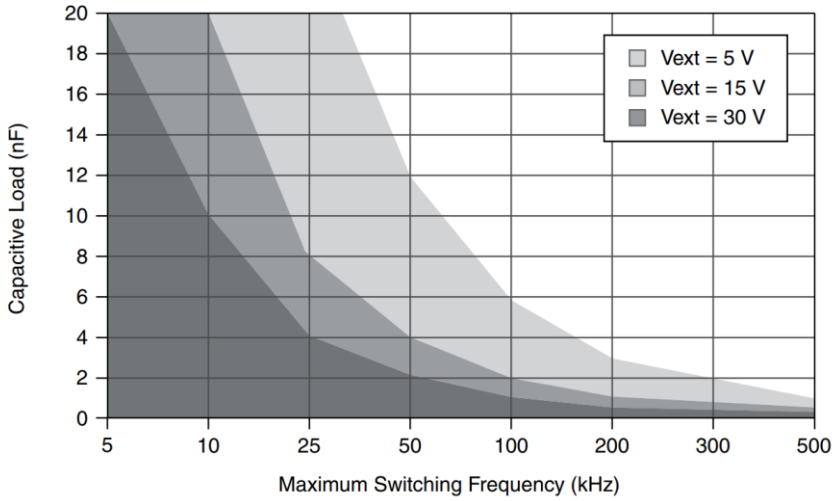

Figure 4- 6: Capacitive Load vs. Maximum Switching Frequency(kHz) in GPIC 9683 from Half-bridge output section (Instruments N.)

In our design we chose our switching frequency 12kHz with Vext= +5 V which is a suitable value according to Figure 4- 6.

For the second feature regarding the Current reading of motor phases, the MC1H has two main possibilities:

1. Reading the current through insolated hall sensors
2. Reading the current through non-isolated part from shunt resistors mounted across the half-bridges

For sake of safety and simplicity, the first method to measure the current of phases was used. Inside MC1H inverter in order to provide isolated current feedback, Hall effect closed loop DC current transducers (LEM LTS 6-NP) devices have been installed. These devices have the following characteristics:

- Single 5V supply with 2.5V (nominal) representing 0A
- Bipolar current sensing with ±19.2A given by 4.5V and 0.5V respectively with a single turn through the transducer.
- >200 kHz bandwidth
- 3 kV AC isolation

The current sensors are used in bi-directional mode, the output voltage of the current sensors are between $\pm 4V$ and the gain of the sensor is 2.4V per Amps.

It worth mentioning that MC1H has only two insulated current sensors and considering our induction machine as symmetrical machine the third phase current can be derived from the sum of the other two phases current.

These outputs from the current sensors were routed to the analog inputs of the GPIC 9683 which are known as Simultaneous Analog Inputs. In GPIC there are 16 channels for these kinds of analog inputs with 12 bits of ADC resolution and input range of $\pm 5V$ or $\pm 10V$ with sampling rate of 100kS/s at maximum.

For the last feature since the controller was made using encoder as the position sensor and the rotor flux observer input in a way that it doesn't need the phase voltages feedbacks, so in practice the voltage feedbacks for the MC1H was not used even though they were routed to the Analog inputs of the GPIC 9683 for further usages.

So, the last step was to make a PCB for the signals from MC1H to GPIC 9683 board to be able to have all the necessary signals and commands for running the motor with the switching board directly from Labview.

4-5 Designing PCB Interface from GPIC 9683 to MC1H

For sake of design of the interface PCB firstly we needed to understand where the positioning of the signals in the D-sub connecter of the MC1H are, so that we could rout them accordingly to proper pins of the GPIC 9683.

By evaluating the MC1H datasheet we can find the Figure 4- 7 which represents the positioning of the signals for MC1H in and out.

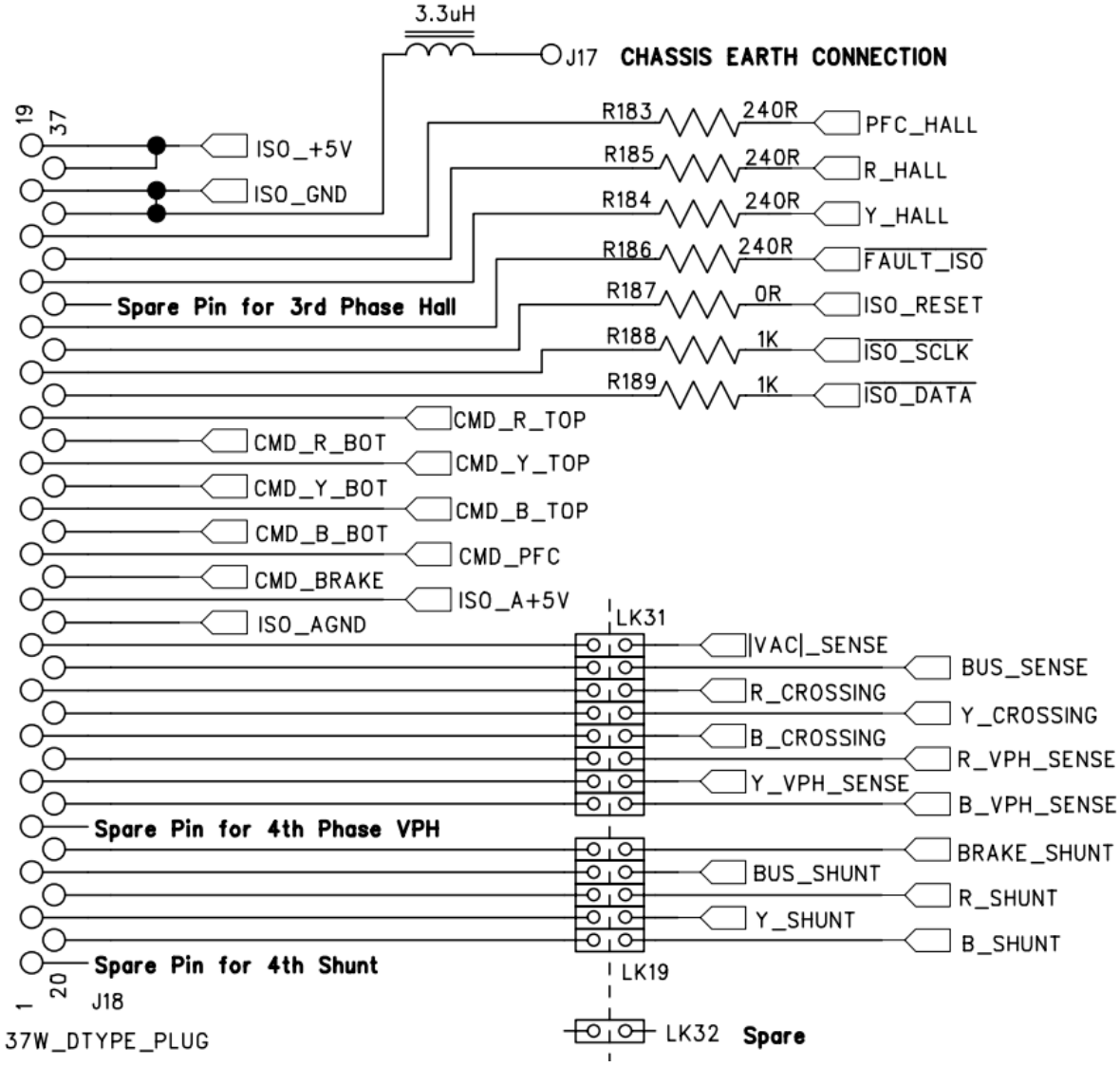

Figure 4- 7: Schematics and positioning of Signals of MC1H In and Out for Commanding and Feedbacking

Based on scheme shown in Figure 4- 7 there are 6 signals of PWM connected from half bridge output of the GPIC to MC1H through the pins written above CMD-X-Y where X stands for name of the phase (R, Y, B) and Y stands for high-side or low-side Mosfet of the inverter half-bridge legs which are in fact are the inputs of the gate drivers connected to the Mosftes.

The current feedbacks are taken from R-Hall and Y-Hall which are the outputs of the isolated Hall current sensors inside MC1H.

So, based on this scheme and other requirements of the GPIC 9683 board and the positioning of the connector pins on GPIC 9683 another stackable PCB has been designed to be mounted on the back of GPIC and being able to be connected directly to MC1H to avoid any extra wiring and introduced unwanted noises for having the maximum quality and durability during testing phases.

In the following the schematics for different parts of the circuit are shown, the very first schematic as in is for the connection of the DSUB connector from the interface to MC1H.

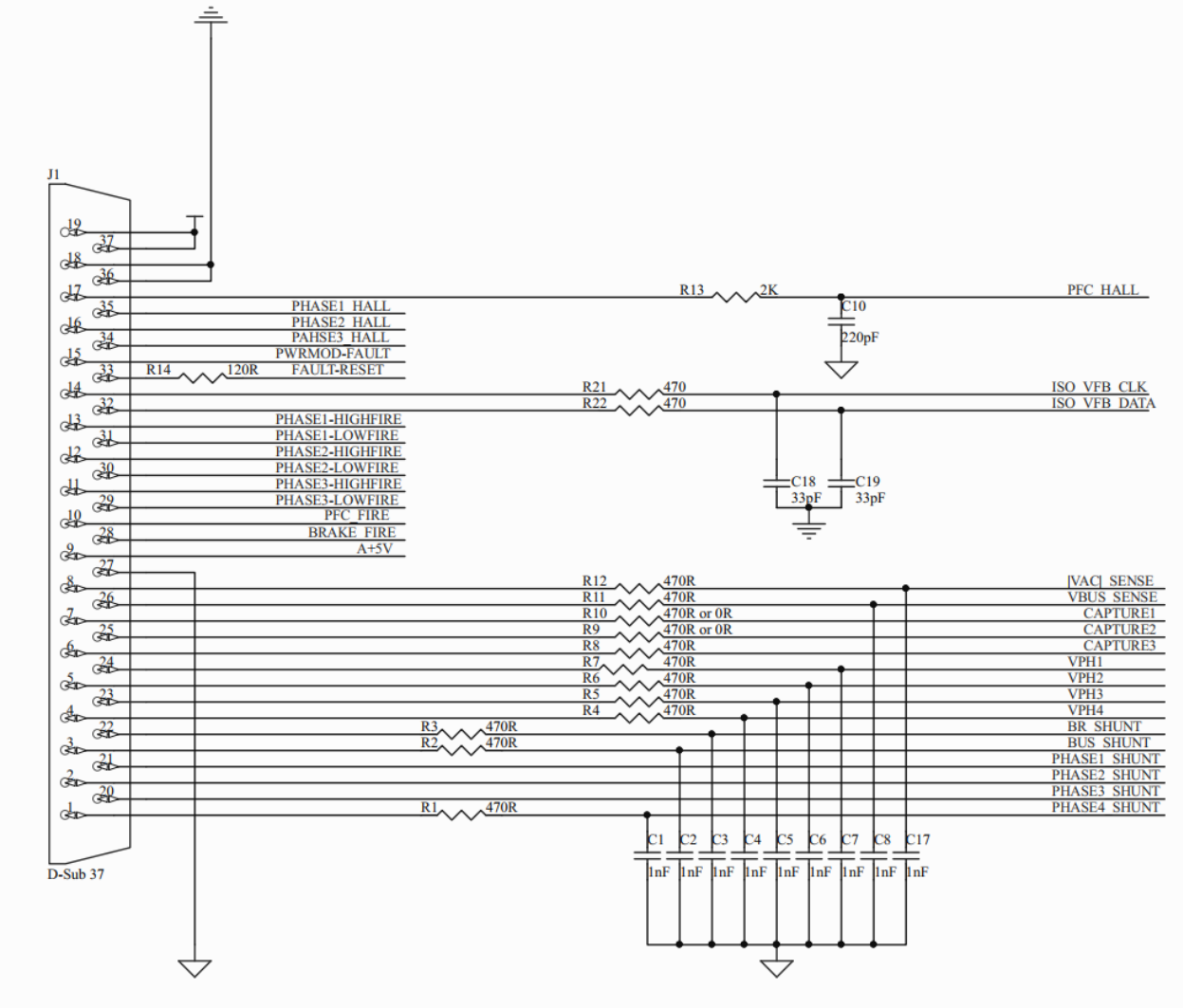

Figure 4- 8: DSUB connector connection schematics from interface board to MC1H

The other part was dedicated for voltage regulation for providing acceptable voltage level to the PWM generator of GPIC board and also the analog 5 volts input of the MC1H for internal biases of MC1H as shown in Figure 4- 9 below:

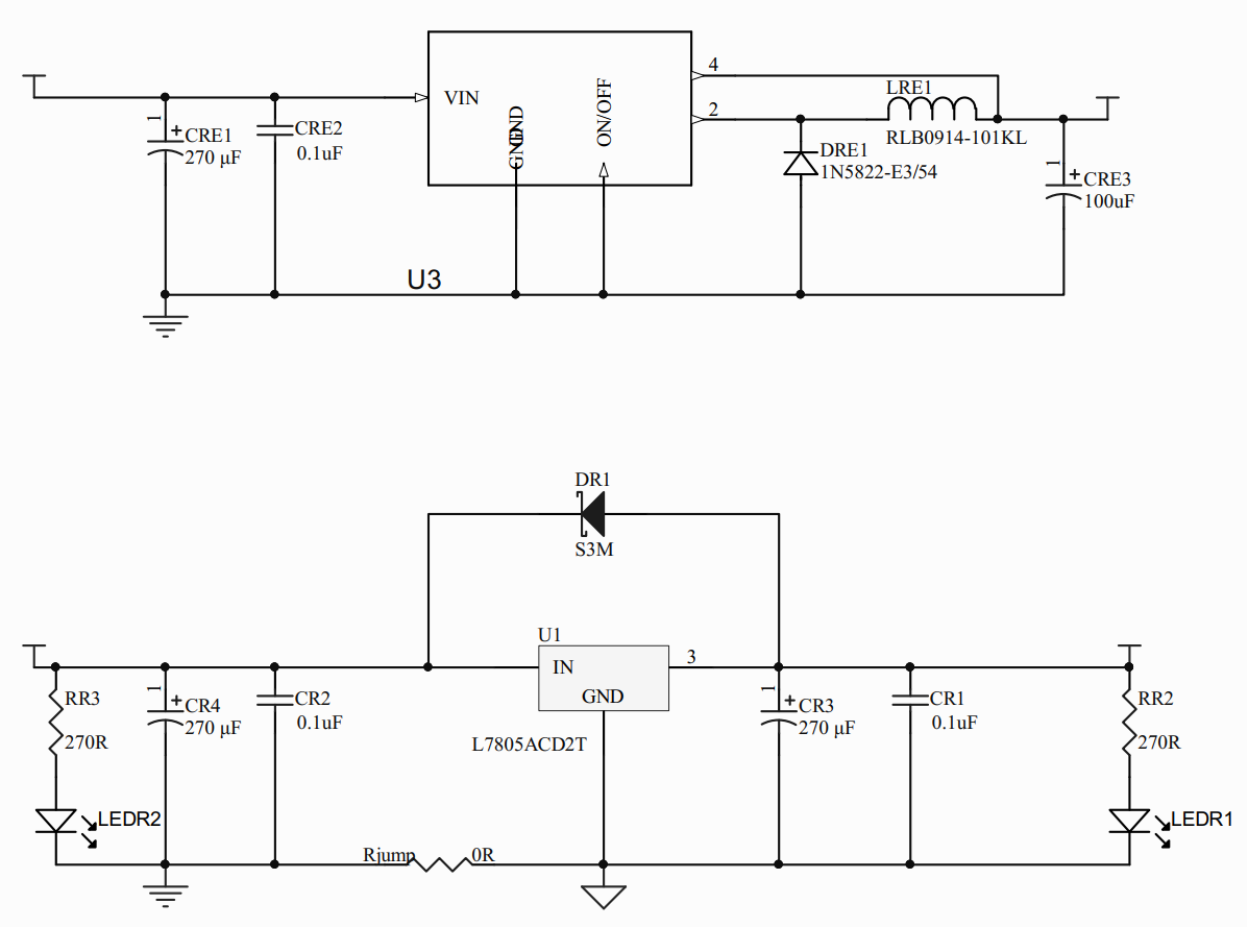

Figure 4- 9: voltage regulation circuity for the interface to MC1H connection

After this there are some header pins which the bridges between signals from MC1H to the interface are or vice versa which are shown in Figure 4- 10 below:

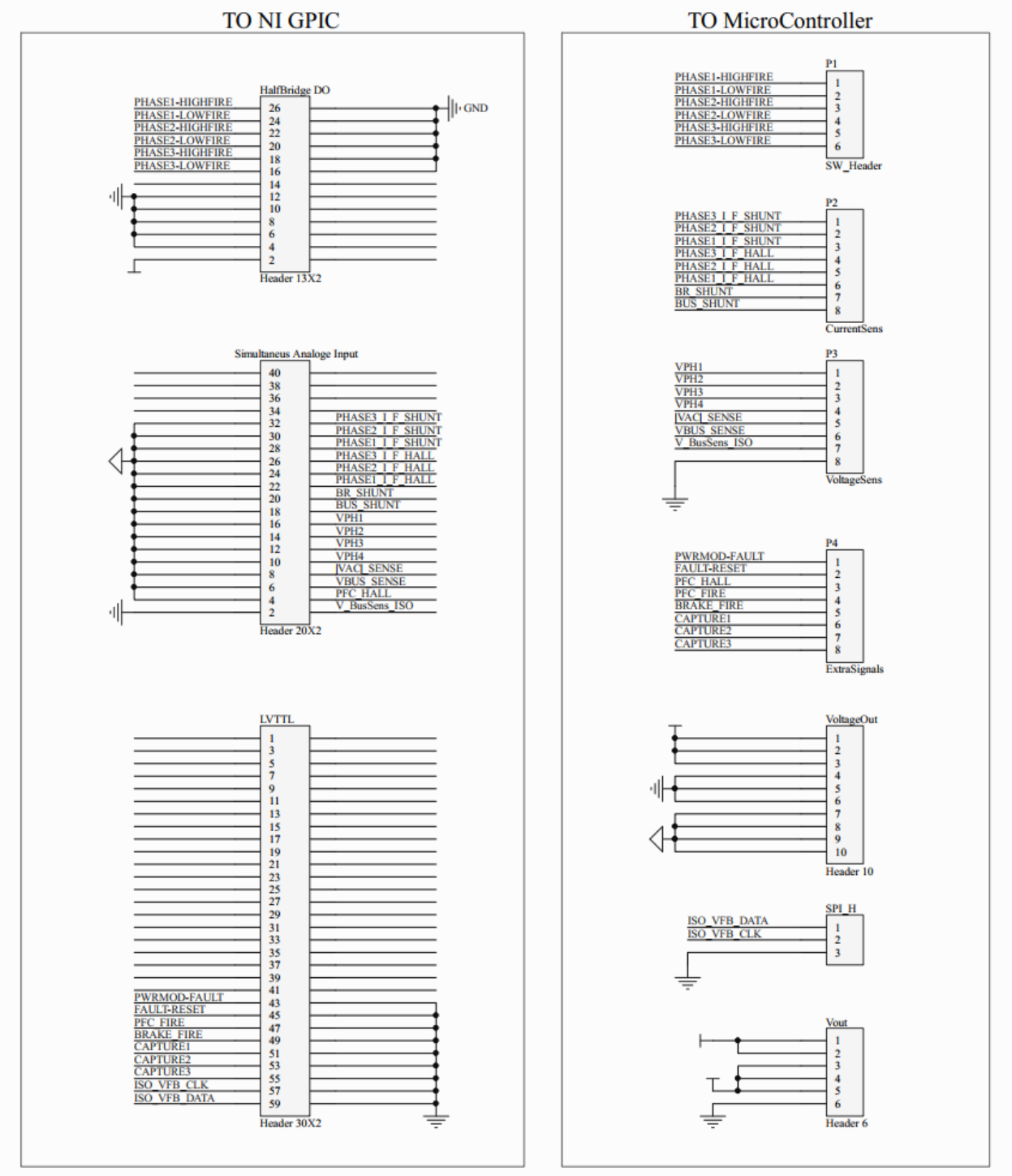

Figure 4- 10: Header Pin connection from/To interface and MC1H

After this step, The PCB was designed as you can see in Figure 4- 11 using Altium[4] software:

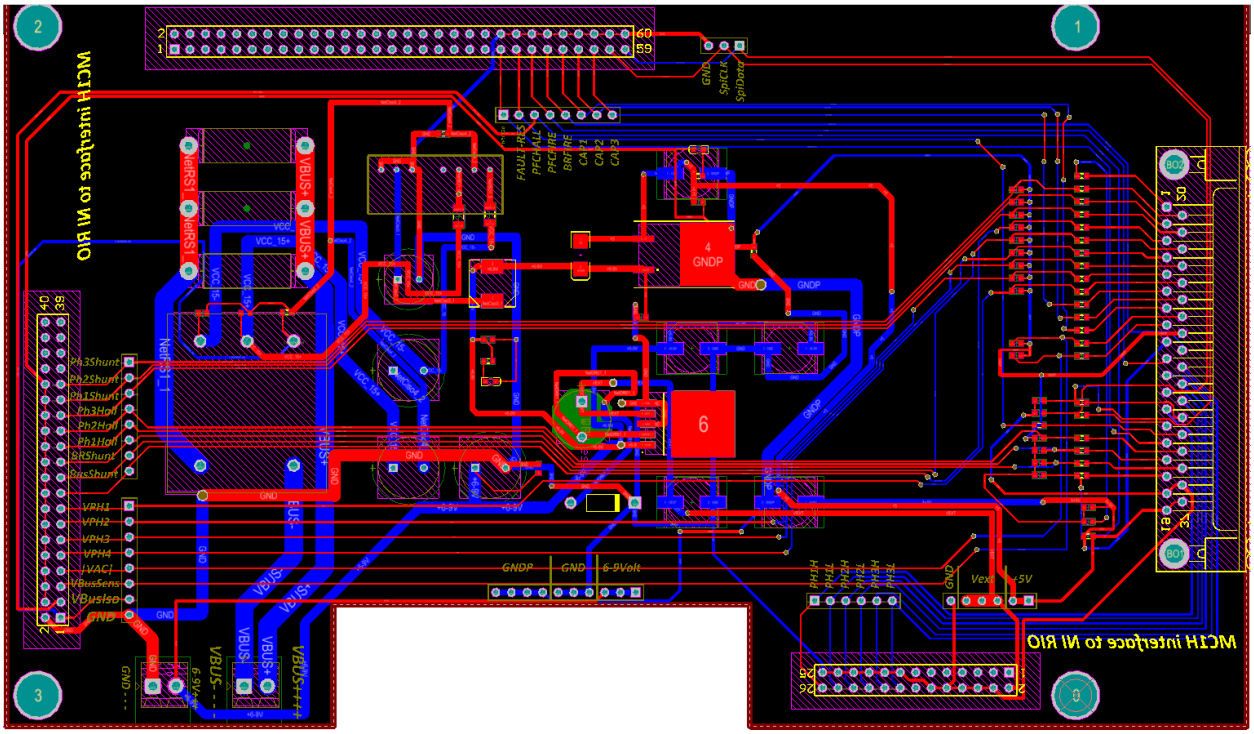

Figure 4- 11: Layout of Interface PCB designed for connecting MC1H to GPIC 9683

Also, in Figure 4- 12 there is the 3D view of the top part of the designed PCB with also a bottom view of the PCB shown in Figure 4- 13 respectively as below:

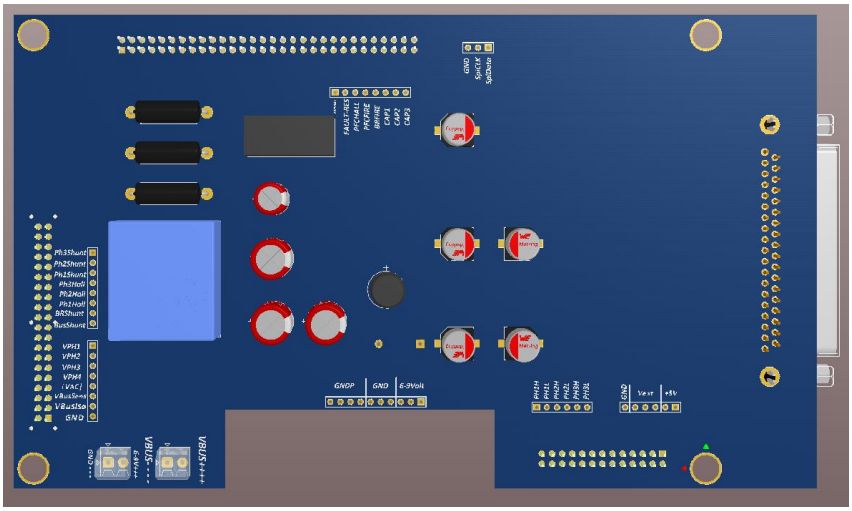

Figure 4- 12: Top 3D view of the Interface board to MC1H from GPIC 9683

[4] https://www.altium.com/

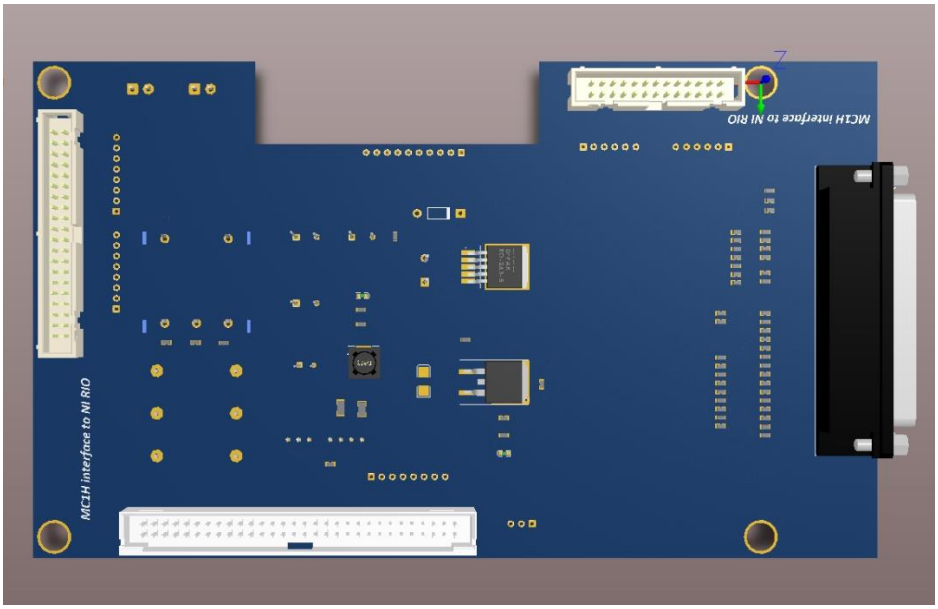

Figure 4- 13: Bottom 3D view of the interface board from MC1H to GPIC 9683

Finally, in Figure 4- 14 all the boards are stacked up on each other with mounted components on the interface side.

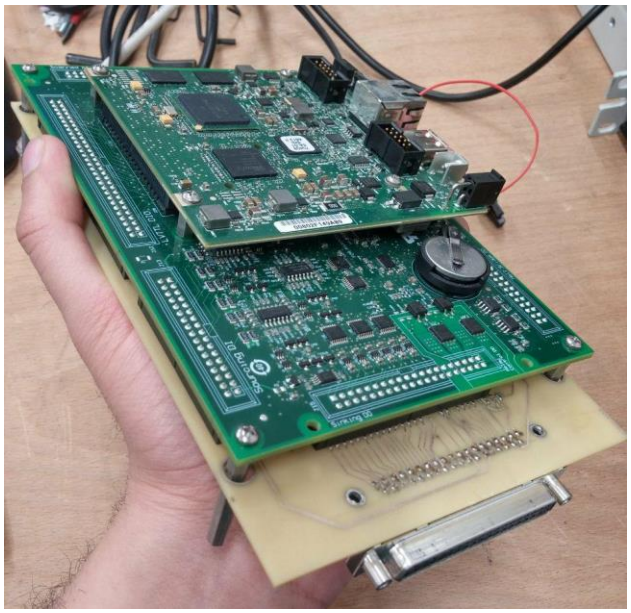

Figure 4- 14: Stacked sbRIO 9606, GPIC 9683 and Interface board together

Chapter 5

5. Motor Parameter Identification

Introduction:

After introducing the field-oriented control method and explaining the hardware architecture used for controlling IM now we want to focus on the methods of identifying the motor parameters which are crucial for designing the controllers of current, flux and speed.

To simplify, the whole architecture of the hardware can be summarized as Figure 5- 1 as below:

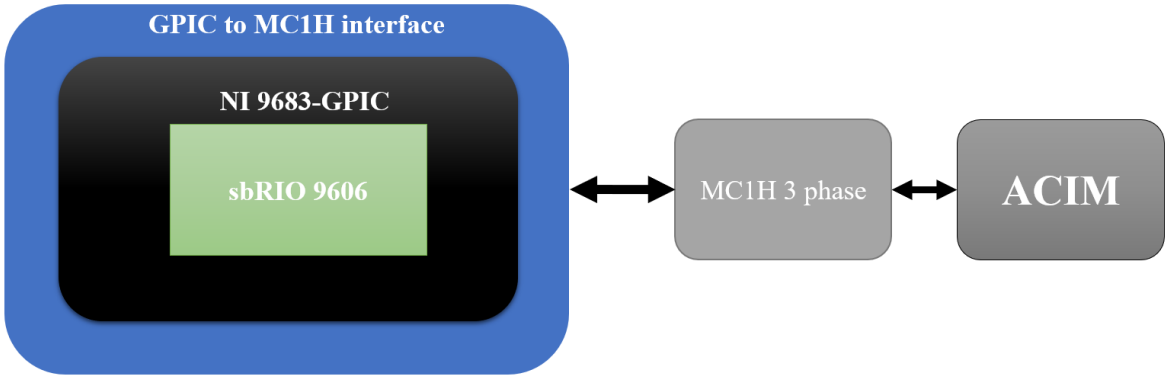

Figure 5- 1: Hardware Architecture used for controlling IM using FOC technique

In order to be able to control an induction machine there is a need to identify the control parameters of motor. These parameters are used in FOC as it mentioned in chapter 2 in order to be able to close the loops of currents, speed and flux.

Generally speaking there are two sets of parameters which should be identified for IM known as Mechanical and Electrical parameters. In the following for each part we will deeply elaborate the theory and then experimental results achieved.

5-1 Induction Motor under Test

The Motor used for testing purposes in this thesis is shown in Figure 5- 2 below:

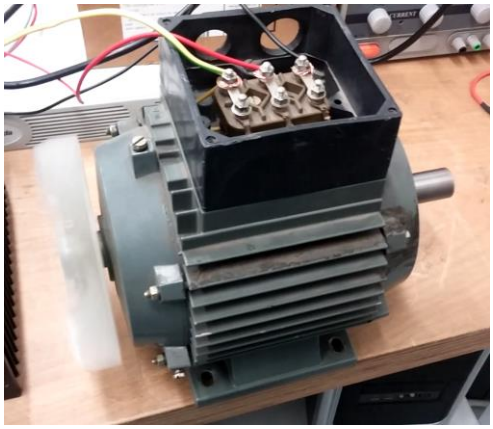

Figure 5- 2: Induction Motor Under Test

To start the identification of the motor parameters the main characteristics of the IM under test are taken from the label attached to the motor as Table 5- 1 below:

Power [KW]	Phase Voltage [V]	Phase Current [A]	Nominal Frequency[Hz]	Cos (ø)- Power Factor
1.1	220/380 Star/Delta	5/2.9	50	0.77

Table 5- 1: Nominal Characteristics of IM used from the manufacturer

For our testing the IM was left in star connection.

5-2 Identification of Mechanical Parameters of IM

The first test we will proceed on is identifying the mechanical parameters of IM, this generally means the value of the motor shaft Inertia known as "J". Identification of Inertia of the shaft of the motor is necessary for closing the speed loop in FOC.

5-2-1 Deceleration Test

Deceleration test is the very first step for identification of Inertia of the motor. The present experiment consists, in first running the motor, being load less (T_L = 0). Then, at some time the stator supply voltage is suddenly turned off. The time when the voltage is turned off is considered as t = 0, for the present experiment, and the motor speed at that moment is denoted ω_{m0}. Therefore, the stator currents and the electromagnetic torque T_{em} vanishes at t = 0 and the mechanical equation for t > 0 will be as equation (5.1) below (Giri, 2013):

$$J\frac{d_{\omega m}}{d_t} = -F\omega_m - T_d \quad (5.1)$$

Clearly, equation (5.1) is linear in the quantities $(\frac{T_d}{J})$ and $(\frac{F}{J})$. Then, the least squares estimator can be resorted to get estimates of these quantities using a sufficiently large sample of measurements (t,ω_{m0}) (Fidan) (Leonard, 2001).

Presently, a quite simpler alternative necessitating only four speed measures is presented, getting benefit of the fact that speed measurements are weakly noisy. The key point is that the solution of the first order equation (5.1) is easily found. Specifically, one has equation (5.2):

$$\omega_m(t) = \left(\omega_{m0} + \frac{T_d}{F}\right)e^{\frac{-t}{\tau_m}} - \frac{T_d}{F} \quad (5.2)$$

using the fact that $\omega_m(0) = \omega_{m0}$, where $\tau_m = \frac{F}{J}$ is the mechanical time constant. The equation (5.2) shows that the rotor speed is exponentially decaying and vanishes at a finite stop time, say t_s. Figure 5- 3 shows the decaying speed curve obtained when our motor submitted to the deceleration test.

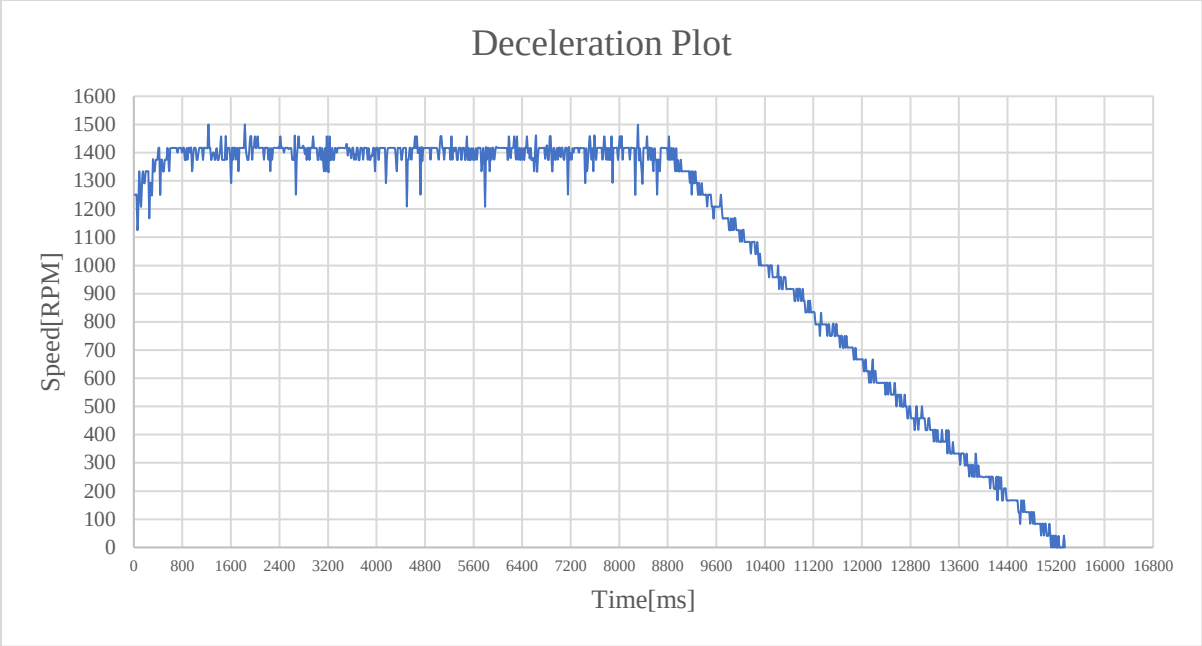

Figure 5- 3: Declaration Curve for IM under test

To achieve this curve the Motor was ran open-loop with nominal frequency of the motor at 50Hz meaning a rotating field of the 50 Hz was generated around the stator so that as you can see the motor reached to the nominal speed of 1400 RPM. Then at a certain time the voltage across the stator will be cut and the motor left to reach the zero speed. The time was recoded and as it's visible in Figure 5- 3 it's around 6000 milli-seconds.

From such a curve, let us get two arbitrary time-speed couples (t_1, ω_{m1}) and (t_2, ω_{m2}) with $t_2 = 2t_1$. Writing equation (5.2) for $t = t_1$ and $t = t_2$, one gets the following equations:

$$\omega_{m1}(t) = \left(\omega_{m0} + \frac{T_d}{F}\right) e^{\frac{-t_1}{\tau_m}} - \frac{T_d}{F} \quad (5.3)$$

$$\omega_{m2}(t) = \left(\omega_{m0} + \frac{T_d}{F}\right) e^{\frac{-2t_1}{\tau_m}} - \frac{T_d}{F} \quad (5.4)$$

Subtracting side-to-side equation (5.3) from equation (5.4), it gives equation (5.5) as following:

$$\omega_{m2} - \omega_{m1} = \left(\omega_{m0} + \frac{T_d}{F}\right) e^{\frac{-t_1}{\tau_m}} \left(e^{\frac{-t_1}{\tau_m}} - 1\right) \quad (5.5)$$

Also, one can get from (5.3) the equation (5.6) as below:

$$\omega_{m1} - \omega_{m0} = \left(\omega_{m0} + \frac{T_d}{F}\right) \left(e^{\frac{-t_1}{\tau_m}} - 1\right) \quad (5.6)$$

Dividing side-to-side equations (5.5) and (5.6), one obtains an equation where the only unknown is τ_m. Solving that equation, one gets the following expression that determines $\tau_m = \frac{F}{J}$,from available information one can drive equation (5.7) as below:

$$\tau_m = \frac{t_1}{\ln(\frac{\omega_{m2}-\omega_{m1}}{\omega_{m1}-\omega_{m0}})} \text{ (5.7)}$$

A second useful expression is immediately obtained from equation (5.2) using the fact that the speed $\omega_m(t)$ vanishes (i.e., the motor stops turning) at $t = t_s$, which is a known time. Doing so, one gets an equation where the only unknown quantity is $\frac{T_d}{F}$. Solving that equation with respect to the unknown, one gets the equation (5.8) as below (Giri, 2013):

$$\frac{T_d}{F} = \frac{\omega_{m0}}{\frac{t_s}{e^{\tau_m}-1}} \text{ (5.8)}$$

So now by considering the Figure 5- 3, our data will be as table 5-2 below:

$\boldsymbol{t_1}$[ms]	$\boldsymbol{t_s}$[ms]	$\boldsymbol{\omega_{m0}}$[RPM]	$\boldsymbol{\omega_{m1}}$[RPM]	$\boldsymbol{\omega_{m2}}$[RPM]
1015	15015	1400	1084	833

Table 5- 2: The main data recorded on deceleration test

Then, applying equations (5.7) and (5.8) and using Table 5- 2 data, one obtains the following values of τ_m and $\frac{T_d}{F}$ as below:

$$\tau_m = 3.78\ [s] \text{ (5.9)}$$

$$\frac{T_d}{F} = 36.1\ \left[\frac{rad}{s}\right] \text{ (5.10)}$$

Where $\tau_m = \frac{F}{J}$, The point is that the two above equations involve three unknowns, namely F, J, and T_d. That is, a supplementary test is needed to determine the mechanical parameters.

5-2-2 Progressive Starting Test

The dry torque T_d can be determined using a progressive start-up experiment. Accordingly, the stator voltage being initially null (stationary machine) is very slowly increased until the motor only just starts moving (Giri, 2013). At this time, the motor speed is still quasi null. Then, it follows from equation (5.11) which is the mechanical equation of the induction machine that describes the motion of the rotor carrying a load. In this condition the dry torque T_d is quasi equal to the electromagnetic couple T_{em}, which we know is given by the expression (5.12).

$$\frac{d_{\omega m}}{dt} = -\frac{F}{J}\omega_m + \frac{T_{em}}{J} - \frac{T_L}{J} - \frac{T_d}{J} \quad (5.11)$$

$$T_{em} = p\frac{p_m}{\omega_s} \quad (5.12)$$

Where in (5.12) the p_m designated the mechanical power developed by the motor.

So, It follows the following expression of the dry torque as equation (5.13) below:

$$T_d = p\frac{p_{as}}{\omega_s} \quad (5.13)$$

Where in (5.13) the term P_{as} is the absorbed power the moment when the motor only just begins turning and ω_s is the grid voltage frequency. Note that all losses (copper, rotational) are null because the rotor speed is quasi zero and the stator voltage is very small.

In our case to find the starting voltage for turning the motor the system was left at open loop at 50 Hz but with zero voltage over the switching module. Then by increasing the voltage of power supply gradually the value of the voltage which motor starts to rotate was found.

According to our measurement the motor starts rotating with line-to-line voltage of 42 volts and the phase current of 1.2 Amps peak to peak.

$$V_{L-L-rms} = 42\ [V] \quad (5.14)$$

$$I_{ph-peak\ to\ peak} = 1.2\ [A] \quad (5.15)$$

$$I_{ph-rms} = \frac{\frac{I_{ph-peak\ to\ peak}}{2}}{\sqrt{2}} = 0.4242\ [\text{A}] \quad (5.16)$$

So according to (5.14) and (5.16) the absorbed power by motor based on equation (5.17) will become as following:

$$P_{as} = \sqrt{3}\, I_{ph-rms}\, V_{L-L-rms}\, Cos(\emptyset) \quad (5.17)$$

Using the motor parameters mentioned by the company according to Table 5- 1 we calculated the absorbed power as (5.18) below:

$$P_{as} = 23.52\ [W] \quad (5.18)$$

So now we can calculate the dry torque value from (5.13) as below keeping into account that the $\omega_s = 50\ [Hz] = 314.159\ [rad/s]$:

$T_d = \frac{23.52}{314.159} = 0.0749$ (5.19)

So from (5.10) we can now calculate the value of F as (5.20) below:

$F = \frac{T_d}{\tau_m} = 0.0021$ (5.20)

So finally, J will be calculated using equation (5.21) knowing that $\tau_m = \frac{F}{J}$:

$J = \frac{F}{\tau_m}$ = 0.00056 [Kg.m^2] (5.21)

Now all the mechanical parameters necessary for controlling of induction machine are found, In the next part we will proceed on with identification of electrical parameters to complete the identification phase of the IM under test.

5-3 Identification of Electrical Parameters of Induction Machine

Traditionally, the steady-state model of a three-phase induction motor is represented by the per phase equivalent circuit (Giri, 2013). The steady state per phase equivalent circuit as viewed from the stator side is represented in Figure 5- 4 below:

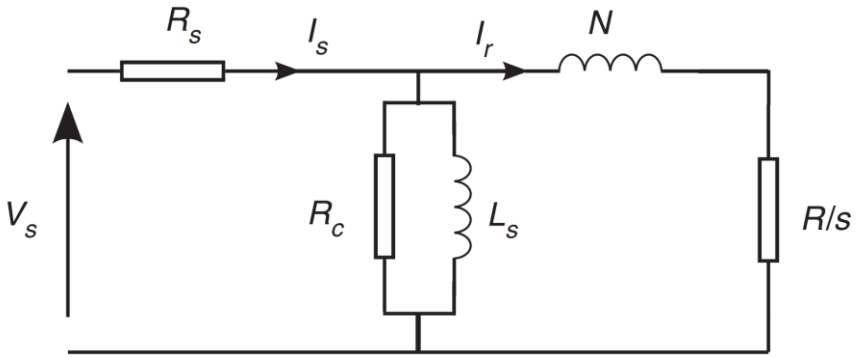

Figure 5- 4: Per Phase Equivalent Circuit, viewed from the stator of the 3-phase induction machine (Giri, 2013)

In Figure 5- 4 , V_s is the stator phase voltage; R_s is stator resistance and R is the rotor winding resistance brought to the stator side; N is the equivalent inductance of both stator and rotor leakage brought to the rotor side. L_s is the magnetizing and stator inductance; R_c is the equivalent resistance for core loss; and s is the slip.

The equivalent circuit parameters for an induction motor can be determined using specific tests on the motor. The tests are quite similar to those performed on transformers.

5-3-1 Stator Winding Resistance Measurements

The resistance of the stator winding is measured at DC, and the measurement is preferably performed after the motor temperature has reached its nominal value (Giri, 2013). This experimental test gives, for the induction motor characterized by Table 5- 1 the following value:

R_s = 13 Ω (5.22)

The measurement here is done using a laboratory multimeter by measuring the resistance of one phase with respect to neutral point in star connection as shown in Figure 5- 5 below:

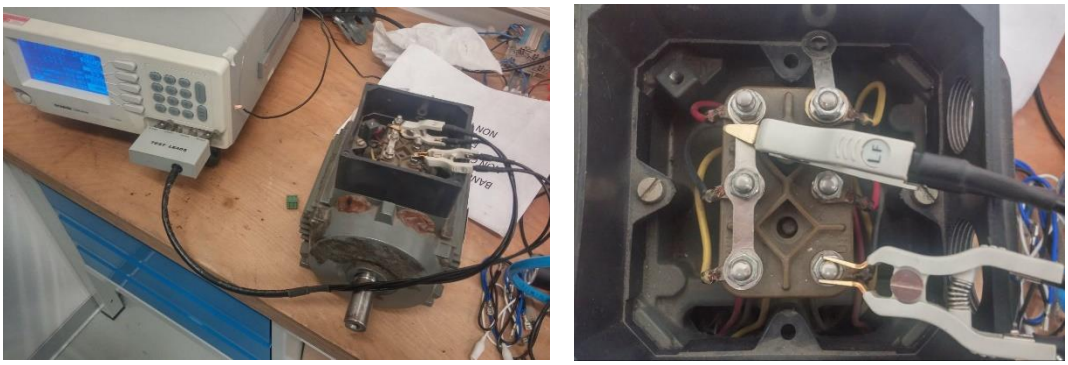

Figure 5- 5: measuring the stator resistance of Induction Motor under tests

5-3-2 Load less Test

This experiment consists of applying a balanced three-phase voltage, at the rated frequency, to the stator terminals, while the rotor is carrying no mechanical load. Currents, voltages, and powers are measured at the motor input. As the slip of the load-less induction motor is very low, the value of the equivalent resistance in the rotor branch of the equivalent circuit is very high. The no-load rotor current is then negligible, and the rotor branch of the equivalent circuit can also be negligible. The approximate equivalent circuit, in the load less test, simplifies as it is shown in Figure 5-6 below:

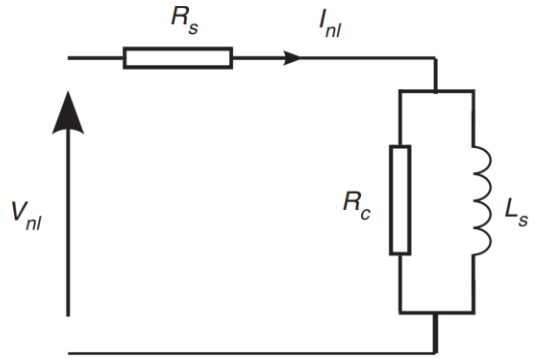

Figure 5- 6: Induction Machine Equivalent Circuit in Load less Test (Giri, 2013)

In the load less test, the losses are caused by the core, the stator copper, and the friction. It turns out that, the load less test input power expresses as follows:

$P_{nl} = P_{cop} + P_{rot} + P_{co}$ (5.23)

Where the stator copper losses are given by:

$P_{cop} = 3\, R_s I_{nl}^2$ (5.24)

The friction losses are given by:

$P_{rot} = F\omega_{m\,nl}^2 + T_d \omega_{m\,nl}$ (5.25)

And the core losses are given by:

$P_{co} = \frac{V_{nl}^2}{R_c}$ (5.26)

Where $V_{nl}, I_{nl}, P_{nl}, Q_{nl}$ and $\omega_{m\,nl}$ denote, respectively, the stator voltage, the stator current, the absorbed active power, the absorbed reactive power, and the rotor speed. The measurements performed, in load less test on the induction motor of Table 5- 1 yields to the following results as we go on by applying a sinusoidal rotating field around the stator by generating appropriate PWM pulses across the switching legs (Giri, 2013).

In this test the switching module MC1H is connected to a power supply with fixed voltage of 106 V. After running the motor in the open loop manner, the voltage and current consumption on the power supply connected to the switching module are visible in Figure 5- 7:

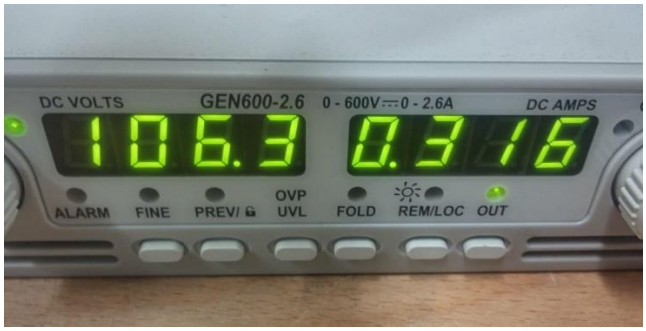

Figure 5- 7: Voltage and Current of IM running in open loop with 50 Hz of rotating field frequency

Now at this time by measuring the phase voltages we have:

$V_{ph} = 45\ V$ (5.27)

So we can now calculate the line to line voltage as below:

$V_{l-l} = V_{ph}\sqrt{3}$ = 77.9 V (5.28)

By monitoring the current on each phase, we see a shape of current as Figure 5- 8 in oscilloscope which is measured using a current probe on one of the phases of the IM under test:

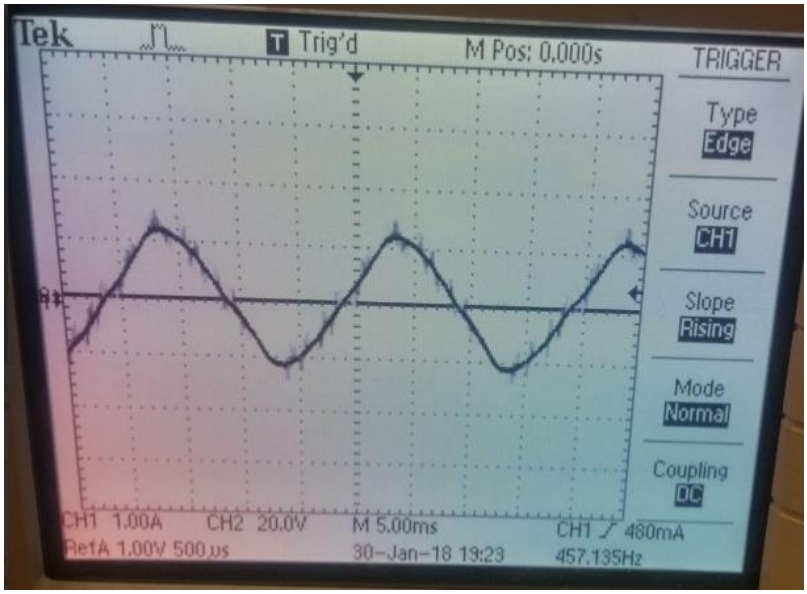

Figure 5- 8: phase current shape in open loop test

As can be seen in Figure 5-8 the peak current value is around 1.25 Amps so we will have the RMS value of the phase current as below:

$$I_{ph-rms} = \frac{I_{ph-peak}}{\sqrt{2}} = 0.88\ A \quad (5.29)$$

The value of (5.29) is also equal to line current in star connection which is known as I_l in the following equations.

Now we go on with calculation of active power P_{ac} as equation (5.30) below:

$$P_{ac} = \sqrt{3} V_l I_l Cos(\emptyset) = 91.42\ \text{W} \quad (5.30)$$

And the apparent power can be calculated using the power supply measurements on Figure 5- 7 as below:

$$P_{app} = V_{dc} I_{dc} = 33.496\ W \quad (5.31)$$

Now from expression of total power in a 3-phase system we can calculate the reactive power Q_{nl} as below in (5.32):

$$Q_{nl} = \sqrt{P_{ac}^2 + P_{app}^2} = 1218.94\ \text{W} \quad (5.32)$$

The stator inductance L_s and the core resistance R_c are given by the following expressions:

$$L_s = \frac{3V_{nl}^2}{\omega_s Q_{nl}} \quad (5.33)$$

$$R_c = \frac{3V_{nl}^2}{P_{nl} - 3R_s I_{nl}^2 - F\omega_{m\,nl}^2 + T_d \omega_{m\,nl}} \quad (5.34)$$

By knowing the value of ω_s=50Hz=314 rad/s we can calculate the value of stator inductance using (5.33) as below:

$L_s = 0.0475$ H (5.34)

5-3-3 Blocked Rotor Test

In this experiment the rotor is blocked, that is, prevented from turning. Then, a balanced three phase voltage is applied to the stator terminals so that the resulting current equals the rated current. Again, all currents, voltages and powers are measured at the motor input. In the blocked rotor operation, the slip *s* is equal to 1 (Giri, 2013). Then, the secondary impedance becomes much smaller, compared to the magnetizing branch, so that the corresponding equivalent circuit boils down to the simpler configuration of Figure 5- 9 as below:

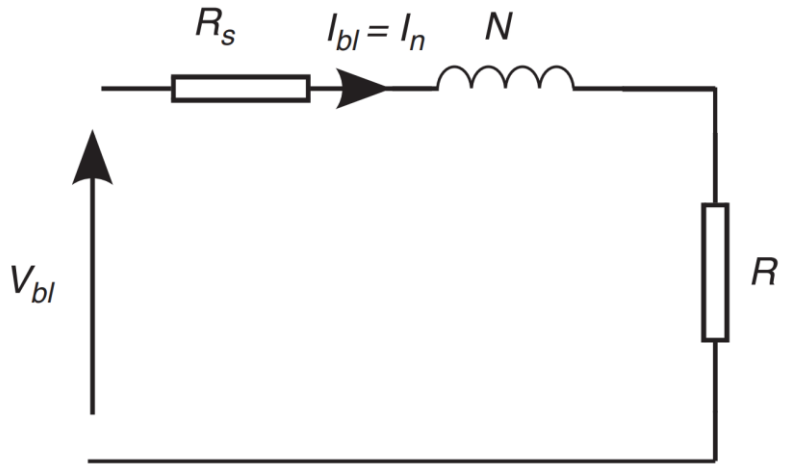

Figure 5- 9: Induction machine equivalent circuit in blocked rotor test (Giri, 2013)

The measurements to be performed in this test are the following:

- The three-phase active power P_{bl}.
- The three-phase reactive power Q_{bl}.
- The line voltage V_{bl}.
- The line current I_{bl}.

R and *N* can be calculated from the equations (5.35) and (5.36) respectively:

$$R = \frac{P_{bl}}{3\, I_{bl}^2} - R_s \quad (5.35)$$

$$N = \frac{\sqrt{(\frac{V_{bl}}{I_{bl}})^2 - (R - R_s)^2}}{\omega_s} \quad (5.36)$$

As $N = \sigma L_r \left(\frac{L_s}{M_{sr}}\right)^2$ and $R = R_r(\frac{L_s}{M_{sr}})^2$,the knowledge of *R* and *N* allows to calculate the rotor time constant and the parameter σ, using the expressions of equations (5.37) and (5.38) below:

$$T_r = \frac{L_s + N}{R} \quad (5.37)$$

$$\sigma = \frac{N}{N + L_s} \quad (5.38)$$

In the real induction machine under test, the rotor was blocked initially, and the controller was generating a sinusoidal PWM waves under switching legs of MC1H to generate a rotating field of 50 Hz.

As can be seen in Figure 5- 10 which is the DC power supply condition at the rotor blocking test.

Figure 5- 10: The Voltage and Current drained from DC power supply during rotor blocking test

As in Figure 5- 10 the voltage which the rotor blocking test was carried on is around 102 Volts and there's an amount of 0.76 Amps of current drained from the power supply.

Figure 5- 11 shows the current shape of one phase of the motor in this condition in oscilloscope which indicates a peak to peak current around 5 Amps which is measured with a Current probe across one of the phases of the IM:

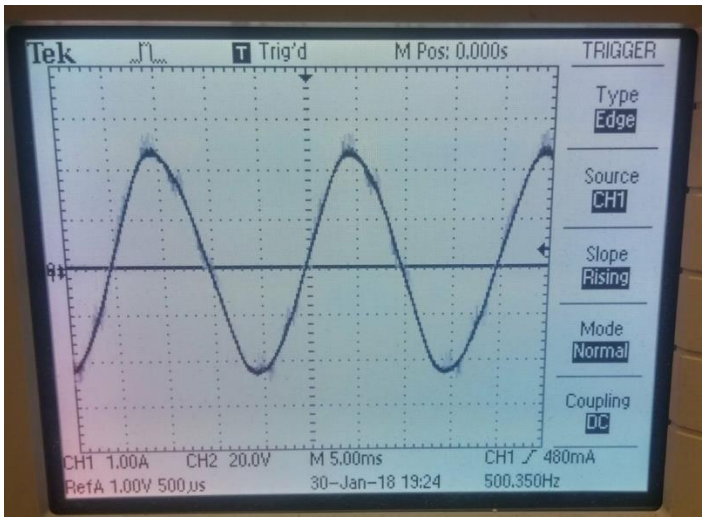

Figure 5- 11: Phase current shape of one of the Induction machine phases in blocked rotor test

In this condition the measured values across the Induction motor are reported as follow:

$V_{ll} = 77.9\ V$ (5.39)

$I_{l-rms} = \frac{2.5}{\sqrt{2}} = 1.76\ A$ (5.40)

$P_{bl} = \sqrt{3} V_{ll} I_{ll} Cos(\emptyset) = 182.85$ W (5.41)

$R = \frac{182.85}{3*1.76^2} - 13 = 6.68\ \Omega$ (5.42)

$N = 0.042$ (5.43)

$\sigma = \frac{0.042}{0.042+0.0475} = 0.497$ (5.44)

$L_r = \sigma \frac{L_s^2}{N} = 0.0266$ H (5.45)

$R_r = R \frac{L_s^2}{L_r^2} = 8.5\ \Omega$ (5.46)

So hereby by taking into account that approximately the magnetizing inductance can be considered equal to rotor inductance $L_r = M$ so we can write the whole identified values for the motor as Table 5- 3 as below:

R_s [Ω]	R_r [Ω]	L_s [H]	L_r [H]	M [H]	J [Kg.m^2]
13	8.5	0.0475	0.0266	0.0266	0.00056

Table 5- 3: Identified Parameters of induction motor under test useful for FOC control

Thanks to the parameter identification done here and using the values on Table 5- 3 now we can control the induction machine in Field Oriented manner and all the crucial parameters are ready.

Chapter 6

6. Controller Design for Induction Machine using FOC

Introduction:

Based on what has been discussed on chapter 3 over the Theory of Field oriented control of induction machines, in this chapter the aim is to discuss about controller design and tunings in order to get a desired motion from the induction motor.

Based on what was talked in chapter 3 we can simplify the whole control structure of an induction motor using FOC with nested loops of current, Flux and speed as below in Figure 6- 1 which is s simplified control diagram of the FOC.

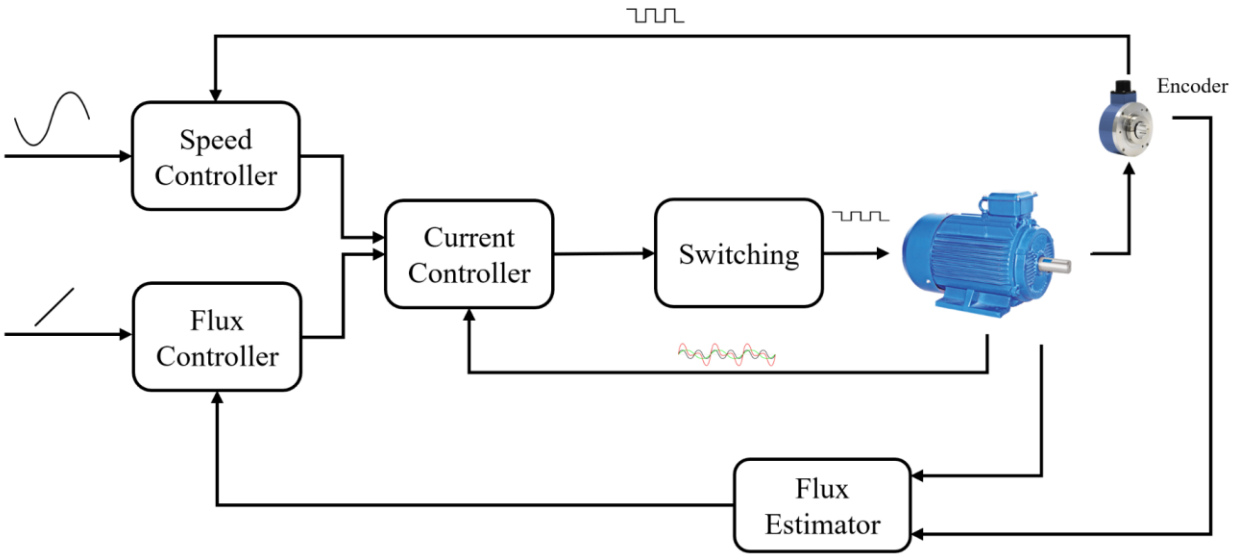

Figure 6- 1: Simplified diagram of FOC method on an induction machine

Based on Figure 6- 1 in order to control an IM there are 2 internal layers of controllers nested inside:

1. **Current controllers:**

   Current controllers are in charge of keeping the track of input current references which are coming from Speed and Flux controllers, in FOC method the Currents read from the phases of the motor are brought into *"d"* and *"q"* axes. The speed controller output will define

the reference for current controller of *"q"* axis and the flux controller output will also define the reference for *"d"* axis current controller.

2. **Speed Controller:**

   This controller is in charge of tracking the speed reference by getting the rotating speed feedback from the shaft of the motor through encoder or other devices and sometimes even observers and estimators.

3. **Flux Controller:**

   This controller keeps the track of rotor flux reference, so using the *"d"* axis it will keep the flux of the motor at desired value.

All of these controllers are acting on different dynamics characteristics of the induction motor, so they need to be tuned and taken care accordingly.

In this chapter we are going to discuss the methods and techniques which was used for achieving this goal starting from Current controllers and following up with speed and rotor flux controller at the end.

6-1 Current Controllers definitions

In current controller design we are intending to control the current dynamics over *"d"* and *"q"* axis, to this end we will firstly start with equations introduced in chapter 3 for vector control of induction machine (Dezza, 2017).

As mentioned in chapter 3 in equations (3.16) and (3.17) the real voltages able to increase or decrease the currents i_{sd} and i_{sq} are:

$$u_{sd} = R_{ks} i_{sd} + L_{ks} p i_{sd} \quad (6.1)$$

$$u_{sq} = R_{ks} i_{sq} + L_{ks} p i_{sq} \quad (6.2)$$

The term "$-\frac{R_r}{M}\Psi_r$" seems to be a kind of disturbance while "$+\dot{\theta}_m \Psi_r$" is very similar to back emf in DC machines. The term "$-\dot{\theta}_s L_{ks} i_{sq}$" and "$+\dot{\theta}_s L_{ks} i_{sd}$" are coupling terms between the i_{sd} and i_{sq} control loops.

The decoupled current controller, with a compensation of the disturbances and the ability to perform a "start on fly", has the structure of Figure 6- 2 as below:

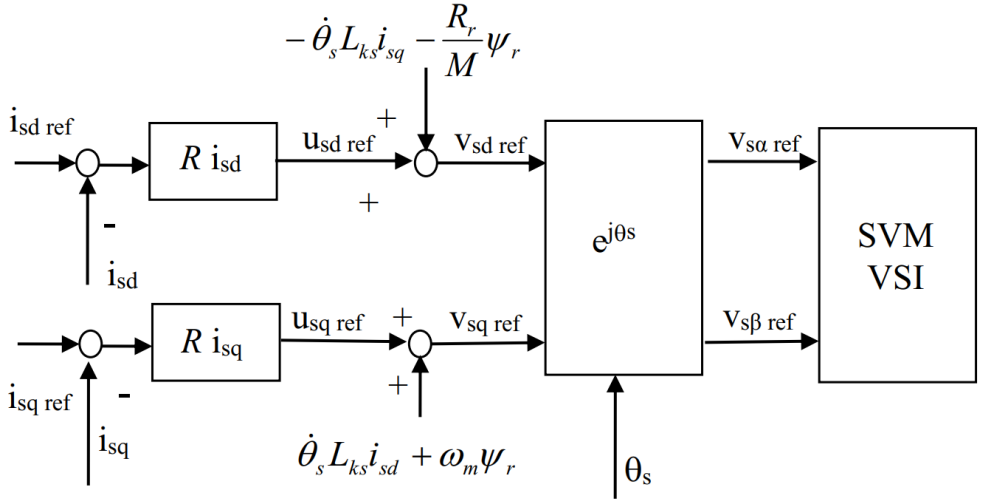

Figure 6- 2: Decoupling and disturbance compensation on current controllers in FOC of IM (Dezza, 2017)

6-1-1: Current Control Loops

If we suppose to have an estimator free from errors, so as to be able to calculate the decoupling terms and feed-forward (disturbances compensation) accurately, the current regulator scheme becomes as the one in Figure 6- 3:

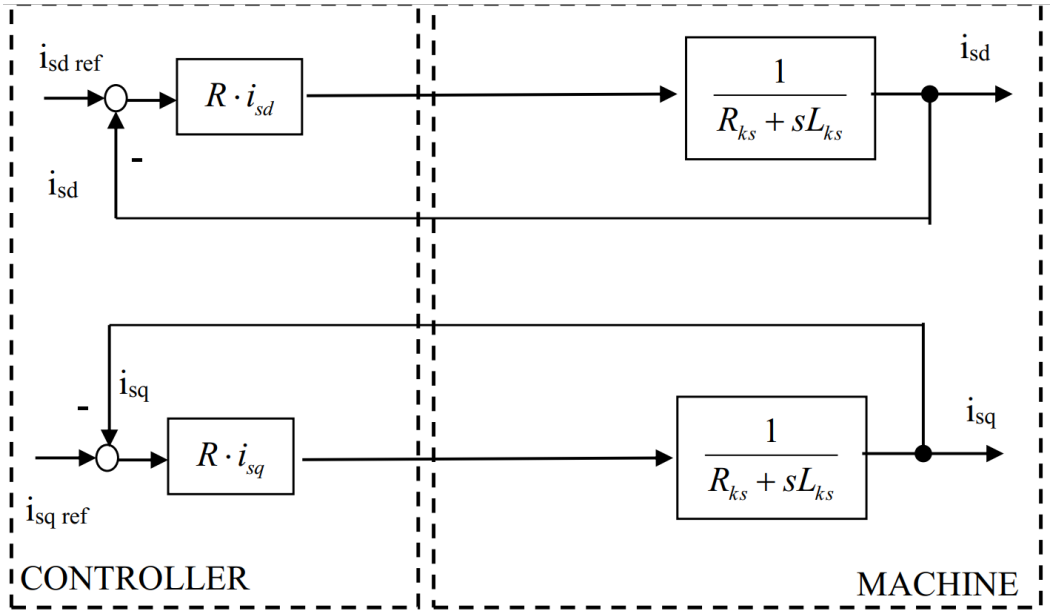

Figure 6- 3: Equivalent block diagram of current control with an ideal decoupling (Dezza, 2017)

The simplification is possible because the coupling terms and the disturbances in the machine are compensated by terms calculated by the controller itself mainly due to the presence of integrator in PI controller.

In these conditions there are no difference between the *"d "* and "*q*" current components regulators. This suggests that the controller synthesis is carried out by calculating the gains for only one loop.

The system transfer function under control is as equation (6.3) below:

$$BI(s) = \frac{1}{R_{ks} + s.L_{ks}} \quad (6.3)$$

The closed loop transfer function, useful for synthesis of speed and flux control loop is as equation (6.4) as following where K_pI and K_iI are the gains of the PI controller:

$$LI(s) = \frac{\left(K_pI + \frac{K_iI}{s}\right).BI(s)}{1 + \left(K_pI + \frac{K_iI}{s}\right).BI(s)} \quad (6.4)$$

6-1-2 Current Controllers Tuning Theories

At this stage while now we have proper modeling of the system under control which will be controlled through a cascade topology which the current controllers are the first internal loops. In the following the intention will be focusing on tuning these controllers for having a robust and reliable control (Instruments T. , InstaSPIN-FOC™ and InstaSPIN-MOTION™, 2017).

The transfer function under control in current loop is a first order transfer function as in equation (6.3). The PI controller firstly designed will be based on the series topology of PI controllers as in Figure 6- 4 below:

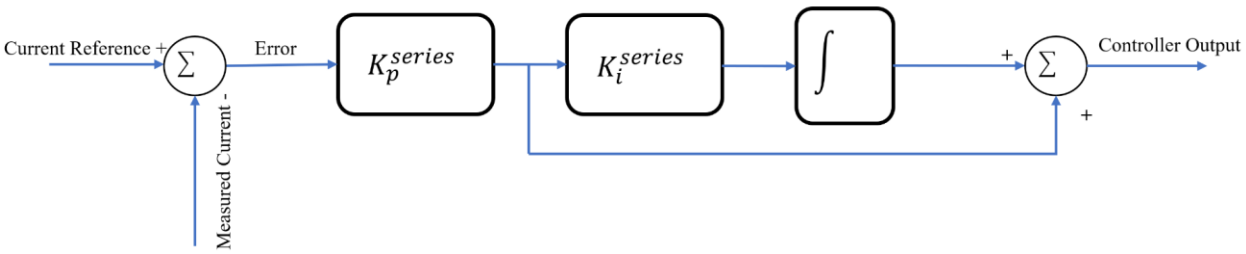

Figure 6- 4: PI controller with series topology

Using the series form of the PI controller, we can define its "s-domain" transfer function from the error signal to the controller output as:

$$PI(s) = \frac{K_P^{Series} * K_i^{Series}}{s} + K_p^{Series} = \frac{K_P^{Series} * K_i^{Series}(1+\frac{s}{K_i^{Series}})}{s} \quad (6.5)$$

We will use a first-order approximation of the motor winding to be a simple series circuit containing a resistor, an inductor, and a back-EMF voltage source. Assuming that the back-EMF voltage is a constant for now (since it usually changes slowly with respect to the current), we can define the small-signal transfer function from motor voltage to motor current as equation (6.6):

$$\frac{I_{(s)}}{V_{(s)}} = \frac{\frac{1}{R}}{(1+\frac{L}{R}s)} \quad (6.6)$$

We can now define the "loop gain" as the product of the PI controller transfer function and the V-to-I transfer function of the RL circuit as equation (6.7):

$$G_{loop} = PI(s) * \frac{I(s)}{V(s)} = \left(\frac{K_p^{Series} * K_i^{Series} * \left(1+\frac{s}{K_i^{Series}}\right)}{s}\right)\left(\frac{\frac{1}{R}}{(1+\frac{L}{R}s)}\right) \quad (6.7)$$

So based on this formula we will be able to calculate the closed loop transfer function considering the feedback gain as unity, as equation (6.8):

$$G(s) = \frac{G_{loop}}{1+G_{loop}} \quad (6.8)$$

After substitution of G_{loop} inside G(s) using equations (6.7) and (6.8) we will obtain equation (6.9):

$$\mathrm{G(s)} = \frac{\left(\frac{\mathrm{K_p^{series}K_i^{series}}\left(1+\frac{s}{\mathrm{K_i^{series}}}\right)}{s}\right)\times\left(\frac{\frac{1}{R}}{1+\frac{L}{R}s}\right)}{1+\left(\frac{\mathrm{K_p^{series}K_i^{series}}\left(1+\frac{s}{\mathrm{K_i^{series}}}\right)}{s}\right)\times\left(\frac{\frac{1}{R}}{1+\frac{L}{R}s}\right)} \quad (6.9)$$

With some simplification on equation (6.9) we will have equation (6.10) as below:

$$\mathrm{G(s)} = \frac{\left(1+\frac{s}{\mathrm{K_i^{series}}}\right)}{\left(\frac{L}{\mathrm{K_p^{series}}\times\mathrm{K_i^{series}}}\right)s^2+\left(\frac{R}{\mathrm{K_p^{series}}\times\mathrm{K_i^{series}}}+\frac{1}{\mathrm{K_i^{series}}}\right)s+1} \quad (6.10)$$

For avoiding complex poles and zeros we have assumed right away that we want to select and in such a way as to avoid complex poles. The reason is according to equation (6.10) we can have two complex poles which in case of any error in calculations we may have poles very close to jω axis which are cause of oscillation and even instability. In other words, we can factor the denominator into an expression as follows in equation (6.11), where C and D are real numbers:

$$\left(\frac{L}{K_p^{Series}*K_i^{Series}}\right)s^2+\left(\frac{R}{K_p^{Series}*K_i^{Series}}+\frac{1}{K_i^{Series}}\right)s+1=(1+Cs)(1+Ds) \quad (6.11)$$

If now we conduct the multiplication of the equation (6.11) on the right side and we compare it to the left side we can conclude some results as equations (6.12) and (6.13) :

$$\frac{L}{K_p^{Series}*K_i^{Series}} = C*D \quad (6.12)$$

$$\frac{R}{K_p^{Series}*K_i^{Series}}+\frac{1}{K_i^{Series}} = C+D \quad (6.13)$$

Considering equation (6.13) a very simple way of selecting values for C and D can be as following equations of (6.14) and (6.15):

$$\frac{R}{\mathrm{K_p^{series}}\times\mathrm{K_i^{series}}} = \mathrm{C} \quad (6.14)$$

$$\frac{1}{\mathrm{K_i^{series}}} = \mathrm{D} \quad (6.15)$$

To clarify the reason of this choice if we substitute the denominator of equation (6.10) with its respective equivalent in equation (6.11) and we use the equations of (6.14) and (6.15), the closed loop transfer function of the current controller becomes as equation (6.16) below:

$$G(s) = \frac{(1+\frac{s}{K_i^{series}})}{(1+\frac{R}{K_p^{series} \times K_i^{series}}s)(1+\frac{1}{K_i^{series}}s)} \quad (6.16)$$

Considering equation (6.16) a proper selection of C and D not only can lead to real poles but also will result that the closed loop current controller transfer function to have only one real pole (elimination of poles and zeros) which results in an non-oscillatory output.

Now by considering C and D as equations (6.14) and (6.15) for upholding equation (6.12) K_i^{series} should be as equation (6.17) below:

$$K_i^{series} = \frac{R}{L} \quad (6.17)$$

Considering $K_i^{Series} = \frac{R}{L}$ and noting that K_i^{Series} is the frequency at which the controller zero occurs, so we have to consider K_i^{Series} to be equal to the pole of the plant, substituting K_i^{Series} inside equation (6.16) we will get equation (6.18) as below:

$$G(s) = \frac{1}{1+\frac{L}{K_p^{series}}s} \quad => \quad K_p^{series} = L \quad \times \text{Bandwidth} \quad (6.18)$$

Since our model is based on parallel model after deriving out the series configuration of gains we need to simply them like equations (6.19) and (6.20) below into parallel PI controller Gains:

$$K_p^{Series} = K_p^{parallel} \quad (6.19)$$

$$K_i^{Parallel} = K_p^{Series} * K_i^{Series} \quad (6.20)$$

6-1-3 Current Controller Simulations and Implementations

In order to control the Current in real environment using the FPGA-based platform we firstly developed a simulation model for each and every component inside the design, then by checking the feasibility of algorithms we moved toward real-world implementation. In this section we will come up with simulation and real models which are used in FPGA for doing the FOC.

In real-world Implementation the very first step will be a proper sampling from the current feedbacks from the motor phases. As it was elaborated in Chapter 4 of this thesis, there are two ways for sampling from the current using MC1H inverter board:

1. Using the Hall effect current sensors which they provide an insulated voltage signal proportional to the current passing through the motor phases
2. Using Shunt resistor-based current measurement section which again provides us a voltage proportional to the current passing through the motor phases, but it's not isolated from the switching side.

In our case we preferred to use the first option which is the Hall current sensors.

The current sensing architecture can be summarized like in Figure 6- 5 as below:

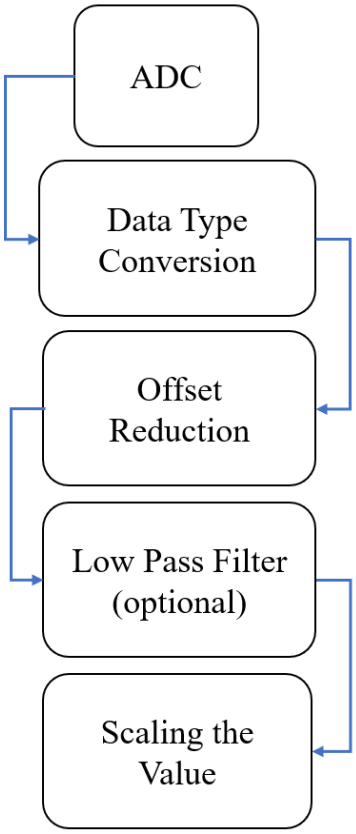

Figure 6- 5: The process of reading the Current from the motor Phases in NI FPGA based environment

As in Figure 6- 5 to read a proper value for the current in each phases of the motor firstly we have to connect the Hall-effect current sensors output to one of the simultaneous analog inputs of the FPGA based platform. Then in the software we proceed as in Figure 6- 5 with the following procedure:

1. Using the Simultaneous Analog Input section of the NI board, with 12 bits of resolution and maximum 100kS/s rate with maximum inputs range of $\pm 9V$, we read the corresponding voltage output from the Hall-effect current sensors for two phases of the IM under test

2. The ADC unit output in LABVIEW environment returns the value of the voltage read from current sensors, in order to be able to process this data inside LABVIEW and FPGA we have to convert this data to a specific type of data known as Fixed-Point data type. Since the timing is very critical in FPGA based programming, there must be a very deep consideration about the minimum and best possible choice of variable sizes without losing their accuracy, overflow and underflows of variables.

3. A fix offset of voltage calculated in the beginning of the process will be deducted from the current sensors output just to make sure that the voltages are not biased

4. In case of need of filtering data for reducing the noise level of the current signals read from ADC unit we can use a simple first order Low-Pass filter

5. Now depending on the type of the current sensors used we have to convert the voltage levels read from the sensors through ADC unit to a real Current value. Since we've used the Hall-effect Current sensors, as it's mentioned in the MC1H data sheet these sensors will show 1 volt per 2.4 Amps in a bidirectional manner, so here we simply need to multiply the voltage read from the sensors to 2.4 to get the real value of the Current passing through each phase in ampere.

In Figure 6- 6 the LABVIEW code to read the Currents from the Hall-effect sensors is depicted, it worth mentioning that this code is running inside a loop with 40 micro-seconds of loop rate, meaning that the whole process of reading Currents and executing a Current controller output to PWM unit will take 40 micro-seconds each time, subsequently the sampling rate of the Current in this thesis is considered as 25kHz.

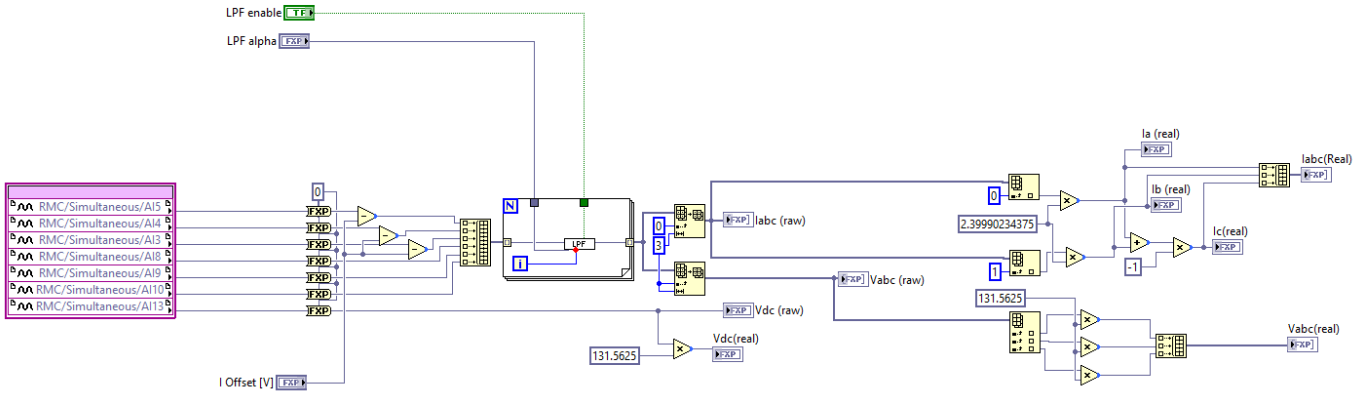

Figure 6- 6: Current reading code inside the LABVIEW software using NI FPGA-based platform

After finding out the Current values of motor phases, we can proceed with current controller design and considerations.

In order to control the Current in FOC and also in FPGA-based platform we have to proceed as in Figure 6- 7 below which is the process of controlling the Current from measures till the switching signals to the motor driver:

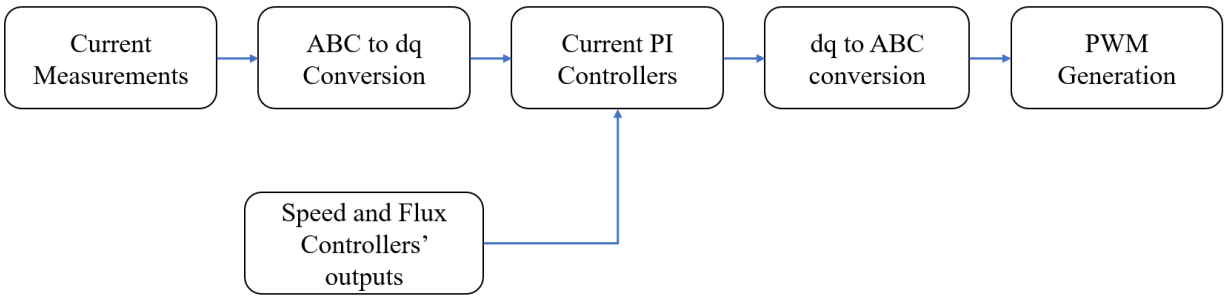

Figure 6- 7: Current Controlling procedure in FOC in FPGA-based NI platform

As can be seen in Figure 6- 7 after reading the 3 phase Currents of the IM we can proceed to control the Currents on "dq" axes separately using the scheme of the Figure 6- 7 as following:

1. First, we measure the Current floating in each phase
2. Secondly, we will operate "ABC" to "dq" conversion which can be done as Figure 6- 8 below using a Clark and Park Transformations using the NI FPGA libraries available in LABVIEW

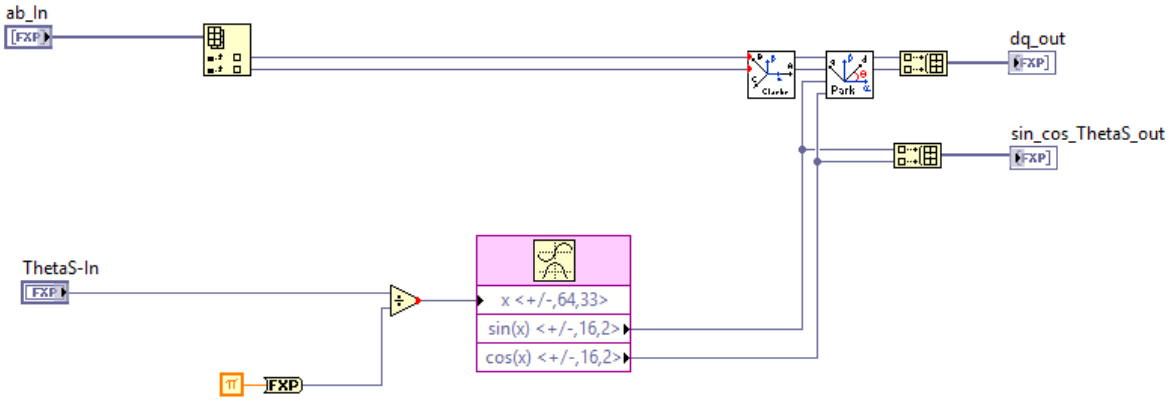

Figure 6- 8: ABC to dq conversion code scheme in LABVIEW

Also, Inside MATLAB Simulink for completely simulating the same behavior we used a similar Park and Clarke transformation as it's implanted inside NI FPGA as can be seen in Figure 6- 9, it worth mentioning that this type of implementation of "ABC" to "dq" is more hardware oriented because it's based on simple arithmetic calculations. Inside Figure 6- 9 the Matrix gain is:

sqrt(2/3)*[1 -1/2 -1/2;0 sqrt(3)/2 -sqrt(3)/2;-1/2 -sqrt(3)/2 1/sqrt(2)]

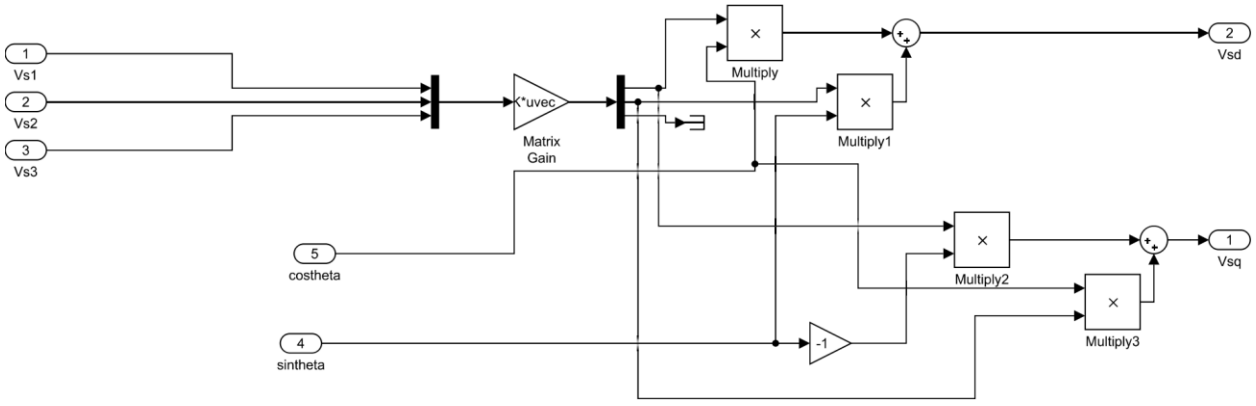

Figure 6- 9: ABC to dq Conversion done in MATLAB Simulink

3. The next step is providing proper feedbacks and references to the PI controllers of Current controllers on "d" and "q" axes separately. As it's mentioned in Chapter 3 the PI current controller acting on *"d"* axis will be fed by the output of the Flux controller and the Current controller acting on *"q"* axis will be fed by the output of the Speed controller. The PI controllers of Current controllers are designed as in Figure 6- 10 below:

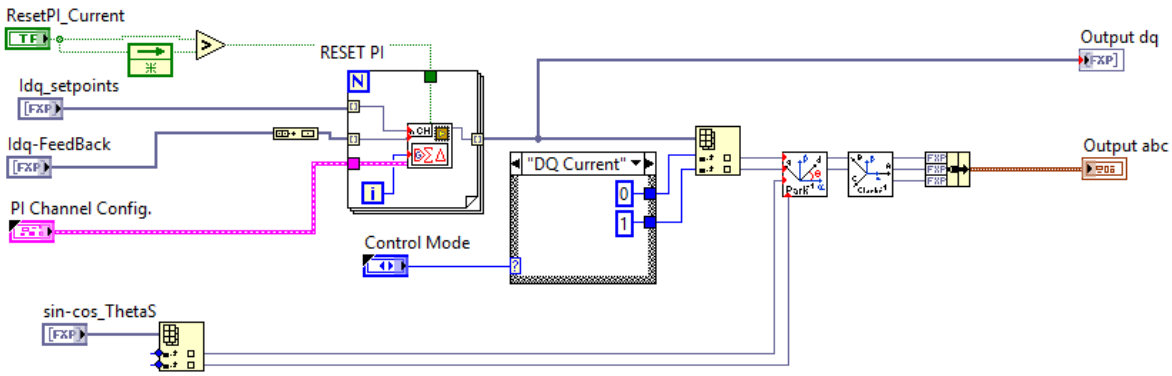

Figure 6- 10: PI Controllers acting on "d" and "q" axis of the Current in FOC implemented on NI FPGA-based Platform

For implementation of PI of current controller inside MATLAB Simulink we used the series model of PI as it was shown in section 6-1-2 as can be seen in Figure 6- 11 below:

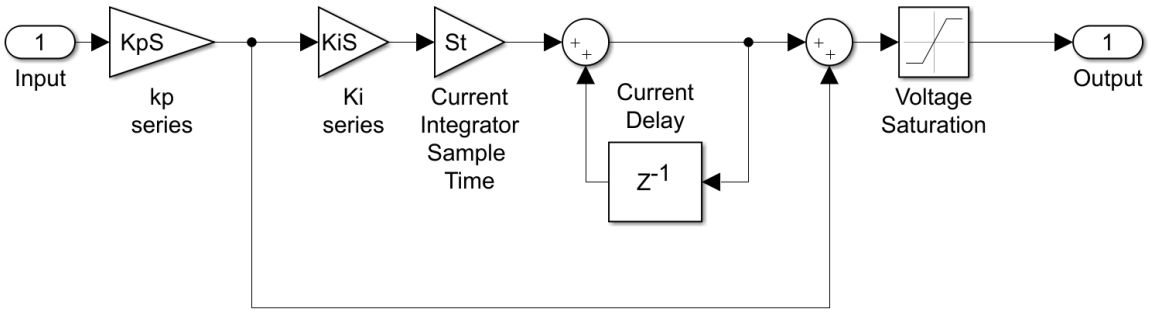

Figure 6- 11: Series PI current Controller done in MATLAB Simulink

4. The forth step is dedicated to converting back from "dq" to "ABC" reference frame again using inverse Clark and Park transformation which are available as libraries for FPGA-based platforms in LABVIEW.
   Again, for this part we made a simulation model as in Figure 6- 12 , which also is adapted to hardware implementation and known as "dq" to "ABC" transformation with implementation of inverse Park and Clarke transformation. The Matrix Gain in Figure 6-12 is:

   sqrt(2/3)*[1 0 0;-1/2 sqrt(3)/2 0;-1/2 -sqrt(3)/2 0]

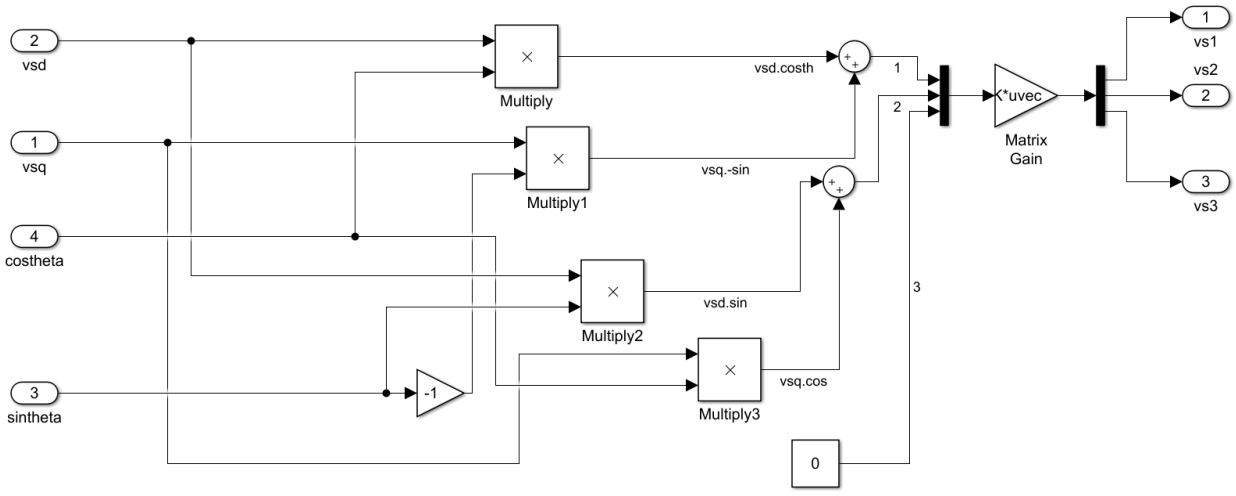

Figure 6- 12: MATLAB Simulink model for Inverse Park and Clarke transformation known as "dq" to "ABC"

5. The last step is sending the calculated ABC values of voltages to the PWM generator section which generates a sinusoidal 3-phase PWM based on the 3 inputs of A,B and C which we feed to it. The PWM generator works based on triangular comparison of the voltage levels of the A,B and C in order to generate proper sinusoidal PWM pulses on the

output of the FPGA-based platform. The PWM frequency used in this thesis is 12kHz and the code which generates our desire sinusoidal PWM pulses for each phase is shown in Figure 6- 13 below:

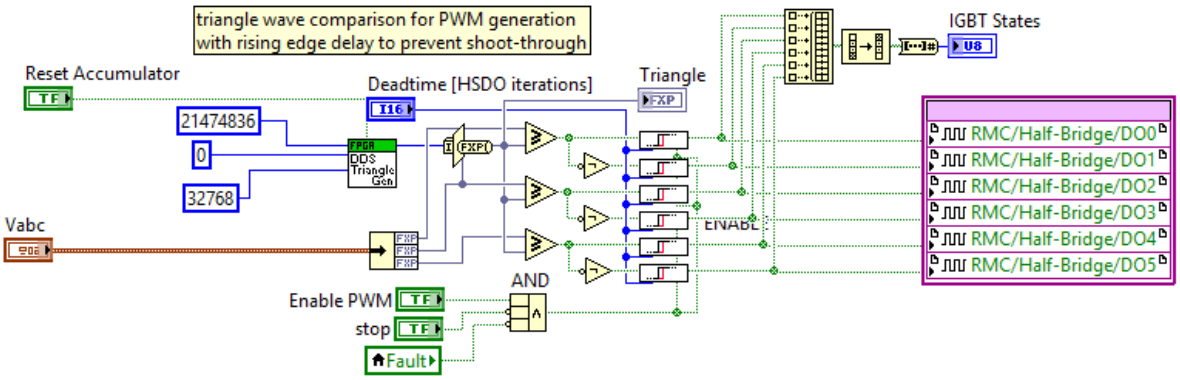

Figure 6- 13: Sinusoidal PWM generation with three phase input voltages

As it's shown in Figure 6- 13 there is a consideration for dead-time inside the PWM signals which is vital specially when one uses MC1H to avoid shoot-through currents, since MC1H doesn't have an internal dead-time generator.

The PWM pulses are sent to the Half-bridge outputs of the GPIC board to be sent to MC1H to control the state of the IGBTs or Mosfet inside the inverter.

After all the final structure of the current controlling scheme inside the LABVIEW software can be shown as in Figure 6- 14 below:

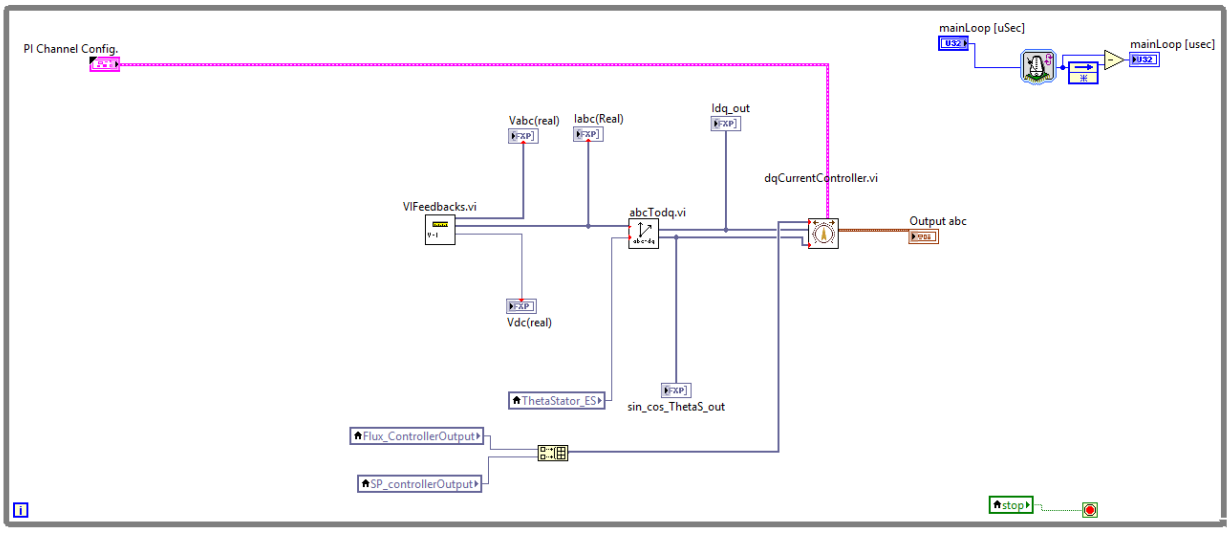

Figure 6- 14: Current Controller loop for Controlling d and q Currents inside the FPGA-based NI platform

As it was mentioned this loop works with 40 micro-seconds of loop rate meaning that the current controller works with 25kHz of sampling and output generation rate.

As can be seen in Figure 6- 14 the speed and flux controller's outputs are fed into the current controller setpoint input to be considered as the desired reference to be followed at each time for the current controllers. In Next chapters we will talk more about the Speed and Flux controllers design and considerations.

6-2 Speed Controller Definitions and Loop

In order to start the tuning of the speed controller in nested loop of the FOC firstly we address the synthesis of the speed controller loop as in Figure 6- 15 below:

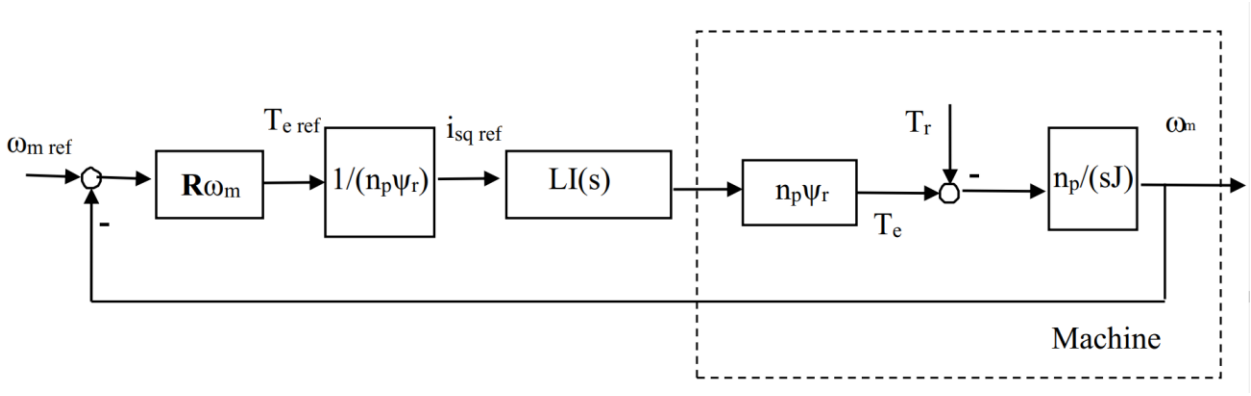

Figure 6- 15: Speed Control Loop block diagram in FOC of induction machine (Dezza, 2017)

The transfer function of the controlled process depends on the mechanical load and on the "q" axis current, which has the same transfer function of the *"d"* axis, the transfer function under control of the speed controller can be written as equation (6.21) below:

$$B_{\omega m}(s) = \frac{LI(s).n_p\Psi_r}{n_p\Psi_r} . \frac{n_p}{J.s} \quad (6.21)$$

The torque T_r is neglected in the expression of the mechanical load because it is considered as an external disturbance.

Now for controlling the speed we will use a PI controller of first order and in the next section we will discuss how to tune this regulator based on the already tuned current controllers internally.

6-2-1 Speed controller tuning

As it was mentioned in current controller design section of this thesis, the closed loop transfer function of the current controller will be as equation (6.22):

$$G(s)_{Current} = \frac{1}{1+\frac{L}{K_p^{series}}s} \quad (6.22)$$

Now again we will consider firstly a series PI controller like current controller for speed controller, so the PI controller transfer function for speed controller can be written as equation (6.23) below (Instruments T. , InstaSPIN-FOC™ and InstaSPIN-MOTION™, 2017):

$$PI_{speed}(s) = \frac{spdK_p^{series} \times spdK_i^{series}}{s} + spdK_p^{series} = \frac{spdK_p^{series} \times spdK_i^{series}(1+\frac{s}{spdK_i^{series}})}{s} \quad (6.23)$$

Now we have to calculate the open-loop transfer function of the whole cascaded controllers, by looking at Figure 6- 15 we can write the system open loop transfer function as equation (6.24) below:

$$GH(s) = \left(\frac{spdK_p^{series} \times spdK_i^{series}\left(1+\frac{s}{spdK_i^{series}}\right)}{s}\right)\left(\frac{1}{1+\frac{L}{K_i^{series}}s}\right)\left(\frac{n_p}{J_m.s}\right) \quad (6.24)$$

With some simplification on equation (6.24) the open loop transfer function of the speed controller will become as equation (6.25) below:

$$GH(s) = \frac{K \times spdK_p^{series} \times spdK_i^{series}(1+\frac{s}{spdK_i^{series}})}{s^2(1+\frac{L}{K_p^{series}}s)} \quad (6.25)$$

Where K in equation (6.25) is :

$$K = \frac{n_p}{J_m} \quad (6.26)$$

Considering the equation (6.24) we can point out to the following specifications about the open loop transfer function:

- Two poles in origin which results a $-40\frac{dB}{Decade}$ slope in low frequencies
- One zero at $s = spdK_i^{series}$ related to speed controller
- One pole at $s = \frac{K_p^{series}}{L}$ related to Current controller

Considering the above mentioned points, we can understand that, existence of two poles in origin results that the phase of the open loop transfer function starts from -180 degrees in low frequencies, in order to have proper stability the pole of the closed loop current regulator $(\frac{K_p^{series}}{L})$ should be placed in higher frequencies than the zero of the speed controller $(spdK_i^{series})$, so if we suppose

that the unity gain frequency is somewhere between these two frequencies, the bode diagram of the open loop transfer function will be similar to Figure 6- 16:

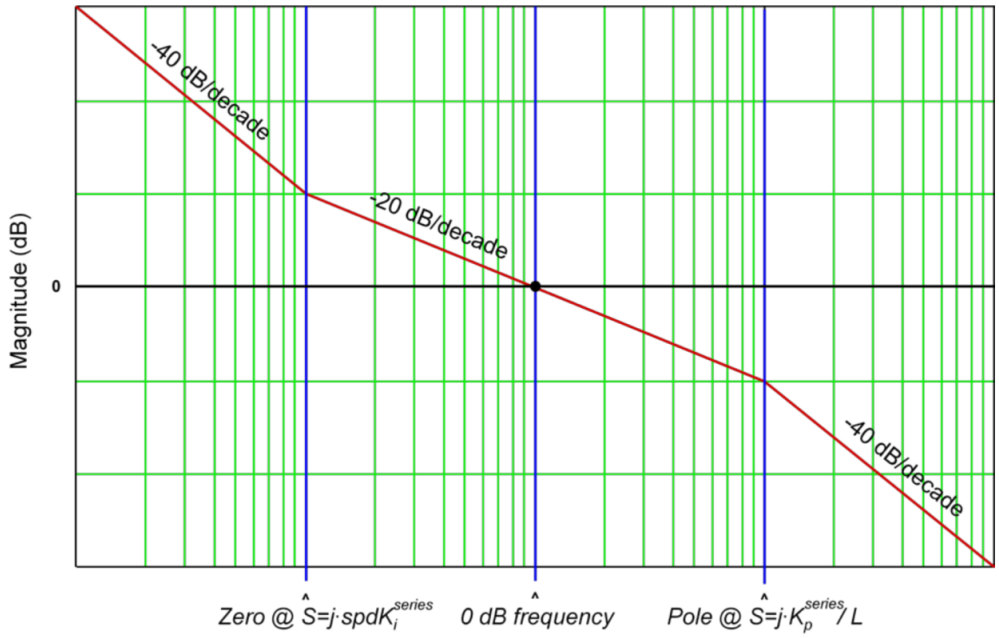

Figure 6- 16: Bode diagram of the open loop system (Instruments T. , InstaSPIN-FOC™ and InstaSPIN-MOTION™, 2017)

The reason the shape of this curve is so important is because the phase shift at the 0-dB frequency determines the stability of the system. In order to achieve maximum phase margin (phase shift: 180°) for a given separation of the pole and zero frequencies, the 0-dB frequency should occur exactly half way in between these frequencies on a logarithmic scale. In other words, we should have equation (6.27) and (6.28) as below:

$\omega_{0dB} = \delta \times \omega_{zero}$ (6.27)

$\omega_{pole} = \delta \times \omega_{0dB}$ (6.28)

Using equations (6.27) and (6.28) we can say that:

$\omega_{pole} = \delta^2 \times \omega_{zero}$ (6.29)

So using equation (6.29) and open-loop transfer function the equation (6.30) can be drive to define the coefficient $spdK_i^{series}$ as below:

$spdK_i^{series} = \frac{K_p^{series}}{\delta^2 \times L}$ (6.30)

Parameter "δ" is called as damping factor, the larger δ is, the further apart the zero corner frequency and the Current loop pole will be. And the further apart they are, the phase margin is allowed to peak to a higher value in-between these frequencies. This improves stability at the expense of speed loop bandwidth.

If δ = 1, then the zero-corner frequency and the current loop pole are equal, which results in pole/zero cancellation and the system will be unstable. Theoretically, any value of δ > 1 is stable since phase margin > 0. However, values of δ close to 1 result in severely underdamped performance.

Now is the time to calculate the final coefficient of cascade controller of speed loop $spdK_p^{series}$. As we see that the open-loop transfer function of the speed loop from equation (6.25) will be unity gain (0 dB) at a frequency equal to the zero-inflection point frequency multiplied by δ. In other words, we will have equation (6.31) as below:

$$\left| \frac{K \times spdK_p^{series} \times spdK_i^{series} \left(1 + \frac{s}{spdK_i^{series}}\right)}{s^2 \left(1 + \frac{s}{\delta^2 spdK_i^{series}}\right)} \right|_{s = j \times \delta \times spdK_i^{series}} = 1 \quad (6.31)$$

By solving equation (6.31) for calculating the coefficient $spdK_p^{series}$ we can have equation (6.32) as below:

$$spdK_p^{series} = \frac{K_p^{series}}{L \times \delta \times K} = \frac{\delta \times spdK_i^{series}}{K} \quad (6.32)$$

The advantage of such a design for speed and current controller is their ease of use for applications benefiting from auto-tuning and specially applications which the controllers are tuned inside hardware. Actually, by using this method all the tunings will follow a logical path which leads to feasible tunings for the hardware in terms of resources needed and loads of calculations.

6-3 Speed and Current Controllers Gains using the Damping factor

In previous sections we have discussed how to distill the design of a cascaded speed controller from four PI coefficients down to two "system" parameters. One of those parameters is simply the bandwidth of the current controller. The other is the damping factor (δ). The damping factor represents the tradeoff between system stability and system bandwidth in a single number. Now we are going to have a look at damping factor in time and frequency domain (Instruments T. , InstaSPIN-FOC™ and InstaSPIN-MOTION™, 2017).

Figure 6- 17 below, illustrates the open-loop magnitude and phase response for a system where the current controller bandwidth is arbitrarily set to 100 Hz. For our purposes, it really doesn't matter what the current bandwidth is, as it only serves to provide a reference point on the frequency axis. However, the shape of the curves won't change, regardless of what the current bandwidth is. The damping factor is swept from 1.5 to 70 in 8 discrete steps to show how it affects system response.

A value of 1.0 corresponds to the condition where the open-loop gain intercepts 0 dB right at the frequency of the current controller bandwidth. This results in pole/zero cancellation at this frequency with a phase margin of zero, which results in instability of the system.

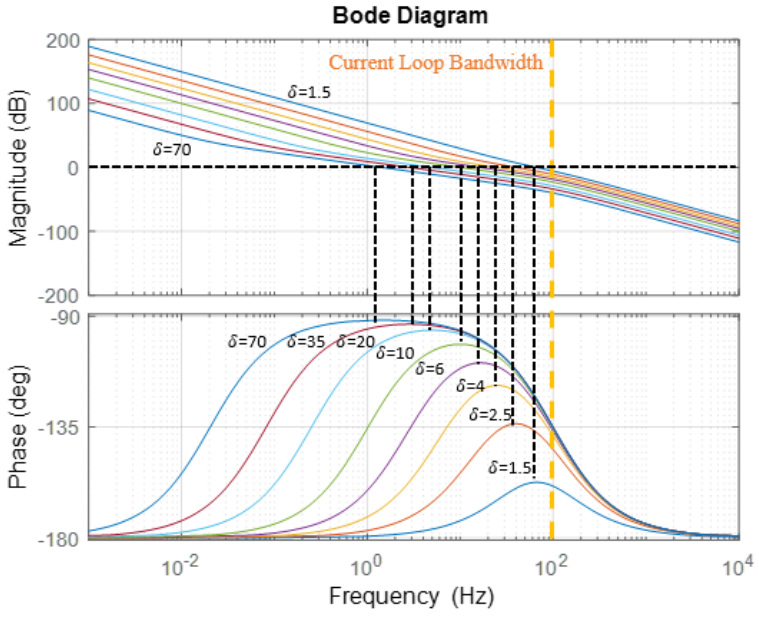

Figure 6- 17: Speed controller Open loop Bode diagram as a function of δ

One of the goals with the damping factor equations is to achieve the maximum stability possible for a given bandwidth. This is seen on the open-loop phase plots which indicate the phase margin peaks to its maximum value right at the frequency where the open-loop gain plots cross 0 db. As the stability factor is increased, you eventually reach a point of diminishing returns as the signal phase shift approaches -90 degrees. However, the gain margin continues to improve at the expense

of a much slower system response. So, in nutshell it worth mentioning that by increasing δ as the stability of the system enhances, the bandwidth decreases.

Figure 6- 18 illustrates the closed loop magnitude response of the speed loop, again assuming a Current controller bandwidth of 100 Hz. Just like the open-loop response, the actual current controller bandwidth is irrelevant in determining the shape of the curves and only serves to associate the curves with a specific frequency reference point.

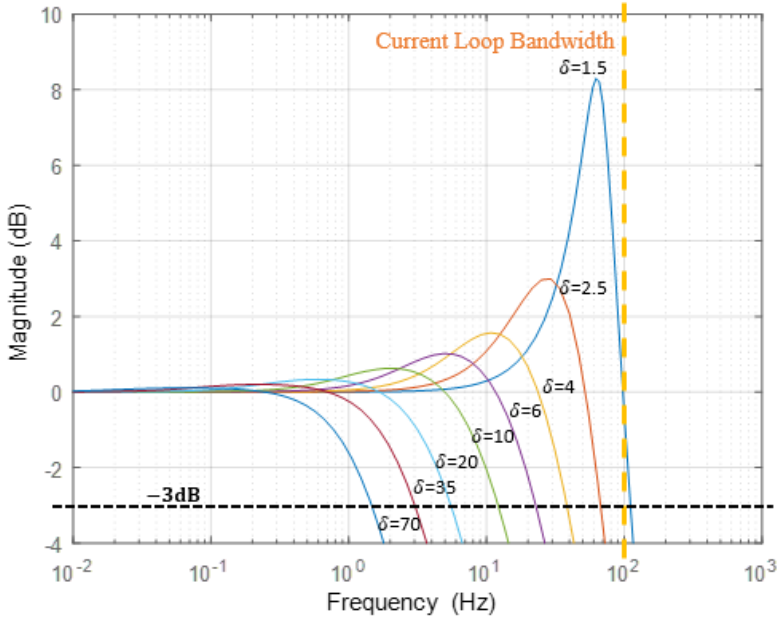

Figure 6- 18: Speed Controller closed loop bandwidth as a function of δ

As can be seen in Figure 6- 18 by increasing the δ, the bandwidth of the closed loop system decreases, and also as δ gets closer to 1, the closed loop response exhibits higher peaks which is the result of pushing the complex poles of the closed loop closer to $j\omega$ axis.

In Figure 6- 19 which shows the normalized step response of the system for various values of the damping factor this phenomenon is more evident. As can be seen, values below 2 are usually unacceptable due to the large amount of overshoot. At the other end of the scale, values much above 30 usually unacceptable due to the large amount of settling time. So, the values in between will be usually a good choice for design values.

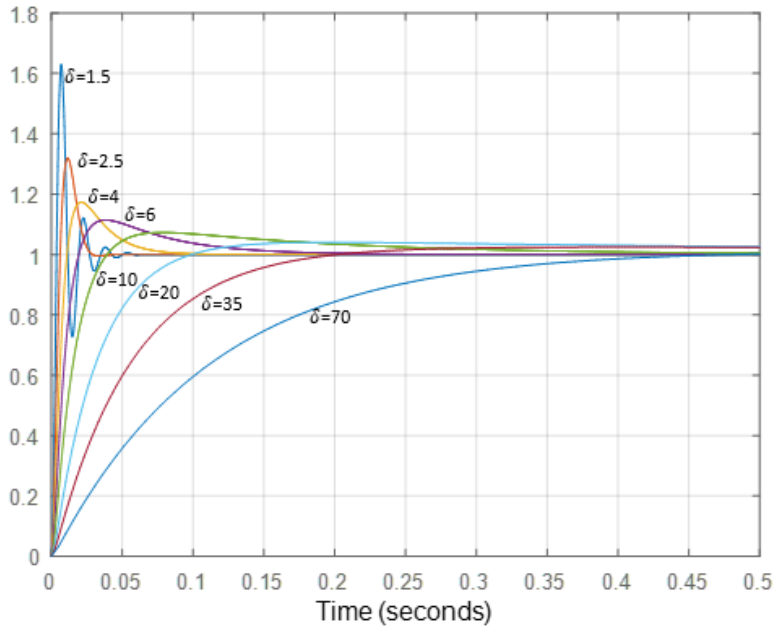

Figure 6- 19: Normalized Step response of speed controller based on different values of δ

The final goal of this chapter was finding an appropriate bandwidth which we are able to continue our design and controller gain's identification with. So up to now the steps we have to pass in order to define our gains for speed controller can be summarized as below:

1. Selecting the desired bandwidth of our speed controller (BW_s).
2. Using the Figure 6- 19 we will choose the minim δ which causes the most satisfactory result for us.
3. The Current controller bandwidth (BW_c) will be derived out using equation (6.33) below which is obtained by curve fitting techniques:

$$BW_c = \frac{K_p^{series}}{L} = BW_s \times \left(\delta + 2.16 \times e^{-\frac{\delta}{2.8}} - 1.86\right) \ (rad/sec) \quad (6.33)$$

6-3-1- Calculations of Speed and Current Controllers' PI gains

According to what has been said during the previous sections, in this part we are going to calculate the numerical values of the current and speed controllers' gains.

At the very first step for designing the PI controllers' and calculation of their gains in cascaded loops of current and speed we need to focus on the bandwidth considerations. Since the current controller is dealing with electrical dynamics of the Induction machine which is much faster than the dynamics of the mechanical part in our case, so the current controller should be designed in a way that it's able to handle and track higher frequency references to reject properly the torque disturbances and better and smoother speed and flux control.

As can be seen in equation (6.33) above, controllers' bandwidth has a relation and by selecting one of them the other one will be driven out.

Table 6- 1 is the final values of critical parameters and results for Current PI controller shown below:

Table 6- 1: Current Controller Tuned Values and Parameters

Sampling Frequency[KHz]	Bandwidth[rad/s]	K_p	K_i
25	907	19.048	19000

The bandwidth of the current controller can be increased or reduced based on the application since the sampling frequency can be increased up to 100 KHz based on the design and code implementation

The next controller to be tuned is the speed controller with the parameters as Table 6- 2 shown below:

Table 6- 2 : Speed Controller Tuned Values and Parameters

Sampling Frequency [KHz]	Bandwidth [rad/s]	K_p	K_i	δ
1	50	0.0254	0.057	20

Now we are all set about the Current controllers and Speed controller, so in the next chapters after talking about the speed controller implementations, we will follow up the design for flux controller and then we will move to real results obtained by testing the motor in laboratory.

6-3-2 Implementation of Speed Controller

After closing the loops of currents on *"d"* and *"q"* axes, now is the time to implement the speed and flux controllers, which firstly here we will start with speed controller design and in the next chapter we will close the discussion by flux controller theory and implementation.

To start with speed controller implementation the very first thing which should be done is providing a proper feedback from the position of the rotor of the motor, in another word we need to be able to keep track of the changes accruing on the shaft of the motor in terms of position and speed because both of these elements are critical in our implementation of the FOC.

In order to provide the speed and position feedbacks there are two main methods:

1. Sensor-base methods: this method is based on using some kinds of sensors on the shaft of the motor to have the feedback from the speed and position of the shaft at each arbitrary time to be able to control these elements in a proper manner. One of the most used types of these sensors are known as incremental optical encoders which they provide some pulses with specific frequency depending on the speed of the motor.

2. Sensor-less methods: in these types of solutions the speed and position of the rotor of the motor is estimated by some sort of estimators or observers using the other feedbacks coming from motor like voltage or current feedbacks depending on the type of the motor and internal design of that.

There are pros and cons in both solutions and to come up with some comparison between these two methods we can point out the following differences:

1. **Ease of use and implementation:**

   of course, when it comes to use a method or solution for motor controlling with hardware considerations, we have to seriously evaluate the feasibility of those methods in terms of implementation and recourse consumption.

   In sensor-base methods as it's mentioned there is a need to provide another hardware beside the electrical motor to have the rotor motion feedbacks, by adding another hardware in the loop we are easing the task of controlling the motor in terms of implementation but at the same time adding another hardware makes the system more prone to hardware malfunctions and higher costs, in contradiction in sensor-less methods we won't have the issue of adding new hardware in the system but we will face other types of problems for acquiring good signals and powerful algorithms to detect properly the position and speed of the motors with the result of having a cheaper and broader solution.

2. **Type of the application**
   Depending on the type of the application it gets very important which methods you're intending to use, for example for applications with very small physical space like drones asking for motors with encoders is not feasible most of the times so the sensor-less methods are much more appreciated, on the other hand in very sensitive industrial machines with very high precision the need to a very accurate position feedback is inevitable since the cost or size are not considered as critical obstacles versus the precision.

There are other types of applications which the sensor-less and sensor-based methods are used beside each other to maximize the smoothness of motion control with use of cross-correction of these methods.

In this thesis we used a simple optical rotary encoder with 360 pulses per revolution to provide us with position and subsequently the speed feedback.

In Figure 6- 20 there is the whole process of controlling the speed of the motor used in this thesis using rotary encoder:

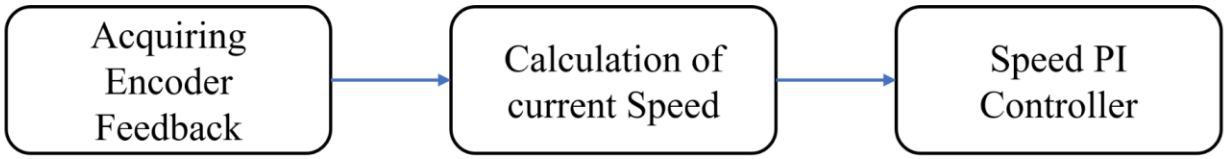

Figure 6- 20: Procedure of controlling speed in IM in FOC method

As can be seen in Figure 6- 20 the very first step is reading the encoder's output signals to be able to calculate the speed which is the rate of change in position in time in the simplest form. So firstly, we need to calculate in a fixed interval of time how many pulses we are acquiring from the encoder, then since the time interval "Δt" is known and within this time interval we are able to calculate how much the position has changed "Δx" so the speed is calculable using equation 6.4.1 below:

$$v = \frac{\Delta x}{\Delta t} \quad (6.4.1)$$

Equation 6.4.1 is the foundation of the speed calculation used in this thesis. Since in most of the applications speed is reported in revolution per minute (RPM), we decided to do the same, and all of the speeds are reported and converted into RPM.

In Figure 6- 21 the LABVIEW code for acquiring the encoder pulses in quadrature mode are shown as below:

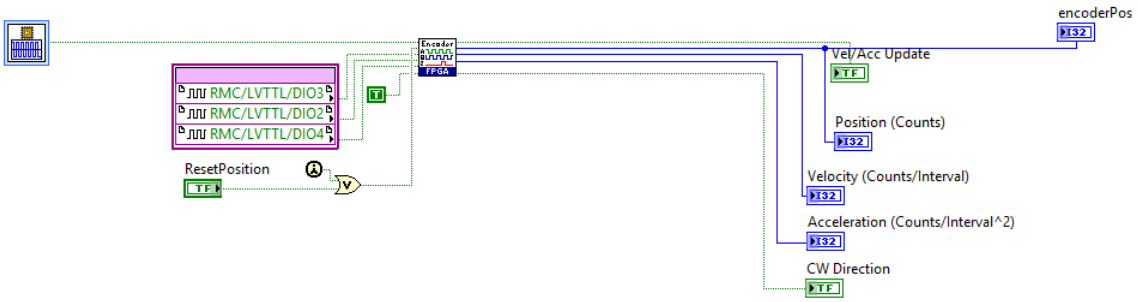

Figure 6- 21: Acquiring encoder signals in quadrature mode from the encoder

To give a short description of what's happening here, we have to say that, firstly most of the rotary encoders have 5 pins as below:

1. Two pins for the supply known as VCC and GND

2. Two pins for signal known as A and B signals which they have 90 degrees of phase shift and are generated based on the resolution of the encoder, as an instance for an encoder with 360 pulses per revolution resolution, we will have 360 pulses both on A and B. The phase shift in A and B signal helps to detect the direction of the rotation.

3. One signal pin known as Z which is generated once in every rotation, the main use of this signal is error correction and avoiding the accumulation of error

Inside Labview there are modules which they can read the encoder's pulses so the only thing we need to do is to wire properly some of the general purpose digital input pins of GPIC board into the encoder pins accordingly and read the signals in a cycle with known timing.

For instance, in Figure 6- 21 there is an encoder module which reads the encoder signals on every rising edge of a clock input to it, so here in order to make sure that we are not losing any pulses of encoder in quadrature mode we have to choose a proper clock based on the maximum speed and maximum frequency of pulse generation from the encoder which depends on the resolution of the encoder.

Here in our setup the maximum clock frequency to be able to safely acquire all the signals by considering that the maximum speed of the motor will be 1432 RPM which is around 150 rad/s, and since our encoder for each radians of revolution generates one pulse, so the maximum frequency required in quadrature mode will become 250kHz which is tuned in the clock generator module in LABVIEW.

In Figure 6- 22 there is the last process before passing the speed feedback to the speed controller which involves the speed calculation from the current and previous sample of position acquired from the encoder and converting this value into RPM.

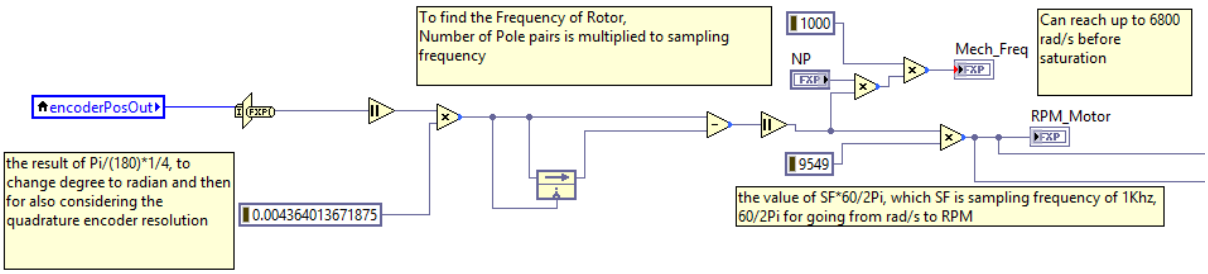

Figure 6- 22: Labview code for speed calculation and conversion of speed into RPM

The speed calculation loop works with 1kHz of sampling rate, so in every 1ms we calculate a new value for speed.

The PI speed controller is identical to the PI controller used for Current controller as in previous chapter and the only difference is that the PI controller for speed ran inside the speed loop which is executing with 1ms of loop rate.

The last thing about calculation of Speed in FPGA environment inside LABVIEW software is, since the size of the fixed-point variables is very crucial in final resource usage for FPGA, we have to design our system considering the maximum and minimum values for speed as well as the accuracy and resolution of data we would like to have, in other word having a very accurate speed controlling will subsequently demands higher data resolutions which ends up in more use of FPGA resources and less capability for other implementations, so one of the biggest challenges is optimizing the whole code for best and worst conditions to be able to have the most efficient code programmed inside FPGA.

6-4 Flux Controller Definitions and Loops

The flux control loop acts on current reference of *"d"* axis of FOC, so the Flux control loop can be depicted as in Figure 6- 23 below:

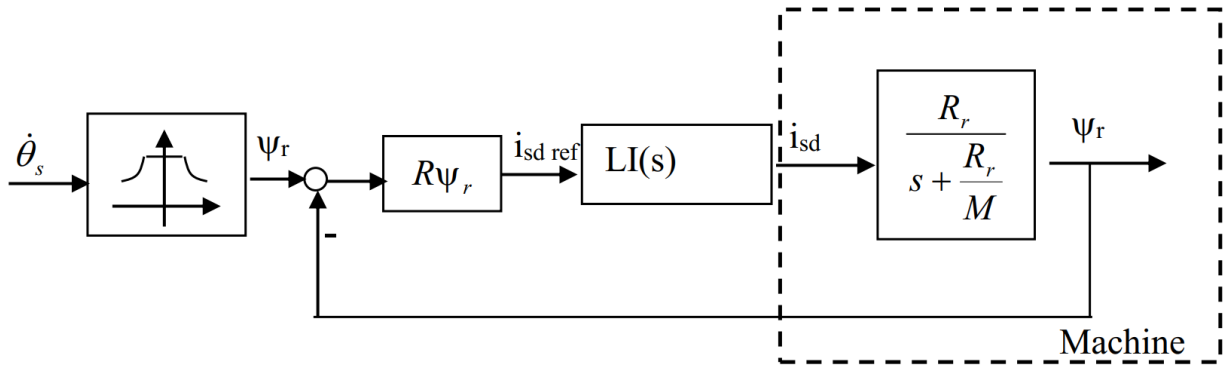

Figure 6- 23: Flux control Loop diagram in FOC (Dezza, 2017)

In this case, the transfer function of the controlled process depends on the relationship between the flux Ψ_r and the current i_{sd} and on the closed loop current control function as in equation (6.34) below:

$$B\Psi(s) = LI(s)\frac{R_r}{s+\frac{R_r}{M}} \quad (6.34)$$

So, the closed loop transfer function can be written as equation (6.35) below:

$$L\Psi(s) = \frac{\left(K_p\Psi+\frac{K_i\Psi}{s}\right).B\Psi(s)}{1+(K_p\Psi+\frac{K_i\Psi}{s}).B\Psi(s)} \quad (6.35)$$

Where the $K_p\Psi$ and $K_i\Psi$ are respectively the proportional and integral gains of the flux PI controller.

In Flux controller case, we need to estimate the rotor flux to be able to track and control it so in the next section we will introduce a type of state estimator which we used in this thesis to estimate the rotor flux Ψ_r as well as the stator electrical angle θ_s which is known as I-Ω estimator.

6-4-1 Rotor Flux I-Ω estimator:

Figure 6- 24 , shows the block diagram of a rotor flux estimator, called I-Ω. For the realization of this estimator, the phase currents measurements and the mechanical speed are necessary.

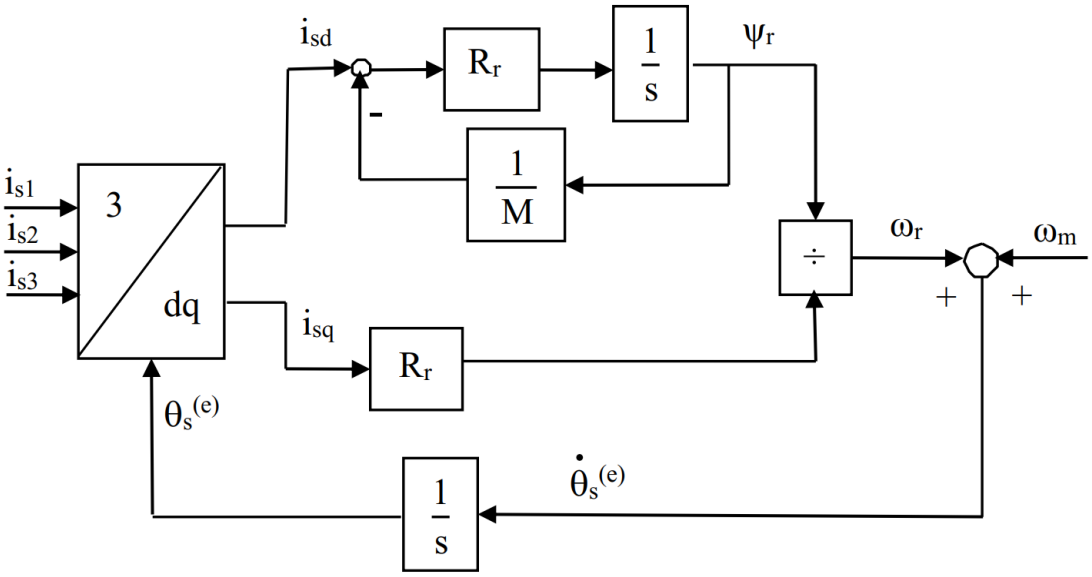

Figure 6- 24: I-Ω estimator of rotor flux in Induction machine with FOC (Dezza, 2017)

The diagram of Figure 6- 24 is driven from the equation (6.36) and (6.37) as below:

$$p\Psi_r = R_r i_{sd} - \frac{R_r}{M}\Psi_r \quad (6.36)$$

$$0 = R_r i_{sq} - \dot{\theta}_r \Psi_r = R_r i_{sq} - (\dot{\theta}_s - \omega_m)\Psi_r \quad (6.37)$$

So by some little modifications we can have:

$$\dot{\theta}_s = \omega_m + \frac{R_r i_{sq}}{\Psi_r} \quad (6.38)$$

$$\theta_s = \int \dot{\theta}_s dt \quad (6.39)$$

From the knowledge of the position of the rotor flux, it is possible, through the reference frame rotation of this angle, to calculate the two current components, from which we get the position angle and the value of the rotor flux.

The Simulink Model developed for rotor Flux estimation developed in MATLAB Simulink is as Figure 6- 25 which is developed based on the I-Ω estimation theory:

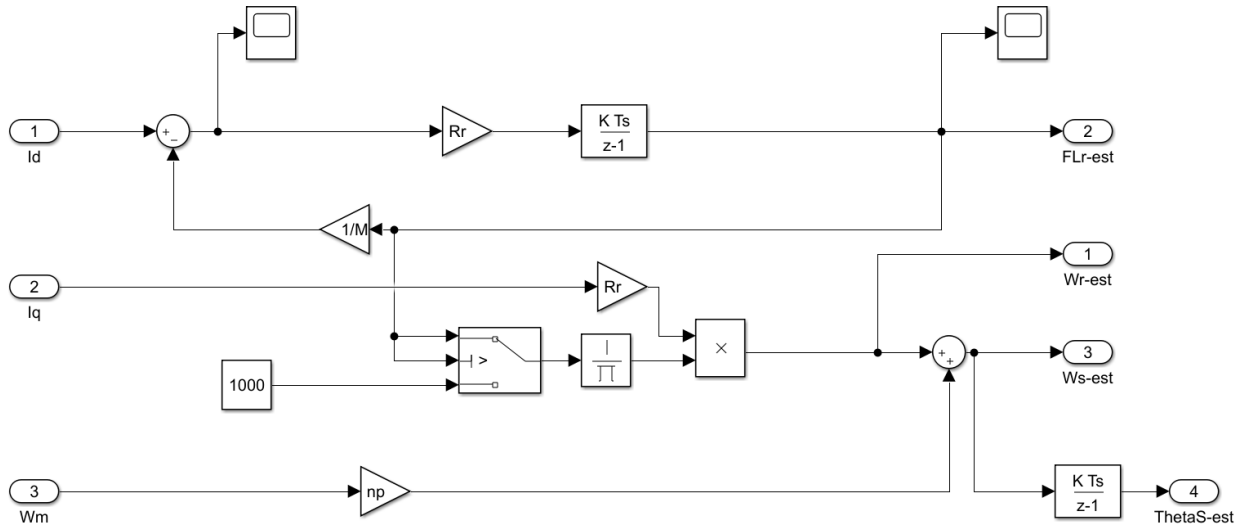

Figure 6- 25: Rotor Flux estimator block simulated in MATLAB Simulink software

6-4-2 Rotor Flux controller design

Rotor flux controller was designed using bode-plot tuning technique of the PI controller of flux loop such that we can have a bandwidth of 300 rad/s with phase margin of 75 degree. The bode diagram of the flux controller loop with the tuned PI controller is shown in Figure 6- 26 below.

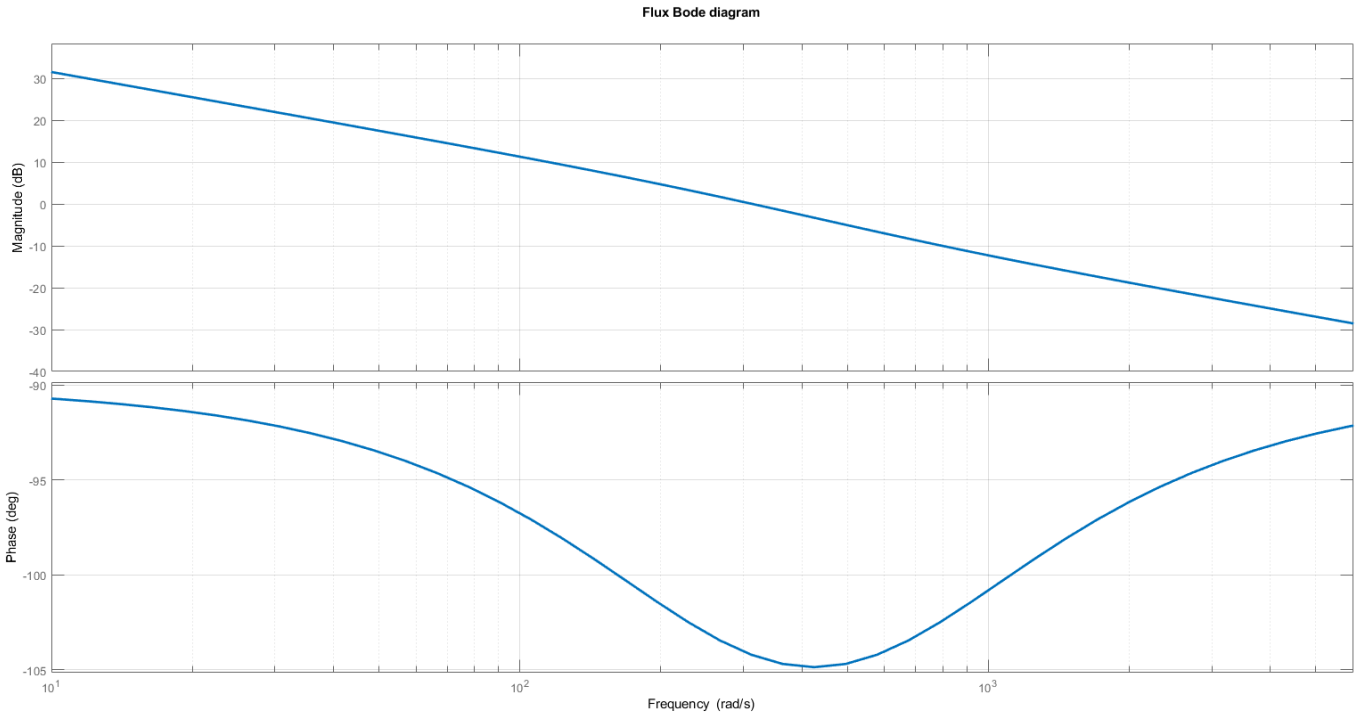

Figure 6- 26 : Bode plot diagram of Flux loop

By doing so, the Flux loop PI controller gains would be as Table 6- 3 , it worth mentioning that the flux controller was in the same loop as the speed controller, so the sampling frequency of the flux controller was the same as the speed controller as 1KHz, but due to the fact that the flux has faster dynamics rather than the speed controller the bandwidth of the flux controller was chosen higher to be able to control the flux disturbances better.

Table 6- 3: Flux controller PI controller gains

Sampling Frequency [KHz]	Bandwidth [rad/s]	K_p	K_i
1	300	26.5	14600

6-4-3 Rotor Flux estimation implementation

For estimating the rotor flux, we used the topology introduced in previous section as I-Ω estimator for LABVIEW code implementations this can be seen in Figure 6- 27 below:

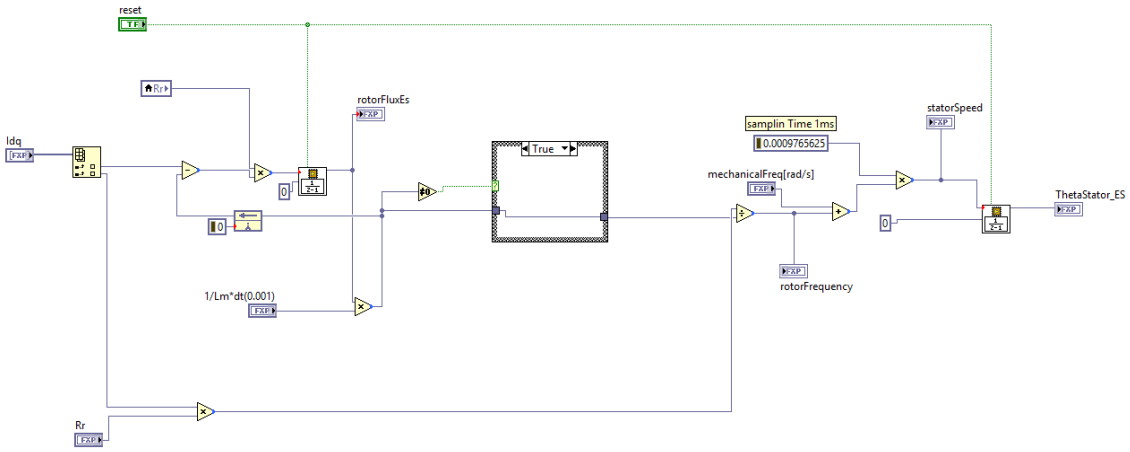

Figure 6- 27: Rotor Flux estimator LABVIEW code using I-Ω estimator

The main idea is exactly like the I-Ω estimator in continuous domain but since our integration is happening in discreet domain we have to multiply the integrators into the sampling time which is a very critical logical operation inside the FPGA, to explain this better we have to talk a little bit more about what happens when two variables are multiplied to each other in fixed point domain of the FPGA.

The fixed-point arithmetic for different operations are as below in equations (6.40) till (6.44) if we consider "A" and "C" as the word length and "B" and "D" as the integer word length in a fixed-point data type with the "+" or "–" as the sign of the variables:

1. **Addition** :

$$<^{+}_{-}, A, B> + <^{+}_{-}, A, B> = <^{+}_{-}, A+1, B+1> \quad (6.40)$$

2. **Subtraction** :

$$<^{+}_{-}, A, B> - <^{+}_{-}, A, B> = <^{+}_{-}, A+1, B+1> \quad (6.41)$$

3. **Multiplication**:

$$<^{+}_{-}, A, B> * <^{+}_{-}, C, D> = <^{+}_{-}, A+C, B+D> \quad (6.42)$$

4. **Division** :

Signed: $<\pm\quad, A, B> / <\pm\quad, C, D> = <\pm\quad, A + C + 1, B + C - D + 1>$ (6.43)
Unsigned: $<+, A, B> / <+, C, D> = <+, A + C, B + C - D>$ (6.44)

As can be seen in equation (6.40) and (6.41) the addition and subtraction have the minimum effect in terms of over-load on the variable sizes, so they will not cause so much of overflow or underflow for fixed-point variables if they are chosen in a proper way.

The biggest challenge is to manage the algorithms with multiplications or even harder when it gets into division because the chance of over-flow or under-flow of previously defined fixed-point variables is much higher.

In our implementation in Figure 6- 27 shown above the integrators and the fixed-point variables are chosen in a way that to keep the maximum possible accuracy with avoiding the overflow or underflow happenings on the output variables as well as keeping the variables' sizes as minimum as possible to save the FPGA resources.

As an example, a fixed-point variable with word length size of 64 bits can badly affect the resource managements inside the FPGA chip and most probably will cause errors in wiring and timings constraints in FPGA synthesis procedure.

In this implementation the rotor flux estimator was put inside a loop with loop rate of 1ms like speed controller with 1 kHz of sampling rate to provide proper data for the flux controller and also calculation of the stator position for park and Clark transformations inside other modules.

As can be seen in the whole loop of the speed and Flux controller is running in a separate loop in parallel with current controller as below in Figure 6- 28 :

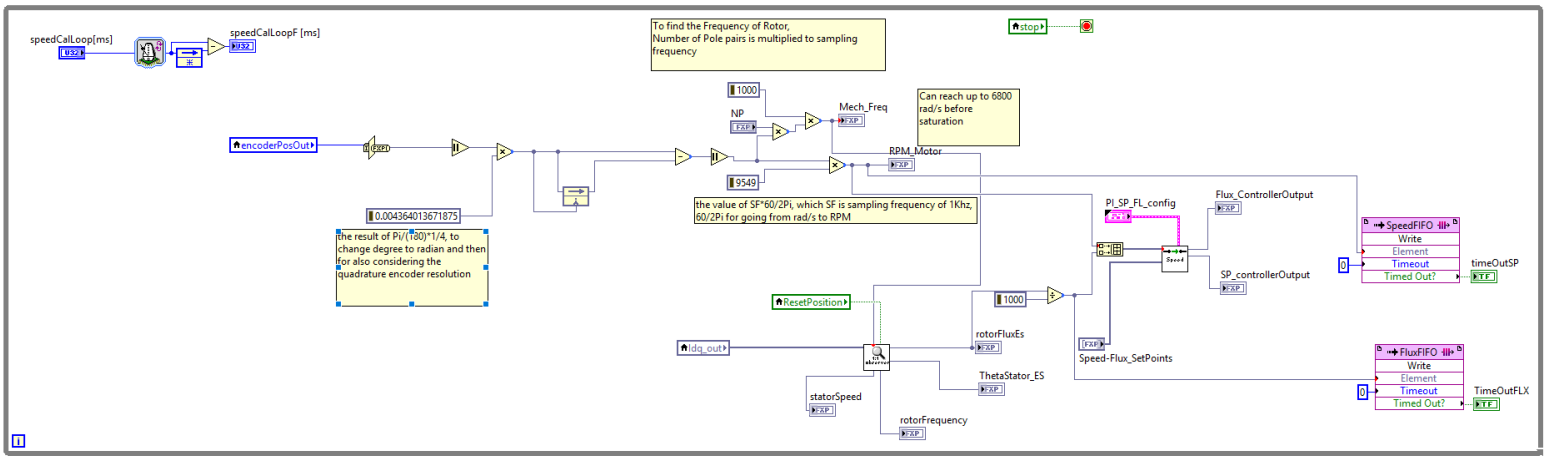

Figure 6- 28: Speed and Flux estimator and calculator Loop running in separate loop

One of the most interesting part of this design is this parallel control over different controllers, as a matter of fact since FOC controlling method by nature is a cascaded controller so eventually everything should happen based on an order, but the main thing here in FPGA environment is the controllers of Speed and Flux will provide data for Current controller loops without braking the operation of them.

In another word the current controller's actions as well as speed and flux controller are perfectly deterministic and independent from the tasks and timings of other parts of FPGA which is a major difference with other type of controller designs using structural hardware architectures like microcontrollers.

The speed and Flux controllers' outputs will be synchronized with the timing of current controller loop to be used as soon as they are ready and necessary, but since the current controller loop is working with much higher update rate in this thesis the synchronization will not become a critical issue because as soon as a new output from the speed or flux controller is generated the current controllers will regard them as the most recent reference to be followed instantly.

Chapter 7

7 Testing Results comparison and Analysis

Introduction

In this chapter the goal is to compare the results of FOC controlling method between the simulations done in MATLAB SIMULINK software versus the real results obtained in laboratory by testing the algorithm in real world condition. We will firstly start with the current controllers' results analysis and we will move up to flux and finally the speed controller results and analysis.

7-1 Current Controller Results on "d" axis of FOC:

In this test in Real world condition we firstly locked the motor rotor to avoid the shaft of the motor from any motion so that we could observer the step response of the current controller on the stator coils directly.

Figure 7- 1 shows the real-world step response of the current controller on "d" axis for a current reference of 0.5 Amps with 25kHz of sampling frequency:

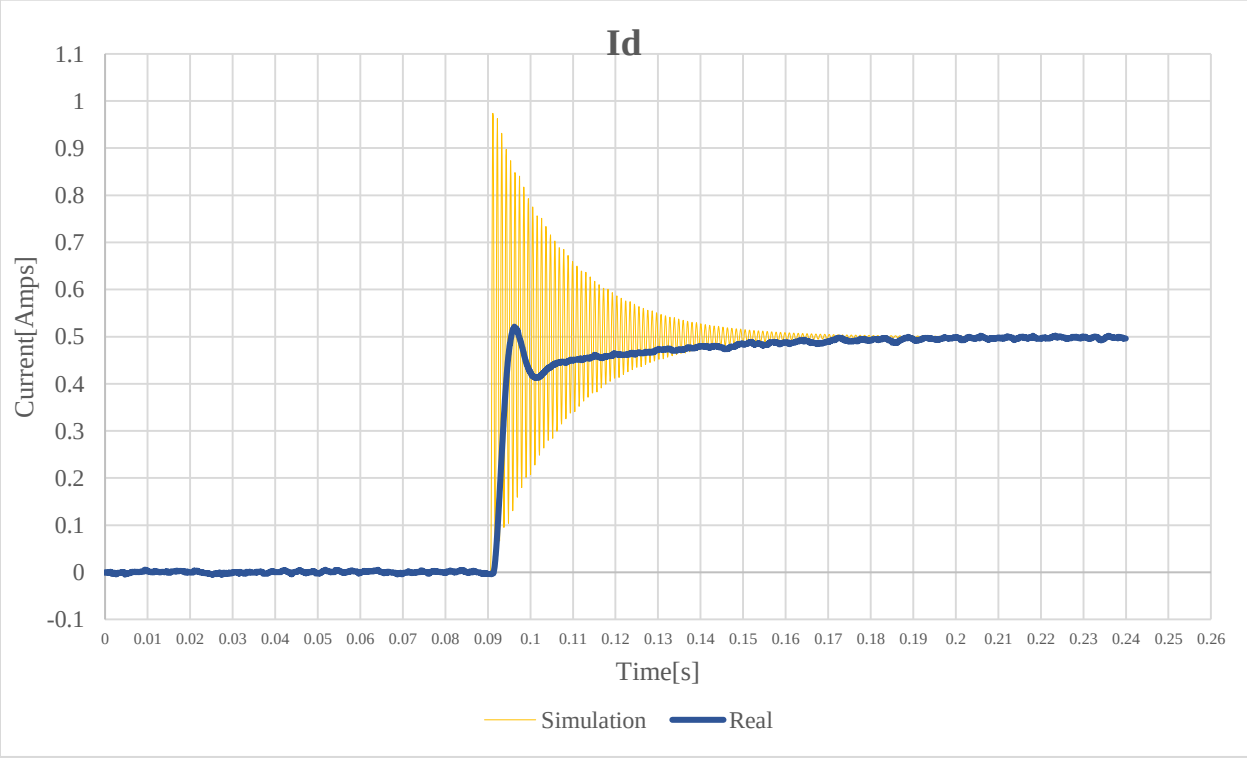

Figure 7- 1 : Step response of current controller on "d" axis for current reference of 0.5Amps

To do this test we fixed the rotor as it mentioned to avoid vibrations as much as possible on the shaft of the motor, so the current controller could follow the reference without any distortion or changes in the field due to rotation of the motor.

As can be seen in Figure 7- 1. since the rotor was fixed the current controller could follow the reference but in simulation since there are very small but considerable vibrations on the shaft of the motor we can observer some high frequency overshoots in the simulation response.

Beside this what can be clearly seen is the fact that the settling time of both controllers are matching and it’s around 110 milli seconds, so the controller will settle down within a same timing in both environments.

7-2 Current Controller Results on “q” axis of FOC:

In this test again like “d” axis of FOC we tried to fix the shaft of the motor for avoiding the rotor to have any kinds of motion, but since the “q” axis is the axis which is acting directly on the torque generated by the motor the fixation of the rotor is much harder and having some small vibrations is inventible when specially the observer is inside the loop of the control. So, to go on with comparisons, we will do two different types of tests, with and without the flux and stator position estimator in the loop.

We will start without the I-Ω observer and as can be seen In Figure 7-2 there is the comparison of the Simulation versus the real results obtained from the current controller on “q” axis for a current reference of 1 Amps with the sampling frequency of 25kHz from Current signal when the observer is deactivated, and the stator angle is fixed inside the software.

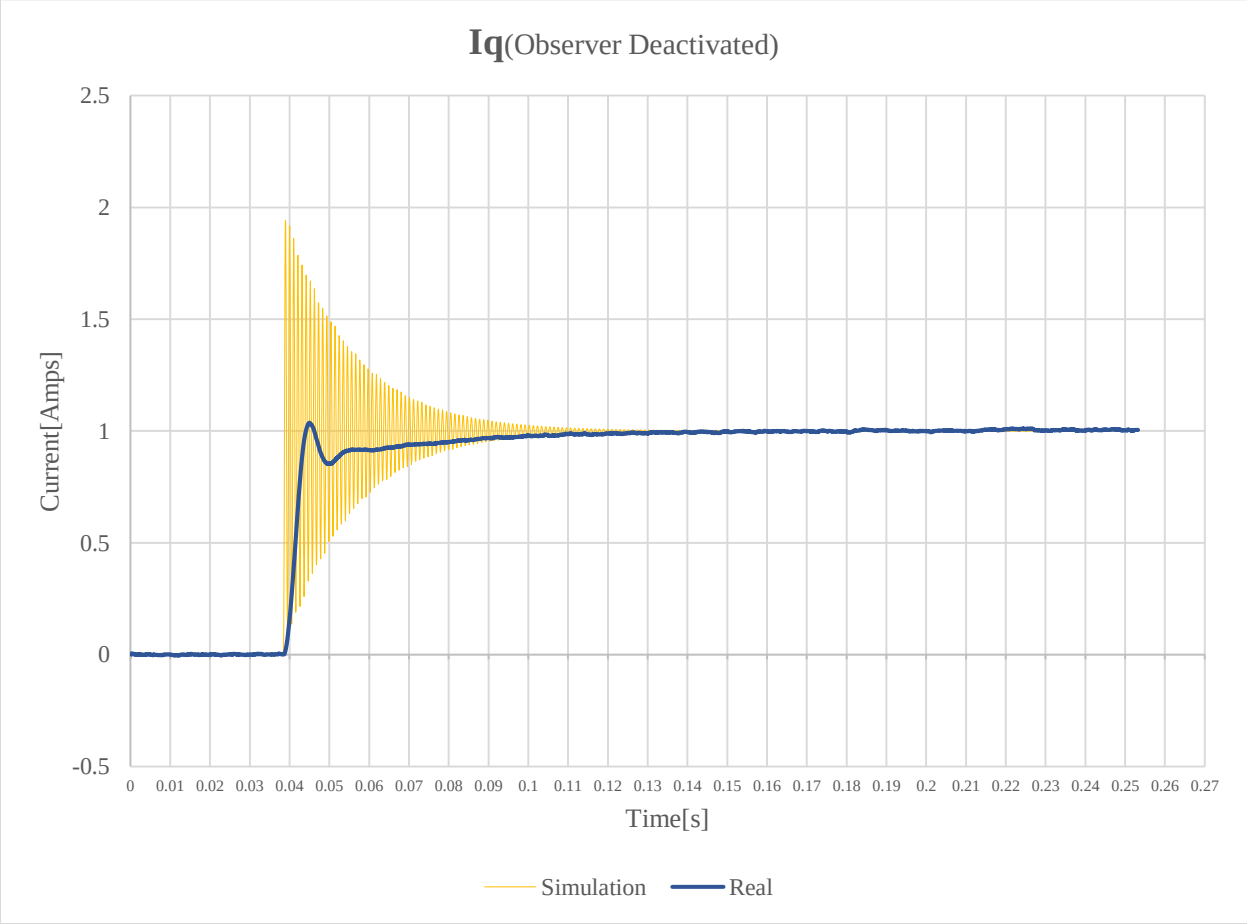

Figure 7- 2 : Zoomed Iq Current controller step response simulation results versus the real result with observer deactivated

As can be seen in Figure 7- 2 , the simulation signal has some high frequency oscillation which is due to the fact that in simulation locking the shaft of the motor is not feasible and eventually there are some high frequency vibrations on the shaft of the motor which causes this problem, but with looking carefully both of the step responses are having similar settling time which is around 110 milli seconds.

Now we performed another test with the observer in the loop, but in this case since the observer is active the amount of Current on the “q” axis will affect the observer and the motor will have much more vibrations when it’s locked with respect to the case where the observer is deactivated.

In Figure 7- 3 you can see the “q” current signal shape with similar inputs as the previous test:

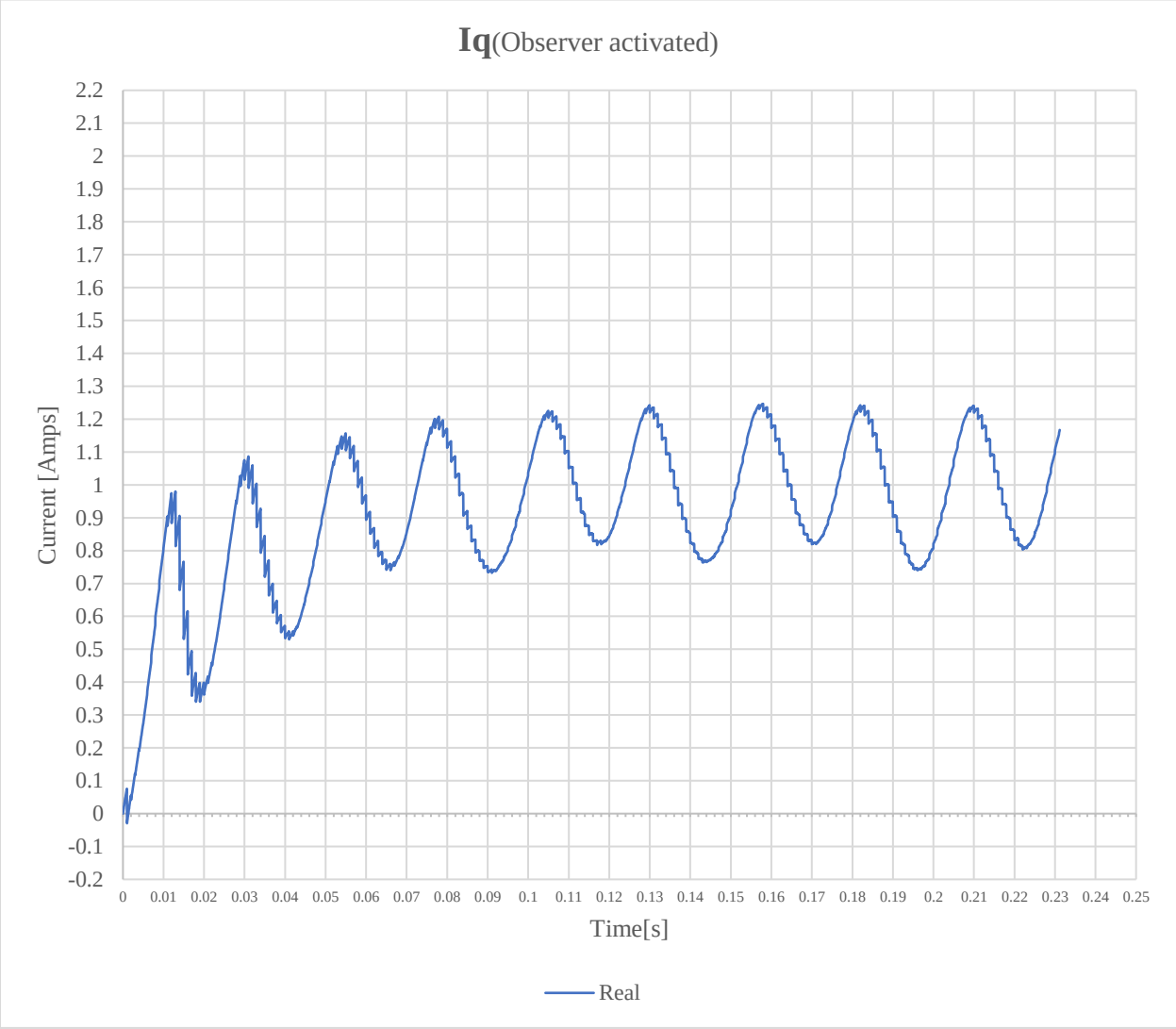

Figure 7- 3 : Current controller step response on “q” axis for a fixed reference when the observer is inside the loop

As can be seen in Figure 7- 3 , still within 110 milli seconds the controller is able to settle down around the reference value but with some oscillations due to the fact that now the stator position observer is active and is updating the value of stator position which results in reaction of the current controller on “q” axis which ends up in some oscillations around the reference since the value of stator angle is slightly changing due to stronger vibrations on the shaft of the real motor.

7-3 Flux Controller Results

The flux controller was tested by giving a fix reference of 0.03 Weber to the flux controller and keeping the speed controller input zero to keep the motor shaft still.

The result of the Flux controller results in simulation versus reality can be seen in Figure 7- 4 below which is done with 1kHz of sampling time from flux signals:

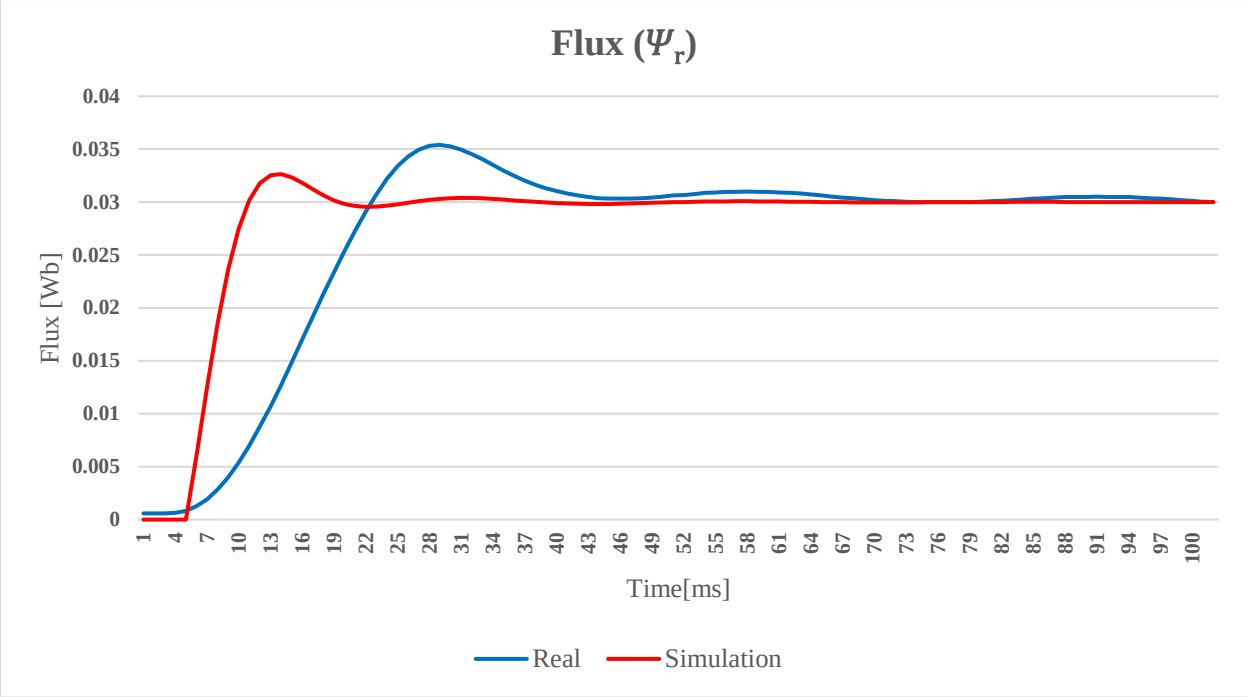

Figure 7- 4: Flux controller step response result in simulation and reality with a fix reference

As can be seen in Figure 7- 4 there is a slight difference in the responses, mainly in rise time and overshoot which the simulation signal has a smaller value in both.

The main cause of this phenomenon can be the error in the motor electrical parameters Identification, more specifically on estimation of rotor resistance "R_r" and magnetizing inductance "L_m" since both of these elements are contributing in the flux transfer function and controller design.

7-4 Speed Controller Results

The Speed controller is the final controller we've tested since it needs all the other controllers to function properly.

To test the speed controller, we gave a fixed reference of 500 RPM to the speed controller as well as a fixed reference of 0.03 Weber to the flux controller to keep the flux fixed during the motion of the rotor.

The result of the speed controller in simulation environment of MATLAB Simulink versus the reality can be seen in Figure 7- 5 below which is done with 1kHz of sampling frequency from speed of the shaft of the motor calculated thanks to an encoder mounted on the shaft:

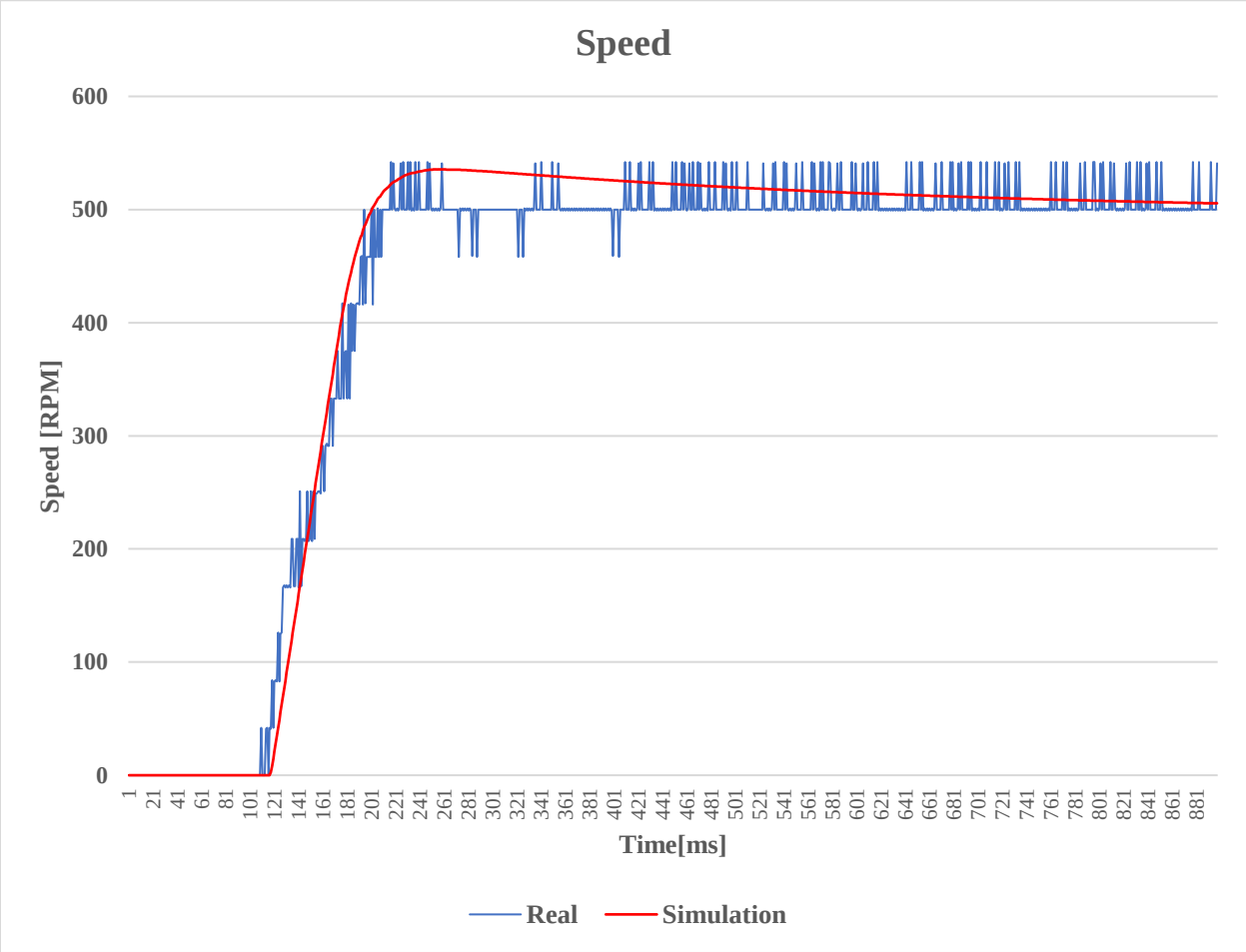

Figure 7- 5 : Speed controller step response results in simulation and real environment against each other

The speed controller as it's depicted in Figure 7- 5 has a very similar behavior to the simulation results which is an indication of accurate identification of the Inertia of the IM under test.

Chapter 8

8 Analytical comparison between FOC implementation on FPGAs versus Microcontrollers

Introduction:

In this chapter the goal is to discuss over the differences that will become significant when one is implementing FOC on microcontrollers and FPGAs. In another word we want to elaborate how using each of these solutions can affect the final results based on the needs of each project.

8-1 Parallel Processing versus Sequential Processing

As it was mentioned in the first chapter there is a fundamental difference between FPGAs and most of microcontrollers which is the ability of parallel processing or multitasking with FPGAs while microcontrollers or more precisely dominant majority of microcontrollers are not able to do a complete parallel processing of tasks and the tasks are eventually done sequentially. To understand better the difference, we can take a look at Figure 8- 1:

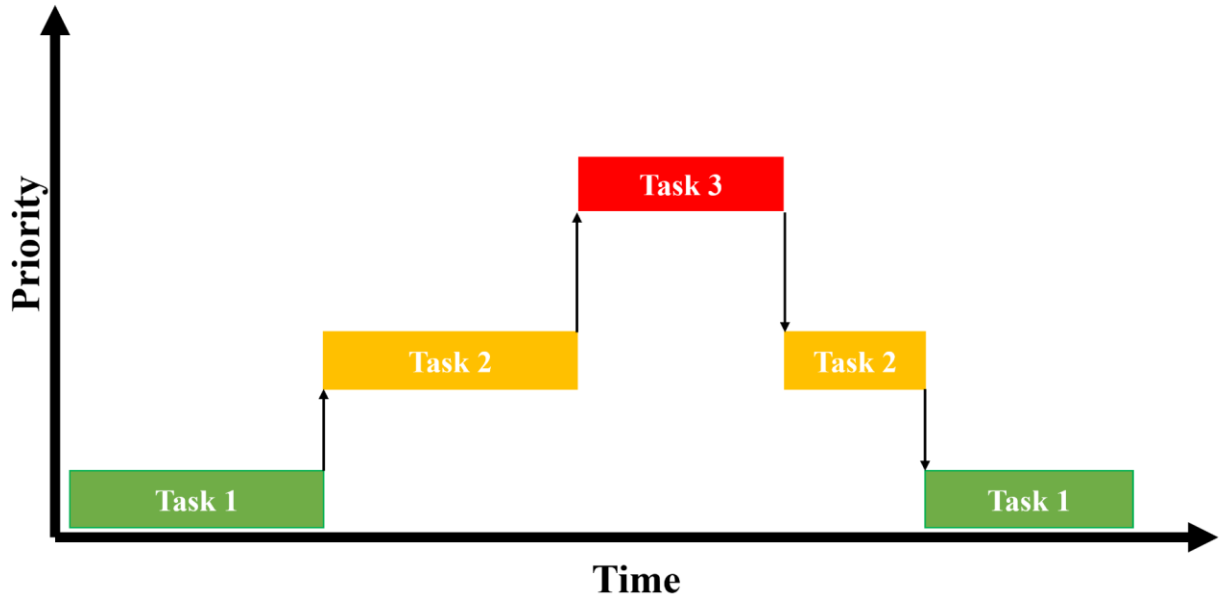

Figure 8- 1: Sequential Task execution and processing in Microcontrollers

Figure 8- 1 shows the task execution and processing inside the microcontrollers in sequential manner, as can be seen, for example Task 1 which is mostly known as background task has the lowest priority, so whenever there is a need of execution of a task with higher priorities, Task 1 will be stopped by the processor and the task with higher priority will be taken care of, then after finishing with tasks with higher priority we again come back to tasks with lower priorities. In the contrary inside the FPGA the tasks are done like Figure 8- 2 below:

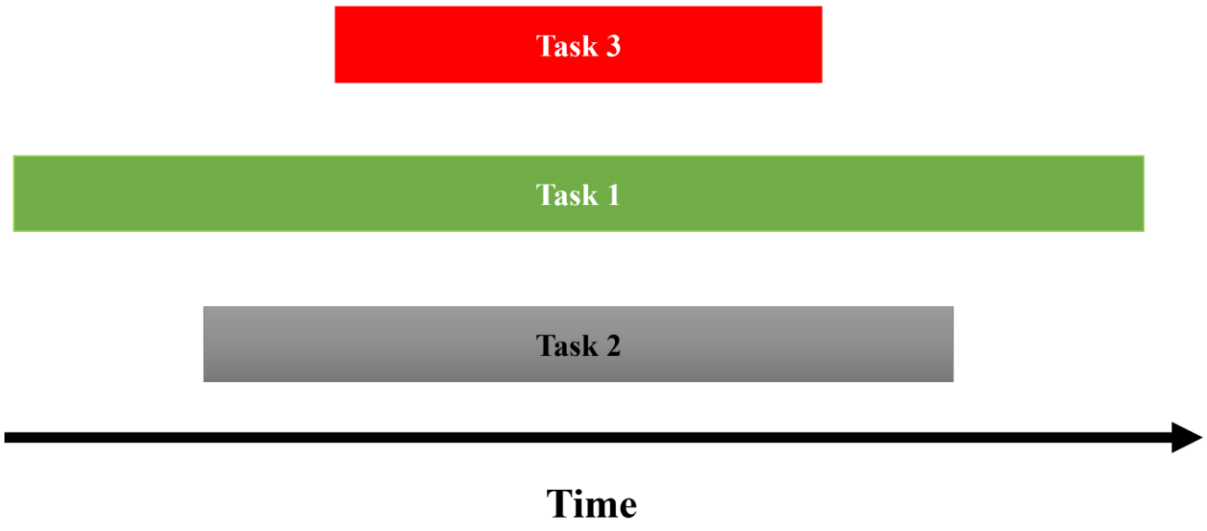

Figure 8- 2 : Task processing inside FPGAs

As can be seen in Figure 8- 2 inside FPGA naturally tasks can be done in parallel, so one task can follow its path as long as it doesn't interfere with other tasks with using of shared memory resources or in general racing conditions which should be avoided by the algorithms which are managing the tasks.

This is an advantage for FPGAs over microcontroller for highly sensitive tasks, like for instance acquiring highly accurate samples and processing them without pausing other tasks that are most probably dealing with the previously generated samples.

But the main question here is how can this effect FOC while we are using parallel processing feature of FPGAs versus sequential processing of microcontroller, so to answer this question first we will have a look on how FOC in general is done on microcontrollers and how it is done in this thesis on FPGA-based platform. To do this firstly in Figure 8- 3 you can see the architecture of FOC done in this thesis, as can be seen, FOC is managed with generally at least 4 main different processes going forward in parallel. The process 1 is dealing with rotor flux estimation and then the flux controller which controls the flux through the "d" axis of the current controller using process 2, the same thing happens for speed controller by acquiring the encoder feedbacks in process 4 and converting them into a value in round per minutes [RPM] unit range so that the speed controller can operate on the same inputs with the same unit and the output will affect the

current controller “q” axis output which will finally generates proper PWM pulses using PWM-generation in process 3 heading toward the switching board.

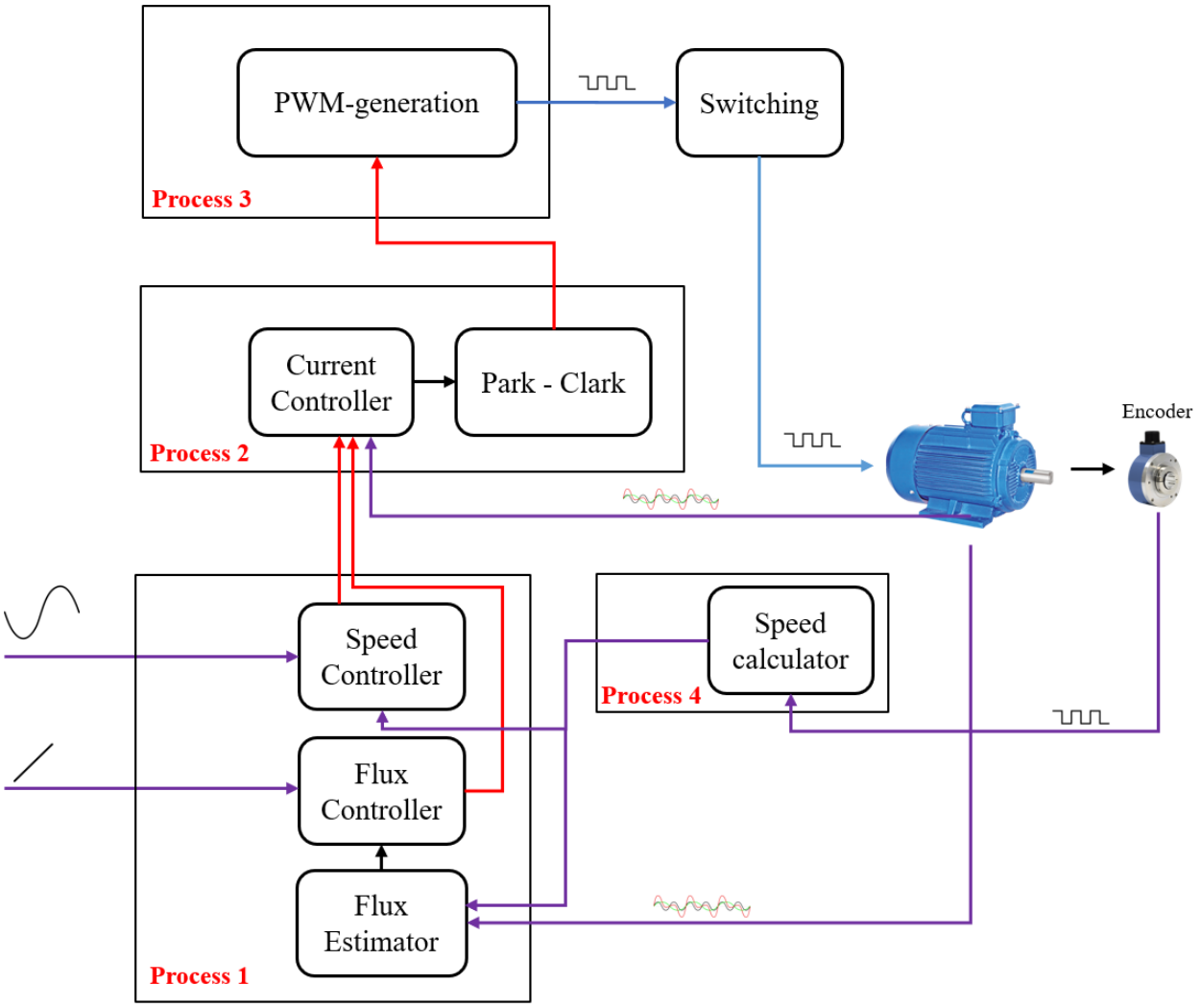

Figure 8- 3 : FOC architecture done on FPGA-based Platform

As in Figure 8- 3 there are at least 4 processes going on with different loop rates and the results are communicated between them by sending the most recent values or it can be done by sending them with handshaking methods and so on, without stopping each other from continuing their executions, in this thesis we used the first method by sending the most recent value to the processes which are demanding an input from other processes.

Using this methodology, the user can completely control the bandwidth of each process as well as execution speeds are completely independent from other processes, which this feature is not possible to achieve with single core microcontrollers, even in multicore microcontrollers like F28069M the interrupting behavior avoids perfect independency of processes from each other (Instruments T. , TMS320F2806x Piccolo™ Microcontrollers datasheet, 2017).

But for generic microcontrollers the story is totally different since as it was mentioned, vast majority of controllers are single cored and there is in action one processor dealing with all the loads and processing requests, so the processes are prioritized and taken care of with respect to their priority. To explain better how the microcontroller mange's different tasks of FOC with only one processor we can take a look at Figure 8- 4 below:

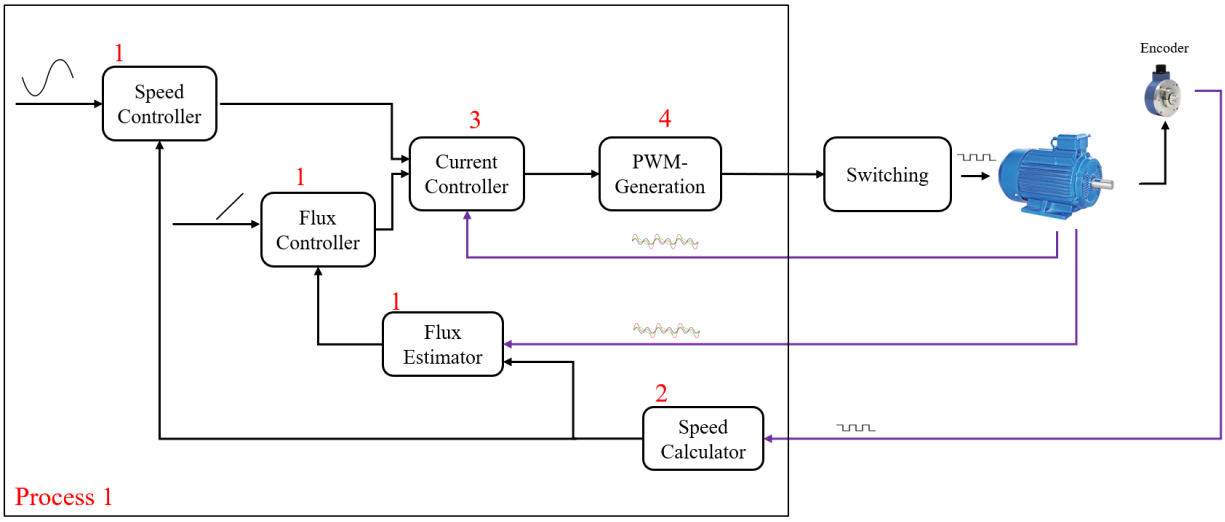

Figure 8- 4: FOC conventional architecture done on Microcontroller-based solutions

As can be seen in Figure 8- 4 there is one core of processing with different tasks prioritized inside, each task has a priority which is indicated as an instance with a number on top of them, in this case the higher the priority the more important is the task for the processor to finish with before processing other tasks with lower priorities.

In Microcontrollers mostly the PWM generation task which is made by a timer and a counter inside, has the highest priority and it kind of synchronizes other tasks like current sampling and control. As long as the higher priority task is being processed other tasks are stopped and waiting to be processed inside a que, so in microcontrollers in order to make the whole processing feasible the tasks with higher priority must be done within a short time frame deadline, otherwise the tasks with lower priority will never find a chance to be processed.

To elaborate this more by looking at Figure 8- 5 we can understand how each kinds of tasks in motor controlling demands different processor loads and also the execution frequency in a microcontroller-based solution (Instruments T. , MotorWare Software Architecture, 2013). As an instance in Figure 8- 5 first of all τ_m stands for mechanical time constant and respectively τ_e stands for electrical time constant. Generally speaking in most of the motors the value of mechanical time constant is higher than the value of electrical time constant so the effective mechanical time constant $\frac{1}{\tau_m}$ is smaller than effective electrical time constant $\frac{1}{\tau_e}$ as considered below.

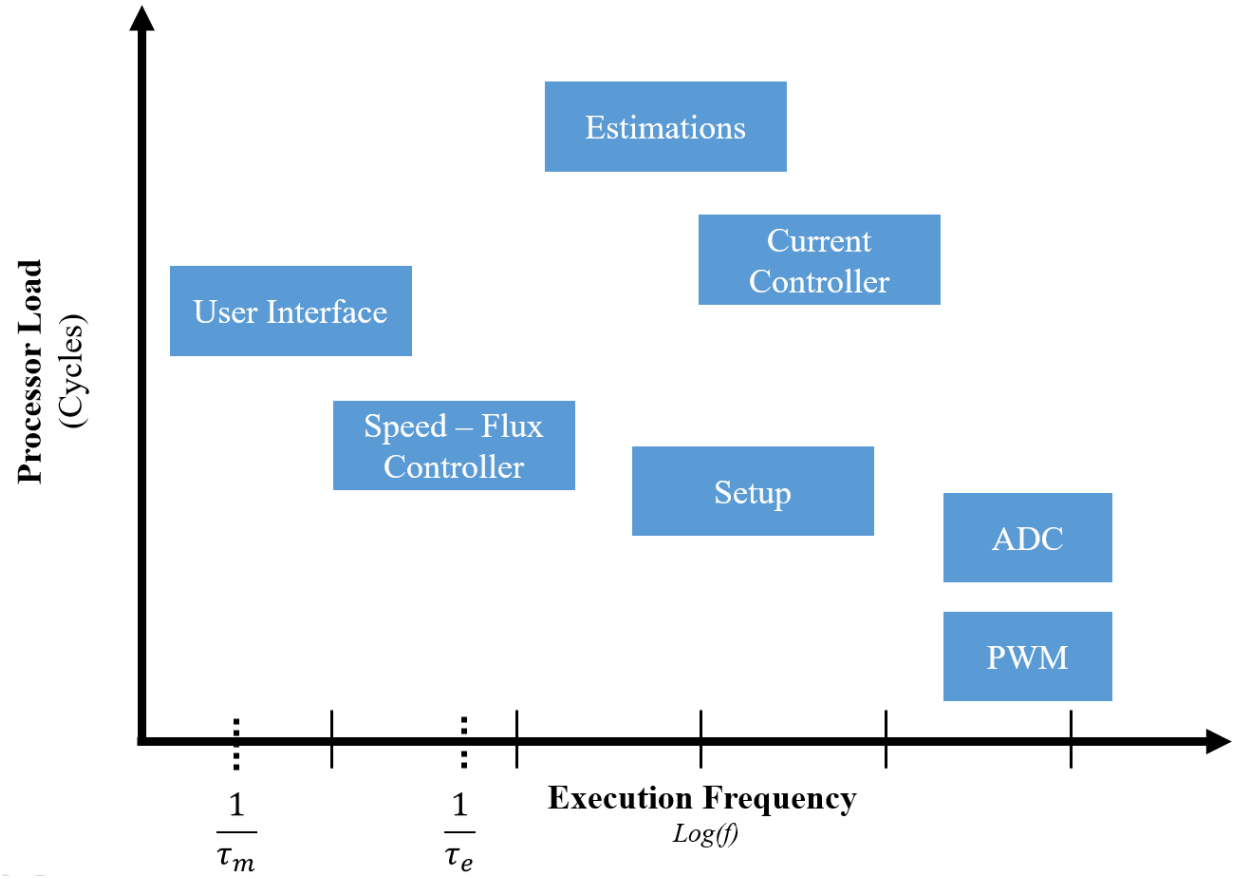

Figure 8- 5: Execution Frequency versus Processor Load in Microcontroller-based solution (Instruments T. , MotorWare Software Architecture, 2013)

The speed controller typically runs 5x to 10x the effective mechanical frequency of the motor to ensure an acceptable speed response. Likewise, the current controllers run 5x to 10x the effective electrical frequency of the motor. Some applications implement a user interface such as a GUI to provide desired set points for position or speed, to observe internal variables, etc. but this module usually runs at a fairly slow frequency (Instruments T. , MotorWare Software Architecture, 2013).

In addition to the conventional speed and current controller, most modern control systems have some type of estimation, whether it is initial parameter estimation, real time parameter estimation or both. These algorithms can be quite sophisticated and can use most of the processor horsepower when viewed on a percentage algorithm usage basis. A/D sampling and data conversion along with PWM are typically the fastest frequency events in the system but they do not require much compute horsepower. However, the entire control system is usually synchronized to these events and they will provide the periodic clocking mechanism for executing the other algorithms. Finally, periodic setup code is needed to modify parameters during control system execution. This code typically does not burden the system with a large compute load but must still execute at a high enough rate to impact the algorithms that rely upon the updated parameters.

Depending on type of the applications there can be some bold differences between FPGA-based solutions versus microcontroller-based solutions considering the parallel processing versus sequential processing features on FOC implementation.

One of the scenarios which can happen is, for electrical motors with very fast mechanical dynamics, τ_m with respect to τ_e becomes considerable and close, then 3 cases of problems may arise:

1- The speed controller bandwidth must be increased and this puts speed controller closer to current controller in plot of Figure 8- 5, meaning that the speed controller for tracking different speed references and best possible rejection of speed disturbances needs to be run at much higher frequency and bandwidth, this can make a conflict with current controller process management because the CPU has to process these loops which both have close bandwidths and sampling time.

2- For motors with high frequency torque disturbances, the current controller may face some issues, suppose that in the times that the processor is dealing with speed controlling process management we have some torque disturbances, but since the value of the current controller reference which is the output of the speed controller is remaining the same (since the process of speed controller has not finished yet for execution of new possible reference), the current controller will act based on the most recent output of the speed controller as the reference, this can become an issue when one needs a very high quality torque and subsequently speed control on the motor. This problem can be enhanced in FPGAs with parallel speed controller to current controller by increasing the speed controller loop rate and most possibly bandwidth without having a conflict with other controllers because it's running in parallel, so the current controller would be able to receive more updated reliable references and tackle better the torque and speed disturbances.

3- When the mechanical dynamics becomes fast the estimator's bandwidth and sampling frequency will increase respectively which is identical to more loads with closer execution frequency to each other, which again in microcontroller-based solutions is problematic and hard to manage.

8-2 deterministic execution versus non-deterministic execution

The other critical differences between FPGA-based solutions versus microcontroller-based solutions for implementation of FOC can be raised in time determinism, which technically means that how much it's possible that a task is done within a time frame and not more than that in simple words, also there is a stricter definition which is, doing a task every time exactly with the same timing all over the execution. Both of these definitions are possible to reach inside the FPGA but not in microcontrollers and in the following we will talk about the reasons.

First, we will start to describe the situation for microcontrollers, because of using interrupts, microcontrollers can accomplish doing tasks with different priorities, this feature is severely affecting a perfect time determinism inside microcontrollers. To understand this better you can take a look at Figure 8- 6 for instance which is the architecture of motor controlling done in Motorware libraries by Texas Instruments on different families of Texas instrument microcontrollers.

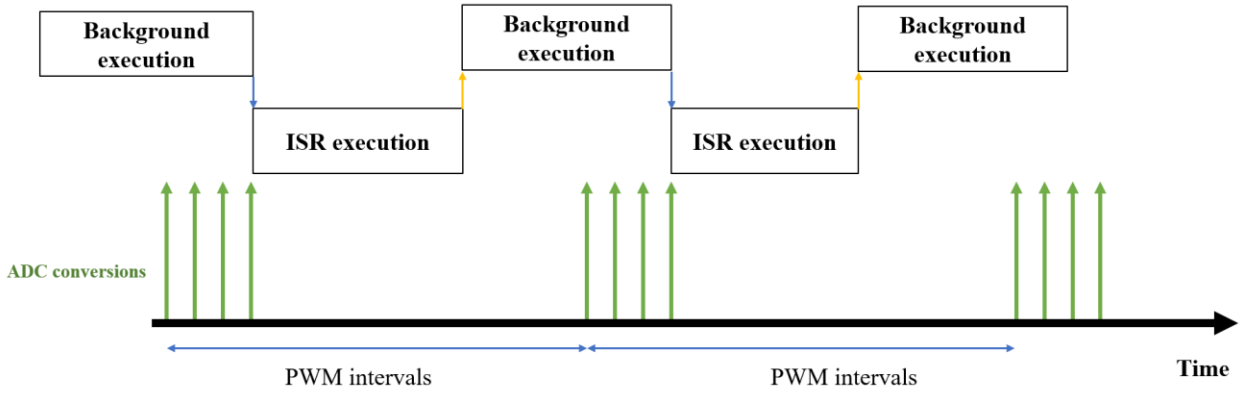

Figure 8- 6 : Software execution time and resource management in Motorware libraries (Instruments T. , MotorWare Software Architecture, 2013)

In Figure 8- 6 there is a main ISR which is running with the fastest possible speed and it's only limited by the PWM frequencies, Since all the software in the main ISR function must complete before the next interrupt, the timeline for the real-time software execution will look something like the one shown in Figure 8- 6 . After the last A/D conversion, an interrupt is generated and the main ISR function executes. It is easy to conclude that the ISR frequency, and thus the PWM frequency, is limited by the execution time of the main ISR function and the A/D conversion time. However, since not all of the software in the control system needs to be executed at the ISR rate, it could be run outside of the main ISR function. The partitioning of the software based upon required execution frequency becomes the motivation for more sophisticated schemes of prioritizing software execution.

The software architecture with the next level of complexity is one with two threads. One thread is commonly called the foreground task, which is high priority, and one thread is commonly called

the background task, which is low priority. All of the software that needs to be executed at a high frequency remains in the main ISR and all of the software that can be executed at a lower frequency is placed in a second function that runs in the background. Whenever an interrupt is received, the background function is interrupted by the ISR, the ISR runs to completion and then the background function continues to execute. It is not difficult to visualize that the background function must contain some sort of infinite loop to provide recurring execution functionality for the background software.

The timeline for the real-time software execution for a background loop and a main ISR function will look something like the one shown in Figure 8- 6 . It is clear to see that the background execution gets interrupted by the main ISR at each interrupt instant, which occurs after the A/D conversions. While this architecture will allow for a higher interrupt frequency (and thus a higher PWM frequency) than a single thread architecture, the maximum achievable interrupt frequency is still quite limited because of the computational needs of the ISR function (Instruments T. , MotorWare Software Architecture, 2013).

Considering all the above said conditions, the perfect time respecting for any kind of the task executions inside microcontrollers is fragile and can be easily broken by a high priority interrupts, which in case of FPGAs this can be avoided due to the fact that we can run processes in parallel and insulate them from each other.

By existence of time determinism, we can make sure that all the control outputs are generated every time with a fixed frequency and this can help to improve the quality of control. Inside FPGAs specially in Labview environment thanks to timed loops or single cycle loops we can have the guarantee that the processes inside are done exactly each time on the specified timings.

8-3 simplicity and cost effectiveness

Beside all the brilliant features that FPGA-based solutions are providing for FOC, the biggest drawbacks of the solutions based on FPGAs can be the high cost of them with respect to the microcontroller-based solutions, especially for small scale products. The main reason for this is the FPGA-based solutions hardware-wise are very complicated to design and they require a very high-level knowledge of high speed circuit design. For instance, to design a circuit based on FPGA, the designer should be able to use ADC modules, Memory module, communication modules and so on to be able to make a circuit which is able to manage a simple controlling algorithm; while in microcontrollers all of these features are existing internally, and the user don't need to implement them.

Another issue here is the software which are capable of communicating with FPGAs and programming them in a way that the user can concentrate on high level block diagram design rather than writing line by line VHDL codes, these software are very expensive and not accessible to all the people out there, so this seems another big obstacle in front of the FPGA-based solution developers, while for microcontrollers there are hundreds of thousands of software and solutions which mostly are free and available and the users can easily access them with much less difficulties with respect to FPGA-based programs.

In general, there is no complete answer to the question which asks which one is better? Microcontroller-based solutions or FPGA based solutions? The answer to this question depends completely on the system under development and the requirements, so it will be defined case by case. For instance, for solutions inside very intensive and risky environments the FPGA-based solutions can be a good choice due to higher degrees of design freedoms which the developers will have, which most probably can lead them to a safer design of the system since all the functionalities can be controlled and managed easily based on the flow of the software they design and have control on, on the other hand for projects which cost and simplicity is crucial, of course microcontroller-based solutions can be far more better and feasible.

Conclusion and Future Perspectives:

In this thesis we addressed the study and development of FOC in parallel processing environment within a FPGA-based solution on an Induction Motor with analytic analysis of the differences between FPGA-based solutions versus conventional microcontroller-based solutions.

To do this we firstly selected the proper hardware and then by providing and designing customized interfaces between them we transformed a microcontroller-based hardware into our desired FPGA-based solution. At the next stages by selecting a generic Induction Motor and identifying the motor parameters we controlled the motor using filed oriented control algorithm with nested loops of Current, speed and flux control. All the implementations were alongside with proper simulations before, so we could make sure that our algorithms will lead us to our expected behaviors. At the end, the results of simulations were compared to the real results obtained in real environment. In the last chapter we analyzed the possible differences that a FPGA-based solution can have with a microcontroller-based solution dealing specifically with FOC.

One of the future developments which can lead to very interesting results could be testing in real environment FOC on microcontroller-based solutions versus FPGA-based solutions for fast dynamics motors, either electrical or mechanical dynamics, since these types of motors can challenge the field-oriented control nested loop design in terms of implementation of controllers and parallel processing versus sequential processing tests. So, there would be the possibility to test in action the performance of the FOC in FPGA-based solution versus microcontroller-based solutions and practically analyzing that under what conditions the differences can become significant and in contrary under what conditions the difference is negligible.

Bibliography

Dezza, F. C. (2017). Vector Control of Induction Motors. In F. C. Dezza.

Fidan, B. I. (n.d.). Adaptive Control Tutorial. *Society for Industrial & Applied Mathematics*.

Giri, F. (2013). *AC ELECTRIC MOTORS CONTROL- ADVANCED DESIGN TECHNIQUES AND APPLICATIONS.* University of Caen Basse-Normandie France.

Instruments, N. (n.d.). *User Manual and Specifications NI 9683 General Purpose Inverter Controller RIO Mezzanine Card.* National Instruments.

Instruments, T. (2013). *MotorWare Software Architecture.* TI.

Instruments, T. (2017). *InstaSPIN-FOC™ and InstaSPIN-MOTION™.* TI.

Instruments, T. (2017). *TMS320F2806x Piccolo™ Microcontrollers datasheet.* TI.

Leonard, W. (2001). *Control of Electrical Drives.* Springer,.

Rik De Doncker, A. V. (2011). *Electric drives analysis, modeling, Control (power Systems).* Springer.