Anders Moldskred

FPGA-Based Real-Time Emulator of Permanent Magnet Synchronous Motor

Master's thesis in Electric Power Engineering
Supervisor: Roy Nilsen
June 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

NTNU
Norwegian University of
Science and Technology

Anders Moldskred

FPGA-Based Real-Time Emulator of Permanent Magnet Synchronous Motor

Master’s thesis in Electric Power Engineering
Supervisor: Roy Nilsen
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

Problem Description

The goal of the project is to develop IP cores for use in a real-time emulator of a permanent magnet motor running on an FPGA. This emulator will be used for development and testing of control software for permanent magnet motor drives. By emulating the dynamic behaviour of the motor and mechanical load the emulator will allow for testing of the drive control system without using a physical test setup.

The IP cores are to be developed using Xilinx System Generator for DSP and tested in Simulink.

The tasks to be performed in the thesis are:

- Develop discrete-time equations for three-phase and six-phase PM synchronous motors and mechanical load.
- Develop IP cores to implement discrete-time motors using System Generator for DSP by Xilinx.
- Test IP cores in Simulink by implementing them into a Simulink motor drive system model and compare simulation outputs to equivalent continuous-time system model.
- Identify differences between the discrete-time models using the IP cores and the continuous-time models.

Supervisor: Prof. Roy Nilsen
Co-supervisor: Aravinda Perera

Preface

This thesis report is the final work in my master degree in "Electric Power Engineering" at the Department of Electric Power Engineering at NTNU.
Working on this project has been has been very challenging but equally rewarding. It has given me valuable insight in a range of topics, especially in programming and digital systems.

I would like to thank my supervisor, Roy Nilsen for sharing his knowledge of motor drives and digital systems and for great help in the interpretation of the simulation results. I would also thank my co-supervisor Aravinda Perera for his invaluable assistance in the IP core programming and for providing feedback and encouragement.

Anders Moldskred

June 24, 2020
Trondheim

Abstract

The thesis is part of a project at the Department of Electrical Power Engineering at NTNU to develop a real-time FPGA-based emulator of a permanent magnet motor drive running on the NTNU Programming Platform.

Real-time emulation of motor drive systems allows for development and testing of digital drive control systems without requiring any hardware test setup, which reduces the cost and equipment needed for development. With advancements in FPGA technology and high-level synthesis tools for HDL programming, this is becoming an increasingly popular and accessible development tool for industrial motor drive control systems.

In this thesis, IP cores for emulation of three-phase and six-phase permanent magnet synchronous motors are developed using Xilinx System Generator for DSP and tested in Simulink. The simulation results from the three-phase and six-phase IP core based emulators are compared to the outputs from continuous-time motor models in Simulink with identical parameters. The effects of changing the discrete sampling time of the IP cores and the FPGA clock speed on the accuracy of the simulation are examined.

The results from the simulations show that the three-phase motor drive system model containing the discrete-time IP cores perform very similar to the continuous-time reference model. In the six-phase model however, there are some oscillations in the electrical torque of the IP core model which are not present in the reference model. The amplitude of the oscillations is reduced by reducing the discrete sampling time on the IP cores. Increasing the FPGA clock period from 10ns to 100ns, thereby reducing the clock frequency, also seem to dampen the oscillations somewhat. An analysis of the eigenvalues of the motor models show that the discrete system is stable, but may be poorly damped. Comparing the frequency of oscillations to the damped frequency of the eigenvalues at different motor speeds, it is found that they are similar, which indicates that the oscillations are linked to the eigenvalues.

Further analysis of the complete drive system, including the control loops is needed to determine the cause of the oscillations in the model containing the IP cores.

Sammendrag

Denne oppgaven er del av et prosjekt på Institutt for elkraftteknikk ved NTNU for utvikling av en sanntids FPGA-basert emulator av en permanent-magnet synkronmotordrift som skal kjøres på NTNU Control Platform.

Sanntids-emulering av motordrifts-systemer gjør det mulig å utvikle og teste det digitale kontrollsystemet uten å ha tilgang på en fysisk testrigg. Dette er med på å redusere kostnadene og utstyret som trengs for utvikling av kontrollsystemet for motordriften. De siste årene har fremskritt innen FPGA-teknologi og nye programvareverktøy for høy-nivå HDL-programmering gjort dette til et stadig mer populært og tilgjengelig utviklingsverktøy for motordrifter i industrien.

I denne oppgaven har det blitt utviklet IP-kjerner for emulering av trefase og seksfase permanent-magnet synkronmotorer ved bruk av Xilinx System Generator for DSP. Disse IP-kjernene har deretter blitt testet ved simulering i Simulink. Simuleringsresultatene fra trefase og seksfase IP-kjernebaserte emulatorer sammenlignes med resultatene fra referansemodeller laget i Simulink med identiske parametere. Effektene av å endre den diskrete sampling-tiden til IP-kjernene og FPGA-klokkehastigheten på nøyaktigheten til simuleringene blir undersøkt.

Resultatene fra simuleringene viser at trefase-modellen der IP-kjernene er implementert som motor og mekanisk last gir omtrent samme resultat som referansemodellen. I seksfase-modellen er det imidlertid en del oscilleringer i det elektriske dreiemomentet til IP-kjernemodellen som ikke er til stede i referansemodellen. Amplituden til oscilleringene reduseres ved å redusere den diskrete sampling-tiden på IP-kjernene. Å øke FPGA-klokkeperioden fra 10 ns til 100 ns, og dermed redusere klokkefrekvensen, ser også ut til å dempe oscilleringene noe. En analyse av egenverdiene til motormodellene viser at det diskrete systemet er stabilt, men kan være dårlig dempet. Ved å sammenligne frekvensen av oscilleringene med svingefrekvensen til egenverdiene ved forskjellige motorhastigheter er det funnet at de er ganske like, noe som styrker teorien om at oscilleringene oppstår som følge av egenverdiene i systemet.

Ytterligere analyse av det komplette motordrifts-systemet, inkludert kontrollsløyfene, er nødvendig for å bestemme årsaken til svingningene i modellen som inneholder IP-kjernene.

Table of Contents

List of Figures

List of Tables

Abbreviations

AC Alternate Current.

ASIC Application-Specific Integrated Circuit.

CPU Central Processing Unit.

DC Direct Current.

DSP Digital Signal Processing.

FPGA Field Programmable Gate Array.

IPMSM Interior Permanent Magnet Synchronous Motor.

LTE Local Truncation Error.

PMSM Permanent Magnet Synchronous Motor.

pu Per Unit.

PWM Pulse Width Modulation.

VSD Vector Space Decomposition.

List of Symbols

List of symbols, superscripts and subscripts used in the thesis.
Lowercase symbols indicate pu values.

Symbol	**Explanation**
R_s, r_s	Stator resistance
I_s, i_s	Stator current
U_s	Stator voltage
Ψ	Flux Linkage
R_f	Fictitious field winding resistance
I_f	Fictitious field winding current
I_f	Fictitious field winding voltage
S_n	Rated apparent power
U_{ph}	Line to neutral (phase) voltage
$\underline{U}^{sr}$	Column vector of voltages
$\underline{I}^{sr}$	Column vector of currents
$\underline{\Psi}^{sr}$	Column vector of flux linkages
$\boldsymbol{R}^{sr}$	Resistance matrix
$\boldsymbol{L}^{sr}$	Inductance matrix
$\boldsymbol{T}^{r}$	Transformation matrix
T_e, τ_e	Electrical motor torque
T_L, τ_L	Mechanical load torque
J	Moment of inertia
T_m	Mechanical time constant
p	Number of polepairs in motor
θ	Electrical angle referred to a-axis
θ_{mech}	Mechanical angle
N, n	Motor speed
Ω_{mech}	Motor angular speed
ω	Electrical angular frequency
ω_n	Nominal electric angular frequency
f_n	Nominal electric frequency

Symbol	Explanation
x_d, x_q	d- and q-axis reactance
$x_{s\sigma}$	Leakage reactance
ψ_m	Magnet flux linkage
T_d	d-axis time constant
T_q	q-axis time constant
T_σ	Leakage time constant
T_{samp}	Discrete system sampling time
T_{clk}	FPGA clock speed
λ	System eigenvalue
ζ	Relative damping factor

Superscript	Explanation
s	Quantity referred to stator
r	Quantity referred to rotor
T	Transposed matrix/vector
$\hat{}$	Peak value
$\dot{}$	Time derivative

Subscript	Explanation
s	Stator quantity
r	Rotor quantity
a, b, c	Phases in machine
α, β	Two-phase stator reference frame axes
d, q	Rotating reference frame axes, dq subspace
$z1, z2$	z1z2 subspace axes
f	Field winding
n	Nominal value

Chapter 1

Introduction

1.1 Background and Motivation

A modern variable speed AC motor drive consists of two main components: the controller stage and the power stage. The power stage consists of the motor which is fed by a voltage source converter, whereas the controller stage includes a digital controller and a PWM modulator, which generates the switching gate signals for the converter. In the development process of an electric motor drive, the controller stage is thoroughly tested in a range of different operating scenarios to ensure that it performs optimally. Traditionally, these tests have required the use of a mechanical test facility with motor-generator sets, converter, sensors and switchgear to test the performance of the drive control system. Such a test facility can be costly to build and maintain[1]. The motor and converter may also be under development in parallel with the control system and are therefore not available for use in the control system tests.

In recent years, improvements in FPGA technology have made the use of real-time hardware emulators a viable alternative to the mechanical test beds. By running a simulation of the motor drive power stage on an FPGA, the drive controller can be tested without the need for any test setup. This makes the development of the drive control system cheaper and more efficient.

A common control platform for motor drive systems is under development at the Department of Electric Power Engineering at NTNU, called the NTNU Control Platform. The platform hardware is made up of a control board from Avnet and a process interface board developed by SINTEF. The control platform software consists of a drive routine which runs on a CPU and is written in C++ and an FPGA, which is programmed using Vivado. Vivado is a programming tool developed by Xilinx which lets the user develop IP cores for the FPGA without having to learn hardware description languages, such as VHDL. This control platform is intended to serve as a foundation for future master and PhD projects in the field of electric motor drives[2].

1.2 Scope of Work

The aim of this thesis is to develop and test IP cores for use in a real-time emulator of a permanent magnet synchronous motor. The IP cores are developed using Xilinx System Generator for DSP and tested in Simulink. These IP cores will in further work be implemented as part of a complete real-time emulator of a PMSM drive system, running on the NTNU Control Platform.

IP cores for simulation of both three-phase and six-phase PM synchronous motors as well as mechanical load are designed. The IP cores are tested by implementing them in previously developed Simulink models for three-phase and six-phase motor drive systems, replacing the continuous-time Simulink motor and mechanical load blocks. The outputs from the models containing the discrete-time IP cores are compared to the original models, which are used as reference.

The motor torque and speed outputs from the models using the System Generator IP cores are compared to the outputs from the reference models to test the accuracy and performance of the IP cores.

1.3 Thesis Outline

The thesis report is divided into 7 main chapters:

- **Chapter 2 - Theoretical Background**
  This chapter gives an overview of the theory on FPGAs, modelling of three-phase and six-phase PM motors, numerical methods of discretization and system stability analysis.

- **Chapter 3 - Programming Structure**
  In this chapter, the NTNU Control Platform structure is discussed, as well as the procedure for design of IP cores using System Generator for DSP.

- **Chapter 4 - Development of Emulator IP Cores**
  The equations used for the modelling of the emulator IP cores are shown, as well as the method for implementing these equations in the IP cores.

- **Chapter 5 - Simulation Results**
  The IP cores which emulate the PM motors are implemented in three-phase and six-phase PM motor drive system models. The performance and output of the discrete IP cores are compared to the output of the reference Simulink models.

- **Chapter 6 - Discussion**
  The results from chapter 5 are discussed, and an eigenvalue analysis of the three-phase and six-phase motor is conducted.

- **Chapter 7 - Conclusion**
  The main findings and conclusion for the thesis report is presented.

- **Chapter 8 - Further Work**
  Propositions are made for further work which can be conducted.

Chapter 2

Theoretical Background

2.1 Field Programmable Gate Array

An FPGA (Field Programmable Gate Array) is an integrated circuit that consists of an array of logic blocks that are connected via programmable interconnects. These logic blocks are made up of two basic components: flip-flops and lookup tables. By interconnecting these logic blocks, the FPGA can be programmed to perform different tasks. The FPGA logic is programmed using a hardware description language such as VHDL or Verilog. The FPGA can therefore be considered as a reconfigurable circuit[3]. The fact that the hardware is programmable makes it highly flexible and separates the FPGA from ASICs (Application Specific Integrated Circuits), which have a fixed circuit layout and are customized for one specific use. FPGAs have taken over many of the tasks that were previously only handled by ASIC (Application-Specific Integrated Circuits). They can be re-programmed and are thus more flexible than ASICs, which have a fixed circuit layout and are costly to develop.

FPGAs can run multiple operations in parallel on a single clock pulse. This separates them from microprocessors or CPUs, which executes a program based on instructions stored in memory, and executes their code sequentially. FPGAs can therefore process data at higher rates and are ideal for tasks which require fast computing and can benefit from performing several operations at once.

Most of the logic in the FPGA is synchronous, meaning that the logic must be encapsulated between registers, which store values for the next enable pulse. The logic between the registers has a certain time delay or latency depending on their complexity. This is known as the propagation delay, or the delay for a signal to travel between registers. To maintain synchronicity in the operations, all inputs to registers must be stable before the next enable pulse. There are two clocks to consider in the design of FPGA hardware code: the FPGA clock period, which is the rate at which the programmable logic operates, and the sample time, which is the period of the pulse signal which enable the registers in the IP cores. If the latency in one of the parallel paths between registers is too high, the calculation will not reach the output register in time. The rate at which the circuit can run is therefore limited by the parallel path with the highest latency[4].

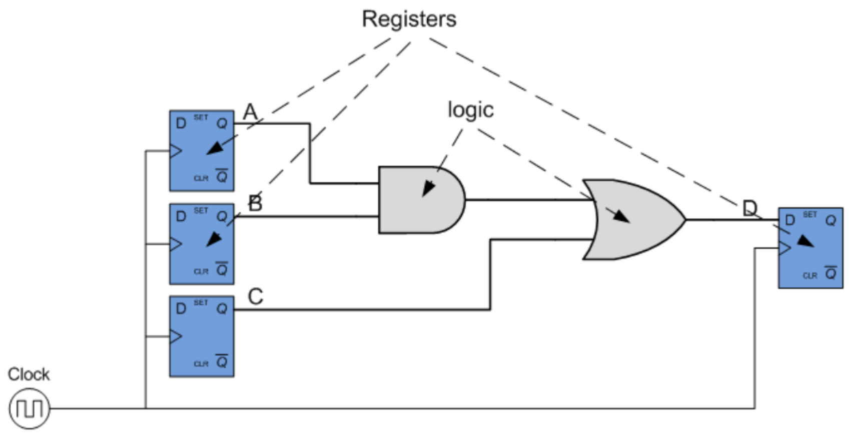


Figure 2.1: Synchronous digital circuit[5]

2.2 Permanent Magnet Synchronous Motor

When selecting an AC motor for a motor drive application, the choice stands between two main types: the asynchronous motor and the synchronous motor. The asynchronous motor is widely used and is known as the workhorse of the industry. It is relatively cheap and reliable. However, it has some disadvantages compared to the synchronous motor; reactive power for magnetization must be supplied from the drive and the power loss is higher due to rotor slip.

Synchronous motors in the kW-MW range come in two main types: wire wound separately magnetized motors, which require an external power source for supply of the DC magnetizing current, and permanent magnet synchronous motors, where magnets in the rotor create the magnetic field. Permanent magnet synchronous motors are becoming an increasingly popular choice for variable speed applications due to their high power to weight ratio, efficiency and reliability[6].

PM synchronous motors can be classified into two categories, depending on how the magnets are mounted on the rotor: surface mounted (PMSM) or internal (IPMSM). In the PMSM, the magnets are mounted on the rotor surface, while in the IPMSM the magnets are embedded in the rotor. The air gap between the rotor and stator in an IPMSM is therefore smaller than for a PMSM. The relative permeability of the permanent magnets are close to the relative permeability of air, therefore the *effective air gap* counts both the actual air gap and the space that the permanent magnets occupy in the rotor. A larger effective air gap gives a smaller inductance. The PMSM therefore has a similar inductance in both the d-axis and q-axis, whereas the IPMSM has a large q-axis inductance and a smaller d-axis inductance due to the low permeability of the magnets.

To achieve field weakening in a PM motor, a negative d-axis current must be applied to counteract the field created by the magnets. For the IPMSM this negative current also contributes to the motor torque, which is not the case for the PMSM. Therefore the IPMSM has a wider field weakening speed range and is the preferred choice for applications where a large field-weakening region is required, such as in electric vehicles[6].

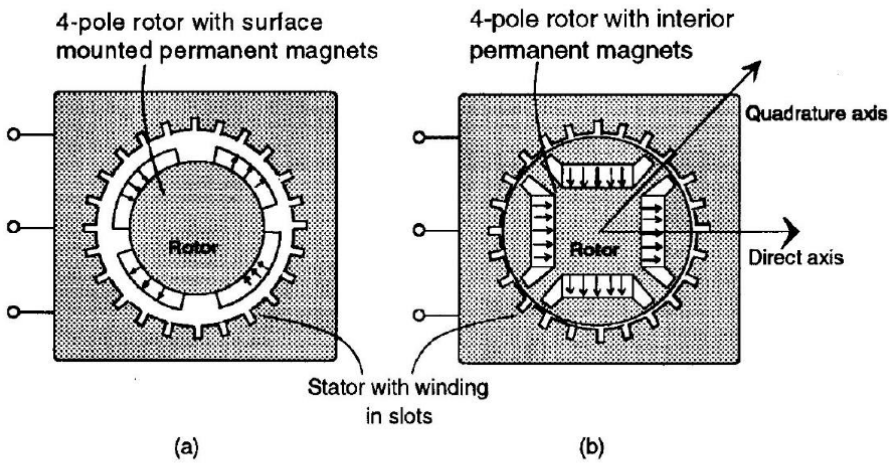

4-pole rotor with surface mounted permanent magnets
4-pole rotor with interior permanent magnets
Quadrature axis
Rotor
Rotor
Direct axis
Stator with winding in slots
(a)
(b)

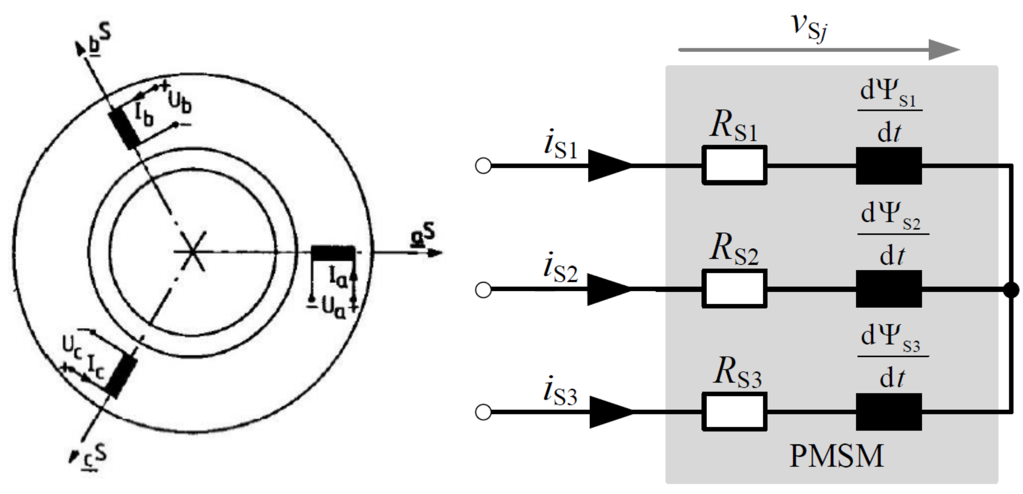
Figure 2.2: Cross-section of surface mounted PM motor versus internal PM motor[7]

2.3 Modelling of Three-Phase PMSM

This section gives an overview of the modelling method for a three-phase PMSM synchronous motor, as presented in [6]. The following simplifications are made in the modelling of the motor:

- Stator windings are modelled as concentrated windings and the magnetic flux in the machine is sinusoidal.
- Magnetic saturation and core losses are neglected.
- Eddy current losses are neglected.
- Resistances and inductances are assumed to be identical for all phases, and independent of changes in temperature and frequency.
- Field from permanent magnets is constant and independent of changes in temperature and frequency.

The stator layout and equivalent circuit of a three-phase motor is shown in figure 2.3.



(a) Three-phase motor with 120° separation between concentrated stator windings[6].

(b) Equivalent circuit of three-phase synchronous PM motor[8]. v_{sj} is the phase voltage for phases $j \in \{1, 2, 3\}$.

Figure 2.3: Modelling of three-phase PM motor.

By applying the voltage balance equation for each winding in the motor and representing the field created by the permanent magnets in the rotor as a field winding fed by a constant current source, the following voltage balance equations are given in matrix form as:

$$\underline{U}^{sr} = \boldsymbol{R}^{sr} \cdot \underline{I}^{sr} + \frac{d\underline{\Psi}^{sr}}{dt} \tag{2.1}$$

Where:

$$\underline{I}^{sr} = [I_a \; I_b \; I_c \; I_f]^T \quad , \quad \underline{U}^{sr} = [U_a \; U_b \; U_c \; U_f]^T \quad , \quad \underline{\Psi}^{sr} = [\Psi_a \; \Psi_b \; \Psi_c \; \Psi_f]^T \tag{2.2}$$

The flux linkages can be expressed as the product of the stator currents and the rotor position dependent inductance matrix, containing the self-inductances and mutual inductances between stator and rotor windings:

$$\underline{\Psi}^{sr} = \boldsymbol{L}^{sr}(\theta) \cdot \underline{I}^{sr} \tag{2.3}$$

The resistance and inductance matrices are given as:

$$\boldsymbol{R}^{sr} = diag[R_s \; R_s \; R_s \; R_f] \quad , \quad \boldsymbol{L}^{sr}(\theta) = \begin{bmatrix} \boldsymbol{L}^s_{ss} & \boldsymbol{L}^s_{sr}(\boldsymbol{\theta}) \\ \boldsymbol{L}^s_{rs}(\boldsymbol{\theta}) & \boldsymbol{L}^s_{rr} \end{bmatrix} \tag{2.4}$$

The three-phase system model described in equation 2.1, with its position-dependent inductance matrix and three independent variables is quite complex to simulate. It is therefore desirable to develop a simplified model where the three-phase machine is represented as a two-phase machine and analyzed in a reference frame in which the inductances are independent of the rotor position.

2.3.1 Three-Phase to Two-Phase Transformation

A balanced three-phase machine can be represented as a two-phase machine, thereby reducing the number of state variables in the model from three to two. This is also known as $\alpha\beta$- or Clarke-transformation.

Transformation matrix

The transformation of three-phase stator currents by use of the Clarke transformation matrix is shown in equation 2.5. The resulting two-phase currents are scaled to be equal in magnitude to the original three-phase currents.

$$\begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} 1 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix} \cdot \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \tag{2.5}$$

The resulting two-phase system is orthogonal with the alpha-component aligned with the a-component and the beta-component leading by 90°.

$$I_\alpha = \hat{I}_s \cos(\omega t), \quad I_\beta = \hat{I}_s \sin(\omega t) \tag{2.6}$$

Transformation of system model

Applying the transformation to equation 2.1 yields the following voltage expressions for the two-phase machine, where superscript s or r indicates if the winding is fixed to the stator or rotor reference frame:

$$\begin{aligned} U^s_{s\alpha} &= R_s \cdot I^s_{s\alpha} + \frac{d\Psi^s_{s\alpha}}{dt} \\ U^s_{s\beta} &= R_s \cdot I^s_{s\beta} + \frac{d\Psi^s_{s\beta}}{dt} \end{aligned} \tag{2.7}$$

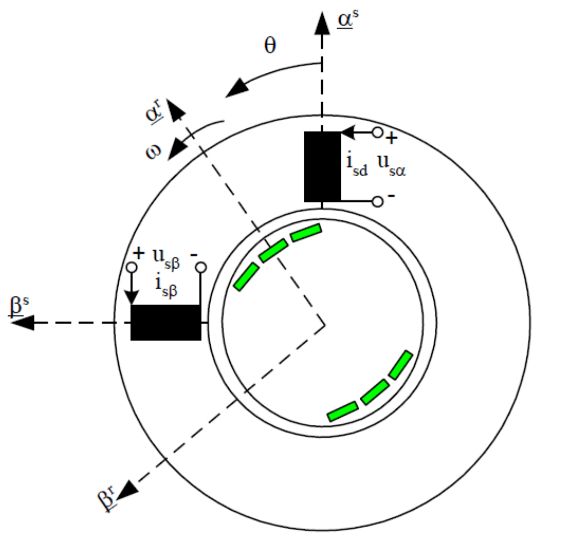
Since there is no winding along the rotor q-axis, the flux linkages can be expressed as:

$$\begin{aligned} \Psi^s_{s\alpha} &= L_{s\alpha s\alpha}(\theta) \cdot I^s_{s\alpha} + L_{s\alpha s\beta}(\theta) \cdot I^s_{s\beta} + \Psi_{\alpha f}(\theta) \\ \Psi^s_{s\beta} &= L_{s\beta s\alpha}(\theta) \cdot I^s_{s\alpha} + L_{s\beta s\beta}(\theta) \cdot I^s_{s\beta} + \Psi_{\beta f}(\theta) \end{aligned} \tag{2.8}$$

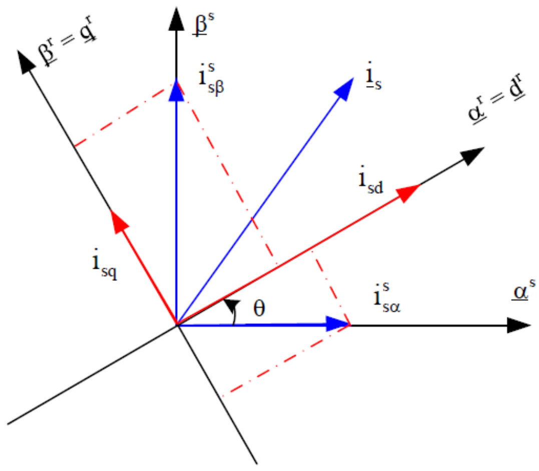
The procedure for calculating the two-phase machine inductances from the actual three-phase machine inductance parameters is outlined in [6].

2.3.2 Rotating Reference Frame Transformation

Next, the model is transformed into a model where the system quantities are DC at steady state, and where the inductance matrix is constant. This is done by replacing the stator windings of the two-phase machine with fictitious windings, which rotates at the same speed as the rotor. This is also known as Park or dq transformation.



(a) Three-phase IPMSM represented as a two-phase machine[6].



(b) Relation between two-phase stationary reference frame and synchronous rotating reference frame[6].

Figure 2.4: Two-phase and rotating reference transformations of three-phase machine.

Transformation matrix

The transformation between is achieved by multiplying the two-phase stator reference frame parameters with the Park transformation matrix, as shown below for the stator currents:

$$\underline{I_s}^r = \boldsymbol{T}_{ss}^r \cdot \underline{I_s}^s \quad \Rightarrow \quad \begin{bmatrix} I_{sd}^r \\ I_{sq}^r \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} I_{s\alpha}^s \\ I_{s\beta}^s \end{bmatrix} \tag{2.9}$$

Where angle θ is the electrical angle of the motor: $\theta = p \cdot \theta_{mech}$.

Transformation of system model

Transformation of the voltage equations in 2.7 to the rotor reference frame is achieved by multiplication with the transformation matrix:

$$\underline{U}^r = \boldsymbol{T}^r \cdot \underline{U}^{sr} \quad , \quad \boldsymbol{T}^r = \begin{bmatrix} \boldsymbol{T}_{ss}^r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{2.10}$$

Which gives the following expressions for the d-axis and q-axis components of the stator voltage, as shown in [6]:

$$\begin{aligned} U_{sd} &= R_s \cdot I_{sd} + \frac{d\Psi_{sd}}{dt} - \omega \cdot \Psi_{sq} \\ U_{sq} &= R_s \cdot I_{sq} + \frac{d\Psi_{sq}}{dt} + \omega \cdot \Psi_{sd} \end{aligned} \tag{2.11}$$

The electric motor torque is calculated from the stator currents and flux linkages as:

$$T_e = \frac{3}{2} p \left(\Psi_{sd} \cdot I_{sq} - \Psi_{sq} \cdot I_{sd} \right) \tag{2.12}$$

Given the electrical motor torque, the motor speed is calculated by using Newton's second law for rotation:

$$\frac{d\Omega}{dt} = \frac{T_e - T_L}{J} \tag{2.13}$$

Where T_L is the torque of the mechanical load the motor is moving and J is the inertia of the rotating system.

2.3.3 Per Unit Scaling

After transformation of the motor model to the rotating reference frame, the model is scaled using the per unit (pu) system. There are several advantages of using the pu system. Since the motor is scaled using the rated motor parameters as base values, it becomes easier to detect if any parameters exceed the rated values of the motor. In addition, simpler model can be achieved by correct selection of base values. The base values for the motor are chosen as:

$$I_{s,base} = \hat{I}_n \quad , \quad U_{s,base} = \hat{U}_n \quad , \quad \Psi_{s,base} = \frac{\hat{U}_{ph,n}}{\omega_n} = \frac{\hat{U}_{ph,n}}{2\pi \cdot f_n} \tag{2.14}$$

The rated apparent power of the motor is used as base power while the torque base is calculated by dividing the base power with the rated motor speed in rad/s.

$$S_{base} = S_n = \frac{3}{2} \cdot \hat{U}_{s,base} \cdot \hat{I}_{s,base} = \sqrt{3} U_{LL,n} \cdot I_{s,n} \quad , \quad T_{base} = \frac{S_n}{\Omega_n} \tag{2.15}$$

As there is no rotor current in the PMSM motor, the s-subscript indicating a stator parameter is redundant and will therefore not be used in the further expressions. Applying the pu-transformation to equation 2.11 yields[6]:

$$\begin{aligned} u_d &= r_s i_d + \frac{1}{\omega_n} \frac{d\psi_d}{dt} - n\psi_q \\ u_q &= r_s i_q + \frac{1}{\omega_n} \frac{d\psi_q}{dt} + n\psi_d \end{aligned} \tag{2.16}$$

Inserting $\psi_d = x_d i_d + \psi_m$ and $\psi_q = x_q i_q$, the pu model can be expressed with currents as state variables:

$$\begin{aligned} u_d &= r_s i_d + \frac{x_d}{\omega_n} \frac{di_d}{dt} - n x_q i_q \\ u_q &= r_s i_q + \frac{x_q}{\omega_n} \frac{di_q}{dt} + n x_d i_d + n\psi_m \end{aligned} \tag{2.17}$$

The electric motor torque in pu is calculated as:

$$\tau_e = \psi_d \cdot i_q - \psi_q \cdot i_d \tag{2.18}$$

The mechanical load equation in pu with mechanical time constant T_m is written as:

$$\frac{dn}{dt} = \frac{\tau_e - \tau_L}{T_m} \quad , \quad T_m = \frac{J\Omega_n^2}{S_n} \tag{2.19}$$

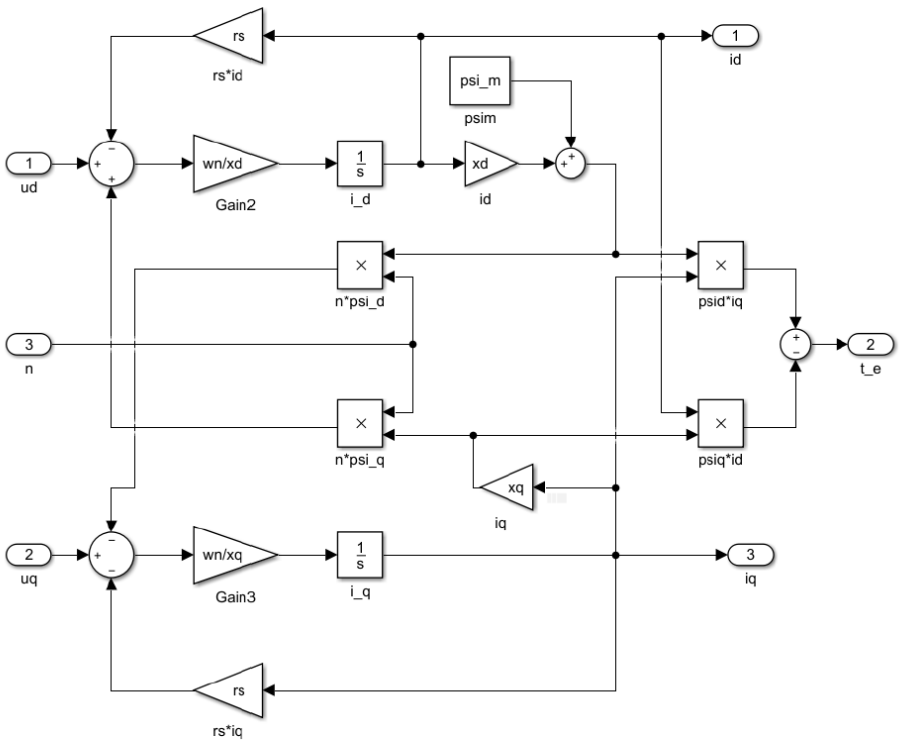


Figure 2.5: Simulink block diagram of dq-frame per unit 3-phase motor[6].

2.4 Modelling of Six-Phase PMSM

There are two main stator winding configurations for a six-phase machine: split-phase asymmetrical configuration or symmetrical configuration. In this thesis, an asymmetrical winding configuration is used, where two three-phase winding sets are displaced by an angle of 30° from each other. In general, for machines with n total windings, it is found that the ideal angular spacing between winding sets is π/n for an even number of sets and $2\pi/n$ for an odd number of sets. In addition, the neutral points of the winding sets are isolated from each other. This configuration has been shown to eliminate 6-th harmonic pulsations in the motor torque[9].

This section gives an overview of the modelling of a six-phase PMSM synchronous motor with the configuration described above, as presented in [10]. The simplifications listed for the modelling of the three-phase PMSM also apply for the six-phase PMSM.

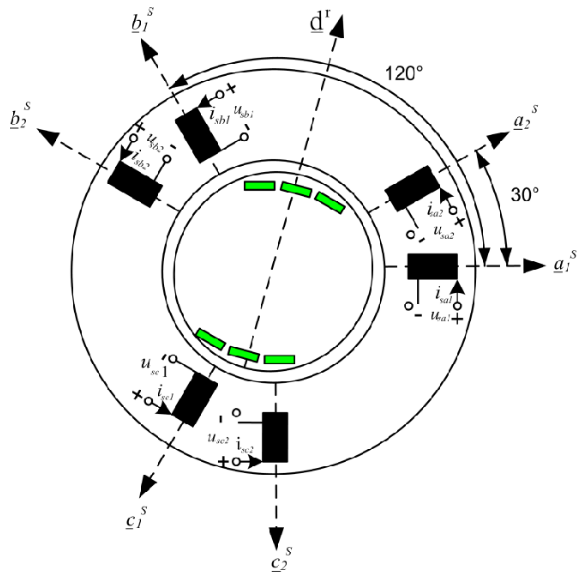


Figure 2.6: Layout of six-phase motor with 30° phase shift between stator winding sets[11].

The voltage balance equation for the six-phase motor is developed the same way as for three-phase, only with six stator components and one rotor component.

$$\underline{U}^{sr} = \boldsymbol{R}^{sr} \cdot \underline{I}^{sr} + \frac{d\underline{\Psi}^{sr}}{dt} \tag{2.20}$$

Where:

$$\begin{aligned} \underline{I}^{sr} &= [I_{a1}\ I_{a2}\ I_{b1}\ I_{b2}\ I_{c1}\ I_{c2}\ I_f]^T \\ \underline{U}^{sr} &= [U_{a1}\ U_{a2}\ U_{b1}\ U_{b2}\ U_{c1}\ U_{c2}\ U_f]^T \\ \underline{\Psi}^{sr} &= [\Psi_{a1}\ \Psi_{a2}\ \Psi_{b1}\ \Psi_{b2}\ \Psi_{c1}\ \Psi_{c2}\ \Psi_f]^T \end{aligned} \tag{2.21}$$

The variables of the second three-phase set lags the first set by 30°. As in the three-phase model, the flux linkages are still expressed by the currents and system inductances as:

$$\underline{\Psi}^{sr} = \boldsymbol{L}^{sr}(\theta) \cdot \underline{I}^{sr} \tag{2.22}$$

The resistance and inductance matrices are given as:

$$\boldsymbol{R}^{sr} = diag[R_s\ R_s\ R_s\ R_s\ R_s\ R_s\ R_f] \quad , \quad \boldsymbol{L}^{sr}(\theta) = \begin{bmatrix} \boldsymbol{L}_{ss}^s & \boldsymbol{L}_{sr}^s(\boldsymbol{\theta}) \\ \boldsymbol{L}_{rs}^s(\boldsymbol{\theta}) & \boldsymbol{L}_{rr}^s \end{bmatrix} \tag{2.23}$$

2.4.1 Rotating Vector Space Decomposition

Like for the three-phase motor model, it is desirable to transform the six-phase model equation into one with fewer state variables and where inductances are independent of rotor position. In this thesis, the model is transformed by a method called *Rotating Vector Space Decomposition*, as outlined in [10]. The six-phase variables are mapped onto two two-dimensional subspaces and a zero sequence subspace. For the motor configuration used in this thesis, the zero-sequence subspace will not be excited and can therefore be exempt from modelling. Additionally, Park transformation is applied on the system, which transforms the AC variables in the first subspace into DC quantities, similar to the dq-transformation performed on the three-phase motor.

The two subspaces are here referred to as the dq subspace and the z1z2 subspace. The dq subspace model all the torque-producing harmonics, that is, the fundamental and the k^{th} order harmonics ($k = 12m + 1, \;\; m = 1, 2, 3...$). The other non-torque producing harmonics are modelled in the z1z2 subspace[12].

Using the rotating vector space decomposition method as presented in [10], the amplitude-invariant VSD transformation matrix for the six-phase system is:

$$\underline{U}^r = \boldsymbol{T}^r \cdot \underline{U}^{sr} \quad , \quad \boldsymbol{T}^r = \begin{bmatrix} \boldsymbol{T}^r_{ss} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}^r_{rr} \end{bmatrix} \tag{2.24}$$

Where:

$$\boldsymbol{T}^r_{ss} = \frac{1}{3} \cdot \left[\begin{matrix} \cos\theta & \cos\left(\theta - \frac{\pi}{6}\right) & \cos\left(\theta - \frac{2\pi}{3}\right) \\ -\sin\theta & -\sin\left(\theta - \frac{\pi}{6}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) \\ \cos\theta & \cos\left(\theta - \frac{7\pi}{6}\right) & \cos\left(\theta - \frac{2\pi}{3}\right) \\ -\sin(\theta - \pi) & -\sin\left(\theta - \frac{\pi}{6}\right) & -\sin\left(\theta - \frac{5\pi}{3}\right) \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix} \right. \cdots$$

$$\cdots \left. \begin{matrix} \cos\left(\theta - \frac{5\pi}{6}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) & \cos\left(\theta - \frac{3\pi}{2}\right) \\ -\sin\left(\theta - \frac{5\pi}{6}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) & -\sin\left(\theta - \frac{3\pi}{2}\right) \\ \cos\left(\theta - \frac{11\pi}{6}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) & \cos\left(\theta - \frac{\pi}{2}\right) \\ -\sin\left(\theta - \frac{5\pi}{6}\right) & -\sin\left(\theta - \frac{\pi}{3}\right) & -\sin\left(\theta - \frac{3\pi}{2}\right) \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} \right] \tag{2.25}$$

and:

$$\boldsymbol{T}^r_{rr} = \boldsymbol{I} \tag{2.26}$$

2.4.2 Per Unit Scaling

The base stator current, voltage and flux linkage for the pu scaling of the six-phase motor are chosen as the rated peak phase values, same as for the three-phase model.

$$I_{s,base} = \hat{I}_n \quad , \quad U_{s,base} = \hat{U}_n \quad , \quad \Psi_{s,base} = \frac{\hat{U}_{ph,n}}{\omega_n} = \frac{\hat{U}_{ph,n}}{2\pi \cdot f_n} \tag{2.27}$$

The rated apparent power of the motor is used as base power, and the base torque is calculated by dividing the base power with the rated motor speed in rad/s.

$$S_{base} = S_n = 3 \cdot \hat{U}_{s,base} \cdot \hat{I}_{s,base} = \sqrt{3} U_{LL,n} \cdot I_{s,n} \quad , \quad T_{base} = \frac{S_n}{\Omega_n} \tag{2.28}$$

After scaling of the six-phase motor model, the voltage expressions for the dq and z1z2 subspaces are expressed as shown in equation 2.29. The pu scaling procedure is explained in detail in [10].

$$\begin{gathered} u_d = r_s \cdot i_d + \frac{x_d}{\omega_n}\frac{di_d}{dt} - n x_q i_q \quad , \quad u_q = r_s \cdot i_q + \frac{x_q}{\omega_n}\frac{di_q}{dt} + n x_d i_d + n\psi_m \\ u_{z1} = r_s \cdot i_{z1} + \frac{x_{s\sigma}}{\omega_n}\frac{di_{z1}}{dt} + n x_{s\sigma} i_{z2} \quad , \quad u_{z2} = r_s \cdot i_{z2} + \frac{x_{s\sigma}}{\omega_n}\frac{di_{z2}}{dt} - n x_{s\sigma} i_{z1} \end{gathered} \tag{2.29}$$

The system equations for the dq subspace are identical to the three-phase system model. This is also the case for the electric torque equation, since torque production occurs only in this subspace:

$$\tau_e = \psi_d \cdot i_q - \psi_q \cdot i_d \tag{2.30}$$

$$\frac{dn}{dt} = \frac{\tau_e - \tau_L}{T_m} \quad , \quad T_m = \frac{J\Omega_n^2}{S_n} \tag{2.31}$$

2.5 Numerical Methods

The dynamic behaviour of a system can be described as a set of first order ordinary differential equations(ODEs), which represents the states of the system at a given point in time. Simulating such a system requires the use of numerical methods for solving the equations, since an exact analytic solution is harder to calculate and often cannot be found at all. These numerical methods work by computing the approximate solution of the equations at successive time steps, given the initial values of the system states.

2.5.1 Forward and Backward Euler Methods

One of the simplest numerical methods is the forward Euler method, which approximates the solution at the next time step by taking a small step along the tangent line of the function.

$$y_{n+1} = y(t_n + T_{samp}) \approx y(t_n) + T_{samp} \cdot y'(t_n) \tag{2.32}$$

Where the step size or sample time $T_{samp} = t_n - t_{n-1}$. The Euler method can be derived by considering the Taylor series expansion of the function $y(t)$ around t_n[13]:

$$y(t_n + h) \equiv y_{n+1} = y(t_n) + \frac{T_{samp} \cdot y'(t_n)}{1!} + \frac{T_{samp}^2 \cdot y''(t_n)}{2!} + ... \tag{2.33}$$

By only evaluating the first order derivative and recognizing that $y'(t_n) = f(y_n, t_n)$, the forward Euler method is given as:

$$y_{n+1} = y_n + T_{samp} \cdot f(y_n, t_n) + O(T_{samp}^2) \tag{2.34}$$

Due to the truncation of the Taylor series, an error is introduced to the solution for every time step. This is known as the local truncation error (LTE). The exact error for each step is not known if the analytical solution of the equation is unknown, but it is evident from the Taylor series that the LTE is approximately proportional to $(T_{samp})^2$ for small values of T_{samp}. The total accumulated (global) error scales with nT_{samp} for a fixed time step, and since the total number of steps taken is proportional to $(T_{samp})^{-1}$, it follows that the global error is approximately proportional to T_{samp}[14]. In general, a numerical method is classified as k-th order if the LTE scales with $(T_{samp})^{k+1}$ and the total accumulated error scales with $(T_{samp})^k$[13]. Higher order methods therefore converge faster to the exact solution when sample time is reduced than a first order method.

The forward Euler method is an *explicit* method, meaning that it calculates the next state of the system based on the current state of the system. Implicit methods can also be used, in which y_{n+1} is given as the solution to an algebraic equation[14]. The implicit counterpart to the forward Euler method is the backward Euler method:

$$y_{n+1} = y_n + T_{samp} \cdot f(y_{n+1}, t_{n+1}) + O(T_{samp}^2) \tag{2.35}$$

Since $f(y_{n+1}, t_{n+1})$ is not known, this equation is implicit, and must be solved by some root-finding algorithm like Newton-Rhapson. It is therefore more computationally demanding than the explicit method. The advantage of using implicit methods is that they are inherently stable, whereas the implicit methods may become numerically unstable for some problems unless the time step is very short. In real-time applications, explicit methods are preferred over implicit methods, since no time-consuming iterations are required to solve the expression.

2.5.2 Numerical Stability

As previously mentioned, explicit methods may become numerically unstable for some problems if the time step is too high. This means that the numerical solution diverges from the exact solution and becomes wildly inaccurate. This can be further examined by considering the first order ordinary differential equation[13]:

$$\frac{dy}{dt} = -\lambda y \quad , \quad y(0) = 1 \tag{2.36}$$

Where $\lambda > 0$. The exact solution for this differential equation is:

$$y(t) = e^{-\lambda t} \tag{2.37}$$

The value of y starts at 1 at $t = 0$, then exponentially approaces 0 as time increases, with a time constant of $1/\lambda$. Discretizing the equation using the forward Euler method yields:

$$\frac{1}{T_{samp}}(y_{n+1} - y_n) = -\lambda y_n \quad \Rightarrow \quad y_{n+1} = y_n(1 - \lambda T_{samp}) \tag{2.38}$$

Which can be expressed with the initial value as:

$$y_{n+1} = y_n(1 - \lambda T_{samp}) = y_{n-1}(1 - \lambda T_{samp})^2 = \cdots = y_0(1 - \lambda T_{samp})^{n+1} \tag{2.39}$$

In order to prevent amplification of errors in the iteration process which will lead to instability, the following criterion is given, which gives an upper bound for the sample time for the forward Euler method[13]:

$$|1 - \lambda T_{samp}| < 1 \quad \Rightarrow \quad 0 < \lambda T_{samp} < 2 \tag{2.40}$$

For complex values of λ, the forward Euler method will be numerically stable if the product λT_{samp} falls within a circle with radius 1 on the complex plane with center in (-1,0), as shown in figure 2.7.

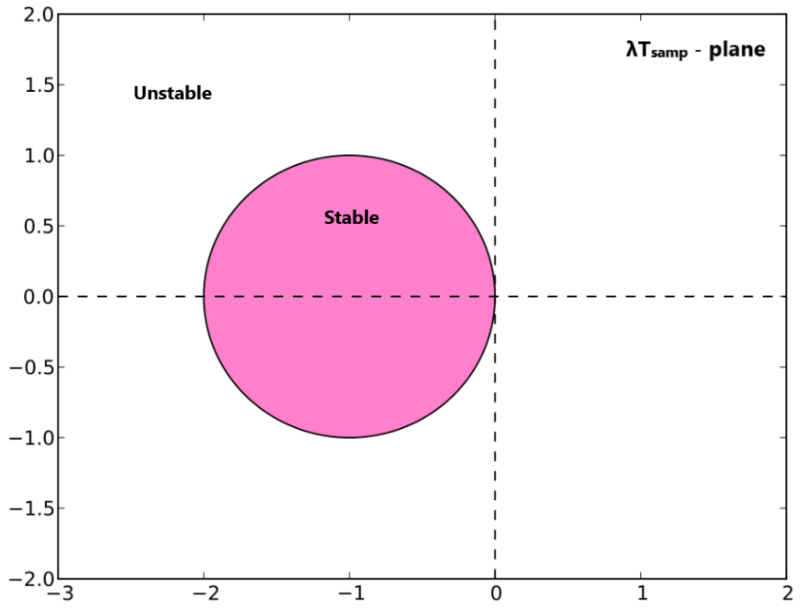


Figure 2.7: Stability region for forward Euler method. Figure adapted from [15].

2.6 System Stability Analysis

2.6.1 Continuous System

A first-order linear time-invariant (LTI) system consisting of inputs, outputs and state variables can be represented as shown in equation 2.41.

$$\begin{aligned}\dot{x}(t) &= \boldsymbol{A} \cdot x(t) + \boldsymbol{B} \cdot u(t) \\ y(t) &= \boldsymbol{C} \cdot x(t) + \boldsymbol{D} \cdot u(t)\end{aligned} \tag{2.41}$$

Where:

$x(t)$	:	*Vector of state variables*
$\dot{x}(t)$	:	*Vector of state variable derivatives*
$u(t)$	:	*Vector of inputs*
$\boldsymbol{A}$	:	*System matrix*
$\boldsymbol{B}$	:	*Input matrix*
$\boldsymbol{C}$	:	*Output matrix*
$\boldsymbol{D}$	:	*Feed-forward matrix*

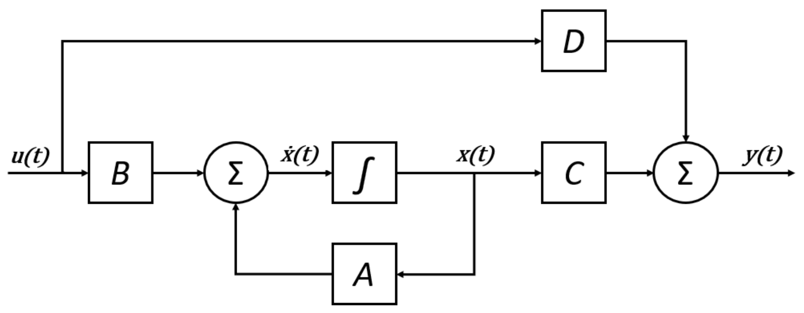


Figure 2.8: Block diagram representation of first order state-space system.

An LTI system can be classified by three different degrees of stability, depending on its behaviour when an impulse signal is applied as input:

- **Asymptotically stable**
  The system returns asymptotically to its previous state after impulse is applied.

- **Marginally stable**
  The system does not return to its previous state, but the output does not go to infinity. A permanent offset or standing oscillations with bounded amplitude appear on the output.

- **Unstable**
  The system output "blows up", i.e., it diverges to infinity.

The solution for the differential state equation is[16]:

$$x(t) = e^{A(t-t_0)} \cdot x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)} \cdot \boldsymbol{B} \cdot u(t) d\tau \tag{2.42}$$

The stability of the system is dependent upon its natural response, i.e., the system response to a bounded initial condition with no external inputs, which from 2.42 is given as:

$$x(t) = e^{At} \cdot x(t_0) \tag{2.43}$$

If the exponent has a negative sign, it will decay to zero as time increases. If it is positive however, it will go to infinity. The stability of such a system can thus be studied by evaluating the eigenvalues of system matrix A. The system eigenvalues are all values of λ that satisfy the following characteristic equation:

$$det(\lambda I - \boldsymbol{A}) = 0 \tag{2.44}$$

Where I is the identity matrix, equal in dimensions to system matrix $\boldsymbol{A}$. The stability of a continuous-time LTI system is determined by the sign of the real part of the eigenvalues of the system. If all eigenvalues lie in the left half of the complex s-plane, i.e., the real parts are negative, the system is asymptotically stable. Marginal stability occurs when the real part is zero, i.e., the eigenvalue is purely imaginary. If any eigenvalues lie in the right half plane, the system is unstable. The imaginary part of the eigenvalue determines the damped oscillation frequency of the mode in [rad/s][17].

$$\lambda = \alpha \pm j\beta \tag{2.45}$$

The value of the relative damping ratio of the eigenvalues determines how well the system is damped. It is given as:

$$\zeta = \frac{-\alpha}{\sqrt{\alpha^2 + \beta^2}} \tag{2.46}$$

If $\zeta > 0$, the system is stable. If $\zeta = 0$, the system is marginally stable. If $\zeta < 0$, the system is unstable.

Eigenvalues for Continuous Three-Phase Motor Model

The system equations for the transformed motor models described earlier in this chapter can be analysed in this manner. Rewriting equation 2.17 for the three-phase model in the format of equation 2.41 yields:

$$\begin{bmatrix} \dot{i}_d \\ \dot{i}_q \end{bmatrix} = \begin{bmatrix} \frac{-\omega_n r_s}{x_d} & \frac{\omega_n n x_q}{x_d} \\ \frac{-\omega_n n x_d}{x_q} & \frac{-\omega_n r_s}{x_q} \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} \frac{\omega_n}{x_d} & 0 \\ 0 & \frac{\omega_n}{x_q} \end{bmatrix} \cdot \begin{bmatrix} u_d \\ u_q \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{-\omega_n n \psi_m}{x_q} \end{bmatrix} \tag{2.47}$$

The characteristic equation for the three-phase system becomes:

$$det\left(\begin{bmatrix} \lambda + \frac{\omega_n r_s}{x_d} & \frac{-\omega_n n x_q}{x_d} \\ \frac{\omega_n n x_d}{x_q} & \lambda + \frac{\omega_n r_s}{x_q} \end{bmatrix}\right) = 0 \tag{2.48}$$

Defining time constants $T_d = \frac{x_d}{\omega_n r_s}$, $T_q = \frac{x_q}{\omega_n r_s}$ and writing out the determinant expression yields the quadratic equation:

$$\lambda^2 + \lambda\left(\frac{1}{T_d} + \frac{1}{T_q}\right) + \frac{1}{T_d T_q} + (\omega_n n)^2 = 0 \tag{2.49}$$

Applying the quadratic formula, the eigenvalue pair for this system is found as:

$$\lambda_{1,2} = \frac{-1}{2}\left(\frac{1}{T_d} + \frac{1}{T_q}\right) \pm \sqrt{\left(\frac{1}{2}\left(\frac{1}{T_d} - \frac{1}{T_q}\right)\right)^2 - (\omega_n n)^2} \tag{2.50}$$

From equation 2.50, it can be seen that the continuous three-phase motor model will always be asymptotically stable, since the real part is always negative. When the motor speed is zero, the eigenvalues are purely real. As the motor speed increases, so do the imaginary value of the eigenvalues.

Eigenvalues for Continuous Six-Phase Motor Model

The six-phase model consists of two decoupled subspaces, and the stability of the system can therefore be evaluated by considering each subspace separately.
As shown earlier, the dq subspace for the six-phase motor model is identical to the three-phase model, so the eigenvalues for this subspace are calculated in the same way as for the three-phase model. For the z1z2 subspace, the system equation in the format of equation 2.41 becomes:

$$\begin{bmatrix}\dot{i}_{z1}\\ \dot{i}_{z2}\end{bmatrix}=\begin{bmatrix}\frac{-\omega_n r_s}{x_{s\sigma}} & -\omega_n n\\ \omega_n n & \frac{-\omega_n r_s}{x_{s\sigma}}\end{bmatrix}\cdot\begin{bmatrix}i_{z1}\\ i_{z2}\end{bmatrix}+\begin{bmatrix}\frac{\omega_n}{x_{s\sigma}} & 0\\ 0 & \frac{\omega_n}{x_{s\sigma}}\end{bmatrix}\cdot\begin{bmatrix}u_{z1}\\ u_{z2}\end{bmatrix}\tag{2.51}$$

The characteristic equation then becomes:

$$det\left(\begin{bmatrix}\lambda+\frac{\omega_n r_s}{x_{s\sigma}} & \omega_n n\\ -\omega_n n & \lambda+\frac{\omega_n r_s}{x_{s\sigma}}\end{bmatrix}\right)=0\tag{2.52}$$

Defining the time constant $T_\sigma=\frac{x_{s\sigma}}{\omega_n r_s}$ and writing out the determinant expression yields the quadratic equation:

$$\lambda^2+\frac{2\lambda}{T_\sigma}+\frac{1}{T_\sigma^2}+(\omega_n n)^2=0\tag{2.53}$$

Using the quadratic formula, the eigenvalues for the z1z2 system are found as:

$$\lambda_{1,2}=\frac{-1}{T_\sigma}\pm\sqrt{\frac{1}{T_\sigma^2}-\left(\frac{1}{T_\sigma^2}+\omega_n^2 n^2\right)}=\frac{-1}{T_\sigma}\pm j\omega_n n\tag{2.54}$$

2.6.2 Discrete-time system

Using the forward Euler method of discretization, the continuous system can be approximated as a discrete-time system. Discretizing equation 2.41 using this method turns the differential equation into a difference equation[16]:

$$\begin{aligned}x[k+1]&=x[k]+(\boldsymbol{A}x[k]+\boldsymbol{B}u[k])T_{samp}\\&=(\boldsymbol{I}+\boldsymbol{A}T_{samp})x[k]+\boldsymbol{B}T_{samp}u[k]\end{aligned}\tag{2.55}$$

Where $u[k]$ is the vector of system inputs sampled at time kT_{samp}. The discretized system matrix is then related to the continuous system matrix as:

$$\boldsymbol{A}_d = \boldsymbol{I} + \boldsymbol{A}T_{samp} \tag{2.56}$$

The discrete-time eigenvalues are found by setting up the characteristic equation for the discrete system matrix:

$$det(\lambda_d \boldsymbol{I} - (\boldsymbol{I} + \boldsymbol{A}T_{samp})) = det(\lambda_d \boldsymbol{I} - \boldsymbol{A}_d) = 0 \tag{2.57}$$

The relation between the continuous-time eigenvalues and the forward Euler discrete-time system eigenvalues is:

$$\lambda_d = T_{samp} \cdot \lambda \tag{2.58}$$

2.6.3 Mapping Continuous-Time Eigenvalues to Z-Plane

The criterion for stability of the discrete system is that all eigenvalues are inside the unit circle on the complex z-plane[18]. The left half-plane in the continuous s-plane it thus mapped inside the unit circle of the z-plane, and the imaginary axis of the s-plane is mapped around the circumference of the unit circle in the z-plane. The relation between the continuous s-plane and the discrete z-plane is:

$$z = e^{sT_{samp}} \tag{2.59}$$

Using the forward Euler method of discretization, this relation is approximated as:

$$z = e^{sT_{samp}} \approx 1 + sT_{samp} \tag{2.60}$$

In order for a continuous eigenvalue to map within the stability region in the z-plane, it must be inside a circle of radius $\frac{1}{T_{samp}}$ with origin of (-1, 0) on the s-plane, as illustrated in figure 2.9.

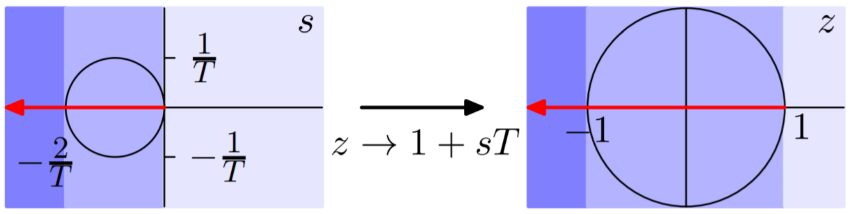


Figure 2.9: Mapping of continuous-time eigenvalues in s-plane to discrete-time eigenvalues in z-plane[18].

This leads to the following stability criterion for a continuous-time system discretized by forward Euler method:

$$\frac{-2}{T_{samp}} < Re\{\lambda\} < 0 \tag{2.61}$$

In some cases it is convenient to instead consider the product of the eigenvalues and the sample time. In which case, to achieve discrete system stability, the product of eigenvalues and sample time must lie within the unit circle with origin of (-1, 0) on the λT_{samp}-plane, as shown in figure 2.7 in section 2.5.2. In the case the stability criterion for the real part of the product becomes:

$$-2 < Re\{\lambda\} \cdot T_{samp} < 0 \tag{2.62}$$

Chapter 3

Programming Structure

3.1 NTNU Control Platform

A control platform for electric motor drives is currently under development at the department of Electric Power Engineering at NTNU. The control platform will be used for teaching and serve as a foundation for future research projects in motor drives at NTNU. The control platform hardware consists of an picoZed 7030 control board from Avnet mounted on a process interface board developed by SINTEF. The process interface board allows for the control of two two-level three-phase inverters and up to eight input measurements.

The control board is equipped with a Zynq-7030 SoC which contains two ARM floating-point processors (CPUs) and one programmable logic (FPGA) device. One of the CPUs runs a Linux program provided by The Switch Marine Drives, which is used for programming and monitoring of the remaining CPU and the FPGA. The remaining CPU and FPGA is available for programming by the user of the control platform[2]. The motor control system and creation of modulation signals is handled in the CPU, while the PWM modulator is run on the FPGA. The motor drive emulator which is the focus of this thesis will also be programmed on the FPGA.

3.2 Design of IP Cores in Xilinx System Generator

The code in the FPGA is built up of interconnected IP (Intellectual Property) cores which each performs different tasks. The IP cores developed in this thesis are made in Xilinx System Generator for DSP, which is an add-on for Simulink. The plug-in contains blockset libraries with blocks for creating IP core designs in Simulink. It allows for block diagram design and testing of HDL (Hardware Description Language) code in the Simulink environment. For implementation of the IP cores in the FPGA, the design is exported from System Generator to Vivado, from which the IP core design can be synthesized into HDL code and run on an FPGA. This is out of the scope for this thesis however, as the IP cores will only be developed and tested in System Generator.

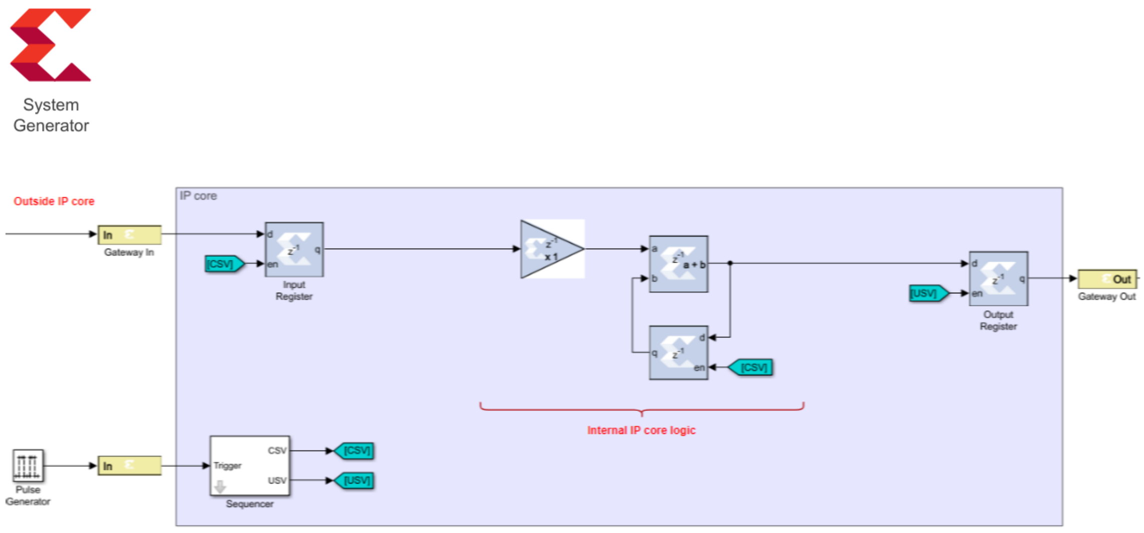


Figure 3.1: Generic IP core design in System Generator

Common components for all IP cores developed in this thesis are: gateway in/out blocks, input/output registers and sequencer for trigger pulse generation. A generic IP core design is shown in figure 3.1. A brief description of components used in IP core generation follows below.

System Generator token
All Simulink models with System Generator components must contain a System Generator token, which serves as a control panel for the simulation parameters for the Xilinx blockset elements. It can also be used to compile the System Generator design into HDL code. The FPGA clock period used in the simulation is also specified here.

Gateway In/Out
The Gateway In and Gateway Out blocks act as input/output ports to the IP core. The Gateway In block converts an external signal to the specified signal type, either boolean, fixed-point or floating point. In this thesis, floating point representation will not be used. The FPGA and processor on the control board communicate by the AXI4-Lite IP interface, which is a 32-bit communication protocol. This means that the word length is limited to 32-bit for data sent from processor to IP cores and vice versa. For this thesis, the internal logic in the IP cores is allowed to run with full output precision where possible. For future development of the emulator, the internal accuracy of the IP cores may also be limited to 32 bits in order to save computational resources in the FPGA.

Sequencer and registers

All IP cores contain a sequencer block which creates the enable pulses for the enabled components in the IP cores, like the input and output registers. The sequencer creates two pulse signals based on the input signal from the Pulse Generator clock in Simulink: the CSV pulse, which enables the input registers and other registers in the IP core, and the USV pulse, which enables the output registers and is delayed by half a period compared to the CSV pulse. At the enable pulse, the registers sample and hold the input signal until the next pulse. The period of the pulse signals define the discrete sampling time of the IP core.

To maintain synchronicity in the IP core, it is important that the maximum delay or latency in the paths between input and output registers in the IP core is shorter than the delay between the CSV and USV enable pulses[4]. The pulse period is defined in the Pulse Generator Simulink block as T_{samp}/T_{clk}. For the default parameters used in this thesis, $T_{samp} = 1\mu s$ and $T_{clk} = 10ns$, the result is a pulse period of:

$$\frac{T_{samp}}{T_{clk}} = \frac{1\mu s}{10ns} = 100 \tag{3.1}$$

Since the USV pulse which enables the output registers comes half a period after the CSV pulse, it follows that the maximum latency for the default FPGA clock speed and sample time is:

$$\frac{1}{2} \cdot \frac{T_{samp}}{T_{clk}} = \frac{1}{2} \cdot \frac{1\mu s}{\cdot 10ns} = 50 \tag{3.2}$$

Chapter 4

Development of Emulator IP Cores

The motor and load IP cores are modelled using the per unit (pu) system, which uses the rated data of the motor as base values. The most important advantage of using the pu system when using fixed-point representation is that the value ranges for different rated motors will be the same. The number of bits reserved for the integer part and for the fractional part can thus be set to give high resolution and accuracy without risking bit overflow.

In order to make the IP core designs generic, all parameter values are taken as inputs to the IP core instead of being "hardcoded" into the design. Parameters can then be easily changed if needed. Base value and per unit calculations are made in the local mask environment where the IP core is located in the Simulink model.

4.1 Three-Phase PMSM Motor

This modelling of the 3-phase motor consists of three separate IP cores: the dq transformation IP core, the motor IP core and the inverse dq transformation IP core. The connections between the IP cores and their internal logic are shown in appendix A.

4.1.1 Motor IP Core

The three-phase motor IP core is derived from the dq-frame differential equation set developed in section 2.3. Rearranging equation 2.17 yields the expressions:

$$\begin{aligned} \frac{di_d}{dt} &= \frac{\omega_n}{x_d}\left(u_d - r_s i_d + n x_q i_q\right) \\ \frac{di_q}{dt} &= \frac{\omega_n}{x_q}\left(u_q - r_s i_q - n x_d i_d - n\psi_m\right) \end{aligned} \tag{4.1}$$

Discretizing 4.1 using the forward Euler method yields:

$$\begin{aligned} i_d[k+1] &= i_d[k] + T_{samp} \cdot \frac{\omega_n}{x_d} \left(u_d[k] - r_s i_d[k] + n[k] x_q i_q[k] \right) \\ i_q[k+1] &= i_q[k] + T_{samp} \cdot \frac{\omega_n}{x_q} \left(u_q[k] - r_s i_q[k] - n[k] x_d i_d[k] - n[k] \psi_m \right) \end{aligned} \tag{4.2}$$

The electric motor torque is calculated as:

$$\tau_e = \psi_d \cdot i_q - \psi_q \cdot i_d \tag{4.3}$$

Where the flux linkages are related to the currents as:

$$\psi_d = x_d i_d + \psi_m \quad , \quad \psi_q = x_q i_q \tag{4.4}$$

The SysGen implementation of the three-phase motor is shown in figure A.4 in appendix A.

4.1.2 Synchronous Reference Frame Transformation IP Cores

The three-phase synchronous motor is modelled in the synchronous rotating (dq) reference frame. The three-phase voltage signals from the inverter must therefore be transformed to the dq plane to be used as inputs for the motor IP core. Likewise, the current outputs from the motor are transformed from the dq frame back into three-phase. Two IP cores are constructed for this purpose, one for dq-transformation of voltages and one for the inverse transformation of currents.

dq Transformation

The three-phase to dq transformation is the product of two transformations: Clarke ($\alpha\beta\gamma$) transformation and Park ($dq0$) transformation. The amplitude invariant Clarke transform is used, with the alpha-component aligned with the a-phase vector. Since the three-phase system is considered to be balanced the γ-component will always be zero and can therefore be neglected. The Clarke transformation for voltages is shown below:

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} 1 & \frac{-1}{2} & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix} \cdot \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} \tag{4.5}$$

Where:

$$u_a = \hat{u} \cos \omega t \ , \ u_b = \hat{u} \cos\left(\omega t - \frac{2\pi}{3} \right) \ , \ u_c = \hat{u} \cos\left(\omega t + \frac{2\pi}{3} \right) \tag{4.6}$$

and:

$$u_\alpha = \hat{u}\cos\omega t \quad , \quad u_\beta = \hat{u}\sin(\omega t) \tag{4.7}$$

For amplitude-invariant transformation of a balanced three-phase system, it also follows that the alpha component is identical to the phase a component. Therefore, only the beta component needs to be calculated. This saves some computational resources in the transformation. The alpha and beta components can then be expressed as:

$$u_\alpha = u_a \quad , \quad u_\beta = \frac{1}{\sqrt{3}} \cdot (u_b - u_c) \tag{4.8}$$

The Park transformation transforms the two-component AC system by rotating the reference frame counter-clockwise at the AC frequency. In this synchronously rotating reference frame, the two components appear as stationary DC values. The transformation is expressed as:

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} \tag{4.9}$$

Where the angle θ is the rotational angle of the alpha component, which is aligned with phase a in the original three-phase system. The sine and cosine of this angle which is needed for transformation is calculated in a separate IP core, see section 4.3. The dq transformation implemented in System Generator is shown in figure A.5 in appendix A.

Inverse dq transformation

Transforming the output dq currents from the motor back into three-phase currents is achieved by using the inverse Park and Clarke transformations. Combining the inverse transformation Park and Clarke matrices yields:

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \cos\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) \\ \cos\left(\theta + \frac{2\pi}{3}\right) & -\sin\left(\theta + \frac{2\pi}{3}\right) \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{4.10}$$

Which yields the expression for the three-phase currents as a function of the dq currents and the electrical angle:

$$\begin{aligned}
i_a &= i_d \cos\theta - i_q \sin\theta \\
i_b &= i_d \cos\left(\theta - \frac{2\pi}{3}\right) - i_q \cos\left(\theta - \frac{2\pi}{3}\right) \\
i_c &= i_d \cos\left(\theta + \frac{2\pi}{3}\right) - i_q \cos\left(\theta + \frac{2\pi}{3}\right)
\end{aligned} \tag{4.11}$$

Since calculating the sine/cosine of an angle is computationally demanding in FPGA, it is desirable to rewrite the equations for phase b and c so that the constant phase shift can be separated from the electrical angle. This is done by using the trigonometric identities:

$$\begin{aligned}
\sin(x \pm y) &= \sin(x)\cos(y) \pm \cos(x)\sin(y) \\
\cos(x \pm y) &= \cos(x)\cos(y) \mp \sin(x)\sin(y)
\end{aligned} \tag{4.12}$$

The three-phase currents can then be expressed by the dq currents and angle θ as:

$$\begin{aligned}
i_a &= i_d \cos\theta - i_q \sin\theta \\
i_b &= i_d \left(-\frac{1}{2}\cos(\theta) + \frac{\sqrt{3}}{2}\sin(\theta)\right) - i_q \left(-\frac{1}{2}\cos(\theta) - \frac{\sqrt{3}}{2}\sin(\theta)\right) \\
i_c &= i_d \left(-\frac{1}{2}\cos(\theta) - \frac{\sqrt{3}}{2}\sin(\theta)\right) - i_q \left(-\frac{1}{2}\cos(\theta) + \frac{\sqrt{3}}{2}\sin(\theta)\right)
\end{aligned} \tag{4.13}$$

In this form, only the sine/cosine of the electrical angle is needed for the inverse transformation. These are calculated in the electrical angle computation IP core, which is discussed in section 4.3.
The inverse dq transformation System Generator IP core is shown in appendix figure A.6.

4.2 Six-Phase PMSM Motor

The modelling of the six-phase motor consists of four separate IP cores: the VSD transformation IP core, the motor IP core, the two inverse VSD transformation IP cores. The inverse VSD transformation IP cores transforms the dq and z1z2 subspace motor parameters into three-phase currents and two-phase flux linkages, respectively.
The complete IP cores and their connections are shown in appendix B.

4.2.1 Motor IP core

In section 2.4, the six-phase per unit motor model equations are developed. Rearranging equation 2.29 yields the differential expressions for the dq and z1z2 subspaces:

$$
\begin{aligned}
&\frac{di_d}{dt} = \frac{\omega_n}{x_d}\left(u_d - r_s i_d + n x_q i_q\right) \quad , \quad \frac{di_q}{dt} = \frac{\omega_n}{x_q}\left(u_q - r_s i_q - n x_d i_d - n\psi_m\right) \\
&\frac{di_{z1}}{dt} = \frac{\omega_n}{x_{s\sigma}}\left(u_{z1} - r_s i_{z1} - n x_{s\sigma} i_{z2}\right) \quad , \quad \frac{di_{z2}}{dt} = \frac{\omega_n}{x_{s\sigma}}\left(u_{z2} - r_s i_{z2} + n x_{s\sigma} i_{z1}\right)
\end{aligned}
\tag{4.14}
$$

The equations for the dq subspace are identical to the three-phase motor. Discretizing the expressions in 4.14 using the Forward Euler method yields:

$$
\begin{aligned}
i_d[k+1] &= i_d[k] + T_{samp} \cdot \frac{\omega_n}{x_d}\left(u_d[k] - r_s i_d[k] + n[k] x_q i_q[k]\right) \\
i_q[k+1] &= i_q[k] + T_{samp} \cdot \frac{\omega_n}{x_q}\left(u_q[k] - r_s i_q[k] - n[k] x_d i_d[k] - n[k]\psi_m\right) \\
i_{z1}[k+1] &= i_{z1}[k] + T_{samp} \cdot \frac{\omega_n}{x_{s\sigma}}\left(u_{z1}[k] - r_s i_{z1}[k] - n[k] x_{s\sigma} i_{z2}[k]\right) \\
i_{z2}[k+1] &= i_{z2}[k] + T_{samp} \cdot \frac{\omega_n}{x_{s\sigma}}\left(u_{z2}[k] - r_s i_{z2}[k] + n[k] x_{s\sigma} i_{z1}[k]\right)
\end{aligned}
\tag{4.15}
$$

The z1z2 subspace does not contribute to electric torque production, so the expression for the electric torque is the same as for the three-phase motor, seen in equation 4.3. The System Generator IP core of the six-phase motor is shown in figure B.3 in appendix B.

4.2.2 Vector Space Decomposition IP Core

By using the rotating vector space decomposition method presented in [10], the six-phase system can be decoupled into three orthogonal subsystems. The transformation from six-phase to the dq, z1z2 and zero sequence subspaces is done by multiplying the six-phase vector with the VSD transformation matrix.

$$
\begin{bmatrix} u_d \\ u_q \\ u_{z1} \\ u_{z2} \\ u_{01} \\ u_{02} \end{bmatrix} = \boldsymbol{T}_{ss}^{r} \cdot \begin{bmatrix} u_{a1} \\ u_{a2} \\ u_{b1} \\ u_{b2} \\ u_{c1} \\ u_{c2} \end{bmatrix}
\tag{4.16}
$$

The transformation matrix is expressed as:

$$\boldsymbol{T}^{r}_{ss} = \frac{1}{3} \cdot \begin{bmatrix} \cos\theta & \cos\left(\theta - \frac{\pi}{6}\right) & \cos\left(\theta - \frac{2\pi}{3}\right) \\ -\sin\theta & -\sin\left(\theta - \frac{\pi}{6}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) \\ \cos\theta & \cos\left(\theta - \frac{7\pi}{6}\right) & \cos\left(\theta - \frac{2\pi}{3}\right) \\ -\sin\left(\theta - \pi\right) & -\sin\left(\theta - \frac{\pi}{6}\right) & -\sin\left(\theta - \frac{5\pi}{3}\right) \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdots$$

$$\cdots \begin{bmatrix} \cos\left(\theta - \frac{5\pi}{6}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) & \cos\left(\theta - \frac{3\pi}{2}\right) \\ -\sin\left(\theta - \frac{5\pi}{6}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) & -\sin\left(\theta - \frac{3\pi}{2}\right) \\ \cos\left(\theta - \frac{11\pi}{6}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) & \cos\left(\theta - \frac{\pi}{2}\right) \\ -\sin\left(\theta - \frac{5\pi}{6}\right) & -\sin\left(\theta - \frac{\pi}{3}\right) & -\sin\left(\theta - \frac{3\pi}{2}\right) \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.17}$$

The system is balanced, so the zero-sequence subspace will not be excited and is therefore neglected from the transformations and modelling. Writing out the expressions for the dq subspace yields the expressions:

$$\begin{aligned} u_d = \frac{1}{3}\Big(& u_{a1}\cos(\theta) + u_{a2}\cos\left(\theta - \frac{\pi}{6}\right) + u_{b1}\cos\left(\theta - \frac{2\pi}{3}\right) \\ &+ u_{b2}\cos\left(\theta - \frac{5\pi}{6}\right) + u_{c1}\cos\left(\theta - \frac{4\pi}{3}\right) + u_{c2}\cos\left(\theta - \frac{3\pi}{2}\right)\Big) \end{aligned} \tag{4.18}$$

$$\begin{aligned} u_q = -\frac{1}{3}\Big(& u_{a1}\sin(\theta) + u_{a2}\sin\left(\theta - \frac{\pi}{6}\right) + u_{b1}\sin\left(\theta - \frac{2\pi}{3}\right) \\ &+ u_{b2}\sin\left(\theta - \frac{5\pi}{6}\right) + u_{c1}\sin\left(\theta - \frac{4\pi}{3}\right) + u_{c2}\sin\left(\theta - \frac{3\pi}{2}\right)\Big) \end{aligned} \tag{4.19}$$

Similarly, for the z1z2 subspace:

$$\begin{aligned} u_{z1} = \frac{1}{3}\Big(& u_{a1}\cos(\theta) + u_{a2}\cos\left(\theta - \frac{7\pi}{6}\right) + u_{b1}\cos\left(\theta - \frac{2\pi}{3}\right) \\ &+ u_{b2}\cos\left(\theta - \frac{11\pi}{6}\right) + u_{c1}\cos\left(\theta - \frac{4\pi}{3}\right) + u_{c2}\cos\left(\theta - \frac{\pi}{2}\right)\Big) \end{aligned} \tag{4.20}$$

$$\begin{aligned} u_{z2} = -\frac{1}{3}\Big(& u_{a1}\sin(\theta - \pi) + u_{a2}\sin\left(\theta - \frac{\pi}{6}\right) + u_{b1}\sin\left(\theta - \frac{5\pi}{3}\right) \\ &+ u_{b2}\sin\left(\theta - \frac{5\pi}{6}\right) + u_{c1}\sin\left(\theta - \frac{\pi}{3}\right) + u_{c2}\sin\left(\theta - \frac{3\pi}{2}\right)\Big) \end{aligned} \tag{4.21}$$

Using the trigonometric identities in equation 4.12 to rewrite the dq subspace expressions yields:

$$\begin{aligned} u_d = \frac{1}{3}\Big(& u_{a1}\left(\cos\theta\right) + u_{a2}\left(\frac{1}{2}\sin\theta + \frac{\sqrt{3}}{2}\cos\theta\right) + u_{b1}\left(\frac{\sqrt{3}}{2}\sin\theta - \frac{1}{2}\cos\theta\right) \\ & + u_{b2}\left(\frac{1}{2}\sin\theta - \frac{\sqrt{3}}{2}\cos\theta\right) + u_{c1}\left(-\frac{\sqrt{3}}{2}\sin\theta - \frac{1}{2}\cos\theta\right) + u_{c2}\left(-\sin\theta\right)\Big) \end{aligned} \tag{4.22}$$

$$\begin{aligned} u_q = -\frac{1}{3}\Big(& u_{a1}\left(\sin\theta\right) + u_{a2}\left(\frac{\sqrt{3}}{2}\sin\theta - \frac{1}{2}\cos\theta\right) + u_{b1}\left(\frac{-1}{2}\sin\theta - \frac{\sqrt{3}}{2}\cos\theta\right) \\ & + u_{b2}\left(-\frac{\sqrt{3}}{2}\sin\theta - \frac{1}{2}\cos\theta\right) + u_{c1}\left(-\frac{1}{2}\sin\theta + \frac{\sqrt{3}}{2}\cos\theta\right) + u_{c2}\left(\cos\theta\right)\Big) \end{aligned} \tag{4.23}$$

And, for the z1z2 subspace:

$$u_{z1} = \frac{1}{3}\left(u_{a1} - \frac{\sqrt{3}}{2}u_{a2} - \frac{1}{2}u_{b1} + \frac{\sqrt{3}}{2}u_{b2} - \frac{1}{2}u_{c1}\right) \tag{4.24}$$

$$u_{z2} = \frac{1}{3}\left(\frac{1}{2}u_{a2} - \frac{\sqrt{3}}{2}u_{b1} + \frac{1}{2}u_{b2} + \frac{\sqrt{3}}{2}u_{c1} - u_{c2}\right) \tag{4.25}$$

The VSD transformation System Generator IP core is implemented using equations 4.22-4.25 and is shown in figure B.5 in appendix B.

4.2.3 Inverse Vector Space Decomposition IP cores

Two inverse VSD IP cores are needed for the model, one to transform the motor currents back to the six-phase stationary reference frame, and one to transform the motor flux linkages to the two-phase stationary reference frame. These flux linkages are used for decoupling and sensorless estimation of rotor position in the drive control system.

dqz1z2 -> 2xabc transformation IP core

The inverse VSD transformation matrix is used to develop the expressions for the currents used in this IP core.

$$\begin{bmatrix} i_{a1} \\ i_{a2} \\ i_{b1} \\ i_{b2} \\ i_{c1} \\ i_{c2} \end{bmatrix} = \boldsymbol{T}_{ss}^{-r} \cdot \begin{bmatrix} i_d \\ i_q \\ i_{z1} \\ i_{z2} \\ i_{01} \\ i_{02} \end{bmatrix} \tag{4.26}$$

The inverse transformation matrix is given as[10]:

$$\boldsymbol{T}_{ss}^{-r} = \begin{bmatrix} \cos\theta & -\sin\theta & \cos\theta & -\sin(\theta-\pi) & 1 & 0 \\ \cos\left(\theta-\frac{\pi}{6}\right) & -\sin\left(\theta-\frac{\pi}{6}\right) & \cos\left(\theta-\frac{7\pi}{6}\right) & -\sin\left(\theta-\frac{\pi}{6}\right) & 0 & 1 \\ \cos\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta-\frac{2\pi}{3}\right) & \cos\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta-\frac{5\pi}{3}\right) & 1 & 0 \\ \cos\left(\theta-\frac{5\pi}{6}\right) & -\sin\left(\theta-\frac{5\pi}{6}\right) & \cos\left(\theta-\frac{11\pi}{6}\right) & -\sin\left(\theta-\frac{5\pi}{6}\right) & 0 & 1 \\ \cos\left(\theta-\frac{4\pi}{3}\right) & -\sin\left(\theta-\frac{4\pi}{3}\right) & \cos\left(\theta-\frac{4\pi}{3}\right) & -\sin\left(\theta-\frac{\pi}{3}\right) & 1 & 0 \\ \cos\left(\theta-\frac{3\pi}{2}\right) & -\sin\left(\theta-\frac{3\pi}{2}\right) & \cos\left(\theta-\frac{\pi}{2}\right) & -\sin\left(\theta-\frac{3\pi}{2}\right) & 0 & 1 \end{bmatrix} \tag{4.27}$$

Writing out the expressions for the six-phase currents, disregarding the zero-sequence subspace and using the trigonometric identities in equation 4.12 to separate the phase shifts from the electrical angle yields:

$$\begin{aligned} i_{a1} &= i_d\cos\theta - i_q\sin\theta + i_{z1} \\ i_{a2} &= i_d\left(\frac{\sqrt{3}}{2}\cos\theta + \frac{1}{2}\sin\theta\right) - i_q\left(-\frac{1}{2}\cos\theta + \frac{\sqrt{3}}{2}\sin\theta\right) - \frac{\sqrt{3}}{2}i_{z1} + \frac{1}{2}i_{z2} \\ i_{b1} &= i_d\left(-\frac{1}{2}\cos\theta + \frac{\sqrt{3}}{2}\sin\theta\right) - i_q\left(-\frac{\sqrt{3}}{2}\cos\theta - \frac{1}{2}\sin\theta\right) - \frac{1}{2}i_{z1} - \frac{\sqrt{3}}{2}i_{z2} \\ i_{b2} &= i_d\left(-\frac{\sqrt{3}}{2}\cos\theta + \frac{1}{2}\sin\theta\right) - i_q\left(-\frac{1}{2}\cos\theta - \frac{\sqrt{3}}{2}\sin\theta\right) + \frac{\sqrt{3}}{2}i_{z1} + \frac{1}{2}i_{z2} \\ i_{c1} &= i_d\left(-\frac{1}{2}\cos\theta - \frac{\sqrt{3}}{2}\sin\theta\right) - i_q\left(\frac{\sqrt{3}}{2}\cos\theta - \frac{1}{2}\sin\theta\right) - \frac{1}{2}i_{z1} + \frac{\sqrt{3}}{2}i_{z2} \\ i_{c2} &= i_d\left(-\sin\theta\right) - i_q\left(\cos\theta\right) - i_{z2} \end{aligned} \tag{4.28}$$

The dqz1z2 -> 2xabc transformation IP core is implemented using the expressions in 4.28 and is shown in figure B.6 in appendix B.

dqz1z2->alphabeta transformation IP core

The relation between the dq and z1z2 subspace flux linkages and the two-phase stationary reference frame is given as:

$$\begin{bmatrix} \psi_\alpha \\ \psi_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 1 & 0 \\ \sin\theta & \cos\theta & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \psi_d \\ \psi_q \\ \psi_{z1} \\ \psi_{z2} \end{bmatrix} \tag{4.29}$$

The dqz1z2->alphabeta transformation IP core is implemented based on 4.29, as shown in figure B.7 in appendix B.

4.3 Electrical Angle Computation IP Core

This IP core performs two tasks. Firstly, it computes the electrical motor angle by integrating the motor speed. Secondly, it calculates the sine and cosine of the electrical angle, which are needed for the transformations performed in the motor IP cores. The SysGen block diagram IP core is shown in figure 4.1.

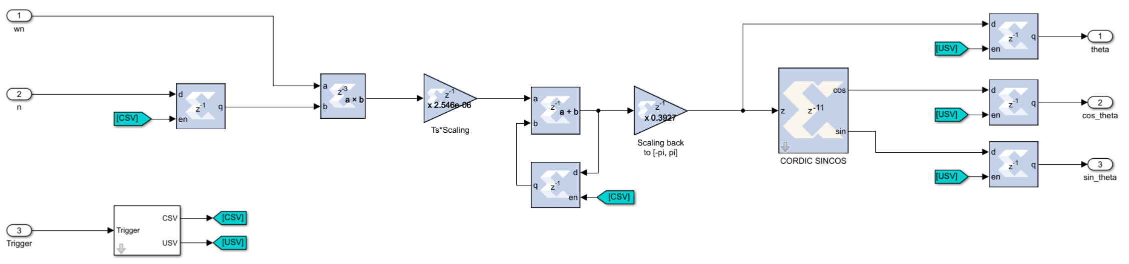


Figure 4.1: Block diagram of angle computation IP core in System Generator.

4.3.1 Computing Electrical Angle by Integrating Motor Speed

The motor speed is input to the IP core in pu.

$$n[pu] = \frac{N}{N_n} = \frac{\Omega_{mech}}{\Omega_{n,mech}} = \frac{p}{\omega_n} \cdot \Omega_{mech} \tag{4.30}$$

The electrical angle is found from the motor speed as:

$$\begin{aligned}\theta(t) &= p \cdot \theta_{mech}(t) = p \cdot \int_{t_0}^{t} \Omega_{mech}(t)dt + \theta(t_0) = p \cdot \int_{t_0}^{t} \frac{\omega_n}{p} \cdot n(t)dt + \theta(t_0) \\ &= \omega_n \cdot \int_{t_0}^{t} n(t)dt + \theta\left(t_0\right)\end{aligned} \tag{4.31}$$

Discretizing equation 4.31 using the forward Euler method yields:

$$\theta[k+1] = \theta[k] + T_{samp}\left(\omega_n \cdot n[k] + \theta(0)\right) \tag{4.32}$$

Wrapping Integrator in System Generator

The integrator is implemented in SysGen as shown in figure 4.1. An adder-block adds its input, scaled by T_{samp} with its previous output which is passed through a feedback register. This is the discrete implementation of an integrator, also referred to as an accumulator. The CORDIC block used for calculation of sine/cosine of the angle discussed in the next section only accepts inputs between $-\pi$ and π. Therefore, the integrator output must wrap between these limits. This can be done by making some logic which compares the integrator output to the minimum threshold and maximum threshold values $[-\pi, \pi]$ and sets the value of the feedback register accordingly. However, the added logic increases the hardware resources necessary for the design and adds a delay of T_{samp} for every wrap.

The more efficient method of achieving the wrapping is to exploit the overflow of the signed fixed-point binary numbers. Since the most significant bit is used to represent the sign of the number, the maximum positive value for a signed 32.28-bit binary number is $0111.11 \cdots 11 = 7.99...$ If the adder block is set to wrap at overflow, the following will occur when adding the smallest possible number to the maximum value:

$$0111.11 \cdots 11 + 0000.00 \cdots 01 = 1000.00 \cdots 00 = -8 \tag{4.33}$$

The output wraps between $[-8, 7.99..]$. By scaling the speed input to the accumulator by a factor of $8/\pi$, and the scaling the accumulator output by $\pi/8$, the accumulator can be made to wrap the angle between the desired values.

4.3.2 Calculating sine/cosine of Angle With CORDIC

The sine and cosine of the electrical angle theta must be calculated as part of the transformations for the 3-phase and 6-phase motor models. Computing the sine and cosine of an angle can be done in the FPGA in two main ways. One approach is to use a look-up table. This is simple, but requires the table to be stored in memory. Higher accuracy will require a bigger table. The DDS Compiler block in the Xilinx blockset can be used for this purpose. The other approach is to use a CORDIC SINCOS block from the Xilinx Reference blockset. This block utilizes a fully parallel CORDIC (COordinate Rotating DIgital Computer) algorithm in Circular Rotation mode to calculate the sine and cosine of the electrical angle[19]. In this thesis, the choice was made to use the latter method.

The CORDIC algorithm is an iterative vector rotation method of calculating hyperbolic and trigonometric functions. Created by Volder[20] in 1956, it is widely used in hand calculators and other applications where methods like look-up tables or power series are not efficient due to limited computing resources[21]. The algorithm calculates these functions in a hardware-efficient way by only using shift operations and additions. The hardware complexity of CORDIC is roughly equivalent to a single multiplier with the same word size, and the result converges by one bit per rotation or iteration[21].

The CORDIC algorithm is derived from the general rotation transform which rotates a vector by angle θ:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.34}$$

Which can be rewritten as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \cos\theta \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.35}$$

By recognizing that rotation of a vector by an arbitrary angle θ is the same as rotating by several smaller angles ϕ_i which sum up to the total angle, and choosing the smaller angles so that $\tan\phi_i = 2^{-i}$, the multiplication of the tangent function can be achieved by a simple bit shift operation[21].

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \cos\phi_i \begin{bmatrix} 1 & -\tan\phi_i \\ \tan\phi_i & -1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.36}$$

The direction of rotation is decided based on the sign of the residual angle. Since $\cos\phi_i = \cos(-\phi_i)$, it is constant regardless of direction of rotation and can be considered as a scaling factor.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = K_i \begin{bmatrix} 1 & -d_i \cdot 2^{-i} \\ d_i \cdot 2^{-i} & -1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4.37}$$

Where: $K_i = \cos(tan^{-1} \cdot 2^{-i}) = \frac{1}{\sqrt{1+2^{-2i}}}$ and $d_i = \pm 1$.

The product of the scaling factors for each iteration K_i converges to 0.6072 as number of iterations approaches infinity, and can be multiplied at the end of the process to achieve the correct scaling of the vector. For calculation of sine and cosine of the angle, the initial coordinate values are set to $x_0 = 1$ and $y_0 = 0$. After n number of rotations, the sine and cosine are found as $\sin\phi_n = K_i \cdot y_n$ and $\cos\phi_n = K_i \cdot x_n$. Alternatively, the initial coordinate values can be set to $x_0 = 0.6072$ and $y_0 = 0$ to account for the scaling from the beginning. Then, no multiplication is needed in the computation.

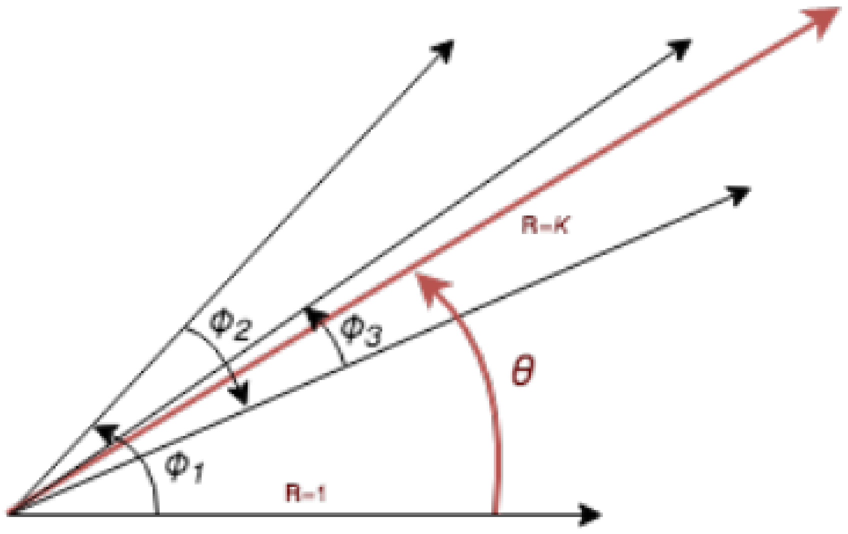


Figure 4.2: Iterative vector rotation to find angle in CORDIC algorithm[22]. The vector magnitude scales with $\cos \phi_i$ for each iteration.

The CORDIC algorithm is implemented in the CORDIC SINCOS System Generator block in three steps[19]:

1. Coarse angle rotation
   The CORDIC algorithm only converges for angles between $\frac{-\pi}{2}$ and $\frac{\pi}{2}$, i.e. the angle must lie in the first or fourth quadrant. If this is not the case, the angle is reflected to the first or fourth quadrant by adding/subtracting $\frac{\pi}{2}$.
2. Fine angle rotation
   This is the iterative vector rotation method described above. The initial vector magnitudes are set to $y_{in} = 0$ and $x_{in} = 0.6072$ to avoid having to multiply with scaling factor.
3. Angle correction
   If the angle was reflected in step 1, the necessary corrections are made here.

4.4 Mechanical Load IP Core

The mechanical load IP core calculates the motor speed based on the input motor torque and the load torque. Two load torque profile choices are available: either a constant load torque independent of motor speed or a load torque which is proportional to the square of the speed. Mechanical losses are neglected in the model, and the load is connected directly to the motor shaft, i.e. no gearbox. The speed calculation is based on Newton's second law for rotational motion:

$$J\frac{d\Omega}{dt} = T_e - T_L \tag{4.38}$$

Where:

Ω	:	*Rotational speed of shaft* $[rad/s]$
J	:	*Total moment of inertia* $[kg \cdot m^2]$
T_e	:	*Electrical motor torque* $[Nm]$
T_L	:	*Mechanical load torque* $[Nm]$

As for the motor IP cores, it is convenient to scale the parameters in the mechanical load IP core using the per unit system, as different rated motors can have vastly different speed and torque ranges. Scaling the values to per unit allows for high bit precision, as more of the 32 bits available can be used without risking overflow.
In per unit, equation 4.38 can be expressed as[6]:

$$T_m\frac{dn}{dt} = \tau_e - \tau_L \tag{4.39}$$

Where:

n	:	*Speed in per unit*
T_m	:	*Mechanical time constant* $[s]$
τ_e	:	*Electrical motor torque in per unit*
τ_L	:	*Mechanical load torque in per unit*

Discretizing 4.39 using the forward Euler method yields:

$$n[k+1] = n[k] + T_{samp}\left(\frac{\tau_e[k] - \tau_L[k]}{T_m}\right) \tag{4.40}$$

The same base values used for the motor is used for per unit scaling of the mechanical load IP core. The base power and speed of the motor is needed for calculation of the pu torque and speed and is calculated in the mechanical load mask initialization commands. The mechanical time constant is calculated as[6]:

$$T_m = \frac{J \cdot \Omega_n^2}{S_n} \tag{4.41}$$

Where S_n is the motor base power and Ω_n is the motor base speed in [rad/s].

In the case where the load torque is speed dependent, it is calculated as:

$$\tau_L = k_n \cdot sign(n) \cdot n^2 \tag{4.42}$$

Where k_n is a constant. The information on the direction of rotation is lost when squaring the speed, so $sign(n)$ is included to ensure that the load torque is opposing the motor torque regardless of direction of rotation.

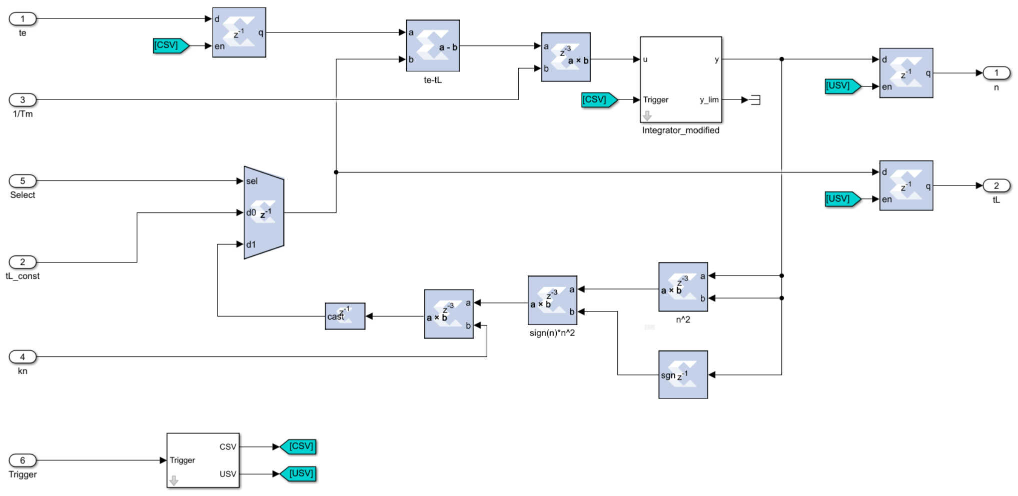


Figure 4.3: Mechanical load implementation in System Generator.

The choice of load profile in SysGen is made by using a multiplexer which selects its output based on the select-port input value, which is either 0 or 1. This value can be set in the dialog window of the mask. As seen in Figure 4.3, the speed dependent load torque is calculated in a feedback loop where the speed is squared and multiplied by the speed-torque constant k_n and the sign of n. Since the word length after each full-precision multiplication is doubled, the load torque is converted down to 64 bits before entering the multiplexer to prevent excessively large word lengths.

Chapter 5

Simulation Results

In order to test the performance of the developed System Generator IP cores, a three-phase and a six-phase motor drive Simulink model developed for the motor drives course at the Department of Electric Power Engineering at NTNU is used. Both models consists of power electronic inverters supplying power from a DC link to a motor, which drives a mechanical load. The switching of the inverters are controlled by PWM signals. Blanking time and turn-off time of switching components in the inverters are also accounted for in the PWM modulation.

The discrete IP cores for motor and mechanical load are inserted into these Simulink models, replacing the existing continuous Simulink blocks used in the models. The modified three-phase and six-phase models are then simulated and their output is compared to the original models. Henceforth, the modified models containing the System Generator IP cores will be referred to as the "SysGen models", and the original models as the "reference models".

If nothing else is stated, the SysGen models are run with a discrete sample time T_{samp} of 1 microsecond and FPGA clock speed T_{clk} of 10 nanoseconds. The models are run for 1 second, starting at zero speed and accelerating up to the motor nominal speed. The electrical motor torque, mechanical load torque and speed from the simulations are logged with a resolution of 1 microsecond and plotted.

Three different cases are studied for both the three-phase and the six-phase models:

- **Case 1: SysGen model vs. reference model**
  Compare the torque and speed outputs from the SysGen models with default sample time T_{samp} and FPGA clock time T_{clk} to the reference models.

- **Case 2: Comparing SysGen model sample times**
  Compare output from SysGen models with three different sample times: $T_{samp} = 0.5\mu s$, $T_{samp} = 1\mu s$ and $T_{samp} = 2\mu s$.

  The forward Euler method used in discretization of the model is cheap to implement, but requires very low sample times to be accurate and stable. In order to examine the effect of changing the discrete sample time T_{samp} on the simulation accuracy, the SysGen models are simulated with both half and double the default sample time: $T_{samp} = 2\mu s$ and $T_{samp} = 0.5\mu s$.

- **Case 3: Comparing SysGen model FPGA clock speeds**
  Compare output results from SysGen models running at FPGA clock speed $T_{clk} = 100ns$ with default, $T_{clk} = 10ns$.

  The models are run with an FPGA clock speed of 100ns, which is 10 times slower than the default value. Using this new clock speed leads to a significant reduction in the simulation time for the Simulink SysGen models: from around 12 hours to 2 hours for the six-phase model. The simulation output of the models with different clock speeds are therefore compared.

5.1 Three-Phase Model

The three-phase model consists of a three-phase two-level voltage source inverter with a DC-link voltage of 1155V connected to a three-phase motor, which drives a mechanical load.

The control system consists of an asymmetrical PWM modulator with the reference signals generated by a digital current controller operating in the dq-frame. The triangular PWM carrier signal frequency is 1.5kHz. There are no outer control loops, such as a speed controller in the control system. A torque reference signal is given as input to the controller, which is then converted to d-axis and q-axis current reference signals.

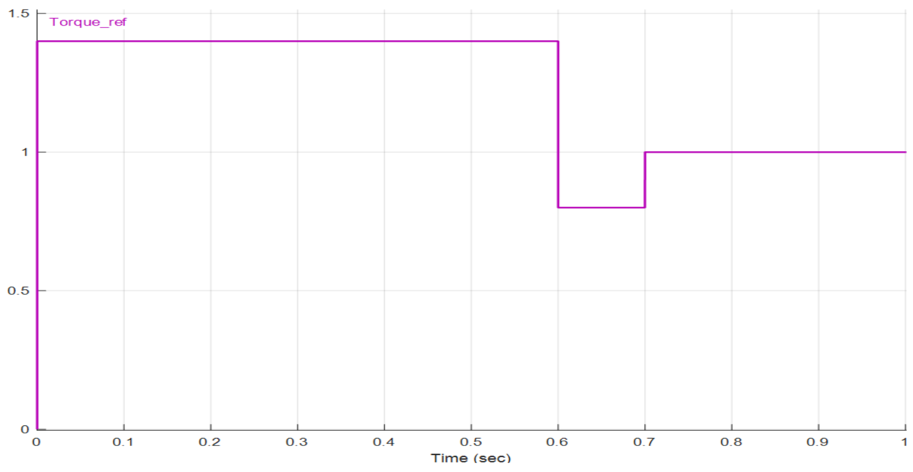
The motor and load parameters used in the simulations of the reference and SysGen model are shown in table 5.1.

5.1.1 Model Parameters

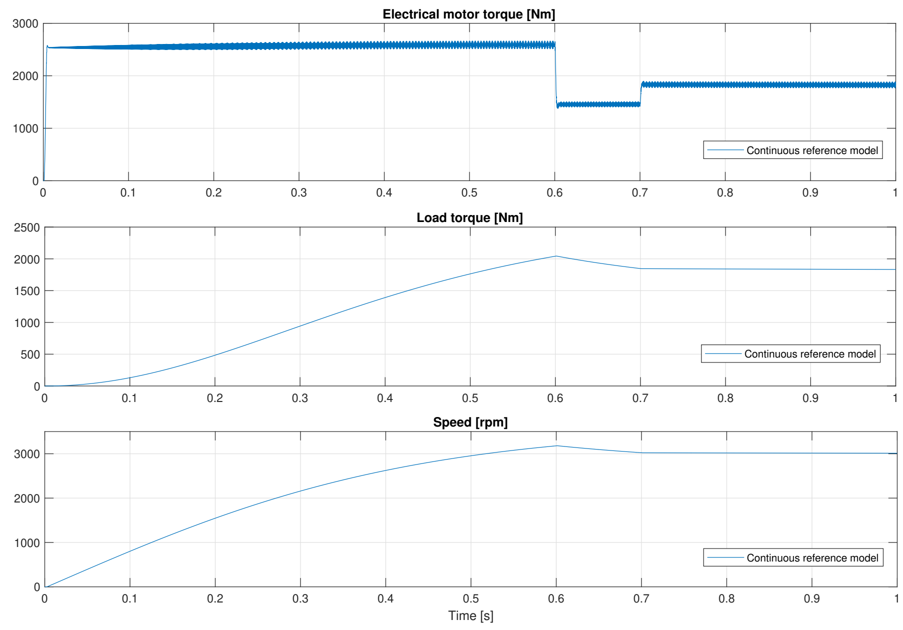
Table 5.1: Motor and mechanical load parameters used in simulation of three-phase model. Per unit base for the mechanical load is the same as for the motor.

Three-phase IPMSM motor	
Nominal line-line voltage, U_n	690 V_{rms}
Nominal current, I_n	478 A_{rms}
Nominal frequency, f_n	50 Hz
Nominal speed, N_n	3000 rpm
Stator resistance, r_s	0.009 pu
d-axis reactance, x_d	0.4 pu
q-axis reactance, x_q	1.0 pu
Magnet flux linkage, ψ_m	0.66 pu
Mechanical load	
Mechanical time constant, T_m	0.5 s
Torque-speed constant, k_n	1 pu

As mentioned above, the control system for the three-phase model does not contain a speed controller, so a reference speed can not be set directly. Instead, a torque reference is explicitly given, which is used for calculation of the current reference signals in the controller. In order to achieve nominal motor speed in the simulation, the torque-speed constant k_n of the mechanical load is set to 1 pu, so that the load torque at rated speed is also equal to 1 pu. A torque reference signal, shown in figure 5.1a was created and used as input to the torque controller. The torque reference signal for the controller starts at maximum torque 1.4 pu for fast acceleration. At t=0.6s it steps down to 0.8 pu before stepping back up to 1 pu at 0.7s. This was done to create some dynamics in the electrical motor torque, emulating the behaviour of a speed controller and to test the capability of the SysGen model to follow sudden changes in reference. The resulting simulation output from the three-phase reference model is shown in figure 5.1b.



(a) Torque reference signal in per unit for three-phase motor controller.

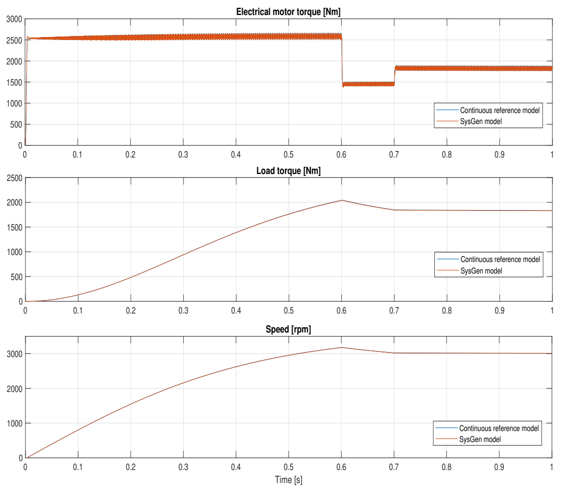


(b) Electrical torque, mechanical load torque and speed output from 3-phase continuous reference model

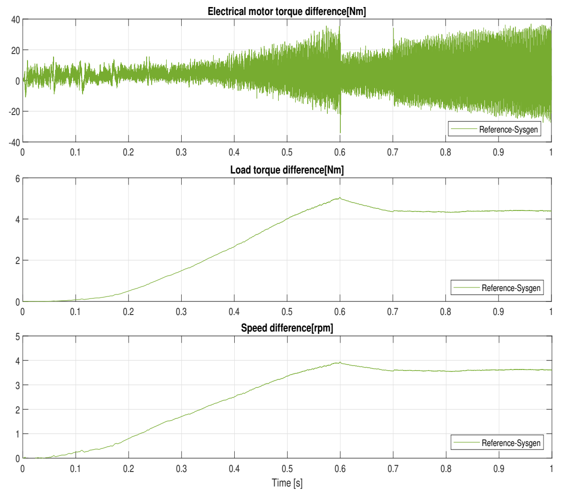
Figure 5.1: Torque reference signal for three-phase models and reference model output values.

5.1.2 Case 1: SysGen Model vs. Reference Model

The speed and torque outputs from the three-phase SysGen and reference model are plotted in figure 5.2a. The difference between reference and SysGen outputs is shown in figure 5.2b.



(a) Torque and speed output for reference model and SysGen model.



(b) Differences in torque and speed output between models.

Figure 5.2: Case 1: Electrical motor torque, load torque and speed output of three-phase SysGen and reference model.

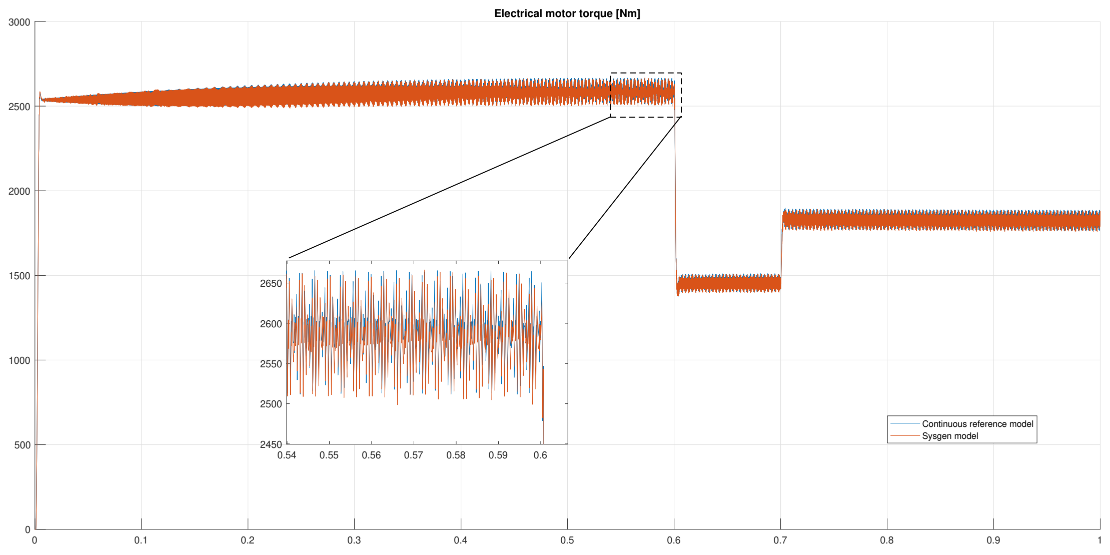


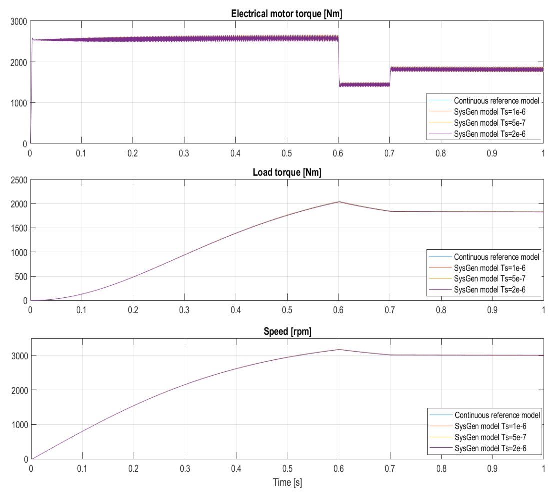
Figure 5.3: Case 1: Electrical torque of 3-phase continuous reference model and SysGen model

As seen from the plots above, the output electrical torque motor torque from the SysGen model is very similar to that of the reference model. The peak-to-peak torque ripple is very similar, around 125Nm for both models.

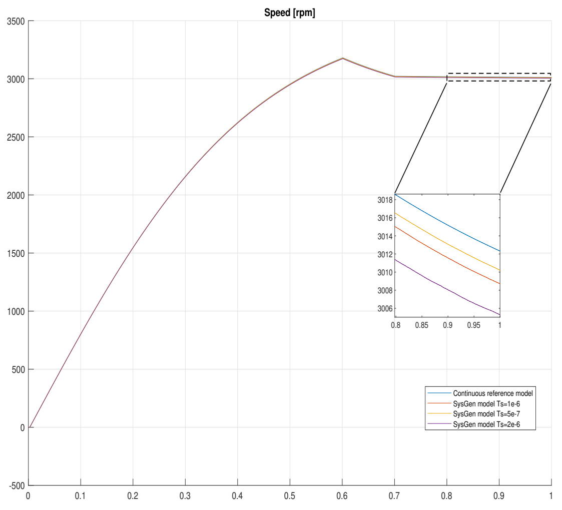
The average value of the torque seems to be slightly lower for the SysGen model however, which results in a speed difference of around 4 rpm after the motor reaches stationary speed, as seen in the speed subplot in figure 5.2b.

5.1.3 Case 2: Comparing SysGen Model Sample Times

The simulation outputs from the SysGen model with three different discrete sample times are shown in figure 5.4, along with the reference model output.



(a) Torque and speed output of three-phase SysGen model with different sample times.



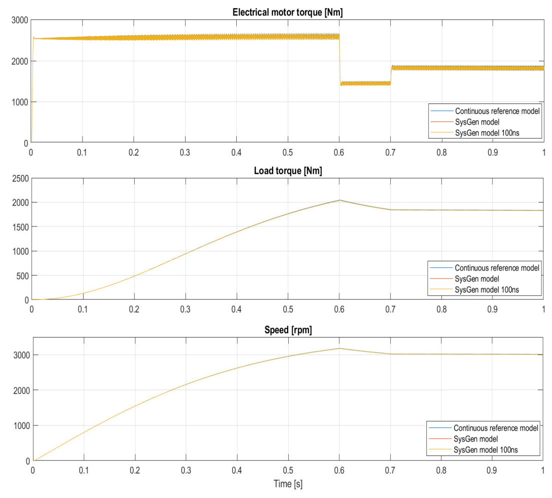
(b) Speed of 3-phase continuous reference model and SysGen model with different sample times.

Figure 5.4: Case 2: Electrical torque, mechanical load torque and speed of 3-phase continuous reference model and SysGen model with different sample times T_{samp}.

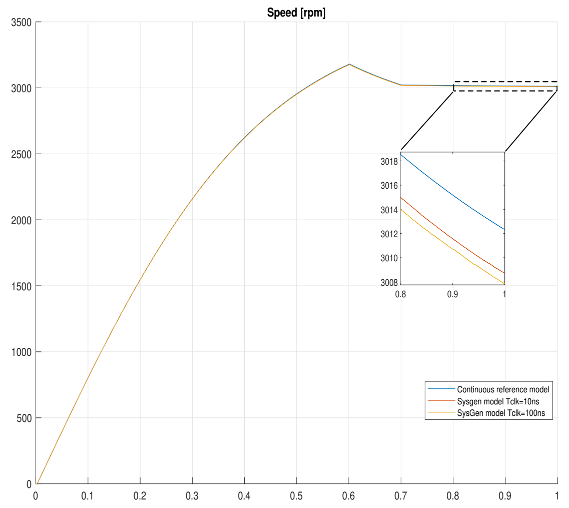
The shape of the electrical torque plots are very similar for all three sample times tested, but it can be seen from figure 5.4b that the speed difference increases for higher sample times. The speed difference is decreased by around 1 rpm for $T_s = 0.5\mu s$ and increased by around 2 rpm for $T_s = 2\mu s$, compared to $T_s = 1\mu s$. The difference in average motor torque appears to increase with the increase in sample time. This indicates that the speed/torque difference between the reference and SysGen models is a property of the discrete sample time and not due to truncation errors in the fixed-point logic in the IP cores.

5.1.4 Case 3: Comparing SysGen Model FPGA Clock Speeds

The simulation results from running the SysGen model at a clock speed of 100 ns are plotted in figure 5.5, along with the outputs from the reference and default clock speed SysGen model.



(a) Torque and speed output of three-phase SysGen model with different FPGA clock speeds.



(b) Speed of three-phase reference model and SysGen model with different FPGA clock speeds.

Figure 5.5: Case 3: Electrical torque, mechanical load torque and speed of three-phase continuous reference model and SysGen model with FPGA clock speed of 10ns and 100ns.

The 100 ns simulation is very similar to the 10 ns simulation. There is an additional speed deviation of 1 rpm at nominal motor speed.

5.2 Six-Phase Model

The six-phase model consists of two three-phase two-level voltage source inverters with a DC-link voltage of 1000V connected to a six-phase motor, which drives a mechanical load.

The switching of each inverter is controlled by an asymmetrical PWM modulator which gets its reference signal from a digital current controller, same as for the three-phase model. The triangular PWM carrier signal frequency is 3kHz. In addition to the inner current control loop, the control system also contains an outer speed control loop and field-weakening controller. A speed reference signal is given as input to the controller. For the simulations conducted in this thesis, the speed reference is set as constant, equal to nominal speed.

5.2.1 Model Parameters

The motor and load parameters used in the simulations of the reference and SysGen model are shown in table 5.2.

Table 5.2: Motor and mechanical load parameters used in simulation of six-phase model. Per unit base for the mechanical load is the same as for the motor.

Six-phase PMSM motor	
Nominal line-line voltage, U_n	601 V_{rms}
Nominal current, I_n	1310 A_{rms}
Nominal frequency, f_n	125 Hz
Nominal speed, N_n	500 rpm
Stator resistance, r_s	0.009 pu
d-axis reactance, x_d	0.3558 pu
q-axis reactance, x_d	0.3558 pu
Leakage reactance, x_σ	0.1 pu
Magnet flux linkage, ψ_m	0.9255 pu
Mechanical load	
Mechanical time constant, T_m	0.437 s
Torque-speed constant, k_n	0.916 pu

The torque and speed simulation output for the continuous reference model is shown in figure 5.6. After controller is enabled at 5 milliseconds, the motor torque climbs rapidly and settles at around 1.2 pu (62 kN). When nominal speed is reached, the torque drops and settles at just below 1 pu. The ripple in motor torque is around 3 kN, or 0.06 pu.

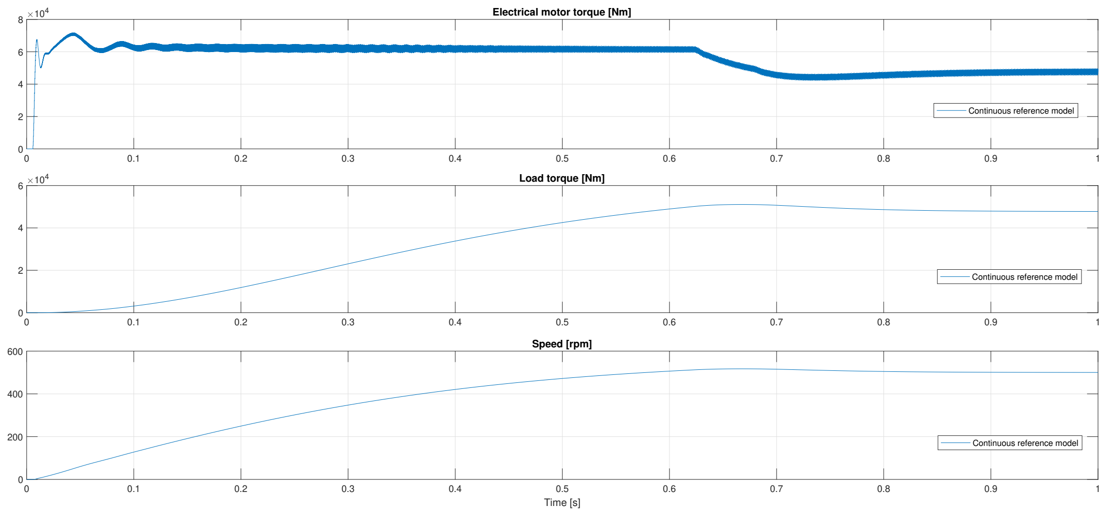
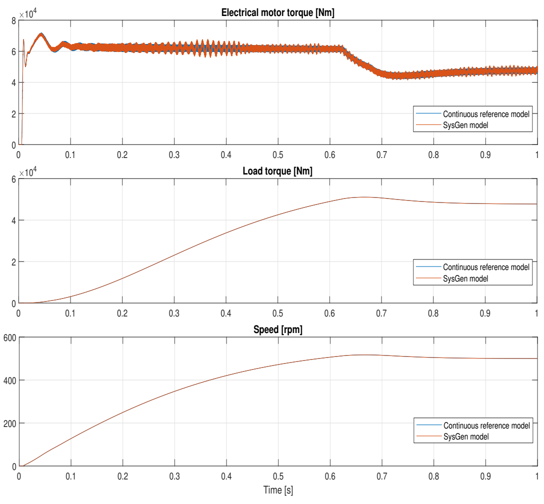


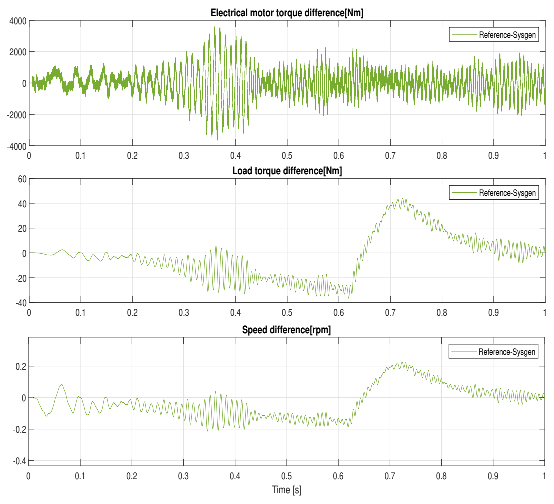
Figure 5.6: Electrical torque, mechanical load and speed of 6-phase continuous reference model

5.2.2 Case 1: SysGen Model vs. Reference Model

The simulation output from the six-phase SysGen model is plotted along with the reference model outputs in the figures below.



(a) Torque and speed output for reference model and SysGen model.



(b) Differences in torque and speed output between models.

Figure 5.7: Case 1: Electrical torque, mechanical load torque and speed of six-phase continuous reference model and SysGen model.

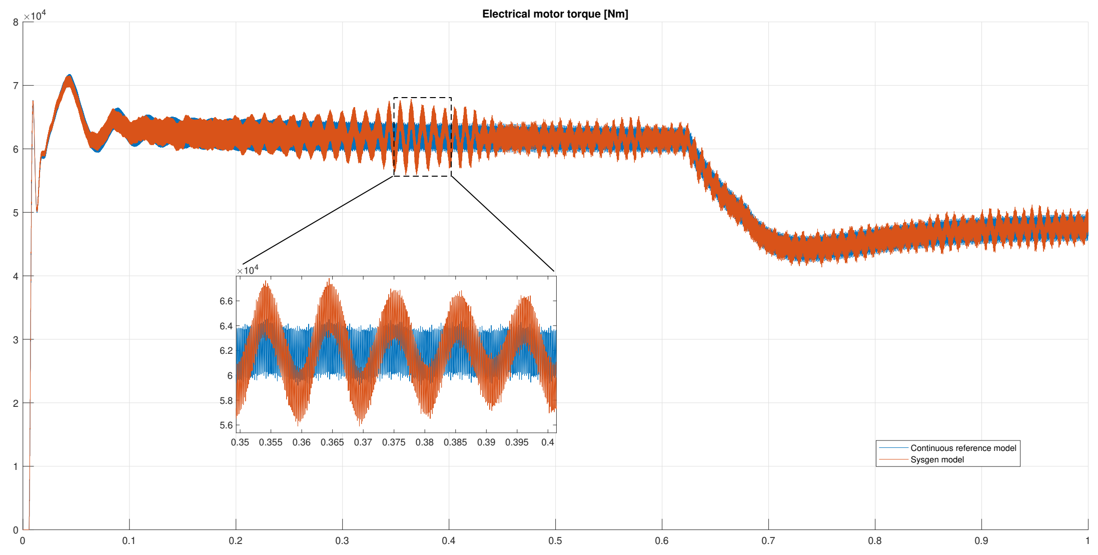


Figure 5.8: Case 1: Electrical torque plot of 6-phase continuous reference model and SysGen model

The plots for SysGen and reference starts out very similar, but oscillations in the electrical motor torque in the SysGen model starts appearing as the speed increases. As seen in figure 5.8 the oscillations are especially large at around 0.4 seconds when the motor speed is about 300 rpm, around 11 kN peak-to-peak at the most. As seen from figure 5.6, the reference model also show some torque oscillations at this time, but at a much lower amplitude.

The approximate frequency of the oscillations at different times in the simulation, found by counting the oscillation peaks between time intervals, are listed in table 5.3. As seen from the table, the frequency of oscillations starts out at around 80 Hz and climbs to around 130 Hz as the motor speed increases.

Table 5.3: Approximate oscillation frequency of electrical motor torque and approximate motor speed of six-phase SysGen model at different simulation time intervals.

Time	Number of	Oscillation	Motor speed	
interval	cycles	frequency [Hz]	[rpm]	[pu]
0.2s - 0.3s	8	80	300	0.6
0.3s - 0.4s	9	94	390	0.78
0.4s - 0.45s	4	114	430	0.86
0.5s - 0.58s	9	123	490	0.98
0.72s - 0.8s	9	130	508	1.016
0.83s - 0.9s	8	121	502	1.004
0.9s - 1s	12	131	500	1

The speed difference between models is much smaller than for the three-phase simulations. This is likely because the six-phase control system includes a speed controller, which sets the torque reference to achieve the desired speed.

5.2.3 Case 2: Comparing SysGen Model Sample Times

The torque outputs for the 6-phase SysGen model with three different sample times are plotted in figure 5.9, along with the torque output from the continuous model.

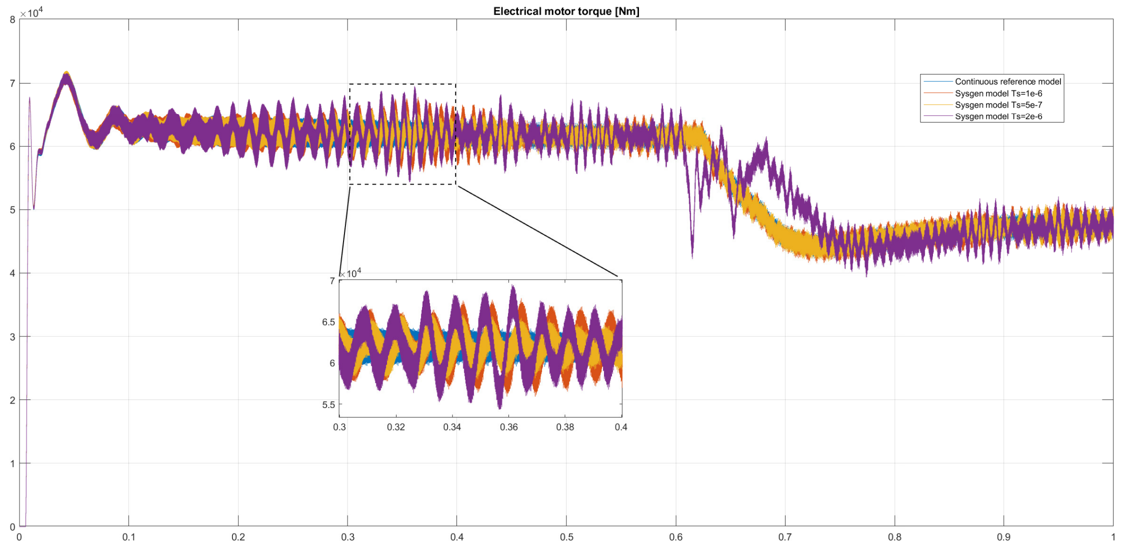


Figure 5.9: Case 2: Electrical torque plot of 6-phase motor with different sample times

The results show that adjusting the sample time has a significant impact on the amplitude of the oscillations in the electrical torque. Doubling the sample time increases the oscillation amplitude over almost the entire speed spectrum. The torque is especially inaccurate in the moments after the motor reaches the rated speed and the torque reference from the controller changes. Halving the sample time has the opposite effect, dampening the oscillations and bringing the result closer to the continuous reference model torque. The frequency of the oscillations seem to be around the same for all three sample times.

5.2.4 Case 3: Comparing SysGen Model FPGA Clock Speeds

The simulation results from running the SysGen model at a clock speed of 100 ns are plotted in figure 5.10, along with the outputs from the reference and default clock speed SysGen model.

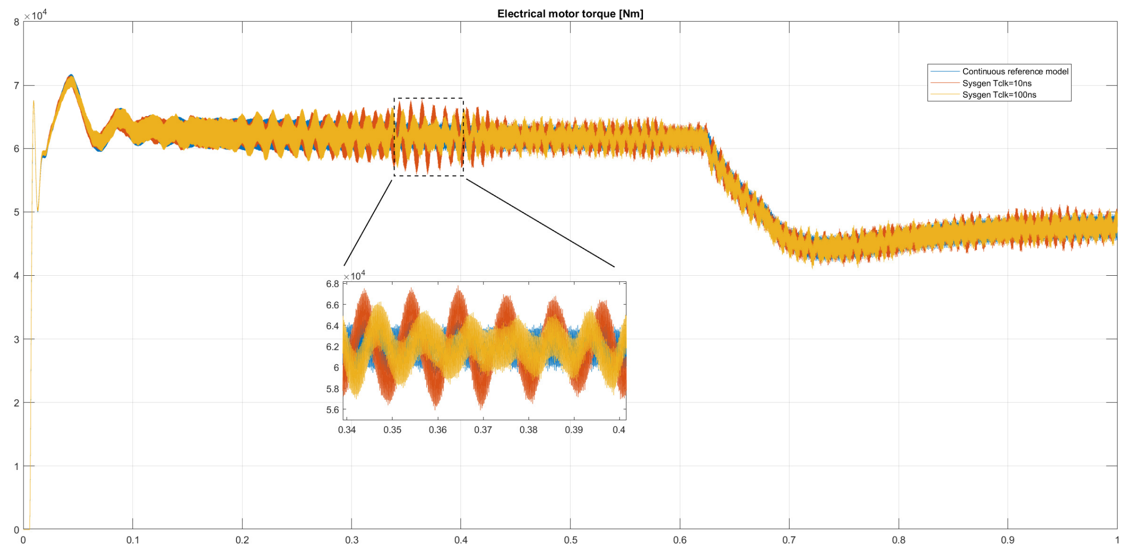


Figure 5.10: Case 3: Electrical torque plot of 6-phase continuous reference model and SysGen model with FPGA clock period of 10ns(default) and 100ns

Surprisingly, the maximum amplitude of the electrical motor torque oscillations is actually lower for the 100ns simulation compared to the 10ns simulation, around 9 kN at t=0.34 s.

Chapter 6

Discussion

6.1 Simulation Results

The simulation results show that even though the electrical torque is more accurate in the three-phase SysGen model than in the six-phase SysGen model, the three-phase SysGen model has a speed deviation of around 4 rpm at stationary speed for the default clock speed and sample time. This difference in speed is caused by a lower average electrical torque value for the SysGen model, which in turn results in a lower speed. This deviation is dependent on the sample time used; a higher sample time causes a higher speed difference. Reducing the FPGA clock speed also increases this deviation slightly. The reason why a similar speed deviation is not observed in the six-phase model is likely because this model contains a speed-controller, which adjusts the torque reference in order to reach the desired speed.
Since the speed deviation increases with increasing sample times, it is likely caused by error in the numerical method used. As discussed in section 2.5, the global error for the forward Euler method scales proportionally with the sample time. At the very end of the simulation, t=1s, the speed of the reference model is 3012.3 rpm. The SysGen model is slower than the reference model by: 2.1 rpm for $T_s = 0.5\mu s$, 3.6 rpm for $T_s = 1\mu s$ and 7 rpm for $T_s = 2\mu s$. It seems like the increase in global error is relatively close to proportional with the increase in sample time. It is hard to say for sure with only three different sample times used however, so more tests with a wider range of sample times should be conducted to confirm this.

Reducing the FPGA clock speed from maximum value of 10ns to 100ns drastically reduces simulation time of SysGen models in Simulink, from around 12 hours to around 2 hours. The accuracy of the simulations do not seem to be drastically worse either. The three-phase model had an increase in stationary speed deviation of 1 rpm, while the six-phase model actually had significantly lower amplitude in the oscillations. The reduced simulation time makes the verification of the IP core designs in Simulink more efficient. On the FPGA, the emulator runs in real-time, so changing the clock speed will not make a difference in this regard. However, reducing the clock frequency might reduce the power consumption of the FPGA.
A possible downside of reducing the clock speed to 100ns is that synchronization prob-

lems could occur at latencies between input and output registers above 5, instead of 50 for $T_{clk} = 10ns$ and $T_{samp} = 1\mu s$. This happens because the Pulse Generator block sets a pulse period of T_s/T_{clk}. This might affect the accuracy of the simulation in IP cores with higher latency than 5 between registers, for example in the VSD transformation IP cores[5]. The models seem to run fine at 100ns anyway, but it is something that should be kept in mind.

One possible explanation for the oscillations observed in the electrical torque in the six-phase SysGen model is that the input voltage signal is not sampled often enough, leading to an error in the average dq voltage which is manifested at the output as an AC on top of the electrical torque. The fact that the switching frequency in the six-phase model is twice as high as in the three-phase model may explain why the oscillations do not occur here.
However, as can be seen in figure 6.1, running the three-phase model at double the default switching frequency does not cause any similar oscillations to appear here, so this does not seem to be a likely explanation.

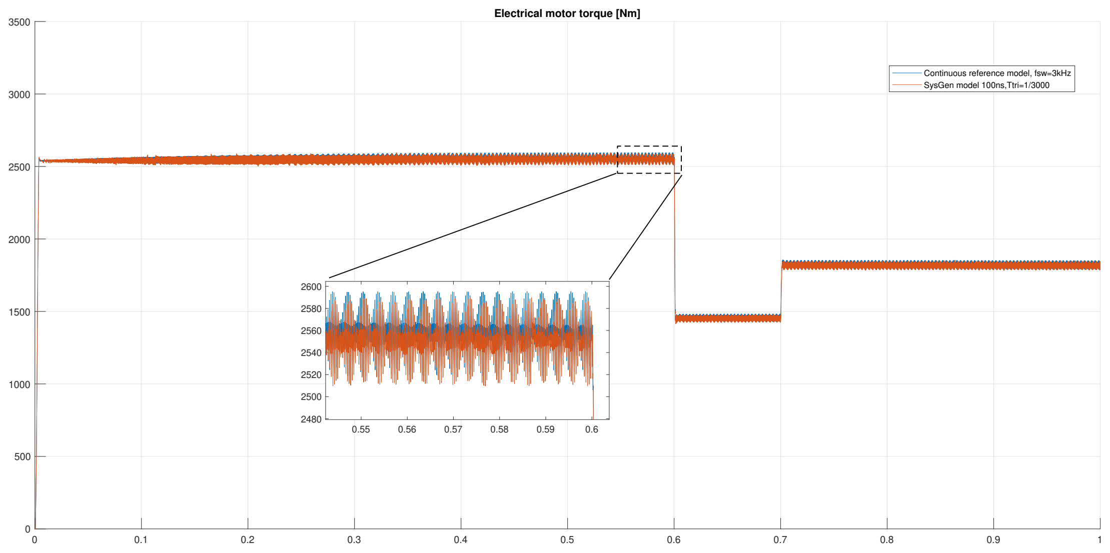


Figure 6.1: Electrical torque output for reference and SysGen three-phase models with double switching frequency, 3kHz.

6.2 Eigenvalue Analysis

In this section, an eigenvalue analysis of the three-phase and six-phase motor is performed in order to determine if the torque oscillations in the six-phase model are caused by instability of the discrete motor model, and explain why no such similar oscillations are observed in the simulations for the three-phase model.

6.2.1 Three-Phase Motor Model

As shown in section 2.6.1, the eigenvalues for the three-phase dq reference frame motor model are:

$$\lambda_{1,2} = \frac{-1}{2}\left(\frac{1}{T_d}+\frac{1}{T_q}\right) \pm \sqrt{\left(\frac{1}{2}\left(\frac{1}{T_d}-\frac{1}{T_q}\right)\right)^2 - (\omega_n n)^2} \tag{6.1}$$

Inserting the model parameters used in the simulations, this gives the following eigenvalues for the motor at rated speed ($n = 1pu$):

$$\begin{aligned}\lambda_{1,2} &= \frac{-1}{2}\left(\frac{1}{141.5ms}+\frac{1}{353.7ms}\right)\\ &\quad \pm \sqrt{\left(\frac{1}{2}\left(\frac{1}{141.5ms}-\frac{1}{353.7ms}\right)\right)^2 - (2\pi \cdot 50Hz \cdot 1)^2}\\ &= -4.95 \pm j314.15\end{aligned} \tag{6.2}$$

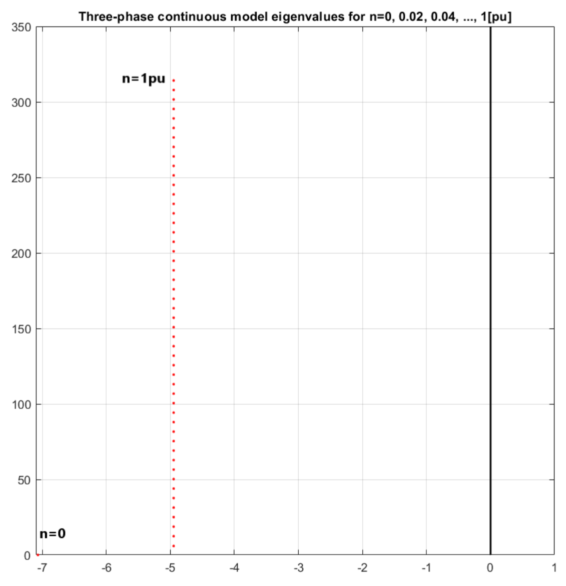
The relative damping ratio is:

$$\zeta = \frac{4.95}{\sqrt{4.95^2 + 314.15^2}} = 0.0157 \tag{6.3}$$

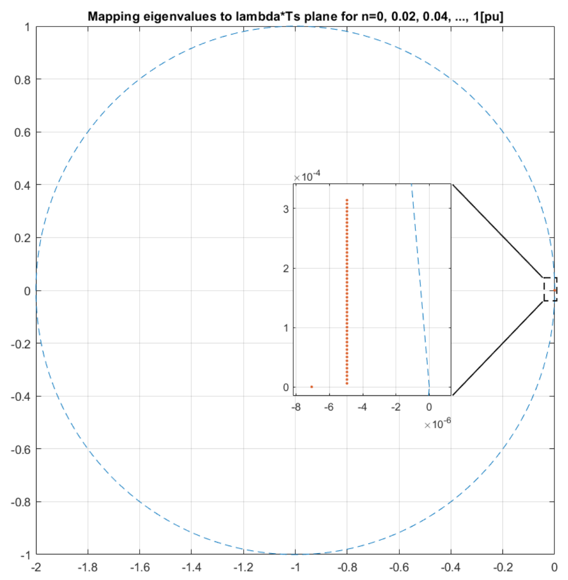
The damped frequency for the system is equal to the imaginary part of the eigenvalue in rad/s. For the rated motor speed, this is:

$$\omega_d = Im\{\lambda\} = 314.15 rad/s = 50Hz \tag{6.4}$$

In figure 6.2, the eigenvalues of the three-phase motor model are plotted for speeds between 0 and 1 pu, both in the s-plane (left) and λT_s-plane (right). Of the complex eigenvalue pairs, only the positive imaginary part is plotted.



(a) Continuous-time eigenvalues.



(b) Eigenvalues mapped to λT_{samp}-plane.

Figure 6.2: Eigenvalues of three-phase motor model for speeds between 0 and 1 pu, $T_{samp} = 1\mu s$.

From the plots, it is evident that the eigenvalues are within the stability limits for the discretized system. The imaginary component increases proportionally with the speed.

6.2.2 Six-Phase Model

dq Subspace

The eigenvalues for the dq subspace are found using the same expression as for the three-phase model, only with different parameters:

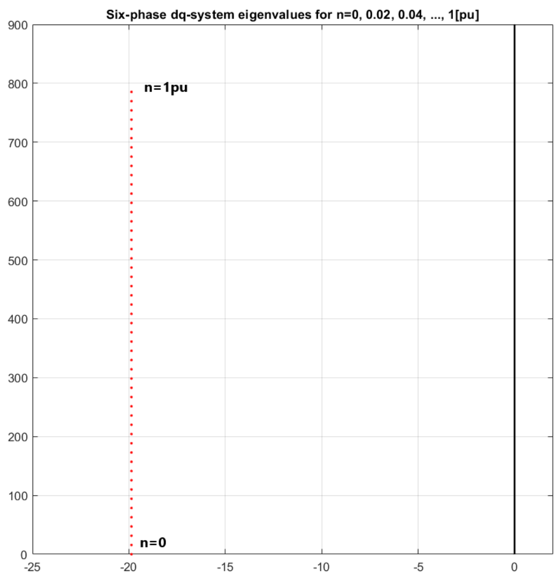
$$\begin{aligned}\lambda_{1,2}^{dq} &= \frac{-1}{2}\left(\frac{1}{50.3ms} + \frac{1}{50.3ms}\right) \pm \sqrt{\left(\frac{1}{2}\cdot 0\right)^2 - (2\pi\cdot 125Hz \cdot 1)^2} \\ &= -19.87 \pm j785.4\end{aligned} \tag{6.5}$$

Since the d-axis and q-axis reactance is equal for the six-phase motor, the only parameters affecting the imaginary value is ω_n and n. The relative damping ratio is:

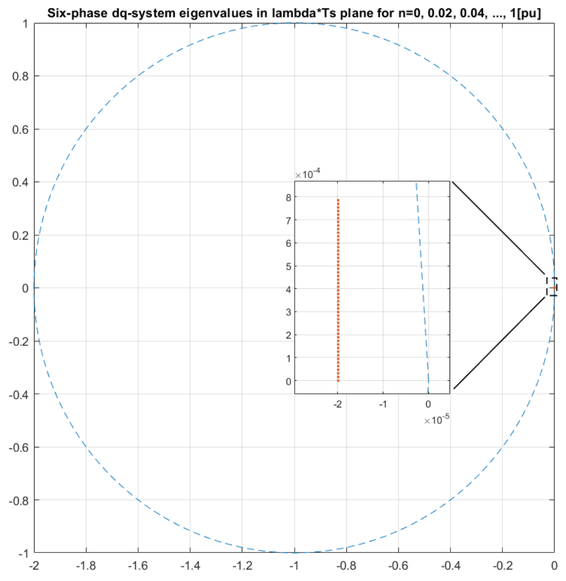
$$\zeta^{dq} = \frac{19.87}{\sqrt{19.87^2 + 785.4^2}} = 0.0253 \tag{6.6}$$

The damped frequency for rated motor speed is:

$$\omega_d^{dq} = 785.4 rad/s = 125Hz \tag{6.7}$$



(a) Continuous-time eigenvalues.



(b) Eigenvalues mapped to λT_{samp}-plane.

Figure 6.3: Six-phase model dq-system eigenvalues for speeds between 0 and 1 pu, $T_{samp} = 1\mu s$.

From figure 6.3b it can be seem that the eigenvalues map inside the unit circle in the λT_s-plane. The discrete system is therefore stable for the sample time used for speeds up to (and well beyond) rated motor speed.

z1z2 Subspace

From section 2.6.1, the eigenvalues for the $z1$-$z2$ subspace found as:

$$\lambda^z_{1,2} = \frac{-1}{T_\sigma} \pm \sqrt{\frac{1}{T^2_\sigma} - \left(\frac{1}{T^2_\sigma} + \omega^2_n n^2\right)} = \frac{-1}{T_\sigma} \pm j\omega_n n \tag{6.8}$$

Inserting the model parameters used in the simulation yields:

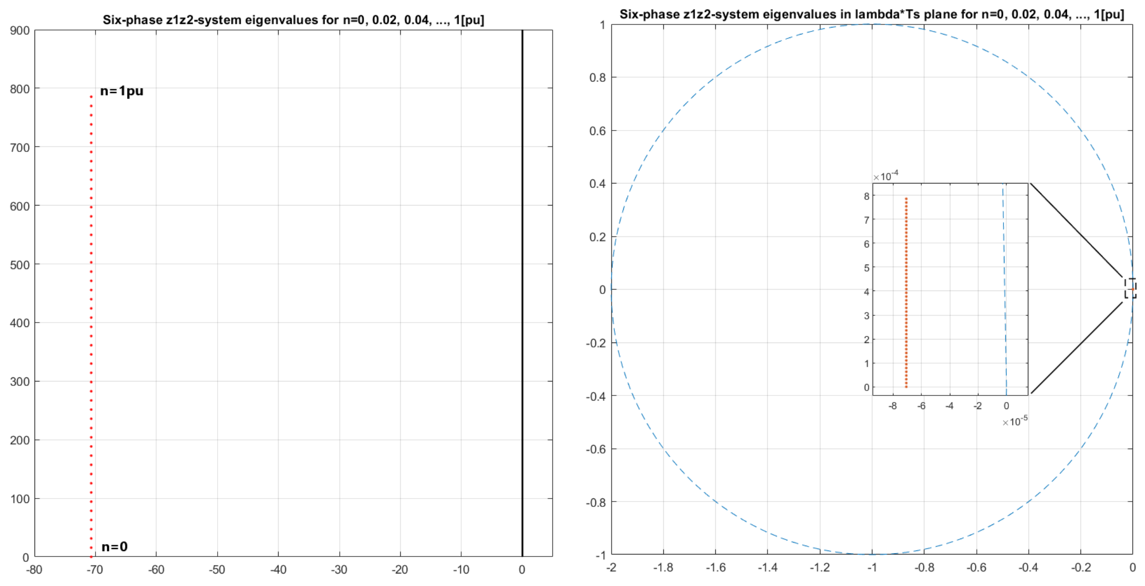
$$\lambda^z_{1,2} = \frac{-1}{14.15ms} \pm j(2\pi \cdot 125Hz \cdot 1) = -70.69 \pm j785.4 \tag{6.9}$$

The relative damping ratio is:

$$\zeta^z = \frac{70.69}{\sqrt{70.69^2 + 785.4^2}} = 0.0896 \tag{6.10}$$

The imaginary part of the eigenvalue is the same in the z_1-z_2 system as for the d-q system, therefore the damped frequencies at rated speed are also the same for both:

$$\omega^z_d = \omega^{dq}_d = 785.4rad/s = 125Hz \tag{6.11}$$



(a) Continuous-time eigenvalues. **(b)** Eigenvalues mapped to λT_{samp}-plane.

Figure 6.4: Six-phase model z1z2-system eigenvalues for speeds between 0 and 1 pu, $T_{samp} = 1us$.

From figure 6.4b, it is observed that the eigenvalues map inside the unit circle in the λT_s-plane, and the system is therefore stable for the default sample time used in the simulations.

6.2.3 Eigenvalue Analysis Conclusion

Both the three-phase motor and the six-phase model have fairly low damping ratios, which means that oscillations in the output may take a long time to dissipate. But the three-phase model has a lower damping ratio than the six-phase model and no oscillations are present in the electrical torque, so this is necessarily not a problem on its own. Also, the damping factor is the same for the continuous and the discrete system, so it does not explain why the oscillations are that much larger in amplitude in the SysGen model simulations.

The fact that the control system for the three-phase model only includes an inner loop current controller, whereas the six-phase control system also contains an outer loop speed controller may explain why the oscillations are only seen in the six-phase model.

Looking at the oscillation frequencies in the six-phase model at different speeds from table 5.3, they seem to fit well with the damped frequencies of the six-phase d-q and z_1-z_2 models. A comparison between the damped frequencies and the oscillation frequencies are shown in table 6.1.

Table 6.1: Comparison of the oscillation frequencies in the electrical torque from the six-phase SysGen model simulation to the damped frequency of the eigenvalues in the six-phase motor model.

Motor speed [pu]	0.6	0.78	0.86	0.98	1.016	1.004	1
Oscillation frequency [Hz]	80	94	114	123	130	121	131
Eigenvalue damped frequency [Hz]	75	97.5	107.5	122.5	127	125.5	125

Since the frequency of oscillations fit well with the damped frequency of the eigenvalues, this indicates that the oscillations are indeed related to excitation of the modes in the system. However, a more thorough analysis of the complete system, including the system controllers and PWM modulator is needed to draw further conclusions about the source of oscillations and how they are connected to the sample time and discretization of the motor model.

The six-phase model could also be run with the outer loop controllers disabled, to test if the torque oscillations are caused by some form of resonance between the discrete motor model and the control system. Perhaps the oscillations can be damped by tuning of the controller parameters, such as the PI controller gains.

Chapter 7

Conclusion

The focus of this master thesis has been to develop IP cores for use in a real-time emulator of a motor drive system on an FPGA. The emulator will be implemented on the NTNU control platform which is under development at the Department of Electric Power Engineering at NTNU. IP cores for simulation of three-phase and six-phase motors as well as mechanical load have been designed in Simulink using System Generator for DSP and tested by inserting the IP cores into existing Simulink models of motor drive systems.

The simulation results show that the System Generator models perform well compared to the continuous reference models. The three-phase SysGen model torque and speed outputs are very similar to the continuous reference model. There is some difference in the stationary speed between the SysGen and reference model, possibly caused by error in the numerical integration method used in the IP cores. The error is reduced when reducing the discrete sample time Ts, indicating that the difference is related to the inherent error of the numerical discretization method.

There is a fair amount of oscillations in the electrical motor torque of the six-phase SysGen model compared to the reference model. The sampling time has a clear effect on the amplitude of these oscillations; a higher sampling time causes larger oscillations. Eigenvalue analysis of motor the motor shows that motor model is stable, but the eigenvalues have a fairly low relative damping factor. The frequency of the oscillations at different speeds and the damped frequency of the eigenvalues at these speeds are quite similar, which indicates that the oscillations may indeed be related to the eigenvalues.

Changing the FPGA clock speed from 10ns to 100ns reduces the time it takes to run the Simulink SysGen models by a factor of around 10. The accuracy of the simulations do not seem to be particularly worse either. The speed error in the three-phase model is increased by around 1 rpm. In the six-phase model however, the motor torque oscillations are actually lower in amplitude in some places, compared to the 10ns simulation. It therefore seems that a clock speed of 100ns is sufficient for simulation of the IP cores.

Chapter 8

Further work

For further investigation into what causes the oscillations in the six-phase SysGen model, the model should be run with a simpler control system without outer loop speed controller, similarly to the three-phase model control system. Also, the three-phase SysGen model could be tested with added speed controller in the control system.

A more complete system analysis of the six-phase model, where the control system and mechanical load is included along with the motor could be performed in order to examine the stability of the model in further detail.

For further development of the emulator, the IP cores should be exported to Vivado and tested on the FPGA.

If the electrical torque oscillations observed in the six-phase SysGen model are caused by insufficiently fast sampling of inverter input voltage, then using a moving average filter to sample the input might be a good solution. This should be further investigated.

In this thesis, the system equations are discretized and implemented in the IP cores using the forward Euler method. This method is simple to implement and computationally efficient, but it is not as accurate or stable as other methods for the same sample time. The system can be discretized using other methods, such as second order Runge-Kutta, and compared to the forward Euler method to test if the added cost of computation is worth the increased accuracy.

Bibliography

[1] G. G. Parma and V. Dinavahi, 'Real-time digital hardware simulation of power electronics and drives', *IEEE Transactions on Power Delivery*, vol. 22, no. 2, pp. 1235–1246, 2007.

[2] V. Fjellanger, 'On-line voltage estimation during sensorless control of an induction machine drive', Master's thesis, Norwegian University of Science and Technology, Department of Electric Power Engineering, Jun. 2019.

[3] *Fpga fundamentals - national instruments*, https://www.ni.com/en-no/innovations/white-papers/08/fpga-fundamentals.html#section-99338513, Accessed: 23.06.2020.

[4] *Xilinx system generator tips and tricks – part 4: Understanding timing issues*, https://www.nutaq.com/blog/xilinx-system-generator-tips-and-tricks-%E2%80%93-part-4-understanding-timing-issues, Accessed: 03.06.2020.

[5] *Diagram of a synchronous digital circuit, xilinx system generator tips and tricks – part 4: Understanding timing issues*, https://www.nutaq.com/blog/xilinx-system-generator-tips-and-tricks-%E2%80%93-part-4-understanding-timing-issues, Accessed: 03.05.2020.

[6] R. Nilsen, *Electric Drives Compendium*, 5th ed. Department of Electric Power Engineering, NTNU, 2018.

[7] A. K. Ådnanes, 'High efficiency, high performance permanent magnet synchronous motor drives', Doktor Ingeniøravhandling, NTH, 1991.

[8] A. Schmitt, J. Richter, U. Jurkewitz and M. Braun, 'Fpga-based real-time simulation of nonlinear permanent magnet synchronous machines for power hardware-in-the-loop emulation systems', pp. 3763–3769, Feb. 2015. DOI: 10.1109/IECON.2014.7049060.

[9] R. H. Nelson and P. C. Krause, 'Induction machine analysis for arbitrary displacement between multiple winding sets', *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-93, no. 3, pp. 841–848, May 1974, ISSN: 0018-9510. DOI: 10.1109/TPAS.1974.293983.

[10] Ø. M. M. Hoem, 'Control of six-phase machines', Master's thesis, Norwegian University of Science and Technology, Department of Electric Power Engineering, Jun. 2010.

[11] K. J.R. and R. Nilsen, *Simulation of a 6 phase pm drive*, Wärtsilä Norway Ship Power Technology R&D, 2009.

[12] I. Zoric, M. Jones and E. Levi, 'Vector space decomposition algorithm for asymmetrical multiphase machines', in *2017 International Symposium on Power Electronics (Ee)*, Oct. 2017, pp. 1–6. DOI: `10.1109/PEE.2017.8171682`.

[13] R. Sureshkumar, *Numerical solution of ordinary differential equations, MIT 10.001 course notes*, `http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/lec24.html`, Accessed: 23.04.2020.

[14] J. Butcher, 'Numerical methods for differential equations and applications', *`http://www.math.auckland.ac.nz/Research/Reports/view.php?id=370`*, vol. 22, Dec. 1997.

[15] H. Podhaisky. (2010). Stability region for euler method. Acessed: 19.06.2020, [Online]. Available: `https://commons.wikimedia.org/wiki/File:Stability_region_for_Euler_method.svg`.

[16] R. Nilsen, *Lecture notes in electronics for power conversion*.

[17] K. Bjørvik and P. Hveem, *Reguleringsteknikk*, 3rd ed. Kybernetes forlag, 2014.

[18] *Signals and systems lecture 7 - discrete approximation of continuous-time systems*, `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-003-signals-and-systems-fall-2011/lecture-videos/MIT6_003F11_lec07.pdf`, Accessed: 10.06.2020, 2011.

[19] *Vivado design suite reference guide*, UG958 (v2019,1), Xilinx, May 2019.

[20] J. Volder, 'Binary computation algorithms for coordinate rotation and function generation', *Convair Report IAR-1*, vol. 148, 1956.

[21] R. Andraka, 'A survey of cordic algorithms for fpga based computers', *ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA*, Dec. 2001. DOI: `10.1145/275107.275139`.

[22] A. Karimova, '[smallsat] fpga implementation of pca dimensionality reduction technique', Master thesis, NTNU, 2018.

Appendix A

3-phase model block diagrams

The first figure in this appendix shows the connections between the IP cores in the 3-phase System Generator motor model in Simulink, whereas the following figures show the internal blocks and connections for each IP core.

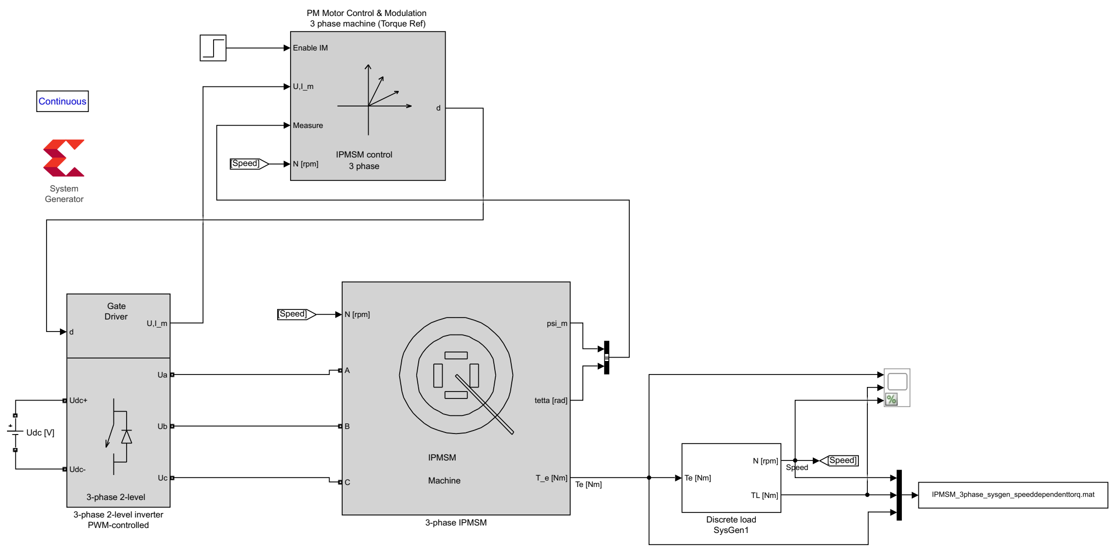


Figure A.1: Top level of three-phase SysGen model in Simulink

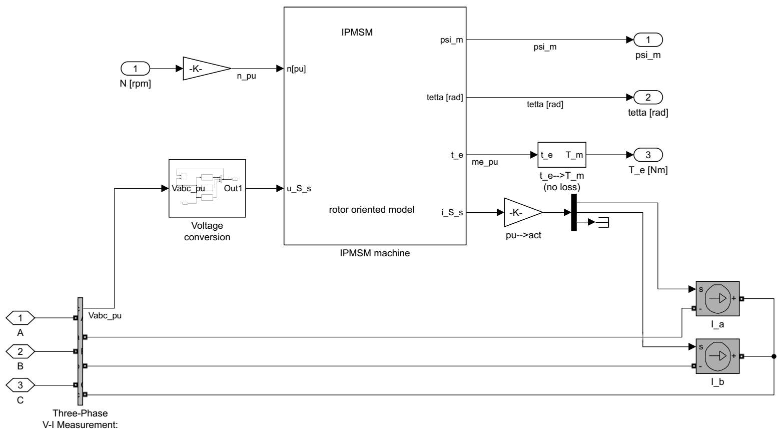


Figure A.2: Inside three-phase IPMSM mask shown in top level system figure A.1.

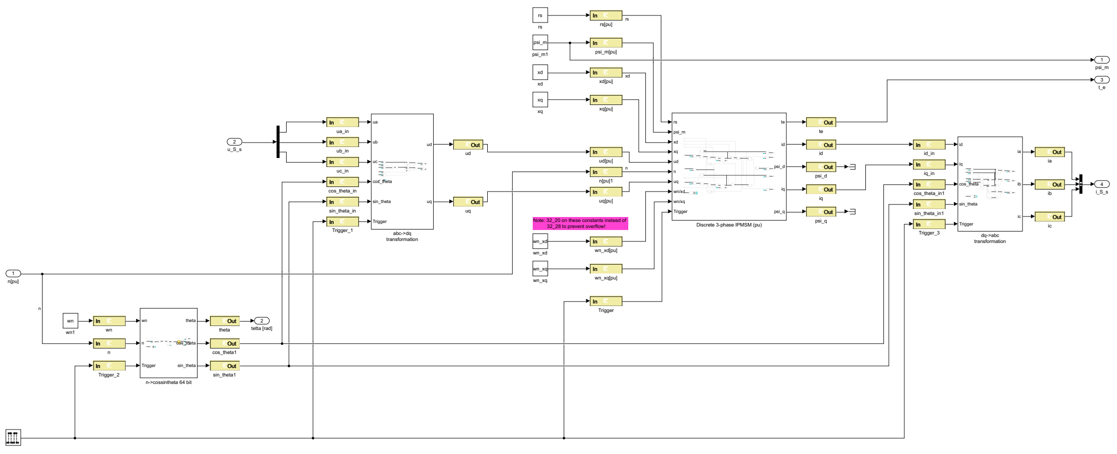

Discrete 3-phase IPMSM (pu)
abc->dq transformation
dq->abc transformation
n->cossintheta 64 bit
Note: 32_20 on these constants instead of 32_28 to prevent overflow!

Figure A.3: Inside IPMSM machine mask shown in figure A.2, showing connections between IP cores in three-phase motor.

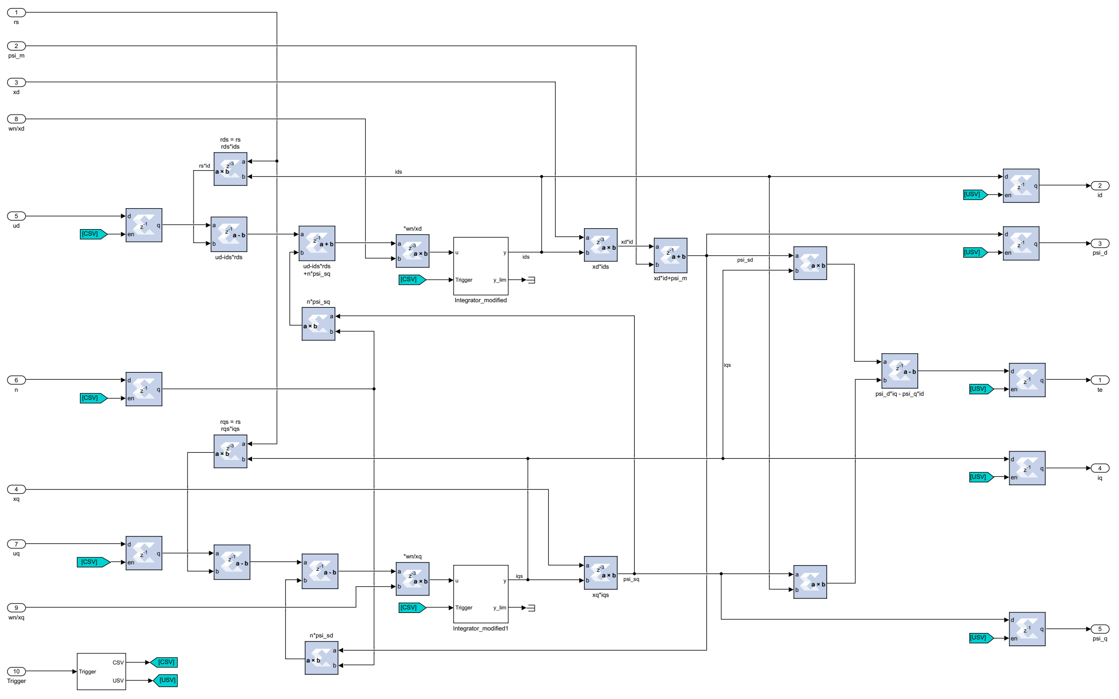


Figure A.4: Inside of 3-phase per unit motor IP core.

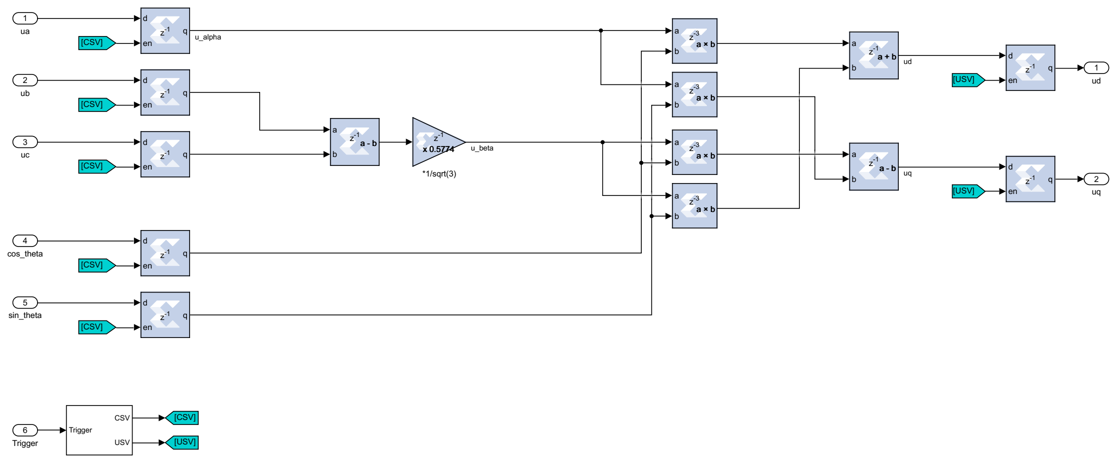


Figure A.5: Inside of abc->dq transformation IP core.

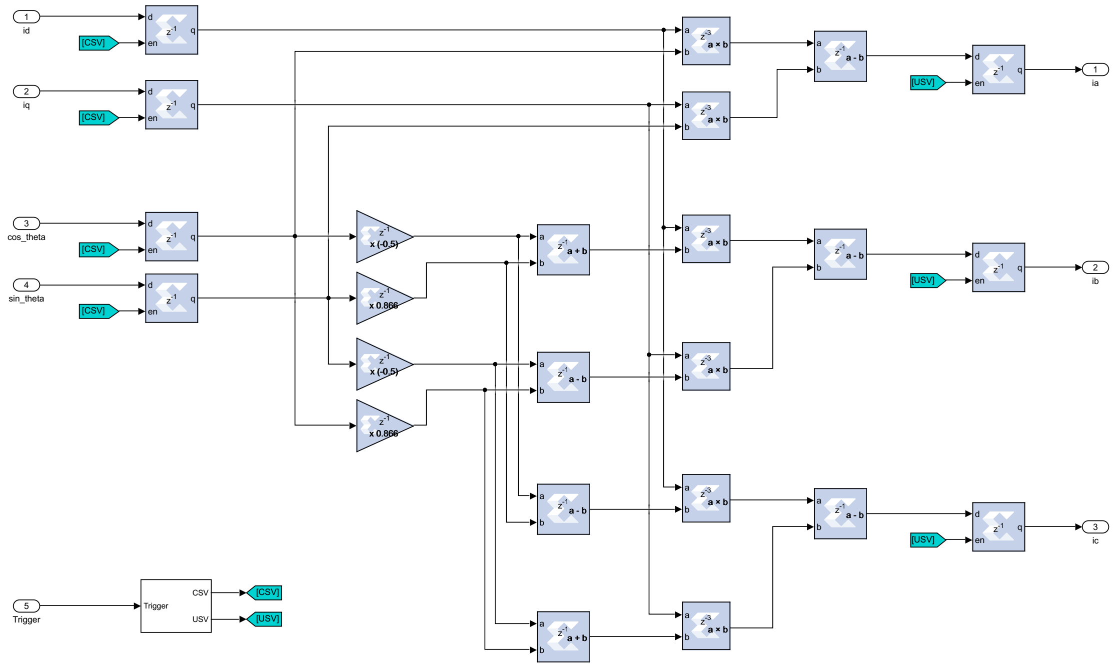


Figure A.6: Inside of dq->abc transformation IP core.

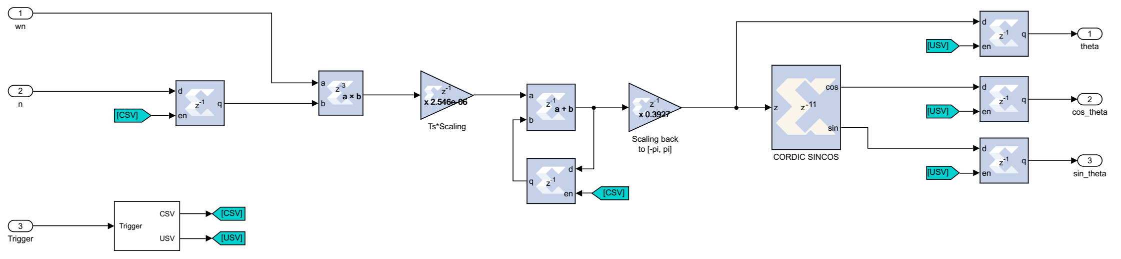


Figure A.7: Inside of n->cossintheta IP core, also used in 6-phase model.

Appendix B

6-phase model SysGen block diagrams

The first figure in this appendix shows the connections between the IP cores in the 6-phase System Generator motor model in Simulink, whereas the following figures show the internal blocks and connections for each IP core.

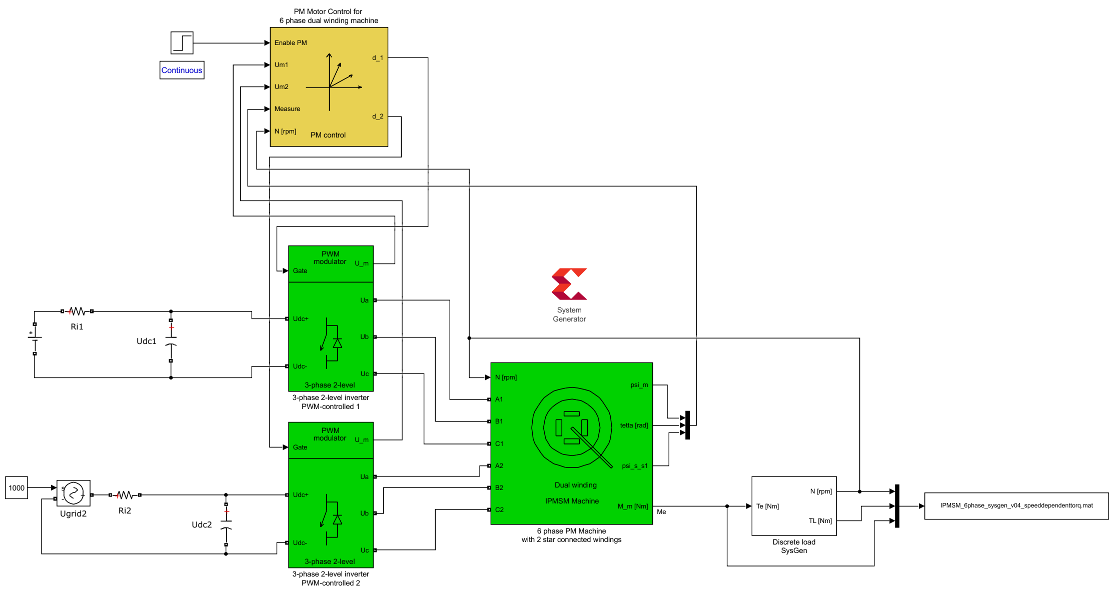


Figure B.1: Top level of six-phase SysGen model in Simulink

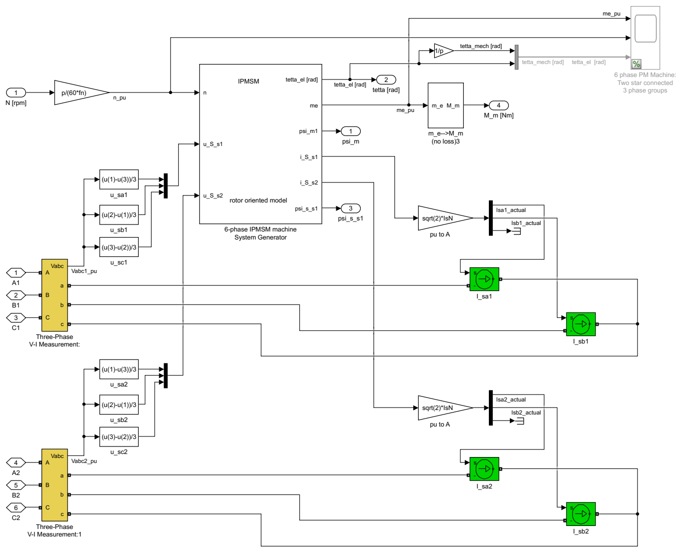


Figure B.2: Inside six-phase PM machine mask shown in top level system figure B.1.

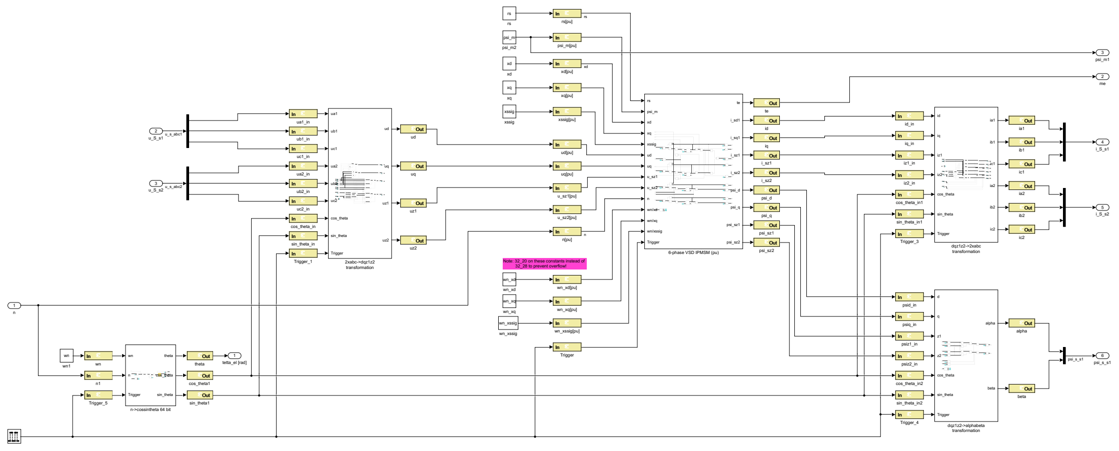


Figure B.3: Inside 6-phase IPMSM machine mask shown in figure B.2, showing connections between IP cores in six-phase motor.

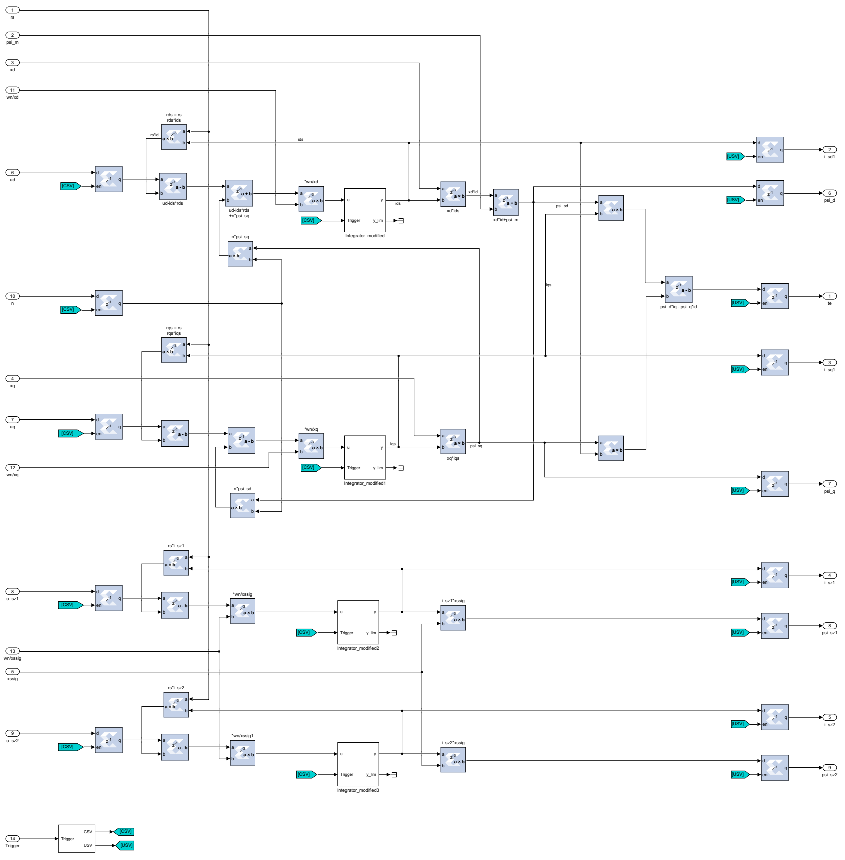


Figure B.4: Inside of 6-phase per unit motor IP core.

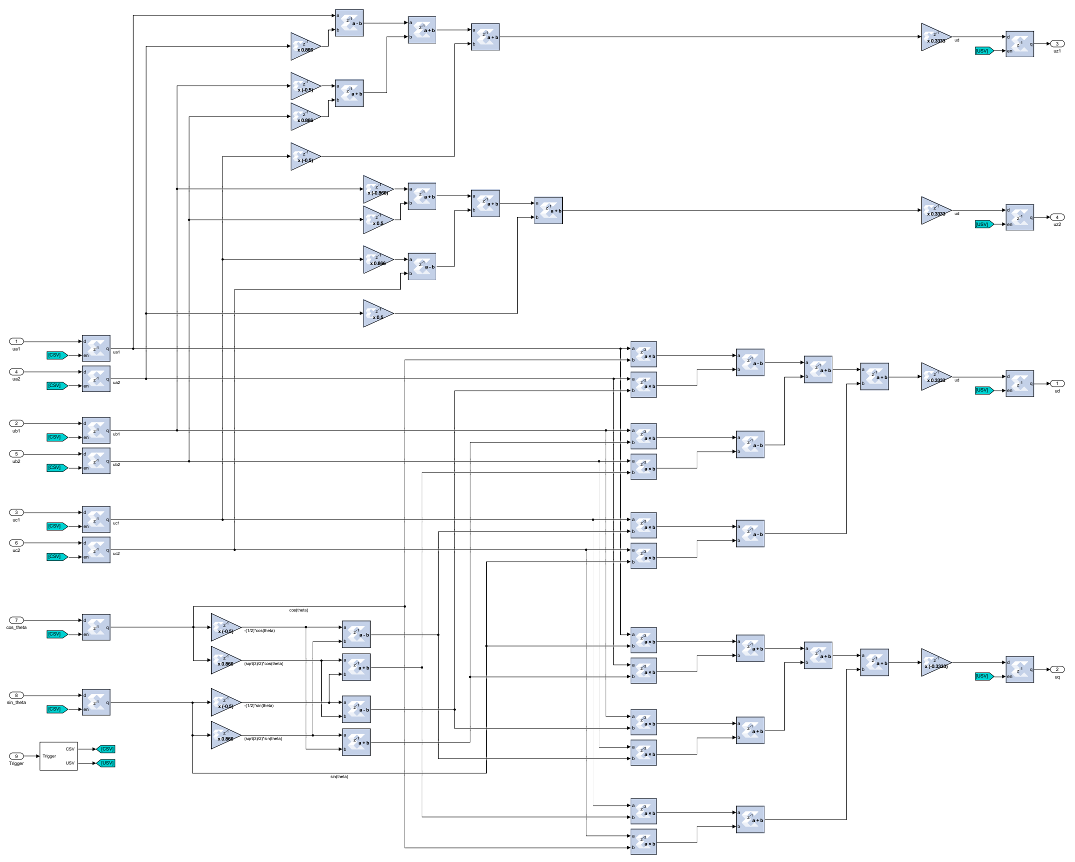


Figure B.5: Inside of 2xabc->dqz1z2 transformation IP core.

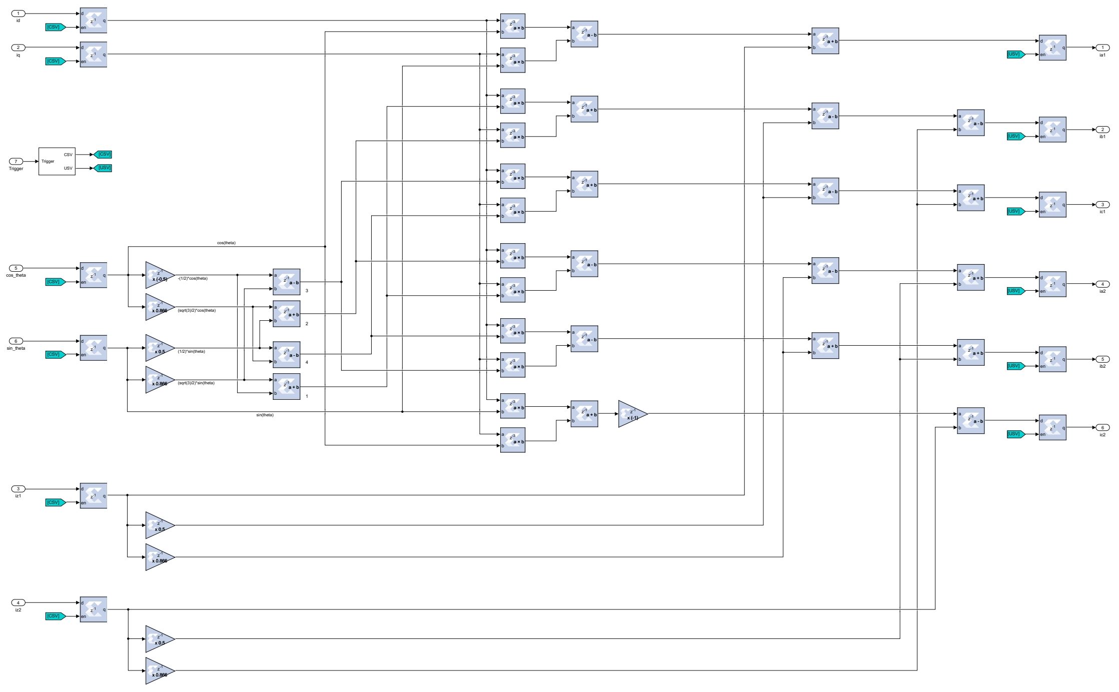


Figure B.6: Inside of dqz1z2->2xabc transformation IP core.

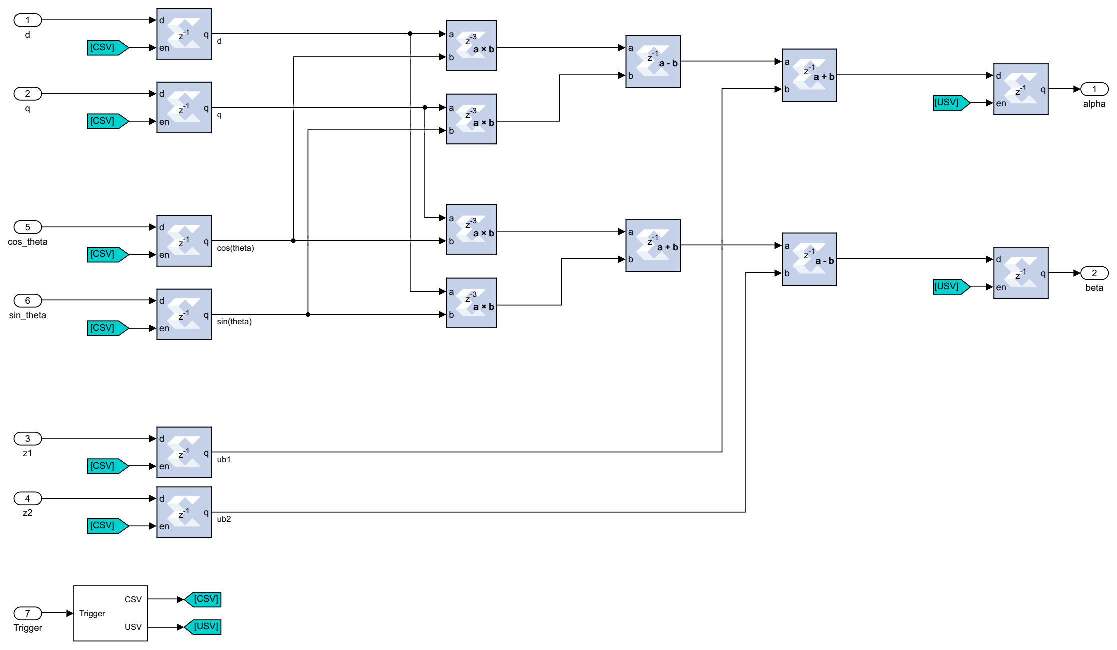


Figure B.7: Inside of dqz1z2->alphabeta transformation IP core.

Appendix C

Mechanical load SysGen block diagrams

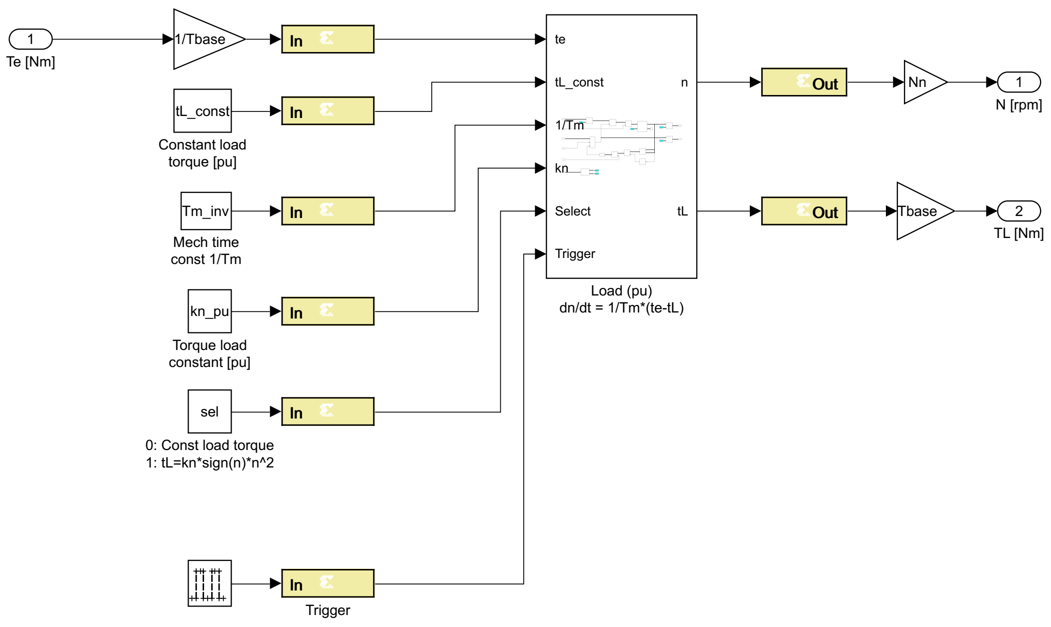


Figure C.1: Mechanical load IP core.

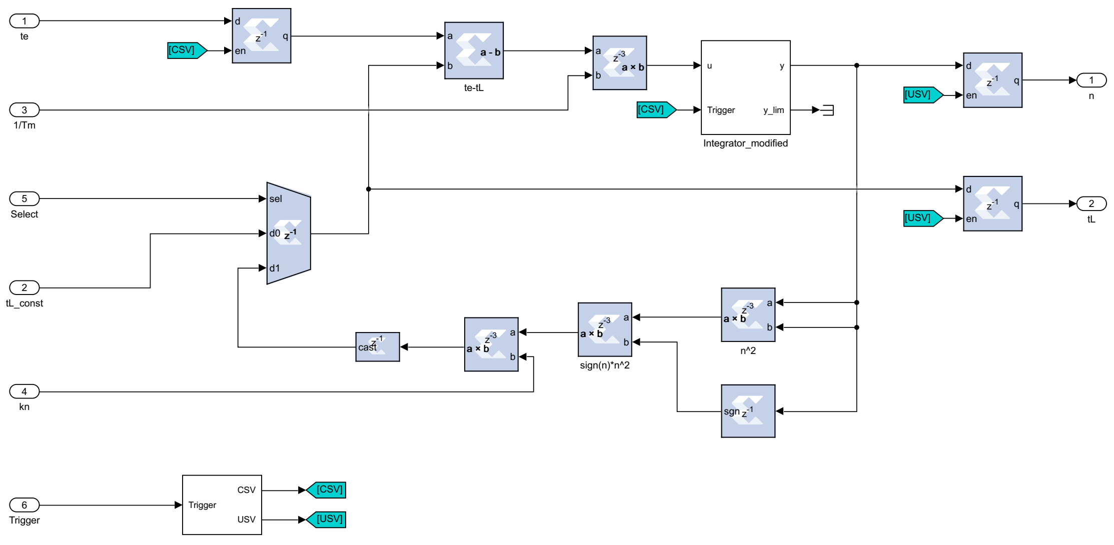


Figure C.2: Inside of mechanical load IP core.