VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍZENÍ BLDC MOTORU POMOCÍ LABVIEW FPGA
BLDC MOTOR CONTROL USING LABVIEW FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Sándor Ruhás

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Petr Blaha, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Sándor Ruhás ***ID:*** 125302
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Řízení BLDC motoru pomocí LabView FPGA

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s používanými principy řízení BLDC motorů, dále s grafickým programovacím prostředím LabView a jeho toolboxy a s platformou MyRIO.
2. Upravte stávající zařízení s BLDC motorem od firmy Honeywell tak, aby k ní bylo možné připojit řídicí platformu MyRIO.
3. Pomocí LabView FPGA vytvořte řídicí algoritmus pro řízení BLDC motoru s pomocí Hallových snímačů. - Nahraďte zpětnou vazbu z Halových snímačů algoritmem, který odhaduje úhel natočení rotoru (na základě znalosti modelu, nebo z měření BEMF napětí).
4. Otestujte vytvořený algoritmus na upraveném zařízení.

DOPORUČENÁ LITERATURA:

[1] Sul, S.K.: Control of Electric Machine Drive Systems. February 2011, Wiley-IEEE Press. ISBN: 978-0-4-0-59079-9.

[2] VLACH, J., HAVLÍČEK, J. a VLACH, M.: Začínáme s LabVIEW. BEN - technická literatura, 2008. ISBN: 978-80-7300-245-9.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 14.5.2018

Vedoucí práce: doc. Ing. Petr Blaha, Ph.D.
Konzultant: Ing. František Křivánek

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

Abstrakt

Tato diplomová práce je zaměřena na řídící algoritmus pro BLDC motor. Modely a algoritmy jsou naprogramovány v prostředí LabView s využitím platformy MyRIO od firmy National Instruments. Pro zpětnovazební řízení za pomocí Hallových sensorů je využito již existující zařízení výkonového budiče užitého v rotačním aktuátoru firmy Honeywell. Pro bezsenzorové řízení motoru zapomocí metody nepřímého snímání BEMF signálu je využito měření fázových proudů motoru. Časově kritická část algoritmu je naprogramována v FPGA, ostatní části kódu jsou naprogramovány v real-time modulu prostředí LabView.

Abstract

This thesis focuses on BLDC motor control algorithms with model based design approach. Models and control algorithms were programmed in LabView, the NI MyRIO was used as a hardware platform. For hall-sensor feedback controlled application an already finished power inverter was used from a Honeywell rotary actuator. For sensor-less motor control an indirect sensing of BEMF signal is applied using motor phase current measurement. The time-critical parts of the algorithms are programmed for FPGA, the non-time-critical parts are programmed for LabView Real-Time module.

Klíčové slová

Bezkartáčový DC motor, LabView, FPGA, NI MyRIO, NI MultiSim, Simulace

Keywords

Brushless DC motor, LabView, FPGA, NI MyRIO, NI MultiSim, Simulation

RUHÁS, S. Řízení BLDC motoru pomocí LabView FPGA. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 74 s. Vedoucí diplomové práce doc. Ing. Petr Blaha, Ph.D..

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Řízení BLDC motoru pomocí LabView FPGA jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Acknowledgments

I can not express enough gratitude to doc. Ing. Petr Blaha, Ph.D for all of his support and guidance throughout this project and for letting me work alone if I wanted to switch to "one-man-team" mode.

A special thanks for František Ráb for supporting my idea and letting me work on this thesis in tandem with my every day tasks in Honeywell. Without him I wouldn't been able to access the required hardware and software tools and also from time point of view it would have been impossible to achieve my goals.

As last, very big thanks for my family. They supported me in every imaginable way, even if I was trying sometimes to ignore them, because I was lost in my thoughts.

Preface

When the word actuator or BLDC motor appears during a conversation, the majority of us will almost immediately recognize the BLDC motor (that it is a motor of some kind), however, the word actuator might come as an unfamiliar one.
Few years back when I joined Honeywell officially, after being a technical support for more than 2 years in the same company but in a different team, I was introduced to actuators. Before that time I have had no experience in this field, nor I had overview what kind of actuators do exist. It didn't take much time for my homework to be done. I made a research what kind of actuators exist, in which fields there are used and how. A quick (and naiv) summary of my research could be formulated as follows: actuators are used basically to make things move, they are everywhere and they are coming in different sizes and different working principles.
Honeywell gave me an opportunity to extend both my mechanical and electrical expertise all together with my know-how from Brno University of Technology regarding to Control Theory. During the development of a low-torque (eg. rotary) actuator for HVAC applications I realized that the development team could use model based design technique for developing motor control algorithms. With this approach, not only the motor control algorithm can be verified but also different power inverter designs can be tested. With model based approach, the development time can be shortened, saving money and human resources- just because the fact that almost no hardware prototyping has to be done. Since in our team LabView was the most widely used tool for almost everything, it was clear that it should be used further on to develop and test the required algorithms and electronics. Also, National Instruments provides everything what is needed to get the process up and running.
This thesis is focused to BLDC motor control techniques in actuators, however, actuator types and applications will not be presented to the reader.

SÁNDOR RUHÁS
Brno, Czech Republic
January 2018

Contents

Preface . . . . . **vii**

List of Tables . . . . . **x**

List of Figures . . . . . **xi**

1 BLDC Motors **1**
1.1 Structure and operating principle [1] [2] 2
1.1.1 Basic Structure 2
1.1.2 Stator Cores 2
1.1.3 Windings 2
1.1.4 PM Rotor 3
1.2 Mathematical model of BLDC motor [1] [2] 5

2 Software Tools Overview **10**
2.1 National Instruments and LabView 10
2.1.1 Add-ons 10
2.1.2 Model Based Design in LabView 12
2.2 LabView and Multisim Co-Simulation 14

3 Realization **16**
3.1 CompactRIO platform and the NI Myrio 16
3.2 Current Measurement 22
3.3 LabVIEW FPGA and RT communication methods 30
3.3.1 FPGA inter-process communication 30
3.4 PID Control [3] [4] 32
3.4.1 CLOSED-LOOP CONTROL VERSUS OPEN-LOOP CONTROL . . . 32
3.4.2 PID loop in FPGA 35
3.5 RPM Measurement 37
3.6 Sensorless Operation 38
3.6.1 Method I 38
3.6.2 Method II 40

3.7 FPGA - 6-step commutation and PWM generation 46
3.8 PC Application 50

4 Conclusion 52

5 Attachments 54
5.1 Compile Time and Resource comparison 56

Bibliography 59

List of Tables

1.1 BLDC Motor parameters from datasheet 9

3.1 Compact RIO parameter comparison 20
3.2 Current measurement method comparison- advantages and disadvantages 23
3.3 Selected subset of parameters of NI9205 C-Series module 26
3.4 Selected subset of parameters for analog input channels on MyRIO 26
3.5 Data sharing methods comparison 31
3.6 G function at each mode 41
3.7 C function at each mode 43

List of Figures

1.1 Classification of the electric machine according to power source and operating principles . . . 1
1.2 Cross-sectional image of a BLDC motor. . . . 2
1.3 Schematic diagram of the BLDC motor. . . . 5
1.4 Equivalent circuit of the BLDC motor. . . . 7
1.5 Connecting NI MyRIO to PWA with power inverter. . . . 9

2.1 BLDC motor model in LabView . . . 12
2.2 DC motor model in LabView . . . 12
2.3 Simple PI controller in LabView . . . 13
2.4 Using Multisim design in LabView. . . . 14
2.5 Simple Inverter Schematic in MultiSim . . . 15
2.6 Setting BLDC motor parameters in MultiSim . . . 15

3.1 NI CRIO 9022 with C-Series modules in place . . . 17
3.2 NI MyRIO-1900 (left) and MyRIO-1950 (right) . . . 18
3.3 NI myRIO-1900 Hardware Block Diagram . . . 19
3.4 LabView Project Structure . . . 19
3.5 Current-measurement methods [5] . . . 22
3.6 INA 240 typical application [6] . . . 24
3.7 Prototype board with INA240 for 3-phase current measuring . . . 24
3.8 Analog input circuitry for (a) NI 9205 module (b) NI MyRIO . . . 25
3.9 Phase current measurement with no clipping- RT VI waveform . . . 26
3.10 Phase current measurement with no clipping- Oscilloscope screen . . . 27
3.11 Phase current measurement with clipping- Oscilloscope screen . . . 27
3.12 FPGA code for current measurement . . . 28
3.13 Butterworth configuration . . . 28
3.14 Filtered phase current waveform- RT VI waveform . . . 29
3.15 Phase current measurement with clipping demonstrated, no filter- RT VI waveform 29
3.16 Typical architecture using LabVIEW FPGA, LabVIEW Real-Time and PC Host [7] 30
3.17 FPGA code with PID controller inside . . . 36

3.18 PID configurable parameters 36
3.19 FPGA code for sampling hall sensors and for RPM calculation 37
3.20 Simulation of $G(\theta)$ function 39
3.21 FPGA realization of $G(\theta)$ calculation in a parallel loop 40
3.22 Calculated G-function values (lower waveform) and detected commutation instants (upper waveform) 41
3.23 Block diagram of BEMF observer [8] 42
3.24 BEMF observer block diagram programmed in LabView (1) 42
3.25 BEMF observer block diagram programmed in LabView (2) 43
3.26 Simulation of CF functions for sensor-less control using real BEMF signals from model 44
3.27 Simulation of CF functions for sensor-less control using observed BEMF signals from model 45
3.28 VI for PWM generation 46
3.29 Digital output circuitry for (a) NI 9205 module (b) NI MyRIO 47
3.30 Look-up table for six-step commutation 47
3.31 FPGA code for PWM signal generation 47
3.32 Switching states and conduction sequence according to the operating modes (a) Mode I. (b) Mode II. (c) Mode III. (d) Mode IV. (e) Mode V. (f) Mode VI. 48
3.33 New project from template- LabView sample projects 50
3.34 Front panel of the final application 50

5.1 Test bench which allows the motor to be loaded with a contact-less method. Also a torque sensor is mounted. 54
5.2 NI MyRIO with wired necessary connections. 55
5.3 NI cRIO9067 final resource utilization 58
5.4 NI MyRIO final resource utilization 58
5.5 NI cRIO9073 final resource utilization 58

One

BLDC Motors

An electric machine drive system usually consists of several parts such as driven mechanical system, electric machine, electric power converter, control system, and so on. The final criterion for the best design would be not only economic reasons such as initial investment, running cost, and so on, but also noneconomic reasons such as environmental friendliness, ethics, and regulations. [1] [2]

Through 100 years of development, the electric machines have diverse shapes, and a suitable shape is applied to the specific area according to the purpose of the machines. The machines can be classified as a rotary motion machine and a linear motion machine according to the motion of the rotor. Also, if the machine is classified according to the electric source and operating principles of the machine, it can be classified as shown in Fig. (1.1). [1]

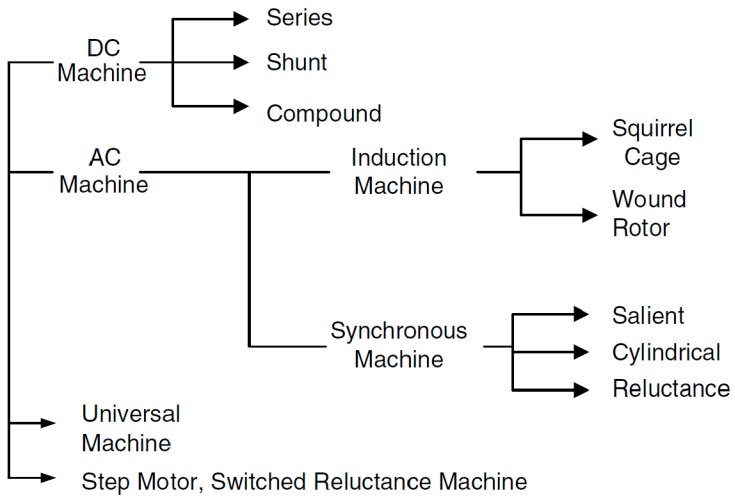


Figure 1.1: Classification of the electric machine according to power source and operating principles

1.1 Structure and operating principle [1] [2]

1.1.1 Basic Structure

The main design principle of a BLDC motor is to replace the mechanical commutator by using an electrical switch circuit. In traditional DC motors, the brushes are used for commutation, making the directions of the main magnetic field and the armature magnetic field perpendicular to each other when the motor is running. For the purpose of realizing commutation without mechanical contact, brushes were abandoned after the "inverted DC motor" was developed in which armature winding and magnet steel are placed on the stator and rotor sides separately. In order to control the motor's rotation speed and direction, a rotor-position sensor, a control circuit, together with a power inverter must be included in a BLDC motor system. The BLDC motor's structure contains a stator with armature winding and a rotor with a permanent magnet, which is similar to PMSM. The cross-sectional image of a four-pole BLDC motor is shown in 1.2.

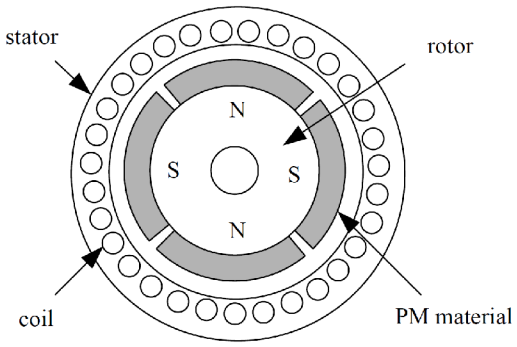


Figure 1.2: Cross-sectional image of a BLDC motor.

1.1.2 Stator Cores

The stator structure of the BLDC motor is similar to that of a general synchronous motor or an induction motor. Single- or multiple-phase symmetric windings are embedded in the iron core, which can be connected in "Y" or "D" type. Considering the performance and the cost of the system, the Y-type is mostly used, in which the three phase windings are connected symmetrically without a neutral point. Note that in the traditional brush DC motor, the armature winding is placed at the rotor, whereas the armature winding is installed at the stator side in the BLDC motor, causing less heating.

1.1.3 Windings

The common winding types used in BLDC motors are concentrated full-pitch windings, distributed full-pitch windings, distributed short-pitch windings, etc. The different types of windings

can affect the waveform of the back-EMF and the performance of the motor.

1. For the concentrated full-pitch winding, the wires of the same phase are placed in one cog, and therefore the air-gap flux density in the motor is the same. By adding the back-EMF generated by wires of each phase, we can get the waveform of the total back-EMF, which has a similar shape as the air-gap flux density. Furthermore, the platform width of the back-EMF waveform is the same as that of the air-gap flux density waveform. Thus, the concentrated full-pitch winding can produce a better trapezoidal back-EMF.

2. For the purpose of cooling the winding effectively through the inner surface space of the stator, the coil winding can be dispersed evenly at the surface of the stator, which is called distributed winding. Under normal circumstances, it is hard for the spatial distribution of air-gap flux density to form an ideal square wave.

3. On the other hand, application of the short-pitch winding makes it possible to shorten the connecting wires at the end of the winding. This can be helpful to save copper material and weaken the torque harmonics.

1.1.4 PM Rotor

The BLDC motor's rotor is constituted by permanent magnets with certain pole pairs embedded in the surface or the inside of the iron core. At present, the permanent magnets are usually made using rare-earth permanent magnetic materials like NdFeB, which have the advantages of high coercivity and remanence intensity. The permanent magnetic steels, in the BLDC motors as well as the brushed motors, are used to produce a sufficient magnetic field in the air gap. The only difference between them is that in BLDC motors, PM steels are installed on the rotor side, whereas they are placed on the stator side in brushed motors. Three typical structures of the BLDC motor rotors are as follows.

1. Surface-mounted PM rotor. For the surface-mounted PM rotor, on the surface of the iron core there is mounted radial magnetized tile-shaped rare-earth permanent magnet. Furthermore, the tile-shaped poles can be assembled by rectangle strips so as to cut the costs of the motor. In the design procedure of the motor, the designer always adopts this structure with its pole arc width larger than 120 degree electric angle in order to generate a square air-gap flux density and decrease torque ripple.

2. Magnet-embedded rotor. When the rectangular permanent magnets are embedded into the iron core of the rotor, we call it a magnet-embedded rotor. Since the magnetism gathering technology can provide larger flux, the flux under one polar pitch is produced by two adjacent poles in parallel. In this case, magnetism-isolating technology or a stainless steel shaft should be adopted.

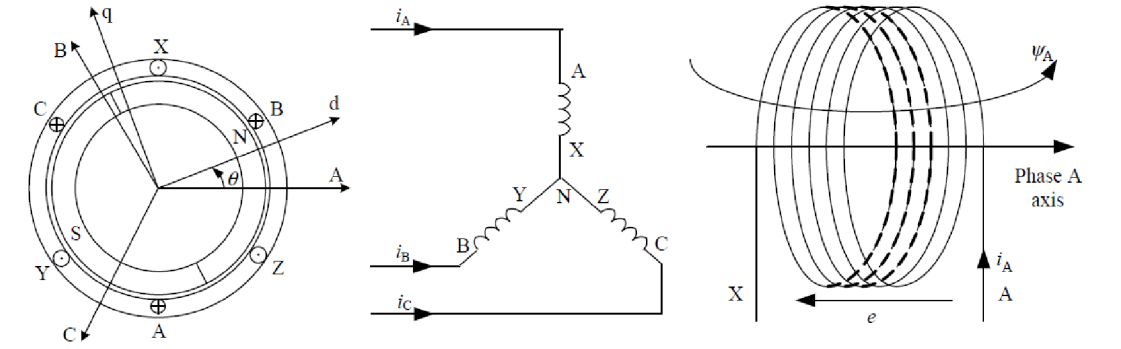
3. Magnetic loop rotor. For the magnetic loop rotor, a rare-earth PM ring magnetized radially in multiple poles through a special way is overlapped around the iron core. Note that it is usually used in low-power motors.

1.2 Mathematical model of BLDC motor [1] [2]

In this section, the differential equation model is built for a three-phase two-pole BLDC motor. The stator has a Y-connected concentrated full-pitch winding, and the inner rotor has a nonsalient pole structure. Three Hall sensors are placed symmetrically at 120°interval. The following assumptions are made to build the differential equation of the BLDC motor.

1. Ignore the core saturation, as well as the eddy current losses and the hysteresis losses.
2. Ignore the armature reaction, and the distribution of air-gap magnetic field is thought to be a trapezoidal wave with a flat-top width of 120°electrical angle.
3. Ignore the cogging effect and suppose the conductors are distributed continuously and evenly on the surface of the armature.
4. Power switches and flywheel diodes of the inverter circuit have ideal switch features.

With the assumptions above, a simplified schematic diagram of the motor is obtained which is shown in Figure 1.3



(a) Structure of BLDC motor (b) Connecting type of winding (c) Provision of positive direction (Phase A)

Figure 1.3: Schematic diagram of the BLDC motor.

The phase voltage of each winding can be expressed as:

$$u_x = R_x i_x + e_{\psi x} \tag{1.1}$$

where

u_x — phase voltage, in which subscript x denotes phase A, B and C;

i_x — phase current;

$e_{\psi x}$ — phase-induced EMF;

R_x —phase resistance. For three-phase symmetrical winding, there exists $R_A = R_B = R_C = R$.

The induced EMF can be written as:

$$e_{\psi x} = \frac{d\psi_x}{dt} \tag{1.2}$$

Then the flux for phase A for example will be:

$$\psi_A = L_A i_A + M_{AB} i_B + M_{AC} i_C + \psi_{pm}(\theta) \tag{1.3}$$

where
$\psi_{pm}(\theta)$ — PM flux linkage of phase A;
θ — position angle of rotor, the angle between rotor d-axis and the axis of phase A;
L_A — self-inductance of phase A;
M_{AB}, M_{AC} — mutual inductance of phase A with phase B and phase C.

When the rotor position is α, the PM flux of phase A is:

$$\psi_{pm}(\alpha) = N\phi_{pm}(\alpha) \tag{1.4}$$

$$\varphi_{pm}(\alpha) = \int_{-\frac{\pi}{2}+\alpha}^{\frac{\pi}{2}+\alpha} B(\theta) S d(\theta) \tag{1.5}$$

where
$\varphi_{pm}(\alpha)$ — PM flux of phase A when the rotor position angle is α;
$B(\theta)$ — PM rotor radial flux density in the air gap, which is in a trapezoidal distribution along θ;
N — turns on winding;
S — product of rotor radius and effective length of conductors.

Substituting equations (1.2), (1.3), (1.5) into equation (1.1) we get:

$$u_A = Ri_A + \tfrac{d}{dt}(L_A i_A + M_{AB} i_B + M_{AC} i_C) + e_A \tag{1.6}$$

where e_A represents the back-EMF of phase A.

The three-phase stator windings are symmetrical, the self-inductances will be equal, and so as the mutual inductance. Substituting $L_A = L_B = L_C, M_{AB} = M_{BA} = M_{BC} = M_{CB} = M_{AC} = M_{CA} = M$ into equation (1.6), we get:

$$u_A = Ri_A + L\tfrac{di_A}{dt} + M\tfrac{di_B}{dt} + M\tfrac{di_C}{dt} + e_A \tag{1.7}$$

e_A can be expressed as:

$$e_A = 2NS\omega B_m f_A(\theta) = \omega \psi_m f_A(\theta) \tag{1.8}$$

where
B_m - maximum value of PM density distribution in air gap;
ψ_m - maximum value of PM flux linkage of each winding, $\psi m = 2NSB_m$;
$f_A(\theta)$ - back-EMF waveform function of phase A.

The currents of the three phases satisfy:

$$i_A + i_B + i_C = 0 \tag{1.9}$$

Hence equation (1.7) can be further simplified:

$$u_A = Ri_A + (L - M)\frac{di_A}{dt} + e_A \tag{1.10}$$

Then the matrix form of phase voltage equation of BLDC motor can be expressed as:

$$\begin{bmatrix} u_A \\ u_B \\ u_C \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} L-M & 0 & 0 \\ 0 & L-M & 0 \\ 0 & 0 & L-M \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} e_A \\ e_B \\ e_C \end{bmatrix} \tag{1.11}$$

According to equation (1.11) the equivalent circuit of BLDC motor is shown in Figure 1.4

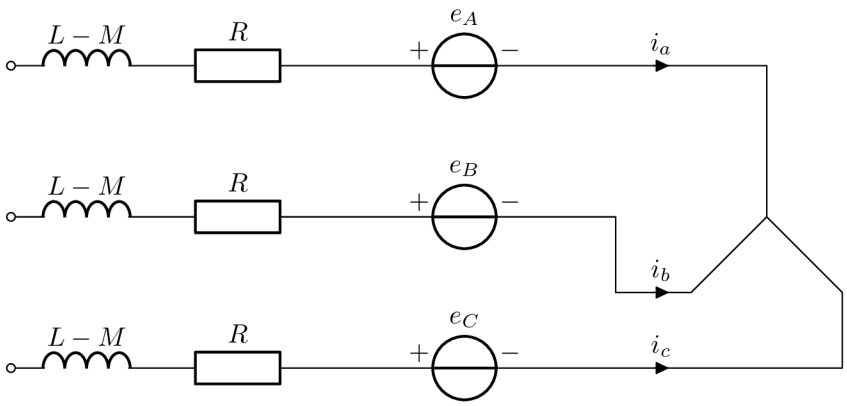


Figure 1.4: Equivalent circuit of the BLDC motor.

The line voltage equation can be obtained through subtraction calculation of the phase-voltage equation:

$$\begin{bmatrix} u_{AB} \\ u_{BC} \\ u_{CA} \end{bmatrix} = \begin{bmatrix} R & -R & 0 \\ 0 & R & -R \\ -R & 0 & R \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} L-M & M-L & 0 \\ 0 & L-M & M-L \\ M-L & 0 & L-M \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} + \begin{bmatrix} e_A - e_B \\ e_B - e_C \\ e_C - e_A \end{bmatrix} \tag{1.12}$$

For electromagnetic torque we have:

$$T_e = \frac{e_A i_A + e_B i_B + e_C i_C}{\Omega} \tag{1.13}$$

where
T_e - electromagnetic torque;
Ω - angular velocity of rotation.
At last, the motion equation to have the mathematical model complete:

$$T_e - T_L = J\frac{d\omega}{dt} + B_V \Omega \tag{1.14}$$

where
T_L - load torque;
J - rotor moment of inertia;
B_V - viscous friction coefficient.

Equations (1.11), (1.13) and (1.14) constitute the differential equation mathematical model of the BLDC motor.

The following table (1.1 contains the datasheet values of the used BLDC motor for this project.

Specification	**Value**	**Unit**
Number of phases	3	[-]
Number of poles, slots	12/9	[-]
Operating voltage	18	[V]
Operating output	1.8	[W]
Operating load	5.7	[mN.m]
Operating speed	3000	[r/min]
Rated current	225 $\pm$ 15 %	[mA]
Back EMF voltage	15 Min	[V_{pp}]
Rated load	5.7	[mN.m]
Phase to phase resistance	25.5 $\pm$ 15 %	[Ohm]
Phase to phase inductance at 1Khz	8.32 $\pm$ 30 %	[mH]
Torque constant	27.3 $\pm$ 15 %	[mN.m/A]
No load current	70 Max	[mA]
RPM constant	36.7	[rad/V.s]
Rotor inertia	6.0 x 10^{-7}	[kg.m^2]
Detent torque	0.6 Max	[mN.m]

Table 1.1: BLDC Motor parameters from datasheet

Figure 1.5 shows how the connection is realized to the inverter bridge. This connection approach was selected because it makes easier to swap the PWA if it is damaged by accident. Also during development the HW designers developed different versions of it, sometimes changing the test-point assignments in which case different pogo-pins had to be used.

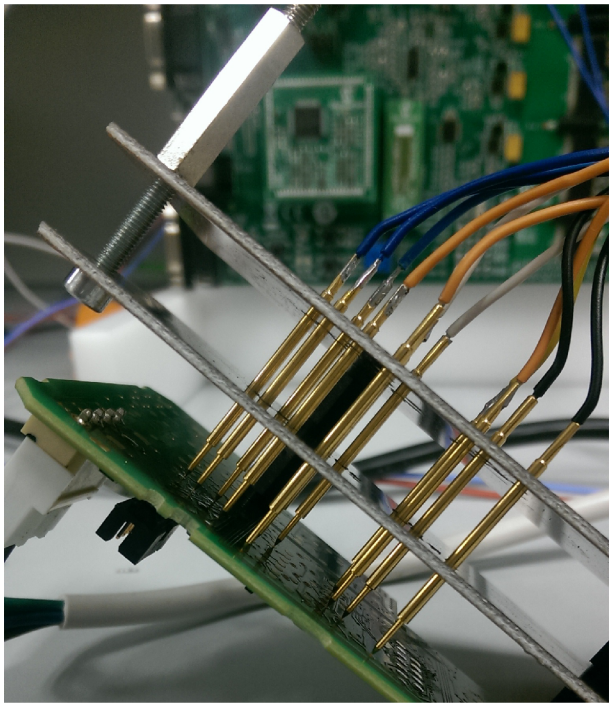


Figure 1.5: Connecting NI MyRIO to PWA with power inverter.

Two

Software Tools Overview

2.1 National Instruments and LabView

National Instruments was founded in 1976, since then, for more than 40 years NI has worked with and supported engineers and scientists to provide answers for the most challenging questions. They announced LabView system design software in 1986. Since that year a lot has been changed, not only in software point of view, but also in the supporting hardware. National Instruments has a wide hardware portfolio, which can provide flexible, in lots of cases off-the-shelf solutions. LabView is an easy to use graphical language, intended for engineers and scientists. This fact however can be deceiving, one could think that it is not capable of solving more complex tasks.

2.1.1 Add-ons

LabView in it's basic version does not contain advanced features which are needed for solving more complex task. Every development system has specific libraries or so called add-ons or toolkits. National Instruments gives an enormous set of modules and toolkits to support different applications in different fields in industry or science.

Labview add-ons are organized as follows [1]:

I Design

1. **LabView Control Design and Simulation module**
2. LabView MathScript RT module
3. LabView Statechart module
4. **LabView NI SoftMotion module**
5. LabView Digital Filter Design toolkit
6. **Electric Motor Simulation toolkit**
7. LabView Robotics module

II Deploy

1. LabView Application Builder
2. **LabView Real-Time Module**

[1] Modules used in this thesis are in **bold**

3. **LabView FPGA module /IP Builder/ Compile Farm**
4. LabView RIO Evaluation kit
5. LabView Wireless Sensor Network Module
6. LabView Touch Panel Module

III Interface

1. Vision Development module / Vision Acquisition module
2. LabView Datalogging and Supervisory Control module
3. LabView FPGA IEC 61131-3 Interface Utility
4. ECU measurement and Calibration toolkit
5. OPC servers
6. GPU Analysis toolkit

IV Integrate

1. LabView Report Generation Toolkit for Microsoft Office
2. Database Connectivity Toolkit
3. LabView DataFinder Toolkit
4. LabView Model Interface Toolkit

V Analyze

1. Sound and Vibration Toolkit
2. Advanced Signal Processing Toolkit
3. **Electrical Power Toolkit**
4. Multicore Analysis and Sparse Matrix Toolkit
5. Jitter Analysis toolkit
6. LabView Analytics and Machine Learning Toolkit
7. Biomedical Toolkit

VI Validate

1. VI Analyser toolkit
2. Desktop Execution Trace toolkit
3. Unit Test Framework toolkit
4. Requirements Gateway

Also an an addition to official National Instruments add-ond and toolkits there are lots of third-party tools, which are available in LabView Tools Network. [2]
Lots of support materials can be found at the NI Power Electronics Development Center group. [3]

[2]LabView Tools Network is available at: http://www.ni.com/labview-tools-network/

[3]NI Power Electronics Development Center: https://forums.ni.com/t5/Power-Electronics-Development/gp-p/grp-1891

2.1.2 Model Based Design in LabView

With the introduction of control design add-on tools for LabVIEW, National Instruments now delivers a single graphical environment for system identification, control design, simulation, and analysis. Without changing our code, we can download control algorithms and simulated systems to real-time hardware for rapid control prototyping or hardware-in-the-loop (HIL) simulation. Because LabVIEW is also a full-featured graphical programming language, we can use the same environment for creating custom functionality in our control applications. The control design and simulation module is shipped with lots of examples, including finished control-schemes for basic applications. One can for example easily find a model for DC motor and also for BLDC motor. (Figures 2.1- 2.2)

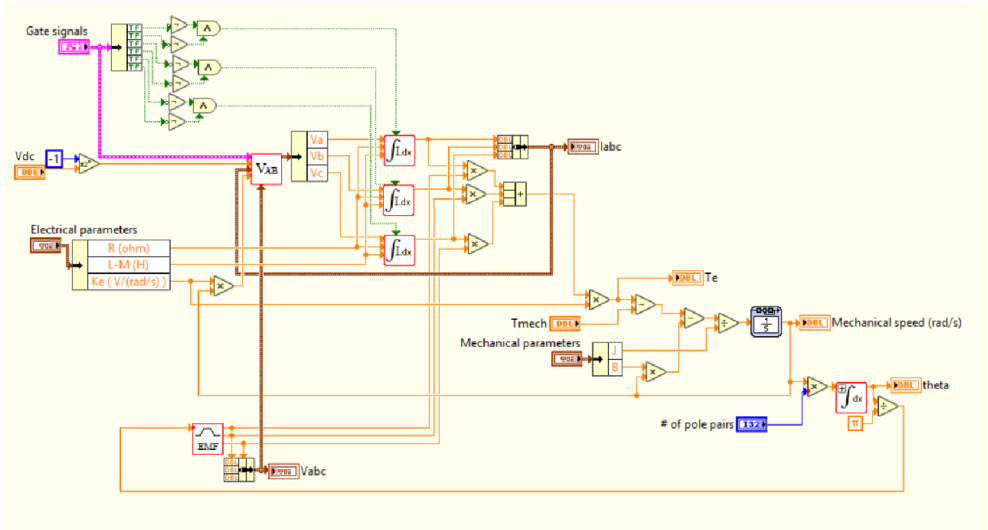


Figure 2.1: BLDC motor model in LabView

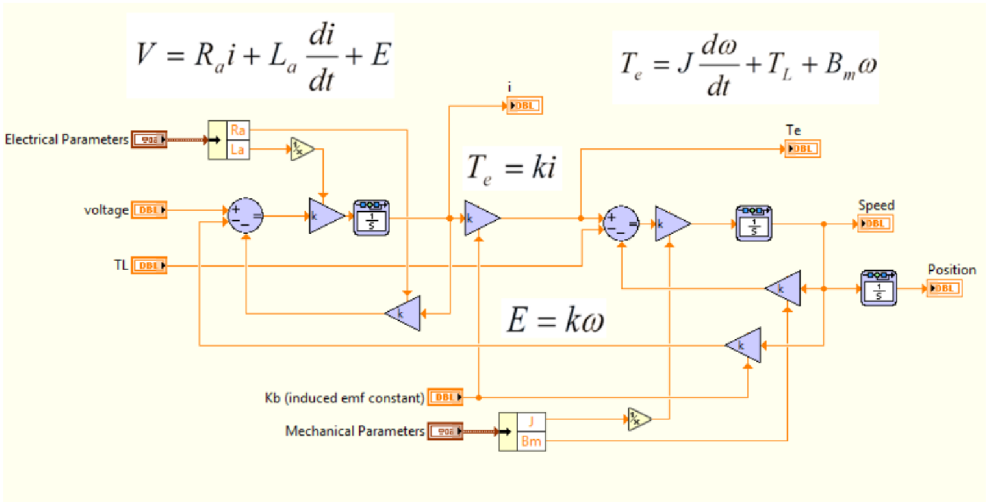


Figure 2.2: DC motor model in LabView

Also different controllers (PI, PID, Advanced PID) can be found in the library. Figure 2.3 shows a simple PI controller model in LabView. These models are usually used as subsystems, but if an update needs to be done for some specific reason, the updated model can be saved as a different subsystem.

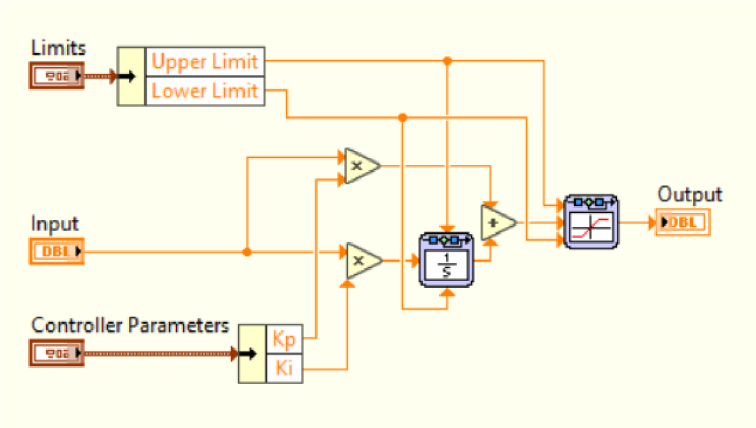


Figure 2.3: Simple PI controller in LabView

2.2 LabView and Multisim Co-Simulation

Multisim is the schematic capture and simulation application of National Instruments Circuit Design Suite, a suite of EDA (Electronics Design Automation) tools. Multisim is designed for schematic entry, simulation, and feeding to downstage steps, such as PCB layout. Multisim simulation and circuit design software gives engineers the advanced analysis and design capabilities to optimize performance, reduce design errors, and shorten time to prototype.
National Instruments made it possible to "connect" LabView with Multisim. This means that we can write FPGA code and simulate it against a high-fidelity SPICE simulation created in Multisim.
Figure 2.5 shows a simple inverter design in Multisim. The simulation of this inverter is used in the LabView program which is displayed on Figure 2.4.

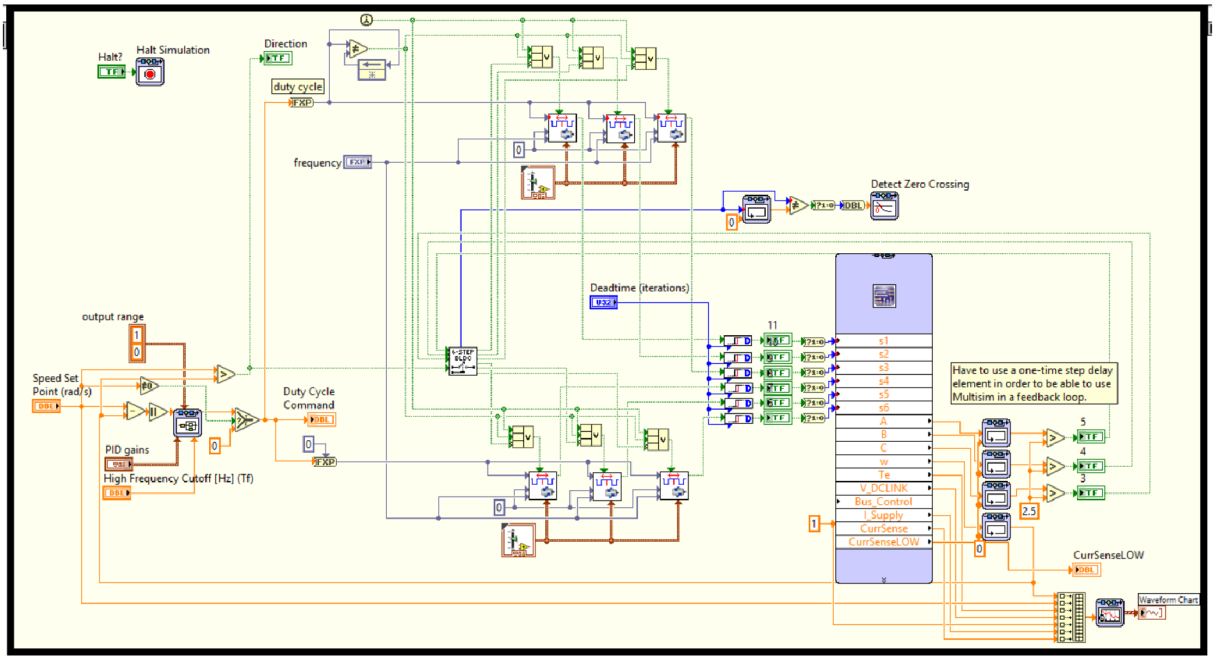


Figure 2.4: Using Multisim design in LabView.

The co-simulation scheme denoted on Figure 2.4 also contains a PID controller which uses feedback information (speed) from the inverter and connected BLDC motor. The control scheme incorporates 6-step commutation to the motor, with dead-times inserted for the transistors in the inverter to avoid short-circuit in the inverter legs.
Just to note, the BLDC motor model is used directly in the Multisim scheme all together with the power inverter itself. LabView also contains a BLDC model, that can be used too if necessary, however in Multisim the model is more detailed and additional parameters can be set. Figure 2.6 denotes electro-mechanical parameters for the BLDC motor model which can be configured in Multisim.

Figure 2.5 shows an inverter schematic design in MultiSim. This was used as the basis of the simulations realized later on in this document. Additional current clamps and connectors were added to fulfill every simulation needs (current clamps for phase current measurement and connectors to pass information to LabView about BEMF signal).

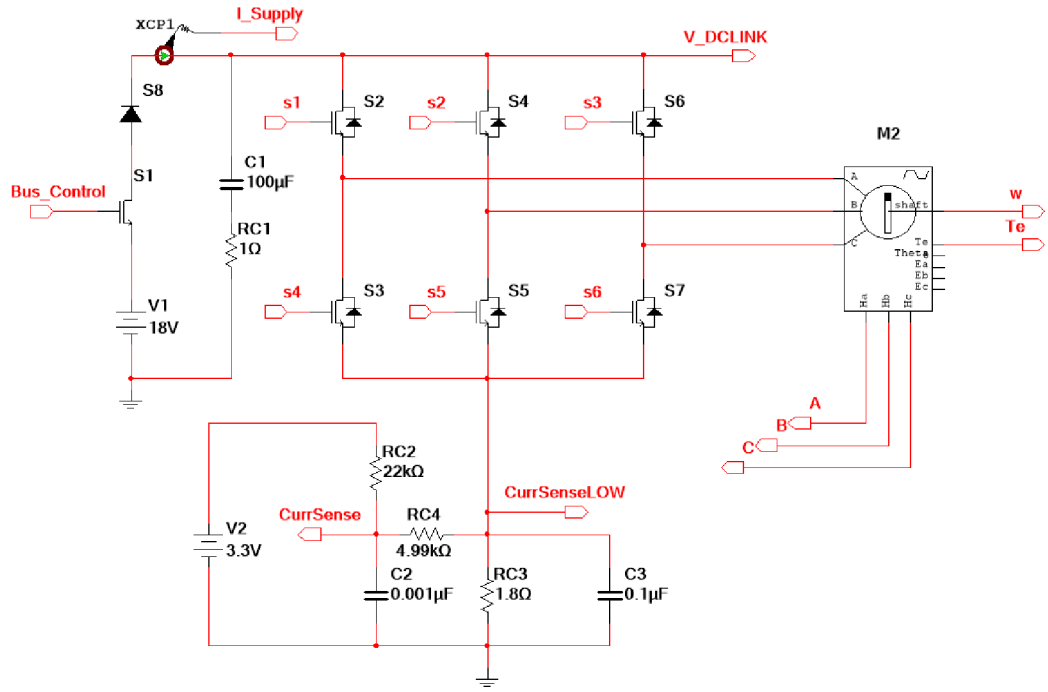


Figure 2.5: Simple Inverter Schematic in MultiSim

Figure 2.6 shows the parameters which can be configured in MultiSim for BLDC motor model. All the parameters are set according to our BLDC motors datasheet. (Table 1.1)

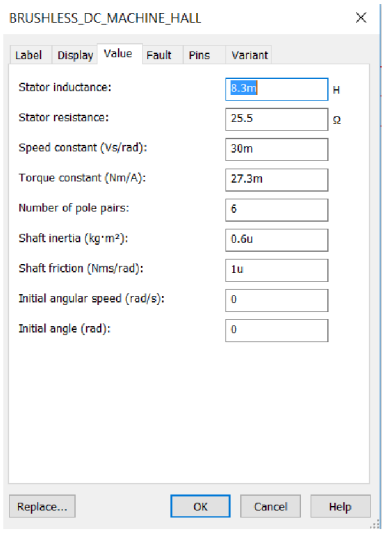


Figure 2.6: Setting BLDC motor parameters in MultiSim

Three

Realization

National Instruments offers a wide range of modular hardware that helps to build user-defined solutions. Their modular hardware portfolio can be arranged into 3 major groups:

1. CompactRIO platform - Used for embedded control.
2. CompactDAQ platform - Used for conditioned measurement of different units.
3. PXI platform - Used for High-Performance testing where precise timing and synchronization is needed.

3.1 CompactRIO platform and the NI Myrio

The CompactRIO platform features highly integrated software, a range of performance and form factor options, and extensive I/O to reduce risk, boost system performance, and simplify the design of advanced embedded control and monitoring systems. CompactRIO controllers offer the performance to execute advanced control algorithms with deterministic response times and low latency. They tale advantage of the latest advancements in processing and heterogeneous computing elements including ARM-based Xilinx Zynq SoCs as well as quad-core Intel Atom processors and Xilinx Kintex-7 FPGAs.

CompactRIO Application Areas:

1. Intelligent Systems for the Industrial Internet of Things
2. Industrial Machine Control
3. Power Electronics and Inverter Control
4. Condition Monitoring of Rotating Equipment
5. Power Quality Monitoring
6. Transportation and Heavy Equipment Monitoring
7. Robotics
8. Laser Control
9. Hydraulic Control

When starting development on CompactRIO platform, few considerations should be made in advance: One has to decide the timing reguirements of the application all together with available

physical space for the whole set-up. The first component to be selected is the **controller** itself. National Instruments offers these in 3 categories:

1. CompactRIO Controllers - High performance, extreme ruggedness, industry standard specifications.
2. CompactRIO Single-Board Controllers - Small, flexible embedded controllers with RTOS equipped with high-density connectors.
3. CompactRIO System on Module.

After an adequate controller has been selected, I/O modules can be choosen. I/O modules are coming in a standardized form factor, circuitry contains application dependent signal-conditioning to achieve the required signal acquisition or generation. The last major decision which has to be done is to pick the software support - again, this is application dependent. For example, if there is no need to do some high-speed signal and image processing or ultra-precise control, then the LabView FPGA module is not needed, hence the development cycle will be shorter, and will result in a smaller investment. National Instruments also offers the opportunity to program the real-time processor in C/C++ language or we can use standardly LabView.

Figure 3.1: NI CRIO 9022 with C-Series modules in place

Figure 3.2: NI MyRIO-1900 (left) and MyRIO-1950 (right)

On Figure 3.3 we can see the hardware block diagram of the NI MyRIO module. It is worth to note that the FPGA and RealTime modules are integrated into one chip. If it is used with factory default settings, no custom FPGA code needs to be loaded, we can do measurements almost immediately. The default FPGA personality includes also communication protocols such as I2C, SPI. Also we can use the UART module for serial communication. Of course, the default personality has it's limitations regarding to measurement throughput capabilities, but National Instruments offers a high-throughput personality too for the MyRIO.

To implement custom BLDC motor control with the MyRIO module, the FPGA code needs to be modified and re-compiled. When developing such an application in LabView, the programming needs to be done in 3 levels. There is an application for the PC itself, an additional code for the RT module and of course the code for the FPGA.

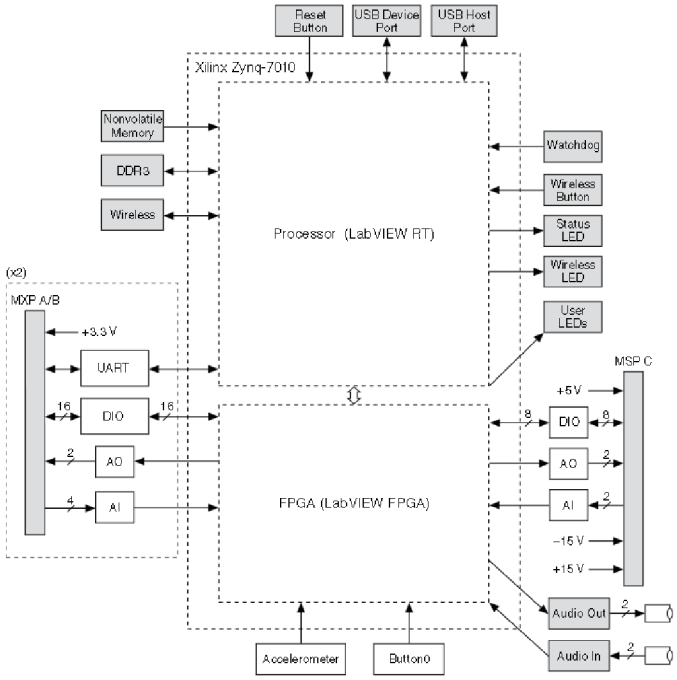


Figure 3.3: NI myRIO-1900 Hardware Block Diagram

Keeping this in mind, Figure 3.4 demonstrates how our LabView project is configured.

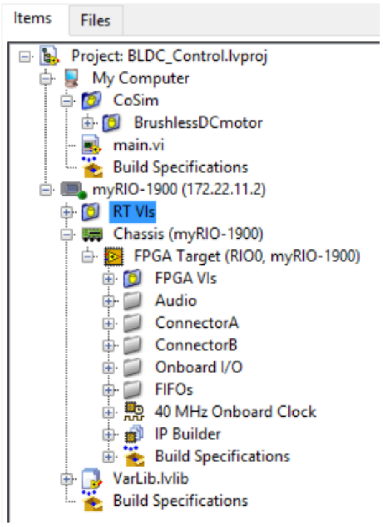


Figure 3.4: LabView Project Structure

At first glance one could say that the NI MyRIO platform is capable of everything and it has enough resources also for more complex tasks. While choosing the hardware (NI MyRIO) for

this project, this mistake was made. After a short time it was realized (and also it was necessary) that a hardware change is needed to allow for interrupt-free coding and experimentation. The initial use of NI MyRIO has been changed for the cRIO 9067 which offers much more resources. Usually during development cycles this approach is followed that a more powerful hardware is selected at the beginning, and after everything is developed and verified, the whole work (result, product) is ported back to the less-powerful hardware. According to this approach, it was always kept in mind that the application will be ported back to NI MyRIO in the future. Because of this, in next parts of this document few compromises will be made which were necessary to allow the porting of developed code between cRIO modules with minimum modifications.
Table 3.1 shows a comparison between NI MyRIO and other cRIOs used in this project.

	MyRIO	**cRIO 9067**	**cRIO 9073**
FPGA type	Xilinx Z-7010 XC7Z010	Xilinx Zynq-7000 XC7Z020	Xilinx Spartan-3 2M XC3S2000
Architecture	ARM Cortex-A9	ARM Cortex-A9	
Nonvolatile memory	512 MB	1 GB	128 MB
Volatile memory (DRAM)	256 MB	512 MB	64 MB
Processor speed	667 Mhz	667 MHz	266 MHz
Processor cores	2	2	1
Number of flip-flops	35,200	106,400	40,960
Number of 6-input LUTs	17,600	53,200	-
Number of 4-input LUTs	-	-	40,960
Number of multipliers	-	-	40
Number of DSP48s	80	220	-
Total block RAM - # of 36Kb blocks	60	140	40
Number of DMA channels	16	16	3
Number of slots	-	8	8

Table 3.1: Compact RIO parameter comparison

While creating the test bench and developing the required algorithms, all of the hardware mentioned in table 3.1 were tried out. The reason for this was that MyRIO just run out of resources, and there was no time to do code optimization. After abandoning for the time being the MyRIO, cRIO 9067 was selected to continue on. This meant that the whole set-up had to be rewired and also a minor code change had to be done. From table 3.1 it is clear that this cRIO incorporates the most resources, hence there was no resource issue afterwards. After upgrading to this cRIO, Murphys law kicked in, and the chassy was needed for a different project, hence again there was a need to move the application to a different cRIO which was available. For this (hopefully) last porting, the cRIO 9073 was the target. Regarding to this chassy, it is worth mentioning that this is a legacy device, with few resources and a completely different FPGA chip. After continuing the development on this chassis, few limitations were experienced. First of all, it is quite tricky to install the required software packages to its RT module and deploy the settings (small nonvolatile memory VS. big SW packages) without causing the RT processor to fail to operate. Secondly, trying to compile the same code that was compiled to cRIO 9067 resulted in timing violation (code tried to use up more multipliers than it was available). Successful compilation was achieved after trimming the code a little bit. For comparison the following figures can be studied in the attachments: Figure 5.3, figure 5.4, figure 5.5. It is worth noticing that the same code - in our case the PWM generation parallel loop - uses up more than 80 % of

the resources on MyRIO. Keeping this fact in mind, the code will have to be optimized because with the current structure the PID controller won't compile together with the PWM generation on MyRIO due to small amount of resources. The compile times are also compared for these 3 targets - same VI is used to do the comparison.

3.2 Current Measurement

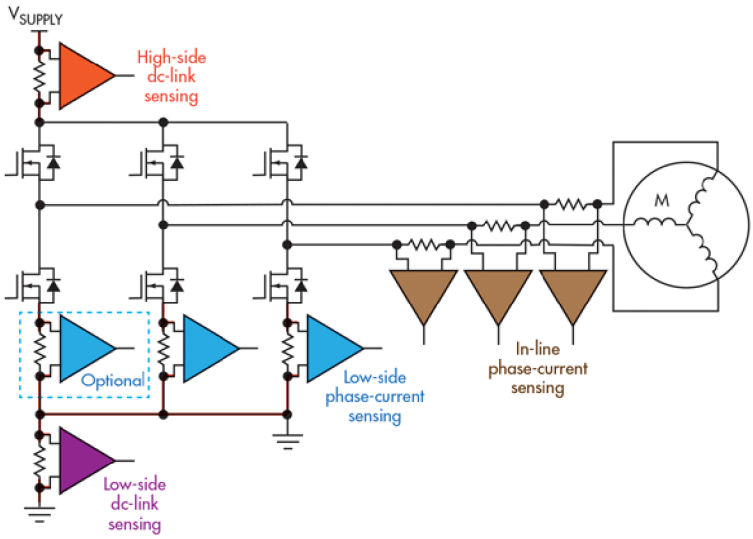


Figure 3.5: Current-measurement methods [5]

High-side dc-link sensing is typically used only for fault detection. It has the advantage of having a stable common-mode voltage and enables motor fault detection. However, depending on the motor, the common-mode voltage could be very high, limiting the choice of devices able to be used in this implementation. In addition, the driver current, which is actually what's being measured, doesn't necessarily equal motor phase current.

Low-side dc-link sensing is also typically used only for fault detection. It has the advantage of having a common-mode voltage that's essentially 0 V, broadening the range of available solutions. However, it doesn't allow for motor fault detection. Furthermore, the driver current, which is actually what's being measured, doesn't necessarily equal the motor phase current. Determining the phase current at this location requires very high-speed, high-slew-rate amplifiers and fairly complex algorithms in the controller.

Low-side phase sensing allows for easier determination of the motor phase currents, but it's not an exact equivalent. Therefore, an error is potentially introduced relative to the true phase current. Low-side phase sensing also introduces a ground variation of the motor relative to system ground. Due to the location of the sense element, fault detection is limited in this implementation. It does offer the advantage of having more options for implementation, as the common-mode voltage is essentially ground, which enables the use of low-voltage amplifiers. However, due to the nature of the current through the drivers, a high-slew-rate amplifier is again required to respond to the dynamic nature of the current being monitored in each leg. In many cases, only two of the phases are measured, with the third phase calculated in the controller.

In-line phase sensing offers true motor phase-current measurement for optimizing the quality of the information being provided to the motor-control algorithm. The major challenge is

that the common-mode voltage is a pulse-width-modulated (PWM) signal, which causes a disruption of the output signal unless good PWM rejection circuitry is enabled. This leads to more strenuous requirements for the current-sense amplifier, which must have both very good dc and ac common-mode rejection ratio (CMRR).[5]

In-line Phase Current Measurement

When measuring in-line, there's no guess work on the phase current. However, the common-mode voltage seen by the current-sense amplifier is a high-voltage PWM that must be rejected. The frequency of the signal seen by the current-sense amplifier has two contributors:

1. Differential signal (useful information) is relatively narrowband and small amplitude.
2. Common-mode PWM signal (not useful) is wideband and large amplitude.

An ideal in-line current-sense amplifier would only process the differential signal, while rejecting the common-mode signal. This high voltage combined with high $\Delta V/\Delta T$ poses a steep challenge that limits the availability of suitable current-sense amplifiers. This tends to limit the adoption of this topology to only those applications that require precise phase-current measurement, such as that for electronic power-steering systems.

Table 3.2 compares resistor based motor current sensing techniques:

	Low Side	High Side	In-Line
Advantages	Low common mode voltage Low voltage Amp possible	Stable Common mode voltage Fault detection	True motor phase current
Disadvantages	Ground variation Unable to detect fault Driver current does not necessarily equal to motor phase current	Stable but high Vcm Driver current does not necessarily equal to motor phase current	PWM common mode voltage Sensing amp must have good DC and AC CMRR

Table 3.2: Current measurement method comparison- advantages and disadvantages

The in-line phase current measurement was selected to continue with, as the integrated circuit the INA240 is used from Texas Instruments. The typical application of this current-sense amplifier can be seen on figure 3.6.

It is capable of handling common-mode voltages as high as $80\ V$, has advanced PWM rejection function implemented, and comes with different available gains. Because we have a relatively small BLDC motor to work with, the INA240A1 is used which has the lowest gain: $20\ V/V$. This means that a change of $1\ V$ on the selected shunt-resistor will cause a $20V$ change in its output signal. Considering the input stages of the equipments and parts which are at out disposal, for each phase 2 pieces of $1.5\ \Omega$ resistors are connected in parallel. The reference voltage to the current-sense amplifier is $5\ V/2 = 2.5\ VDC$. This means that it will be able to handle bi-directional current measurement. For one direction of current-flow the output signal from the amplifier will rise from $2.5\ VDC$ to the maximum of $5\ VDC$, for the opposite current flow it will decrease from $2.5\ VDC$ to ground reference. When the given motor is considered, it is obvious that without decreasing the value of the shunt-resistor we will be not able to measure the full scale current when the motor operates under its maximum load. At $I_m = 200\ mA$ the

The op amp common-mode rejection ratio (CMRR) is the ratio of the common-mode gain to differential-mode gain.

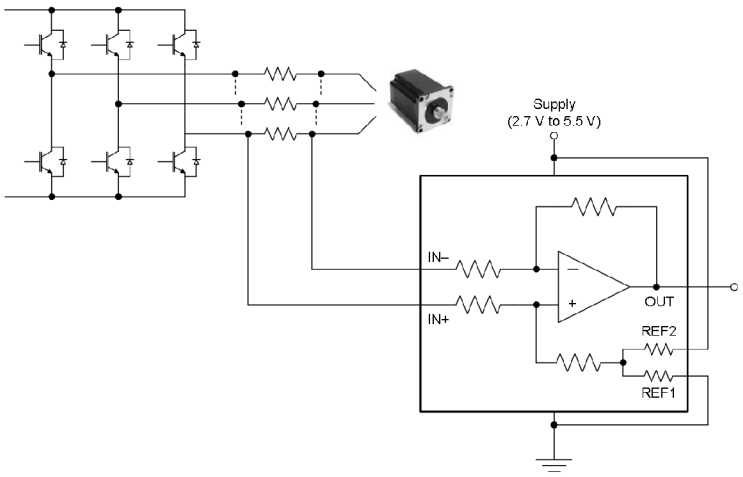


Figure 3.6: INA 240 typical application [6]

output voltage of the amplifier would have to change $3\ Volts$ in one direction. This situation will cause clipping. This is demonstrated on figure 3.15. The current measurement should look like as it is on figure 3.9.

Figure 3.7 shows the quick prototype that has been made to verify the functionality of the selected current-sense amplifier.

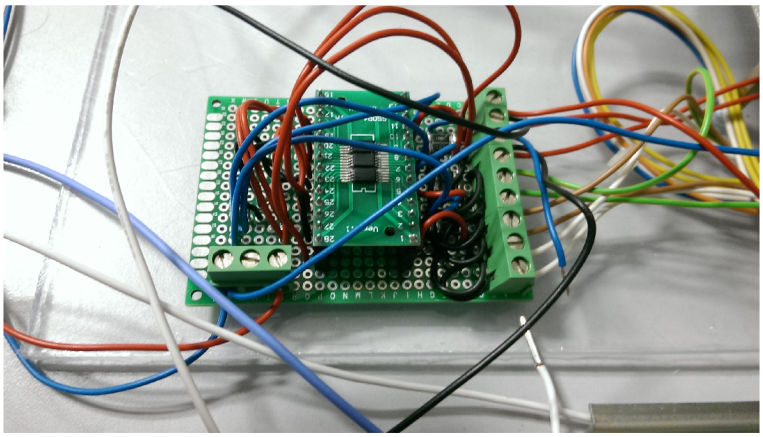
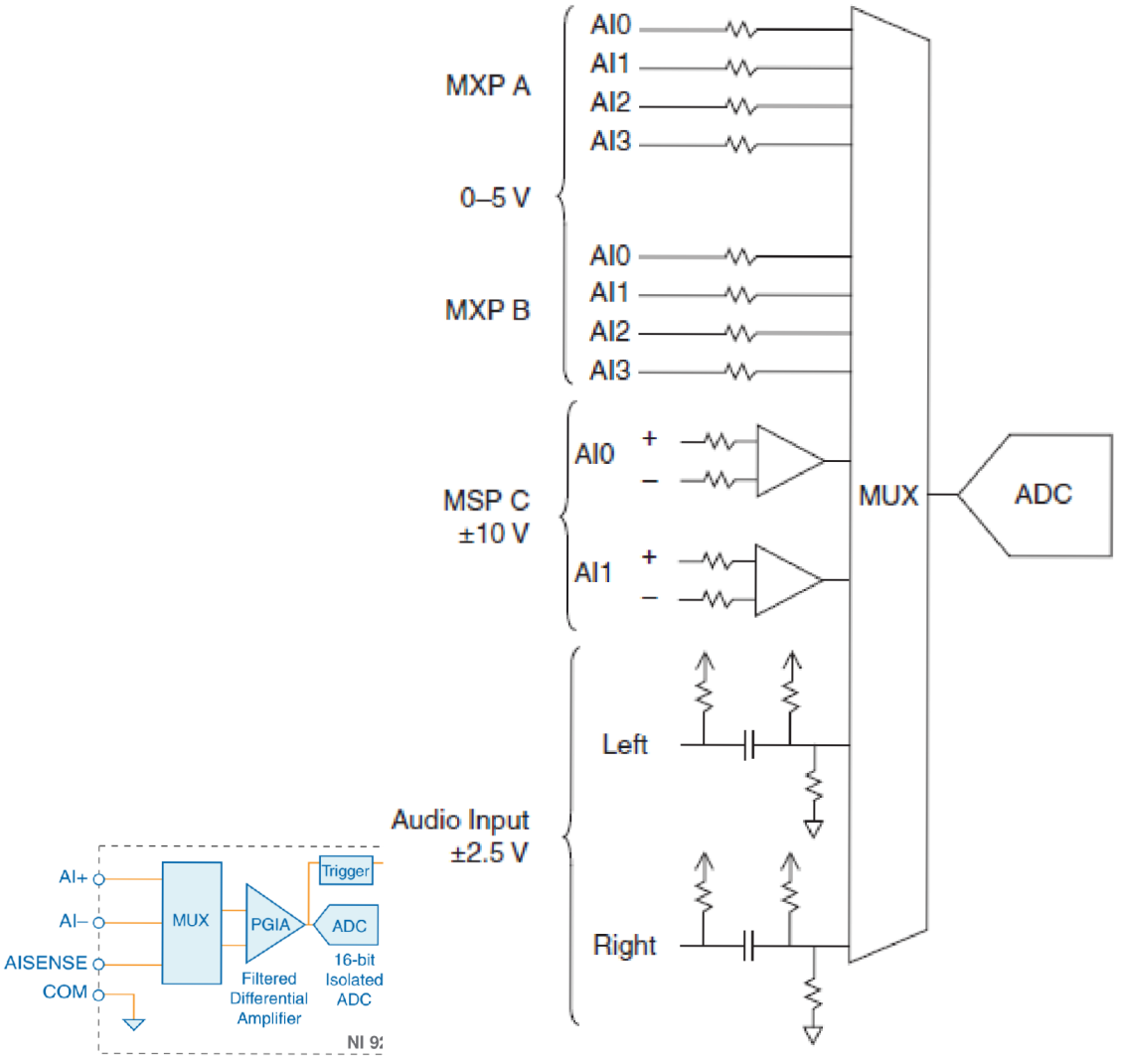


Figure 3.7: Prototype board with INA240 for 3-phase current measuring

Before selecting $5VDC$ as the reference signal for the current-sense amplifier, the capabilities of the analog measurement devices were considered. Since both NI MyRIO and the NI9205 is able to handle $\pm 10VDC$ differential signals on their input, a higher reference signal could have been choosen, however, the MyRIO has only 2 differential inputs. Because of this, a decision was made that if needed, the measurement will be done in a ground-referenced configuration, for which the MyRIOs inputs are usable for up to $5VDC$. Figure 3.8 shows the input stages of the NI MyRIO and the NI9205 C-series module for comparison.



(a) (b)

Figure 3.8: Analog input circuitry for (a) NI 9205 module (b) NI MyRIO

Tables 3.3 - 3.4 show the analog input parameters both for NI MyRIO and the NI9205 measurement card.

Parameter	Value
ADC resolution	16 bits
Conversion time (maximum sampling rate)	4.00 µs (250 kS/s)
Nominal input ranges	± 10 V, ± 5 V, ± 1 V, ± 0.2 V
Minimum overrange, ± 10 V range	4%
Maximum working voltage for analog inputs	Each channel must remain within (signal + common mode) ± 10.4 V of COM
Analog bandwidth	370 kHz
Scaling coefficients	± 5 V range: 164.2 µV/LSB
CMRR, DC to 60 Hz	100 dB
Absolute Accuracy	± 5 V range: Accuracy at full scale 1 3,230 µV Random noise, σ 116 µVrms Sensitivity 2 46.4 µV

Table 3.3: Selected subset of parameters of NI9205 C-Series module

Parameter	Value
ADC resolution	12 bits
Aggregate sample rate	500 kS/s
Overvoltage protection	± 16 V
Maximum working voltage for analog inputs on MSP connector	± 10 V of AGND (signal + common mode)
Nominal range for analog inputs on MXP connector	0 to +5 V
Analog bandwidth for inputs on MSP connector	20 kHz min., >50 kHz typical
Analog bandwidth for inputs on MXP connector	>300 kHz

Table 3.4: Selected subset of parameters for analog input channels on MyRIO

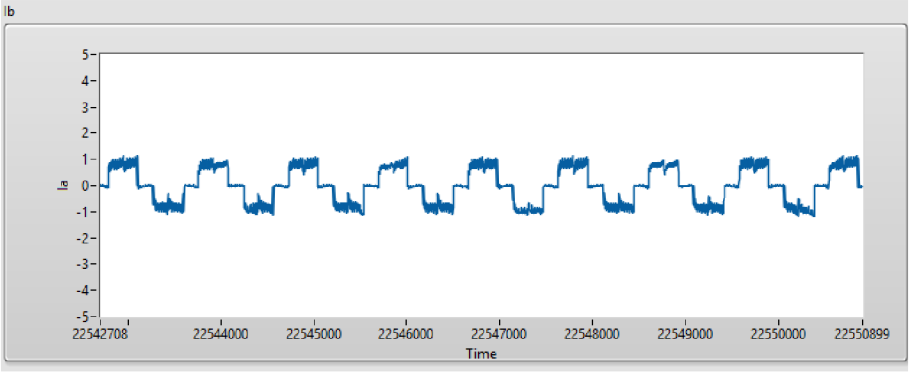


Figure 3.9: Phase current measurement with no clipping- RT VI waveform

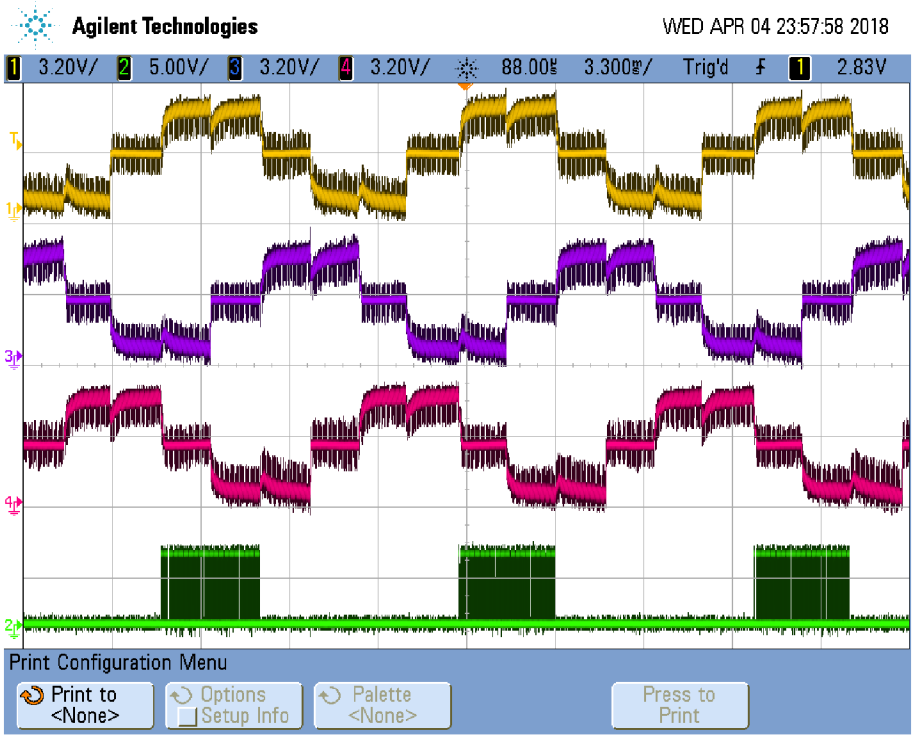


Figure 3.10: Phase current measurement with no clipping- Oscilloscope screen

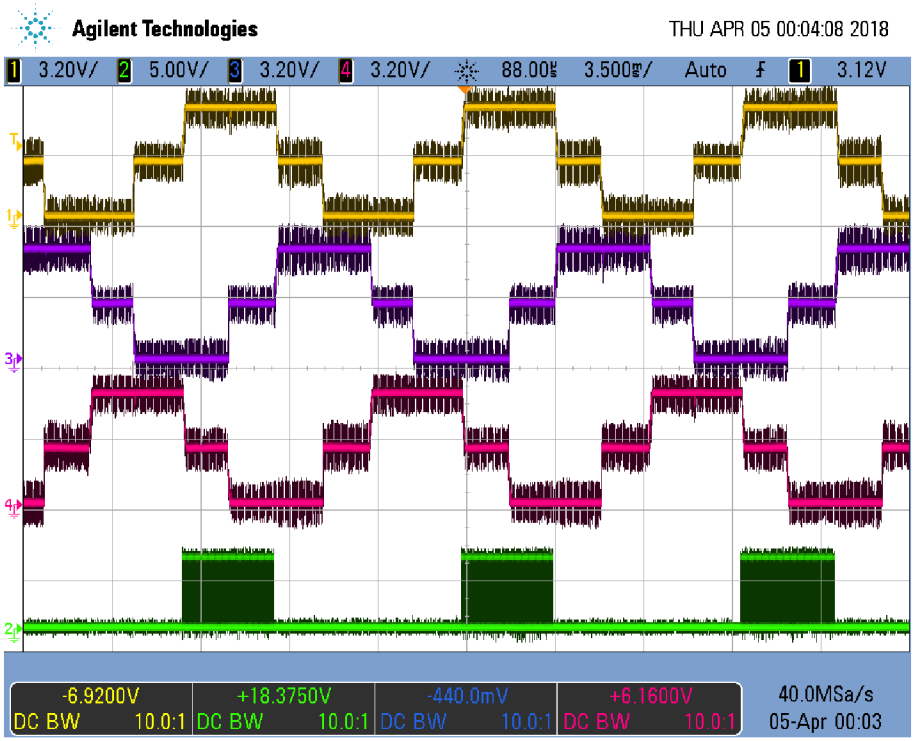


Figure 3.11: Phase current measurement with clipping- Oscilloscope screen

Figure 3.12 shows the implemented FPGA code for current measurement. Only 2 channels are sampled, the third current is calculated. according to equation [1.10]. The acquired samples are passed to a Butterworth filter (if selected) or they are passed to a DMA FIFO after interleaving with samples from other measurement (all together with some data for debugging). [2] Because the analog input channels are multiplexed, sampling 2 channels will take up 8µs, with additional code executing in the same loop, everything should be executed under 12µs which is sufficient enough for our application.

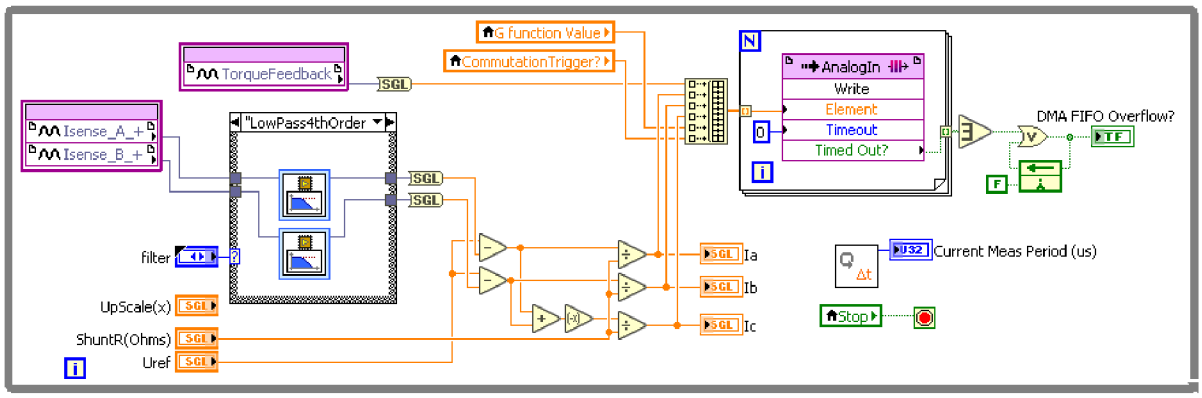


Figure 3.12: FPGA code for current measurement

The parameters of the mentioned butterworth filter can be seen on figure 3.13

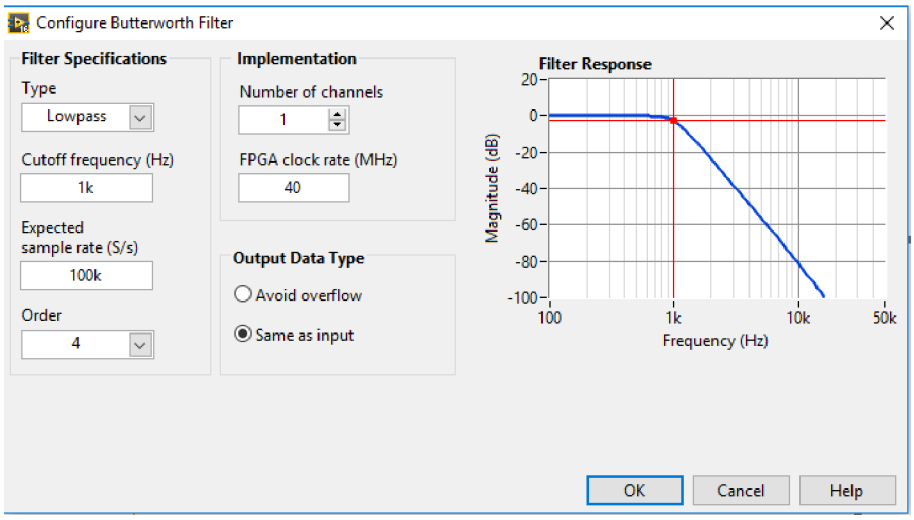


Figure 3.13: Butterworth configuration

After applying the filter to the measured current, the shape of the signal will be as it is demonstraded on figure 3.14. The snapshot is taken from the RT VI which reads the samples from the DMA FIFO and decimates the interleaved array.

[2] Interleaving: usually a limited number of DMA channels are available on the FPGA. Sometimes it is necessary to write data from multiple sources to a single DMA channel. One way to do this is to build an array of data elements from different sources and write them to the DMA channel. This technique is referred as interleaving.

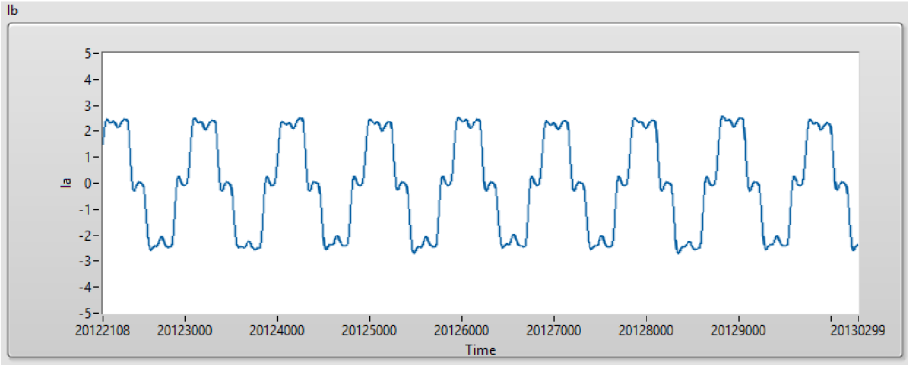


Figure 3.14: Filtered phase current waveform- RT VI waveform

If we compare the measured current with and without filtering, it can be seen that the shape of the acquired waveform is slightly changed. This has a big influence when applying sensor-less methods based on phase current measurement. When considering the use of any kind of filter, its influence has to be investigated and taken into account while developing and simulating sensor-less control.

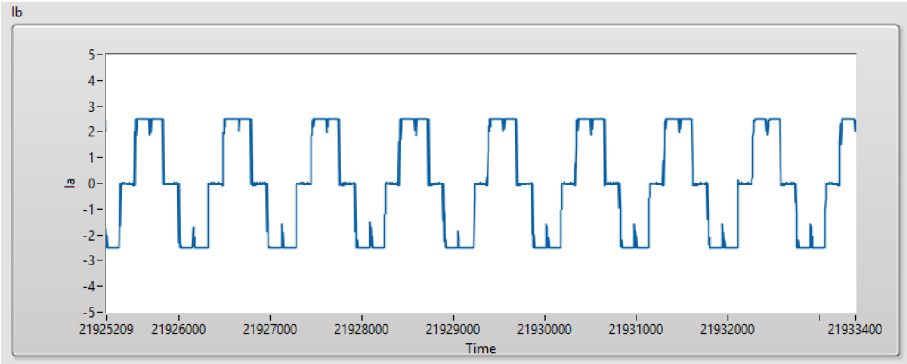


Figure 3.15: Phase current measurement with clipping demonstrated, no filter- RT VI waveform

3.3 LabVIEW FPGA and RT communication methods

In this section a short overview is presented regarding possible communication methods available for FPGA and RT modules.

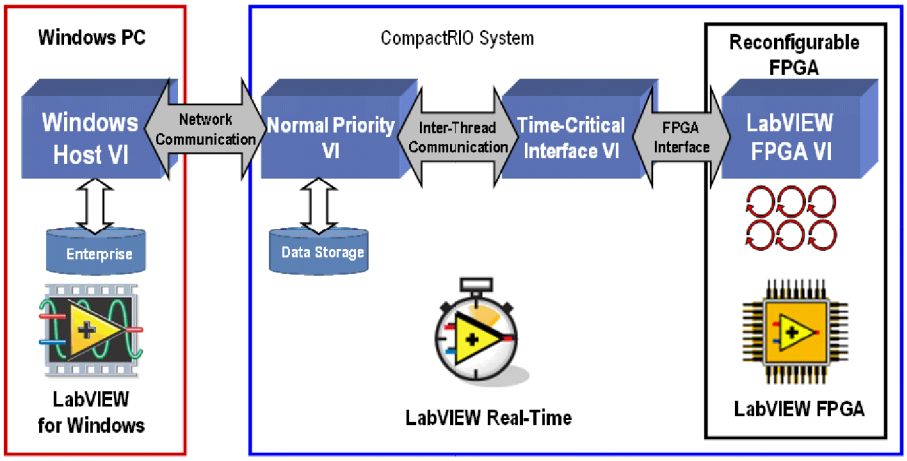


Figure 3.16: Typical architecture using LabVIEW FPGA, LabVIEW Real-Time and PC Host [7]

3.3.1 FPGA inter-process communication

Parallel operations are a very powerful concept in current computer architecture. However in a standard processor-based architecture, parallel operations are not truly parallel. In processor-based architectures, programs running on the processor are sliced into many fragments and are interleaved with code fragments of other processes. The operating system then decides which processes are the most important and schedules the fragments of code accordingly. In an FPGA VI, parallel functions and loops execute in true parallel.

Data sharing methods:

1. FIFO
2. Register item
3. Memory item
4. Variables

Transferring latest data- "tag" communication

The following methods can be used to share, store, and access latest data on the FPGA:

1. Local and global variables
2. Memory items
3. Register items

In LabView, **variables** are block diagram elements that allow you to access or store data in another location. The actual location of the data varies depending on the type of the variable. Local variables store data in front panel controls and indicators. Global variables and single-process shared variables store data in special repositories that can be accessed from multiple VIs. Functional global variables allows the programmer to circumvent normal dataflow by passing data from one place to another without connecting the two places by wire.

Memory items can be used to store data in the FPGA block memory. Memory items reference the block memory on the FPGA target in multiples of 2 kilobytes. Each memory item references a separate address or block of addresses, and memory items can be used to access all available memory on the FPGA. Memory items do not consume logic resources on the FPGA, because they don't include the extra logic necessary to ensure data integrity across clock domains. Each memory address on a memory item stores only the latest value. If a memory address is written N times before reading from the address, the N-1 values preceding the latest value are lost. If there is no need for every acquired data point, memory items are a good choice because there is no need to write extra code to discard unnecessary values.

Register items can be used if there is a need to access stored data from multiple clock domains or from different part of the design and there is a need to write reusable code. Register items consume fewer FPGA logic resources than FIFOs, and they do not consume block memory, which is the most limited type of FPGA resource.

Transferring buffered data - stream, message

To transfer buffered data between different portions of an FPGA VI or between VIs in an FPGA target, a FIFO can be used. A FIFO is a first-in-first-out buffer, where the first data item written to memory is the first item read and removed from the memory.
The following table compares data sharing methods [3]:

Transfer Method	FPGA Resource	Type of Transfer	Between Clock Domains?	Common usage
Variables	Logic	Tag	Yes	Share Latest Data
Memory Items	Memory	Tag	No	Share Latest Data
Register Items	Logic	Stream, Message	Yes	Share Latest Data
Flip-Flop FIFOs	Logic	Stream, Message	No	Transfer Buffered Data (FIFOs <100 bytes)
Look-Up Table FIFO's	Logic	Stream, Message	No	Transfer Buffered Data (FIFOs from 100 to 300 bytes)
Block Memory FIFO's	Logic and Memory	Stream, Message	Yes	Transfer Buffered Data (FIFOs >300 bytes)

Table 3.5: Data sharing methods comparison

[3] information source: www.ni.com/training/

3.4 PID Control [3] [4]

3.4.1 CLOSED-LOOP CONTROL VERSUS OPEN-LOOP CONTROL

A system that maintains a prescribed relationship between the output and the reference input by comparing them and using the difference as a means of control is called a feedback control system. An example would be a room temperature control system. By measuring the actual room temperature and comparing it with the reference temperature (desired temperature), the thermostat turns the heating or cooling equipment on or off in such a way as to ensure that the room temperature remains at a comfortable level regardless of outside conditions. Feedback control systems are not limited to engineering but can be found in various nonengineering fields as well.

Closed-Loop Control Systems
Feedback control systems are often referred to as closed-loop control systems. In practice, the terms feedback control and closed-loop control are used interchangeably. In a closed-loop control system the actuating error signal, which is the difference between the input signal and the feedback signal (which may be the output signal itself or a function of the output signal and its derivatives and/or integrals), is fed to the controller so as to reduce the error and bring the output of the system to a desired value. The term closed-loop control always implies the use of feedback control action in order to reduce system error.

Open-Loop Control Systems
Those systems in which the output has no effect on the control action are called open-loop control systems. In other words, in an open-loop control system the output is neither measured nor fed back for comparison with the input. One practical example is a washing machine. Soaking, washing, and rinsing in the washer operate on a time basis. The machine does not measure the output signal, that is, the cleanliness of the clothes. In any open-loop control system the output is not compared with the reference input. Thus, to each reference input there corresponds a fixed operating condition; as a result, the accuracy of the system depends on calibration. In the presence of disturbances, an open-loop control system will not perform the desired task. Open-loop control can be used, in practice, only if the relationship between the input and output is known and if there are neither internal nor external disturbances. Clearly,such systems are not feed- back control systems. Note that any control system that operates on a time basis is open loop. For instance, traffic control by means of signals operated on a time basis is another example of open-loop control.

The major advantages of open-loop control systems are as follows:

- Simple construction and ease of maintenance.
- Less expensive than a corresponding closed-loop system.

- There is no stability problem.
- Convenient when output is hard to measure or measuring the output precisely is economically not feasible. (For example, in the washer system, it would be quite expensive to provide a device to measure the quality of the washer's output, cleanliness of the clothes.)

The major disadvantages of open-loop control systems are as follows:

- Disturbances and changes in calibration cause errors, and the output may be different from what is desired.
- To maintain the required quality in the output, recalibration is necessary from time to time.

The Three Actions of PID Control

Applying a PID control law consists of applying properly the sum of three types of control actions: a *proportional action*, an *integral action* and a *derivative* one.

Proportional Action

The proportional control action is proportional to the current control error, according to the expression:

$$u(t) = K_p e(t) = K_p(r(t) - y(t)), \tag{3.1}$$

where K_p is the proportional gain. Its meaning is straightforward, since it implements the typical operation of increasing the control variable when the control error is large (with appropriate sign). The transfer function of a pro- portional controller can be derived trivially as:

$$C(s) = K_p \tag{3.2}$$

The main drawback of using a pure proportional controller is that it produces a steady-state error.

Integral Action

The integral action is proportional to the integral of the control error, i.e., it is:

$$u(t) = \int_0^t e(\tau)d\tau \tag{3.3}$$

where K_i is the integral gain. It appears that the integral action is related to the past values of the control error. The corresponding transfer function is:

$$C(s) = \frac{K_i}{s} \tag{3.4}$$

Derivative Action

While the proportional action is based on the current value of the control error and the integral

action is based on the past values of the control error, the derivative action is based on the predicted future values of the control error. An ideal derivative control law can be expressed as:

$$u(t) = K_d \frac{de(t)}{dt} \tag{3.5}$$

where K_d is the derivative gain. The corresponding controller transfer function is:

$$C(s) = K_d s \tag{3.6}$$

3.4.2 PID loop in FPGA

There are lots of PID controller types available nowadays, designed for specific applications. Some of the applications only require the classic PID controller without any advanced feature, other applications need anti-windup implementation, set-point weighting techniques and so on. Since in our case the controller is realized on an FPGA, additional variations are possible using different data types for realization which in the end will influence the PID controller performance also FPGA resource utilization. Before selecting the type of PID controller to be implemented, usually a compromise has to be made between resource utilization, performance and advanced features. LabView FPGA module comes with an advanced multi-channel controller, which can be configured as a proportional, proportional-integral or proportional-integral-derivative controller. The algorithm is realized in single-precision floating-point arithmetic. Also there are implementations which are using fixed-point [4] implementation which requires less FPGA resources. In this project the controller is realized in floating-point. The used implementation is not shipped with LabView FPGA module, however it is available at: [10]

The PID VI calculates the output, u(k), according to the following equations:

$$\begin{aligned} u(k) &= uP(k) + uI(k) + uD(k) \\ uP(k) &= K_p.e'(k) \\ uI(k) &= K_i.\sum_i^k \left[\frac{e(i) + e(i-1)}{2}\right] \\ uD(k) &= K_d.[e''(k) - e''(k-1)] + a.uD(k-1) \\ e'(k) &= SP(k).beta - PV(k) \\ e''(k) &= SP(k).gamma - PV(k) \end{aligned} \tag{3.7}$$

where
Kp — is the proportional gain;
Ki — is the integral gain;
Kd — is the derivative gain;
a — is the filter coefficient;
SP — is the setpoint;
$beta$ — is the proportional weighting;
PV — is the process variable;
$gamma$ — is the derivative weighting.

Figure 3.17 displays the FPGA PID VI used in a parallel loop to achieve speed control of the BLDC motor. When there is no control required, e.g the PWM generation loop is switched to

[4] Fixed point is a format for representing numbers on digital processing devices. It is a data type used by a programming language or hardware descriptive language (HDL) to determine how to interpret bits in a memory location. [9]

safe-state, this loop also will change to safe-state, hence it will not use the PID controller. When switching back from safe-state to PID control, the controller itself will be re-set to achieve bump-less activation of the algorithm. Motor speed (RPM) information is passed to the controller via VI-defined register, the calculated control action (PWM duty) is passed to the PWM generation parallel loop using again a VI-defined register.

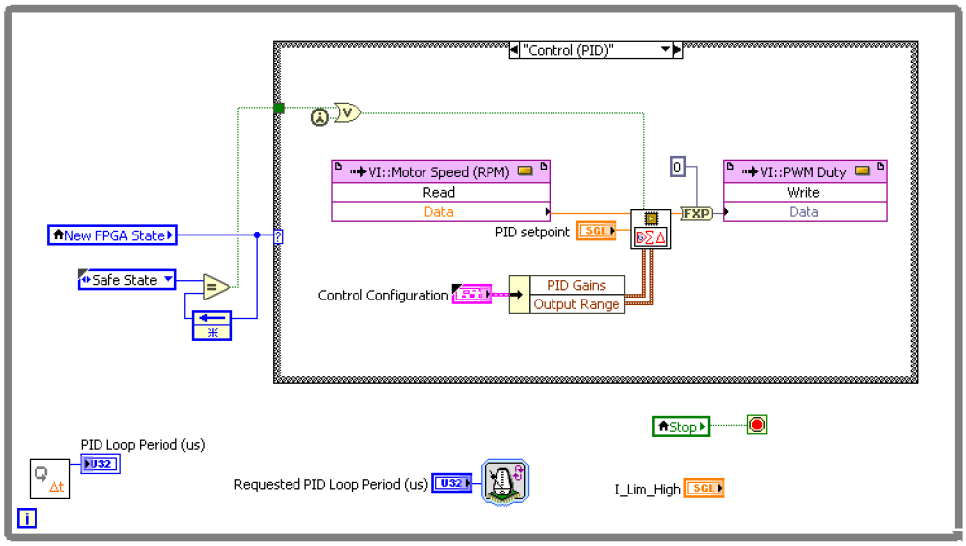


Figure 3.17: FPGA code with PID controller inside

Figure 3.18 displays the available parameters to be set for the controller. During development the maximum output range was limited to 0.7 which means that the maximum duty cycle for the PWM generation is 70 %. Controller configuration is done using programmatic front panel communication, which means that the latest configuration data is used by the controller (Configuration set in PC application, then sent to RT VI, then passed to FPGA VI via front panel control). The set-point of the controller is set using the same data-transfer methodology.

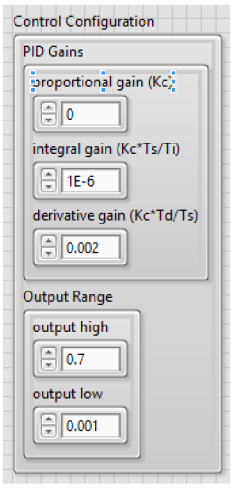


Figure 3.18: PID configurable parameters

3.5 RPM Measurement

When controlling the BLDC motor in sensored mode, the rotational speed can be measured using the hall-effect sensors which are mounted close to the motor's rotor. Hall-sensor signals are read with standard bi-directional DIO, with update rate of 7 µ seconds. The input lines are not multiplexed, hence the measurement time will not increase if we sample more than one channel. To calculate the RPM value, the high-period of the input signal is measured, and passed over to a parallel loop which does the calculation according the following formula:

$$T_{actual}(sec/rev) = T(\mu s/pulse) * \frac{1}{1,000,000} * (12 pulses/rev) \tag{3.8}$$

$$RPM = \frac{1}{T_{actual}(sec/rev)} * (60 sec/min) \tag{3.9}$$

Before passing the information about the high-period of the signal, it is filtered to remove noise introduced by used long wires in our experimental set-up. 10-15 cycles to be filtered out is sufficient for our purpose. The resulting RPM value is loaded into a VI-defined register which is read afterwards by the PID control loop. Sampled hall-sensor signals are also passed to the highest-priority parallel loop which is responsible for PWM signal generation. For this inter-loop communication the "tag" method is used.

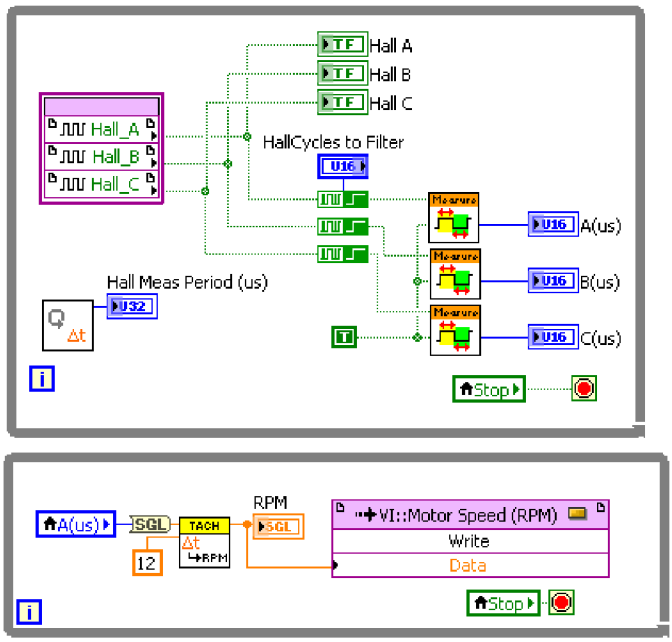


Figure 3.19: FPGA code for sampling hall sensors and for RPM calculation

Figure 3.19 illustrates two parallel loops realized in FPGA. It can be noted that for RPM calculation only one hall sensor is used. It proved to be sufficient for speed control.

3.6 Sensorless Operation

There are different well-documented sensor-less motor control techniques available, with different approaches and incorporated mathematical apparatus. Majority of these techniques uses the measurement of BEMF signal. The selected approach in this document does not measure BEMF signal directly, it requires in-line current measurement instead which is used to indirectly estimate BEFM signal and calculate the commutation instants. The challenges and design considerations were already described, in this section the selected sensor-less technique(s) is (are) described. The next 2 technique are similar to each other - basically both are using phase current measurements and the switching states of the inverter.

3.6.1 Method I

This method presents an approach, which applied correctly, can be used from near zero speed to control a BLDC motor using motor phase current measurement (at least 2 phases) and the known switching states of the controller. It is presented in [11]. The $H(\theta)_{ab}$ function is defined as follows:

$$H(\theta)_{ab} = \frac{df_{abr}(\theta)}{d\theta} \tag{3.10}$$

,where
$f_{abr}(\theta)$ - is a line-to-line flux linkage form function that is a function of the rotor position.
Then $H(\theta)_{ab}$ can be derived as:

$$H(\theta)_{ab} = \frac{1}{\omega.k_e}\left[(V_a - V_b) - R(i_a - i_b) - L\left(\frac{di_a}{dt} - \frac{di_b}{dt}\right)\right] \tag{3.11}$$

Since $H(\theta)_{ab}$ the function itself has a one to one relationship with rotor position, it is possible to use this function for position estimation. To know the function, the instantaneous speed term, that is unknown for dynamic operations, is required to calculate the function.
To eliminate the instantaneous speed term, ω , that causes trouble in using the function for position estimation, one line-to-line function is divided by another line-to-line function, and the divided new speed independent function is named $G(\theta)$. [5]
Equation 3.12 shows how we can eliminate mathematically the speed term if we divide two line-to-line $H(\theta)_{xx}$ functions.

[5] Sensorless control of the BLDC motors from near-zero to high speeds [11]

$$\frac{H(\theta)_{bc}}{H(\theta)_{ab}} = \frac{\frac{1}{\omega.k_e}\left[(V_b - V_c) - R(i_b - i_c) - L\left(\frac{di_b}{dt} - \frac{di_c}{dt}\right)\right]}{\frac{1}{\omega.k_e}\left[(V_a - V_b) - R(i_a - i_b) - L\left(\frac{di_a}{dt} - \frac{di_b}{dt}\right)\right]} = \frac{\left[(V_b - V_c) - R(i_b - i_c) - L\left(\frac{di_b}{dt} - \frac{di_c}{dt}\right)\right]}{\left[(V_a - V_b) - R(i_a - i_b) - L\left(\frac{di_a}{dt} - \frac{di_b}{dt}\right)\right]} = G(\theta)_{bc/ab} \tag{3.12}$$

Equation 3.13 shows the digitized version of the derived function. Where, $S_a(k)$, $S_b(k)$ and $S_c(k)$ are switching status of phase A,B, and C- and they are known values by the controller.

$$\frac{H(\theta)_{bc}}{H(\theta)_{ab}} = \frac{V_{bus}.(S_b(k) - S_c(k)) - R.(i_b(k) - i_c(k)) - L.\left(\frac{(i_b(k) - i_b(k-1)) - (i_c(k) - i_c(k-1))}{t_k - t_{k-1}}\right)}{V_{bus}.(S_a(k) - S_b(k)) - R.(i_a(k) - i_b(k)) - L.\left(\frac{(i_a(k) - i_a(k-1)) - (i_b(k) - i_b(k-1))}{t_k - t_{k-1}}\right)} \tag{3.13}$$

The presented sensor-less method has been wired and simulated in LabView, using NI Multisim co-simulation. As for the power inverter realization, for simplicity, the inverter presented on figure 2.5 is used all together with the BLDC motor model. The simulation loop was set to have discrete states only (eg. simulation in discrete-time) with minimum step size: $1x10^{-7}$ Figure 3.20 displays the simulation results.

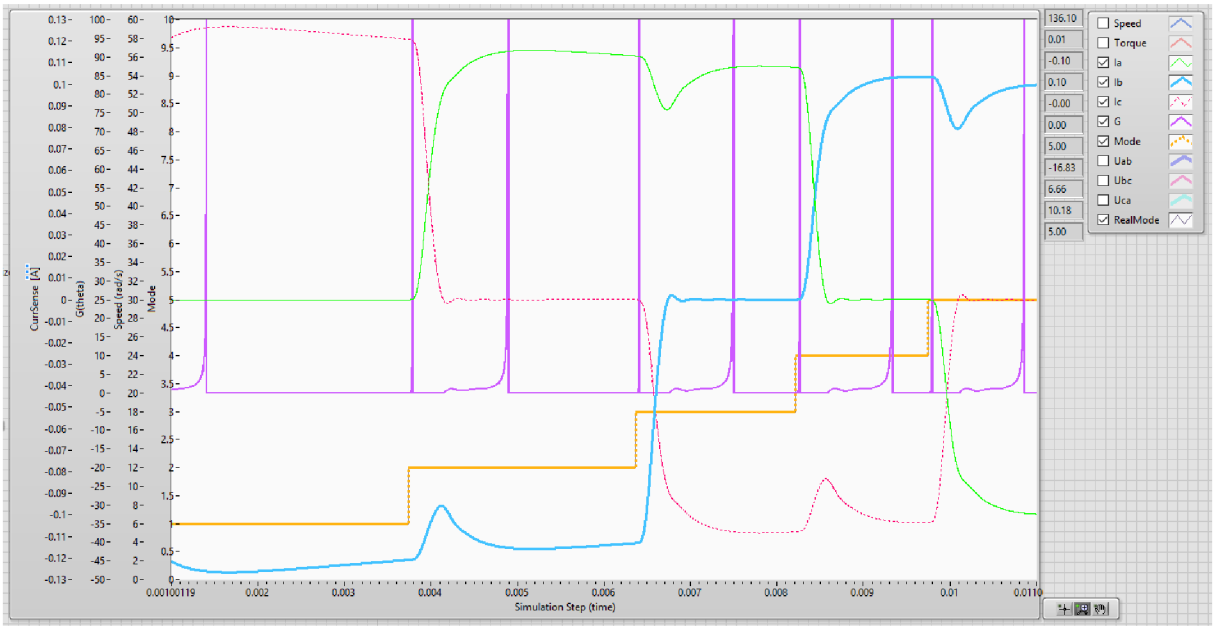


Figure 3.20: Simulation of $G(\theta)$ function

The simulation is programmed in a way, that it uses hall-sensor feedback for generating the commutation instants, calculating the G function in each mode. This way the correct implementation

of the G function can be verified. Calculated G function values are represented in the resulting chart with *magenta* color. Function values below 0 has been clipped to have zero value. It can be seen that the estimated commutation instant comes "early" (= peaks in G function value), afterwards there is again a peak value when the real commutation instant is happening (according to hall-sensor feedback). With higher motor speed, the gap between the estimated commutation instant and the real one will be narrow. This means that the estimation error at lower speeds will be bigger, at higher speeds it will be negligible. Figure 3.20 also displays phase currents (red, blue, green lines) and the switching state of the inverter (orange line).

This sensor-less approach has been implemented to FPGA, figure 3.21 displays the realized code.

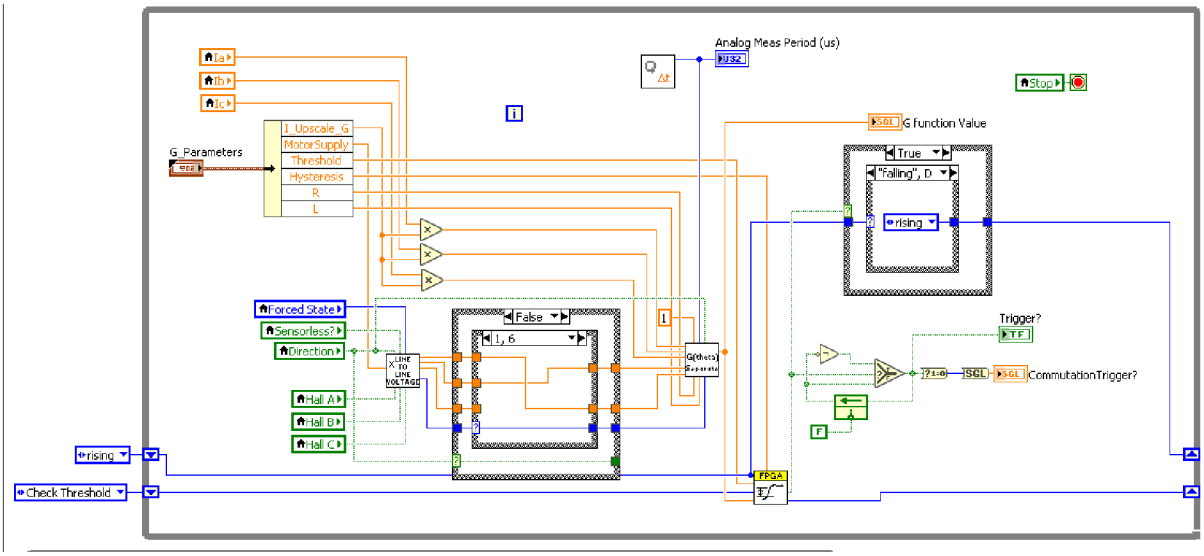


Figure 3.21: FPGA realization of $G(\theta)$ calculation in a parallel loop

Again, as in the simulation, the same approach was taken in FPGA implementation as it was in the simulation part: implement the algorithm, and investigate the results *during* sensored commutation using hall-sensor feedback. This approach helps in finding programming mistakes. Figure 3.22 displays the calculated G functions during sensored commutation all together with *should-be* commutation instants which were detected from the calculated G function.

3.6.2 Method II

This method can be considered as an upgrade (improvement) to the first one. The method is presented in [8] and it does not use differentiation while calculating the commutation instants. The approach proposed in the above mentioned paper uses an unknown input observer to estimate BEMF from current measurement. [6]

[6] State Observer: a state observer estimates the state variables based on the measurements of the output and control variables. State observers can be designed if and only if the observability condition is satisfied. [4]

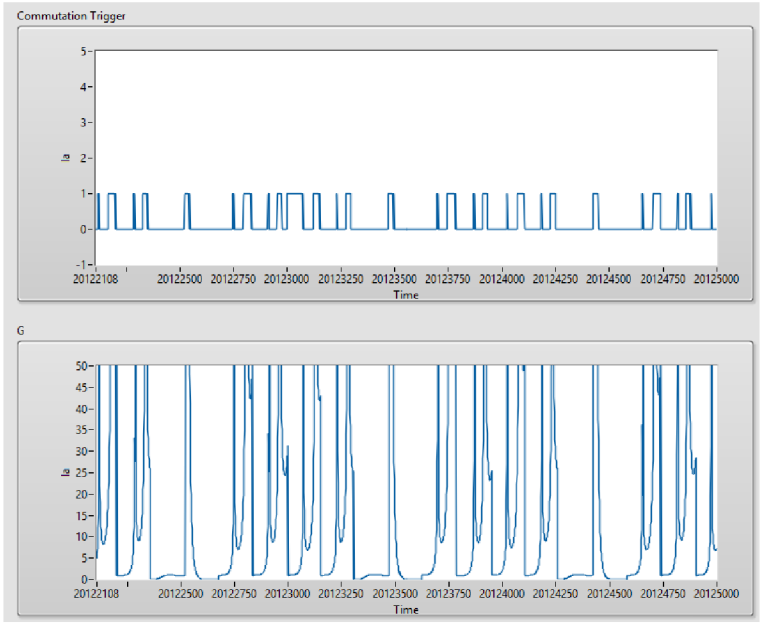


Figure 3.22: Calculated G-function values (lower waveform) and detected commutation instants (upper waveform)

Mode I and IV	$G(\theta)_1 = \frac{H(\theta)_{ca}}{H(\theta)_{bc}} = \frac{V_{ca} + Ri_{ca} + L\frac{di_{ca}}{dt}}{V_{bc} - Ri_{bc} - L\frac{di_{bc}}{dt}}$
Mode II and V	$G(\theta)_2 = \frac{H(\theta)_{bc}}{H(\theta)_{ab}} = \frac{V_{bc} + Ri_{bc} + L\frac{di_{bc}}{dt}}{V_{ab} - Ri_{ab} - L\frac{di_{ab}}{dt}}$
Mode III and VI	$G(\theta)_3 = \frac{H(\theta)_{ab}}{H(\theta)_{ca}} = \frac{V_{ab} + Ri_{ab} + L\frac{di_{ab}}{dt}}{V_{ca} - Ri_{ca} - L\frac{di_{ca}}{dt}}$

Table 3.6: G function at each mode

Figure 3.23 illustrates the block diagram of the BEMF observer.

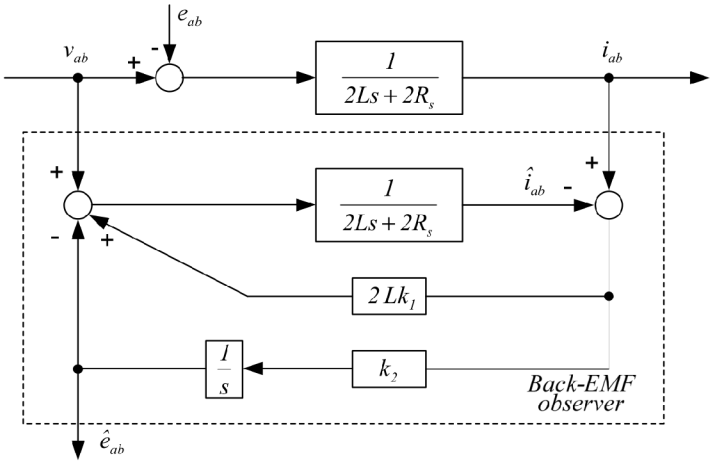


Figure 3.23: Block diagram of BEMF observer [8]

Figure 3.24 illustrates the LabView counterpart of the above mentioned observer created for simulation.

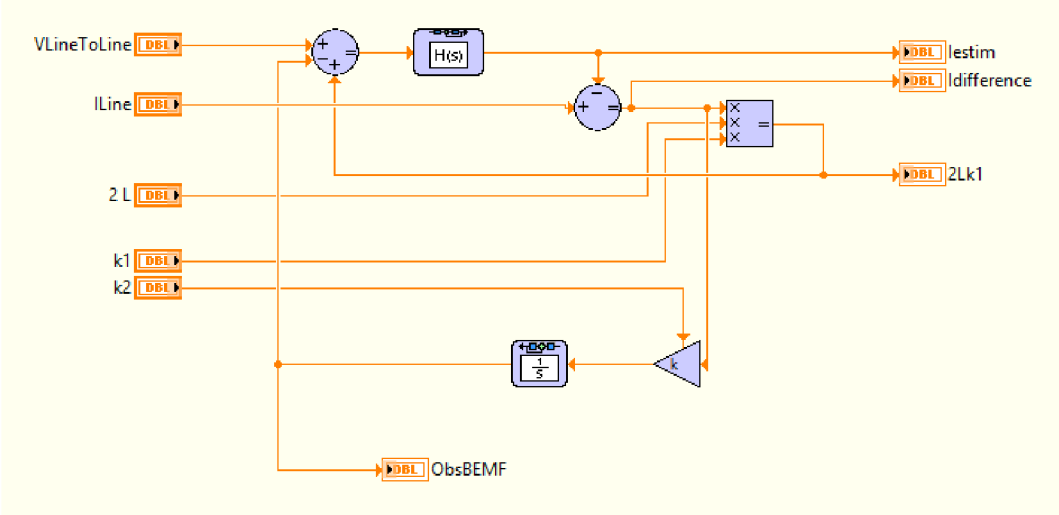


Figure 3.24: BEMF observer block diagram programmed in LabView (1)

If the gain of the observer is selected properly, this observer can accurately estimate line-to-line currents and back- EMFs of motors. Using the estimated BEMFs, using functions in table 3.7 the commutation instants can be estimated choosing an adequate threshold.
Note the similarity between equation 3.7 and 3.6. If the observer gains are properly set in equation 3.14 then the observer can accurately estimate line-to-line currents and BEMFs of the BLDC motor.

Mode I and IV	$CF(\theta)_1 = \frac{\hat{e}_{bc}}{\hat{e}_{ca}}$
Mode II and V	$CF(\theta)_1 = \frac{\hat{e}_{bc}}{\hat{e}_{ca}}$
Mode III and VI	$CF(\theta)_1 = \frac{\hat{e}_{bc}}{\hat{e}_{ca}}$

Table 3.7: C function at each mode

$$\frac{d}{dt}\begin{bmatrix}\hat{i}_{ab}\\ \hat{e}_{ab}\\ \hat{i}_{bc}\\ \hat{e}_{bc}\\ \hat{i}_{ca}\\ \hat{e}_{ca}\end{bmatrix} = \begin{bmatrix} -\frac{2R_s}{2L} & -\frac{1}{2L} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{2R_s}{2L} & -\frac{1}{2L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{2R_s}{2L} & -\frac{1}{2L} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix}\hat{i}_{ab}\\ \hat{e}_{ab}\\ \hat{i}_{bc}\\ \hat{e}_{bc}\\ \hat{i}_{ca}\\ \hat{e}_{ca}\end{bmatrix} + \begin{bmatrix} \frac{1}{2L} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2L} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2L} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_{ab} \\ v_{bc} \\ v_{ca} \end{bmatrix} + \begin{bmatrix} k1 & 0 & 0 \\ k2 & 0 & 0 \\ 0 & k3 & 0 \\ 0 & k4 & 0 \\ 0 & 0 & k5 \\ 0 & 0 & k6 \end{bmatrix} \begin{bmatrix} i_{ab} - \hat{i}_{ab} \\ i_{bc} - \hat{i}_{bc} \\ i_{ca} - \hat{i}_{ca} \end{bmatrix} \tag{3.14}$$

Figure 3.25 illustrates the block diagram of a different modeled BEMF observer. The BEMF voltage e_a, is calculated indirectly by passing the error between actual measured current and estimated current through PI controller. The idea is pretty straight forward, in order to make error between actual current and measured current to zero, the PI controller output has to become the actual BEMF e_a [28]

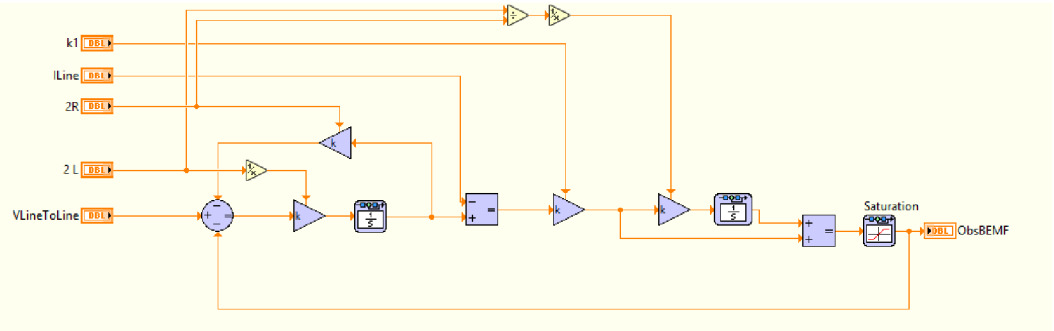


Figure 3.25: BEMF observer block diagram programmed in LabView (2)

To verify the basic functionality (partially) of the proposed method, the Multisim model has been modified, and the BEMF signals are pulled out from the BLDC motor model. The signals are further used in LabView to demonstrate the function shapes, verifying that if the BEMF signals are correctly observed (estimated) than the correct commutation instant can be estimated.

Figure 3.26 illustrates the output waveforms of the realized simulation according to 3.7. For this simulation the real BEMF signals used which were pulled from Multisim BLDC motor model.

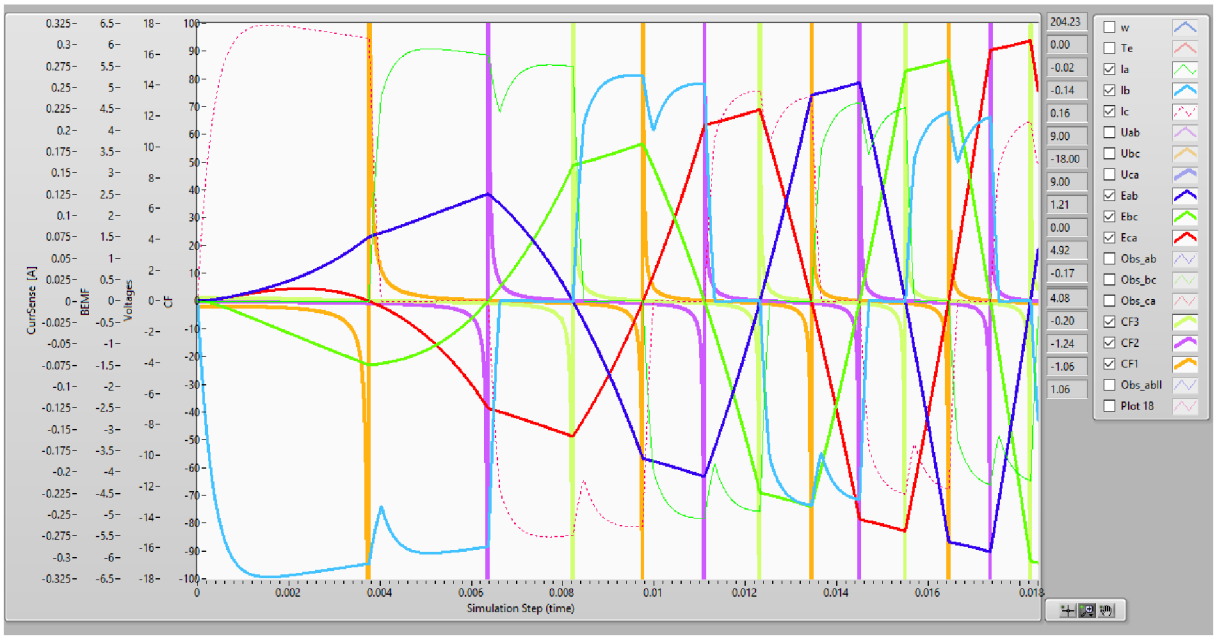


Figure 3.26: Simulation of CF functions for sensor-less control using real BEMF signals from model

Figure 3.27 illustrates the output waveforms of the realized simulation with the use of real BEMF signal and the observed (estimated) one using the observer model shown on figure 3.25

For simplicity on this simulation waveform the C function is only calculated for 2 modes of operation of the inverter. In this simulation result also can be noted (as before when calculating G function) that the estimated commutation instant precedes the real commutation instants with the use of hall-sensor feedback.

Unfortunately the later method - BEMF observer - was not implemented into FPGA, due to limited amount of time, however with high probability this approach will probably be more robust because it does not require the use of difference calculation, hence it should be less noise-sensitive.

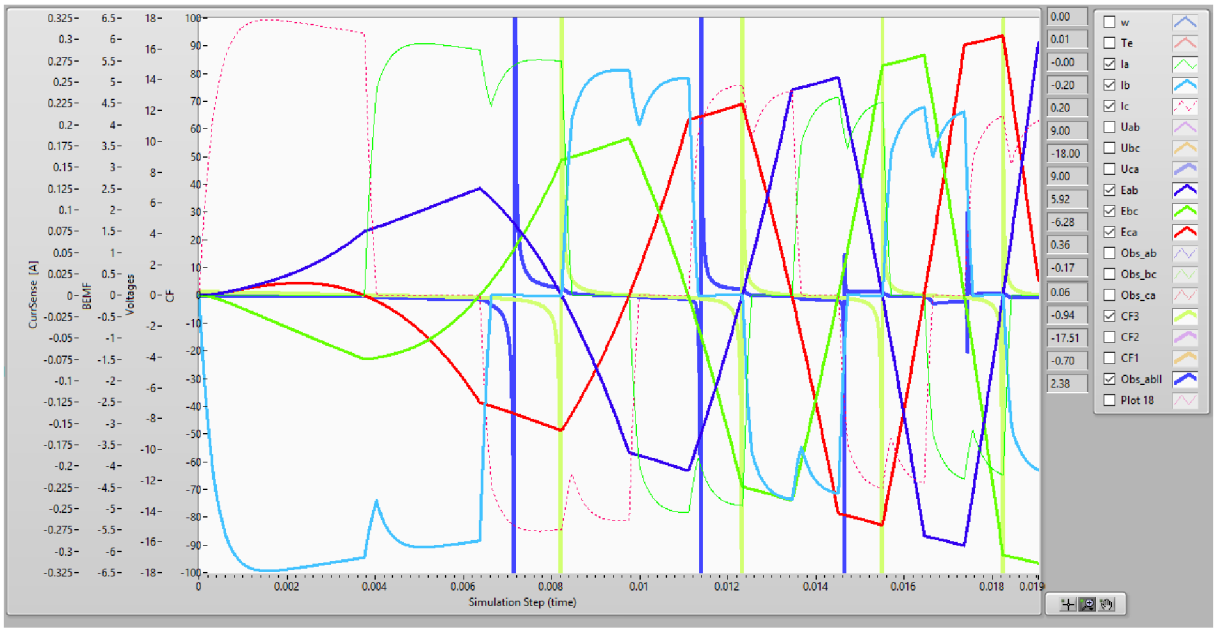


Figure 3.27: Simulation of CF functions for sensor-less control using observed BEMF signals from model

3.7 FPGA - 6-step commutation and PWM generation

As for the control method the well-known six-step commutation method is used, which is relatively easy to implement (both in simulation and FPGA) using a simple look-up table. The upper transistors in the inverter bridge are driven with PWM signal, the lower transistors are switched with standard DIO lines. To generate PWM signals, the VI in figure 3.28 is used. It is configured to generate 1 center-aligned PWM signal with implicit timing. The PWM frequency can be configured, however, if the default setting is used the frequency will be $f = 25\ kHz$ which is sufficient for most applications.

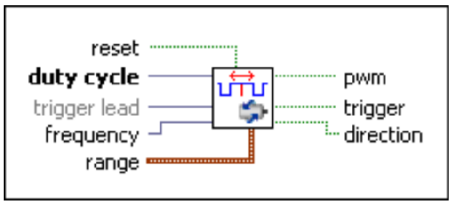


Figure 3.28: VI for PWM generation

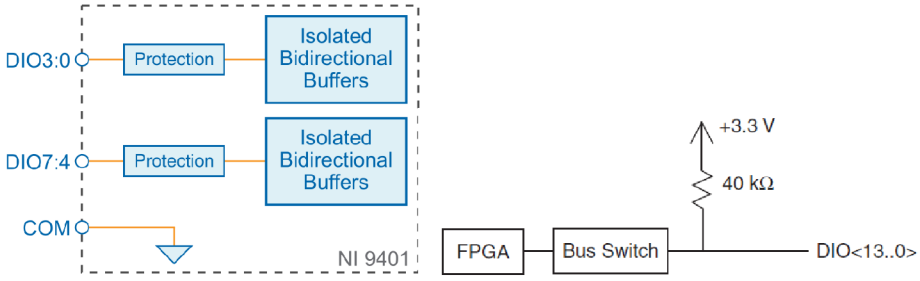
This VI is delivered together with the NI SoftMotion module, and has advanced features. The most important feature here (after the fact that it is generating PWM signal) that it has the **trigger** output which can be used for current measurement synchronization (trigger) to sample the currents close to the middle point of the high-period of the PWM signal.

To avoid shorting the transistors in the inverters legs, a PWM delay (dead-time) is added. [7] For our inverter bridge a dead-time of $50\ ticks$ or $1.25\ \mu s$ is used- if there would be a change in the transistor types, the dead-time can be changed accordingly.
To generate the PWM signals the NI9401 C-series module is used, its output stage is shown on figure 3.29 together with the NI MyRIO output stage for comparison.

It is important to note that because of the configuration of these output stages, during powering up the cRIO, it can happen that the outputs will for a shorter time go to an undefined state before outputting low-logic level. This needs to be considered when the system will be used in the future, because if the inverter is already powered-on before the output stages are correctly initialized, the transistors will be destroyed immediately rendering the inverter useless.

Figure 3.30 shows the simple look-up table for decoding the sampled hall signals and converting them to 6 digital signals, each one representing a digital state of the transistors in the inverter bridge. The switching order is defined separately for each direction of the motor.

[7] IGBTs and other transistors are not ideal switches, turn on times and turn off times are not strictly identical. In order to avoid bridge shoot through it is always recommended to add a so called “interlock delay time” or more popular “dead time” into the control scheme. With this additional time one transistor will be always turned off first and the other will be turned on after dead time is expired, hence bridge shoot through caused by the unsymmetrical turn on and turn off times of the transistors can be avoided [12]



(a) (b)

Figure 3.29: Digital output circuitry for (a) NI 9205 module (b) NI MyRIO

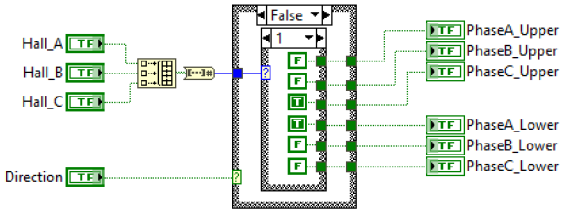


Figure 3.30: Look-up table for six-step commutation

Figure 3.31 shows the FPGA loop which is responsible for generating the PWM signals for the inverter bridge. It also has some additional logic which was implemented during experimenting with sensor-less control. If the loop is in safe-state, it will continuously output low-logic level to the digital outputs switching of all the transistors and leaving the motor unenergized.

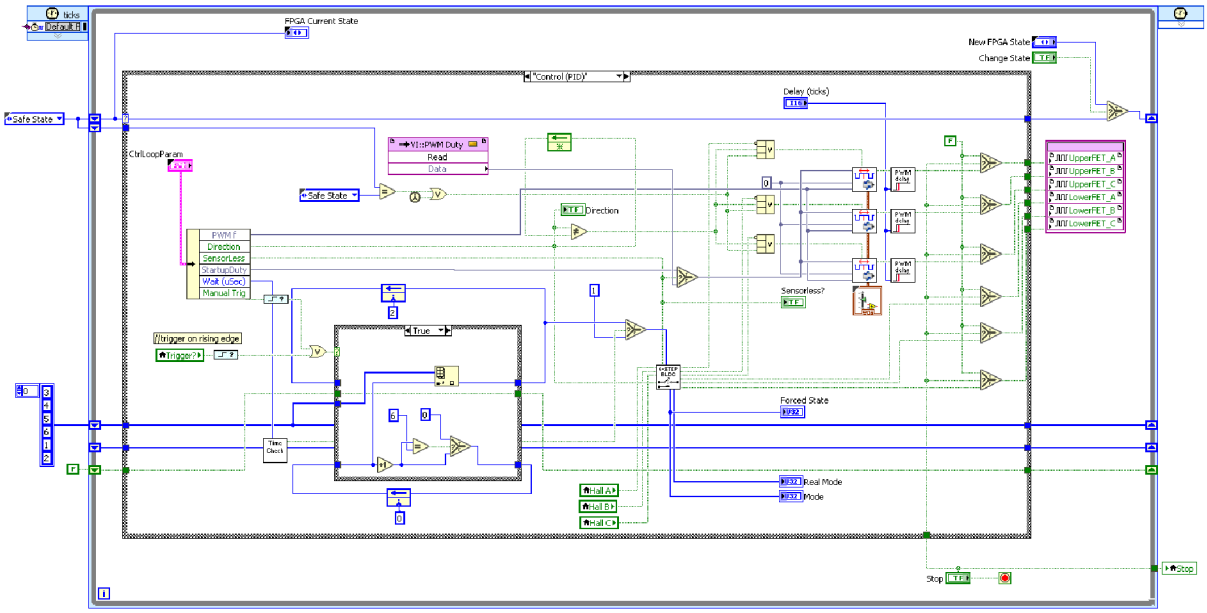


Figure 3.31: FPGA code for PWM signal generation

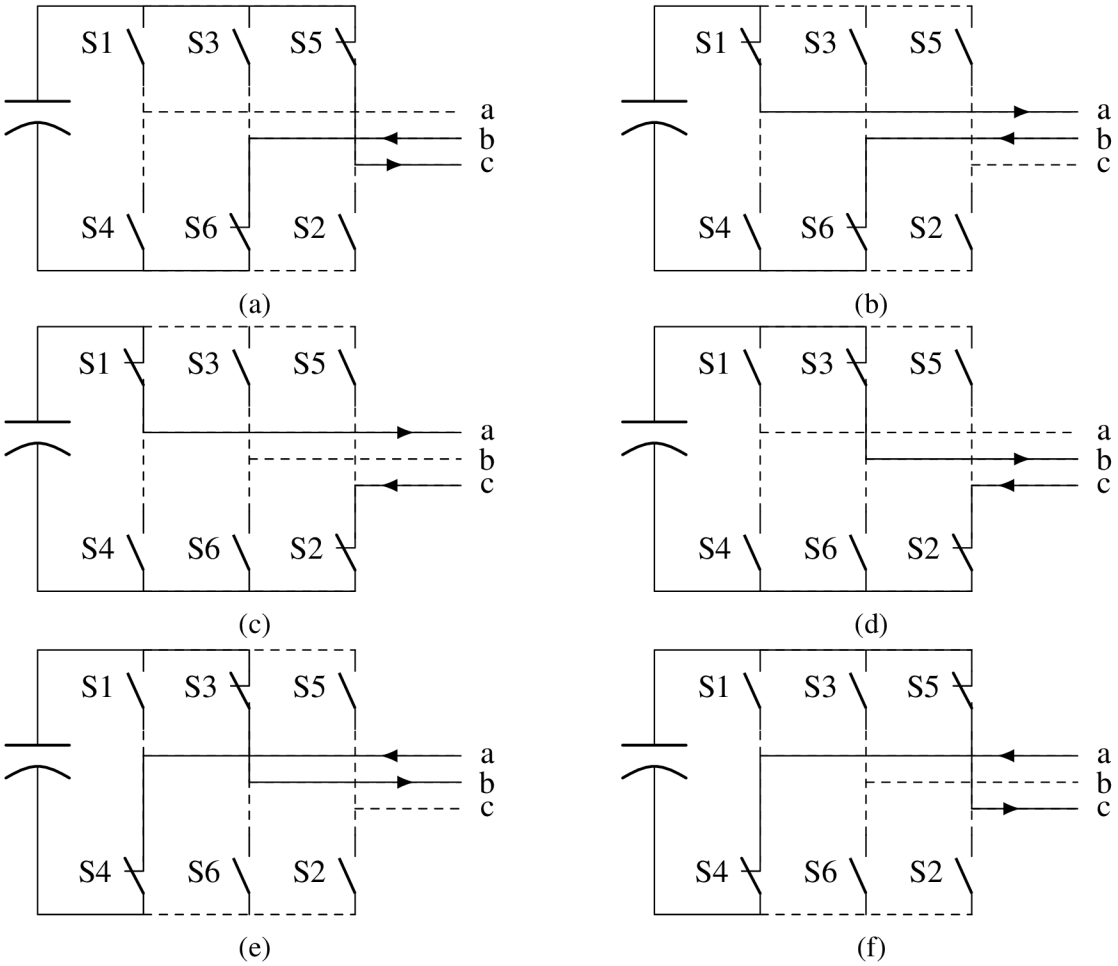


Figure 3.32: Switching states and conduction sequence according to the operating modes (a) Mode I. (b) Mode II. (c) Mode III. (d) Mode IV. (e) Mode V. (f) Mode VI.

Figure 3.32 shows the switching states and the conduction sequence for each operating mode of the inverter bridge.

For the presented sensor-less methods it is important to know in which state is the inverter, to allow the calculation of line-to-line voltages. Using the *switching function concept* which is presented in [13], the line-to-line voltages can be calculated at each mode using the following equations:

$$V_{a0} = \frac{V_d}{2}.SF_a \tag{3.15}$$

$$V_{b0} = \frac{V_d}{2}.SF_b \tag{3.16}$$

$$V_{c0} = \frac{V_d}{2}.SF_c \tag{3.17}$$

$$V_{ab} = V_{a0} - V_{b0} = \frac{V_d}{2}(SF_a - SF_b) \tag{3.18}$$

$$V_{bc} = V_{b0} - V_{c0} = \frac{V_d}{2}(SF_b - SF_c) \tag{3.19}$$

$$V_{ca} = V_{c0} - V_{a0} = \frac{V_d}{2}(SF_c - SF_a) \tag{3.20}$$

,where

V_{ab}, V_{bc}, V_{ca} — are the line-to-line voltages in the inverter bridge,

V_d — is the supply voltage used,

SF_a, SF_b, SF_c — are the switching functions for each phase.

3.8 PC Application

To use the algorithms implemented into FPGA, and to get some of the data for visualization and analysis to our PC, the adequate VIs have to be programmed on the PC side too. To speed things up regarding this matter, the whole LabView project (FPGA part, RT part, PC part) was created from an initial template. This template already had prepared code from National Instruments in advance, implementing basic communication between the FPGA, Real Time and the PC itself. While developing and experimenting, the template has been slightly modified to suite every need of this project. Figure 3.33 shows which template was used as an initial one.

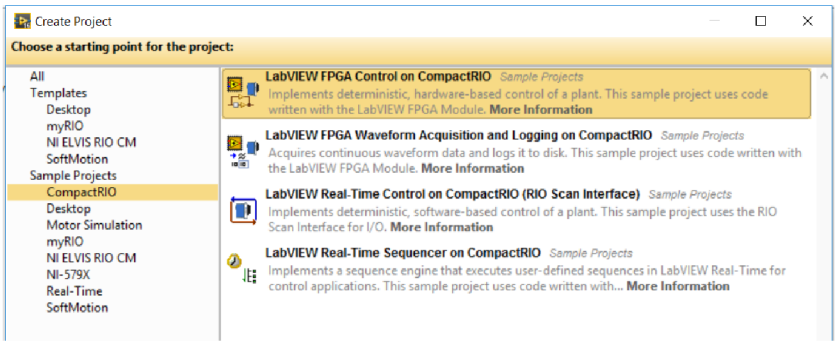


Figure 3.33: New project from template- LabView sample projects

Figure 3.34 shows the final front-panel of our application while the PID hall-sensored control is used. The front panel shows the measured RPM of the motor, generated torque by the motor, PID configuration settings, measurement loop periods and additional controls to configure some of the setting for sensor-less control.

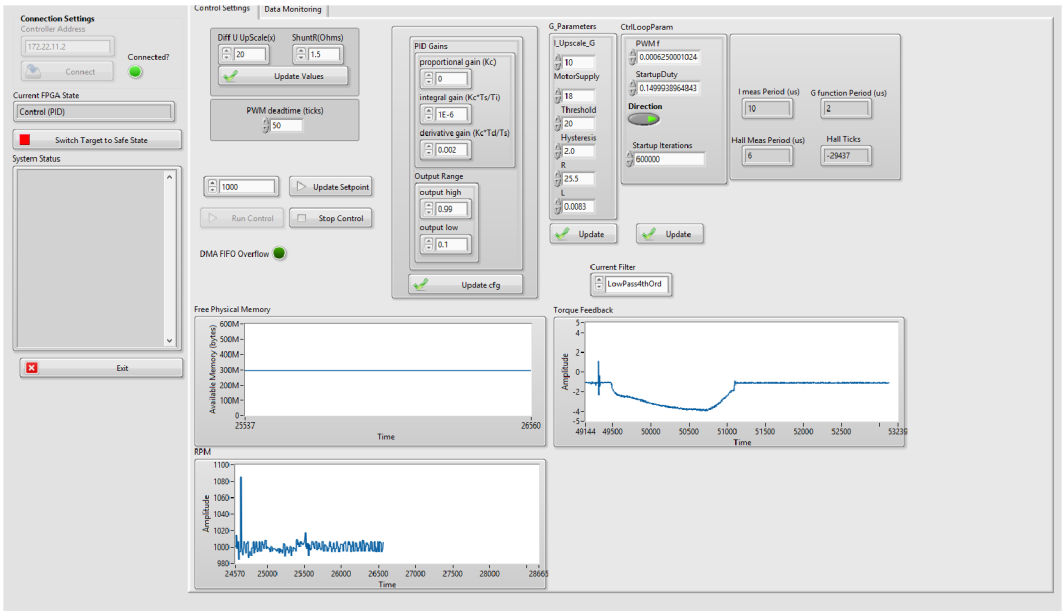


Figure 3.34: Front panel of the final application

The application at it's latest state can be used to tune the parameters of the PID controller for different motors and for different inverter designs. For the motor load can be generated using a contact-less method. This method (which uses eddy-currents) is not discussed here, however, the set-up is illustrated on figure 5.1.

Four

Conclusion

The aim of this thesis was to use the existing electronics from a rotary actuator with all of its components , except the microcontroller, connect it to NI MyRIO platform and use model based design approach and simulations to implement and try out BLDC motor control algorithms.

At first, a connection method had to be selected to easily realize detachable PWA connection to the cRIO platform. The only suitable method which facilitates easy PWA replacement happened to be the connection method which is widely used in factory in-circuit-testers: the use of so called pogo-pins. To realize this, a small prototype board was made manually. Later on this can be exchanged to a 3D printed fixture if needed. With pogo-pins, we can access any of the test-points on the PWA, in this way not just the inverter bridge is accessed, but for example the main power stage too which is responsible to supply the inverter itself. This way the performance of the BLDC motor can be tested with different power supply.

After the connection was realized between the NI MyRIO (later a different cRIO), at every step a new design challenge was unfolding. At the beginning it was assumed that for the motor current measurement the low-side DC-link sensing approach will be sufficient, later on it was realized that the sensor-less methods selected require in-line (phase) current measurements. For this measurement to be done a special integrated circuit has been selected from Texas Instruments which can tolerate high common mode voltage and is immune to PWM switching noise. Selecting this circuit resulted in a need of fast manual prototyping to try everything out. Successfully the prototype is working as expected, leaving plan B in reserve- plan B was to use a current measurement based on hall-effect sensors, which would have the advantage of insulating the measurement circuit from the motor phases. In the final realization information from the current measurement is not fed back to the PID controller, because the intention of this measurement was to implement sensor-less control using the presented methods.

Regarding the sensored control algorithm - based on hall effect sensor feedback - the six-step commutation method was implemented and tested. During implementation, the possibilities and limitations of FPGA programming were explored, amongst with FPGA inter-loop communication methods and FPGA-RT communication methods. It is worth mentioning that if one has the complete overview of these methods, and the design rules are followed, control algorithms can be swiftly implemented without recompiling the code 5 times a day (which is a quite long process, highly depending of code complexity). As for the details of this algorithm, the sampled hall

signals are converted to a number, which represents the mode of the inverter (from 1 to 6), then a look-up table is used to switch on and off the adequate transistors in the bridge. High-side FETs are connected to a PWM output, low-side FETs are connected to a standard digital output. Also, when no control is needed -e.g the motor needs to be switched off- there is a safe-state implemented which outputs zero to all transistors, switching them off. The simulation of this control technique was done at the beginning, but it was expected that the six-step commutation will be working just fine. This simulation was useful to study how the co-simulation works between LabView and MultiSim.

The realization of the sensor-less control was only partially done, however it was successfully simulated, with different approaches. The first selected method was also implemented into FPGA, but there was not enough time to debug everything and correct all the design errors which were made. The main realization issue of the first method was that the current measurement was not synchronized with the PWM signal generation, introducing error to the calculation of the function which would be used for commutation instant detection. The second method which was selected for sensor-less control uses an observer, which can estimate indirectly the BEMF signal from phase current measurements. Two types of BEMF observers were simulated.

Regarding the hardware platform, not only the MyRIO was used, everything was ported to 2 additional cRIOs. This was done due to resource usage issues on MyRIOs FPGA, and additional developmental needs. The portability between these platforms has been maintained.

Five

Attachments

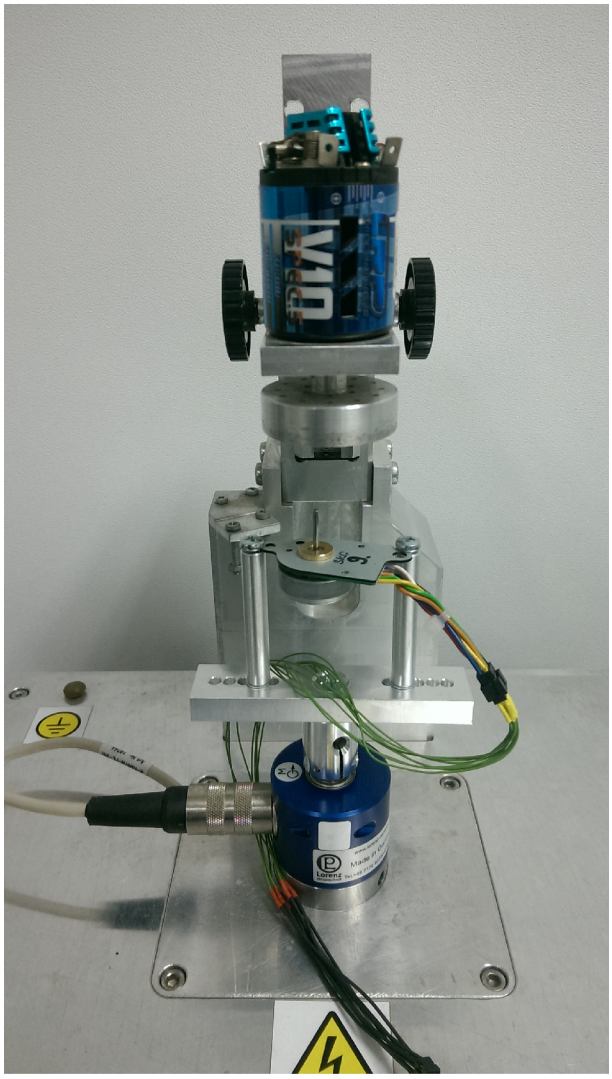


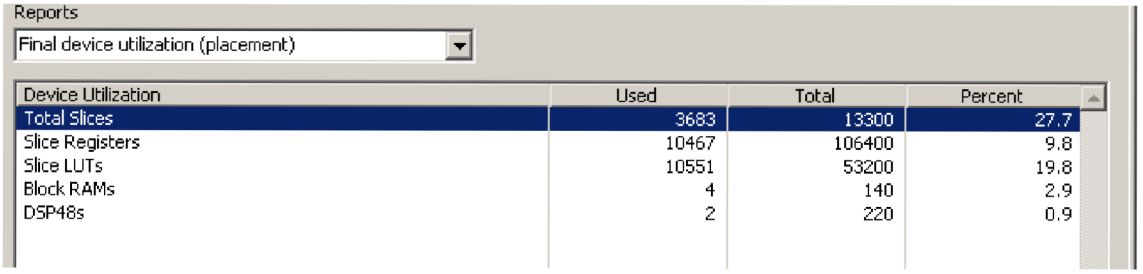
Figure 5.1: Test bench which allows the motor to be loaded with a contact-less method. Also a torque sensor is mounted.

Figure 5.2: NI MyRIO with wired necessary connections.

5.1 Compile Time and Resource comparison

```
Intel(R) Core(TM) i5-5300U CPU @ 2.30 GHz
RAM: 8 GB
64-bit Operating System

~13 min compile time

Project: BLDC_Control.lvproj
Target: FPGA Target 2 (RIO0, cRIO-9067)
Build Specification: PWM_FPGA_13_ResourceComparison
Top level VI: PWM_FPGA_13_ResourceComparison.vi
Compiling on local compile server
Compilation Tool: Xilinx Vivado 2015.4
Compilation Submitted: 4/4/2018 2:01 PM

Xilinx Options
------------------------------------------------

Implementation Strategy: Custom
Design Optimization Directive: Default
Placement Directive: Default
Physical Design Optimization Directive: Default
Routing Directive: Default

JobId: Z3JYV7R
Working Directory: C:\NIFPGA\compilation\BLDCControl...

~10 min compile time

Project: BLDC_Control.lvproj
Target: FPGA Target (RIO0, myRIO-1900)
Build Specification: PWM_FPGA_13_ResourceComparison
Top level VI: PWM_FPGA_13_ResourceComparison.vi
Compiling on local compile server
Compilation Tool: Xilinx Vivado 2015.4
Compilation Submitted: 4/4/2018 2:22 PM

Xilinx Options
------------------------------------------------
```

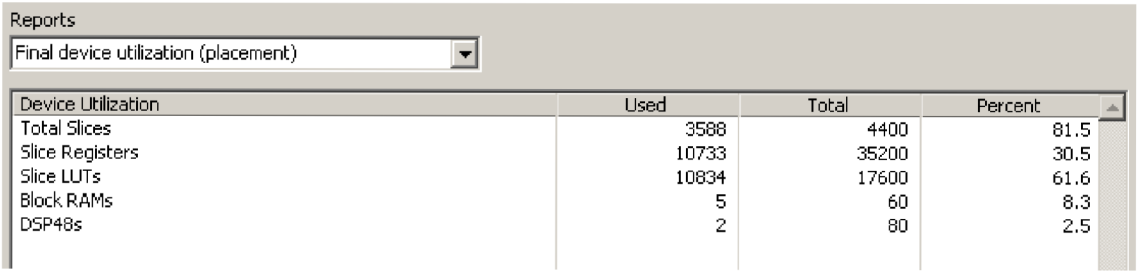
```
Implementation Strategy: Default
Design Optimization Directive: Default
Placement Directive: Default
Physical Design Optimization Directive: Default
Routing Directive: Default

JobId: p2Csb52
Working Directory: C:\NIFPGA\compilation\BLDCControl...

~15 min compile time

Project: BLDC_Control.lvproj
Target: FPGA Target (RIO0, cRIO-9073)
Build Specification: PWM_FPGA_13_ResourceComparison
Top level VI: PWM_FPGA_13_ResourceComparison.vi
Compiling on local compile server
Compilation Tool: Xilinx 14.7
Compilation Submitted: 4/4/2018 2:37 PM
Run when loaded to Fpga: FALSE
Maximum Fanout: 0

Xilinx Options
------------------------------------------------

Design Strategy: Balanced
Synthesis Optimization Goal: Speed
Synthesis Optimization Effort: Normal
Map Overall Effort Level: High
Place and Route Overall Effort Level: Standard

JobId: EeyCspy
Working Directory: C:\NIFPGA\compilation\BLDCControl...
```


Reports

Final device utilization (placement)

Device Utilization	Used	Total	Percent
Total Slices	3683	13300	27.7
Slice Registers	10467	106400	9.8
Slice LUTs	10551	53200	19.8
Block RAMs	4	140	2.9
DSP48s	2	220	0.9

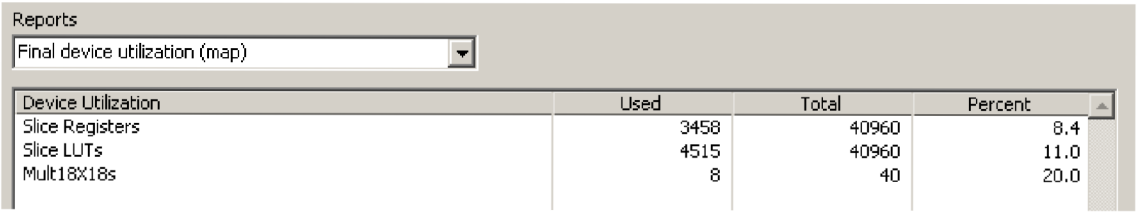
Figure 5.3: NI cRIO9067 final resource utilization


Reports

Final device utilization (placement)

Device Utilization	Used	Total	Percent
Total Slices	3588	4400	81.5
Slice Registers	10733	35200	30.5
Slice LUTs	10834	17600	61.6
Block RAMs	5	60	8.3
DSP48s	2	80	2.5

Figure 5.4: NI MyRIO final resource utilization


Reports

Final device utilization (map)

Device Utilization	Used	Total	Percent
Slice Registers	3458	40960	8.4
Slice LUTs	4515	40960	11.0
Mult18X18s	8	40	20.0

Figure 5.5: NI cRIO9073 final resource utilization

Bibliography

[1] S.K. Sul. *Control of Electric Machine Drive Systems*. IEEE Press Series on Power Engineering. Wiley, 2011.

[2] R. Krishnan. *Electric Motor Drives: Modeling, Analysis, and Control*. Prentice Hall, 2001.

[3] A. Visioli. *Practical PID Control*. Advances in Industrial Control. Springer London, 2006.

[4] K. Ogata. *Modern Control Engineering*. Instrumentation and controls series. Prentice Hall, 2010.

[5] Dan Harmon. Overcome the challenges of in-line phase motor current measurement. `http://www.electronicdesign.com/test-measurement/overcome-challenges-line-phase-motor-current-measurement`, 2016. [online].

[6] Texas Instruments. Ina240 high- and low-side,bidirectional, zero-drift,current-sense amplifier with enhanced pwm rejection. `http://www.ti.com/lit/ds/symlink/ina240.pdf`, 2016. [online].

[7] National Instruments. Custom development of hardware i/o. measurement and control with labview fpga module. `http://www.ni.com/white-paper/11162/pt/`, 2004. [online].

[8] Tae-Sung Kim, Byoung-Gun Park, Dong-Myung Lee, Ji-Su Ryu, and Dong-Seok Hyun. A new approach to sensorless control method for brushless dc motors. 6(4), 08 2008.

[9] The labview fixed-point data type part 1 – fixed-point 101.

[10] Floating-point implementation of a labview fpga pid control.

[11] Tae-Hyung Kim and M. Ehsani. Sensorless control of the bldc motors from near-zero to high speeds. *IEEE Transactions on Power Electronics*, 19(6):1635–1645, Nov 2004.

[12] Infineon Technologies AG. How to calculate and minimize the dead time requirement for igbts properly. `https://www.infineon.com/dgdl/Infineon-AN2007_04_Deadtime_calculation_for_IGBT_modules-AN-v01_00-EN.pdf?fileId=db3a30431a5c32f2011a5daefc41005b`, 2016. [online].

[13] Byoung-Kuk Lee and M. Ehsami. A simplified functional simulation model for three-phase voltage-source inverter using switching function concept. *IEEE Transactions on Industrial Electronics*, 48(2):309–321, Apr 2001.

[14] C. Xia. *Permanent Magnet Brushless DC Motor Drives and Controls*. Wiley, 2012.

[15] José Carlos Gamazo-Real, Ernesto Vázquez-Sánchez, and Jaime Gómez-Gil. Position and speed control of brushless dc motors using sensorless techniques and application trends. *Sensors*, 10(7):6901–6947, 2010.

[16] H. Kim, J. Son, and J. Lee. A high-speed sliding-mode observer for the sensorless speed control of a pmsm. *IEEE Transactions on Industrial Electronics*, 58(9):4069–4077, Sept 2011.

[17] T. H. Kim, H. W. Lee, and M. Ehsani. Advanced sensorless drive technique for multiphase bldc motors. In *30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004*, volume 1, pages 926–931 Vol. 1, Nov 2004.

[18] E Kaliappan and Chinna Chellamuthu. Modelling, analysis and simulation of permanent magnet brushless dc motors for sensorless operation. 61, 11 2012.

[19] Mitesh B. Astik, Praghnesh Bhatt, and Bhavesh R. Bhalja. Analysis of sensorless control of brushless dc motor using unknown input observer with different gains. 68, 01 2017.

[20] S. R. Snehasree. Modeling of permanent magnet bldc motor using state space analysis. *International Journal of Innovative Research and Development*, 2(6), 2013.

[21] U. Vinatha, S. Pola, and K. P. Vittal. Simulation of four quadrant operation and speed control of bldc motor on matlab / simulink. In *TENCON 2008 - 2008 IEEE Region 10 Conference*, pages 1–6, Nov 2008.

[22] BYOUNG-KUK LEE and MEHRDAD EHSANI. Advanced simulation model for brushless dc motor drives. 31:841–868, 09 2003.

[23] B. K. Lee and M. Ehsani. Advanced bldc motor drive for low cost and high performance propulsion system in electric and hybrid vehicles. In *IEMDC 2001. IEEE International Electric Machines and Drives Conference (Cat. No.01EX485)*, pages 246–251, 2001.

[24] A. Tashakori, M. Ektesabi, and N. Hosseinzadeh. Modeling of bldc motor with ideal back-emf for automotive applications. In *Proceedings of the World Congress on Engineering 2011, WCE 2011*, volume 2, pages 1504–1508, 2011.

[25] Yong Zhou, H.-K Jiang, Q.-X Zhou, and Q.-J Zhang. A novel method for modeling and simulation of brushless dc motor with kalman filter. 163:305–314, 01 2012.

[26] Tae-Hyung Kim, Hyung-Woo Lee, and Mehrdad Ehsani. State of the art and future trends in position sensorless brushless dc motor/generator drives. In *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.*, pages 8 pp.–, Nov 2005.

[27] T. Kim and M. Eshani. Advanced sensorless drive technique for brushless dc motors, 2004. U.S. Patent 60/438,949.

[28] Milan Rajne. Have you considered using back-emf observers for sensorless speed estimation? `https://e2e.ti.com/blogs_/b/motordrivecontrol/archive/2014/03/14/\have-you-considered-using-back-emf-observers-for\-sensor-less-speed-estimation`, 2014. [online].

Nomenclature

L_A self-inductance of phase A.

M_{AB}, M_{AC}, M_{BC} phase mutual inductance.

R_x phase resistance, in which subscript x denotes phase A, B and C.

$\psi_{pm}\theta$ PM flux linkage.

θ relative angular displacement between rotor and stator; rotor position angle.

$e_{\psi x}$ phase-iduced EMF.

i_x phase current, in which subscript x denotes phase A, B and C.

u_x phase voltage, in which subscript x denotes phase A, B and C.