VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VYUŽITÍ FPGA PRO ŘÍZENÍ A MODELOVÁNÍ BLDC MOTORU

FPGA APPLICATION FOR CONTROL AND MODELLING OF BLDC MOTOR

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. VÁCLAV SOVA

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ROBERT GREPL, Ph.D.

SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Václav Sova

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Mechatronika (3906T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Využití FPGA pro řízení a modelování BLDC motoru

v anglickém jazyce:

FPGA application for control and modelling of BLDC motor

Stručná charakteristika problematiky úkolu:

S rozvojem automatického generování HDL kódu pro programovatelná hradlová pole se rozšiřuje jejich použití i pro inženýrské přístupy jako HIL a RCP (rychlé prototypování řídicích systémů). Výhodou FPGA proti klasickým systémům s procesorem je možnost dosáhnout mnohem vyšší rychlosti výpočtu, vyšší vzorkovací frekvence. Je tedy možné například simulovat zvlnění proudu při řízení pomocí PWM s vysokou frekvencí.

Práce by se měla zaměřit na ty problémy okolo BLDC motorů, kde by se měly projevit výhody programovatelných hradlových polí nebo kde klasický přístup s proceserem není možný.

Práce bude probíhat na experimentálním BLDC standu a řízení se bude provádět pomocí modulárního systému dSPACE s FPGA kartou DS5203. HDL kód bude generován automaticky ze Simulinku pomocí System Generatoru od firmy Xilinx.

Cíle diplomové práce:

1) Proveďte rešerši v oblastech:
- studium práce [1]
- publikace na IEEE, ScienceDirect apod. v oblasti aplikace FPGA pro modelování a řízení BLDC motorů
2) Implementujte sensorové a bezsensorové řízení BLDC motoru na FPGA a ověřte chování na BLDC standu.
3) Navrhněte a implementujte řízení v tzv. degradovaném módu (výpadek jednoho ze tří hallových sensorů). Ověřte vlastnosti algoritmu při nízkých otáčkách.
4) Implementujte model BLDC motoru pro HIL včetně příslušných periferií na FPGA.

Seznam odborné literatury:

[1] DP Križan, 2012, FSI VUT v Brně
[2] www.xilinx.com
[3] www.dspace.com

Vedoucí diplomové práce: doc. Ing. Robert Grepl, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2012/2013.

V Brně, dne 19.11.2012

L.S.

prof. Ing. Jindřich Petruška, CSc. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c. Děkan fakulty

Abstrakt

Práce se zabývá řešením úkolů v oblasti řízení BLDC motorů s využitím programovatelných hradlových polí (FPGA). S využitím modulárního HW dSPACE s FPGA kartou jsou řešeny problémy jako senzorové i bezsenzorové řízení BLDC motoru, real-time simulace tohoto motoru a řízení v tzv. degradovaném módu. Design FPGA je tvořen pomocí nástroje System Generator for DSP od firmy Xilinx. Vedlejším cílem práce je ukázat, že s rozvojem vysokoúrovňových nástrojů pro design FPGA je implementace algoritmů pro FPGA poměrně rychlá a efektivní a nevyžaduje dlouholeté zkušenosti a znalosti s programovatelnými hradlovými poly. Implementací algoritmů na FPGA místo procesorů získáme mnoho výhod, především v rychlosti zpracování a nízké latenci.

Klíčová slova

BLDC, EC motor, bezkartáčový motor, FPGA

Abstract

Thesis deals with the challenges in the field of BLDC motors control with the utilization of Field Programmable Gate Arrays (FPGA). Using the modular dSPACE hardware with the FPGA board, these issues are solved: sensored and sensorless control, real-time simulation of BLDC motor and control of BLDC motor in degraded mode. FPGA design is made using the System Generator for DSP from Xilinx. The side effect of work is to show that with the expansion of high-level tools for FPGA design, the implementation of algorithms for FPGA is relatively quick and efficient and does not require years of experience and big knowledge of field programmable gate arrays. The implementation of algorithms on FPGA instead of processors brings many advantages, in the first place the high speed processing and low latency.

Keywords

BLDC, EC motor, brushless DC motor, FPGA

Bibliografická citace práce

SOVA, V. *Využití FPGA pro řízení a modelování BLDC motoru.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 67 s. Vedoucí diplomové práce doc. Ing. Robert Grepl, Ph.D..

Čestné prohlášení

Čestně prohlašuji, že jsem tuto práci vypracoval samostatně s použitím uvedené literatury a pod vedením vedoucího DP.

Václav Sova, Brno, 2013

Poděkování

Na tom to místě bych rád poděkoval především vedoucímu diplomové práce Doc. Ing. Robertu Greplovi Ph.D za cenné rady a možnost pracovat v dobře vybavené laboratoři na moderním zařízení. Také bych chtěl poděkovat kolegům v laboratoři za přátelské pracovní prostředí a inspiraci. Dále mé díky patří rodičům za podporu během studia.

Seznam obrázků

Seznam tabulek

Obsah

1 Úvod

Popularita BLDC motorů neustále roste. Tyto motory jsou stále častěji nasazovány do aplikací v automobilovém a leteckém průmyslu, do průmyslových strojů a zařízení a do domácích spotřebičů. Motor byl vytvořen jako náhrada stejnosměrného komutátorového motoru. Při nepříliš zvýšených výrobních nákladech odstraňuje hlavní nedostatky komutátorových motorů. Odstraňuje komutátor a s tím spojené problémy s opotřebením, nutnou údržbou, nízkou životností a nižší účinností.

V nízkonákladových aplikacích může řízení motoru obstarat jeden integrovaný obvod, který v sobě obsahuje i tranzistorový měnič a elektronika je velmi jednoduchá a levná. Řízení pohonů střední a vyšší cenové kategorie obstarává většinou program naprogramovaný v mikrokontroléru. U pohonů, kde nehraje cena hlavní roli, ale soustřeďujeme se i na jiné parametry jako vysoká účinnosti, nízké zvlnění momentu, spolehlivost, stále probíhá vývoj a může být dosaženo určitého zlepšení.

Určitou alternativou mikrokontrolérům mohou být programovatelná hradlová pole (FPGA), která nabízejí určité výhody. Především vysokou výpočetní rychlost, možnost paralelizace a nízkou latenci. S rozvojem vysokoúrovňových nástrojů pro design FPGA se jejich použití dostává k mnohem širšímu okruhu zájemců, především z řad inženýrů. Dosud mohli navrhovat funkcionalitu FPGA a těžit z jeho výhod především hardwarový vývojáři s dlouholetou praxí. Plně uživatelsky konfigurovatelné FPGA se dostavají do zařízení pro Rapid Control Prototyping (RCP) a Hardware in the Loop (HIL) a může je konfigurovat běžný uživatel s pouze minimem znalostí o FPGA.

2 Rešerše

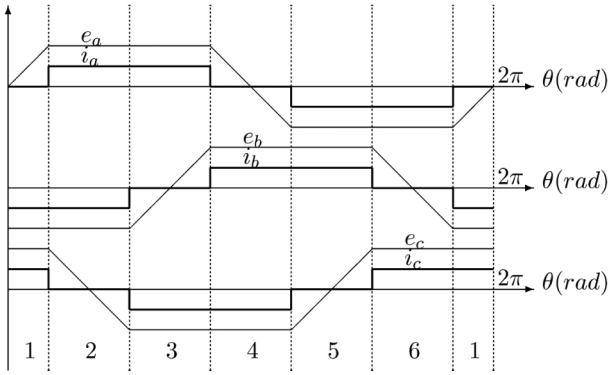
2.1 BLDC motory a jejich řízení

Ačkoliv nejvíce používaná anglická zkratka BLDC (Brushless Direct Current) motor značí spojitost se stejnosměrným motorem, motor se řadí do kategorie synchronních strojů. Další používaná anglická zkratka je EC (Electronically Commutated) motor. V české literatuře je motor znám pod pojmem elektronicky komutovaný motor nebo stejnosměrný bezkartáčový motor.

2.1.1 Konstrukce BLDC motorů

Konstrukčně je motor velmi podobný synchronnímu motoru s permanentními magnety. Ve statoru složeném z plechů je v drážkách vloženo trojfázové vinutí většinou zapojené do hvězdy. Statorové drážky jsou často zešikmeny o jednu drážkovou rozteč z důvodu snížení reluktančních momentů.

Rotor obsahuje permanentní magnety, které jsou buď nalepeny na povrchu rotoru a nebo jsou vsazeny do rotoru s koncentrací magnetického toku pólovými nástavci. Jako magnety se používají materiály na bázi vzácných zemin (samarium - kobalt, neodym - železo - bór) nebo pro levnejší aplikace ferity.



Obrázek 2.1: Průběhy proudů a indukovaného napětí v ideálním BLDC motoru[1]

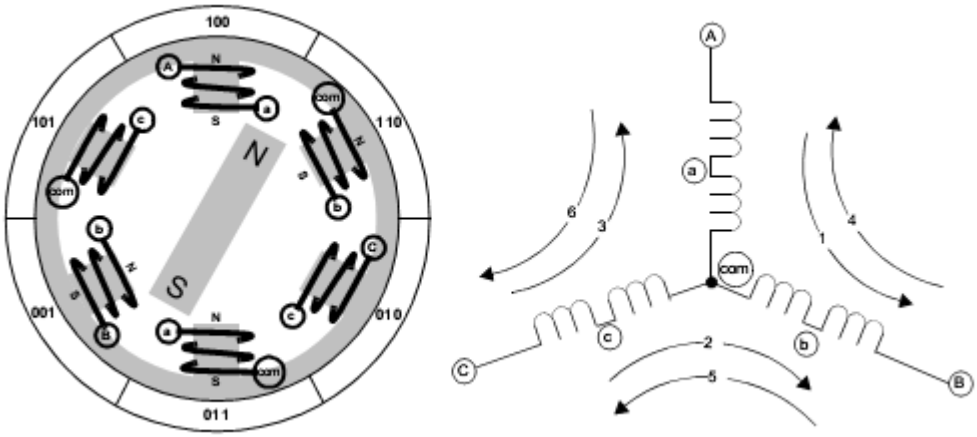
Od klasického synchronního stroje se BLDC motor liší především jiným rozložením vinutí v drážkách a tím i jiným průběhem indukovaného napětí. Klasický synchronní motor má sinusové průběhy indukovaného napětí a musí být také napájem napětím o sinusovém průběhu. Ideální BLDC motor má průběhy indukovaného napětí lichoběžníkové (obr.: 2.1) a bývá napájen konstantním napětím, které

je přepínáno do jednotlivých fází v určitém pořadí.

Používají se následující typy vinutí:

- jedna drážka na pól a fázi
- dvě drážky na pól a fázi
- segmentový stator s vinutím typu jedna cívka na pól a fázi

Tyto typy vinutí zajišťují, aby se indukované napětí více či méně blížilo ideálnímu lichoběžníkovému průběhu. Zvláštní typ vinutí je se segmentovým statorem. Předchozí dva typy mají takové rozložení vinutí, že dvě strany každé fázové cívky jsou od sebe vzdáleny 180° elektrických. Díky této vlastnosti dochází stejně jako u sychronního motoru ke křížení cívek v čelech statoru a nárůstu osové délky stroje. Dále je velká nevyužitá délka vodičů a dochází k větším Joulovým ztrátám v mědi. Segmentový stator (obr.: 2.2) tyto nedostatky odstraňuje. Jedná se o samostatné cívky nasunuté na trny statorového jha. Další výhodou je také jednodušší navíjení cívek.



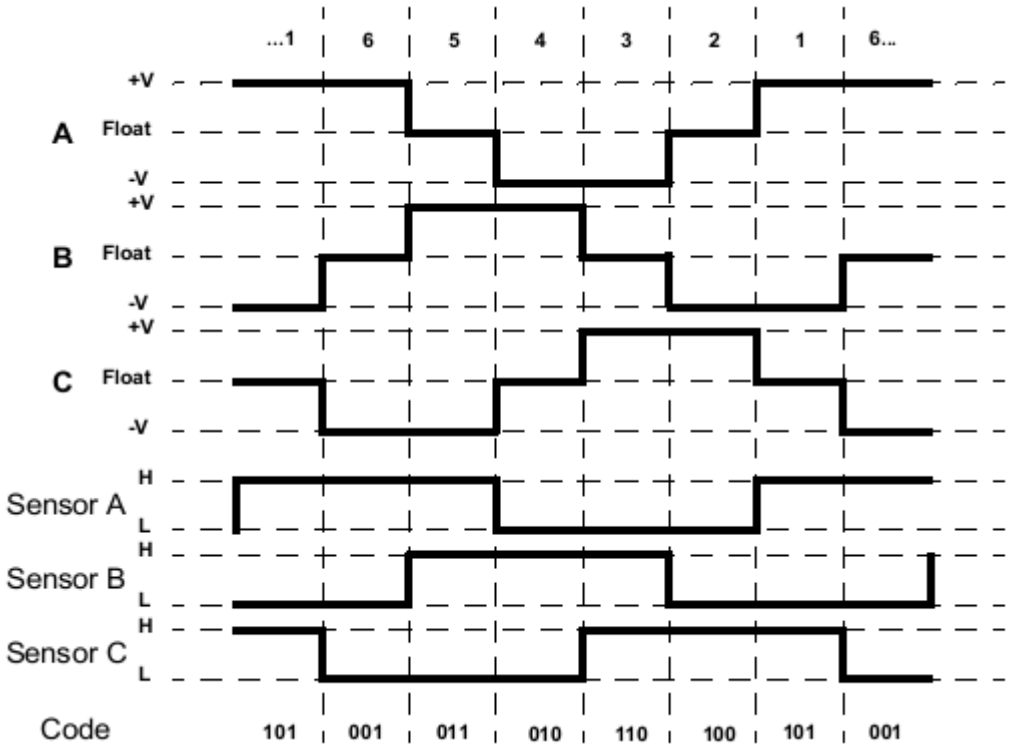
Obrázek 2.2: Segmentové vinutí statoru BLDC motoru[2]

2.1.2 Řízení BLDC motorů

Jak je patrné z grafu na obrázku 2.1, je možné jednu elektrickou otáčku BLDC motoru rozdělit na šest úseků po 60°el. Indukované napětí každé fáze je po dobu 120°el. konstantní a poté se během 60°el. lineárně změní na opačnou polaritu. V konstantním úseku je třeba danou fázi napájet konstantním napětím, což znamená konstantní proud. Během úseku, kde daná fáze mění polaritu, je daná fáze nenapájená

a říkame, že komutuje. V každém okamžiku jsou napájeny dvě fáze motoru a třetí je plovoucí.

Pro správné řízení tedy stačí vědět, ve kterém z šesti úseků se rotor nachází a podle toho napájet svorky motoru. Nepotřebujeme tedy senzor otáček s vysokým rozlišením jako klasický synchronní motor. Nejčastěji se používá tří hallových senzorů umístěných na statoru natočených vzájemně o 120°el. Proti hallovým senzorům je umístěn přímo rotor s magnety nebo zvláštní magnetický kroužek na hřídeli. Binární výstupy z hallových senzorů potom určují, ve které poloze se rotor nachází, a podle toho přepínáme napětí na svorky motoru (obr.: 2.3). Toto řízení využívající hallových senzorů se označuje jako senzorové.



Obrázek 2.3: Výstupy hallových senzorů a napájecí napětí svorek BLDC motoru[2]

Existují metody řízení, které nepotřebují senzory polohy. Většinou jsou založeny na měření napětí právě komutující fáze a detekcí průchodu tohoto napětí nulou. Z toho je potom možné rekonstruovat polohu. Metod je více a jsou více či méně komplikované. Mají ale společné to, že neumožňují spolehlivý běh motoru v nízkých a nulových otáčkách z toho důvodu, protože v nízkých otáčkách je indukované napětí malé. Velkou roli hraje šum a měření napětí není přesné. Tyto metody řízení se označují jako bezsenzorové.

2.1.3 Výhody a nevýhody BLDC motorů

Výhodou BLDC motorů je vysoká výkonová hustota (poměr výkon/velikost a výkon/hmotnost). Při stejné velikosti je výkon BLDC motoru vyšší než u klasického synchronního stroje. Přitom cena je nižší především z důvodu absence přesného senzoru otáček a jednoduššího řízení. Proti stejnosměrnému motoru je výhoda v absenci kartáčů a komutátoru. To znamená bezúdržbovost a dlouhou životnost.

Podstatnout nevýhodou BLDC motorů je velké zvlnění momentu. Teoreticky by BLDC motor měl mít konstantní velikost momentu. V jednotlivých úsecích elektrické otáčky se přepínají oblasti s konstantním proudem a konstantním napětím, tedy i výkon a moment by měl být konstatní. Z důvodu nenulové indukčnosti vinutí nedochází při přepínání fází ke skokovému nárůstu a poklesu proudu, ale změny proudu jsou pozvolné a moment není konstantní.

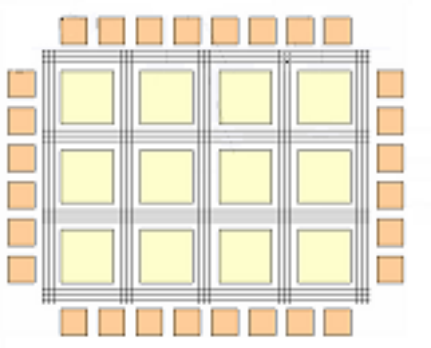
Podrobnější vlastnosti motorů je možno nalézt v [3], [4], literatura [5] se potom zabývá přímo porovnáním BLDC motorů s klasickým sychronním strojem.

2.2 FPGA

Programovatelná hradlová pole, anglicky Field Programmable Gate Array (FPGA). Anglický název vznikl spojením dvou sousloví:

- *Field Programmable* - funkcionalita je definnovává "v poli"uživatelem.
- *Gate Array* - jedná se o dvoudimenzionální matici hradel, základních stavebních prvků.

Jedná se o polovodičový čip, který je založen na dvoudimenzionální matici konfigurovatelných logických bloků, které jsou vzájemně propojovány konfigurovatelnými propoji. Chování tohoto čipu, funkcionalita, je dána vzájemným propojením a nastavením jednotlivých logických bloků. Toto nastavení se vytváří (programuje) pomocí grafické reprezentace logického obvodu nebo pomocí tzv. jazyků popisujících hardware (HDL - Hardware Description Language). Nejpoužívanější HDL jazyky jsou VHDL a Verilog. U FPGA nemluvíme o programování, ale spíše o designu, vytváříme totiž popis zapojení hardwaru.



Obrázek 2.4: Struktura FPGA[6]

Výhodou FPGA je především jejich velká flexibilita a rychlost ve srovnání s procesory. Na procesoru se program provadí sekvenčně instrukce po instrukci. Na FPGA ale může být vytvořeno skutečné paralelní zpracování, jsme omezeni pouze

velikostí čipu. Při práci na FPGA máme také volnou ruku ohledně bitové šířky operandů. Procesory mají pevnou architekturu (8, 16, 32, 64 bitová architektura), kdežto na FPGA můžeme pracovat klidně s 11-ti bitovými operandy. Na druhou stranu na FPGA je velmi obtížná a na zdroje FPGA velmi náročná implementace výpočtů s plovoucí desetinnou čárkou.Výsledný design na FPGA by měl být také více spolehlivý, protože se vlastně jedná o účelově propojený hardware.

Principielní obdoba FPGA jsou zákaznické integrovaný obvody (ASIC - Application-Specific Integrated Circuit). Od FPGA se ale liší tím, že funkcionalita jim je dána již při výrobě. Jejich cena, především při velkém počtu odebraných kusů, je příznivější. Výhodou FPGA je ale kratší doba vývoje a návrhu, který si výrobce zařízení může provést sám.

2.2.1 Architekura FPGA

FPGA se skládá z globální propojovací matice rozdělující plochu čipu na stejné oddíly, ve kterých se nachází lokální propojovací matice a logický blok. Kombinace obou propojovacích matic umožňuje propojit libovolně vstupy a výstupy jednotlivých logických bloků. Tyto logické bloky obsahují v závislosti na modelu FPGA více identických celků (aglický název je slice). Základním stavebním kamenem každého tohoto celku (slice) je look-up tabulka a klopný obvod typu D. N-vstupová Look-up tabulka (LUT) umožňuje vytvářet kombinační logiku. Počet vstupů Look-up tabulky je opět závislý na modelu FPGA. Klopný obvod typu D je vlastně jednobitová pamět (registr) umožňující vytvářet sekvenční logiku.

V průběhu práce s FPGA se ukázalo, že násobení je velmi často používaná operace, která ale zabírá příliš mnoho zdrojů FPGA. Potřebné zdroje (zabraná plocha) FPGA při násobení narůstají s čtvercem bitové šířky operandů. Proto ještě většina FPGA čipů obsahuje vestavěné násobičky, které šetří zdroje FPGA. Tyto vestavěné násobičky, v terminologii firmy Xilinx DSP48x slices [6] , obsahují až 25x18-bitovou násobičku, předsčítačku a 48-bitový akumulátor. Bloky RAM pamětí se také zavedly to struktury FPGA za cílem ušetřit zdroje.

Pro představu uvádíme parametry FPGA použitého na kartě DS5203 v dSPACE [7]:

Tabulka 2.1: Xilinx Virtex-5 XC5VSX95T [8]

Počet slice	14720
Počet LUT	58880
Počet registrů	58880
Počet DSP48 slice	640
Velikost RAM paměti	8784 kB

2.2.2 Vysokoúrovňové nástroje pro design FPGA

Ve srovnání s procesory byl návrh funkcionality pro FPGA s využitím grafického popisu i HDL jazyků mnohem komplikovanější. Vývojový cyklus byl mnohem delší, bylo třeba složitě nastavovat simulátory kódu, každý překlad kódu pro FPGA trvá na běžném osobním počítači desítky minut až hodiny. S rozvojem vysokoúrovňových nástrojů pro design FPGA se tato technologie nabízí širšímu počtu zájemců, i těm kteří se problematice FPGA stoprocentně nevěnují.
Nejznámější vysokoúrovňové nástroje pro design FPGA jsou:

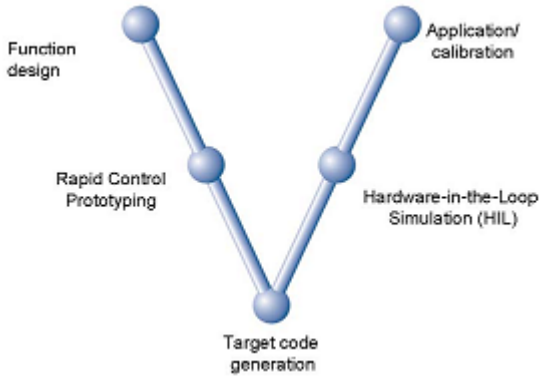
- **Xilinx System Generator for DSP** [9] - Rozšiřující knihovna pro Simulink, obsahuje velké množství funkcí. Podporuje FPGA od firmy Xilinx a umožňuje jejich nastavení a specifikování vstupů a výstupů. Jedním stisknutím tlačítka je možno vygenerovat konfigurační soubor pro FPGA. Funguje zda také klasická simulace jako v běžném Simulinku a hardwarová kosimulace.
- **National Instruments LabVIEW FPGA Module** [10] - Rozšiřující modul do LabVIEW. Slouží především ke konfigurování FPGA na zařízeních od National Instruments (RIO Architektura a jiné). Je dostupná omezená paleta funkcí v LabVIEW.
- **MathWorks HDL Coder** [11] - Generuje obecný VHDL nebo Verilog kód z funkcí Matlabu, modelů v Simulinku a Stateflow grafů.

2.3 Hardware in the Loop - HIL

Vývoj ECU většinou probíhá v takzvaném V-cyklu (obr.: 2.5). Hlavní body tohoto V-cyklu jsou:

- **Function design** - Specifikování požadavků, testování funkcí na modelu.
- **Rapid Control Prototyping** - Testování funkcí na skutečném zařízení.
- **Target code generation** - Tvorba kódu pro ECU, často s využitím nástrojů pro automatické generování kódu.
- **Hardware-in-the-Loop Testing** - Testování ECU a její funkce na simulované soustavě,prostředí.
- **Aplication/calibration** - Závěrečné testování na skutečném zařízení.

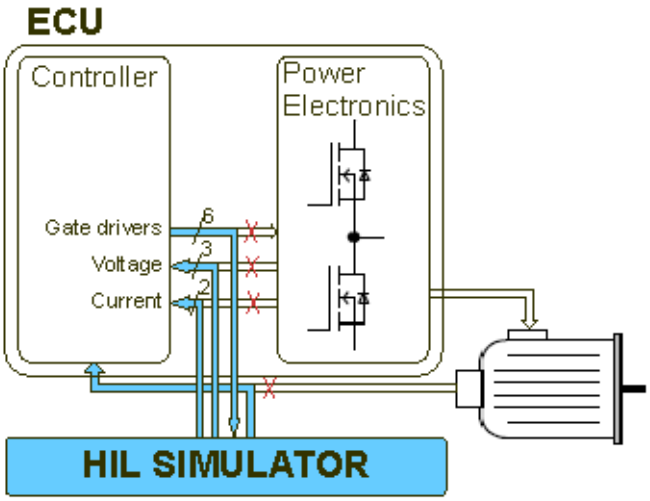
HIL simulace a testování má v tomto cyklu vývoje ECU důležitou roli. Při tomto testování je ECU připojena svými vstupy a výstupy k simulované soustavě, simulovanému prostředí. Proti klasickému testování se skutečným zařízením je zde mnoho výhod. Možnost automatizování testů, v kratším čase je možno otestovat více funkcí ECU, zaručení opakovatelnosti testů, je možné jednoduše simulovat chybové



Obrázek 2.5: Vývojový V-cyklus ECU[12]

stavy signálů a testovat simulovanou soustavu ve stavech, které by ve skutečnosti mohly vést ke zničení soustavy, ohrožení bezpečnosti.

Pro simulaci prostředí je důležitý specializovaný hardware se vstupy a výstupy, který má poměrně malou latenci a výpočetní výkon umožňující deterministický běh modelu soustavy, prostředí v reálném čase. Scénář pro naší HIL simulaci BLDC motoru je na obrázku 2.6.



Obrázek 2.6: Testovací scénář pro ECU BLDC motoru

2.4 Shrnutí několika prací týkajících se tématu diplomové práce

Bezsenzorové řízení BLDC motoru, Diplomová práce [13]

Ucelená práce, která se zabývá problematikou BLDC motorů. Je zde podrobně uveden princip BLDC motorů, jeho matematický popis a simulační model v prostředí Matlab/Simulink. Dále jsou v práci uvedeny metody senzorového i bezsenzorového řízení a rozběh motoru v otevřené smyčce. Většina těchto metod je implementována a ověřena na jednodeskovém HW dSPACE založeném na procesoru. Za zmínku stojí vzájemné porovnání několika metod bezsenzorového řízení a dosažené parametry při jejich implementaci. Také se práce zabývá poměrně speciálním úkolem, zachováním funkce pohonu i při poruše hallova senzoru (tzv. degradovaný mód).

Real-Time Simulation of BLDC Motors for Hardware-in-the-Loop Applications Incorporating Sensorless Control [14]

Práce zabývající se HIL simulací BLDC motoru. Ovšem nejedná se o klasickou HIL simulaci na signálové úrovni, ale o simulaci na úrovni výkonové. Testovaná jednotka včetně výkonového měniče je připojena k umělým zátěžím, které na výkonové úrovni simulují BLDC motor. Tento typ simulace se spíše označuje jako Power HIL (PHIL). V článku jsou přehledně ilustrovány jednotlivé přístupy k testování ECU, je zde zdůvodněna nutnost využití FPGA pro simulaci elektrické části BLDC motoru. Také je zde zběžně představena elektronika použitá jako umělá zátěž simulující chování motoru. RT model BLDC motoru běží rozdělen na FPGA a procesoru na podobném zařízení (dSPACE), které máme k dispozici my. Tento článek je poměrně stručný na podrobnější informace, ale inspiroval nás pro rozdělení modelu mezi FPGA a procesor.

Application Characteristics of Permanent Magnet Synchronous and Brushless dc Motors for Servo Drives [5]

Jak je z názvu patrné, práce se zabývá porovnáním BLDC motoru a synchronního motoru s permanentními magnety, které mají konstrukčně k sobě velmi blízko. Jsou zde probrány podobnosti a rozdíly těchto motorů z kontrukčního hlediska a porovnání je také provedeno v mnoha parametrech. Například hledisko ceny, výkonové hustoty, rozsahů otáček, poměr točivého momentu k velikosti, možnosti brždění, parametry měniče, ztráty a tepelné vlastnosti. Jednotlivé parametry a vlastnosti jsou probrány poměrně do hloubky a závěry často zdůvodněny matematickými důkazy. Práce je velmi podrobná.

Real-Time Simulation of a Complete PMSM Drive at 10μS Time Step [15]

Článek se zabývá HIL simulací celého pohonu se synchronním motorem. HIL simulace probíhá na signálové úrovni a je simulován motor, měnič i jeho usměrňovač. V práci je zdůrazněna potřeba krátkého výpočetního kroku pro simulaci elektrických částí. Jak je z názvu práce patrné, bylo dosaženo výpočetního kroku 10 μS. Takto krátkého výpočetního kroku bylo dosaženo rozdělením modelu na dva procesory Intel Pentium-4 taktovaných na $2,8\ GHz$ na HW RT-LAB. Práce je poměrně staršího data (rok 2005). Dosažený výpočetní krok byl v té době na procesorových systémech unikátní, ale i v dnešní době nebývá o mnoho překonáván. Spíše stručná přehledová práce, která ale ukazuje hranice běžných procesorových simulátorů.

An FPGA Implementation of a Brushless DC Motor Speed Controller[16]

Článek popisuje implementaci jednoduchého otáčkového PI regulátoru BLDC motoru na FPGA. Jedná se nízkonákladový kontrolér, který je založen na low-cost FPGA Xilinx Spartan-3E. Jsou zde podrobněji popsány jednotlivé funkční bloky designu, který je proveden pomocí System Generatoru od Xilinx. I s tímto jednodychým FPGA se celý design provede v jednom cyklu FPGA a výpočetní krok je 20 nS. Článek je podrobný, ale cíl práce je poměrně jednoduchý.

FPGA Based BLDC Motor Current Control with Spectral Analysis[17]

Článek se věnuje implementaci dvou metod momentového řízení BLDC motoru na FPGA v zařízení od National Instruments. Design tedy probíhá pomocí LabVIEW a jeho FPGA modulu. Pro regulaci proudu se používá klasický PI regulátor a tzv. sliding mode regulátor. V práci je podrobně popsána funkce obou regulátorů a je zde také poměrně dobře popsána a vysvětlena implementace na FPGA v LabVIEW. Mnoho obrázků grafického designu v LabVIEW s vysvětlením je zobrazeno. Na závěr práce jsou porovnány obě metody z hlediska frekvenčního spektra průběhů proudů.

3 Rozbor zadání a stanovení cílů řešení

Zadání diplomové práce obsahuje tyto body:

1. Proveďte rešerši v oblastech:
   - studium práce [13]
   - publikace na IEEE, ScienceDirect apod. v oblasti aplikace FPGA pro modelování a řízení BLDC motorů
2. Implementujte sensorové a bezsensorové řízení BLDC motoru na FPGA a ověřte chování na BLDC standu.
3. Navrhněte a implementujte řízení v tzv. degradovaném módu (výpadek jednoho ze tří hallových sensorů). Ověřte vlastnosti algoritmu při nízkých otáčkách.
4. Implementujte model BLDC motoru pro HIL včetně příslušných periferií na FPGA.

Rešerši se věnuje předchozí kapitola, která má za úkol čtenáře stručně seznámit s problematikou BLDC motorů, programovatelných hradlových polí a seznámit s několika pracemi, které nám budou inspirací.

Implementace senzorového řízení bude první úkol, kterým se budeme zabývat. Ze všech úkolů je tento nejjednodušší a umožní nám se podrobněji seznámit s použitým HW a SW a vytvořit základní stavební bloky designu, které se budou opakovat v dalších částech. Pro bezsenzorové řízení bude použita metoda, u které se dá nejvíce využít výhod, které nabízí FPGA. Bude pokud možno vybrána taková metoda, která při implementaci na mikrokontrolérech naráží na omezení, které by FPGA mohlo odstranit. Nebo by alespoň mohlo dojít ke zlepšení parametrů řízení.

Řízení BLDC motoru v degradovaném módu, tj. s poruchou jednoho Hallova senzoru je poměrně neprobádaný požadavek, který má ale své opodstatnění v kritických aplikacích. Zabývá se jim literatura [13], kde však tato metoda trpí mnoha nedostaky především proto, že je implementována na procesoru. Při implementaci této nebo podobné metody na FPGA určitě dosáhneme výrazného zlepšení.

Poslední úkol, HIL simulace BLDC motoru vyžaduje model motoru schopný běžet v RT. Při tomto úkolu se naplno ukáží výhody implementace na FPGA, protože korektní běh RT modelu na běžných procesorových systémech je nemožný. K dispozici máme rozpracovaný model BLDC motoru z [13], který musíme vylepšit a poté implementovat na FPGA. Díky literatuře [14], která se zabývá podobným

úkolem na podobném zařízení jsme dospěli k rozdělení modelu motoru mezi FPGA a procesorovou část.

Vedlejším produktem práce je zjistit, odhadnout, jak komplikované by bylo vytvořit ECU pro řízení BLDC motorů založenou na FPGA místo mikrokontroléru. Které úskálí a které výhody to přináší. Technologie FPGA není v současné době o moc nákladnější než technologie založená na mikrokontrolérech a přináší zajímavé přednosti, které by mohly ospravedlnit vyšší cenu. Například certifikace systémů pro kritické aplikace založených na FPGA by mohla být jednodušší než u mikrokontrolérů.

4 Použité zařízení

V této kapitole bude podrobněji popsáno nejdůležitější zařízení použité v průběhu práce. BLDC motor, výkonový třífázový měnič pro jeho buzení a řídicí HW od firmy dSPACE.

4.1 BLDC motor

Byl použit motor B8672-48 od firmy Transmotec [18]. Jedná se o třífázový BLDC motor zapojený do hvězdy s integrovanými hallovými senzory.

Tabulka 4.1: Významné parametry použitého motoru B8672-48[18]

Jmenovité napětí	48 V
Jmenovitý výkon	117 W
Otáčky naprázdno	3700 rpm
Jmenovité otáčky	3102 rpm
Jmenovitý moment	0,359 Nm
Jmenovitý proud	19,4 A
Počet pólových dvojic	4

4.2 Třífázový měnič

Byl použit výkonový třífázový měnič MC1L od firmy Michrochip [19]. Jedná se o univerzální modul, který slouží pro vývoj řídicích algoritmů pro třífázové motory. Obsahuje opticky oddělené spínání jednotlivých tranzistorů a měření proudu a napětí v jednotlivých větvích a stejnosměrném meziobvodu. Snímání proudu je provedeno jak pomocí senzorů založených na hallovém principu, tak pomocí měření úbytku napětí na bočnících.

Tabulka 4.2: Významné parametry použitého měniče MC1L[19]

Maximální napětí	48 V
Maximální efektivní proud	15 A
Maximální doporučená spínací frekvence tranzistorů	16 kHz
Minimální ochranná doba tranzistorů	2 μS

4.3 dSPACE

Veškeré experimenty, návrhy řídicích algoritmů, měření dat jsme prováděli pomocí dSPACE zařízení založeného na modulární dSPACE platformě [20]. Jedná se o jednotlivé karty, výpočetní nebo I/O, které se zasouvají do slotů v boxu. Tento box obsahuje napájecí zdroje a jednotlivé karty propojuje pomocí vysokorychlostní PHS sběrnice.

Modulární dSPACE dostupný v mechatronické laboratoři [21] se skládá z následujících součástí:

- **DS1006 Processor Board** - Výpočetní procesorová karta
- **DS5203 FPGA Board** - FPGA karta
- **DS5203M1 Multi-I/O Module** - Rozšiřující I/O pro FPGA kartu
- **DS2202 HIL I/O Board** - Vstupní a výtupní signály
- **DS814 Link Board** - Slouží k propojení s PC
- **PX20 Expansion Box** - Box

Podrobněji představíme nejduležitější součásti.

DS1006 Processor Board je výpočetní procesorová karta. Je založena na čtyřjádrovém 64-bitovém procesoru AMD Opteron^{TM} taktovaném na $2,8\ kHz$. Kromě procesoru obsahuje karta 1 GB DDR2 RAM paměti pro model a pro každé jádro je ještě k dispozici 128 MB RAM pro ukládání dat a komunikaci. Je možné vytvářet multiprocesorové systémy použitím více těchto karet v jednom boxu.

DS5203 FPGA board obsahuje FPGA Virtex-5 SX95T s hodinovým signálem 100 MHz, jehož parametry jsou uvedeny v tabulce 2.1. Dále obsahuje karta tyto vstupy a výstupy:

- 16x Digitalní vstup/výstup. Konfigurovatelná výstupní logika (3,3 nebo 5 V). Pro vstup nastavitelná hranice, napětí mezi Log 1 a Log 0.
- 6x ADC s rozlišením 14 bitů, vzorkování 10 MSPS a volitelné vstupní napětí $\pm 5\ V$ nebo $\pm 30\ V$.
- 6x DAC s rozlišením 14 bitů a výstupním napětím $\pm 10\ V$

5 Senzorové řízení BLDC motoru

Tato kapitola se zabývá nejjednodušším možným způsobem řízením BLDC motoru. Řízení využívá přímo informaci ze senzoru natočení, proto se tato metoda řízení nazývá senzorová. V této kapitole budou představeny také základní stavební bloku designu, které se objevují i ve zbytku práce (PWM modulace, dead-time generátor, určování rychlosti).

5.1 Úvod do metody senzorového řízení

Senzorové řízení využívající tří hallových senzorů je nejběžnější a nejjednodušší způsob jak řídit BLDC motor. To je důvod, proč jsme si toto vybrali jako první úkol. Senzorové řízení se zdá velmi jednoduché. Podle stavu tří hallových senzorů se spínají jednotlivé tranzistory třífázového měniče. Tuto posloupnost spínání jednotlivých tranzistorů v závislosti na stavu hallových senzorů nazýváme komutační tabulka 5.1.

Tabulka 5.1: Komutační tabulka senzorového řízení

Natočení [°el.]	Hall senzor			Stavy tranzistorů					
	A	B	C	Top A	Bot A	Top B	Bot B	Top C	Bot C
0-60	1	1	0	Off	On	Off	Off	On	Off
60-120	1	0	0	Off	On	On	Off	Off	Off
120-180	1	0	1	Off	Off	On	Off	Off	On
180-240	0	0	1	On	Off	Off	Off	Off	On
240-300	0	1	1	On	Off	Off	On	Off	Off
300-360	0	1	0	Off	Off	Off	On	On	Off

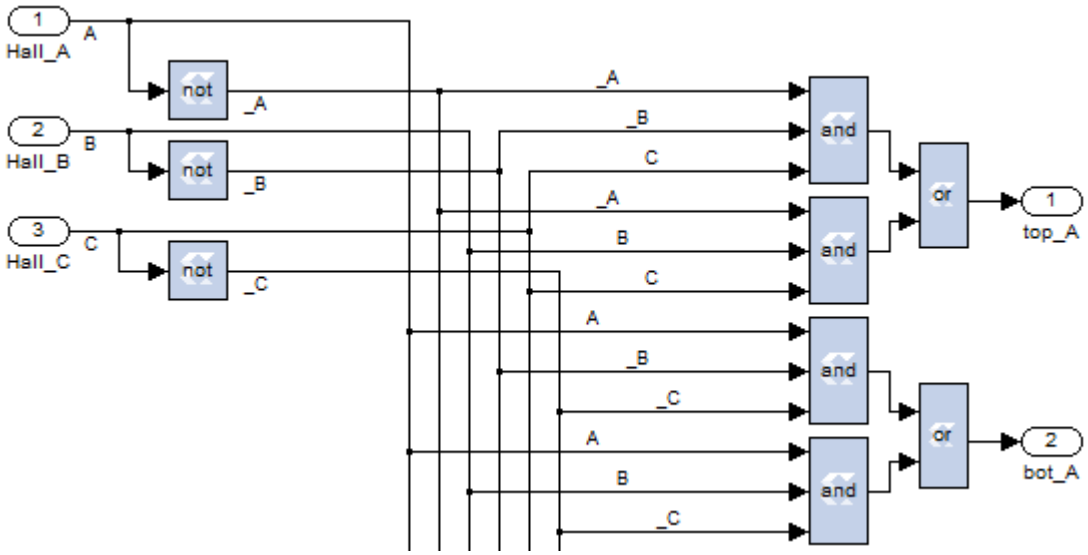
V levných aplikacích je toto vyřešeno jedním specializovaným integrovaným obvodem nebo se posloupnost spínání vytvoří ze základních logických obvodů. Implementace této komutační tabulky v FPGA je také velmi jednoduchá. Kontrolér BLDC motoru by měl kromě komutace zvládat také regulaci rychlosti pomocí PWM, s čímž souvisí vkládání ochranné doby (dead-time) pro tranzistory. Také je žádoucí určovat rychlost otáčení motoru.

V následujících podkapitolách budou podrobněji rozebrány jednotlivé bloky senzorového řízení. Tok dat v FPGA je následující: Získání informací z hallových senzorů, z informací z hallových senzorů na základě komutační tabulky vytvořit spínací

signály pro jednotlivé tranzistory, na signály pro horní tranzistory aplikovat PWM modulaci a dále na všechny signály pro tranzistory aplikovat ochrannou dobu.

5.2 Realizace komutační tabulky

Pro spínání každého tranzistoru existuje jednoduchá logická funkce, která se dá realizovat pomocí několika základních logických hradel. Situace je znázorněna na obrázku 5.1 pro horní a spodní tranzistor jedné větvě měniče. Pro zbylé tranzistory existuje obdobná kombinace.



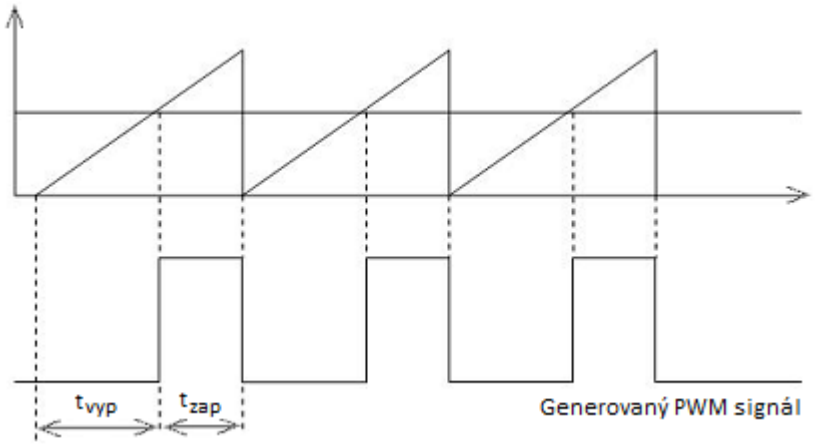
Obrázek 5.1: Realizace komutační tabulky pro jednu větev měniče

5.3 PWM modulace

Pro řízení rychlosti motoru se využívá pulsně šířkové modulace. Většina běžných mikrokontrolérů, měřící a řídicí HW má specializované periferie pro PWM. Při práci s FPGA si ale tuto periferii musíme vytvořit sami. Je to poměrně jednoduché a také výhodné, protože máme plně pod kontrolou rozlišení PWM signálu, frekvenci, apod. Například ani nejsme limitování při volbě rozlišení PWM signálu na mocniny dvou jako všechny běžné periferie [1].

Princip generování PWM signálu spočívá v komparaci pilovitého nebo trojúhelníkového signálu s referenční hodnotou, která představuje střídu (obr.: 5.2).

[1] Například 10, 12 bitová PWM znamená rozlišení 1024, 4096 hodnot



Obrázek 5.2: Princip generování PWM signálu

Perioda pilovitého signálu odpovídá periodě PWM signálu a výška, rozlišení pilovitého signálu odpovídá rozlišení PWM signálu. Pro vytvoření pilovitého signálu využijeme s výhodou n-bitový čítač, který po dosažení maximální hodnoty přeteče zpět do nuly a vytváří tak pilovitý signál. Protože čítač inkrementuje při každém hodinovém cyklu FPGA, musí být jeho velikost zvolena vhodně tak, aby se výsledná frekvence se blížila co nejvíce naší žádané.

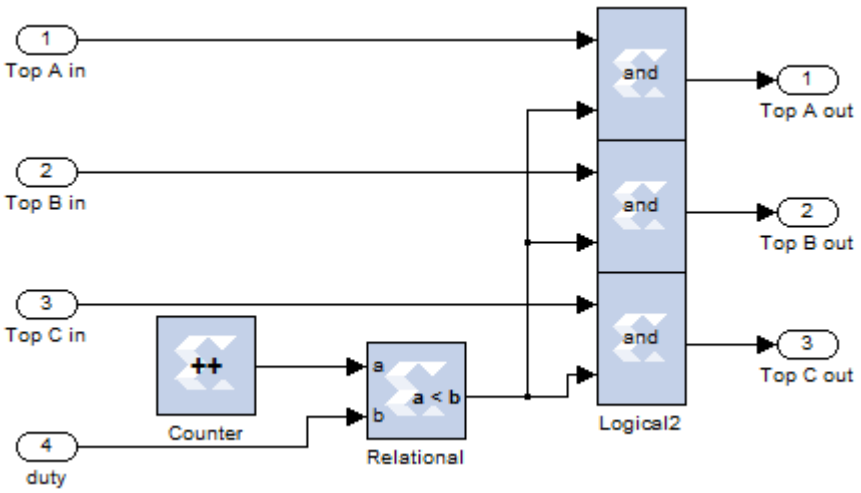
Maximální doporučená frekvence PWM námi použité výkonové elektroniky je 20 kHz [19] z důvodu omezení spínacích ztrát. Pro minimalizaci zvlnění proudu se snažíme použít spíše vyšší frekvenci PWM. Jako vhodné bylo zvoleno nepříliš standartní, ale na FPGA možné 13-ti bitové rozlišení PWM, tedy 8192 hodnot. Frekvence PWM signálu je dána vzorcem:

$$f_{pwm} = \frac{f_{FPGA}}{2^n} \tag{5.1}$$

Pro taktovací frekvenci FPGA $f_{FPGA} = 100Mhz$ a rozlišení $n = 13$ vychází frekvence PWM signálu $f_{PWM} \approx 12,2kHz$. Při realizaci na FPGA tedy použijeme 13-ti bitový čítač a jeho hodnotu budeme porovnávat s 13-ti bitovou hodnotou reprezentující střídu (obr.: 5.3). Při řízení rychlosti BLDC motoru stačí PWM signály řídit pouze horní nebo spodní tranzistory v měniči (omezení spínacích ztrát).

5.4 Realizace ochranné doby tranzistorů

Z důvodu přechodových jevů u tranzistorů, při jeho otevírání a zavíraní, které není okamžité, by mohlo dojít ke krátkodobému zkratu v jedné větvi měniče. Abychom tomotu jevu zabránili, zpožďuje se náběžná hrana spínacího signálu tranzistoru o

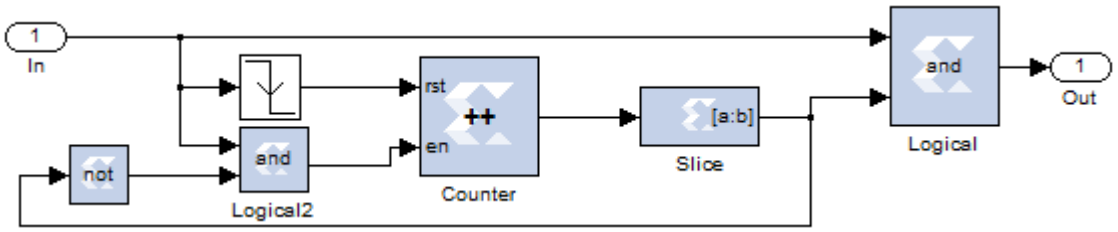


Obrázek 5.3: Realizace PWM modulace pro horní tranzistory měniče

určitý čas, který by měl být delší než délka přechodových jevů tranzistoru. Tato doba se nazývá ochranná (anglicky dead-time).

Při našem způsobu řízení BLDC motoru tato ochranná doba není potřebná. Vždy po vypnutí jednoho tranzistoru ve větvi jsou následně po dobu 60°el. oba tranzistory vyplé a zkrat nehrozí. Zkoušeli jsme ale i jiné způsoby PWM modulace, u kterých je ochranná doba nezbytná. Z datasheetu k použitému měniči [19] jsme vyčetli, že je nutné vkládat dead-time $2\mu S$.

Blok programu pro FPGA na obrázku 5.4 způsobí zpoždění náběžné hrany o $2,56\mu S$. Základem je čítač, který se vynuluje při sestupné hraně vstupního signálu a který čítá pokud je vstupní signál v log. 1. Blok *Slice* za čítačem slouží k vybrání určitého bitu nebo rozsahu bitů ze vstupního signálu. V tomto konkrétním případě vybíráme bit na osmé pozici (číslování jednotlivých bitů je od nejmémě významného bitu a od nuly). Až čítač napočítá do 256, osmý bit bude v log. 1 a podmiňuje aktivní stav výstupního signálu a také zabrání dalšímu čítání.



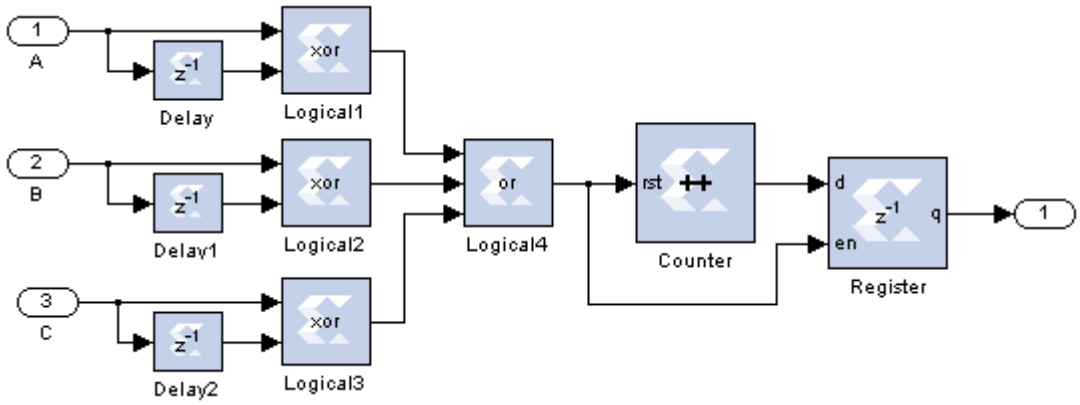
Obrázek 5.4: Realizace ochranné doby tranzistorů

5.5 Určování rychlosti otáčení motoru

Dalším krokem je zjištění rychlosti otáčení motoru. Na motoru není obvykle umístěn senzor otáček, ale rychlost otáčení je možné určit z informací z hallových senzorů. U bezsenzorového řízení potom z okamžiků průchodu indukovaného napětí nulou. Prvním testovaným způsobem bylo měření doby mezi změnami stavů na hallových senzorech.

Realizace tohoto způsobu na FPGA je na obrázku 5.5. Na každém vstupním signálu je vytvořena detekce změny pomocí negace a bitového operátoru XOR. Detekce změn je vedena na třívstupové hradlo OR a výstupní signál poté nuluje čítač. Při každé detekci změny se hodnota čítače uloží v paměti (registru). Máme tedy k dispozici informaci o době mezi přechody stavů, ze které je možno určit rychlost.

Ve skutečnosti ale tato informace o rychlosti velmi kolísala a prakticky byla nepoužitelná. Je to způsobeno nepřesnou montáží hallových senzorů a nepřesnou magnetizací magnetického disku. Při orientačním měření se doba jednotlivých přechodů lišila v řádu jednotek procent. Z výrobních důvodů je jasné, že informace z hallových senzorů nemůže být absolutně přesná. Situaci jsme vyřešili tím způsobem, že jsme sečetli několik posledních informací o době mezi přechody stavů, jistá obdoba plovoucího průměru. K úplné eliminaci chyb způsobených nepřesnou instalací senzorů by bylo potřeba pro osmipólový stroj sečíst posledních 24 hodnot změn stavů. Mohlo by ale vadit velké zpoždění signálu a tak bylo jako kompromis zvoleno sečtení pouze čtyř hodnot. Hodnota rychlosti již byla dostatečně stabilní. Sečtení pouze čtyř vzorků bylo zvoleno také proto, že doba trvání čtyř vzorků odpovídá 240°el. V následující kapitole o bezsenzorovém řízení je důležité znát informaci o trvání 30°el., kterou získáme dělením osmi z 240°el. Tím, že se ale jedná o dělení mocninou dvojky, můžeme obejít dělení nebo násobení převrácenou hodnotou a využijeme bitový posun, který je v FPGA téměř "zadarmo".



Obrázek 5.5: Realizace určování rychlosti ze stavů hallových senzorů

6 Bezsenzorové řízení BLDC motoru

Tato kapitola se zabývá tzv. bezsenzorovým řízením BLDC motoru, tj. řízením motoru bez použití senzorů polohy. V úvodu kapitoly je krátký teoretický úvod do metod bezsenzorového řízení a poté představíme implementaci jedné metody na FPGA.

6.1 Úvod do metod bezsenzorového řízení

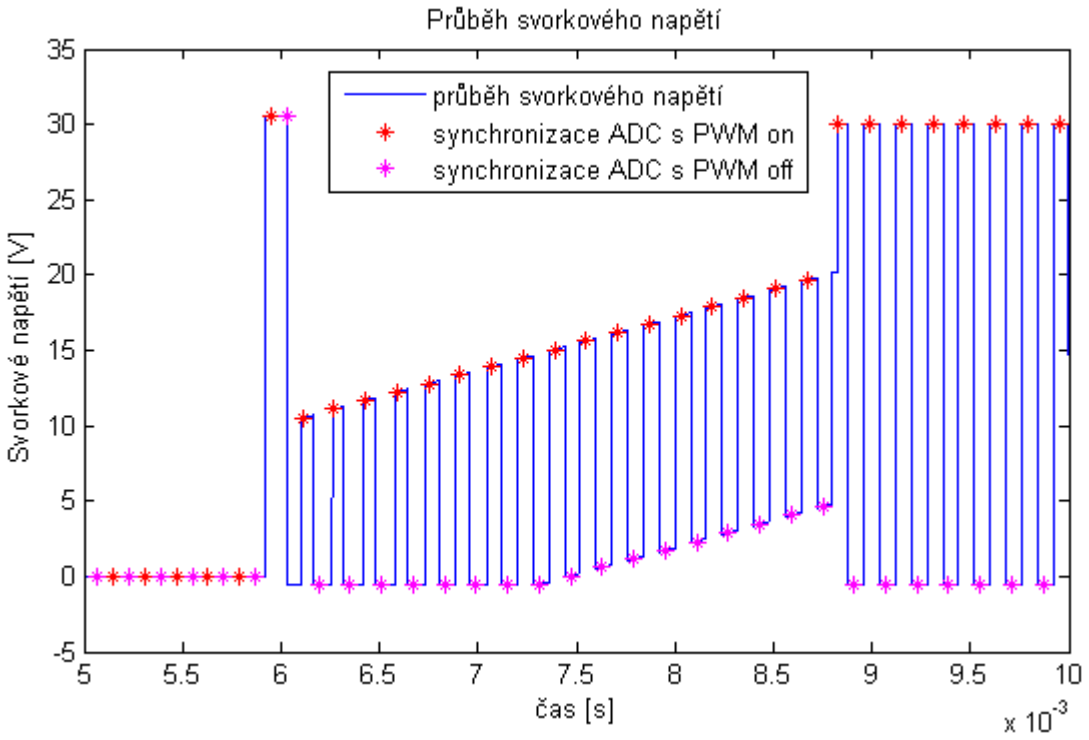
Bezsenzorové metody řízení BLDC motorů se stávají stále populárnějšími. Mezi jejich výhody patří zejména nízká cena pohonu (absence snímače polohy, jednodušší kabeláž) a teoreticky větší spolehlivost (opět absence senzorů). Někde ani není instalace senzorů polohy možná (motor je ponořen v kapalině u čerpadel). Bezsenzorové metody se z principu hodí zejména tam, kde nedochází k velkým změnám v rychlosti otáčení motoru. Podstatným nedostatkem velké většiny bezsenzorových metod je neschopnost řídit motor ve velmi malých a nulových otáčkách. Pro rozběh motoru musí být použita například některá metoda rozběhu v otevřené smyčce [22], [23]. V literatuře [13] je shrnutí většiny používaných metod bezsenzorového řízení. My se zaměříme pouze na modifikaci nejpožívanější metody, a tou je metoda detekce průchodu indukovaného napětí nulou.

6.2 Detekce průchodu indukovaného napětí nulou

Tato metoda je založena na měření indukovaného napětí na nenapájené fázi a porovnáváním s polovinou napájecího napětí. Jakmile překročí napětí na nenapájené fázi polovinu napájecího napětí, odměří se doba trvání 30°el. a provede se přepnutí fází. Situace je jasná z obrázku 2.1. Velikost indukovaného napětí je závislá na otáčkách. V nízkých otáčkách je indukované napětí malé, odstup napětí od šumu se snižuje, a proto tato metoda v nízkých otáčkách nefunguje spolehlivě. Při použití PWM modulace se situace ještě více komplikuje. Průběh napětí na svorce motoru při PWM modulaci je zobrazen na obrázku 6.1. Je tedy zřejmé, že signál musí být filtrován. Často se používají analogové filtry, které ale zavádějí do signálu zpoždění a je třeba s tím počítat a správně odměřit čas do přepnutí fází.

Existují také metody, které nepoužívají analogový filtr a zpracování signálu je provedeno v digitální podobě. Při digitalizaci signálu musí být snímání ADC

převodníku synchronizováné s nosnou frekvencí PWM modulátoru. Snímání může být synchronizováno s dobou, ve které je PWM signál ve stavu vypnuto nebo zapnuto. Pokud provedeme synchronizaci snímání ADC s PWM, vypadají průběhy signálu podobně jako na obrázku 6.1. K průchodu signálu nulou nebo polovinou napájecího napětí dochází na tomto obrázku přibližně v čase $7,5\ ms$. Signál, který je synchronizován s dobou PWM zapnuto a porovnává se s polovinou napájecího napětí má v okolí přechodu monotónní průběh , a proto je detekce průchodu poměrně snadná. Signál, který je synchronizován s dobou PWM vypnuto, je většinu času v nule a přechod by měl být detekován v okamžiku, jakmile signál začne stoupat od nuly. Ze pozornost ještě stojí komutační špička na začátku volnoběžně části, která vytváří falešný přechod. Tyto metody synchronizace ADC s PWM se v anglické literatuře označují jako *Direct Back EMF* [24], [25].



Obrázek 6.1: Průběh svorkového napětí při komutaci a jeho podoba při synchronizaci ADC s PWM

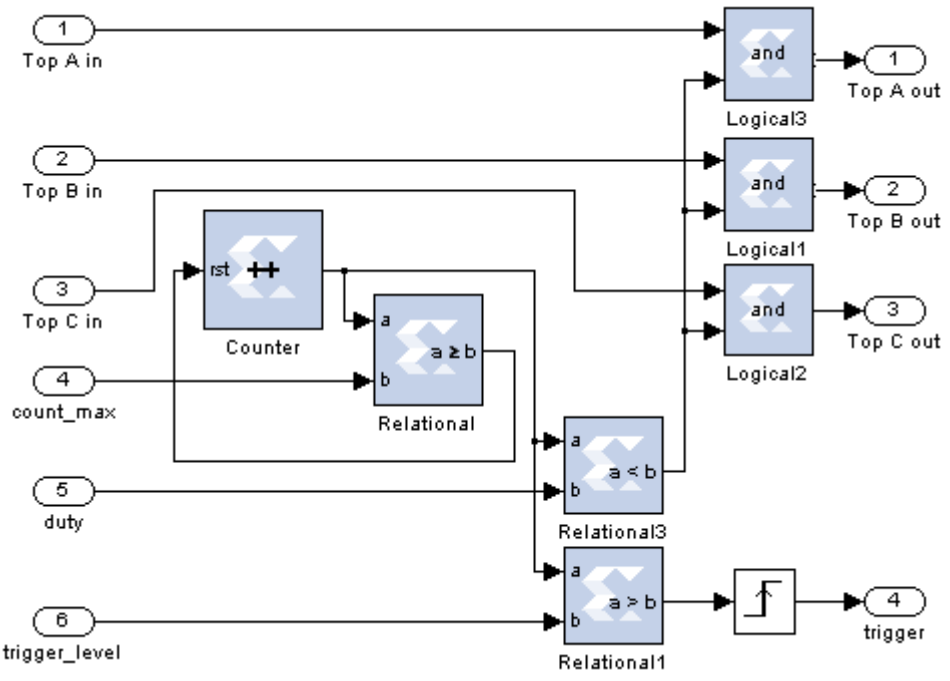
Pro aplikaci na FPGA jsme si vybrali metodou synchronizace ADC s PWM v době PWM zapnuto. Použití analogového filtru jsme zamítli především z toho důvodu, že tato metoda je poměrně známá, prozkoumaná a FPGA nám nabízí obrovském možnosti v digitálním zpracování signálu.

6.3 Implementace metody synchronizace ADC s PWM na FPGA

Implementace této metody probíhala ve dvou krocích. Nejprve jsme navrhli design sloužící k zobrazování naměřených vzorků dat při synchronizaci ADC s PWM. Design obsahoval PWM modulátor s plynule měnitelnou nosnou frekvencí a s možností přesně nastavovat, ve které části PWM cyklu se má načíst hodnota z ADC. Jakmile jsme zjistili, která frekvence PWM je nejoptimálnější a ve kterém čase načíst hodnotu z ADC, přistoupili jsme k digitální filtraci. Bylo třeba odstranit komutační špičky ze začátku volnoběžné části, které způsobovaly falešnou detekci.

6.3.1 Určení vhodné doby sychronizace a PWM frekvence

Pro toto měření bylo především nutné modifikovat PWM modulátor představený v kapitole 5.3 tak, aby umožňoval plynule nastavovat nosnou frekvenci. Dále jsme přidali druhý výstup z modulátoru s názávisle nastavitelnou střídou, který sloužil jako synchronizační signál pro ADC. Modifikaci, kterou jsme provedli, byla úprava přetékajícího čítače, který generoval pilovitý signál. Jakmile čítač dosáhl předem stanovené hodnoty, tak se vynuloval resetovacím vstupem (obr.: 6.2).



Obrázek 6.2: Generování PWM signálu s plynule měnitelnou nosnou frekvencí

Po úpravě modulátoru a přidání synchronizovaných vstupů z AD převodníků jsme přistoupili k vlastnímu měření. V ControlDesku jsme si vytvořili prostředí, přes které jsmě mohli nastavovat frekvenci a střídu PWM signálu a okamžik, ve kterém se má načíst hodnota z ADC. Součástí prostředí bylo vykreslování aktuálního a synchronizovaného signálu z ADC.

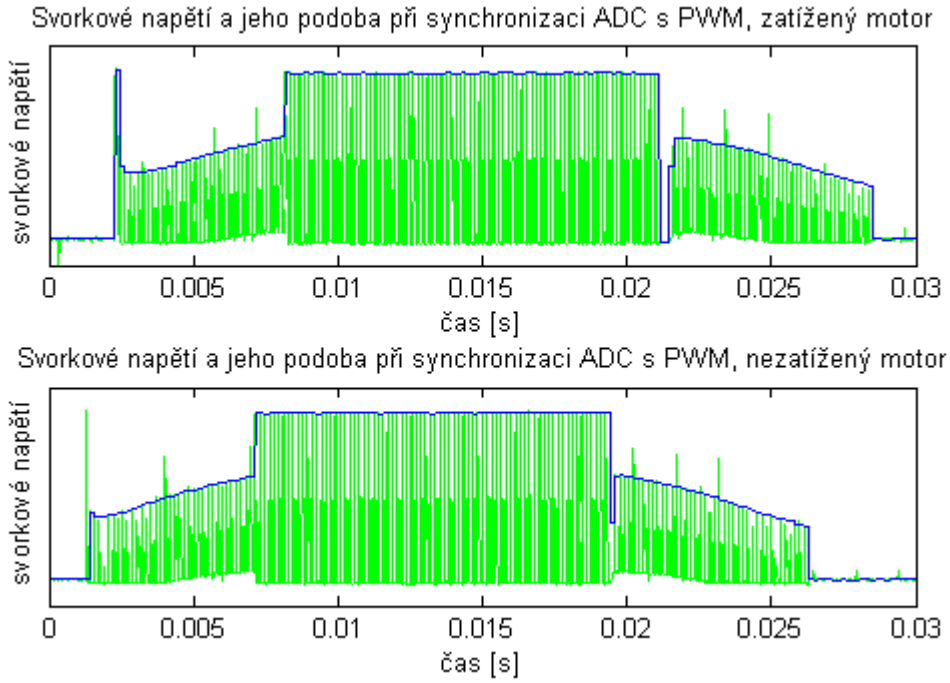
Změnami okamžiku snínání jsme si mohli vyzkoušet obě metody snímání, ve vypnuté i zapnuté části PWM cyklu. Zaměřili jsme se na řízení při zapnuté části cyklu. Podle očekávání jsme nejčistšího signálu dosáhli, když jsme hodnotu z ADC načetli na konci aktivní části PWM, těsně před sestupnou hranou. Další očekávání se potvrdilo, když signál byl nejčistší při nížších frekvencích PWM. Tímto požadavkem se ale nemůžene úplně řídit, protože při nižší střídě dochází k většímu zvlnění proudu. Požadavek nízké střídy také znemožňuje řídit motor touto metodou při vysokých otáčkách. Při vysokých otáčkách je během volnoběžné části PWM cyklu méně vzorků z ADC, průběh je výrazně schodovitý a není možné přesně určit okamžik průchodu. Situaci přiblížíme na modelovém příkladu. Mějme PWM signál o rozlišení 14-bitů, tedy frekvenci $6,1kHz$. Při 3000 $ot \cdot min^{-1}$ mechanických jsou elektrické 12000 $ot \cdot min^{-1}$ a na jednu otáčku tedy vychází asi 30 hodnot z ADC. Během volnoběžné části je to asi pouze 5 vzorků a nemůžeme tedy přesně určit okamžik průchodu. Volba frekvence nám omezuje maximální dosažitelné otáčky motoru.

Na grafech v obrázku 6.3 jsou zobrazeny průběhy napětí při synchronizaci ADC s PWM při zatíženém a nezatíženém motoru. Za povšinutí stojí, že u nezatíženého motoru nejsou patrny komutační špičky.

6.3.2 Digitální filtrace

Průběhy napětí v předchozí sekci vypadají poměrně čistě. Při detekci průchodu polovinou napájecího napětí ale situaci komplikuje komutační špička, která vytváří falešný průchod a musíme ji ignorovat. Úkol se ukázal jako poměrně komplikovaný, protože komutační špička může být různě široká a výška ani nemusí dosahovat plného napájecího napětí. Tato špička by se dala poměrně jednoduše ignorovat zavedením časového ignoračního pásma, kdybychom znali rychlost motoru. Ale informace o rychlosti není spolehlivá při náhlých změnách rychlosti otáčení a po rozběhu motoru metodou v otevřené smyčce. Proto jsme se snažili špičku ignorovat pouze zpracováním toho konkrétního signálu.

První kritérium pro rozpoznání skutečného přechodu bylo vytvoření napěťového tolerančního pásma v okolí hodnoty přechodu. Toto kritérium správně ignorovalo většinu chybných přechodů od komutačních špiček. Občas se, hlavně při nezatíženém motoru objevila komutační špička, jejíž velikost zasahovala právě do tohoto tolerančního pásma, což způsobilo její chybnou detekci. Jako další kontrolní kritérium se určoval rozdíl tří posledních vzorků napětí. Pokud byl rozdíl malý, malé změny signálu, jednalo se o skutečný přechod. Konktrétní hodnoty velikosti tolerančního



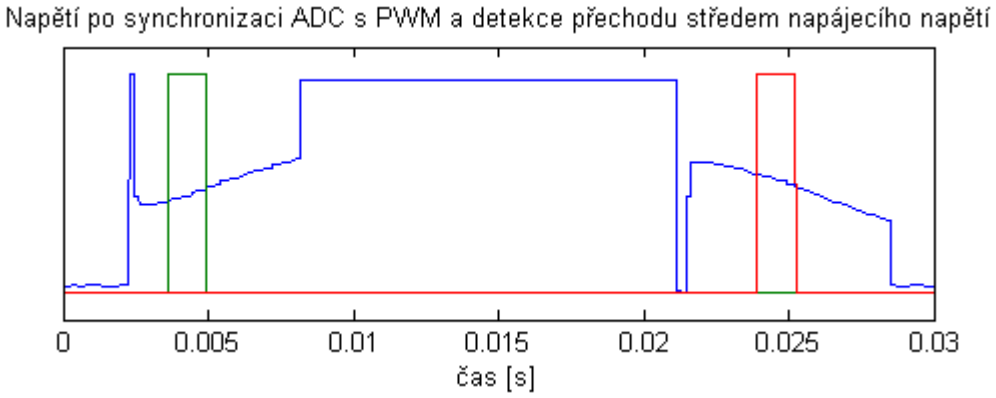
Obrázek 6.3: Průběh napětí při synchronizaci ADC s PWM

pásma a velikosti rozdílu vzorků jsme určovali metodou pokus, omyl tak, aby bezsenzorové řízení fungovalo v co největším rozsahu otáček. Protože se hodnoty ukázaly závislé na rychlosti otáčení motoru, tak ve finální verzi řízení jsou odvozeny od střídy a s rostoucí střídou se zvětšují.

Tato metoda řízení fungovalo od asi 300 $ot \cdot min^{-1}$. Při nižších otáčkách docházelo k chybným detekcím přechodu a motor se zastavil. Vytvořili jsme tedy ještě dva kontrolní mechanismy, které jsou aktivní pouze při nízkých otáčkách. Kontrolují se poslední tři vzorky logického výstupu předchozích kritérií. Výstup je považován za správný, pokud mají všechny tři vzorky log 1. Posledním kritériem je kontrola napětí na zbývajících fázích, které musí být v určitém rozsahu vyplývajícím z činnosti BLDC motoru (Obr.: 2.1).

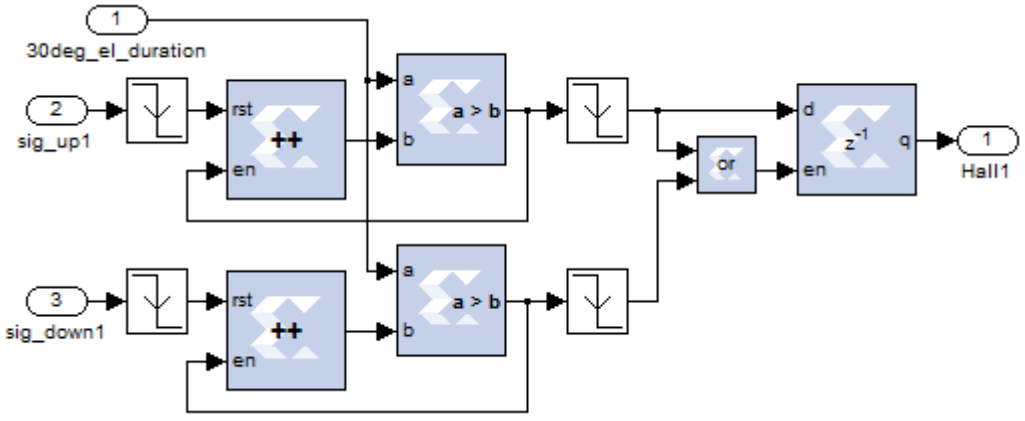
Reprezentace tohoto digitálního zpracování na FPGA je poměrně rozsáhlý a komplikovaný design, skládající se z mnoha jednoduchých operací. Jeho podrobnější vysvětlení by zabralo zbytečně mnoho stran práce. Výsledné průběhy detekcí přechodu jsou znázorněny na obrázku 6.4. Je zde zobrazen průběh napětí snímaného synchronizovaně s PWM a dva obdelníky, které ukazují přechod středem napájecího napětí. K přechodu dochází při sestupných hranách těchto obdelníků.

Přechody signálu středem napájecího napětí jsou tedy poměrně spolehlivě identifikovány a zbývá z nich vygenerovat signál odpovídající signálu z hallova senzoru.



Obrázek 6.4: Zvýraznění detekce přechodu středem napájecího napětí

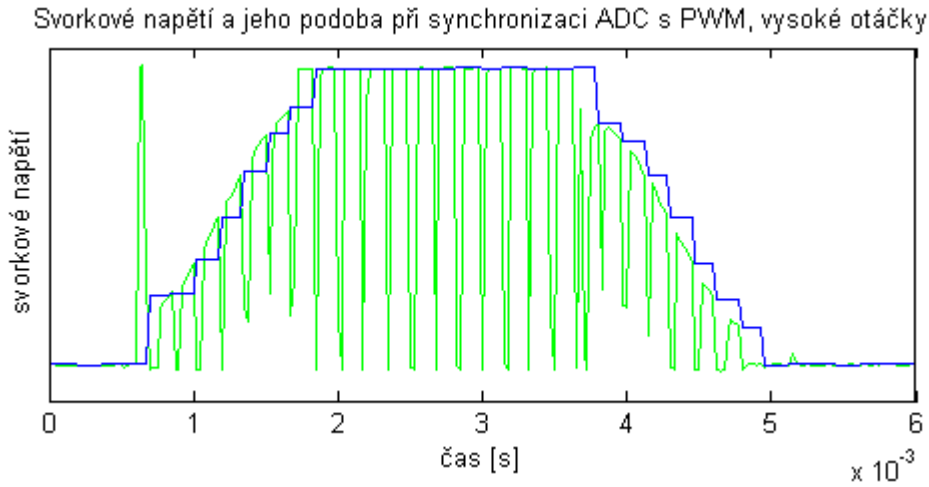
Okamžik přepnutí fází a změny stavu hallova senzoru je vzdálen 30 °*el*.. Tuto dobu získáme bitovým posunem (děleným osmi) z informace používanou pro určování rychlosti otáčení (Kapitola 5.5). Realizace rekonstrukce signálu hallova senzoru je na obrázku 6.5. Signál značící přechod při rostoucím napětí vstupuje do horní části schématu, která je zodpovědná za nastavení log 1 ve výstupním signálu. Sestupná hrana vstupních signálů nuluje čítač, který počítá až do hodnoty danou trvaním 30 °*el*. Jakmile čítač této hodnoty dosáhne, nastaví výstupnímu signálu odpovídající logickou úroveň.



Obrázek 6.5: Generování řídícího signálů ze signálů přechodů polovinou napájecího napětí

6.3.3 Vlastnosti použité metody

Při testování algoritmus spolehlivě fungoval od asi 150 $ot \cdot min^{-1}$ mechanických (600 $ot \cdot min^{-1}$ elektrických), což odpovídá amplitudě indukovaného napětí přibližně $1,8\ V$. Vzhledem k tomu, že signálové kabely byly s řídicím HW propojeny poměrně dlouhými, nestíněnými vodiči a signály, které nesly informaci o napětí nebyly příliš přizpůsobeny rozsahu ADC, můžeme říci, že metoda je poměrně úspěšná. Při rychlých změnách otáček metoda také fungovala spolehlivě. Pouze při vysokých otáčkách není řízení příliš vhodné. V závislosti na nosné frekvenci PWM signálu je při vyšších otáčkách k dispozici méně vzorků napětí během jedné otáčky a rozlišení na časové ose je malé. Na obrázku 6.6 při vyšších otáčkách je vidět, jak se již napětí ve volnoběžné části stává vyrazně schodovité a není možné přesně určit okamžik průchodu.



Obrázek 6.6: Průběh napětí při synchronizaci ADC s PWM, vysoké otáčky

Zpracovnání signálu, odstranění chybného přechodu způsobeného komutační špičkou je námi vytvořená metoda bez inspirace jiným řešením. Je zde poměrně mnoho rozhodovacích kritérií a metoda by měla měla být pořádně otestována. Určité zlepšení spolehlivosti by mohlo být dosaženo zavedením časového ignoračního pásma.

Implemetací této, ale obecně jakékoliv metody bezsenzorového řízení na FPGA získáme výhodu ve velkých možnostech zpracování signálu. Také se vyhneme problémům, které mohou mít aplikace na procesorech s přesným generováním zpoždění.

7 Řízení motoru v degradovaném módu

Tato kapitola se zabývá řízením motoru v tzv. degradovaném módu, tj. snaží se zachovat funkčnost pohonu i při nefunkčním signálu z jednoho hallova senzoru. Na záčátku kapitoly se seznámíme s metodou představenou v [13] a poté představíme vlastní metodu implementovanou na FPGA.

7.1 Požadavky na funkci algoritmu

Cílem je zachovat alespoň částečnou funkci pohonu při výpadku signálu z jednoho hallova senzoru při předpokladu, že není dostupné řízení motoru pomocí bezsenzorových metod. Algoritmus by měl vyhodnotit chyby, při kterých je na výstupu z hallova senzoru trvalá hodnota log 0 nebo log 1, případně nahodilé střídání těchto hodnot. Algoritmus by se měl také v případě opětovné správné funkce chybného senzoru návrátit do normálního režimu.

7.2 Metoda představená v [13]

Metoda je založena na předpokladu, že k okamžiku komutace, přechodu do dalšího stavu by mělo dojít v určitém časovém intervalu v závislosti na rychlosti otáčení motoru. Pokud se informace ze senzoru změní mimo tento interval, nebo pokud má konstantní hodnotu, provede se náhradní komutace na konci intervalu očekávající příchod signálu. V případě, že dojde k chybám několik cyklů po sobě, dojde k vyřazení senzoru a provádí se komutace na základě časové interpolace. Vyřazený senzor je ale dále monitorován a v případě opětovné správné funkce je použit k řízení. Celý algoritmus je vytvořen ve StateFlow a je poměrně komplikovaný.

Algoritmus má několik nedostatků. Díky tomu, že je implementován na procesoru, kde výpočetní smyčka běží na frekvenci 50 kHz jsou omezeny maximální otáčky na asi 10000 $ot \cdot min^{-1}$ elektrických (u použitého motoru 2500 $ot \cdot min^{-1}$ mechanických). Dalším závažným problémem jsou rychlé změny otáček, kdy dochází k chybnému vyhodnocení nefunkčnosti senzoru. Rychlost motoru určená z dob změn několika posledních stavů hallových senzorů se při rychlých změnách rychlosti stává neaktuální a komutace se očekává ve špatný okamžik. Posledním nedostatkem je, že algoritmus neumožňuje rozběh motoru s poruchou jednoho senzoru a algoritmus je

i po rozběhu nefunkční do té doby, dokud se nepodaří načíst informaci o rychlosti (potřeba funkce všech senzorů).

7.3 Vlastní vyvinutá metoda

Před vývojem nové metody a implementací na FPGA jsme si dali za cíl odstranit nedostatky metody představené v předchozí sekci. Vzhledem k tomu, že implementace se bude provádět na FPGA, tak první nedostatek, omezení maximální rychlosti otáčení motoru, se ihned sám vyřešil. Druhý nedostatek, chybné vyhodnocení senzoru při rychlých změnách otáček, bude vyžadovat úplně jiný přístup a změnu algoritmu. Pokusíme se vyřešit i třetí nedostatek, a to rozběh motoru s nefunkční hallovou sondou.

Náš algoritmus můžeme rozdělit do tří částí:

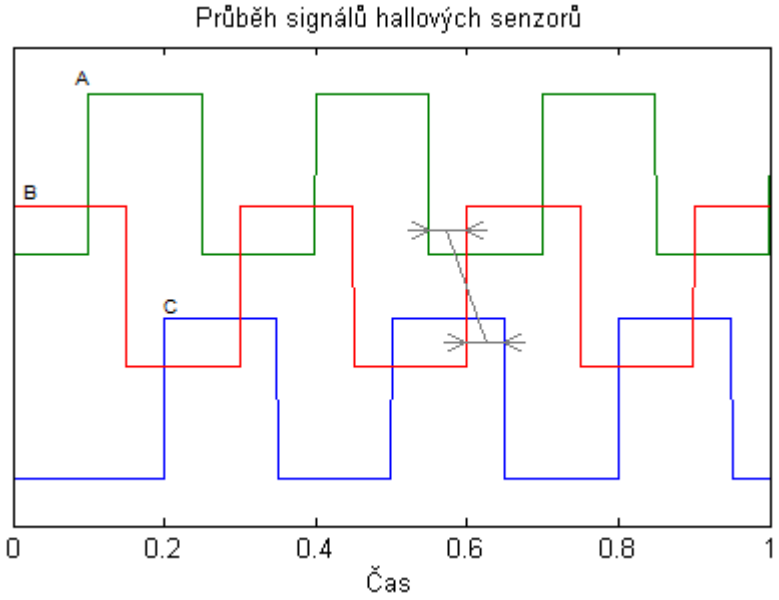
- Generování jednoho signálu hallova senzoru pomocí dvou zbývajících.
- Rozpoznávání chybného senzoru a přepnutí na generovaný signál.
- Ochranný algoritmus, který v případě výskytu chybné kombinace stavu senzorů vypne všechny tranzistory.

7.3.1 Generování signálu senzoru za pomocí dvou zbývajících

Průběhy signálů hallových senzorů jsou periodické, obdelníkové, vzájemně fázově posunuté o 120°. Teoreticky by tedy stačil jeden signál pro generování zbývajících dvou. Simulačně jsme si to vyzkoušeli. Na základě délky periody jednoho signálu jsme generovali zbývající dva signály. Při praktické realizaci se tato metoda ukázala jako nepoužitelná především z důvodu nepřesnosti instalace hallových senzorů. Také při rychlých změnách otáček byly generované signály značně nepřesné, protože délka periody se měřila průměrováním několika posledních period.

Nakonec bylo rozhodnuto využívat dvou funkčních signálů pro generování jednoho chybějícího a využívat pokud možno co nejaktuálnějších informací bez průměrování několika posledních vzorků. Měří se doba mezi sestupnou hranou jednoho signálu a vzestupnou hranou druhého signálu. Za tuto dobu se potom provede změna na generovaném signálu. Znázorněno je to na obrázku 7.1. Analogicky se provede změna pro druhou změnu generovaného signálu.

Praktická realizace pro implementaci na FPGA je na obrázku 7.2. Na základě funkčních signálů A a B se generuje signál C. Schéma můžeme rozdělit na horní a spodní větev. Horní větev nastavuje generovaný signál do log 0 a spodní větev nastavuje generovaný signál do log 1. Princip vysvětlíme na horní větvi, situace pro spodní větev je analogická.



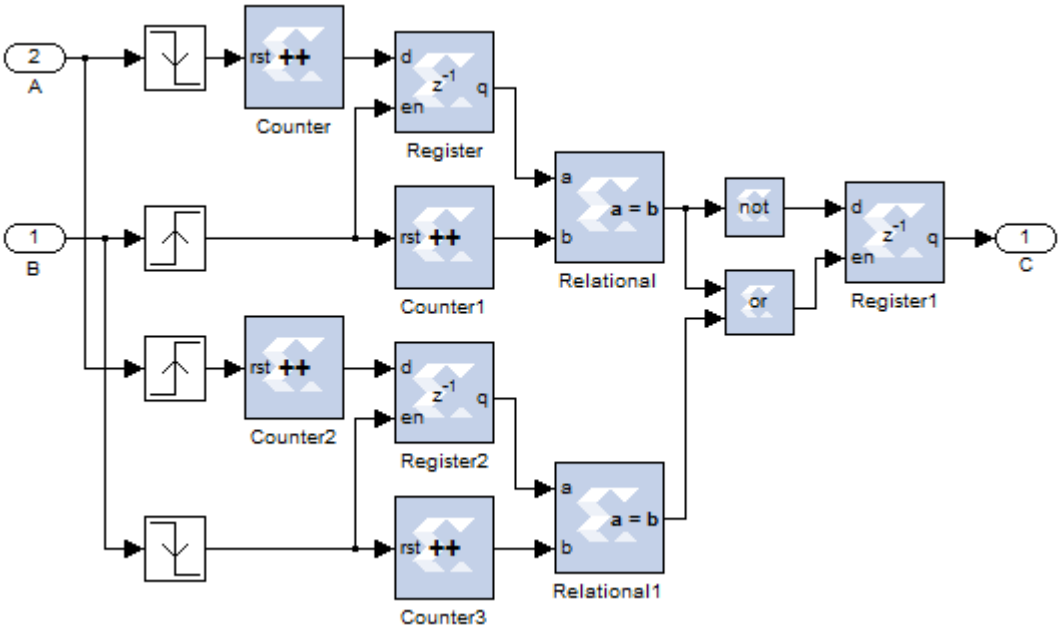
Obrázek 7.1: Průběh signálů hallových senzorů a generování náhradního signálu

Sestupná hrana signálu A nuluje první čítač, který inkrementuje při každém taktu FPGA. Při příchodu vzestupné hrany signálu B se uloží hodnota prvního čítače do registru a nuluje se druhý čítač. Jakmile dosáhne hodnota druhého čítače hodnoty, kterou měl první čítač, provede se změna na generovaném signálu. Tímto je uspokojivě vyřešeno generování chybného signálu.

7.3.2 Rozpoznání chybného signálu

Rozpoznání chybného senzoru se ukázalo jako velmi komplikovaný úkol, nakterém závisí uspěch algoritmu pro řízení motoru v degradovaném módu. V [13] je rozpoznání chybného stavu provedeno na základě toho, zda změna signálu přijde v očekávaném časovém intervalu. Problém je ale s určením tohoto časového intervalu, protože díky nepřesnostem při montáži hallových sond a při rychlých změnách rychlosti není možné spolehlivě určit tento interval. Hlavně náhlé změny rychlosti představují problém. Určitým zlepšením by mohlo být korigování očekáváného intervalu podle aktuální změny proudu (změna zatížení, otáček).

My jsme vyvinuli jinou metodu, založenou na kontrole přechodu stavů hallových senzorů. V tabulce 7.1 jsou vypsány stavy, tak jak jdou po sobě při otáčení motoru na jednu stranu. Podíváme-li se podrobněji na dvojice **BA**, **CB** a **AC**, vidíme, že se stále opakuje sekvence stavů **00, 01, 11, 10**. Naše metoda je založena na kontroluje, zda dochází k přechodu stavů v tomto pořadí. Dojde-li k přechodu do



Obrázek 7.2: Realizace generování jednoho signálu na základě dvou funkčních

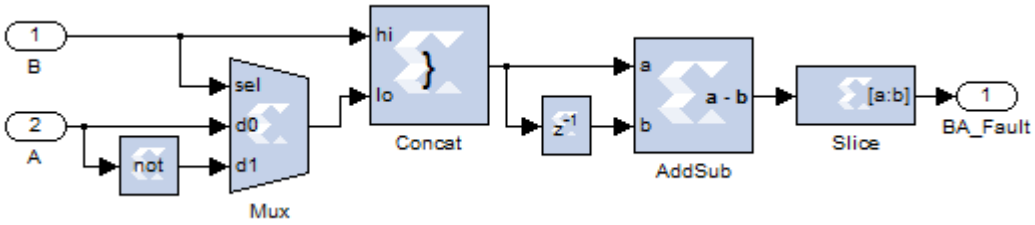
stavu, který není navazující, algoritmus nahlásí chybu a je jasné, že jeden ze signálů je chybný. Kontrola probíhá pro tři dvojice signálů, abychom mohli s jistotou říci, který konkrétní signál je chybný. V případě, že je chybný například signál **A**, tak kontrola dvojic **BA** a **AC** hlásí chybu a my víme, že chybný musí být signál **A**.

Tabulka 7.1: Stavy hallových senzorů

Pořadové číslo	$\varphi[^\circ\ el.]$	A	B	C
1.	0-60	0	1	0
2.	60-120	0	1	1
3.	120-180	0	0	1
4.	180-240	1	0	1
5.	240-300	1	0	0
6.	300-360	1	1	0

Realizaci kontroly stavů na FPGA jsme provedli následovně (obr.: 7.3). Sekvenci **00, 01, 11, 10** jsme převedli na sekvenci **00, 01, 10, 11**, která představuje binární počítání. Převedení jsme provedli multiplexorem, který v případě, že signál **B** je v log 1 negoval hodnotu signálu **A**. Výslednou dvoubitovou hodnotu jsme vedli na sčítačku, která odečítala zpožděnou hodnotu tohoto signálu s aktuální hodnotou.

Pokud je rozdíl těchto hodnot 1 nebo 0, je vše v pořádku. V opačném případě došlo k neočekávanému přechodu a jeden signál musí být chybný.



Obrázek 7.3: Realizace kontroly přechodu stavů dvojice signálů BA

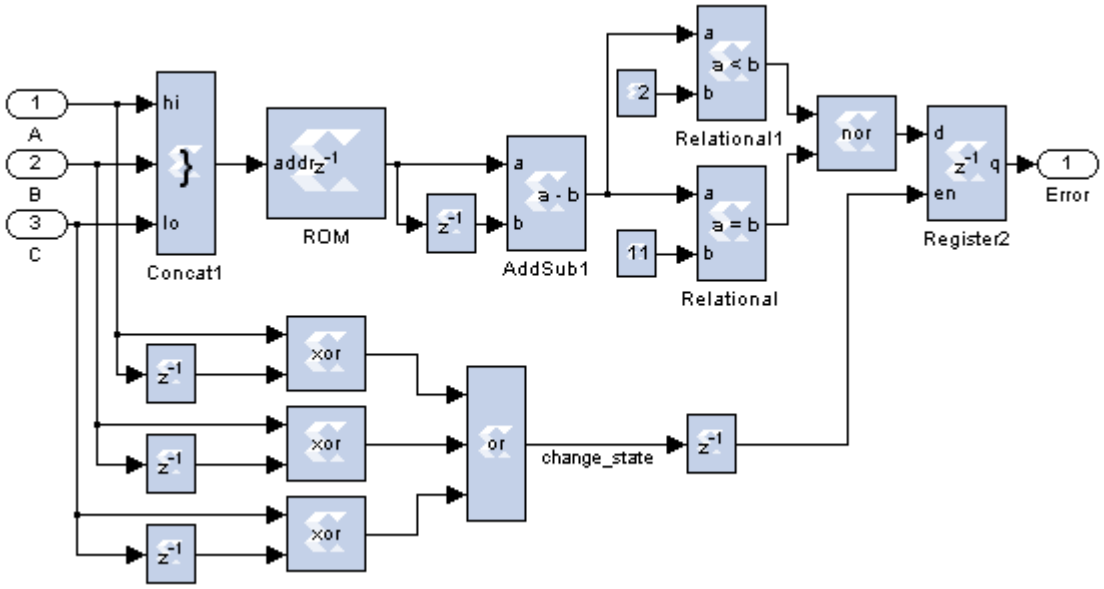
7.3.3 Ochranný algoritmus

Protože předchozí algoritmus má určité zpoždění, než dojde k přepnutí na náhradní signál hallova senzoru, vytvořili jsme ještě ochranný algoritmus, který kontroluje přechody stavů a případně hlasí chybu. Toto chybové hlášení použijeme k tomu, abychom dočasně vypnuli tranzistory. Děláme to z toho důvodu, protože chyba hallova senzoru by mohla způsobit sepnutí špatné fáze motoru a mohlo by dojít k nadproudu.

Algoritmus funguje na podobném principu jako předchozí algoritmus rozpoznávání chybového signálu, ale pracuje se všemi třemi signály. Realizace na FPGA je na obrázku 7.4. Tři signály sloučíme do jednoho tříbitového a podle stavu tohoto signálu se v ROM paměti vybere pořadové číslo stavu podle tabulky 7.1 (Paměť ROM se chová jako look-up tabulka). Následuje odečítání předchozího stavu od aktuálního a podle výsledku se určí, zda došlo ke korektnímu přechodu. Spodní větev s hradly XOR a OR provadí detekci změny stavů na vstupních signálech. Pokud dojde ke změně na vstupních signálech, uloží se aktuální chybová hodnota v registru.

7.3.4 Testování algoritmu

Testování probíhalo z velké části jako offline simulace během vývoje algoritmu. Jedna ze simulací, kdy na horním grafu je průběh tří signálů, z nichž na jednom je simulována chyba, je na obrázku 7.5. Na spodním grafu je potom průběh tří signálů po průchodu alogoritmem pro řízení v degradovaném módu. Je vidět, že chybný signál je z větší části opraven. Algoritmus ovlivnil i průběh prvního signálu, kde jsou vidět dvě špičky. Způsobil to ochranný algoritmus, který do doby, než dojde ke správnému generování chybějícího signálu nastavil všechny signály do log 0 a tím



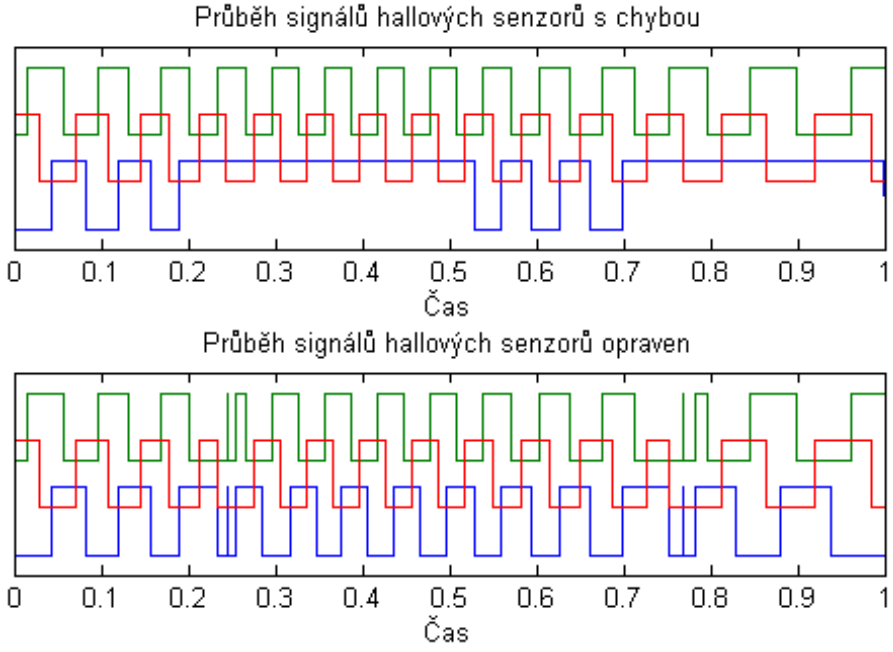
Obrázek 7.4: Ochranný mechanismus

vypnul všechny tranzistory. Předpokládá se, že rotor se díky setrvačnosti dotočí do další pozice, kde začne algoritmus správně generovat chybějící signál. V simulaci bylo testováno mnoho typů chyb, kdy se chybový signál nahodile měnil a vždy se po krátké době správně začal generovat náhradní signál.

Pro testování na reálném motoru jsme si vytvořili kontrolní prostředí v ControlDesku, kde jsme mohli simulovat chyby na signálech. Při řízení motoru s jedním chybovým signálem nebyla patrna žádná změna proti chování se správnými signály. Algoritmus fungoval korektně při velmi malých i maximálních otáčkách motoru a při velkých změnách rychlosti.

Dosáhli jsme tedy výrazného zlepšení proti metodě představené v [13]. Nejsme omezeni maximálními otáčkami, komutace přichází vždy ve správný okamžik a algoritmus funguje korektně i při velkých změnách rychlosti. Nepodařil se ale vyřešit rozběh motoru s chybným hallovým senzorem, v okolí nulových otáček algoritmus nefunguje. Situace je pravděpodobně tímto způsobem neřešitelná a rozběh motoru by musel být zajištěn nějakou rozběhovou metodou pracující v otevřené smyčce.

Vzhledem k tomu, že se nám v kapitole 8 podařilo implementovat RT model BLDC motoru, dala by se uvažovat o využití tohoto modelu pro řízení motoru v degradovaném módu. Něco na způsob obdoby stavového pozorovatele.



Obrázek 7.5: Simulace chybného signálu a jeho opravená podoba

8 HIL simulace BLDC motoru

V této kapitole si projdeme od sestavení matematického modelu BLDC motoru přes jeho simulační model v prostředí Matlab-Simulink, až po jeho implementaci jako model pro HIL simulaci běžící v reálném čase na modulárním HW dSPACE.

8.1 Matematický model BLDC motoru

Matematické modely jsou důležité pro praxi, ať už pro návrh regulačních algoritmů, simulování celého systéue během návrhu, atd. Jednotlivé modely jsou více či méně přesné. Volíme vždy model takový, který nám svou přesností dostačuje, aby nebyl zbytečně složitý, výpočetně náročný.

Při matematickém popisu jsme vycházeli z [13], [4].Pro náš model BLDC motoru jsme si zavedli následující zjednodušující předpoklady:

- Stroj je symetrický, trojfázový, zapojený do hvězdy. Odpory a indukčnosti jednotlivých fází jsou stejné.
- Indukčnosti jednotlivých fází a jejich vzájemné indukčnosti jsou konstantní, nezávislé na natočení stroje.
- Magnetický obvod je nenasycený, zanedbáváme rozptylové indukčnosti, ztráty v železe a vířivé proudy.
- Průběhy indukovaného napětí jsou ideální trapézoidy.
- Neuvažujeme oteplení motoru a teplotní vlivy.
- Uvažujeme pouze viskózní tření.

8.1.1 Elektrická část stroje

Rovnice popisující elektrickou část třífázového BLDC motoru jsou:

$$u_{as} = R_a i_a + L_a \frac{di_a}{dt} + M_{ab} \frac{di_b}{dt} + M_{ac} \frac{di_c}{dt} + e_a \tag{8.1}$$

$$u_{bs} = R_b i_b + L_b \frac{di_b}{dt} + M_{bc} \frac{di_c}{dt} + M_{ab} \frac{di_a}{dt} + e_b \tag{8.2}$$

$$u_{cs} = R_c i_c + L_c \frac{di_c}{dt} + M_{ac} \frac{di_a}{dt} + M_{bc} \frac{di_b}{dt} + e_c \tag{8.3}$$

a rovnice plynoucí z prvního Kirchhoffova zákona:

$$i_a + i_b + i_c = 0 \tag{8.4}$$

, kde:

u_{as}, u_{bs}, u_{cs}	jsou jednotlivá fázová napětí vztažená ke středu
R_a, R_b, R_c	jsou odpory jednotlivých vinutí
L_a, L_b, L_c	jsou indukčnosti jednotlivých vinutí
M_{ab}, M_{ac}, M_{bc}	jsou vzájemné indukčnosti
e_a, e_b, e_c	jsou indukovaná napětí
i_a, i_b, i_c	jsou fázové proudy

Při uvažování symetrického motoru, po algebraických úpravách a zavedení substituce:

$$L_s = L - M \tag{8.5}$$

dostáváme rovnice v jednodušší podobě:

$$u_{as} = R_a i_a + L_s \frac{di_a}{dt} + e_a \tag{8.6}$$

$$u_{bs} = R_b i_b + L_s \frac{di_b}{dt} + e_b \tag{8.7}$$

$$u_{cs} = R_c i_c + L_s \frac{di_c}{dt} + e_c \tag{8.8}$$

Pro úplnost předchozích rovnic je ještě potřeba vyjádřit indukované napětí. Velikost indukovaného napětí je obecně funkcí úhlové rychlosti a polohy rotoru:

$$e_x(\omega_m, \varphi_{el}) = \lambda f_x(\varphi_{el})\omega_m \tag{8.9}$$

, kde:

e_x	indukované napětí fáze
λ	napěťová konstanta (závislá na konstrukci stroje)
ω_m	mechanická úhlová rychlost stroje
$f_x(\varphi_{el})$	funkce závislosti indukovaného napětí na elektrickém natočení stroje, periodická lichoběžníková funkce s amplitudou 1, tvarově podobná funkci indukovaného napětí na obrázku 2.1

8.1.2 Mechanická část stroje

Momentová rovnice stroje je dána vztahem:

$$M_i = J\frac{d\omega_m}{dt} + B\omega_m + M_z \tag{8.10}$$

, kde:

M_i je vnitřní moment stroje
J je moment setrvačnosti rotoru
B je koeficient viskózního tření
M_z je zátěžný moment

Vnitřní moment stroje je potom dán rovnicí:

$$M_i = \frac{i_a e_a + i_b e_b + i_c e_c}{\omega_m} \tag{8.11}$$

Pro úplnost chybí ještě vztah mezi elektrickým natočením a mechanickou úhlovou rychlostí:

$$\omega_m = \frac{1}{p}\frac{d\varphi_{el}}{dt} \tag{8.12}$$

, kde:

p je počet pólových dvojic

8.2 Simulační model BLDC motoru

Simulace modelu je provedena v prostředí Matlab-Simulink. Vycházíme z modelu demonstrovaném v [13], kde je ale využito stavového popisu. Pro naši budoucí potřebu, běh v RT na FPGA využijeme bežného popisu pomocí rovnic. Dále si model upravíme a rozdělíme na funkční celky, protože ty části, které mají rychlou dynamiku (elektrická část a měnič) umístíme na FPGA a funkční celky s pomalejší dynamikou (mechanická část) necháme bežet na RT procesoru.

Ještě si ale vhodně upravíme elektrickou část modelu. V ní jsou vyjádřena fázová napětí vůči středovému bodu. BLDC motor je ale většinou zapojen do hvězdy a středový bod není dostupný, proto provedeme přepočet na svorkové napětí stroje:

$$u_a - u_s = R_a i_a + L_s \frac{di_a}{dt} + e_a \tag{8.13}$$

$$u_b - u_s = R_b i_b + L_s \frac{di_b}{dt} + e_b \tag{8.14}$$

$$u_c - u_s = R_c i_c + L_s \frac{di_c}{dt} + e_c \tag{8.15}$$

, kde:

u_a, u_b, u_c jsou svorková napětí stroje
u_s je napětí středového bodu

Sečtením rovnic 8.13 - 8.14, užitím Kirchhoffova zákona a následnou úpravou dostáváme napětí středového bodu jako:

$$u_s = \frac{u_a + u_b + u_c - e_a - e_b - e_c}{3} \tag{8.16}$$

Za zmínku stojí ještě zdůraznit, že u sinusově napájených strojů (synchronní i asynchronní), je v každém okamžiku napájena každá fáze motoru definovaným napětím. Vstupem do modelu může být přímo napětí a není potřeba modelovat měnič. U BLDC motoru je situace komplikovanější z toho důvodu, že vždy jedna fáze není napájena, je plovoucí. Pro korektní simulovaní BLDC motoru je nutno tento problém nějak řešit. Situace byla vyřešena tak, že model měniče simuluje nenapájené fázi takové napětí, aby v ní byl zaručen nulový proud. Tímto je zajištěno, že na této fázi je napětí jakoby definované motorem.

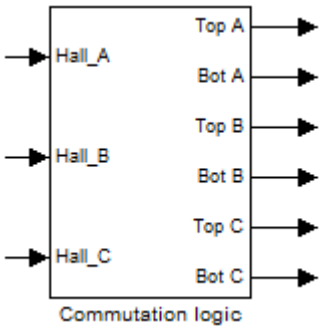
V přílohách na obrázku 12.2 je zobrazena celková struktura simulačního modelu. Jednotlivé bloky zajišťující různé funkce jsou zleva tyto:

- Komutační logika (Commutation logic)
- Měnič (Power Inverter)
- Přepočet napětí ze svorkového na fázové (Voltage conversion)
- Elekrická část (Electrical part)
- Mechanická část včetně výpočtu indukovaného napětí a stavu hallových senzorů (Mechanical part + EMFs + Halls)

Popíšeme si nyní jednotlivé bloky:

8.2.1 Komutační logika

Vstupem do tohoto bloku jsou tři signály z hallových sond. V bloku je realizována komutační tabulka pomocí logických hradel obdobně jako na obrázku 5.1 a výstupem je šest signálů pro spínání tranzistorů.



Obrázek 8.1: Komutační logika

8.2.2 Model měniče

Správná funkce měniče je nezbytná pro simulaci běhu BLDC motoru. Měnič je poměrně komplikovaný, skládá se ze tří podobných částí (obr.: 8.2), které reprezentují jednotlivé větve skutečného měniče.

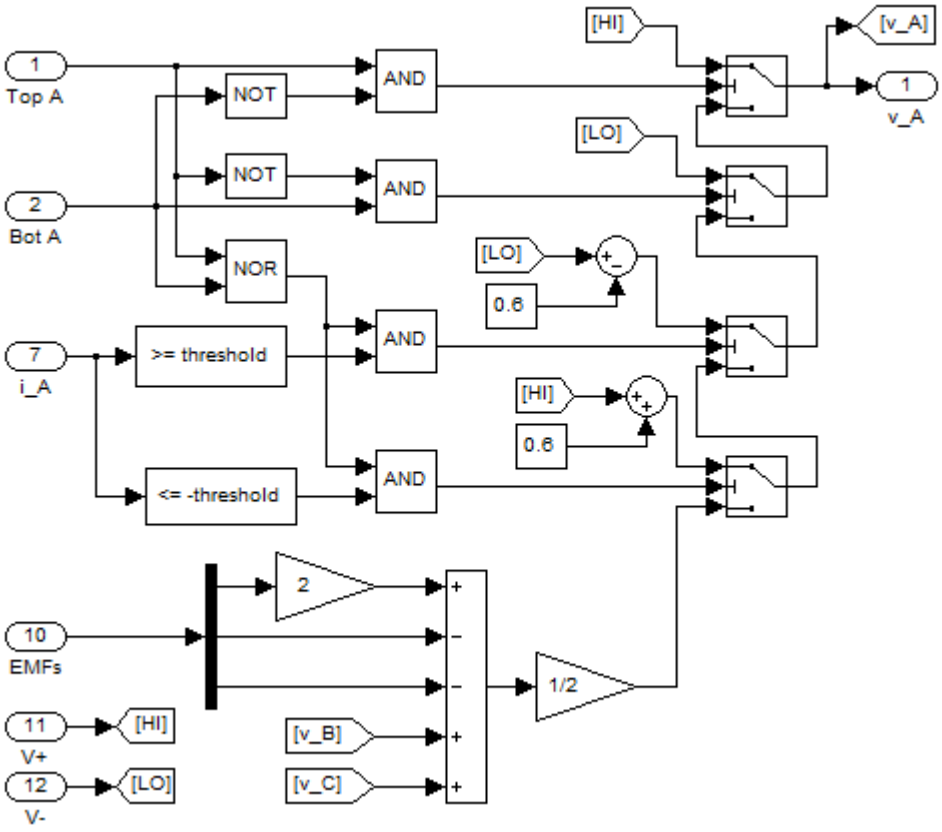
Základní vstupy měniče jsou signály pro spínání tranzistorů a výstupy by měly být jednotlivá svorková napětí. Pro správnou funkci měniče je ale potřeba znát ještě jedotlivé fázové proudy a indukovaná napětí (ve schématu jsou označena jako EMFs, anglicky se označuje jako Back Electromotive Force). Na modelu jedné větve měniče je vidět, že výstupní napětí jde přes čtyři přepínače (multiplexory), rozlišujeme tedy pět stavů. Zhora dolů to jsou tyto případy:

- Horní tranzistor je sepnut a dolní vypnut, na výstupu se objeví horní napájecí napětí.
- Dolní tranzistor je sepnut a horní vypnut, na výstupu se objeví spodní napájecí napětí (0 V).
- Oba tranzistory jsou vyplé a proud je kladný, proud musí téct přes diodu paralelní se spodním tranzistorem. Na výstupu se musí objevit spodní napájecí napětí snížené o úbytek napětí na diodě (-0,6 V).
- Oba tranzistory jsou vyplé a proud je záporný, proud musí téct přes diodu paralelní s horním tranzistorem. Na výstupu se musí objevit horní napájecí napětí zvýšené o úbytek napětí na diodě.

- Oba tranzistory jsou vyplé a proud je nulový. Napětí podle rovnice 8.17 je vlastně napětí generované motorem a zajišťuje nulový proud.

Ve skutečnosti není porovnávání proudu s nulou, ale s velmi malou hodnotou 0.01 A. Vytvoří se tak jisté pásmo kolem nuly, které je důležité pro správnou funkci simulace. Algoritmus není tak citlivý na problém zero-crossingu (průchodu nulou).

$$u_a = \frac{u_b + u_c - e_b - e_c + 2e_a}{2} \tag{8.17}$$



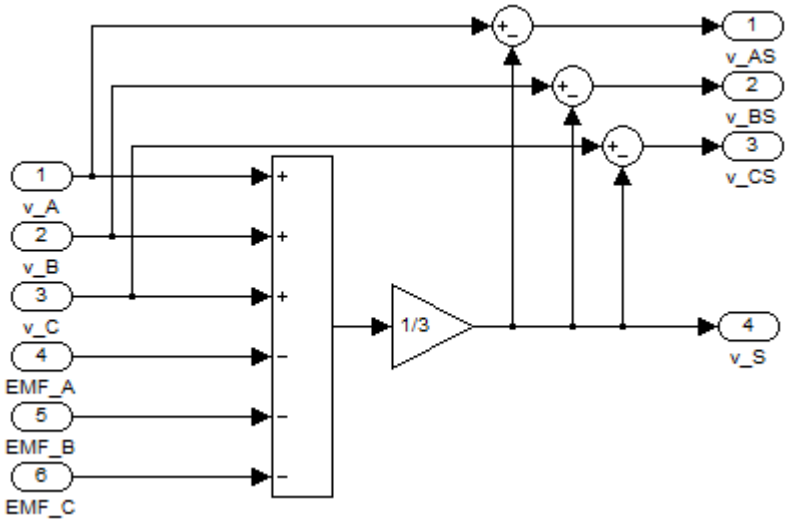
Obrázek 8.2: Model jedné větve měniče

8.2.3 Přepočet svorkového napětí na napětí jednotlivých fází

Tento blok zajišťuje přepočet ze svorkového napětí na napětí v jednotlivých fázích motoru. Jedná se o realizaci rovnic:

$$u_{as} = u_a - u_s \tag{8.18}$$
$$u_{bs} = u_b - u_s \tag{8.19}$$
$$u_{cs} = u_c - u_s \tag{8.20}$$

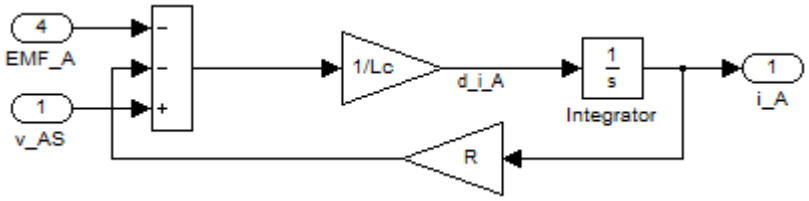
Středové napětí u_s je dáno rovnicí 8.16.



Obrázek 8.3: Realizace přepočtu svorkového napětí

8.2.4 Elektrická část BLDC motoru

Na obrázku 8.4 je zobrazena reprezentace jedné fáze motoru podle rovnice 8.6.

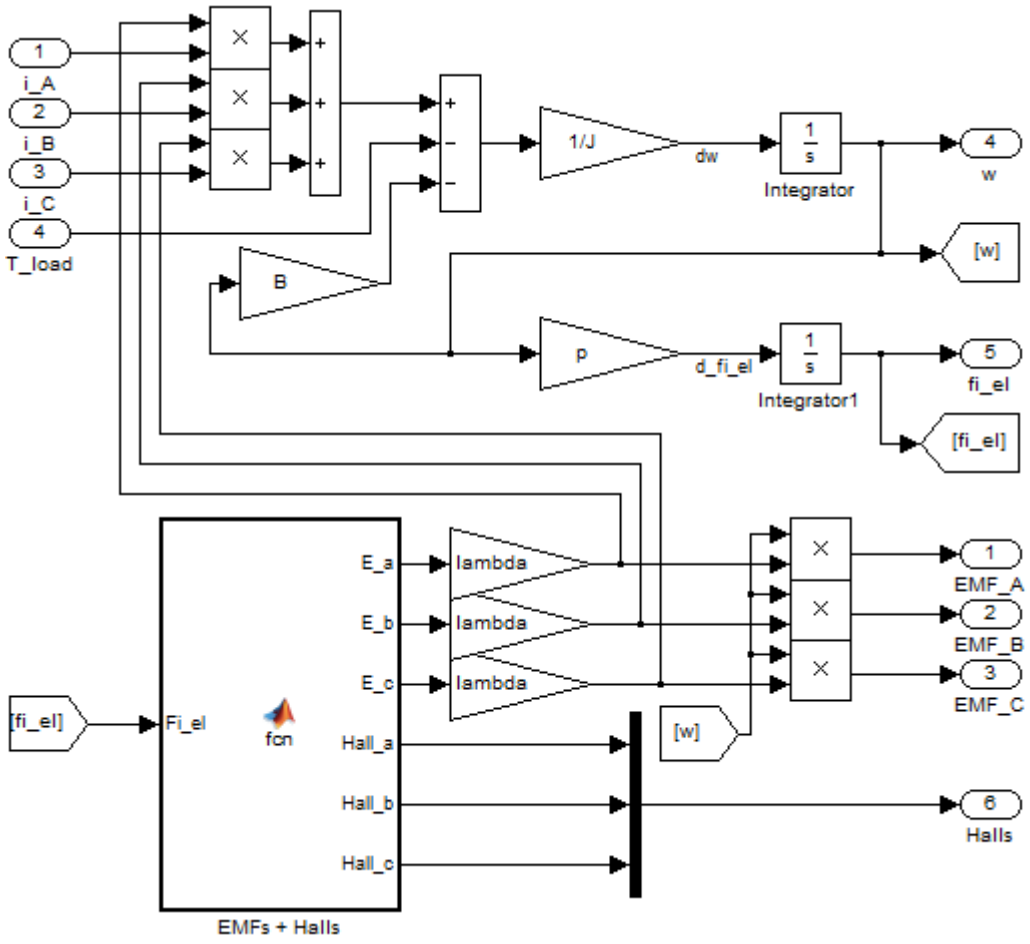


Obrázek 8.4: Reprezentace rovnice jedné fáze motoru

8.2.5 Mechanická část BLDC motoru

Blok mechanické části BLDC motoru obsahuje reprezentaci rovnice momentové rovnováhy 8.10, výpočet elektrického natočení (rovnice 8.12), výpočet indukovaného napětí (rovnice 8.9), vnitřního momentu (rovnice 8.11) a stavu hallových senzorů během elektrické otáčky. Vstupy do bloku jsou proudy pro potřeby výpočtu vnitřního momentu a zátěžný moment. Výstupy pak jsou mechanická úhlová rychlost, elektrické natočení, indukovaná napětí a stavy hallových senzorů.

Jednotková funkce závislosti indukovaného napětí a závislost stavu hallových senzorů na elektrickém natočení jsou naprogramovány pomocí Matlab funkce.



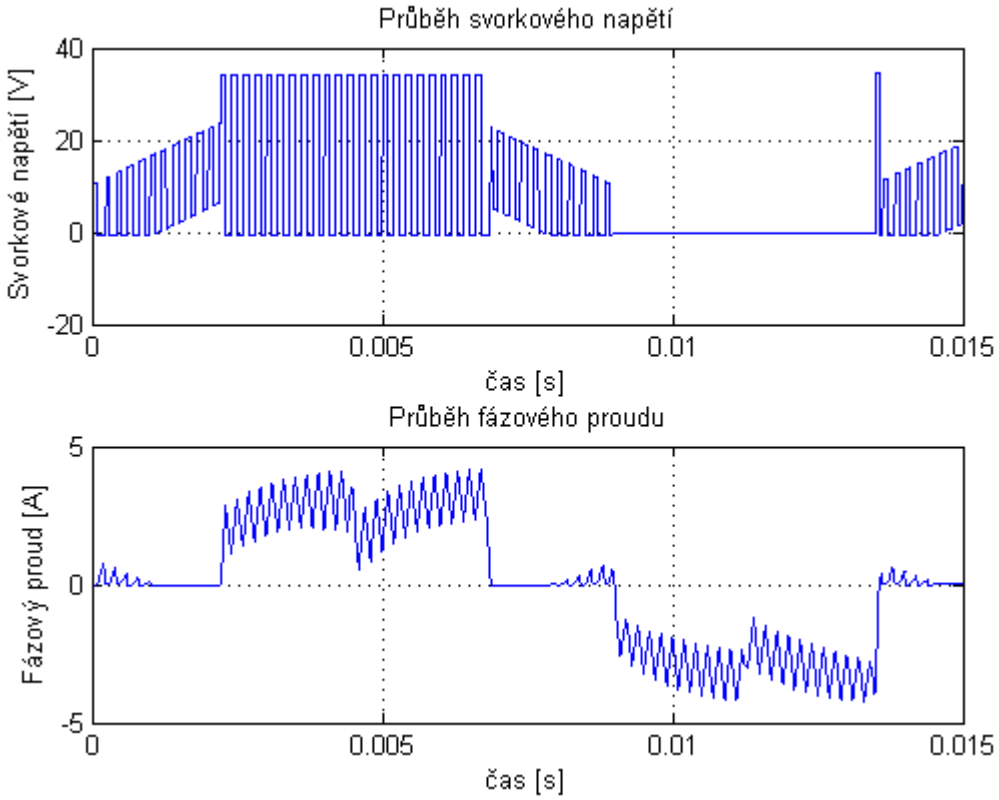
Obrázek 8.5: Reprezentace mechanické části BLDC motoru

8.2.6 Výsledky simulace BLDC motoru

Po skompletování modelu jsme přistoupili k vlastní simulaci s použitím parametrů naidentifikovaných pro námi použitý motor v [13]. Parametry motoru jsou tyto:

$R = 0,5\ \Omega$	odpor vinutí
$L_s = 0,47\ mH$	indukčnost
$\lambda = 0,0573\ V \cdot s$	napěťová konstanta
$J = 0,00004\ kg \cdot m^2$	moment setrvačnosti
$B = 0,000188\ N \cdot m \cdot s$	koeficient viskózního tření
$p = 4$	počet pólových dvojic

Průběhy simulovaných hodnot modelu odpovídají skutečnému chování BLDC motoru i chování komutující fáze je správné. Úspěšně dopadlo i chování simulačního modelu při řízení horních tranzistorů pomocí PWM. Objevovaly se jisté potíže se zerro-crossingem. V simulaci je tento problém poměrně častý a zřejmě se nám nepodařilo zerro-crossing správně nastavit. Pomohli jsme si zjemněním kroku výpočtu. Tento problém se objevoval převážně při PWM řízení.



Obrázek 8.6: Simulované průběhy napětí a proudu BLDC motoru při PWM řízení

Na grafech průběhu simulovaného napětí a proudu (obr.: 8.6) je dobře vidět zvlnění proudu nejen od PWM, ale také jeho chování při přepínání fází a z toho vyplývající zvlnění momentu. Na grafu průběhu napětí stojí za pozornost špičky napětí, které se objevují na začátku přechodu do plovoucí fáze. Proud který tekl horním nebo spodním tranzistorem po vypnutí nemůže kvůli indukčnosti ihned zaniknout a najde si cestu diodou paralelní k druhému tranzistoru. Tím se na chvilku jakoby připojí k opačnému napětí. Délka této komutační špičky ja závislá na zatížení motoru, tedy velikosti proudu.

8.3 Real-time model pro HIL simulaci

V předchozích kapitolách o matematickém modelu a simulačním modelu BLDC motoru jsme si připravovali půdu pro hlavní úkol. Tím je zajistit, aby model motoru byl schopen běžet v reálném čase a tím mohl sloužit pro HIL simulaci.

Časová konstanta elektrického obvodu $\tau_{el} = L/R$ je menší než 1 mS a elektrický obvod je často řízen pulsním PWM signálem o frekvenci až desítek kHz, perioda takového signálu může být tedy menší než $0,1mS$. Aby byla simulace užitečná a funkční, musí být krok výpočtu mnohem menší (alespoň o řád) než nejmenší časová konstanta systému a také mnohem menší než změny (perioda) vstupního signálu. Pro simulaci BLDC motoru je tedy nutný krok výpočtu alespoň $10\mu S$, lépe však mnohem menší. Tím se však dostáváme za možnosti dnešních RCP a HIL systémů založených na procesorech a musíme využít FPGA.

Přechod z procesorových systému na FPGA je poměrně komplikovaný. Budeme nyní mluvit o generování kódu ze Simulinku pro procesorové systémy a pro FPGA pomocí System Generatoru for DSP. Při generování kódu pro procesorové systémy máme k dispozici téměř celou základní knihovnu Simulinku + Stateflow. Při přechodu na FPGA máme mnohem omezenější počet operací (bloků). Nemáme zde blok pro integraci, podporu maticových operací. Musíme pracovat s čísly s pevnou desetinnou čárkou, potýkáme se s problémy s dodržením časování FPGA (pepilining), velikost FPGA čipu je také omezená.

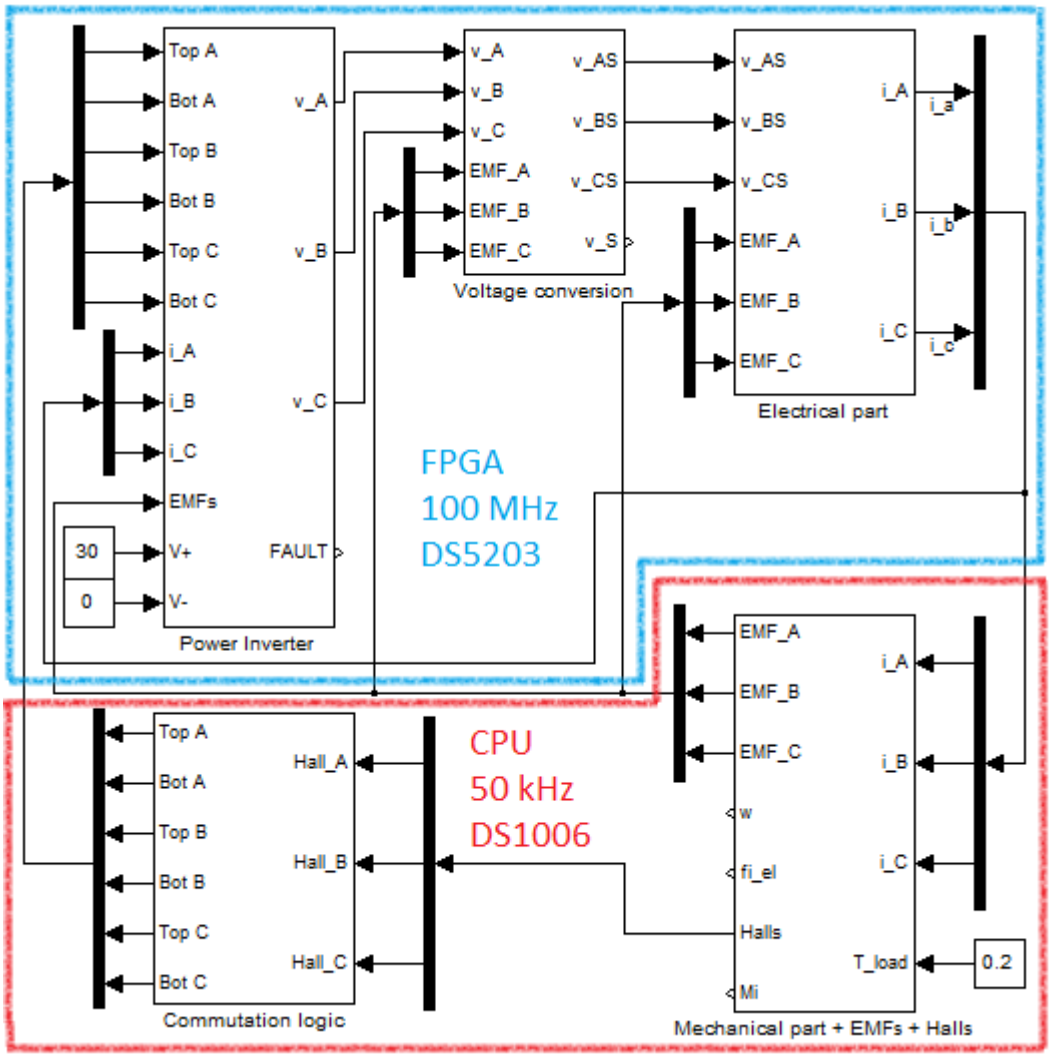
Nemusíme ale předělávat celý model pro FPGA. Pracujeme na modulárním dSPACE zařízení, které má výpočetní procesorovou kartu a FPGA kartu propojenou velice rychlou PHS sběrnicí, která nám umožňuje sdílet 32-bitové data mezi FPGA a procesorem. Simulační model si rozdělíme na části, které musí běžet velmi rychle, tedy na FPGA a na části, které mohou běžet pomaleji na procesoru. Části, které musí bežet velmi rychle na FPGA jsou:

- Měnič (Power Inverter)
- Přepočet napětí ze svorkového na fázové (Voltage conversion)
- Elekrická část (Electrical part)

Části, které mohou běžet na procesoru:

- Komutační logika (Commutation logic)
- Mechanická část včetně výpočtu indukovaného napětí a stavu hallových senzorů (Mechanical part + EMFs + Halls)

Tím, že jsme si model rozdělili na FPGA a procesorovou část získáme i tu výhodu, že část na procesoru se modeluje (programuje) jednodušeji, efektivněji a může být mnohem komplexnější.



Obrázek 8.7: Struktura a rozdělení RT modelu pro HIL simulaci

8.3.1 Přechod od simulačního modelu k RT modelu pro HIL

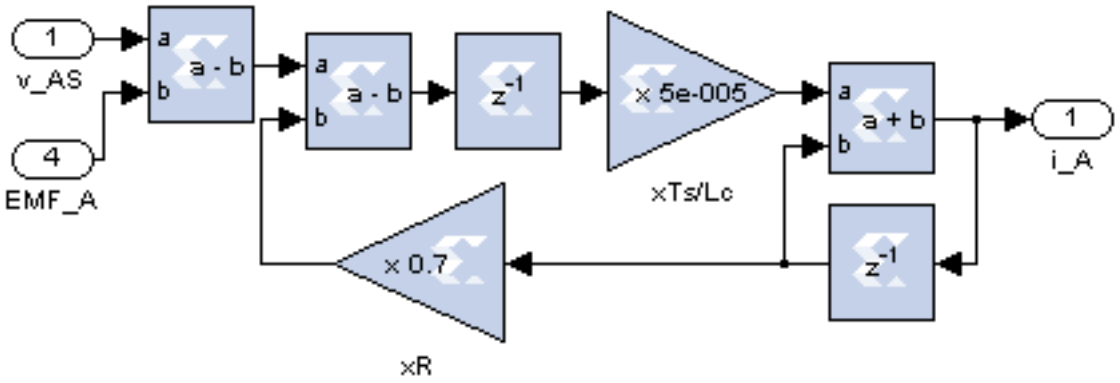
Nejprve jsme rozdělili simulační model na části, které se budou implementovat na FPGA a na části pro procesor a těm jsme potom nastavili rozdílnou velikost výpočetního kroku. Hodinový signál pro FPGA je 100 MHz a této maximální frekvence jsme také využili a nastavili výpočetní krok pro FPGA část na 10 nS. Výpočetní krok pro procesorovou část jsme nastavili na 20 μS. Očekáváme, že výpočetní smyčka na procesoru bude schopna běžet na 50 kHz. Mezi tyto dvě části s rozdílným výpočetním krokem jsme namodelovali dopravní zpoždění, snažili jsme se postihnout komunikační zpoždění mezi procesorem a FPGA, které jsme si na dSPACE skutečně přibližně změřili. Komunikace je ale velmi rychlá (trvá kolem 150 nS) a nečekali jsme žádné problémy. Ještě spojité integrace byly nahrazeny diskrétními. Po spuštění tohoto modelu s rozdílnými výpočetními kroky model stále správně fungoval a nevyskytly se žádné problémy.

Dalším krokem bylo nahrazení datových typů s plovoucí desetinnou čárkou datovými typy s pevnou desetinnou čárkou. Nahrazování probíhalo pouze u částí, které se budou provádět na FPGA. V simulačním modelu jsme zjistili rozsahy, ve kterých se pohybují všechny signály při očekávaných vstupech a podle toho jsme zvolili nastavení počtu bitů před a za desetinnou čárkou u pevného datového typu. Co se týká celkového počtu bitů jednotlivých signálů, tady jsme většinou použili 48 bitů a měnili jsme pouze počty před a za desetinnou čárkou. Nebyli jsme omezeni využitím FPGA čipu, a proto jsme volili spíše vyšší rozlišení. Po provedení simulace se celkové chování modelu nezměnilo a tak jsme mohli přistoupit k vlastnímu procesu transformace pro FPGA a procesorovou část.

Transformace pro procesorovou část proběhla velmi rychle. S využitím dSPACE RTI jsme části ze simulačního modelu využili téměř beze změny. Pouze jsme z knihovny dSPACE RTI FPGA použili bloky pro komunikaci s FPGA částí. Části modelu pro FPGA bylo ale třeba přepsat s využitím bloků knihovny System Generatoru. Protože v simulačním modelu byly použity základní bloky Simulinku, téměř ke všem existoval blok ekvivalentí. Pouze numerickou integraci jsme si museli vytvořit ze základních bloků. Ale protože jsme použili Eulerovu numerickou integraci prvního řádu, bylo to velmi jednoduché a spolehlivé. Pro představu je zobrazena reprezentace jedné fáze motoru v SysGenu pro FPGA (obr.: 8.8) podle rovnice 8.6. Lze srovnat s reprezentací v Simulinku (obr.: 8.4).

8.3.2 Zhodnocení výsledků HIL simulace

Po sestavení modelu a nahrání do dSPACE proběhla úspěšně simulace. Dosud jsme neměli možnost ověřit model se skutečnou a samostatnou jednotkou pro řízení BLDC motoru, ale ověření proběhlo se simulovanou jednotkou, což je vlastně jen komutační tabulka a PWM řízení. Pro řízení experimentu jsme si vytvořili kontrolní rozhraní v



Obrázek 8.8: Reprezentace rovnice jedné fáze motoru v SysGenu

ControlDesku, přes které můžeme měnit parametry a monitorovat průběhy jednotlivých signálů (obr.: 12.1).

Takto koncipovaný RT model, kde mechanická část běží na procesorovém systému s krokem výpočtu 20 μS vyhovuje do asi 30000 elektrických otáček za minutu. Pro simulaci při vyšších otáčkách by bylo potřeba celý model transformovat na FPGA. Využití FPGA čipu pouze elektrickou částí a komunikací s procesorem je velmi malé a velká část čipu je dostupná pro případné budoucí rozšiřování a zpřesňování modelu. Z tabulky 8.1 vidíme, že nebyla využita ani jedna vestavěná násobička (DSP48 slice). Veškerá násobení je násobení konstantou, které se rozvine ve využití pouze LUT.

Tabulka 8.1: Využití FPGA čipu při HIL simulaci

Počet obsazených Slice	2376 z 14720
Počet obsazených Registrů	2636 z 58880
Počet obsazených LUT	6490 z 58880
Počet obsazených DSP48 Slice	0 z 640
Obsazení Block RAM	0 kB z 8784 kB

9 Závěr

Práce se zabývá využitím FPGA pro poměrně rozsáhlou oblast týkající se senzorového i bezsenzorového řízení BLDC motorů a řízením motoru v tzv. degradovaném módu, tj. snaží se zachovat funkčnost pohonu i při jednom nefunkčním hallově senzoru. Cílem bylo především prozkoumat, jak se projeví výhody aplikace FPGA. Mezi výhody FPGA proti miktrokontrolerům běžně používaným jako řídicí člen patří především vysoká výpočetní frekvence, nízká odezva a možnost paralelizace výpočetního procesu.

Senzorové řízení BLDC motoru byl první úkol, se kterým jsme se potýkali. Implementace senzorového řízení na FPGA nepřinesla žádné výhody především proto, že algoritmus je velmi jednoduchý. Za určitou přednost lze považovat možnost vytvořit si PWM o téměř libovolném rozlišení a z toho vyplývající frekvence. Při volbě rozlišení PWM signálu nejsme vázáni na mocniny dvou. Také jsme měli větší možnosti při zpracovávání informace o rychlosti otáčení motoru.

Při bezsenzorovém řízení jsme si vybrali takovou metodu, aby se co nejvíce projevily výhody FPGA. Byla zvolena metoda sychronizace načítání ADC s PWM. Díky velkým možnostem digitálního zpracování a filtrace signálu byla tato metoda schopna řídit motor od poměrně nízkých otáček. Také určitého zlepšení bylo dosaženo při přechodových stavech, rychlých změnách otáček. Ale zlepšení proti procesorově založeným systémům nebylo příliš velké. A v zájmu spolehlivosti je výhodnější pro provoz při nízkých otáčkách nebo při vysoké dynamice pohonu používat senzorovou metodu.

Při řízení motoru v degradovaném módu se naplno projevily výhody FPGA. Pro porovnání jsme měli pouze jednu práci, která se zabývala stejným úkolem. Představený algoritmus ale nebyl příliš optimální a trpěl mnoha nedostatky, protože byl implementován na procesoru. Úpravou algoritmu a implementací na FPGA jsme odstranili mnoho nedostatků. Při simulované poruše hallova senzoru nebyl na první pohled patrný rozdíl v chování pohonu ani při rychlých změnách otáček. Pouze fungování motoru v okolí nulových otáček není s použitým algoritmem možné.

Hlavním úkolem práce byla HIL simulace BLDC motoru, která vyžaduje model schopný běžet v RT. Korektní simulace elektrických částí BLDC motoru vyžaduje alespoň o řád vyšší výpočetní frekvenci než je frekvence vstupního PWM signálu. Takto vysoká výpočetní frekvence na procesorových systémech není dosažitelná. Aplikace FPGA je nutná. Díky tomu, že máme k dispozici HW dSPACE s procesorem a FPGA, tak mohl být model rozdělen mezi tyto dvě časti. Části s rychlou dynamikou (elektrická část) na FPGA a časti s pomalejší dynamikou (mechanická část) na procesor. Takto pojatý model byl ověřen se simulovanou řídicí jednotkou BLDC motoru a korektně fungoval. Tím, že se počítá mechanická část BLDC

motoru na procesoru jsme poněkud omezili maximální otáčky, kterých může simulace dosáhnout. Omezení je asi na 30000 elektrických otáček za minutu. V případě potřeby vyšších otáček je nutné i mechanickou část BLDC motoru implementovat na FPGA.

Během práce byly použity vysokoúrovňové nástroje pro programování procesoru i pro design FPGA. Návrh veškerých algoritmů byl poměrně efektivní a rychlý. Cenné byly i zkušenosti a informace získáné během práce s FPGA. Na záčátku práce jsme neměli téměř žádné informace a zkušenosti s FPGA. S využitím nástroje System Generator for DSP jsme byli v poměrně krátké době schopni vytvářet komplexní algoritmy.

10 Použité zdroje

[1] Asma Ben Rhouma, Ahmed Masmoudi, Ahmed Elantably: *On the analysis and control of a three-switch three-phase inverter-fed brushless DC motor drive*, COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, Vol. 26 Iss: 1, pp.183 - 200, 2007

[2] Ward Brown: AN857 - *Brushless DC Motor Control Made Easy*, Microchip Technology Inc:, http://ww1.microchip.com/downloads/en/AppNotes/00857B.pdf, 2002

[3] Vorel, P.: *Synchronní stroje s permanentnímy magnety*, Akademické nakladatelství CERM, Brno, ISBN 80-7204-417-6, 2005

[4] Skalický, J.: *Elektrické servopohony*, FEKT, VUT v Brně, ISBN 80-214-1978-4, 2002

[5] Pragasen Pillay: *Application Characteristics of Permanent Magnet Synchronous and Brushless dc Motors for Servo Drives*, IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, VOL. 21, NO. 5, 1991

[6] University of Strathclyde and Steepest Ascent: *DSPforFPGA Primer teaching materials*, Xilinx University Program, Q4 2011

[7] *DS5203 FPGA Board Overview*, dSPACE GmbH :, http://www.dspace.com/en/pub/home/products/hw/modular_hardware_introduction/i_o_boards/ds5203_fpga_board.cfm, 2013-05

[8] *Virtex-5 Family Overview*, Xilinx, Inc.:, http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, 2009

[9] *System Generator for DSP Overview*, Xilinx, Inc.:, http://www.xilinx.com/tools/sysgen.htm, 2013-05

[10] *LabVIEW FPGA Module*, National Instruments Corporation:, http://sine.ni.com/nips/cds/view/p/lang/cs/nid/11834, 2013-05

[11] *HDL Coder Overview*, The MathWorks, Inc.:, http://www.mathworks.com/products/hdl-coder/, 2013-05

[12] Lamberg K., Wältermann P.: *Using HIL Simulation to Test Mechatronic Components in Automotive Engineering*, dSPACE GmbH:, www.dspaceinc.com/ftp/papers/hdt00_e.pdf, 2000

[13] Križan, J.: *Bezsenzorové řízení BLDC motoru*, Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012

[14] Schulte, T.; Bracker, J.: *Real-time simulation of BLDC motors for hardware-in-the-loop applications incorporating sensorless control*, Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on , vol., no., pp.2195,2200, 2008

[15] Nagano, T.; Abourida, S.; Harakawa M.; Belanger, J.; Yamasaki H.; Dufour, C.: *Real-Time Simulation of a Complete PMSM Drive at* 10 μS *Time Step*, InternationalPower Electronics Conference, Niigata, Japan, 2005

[16] Alecsa, B.; Onea, A.: *An FPGA implementation of a brushless DC motor speed controller*, Design and Technology in Electronic Packaging (SIITME), 2010 IEEE 16th International Symposium for , vol., no., pp.99,102, 23-26, 2010

[17] Kos, Dejan; Curkovic, M.; Jezernik, K.: *FPGA Based BLDC Motor Current Control with Spectral Analysis*, Power Electronics and Motion Control Conference, 2006. EPE-PEMC 2006. 12th International , vol., no., pp.1217,1222, 2006

[18] *Brushless DC motors 1W-1300W General Catalogue*, Transmotec:, http://www.transmotec.com/download/Brushless-DC-Motors/No-Gear/Square-Type/B-Series/Complete-B-Series-Catalogue.pdf, 2010-12

[19] *dsPICDEM MC1L 3-Phase Low Voltage Power Module User's Guide*, Microchip Technology Inc:, http://ww1.microchip.com/downloads/en/devicedoc/70097A.pdf, 2003

[20] *Modular Hardware*, dSPACE GmbH:, http://www.dspace.com/en/pub/home/products/hw/modular_hardware_introduction.cfm, 2013-05

[21] *MechLab - Laboratoř mechatroniky*, Ústav mechaniky těles, mechatroniky a biomechaniky, Fakulta strojního inženýrství, Vysoké učení technické v Brně:, mechlab.fme.vutbr.cz, 2013-05

[22] Wook-Jin Lee, Seung-Ki Sul: *A New Starting Method of BLDC Motors Without Position Sensor*, Industry Applications, IEEE Transactions on , vol.42, no.6, pp.1532,1538, Nov.-dec. 2006

[23] Ming Lu, Lifu Yu, Yaohua Li: *New reliable sensorless startup strategy of brushless DC motor for large inertia system*, Electrical Machines and Systems (ICEMS), 2012 15th International Conference on , vol., no., pp.1,4, 21-24, Oct. 2012

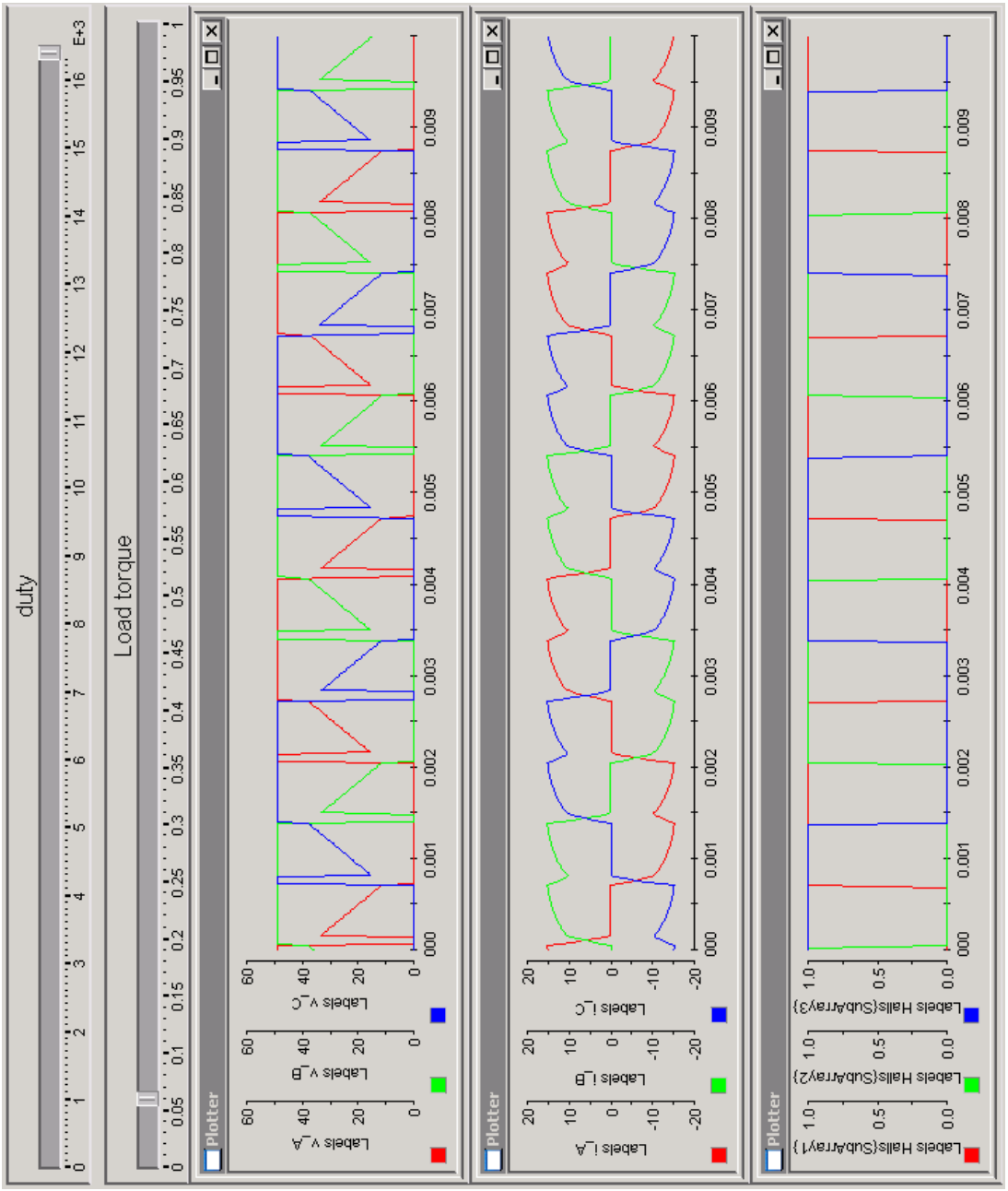
[24] AN2030 - *BACK EMF DETECTION DURING PWM ON TIME BY ST7MC*, STMicroelectronics:, http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/application_note/CD00043112.pdf, 2013-05

[25] Shao J., Nolan D., Hopkins T: *A Novel Direct Back EMF Detection for Sensorless Brushless DC (BLDC) Motor Drives*, Applied Power Electronics Conference and Exposition, 2002. APEC 2002. Seventeenth Annual IEEE , vol.1, no., pp.33,37 vol.1, 2002

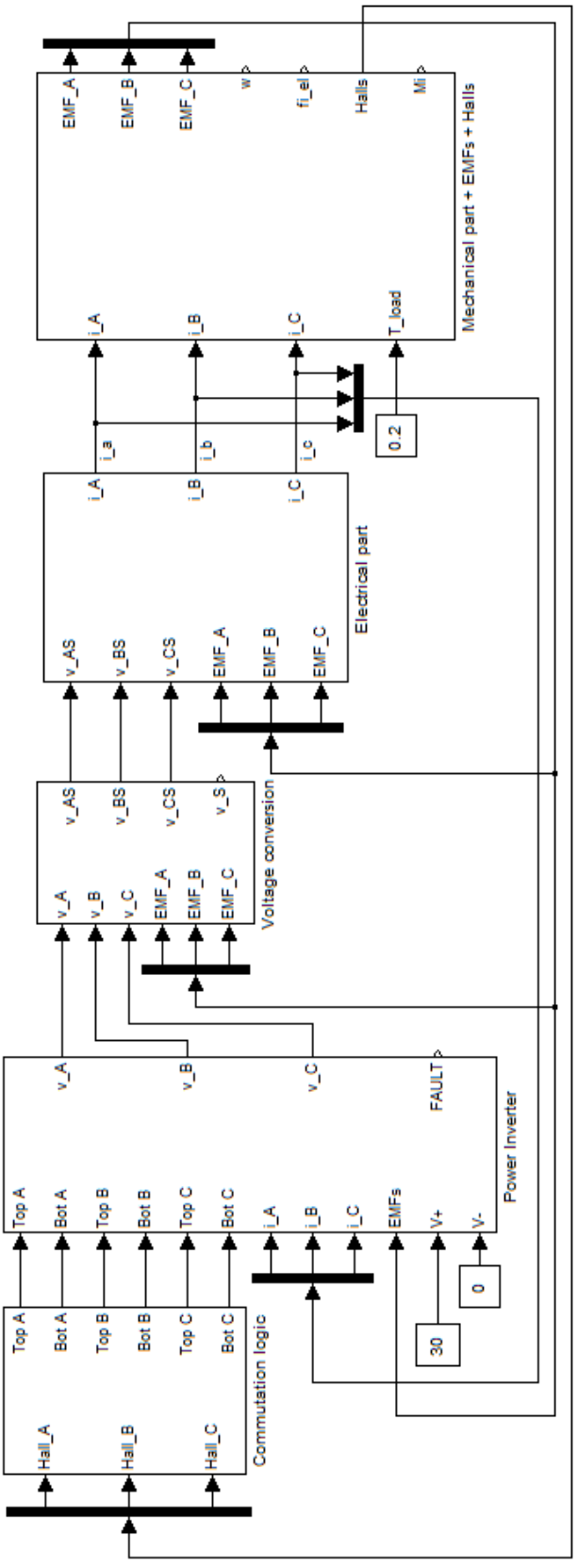
11 Seznam použitých zkratek

$^{\circ}el.$	stupeň elektrický
AD - Analog to digital	Analogový na digitální
ADC - Analog to digital conveter	A/D převodník
ASIC - Application-specific integrated circuit	zákaznický integrovaný obvod
BLDC - Brushless direct current	stejnosměrný bezkartáčový
CPU - Central Processing Unit	procesor
DAC - Digital to analog	D/A převodník
DSP - Digital signal processing	digitální zpracování signálu
EC - Electronically commutated	elektronicky komutovaný
ECU - Electronic control unit	elektronická řídicí jednotka
EMF - Electromotive force	indukované napětí
FPGA - Field programmable gate array	programovatelné hradlové pole
HDL - Hardware description language	jazyk popisující hardware
HIL - Hardware in the loop	nepřekládá se
HW - Hardware	nepřekládá se
I/O - Input/Output	vstup/výstup
LUT - Lookup table	nepřekládá se
MSPS - Megasamples per Second	milión vzorků za sekundu
PWM - Pulse-width modulation	pulsně šířková modulace
RAM - Random-access memory	nepřekládá se
RCP - Rapid Control Prototyping	nepřekládá se
ROM - Read-only memory	nepřekládá se
RT - Real-time	nepřekládá se
SW - Software	programové vybavení
VHDL - VHSIC Hardware Description Language	nepřekládá se

12 Přílohy



Obrázek 12.1: Řízení HIL simulace v ControlDesku



Obrázek 12.2: Struktura simulačního modelu BLDC motoru