SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

Ing. Michal Kocúr

Autoreferát dizertačnej práce

MODERNÉ METÓDY A ALGORITMY AUTOMATICKÉHO RIADENIA REALIZOVANÉ POMOCOU FPGA ŠTRUKTÚR

na získanie akademickej hodnosti doktor (philosophiae doctor, PhD.)

v doktorandskom študijnom programe: **Mechatronické systémy**

v študijnom odbore 5.2.16. mechatronika

Miesto a dátum: Bratislava, 24.06.2016

SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Ing. Michal Kocúr

Autoreferát dizertačnej práce

MODERNÉ METÓDY A ALGORITMY AUTOMATICKÉHO RIADENIA REALIZOVANÉ POMOCOU FPGA ŠTRUKTÚR

na získanie akademickej hodnosti doktor (philosophiae doctor, PhD.)

v doktorandskom študijnom programe: Mechatronické systémy

Miesto a dátum: Bratislava, 24.06.2016

Na Oddelení informačných, komunikačných a riadiacich systémov,
Ústave automobilovej mechatroniky,
Fakulty elektroniky a informatiky STU v Bratislave

Predkladateľ: Ing. Michal Kocúr,
Ústav automobilovej mechatroniky, OIKR
FEI STU BA, Ilkovičova 3, 812 19 Bratislava

Školiteľ: prof. Ing. Štefan Kozák, PhD.
Ústav automobilovej mechatroniky, OIKR
FEI STU BA, Ilkovičova 3, 812 19 Bratislava

Oponenti: prof. Ing Boris Rohaľ-Ilkiv, CSc.
Ústav automatizácie, merania a aplikovanej informatiky
SJF STU BA, Námestie slobody 17, 812 31 Bratislava

doc. Ing. Tibor Krajčovič, PhD.
Ústav počítačového inžinierstva a aplikovanej informatiky
FIIT STU BA, Ilkovičova 2, 842 16 Bratislava 4

Autoreferát bol rozoslaný: .....

Obhajoba dizertačnej práce sa koná: 25. 08. 2012 o hod.

Na Faklulte elektrotechniky a informatiky STU v Bratislave, Ilkovičova 3, 812 19 Bratislava

Anotácia

Názov práce: Moderné metódy a algoritmy automatického riadenia realizované pomocou FPGA štruktúr

Kľúčové slová: FPGA, SoC, Prediktívne riadenie, MPC, PID, systém DC motorov, Xilinx Artix-7, Zynq 7000 SoC

Predložená práca je zameraná na návrh, realizáciu a overenie moderných metód automatického riadenia s využitím obvodov FPGA. Pre realizáciu moderných algoritmov automatického riadenia je v práci navrhnutý všeobecný postup a metodika vývoja a implementácie hardvérových a softvérových modulov a prostriedkov, potrebných pre efektívne riadenie z využitím FPGA štruktúr. V dizertačnej práci je spracovaný všeobecný postup pre hardvérovú realizáciu tak diskrétnych PID algoritmov riadenia ako aj prediktívnych algoritmov riadenia s uvažovaním a bez uvažovania ohraničení na riadiaci zásah. Spracované algoritmy boli verifikovane na fyzikálnych laboratórnych modeloch, ktoré predstavuje sústava DC motorov. Pre realizáciu algoritmov bol použitý FPGA obvod Artix-7 a Zynq 7000 SoC. Realizované algoritmy potvrdili vhodnosť navrhovanej metodiky pre aplikácie moderných metód automatického riadenia a ich využitie pre široké spektrum riadenia procesov s rýchlou dynamikou pre real time aplikácie.

Abstract

Title: Modern methods and algorithms of automatic control realized on FPGA

Key words: FPGA, SoC, Predictive control, MPC, PID, DC motor system, Artix-7, Zynq 7000 SoC

The dissertation thesis deals with the design, realization and verification of modern control methods using FPGA. A general approach for implementation and effective realization of hardware and software modules has been developed, and modern algorithms have been implemented using FPGA. The dissertation focuses on development of a general approach for hardware realization of discrete PID as well as predictive control algorithms considering both unconstrained and constrained control outputs. Developed algorithms were verified on the physical laboratory plants built from two DC motors. The algorithms were realized using the Artix-7 FPGA and Zynq 7000 SoC. Realized algorithms verified suitability of proposed methodology of application of modern control methods and their real-time implementation for high-speed dynamics processes.

Tézy dizertačnej práce

1. Analýza a súčasného stavu výskumu, vývoja a implementácie moderných metód automatického riadenia pre procesy s rýchlou dynamikou.

2. Návrh metód pre hardvérovú realizáciu algoritmov automatického riadenia na vnorených mikropočítačových systémoch a obvodoch FPGA.

3. Realizácia metód automatického riadenia prostredníctvom hardvérovej realizácie na FPGA obvodoch.

4. Overenie pokročilých metód automatického riadenia realizovaných prostredníctvom FPGA pre procesy s rýchlou dynamikou.

Obsah

Úvod

Na implementáciu algoritmov riadenia do riadiacich systémov založených na digitálnych technológiách je možné využiť dva základné prístupy. Prvým z nich je prístup založený na procesore, ktorý sekvenčne vykonáva inštrukcie podľa programu. Program je uložený vo forme strojového kódu v operačnej pamäti. Výpočtové prostriedky postavené na softvérovej implementácií sú napr. programovateľné logické automaty, mikrokontroléry, mikroprocesory, digitálne signálne procesory a pod.

Druhý prístup je založený na hardvéri. Prvé hardvérové implementácie boli realizované na báze magnetických relé. Po príchode nových technológií ako napr. tranzistorov a DPS sa začalo využívať riadenie s logickými hradlami. Zvyšujúce sa možnosti integrácie umožnili čoraz zložitejšie zapojenie logických hradiel združovať do jedného ASIC čipu. ASIC sú zákaznícke integrované obvody, ktorých proces výroby trvá niekoľko mesiacov.

FPGA sú obvody slúžiace na implementáciu logických funkcií, ktoré majú na rozdiel od ASIC predpripravenú sieť logických hradiel, multiplexorov a preklápacích obvodov, ktoré sú pospájané programovateľnými prepojeniami. Pomocou prepojení je možné navrhnutý obvod fyzicky „zadrôtovať“. Spojenia nie sú trvalé a je možné ich preprogramovanie. Na FPGA obvodoch je možné implementovať riadiace systémy s vysokou dynamikou. Využitím aritmetiky s pevnou desatinnou čiarkou a štruktúrami pre paralelné výpočty dokážu FPGA niektoré algoritmy riešiť výrazne efektívnejšie ako konvenčné mikroprocesory. Stále je pritom zachovaná možnosť preprogramovania (Kocúr, 2013).

Programovanie funkcionality obvodov FPGA je značne odlišné od programovania mikrokontrolérov. Mikrokontrolér má funkcie ako radiče prerušenia, čítače, časovače, A/D a D/A prevodníky pripravené na použite. Ich konfigurácia je zväčša možná jednoduchou zmenou konfiguračného registra. Navrhnutý kód v jazyku C pracuje na rovnakých princípoch ako na obyčajných desktopových aplikáciách. Na rozdiel od mikrokontrolérov obvody FPGA nemajú tak bohato predpripravenú funkcionalitu. Jazyk VHDL je jazykom na opis hardvéru a funguje na výrazne odlišných princípoch ako napr. jazyk C. Najvýraznejším rozdielom je, že príkazy v kóde sa vykonávajú paralelne a nie sekvenčne riadok po riadku, ako sme u konvenčných programovacích jazykov zvyknutí. Výsledným produktom alogritmu riadenia totiž nie je spustiteľný binárny súbor pre procesor, ale zapojenie kombinačných a sekvenčných logických prvkov, ktoré sa môžu vyjadriť RTL schémou.

Výhodou takéhoto návrhu aplikácií na FPGA je možnosť využiť modulárny prístup. Keďže má obvod dostatočný počet IO portov, je teoreticky možné navrhnúť aj viac aplikácií na jednom čipe bez toho, aby jedna aplikácia akokoľvek ovplyvňovala druhú. Aplikácie budú od seba funkcionálne oddelené. Táto decentralizácia patrí medzi hlavné špecifiká programovateľných logických obvodov. Navyše každý modul môže byť opísaný iným jazykom pre opis hardvéru. Vývojové prostredia ISE a Vivado podporujú jazyky VHDL a Verilog. Návrh zapojenia je možný aj priamo vytváraním grafickej RTL schémy z logických hradiel a preklápacích obvodov.

1 Hardvérová realizácia PSD algoritmov

V tejto časti práce je ukázaná a prezentovaná hardvérová realizácia PSD algoritmov riadenia s uvažovaním ohraničení na riadiaci zásah.

1.1 Návrh PSD regulátora prepočtom z PID regulátora

Parametre diskrétneho regulátora môžeme navrhovať priamo v diskrétnej oblasti napríklad na základe zvoleného kritéria kvality alebo rozmiestnením pólov. Existuje množstvo metód návrhu koeficientov PID alebo PSD regulátora, ktoré môžeme nájsť v dostupnej literatúre (Kozák, 1991). Ďalší používaný spôsob návrhu je prepočet zo spojitého PID regulátora (Kocúr, 2013). Úlohou je prepočet PID regulátora na diskrétny PSD pri dobe periódy vzorkovania T. Uvažujme teda o spojitom PID regulátore, ktorého akčný zásah v časovej oblasti môžeme vyjadriť ako:

$$u(t) = P\left\{e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right\} \quad (1.1)$$

Kde: K je zosilnenie,T_i je integračná časová konštanta, T_d je derivačná časová konštanta.

Pri zvolenej dobe vzorkovania T rovnicu diskretizujeme na diferenčnú rovnicu, pri ktorej sme použili obdĺžnikovú náhradu. Dostaneme vzťah:

$$u(k) = P\left\{e(k) + \frac{T}{T_i}\sum_{i=0}^{k-1} e(i) + \frac{T_d}{T}[e(k) - e(k-1)]\right\} \quad (1.2)$$

Tento typ algoritmu nazývame pozičný. Na realizáciu ale nie je vhodný, pretože pri každom výpočte akčného zásahu treba do sumy pripočítavať nové hodnoty. To znamená, že po konečnom čase by premenná do ktorej sumu ukladáme mohla naraziť na maximum vymedzené svojím dátovým typom a pretiecť. Aby sme sa vyhli tomuto nežiaducemu javu, využijeme rýchlostnú (rekurzívnu) formu PSD algoritmu. Akčný zásah v kroku k je vyjadrený v rovnici (1.2). V kroku $k-1$ (predchádzajúci krok) akčný zásah vyjadríme ako

$$u(k-1) = P\left\{e(k-1) + \frac{T}{T_i}\sum_{i=0}^{k-2} e(i) + \frac{T_d}{T}[e(k-1) - e(k-2)]\right\} \quad (1.3)$$

Ich rozdiel potom je

$$u(k) - u(k-1) = P\left\{e(k) - e(k-1) + \frac{T}{T_i}[e(k-1)] + \frac{T_d}{T}[e(k) - 2e(k-1) + e(k-2)]\right\} \quad (1.4)$$

Akčný zásah v kroku k vypočítame:

$$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \quad (1.5)$$

Ak použijeme obdĺžnikovú náhradu potom

$$q_0 = P\left(1 + \frac{T_d}{T}\right) \qquad q_1 = -P\left(1 - \frac{T}{T_i} + 2\frac{T_d}{T}\right) \qquad q_2 = P\frac{T_d}{T} \quad (1.6)$$

Týmto spôsobom si vieme prepočítať diskrétny PSD regulátor zo spojitého PID regulátora, ktorého koeficienty si môžeme navrhnúť vhodnou metódou (Kocúr, 2011).

1.2 Modifikácia PSD algoritmu pri ohraničení riadiaceho zásahu.

Pri riadení reálneho systému môže nastať situácia že regulátor vypočíta väčší akčný zásah ako môže fyzicky realizovať. Sú to napríklad obmedzenia tlakov, teplôt, prietoku a podobne. V našom prípade je obmedzený akčný zásah tým, že DC-motor dokáže prijať vstupné napätie v rozmedzí 0 až 12 V. Po skokovej zmene žiadanej hodnoty *w* je regulačná odchýlka pomerne veľká a akčný zásah ,ktorý vypočíta regulátor je väčší ako 12 V. V skutočnosti sa nám riadiaci zásah orezáva na tejto hodnote. Pri použití PSD regulátora je ale stále integračná zložka aktívna a pričítava nové hodnoty *k* akčnému zásahu. Ak sa regulačná odchýlka zníži natoľko, že by regulátor mal generovať akčný zásah pod hodnotou orezania, nestane sa tak okamžite. Zabráni tomu totiž integračná zložka, ktorá počas nasýtenia pričítavala hodnoty k akčnému zásahu a teraz ich spätne odpočítava. V obvode môže vzniknúť preregulovanie a zároveň sa predĺži aj doba regulácie. Tento jav sa označuje v teórií automatického riadenia „windup“. Existuje niekoľko spôsobov ako sa tento nežiadúci jav odstraňuje (Kozák, 1991).

Môžeme ho eliminovať úpravou riadiaceho algoritmu a to tak že budeme nulovať integračnú zložku ak $u(k) \geq u_{max}$ alebo $u(k) \leq u_{min}$. Podobnou úpravou je metóda, podľa ktorej do rýchlostného algoritmu nevkladáme minulú hodnotu riadiaceho zásahu ktorý bol vypočítaný, ale riadiaci zásah ktorý bol realizovaný.

$$u(k) = u_m(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \tag{1.7}$$

Kde

$$u_m(k-1) = \begin{cases} u_{min}, & \rightarrow \; ak \quad u(k-1) < u_{min} \\ u(k-1) & \rightarrow \; ak \quad u_{min} \leq u(k-1) \leq u_{max} \\ u_{max} & \rightarrow \; ak \quad u(k-1) > u_{max} \end{cases} \tag{1.8}$$

Pri tomto druhu ochrany už nedochádza k veľkému preregulovaniu, čo je pre windup príznačné, no výpočet riadiaceho zásahu zostáva prakticky nezmenený a neberie do úvahy prípadné ohraničenia. Riadiaci zásah tak nie je optimálny.

Ak vypočítaný riadiaci zásah regulátora je rozdielny od realizovaného, lineárny regulačný dej sa mení na nelineárny. PSD algoritmus vo forme (1.7) sa s touto situáciou vysporiadať nedokáže, a preto sa musí modifikovať takým spôsobom, aby bol vypočítaný riadiaci zásah rovnaký ako realizovaný. PSD algoritmus s rešpektovaním ohraničujúcich podmienok na riadenie je odvodený v (Kozák, 1991) a má tvar:

$$\begin{aligned} u(k) = {} & u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \\ & - p_1 u_e(k-1) - p_2 u_e(k-2) \end{aligned} \tag{1.9}$$

Kde

$$q_0 = K_p + K_d \qquad q_1 = -K_p - 2K_d + K_i \qquad q_2 = K_d$$

$$p_1 = \frac{K_i - K_d}{K_p + K_i + K_d} \qquad p_2 = \frac{K_d}{K_p + K_i + K_d} \qquad u_e(k) = u(k) - u_m(k) \tag{1.10}$$

$$u_m(k) = \begin{cases} u_{min} & \rightarrow & ak & u(k) < u_{min} \\ u(k) & \rightarrow & ak & u_{min} \leq u(k) \leq u_{max} \\ u_{max} & \rightarrow & ak & u(k) > u_{max} \end{cases}$$

K_p, K_i, K_d sú proporcionálne, integračné a derivačné zosilnenie regulátora:

$$K_p = P \qquad K_i = PT/T_i \quad K_d = PT_d/T \tag{1.11}$$

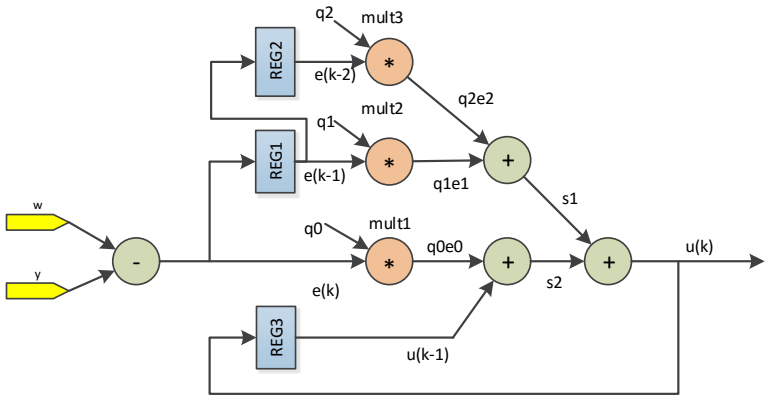
Rekurentný vzťah (1.9) predstavuje súčet dvoch častí. Prvá časť opisuje výpočet riadiaceho zásahu bez obmedzení, druhá predstavuje modifikáciu algoritmu aj s obmedzením hodnôt riadiaceho zásahu.

1.3 Implementácia PSD algoritmov

Pre potreby realizácie základného PSD algoritmu na obvodoch FPGA je potrebné dekomponovať rovnicu (1.7) na jednoduché aritmetické operácie (Kocúr, HW realizácia PID algoritmov na báze FPGA štruktúr., 2013):

$$\begin{gathered} e(k) = w(k) - y(k) \\ q0e0 = q_0 e(k) \\ q1e1 = q_1 e(k-1) \\ e2e2 = q_2 e(k-2) \\ s1 = q2e2 + q1e1 \\ s2 = q0e0 + u(k-1) \\ u(k) = s1 + s2 \end{gathered} \tag{1.12}$$

V tomto prípade je využitý paralelný návrh PSD algoritmu, čo znamená, že každá operácia bude mať svoju aritmetickú jednotku, a to buď sumátor, alebo multiplikátor. Moderné obvody FPGA realizujú sčítanie a násobenie viacbitových signálov pomocou DSP jadier. Pridaním registrov na uchovanie signálov *y*, *e* a *u* pre výpočet akčného zásahu v nasledujúcom kroku sa môže riadiaci algoritmus vyjadriť graficky:

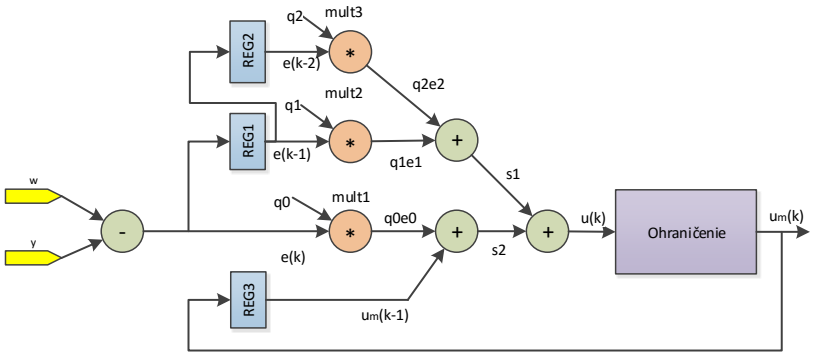


Obr. 1.1 – Štruktúra paralelného návrhu PSD algoritmu

Odčítaním *y* od referenčnej hodnoty *w* sa vypočíta regulačná odchýlka *e* v kroku *k*. Pri nábežnej hrane hodinového impulzu *clk* sa hodnota v registri *REG2* preklopí a *e(k)* sa stáva signálom *e(k-1)*. Analogicky to isté platí v registri *REG2* a *REG3* so signálom *e(k-1)* a *u*. Registre nadobudnú inicializačnú hodnotu 0 po prvom štarte algoritmu alebo po nábežnej hrane signálu *reset*. V bloku *MULT1* sa vynásobí koeficient regulátora q_0 so signálom *e(k)*, vzniká tak signál *q1e1*. Podobne v *MULT2* prebehne násobenie q_1 s *e(k-*

1), a v *MULT3* q_2 s *e(k-2)*. Výsledný akčný zásah *u* v kroku *k* sa získa sčítaním signálov *q0e0*, *q1e1*, *q2e2*, *u(k-1)*.

Doplňme základný PSD algoritmus o jednoduchú anti-windup ochranu podľa (1.7) a (1.8). Do štruktúry regulátora vložíme blok, ktorý bude ohraničovať vypočítaný riadiaci zásah. Tento spôsob je vhodný v situáciách kde vypočítaná hodnota $u(k)$ pomerne dobre súhlasí so skutočnou hodnotou riadiaceho zásahu.

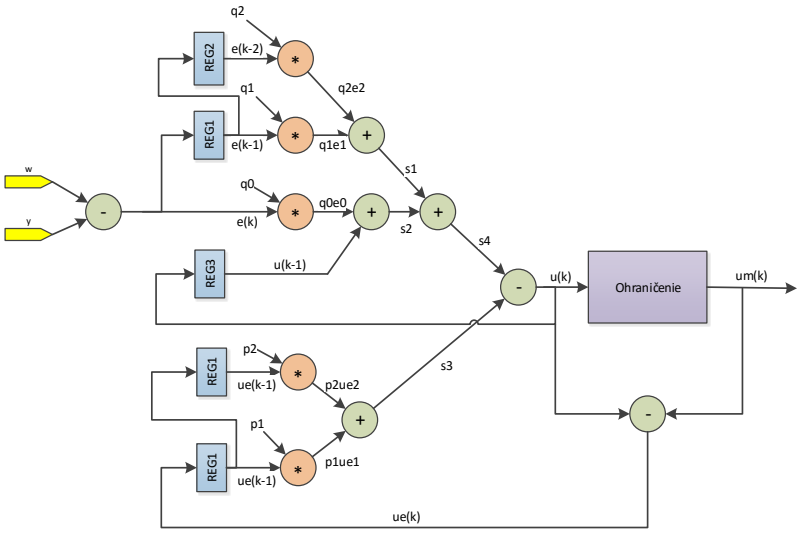


Obr. 1.2 – Štruktúra PSD algoritmu s anti-windup

Dekomponovaním rekurentného PSD algoritmu (1.6), ktorý rešpektuje ohraničenia dostaneme:

	$e(k) = w(k) - y(k)$ $u_e(k) = u(k) - u_m(k)$ $q0e0 = q_0 * e(k)$ $q1e1 = q_1 * e(k-1)$ $e2e2 = q_2 * e(k-2)$ $p1ue1 = p_1 * u_e(k-1)$	$p2ue2 = p_2 * u_e(k-2)$ $s1 = q1e1 + q2e2$ $s2 = q0e0 + u(k-1)$ $s3 = p1ue1 + p2ue2$ $s4 = s1 + s2$ $u(k) = s4 - s3$	*(1.13)*

Dekomponovaný algoritmus (1.13) v grafickom vyjadrení opisuje Obr. 1.3.



Obr. 1.3 - PSD algoritmus rešpektovaním obmedzení

1.4 Verifikácia PSD algoritmov riadenia

Majme dynamický systém DC-motora identifikovaný prenosovou funkciou

$$G_p(s) = \frac{K}{T_1 s + 1} e^{-Ds} = \frac{25{,}28}{0{,}004856 s + 1} e^{-0{,}001 s} = \frac{\Omega(s)}{U(s)} \tag{1.14}$$

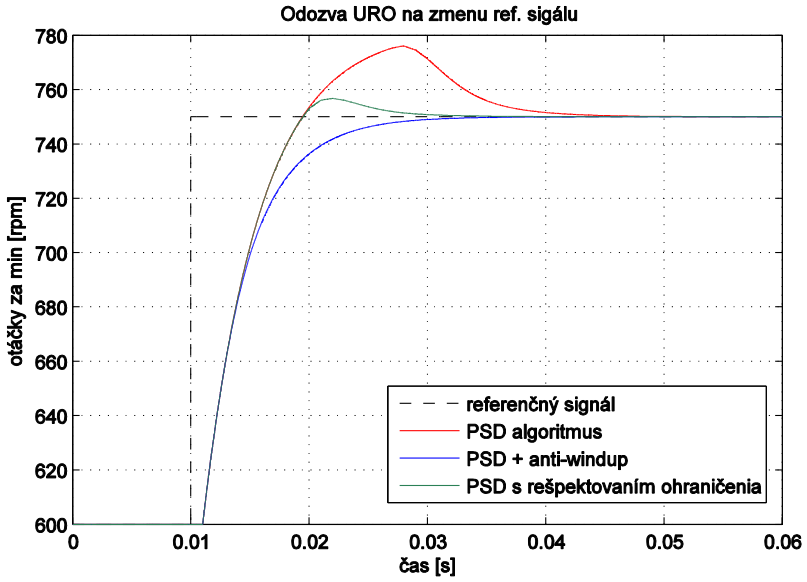
kde vstup do systému predstavuje vstupné napätie motora vo voltoch a výstup predstavuje počet otáčok motora za minútu. Úlohou je riadiť otáčky motora podľa zvoleného referenčného signálu. Diskrétna prenosová funkcia regulátora vypočítaná metódou optimálneho modulu (Kocúr, 2011) pri $T_v = 0{,}001\ s$ má tvar:

$$G_r(z) = \frac{q_0 z + q_1 z^{-1} + q_1 z^{-2}}{1 - z^{-1}} = \frac{0{,}0707 z - 0{,}0561 z^{-1} + 0 z^{-2}}{1 - z^{-1}} \tag{1.15}$$

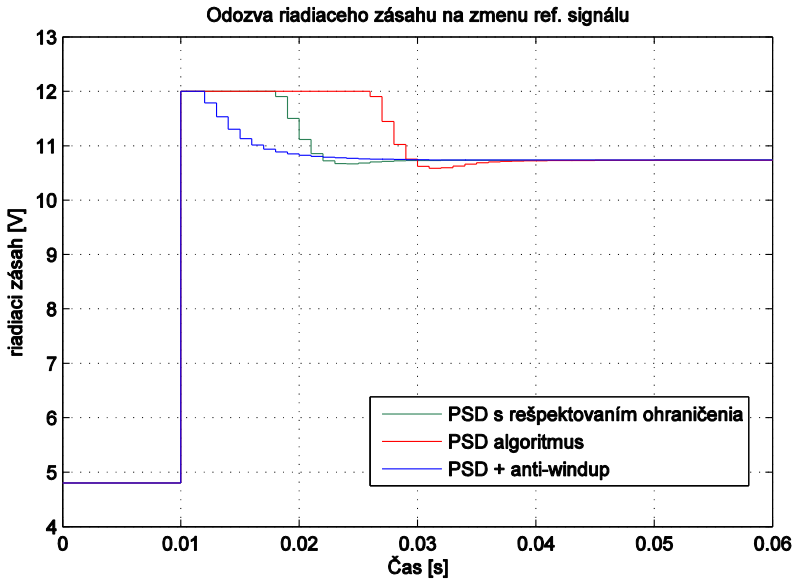
Rekurentná forma PSD algoritmu je potom

$$\begin{aligned} u(k) &= u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \\ &= u(k-1) + 0{,}0707 e(k) + 0{,}0561 e(k-1) \end{aligned} \tag{1.16}$$

Aplikujme všetky tri formy PSD algoritmu, ktoré sme si predstavili v predošlej kapitole, na riadenie v regulačnom obvode so zápornou spätnou väzbou. Porovnajme regulované veličiny z hľadiska ustáleného (veľkosť regulačnej odchýlky) a prechodového stavu (kritérium ISE (Corriou, 2004)). Odozvy URO a riadiaceho zásahu môžeme vidieť na Obr. 1.4 a Obr. 1.5.



Obr. 1.4 – Odozva URO na zmenu referenčného signálu



Obr. 1.5 – Odozva riadiaceho zásahu na zmenu referenčného signálu

Tab. 1.1 – Kvalita riadenia v ustálenom a prechodovom stave

Algoritmus	***e(∞)***	***ISE***
PSD – základný	0	67,51
PSD – anti-windup	0	64,96
PSD – s rešpektovaním ohraničenia	0	62,63

2 Implementácia a hardvérová realizácia moderných metód riadenia

V tejto časti práce opíšeme implementáciu vybraných moderných metód riadenia: metódy rozmiestňovania pólov, IMC regulátora a robustného regulátora založeného na metóde reflexných polynómov. Uvedené metódy sú implementované a realizované pre riadenie malého DC-motora opísaného v kap. 2.1.

2.1 Opis riadeného systému - DC motor

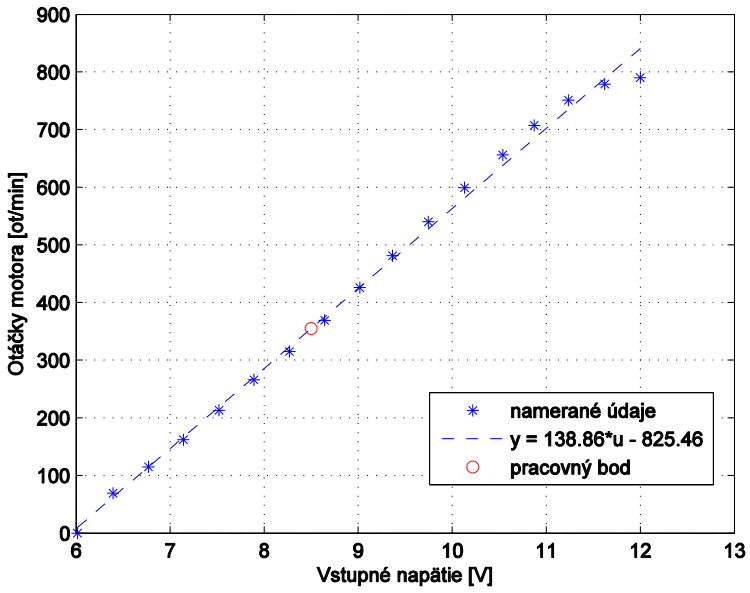
Riadený systém reprezentuje 12V jednosmerný motor zobrazený na Obr. 2.1. Na motorčeku je umiestnená prevodovka s pomerom 1:19. Maximálne vstupné napätie je v rozmedzí 0 – 12 V. Technické parametre motorčeka môžeme nájsť v technickej špecifikácií (Shayang Ye Industrial Co.,LTD., 2013).

Obr. 2.1 – Jednosmerný motor

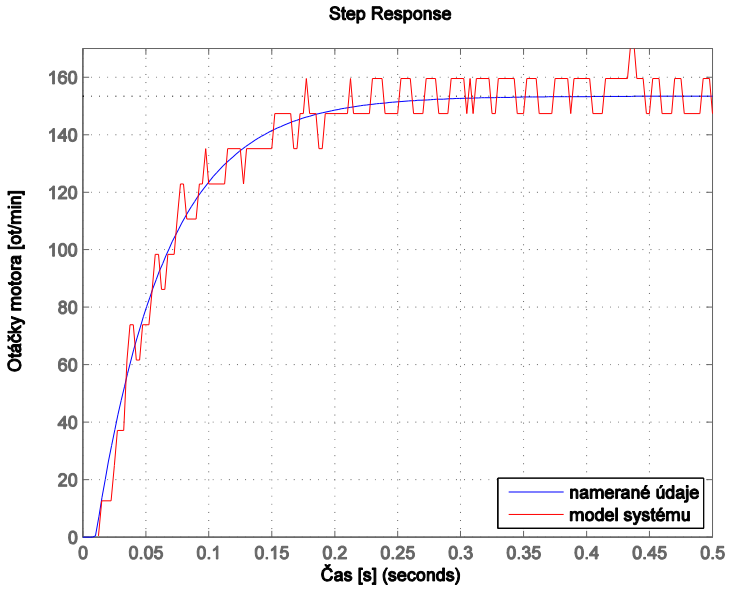
Veľkosť vstupu ovládame šírkou impulzu - PWM moduláciou. Výstup systému je reprezentovaný rýchlosťou otáčania hriadeľa za prevodom reprezentované v otáčkach za minútou [ot/min]. Rýchlosť otáčania motora sledujeme pomocou magnetické enkodéra. Enkodér s halovými sondami dokáže detegovať rýchlosť (pred prevodom) a smer otáčania hriadeľa. Z prevodovej charakteristiky (Obr. 2.2) sme vybrali pracovný bod v oblasti 350 ot/min. V tomto bode sme skokom vstupnej veličiny vyvolali odozvu výstupnej veličiny. Z nameraných údajov sme pomocou dvojbodovej metódy identifikácie získali prenosovú funkciu modelu jednosmerného motora:

$$G_p(s) = \frac{K}{T_1 s + 1} e^{-Ds} = \frac{153{,}4}{0{,}07392s + 1} e^{-0{,}01s} = \frac{\Omega(s)}{U(s)} \qquad (2.1)$$

Kde K je zosilnenie systému, T_1 je časová konštanta systému a D je dopravné oneskorenie. Odozvu modelu na jednotkový skok sme na Obr. 2.3 porovnali s nameranými údajmi (pôvodné dáta boli upravené na jednotkový skok).



Obr. 2.2 – Prevodová charakteristika DC motora



Obr. 2.3 – Odozva systému na jednotkový skok

2.1.1 Návrh regulátora metódou rozmiestňovania pólov

Návrh vychádza z diskrétneho modelu DC motora (2.2), ktorý bol prepočítaný zo spojitej prenosovej funkcií (2.1) zo vzorkovacou periódou *Ts = 0.01 s.*

$$G_P(z) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{19.41z^{-2}}{1 - 0.8735z^{-1}} \tag{2.2}$$

Parametre diskrétneho regulátora boli vypočítané z Diofantovej rovnice zvolením rádu čitateľa regulátora $r_n = 1$ a menovateľa regulátora $r_d = 2$. Zvolený polynóm $A_{CH}(z)$ je opisuje správanie sa uzavretého regulačného obvodu zo stabilnými pólmi: 0,4; 0,6; 0,9; a 1. Posledný pól bol zvolený na zabezpečenie nulovej trvalej regulačnej odchýlky. Diskrétna prenosová funkcia regulátora má potom formu

$$G_{c\_PP}(z) = \frac{0.01117 - 0.009936z^{-1}}{1 - 1.0265z^{-1} + 0.0265z^{-2}} \tag{2.3}$$

2.1.2 Implementácia regulátora metódou rozmiestňovania pólov

Zákon riadenia, ktorý vychádza z diskrétnej prenosovej funkcie regulátora a má tvar:

$$u(k) = q_0 e(k) + q_1 e(k-1) - p_1 u(k-1) - p_2 u(k-2) \tag{2.4}$$

kde parametre $\{q_i, p_i\}$ nadobúdajú nasledovné hodnoty: $q_0 = 0{,}01117$; $q_1 = -0{,}009936$; $p_1 = -1{,}0265$; $p_2 = 0{,}0265$.

Na rozdiel od PS resp. PSD regulátora nie je štruktúra zákona riadenia pevne definovaná. Stupeň polynómu v čitateli a menovateli sa môže v jednotlivých realizáciách meniť, preto musíme k jednotlivým

návrhom pristupovať individuálne. Pri realizácií treba brať ohľad aj na veľkosť koeficientov q_i a p_i, a to kvôli použitiu fixed-point aritmetiky. V tomto konkrétnom prípade sme zvolili rozsah pre q_0, q_1, p_2 SFIXED(0,20) – jeden bit pre znamienko, nula bitov pre miesta pred rádovou čiarkou a 20 bitov pre miesta za rádovou čiarkou. Pre parameter p_1 sme zvolili rozsah SFIXED(1,20).

Vstupom do regulátora je regulačná odchýlka e, ktorá je realizovala ako rozdiel žiadanej rýchlosti w a aktuálnej rýchlosti motora y. Oba tieto signály sú neznamienkové a majú rozsah 12bitov. Signál e má potom 13bitov (0 bitov pre desatinné miesta) a môže nadobúdať aj záporné hodnoty. Kvôli kompatibilite má vstupný signál e dátový typ *std_logic_vector(11 downto 0)* . Ten si prevedieme na typ SFIXED príkazmi

```
signal err        :     sfixed(12 downto 0) := to_sfixed(0,12,0);
...
err <= to_sfixed(e,err);
```

V prvom riadku ukážky je zavedený SFIXED(12,0) signál *err* do ktorého sme priradili hodnotu vstupného signálu *e*. Konverziu zabezpečila funkcia *to_sfixed().*

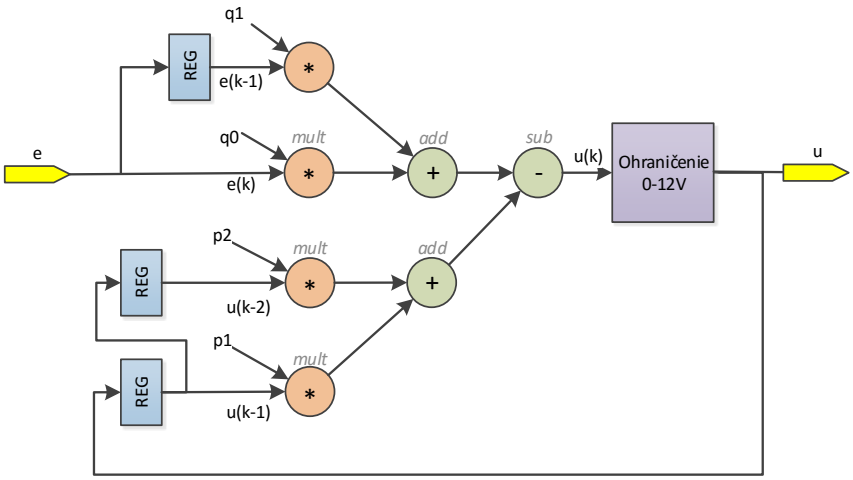
Regulačnú odchýlku postupne posúvame pri každej nábežnej hrane hodinového impulzu do registra *e(k-1) – e* v kroku *k-1*, ktorý je tiež využívaný riadiacom zákone. Podobne ukladáme aj hodnoty riadiaceho zásahu *u* do registrov *u(k-1)* a *u(k-2).*

```
    -- Realizácia registrov
    Reg_proc        :        process(clk, rst)
    begin
        if rst = '1' then
            e_k1 <= to_sfixed(0,e_k1);
            u_k1 <= to_sfixed(0,u_k1);
            u_k2 <= to_sfixed(0,u_k2);
        elsif rising_edge(clk) then
            e_k1 <= err;            --Reg1
            u_k1 <= u_k0;           --Reg2
            u_k2 <= u_k1;           --Reg3
        end if;
    end process Reg_proc;
```

Riadiaci zásah v kroku *k* vypočítame sériou príkazov.

```
    --Nasobenie
    q1e_k1    <= e_k1   * q1;
    q0e       <= err  * q0;
    p1u_k1    <= u_k1 * p1;
    p2u_k2    <= u_k2 * p2;
    --Scitavanie
    s1        <= q1e_k1 + q0e;
    s2        <= p1u_k1 + p2u_k2;
    s3        <= s1 - s2;
```

Kvôli prehľadnosti sme si zvolili metódu kde každý riadok pripadá na jednu aritmetickú operáciu násobenia alebo sčítania. V kompletnej RTL schéme algoritmu v prílohe A.2 dizertačnej práce vidíme, že sú operácie oddelené aj v samotnej schéme zapojenia obvodu. Zjednodušenú hardvérovú realizáciu algoritmu môžeme vidieť na Obr. 2.4.



Obr. 2.4 – Realizácia Pole-Placement algoritmu

Pri jednotlivých operáciách sčítania a násobenia boli dodržané pravidlá pre rozsah výsledných signálov. Výsledný signál *s3* nadobudol rozsah SFIXED(15,32), výstup regulátora je ale obmedzený na 12bitov, preto musíme signál *s3* vhodne ohraničiť. Dovolený rozsah akčného zásahu regulátora je 0 - 12 Voltov. Hodnota $12_{(10)}$ je prepočte do binárneho tvaru $1100_{(2)}$ - na jeho realizáciu potrebujeme 4 bity pre miesta pred rádovou čiarkou. Ostatné bity vyhradíme miesta za rádovou čiarkou. Keďže akčný zásah nemá dovolené nadobúdať záporné hodnoty bit pre znamienko nie je potrebný. Signál môžeme ohraničiť a orezať manuálne

```
    u_k0    <= u_max when s3 > u_max else
               u_min when s3 < u_min else
               to_sfixed(to_slv(s3(4 downto -7)),u_k0); --do registra
    u     <=  to_slv(u_sf);                             --na vystup
```

alebo môžeme použiť funkciu zaokrúhlenia, čo je presnejšie, no za cenu väčšej hardvérovej náročnosti.

```
    s3        <= resize(s1-s2,4,-7);
      ...
    u_k0    <= u_max when s3 > u_max else
               u_min when s3 < u_min else
               s3;                                        --do registra
    u     <=  to_slv(s3);                                 --na vystup
```

Táto technika zároveň slúži aj ako anti-windup ochrana.

Hardvérová realizácia uzavretého regulačného obvodu je opísaná v kap. 2.2. Verifikáciu riadiaceho algoritmu môžeme nájsť v kapitole 2.3.

2.1.3 Návrh robustného regulátora

Robustný regulátor je navrhnutý tak, aby charakteristický polynóm uzatvoreného regulačného obvodu bol umiestnený v stabilnom polytope (lineárne pokrytie) reflexných vektorov. To znamená, že je treba vyriešiť nasledovné úlohy:

1. výber východiskového polynómu $C(z^{-1})$ pre generovanie polytopu $V(C)$,
2. výber $k+1$najvhodnejších vrcholov polytopu $V(C)$ pre vytvorenie cieľového simplexu S,
3. výber cieľového polynómu $E(z^{-1})$.

V ďalšej časti sú uvedené niektoré „overené pravidlá“ pre výber stabilného cieľového simplexu S. Pri výbere $k+1$ vrcholov cieľového simplexu S sa využíva fakt, že póly s kladnou reálnou časťou sa uprednostňujú pred tými so zápornou reálnou časťou (Ackermann, a iní, 1993). Pozitívne reflexné vektory $v_i^+(C)$ sa vyberajú s i-nepárnym a negatívne reflexné vektory $v_i^-(C)$ naopak s i párnym, čo spolu dáva k vrcholov. $(k+1)$-ty vrchol cieľového simplexu S je vybraný ako stred zostávajúcich reflexných vektorov.

Žiadaný cieľový polynóm $E(z^{-1})$rádu k môže byť v princípe vybraný ľubovoľne. Napriek tomu je nutné, aby bol vybraný z vnútra stabilného polytopu reflexných vektorov $V(C)$. Zaužívaným býva výber $E(z^{-1}) = C(z^{-1})$.

Pre polynómy vyšších rádov býva rozmer cieľového simplexu S značne menší než rozmer polytopu reflexných vektorov V. Je to preto, aby metóda kvadratického programovania s prevoleným cieľovým simplxom S fungovala, len ak sú neučitosti dostatočne malé. Inak je nutné použiť iné metódy pre nájdenie robustného regulátora takého, aby polytop charakteristických polynómov uzavretého obvodu bol umiestnený vo vnútri stabilného polytopu reflexných vektorov $V(C)$.

Navrhnime robustný regulátor pre jednosmerný motor z kapitoly 2.1. Prenosová funkcia riadeného systému - jednosmerného motora je v tvare spojitej prenosovej funkcie je

$$G_P(s) = \frac{K}{T_1 s + 1} e^{-Ds} = \frac{153.4}{0.07932s + 1} e^{-0.01s} \tag{2.5}$$

kde koeficienty K, T_1 sa môžu pohybovať v intervaloch neurčitosti $K \in \langle 150; 160\rangle$, $T_1 \in \langle 0{,}070; 0{,}078\rangle$. Uvedená spojitá prenosová funkcia je po prepočítaní do diskrétnej formy so vzorkovacou frekvenciou $Ts = 0{,}01s$:

$$G_P(z^{-1}) = \frac{19{,}41z^{-2}}{1 - 0{,}8735z^{-1}} \tag{2.6}$$

Našou úlohou je navrhnúť diskrétny regulátor so stupňami polynómov $v = 1, \mu = 2$. Z prenosovej funkcie (2.6) a maticovej reprezentácie získame

$$c = \begin{bmatrix} 0 & 0 & 0 & 19{,}4097 & 0 \\ -0{,}8735 & 0 & 0 & 0 & 19{,}4097 \\ 1 & -0{,}8735 & 0 & 0 & 0 \\ 0 & 1 & -0{,}8735 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ 1 \\ q_1 \\ q_2 \end{bmatrix} \tag{2.7}$$

Zvoľme počiatočný polynóm $C(z^{-1})$ pre generovanie polytopu $V(C)$ nasledovne

$$C(z^{-1}) = 1 + 0{,}1z^{-1} \tag{2.8}$$

s reflexnými koeficientami $r_1 = -0{,}1$, $r_2 = r_3 = r_4 = 0$.

Teraz môžeme nájsť reflexné vektory $v_i(C)$ počiatočného polynómu $C(z^{-1})$ vedenej do maticovej reprezentácie želaného S (vrcholové polynomiálne koeficienty)

$$S = \begin{bmatrix} -2{,}5 & 0 & 0{,}3 & 0{,}15 & 0{,}1 \\ 0 & -0{,}7 & -0{,}3 & 0{,}5 & 0 \\ -0{,}25 & 0{,}5 & 0 & -0{,}5 & 0{,}2 \\ 0{,}5 & 0 & 0{,}4 & -0{,}4 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.9}$$

Úloha návrhu diskrétneho regulátora pre nominálnu prenosovú funkciu systému (2.6) je riešená prostredníctvom kvadratického programovania s koeficientom $\propto = 0{,}05$ v účelovej funkcií J Pre zvolenú žiadaný simplex S, dostávame nasledujúci diskrétny spätoväzobný regulátor

$$G_{FB}(z^{-1}) = \frac{Q(z^{-1})}{P(z^{-1})} = \frac{0{,}03145z^{-1} + 0{,}01117z^{-2}}{1 + \ 0{,}9311z^{-1} + \ 0{,}6679z^{-2}}, \tag{2.10}$$

ktorého riadiaci zákon vyjadrený v rekurzívnej forme je

$$u(k) = 0{,}0315e(k-1) + 0{,}01117e(k-2) - 0{,}931u(k-1) - 0{,}668u(k-2). \tag{2.11}$$

2.1.4 Implementácia robustného regulátora

Rekurzívna forma diskrétneho regulátora (2.11) pre systém jednosmerného motora je vo všeobecnom tvare vyjadrená v tvare

$$u(k) = q_1 e(k-1) + q_2 e(k-2) - \ p_1 u(k-1) - p_2 u(k-2). \tag{2.12}$$

Podobne ako v prípade regulátora založeného na metóde rozloženia pólov, môžu mať pre rôzne modely systému regulátory odlišnú štruktúru. Implementácia riadiaceho zákona je analogická k návrhu PSD regulátora alebo regulátora založeného na metóde rozloženia pólov. Riadiaci algoritmus je impleplementačným softvérom rozložený na jednotlivé aritmetické operácie:

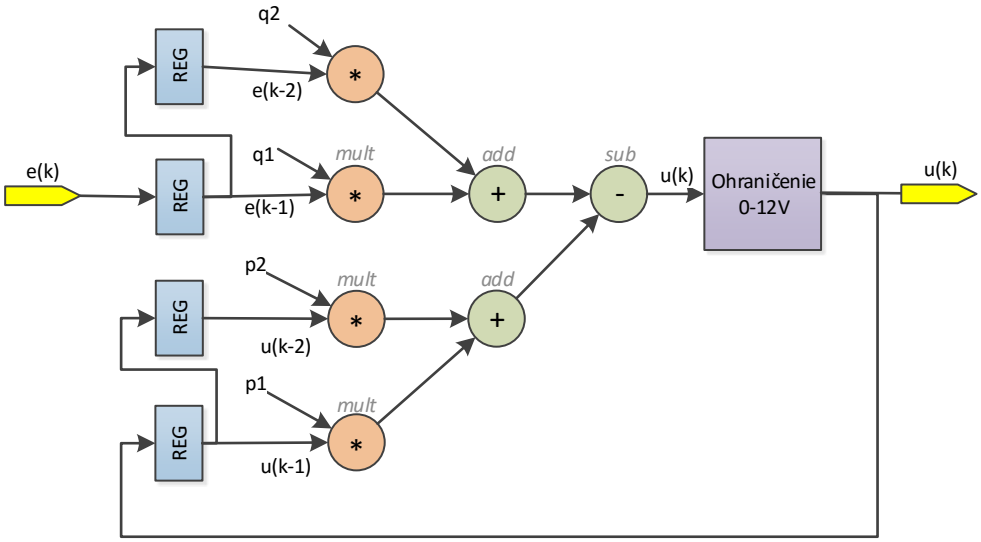
$$\begin{gathered} e(k) = w(k) - y(k) \\ q1e1 = q_1 * e(k-1) \\ e2e2 = q_2 * e(k-2) \\ p1u1 = p_1 * u(k-1) \\ p2u2 = p_2 * u(k-2) \\ s1 = q1e1 + q2e2 \\ s2 = p1u1 + p2u2 \\ s3 = s1 - s2. \end{gathered} \tag{2.13}$$

Rozsahy vstupných a výstupných vektorov sú identické ako v prípade Pole-placement regulátora. Koeficienty regulátora $\{q_i, p_i\}$ nadobúdajú nasledovné hodnoty: $q_1 = \ 0{,}03145;\ q_2 = \ -0{,}01117;\ p_1 = \ -0{,}9311;\ p_2 = 0{,}6679$. Rozsah pre všetky koeficienty bol stanovený na SFIXED(1,10).

Výstup regulátora je ohraničený v rozsahu $\langle u_{min}; u_{max} \rangle$, kde $u_{max} = 12\ V$ a $u_{min} = -12V$, tak že

$$u(k) = \begin{cases} u_{max} & \rightarrow \quad ak \quad s3 > \ u_{max} \\ s3 & \rightarrow \quad ak \quad s3 \in \langle u_{min}; u_{max} \rangle \\ u_{min} & \rightarrow \quad ak \quad s3 < \ u_{min} \end{cases}. \tag{2.14}$$

Riadiaci zákon rozložený na jednotlivé aritmetické operácie je graficky vyjadrený na Obr. 2.5



Obr. 2.5 - Paralelný návrh robustného diskrétneho regulátora

2.1.5 Návrh IMC regulátora pre DC-motor

Rovnako ako pri predchádzajúcich návrhoch regulátorov budeme regulátor navrhovať na riadenie Jednosmerného motora z kapitoly 2.1. Návrh realizujeme v diskrétnej oblasti. Diskrétna prenosová funkcia modelu jednosmerného motora po prepočítaní do diskrétnej formy so vzorkovacou frekvenciou $Ts = 0{,}01s$:

$$G_P(z^{-1}) = \frac{19{,}41z^{-2}}{1 - 0{,}8735z^{-1}} \tag{2.15}$$

Oddeľme model (2.15) na invertibilnú G_p^- a neinvertibilnú G_p^+časť

$$G_p^- = G_P(z^{-1}) \tag{2.16}$$

$$G_p^+ = 1 \tag{2.17}$$

V nasledujúcom kroku zvolíme časovú konštantu filtra τ_c a rád filtra N_f. Zvoľme filter vo forme spojitej prenosovej funkcie

$$G_f(s) = \frac{1}{(0{,}01s - 1)^2} \tag{2.18}$$

Spojitá prenosová funkcia filtra prepočítaná do diskrétnej oblasti s periódou vzorkovania $Ts = 0{,}01s$ je

$$G_f(z^{-1}) = \frac{0{,}6321z^{-1}}{1 - 0{,}3679z^{-1}}. \tag{2.19}$$

Potom diskrétna prenosová funkcia IMC regulátora je v tvare

$$G_{c\_IMC}(z) = \frac{0.3996 - 0.349z^{-1}}{19.41 - 14.28z^{-1} + 2.627z^{-2}} \tag{2.20}$$

Prepočtom (opísanom v dizertačnej práci) získame klasickú štruktúru PID regulátora

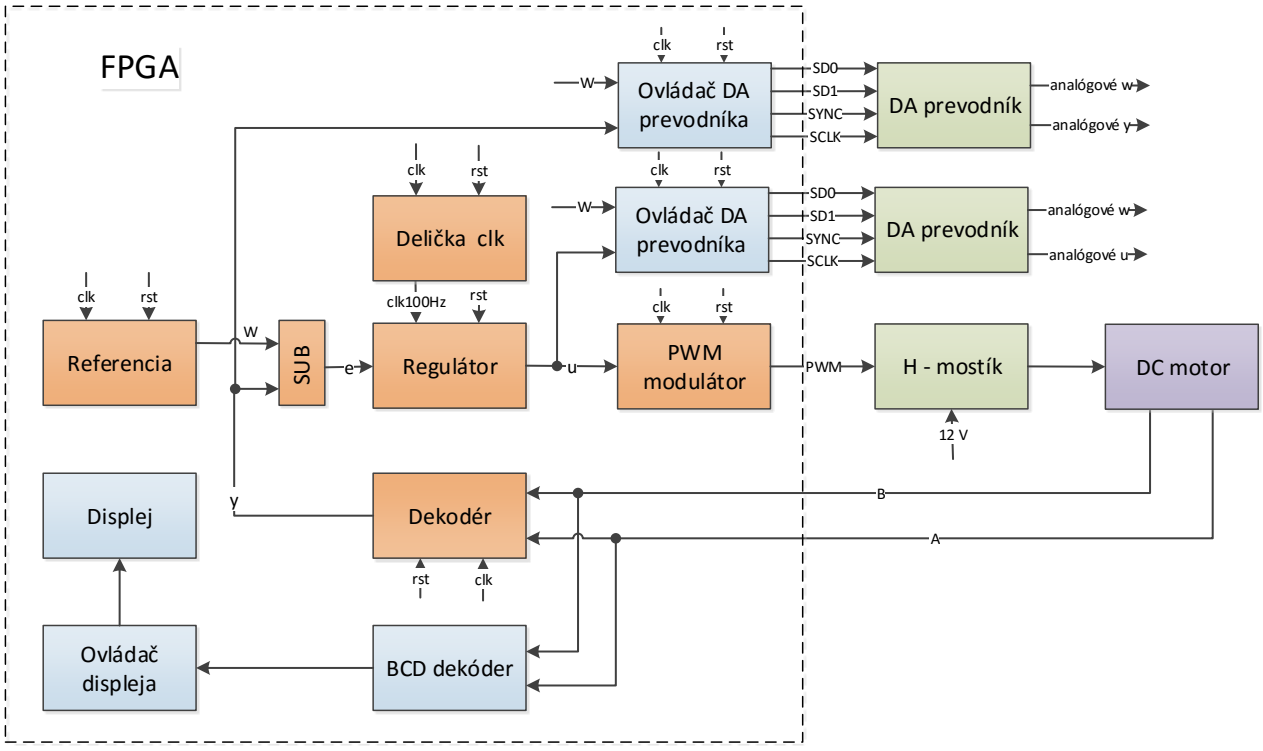
$$G_{c\_PID}(z) = \frac{0.0206 - 0.018z^{-1}}{1 - 0.7358z^{-1} - 0.2642z^{-2}} = \frac{U(z)}{E(z)} \qquad (2.21)$$

Výsledkom simulácie a hardvérovej realizácie sa venuje kapitola 2.3.

2.2 Hardvérová realizácia regulačného obvodu s DC motorom

V prechádzajúcich kapitolách sme opísali možné spôsoby implementácie algoritmov riadenia na PPGA štruktúrach. Pre úspešné splnenie úlohy riadenia na obvodoch PFGA však potrebujeme aj ďalšiu funkcionalitu. Riadený systém, v tomto prípade jednosmerný motor, je vybavený enkodérom s Hallovými sondami. Informácie o rýchlosti sa z enkodéra prenášajú do FPGA obvodu cez rozhranie PMOD. Tieto informácie nie sú priamo vstupom do regulátora a je potrebné ich najskôr dekódovať. Výstup regulátora taktiež nie je pripojený priamo na jednosmerný motor. 12-bitový signál je najskôr prevedený na jednobitový signál PWM, ten je následne poslaný na rozhranie PMOD. Za rozhraním PMOD sa nachádza H-mostík ktorý zosilňuje a dodáva dostatočný prúd do motora. V tejto kapitole sú opísané jednotlivé obvody ktoré boli na realizáciu využité.

Hardvérová realizácia uzatvoreného regulačného obvodu je na úrovni blokov zobrazená na Obr. 2.6. Dôležité bloky pre úlohu riadenia implementované na FPGA sú označené oranžovou farbou. Diagnostické bloky implementované na FPGA sú označené modrou farbou. Periférne obvody pripojené na rozhranie PMOD sú označené zelenou farbou a nakoniec riadený systém je fialový.



Obr. 2.6 – Bloková schéma regulačného obvodu

2.2.1 Blok referencie

Tento blok má za úlohu generovať žiadanú hodnotu w. Výstupom je 12-bitový znamiekový signál. Žiadna hodnota sa mení v pravidelných periódach z 300 ot/min na 400 ot/min. Výstupný signál nemá vyhradené žiadne bity pre čísla za desatinnou čiarkou, môže byť definovaný ako SFIXED(12,0). Po

prevode do binárnej sústavy nadobúda výstup hodnotu $300_{(10)} = 0001\ 0010\ 1100_{(2)}$ resp. $400_{(10)} = 0001\ 1001\ 0000_{(2)}$

2.2.2 Blok dekodéra

Dekodér dekóduje informáciu rýchlosti zo signálov A a B a prevádza ju na 12-bitový znamienkový signál. Informáciu o rýchlosti vie algoritmus vyhodnotiť ako rozdiel časov dvoch po sebe idúcich nábežných hrán signálov A resp. B. Rýchlosť tak získame po každej tretine otáčky motora (Enkodér zaznamená tri nábežné hrany signálu A alebo B za otáčku). Rýchlosť motora vyjadrenú v otáčkach za minútu [ot/min], vypočítame ako

$$y = \frac{60 \times 10^6}{t_A p_p p_o} = \frac{60 \times 10^6}{t_A \times 19 \times 3} = \frac{1052632}{t_A} \left[\frac{ot}{min}\right] \qquad (2.22)$$

kde t_A je čas medzi dvoma nábežnými hranami signálu A vyjadrený v µs, p_p je prevodový pomer motora (v našom prípade 19) a p_o je počet impulzov na jednu otáčku. Zo vzťahu (2.22) je zrejmé že na výpočet rýchlosti je potrebné delenie. Problémom delenia sme sa zaoberali v dizertačnej práci v kapitole 2.. V tomto prípade sme využili predpripravený funkčný blok delenia *Divider Generator* vo vývojovom prostredí Vivado s predvoleným algoritmov Radix2.

2.2.3 Blok regulátora

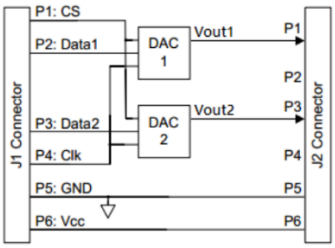
Za blok regulátora môžeme pri konkrétnych realizáciách dosadiť: PID regulátor, regulátor navrhnutý metódou rozmiestnenia pólov, robustný regulátor navrhnutý metódou reflexných vektorov a IMC regulátor. Všetky tieto regulátory sú navrhnuté tak, aby ich periférie boli kompatibilné. Vstupom do všetkých regulátorov je 13-bitový znamienkový signál regulačnej odchýlky e, vzorkovacia frekvencia $clk100Hz$ a signál pre reset rst.

2.2.4 PWM modulátor

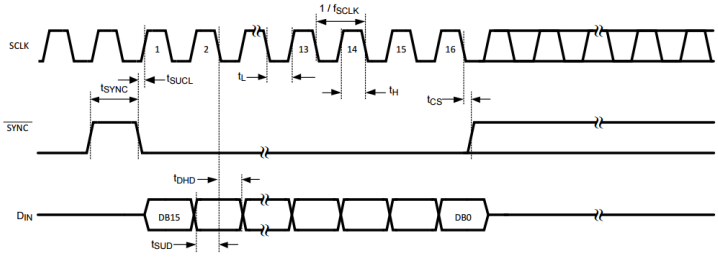
Blok PWM modulácie transformuje 12bitový signál akčného zásahu na 1-bitový signál PWM. Blok je pripojený na základný CLK signál obvodu s frekvenciou 100MHz. Maximálna hodnota, ktorú blok moduluje je $0110\ 0000\ 0000_{(2)} = 1536_{(10)}$. Nosná frekvencia sa skladá z 1536 impulzov 100MHz hodinového signálu, čo je po prepočítaní 65 104,2Hz . Strieda modulácie sa pohybuje podľa hodnoty akčného zásahu od 0 impulzov do spomínaných 1536 impulzov 100Mhz signálu.

2.2.5 Ovládač DA prevodníka

Modul obsluhy pre D/A prevodník má zabezpečiť komunikáciu s prevodníkom PmodDA2 tak, aby zariadenie pracovalo korektne. Vstupmi modulu sú dva 12-bitové vektory. Ďalšími vstupmi sú hodinový takt *clk* a signál *reset*. Výstupmi systému sú dva dátové signály a dva riadiace signály *SYNC* a *SCLK*, smerujúce do prevodníka. Prevodník PmodDA1 obsahuje dva čipy DAC121S101 (Texas Instruments, 2005). Štruktúra prevodníka je zobrazená na Obr. 2.7, proces prevodu je zobrazený na Obr. 2.8. Na ovládanie prevodníka je naprogramovaný stavový automat.



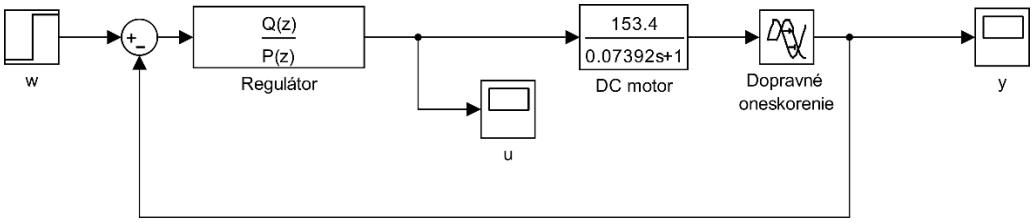
Obr. 2.7 - Bloková schéma DA prevodníka



Obr. 2.8 - Proces prevodu previdníka DAC121S101

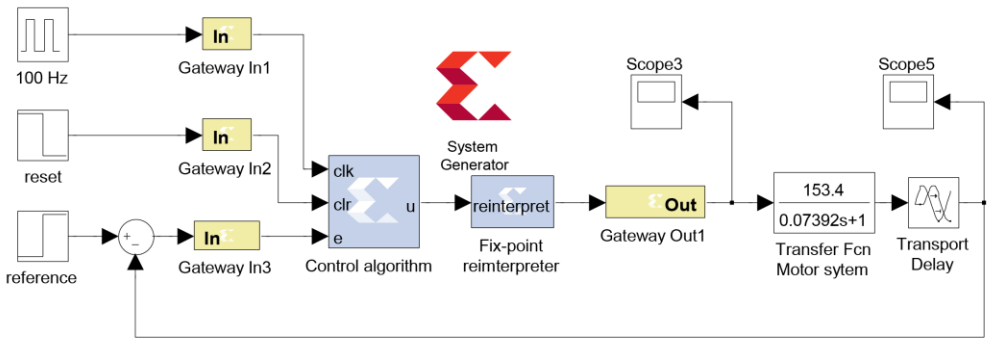
2.3 Verifikácia regulačných obvodov s DC motorom

V tejto kapitole porovnáme a vyhodnotíme získané výsledky hardvérových realizácií regulačných obvodov pre jednosmerný motor uvedené v prechádzajúcich kapitolách. Na verifikáciu regulátorov sme v prvom kroku overili simulačne pomocou SIMULUNKu vo výpočtovom prostredí MATLAB. S výnimkou bloku regulátora je simulačná schéma URO (Obr. 2.9) pre všetky regulátory rovnaká.



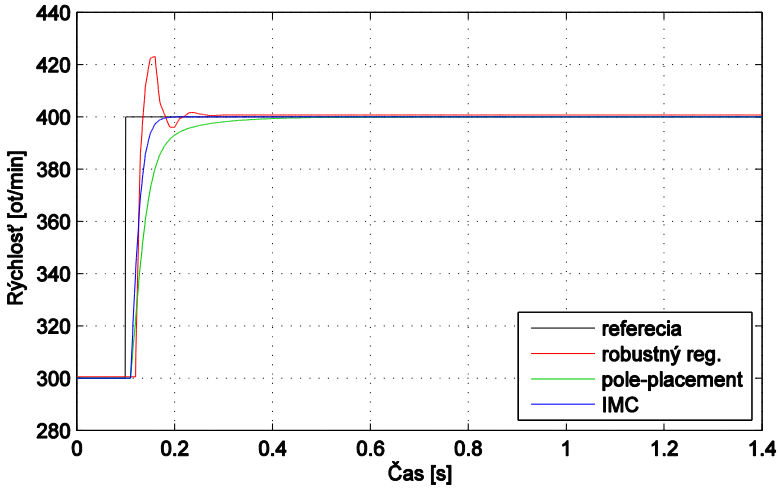
Obr. 2.9 – Simulačná schéma URO s jednosmerným motorom

Blok regulátora sme nahradili blokmi Xilinx, ktoré sú dostupné po nainštalovaní System Generator Toolboxu pre MATLAB. Systém generátor zaručuje, že sa obvod medzi blokmi Gateway In a Gateway Out správa tak, ako by bol naimplementovaný na FPGA. Medzi týmito blokmi sme prešli z reprezentácie reálnych čísel s pohyblivou čiarkou (Floating-point) na reprezentáciu s pevnou rádovou čiarkou (Fixed-point). Samotný blok regulátora je realizovaný v jazyku VHDL a je vložený do simulačnej schémy pomocou bloku Black Box. Tento blok zmapuje vstupy a výstupy VHDL súboru na prepojenie z ostatnými blokmi simulačnej schémy. Simulácia bloku prebieha v externom simulátore pre VHDL, ktorý využíva aj vývojové prostredie Xilinx Vivado alebo ISE. Totožný VHDL súbor regulátora je použitý aj v hardvérovej realizácií. Tento nástroj nám pri návrhu regulačných obvodov pomohol zadefinovať minimálne rozsahy vnútorných signálov regulátora tak aby sa presnosť regulátora približovala k floating-point reprezentácií. Po odladení boli výsledky simulácií boli pre základnú schému a schému zloženú s blokov Xilinx prakticky rovnaké. Simulačnú schému zloženú z Xilinx blokov môžeme vidieť na Obr. 2.10.



Obr. 2.10 – Simulačná schéma zložená z Xilinx blokov

Odozvy na skokovú zmenu žiadanej hodnoty rýchlosti motora sú pre jednotlivé simulácie URO zobrazené na Obr. 2.11.



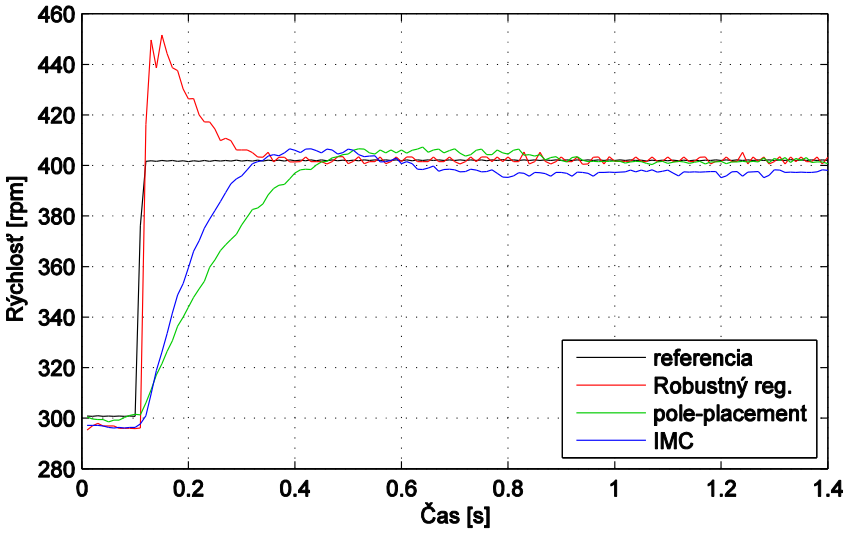
Obr. 2.11 – Odozva URO na skokovú zmenu žiadanej hodnoty

Na vyhodnotenie kvality URO v ustálených stavoch sme zvolili kritérium maximálneho preregulovania. Na vyhodnotenie kvality URO v prechodových stavoch boli použité dve kritériá a to čas regulácie a integrálne kritérium ITSE.

Tab. 2.1 – Kritéria kvality URO

Typ regulátora	Preregulovanie [%]	Doba regulácie ±1% [s]	Kritérium ITSE
Robustný	23,1%	0,153	371
Pole-placement	0%	0,257	498
IMC	0%	0,073	211

Po úspešnej verifikácií na simulačnej úrovni sme pristúpili k samotnej hardvérovej realizácií tak, ako je popísaná v kapitole 2.2. Skok žiadanej z 300 ot/min na 400 ot/min sa uskutočnil v čase 0.1 s. Odozva na skokovú zmenu je graficky vyjadrená na Obr. 2.12.



Obr. 2.12 – Odozva hardvérovej realizácie URO na skokovú zmenu žiadanej hodnoty

Podobne ako pri simuláciách sme kvalitu realizácie URO vyhodnotili v Tab. 2.2

Tab. 2.2 – Kritéria kvality hardvérovej realizácie URO

Typ regulátora	Preregulovanie [%]	Doba regulácie ±1% [s]	Kritérium ITSE
Robustný	51,6%	0,340	1097
Pole-placement	6,4%	0,790	5851
IMC	9%	0,560	3182

3 Realizácia prediktívneho riadenia na čipe

V tejto kapitole je navrhovaný prediktívny algoritmus riadenia využívajúci matematický model procesu na základe poznatkov uvedených v dizetačenj práci vkapitole 5. Riadený proces predstavuje laboratórny model – systém DC motorov opísaný v nasledujúcej kapitole 3.1. Na základe modelu systému je naformulovaný problém MPC riadenia a vyjadrená jeho explicitná forma. Explicitná forma riadenia je následne overená praktickým experimentom na obvode SoC, kde samotný výpočet akčného zásahu je realizovaný na procesore a ostatné časti ako sú pozorovateľ, prevodníky a pod. sú implementované na programovateľnej logike (FPGA časť obvodu SoC). Po overení algoritmu je ďalej uvedená konverzia algoritmu explicitného MPC z jazyka C na hardvérovú realizáciu opísanú v jazyku VHDL. Po tejto konverzií je hardvérová realizácia explicitného MPC verifikovaná na obvode FPGA Artix-7.

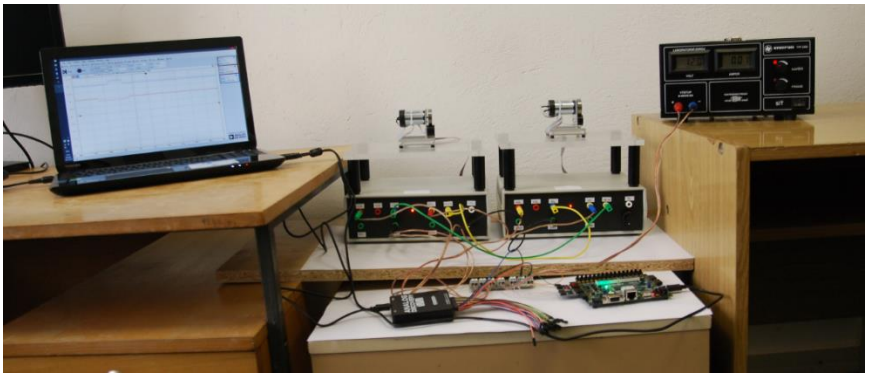
3.1 Opis riadeného systému – Laboratórny model s DC-motormi

Pre účely riadenia MPC metódami riadenia bol vybraný laboratórny model - systém DC-motorov zobrazený na Obr. 3.1. Tento systém sa skladá z dvoch modulov. Každý modul ma tri vstupy *U,U+*, *U-* a dva výstupy *X* a *1/X* (zoslabený výstup). Tieto moduly môžu byť prepojené rôznymi spôsobmi, napr. prepojením krížových väzieb a pod. Okrem prepojení je možné na moduloch konfigurovať aj veľkosť záťaže. Na účely verifikácie riadenia MPC na obvodoch FPGA bola vybraná nasledovná konfigurácia:

- na krížové väzby bol využité výstupy *1/x*,
- krížové väzby medzi motormi sú pripojené na záporný vstup,
- záporná väzba je vzájomná medzi oboma motormi,
- na oboch motoroch bola zvolená minimálna záťaž.

Schematické prepojenie modulov laboratórneho modelu je zobrazené na Obr. 3.1. Touto konfiguráciou vznikol systém s dvoma vstupmi u_1, u_2 a dvoma výstupmi y_1, y_2. Vo všeobecnosti takýto systém označujeme ako MIMO (multiple-input and multiple output).

Riadený systém s uvedenou konfiguráciou bol identifikovaný v práci (Noge, 2015). Na systéme boli namerané prevodové charakteristiky jednotlivo pre priame a krížové väzby. Prevodové charakteristiky boli porovnané medzi sebou, taktiež bola vykonaná aj voľba pracovných bodov. Na základe nameraných prevodových charakteristík boli potom zmerané prechodové charakteristiky v definovaných pracovných bodoch $u_1 = 1{,}24\ V, u_2 = 2{,}07\ V$.



Obr. 3.1 – Systém DC motorov

Ako výstup procesu identifikácie boli zvolené modely v tvare prechodových charakteristík. Cieľom bolo získať čo najjednoduchšie modely pri zachovaní čo najväčšej zhody s reálnym systémom. Výsledkom identifikácie v pracovných bodoch bolo viacero modelov z ktorých pre potreby tejto práce bol vybraný model s prenosovými funkciami prvého rádu.

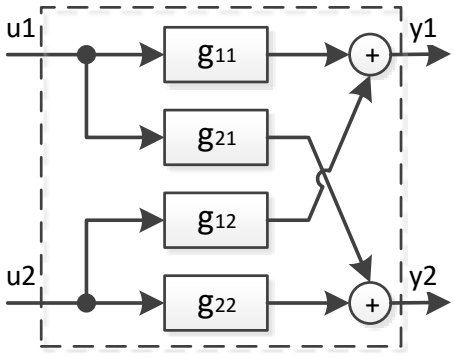
$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} = \begin{bmatrix} \frac{1{,}4257}{1{,}029s+1} & \frac{-0{,}4399}{1{,}404s+1} \\ \frac{-0{,}7322}{1{,}161s+1} & \frac{1{,}191}{0{,}5369s+1} \end{bmatrix} \qquad (3.1)$$

V Tab. 3.1 môžeme vidieť, že použitie zložitejšieho modelu by neprinieslo výraznejšiu zhodu z nameranými dátami.

Tab. 3.1 - Porovnanie kvality identifikovaných modelov

typ prenosovej funkcie	**zhoda v %**			
	g11	g12	g21	g22
prvého rádu	93,10	94,37	92,15	96,88
prvého rádu s nulou	93,28	95,87	93,29	96,95
druhého rádu	93,37	97,05	94,24	96,96
druhého rádu s nulou	93,42	97,35	94,36	96,96
druhého rádu s komplexnými koreňmi	93,37	97,1	94,33	96,96
druhého rádu s komplexnými koreňmi a nulou	93,42	97,35	94,37	96,96

Model systému je zložený zo štyroch prenosových funkcií. Prenosové funkcie $g_{11}(s)$ a $g_{22}(s)$ opisujú správanie sa modulov bez pripojenia krížových väzieb. Prenosové funkcie $g_{12}(s)$ a $g_{21}(s)$ modelujú krížové väzby. Ako môžeme vidieť na modeli (3.1), krížové väzby majú záporné zosilnenie, pretože sú pripojené na záporné vstupy modulov. Jeden kanál tak „brzdí“ kanál druhý a naopak. Vnútorné zapojenie modelu je zobrazené na Obr. 3.2.



Obr. 3.2 – Vnútorné zapojenie modelu systému

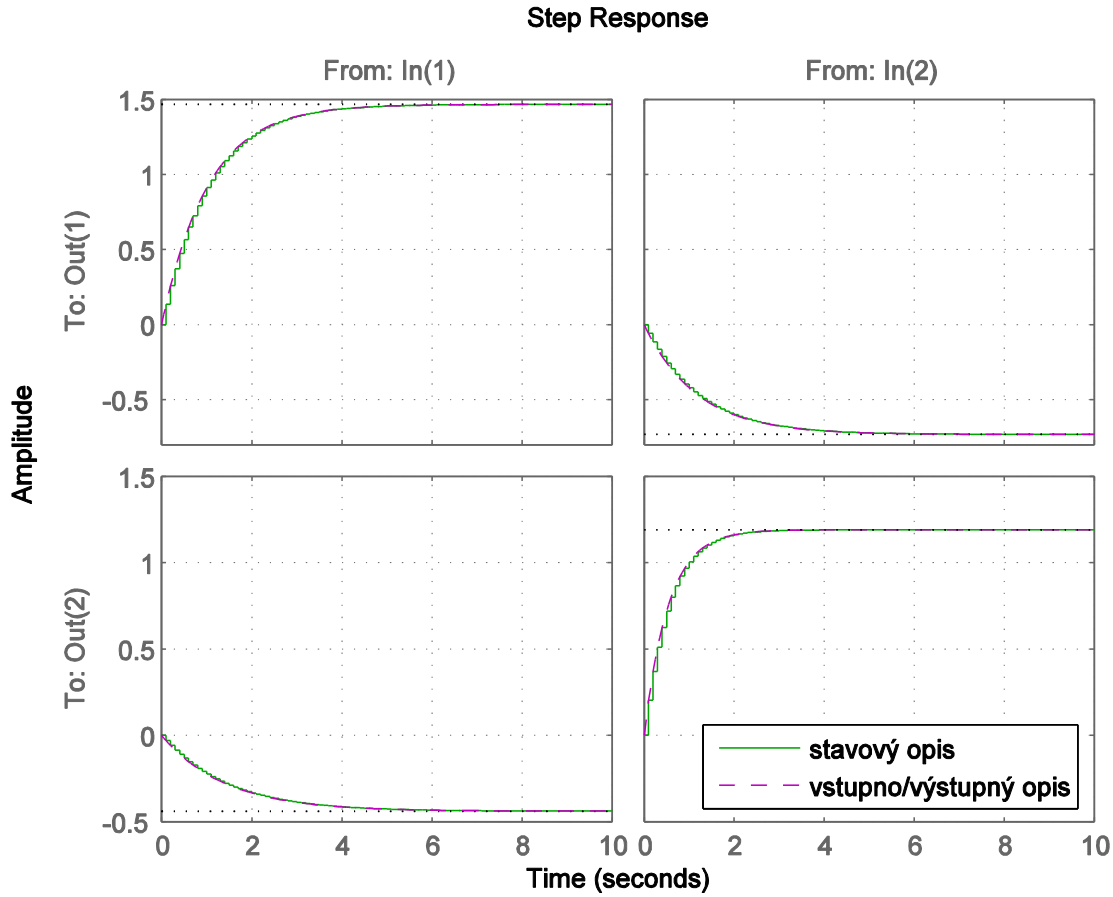
Pri návrhu prediktívneho regulátora budeme uvažovať opis pomocou stavových dynamických modelov

$$x_{t+1} = Ax_t + Bu_t$$
$$y_t = Cx_t + Du_t \tag{3.2}$$

Pred samotným riešením úloh MPC riadenia je potrebné transformovať opis riadeného systému z formy spojitých prenosových funkcií do tvaru diskrétneho stavového modelu (3.2). Prevod je možné realizovať viacerými metódami od analytických až po vygenerovanie funkciou *tf2ss a c2d* v programe Matlab. Pre laboratórny model – Systém s DC motormi opísaného pomocou (3.2) pri vzorkovacej frekvencií 10Hz (0,1s) platí že

$$A = \begin{bmatrix} 0.9074 & 0 & 0 & 0 \\ 0 & 0.9312 & 0 & 0 \\ 0 & 0 & 0.9175 & 0 \\ 0 & 0 & 0 & 0.8301 \end{bmatrix} \quad B = \begin{bmatrix} 0.1359 & 0 \\ -0.0302 & 0 \\ 0 & -0.0604 \\ 0 & 0.2024 \end{bmatrix} \tag{3.3}$$
$$C = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Správnosť transformácie je možné overiť porovnaním prechodovej charakteristiky (vstupno/výstupný opis) modelu (3.1) s diskrétnym stavovým opisom (3.3), ktorý je zobrazený na Obr. 3.3.



Obr. 3.3 - Porovnanie vstupno/výstupného a stavového opisu systému

3.2 Návrh Explicitnej formy MPC pre laboratórny model s DC-motormi

Pre návrh Explicitnej formy MPC (E-MPC) bol využitý MPT toolbox pre Matlab. Pri návrhu sme vychádzali z diskrétneho opisu systému (3.2),(3.3). Predikčný horizont bol nastavený na N=3, váhové matice zvolené nasledovne:

$$Q = \begin{bmatrix} 10000 & 0 \\ 0 & 10000 \end{bmatrix}, \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.4}$$

MPT kód pre nastavenie MPC regulátora je potom nasledovný:

```
% nastavenie regulácie na základe referencie
model.y.with('reference');
model.y.reference = 'free';

% nastavenie ohraničení akčného zásahu
model.u.min = [0,0];
model.u.max = [3.3, 3.3];

% nastavenie váhových koeficientov
model.y.penalty = OneNormFunction([1 0;0 1]);


% nastavenie horizontu predikcie
N = 3;

% vytovenie MPC
mpc = MPCController(model, N)
% vytvorenie explicitnej formy MPC
empc = mpc.toExplicit()
```

Takto vytvorený explicitný MPC regulátor má 395 regiónov. Po pridaní váhovej matice R pre akčný zásah sa zvýšil počet regiónov na 536. Rozhodli sme sa nepridať do váhovú maticu R do optimalizačného kritéria, obmedzenie rozsahu 0 – 3,3 V sme zachovali.

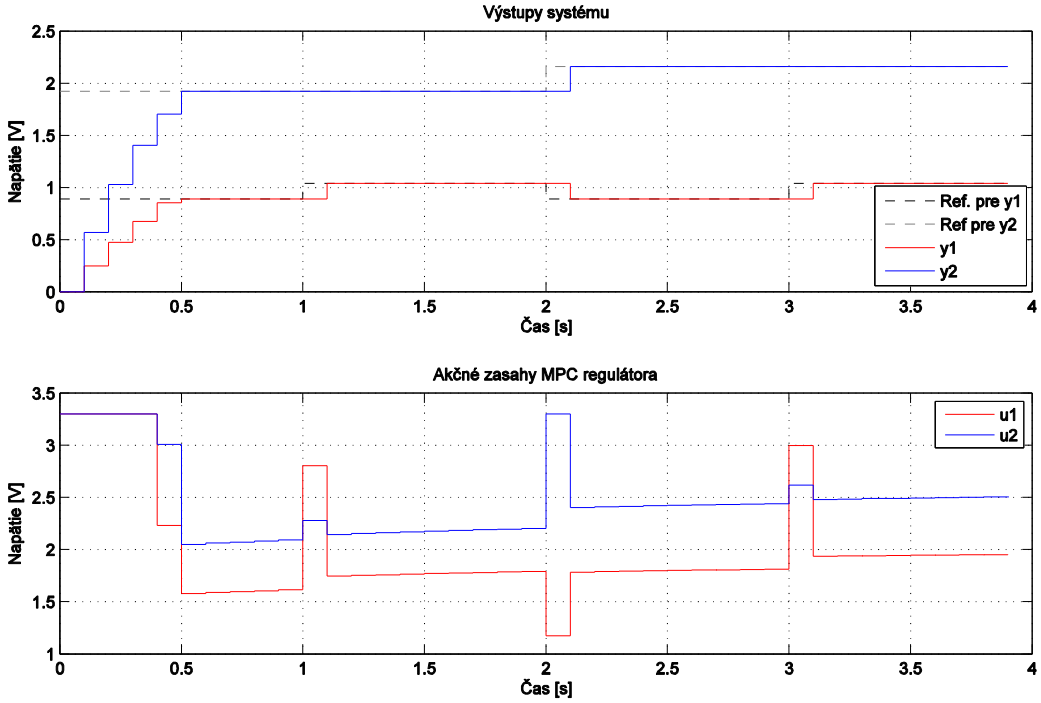
Vygenerovaný explicitný regulátor sme v prostredí MATLAB overili simuláciou, ktorú sme vytvorili príkazmi

```
% dlzka simulacie v krokoch
Nsim = 40;

% nastavenie referencie
yref = [repmat([  0.89; 1.92],1,Nsim/4), repmat([ 1.04; 1.92],1,Nsim/4), repmat([
0.89; 2.16],1,Nsim/4), repmat([ 1.04; 2.16],1,Nsim/4)];

% simulacia
data = loop.simulate(x0, Nsim,'u.previous', u0, 'y.reference', yref);
```

Výsledky simulácie sú zobrazené na Obr. 3.4. Po počiatočnom vybudení do pracovných bodov sa systém pri skokoch v okolí pracovných bodov ustáli na žiadanej hodnote za jednu periódu vzorkovania.



Obr. 3.4 – Časové priebehy regulovanej veličiny a riadiacich zásahov explicitného MPC v prostredí MATLAB

Explicitné MPC je potom možné exportovať do jazyka C príkazom:

```
empc.exportToC('controller', 'directory');
```

Príkaz v skutočnosti vyexportuje 3 súbory: *controller.c*, *controlle_mex.c a controller_sfunc.c*. Posledný vymenovaný slúži ako adaptér súboru *controller.c* pre simulácie v prostredí SIMULINK využitím bloku *S_function*. V reálnej aplikácií nebude MPC regulátor pripojený priamo na systém, ktorý produkuje stavy systému $x_1, \dots, x_4$ potrebné na výpočet riadiacich zásahov u_1, u_2. Výstupom reálneho systému sú len signály y_1, y_2. Na získanie stavov $x_1, \dots, x_4$ je nutné skonštruovať pozorovateľa stavov.

Štruktúra pozorovateľa je zobrazená na Obr. 3.5. Od modelu systému sa líši pridaním matice L, ktorá pôsobí na dynamiku chyby odhadu podľa vzťahu

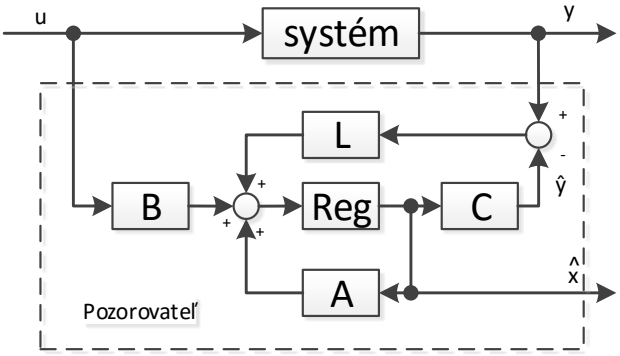
$$e(k+1) = x(k+1) - \hat{x}(k+1) = (A - LC)e(k) \tag{3.5}$$

Premennou $\hat{x}$ označujeme odhad skutočnej hodnoty x. Maticu L môžeme nájsť minimalizovaním kvadratického kritéria príkazom

```
L = dlqr(A',C',Q,R)
```

Kde A' je transponovaná matica A modelu systému, matica C' je transponovaná matica C modelu sytému, Q a R sú váhové matice príslušných rozmerov. Nájdená matica má hodnoty:

$$L = \begin{bmatrix} 0{,}3182 & 0 & 0{,}341 & 0 \\ 0 & 0{,}405 & 0 & 0{,}2382 \end{bmatrix} \tag{3.6}$$



Obr. 3.5 - Štruktúra pozorovateľa stavov

Na overenie funkcie pozorovateľa stavov bola vytvorená schéma simulácie, kde porovnávame výstupy spojitého systému a pozorovateľa. Počiatočné podmienky pozorovateľa sme nastavili tak, aby sa odlišovali od počiatočných podmienok systému. Ak pozorovateľ pracuje právne, okrem odhadov stavov sa odhadovaný výstup systému prispôsobí výstupu zo spojitého systému.

3.3 Implementácia explicitného MPC regulátora na obvode SoC

Na základe úspešnej verifikácie explicitného MPC algoritmu simuláciou v prostredí MATLAB môžeme pristúpiť k implementácií na reálnom systéme. V tejto kapitole navrhneme na hardvérovej úrovni nové IP jadro - pozorovateľa stavov, ostatné bloky (IP jadrá) sme už využili pri predchádzajúcich implementáciách. Samotný algoritmus explicitného MPC sa však bude vykonávať na PS časti SoC obvodu, teda na procesore. Využijeme tak C zdrojový kód, ktorý vygeneroval MPT toolbox. PS a PL časť je prepojená zbernicou AXI4-lite.

Keďže oproti implementáciám v kapitole 4 hardvérová zložitosť úlohy signifikantne narástla, rozhodli sme sa zvoliť komplexnejší prístup k návrhu hardvérovej časti. Bol využitý dizajn na báze IP (Intellectual Property) jadier (Xilinx, 2016) . IP dizajn prináša niekoľko výhod. V prvom rade je to oveľa efektívnejšie znovu využite už navrhnutých jadier. Vytvorené IP jadrá sa uložia do databázy a je možné ich vyvolať v ľubovoľnom projekte. Spoločnosť Xilinx vo svojom vývojom prostredí ponúka mnoho predpripravených IP jadier. Okrem toho je možné IP jadrá získať za úhradu alebo na platforme Open Cores, čo je Open Source ekvivalent pre IP jadrá. Ďalšou výhodou je jednoduchší proces návrhu hardvéru s IP jadrami. Vo vývojom prostredí Vivado je možné na tejto úrovni navrhovať dizajn graficky, čo podstatne urýchľuje proces návrhu. Ak by dizajnér navrhoval hardvér len na úrovni IP dizajnu, nepotepoval by žiadne znalosti jazyka HDL. Ide o prístup na vyššej úrovni abstrakcie. Nevýhodou je že takýto dizajn sa môže skladať len z IP jadier. Všetky bloky ktoré sme chceli využiť v tomto novom návrhu bolo potrebné najskôr zapuzdriť do IP.

Celkový IP dizajn navrhnutý pre EMPC na obvode SoC je v prílohe A.3. Dominantným blokom v celovej schéme je jadro *ZYNQ7 Processing System* ktoré reprezentuje ARM procesor. PS časť obvodu SoC reprezentujú ešte ďalšie dve jadrá a to resetovací systém *Processor System Reset* a radič zbernice *AXI Interconnect*. Ostatné Jadrá sú implementované na PL časti čipu.

Dizajn obsahuje dva DA prevodníky (*DA2RefComp*) a jeden AD prevodník (*ADprevodnik*), ktoré sú pripojené na PMOD JA, JB, JC periférne porty.

Výstupy AD prevodníka *data1out* a *data2out* sú pripojené na jadrá *mult_signal*. Tieto jadrá prevádzajú 11bitový signál z prevodníka na 11bitovu UFIXED(1,10) reprezentáciu. Prevodníky pracujú s rozsahom 0 – 3.3 V. Ak príde na vstup AD prevodníka napätie 3.3V AD prevodník prevedie toto napätie na 11bitový signál s hodnotou $1111\_1111\_1111_{(2)}$. Ak je jadro *mult_signal* nakonfigurované ako vstupné,

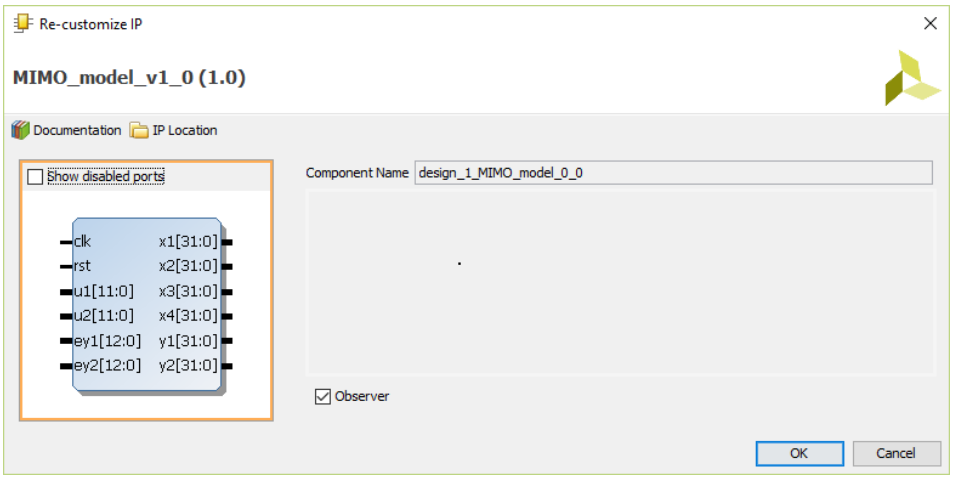
prevedie tento signál na fixed point reprezentáciu $11.0100110011_{(2)} = 3.2998046875_{(10)}$. Jadro *mult_signa*l je navrhnuté tak, aby korektne prevádzalo signál v celom rozsahu.

Tam kde je potrebné, sú v návrhu použité jadrá *bit_reducer*, ktoré menia rozsah signálu na menší počet bitov. Sú navrhnuté tak, aby prepúšťali menej významné bity.

Jadro s názvom *pevna_referencia* cyklicky generuje signál referencie s pevnou postupnosťou skokov okolo pracovných bodov

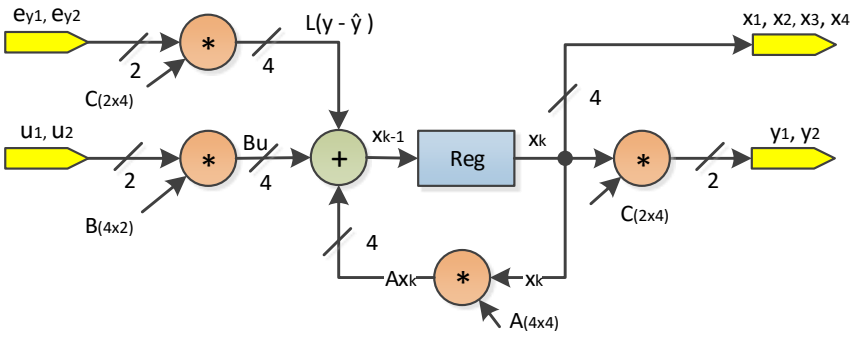
3.4 Hardvérová realizácia matematického modelu riadeného systému

Model systému (*MIMO_model*) je navrhnutý ako IP jadro, ktoré má 6 vstupov a 6 výstupov. Môže pracovať v dvoch módoch a to buď ako model alebo pozorovateľ (Obr. 3.6). Pri móde pozorovateľ sa využívajú vstupné signály *ey1* a *ey2*, ktoré zachytávajú rozdiel medzi odhadom výstupných signálov pozorovateľa $\hat{y}$ a výstupom reálneho systému y.



Obr. 3.6 – Konfigurácia IP jadra MIMO_model

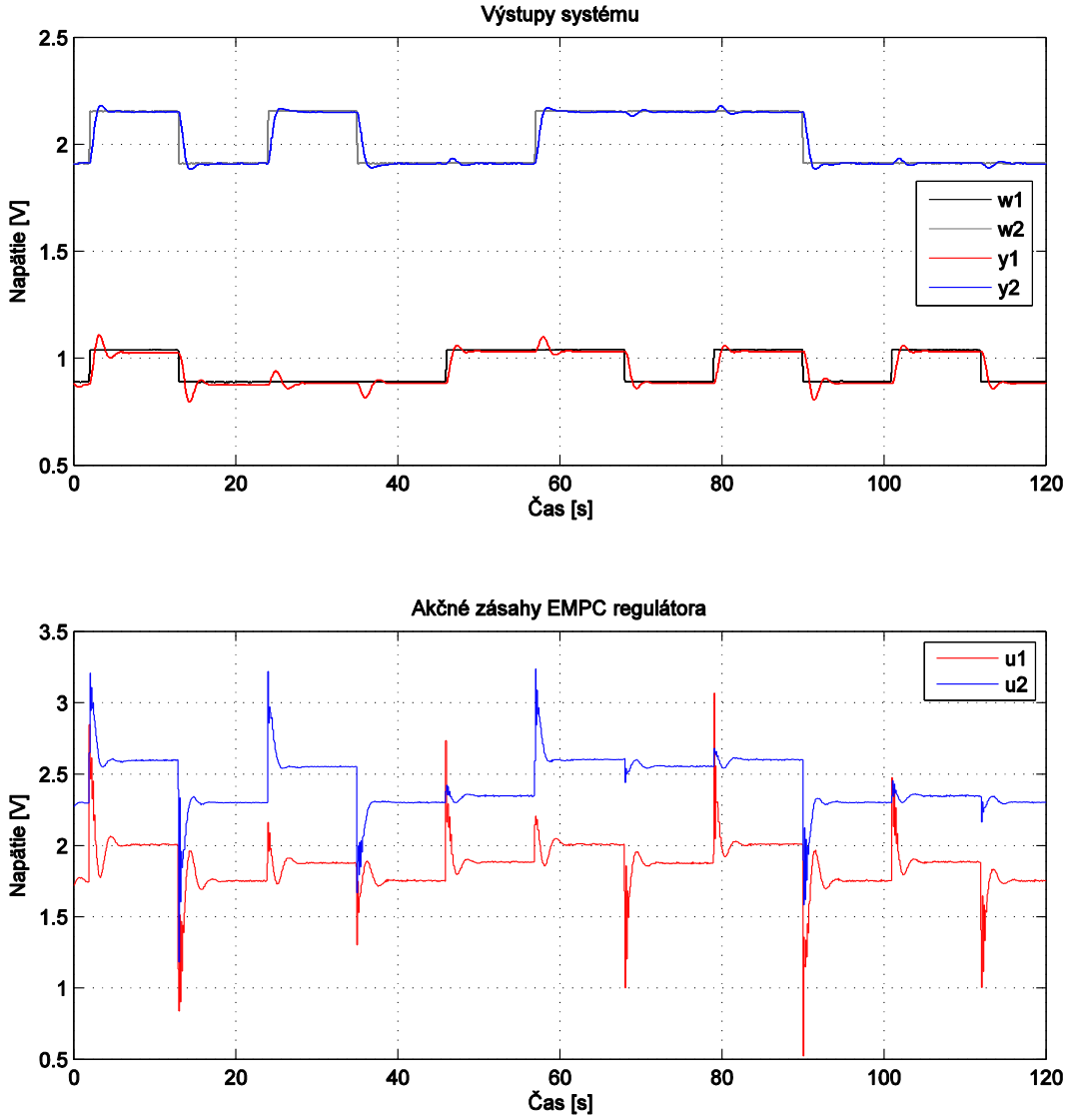
Výstupy systému x a y majú rozsah 32-bitov (10-bitov pre hodnotu za rádovou čiarkou), aj keď v tejto aplikácií nedosahujú hodnoty, ktoré by potrebovali takýto pomerne veľký rozsah. 32-btový výstup je použitý kvôli jednoduchšiemu spracovaniu v procesore, ktoré vysvetlíme v ďalšej kapitole. Na Obr. 3.7 je zobrazená vnútorná schéma zapojenia modelu systému. Operácie násobenia a sčítania sú maticové, realizované spôsobom prezentovaným v kapitole 2.



Obr. 3.7 – Vnútorná schéma modelu MIMO systému

3.5 Verifikácia

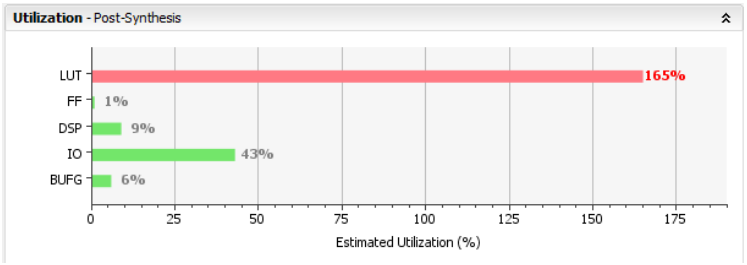
Na verifikáciu prediktívneho E-MPC algoritmu riadenia bol vytvorený experiment na reálnom systéme Obr. 3.1. Výsledky experimentu sú zobrazené na Obr. 3.8.



Obr. 3.8 – Časové priebehy regulovanej veličiny a riadiaceho zásahu E-MPC regulátora realizovaná na obvode SoC

3.6 Hardvérová realizácia explicitného MPC regulátora na obvode FPGA

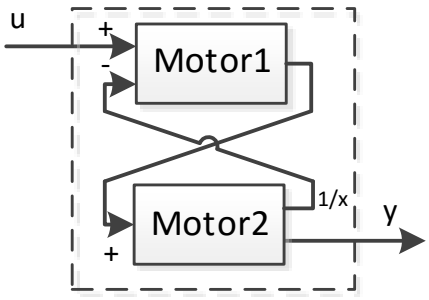
V tejto kapitole opíšeme hardvérovou realizáciu explicitného MPC regulátora na obvode FPGA. S kapacitných dôvodov sa nám nepodarilo zrealizovať EMPC regulátor tak, ako bol navrhnutý na obvode SoC. Teoreticky táto realizácia neobsahovala žiadne logické ani syntaktické chyby, kapacita obvodu FPGA Artix7- XC7A100T-CSG324 však nedostačovala navrhovanej implementácií. Výstupný graf implementácie je zobrazený na Obr. 3.9.



Obr. 3.9 – Odhadované využitie obvodu FPGA

Pre úspešnú realizáciu EMPC regulátora sme prekonfigurovali riadený systém tak, aby bol realizovaný EMPC regulátor jednoduchší, teda mal menej regiónov. Pôvodný explicitný regulátor mal 395 regiónov.

Riadiaci systém bol prekonfigurovaný podľa Obr. 3.10. Výstup prvého modulu je pripojený ako vstup do druhého modulu. Zoslabený výstup druhého modulu je pripojený ako záporný vstup prvého modulu. Výstup z druhého modulu reprezentuje celkový výstup systému. Vytvorili sme tak kaskádové zapojenie s jednou krížovou väzbou. Takto vytvorený systém má jeden vstup a jeden výstup.



Obr. 3.10 – Kaskádové zapojenie systému s DC motormi

Kaskádové zapojenie systému s DC motormi bolo identifikované v programe MATLAB pomocou toolboxu *Ident*. Výstupný model systému vo forme diskrétnej prenosovej funkcie je pri perióde vzorkovania 0,25 s

$$g_p(z^{-1}) = \frac{0{,}04872z^{-1} + 0{,}01481z^{-2} - 0{,}02239z^{-3}}{1 - 2{,}009z^{-1} + 1{,}398z^{-2} - 0{,}3309z^{-3}} z^{-1}. \tag{3.7}$$

Ekvivalentný stavový opis systému je definovaný

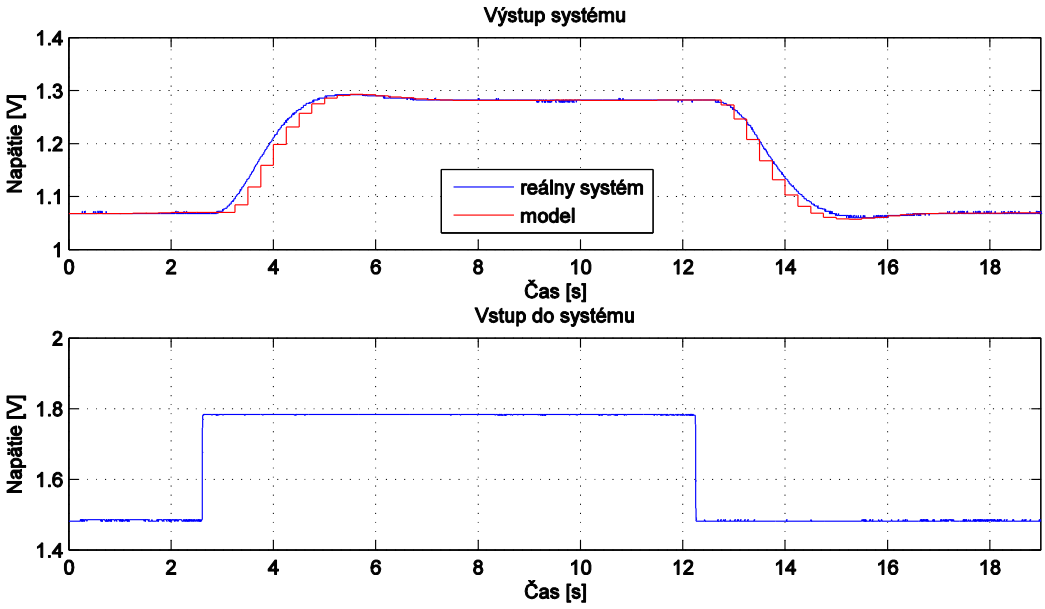
$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \end{aligned} \tag{3.8}$$

kde jednotlivé matice nadobúdajú hodnoty

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}3309 & -1{,}938 & 2{,}0089 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.9}$$

$$C = [-0{,}02239 \quad 0{,}0148 \quad 0{,}0487 \quad 0] \qquad D = [0]$$

Na Obr. 3.11 je zobrazené porovnanie stavového modelu systému a výstupu reálneho systému, ktorý sme získali experimentom. Na základe porovnania hodnotíme model ako vhodný.



Obr. 3.11 – Overenie metódy identifikácie modelu riadeného systému (6.15)

Návrh E-MPC regulátora sme realizovali nasledujúcim skriptom v prostredí MATLAB, ako aj pri predchádzajúcom postupe bol vyžitý MPT Toolbox (Herceg, a iní, 2013)

```
%Model pre MPT toolbox
model = LTISystem('A',A,'B',B,'C',C,'D',D,'Ts',Ts);
model.y.with('reference');
model.y.reference = 'free';

%Obmezenia
model.y.min = 0;
model.y.max = 3.3;
model.u.min = 0;
model.u.max = 3.3;

%Kriteria
model.y.penalty = QuadFunction(1000);
R = 1;
model.u.penalty = QuadFunction(R);

%Horizont predikcie
N = 5;

%Vytvorenie MPC
mpc = MPCController(model, N)
mpc.evaluate(x0,'y.reference',y);
expmpc = mpc.toExplicit()
```

Vygenerovaný zdrojový kód v jazyku C sme následne prekonvertovali do hardvérového opisu v jazyku VHDL. Kód E-MPC je možné rozložiť na dve logické časti. V prvej časti kódu sa nachádzajú rozmerné tabuľky, ktoré sú potrebné na výpočet riadiaceho zásahu, druhá časť kódu je samotný algoritmus riadenia.

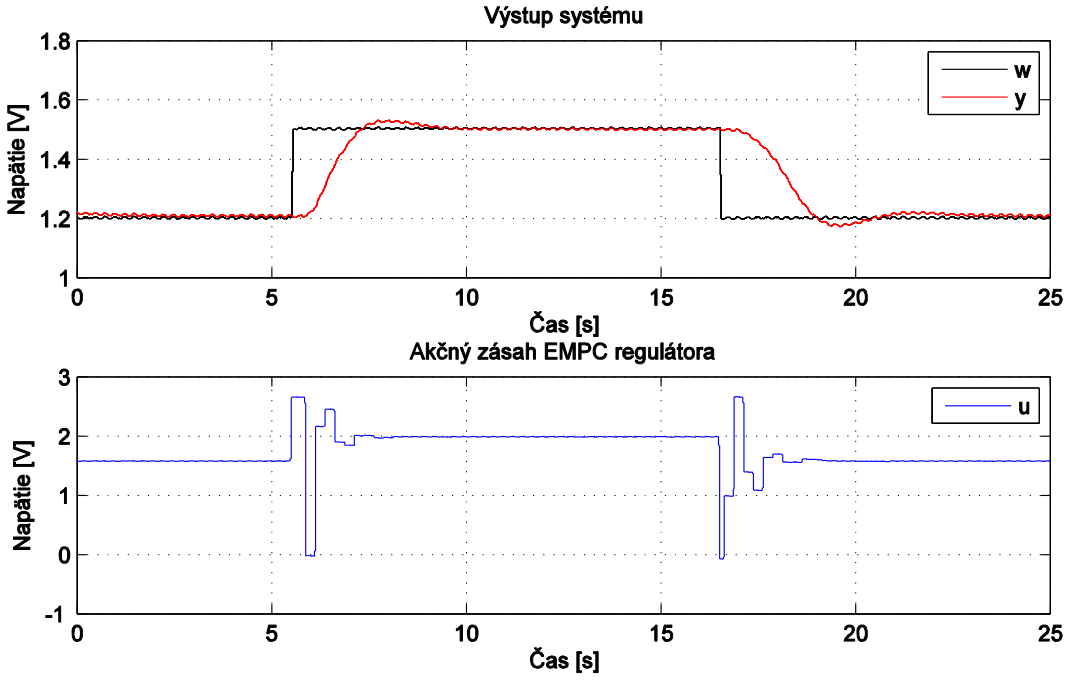
Tabuľky, presnejšie jednorozmerné vektory reálnych čísel, sme prekonvertovali na jednorozmerné vektory typu fixed point. Pri jednotlivých vektoroch sme dbali na rozsah tak, aby bolo možné implementovať všetky hodnoty vektora. Pre všetky vektory sme vyhradili 24 bitov za rádovou čiarkou, miesta pred rádovou čiarkou sa líšili v závislosti od vektora. Hardvérový opis v jazyku VHDL je uvedený v prílohe B.5.

Hardvérový opis je možné navrhnúť tak, aby sa akčný zásah vypočítal za jeden takt čipu. Môžeme však naraziť na problém s kapacitou čipu. Hardvérová náročnosť narastá najmä vo vnorených cykloch *for*. V opise hardvéru sa ani nedá hovoriť o cykle ako takom, pretože sa všetky vetvy vykonajú v jednom čase a každá vetva má svoju vlastnú hardvérovú štruktúru. Ak je implementovaný cyklus *for* v inom cykle *for*, hardvérová náročnosť mohutne narastá. V našom opise sme algoritmus preformulovali tak, aby sme sa vyhli vnoreným cyklom *for*. Nadradený cyklus sa vykonáva sekvenčne a tak je pri ňom využitá len jedna štruktúra. Výpočet akčného zásahu sa však už nevykoná za jeden takt čipu. Algoritmus sme preformulovali tak že:

- Rozmery riadeného systému nevplývajú priamo na čas výpočtu, ale na hardvérovú náročnosť,
- Dĺžka horizontu predikcie nevplýva priamo na čas výpočtu, ale na hardvérovú náročnosť,
- Počet regiónov vplýva na čas výpočtu.

Samozrejme ak navrhujeme systém s väčším počtom stavov alebo dlhším horizontom predikcie, prejaví sa to aj v počte regiónov, takže aj tieto parametre vplývajú nepriamo na čas výpočtu. Takto navrhnutý algoritmus ale aj tak vykonáva mnohé operácie v jednom takte, čo výrazne urýchľuje výpočet.

Výsledky hardvérovej realizácie sú zobrazené na Obr. 3.12. Na výslednom grafe môžeme vidieť že sa výstup systému ustálil na požadovanej hodnote *w*. Malé zvlnenie referencie a výstupnej hodnoty môžeme pripísať kvantizačnému šumu.



Obr. 3.12 – Časové priebehy regulovanej veličiny a riadiaceho zásahu získané E-MPC reguláciou na obvode FPGA.

Záver

Moderné metódy automatického riadenia umožňujú riešiť zložité úlohy riadenia procesov s rýchlou dynamikou. V predloženej dizertačnej práci je preto hlavným cieľom návrh takých moderných metód automatického riadenia, ktorých realizácia sa dá uskutočniť prostredníctvom FPGA obvodov a pomocou vnorených mikropočítačových systémov umožňujúcu ich realizáciu na čipe.

Predložená práca ponúka niekoľko vybraných metód automatického riadenia: PID štruktúry, IMC štruktúr, pole-placement metódy, robustné metódy a explicitné MPC, ktoré sú moderným prístupom na riadenie zložitejších a komplexnejších technologický procesov. Prediktívne metódy riadenia poskytujú optimálne, robustné riadenie s ohraničeniami procesných veličín, sú vhodné aj pre riadenie viacrozmerných systémov a potláčajú dopravné oneskorenia za predpokladu dopredu známej referencie. Tieto vlastnosti dosahujú za cenu pomerne vysokých nárokov na výpočtový výkon.

Obvody FPGA sú zložené s kombinačných a sekvenčných logických blokov, ktoré je možné programovo poprepájať. Takto je možné realizovať aj komplexné číslicové obvody. FPGA ponúkajú možnosť decentralizácie funkcionality a paralelného spracovania signálov, na rozdiel od mikroprocesorov, kde sa v jednom momente vykonáva len jedna inštrukcia.

Schopnosť paralelného spracovania a pomerne vysoká rýchlosť obvodov FPGA môže byť využitá na riešenie výpočtovo náročného problému prediktívneho riadenia. Predložená práca analyzuje možnosti implementácie prediktívnych algoritmov riadenia na obvodoch FPGA. Práca je rozdelená na niekoľko na seba nadväzujúcich častí.

V prvá časť práce je venovaná analýze samotných obvodov FPGA. Venuje sa histórií a postupnému vývoju programovateľných logických obvodov až po moderné obvody FPGA a SoC. Opisuje základnú štruktúru a technické parametre moderných FPGA čipov. Porovnáva a sumarizuje výhody a nevýhody obvodov FPGA a mikroprocesorov a definuje ich možnosti efektívneho využitia.

Druhá časť práce je venovaná opisu matematických algoritmov a numerických postupov a metód pre racionálne čísla a opisuje hardvérovú efektívnu realizáciu sčítania, súčinu a kroneckerovho súčinu matíc.

Tretia časť práce sa venuje hardvérovej realizácií algoritmov riadenia vo forme diskrétnych foriem PID. Rekurentná forma diskrétnych regulátorov je upravená tak, aby už priamo vo výpočte rešpektovala ohraničenia riadiaceho zásahu. To prináša efektívne riadenie procesov s ohraničením riadiaceho zásahu a zabraňuje neželanému windup efektu.

Štvrtá kapitola sa venuje implementácií a hardvérovej realizácií vybraných moderných metód riadenia. V práci sú predstavené realizácie regulátora založeného na metóde rozloženia pólov, IMC regulátora a robustného regulátora založeného na metóde reflexných polynómov.

Piata časť práce sa venuje možnostiam implementácie prediktívneho riadenia na PFGA. Opisuje podstatu MPC a naznačuje štruktúru algoritmov, ktoré môžu byť efektívne realizovateľné na obvodoch FPGA.

Samotná implementácia a hardvérová realizácia je realizovaná pre explicitnú formu MPC algoritmu. Všetky hardvérové realizácie sú overené experimentmi. Výsledky overenia sú vyjadrené tak numerickými výsledkami ako aj pomocou grafických prezentácií časových priebehov regulovaných veličín a riadiacich zásahov.

Výsledky prezentované v dizertačnej práci sú tak teoretického ako aj praktického charakteru. Teoretická časť práce sa zaoberá návrhom a modifikáciou existujúcich metód automatického riadenia pre ich možnú

realizáciu prostredníctvom FPGA obvodov a vnorených mikropočítačových systémov. Praktická časť dizertačnej práce je zameraná na overenie a implementáciu vybraných metód automatického riadenia na zvolených typoch FPGA obvodov: Artix-7 a Zynq 7000 SoC. Algoritmy boli testované tak pre SISO ako aj pre MIMO systémy, reprezentované matematickými modelmi ako aj reálnymi fyzikálnymi laboratórnymi modelmi.

Dosiahnuté výsledky potvrdzujú výskokú použiteľnosť metodiky riadenia prostredníctvom hardvérových foriem algoritmov riadenia. Prínos práce je signifikantný tak v teoretickej ako aj praktickej oblasti, čo potvrdzujú aj publikácie autora dizertačnej práce a kolektívu pracovníkov ÚAMT na významných svetových konferenciách spolu s ohlasmi a citáciami.

V tomto autoreferáte sú publikované vybrané časti z tretej, stvrtej a šiestej kapitoly dizertačnej práce.

Literatúra

Ackermann, J. a Bartlett, Andrew. 1993. *Robust control: Systems with Uncertain Physical Parameters.* London : Springer-Verlag, 1993.

Bemporad, Alberto, a iní. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica.* 2002, Zv. 31, 1.

Cigánek, Ján. 2010. *Príspevok k problémom robustného riadenia s využitím stabilných polytopov reflexných vektorov.* Bratislava : s.n., 2010.

Corriou, Jean-Pierre. 2004. *Process Control: Theory and Applications.* s.l. : Springer, 2004. ISBN 1-85233-776-1.

Diaz-Barrero, José Luis a Egozcue, Juan José. 2004. Characterization of polynomials using reflection coefficients. *Applied Mathematics E-Notes.* 4.1, 2004, s. 114-121.

Digilent. 2016. Nexys 4 DDR Artix-7 FPGA: Trainer Board Recommended for ECE Curriculum. [Online] 2016. [Dátum: 3. Marec 2016.] http://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/.

Edwards, Steve. 2011. Microprocessors or FPGAs?: Making the Right Choice. *RTC magazine.* [Online] Február 2011. [Dátum: 5. November 2012.] http://rtcmagazine.com/articles/view/102015.

Galajda, P. 2011. *FPGA obvody.* [Prednášky k predmetu FPGA obvody] FEI TUKE : s.n., 2011.

Haskell, Richard E. a Hanna, Darrin M. 2012. *Digital Design Using FPGA Boards.* Rochester Hills, MI : LBE Books, 2012. ISBN 978-0-9801337-8-3.

Herceg, M., a iní. 2013. Multi-Parametric Toolbox 3.0. *Proc. of the European Control Conference.* 2013, s. 502-510.

Ingole, D. a Kvasnica, M. 2015. FPGA Implementation of Explicit Model Predicitive Control fo Closed Loop Control of Depth of Anesthesia. *5th IFAC Conference on Nonlinear Model Predictive Control.* 2015.

Kay, Steven M. 1988. *Modern spectral estimation.* Englewood Cliffs, N.J. : Prentice Hall, 1988.

Kocúr, Michal. 2011. *Aplikácia vnorených systémov pre riadenie laboratórnych fyzikálnych modelov - Bakalárska práca.* Bratislava : STU BA, 2011.

Kocúr, Michal. 2013. *HW realizácia PID algoritmov na báze FPGA štruktúr.* Bratislava : s.n., 2013.

Kozák, Šefan. 1991. *Lineárne číslicové systémy I.* Bratislava : Slovenská technická univerzita, 1991. ISBN 80-227-0435-0.

L. H. Keel, S. P. Bhattachayya. 1999. A linear programming approach to controller design. *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on.* 1999, s. 1717–1724.

Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz and Robert W. Stewart. 2014. *The Zynq Book: Embedded Processing with the ARM CortexA9.* s.l. : Strathclyde Academic Media, 2014.

Martin, Šimka. *Vložené architektúry v kryptografických systémoch - písomná práca k dizertačnej skúške.* [online] Košice : s.n. http://www.martinsimka.com/files/sim_phd_04.pdf.

Noge, Filip. 2015. *Príspevok k realizácii metód prediktívneho riadenia pomocou FPGA štruktúr.* Bratislava : s.n., 2015.

Nurges, U. New stability conditions via reflection coefficients of polynomials. *IEEE Transactions on Automatic Control.* Vol. 50, 9, pp. 1354-1360.

Oppenheim, Alan V a Schafer, Ronald W. 1989. *Discrete-time signal processing.* Englewood Cliffs, N.J. : Prentice Hall, 1989.

Picinbono, Bernard and Messaoud, Benidir. 1986. Some properties of lattice autoregressive filters. *Acoustics, Speech and Signal Processing.* 1986, Vol. 2, 34.

Qin, Leon. 2012. Using NEON for Parallel Data Processing; Zynq-7000 Hardware Architecture. [Online] 2012. http://www.xilinx.com/Attachment/53775/Neon_Introduction_for_Avnet_training.pdf.

Rivera, Daniel E., Morari, Manfred and Skogestad, Sigurd. 1986. Internal model control: PID controller design. *Industrial & Engineering Chemistry Process Design and Development.* 1986, Vol. 25, 1, pp. 252-265.

Shayang Ye Industrial Co.,LTD. 2013. Digilent Inc. . [Online] 13. December 2013. [Dátum: 15. Marec 2015.] https://reference.digilentinc.com/_media/motor_gearbox:290-006_ig220019x00015r_ds.pdf.

Smith, Michael John Sebastian. 1997. *Application-Specific Integrated Circuits.* s.l. : Addison Wesley Professional, 1997. 9780321602756.

Texas Instruments. 2005. DAC121S101/-Q1 12-Bit Micro Power, RRO Digital-to-Analog Converter. *www.ti.com.* [Online] 2005. [Dátum: 15. Október 2015.] http://www.ti.com/lit/ds/symlink/dac121s101.pdf.

Viswanathan, V. 2005. *Embedded Control Using FPGA.* Indian Institute of Technology, Bombay : s.n., 2005.

Xilinx. 2014. 7 Series DSP48E1 User Guide. *www.xilinx.com.* [Online] 10. November 2014. [Dátum: 9. Januar 2015.] http://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf.

Xilinx. 2014. 7 Series FPGAs and Zynq-7000 All Programmable SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide. *www.xilinx.com.* [Online] 21. Október 2014. [Dátum: 2015. Január 10.] http://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf.

Xilinx. 2014. 7 Series FPGAs Overview. *www.xilinx.conm.* [Online] 17. December 2014. [Dátum: 20. December 2014.] http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

Xilinx. 2013. Aerospace and Defense. [Online] 2013. http://www.xilinx.com/applications/aerospace-and-defense.html.

Xilinx. 2014. CPLD. *http://www.xilinx.com/.* [Online] 2014. [Dátum: 15. 12 2014.] http://www.xilinx.com/cpld/.

Xilinx. 2013. Industrial Networking. *Xilinx All Programmable.* [Online] 2013. [Dátum: 2013. 4 10.] http://www.xilinx.com/applications/industrial/industrial-networking/index.htm.

Xilinx. 2016. Intellectual Property. *www.xilinx.com.* [Online] 2016. [Dátum: 10. Február 2016.]

Xilinx. 2011. PicoBlaze 8-bit Embedded Microcontroller. *www.xilinx.com.* [Online] 22. Jún 2011. [Dátum: 2014. Január 12.] http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf.

Zhengwei Fang, Joan E. Carletta, Robert J. Veillette. 2005. A Methodology for FPGA-Based Control Implementation. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.* 2005, Zv. 13, 6.

Publikačná činnosť autora

ADE Vedecké práce v ostatných zahraničných časopisoch

ADE01 CIGÁNEK, Ján - KOCÚR, Michal - KOZÁK, Štefan. FPGA as a tool for hardware realization of feedback control. In *Journal of Electrical Systems and Information Technology*. Vol. 2, Iss. 3 (2015), s. 328-337. ISSN 2314-7172.

ADM Vedecké práce v zahraničných časopisoch registrovaných v databázach Web of Science alebo SCOPUS

ADM01 OSUSKÝ, Jakub - NOGE, Filip - KOCÚR, Michal. Multivariable generalized predictive control design: Application for couple of DC motors. In *International Review of Automatic Control*. Vol. 8, No. 3 (2015), s. 197-202. ISSN 1974-6059. V databáze: SCOPUS: s2.0-84938406151.

AFC Publikované príspevky na zahraničných vedeckých konferenciách

AFC01 CIGÁNEK, Ján - KOCÚR, Michal - KOZÁK, Štefan. Hardware realization of robust controller designed using reflection vectors. In *Computation tools 2015 : 6th International conference on computational logics, algebras, programming, tools, and benchmarking. Nice, France, March 22 - 27, 2015*. [s.l.] : IARIA, 2015, S. 14-19. ISBN 978-1-61208-394-0.

AFC02 CIGÁNEK, Ján - KOCÚR, Michal - KOZÁK, Štefan. Realization of robust controller algorithm using FPGA. In *CAO' 2015 : 16th IFAC workshop on control applications of optimization. Garmisch-Partenkirchen, Germany. October 6-9, 2015*. München : Universität der Bundeswehr München, 2015, S. 156-161. ISBN 978-3-943207-12-5. V databáze: SCOPUS: 2-s2.0-84957950469.

AFC03 KOCÚR, Michal - KOZÁK, Štefan - DVORŠČÁK, Branislav. Design and implementation of FPGA - digital based PID controller. In *Proceedings of the 15th International Carpathian Control Conference [elektronický zdroj] : ICCC 2014; Velké Karlovice, Czech Republic, May 28-30, 2014*. [s.l.] : IEEE- Czechoslovakia Section of IEEE, 2014, CD-ROM, p. 233-236. ISBN 978-1-4799-3527-7.

AFD Publikované príspevky na domácich vedeckých konferenciách

AFD01 CIGÁNEK, Ján - KOCÚR, Michal - KOZÁK, Štefan. Advanced methods of controller design: Simulation and co-simulation on FPGA. In *2016 Cybernetics & informatics (K&I) [elektronický zdroj] : 28th International conference. Levoča, Slovak Republic, February 2-5, 2016*. 1. ed. Danvers : IEEE, 2016, CD-ROM, [6] s. ISBN 978-1-5090-1834-5. V databáze: IEEE.

AFD02 KOCÚR, Michal - KOZÁK, Štefan - DVORŠČÁK, Branislav. Hardware co-simulation of digital PID controller implemented in FPGA. In *Kybernetika a informatika 2014 [elektronický zdroj] : Medzinárodná konferencia SSKI. Oščadnica, Slovenská republika, 5.-8. 2. 2014*. 1.vyd. Bratislava : STU v Bratislave FEI, 2014, CD-ROM [4] p. ISBN 978-80-227-4122-4.

AFD03 KOCÚR, Michal - CIGÁNEK, Ján. Reflection vectors based robust controller design: Implementation using FPGA. In *ELITECH´15 [elektronický zdroj] : 17th Conference of doctoral students. Bratislava, Slovak Republic, May 25, 2015*. 1. vyd. Bratislava : Nakladateľstvo STU, 2015, CD-ROM, [7] s. ISBN 978-80-227-4358-7.

BEE Odborné práce v zahraničných zborníkoch (konferenčných aj nekonferenčných)

BEE01 CIGÁNEK, Ján - KOCÚR, Michal - KOZÁK, Štefan. Robust controller design realized on FPGA. In *ACCS/PEIT 2015 [elektronický zdroj] : 4th International conference on advanced control circuits and systems; 3rd International conference on new paradigms in electronics & information technology. Luxor, Egypt. 15 - 19 November 2015*. Cairo : Electronics Research Institute, 2015, CD-ROM, [7] s.

BEE02 KOZÁKOVÁ, Alena - KOCÚR, Michal. Observer-based digital reference tracking: FPGA realization. In *SVCS 2016 : 13th International symposium on stability, vibration, and control of systems. Budapest, Hungary. June 16-18, 2016*. Budapest : University of Technology and Economics, 2016, [12] s.

Prezentované príspevky na zahraničných vedeckých konferenciách

CIGÁNEK, Ján - KOCÚR, Michal - KOZÁK, Štefan. FPGA Realization of Robust Controller Design. In *SVCS 2016 : 13th International symposium on stability, vibration, and control of systems. Budapest, Hungary. June 16-18, 2016*. Budapest : University of Technology and Economics, 2016.

CIGÁNEK, Ján - KOCÚR, Michal - KOZÁK, Štefan. Hardware Realization of Advanced Controller Design Methods using FPGA. In *ICONS' 2016 : 14th IFAC International Conferece on Inteligent Control Sciences. Reims, France. June 1-3, 2016*.