Diplomová práce

České vysoké učení technické v Praze

Katedra řídicí techniky

Vypracoval: Lukáš Malý v květnu 2003

Kontakt: AVES@centrum.cz

Prohlášení

Prohlašuji, že jsem svou diplomovou (bakalářskou) práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ……. …….

podpis

Anotace v českém jazyce:

Tento dokument je výsledkem diplomové práce, ve které se budeme zabývat návrhem a realizací řídicího systému pro víceosý servomechanismus, kde je třeba řídit polohu a rychlost pohybu jednotlivých os nezávisle na sobě. K realizaci má být využito programovatelné hradlové pole FPGA typu Spartan II od firmy Xilinx. Vstupem z regulované soustavy jsou signály od IRC snímačů, výstupem pulzní šířková modulace pro oba směry otáčení. Návrh konfigurace bude proveden v jazyce VHDL. Data se do řídicího systému mají posílat po navržené paralelní sběrnici. Všechny vstupy a výstupy musí být galvanicky oddělené.

Anotation in English:

This document is a result of a diploma work occupying with a design and realisation of control system for two-axes servomechanism. We need to control speed and position of each axis independently. For realisation we will use programmable gate array FPGA Spartan II from Xilinx. Inputs of the system are signals from IRC sensors, output is pulse width modulation for both drive directions. The design will be done in VHDL language. Input and output data will be sent by designed parallel bus. All inputs and outputs should be galvanic isolated.

Obsah:

1 Úvod

Účelem práce je seznámit se s programovatelnými hradlovými poli FPGA využitím jejich možností k sestrojení řídicího systému pro víceosý obráběcí stroj. Proč v tomto případě nezůstat při starších osvědčených metodách konstrukce? To vysvětlí následující odstavec.

Vyvíjíme-li číslicový elektronický systém, je třeba zvolit vhodný způsob provedení. Mohli bychom takový elektrický obvod postavit z diskrétních součástek což by samozřejmě zabralo mnoho času, prostoru, výroba by byla nákladná, spolehlivost takto zkonstruovaného zařízení by byla nízká a spotřeba energie vysoká. Šlo by rovněž poskládat takový systém z vybraných větších celků, jako jsou např. logická hradla, klopné obvody a podobně. Bylo by to výhodnější než předchozí možnost, nicméně stále příliš neekonomické. Navíc není snadné v takovém zařízení měnit logické funkce. Znamenalo by to nejčastěji výměnu celých desek plošných spojů. Bylo by také možné navrhnout celý logický obvod do jednoho čipu (ASIC). Taková varianta by se vyplatila při sériové výrobě, neboť návrh a výroba masek pro čip je velmi nákladná. Je však ještě jedna, velmi perspektivní možnost, a sice použít k návrhu, vývoji i výrobě programovatelné logické pole FPGA. Jedná se o součástku velmi vysoké integrace vyrábějící se nejnovějšími technologiemi, jejíž možnosti využití v číslicové technice jsou nesčetné. FPGA můžeme nazvat univerzálním čipem s pamětí, do kterého (zjednodušeně řečeno) naprogramujeme všechna požadovaná vnitřní pospojování mezi jednotlivými logickými bloky. Většina těchto polí se dodává s přepisovatelnými typy konfiguračních pamětí, takže je možno je kdykoli přeprogramovat, některé dokonce za provozu. To velmi usnadňuje a urychluje vývoj číslicových obvodů, neboť modifikace zapojení se vlastně provede pouhým nahráním dat do čipu. Při použití dnes už běžného rozhraní ISP (In System Programming) nemusíme ani integrovaný obvod vyjímat z patice. Pokusme se tedy této relativně nové technologie využít ke stavbě zařízení pro řízení víceosého obráběcího stroje.

2 Charakteristika řešeného problému

Cílem této diplomové práce je navrhnout a implementovat řídicí systém pro víceosý NC stroj s využitím programovatelných obvodů FPGA firmy XILINX. K dispozici je soustava se dvěma stejnosměrnými motory s hotovými obvody pro připojení řídicích signálů a k práci byla využita vývojová deska osazená čipem Spartan II XC2S100 s návrhovým prostředím Webpack ISE 4.1, které je produktem Insight Memec [A,C] .

Vstupem řídicího systému budou požadované hodnoty rychlosti nebo polohy a výstupem pulzně šířkově modulovaný signál pro motory os. Po komunikační sběrnici musí být možné nastavit vnitřní konfiguraci zařízení, jako jsou regulační meze, maximání hodnoty napětí, perioda vzorkování, nastavení regulátorů a další.

Poloha a otáčky motoru budou snímány rotačním inkrementálním čidlem (IRC) spojeným s hřídelem motoru. Na jejich výstupech najdeme dva dvoustavové signály A a B vzájemně fázově posunuté, takže je možno určit krok i směr otáčení. Lze tak snímat pouze relativní polohu. Někdy se IRC doplňuje ještě o třetí signál určující referenční nulu. V této úloze je ale potřeba referenční nulu nastavovat pomocí dorazových snímačů, takže třetí signál nevyužijeme.

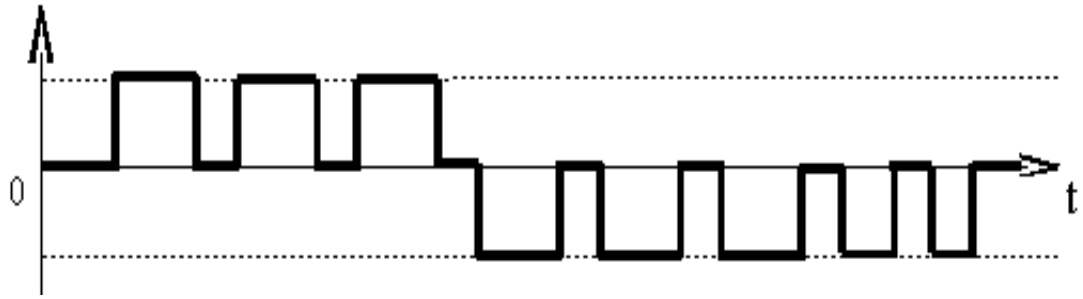
Řízení otáček a polohy má být provedeno PI regulátory implementovanými uvnitř zmíněného programovatelného obvodu. PI regulátor je soustava určená ke stabilizaci systému, kde vnitřními parametry jsou proporcionální složka a integrační složka. Mohli bychom použít i regulátor obohacený o složku derivační, kterou by šlo zlepšit kvalitu řízení, ale pro naši aplikaci vystačíme i bez ní.

Do použitého programovatelného pole se má podařit implementovat jak potřebné regulátory, tak příslušné moduly potřebné pro měření polohy a otáček, generování PWM a komunikaci s ovládacím prostředím, jímž může být mikropočítač připojený ke komunikační sběrnici.

Programovatelné pole je univerzální čip se schopností konfigurace vnitřního zapojení (většinou i přeprogramovatelnosti), kde programováním rozumíme vnitřní pospojování logických členů do číslicového obvodu. V současnosti jsou FPGA na takové úrovni, že pracují při hodinových frekvencích řádově stovky MHz a obsahují stovky tisíc ekvivatelntních hradel při nízké spotřebě energie, takže do nich můžeme implementovat i systémy relativně velké složitosti. K jejich návrhu potřebujeme jazyk a návrhové prostředí. Součástí práce bylo také naučit se s jazykem a s prostředím včetně simulátoru pracovat.

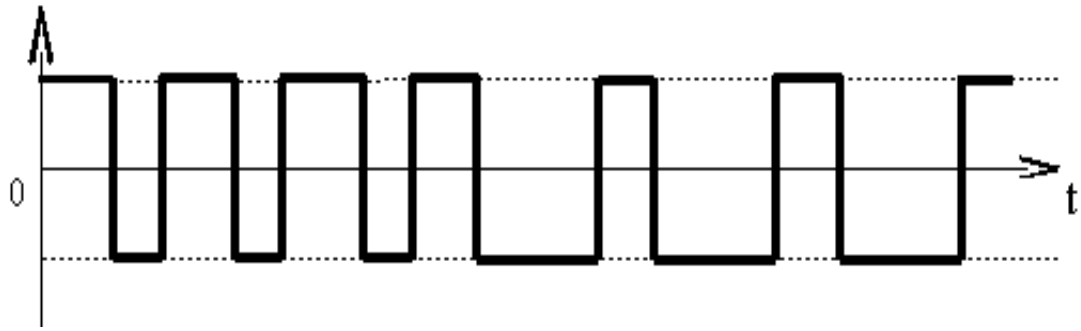
3 Navržené řešení

Stejnosměrné motory lze s výhodou budit pulzní šířkovou modulací (PWM z anglického *Pulse Width Modulation*), kdy se vhodnou frekvencí přerušovaně spíná stejnosměrné napětí přivedené na kotvu motoru, čímž vznikne obdélníkový průběh napětí a šířkou těchto pulzů lze volit velikost proudu protékajícího spotřebičem, která je dána střídou. Můžeme se setkat se dvěma definicemi střídy, a to: poměr šířky impulzu k mezeře [1] a poměr šířky impulzu k periodě. Zde bude lepší se přidržet druhé definice, jelikož vynásobením její velikosti napájecím napětím získáme přímo střední hodnotu napětí na výstupu modulace. Šířku impulzů tak lze měnit dvěma způsoby: změnou střídy při zachování frekvence nebo změnou frekvence při konstantní šířce pulzu resp. mezery (popřípadě kombinací obou dvou způsobů). Jako výhodnější se jeví první možnost, neboť umožňuje měnit střídu v uzavřeném intervalu <0, 1>, a tedy dosahovat i stavu vypnuto a maximálního buzení, které je zdroj schopen dodat. Motory je ale třeba řídit obousměrně. A i zde lze zvolit ze dvou možností, zda se bude budit buď jedna polarita a nebo druhá (obr. 3.1) a nebo druhá možnost – budit obě najednou s tím, že kladný impulz nevystřídá mezera, ale impulz opačné polarity (obr. 3.2). Při řízení oběma polaritami však motorem protéká proud i když stojí, což by v tomto případě bylo nežádoucí – je nutné mít možnost odepnout od motoru napájení. Proto je výhodnější způsob řízení každého



směru zvláštním PWM signálem, jednoho pro kladný, druhého pro záporný smysl otáčení.

Obr. 3.1: Příklad pulzní šířkové modulace prvního druhu.



Obr. 3.2: Příklad pulzní šířkové modulace druhého druhu.

Bude-li potřeba regulovat rychlost, použije se přímo PI-regulátor se zpětnou vazbou. Pro řízení polohy přepne regulační obvod na zapojení s jedním regulátorem řídícím polohu s výstupem pro rychlost a s vnitřní smyčkou, která se bude starat o stabilizaci rychlosti motoru. PI-regulátor se skládá z následujících složek: člen pro vyhodnocení regulační odchylky z rozdílu skutečné a požadované hodnoty, proporcionálního členu, který násobí regulační odchylku a integračního členu, ten nasčítává hodnoty regulační odchylky násobené příslušnou integrační kostantou. Výsledek je sečten ve výstupní regulační zásah. Vhodným nastavením takového regulátoru dosáhneme uspokojivého řízení rychlosti i polohy s nulovou odchylkou po dosažení ustáleného stavu. Při využití u NC stroje bývá často požadavkem řídit bez překmitu. Toho lze také dosáhnout správným nastavením hodnot regulátoru. Protože je třeba zařízení implementovat do číslicového obvodu, musí být proveden v diskrétní formě. Integraci tedy nahradí sumace.

Nemůžeme však zapomenout, že řízený ani řícicí systém není ideálně linární a to z důvodu fyzikálních omezení typu saturace. Motor má maximální otáčky, které nepřekročí, napájecí zdroj je omezen maximálním napětím a i datové struktury v řídicím systému jsou omezené. Nepříjemným důsledkem tohoto jevu by mohl být tzv. windup (přesycení), ke kterému dochází vlivem integrátoru. Bude-li totiž po nějakou dobu výstup regulátoru v saturaci, nasčítá se do sumačního členu velká hodnota, která se začne zmenšovat teprve po přesažení požadovaného stavu a bude dlouho trvat, než se z velkého čísla ustálí. Z toho důvodu je výhodné omezit amplitudu signálu už v místě integrace – tzv. antiwindup. [3]

Součástek pro vývoj číslicových zařízení je mnoho druhů s různými vlastnostmi. V této práci byl zvolen Spartan II XC2S100, jehož kapacita je až 100 K ekvivalentních hradel, 2700 Logic Cells, CLB pole 20_30, má využitelných 196 vstupně-výstupních vývodů, vyrábí se v technologii 0,22 _m, obsahuje bloky RAM po 4 Kb v deseti blocích a pracuje s logickými hodnotami v rozmezí

2,5 – 3,3 V. Po zapnutí je čip schopen sám natáhnout konfigurační data do své paměti z externí PROM. Uvnitř máme k dispozici několik různých hodinových signálů s fázovými závěsy a kmitočtovými násobičkami. Způsob programování, který jsem použil, je ISP (In System Programming) rozhraní pomocí sběrnice JTAG v Boudary Scan módu. Malou nevýhodou tohoto čipu je, že ztrácí konfiguraci po vypnutí napájení, je tedy nutno po každém zapnutí znovu natáhnout program. Tento problém můžeme snadno vyřešit právě externí EPROM a správným nastavením hodnot na konfiguračních vývodech čipu.

Struktura FPGA

Programovatelné hradlové pole FPGA se skládá z bloků CLB (Configurable Logic Blocks). Jsou to jednotky, ve kterých se implementují jednoduché logické funkce za pomoci tabulky LUT (Look Up Table) a dalších logických členů, jako např. XOR. Jednotlivé CLB obsahují propojovací matice GRM (General Routing Matrix) spojené se sousedy ve všech čtyřech směrech. Kromě nich jsou ještě pospojovány linkami na větší vzdálenosti, některé vedou až ke konci pole a část z nich je obousměrná. Čtyři vyhrazené primární globální propojovací sítě se zvláštními I/O vývody zajišťují distribuci hodinového signálu s vysokou kvalitou (velkou strmostí hran). Sekundární globální propojovací síť dává uživateli možnost využití dvanácti dalších volných globálních linek. Po obvodu pole je vytvořen propojovací prstenec zvaný VersaRing, pomocí něhož snadno zaměníme pořadí I/O vývodů bez změny naprogramování vnitřních propojení. Velikou výhodou jsou také třístavové sběrnice a propagace aritmetického přenosu mezi sousedními bloky. Všechny vstupy a výstupy jsou opatřeny volitelnými registry se společným hodinovým a odděleným signálem uvolnění a synchronními i asynchronními reset a set. Rovněž máme k dispozici vnitřní pull up a pull down rezistory. Pokud potřebujeme syntetizovat RAM v logickém bloku, máme tyto možnosti: distribuovaná synchronní RAM 16_2 nebo 32_1, popřípadě dvoubránovou konfiguraci 16_1. Kromě toho se nabízejí k použití zmíněné bloky synchronní RAM s kapacitou celkem 40 Kb.

Hardwarové a softwarové vývojové prostředí

Pro vývoj aplikací s programovatelnými poli nabízí dnešní trh hotové vývojové desky osazené vybranými typy hradlových polí s dalšími doplňkovými prvky. S polem Spartan II se vyrábí například Spartan II Development Board 2S100-PAK-EURO [B], který byl využit k řešení

problému v této diplomové práci. Obsahem této pomůcky je Spartan II XC2S100 se všemi vstupně-výstupními piny vyvedenými na socket, regulátor napájecího napětí 3,3 V a 2,5 V, patice pro externí paměť PROM, JTAG programovací port, DB9 konektor s RS-232 transceiverem, dvoumístným sedmisegmentovým LCD displejem, několika indikačními LED, spínači a dalšími prvky. Ne všechny budou využity. Spartan je zde možno programovat dvěma hlavními způsoby: buď přes paralelní port po sběrnici JTAG přímo do čipu, a nebo nechat obvod naprogramovat ihned po zapnutí z externí naprogramované paměti PROM.

Spolu s vývojovým kitem je dodáváno návrhové prostředí WebPack (zdarma ke stažení na internetu [C]). V něm byl zrealizován návrh řídicího systému a po částech odsimulován. Tento software podporuje hierarchický způsob návrhu, což zlepšuje přehlednost a příjemnost práce. Nabízí také grafické způsoby návrhu a dobře ovladatelné překládání programu. Nechybí ani simulátor, ve kterém lze vizuálně ověřit správnou či nesprávnou funkčnost navržené architektury. Poměrně jednoduchým způsobem lze i vygenerovat programovací soubor a nakonfigurovat jím v Boundary Scan módu čip připojený k počítači sběrnicí JTAG.

Volba programovacího jazyka

K samotnému provedení návrhu zařízení je třeba zvolit jazyk. Jazykům vytvořeným k těmto účelům říkáme HDL (Hardware Description Language). První používané jazyky (např. ABEL, OHDL) byly závislé na architektuře obvodů, a tedy nepřenositelné a pro složité návrhy nejsou vhodné. Dnes máme k dispozici dva stejně mocné přenositelné, na architektuře nezávislé jazyky, kterými jsou Verilog HDL a VHDL (VHSIC Hardware Description Language). Z důvodu podobnosti jazykům, které jsem už znal, jsem zvolil VHDL. Existují dvě standardní verze tohoto jazyka: VHDL-87 a VHDL-93. Přidržel jsem se novějšího standardu, tedy VHDL-93. Jedná se o popisný neobjektový jazyk s datovými typy. Ne všechny jazykové přípustné sekvence ve VHDL jsou však syntetizovatelné, neboť jazyk je nástrojem mnohem obecnějším, lze jím popisovat i chování jiných sekvenčních soustav než jen logických obvodů a ty pak můžeme pozorovat v simulátoru.

Jazyk VHDL [2]

Každý proveditelný program VHDL se skládá alespoň ze dvou jednotek: **entity** a **architecture**.

kde entita popisuje vstupní a výstupní porty a parametry a architektura definuje chování modelu a strukturní implementaci (čímž je možno vytvářet hierarchický model). V jazyce máme k dispozici datové typy, signály, proměnné, konstanty. Mezi jednotlivými datovými typy je nutno provádět konverze. Porty mohou být vstupní, výstupní, vstupně-výstupní nebo buffer a na jejich úrovni lze použít i parametry. Z předdefinovaných typů máme k dispozici dvoustavovou logiku, třístavové signály (obohacené ještě o pomocné stavy zejména k snadné simulaci) vhodné k implementaci sběrnic, aritmetické typy včetně celočíselných a reálných s volitelnými rozsahy i uspořádáním, znakové řetězce a další. Je i možnost vytvořit vlastní datový typ, např. výčtový nebo pole. Porty lze doplnit i o parametry entity uvedené klíčovým slovem **generic**, což rozšiřuje univerzálnost funkcí a entit, které programujeme. Jazyk dovede přeložit i aritmetické operace se signály a proměnnými včetně násobení a dělení. Je možno volit mezi znaménkovými a neznaménkovými operacemi. Překladač pracuje jak s binárními, tak s dekadickými, hexadecimálními i oktalovými čísly. Pro práci s poli existuje nástroj k paralelnímu zápisu několika hodnot do řezu pole užitečný například k práci s vektory logických hodnot. Klíčové slovo **others** je přitom zkratkou k zápisu do ostatních (nevyjmenovaných) míst v poli. Dále budeme potřebovat podmíněné operace, a tedy příkazy **if** ... **then** ... **else** ... **end if;**. Každý signál má své atributy, kterými jsou mimo jiné. **event**, **high**, **low** pro určení rozsahů, změny hodnot a jiných vlastností. Jejich použití se provádí vyjmenováním signálu a atributu odděleného apostrofem. Tak je možno detekovat náběžnou hranu hodinového signálu: *if clk'event and clk='1' then* ... Ve VHDL-93 jsou navíc k použití operátory bitových posuvů a rotací pro práci s poli a další nové příkazy, které ovšem nebudem k této syntéze potřebovat.

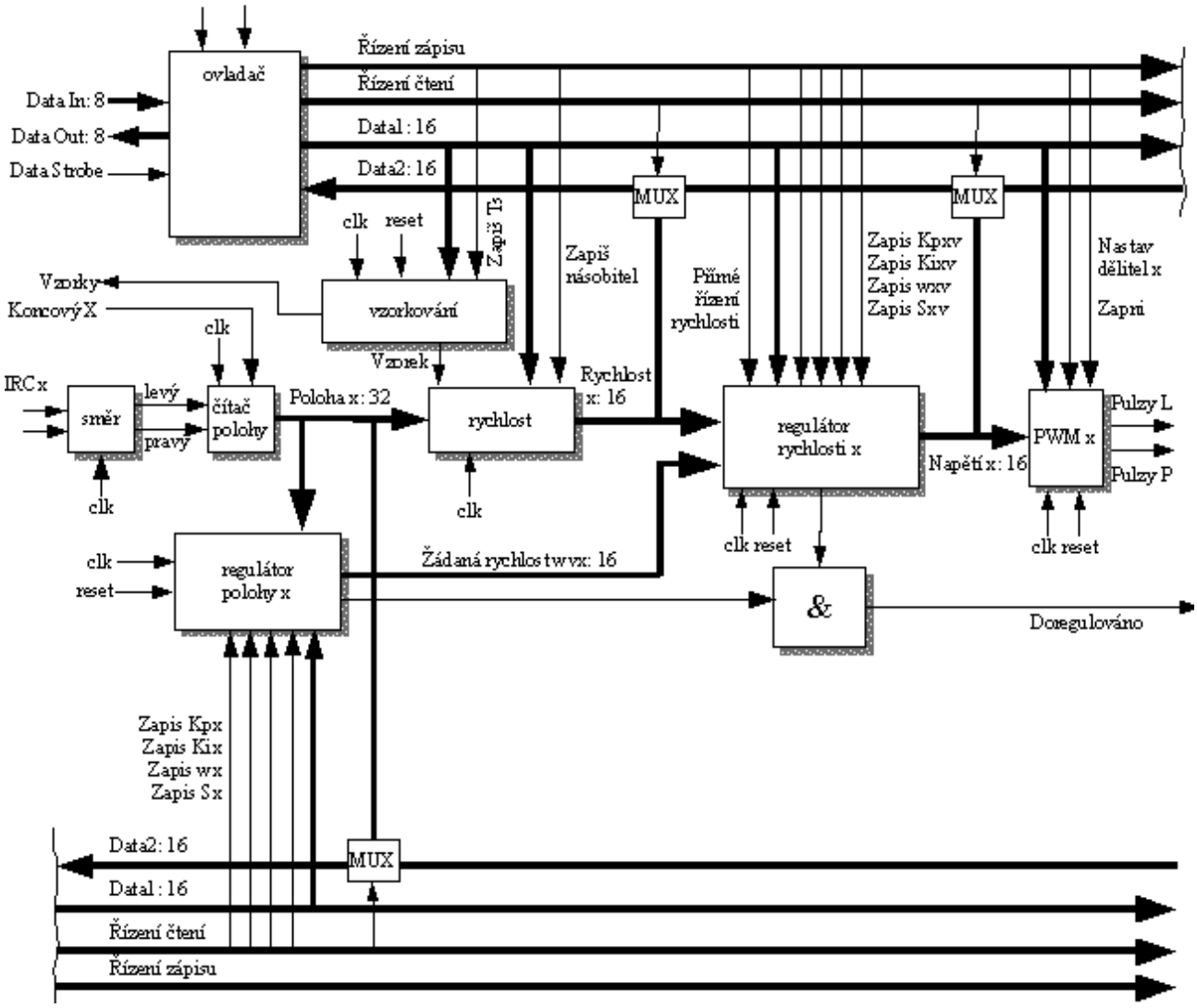
Volba typu komunikační sběrnice

V použitém vývojovém kitu sice je hotový konektor pro sériovou komunikaci RS-232, ale obsahuje pouze signály TX a RX (chybí synchronizační) a navíc je implementace i obsluha sériové komunikace složitější než paralelní. Z toho důvodu jsem zvolil paralelní osmibitovou sběrnici s odděleným kanálem pro čtení (8 bitů) a zápis (8 bitů) a synchronizačními signály, vše na logických úrovních odpovídajících TTL, je tedy možno sběrnici přímo připojit k paralelnímu portu LPT u běžného PC. Mějme tedy (z pohledu regulátoru) na sběrnici tyto signály: DataIn, DataOut, DataStrobe. Synchronizační signál se považuje za hladinově citlivý. O synchronizaci komunikace a provádění příkazů se stará stavový automat. Abychom mohli sběrnici připojit přímo k LPT portu u PC, musíme použít správný počet vstupních a vstupních signálů, jejich počet je omezen. Z pohledu Spartanu II bude vstupních devět a výstupních osm. Více jich není k dispozici. Devátý vstupní

signál bude mít význam synchronizace dat, kterou bude řídit PC. Jím se také budou strobovat čtená data ze systému. Protokol komunikace je uveden níže (tabulka 1).

Z důvodu ochrany zařízení před vnějšími elektrickými vlivy bude nutno všechny vstupy a výstupy galvanicky oddělit. K tomu účelu se využije hotového oddělovacího modulu se 16-ti vstupy a 16-ti výstupy obsahujícího optočleny PC 817, které sice pracují jen do 10 kHz, ale pro komunikaci to bude stačit.

Určitou potíž by mohly představovat napěťové úrovně logických hodnot na vstupech a výstupech Spartanu. Jejich jmenovitá hodnota je totiž 2,5 V a jsou schopné pracovat i s logikou 3,3 V. Zde se však požaduje komunikace při napětí do 5 V. Podle parametrů od výrobce [D] je však Spartan II možno připojit i přímo na pětivoltovou logiku, neboť vstupy i výstupy jsou dimenzovány na maximální napětí 5,5V a logické prahy si odpovídají. Výstupní proud smí být 10 mA, což k buzení TTL logiky stačí (při použití typů LS je to méně než 1 mA) a nebudou tedy výstupy programovatelného pole přetěžovány. To znamená, že zatížení výstupu nezmění napětí natolik, aby došlo k chybnému vyhodnocení logické úrovně. Obyčejné optočleny pro galvanické oddělení budou invertovat logické úrovně, s čímž se v FPGA snadno vypořádáme. Povšimněme si ještě signálů přicházejících z IRC snímačů. Je-li například počet impulzů takového snímače 200 na otáčku a rychlost otáčení hřídele stanovíme na 6000 otáček za minutu, získáme frekvenci impulzů v jednotlivých vodičích 20 kHz. Nad frekvencí 10 kHz už běžné optočleny typu dioda-tranzistor přestávají pracovat. Z toho důvodu bude nutno pro výstupy z IRC použít optočleny s vyšším frekvenčním přenosovým pásmem. Budou tedy typu dioda-dioda.



Obr. 3.3: : Blokové schema zapojení pro jednu osu.

Blokové schema vnitřního systému

Na obr. 3.3 vidíme navržené blokové schema řídicího systému pro jednu osu *x*. Většina bloků pracuje synchronně s hodinovým signálem *clk*. Navíc je přiveden signál *reset*, který po zapnutí nastaví počáteční konfiguraci. To je důležité, aby se nám řízený stroj nerozběhl dřív, než zadáme požadované parametry a nastavíme regulátory. Hlavním vstupem zařízení jsou signály přicházející z

IRC snímačů. Ty vyhodnocuje blok s titulkem *směr*, ze kterého vycházejí dva impulzní signály o směru otáčení. Ty pak vedou do vratného dvaatřicetibitového *čítače polohy* (číselná reprezentace je v doplňkovém kódu), který je resetovatelný koncovým spínačem na příslušné ose. Sběrnice s informací o poloze se potom větví do regulátoru polohy a členu pro měření rychlosti. *Rychlost* je šestnáctibitový signál a vyhodnocuje se jako rozdíl souřadnic polohy mezi dvěma následujícími vzorky a je násobený přednastaveným číslem. Tento způsob výpočtu má sice jisté nevýhody pro kvalitu regulace, ale je snadné jej implementovat jako logický obvod a nezabere mnoho místa v obvodu, neboť obsahuje jen odčítačku a násobičku. Takto vyhodnocená rychlost se přenáší do *regulátoru rychlosti*, kde se používá k výpočtu regulační odchylky. Vraťme se ještě k *regulátoru polohy*. Bylo by možné uzavřít jednoduchou regulační smyčku, kde na levé straně by byla měřená hodnota polohy soustavy a na druhé straně přímo napětí pro motor. Jako výhodnější se však pro servomechanizmus jeví uspořádání s vnitřní smyčkou, která obstarává rychlost motoru. Výstupem prvního regulátoru je potom požadovaná rychlost a ta je vstupem druhého regulátoru. Požadované stavy od uživatele se zapisují přes sběrnici *Data1* přímo do konfiguračních registrů K_i a K_p. Kromě řízení polohy ale můžeme z vnějšku řídit přímo rychlost motoru tím, že přepneme vstup regulátoru rychlosti. Ten pak nebude brát ohled na signály přicházející od regulačního členu pro polohu, ale bude uvažovat žádanou rychlost, která mu byla předána. Výstup budí pomocí sběrnice charakterizující napětí na motoru člen *PWM*, jehož výstupy najdeme na výstupech čipu jako *Pulzy P* a *Pulzy L*, každý pro jeden směr otáčení a jednu polaritu napětí. Před započetím regulace je nutno všem členům správně nastavit konstanty. U regulátorů jsou to K_p a K_i, jež zastupují proporcionální a integrační složku. Malou nevýhodou je, že většina konstant je v tomto technickém provedení závislá na vzorkovací periodě, a tudíž se musí předem spočítat ve vnějším zařízení, které s aplikací komunikuje. Na výpočet těchto funkcí uvnitř není dost místa a bylo by složité a časově náročné pro takové aritmetické operace vytvořit příslušné architektury ve VHDL. Navíc se v celém systému počítá pouze s celými čísly v doplňkovém kódu, což zmenšuje nároky na potřebnou plochu (nebo-li počet elementárních jednotek) pole narozdíl od reálných čísel, jejichž zpracování je podstatně složitější.

Dalším konfiguračním místem je nastavení frekvence pulzní šířkové modulace. Horní hranice je asi 700 Hz za předpokladu, že hodinová frekvence bude 50 MHz. Rozsah všech konstant je totiž 16 bitů. Rovněž periodu vzorkování musíme zvolit s ohledem na hodinový kmitočet. Aby se zabránilo nežádoucím výstupům z modulátoru PWM během konfigurování, lze deaktivovat všechny řídicí výstupy systému logickou nulou v signálu *Zapni*.

Jestliže budeme chtít řídit přímo rychlost jednotlivých os, potřebujeme mít možnost dotázat se

systému na okamžité souřadnice polohy. Užitečné bude i čtení aktuální rychlosti a napětí. O tyto funkce se starají multiplexory přepínající sběrnice s jednotlivými informacemi směrovanými na *Data2*, takže je možné je číst. Všechna data kromě souřadnic polohy jsou šestnáctibitová. Souřadnice polohy má rozsah dvojnásobný, a proto se zvlášť odesílá horní a dolní polovina. Je také potřeba znát čas, kdy bylo dosaženo požadovaného stavu, abychom mohli z vnějšku zadat další cíl. Tuto informaci dostaneme přímo jako signál *Doregulováno*, jehož hodnota je logickým součinem jednotlivých bitů od každého aktivního regulátoru. Posledním pomocným signálem je *Vzorek*, který nabývá logické hodnoty 1 právě v tom hodinovém cyklu, kdy se vzorkují hodnoty na vstupu. Signály *Doregulováno* a *Vzorek* můžeme číst na komunikační sběrnici na příslušných bitech po zadání příkazu k jejich kontinuálnímu čtení (podle tabulky 1).

Zbývá poslední blok, *ovladač*. Jeho úkolem je sledovat stav komunikační sběrnice a provádět příkazy, které z vnějšku dostává. Nazvěme soubor signálů vedoucích na vstupně výstupní piny jako vnější stranu a ostatní signály jako vnitřní stranu. Vnější sběrnice je osmibitová, a je tedy nutno vnitřní data transformovat na sekvenci nižšího a vyššího bajtu. Synchronizaci dat zajistí vstupní signál *DataStrobe*. Z vnitřní strany tato entita řídí všechny zapisovací a čtecí operace podle přicházejících příkazů. Vnitřní sběrnice na schematu zapojení bloků s šipkou směřující doprava jsou vzhledem k ovladači výstupní, doleva vstupní. Následuje tabulka příkazů pro blok *ovladač*. Čtení stavových bitů je uspořádáno tak, že na místě nultého bitu je signál *Vzorek* a na prvním bitu *Doregulováno.*

Tabulka 1.

HEX kód příkazu	Význam	Počet bajtů
00h	Čtení stavových bitů při aktivním *Strobe*	1
01h	Nastav dělitel pro frekvenci PWM	2
02h	Nastav hodnotu saturace v regulátoru rychlosti	2
03h	Nastav integrační činitel K_i pro rychlost	2
04h	Nastav proporcionální činitel K_p pro rychlost	2
05h	Zapiš žádanou hodnotu *w* rychlosti	2
06h	Nastav hodnotu saturace v regulátoru polohy	2
07h	Nastav integrační činitel K_i pro polohu	2
08h	Nastav proporcionální činitel K_p pro polohu	2
09h	Zapiš nižší bajt žádané hodnoty *w* polohy	2
0Ah	Zapiš vyšší bajt žádané hodnoty *w* polohy	2
0Bh	Nastav násobitel pro výpočet rychlosti	2
0Ch	Vypni přímé řízení rychlosti	0
0Dh	Zapni přímé řízení rychlosti	0
0Eh	Vypni generování PWM	0
0Fh	Zapni generování PWM	0
1Fh	Nastav periodu vzorkování	2
80h	Čti dolní bajt okamžité polohy	2
81h	Čti horní bajt okamžité polohy	2
82h	Čti okamžitou rychlost	2
83h	Čti hodnotu z PWM	2

Podmínky správného provozu

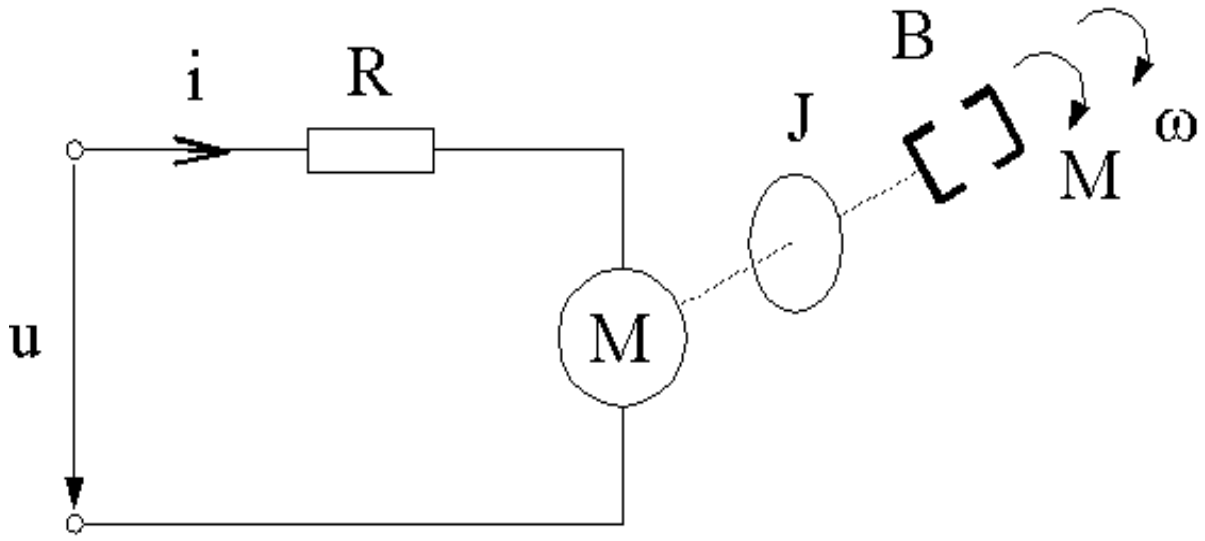
1.Při komunikaci musíme dodržet následující podmínky provozu. S ohledem na mezní přenosový kmitočet použitých optočlenů nesmí žádné impulzy na sběrnici mít kratší délku než 100 _s.

2.Při žádání příkazu je nutno vyzvednout všechna data podle příslušného počtu bajtů v tabulce, teprve potom smí přijít na sběrnici nový příkaz. Kód příkazu stačí ponechat na sběrnici pro první cyklus signálu *Strobe*.

3.Jestliže používáme příkaz pro čtení polohy, který je dvaatřicetibitový, musíme zajistit, abychom nejprve přečetli nižší a následovně vyšší bajt, jinak nedostaneme správnou hodnotu, neboť čtení nižšího bajtu zapíše vyšší polovinu do pomocného vyrovnávacího registru, odkud je posléze čtena. Chování systému při přijetí kódu příkazu neuvedeného v tabulce není definováno a může vést k nekorektnímu chování.

Model řízeného systému

Zvolme za model systému schema na obr. 3.4. Pro jednoduchost zanedbejme indukčnost motoru, uvažujme jen jeho vnitřní odpor a setrvačný moment s tlumením připojeným na hřídel. Stejného modelu použijeme pro obě osy stroje. Na vstupní svorky připojujeme řídící napětí, které vyvolá proud sériovou kombinací odporu kotvy a motoru. Proud je pak původcem otáčivého momentu na hřídeli.Otáčením hřídele vzniká napětí na kotvě motoru a tím se také zmenšuje prou protékající

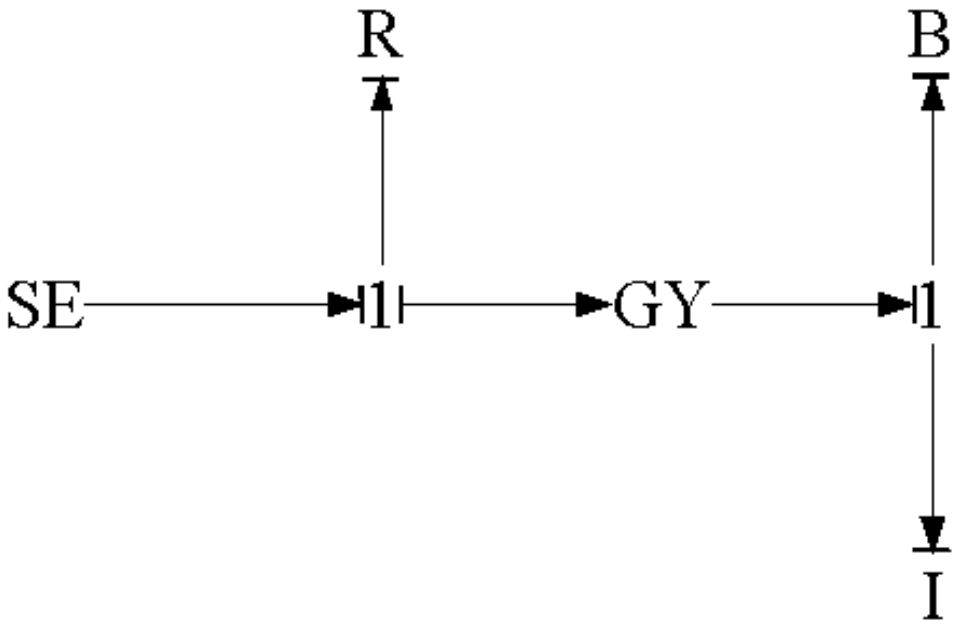


obvodem.

Obr. 3.4: Schematický model.

Před vytvořením stavového schematu sestrojíme vazební graf (*Bond Graph*) znázorňující tok

výkonu v soustavě pomocí šipek. Kolmá zakončení čar ukazují směry závislostí fyzikálních veličin. Symbol *SE* zastupuje zdroj napětí (určuje se napětí a důsledkem je proud). Číslice 1 je sériové spojení prvků. V něm vystupuje elektrický odpor kotvy *R* a motor namodelovaný jako gyrátor (napětí převádí na otáčky a moment na proud.) Za motorem končí systém sériovou kombinací tlumení *B* a momentem setrvačnosti, což je prvek *I*. Symbol _ znamená konstantu motoru, která

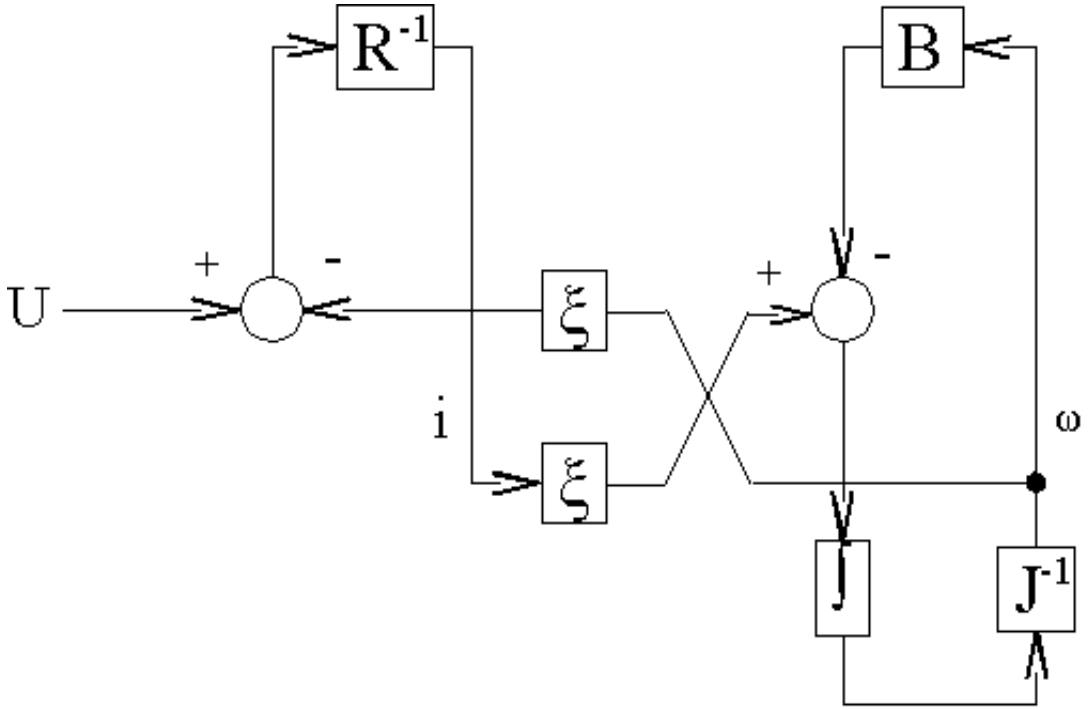


závisí na jeho konstrukci. Teď už můžeme přikročit k vytvoření stavového schematu.

Obr. 3.5: Vazební graf.

Vazební graf

První sériová kombinace znamená sčítání napětí ve směru výkonu. Z toho důvodu se objeví záporné znaménko u veličiny, která má význam napětí na kotvě motoru. Napětí na odporu *R* tedy získáme odečtením napětí kotvy od napájení. Odporem *R* pak bude protékat proud *i*, který je roven podílu výsledného napětí a odporu. Při průchodu motorem se proud transformuje na moment a napětí na otáčky. Další sériová kombinace znamená odčítání brzdného momentu způsobeného tlumením *B* od momentu motoru. Výsledný moment působí na setrvačnost hřídele a důsledkem je změna otáček. Tento vztah vyjadřuje symbol integrace _ s momentem setrvačnosti *J*.



Obr. 3.6: Stavové schema

Stavové schema

Nyní můžeme ze stavového schematu odvodit stavové rovnice. Za stavy bude nejlepší zvolit přímo výstup systému, kterým jsou otáčky hřídele. Dalším stavem, odvozeným z otáček, bude poloha. Tu získáme prostou integrací otáček s nějakou počáteční podmínkou (můžeme ji pro tento případ položit rovnu nule). Sestavme tedy stavové rovnice:

$$\dot{\varphi} = \omega \qquad (3.1)$$

$$\dot{\omega} = \frac{\xi U}{JR} - \omega \frac{\xi + BR}{JR},$$

Z rovnic dostaneme matice **A**, **B**, **C**, **D** charakterizující řízenou soustavu:

$$A = \begin{bmatrix} -\frac{\xi + BR}{JR} & 0 \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} \frac{\xi}{JR} \\ 0 \end{bmatrix} \qquad (3.2)$$

kde výstupem $C=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $D=\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ rozumíme $y=\begin{bmatrix} \omega \\ \varphi \end{bmatrix}$, který je přímo roven stavům.

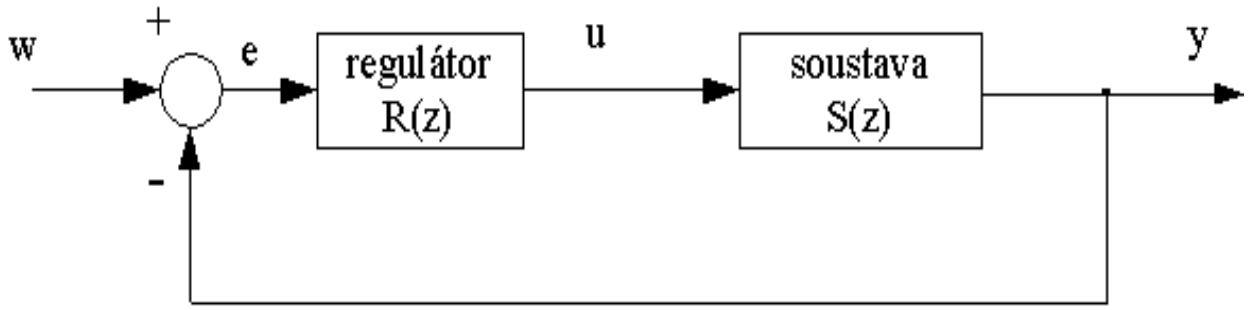
Z těchto matic spočítáme přenos systému jako

$$G(s)=\frac{\xi}{JRs+\xi+BR}=\frac{\frac{\xi}{\xi+BR}}{\frac{JR}{\xi+BR}s+1}=\frac{k}{\tau s+1}, \qquad (3.3)$$

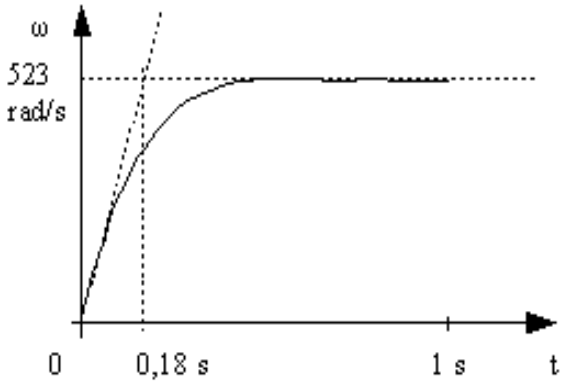
kde _ je časová konstanta. Protože neznáme parametry motoru, které vystupují ve výpočtech, je třeba je změřit. Nejsnáze získáme hodnoty k a _, jestliže změříme přechodovou charakteristiku. Chceme-li řídit systém diskrétním regulátorem, musíme spojitý přenos převést na diskrétní podobu za použití obrazu v *Z*-transformaci. To provedeme přibližnou metodou, kdy nahradíme operátor *s* výrazem $\frac{z-1}{T_s}$ a dostaneme $G(z)=\frac{kT_s}{\tau z-\tau+T_s}$.

Model diskrétního PS regulátoru má pak přenos $R(z)=r_0+\frac{r_{-1}z}{z-1}T_s=r_0\left(1+\frac{z}{T_i(z-1)}T_s\right)$ a uzavřená regulační smyčka podle obrázku 3.7 bude s přenosem $G(z)=\frac{R(z)S(z)}{1+R(z)S(z)}=\frac{kB(z)}{kB(z)+A(z)}$, kde *B(z)* je čitatel, *A(z)* jmenovatel a *k* zesílení otevřené smyčky.

Zbývá nastavit konstanty regulátoru tak, abychom splnili požadované vlastnosti na dynamiku systému, regulační odchylky (polohy i rychlosti), a aby byly stabilní. Pro řízení polohy je navíc třeba zajistit, aby nedošlo k překmitu.



Obr. 3.7: Uzavřená regulační smyčka.



Obr. 3.8: Přechodová charakteristika

Identifikace systému a návrh regulátoru

Abychom získali parametry řízeného systému, musíme jej identifikovat. Nejsnáze se to provede změřením parametrů přechodové funkce. Naměřená křivka je na obr. 3.8. Z nich nás zajímá doba náběhu a ustálená hodnota v nekonečnu. Z naměřené přechodové charakteristiky určíme časovou konstantu _ vynesením tečny ke křivce v počátku. Tam, kde polopřímka protnula vodorovnou asymptotu přechodového průběhu, najdeme druhý parametr modelu – hodnotu _, jejíž velikost je 0,2 s. Nejprve zvolíme periodu vzorkování T_s podle doby náběhu tak, aby byla rovna zhruba její desetině. V našem případě to je 0,02 s a od této chvíle pracujeme kromě spojitého času t s

diskrétním časem *n*, kde platí rovnost $t=nT_s$. Předpokládáme-li model systému podle (3.3), získáme jeho parametry. Limita naměřené posloupnosti v nekonečnu $\lim_{n\to\infty} f_n = \lim_{z\to 1,} (z-1)F(z)$, kde *F(z)* je *Z*-transformace funkce f_n a *z* je operátor této transformace. Parametr modelu *k* spočítáme z této limity dosazením za *z* (nebo *s* ve spojitém modelu) a bude mít hodnotu 523, jestliže vstupem byla hodnota 1. Získali jsme tím číselný přenos modelu řízeného systému *G(z).*

Regulační algoritmus

1.u(k+1)=e(k)_*K_p+__*K_i

2._ = _ +e(k)

Konstanty K_p a K_i je nutno přepočítat na celá čísla pomocí následujících koeficientů, protože uvnitř systému dochází vlivem zjednodušení výpočtů k vynásobení konstantou. K_p a K_i se uvnitř dělí 256, aby bylo možno použít K_p a K_i jako desetinná čísla. Jsou tedy s vynecháním nuly v rozsahu <0,0039; 256> pro obě konstanty.

Dalším multiplikátorem je násobitel K_r použitý při výpočtu rychlosti. Snímá se počet pulzů ve směru pohybu během periody vzorkování a pak se vynásobí tímto číslem.

Při výpočtu těchto koeficientů pro regulaci použijeme následujících vzorců pro polohu

$$K_p = 256\,P \tag{3.4}$$

$$K_i = \frac{256\,P\,T_s}{T_i} \tag{3.5}$$

a pro rychlost

$$Kp = \frac{256\,P}{k} \tag{3.6}$$

$$K_i = \frac{256\,T_s}{k\,T_i} \tag{3.7}$$

$$K_r = \frac{k}{T_s}, \tag{3.8}$$

kde k je zvolený koeficient ovliňující rozsah hodnot rychlosti. Volba k závisí na kontkréním řízeném modelu.

Z použité periody vzorkování T_s spočítáme dělitele D pro děličku kmitočtu na vzorkování

$$D = \frac{T_s}{T_{clk}}. \tag{3.9}$$

Z přenosu uzavřené nebo otevřené smyčky už můžeme navrhnout parametry PI regulátoru, například pomocí frekvenčních charakteristik. Také lze použít přibližnou metodu podle Ziegler Nicholse [3], pro kterou bychom museli změřit mezní zesílení regulátoru a kritickou periodu kmitů. Nejjednodušší metodou je ruční cyklická optimalizace konstant regulátoru, která byla použita pro počáteční odzkoušení chodu systému.

4 Výsledky práce

Při návrhu zařízení vznikly komplikace s místem na čipu. Ukázalo se, že se do čipu nevejde dvouosé řízení při sebevětší optimalizaci. Požadovaný počet LUT přesahoval 136% těch, které jsou k dispozici. Z toho důvodu bylo nutno úlohu zredukovat s tím, že se bude řídit pouze jedna osa. Větší programovatelné pole není v současné době na katedře k dispozici. Je také možné použít dvě pole FPGA, každé pro jednu osu, takže vlastně dva ekvivalentní nezávislé řídicí systémy. Po redukci na jednoosé řízení je nyní využití LUT na čipu přibližně 90%.

Požadovaný řídicí systém se po těchto úpravách podařilo implementovat do programovatelného hradlového pole Spartan II. K zjištění, které části systému zabírají nejvíce místa bylo nutno udělat analýzu. Při ní se jevilo, že nejvíce místa zabírají paralelní násobičky, což se ověřilo zredukováním jejich počtu, takže procento využití plochy čipu dramaticky kleslo. Využitelných LUT je v čipu 1200 a projekt vycházel v původním provedení na 2500. Paralelních násobiček bylo v systému celkem deset. Proto bylo výhodné vytvořit pouze jednu sdílenou a přepínat její vstupy a výstupy pomocí multiplexoru, který zabere podstatně méně místa.

Při prvních pokusech o naprogramování desky se Spartanem vznikl ještě jeden problém. Data se sice správně načítala z připojené PROM, ale ač byly všechny jumpery přesně podle manuálu, nefungovalo programování přes sběrnici JTAG. Posléze se shledalo, že v manuálu jsou chybně popsané některé konfigurace jumperů a po jejich opravení už sběrnice pracovala správně.

Ke galvanickému oddělení vstupů a výstupů byl použit hotový modul s šestnácti vstupy a výstupy obsahující optočleny PC 817 pracující řádově do 10 kHz. Pro oddělení signálů z IRC snímačů byly použity optočleny HCPL 2530 s mezní frekvencí větší než 1 MHz.

Taktovací frekvence kitu se Spartanem je nastavena na 50 MHz. Lze ji sice změnit, ale z časových důvodů jsem se programováním čipu DS 1073 nemohl zbývat. Přednastavená frekvence je dostačující, je rovna polovině maximálního kmitočtu. Programovací vývod je vyveden ke Spartanu, kterým je pak možno nastavit dělitel podle komunikačního protokolu [E].

4.1 Diskuse řešení

Úloha byla zhotovena tak, aby bylo možné ji v daném rozsahu a čase stihnout dokončit a ověřit její vlastnosti. Z toho důvodu nebyly zdokonaleny některé vlastnosti řidicího systému. Tyto doplňky by mohly být námětem k další práci. Nebyl navržen filtr pro měřenou rychlost, ač by byl vhodný, jelikož se jedná o derivační článek. Rychlost je měřena způsobem vhodným pro vyšší otáčky, při nízkých hodnotách by bylo vhodnější měřit periodu signálů z IRC. Rovněž by bylo dobré doplnit ovládací příkazy o funkci čtení konfiguračních hodnot z důvodu verifikace. Není však jisté, zda by se tato rozšíření nedotkla velikosti daného FPGA. V opačném případě by buď bylo nutno úsporněji minimalizovat syntézu, nebo pořídit větší programovatelné pole. V každém případě je však tato práce kompromisem mezi složitostí syntetizovaného logického obvodu a vnějšího ovládacího zařízení, na kterém vžycky spočívá určitá část regulačního procesu. V případě realizace, která byla provedena, je na vnějším zařízení nejsložitější operací výpočet konstant regulátoru. Kromě těchto vlastností by také mohlo být učitečné rozšířit systém o třetí osu. Z hlediska elektrického je možné navrženou sběrnici zaměnit za sériovou RS232, případně USB s tím, že by programování bloku pro řízení takové komunikace bylo složitější. Dalším vylepšením pro zlepšení univerzálnosti by mohl být příkaz pro nastavení aktivní logické úrovně pro vstup z koncového spínače.

Bylo by dobré při návrhu odsimulovat celý návrh a tím jej otestovat. To ovšem není proveditelné, neboť software není schopen zpracovat tak obrovské množství hodinových cyklů. Musely by se snížit všechny dělicí konstanty v čítačích a to by byl veliký zásah do projektu.

4.2 Ověření funkčnosti

Zařízení správně reguluje rychlost v rozsahu od 200 otáček za minutu do maxima bez překmitu, jestliže nastavíme správné konstanty regulátorů. Při nižších rychlostech se projevuje nelinearita způsobená čepovým třením a dochází ke kmitání.

Při programování Spartanu II se někdy stává, že se chybně nakonfiguruje. Je proto lepší jej nejprve zresetovat tlačítkem S1 na desce, čímž se tento problém spolehlivě odstraní.

Z praktických důvodů je invertovaný signál o doregulování vyveden na svítivou LED na desce, takže signalizuje činnost regulátoru. Je-li zhasnutá, systém doreguloval a je v klidu.

Před použitím zařízení pro řízení konkrétního modelu je třeba přizpůsobit nastavení konfiguračních registrů jeho elektromechanickým vlastnostem. Při tom nemusí být zaručena regulace bez překmitů, jestliže je systém příliš nelineární nebo pokud je na osu motoru připojena tak proměnná zátěž, že by její změny regulátor nestihl vyrovnávat. Díky opatření proti přesycení integračního členu nevadí dočasné zastavení motoru vnější silou, takže nevznikne nežádoucí překmit. Činnost byla odzkoušena při tomto nastavení:

$K_i = 0010h$

$K_p = 0210h$

$T_s = 0{,}02\ s$

$f_{clk} = 50\ MHz$

$K_r = 1$

5 Závěr

Elektronický systém pro řízení víceosého obráběcího stroje byl úspěšně vyřešen v jazyce VHDL a čipu Spartan II, jeho vlastnosti ověřeny a odpovídaly předpokládaným hodnotám. Zařízení splňuje požadavky dané v zadání s tím, že k použití pro dvě osy je třeba použít dvě stejná zařízení, a je použitelné k řízení víceosého servomechanismu. Z důvodu nedostatečné kapacity čipu nebylo možné, aby se vešel systém pro dvě osy do jednoho čipu, a proto byl návrh rozdělen na dvě samostatné ekvivalentní části, každá pro jednu osu. Po nutných optimalizacích bylo využití LUT ve Spartanu přibližně 90% narozdíl od předchozích 136% při pokusu o vytvoření dvouosého zařízení. Pokud bychom měli trvat na tom aby se vešel projekt do jednoho čipu, zvolili bychom typ s větší kapacitou.

Rozlišení všech číselných hodnot v systému je 16 bitů kromě souřadnic polohy, které je 32 bitů. Tako PWM má příslušný počet úrovní, neboť pracuje se šestnáctibitovými hodnotami, avšak se znaménkem, takže výsledný počet úrovní pro jeden směr je poloviční. Umožňuje ale nastavit jak nulovou, tak maximální (saturační) hodnotu.

Poděkování

Zejména chci tímto poděkovat za spolupráci a věnovaný čas ke konzultacím týkajících se zejména programování v jazyce VHDL panu Ing. Torstenu Törökovi z firmy TTC Telekomunice.

6 Vysvětlivky

b zkratka pro jednotku informace **bit**

K zkratka pro předponu **kilo**, v tomto případě s významem používaným v číslicové technice, tedy násobek 1024

LUT z anglického Look Up Table, znamená tabulku pro definici logické funkce

7 Literatura

[1]Láníček, R.: *Obvody – součástky – děje.* Praha, BEN – technická literatura 1998

[2]Douša, J.: *Jazyk VHDL.* Praha, FEL ČVUT 2003

[3]John, J: *Systémy a řízení.* Praha, FEL ČVUT 1996

[A]http://www.insight.memec.com

stránka Insight, výrobce kitu se Spartanem II

[B]www.insight-electronics.com/solutions/kits/xilinx

prezentace vývojového kitu

[C]www.xilinx.com/sxpresso/webpack.htm

stránka se stáhnutelným Webpack návrhovým softwarem

[D]http://rt.cs.tu-berlin.de/lehre/aes/xilinx-spartan2-switching.pdf

elektrické specifikace Spartanu II

[E]http://pdfserv.maxim-ic.com/arpdf/DS1073.pdf

programování čipu DS 1073 jako děliče kmitočtu

8 Příloha na disku

Zdrojový kód programu ve VHDL – seznam modulů

Zaklad.vhd (top level)

Citacpolohy.vhd (čítač polohy)

Nasobicka.vhd (paralelní násobička 16ī16 b)

Ovladac.vhd (komunikacni procesor)

PWM.vhd (generátor pulzní šířkové modulace)

Regulator.vhd (diskrétní podoba PI regulátoru)

Rychlost.vhd (diferenční člen pro výpočet rychlosti pohybu)

Smer.vhd (detektor směru pohybu ze signálu od IRC)

Vzorkovani.vhd (frekvenční dělička pro generování vzorkovacích pulzů)

Zaklad.ucf (user constraints file – popis namapování vstupů a výstupů)