

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Hostnik

**SISTEM ZA RAČUNALNIŠKO
KRMILJENJE STRUŽNICE**

Diplomska naloga
na univerzitetnem študiju

Mentor: prof. dr. Dušan Kodek

Ljubljana, 2008

Zahvala

Iskrena hvala najprej mentorju prof. dr. Dušanu Kodeku za vse nasvete in potrpežljivo vztrajanje.

Zahvaljujem se članom LAPS, zlasti Zvonetu Petkovšku, za nesebično pomoč in potrpežljivost v tednih testiranja in izdelave vezja.

Hvala bratu Tomažu, ki mi je s svojo iznajdljivostjo na tem področju dal navdih in možnost za to diplomsko delo.

Najlepša zahvala pa gre staršem in tudi vsem ostalim, ki ste me v tem obdobju spodbujali in podpirali.

Vsebina

Povzetek	1
1. Uvod	3
2. Predstavitev problema	5
2.1. Stručnica.....	5
2.2. Krmiljenje suporta.....	6
2.3. Povratna informacija	7
3. Zamisel rešitve	9
3.1. Shema	9
3.2. Logične funkcije vezja	10
3.3. Želene zmogljivosti	10
3.3.1. Frekvenca tvorjenja signalov	10
3.3.2. Frekvenca zajemanja signalov	11
3.3.3. Pasovna širina komunikacije	11
4. Izbira tehnologije	13
4.1. Komunikacija z osebnim računalnikom	13
4.1.1. IEEE 1284.....	13
4.1.2. RS-232	13
4.1.3. USB.....	14
4.1.4. PCI	14
4.2. Realizacija logike	18
4.2.1. Prilagojeno vezje.....	18
4.2.2. Mikrokrmilnik.....	19
4.2.3. DSP	19
4.2.4. FPGA	19
4.3. Izbrana kombinacija	20
4.3.1. Razvojna plošča Raggedstone1	21

5. Realizacija	23
5.1. Kodiranje	23
5.1.1. Komunikacija PCI.....	24
5.1.2. Štetje impulzov	27
5.1.3. Krmiljenje elektromotorjev	29
5.1.4. Celotno digitalno vezje znotraj FPGA čipa	29
5.2. Vezje.....	31
5.2.1. Priklop dajalca impulzov	32
5.2.2. Priklop elektromotorja	33
5.2.3. Priklop poljubnih vhodnih signalov	34
5.2.4. Priklop poljubnih izhodnih signalov	34
5.3. Uporabniška navodila	35
5.3.1. Namestitev	35
5.3.2. Shema in priklop signalov	36
5.3.3. Programski dostop	38
5.3.4. Primer uporabe	41
6. Sklepne ugotovitve	43
Dodatek A – Fotografije sistema	45
Uporabljeni viri	49

Seznam uporabljenih kratic

AC	Alternating current
BIOS	Basic input/output system
BLDC	Brushless DC electric motor
CNC	Computer numerical control
DC	Direct current
DIP	Dual in-line package
DSP	Digital signal processor
FPGA	Field-programmable gate array
HDL	Hardware description language
JTAG	Joint test action group
NC	Numerical control
PCI	Peripheral component interconnect
PLD	Programmable logic device
UART	Universal asynchronous receiver/transmitter
VHDL	VHSIC hardware description language
VHSIC	Very-high-speed integrated circuits

Povzetek

V tem diplomskem delu je opisan razvoj sistema za krmiljenje stružnice s pomočjo osebnega računalnika. Predstavljen je sistem, ki elektroniko na stružnici naredi dostopno računalniku preko registrov. Pod elektroniko so mišljeni elektromotorji, senzorji in stikala. Delo se osredotoča predvsem na sam sistem, medtem ko mehanično predelavo stružnice na eni strani in programsko obdelavo kontrole na drugi prepušča uporabniku.

V drugem poglavju, takoj po uvodu, je predstavljen problem predelave stružnice – kaj je potrebno krmiliti, ter vse naprave, ki so tu potrebne. Za opravljanje pomikov so izbrani *enosmerni elektromotorji s krtačkami*. Kontrolni sistem pri njih potrebuje tudi povratno informacijo, za kar je uporabljena naprava *dajalec impulzov*. Sistem torej deluje po principu *zaprte zanke*.

V tretjem poglavju je predstavljena zamisel sistema: vezje, ki opravlja nizkonivojske operacije ter da vse krmiljenje na voljo računalniku preko registrov. Predstavljene so tudi konkretne funkcije, katere mora vezje opravljati, ter izračun potrebnih zmogljivosti.

V četrtem poglavju so predstavljene tehnologije, ki so na voljo, na koncu pa tudi izbrana kombinacija - razvojna plošča *Raggedstone1*, katere jedro predstavlja *Spartan-3 FPGA* čip in ima *PCI vmesnik* za komunikacijo z računalnikom.

V petem poglavju je predstavljena realizacija same rešitve. Opisana je funkcionalnost posameznih modulov v programskem jeziku *VHDL*. Tu so tudi opisane težave s protokolom *PCI* in njihovo reševanje. Predstavljen je *dodaten modul vezja*, ki je nastal ob plošči *Raggedstone1* za zaščito, optično izolacijo in napetostno pretvorbo signalov. Zadnji del tega poglavja predstavi uporabnost naprave skozi uporabniška navodila - kam in kako se priklapi vhodno/izhodne signale, preko katerih registrov se dostopa do njih itd. Na koncu poglavja se nahaja tudi primer uporabe naprave v operacijskem sistemu *Linux*.

Ključne besede: stružnica, CNC, krmiljenje, enosmerni elektromotor s krtačkami, dajalec impulzov, *PCI*, *FPGA*.

1. Uvod

Obdelovalni stroji pomagajo človeku že stoletja. Prvi začetki segajo v konec osemnajstega stoletja, kjer so v obliki preprostih rezkalnikov bili pogoj za izdelavo natančnih cilindrov in tako botrovali nastanku industrijske revolucije. Danes predstavljajo osnovo naše industrije in so posredno ali neposredno uporabljeni v proizvodnji praktično vseh izdelkov moderne civilizacije.

Prve obdelovalne stroje so upravljali in celo poganjali ročno. Kmalu po drugi svetovni vojni pa so se razvili stroji NC (*Numerical control*), ki so postopke obdelave avtomatizirali z uporabo shranjenih ukazov. Tako so omogočili serijsko izdelavo v velikih količinah in hkrati v večji natančnosti. Ukaze so tem strojem večinoma podajali ročno luknjane na papirne ali druge trakove, kar je bilo precej zamudno, poleg tega pa je bilo program zelo težko spreminjati.

Kmalu za stroji NC so ugledali luč sveta stroji CNC (*Computer numerical control*), ki so sistemu dodali računalnik. Le-ta je človeku precej poenostavil krmiljenje obdelovalnih strojev in prinesel nove možnosti. Krivulje je postalo enako težko rezati kot ravne linije, kompleksne 3-D strukture so se relativno preprosto izdelovale, in število korakov, ki so zahtevali prisotnost človeka, se je konkretno zmanjšalo. Tako so CNC operaterji pridobili čas, da se posvetijo dodatnim nalogam. Avtomatizacija CNC je torej zmanjšala število napak, razširila spekter možnosti izdelave in skrajšala čas za menjavo programa na stroju. Več o NC in CNC je podano v [2] in [3].

Še vedno pa ostaja na trgu precej obdelovalnih strojev, ki nimajo CNC niti NC podpore. Vendar to še ne pomeni, da jim ta podpora ne more biti dograjena. V tem diplomskem delu je poudarek predvsem na njihovi najbolj tipični in najpogostejši predstavnici – stružnici. Hkrati je potrebno še dodati, da v naslednjih vrsticah omenjeni sistem ni omejen le na stružnice, ampak je uporaben na vseh obdelovalnih strojih.

Zamisel je torej preprosta – ceneno staro stružnico povezati z osebnim računalnikom in jo tako predelati v moderen CNC stroj. Tako se pridobi dober uporabniški vmesnik, zelo prilagodljiv sistem in visoko procesno moč za nizko ceno.

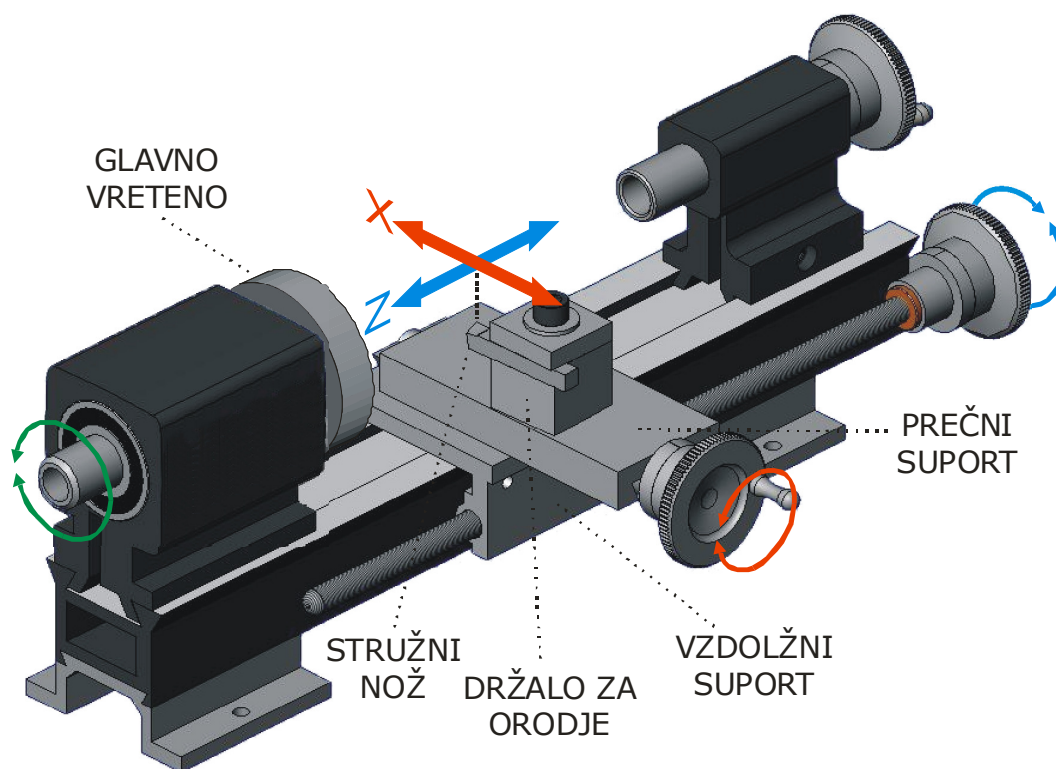
Seveda so potrebne tudi mehanične prilagoditve, vendar se to delo zaradi obsežnosti osredotoča predvsem na vmesnik z računalnikom - vmesnik med motorji in senzorji, ki poganjajo vodila stružnice, ter osebnim računalnikom, ki vodi vse dogajanje. Omenjeni vmesnik predstavlja zaokrožen problem, ki je že sam po sebi precej obsežen, zato se pričujoče delo ne pogloblja v samo krmiljenje oziroma nadzor, ampak predstavlja samo vmesnik zanj. Krmiljenje se lahko izvede na zelo različne načine, od uporabe trivialnih pa do uporabe zelo zapletenih in optimiziranih algoritmov. Da se ne izumlja tople vode, se lahko prilagodi tudi kakšnega od obstoječih programov, kot je npr. EMC2 (glej [11]).

Na trgu sicer že obstajajo podobne rešitve, vendar v večini uporabljajo paralelna vrata za komunikacijo in koračne motorje za premikanje vodil. To dvoje pa ne omogoča želene hitrosti in natančnosti, kot bomo ugotovili v naslednjih poglavjih.

2. Predstavitev problema

2.1. Stružnica

Stružnica je nasploh zelo poznan stroj in je kot take najbrž ni potrebno posebej predstavljati. Vsekakor pa je važen vidik, skozi katerega sistem vidi stružnico.



Slika 2.1: Tipičen primerek stružnice z označenimi osnovnimi deli

Na sliki 2.1 je prikazan enostaven model stružnice. Na njem so označeni deli, ki so potrebni za razumevanje koncepta stružnice. Prvi je glavno vreteno, kamor se vpne obdelovanec in se vrti pod veliko hitrostjo. Drugi pa je držalo za orodje, v katerega je vpet stružni nož. S pomočjo dveh pravokotnih suportov se nož lahko premika kjerkoli po dvodimenzionalnem prostoru. Tako se lahko ustvari poljuben krožno-simetričen kos. Vsak suport se premika s pomičnim vretenom, katerega se na sliki vrti ročno s pomočjo označene ročke. Podrobnosti so v [1].

Pri predelavi stružnice v CNC stružnico je potrebno na obe pomični vreteni montirati elektromotor in po potrebi še senzor pozicije, poleg tega pa še na več mest razna stikala, ki lahko vklapljajo tudi kakšno dodatno napravo, kot je npr. črpalka hladilne tekočine. Vse te komponente proizvajajo različne signale ali pa se s signali le-te krmili.

Sistem za krmiljenje torej vidi stružnico kot tri elektromotorje (eden na glavnem vretenu ter dva za pomik suportov) in njihove senzorje pozicije ter neko množico stikal, bodisi vhodnih bodisi izhodnih. Naloga sistema je na eni strani brati in krmiliti te signale ter na drugi strani dati dostop do njih računalniku.

2.2. Krmiljenje suporta

Da se suport lahko krmili s pomočjo računalnika, ga je torej potrebno opremiti z elektromotorjem. Tako se lahko preko sistema za krmiljenje in elektromotorja določa pozicija stružnega noža na suportu. Glede same vrste elektromotorja pa je velika izbira. Na področju strojev CNC so se glede krmilnih motorjev stvari razvijale nekako čez tri generacije.

Prva generacija uporablja koračne motorje. Njihova prednost je predvsem v tem, da so preprosti za krmiljenje. Na vsak obrat osi imajo točno določeno število korakov in krmilni sistem s signali samo sporoča, koliko korakov naj naredijo v katero smer. Zaradi tega principa ne potrebujejo povratne informacije in se jih zato uvršča pod *odprte* sisteme. Računalnik samo sporoči želeno pozicijo in izvedbe več ne preverja. Ta način delovanja je preprost, vendar ni odporen na morebitne napake (stružni nož se lahko zaleti in računalnik ne ve o tem nič), po drugi strani pa ne omogoča optimalnega izkoristka stroja. Če je na voljo povratna informacija, se obdelava lahko bolj približa mejam zmogljivosti stroja in se s tem optimizira izkoristek. Pri odprtem sistemu te informacije ni in mora zato delovati bolj previdno.

Druga generacija uporablja enosmerne motorje s krtačkami. Le-ti so vsaj trikrat hitrejši od primerljivih koračnih in premikanje je bolj gladko kot pri njih. Koračni motorji imajo namreč sunkovite gibe zaradi svoje koračne narave. Ena slabost enosmernih motorjev s krtačkami je ta, da potrebujejo dodatno napravo, ki daje krmilnemu sistemu povratno informacijo o premikih. Taki sistemi se imenujejo *zaprti* sistemi ali drugače tudi *servo* sistemi. Če se izvzame zapletenejše kontrolne algoritme, pa je to pravzaprav prednost, saj omogočajo večjo natančnost in hitrost. Ob bolj upornih delih obdelave kosa sistem omogoča motorju, da dobi več moči. Glavna slabost pa je, da so osnovani na mehanskem kontaktu – krtačkah. Le-te se počasi obrabljajo in malenkost zavirajo gibanje, poleg tega pa povzročajo iskrenje in s tem motnje v električnem omrežju.

Tretja generacija uporablja enosmerne motorje brez krtačk (*BLDC – Brushless DC electric motor*). Tako odpravljajo glavno slabost druge generacije, ta lastnost pa ima tudi negativno plat – signali za njihov nadzor so precej kompleksni, poleg tega so motorji te vrste dražji.

Omeniti je potrebno še zmedo, ki vlada na področju terminologije pri elektromotorjih. Enosmerni motorji s krtačkami so drugače poimenovani tudi DC (Direct current) servo motorji. Za enosmerne motorje brez krtačk pa se uporablja tudi

izraz AC servo motorji. Slednji izraz se zlahka zamenja z izmeničnimi motorji. Razlika je v tem, da so tako imenovani AC (Alternating current) servo motorji krmiljeni pravzaprav z enosmernim tokom.

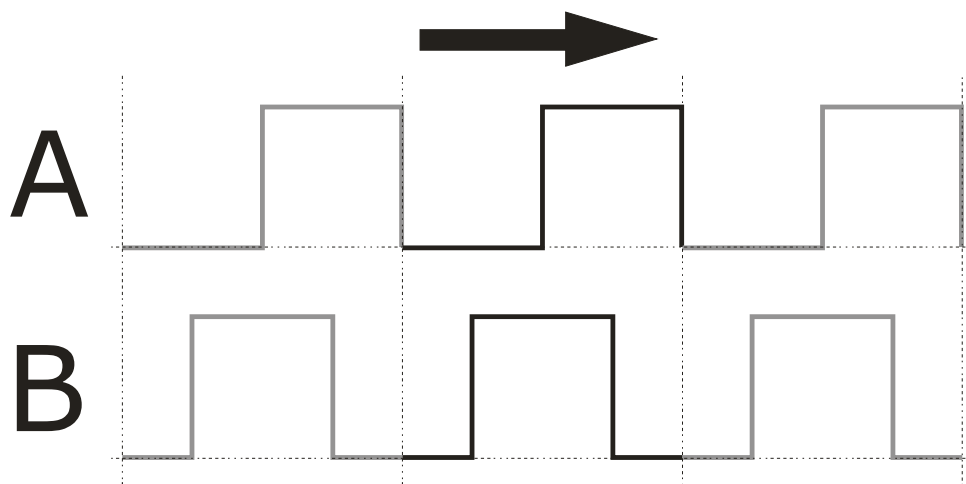
Krtačni motorji so še vedno zelo popularni predvsem na področju doma izdelanih stružnic CNC. Za serijsko proizvodnjo pa se priporočajo servo motorji, s krtačkami ali brez.

Na koncu sem sklenil kompromis in se odločil za enosmerne motorje s krtačkami. Imajo namreč vse potrebne prednosti za serijsko proizvodnjo, hkrati pa je zahtevnost krmilne logike v sprejemljivih mejah za to diplomsko delo.

Enosmerne motorje s krtačkami se krmili s PWM signalom. Dodan pa mu je še en signal, ki določa smer. Ta dodatni signal deluje na visoki ali nizki nivo in preprosto krmili preklopni rele.

2.3. Povratna informacija

Povratno informacijo o dejanskem obračanju pomičnega vretena zajemamo s pomočjo naprave, ki se imenuje *dajalec impulzov*. Napravo se pričvrsti neposredno na konec vretena in tvori tri signale: A, B in R. Signala A in B označujeta korake premikanja. Dva signala sta potrebna, da lahko določimo tudi smer premika; kombinacija spremembe namreč enolično določa smer premika. R predstavlja referenco, redundantni signal, ki služi za preverjanje pravilnosti signalov A in B ter se pojavi enkrat na obrat.



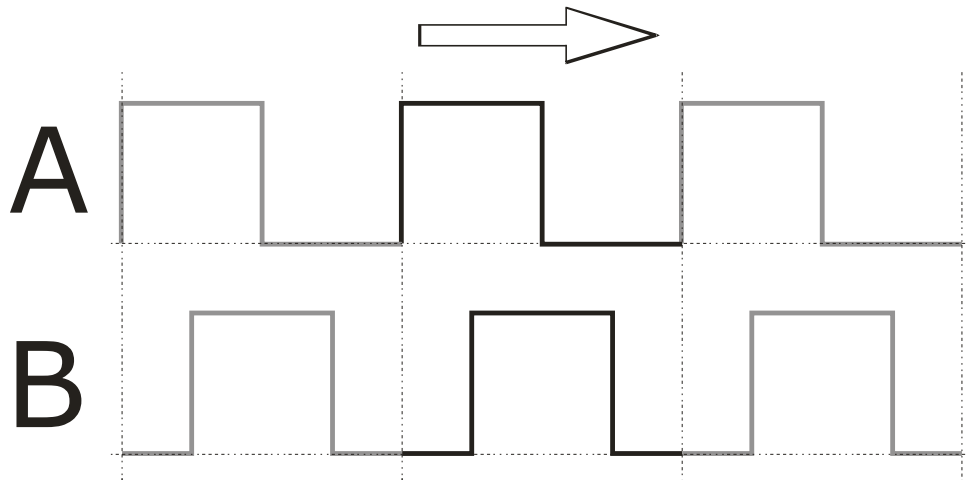
Slika 2.2: Prikaz signalov A in B, ki ju tvori dajalec impulzov. S puščico je označena pozitivna smer.

Na sliki 2.2 je prikazan značilen potek signalov A in B. Impulz predstavlja spremembo enega od signalov iz nizkega stanja v visoko (ali obratno), medtem ko drugi signal praviloma ostaja nizek ali visok. Tip impulza enolično označuje smer vrtenja vretena.

Puščica na sliki 2.2 označuje pozitivno smer obrata. Če se dajalec impulzov obrača v tej smeri, potem sprejemna naprava dobi npr. impulz, pri katerem je signal A v

nizkem stanju, B pa se spremeni in nizkega v visoko. Ostale kombinacije in več o tem je napisano v podpoglavju 5.1.2.

Če se dajalec impulzov obrača v negativni smeri, pa so impulzi ravno obratni, npr. impulz, pri katerem je signal A v nizkem stanju, B pa se spremeni iz visokega v nizko. Na sliki 2.3 je prikazan potek signalov A in B za negativno smer. Iz primerjave s sliko 2.2 je razvidno, da je slika podobna, le zrcaljena, saj so vsi prehodi ravno obratni.



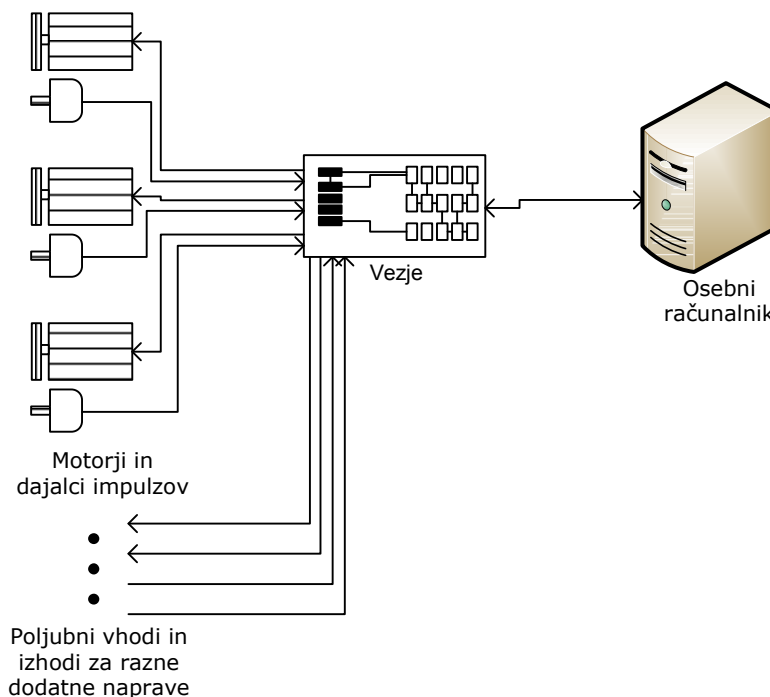
Slika 2.3: Prikaz signalov A in B, ki ju tvori dajalec impulzov. S puščico je označena negativna smer.

Vmesnik mora torej iz enega dajalca impulzov sprejemati in dekodirati tri signale: A, B in R.

3. Zamisel rešitve

V prejšnjem poglavju je bila predstavljena problematika, torej katere signale je potrebno posredno ali neposredno krmiliti z računalnikom. Večina preprostejših sistemov te signale pripelje neposredno do računalnika, ponavadi preko vzporednih vrat (ang. *parallel port*). Ta vrata sodijo med počasnejše, poleg tega pa obremenjujejo računalnik, saj se mora ta odzivati na vsako spremembo signala. To po nepotrebnem jemlje procesni čas, saj se take nizkonivojske operacije dajo izvesti tudi strojno. Poleg tega zahtevajo takojšen in stalen odziv, kar se v modernih večopravilnih operacijskih sistemih težko zagotovi za višje frekvence signalov. Več o delitvah funkcionalnosti med računalnikom samostojnim krmilnikom je podano na [4].

3.1. Shema



Slika 3.1: Shema sistema vmesnika

Vsi signali za elektromotorje in signali iz dajalcev impulzov so priključeni na posebno vezje. To opravlja funkcijo prilagajanja napetostnih nivojev, optično

izolacijo in nekatere logične funkcije. Optična izolacija se uporablja predvsem zaradi zaščite računalnika.

Poleg signalov elektromotorjev in dajalcev impulzov pa so na sliki 3.1 vidni tudi signali za razne dodatne naprave. To so razna stikala, npr. na koncu osi ali vklopi črpalk, mehanskih preklopnikov itd.

3.2. Logične funkcije vezja

Kot je bilo že ugotovljeno, ni praktično, da program na računalniku izvaja vse nizkonivojske operacije nad signali. Po drugi strani pa ni smiselno vse logike izvesti strojno, saj se izgubi kontrola in splošnost rešitve. Najboljša rešitev je torej nekje vmes.

Krmiljenje elektromotorja se lahko predstavi računalniku tako, da le-ta dobi na voljo register, kamor lahko vpiše želeno moč in smer motorja. To se lahko reši z vpisom predznačenega celega števila; torej zadostuje en register za en elektromotor. Vsem tem registrom pa se doda še enega, in sicer za vnos frekvence PWM. Tako se ohrani poln nadzor računalnika in se zato ne izgubi na splošnosti rešitve. Iz vseh teh podatkov pa lahko elektronika enolično in dokaj enostavno tvori signala za PWM in smer elektromotorja.

Pri dajalcu impulzov tudi obstaja preprosta rešitev – sistem sam šteje impulze in jih v obliki registra - števec da na voljo računalniku. Tako se računalnik razbremeni skrbi, da bi kak impulz izgubil zaradi prepoznega odziva. Hkrati pa so potrebni podatki o poziciji vedno na voljo. Potrebno je le zagotoviti, da poteka zajemanje vzorcev v strojnem delu z dovolj veliko frekvenco.

Vmesnik mora biti torej dovolj zmogljiv, da zmore:

- a) tvoriti zadovoljiv PWM signal,
- b) dovolj hitro obdelovati signale iz dajalca impulzov,
- c) imeti dovolj veliko pasovno širino za komunikacijo z računalnikom.

3.3. Želene zmogljivosti

V tem podpoglavju so poiskani in postavljeni okviri zmogljivosti parametrov, ki so bili omenjeni v prejšnjem podpoglavju.

3.3.1. Frekvenca tvorjenja signalov

Najbolj občutljiv izhodni signal je PWM signal za krmiljenje motorja. Željeno je, da je njegova frekvenca programsko nastavljiva. Kot je že omenjeno, je to doseženo s pomočjo programu dostopnega registra. Sistem naj bi omogočal frekvence signala od 20Hz do 40kHz. Temu dodamo še željo po vsaj 100 možnih različnih hitrostih. Ko se to dvoje zmnoži, se ugotovi, da mora vmesnik podpirati tvorjenje PWM signala z

vsaj 4MHz . Seveda so zadovoljive tudi manjše vrednosti, vendar bi morali zato zmanjšati želje.

3.3.2. Frekvenca zajemanja signalov

Najbolj kritični vhodni signali pa so pri dajalcih impulzov. Zanje ni dovolj samo zajemanje, ampak mora sistem omogočati tudi dovolj hitro obdelavo. Zaradi tega pri npr. mikrokrmilnikih ni dovolj, da se gleda samo frekvenco procesorja, ampak je potrebno upoštevati celoten čas obdelave.

Za izračun maksimalne frekvence impulzov ($f[\text{Hz}]$) je potrebno pridobiti maksimalno hitrost pomika suporta ($v[\text{m/min}]$) in število tvorjenih impulzov na dolžino pomika ($n[\text{m}^{-1}]$). Slednje se izračuna z deljenjem pomika suporta na obrat vretena ($s[\text{mm}]$) in števila impulzov na en obrat dajalca impulzov ($m[\text{I}]$). Število impulzov na obrat pa se dobi kot produkt števila period na obrat ($p[\text{I}]$) in impulzov na periodo ($r[\text{I}]$). Po Nyquistu je potrebno dobljeno frekvenco impulzov pomnožiti z 2, da se dobi potrebna frekvenca vzorčenja. Vse to je povzeto v formuli (3.1).

$$\begin{aligned} f &= 2 \times v \times n \\ n &= \frac{m}{s} \\ m &= p \times r \end{aligned} \tag{3.1}$$

Iz slike 2.2 je razvidno, da se signala na eno periodo spremenita štirikrat, torej $r=4$. Najbolj natančni dajalci impulzov na trgu imajo $p=2500$. Vretena imajo standardne navoje s $s=2,5\text{mm}$. Hitrosti suportov pri najnovejših stružnicah CNC dosegajo tudi do 30m/min , predelane starejše stružnice pa maksimalno $v=10\text{m/min}$.

Rezultat preračuna je enak $f=1,3\text{MHz}$.

Žal se je po izdelavi vezja izkazalo, da uporabljeni ceneni opto sklopniki (ang. *optocoupler*) ne prepuščajo signalov z večjimi frekvencami od 50kHz . Ob upoštevanju dejstva, da gre za dva signala (A in B), to omogoča frekvenco impulzov 100kHz . K sreči tudi mehanični del pri predelanih stružnicah nima visoke natančnosti in so dajalci impulzov s $p=250$ čisto zadovoljivi. Tako se pride na frekvenco impulzov 67kHz , kar znese $f=133\text{kHz}$.

3.3.3. Pasovna širina komunikacije

Iz prakse sledi, da bi bilo dobro vsako sekundo izvršiti vsaj 200 branj in popravkov krmiljenja. Če se temu doda še število podatkov, ki se ob tem prenese, se lahko izračuna minimalno potrebno pasovno širino.

Ob enem branju/popravljanju mora biti prebran časovnik (več o njem v podpoglavju 5.3.3.) in vsi trije števcji, ter biti nazaj krmiljeni trije motorji. Poleg tega je možno še branje in pisanje prostih vhodov/izhodov ter preverjanje napak. To nanese interakcijo z 10 registri. Vsi registri so 32-bitni, torej to pomeni 320b na interakcijo. Če se to pomnoži z 200Hz , se dobi minimalno potrebno pasovno širino 64kbit/s .

4. Izbira tehnologije

Na sliki 3.1 je predstavljena zamisel sheme sistema. Za samo realizacijo pa je na voljo več tehnologij. Osebni računalnik je zaključen in precej dobro definiran sistem, zato se vanj ni potrebno poglobljati. Prav tako sta določena tudi elektromotor in dajalec impulzov. Odprta ostajata še tip komunikacije vmesnika z računalnikom in sam način realizacije vmesnika. V naslednjih podpoglavjih so predstavljene nekatere bolj izstopajoče od tehnologij, ki so na voljo.

4.1. Komunikacija z osebnim računalnikom

4.1.1. IEEE 1284

Znan je po uporabi na vzporednih vratih (ang. *parallel port*) pri osebnih računalnikih. Je najbolj pogost vmesnik za komunikacijo pri doma izdelanih strožnicah CNC, saj je njegova uporaba za izhodno komunikacijo zelo preprosta. Program lahko preprosto piše podatke na določeno lokacijo v vhodno/izhodnem pomnilniškem prostoru in se preslikajo neposredno na vmesnik. Od tu tudi izvira njegova popularnost za doma izdelane strožnice – večinoma uporabljajo koračne motorje, ki potrebujejo enosmerno krmiljenje in ne pošiljajo nobenih povratnih informacij. Za serijsko proizvodno pa se bolj priporočajo servo motorji, ki obvezno potrebujejo povratno informacijo. Če hočemo protokol IEEE 1284 uporabiti za vhod, se stvari zapletejo. Kompleksnejši postane vmesnik na zunanji strani in tudi sam program na računalniku, ki te podatke obdeluje.

Glede pasovne širine ta protokol ne predstavlja težav. Najpočasnejši načini delovanja (npr. *Centronics* način) tega protokola omogočajo 200KB/s, kar je več kot dovolj za potrebe tega vmesnika.

4.1.2. RS-232

Zelo razširjen vmesnik, ki ga najdemo na zaporednih vratih (ang. *serial port*) pri osebnih računalnikih. Je zelo pogost protokol za komunikacijo z raznimi doma izdelanimi zunanji napravami.

RS-232, za razliko od IEEE 1284, omogoča polno dvosmerno komunikacijo (ang. *full duplex*) in je tako za ta projekt bolj primeren. Poleg tega je njegova uporaba široko podprta s pomočjo posebnih UART (*Universal asynchronous receiver/transmitter*) modulov in zato zelo preprosta. Veliko mikrokrmilnikov med posebnimi funkcijami vključuje tudi podporo za ta protokol.

Pasovna širina v praksi znaša do 115Kbit/s, kar nekako zadošča potrebam. Ni pa ravno spodbudno v primeru potreb po pogostejših popravkih ali v primeru kakih kasnejših razširitev vmesnika.

4.1.3. USB

Danes je to najbolj razširjen protokol za komunikacijo z zunanjimi napravami, vendar njegova uporaba žal ni tako preprosta. Potrebno je poskrbeti za elektroniko, ki zna komunicirati po tem protokolu, poleg tega pa je treba priskrbeti še gonilnike za želeni operacijski sistem. Vsaka od teh nalog bi zadostovala za samostojno diplomsko delo.

Obstaja pa druga možnost, ki uporabo precej poenostavi – FTDI čip. Ta izkoristi USB komunikacijo in se računalniku predstavi kot zaporedna vrata. Tako se izogne potrebi po dodatnem gonilniku. Ostale naprave z njim komunicirajo tako, da vanj pišejo ali iz njega berejo podatke. Tako je vsa zapletena strojna logika realizirana v enem čipu. Slabost tega pristopa je le, da je pasovna širina potem enaka kot pri protokolu RS-232.

4.1.4. PCI

PCI (*Peripheral component interconnect*) standard definira računalniško vodilo za priklop perifernih naprav. Za razliko od prvih treh protokolov se tu naprave skrijejo neposredno v računalniško ohišje in se ne priklapljajo nanj s kablom. Standard temelji na polni dvosmerni komunikaciji in že v osnovi omogoča 133MB/s, kar daleč presega vse morebitne potrebe vmesnika.

Protokol ni ravno pogosto uporabljen v izdelavi domačih naprav, ker naj bi bil precej zapleten za uporabo. Specifikacije, ki so podane v [5], spadajo med res obsežne dokumente, saj pokrivajo vse možne načine uporabe. V našem primeru ni potrebna polna izvedba PCI protokola, ampak le branje in pisanje. Slednje dvojje pa niti ni tako zapleteno.

Napravo PCI lahko izvedemo na vsaj dva možna načina; tako, da:

- a) implementiramo polno funkcionalnost naprave, da se predstavi operacijskemu sistemu in omogoči tudi uporabo gonilnika,
- b) implementiramo samo branje in pisanje, ki se odzivata na fiksne naslove.

Prvi način je sicer bolj zaželen, a je tudi precej bolj zapleten. Poleg tega je potreben še gonilnik, da se naprava sploh lahko uporabi. Pri drugem načinu se v operacijskem sistemu najprej ogleda tabelo naprav v vhodno/izhodnem pomnilniku, potem poišče prazne naslove, nato pa PCI napravo izvede tako, da se na te fiksne naslove odziva. Več o tem načinu je podano v [6].

PCI signali

Protokol definira veliko signalov, vendar v osnovni izvedbi niso vsi potrebni. V tabeli 4.1 so navedeni potrebni signali za izvedbo preproste ciljne (ang. *target*) naprave. V prvem stolpcu se nahaja ime signala in v oglatem oklepaju informacija,

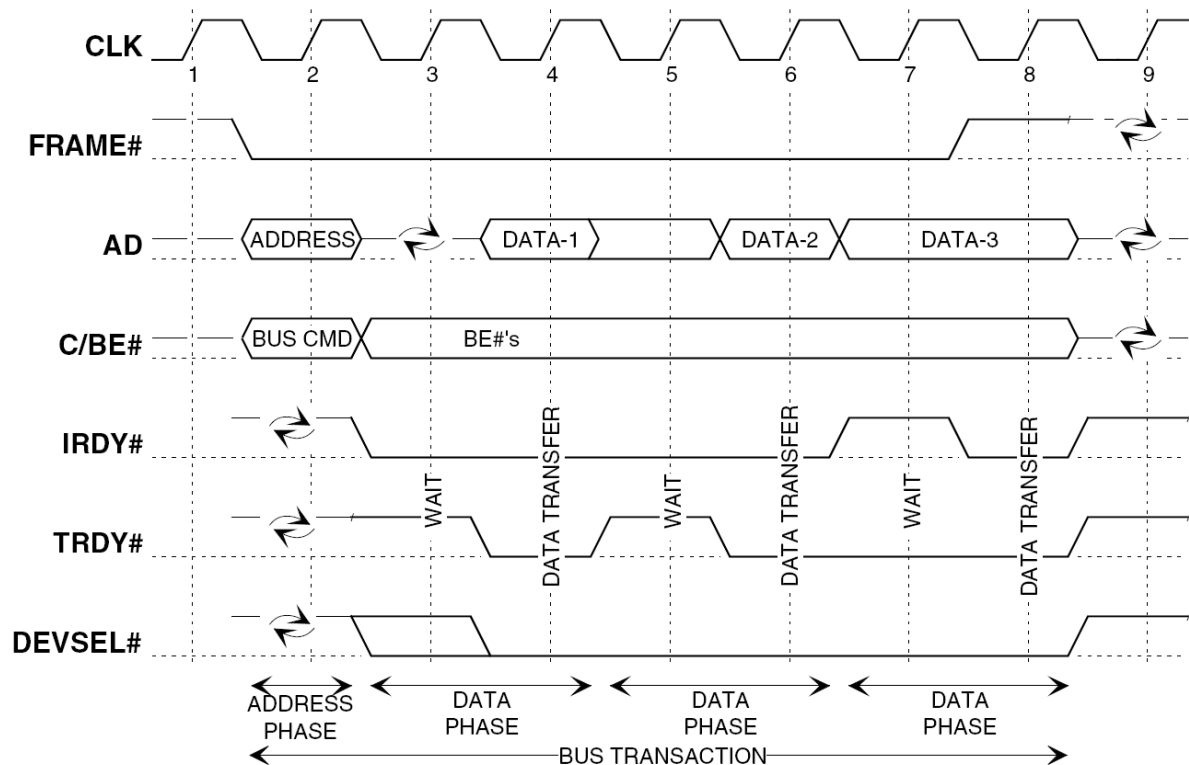
koliko linij potrebuje. V drugem stolpcu se nahaja smer signala glede na ciljno napravo. »V« predstavlja vhodni signal in »I« izhodni. »V/I« pomeni, da je signal dvosmeren. Smeri so zaradi poenostavitve prilagojene ciljni napravi in niso tako splošne, kot po standardu. Simbol »#« pa pomeni, da je signal definiran v negativni logiki. V tretjem stolpcu sledi opis signala.

<i>Signal</i>	<i>Smer</i>	<i>Opis</i>
CLK	V	Ura.
RST#	V	<i>Reset</i> signal je uporabljen, da vse registre in signale, povezane s PCI, postavi v začetno stanje.
AD[31..0]	V/I	Naslovi in podatki (ang. <i>address and data</i>) so multipleksirani na istih linijah. Transakcija na vodilu se prične z naslovno fazo in nadaljuje z eno ali več podatkovnih faz.
C/BE#[3..0]	V	Ukazi in izbira posameznih bajtov (ang. <i>Bus Command and Byte Enables</i>) so prav tako multipleksirani na istih linijah. V naslovni fazi so na vodilu prisotni ukazi, v podatkovnih fazah pa izbire bajtov. Ukaz določa tip prenosa – branje ali pisanje, vhodno/izhodno ali pomnilniško ali konfiguracijsko.
PAR	I	Soda pariteta. Redundantni signal, ki poveča zanesljivost prenosa. V praksi ga večina implementacij PCI ignorira.
FRAME#	V	Določa ga trenutni gospodar vodila in določa začetek ter trajanje prenosa.
IRDY#	V	Pobudnik pripravljen (ang. <i>initiator ready</i>) označuje gospodarjevo pripravljenost, da zaključi trenutno fazo prenosa.
TRDY#	I	Cilj pripravljen (ang. <i>target ready</i>) označuje pripravljenost ciljne naprave, da zaključi trenutno fazo prenosa.
DEVSEL#	I	Izbor naprave (ang. <i>device select</i>) sporoča, da je ciljna naprava zaznala svoj naslov za trenutni prenos in da se nanj odziva.
STOP#	I	Stop lahko sproži ciljna naprava, če želi ustaviti trenutni prenos.
IDSEL	V	Začetni izbor naprave (ang. <i>initialization device select</i>) je edini signal, ki ni skupen vsem napravam na vodilu, ampak ima vsaka reža svojega. Uporabljen je za konfiguriranje.
PERR	I	Paritetna napaka (ang. <i>parity error</i>). Za poročanje napak.
SERR	I	Sistemska napaka (ang. <i>system error</i>).

Tabela 4.1: Signali na PCI vodilu, potrebni za izvedbo ciljne naprave.

Bralni prenos

Bralni prenos je prikazan na sliki 4.1.

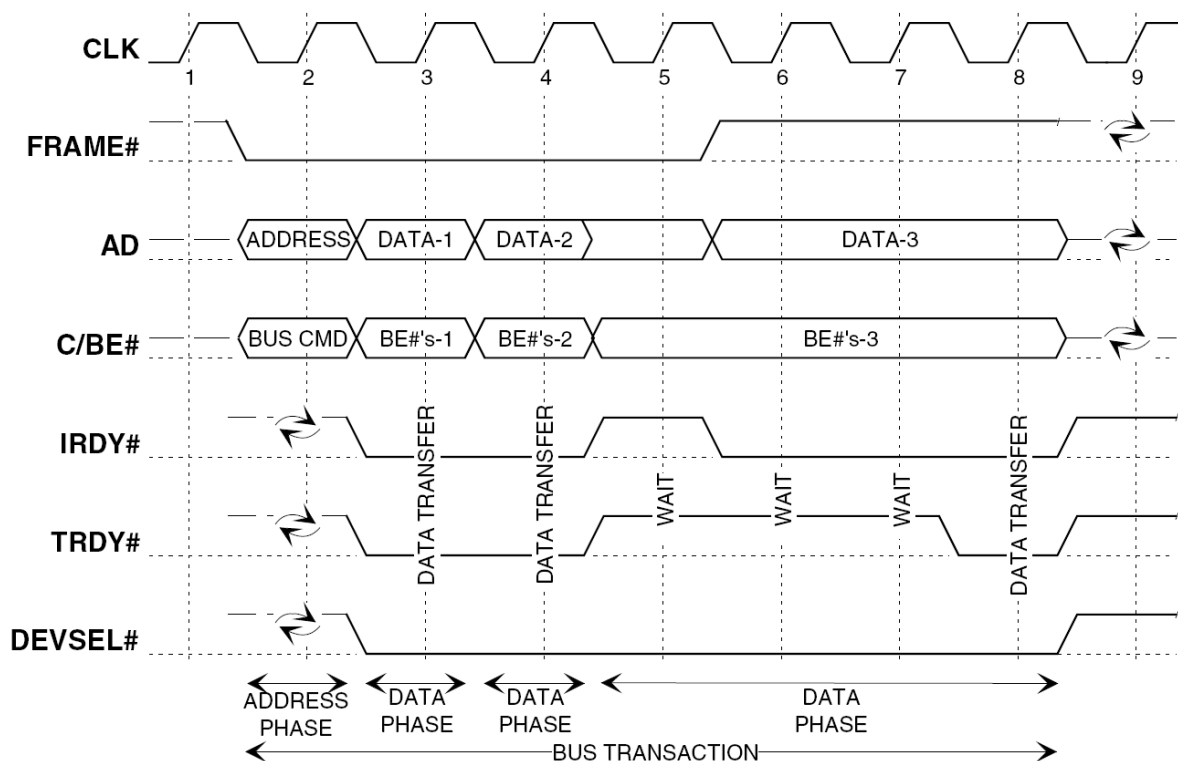


Slika 4.1: Potek signalov pri bralnem prenosu na PCI vodilu

Gospodar najprej sproži FRAME# in določi AD ter C/BE#. Naslednjo urino periodo aktivira IRDY#, nato pa čaka ciljno napravo. Ciljna naprava se drugo ali tretjo periodo odzove s signalom DEVSEL#, v primeru, da zazna svoj naslov. Ko podatke dostavi na AD, postavi signal TRDY# in ob tisti periodi gospodar prebere podatke. Faza branja se lahko ponavlja, dokler gospodar ne umakne signala FRAME#.

Pisalni prenos

Na sliki 4.2 pa je prikazan pisalni prenos.



Slika 4.2: Potek signalov pri pisalnem prenosu na PCI vodilu

Poteka podobno kot branje. Gospodar vodila najprej aktivira FRAME#, določi AD in C/BE#. Naslednjo urino periodo na AD že postavi veljavne podatke ter aktivira IRDY#. Če ciljna naprava zazna svoj naslov, postavi DEVSEL# in lahko še isto urino periodo podatke prebere, kar označi s TRDY#.

4.2. Realizacija logike

4.2.1. Prilagojeno vezje

Najbolj osnovna možnost je, da se pregleda vse možne kataloge, nabere potrebne čipe in na koncu skupaj sestavi vezje, ki opravlja želeno funkcijo. Obstaja mnogo čipov, ki opravljajo nekatere specializirane funkcije, ki jih potrebuje ta vmesnik, a jih je hkrati težko najti in so poleg tega še precej dragi.

Prednost je, da bi bil sistem robusten, slabosti pa sta, da bi iskanje čipov dolgo trajalo in nefleksibilnost izdelanega sistema. Če bi ga hoteli kakorkoli spremeniti, bi bilo

potrebno spremeniti vezje, oziroma narediti novega. To je zelo nedobrodošlo pri sistemih v razvoju.

4.2.2. Mikrokrmilnik

Mikrokrmilnik kot računalnik na čipu je gotovo bolj fleksibilen. Vsebuje procesor, pomnilnik in V/I funkcije. Pomnilnik je ponavadi razdeljen na bralno-pisalni pomnilnik za shranjevanje podatkov, bralni pomnilnik za program in EEPROM za permanentno shranjevanje podatkov. Ponavadi se jih preprosto programira s posebnim kablom, lahko tudi v vezju. Slednje pride zelo prav pri sistemih v razvoju. Mnogi imajo v svojem naboru V/I funkcij tudi PWM in UART, kar doda k njihovi vrednosti.

Pozornost je treba posvetiti le še temu, da program teče zaporedno. Zato bi bilo potrebno izbrati dovolj hiter model mikrokrmilnika, ki bi zmogel zagotoviti sprotno obdelavo signalov in hkrati komunikacijo z računalnikom.

4.2.3. DSP

Na prvi pogled se najbrž vsak vpraša, kakšno povezavo s strožnico ima lahko DSP (*Digital signal processor*), specializiran procesor za digitalno procesiranje signalov. Vendar se od devetdesetih naprej vztrajno prebijajo tudi na tem področju. Kajti tradicionalno procesiranje signalov in teorija, ki sloni za krmiljenjem strojev CNC, imata kar nekaj skupnega. Več o tem je podano na [7]. Matematika je enaka, le z drugačnim pogledom. Poudarki se razlikujejo, ampak oboje neizogibno vključuje časovno komponento.

Področje je videti obetavno, vendar z njim nimam veliko izkušenj. Signalni procesorji zahtevajo drugačen način programiranja, katerega učenje ni cilj tega diplomskega dela.

4.2.4. FPGA

FPGA (*Field-programmable gate array*) je polprevodniška naprava, ki vsebuje programabilne logične komponente, imenovane *logični bloki*, ter programabilne povezave med njimi. Logični bloki so lahko programirani za izvajanje funkcij osnovnih logičnih vrat, kot so *IN* ali *ekskluzivni ALI*, ali zapletenejših kombinatoričnih funkcij, npr. dekodeerjev ali matematičnih funkcij. Logični bloki pri večini FPGA-jev vključujejo tudi pomnilne elemente, od preprostih pomnilnih celic do več blokov pomnilnika. Hierarhija programabilnih povezav omogoča poljubno povezovanje logičnih blokov, kakor potrebuje razvijalec sistema.

FPGA, kot najbolj primeren predstavnik programabilnih logičnih naprav (ang. *PLD - Programmable logic device*), gotovo spada med najbolj fleksibilne rešitve. Lahko ga sprogramiramo, da opravlja skoraj poljubne kombinatorične funkcije, vsebuje pa tudi pomnilne elemente. Njihova prednost torej je, da lahko na enem čipu realiziramo poljubno digitalno vezje. V nekem HDL (*Hardware description language*) jeziku se opiše želena funkcionalnost vezja (torej programira), nato pa se program naloži na FPGA čip, kjer postane logično vezje. Prednost tega je, da se program, torej vezje,

lahko razvija in popravlja kar med delovanjem. Kot sem že omenil, je to zelo dobrodošlo pri sistemih v razvoju.

Kakšna pa je razlika med mikrokrmilniki in FPGA tehnologijo? Na videz je podobno. Oboje je na koncu realizirano v nekem čipu s končno mnogo nožicami, kamor priklopimo želene signale. Vsebino obojega sprogramiramo in naložimo na čip, ki teče z neko frekvenco. Vendar obstaja bistvena razlika. Na mikrokrmilniku je (ponavadi) en procesor, na katerem teče en proces. Problemov se lahko loteva zaporedno, torej enega na enkrat. Ne more v istem trenutku šteti impulzov in komunicirati z računalnikom (razen z uporabo časovnega deljenja). Pri FPGA tehnologiji pa se vsi procesi izvajajo hkrati, vzporedno. To je tudi eden najtežjih preskokov v miselnosti, ko se človek uči HDL jezika. Štetje impulzov poteka ločeno od tvorjenja PWM signala in ločeno od komuniciranja z računalnikom. Vse poteka hkrati in nemoteno drug od drugega, kar je idealno za želeni krmilni sistem.

Ta rešitev ima tudi nekaj negativnih strani – cena je bistveno višja, kot pri primerljivih mikrokrmilnikih, pa tudi uporaba čipa v vezju ni tako enostavna. Potrebuje zunanjo uro, ob izgubi napajanja izgubi vsebino, praktično nemogoče je čipe dobit v DIP (*Dual in-line package*) ohišju itd. Vse slednje stvari so rešljive, vendar zahtevajo določeno znanje, ki zahteva določen čas.

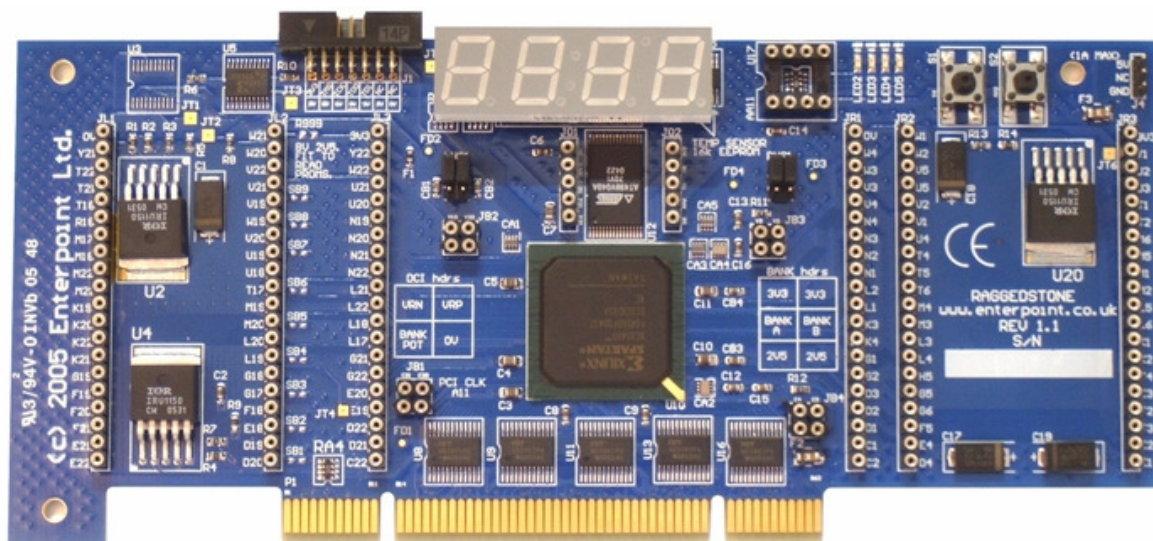
Več o FPGA tehnologiji v [6].

4.3. Izbrana kombinacija

Ob iskanju primerne rešitve in tehtanju zgornjih argumentov sem naletel na razvojno ploščo Raggedstone1 iz podjetja Enterpoint Ltd. Njeno srce predstavlja Xilinx-ov Spartan-3 FPGA čip in nanj ima povezan vmesnik za protokol PCI, poleg tega pa še večjo množico priključkov za signale iz zunanjih naprav. Združuje torej obe zelo primerni tehnologiji za rešitev problema tega diplomskega dela. Na plošči ima tudi vse potrebno za nemoteno delovanje FPGA čipa in s tem rešuje problematiko te tehnologije. Poleg tega ima za vso svojo funkcionalnost kar sprejemljivo ceno.

Potrebno je le še implementirati funkcionalnost ter poskrbeti za optično izolacijo in pravi priklop zunanjih signalov. Raggedstone1 žal ne vsebuje kakršnegakoli logičnega dela PCI komunikacije, ampak samo fizični vmesnik in priklop signalov na FPGA čip. Zato je potrebno sprogramirati in na FPGA čipu izvesti tudi ciljno PCI napravo. Slednje ni bilo upoštevano za večji problem, saj obstaja nekaj preprostih delujočih zgledov, a se je kasneje izkazalo, da ni tako preprosto.

4.3.1. Razvojna plošča Raggedstone1



Slika 4.3: Slika razvojne plošče Raggedstone1

Razvojna plošča Raggedstone1 vsebuje vse potrebno za izvedbo naprave na PCI vodilu. Vsebuje priključek, ki se vstavi v režo na PCI vodilu in vsebuje vse potrebne napetostne prilagoditve, da se lahko signale poveže na Spartan-3 FPGA čip. Podpira 32b in 33MHz izvedbo PCI protokola. Plošča se tudi napaja iz vodila, podpira pa obe izvedbi – 5V in novejšo 3,3V. Prav tako se iz PCI vodila uporablja 33MHz ura, ki poganja Spartan-3 čip.

Raggedstone1 vsebuje tudi JTAG (*Joint test action group*) priključek, preko katerega se FPGA čip programira. Priloženi kabel se preprosto poveže z računalnikom, ki ima potrebno programsko opremo, in vse je nared za razvoj. Prav tako vsebuje FLASH pomnilnik, ki hrani načrt programabilnega vezja, ko je to izključeno. Ob vklopu se načrt preprosto naloži iz FLASH pomnilnika.

Na levi in desni strani ima veliko množico priključkov v 2,54mm mreži za razširitvene module, kar je lahko uporabljeno za priklop elektromotorjev, števecv in ostalih naprav. Plošča podpira signale v napetostih do 2,5V ali 3,3V, odvisno od nastavitev.

Ima tudi množico dodatnih naprav: 4 LED diode, 4-mestni 7-segmentni LED prikazovalnik, 2 stikali, FLASH in EEPROM pomnilnik, temperaturni senzor itd. V sklopu razvoja sem uporabljal predvsem stikali in LED elemente za sporočanje stanja v sistemu.

Več o tem je podano v [8].

5. Realizacija

Prvi opravljeni korak je bila komunikacija z računalnikom po protokolu PCI. Priskrbeti je bilo potrebno ciljni računalnik, se odločiti za operacijski sistem in začeti poskušati. Nekaj težav je bilo z malenkost različnimi izvedbami protokola – nekatere variante podpirajo 3,3V napajalno napetost, nekatere ne – ampak je stvar stekla. Močno se je zapletlo pri samem logičnem jedru protokola, ampak več o tem v naslednjih podpoglavjih.

Ko je PCI komunikacija nekako delovala, se je delo nadaljevalo na razvoju logična dela komunikacije z dajalcem impulzov in elektromotorjem. Optična izolacija oziroma deli vezja, ki so bili potrebni, so bili v tej fazi narejeni kar na prototipni plošči (ang. *proto-board*). Tu se je pričelo tudi izbiranje pravih elektronskih elementov, kar vzame kar nekaj časa. Pri dajalcu impulzov pa je prihajalo do nekih čudnih pojavov, zaradi katerih sem porabil kar nekaj dni, na koncu pa se je izkazalo, da je bil problem čisto banalen: sinhronizacija vhodnih signalov v vezje namreč ni bila pravilno opravljena. Lekcija naučena.

V tretji fazi tega diplomskega dela pa je bil narejen dodatni modul z vezjem, ki vsebuje potrebno elektroniko. Zadeva je izdelana na posebni prototipni plošči, kjer se vse potrebne povezave naredi z žičkami. Ker je teh povezav ogromno, je tudi ta izdelava vzela kar nekaj dni.

To poglavje je razdeljeno na nekako tri enote – prva predstavlja logični del, ki se je razvijal na računalniku, druga predstavlja strojni del, tretja pa je oblikovana kot uporabniški priročnik, ki predstavi pogled uporabnika na vmesnik in demonstrira njegovo uporabo.

5.1. Kodiranje

S HDL smo na fakulteti tekom študija delali pri kar nekaj predmetih, kjer smo vedno uporabljali jezik VHDL (*VHSIC hardware description language*). Poleg tega je jezik precej razširjen tudi na spletu in se zanj najde veliko pomoči in primerov uporabe, zato sem se tudi jaz odločil zanj. Vsi moduli, ki so predstavljeni v naslednjih podpoglavjih, so realizirani v njem.

V samo sintakso in predstavitev jezika se v tem delu ne bi poglobljal in zato tudi ni dodanih primerov z izvorno kodo. Bom pa vsekakor poskušal funkcionalnost logično kar najbolj predstaviti in približati. Kdor jezik VHDL zna, same kode ne potrebuje, ker izhaja implementacija precej neposredno iz definicije. Kdor ga ne zna, se ga tudi v tem kratkem diplomskem delu ne bo naučil.

Poudariti je še potrebno, da so vsi sledeči moduli v tem podpoglavju napisani na računalniku in realizirani znotraj Spartan-3 FPGA čipa. Torej NE gre za realizacijo z več različnimi čipi. Naj logične sheme bralca ne zmedejo.

5.1.1. Komunikacija PCI

Protokol PCI sprva zveni zelo strašljivo. Ti strahovi hitro zginejo, ko si človek enkrat ogleda specifikacije, signale, njihove prehode in potrebne funkcije za delovanje branja in pisanja. Ko pa se dejansko loti implementiranja naprave, kaj hitro ugotovi, da stvari le niso tako enostavne. Pri samem diplomskem delu je bilo največ časa porabljenega ravno za izvedbo komunikacije po tem protokolu.

V osnovi sta dve možnosti za izvedbo:

- a) uporaba že narejenega odprtega *logičnega jedra*¹ pci32tlite (glej [9]),
- b) lastna izvedba branja/pisanja na fiksne naslove.

Možnost a) je problematična, ker je potrebno imeti gonilnik za želeni operacijski sistem. Napisati le-tega pa spet ni trivialna naloga, zato sem se odločil za možnost b).

Poskus lastne izvedbe

Začela se je z implementacijo funkcije pisanja. Med testiranjem pa se je izkazalo, da se računalnik ne odziva. Prižgal se je, vendar dlje kot do začetnega okna sistema ni prišlo. Kartica je bila vstavljena v drug računalnik in deloval je normalno, kartica pa ne. Sodeč po LED prikazovalniku, ki je obstal na eni cifri, je matična plošča PCI reži odvezla urin signal ali pa kako drugače zablokirala delovanje. V obeh možnostih PCI komunikacija ni delovala.

Poskušano je bilo dodajanje še drugih funkcionalnosti v upanju, da se bo kaj spremenilo, a zaman. Na tej točki sem obupal in se podal v možnost a).

Poskus uporabe logičnega jedra pci32tlite

Na spletnih straneh [9] sem dobil logično jedro pci32tlite, ki vsebuje vso potrebno logiko za komunikacijo preko PCI protokola. VHDL kodo je bilo potrebno vstaviti v obstoječi projekt in pravilno povezati vse signale, povezane s PCI. Ko je bilo to opravljeno in je bil zagnan računalnik, se je izkazalo, da zadeva deluje. Operacijski sistem je celo zaznal novo napravo, tu pa je tudi ostalo. Dostopanje do naprave brez gonilnika je nemogoče, saj napravi ni dodeljen niti naslovni prostor. Kaj narediti? Naprava je zaznana, to je bilo opogumljajoče. Ker je manjkal samo še gonilnik, sem se poskusil lotil tega.

Področju gonilnikov je precej novo zame, zato se je bilo potrebno najprej malo razgledati po njem. Izkazalo se je, da za operacijski sistem Windows obstajata dve možnosti. Prva so draga profesionalna orodja, namenjena samo PCI gonilnikom, s

¹ Logično jedro predstavlja funkcionalno zaokrožen modul, običajno v VHDL jeziku, ki na FPGA čipu postane logično vezje.

pomočjo katerih je gonilnike precej enostavno narediti. Druga možnost pa je uporaba specializiranih knjižnic in programiranje z orodjem Visual Studio v programskem jeziku C++. Prva možnost je predraga, druga pa bi vzela preveč časa, saj s tem orodjem in jezikom še nisem imel izkušenj. Presedlal sem na operacijski sistem Linux, kjer se stvari piše enostavno v programskem jeziku C, s katerim pa imam že kar nekaj izkušenj.

Za lažji začetek je bil uporabljen nek že narejen gonilnik, ki je bil precej blizu temu, kar bi bilo potrebno za ta sistem. Gonilnik bi bilo potrebno najprej prevesti, da bi se ga lahko namestilo v operacijski sistem, vendar se je ustavilo že tu. Linux pozna namreč precej različnih jeder in vsako ima svoje specifične knjižnice. Poleg tega se dosti stvari spreminja tudi med verzijami. Precej časa je bilo porabljeno pri ukvarjanju s temi težavami in nekaj jih je uspelo celo rešiti, vendar se je stvar vlekla v nedogled. Ko je bil kak konflikt rešen, se je pokazalo par novih. Zato sem na tej točki odnehal. Pisanje gonilnikov žal zahteva precej temeljito poznavanje nekega programskega jezika in operacijskega sistema.

Analiza stanja

Opravljen je bila primerjava obeh rešitev. Zakaj se logičnemu jedru pci32tlite računalnik odziva, lastni izvedbi pa ne? Hotel sem poiskati to razliko in jo vstaviti v svojo rešitev. Sumil sem, da jedro pravilno upošteva nekatere od signalov, ki jih jaz ne. Odločil sem se torej jedru odklapljati kontrolne signale, drugega za drugim in preverjati, do katere meje je kartica še prepoznavna v sistemu.

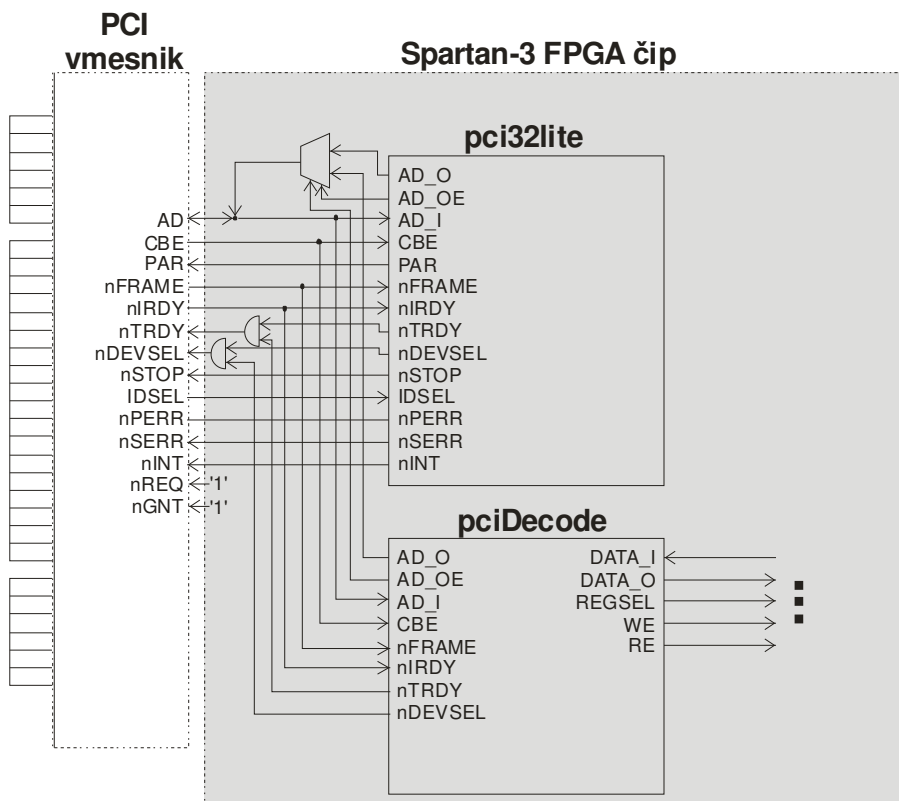
Ko je bil odklopljen signal IDSEL, se je računalnik začel obnašati enako kot ob uporabi lastne izvedbe. Ob nadaljnji analizi je bilo ugotovljeno, da je ta signal lasten za vsako režo na PCI vodilu in, da je uporabljen za začetno konfiguracijo kartice. Poskusil sem vpeljati njegovo uporabo, a žal neuspešno. Ta signal očitno sproža še nekatera druga branja in pisanja v specialne registre. Izkazalo se je torej, da BIOS (Basic input/output system) uporablja signal IDSEL, da opravi kompleksnejše konfiguracije in preverjanje. Če ugotovi, da se naprava ne odziva pravilno, ji odklopi krmilne signale. V primeru starejših računalnikov pa BIOS očitno obstane pri preverjanju in računalnik »zmrzne«.

Rešitev

Pri lastni izvedbi je mogoče branje/pisanje, vendar sistem ne deluje. Pri jedru pci32tlite pa sistem deluje, vendar branje/pisanje ni mogoče. Rodila se je zamisel, da bi združil obe opciji. Tehnično gledano je to prilagoditev pci32tlite jedra, v praksi pa stvar zgleda tako, da se na iste PCI signale priklopi obe rešitvi – logično jedro pci32tlite in zraven vzporedno še lastno izvedbo branja in pisanja, ki se odziva na fiksne naslove. Izkazalo se je, da rešitev deluje!

Jedro pci32tlite reši zaplete ob zagonu in komunikacijo z BIOS, lastna izvedba pa omogoča preprosto branje in pisanje. Naprava se tako pojavi na fiksnih naslovih v vhodno/izhodnem pomnilniškem prostoru. Prednost te rešitve je, da se lahko uporabi na kateremkoli operacijskem sistemu, saj ne potrebuje dodatnega gonilnika. Slabost

je le, da je potrebno pred namestitvijo preveriti, da niso isti naslovi v uporabi kake druge naprave.



Slika 5.1: Prikaz logike, povezane s PCI, znotraj Spartan-3 FPGA čipa. Priključitev obeh logičnih jeder na PCI vodilo in lepilna logika.

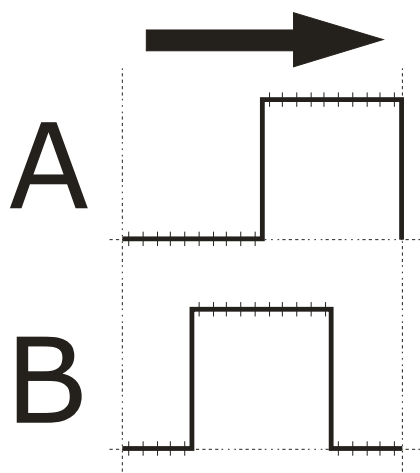
Na sliki 5.1 je prikazan priključitev obeh jeder na PCI vmesnik. Iz slike je lepo razvidno, da pci32lite poskrbi za vse kontrolne signale, lastna implementacija, »pciDecode«, pa uporablja samo tiste, ki so potrebni za branje in pisanje. Prav tako je razvidno, da ima pciDecode na drugi strani tudi precej preprostejše signale za interno komunikacijo z registri.

Logično jedro pciDecode mora pravilno sprejemati kontrolne signale CBE, nFRAME in nIRDY. Nanje se odziva s kontrolnima signaloma nTRDY in nDEVSEL. Ker jedro pci32lite prav tako uporablja tudi ta dva signala, je pred izhodom na PCI za vsakega od teh dveh signalov logični element IN, ki signala iz obeh jeder združi. Ker gre za signale v negativni logiki, to pravzaprav predstavlja logični ALI. Torej spusti signal na PCI vodilo, če ga aktivira katerikoli od obeh jeder.

Podobno je tudi pri signalu AD. Bereta ga lahko obe jedri, za pisanje nazaj pa morata aktivirati signal AD_OE. Multiplekser na zgornji sliki tako odloča, kateri AD_O bo spustil na AD vodilo. Če AD_OE slučajno sprožita obe jedri hkrati, ima prednost pciDecode jedro.

V samo logiko za tvorjenje kontrolnih signalov se v tem delu ne bom poglobljal, ker je malce bolj obsežna, poleg tega pa ni ključna za razumevanje celotnega problema tega diplomskega dela. Če bralec želi, si jo lahko ogleda v [6].

5.1.2. Štetje impulzov



Slika 5.2: Prikaz ene periode signalov A in B, z označenim vzorčenjem in puščico, ki predstavlja pozitivno smer

Na sliki 5.2 je še enkrat podano, kakšen je potek signalov iz dajalca impulzov za eno periodo. Posamezen impulz pomeni spremembo signala A ali B iz nizkega stanja v visoko ali obratno. Glede na tip spremembe je enolično določena smer obrata. V tabelah 5.1 in 5.2 so predstavljeni vsi možni impulzi.

A	–	↑	–	↓
B	↑	–	↓	–

Tabela 5.1: Možni impulzi za pozitivno smer

A	–	↓	–	↑
B	↓	–	↑	–

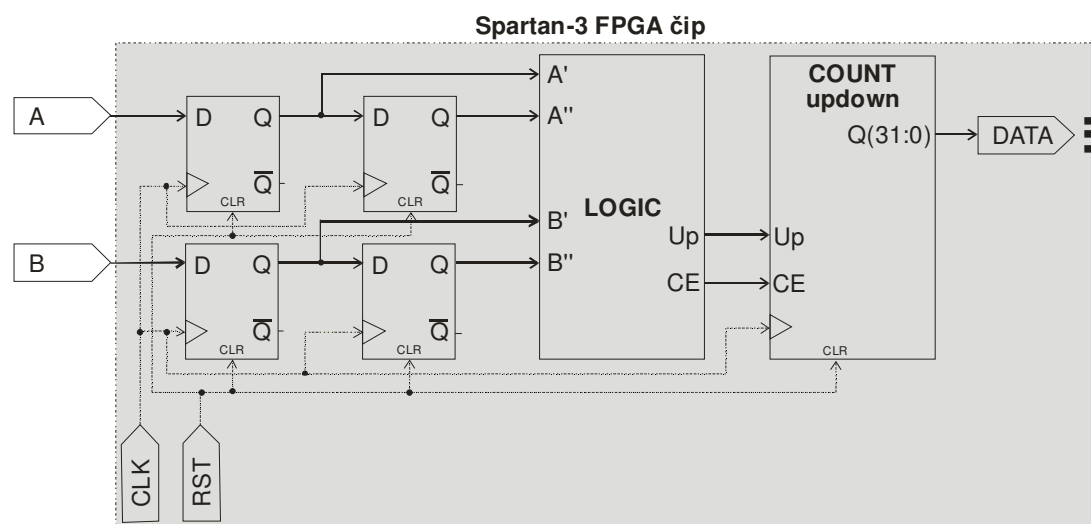
Tabela 5.2: Možni impulzi za negativno smer

Puščica označuje smer spremembe signala – iz nizkega v visoko ali iz visokega v nizko, črtica pa označuje nizko ali visoko konstantno stanje.

Če naj bo stanje pozicije na osi na voljo računalniku preko registra, je edina rešitev števec. Naloga modula je torej prepoznavati impulze in takrat števec bodisi povečati bodisi zmanjšati. Logika, ki to opravlja, ne more biti zgolj kombinatorična, ampak potrebuje tudi pomnilne elemente. Signal, ki vstopa v digitalni sistem, se vzorči z neko frekvenco. Za zaznavo spremembe je potrebna informacija o stanju signala ob vsaj dveh zaporednih periodah. V ta namen so na Spartan-3 čipu na voljo D pomnilne celice.

Na sliki 5.3 je podana logična shema števca impulzov, kakor je realizirana na Spartan-3 čipu. Signala A in B, ki prideta iz dajalca impulzov, sta na FPGA čipu speljana najprej vsak v svojo D pomnilno celico. To je potrebno zaradi sinhronizacije signala z notranjo uro. Če bi signala povezali neposredno na kombinatorično logiko, bi lahko prihajalo do sprememb signalov tudi takrat, ko bi morali biti stabilni, kar bi vodilo do nepredvidljivega obnašanja.

Ker sta oba signala potrebna v dveh zaporednih periodah, sta speljana skozi še eno D pomnilno celico.



Slika 5.3: Shema števca impulzov

Izhodi iz vseh pomnilnih celic (trenutni A: A', zakasneni A: A'', trenutni B: B', zakasneni B: B'') so speljani v kombinatorično logiko, ki iz teh podatkov razločuje impulz in smer obrata. Ti dve informaciji sta uporabljeni za krmiljenje števca. Logika precej neposredno izhaja iz definicije in je zato ne bom posebej predstavljal.

Omeniti pa je potrebno še en pojav – meta-stabilnost. Meta-stabilnost v digitalnih sistemih se lahko pojavi, kjer je vhod v pomnilno celico asinhron sistemski uri. To lahko vodi do kršenja vzpostavitevnege (ang. *setup time*) in zadrževalnega¹ (ang. *hold time*) časa. Po [10] se pri pomnilnih celicah pojavlja v obliki poznega preklopa ali celo pomanjkanje preklopa, kjer bi moral biti. Lahko se pojavi kot kratek pulz na izhodu ali celo osciliranje izhodnega signala. Meta-stabilnost je statističen pojav, katerega verjetnost pojavitve se lahko med drugim močno zmanjša tudi z uporabo zaporednih pomnilnih celic.

Delno zaradi meta-stabilnosti, delno pa zaradi možnosti šuma pri vhodnih signalih, je sistemu dodana še ena funkcionalnost, ki je ni na sliki 5.3, in sicer filter vhodnih signalov. Le-ta deluje tako, da vzame 128 vzorcev vhodnega signala in da naprej tisto vrednost, ki se največkrat pojavi. Tako se iz signala izseje morebitne visokofrekvenčne motnje in zagotovi dodatno robustnost. To si lahko privoščimo,

¹ Čas pred in po urini fronti, ko mora biti vhodni signal stabilen in veljaven.

ker ima Spartan-3 na voljo veliko resursov in, ker teče ura sistema s 33 MHz, na vhodu pa lahko pričakujemo signale z največ 50 kHz.

Pozoren bralec bo opazil, da v tem podpoglavju še nisem omenil signala R, ki predstavlja referenco. Le-to dajalec impulzov proži enkrat na obrat in služi zgolj za preverjanje pravilnosti delovanja. Njena obravnava bi sistem precej zapletla, saj imajo dajalci lahko različna števila impulzov na obrat in prav tako lahko referenca traja različno število impulzov. Zato preverjanje reference zaenkrat še ni podprto, je pa vsekakor v prvem planu za naslednje verzije.

Mehanizem za obravnavanje napak je torej v pripravi in v omejenem delovanju že deluje. Poroča se pojavitev nepravilne kombinacije signalov A in B. Torej, če se slučajno pojavi kombinacija, po kateri ne moremo sklepati niti za premik v pozitivno niti v negativno smer, se napako javi uporabniku. Več o mehanizmu poročanja napak je podano v podpoglavju 5.3.3. Ta mehanizem na sliki 5.3. ni prikazan zaradi ohranjanja enostavnosti in sporočilnosti sheme.

5.1.3. Krmiljenje elektromotorjev

Krmiljenje motorjev pomeni tvorjenje signala PWM in signala za smer iz nastavitve, ki jo poda uporabnik preko registra. Vsi signali naj bi imeli isto frekvenco, zato za nastavitve le-te zadošča en register. Poleg tega so potrebne še nastavitve moči in smeri za vsak motor posebej. Če se uporabi predznačena števila, zadošča za en motor en register; torej so potrebni še trije registri.

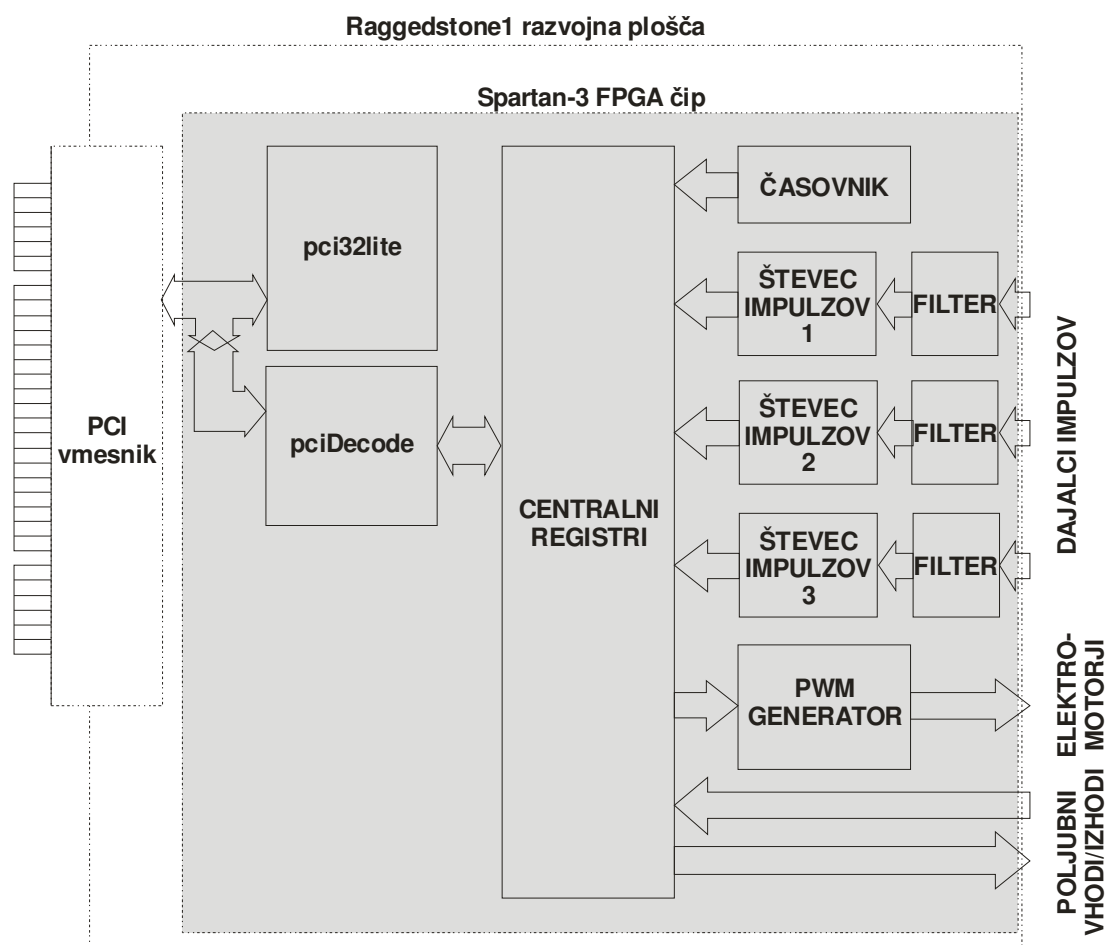
Za tvorjenje signalov ni potrebna informacija o frekvenci in procentih moči, ampak predvsem podatek, koliko period osnovne ure meri osnovna frekvenca PWM in koliko period od tega naj bo posamezen signal v visokem stanju. Ta preračunavanja v strojnem delu lahko zahtevajo precej virov, v programskem delu na računalniku pa so to trivialne operacije. Zato registri vsebujejo samo informacije o številnih period. Register o osnovni frekvenci torej hrani vrednost, koliko period osnovne ure znaša ena perioda PWM signala, registri o moči pa, koliko period naj bo signal v visokem stanju. Smer se določi iz predznaka v registru o moči. Če je moč pozitivna, je smer »0«, če je moč negativna, je smer »1«.

Sama realizacija z logičnimi elementi je od tu naprej precej enostavna. Potreben je en časovnik, ki se ob prečkanju vrednosti v osnovnem registru »frekvence« ponastavi spet na začetek in par primerjalnikov, ki v povezavi z registri moči dajejo visoko ali nizko vrednost PWM signalov.

Več podrobnosti o registrih in nastavitvah pa je podano v podpoglavju 5.3.3.

5.1.4. Celotno digitalno vezje znotraj FPGA čipa

V prejšnjih podpoglavjih so že predstavljeni posamezni moduli, v tem poglavju pa pride na vrsto sestavljanje skupaj; torej celotna shema digitalnega vezja na Spartan-3 čipu, ki si je podana na sliki 5.4.



Slika 5.4: Celotna shema digitalnega vezja, ki je realizirano na Spartan-3 čipu. Na sliki zaradi enostavnosti manjkata signala RST in CLK, ki sta napeljana do vseh modulov. Puščice označujejo tok podatkov.

Del, ki skrbi za PCI komunikacijo, je povezan z modulom »CENTRALNI REGISTRI«. Nanj so priključeni tudi ostali vhodno/izhodni moduli in je torej zadolžen za hranjenje podatkov, njihovo obdelavo in primerno predstavitev. Poskrbeti mora, da se za določen naslov odzove pravi register. Prav tako skrbi, da se ob branju časovnika shranijo tudi vrednosti vseh treh števecov – več o tem je podano v poglavju 5.3.3. Ob ostalih registrih pa hrani tudi zastavice napak in skrbi za možnost njihovega brisanja.

Na desni strani sheme so nakazani zunanji signali, vendar ti signali niso neposredno povezani na naprave. Vmes se nahaja še vezje, ki zagotavlja zaščito in optično izolacijo.

5.2. Vezje

Razvojna plošča Raggedstone1 vsebuje vse potrebno za izvedbo logičnega dela vmesnika. Ima tudi množico vhodno/izhodnih priključkov, kamor so povezani signali iz stružnice.

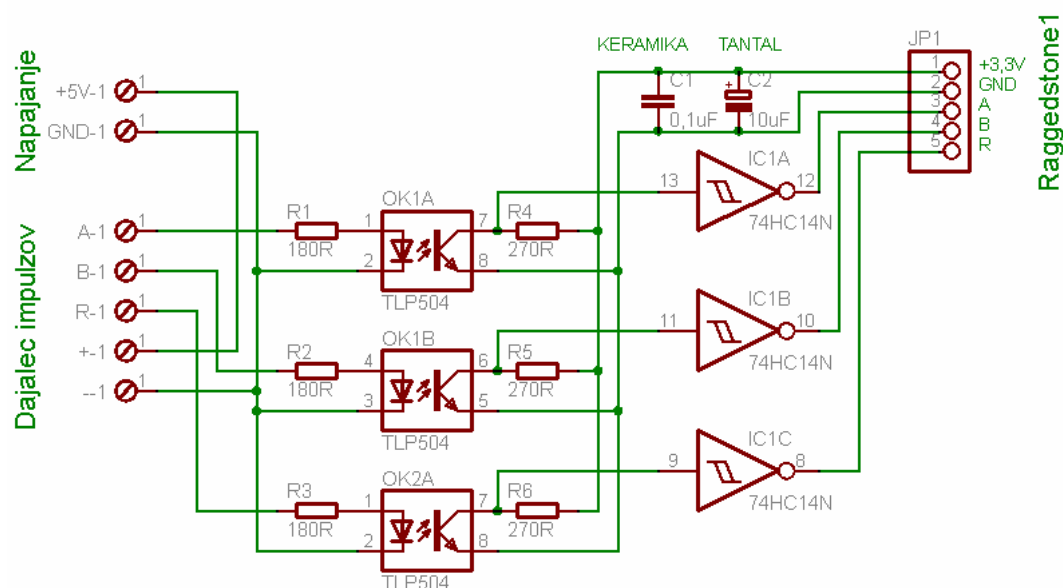
Elektronika na stružnici deluje pod velikimi napetostmi in tokovi, zato je ni pametno neposredno priklapljati na občutljiv Spartan-3 in računalnik. V ta namen je vstavljena vmes optična izolacija. Njena funkcija je preprečevanje širjenja prenapetostnih motenj in po drugi strani prilagoditev napetosti signalov. Glavna vloga razširitvenega modula je torej optična izolacija.

Posamezni deli tega vezja so bili realizirani že sproti, med kodiranjem, saj je bilo potrebno posamezne sklope kode testirati. V ta namen so bili sestavljeni na plastični prototipni plošči (ang. *proto-board*). Ta način se je za potrebe testiranja posameznih sklopov izkazal kot zadosten. Ni pa primeren za realizacijo celotnega sistema, saj je tam ogromno povezav, na prototipni plošči pa se lahko hitro kak kabel iztakne, odkrivati pravega pri takšni množici pa je nočna mora.

Za izdelavo prototipa se je izkazala najbolj primerna malo drugačna prototipna plošča, ne klasična plastična, ampak vnaprej tiskano vezje, ki vsebuje mrežo luknjic. Elemente se nanj preprosto spajka, prav tako pa se s spajkanjem ustvarjajo tudi povezave, bodisi direktno, bodisi s pomočjo žičk. Izdelava je vzela precej časa, saj je bilo potrebno ročno spajkati vsako povezavo, ampak rezultat je zelo zadovoljiv.

Vezje se lahko razdeli na par osnovnih sklopov, ki se potem podvajajo za vsak vhodno/izhodni signal ali napravo. Sheme posameznih sklopov so predstavljene v naslednjih podpoglavjih.

5.2.1. Priklop dajalca impulzov



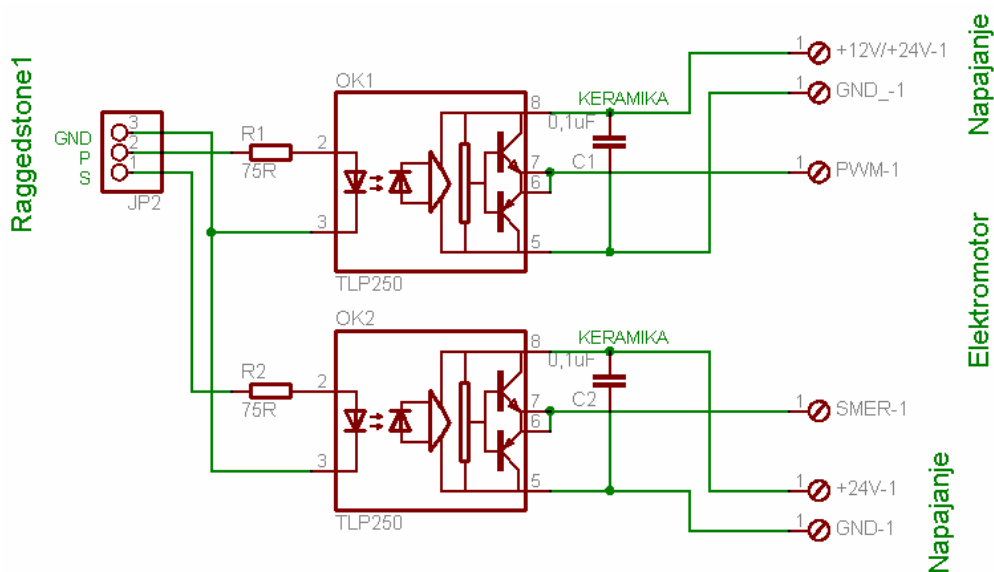
Slika 5.5: Vezje za priklop dajalca impulzov

Na sliki 5.5 je podana shema vezja, ki je potrebno za priklop dajalca impulzov. Posamezni signal, ki pride iz dajalca, je preko upora vezan na svetlobno diodo znotraj opto sklopnika. Na drugi strani se nahaja »pull-up« upor proti 3,3V, kar signal invertira. Signal gre potem preko invertirajočega »Schmitt trigger« elementa proti FPGA čipu na Raggedstone1 plošči. »Schmitt trigger« je potreben, ker opto sklopnik signal preklaplja precej počasi. Tako so dvižni (ang. *risetime*) in spustni (ang. *falltime*) časi precej veliki – reda velikosti mikrosekund. V tem času pa FPGA vzame že več vzorcev signala in bi zato lahko prišlo do meta-stabilnega stanja. »Schmitt trigger« te čase zmanjša na red velikosti nanosekund in tako precej zmanjša verjetnost pojava meta-stabilnosti.

Na desnem zgodnjem delu sta vidna še dva kondenzatorja – uporabljata se za izboljšanje delovanja 74HC14 čipa.

Upori pred opto sklopnikom so izbrani tako, da ob predpostavljeni napetosti zagotavljajo potreben tok. »Pull-up« upori pa so kompromis med dvižnim časom in porabo toka v visokem stanju signala. Vrednosti uporov so bile tudi testirane s pomočjo osciloskopa in generatorja impulzov. Ugotovljeno je bilo, da omenjena kombinacija verno prepušča frekvence signalov do 50kHz. Ker se dajalcu impulzov nahajata dva signala, to znese podporo frekvenci impulzov do 100 kHz. To potrebam tega diplomskega dela povsem zadošča.

5.2.2. Priklop elektromotorja

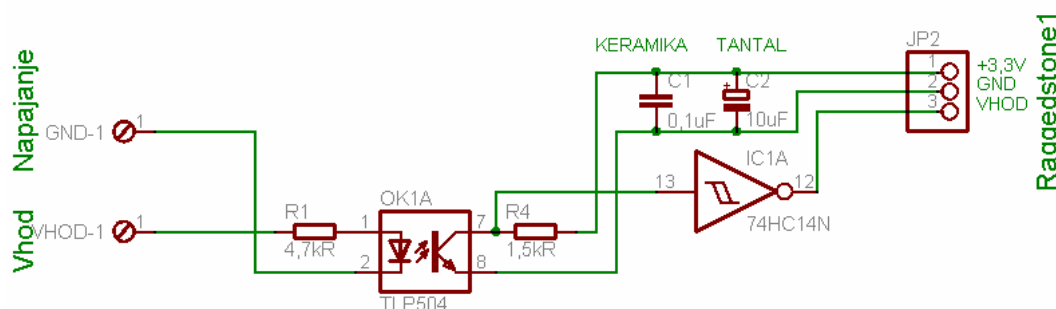


Slika 5.6: Vezje za priklop elektromotorja

Na sliki 5.6 je podana shema vezja, ki je potrebno za priklop elektromotorja. Posamezni signal, ki pride iz FPGA čipa na Raggedstone1 plošči, je preko upora vezan na svetlobno diodo znotraj opto sklopnika. Tu so v uporabi posebni, močnejši opto sklopniki, ki imajo na izhodu že kompleten signal in ne potrebujejo »pull-up« uporov. Po specifikaciji pa potrebujejo zunanji kondenzator, zato sta na shemi tudi dva prisotna.

Signal iz tega opto sklopnika gre neposredno na izhodni priključek modula. Signal PWM naj bi krmilil močnejši tranzistor, ki regulira tok za poganjanje elektromotorja. Signal za smer pa naj bi preklapljal rele ali H-most, ki zamenjata priklop signalov na elektromotor ter tako zamenjata smer. Za oboje (PWM in smer) so ti opto sklopniki dovolj močni, saj podpirajo tudi do 1A izhodnega toka.

5.2.3. Priklop poljubnih vhodnih signalov



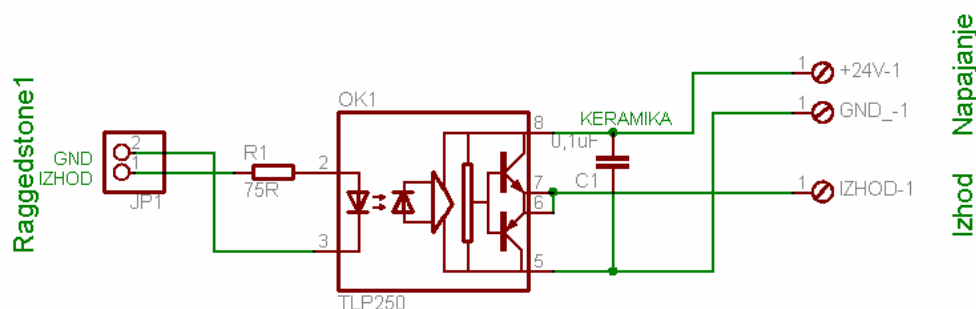
Slika 5.7: Vezje za priklop vhodnih signalov

Poleg priključkov za dajalce impulzov ima vezje priključke tudi za poljubne vhodne signale, na katere se lahko priklopijo različne naprave ali stikala.

Shema za primerek vhodnega signala, ki je podana na sliki 5.7, je zelo podobna shemi vezja za dajalca impulzov. Razlike so le pri vrednostih uporov. Na vhodni strani je ta vrednost precej višja, ker so pričakovani vhodni signali z napetostmi do 24V. Na izhodni strani pa je močnejši upor, ker tu načeloma ni potrebna visoka frekvenca. Upor je močnejši zaradi manjše porabe toka. Izbira uporov je bila narejena na predpostavki, da manjši upor porablja več toka, a omogoča prepuščanje višjih frekvenc signala. Ko je bilo vezje že dokončano, se je izkazalo, da to ne drži. Manjši upor zmanjšuje čas za dvig signala, a zato ohranja daljši čas za spust signala.

Z generatorjem impulzov in osciloskopom je bila sestavljena kompozicija preverjena in izkazalo se je, da omogoča prepuščanje frekvence tudi do 70kHz, kljub temu, da je »pull-up« upor načeloma počasnejši.

5.2.4. Priklop poljubnih izhodnih signalov



Slika 5.8: Vezje za priklop izhodnih signalov

Shema za primerek izhodnega signala na sliki 5.8 je praktično enaka shemi izhodnega signala za krmiljenje elektromotorja.

Ti poljubni izhodni signali se, kot že omenjeno, uporabljajo za krmiljenje raznih dodatnih naprav na stružnici.

5.3. Uporabniška navodila

To poglavje je napisano v obliki navodil za uporabnika sistema. Predstavi način uporabe in pogled na vmesnik z več vidikov.

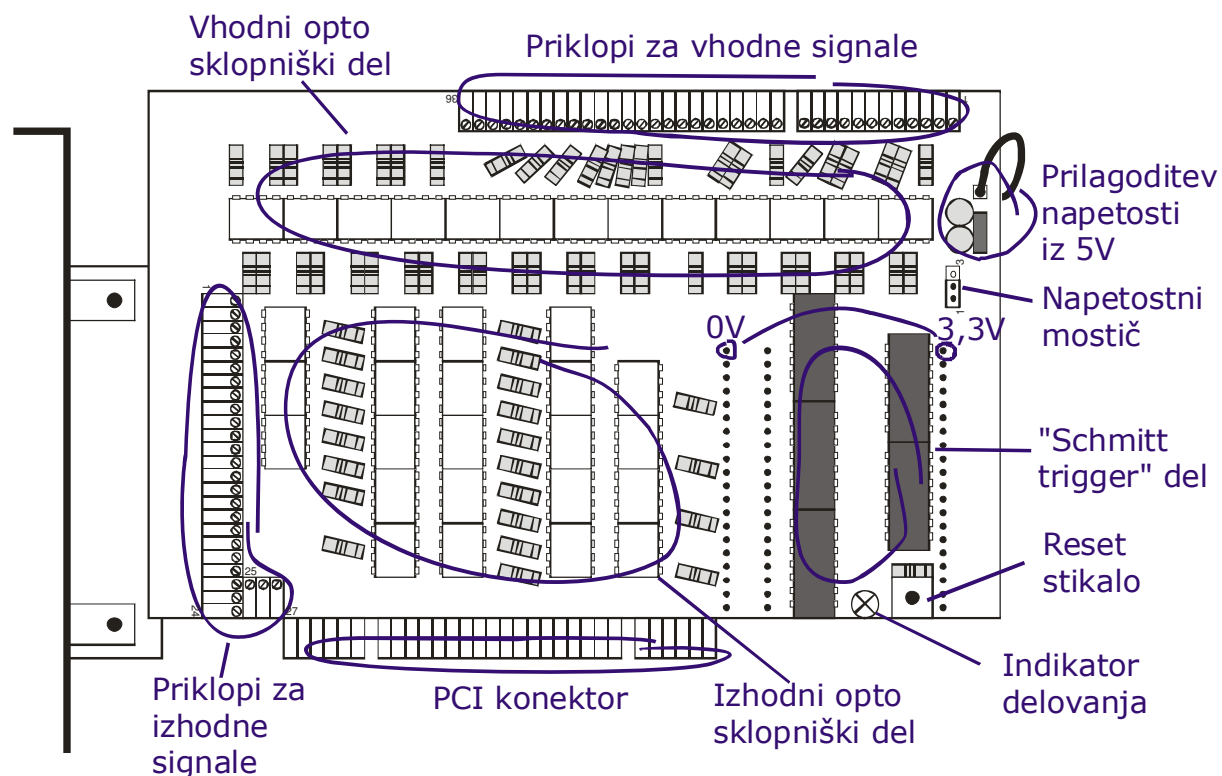
5.3.1. Namestitev

Preden kartico namestite v računalnik, morate preveriti, če je potreben V/I prostor v operacijskem sistemu na voljo. V primeru, da je prostor zaseden in kartico vseeno vstavite, lahko operacijskemu sistemu povzročite resne težave. Vmesnik potrebuje proste V/I naslove od *0x0200* do *0x022F*. Na operacijskem sistemu Linux so že dodeljeni naslovi zapisani v datoteki *»/proc/iports«*, torej jih lahko pregledate z ukazom *»more /proc/iports«*. Če so potrebni naslovi še prosti, lahko nadaljujete z vstavitvijo kartice v računalnik. V primeru, da so naslovi zasedeni, še ni vse izgubljeno. Lahko poskusite prestaviti napravo, ki jih zaseda ali pa kontaktirate avtorja vmesnika, ki vam lahko prilagodi kartico, da se odziva na druge naslove.

Ko preverite, da so potrebni naslovi na voljo, lahko kartico z dodanim modulom namestite znotraj računalnika v prosto PCI režo.

Edina stvar, za katero morate na tej točki še poskrbeti, je nastavitev napetosti. Preveriti morate, če vaše PCI vodilo podpira napetost 3,3V, in ustrezno nastaviti napetostni mostič (ang. *jumper*). To lahko preverite z navadnim voltmetrom. Oglejte si sliko 5.9 in sonde nastavite na nožici, kjer piše *0V* in *3,3V*. Voltmeter mora pokazati napetost okoli 3,3V. Če kaže manj, npr. 2,5V, potem vaše vodilo ne podpira želene napetosti in morate napetostni mostič nastaviti na regulirano napetost. Več o nastavitvi mostiča je podano v naslednjem podpoglavju.

5.3.2. Shema in priklop signalov



Slika 5.9: Shema razširitvenega vezja vmesnika

Na sliki 5.9 je podana shema vmesnika, oziroma predvsem njegovega razširitvenega modula. Označeni so posamezni deli.

Napetostni mostič

Razširitveni modul se lahko napaja iz dveh virov – iz PCI vodila, če le-to podpira 3,3V napetost, ali s pomočjo regulatorja iz napetosti 5V, ki pa je prisotna na vseh PCI vodilih. V naslednji tabeli si lahko ogledate možnosti nastavitve.

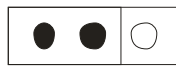
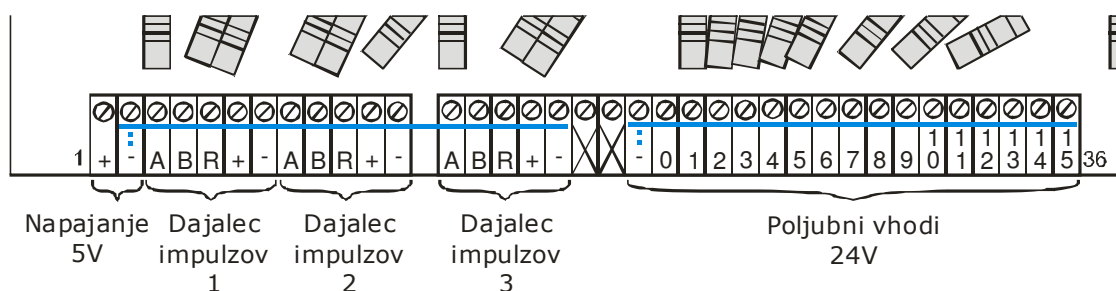
 1 2 3	Mostič	Nastavitev
	1-2	Neposredno napajanje 3,3V.
	2-3	Regulirano 3,3V napajanje iz 5V.

Tabela 5.3: Nastavitve napetostnega mostiča

Vhodni signali



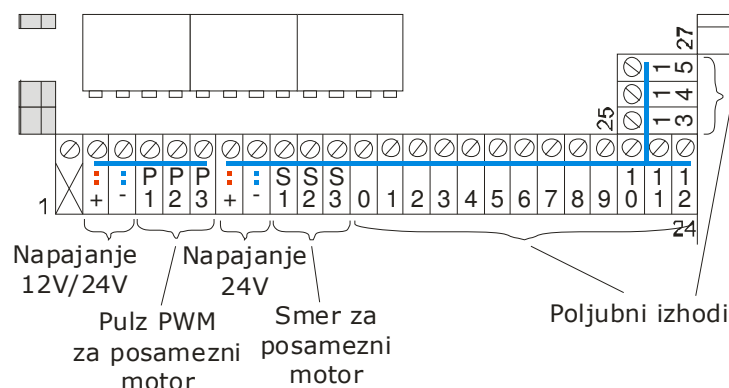
Slika 5.10: Shema vhodnih signalov razširitvenega modula

Zaradi večje fleksibilnosti in lažjega pritrdjevanja kablov so na plošči kabelski priključki, v katere se jih lahko privije. Iz slike 5.10 se lahko razbere pomen posameznih priključkov. Z modro črto je označeno, katere priključke pokriva posamezna masa. Vsi signali so optično ločeni od logičnega dela in računalnika.

Napajanje na levi strani skrbi za napajanje vseh treh dajalcev impulzov. Napajanje mora biti dovolj močno, da lahko zadosti potrebam dajalcev ter še približno 200mA več. Vsak posamezni dajalec ima priključke za priklop podatkovnih signalov in priključke, iz katerih črpa napajanje.

Poljubni vhodi imajo vsi skupno maso, za vsak posamezen signal pa je predvideno, da podpira napetosti vsaj do 24V.

Izhodni signali



Slika 5.11: Shema izhodnih signalov razširitvenega modula

Za izhodne signale so enaki priključki kot za vhodne. Na sliki 5.11 je prikazan pomen posameznih priključkov. Z modro črto je označeno, katere priključke pokrivata posamezna masa in napajalna napetost. Tako lahko s slike razberete, kakšno napetost lahko pričakujete na posameznem priključku. Tudi tu so vsi signali optično ločeni od logike in računalnika.

Signali za pulz PWM imajo ločeno napajanje, ker je v nekaterih primerih predvidena uporaba 12V in ne 24V. Ostali priključki, vključno s tremi za smer elektromotorja, imajo skupno napajanje 24V. Posamezni priključki imajo zadaj močne opto sklopnike, zato je res, da zmorejo bremena tudi do 1A. Vendar naj skupni seštevek bremen ne presega 3A, zaradi povezav na plošči in skupnega napajanja. Tudi napajalnik mora biti zato sposoben prenašati breme 3A. Izhodni priključki so načrtovani predvsem za vklapljanje in izklapljanje relejev.

Nadzor delovanja

V spodnjem desnem delu sheme na sliki 5.9 lahko vidite indikator delovanja v obliki LED diode. Ta označuje, ali je vmesnik v delovanju. Če gremo v podrobnosti, LED dioda gori, če ima vmesnik tekoč urin signal.

Zraven se nahaja *reset* stikalo, ki celotno vezje ponastavi (ang. *reset*) v začetno stanje. POZOR! Stikalo postavi na privzeto vrednost vse spremenljivke, števce in registre. Če so bile opravljene kakršnekoli programske nastavitve (npr. frekvenca PWM) ali nastavljeni izhodni signali, jih je vse potrebno ponovno nastaviti!

5.3.3. Programski dostop

S programskega vidika je dostop do vmesnika mogoč preko nekaterih registrov, ki so preslikani v vhodno/izhodni naslovni prostor osebnega računalnika. Seznam vseh registrov in njihove osnovne opise si lahko ogledate v naslednji tabeli.

Naslov	Naziv	Tip	Opis
0x200	ČASOVNIK	B, N	Časovnik, ki teče s frekvenco 260.416,66 Hz. Branje sproži prenos vrednosti števecv DI v registre.
0x204	DI1	B, Z	Predznačeni števec dajalca impulzov številka 1. Pred branjem mora bit prebran register ČASOVNIK.
0x208	DI2	B, Z	Predznačeni števec dajalca impulzov številka 2. (enako)
0x20C	DI3	B, Z	Predznačeni števec dajalca impulzov številka 3. (enako)
0x210	PWMAKS	P, N	Število urinih period, ki označuje frekvenco PWM. Ura teče s frekvenco 33MHz.
0x214	EM1	P, Z	Smer in moč elektromotorja številka 1. Smer je izražena s predznakom, moč pa v številu urinih period, ko naj bo PWM signal v visokem stanju.
0x218	EM2	P, Z	Smer in moč elektromotorja številka 2. (enako)
0x21C	EM3	P, Z	Smer in moč elektromotorja številka 3. (enako)
0x220	PV	B	Poljubni vhodi.
0x224	PI	P	Poljubni izhodi.
0x228	NAPAKE	BP	Register z zastavicami napak.

Tabela 5.4: Seznam programsko dostopnih registrov in njihovih lastnosti. »B« pomeni bralni, »P« pisalni register; »N« predstavlja nepredznačena naravna števila, »Z« pa predznačena cela števila.

V prvem stolpcu se nahaja heksadecimalni naslov registra. Vsi registri so 32 bitni, zato vsak zaseda štiri naslovne lokacije. Vsi dostopi do registrov, torej branja in pisanja, naj bodo 32 bitni. Krajši niso podprti. Registri so bralni ali pisalni in se iz njih samo bere ali samo piše. Tip je podan v tabeli. Branje pisalnega registra vrne neveljavne podatke, pisanje v bralni register pa je prazna operacija. Edina izjema je register NAPAKE, pri katerem se zastavice brišejo s pisanjem vanj. Vrednosti v registrih predstavljajo bodisi nepredznačena cela števila (ang. *unsigned integer*) ali predznačena cela števila (ang. *signed integer*).

Časovnik se lahko uporablja za usklajevanje procesov s časovno komponento. Poleg tega ima posebno funkcijo – ob branju časovnika se v registre DI1, DI2 in DI3 preslikajo vrednosti iz števecv. Če hočete brati ažurne vrednosti števecv, morate torej najprej prebrati števec. To ni omejitev, ampak posebna funkcionalnost, ki omogoča, da so stanja vseh treh števecv sinhronizirana. Branje števecv poteka namreč zaporedno in, ko bi prišli do zadnjega, bi se lahko prva dva vmes že spremenila in njihovih vrednosti ne bi imeli usklajenih. Tako pa se vsi trije registri vzporedno preslikajo ob branju števca in teh težav ni.

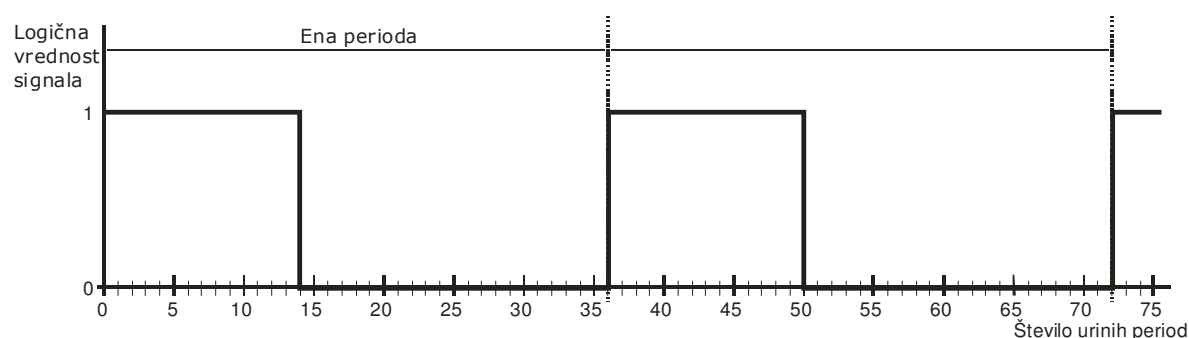
Števci dajalcev impulzov hranijo pozicijo na posamezni osi. Ob premiku v pozitivni smeri se vrednost povečuje, v negativni smeri pa zmanjšuje. Z morebitnim prelivom se ne potrebujete obremenjevati, saj register zadostuje za vsaj kilometer pomika v vsako smer pri najboljših dajalcih impulzov. Točka »0« se nahaja tam, kjer je bil sistem nazadnje ponastavljen. Če želite uporabiti drugo točko, lahko preprosto uporabite programski odmik in ga vedno odštevate od brane vrednosti.

Register PWMMAKS hrani število urinih period, kar določa frekvenco PWM signalov. Vsi signali imajo skupno frekvenco. Lahko se jo izračuna po formuli (5.1).

$$f = \frac{33.333.333}{n} [Hz] \quad (5.1)$$

Privzeta vrednost PWMMAKS je 833.333, kar predstavlja frekvenco 40MHz.

V registre EM1, EM2 in EM3 zapišete predznačeno število, ki predstavlja smer in moč. Če je število pozitivno, bo na smernem izhodu za dotični elektromotor signal, ki bo ustrezal logični »0«, če pa bo število negativno, bo signal za logično vrednost »1«. Sama vrednost pa predstavlja število urinih period, ko naj bo PWM signal visok. Če je vrednost »0«, je PWM signal stalno nizek. Če pa je vrednost večja ali enaka PWMMAKS, potem je signal stalno visok.



Slika 5.12: Primer tvorjenega PWM signala

V primeru na sliki 5.12 je PWMMAKS nastavljen na 36, kar predstavlja frekvenco okoli 926kHz. EM pa je nastavljen na 14, kar predstavlja $14/36=39\%$ moč motorja.

Register PV vsebuje logično predstavitev stanj vhodnih signalov. Zgornjih 16 bitov je rezerviranih, spodnjih 16 pa vsebuje predstavitve ustreznih signalov. Številčenje bitov je po ozki-konec (ang. *little-endian*) terminologiji. Bitu na nekem mestu v PV registru ustreza vhodni signal na istoštevilskem priključku.

V register PI vpišemo krmiljenje poljubnih izhodnih signalov. Zgornjih 16 bitov je rezerviranih, spodnjih 16 pa vsebuje predstavitve ustreznih signalov. Številčenje bitov je tudi tu po ozki-konec terminologiji. Bit na nekem mestu v PI registru ustreza izhodnemu signalu na istoštevilskem priključku.

Register napake vsebuje zastavice, ki označujejo, da se je zgodila posamezna napaka. Seznam zastavic je v naslednji tabeli.

Bit	Oznaka	Opis
0	DIREF	Dajalec impulzov, referenca. Signal referenca ne ustreza stanju števca. Zgodila se je napaka bodisi v štetju – lahko so izgubljeni ali podvojeni impulzi – bodisi na signalu reference. <i>POZOR! Obravnava te napake še ni ustrezno implementirana!</i>
1	DIKOMB	Dajalec impulzov, kombinacija. Od enega dajalca impulzov je prišla neveljavna kombinacija signalov.
2-31	Rezervirano	

Tabela 5.5: Seznam zastavic v registru napak

V primeru pojavitve posamezne napake se ustrezna zastavica postavi na »1« in tako ostane, dokler je ne *izbrišemo*. Izbriše pa se tako, da se na ustrezno mesto v registru vpiše vrednost »1«.

5.3.4. Primer uporabe

Najenostavnejša uporaba vmesnika je na operacijskem sistemu Linux v programskem jeziku C. Zato je ta kombinacija tudi uporabljena za demonstracijo delovanja. Za uporabo potrebujete imeti dostop do uporabnika »root«.

Ustvarite novo datoteko z imenom »struznica.c«, kjerkoli želite. Vanjo prenesite naslednji program:

```
#include <sys/io.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]){
    unsigned int casovnik;
    int di1, di2, di3;
    unsigned int pv, nap;
    unsigned int frek, pwmn;
    int moc1, moc2, moc3;
    frek = 80; //pwm frekvenca = 80Hz
    moc1=20; //motor 1, 20% moci
    moc2=-40; //motor 2, 40% moci, negativna smer
    moc3=100; //motor 3, 100% moci

    pwmn=33333333/frek;
    if (ioperm(0x200,48,3)==0){
        outl(pwmn,0x210); //nastavitev pwm frekvence
        outl(moc1*((int)pwmn)/100,0x0214); //nastavitev motorja 1
        outl(moc2*((int)pwmn)/100,0x0218); //nastavitev motorja 2
        outl(moc3*((int)pwmn)/100,0x021C); //nastavitev motorja 3
        outl(0xF,0x224); //postavi zadnje 4 PI v visoko stanje
        outl(3,0x228); //ponastavi obstojece zastavice napak

        do {
            casovnik = inl(0x200);
            di1 = inl(0x204); //branje stevca impulzov 1
```

```

    di2 = inl(0x208); //branje stevca impulzov 2
    di3 = inl(0x20C); //branje stevca impulzov 3
    pv = inl(0x220); //branje poljubnih vhodov
    nap = inl(0x228); //branje zastavic napak
    printf("\rCASOVNIK: %d OS1: %d OS2: %d OS3: %d PV: %X NAP: %X"
           ,casovnik,di1,di2,di3,pv,nap);
    fflush(stdout);
} while (1);
} else {
    printf("NAPAKA: ioperm()");
    return -1;
}
return 0;
}

```

Ko to shranite, morate odpreti terminal in v njem poiskati mapo, v katero ste shranili program. Z naslednjim ukazom program prevedete:

```
gcc struznica.c -o struznica
```

Z naslednjim ukazom pa program poženete:

```
sudo ./struznica
```

Če ukaze izvajate že kot uporabnik »root«, lahko ukaz »sudo« izpustite.

Program je sestavljen iz treh delov. Po inicializaciji spremenljivk sledi klic ukaza »ioperm«, ki programu omogoči dostop do želenega razpona naslovov v vhodno/izhodnem prostoru. Nato sledi drugi del – nastavljanje elektromotorjev, poljubnih izhodov in ponastavljanje zastavic napak.

Tretji del pa se v zanki ponavlja – branje vseh števcov impulzov, poljubnih vhodov in registra napak ter izpisovanje tega na zaslon.

Program lahko prekinete z uporabo kombinacije tipk »CTRL-C«.

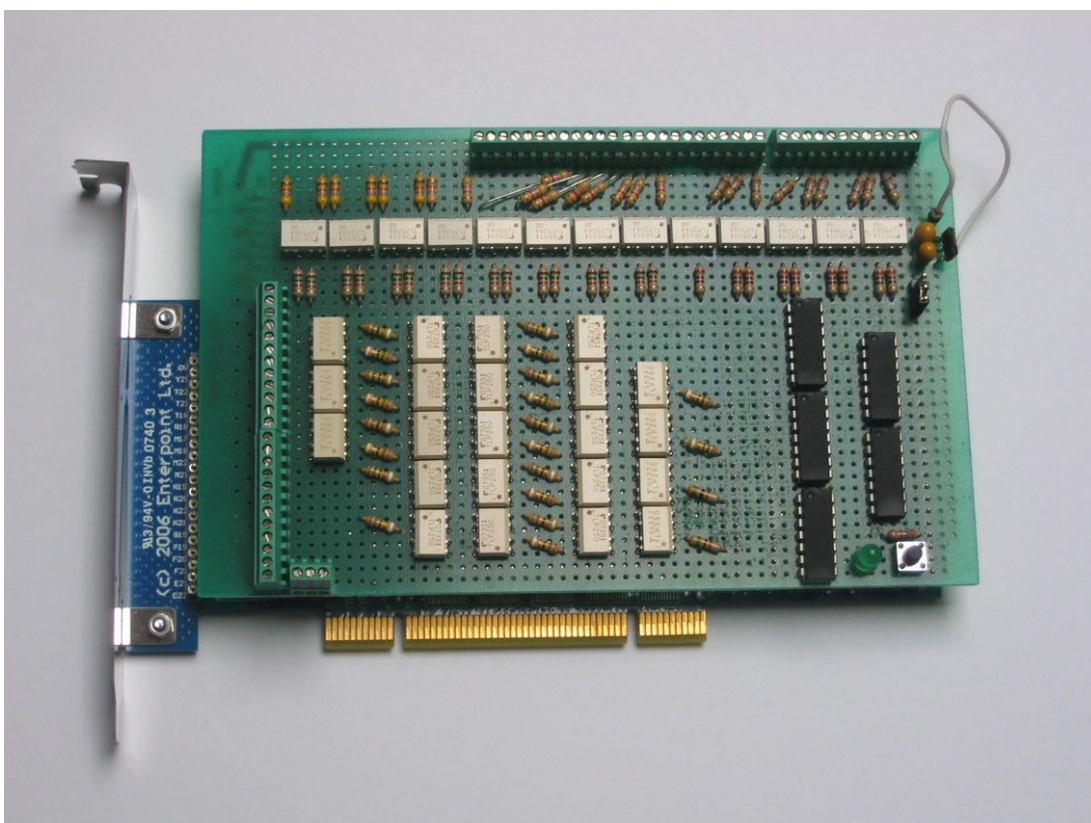
6. Sklepne ugotovitve

V diplomskem delu sem skušal razviti sistem za računalniško krmiljenje stružnice. Lotil sem se področja, ki ga nisem najbolj poznal, in uporabil nekatere tehnologije, ki so bile zame nove. Spoznavanje vsega tega in s tem povezani problemi so mi vzeli precej več časa, kot sem ga predvidel za to delo. Kljub nekaterim zapletom je zastavljena naloga uspela in naučil sem se veliko novega. Nepoznano področje je postalo bolj poznano in učenje na lastnih napakah je še vedno najbolj učinkovito učenje.

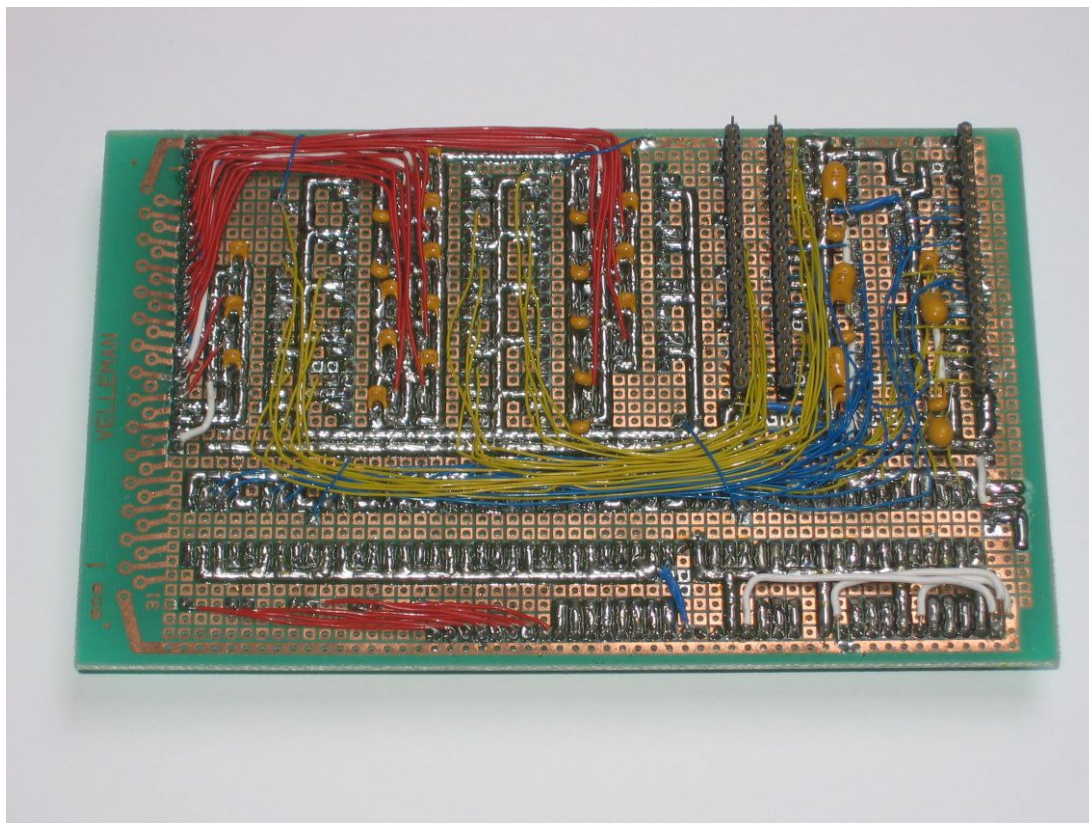
Sistem je delujoč, vendar še ni v celoti zaključen. Med razvojem vedno prihajajo ideje, kaj bi se še dalo izboljšati – od zgolj praktičnih do pravih sistemskih izboljšav. Naj jih nekaj navedem:

- Razširitev na več osi. Tako bi se lahko isti sistem uporabil tudi za druge obdelovalne stroje, ki uporabljajo več osi, npr. rezkalni stroj, koordinatni vrtalni stroj, specialni tiskalniki ...
- Izbrati hitrejša vhodna opto sklopnike. Tako bi lahko sistem štel impulze tudi s hitrejših naprav.
- Upoštevanje reference pri štetju impulzov. Ni nujen pogoj, saj sistem brez tega deluje, bi pa vsekakor precej povečalo zanesljivost.
- Prenovitev PWM generatorja, da ne bi proizvajal sinhroniziranih signalov, ampak bi bili razpršeni. Sedaj se namreč dogaja, da vsi signali hkrati prehajajo v visoko stanje, kar povzroča hkratno porabo toka vseh elektromotorjev. S prenovitvijo bi se dalo to porabo malo razpršiti.
- Uporabiti podnožja za čipe. Posebno praktično bi bilo to pri opto sklopnikih. Če bi prišlo do prenapetosti ali drugih težav in bi element pregorel, bi ga preprosto zamenjali z drugim. Sedaj je to zelo težko, če ne celo nemogoče.

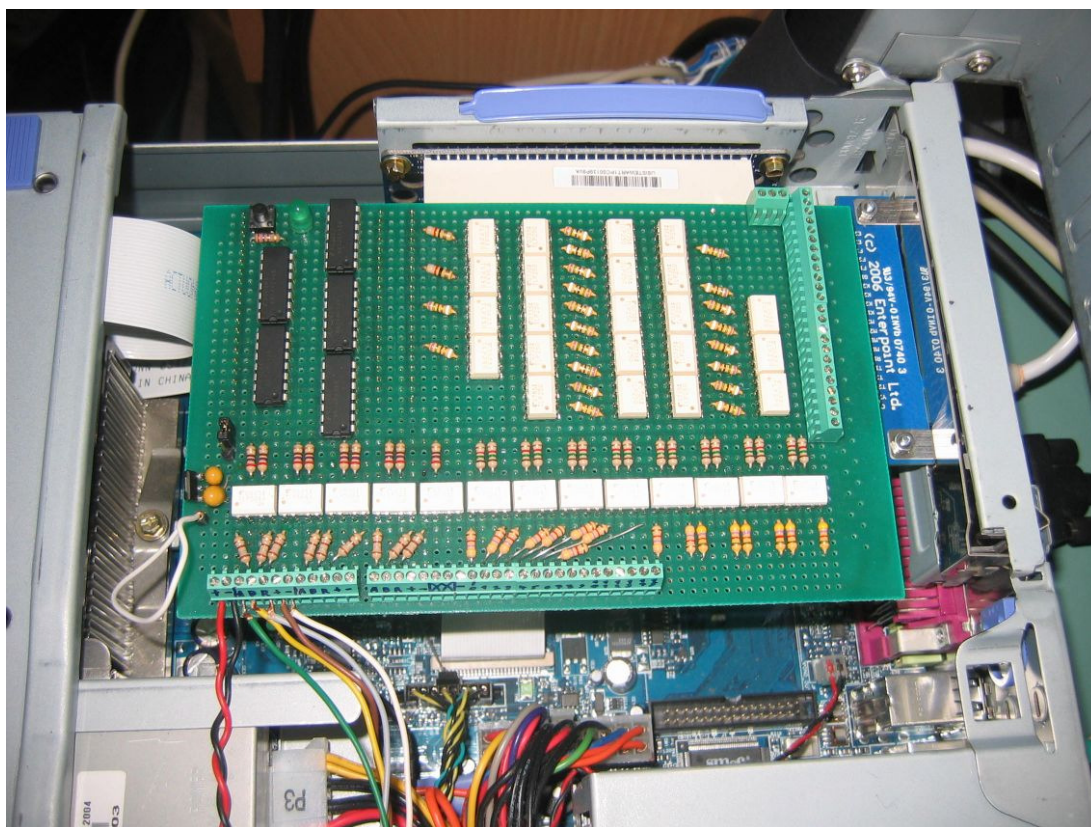
Dodatek A – Fotografije sistema



Slika 6.1: Fotografija celotnega sistema. Na sliki je viden predvsem modul vezja za optično izolacijo. Pod njim se lahko vidi tudi nekaj delov Raggedstone1 razvojne plošče.



Slika 6.2: Fotografija modula za optično izolacijo od spodaj. Na sliki je dobro vidna množica povezav, ki so morale biti vse ročno speljane.



Slika 6.3: Fotografija sistema, vstavljenega v računalnikovo PCI režo. Spodaj levo je tudi viden priklop dajalca impulzov.

Uporabljeni viri

- [1] G. Boothroyd, W. A. Knight, *Fundamentals of Machining and Machine Tools, Third Edition*, New York: Marcel Dekker, 1988, pogl. 1.
- [2] (2008) Wikipedia: Numerical control. Dostopno na:
http://en.wikipedia.org/wiki/Numerical_control
- [3] (2008) Wikipedia: CNC. Dostopno na:
<http://en.wikipedia.org/wiki/Cnc>
- [4] (2008) Technology, PC based machine control, the pros and cons. Dostopno na:
<http://www.neecontrols.com/pc-based-machine-control.html>
- [5] *PCI Local Bus Specification, Revision 3.0*, Oregon: PCI-SIG, 2002
- [6] (2007) Patricio Bulić: Načrtovanje in sinteza digitalnih in mikroprocesorskih sistemov v FPGA. Dostopno na:
https://www.fri.uni-lj.si/file/84055/vhdl_zapiski.pdf
- [7] (2008) Don Morgan: DSP and Motion Control. Dostopno na:
<http://www.embedded.com/2000/0006/0006spectra.htm>
- [8] (2008) Raggedstone1 User Manual, Issue – 1.03. Dostopno na:
http://www.enterpoint.co.uk/moelbryn/RaggedStone_User_Manual_Issue_1_03.zip
- [9] (2008) PCI Target: Overview. pci32tlite_oc. Dostopno na:
http://www.opencores.org/projects.cgi/web/pci32tlite_oc/overview
- [10] (1997) Metastability Considerations, Application Note - Version 1.0. Dostopno na:
http://www.klabs.org/richcontent/General_Application_Notes/mestablestates/xilinx_metastable_considerations.pdf
- [11] (2008) Enhanced Machine Controller project. Dostopno na:
<http://www.linuxcnc.org/>

Izjava o avtorstvu

Izjavljam, da sem diplomsko delo izdelal samostojno pod vodstvom mentorja prof. dr. Dušana Kodeka. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Blaž Hostnik