Univerza v Ljubljani

Fakulteta za elektrotehniko

Nejc Recelj

Izboljšanje ločljivosti magnetnega senzorja zasuka z vezjem FPGA

Magistrsko delo

Mentor: prof. dr. Andrej Žemva

Ljubljana, 2017

Zahvala

Zahvaljujem se mentorju Andreju Žemvi za napotke in pomoč pri izvedbi magistrske naloge. Prav tako se zahvaljujem podjetju RLS, ki mi je omogočilo izdelavo magistrske naloge, mentorju v podjetju Radu Glasu, sodelavcu Juretu Zdovcu in tudi ostalim sodelavcem, ki so mi pomagali z nasveti. Velika zahvala gre vsem mojim najbližjim, ki so mi stali ob strani tekom celotnega študijskega procesa in me ves čas spodbujali.

Vsebina

Povzetek

Cilj magistrske naloge je bil doseči višjo ločljivost za 12-bitni magnetni senzor zasuka AM4096, ki bere položaj magneta in sporoča njegovo pozicijo v različnih izhodnih oblikah. Senzor ima prisoten šum na inkrementalnem izhodu in se pojavlja ob prehodu iz ene pozicije na drugo ter je v trenutni izvedbi enkoderja odpravljen s histerezo.

V moji nalogi sem šum na inkrementalnem izhodu filtriral s filtrom tekočega povprečja. Zaradi potrebe po hitrih odzivih sem filtriranje implementiral na vezju FPGA, ki je hitrejši od mikroprocesorjev in lahko več operacij izvaja hkrati. To je bil tudi razlog, da sem uporabil najenostavnejši FIR filter, ker je za implementacijo na FPGA najlažji.

Na začetku sem podal teoretično rešitev, ki je potrdila, da filtriranje lahko poveča ločljivost senzorja. V drugem delu je opisana implementirana rešitev in delovanje filtra v realnih razmerah ter prikaz rezultatov filtriranega inkrementalnega signala.

S filtriranjem je bilo dosežena 14-bitna ločljivost in odpravljen šum. Dodana sta 2 nova bita, kar omogoča, da senzor loči 16384 pozicij namesto prejšnjih 4096. Na filtriranem izhodnem signalu je negativen efekt to, da je prisoten zamik za velikost filtra, kar pa je bilo predvideno že na začetku.

Ključne besede: magnetni senzor zasuka, filtriranje, FIR, tekoče povprečje

Abstract

The of objective in this master thesis was to a achieve higher resolution for 12 bits magnetic rotational sensor AM4096, which gives positional information of the magnet, with different output formats. Sensor makes noise at every time it changes position, which is in the current encoder eliminated with hysteresis.

In my work I filtered the present noise with the moving average filter. Since fast responses are required, I used FPGA instead of a microprocessor because all processes thus run faster and in parallel. That was also the main reason why I used the simplest FIR filter, the easiest to implement on the moving average filter.

At the beginning of my work I presented a theoretical solution which proved that with filtering, higher resolution can be achieved. In the second part I described the implemented solution and presented the results of the filtered incremental signal.

With filtering, 14 bits resolution and reduced noise were achieved. With 2 added bits the encoder can separate 16384 positions of the magnet instead of the previous 4096. Negative effect of the filtering, a delay, was predicted in advance and taken in consideration. Filtering, with all its positive and negative effects, gave positive results and is therefore termed as successful.

Key words: rotational magnetic sensor, filtering, FIR, moving average

1 Uvod

Podjetje RLS je eden vodilnih proizvajalcev lineranih in rotacijskih magnetnih enkoderjev na svetu. Imajo velik oddelek razvoja, kjer skrbijo, da je kar se da veliko izdelkov v celoti plod njihovega znanja in truda. Eden izmed teh je tudi senzor AM4096, ki daje podatke o zasuku oziroma poziciji [1]. Prodaja senzorja upada, saj na trgu obstajajo konkurenčni izdelki, ki dosegajo višjo ločljivost. AM4096 dosega 12-bitno ločljivost, kar pomeni, da zazna 4096 pozicij ob opisu 360 stopinj.

Senzor ima ob prehodih pozicij prisoten šum, ki je v trenutni izvedbi enkoderja odpravljen s histerezo. Če te histereze ni, lahko prihaja do približno pet nezaželjenih prehodov, katerih število se z manjšanjem hitrosti vrtenja povečuje. V moji nalogi je bil cilj, da s povprečjem šuma, ki se pojavlja, pridobim 2 dodatna bita in v čim večji meri odpravim šum.

Za uspešno filtriranje so potrebni hitri odzivi, kar je dobro izvedljivo z uporabo FPGA (angl.: Field-programmable gate array), kjer procesi potekajo veliko hitreje kot v mikrokontrolerju, saj se vsi izvajajo vzporedno. Glede na to, da novi izdelek ne bo smel cenovno bistveno odstopati od trenutnega, je potrebno izbrati cenovno ugodno vezje FPGA, ki pa je posledično tudi manj zmogljiv. To je glavni razlog, da se za filtriranje uporabi filter tekočega povprečja, ki je najenostavnejši FIR (angl.: Finite impulse response) filter.

V prvem delu magistrske naloge je predstavljen senzor AM4096, vezje FPGA, ki skrbi za filtriranje signala ter nekaj osnovnih podatkov o digitalnih filtrih in njihovem delovanju.

V drugem delu je predstavljena teoretična rešitev filtriranja. Pred izvedbo naloge je bilo najprej potrebno v teoriji preveriti, če je s šumom, ki je prisoten ob

prehodih pozicij, možno določiti nove vmesne pozicije in s tem višjo ločljivost senzorja.

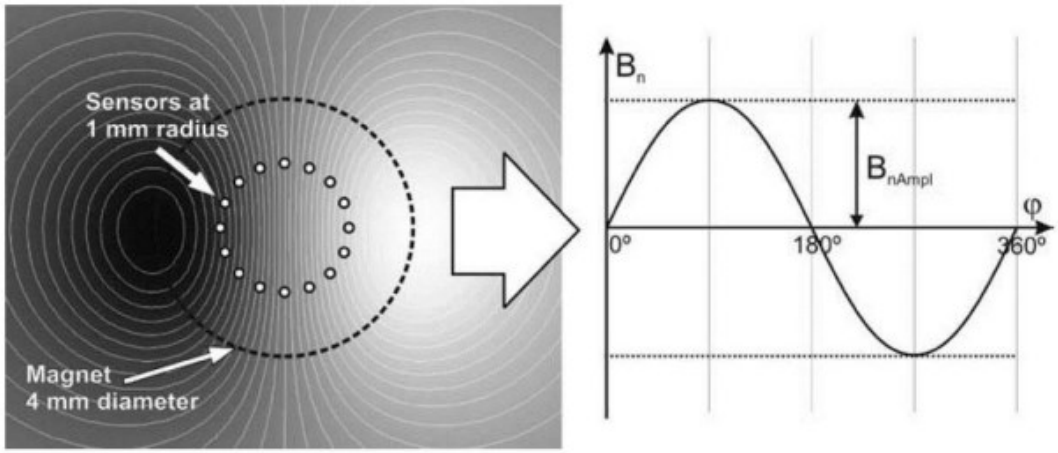
V tretjem delu je opisano, kako sem v realnosti implementiral rešitev, ki sem jo predstavil v drugem delu. Najprej sem z opisnim jezikom Verilog napisal programsko kodo v programskem okolju Vivado ter preveril delovanje s simulacijo, za katero sem z logičnim analizatorjem zajel realne podatke iz senzorja AM4096. Ko je simulacija dala željene rezultate, sem program implementiral na vezju FPGA. V realnem času sem A in B signala pošiljal v vezje FPGA, ga filtriral in z logičnim analizatorjem bral na izhodih vezja FPGA. Določiti sem moral še najbolj primerne parametre filtra, kar sem naredil s skripti, spisanimi v programskem jeziku Python. Z njimi sem preko serijske komunikacije spreminjal parametre filtra, implementiranega v vezju FPGA ter upravljal motor, ki skrbi za vrtenje magneta, da sem lahko delovanje filtra preveril v različnih razmerah.

V zaključku sem podal rešitve in ugotovitve za filtriranje šuma na magnetnem senzorju AM4096 in predstavil načrte za nadaljnje delo. Delo, opravljeno v okviru magistrske naloge, je namreč le del končne izvedbe novega, nadgrajenega izdelka.

2 Teoretično ozadje

2.1 AM4096

AM4096 je senzor, ki zaznava kotno pozicijo cilindričnega magneta, ki je diametralno polariziran in leži od 0,5 mm do 1,5 mm nad vezjem. Hallovi senzorji, ki so cilindrično razporejeni okoli središča integriranega vezja, zaznavajo pravokotne komponente magnetnega polje ter se prezentirajo z napetostjo. Signali so potem med seboj sešteti in nato ojačani. Sinus in kosinus, ki sta za optimalno delovanje tovarniško kalibrirana, se generirata, ko se magnet nad vezjem vrti. 12-bitni interpolator v integriranem vezju poskrbi za seštevek sinusa in kosinusa, kar predstavlja kotno pozicijo. Izhodni signali so lahko v analognih in različnih digitalnih formatih. Za potrebe magistrske naloge sem uporabljal le inkrementalna signala A in B.

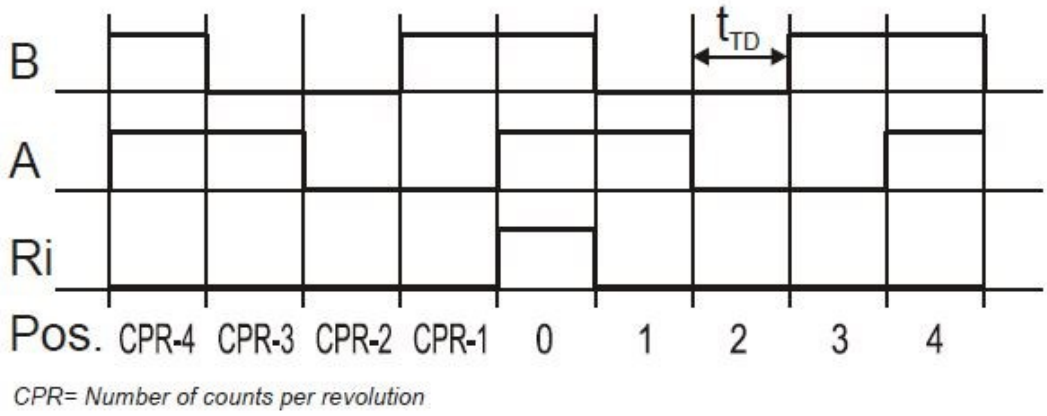


Slika 2.1: Razporeditev magnetnega polja in njegova modulacija ob rotaciji magneta za 360 stopinj [1]

2.1.1 Inkrementalni signali

Eden izmed možnih izhodov iz AM4096 so trije inkrementalni signali: A, B in Ri.

Relativna pozicija je kodirana s signaloma A in B, ki sta med seboj zamaknjena za 90 stopinj. Signala A in B dajeta štiri možne kombinacije izhodov, iz katerih se dekodira smer vrtenja. Če se magnet vrti v smeri urinega kazalca, potem signal B fazno prehiteva A. Absolutno pozicijo dobimo z referenčnim signalom Ri, ki določi pozicijo 0, kot je prikazano na sliki 2.2. Pojavi se vsakič, ko magnet opiše 360 stopinj.



Slika 2.2: Inkrementalni A B signal [1]

2.2 Vezje FPGA

Vezje FPGA je programirljiv digitalni logični čip. Glavna razlika med mikrokontrolerjem in vezjem FPGA je ta, da se v prvega le zapiše programska koda, medtem ko pri vezju FPGA sami oblikujemo vezje. Mikrokontroler hrani program v bliskovnem spominu (angl.: flash memory), dokler ta ni izbrisan ali zamenjan.

V vezju FPGA naredimo vezje, saj ne vsebuje procesorja, na katerem bi lahko hranili programsko kodo. Lahko oblikujemo tako povsem preprosta vezja kot tudi večjedrne procesorje. Naprednejša vezja FPGA poleg osnovnih gradnikov (vhodno izhodnih enot, logičnih vrat in stikal) lahko vsebujejo tudi fazno ujete zanke (PLL),

enote za digitalno procesiranje signalov (DSP), pomnilnik (RAM) in druge. Zadnje skrbijo za hranjenje konfiguracije čipa in se ob prekinitvi napajanja izgubijo, kar pomeni, da je treba čip konfigurirati vsakič, ko priklopimo napajanje.

V primerjavi z mikrokontrolerjem je vezje FPGA veliko hitrejše, saj se vsi procesi v njem izvajajo vzporedno.

2.2.1 Konfiguriranje vezja FPGA

Kot sem že omenil, se na vezje ne naloži programa, ki se potem izvaja, temveč je potrebno vezje FPGA konfigurirati. Med vhode in izhode postavimo logična vrata, ki jih med seboj povežemo s stikali, tako da dosežemo želeno delovanje vezja.

To lahko storimo s shematskim načrtovanjem, ki je veliko bolj zamudno in se uporablja manj, ali pa s strojnim opisnim jezikom HDL (angl.: Hardware description language). Najbolj razširjena jezika sta VHDL in Verilog, ki sem ga uporabil tudi sam. Z izbranim jezikom opišemo delovanje vezja s programom, ki ga potem sintetiziramo v ustrezno vezje. Vezje je potrebno preveriti tudi na realnih podatkih, kar storimo s simulacijo. Ko simulacija deluje tako, kot si želimo, sintetizirano vezje z implementacijo prevedemo v obliko, primerno za izbrano vezje FPGA. V naslednjem koraku naredimo .bin datoteko, ki jo lahko naložimo na vezje FPGA. Po uspešno opravljenih zgoraj opisanih korakih (pisanju programske kode, sintezi, simulaciji, implementaciji, narejeni .bin datoteki) pričakujemo, da se bo konfigurirano vezje obnašalo tako kot pri simulaciji, vendar v veliki večini primerov temu ni tako in je potrebno več ponovitev vseh zgoraj opisanih korakov, da odpravimo vse napake.

2.3 Digitalni filtri

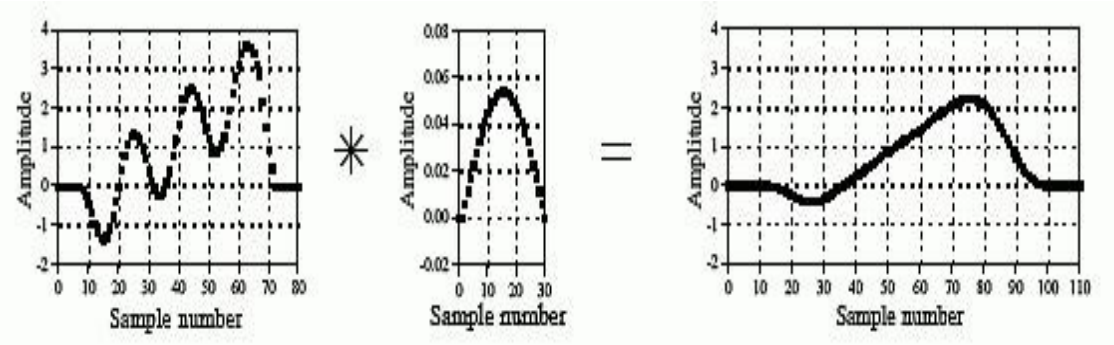
Za filtriranje vhodnega digitalnega signala se uporablja digitalne filtre [2]. Za optimalno delovanje filtra je potrebno izbrati ustrezno jedro, v nadaljevanju kernel Enačba (2.1). predstavlja odziv vhodnega signala na filter rezultat konvolucije vhodnega signala in kernela.

H(k) je kernel, *x(k)* vhodni signal in *y(k)* filtriran izhodni signal [3]. Primer konvolucije prikazuje slika 2.3.

$$y(k) = h(k) * x(k) \tag{2.1}$$

$$y[k] = \sum_{i=0}^{N} b_i x[k-i] - \sum_{i=1}^{M} a_i y[k-i] \tag{2.2}$$

Enačba (2.2) je splošna enačba za digitalne filtre, kjer *y[k]* predstavlja izhodni vzorec, ki je linearna kombinacija trenutnega vhodnega vzorca *x[k]* in končnega števila preteklih vhodnih *x[k-i]* in izhodnih vzorcev *y[k-i]*. *N* je število preteklih vhodnih, *M* pa izhodnih vzorcev.



Slika 2.3: Konvolucija vhodnega signala in filtra [2]

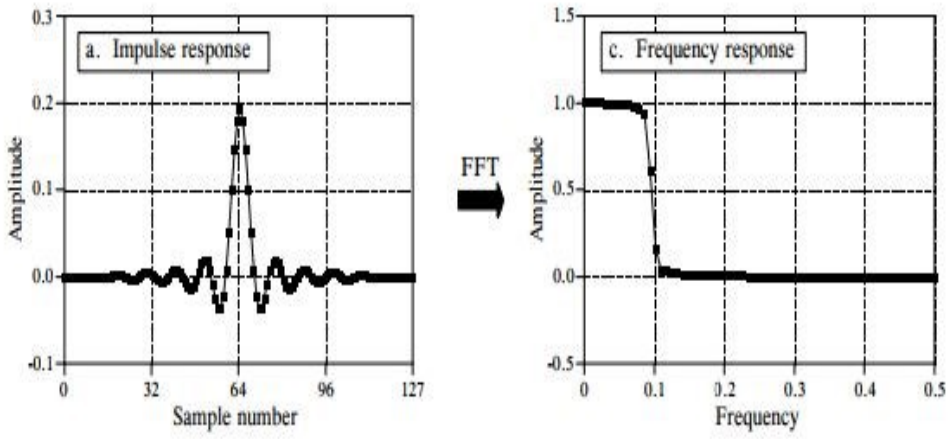
Glavna naloga filtra je prepuščanje določenih frekvenc in slabljenje drugih, s čimer signal ločimo od šuma. Filtri se uporabljajo tudi za obnavljanje, izboljšavo zajetega signala. Kot primer lahko navedem fotografijo, ki je bila zajeta ob

premikajoči se kameri ali pa z neprimernimi lečami, kar privede do zamegljenosti. V obeh primerih je potrebno sliko obnoviti. V primerjavi z analognimi filtri se da z digitalnimi doseči zahtevnejša filtriranja, imajo pa to slabost, da se z višanjem reda pojavlja tudi večja zakasnitev. Filtre lahko prikažemo s prenosno funkcijo H(z) (2.3), ki jo dobimo s transformacijo Z enačbe (2.2) ali pa frekvenčnim odzivom, ki ga dosežemo s Furierjevo transformacijo prenosne funkcije, kar ponazarja enačba (2.4), oziroma v diskretnem frekvenčnem prostoru enačba (2.5), kar vidimo na sliki 2.4.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N} b_i z^{-i}}{\sum_{i=1}^{M} a_i z^{-i}} \tag{2.3}$$

$$H(\omega) = \int_{\tau=-\infty}^{\infty} h(\tau) e^{-i\omega t} d\tau \tag{2.4}$$

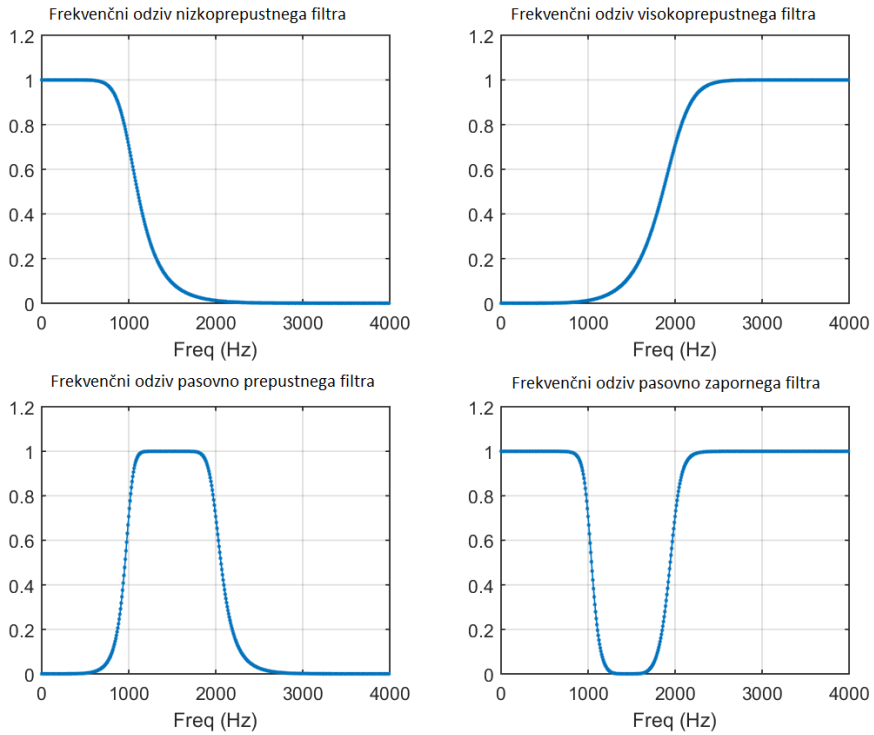
$$H(\Omega) = \sum_{k=-\infty}^{\infty} h(k) e^{-i\Omega k} \tag{2.5}$$



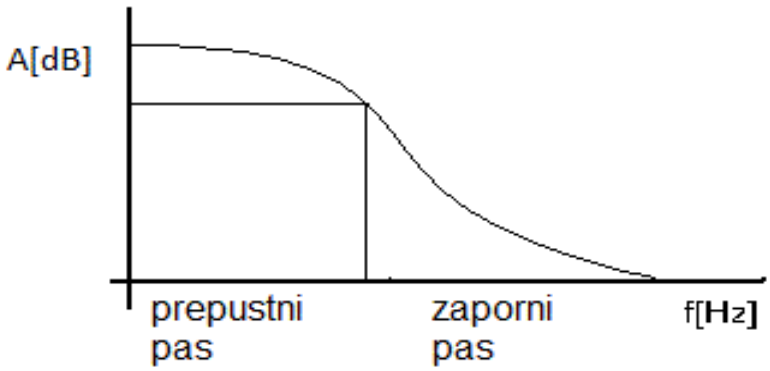
Slika 2.4: Pretvorba kernela v prenosno funkcijo [2]

Digitalne filtre se lahko načrtuje kot nizko prepustne, s katerimi se odrežejo frekvence višjih redov. Po potrebi jih potem s frekvenčnimi transformacijami lahko pretvorimo v pasovno prepustne, pasovno zaporne in visoko prepustne, slika 2.5.

Njihovo obnašanje je določeno z razmerjem med mejno frekvenco in polovico frekvence vzorčenja, ki ji pravimo Nyquistova frekvenca.



Slika 2.5: Vrste filtrov [4]

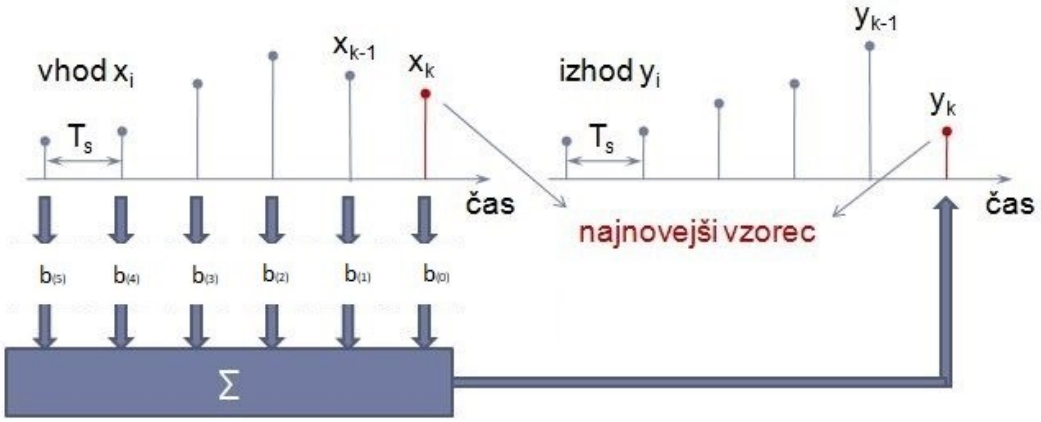


Slika 2.6: Pasovi v filtru

Digitalne filtre glede na trajanje impulznega odziva ločimo dva tipa: FIR in IIR (angl.: Infinite impulse response) [5].

2.3.1 Filter FIR

FIR filtri imajo končen impulzni odziv ter jih lahko imenujemo tudi nerekurzivni filtri [6]. Tako jih imenujemo, ker nimajo povratne zanke in so tako odvisni le od vhodnih podatkov. V primerjavi z enačbo (2.2) za enačbo filtra FIR (2.6) odpadejo členi za povratno zanko. Za njih je značilno, da so vedno stabilni in imajo linearni fazni potek, zaradi česar ne vnašajo faznih popačenj. Dolžina odziva je za eno večja od dolžine reda filtra.

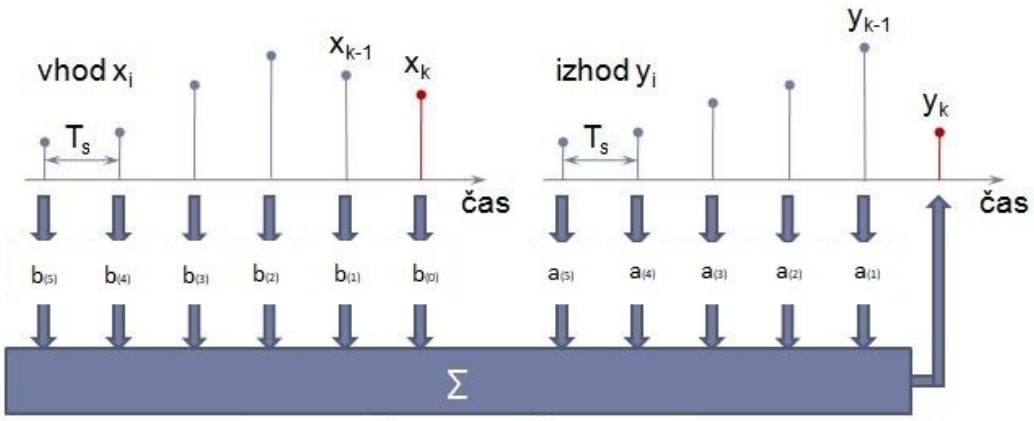


Slika 2.7: FIR filter

$$y_k = \sum_{i=0}^{N} b_i x_{k-i} \tag{2.6}$$

2.3.2 Filter IIR

Za filtre IIR je značilno, da je njihov izhod, poleg trenutnega vhodnega vzorca, odvisen tudi od enega ali več prejšnjih izhodnih vrednosti, kar vidimo v enačbi (2.2) [6].



Slika 2.8: IIR filter

Njihova težava je, da ob napačni izbiri koeficientov postanejo nestabilni, kar je zelo moteče pri sistemih, kjer se koeficienti spreminjajo med samim delovanjem. V primerjavi s filtri FIR nimajo linearnega faznega poteka, ker imajo neskončno dolg odziv h[k], kar ne omogoča simetrije funkcije h[k]. Prednosti filtrov IIR v primerjavi s FIR so v tem, da porabijo veliko manj pomnilniškega prostora in imajo tudi manjšo zakasnitev vhodnega signala. Za enake zahteve filtriranja imajo lahko tudi trikrat nižje stopnje, pri čimer vsaka dodatna stopnja prinaša zakasnitev izhodnega signala za eno periodo.

2.3.3 Tekoče povprečje

Filter tekočega povprečja (angl.: Moving average) je najosnovnejša oblika filtra FIR, ki filtrira preko zadnjih N+1 vzorcev (2.7).

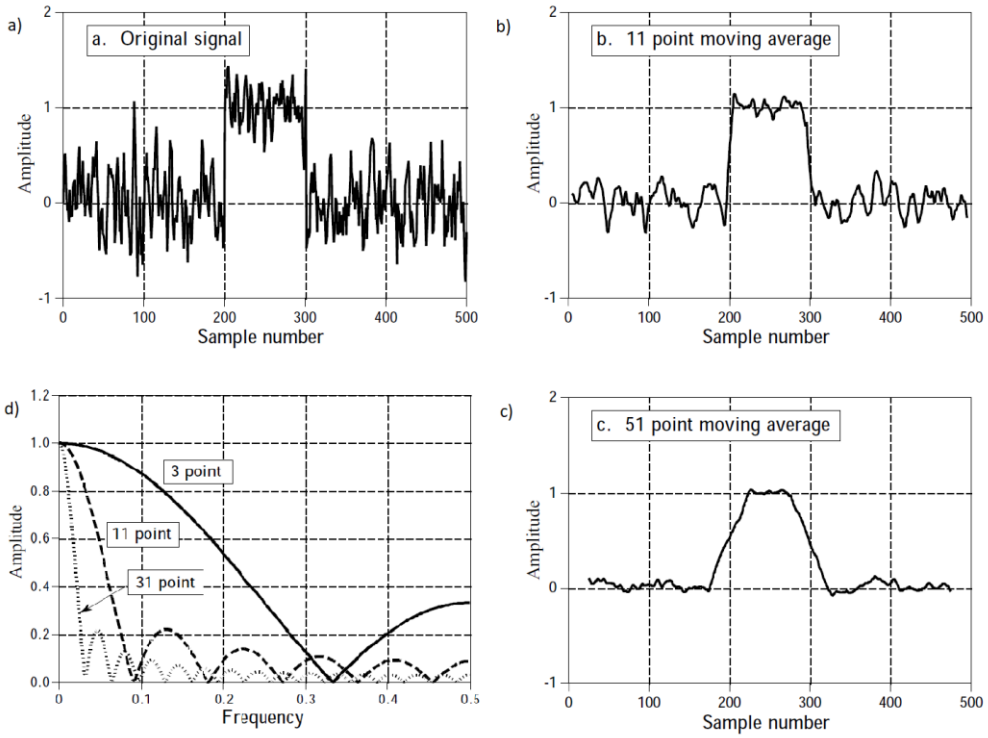
$$H(z) = \frac{1}{N+1} \sum_{i=0}^{N} z^{-n} \tag{2.7}$$

S filtriranjem poteka konvolucija signala s kernelom, ki je v primeru tekočega povprečja enak pravokotnemu oknu, ki ima površino 1. V primeru povprečja petih elementom je kernel: […0, 0, 1/5, 1/5, 1/5, 1/5, 1/5, 0, 0, …]. Dobra lastnost filtra je odpravljanje belega šuma in pa vseeno ohranjanje ostrih prehodov. Z večanjem števila vzorcev se odstrani več šuma, ob tem pa se izgublja ostrina robov.

Frekvenčni odziv filtra tekočega povprečja je opisan v enačbi (2.8), kjer je f frekvenca vzorčenja.

$$H[f] = \frac{\sin(\pi f N)}{\mathrm{M}\,\sin(\pi f)} \tag{2.8}$$

Znano je, da filtri, ki imajo dobre lastnosti v časovnem prostoru, navadno dajejo slabše rezultate v frekvenčnem prostoru, kar velja tudi za filter tekočega povprečja. Z večanjem števila vzorcev se namreč tudi manjša prepustni pas. Lahko rečemo, da je tekoče povprečje dober filter za glajenje signala, vendar slab za prepuščanje nizkih frekvenc. Na sliki 2.9 lahko vidimo, kako se s filtriranjem signala, prikazanega na primeru a, z večanjem števila vzorcev na primerih b in c signal gladi in tudi brišejo ostri robovi. Na sliki 2.9, primer d, lahko vidimo, da se z večanjem števila vzorcev oža prepustni pas. Zaporni in prepustni pas sta prikazana na sliki 2.6.



Slika 2.9: Odzivi signalov na filter tekočega povprečja [2]

3 Teoretična rešitev

3.1 Uvod

Glede na podatkovni list senzorja AM4096, mora enkoder delovati do frekvence vrtenja 500 Hz pri 12-bitni ločljivosti. To pomeni 4096 prehodov vsaki dve milisekundi. V primeru inkrementalnega izhodnega AB signala je za takšno hitrost vrtenja frekvenca preklapljanja izhodov A in B enaka 2.048 MHz. Glede na to, da je frekvenca osveževanja v čipu 10 MHz, lahko pride do približno 5 dodatnih preklopov med dvema realnima stanjema zaradi šuma. Pri nižji hitrosti vrtenja je teh neželenih prehodov zaradi šuma proporcionalno več.

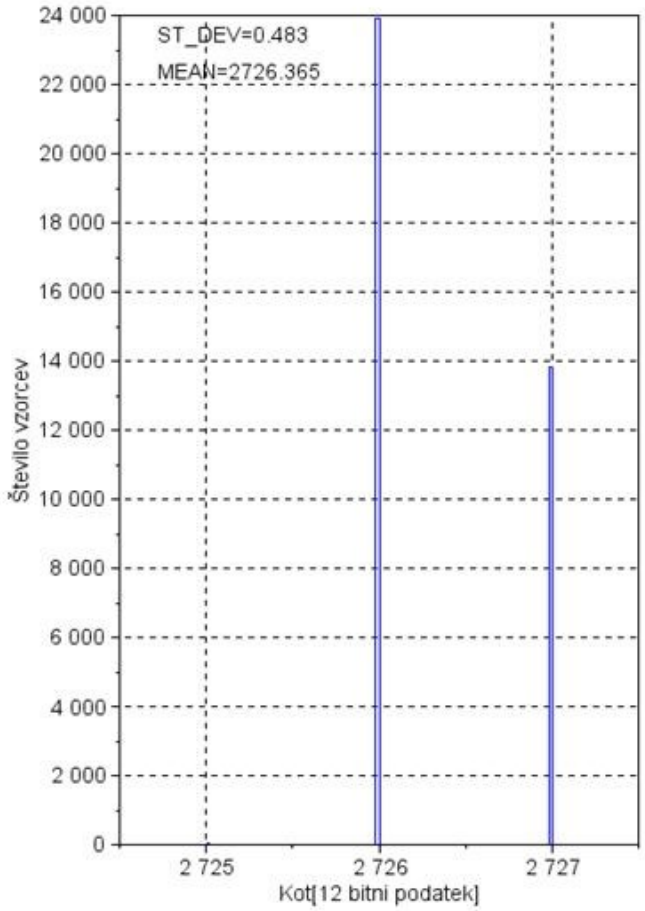
Trenutno je zaradi šuma vključena dodatna histereza, ki preprečuje te prehode med stanji zaradi šuma. Če želimo dobiti en ali dva dodatna bita, lahko informacijo o kotu, ki je "umazana" s šumom, povprečimo. Na ta način zmanjšamo šum, hkrati pa zmanjšamo pasovno širino samega enkoderja.

3.1.1 Tekoče povprečje (moving average)

To je najbolj enostaven algoritem za računanje povprečne vrednosti N vzorcev in hkrati tudi zelo primeren način za implementacijo v FPGA. Vsi ostali filtri bi namreč porabili veliko več logičnih enot in gradnikov, kar bi povišalo njegovo ceno, kar pa ni želja podjetja. Natančnejši opis filtriranja je opisan v poglavju 2.3.3.

3.2 Meritev šuma in povprečje

Informacijo o kotu sem zajemal preko protokola I^2C. Pred tem sem histerezo nastavil na 0, da je bil šum prisoten. Pri nespremenjenem kotu, v mirovanju, sem zajel 30.000 vzorcev, ki sem jih obdelal v programskem jeziku Matlab. Na sliki 3.1 je prikazano, da se vrednosti kota gibljejo med vrednostmi 2725, 2726 in 2727, za ±1 bit.

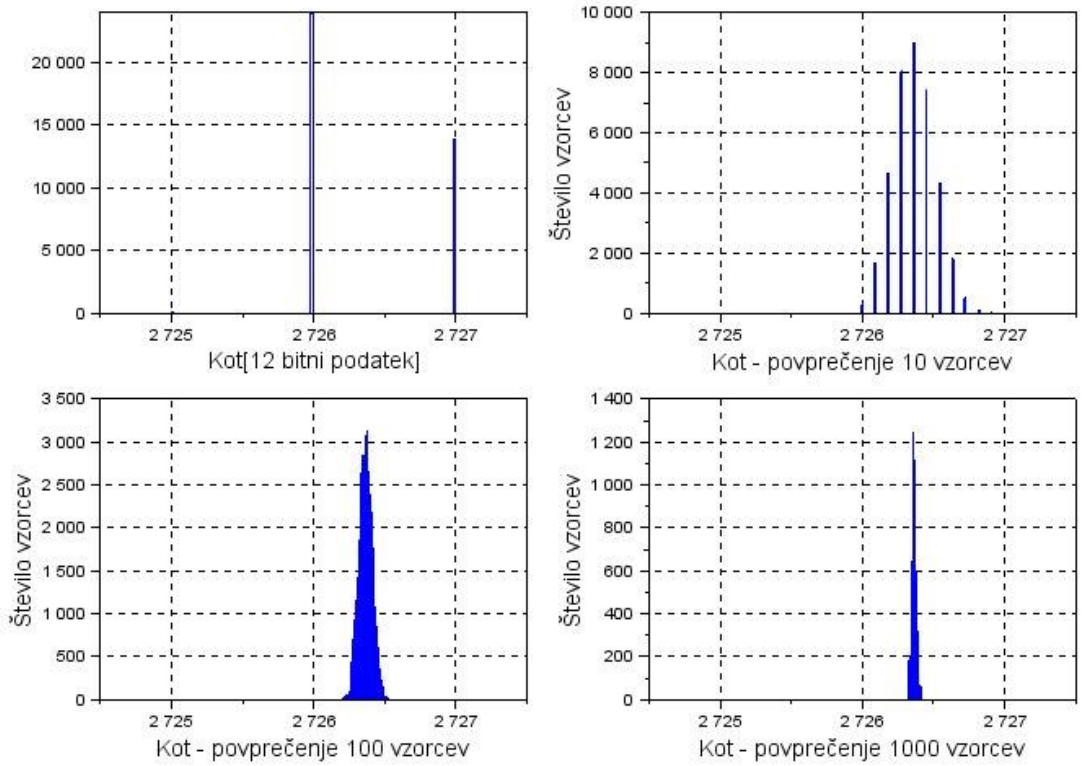


Slika 3.1: Vrednosti kota v mirovanju

Nato sem izračunal standardno deviacijo po tekočem povprečju z različnim številom vzorcev 3.1 in izrisal vrednosti kota, slika 3.2.

Število vzorcev	Standardna deviacija
1	0.4833
10	0.1469
100	0.0482
1000	0.0152

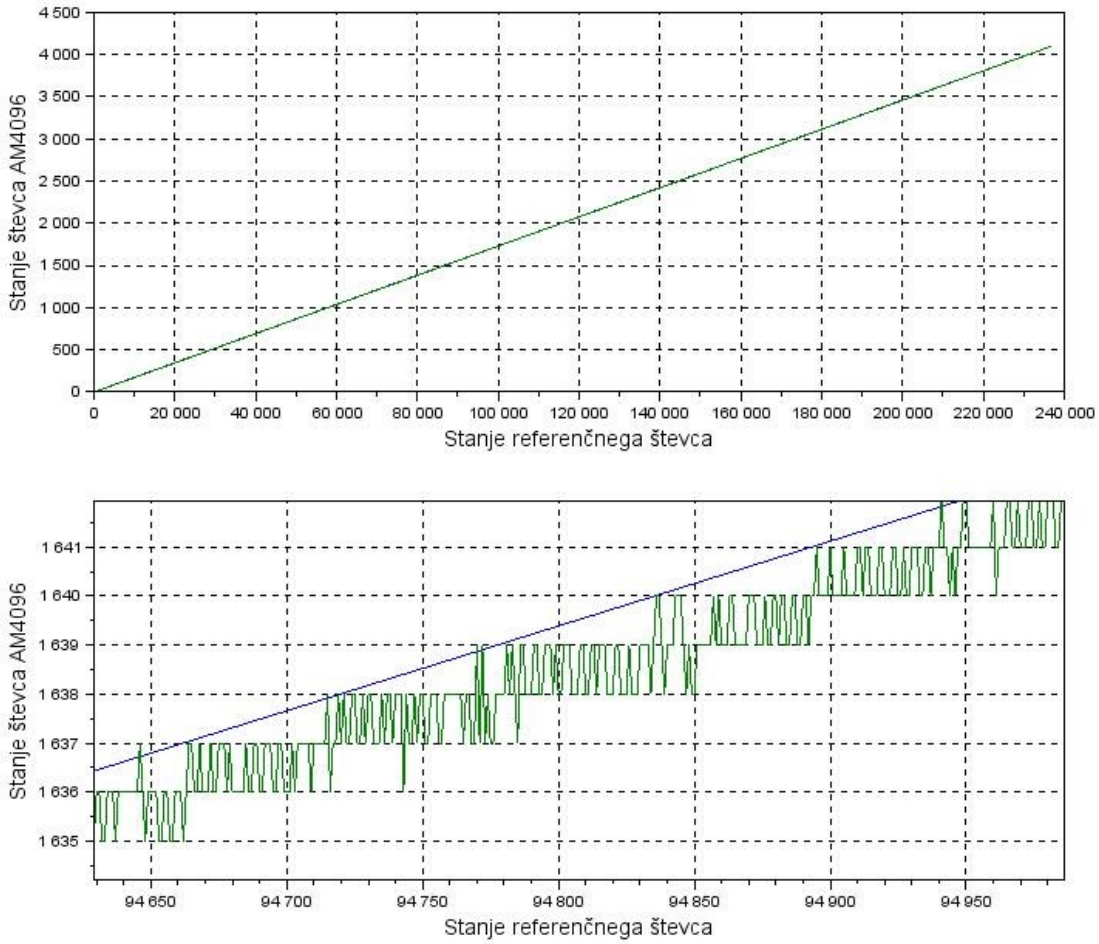
Tabela 3.1: Standardna deviacija ob različnem številu vzorcev



Slika 3.2: Primerjava porazdelitve vrednosti kota pred in po povprečenju

3.3 Meritev celotnega kroga z referenčnim enkoderjem

Posnel sem celotni krog z referenčnim enkoderjem ter AM4096, ki z 12-bitno ločljivostjo zazna 4096 sprememb, medtem ko jih referenčni enkoder zazna 236800. V idealnih razmerah bi tako AM4096 prehod zaznal na vsakih 59 stanj referenčnega enkoderja. Na sliki 3.3, v prvem grafu, je prikazana idealna vrednost kota, v drugem pa so poleg idealne vrednosti še neželeni prehodi zaradi šuma (z zeleno barvo) .



Slika 3.3: Idealne vrednosti kota in realne vrednosti

3.4 Sklepi zgornjih rezultatov

Glede na zgornje rezultate se vidi, da se s povprečjem lahko pridobi dodatno ločljivost. V primeru večjega števila vzorcev so rezultati bolj točni, imajo manjšo standardno deviacijo, rabimo pa filter višje stopnje. Glede na teoretične rezultate lahko z gotovostjo trdim, da bi se dalo dobiti dodaten bit, medtem ko je bila želja pridobitev dveh dodatnih bitov in s tem še višje ločljivosti. Celoten postopek filtriranja je tako stremel k pridobitvi dveh bitov, kar je podrobneje predstavljeno v naslednjem poglavju.

4 Zasnova in izvedba vezja

Za izvedbo sem uporabil več programskih okolij in naprav. Realne podatke sem moral uvoziti v simulacijski program, s katerim sem preveril delovanje filtra, ki sem ga spisal s programskim jezikom Verilog. Po želenem delovanju simulacije sem filter implementiral na vezju FPGA in v realnem času testiral njegovo delovanje, kjer se je pojavilo kar nekaj težav. Njihovo odpravljanje je bilo precej zamudno, saj je bilo potrebno prvotno izvedbo filtriranja prilagoditi. Pri prvi izvedbi sem namreč filtriral pozicije in ne njenih diferenc, zato se je pojavil preliv (angl.: Overflow), ko je magnet opisal 360 stopinj.

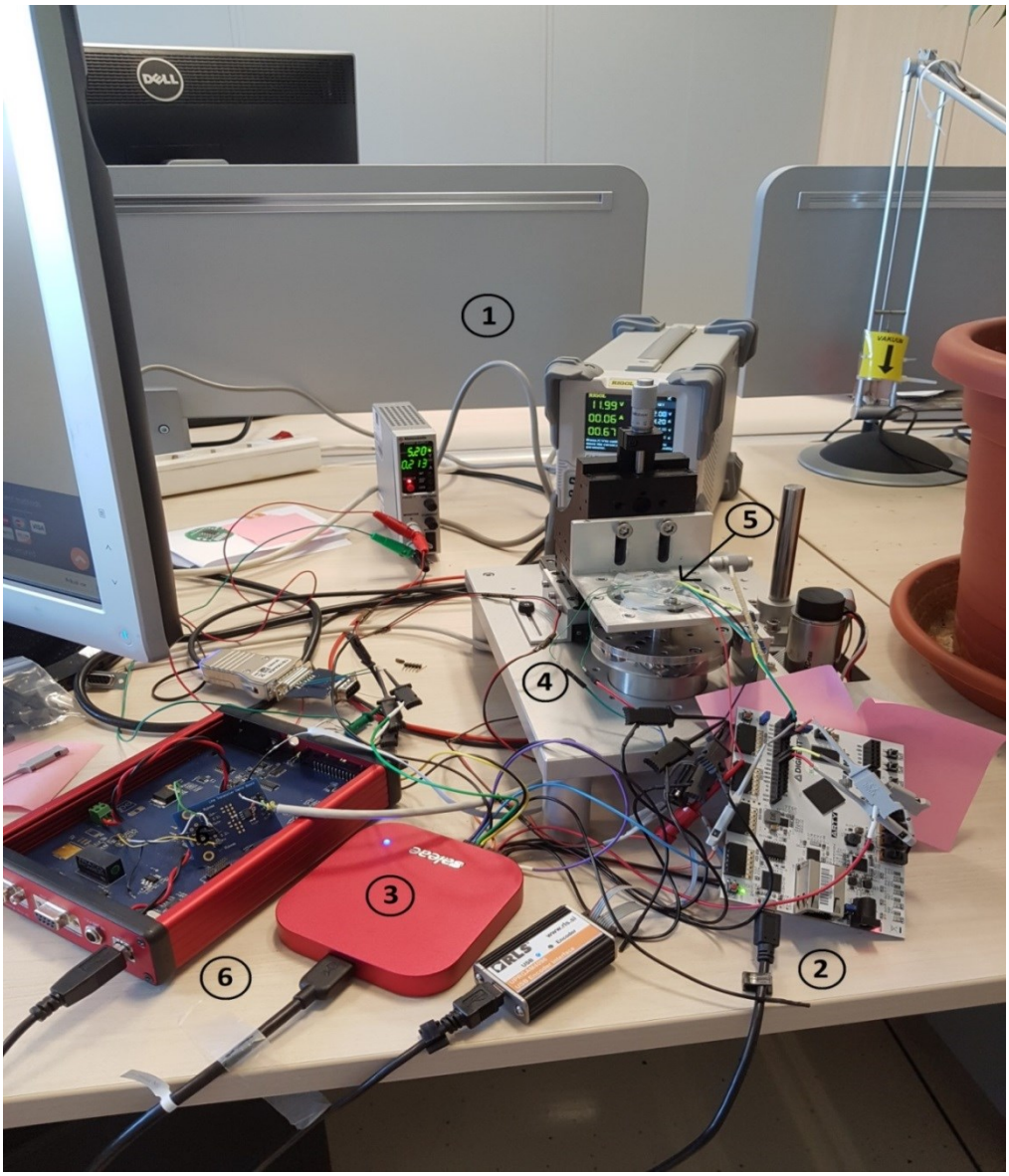
4.1 Uporabljeni programski jeziki in naprave

Pri nalogi sem uporabljal naslednje programe in naprave:

1) Programski jeziki:
   a) Python: uporabljal sem ga za upravljanje motorja, ki vrti magnet, testiranje parametrov filtra ter njihovo spreminjanje in pošiljanje v FPGA;
   b) Matlab: uporabljal sem ga pri izvedbi teoretične rešitve, opisane v prejšnjem poglavju in za analizo parametrov filtra;
   c) Verilog: uporabljal sem ga v programskem okolju Vivado za pisanje opisne programske koda za FPGA;
   d) Excel: uporabljal sem ga za shranjevanje podatkov ob testiranju in simulaciji.

2) Naprave:
   a) Dva napajalnika (številka 1 na sliki 4.1);
   b) Razvojna plošča Arty z FPGA-jem Artix 7 (številka 2 na sliki 4.1);

c) Logični analizator Saleae (številka 3 na sliki 4.1);
d) Priprava podjetja RLS, z motorjem, magnetom, stojalom za enkoder ter referenčnim Renishaw enkoderjem ″TONiC″ in povezavo ″TONiC TD interface″ (številka 4 na sliki 4.1);
e) Inkrementalni enkoder podjetja RLS, RMB20IC, s senzorjem AM4096 (številka 5 na sliki 4.1);
f) Programatorski vmesnik podjetja RLS, UPRGAM4096, ki deluje s protokolom I^2C, za programiranje senzorja AM4096 (številka 6 na sliki 4.1).



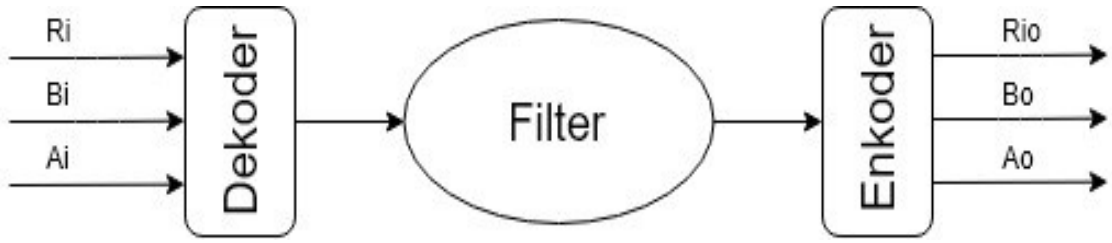
Slika 4.1: Naprave

4.2 Filtriranje

Tehnično rešitev, ki sem jo opisal v poglavju 3, sem pisal z opisnim jezikom Verilog. Na sliki 4.2 je predstavljena blokovna shema modulov v FPGA-ju. Poleg spodaj narisanih modulov sem uporabljal še modul za komunikacijo med računalnikom in FPGA, kar mi je koristilo ob testiranju in spreminjanju parametrov filtra.

Implementirano filtriranje na FPGA deluje na frekvenci 50 MHz, medtem ko se v AM4096 najhitrejši prehodi stanj dogajajo s frekvenco 4 MHz. To zadostuje Nquistovemu pogoju, pri katerem je treba vzorčiti najmanj z dvakratno frekvenco signala.

S filtriranjem vhodnih signalov sem dodal dodatna dva bita in odstranil šum. Za implementacijo sem si izbral filter tekočega povprečja, ki je za implementacijo v FPGA zelo primeren; z implementacijo naprednejšega filtra bi namreč porabil veliko več enot, kar pa posledično pomeni tudi dražji FPGA čip. Glede na to, da se bo nov enkoder prodajal v zelo velikih količinah, to ne bi bilo primerno.



Slika 4.2: Blokovna shema delovanja FPGA

V FPGA prihajajo trije signali: Ai, Bi in Zi. Na vsako pozitivno spremembo ure (angl.: Clock) sem prebral vrednosti signalov A in B ter si jih zapomnil do naslednje spremembe, kjer sem novi vhodni vrednosti prepisal čez stari. V primeru spremembe signala sem pogledal v tabelo 4.1 in določil, ali se magnet vrti v pozitivni smeri, smeri urinega kazalca ali obratni. Če se sprememba ni pojavila, sem vrednost postavil na 0. S tem sem dobil odvod pozicije (4.1).

$$diferenca = \frac{d}{dt}x \tag{4.1}$$

Kakor sem omenil že na začetku tega poglavja, je bilo sprva filtriranje implementirano na absolutnih pozicijah in ne njenih odvodih. To je vodilo do preliva. Ob prehodu magneta za 360 stopinj je filter tekočega povprečja na absolutnih pozicijah odpovedal, saj je zgladil prehod med zadnjo pozicijo kroga in prvo. V primeru, da bi povprečil na petih vzorcih, bi to na primer pomenilo:

16384/5 + 16384/5 + 16383/5 + 1/5 + 1/5 = 98308. Pozicija 98308 seveda ni realen odraz pozicije enkoderja, zato sem moral filtriranje izvesti na diferencah pozicij.

$\mathbf{A_{star}}$	$\mathbf{B_{star}}$	$\mathbf{A_{nov}}$	$\mathbf{B_{nov}}$	Vrednost
0	0	0	1	1
0	1	1	1	1
1	1	1	0	1
1	0	0	0	1
0	0	1	0	-1
0	1	0	0	-1
1	1	0	1	-1
1	0	1	1	-1

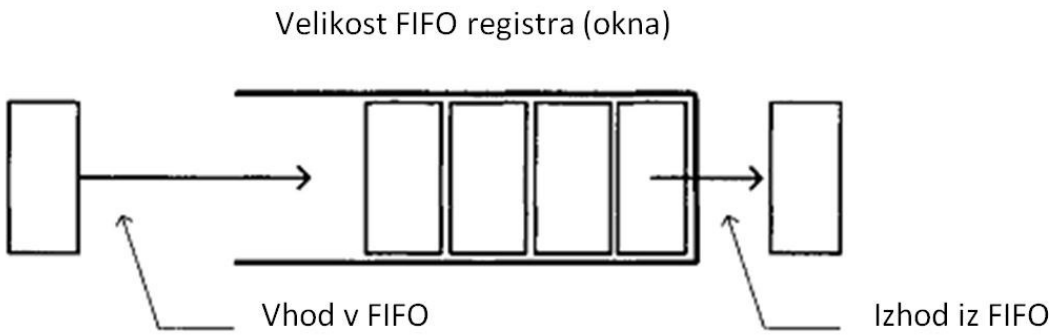
Tabela 4.1: Možne spremembe signala

Filtriranje sem opravil s filtrom tekočega povprečja, ki je FIR filter in je zelo primeren za uporabo v FPGA. Glede na to, da je filtriranje konvolucija vhodnega signala in odziva filtra, sem lahko uporabil lastnost konvolucije, ki jo prikazuje enačba (4.2).

$$\frac{d}{dt} g * f = \frac{d}{dt} (g * f) \tag{4.2}$$

Zaradi enačbe (4.2) je bilo potrebno izhodni filtriran signal še integrirati.

V realnosti sem filtriranje izvedel s FIFO registrom, v katerega sem shranjeval spremembe pozicije. FIFO register deluje tako, da element, ki se zapiše na vhodu registra, lahko prebereš na izhodu, ko je register poln, to pomeni z zakasnitvijo toliko urinih ciklov, kolikor je velikost registra.



Slika 4.3: FIFO register

Ko je bil register poln, sem seštevku diferenc v registru z vsakim ciklom prištel novo vhodno vrednost, ki je prišla v register in odštel zadnjo vrednost, ki je šla iz registra ven. V naslednjem koraku sem vrednosti integriral med sabo in tako dobil rekonstruiran signal. Temu signalu sem dodal še histerezo, da sem še malo bolj odpravil šum.

Želja je bila, da bi pridobil dva dodatna bita, kar pomeni povečanje natančnosti za faktor 4. Zaradi tega sem integrirano vrednost delil z velikostjo okna/4. Uporabljen filter je bil nenormiran, da je bil lažji za implementacijo in da sem sploh lahko dobil dodatno ločljivost. Za normiran filter bi integrirano vrednost delil samo z velikostjo okna.

Dobljene vrednosti sem ponovno kodiral v inkrementalni signal s pomočjo tabele 4.1.

4.3 Testiranje

Za filter, implementiran v FPGA, je bilo potrebno nastaviti najbolj primerne parametre. Glavna parametra, ki ju je bilo potrebno določiti, sta bila velikost FIFO registra in histereza, ki pa ima na rezultate manjši vpliv. Postopek testiranja sem izvedel s pomočjo python skript, ki so nadzirale smer in hitrost vrtenja magneta ter pošiljale različne parametre filtra v FPGA. Podatke sem bral z logičnim analizatorjem Saleae, in sicer vhodni signal, filtriran izhodni signal z dodanima bitoma in inkrementalni signal referenčnega enkoderja. Shranjene podatke sem nato analiziral na podlagi več parametrov.

Odpravo šuma sem preveril z beleženjem števila prehodov stanj v primerjavi z referenčnim enkoderjem. Glede na to, da referenčni enkoder zazna 1184000 prehodov, AM4096 pa naj bi jih po dodanih 2 bitih zaznal 16384, bi se v optimalnih razmerah na vsak preklop AM4096 zgodilo 72 preklopov referenčnega enkoderja. Uporabljal sem drug referenčni enkoder kot pri teoretični rešitvi, saj je imel 5-krat večjo ločljivost.

Drugi parameter, ki sem ga opazoval, je bila simetričnost signalov. To sem preveril tako, da sem izračunal standardno deviacijo časov preklopov stanj pri izhodnem filtriranem signalu.

Pri hitrem vrtenju so rezultati zelo dobri, malo slabši pa so pri počasnem vrtenju magneta, kjer je prehodov zaradi šuma več, kakor sem opisal že v poglavju 3. Velikost FIFO registra sem spreminjal s korakom 5000. Za vsako spremembo velikosti registra sem spremenil še velikost histereze, in sicer 0, velikost filtra/ 100 in velikost filtra/10.

4.3.1 Mirovanje

Prva stvar, ki sem jo preveril, je bila delovanje filtra v mirovanju. Pomembno je namreč, da signal ne bi odplaval. To se lahko pojavi v primeru napačnih vhodnih signalov ali pa pri napačnem delovanju filtra. V primeru, da vhodni signal vnese še dodaten šum zaradi načina zajema podatkov, ki ni odvisen od delovanja čipa, filter lahko zazna prehode, ki se v realnosti ne pojavljajo. Logični analizator, ki sem ga uporabljal na začetku, ni imel možnosti, da bi nastavil pragovno vrednost, kar je

vnašalo dodatni šum v FPGA. Sprva sem domneval, da filtriranje ne deluje pravilno, vendar sem potem signal preveril še z osciloskopom in picoskopom, ki sta mi dala potrditev, da se ne pojavljajo prepovedani prehodi iz čipa samega, to so prehodi, ki niso navedeni v tabeli 4.1. Težavo sem odpravil tako, da sem signal iz čipa v FPGA peljal še preko preprostega napetostnega delilnika, kar je odpravilo vnos dodatnega šuma v FPGA. Ob zajemu signala sem nato preveril delovanje filtra in ugotovil, da se na izhodu ne pojavlja več kot en prehod, kar sem tudi pričakoval. Na sliki 4.4 sta zgornja signala vhodna in spodnja izhodna. Izhodni signal ima veliko manj šuma kakor vhodni, kar potrjuje, da filter dobro odpravlja šum.

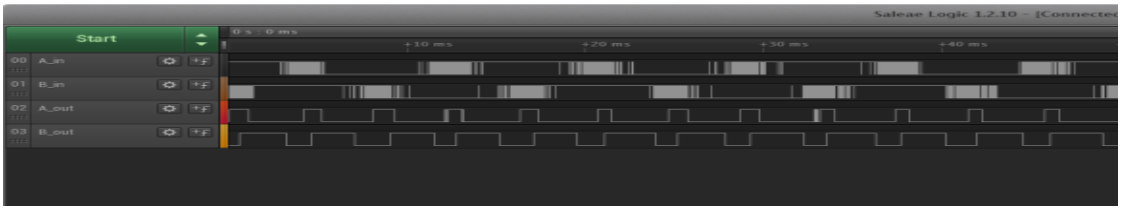
Slika 4.4: Mirovanje

4.3.2 Počasno vrtenje

V naslednjem koraku sem preverjal delovanje filtra pri počasnem vrtenju, pri katerem se pojavlja veliko več šuma kot pri hitrem vrtenju, kar sem že omenil. Na sliki 4.5 je prikazano delovanje filtra pri velikosti FIFO registra 300. Kakor je razvidno s slike, so rezultati povsem neuporabni, saj so vsi prehodi zabrisani.

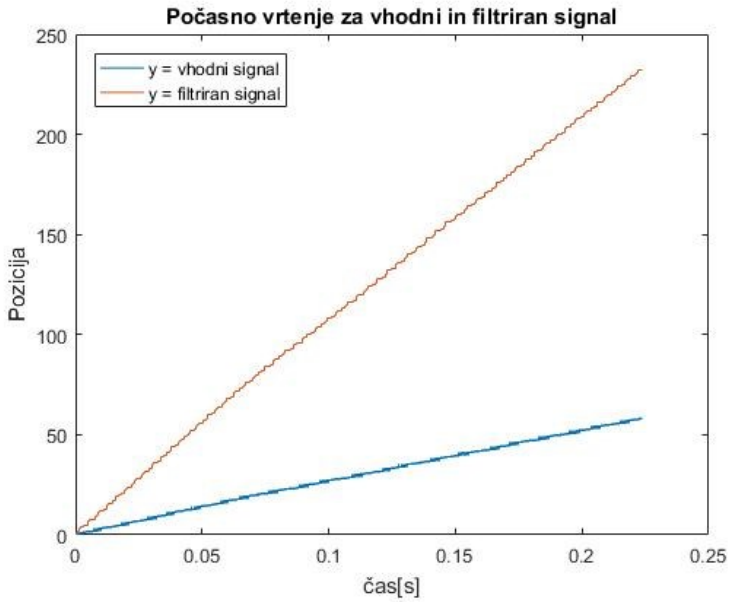
Slika 4.5: Zajem signala pri velikosti filtra 300

S postopkom vrednotenja sem najboljše rezultate dobil pri velikosti FIFO registra 30000 in histereze 300. Na sliki 4.6 so posneti signali pri optimalnih parametrih.

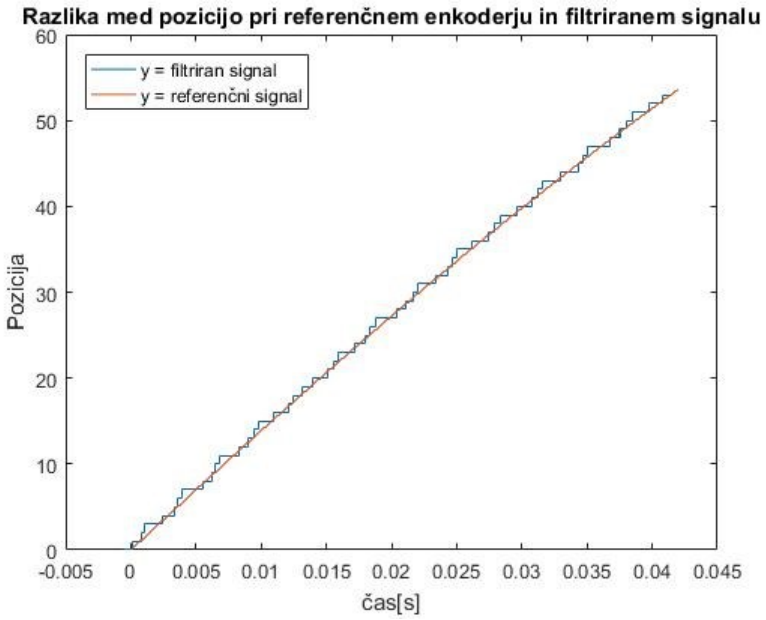


Slika 4.6: Zajem signala pri velikosti filtra 30000 in histerezi 300 za počasno vrtenje

Prisotno je nekaj šuma in nesimetričnost signalov, vendar se z dvema pridobljenima bitoma in uporabo filtra tekočega povprečja ni dalo dobiti boljših rezultatov. Kljub temu so rezultati filtriranja dovolj dobri za uporabo v realnosti. Na slikah 4.7 in 4.8 se lepo vidi, da se je ločljivost in s tem število pozicij povečalo za faktor 4, (slika 4.7), in da pozicija filtriranega signala lepo sledi pozicijam referenčnega enkoderja (slika 4.8).



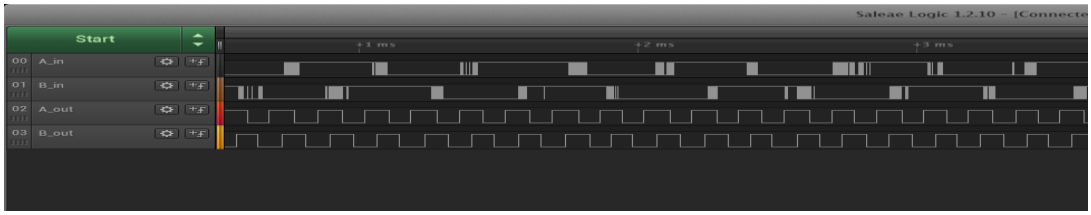
Slika 4.7: Primerjava vhodnega in izhodnega filtriranega signala za počasno vrtenje



Slika 4.8: Primerjava pozicije med referenčnim enkoderjem in filtriranim signalom za počasno vrtenje

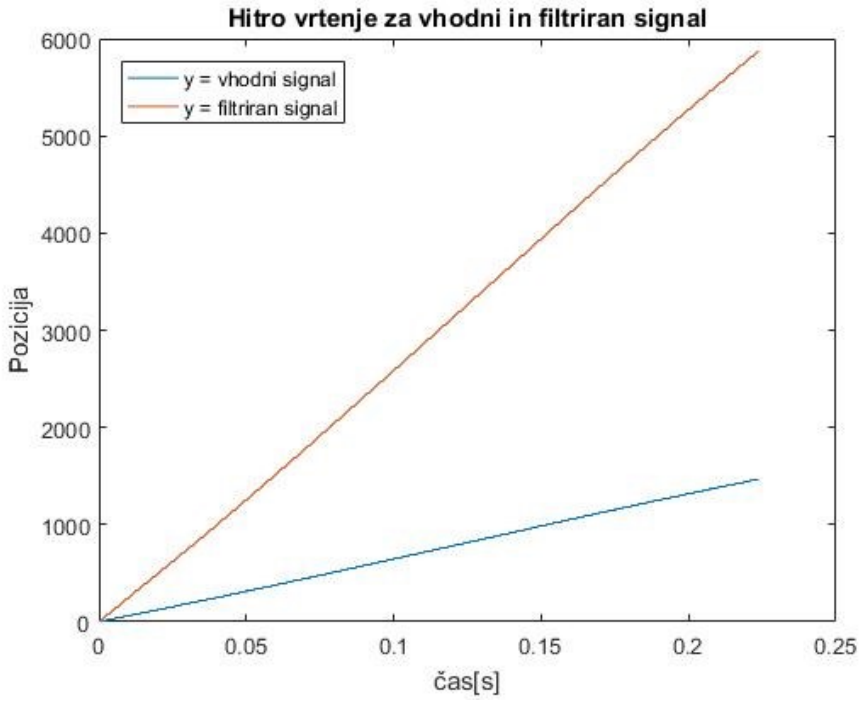
4.3.3 Hitro vrtenje

Pri hitrem vrtenju so rezultati po pričakovanjih veliko boljši kot pri počasnem, kar je tudi razvidno na sliki 4.9. Šum je skoraj povsem odpravljen, poleg tega pa so tudi signali simetrični.

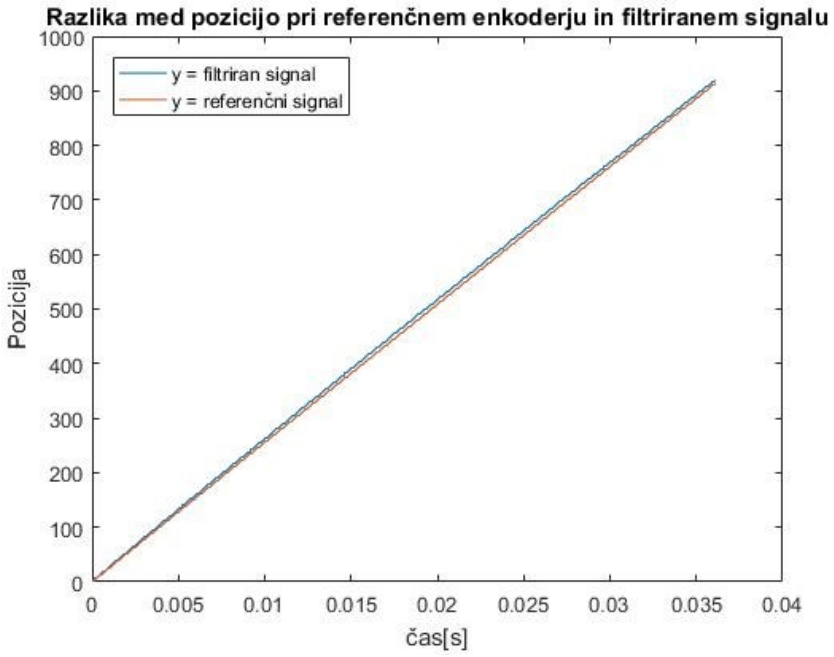


Slika 4.9: Zajem signala pri velikosti filtra 30000 in histerezi 300 za hitro vrtenje

Tudi pri hitrem vrtenju je lepo opazna povečana ločljivost in sledenje pozicije izhodnega signala pozicijam referenčnega enkoderja.



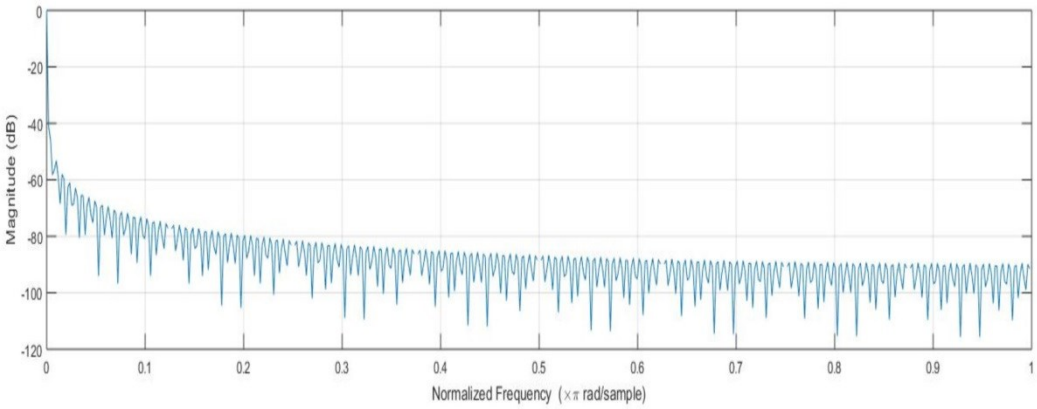
Slika 4.10: Primerjava vhodnega in izhodnega filtriranega signala za hitro vrtenje



Slika 4.11: Primerjava pozicije med referenčnim enkoderjem in filtriranim signalom za hitro vrtenje

4.3.4 Odziv filtra

Z okoljem Matlab sem opravil še analizo filtra. Frekvenčni odziv filtra je prikazan v logaritemski skali in ima enote v decibelih, slika 4.12.



Slika 4.12: Analiza filtra

Zaporna frekvenca se izmeri pri -20 dB. Takrat prepustnost frekvenc pade na 1 %. V mojem primeru filter prepušča signale do frekvenc 25 kHz, kar pomeni, da odreže vse signale, ki imajo frekvenco višjo od te vrednosti.

5 Zaključek

V moji nalogi sem dokazal predvidevanja, da se s filtriranjem lahko doseže povečanje ločljivosti še za dva dodatna bita, kar pomeni, da lahko AM4096 sedaj deluje s 14-bitno ločljivostjo. Za implementacijo filtra tekočega povprečja je bil uporabljen FIFO register, ki je v mojem primeru nastavljen na velikost 30000 vzorcev in histerezo 300. Pri hitrem vrtenju so rezultati zelo dobri, medtem ko je pri počasnem vrtenju še prisotnega nekaj šuma in nesimetričnost. Neželjen učinek pri filtriranju je zamik (angl.: Delay), ki je 30000 urinih ciklov, kar v mojem primeru pomeni 0,6 ms.

Zavedam se, da je ta zamik precej velik in da bi se ga lahko odpravilo na več načinov. Lahko bi se uporabljalo drug način filtriranja, vendar bi to oteževalo implementacijo v vezju FPGA, saj bi zahtevalo zmogljivejši in posledično dražje vezje FPGA. Ta možnost ni izvedljiva, ker se bo vezja FPGA naročalo v velikih količinah in bi to pomenilo prevelik dodaten strošek.

V nadaljevanju bo projekt deležen še precej nadgradenj. Filtriranje bo potrebno implementirati na vezju FPGA proizvajalca Lattice, saj so precej cenejši od proizvajalca Xilinx. To bo zmanjšalo stroške proizvodnje. Glede na ceno bo vezje FPGA tudi manj zmogljivo, zato bodo morda potrebne še dodatne optimizacije v programu. Ena izmed možnosti je, da se bo zmanjšala frekvenca vzorčenja, posledično velikost registra FIFO in s tem tudi zasedenost blokov RAM. To bo odvisno od tega, koliko jih bo na voljo v izbranem čipu. Na enkoderju bo potrebna komunikacija med vezjem FPGA in mikroprocesorjem, ki ga bo ob zagonu tudi sprogramiral, zato bo potrebno pripraviti protokole za komunikacijo med njima. Ena izmed zahtev za nov produkt je tudi izhod v desetiških vrednostih, ki so ljudem

veliko bolj berljive. Izvedljivost bo odvisna od tega, ali bodo na novem čipu prisotni množilniki.

Za potrebe vseh zgoraj opisanih nadgradenj bomo pripravili testno vezje, kjer bomo testirali vse zahteve za nov izdelek. Na matično ploščo bosta nameščeni dve plošči, ena z vezjem FPGA in druga z mikroprocesorjem. Za potrebe testiranj bo na matični plošči moč dostopati do vseh signalov, ki prihajala iz FPGA in mikrokrmilnika in bodo skrbeli za vso komunikacijo, filtriranje in branje ter pošiljanje vseh potrebnih podatkov za delovanje enkoderja.

Ob koncu in vseh delujočih zahtevah bo pripravljeno še končno proizvodno vezje, ki bo v mesecu novembru predstavljeno na sejmu SPS IPS Drives.

Literatura

[1] »Podjetje RLS.« Dosegljivo: https://www.rls.si/am4096-12-bit-rotary-magnetic-encoder-chip. [Dostopano: 1. 9. 2017].

[2] Steven W. Smith, *The Scientist and engineer's Guide to Digital Signal Processing*. California Technical Pub, 1997.

[3] Sašo Tomažič, Savo Leonardis, *Diskretni signali in sistemi*. Ljubljana, 2001.

[4] »Podjetje Mirlab.« Dosegljivo: http://mirlab.org/jang/books/audioSignalProcessing/example/output/butbut05.png. [Dostopano: 1. 9. 2017].

[5] »Podjetje miniDSP.« Dosegljivo: https://www.minidsp.com/applications/dsp-basics/fir-vs-iir-filtering. [Dostopano: 1. 9. 2017]

[6] » Fakulteta za elektrotehniko, Univerza v Ljubljani, prosojnice predavanj iz predmeta: Procesorski sistemi v telekomunikacijah.« Dosegljivo: http://www.fe.uni-lj.si/izobrazevanje/1_stopnja_vss/aplikativna_elektrotehnika/predmeti/2009011311172923/. [Dostopano: 1. 9. 2017].