ANALIZA RAZLIČITIH NAČINA UPRAVLJANJA ASINKRONOG MOTORA

Galić, Mirko

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split / Sveučilište u Splitu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:228:059993

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-09-01**

Repository / Repozitorij:

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij elektroenergetika

Mirko Galić

ZAVRŠNI RAD

ANALIZA RAZLIČITIH NAČINA UPRAVLJANJA ASINKRONOG MOTORA

Split, rujan 2020.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij elektroenergetika

Predmet: Elektroakustika

ZAVRŠNI RAD

Kandidat: Mirko Galić

Naslov rada: ANALIZA RAZLIČITIH NAČINA UPRAVLJANJA ASINKRONOG MOTORA

Mentor: Predrag Đukić

Split, rujan 2020.

Sažetak

Svrha ovog rada je usporedba između različitih načina upravljanja asinkronog motora. Posebna pažnju je posvećena na upotrebi digitalnih filtera koji poboljšavaju i pojednostavljuju izvedbu upravljanja elektromotorima. Digitalni filter *sinc* je implementiran u Verilogu za FPGA integrirane krugove. Opisane su različite metode upravljanja asinkronih motora, dok je naglasak dan na algoritme koji koriste bezsenzorsko vektorsko upravljanje. Upotreba bezsenzorske vrste upravljanja znatno snižava cijenu sustava upravljanja, te omogućuje lakšu primjenu u praksi. Predložena arhitektura koristi FPGA kontroler, čime poboljšava sigurnost i zaštitu, te primjenjuje digitalne filtere.

Implementacija bezsenzorskog algoritma je izvedena preko platforme otvorenog koda VESC. Uz to je dan prijedlog poboljšanja, sa shemom za upravljanje elektromotorima baziran na VESC platformi, za kontrolu asinkronog motora, maksimalnog napona 600V i maksimalne struje 50A. Upotrebom izolacijskih ADC-ova smo omogućili sigurno mjerenje visokog napona, dok se za upravljanje glavnih tranzistora (MOSFET-a ili IGBT-ova) koriste izolirani upravljači vrata.

Ključne riječi: asinkroni motor, FPGA, digitalni fitler, izolacija, upravljanje

Analysis of different principles of controlling induction motor

Summary

The purpose of this work was comparison of different options for controlling induction motor. Digital filter was one of main focuses, because use of digital filters in electro motor control is highly beneficial in terms of cost and performance. *Sinc* filter was implemented in Verilog for FPGA chip. Different algorithms and methods of induction motor control are described, especially sensorless algorithms. Use of sensorless algorithms in control of induction motor lowers the cost of control system of induction motor and provide more flexible use in practice. Proposed architecture in this paper use FPGA, which adds to safety and protection. Digital filters are implemented on FPGA.

Implementation of sensorless algorithm was done on open-source VESC platform. In addition to that electric scheme was designed for control of induction motor, with max voltage of 600V and max current of 50A, as proposition for improvement of VESC hardware. With usage of isolated ADC's safe measurement of high voltage is provided. Isolated gate drive for MOSFET's and IGBT's is used so we can ensure maximum safety.

Key words: induction motor, FPGA, digital filter, isolation, contorl

SADRŽAJ

1. UVOD

Električni motor ili elektromotor je stroj koji pretvara električnu energiju u mehanički rad. Elektromotori koriste djelovanje magnetskih polja na električni vodič kojim teče električna struja. U uporabi je mnogo vrsta i izvedbi elektromotora, i danas su najviše korišteni pogonski strojevi u gotovo svim područjima ljudske djelatnosti; u kućanstvu, industriji, prometu (hladnjaci, perilice, lokomotive, tramvaji, električni automobili i sl.).

Upravljanje je jedna on najbitnijih stvari kada su u pitanju elektromotori. Ono se danas vrši sa algoritmima koji su omogućili odlične karakteristike elektromotora u vidu odziva, početnog momenta itd.. Sa modernim sustavima upravljanja elektromotorima, možemo unaprijediti postojeća industrijska postrojenja uz relativno mali trošak, tj. bez mijenjanja elektromotora, nego samo dodavanjem boljeg upravljanja elektromotorom. Danas imamo mnogo načina upravljanja, a sve ih možemo podijeliti na skalarne i vektorske.

Skalarno upravljanje je jedno od najstarijih metoda koja se koristi i dalje, radi svoje jednostavnosti, cijene i prihvatljivih nedostataka u odnosu na moderne metode.

Današnja moderna metoda, koja se vrlo često koristi, je vektorsko upravljanje koje možemo podijeliti na upravljanje direktno momentom (engl. Direct Torque Control- DTC), te vektorsko upravljanje zasnovano na ulančanom magnetnom toku (engl. Field-oriented Control- FOC).

U ovome radu će se opisati navedene metode, te njihova primjena i algoritmi, kao i različite iteracije istih algoritama od kojih svaka ima svoje mane i prednosti. Također će biti opisane prednosti primjene FPGA integriranog kruga i digitalnih filtera kod upravljanja elektromotorima. Takva arhitektura primjenom FPGA integriranog kruga i digitalnih filtra kod upravljanja elektromotorima je dizajnirana i simulirana u ovom radu. Implementacija

bezsenzorskog algoritma je izvedena preko platforme otvorenog koda VESC. Uz to je dan prijedlog poboljšanja, sa shemom za upravljač elektromotorima, baziran na VESC platformi, i to za kontrolu asinkronog motora, maksimalnog napona 600V i maksimalne struje 50A.

2. ASINKRONI MOTORI

Indukcijski elektromotor ili asinkroni elektromotor ima rotirajući dio (rotor) na koji se električna nergija prenosi indukcijom, pod djelovanjem okretnog magnetskog polja koje stvara sustav višefaznih struja u statoru.

Prema izvedbi rotorskog namota dijele se na kavezne i klizno-kolutne elektromotore. Ovakvi električni strojevi su jednostavne konstrukcije i pouzdani u pogonu, pa se i najčešće koriste u svim vrstama elektromotornih pogona. Princip rada asinkronih elektromotora je takav da stator proizvodi okretno magnetno polje koje se vrti brzinom frekvencije napona/struje koju dovedemo na stezaljke statorskih izvoda. Rotor asinkronog elektromotora u motornom režimu rada se uvijek vrti sporije od statorskog magnetnog polja, dok je kod generatorskog režima rada klizanje veće od jedan. Radi ovoga se magnetno polje statora rotira relativno u odnosu na rotor. Zbog ove relativne brzine postoji indukcijska sprega između statora i rotora, koja proizvodi struju u rotoru. Struje u rotoru proizvode magnetno polje, koje se radi Lorentzovog zakona opire statorskom polju. Iz tog razloga da bi se struja rotora oduprla izvoru, rotor se krene okretati u smjeru statorskog okretnog magnetnog polja. Rotor krene ubrzavati do blizu sinkrone brzine, ali nikada ne dostigne sinkronu brzinu, jer tada ne bi postojalo indukcije magnetnog polja rotora, a samim time i momenta. Ovu razliku u brzini između rotora i sinkrone brzine nazivamo klizanje.

Asinkroni elektromotori su po tome i dobili naziv, jer brzine magnetnog polja statora i mehanička brzina rotora nisu jednake. Formula za klizanje je jednaka razlici sinkrone i trenutne brzine podijeljena sa sinkronom brzinom.[9]

$$s = \frac{n_s - n}{n_s} \tag{2.1}$$

Gore navedene brzine su svedene na mehaničke brzine, gdje je *s* - klizanje, n_s - sinkrona brzina, *n* - brzina rotora elektromotora. Brzinu okretnog magnetnog polja statora možemo

izračunati preko frekvencije statorske struje.

$$\omega_{el} = 2 * \pi * f \tag{2.2}$$

Veza između brzine okretnog polja statora ω_{el} i sinkrone brzine rotora je broj pari polova - p.

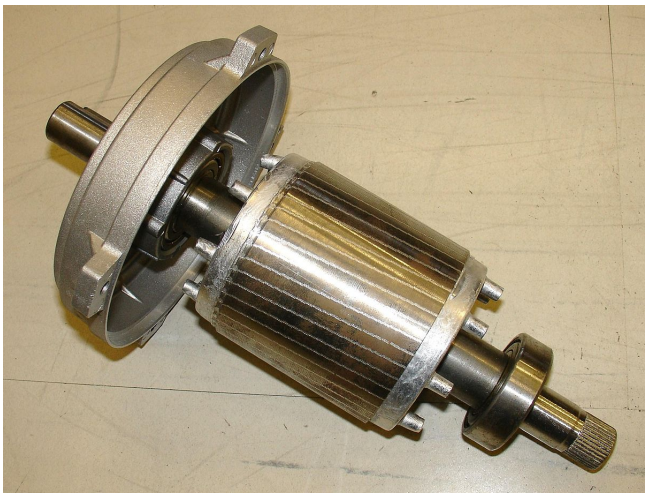
$$\omega_s = \omega_{el} * p \tag{2.3}$$

Na kraju sve možemo preračunati u mehaničku brzinu okretaja po minuti.

$$n_s = \frac{120 * f}{p} \tag{2.4}$$

Gdje je f - frekvencija struje, p - broj pari polova, a n_s - mehanička brzina rotora izražena u okretajima po minuti (*okr./min.*).

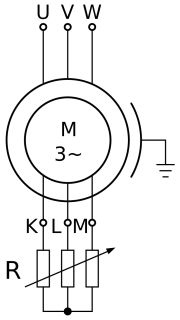
Indukcijski/asinkroni elektromotor je danas vrlo često upotrebljavan elektromotor, osobito u industriji. Razlog tome je relativno niska cijena i robusnost samog stroja. Najčešće primjenjivani asinkroni elektromotor je asinkroni elektromotor kaveznog tipa. Naziv je dobio prema rotorskom namotu koji se sastoji od neizoliranih i kratko spojenih aluminijskih vodiča raspoređenih po obodu željezne jezgre rotora, što izgleda slično kavezu.[7] Kako



Slika 2.1: Kavezni rotor asinkronog elektromotora.

rotor izgleda kao kavez, takav elektromotor nazivamo kaveznim elektromotorom. Drugi tip asinkronih elektromotora, koji je gore spomenut, je klizno-kolutni asinkroni elektromotor. Ovakva izvedba asinkronog elektromotora ima rotorske namote spojene preko kliznih koluta na vanjske otpornike. Vanjski otpornici služe za smanjenje startne struje, te struje tokom

rada. Prilikom starta, klizno-kolutni elektromotor ima namote rotora spojene na vanjski napon, preko otpornika. Nakon što elektromotor dosegne punu brzinu, namoti rotora se odspoje od vanjskog napona, te se ukratko spoje. Upravo iz ovoga razloga početni moment klizno-kolutnog asinkronog elektromotora je dosta velik. Vanjski otpornici se mogu koristiti za jednostavnu izvedbu kontrole brzine ove vrste elektromotora. Danas ovaj tip elektromotora nalazi primjenu gdje je bitna cijena i iznimna robusnost, iako sve rijeđe, jer je današnje upravljanje asinkronih elektromotora, konkretno kaveznog asinkronog elektromotora, premostilo sve dobre strane klizno kolutnog elektromotora.



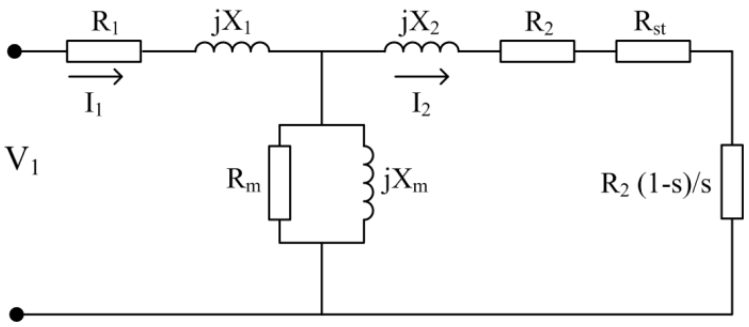
Slika 2.2: Shema klizno kolutnog elektromotora - statički model.[7]

2.1. Nadomjesna shema asinkronog elektromotora

Na Slici 2.2 nadomjesna shema asinkronog elektromotora. Ovdje je R_1 - otpor statorskog namota, X_1 - rasipni induktivitet statora, X_m - induktivitet gubitaka jezgre, R_0 - otpor gubitaka u jezgri. X_2 je induktivitet rotora reduciran na stranu statorske strane, te R_2 - otpor rotora, R_{st} - otpor rotora ovisan o klizanju. R_2 i R_{st} možemo zajedno napisati kao $R_2 = \frac{R_2}{s}$. Iz nadomjesne sheme možemo napisati da je ukupna struja elektromotora jednaka:

$$I_s = \frac{U}{Z_{eq}} \tag{2.5}$$

Gdje je I_s - statorska struja, U - narinuti napon na statoru, a Z_{eq} - ekvivalentni otpor kruga. Iz nadomjesne sheme možemo vidjeti da kako se povećava klizanje, tako raste struja rotora, a struja statora se smanjuje.



Slika 2.3: Shema klizno kolutnog elektromotora.[7]

Snaga elektromotora za sve tri faze je:

$$P_{f3} = 3 * I_s^2(\frac{1-s}{s})R_2' \tag{2.6}$$

, a moment je:

$$M = \frac{P_{f3}}{\omega} \tag{2.7}$$

U realizaciji upravljanja asinkronog elektromotora, javlja se povećana potreba za transformacijom struja, napona i ulančanih tokova statora i rotora, iz trofaznog sustava u dvofazni i obratno. Također postoji potreba za prebacivanjem iz različitih koordinatnih sustava, rotorskog, statorskog i rotirajućeg.

2.2. Clark-ova transformacija

Da bi mogli raditi gore navedene transformacije potrebna nam je Clarkova transformacija. Ona se temelji na tome da tri struje vektora *a,b,c*, možemo prikazati kao dva vektora α i β, s tim da je suma vektora i_s jednak za oba sustava.[17]

$$\alpha = a \tag{2.8}$$

$$\beta = \frac{(b-c)}{\sqrt{3}} \tag{2.9}$$

Također možemo napisati:

$$i_s = i_\alpha + ji\beta = \frac{2}{3}(i_{as} + \overline{a}i_{bs} + \overline{a^2}i_{cs}) \tag{2.10}$$

2.3. Park-ova transformacija

Pomoću Parkove transformacije dobijemo vrijednosti vektora rotirajućeg sustava.[17] Transformacija se vrši na α,β vektorima te dobijamo *d,q* vektore.

$$i_d = i_\alpha * cos(\theta)_\lambda + i\beta * sin(\theta)_\lambda \tag{2.11}$$

$$i_q = -i_\alpha * sin(\theta)_\lambda + i\beta * cos(\theta)_\lambda \tag{2.12}$$

Povratna transformacija je također vrlo jednostavna, s tim da su ove jednadžbe napisane u naponskom obliku:

$$v_\alpha = v_d * cos(\theta)_\lambda - vq * sin(\theta)_\lambda \tag{2.13}$$

$$v_\beta = v_d * sin(\theta)_\lambda + vq * cos(\theta)_\lambda \tag{2.14}$$

3. UPRAVLJANJE ASINKRONIM MOTORIMA

Postoji više metoda za upravljanje asinkronog motora. Možemo ih podijeliti u dvije osnovne skupine:

- Skalarnu
- Vektorsku

3.1. Skalarna V/Hz metoda

U skalarnoj V/Hz metodi, brzina asinkronog motora se kontrolira pomoću amplitude i frekvencije napona statora, tako da se magnetni tok u zračnom rasporu održava na željenoj vrijednosti. Ova metoda se može jednostavno objasniti preko pojednostavljene nadomjesne sheme statičkog stanja. Može se uzeti da je radni otpor statora jednak nuli, i da je rasipni induktivitet statora ugrađen u rasipni induktivitet rotora.[9] Tako da se može pisati:

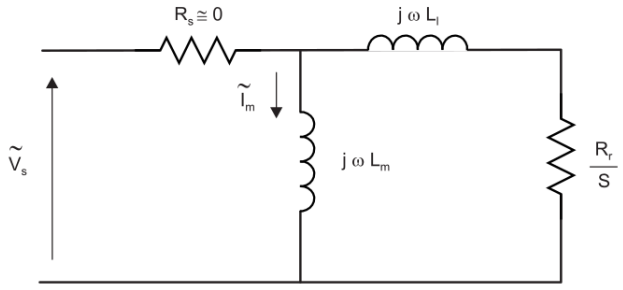
$$I_m \approx \frac{V_s}{j\omega L_m} \tag{3.1}$$

Ako je elektromotor u linearnom području, L_m je konstanta, tako da se može pisati[7]:

$$I_m = \frac{V_s}{(2\pi f)L_m} = \frac{\Psi}{L_m} \tag{3.2}$$

Gdje je Ψ - magnituda statorskog toka. Iz ovoga se vidi da ako se odnos V/Hz drži konstantnim, tada će i tok ostati konstantan.

Ova metoda upravljanja asinkronim elektromotorom se može primijeniti sa otvorenom petljom, kao što je prikazano na Slici 3.2, ili se može primijeniti u zatvorenoj petlji, gdje je povratna informacija brzina i struja asinkronog elektromotora.[2] Slika 3.3 prikazuje



Slika 3.1: Asinkroni motor - statičko stanje[9]

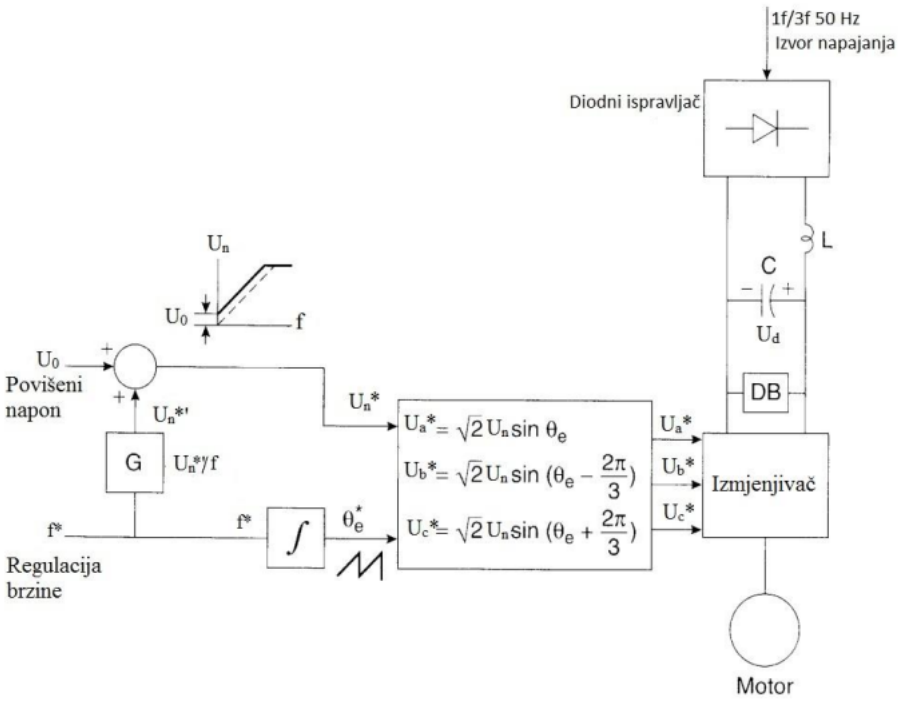
karakteristiku ovisnosti brzine tereta u mirovanju, i pogonu kad je spojen nekakav teret. Kako se V/Hz postepeno povećava, tako proporcionalno raste i brzina, a krivulja momentna ostaje konstantna, što se može vidjeti iz točaka od 1 do 7. Ova vrsta tereta mijenja moment ovisno o brzini što se također vidi na točkama od 1 od 7. Nakon što elektromotor dosegne maksimalni nazivni napon, dalje možemo povećavati samo frekvenciju, što dovodi do slabljenja polja. To vidimo na točkama od 5 do 7. Također možemo primijetiti da prilikom ovog efekta slabljenja polja, krivulja momenta se smanjuje, tj. maksimalni moment za brzinu poviše nazivne je manji.[9]

3.2. Vektorska kontrola

Vektorska metoda upravljanja elektromotorom dalje može podijeliti na FOC (engl. *Field Oriented Control*) - upravljanje zasnovano na ulančanom magnetnom toku, te DTC (engl. *Direct Torque Control*) - izravna kontrola momenta.

3.2.1. FOC - kontrola zasnovana na ulančanom magnetnom toku

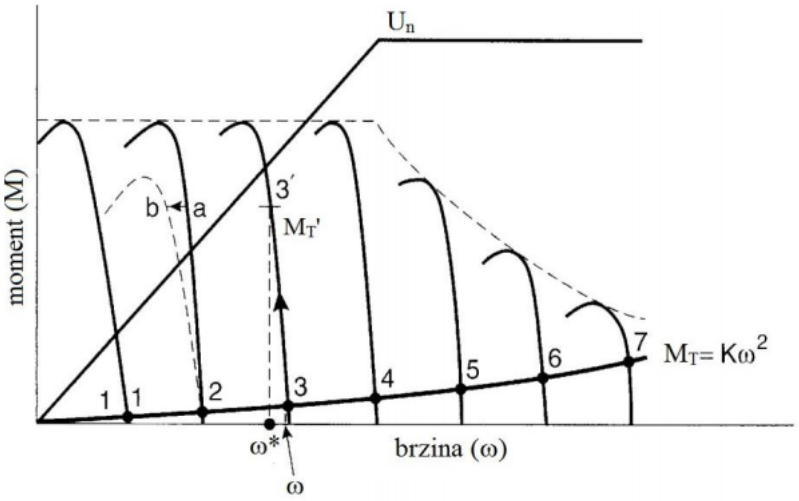
Vektorska kontrola motora - FOC (engl. *Field Oriented Control*) je metoda upravljanja statorskih struja trofaznog asinkronog/sinkronog elektromotora sa dvije ortogonalne vektorske komponente. Jedna komponenta definira magnetski tok motora, a druga moment. Uređaj koji upravlja elektromotorom na osnovu zadane brzine i/ili momenta upravlja sa ovim komponentama, kako bi se postigao željeni rezultat. Iako je prijedlog za vektorsku regulaciju postoji još od 1968. godine, njegova primjena je tek započela pojavom mikrokontrolera,



Slika 3.2: Blok shema V/Hz metode s otvorenom petljom[4]

početkom 1980. godine.

Vektorsko upravljanje motora temelji se na regulaciji momenta nezavisno uzbuđenog istosmjernog motora. Upravljanje nezavisno uzbuđenog istosmjernog motora sastoji se od upravljanja dva zasebna kruga, uzbudnog i armaturnog, tj. upravljanjem dvije zasebne struje; struje armature i struje uzbude. To znači da je moment upravljan strujom armature, a magnetni tok rotora strujom uzbude. Ovi tokovi kod asinkronog motora su okomiti, pa možemo reći da su raspregnuti. Ovakav način upravljanja može se implementirati kod asinkronih motora, ako se upravljanje promatra sa referentnog okvira rotirajućeg koordinatnog sustava, gdje se sinusne varijable pojavljuju kao istosmjerne varijable *Id* i *Iq*. Kao što je prikazano na Slici 3.5. Da bi izračunali varijable rotirajućeg sustava *d,q*, prvo se treba izračunati dvofazne varijable statorskog sustava α i β[9]. Varijable α i β se računaju iz statorskih struja/napona preko Clarkove transformacije. Zatim se iz statorskih varijabli α i β preko Parkove transformacije računa struja/napon *d,q*. Obzirom da je asinkroni elektromotor zatvorena strujna petlja, mogu se mjeriti samo dvije struje, a treću možemo proračunati iz zbroja dvije izmjerene struje. Kao što je već navedeno, upravljanje se vrši preko rotirajućih



Slika 3.3: Karakteristika momenta u odnosu na brzinu pri promjeni frekvencije, teretnog momenta i napona[4]

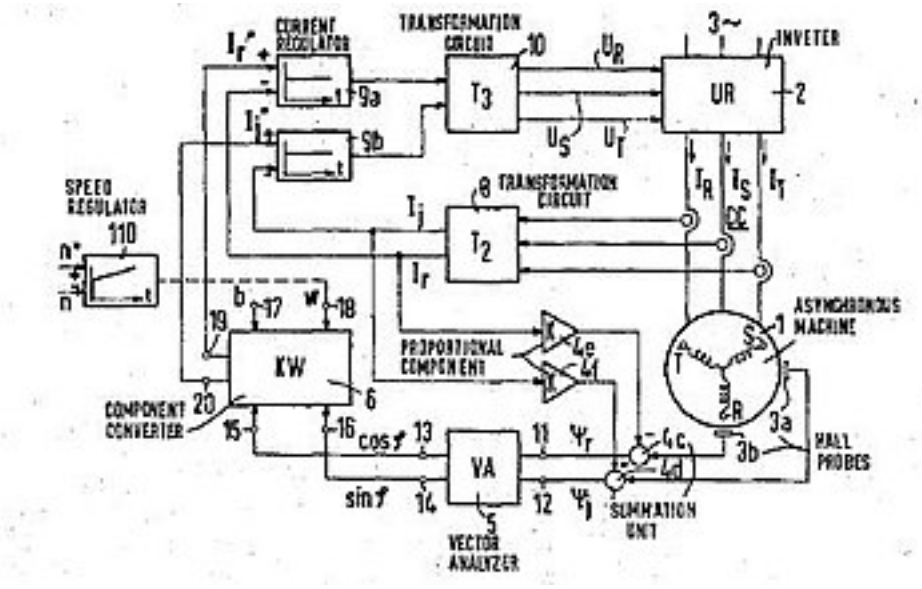
veličina *d,q* gdje je *Iq* - struja momenta, a *Id* - struja magnetnog polja, što se može vidjeti sa Slike 3.6.

Vektorska kontrola upravlja sa ove dvije veličine, pri tome struja *Id* treba biti konstanta, a *Iq* treba odgovarati iznosu momenta tj. snage. Kada ”korisnik” povećava iznos tj. traži od elektromotora veću snagu, ono što kontroler treba napraviti je da poveća struju *Iq* tj. moment, a samim time i snagu jer je:

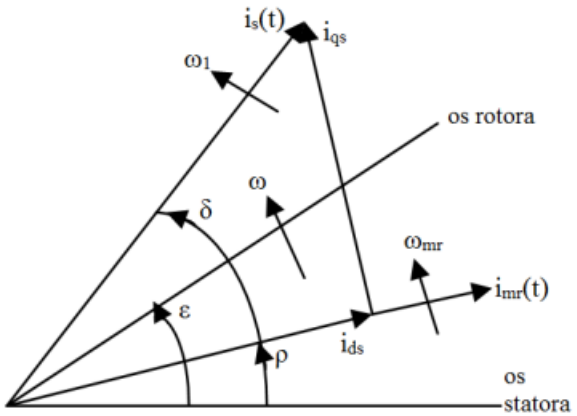
$$P = \omega * M \tag{3.3}$$

3.2.2. Senzorska kontrola

Kao što je prikazano na Slici 3.6, kako bi se moglo proračunati potrebnu struju *Iq* s obzirom na zahtjev ”korisnika”, mora se poznavati trenutačna brzinu rotora. Ta informacija se može dobiti preko ”promatrača”, što je prikazano na Slici 3.6. Kako bi se vrijednosti *d,q* proračunale iz vrijednosti α, β, mora se poznavati kut rotora. Umjesto ”promatrača”, senzorska kontrola koristi enkoder koji nam omogućuje preciznu brzinu i kut rotora.



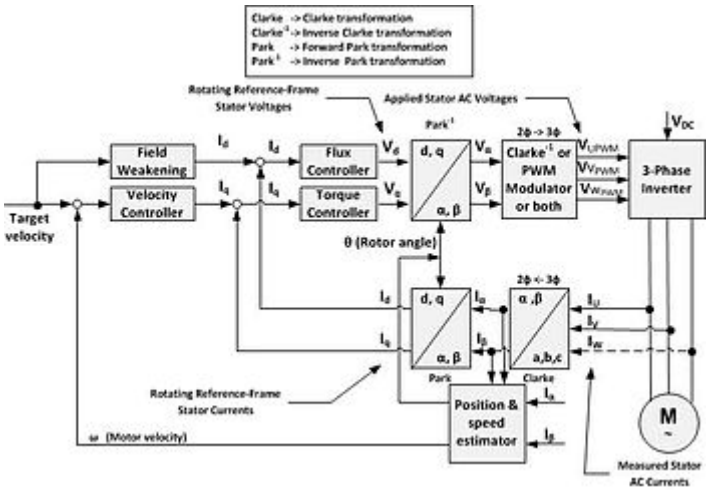
Slika 3.4: Blok dijagram FOC algoritma Blaschke 1971 US patent[3]



Slika 3.5: Vektorski dijagram[11]

3.2.3. Bezsenzorska kontrola

Glavna stvar kod bezsenzorske kontrole je pokušati dobiti iz vrijednosti struje i napona asinkronog motora, vrijednost pozicije rotora i brzine rotora, kako bi ih mogli primijeniti u FOC algoritmu, kao što je prikazano na Slici 3.6.



Slika 3.6: Bezsenzorski FOC algoritam[8]

3.2.3.1. ”Promatrač”

”Promatrač” je dio FOC algoritma koji izračunava kut zakreta rotora. Algoritmi koji se koriste su:

- Rotorski model
- Statorski model
- MRAS observer
- Kalman filter

Rotorski model se dobije iz diferencijalne jednadžbe rotorskog namota. Algoritam se može implementirati sa statorskih koordinata ili koordinata rotirajućeg polja.[8]

$$\tau_r \frac{d\Psi_r}{d\tau} + \Psi_r = j\omega\tau_r\Psi_r + l_m i_s \tag{3.4}$$

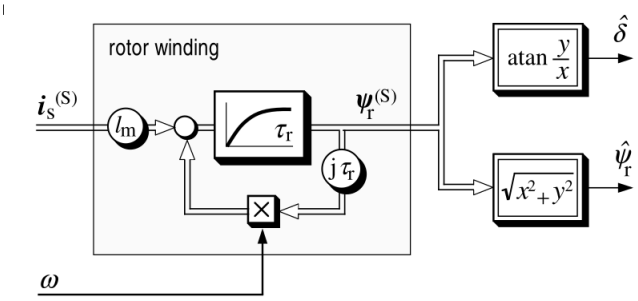
Ulančano magnetno polje rotora $\Psi_r^{(S)}$ se može izračunati preko statorskih koordinata kako je prikazano na slici 3.7. Argument od Ψ_r jednak je kutu rotorskog polja δ.

$$\Psi_r^{(S)} = \Psi_r cos(\delta) + j\Psi_r sin(\delta) = \Psi_{r\alpha} + j\Psi_{r\beta} \tag{3.5}$$

Oba signala $\Psi_{r\alpha}$ i $\Psi_{r\beta}$ se dobiju preko slijedećih funkcija.

$$\delta = arctan\frac{\Psi_{r\beta}}{Psi_{r\alpha}} \tag{3.6}$$

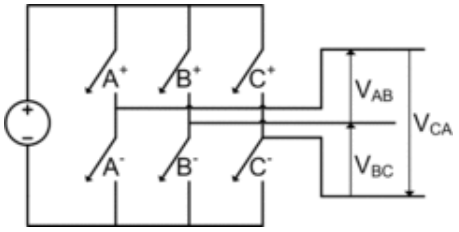
$$\Psi_r = \sqrt{{\Psi_{r\beta}}^2 + {Psi_{r\alpha}}^2} \tag{3.7}$$



Slika 3.7: Rotorski model u statorskim koordinatama

3.3. Vektorska modulacija napona

Vektorska modulacija napona - SVM (engl. *Space Vector Modulation*) je predstavljanje trofaznih veličina *a,b,c* s pomakom od $120°$. S obzirom na topologiju koja je prikazana na Slici 3.8 ima 8 mogućih kombinacija vektora. S obzirom na način na koji se koriste ovi vektori možemo smanjiti/povećati harmonike i/ili gubitke. Referenti napon V_{ref} se uzorkuje



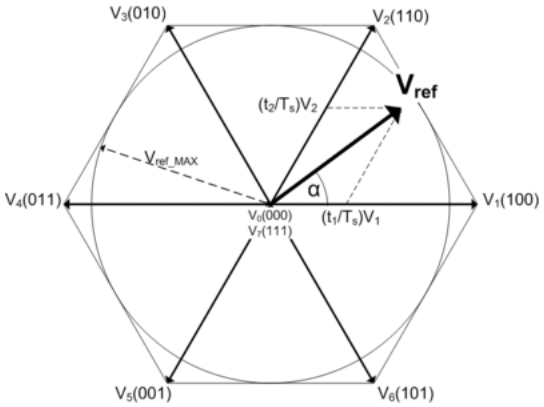
Slika 3.8: Topologija trofaznog invertera

sa vremenom uzorkovanja T_s. Za vrijeme perioda uzorkovanja V_{ref} se rekonstruira na temelju najbližih vektora heksagona i nul-vektora.[17]

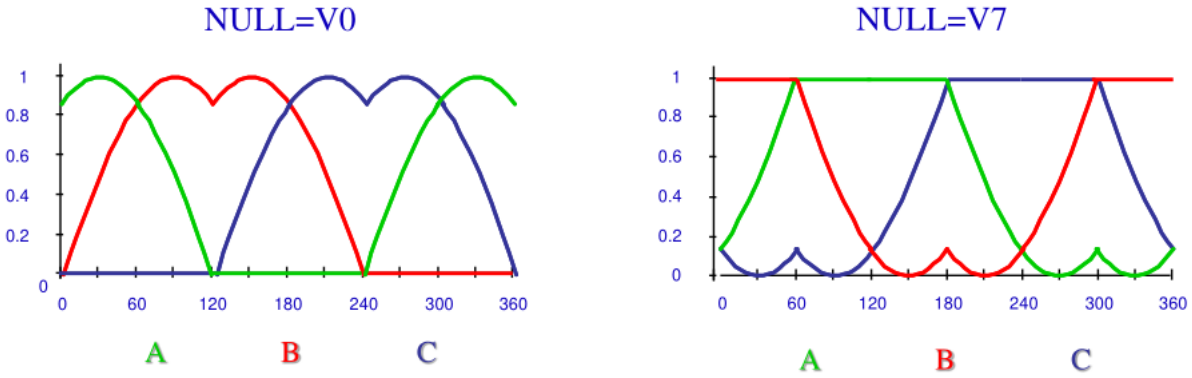
$$V_{ref} = V_x * T_1 + V_y * T_2 + V_{null} * T_0 \tag{3.8}$$

Gdje je V_x - vektor sa najmanjim kutom, a V_y- vektor sa najvećim kutom kao što je prikazano na Slici 3.9.

Pošto se kao izbor pojavljuju dva nul-vektora, može se odlučiti koji ćemo odabrati. Za motor nije bitno koji se nul-vektor odabere, jer će motor uvijek vidjeti sinusnu struju. Jedina razlika je ta da prilikom odabira V_0 vektora u odnosu na V_7 mogu se smanjiti sklopni gubitci kao što je prikazano na Slici 3.10. Vektorska modulacija napona je simulirana preko LTSpice

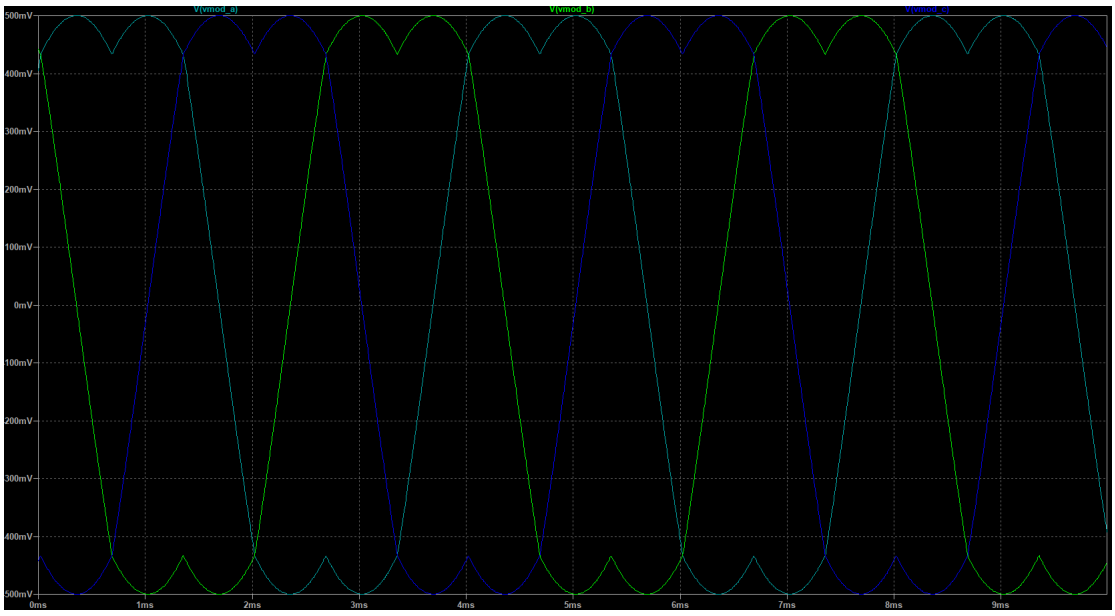


Slika 3.9: Svih osam mogućih vektora[17]



Slika 3.10: Vektorska modulacija razlika izmedu korištenja nul-vektora[17]

programa za simulaciju, a sama shema se nalazi u Prilogu 2. Na Slici 3.11 je prikazan rezultat simulacije koji nam prikazuje signale prije modulatora V_{modA}, V_{modB}, V_{modC}.



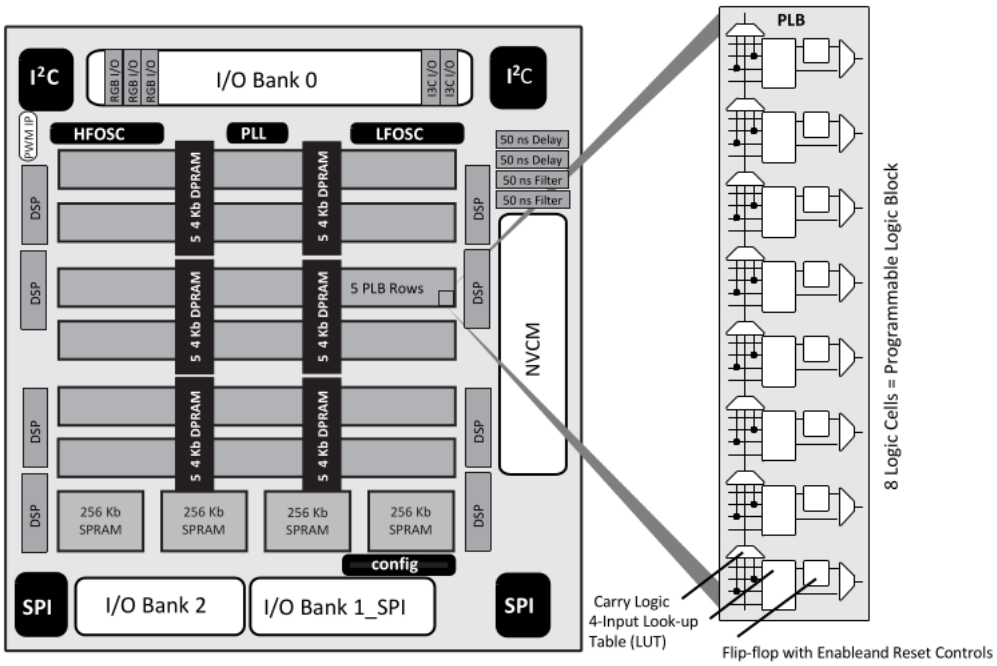
Slika 3.11: Vektorska modulacija - LTSpice simulacija

4. FPGA

FPGA - (engl. *Field Programmable Gate Array*) je integrirani krug dizajniran tako da korisnik može raditi promjenu u njemu nakon izrade. FPGA je sastavljen od niza malih logičkih jedinica poznatih kao LUT-ovi (engl. *Lookup Table*), kao što je prikazano na Slici 4.1. Logičke jedince se sastoje od neke kombinacije *I (engl. AND)* i *XILI (eng. XOR)*, ili nekih drugih kompleksnih digitalnih elemenata. Ove logičke jedince se mogu povezivati po želji korisnika, te se tako se mogu dobiti vrlo kompleksni algoritmi. FPGA integrirani krugovi se programiraju sa takozvanim hardverskim opisnim jezicima HDL (engl. *Hardware Description Language*). Primjeri takvih vrsta jezika su Verilog, VHDL te SystemVerilog.

Najznačajnija osobina FPGA integriranog kruga je mogućnost definiranja hardvera po želji korisnika. U konkretnom slučaju upravljanja elektromotorom, to nam omogućuje da neželjene radnje (paljenje dva tranzistora koji se ne bi smjeli istovremeno otvoriti jer stvaraju kratki spoj) i greške (prevelika struja, napon itd.) otklone tj. registriraju se vrlo brzo, osobito u odnosu na mikrokontroler. Također, implementacija zaštite tranzistora i motora u FPGA integriranom krugu se tehnički smatra kao hardverska implementacija. Iz tog razloga sigurnost i zaštite su na visokom nivou. Ovakva implementacija omogućuje da zaštita radi, bez obzira na to koji program je stavljen u mikrokontroler.

Moderni FPGA integrirani krugovi danas u sebi imaju ugrađene i DSP (engl. *Digital Signal Processing*) blokove, tako da je i filtriranje ulaznih digitalnih signala struje i napona vrlo brzo i efikasno. Sa ovime su se oslobodili resursi mikrokontrolera, te mu omogućeno da obavlja druge zadatke.

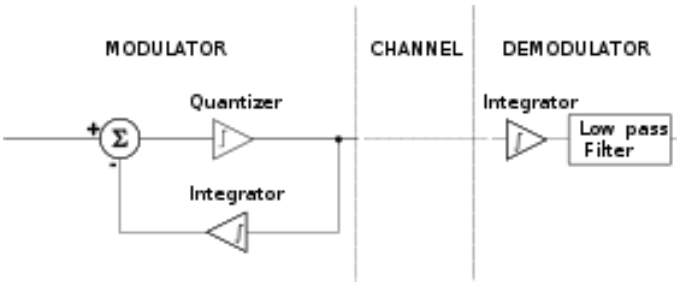


Slika 4.1: Arhitektura FPGA integriranog kruga (Lattice ice40UP)[15]

5. DELTA-SIGMA ANALOGNO DIGITALNI PRETVORNIK

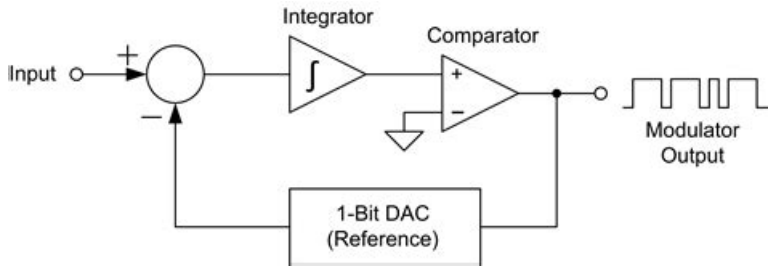
Delta-sigma modulacija je metoda kodiranja analognih signala u digitalne signale kao ADC - analogno digitalni pretvornik (engl. *Analog-Digital Converter*). Također se koristi i kao metoda za dekodiranje digitalnih signala u analogne signale kao DAC - digitalno analogni pretvornik (engl. *Digital-Analog Converter*).

Prvi korak delta-sigma ADC-a je delta modulacija. Delta modulacija ne prikazuje apsolutnu vrijednost signala nego njegovu razliku u odnosu na prethodni signal. Modulator



Slika 5.1: Delta modulator shema[14]

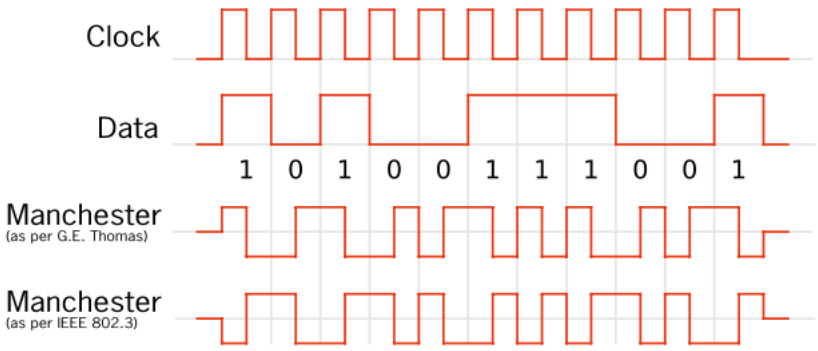
je napravljen tako da kvantizator konvertira razliku između ulaznog signala i usrednjenog signala iz prijašnjeg koraka. Kvantizator se može izvesti kao komparator koji je referenciran prema nuli. Takav komparator na izlazu daje 1 i 0 ovisno o tome je li ulazni signal pozitivan ili negativan. Rezultat ovoga je niz pulseva. Ako se ovaj niz pulseva integrira, te provede kroz nisko propusni filter, dobijemo ulazni signal. U delta ADC-u preciznost može se povećati tako da se doda 1-bitni DAC, te se doda rezulatat integracije izlaznog signala delta modulacije u ulazni signal, te se time smanji greška proizvedena delta modulacijom. Ovaj korak se naziva sigma i sa time se dobije delta-sigma ADC.[1] Da bi mikrokontroleri i ostali integrirani krugovi mogli razumijeti i manipulirati digitalnim podacima proizvedenim delta-sigma ADC-om, kao izlaz trebamo imati takt pri kojem ADC vrši operacije, te izlaz samog ADC-a.



Slika 5.2: Principijelna shema delta-sigma analogno-digitalnog pretvornika[14]

5.0.1. Manchester kodiranje

Ponekad se radi smanjenja kompleksnosti ili radi olakšanja izvedbe moguće koristiti kodiranje signala u kojem se takt i podaci prenose zajedno. Primjer takvog kodiranja je Manchester kodiranje. Manchester kod se temelji na tome da su bitovi predstavljeni

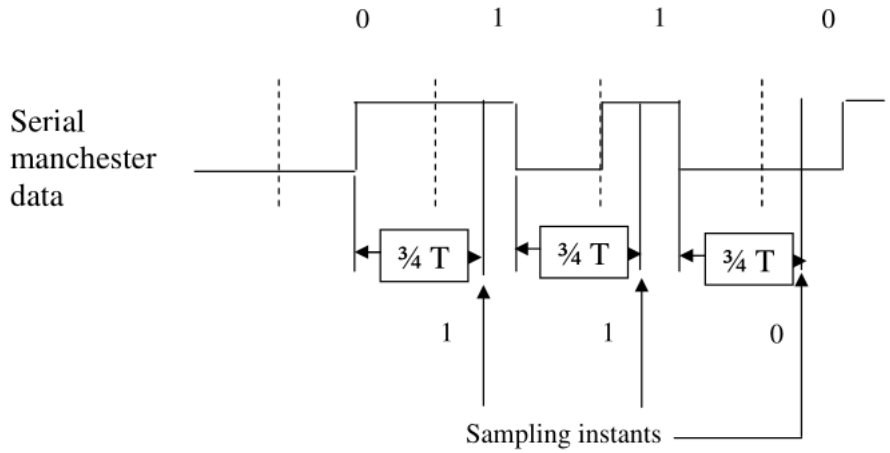


Slika 5.3: Dijagram Manchester kodiranja[10]

prijelazima iz 0 u 1 i obratno. Kod originalnog G.E Thomas koda prijelaz iz 0 u 1 predstavlja 0, a prijelaz iz 1 u 0 predstavlja 1. Dok po IEEE802.4 standardu je obratno, prijelaz iz 1 u 0 predstavlja 0, a prijelaz iz 0 u 1 predstavlja 1. Radi toga Manchester kod uvijek ima prijelaz na sredini svakoga bita. Ova vrstma kodiranja se primjenjuje kod prijenosa signala preko galvanske izolacije jer istosmjerna komponenta ne ovisi o ulaznom signalu.[10] Manchester kodiranje se može primjeniti uz pomoć XILI - ekskluzivno ILI vrata (engl. *(*XOR)). Jedan od načina dekodiranja se bazira na principu da se iz signala zasebno dobiju takt i podaci. Radi primjene FPGA integriranog kruga treba implementirati dekodiranje koje je prilagođeno za hardversku implementaciju koja najviše odgovara FPGA integriranom krugu. Jedan takav algoritam je baziran na tome da je vrijednost predstavljena u prvoj polovici vremena bita,

Podaci		Takt	Manchester kod
0	XOR	0	0
		1	1
1		0	1
		1	0

Table 5.1: Manchester kodiranje IEE802.4



Slika 5.4: Dekodiranje Manchester koda[16]

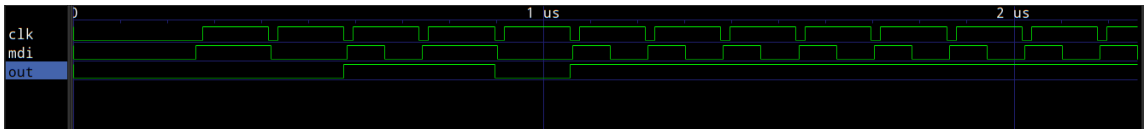
prije nego li se dogodi tranzicija. Ako se pokrene brojač koji je jednak 3/4 vremena takta, na početku tranzicije, vrijednost koja će se dobiti kada brojač završi je vrijednost sljedećeg bita. Potrebno vrijeme se može dobiti iz razlike vremena između dva najbliža brida.[16] Dobijeno vrijeme je jednako polovici vremena takta. Ovaj algoritam je implementiran u Verilogu:

Kod 5.1 : Dekodiranje Manchester podataka

```
// Dekodiranje podataka
always @(posedge clk_in)
begin
    // Poceti brojanje na pozitivnom i negativnom rubu
    if (((!mdi1 && mdi2)||(!mdi2 && mdi1)) && clk == 1'b0)
    begin
        clk <= ~clk;
    end
    // 3/4 takta
```

```
    if (clk > 0)
    begin
        if (count < delay)
        begin
            count++;
        end
        else
        begin
            nrz <= ~mdi;
            count <= 0;
            clk <= ~clk;
        end
    end
end
endmodule
```

Ovaj algoritam smo također i simulirali što je prikazano na Slici 5.5



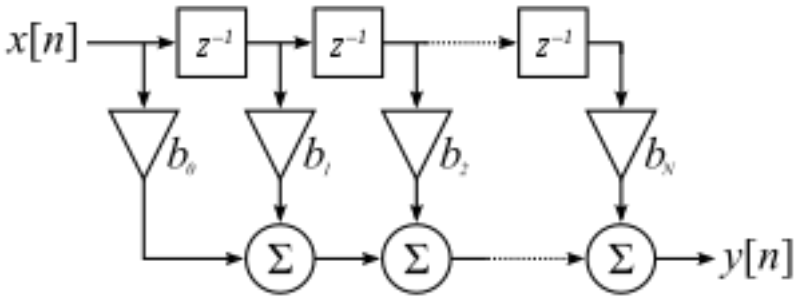
Slika 5.5: Rezultat Verilog simulacije

6. DIGITALNI FILTERI

Digitalni filter je sistem koji vrši matematičke operacije nad uzorkovanim diskretnim signalom kako bi smanjio ili poboljšao dijelove tog signala. Digitalni filter se većinom sastoji od ADC-a koji uzorkuje signal, te FPGA, ASIC, DSP ili MCU koji vrše potrebne matematičke operacije nad digitalnim signalom kako bi ga mogli filtrirati. Karakteristika digitalnog filtera se može prikazati preko prijenosne funkcije u z-domeni.

$$H(z) = \frac{B(z)}{A(z)} \tag{6.1}$$

B(z) i *A(z)* su polovi tj. nule te iste prijenosne funkcije. U praksi postoje i koriste se dvije vrste digitalnih filtera. IIR - beskonačni impulsni odziv (eng *Infinite Impulse Response*) i FIR - konačni impulsni odziv (engl. *Finite Impulse Response*) digitalni filteri. FIR filtri su takvi da im je odziv na konačni impuls takav da ima konačno trajanje. IIR filteri mogu na isti impuls imati beskonačan odziv zbog unutarnjeg pojačanja.[12] Kao što vidimo sa slike,



Slika 6.1: FIR filter u diskretnom vremenu[13]

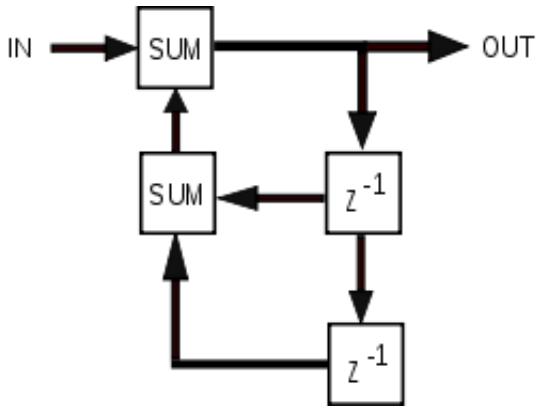
FIR filter ne treba nikakvu povratnu vezu, te je posljedično radi toga stabilan. To je glavna prednost FIR filtra u odnosu na IIR filter. Problem FIR-a je zahtijevanje velike procesorske moći u odnosu na IIR filter iste selektivnosti i oštrine. Sa zahtjevom velike procesorske moći

također dolazi do velike latencije FIR filtera, dok IIR za iste uvjete ima manju latenciju.[13] Prilikom dizajniranja FIR filtera imamo mogućnost koristiti sljedeće metode:

- Metoda dizajna ”prozorima”
- Parks-McClellan metoda
- Metoda frekvencijskog uzorkovanja
- Metoda najmanje srednje kvadratne pogreške

Ovdje se može naići na još jednu prednost IIR-a, a to je taj da za dizajniranje IIR-a ne trebaju se koristiti nove metode, nego se mogu uzeti postojeće metode za dizajniranje analognih filtera.[13] Nakon dizajniranja u analognoj domeni, prijenosna funkcija analognog filtera se može prebaciti u digitalnu domenu preko:

- Bilinearne transformacije
- Prebacivanje polova i nula iz Laplaceove *s* u *z*-domenu

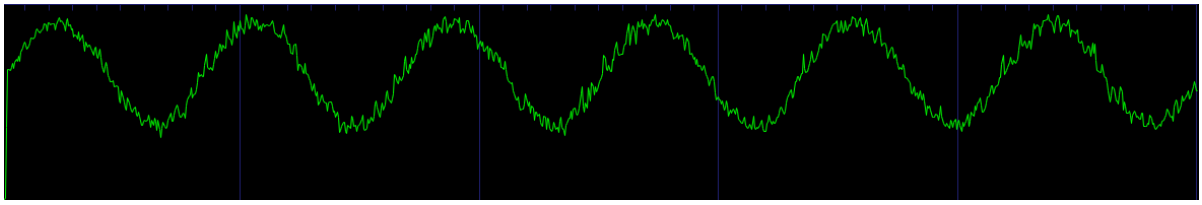


Slika 6.2: IIR filter u diskretnom vremenu[5]

Filter koji je vrlo jednostavan i praktičan za primjenu za izlaz delta-sigma ADC-a je *sinc* filter. *Sinc* filter se se ponaša kao idealni nisko propusni filter, te ima linearni fazni odziv. Odziv ovog filter na impulsu funkciju u vremenskoj domeni je matematička *sinc* funkcija.[6]

$$sinc(x) = \frac{sin(x)}{x} \tag{6.2}$$

Još jedna bitna značajka ovoga filtera je njegova brzina, čime se rješava potreba da filter ima što manji vremenski zaostatak za realnim vrijednostima, što je vrlo bitno kod upravljanja elektromotorom. Ovaj filter smanjuje broj uzoraka delta-sigma ADC-a, što nam je potrebno, jer se time na izlazu iz filtera dobije više-bitna riječ tj. vrijednost signal u tom trenutku.[1] (ulaz je samo 1 bit). Kod je implementiran u Verilogu, i simuliran simulatorom IcarusVerilogu. Na slici je prikazan digitalni signal koji je interpoliran u



Slika 6.3: Rezultat filtriranja *sinc3* filterom

programu GTKWave.

7. IMPLEMENTACIJA VEKTORSKE FOC KONTROLE ASINKRONOG MOTORA

Implementacija upravljanja je izvedena sa VESC - (engl. *Vedder Electric Speed Control*) platformom otvorenog koda za upravljanje BLDC/PMSM - (engl. *Brushless Direct Current Motor/Permanent Magnet Syncronus Motor*). Na ovoj platformi algoritam ”promatrača” za BLDC motor je zamijenjen sa algoritmom koji radi sa asinkronim motorima. Algoritam koji je primjenjen je jednostavni ”promatrač”, baziran na rotorskom modelu.

Najučestaliji način bezsenzorske kontrole je preko statorskog ulančanog magnetskog polja, koje se računa preko mjerenja struja i napona statora. Algoritam radi na principu integriranja statorskog napona V_{qds} i iz njega se dobije ulančano magnetno polje statora.

$$\Psi_{(}\alpha\beta s) = \int (V_{\alpha\beta s} - \hat{R}_s I_{\alpha\beta s}) \tag{7.1}$$

Sljedeća potrebna stavka je izračun ulančanog magnetnog polja rotora, koje se vrši na slijedeći način:

$$\Psi_{(}\alpha\beta s) = \frac{L_r}{L_m}(\Psi_{(}\alpha\beta s) - L_s^{'} i_{\alpha\beta s}^{'})) \tag{7.2}$$

Gdje je L_r - induktivitet rotora, L_m - međuinduktivitet rotora i statora, R_s - otpor statora. Kut zakreta rotora na kraju dobijemo iz sljedeće jednadžbe:

$$\theta = arctan(\frac{\Psi_\beta}{\Psi_\alpha}) \tag{7.3}$$

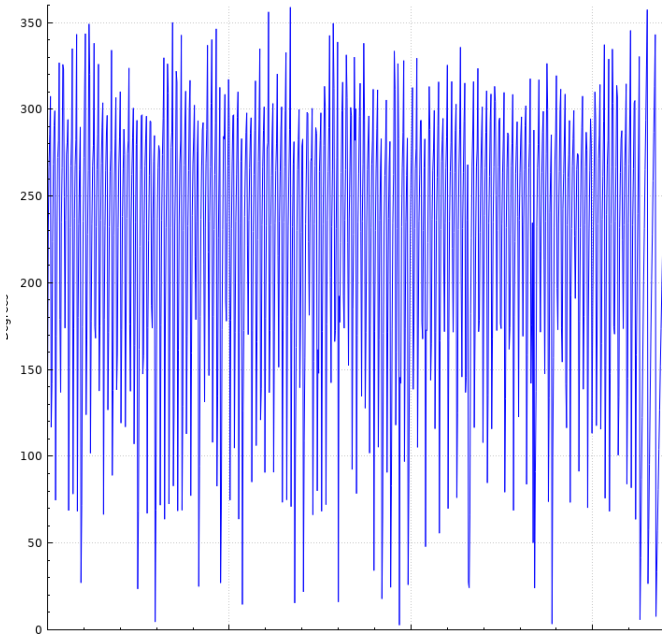
Kod 7.1 : ”Promatrač” baziran na rotorskom modelu

```
void observer_update(float v_alpha, float v_beta, float i_alpha, float
    i_beta,
                     float dt, volatile float *x1, volatile float *x2,
    volatile float *phase, volatile motor_all_state_t *motor) {
```

```
volatile mc_configuration *conf_now = motor->m_conf;

const float L = (3.0 / 2.0) * conf_now->foc_motor_l;
float R = (3.0 / 2.0) * conf_now->foc_motor_r;

// Saturation compensation
const float sign = (motor->m_motor_state.iq * motor->m_motor_state.vq
) >= 0.0 ? 1.0 : -1.0;
R -= R * sign * conf_now->foc_sat_comp * (motor->m_motor_state.
i_abs_filter / conf_now->l_current_max);

// Temperature compensation
const float t = mc_interface_temp_motor_filtered();
if (conf_now->foc_temp_comp && t > -25.0) {
    R += R * 0.00386 * (t - conf_now->foc_temp_comp_base_temp);
}

float rotor_flux_alpha = 0.0;
float rotor_flux_beta = 0.0;
float cos_rotor = 0.0;
float sin_rotor = 0.0;
const float L_ia = L * i_alpha;
const float L_ib = L * i_beta;
const float R_ia = R * i_alpha;
const float R_ib = R * i_beta;
const float lambda_2 = SQ(conf_now->foc_motor_flux_linkage);
const float gamma_half = motor->m_gamma_now * 0.5;

case FOC_OBSERVER_ORTEGA_ITERATIVE: {
    float LrLm = .5;
    float stator_flux_alpha = v_alpha - R_ia;
    float stator_flux_beta = v_beta - R_ib;
    // Integrate
    stator_flux_alpha = stator_flux_alpha * dt;
    stator_flux_beta = stator_flux_beta * dt;
    rotor_flux_alpha = LrLm * (stator_flux_alpha - L_ia) ;
    rotor_flux_beta = LrLm * (stator_flux_beta - L_ib) ;
    *x1 += stator_flux_alpha;
```

```
            *x2 += stator_flux_beta;

        } break;

        default:
            break;
        }

        UTILS_NAN_ZERO(*x1);
        UTILS_NAN_ZERO(*x2);

        if (phase) {
            else{
                *phase = utils_fast_atan2(rotor_flux_beta, rotor_flux_alpha);
            }
        }
}
```

Kod je testiran na VESC platformi. Rezultat je prikazan na Slici 7.1



Slika 7.1: Kut θ "promatrača" pri 100 *okr/min*

Drugi dio implementacije je prijedlog kako ovu platformu otvorenog koda pretvoriti u platformu za kontroliranje visokonaponskih asinkronih elektromotora. Platforma koristi

DRV8320 integrirani krug za kontrolu glavnih tranzistora. Ovaj krug ima sigurnosne osobine poput sprječavanja kratkog spoja na tranzistorima, itd.. Za proračun i upravljanje se koristi STM32 mikrokontroler koji je ostao i u predloženom dizajnu. *DRV8320* se više ne koristi iz razloga što je njegov maksimalni podnosivi napon 60V.

Ciljana vrijednost za ovaj prijedlog upravljača za asinkrone motore je napon od 600V i struja od 50A. Zbog tog razloga se koristi *Si8273* upravljač vrata za IGBT-ove i MOSFET-e koji je konstruiran za napone do 2kV, te je logička strana izolirana od upravljačke strane. Ovaj integrirani krug nema zaštitu od otvaranja dvaju serijskih tranzistora što će izazvati kratki spoj i uništenje tranzistora. Iz tog razloga je korišten Lattice-ice40UP FPGA integrirani krug, koji hardverski omogućuje sprječavanje navedenog slučaja. Zbog velikog napona, mjerenje napona i struje su izvedena preko *AMC1303* izolacijskih ADC-ova. Postoje dvije različite verzije izolacijskih ADC-ova, verzija sa Manchester kodiranjem i druga bez Manchester kodiranja. Sveukupno ima šest izolacijskih ADC-ova (za svaku fazu jedan naponski i jedan strujni). Zbog manjka pinova na korištenom Lattice-ice40UP FPGA integriranom krugu koristi se Manchester verzija izolacijskog ADC-a.

Unutar samog FPGA-a je implementiran *sinc* filter. Zbog načina rada *sinc* filtera, prenaponske i nadstrujne pojave se mogu registrirati i prije nego što filter završi konverziju. Nakon uočenog neželjenog stanja, FPGA prestaje davati signale prema upravljačima vrata. Jedna od velikih prednosti primjene FPGA-a, a samim time i digitalnih filtera, je ta što analogni filteri za ovu primjenu moraju biti precizni i stabilni, što diže cijenu proizvoda. Odabirom digitalnih filtera, ne samo da smo smanjili cijenu, nego smo i izbjegli probleme uzrokovane korištenjem analognih filtera uslijed starenja komponenti.

Mikrokontroler i FPGA imaju međusobnu komunikacije preko FSMC (engl. *Flexible Static Memory Controller*) protokola. Ovo omogućuje bržu i bezbolniju komunikaciju među njima.

8. ZAKLJUČAK

Ovaj rad se fokusirao na predstavljanje metoda za kontroliranje asinkronih elektromotora, te implementaciji nekih od njih na postojećoj platformi otvorenog koda VESC.

Kao što je opisano u radu, skalarna metoda bazirana na *V/Hz=konst.* ima svoje nedostatake, ali i prednosti, tako da se još primjenjuje u pumpama, ventilatorima gdje precizna kontrola brzine nije potrebna.

Upotreba digitalnih filtera je iznimo povoljna u slučaju upravljanja elektromotora jer omogućuje vrlo jednostavnu implementaciju nadstrujne i prenaponske zaštite koja je u ovom slučaju implementira u FPGA integriranom krugu.

FOC (engl. *Field Oriented Control*) se bazira na upravljanju zasnovanom na ulančanom magnetnom toku. Ova metoda upravljanja kompleksan problem svodi na jednostavniji način upravljanja, vrlo sličan upravljanju nezavisno uzbuđenog istosmjernog motora, gdje upravljamo sa dvije skalarne struje i_d i i_q.

Većina asinkronih motora u praksi nema ugrađen enkoder koji je prijeko potreban za upravljač elektromotora baziran na FOC algoritmu, jer mu je za kontrolu potrebna informacija o trenutnom kutu rotora θ i trenutnoj brzini rotora. Iz ovog razloga bezsenzorski FOC algoritam je vrlo poželjan u praksi jer omogućuje primjenu upravljača elektromotora na većem broju elektromotora.

LITERATURA

[1] Bonnie Baker. *Delt-Sigma ADCs*. URL: https://www.ti.com/analog-circuit/bonnie-baker-ebook.html. 10.06.2020.

[2] Nishant Garg Bilal Akin. URL: https://www.ti.com/lit/an/sprabq8/sprabq8.pdf. (26.06.2020).

[3] F Blaschke. *Apparatus for field-oriented control or regulation of asynchronous machines*. URL: https://patents.google.com/patent/US3805135A/en. (accessed: 01.09.2020).

[4] Bimal Kumar Bose. *Modern Power Electronics and AC Drives*. 2002.

[5] CCASL. *Infinite impulse response*. URL: https://en.wikipedia.org/wiki/Infinite-impulse-response. 05.06.2020.

[6] Gray R.M Chou W. Meng T.H. "Time domain analysis of sigma delta modulation". In: *Acoustics, Speech, and Signal Processing* 3 (1990), pp. 1751–1754. DOI: 10.1109/ICASSP.1990.115820.

[7] Martin J. Bozidar F. *Dinamika Električnih Strojeva*.

[8] J. Holtz. "Sensorless control of induction motor drives". In: 90.8 (2002), pp. 1359–1394. DOI: 10.1109/jproc.2002.800726.

[9] Texas Instruments. *Field Orientated Control of 3-Phase AC-Motors*. URL: https://www.ti.com/lit/an/bpra073/bpra073.pdf. (accessed: 26.06.2020).

[10] Brian Bailey Jatinder Gharoo. *Manchester Decoder Using the CLC and NCO*. URL: http://ww1.microchip.com/downloads/en/AppNotes/01470A.pdf. 07.08.2020.

[11] *Kompleksni vektor struje statora kod vektorskog upravljanja.* URL: https://www.fer.unizg.hr/_download/repository/DANFOSS_upute.pdf. (accessed: 26.06.2020).

[12] Alan S.; Oppenheim Alan V.; Willsky and Ian T. Young. *Signals and Systems*. Prentice-Hall Inc., 1980. ISBN: 0-13-809731-3.

[13] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. URL: https://ocw.mit.edu/resources/res-6-008-digital-signal-processing-spring-2011/index.htm. 05.05.2020.

[14] Arifur Rahman. "Assignment on Sigma-Delta Modulation". In: *Research gate* (2017). DOI: 10.13140/RG.2.2.30090.36807.

[15] Lattice Semiconductors. *FPGA ice40UP*. URL: https://www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus. (accessed: 16.06.2020).

[16] Signalpro. *Manchester Decoder*. URL: http://www.signalpro.biz/mandec.pdf. 10.09.2020.

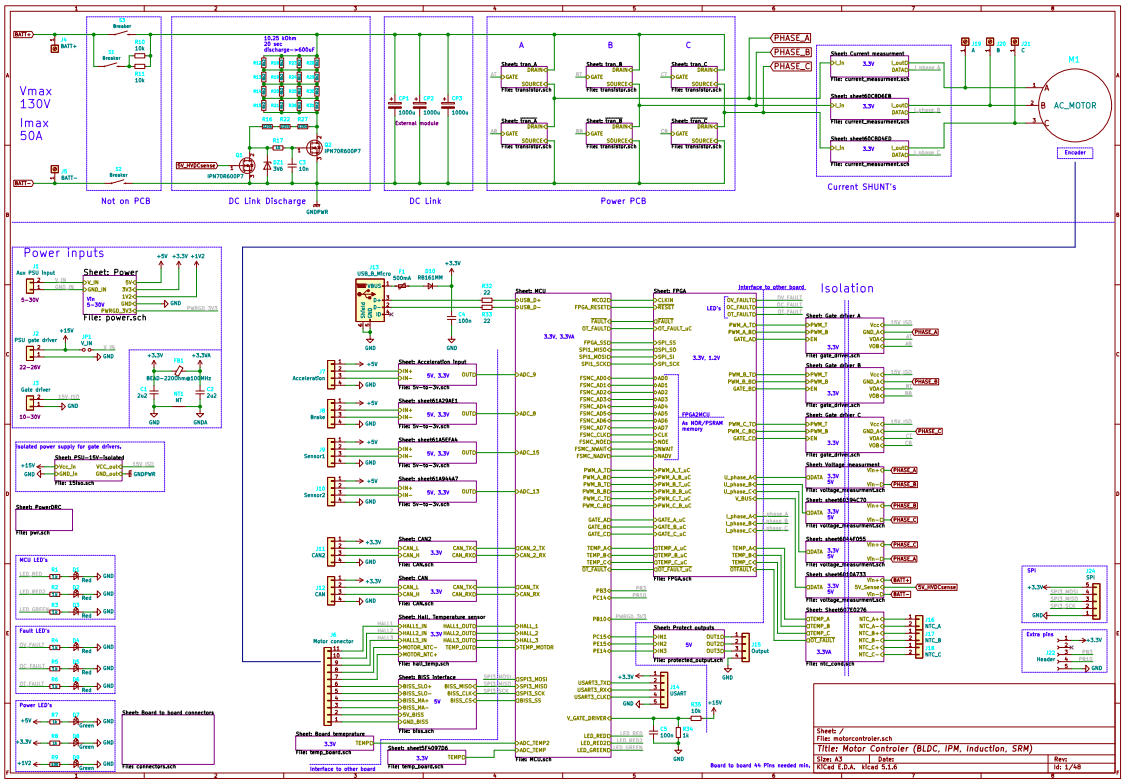
[17] David Wilson. *Teaching old motors new tricks*. URL: https://e2e.ti.com/support/archive/motor/m/pdf_presentations/. (accessed: 01.09.2020).

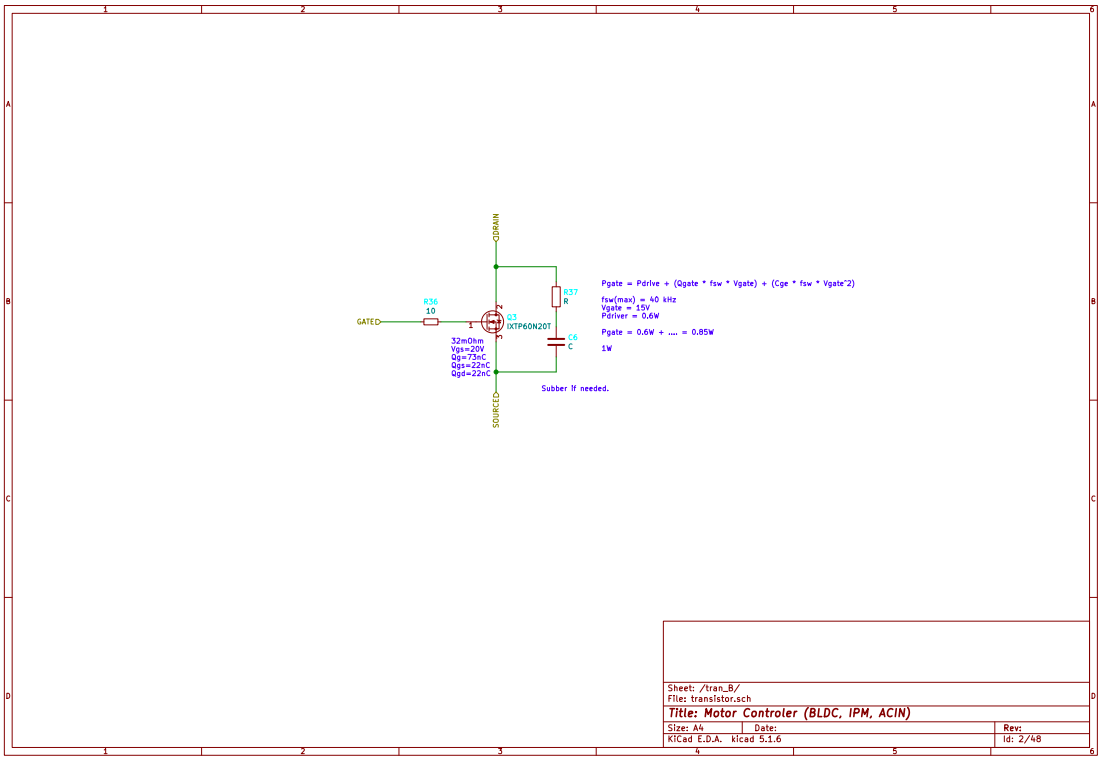
POPIS SLIKA

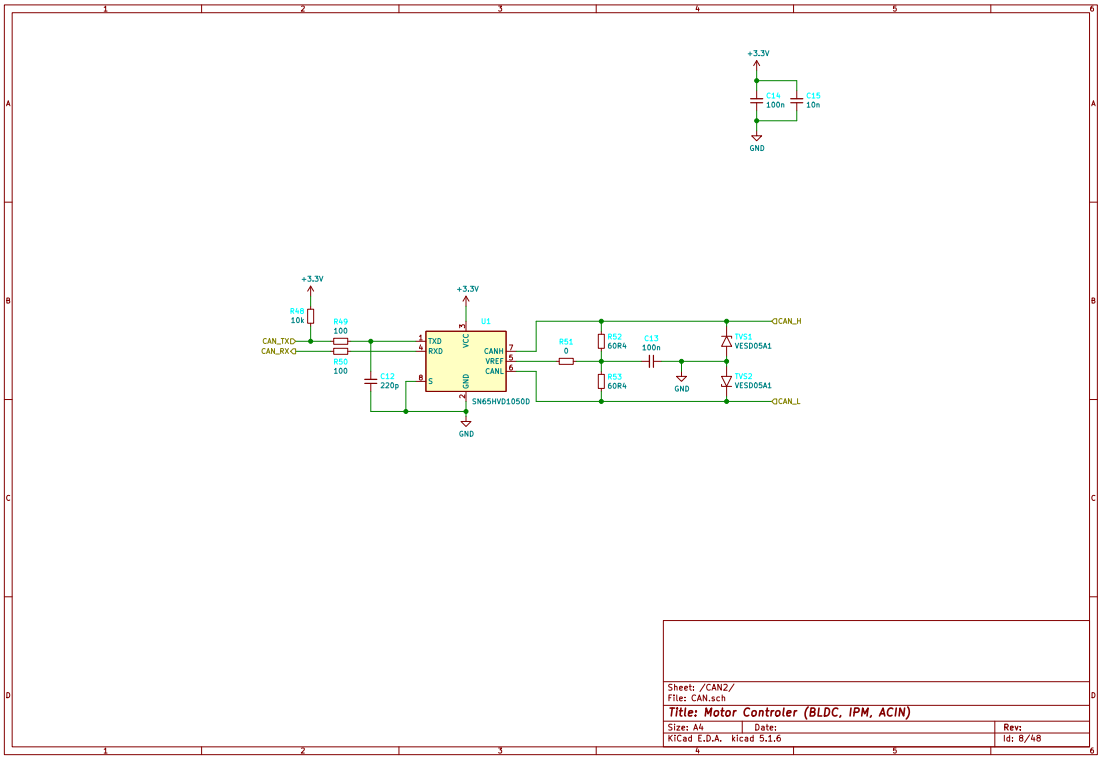
POPIS TABLICA

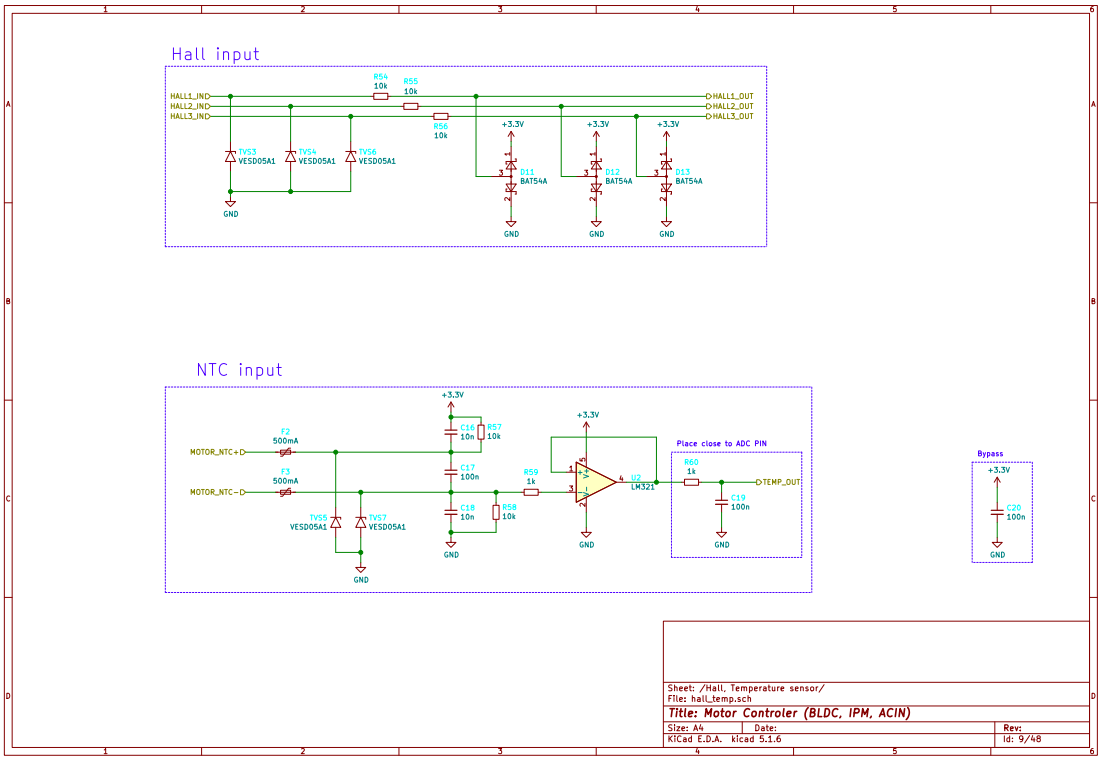
PRILOZI

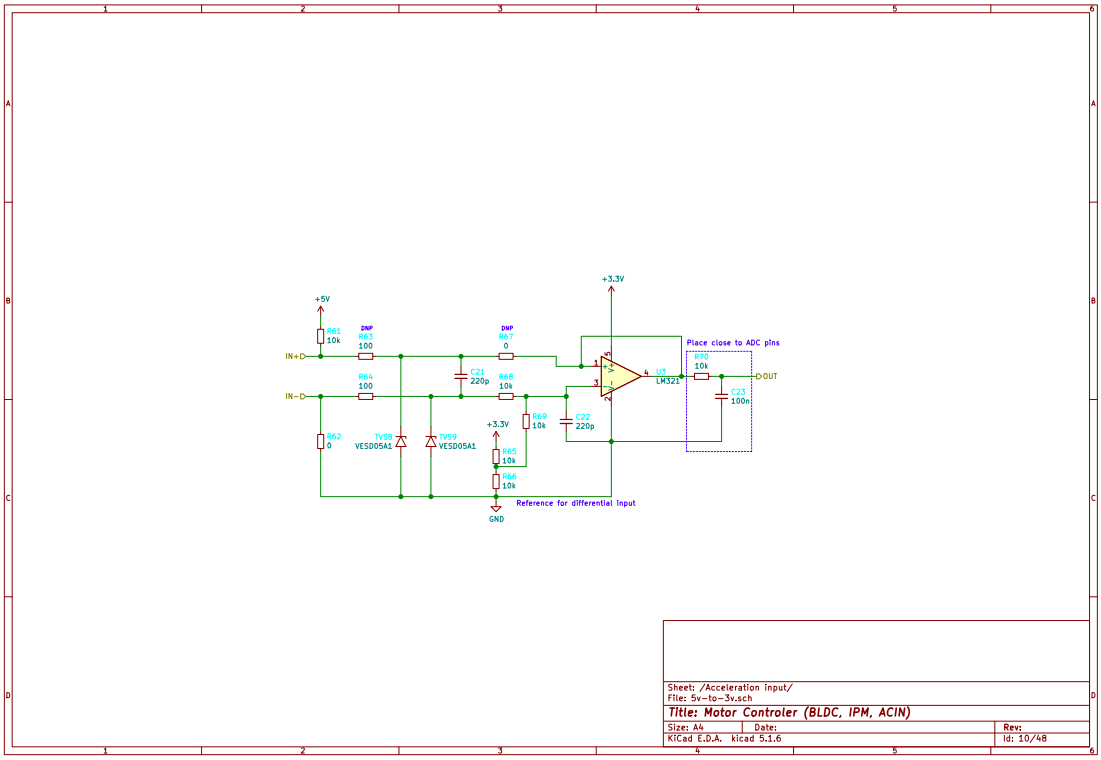
1.. Prilog 1 - Električna shema predloženog upravljača elektromotora

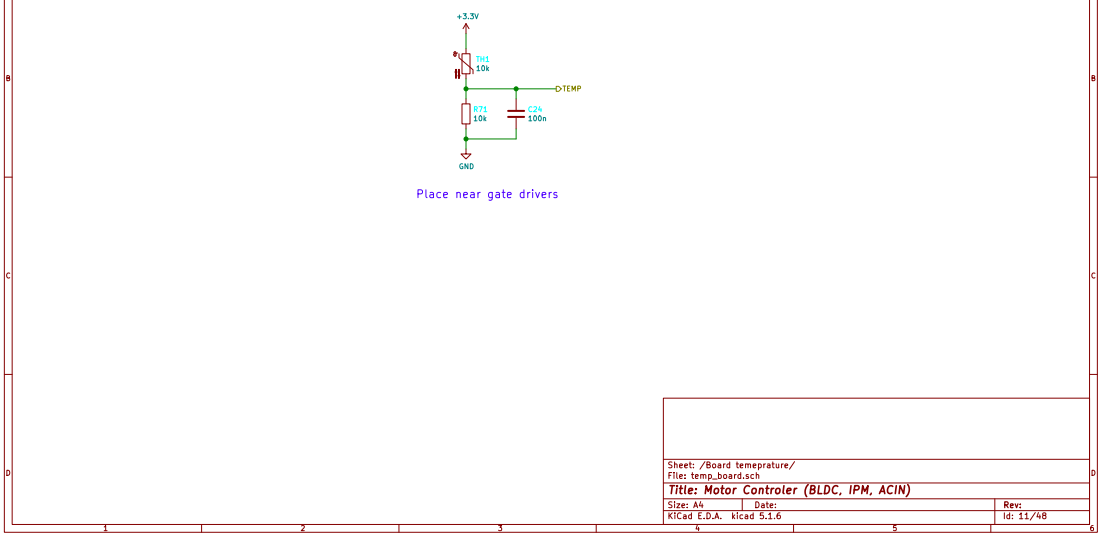


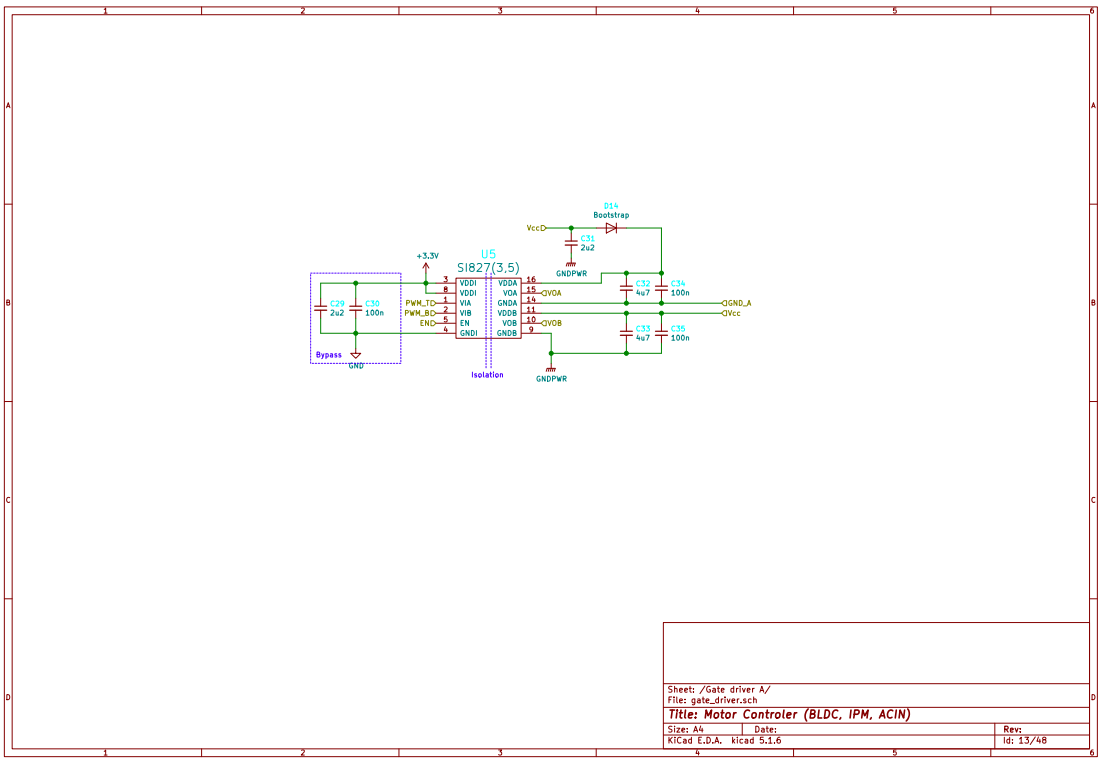

DRAIN
SOURCED
GATED
R36
10
R37
R
C6
C
Q3
IXTP60N20T
32mOhm
Vgs=20V
Qg=73nC
Qgs=22nC
Qgd=22nC
Pgate = Pdrive + (Qgate * fsw * Vgate) + (Cge * fsw * Vgate^2)
fsw(max) = 40 kHz
Vgate = 15V
Pdriver = 0.6W
Pgate = 0.6W + ... = 0.85W
1W
Subber if needed.
Sheet: /tran_B/
File: translator.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 2/48

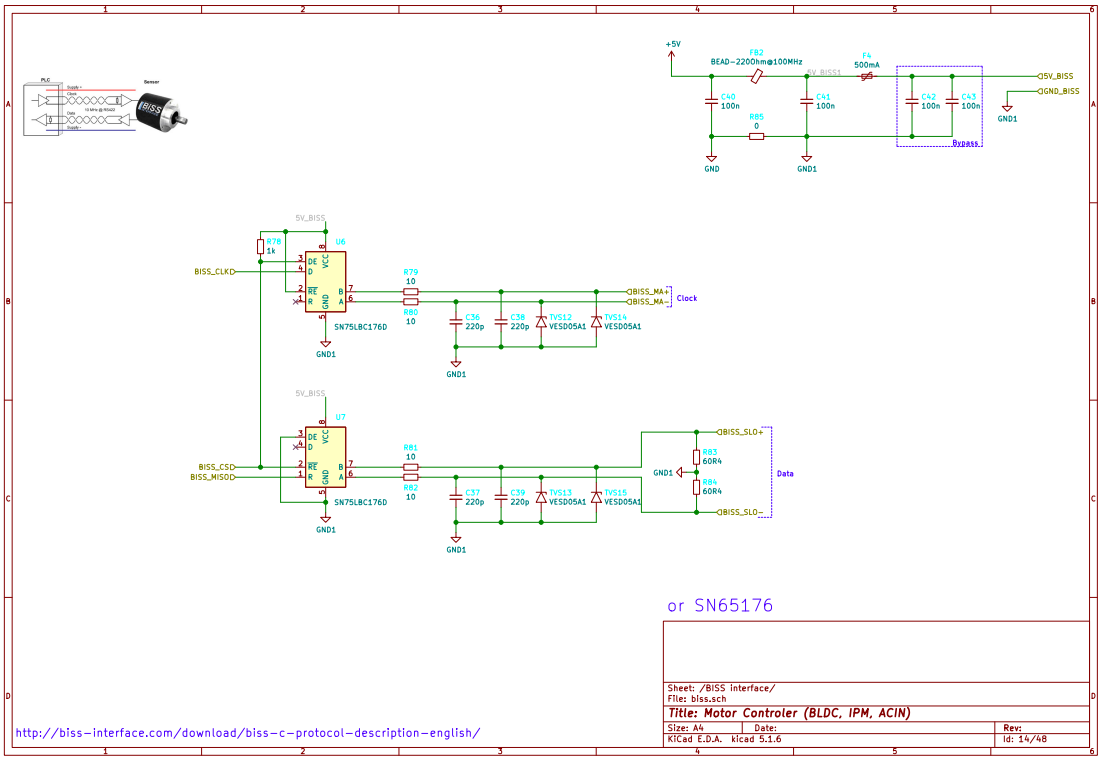

+3.3V
C14
100n
C15
10n
GND
+3.3V
R48
10k
R49
100
CAN_TXD
CAN_RX
R50
100
C12
220p
+3.3V
U1
TXD
RXD
VCC
GND
CANH
VREF
CANL
SN65HVD1050D
GND
R51
0
R52
60R4
R53
60R4
C13
100n
GND
TVS1
VESD05A1
TVS2
VESD05A1
CAN_H
CAN_L
Sheet: /CAN2/
File: CAN.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 8/48

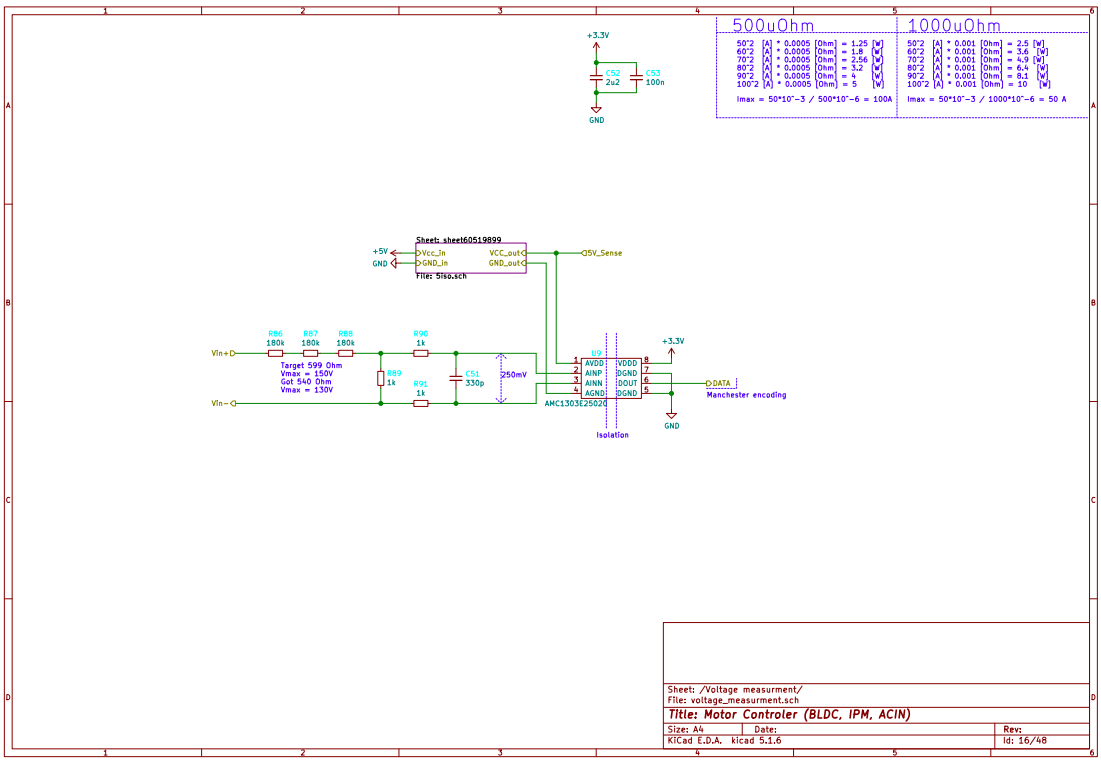

Hall input
HALL1_IND
HALL2_IND
HALL3_IND
R54 10k
R55 10k
R56 10k
HALL1_OUT
HALL2_OUT
HALL3_OUT
TVS3 VESD05A1
TVS4 VESD05A1
TVS6 VESD05A1
+3.3V
D11 BAT54A
D12 BAT54A
D13 BAT54A
GND
NTC input
MOTOR_NTC+
MOTOR_NTC-
F2 500mA
F3 500mA
C16 10n
R57 10k
C17 100n
C18 10n
R58 10k
R59 1k
TVS5 VESD05A1
TVS7 VESD05A1
U2 LM321
Place close to ADC PIN
R60 1k
C19 100n
TEMP_OUT
Bypass
C20 100n
Sheet: /Hall, Temperature sensor/
File: hall_temp.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
KiCad E.D.A. kicad 5.1.6
Rev:
Id: 9/48

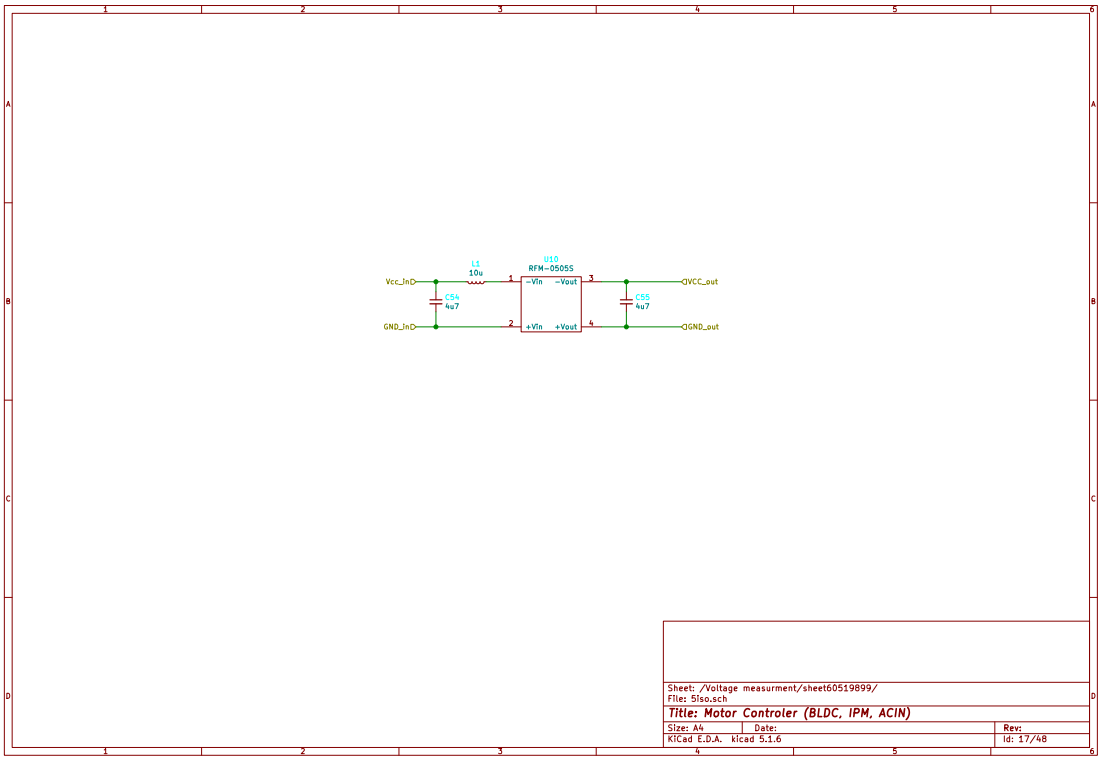

+5V
+3.3V
IN+
IN-
R61 10k
DNP
R63 100
R64 100
R62 0
DNP
R67 0
C21 220p
R68 10k
R69 10k
C22 220p
TVS8 VESD05A1
TVS9 VESD05A1
R65 10k
R66 10k
Reference for differential input
Place close to ADC pins
R70 10k
C23 100n
OUT
U3 LM321
Sheet: /Acceleration input/
File: 5v-to-3v.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
KiCad E.D.A. kicad 5.1.6
Rev:
Id: 10/48

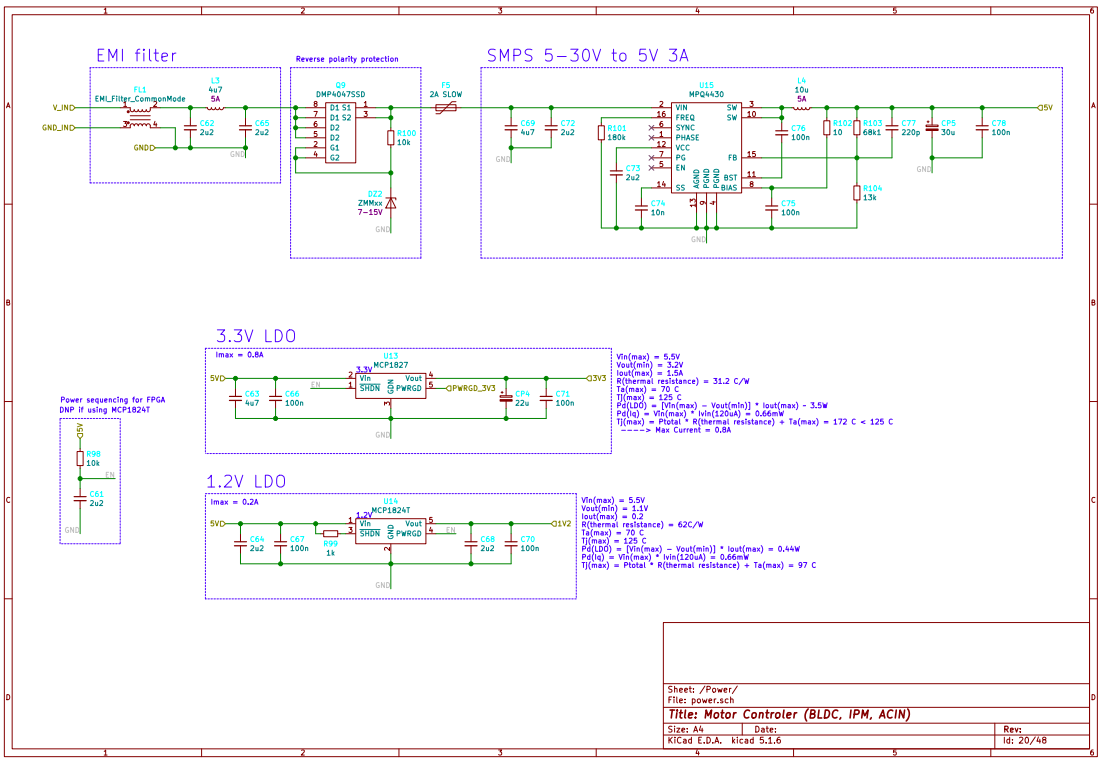

+3.3V
TH1
10k
TEMP
R71
10k
C24
100n
GND
Place near gate drivers
Sheet: /Board temeprature/
File: temp_board.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 11/48

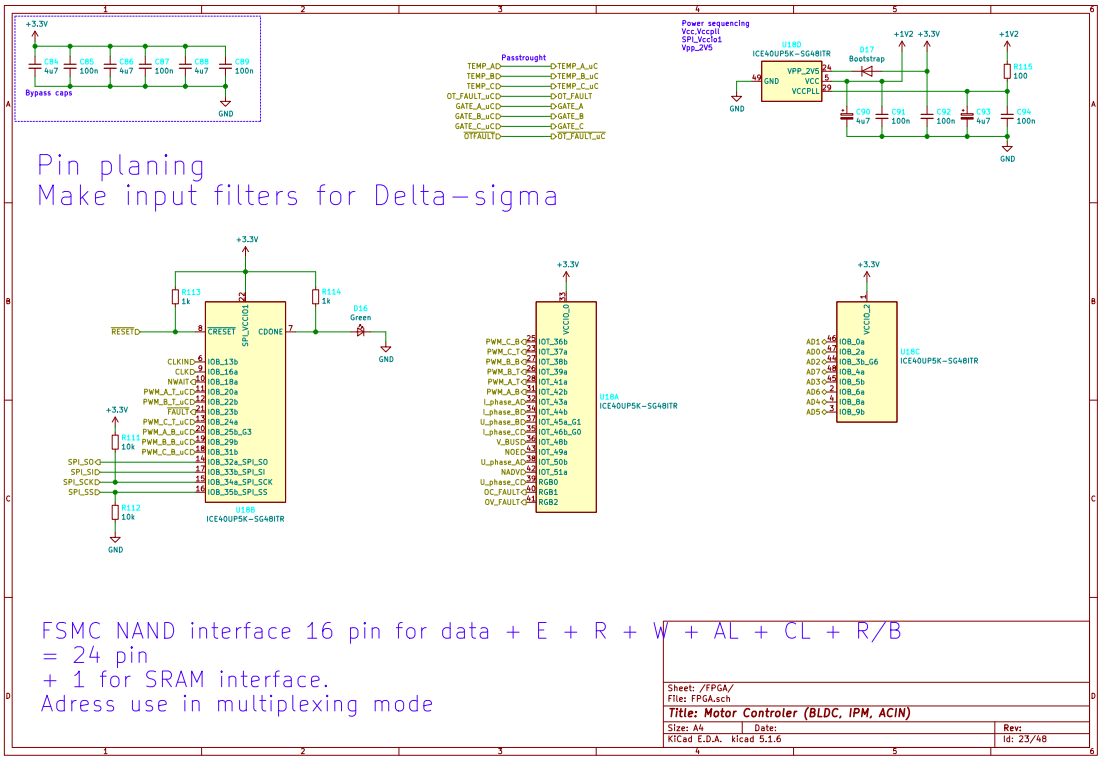

D14
Bootstrap
VccD
C31
2u2
GNDPWR
U5
SI827(3,5)
+3.3V
C29
2u2
C30
100n
Bypass
GND
PWM_TD
PWM_BD
END
VDDI
VDDI
VIA
VIB
EN
GNDI
VDDA
VOA
GNDA
VDDB
VOB
GNDB
VOA
VOB
C32
4u7
C34
100n
C33
4u7
C35
100n
GND_A
Vcc
Isolation
GNDPWR
Sheet: /Gate driver A/
File: gate_driver.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 13/48

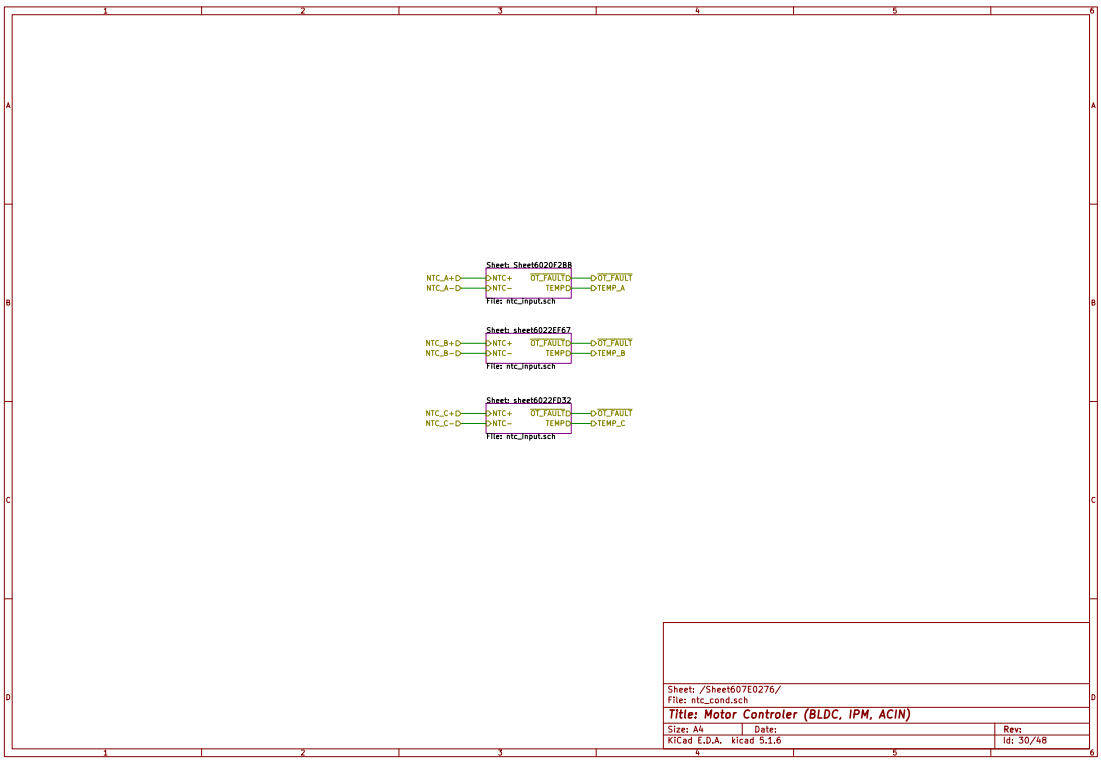

PLC
Sensor
BISS
+5V
FB2
BEAD-2200hm@100MHz
5V_BISS1
F4
500mA
C40
100n
C41
100n
C42
100n
C43
100n
R85
0
Bypass
5V_BISS
GND
GND1
GND_BISS
U6
U7
R78
1k
BISS_CLK
BISS_CS
BISS_MISO
SN75LBC176D
R79
10
R80
10
R81
10
R82
10
C36
220p
C38
220p
C37
220p
C39
220p
TVS12
VESD05A1
TVS14
VESD05A1
TVS13
VESD05A1
TVS15
VESD05A1
BISS_MA+
BISS_MA-
Clock
BISS_SLO+
BISS_SLO-
R83
60R4
R84
60R4
Data
or SN65176
Sheet: /BISS interface/
File: biss.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 14/48
http://biss-interface.com/download/biss-c-protocol-description-english/

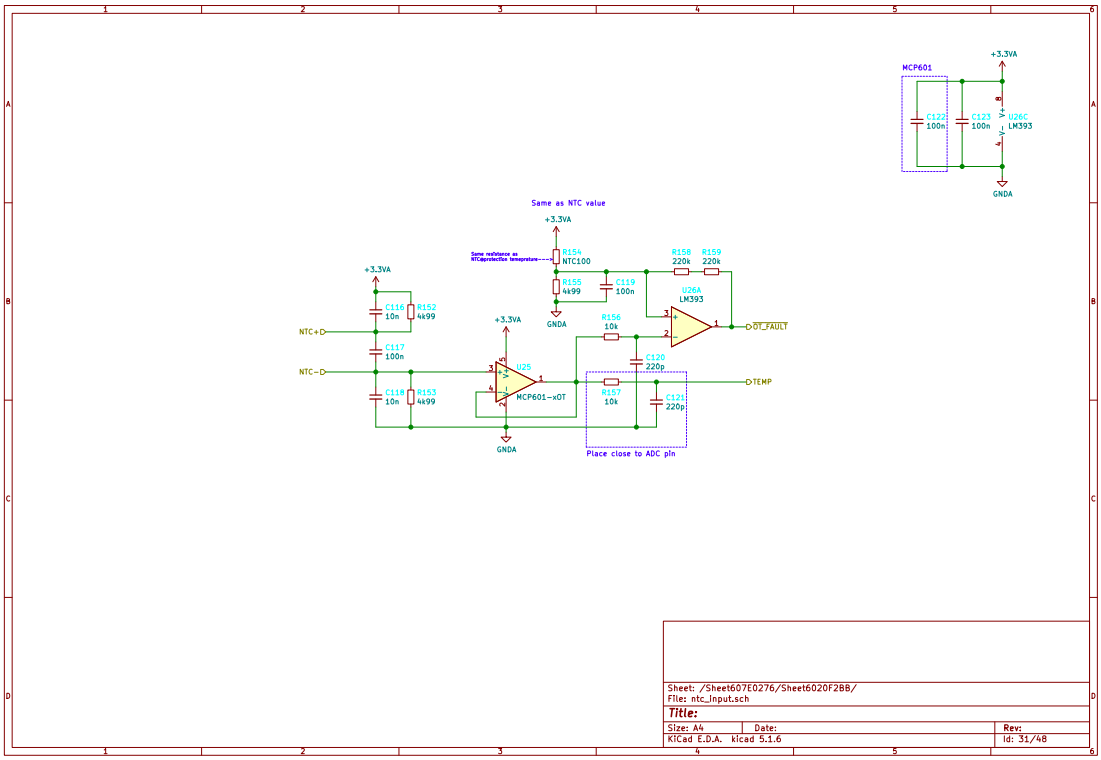

+3.3V
C52
2u2
C53
100n
GND
500uOhm
50^2 [A] * 0.0005 [Ohm] = 1.25 [W]
60^2 [A] * 0.0005 [Ohm] = 1.8 [W]
70^2 [A] * 0.0005 [Ohm] = 2.56 [W]
80^2 [A] * 0.0005 [Ohm] = 3.2 [W]
90^2 [A] * 0.0005 [Ohm] = 4 [W]
100^2 [A] * 0.0005 [Ohm] = 5 [W]
Imax = 50*10^-3 / 500*10^-6 = 100A
1000uOhm
50^2 [A] * 0.001 [Ohm] = 2.5 [W]
60^2 [A] * 0.001 [Ohm] = 3.6 [W]
70^2 [A] * 0.001 [Ohm] = 4.9 [W]
80^2 [A] * 0.001 [Ohm] = 6.4 [W]
90^2 [A] * 0.001 [Ohm] = 8.1 [W]
100^2 [A] * 0.001 [Ohm] = 10 [W]
Imax = 50*10^-3 / 1000*10^-6 = 50 A
Sheet: sheet60519899
File: 5150.sch
+5V
GND
Vcc_in
GND_in
VCC_out
GND_out
5V_Sense
R86
180k
R87
180k
R88
180k
R90
1k
R89
1k
R91
1k
C51
330p
250mV
Vin+
Vin-
Target 599 Ohm
Vmax = 150V
Got 540 Ohm
Vmax = 130V
U9
AMC1303E2520
AVDD
AINP
AINN
AGND
VDDD
DGND
DOUT
DGND
+3.3V
GND
DATA
Manchester encoding
Isolation
Sheet: /Voltage measurment/
File: voltage_measurment.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 16/48

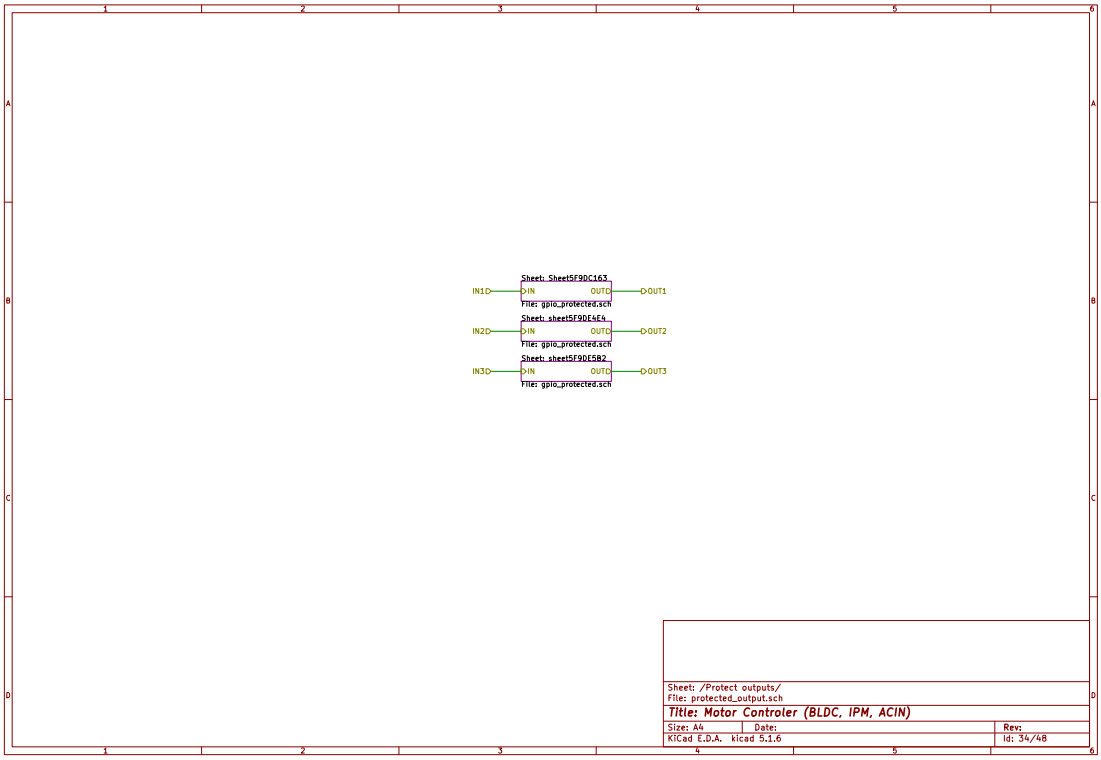

L1
10u
U10
RFM-0505S
Vcc_In
GND_In
-Vin
-Vout
+Vin
+Vout
C54
4u7
C55
4u7
VCC_out
GND_out
Sheet: /Voltage measurment/sheet60519899/
File: 5iso.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 17/48

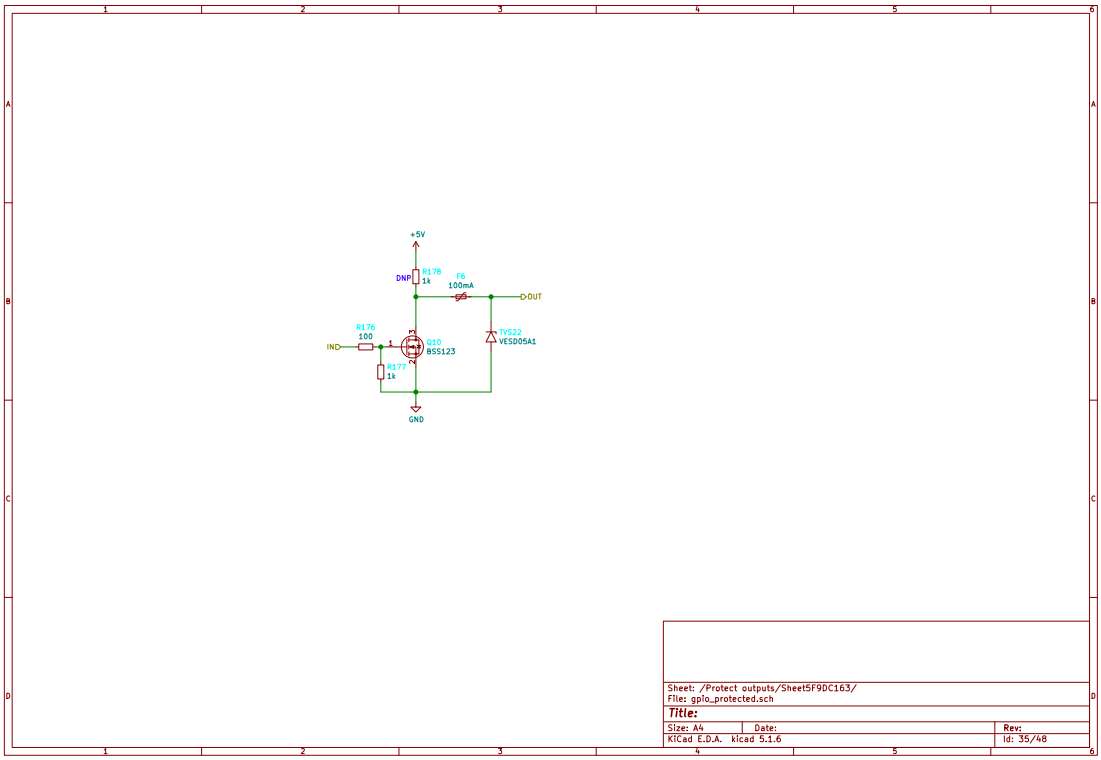

EMI filter
Reverse polarity protection
SMPS 5-30V to 5V 3A
FL1
EMI_Filter_CommonMode
L3
4u7
5A
V_IN
GND_IN
GNDD
C62
2u2
C65
2u2
Q9
DMP4047SSD
R100
10k
DZ2
ZMMxx
7-15V
F5
2A SLOW
U15
MPQ4430
VIN
FREQ
SYNC
PHASE
VCC
PG
EN
SS
SW
FB
BST
AGND
PGND
PGND
BIAS
L4
10u
5A
C69
4u7
C72
2u2
R101
180k
C73
2u2
C74
10n
C76
100n
C75
100n
R102
10
R103
68k1
R104
13k
C77
220p
CP5
30u
C78
100n
5V
3.3V LDO
Imax = 0.8A
U13
MCP1827
3.3V
5VD
3V3
EN
PWRGD_3V3
C63
4u7
C66
100n
CP4
22u
C71
100n
Vin(max) = 5.5V
Vout(min) = 3.2V
Iout(max) = 1.5A
R(thermal resistance) = 31.2 C/W
Ta(max) = 70 C
Tj(max) = 125 C
Pd(LDO) = [Vin(max) - Vout(min)] * Iout(max) = 3.5W
Pd(Iq) = Vin(max) * Iin(120uA) = 0.66mW
Tj(max) = Ptotal * R(thermal resistance) + Ta(max) = 172 C < 125 C
----> Max Current = 0.8A
Power sequencing for FPGA
DNP if using MCP1824T
R98
10k
C61
2u2
1.2V LDO
Imax = 0.2A
U14
MCP1824T
1V2
C64
2u2
C67
100n
R99
1k
C68
2u2
C70
100n
Vin(max) = 5.5V
Vout(min) = 1.1V
Iout(max) = 0.2
R(thermal resistance) = 62C/W
Ta(max) = 70 C
Tj(max) = 125 C
Pd(LDO) = [Vin(max) - Vout(min)] * Iout(max) = 0.44W
Pd(Iq) = Vin(max) * Iin(120uA) = 0.66mW
Tj(max) = Ptotal * R(thermal resistance) + Ta(max) = 97 C
Sheet: /Power/
File: power.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 20/48

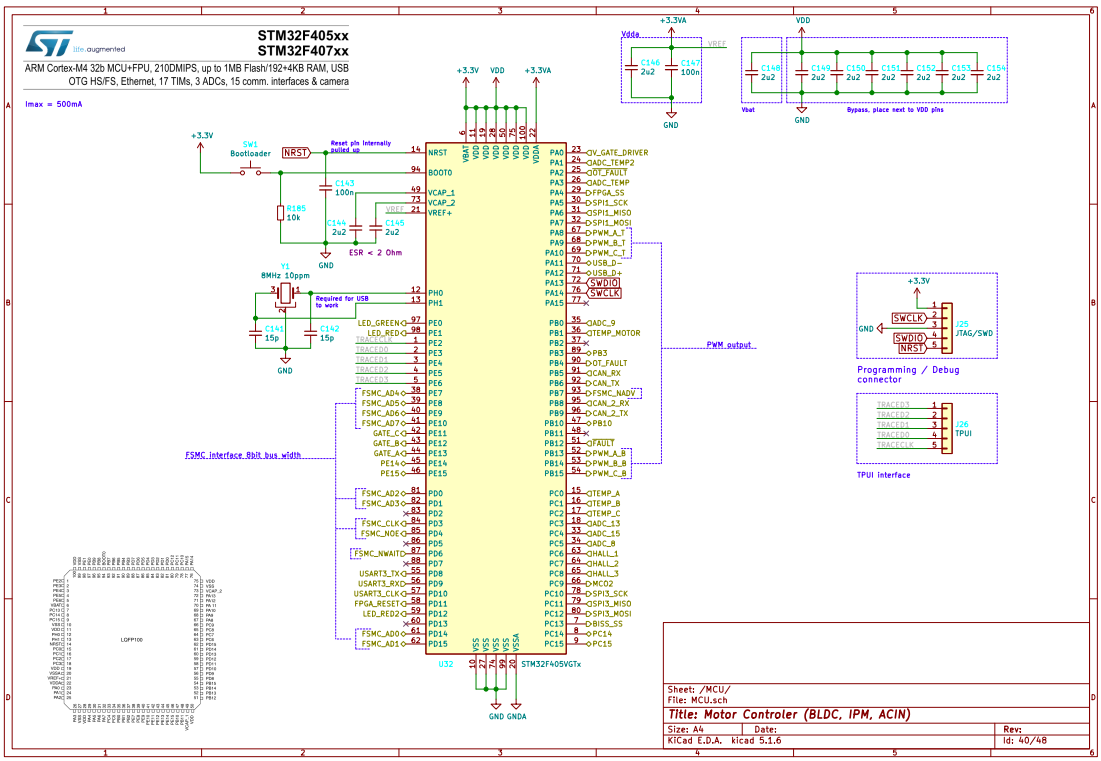

Bypass caps
Passtrought
Power sequencing
Vcc_Vccpll
SPI_Vccio1
Vpp_2V5
Bootstrap
Pin planing
Make input filters for Delta–sigma
FSMC NAND interface 16 pin for data + E + R + W + AL + CL + R/B
= 24 pin
+ 1 for SRAM interface.
Adress use in multiplexing mode
Sheet: /FPGA/
File: FPGA.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 23/48

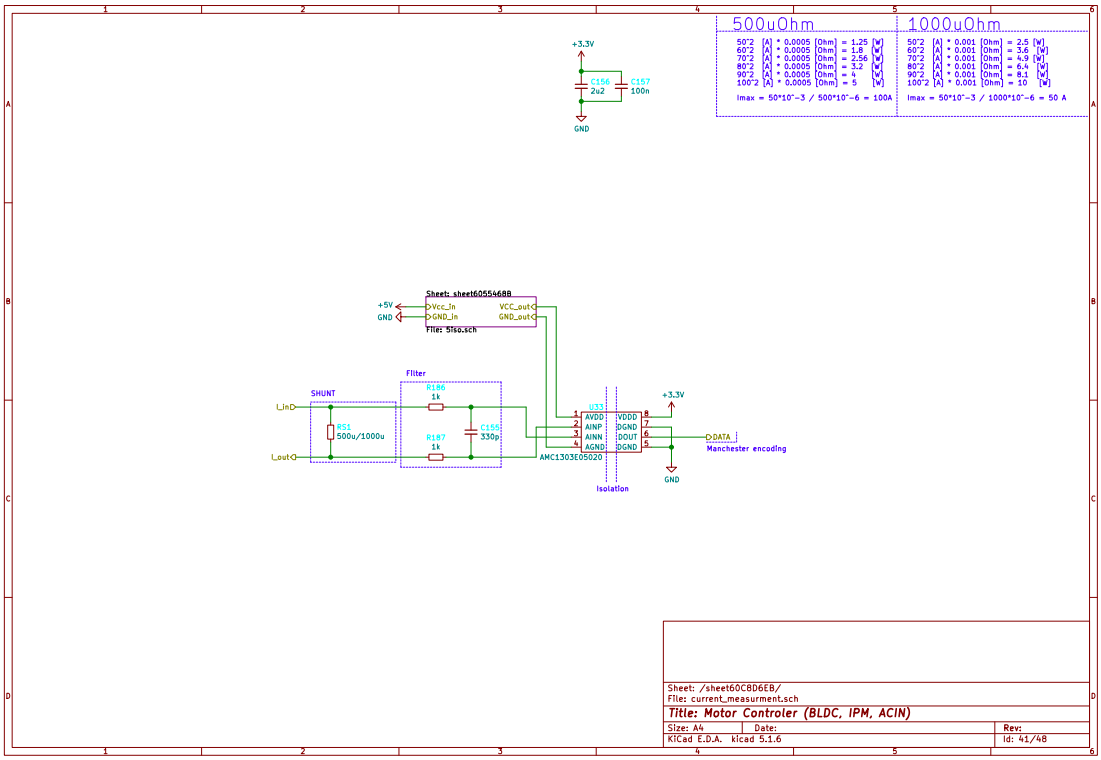

Sheet: Sheet6020F2BB
Sheet: sheet6022EE67
Sheet: sheet6022ED32
File: ntc_input.sch
Sheet: /Sheet607E0276/
File: ntc_cond.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 30/48

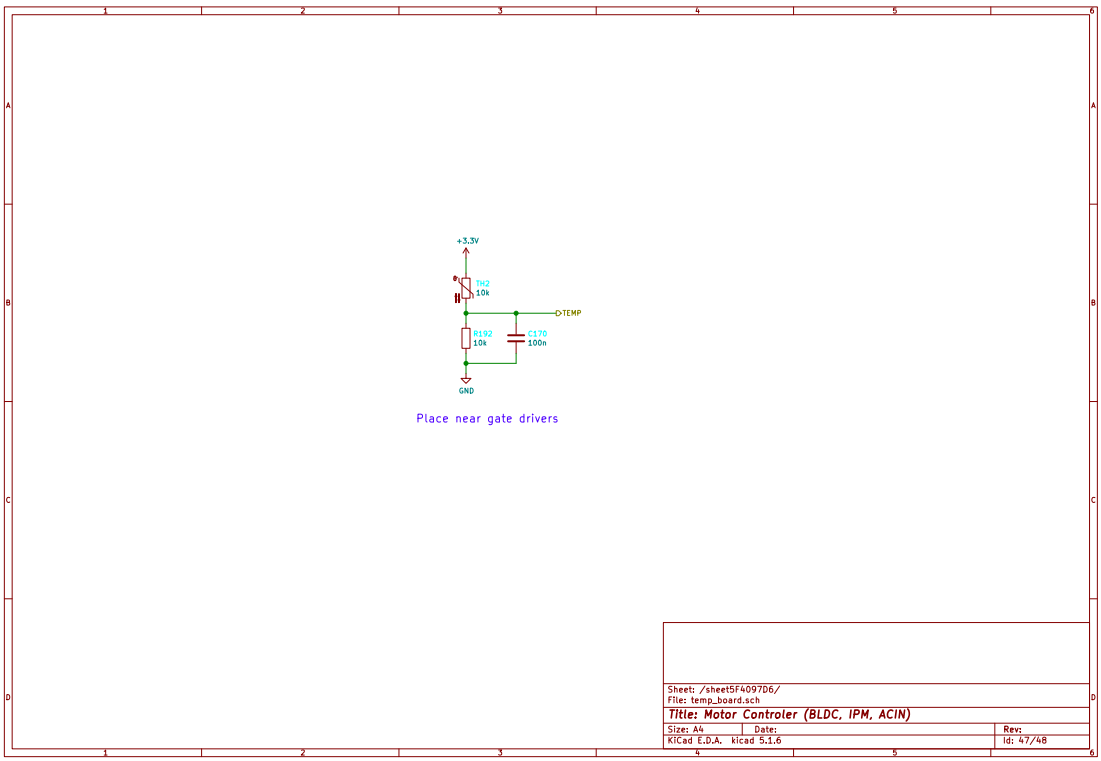

MCP601
+3.3VA
C122
100n
C123
100n
U26G
LM393
GNDA
Same as NTC value
+3.3VA
Same resistance as NTC@protective temperature
R154
NTC100
R155
4k99
C119
100n
GNDA
R158
220k
R159
220k
U26A
LM393
R156
10k
OT_FAULT
+3.3VA
C116
10n
R152
4k99
NTC+
C117
100n
NTC-
C118
10n
R163
4k99
U25
MCP601-xOT
C120
220p
R157
10k
C121
220p
TEMP
GNDA
Place close to ADC pin
Sheet: /Sheet607E0276/Sheet6020F2BB/
File: ntc_input.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 31/48


Sheet: Sheet5F9DC163
IN1
IN
OUT
OUT1
File: gpio_protected.sch
Sheet: sheet5F9DEAE4
IN2
IN
OUT
OUT2
File: gpio_protected.sch
Sheet: sheet5F9DE5B2
IN3
IN
OUT
OUT3
File: gpio_protected.sch
Sheet: /Protect outputs/
File: protected_output.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 34/48

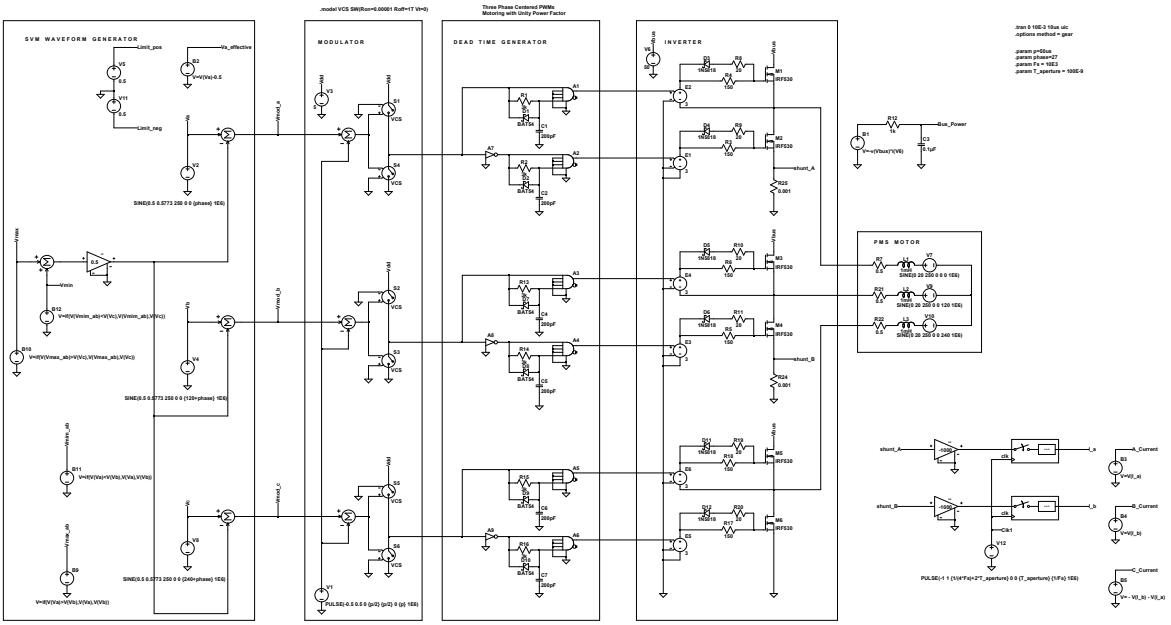

+5V
DNP
R178
1k
F6
100mA
OUT
R176
100
IND
Q10
BSS123
TVS22
VESD05A1
R177
1k
GND
Sheet: /Protect outputs/Sheet5F9DC163/
File: gpio_protected.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 35/48


STM32F405xx
STM32F407xx
ARM Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, USB
OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera
Imax = 500mA
+3.3V
VDD
+3.3VA
Vdda
VREF
C146
2u2
C147
100n
C148
2u2
C149
2u2
C150
2u2
C151
2u2
C152
2u2
C153
2u2
C154
2u2
Vbat
Bypass, place next to VDD pins
SW1
Bootloader
NRST
Reset pin internally pulled up
R185
10k
C143
100n
C144
2u2
C145
2u2
ESR < 2 Ohm
Y1
8MHz 10ppm
Required for USB to work
C141
15p
C142
15p
BOOT0
VCAP_1
VCAP_2
VREF+
PWM output
FSMC interface 8bit bus width
U32
STM32F405VGTx
LQFP100
SWCLK
SWDIO
NRST
J25
JTAG/SWD
Programming / Debug connector
TRACED3
TRACED2
TRACED1
TRACED0
TRACECLK
J26
TPUI
TPUI interface
Sheet: /MCU/
File: MCU.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 40/48


500uOhm
50^2 [A] * 0.0005 [Ohm] = 1.25 [W]
60^2 [A] * 0.0005 [Ohm] = 1.8 [W]
70^2 [A] * 0.0005 [Ohm] = 2.56 [W]
80^2 [A] * 0.0005 [Ohm] = 3.2 [W]
90^2 [A] * 0.0005 [Ohm] = 4 [W]
100^2 [A] * 0.0005 [Ohm] = 5 [W]
Imax = 50*10^-3 / 500*10^-6 = 100A
1000uOhm
50^2 [A] * 0.001 [Ohm] = 2.5 [W]
60^2 [A] * 0.001 [Ohm] = 3.6 [W]
70^2 [A] * 0.001 [Ohm] = 4.9 [W]
80^2 [A] * 0.001 [Ohm] = 6.4 [W]
90^2 [A] * 0.001 [Ohm] = 8.1 [W]
100^2 [A] * 0.001 [Ohm] = 10 [W]
Imax = 50*10^-3 / 1000*10^-6 = 50 A
+3.3V
C156
2u2
C157
100n
GND
Sheet: sheet6055468B
File: 5iso.sch
+5V
GND
Vcc_in
GND_in
VCC_out
GND_out
Filter
SHUNT
I_in
I_out
RS1
500u/1000u
R186
1k
R187
1k
C155
330p
U33
AVDD
AINP
AINN
AGND
VDDD
DGND
DOUT
DGND
AMC1303E05020
Isolation
+3.3V
GND
DATA
Manchester encoding
Sheet: /sheet60C8D6EB/
File: current_measurment.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 41/48


+3.3V
TH2
10k
TEMP
R192
10k
C170
100n
GND
Place near gate drivers
Sheet: /sheet5F4097D6/
File: temp_board.sch
Title: Motor Controler (BLDC, IPM, ACIN)
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.6
Id: 47/48

2.. Prilog 2 - Shema LTSpice simulacije vektorske modulacije napona



3.. Prilog 3 - Manchester dekoder

3..1. Manchester dekoder Verilog kod

prilozi/md.v

```
/*
    ********************************************************************************

*     File Name:  md.v
*     Model:  Manchester Decoder
********************************************************************************
    */

module md (
    input clk_in,
    input mdi,
    output clk_out,
    output out
);

// Buffers for edge detection
reg mdi1;
reg mdi2;
// Counter and delay(in input clock ticks)
reg [5:0]count;
localparam delay = 13;
// Decoded data
reg nrz;
reg clk;
```

```
// Initalization
initial begin
    nrz <= 1'b0;
    clk <=1'b0;
    count <= 0;
end

// Assign outputs
assign out = nrz;
assign clk_out = clk;

// Generate 2 FF register to accept serial Manchester data in
always @(posedge clk_in)
begin
    begin
        mdi2 <= mdi1 ;
        mdi1 <= mdi ;
    end
end

// Data decoding
always @(posedge clk_in)
begin
    // Start clk on rising or falling edge
    if (((!mdi1 && mdi2)||(!mdi2 && mdi1)) && clk == 1'b0)
    begin
        clk <= ~clk;
    end
    // Delay clock 3/4 of T
    if (clk > 0)
    begin
        if (count < delay)
        begin
            count++;
        end
        else
        begin
            nrz <= ~mdi;
```

```
            count <= 0;
            clk <= ~clk;
        end
    end
end
endmodule
```

3..2. Manchester dekoder Verilog test kod

prilozi/md_tf.v

```
`timescale 1 ns / 1 ns
`include "md.v"

module md_tf ;
reg rst ;
reg clk16x ;
reg mdi ;
wire clk;
wire out;

md u1 (clk16x,mdi,clk,out) ;

initial begin
    rst = 1'b0 ;
    clk16x = 1'b0 ;
    mdi = 1'b0 ;
end

parameter clock_period = 10;
always #(clock_period/2) clk16x = ~clk16x;
initial begin
    $display("Verilog simulation of Manchester decoder\n") ;
    $dumpfile("md_tf.vcd");
    $dumpvars(0,u1);

    #1 rst = 1'b1 ;
    #100 rst = 1'b0 ;
```

```
    // Input 8 logic 0s
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;
    #80 mdi = 1'b1 ;
    #80 mdi = 1'b0 ;

    $display ("\nSimulation of Manchester decoder is complete.") ;
    $finish ;
end
endmodule
```

4.. Prilog 4 - $sinc^3$ filter

4..1. $sinc^3$ Verilog kod

prilozi/sinc3.v

```
`timescale 1 ns / 100 ps

module dec256sinc24b
(
    input                       reset_i,
    input                       mclkout_i,
    input                       mdata_i,
    output                      data_rdy_o,     // signals when new data
   is available
    output reg  [15:0]          data_o          // outputs filtered data
);

reg ip_data1;
reg [23:0]  acc1;
reg [23:0]  acc2;
reg [23:0]  acc3;
reg [23:0]  acc3_d1;
reg [23:0]  acc3_d2;
reg [23:0]  diff1;
reg [23:0]  diff2;
reg [23:0]  diff3;
reg [23:0]  diff1_d;
reg [23:0]  diff2_d;
reg [7:0]   word_count;
reg         word_clk;
```

```
assign data_rdy_o = word_clk;

/*ACCUMULATOR (INTEGRATOR)
* Perform the accumulation (IIR) at the speed of the modulator.
* mclkout_i = modulators conversion bit rate */
always @(negedge mclkout_i or posedge reset_i)
begin
    if( reset_i == 1'b1 )
    begin
        /*initialize acc registers on reset*/
        acc1    <= 0;
        acc2    <= 0;
        acc3    <= 0;
    end
    else
    begin
        /*perform accumulation process*/
        acc1    <= acc1 + mdata_i;
        acc2    <= acc2 + acc1;
        acc3    <= acc3 + acc2;
    end
end

/*DECIMATION STAGE (MCLKOUT_I/ WORD_CLK) */
always@(posedge mclkout_i or posedge reset_i )
begin
    if(reset_i == 1'b1)
    begin
        word_count  <= 0;
    end
    else
    begin
        word_count <= word_count + 1;
    end
end
```

```
always @(word_count)
begin
    word_clk <= word_count[7];
end

/*DIFFERENTIATOR (including decimation stage)
* Perform the differentiation stage (FIR) at a lower speed.
WORD_CLK = output word rate */
always @(posedge word_clk or posedge reset_i)
begin
    if(reset_i == 1'b1)
    begin
        acc3_d2 <= 0;
        diff1_d <= 0;
        diff2_d <= 0;
        diff1   <= 0;
        diff2   <= 0;
        diff3   <= 0;
    end
    else
    begin
        diff1   <= acc3 - acc3_d2;
        diff2   <= diff1 - diff1_d;
        diff3   <= diff2 - diff2_d;
        acc3_d2 <= acc3;
        diff1_d <= diff1;
        diff2_d <= diff2;
    end
end

/*  Clock the Sinc output into an output register
    Clocking Sinc Output into an Output Register
WORD_CLK = output word rate */
always @(posedge word_clk)
begin
    data_o[15]  <= diff3[23];
    data_o[14]  <= diff3[22];
    data_o[13]  <= diff3[21];
```

```
    data_o[12]  <= diff3[20];
    data_o[11]  <= diff3[19];
    data_o[10]  <= diff3[18];
    data_o[9]   <= diff3[17];
    data_o[8]   <= diff3[16];
    data_o[7]   <= diff3[15];
    data_o[6]   <= diff3[14];
    data_o[5]   <= diff3[13];
    data_o[4]   <= diff3[12];
    data_o[3]   <= diff3[11];
    data_o[2]   <= diff3[10];
    data_o[1]   <= diff3[9];
    data_o[0]   <= diff3[8];
end

endmodule
```

4..2. $sinc^3$ Verilog test kod

prilozi/sinc3_file_tb.v

```
`default_nettype none
`timescale 10 ns / 1 ns
`include "sinc3.v"

module main_tb;

// Simulation time: 100ns (10 * 10ns)
parameter DURATION = 10;
localparam SIZE = 500000;

// integer fin;
integer i;
reg d[SIZE:0];
// Input/Output
reg clk;
reg data;
reg reset;
```

```
wire [15:0] data_out;
wire data_rdy;

// Module instance
dec256sinc24b SINC(
    reset,
    clk,
    data,
    data_rdy,
    data_out
);

// Clock signal
always #1 clk = ~clk;
initial begin
    // File were to store the simulation results
    $dumpfile("sinc3_file.vcd");
    $dumpvars(0, SINC);

    clk = 0;
    #1 data = 0;
    #1 reset = 1;
    #1 reset = 0;
    #5 reset = 1;
    #1 reset = 0;
    $readmemh("ds", d);
    for(i = 0; i < SIZE ; i++)
    begin
        #1 data =  d[i];
    end

    #(100) $display("End of simulation");
    $finish;
end

endmodule
```