SKRIPSI

SISTEM KENDALI DIGITAL BERBASIS FPGA UNTUK MENGENDALIKAN MOTOR ARUS SEARAH TANPA SIKAT

Disusun dan diajukan oleh

NURUL HIDAYAT

D041 17 1319

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

KATA PENGANTAR

Segala puji hanya milik Allah Subhanahu Wa Ta'ala yang telah memberikan rahmat, hidayah, taufik dan pertolongan-Nya dalam menyelesaikan skripsi ini yang berjudul " **SISTEM KENDALI DIGITAL BERBASIS FPGA UNTUK MENGENDALIKAN MOTOR ARUS SEARAH TANPA SIKAT**" sebagai salah satu syarat dalam menyelesaikan jenjang Sarjana pada Departemen Teknik Elektro Fakultas Teknik Universitas Hasanuddin. Sholawat serta salam semoga tetap tercurahkan kepada Nabi Muhammad Shallallahu 'alaihi Wa Sallam, beserta keluarga dan para sahabatnya yang telah membimbing dan menuntun kita dari zaman jahilia menuju zaman yang beradap.

Penulis menyadari bahwa tanpa bantuan dari berbagai pihak, sangatlah sulit untuk menyelesaikan skripsi ini. Oleh karenanya, penulis berterima kasih kepada kedua orang tua yang senantiasa selalu memberikan dukungan moril maupun materil serta doa yang senantiasa terus dipanjatkan. Ucapan terima kasih juga kepada saudara saya yang turut memberikan dukungan dalam bentuk doa dan senda gurau serta semangat agar penulis tetap ceria dan semangat dalam menyelesaikan skripsi ini.

Penulis juga menghaturkan ucapan terima kasih kepada Bapak **Prof. Dr.-Ing. Faizal Arya Samman, ST., MT** selaku pembimbing I dan Bapak **Dr. Ir. Rhiza S. Sadjad, MSEE.** selaku pembimbing II dalam kesediaannya menyisihkan waktu, tenaga, dan ilmu dalam segala permasalahan dalam penulisan skripsi ini, serta seluruh tim penguji yang banyak memberikan arahan dan kritik yang membangun agar skripsi ini semakin baik.

Penulis tentu tidak lupa mengucapkan terima kasih kepada teman – teman seperjuangan atas kerjasama dan kekompakannya selama ini, serta teman – teman di Laboratorium *Laboratorium Elektronika dan Divais Departemen Teknik Elektro* yang telah berbaik hati berbagi pengalaman dan masukan. Semoga kedepannya kita tetap menjaga kekompakan ini.

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna. Dengan demikian, penulis tetap mengharapkan saran dan kritik dengan harapan semoga tulisan ini bisa memberikan manfaat kepada seluruh pihak. Akhir kata penulis mendoakan semoga Allah Subhana Wa Ta'ala terus memberikan taufik dan hidayah – Nya kepada semua pihak untuk dapat terus melakukan terobosan – terobosan dan inovasi baru dalam peningkatan kualitas ilmu pengetahuan. Aamiin ya Rabbal Alamin…

Gowa, 9 Mei 2022

Nurul Hidayat

ABSTRAK

NURUL HIDAYAT, *Sistem Kendali Digital berbasis FPGA untuk Mengendalikan Motor Arus Searah Tanpa Sikat* "dibimbing oleh Faizal Arya Samman dan Rhiza S. Sadjad".

Pada penelitian ini dirancang pengendali digital berbasis FPGA untuk mengendalikan kecepatan motor arus searah tanpa sikat (BLDC). Digunakan metode *trapezoid control* atau 6-step komutasi dengan unipolar switching pwm. Metode komutasi motor BLDC berdasarkan sensor hall efek yang tertanam dalam motor bldc. metode ini diimplementasikan dengan menggunakan model mesin keadaan. Selain metode control, didesain pula modul untuk mengukur kecepatan motor bldc. Pengukur kecepatan menggunakan counter untuk menghitung siklus electric motor dan look up tabel untuk menyimpan hasil perhitungan revolusi per menit motor. Agoritma control dan pengukur kecepatan di tulis menggunakan Verilog HDL (*hardware description language*) dan diverifikasi melaui simulasi menggunakan modelsim-intel. Kemudian, diimplementasikan pada board FPGA terasic DE0-NANO EPCE22. Semua sistem dibangun berdasarkan base clock fpga 50 MHz. Pengujian kendali motor dilakukan pada motor bldc 350 watt, 36 v dengan inverter tiga fasa sebagai drivernya. Hasil yang diperoleh cukup memuaskan, kecepatan motor bldc dapat diatur dengan piranti potensiometer hingga spesifikasi kecepatan maksimum motor BLDC.

Kata Kunci: Motor Arus Searah Tanpa Sikat, *Field Programmable Gate Array* (FPGA), *Trepezoid Control*, *Unipolar Independent PWM Switching*

ABSTRACT

NURUL HIDAYAT, *FPGA Based Digital Control Design for BLDC Motor Control* "supervised by Faizal Arya Samman and Rhiza S. Sadjad".

This research presents the design of digital control based on FPGA to control the speed of a brushless direct current (BLDC) motor. The control algorithm is using trapezoid control or six-step commutation and unipolar independent PWM switching, the commutation itself depends on 3 built-in hall effect sensors. This method is implemented on FPGA using a state machine model. Besides the speed control algorithm, this paper also presents an algorithm to calculate the speed of the BLDC motor. The speed calculation builds up with a counter to count the electric cycles for one second then the result is stored in the register and look-up table to convert the electric cycles data into revolutions per minute (rpm) data. The speed control and speed calculation are written using Verilog hardware description language (HDL) and verified through simulation using ModelSim-intel, the code is implemented using the FPGA DE0-nano EP4CE22 board provided by terasic. Motor control testing was carried out on a 350 watt 36 v BLDC motor with a three-phase inverter as the driver. The results obtained are quite satisfactory, the speed of the BLDC motor can be adjusted with a potentiometer device to the maximum speed specification of the BLDC motor.

Keywords: Brushless Direct Current (BLDC) Motor, Field Programmable Gate Arrey (FPGA), Trepezoid Control, Unipolar Independent PWM Switching

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengunaan *Field Programmable Logic Array (FPGA)* dalam aplikasi motor control meningkat belakangan tahun terakhir, ditunjukkan pada publikasi [1] [2] [3] [4], hal ini dipengaruhi karena pemrogramannya yang fleksibel. Selain itu, dalam mengontrol motor diperlukan *high-speed switching* dengan alasan tersebut fpga digunakan [5]. Sebagai tambahan, fitur yang paling penting adalah bahwa FPGA memungkinkan untuk mengembangkan algoritma yang kompleks dan dapat melakukan proses secara parallel (*concerrent*) [6]. Adapun untuk memrogram pada FPGA diperlukan pengetahuan sistem digital dan HDL *(hardware description language).*

Dalam penelitian ini digunakan motor BLDC tiga fasa, motor ini memiliki beberapa keuntungan, seperti harganya yang relatif lebih murah, biaya perawatan yang rendah, dan menghasilkan *noise* (suara bising) yang rendah disebabkan oleh tidak adanya komutator makanik melainkan menggunakan komutator elektronik. Selain keuntungan tersebut motor BLDC juga memiliki beberapa kekurangan yaitu saat motor DC brushless dioperasikan pada kecepatan rendah, getaran kecil terjadi selama putaran kecepatan rendah. Namun, getaran berkurang pada kecepatan tinggi.

Motor DC tanpa sikat terdiri atas bagian yang diam yang disebut stator dan bagian yang bergerak disebut rotor. Ruang antara stator dan rotor disebut celah udara (*air gap*), pada stator sendiri terdiri atas belitan yang terhubung dengan konfigurasi star ataupun konfigurasi delta sedangkan pada rotor terdapat magnet permanen. penempatan rotor pada motor DC tanpa sikat pada umumnya ada dua yaitu rotor di dalam (*Inrunner*) dan rotor di luar (*Outrunner*). Keuntungan motor BLDC rotor internal terletak pada inersia rotornya yang rendah dan pembuangan panas lebih baik. Sebaliknya, untuk motor rotor eksternal, kumparan penghasil panas diisolasi dari lingkungannya oleh rumah rotor dan magnet. Motor

dengan rotor eksternal memiliki keunggulan untuk aplikasi yang diproduksi secara massal, karena dapat diproduksi dengan lebih murah.

Metode yang digunakan untuk mengontrol motor BLDC ialah metode enam step komutasi (*6-step commutation*) dengan menggunakan *unipolar switching PWM*. Komutasi motor BLDC berdasarkan pada 3 sensor hall efek yang tertanam dalam motor. metode enam step komutasi ini digunakan untuk mengontrol Inverter tiga fasa/Driver motor BLDC dalam enam rangkaian pensaklaran. Dalam metode ini hanya dua fasa motor BLDC yang diberi energy (*energized*) pada setiap urutan pensaklaran sedangkan fase lainnya tidak aktif. Pembalikan urutan switching mengubah arah putaran motor. Kecepatan motor berbanding lurus dengan lebar sinyal sensor hall effect. Terakhir metode tersebut akan didesain dan diimplementasikan pada board FPGA pada penelitian/skripsi ini yang berjudul "**Perancangan Sistem Digital Berbasis FPGA untuk Pengendalian Motor BLDC"**.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka pokok permasalahan yang akan dipecahkan dari penelitian ini sebagai berikut:

1. Bagaimana merancang kendali untuk mengatur kecepatan motor BLDC dengan *trapezoid control* pada FPGA?
2. Bagaimana merancang modul pengukur kecepatan motor BLDC dengan sensor hall efek pada FPGA?
3. Bagaimana karakteristik kecepatan motor BLDC terhadap referensi kecepatan (potensiometer) dengan metode *trapezoid control*?

1.3 Tujuan

Tujuan dari penelitian ini ialah:

1. Merancang kendali untuk mengatur kecepatan motor BLDC dengan menggunakan metode *trapezoid control* pada FPGA.
2. Merancang modul pengukur kecepatan motor BLDC dengan sensor hall efek pada FPGA.

3. Mengetahui karekteristik kecepatan motor BLDC terhadapt referensi kecepatan (pontensiometer) dengan metode *trapezoid control*?

1.4 Batasan Masalah

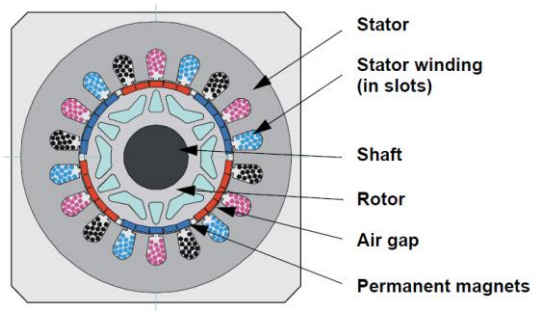
Dalam melaksanakan penelitian ini, permasalahan yang akan dibahas dibatasi dengan ketentuan berikut:

1. Algoritma trapezoid control, dan modul speed computation didesain dengan menggunakan *Verilog HDL (hardware description language)* beserta dengan modul-modul pendukung lainnya dan diimplementasikan pada FPGA.
2. Implementasi berfokus pada karakteristik kecepatan motor BLDC terhadap nilai ADC, dan Arus terhadap kecepatan putar motor BLDC.

BAB II
DAFTAR PUSTAKA

2.1 BLDC Motor

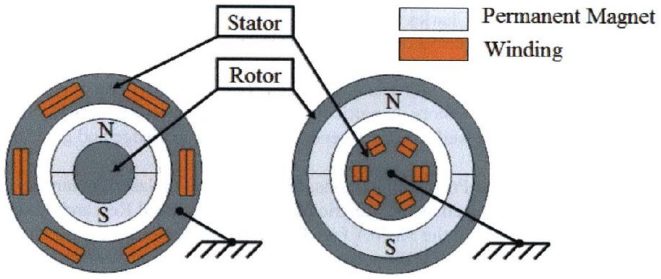
Motor *Brushless Direct Current* (BLDC) merupakan jenis *permanent magnet motor*. Motor BLDC memiliki *back electromotive force* (*back emf*) yang berbentuk trapezoid berbeda dengan motor AC sinkron yang memiliki *back EMF* berbentuk sinusoidal [7], tetapi kedua motor ini dapat dikendalikan dengan metode 6 step komutasi. Adapun perbedaan antara motor DC sikat dengan motor DC tanpa sikat yaitu Motor DC tanpa sikat menggunakan komutator elektronik daripada komutator mekanik, dengan ini BLDC memiliki keuntungan yaitu biaya perawatan yang rendah dan noise (suara bising) yang dihasilkan lebih rendah. seperti disebutkan pada [8], motor BLDC lebih cocok digunakan pada kendaraan listrik karean ukuran dan efisiensinya dibandingkan dengan Motor DC dan Motor Induksi.



Gambar 1. *Cross section* Motor DC tanpa sikat

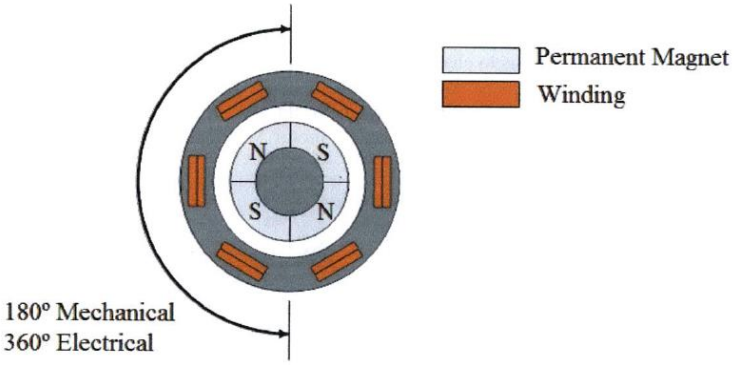
Motor DC tanpa sikat terdiri atas bagian yang diam yang disebut stator dan bagian yang bergerak disebut rotor. Ruang antara stator dan rotor disebut celah udara (*air gap*) Gambar 1, pada stator terdiri atas belitan yang terhubung dengan konfigurasi star ataupun konfigurasi delta sedangkan pada rotor terdapat magnet permanen. penempatan rotor pada motor DC tanpa sikat pada umumnya ada dua yaitu rotor di dalam (*Inrunner*) dan rotor di luar (*Outrunner*) ditunjukkan pada Gambar 2. Keuntungan motor BLDC rotor internal terletak pada inersia rotornya

yang rendah dan pembuangan panas lebih baik. Sebaliknya, untuk motor rotor eksternal, kumparan penghasil panas diisolasi dari lingkungannya oleh rumah rotor dan magnet. Motor rotor eksternal memiliki keunggulan untuk aplikasi yang diproduksi secara massal, karena dapat diproduksi dengan lebih murah [9]



Gambar 2. posisi rotor di dalam /*Inrunner* (kiri), dan posis rotor di luar/*outrunner* (kanan)

Fasa adalah kumpulan individu dari belitan dengan terminal tunggal yang dapat diakses dari luar motor. Kebanyakan motor brushless adalah tiga fase. Setiap loop individu dari kawat membentuk fasa lilitan disebut belokan (*turn*). Kutub merupakan kutub magnet permanen tunggal, baik utara atau selatan. Jumlah kutub minimum adalah dua, tetapi motor dapat memiliki jumlah kutub yang genap. Motor yang lebih besar cenderung memiliki kutub yang lebih banyak. Jumlah kutub tidak berhubungan langsung dengan jumlah slot, meskipun terdapat kombinasi umum antara slot dan jumlah kutub agar performa motor bekerja lebih baik. Pada motor dengan lebih dari dua kutub, penting untuk menentukan perbedaan antara kecepatan sudut mekanis dan kecepatan sudut elektrik. Kecepatan sudut mekanis adalah kecepatan fisik yang diukur dengan busur derajat atau tachometer. kecepatan sudut listrik mewakili posisi relatif dalam satu periode magnet, yang membentang di dua kutub. Perbedaan ini diilustrasikan dengan motor 4 kutub atau 2 pasang kutub pada Gambar 3.



Gambar 3. Perbedaan antara kecepatan sudut mekanik dan kecepatan sudut elektrik pada motor 4 kutub

persamaan sederhana untuk menghitung kecepatan sudut electric dan mekanik untuk motor 4 kutub yaitu:

$$\omega_e = p\omega_m$$

Dimana ω_e merupakan kecepatan sudut elektrik, p merupakan jumlah pasang kutub, dan ω_m merupakan kecepatan sudut elektrik.

$$\omega_e = 2\ x\ 180^o$$

Berarti dalam setegah putaran mekanik sama dengan 360^o dan pada 1 putaran mekanik penuh sama dengan 720^o. Jika hasil dari kecepatan sudut dikonversi ke dalam satuan frekuensi maka digunakan persamaan di bawah, hasil dari persamaan ini akan digunakan untuk menghitung kecepatan putaran motor dalam RPM.

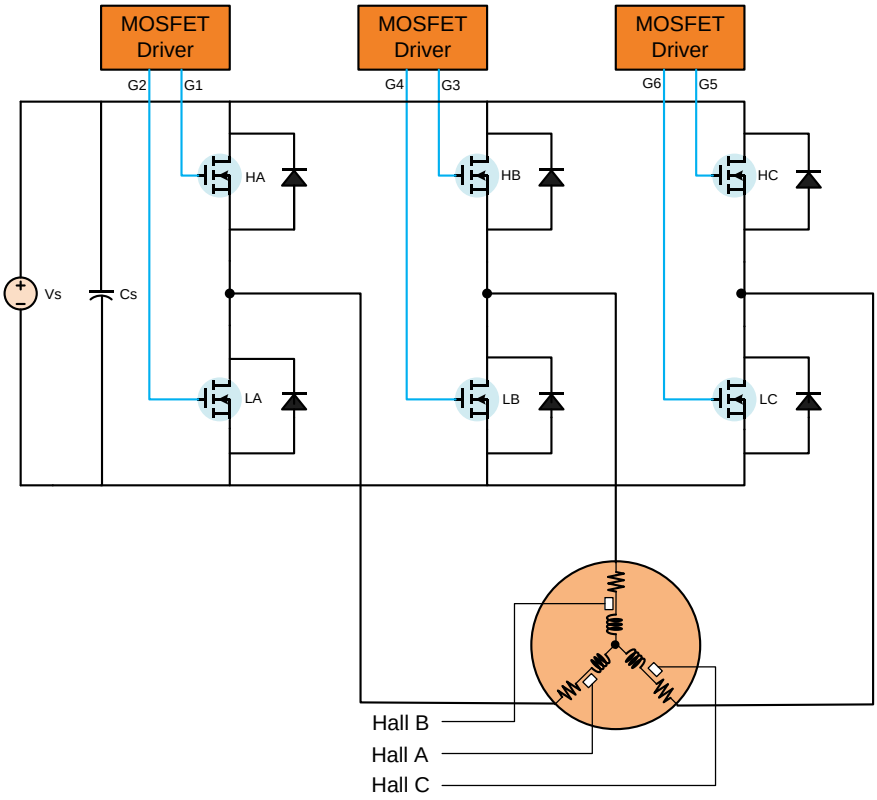
$$f = \frac{\omega}{2\pi}$$

Dimana, f merupakan frekuensi atau biasa juga disimbolkan dengan huruf v.

Telah dijelaskan sebelumnya bahwa komutasi motor BLDC dikendalikan secara elektronik. Untuk memutar motor BLDC, gulungan stator harus diberi energi secara berurutan. Penting untuk mengetahui posisi rotor untuk memahami belitan mana yang akan diberi *energize* mengikuti urutan

komutasi. Posisi rotor dideteksi menggunakan sensor hall efek yang tertanam ke dalam stator. Prinsip kerja dari sensor hall efek ialah akan mengeluarkan logika 1 (high) jika mendeteksi atau berada didekat medan magnet. Biasanya, tiga sensor hall efek dipindahkan (*displaced*) pada 120 derajat listrik. Setiap Sensor menghasilkan nilai 1 (*high*) untuk 180 derajat putaran listrik dan nilai 0 (*low*) untuk 180 derajat putaran listrik [10] [11].

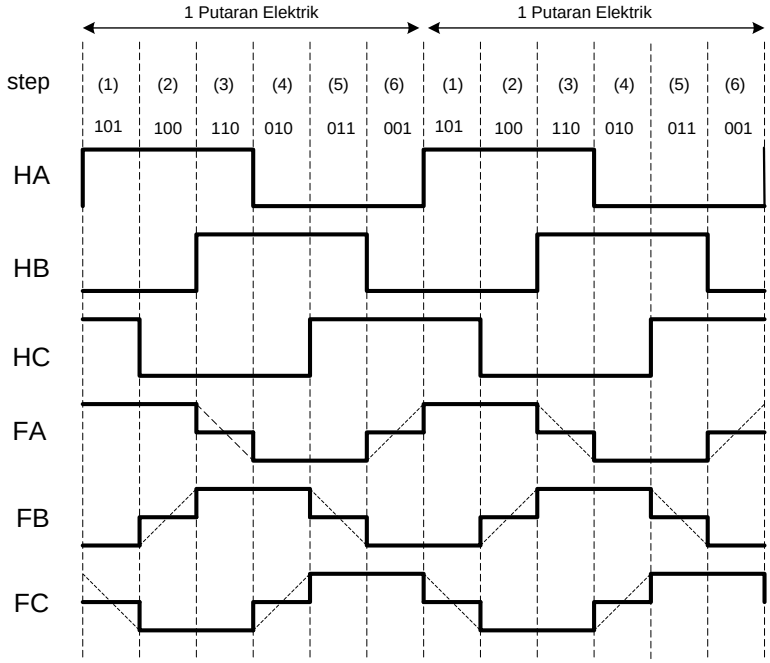
2.2 *Trapezoid Control* atau Blok Komutasi



Gambar 4. Inverter tiga fasa

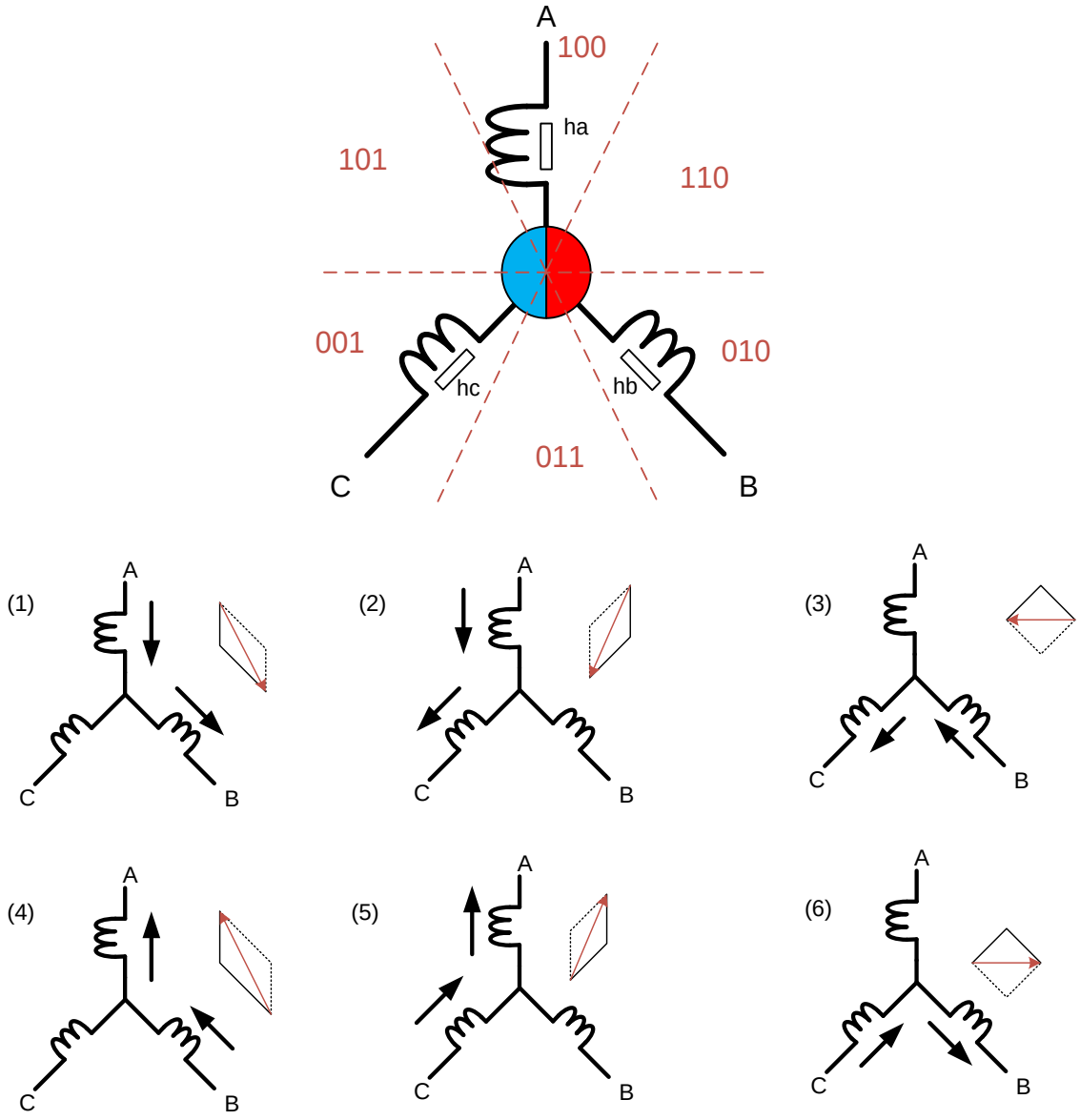
Juga dikenal sebagai *six-step control*, Teknik komutasi enam langkah digunakan untuk mengontrol Inverter tiga fasa (Gambar 4) / Driver motor BLDC dalam enam rangkaian pensaklaran. Dalam teknik ini hanya dua fasa motor BLDC yang diberi energy (*energized*) pada setiap urutan pensaklaran sedangkan fase lainnya tidak aktif. Pembalikan urutan switching mengubah arah putaran motor. Kecepatan motor berbanding

lurus dengan lebar sinyal sensor hall effect. Tegangan EMF balik, sinyal pergantian, arus fasa dan pola switching enam langkah dari motor BLDC tiga fasa ditunjukkan pada Gambar 5 [12]. *trapezoid control* adalah algoritma yang paling sederhana. Metode ini menghasilkan riak torsi tinggi, yang menyebabkan getaran, kebisingan, dan kinerja yang lebih buruk dibandingkan dengan algoritma lain.



Gambar 5. Bentuk gelombang kotak yang dihasilkan oleh *trapezoid control*

Dari komutasi pada Gambar 5 diatas, berikut gambar yang menunjukkan putaran motor berdasarkan pola komutasinya.



Gambar 6. posisi komutasi sensor hall efek pada motor dan rotasi motor berdasarkan komutasi hall sensor dan fasa pada Gambar 5

2.3 Metode *Switching* PWM

Kecepatan motor BLDC berbanding lurus dengan tegangan yang diberikan ke stator. Kecepatan di mana rotor dipaksa ke posisi berikutnya ditentukan oleh kekuatan gaya magnet, hal ini ditentukan oleh tegangan yang diberikan pada belitan stator. Dengan menggunakan PWM dengan frekuensi yang lebih tinggi dari pada frekuensi komutasi, besarnya

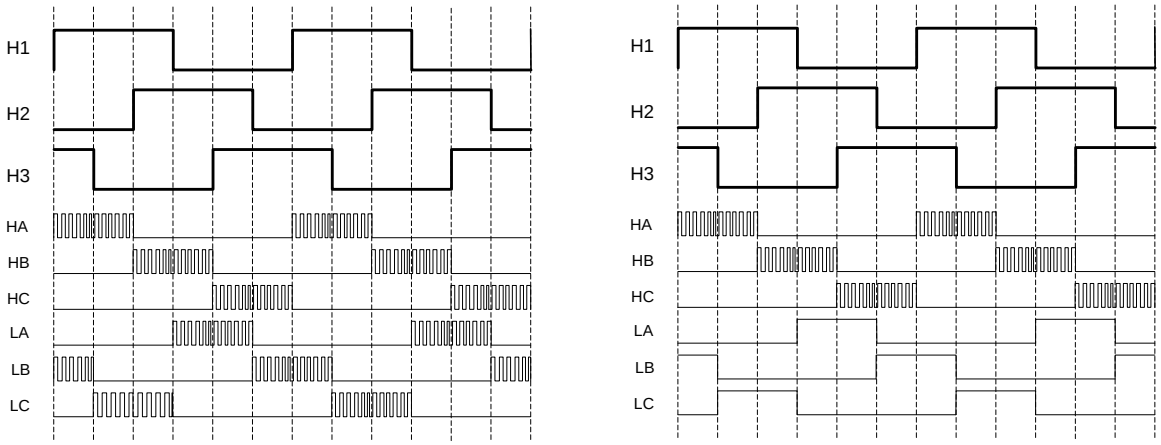
tegangan yang diberikan pada stator dapat dengan mudah dikontrol, sehingga kecepatan motor dapat dikontrol.

Kontroler PWM enam langkah tipikal menggunakan salah satu dari dua teknik [13]:

Unipolar (soft) PWM Switching merupakan Teknik ini mengacu pada fase motor yang dialihkan sedemikian rupa sehingga salah satu fase mengembalikan arus sementara modulasi PWM terjadi di fase lain, atau dengan kata lain hanya mosfet bagian atas saja yang di *drive* dengan sinyal pwm, untuk mosfet bagian bawah berupa switch on dan off saja.

Bipolar (hard) PWM Switching merupakan Teknik ini mengacu pada tegangan yang melewati dua fase sebagai dimodulasi dengan PWM, baik input maupun output arus sedang dimodulasi. Maksudnya ialah baik mosfet bagian atas maupun bagian bawah di *drive* dengan sinyal pwm

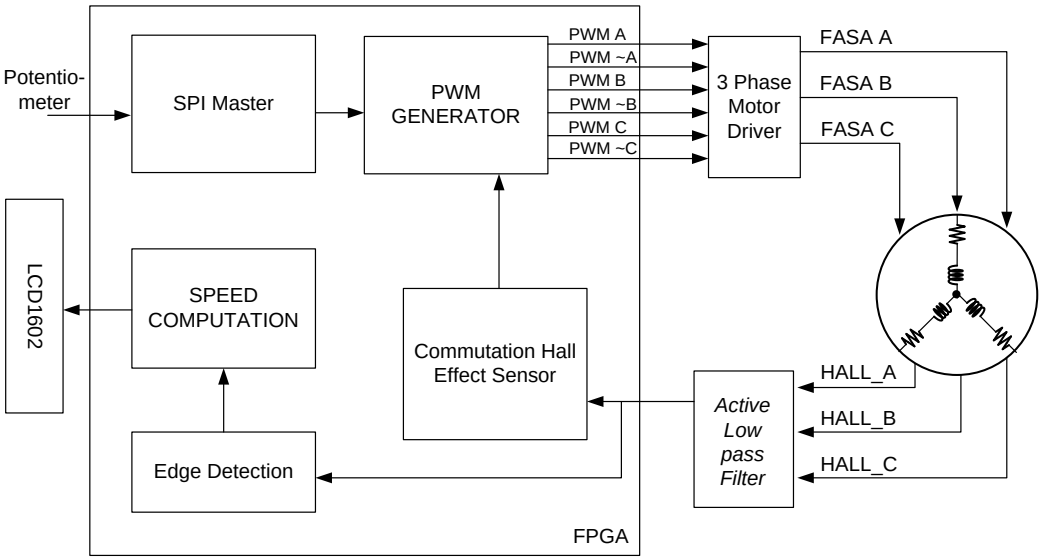
Metode *switching unipolar* dan *bipolar* memiliki keunggulan spesifik. Switching secara unipolar mengurangi kebisingan elektromagnetik dan riak DC karena ada lebih sedikit switching. Switching bipolar lebih cocok untuk pendekatan tanpa sensor di mana perlu untuk merasakan gaya elektromagnetik balik (BEMF). Pendekatan bipolar memiliki titik voltase nol pada siklus kerja 50%, oleh karena itu ada lebih banyak waktu untuk merasakan BEMF. Baik pendekatan unipolar dan bipolar dapat bersifat independen atau komplementer. Pendekatan unipolar dan bipolar mengacu pada hubungan dua fase. Pendekatan komplementer mengacu pada hubungan dua sinyal yang mengendalikan satu fase. Pendekatan independen berlaku untuk PWM hanya di satu sisi fase.



Gambar 7. Bagian kiri merupakan metode switching bipolar dan bagian kanan merupakan metode swithing unipolar

BAB III
PERANCANGAN SISTEM

Sistem kendali motor BLDC yang dirancang terbagi menjadi 2 bagian yaitu Sistem perangkat keras (*Hardware*) dan Perangkat lunak (*Software*). Bagian perangkat keras berupa Driver Tiga Fasa yang terdiri atas mosfet, bus kapasitor, driver mosfet, filter hall sensor, dan pembagi tegangan. Filter hall sensor dengan *active low pass filter* dibuat berdasarkan saran dari penelitian [14] untuk mengurangi noise dan meningkatkan performa. Sedangkan software berupa modul modul yang ditulis dalam Verilog HDL *(Hardware Description Language)* yang diimplementasikan pada Terasic DE0-NANO board yang di dukung dengan chip FPGA Intel/Altera cyclone-iv dengan seri EP4CE22F17C6. Secara abstrak sistem yang dirancang ditunjukkan pada Gambar 8. Berbeda dengan penelitian [14] yang menggunakan CPLD, pada penelitian ini menggunakan FPGA. Tidak hanya itu, beberapa modul yang belum ada pada penelitian [14] seperti modul speed computation, modul Edge Detection juga didesain. Selanjutnya, untuk modul komutasi di desain dengan menggunakan model mesin keadaan, berbeda dengan modul komutasi [14] yang hanya menggunakan *procedural assignment* yang outputnya hanya diperungaruhi oleh perubahan input.

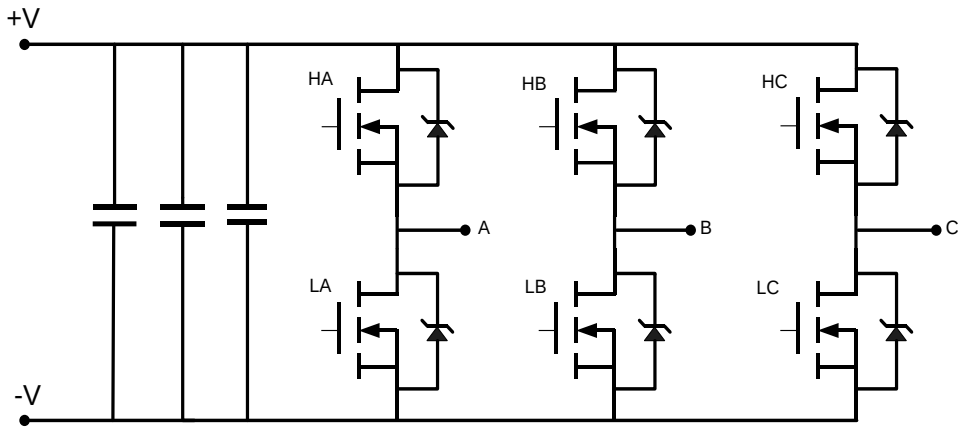


Gambar 8. Diagram block sistem pengendalian motor BLDC

3.1 Perancangan Driver Controller Motor BLDC

3.1.1 Gambaran Umum Driver 3 Fasa

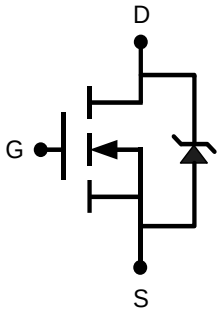
Inverter DC/AC 3-fasa merupakan perangkat elektronika daya yang mengubah sumber DC menjadi sumber listrik AC 3-fasa, frekuensi keluarannya dapat diubah-ubah, sehingga Inverter 3-fasa digunakan juga sebagai driver motor induksi tiga fasa. Berikut rangkaian inverter 3-fasa:



Gambar 9. Driver 3 Fasa

3.1.2 Mosfet

Mosfet (ditunjukkan pada Gambar 10) merupakan komponen elektronika daya pada umumnya digunakan untuk switching atau untuk mengendalikan tengangan tinggi yang mengalir dari Drain ke Source dengan mengatur tengangan rendah pada Gate dengan kata lain mengatur dutycycle sinyal yang masuk pada Gate Mosfet hal ini dapat diimplementasikan dengan PWM (*Pulse Switch Modulation*). Pada driver controller digunakan jenis Mosfet N-Channel dari Internation Rectifier dengan part number IRFB3607. Salah satu keutungan menggunakan mosfet ini sebagai driver motor BLDC berdasarkan datasheet [15] ialah *high speed power switching* menjadi alasan jenis mosfet ini digunakan selain itu arus yang dapat ditangani cukup tinggi sekitar 80 A pada temperatur normal, spesifikasi mosfet ditunjukkan pada Tabel 1.



D : Drain

G : Gate

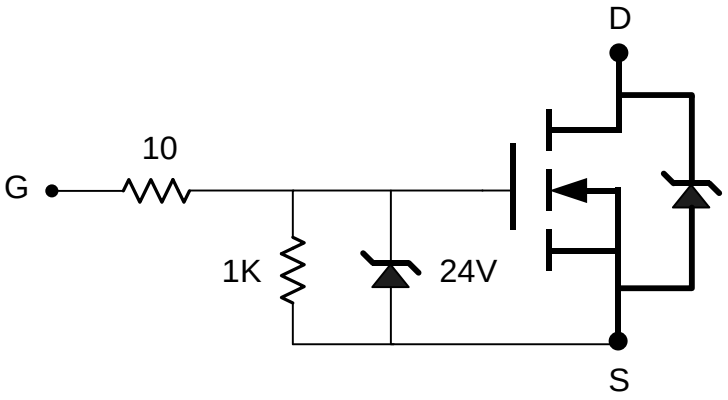
S : Source

Gambar 10. Simbol Mosfet N-Channel

Tabel 1. Spesifikasi Mosfet

Symbol	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, VGS @ 10V	80ⓘ	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	56ⓘ	
I_{DM}	Pulsed Drain Current ⓘ	310	
P_D @T_C = 25°C	Maximum Power Dissipation	140	W
	Linear Derating Factor	0.96	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V

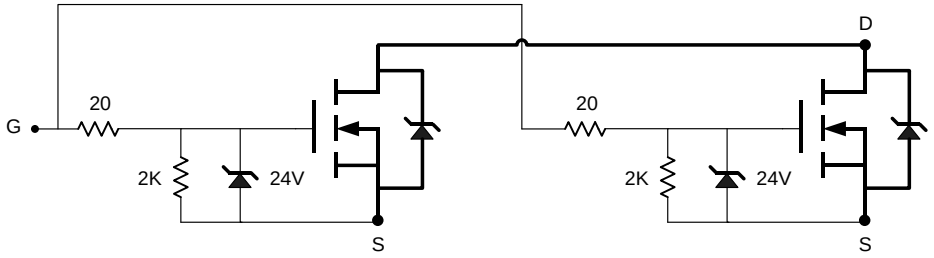
Rangkain mosfet yang digunakan pada driver controller ditunjukkan pada Gambar 11, resistor 10 ohm yang menghubungkan gate driver ke gate mosfet yang biasa disebut dengan "*gate resistor",* membatasi arus ketika gate driver men-switch gate on dan off. Resistor 1K ohm disebut dengan "*pull-down"* resistor, bertujuan untuk mengantisipasi jika gate driver gagal dan menjadi impedansi tinggi, maka resistor 1K ohm menjadi jalur gate discharge, dan men-off kan Mosfet. Diode zener 24V merupakan diode khusus yang disebut dengan "*transient voltage suppressor"* atau TVS. Didesain untuk jebol (*break down*) ketika tegangan balik melebihi 24V, zener tersebut mempertahankan tegangan yang melewati terminal pada 24V, dan menyerap energy berlebih dan melepaskannya sebagai panas [16]. Jika gate driver bekerja secara normal pada 15V diode zener tidak dibutuhkan, tetapi jika driver rusak dan terjadi tetagan berlebih, TVS dapat melindungi gate Mosfet itu sendiri. Sebetulnya nilai TVS yang tepat ialah 17V Karena maksimum tegangan gate mosfet sekitar 20V tetapi karena keterbatan komponen maka digunakan diode zener 24V.



Gambar 11. Rangkaian mosfet

Selanjutnya, untuk meningkatkan arus yang dapat mengalir pada driver controller dapat dilakukan dengan memparelelkan 2 mosfet sehingga arus yang mengalir pada drain ke source dapat terbagi. Pada

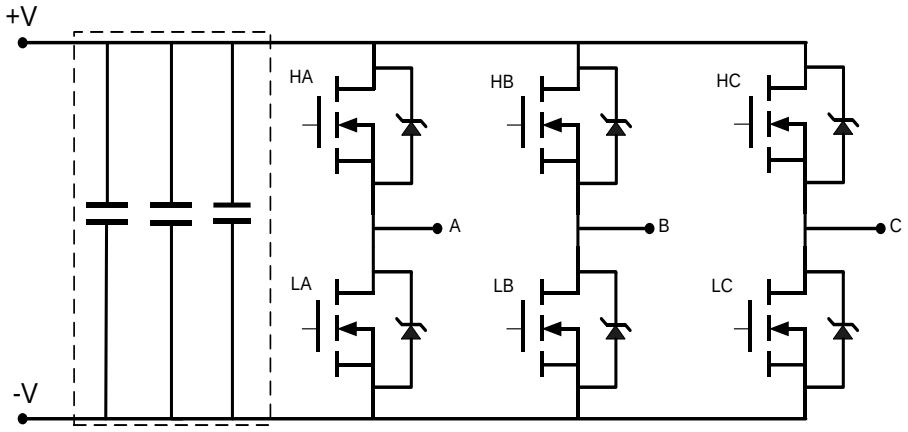
Gambar 12. Paralel Mosfet, setiap mosfet memiliki resistor gate dan pull-down resistor masing-masing.



Gambar 12. Paralel Mosfet

3.1.3 Bus Kapasitor

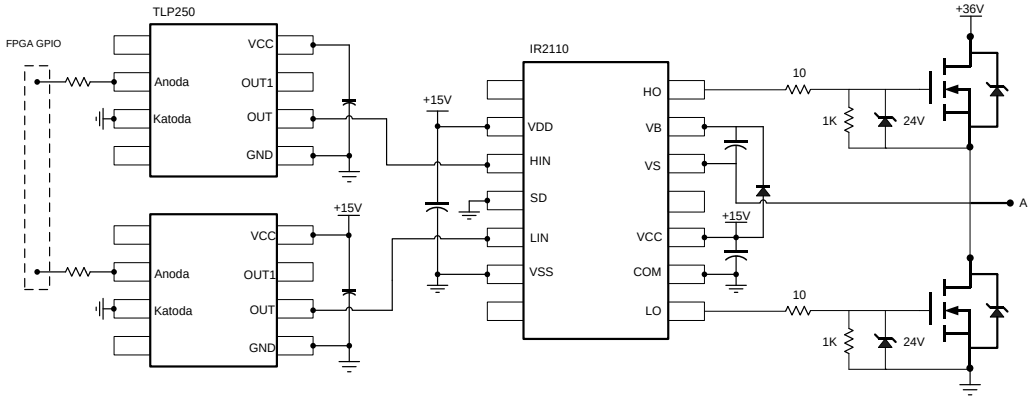
Bus kapasitor merupakan komponen passive yang umum digunakan untuk mengatasi riak dan arus transient yang dihasilkan oleh battery. Pada Gambar 13 terdapat tiga bus kapasitor yang diparalelkan, dalam desian PCB setiap inverter memiliki bus kapasitor yang ditempatkan sedekat mungkin dengan inverter terminal power. Nilai masing masing bus kapasitor yang digunakan ialah 450µf 63V.



Gambar 13. Rangkain bus kapasitor pada driver controller

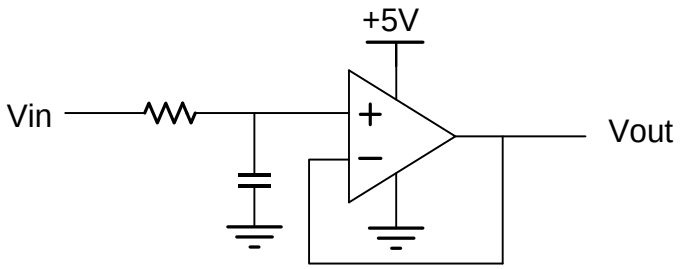
3.1.4 Driver Mosfet

Dalam design digunakan 2 jenis Driver Mosfet yaitu Optocoupler (TLP250) dan *high voltage* Driver Mosfet (IR2110). Optocoupler merupakan komponen yang digunakan untuk mengisolasi antara fpga board dengan inverter, TLP250 terdiri ata *infrared emmiting diode* dan *integrated photodector* untuk mentransfer sinyal dari dua komponen yang terisolasi dengan menggunakan cahaya tepatnya dari FPGA GPIO ke IR2110. Informasi yang lebih jelas terkait TLP250 dapat dilihat pada datasheet [17]. IR2110 [18] merupakan komponen tambahan yang digunakan sebagai gate driver mosfet, komponen ini langsung terhubung dengan rangkaian gate mosfet seperti yang telah ditujukkan pada Gambar 11. detail rangkaian Mosfet driver ditunjukkan pada Gambar 14 pada desain pcb digunakan 6 buah TLP250 dan 3 buah IR2110 tepatnya terdapat 3 rangkain yang sama dengan Gambar 14.



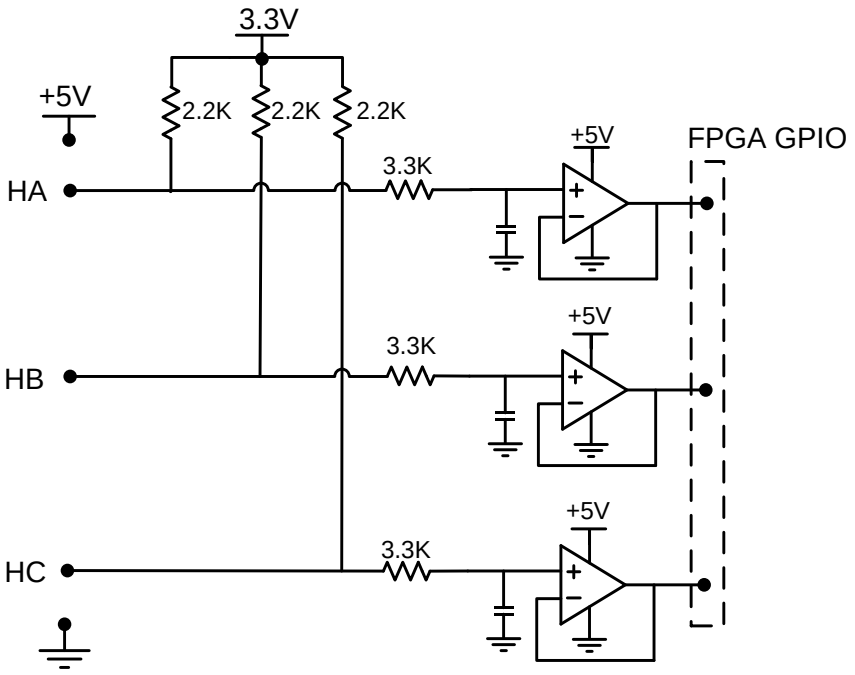
Gambar 14. Rangkaian Mosfet driver dan half bridge

3.1.5 Filter Sensor Hall Effect



Gambar 15. Rangkaian *active low pass filter*

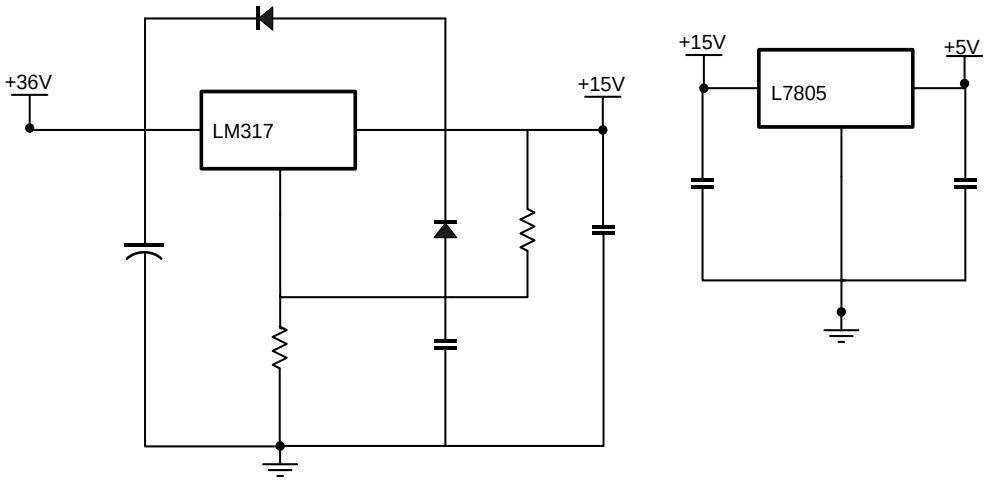
Filter analog yang digunakan dalam desain ini adalah *active low pass filter* Gambar 15. adapun rangkian filter sensor hall effect yang diimplementasikan secara hardware ditunjukkan pada Gambar 16, rangkian filter ini menggunakan rangkaian pull-up resistor dengan resistor bernilai 2.2K ohm yang terhubung dengan sumber 3.3V dan *active low fass filter* dengan menggunakan resistor 3.3K ohm yang parallel dengan capacitor yang terhubung dengan pin non-iverting op-amp. Pull-up resistor sebetulnya sudah cukup untuk memperoleh data digital dari sensor hall effect, tetapi dalam implementasi terdapat riak dan *noise* khususnya ketika motor berputar. Dengan *passive low pass filter* dengan kombinasi komponen R dan C riak tersebut dapat dikurangi.



Gambar 16. Filter Sensor Hall Effect

3.1.6 Pembagi Tegangan

Kontroller yang bagus ialah kontroller hanya disuplay dari satu sumber saja biasanya battery, baik sumber tegangan untuk komponen arus besar maupun arus kecil seperti sinyal control. Untuk memenuhi tegangan sinyal control dana komponen seperti mosfet driver biasanya digunakan IC pembagi tegangan. Dalam perancangan driver ini digunakan 2 IC pembagi tegangan yaitu LM317 yang mengatur tegangan dari 36V ke 15V untuk menyuplai mosfet driver (TLP250 dan IR2110) dan L7805 mengatur tegangan dari 15V ke 5V untuk menyuplai FPGA. Rangkain detail pembagi tegangan yang digunakan ditunjukkan pada gambar Gambar 17.

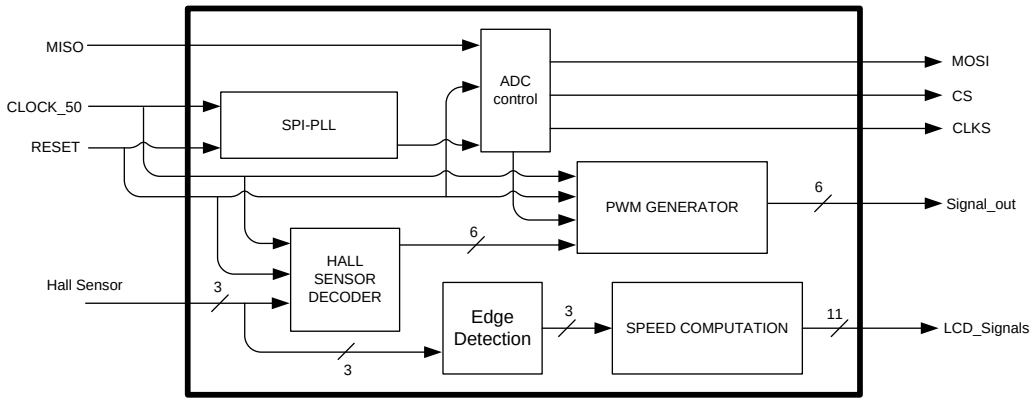


Gambar 17. Rangkaian pembagi tegangan

3.2 Perancangan Algoritma Controller Motor BLDC

3.2.1 Deskripsi sistem

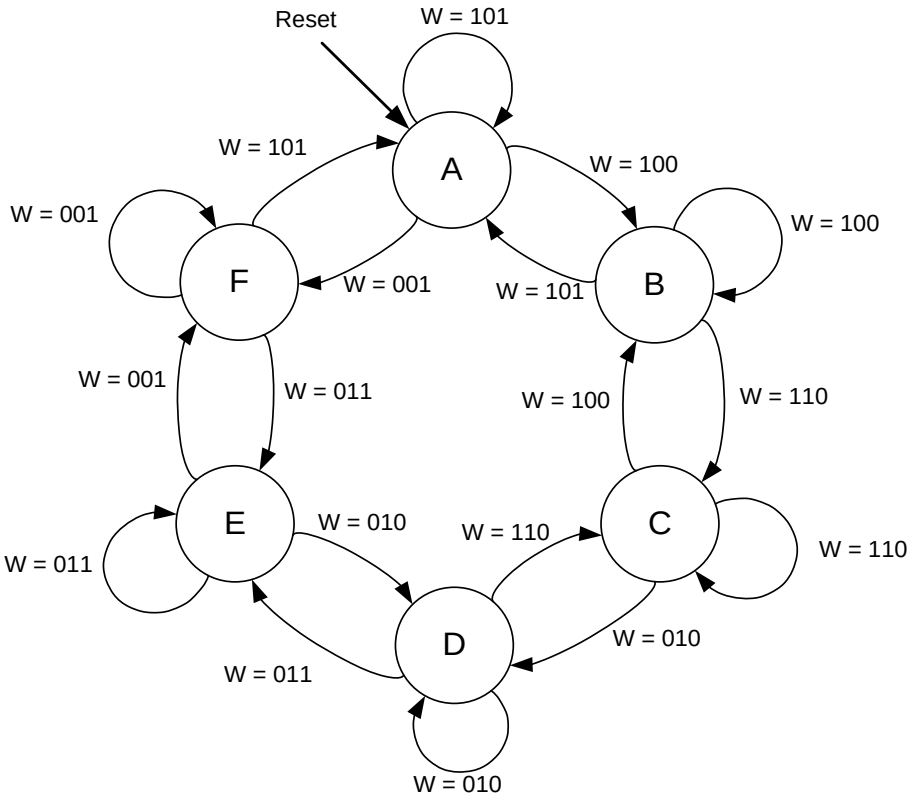
Algoritma pengendali motor BLDC ditulis dalam Verilog HDL (*Hardware Description Language*) yang akan diimplementasikan pada FPGA board. Code Verilog yang dibuat mengacu pada standar IEEE Std 1364^{TM}-2005. Secara umum modul yang dirancang (Gambar 18) terdiri dari ADC control, PWM Generator, dan Dekoder sensor hall effect. Semua modul – modul Verilog akan dirancang pada *software* QuartusII Lite versi 18.1. Untuk penjelasan secara lebih spesifik akan dijelaskan pada sub-bab ini.



Gambar 18. Diagram block modul-mudul algoritma kendali motor BLDC

3.2.2 Komutasi Menggunakan Sensor Hall Effect

6-step komutasi di desain dengan menggunakan algoritma mesin keadaan yang ditunjukkan pada Gambar 19. struktur code Verilog mesin keadaan terbagi menjadi tiga yaitu program mendefinisikan keadaan berikutnya, mendefinisikan output, dan mendefiniskan blok sekuensial.



Gambar 19. Diagram mesin keadaan komutasi 6 langkah

Hasil dari mesin keadaan ini berupa 6 sinyal yang digunakan untuk mengaktifkan mosfet, dengan 2 sinyal bernilai 1 secara bersamaan yaitu 1 dari mosfet atas dan 1 lagi dari mosfet bagian bawah. Konfigurasi ini berlangsung secara terus menerus berdasarkan tabel komputasi menggunakan sensor hall effect yang ditunjukkan pada Tabel 2 dan Tabel 3 [19].

Tabel 2. Komutasi searah jarum jam

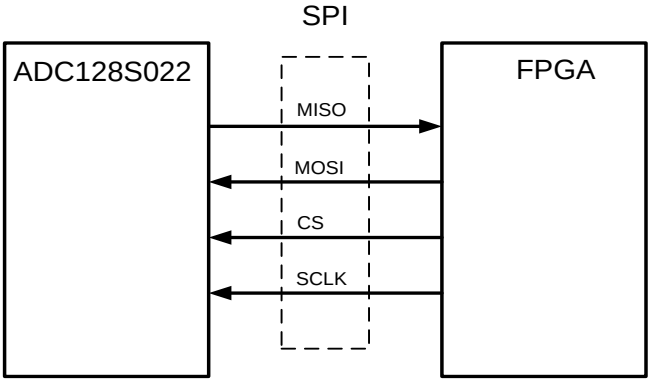
Step	Sensor Hall Effect			Mosfet						Fasa		
	H1	**H2**	**H3**	**HA**	**HB**	**HC**	**LA**	**LB**	**LC**	**F1**	**F2**	**F3**
1	1	0	1	1	0	0	0	1	0	DC+	DC-	Off
2	1	0	0	1	0	0	0	0	1	DC+	Off	DC-
3	1	1	0	0	1	0	0	0	1	Off	DC+	DC-
4	0	1	0	0	1	0	1	0	0	DC-	DC+	Off
5	0	1	1	0	0	1	1	0	0	DC-	Off	DC+
6	0	0	1	0	0	1	0	1	0	Off	DC-	DC+

Tabel 3. Komutasi berlawanan arah jarum jam

Step	Sensor Hall Effect			Mosfet						Fasa		
	H1	**H2**	**H3**	**HA**	**HB**	**HC**	**LA**	**LB**	**LC**	**F1**	**F2**	**F3**
6	0	0	1	0	1	0	0	0	1	Off	DC+	DC-
5	0	1	1	1	0	0	0	0	1	DC+	Off	DC-
4	0	1	0	1	0	0	0	1	0	DC+	DC-	Off
3	1	1	0	0	0	1	0	1	0	Off	DC-	DC+
2	1	0	0	0	0	1	1	0	0	DC-	Off	DC+
1	1	0	1	0	1	0	1	0	0	DC-	DC+	Off

Jika dibandingkan Tabel 2 dengan Tabel 3 mosfet yang aktif dapat diperhatikan bahwa mosfet yang aktif untuk bagian atas merupakan negasi dari mosfet bagian bawah begitu dengan sebaliknya. Oleh karena itu, program dapat dibuat lebih sederhana dengan memanipulasi output dari mesin keadaan.

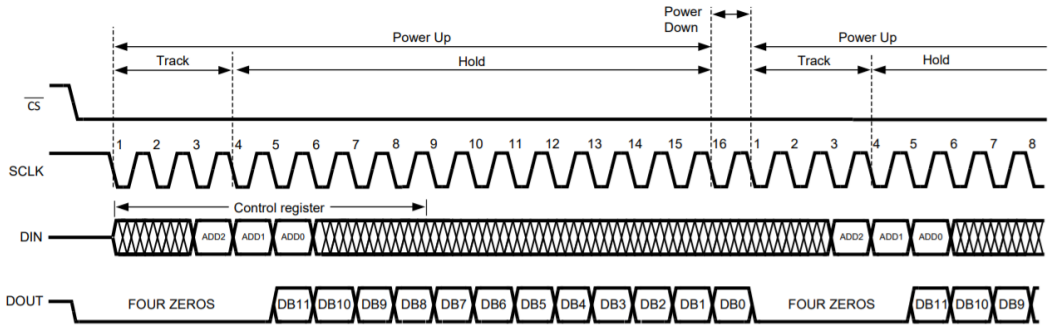
3.2.3 *Serial Peripheral Interface (SPI)*



Gambar 20. Interface komunikasi SPI

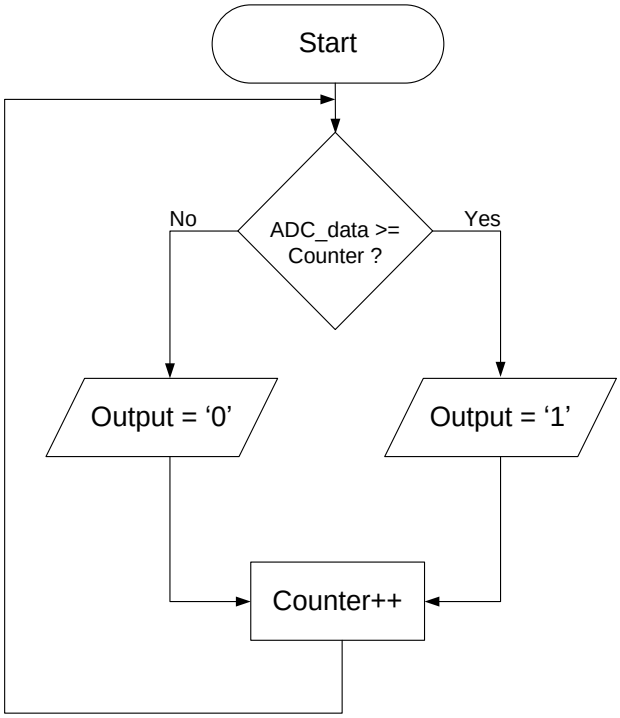
SPI adalah protokol serial standar industri yang biasa digunakan dalam *embedded system* untuk menghubungkan mikroprosesor ke berbagai sensor off-chip, konversi, memori, dan perangkat kontrol. Untuk membangun protokol komunikasi secara serial antara komponen master (chip FPGA) dengan komponen *slave* (chip ADC) perlu dibuat *interface* yang memungkinkan kedua komponen tersebut bertukar data secara serial. Dengan mengikuti standar interface dari SPI (Gambar 20) modul master yang didesain setidaknya memiliki input berupa MISO (*master in slave out*) yaitu berupa data ADC 1 bit yang dikirim secara serial sebagai data, kemudian output terdiri dari MOSI (*master out slave in*) berupa data alamat yang dikirim dari fpga ke chip ADC untuk memilih channel ADC yang digunakan untuk membaca data analog, CS berupa signal 1 bit yang dikirim dari master komponen yang menandakan bahwa proses inisiasi komunikasi data serial akan dimulai, jika signal CS berubah menjadi 0 menandakan bahwa komunikasi telah dimulai, dan kemudian CS berubah menjadi 1 menandakan bahwa komunikasi data sebanyak n bit telah selesai dan bersiap untuk inisiasi data selanjutnya. Output yang terakhir CLKS (clock slave) merupakan clock yang digunakan ADC untuk menyampel data agar proses komunikasi dapat sinkron. Untuk CLKS (clock slave) digunakan library PLL yang telah tersedia pada Quartus 18.1. Chip ADC yang digunakan tertanam pada

Board FPGA DE0-nano dari Terasic menggunakan ADC128S022, diagram komunikasi serial ditunjukkan pada Gambar 21.



Gambar 21. Diagram waktu pada komunikasi serial dengan ADC128S022

3.2.4 PWM Generator dan Switching PWM



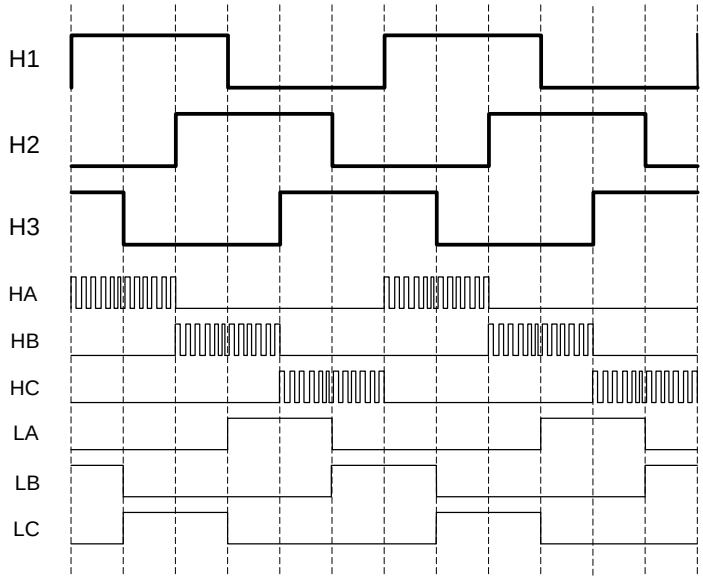
Gambar 22. *Flowchart* PWM Generator

Dalam FPGA PWM didesain secara digital dengan menggunakan counter dan comporator, dengan input nilai data dari ADC sebanyak n bit. Pertama counter menghitung dari nilai 0 hingga 2^n - 1 dengan referensi clock awal 25MHz, kemudian hasil counter setiap clock sebagai input dari comparator. Selanjutnya, komparator membandingkan nilai dari counter dan nilai ADC dengan clock yang sama dengan counter, jika nilai ADC

lebih besar daripada nilai counter maka comparator akan bernilai 1 jika sebaliknya akan bernilai 0. Untuk frekuensi pwm yang didesain di uraikan pada persamaan dibawah

$$f(pwm) = Base\_Clock/2^n$$

Dimana n = jumlah bit data ADC. Yang perlu diperhatikan dalam mendesain PWM ialah frekuensi sinyal PWM setidaknya 10 kali lebih cepat dari maksimum frekuensi motor (electrical) [20]. dan terakhir switching PWM menggunakan metode *Unipolar Independent PWM Switching* ditunjukkan pada Gambar 23.

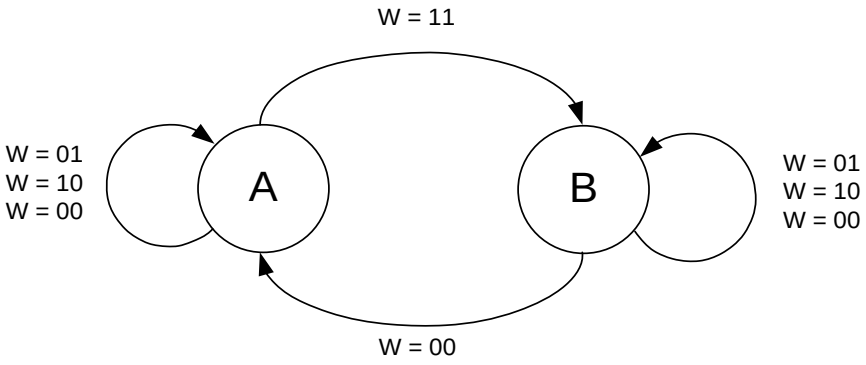


Gambar 23. *Unipolar Independent PWM Switching*

3.2.5 Filter Sensor Hall Effect

Filter sensor hall effect secara digital diimplementasikan dalam code Verilog, filter ini didesain dengan menggunakan mesin keadaan dan register geser. Register geser digunakan untuk menyampel data sensor hall effect hingga n bit data dengan clock sekitar 10MHz, selanjutnya Mesin keadaan digunakan untuk mendeteksi transisi ‘0’ ke ‘1’ dan begipula sebaliknya, diagram mesin keadaan ditunjukkan pada

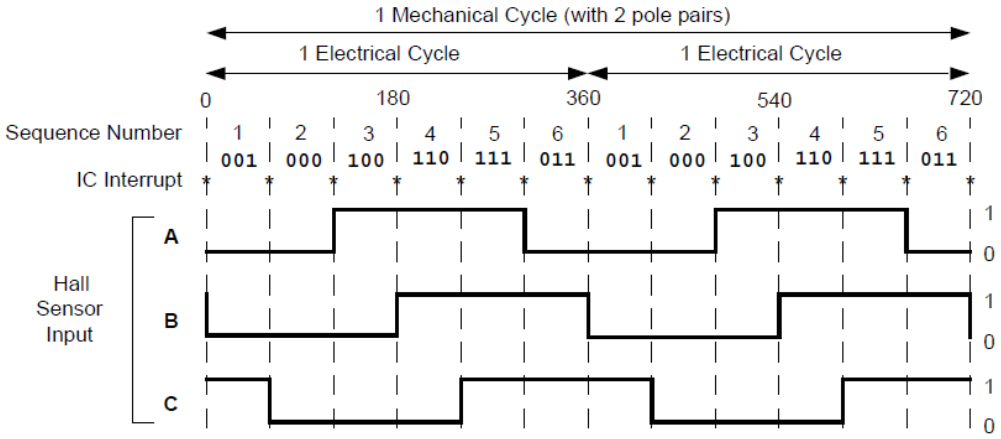
gambar B5.1. Input pada gambar dibawah merupakan hasil dari fungsi logika or dan and dari n-bit keluaran dari register, MSB merupakan hasil logika and dari n-bit data register geser, LSB merupakan hasil logikan or dari n-bit data register geser.



Gambar 24. Diagram mesin keadaan transisi ‘0’ ke ‘1’ dan ‘1’ ke ‘0’

3.2.6 Pengukur Kecepatan

Pengukur kecepatan didesain dengan menggunakan input dari hall sensor pada Gambar 25 menggambarkan bahwa 6 pola komutasi dari 3 hall sensor menghasilkan 1 putaran secara elektrik, dan dua putaran elektrik menghasilkan 1 putaran mekanik untuk motor dengan 2 pasang kutub [21]. Karena motor yang digunakan berdasarkan spesifikasi, yaitu memiliki 30 kutub atau 15 pasang kutub maka untuk memperoleh satu putaran mekanik penuh membutuhkan 15 kali putaran elektrik atau 15x6 pola komutasi.



Gambar 25. Pola hall sensor

Dalam mendesain modul penghitung kecepatan motor bldc digunakan counter, flip flop D, dan lookup table. Counter digunakan untuk menghitung putaran eletrik dari motor berdasarkan input hall sensor, flip flop D digunakan untuk menyimpan nilai kecepatan sebelumnya selama kurang lebih 1 detik, dan lookup table digunakan untuk menyimpan hasil perhitungan dari persamaan 6.1

$$RPM = \frac{60}{pairs\ of\ poles}\ x\ f$$

dimana f merupakan banyaknya putaran elektrik dalam satu detik(Hz). Lookup table menyimpan data rpm untuk setiap nilai f, selain itu untuk mendapatkan hasil perhitungan kecepatan yang lebih presisi digunakan 3 modul pengukur kecepatan untuk masing masing sensor hall effect kemudian dijumlahkan dan dihitung rata-ratanya dengan menggunakan modul divider. Adapun sensor hall effect yang digunakan telah difilter oleh modul filter sensor hall effect agar pembacaan sensor hall lebih tepat sehingga perhitungan lebih presisi.

3.3 Skenario Pengujian

Pengujian kendali motor BLDC yang dirancang akan dilakukan dengan 2 tahap scenario pembebanan:

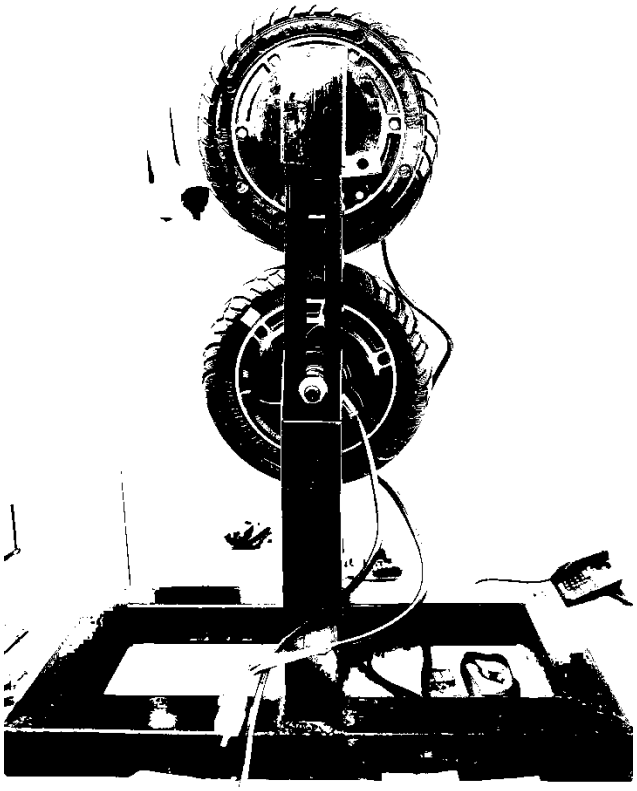
1. **Pengujian kendali motor BLDC tanpa beban**

   Pengujian tanpa beban aka dilakukan dengan mengobservasi kecepatan putar motor BLDC dengan mengubah-ubah nilia potensiometer, tegangan BEMF motor BLDC Ketika berputar, dan arus terhadap perubahan kecepatan motor BLDC.

2. **Pengujian kendali motor BLDC dengan beban**

   Karakteristik pengujian kendali motor BLDC dengan beban meliputi kecepatan putar motor BLDC dengan mengubah-ubah potensiometer, arus terhadap perubahan kecepatan motor BLDC dengan model

pembebanan ditunjukkan pada Gambar 26 diharapkan motor dapat menunjukkan perbedaan antara tanpa beban dan berbeban.

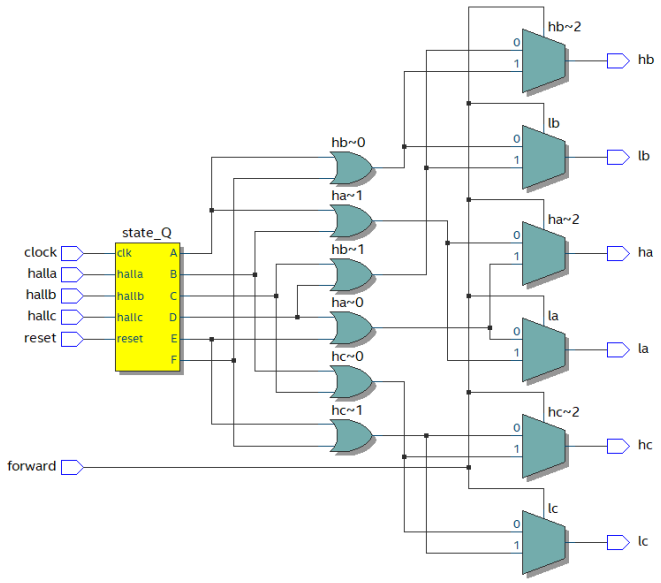


Gambar 26. Pembebanan

BAB IV
HASIL SIMULASI ALGORITMA KENDALI MOTOR BLDC

4.1 Modul Dekoder Komutasi

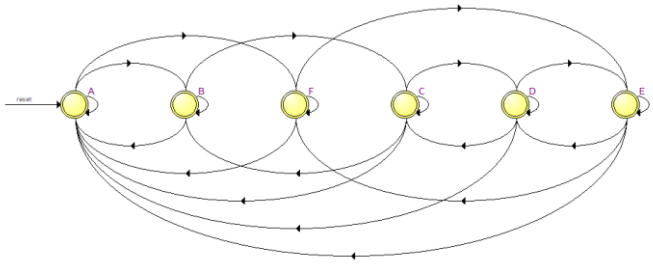
Modul komutasi merupakan modul utama dalam pengendali motor BLDC, modul ini mengatur putaran motor dengan menggunakan sensor hall effect sebagai input referensinya. Rangkaian logika modul dekoder komutasi ditunjukkan pada Gambar 27, modul ini terdiri dari sebuah *state machine*/mesin keadaan yang merupakan rangkaian sekuensial dimana inputnya bergantung pada input dan keadaan sebelumnya, kemudian gerbang logika OR, dan multiplexer merupakan rangkaian kombinatorial yang diterjemahkan oleh compiler Quartus Prime dari kode Verilog 'if-else'.



Gambar 27. *Netlist viewer* modul komutasi

Pengendali utama dalam modul komutasi ialah mesin sekuensial/*state machine*, tidak hanya dalam modul ini, tetapi dapat dikatakan merupakan otak dari pengendali Motor BLDC yang dirancang. State mechine ini mengatur output kapan bernilai logika '1' atau bernilai logika '0'

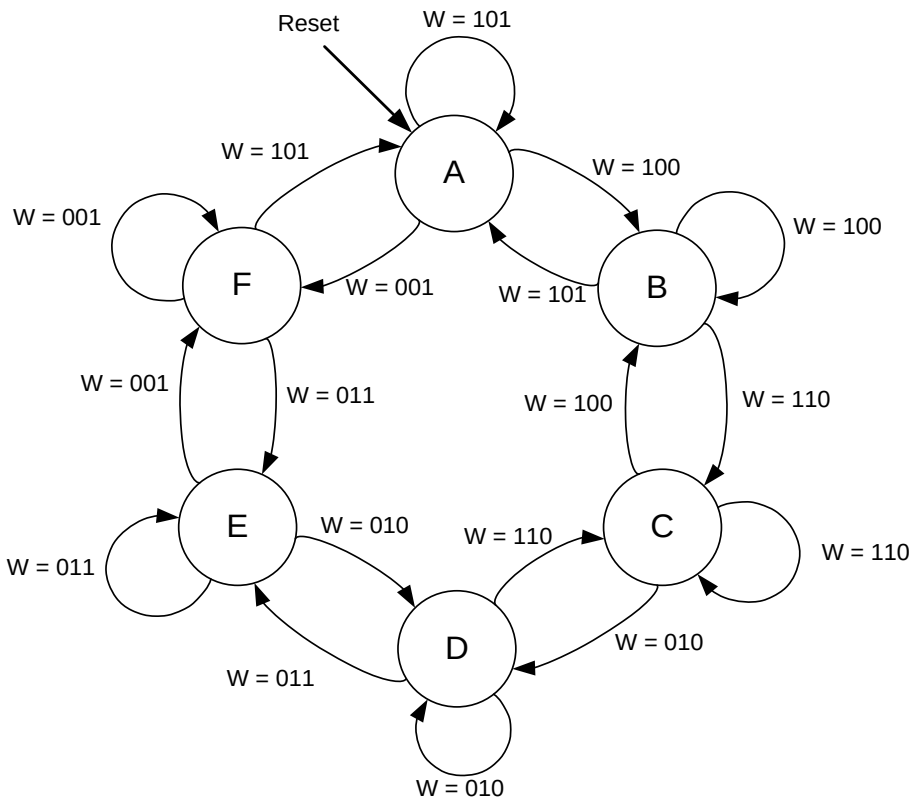
berdasarkan keadaan sekarang. Diagram *state mechine* ditunjukkan pada Gambar 28, terdiri dari 6 keadaan yaitu keadaan A, B, C, D, E, dan F.



Gambar 28. Diagram mesin keadaan modul komutasi terjemahan *compiler* Quartus Prime 18.1

Masing-masing keadaan direpresentasikan dalam tiga bit parameter secara berurutan yaitu “000”, “001”, “010”, “011”, “100”, “101”. *State* inisial yaitu *state A*, jika keadaan selain A dan ketika Reset bernilai logika ‘1’ maka secara *default state* berikutnya ialah *state A*, perpindahan ke keadaan berikutnya ditentukan oleh nilai sensor hall effect pada Gambar 29 di simbolkan dengan huruf W , Urutan keadaan untuk putaran CW (*clockwise*) atau setara jarum jam ialah A-B-C-D-E-F, sedangkan keadaan untuk putaran CCW (*couterclockwise*) atau berlawaman arah jarum jam ialah F-E-D-C-B-A arah putaran ini diatur oleh sinyal “forward”. Nilai sensor hall efek dari MSB ke LSB merupakan gabungan dari sensor hall efek 1, 2, dan 3 secara berurutan.

Output mesin keadaan di tunjukkan pada gambar 28 yaitu 6 sinyal komutasi berupa sinyal **ha**, **hb**, **hc**, **la**, **lb**, **lc**. **ha** dan **lb** merepresentasikan sinyal untuk mosfet fasa A bagian atas dan bagian bawah secara berurutan, begitupun **hb**, dan **lb** untuk mosfet fasa B, dan **hc** dan **lc** untuk mosfet fasa C. ketika sinyal “forward” bernilai nol menandakan bahwa motor akan berputar searah jarum jam, dengan output sinyal yaitu sinyal **ha** bernilai satu pada *state A* dan *state B,* sinyal **hb** bernilai satu pada *state C* dan *state D,* **hc** bernilai satu pada *state E* dan *state F,* **la** bernilai satu pada *state D* dan *state* E, **lb** bernilai satu pada *state A* dan *state F,* dan terakhir **lc** bernilai satu pada *state B* dan *state C.*

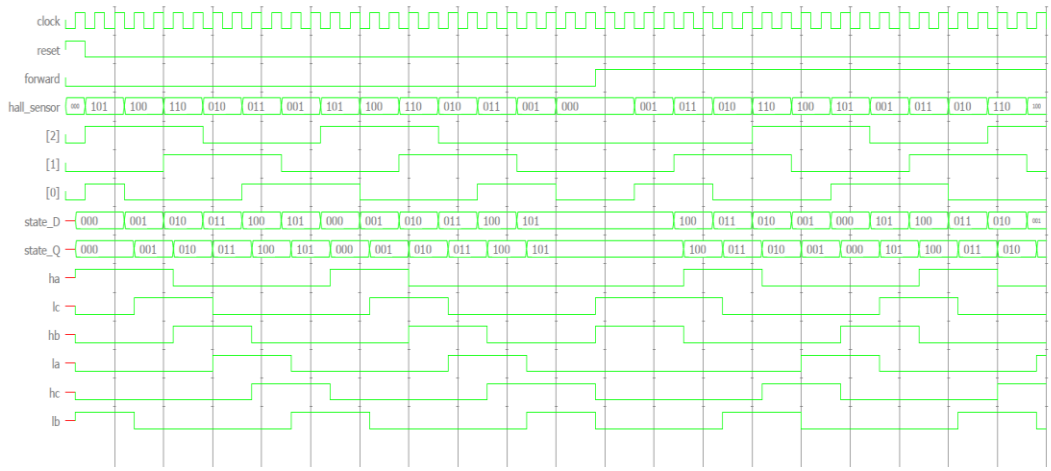


Gambar 29. Urutan keadaan berikutnya berdasarkan sensor hall efek

Selanjutnya untuk output ketika sinyal “forward” bernilai satu yang menandakan bahwa motor akan berputar berlawanan arah jarum jam, sinyal **ha** bernilai satu pada *state D* dan *state E*, **hb** bernilai satu pada *state A* dan *state F*, **hc** bernilai satu pada *state B* dan *state C*, **la** bernilai satu pada *state A* dan *state B*, **lb** bernilai satu pada *state C* dan *state D*, dan **lc** bernilai satu pada *state E* dan *state F*. pola output ketika motor berputar berlawanan arah jarum jam merupakan pertukaran antara output high (ha, hb, hc) dengan output low (la, lb, lc) dari pola putaran searah jarum jam.

Keuntungan menggunakan mesin keadaan yaitu output dari kontroller dapat dikendalikan dengan baik walaupun pembacaan dari hall sensor tidak tepat, sehingga tidak ada *overshoot* pada saat *starting* Motor BLDC. Selain itu, switching mosfet bagian atas dan bagian bawah dapat diatur

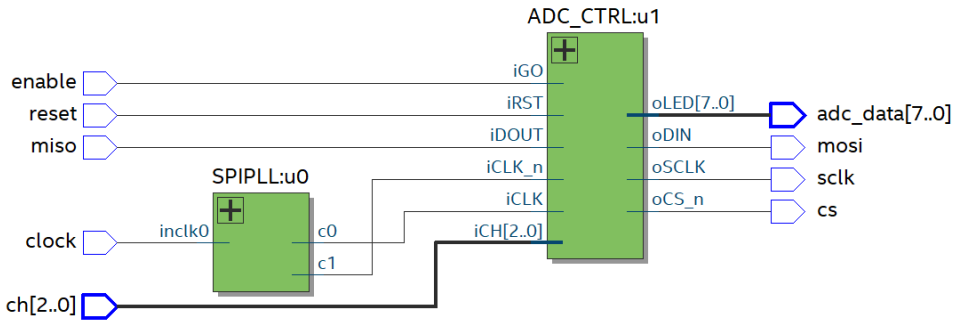
sehingga menghindari terjadinya *short circuit* yang dapat merusak driver/inverter tiga fasa. *Short circuit* pernah terjadi ketika tidak menggunakan *state machine* dikarenakan pembacaan hall sensor kadang tidak tepat, kemudian tanpa *state mahine* sering kali menimbulkan overshoot.



Gambar 30. *waveforms* hasil simulasi modul komutasi

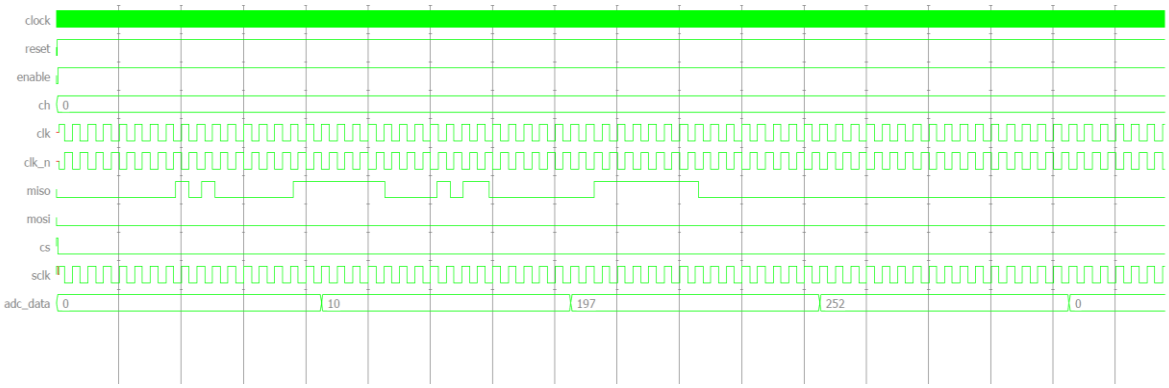
4.2 Modul SPI Master

SPI merupakan protocol komunikasi secara serial, maka data yang dikirim dari ADC (chip slave) ke FPGA (chip master) melalui interface miso hanya berupa data dengan lebar 1 bit, begitupun dengan interface mosi mengirimkan data dari FPGA (chip master) ke ADC (chip slave) berupa data 1 bit. Selain itu terdapat interface CS (chip select) dan SCLK (serial clock). PLL digunakan untuk membagi clock dari 50 MHz menjadi 2 MHz, c0 dan c1. Fasa clock c1 digeser sebesar 180^{o}, sebagai clock untuk mengirim data untuk memilih channel ADC yang akan digunakan melalui interface mosi, dan c0 digunakan sebagai SCLK. Gambar 31 menunjukkan koneksi antara SPIPLL modul dengan modul ADC control.



Gambar 31. *Netlist viewer* modul SPI Master

Adapun CS digunakan untuk menginisiasi komunikasi dan membigkai transfer data serial, pada Gambar 32 sinyal CS bernilai logika ‘0’ jika enable bernilai logika ‘1’. Untuk memperoleh data adc sebanyak 12 bit dibutuhkan sebanyak 16 siklus dari clock SCKL atau c0.

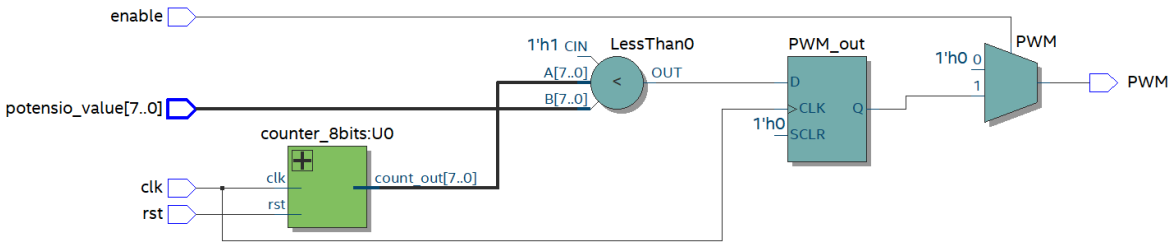


Gambar 32. *Waveforms* hasil simulasi modul SPI Master

4.3 Modul PWM Generator

Hasil dari pembacaan ADC berupa data digital dari potensio meter digunakan untuk mengatur duty cycles dari pwm yang dirancang, 12 bit data ADC hanya digunakan 8 bit data, Gambar 33 menunjukkan sebuah modul counter_8bits. Modul counter_8 bit ini merupakan pencacah yang menghitung dari 0 hingga 255. Kemudian, output dari pencacah ini kemudian dibandingkan dengan data dari ADC, jika data ADC lebih dari data pencacah pada siklus clock tersebut maka PWM akan benilai logika

‘1’. Data ADC dan data counter akan terus dibandingkan setiap siklus clock.



Gambar 33. *Netlish viewer* modul PWM generator

Clock yang digunakan modul pwm generator ialah 25 Mhz, maka frekuensi dan periode sinyal pwm yang dihasilkan ialah

Persamaan menghitung frekuensi:

$$f(pwm) = \frac{25\ Mhz}{2^8}$$

$$f(pwm) \approx 97\ Khz$$

Persamaan menghitung periode:

$$T(pwm) = \frac{1}{f(pwm)} \quad \text{atau} \quad T(pwm) = 256\ x\ \frac{1}{25\ Mhz}$$

$$T(pwm) = \frac{1}{97\ Khz} \quad \text{atau} \quad T(pwm) = 256\ x\ 40\ ns$$

$$T(pwm) = 10{,}240\ ns \approx 10.24\ \mu s$$

Persamaan menghitung waktu *on*:

$$t_{on}\ (pwm) = (data_{ref} + 1)\ x\ \frac{1}{25\ Mhz}$$

$$t_{on}\ (pwm) = (data_{ref} + 1)\ x\ 40\ ns$$

Untuk memastikan bahwa frekuensi pwm memenuhi untuk pengendali motor BLDC yang setidaknya 10 kali lebih besar dari frekuensi output komutasi,

Persamaan menghitung putaran elektrik motor BLDC:

Misalkan maksimum kecepatan motor bldc yang digunakan ialah 700 RPM

$$f(electric) = \frac{RPM\ x\ pole\ pairs}{60}$$

$$f(electric) = \frac{700\ x\ 15}{60}$$

$$f(electric) = 175\ KHz$$

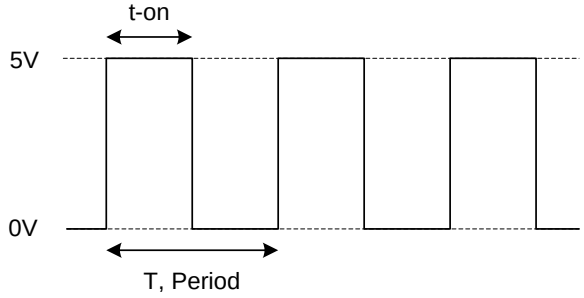
Frekuensi pwm paling rendah pada kecepatan maksimum BLDC:

$$f(pwm) = \frac{f(pwm)}{f(electi}$$

$$f(pwm) = \frac{97\ Khz}{175}$$

$$f(pwm) \approx 558\ KHz$$

Duty cycle PWM dihitung dengan menggunakan persamaan dibawah:



$$duty\ cycle = \frac{t_{on}}{T}\ x\ 100\%$$

Berdasarkan hasil simulasi Gambar 34 misalkan data_ref yang digunakan ialah 127 maka duty cyclenya ialah:

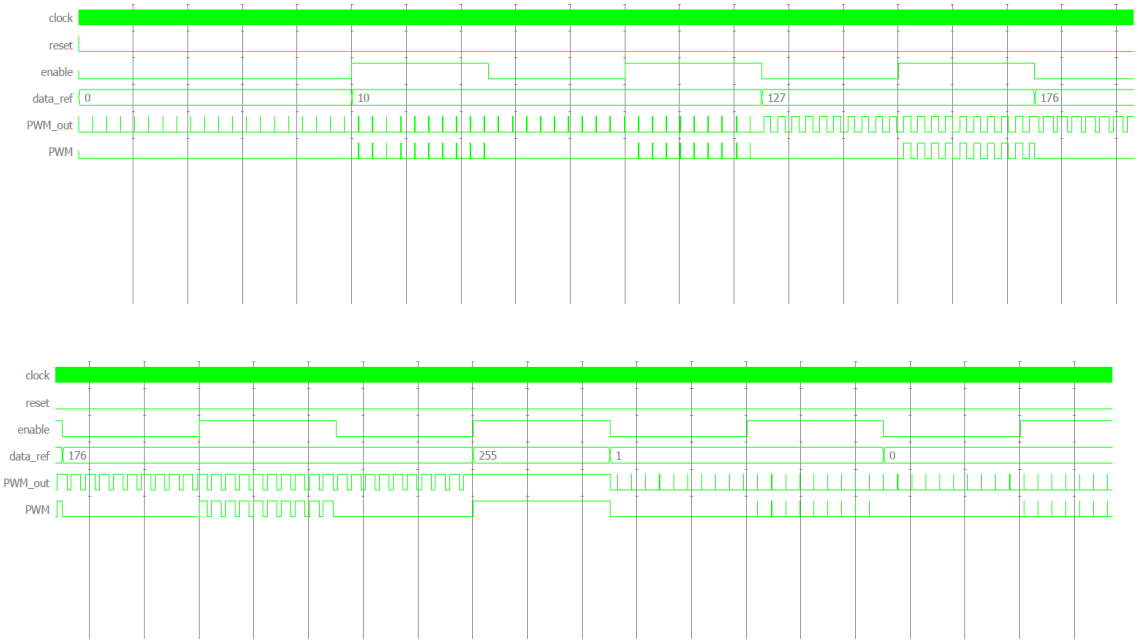
$$t_{on} = (127 + 1)\ x\ 40ns$$

$$t_{on} = 5{,}120\ ns\ \approx 5.12\ \mu s$$

sehingga

$$duty\ cycle = \frac{5.12\ \mu s}{10.24\ \mu s}\ x\ 100\%$$

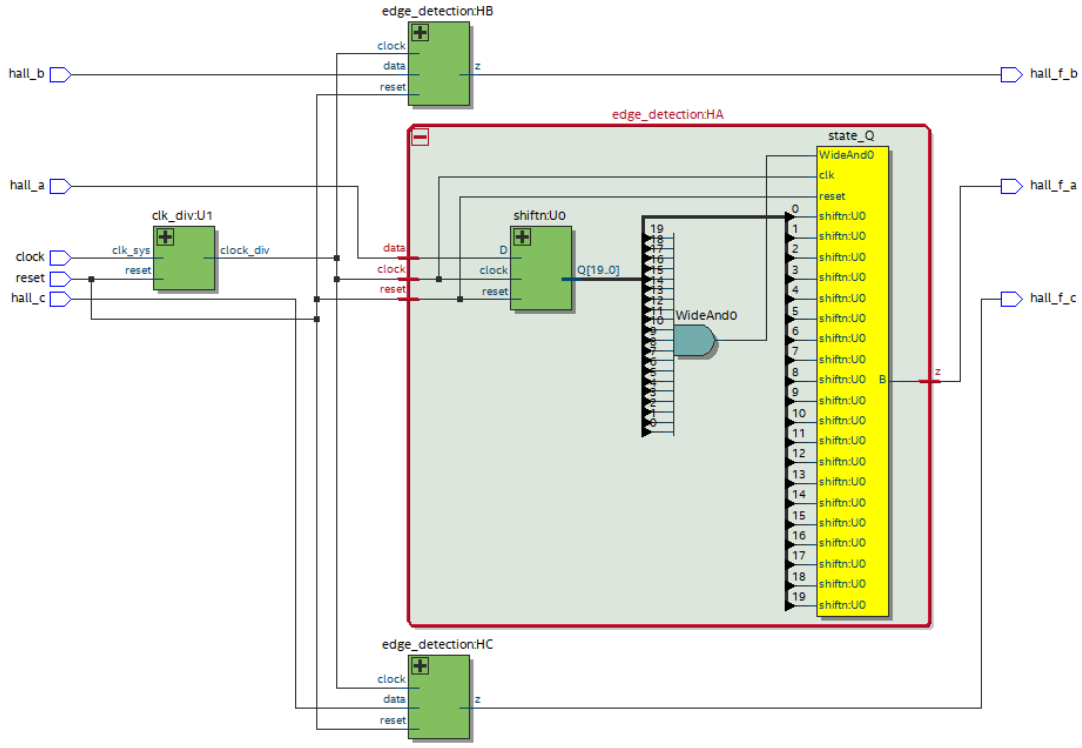
$$duty\ cycle\ \approx 50\%$$



Gambar 34. *Waveforms* hasil simulasi modul PWM generator

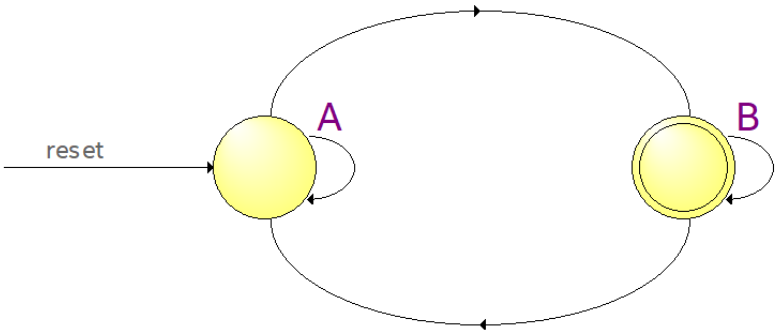
4.4 Modul Edge Detection / filter sensor hall effect

Input hall sensor selain digunakan untuk mendeteksi posisi rotor, juga digunakan untuk menghitung/mengukur revolusi motor BLDC. Sinyal hall sensor merupakan sinyal analog, sehingga diperlukan filter baik secara hardware maupun software, pada pembahasan ini focus pada filter secara software. Filter dengan modul Edge Detection digunakan untuk memfilter sinyal sensor hall effect yang masuk ke fpga, kemudian hasil filter dari modul ini digunakan untuk sebagai input untuk modul speed calculation, tujuan dari modul ini ialah untuk memperkecil error dari permbacaan sensor hall effek. Adapun modul yang telah dirancang ditunjukkan pada Gambar 35, yang terdiri atas register geser, mesin keadaan dan pembagi frekuensi.



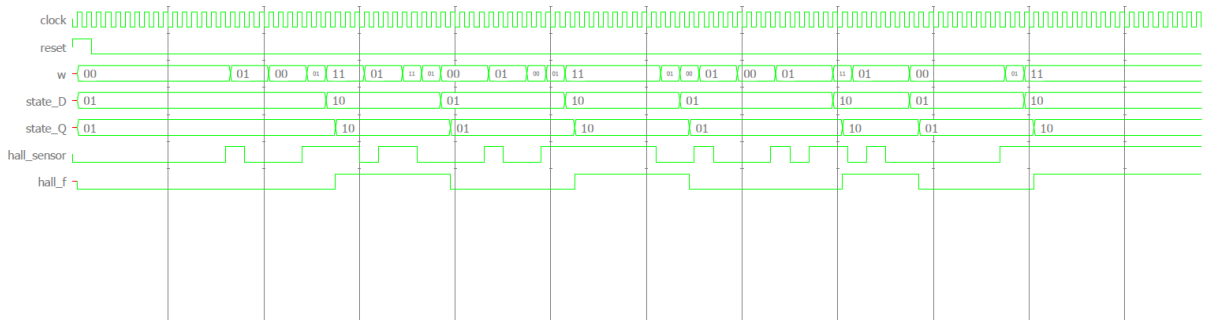
Gambar 35. *Netlist viewer* modul edge detection

Prinsip kerja dari edge detection ini, yaitu menggunakan mesin keadaan sebagai control untuk mengatur transisi antara ‘1’ dan ‘0’ dengan input 2 bit data. Bit-1 merupakan *Reduction AND* dari n-bit data output register geser, dan bit-0 merupakan *Reduction OR* dari n-bit data output register geser. Terdapat 2 keadaan yaitu A dan B (Gambar 36) yang direpresentasikan dengan ‘01’ dan ‘10’ secara berurutan.



Gambar 36. diagram mesin keadaan modul edge detection terjemahaan *compiler* Quartus Prime 18.1

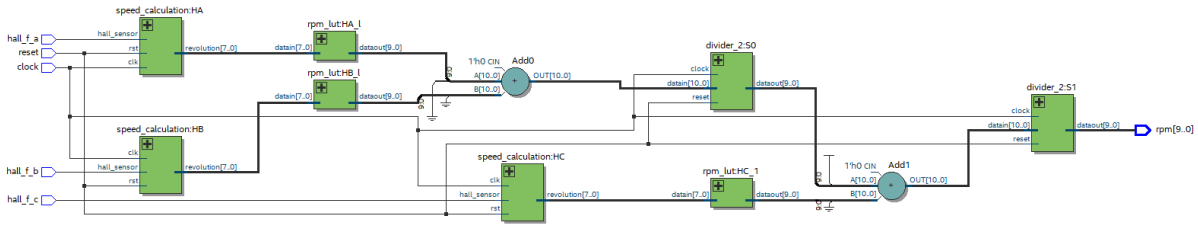
kemudian jika input w = ‘11’ dan state sekarang adalah A maka keadaan selanjutnya adalah B, jika output selain ‘11’ maka keadaan akan tetap pada A. selanjutnya jika input w = ‘00’ dan state sekarang adalah B maka keadaan selanjutnya adalah A, tetapi jika input selain itu maka state sekarang tetap B. pada Gambar 37 output dari modul edge_detection yaitu hall_f akan bernilai logika ‘1’ pada state B, dan bernilai logika ‘0’ pada state A.



Gambar 37. *Waveforms* hasil simulasi modul edge detection

4.5 Modul Speed Calculation

Secara keseluruhan modul untuk menghitung revolusi per menit dari motor BLDC, di rancang dalam modul speed calculation wrapper (Gambar 38), modul ini meliputi modul speed calculation, rpm_lut, dan divider.

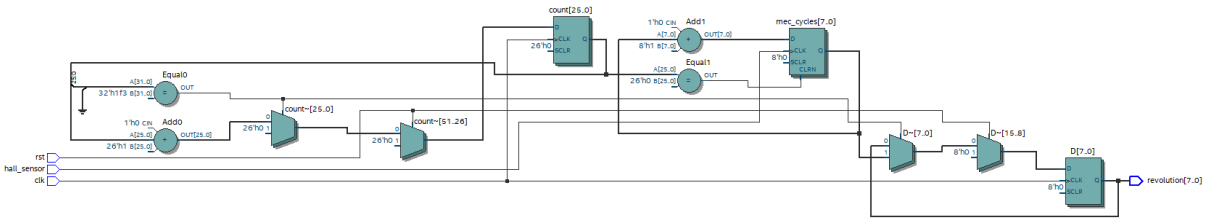


Gambar 38. *Netlist viewer* modul speed calculation wrapper

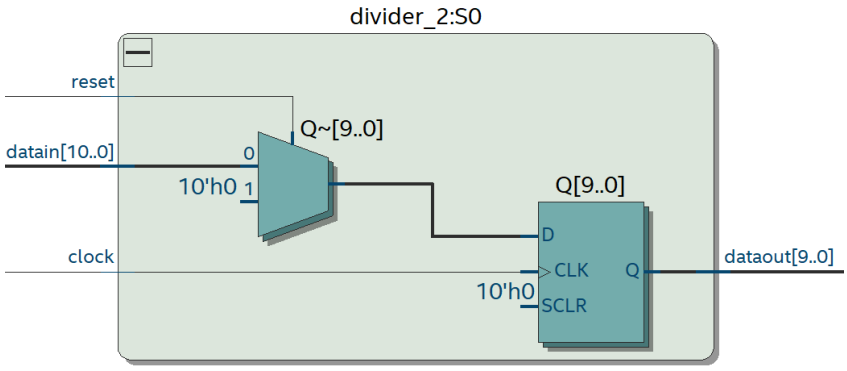
Pertama, modul speed calculation (Gambar 39) digunakan untuk menghitung siklus putaran elektrik motor bldc. Untuk menghitung siklus digunakan sinyal input dari hall sensor yang telah di filter dengan modul edge_detection, satu gelombang sinyal hall sensor menandakan satu putaran elektrik. Dengan counter dihitung banyaknya siklus hall sensor selama satu detik. Hasil dari counter tersebut selanjutnya disimpan

kedalam register, dan counter di reset ke nilai 0 untuk menghitung siklus selanjutnya. Hasil atau output dari modul speed calculation merupakan frekuensi elektrik motor, untuk mengubah dari fekuensi elektrik motor ke nilai rpm (revolusi per menit) digunakan modul RPM_lut yang mana outputnya merupakan nilai RPM yang telah dihitung dengan menggunakan persamaan:

$$RPM = \frac{60}{15}\ x\ f(elektrik)$$



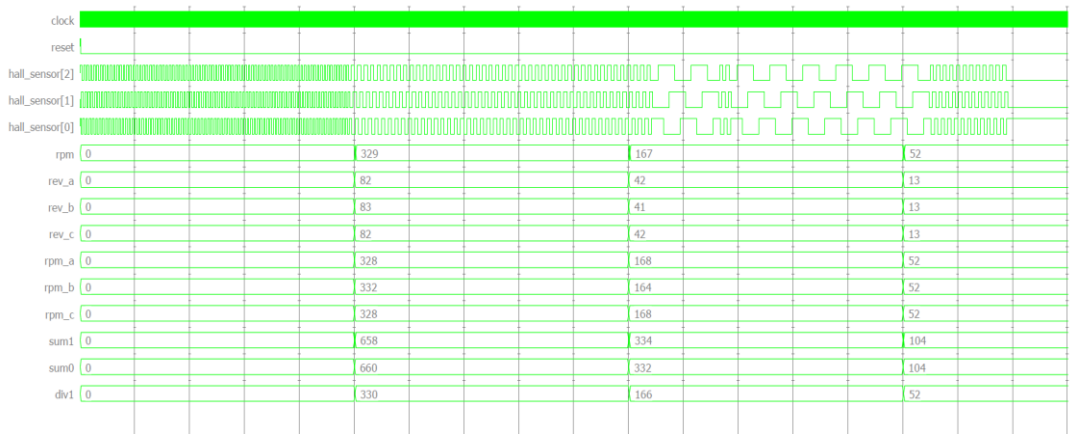
Gambar 39. *Netlist viewer* modul speed calculation



Gambar 40. *Netlist viewer* modul divider/pembagi

Daripada menggunakan satu sensor hall efek saja untuk menghitung revolusi motor BLDC, lebih baik menggunakan tiga sensor hall efek karena dapat meningkatkan presisi atau ketepatan pembacaan kecepatan motor BLDC. Prinsip kerjanya yaitu masing masing sensor hall efek dihitung rpm-nya dengan modul speed_calculation dan RPM_lut, kemudian hasil rpm dari masing-masing sensor dihitung rata-ratanya. Berdasarkan hasil simulasi Gambar 41, Pertama rpm sensor hall 1 dan sensor hall 2 dijumlahkan kemudian dibagi dengan dua dengan

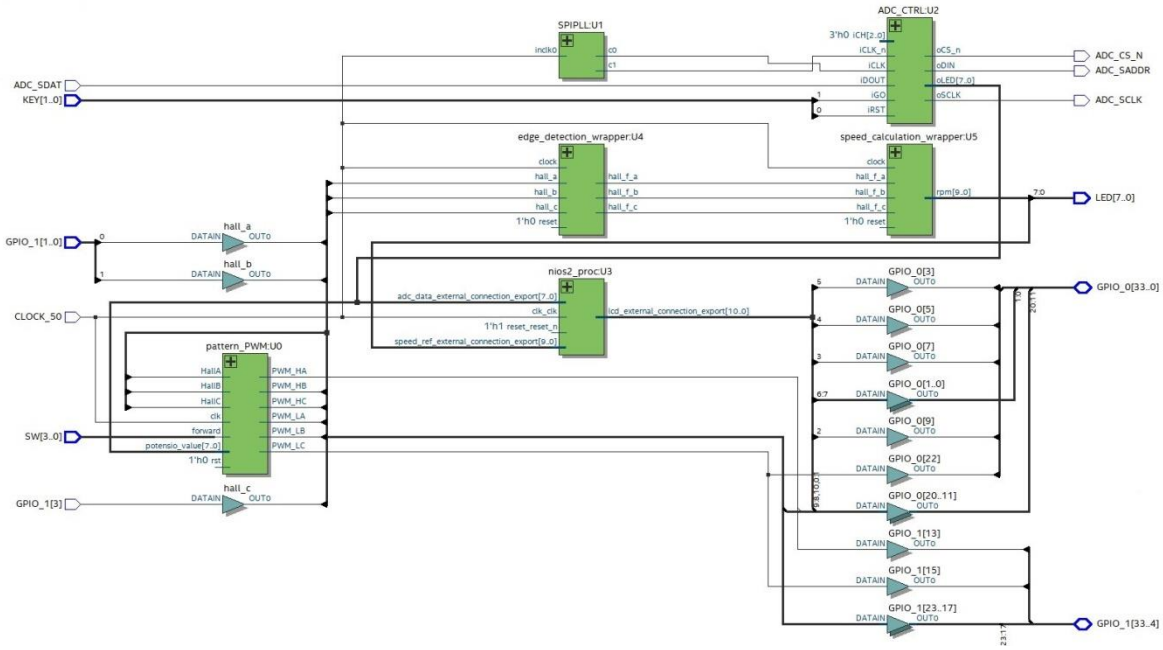
menggunakan *left-to-right shift register* n bit sebanyak satu kali kemudian hasilnya disimpan pada net div1. Kemudian rpm sensor hall 3 dijumlahkan dengan data div1, kemudian dibagi dua dengan menggunakan *left-to-right shift register* n bit sebanyak satu kali kemudian hasilnya disimpan pada net rpm.



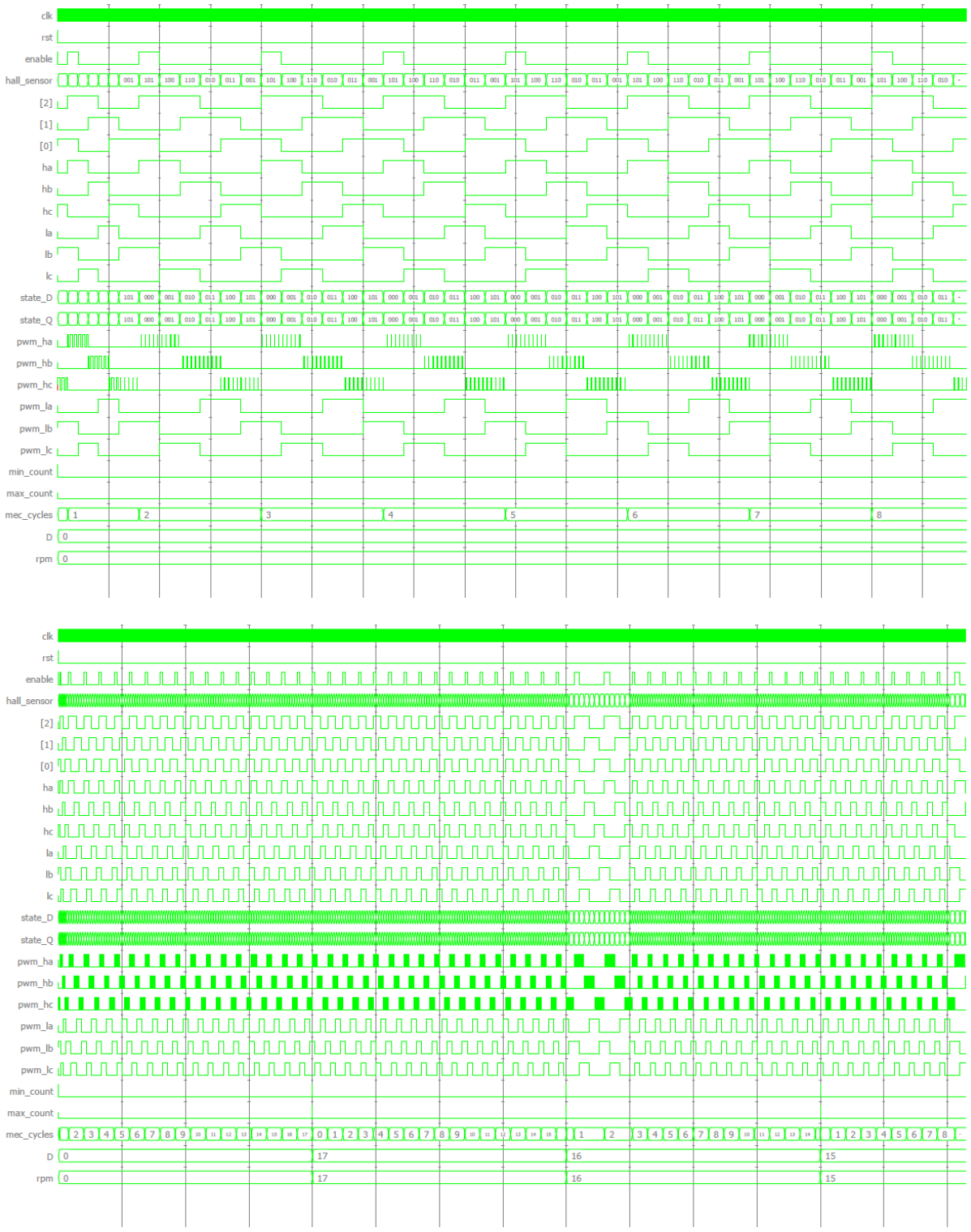
Gambar 41. *Waveforms* hasil simulasi modul speed calculation

4.6 Modul Pengendali Motor BLDC

Gambar 42 menunjukkan top-level dari pengendali yang dirancang pada FPGA, dan Gambar 43 menunjukkan hasil simulasi dari keseluruhan modul yang telah dirancang



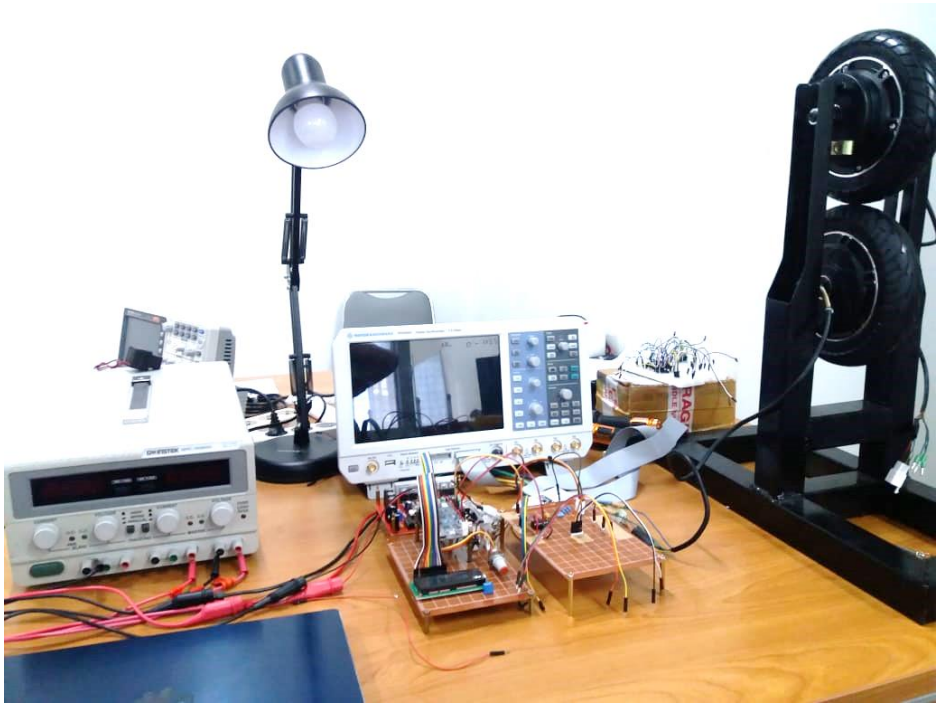
Gambar 42. *Netlist viewer* modul pengendali Motor BLDC



Gambar 43. *Waveforms* hasil simulasi pengendali motor BLDC

BAB V
HASIL IMPLEMENTASI KENDALI MOTOR BLDC

Pengendali motor bldc di implementasikan pada motor BLDC dengan spesifikasi yaitu daya 350 watt, tegangan 36 volt. Adapun untuk mengetahui jumlah pasang kutub dari motor BLDC yang digunakan dilakukan secara manual dengan board FPGA dan sensor hall effect sebagai input terhubung dengan GPIO FPGA kemudian jika sensor hall effect bernilai logika ‘1’ atau tegangan sekitar 3.3 volt maka LED akan menyala, dari hasil pengamatan tersebut di peroleh bahwa jumlah pasang kutub adalah 15. Jumlah pasang kutub ini digunakan untuk menghitung kecepatan motor. Kemudian, untuk *experiment set-up* diilustrasikan pada Gambar 44.

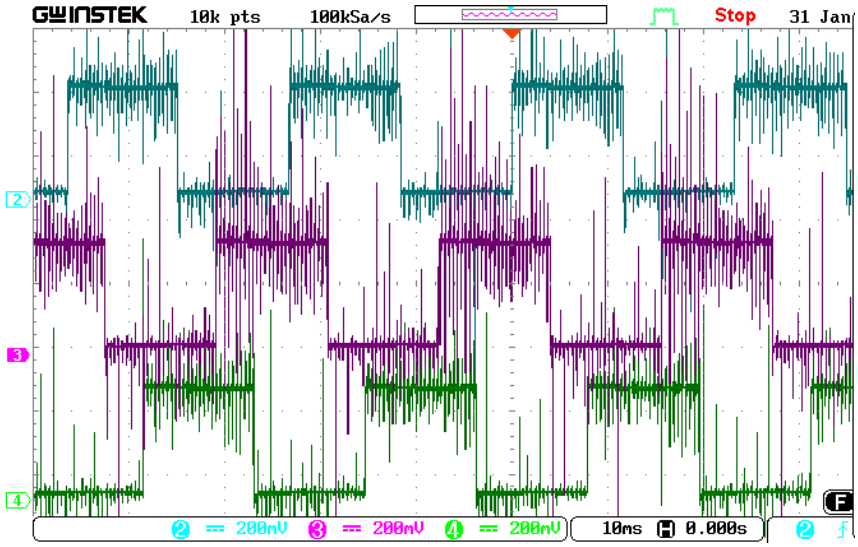


Gambar 44. *Experiment set-up*

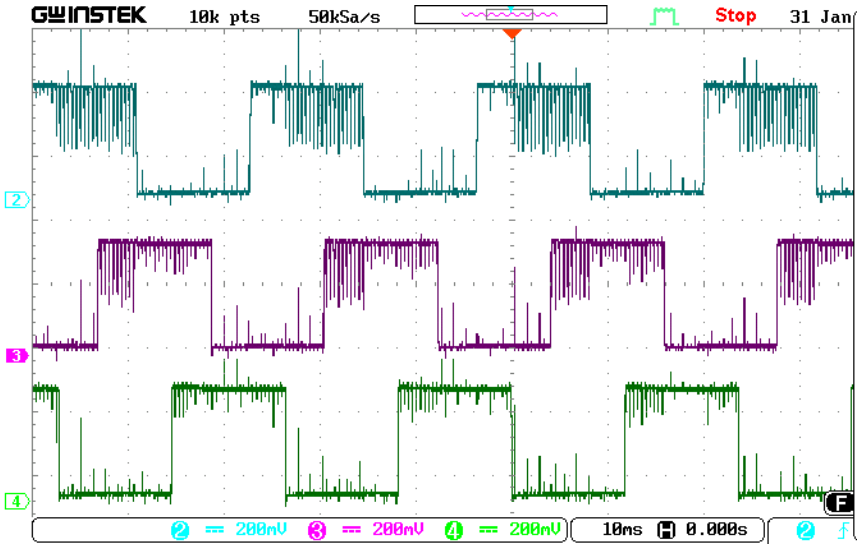
5.1 Hasil Filter hall sensor

Sinyal filter tidak hanya di filter secara software tatepi juga di filter dengan hardware, yaitu menggunakan *active low pass filter*. Komponen yang digunakan ialah resistor 3.3k, kapasitor 2.2nf dan op-amp LM324 dan motor berputar dengan kecepatan 173 rpm diperoleh hasil pada Gambar 46. jika dibandingkan dengan Gambar 45 dengan tanpa *active low pass filter* terlihat

bahwa rangkaian *active low pass filter* berkerja walaupun masih ada frekuensi tinggin yang dilewatkan.



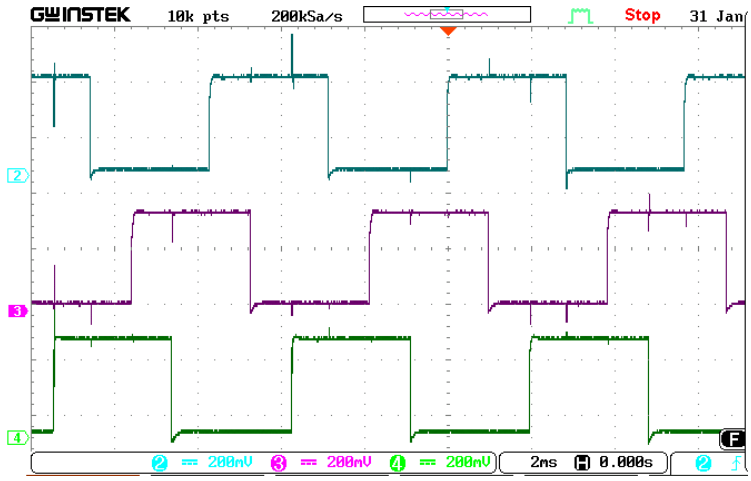
Gambar 45. Sensor hall efek pada kecepatan 173 rpm tanpa *active low pass filter*



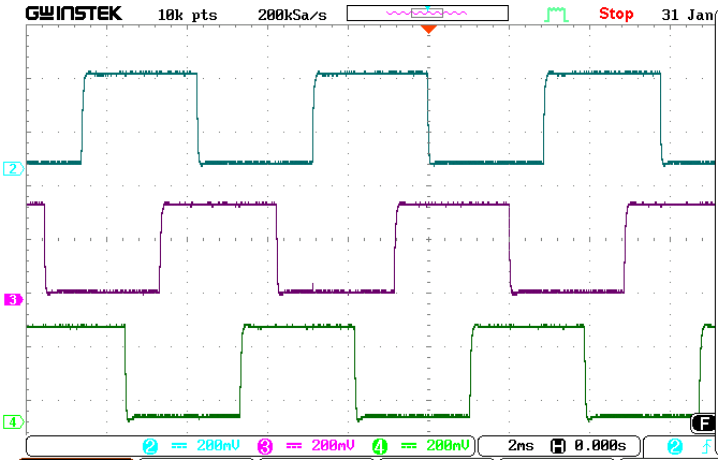
Gambar 46. sensor hall efek pada kecepatan 173 rpm dengan *active low pass filter*

Pada kecepatan maksimum yaitu sekitar 696 rpm sinyal sensor hall efek cenderung lebih baik walaupun tanpa *active low pass filter* ditunjukkan pada Gambar 47, akan tetapi masih ada beberapa riak. Dengan *active low pass filter* ditunjukkan dengan Gambar 48 riak tersebut dapat diatasi dengan baik.

Filter sensor hall efek dalam pengendali motor BLDC ini sangat penting, pertama untuk menghasilkan pembacaan kecepatan yang presisi. kedua, data hall sensor yang dihasilkan lebih tepat dalam mendeteksi posisi rotor.



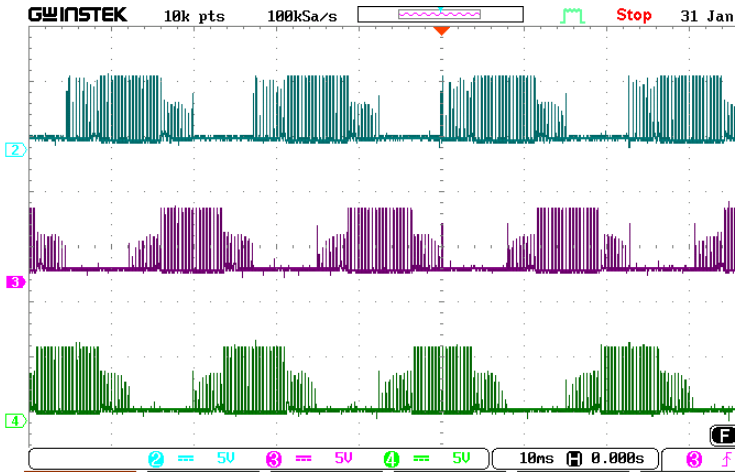
Gambar 47. Sensor hall efek pada kecepatan maksimum 696 rpm tanpa *active low pass filter*



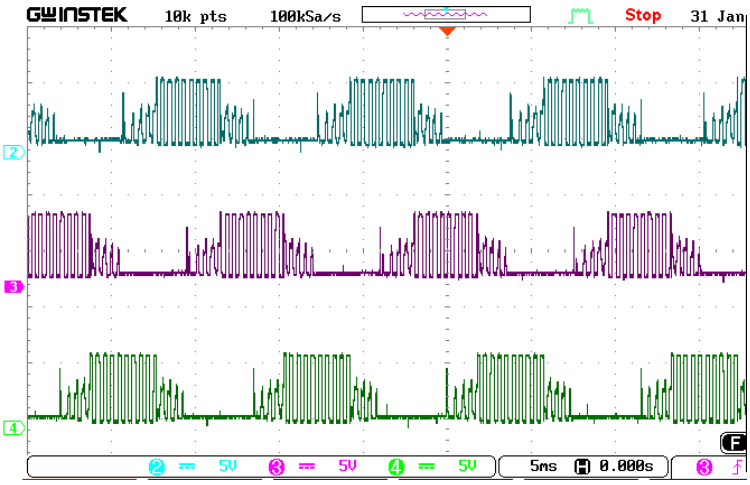
Gambar 48. Sensor hall efek pada kecepatan maksimum 690 rpm dengan active low pass filter

5.2 Tegangan line motor BLDC

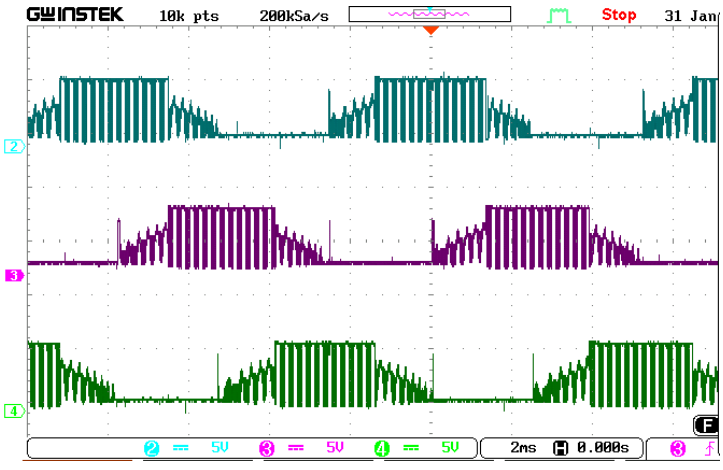
Tegangan line motor BLDC ditunjukkan pada Gambar 49 a, b, c, dan d dengan duty cycle 25%, 50%, 75% dan 100% secara berurutan.



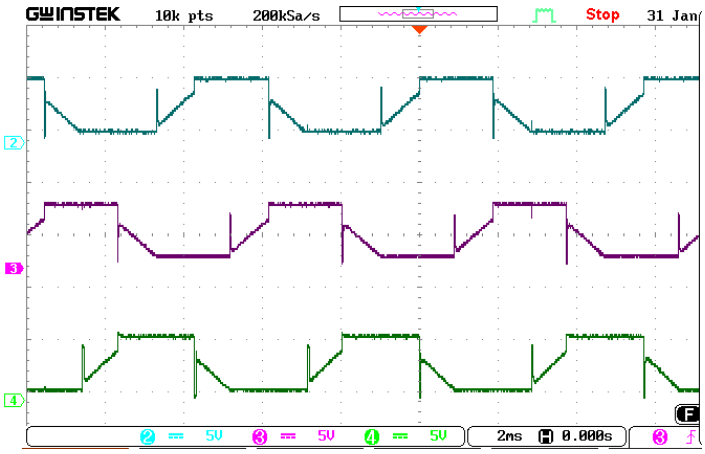
a) Duty cycle 25%



b) duty cycles 50%



c) duty cycles 75%

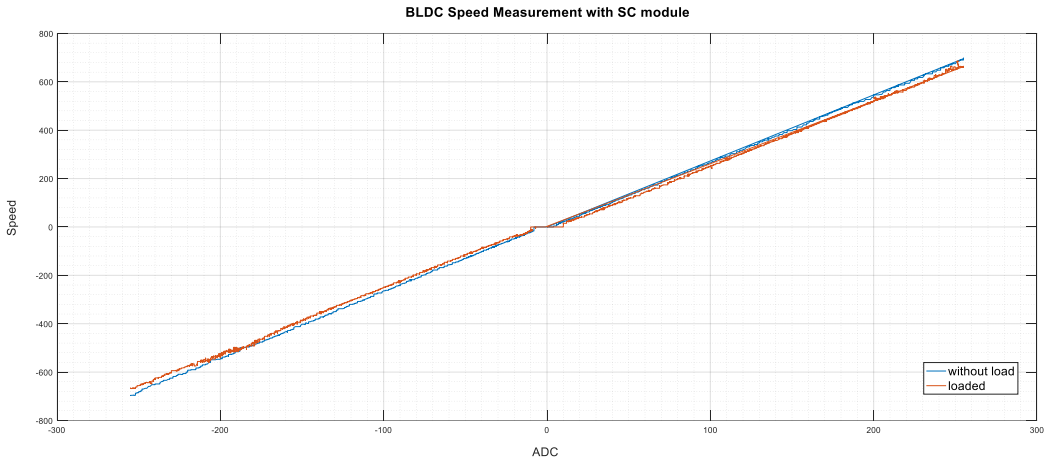


d) Duty cycles 100%

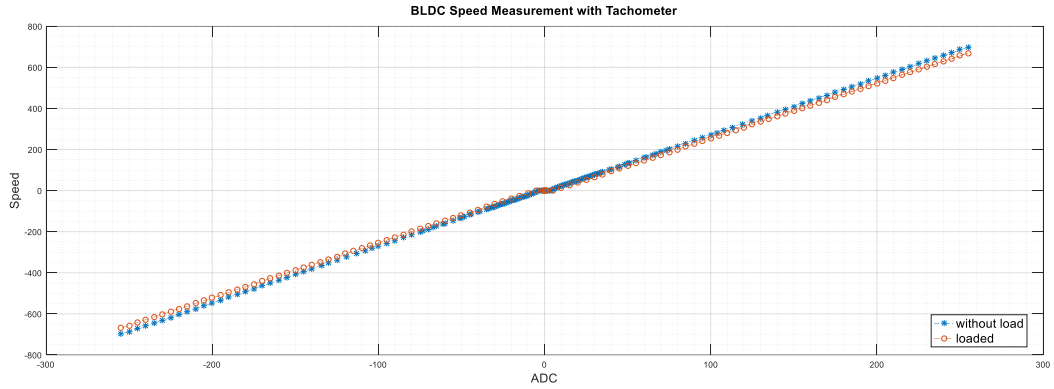
Gambar 49. tegangan line motor BLDC

5.3 Perbandingkan Hasil Pembacaan Modul Speed Calculation (SC) dan Tachometer

hasil perbandingan kecepatan dengan menggunakan modul speed calculation diperoleh grafik pada Gambar 50, sumbu x merupakan nilai ADC dan sumbu y merupakan nilai kecepatan. Berdasarkan grafik tersebut maksimum kecepatan pada duty cycle 100% jika tanpa beban ialah 696 RPM baik berputar searah jarum jam maupun berlawanan dan dengan beban maksimum kecepatan ialah 667 RPM berlaku untuk CW maupun CCW. Selajutnya dengan menggunakan pengukur keceapatan tachometer diperoleh grafik pada Gambar 51. hasilnya tidak jauh berbeda dengan hasil pembacaan dengan modul SC yaitu pada kecepatan maksimum dengan duty cycle 100% untuk tanpa beban ialah 698 RPM baik CW maupun CCW dan untuk berbaban ialah 668 RPM baik CW maupun CCW.



Gambar 50. Grafik kecepatan berbeban dan tanpa beban menggunakan modul SC



Gambar 51. Grafik kecepatan berbeban dan tanpa beban menggunakan Tachometer

Salanjutnya untuk memverifikasi ketepatan pembacaan modul SC dilakukan perbandingan antara hasil pembacaan modul SC dengan Tachometer diperoleh grafik Gambar 52 untuk tanpa beban dan grafik Gambar 53 untuk berbeban.

Persentase keakuratan/error modul SC terhadap tachometer untuk tanpa beban:

$$\%\ error = \left| \frac{RPM\ Modul\ SC - RPM\ Tachometer}{RPM\ Tachometer}\ x\ 100\% \right|$$

Data kecepatan dari modul SC diambil pada plot yang memiliki nilai paling jauh dari nilai kecepatan Tachometer dengan duty cycle yang sama yaitu pada nilai ADC 220:

$$\% \, error = \left| \frac{594 - 604}{604} \, x \, 100\% \right|$$

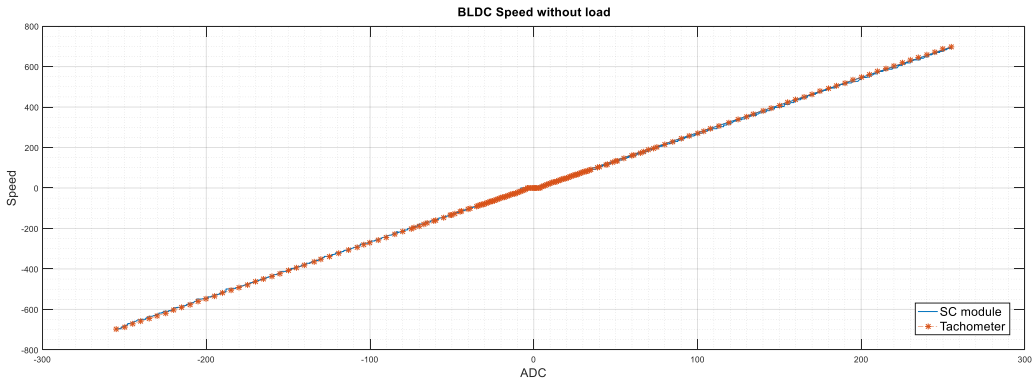
$$\% \, error = \ 1.5 \, \%$$

Untuk berbeban sampel data yang diambil ialah pada nilai adc 185, Persentase kuakuratan/error diuraikan berikut:

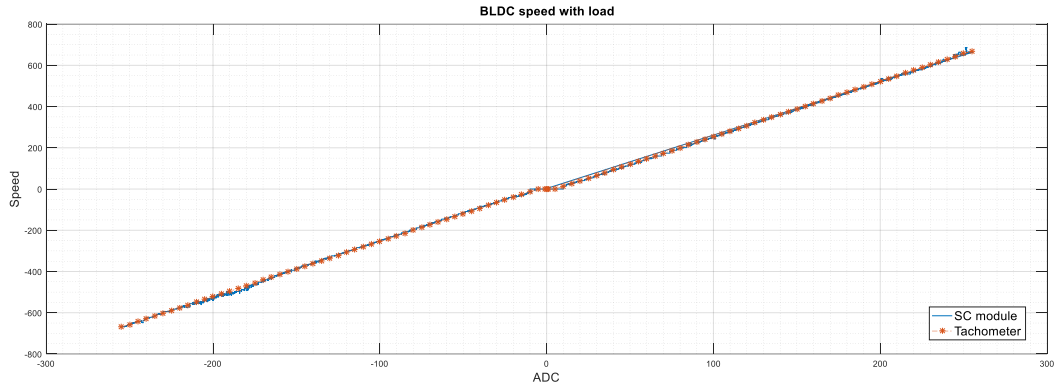
$$\% \, error = \left| \frac{492 - 482}{482} \, x \, 100\% \right|$$

$$\% \, error = \ 2.1\%$$

Berdasarkan hasil perhitungan *error* kecepatan baik dengan beban maupun tanpa beban diperoleh presisi ketepatan modul SC untuk mengukur kecepatan putar Motor BLDC lebih dari 97%. Kekurangan dari modul SC ialah nilai kecepatan hanya di perbaharui setiap detik.



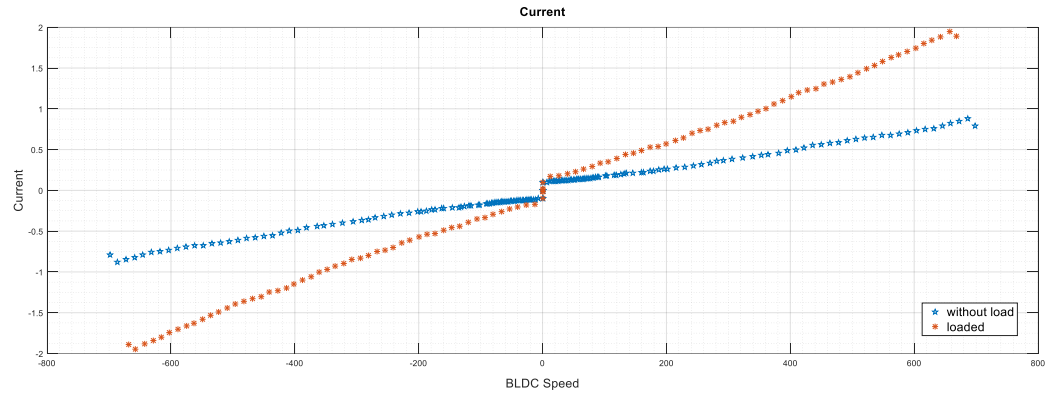
Gambar 52. Grafik kecepatan tanpa beban menggunakan speed SC dan tachometer



Gambar 53. grafik kecepatan berbeban menggunakan modul SC dan tachometer

5.4 Perbandingan Arus berbeban dengan tanpa beban

Perbandingan arus dengan dan tanpa beban di tunjukkan pada grafik Gambar 54, sumbu x merupakan nilai kecepatan dan sumbu y merupakan nilai arus. Maksimum arus yang mengalir ke motor pada tanpa beban ialah 0.88 Amper dan maksimum arus yang mengalir ke motor pada saat berbeban ialah 1.90 Amper.



Gambar 54. grafik perbandingan arus beban dan tanpa beban

Hasil perbandingkan performa controller terhadapat kecepatan motor bldc dengan penelitian [14] ditunjukkan pada tabel 4.

Tabel 4. Hasil Perbandingan Kecepatan

	Penelitian [14]	Penelitian ini
Tanpa Beban	600 RPM (0.76 A)	696 RPM (0.88 A)
Berbeban	500 RPM (1.34 A)	667 RPM (1.90 A)

Bersarkan tabel di atas, maka performa controller yang dirancang lebih baik dibandingkan dengan controller sebelumnya.

5.5 Perbandingan Frekuensi PWM terhadap kecepatan motor BLDC

Pada Tabel 5 menunjukkan perbandingan kecepatan putar motor BLDC dengan frekuensi pwm yang berbeda adapun data kecepatan diperoleh dengan menggunakan tachometer.

Tabel 5. Frekuensi PWM terhadap kecepatan motor BLDC

Base Clock	PWM frequency	RPM on Duty cycles 25%	RPM on Duty cycles 50%	RPM on Duty cycles 100%
50 Mhz	195 Khz	176.8	353.8	699
25 Mhz	97 Khz	167.6	343.6	698.7
12.5 Mhz	48 Khz	164.1	342.5	698.7
6.25 Mhz	24 Khz	162.3	340.7	698.6
3.125 Mhz	12 Khz	160	335.9	698
1.562 Mhz	6 Khz	172	368	696

Berdasarkan tabel 5, performa kecepatan putar motor bldc pada duty cycles 100% menghasilkan RPM yang hampir sama untuk setiap frekuensi pwm. Hal ini terjadi karena pada duty cycles 100% sinyal yang di*feed* ke inverter hanya berupa logika 1 atau 0 bergantung pada komutasi sensor hall effect. Berbeda dengan duty cycles 25% dan 50% atau kurang dari 100%, RPM yang dihasilkan dipengaruhi oleh besarnya frekuensi PWM. Dari tabel 5 di atas frekuensi pwm yang dapat digunakan untuk memutar motor BLDC dengan performa yang baik yaitu frekuensi $\geq$ 20 Khz, Berdasarkan hasil pengamatan pada motor bldc menghasilkan *noise* penggunakan frekuensi pwm $<$ 20 Khz akan menghasilkan *noise, noise* ini akan bertambah besar jika frekuensi pwm semakin kecil.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Bardasarkan seluruh proses implementasi dan pengujian yang telah dilakukan telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Metode six-step komutasi untuk mengendalikan motor BLDC diimplementasikan pada piranti FPGA menggunakan model mesin keadaan dengan frekuensi 50 Mhz.
2. kecepatan motor BLDC dapat dikendalikan dengan menggunakan piranti potensio meter untuk mengatur duty cycle PWM.
3. Karakteristik motor BLDC yang telah diuji memiliki kecepatan maksimal masing-masing pada pengujian tanpa beban, dan dengan beban yaitu 698 RPM dan 668 RPM baik berputar searah jarum jam maupun berlawanan arah jarum jam.
4. modul speed calculation yang dirancang dapat menghitung kecepatan putar motor BLDC dengan ketepatan hingga 98%.

6.2 Saran

Adapun saran yang penulis dapat berikan yaitu, dengan adanya modul speed calculation sebagai feedback dari motor BLDC, maka dapat dirancang system pengendali dengan system tertutup dengan PID controller.

DAFTAR PUSTAKA

[1] P. Mishra, A. Banerjee and M. Ghosh, "FPGA Based Real-Time Implementation of Quadral-Duty Digital PWM Controlled Permanent Magnet BLDC Drive," *IEEE/ASME Transactions on Mechatronics,* vol. 25, no. 3, pp. 1456-1467, June 2020.

[2] G. Mihalache and A. -D. Ioan, "FPGA Implementation of BLDC Motor Driver of BLDC Motor Driver with Hall Sensor Feedback," *2018 International Conference and Exposition on Electrical and Power Engineering (EPE),* pp. 0624-0629, 2018.

[3] A. Usman and B. S. Rajpurohit, "Design and Control of a BLDC Motor drive using Hybrid Modeling Technique and FPGA based Hysteresis Current Controller," *2020 IEEE 9th Power Indian International Conference (PIICON),* pp. 1-5, 2020.

[4] B. Tufekci, B. Onal, H. Dere and H. F. Ugurdag, "Efficient FPGA Implementation of Field Oriented Control for 3-Phase Machine Drives," *2020 IEEE East-West Design & Test Symposioum (EWDTS),* pp. 1-5, 2020.

[5] C. Bucella, C. Cecati and H. Latafat, "Digital Control of Power Converters—A Survey," *IEEE Transaction on Industrials on Industrial Informatics,* vol. 8, no. 3, pp. 166-171, 2012.

[6] J. Cervantes, E. Córdova, A. I. S. Marrufo, I. U. P. Monarrez and M. Nadayapa, "BLDC Motor Commutation Based on DSP Builder for FPGA," *2016 13th International Conference on Power Electronics (CIEP),* pp. 166-171, 2016.

[7] M. Jain and S. S. Wiiloamson, "Suitability Analysis of In-Wheel Motor Direct Drives for Electric and Hybrid Electric Vehicles," *2009 IEEE Electrical Power & Energy Conference (EPEC),* pp. 1-5, 2009.

[8] M. Yildirim, M. Polat and H. Kurut, "A Survey on Comparison of Electric Motor Types and Drives Used for Electric Vehicles," *2014 16th International Power Electronics and Motion Control Conference and Exposition,* pp. 218-223, 2014.

[9] F. Qi, D. Scharfenstein, C. Weiss, C. Muller and U. Schwarzer, Motor Handbook, Munich/Germany, 2019.

[10] G. Scelba, G. D. Donato, M. Pulvirenti, F. G. Capponi and G. Scarcella, "Hall-Effect Sensor Fault Detection, Indentification and Compensation in Brushless DC Drives," *IEEE Transactions on Industry Applications*, vol. 52, no. 2, pp. 1542-1554, March-April 2016.

[11] A. Tashakori and M. Ektesabi, "Stability Analysis of Sensorles BLDC Motor Drive Using Digital PWM Technique for Electric Vehicles," *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pp. 4898-4903, 2012.

[12] A. Tashakori, M. Hassanudeen and M. Ektesabi, "FPGA Based Controller Drive of BLDC Motor Using Digital PWM Technique," *2015 IEEE 11th International Conference on Power Electronics and Drive Systems*, pp. 658-662, 2015.

[13] Semiconductor, F. (2009). *AN4058, BLDC Motor Control with Hall Effect Sensors Using the 9S08MP - Application Notes*. www.freescale.com.

[14] M. F. Sachruddin, "Sistem Kendali Motor BLDC dengan Hall Sensors Berbasis CPLD," Universitas Hasanuddin, Makassar, 2022.

[15] International Rectifier, “Power Mosfet,” IRFB3607PbF datasheet, Jan. 2012.

[16] W. S. Colton, "Design and Prototyping for Brushless Motors and Motor Control," Massachusetss Institute of Technology, 2010.

[17] Thosiba, “Thosiba Photocoupler IRED & Photo-IC,” TLP250 datasheet, Nov. 1990 [Revised Jun. 2019].

[18] Internation Rectifier, “High and Low Side Driver,” IR2110 datasheet, Mar. 2019.

[19] Ward Brown, “Brushless DC Motor Control Made Easy”, Appl. Note 857, Microchip, p.4, 2002.

[20] Padmaraja Yedamale, "Brushless DC motor Fundamentals”, Appl. Note 885 Microchip, p.8, 2003.

[21] Padmaraja Yedamale, "Brushless DC Motor Control Using PIC18FXX31 MCUs", Appl. Note *899, Microchip*, p.4, 2004.

LAMPIRAN

Lampiran 1 Paper Seminar Hasil

Perancangan Kendali Digital Berbasis FPGA untuk Mengendalikan Motor BLDC

Nurul Hidayat
Department Electrical Engineering
Universitas Hasanuddin
Makassar, Indonesia
hidayatnl17d@student.unhas.ac.id

Faizal Arya Samman
Department Electrical Engineering
Universitas Hasanuddin
Makassar, Indonesia
faizalas@unhas.ac.id

Rhiza S. Sadjad
Department Electrical Engineering
Universitas Hasanuddin
Makassar, Indonesia
rhizas@unhas.ac.id

Acc. Seminar Hasil 28/02/2022

Acc Seminar Hasil 25/11/2021

***Abstrak*— Pada penelitian ini dirancang pengendali digital berbasis FPGA untuk mengendalikan kecepatan motor BLDC (*brushless direct current*). Digunakan metode *trapezoid control* atau *6-step* komutasi dengan *unipolar independent pwm switching*. Metode komutasi motor bldc berdasarkan sensor hall efek yang tertanam dalam motor bldc. metode ini diimplementasikan dengan menggunakan model mesin keadaan. Selain metode control, didesain pula modul untuk mengukur kecepatan motor bldc. Pengukur kecepatan menggunakan *counter* untuk menghitung siklus *electric* motor dan *look-up* tabel untuk menyimpan hasil perhitungan revolusi per menit motor. Agoritma *control* dan pengukur kecepatan ditulis menggunakan Verilog HDL (*hardware description language*) dan diverifikasi melalui simulasi menggunakan modelsim-intel. Kemudian, diimplementasikan pada board FPGA terasic DE0-NANO EP4CE22. Semua sistem dibangun berdasarkan *base clock* FPGA 50 MHz. Pengujian kendali motor dilakukan pada motor BLDC 350 watt 36 v dengan inverter tiga fasa sebagai drivernya.**

Keywords*—BLDC motor, FPGA, *Trapezoid Control*, *Unipolar Independent* PWM *Switching

I. PENDAHULUAN

Pengunaan *Field Programmable Logic Array* (FPGA) dalam aplikasi motor control meningkat belakangan tahun terakhir, ditunjukkan pada publikasi [1] [2] [3] [4], hal ini dipengaruhi karena pemrogramannya yang fleksibel. Selain itu, dalam mengontrol motor diperlukan *high-speed switching* dengan alasan tersebut FPGA digunakan [5]. Sebagai tambahan, fitur yang paling penting adalah bahwa FPGA memungkinkan untuk mengembangkan algoritma yang kompleks dan dapat melakukan proses secara parallel (*concerrent*) [6]. Adapun untuk memrogram pada FPGA diperlukan pengetahuan sistem digital dan HDL (*hardware description language*).

Dalam penelitian ini digunakan motor BLDC tiga fasa, motor ini memiliki beberapa keuntungan, seperti harganya yang relatif lebih murah, biaya perawatan yang rendah, dan menghasilkan noise (suara bising) yang rendah disebabkan oleh tidak adanya komutator makanik melainkan menggunakan komutator elektronik. Selain itu kekurangan BLDC yaitu dioperasikan pada kecepatan rendah, getaran kecil terjadi selama putaran kecepatan rendah. Namun, getaran berkurang pada kecepatan tinggi.

Motor DC tanpa sikat terdiri atas bagian yang diam yang disebut stator dan bagian yang bergerak disebut rotor. Ruang antara stator dan rotor disebut celah udara (*air gap*), pada stator sendiri terdiri atas belitan yang terhubung dengan konfigurasi star ataupun konfigurasi delta sedangkan pada rotor terdapat magnet permanen. penempatan rotor pada motor DC tanpa sikat pada umumnya ada dua yaitu rotor di dalam (*Inrunner*) dan rotor di luar (*Outrunner*). Keuntungan motor BLDC rotor internal terletak pada inersia rotornya yang rendah dan pembuangan panas lebih baik. Sebaliknya, untuk motor rotor eksternal, kumparan penghasil panas diisolasi dari lingkungannya oleh rumah rotor dan magnet. Motor dengan rotor eksternal memiliki keunggulan untuk aplikasi yang diproduksi secara massal, karena dapat diproduksi dengan lebih murah.

Fasa adalah kumpulan individu dari belitan dengan terminal tunggal yang dapat diakses dari luar motor. Kebanyakan motor *brushless* adalah tiga fasa. Setiap loop individu dari kawat membentuk fasa lilitan disebut belokan (*turn*). Kutub merupakan kutub magnet permanen tunggal, baik utara atau selatan. Jumlah kutub minimum adalah dua, tetapi motor dapat memiliki jumlah kutub yang genap. Motor yang lebih besar cenderung memiliki kutub yang lebih banyak. Jumlah kutub tidak berhubungan langsung dengan jumlah slot, meskipun terdapat kombinasi umum antara slot dan jumlah kutub agar performa motor bekerja lebih baik. Pada motor dengan lebih dari dua kutub, penting untuk menentukan perbedaan antara kecepatan sudut mekanis dan kecepatan sudut elektrik. Kecepatan sudut mekanis adalah kecepatan fisik yang diukur dengan busur derajat atau *tachometer*, kecepatan sudut listrik mewakili posisi relatif dalam satu periode magnet, yang membentang di dua kutub.

persamaan sederhana untuk menghitung kecepatan sudut electric dan mekanik yaitu

$$\omega_e = p\omega_m \qquad (1)$$

Dimana ω_e merupakan kecepatan sudut elektrik, p merupakan jumlah pasang kutub, dan ω_m merupakan kecepatan sudut elektrik, misalkan motor adalah 4 kutub.

$$\omega_e = 2 \times 180^\circ \qquad (2)$$

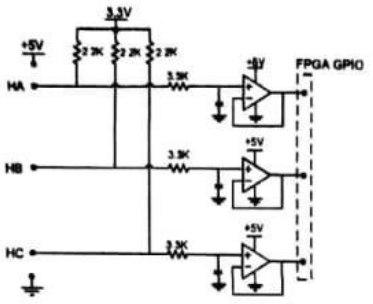
Berarti dalam setegah putaran mekanik sama dengan 360° putaran elektrik dan pada 1 putaran mekanik penuh sama dengan 720° putaran elektrik. Jika hasil dari kecepatan sudut dikonversi ke dalam satuan frekuensi maka digunakan persamaan 3, hasil dari persamaan ini akan digunakan untuk menghitung kecepatan putaran motor dalam RPM.

$$f = \frac{\omega}{2\pi} \qquad (3)$$

Dimana, f merupakan frekuensi atau biasa juga disimbolkan dengan huruf v.

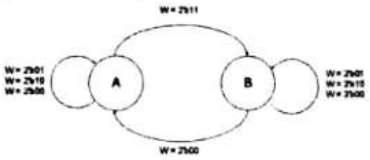
Telah dijelaskan sebelumnya bahwa komutasi motor BLDC dikendalikan secara elektronik. Agar dapat berputar,

B. *Filter Sensor Hall Efek*



Gambar 4. Rangkaian *Active low pass filter*

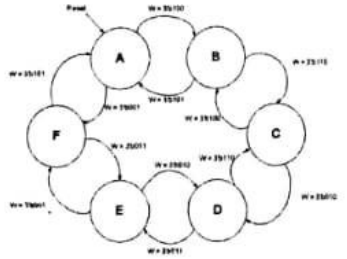
Filter sensor hall efek menggunakan filter analog dan digital. Pertama filter analog, rangkaian filter ini menggunakan pull-up resistor dengan resistor bernilai 2.2K ohm yang terhubung dengan sumber 3.3V dan *active low fass filter* dengan menggunakan resistor 3.3K ohm yang parallel dengan kapacitor yang terhubung dengan pin *non-iverting op-amp*. Kedua, filter secara digital diimplementasikan dalam code Verilog, filter ini didesain dengan menggunakan mesin keadaan dan register geser. Register geser digunakan untuk menyampel data sensor hall efek hingga n bit data dengan *clock* 10MHz.



Gambar 5. Mesin keadaan filter digital sensor hall efek

C. *Komutasi Menggunakan Sensor Hall Effect*

Algoritma *6-step* komutasi didesain dengan menggunakan model mesin keadaan yang ditunjukkan pada gambar 6. struktur code Verilog mesin keadaan terbagi menjadi tiga yaitu program mendefinisikan keadaan berikutnya, mendefinisikan output, dan mendefiniskan blok sekuensial.



Gambar 6. Diagram mesin keadaan komutasi 6 langkah

Hasil dari mesin keadaan ini berupa 6 sinyal yang digunakan untuk mengaktifkan mosfet, dengan 2 sinyal bernilai 1 secara bersamaan yaitu 1 dari mosfet atas dan 1 lagi dari mosfet bagian bawah ditunjukkan pada gambar 7. Konfigurasi ini berlangsung secara terus menerus berdasarkan tabel komputasi menggunakan sensor hall effect yang ditunjukkan pada tabel 1 dan tabel 2

D. *PWM Generator dan Switching PWM*

Dalam FPGA, PWM didesain secara digital dengan menggunakan *counter* dan *comparator*, dengan input nilai data dari ADC sebanyak 8 bit. Pertama *counter* menghitung dari nilai 0 hingga 255 dengan referensi clock 25 MHz, kemudian hasil *counter* setiap *clock* sebagai input dari *comparator*. Selanjutnya, *comparator* membandingkan nilai dari *counter* dan nilai ADC dengan *clock* yang sama dengan *counter*, jika nilai ADC lebih besar daripada nilai *counter* maka *comparator* akan bernilai '1' jika sebaliknya akan bernilai '0'. Untuk frekuensi pwm yang didesain di uraikan pada persamaan 4.

$$f(pwm) = (Base\ clock)/2^n \qquad (4)$$

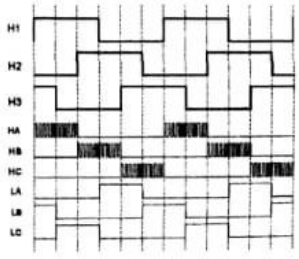
TABEL I. POLA KOMUTASI SEARAH JARUM JAM

Step	Sensor hall efek			Fasa		
	Ha	*Hb*	*Hc*	*A*	*B*	*C*
1	1	0	1	DC+	DC-	Off
2	1	0	0	DC+	Off	DC-
3	1	1	0	Off	DC+	DC-
4	0	1	0	DC-	DC+	Off
5	0	1	1	DC-	Off	DC+
6	0	0	1	Off	DC-	DC+

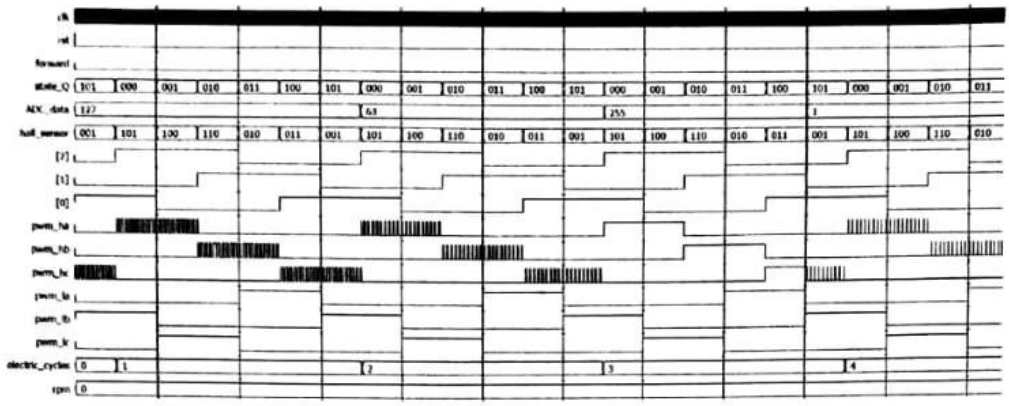
TABEL II. POLA KOMUTASI BERLAWANAN ARAH JARUM JAM

Step	Sensor hall efek			Fasa		
	Ha	*Hb*	*Hc*	*A*	*B*	*C*
6	0	0	1	Off	DC+	DC-
5	0	1	1	DC+	Off	DC-
4	0	1	0	DC+	DC-	Off
3	1	1	0	Off	DC-	DC+
2	1	0	0	DC-	Off	DC+
1	1	0	1	DC-	DC+	Off

Dimana n adalah jumlah bit data ADC. Yang perlu diperhatikan dalam mendesain PWM ialah frekuensi sinyal PWM setidaknya 10 kali lebih cepat dari maksimum frekuensi motor (*electrical*) [11]. dan terakhir switching PWM menggunakan metode Unipolar *Independent PWM Switching* ditunjukkan pada Gambar 7.



Gambar 7. *Unipolar Independent PWM Switching*



Gambar 8. Modelsim-Altera Simulation

E. *Pengukur Kecepatan*

Pengukur kecepatan menggunakan input dari sensor hall. pada gambar 1 menggambarkan bahwa 6 pola komutasi dari 3 hall sensor menghasilkan 1 putaran secara elektrik, untuk mencapai satu putaran mekanik penuh diperlukan putaran elektrik sebanyak jumlah pasang kutub motor. modul penghitung kecepatan motor terdiri atas *counter, flip flop D,* dan *lookup table*. *Counter* digunakan untuk menghitung putaran eletrik dari motor berdasarkan input sensor hall, flip flop D digunakan untuk menyimpan nilai kecepatan sebelumnya selama kurang lebih 1 detik, dan *lookup* table digunakan untuk menyimpan hasil perhitungan dari persamaan 5

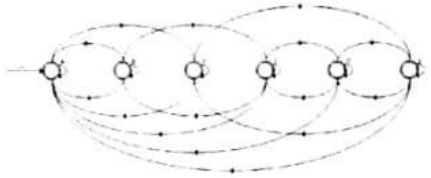
$$RPM = \frac{60}{pairs\ of\ poles} x\ f \qquad (5)$$

dimana f merupakan banyaknya putaran elektrik dalam satu detik (Hz). *Lookup table* menyimpan data rpm untuk setiap nilai f, selain itu untuk mendapatkan hasil perhitungan kecepatan yang lebih presisi digunakan 3 modul pengukur kecepatan untuk masing masing sensor hall effect kemudian dijumlahkan dan dihitung rata-ratanya dengan menggunakan modul divider. Adapun sensor hall effect yang digunakan telah difilter oleh modul filter sensor hall effect agar pembacaan sensor hall lebih tepat sehingga perhitungan lebih presisi.

III. Hasil Simulasi Algorima Kendali Motor BLDC

Simulasi dilakukan menggunakan Modelsim-Intel FPGA Starter Edition 10.5b, dan program dibuat pada Quartus Prime 18.1 Lite Edtion untuk mememudahkan mengecek error atau melihat hasil rtl dari kode Verilog yang dibuat. Tujuan dari simulasi ialah untuk memastikan code yang dibuat berfungsi dengan output yang sesuai selain itu, untuk mengoptimalkan kinerja sistem dan mengurangi kerugian sebelum diimplementasikan pada perangkat keras.

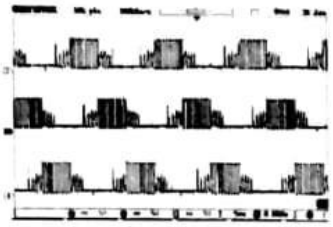
Algoritma 6-step komutasi dirancang berdasarkan state mechine. Diagram *state mechine* ditunjukkan pada gambar 9, terdiri dari 6 keadaan yaitu keadaan A, B, C, D, E, dan F. Masing-masing keadaan direpresentasikan dalam tiga bit parameter secara berurutan yaitu "000", "001", "010", "011", "100", "101". Urutan keadaan untuk putaran CW (*clockwise*) atau setara jarum jam ialah A-B-C-D-E-F, sedangkan keadaan untuk putaran CCW (*couterclockwise*) atau berlawanan arah jarum jam ialah F-E-D-C-B-A arah putaran ini diatur oleh sinyal "*forward*". Nilai sensor hall efek dari MSB ke LSB merupakan gabungan dari sensor hall efek 1, 2, dan 3 secara berurutan.



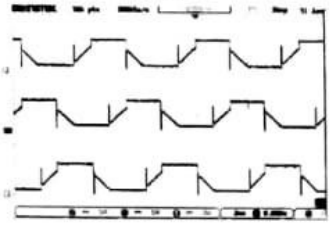
Gambar 9. Diagram state machine modul komutasi

Output mesin keadaan di tunjukkan pada gambar 8 yaitu 6 sinyal komutasi berupa sinyal ha, hb, hc, la, lb, lc. ha dan lb merepresentasikan sinyal untuk mosfet fasa A *high side* dan *low side* secara berurutan, begitupun hb, dan lb untuk mosfet fasa B, dan hc dan lc untuk mosfet fasa C. dengan metode *unipolar independent PWM switching*, hanya output high side ha, hb, dan hc sinyal pwm. Duty cycle pwm diatur oleh *counter* dan *comparator* berdasarkan 8 bit data ADC.

Pada penelitian ini juga dibuat modul *speed calculation*, Pertama, modul speed calculation digunakan untuk menghitung siklus putaran elektrik motor bldc. Untuk menghitung siklus digunakan sinyal input dari 3 hall sensor yang telah di filter, satu gelombang sinyal hall sensor menandakan satu putaran elektrik ditunjukkan pada gambar 8 melalui sinyal electric_cycles. Dengan *counter* dihitung banyaknya siklus hall sensor selama satu detik. Hasil dari counter tersebut selanjutnya disimpan kedalam register, dan counter di reset ke nilai '0' dengan menggunakan *asinkronous reset* untuk menghitung siklus selanjutnya. output dari modul *speed calculation* merupakan revolusi permenit (RPM) motor yang direpresentasikan dengan sinyal rpm 10 bit.



(a)



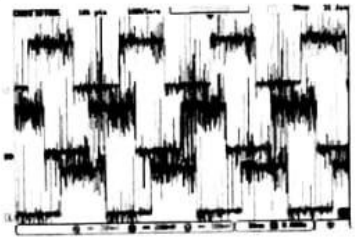
(b)

Gambar 10. (a) duty cycle 50%, (b) duty cycle 100%

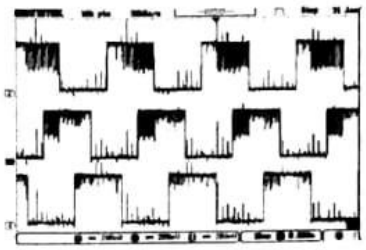
IV. HASIL IMPLEMENTASI

Pengendali motor bldc di implementasikan pada motor BLDC dengan spesifikasi yaitu daya 350 watt, tegangan 36 volt. Adapun untuk mengetahui jumlah pasang kutub dari motor BLDC yang digunakan dilakukan secara manual dengan board FPGA dan sensor hall effect sebagai input terhubung dengan GPIO FPGA kemudian jika sensor hall effect bernilai logika '1' atau tegangan sekitar 3.3 volt maka LED akan menyala, dari hasil pengamatan tersebut di peroleh bahwa jumlah pasang kutub adalah 15. Jumlah pasang kutub ini digunakan untuk menghitung kecepatan motor. Hasil percobaan diperoleh gambar 10 (a), menunjukkan tegangan emf motor BLDC pada duty cycle 50 % dengan kecepatan motor sekitar 350 rpm tanpa beban, dan 10 (b) pada duty cycle 100% dengan kecepatan motor maksimum 696 rpm.

Selain itu, output hasil filter hall sensor ditunjukkan pada gambar 11(b) pada kecepatan 173 rpm. jika dibandingkan dengan gambar 11(a) dengan tanpa *active low pass filter* terlihat bahwa rangkaian *active low pass filter* berkerja walaupun masih ada frekuensi tinggi yang dilewatkan. Pada kecepatan maksimum yaitu sekitar 696 rpm sinyal sensor hall efek cenderung lebih baik walaupun tanpa *active low pass filter*.



(a)

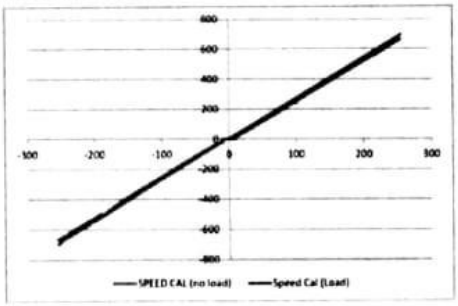


(b)

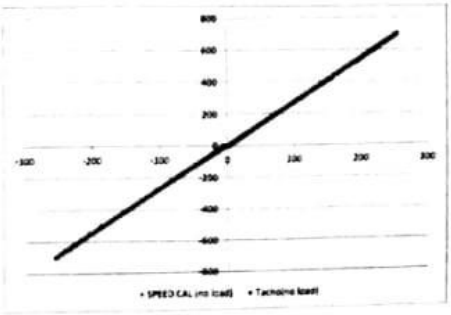
Gambar 11. (a) hall sensor tanpa filter, (b) hall sensor dengan active low pass filter

Hasil perbandingan kecepatan dengan menggunakan modul speed calculation diperoleh grafik pada gambar 12, sumbu x merupakan nilai ADC dan sumbu y merupakan nilai kecepatan. Berdasarkan grafik tersebut maksimum kecepatan pada duty cycle 100% jika tanpa beban ialah 696 RPM baik berputar searah jarum jam maupun berlawanan dan dengan beban maksimum kecepatan ialah 667 RPM berlaku untuk CW maupun CCW.

Salanjutnya untuk memverifikasi ketepatan pembacaan modul SC dilakukan perbandingan antara hasil pembacaan modul SC dengan Tachometer diperoleh grafik gambar 13. Terakhir, Perbandingan arus dengan dan tanpa beban di tunjukkan pada grafik gambar 14, sumbu x merupakan nilai kecepatan dan sumbu y merupakan nilai arus. Maksimum arus yang mengalir ke motor pada tanpa beban ialah 0.88 A dan maksimum arus yang mengalir ke motor pada saat berbeban ialah 1.90 A.



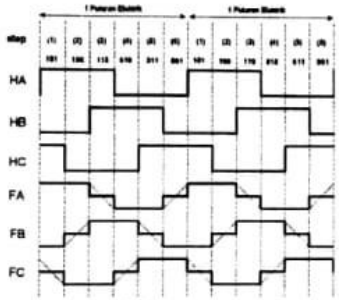
Gambar 12. Grafik kecepatan tanpa beban dan berbeban dengan modul SC



Gambar 13. Grafik kecepatan modul SC vs Tachometer

gulungan stator harus diberi energi secara berurutan. Penting untuk mengetahui posisi rotor untuk memahami belitan mana yang akan diberi *energize* mengikuti urutan komutasi. Posisi rotor dideteksi menggunakan sensor hall efek yang tertanam ke dalam stator. Prinsip kerja dari sensor hall efek ialah akan mengeluarkan logika '1' (high) jika mendeteksi atau berada didekat medan magnet. Biasanya, tiga sensor hall efek dipindahkan (*displaced*) pada 120 derajat listrik. Setiap Sensor menghasilkan nilai '1' (high) untuk 180 derajat putaran listrik dan nilai '0' (low) untuk 180 derajat putaran listrik [7] [8].

Trapezoid control juga dikenal sebagai *six-step control*, metode ini digunakan untuk mengontrol Inverter tiga fasa/Driver motor BLDC (gambar 3) dalam enam rangkaian pensaklaran. Dalam teknik ini hanya dua fasa motor BLDC yang diberi energy (*energized*) pada setiap urutan pensaklaran sedangkan fase lainnya tidak aktif. Pembalikan urutan *switching* mengubah arah putaran motor. Kecepatan motor berbanding lurus dengan lebar sinyal sensor hall effect. Tegangan EMF balik, sinyal pergantian, arus fasa dan pola switching enam langkah dari motor BLDC tiga fasa ditunjukkan pada Gambar 1 [9].



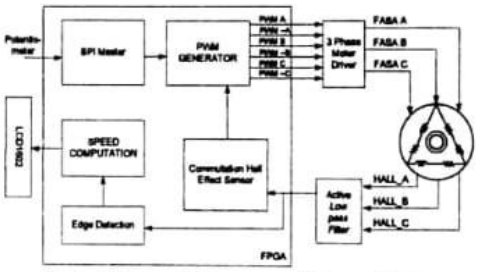
Gambar 1. Bentuk gelombang kotak yang dihasilkan oleh *trapezoid control*

Kecepatan motor BLDC berbanding lurus dengan tegangan yang diberikan ke stator. Kecepatan di mana rotor dipaksa ke posisi berikutnya ditentukan oleh kekuatan gaya magnet, hal ini ditentukan oleh tegangan yang diberikan pada belitan stator. Dengan menggunakan PWM dengan frekuensi yang lebih tinggi dari pada frekuensi komutasi, besarnya tegangan yang diberikan pada stator dapat dengan mudah dikontrol, sehingga kecepatan motor dapat dikontrol.

Kontroler PWM enam langkah tipikal menggunakan salah satu dari dua teknik PWM [10] yaitu : *Unipolar (soft) PWM Switching* merupakan Teknik ini mengacu pada fase motor yang dialihkan sedemikian rupa sehingga salah satu fase mengembalikan arus sementara modulasi PWM terjadi di fase lain, atau dengan kata lain hanya mosfet bagian atas saja yang di drive dengan sinyal pwm, untuk mosfet bagian bawah berupa *switch on* dan *off* saja. *Bipolar (hard) PWM Switching* merupakan Teknik ini mengacu pada tegangan yang melewati dua fase sebagai dimodulasi dengan PWM, baik input maupun output arus sedang dimodulasi. Maksudnya ialah baik mosfet bagian atas maupun bagian bawah di drive dengan sinyal pwm

II. DESKRIPSI SISTEM

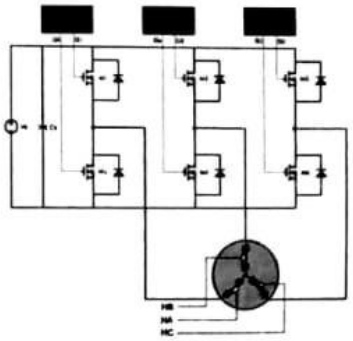
Sistem kendali motor BLDC yang dirancang terbagi menjadi 2 bagian yaitu Sistem perangkat keras (*Hardware*) dan Perangkat lunak (*Software*). Bagian perangkat keras berupa Driver Tiga Fasa yang terdiri atas mosfet, bus kapasitor, driver mosfet, filter hall sensor, dan pembagi tegangan. Filter hall sensor dengan *active low pass filter* dibuat berdasarkan saran dari penelitian [11] untuk mengurangi noise dan meningkatkan performa. Sedangkan software berupa modul modul yang ditulis dalam Verilog HDL (Hardware Description Language) yang akan diimplementasikan pada Terasic DE0-NANO board yang di dukung dengan chip FPGA Intel/Altera cyclone-iv dengan seri EP4CE22F17C6. Secara abstrak sistem yang dirancang ditunjukkan pada gambar 2. Berbeda dengan penelitian [11] yang menggunakan CPLD, pada penelitian ini menggunakan FPGA. Tidak hanya itu, beberapa modul yang belum ada pada penelitian [11] seperti modul speed computation, modul Edge Detection juga didesain. Selanjutnya, untuk modul komutasi di desain dengan menggunakan model mesin keadaan, berbeda dengan modul komutasi pada [11] yang hanya menggunakan procedural assignment yang outputnya hanya diperungaruhi oleh perubahan input. Diharapakan dengan modul mesin keadaan tersebut dapat meningkatkan performa motor BLDC.



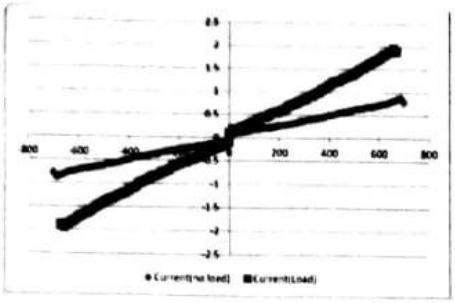
Gambar 2. Diagram blok sistem pengendalian motor BLDC

A. *Inverter Tiga Fasa*

Pada penelitian ini driver motor bldc menggunakan inverter tiga fasa dengan mosfet IRFB3607 sebagai saklar semikonduktor. Gambar 3 menunjukkan rangkaian inverter tiga fasa. Pengoperasian sakelar semikonduktor tidak dapat dioperasikan pada lengan yang sama untuk menghindari hubungan pendek pada rangkaian.



Gambar 3. Rangkaian inverter tiga fasa



Gambar 14. Grafik arus berbeban dan tanpa beban

Hasil perbandingkan performa controller terhadapat kecepatan motor bldc dengan penelitian [11] ditunjukkan pada tabel 3.

TABEL III. HASIL PERBANDINGAN KECEPATAN

	Penelitian [XX]	Penelitian ini
Tanpa Beban	600 RPM (0.76 A)	696 RPM (0.88 A)
Berbeban	500 RPM (1.34 A)	667 RPM (1.90 A)

apakah sama bebannya ?

Bersarkan tabel di atas, maka performa controller yang dirancang lebih baik dibandingkan dengan controller sebelumnya.

V. KESIMPULAN

Pertama, Metode six-step komutasi untuk mengendalikan motor bldc diimplementasikan pada fpga menggunakan algoritma mesin keadaan, dengan pwm generator kecepatan motor bldc dapat diatur berdasarkan potensio meter. Kelebihan menggunakan mesin keadaan ialah kecepatan motor dapat dengan mudah diatur baik pada kecepatan rendah maupun kecepatan tinggi. Motor dapat berputar dengan kecepatan paling rendah pada 5 rpm tanpa beban dan 12 rpm dengan beban, dengan kecepatan maksimum 696 rpm tanpa beban dan 668 rpm dengan beban.

Kedua, modul speed calculation yang dirancang dapat menghitung kecepatan putar motor bldc dengan ketepatan hingga 98%.

REFERENCES

[1] P. Mishra, A. Banerjee and M. Ghosh, "FPGA Based Real-Time Implementation of Quadral-Duty Digital PWM Controlled Permanent Magnet BLDC Drive," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1456-1467, June 2020.

[2] G. Mihalache and A. -D. Ioan, "FPGA Implementation of BLDC Motor Driver of BLDC Motor Driver with Hall Sensor Feedback," *2018 International Conference and Exposition on Electrical and Power Engineering (EPE)*, pp. 0624-0629, 2018.

[3] A. Usman and B. S. Rajpurohit, "Design and Control of a BLDC Motor drive using Hybrid Modeling Technique and FPGA based Hysteresis Current Controller," *2020 IEEE 9th Power Indian International Conference (PIICON)*, pp. 1-5, 2020.

[4] B. Tufekci, B. Onal, H. Dere and H. F. Ugurdag, "Efficient FPGA Implementation of Field Oriented Control for 3-Phase Machine Drives," *2020 IEEE East-West Design & Test Symposioum (EWDTS)*, pp. 1-5, 2020.

[5] C. Bucella, C. Cecati and H. Latafat, "Digital Control of Power Converters—A Survey," *IEEE Transaction on Industrials on Industrial Informatics*, vol. 8, no. 3, pp. 166-171, 2012.

[6] J. Cervantes, E. Córdova, A. I. S. Marrufo, I. U. P. Monarrez and M. Nadayapa, "BLDC Motor Commutation Based on DSP Builder for FPGA," *2016 13th International Conference on Power Electronics (CIEP)*, pp. 166-171, 2016.

[7] G. Scelba, G. D. Donato, M. Pulvirenti, F. G. Capponi and G. Scarcella, "Hall-Effect Sensor Fault Detection, Indentification and Compensation in Brushless DC Drives," *IEEE Transactions on Industry Applications*, vol. 52, no. 2, pp. 1542-1554, March-April 2016.

[8] A. Tashakori and M. Ektesabi, "Stability Analysis of Sensorles BLDC Motor Drive Using Digital PWM Technique for Electric Vehicles," *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pp. 4898-4903, 2012.

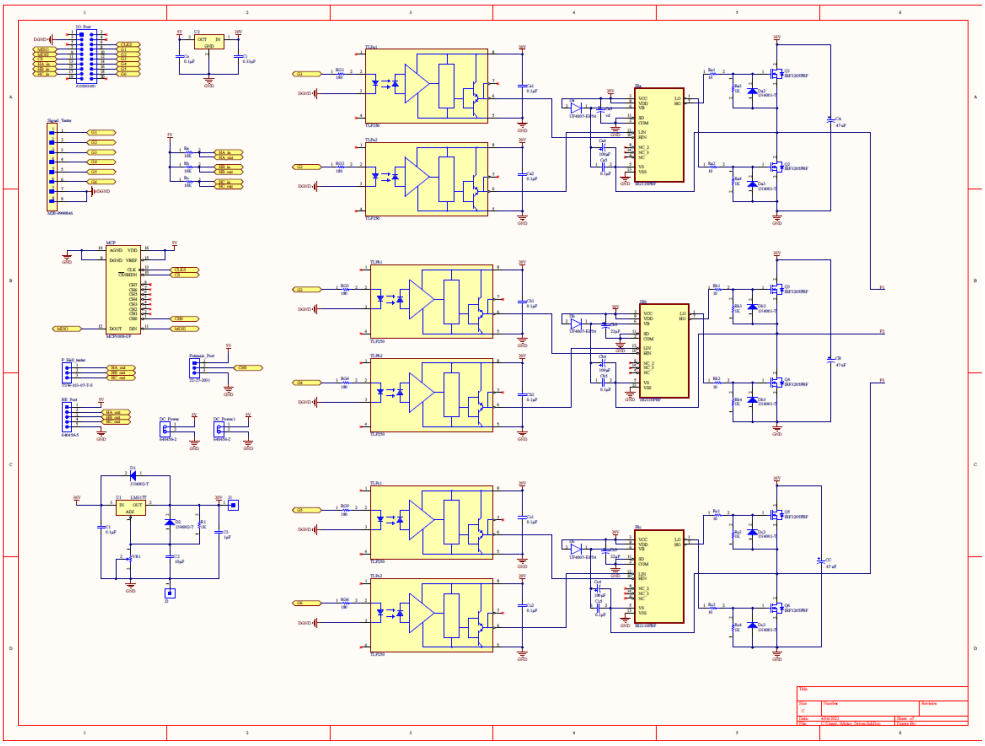
[9] A. Tashakori, M. Hassanudeen and M. Ektesabi, "FPGA Based Controller Drive of BLDC Motor Using Digital PWM Technique," *2015 IEEE 11th International Conference on Power Electronics and Drive Systems*, pp. 658-662, 2015.

[10] E. Viramontes, "BLDC Motor Control with Hall Effect Sensor Using the 9S08MP".

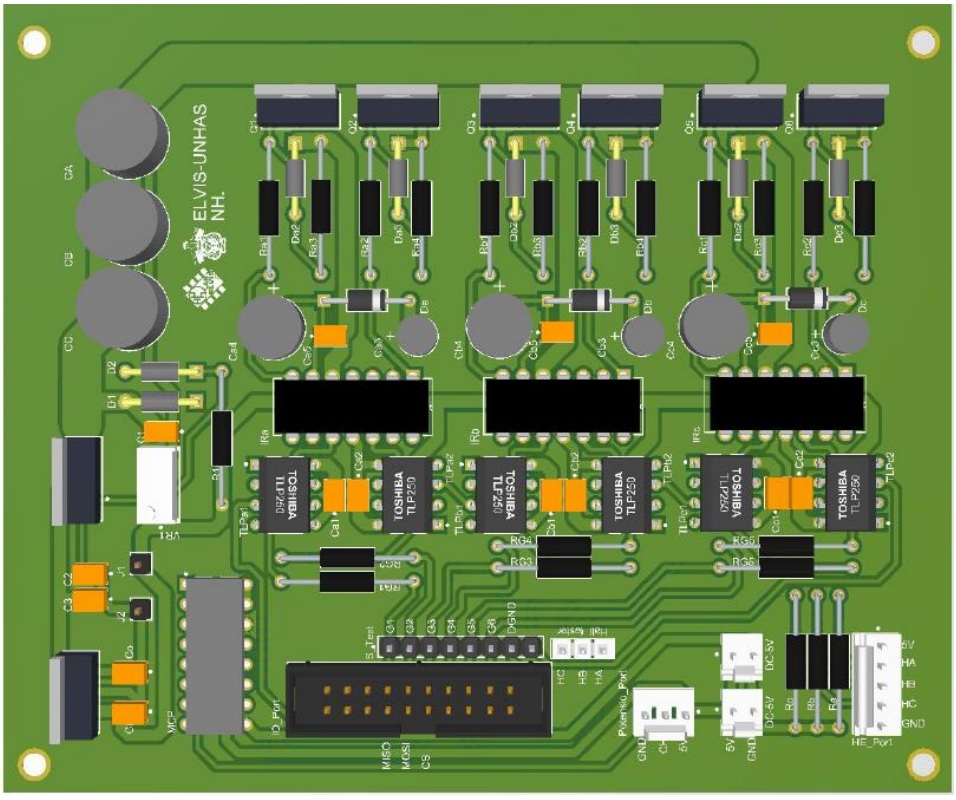
[11] M. F. Sachruddin, "Sistem Kendali Motor BLDC dengan Hall Sensors Berbasis CPLD," Universitas Hasanuddin, Makassar, 2022.

[12] Padmaraja Yedamale, "Brushless DC motor Fundamentals", Appl. Note 885 Microchip, p.8, 2003.

Lampiran 2 Skematik Motor Driver Tiga Fasa



Lampiran 3 Layout PCB Motor Driver Tiga Fasa



Lampiran 4 Kode Verilog Top Level BLDC Controller

```
module DE0_NANO (
   ////////////// CLOCK //////////
   CLOCK_50,

   ////////////// LED //////////
   LED,

   ////////////// KEY //////////
   KEY,

   ////////////// SW //////////
   SW,

   ////////////// SDRAM //////////
   DRAM_ADDR,
   DRAM_BA,
   DRAM_CAS_N,
   DRAM_CKE,
   DRAM_CLK,
   DRAM_CS_N,
   DRAM_DQ,
   DRAM_DQM,
   DRAM_RAS_N,
   DRAM_WE_N,

   ////////////// EPCS //////////
   EPCS_ASDO,
   EPCS_DATA0,
   EPCS_DCLK,
   EPCS_NCSO,

   ////////////// Accelerometer and EEPROM //////////
   G_SENSOR_CS_N,
   G_SENSOR_INT,
   I2C_SCLK,
   I2C_SDAT,

   ////////////// ADC //////////
   ADC_CS_N,
   ADC_SADDR,
   ADC_SCLK,
   ADC_SDAT,

   ////////////// 2x13 GPIO Header //////////
   GPIO_2,


   GPIO_2_IN,

   ////////////// GPIO_0, GPIO_0 connect to GPIO Default //////////
   GPIO_0,
   GPIO_0_IN,

   ////////////// GPIO_0, GPIO_1 connect to GPIO Default //////////
   GPIO_1,
   GPIO_1_IN

);
```

```
//=======================================================
//  PORT declarations
//=======================================================

////////////// CLOCK //////////
input                       CLOCK_50;

////////////// LED //////////
output           [7:0]      LED;

////////////// KEY //////////
input            [1:0]      KEY;

////////////// SW //////////
input            [3:0]      SW;

////////////// SDRAM //////////
output          [12:0]      DRAM_ADDR;
output           [1:0]      DRAM_BA;
output                      DRAM_CAS_N;
output                      DRAM_CKE;
output                      DRAM_CLK;
output                      DRAM_CS_N;
inout           [15:0]      DRAM_DQ;
output           [1:0]      DRAM_DQM;
output                      DRAM_RAS_N;
output                      DRAM_WE_N;

////////////// EPCS //////////
output                      EPCS_ASDO;
input                       EPCS_DATA0;
output                      EPCS_DCLK;
output                      EPCS_NCSO;

////////////// Accelerometer and EEPROM //////////
output                      G_SENSOR_CS_N;
input                       G_SENSOR_INT;
output                      I2C_SCLK;
inout                       I2C_SDAT;

////////////// ADC //////////
output                      ADC_CS_N;
output                      ADC_SADDR;
output                      ADC_SCLK;

input                       ADC_SDAT;

////////////// 2x13 GPIO Header //////////
inout           [12:0]      GPIO_2;
input            [2:0]      GPIO_2_IN;

////////////// GPIO_0, GPIO_0 connect to GPIO Default //////////
inout           [33:0]      GPIO_0;
input            [1:0]      GPIO_0_IN;

////////////// GPIO_0, GPIO_1 connect to GPIO Default //////////
inout           [33:0]      GPIO_1;
input            [1:0]      GPIO_1_IN;
```

```
//=======================================================
//  REG/WIRE declarations
//=======================================================

   //internal signals
   wire clk_sys = CLOCK_50;
   wire rst = 1'b0;

   //hall input signals
   wire hall_a = GPIO_1[0];
   wire hall_b = GPIO_1[1];
   wire hall_c = GPIO_1[3];

   //output signals
   wire pwm_ha;
   wire pwm_hb;
   wire pwm_hc;
   wire pwm_la;
   wire pwm_lb;
   wire pwm_lc;

   //temporary signals
   wire ha;
   wire hb;
   wire hc;
   wire la;
   wire lb;
   wire lc;

   //spi signals
   wire wSPI_CLK;
   wire wSPI_CLK_n;
   wire [7:0] ADC_data;

   //LCD signals
   wire [7:0] LCD_DATA;
   wire       LCD_D_cn;     //RS
   wire       LCD_WEn;      //RW
   wire       LCD_CSn;      //Enable

   //bldc speed measurement signal
   wire [9:0] rpm;

   //nios input data;
   wire [7:0] pwm;

   //filtered hall sensor
   wire hall_f_a;
   wire hall_f_b;
   wire hall_f_c;
```

```
//=======================================================
//  Structural coding
//=======================================================
   //instantiate pattern_pwm module
   pattern_PWM #(.dwidth(8)) U0 (
      .clk(clk_sys),
      .rst(rst),
      .potensio_value(ADC_data),
      .HallA(hall_a),
      .HallB(hall_b),
      .HallC(hall_c),
      .forward(SW[0]),
      .PWM_HA(pwm_ha),
      .PWM_HB(pwm_hb),
      .PWM_HC(pwm_hc),
      .PWM_LA(pwm_la),
      .PWM_LB(pwm_lb),
      .PWM_LC(pwm_lc),
      .HA(ha),
      .HB(hb),
      .HC(hc),
      .LA(la),
      .LB(lb),
      .LC(lc)
   );

   assign GPIO_0[12] = pwm_ha;
   assign GPIO_0[14] = pwm_la;
   assign GPIO_0[16] = pwm_hb;
   assign GPIO_0[18] = pwm_lb;
   assign GPIO_0[20] = pwm_hc;
   assign GPIO_0[22] = pwm_lc;

   assign LED = ADC_data;

   assign GPIO_1[13] = pwm_ha;
   assign GPIO_1[15] = pwm_lc;
   assign GPIO_1[17] = pwm_hb;
   assign GPIO_1[19] = pwm_la;
   assign GPIO_1[21] = pwm_hc;
   assign GPIO_1[23] = pwm_lb;

   assign GPIO_1[18] = hall_a;
   assign GPIO_1[20] = hall_b;
   assign GPIO_1[22] = hall_c;
```

```
    SPIPLL      U1 (
        .inclk0(CLOCK_50),
        .c0(wSPI_CLK),
        .c1(wSPI_CLK_n)
    );

    ADC_CTRL    U2 (
        .iRST(KEY[0]),
        .iCLK(wSPI_CLK),
        .iCLK_n(wSPI_CLK_n),
        .iGO(KEY[1]),
        .iCH(3'b000),
        .oLED(ADC_data),

        .oDIN(ADC_SADDR),
        .oCS_n(ADC_CS_N),
        .oSCLK(ADC_SCLK),
        .iDOUT(ADC_SDAT)
    );

    nios2_proc U3 (
        .clk_clk                                (CLOCK_50),
        .reset_reset_n                          (1'b1),
        .pwm_out_external_connection_export     (pwm),
        .speed_ref_external_connection_export (rpm),
        .lcd_external_connection_export         ({LCD_CSn, LCD_D_cn, LCD_WEn, LCD_DATA}),
        .adc_data_external_connection_export    (ADC_data)
    );

    assign GPIO_0[0] = LCD_DATA[7];
    assign GPIO_0[1] = LCD_DATA[6];
    assign GPIO_0[3] = LCD_DATA[5];
    assign GPIO_0[5] = LCD_DATA[4];
    assign GPIO_0[7] = LCD_DATA[3];
    assign GPIO_0[9] = LCD_DATA[2];
    assign GPIO_0[11] = LCD_DATA[1];
    assign GPIO_0[13] = LCD_DATA[0];

    assign GPIO_0[15] = LCD_CSn;
    assign GPIO_0[17] = LCD_WEn;
    assign GPIO_0[19] = LCD_D_cn;

    // hall filter
    edge_detection_wrapper U4 (
        .clock(CLOCK_50),
        .reset(rst),
        .hall_a(hall_a),
        .hall_b(hall_b),
        .hall_c(hall_c),
        .hall_f_a(hall_f_a),
        .hall_f_b(hall_f_b),
        .hall_f_c(hall_f_c)
    );

    //speed calculation of 3 hall sensors
    speed_calculation_wrapper U5 (
        .clock(CLOCK_50),
        .reset(rst),
        .hall_f_a(hall_f_a),
        .hall_f_b(hall_f_b),
        .hall_f_c(hall_f_c),
        .rpm(rpm)
    );
endmodule
```

Lampiran 5 Kode Verilog PWM Generator

```
`timescale 1ns/1ps

module pwm_generator(clk, rst, enable, potensio_value, PWM);
   parameter n = 10;
   input clk, rst, enable;
   input [n-1:0] potensio_value;
   output wire PWM;
   reg PWM_out;

   wire [n-1:0] counter_out;

   always @ (posedge clk)
      if(potensio_value > counter_out)
         PWM_out <= 1'b1;
      else PWM_out <= 1'b0;
   assign PWM = enable ? PWM_out : 1'b0;

   //instantiate counter module
   counter_10bits #(.k(n)) U0 (
      .clk(clk),
      .rst(rst),
      .count_out(counter_out)
   );

endmodule

module counter_10bits(clk, rst, count_out);
   parameter k = 10;
   input clk, rst;
   output reg [k-1:0] count_out;

   always @ (posedge clk, posedge rst)
   begin
      if(rst)
         count_out <= 0;
      else
         count_out <= count_out + {{(k-1){1'b0}}, 1'b1};
   end
endmodule
```

Lampiran 6 Kode Verilog Komutasi Sensor Hall Efek

```
module fw_rv_commutation (
   clock,
   reset,
   forward,
   halla,
   hallb,
   hallc,
   ha,
   hb,
   hc,
   la,
   lb,
   lc
);
   parameter A = 3'b000,
             B = 3'b001,
             C = 3'b010,
             D = 3'b011,
             E = 3'b100,
             F = 3'b101;

   input clock;
   input reset;
   input forward;
   input halla;
   input hallb;
   input hallc;

   output ha;
   output hb;
   output hc;
   output la;
   output lb;
   output lc;

   reg [2:0] state_D, state_Q;

   wire [2:0] hall_sensor;

   assign hall_sensor = {halla, hallb, hallc};

   //define the next state combinatorial circuit
   always @ (hall_sensor, forward, state_Q)
      case (state_Q)
         A: if(hall_sensor == 3'b101) state_D = A;
            else if(hall_sensor == 3'b100) state_D = B;
            else if(hall_sensor == 3'b001) state_D = F;
            else state_D = A;
         B: if(hall_sensor == 3'b101) state_D = A;
            else if(hall_sensor == 3'b100) state_D = B;
            else if(hall_sensor == 3'b110) state_D = C;
            else state_D = B;
         C: if(hall_sensor == 3'b100) state_D = B;
            else if(hall_sensor == 3'b110) state_D = C;
            else if(hall_sensor == 3'b010) state_D = D;
            else state_D = C;
         D: if(hall_sensor == 3'b110) state_D = C;
            else if(hall_sensor == 3'b010) state_D = D;
            else if(hall_sensor == 3'b011) state_D = E;
            else state_D = D;
         E: if(hall_sensor == 3'b010) state_D = D;
            else if(hall_sensor == 3'b011) state_D = E;
            else if(hall_sensor == 3'b001) state_D = F;
            else state_D = E;
         F: if(hall_sensor == 3'b101) state_D = A;
            else if(hall_sensor == 3'b011) state_D = E;
            else if(hall_sensor == 3'b001) state_D = F;
            else state_D = F;
         default: ;
      endcase
```

```
    //define the sequential block
    always @ (posedge clock)
    begin
       if (reset)
          state_Q <= A;
       else
          state_Q <= state_D;
    end

    //define output
    assign ha = (forward == 1'b1) ? ((state_Q == D) || (state_Q == E))
                                  : ((state_Q == A) || (state_Q == B));
    assign hb = (forward == 1'b1) ? ((state_Q == A) || (state_Q == F))
                                  : ((state_Q == C) || (state_Q == D));
    assign hc = (forward == 1'b1) ? ((state_Q == B) || (state_Q == C))
                                  : ((state_Q == E) || (state_Q == F));
    assign la = (forward == 1'b1) ? ((state_Q == A) || (state_Q == B))
                                  : ((state_Q == D) || (state_Q == E));
    assign lb = (forward == 1'b1) ? ((state_Q == C) || (state_Q == D))
                                  : ((state_Q == A) || (state_Q == F));
    assign lc = (forward == 1'b1) ? ((state_Q == E) || (state_Q == F))
                                  : ((state_Q == B) || (state_Q == C));

endmodule
```

Lampiran 7 Kode Verilog ADC Control

```
module ADC_CTRL   (
                iRST,
                iCLK,
                iCLK_n,
                iGO,
                iCH,
                oLED,

                oDIN,
                oCS_n,
                oSCLK,
                iDOUT
            );

input           iRST;
input           iCLK;
input           iCLK_n;
input           iGO;
input [2:0]     iCH;
output   [7:0]     oLED;

output             oDIN;
output             oCS_n;
output             oSCLK;
input           iDOUT;

reg                data;
reg                go_en;
wire  [2:0]     ch_sel;
reg                sclk;
reg      [3:0]     cont;
reg      [3:0]     m_cont;
reg      [11:0]       adc_data;
reg      [7:0]     led;

assign   oCS_n     =  ~go_en;
assign   oSCLK     =  (go_en)? iCLK:1'h1;
assign   oDIN      =  data;
assign   ch_sel       =  iCH;
assign   oLED      =  led;

always@(posedge iGO or negedge iRST)
begin
   if(!iRST)
      go_en <= 0;
   else
   begin
      if(iGO)
         go_en <= 1;
   end
end

always@(posedge iCLK or negedge go_en)
begin
   if(!go_en)
      cont  <= 0;
   else
   begin
      if(iCLK)
         cont  <= cont + 4'h1;
   end
end
```

```
always@(posedge iCLK_n)
begin
   if(iCLK_n)
      m_cont   <= cont;
end

always@(posedge iCLK_n or negedge go_en)
begin
   if(!go_en)
      data  <= 0;
   else
   begin
      if(iCLK_n)
      begin
         if (cont == 2)
            data  <= iCH[2];
         else if (cont == 3)
            data  <= iCH[1];
         else if (cont == 4)
            data  <= iCH[0];
         else
            data  <= 0;
      end
   end
end

always@(posedge iCLK or negedge go_en)
begin
   if(!go_en)
   begin
      adc_data <= 0;
      led         <= 8'h00;
   end
   else
   begin
      if(iCLK)
      begin
         if (m_cont == 4)
            adc_data[11]   <= iDOUT;
         else if (m_cont == 5)
            adc_data[10]   <= iDOUT;
         else if (m_cont == 6)
            adc_data[9]    <= iDOUT;
         else if (m_cont == 7)
            adc_data[8]    <= iDOUT;
         else if (m_cont == 8)
            adc_data[7]    <= iDOUT;
         else if (m_cont == 9)
            adc_data[6]    <= iDOUT;
         else if (m_cont == 10)
            adc_data[5]    <= iDOUT;
         else if (m_cont == 11)
            adc_data[4]    <= iDOUT;
         else if (m_cont == 12)
            adc_data[3]    <= iDOUT;
         else if (m_cont == 13)
            adc_data[2]    <= iDOUT;
         else if (m_cont == 14)
            adc_data[1]    <= iDOUT;
         else if (m_cont == 15)
            adc_data[0]    <= iDOUT;
         else if (m_cont == 1)
            led   <= adc_data[11:4];
      end
   end
end
endmodule
```

Lampiran 8 Kode Verilog Edge Detection

```
module edge_detection
#(parameter n = 20, parameter div = 4)
(
   clock,
   reset,
   data,
   z
);

   input clock;
   input reset;
   input data;
   output z;

   localparam A = 2'b01, B = 2'b10;

   wire [n-1:0] Q;
   wire [1:0] w;
   wire clk_div;
   reg [1:0] state_D, state_Q;

   shiftn #(.k(n)) U0 (
      .clock(clk_div),
      .reset(reset),
      .D(data),
      .Q(Q)
   );

   clk_div #(.n(div)) U1(
      .clk_sys(clock),
      .reset(reset),
      .clock_div(clk_div)
   );

   assign w[1] = &Q;
   assign w[0] = |Q;

   //Define next state
   always @ (w, state_Q)
      case (state_Q)
         A : if(w == 2'b11) state_D = B;
             else           state_D = A;
         B : if(w == 2'b00) state_D = A;
             else           state_D = B;
         default: ;

      endcase


   //Define the sequential block
   always @ (posedge clk_div)
      if(reset)
         state_Q <= A;
      else
         state_Q <= state_D;

   //define output
   assign z = (state_Q == B) ? 1'b1 : 1'b0;

endmodule
```

```
module shiftn
#(parameter k = 4)
(
   clock,
   reset,
   D,
   Q
);

   input clock;
   input reset;
   input D;
   output reg [k-1:0] Q;

   integer i;
   always @ (posedge clock)
   begin
      if(reset)
         Q <= 0;
      else
      begin
         for(i = 0; i < k - 1; i = i + 1)
            Q[i] <= Q[i+1];
         Q[k-1] <= D;
      end
   end

endmodule
```

Lampiran 9 Kode Verilog Speed Calculation

```
module speed_calculation
#(parameter MAX_VALUE = 49_999_999)
(
   clk,
   rst,
   hall_sensor,
   revolution
);

   input clk;
   input rst;
   input hall_sensor;

   output [7:0] revolution;

   //wire enable;

   //assign enable = (HA & ~LA & ~HB & LB & ~LC);

   reg [25:0] count;
   wire max_count = (count == MAX_VALUE); //max_count = 49,999,999
   always @ (posedge clk)
      if(rst)
         count <= 26'h0;
      else if(max_count)
         count <= 26'h0;
      else
         count <= count + 26'h1;

   //counter mechanical cycles
   reg [7:0] mec_cycles = 8'h0;
   wire min_count = (count == 26'h0);
   always @ (posedge hall_sensor, posedge min_count)
      if(min_count)
         mec_cycles <= 8'h0;
      else
         mec_cycles <= mec_cycles + 8'h1;

   reg [7:0] D;
   always @ (posedge clk)
      if(rst)
         D <= 8'h0;
      else if(max_count)
         D <= mec_cycles;

   assign revolution = D;
endmodule
```