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.

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

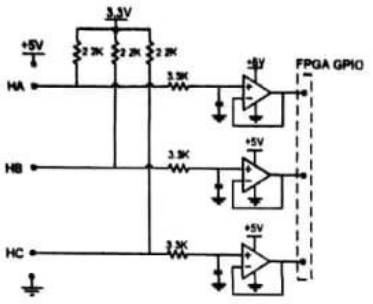
Berarti dalam setegah putaran mekanik sama dengan 360° putaran elektrik dan pada 1 putaran mekanik penuh sama dengan 720° putaran elektrik. Jika hasil dari kecepatan sudut dikonversi ke dalam satuan frekuensi maka digunakan persamaan 3, hasil dari persamaan ini akan digunakan untuk menghitung kecepatan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

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

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

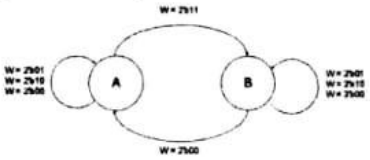
Telah dijelaskan sebelumnya bahwa komutasi motor BLDC dikendalikan secara elektronik. Agar dapat berputar,

B. *Filter Sensor Hall Efek*



Gambar 4. Rangkaian *Active low pass filter*

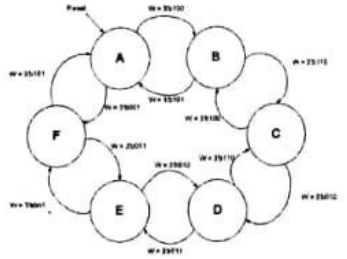
Filter sensor hall efek menggunakan filter analog dan digital. Pertama filter analog, rangkaian filter ini menggunakan pull-up resistor dengan resistor bernilai 2.2K ohm yang terhubung dengan sumber 3.3V dan *active low fass filter* dengan menggunakan resistor 3.3K ohm yang parallel dengan kapacitor 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.

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

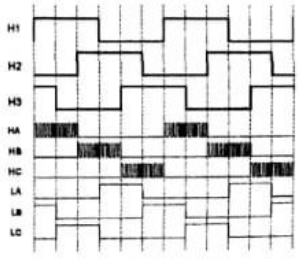
TABEL I. POLA KOMUTASI SEARAH JARUM JAM

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+

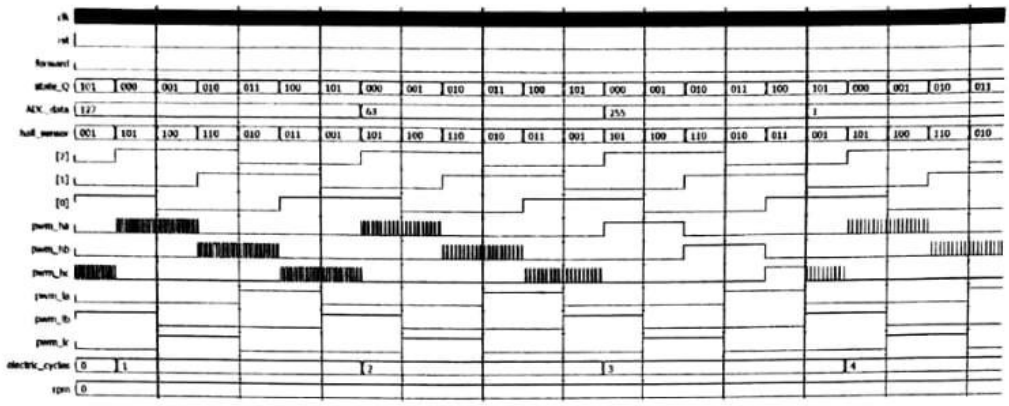
TABEL II. POLA KOMUTASI BERLAWANAN ARAH JARUM JAM

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

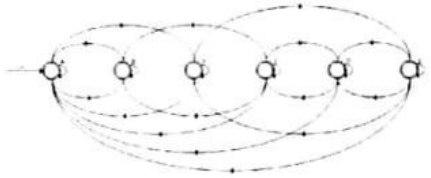
$$RPM = \frac{60}{pairs\ of\ poles} x\ f \qquad (5)$$

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.

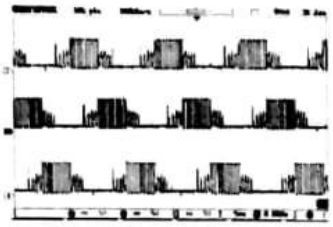
Algoritma 6-step komutasi dirancang berdasarkan state mechine. Diagram *state mechine* ditunjukkan pada gambar 9, terdiri dari 6 keadaan yaitu keadaan A, B, C, D, E, dan F. Masing-masing keadaan direpresentasikan dalam tiga bit parameter secara berurutan yaitu "000", "001", "010", "011", "100", "101". 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.



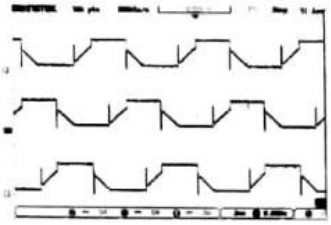
Gambar 9. Diagram state machine modul komutasi

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)



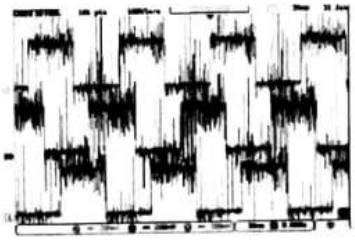
(b)

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

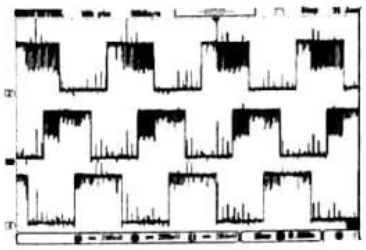
IV. HASIL IMPLEMENTASI

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)

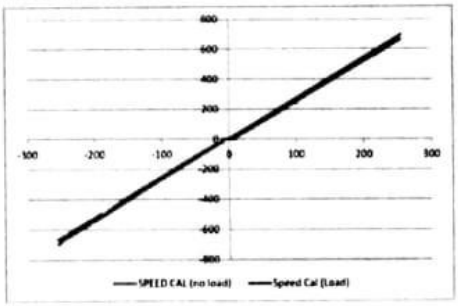


(b)

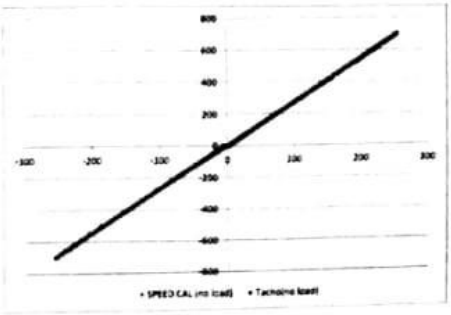
Gambar 11. (a) hall sensor tanpa filter, (b) hall sensor dengan active low pass filter

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.



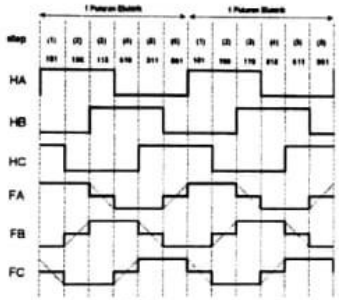
Gambar 12. Grafik kecepatan tanpa beban dan berbeban dengan modul SC



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].



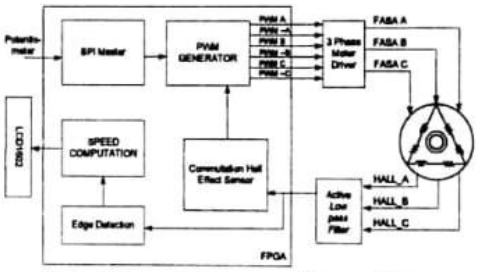
Gambar 1. Bentuk gelombang kotak yang dihasilkan oleh *trapezoid control*

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

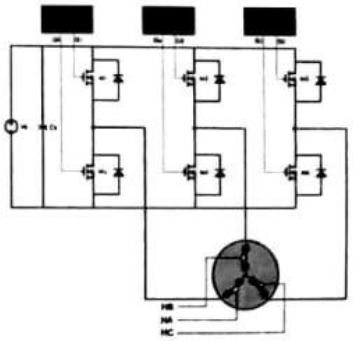
Sistem kendali motor BLDC yang dirancang terbagi menjadi 2 bagian yaitu Sistem perangkat keras (*Hardware*) dan Perangkat lunak (*Software*). Bagian perangkat keras berupa Driver Tiga Fasa yang terdiri atas mosfet, bus kapasitor, driver mosfet, filter hall sensor, dan pembagi tegangan. Filter hall sensor 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.



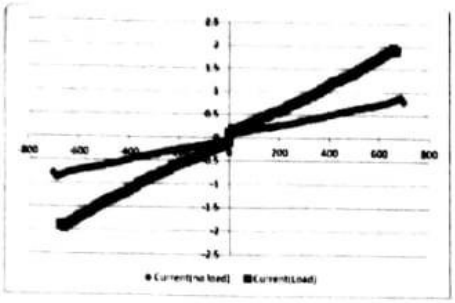
Gambar 2. Diagram blok sistem pengendalian motor BLDC

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.

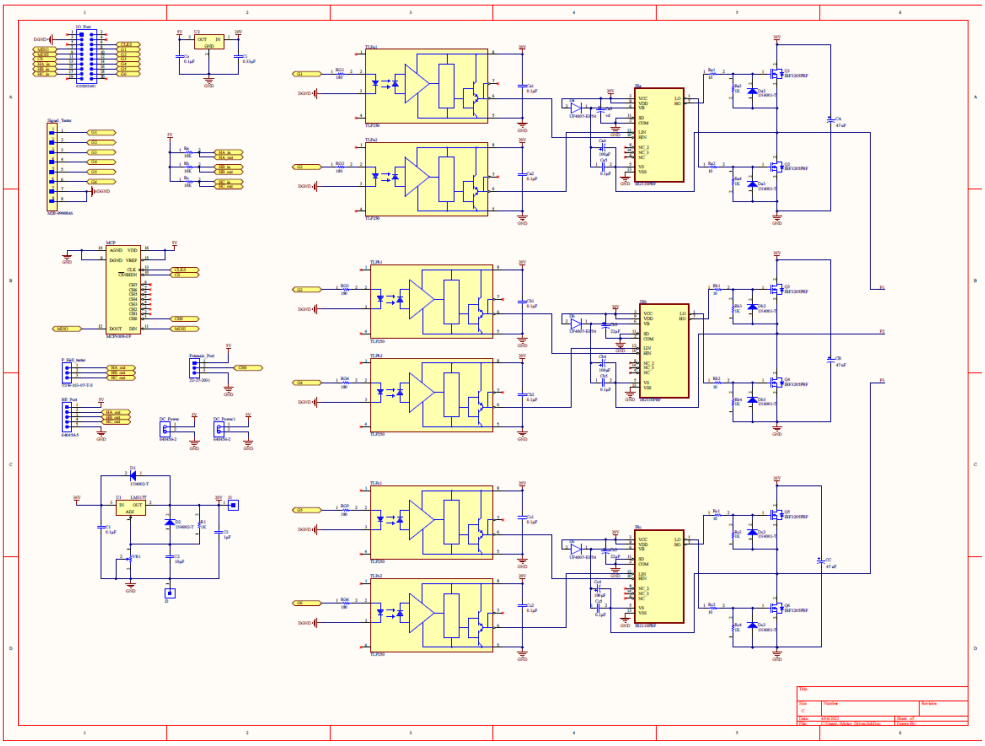
[9] A. Tashakori, M. Hassanudeen and M. Ektesabi, "FPGA Based Controller Drive of BLDC Motor Using Digital PWM Technique," *2015 IEEE 11th International Conference on Power Electronics and Drive Systems*, pp. 658-662, 2015.

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

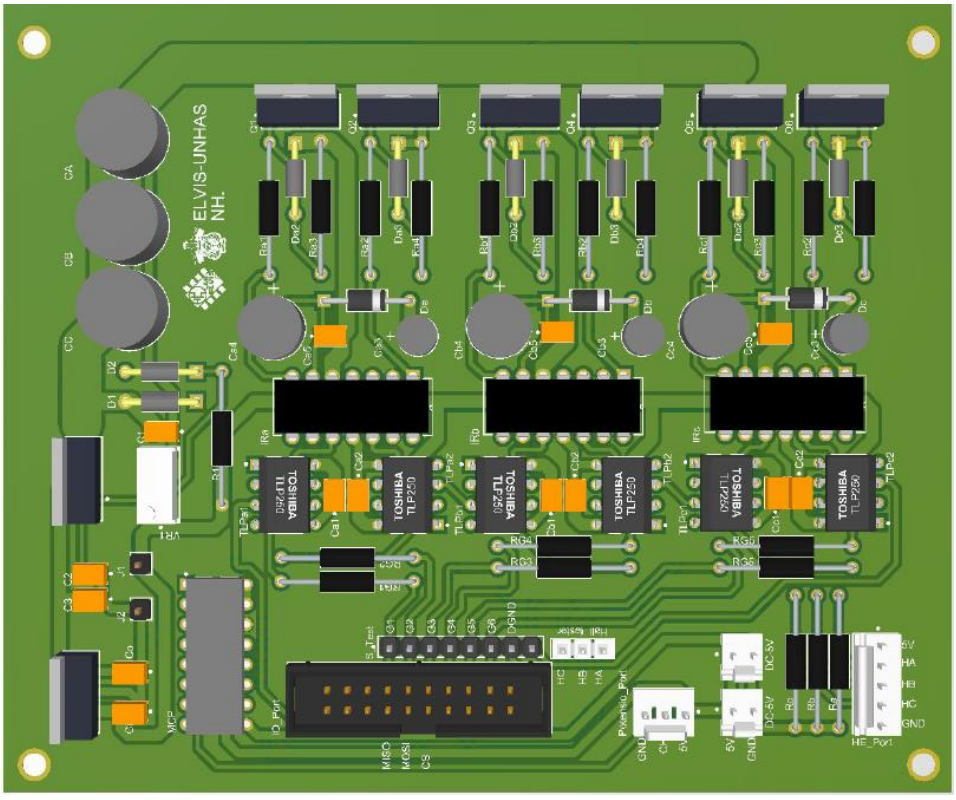
[11] M. F. Sachruddin, "Sistem Kendali Motor BLDC dengan Hall Sensors Berbasis CPLD," Universitas Hasanuddin, Makassar, 2022.

[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
```

Lampiran 10 SK Dosen Pembimbing

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
Poros Malino Km.6Bontomarannu(92172) Gowa, Sulawesi Selatan 92172, Sulawesi Selatan
Telp. (0411) 586015, 586262 Fax (0411) 586015
http://eng.unhas.ac.id, Email : teknik@unhas.ac.id

SURAT PENUGASAN
No. 9238/UN4.7.1/PT.01.06/2021

Dari : Dekan Fakultas Teknik Universitas Hasanuddin

Kepada : 1. Prof.Dr.-Ing. Faizal Arya Samman.,ST,MT — Pemb. I
2. Dr.Ir.H.Rhiza S.Sadjad.,MSEE — Pemb.II

I s i : 1. Berdasarkan Surat Ketua Departemen Teknik Elektro Fakultas Teknik Nomor. 6129/UN4.7.7.1/PT.01.06/2021 tanggal 31 Maret 2021 tentang usul DOSEN PEMBIMBING MAHASISWA, maka dengan ini kami menugaskan Saudara untuk membimbing penulisan Skripsi/Tugas Akhir mahasiswa DepartemenTeknik Elektro Fakultas Teknik Universitas Hasanuddin di bawah ini :

N a m a : Nurul Hidayat
No. Stambuk : D041 17 1319

Judul Skripsi/Tugas Akhir:
"Perancangan Sistem Digital Berbasis FPGA untuk Pengendalian Motor listrik"

2. Surat penugasan pembimbing ini mulai berlaku sejak tanggal ditetapkannya dan berakhir sampai selesainya penulisan Skripsi/Tugas Akhir mahasiswa tersebut.

3. Agar penugasan ini dilaksanakan sebaik-baiknya dengan penuh rasa tanggung jawab.

Ditetapkan di Gowa
Pada tanggal 9 Juni 2021
a.n. Dekan,
Wakil Dekan Bidang Akademik, Riset dan Inovasi

Prof. Baharuddin Hamzah, S.T.,M.Arch.,Ph.D
NIP. 19690308 199512 1 001

Tembusan :
1. Dekan FT-UH,
2. Ketua Departemen Teknik Elektro FT-UH,
3. Mahasiswa yang bersangkutan

Lampiran 11 SK Ujian Sarjan

KEMENTERIAN PENDIDIKAN, KEBUDAYAN, RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
Jl. Poros Malino Km. 6 Gowa, 92171, Sulawesi Selatan
☎ (0411) 586015, 586262 Fax (0411) 586015
http://eng.unhas.ac.id., Email : teknik@unhas.ac.id

SURAT PENUGASAN

No. 8343/UN4.7.1/PT.01.06/2022

Dari : Dekan Fakultas Teknik Universitas Hasanuddin.

Kepada : Mereka yang tercantum namanya di bawah ini.

Isi : 1. Bahwa berdasarkan Peraturan Rektor Universitas Hasanuddin Nomor: 2781/UN4.1/KEP/2018, dengan ini menugaskan Saudara sebagai PANITIA UJIAN SARJANA Strata Satu (S1) Departemen Teknik Elektro Fakultas Teknik Universitas Hasanuddin dengan susunan sebagai Berikut :

Pembimbing I / Ketua	: 1. Prof. Dr.-Ing. Faizal Arya Samman.,ST,MT
Pembimbing II / Sekretaris	: 2. Dr. Ir. H. Rhiza S.Sadjad.,MSEE
Anggota	: 1. Dr. Hj. A. Ejah Umraeni Salam.,ST,MT
	2. Muh Anshar.,ST,MSc,Ph.D

untuk menguji bagi mahasiswa tersebut di bawah ini :

Nama/NIM	: Nurul Hidayat/D041171319
Departemen	: Teknik Elektro
Judul Thesis/Skripsi	: **"Sistem Kendali Digital Berbasis FPGA untuk Mengendalikan Motor Arus Searah Tanpa Sikat"**

2. Waktu seminar ditetapkan oleh Panitia Ujian Sarjana Strata Satu (S1).
3. Agar Surat penugasan ini dilaksanakan sebaik-baiknya dengan penuh rasa tanggung jawab.
4. Surat penugasan ini berlaku sejak tanggal ditetapkan sampai dengan berakhirnya Seminar tersebut dengan ketentuan bahwa segala sesuatunya akan ditinjau dan diperbaiki sebagaimana mestinya apabila dikemudian hari ternyata terdapat kekeliruan dalam keputusan ini.

Ditetapkan di Gowa,
Pada tanggal 27 April 2022
a.n. Dekan.

Wakil Dekan Bidang Akademik, Riset dan Inovasi

Prof. Baharuddin Hamzah, ST., M.Arch., Ph.D
NIP. 19690308 199512 1 001

Tembusan :
1. Dekan Fak. Teknik Unhas
2. Ketua Departemen Teknik Elektro FT-UH
3. Mahasiswa yang bersangkutan

Lampiran 12 Berita Acara Ujian Sarjana

KEMENTERIAN PENDIDIKAN, KEBUDAYAN, RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
DEPARTEMEN ELEKTRO
Jl. Poros Malino Km. 6 Gowa, 92171, Sulawesi Selatan
☎ (0411) 586015, 586262 Fax (0411) 586015
http://eng.unhas.ac.id/elektro, Email : elektro@unhas.ac.id

BERITA ACARA UJIAN SARJANA

Pada hari ini **Rabu** tanggal **11 Mei 2022** Pukul **14.00 WITA - Selesai** bertempat di Meeting Room Elvis dan **(Daring On-line Zoom),** telah dilaksanakan ujian sarjana bagi Saudara :

Nama : Nurul Hidayat

No. Stambuk : D041171319

Program Studi : Teknik Elektro

Judul Skripsi/TA : **"Sistem Kendali Digital Berbasis FPGA untuk Mengendalikan Motor Arus Searah Tanpa Sikat"**

Yang dihadiri oleh Tim Penguji Ujian Sarjana sebagai berikut :

No.	N a m a	Jabatan	Tanda tangan
1.	Prof. Dr.-Ing. Faizal Arya Samman.,ST,MT	Pemb. I / Ketua	1.
2.	Dr. Ir. H. Rhiza S.Sadjad.,MSEE	Pemb. II / Sekretaris	2.
3.	Dr. Hj. A. Ejah Umraeni Salam.,ST,MT	Anggota	3.
4.	Muh Anshar.,ST,MSc,Ph.D	Anggota	4.

Hasil keputusan panitia penilai ujian sarjana : **Lulus / Tidak lulus** dengan nilai angka 89 dan huruf A

Gowa, 11 Mei 2022

Ketua/Sekretaris Panitia Ujian Sarjana

Prof. Dr.-Ing. Faizal Arya Samman.,ST,MT