# DAFTAR PUSTAKA

[1] J. de Santiago *et al.*, “Electrical Motor Drivelines in Commercial All-Electric Vehicles: A Review,” *IEEE Trans. Veh. Technol.*, vol. 61, no. 2, pp. 475–484, Feb. 2012, doi: 10.1109/TVT.2011.2177873.

[2] A. Hughes, *Electric Motor and Drives*. Elsevier Ltd., 2006.

[3] G. Tao, Z. Ma, L. Zhou, and L. Li, “A novel driving and control system for direct-wheel-driven electric vehicle,” *2004 12th Symp. Electromagn. Launch Technol.*, vol. 41, no. 1, pp. 514–517, 2004, doi: 10.1109/elt.2004.1398134.

[4] M. Jafarboland and M. H. R. Silabi, “New sensorless commutation method for BLDC motors based on the line-to-line flux linkage theory,” *IET Electr. Power Appl.*, vol. 13, no. 6, pp. 703–711, 2019, doi: 10.1049/iet-epa.2018.5356.

[5] M. Baszynski and S. Pirog, “A novel speed measurement method for a high-speed BLDC motor based on the signals from the rotor position sensor,” *IEEE Trans. Ind. Informatics*, vol. 10, no. 1, pp. 84–91, 2014, doi: 10.1109/TII.2013.2243740.

[6] Hong-xing Wu, Shu-kang Cheng, and Shu-mei Cui, “A controller of brushless DC motor for electric vehicle,” *IEEE Trans. Magn.*, vol. 41, no. 1, pp. 509–513, Jan. 2005, doi: 10.1109/TMAG.2004.839304.

[7] C.-L. Xia, *Permanent Magnet Brushless DC Motor Drives and Controls*. John Wiley & Sons Singapore Pte. Ltd., 2012.

[8] S. J. Chapman, *Electric Machinery Fundamentals*, Fourth Edi. McGraw-Hill, 2005.

[9] Tae-Hyung Kim and M. Ehsani, "Sensorless control of the BLDC motors from near-zero to high speeds," *IEEE Trans. Power Electron.*, vol. 19, no. 6, pp. 1635–1645, Nov. 2004, doi: 10.1109/TPEL.2004.836625.

[10] K. Kolano, "Improved sensor control method for BLDC motors," *IEEE Access*, vol. 7, pp. 186158–186166, 2019, doi: 10.1109/ACCESS.2019.2960580.

[11] B. P. Reddy and A. Murali, "SoC FPGA-based field oriented control of BLDC motor using low resolution Hall sensor," *IECON Proc. (Industrial Electron. Conf.*, pp. 2941–2945, 2016, doi: 10.1109/IECON.2016.7793092.

[12] R. Krishnan, *Electric Motor Drives - Modeling Analysis and Control*. Prentice Hall, Inc, 2001.

[13] M. Jafarboland and M. H. R. Silabi, "New sensorless commutation method for BLDC motors based on the line-to-line flux linkage theory," *IET Electr. Power Appl.*, vol. 13, no. 6, pp. 757–765, 2019, doi: 10.1049/iet-epa.2018.5356.

[14] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics : Converters, Applications, and Design*, Second Edi. John Wiley & Sons, Inc, 1995.

[15] M. H. Rashid, *The power electronics handbook*, Third edit. Elsevier Inc, 2011.

[16] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*. McGraw-Hill, 2003.

[17] C. Maxfield, *The Design Warrior's Guide to FPGAs*. Elsevier Inc, 2004.

[18] M. Poorani and R. Kurunjimalar, "Design implementation of UART and SPI in single FGPA," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, Jan. 2016, pp. 1–5, doi: 10.1109/ISCO.2016.7726983.

[19] R. M. Pindoriya, A. K. Mishra, B. S. Rajpurohit, and R. Kumar, "FPGA Based Digital Control Technique for BLDC Motor Drive," *IEEE Power Energy Soc. Gen. Meet.*, vol. 2018-Augus, pp. 1–5, 2018, doi: 10.1109/PESGM.2018.8586472.

[20] M. T. Inc., "Mcp3004/3008." pp. 1–35, 2002.

[21] G. Mihalache and A. Ioan, "FPGA Implementation of BLDC Motor Driver with Hall Sensor Feedback," in *2018 International Conference and Exposition on Electrical And Power Engineering (EPE)*, Oct. 2018, pp. 624–629, doi: 10.1109/ICEPE.2018.8559886.

[22] R. Mancini and J. Karki, *Op Amps For Everyone*, no. August. Texas Instruments, 2002.

# LAMPIRAN

Lampiran 1 Paper Seminar Hasil

# BLDC Motor Control using a Complex Programmable Logic Device with Hall-Sensors

Muhammad Fajri Sachruddin
*Department of Electrical Engineering*
*Universitas Hasanuddin*
Makassar, Indonesia
sachruddinmf16d@student.unhas.ac.id

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

Rhiza S. Sadjad
*Department of Electrical Eningeering*
*Universitas Hasanuddin*
Makassar, Indonesia
rhiza@unhas.ac.id

***Abstract*—This paper presents the design and implementation of speed control for a brushless direct current (BLDC) motor using a complex programmable logic device (CPLD). Implementation of speed control is using a PWM technique by varying duty cycles applied to a three-phase inverter. Rotor position determines by hall sensors which are used as references to synchronize the PWM control signals. The control model is written using Verilog Hardware Description Language (HDL) and verified by simulation using ModelSim-Altera. An experimental setup is built to test the performance of the BLDC Motor under the PWM control. The control algorithm is implemented using Max II EPM240T100C5 devices on a 350W 36V rated BLDC motor. The number of the used logic elements (LEs) of the CPLD is about 133 of 240 LEs. PWM with controllable duty cycle generated in this system for having working frequency about 20KHz and commutate sequentially according to six-step commutation up to 600 RPM on no-load condition.**

***Keywords—CPLD, PWM, BLDC, Hall Sensor, ADC***

## I. Introduction

The development and usage of electric vehicles (EVs) are becoming more popular and replacing fossil energy [1]. There are several obstacles to implement EV, one of it is a complicated control system [2], and therefore many publications and implementation encounter this problem. An electric motor is one important part of the electric vehicle, and many various motor types are currently on the market.

Brushless DC motor is widely used on many applications like computers, military, automotive, and many more [3]. In EV, this motor has become a good option compared to other motors. Several advantages of BLDC motor are high efficiency, convenient maintenance, and high torque density [2][4][5][6]. All these basic characteristics are required for traction application, such as in EV [7]. However, an electronic commutator is required for the BLDC motor instead of a mechanical commutator like in a DC motor.

In order to commutate BLDC motors, it is required to identify rotor position [4]. There are two ways to get the rotor position, i.e., using sensor and sensorless methods. Sensor method typically uses hall sensor to detect electromagnetic flux from a permanent magnet. The sensorless method detects it using zero-crossing back-EMF. Both methods have advantages and disadvantages. Hall sensors are much easier to use, but production costs can be higher and have low resolution. However, there are several methods to overcome this problem [8][9]. The sensorless method can be universally used on various BLDC motors, but it is rather complex to implement the algorithm [4][10].

The control system will be implemented in a complex programmable logic device (CPLD). Compared to microcontrollers, it offers several advantages due to flexibility and low power consumption [9] [11]. A FPGA and CPLD has a higher processing speed and can perform complex computations and parallel processes [11] [12]. Furthermore, CPLD and FPGA are programmed using a hardware description language and potentially implemented in system-on-chip devices for mass production. FPGA and CPLD also can easily simulate the design before being implemented on hardware to reduce error during implementation.

This paper presents a BLDC control system based on CPLD as the main controller to generate PWM signals. The commutation sequence is determined by using built-in hall-effect sensor to detect permanent magnet rotor position. Compared to other works [9] [12] [13][13], the implementation of CPLD offers more benefits than microcontroller or FPGA. CPLD process speed is better than microcontrollers for such parallel processing. With the minimal algorithm in this design, the use case of FPGA will overrate resources. Furthermore, since CPLD is a non-volatile device, there is no need for programming and configuring when initial power-up of the system frequently occurs.

The remaining chapter of this paper is organized as follows. Chapter II describes in detail about control algorithm using for this BLDC motor drive system. Chapter III explains simulation results, and Chapter IV is implementation and verification of BLDC control system through experimental testing to BLDC motor.

## II. BLDC Motor Control System

The construction of a brushless DC motor is similar to an induction motor and a DC motor without brushes. It can have various configurations on total phase or winding arrangement, either star or delta pattern. In this paper, we will focus on control the three-phase BLDC motor.

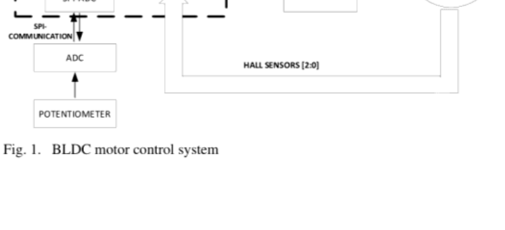


Fig. 1. BLDC motor control system

Commutation of BLDC motor is controlled electronically, based on rotor-stator position. The speed of BLDC motor is proportional to the applied voltage, and therefore we are required to control voltage on BLDC motor [14]. Pulse width modulation technique will be used to control voltage by varying the duty cycle of PWM signals.

The overall system of this motor control system is presented in Figure 1. There are two inputs in our system, the signal from hall sensors and potentiometer. The built-in hall sensors are used to determine rotor position of the motor and potentiometer to adjust speed of BLDC motor. Logic part will be controlled using complex programmable logic device (CPLD) as the brain of our system.

## *A. Phase Sequence Based on Hall Sensor*

Figure 2 shows a three-phase inverter using six MOSFETs as switch with PWM signal to gate of MOSFET. This will be VSI to drive BLDC motor and using two-phase conduction method [15].

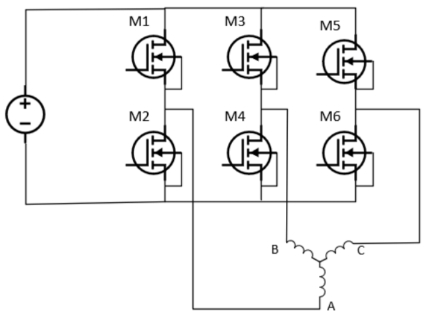


Fig. 2. 3-Phase Inverter Schematic

The commutation of BLDC motor is determined by hall sensor pattern and called six-step commutation. Relation between active phase and rotor position can be seen on manufacturer datasheet or manually determined [13]. Table I and Table II shows sequence for six-step commutation in both direction clockwise and counterclockwise direction. Every MOSFET is commutate every 60 electrical degrees.

TABLE I. CLOCKWISE SIX-STEP COMMUTATION

| **Hall Sensors** | | | **Motor Phases** | | |
|---|---|---|---|---|---|
| ***A*** | ***B*** | ***C*** | ***Phase A*** | ***Phase B*** | ***Phase C*** |
| 0 | 0 | 1 | Off | - | + |
| 0 | 1 | 1 | - | Off | + |
| 0 | 1 | 0 | - | + | Off |
| 1 | 1 | 0 | Off | + | - |
| 1 | 0 | 0 | + | Off | - |
| 1 | 0 | 1 | + | - | Off |

TABLE II. COUNTERCLOCKWISE SIX-STEP COMMUTATION

| **Hall Sensors** | | | **Motor Phases** | | |
|---|---|---|---|---|---|
| ***A*** | ***B*** | ***C*** | ***Phase A*** | ***Phase B*** | ***Phase C*** |
| 0 | 0 | 1 | Off | + | - |
| 0 | 1 | 1 | + | Off | - |
| 0 | 1 | 0 | + | - | Off |
| 1 | 1 | 0 | Off | - | + |
| 1 | 0 | 0 | - | Off | + |
| 1 | 0 | 1 | - | + | Off |

## *B. Interfacing ADC for Potentiometer*

The output of potentiometer is analog voltage signal and cannot directly use on CPLD. To process the signal digitally, it must be converted to digital form using ADC chip. MCP3008 ADC chip is used in this case. It is 10-bit ADC with 8 channels and using SPI protocol-based to communicate to control unit.

SPI-protocol is communication for short-distance using serial interface. To initiating communication with ADC and CPLD as master device by send low signal to CS line. Figure 3 shows complete communication through MCP3008 ADC[16].

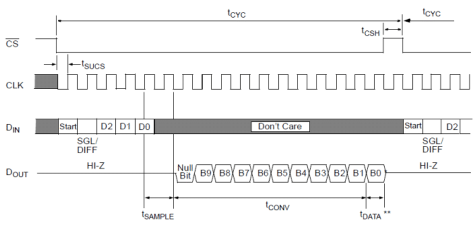


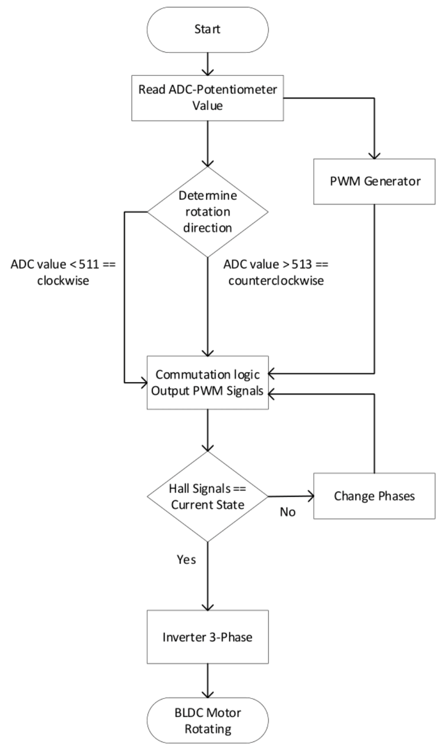
Fig. 3. Communication of MCP3008

Module to use this ADC is written in Verilog HDL and performed like FSM based. The data is sending through MOSI and MISO lines. Theoretical digital output of ADC is determined using equation as shown below.

$$Digital\ output\ code = \frac{1024 \times Vin}{Vref} \qquad (1)$$

## *C. The Generated PWM signals*

Figure 4 shows a flowchart for speed control of BLDC. Digital value of potentiometer is used to determine direction of rotation and duty cycle on our PWM signal on high side of the MOSFET. Using counter and compare it to value of ADC input as a reference.



Start
Read ADC-Potentiometer Value
PWM Generator
Determine rotation direction
ADC value < 511 == clockwise
ADC value > 513 == counterclockwise
Commutation logic Output PWM Signals
Hall Signals == Current State
Change Phases
No
Yes
Inverter 3-Phase
BLDC Motor Rotating

Fig. 4. Flowchart of the BLDC Control System.

## III. Simulation Results

The control algorithm of BLDC motor drive is simulate using Modelsim-Altera. Testbench was developed using Verilog HDL and our control_bldc module as a unit under test. This simulation aims to validate PWM signals generated in our design before being implemented to hardware.

The signals input from hall sensors and potentiometer are required to simulate the testbench. It simulates the changing potentiometer value reference and hall effects to examine output of the control motor. The clock frequency for this system is 50 MHz based on targeted device that will be implemented.

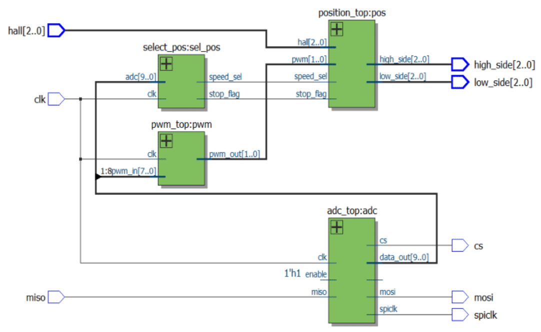


Fig. 6. RTL Viewer of Verilog HDL Design.

The simulation result for control algorithm is shown in Figure 5. PWM in this simulation is outputting duty cycle according to increasing value of ADC to the high side signals. By simulate 3-bit signals to hall input and potentiometer value, we can see correct commutation sequence for each phase according to Table I and Table II.

## IV. Implementation and Experimental Result

The simulation results verified the control algorithm according to the proposed system. Logic control was then implemented to CPLD Max II: EPM240T100C5 using JTAG via Quartus software. Figure 6 shows the logic schematic implemented to the CPLD. Total logic elements used in this design are 133 of 240 logic elements (LEs), utilize around 55% of EPM240 capability, and take 14 pins of 80 for input and output.

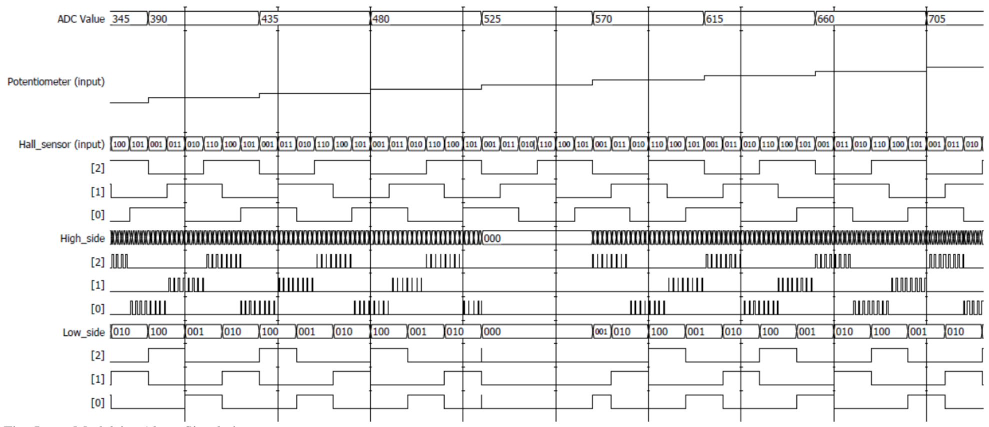
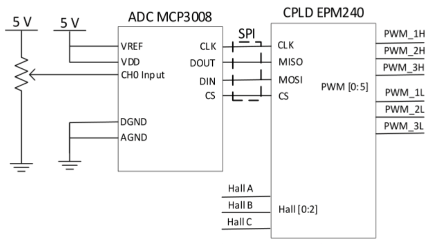
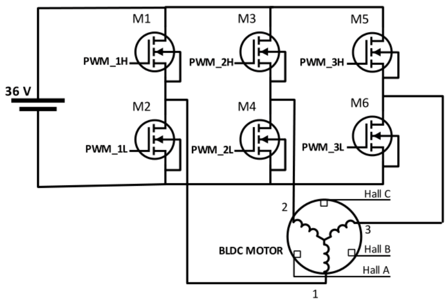


Fig. 5. Modelsim-Altera Simulation.



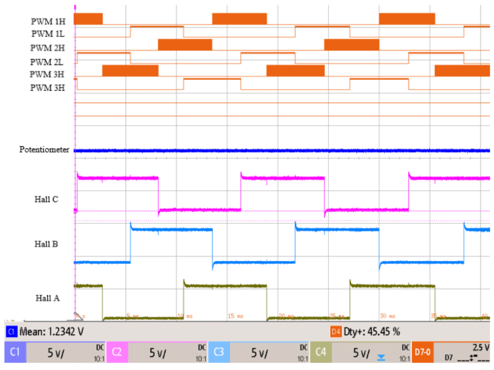
(a) Input and output connection



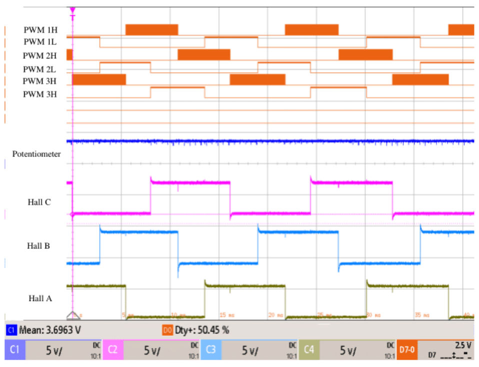
(b) 3-phase Inverter

Fig. 7. Hardware Implementation Setup.

The experimental setup is shown in Figure 7. BLDC motor used is rated for 36V and 350W, and six MOSFET is IRF3808. Hall built-in sensor on BLDC motor directly connected to CPLD through a pull-up resistor. Input signal from hall sensor is 5 volts for high logic signal and 0 volts for low logic signal. CPLD EPM240 runs at 3.3 V logic, and therefore it is needed a MOSFET driver to amplify the signal to the gate of MOSFETs. The result in Figure 8 shows the correlation between hall sensor, potentiometer value, and phase sequence. The correlation shows both commutation clockwise and counterclockwise direction at 50% duty cycle. Each high side output variable duty cycle according to potentiometer, and for low side, it changes whenever hall sensor jumps to next pattern.



(a) Clockwise Direction



(b) Counterclockwise Direction

Fig. 8. BLDC control motor signals on 50% duty cycle (experimental result).

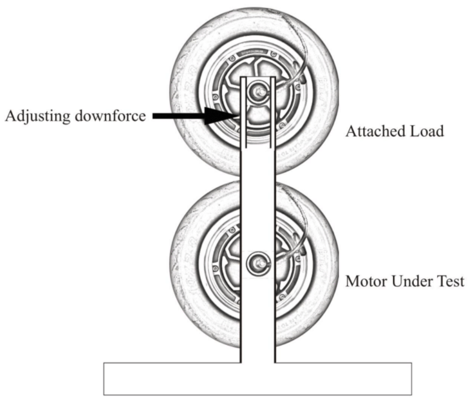


Fig. 9. Load testing experimental setup.

The testing of BLDC motor established in a condition without load and two variations with different loads. The load testing using other motor attach on top of the tested motor. Figure 10 shows speed measured of the BLDC motor during three loads variation on both directions. The changing speed of BLDC motor based on changes value of potentiometer voltage as PWM set reference. Maximum speed achieved on no-load condition is 600 RPM, variation 1 load is 500RPM, and variation 2 load is 370 RPM. The current for no-load, variation 1 and variation 2 are around 0,76A, 1,34A, and 2,27A.

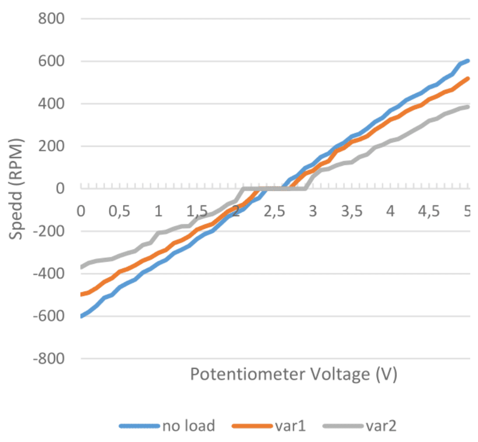


Fig. 10. Speed results on clockwise and counterclockwise direction.

## V. CONCLUSION

In this paper, the control system of the BLDC motor using a complex programmable logic device (CPLD) is presented. The control system generates PWM signals according to six-step commutation using hall sensors. BLDC motor speed control is achieved by varying duty cycles of PWM with a frequency 20KHz. Duty cycle controlled with comparing ADC value from a potentiometer and compare with an internal counter. The control system was created with Verilog HDL and simulated using Modelsim-Altera. The algorithm implemented to CPLD Max II: EPM240T100C5 with total logic elements (LEs) is about 133 out of 240, around 55% of EPM240 capability, and takes 14 pins for input and output for the overall system.

## ACKNOWLEDGMENT

The authors gratefully acknowledge Ministry of Research and Technology of the Republic of Indonesia, Board of National Research and Innovation for supporting the research under the scheme of Applied Research Grant with Grant Contract Number 7/E1/KP.PTNBH/2021.

## REFERENCES

[1] J. De Santiago *et al.*, "Electrical motor drivelines in commercial all-electric vehicles: A review," *IEEE Trans. Veh. Technol.*, vol. 61, no. 2, pp. 475–484, 2012, doi: 10.1109/TVT.2011.2177873.

[2] G. Tao, Z. Ma, L. Zhou, and L. Li, "A novel driving and control system for direct-wheel-driven electric vehicle," *2004 12th Symp. Electromagn. Launch Technol.*, vol. 41, no. 1, pp. 514–517, 2004, doi: 10.1109/elt.2004.1398134.

[3] Tae-Hyung Kim and M. Ehsani, "Sensorless control of the BLDC motors from near-zero to high speeds," *IEEE Trans. Power Electron.*, vol. 19, no. 6, pp. 1635–1645, Nov. 2004, doi: 10.1109/TPEL.2004.836625.

[4] M. Jafarboland and M. H. R. Silabi, "New sensorless commutation method for BLDC motors based on the line-to-line flux linkage theory," *IET Electr. Power Appl.*, vol. 13, no. 6, pp. 757–765, 2019, doi: 10.1049/iet-epa.2018.5356.

[5] M. Baszynski and S. Pirog, "A novel speed measurement method for a high-speed BLDC motor based on the signals from the rotor position sensor," *IEEE Trans. Ind. Informatics*, vol. 10, no. 1, pp. 84–91, 2014, doi: 10.1109/TII.2013.2243740.

[6] H. X. Wu, S. K. Cheng, and S. M. Cui, "A controller of brushless DC Motor for electric vehicle," *2004 12th Symp. Electromagn. Launch Technol.*, vol. 41, no. 1, pp. 528–533, 2004.

[7] Z. Q. Zhu and D. Howe, "Electrical machines and drives for electric, hybrid, and fuel cell vehicles," *Proc. IEEE*, vol. 95, no. 4, pp. 746–765, 2007, doi: 10.1109/JPROC.2006.892482.

[8] K. Kolano, "Improved Sensor Control Method for BLDC Motors," *IEEE Access*, vol. 7, pp. 186158–186166, 2019, doi: 10.1109/ACCESS.2019.2960580.

[9] B. P. Reddy and A. Murali, "SoC FPGA-based field oriented control of BLDC motor using low resolution Hall sensor," *IECON Proc. (Industrial Electron. Conf.*, pp. 2941–2945, 2016, doi: 10.1109/IECON.2016.7793092.

[10] T. H. Kim and M. Ehsani, "Sensorless control of the BLDC motors from near-zero to high speeds," *IEEE Trans. Power Electron.*, vol. 19, no. 6, pp. 1635–1645, 2004, doi: 10.1109/TPEL.2004.836625.

[11] J. Cervantes, E. Cordova, A. I. S. Marrufo, I. U. P. Monarrez, and M. Nandayapa, "BLDC motor commutation based on DSP builder for FPGA," *Int. Power Electron. Congr. - CIEP*, vol. 2016-Augus, pp. 166–171, 2016, doi: 10.1109/CIEP.2016.7530750.

[12] A. Tashakori, M. Hassanudeen, and M. Ektesabi, "FPGA based controller drive of BLDC motor using digital PWM technique," in *2015 IEEE 11th International Conference on Power Electronics and Drive Systems*, Jun. 2015, pp. 658–662, doi: 10.1109/PEDS.2015.7203584.

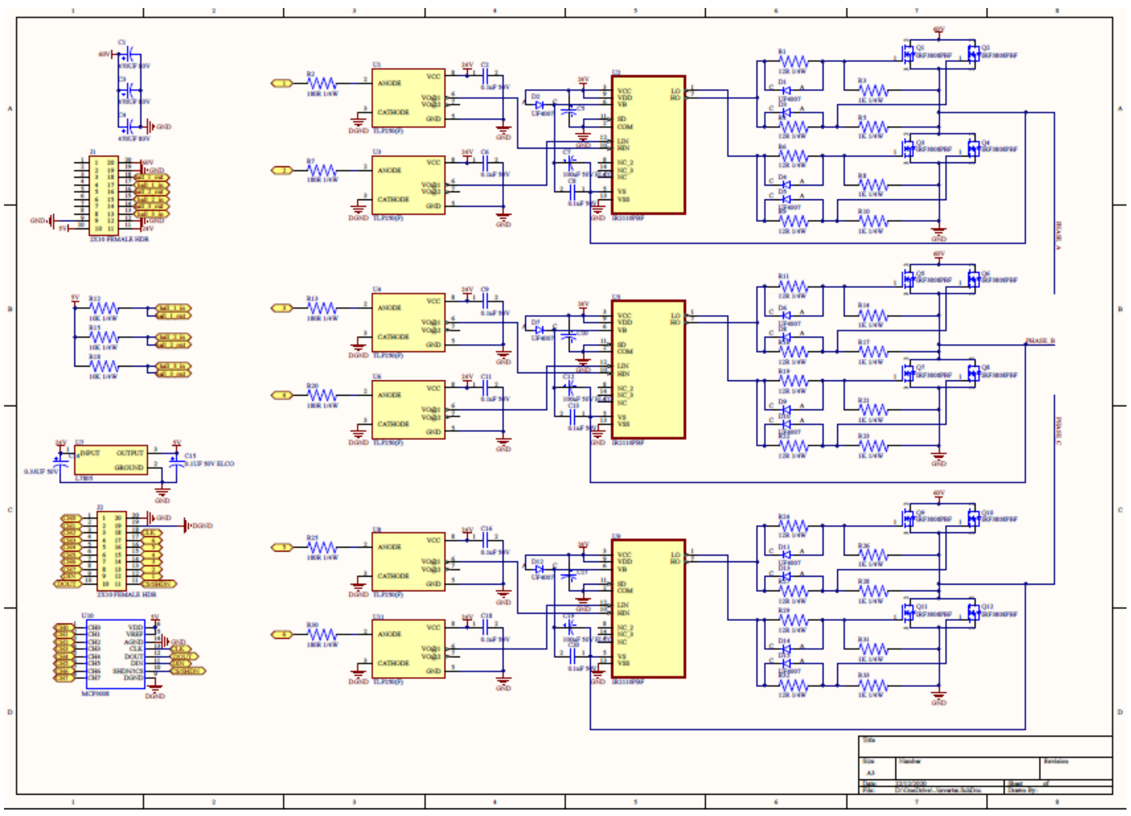
[13] G. Mihalache and A. D. Ioan, "FPGA Implementation of BLDC Motor Driver with Hall Sensor Feedback," *EPE 2018 - Proc. 2018 10th Int. Conf. Expo. Electr. Power Eng.*, pp. 624–629, 2018, doi: 10.1109/ICEPE.2018.8559886.

[14] R. M. Pindoriya, A. K. Mishra, B. S. Rajpurohit, and R. Kumar, "FPGA Based Digital Control Technique for BLDC Motor Drive," *IEEE Power Energy Soc. Gen. Meet.*, vol. 2018-Augus, pp. 1–5, 2018, doi: 10.1109/PESGM.2018.8586472.

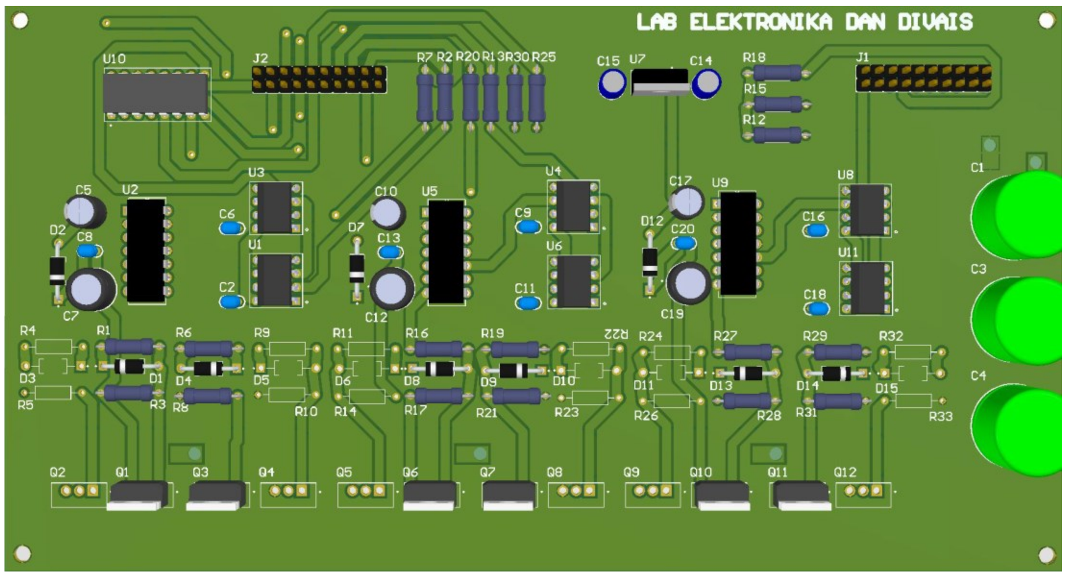
[15] M. H. Rashid, *The power electronics handbook*, Third edit. Elsevier Inc, 2011.

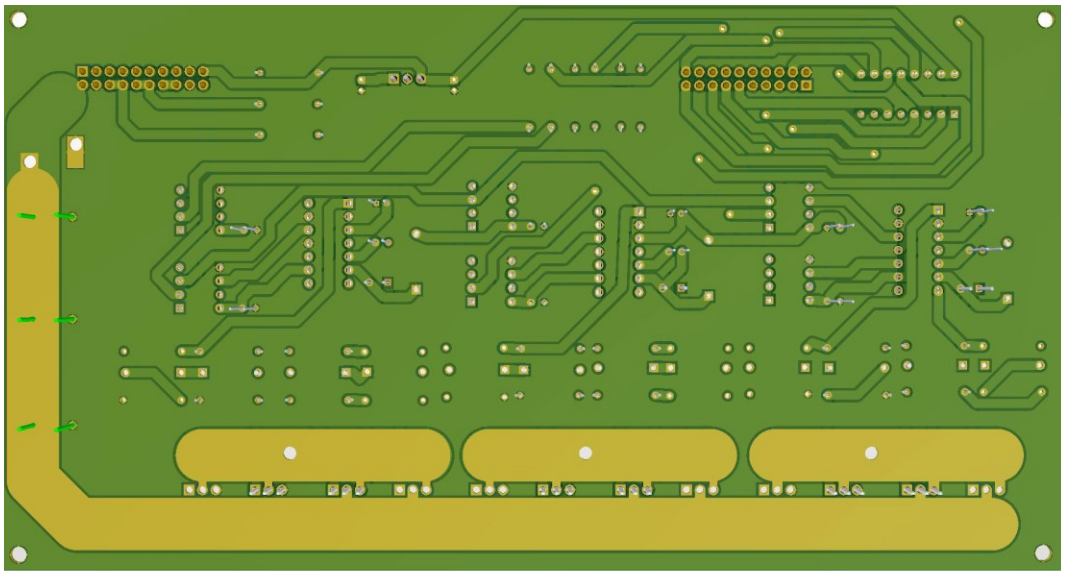
[16] M. T. Inc., "Mcp3004/3008." pp. 1–35, 2002.

Lampiran 2 Skematik Rangkaian Inverter 3 Fasa



Lampiran 3 Layout PCB Inverter Tiga Fasa





Lampiran 4 Netlist LTSpice Rangkaian Penyesuai Tegangan

```
R1 N006 N005 3100 tol=1 pwr=0.1
R2 N006 0 500 tol=1 pwr=0.1
R3 N002 N001 2K tol=1 pwr=0.1
R4 N003 N002 1K tol=1 pwr=0.1
V3 N005 0
V5 N004 0 12
V6 N007 0 -12
V4 N001 0 -5
XU1 N006 N002 N004 N007 N003 AD712
.dc V3 -12 +12 1
.lib ADI1.lib
.backanno
.end
```

Lampiran 5 Kode Verilog HDL SPI-MCP3008 ADC

```
`timescale 1ns / 1ps

module adc_top(clk,enable,mosi,miso,spiclk,cs,done,data_out);

input clk,enable,miso;
output cs, done, mosi,spiclk;
output [9:0] data_out;

wire [9:0] data;

adc_spi spi(spiclk, enable, 4'b1000, cs, done, miso, mosi, data);
clk_div a1(clk, 1'b0, spiclk);
fdc10 a0(data,spiclk,done,1'b0,data_out);

endmodule
```

```verilog
`timescale 1ns / 1ps

module adc_spi(clk, enable, control_bit, cs, done, miso, mosi, data);

   input clk, enable, miso;
   input [3:0] control_bit;
   output reg cs, done, mosi;
   output reg [9:0] data;

   parameter N = 4;

   reg [2:0] state = 0;
   reg [4:0] count = 0;
   reg [2:0] bit_posout = 3;
   reg [4:0] bit_pos = 0;


   always @(negedge clk) begin
      if (enable == 1) begin
         case (state)
            0: begin
                  cs <= 0;
                  done <= 0;
                  mosi = 1;
                  state <= state + 3'b001;
                  count <= count + 5'b00001;
                  data <= 0;
               end
            1: begin
                  if (count < 4 ) begin
                     cs <= 0;
                     done <= 0;
                     mosi = control_bit[bit_posout];
                     bit_posout = bit_posout - 3'b001;
                     count <= count + 5'b00001; //4
                     state <= 1;
                  end
```

```
        else begin
           cs <= 0;
           done <= 0;
           mosi = control_bit[bit_posout];
           bit_posout = bit_posout - 3'b001;
           count <= count + 5'b00001; //5
           state <= state + 3'b001;
        end
     end
  2: begin
        if (count < 8) begin
           cs     <= 0;
           done   <= 0;
           mosi = 0;
           state <= 2;
           count <= count + 5'b00001; end  //6
        else begin
           cs     <= 0;
           done   <= 0;
           mosi = 0;
           state <= 3;
           bit_pos = 5'b10001 - count;
           data[bit_pos] <= miso;
           count <= count + 5'b00001;
        end

     end
  3: begin
        if (count < 17) begin
           cs     <= 0;
           done   <= 0;
           mosi = 0;
           state <= 3;
           bit_pos = 5'b10001 - count;
           data[bit_pos] <= miso;
           count <= count + 5'b00001;end
        else begin
```

```
                     cs    <= 0;
                     done  <= 1;
                     mosi = 0;
                     state <= state + 3'b001;
                     bit_pos = 5'b10001 - count;
                     data[bit_pos] <= miso;
                     count <= count + 5'b00001;
                  end
               end
            4:    begin
                  if (count == 18) begin
                     cs    <= 1;
                     done  <= 0;
                     mosi = 0;
                     state <= 0;
                     count <= 0;
                     bit_posout = 3;
                  end
               end

            default: begin
                  cs <= 0;
                  state <= 0;
                  count <= 0;
                  done <= 0;
               end
         endcase
      end
      else begin
      //reset
         cs <= 0;
         state <= 0;
         count <= 0;
         done <= 0;
         data <= 0;
      end
   end

endmodule
```

```verilog
module clk_div
#(
parameter WIDTH = 8, // Width of the register required
parameter N = 250// 6 We will divide by 12 for example in this case
)
(clk,reset, clk_out);

input clk;
input reset;
output clk_out;

reg [WIDTH-1:0] r_reg = 1'b0;
wire [WIDTH-1:0] r_nxt;
reg clk_track = 1'b0;

always @(negedge clk)

begin
  if (reset)
     begin
        r_reg <= 0;
   clk_track <= 1'b0;
     end

  else if (r_nxt == N)
      begin
        r_reg <= 0;
        clk_track <= ~clk_track;
      end

  else
      r_reg <= r_nxt;
end

 assign r_nxt = r_reg+1'b1;
 assign clk_out = clk_track;
endmodule
```

```
`timescale 1ns / 1ps

module fdc10(a,clk,en,reset,y);
    input [9:0] a;
    input clk,en,reset;
    output [9:0] y;
fdcel d1(y[0],clk,en,reset,a[0]);
fdcel d2(y[1],clk,en,reset,a[1]);
fdcel d3(y[2],clk,en,reset,a[2]);
fdcel d4(y[3],clk,en,reset,a[3]);
fdcel d5(y[4],clk,en,reset,a[4]);
fdcel d6(y[5],clk,en,reset,a[5]);
fdcel d7(y[6],clk,en,reset,a[6]);
fdcel d8(y[7],clk,en,reset,a[7]);
fdcel d9(y[8],clk,en,reset,a[8]);
fdcel d10(y[9],clk,en,reset,a[9]);

endmodule
```

---

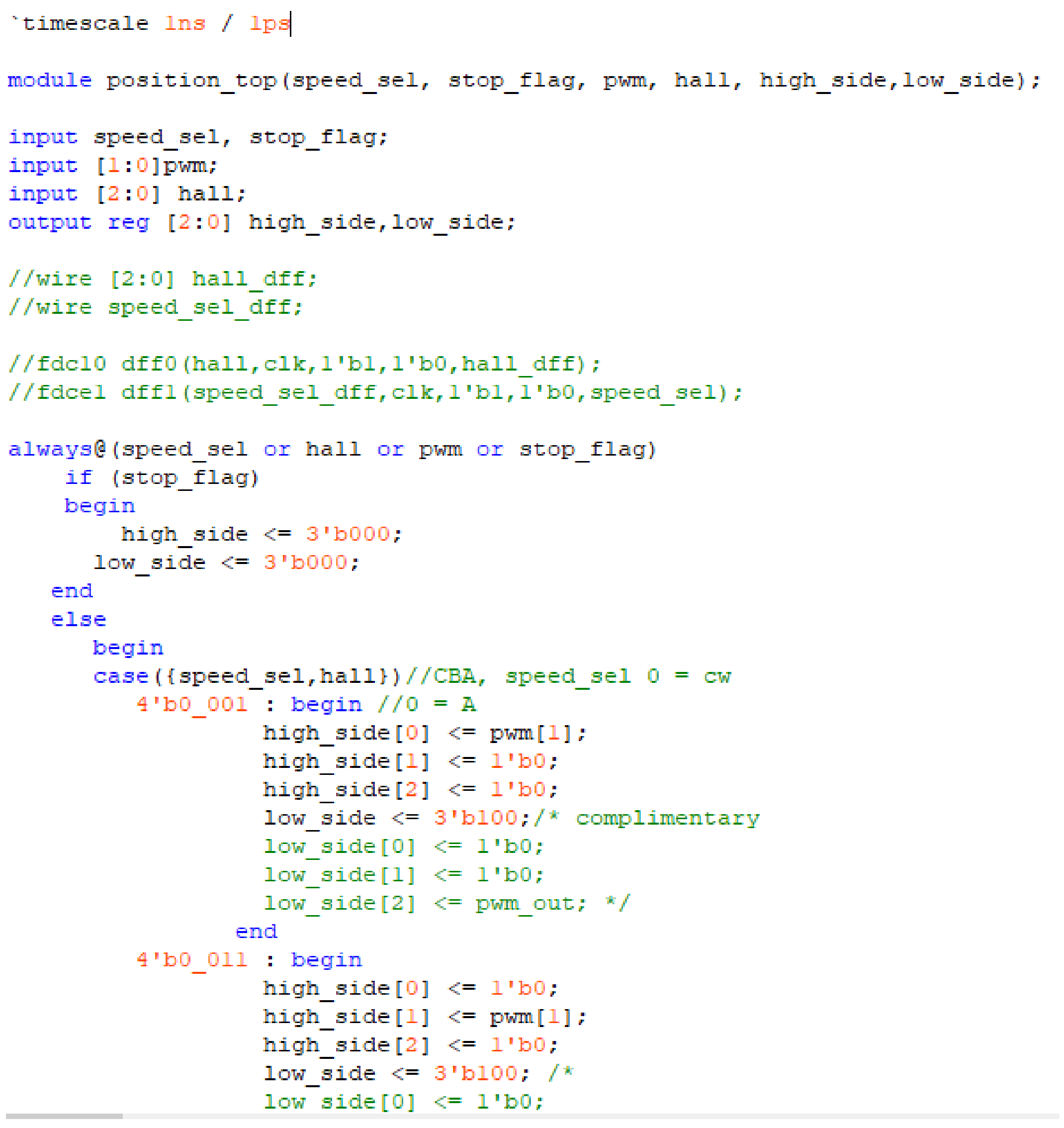
```
`timescale 1ns / 1ps

module fdcel(q,clk,ce,reset,d);
    input d,clk,ce,reset;
    output reg q;
initial begin q=0; end
always @ (negedge (clk)) begin
if (reset)
  q <= 1'b0;
 else if (ce)
  q <= d;
 else
 q<= q ;
end
endmodule
```

Lampiran 6 Kode Verilog HDL PWM Generator

```
`timescale 1ns / 1ps

module pwm_top(clk, pwm_in, pwm_out);

input clk;
input [7:0] pwm_in;
output [1:0] pwm_out;

wire clk_out;
reg[7:0] counter_PWM=5'd0;


clk_div clk0(clk,1'b0, clk_out);
   defparam clk0.N = 5;

always @(posedge clk_out)
begin
   begin
      counter_PWM <= counter_PWM + 8'b00000001;
      if(counter_PWM>=255)
       counter_PWM <= 0;
   end
end

assign pwm_out[0] = counter_PWM <= pwm_in ? 1'b1:1'b0;
assign pwm_out[1] = counter_PWM >= pwm_in ?1'b1:1'b0;

/*
pwm_high pwm0(clk_div, data, pwm_tmp0);
pwm_low pwm1(clk_div, data, pwm_tmp1);

fd a1(clk_div, 1'b0, 1'b1, pwm_in, data);
clock_div prescaler(clk, clk_div);

assign pwm_out = (speed_sel == 1'b1) ? pwm_tmp0:pwm_tmp1;
*/


endmodule
```

```
module clk_div
#(
parameter WIDTH = 8, // Width of the register required
parameter N = 250// 6 We will divide by 12 for example in this case
)
(clk,reset, clk_out);

input clk;
input reset;
output clk_out;

reg [WIDTH-1:0] r_reg = 1'b0;
wire [WIDTH-1:0] r_nxt;
reg clk_track = 1'b0;

always @(negedge clk)

begin
  if (reset)
     begin
        r_reg <= 0;
   clk_track <= 1'b0;
     end

  else if (r_nxt == N)
      begin
        r_reg <= 0;
        clk_track <= ~clk_track;
      end

  else
      r_reg <= r_nxt;
end

 assign r_nxt = r_reg+1'b1;
 assign clk_out = clk_track;
endmodule
```

Lampiran 7 Kode Verilog HDL Logika Komutasi



```
`timescale 1ns / 1ps

module position_top(speed_sel, stop_flag, pwm, hall, high_side,low_side);

input speed_sel, stop_flag;
input [1:0]pwm;
input [2:0] hall;
output reg [2:0] high_side,low_side;

//wire [2:0] hall_dff;
//wire speed_sel_dff;

//fdcl0 dff0(hall,clk,1'b1,1'b0,hall_dff);
//fdce1 dff1(speed_sel_dff,clk,1'b1,1'b0,speed_sel);

always@(speed_sel or hall or pwm or stop_flag)
    if (stop_flag)
    begin
        high_side <= 3'b000;
      low_side <= 3'b000;
   end
   else
      begin
      case({speed_sel,hall})//CBA, speed_sel 0 = cw
         4'b0_001 : begin //0 = A
                  high_side[0] <= pwm[1];
                  high_side[1] <= 1'b0;
                  high_side[2] <= 1'b0;
                  low_side <= 3'b100;/* complimentary
                  low_side[0] <= 1'b0;
                  low_side[1] <= 1'b0;
                  low_side[2] <= pwm_out; */
                end
         4'b0_011 : begin
                  high_side[0] <= 1'b0;
                  high_side[1] <= pwm[1];
                  high_side[2] <= 1'b0;
                  low_side <= 3'b100; /*
                  low_side[0] <= 1'b0;
         low_side[1] <= 1'b0;
         low_side[2] <= pwm_out; */
       end
4'b0_010 : begin
         high_side[0] <= 1'b0;
         high_side[1] <= pwm[1];
         high_side[2] <= 1'b0;
         low_side <= 3'b001;/*
         low_side[0] <= pwm_out;
         low_side[1] <= 1'b0;
         low_side[2] <= 1'b0;    */
       end
4'b0_110 : begin
         high_side[0] <= 1'b0;
         high_side[1] <= 1'b0;
         high_side[2] <= pwm[1];
         low_side <= 3'b001; /*
         low_side[0] <= pwm_out;
         low_side[1] <= 1'b0;
         low_side[2] <= 1'b0; */
       end
4'b0_100 : begin
         high_side[0] <= 1'b0;
         high_side[1] <= 1'b0;
         high_side[2] <= pwm[1];
         low_side <= 3'b010;/*
         low_side[0] <= 1'b0;
         low_side[1] <= pwm_out;
         low_side[2] <= 1'b0; */
       end
4'b0_101 : begin
         high_side[0] <= pwm[1];
         high_side[1] <= 1'b0;
         high_side[2] <= 1'b0;
         low_side <= 3'b010;/*
         low_side[0] <= 1'b0;
         low_side[1] <= pwm_out;
         low_side[2] <= 1'b0; */
       end
```

```
         4'b1_001 : begin //0 = A
                  high_side[0] <= 1'b0;
                  high_side[1] <= 1'b0;
                  high_side[2] <= pwm[0];
                  low_side <= 3'b001;
                end
         4'b1_011 : begin
                  high_side[0] <= 1'b0;
                  high_side[1] <= 1'b0;
                  high_side[2] <= pwm[0];
                  low_side <= 3'b010;
                end
         4'b1_010 : begin
                  high_side[0] <= pwm[0];
                  high_side[1] <= 1'b0;
                  high_side[2] <= 1'b0;
                  low_side <= 3'b010;
                end
         4'b1_110 : begin
                  high_side[0] <= pwm[0];
                  high_side[1] <= 1'b0;
                  high_side[2] <= 1'b0;
                  low_side <= 3'b100;
                end
         4'b1_100 : begin
                  high_side[0] <= 1'b0;
                  high_side[1] <= pwm[0];
                  high_side[2] <= 1'b0;
                  low_side <= 3'b100;
                end
         4'b1_101 : begin
                  high_side[0] <= 1'b0;
                  high_side[1] <= pwm[0];
                  high_side[2] <= 1'b0;
                  low_side <= 3'b001;
                end

         default: begin
                  high side <= 3'b000;


                  low_side <= 3'b000;
                end
      endcase
end


endmodule
```

Lampiran 8 Kode Verilog HDL Logika Arah Putar

```
module select_pos(clk, adc, stop_flag,speed_sel);

input clk;
input [9:0] adc;
output reg stop_flag, speed_sel;

always@(posedge clk)
begin
   if(adc > 10'b01_1111_1110 && adc < 10'b10_0000_0001)
   begin
      stop_flag <= 1'b1;
   end
   else
      stop_flag <= 1'b0;
end

always@(adc)
      begin
      case({adc[9]})
         1'b1 : begin //0 = A
                     speed_sel <= 1'b1;
                  end
         1'b0 : begin
                        speed_sel <= 1'b0;
                   end
      endcase
      end

endmodule
```

Lampiran 9 Kode Verilog HDL Top-level Kontrol BLDC

```
`timescale 1ns / 1ps

module control_top(clk, hall, mosi,miso,spiclk,cs, high_side,low_side);

input clk;
input miso;
input [2:0] hall;
output mosi,spiclk,cs;
output [2:0] high_side,low_side;

wire [9:0] adc_value;
wire done;
wire speed_sel;
wire stop_flag;
wire [1:0] pwm_out;

adc_top adc(clk,1'b1,mosi,miso,spiclk,cs,done,adc_value);
pwm_top pwm(clk, adc_value[8:1], pwm_out);
position_top pos(speed_sel, stop_flag, pwm_out, hall, high_side,low_side);
select_pos sel_pos(clk, adc_value, stop_flag,speed_sel);


endmodule
```

Lampiran 10 Kode Verilog HDL Testbench

```
`timescale 1 ns / 10 ps

module control_bldc_tb();

reg clk;
reg [2:0] out_hall;
reg [9:0] out_adc = 10'd210;
wire [2:0] high_side, low_side;

reg [2:0] hall [0:5];

integer i = 0, j = 0;
integer period_hall;

control_top DUT(clk, out_hall, out_adc, high_side, low_side);

always
begin
   clk = 1'b1;
   #500;
   clk = 1'b0;
   #500;
end

//array hall position
initial begin
   hall[0] = 3'b001;
   hall[1] = 3'b011;
   hall[2] = 3'b010;
   hall[3] = 3'b110;
   hall[4] = 3'b100;
   hall[5] = 3'b101;
end


initial begin
   period_hall = 20_000_000;

      for(i=0;i<24; i = i + 1) begin
```

```
         out_adc <= out_adc + 45;
         for(j=0; j< 6 ; j=  j+1) begin
            out_hall <= hall[j];
            #10000000;
            if(period_hall>2000000) begin
               period_hall = period_hall - 170000;
            end else begin
               period_hall = 1930000;
            end
         end
      end

    if (period_hall == 1930000) begin
      out_adc <= 10'd920;
      for(i=24;i<50; i = i + 1) begin
         for(j=0; j< 6 ; j=  j+1) begin
            out_hall <= hall[j];
            #10000000;
         end
      end
   end
end



endmodule
```