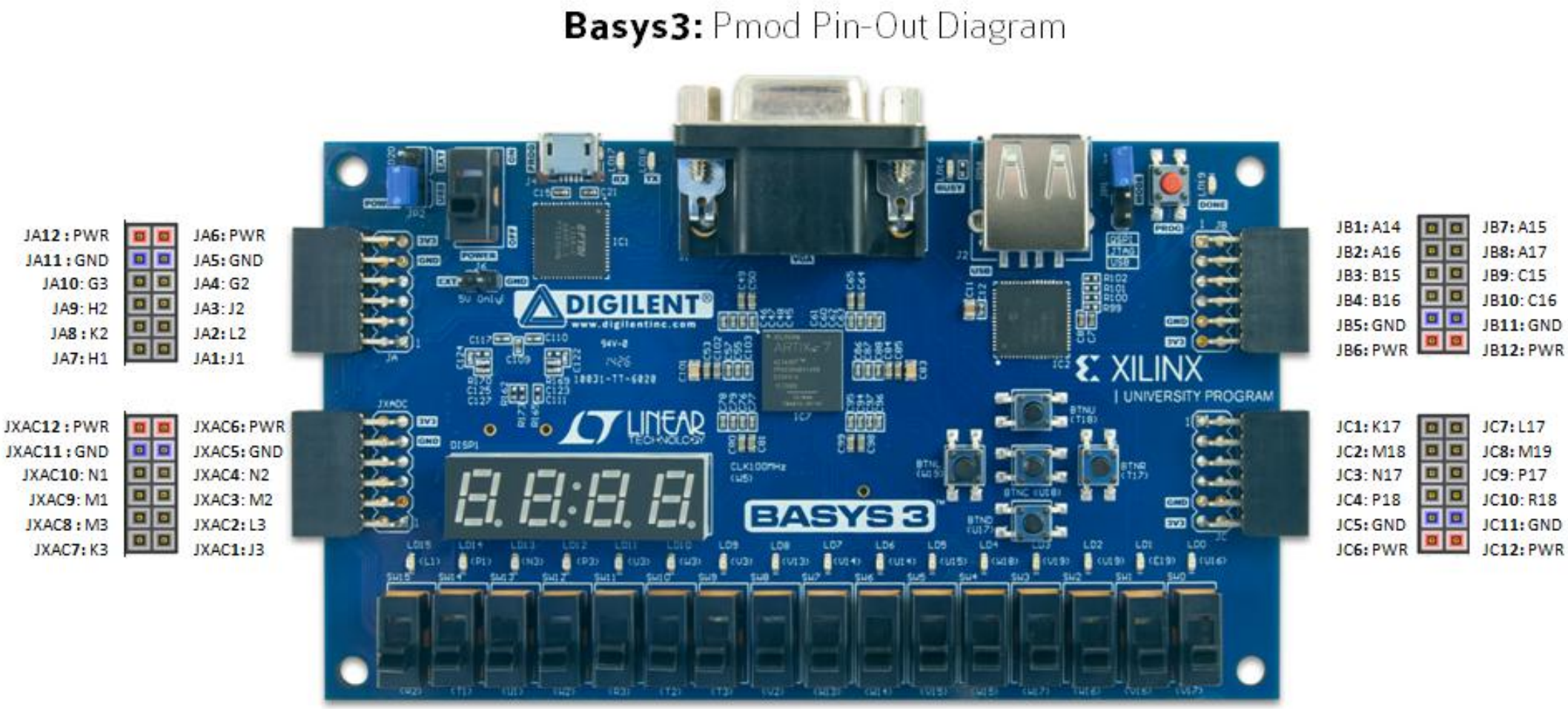
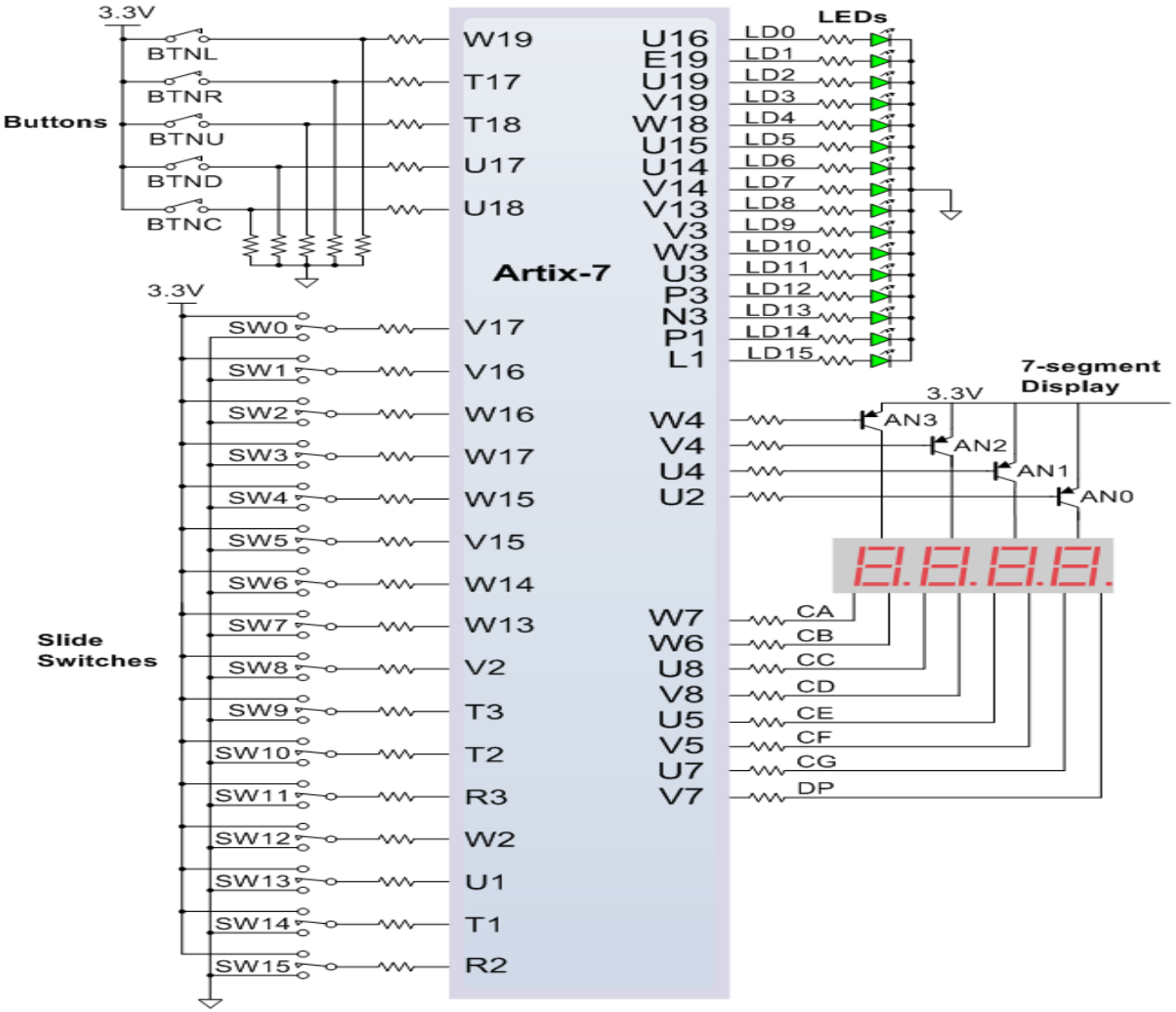
verilog HDL로
DC motor PWM제어
디지털 회로설계

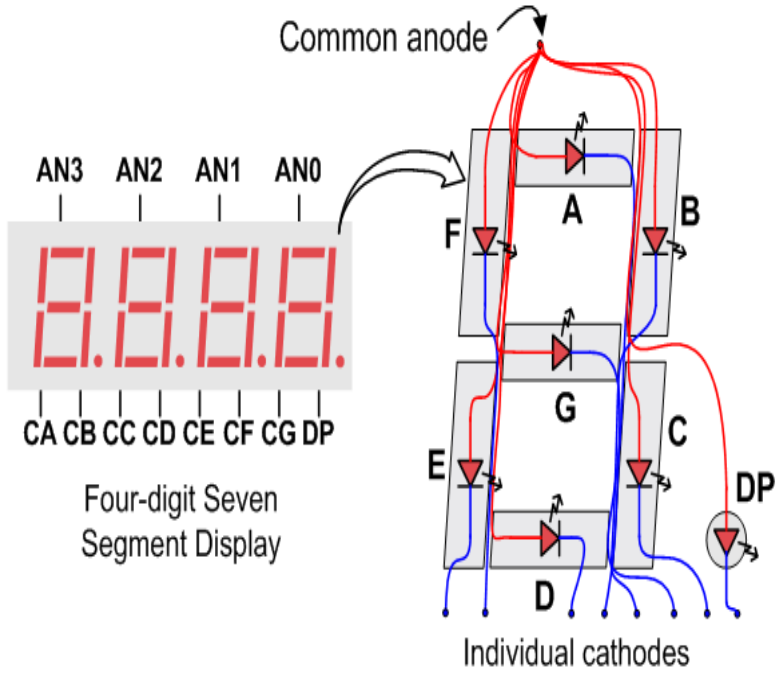
June 19,2025
by SIKWON

basys3 pmod 포트

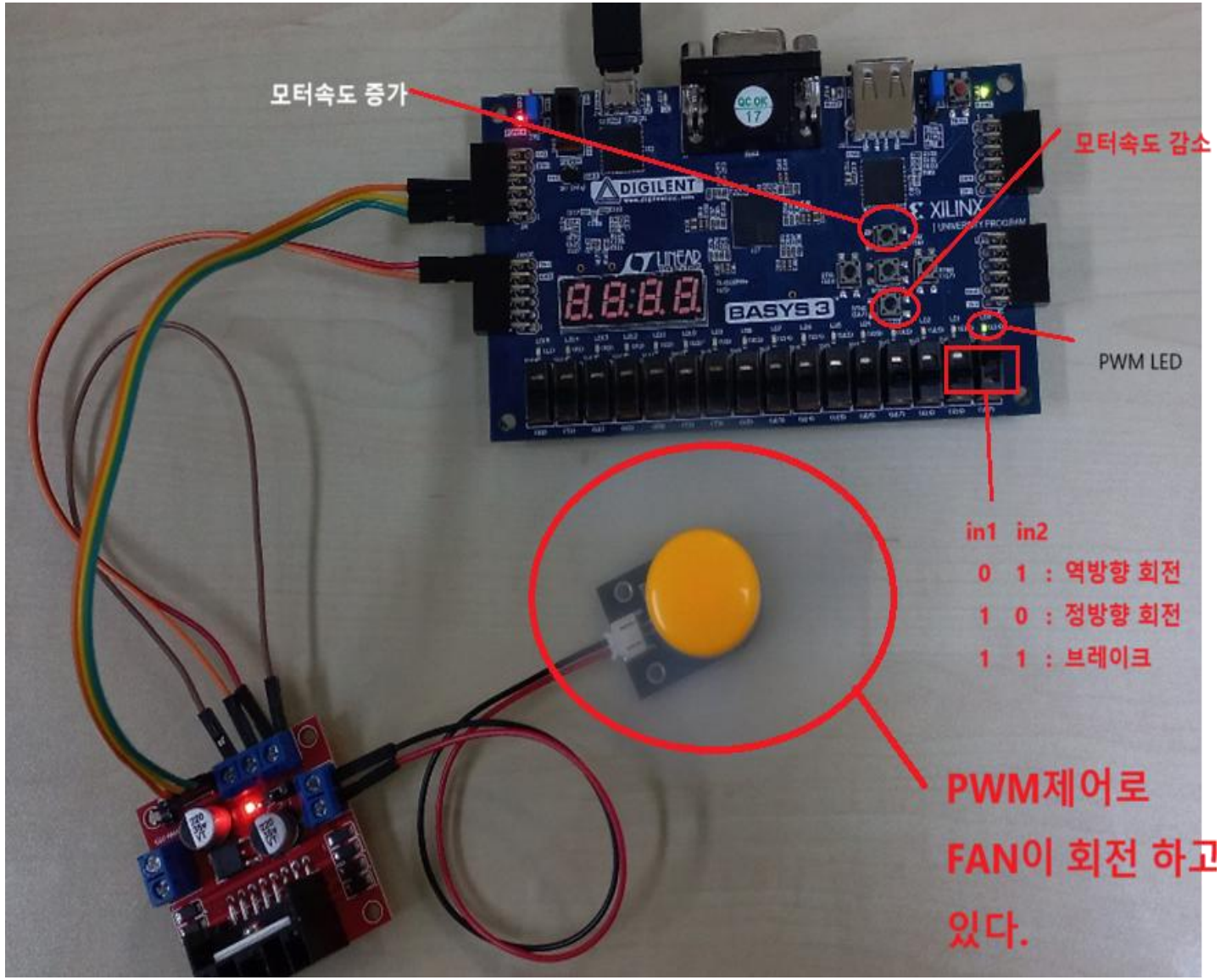


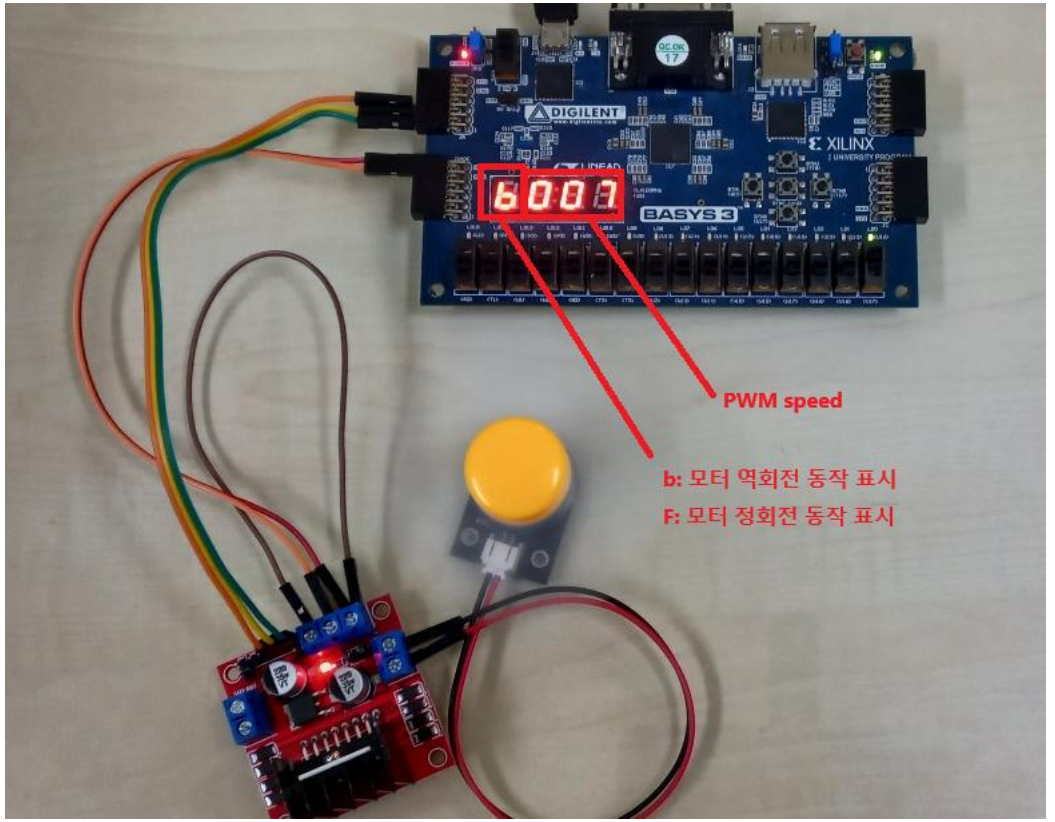
basys3 pmod 포트



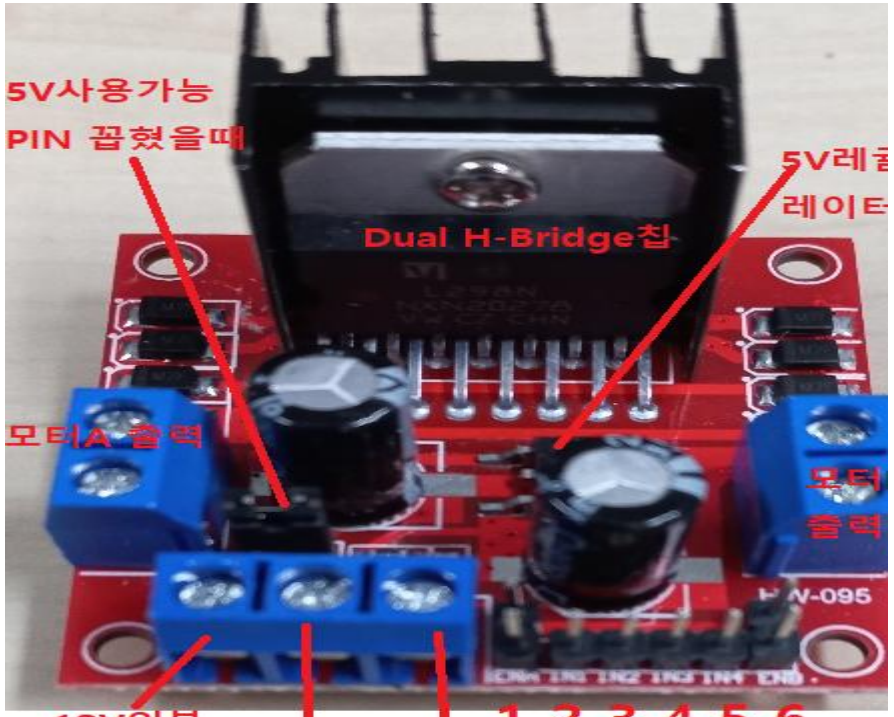


DC Motor PWM control





DC Motor PWM control



1.모터A PWM 제어
2.모터A 컨트롤 IN1
3.모터A 컨트롤 IN2
4.모터B 컨트롤 IN3
5.모터B 컨트롤 IN4
6.모터B PWM 제어

pwm_dcmotor.v

```
module pwm_dcmotor (
    input clk,   // 100MHz clock input
    input increase_duty_btn,   // input to increase 10% duty cycle
    input decrease_duty_btn,   // input to decrease 10% duty cycle
    input [1:0] motor_direction,  // sw0 sw1 : motor direction
    output PWM_OUT,        // 10MHz PWM output signal
    output PWM_OUT_LED,
    output [1:0] in1_in2,  // motor direction switch sw[0] sw[1]
    output [3:0] an,    // anode signals of the 7-segment LED display
    output [6:0] seg,    // cathode patterns of the 7-segment LED display
    output dp
    /*
      in1   in2
        0    1   :  역방향 회전
        1    0   :  정방향 회전
        1    1   :  브레이크
    */
    );

    wire w_debounced_inc_btn;
    wire w_debounced_dec_btn;
    wire [3:0] w_DUTY_CYCLE;

    debounce_pushbutton u_debounce_pushbutton (
        .clk(clk),   // 100MHz clock input
        .increase_duty_btn(increase_duty_btn),   // input to increase 10% duty cycle
        .decrease_duty_btn(decrease_duty_btn),   // input to decrease 10% duty cycle
        .debounced_inc_btn(w_debounced_inc_btn),      // debounce inc button
        .debounced_dec_btn(w_debounced_dec_btn)
    );

    pwm_duty_cycle_control u_pwm_duty_cycle_control (
            .clk(clk),
            .duty_inc(w_debounced_inc_btn),
            .duty_dec(w_debounced_dec_btn),
            .DUTY_CYCLE(w_DUTY_CYCLE),
            .PWM_OUT(PWM_OUT),        // 10MHz PWM output signal
            .PWM_OUT_LED(PWM_OUT_LED)
        );

        fnd_display u_fnd_display (
            .clock_100Mhz(clk),
            .in1_in2(in1_in2),
            .display_number(w_DUTY_CYCLE),
            .an(an),
            .seg(seg),
            .dp(dp)
        );

        assign in1_in2 = motor_direction;
endmodule
```

debounce_pushbutton.v

```
`timescale 1ns / 1ps

//----------------- debounce_pushbutton -----------------
module debounce_pushbutton (
    input clk,    // 100MHz clock input
    input increase_duty_btn,    // input to increase 10% duty cycle
    input decrease_duty_btn,    // input to decrease 10% duty cycle
    output debounced_inc_btn,      // debounce inc button
    output debounced_dec_btn
);

 wire w_clk4hz_enable;   // slow clock enable signal for debouncing FFs
 reg[27:0] r_counter_debounce=0;  // counter for creating slow clock enable signals
 wire w_Q1_DFF1, w_Q2_DFF2;
 wire w_Q1_DFF3, w_Q2_DFF4;
 reg[1:0] r_motor_dir;
  // Debouncing 2 buttons for inc/dec duty cycle
  // Firstly generate slow clock enable for debouncing flip-flop (4Hz)

 always @(posedge clk)
 begin
   r_counter_debounce <= r_counter_debounce + 1;
   if (r_counter_debounce >= 25000000)
    r_counter_debounce <= 0;
 end

// 0.00000001sec(10ns) x 25000000 = 0.25sec(250ms)
 // 250ms가 되면 4Hz의 1주기를 나타내는 w_clk4hz_enable이 1로 set하여
 // DFF의 4Hz clock이 동작 되도록 한다.
 assign w_clk4hz_enable = r_counter_debounce == 25000000 ? 1:0;

 DFF u_PWM_DFF1(clk,w_clk4hz_enable,increase_duty_btn,w_Q1_DFF1);
 DFF u_DFF2(clk,w_clk4hz_enable,w_Q1_DFF1, w_Q2_DFF2);
 assign debounced_inc_btn =  w_Q1_DFF1 & (~w_Q2_DFF2) & w_clk4hz_enable;

 // debouncing FFs for decreasing button
 DFF u_DFF3(clk, w_clk4hz_enable, decrease_duty_btn, w_Q1_DFF3);
 DFF u_DFF4(clk, w_clk4hz_enable, w_Q1_DFF3, w_Q2_DFF4);
 // button의 debounce를 위해서 첫번째 DFF의 출력 w_Q1_DFF3(Q1)
  //두번째 DFF의 출력 w_Q2_DFF4(Q2바) 를 and 해서
 //  debounce 처리된 1(debounced_inc_btn, debounced_dec_btn)이 나온다.
 assign debounced_dec_btn =  w_Q1_DFF3 & (~w_Q2_DFF4) & w_clk4hz_enable;
 // vary the duty cycle using the debounced buttons above
endmodule
```

pwm_duty_cycle_control.v

```
`timescale 1ns / 1ps
//----------------- pwm_duty_cycle_control ---------------------
module pwm_duty_cycle_control (
    input clk,
    input duty_inc,
    input duty_dec,
    output [3:0] DUTY_CYCLE,
    output PWM_OUT,        // 10MHz PWM output signal
    output PWM_OUT_LED
 );

  reg[3:0] r_DUTY_CYCLE=5;      // initial duty cycle is 50%
  reg[3:0] r_counter_PWM=0;     // counter for creating 10Mhz PWM signal

 always @(posedge clk)
 begin
   if (duty_inc==1 && r_DUTY_CYCLE <= 9)
      r_DUTY_CYCLE <= r_DUTY_CYCLE + 1; // increase duty cycle by 10%
   else if(duty_dec==1 && r_DUTY_CYCLE >= 1)
      r_DUTY_CYCLE <= r_DUTY_CYCLE - 1; //decrease duty cycle by 10%
 end

// Create 10MHz PWM signal with variable duty cycle controlled by 2 buttons
// DC로 10MHz PWM 신호를 보내도록 한다.
// default r_DUTY_CYCLE은 50%로 설정 r_counter_PWM는 10ns(1/100MHz) 마다 10%씩 증가
 always @(posedge clk)
 begin
   r_counter_PWM <= r_counter_PWM + 1;
   if (r_counter_PWM >= 9)
    r_counter_PWM <= 0;
 end

 assign PWM_OUT = r_counter_PWM < r_DUTY_CYCLE ? 1:0;
 assign PWM_OUT_LED = PWM_OUT;
 assign DUTY_CYCLE = r_DUTY_CYCLE;
endmodule
```

fnd_display.v

```
`timescale 1ns / 1ps

module fnd_display(
    input clock_100Mhz, // 100 Mhz clock source on Basys 3 FPGA
    input [1:0] in1_in2,  // motor direction switch sw[0] sw[1]
    input [3:0] display_number,
    output [3:0] an,    // anode signals of the 7-segment LED display
    output [6:0] seg,    // cathode patterns of the 7-segment LED display
    output dp
    );

parameter T250_MS = 25000000;
parameter T500_MS = 50000000;
parameter FORWARD =  4'b1010;      // f
parameter BACKWARD =  4'b1011;     // b


    reg [3:0] LED_BCD;
    reg [26:0] fnd_toggle_counter;
    reg dis_mode = 1'b1;
    reg [19:0] refresh_counter; // 20-bit for creating 10.5ms refresh period or 380Hz refresh rate
             // the first 2 MSB bits for creating 4 LED-activating signals with 2.6ms digit period
             // bin 0011 1111 1111 1111 1111   <---> dec 262143
    // org wire [1:0] LED_activating_counter;
    reg  [1:0] LED_activating_counter;
            // count          0     ->  1  ->  2  ->  3
            // activates     LED1    LED2   LED3   LED4
            // and repeat

always @(posedge clock_100Mhz)
    begin
        refresh_counter <= refresh_counter + 1;
        if (fnd_toggle_counter >= 99_999_999) begin
            fnd_toggle_counter <= 0;
            dis_mode <= ~dis_mode;
        end else
            fnd_toggle_counter <= fnd_toggle_counter + 1;
    end
    // org assign LED_activating_counter = refresh_counter[19:18];
    // anode activating signals for 4 LEDs, digit period of 2.6ms
    // decoder to generate anode signals
    reg dp_toggle = 1'b0;    // toggle per every 1 sec
    reg r_dp;
    reg [3:0] r_an;
    assign an = r_an;
```

fnd_display.v

```
`timescale 1ns / 1ps

module fnd_display(
    input clock_100Mhz, // 100 Mhz clock source on Basys 3 FPGA
    input [1:0] in1_in2,  // motor direction switch sw[0] sw[1]
    input [3:0] display_number,
    output [3:0] an,    // anode signals of the 7-segment LED display
    output [6:0] seg,    // cathode patterns of the 7-segment LED display
    output dp
    );

parameter T250_MS = 25000000;
parameter T500_MS = 50000000;
parameter FORWARD =  4'b1010;      // f
parameter BACKWARD =  4'b1011;     // b


    reg [3:0] LED_BCD;
    reg [26:0] fnd_toggle_counter;
    reg dis_mode = 1'b1;
    reg [19:0] refresh_counter; // 20-bit for creating 10.5ms refresh period or 380Hz refresh rate
             // the first 2 MSB bits for creating 4 LED-activating signals with 2.6ms digit period
             // bin 0011 1111 1111 1111 1111  <---> dec 262143
    // org wire [1:0] LED_activating_counter;
    reg  [1:0] LED_activating_counter;
            // count          0    ->  1  ->  2  ->  3
            // activates    LED1    LED2   LED3   LED4
            // and repeat

always @(posedge clock_100Mhz)
    begin
        refresh_counter <= refresh_counter + 1;
        if (fnd_toggle_counter >= 99_999_999) begin
            fnd_toggle_counter <= 0;
            dis_mode <= ~dis_mode;
        end else
            fnd_toggle_counter <= fnd_toggle_counter + 1;
    end
    // org assign LED_activating_counter = refresh_counter[19:18];
    // anode activating signals for 4 LEDs, digit period of 2.6ms
    // decoder to generate anode signals
    reg dp_toggle = 1'b0;    // toggle per every 1 sec
    reg r_dp;
    reg [3:0] r_an;
    assign an = r_an;
```

fnd_display.v

```
always @(*)
    begin
        // org case(LED_activating_counter)
        case(refresh_counter[19:18])
        2'b00: begin
            r_an <= 4'b0111;
            // activate LED1 and Deactivate LED2, LED3, LED4
            // LED_BCD <= display_number/1000;
            if (dis_mode == 1'b1) begin
                if (in1_in2 == 2'b10)
                    LED_BCD <= FORWARD;
                else if (in1_in2 == 2'b01)
                    LED_BCD <= BACKWARD;
                else LED_BCD <= 0;
            end
                else r_an <= 4'b1111;     // 1'st digit off
            r_dp <= 1;
            // the first digit of the 16-bit number
              end
        2'b01: begin
            r_an <= 4'b1011;
            // activate LED2 and Deactivate LED1, LED3, LED4
            LED_BCD = (display_number % 1000)/100;

                // the second digit of the 16-bit number
                // r_dp <= dp_toggle;  // dp toggle every 1 sec
             r_dp <= 1;
            end
        2'b10: begin
                r_an <= 4'b1101;
                // activate LED3 and Deactivate LED2, LED1, LED4
                 LED_BCD <= ((display_number % 1000)%100)/10;
                // the third digit of the 16-bit number
                    r_dp <= 1;
                    end
          2'b11: begin
                r_an <= 4'b1110;
                // activate LED4 and Deactivate LED2, LED3, LED1
                LED_BCD <= ((display_number % 1000)%100)%10;
                // the fourth digit of the 16-bit number
                    r_dp <= 1;
                   end
          endcase
    end

reg [6:0] r_fnd_dis;
    assign seg = r_fnd_dis;
    // assign dp = r_dp;
```

fnd_display.v

```
// Cathode patterns of the 7-segment LED display
    always @(*)
    begin
        case(LED_BCD)
        4'b0000: r_fnd_dis <= 7'b0000001; // "0"
        4'b0001: r_fnd_dis <= 7'b1001111; // "1"
        4'b0010: r_fnd_dis <= 7'b0010010; // "2"
        4'b0011: r_fnd_dis <= 7'b0000110; // "3"
        4'b0100: r_fnd_dis <= 7'b1001100; // "4"
        4'b0101: r_fnd_dis <= 7'b0100100; // "5"
        4'b0110: r_fnd_dis <= 7'b0100000; // "6"
        4'b0111: r_fnd_dis <= 7'b0001111; // "7"
        4'b1000: r_fnd_dis <= 7'b0000000; // "8"
        4'b1001: r_fnd_dis <= 7'b0000100; // "9"
        4'b1010: r_fnd_dis <= 7'b0111000; // "f"
        4'b1011: r_fnd_dis <= 7'b1100000; // "b"
        default: r_fnd_dis <= 7'b0000001; // "0"
        endcase
    end
endmodule
```

dff.v

```
`timescale 1ns / 1ps

//------------------  DFF ----------------------
// Debouncing DFFs for push buttons on FPGA
// 4Hz 주파수의 1주기가 250ms 100MHz/4 --> 25,000,000cycle을 count하면
// w_clk4hz_enable이 1로 set되어 en이 1로 mapping 된다.
module DFF (
    input clk,
    input en,
    input D,
    output reg Q
);

  always @(posedge clk)
  begin
  if (en == 1) // slow clock enable signal
    Q <= D;
  end
endmodule
```

constraints file

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {motor_direction[0]}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {motor_direction[1]}]

## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports PWM_OUT_LED ]

##7 Segment Display
set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]
set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]
set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]
set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]
set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]
set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]
set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]
set_property -dict { PACKAGE_PIN V7   IOSTANDARD LVCMOS33 } [get_ports dp]
set_property -dict { PACKAGE_PIN U2   IOSTANDARD LVCMOS33 } [get_ports {an[0]}]
set_property -dict { PACKAGE_PIN U4   IOSTANDARD LVCMOS33 } [get_ports {an[1]}]
set_property -dict { PACKAGE_PIN V4   IOSTANDARD LVCMOS33 } [get_ports {an[2]}]
set_property -dict { PACKAGE_PIN W4   IOSTANDARD LVCMOS33 } [get_ports {an[3]}]
```

constraints file

```
##Buttons
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports btnC]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports btnU]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports increase_duty_btn]
#set_property -dict { PACKAGE_PIN W19   IOSTANDARD LVCMOS33 } [get_ports btnL]
#set_property -dict { PACKAGE_PIN T17   IOSTANDARD LVCMOS33 } [get_ports btnR]
#set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports btnD]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports decrease_duty_btn]

##Pmod Header JA
set_property -dict { PACKAGE_PIN J1   IOSTANDARD LVCMOS33 } [get_ports {PWM_OUT}]; #Sch name = JA1
set_property -dict { PACKAGE_PIN L2   IOSTANDARD LVCMOS33 } [get_ports {in1_in2[0]}]; #Sch name = JA2
set_property -dict { PACKAGE_PIN J2   IOSTANDARD LVCMOS33 } [get_ports {in1_in2[1]}]; #Sch name = JA3
#set_property -dict { PACKAGE_PIN G2   IOSTANDARD LVCMOS33 } [get_ports {JA[3]}];#Sch name = JA4
#set_property -dict { PACKAGE_PIN H1   IOSTANDARD LVCMOS33 } [get_ports {JA[4]}];#Sch name = JA7
#set_property -dict { PACKAGE_PIN K2   IOSTANDARD LVCMOS33 } [get_ports {JA[5]}];#Sch name = JA8
#set_property -dict { PACKAGE_PIN H2   IOSTANDARD LVCMOS33 } [get_ports {JA[6]}];#Sch name = JA9
#set_property -dict { PACKAGE_PIN G3   IOSTANDARD LVCMOS33 } [get_ports {JA[7]}];#Sch name = JA10

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```