ECE3300L Lab 8

RGB LED PWM Controller

Group X
Czyrone Agbayani (BID: 014766336)
Caleb Jala-Guinto (BID: 015446587)

Objective:

The goal of this lab is to design and implement a hardware-controlled RGB LED using Pulse Width Modulation (PWM) on the Nexys A7 FPGA board. It uses a fixed clock divider to generate ~1 kHz for button debouncing and state machine operation, and ~20 kHz for flicker-free PWM generation. A single push-button, processed through a debounce and one-pulse circuit, cycles a finite state machine that sequentially loads the PWM resolution and the red, green, and blue duty cycles from an 8-bit switch input into dedicated registers. These values are synchronized from the slow clock domain into the PWM domain and applied to a PWM core to produce independent color channel signals. An RGB LED driver inverts these signals for active-low LEDs, and debug LEDs LED3–LED0 indicate the currently selected load slot.

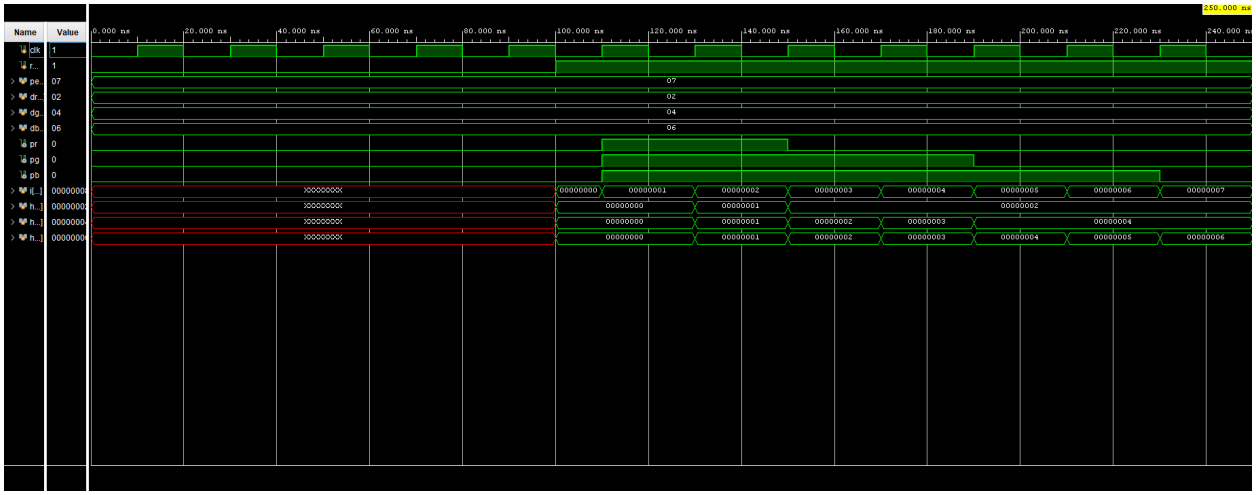
XDC File:

```
## This file is a general .xdc for the Nexys A7-100T
## 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
## Note: As the Nexys 4 DDR was rebranded to the Nexys A7 with no substantial changes, this XDC file will also work for the Nexys 4 DDR.

## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk100mhz }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk100mhz}];


##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { sw[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { sw[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { sw[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { sw[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
#set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { SW[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]


## RGB LEDs
set_property -dict { PACKAGE_PIN R12   IOSTANDARD LVCMOS33 } [get_ports { rgb_b }]; #IO_L5P_T0_D06_14 Sch=led16_b
set_property -dict { PACKAGE_PIN M16   IOSTANDARD LVCMOS33 } [get_ports { rgb_g }]; #IO_L10P_T1_D14_14 Sch=led16_g
set_property -dict { PACKAGE_PIN N15   IOSTANDARD LVCMOS33 } [get_ports { rgb_r }]; #IO_L11P_T1_SRCC_14 Sch=led16_r
#set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { LED17_B }]; #IO_L15N_T2_DQS_ADV_B_15 Sch=led17_b
#set_property -dict { PACKAGE_PIN R11   IOSTANDARD LVCMOS33 } [get_ports { LED17_G }]; #IO_0_14 Sch=led17_g
#set_property -dict { PACKAGE_PIN N16   IOSTANDARD LVCMOS33 } [get_ports { LED17_R }]; #IO_L11N_T1_SRCC_14 Sch=led17_r

##7 segment display
#set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { CA }]; #IO_L24N_T3_A00_D16_14 Sch=ca
#set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { CB }]; #IO_25_14 Sch=cb
#set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { CC }]; #IO_25_15 Sch=cc
#set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { CD }]; #IO_L17P_T2_A26_15 Sch=cd
#set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { CE }]; #IO_L13P_T2_MRCC_14 Sch=ce
#set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { CF }]; #IO_L19P_T3_A10_D26_14 Sch=cf
#set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { CG }]; #IO_L4P_T0_D04_14 Sch=cg
#set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
#set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
#set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
#set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
#set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
#set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
#set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
#set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
#set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]

##CPU Reset Button
#set_property -dict { PACKAGE_PIN C12   IOSTANDARD LVCMOS33 } [get_ports { CPU_RESETN }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn

##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { btnc_n }]; #IO_L9P_T1_DQS_14 Sch=btnc
#set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { BTNU }]; #IO_L4N_T0_D05_14 Sch=btnu
#set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports { BTNL }]; #IO_L12P_T1_MRCC_14 Sch=btnl
set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { btnr }]; #IO_L10N_T1_D15_14 Sch=btnr
```

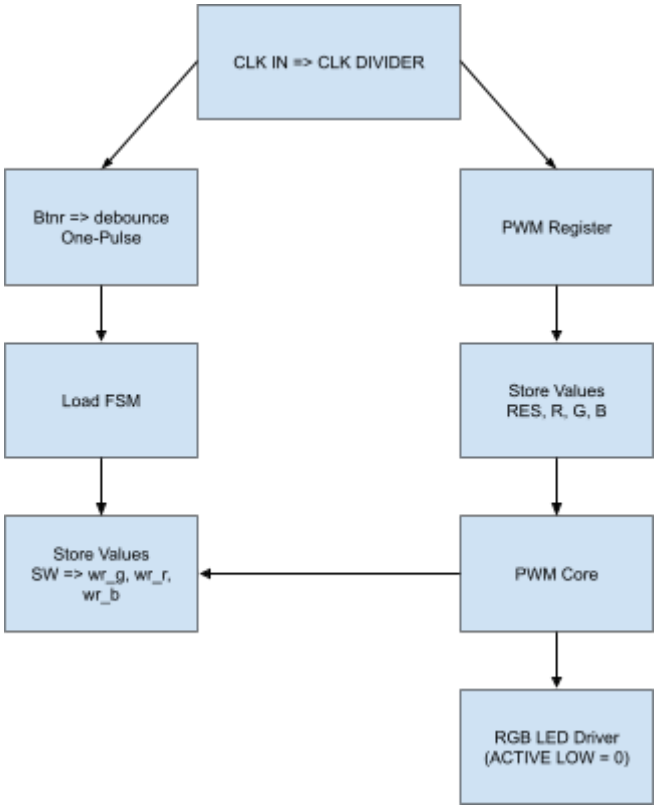
Top module:

```
module top_lab8(
    input wire clk100mhz,
    input wire btnc_n,
    input wire btnr,
    input wire [7:0] sw,
    output wire [3:0] led,
    output wire rgb_r, rgb_g, rgb_b
);
    wire rst_n = ~btnc_n;
    wire clk_1k, clk_pwm;

    clock_divider_fixed
u_div(.clk_in(clk100mhz), .rst_n(rst_n), .clk_1k(clk_1k), .clk_pwm(clk_pwm));
    wire load_pulse;
    debounce_onepulse #(.STABLE_TICKS(20))
u_db(.clk(clk_1k), .rst_n(rst_n), .din(btnr), .pulse(load_pulse));
    wire [1:0] slot;
    wire [3:0] slot_oh;
    wire wr_res, wr_r, wr_g, wr_b;
    load_fsm u_fsm(
        .clk(clk_1k), .rst_n(rst_n), .load_pulse(load_pulse),
        .slot(slot), .slot_onehot(slot_oh),
        .wr_res(wr_res), .wr_r(wr_r), .wr_g(wr_g), .wr_b(wr_b)
    );
    assign led = slot_oh;

    reg [7:0] reg_res, reg_r, reg_g, reg_b;
    always @(posedge clk_1k or negedge rst_n) begin
        if (!rst_n) begin reg_res<=8'd63; reg_r<=0; reg_g<=0; reg_b<=0; end
        else begin
            if (wr_res) reg_res <= sw;
            if (wr_r) reg_r <= sw;
            if (wr_g) reg_g <= sw;
            if (wr_b) reg_b <= sw;
        end
    end

    reg [7:0] res_q1,res_q2,r_q1,r_q2,g_q1,g_q2,b_q1,b_q2;
    always @(posedge clk_pwm or negedge rst_n) begin
        if (!rst_n) begin res_q1<=0; res_q2<=0; r_q1<=0; r_q2<=0; g_q1<=0;
g_q2<=0; b_q1<=0; b_q2<=0; end
        else begin
            res_q1<=reg_res; res_q2<=res_q1;
            r_q1<=reg_r; r_q2<=r_q1; g_q1<=reg_g; g_q2<=g_q1; b_q1<=reg_b;
b_q2<=b_q1;
        end
    end

    wire pwm_r, pwm_g, pwm_b;
    pwm_core u_pwm(.clk(clk_pwm), .rst_n(rst_n),
        .period(res_q2), .duty_r(r_q2), .duty_g(g_q2), .duty_b(b_q2),
        .pwm_r(pwm_r), .pwm_g(pwm_g), .pwm_b(pwm_b));

    rgb_led_driver #(.ACTIVE_LOW(0)) u_led(
        .pwm_r(pwm_r), .pwm_g(pwm_g), .pwm_b(pwm_b),
        .led_r(rgb_r), .led_g(rgb_g), .led_b(rgb_b));
endmodule
```

pwm_core_tb:



Block Diagram:



Video Link:
https://youtube.com/shorts/y-kL5k3PJXc?feature=share

Contributions:
Czyrone (50%) - Physical demo
Caleb (50%) - Testbench, Running the simulation
Both worked on the lab report together and troubleshooted code

Reflection on 4-slot loading and RES + 1:

The goal of this lab is to implement a 4-slot loading system to store separate 8-bit values for the PWM resolution (RES) and the red, green, and blue duty cycles (R, G, B). A load finite state machine (FSM), driven by a debounced one-pulse input, cycles through the four slots in sequence, with each press of the LOAD button storing the value on sw[7:0] into the active slot. Slot 0 holds the RES value, determining the PWM period, while slots 1 through 3 hold the duty cycle values for the red, green, and blue channels, respectively. The PWM core counts from 0 to RES + 1, producing a period that is one clock cycle longer than the RES register value to avoid off-by-one errors. This allows a RES value of 0xFF to generate a full-scale 256-step PWM cycle for precise brightness control.