Class Report 1: Reaction Timer

Cameron Anderson

May 12, 2019

1 Introduction

The goal of this assignment is to use Vivado Design Suite to program the Nexys 4 DDR board to create a reaction timer that displays in milliseconds on the seven-segment display how long it takes for a user to press a button after noticing a visual stimulus, in this case a lit LED. Additionally, the assignment should be able to be reset after each use and tell when the user has cheated by pressing the button before the visual stimulus has appeared.

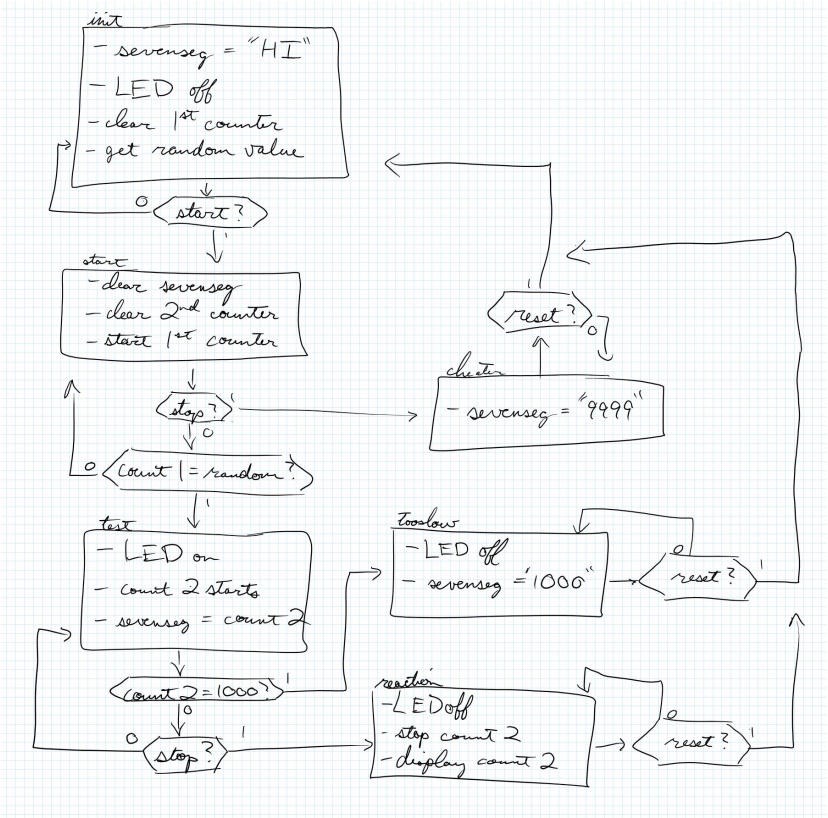
2 Experimental Plan

As seen in Figure 1 on page 2, the project needs a finite state machine to transition the different states of interaction with the device. Additionally, a clock divider is needed to return values in milliseconds to the user. This particular clock divider also features the state machine shown in Figure 2 on page 3 that loads a register with the predetermined value and counts down until half of a millisecond has passed. The clock divider then changes the millisecond clock output. To keep users from anticipating when the LED will light, a pseudo-random number generator is incorporated into the project. The LSFR module outputs a single random bit that is shifted each clock cycle into the counter register in the top module that determines the length of wait time until the LED lights. Two additional modules to drive the seven-segment display and to convert hexidecimal to seven-segment values are also required for this project.

3 Analysis

For readability sake, the code for this project is listed in the Appendix. Listing 1 in the Appendix shows the top level module that operates as the timer fsm. The necessary inputs and outputs that the user will need are the same in the top level module as in the board's constraint file. It is worth noting that the timer's main state machine sends simple signals to the seven-segment display mux when a simple message is to be displayed. This was added to easily change the seven-segment display in the top module. In Listing 2 in the Appendix, the

Figure 1: Main State Machine

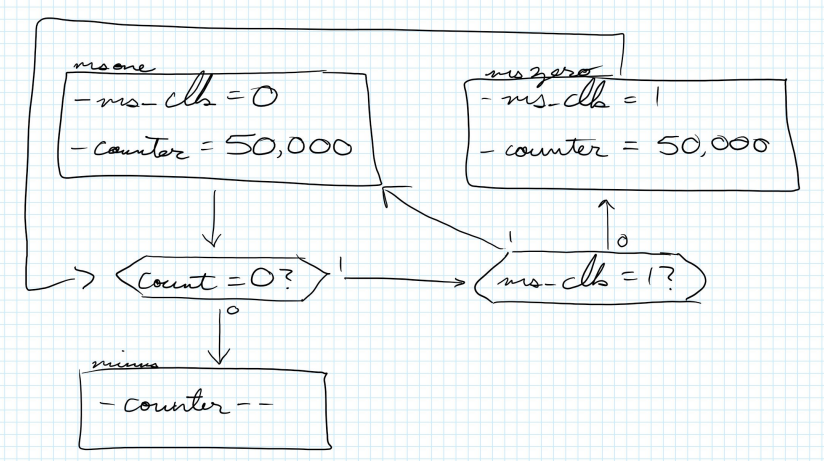


clock divider is shown. This module also uses a state machine to alternate the ms_clk bit every half of a millisecond to output a millisecond clock. Listing 3 in the Appendix shows the code for the random number generator. The output of this module is used to change the timing of the visual stimulus so that the user cannot anticipate the lighting of the LED. Listing 4 shows the code for the display mux, and Listing 5 shows the code that converts hexidecimal to seven-segment display values. Lastly, Listing 6 shows the constraints file used for the Nexys 4 DDR.

4 Conclusion

The use of state machines in projects such as this one makes interacting with devices much easier. The alternative would likely be a vast number of if-statement muxes which could slow the desired output. While it would still work in this case, timing critical projects are better served using state machines.

Figure 2: Clock Divider State Machine



5 Appendix

Listing 1: System Verilog code for the timer state machine top module

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//
// Cameron Anderson
//
//////////////////////////////////////////////////////////////////////////////////


module state_machine
    (
    input logic clk,
    input logic reset,
    input logic [1:0] btn,
    output logic [3:0] an,
    output logic [7:0] sseg,
    output logic ld[0]
    );

// fsm state type
typedef enum {init, starter, test, reaction, tooslow, cheater} state_type;

// declarations
state_type state_reg, state_next;
```

```
logic ms_clk;
logic r;
logic CHEAT; // initialize "9999" display signal
logic HI; // initialize "HI" display signal
logic ERR; // initialize "ERR" display signal
logic SLOW; // initialize sevenseg counter display
logic [15:0] count1; // initialize 1st counter memory
logic [15:0] count1_next; // initialize 1st counter signal
logic [15:0] count2; // initialize 2nd counter memory
logic [15:0] count2_next; // initialize 2nd counter signal
logic [12:0] random; // initialize random number to make the timer unpredictable
logic [12:0] rcount;
logic [27:0] seed;
logic start;
logic stop;
logic [7:0] led0, led1, led2, led3; // output from hex_to_sseg and input to disp
logic [3:0] hex0, hex1, hex2, hex3;

//instantiate clock divider
clock_divider clock_divider
    (
    .clk(clk),
    .reset(reset),
    .ms_clk(ms_clk)
    );

// instantiate Fibonacci random number generator
LFSR_fib168 fibonacci
    (
    .clk(clk),
    .reset(reset),
    .seed(seed),
    .r(r)
    );

// instantiate 7-seg LED display time-multiplexing module
disp_mux disp_unit
    (
    .clk(clk),
    .reset(reset),
    .SLOW(SLOW),
    .HI(HI),
    .ERR(ERR),
    .CHEATER(CHEAT),
    .in0(led0), .in1(led1), .in2(led2), .in3(led3),
    .an(an),
```

```
    .sseg(sseg)
    );

// instantiate 1st seven segment digit module
hex_to_sseg sseg_unit_0
    (
    .hex(hex0),
    .decimal_point(1'b1),
    .sseg(led0)
    );

//instantiate 2nd seven segment digit module
hex_to_sseg sseg_unit_1
    (
    .hex(hex1),
    .decimal_point(1'b0),
    .sseg(led1)
    );

// instantiate 3rd seven segment digit module
hex_to_sseg sseg_unit_2
    (
    .hex(hex2),
    .decimal_point(1'b1),
    .sseg(led2)
    );

// instantiate 3rd seven segment digit module
hex_to_sseg sseg_unit_3
    (
    .hex(hex3),
    .decimal_point(1'b0),
    .sseg(led3)
    );

// memory logic for fsm
/*always_ff @(posedge clk, posedge reset)
    if (reset)
    begin

    end
    else // enable state machine
    begin

    end*/
```

```
// memory logic for counter
always_ff @ (posedge ms_clk, posedge reset)
    if (reset)
    begin
        state_reg <= init; // start
        count1 <= 16'b0000000000000000; // reset the 1st counter
        count2 <= 16'b0000000000000000; // reset the 2nd counter
    end
    else
    begin
        state_reg <= state_next;
        count1 <= count1_next;
        count2 <= count2_next;
        random[12:10] <= {random[11:10],r};
    end

always_comb
begin
    state_next = state_reg;
    case (state_reg)
        init:
        begin
            HI = 1'b1; // turn on "HI"
            ld[0] = 1'b0; // turn LED off
            count1_next = 0; // clear 1st counter
            if (start) // start button is pressed
            begin
                HI = 1'b0; // turn off "HI"
                rcount = random;
                state_next = starter;
            end
        end
        starter:
        begin
            count2_next = 0; // clear the 2nd counter
            count1_next = count1_next + 1; // start 1st counter
            if (stop) // used to pickup a false start
            begin
                count1_next = count1_next; // stop the 1st counter
                state_next = cheater; // move to the next state
            end
            else if (count1 == rcount) // used to start the timer at a random ti
            begin
                count1_next = count1_next; // stop the 1st counter
                state_next = test; // move to the next state
            end
```

```
        end
        test:
        begin
            ld[0] = 1'b1; // turn on stimulus LED
            count2_next = count2_next + 1; // start 2nd counter
            if (count2 == 1000)
            begin
                count2_next = count2_next; // stop 2nd counter
                state_next = tooslow; // move to the next state
            end
            else if (stop)
            begin
                count2_next = count2_next; // stop 2nd counter
                state_next = reaction; // move to the next state
            end
        end
        reaction:
        begin
            ld[0] = 1'b0; // turn off LED
            hex0 = count2[3:0]; // display time from the 2nd counter
            hex1 = count2[7:4]; // display time from the 2nd counter
            hex2 = count2[11:8]; // display time from the 2nd counter
            hex3 = count2[15:12]; // display time from the 2nd counter
            if (reset) // reset the timer
            begin
                state_next = init;
            end
        end
        tooslow:
        begin
            ld[0] = 1'b0; // turn on stimulus LED
            SLOW = 1'b1; // turn on "1000"
            if (reset) // reset the timer
            begin
                SLOW = 1'b0; // turn off "1000"
                state_next = init;
            end
        end
        cheater:
        begin
            CHEAT = 1'b1; // turn on "9999"
            if (reset) // reset the timer
            begin
                CHEAT = 1'b0; // turn off "9999"
                state_next = init;
            end
```

```
        end
        default: // turn on "ERR"
        begin
            ERR = 1'b1;
        end
    endcase
end

assign start = btn[0];
assign stop = btn[1];

endmodule
```

Listing 2: System Verilog code for the clock divider state machine

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//
// Cameron Anderson
//
//////////////////////////////////////////////////////////////////////////////////

module clock_divider
    (
    input logic clk, // input of 100MHz system clock
    input logic reset, // input to reset the counter
    output logic ms_clk // outputs 4 digits to represent whichever fraction of a
    );

    typedef enum {minus, ms_zero, ms_one} state_type;

    // declarations
    state_type state_reg, state_next;
    logic [15:0] count;
    logic [15:0] count_next;

    always_ff @(posedge clk, posedge reset)
        if (reset)
        begin
            state_reg <= ms_zero;
        end
        else
        begin
            state_reg <= state_next;
            count <= count_next;
        end
```

```
    always_comb
    begin
        state_next = state_reg;
        case (state_reg)
            ms_one:
            begin
                ms_clk = 0;
                count_next = 16'd50000;
                state_next = minus;
            end
            ms_zero:
            begin
                ms_clk = 1;
                count_next = 16'd50000;
                state_next = minus;
            end
            minus:
            begin
                if (count == 0 && ms_clk == 0)
                begin
                    state_next = ms_zero;
                end
                else if (count == 0 && ms_clk == 1)
                begin
                    state_next = ms_one;
                end
                else
                begin
                    count_next = count_next - 1;
                end
            end
            default:
            begin

            end // error
        endcase
    end

endmodule
```

Listing 3: System Verilog code for the random number generator

```
`timescale 1ns / 1ps

// LFSR Fibonacci 168 bit pseudorandom number generator
```

```
//      maximal poly for 168 is 168,166,153,151
//      don't forget to subtract 1 since we are 0-167
//      key1-4 are binary digits of pi used to add initial informatio
//      Note this is pseudorandom, and linear.  It won't repeat for a long time
//      but it is predictable after a much shorter time so it should not be used
//      for apps needing true randomness, such as cryptography, gambling, etc.

module LFSR_fib168
(
  input logic clk,
  input logic reset,
  input logic [27:0] seed,
  output logic r // r is either 1 or 0 every clock cycle
);

  logic [167:0] state, next_state;
  logic polynomial;

  assign polynomial = state[167]^state[165]^state[152]^state[151];
  assign next_state = {state[166:0],polynomial};

  always_ff @(posedge(clk), posedge(reset))
    if(reset)
      state <= {seed,~seed,seed,seed,~seed,~seed};
    else
      state <= next_state;

  assign r = polynomial;

  endmodule
```

Listing 4: System Verilog code for the seven-segment display mux

```
// Listing 4.15
module disp_mux
   (
    input  logic clk,
    input  logic reset,
    input  logic SLOW,
    input  logic HI, // input to display "HI" on the seven segment display
    input  logic ERR, // input to display "ERR" on the seven segment display
    input  logic CHEATER, // input to display "9999" on the seven segment displa
    input  logic [7:0] in3, in2, in1, in0, // in 0 is the 8-digit binary combina
    output logic [3:0] an,   // anode 0 is the rightmost digit on the board (8 t
    output logic [7:0] sseg  // this output mirrors the in input to light the sp
   );
```

```
localparam N = 18; // refreshing rate around 800 Hz (50 MHz/2^16)

// signal declaration
logic [N-1:0] q_reg;
logic [N-1:0] q_next;

// N-bit counter
// register
always_ff @(posedge clk,  posedge reset)
   if (reset)
     begin
         q_reg <= 0;
     end
   else
     begin
         q_reg <= q_next;
     end

// next-state logic
assign q_next = q_reg + 1;

// 2 MSBs of counter to control 4-to-1 multiplexing
// and to generate active-low enable signal
always_comb
   case (q_reg[N-1:N-2])
      2'b00:
         begin
            an = 4'b1110;
            if (HI == 1'b1)
             begin
                 sseg = 8'b11111001; // I
             end
            else if (ERR == 1'b1)
             begin
                 sseg = 8'b10001000; // R
             end
            else if (SLOW == 1'b1)
             begin
                 sseg = 8'b11000000; // 0
             end
            else if (CHEATER == 1'b1)
             begin
                 sseg = 8'b10010000; // 9
             end
            else
```

```
            begin
                sseg = in0;
            end
        end
    2'b01:
        begin
            an =  4'b1101;
            if (HI == 1'b1)
            begin
                sseg = 8'b10001001; // H
            end
            else if (ERR == 1'b1)
            begin
                sseg = 8'b10001000; // R
            end
            else if (SLOW == 1'b1)
              begin
                sseg = 8'b11000000; // 0
              end
            else if (CHEATER == 1'b1)
              begin
                sseg = 8'b10010000; // 9
              end
            else
            begin
                sseg = in1;
            end
        end
    2'b10:
        begin
            an =  4'b1011;
            if (HI == 1'b1)
            begin
                sseg = 8'b11111111; // blank
            end
            else if (ERR == 1'b1)
            begin
                sseg = 8'b10000110; // E
            end
            else if (SLOW == 1'b1)
              begin
                sseg = 8'b11000000; // 0
              end
            else if (CHEATER == 1'b1)
              begin
                sseg = 8'b10010000; // 9
```

```
                end
            else
             begin
                sseg = in2;
             end
        end
      default:
        begin
          an =  4'b0111;
          if (HI == 1'b1)
           begin
                sseg = 8'b11111111; // blank
           end
          else if (ERR == 1'b1)
           begin
                sseg = 8'b11111111; // blank
           end
          else if (SLOW == 1'b1)
            begin
                 sseg = 8'b11111001; // 1
            end
          else if (CHEATER == 1'b1)
           begin
                sseg = 8'b10010000; // 9
           end
          else
           begin
                sseg = in3;
           end
        end
    endcase
endmodule
```

Listing 5: System Verilog code for the hexidecimal to seven-segment converter

```
// Listing 3.14
module hex_to_sseg
   (
    input  logic [3:0] hex,
    input  logic decimal_point,
    output logic [7:0] sseg  // output active low
   );

   always_comb
   begin
      case(hex)
```

```
            4'h0: sseg[6:0] = 7'b1000000;
            4'h1: sseg[6:0] = 7'b1111001;
            4'h2: sseg[6:0] = 7'b0100100;
            4'h3: sseg[6:0] = 7'b0110000;
            4'h4: sseg[6:0] = 7'b0011001;
            4'h5: sseg[6:0] = 7'b0010010;
            4'h6: sseg[6:0] = 7'b0000010;
            4'h7: sseg[6:0] = 7'b1111000;
            4'h8: sseg[6:0] = 7'b0000000;
            4'h9: sseg[6:0] = 7'b0010000;
            4'ha: sseg[6:0] = 7'b0001000;
            4'hb: sseg[6:0] = 7'b0000011;
            4'hc: sseg[6:0] = 7'b1000110;
            4'hd: sseg[6:0] = 7'b0100001;
            4'he: sseg[6:0] = 7'b0000110;
            default: sseg[6:0] = 7'b0001110;   //4'hf
        endcase
        sseg[7] = decimal_point;
    end
endmodule
```

Listing 6: System Verilog code for the Nexys 4 DDR constraints

```
## This file is a general .xdc for the Nexys4 DDR Rev. C
## 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

##================================================================================
## Clock signal
##================================================================================
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_

##================================================================================
## To facilitate Quad-SPI flash programming
##================================================================================
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]



##================================================================================
##Switches
##================================================================================
##set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {sw[0]}]
```

```
#set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {sw[1]}]
#set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {sw[2]}]
#set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {sw[3]}]
#set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {sw[4]}]
#set_property -dict {PACKAGE_PIN T18 IOSTANDARD LVCMOS33} [get_ports {sw[5]}]
#set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {sw[6]}]
#set_property -dict {PACKAGE_PIN R13 IOSTANDARD LVCMOS33} [get_ports {sw[7]}]
#set_property -dict {PACKAGE_PIN T8 IOSTANDARD LVCMOS33} [get_ports {sw[8]}]
#set_property -dict {PACKAGE_PIN U8 IOSTANDARD LVCMOS33} [get_ports {sw[9]}]
#set_property -dict {PACKAGE_PIN R16 IOSTANDARD LVCMOS33} [get_ports {sw[10]}]
#set_property -dict {PACKAGE_PIN T13 IOSTANDARD LVCMOS33} [get_ports {sw[11]}]
#set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {sw[12]}]
#set_property -dict {PACKAGE_PIN U12 IOSTANDARD LVCMOS33} [get_ports {sw[13]}]
#set_property -dict {PACKAGE_PIN U11 IOSTANDARD LVCMOS33} [get_ports {sw[14]}]
#set_property -dict {PACKAGE_PIN V10 IOSTANDARD LVCMOS33} [get_ports {sw[15]}]

##=============================================================================
#Buttons
##=============================================================================
# ***** CPU_reset button; active low
set_property -dict {PACKAGE_PIN C12 IOSTANDARD LVCMOS33} [get_ports reset]
# ***** Other buttons; active high
# ***** btn(0): btnu;  btn(1): btnr;  btn(2): btnd; btn(3): btnl;
btn(4): btnc;
#set_property -dict {PACKAGE_PIN N17 IOSTANDARD LVCMOS33} [get_ports {btn[4]}]
set_property -dict {PACKAGE_PIN M18 IOSTANDARD LVCMOS33} [get_ports {btn[0]}]
#set_property -dict {PACKAGE_PIN P17 IOSTANDARD LVCMOS33} [get_ports {btn[3]}]
set_property -dict {PACKAGE_PIN M17 IOSTANDARD LVCMOS33} [get_ports {btn[1]}]
#set_property -dict {PACKAGE_PIN P18 IOSTANDARD LVCMOS33} [get_ports {btn[2]}]

##=============================================================================
# discrete LEDs
##=============================================================================
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {ld[0]}]
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {ld[1]}]
set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {ld[2]}]
set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports {ld[3]}]
set_property -dict {PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports {ld[4]}]
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {ld[5]}]
set_property -dict {PACKAGE_PIN U17 IOSTANDARD LVCMOS33} [get_ports {ld[6]}]
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {ld[7]}]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {ld[8]}]
set_property -dict {PACKAGE_PIN T15 IOSTANDARD LVCMOS33} [get_ports {ld[9]}]
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {ld[10]}]
set_property -dict {PACKAGE_PIN T16 IOSTANDARD LVCMOS33} [get_ports {ld[11]}]
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {ld[12]}]
```

```
set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {ld[13]}]
set_property -dict {PACKAGE_PIN V12 IOSTANDARD LVCMOS33} [get_ports {ld[14]}]
set_property -dict {PACKAGE_PIN V11 IOSTANDARD LVCMOS33} [get_ports {ld[15]}]

##-----------------------------------------------------------------------
## tri-color LEDs
##-----------------------------------------------------------------------
#set_property -dict {PACKAGE_PIN R12 IOSTANDARD LVCMOS33} [get_ports {rgb_led1[0
#set_property -dict {PACKAGE_PIN M16 IOSTANDARD LVCMOS33} [get_ports {rgb_led1[1
#set_property -dict {PACKAGE_PIN N15 IOSTANDARD LVCMOS33} [get_ports {rgb_led1[2
#set_property -dict {PACKAGE_PIN G14 IOSTANDARD LVCMOS33} [get_ports {rgb_led2[0
#set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVCMOS33} [get_ports {rgb_led2[1
#set_property -dict {PACKAGE_PIN N16 IOSTANDARD LVCMOS33} [get_ports {rgb_led2[2

##-----------------------------------------------------------------------
##7 segment display
##-----------------------------------------------------------------------
set_property -dict {PACKAGE_PIN T10 IOSTANDARD LVCMOS33} [get_ports {sseg[0]}]
set_property -dict {PACKAGE_PIN R10 IOSTANDARD LVCMOS33} [get_ports {sseg[1]}]
set_property -dict {PACKAGE_PIN K16 IOSTANDARD LVCMOS33} [get_ports {sseg[2]}]
set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33} [get_ports {sseg[3]}]
set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports {sseg[4]}]
set_property -dict {PACKAGE_PIN T11 IOSTANDARD LVCMOS33} [get_ports {sseg[5]}]
set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {sseg[6]}]
#decimal point
set_property -dict {PACKAGE_PIN H15 IOSTANDARD LVCMOS33} [get_ports {sseg[7]}]
# enable
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {an[0]}]
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {an[1]}]
set_property -dict {PACKAGE_PIN T9 IOSTANDARD LVCMOS33} [get_ports {an[2]}]
set_property -dict {PACKAGE_PIN J14 IOSTANDARD LVCMOS33} [get_ports {an[3]}]
#set_property -dict {PACKAGE_PIN P14 IOSTANDARD LVCMOS33} [get_ports {an[4]}]
#set_property -dict {PACKAGE_PIN T14 IOSTANDARD LVCMOS33} [get_ports {an[5]}]
#set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVCMOS33} [get_ports {an[6]}]
#set_property -dict {PACKAGE_PIN U13 IOSTANDARD LVCMOS33} [get_ports {an[7]}]



##-----------------------------------------------------------------------
## PWM Audio Amplifier
##-----------------------------------------------------------------------
#set_property -dict {PACKAGE_PIN A11 IOSTANDARD LVCMOS33} [get_ports audio_pdm]
#set_property -dict {PACKAGE_PIN D12 IOSTANDARD LVCMOS33} [get_ports audio_on]



##-----------------------------------------------------------------------
## USB-RS232 Interface
```

```
#------------------------------------------------------------------------
#set_property -dict {PACKAGE_PIN D4 IOSTANDARD LVCMOS33} [get_ports tx]
#set_property -dict {PACKAGE_PIN C4 IOSTANDARD LVCMOS33} [get_ports rx]
# ************CTS/RTS not used
#set_property -dict { PACKAGE_PIN D3    IOSTANDARD LVCMOS33 } [get_ports { uart_
#set_property -dict { PACKAGE_PIN E5    IOSTANDARD LVCMOS33 } [get_ports { uart_

#------------------------------------------------------------------------
# USB HID (PS/2)
#------------------------------------------------------------------------
#set_property -dict {PACKAGE_PIN F4 IOSTANDARD LVCMOS33} [get_ports ps2c]
#set_property -dict {PACKAGE_PIN B2 IOSTANDARD LVCMOS33} [get_ports ps2d]



#------------------------------------------------------------------------
# I2C temperature sensor
# tmp_int / tmp_ct signals are not used
#------------------------------------------------------------------------
#set_property -dict {PACKAGE_PIN C14 IOSTANDARD LVCMOS33} [get_ports tmp_i2c_scl
#set_property -dict {PACKAGE_PIN C15 IOSTANDARD LVCMOS33} [get_ports tmp_i2c_sda
# ************ tmp_int / tmp_ct not used
#set_property -dict { PACKAGE_PIN D13   IOSTANDARD LVCMOS33 } [get_ports { tmp_i
#set_property -dict { PACKAGE_PIN B14   IOSTANDARD LVCMOS33 } [get_ports { tmp_c



#------------------------------------------------------------------------
# SPI Accelerometer
# aclInt1 / aclInt2 signals are not used
#------------------------------------------------------------------------
#set_property -dict {PACKAGE_PIN E15 IOSTANDARD LVCMOS33} [get_ports acl_miso]
#set_property -dict {PACKAGE_PIN F14 IOSTANDARD LVCMOS33} [get_ports acl_mosi]
#set_property -dict {PACKAGE_PIN F15 IOSTANDARD LVCMOS33} [get_ports acl_sclk]
#set_property -dict {PACKAGE_PIN D15 IOSTANDARD LVCMOS33} [get_ports acl_ss_n]
# ************ aclInt1 / aclInt2 signals are not used
#set_property -dict { PACKAGE_PIN B13   IOSTANDARD LVCMOS33 } [get_ports { aclIn
#set_property -dict { PACKAGE_PIN C16   IOSTANDARD LVCMOS33 } [get_ports { aclIn



#------------------------------------------------------------------------
# VGA Port
#------------------------------------------------------------------------
#set_property -dict {PACKAGE_PIN B11 IOSTANDARD LVCMOS33} [get_ports hsync]
#set_property -dict {PACKAGE_PIN B12 IOSTANDARD LVCMOS33} [get_ports vsync]
#set_property -dict {PACKAGE_PIN A3 IOSTANDARD LVCMOS33} [get_ports {rgb[8]}]
#set_property -dict {PACKAGE_PIN B4 IOSTANDARD LVCMOS33} [get_ports {rgb[9]}]
#et_property -dict {PACKAGE_PIN C5 IOSTANDARD LVCMOS33} [get_ports {rgb[10]}]
```

```
#set_property -dict {PACKAGE_PIN A4 IOSTANDARD LVCMOS33} [get_ports {rgb[11]}]
#set_property -dict {PACKAGE_PIN C6 IOSTANDARD LVCMOS33} [get_ports {rgb[4]}]
#set_property -dict {PACKAGE_PIN A5 IOSTANDARD LVCMOS33} [get_ports {rgb[5]}]
#set_property -dict {PACKAGE_PIN B6 IOSTANDARD LVCMOS33} [get_ports {rgb[6]}]
#set_property -dict {PACKAGE_PIN A6 IOSTANDARD LVCMOS33} [get_ports {rgb[7]}]
#set_property -dict {PACKAGE_PIN B7 IOSTANDARD LVCMOS33} [get_ports {rgb[0]}]
#set_property -dict {PACKAGE_PIN C7 IOSTANDARD LVCMOS33} [get_ports {rgb[1]}]
#set_property -dict {PACKAGE_PIN D7 IOSTANDARD LVCMOS33} [get_ports {rgb[2]}]
#set_property -dict {PACKAGE_PIN D8 IOSTANDARD LVCMOS33} [get_ports {rgb[3]}]

#============================================================================
#micro SD card Connector
#============================================================================
#set_property -dict {PACKAGE_PIN E2 IOSTANDARD LVCMOS33} [get_ports sd_reset]
#set_property -dict {PACKAGE_PIN B1 IOSTANDARD LVCMOS33} [get_ports sd_sclk]
#set_property -dict {PACKAGE_PIN C1 IOSTANDARD LVCMOS33} [get_ports sd_mosi]
#set_property -dict {PACKAGE_PIN C2 IOSTANDARD LVCMOS33} [get_ports sd_miso]
#set_property -dict { PACKAGE_PIN E1    IOSTANDARD LVCMOS33 } [get_ports { SD_DA
#set_property -dict { PACKAGE_PIN F1    IOSTANDARD LVCMOS33 } [get_ports { SD_DA
#set_property -dict {PACKAGE_PIN D2 IOSTANDARD LVCMOS33} [get_ports sd_ss_n]
#set_property -dict { PACKAGE_PIN A1    IOSTANDARD LVCMOS33 } [get_ports { SD_CD
//** card detect not used

#set_property -dict { PACKAGE_PIN E2    IOSTANDARD LVCMOS33 } [get_ports { SD_RE
#set_property -dict { PACKAGE_PIN A1    IOSTANDARD LVCMOS33 } [get_ports { SD_CD
#set_property -dict { PACKAGE_PIN B1    IOSTANDARD LVCMOS33 } [get_ports { SD_SC
#set_property -dict { PACKAGE_PIN C1    IOSTANDARD LVCMOS33 } [get_ports { SD_CM
#set_property -dict { PACKAGE_PIN C2    IOSTANDARD LVCMOS33 } [get_ports { SD_DA
#set_property -dict { PACKAGE_PIN E1    IOSTANDARD LVCMOS33 } [get_ports { SD_DA
#set_property -dict { PACKAGE_PIN F1    IOSTANDARD LVCMOS33 } [get_ports { SD_DA
#set_property -dict { PACKAGE_PIN D2    IOSTANDARD LVCMOS33 } [get_ports { SD_DA
#----------SPI mode pin map-------------------------------------------------
#Clock    [SCK]
#CMD/DI   [MOSI]
#DAT0/D0  [MISO]
#DAT1/IRQ Unused/I
#DAT2/NC  Unused
#DAT3/CS  [SS]
#VSS1     Ground
#Vss2     Ground
#Vdd      voltage
#============================================================================

#============================================================================
# PMOD xadc (JXADC)
# originally with LVDS (error for adcn/p[3]); not LVCOMS33
```

```
#============================================================
#set_property -dict {PACKAGE_PIN A14 IOSTANDARD LVCMOS33} [get_ports {adc_n[0]}]
#set_property -dict {PACKAGE_PIN A13 IOSTANDARD LVCMOS33} [get_ports {adc_p[0]}]
#set_property -dict {PACKAGE_PIN A16 IOSTANDARD LVCMOS33} [get_ports {adc_n[1]}]
#set_property -dict {PACKAGE_PIN A15 IOSTANDARD LVCMOS33} [get_ports {adc_p[1]}]
#set_property -dict {PACKAGE_PIN B17 IOSTANDARD LVCMOS33} [get_ports {adc_n[2]}]
#set_property -dict {PACKAGE_PIN B16 IOSTANDARD LVCMOS33} [get_ports {adc_p[2]}]
#set_property -dict {PACKAGE_PIN A18 IOSTANDARD LVCMOS33} [get_ports {adc_n[3]}]
#set_property -dict {PACKAGE_PIN B18 IOSTANDARD LVCMOS33} [get_ports {adc_p[3]}]



#============================================================
#Pmod Header JA
# 4 out of 10 pins are for Vcc/Gnd
# use two signals (for top and bottom rows) to maintain pin#
#============================================================
#set_property -dict {PACKAGE_PIN C17 IOSTANDARD LVCMOS33} [get_ports {ja_top[1]}
#set_property -dict {PACKAGE_PIN D18 IOSTANDARD LVCMOS33} [get_ports {ja_top[2]}
#set_property -dict {PACKAGE_PIN E18 IOSTANDARD LVCMOS33} [get_ports {ja_top[3]}
#set_property -dict {PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports {ja_top[4]}
#set_property -dict {PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports {ja_btm[7]}
#set_property -dict {PACKAGE_PIN E17 IOSTANDARD LVCMOS33} [get_ports {ja_btm[8]}
#set_property -dict {PACKAGE_PIN F18 IOSTANDARD LVCMOS33} [get_ports {ja_btm[9]}
#set_property -dict {PACKAGE_PIN G18 IOSTANDARD LVCMOS33} [get_ports {ja_btm[10]

#============================================================
#Pmod Header JB
# 4 out of 10 pins are for Vcc/Gnd
# use two signals (for top and bottom rows) to maintain pin#
#============================================================
#set_property -dict {PACKAGE_PIN D14 IOSTANDARD LVCMOS33} [get_ports {jb_top[1]}
#set_property -dict {PACKAGE_PIN F16 IOSTANDARD LVCMOS33} [get_ports {jb_top[2]}
#set_property -dict {PACKAGE_PIN G16 IOSTANDARD LVCMOS33} [get_ports {jb_top[3]}
#set_property -dict {PACKAGE_PIN H14 IOSTANDARD LVCMOS33} [get_ports {jb_top[4]}
#set_property -dict {PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports {jb_btm[7]}
#set_property -dict {PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports {jb_btm[8]}
#set_property -dict {PACKAGE_PIN G13 IOSTANDARD LVCMOS33} [get_ports {jb_btm[9]}
#set_property -dict {PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports {jb_btm[10]

#============================================================
#Pmod Header JC
# 4 out of 10 pins are for Vcc/Gnd
# use two signals (for top and bottom rows) to maintain pin#
#============================================================
#set_property -dict {PACKAGE_PIN K1 IOSTANDARD LVCMOS33} [get_ports {jc_top[1]}]
```

```
#set_property -dict {PACKAGE_PIN F6 IOSTANDARD LVCMOS33} [get_ports {jc_top[2]}]
#set_property -dict {PACKAGE_PIN J2 IOSTANDARD LVCMOS33} [get_ports {jc_top[3]}]
#set_property -dict {PACKAGE_PIN G6 IOSTANDARD LVCMOS33} [get_ports {jc_top[4]}]
#set_property -dict {PACKAGE_PIN E7 IOSTANDARD LVCMOS33} [get_ports {jc_btm[7]}]
#set_property -dict {PACKAGE_PIN J3 IOSTANDARD LVCMOS33} [get_ports {jc_btm[8]}]
#set_property -dict {PACKAGE_PIN J4 IOSTANDARD LVCMOS33} [get_ports {jc_btm[9]}]
#set_property -dict {PACKAGE_PIN E6 IOSTANDARD LVCMOS33} [get_ports {jc_btm[10]}


##Pmod Header JD
## 4 out of 10 pins are for Vcc/Gnd
## use two signals (for top and bottom rows) to maintain pin#
##set_property -dict {PACKAGE_PIN H4 IOSTANDARD LVCMOS33} [get_ports {jd_top[1]}]
##set_property -dict {PACKAGE_PIN H1 IOSTANDARD LVCMOS33} [get_ports {jd_top[2]}]
##set_property -dict {PACKAGE_PIN G1 IOSTANDARD LVCMOS33} [get_ports {jd_top[3]}]
##set_property -dict {PACKAGE_PIN G3 IOSTANDARD LVCMOS33} [get_ports {jd_top[4]}]
##et_property -dict {PACKAGE_PIN H2 IOSTANDARD LVCMOS33} [get_ports {jd_btm[7]}]
##set_property -dict {PACKAGE_PIN G4 IOSTANDARD LVCMOS33} [get_ports {jd_btm[8]}]
##set_property -dict {PACKAGE_PIN G2 IOSTANDARD LVCMOS33} [get_ports {jd_btm[9]}]
##set_property -dict {PACKAGE_PIN F3 IOSTANDARD LVCMOS33} [get_ports {jd_btm[10]}]

##
## **** not used in the FPro system
##

##Omnidirectional Microphone
#set_property -dict { PACKAGE_PIN J5    IOSTANDARD LVCMOS33 } [get_ports { M_CLK
#set_property -dict { PACKAGE_PIN H5    IOSTANDARD LVCMOS33 } [get_ports { M_DAT
#set_property -dict { PACKAGE_PIN F5    IOSTANDARD LVCMOS33 } [get_ports { M_LRS


##Quad SPI Flash

#set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCMOS33 } [get_ports { QSPI_I
#set_property -dict { PACKAGE_PIN K18   IOSTANDARD LVCMOS33 } [get_ports { QSPI_I
#set_property -dict { PACKAGE_PIN L14   IOSTANDARD LVCMOS33 } [get_ports { QSPI_I
#set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 } [get_ports { QSPI_I
#set_property -dict { PACKAGE_PIN L13   IOSTANDARD LVCMOS33 } [get_ports { QSPI_C


##SMSC Ethernet PHY

#set_property -dict { PACKAGE_PIN C9    IOSTANDARD LVCMOS33 } [get_ports { ETH_M
#set_property -dict { PACKAGE_PIN A9    IOSTANDARD LVCMOS33 } [get_ports { ETH_M
```

```
#set_property -dict { PACKAGE_PIN B3    IOSTANDARD LVCMOS33 } [get_ports { ETH_R
#set_property -dict { PACKAGE_PIN D9    IOSTANDARD LVCMOS33 } [get_ports { ETH_C
#set_property -dict { PACKAGE_PIN C10   IOSTANDARD LVCMOS33 } [get_ports { ETH_R
#set_property -dict { PACKAGE_PIN C11   IOSTANDARD LVCMOS33 } [get_ports { ETH_R
#set_property -dict { PACKAGE_PIN D10   IOSTANDARD LVCMOS33 } [get_ports { ETH_R
#set_property -dict { PACKAGE_PIN B9    IOSTANDARD LVCMOS33 } [get_ports { ETH_T
#set_property -dict { PACKAGE_PIN A10   IOSTANDARD LVCMOS33 } [get_ports { ETH_T
#set_property -dict { PACKAGE_PIN A8    IOSTANDARD LVCMOS33 } [get_ports { ETH_T
#set_property -dict { PACKAGE_PIN D5    IOSTANDARD LVCMOS33 } [get_ports { ETH_R
#set_property -dict { PACKAGE_PIN B8    IOSTANDARD LVCMOS33 } [get_ports { ETH_I
```