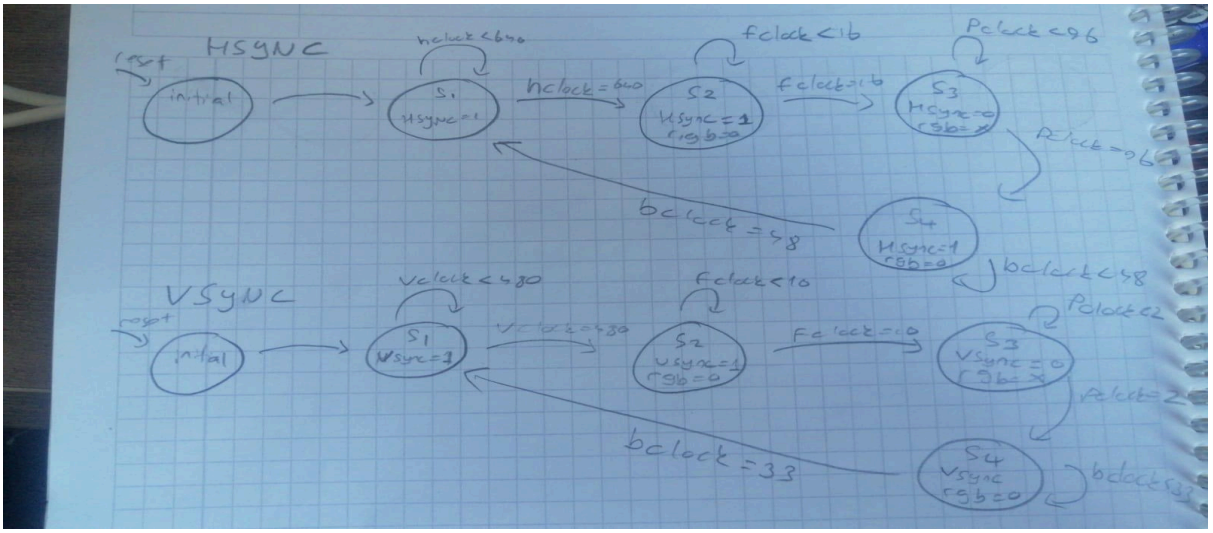
# CS223 Digital Design-Section 2
# Final Project Report

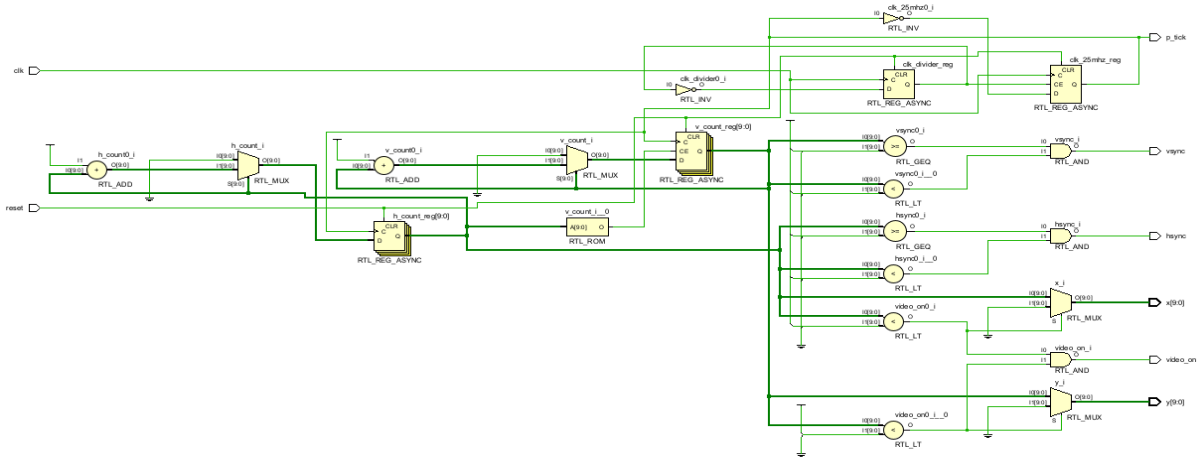
# Name: Onur Ege Yakar
# ID: 22202950

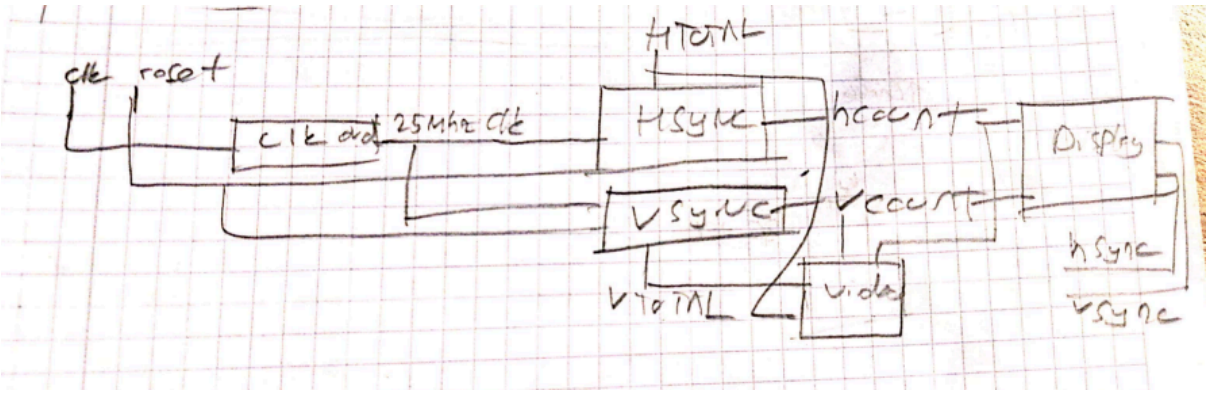
# Date: 13/12/2024

# VGA CONTROLLER

The vga controller module sets the pixels that are given in the pdf as; display, front porch, back porch, sync area and total of those which is for both vertical and horizontal. Sets a clock divider that 25mhz after that under the posedge of 25 mhz clock it controls the registers that is set 10 for both for vertical and horizontal called as count. After that it syncs hsync and vsync according to the given registers and pixels.

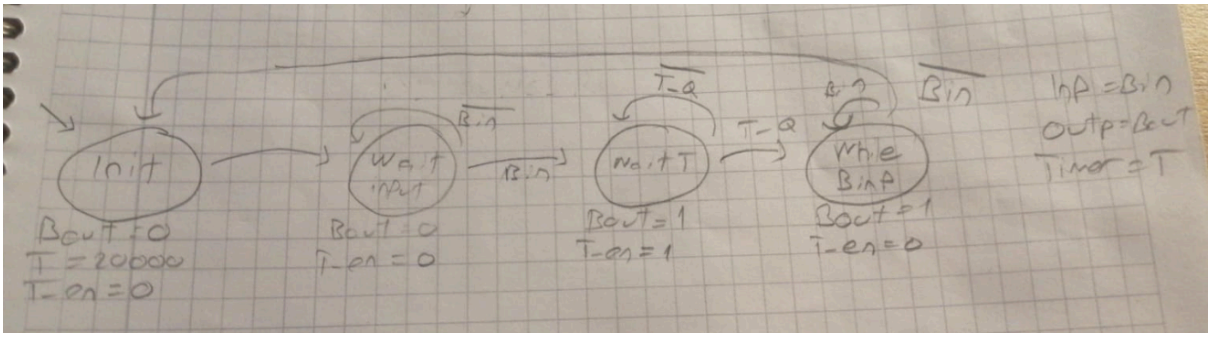


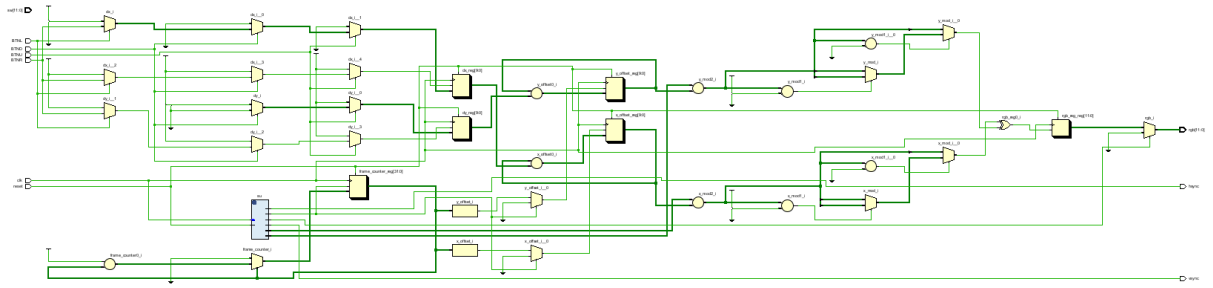



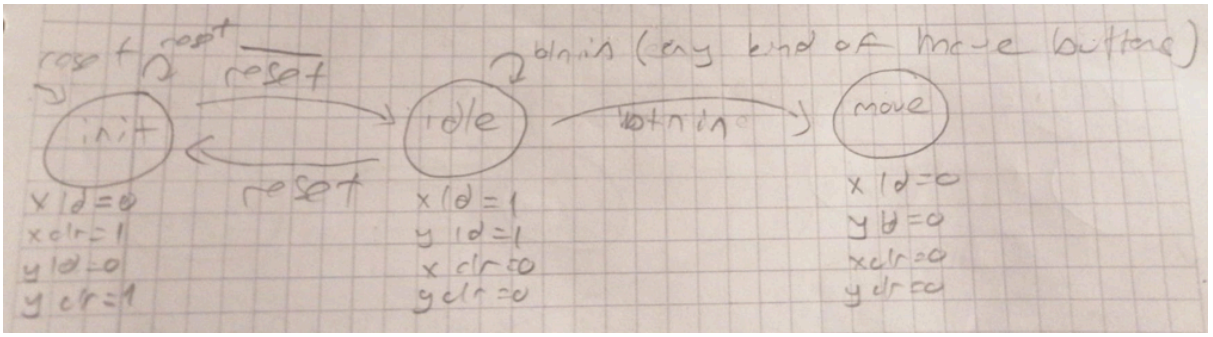



# CURSOR CONTROL

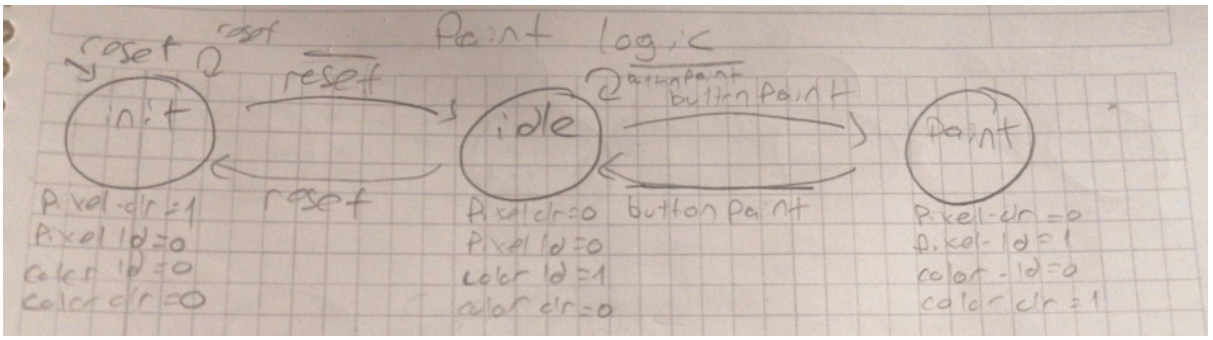
There is a little debouncing logic that allows the cursor to move precisely and having a different clock than the painting logic so that these two do not have any confusion.

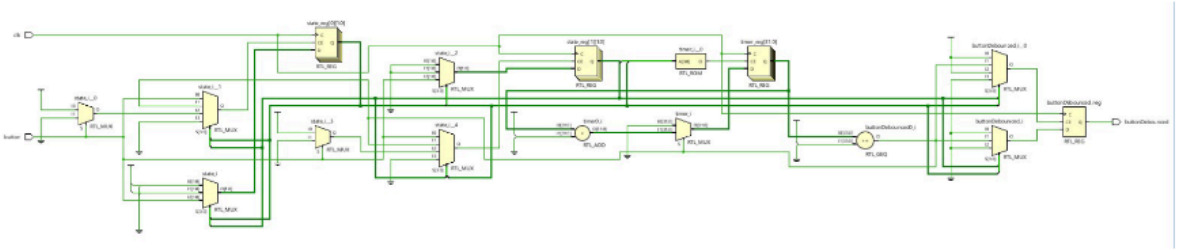


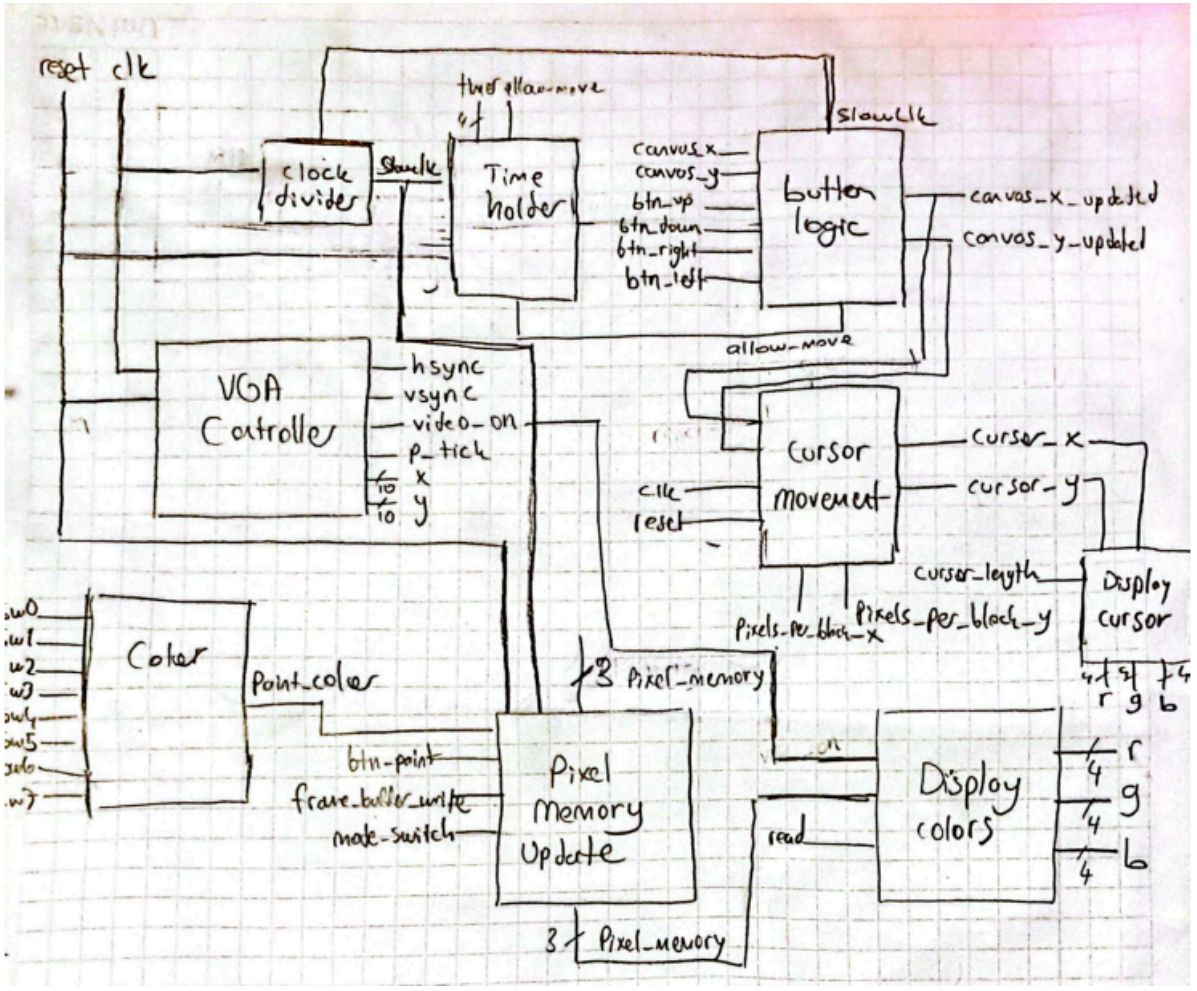





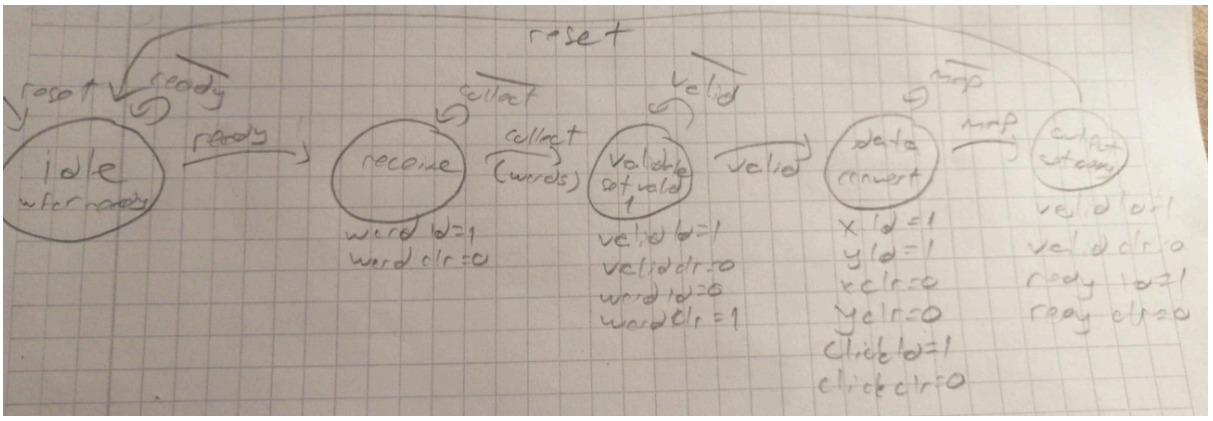
# PAINT LOGIC

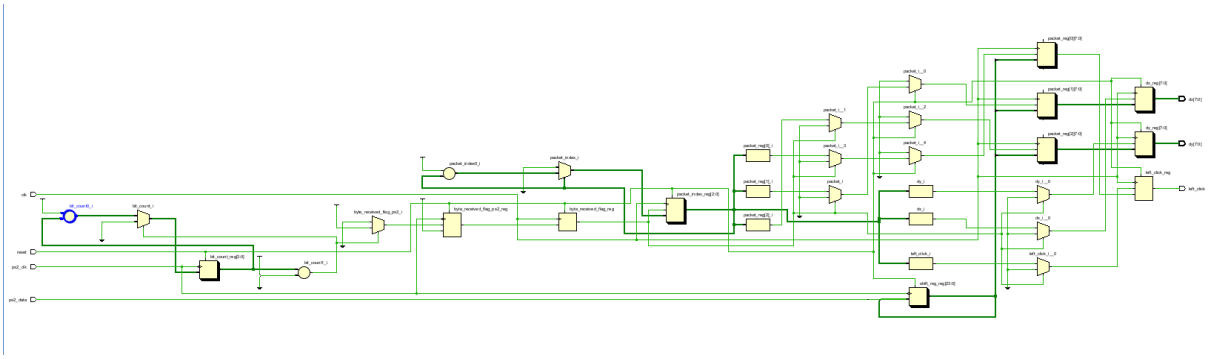

Paint logic
reset
reset
reset
button Paint
button Paint
init
idle
Paint
Pixel_clr=1
Pixel_id=0
color_id=0
color_clr=0
Pixel_clr=0
Pixel_id=0
color_id=1
color_clr=0
Pixel_clr=0
Pixel_id=1
color_id=0
color_clr=1

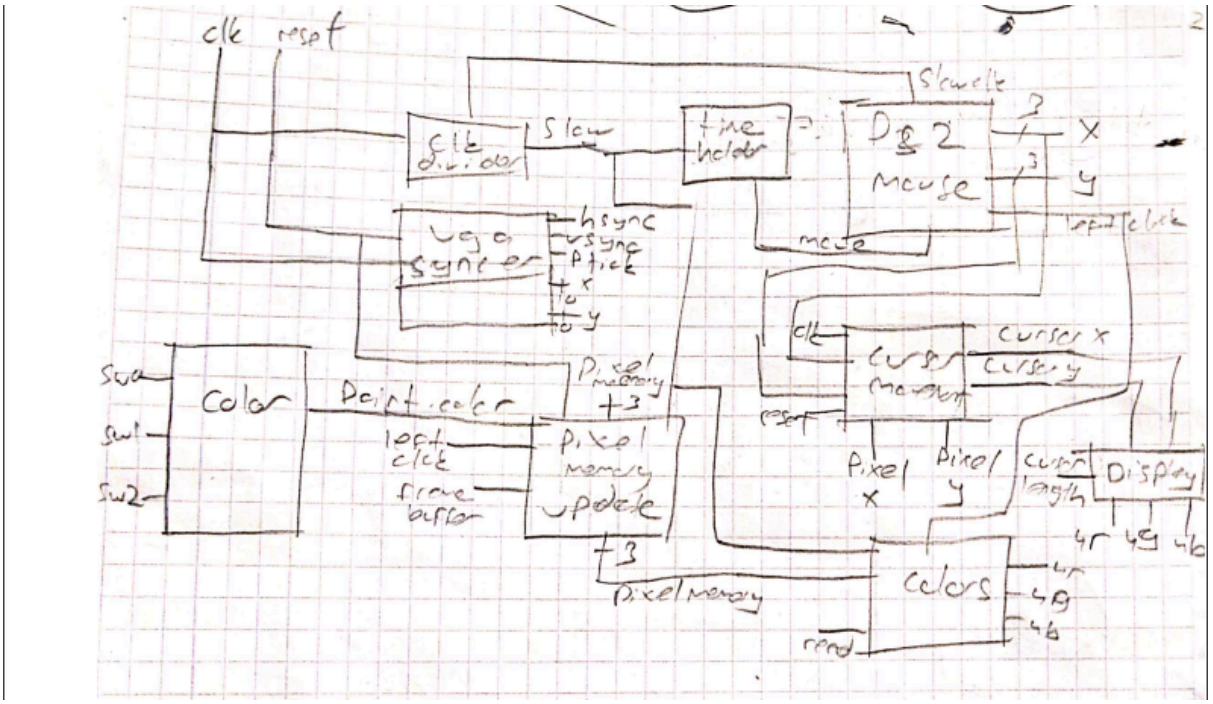



reset
clk
clock divider
slowclk
Time holder
two_allow_move
slowclk
canvas_x
canvas_y
btn_up
btn_down
btn_right
btn_left
button logic
canvas_x_updated
canvas_y_updated
allow_move
VGA Controller
hsync
vsync
video_on
p_tick
x
y
Cursor movement
clk
reset
cursor_x
cursor_y
cursor_length
Display cursor
Pixels_per_block_x
Pixels_per_block_y
r
g
b
Color
Point_color
3
Pixel_memory
btn_point
frame_buffer_write
make_switch
Pixel Memory Update
read
Display colors
3
Pixel_memory

# PS/2 MOUSE LOGIC





# Appendix

//most important module

**VGA CONTROLLER:**

```
module vga_syncer (
    input  logic clk,
    input  logic reset,
    output logic hsync,
    output logic vsync,
    output logic video_on,
    output logic p_tick,
    output logic [9:0] x,
    output logic [9:0] y
);

    localparam int H_DISPLAY   = 640;
    localparam int H_FRONT     = 16;
    localparam int H_SYNC      = 96;
    localparam int H_BACK      = 48;
    localparam int H_TOTAL     = H_DISPLAY + H_FRONT + H_SYNC + H_BACK;

    localparam int V_DISPLAY   = 480;
    localparam int V_FRONT     = 10;
    localparam int V_SYNC      = 2;
    localparam int V_BACK      = 33;
    localparam int V_TOTAL     = V_DISPLAY + V_FRONT + V_SYNC + V_BACK;

    logic [9:0] h_count;
    logic [9:0] v_count;

    logic clk_25mhz;
    logic clk_divider;

    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            clk_divider <= 0;
            clk_25mhz <= 0;
        end else begin
            clk_divider <= ~clk_divider;
            if (clk_divider)
                clk_25mhz <= ~clk_25mhz;
        end
    end

    assign p_tick = clk_25mhz;
```

```
    always_ff @(posedge clk_25mhz or posedge reset) begin
        if (reset)
            h_count <= 0;
        else if (h_count == H_TOTAL - 1)
            h_count <= 0;
        else
            h_count <= h_count + 1;
    end

    always_ff @(posedge clk_25mhz or posedge reset) begin
        if (reset)
            v_count <= 0;
        else if (h_count == H_TOTAL - 1) begin
            if (v_count == V_TOTAL - 1)
                v_count <= 0;
            else
                v_count <= v_count + 1;
        end
    end

    assign hsync = (h_count >= H_DISPLAY + H_FRONT) && (h_count < H_DISPLAY +
H_FRONT + H_SYNC);
    assign vsync = (v_count >= V_DISPLAY + V_FRONT) && (v_count < V_DISPLAY +
V_FRONT + V_SYNC);

    assign video_on = (h_count < H_DISPLAY) && (v_count < V_DISPLAY);

    assign x = (h_count < H_DISPLAY) ? h_count : 0;
    assign y = (v_count < V_DISPLAY) ? v_count : 0;

endmodule
```

**CHECKERBOARD:**

```
module vga_checker_top
(
    input  logic       clk,
    input  logic       reset,
    input  logic [11:0] sw,
    input  logic       btn_up,
    input  logic       btn_dwn,
    input  logic       btn_right,
    input  logic       btn_left,
    output logic        hsync,
    output logic        vsync,
    output logic [11:0] rgb
);

    logic video_on;
```

```
logic p_tick;
logic [9:0] x, y;
logic [11:0] rgb_reg;

logic signed [9:0] x_offset;
logic signed [9:0] y_offset;

logic signed [9:0] dx;
logic signed [9:0] dy;

logic [31:0] frame_counter;

vga_syncer su (
    .clk      (clk),
    .reset    (reset),
    .hsync    (hsync),
    .vsync    (vsync),
    .video_on (video_on),
    .p_tick   (p_tick),
    .x        (x),
    .y        (y)
);

logic signed [10:0] x_temp, y_temp;
logic [9:0] x_mod, y_mod;

always_comb begin
    x_temp = x + x_offset;
    y_temp = y + y_offset;

    if (x_temp < 0)
        x_mod = x_temp + 640;
    else if (x_temp >= 640)
        x_mod = x_temp - 640;
    else
        x_mod = x_temp[9:0];

    if (y_temp < 0)
        y_mod = y_temp + 320;
    else if (y_temp >= 320)
        y_mod = y_temp - 320;
    else
        y_mod = y_temp[9:0];
end

always_ff @(posedge clk or posedge reset) begin
    if (reset) begin
        dx <= 0;
```

```
            dy <= 0;
        end else begin
            if (btn_up) begin
                dx <= 0;
                dy <= -1;  // scroll up
            end else if (btn_dwn) begin
                dx <= 0;
                dy <= 1;   // scroll down
            end else if (btn_left) begin
                dx <= 1;
                dy <= 0;   // scroll left
            end else if (btn_right) begin
                dx <= -1;
                dy <= 0;   // scroll right
            end
        end
    end

    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            x_offset      <= 0;
            y_offset      <= 0;
            frame_counter <= 0;
        end else if (p_tick) begin
            frame_counter <= frame_counter + 1;
            if (frame_counter == 100000) begin
                x_offset <= x_offset + dx;
                y_offset <= y_offset + dy;
                frame_counter <= 0;
            end
        end
    end

    always_ff @(posedge clk or posedge reset) begin
        if (reset)
            rgb_reg <= 12'h000;
        else begin
            if (x_mod[5] ^ y_mod[5])
                rgb_reg <= 12'hFFF;
            else
                rgb_reg <= 12'h000;
        end
    end

    assign rgb = video_on ? rgb_reg : 12'b0;

endmodule
```

**DRAWING CANVAS BUTTON:**

```
module vga_drawing_with_buttons(
    input logic clk,
    input logic reset,
    input logic brush,
    input logic btn_up,
    input logic btn_down,
    input logic btn_left,
    input logic btn_right,
    input logic btn_paint,
    input logic [2:0] sw,
    output logic hsync,
    output logic vsync,
    output logic [11:0] rgb
);

    logic [11:0] rgb_reg;
    logic video_on;
    logic [9:0] x, y;

    logic [5:0] cursor_x = 20;
    logic [4:0] cursor_y = 15;

    logic [3:0] button_reg, button_prev;
    logic move_up, move_down, move_left, move_right;

    assign move_up    = ~button_prev[3] & button_reg[3];
    assign move_down  = ~button_prev[2] & button_reg[2];
    assign move_left  = ~button_prev[1] & button_reg[1];
    assign move_right = ~button_prev[0] & button_reg[0];

    (* ram_style = "block" *) logic [2:0] pixel_memory [0:1199];

    vga_syncer vga_c(
        .clk(clk),
        .reset(reset),
        .hsync(hsync),
        .vsync(vsync),
        .video_on(video_on),
        .p_tick(),
        .x(x),
        .y(y)
    );

    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            cursor_x <= 20;
            cursor_y <= 15;
```

```
        button_reg <= 4'b0000;
        button_prev <= 4'b0000;
        for (int i = 0; i < 1200; i++) begin
            pixel_memory[i] <= 3'b110;
        end
    end else begin
        button_prev <= button_reg;
        button_reg <= {btn_up, btn_down, btn_left, btn_right};

        if (move_up && cursor_y > 0)
            cursor_y <= cursor_y - 1;
        if (move_down && cursor_y < 29)
            cursor_y <= cursor_y + 1;
        if (move_left && cursor_x > 0)
            cursor_x <= cursor_x - 1;
        if (move_right && cursor_x < 39)
            cursor_x <= cursor_x + 1;

        if (btn_paint) begin
            if (brush) begin
                pixel_memory[(cursor_y-1) * 40 + cursor_x-1] <= sw[2:0];
                pixel_memory[(cursor_y-1) * 40 + cursor_x] <= sw[2:0];
                pixel_memory[(cursor_y-1) * 40 + cursor_x+1] <= sw[2:0];

                pixel_memory[cursor_y * 40 + cursor_x-1] <= sw[2:0];
                pixel_memory[cursor_y * 40 + cursor_x] <= sw[2:0];
                pixel_memory[cursor_y * 40 + cursor_x+1] <= sw[2:0];

                pixel_memory[(cursor_y+1) * 40 + cursor_x-1] <= sw[2:0];
                pixel_memory[(cursor_y+1) * 40 + cursor_x] <= sw[2:0];
                pixel_memory[(cursor_y+1) * 40 + cursor_x+1] <= sw[2:0];
            end else begin
                pixel_memory[cursor_y * 40 + cursor_x] <= sw[2:0];
            end
        end
    end
end

always_comb begin
    logic [5:0] block_x = x >> 4;
    logic [4:0] block_y = y >> 4;

    logic [2:0] pixel_color;
    pixel_color = pixel_memory[block_y * 40 + block_x];

    case (pixel_color)
        3'b000: rgb_reg = 12'hF00;
        3'b001: rgb_reg = 12'h0F0;
```

```
            3'b010: rgb_reg = 12'h00F;
            3'b011: rgb_reg = 12'hFF0;
            3'b100: rgb_reg = 12'hF0F;
            3'b101: rgb_reg = 12'h0FF;
            3'b110: rgb_reg = 12'hFFF;
            3'b111: rgb_reg = 12'h000;
            default: rgb_reg = 12'hFFF;
        endcase

        if ((block_x == cursor_x && block_y >= cursor_y - 1 && block_y <= cursor_y + 1) ||
            (block_y == cursor_y && block_x >= cursor_x - 1 && block_x <= cursor_x + 1)) begin
            rgb_reg = 12'h000;
        end
    end

    assign rgb = (video_on) ? rgb_reg : 12'b0;

endmodule
```

**DRAWING CANVAS MOUSE:**

```
`timescale 1ns / 1ps

module vga_drawing_with_mouse(
    input logic clk,
    input logic reset,
    input logic [2:0] sw,
    input logic ps2_data,
    input logic ps2_clk,
    output logic hsync,
    output logic vsync,
    output logic [11:0] rgb
);

    localparam int W = 640;
    localparam int H = 480;
    localparam int GRIDW=16;
    localparam int GRIDH = 16;
    localparam int COLUMNS = W / GRIDW;
    localparam int ROWS = H / GRIDH;

    // Cursor size
    localparam int sizeofcursor = 30;

    logic [7:0] x_delta;
    logic x_sign, x_ov;
    logic [7:0] y_delta;
    logic y_sign, y_ov;
```

```
    logic valid, l_click;

    logic [9:0] vga_x, vga_y;
    logic video_on, pixel_tick;

    logic [5:0] cursor_col;
    logic [5:0] cursor_row;

    logic [2:0] grid_colors [0:ROWS * COLUMNS - 1];

    // Instantiating the mouse movement with ps2 data
    ps2_mouse mousemotion(
        .i_clk(clk),
        .i_reset(reset),
        .ps2_data(ps2_data),
        .ps2_clk(ps2_clk),
        .o_x(x_delta),
        .o_x_ov(x_ov),
        .o_x_sign(x_sign),
        .o_y(y_delta),
        .o_y_ov(y_ov),
        .o_y_sign(y_sign),
        .o_l_click(l_click),
        .o_valid(valid)
    );

    // Using vga sync which is the controller unit of vga, allows to see the movements in the
screen and so
    vga_syncer vgasyncer(
        .clk(clk),
        .reset(reset),
        .hsync(hsync),
        .vsync(vsync),
        .video_on(video_on),
        .p_tick(p_tick),
        .x(vga_x),
        .y(vga_y)
    );

    // Updating the cursor according to mouse movement and input
    logic [23:0] move_counter;
    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            cursor_col <= COLUMNS / 2;
            cursor_row <= ROWS / 2;
            move_counter <= 0;
        end else if (valid && move_counter == 0) begin
            if (x_delta != 0) begin
```

```
            if (x_sign && cursor_col > 0)
                cursor_col <= cursor_col - 1;
            else if (!x_sign && cursor_col < COLUMNS - 1)
                cursor_col <= cursor_col + 1;
        end

        if (y_delta != 0) begin
            if (y_sign && cursor_row < ROWS - 1)
                cursor_row <= cursor_row + 1;
            else if (!y_sign && cursor_row > 0)
                cursor_row <= cursor_row - 1;
        end

        move_counter <= 24'd7_500_000; // to obtain a slower motion of the cursor
    end else if (move_counter > 0) begin
        move_counter <= move_counter - 1;
    end
end

// Mouse drawing logic when left click tapped
always_ff @(posedge clk or posedge reset) begin
    if (reset) begin
        for (int i = 0; i < ROWS * COLUMNS; i++)
            grid_colors[i] <= 3'b110;
    end else if (valid && l_click && x_ov == 0 && y_ov == 0) begin
            grid_colors[cursor_row * COLUMNS + cursor_col] <= sw[2:0];
    end
end

logic [2:0] cell_color = grid_colors[(vga_y / GRIDH) * COLUMNS + (vga_x / GRIDW)];

logic [9:0] cursor_x = cursor_col * GRIDW + GRIDW / 2;
logic [9:0] cursor_y = cursor_row * GRIDH + GRIDH / 2;

logic is_cursor_pixel = video_on &&
                    ((vga_x >= cursor_x - sizeofcursor / 2 && vga_x <= cursor_x + sizeofcursor
/ 2 && vga_y == cursor_y) ||
                     (vga_y >= cursor_y - sizeofcursor / 2 && vga_y <= cursor_y + sizeofcursor
/ 2 && vga_x == cursor_x));

always_comb begin
    if (video_on) begin
        if (is_cursor_pixel)
            rgb = 12'h000;
        else begin
            case (cell_color)
                3'b000: rgb = 12'hF00;
                3'b001: rgb = 12'h0F0;
```

```
                3'b010: rgb = 12'h00F;
                3'b011: rgb = 12'hFF0;
                3'b100: rgb = 12'hF0F;
                3'b101: rgb = 12'h0FF;
                3'b110: rgb = 12'hFFF;
                3'b111: rgb = 12'h000;
                default: rgb = 12'hFFF;
            endcase
        end
    end else begin
        rgb = 12'h000;
    end
end

endmodule
```

**PS2MOUSETOP:**

```
`timescale 1ns / 1ps

module ps2_mouse (
    input  logic       i_clk,
    input  logic       i_reset,
    input  logic       ps2_data,
    input  logic       ps2_clk,
    output logic [7:0] o_x,
    output logic       o_x_ov,
    output logic       o_x_sign,
    output logic [7:0] o_y,
    output logic       o_y_ov,
    output logic       o_y_sign,
    output logic       o_r_click,
    output logic       o_l_click,
    output logic       o_valid
);

    logic [10:0] word1, word2, word3, word4;
    logic [7:0]  signal1, signal2, signal3, signal4;
    logic        valid, ready;

    assign o_valid = ready && valid;

    ps2_signal ps2_signal_inst (
        .i_clk(i_clk),
        .i_reset(i_reset),
        .ps2_clk(ps2_clk),
        .ps2_data(ps2_data),
        .o_word1(word1),
        .o_word2(word2),
```

```
        .o_word3(word3),
        .o_word4(word4),
        .o_ready(ready)
    );

    ps2_validator ps2_validator_inst (
        .i_word1(word1),
        .i_word2(word2),
        .i_word3(word3),
        .i_word4(word4),
        .o_signal1(signal1),
        .o_signal2(signal2),
        .o_signal3(signal3),
        .o_signal4(signal4),
        .o_valid(valid)
    );

    ps2_mouse_map ps2_mouse_map_inst (
        .i_clk(i_clk),
        .i_reset(i_reset),
        .i_signal1(signal1),
        .i_signal2(signal2),
        .i_signal3(signal3),
        .i_signal4(signal4),
        .o_x(o_x),
        .o_y(o_y),
        .o_x_overflow(o_x_ov),
        .o_y_overflow(o_y_ov),
        .o_x_sign(o_x_sign),
        .o_y_sign(o_y_sign),
        .o_l_click(o_l_click),
        .o_r_click(o_r_click)
    );

endmodule
```

**PS2SIGNAL:**

```
module ps2_signal (
    input  logic        i_clk,
    input  logic        i_reset,
    input  logic        ps2_clk,
    input  logic        ps2_data,
    output logic [10:0]  o_word1,
    output logic [10:0]  o_word2,
    output logic [10:0]  o_word3,
    output logic [10:0]  o_word4,
    output logic         o_ready
);
```

```
logic [43:0] fifo;
logic [43:0] buffer;
logic [5:0]  counter;
logic [1:0]  PS2Clk_sync;
logic        ready;
logic        PS2Data;
logic        PS2Clk_negedge;

assign o_word1 = fifo[33 +: 11];
assign o_word2 = fifo[22 +: 11];
assign o_word3 = fifo[11 +: 11];
assign o_word4 = fifo[0 +: 11];
assign o_ready = ready;
assign PS2Clk_negedge = (PS2Clk_sync == 2'b10);

initial begin
    fifo <= 44'b0;
    buffer <= 44'b0;
    counter <= 6'b0;
    PS2Clk_sync <= 2'b1;
    ready <= 1'b0;
    PS2Data <= 1'b0;
end

always_ff @(posedge i_clk) begin
    if (i_reset) begin
        fifo <= 44'b0;
        buffer <= 44'b0;
        counter <= 6'b0;
        ready <= 1'b0;
        PS2Clk_sync <= 2'b1;
        PS2Data <= 1'b0;
    end else begin
        PS2Clk_sync <= {PS2Clk_sync[0], ps2_clk};
        PS2Data <= ps2_data;

        if (PS2Clk_negedge) begin
            buffer <= {buffer, PS2Data};
            counter <= counter + 6'b1;
        end

        if (counter == 6'd44) begin
            fifo <= buffer;
            buffer <= 44'b0;
            counter <= 6'b0;
            ready <= 1'b1;
        end else begin
```

```
            ready <= 1'b0;
            fifo <= 44'b0;
          end
        end
    end
endmodule
```

**PS2 VALIDATOR:**

```
`timescale 1ns / 1ps

module ps2_validator (
    input  logic [10:0] i_word1,
    input  logic [10:0] i_word2,
    input  logic [10:0] i_word3,
    input  logic [10:0] i_word4,
    output logic [7:0]  o_signal1,
    output logic [7:0]  o_signal2,
    output logic [7:0]  o_signal3,
    output logic [7:0]  o_signal4,
    output logic        o_valid
);


    logic parity1, parity2, parity3, parity4, parity;
    logic start1, start2, start3, start4, start;
    logic stop1, stop2, stop3, stop4, stop;
    logic valid1, valid2, valid3, valid4;

    assign {start1, o_signal1, parity1, stop1} = i_word1;
    assign {start2, o_signal2, parity2, stop2} = i_word2;
    assign {start3, o_signal3, parity3, stop3} = i_word3;
    assign {start4, o_signal4, parity4, stop4} = i_word4;

    assign valid1 = ~^o_signal1 == parity1;
    assign valid2 = ~^o_signal2 == parity2;
    assign valid3 = ~^o_signal3 == parity3;
    assign valid4 = ~^o_signal4 == parity4;

    assign parity = valid1 && valid2 && valid3 && valid4;
    assign start = (!start1 && !start2 && !start3 && !start4);
    assign stop = (stop1 && stop2 && stop3 && stop4);
    assign o_valid = (start && stop && parity);

endmodule
```

**PS2MAP:**

```
`timescale 1ns / 1ps

module ps2_mouse_map (
    input  logic       i_clk,
    input  logic       i_reset,
    input  logic [7:0]  i_signal1,
    input  logic [7:0]  i_signal2,
    input  logic [7:0]  i_signal3,
    input  logic [7:0]  i_signal4,
    output logic [7:0]  o_x,
    output logic [7:0]  o_y,
    output logic       o_x_sign,
    output logic       o_y_sign,
    output logic       o_l_click,
    output logic       o_r_click,
    output logic       o_x_overflow,
    output logic       o_y_overflow
);


    assign o_x = i_signal2;
    assign o_y = i_signal3;
    assign {o_x_overflow, o_y_overflow} = i_signal1[1:0];
    assign {o_x_sign, o_y_sign} = i_signal1[3:2];
    assign {o_l_click, o_r_click} = i_signal1[7:6];
endmodule
```

**MOUSEINTERFACE:**

```
module mouse_interface(
    input logic clk,
    input logic reset,
    input logic ps2_clk,
    input logic ps2_data,
    output logic [7:0] dx,
    output logic [7:0] dy,
    output logic left_click
);

    logic [23:0] shift_reg;
    logic [3:0] bit_count;
    logic byte_received_flag_ps2;
    logic byte_received_flag;
    logic [2:0] packet_index;
    logic [7:0] packet[0:2];

    always_ff @(posedge clk or posedge reset) begin
        if (reset)
```

```
            byte_received_flag <= 0;
        else
            byte_received_flag <= byte_received_flag_ps2;
    end

    always_ff @(negedge ps2_clk or posedge reset) begin
        if (reset) begin
            shift_reg <= 24'b0;
            bit_count <= 0;
            byte_received_flag_ps2 <= 0;
        end else begin
            shift_reg <= {ps2_data, shift_reg[23:1]};
            if (bit_count < 10) begin
                bit_count <= bit_count + 1;
            end else begin
                byte_received_flag_ps2 <= 1;
                bit_count <= 0;
            end
        end
    end

   always_ff @(posedge clk or posedge reset) begin
    if (reset) begin
        packet_index <= 0;
        dx <= 0;
        dy <= 0;
        left_click <= 0;
    end else if (byte_received_flag) begin

        packet[packet_index] <= shift_reg[8:1];
        packet_index <= packet_index + 1;


        if (packet_index == 2) begin
            // Decode the mouse packet
            left_click <= packet[0][0];

            // Handle signed movement for dx and dy
            dx <= {packet[0][4], packet[1]};
            dy <= {packet[0][5], packet[2]};

            packet_index <= 0;
        end
    end
end

endmodule
```

**Mouse Movement:**

```
`timescale 1ns / 1ps

module mouse_movement_logic(
    input  logic        clk,              // System clock
    input  logic        reset,
    input  logic [2:0]  sw,
    input  logic        ps2_data,
    input  logic        ps2_clk,
    output logic         hsync,
    output logic         vsync,
    output logic [11:0] rgb
);

    // VGA Controller signals
    logic video_on;                    // Active video region signal
    logic [9:0] x, y;                  // Current pixel coordinates

    // Cursor position (40x30 grid)
    logic signed [5:0] cursor_x;
    logic signed [4:0] cursor_y;

    // PS/2 Mouse signals
    logic [7:0] mouse_x;
    logic [7:0] mouse_y;
    logic mouse_x_sign;
    logic mouse_y_sign;
    logic mouse_left_click;
    logic mouse_valid;

    // Pixel memory for 40x30 grid
    (* ram_style = "block" *) logic [2:0] pixel_memory [0:1199]; // 3 bits per cell

    // Instantiate PS/2 Mouse Decoder
    ps2_mouse ps2_mouse_inst (
        .i_clk(clk),
        .i_reset(reset),
        .i_PS2Data(ps2_data),
        .i_PS2Clk(ps2_clk),
        .o_x(mouse_x),
        .o_x_ov(),
        .o_x_sign(mouse_x_sign),
        .o_y(mouse_y),
        .o_y_ov(),
        .o_y_sign(mouse_y_sign),
        .o_r_click(),
        .o_l_click(mouse_left_click),
        .o_valid(mouse_valid)
```

```
    );

    // Instantiate VGA Controller
    vga_syncer vga_sync_inst (
        .clk(clk),
        .reset(reset),
        .hsync(hsync),
        .vsync(vsync),
        .video_on(video_on),
        .p_tick(),
        .x(x),
        .y(y)
    );

    // Cursor movement and boundary constraints
    logic signed [11:0] accumulated_x; // Accumulator for X movement
    logic signed [11:0] accumulated_y; // Accumulator for Y movement

    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            // Reset cursor to center of the grid
            cursor_x <= 20;
            cursor_y <= 15;
            accumulated_x <= 0;
            accumulated_y <= 0;

            // Clear pixel memory to default color (white)
            for (int i = 0; i < 1200; i++) begin
                pixel_memory[i] <= 3'b110; // White
            end
        end else if (mouse_valid) begin
            // Accumulate PS/2 mouse movement
            if (mouse_x_sign) accumulated_x <= accumulated_x - mouse_x;
            else accumulated_x <= accumulated_x + mouse_x;

            if (mouse_y_sign) accumulated_y <= accumulated_y + mouse_y; // Inverted Y-axis
for VGA
            else accumulated_y <= accumulated_y - mouse_y;

            // Apply movement when accumulated value exceeds the threshold
            if (accumulated_x >= 4 || accumulated_x <= -4) begin
                cursor_x <= cursor_x + (accumulated_x / 4); // Apply scaled movement
                accumulated_x <= accumulated_x % 4;       // Retain remainder
            end

            if (accumulated_y >= 4 || accumulated_y <= -4) begin
                cursor_y <= cursor_y + (accumulated_y / 4); // Apply scaled movement
                accumulated_y <= accumulated_y % 4;       // Retain remainder
```

```
        end

        // Clamp cursor position within grid boundaries
        if (cursor_x > 39) cursor_x <= 39;
        if (cursor_x < 0) cursor_x <= 0;
        if (cursor_y > 29) cursor_y <= 29;
        if (cursor_y < 0) cursor_y <= 0;

        // Paint the current cursor location on left-click (only 1x1 pixel)
        if (mouse_left_click) begin
            // Paint a single pixel (1x1)
            pixel_memory[cursor_y * 40 + cursor_x] <= sw[2:0];
        end
    end
end

// Generate RGB output
logic [11:0] rgb_reg;
always_comb begin
    // Scale x and y coordinates to match the 16x16 blocks
    logic [5:0] block_x = x >> 4; // Divide by 16
    logic [4:0] block_y = y >> 4; // Divide by 16

    // Memory read
    logic [2:0] pixel_color;
    pixel_color = pixel_memory[block_y * 40 + block_x];

    // Determine output pixel color based on color value (0-7)
    case (pixel_color)
        3'b000: rgb_reg = 12'hF00; // Red
        3'b001: rgb_reg = 12'h0F0; // Green
        3'b010: rgb_reg = 12'h00F; // Blue
        3'b011: rgb_reg = 12'hFF0; // Yellow
        3'b100: rgb_reg = 12'hF0F; // Magenta
        3'b101: rgb_reg = 12'h0FF; // Cyan
        3'b110: rgb_reg = 12'hFFF; // White
        3'b111: rgb_reg = 12'h000; // Black
        default: rgb_reg = 12'hFFF; // Default white
    endcase

    // Cursor is a "+" shape with priority over drawn pixels
    if ((block_x == cursor_x && block_y >= cursor_y - 1 && block_y <= cursor_y + 1) ||
        (block_y == cursor_y && block_x >= cursor_x - 1 && block_x <= cursor_x + 1)) begin
        rgb_reg = 12'h000;  // Black cursor
    end
end
assign rgb = (video_on) ? rgb_reg : 12'b0;
endmodule
```

## CONSTRAINTS

**Checkerboard const:**

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]

        set_property IOSTANDARD LVCMOS33 [get_ports clk]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

# Switches
set_property PACKAGE_PIN V17      [get_ports {sw[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN V16      [get_ports {sw[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W16      [get_ports {sw[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
set_property PACKAGE_PIN W17      [get_ports {sw[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
set_property PACKAGE_PIN W15      [get_ports {sw[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
set_property PACKAGE_PIN V15      [get_ports {sw[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
set_property PACKAGE_PIN W14      [get_ports {sw[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
set_property PACKAGE_PIN W13      [get_ports {sw[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
set_property PACKAGE_PIN V2       [get_ports {sw[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
set_property PACKAGE_PIN T3       [get_ports {sw[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
set_property PACKAGE_PIN T2       [get_ports {sw[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
set_property PACKAGE_PIN R3       [get_ports {sw[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]


#set_property PACKAGE_PIN U18 [get_ports btnC]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnC]
set_property PACKAGE_PIN T18 [get_ports btn_up]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_up]
set_property PACKAGE_PIN W19 [get_ports btn_left]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_left]
set_property PACKAGE_PIN T17 [get_ports btn_right]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_right]
```

```
set_property PACKAGE_PIN U17 [get_ports btn_dwn]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_dwn]

##VGA Connector
set_property PACKAGE_PIN G19     [get_ports {rgb[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN H19     [get_ports {rgb[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN J19     [get_ports {rgb[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN N19     [get_ports {rgb[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[3]}]
set_property PACKAGE_PIN J17     [get_ports {rgb[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[4]}]
set_property PACKAGE_PIN H17     [get_ports {rgb[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[5]}]
set_property PACKAGE_PIN G17     [get_ports {rgb[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[6]}]
set_property PACKAGE_PIN D17     [get_ports {rgb[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[7]}]
set_property PACKAGE_PIN N18     [get_ports {rgb[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[8]}]
set_property PACKAGE_PIN L18     [get_ports {rgb[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[9]}]
set_property PACKAGE_PIN K18     [get_ports {rgb[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[10]}]
set_property PACKAGE_PIN J18     [get_ports {rgb[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {rgb[11]}]
set_property PACKAGE_PIN P19     [get_ports hsync]

set_property IOSTANDARD LVCMOS33 [get_ports hsync]
set_property PACKAGE_PIN R19     [get_ports vsync]

set_property IOSTANDARD LVCMOS33 [get_ports vsync]

##Buttons
set_property PACKAGE_PIN U18      [get_ports reset]

set_property IOSTANDARD LVCMOS33 [get_ports reset]
```

**BUTTON CONST:**

```
## 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 PACKAGE_PIN W5 [get_ports clk]

        set_property IOSTANDARD LVCMOS33 [get_ports clk]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

# Switches
set_property PACKAGE_PIN V17 [get_ports {reset}]
        set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property PACKAGE_PIN V16 [get_ports {sw[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN W16 [get_ports {sw[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W17 [get_ports {sw[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W13 [get_ports {sw[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
set_property PACKAGE_PIN R2 [get_ports {brush}]
        set_property IOSTANDARD LVCMOS33 [get_ports {brush}]


## PS/2 Data
#set_property PACKAGE_PIN J2 [get_ports ps2_data]
#set_property IOSTANDARD LVCMOS33 [get_ports ps2_data]

# LEDs
#set_property PACKAGE_PIN U16 [get_ports {q[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {q[3]}]
```

```
#set_property PACKAGE_PIN E19 [get_ports {q[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {q[2]}]
#set_property PACKAGE_PIN U19 [get_ports {q[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {q[1]}]
#set_property PACKAGE_PIN V19 [get_ports {q[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {q[0]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports {led[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports {led[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports {led[8]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
#set_property PACKAGE_PIN V3 [get_ports {led[9]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports {led[10]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports {led[11]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports {led[12]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
#set_property PACKAGE_PIN N3 [get_ports {led[13]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
#set_property PACKAGE_PIN P1 [get_ports {led[14]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
#set_property PACKAGE_PIN L1 [get_ports {led[15]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]


##7 segment display
#set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
#set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
#set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
#set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
#set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
#set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
#set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
```

```
#set_property PACKAGE_PIN V7 [get_ports dp]

        #set_property IOSTANDARD LVCMOS33 [get_ports dp]

#set_property PACKAGE_PIN U2 [get_ports {an[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
#set_property PACKAGE_PIN U4 [get_ports {an[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
#set_property PACKAGE_PIN V4 [get_ports {an[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
#set_property PACKAGE_PIN W4 [get_ports {an[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]


##Buttons
set_property PACKAGE_PIN U18 [get_ports btn_paint]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_paint]
set_property PACKAGE_PIN T18 [get_ports btn_up]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_up]
set_property PACKAGE_PIN W19 [get_ports btn_left]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_left]
set_property PACKAGE_PIN T17 [get_ports btn_right]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_right]
set_property PACKAGE_PIN U17 [get_ports btn_down]

        set_property IOSTANDARD LVCMOS33 [get_ports btn_down]



##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
```

```
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]



##Pmod Header JB
##Sch name = JB1
#set_property PACKAGE_PIN A14 [get_ports {JB[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[0]}]
##Sch name = JB2
#set_property PACKAGE_PIN A16 [get_ports {JB[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[1]}]
##Sch name = JB3
#set_property PACKAGE_PIN B15 [get_ports {JB[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]
##Sch name = JB4
#set_property PACKAGE_PIN B16 [get_ports {JB[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[3]}]
##Sch name = JB7
#set_property PACKAGE_PIN A15 [get_ports {JB[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[4]}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]



##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports {JC[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[0]}]
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports {JC[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[1]}]
```

```
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports {JC[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports {JC[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports {JC[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[4]}]
##Sch name = JC8
#set_property PACKAGE_PIN M19 [get_ports {JC[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[5]}]
##Sch name = JC9
#set_property PACKAGE_PIN P17 [get_ports {JC[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[6]}]
##Sch name = JC10
#set_property PACKAGE_PIN R18 [get_ports {JC[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[7]}]


##Pmod Header JXADC
##Sch name = XA1_P
#set_property PACKAGE_PIN J3 [get_ports {JXADC[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[0]}]
##Sch name = XA2_P
#set_property PACKAGE_PIN L3 [get_ports {JXADC[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[1]}]
##Sch name = XA3_P
#set_property PACKAGE_PIN M2 [get_ports {JXADC[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[2]}]
##Sch name = XA4_P
#set_property PACKAGE_PIN N2 [get_ports {JXADC[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[3]}]
##Sch name = XA1_N
#set_property PACKAGE_PIN K3 [get_ports {JXADC[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[4]}]
##Sch name = XA2_N
#set_property PACKAGE_PIN M3 [get_ports {JXADC[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[5]}]
##Sch name = XA3_N
#set_property PACKAGE_PIN M1 [get_ports {JXADC[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[6]}]
##Sch name = XA4_N
#set_property PACKAGE_PIN N1 [get_ports {JXADC[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[7]}]
```

```
##VGA Connector
set_property PACKAGE_PIN G19 [get_ports {rgb[0]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN H19 [get_ports {rgb[1]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN J19 [get_ports {rgb[2]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN N19 [get_ports {rgb[3]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[3]}]
set_property PACKAGE_PIN N18 [get_ports {rgb[4]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[4]}]
set_property PACKAGE_PIN L18 [get_ports {rgb[5]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[5]}]
set_property PACKAGE_PIN K18 [get_ports {rgb[6]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[6]}]
set_property PACKAGE_PIN J18 [get_ports {rgb[7]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[7]}]
set_property PACKAGE_PIN J17 [get_ports {rgb[8]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[8]}]
set_property PACKAGE_PIN H17 [get_ports {rgb[9]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[9]}]
set_property PACKAGE_PIN G17 [get_ports {rgb[10]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[10]}]
set_property PACKAGE_PIN D17 [get_ports {rgb[11]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[11]}]
set_property PACKAGE_PIN P19 [get_ports hsync]
	set_property IOSTANDARD LVCMOS33 [get_ports hsync]
set_property PACKAGE_PIN R19 [get_ports vsync]
	set_property IOSTANDARD LVCMOS33 [get_ports vsync]


##USB-RS232 Interface
#set_property PACKAGE_PIN B18 [get_ports RsRx]

	#set_property IOSTANDARD LVCMOS33 [get_ports RsRx]
#set_property PACKAGE_PIN A18 [get_ports RsTx]

	#set_property IOSTANDARD LVCMOS33 [get_ports RsTx]


##USB HID (PS/2)
#set_property PACKAGE_PIN C17 [get_ports ps2_clk]

	#set_property IOSTANDARD LVCMOS33 [get_ports ps2_clk]
	#set_property PULLUP true [get_ports ps2_clk]
#set_property PACKAGE_PIN B17 [get_ports ps2_data]
	#set_property IOSTANDARD LVCMOS33 [get_ports ps2_data]
	#set_property PULLUP true [get_ports ps2_data]
```

##Quad SPI Flash
##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the
##STARTUPE2 primitive.
#set_property PACKAGE_PIN D18 [get_ports {QspiDB[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[0]}]
#set_property PACKAGE_PIN D19 [get_ports {QspiDB[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[1]}]
#set_property PACKAGE_PIN G18 [get_ports {QspiDB[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[2]}]
#set_property PACKAGE_PIN F18 [get_ports {QspiDB[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[3]}]
#set_property PACKAGE_PIN K19 [get_ports QspiCSn]
	#set_property IOSTANDARD LVCMOS33 [get_ports QspiCSn]

**MOUSE CONST:**
## 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 PACKAGE_PIN W5 [get_ports clk]

	set_property IOSTANDARD LVCMOS33 [get_ports clk]
	create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

# Switches
set_property PACKAGE_PIN V17 [get_ports {reset}]
	set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property PACKAGE_PIN V16 [get_ports {sw[0]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN W16 [get_ports {sw[1]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W17 [get_ports {sw[2]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W13 [get_ports {sw[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[7]}]

```
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {brush}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {brush}]


## PS/2 Data
#set_property PACKAGE_PIN J2 [get_ports ps2_data]
#set_property IOSTANDARD LVCMOS33 [get_ports ps2_data]

# LEDs
#set_property PACKAGE_PIN U16 [get_ports {q[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {q[3]}]
#set_property PACKAGE_PIN E19 [get_ports {q[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {q[2]}]
#set_property PACKAGE_PIN U19 [get_ports {q[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {q[1]}]
#set_property PACKAGE_PIN V19 [get_ports {q[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {q[0]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports {led[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports {led[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports {led[8]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
#set_property PACKAGE_PIN V3 [get_ports {led[9]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports {led[10]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports {led[11]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports {led[12]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
```

```
#set_property PACKAGE_PIN N3 [get_ports {led[13]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
#set_property PACKAGE_PIN P1 [get_ports {led[14]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
#set_property PACKAGE_PIN L1 [get_ports {led[15]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]


##7 segment display
#set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
#set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
#set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
#set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
#set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
#set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
#set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]

	#set_property IOSTANDARD LVCMOS33 [get_ports dp]

#set_property PACKAGE_PIN U2 [get_ports {an[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
#set_property PACKAGE_PIN U4 [get_ports {an[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
#set_property PACKAGE_PIN V4 [get_ports {an[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
#set_property PACKAGE_PIN W4 [get_ports {an[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]


##Buttons
#set_property PACKAGE_PIN U18 [get_ports btn_paint]

	#set_property IOSTANDARD LVCMOS33 [get_ports btn_paint]
#set_property PACKAGE_PIN T18 [get_ports btn_up]

	#set_property IOSTANDARD LVCMOS33 [get_ports btn_up]
#set_property PACKAGE_PIN W19 [get_ports btn_left]

	#set_property IOSTANDARD LVCMOS33 [get_ports btn_left]
```

```
#set_property PACKAGE_PIN T17 [get_ports btn_right]

        #set_property IOSTANDARD LVCMOS33 [get_ports btn_right]
#set_property PACKAGE_PIN U17 [get_ports btn_down]

        #set_property IOSTANDARD LVCMOS33 [get_ports btn_down]



##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]



##Pmod Header JB
##Sch name = JB1
#set_property PACKAGE_PIN A14 [get_ports {JB[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[0]}]
##Sch name = JB2
#set_property PACKAGE_PIN A16 [get_ports {JB[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[1]}]
##Sch name = JB3
#set_property PACKAGE_PIN B15 [get_ports {JB[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]
##Sch name = JB4
```

```
#set_property PACKAGE_PIN B16 [get_ports {JB[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[3]}]
##Sch name = JB7
#set_property PACKAGE_PIN A15 [get_ports {JB[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[4]}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]



##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports {JC[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[0]}]
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports {JC[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[1]}]
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports {JC[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports {JC[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports {JC[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[4]}]
##Sch name = JC8
#set_property PACKAGE_PIN M19 [get_ports {JC[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[5]}]
##Sch name = JC9
#set_property PACKAGE_PIN P17 [get_ports {JC[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[6]}]
##Sch name = JC10
#set_property PACKAGE_PIN R18 [get_ports {JC[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[7]}]



##Pmod Header JXADC
##Sch name = XA1_P
#set_property PACKAGE_PIN J3 [get_ports {JXADC[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[0]}]
```

```
##Sch name = XA2_P
#set_property PACKAGE_PIN L3 [get_ports {JXADC[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[1]}]
##Sch name = XA3_P
#set_property PACKAGE_PIN M2 [get_ports {JXADC[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[2]}]
##Sch name = XA4_P
#set_property PACKAGE_PIN N2 [get_ports {JXADC[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[3]}]
##Sch name = XA1_N
#set_property PACKAGE_PIN K3 [get_ports {JXADC[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[4]}]
##Sch name = XA2_N
#set_property PACKAGE_PIN M3 [get_ports {JXADC[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[5]}]
##Sch name = XA3_N
#set_property PACKAGE_PIN M1 [get_ports {JXADC[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[6]}]
##Sch name = XA4_N
#set_property PACKAGE_PIN N1 [get_ports {JXADC[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[7]}]



##VGA Connector
set_property PACKAGE_PIN G19 [get_ports {rgb[0]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN H19 [get_ports {rgb[1]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN J19 [get_ports {rgb[2]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN N19 [get_ports {rgb[3]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[3]}]
set_property PACKAGE_PIN N18 [get_ports {rgb[4]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[4]}]
set_property PACKAGE_PIN L18 [get_ports {rgb[5]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[5]}]
set_property PACKAGE_PIN K18 [get_ports {rgb[6]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[6]}]
set_property PACKAGE_PIN J18 [get_ports {rgb[7]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[7]}]
set_property PACKAGE_PIN J17 [get_ports {rgb[8]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[8]}]
set_property PACKAGE_PIN H17 [get_ports {rgb[9]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[9]}]
set_property PACKAGE_PIN G17 [get_ports {rgb[10]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {rgb[10]}]
set_property PACKAGE_PIN D17 [get_ports {rgb[11]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {rgb[11]}]
set_property PACKAGE_PIN P19 [get_ports hsync]
        set_property IOSTANDARD LVCMOS33 [get_ports hsync]
set_property PACKAGE_PIN R19 [get_ports vsync]
        set_property IOSTANDARD LVCMOS33 [get_ports vsync]


##USB-RS232 Interface
#set_property PACKAGE_PIN B18 [get_ports RsRx]

        #set_property IOSTANDARD LVCMOS33 [get_ports RsRx]
#set_property PACKAGE_PIN A18 [get_ports RsTx]

        #set_property IOSTANDARD LVCMOS33 [get_ports RsTx]


##USB HID (PS/2)
set_property PACKAGE_PIN C17 [get_ports ps2_clk]

        set_property IOSTANDARD LVCMOS33 [get_ports ps2_clk]
        set_property PULLUP true [get_ports ps2_clk]
set_property PACKAGE_PIN B17 [get_ports ps2_data]
        set_property IOSTANDARD LVCMOS33 [get_ports ps2_data]
        set_property PULLUP true [get_ports ps2_data]


##Quad SPI Flash
##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the
##STARTUPE2 primitive.
#set_property PACKAGE_PIN D18 [get_ports {QspiDB[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[0]}]
#set_property PACKAGE_PIN D19 [get_ports {QspiDB[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[1]}]
#set_property PACKAGE_PIN G18 [get_ports {QspiDB[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[2]}]
#set_property PACKAGE_PIN F18 [get_ports {QspiDB[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[3]}]
#set_property PACKAGE_PIN K19 [get_ports QspiCSn]
        #set_property IOSTANDARD LVCMOS33 [get_ports QspiCSn]
```