Alexander Maso
ECE 540 - Winter 2023

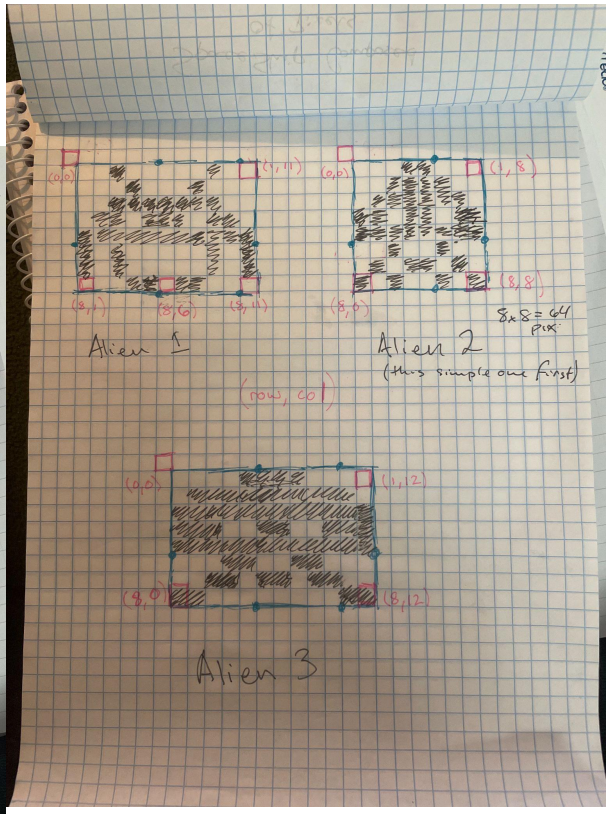
Final Project

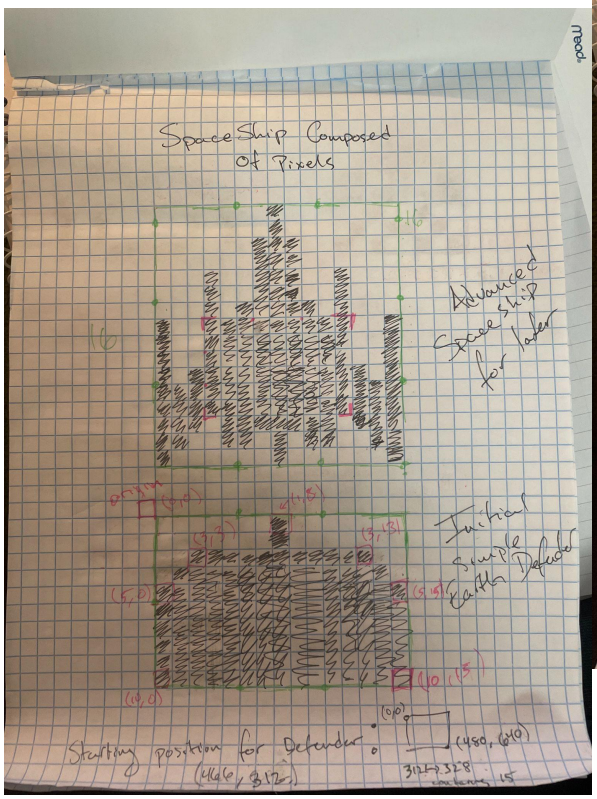
Interim Report

Rutuja and I have elected to write separate interim reports for our Final Project since our deliverables are totally separate from the other's. This way, we are each able to focus on describing the individual progress we have made towards the goals laid out in our initial report. Over the past week, Rutuja has been focused on working out how to connect and send data back and forth between the android application, the ESP32 microcontroller and the rvfgpa board, whereas I have been focused on the sprite images and movement that will be controlled on the rvpfga board itself.

Progress since Thursday, March 2

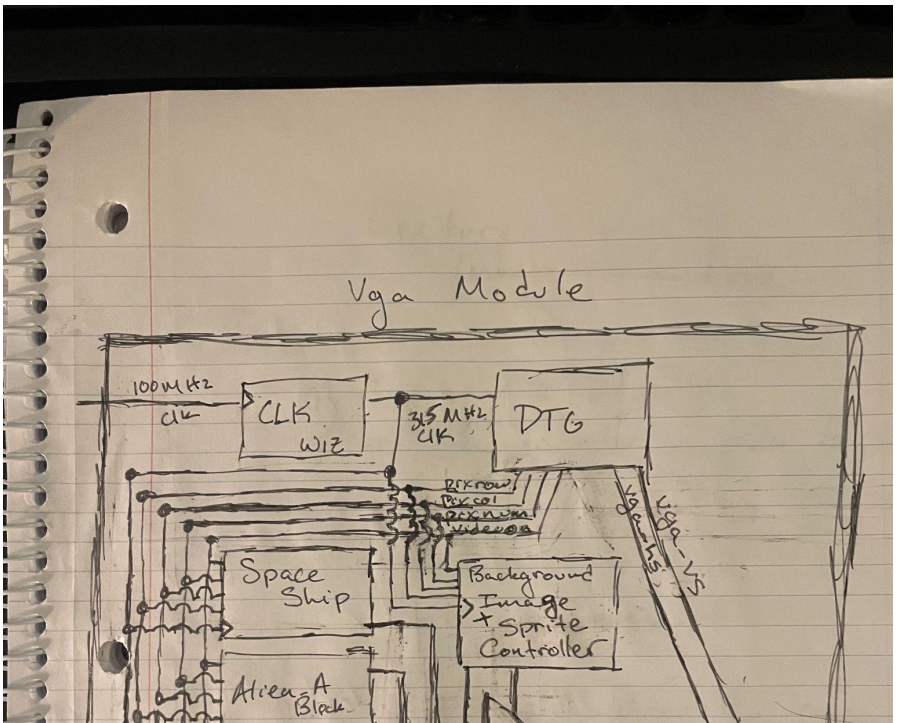
1. After submitting the Final Project Proposal last week, I began by creating a remote GitHub repo to store/track changes to my RVfpga src code files and uploading a clean, untouched src directory.
2. Next I created a button module with registers to hold the row and column value and connected them to the required pins just like in Lab 3 and Lab 4.
3. I then created a vga module, used the Clk Wiz to create a 31.5 MHz clock, add the dtg source file from lab 4 and had the module output gray for now (vga_output = 4'b0011);
4. In order to create the Player and Alien sprites, I googled old images of Space Invaders games and recreated a few of the pixel images on graph paper.





5. Focusing on the Player's sprite first, I added logic to the vga module that would display the player's sprite (the simple one for now) whenever in the sprite's area.
6. Next, I tied the player sprite's row and column registers to the corresponding registers in the button module just as in lab 4. Finally, using the assembly code I wrote for Lab 4, I enabled control of the sprite's position through the push buttons. (Temporary solution to allow testing of other functions….Will eventually be controlled through android app)
7. Next were the alien sprites. I quickly realized it was not practical to keep all of the different sprite information in the vga module itself. There were just too many registers and too much logic required for one module. To help myself map out the design, I sketched a new diagram of all the modules involved. (Included below)
8. Following the new design, I moved the code for the player's sprite into its own module and created similar modules for each of the alien sprites as well (3 designs for now).
9. I added instantiations of the clk wiz, dtg and sprite modules within the vga module and added logic to determine when to display the different sprite images.

```
module alien2(
   input  wire               clk, rst,
   input  wire    [11:0]     pixel_row, pixel_column,
output wire    [3:0]   alien2_output,
   output wire              alien2_active
   );

// Internals
logic   [11:0]            sprite_row;
logic   [11:0]            sprite_column;
logic   [11:0]            sprite_row_ff;
logic   [11:0]            sprite_column_ff;
logic   [3:0]             alien2_pix_ff;
logic   [3:0]             alien2_pix;
logic                     active;

initial begin
   active = 1'b0;
   alien2_pix = 4'b0000;
   alien2_pix_ff = 4'b0000;

   // Initializing Alien2 20 rows from the top of the screen. Rows 21 <-> 28
   // and close to centered as possible. Columns: 317 <-> 324
   sprite_column_ff = 316;
   sprite_row_ff = 20;
end

always_comb begin
   // Enable output of the player's sprite when in the proper region
   // Alien2's Sprite is 8 rows by 8 columns of pixels
   // *** Still unsure on the implementation of this active signal....
   // ...Do I want to just use two assignments with a ternary operator
instead?
   //
   // assign active = (sprite_row < pixel row) && .....
   // assign player_active = active ? 1'b1 : 1'b0;
   //
   /*
   if ((sprite_row < pixel_row) && (pixel_row < sprite_row + 9) &&
(sprite_column < pixel_column) && (pixel_column < sprite_column
+ 9))
      begin
         active = 1'b1;
      end
   else
      begin
         active = 1'b0;
      end

   */

   // Are we in the Sprite Region???
```



```
    active = ((sprite_row < pixel_row) && (pixel_row < sprite_row + 9) && (sprite_column < pixel_column) && (pixel_column < sprite_column + 9));


    // Sprite data for Alien2
    // Row one of Alien2's Sprite
    if ((pixel_row == sprite_row  + 1) && (sprite_column + 3 < pixel_column) && (pixel_column < sprite_column + 6))
        begin
            alien2_pix = 4'b1111;
        end
    // Row two of Alien2's Sprite
    else if ((pixel_row == sprite_row  + 2) && (sprite_column + 2 < pixel_column) && (pixel_column < sprite_column + 7))
        begin
            alien2_pix = 4'b1111;
        end
    // Row three of Alien2's Sprite
    else if ((pixel_row == sprite_row  + 3) && (sprite_column + 1 < pixel_column) && (pixel_column < sprite_column + 8))
        begin
            alien2_pix = 4'b1111;
        end
    // Row four of Alien2's Sprite
    else if ((pixel_row == sprite_row  + 4) && (pixel_column != sprite_column + 3) && (pixel_column != sprite_column + 6))
        begin
            active = 1'b1;
        end
    // Row five of Alien2's Sprite
    else if ((pixel_row == sprite_row + 5) && (sprite_column < pixel_column) && (pixel_column < sprite_column + 9))
        begin
            active = 1'b1;
        end
    // Row six of Alien2's Sprite
    else if ((pixel_row == sprite_row  + 6) && ((sprite_column + 3 == pixel_column) || (pixel_column == sprite_column + 6)))
        begin
            alien2_pix = 4'b1111;
        end
    // Row seven of Alien2's Sprite
    else if ((pixel_row == sprite_row  + 7) && (pixel_column != sprite_column + 1) && (pixel_column != sprite_column + 3) && (pixel_column != sprite_column + 6) &&
(pixel_column != sprite_column + 8))
        begin
            alien2_pix = 4'b1111;
        end
    // Row eight of Alien2's Sprite
    else if ((pixel_row == sprite_row  + 8) && (pixel_column != sprite_column + 2) && (pixel_column != sprite_column + 4) && (pixel_column != sprite_column + 5) &&
(pixel_column != sprite_column + 7))
        begin
            alien2_pix = 4'b1111;
        end
    else
        begin
            alien2_pix = 4'b0000;
        end
end


always_ff @ (posedge clk or posedge rst)
begin
    if (rst)
    begin
        alien2_pix_ff <= 0;
    end
    else
    begin
        alien2_pix_ff <= alien2_pix;
    end
end

assign alien2_active = active ? 1'b1 : 1'b0;

assign sprite_row = sprite_row_ff;
assign sprite_column = sprite_column_ff;
assign alien2_output = alien2_pix_ff;

endmodule


module player(
    input wire              clk, rst,
    input wire  [11:0]      btn_row, btn_column,
    input wire  [11:0]      pixel_row, pixel_column,
    output wire [3:0]       player_output,
    output wire             player_active
```

```
    );

    // Internals
    logic   [11:0]          sprite_row;
    logic   [11:0]          sprite_column;
    logic   [11:0]          sprite_row_ff;
    logic   [11:0]          sprite_column_ff;
    logic   [11:0]          btn_row_ff;
    logic   [11:0]          btn_column_ff;
    logic   [3:0]           player_pix_ff;
    logic   [3:0]           player_pix;
    logic                   active;

    initial begin
    active = 1'b0;
    player_pix = 4'b0000;
    player_pix_ff = 4'b0000;
    sprite_column_ff = 0;
    sprite_row_ff = 0;
    btn_column_ff = 0;
    btn_row_ff = 0;
    end

    // Player is centered in the upper left corner | Row x Col / (0,0)
    // Row one and Row two of the ship sprite
    always_comb begin
    // Enable output of the player's sprite when in the proper region
    // Player's Sprite is 10 rows by 15 columns of pixels
    // *** Still unsure on the implementation of this active signal....
    // ...Do I want to just use two assignments with a ternary operator instead?
    //
    // assign active = (sprite_row < pixel row) && .....
    // assign player_active = active ? 1'b1 : 1'b0;
    //

/*
    if ((sprite_row < pixel_row) && (pixel_row < sprite_row + 11) && (sprite_column < pixel_column) && (pixel_column < sprite_column + 16))
        begin
            active = 1'b1;
        end
    else
        begin
            active = 1'b0;
        end
*/
    // In the Sprite Region???
    active = ((sprite_row < pixel_row) && (pixel_row < sprite_row + 11) && (sprite_column < pixel_column) && (pixel_column < sprite_column + 16));


    if ((sprite_row < pixel_row) && (pixel_row < sprite_row  + 3) && (pixel_column == sprite_column + 8))
        begin
            player_pix = 4'b1111;
        end
    // Row three of the player's sprite
    else if ((sprite_row + 3 == pixel_row) && (sprite_column + 2 < pixel_column) && (pixel_column < sprite_column + 14))
        begin
            player_pix = 4'b1111;
        end
    // Row four of the player's sprite
    else if ((sprite_row + 4 == pixel_row) && (sprite_column + 1 < pixel_column) && (pixel_column < sprite_column + 15))
        begin
            player_pix = 4'b1111;
        end
    // Row five through 10 of the player's sprite
    else if (((sprite_row + 4 < pixel_row) && (pixel_row < sprite_row  + 11)) && (sprite_column < pixel_column) && (pixel_column < sprite_column + 16))
        begin
            player_pix = 4'b1111;
        end
    else
        begin
```

```
            player_pix = 4'b0000;
        end
end

always_ff @ (posedge clk or posedge rst)
begin
    if (rst)
        begin
        btn_row_ff <= 0;
        btn_column_ff <= 0;
        sprite_row_ff <= 0;
        sprite_column_ff <= 0;
        player_pix_ff <= 0;
        end
    else
        begin
        btn_row_ff <= btn_row;
        btn_column_ff <= btn_column;
        sprite_row_ff <= btn_row_ff;
        sprite_column_ff <= btn_column_ff;
        player_pix_ff <= player_pix;
        end
end


assign player_active = active ? 1'b1 : 1'b0;

assign sprite_row = sprite_row_ff;
assign sprite_column = sprite_column_ff;
assign player_output = player_pix_ff;


endmodule
```