**ECE241 Final Project Report**

“Terry Crews vs. Aliens”

Nathan Jones and Jordan Greenberg

(1005003023 and 1005046530)

PRA0102

BA 3145

Station 13

December 2, 2019

**1.0 Introduction**

For our final project, we wanted to create a fun game using our knowledge that we gained throughout the course. Our hope was to make this project an enjoyable experience to conceptualize, design, and build, all while incorporating many concepts covered in the course. Our goal was to finish the project with a fully functional game, along with the satisfaction that we were able to accomplish this using programmable hardware.

Our project is called “Terry Crews vs. Aliens.” It consists of custom drawn tanks, aliens, and rockets (Terry Crews sits in the tank). The tank is controlled by the user using the keys on the FPGA board. The user can move the tank left or right, as well as fire a rocket. The twist is that the user has to fire strategically. We only allow one rocket on the game screen at once. This does not allow for the user to fire as many times as possible, challenging the user to devise a strategy to shoot down the falling aliens. The aliens fall vertically, spawning randomly each game. This ensures that the game stays exciting and does not become repetitive and easy for the user. The user wins by shooting down all 6 aliens. If one alien lands on the tank, the game instantly ends and the user loses. The six aliens are spread out along the game screen to make the game challenging for the user.

Through the creation of our game “Terry Crews vs. Aliens,” we wanted to apply our knowledge of programmable hardware to develop a challenging game that we could play, as well as share that enjoyment with our peers. At the end of the project, we were very proud of our finished project and the enjoyment of playing our own game was very rewarding.

**2.0 The Design**

The “Tank Move” module is our top level module of our design, and it combines all our modules to produce our fully functional game (see block diagram below). It takes in user input from both the onboard switches and keys. The switches are used to reset the game, change the background colour according to the users preference, and switch the mode of the game into one where the aliens do not fall (a beginner mode). We decided to use the switches primarily for input that was a “one-time” selection instead of “in-game” input. The keys are used more for the crucial “in-game” input that allows the user to play. The user can control the direction the tank is moving, as well as fire a rocket using the keys. This module outputs the user’s score and x position to a HEX display, as well as the information necessary to display the game to the VGA adapter.

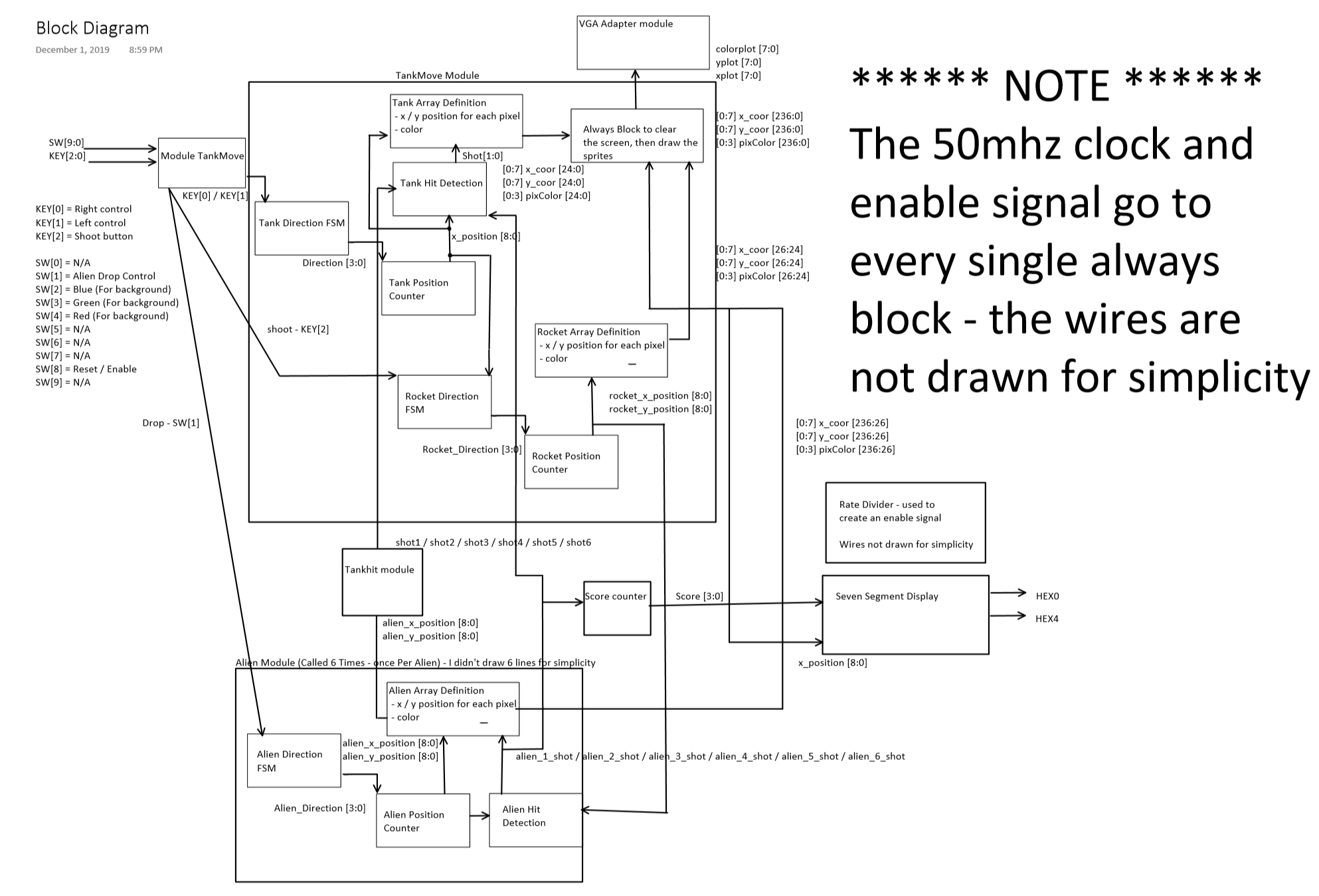
There are many components within the “Tank Move” module to facilitate the functionality of our game. We have a Finite State Machine that controls the direction of the tank movement based on the key input of the user. This FSM controls the Tank Position Counter that stores the current position of the tank. Similarly, we have a Rocket Direction Finite State Machine (since the rocket only moves in one direction, this is only to determine whether or not to ‘launch’ the rocket) and Rocket Position Counter to store where the rocket is on the screen, and whether or not the game is in a state where the user can fire a rocket. As mentioned above, our unique twist is that the user is only allowed to fire one rocket at a time, and must wait for that rocket to fly off the screen before they can fire another (to balance the game).

Within the “Tank Move” module we also have our Tank Hit Detection to detect if any aliens have landed on the tank. This determines whether or not the current position of the tank is to be coloured green (the colour of the tank) or the background colour (the tank has been destroyed). This information is stored in the Tank Array Definition. This definition defines a portion of three “master” arrays. These 3 arrays hold the x position, the y position, and the color of each pixel. The Tank Array definition defines 24 positions in each array (as the tank is drawn using 24 pixels: 24 x - positions, 24 y - positions, and each of the 24 pixels has a color assigned to it). Similarly, we have a Rocket Array Definition that defines another portion of the arrays. The Rocket Array Definition defines 2 positions in each array. Each alien (discussed later in the document) also hold positions in the three ‘master’ arrays. The array definitions are constantly being updated [always@(\*)]. In order to draw each sprite on screen, (and change its position) we first clear the screen by using an always block (updated on the posedge of the 50mhz clock) that functions similarly to a nested for-loop to draw over the entire screen with pixels of the background color. Once the screen is cleared, we cycle through the arrays (using an always block) and send the x coordinate, y coordinate, and color of one pixel at a time to the vga adapter - this is done using the 50mhz clock (to draw all pixels before the adapter can update).

Another major part of our design is our “Alien Module.” We created this module to allow us to easily “instantiate” or “create” aliens, which made our design extremely scalable. Each alien on the screen is a copy of the “Alien Module,” meaning we only had to define the alien parameters and functionality once within our design. This module contains the Alien Finite State Machine which controls the Alien’s direction, and its counter which controls the Alien’s central x and y positions (the counter is dependent on the FSM). The central x-positions and y-positions are used to determine all the pixel positions and colours needed to draw the Alien, and this is stored in the Alien Array Definition. This definition has the same structure as the Tank and Rocket Array Definitions mentioned above. The central positions are also used in our Alien Hit Detection to determine if the alien was shot down by a rocket.

Since we defined all these structures and functionalities within our “Alien Module,” we were easily able to create multiple aliens by instantiating this module multiple times. The inputs to this module included initial x and y positions for the alien. Therefore, by differing these inputs to each alien, we were able to easily create aliens across the entire game screen. Additionally, using a counter driven by the onboard 50 MHz clock, we were able to change the initial y position inputs to the alien modules for each game, creating the effect of “random spawning” for the aliens. The outputs of this module were used to draw the aliens on the screen within the above mentioned always block that clears and re-draws the screen and to our score counter module.

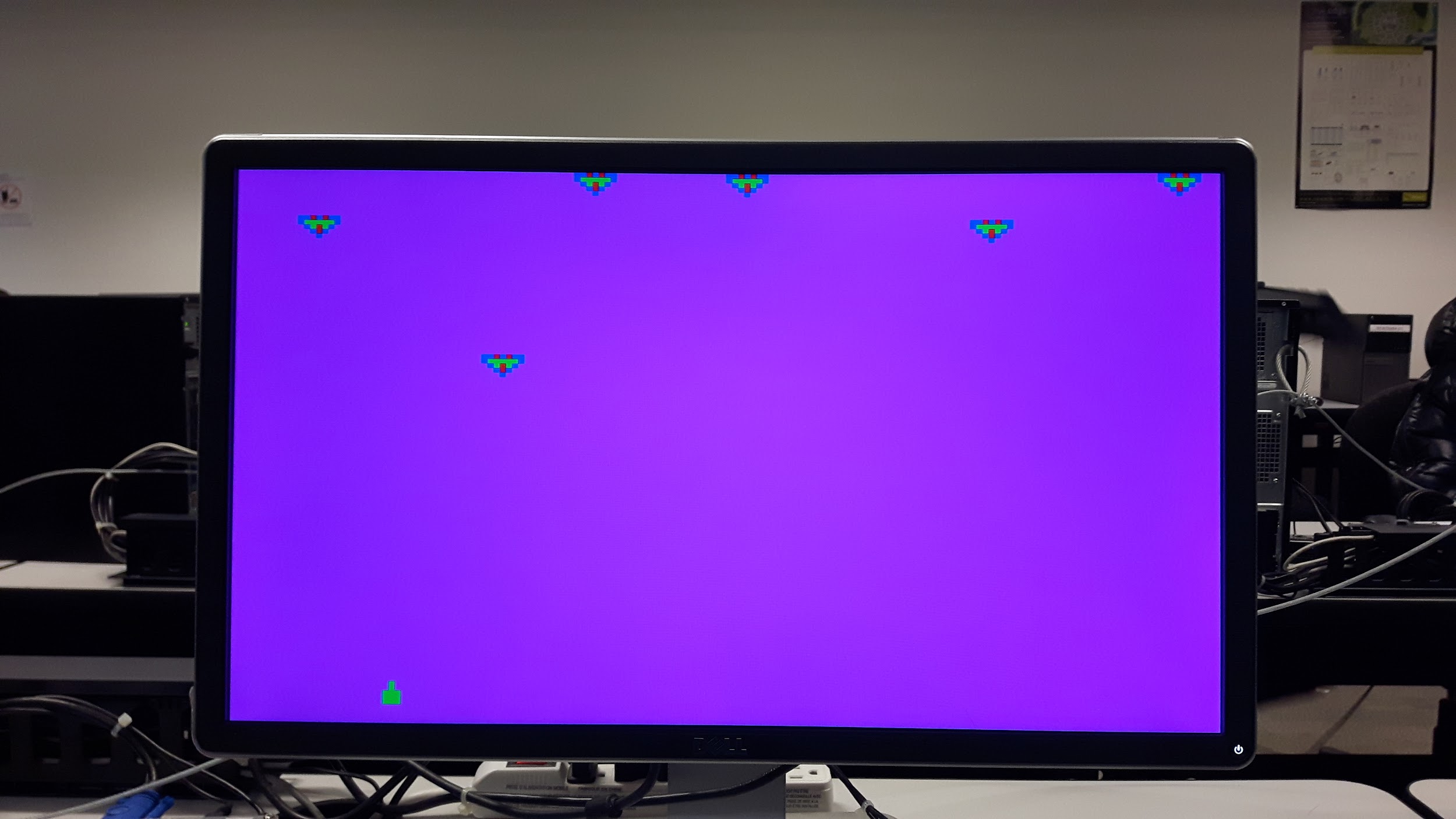
In addition to our two major modules described above, we have a few smaller modules that tie our design together. This includes our “Tank Hit” module. This is a simple module that combines all the x and y positions from each Alien Module and determine if they intersect with the tank x / y positions. If any of the aliens intersect with the tank, signals are sent to the “Tank Move” module, triggering an always block to change the color of the tank from green to the background color (in order to ‘destroy’ the tank). The ‘hit’ module functions identically (except it checks for an intersection between the rocket and aliens). Additionally, we have a “Score Counter” module. This simply takes in the output signals from each alien and determines how many aliens have been shot, thus computing the user’s score. This information is sent to our seven segment display module to allow this information to be deciphered and output to a HEX display on the FPGA board. Our last smaller but crucial module is our rate divider. This module ensures that the timing of all computations within our design are compatible with the limitation of the 60 Hz clock that the VGA adapter runs on and the 50 MHz clock that the rest of our modules operate on. It provides us the ability to “slow down” the parts of our design that should not operate at 50 MHz by producing “enable” signals at the appropriate time. Without this module, our design would operate “too fast” (the tank and aliens would fly across the screen at a speed that is not visible to the human eye) and would not function correctly.

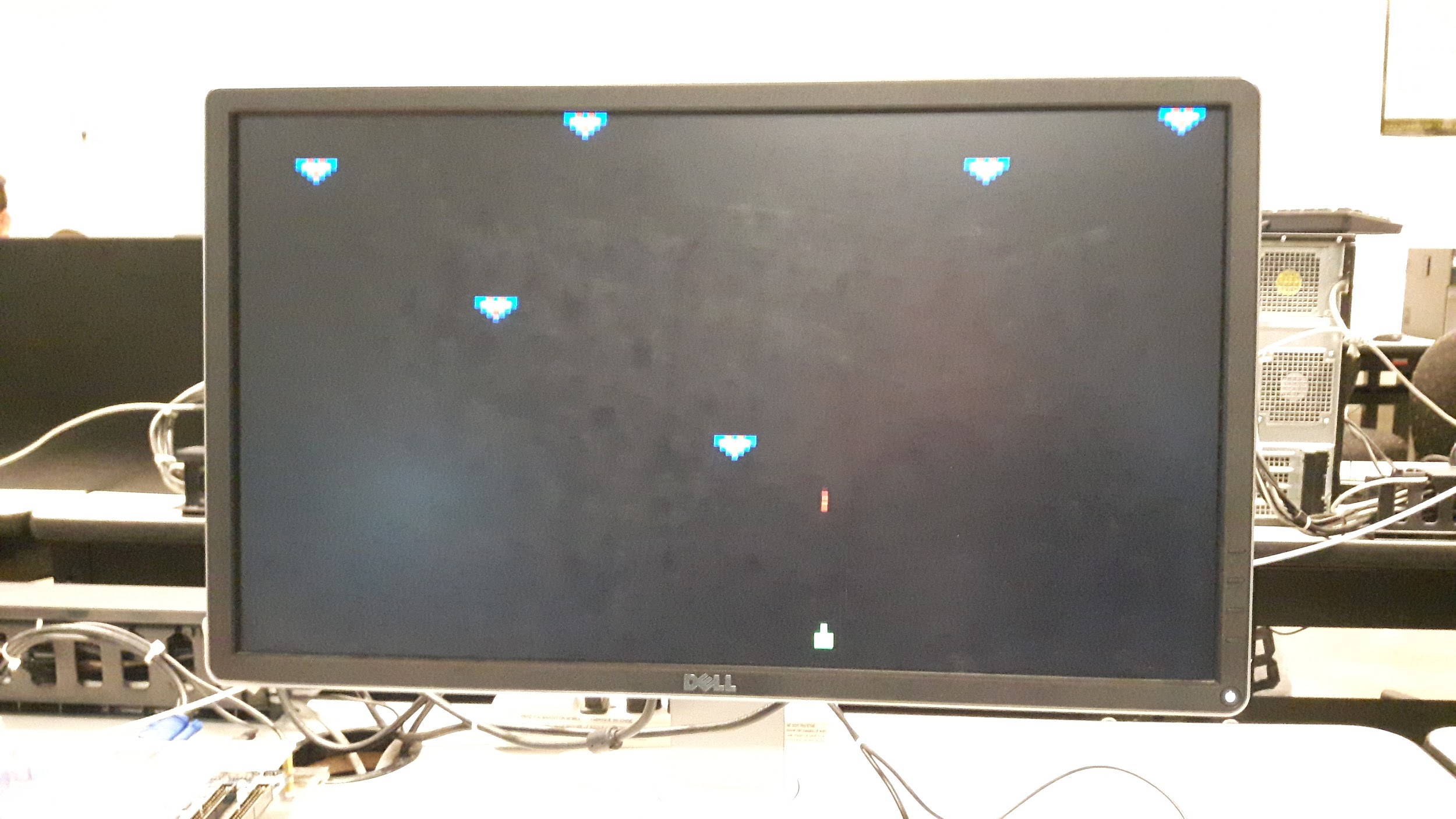


Block Diagram of our design

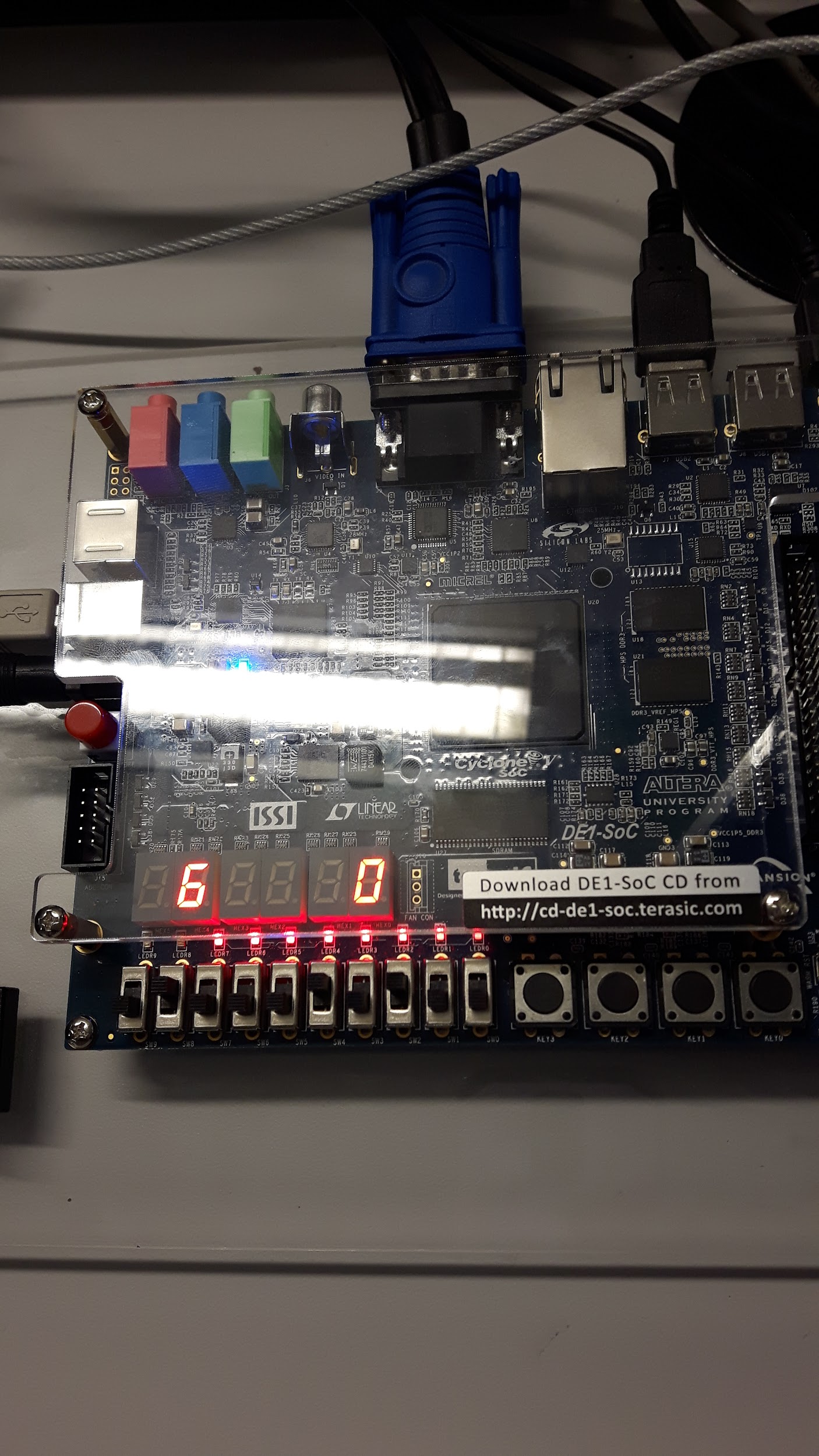
**3.0 Report on Success**

Our design worked perfectly. We could have added more polish, but with the limited time we had, it matched our initial specification exactly. We were able to achieve a user controlled tank that moved side to side, multiple aliens dropping on the screen, and user controlled rocket firing. What brought our design together was our success with hit detection. We were able to implement Aliens being shot down from rocket fires, as well as Tank destruction caused by the Aliens. This completed our game and made it fully functional and playable. It also fulfilled all of our specifications that we laid out at the start of the project. Luckily, we were able to get every part of our design fully working to complete our game. We were particularly proud that we were able to meet all of our milestones and produce a fun and fully functional product that we could actually play.

User can change the background according to their preference



Tank firing



Score on the Seven Segment Display

**4.0 What would we do differently?**

To be completely honest, I believe we wouldn’t change a single thing. At first, we were both very concerned that we wouldn’t be able to adequately complete an assignment using both finite state machines and the vga adapter (due to the increasing workloads from other courses, and the fact that we both did not complete the final lab). We decided early on that we were not going to use any rom (read only memory) or alter any .mif files, just due to the complexity of our game, and the fact that it would most likely take an extended amount of time to implement those ideas (to be honest this was a stroke of luck). As we slowly came up with ideas to implement our game, we had the idea to use a ‘master array’ which holds the coordinates and color of every single pixel on screen. This allowed our game to be extremely scalable, as well as being extremely unique - mostly everyone else in our year used rom in accordance with mif files to print sprites. The array also made hit detection relatively easy to implement. In the end, we were extremely happy with the knowledge we gained, and the quality of the final project.

**Appendix**

**A-Code**

module TankMove(input [9:0]SW, input CLOCK\_50, input [3:0]KEY, output [9:0]LEDR, output [6:0]HEX0, output [6:0]HEX4,

output VGA\_CLK, // VGA Clock

output VGA\_HS, // VGA H\_SYNC

output VGA\_VS, // VGA V\_SYNC

output VGA\_BLANK\_N, // VGA BLANK

output VGA\_SYNC\_N, // VGA SYNC

output [7:0]VGA\_R, // VGA Red[9:0]

output [7:0]VGA\_G, // VGA Green[9:0]

output [7:0]VGA\_B // VGA Blue[9:0]

);

wire Lcontrol, Rcontrol, clk, reset, enable;

reg [2:0]color;

reg [7:0]Last\_x\_position;

wire [26:0]N;

wire [1:0]Left;

wire [1:0]Right;

wire [2:0] colorInput;

wire [1:0]stationary;

reg [7:0]x\_position;

reg [7:0]x\_plot;

wire shoot;

//assign controls

assign Lcontrol = ~KEY[1];

assign Rcontrol = ~KEY[0];

assign colorInput = {SW[2], SW[1], SW[0]};

assign shoot = ~KEY[2];

//assign Clock

assign clk = CLOCK\_50;

assign LEDR[9] = enable;

assign N = 49000000;

//Test

//assign N = 33333332;

//assign Reset, 0 = Reset

assign reset = SW[8];

//Find the direction to move in

reg [1:0]Direction;

assign Left = 2'b10;

assign Right = 2'b01;

assign stationary = 2'b00;

reg [2:0] current\_state;

reg [2:0] next\_state;

//VGA definitions

defparam VGA.RESOLUTION = "160x120";

defparam VGA.MONOCHROME = "FALSE";

defparam VGA.BITS\_PER\_COLOUR\_CHANNEL = 1;

defparam VGA.BACKGROUND\_IMAGE = "black.mif";

//create the correct clock rate

rateDivider #(.width(27)) R1(CLOCK\_50, N, reset, enable);

////Direction//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//FSM for determining direction

always@(\*)

//depending on which controls are pressed, the direction changes accordingly

if(reset == 0)

begin

Direction = stationary;

end

else if((Lcontrol == 1) && (Rcontrol == 0))

begin

Direction = Left;

end

else if((Lcontrol == 0) && (Rcontrol == 1))

begin

Direction = Right;

end

else if((Lcontrol == 0) && (Rcontrol == 0))

begin

Direction = stationary;

end

////Position//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//Always block to change the position according to the direction

always@(posedge clk)

if (reset == 0)

begin

x\_position <= 8'b0;

end

//Block to teleport to sides of the screen if bounds are hit

else if((x\_position + 3) == 8'b10100001)

begin

x\_position <= 8'b0;

end

else if(x\_position == 8'b11111111)

begin

x\_position <= 8'b10011100;

end

//move in the direction defined

else if((Direction == Left) && (enable == 1))

begin

Last\_x\_position <= x\_position;

x\_position <= x\_position - 8'b1;

end

else if((Direction== Right) && (enable == 1))

begin

Last\_x\_position <= x\_position;

x\_position <= x\_position + 8'b1;

end

else if ((Direction == stationary))

begin

x\_position <= x\_position;

end

////Xcoordinate output//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Sevenseg s1(x\_position[3], x\_position[2], x\_position[1], x\_position[0], HEX0);

//Score Count Output///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

wire [3:0]countScore;

scoreCount score1(alien\_1\_shot, alien\_2\_shot, alien\_3\_shot, alien\_4\_shot, alien\_5\_shot, alien\_6\_shot, reset, countScore);

Sevenseg s2(countScore[3], countScore[2], countScore[1], countScore[0], HEX4);

////Tank color assignment//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

reg [2:0]red;

reg [2:0]green;

reg [2:0]blue;

wire [2:0]black;

wire [2:0]background;

assign background = {SW[4], SW[3], SW[2]};

assign black = background;

////Tank hit detection//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

always@(\*)

if ((((shot1 == 1) && (alien\_1\_shot == 0)) || ((shot2 == 1) && (alien\_2\_shot == 0)) || ((shot3 == 1) && (alien\_3\_shot == 0)) || ((shot4 == 1) && (alien\_4\_shot == 0)) || ((shot5 == 1) && (alien\_5\_shot == 0)) || ((shot6 == 1) && (alien\_6\_shot == 0))))

begin

red = black;

green = black;

blue = black;

end

else

begin

red = 3'b100;

green = 3'b010;

blue = 3'b001;

end

////Array Definitions//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//x coordinate array

wire [0:7] x\_coor [236:0];

reg [7:0] x\_clr;

//y coordinate array

wire [0:7] y\_coor [236:0];

reg [7:0] y\_clr;

wire [0:2] pixColor [236:0];

reg [7:0]xplot;

reg [7:0]yplot;

reg [2:0]colorplot;

reg [20:0]count;

///Tank color array assignments//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//Draw the Tank

assign pixColor[0] = black;

assign pixColor[1] = green;

assign pixColor[2] = green;

assign pixColor[3] = green;

assign pixColor[4] = black;

assign pixColor[5] = black;

assign pixColor[6] = green;

assign pixColor[7] = green;

assign pixColor[8] = green;

assign pixColor[9] = black;

assign pixColor[10] = black;

assign pixColor[11] = green;

assign pixColor[12] = green;

assign pixColor[13] = green;

assign pixColor[14] = black;

assign pixColor[15] = black;

assign pixColor[16] = black;

assign pixColor[17] = green;

assign pixColor[18] = black;

assign pixColor[19] = black;

assign pixColor[20] = black;

assign pixColor[21] = black;

assign pixColor[22] = green;

assign pixColor[23] = black;

assign pixColor[24] = black;

////Tank position array assignments//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

assign x\_coor[0] = x\_position;

assign y\_coor[0] = 115;

assign x\_coor[1] = x\_position + 1;

assign y\_coor[1] = 115;

assign x\_coor[2] = x\_position + 2;

assign y\_coor[2] = 115;

assign x\_coor[3] = x\_position + 3;

assign y\_coor[3] = 115;

assign x\_coor[4] = x\_position + 4;

assign y\_coor[4] = 115;

assign x\_coor[5] = x\_position;

assign y\_coor[5] = 114;

assign x\_coor[6] = x\_position + 1;

assign y\_coor[6] = 114;

assign x\_coor[7] = x\_position + 2;

assign y\_coor[7] = 114;

assign x\_coor[8] = x\_position + 3;

assign y\_coor[8] = 114;

assign x\_coor[9] = x\_position + 4;

assign y\_coor[9] = 114;

assign x\_coor[10] = x\_position;

assign y\_coor[10] = 113;

assign x\_coor[11] = x\_position + 1;

assign y\_coor[11] = 113;

assign x\_coor[12] = x\_position + 2;

assign y\_coor[12] = 113;

assign x\_coor[13] = x\_position + 3;

assign y\_coor[13] = 113;

assign x\_coor[14] = x\_position + 4;

assign y\_coor[14] = 113;

assign x\_coor[15] = x\_position;

assign y\_coor[15] = 112;

assign x\_coor[16] = x\_position + 1;

assign y\_coor[16] = 112;

assign x\_coor[17] = x\_position + 2;

assign y\_coor[17] = 112;

assign x\_coor[18] = x\_position + 3;

assign y\_coor[18] = 112;

assign x\_coor[19] = x\_position + 4;

assign y\_coor[19] = 112;

assign x\_coor[20] = x\_position;

assign y\_coor[20] = 111;

assign x\_coor[21] = x\_position + 1;

assign y\_coor[21] = 111;

assign x\_coor[22] = x\_position + 2;

assign y\_coor[22] = 111;

assign x\_coor[23] = x\_position + 3;

assign y\_coor[23] = 111;

assign x\_coor[24] = x\_position + 4;

assign y\_coor[24] = 111;

////////////rocket//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

reg [7:0]rocket\_x\_position;

reg [7:0]rocket\_y\_position;

reg [2:0]rocket\_color;

reg [7:0]rocket\_count;

reg shootInProgress;

reg shootUpdate;

wire shootClear;

assign x\_coor[25] = rocket\_x\_position;

assign y\_coor[25] = rocket\_y\_position;

assign pixColor[25] = rocket\_color;

assign x\_coor[26] = rocket\_x\_position;

assign y\_coor[26] = rocket\_y\_position - 1;

assign pixColor[26] = rocket\_color;

////When to launch the rocket//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

always@(\*)

if(shoot == 1)

begin

shootUpdate = 1;

end

else

begin

shootUpdate = shootClear;

end

////Rocket position//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

always@(posedge clk)

if(reset == 0)

begin

rocket\_count = 0;

end

else if(shootUpdate == 1 && enable == 1 && rocket\_count == 0)

begin

rocket\_x\_position <= x\_position + 2;

rocket\_y\_position <= 111;

rocket\_color <= red;

rocket\_count <= rocket\_count + 1;

shootInProgress <= 1;

end

else if (shootUpdate == 1 && enable == 1 && rocket\_count > 0)

begin

rocket\_x\_position <= rocket\_x\_position;

rocket\_y\_position <= rocket\_y\_position - 1;

rocket\_count <= rocket\_count + 1;

end

else if(rocket\_y\_position == 8'b0)

begin

rocket\_count <= 0;

shootInProgress <= 0;

rocket\_color <= black;

end

assign shootClear = shootInProgress;

////Alien Hit detection//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

wire drop;

wire [7:0] alien\_1\_x\_out;

wire [7:0] alien\_2\_x\_out;

wire [7:0] alien\_3\_x\_out;

wire [7:0] alien\_4\_x\_out;

wire [7:0] alien\_5\_x\_out;

wire [7:0] alien\_6\_x\_out;

wire [7:0] alien\_1\_y\_out;

wire [7:0] alien\_2\_y\_out;

wire [7:0] alien\_3\_y\_out;

wire [7:0] alien\_4\_y\_out;

wire [7:0] alien\_5\_y\_out;

wire [7:0] alien\_6\_y\_out;

wire alien\_1\_shot;

wire alien\_2\_shot;

wire alien\_3\_shot;

wire alien\_4\_shot;

wire alien\_5\_shot;

wire alien\_6\_shot;

wire shot1;

wire shot2;

wire shot3;

wire shot4;

wire shot5;

wire shot6;

//module to see if the rocket and a alien intersect

hit h1(rocket\_x\_position, rocket\_y\_position, reset, alien\_1\_x\_out, alien\_1\_y\_out, alien\_1\_shot, clk);

hit h2(rocket\_x\_position, rocket\_y\_position, reset, alien\_2\_x\_out, alien\_2\_y\_out, alien\_2\_shot, clk);

hit h3(rocket\_x\_position, rocket\_y\_position, reset, alien\_3\_x\_out, alien\_3\_y\_out, alien\_3\_shot, clk);

hit h4(rocket\_x\_position, rocket\_y\_position, reset, alien\_4\_x\_out, alien\_4\_y\_out, alien\_4\_shot, clk);

hit h5(rocket\_x\_position, rocket\_y\_position, reset, alien\_5\_x\_out, alien\_5\_y\_out, alien\_5\_shot, clk);

hit h6(rocket\_x\_position, rocket\_y\_position, reset, alien\_6\_x\_out, alien\_6\_y\_out, alien\_6\_shot, clk);

//Module to find if the tank and a alien intersect

Tankhit T1(x\_position, y\_position, reset, alien\_1\_x\_out, alien\_1\_y\_out, shot1, clk);

Tankhit T2(x\_position, y\_position, reset, alien\_2\_x\_out, alien\_2\_y\_out, shot2, clk);

Tankhit T3(x\_position, y\_position, reset, alien\_3\_x\_out, alien\_3\_y\_out, shot3, clk);

Tankhit T4(x\_position, y\_position, reset, alien\_4\_x\_out, alien\_4\_y\_out, shot4, clk);

Tankhit T5(x\_position, y\_position, reset, alien\_5\_x\_out, alien\_5\_y\_out, shot5, clk);

Tankhit T6(x\_position, y\_position, reset, alien\_6\_x\_out, alien\_6\_y\_out, shot6, clk);

assign drop = SW[1];

//Create and define all aliens

alienModule a1(.clk(clk), .enable(enable), .reset(reset), .alien\_initial\_x(10), .alien\_initial\_y(10), .drop(drop),

.x\_coordinates(x\_coor[61:27]), .y\_coordinates(y\_coor[61:27]), .alien\_colour(pixColor[61:27]), .x\_out(alien\_1\_x\_out), .y\_out(alien\_1\_y\_out), .shot(alien\_1\_shot),.background(background));

alienModule a2(.clk(clk), .enable(enable), .reset(reset), .alien\_initial\_x(40), .alien\_initial\_y(40), .drop(drop),

.x\_coordinates(x\_coor[96:62]), .y\_coordinates(y\_coor[96:62]), .alien\_colour(pixColor[96:62]), .x\_out(alien\_2\_x\_out), .y\_out(alien\_2\_y\_out), .shot(alien\_2\_shot),.background(background));

alienModule a3(.clk(clk), .enable(enable), .reset(reset), .alien\_initial\_x(80), .alien\_initial\_y(70), .drop(drop),

.x\_coordinates(x\_coor[131:97]), .y\_coordinates(y\_coor[131:97]), .alien\_colour(pixColor[131:97]), .x\_out(alien\_3\_x\_out), .y\_out(alien\_3\_y\_out), .shot(alien\_3\_shot),.background(background));

alienModule a4(.clk(clk), .enable(enable), .reset(reset), .alien\_initial\_x(120), .alien\_initial\_y(10), .drop(drop),

.x\_coordinates(x\_coor[166:132]), .y\_coordinates(y\_coor[166:132]), .alien\_colour(pixColor[166:132]), .x\_out(alien\_4\_x\_out), .y\_out(alien\_4\_y\_out), .shot(alien\_4\_shot),.background(background));

alienModule a5(.clk(clk), .enable(enable), .reset(reset), .alien\_initial\_x(55), .alien\_initial\_y(100), .drop(drop),

.x\_coordinates(x\_coor[201:167]), .y\_coordinates(y\_coor[201:167]), .alien\_colour(pixColor[201:167]), .x\_out(alien\_5\_x\_out), .y\_out(alien\_5\_y\_out), .shot(alien\_5\_shot),.background(background));

alienModule a6(.clk(clk), .enable(enable), .reset(reset), .alien\_initial\_x(150), .alien\_initial\_y(0), .drop(drop),

.x\_coordinates(x\_coor[236:202]), .y\_coordinates(y\_coor[236:202]), .alien\_colour(pixColor[236:202]), .x\_out(alien\_6\_x\_out), .y\_out(alien\_6\_y\_out), .shot(alien\_6\_shot),.background(background));

////////////refresh the Screen//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

always@(posedge clk)

if (reset == 0)

begin

count <= 0;

x\_clr <= 8'b0;

y\_clr <= 8'b0;

end

else if (count <= 19199)

begin

colorplot = {SW[4], SW[3], SW[2]};

if(x\_clr == 160)

begin

y\_clr = y\_clr + 1;

yplot = y\_clr;

x\_clr <= 0;

end

else if (x\_clr < 160)

begin

xplot <= x\_clr;

x\_clr <= x\_clr + 1;

end

count <= count + 1;

end

////Draw all actors (Tank / Rocket / Alien)//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

else if(count <= 19435)

begin

xplot <= x\_coor[count-19199];

yplot <= y\_coor[count-19199];

colorplot <= pixColor[count-19199];

count <= count + 1;

end

else if (count <= 833334)

begin

count = count + 1;

end

else if (count >= 833334)

begin

count <= 0;

end

////VGA Adapter//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

assign LEDR[7:0] = count;

vga\_adapter VGA(.resetn(reset),.clock(CLOCK\_50),.colour(colorplot),.x(xplot),.y(yplot),.plot(1'b1),.VGA\_R(VGA\_R),.VGA\_G(VGA\_G),.VGA\_B(VGA\_B),.VGA\_HS(VGA\_HS),.VGA\_VS(VGA\_VS),.VGA\_BLANK(VGA\_BLANK\_N),.VGA\_SYNC(VGA\_SYNC\_N),.VGA\_CLK(VGA\_CLK));

endmodule

//7 Segment Display ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module seg0 (input c0, c1, c2, c3, output LED);

assign LED = (~c0 & ~c1 & ~c2 & c3) | (~c0 & c1 & ~c2 & ~c3) | (c0 & ~c1 & c2 & c3) | (c0 & c1 & ~c2 & c3);

endmodule

module seg1 (input c0, c1, c2, c3, output LED);

assign LED = (~c0 & c1 & ~c2 & c3) | (~c0 & c1 & c2 & ~c3) | (c0 & ~c1 & c2 & c3) | (c0 & c1 & ~c2 & ~c3) | (c0 & c1 & c2 & ~c3) | (c0 & c1 & c2 & c3);

endmodule

module seg2 (input c0, c1, c2, c3, output LED);

assign LED = (~c0 & ~c1 & c2 & ~c3) | (c0 & c1 & ~c2 & ~c3) | (c0 & c1 & c2 & ~c3) | (c0 & c1 & c2 & c3);

endmodule

module seg3 (input c0, c1, c2, c3, output LED);

assign LED = (~c0 & ~c1 & ~c2 & c3) | (~c0 & c1 & ~c2 & ~c3) | (~c0 & c1 & c2 & c3) | (c0 & ~c1 & c2 & ~c3) | (c0 & c1 & c2 & c3);

endmodule

module seg4 (input c0, c1, c2, c3, output LED);

assign LED = (~c0 & ~c1 & ~c2 & c3) | (~c0 & ~c1 & c2 & c3) | (~c0 & c1 & ~c2 & ~c3) | (~c0 & c1 & ~c2 & c3) | (~c0 & c1 & c2 & c3) | (c0 & ~c1 & ~c2 & c3);

endmodule

module seg5 (input c0, c1, c2, c3, output LED);

assign LED = (~c0 & ~c1 & ~c2 & c3) | (~c0 & ~c1 & c2 & ~c3) | (~c0 & ~c1 & c2 & c3) | (~c0 & c1 & c2 & c3) | (c0 & c1 & ~c2 & c3);

endmodule

module seg6 (input c0, c1, c2, c3, output LED);

assign LED = (~c0 & ~c1 & ~c2 & ~c3) | (~c0 & ~c1 & ~c2 & c3) | (~c0 & c1 & c2 & c3) | (c0 & c1 & ~c2 & ~c3);

endmodule

module Sevenseg(Se7[0], Se7[1], Se7[2], Se7[3], FHEX0);

input [3:0]Se7;

output [6:0] FHEX0;

seg0 U1(.c0(Se7[0]), .c1(Se7[1]), .c2(Se7[2]), .c3(Se7[3]), .LED(FHEX0[0]));

seg1 U2(.c0(Se7[0]), .c1(Se7[1]), .c2(Se7[2]), .c3(Se7[3]), .LED(FHEX0[1]));

seg2 U3(.c0(Se7[0]), .c1(Se7[1]), .c2(Se7[2]), .c3(Se7[3]), .LED(FHEX0[2]));

seg3 U4(.c0(Se7[0]), .c1(Se7[1]), .c2(Se7[2]), .c3(Se7[3]), .LED(FHEX0[3]));

seg4 U5(.c0(Se7[0]), .c1(Se7[1]), .c2(Se7[2]), .c3(Se7[3]), .LED(FHEX0[4]));

seg5 U6(.c0(Se7[0]), .c1(Se7[1]), .c2(Se7[2]), .c3(Se7[3]), .LED(FHEX0[5]));

seg6 U7(.c0(Se7[0]), .c1(Se7[1]), .c2(Se7[2]), .c3(Se7[3]), .LED(FHEX0[6]));

endmodule

//Rate Divider////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module rateDivider(CLOCK\_50, N, reset, enable);

parameter width = 26;

parameter ratio = 1;

input CLOCK\_50, reset;

input [width-1:0] N;

reg [width-1:0] count;

output enable;

reg clkRate;

always@(posedge CLOCK\_50)

if (reset == 0)

begin

count <= 49999999;

clkRate <= 1'b0;

end

else if(count == N)

begin

count <= 49999999;

clkRate <= 1'b1;

end

else if(count != N)

begin

clkRate <= 1'b0;

count <= count - ratio;

end

assign enable = clkRate;

endmodule

//Alien module////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module alienModule(clk, enable, reset, alien\_initial\_x, alien\_initial\_y, drop, x\_coordinates, y\_coordinates, alien\_colour, x\_out, y\_out, shot, background);

input clk, enable, reset;

input [7:0]alien\_initial\_x;

input [7:0]alien\_initial\_y;

input drop;

input shot;

output [0:7] x\_coordinates [34:0];

output [0:7] y\_coordinates [34:0];

output [0:2] alien\_colour [34:0];

output [7:0] x\_out;

output [7:0] y\_out;

assign x\_out = alien\_x\_position;

assign y\_out = alien\_y\_position;

//assign Colors;

reg [2:0]red;

reg [2:0]green;

reg [2:0]blue;

wire [2:0]black;

input [2:0]background;

assign black = background;

reg [7:0]alien\_x\_position;

reg [7:0]alien\_y\_position;

always@(\*)

if(shot == 1)

begin

blue = black;

red = black;

green = black;

end

else

begin

red = 3'b100;

green = 3'b010;

blue = 3'b001;

end

//define the position arrays of the alien

assign x\_coordinates[0] = alien\_x\_position;

assign y\_coordinates[0] = alien\_y\_position;

assign x\_coordinates[1] = alien\_x\_position + 1;

assign y\_coordinates[1] = alien\_y\_position;

assign x\_coordinates[2] = alien\_x\_position + 2;

assign y\_coordinates[2] = alien\_y\_position;

assign x\_coordinates[3] = alien\_x\_position + 3;

assign y\_coordinates[3] = alien\_y\_position;

assign x\_coordinates[4] = alien\_x\_position + 4;

assign y\_coordinates[4] = alien\_y\_position;

assign x\_coordinates[5] = alien\_x\_position + 5;

assign y\_coordinates[5] = alien\_y\_position;

assign x\_coordinates[6] = alien\_x\_position + 6;

assign y\_coordinates[6] = alien\_y\_position;

assign x\_coordinates[7] = alien\_x\_position;

assign y\_coordinates[7] = alien\_y\_position + 1;

assign x\_coordinates[8] = alien\_x\_position + 1;

assign y\_coordinates[8] = alien\_y\_position + 1;

assign x\_coordinates[9] = alien\_x\_position + 2;

assign y\_coordinates[9] = alien\_y\_position + 1;

assign x\_coordinates[10] = alien\_x\_position + 3;

assign y\_coordinates[10] = alien\_y\_position + 1;

assign x\_coordinates[11] = alien\_x\_position + 4;

assign y\_coordinates[11] = alien\_y\_position + 1;

assign x\_coordinates[12] = alien\_x\_position + 5;

assign y\_coordinates[12] = alien\_y\_position + 1;

assign x\_coordinates[13] = alien\_x\_position + 6;

assign y\_coordinates[13] = alien\_y\_position + 1;

assign x\_coordinates[14] = alien\_x\_position;

assign y\_coordinates[14] = alien\_y\_position + 2;

assign x\_coordinates[15] = alien\_x\_position + 1;

assign y\_coordinates[15] = alien\_y\_position + 2;

assign x\_coordinates[16] = alien\_x\_position + 2;

assign y\_coordinates[16] = alien\_y\_position + 2;

assign x\_coordinates[17] = alien\_x\_position + 3;

assign y\_coordinates[17] = alien\_y\_position + 2;

assign x\_coordinates[18] = alien\_x\_position + 4;

assign y\_coordinates[18] = alien\_y\_position + 2;

assign x\_coordinates[19] = alien\_x\_position + 5;

assign y\_coordinates[19] = alien\_y\_position + 2;

assign x\_coordinates[20] = alien\_x\_position + 6;

assign y\_coordinates[20] = alien\_y\_position + 2;

assign x\_coordinates[21] = alien\_x\_position;

assign y\_coordinates[21] = alien\_y\_position + 3;

assign x\_coordinates[22] = alien\_x\_position + 1;

assign y\_coordinates[22] = alien\_y\_position + 3;

assign x\_coordinates[23] = alien\_x\_position + 2;

assign y\_coordinates[23] = alien\_y\_position + 3;

assign x\_coordinates[24] = alien\_x\_position + 3;

assign y\_coordinates[24] = alien\_y\_position + 3;

assign x\_coordinates[25] = alien\_x\_position + 4;

assign y\_coordinates[25] = alien\_y\_position + 3;

assign x\_coordinates[26] = alien\_x\_position + 5;

assign y\_coordinates[26] = alien\_y\_position + 3;

assign x\_coordinates[27] = alien\_x\_position + 6;

assign y\_coordinates[27] = alien\_y\_position + 3;

assign x\_coordinates[28] = alien\_x\_position;

assign y\_coordinates[28] = alien\_y\_position + 4;

assign x\_coordinates[29] = alien\_x\_position + 1;

assign y\_coordinates[29] = alien\_y\_position + 4;

assign x\_coordinates[30] = alien\_x\_position + 2;

assign y\_coordinates[30] = alien\_y\_position + 4;

assign x\_coordinates[31] = alien\_x\_position + 3;

assign y\_coordinates[31] = alien\_y\_position + 4;

assign x\_coordinates[32] = alien\_x\_position + 4;

assign y\_coordinates[32] = alien\_y\_position + 4;

assign x\_coordinates[33] = alien\_x\_position + 5;

assign y\_coordinates[33] = alien\_y\_position + 4;

assign x\_coordinates[34] = alien\_x\_position + 6;

assign y\_coordinates[34] = alien\_y\_position + 4;

//Draw the Alien (colour)

assign alien\_colour[0] = blue;

assign alien\_colour[1] = blue;

assign alien\_colour[2] = red;

assign alien\_colour[3] = blue;

assign alien\_colour[4] = red;

assign alien\_colour[5] = blue;

assign alien\_colour[6] = blue;

assign alien\_colour[7] = blue;

assign alien\_colour[8] = green;

assign alien\_colour[9] = green;

assign alien\_colour[10] = green;

assign alien\_colour[11] = green;

assign alien\_colour[12] = green;

assign alien\_colour[13] = blue;

assign alien\_colour[14] = black;

assign alien\_colour[15] = blue;

assign alien\_colour[16] = green;

assign alien\_colour[17] = red;

assign alien\_colour[18] = green;

assign alien\_colour[19] = blue;

assign alien\_colour[20] = black;

assign alien\_colour[21] = black;

assign alien\_colour[22] = black;

assign alien\_colour[23] = blue;

assign alien\_colour[24] = red;

assign alien\_colour[25] = blue;

assign alien\_colour[26] = black;

assign alien\_colour[27] = black;

assign alien\_colour[28] = black;

assign alien\_colour[29] = black;

assign alien\_colour[30] = black;

assign alien\_colour[31] = blue;

assign alien\_colour[32] = black;

assign alien\_colour[33] = black;

assign alien\_colour[34] = black;

reg dropUpdate;

reg dropInProgress;

wire dropClear;

reg [8:0]alien\_count;

////Alien Movement//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

always@(\*)

if(drop == 1)

begin

dropUpdate = 1;

end

else

begin

dropUpdate = dropClear;

end

always@(posedge clk)

if(reset == 0)

begin

alien\_count = 0;

alien\_x\_position = alien\_initial\_x;

alien\_y\_position = alien\_initial\_y;

end

else if (dropUpdate == 1 && enable == 1)

begin

dropInProgress <= 1;

alien\_x\_position <= alien\_x\_position;

alien\_y\_position <= alien\_y\_position + 1;

alien\_count <= alien\_count + 1;

end

else if(alien\_y\_position == 8'b1110111)

begin

alien\_count <= 0;

dropInProgress <= 0;

alien\_y\_position <= 0;

end

assign dropClear = dropInProgress;

endmodule

////Rocket hit detection module//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module hit(rocket\_x\_position, rocket\_y\_position, reset, alien\_x\_position, alien\_y\_position, shot, clk);

input clk;

input reset;

input [7:0]rocket\_x\_position;

input [7:0]rocket\_y\_position;

input [7:0]alien\_x\_position;

input [7:0]alien\_y\_position;

reg [7:0]alien\_x\_hit\_position;

reg [7:0]alien\_y\_hit\_position;

output reg shot;

always@(\*)

begin

alien\_x\_hit\_position <= alien\_x\_position;

alien\_y\_hit\_position <= alien\_y\_position;

end

always@(\*)

if(reset == 0)

shot <= 0;

else if ((rocket\_x\_position == alien\_x\_hit\_position) && ((rocket\_y\_position == alien\_y\_hit\_position) || ((rocket\_y\_position - 1) == alien\_y\_hit\_position)))

shot <= 1;

else if ((rocket\_x\_position == (alien\_x\_hit\_position + 1)) && ((rocket\_y\_position == alien\_y\_hit\_position) || ((rocket\_y\_position - 1) == alien\_y\_hit\_position)))

shot <= 1;

else if ((rocket\_x\_position == (alien\_x\_hit\_position + 2)) && ((rocket\_y\_position == alien\_y\_hit\_position) || ((rocket\_y\_position - 1) == alien\_y\_hit\_position)))

shot <= 1;

else if ((rocket\_x\_position == (alien\_x\_hit\_position + 3)) && ((rocket\_y\_position == alien\_y\_hit\_position) || ((rocket\_y\_position - 1) == alien\_y\_hit\_position)))

shot <= 1;

else if ((rocket\_x\_position == (alien\_x\_hit\_position + 4)) && ((rocket\_y\_position == alien\_y\_hit\_position) || ((rocket\_y\_position - 1) == alien\_y\_hit\_position)))

shot <= 1;

else if ((rocket\_x\_position == (alien\_x\_hit\_position + 5)) && ((rocket\_y\_position == alien\_y\_hit\_position) || ((rocket\_y\_position - 1) == alien\_y\_hit\_position)))

shot <= 1;

else if ((rocket\_x\_position == (alien\_x\_hit\_position + 6)) && ((rocket\_y\_position == alien\_y\_hit\_position) || ((rocket\_y\_position - 1) == alien\_y\_hit\_position)))

shot <= 1;

endmodule

////Tank hit detection module//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module Tankhit(x\_position, y\_position, reset, alien\_x\_position, alien\_y\_position, shot, clk);

input clk;

input reset;

input [7:0]x\_position;

input [7:0]y\_position;

input [7:0]alien\_x\_position;

input [7:0]alien\_y\_position;

reg [7:0]alien\_x\_hit\_position;

reg [7:0]alien\_y\_hit\_position;

output reg shot;

always@(\*)

begin

alien\_x\_hit\_position <= alien\_x\_position;

alien\_y\_hit\_position <= alien\_y\_position;

end

always@(\*)

if(reset == 0)

begin

shot = 0;

end

else if(((x\_position) == (alien\_x\_hit\_position)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position) == (alien\_x\_hit\_position + 1)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position) == (alien\_x\_hit\_position + 2)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position) == (alien\_x\_hit\_position + 3)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position) == (alien\_x\_hit\_position + 4)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position) == (alien\_x\_hit\_position + 5)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position) == (alien\_x\_hit\_position + 6)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 2) == (alien\_x\_hit\_position)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 2) == (alien\_x\_hit\_position + 1)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 2) == (alien\_x\_hit\_position + 2)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 2) == (alien\_x\_hit\_position + 3)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 2) == (alien\_x\_hit\_position + 4)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 2) == (alien\_x\_hit\_position + 5)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 2) == (alien\_x\_hit\_position + 6)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 3) == (alien\_x\_hit\_position)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 3) == (alien\_x\_hit\_position + 1)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 3) == (alien\_x\_hit\_position + 2)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 3) == (alien\_x\_hit\_position + 3)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 3) == (alien\_x\_hit\_position + 4)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 3) == (alien\_x\_hit\_position + 5)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 3) == (alien\_x\_hit\_position + 6)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 4) == (alien\_x\_hit\_position)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 4) == (alien\_x\_hit\_position + 1)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 4) == (alien\_x\_hit\_position + 2)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 4) == (alien\_x\_hit\_position + 3)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 4) == (alien\_x\_hit\_position + 4)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 4) == (alien\_x\_hit\_position + 5)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

else if(((x\_position + 4) == (alien\_x\_hit\_position + 6)) && (alien\_y\_hit\_position == 111))

begin

shot = 1;

end

endmodule

//Score counter module

module scoreCount(alien\_1\_shot, alien\_2\_shot, alien\_3\_shot, alien\_4\_shot, alien\_5\_shot, alien\_6\_shot, reset, count);

input reset;

input alien\_1\_shot;

input alien\_2\_shot;

input alien\_3\_shot;

input alien\_4\_shot;

input alien\_5\_shot;

input alien\_6\_shot;

output reg [3:0]count;

always@(\*)

if (reset == 0)

begin

count = 4'b0;

end

else

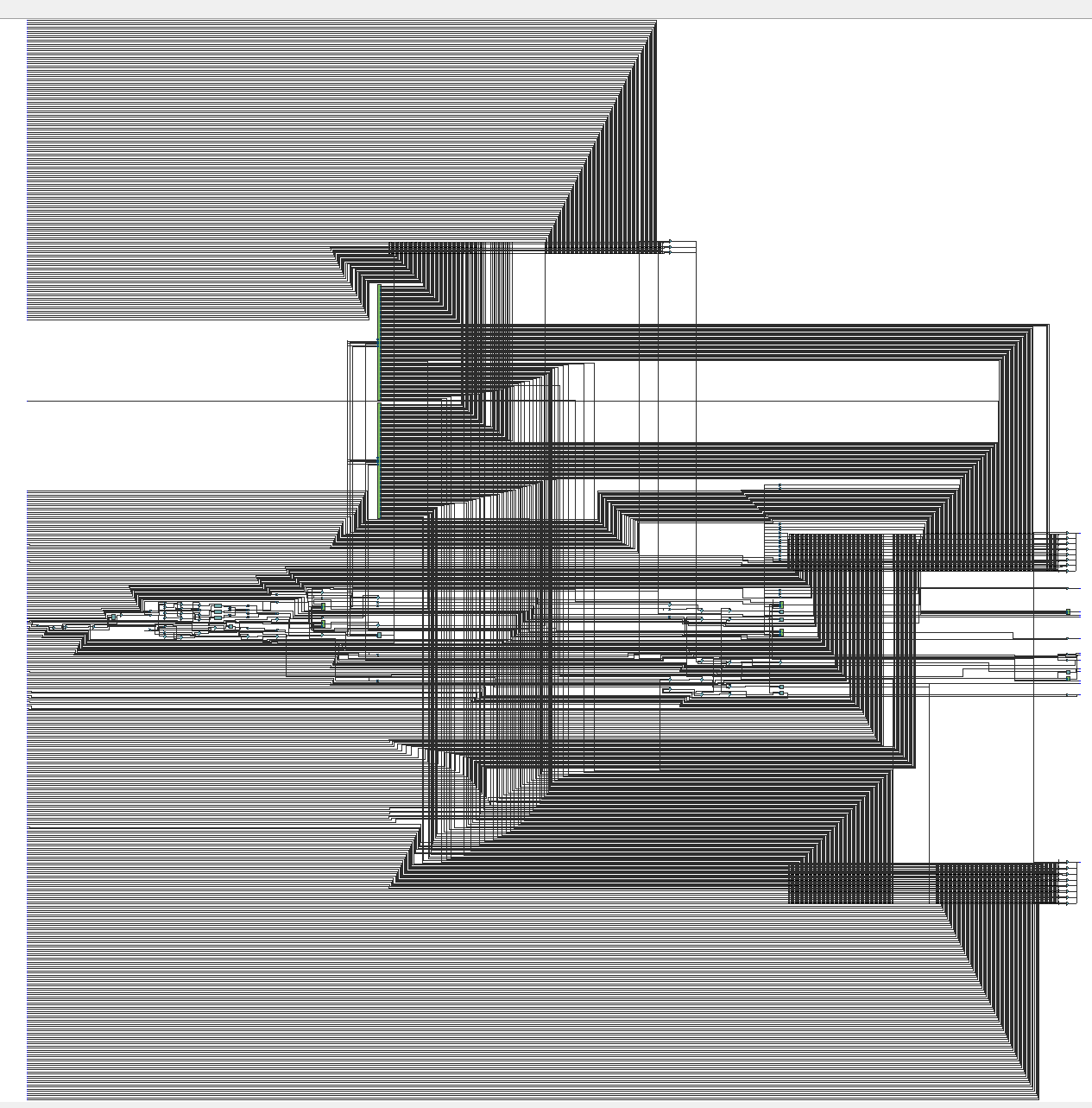
begin

count = alien\_1\_shot + alien\_2\_shot + alien\_3\_shot + alien\_4\_shot + alien\_5\_shot + alien\_6\_shot;

end

endmodule

**B - RTL Viewer**

****