PROJECT: BEE INVADERS

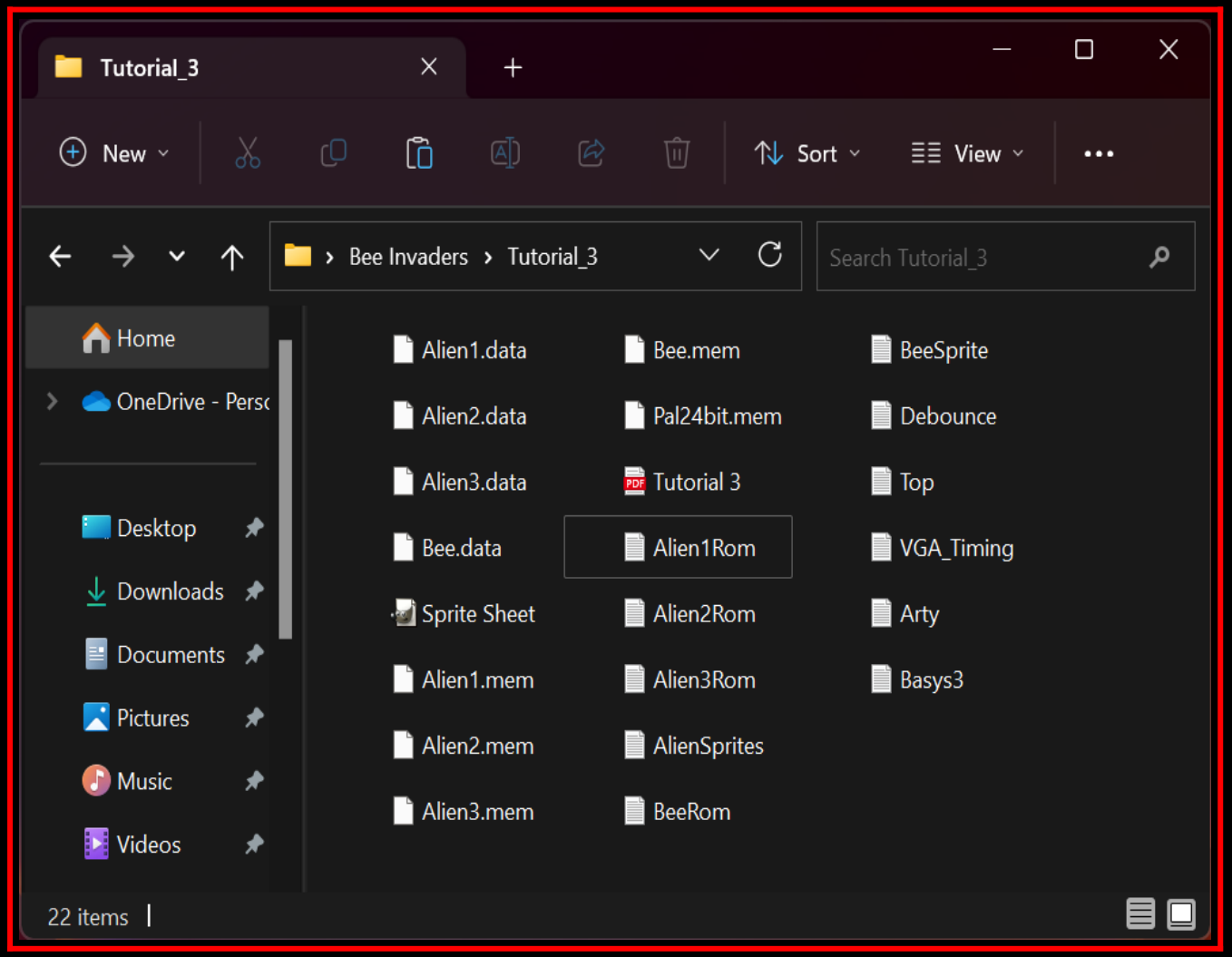
This Tutorial Is For The
Basys3 FPGA Board Or The Arty A7-35 FPGA Board With A VGA Pmod Connected
But Can Be Adapted To Other FPGA Boards
A Modern Version Of The Popular Arcade Game
Space Invaders

Tutorial 3 - Move The Bee Left / Right And Display The 55 Alien Bees On The Screen

CONTENTS

(A) USING GIMP TO GENERATE THE GRAPHICS FOR THE ALIENS

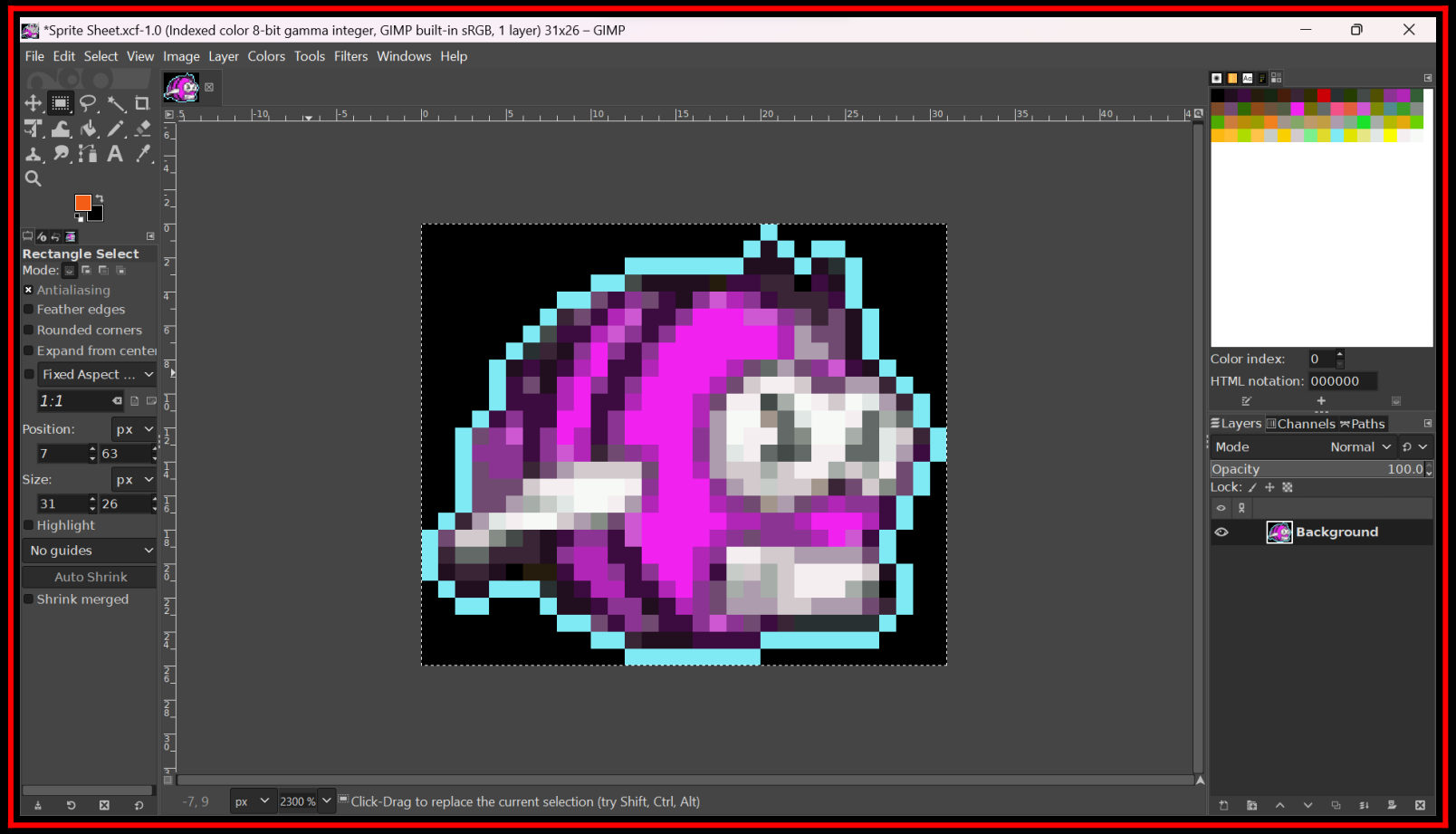
01 In the folder "Bee Invaders" create a folder called "Tutorial_3" and extract the files from the downloaded file "Tutorial_3 Files.zip" to this folder



02

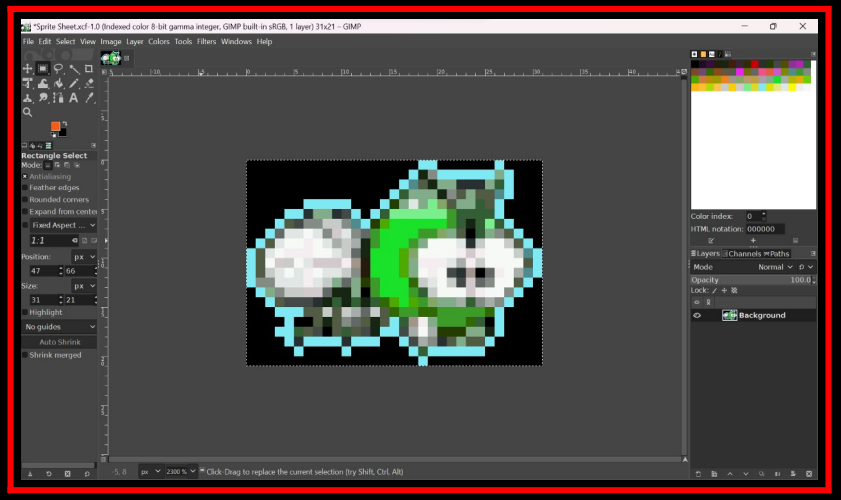
The files for the Aliens are in the files which were extracted in Step 01, so jump to section (B) if you do not wish to see how the files were made in Gimp

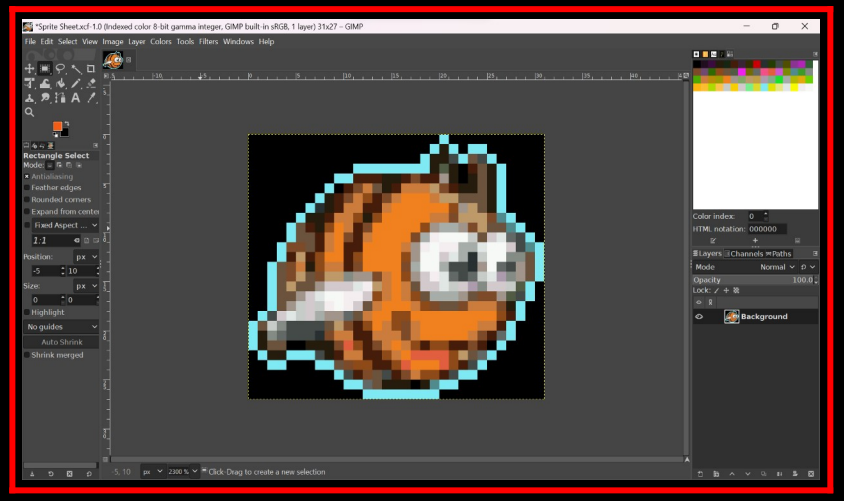
Open "Sprite Sheet.xcf" in the "Tutorial_3" folder with Gimp, convert it to 64 colours (Image → Mode → Indexed), set the maximum number of colours to 64 and make sure that Remove unused and duplicate colors from colormap is not selected, then select "Convert". Zoom in on the first Alien character and using the "rectangle select tool" select around the Alien (this should be a rectangle 31 x 26 pixels) and crop it



03 The image needs to be saved as a Raw Data File, do this using File → Export As → Raw image data. Call the file "Alien1.data"

Do the same for the second and third Aliens, saving them as "Alien2.data" (31 x 21 pixels) and "Alien3.data" (31 x 27 pixels)





Using HxD Hex Editor (or similar) load the file "Alien1.data", select all the data and copy it

Then paste the data into a Notepad file and save it as "Alien1.mem"

Do the same for "Alien2.data" and "Alien3.data" and save them as "Alien2.mem" and "Alien3.mem"

(B) CREATING THE PROJECT IN VIVADO

01 Follow the instructions in "Tutorial 1" to create a new project in the "Tutorial_3" folder in Vivado but call it "BeeInvaders_WIP"

Add these design sources from the "Tutorial 3" folder:

Top.v
VGA_Timing.v
BeeSprite.v
AlienSprites.v
BeeRom.v
Alien1Rom.v
Alien2Rom.v
Alien3Rom.v
Bee.mem
Alien1.mem
Alien2.mem
Alien3.mem
Pal24bit.mem
Debounce.v

Add a constraints file from the "Tutorial 3" folder:

Basys3.xdc for the Basys3 board
Arty.xdc for the Arty A7-35 board

Create the 25.2MHz pixel clock as we did in "Tutorial 1"

For this to work on the Arty A7-35 all you need to do is replace this line in "Top.v":

```
.reset(btn_rst_n),      // reset button is active high
```

With:

```
.reset(!btn_rst_n),      // reset button is active low
```

02

Click on "Run Synthesis" and when the window "Synthesis Completed" appears ensure "Run implementation" is selected and click "OK". When the "Implementation Completed" window appears select "Generate Bitstream" and click "OK"

Now select "Open Hardware Manager" and click "OK". Next click "Open Target" and select "Auto Connect". Now click "Program Device". When the "Program Device" box appears make sure the "Bitsteam file" path is correct and then click "Program".

You should see our Bee and the 55 Aliens on your VGA monitor, as shown below

Press the left or right button on the FPGA board to move to Bee

(C) THE CODE FOR THIS TUTORIAL

This is the code from the file "Top.v"

```
//------------------------------------------------
// Top.v module
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//------------------------------------------------

`default_nettype none
`timescale 1ns / 1ps

// Setup Top module
module Top (
    input  wire clk_100m,                   // 100 MHz clock
    input  wire btn_rst_n,                  // reset button
    output wire vga_hsync,                  // VGA horizontal sync
    output wire vga_vsync,                  // VGA vertical sync
    output reg [3:0] vga_r,                 // 4-bit VGA red
    output reg [3:0] vga_g,                 // 4-bit VGA green
    output reg [3:0] vga_b,                 // 4-bit VGA blue
    input wire btnR,                        // Right button
    input wire btnL                         // Left button
    );

    // Instantiate VGA_Clock
    reg reset;                              // Reset Button
    wire clk_pix;                           // 25.2Mhz Pixel clock
    wire clk_pix_locked;                    // Pixel clock locked?

    VGA_Clock clock_pix_inst (
       .clk_100m(clk_100m),
       .reset(btn_rst_n),                   // reset button is active high
       .clk_pix(clk_pix),
       .clk_pix_locked(clk_pix_locked)
    );

    // Instantiate VGA_Timing
    localparam CORDW = 10;                  // screen coordinate width in bits
   reg rst_pix;
    wire [CORDW-1:0] sx, sy;
    wire hsync;
    wire vsync;
    wire de;
    VGA_Timing display_inst (
        .clk_pix(clk_pix),
        .rst_pix(!clk_pix_locked),          // wait for clock lock
        .sx(sx),
        .sy(sy),
        .hsync(hsync),
        .vsync(vsync),
        .de(de)
    );
```

```
// Instantiate BeeSprite
wire [1:0] BeeSpriteOn;                 // 1=on, 0=off
wire [7:0] dout;                        // pixel value from Bee.mem
BeeSprite BeeDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .BeeSpriteOn(BeeSpriteOn),
    .btnR(btnR),
    .btnL(btnL),
    .dataout(dout)
);

// Instantiate AlienSprites
wire [1:0] Alien1SpriteOn;              // 1=on, 0=off
wire [1:0] Alien2SpriteOn;              // 1=on, 0=off
wire [1:0] Alien3SpriteOn;              // 1=on, 0=off
wire [7:0] Alien1dout;                  // pixel value from Alien1.mem
wire [7:0] Alien2dout;                  // pixel value from Alien2.mem
wire [7:0] Alien3dout;                  // pixel value from Alien3.mem
AlienSprites AlienDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .Alien1SpriteOn(Alien1SpriteOn),
    .Alien2SpriteOn(Alien2SpriteOn),
    .Alien3SpriteOn(Alien3SpriteOn),
    .A1dout(Alien1dout),
    .A2dout(Alien2dout),
    .A3dout(Alien3dout)
);

// Load colour palette
reg [7:0] palette [0:191];              // 8 bit values from the 192 hex entries in the colour palette
reg [7:0] COL = 0;                      // background colour palette value
initial begin
    $readmemh("pal24bit.mem", palette); // load 192 hex values into "palette"
end

// VGA Output
assign vga_hsync = hsync;
assign vga_vsync = vsync;
always @ (posedge clk_pix)
begin
    if(de)
        begin
            if (BeeSpriteOn==1)
                begin
                    vga_r <= (palette[(dout*3)])>>4;            // RED bits(7:4) from colour palette
                    vga_g <= (palette[(dout*3)+1])>>4;          // GREEN bits(7:4) from colour palette
                    vga_b <= (palette[(dout*3)+2])>>4;          // BLUE bits(7:4) from colour palette
                end
            else
            if (Alien1SpriteOn==1)
                begin
                    vga_r <= (palette[(Alien1dout*3)])>>4;      // RED bits(7:4) from colour palette
                    vga_g <= (palette[(Alien1dout*3)+1])>>4;    // GREEN bits(7:4) from colour palette
                    vga_b <= (palette[(Alien1dout*3)+2])>>4;    // BLUE bits(7:4) from colour palette
                end
```

```
                else
                if (Alien2SpriteOn==1)
                    begin
                        vga_r <= (palette[(Alien2dout*3)])>>4;      // RED bits(7:4) from colour palette
                        vga_g <= (palette[(Alien2dout*3)+1])>>4;    // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(Alien2dout*3)+2])>>4;    // BLUE bits(7:4) from colour palette
                    end
                else
                if (Alien3SpriteOn==1)
                    begin
                        vga_r <= (palette[(Alien3dout*3)])>>4;      // RED bits(7:4) from colour palette
                        vga_g <= (palette[(Alien3dout*3)+1])>>4;    // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(Alien3dout*3)+2])>>4;    // BLUE bits(7:4) from colour palette
                    end
                else
                    begin
                        vga_r <= (palette[(COL*3)])>>4;             // RED bits(7:4) from colour palette
                        vga_g <= (palette[(COL*3)+1])>>4;           // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(COL*3)+2])>>4;           // BLUE bits(7:4) from colour palette
                    end
            end
        else
            begin
                vga_r <= 0; // set RED, GREEN & BLUE
                vga_g <= 0; // to "0" when x,y outside of
                vga_b <= 0; // the active display area
            end
    end
endmodule
```

This is the code from the file "VGA_Timing.v"

```
//---------------------------------------------
// VGA_Timing.v module
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//---------------------------------------------

`default_nettype none
`timescale 1ns / 1ps


module VGA_Timing (
       input  wire clk_pix,   // pixel clock
       input  wire rst_pix,   // reset in pixel clock domain
       output reg [9:0] sx,   // horizontal screen position
       output reg [9:0] sy,   // vertical screen position
       output wire hsync,     // horizontal sync
       output wire vsync,     // vertical sync
       output wire de         // data enable (low in blanking interval)
       );

       // horizontal timings
       parameter HA_END = 639;           // end of active pixels
       parameter HS_STA = HA_END + 16;   // sync starts after front porch
       parameter HS_END = HS_STA + 96;   // sync ends
       parameter LINE   = 799;           // last pixel on line (after back porch)
```

```
    // vertical timings
    parameter VA_END = 479;           // end of active pixels
    parameter VS_STA = VA_END + 10;   // sync starts after front porch
    parameter VS_END = VS_STA + 2;    // sync ends
    parameter SCREEN = 524;           // last line on screen (after back porch)

    assign hsync = ~(sx >= HS_STA && sx < HS_END);  // invert: negative polarity
    assign vsync = ~(sy >= VS_STA && sy < VS_END);  // invert: negative polarity
    assign de = (sx <= HA_END && sy <= VA_END);

    // calculate horizontal and vertical screen position
    always @(posedge clk_pix) begin
        if (sx == LINE) begin  // last pixel on line?
            sx <= 0;
            sy <= (sy == SCREEN) ? 0 : sy + 1;  // last line on screen?
        end else begin
            sx <= sx + 1;
        end
        if (rst_pix) begin
            sx <= 0;
            sy <= 0;
        end
    end
endmodule
```

This is the code from the file "BeeSprite.v"

```
//-----------------------------------------------
// BeeSprite.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup BeeSprite module
module BeeSprite(
    input wire clk_pix,
    input wire [9:0] sx,
    input wire [9:0] sy,
    input wire de,
    output reg [1:0] BeeSpriteOn,   // 1=on, 0=off
    input wire btnR,                // right button
    input wire btnL,                // left button
    output wire [7:0] dataout
    );

    // instantiate BeeRom code
    reg [9:0] address;              // 2^10 or 1024, need 34 x 27 = 918
    BeeRom BeeVRom (
        .address(address),
        .clk_pix(clk_pix),
        .dataout(dataout)
    );

    // Instantiate Debounce
    wire sig_right;
```

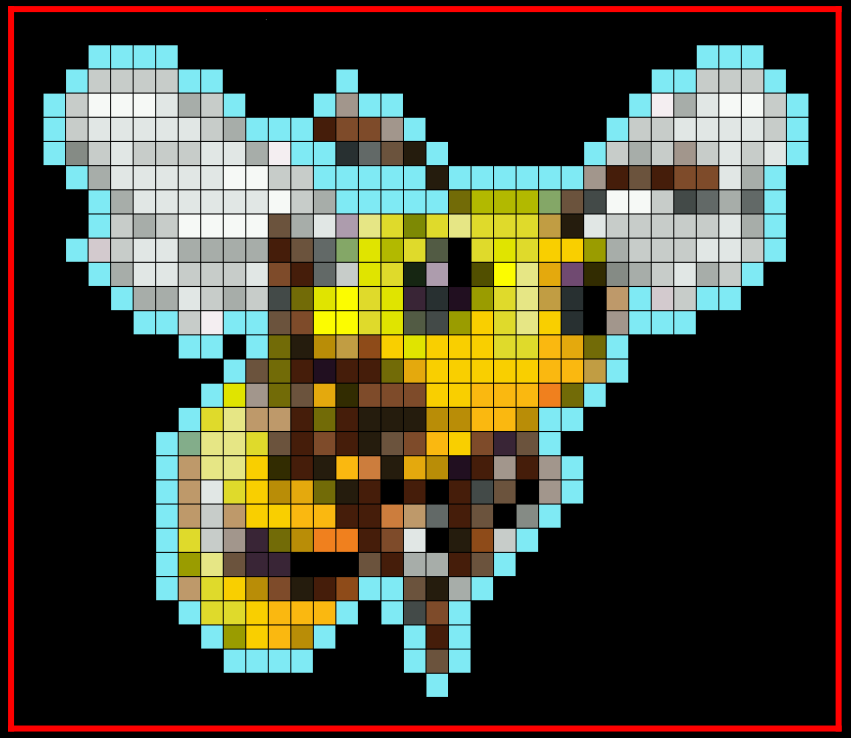
```
    wire sig_left;

    Debounce deb_right (
        .clk_pix(clk_pix),
        .btn(btnR),
        .out(sig_right)
    );

    Debounce deb_left (
        .clk_pix(clk_pix),
        .btn(btnL),
        .out(sig_left)
    );

    // setup character positions and sizes
    reg [9:0] BeeX = 297;           // Bee X start position
    reg [8:0] BeeY = 433;           // Bee Y start position
    localparam BeeWidth = 34;       // Bee width in pixels
    localparam BeeHeight = 27;      // Bee height in pixels

    always @ (posedge clk_pix)
    begin
        // if sx,sy are within the confines of the Bee character, switch the Bee On and
        if(de)
            begin
                if(sx==BeeX-2 && sy==BeeY)                  // Initially sx = 295 (297 - 2) = 1 pixel
                    begin
                        address <= 0;                       // Initially address = 0
                        BeeSpriteOn <=1;
                    end
                if((sx>BeeX-2) && (sx<BeeX+BeeWidth-1) && (sy>BeeY-1) && (sy<BeeY+BeeHeight)) // Thereafter sx = 296 to 329 = 33 pixels
                    begin
                        address <= address +1;              // Secondly address = (296 + 2 - 297) + (0 * 34) = 1
                        BeeSpriteOn <=1;
                    end
                else
                    BeeSpriteOn <=0;
            end
        // if left or right button pressed, move the Bee
        if (sx==639 && sy==479)                             // check for movement once every frame
            begin
                if (sig_right == 1 && BeeX<640-BeeWidth)    // Check for right button
                    BeeX<=BeeX+1;
                else
                if (sig_left == 1 && BeeX>2)                // Check for left button
                    BeeX<=BeeX-1;
            end
    end
endmodule
```

This is the code from the file "BeeRom.v"

```
//----------------------------------------------
// BeeRom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps

// Setup BeeRom module
module BeeRom(
    input wire [9:0] address, // (9:0) or 2^10 or 1024, need 34 x 27 = 918
    input wire clk_pix,
    output reg [7:0] dataout // (7:0) 8 bit pixel value from Bee.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] memory_array [0:917]; // 8 bit values for 918 pixels of Bee (34 x 27)

    initial
    begin
        $readmemh("Bee.mem", memory_array);
    end

    always@(posedge clk_pix)
            dataout <= memory_array[address];
endmodule
```

This is the data from the file "Bee.mem" - Sprite Size 34 x 27 pixels

```
00 00 39 39 39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 39 00 00 00 39 34 34 34 34 39 39 00 00 00 00 00 39 00 00 00 00 00
00 00 00 00 00 00 00 00 39 39 34 34 34 39 00 39 34 3F 3F 3F 3C 2C 34 39 00 00 00 39 25 39 39 00 00 00 00 00 00 00 00 00 00 39 3E 2C 3C 3F 3F 34 39 39 34 3C 3C
3C 3C 3C 34 2C 39 39 39 05 10 10 25 39 00 00 00 00 00 00 00 00 00 39 34 34 3C 3C 3C 3C 34 39 39 1F 34 3C 34 34 34 3C 3C 2C 3E 39 39 09 18 14 03 39 00 00 00 00 00
00 39 34 2C 34 25 34 3C 34 3C 39 00 39 2C 3C 3C 3C 3C 3C 3F 3F 34 34 39 39 39 39 39 39 03 39 39 39 39 39 39 39 25 05 14 05 10 10 3C 2C 39 00 00 00 39 2C 3C 3C 3C 3C
3C 3C 3F 34 2C 39 39 39 39 39 17 2D 2D 2D 26 14 0B 3F 3F 34 0B 18 2C 18 39 00 00 00 39 34 2C 34 3F 3F 3F 3F 14 2C 3C 29 3B 38 1C 38 3B 38 38 38 28 03 3C 34 34
34 34 34 3C 2C 39 00 00 00 39 36 34 3C 3C 2C 2C 2C 2C 05 14 18 26 3A 2D 38 15 00 38 3A 38 35 35 23 2C 34 34 34 3C 3C 34 39 00 00 00 39 2C 3C 3C 34 34 34 3C 10 05
18 34 3A 38 04 29 00 0C 3D 3B 2E 11 07 1F 2C 34 3C 2C 34 39 00 00 00 00 00 39 2C 2C 3C 34 2C 34 0B 17 3A 3D 38 3A 06 09 01 23 38 3B 28 09 00 27 39 36 34 39 39
00 00 00 00 00 00 00 00 39 39 34 3E 39 39 14 10 3D 3D 38 3A 15 0B 23 35 38 3B 35 09 00 25 39 39 39 00 00 00 00 00 00 00 00 00 00 00 39 39 00 39 17 03 22 28 13 35
3A 35 35 35 38 38 30 2E 17 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 14 17 05 01 05 05 17 2E 35 35 35 35 35 30 30 28 39 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 39 3A 25 17 14 2E 07 10 10 10 35 35 30 30 30 24 17 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 38 3B 27 27 05 17 05 03 03 03 22 22 30
30 22 39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 2A 3B 3B 38 14 05 10 05 03 14 10 30 35 10 06 14 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
39 27 3B 3B 35 07 05 03 30 21 03 2E 22 01 05 25 05 25 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 27 3C 38 35 22 2E 17 03 05 00 05 00 05 0B 14 00 25 39
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 27 34 27 35 35 30 30 05 05 21 27 18 05 14 00 1F 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 38 34 25
06 17 22 24 24 05 10 3C 00 03 13 34 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 23 3B 14 06 06 00 00 00 14 05 2C 2C 05 14 39 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 39 27 38 35 22 10 03 05 13 39 39 14 03 2C 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 38 38 35 30 30 30
39 00 39 0B 10 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 23 35 30 22 39 00 00 00 39 05 39 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 39 39 39 39 00 00 00 00 39 14 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



This is the code from the file “Debounce.v”

```
//----------------------------------------------
// Debounce.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps

// Setup Debounce module
module Debounce (
    input wire clk_pix,             // Clock signal to synchronize the button input
    input wire btn,
    output wire out
  );

  reg [19:0] ctr_d;                 // 20 bit counter to increment when button is pressed or released
  reg [19:0] ctr_q;                 // 20 bit counter to increment when button is pressed or released
  reg [1:0] sync_d;                 // button flip-flop for synchronization
  reg [1:0] sync_q;                 // button flip-flop for synchronization

  assign out = ctr_q == {20{1'b1}}; // if ctr_q = 11111111111111111111

  always @(*)
  begin
    sync_d[0] = btn;
    sync_d[1] = sync_q[0];
    ctr_d = ctr_q + 1'b1;

    if (ctr_q == {20{1'b1}})
      ctr_d = ctr_q;

    if (!sync_q[1])
      ctr_d = 20'd0;
  end

  always @(posedge clk_pix)
  begin
    ctr_q <= ctr_d;
    sync_q <= sync_d;
  end
endmodule
```

This is the code from the file "AlienSprites.v"

```
//----------------------------------------------
// AlienSprites.v module
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps

// Setup AlienSprites Module
module AlienSprites(
    input wire clk_pix,                 // 25MHz pixel clock
    input wire [9:0] sx,                // current x position
    input wire [9:0] sy,                // current y position
    input wire de,                      // high during active pixel drawing
    output reg Alien1SpriteOn,          // 1=on, 0=off
    output reg Alien2SpriteOn,          // 1=on, 0=off
    output reg Alien3SpriteOn,          // 1=on, 0=off
    output wire [7:0] A1dout,           // 8 bit pixel value from Alien1.mem
    output wire [7:0] A2dout,           // 8 bit pixel value from Alien2.mem
    output wire [7:0] A3dout            // 8 bit pixel value from Alien3.mem
    );

// instantiate Alien1Rom code
    reg [9:0] A1address;                // 2^10 or 1024, need 31 x 26 = 806
    Alien1Rom Alien1VRom (
        .A1address(A1address),
        .clk_pix(clk_pix),
        .A1dout(A1dout)
    );

// instantiate Alien2Rom code
    reg [9:0] A2address;                // 2^10 or 1024, need 31 x 21 = 651
    Alien2Rom Alien2VRom (
        .A2address(A2address),
        .clk_pix(clk_pix),
        .A2dout(A2dout)
    );

// instantiate Alien3Rom code
    reg [9:0] A3address;                // 2^10 or 1024, need 31 x 27 = 837
    Alien3Rom Alien3VRom (
        .A3address(A3address),
        .clk_pix(clk_pix),
        .A3dout(A3dout)
    );

// setup character positions and sizes
    reg [9:0] A1X = 135;                // Alien1 X start position
    reg [8:0] A1Y = 85;                 // Alien1 Y start position
    localparam A1Width = 31;            // Alien1 width in pixels
    localparam A1Height = 26;           // Alien1 height in pixels
    reg [9:0] A2X = 135;                // Alien2 X start position
    reg [8:0] A2Y = 120;                // Alien2 Y start position
    localparam A2Width = 31;            // Alien2 width in pixels
    localparam A2Height = 21;           // Alien2 height in pixels
    reg [9:0] A3X = 135;                // Alien3 X start position
    reg [8:0] A3Y = 180;                // Alien3 Y start position
    localparam A3Width = 31;            // Alien3 width in pixels
```

```
localparam A3Height = 27;           // Alien3 height in pixels

reg [9:0] AoX = 0;                  // Offset for X Position of next Alien in row
reg [8:0] AoY = 0;                  // Offset for Y Position of next row of Aliens
reg [9:0] AcounterW = 0;            // Counter to check if Alien width reached
reg [9:0] AcounterH = 0;            // Counter to check if Alien height reached
reg [3:0] AcolCount = 11;           // Number of horizontal aliens in all columns

always @ (posedge clk_pix)
begin
    if (de)
        begin
            // check if sx,sy are within the confines of the Alien characters
            // Alien1
            if (sx==A1X+AoX-2 && sy==A1Y+AoY)
                begin
                    A1address <= 0;
                    Alien1SpriteOn <=1;
                    AcounterW<=0;
                end
            if ((sx>A1X+AoX-2) && (sx<A1X+A1Width+AoX) && (sy>A1Y+AoY-1) && (sy<A1Y+A1Height+AoY))
                begin
                    A1address <= A1address + 1;
                    AcounterW <= AcounterW + 1;
                    Alien1SpriteOn <=1;
                    if (AcounterW==A1Width-1)
                        begin
                            AcounterW <= 0;
                            AoX <= AoX + 40;
                            if(AoX<(AcolCount-1)*40)
                           A1address <= A1address - (A1Width-1);
                         else
                         if(AoX==(AcolCount-1)*40)
                            AoX<=0;
                       end
                end
            else
                Alien1SpriteOn <=0;

            // Alien2
            if (sx==A2X+AoX-2 && sy==A2Y+AoY)
                begin
                    A2address <= 0;
                    Alien2SpriteOn <=1;
                    AcounterW<=0;
                end
            if ((sx>A2X+AoX-2) && (sx<A2X+A2Width+AoX) && (sy>A2Y+AoY-1) && (sy<A2Y+AoY+A2Height))
                begin
                    A2address <= A2address + 1;
                    AcounterW <= AcounterW + 1;
                    Alien2SpriteOn <=1;
                    if (AcounterW==A2Width-1)
                        begin
                            AcounterW <= 0;
                            AoX <= AoX + 40;
                            if(AoX<(AcolCount-1)*40)
                           A2address <= A2address - (A2Width-1);
                         else
                         if(AoX==(AcolCount-1)*40)
                                    begin
                               AoX<=0;
                               AcounterH <= AcounterH + 1;
```

```
                        if(AcounterH==A2Height-1)
                                    begin
                                AcounterH<=0;
                                AoY <= AoY + 30;
                                if(AoY==30)
                                    begin
                                       AoY<=0;
                                       AoX<=0;
                                       end
                                end
                            end
                    end
                else
                    Alien2SpriteOn <=0;

                // Alien3
                if (sx==A3X+AoX-2 && sy==A3Y+AoY)
                    begin
                        A3address <= 0;
                        Alien3SpriteOn <=1;
                        AcounterW<=0;
                        AcounterH<=0;
                    end
                if ((sx>A3X+AoX-2) && (sx<A3X+AoX+A3Width) && (sy>A3Y+AoY-1) && (sy<A3Y+AoY+A3Height))
                    begin
                        A3address <= A3address + 1;
                        AcounterW <= AcounterW + 1;
                        Alien3SpriteOn <=1;
                        if (AcounterW==A3Width-1)
                            begin
                                AcounterW <= 0;
                                AoX <= AoX + 40;
                                if(AoX<(AcolCount-1)*40)
                               A3address <= A3address - (A3Width-1);
                         else
                         if(AoX==(AcolCount-1)*40)
                                    begin
                                AoX<=0;
                                AcounterH <= AcounterH + 1;
                                if(AcounterH==A3Height-1)
                                            begin
                                        AcounterH<=0;
                                        AoY <= AoY + 36;
                                        if(AoY==36)
                                            begin
                                               AoY<=0;
                                               AoX<=0;
                                               end
                                        end
                                end
                           end
                    end
                else
                    Alien3SpriteOn <=0;
            end
    end
endmodule
```

This is the code from the file "Alien1Rom.v"

```
//-----------------------------------------------
// Alien1Rom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup Alien1Rom module
module Alien1Rom(
    input wire [9:0] A1address, // (9:0) or 2^10 or 1024, need 31 x 26 = 806
    input wire clk_pix,
    output reg [7:0] A1dout     // (7:0) 8 bit pixel value from Alien1.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] A1memory_array [0:805]; // 8 bit values for 806 pixels of Alien1 (31 x 26)

    initial
    begin
        $readmemh("Alien1.mem", A1memory_array);
    end

    always @ (posedge clk_pix)
            A1dout <= A1memory_array[A1address];
endmodule
```

This is the code from the file "Alien2Rom.v"

```
//-----------------------------------------------
// Alien2Rom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup Alien2Rom module
module Alien2Rom(
    input wire [9:0] A2address, // (9:0) or 2^10 or 1024, need 31 x 21 = 651
    input wire clk_pix,
    output reg [7:0] A2dout     // (7:0) 8 bit pixel value from Alien2.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] A2memory_array [0:650]; // 8 bit values for 651 pixels of Alien2 (31 x 21)

    initial
    begin
        $readmemh("Alien2.mem", A2memory_array);
    end

    always @ (posedge clk_pix)
            A2dout <= A2memory_array[A2address];
```

```
endmodule
```

This is the code from the file “Alien3Rom.v”

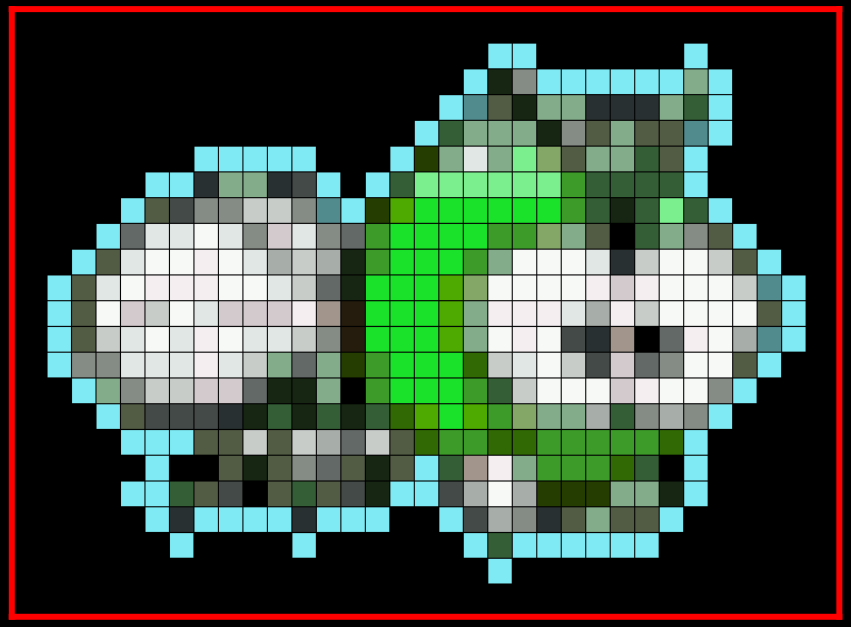
```
//----------------------------------------------
// Alien3Rom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps

// Setup Alien3Rom module
module Alien3Rom(
    input wire [9:0] A3address, // (9:0) or 2^10 or 1024, need 31 x 27 = 837
    input wire clk_pix,
    output reg [7:0] A3dout     // (7:0) 8 bit pixel value from Alien3.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] A3memory_array [0:836]; // 8 bit values for 837 pixels of Alien3 (31 x 27)

    initial
    begin
        $readmemh("Alien3.mem", A3memory_array);
    end

    always @ (posedge clk_pix)
            A3dout <= A3memory_array[A3address];
endmodule
```

This is the data from the file "Alien1.mem" - Sprite Size 31 x 26 pixels

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 01 39
00 39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 39 39 39 39 39 01 01 01 39 01 09 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 0B
01 01 01 01 03 02 02 06 06 00 02 01 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 11 02 0D 11 02 01 0D 1B 0E 0D 02 06 06 06 02 39 00 00 00 00 00 00 00 00 00
00 00 00 39 06 11 02 0E 1B 02 02 0D 16 16 16 1B 1B 0D 11 06 11 01 39 00 00 00 00 00 00 00 00 00 00 00 39 0B 02 02 0E 1B 0D 02 0D 16 16 16 16 16 16 0E 29 11 11 01
39 00 00 00 00 00 00 00 00 00 39 09 06 01 0D 16 0D 02 0D 16 16 16 16 16 16 16 0E 29 29 11 01 39 00 00 00 00 00 00 00 00 39 01 02 01 02 0E 16 02 02 0E 16 16 16
16 0E 11 18 29 11 0D 1B 02 01 39 00 00 00 00 00 00 00 00 39 02 11 01 0E 16 0D 02 0D 16 16 16 16 0E 11 36 3E 3E 36 11 11 29 29 06 39 00 00 00 00 00 00 39 02 02 02
0E 16 02 02 0E 16 16 16 0E 11 3E 3E 0B 2C 3F 3E 36 3F 3C 29 06 39 00 00 00 00 39 02 0D 02 02 16 16 16 16 02 0D 16 16 16 16 0E 29 3F 3F 25 18 3C 3F 3F 3E 18 2C 18 39
00 00 00 39 11 0D 1B 02 0D 16 0E 02 0D 16 16 16 16 0D 29 3F 3E 3E 18 1F 3F 3F 2C 18 3C 36 06 39 00 00 39 0D 0D 0D 02 0D 1B 0E 06 11 0D 0E 16 16 0D 29 3C 3E 18
0B 18 3E 3E 18 18 3C 29 02 39 00 00 39 11 0D 0D 06 11 29 36 36 36 36 11 16 16 0E 29 36 36 2C 36 3E 3E 2C 36 2C 3E 11 39 00 00 00 39 11 11 11 36 3E 3E 3F 3F 3F
3E 0D 16 16 16 0D 29 36 34 36 36 25 06 11 36 1F 11 39 00 00 00 39 11 11 29 2C 34 3C 3F 3F 3E 29 0E 16 16 16 0E 0D 11 25 29 11 0D 0E 0D 0D 02 39 00 00 00 39 06
29 3C 3E 3F 3F 36 29 1F 11 1B 16 16 16 16 16 0E 0E 0D 0D 0E 16 16 16 0E 02 39 00 00 39 11 36 36 1F 0B 06 01 02 0D 0D 02 1B 16 16 16 16 0E 0D 11 0D 0D 0D 0E 16
1B 02 39 00 00 00 39 06 18 18 06 06 06 01 0D 0E 0E 02 02 16 16 16 0E 0D 3E 3E 36 36 29 29 29 11 39 00 00 00 39 01 06 06 01 00 03 03 02 0E 0E 0D 02 0D 16 16
0E 02 34 36 2C 36 3E 3E 3E 3E 36 01 39 00 00 00 39 01 01 39 39 39 09 01 02 0E 0E 02 02 0E 16 0D 11 2C 29 2C 36 3E 3E 3E 2C 18 00 39 00 00 00 00 39 39 00 00 00
39 01 01 02 0D 0D 01 02 0E 0D 11 29 29 29 36 36 29 29 29 11 06 39 00 00 00 00 00 00 00 00 00 00 00 39 39 11 02 0D 11 02 02 0D 1B 0D 11 02 06 09 09 09 09 09 39 00
00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 06 01 01 01 02 02 02 02 39 39 39 39 39 39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 39 39 39 39 39 39
00 00 00 00 00 00 00 00 00 00 00
```

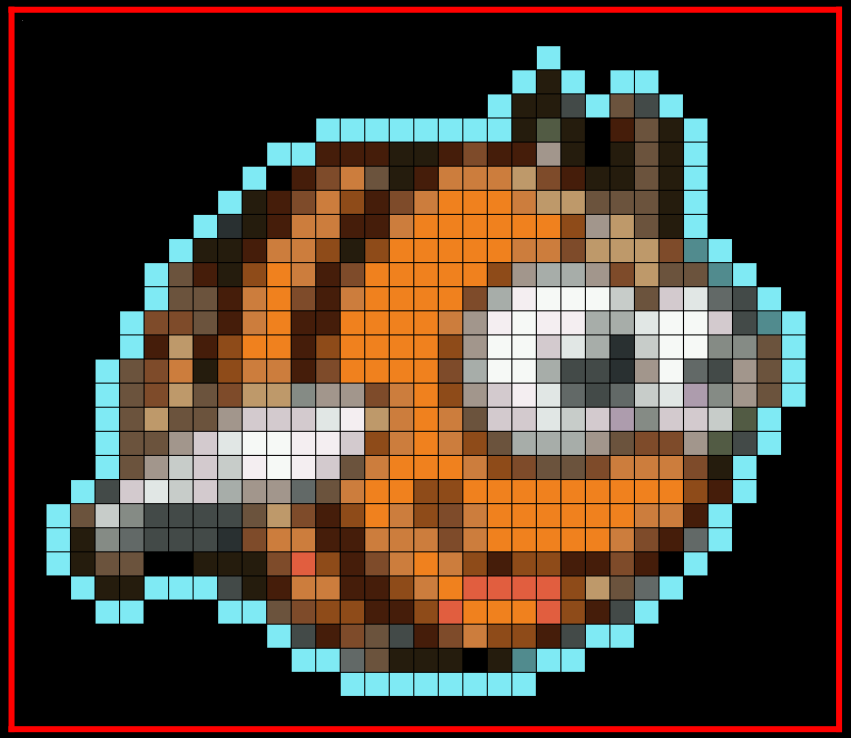
This is the data from the file "Alien2.mem" - Sprite Size 31 x 21 pixels

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 00 00 00 00 00 00 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 04 1F 39 39
39 39 39 39 2A 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 1D 15 04 2A 2A 09 09 09 2A 0F 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 39 0F 2A 2A 2A 04 1F 15 2A 15 15 1D 39 00 00 00 00 00 00 00 00 00 00 39 39 39 39 39 00 00 00 39 0A 2A 3C 2A 37 26 15 2A 2A 0F 15 39 00 00 00 00 00 00 00 00 00
39 39 09 2A 2A 09 0B 39 00 39 0F 37 37 37 37 37 37 1E 0F 0F 0F 0F 39 00 00 00 00 00 00 00 39 15 0B 1F 1F 34 34 1F 1D 39 0A 20 2B 2B 2B 2B 2B 2B 1E 0F 04 0F 37
0F 39 00 00 00 00 00 39 18 3C 3C 3F 3C 1F 36 3C 1F 18 1E 2B 2B 2B 2B 1E 1E 26 2A 15 00 0F 2A 1F 15 39 00 00 00 39 15 3C 3F 3F 3E 3F 3C 2C 34 2C 04 1E 2B 2B 2B
1E 2A 3F 3F 3F 3C 09 34 3F 3F 34 15 39 00 39 15 3C 3F 3E 3E 3E 3F 3F 3C 34 18 04 2B 2B 2B 20 26 3F 3F 3F 3F 3E 36 3E 3F 3F 3F 34 1D 39 39 15 3F 36 34 3F 3C 36
36 36 3E 25 03 2B 2B 2B 20 2A 3E 3E 3E 3C 2C 3E 34 3F 3F 3F 3F 15 39 39 15 34 3C 3F 3C 3E 3F 3C 3C 34 1F 03 2B 2B 2B 20 2A 3F 3E 3F 0B 09 25 00 18 3E 3F 2C 1D
39 39 1F 1F 3C 3C 3C 3E 3C 34 2A 18 2A 0A 1E 2B 2B 2B 12 34 3C 3F 34 0B 36 18 1F 3F 3F 15 39 00 00 39 2A 1F 34 34 36 36 18 04 04 2A 00 1E 2B 2B 2B 1E 0F 34 3F
3F 3F 36 3E 3F 3F 1F 39 00 00 00 00 39 15 0B 0B 0B 09 04 0F 04 0F 04 0F 12 20 2B 20 1E 26 2A 2A 2C 0F 1F 2C 1F 39 00 00 00 00 00 00 39 39 39 15 15 34 15 34 2C
18 34 15 12 1E 1E 12 12 1E 1E 1E 1E 1E 12 39 00 00 00 00 00 00 00 00 39 00 00 15 04 15 1F 18 15 04 15 39 0F 25 3E 2A 1E 1E 1E 12 0F 00 39 00 00 00 00 00 00 00
39 39 0F 15 0B 00 15 0F 15 0B 04 39 39 0B 2C 3F 2C 0A 0A 0A 2A 2A 04 39 00 00 00 00 00 00 00 00 39 09 39 39 39 39 09 39 39 39 00 00 39 0B 2C 1F 09 15 2A 15 15
39 00 00 00 00 00 00 00 00 00 00 00 39 00 00 00 00 39 00 00 00 00 00 00 39 0F 39 39 39 39 39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 39 00 00 00 00 00 00 00 00 00 00 00 00
```



This is the data from the file "Alien3.mem" - Sprite Size 31 x 27 pixels

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 03 39
00 39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 03 03 0B 39 14 0B 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39
39 39 39 39 39 39 03 15 03 00 05 14 03 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 05 05 05 03 03 05 10 05 05 25 03 00 03 14 03 39 00 00 00 00 00 00 00 00
00 00 00 00 39 00 05 10 21 14 03 05 21 21 21 27 10 05 03 03 14 03 39 00 00 00 00 00 00 00 00 00 00 00 00 00 39 03 05 10 21 13 05 10 21 24 24 24 21 27 27 14 14 14 03
39 00 00 00 00 00 00 00 00 00 00 39 09 03 05 21 21 05 05 21 24 24 24 24 24 24 24 13 25 27 14 03 39 00 00 00 00 00 00 00 00 00 39 03 03 05 21 21 13 03 13 24 24 24
24 24 21 21 10 27 27 27 10 1D 39 00 00 00 00 00 00 00 00 39 14 05 03 13 24 21 05 10 24 24 24 24 24 24 13 25 2C 2C 25 10 27 14 14 1D 39 00 00 00 00 00 00 39 14 14 05
21 24 10 05 21 24 24 24 24 10 2C 3E 3F 3F 3F 34 14 36 3C 18 0B 39 00 00 00 00 39 10 10 14 05 21 24 05 05 24 24 24 24 21 25 3E 3F 3E 3E 2C 2C 3C 3F 3F 36 0B 1D
39 00 00 00 39 05 27 05 13 24 24 05 13 24 24 24 24 13 25 3F 3F 36 3C 2C 09 34 3F 3F 1F 1F 14 39 00 00 39 14 10 21 03 13 21 21 05 10 24 24 24 24 10 2C 3F 3F 2C
0B 0B 09 25 3F 18 0B 25 14 39 00 00 39 14 10 27 14 10 27 27 1F 25 25 10 21 24 13 25 36 3E 3C 18 0B 18 34 3C 29 1F 25 14 39 00 00 39 14 27 14 14 25 36 36 36 3C
3E 27 21 24 21 14 36 36 3C 34 36 29 1F 36 36 34 15 39 00 00 00 39 14 14 25 36 3C 3F 3F 3E 3E 36 13 21 24 21 13 14 2C 2C 2C 25 14 10 10 25 15 0B 39 00 00 00 39
14 25 34 36 34 3E 3F 3E 36 14 21 24 24 24 21 13 10 14 14 10 21 21 21 10 03 39 00 00 00 39 0B 36 3C 34 36 2C 25 25 18 14 21 24 24 13 13 24 24 24 24 24 24 24 24
24 13 05 39 00 00 39 14 34 1F 0B 0B 0B 0B 14 27 10 05 13 24 21 13 10 21 24 24 24 24 24 21 21 05 39 00 00 00 39 03 1F 18 0B 0B 0B 09 10 21 21 05 05 21 21 21
10 21 24 24 24 24 24 21 10 05 18 39 00 00 00 39 03 14 14 00 00 03 03 03 10 1A 13 05 10 21 24 21 13 05 13 13 05 05 10 05 00 39 00 00 00 00 00 00 39 03 03 39 39 39
0B 03 05 21 21 05 05 13 21 24 1A 1A 1A 1A 13 27 14 18 39 00 00 00 00 00 00 00 39 39 00 00 00 39 39 14 10 13 13 05 05 13 1A 24 24 24 1A 13 05 0B 39 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 39 0B 05 10 14 0B 05 10 21 13 13 05 0B 39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 18 14 03 03 03 00 03 1D
39 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 39 39 39 39 39 00 00 00 00 00 00 00 00 00 00 00
```



This is the data from the file “Pal24bit.mem”

```
00 00 00 21 0F 20 3C 0B 40 25 1C 0D 16 26 12 45 1D 0A 39 25 36 32 2C 01 CA 00 06 28 30 31 26 3D 01 43 4A 48 51 4F 00 8C 34 93 B1 26 B7 34 5F 36 7E 4B 2A 70 4A
71 31 69 04 8E 4B 19 6B 53 3D 52 5B 44 F1 21 F0 72 6B 07 62 69 67 F2 52 87 E1 5E 3F CD 58 D1 7D 89 04 51 8B 8E 3C 9B 29 85 8B 85 4E A9 01 CC 7D 3D B9 8D 07 9A
9C 00 F0 80 1E A2 96 8C 84 A7 67 BE 99 6A C1 9D 43 AD 9C AD 83 AD 8A 1A E2 2A A7 AD A9 B2 B9 00 E4 A9 0D 6C D1 00 FA B8 10 FD BE 32 AF DC 00 FD C3 4B C7 CC C9
F9 CF 00 D3 CA CE 7B EF 8D DF DA 2B 7F EA F4 E0 E4 00 E6 E6 85 E1 E7 E5 FC FC 00 F4 EE F1 F6 F9 F6
```

This is the code from the file “Basys3.xdc”

```
##-----------------------------------
## Constraints Module
## Digilent Basys 3
## BeeInvaders : Onboard clock 100MHz
## VGA Resolution: 640x480 @ 60Hz
## Pixel Clock 25.2MHz
##-----------------------------------
## Clock
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports {clk_100m}];
create_clock -add -name sys_clk_pin -period 10.00 \
 -waveform {0 5} [get_ports {clk_100m}];
## Use BTNC as Reset Button (active high)
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {btn_rst_n}];
set_property PACKAGE_PIN W19 [get_ports btnL]
   set_property IOSTANDARD LVCMOS33 [get_ports btnL]
set_property PACKAGE_PIN T17 [get_ports btnR]
   set_property IOSTANDARD LVCMOS33 [get_ports btnR]
## VGA Connector
set_property -dict {PACKAGE_PIN G19 IOSTANDARD LVCMOS33} [get_ports {vga_r[0]}];
set_property -dict {PACKAGE_PIN H19 IOSTANDARD LVCMOS33} [get_ports {vga_r[1]}];
set_property -dict {PACKAGE_PIN J19 IOSTANDARD LVCMOS33} [get_ports {vga_r[2]}];
set_property -dict {PACKAGE_PIN N19 IOSTANDARD LVCMOS33} [get_ports {vga_r[3]}];
set_property -dict {PACKAGE_PIN N18 IOSTANDARD LVCMOS33} [get_ports {vga_b[0]}];
set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {vga_b[1]}];
set_property -dict {PACKAGE_PIN K18 IOSTANDARD LVCMOS33} [get_ports {vga_b[2]}];
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {vga_b[3]}];
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {vga_g[0]}];
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {vga_g[1]}];
set_property -dict {PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports {vga_g[2]}];
set_property -dict {PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports {vga_g[3]}];
set_property -dict {PACKAGE_PIN P19 IOSTANDARD LVCMOS33} [get_ports {vga_hsync}];
set_property -dict {PACKAGE_PIN R19 IOSTANDARD LVCMOS33} [get_ports {vga_vsync}];
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

This is the code from the file "Arty.xdc"

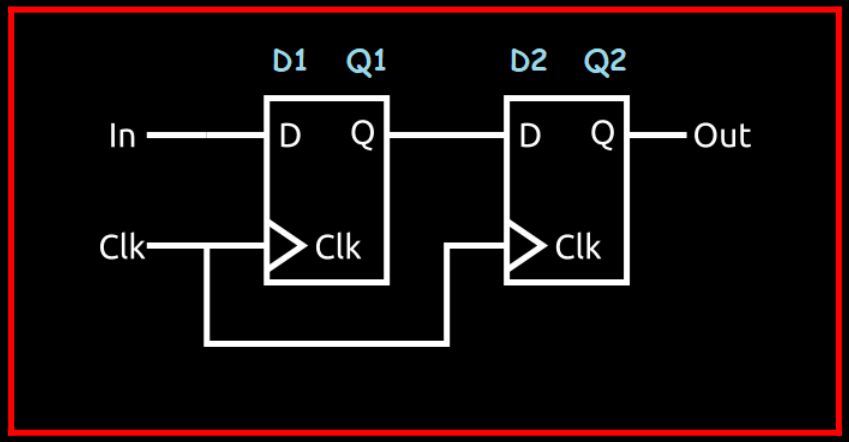
```
##-----------------------------------------------
## Constraints File
## Digilent Arty A7-35
## Bee Invaders Tutorial_1
## Onboard clock 100MHz
## VGA Resolution: 640x480 @ 60Hz
## Pixel Clock 25.2MHz
//-----------------------------------------------

## FPGA Configuration I/O Options
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## Board Clock: 100 MHz
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {clk_100m}];
create_clock -name clk_100m -period 10.00 [get_ports {clk_100m}];

## Buttons
set_property -dict {PACKAGE_PIN C2 IOSTANDARD LVCMOS33} [get_ports {btn_rst_n}];
set_property -dict {PACKAGE_PIN D9 IOSTANDARD LVCMOS33} [get_ports {btnL}];
set_property -dict {PACKAGE_PIN B8 IOSTANDARD LVCMOS33} [get_ports {btnR}];

## VGA Pmod on Header JB/JC
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {vga_hsync}];
set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {vga_vsync}];
set_property -dict {PACKAGE_PIN E15 IOSTANDARD LVCMOS33} [get_ports {vga_r[0]}];
set_property -dict {PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports {vga_r[1]}];
set_property -dict {PACKAGE_PIN D15 IOSTANDARD LVCMOS33} [get_ports {vga_r[2]}];
set_property -dict {PACKAGE_PIN C15 IOSTANDARD LVCMOS33} [get_ports {vga_r[3]}];
set_property -dict {PACKAGE_PIN U12 IOSTANDARD LVCMOS33} [get_ports {vga_g[0]}];
set_property -dict {PACKAGE_PIN V12 IOSTANDARD LVCMOS33} [get_ports {vga_g[1]}];
set_property -dict {PACKAGE_PIN V10 IOSTANDARD LVCMOS33} [get_ports {vga_g[2]}];
set_property -dict {PACKAGE_PIN V11 IOSTANDARD LVCMOS33} [get_ports {vga_g[3]}];
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {vga_b[0]}];
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {vga_b[1]}];
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {vga_b[2]}];
set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {vga_b[3]}];
```

(D) DEBOUNCING A BUTTON

When you press one of the buttons on the FPGA board there is a chance that the button will not simply go from open to close. Since a button is a mechanical device, the contacts can bounce and for a short period after the button is pressed the value you read from an IO pin may toggle between 0 and 1 a few times before settling on the actual value

To debounce a button it must look like it is being pressed for a set amount of time. If this is not done and when using the button to increment a counter, then the counter may increase by more than 1 per button press since it will appear that each bounce was a separate press

Creating a delay in an FPGA is the most common use of a shift register. The delay is often used to align data in time



The figure above shows this simple type of shift register (2 x 3 pin D Flip-Flops are used to stabilise a button press)

Our design uses 2 flip-flops and 2 counters (20 bit or [19:0] or max "11111111111111111111" binary) to stabilise a button press

In will hold the state of a Button Pressed ("0"or "1")

Clk will be used to synchronise passing values between the 2 flip-flops

Q (Q1, Q2) will equal the value in D (D1, D2 i.e. Q1 = D1, Q2 = D2) every "positive clock pulse"

D (D1, D2) is passed to Q (Q1, Q2) every "positive clock pulse"), see Q above

Out will equal "1" when counter ctr_q = "11111111111111111111" binary (see below)

We use 2 x 2 bit counters:

ctr_d which is a 20 bit [19:0] (max "11111111111111111111" binary) register counter
ctr_q which is a 20 bit [19:0] (max "11111111111111111111" binary) register counter

ctr_q = ctr_d every "positive clock pulse"
if ctr_q equals "11111111111111111111" binary then ctr_d = ctr_q and this also means Out will = "1"

ctr_d = ctr_q + "1' binary, this increments the counters

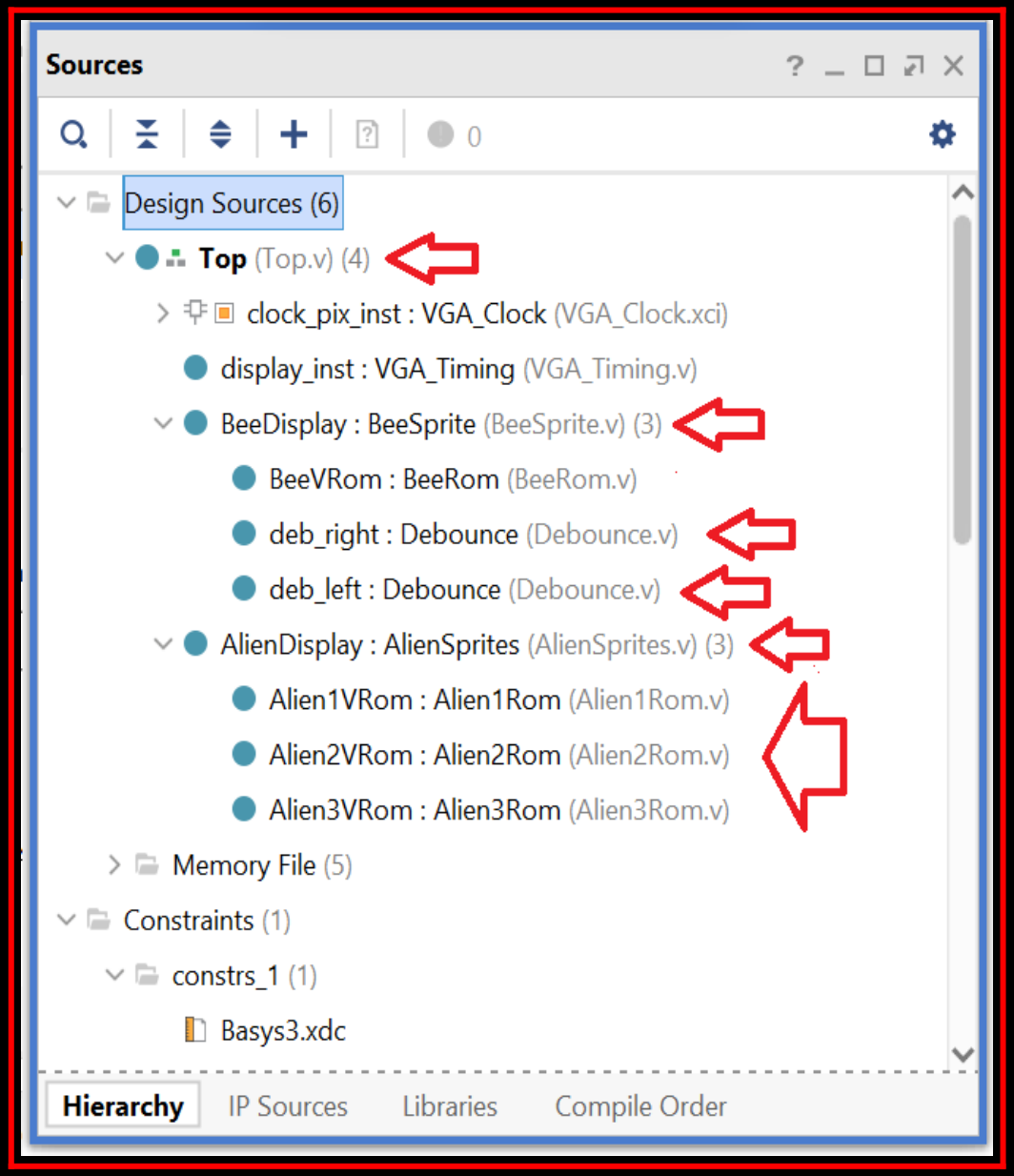
We use sync_d and sync_q registers and they are updated when their values change:

sync_d[1:0] = "0" or "1" when a button is pressed

sync_d[1:0] = sync_q[1;0], i.e. this passes the value from flip-flop 1 to flip-flop 2
sync_q[1:0] = sync_d[1:0]

sync_q[1:0] = if this equals "0" then ctr_d = "00000000000000000000" binary

(E) EXPLANATION OF THE VERILOG CODE USED



01 Top.v module additional code

```
// Setup Top module
module Top (
    ..
    ..
    input wire btnR,                    // Right button
    input wire btnL                     // Left button
    );
```

We add to the Top module setup 2 inputs, one for the Right button and one for the Left button

```
// Instantiate BeeSprite
    wire [1:0] BeeSpriteOn;             // 1=on, 0=off
    wire [7:0] dout;                    // pixel value from Bee.mem
    BeeSprite BeeDisplay (
        ..
        ..
        .btnR(btnR),
        .btnL(btnL),
        ..
```

This adds the Right and Left buttons to the instantiation of BeeSprite

```
// Instantiate AlienSprites
wire [1:0] Alien1SpriteOn;          // 1=on, 0=off
wire [1:0] Alien2SpriteOn;          // 1=on, 0=off
wire [1:0] Alien3SpriteOn;          // 1=on, 0=off
wire [7:0] Alien1dout;              // pixel value from Alien1.mem
wire [7:0] Alien2dout;              // pixel value from Alien2.mem
wire [7:0] Alien3dout;              // pixel value from Alien3.mem
AlienSprites AlienDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .Alien1SpriteOn(Alien1SpriteOn),
    .Alien2SpriteOn(Alien2SpriteOn),
    .Alien3SpriteOn(Alien3SpriteOn),
    .A1dout(Alien1dout),
    .A2dout(Alien2dout),
    .A3dout(Alien3dout)
);
```

This instantiates the AlienSprites module (3 different types of aliens)

```
// VGA Output
assign vga_hsync = hsync;
assign vga_vsync = vsync;
always @ (posedge clk_pix)
begin
   if(de)
      begin
         ..
         ..
         else
         if (Alien1SpriteOn==1)
            begin
               vga_r <= (palette[(Alien1dout*3)])>>4;       // RED bits(7:4) from colour palette
               vga_g <= (palette[(Alien1dout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
               vga_b <= (palette[(Alien1dout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
            end
         else
         if (Alien2SpriteOn==1)
            begin
               vga_r <= (palette[(Alien2dout*3)])>>4;       // RED bits(7:4) from colour palette
               vga_g <= (palette[(Alien2dout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
               vga_b <= (palette[(Alien2dout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
            end
         else
         if (Alien3SpriteOn==1)
            begin
               vga_r <= (palette[(Alien3dout*3)])>>4;       // RED bits(7:4) from colour palette
               vga_g <= (palette[(Alien3dout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
               vga_b <= (palette[(Alien3dout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
            end
         else
```

We have added Alien1, Alien2 and Alien3 to the VGA Output routine. If any of the Aliens are On (equal to "1") then the Alien data is sent to the VGA Output

02 BeeSprite.v module additional code

```
// Setup BeeSprite module
module BeeSprite(
    ..
    ..
    input wire btnR,                    // right button
    input wire btnL,                    // left button
    output wire [7:0] dataout
    );
```

We have added the Right and Left buttons to the BeeSprite module setup

```
// Instantiate Debounce
wire sig_right;
wire sig_left;

Debounce deb_right (
    .clk_pix(clk_pix),
    .btn(btnR),
    .out(sig_right)
);

Debounce deb_left (
    .clk_pix(clk_pix),
    .btn(btnL),
    .out(sig_left)
);
```

This instantiates the Debounce routine which stabilises the Left or Right button when pressed. The returned signal is passed to sig_right or sig_left

```
 always @ (posedge clk_pix)
begin
   ..
   ..
   // if left or right button pressed, move the Bee
   if (sx==639 && sy==479)                          // check for movement once every frame
      begin
         if (sig_right == 1 && BeeX<640-BeeWidth)   // Check for right button
            BeeX<=BeeX+1;
         else
         if (sig_left == 1 && BeeX>2)               // Check for left button
            BeeX<=BeeX-1;
      end
```

The line if (sx==639 && sy==479) ensures that, the following check to see if the Left or Right button has been pressed, only happens once every frame

If true:

BeeX is incremented if the Right button has been pressed and BeeX does not go off the right hand side of the screen

BeeX is decremented if the Left button has been pressed and BeeX does not go off the left hand side of the screen

03 Debounce.v module

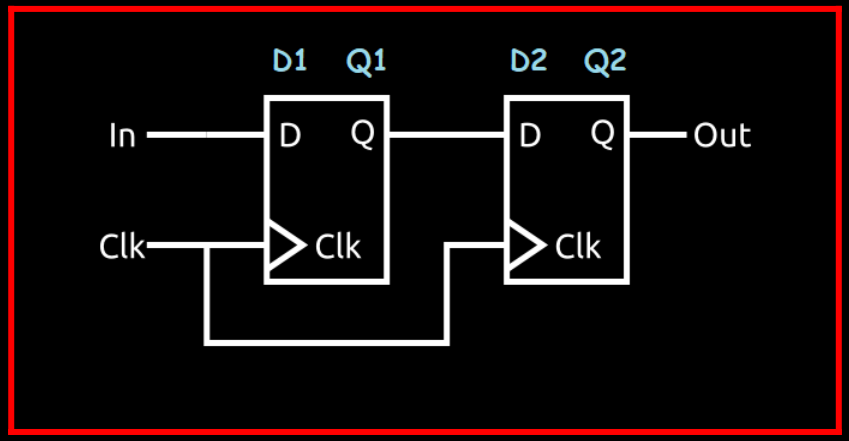
```
//----------------------------------------------
// Debounce.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps


// Setup Debounce module
module Debounce (
    input wire clk_pix,          // Clock signal to synchronize the button input
    input wire btn,
    output wire out
  );

  reg [19:0] ctr_d;              // 20 bit counter to increment when button is pressed or released
  reg [19:0] ctr_q;              // 20 bit counter to increment when button is pressed or released
  reg [1:0] sync_d;              // button flip-flop for synchronization
  reg [1:0] sync_q;              // button flip-flop for synchronization

  assign out = ctr_q == {20{1'b1}}; // if ctr_q = 11111111111111111111
```

Our design uses 2 flip-flops and 2 counters to stabilise a button press

input wire btn, will hold the state of a Button Pressed (“0“or ”1”)

reg [19:0] ctr_d; is a 20 bit (max “11111111111111111111” binary) register counter
reg [19:0] ctr_q; is a 20 bit (max “11111111111111111111” binary) register counter

reg [1:0] sync_d; represents the D side of each Flip-flop
reg [1:0] sync_q; represents the Q side of each Flip-flop



assign out = ctr_q == {20{1'b1}}; out will equal "1" when counter ctr_q = "11111111111111111111" binary

```
always @(*)
  begin
    sync_d[0] = btn;
    sync_d[1] = sync_q[0];
    ctr_d = ctr_q + 1'b1;

    if (ctr_q == {20{1'b1}})
      ctr_d = ctr_q;

    if (!sync_q[1])
      ctr_d = 20'd0;
  end

  always @(posedge clk_pix)
  begin
    ctr_q <= ctr_d;
    sync_q <= sync_d;
  end
endmodule
```

sync_d[0] = btn; sets it to "0" or "1" when a button is pressed

sync_d[1] = sync_q[0]; passes the value from flip-flop 1 (Q1) to flip-flop 2 (D2)

ctr_d = ctr_q + 1'b1; this increments the counters

if (ctr_q == {20{1'b1}})
  ctr_d = ctr_q; this transfers the value in ctr_q to ctr_d if ctr_q = "11111111111111111111" binary

if (!sync_q[1])
  ctr_d = 20'd0; this sets ctr_d equal to "00000000000000000000" if sync_q = "0"

ctr_q <= ctr_d; this transfers ctr_d to ctr_q on every positive clock pulse

sync_q <= sync_d; this transfers sync_d to sync_q on every positive clock pulse

04 AlienSprites.v module

```
//----------------------------------------------
// AlienSprites.v module
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps

// Setup AlienSprites Module
module AlienSprites(
    input wire clk_pix,                 // 25MHz pixel clock
    input wire [9:0] sx,                // current x position
    input wire [9:0] sy,                // current y position
    input wire de,                      // high during active pixel drawing
    output reg Alien1SpriteOn,          // 1=on, 0=off
    output reg Alien2SpriteOn,          // 1=on, 0=off
    output reg Alien3SpriteOn,          // 1=on, 0=off
    output wire [7:0] A1dout,           // 8 bit pixel value from Alien1.mem
    output wire [7:0] A2dout,           // 8 bit pixel value from Alien2.mem
    output wire [7:0] A3dout            // 8 bit pixel value from Alien3.mem
    );
```

This sets up the AlienSprites module for the 3 different types of Aliens

```
// instantiate Alien1Rom code
    reg [9:0] A1address;                // 2^10 or 1024, need 31 x 26 = 806
    Alien1Rom Alien1VRom (
        .A1address(A1address),
        .clk_pix(clk_pix),
        .A1dout(A1dout)
    );

// instantiate Alien2Rom code
    reg [9:0] A2address;                // 2^10 or 1024, need 31 x 21 = 651
    Alien2Rom Alien2VRom (
        .A2address(A2address),
        .clk_pix(clk_pix),
        .A2dout(A2dout)
    );

// instantiate Alien3Rom code
    reg [9:0] A3address;                // 2^10 or 1024, need 31 x 27 = 837
    Alien3Rom Alien3VRom (
        .A3address(A3address),
        .clk_pix(clk_pix),
        .A3dout(A3dout)
    );
```

This instantiates Alien1Rom, Alien2Rom and Alien3Rom, which retrieve the data from the 3 mem files

```
// setup character positions and sizes
reg [9:0] A1X = 135;              // Alien1 X start position
reg [8:0] A1Y = 85;               // Alien1 Y start position
localparam A1Width = 31;          // Alien1 width in pixels
localparam A1Height = 26;         // Alien1 height in pixels
reg [9:0] A2X = 135;              // Alien2 X start position
reg [8:0] A2Y = 120;              // Alien2 Y start position
localparam A2Width = 31;          // Alien2 width in pixels
localparam A2Height = 21;         // Alien2 height in pixels
reg [9:0] A3X = 135;              // Alien3 X start position
reg [8:0] A3Y = 180;              // Alien3 Y start position
localparam A3Width = 31;          // Alien3 width in pixels
localparam A3Height = 27;         // Alien3 height in pixels

reg [9:0] AoX = 0;                // Offset for X Position of next Alien in row
reg [8:0] AoY = 0;                // Offset for Y Position of next row of Aliens
reg [9:0] AcounterW = 0;          // Counter to check if Alien width reached
reg [9:0] AcounterH = 0;          // Counter to check if Alien height reached
reg [3:0] AcolCount = 11;         // Number of horizontal aliens in all columns
```

This defines Alien1, Alien2 and Alien3 x, y positions and the width, height of the sprites

AoX and AoY are used as offsets for horizontal (11 sprites) and vertical (2 rows) Alien sprites

AcounterW is a counter, reset to "0" when it equals the last pixel of the Alien (horizontal line)

AcounterH is a counter, reset to "0" when it equals the last pixel of the Alien (horizontal and vertical)

AcolCount contains the number of Aliens (11) per row

```
always @ (posedge clk_pix)
   begin
      if (de)
         begin
            // check if sx,sy are within the confines of the Alien characters
            // Alien1
            if (sx==A1X+AoX-2 && sy==A1Y+AoY)
               begin
                  A1address <= 0;
                  Alien1SpriteOn <=1;
                  AcounterW<=0;
               end
            if ((sx>A1X+AoX-2) && (sx<A1X+A1Width+AoX) && (sy>A1Y+AoY-1) && (sy<A1Y+A1Height+AoY))
               begin
                  A1address <= A1address + 1;
                  AcounterW <= AcounterW + 1;
                  Alien1SpriteOn <=1;
                  if (AcounterW==A1Width-1)
                     begin
                        AcounterW <= 0;
                        AoX <= AoX + 40;
                        if(AoX<(AcolCount-1)*40)
                              A1address <= A1address - (A1Width-1);
                           else
                           if(AoX==(AcolCount-1)*40)
                              AoX<=0;
                        end
               end
            else
               Alien1SpriteOn <=0;
```

This checks if sx and sy are within Alien1 x and y locations and dimensions

AoX is an offset and is incremented by "40" each time the last pixel in Alien1 horizontal line has been reached. This is how multiple Aliens are displayed. AoX cannot be incremented past 11 aliens per row

AoY is also an offset but is not used with Alien1 as there is only 1 row of these aliens

AcounterW is used as a counter and when the last pixel in Alien1 (horizontal line) has been reached, "40" is added to AoX and AcounterW is rest to "0"

```
// Alien2
if (sx==A2X+AoX-2 && sy==A2Y+AoY)
   begin
      A2address <= 0;
      Alien2SpriteOn <=1;
      AcounterW<=0;
   end
if ((sx>A2X+AoX-2) && (sx<A2X+A2Width+AoX) && (sy>A2Y+AoY-1) && (sy<A2Y+AoY+A2Height))
   begin
      A2address <= A2address + 1;
      AcounterW <= AcounterW + 1;
      Alien2SpriteOn <=1;
      if (AcounterW==A2Width-1)
         begin
            AcounterW <= 0;
            AoX <= AoX + 40;
            if(AoX<(AcolCount-1)*40)
                  A2address <= A2address - (A2Width-1);
               else
               if(AoX==(AcolCount-1)*40)
               begin
                     AoX<=0;
                     AcounterH <= AcounterH + 1;
                     if(AcounterH==A2Height-1)
                  begin
                        AcounterH<=0;
                        AoY <= AoY + 30;
                        if(AoY==30)
                           begin
                              AoY<=0;
                              AoX<=0;
                            end
                     end
               end
         end
   end
else
   Alien2SpriteOn <=0;
```

This is the same as Alien1 but with the following:

AcounterH s used as a counter and when the last pixel in Alien2 (horizontal and vertical) has been reached, “30” is added to AoY and AcounterH is reset to “0”. This allows a second row of Alien2 sprites to be drawn

Alien3 is the same as Alien2

05 Alien1Rom.v module

```
//----------------------------------------------
// Alien1Rom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_3
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps


// Setup Alien1Rom module
module Alien1Rom(
    input wire [9:0] A1address, // (9:0) or 2^10 or 1024, need 31 x 26 = 806
    input wire clk_pix,
    output reg [7:0] A1dout      // (7:0) 8 bit pixel value from Alien1.mem
    );


    (*ROM_STYLE="block"*) reg [7:0] A1memory_array [0:805]; // 8 bit values for 806 pixels of Alien1 (31 x 26)


    initial
    begin
        $readmemh("Alien1.mem", A1memory_array);
    end


    always @ (posedge clk_pix)
            A1dout <= A1memory_array[A1address];
endmodule
```

This module creates a Single Port ROM which reads the data from the "Alien1.mem" file

"Alien2Rom.v" and "Alien3Rom.v" are the same and read the data from the "Alien2.mem" and "Alien3.mem" files

06 Basys.xdc module

```
##-----------------------------------
## Constraints Module
## Digilent Basys 3
## BeeInvaders : Onboard clock 100MHz
## VGA Resolution: 640x480 @ 60Hz
## Pixel Clock 25.2MHz
##-----------------------------------
..
..
set_property PACKAGE_PIN W19 [get_ports btnL]
    set_property IOSTANDARD LVCMOS33 [get_ports btnL]
set_property PACKAGE_PIN T17 [get_ports btnR]
    set_property IOSTANDARD LVCMOS33 [get_ports btnR]
```

This adds the Left and Right buttons to the constraints file