BEE INVADERS

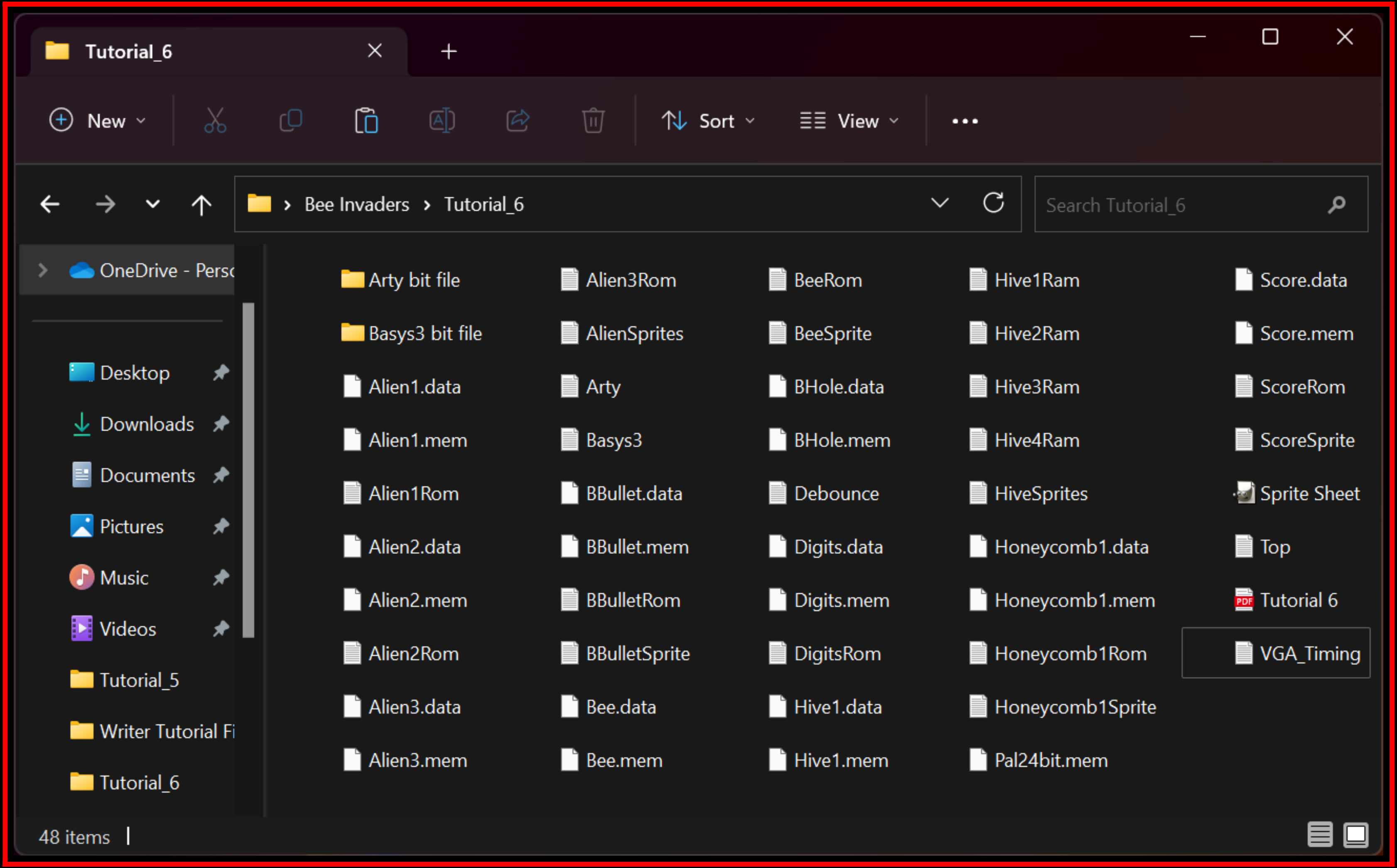
This Step By Step Tutorial Is For The
Digilent Basys3 FPGA Board Or The Digilent Arty A7-35 FPGA Board With A VGA Pmod Connected
But Can Be Adapted To Other FPGA Boards
A Modern Version Of The Popular Arcade Game
Space Invaders

Tutorial 6 – Shooting The Bee Invaders, Speeding Up The Invaders, Creating A Score And Level Indicator

CONTENTS

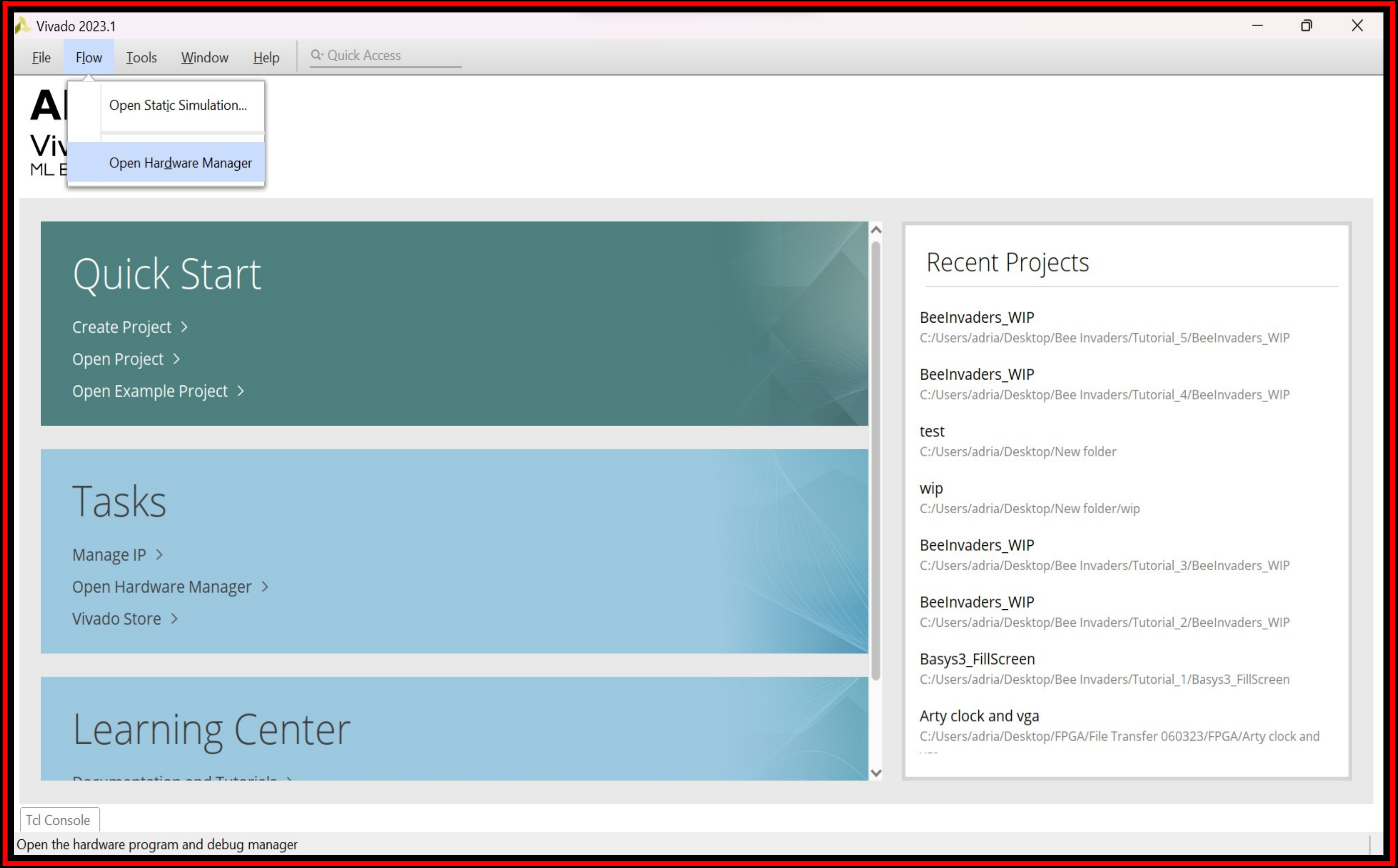
(A) EXTRACTING THE FILES AND RUNNING THE COMPILED BIT FILE

01 In the folder "Bee Invaders" create a folder called "Tutorial_6" and extract the files from the downloaded file "Tutorial_6 Files.zip" to this folder

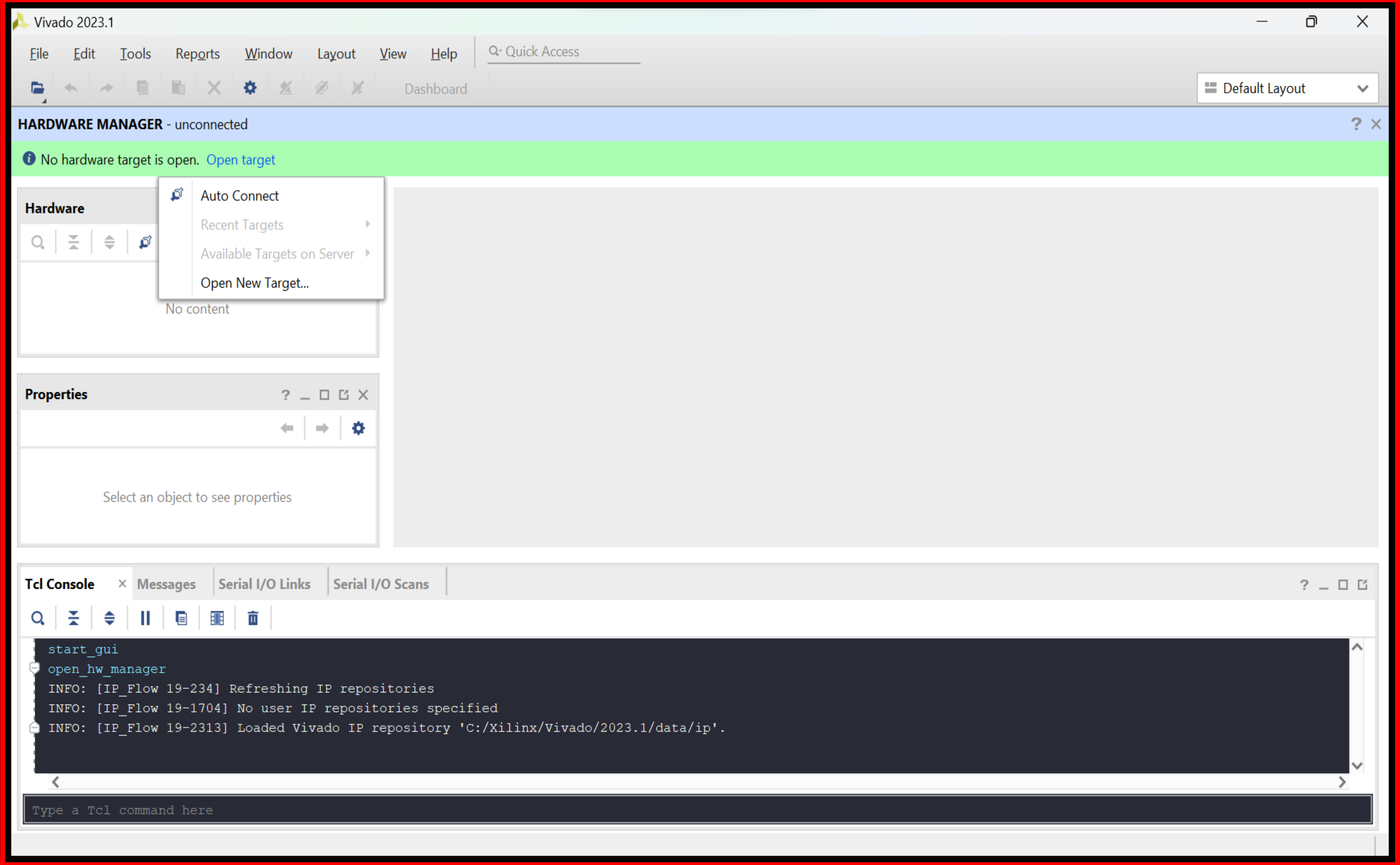


Arty bit file	Alien3Rom	BeeRom	Hive1Ram	Score.data
Basys3 bit file	AlienSprites	BeeSprite	Hive2Ram	Score.mem
Alien1.data	Arty	BHole.data	Hive3Ram	ScoreRom
Alien1.mem	Basys3	BHole.mem	Hive4Ram	ScoreSprite
Alien1Rom	BBullet.data	Debounce	HiveSprites	Sprite Sheet
Alien2.data	BBullet.mem	Digits.data	Honeycomb1.data	Top
Alien2.mem	BBulletRom	Digits.mem	Honeycomb1.mem	Tutorial 6
Alien2Rom	BBulletSprite	DigitsRom	Honeycomb1Rom	VGA_Timing
Alien3.data	Bee.data	Hive1.data	Honeycomb1Sprite	
Alien3.mem	Bee.mem	Hive1.mem	Pal24bit.mem	

02 If you would like to run the compiled tutorial, you will find a bit file in the folders "Basys3 bit file" and "Arty bit file" for the Basys3 FPGA board and the Arty A7-35 FPGA board. To do this, run Vivado and select "Flow" and "Open Hardware Manager"


Vivado 2023.1
File
Flow
Tools
Window
Help
Quick Access
Open Static Simulation...
Open Hardware Manager
Quick Start
Create Project >
Open Project >
Open Example Project >
Tasks
Manage IP >
Open Hardware Manager >
Vivado Store >
Learning Center
Recent Projects
BeeInvaders_WIP
C:/Users/adria/Desktop/Bee Invaders/Tutorial_5/BeeInvaders_WIP
BeeInvaders_WIP
C:/Users/adria/Desktop/Bee Invaders/Tutorial_4/BeeInvaders_WIP
test
C:/Users/adria/Desktop/New folder
wip
C:/Users/adria/Desktop/New folder/wip
BeeInvaders_WIP
C:/Users/adria/Desktop/Bee Invaders/Tutorial_3/BeeInvaders_WIP
BeeInvaders_WIP
C:/Users/adria/Desktop/Bee Invaders/Tutorial_2/BeeInvaders_WIP
Basys3_FillScreen
C:/Users/adria/Desktop/Bee Invaders/Tutorial_1/Basys3_FillScreen
Arty clock and vga
C:/Users/adria/Desktop/FPGA/File Transfer 060323/FPGA/Arty clock and
Tcl Console
Open the hardware program and debug manager

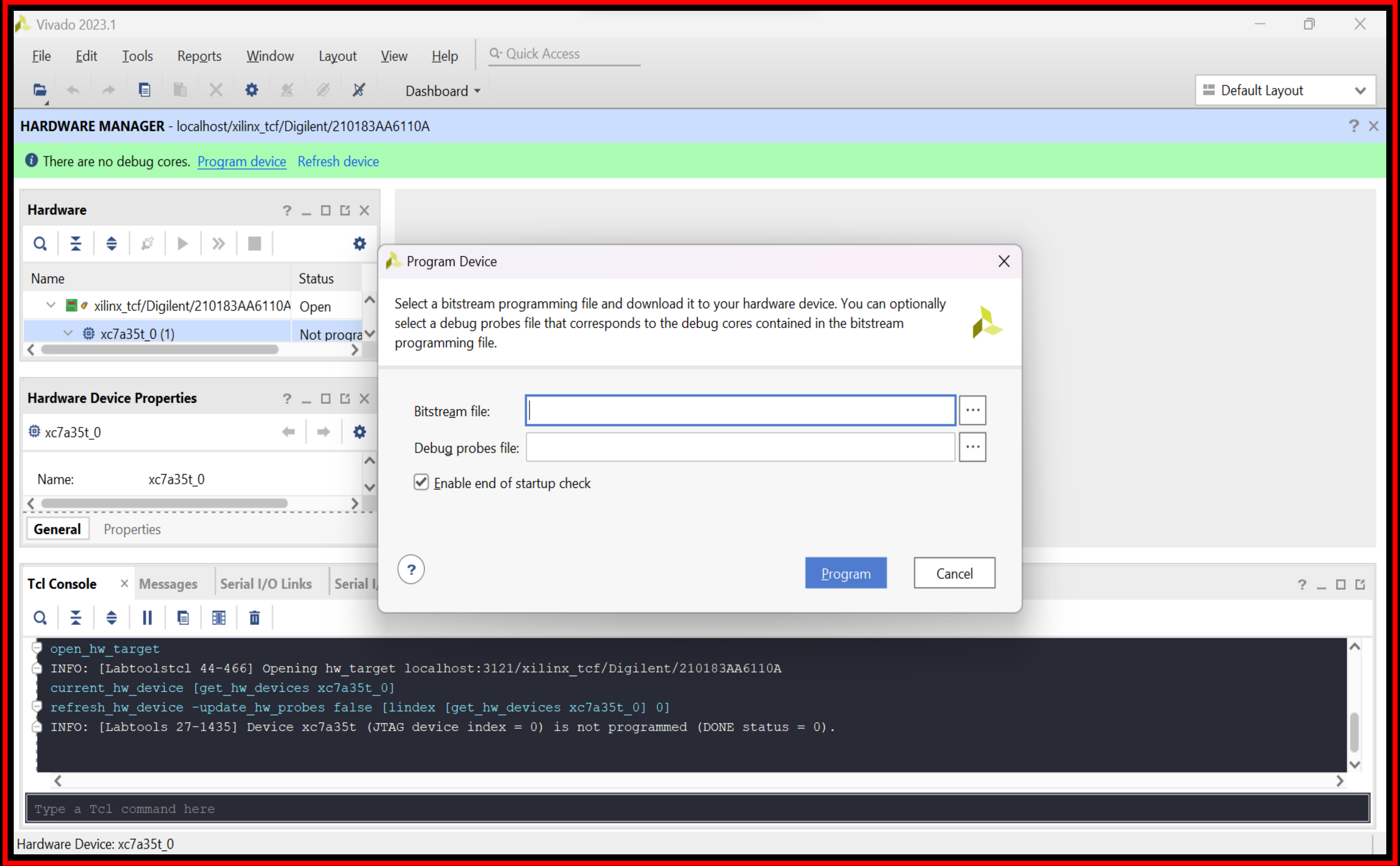
03 Connect the FPGA board to your computer / VGA screen and switch the board on (Basys3). Select "Open Target" and "Auto Connect"



No hardware target is open. Open target

```
start_gui
open_hw_manager
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2023.1/data/ip'.
```

04 When Vivado has connected to the FPGA board select "Program device" and click on the 3 dots next to the "Bitstream file:" box


Vivado 2023.1
File Edit Tools Reports Window Layout View Help
Quick Access
Dashboard
Default Layout
Hardware
Hardware Device Properties
General Properties
Tcl Console Messages Serial I/O Links
Program Device
Program Cancel

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183AA6110A

There are no debug cores. Program device Refresh device

Name	Status
xilinx_tcf/Digilent/210183AA6110A	Open
xc7a35t_0 (1)	Not progra

Name: xc7a35t_0

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file:

Debug probes file:

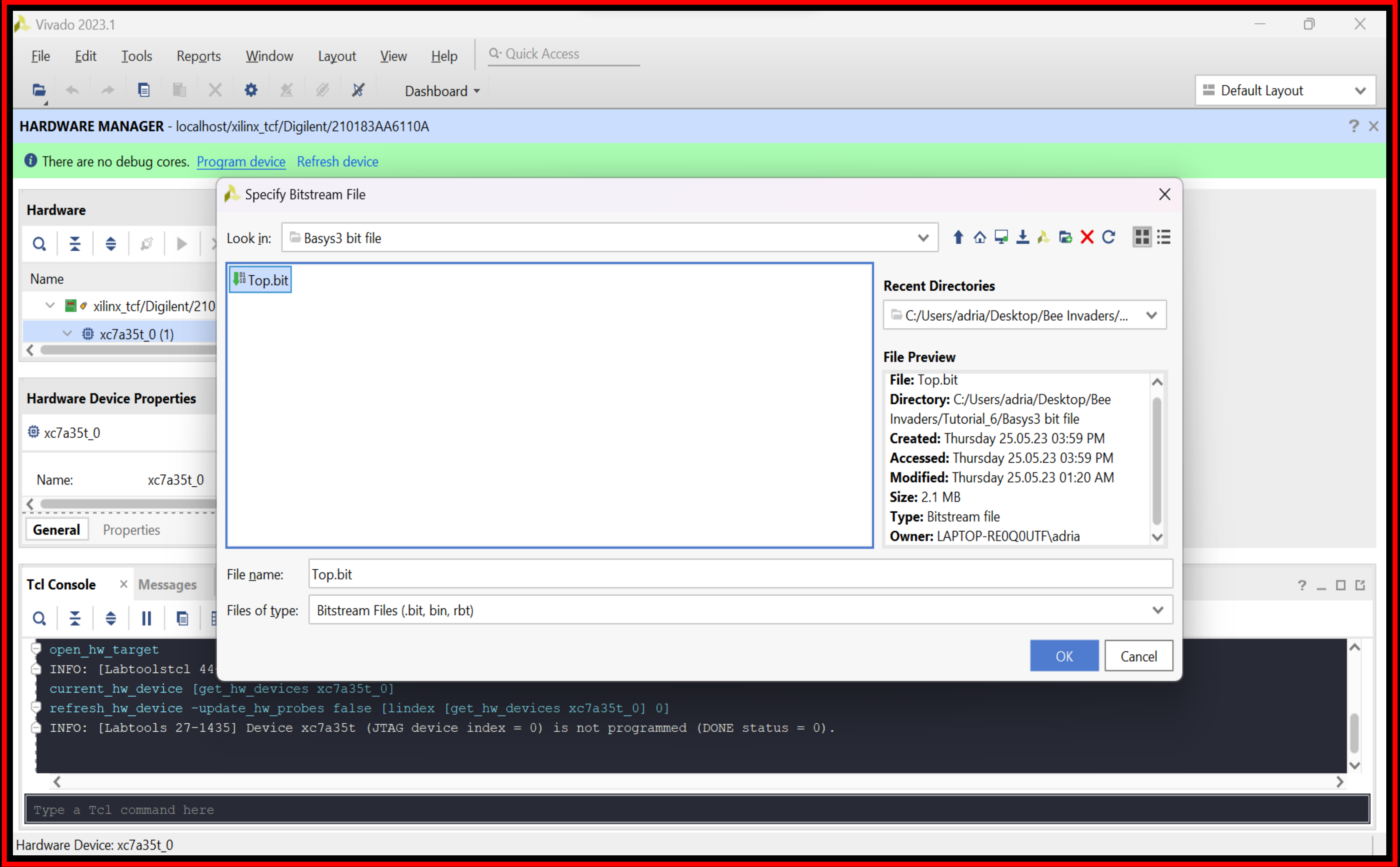
☑ Enable end of startup check

```
open_hw_target
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210183AA6110A
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1435] Device xc7a35t (JTAG device index = 0) is not programmed (DONE status = 0).
```

Type a Tcl command here

Hardware Device: xc7a35t_0

05 Navigate to the "Bee Invaders/Tutorial_6/Basys3 bit file" or the "Arty bit file" folder if you are using the Arty A7-35 board. Select the "Top.bit" file and select "OK"



HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183AA6110A

There are no debug cores. Program device Refresh device

xilinx_tcf/Digilent/210
xc7a35t_0 (1)

xc7a35t_0
Name: xc7a35t_0

Look in: Basys3 bit file

Top.bit

C:/Users/adria/Desktop/Bee Invaders/...

File: Top.bit
Directory: C:/Users/adria/Desktop/Bee Invaders/Tutorial_6/Basys3 bit file
Created: Thursday 25.05.23 03:59 PM
Accessed: Thursday 25.05.23 03:59 PM
Modified: Thursday 25.05.23 01:20 AM
Size: 2.1 MB
Type: Bitstream file
Owner: LAPTOP-REOQ0UTF\adria

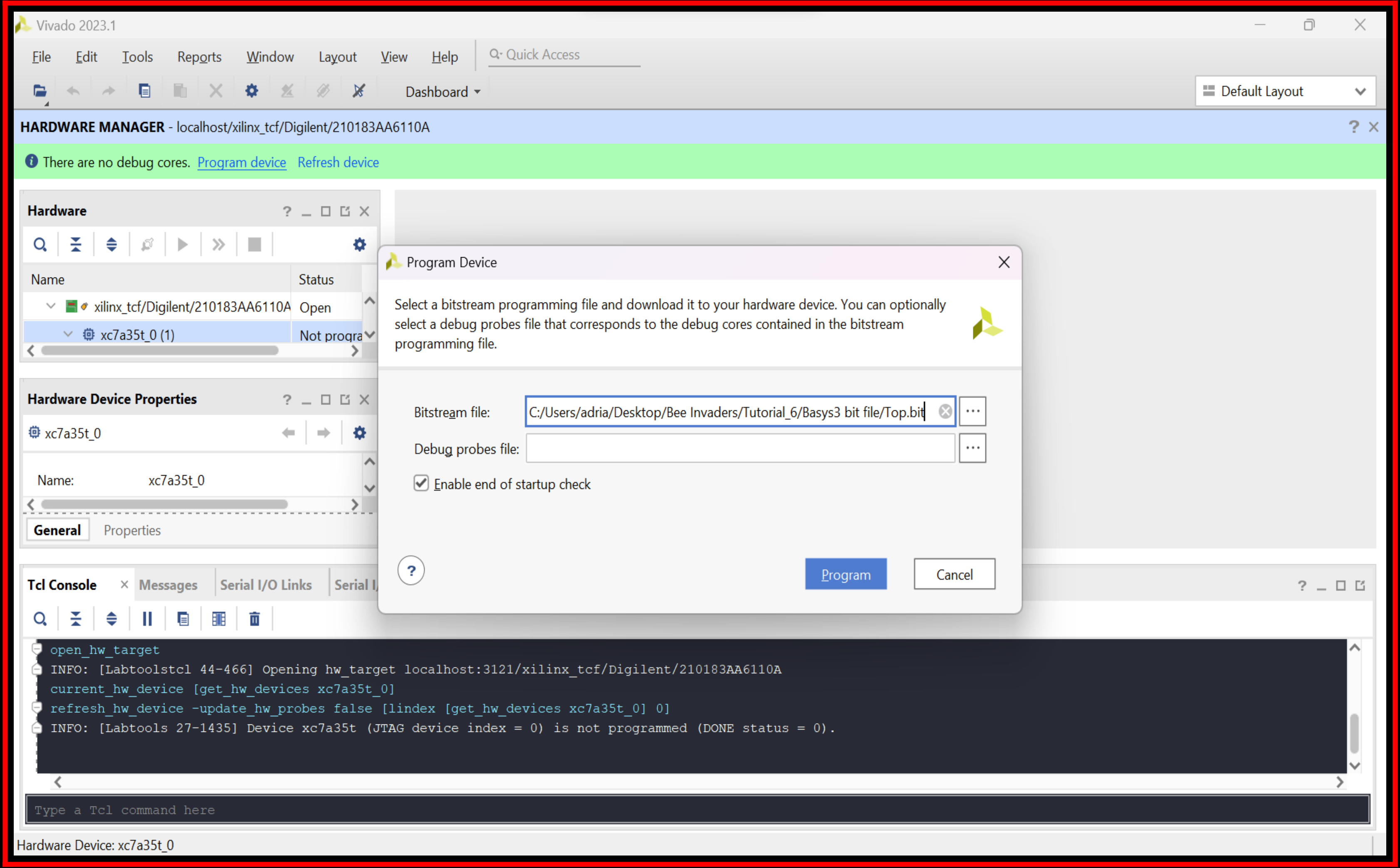
File name: Top.bit
Files of type: Bitstream Files (.bit, bin, rbt)

```
open_hw_target
INFO: [Labtoolstcl 44
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1435] Device xc7a35t (JTAG device index = 0) is not programmed (DONE status = 0).
```

Type a Tcl command here

Hardware Device: xc7a35t_0

06 Then click on the "Program" button and you should see the game running on the FPGA board / VGA screen


Vivado 2023.1
File Edit Tools Reports Window Layout View Help
Quick Access
Dashboard
Default Layout
HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183AA6110A
There are no debug cores. Program device Refresh device
Hardware
Name Status
xilinx_tcf/Digilent/210183AA6110A Open
xc7a35t_0 (1) Not progra
Hardware Device Properties
xc7a35t_0
Name: xc7a35t_0
General Properties
Tcl Console Messages Serial I/O Links Serial I/
Program Device
Program
Cancel
Type a Tcl command here
Hardware Device: xc7a35t_0

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

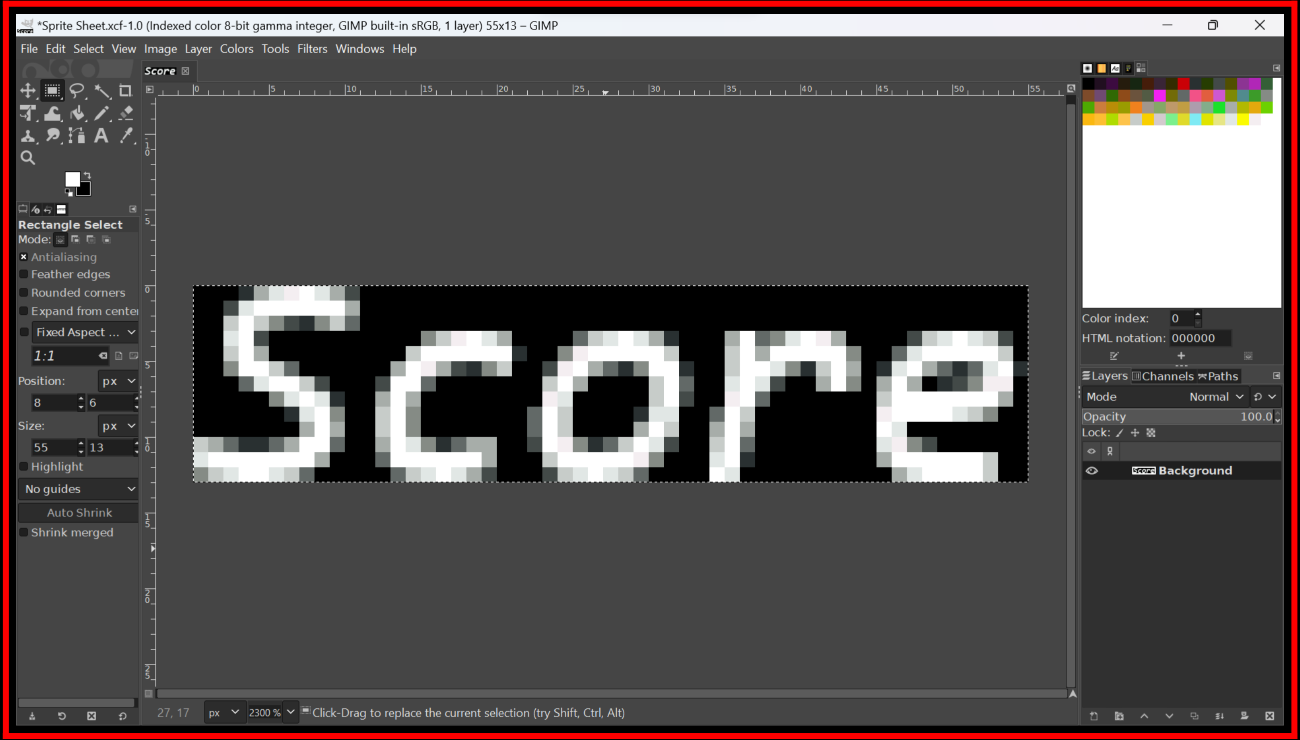
Bitstream file: C:/Users/adria/Desktop/Bee Invaders/Tutorial_6/Basys3 bit file/Top.bit
Debug probes file:
[✓] Enable end of startup check

```
open_hw_target
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210183AA6110A
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1435] Device xc7a35t (JTAG device index = 0) is not programmed (DONE status = 0).
```

(B) USING GIMP TO GENERATE THE GRAPHICS FOR THE SCORE

01 The files for the Score, Digits and Honeycomb Level Indicator are in the files which were extracted in section (A). Jump to section (C) if you do not wish to see how the files were made in Gimp

Open “Sprite Sheet.xcf” in the “Tutorial_6” 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 Score character and using the “rectangle select tool” select around the Score (inside the yellow rectangle), this should be a rectangle 55 x 13 pixels and crop it



*Sprite Sheet.xcf-1.0 (Indexed color 8-bit gamma integer, GIMP built-in sRGB, 1 layer) 55x13 – GIMP

Rectangle Select
Position: 8, 6 px
Size: 55, 13 px
Fixed Aspect ... 1:1
Color index: 0
HTML notation: 000000
Mode Normal
Opacity 100.0
Background
27, 17 px 2300 % Click-Drag to replace the current selection (try Shift, Ctrl, Alt)

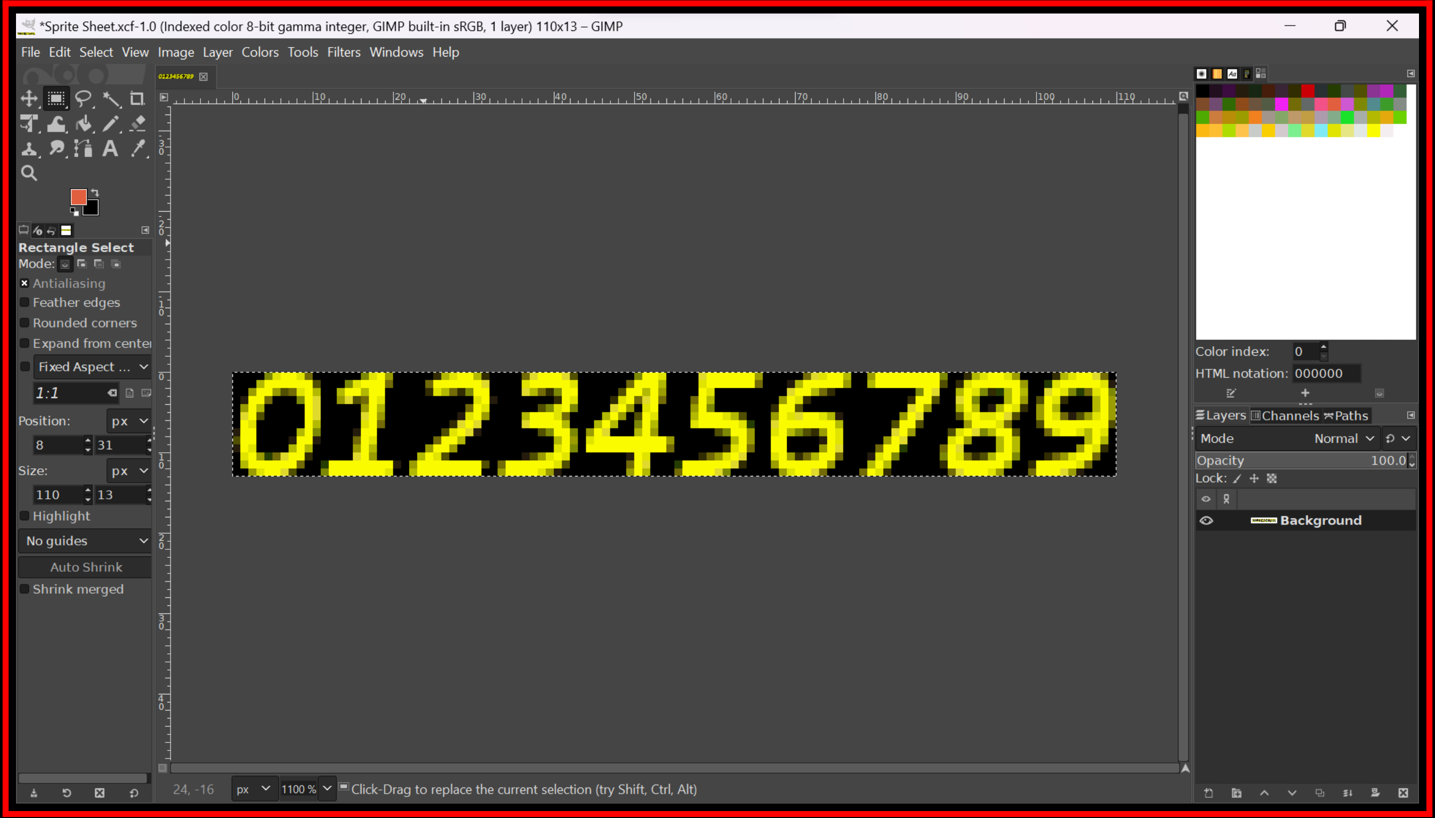
02

The image needs to be saved as a Raw Data File, do this using File → Export As → Raw image data. Call the file “Score.data”

Using HxD Hex Editor (or similar) load the file “Score.data”, select all the data and copy it

Then paste the data into a Notepad file and save it as “Score.mem” in the folder “Tutorial_6”

03 Zoom in on the Digits character (inside the white border) and using the "rectangle select tool" select around the Digits (this should be a rectangle 110 x 13 pixels) and crop it


*Sprite Sheet.xcf-1.0 (Indexed color 8-bit gamma integer, GIMP built-in sRGB, 1 layer) 110x13 – GIMP

Rectangle Select
Antialiasing
Feather edges
Rounded corners
Expand from center
Fixed Aspect ratio: 1:1
Position: 8, 31 px
Size: 110, 13 px
Highlight
No guides
Shrink merged

Color index: 0
HTML notation: 000000

Mode: Normal
Opacity: 100.0

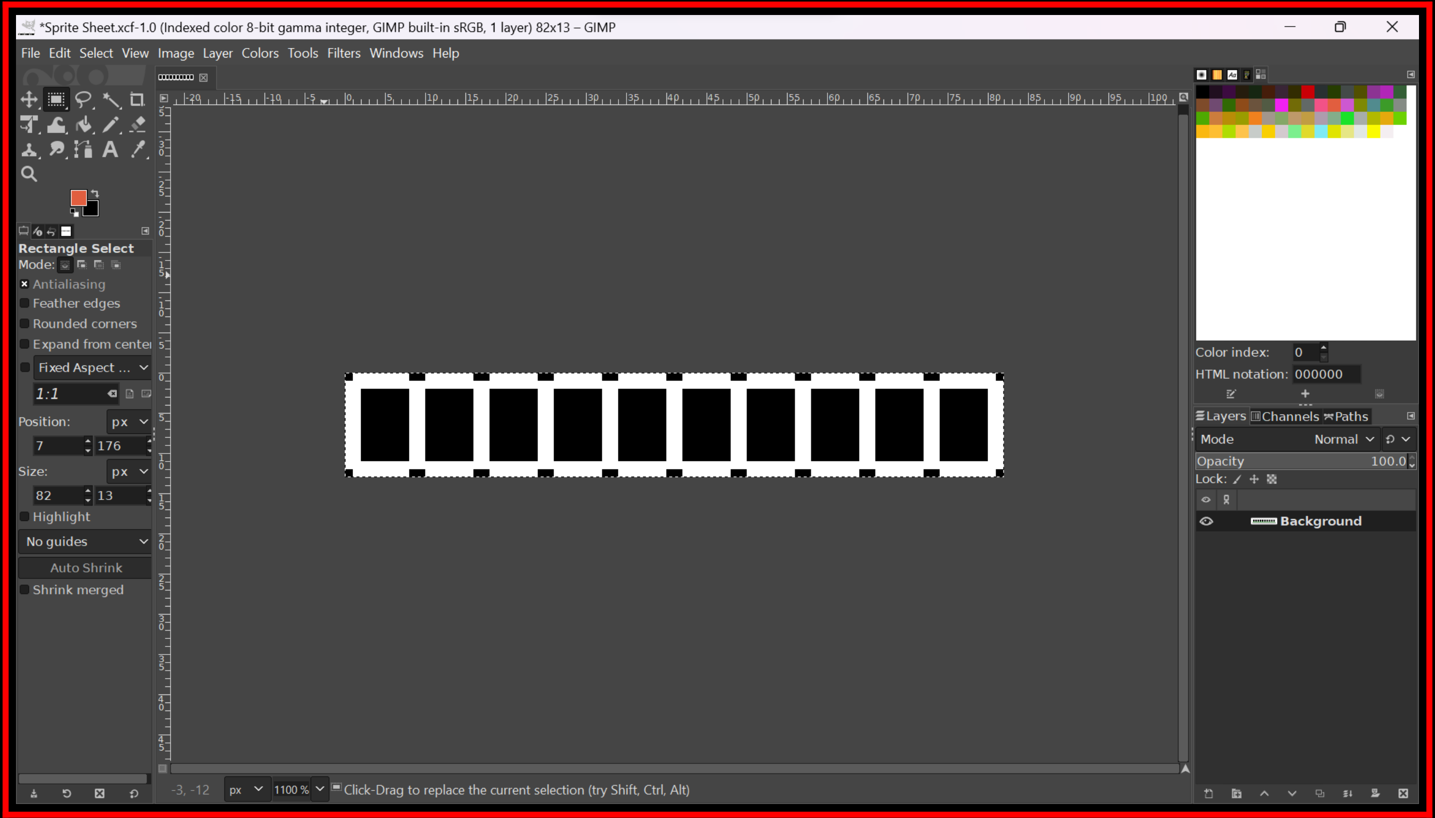
24, -16 px 1100 % Click-Drag to replace the current selection (try Shift, Ctrl, Alt)

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

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

Then paste the data into a Notepad file and save it as "Digits.mem" in the folder "Tutorial_6"

04 Zoom in on the Honeycomb Level Indicator character (inside the yellow border) and using the "rectangle select tool" select around the Level Indicator (this should be a rectangle 82 x 13 pixels) and crop it


*Sprite Sheet.xcf-1.0 (Indexed color 8-bit gamma integer, GIMP built-in sRGB, 1 layer) 82x13 – GIMP

Rectangle Select
Position: 7 176 px
Size: 82 13 px

Click-Drag to replace the current selection (try Shift, Ctrl, Alt)

The image needs to be saved as a Raw Data File, do this using File → Export As → Raw image data. Call the file "Honeycomb1.data"

Using HxD Hex Editor (or similar) load the file "Honeycomb1.data", select all the data and copy it

Then paste the data into a Notepad file and save it as "Honeycomb1.mem" in the folder "Tutorial_6"

05

Using HxD Hex Editor (or similar) load the file “Honeycomb1.data.pal”, select all the data and copy it

Then paste this (overwrite) the data in “Pal24bit.mem” in the folder “Tutorial_6”

(C) CREATING THE PROJECT IN VIVADO

01 Follow the instructions in "Tutorial 1" to create a new project in the "Tutorial_6" folder in Vivado but call the project "BeeInvaders_WIP"

Add these design sources from the "Tutorial 6" folder:

Alien1.mem	BBulletRom.v	DigitsRom.v	Honeycomb1Rom.v
Alien1Rom.v	BBulletSprite.v	Hive1.mem	Honeycomb1Sprite.v
Alien2.mem	Bee.mem	Hive1Ram.v	Pal24bit.mem
Alien2Rom.v	BeeRom.v	Hive2Ram.v	Score.mem
Alien3.mem	BeeSprite.v	Hive3Ram.v	ScoreRom.v
Alien3Rom.v	BHole.mem	Hive4Ram.v	ScoreSprite.v
AlienSprites.v	Debounce.v	HiveSprites.v	Top.v
BBullet.mem	Digits.mem	Honeycomb1.mem	VGA_Timing.v

Add a constraints file from the "Tutorial 6" 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:

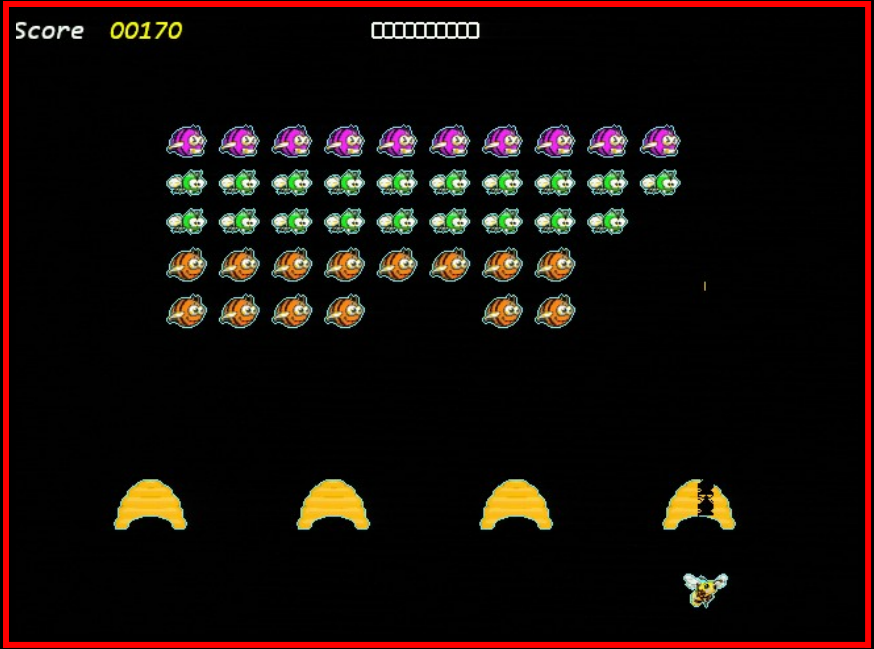
```
.reset(!btn_rst_n),      // reset button is active low
```

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”

With your FPGA board connected, 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 on your VGA monitor that the Alien Bees disappear when shot, the Score is updated, the Alien Bees move faster as more Aliens are shot, the Hives graphics reset after each wave and the Level Indicator increases each time a wave of Aliens are destroyed



(D) THE CODE FOR THIS TUTORIAL

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

```
//----------------------------------------------
// Top.v module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// 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
    input wire btnF                          // Fire 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
wire [9:0] BeeX;
BeeSprite BeeDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .btnR(btnR),
    .btnL(btnL),
    .BeeX(BeeX),
    .BeeSpriteOn(BeeSpriteOn),
    .dataout(dout)
);

// Instantiate BBulletSprite
wire [1:0] BBulletSpriteOn;             // 1=on, 0=off
wire [7:0] BBdout;                      // pixel value from BBullet.mem
wire [9:0] yBBullet;                    // y coordinate for Bee Bullet
wire [9:0] xBBullet;                    // x coordinate for Bee Bullet
reg [1:0] BBulletHive1 = 0;             // 1 = bullet hit hive1, 0 = no hit
reg [1:0] BBulletHive2 = 0;             // 1 = bullet hit hive2, 0 = no hit
reg [1:0] BBulletHive3 = 0;             // 1 = bullet hit hive3, 0 = no hit
reg [1:0] BBulletHive4 = 0;             // 1 = bullet hit hive4, 0 = no hit
wire [1:0] BBulletstate;                // 1 = moving, 2 = stopped
BBulletSprite BBulletDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .btnF(btnF),
    .BeeX(BeeX),
    .BBhithive1(BBhithive1),
    .BBhithive2(BBhithive2),
    .BBhithive3(BBhithive3),
    .BBhithive4(BBhithive4),
    .BBhitAlien1(BBhitAlien1),
    .BBhitAlien2(BBhitAlien2),
    .BBhitAlien3(BBhitAlien3),
    .BBulletSpriteOn(BBulletSpriteOn),
    .BBdout(BBdout),
    .yBBullet(yBBullet),
    .xBBullet(xBBullet),
    .BBulletstate(BBulletstate)
);

// 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 [1:0] LevelSpriteOn;               // Level Indicator
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
wire [7:0] Ldout;                       // pixel value from Honeycomb1.mem
wire [1:0] BBhitAlien1;                 // Bee Bullet Hit Alien1 (0 = No, 1 = Yes)
wire [1:0] BBhitAlien2;                 // Bee Bullet Hit Alien2 (0 = No, 1 = Yes)
wire [1:0] BBhitAlien3;                 // Bee Bullet Hit Alien3 (0 = No, 1 = Yes)
wire [16:0] Score;                      // Score value
wire [1:0] ResetHives;                  // set to 1 if level completed in order to reset hive graphics
AlienSprites AlienDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .xBBullet(xBBullet),                // x coordinate for Bee Bullet
    .yBBullet(yBBullet),                // y coordinate for Bee Bullet
    .BBhitAlien1(BBhitAlien1),
    .BBhitAlien2(BBhitAlien2),
    .BBhitAlien3(BBhitAlien3),
    .Alien1SpriteOn(Alien1SpriteOn),
    .Alien2SpriteOn(Alien2SpriteOn),
    .Alien3SpriteOn(Alien3SpriteOn),
    .LevelSpriteOn(LevelSpriteOn),
    .A1dout(Alien1dout),
    .A2dout(Alien2dout),
    .A3dout(Alien3dout),
    .Ldout(Ldout),
    .Score(Score),
    .ResetHives(ResetHives)
);

// instantiate HiveSprites
wire [1:0] Hive1SpriteOn;               // 1=on, 0=off
wire [1:0] Hive2SpriteOn;               // 1=on, 0=off
wire [1:0] Hive3SpriteOn;               // 1=on, 0=off
wire [1:0] Hive4SpriteOn;               // 1=on, 0=off
wire [7:0] H1dataout;                   // pixel value from Hive1
wire [7:0] H2dataout;                   // pixel value from Hive2
wire [7:0] H3dataout;                   // pixel value from Hive3
wire [7:0] H4dataout;                   // pixel value from Hive4
wire [1:0] BBhithive1;                  // Bee Bullet Hit Hive1 (0 = No, 1 = Yes)
wire [1:0] BBhithive2;                  // Bee Bullet Hit Hive2 (0 = No, 1 = Yes)
wire [1:0] BBhithive3;                  // Bee Bullet Hit Hive3 (0 = No, 1 = Yes)
wire [1:0] BBhithive4;                  // Bee Bullet Hit Hive4 (0 = No, 1 = Yes)
wire [11:0] lpcounter;                  // Loop counter used in resetting hive graphics

HiveSprites HDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .xBBullet(xBBullet),
    .yBBullet(yBBullet),
    .ResetHives(ResetHives),
    .BBhithive1(BBhithive1),
    .BBhithive2(BBhithive2),
    .BBhithive3(BBhithive3),
    .BBhithive4(BBhithive4),
    .Hive1SpriteOn(Hive1SpriteOn),
    .Hive2SpriteOn(Hive2SpriteOn),
    .Hive3SpriteOn(Hive3SpriteOn),
    .Hive4SpriteOn(Hive4SpriteOn),
    .H1dout(H1dataout),
    .H2dout(H2dataout),
    .H3dout(H3dataout),
    .H4dout(H4dataout),
    .lpcounter(lpcounter)
);

// Instantiate Score
wire [1:0] ScoreSpriteOn;               // 1=on, 0=off
wire [7:0] Scoredout;                   // pixel value from Score.mem
```

```
wire [1:0] DigitsSpriteOn;              // 1=on, 0=off
wire [7:0] Digitsdout;                  // pixel value from Digits.mem
wire [10:0] Digitsaddress;              // 11^10 or 2047, need 110 x 13 = 1430
ScoreSprite ScoreDisplay (
    .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .Score(Score),
    .ScoreSpriteOn(ScoreSpriteOn),
    .Scoredout(Scoredout),
    .DigitsSpriteOn(DigitsSpriteOn),
    .Digitsdout(Digitsdout)
);

// Instantiate Honeycomb1
wire [1:0] Honeycomb1SpriteOn;          // 1=on, 0=off
wire [7:0] Honeycomb1dout;              // pixel value from Score.mem
Honeycomb1Sprite Honeycomb1Display (
     .clk_pix(clk_pix),
    .sx(sx),
    .sy(sy),
    .de(de),
    .Honeycomb1SpriteOn(Honeycomb1SpriteOn),
    .Honeycomb1dout(Honeycomb1dout)
);

// 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 (BBulletSpriteOn==1)
                begin
                    vga_r <= (palette[(BBdout*3)])>>4;      // RED bits(7:4) from colour palette
                    vga_g <= (palette[(BBdout*3)+1])>>4;    // GREEN bits(7:4) from colour palette
                    vga_b <= (palette[(BBdout*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
                if (Hive1SpriteOn==1)
                    begin
                        vga_r <= (palette[(H1dataout*3)])>>4;       // RED bits(7:4) from colour palette
                        vga_g <= (palette[(H1dataout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(H1dataout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
                    end
                else
                if (Hive2SpriteOn==1)
                    begin
                        vga_r <= (palette[(H2dataout*3)])>>4;       // RED bits(7:4) from colour palette
                        vga_g <= (palette[(H2dataout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(H2dataout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
                    end
                else
                if (Hive3SpriteOn==1)
                    begin
                        vga_r <= (palette[(H3dataout*3)])>>4;       // RED bits(7:4) from colour palette
                        vga_g <= (palette[(H3dataout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(H3dataout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
                    end
                else
                if (Hive4SpriteOn==1)
                    begin
                        vga_r <= (palette[(H4dataout*3)])>>4;       // RED bits(7:4) from colour palette
                        vga_g <= (palette[(H4dataout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(H4dataout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
                    end
                else
                if (ScoreSpriteOn==1)
                    begin
                        vga_r <= (palette[(Scoredout*3)])>>4;       // RED bits(7:4) from colour palette
                        vga_g <= (palette[(Scoredout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(Scoredout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
                    end
                else
                if (DigitsSpriteOn==1)
                    begin
                        vga_r <= (palette[(Digitsdout*3)])>>4;       // RED bits(7:4) from colour palette
                        vga_g <= (palette[(Digitsdout*3)+1])>>4;     // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(Digitsdout*3)+2])>>4;     // BLUE bits(7:4) from colour palette
                    end
               else
                    if (LevelSpriteOn==1)
                    begin
                        vga_r <= (palette[(Ldout*3)])>>4;           // RED bits(7:4) from colour palette
                        vga_g <= (palette[(Ldout*3)+1])>>4;         // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(Ldout*3)+2])>>4;         // BLUE bits(7:4) from colour palette
                    end
                else
                    if (Honeycomb1SpriteOn==1)
                    begin
                        vga_r <= (palette[(Honeycomb1dout*3)])>>4;        // RED bits(7:4) from colour palette
                        vga_g <= (palette[(Honeycomb1dout*3)+1])>>4;      // GREEN bits(7:4) from colour palette
                        vga_b <= (palette[(Honeycomb1dout*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_6
// 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”

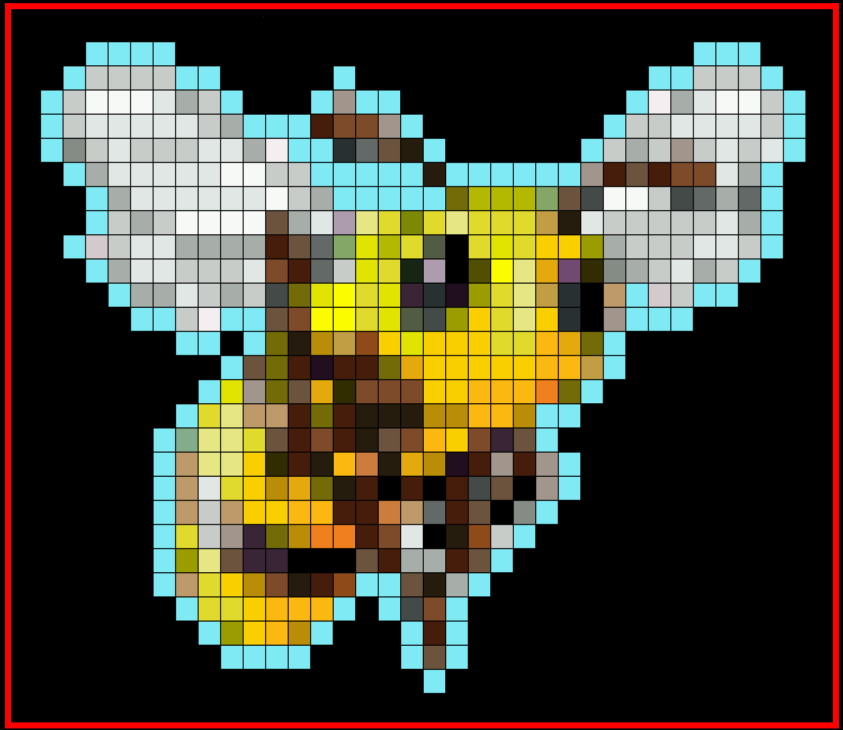
```
//--------------------------------------------
// BeeSprite.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//--------------------------------------------
`timescale 1ns / 1ps

// Setup BeeSprite module
module BeeSprite(
    input wire clk_pix,                 // 25.2MHz 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
    input wire btnR,                    // right button
    input wire btnL,                    // left button
    output reg [9:0] BeeX,              // Bee X position
    output reg [1:0] BeeSpriteOn,       // 1=on, 0=off
    output wire [7:0] dataout           // pixel value from Bee.mem
    );

    // instantiate BeeRom
    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 [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
        if(de)
            begin
                if((sx==BeeX-2) && (sy==BeeY))
                    begin
                        address <= 0;
                        BeeSpriteOn <=1;
                    end
                if((sx>BeeX-2) && (sx<BeeX+BeeWidth-1) && (sy>BeeY-1) && (sy<BeeY+BeeHeight))
                    begin
                        address <= address +1;
                        BeeSpriteOn <=1;
                    end
                else
                        BeeSpriteOn <=0;
            end

        // if left or right button pressed, move the Bee
        if (BeeX == 0)
            BeeX <= 320;                                        // initailise Bee x position
        if ((sx==64) && (sy==480))                              // check for buttons once every frame
            begin
                if ((sig_right == 1) && (BeeX<640-BeeWidth))    // Check for right button
                    BeeX<=BeeX+1;                               // move right
                if ((sig_left == 1) && (BeeX>2))                // Check for left button
                    BeeX<=BeeX-1;                               // move left
            end
    end
endmodule
```

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

```
//------------------------------------------------
// BeeRom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// 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,         // pixel clock
    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 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 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 03 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 39 35 34 3C 3C 2C 2C 2C 2C 05 14 18 26 3A 2D 38 15 00 38 3A 38 36 36 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 35 34 39 39
00 00 00 00 00 00 00 39 39 34 3E 39 39 14 10 3D 3D 38 3A 15 0B 23 36 38 3B 36 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 38
3A 36 36 36 38 38 30 2E 17 39 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 36 36 36 36 36 30 30 28 39 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 36 36 30 30 30 24 17 39 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 39 2A 3B 3B 38 14 05 10 05 03 14 10 30 36 10 06 14 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
39 27 3B 3B 36 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 39 27 3C 38 36 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 39 27 34 27 36 36 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 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 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 39 27 38 36 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 36 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 39 23 36 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 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_6
// 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 “BBulletSprite.v”

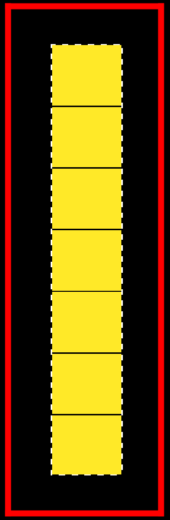
```
//--------------------------------------------
// BBulletSprite.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//--------------------------------------------
`timescale 1ns / 1ps

// Setup BBulletSprite module
module BBulletSprite(
    input wire clk_pix,                 // 25.2MHz pixel clock
    input wire [9:0] sx,                // current x position
    input wire [9:0] sy,                // current y position
    input wire de,                      // 1 = visible pixels, 0 = blanking period
    input wire btnF,                    // fire button
    input wire [9:0] BeeX,              // bee bullet x position
    input wire [1:0] BBhithive1,        // 1 = bullet hit hive, 0 = no hit
    input wire [1:0] BBhithive2,        // 1 = bullet hit hive, 0 = no hit
    input wire [1:0] BBhithive3,        // 1 = bullet hit hive, 0 = no hit
    input wire [1:0] BBhithive4,        // 1 = bullet hit hive, 0 = no hit
    input wire [1:0] BBhitAlien1,       // 1 = bullet hit Alien1, 0 = no hit
    input wire [1:0] BBhitAlien2,       // 1 = bullet hit Alien2, 0 = no hit
    input wire [1:0] BBhitAlien3,       // 1 = bullet hit Alien3, 0 = no hit
    output reg [1:0] BBulletSpriteOn,   // 1=on, 0=off
    output wire [7:0] BBdout,           // pixel value from BBullet.mem
    output reg [9:0] yBBullet,          // y coordinate for Bee Bullet
    output reg [9:0] xBBullet,          // bee bullet x position
    output reg [1:0] BBulletstate       // 1 = moving, 2 = stopped
    );

    // instantiate BBulletRom
    reg [3:0] BBaddress;                // 2^4 or 15, need 1 x 7 = 7
    BBulletRom BBulletVRom (
        .BBaddress(BBaddress),
        .clk_pix(clk_pix),
        .BBdout(BBdout)
    );

    // Instantiate Debounce
    Debounce deb_fire (
        .clk_pix(clk_pix),
        .btn(btnF),
        .out(sig_fire)
    );

    // setup character size
    localparam BBWidth = 1;             // Bee Bullet width in pixels
    localparam BBHeight = 7;            // Bee Bullet height in pixels

    always @ (posedge clk_pix)
    begin
        // if sx,sy are within the confines of the Bee Bullet character, switch the Bee Bullet On
        if((de) && (BBulletstate == 1))
            begin
                if((sx==xBBullet-2) && (sy==yBBullet))
                    begin
                        BBaddress <= 0;
                        BBulletSpriteOn <=1;
                    end
                if((sx>xBBullet-2) && (sx<xBBullet+BBWidth-1) && (sy>yBBullet-1) && (sy<yBBullet+BBHeight))
                    begin
                        BBaddress <= BBaddress +1;
                        BBulletSpriteOn <=1;
                    end
                else
                    BBulletSpriteOn <=0;
            end
        else
        // if fire button pressed, move the Bee Bullet up the screen
        if ((sx==640) && (sy==480))                         // check for movement once every frame
        begin
            if (BBulletstate == 0)
                begin
                    BBulletstate <= 2;                      // initialise BBulletstate = stopped
                end
            if ((sig_fire == 1) && (xBBullet == 0))         // Check for fire button and bullet stopped
                begin
                    xBBullet <= BeeX + 16;
                    yBBullet<=425;
                    BBulletstate<=1;                        // 1 = bullet moving, 2 = bullet stopped
                end
            if ((BBulletstate == 1))
                begin
                    yBBullet<=yBBullet-2;                   // move bullet up the screen
                    if ((BBhithive1 == 1) || (BBhithive2 == 1) || (BBhithive3 == 1) || (BBhithive4 == 1) || (BBhitAlien1 == 1) || (BBhitAlien2 == 1) ||
(BBhitAlien3 == 1)) // Check if Bee Bullet has hit hive1-4 and Alien1-3
                        begin
                            BBulletstate<=2;                // stop the bullet
                            yBBullet<=425;                  // bullet y start position
                            xBBullet<=0;
                        end
                    if (yBBullet<10)                        // Check if Bee Bullet at top of screen
                        begin
                            BBulletstate<=2;                // stop the bullet
                            yBBullet<=425;                  // bullet y start position
                            xBBullet<=0;
                        end
                end
        end
    end
endmodule
```

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

```
//--------------------------------------------
// BBulletRom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//--------------------------------------------
`timescale 1ns / 1ps

// Setup BBulletRom module
module BBulletRom(
    input wire [3:0] BBaddress, // 2^4 or 15, need 1 x 7 = 7
    input wire clk_pix,
    output reg [7:0] BBdout     // (7:0) 8 bit pixel value from BBullet.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] BBmemory_array [0:6]; // 8 bit values for 7 pixels of BBullet (1 x 7)

    initial
    begin
        $readmemh("BBullet.mem", BBmemory_array);
    end

    always @ (posedge clk_pix)
            BBdout <= BBmemory_array[BBaddress];
endmodule
```

This is the data from the file "Bbullet.mem" - Sprite Size 1 x 7 pixels

```
35 35 35 35 35 35 35
```



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

```
//--------------------------------------------------
// AlienSprites.v module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// 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
    input wire [9:0] xBBullet,          // x coordinate for Bee Bullet
    input wire [9:0] yBBullet,          // y coordinate for Bee Bullet
    input wire [11:0] lpcounter,        // Loop counter used in resetting hive graphics
    output reg [1:0] BBhitAlien1,       // 1=hit, 0=no hit
    output reg [1:0] BBhitAlien2,       // 1=hit, 0=no hit
    output reg [1:0] BBhitAlien3,       // 1=hit, 0=no hit
    output reg [1:0] Alien1SpriteOn,    // 1=on, 0=off
    output reg [1:0] Alien2SpriteOn,    // 1=on, 0=off
    output reg [1:0] Alien3SpriteOn,    // 1=on, 0=off
    output reg [1:0] LevelSpriteOn,     // 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
    output reg [7:0] Ldout,             // 8 bit pixel value from Honeycomb1.mem
    output reg [16:0] Score,            // 8 bit pixel value from Score.mem
    output reg [1:0] ResetHives         // set to 1 if level completed in order to reset hive graphics
    );

// 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 Alien 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
    reg [1:0] Adir = 2;                 // direction of aliens: 2=right, 1=left
    reg [3:0] delaliens = 0;            // counter to slow alien movement
    reg [3:0] delloop = 8;              // counter end value for delaliens
    reg [3:0] M = 4;                    // move Aliens by this number of pixels

    reg [0:10] alienc1=11'b11111111111;  //------------------------------------
    reg [0:10] alienc2=11'b11111111111; //
    reg [0:10] alienc3=11'b11111111111;  // Set pattern of Aliens 11 x 5 = 55
    reg [0:10] alienc4=11'b11111111111;  //
    reg [0:10] alienc5=11'b11111111111;  //------------------------------------

    // setup Level Indicator positions and sizes
    localparam LevelWidth = 6;          // Level Indicator width in pixels
    localparam LevelHeight = 9;         // Level Indicator height in pixels
    reg [9:0] LevelX = 281;             // X position for Honeycomb1 Level Indicator
    reg [8:0] LevelY = 8;               // Y position for Honeycomb1 Level Indicator
    reg [3:0] Level=0;                  // Level Number
    reg [5:0] AlienQ=55;                // Alien quantity in the wave
    reg [9:0] LcounterW=0;              // Counter to check if Honeycomb1 Level Indicator width reached
    reg [9:0] LcounterH=0;              // Counter to check if Honeycomb1 Level Indicator height reached

    always @ (posedge clk_pix)
    begin
        // Initially set ResetHives to 0
        if(Score==0)
            ResetHives<=0;
        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-1) && (sy>A1Y+AoY-1) && (sy<A1Y+A1Height+AoY))
                    begin
                        A1address <= A1address + 1;
                        AcounterW <= AcounterW + 1;
                        Alien1SpriteOn <=1;
                        if(alienc1[AoX/40]==0)
                            Alien1SpriteOn <=0;
                        if (AcounterW==A1Width-1)
                            begin
                                AcounterW <= 0;
                                if(AcolCount>1)
                                    AoX <= AoX + 40;
                                if(AoX<(AcolCount-1)*40)
                           A1address <= A1address - (A1Width-1);
                         if(AoX==(AcolCount-1)*40)
                           AoX<=0;
                       end
                        // Check if Bee Bullet Has Hit Alien1
                        if ((sx == xBBullet) && (sy == yBBullet) && (sx>A1X+AoX-1) && (BBhitAlien1 == 0) && (A1dout > 0) && (alienc1[AoX/40]==1))
                            begin
                                BBhitAlien1 <= 1;
                                alienc1[AoX/40]<=0;
                            end
                    end
                else
                    Alien1SpriteOn <=0;

                // Alien2
                if (sx==A2X+AoX-2 && sy==A2Y+AoY)
                    begin
                        A2address <= 0;
                        Alien2SpriteOn <=1;
                        AcounterW<=0;
                        AcounterH<=0;
                    end
                if ((sx>A2X+AoX-2) && (sx<A2X+A2Width+AoX-1) && (sy>A2Y+AoY-1) && (sy<A2Y+AoY+A2Height))
                    begin
                        A2address <= A2address + 1;
                        AcounterW <= AcounterW + 1;
                        Alien2SpriteOn <=1;
                        if((alienc2[AoX/40]==0) && (sy-A2Y<A2Height+1))
                            Alien2SpriteOn <=0;
                        if((alienc3[AoX/40]==0) && (sy-A2Y>A2Height))
                            Alien2SpriteOn <=0;
                        if (AcounterW==A2Width-1)
                            begin
                                AcounterW <= 0;
                                if(AcolCount>1)
                                    AoX <= AoX + 40;
                                if(AoX<(AcolCount-1)*40)
                           A2address <= A2address - (A2Width-1);
                         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
                        // Check if Bee Bullet Has Hit Alien2
                        if ((sx == xBBullet) && (sy == yBBullet) && (sx>A2X+AoX-1) && (BBhitAlien2 == 0) && (A2dout > 0))
                            begin
                                if((sy-A2Y<A2Height+1) && (alienc2[AoX/40]==1))
```

```
                    begin
                        BBhitAlien2 <= 1;
                        alienc2[AoX/40]<=0;
                    end
                if((sy-A2Y>A2Height) && (alienc3[AoX/40]==1))
                    begin
                        BBhitAlien2 <= 1;
                        alienc3[AoX/40]<=0;
                    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-1) && (sy>A3Y+AoY-1) && (sy<A3Y+AoY+A3Height))
    begin
        A3address <= A3address + 1;
        AcounterW <= AcounterW + 1;
        Alien3SpriteOn <=1;
        if((alienc4[(AoX)/40]==0) && (sy-A3Y<A3Height+1))
            Alien3SpriteOn <=0;
        if((alienc5[(AoX)/40]==0) && (sy-A3Y>A3Height))
            Alien3SpriteOn <=0;
        if (AcounterW==A3Width-1)
            begin
                AcounterW <= 0;
                if(AcolCount>1)
                    AoX <= AoX + 40;
                if((AoX<(AcolCount-1)*40))
            A3address <= A3address - (A3Width-1);
         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
        // Check if Bee Bullet Has Hit Alien3
        if ((sx == xBBullet) && (sy == yBBullet) && (sx>A3X+AoX-1) && (BBhitAlien3 == 0) && (A3dout > 0))
            begin
                if((sy-A3Y<A3Height+1) && (alienc4[(AoX)/40]==1))
                    begin
                        BBhitAlien3 <= 1;
                        alienc4[(AoX)/40]<=0;
                    end
```

```
                                        if((sy-A3Y>A3Height) && (alienc5[(AoX)/40]==1))
                                            begin
                                                BBhitAlien3 <= 1;
                                                alienc5[(AoX)/40]<=0;
                                            end
                                    end
                            end
                        else
                            Alien3SpriteOn <=0;

                        //Level Indicator
                        if(Level>0)
                            begin
                                if(Level>9)
                                    Level<=0;
                                if (sx==LevelX-2 && sy==LevelY)
                                    begin
                                        Ldout <= 26;
                                        LevelSpriteOn <=1;
                                        LcounterW<=0;
                                        LcounterH<=0;
                                    end
                                if ((sx>LevelX-2) && (sx<LevelX+LevelWidth) && (sy>LevelY-1) && (sy<LevelY+LevelHeight))
                                    begin
                                        Ldout <= 26;
                                        LcounterW <= LcounterW + 1;
                                        LevelSpriteOn <=1;
                                        if(LcounterW==LevelWidth-1)
                                            begin
                                                if(LevelX<((Level-1)*8)+281)
                                                    begin
                                                      LcounterW <= 0;
                                                      LevelX <= LevelX + 8;
                                                    end
                                                else
                                                    begin
                                               LevelX = 281;
                                               LcounterH <= LcounterH + 1;
                                               LcounterW <= 0;
                                               if(LcounterH==LevelHeight-1)
                                                    begin
                                                         LevelX = 281;
                                                         LcounterH <= 0;
                                                         LcounterW <= 0;
                                                       end
                                              end
                                   end
                                    end
                               else
                                    LevelSpriteOn <=0;
                            end
                    end
                else
                // check if a column of Aliens have all been shot
                if (sx>640 && sy>480)
                    begin
                        if((alienc1[AcolCount-1]==0) && (alienc2[AcolCount-1]==0) && (alienc3[AcolCount-1]==0) && (alienc4[AcolCount-1]==0) && (alienc5[AcolCount-1]==0)
        && (AcolCount>1))
                                AcolCount<=AcolCount-1;
                        else
                        if((alienc1[0]==0) && (alienc2[0]==0) && (alienc3[0]==0) && (alienc4[0]==0) && (alienc5[0]==0) && (AcolCount>1))
                            begin
                                AcolCount<=AcolCount-1;
                                alienc1<=alienc1<<1;
                                alienc2<=alienc2<<1;
                                alienc3<=alienc3<<1;
                                alienc4<=alienc4<<1;
                                alienc5<=alienc5<<1;
                                A1X<=A1X+40;
                                A2X<=A2X+40;
                                A3X<=A3X+40;
                            end
                   end
                   // slow down the alien movement / move aliens left or right
                   if (sx==640 && sy==480)
                    begin
                        delaliens<=delaliens+1;
                        if (delaliens>delloop)
                            begin
                                delaliens<=0;
                                if (Adir==2)
                                    begin
                                        A1X<=A1X+M;
                                        A2X<=A2X+M;
                                        A3X<=A3X+M;
                                        if (A1X+A1Width+((AcolCount-1)*40)>(640-(M*2)-1))
                                            Adir<=1;
                                    end
                                else
                                if (Adir==1)
                                    begin
                                        A1X<=A1X-M;
                                        A2X<=A2X-M;
                                        A3X<=A3X-M;
                                        if (A1X<(M*2)+2)
                                            Adir<=2;
                                    end
                            end
                       // If Alien has been shot increase the Score / If all Aliens have been shot reset the wave and Hives
                       if(BBhitAlien1==1)
                            begin
                                BBhitAlien1<=0;
                                Score<=Score+30;
                                AlienQ<=AlienQ-1;
                            end
                       if(BBhitAlien2==1)
                            begin
                                BBhitAlien2<=0;
                                Score<=Score+20;
                                AlienQ<=AlienQ-1;
                            end
                       if(BBhitAlien3==1)
                            begin
                                BBhitAlien3<=0;
                                Score<=Score+10;
                                AlienQ<=AlienQ-1;
                            end
                        if(Score>99999)
                                begin
                                    Score<=Score-100000;
                                end
                        if(AlienQ<1 && Score>0)
                            begin
                                Level<=Level+1;
```

```
                        AcolCount<=11;
                        AlienQ<=55;
                        delloop<=8;
                        A1X<=135;
                        A2X<=135;
                        A3X<=135;
                        M<=4;
                        alienc1<=11'b11111111111;
                        alienc2<=11'b11111111111;
                        alienc3<=11'b11111111111;
                        alienc4<=11'b11111111111;
                        alienc5<=11'b11111111111;
                        ResetHives<=1;
                    end
                if(AlienQ>0)
                        ResetHives<=0;

                // Alien speed up stages
                if(AlienQ==50)
                    delloop<=7;
                else
                if(AlienQ==43)
                    delloop<=6;
                else
                if(AlienQ==36)
                    delloop<=5;
                else
                if(AlienQ==29)
                    delloop<=4;
                else
                if(AlienQ==22)
                    delloop<=3;
                else
                if(AlienQ==15)
                    delloop<=2;
                else
                if(AlienQ==8)
                    delloop<=1;
                else
                if(AlienQ==1)
                    M<=6;
            end
        end
    endmodule
```

This is the code from the file “Alien1Rom.v”

```
//-----------------------------------------------
// Alien1Rom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// 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”

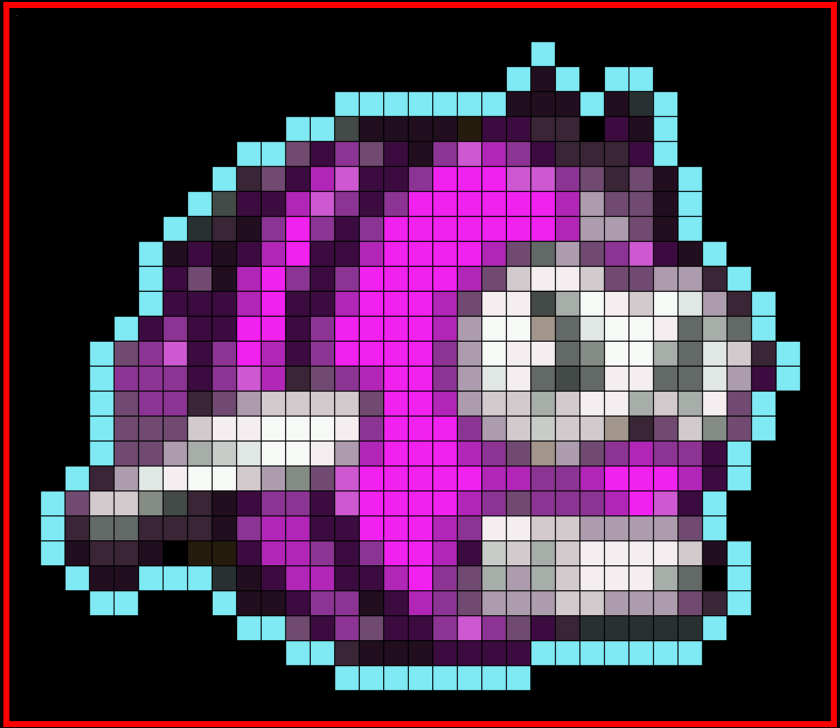
```
//-----------------------------------------------
// Alien2Rom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// 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_6
// 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
```

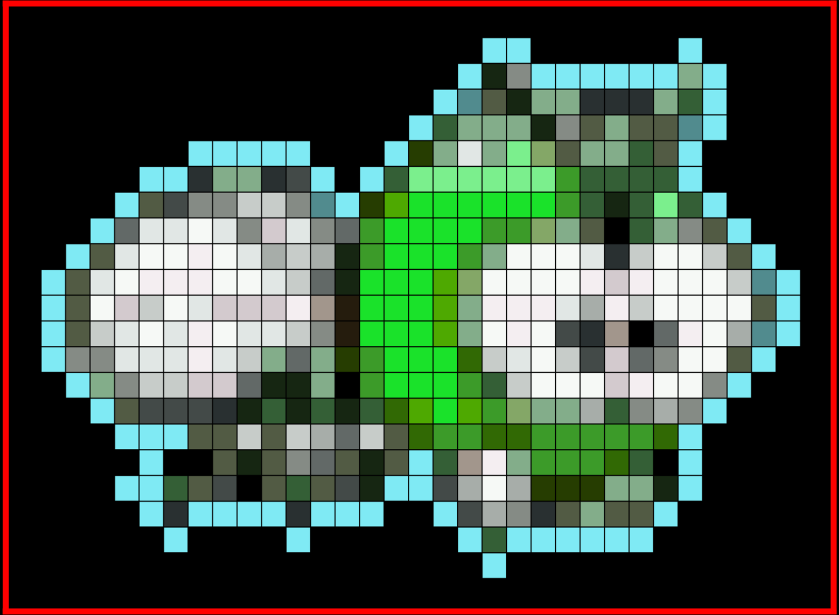
This is the data from the file "Alien1.mem" - Sprite Size 31 x 26 pixels

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 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 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 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 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 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 39 02 11 01 0E 16 0D 02 0D 16 16 16 16 0E 11 35 3E 3E 35 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 35 3F 3C 29 06 39 00 00 00 00 39 02 0D 02 02 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 35 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 35 35 35 35 11 16 16 0E 29 35 35 2C 35 3E 3E 2C 35 2C 3E 11 39 00 00 00 39 11 11 11 35 3E 3E 3F 3F 3F
3E 0D 16 16 16 0D 29 35 34 35 35 25 06 11 35 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 35 29 1F 11 1B 16 16 16 16 16 0E 0E 0D 0D 0E 16 16 16 0E 02 39 00 00 39 11 35 35 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 35 35 29 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 35 2C 35 3E 3E 3E 3E 35 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 35 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 35 35 29 29 29 11 06 39 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 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
```



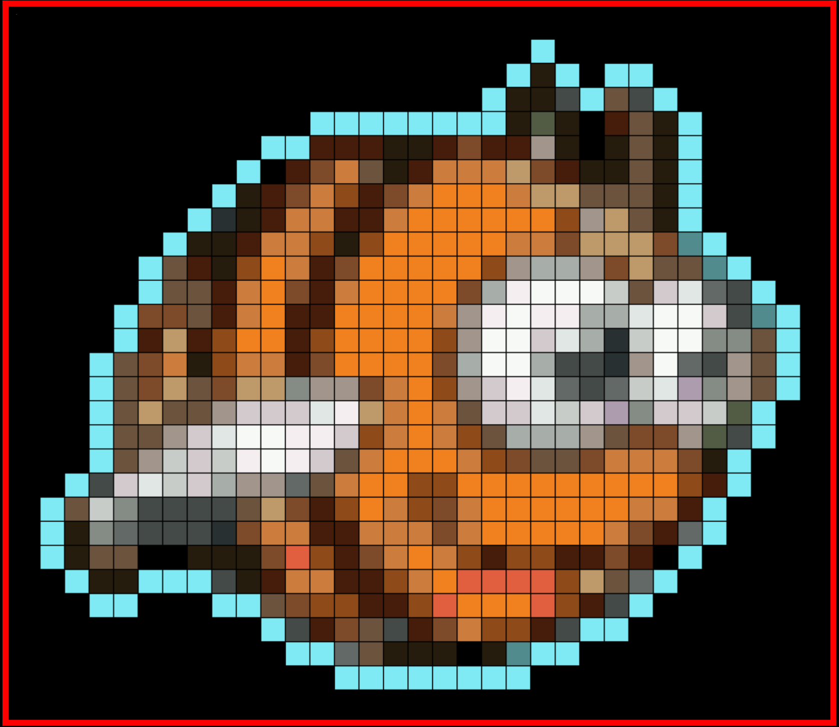
This is the data from the file "Alien2.mem" - Sprite Size 31 x 21 pixels

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 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 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 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 39 0F 2A 2A 2A 04 1F 15 2A 15 15 1D 39 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
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 35 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 35 3E 3F 3F 3F 34 1D 39 39 15 3F 35 34 3F 3C 35
35 35 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 35 18 1F 3F 3F 15 39 00 00 39 2A 1F 34 34 35 35 18 04 04 2A 00 1E 2B 2B 2B 1E 0F 34 3F
3F 3F 35 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 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 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 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 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 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 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 13 25 27 14 03 39 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 39 14 05 03 13 24 21 05 10 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 3F 3F 3F 3F 34 14 35 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 35 0B 1D
39 00 00 00 39 05 27 05 13 24 24 05 13 24 24 24 24 13 25 3F 3F 35 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 35 3E 3C 18 0B 18 34 3C 29 1F 25 14 39 00 00 39 14 27 14 14 25 35 35 35 3C
3E 27 21 24 21 14 35 35 3C 34 35 29 1F 35 35 34 15 39 00 00 00 39 14 14 25 35 3C 3F 3F 3E 3E 35 13 21 24 21 13 14 2C 2C 2C 25 14 10 10 25 15 0B 39 00 00 00 39
14 25 34 35 34 3E 3F 3E 35 14 21 24 24 24 21 13 10 14 14 10 21 21 21 10 03 39 00 00 00 39 0B 35 3C 34 35 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 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 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 39 39 39 39 39 39 39 39 00 00 00 00 00 00 00 00 00 00 00
```



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

```
//-----------------------------------------------
// HiveSprites.v module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup HiveSprites Module
module HiveSprites(
    input wire clk_pix,             // 25.2MHz 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
    input wire [9:0] xBBullet,      // x coordinate for Bee Bullet
    input wire [9:0] yBBullet,      // y coordinate for Bee Bullet
    input wire [1:0] ResetHives,    // set to 1 if level completed in order to reset hive graphics
    output reg [1:0] BBhithive1,    // set to 1 when Bee Bullet hit hive1
    output reg [1:0] BBhithive2,    // set to 1 when Bee Bullet hit hive2
    output reg [1:0] BBhithive3,    // set to 1 when Bee Bullet hit hive3
    output reg [1:0] BBhithive4,    // set to 1 when Bee Bullet hit hive4
    output reg [1:0] Hive1SpriteOn, // 1=on, 0=off
    output reg [1:0] Hive2SpriteOn, // 1=on, 0=off
    output reg [1:0] Hive3SpriteOn, // 1=on, 0=off
    output reg [1:0] Hive4SpriteOn, // 1=on, 0=off
    output wire [7:0] H1dout,       // 8 bit pixel value from Hive1
    output wire [7:0] H2dout,       // 8 bit pixel value from Hive2
    output wire [7:0] H3dout,       // 8 bit pixel value from Hive3
    output wire [7:0] H4dout,       // 8 bit pixel value from Hive4
    output reg [11:0] lpcounter     // Loop counter used in resetting hive graphics
    );

    // instantiate Hive1Ram
    reg [1:0] write1;               // write1 = 0 (read data), write1 = 1 (write data)
    reg [7:0] data1;                // Data to write if write1 = 1 (write data)
    reg [11:0] H1address;           // Address to read/write data from/to Hive1 (2^12 or 4095, need 66 x 39 = 2574)

    Hive1Ram Hive1VRam (
        .clk_pix(clk_pix),
        .H1address(H1address),
        .write1(write1),
        .data1(data1),
        .H1dout(H1dout)
    );

    // instantiate Hive2Ram
    reg [1:0] write2;               // write2 = 0 (read data), write2 = 1 (write data)
    reg [7:0] data2;                // Data to write if write2 = 1 (write data)
    reg [11:0] H2address;           // Address to read/write data from/to Hive2 (2^12 or 4095, need 66 x 39 = 2574)

    Hive2Ram Hive2VRam (
        .clk_pix(clk_pix),
        .H2address(H2address),
        .write2(write2),
        .data2(data2),
        .H2dout(H2dout)
    );
```

```
// instantiate Hive3Ram
reg [1:0] write3;                // write3 = 0 (read data), write3 = 1 (write data)
reg [7:0] data3;                 // Data to write if write3 = 1 (write data)
reg [11:0] H3address;            // Address to read/write data from/to Hive3 (2^12 or 4095, need 66 x 39 = 2574)

Hive3Ram Hive3VRam (
    .clk_pix(clk_pix),
    .H3address(H3address),
    .write3(write3),
    .data3(data3),
    .H3dout(H3dout)
);

// instantiate Hive4Ram
reg [1:0] write4;                // write4 = 0 (read data), write4 = 1 (write data)
reg [7:0] data4;                 // Data to write if write4 = 1 (write data)
reg [11:0] H4address;            // Address to read/write data from/to Hive4 (2^12 or 4095, need 66 x 39 = 2574)

Hive4Ram Hive4VRam (
    .clk_pix(clk_pix),
    .H4address(H4address),
    .write4(write4),
    .data4(data4),
    .H4dout(H4dout)
);

// Load Hive1.mem into register, used to reset the hive graphics
(*RAM_STYLE="block"*) reg [7:0] Hivememory_array [0:2573]; // 8 bit values for 2574 pixels of Hive1 (66 x 39)
initial $readmemh("Hive1.mem", Hivememory_array);

// Load BHole.mem
reg [7:0] BHoleaddress;                  // 2^8 or 255, need 11 x 16 = 176
reg [7:0] BHoledata [0:175];             // 8 bit values from BHole.mem
initial begin
    $readmemh("BHole.mem", BHoledata);  // load 176 hex values into BHole.mem
end

// setup Hive character positions and sizes
localparam Hive1X = 78;             // Hive1 X start position
localparam Hive1Y = 360;            // Hive1 Y start position
localparam Hive2X = 217;            // Hive2 X start position
localparam Hive2Y = 360;            // Hive2 Y start position
localparam Hive3X = 356;            // Hive3 X start position
localparam Hive3Y = 360;            // Hive3 Y start position
localparam Hive4X = 495;            // Hive4 X start position
localparam Hive4Y = 360;            // Hive4 Y start position
localparam HiveWidth = 66;          // Hive width in pixels
localparam HiveHeight = 39;         // Hive height in pixels

reg [9:0] BHit1x;                   // Saves the x position of where the bee bullet hit hive1
reg [9:0] BHit1y;                   // Saves the y position of where the bee bullet hit hive1
reg [9:0] BHit2x;                   // Saves the x position of where the bee bullet hit hive2
reg [9:0] BHit2y;                   // Saves the y position of where the bee bullet hit hive2
reg [9:0] BHit3x;                   // Saves the x position of where the bee bullet hit hive3
reg [9:0] BHit3y;                   // Saves the y position of where the bee bullet hit hive3
reg [9:0] BHit4x;                   // Saves the x position of where the bee bullet hit hive4
reg [9:0] BHit4y;                   // Saves the y position of where the bee bullet hit hive4
reg [3:0] hzcounter1 = 0;           // Hole 11 pixels wide - Hive1
reg [4:0] vtcounter1 = 0;           // Hole 16 pixels high - Hive1
reg [3:0] hzcounter2 = 0;           // Hole 11 pixels wide - Hive2
reg [4:0] vtcounter2 = 0;           // Hole 16 pixels high - Hive2
reg [3:0] hzcounter3 = 0;           // Hole 11 pixels wide - Hive2
reg [4:0] vtcounter3 = 0;           // Hole 16 pixels high - Hive2
reg [3:0] hzcounter4 = 0;           // Hole 11 pixels wide - Hive2
reg [4:0] vtcounter4 = 0;           // Hole 16 pixels high - Hive2

always @ (posedge clk_pix)
begin
    if (de)
        begin
            // check if sx,sy are within the confines of the Hive character - hive1
            if ((sx==Hive1X-2) && (sy==Hive1Y))
                begin
                    write1<=0;
                    H1address <= 0;
                    Hive1SpriteOn <=1;
                end
            if ((sx>Hive1X-2) && (sx<Hive1X+HiveWidth-1) && (sy>Hive1Y-1) && (sy<Hive1Y+HiveHeight))
                begin
                    write1<=0;
                    H1address <= H1address + 1;
                    Hive1SpriteOn <= 1;
                    // Check if Bee Bullet Has Hit Hive 1
                    if ((sx == xBBullet) && (sy == yBBullet) && (sx>Hive1X-1) && (BBhithive1 == 0) && (H1dout > 0))
                        begin
                            BBhithive1 <= 1;
                            BHit1x <= xBBullet - 5;
                            BHit1y <= yBBullet - 13;
                        end
                end
            else
                Hive1SpriteOn <= 0;

            // check if sx,sy are within the confines of the Hive character - hive2
            if ((sx==Hive2X-2) && (sy==Hive2Y))
                begin
                    write2<=0;
                    H2address <= 0;
                    Hive2SpriteOn <=1;
                end
            if ((sx>Hive2X-2) && (sx<Hive2X+HiveWidth-1) && (sy>Hive2Y-1) && (sy<Hive2Y+HiveHeight))
                begin
                    write2<=0;
                    H2address <= H2address + 1;
                    Hive2SpriteOn <= 1;
                    // Check if Bee Bullet Has Hit Hive 2
                    if ((sx == xBBullet) && (sy == yBBullet) && (sx>Hive2X-1) && (BBhithive2 == 0) && (H2dout > 0))
                        begin
                            BBhithive2 <= 1;
                            BHit2x <= xBBullet - 5;
                            BHit2y <= yBBullet - 13;
                        end
                end
            else
                Hive2SpriteOn <= 0;

             // check if sx,sy are within the confines of the Hive character - hive3
            if ((sx==Hive3X-2) && (sy==Hive3Y))
                begin
                    write3<=0;
                    H3address <= 0;
                    Hive3SpriteOn <=1;
                end
            if ((sx>Hive3X-2) && (sx<Hive3X+HiveWidth-1) && (sy>Hive3Y-1) && (sy<Hive3Y+HiveHeight))
                begin
                    write3<=0;
                    H3address <= H3address + 1;
                    Hive3SpriteOn <= 1;
                    // Check if Bee Bullet Has Hit Hive 3
                    if ((sx == xBBullet) && (sy == yBBullet) && (sx>Hive3X-1) && (BBhithive3 == 0) && (H3dout > 0))
                        begin
                            BBhithive3 <= 1;
                            BHit3x <= xBBullet - 5;
                            BHit3y <= yBBullet - 13;
                        end
                end
            else
                Hive3SpriteOn <= 0;

             // check if sx,sy are within the confines of the Hive character - hive4
            if ((sx==Hive4X-2) && (sy==Hive4Y))
                begin
                    write4<=0;
                    H4address <= 0;
                    Hive4SpriteOn <=1;
                end
            if ((sx>Hive4X-2) && (sx<Hive4X+HiveWidth-1) && (sy>Hive4Y-1) && (sy<Hive4Y+HiveHeight))
                begin
                    write4<=0;
                    H4address <= H4address + 1;
                    Hive4SpriteOn <= 1;
                    // Check if Bee Bullet Has Hit Hive 4
                    if ((sx == xBBullet) && (sy == yBBullet) && (sx>Hive4X-1) && (BBhithive4 == 0) && (H4dout > 0))
                        begin
                            BBhithive4 <= 1;
                            BHit4x <= xBBullet - 5;
                            BHit4y <= yBBullet - 13;
                        end
                end
            else
                Hive4SpriteOn <= 0;
        end
    else

    // insert bullet hole - Hive1
    if ((sx>640) && (sy>480) && (BBhithive1 == 1))
        begin
            if ((hzcounter1 == 0) && (vtcounter1 == 0))
                begin
                    H1address <= (BHit1x - Hive1X) + ((BHit1y - Hive1Y) * HiveWidth);
                    BHoleaddress <= 0;
                    hzcounter1 <= hzcounter1 + 1;
                end
            else
            if (hzcounter1 < 12)
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write1<=0;
                    else
                        begin
                            write1<=1;
                            data1 <= BHoledata[BHoleaddress];
                        end

                    BHoleaddress <= BHoleaddress + 1;
                    H1address <= H1address + 1;
                    hzcounter1 <= hzcounter1 + 1;
                end
            else
            if ((hzcounter1 == 12) && (vtcounter1 < 16))
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write1<=0;
                    else
                        begin
                            write1<=1;
                            data1 <= BHoledata[BHoleaddress];
                        end

                    H1address <= H1address + HiveWidth - 11;
                    vtcounter1 <= vtcounter1 + 1;
                    hzcounter1 <= 1;
                end

            if (vtcounter1 > 15)
                begin
                    BBhithive1 <= 0;
                    write1 <= 0;
                    hzcounter1 <= 0;
                    vtcounter1 <= 0;
                end
        end

    // insert bullet hole - Hive2
    if ((sx>640) && (sy>480) && (BBhithive2 == 1))
        begin
            if ((hzcounter2 == 0) && (vtcounter2 == 0))
                begin
                    H2address <= (BHit2x - Hive2X) + ((BHit2y - Hive2Y) * HiveWidth);
                    BHoleaddress <= 0;
                    hzcounter2 <= hzcounter2 + 1;
                end
            else
            if (hzcounter2 < 12)
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write2<=0;
                    else
                        begin
                            write2<=1;
                            data2 <= BHoledata[BHoleaddress];
                        end

                    BHoleaddress <= BHoleaddress + 1;
                    H2address <= H2address + 1;
                    hzcounter2 <= hzcounter2 + 1;
                end
            else
            if ((hzcounter2 == 12) && (vtcounter2 < 16))
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write2<=0;
                    else
                        begin
                            write2<=1;
                            data2 <= BHoledata[BHoleaddress];
                        end
                    H2address <= H2address + HiveWidth - 11;
                    vtcounter2 <= vtcounter2 + 1;
                    hzcounter2 <= 1;
                end

            if (vtcounter2 > 15)
                begin
                    BBhithive2 <= 0;
                    write2 <= 0;
                    hzcounter2 <= 0;
                    vtcounter2 <= 0;
                end
        end
    // insert bullet hole - Hive3
    if ((sx>640) && (sy>480) && (BBhithive3 == 1))
        begin
            if ((hzcounter3 == 0) && (vtcounter3 == 0))
                begin
                    H3address <= (BHit3x - Hive3X) + ((BHit3y - Hive3Y) * HiveWidth);
                    BHoleaddress <= 0;
                    hzcounter3 <= hzcounter3 + 1;
                end
            else
            if (hzcounter3 < 12)
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write3<=0;
                    else
                        begin
                            write3<=1;
                            data3 <= BHoledata[BHoleaddress];
                        end

                    BHoleaddress <= BHoleaddress + 1;
                    H3address <= H3address + 1;
                    hzcounter3 <= hzcounter3 + 1;
                end
            else
            if ((hzcounter3 == 12) && (vtcounter3 < 16))
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write3<=0;
                    else
                        begin
                            write3<=1;
                            data3 <= BHoledata[BHoleaddress];
                        end

                    H3address <= H3address + HiveWidth - 11;
                    vtcounter3 <= vtcounter3 + 1;
                    hzcounter3 <= 1;
                end

            if (vtcounter3 > 15)
                begin
                    BBhithive3 <= 0;
                    write3 <= 0;
                    hzcounter3 <= 0;
                    vtcounter3 <= 0;
                end
        end
    // insert bullet hole - Hive4
```

```
    if ((sx>640) && (sy>480) && (BBhithive4 == 1))
        begin
            if ((hzcounter4 == 0) && (vtcounter4 == 0))
                begin
                    H4address <= (BHit4x - Hive4X) + ((BHit4y - Hive4Y) * HiveWidth);
                    BHoleaddress <= 0;
                    hzcounter4 <= hzcounter4 + 1;
                end
            else
            if (hzcounter4 < 12)
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write4<=0;
                    else
                        begin
                            write4<=1;
                            data4 <= BHoledata[BHoleaddress];
                        end

                    BHoleaddress <= BHoleaddress + 1;
                    H4address <= H4address + 1;
                    hzcounter4 <= hzcounter4 + 1;
                end
            else
            if ((hzcounter4 == 12) && (vtcounter4 < 16))
                begin
                    if (BHoledata[BHoleaddress] > 0)
                        write4<=0;
                    else
                        begin
                            write4<=1;
                            data4 <= BHoledata[BHoleaddress];
                        end

                    H4address <= H4address + HiveWidth - 11;
                    vtcounter4 <= vtcounter4 + 1;
                    hzcounter4 <= 1;
                end

            if (vtcounter4 > 15)
                begin
                    BBhithive4 <= 0;
                    write4 <= 0;
                    hzcounter4 <= 0;
                    vtcounter4 <= 0;
                end
        end
        // Reset Hives at the end of each Level
        if ((sx>640) && (sy>480) && (ResetHives==1) && (lpcounter<2574))
                begin
                    write1 <= 1;
                    write2 <= 1;
                    write3 <= 1;
                    write4 <= 1;
                    H1address<=lpcounter;
                    H2address<=lpcounter;
                    H3address<=lpcounter;
                    H4address<=lpcounter;
                    data1 <= Hivememory_array[lpcounter];
                    data2 <= Hivememory_array[lpcounter];
                    data3 <= Hivememory_array[lpcounter];
                    data4 <= Hivememory_array[lpcounter];
```

```
                        lpcounter<=lpcounter+1;
                        if(lpcounter==2573)
                            lpcounter<=0;
                    end
end
endmodule
```

This is the code from the file “Hive1Ram.v”

```
//-----------------------------------------------
// Hive1Ram.v Module - Single Port RAM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup Hive1Ram Module
module Hive1Ram(
    input wire clk_pix,
    input wire [11:0] H1address,        // Address to read/write data from/to Hive1 (2^12 or 4095, need 66 x 39 = 2574)
    input wire [1:0] write1,            // 1 = write, 0 = read data
    input wire [7:0] data1,             // 8 bit pixel value to Hive1
    output reg [7:0] H1dout             // 8 bit pixel value from Hive1
    );

    (*RAM_STYLE="block"*) reg [7:0] H1memory_array [0:2573]; // 8 bit values for 2574 pixels of Hive1 (66 x 39)
    initial $readmemh("Hive1.mem", H1memory_array);

    always @ (posedge clk_pix)
        if (write1==1)
            H1memory_array[H1address] <= data1;
        else
            H1dout <= H1memory_array[H1address];
endmodule
```

This is the code from the file “Hive2Ram.v”

```
//-----------------------------------------------
// Hive2Ram.v Module - Single Port RAM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup Hive2Ram Module
module Hive2Ram(
    input wire clk_pix,
    input wire [11:0] H2address,        // Address to read/write data from/to Hive2 (2^12 or 4095, need 66 x 39 = 2574)
    input wire [1:0] write2,            // 1 = write, 0 = read data
    input wire [7:0] data2,             // 8 bit pixel value to Hive2
    output reg [7:0] H2dout             // 8 bit pixel value from Hive2
    );

    (*RAM_STYLE="block"*) reg [7:0] H2memory_array [0:2573]; // 8 bit values for 2574 pixels of Hive2 (66 x 39)
    initial $readmemh("Hive1.mem", H2memory_array);

    always @ (posedge clk_pix)
        if (write2==1)
            H2memory_array[H2address] <= data2;
        else
            H2dout <= H2memory_array[H2address];
endmodule
```

This is the code from the file “Hive3Ram.v”

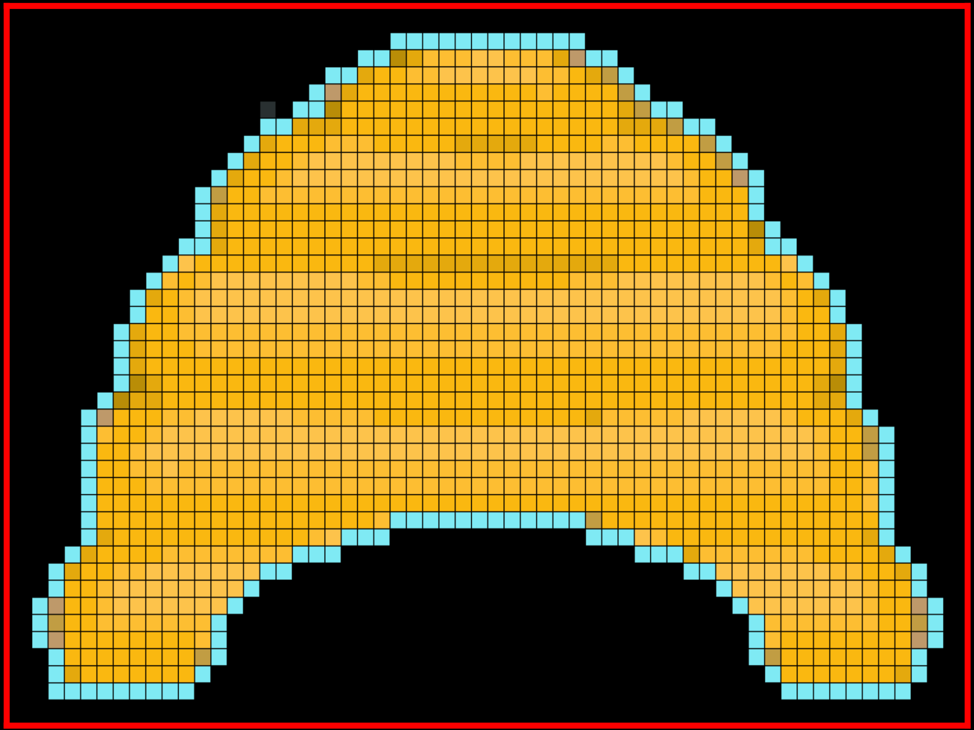
```
//-----------------------------------------------
// Hive3Ram.v Module - Single Port RAM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup Hive3Ram Module
module Hive3Ram(
    input wire clk_pix,
    input wire [11:0] H3address,        // Address to read/write data from/to Hive3 (2^12 or 4095, need 66 x 39 = 2574)
    input wire [1:0] write3,            // 1 = write, 0 = read data
    input wire [7:0] data3,             // 8 bit pixel value to Hive3
    output reg [7:0] H3dout             // 8 bit pixel value from Hive3
    );

    (*RAM_STYLE="block"*) reg [7:0] H3memory_array [0:2573]; // 8 bit values for 2574 pixels of Hive3 (66 x 39)
    initial $readmemh("Hive1.mem", H3memory_array);

    always @ (posedge clk_pix)
        if (write3==1)
            H3memory_array[H3address] <= data3;
        else
            H3dout <= H3memory_array[H3address];
endmodule
```

This is the code from the file “Hive4Ram.v”

```
//-----------------------------------------------
// Hive4Ram.v Module - Single Port RAM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup Hive4Ram Module
module Hive4Ram(
    input wire clk_pix,
    input wire [11:0] H4address,        // Address to read/write data from/to Hive4 (2^12 or 4095, need 66 x 39 = 2574)
    input wire [1:0] write4,            // 1 = write, 0 = read data
    input wire [7:0] data4,             // 8 bit pixel value to Hive4
    output reg [7:0] H4dout             // 8 bit pixel value from Hive4
    );
    (*RAM_STYLE="block"*) reg [7:0] H4memory_array [0:2573]; // 8 bit values for 2574 pixels of Hive4 (66 x 39)
    initial $readmemh("Hive1.mem", H4memory_array);

    always @ (posedge clk_pix)
        if (write4==1)
            H4memory_array[H4address] <= data4;
        else
            H4dout <= H4memory_array[H4address];
endmodule
```

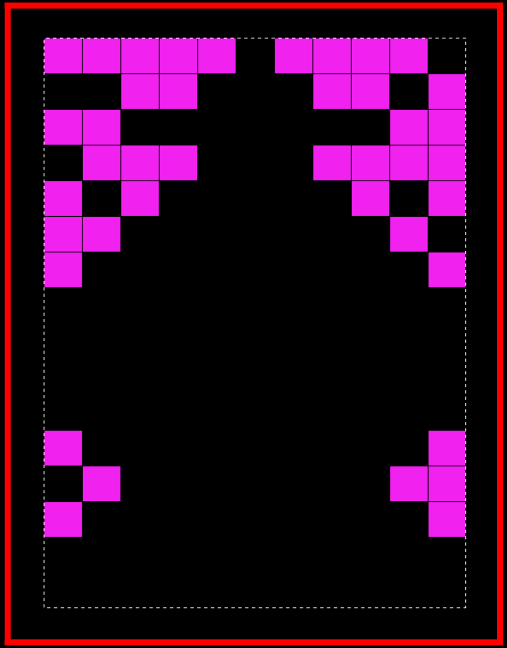
This is the data from the file "Hive1.mem" - Sprite Size 66 x 39 pixels

```
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 39 39 39 39 39 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 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 39 22 2E 31 31 31 33 33 31 31 31 2E 27 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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 2E 30
30 31 31 33 33 33 33 33 33 31 31 30 2E 28 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 00 00 00 00 00 00
00 00 00 00 00 00 00 00 39 27 2E 30 30 30 30 30 30 30 30 30 30 30 31 30 30 30 30 28 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 00 00 00 00 00 00 00 00 00 00 09 00 39 39 22 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 2E 28 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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 39 2E 2E 2E 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
2E 2E 2E 28 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 00 00 00 00 00 00 00 00 00 00 00 39 2E 30 30 30 31 31 31 30 30
30 30 30 2E 2E 2E 2E 2E 30 30 30 30 31 31 30 30 30 30 28 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 00
00 00 39 2E 30 30 31 33 33 33 33 33 33 33 33 33 31 31 31 31 33 33 33 33 33 33 33 33 33 31 30 30 28 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 00 39 2E 30 30 31 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 31 30 30 27 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 39 28 30 30 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 30 30 30 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 39 2E 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 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 39
2E 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 22 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 39 39 2E 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 2E 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 00 39 33 30 30 30 30 30 30 30 30 30 30 30 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 30 30
30 30 30 30 30 30 30 30 33 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 31 30 31 33 33 33 33 33 33 33 33 33 31 31 30 30 30
30 30 30 30 30 30 30 30 31 31 31 33 33 33 33 33 33 33 33 33 31 30 31 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 2E 30 31 33 33
33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 31 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 39 30 30 31 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 31 30 30 39 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 2E 30 30 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 2E 30 30 30 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 30 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 2E 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 39 22 2E 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 2E 22 39 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 22 2E 2E 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 2E 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 27 30 30 30 31 31 33 33 33 33 33 33 31 31 31 31 31 30 30 30 30 30 30
30 30 30 30 30 30 30 2E 31 31 31 31 31 33 33 33 33 33 33 31 30 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 31 30 30 31 33 33 33 33 33 33
33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 31 30 30 28 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 39 30 30 31 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 31 30 30 28 39 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 30 30 31 31 33 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 30 30 31 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 30 30 30 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 30 30 31 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 39 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 39 39 39 39 39 39 39 39 39 39 39 39 28 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 39 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 2E 30 30 30 30 30 30 30 30 30 30 30 30 31 33 39 39 39 00 00 00 00 00 00 00 00 00 00 00 00 39 39 39 33 31 30 30 30
30 30 30 30 30 30 30 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 2E 30 30 30 30 31 31 31 31 31 31 31 31 39 39 39 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 39 39 39 2E 31 31 31 31 31 31 31 30 30 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 00 39 2E 30 30 31 31 33 33 33 33 33 33 33 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 39 33 33 33 33 33 33 33 31 31 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 39 30
30 31 33 33 33 33 33 33 33 33 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 39 33 33 33 33 33 33 33 33 31 30 30 39 00
00 00 00 00 00 00 00 00 00 00 39 27 30 30 31 33 33 33 33 33 33 33 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
39 33 33 33 33 33 33 33 31 30 30 27 39 00 00 00 00 00 00 00 00 00 00 39 28 30 30 31 31 31 31 31 31 31 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 31 31 31 31 31 31 31 30 30 28 39 00 00 00 00 00 00 00 00 00 00 39 27 30 30 30 30 30 30 30 30 31 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 31 30 30 30 30 30 30 30 30 27 39 00 00 00 00 00 00 00 00 00 00 00 39 30 30
30 30 30 30 30 30 28 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 39 28 30 30 30 30 30 30 30 30 39 00 00
00 00 00 00 00 00 00 00 00 00 39 2E 30 30 30 30 30 30 30 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 00
00 39 30 30 30 30 30 30 30 2E 39 00 00 00 00 00 00 00 00 00 00 00 00 39 39 39 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 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
```



This is the data from the file “BHole.mem” - Sprite Size 11 x 16 pixels

```
16 16 16 16 16 00 16 16 16 16 00 00 00 16 16 00 00 00 16 16 00 16 16 16 00 00 00 00 00 00 16 16 00 16 16 16 00 00 00 16 16 16 16 16 00 16 00 00 00 00 00 16
00 16 16 16 00 00 00 00 00 00 00 16 00 16 00 00 00 00 00 00 00 00 16 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 00 00 00 00 00 00 00 00 00 00 00 00 00 16 00 00 00 00 00 00 00 00 16 00 16 00 00 00 00 00 00 00 00 16 16 16 00 00 00 00 00 00 00 00 00 16 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



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

```
//-----------------------------------------------
// ScoreSprite.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//-----------------------------------------------
`timescale 1ns / 1ps

// Setup ScoreSprite module
module ScoreSprite(
    input wire clk_pix,                 // 25.2MHz pixel clock
    input wire [9:0] sx,
    input wire [9:0] sy,
    input wire de,
    input wire [16:0] Score,            // Score value
    output reg [1:0] ScoreSpriteOn,     // 1=on, 0=off
    output wire [7:0] Scoredout,        // pixel value from Score.mem
    output reg [1:0] DigitsSpriteOn,    // 1=on, 0=off
    output wire [7:0] Digitsdout        // pixel value from Digit.mem
    );

    // instantiate ScoreRom
    reg [9:0] Scoreaddress;             // 2^10 or 1023, need 55 x 13 = 715
    ScoreRom ScoreVRom (
        .Scoreaddress(Scoreaddress),
        .clk_pix(clk_pix),
        .Scoredout(Scoredout)
    );

    // instantiate DigitsRom
    reg [10:0] Digitsaddress;           // 2^11 or 2047, need 110 x 13 = 1430
    DigitsRom DigitsVRom (
        .Digitsaddress(Digitsaddress),
        .clk_pix(clk_pix),
        .Digitsdout(Digitsdout)
    );

    // setup Score and Digits positions, sizes and variables
    localparam ScoreWidth = 55;         // Score width in pixels
    localparam ScoreHeight = 13;        // Score height in pixels
    localparam ScoreX = 6;              // Score X position
    localparam ScoreY = 6;              // Score Y position
    localparam scorevalx = 80;          // Score value X position
    localparam scposxl = 11;            // One digit - pixel width
    localparam scposyl = 13;            // One digit - pixel height
    reg [10:0] scoreyoffset=0;          // Y offset used in calculating Digitsaddress
    reg [16:0] digit5=0;                // 5th digit value of score
    reg [13:0] digit4=0;                // 4th digit value of score
    reg [9:0] digit3=0;                 // 3rd digit value of score
    reg [6:0] digit2=0;                 // 2nd digit value of score
    reg [3:0] digit1=0;                 // 1st digit value of score
    reg [6:0] t10=10;                   // used to calculate digit1-5
    reg [9:0] t100=100;                 // used to calculate digit1-5
    reg [13:0] t1000=1000;              // used to calculate digit1-5
    reg [16:0] t10000=10000;            // used to calculate digit1-5
    reg [7:0] counter = 0;              // used to calculate digit1-5

    always @ (posedge clk_pix)
    begin
        // if sx,sy are within the confines of the Score character, switch the Score On
        if(de)
            begin
                if((sx==ScoreX-2) && (sy==ScoreY))
                    begin
                        Scoreaddress <=0;
                        ScoreSpriteOn <=1;
                    end
                if((sx>ScoreX-2) && (sx<ScoreX+ScoreWidth-1) && (sy>ScoreY-1) && (sy<ScoreY+ScoreHeight))
                    begin
                        ScoreSpriteOn <=1;
                        Scoreaddress <= Scoreaddress + 1;
                    end
                else
                        ScoreSpriteOn <=0;
                if((sx==scorevalx-2) && (sy==ScoreY))
            begin
                scoreyoffset <= 110;
                DigitsSpriteOn <=1;
               digit5 <= Score / t10000;
               digit4<= (Score-(digit5*t10000)) / t1000;
               digit3<= (Score-((digit5*t10000)+(digit4*t1000))) / t100;
```

```
                digit2<= (Score-((digit5*t10000)+(digit4*t1000)+(digit3*t100))) / t10;
                digit1<= (Score-((digit5*t10000)+(digit4*t1000)+(digit3*t100)+(digit2*t10)));
                Digitsaddress <= digit5 * scposxl;
                counter<=0;
             end
             if((sx>scorevalx-2) && (sx<scorevalx+(scposxl*5)-1) && (sy>ScoreY-1) && (sy<ScoreY+scposyl))
                 begin
                         DigitsSpriteOn <=1;
                         counter <= counter + 1;
                         if(counter==(scposxl*1)-1)
                             Digitsaddress<=(digit4*scposxl)+(scoreyoffset-110);
                         else
                         if(counter==(scposxl*2)-1)
                             Digitsaddress<=(digit3*scposxl)+(scoreyoffset-110);
                         else
                         if(counter==(scposxl*3)-1)
                     Digitsaddress<=(digit2*scposxl)+(scoreyoffset-110);
                   else
                         if(counter==(scposxl*4)-1)
                             Digitsaddress<=(digit1*scposxl)+(scoreyoffset-110);
                         else
                             Digitsaddress <= Digitsaddress + 1;
                 end
                 else
                             DigitsSpriteOn <=0;
                             if(counter==scposxl*5)
                             begin
                                 scoreyoffset <= scoreyoffset + 110;
                                 Digitsaddress <= (digit5 * scposxl) + scoreyoffset;
                                 counter<=0;
                             end
             end
     end
endmodule
```

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

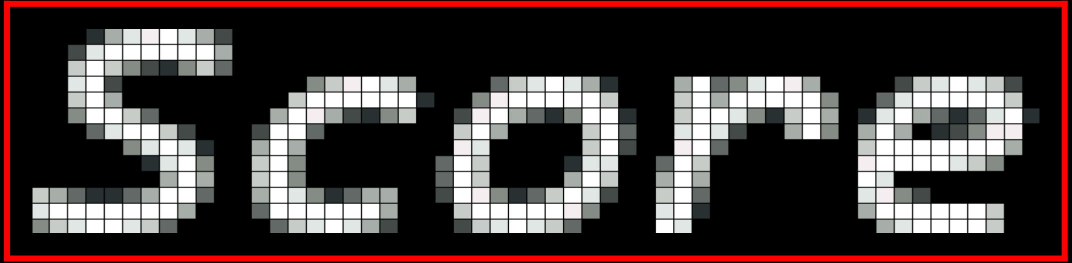
```
//--------------------------------------------------
// ScoreRom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//--------------------------------------------------
`timescale 1ns / 1ps

// Setup ScoreRom module
module ScoreRom(
    input wire [9:0] Scoreaddress,      // (2^10 or 1023, need 55 x 13 = 715
    input wire clk_pix,                 // pixel clock
    output reg [7:0] Scoredout          // (7:0) 8 bit pixel value from Score.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] Scorememory_array [0:714]; // 8 bit values for 715 pixels of Score (55 x 13)

    initial
    begin
        $readmemh("Score.mem", Scorememory_array);
    end

    always@(posedge clk_pix)
            Scoredout <= Scorememory_array[Scoreaddress];
endmodule
```

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

```
//----------------------------------------------
// DigitsRom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//----------------------------------------------
`timescale 1ns / 1ps

// Setup DigitsRom module
module DigitsRom(
    input wire [10:0] Digitsaddress,        // 2^11 or 2047, need 110 x 13 = 1430
    input wire clk_pix,                     // pixel clock
    output reg [7:0] Digitsdout             // 8 bit pixel value from Score.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] Digitsmemory_array [0:1429]; // 8 bit values for 1430 pixels of Digits (110 x 13)

    initial
    begin
        $readmemh("Digits.mem", Digitsmemory_array);
    end

    always@(posedge clk_pix)
            Digitsdout <= Digitsmemory_array[Digitsaddress];
endmodule
```

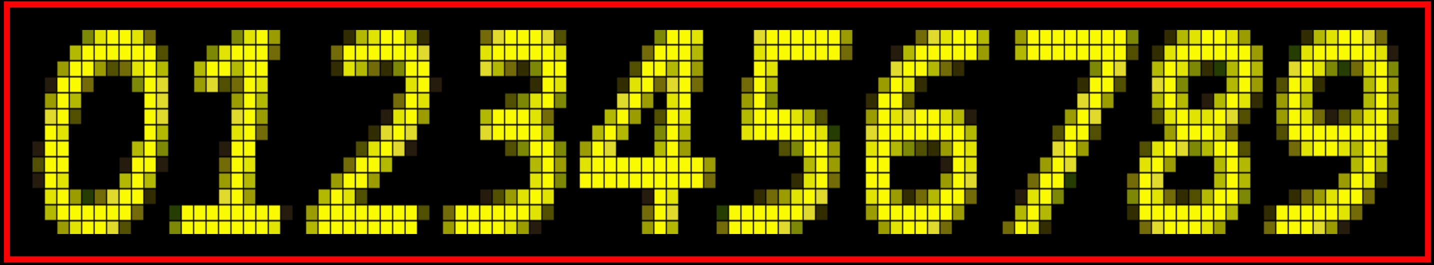
This is the data from the file “Score.mem” - Sprite Size 55 x 13 pixels

```
00 00 00 09 2C 3C 3E 3F 3C 2C 0B 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 00 00 00 00 00 00 00 00 00 00
00 00 00 00 0B 3C 3F 3F 3F 3F 3F 3F 2C 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 00 00 00 00 00 00 00 00
00 00 00 00 00 00 34 3F 34 1F 0B 09 1F 34 18 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 00 00 00 00 00 00
00 00 00 00 00 00 00 00 3C 3F 0B 00 00 00 00 00 00 00 00 00 00 1F 34 3E 3F 3C 2C 00 00 00 00 18 34 3C 3F 3C 2C 09 00 00 00 2C 3F 18 1F 3C 3F 3E 2C 00 00 00 00
0B 34 3C 3F 3C 34 0B 00 00 00 34 3F 2C 00 00 00 00 00 00 00 00 34 3F 3F 3F 3F 3F 3F 09 00 00 1F 3E 3F 3F 3F 3F 3F 34 00 00 00 34 3F 2C 3F 3F 3F 3F 3F 1F 00
00 18 3C 3F 3F 3F 3F 3F 34 00 00 00 1F 3F 3F 34 18 00 00 00 00 00 2C 3F 3E 2C 0B 09 1F 34 00 00 0B 3E 3F 2C 18 09 1F 3F 3F 09 00 00 34 3F 3F 3C 1F 09 34 3F
2C 00 09 3C 3F 2C 18 09 18 3C 3F 09 00 00 00 18 3C 3F 3F 34 0B 00 00 00 0B 3F 3F 18 00 00 00 00 00 00 00 34 3F 2C 00 00 00 00 34 3F 18 00 00 3C 3F 3C 0B 00 00
2C 3F 1F 00 2C 3F 2C 00 09 0B 1F 3E 3E 00 00 00 00 00 00 1F 3C 3F 3C 09 00 00 2C 3F 2C 00 00 00 00 00 00 00 00 3E 3C 00 00 00 00 00 34 3F 0B 00 00 3E 3F 18 00
00 00 00 00 00 00 34 3F 3F 3F 3F 3F 3F 3F 2C 00 00 00 00 00 00 00 09 3C 3F 1F 00 00 34 3F 1F 00 00 00 00 00 00 00 18 3F 34 00 00 00 00 09 3C 3C 00 00 18 3F 34
00 00 00 00 00 00 00 00 3E 3F 3F 3F 3F 3C 34 1F 00 00 00 00 00 00 00 00 00 2C 3F 2C 00 00 34 3F 1F 00 00 00 00 00 00 00 18 3F 34 00 00 00 00 2C 3F 34 00 00 2C
3F 2C 00 00 00 00 00 00 00 00 3F 3C 00 00 00 00 00 00 00 00 34 2C 18 09 09 18 2C 3F 3F 18 00 00 2C 3F 3C 1F 09 18 2C 2C 00 00 0B 3F 3E 1F 09 18 34 3F 3C 0B 00
00 34 3F 18 00 00 00 00 00 00 00 00 3C 3E 1F 0B 00 00 00 00 00 00 3C 3F 3F 3F 3F 3F 3F 3F 2C 00 00 00 18 3F 3F 3F 3F 3F 3F 2C 00 00 00 34 3F 3F 3F 3F 3F 3E 1F
00 00 00 3C 3F 09 00 00 00 00 00 00 00 00 2C 3F 3F 3F 3F 3F 3F 34 00 00 1F 34 3C 3F 3F 3C 34 18 00 00 00 00 00 18 34 3C 3F 3C 2C 0B 00 00 00 0B 34 3C 3F 3C 34
18 00 00 00 00 3F 3C 00 00 00 00 00 00 00 00 00 00 2C 3C 3F 3F 3F 3F 34 00 00
```



This is the data from the file “Digits.mem” - Sprite Size 110 x 13 pixels

```
00 00 00 00 1C 3A 3D 3D 3A 17 00 00 00 00 00 00 1C 3A 3D 23 00 00 00 00 00 07 2D 3A 3D 3D 2D 07 00 00 00 00 17 38 3D 3D 3D 38 0C 00 00 00 00 00 00 00 1C 3D 3D
3A 00 00 00 00 38 3D 3D 3D 3D 3D 3D 23 00 00 00 00 00 17 38 3A 3D 3D 2D 00 00 23 3D 3D 3D 3D 3D 3D 3D 3D 1C 00 00 07 2D 3A 3D 3D 3A 23 00 00 00 00 07 2D 3A 3D
3D 38 17 00 00 00 00 2D 3D 3D 3D 3D 3D 3D 0C 00 00 00 1C 3A 3D 3D 3D 17 00 00 00 00 17 3D 3D 3D 3D 3D 3D 38 00 00 00 00 3A 3D 3D 3D 3D 3D 3A 00 00 00 00 00 00
17 3D 3D 3D 2D 00 00 00 00 3A 3D 3D 3D 3D 3D 3D 17 00 00 00 03 2D 3D 3D 3D 3D 3D 23 00 00 2D 3D 3D 3D 3D 3D 3D 3D 3D 0C 00 07 3A 3D 3D 3D 3D 3D 3D 23 00 00 0A
3A 3D 3D 3D 3D 3D 3A 03 00 00 23 3D 3D 23 0C 17 3A 3D 2D 00 00 2D 3D 3D 2D 3D 3D 00 00 00 00 00 07 3A 2D 17 07 1C 3D 3D 00 00 00 00 38 2D 17 07 1C 3D 3D 00 00
00 00 00 0C 3D 3D 2D 3D 23 00 00 00 00 3D 3A 00 00 00 00 00 00 00 00 00 38 3D 3D 2D 17 07 00 00 00 00 00 00 00 00 00 00 1C 3D 38 00 00 2D 3D 3A 1C 07 0A 2D 3D
2D 00 00 38 3D 3A 1C 0A 17 3A 3D 1C 00 03 3D 3D 17 00 00 00 1C 3D 38 00 00 23 38 0C 17 3D 3A 00 00 00 00 00 00 00 00 00 00 00 3A 3D 03 00 00 00 00 00 00 00 07
3D 3A 00 00 00 00 07 3A 3D 17 38 3D 17 00 00 00 17 3D 2D 00 00 00 00 00 00 00 00 23 3D 3A 07 00 00 00 00 00 00 00 00 00 00 00 00 07 3D 3D 0C 00 00 38 3D 1C 00
00 00 2D 3D 23 00 0C 3D 3A 07 00 00 00 2D 3D 23 00 23 3D 2D 00 00 00 00 00 3D 38 00 00 00 00 00 23 3D 2D 00 00 00 00 00 00 00 00 00 00 17 3D 3A 00 00 00 00 00
00 0C 1C 3A 3D 1C 00 00 00 00 38 3D 23 00 3A 3D 00 00 00 00 23 3D 1C 00 00 00 00 00 00 00 07 3D 3D 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 38 3D 23 00 00 00
2D 3D 2D 00 03 2D 3D 3A 07 00 23 3D 2D 00 00 00 00 2D 3D 23 00 38 3D 0C 00 00 00 00 00 3D 38 00 00 00 00 00 38 3D 23 00 00 00 00 00 00 00 00 00 03 3A 3D 23 00
00 00 00 2D 3D 3D 3D 3A 17 00 00 00 00 2D 3D 23 00 0C 3D 3A 00 00 00 00 2D 3D 3D 3D 3A 2D 0C 00 00 00 23 3D 3A 38 3D 3D 3A 2D 03 00 00 00 00 00 00 00 23 3D 38
00 00 00 00 0C 3A 3D 3A 3A 3D 38 0C 00 00 23 3D 3A 17 03 0C 23 3A 3D 23 00 3D 3A 00 00 00 00 00 00 3D 2D 00 00 00 00 00 3A 3D 17 00 00 00 00 00 00 00 00 07 38
3D 38 00 00 00 00 00 38 3D 3D 3D 3D 2D 00 00 00 2D 3D 2D 00 00 1C 3D 2D 00 00 00 00 3A 3D 3D 3D 3D 3D 3A 0A 00 00 38 3D 3D 3D 3D 3D 3D 3D 2D 00 00 00 00 00 00
0C 3D 3D 0C 00 00 00 00 00 23 3D 3D 3D 3A 17 00 00 00 0C 3D 3D 3D 3D 3D 3D 3D 3D 0C 03 3D 3D 00 00 00 00 00 2D 3D 1C 00 00 00 00 03 3D 3D 00 00 00 00 00 00 00
00 0C 3A 3D 38 03 00 00 00 00 00 00 00 0C 1C 3D 3D 1C 00 23 3D 38 00 00 00 2D 3D 23 00 00 00 00 00 00 00 0C 1C 3D 3D 23 00 00 3A 3D 2D 1C 0C 07 23 3D 3A 00 00
00 00 00 00 38 3D 23 00 00 00 00 0C 3A 3D 38 1C 2D 3D 3D 0C 00 00 00 17 3A 3D 3D 3A 2D 3D 3A 00 0C 3D 3D 00 00 00 00 03 3D 3D 03 00 00 00 00 17 3D 3A 00 00 00
00 00 00 00 17 3D 3D 2D 00 00 00 00 00 00 00 00 00 00 00 2D 3D 23 00 3A 3D 3D 3D 3D 3D 3D 3D 3D 3D 23 00 00 00 00 00 00 00 2D 3D 23 00 00 3D 3D 00 00 00 00 03
3D 3A 00 00 00 00 00 23 3D 38 00 00 00 00 00 3A 3D 23 00 00 00 2D 3D 2D 00 00 00 00 00 00 00 00 2D 3D 2D 00 03 3D 3A 00 00 00 00 2D 3D 2D 00 00 00 00 00 23 3D
2D 00 00 00 00 00 00 23 3D 3D 23 00 00 00 00 00 00 00 00 00 00 00 00 3A 3D 1C 00 3D 3D 3D 3D 3D 3D 3D 3D 3D 17 00 00 00 00 00 00 07 3A 3D 17 00 00 3D 3D 00
00 00 00 23 3D 38 00 00 00 00 17 3D 3D 0A 00 00 00 00 17 3D 38 00 00 00 00 2D 3D 2D 00 00 00 00 00 00 00 23 3D 3A 07 00 00 3A 3D 23 0A 17 38 3D 3D 07 00 00 00
00 00 38 3D 23 00 00 00 00 00 2D 3D 3A 0C 00 00 00 00 00 00 00 00 00 03 0C 23 3A 3D 3A 00 00 00 00 00 00 00 03 3D 3A 00 00 00 00 00 00 07 17 23 3A 3D 38 00 00
00 3A 3D 23 07 17 2D 3D 3D 17 00 00 00 03 3A 3D 1C 00 00 00 00 00 1C 3D 3A 17 07 0C 2D 3D 3D 1C 00 00 00 07 17 23 3A 3D 3D 17 00 00 00 2D 3D 3D 3D 3D 3D 3D 17
00 00 0A 3D 3D 3D 3D 3D 3D 3D 3D 03 00 23 3D 3D 3D 3D 3D 3D 3D 3D 0C 00 17 3D 3D 3D 3D 3D 3D 3A 0C 00 00 00 00 00 00 00 17 3D 38 00 00 00 0A 3D 3D 3D 3D 3D 3D
38 07 00 00 00 23 3D 3D 3D 3D 3D 3D 23 00 00 00 00 2D 3D 2D 00 00 00 00 00 00 07 3D 3D 3D 3D 3D 3D 3D 2D 00 00 07 3D 3D 3D 3D 3D 3A 17 00 00 00 00 00 23 3A 3D
3D 38 0C 00 00 00 1C 3D 3D 3D 3D 3D 3D 3D 3A 00 00 2D 3D 3D 3D 3D 3D 3D 3D 3D 00 00 23 3D 3D 3D 3D 3A 23 03 00 00 00 00 00 00 00 00 23 3D 2D 00 00 00 17 3D 3D
3D 3D 38 1C 00 00 00 00 00 00 23 3A 3D 3D 38 17 00 00 00 00 17 3D 3A 07 00 00 00 00 00 00 17 38 3D 3D 3D 3A 23 00 00 00 17 3D 3D 3D 38 23 03 00 00 00 00
```



This is the code from the file “Honeycomb1Sprite.v”

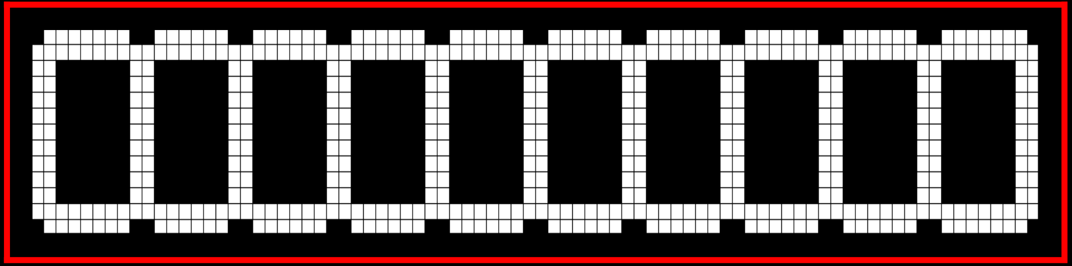
```
//--------------------------------------------
// Honeycomb1Sprite.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//--------------------------------------------
`timescale 1ns / 1ps

// Setup Honeycomb1Sprite module
module Honeycomb1Sprite(
    input wire clk_pix,                        // 25.2MHz pixel clock
    input wire [9:0] sx,
    input wire [9:0] sy,
    input wire de,
    output reg [1:0] Honeycomb1SpriteOn,       // 1=on, 0=off
    output wire [7:0] Honeycomb1dout           // pixel value from Honeycomb1.mem
    );

    // instantiate Honeycomb1Rom
    reg [10:0] Honeycomb1address;              // 2^11 or 2047, need 82 x 13 = 1066
    Honeycomb1Rom Honeycomb1VRom (
        .Honeycomb1address(Honeycomb1address),
        .clk_pix(clk_pix),
        .Honeycomb1dout(Honeycomb1dout)
    );

    // setup Honeycomb1 character positions and sizes
    localparam Honeycomb1Width = 82;           // Score width in pixels
    localparam Honeycomb1Height = 13;          // Score height in pixels
    localparam Honeycomb1X = 279;              // Honeycomb1 X position
    localparam Honeycomb1Y = 6;                // Honeycomb1 Y position

    always @ (posedge clk_pix)
    begin
        // if sx,sy are within the confines of the Score character, switch the Score On
        if(de)
            begin
                if((sx==Honeycomb1X-2) && (sy==Honeycomb1Y))
                    begin
                        Honeycomb1address <=0;
                        Honeycomb1SpriteOn <=1;
                    end
                if((sx>Honeycomb1X-2) && (sx<Honeycomb1X+(Honeycomb1Width*1)-1) && (sy>Honeycomb1Y-1) && (sy<Honeycomb1Y+Honeycomb1Height))
                    begin
                        Honeycomb1SpriteOn <=1;
                            Honeycomb1address <= Honeycomb1address + 1;
                    end
                else
                        Honeycomb1SpriteOn <=0;
            end
    end
endmodule
```

This is the code from the file “Honeycomb1Rom.v”

```
//---------------------------------------------------
// Honeycomb1Rom.v Module
// Single Port ROM
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//---------------------------------------------------
`timescale 1ns / 1ps

// Setup Honeycomb1Rom module
module Honeycomb1Rom(
    input wire [10:0] Honeycomb1address,    // (2^11 or 2047, need 82 x 13 = 1066
    input wire clk_pix,                     // pixel clock
    output reg [7:0] Honeycomb1dout         // 8 bit pixel value from Honeycomb1.mem
    );

    (*ROM_STYLE="block"*) reg [7:0] Honeycomb1memory_array [0:1065]; // 8 bit values for 1066 pixels of Honeycomb1

    initial
    begin
        $readmemh("Honeycomb1.mem", Honeycomb1memory_array);
    end

    always@(posedge clk_pix)
            Honeycomb1dout <= Honeycomb1memory_array[Honeycomb1address];
endmodule
```

This is the data from the file “Honeycomb1.mem” - Sprite Size 82 x 13 pixels

```
00 3F 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F
3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 3F 00 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F
3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F
3F 3F 3F 3F 3F 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00
3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00
00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00
3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00
00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00
00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00
00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00
00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00 00 00 00 00 3F
3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00
00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F
3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00
00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F
3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00
00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F 3F 00 00 00 00 00 00 3F
3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F
3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 00 3F 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F
3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F 3F 3F 3F 3F 00 00 3F 3F
3F 3F 3F 3F 3F 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 29 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 F1 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 81 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 E0 DA 2C 7F EA F4 E0 E4 00 E6 E6 85 E1 E7 E5 FB FB 00 F4 EE F1 FF FF FF
```

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

```
##-----------------------------------
## Constraints Module
## Digilent Basys 3
## BeeInvaders Tutorial_6
## Onboard clock 100MHz
## VGA Resolution: 640x480 @ 60Hz
## Pixel Clock 25.2MHz
##-----------------------------------

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk_100m]
   set_property IOSTANDARD LVCMOS33 [get_ports clk_100m]
   create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk_100m]


## Buttons
set_property PACKAGE_PIN U18 [get_ports btn_rst_n]
   set_property 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]
set_property PACKAGE_PIN T18 [get_ports btnF]
   set_property IOSTANDARD LVCMOS33 [get_ports btnF]

## 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_5
## 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 C9 IOSTANDARD LVCMOS33} [get_ports {btnF}]
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]}]
```

(E) EXPLANATION OF THE VERILOG CODE USED

01 Top.v module additional code

```
// Instantiate BBulletSprite
BBulletSprite BBulletDisplay (
   ..
   .BBhitAlien1(BBhitAlien1),
   .BBhitAlien2(BBhitAlien2),
   .BBhitAlien3(BBhitAlien3),
```

BBhitAlien1 – 3 values are passed from AlienSprites.v to BBulletSprite.v

```
// Instantiate AlienSprites
   wire [1:0] LevelSpriteOn;           // Level Indicator
   wire [7:0] Ldout;                   // pixel value for Level
   wire [1:0] BBhitAlien1;             // Bee Bullet Hit Alien1 (0 = No, 1 = Yes)
   wire [1:0] BBhitAlien2;             // Bee Bullet Hit Alien2 (0 = No, 1 = Yes)
   wire [1:0] BBhitAlien3;             // Bee Bullet Hit Alien3 (0 = No, 1 = Yes)
   wire [16:0] Score;                  // Score value
   AlienSprites AlienDisplay (
      .xBBullet(xBBullet),             // x coordinate for Bee Bullet
      .yBBullet(yBBullet),             // y coordinate for Bee Bullet
      .BBhitAlien1(BBhitAlien1),
      .BBhitAlien2(BBhitAlien2),
      .BBhitAlien3(BBhitAlien3),
      .LevelSpriteOn(LevelSpriteOn),
      .Ldout(Ldout),
      .Score(Score),
      .ResetHives(ResetHives)
```

xBBullet(xBBullet), yBBullet(yBBullet), BBhitAlien1(BBhitAlien1), BBhitAlien2(BBhitAlien2), BBhitAlien3(BBhitAlien3), LevelSpriteOn(LevelSpriteOn), Ldout(Ldout), Score(Score), ResetHives(ResetHives) are all passed via AlienSprite.v

```
HiveSprites HDisplay (
        ..
        ..
        .ResetHives(ResetHives),
        ..
        .lpcounter(lpcounter)
   );
```

ResetHives(ResetHives), lpcounter(lpcounter) are all passed via HiveSprites.v

```
// Instantiate Score
    wire [1:0] ScoreSpriteOn;            // 1=on, 0=off
    wire [7:0] Scoredout;                // pixel value from Score.mem
    wire [1:0] DigitsSpriteOn;           // 1=on, 0=off
    wire [7:0] Digitsdout;               // pixel value from Digits.mem
    wire [10:0] Digitsaddress;           // 11^10 or 2047, need 110 x 13 = 1430
    ScoreSprite ScoreDisplay (
        .clk_pix(clk_pix),
        .sx(sx),
        .sy(sy),
        .de(de),
        .Score(Score),
        .ScoreSpriteOn(ScoreSpriteOn),
        .Scoredout(Scoredout),
        .DigitsSpriteOn(DigitsSpriteOn),
        .Digitsdout(Digitsdout)
);
```

This is a new module to display the Score at the top of the screen

```
// Instantiate Honeycomb1
wire [1:0] Honeycomb1SpriteOn;              // 1=on, 0=off
wire [7:0] Honeycomb1dout;                  // pixel value from Honeycomb1.mem
Honeycomb1Sprite Honeycomb1Display (
    .clk_pix(clk_pix),
   .sx(sx),
   .sy(sy),
   .de(de),
   .Honeycomb1SpriteOn(Honeycomb1SpriteOn),
   .Honeycomb1dout(Honeycomb1dout)
);
```

This is a new module to display the Honeycomb Level Indicator at the top of the screen

```
if (ScoreSpriteOn==1)
    begin
        vga_r <= (palette[(Scoredout*3)])>>4;           // RED bits(7:4) from colour palette
        vga_g <= (palette[(Scoredout*3)+1])>>4;         // GREEN bits(7:4) from colour palette
        vga_b <= (palette[(Scoredout*3)+2])>>4;         // BLUE bits(7:4) from colour palette
    end
else
if (DigitsSpriteOn==1)
   begin
        vga_r <= (palette[(Digitsdout*3)])>>4;          // RED bits(7:4) from colour palette
        vga_g <= (palette[(Digitsdout*3)+1])>>4;        // GREEN bits(7:4) from colour palette
        vga_b <= (palette[(Digitsdout*3)+2])>>4;        // BLUE bits(7:4) from colour palette
    end
else
if (LevelSpriteOn==1)
    begin
        vga_r <= (palette[(Ldout*3)])>>4;               // RED bits(7:4) from colour palette
        vga_g <= (palette[(Ldout*3)+1])>>4;             // GREEN bits(7:4) from colour palette
        vga_b <= (palette[(Ldout*3)+2])>>4;             // BLUE bits(7:4) from colour palette
    end
```

```
else
if (Honeycomb1SpriteOn==1)
    begin
        vga_r <= (palette[(Honeycomb1dout*3)])>>4;          // RED bits(7:4) from colour palette
        vga_g <= (palette[(Honeycomb1dout*3)+1])>>4;        // GREEN bits(7:4) from colour palette
        vga_b <= (palette[(Honeycomb1dout*3)+2])>>4;        // BLUE bits(7:4) from colour palette
    end
```

If ScoreSprite, DigtisSprite, LevelSprite or Honeycomb1Sprite are switched on, display them

02 BBulletSprite.v module additional code

```
// Setup BBulletSprite module
module BBulletSprite(
   ..
   input wire [1:0] BBhitAlien1,                    // 1 = bullet hit Alien1, 0 = no hit
   input wire [1:0] BBhitAlien2,                    // 1 = bullet hit Alien2, 0 = no hit
   input wire [1:0] BBhitAlien3,                    // 1 = bullet hit Alien3, 0 = no hit
```

BBhitAlien1 – 3 values have been passed from AlienSprites.v, set to 1 if the Bee Bullet has hit an Alien

```
// if fire button pressed, move the Bee Bullet up the screen
if ((sx==640) && (sy==480))                         // check for movement once every frame
    ..
    if ((BBulletstate == 1))
        begin
            yBBullet<=yBBullet-2;                   // move bullet up the screen
            if ((BBhithive1 == 1) || (BBhithive2 == 1) || (BBhithive3 == 1) || (BBhithive4 == 1) || (BBhitAlien1 == 1) || (BBhitAlien2 == 1) || (BBhitAlien3 == 1))
            // Check if Bee Bullet has hit hive1-4 and Alien1-3
                    begin
                        BBulletstate<=2;            // stop the bullet
                        yBBullet<=425;              // bullet y start position
                        xBBullet<=0;
                    end
```

yBBullet<=yBBullet-2; this moves the bullet up the screen more faster (was -1)

A check to see if BBhithive1 – 4 or BBhitAlien1 - 3 are equal 1, if so the Bee Bullet is stopped and reset

03 AlienSprites.v module additional code

```
// Setup AlienSprites Module
module AlienSprites(
    input wire [9:0] xBBullet,                   // x coordinate for Bee Bullet
    input wire [9:0] yBBullet,                   // y coordinate for Bee Bullet
    input wire [11:0] lpcounter,                 // Loop counter used in resetting hive graphics
    output reg [1:0] BBhitAlien1,                // 1=hit, 0=no hit
    output reg [1:0] BBhitAlien2,                // 1=hit, 0=no hit
    output reg [1:0] BBhitAlien3,                // 1=hit, 0=no hit
    output reg [1:0] LevelSpriteOn,              // 1=on, 0=off
    output reg [7:0] Ldout,                      // 8 bit pixel value for Level
    output reg [16:0] Score,                     // 8 bit pixel value from Score.mem
    output reg [1:0] ResetHives                  // set to 1 if level completed in order to reset hive graphics
);
```

xBBullet and yBBullet are passed from BBulletSprite.v to AlienSprites.v
lpcounter is passed from HiveSprites.v to AlienSprites.v
BBhitAlien1 – 3 are passed from AlienSprites.v to BBulletSprite.v
LevelSpriteOn and Ldout are passed from AlienSprites.v to Top.v
Score is passed from AlienSprites.v to ScoreSprite.v
ResetHives is passed from AlienSprites.v to HiveSprites.v

```
reg [3:0] delloop = 8;                           // counter end value for delaliens
reg [3:0] M = 4;                                 // move Aliens by this number of pixels

reg [0:10] alienc1=11'b11111111111;              //----------------------------------
reg [0:10] alienc2=11'b11111111111;              //
reg [0:10] alienc3=11'b11111111111;              // Set pattern of Aliens 11 x 5 = 55
reg [0:10] alienc4=11'b11111111111;              //
reg [0:10] alienc5=11'b11111111111;              //----------------------------------
```

delloop has been increased to 8 to slow the Aliens down more
M has been used to determine how many pixels the Aliens are moved
alienc1 – 5 state the pattern of the Aliens (0=off, 1=on), note how the register has been defined [0:10] which declares the registers as [MSB:LSB] or [Left most Aliens:Right most Aliens]

```
 // setup Level Indicator positions and sizes
localparam LevelWidth = 6;                   // Level Indicator width in pixels
localparam LevelHeight = 9;                  // Level Indicator height in pixels
reg [9:0] LevelX = 281;                      // X position for Honeycomb1 Level Indicator
reg [8:0] LevelY = 8;                        // Y position for Honeycomb1 Level Indicator
reg [3:0] Level=0;                           // Level Number
reg [5:0] AlienQ=55;                         // Alien quantity in the wave
reg [9:0] LcounterW=0;                       // Counter to check if Honeycomb1 Level Indicator width reached
reg [9:0] LcounterH=0;                       // Counter to check if Honeycomb1 Level Indicator height reached
```

This declares the size, screen x,y position and variables used in displaying the level indicator at the top (middle) of the screen

```
 // Initially set ResetHives to 0
     if(Score==0)
         ResetHives<=0;
 // 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-1) && (sy>A1Y+AoY-1) && (sy<A1Y+A1Height+AoY))
     begin
         A1address <= A1address + 1;
         AcounterW <= AcounterW + 1;
```

```
        Alien1SpriteOn <=1;
        if(alienc1[AoX/40]==0)
            Alien1SpriteOn <=0;
        if (AcounterW==A1Width-1)
                begin
                    AcounterW <= 0;
                    if(AcolCount>1)
                        AoX <= AoX + 40;
                    if(AoX<(AcolCount-1)*40)
                        A1address <= A1address - (A1Width-1);
                    if(AoX==(AcolCount-1)*40)
                        AoX<=0;
                end
        // Check if Bee Bullet Has Hit Alien1
        if ((sx == xBBullet) && (sy == yBBullet) && (sx>A1X+AoX-1) && (BBhitAlien1 == 0) && (A1dout > 0) && (alienc1[AoX/40]==1))
            begin
                BBhitAlien1 <= 1;
                alienc1[AoX/40]<=0;
            end
    end
    else
        Alien1SpriteOn <=0;
```

First the ResetHives variable is set to 0, the remaining code replaces the previous code check if sx,sy are within the confines of the Alien characters for Alien1 and is very similar to the code for Alien2 and Alien3

The additional code being:

```
if(alienc1[AoX/40]==0)
Alien1SpriteOn <=0;
```

This calculates the current sprite number, if the bit pattern equals 0 then the sprite is switched off

```
if(AcolCount>1)
    AoX <= AoX + 40;
if(AoX<(AcolCount-1)*40)
    A1address <= A1address - (A1Width-1);
if(AoX==(AcolCount-1)*40)
    AoX<=0;
```

This moves AoX to the next horizontal sprite position if AcolCount (the number of alien columns) is greater than 1

A1address is then reset to the start horizontal position of the sprite if AoX is less than the last alien position (far right). If AoX equals the last sprite position reset AoX to 0

```
// Check if Bee Bullet Has Hit Alien1
if ((sx == xBBullet) && (sy == yBBullet) && (sx>A1X+AoX-1) && (BBhitAlien1 == 0) && (A1dout > 0) && (alienc1[AoX/40]==1))
    begin
        BBhitAlien1 <= 1;
        alienc1[AoX/40]<=0;
    end
```

If sx,sy are equal to the Bee Bullet x,y position and the alien sprite is switched on, then set the state of BBhitAlien1 (this value is passed to BBulletSprite.v to switch the Bee Bullet sprite off) to 1 and switch the alien sprite off

```
//Level Indicator
if(Level>0)
    begin
        if(Level>9)
            Level<=0;
        if (sx==LevelX-2 && sy==LevelY)
            begin
                Ldout <= 26;
                LevelSpriteOn <=1;
                LcounterW<=0;
                LcounterH<=0;
            end
        if ((sx>LevelX-2) && (sx<LevelX+LevelWidth) && (sy>LevelY-1) && (sy<LevelY+LevelHeight))
            begin
                Ldout <= 26;
                LcounterW <= LcounterW + 1;
                LevelSpriteOn <=1;
                if(LcounterW==LevelWidth-1)
                    begin
                        if(LevelX<((Level-1)*8)+281)
                            begin
                                LcounterW <= 0;
                                LevelX <= LevelX + 8;
                            end
                        else
                            begin
                                LevelX = 281;
                                LcounterH <= LcounterH + 1;
                                LcounterW <= 0;
                                if(LcounterH==LevelHeight-1)
                                    begin
                                        LevelX = 281;
                                        LcounterH <= 0;
                                        LcounterW <= 0;
                                    end
                            end
                    end
            end
            else
                LevelSpriteOn <=0;
    end
end
```

This section shows how many levels have been completed. If Level equals 0 (this will be the case at the start of a game) all the routine does is to switch the Level Sprite off. Once a level has been completed the routine fills the next empty Honeycomb level box with the colour Red (colour index number 26 in our colour palette)

The routine currently resets Level to 0 if it is greater than 9 (this will change as the code for the games progresses, it will eventually signify that you have beat the Alien Invaders)

If Level is greater than 0 and sx,sy are in the confines of the completed level number (LevelX initially equals the x position inside the first level indicator box):

Ldout (passed to Top.v where it is used to extract the RGB colours) is set to Red (26);

LevelSpriteOn is switched on (1)

LcounterW and LcounterH are initialised to 0. These are counters used to detect when the width (LevelWidth) and height (LevelHeight) of the empty space inside each level indicator has been reached (6 x 9 pixels)

LcounterW is incremented until it equals LevelWidth-1. At this point, as long as LevelX is less than the x position of the current level indicator (Level-1) multiplied by 8 (the distance between each indicator) plus 281 (the original start x position of the level indicator) , LcounterW is rest to 0 and LevelX is incremented by 8

If this is not the case, LevelX is rest to its original x position of 281, LcounterH is incremented by 1 and LcounterW is rest to 0. At this point, if LcounterH equals LevelHeight-1 then LevelX is reset to 281 and LcounterH & LcounterW are rest to 0

```
// check if a column of Aliens have all been shot
if (sx>640 && sy>480)
    begin
        if((alienc1[AcolCount-1]==0) && (alienc2[AcolCount-1]==0) && (alienc3[AcolCount-1]==0) && (alienc4[AcolCount-1]==0) && (alienc5[AcolCount-1]==0)
        && (AcolCount>1))
            AcolCount<=AcolCount-1;
        else
        if((alienc1[0]==0) && (alienc2[0]==0) && (alienc3[0]==0) && (alienc4[0]==0) && (alienc5[0]==0) && (AcolCount>1))
            begin
                AcolCount<=AcolCount-1;
                alienc1<=alienc1<<1;
                alienc2<=alienc2<<1;
                alienc3<=alienc3<<1;
                alienc4<=alienc4<<1;
                alienc5<=alienc5<<1;
                A1X<=A1X+40;
                A2X<=A2X+40;
                A3X<=A3X+40;
            end
    end
```

This section of the code checks if a column of Aliens have all been shot

The last column (far right column) is checked first:

If alienc1–5 [AcolCount-1] (right most column of aliens) all equal 0 and AcolCount is greater than 0:

1. Decrement AcolCount by 1

The first column (far left column) is not as straight forward:

If alienc1–5 [0] (the left most column of aliens) all equal 0 and AcolCount is greater than 0:

1. Decrement AcolCount by 1

2. Shift the bit patterns of alienc1–5 left by 1 bit

Example	If alienc1 left most alien has been shot the bit pattern equals	0 1 1 1 1 1 1 1 1 1 1
	After shifting alienc1 bits left by 1 the pattern will be	1 1 1 1 1 1 1 1 1 1
	And a 0 will be shifted into the far right of alienc1 bit pattern	1 1 1 1 1 1 1 1 1 1 0

3. Increment A1X, A2X and A3X by 40. This moves the x start position of all aliens right by one alien position. We do this because shifting the bit patterns left by 1 has the effect of moving all visible aliens left by one alien position

```
// slow down the alien movement / move aliens left or right
if (sx==640 && sy==480)
    begin
        delaliens<=delaliens+1;
        if (delaliens>delloop)
            begin
                delaliens<=0;
                if (Adir==2)
                    begin
                        A1X<=A1X+M;
                        A2X<=A2X+M;
                        A3X<=A3X+M;
                        if (A1X+A1Width+((AcolCount-1)*40)>(640-(M*2)-1))
                            Adir<=1;
                    end
                else
                if (Adir==1)
                    begin
                        A1X<=A1X-M;
                        A2X<=A2X-M;
                        A3X<=A3X-M;
                        if (A1X<(M*2)+1)
                            Adir<=2;
                    end
            end
```

This section of the code slows down the alien movement and moves the aliens left or right

delaliens is initially set to 0 and is incremented by 1 every frame cycle. If it is greater than delloop (initially set to 8) then delaliens is rest to 0. At this point, we need to determine if the aliens are travelling left or right on the screen

If Adir equals 2 (aliens moving right) then A1X, A2X and A3X are incremented by the value of M (initially set to 4). Lastly, the x position of the right most alien is checked to see if it has reached the far right hand side of the screen. If this is the case, the direction of the aliens are changed

If Adir equals 1 (aliens moving left) then A1X, A2X and A3X are decremented by the value of M (initially set to 4). Lastly, the x position of the left most alien is checked to see if it has reached the far left hand side of the screen. If this is the case, the direction of the aliens are changed

```
// If Alien has been shot increase the Score / If all Aliens have been shot reset the wave and Hives
if(BBhitAlien1==1)
    begin
        BBhitAlien1<=0;
        Score<=Score+30;
        AlienQ<=AlienQ-1;
    end
if(BBhitAlien2==1)
    begin
        BBhitAlien2<=0;
        Score<=Score+20;
        AlienQ<=AlienQ-1;
    end
if(BBhitAlien3==1)
    begin
        BBhitAlien3<=0;
        Score<=Score+10;
        AlienQ<=AlienQ-1;
    end
if(Score>99999)
    begin
        Score<=Score-100000;
    end
if(AlienQ<1 && Score>0)
    begin
        Level<=Level+1;
        AcolCount<=11;
        AlienQ<=55;
        delloop<=8;
        A1X<=135;
        A2X<=135;
        A3X<=135;
        M<=4;
        alienc1<=11'b11111111111;
        alienc2<=11'b11111111111;
        alienc3<=11'b11111111111;
        alienc4<=11'b11111111111;
        alienc5<=11'b11111111111;
```

```
            ResetHives<=1;
        end
        if(AlienQ>0)
            ResetHives<=0;
```

This section of code checks if an alien has been shot and increases the score accordingly. It also checks if all aliens have been shot and if so, sets up a new wave by resetting the associated variables

If BBhitAlien1-3 equal 1: set BBhitAlien1-3 to 0 (no alien hit), increment Score accordingly and decrement AlienQ by 1

If Score is greater than 99999 (the maximum score value is a 5 digit number), then 100000 is deducted from Score

The code then checks if AlienQ equals 0 (no aliens left to shoot) and if Score is greater than 0. If this is true then Level is incremented by 1 and the associated variables are reset to their original values, in order to start a new wave of aliens

ResetHives is set to 1 in order that the HiveSprites.v routine can reset the Hives at the end of each level

ResetHives is set back to 0 once it has been done and when AlienQ is greater then 0

```
        // Alien speed up stages
        if(AlienQ==50)
            delloop<=7;
        else
        if(AlienQ==43)
            delloop<=6;
        else
        if(AlienQ==36)
            delloop<=5;
        else
        if(AlienQ==29)
            delloop<=4;
        else
        if(AlienQ==22)
            delloop<=3;
        else
        if(AlienQ==15)
            delloop<=2;
        else
        if(AlienQ==8)
            delloop<=1;
        else
        if(AlienQ==1)
            M<=6;
    end
```

The final section of code sets delloop and M to a value which speeds up the aliens movement, depending on how many aliens remain on the screen

04 HiveSprites.v module additional code

```
module HiveSprites(
    ..
    input wire [1:0] ResetHives,    // set to 1 if level completed in order to reset hive graphics
    ..
    output reg [11:0] lpcounter     // Loop counter used in resetting hive graphics
    );
```

This section of code retrieves the value of ResetHives from AlienSprites.v, used to check if a level has been completed

lpcounter is outputted to AlienSprites.v and used in the module to reset ResetHives to 0

```
// Load Hive1.mem into register, used to reset the hive graphics
(*RAM_STYLE="block"*) reg [7:0] Hivememory_array [0:2573];          // 8 bit values for 2574 pixels of Hive1 (66 x 39)
initial $readmemh("Hive1.mem", Hivememory_array);
```

Hivememory_array is used to save a copy of the original Hive, used when the Hives require resetting

```
// Reset Hives at the end of each Level
        if ((sx>640) && (sy>480) && (ResetHives==1) && (lpcounter<2574))
            begin
                write1 <= 1;
                write2 <= 1;
                write3 <= 1;
                write4 <= 1;
                H1address<=lpcounter;
                H2address<=lpcounter;
                H3address<=lpcounter;
                H4address<=lpcounter;
                data1 <= Hivememory_array[lpcounter];
                data2 <= Hivememory_array[lpcounter];
                data3 <= Hivememory_array[lpcounter];
                data4 <= Hivememory_array[lpcounter];
                lpcounter<=lpcounter+1;
                if(lpcounter==2573)
                    lpcounter<=0;

            end
```

If ResetHives equals 1 (level completed) all 4 Hives are reset with their original data which is saved in Hivememory_array

lpcounter is used as a pointer into ResetHives and once it reaches 2573 it is reset to 0

05 ScoreSprite.v module

```
//--------------------------------------------
// ScoreSprite.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//--------------------------------------------
`timescale 1ns / 1ps

// Setup ScoreSprite module
module ScoreSprite(
    input wire clk_pix,                         // 25.2MHz pixel clock
    input wire [9:0] sx,
    input wire [9:0] sy,
    input wire de,
    input wire [16:0] Score,                    // Score value
    output reg [1:0] ScoreSpriteOn,             // 1=on, 0=off
    output wire [7:0] Scoredout,                // pixel value from Score.mem
    output reg [1:0] DigitsSpriteOn,            // 1=on, 0=off
    output wire [7:0] Digitsdout                // pixel value from Digit.mem
    );
```

This module sets up the current score at the top left hand corner of the screen

Score is updated in the AlienSprites.v module and inputted into this section

ScoreSpriteOn, Scoredout, DigitsSpriteOn and Digitsdout are all outputted to the Top.v module

```
 // instantiate ScoreRom
reg [9:0] Scoreaddress;                    // 2^10 or 1023, need 55 x 13 = 715
ScoreRom ScoreVRom (
   .Scoreaddress(Scoreaddress),
   .clk_pix(clk_pix),
   .Scoredout(Scoredout)
);

// instantiate DigitsRom
reg [10:0] Digitsaddress;                  // 2^11 or 2047, need 110 x 13 = 1430
DigitsRom DigitsVRom (
   .Digitsaddress(Digitsaddress),
   .clk_pix(clk_pix),
   .Digitsdout(Digitsdout)
);
```

ScoreRom.v is instantiated to load the graphics for the word Score (55 x 13 = 715 pixels)

DigitsRom.v is instantiated to load the graphics for the digits 0123456789 (110 x 13 = 1430 pixels)

```
// setup Score and Digits positions, sizes and variables
localparam ScoreWidth = 55;             // Score width in pixels
localparam ScoreHeight = 13;            // Score height in pixels
localparam ScoreX = 6;                  // Score X position
localparam ScoreY = 6;                  // Score Y position
localparam scorevalx = 80;              // Score value X position
localparam scposxl = 11;                // One digit - pixel width
localparam scposyl = 13;                // One digit - pixel height
reg [10:0] scoreyoffset=0;              // Y offset used in calculating Digitsaddress
reg [16:0] digit5=0;                    // 5th digit value of score
reg [13:0] digit4=0;                    // 4th digit value of score
reg [9:0] digit3=0;                     // 3rd digit value of score
reg [6:0] digit2=0;                     // 2nd digit value of score
reg [3:0] digit1=0;                     // 1st digit value of score
reg [6:0] t10=10;                       // used to calculate digit1-5
reg [9:0] t100=100;                     // used to calculate digit1-5
reg [13:0] t1000=1000;                  // used to calculate digit1-5
reg [16:0] t10000=10000;                // used to calculate digit1-5
reg [7:0] counter = 0;                  // used to calculate digit1-5
```

This section sets up the variables used

scorevalx represents the start x position of where to display the points scored
scposxl, scposyl represent the width and height of each displayed digit
scoreyoffset is used in calculating Digitsaddress and is incremented by 110 thirteen times
digit5-digit1 contain the values of each digit (5 in total, Score range is 00000 to 99999)
t10-t10000 are used in the formulas to calculate the value of each of the 5 digits
counter used in the formulas to calculate the value of each of the 5 digits

```
always @ (posedge clk_pix)
begin
    // if sx,sy are within the confines of the Score character, switch the Score On
    if(de)
        begin
            if((sx==ScoreX-2) && (sy==ScoreY))
                begin
                    Scoreaddress <=0;
                    ScoreSpriteOn <=1;
                end
            if((sx>ScoreX-2) && (sx<ScoreX+ScoreWidth-1) && (sy>ScoreY-1) && (sy<ScoreY+ScoreHeight))
                begin
                    ScoreSpriteOn <=1;
                    Scoreaddress <= Scoreaddress + 1;
                end
            else
                    ScoreSpriteOn <=0;
```

This section of the code switches ScoreSpriteOn on if sx, sy are in the confines of the word Score and fetches the characters pixel values using Scoreaddress

```
        if((sx==scorevalx-2) && (sy==ScoreY))
          begin
             scoreyoffset <= 110;
             DigitsSpriteOn <=1;
              digit5 <= Score / t10000;
              digit4<= (Score-(digit5*t10000)) / t1000;
              digit3<= (Score-((digit5*t10000)+(digit4*t1000))) / t100;
              digit2<= (Score-((digit5*t10000)+(digit4*t1000)+(digit3*t100))) / t10;
              digit1<= (Score-((digit5*t10000)+(digit4*t1000)+(digit3*t100)+(digit2*t10)));
              Digitsaddress <= digit5 * scposxl;
              counter<=0;
          end
```

This section works out the value for each of the 5 digits

Example Score = 01490

digit5 = 01490 / 10000 =	0
digit4 = (01490 – (0 * 10000)) / 1000 =	1
digit3 = (01490 – ((0 * 10000) + (1 * 1000))) / 100 =	4
digit2 = (01490 – ((0 * 10000) + (1 * 1000) + (4 * 100))) / 10 =	9
digit1 = (01490 – ((0 * 10000) + (1 * 1000) + (4 * 100) + (9 * 10))) =	0

```
          if((sx>scorevalx-2) && (sx<scorevalx+(scposxl*5)-1) && (sy>ScoreY-1) && (sy<ScoreY+scposyl))
            begin
             DigitsSpriteOn <=1;
             counter <= counter + 1;
             if(counter==(scposxl*1)-1)
               Digitsaddress<=(digit4*scposxl)+(scoreyoffset-110);
             else
             if(counter==(scposxl*2)-1)
               Digitsaddress<=(digit3*scposxl)+(scoreyoffset-110);
             else
             if(counter==(scposxl*3)-1)
                  Digitsaddress<=(digit2*scposxl)+(scoreyoffset-110);
               else
             if(counter==(scposxl*4)-1)
               Digitsaddress<=(digit1*scposxl)+(scoreyoffset-110);
             else
               Digitsaddress <= Digitsaddress + 1;
        end
            else
              DigitsSpriteOn <=0;
              if(counter==scposxl*5)
              begin
                scoreyoffset <= scoreyoffset + 110;
                Digitsaddress <= (digit5 * scposxl) + scoreyoffset;
                counter<=0;
              end
      end
  end
endmodule
```

counter is incremented by 1 each clock cycle. Four checks are made to see if counter equals the start of the next digits x co-ordinate. The 5th digit will always equal 0, so the first check will be if counter equals the 4th digit x co-ordinate. If it does then Digitsaddress value is changed to point to the correct digit location (specified by digit1-digit4). Digitsaddress is incremented by 1 if none of the 4 checks are met

If sx, sy are not in the confines of the displayed 5 digit score and counter has actually been used, i.e. If counter equals the end of the 5 digits, then increment scoreyoffset by 110 (remember the 5 digit score will be 55 x 13 pixels in size)

Digitsaddress is then recalculated to point to the next pixel and counter is reset to 0

06 Honeycomb1Sprite.v module

```verilog
//--------------------------------------------
// Honeycomb1Sprite.v Module
// Digilent Basys 3
// Bee Invaders Tutorial_6
// Onboard clock 100MHz
// VGA Resolution: 640x480 @ 60Hz
// Pixel Clock 25.2MHz
//--------------------------------------------
`timescale 1ns / 1ps

// Setup Honeycomb1Sprite module
module Honeycomb1Sprite(
    input wire clk_pix,                          // 25.2MHz pixel clock
    input wire [9:0] sx,
    input wire [9:0] sy,
    input wire de,
    output reg [1:0] Honeycomb1SpriteOn,         // 1=on, 0=off
    output wire [7:0] Honeycomb1dout             // pixel value from Honeycomb1.mem
    );
```

This module sets up the Honeycomb Level Indicator

Honeycomb1SpriteOn, Honeycomb1dout are outputted to the Top.v module

```
// instantiate Honeycomb1Rom
reg [10:0] Honeycomb1address;                      // 2^11 or 2047, need 82 x 13 = 1066
Honeycomb1Rom Honeycomb1VRom (
    .Honeycomb1address(Honeycomb1address),
    .clk_pix(clk_pix),
    .Honeycomb1dout(Honeycomb1dout)
);

// setup Honeycomb1 character positions and sizes
localparam Honeycomb1Width = 82;                   // Score width in pixels
localparam Honeycomb1Height = 13;                  // Score height in pixels
localparam Honeycomb1X = 279;                      // Honeycomb1 X position
localparam Honeycomb1Y = 6;                        // Honeycomb1 Y position
```

This section of code instantiates Honeycomb1Rom.v (loading the pixel values for the Honeycomb Level Indicator

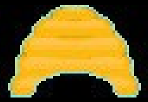
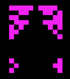
The Level Indicators width, height, x and y co-ordinates are then set

```
    always @ (posedge clk_pix)
    begin
        // if sx,sy are within the confines of the Score character, switch the Score On
        if(de)
            begin
                if((sx==Honeycomb1X-2) && (sy==Honeycomb1Y))
                    begin
                        Honeycomb1address <=0;
                        Honeycomb1SpriteOn <=1;
                    end
                if((sx>Honeycomb1X-2) && (sx<Honeycomb1X+(Honeycomb1Width*1)-1) && (sy>Honeycomb1Y-1) && (sy<Honeycomb1Y+Honeycomb1Height))
                    begin
                        Honeycomb1SpriteOn <=1;
                            Honeycomb1address <= Honeycomb1address + 1;
                    end
                else
                        Honeycomb1SpriteOn <=0;
            end
    end
endmodule
```

The final part of the code switches the Honeycomb Level Indicator on if sx, sy are within its confines.

It also increments Honeycomb1address in order to retrieve all the pixel data for the indicator character

F) SPRITE SIZES

	Bee	34 x 27 Pixels (WxH)	918 Total Pixels
	Bee Bullet	1 x 7 Pixels (WxH)	7 Total Pixels
	Bee Hive	66 x 39 Pixels (WxH)	2574 Total Pixels
	Bullet Hole	11 x 16 Pixels (WxH)	176 Total Pixels
	Alien1	31 x 26 Pixels (WxH)	806 Total Pixels
	Alien2	31 x 21 Pixels (WxH)	651 Total Pixels
	Alien3	31 x 27 Pixels (WxH)	837 Total Pixels
Score	Score	55 x 13 pixels (WxH)	715 Total Pixels
0123456789	Digits	110 x 13 pixels (WxH)	1430 Total Pixels
	Honeycomb1	82 x 13 pixels (WxH)	1066 Total Pixels