

PROJECT: BEE INVADERS

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

Tutorial 2 - Display Our Bee At The Bottom Of The Screen



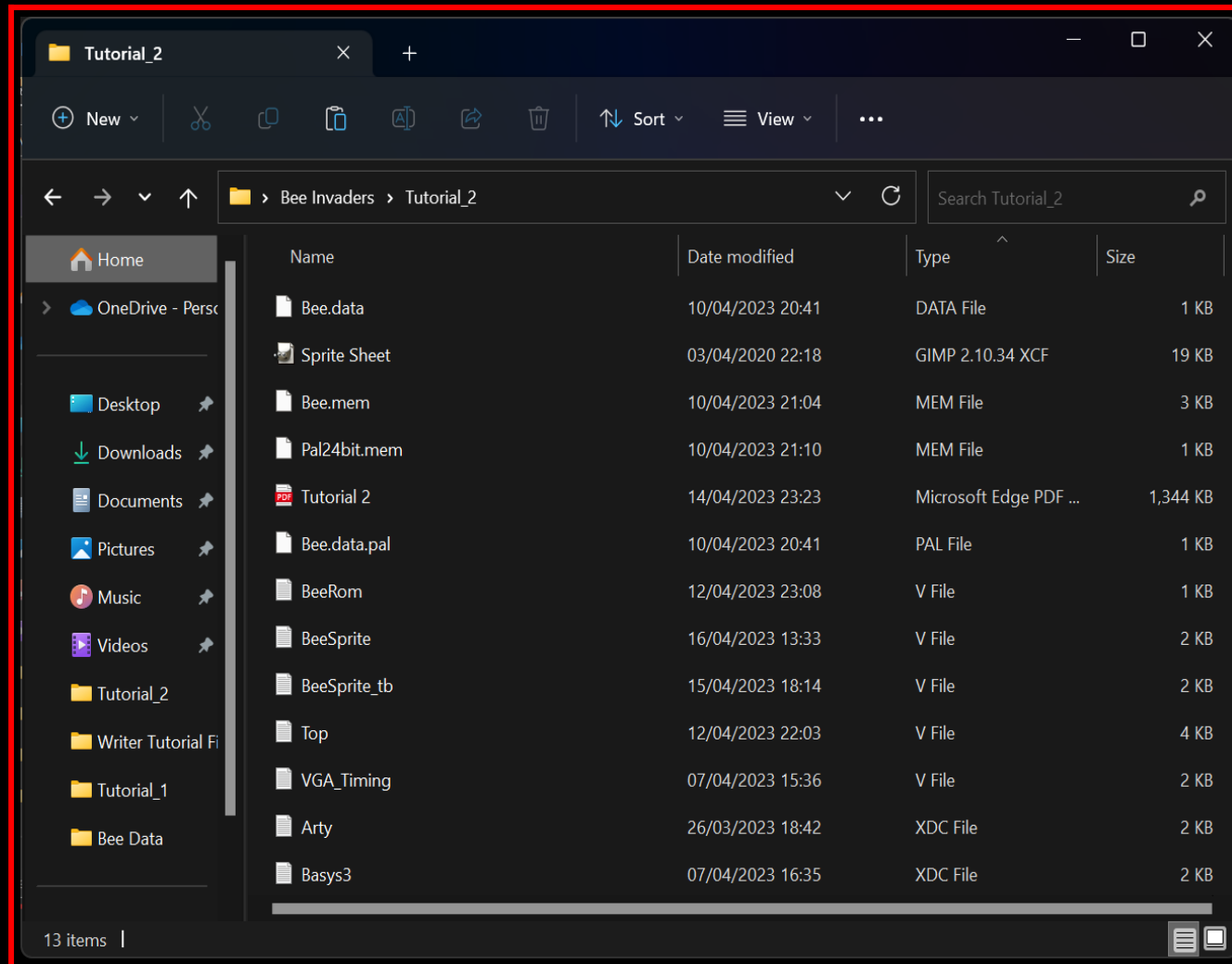
CONTENTS

- (A) USING GIMP TO GENERATE THE GRAPHIC FOR OUR BEE
- (B) CREATING THE PROJECT IN VIVADO
- (C) TESTBENCH SHOWING THE DATA LOADING OF THE BEE
- (D) THE CODE FOR THIS TUTORIAL
- (E) THE BLOCK RAM OF THE BASYS3 FPGA BOARD
- (F) EXPLANATION OF THE VERILOG CODE USED

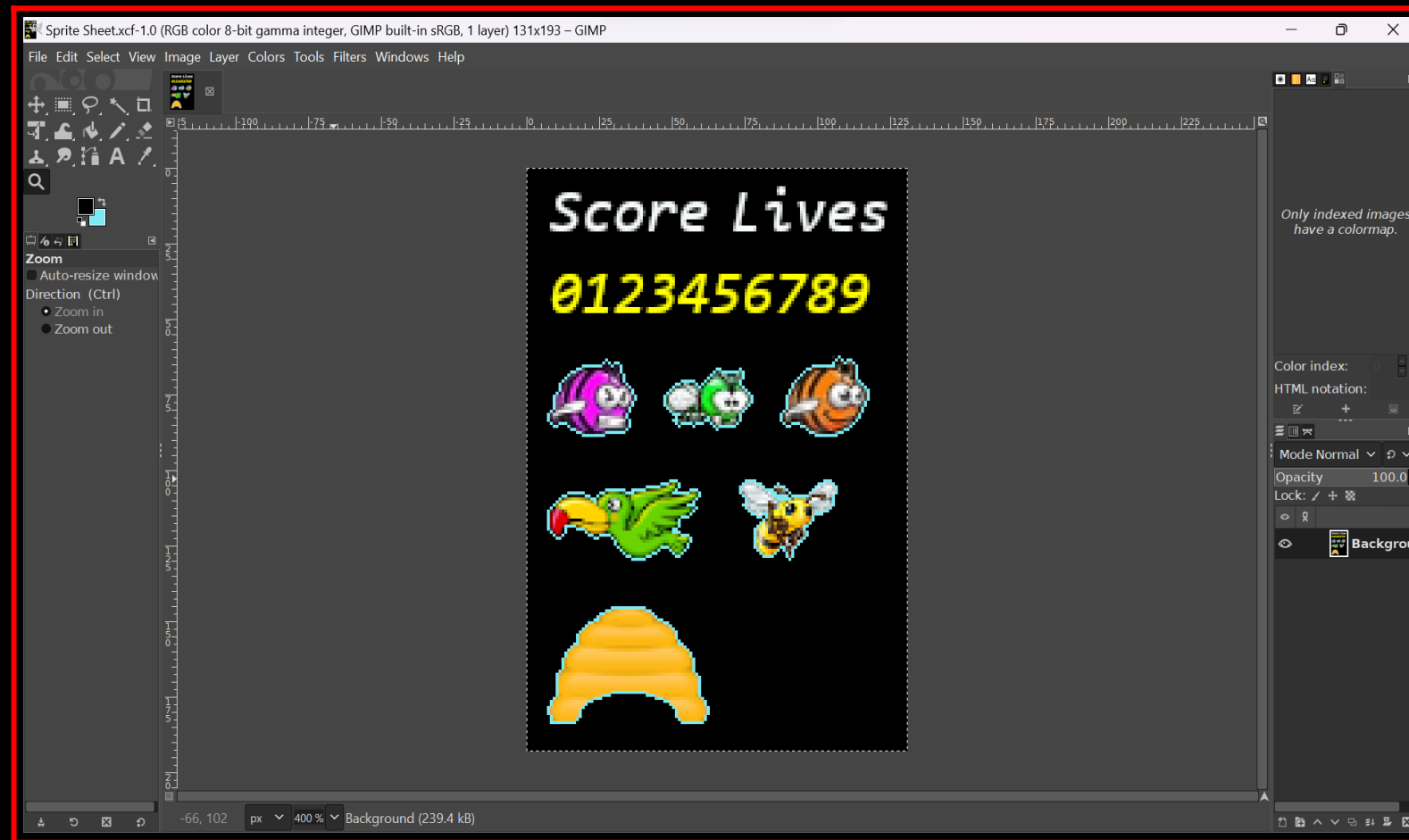
(A) USING GIMP TO GENERATE THE GRAPHIC FOR OUR BEE

01

In the folder "Bee Invaders" create a folder called "Tutorial_2" and extract the files from the downloaded file "Tutorial_2 Files.zip" to this folder. In the "Tutorial_2" folder you will find the file "Sprite Sheet.xcf" (the graphics used in this game).



02 Download / install from the Gimp website their Free image manipulation software and open the file "Sprite Sheet.xcf" in the "Tutorial_2" folder with Gimp

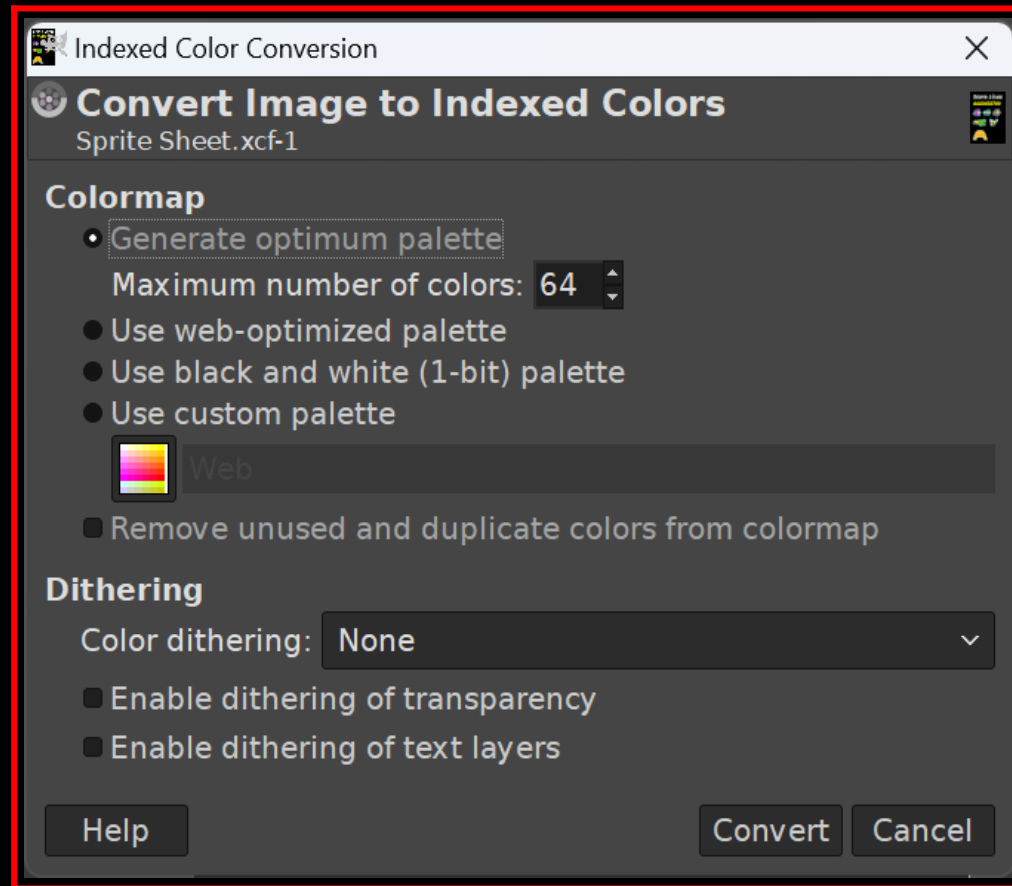


03

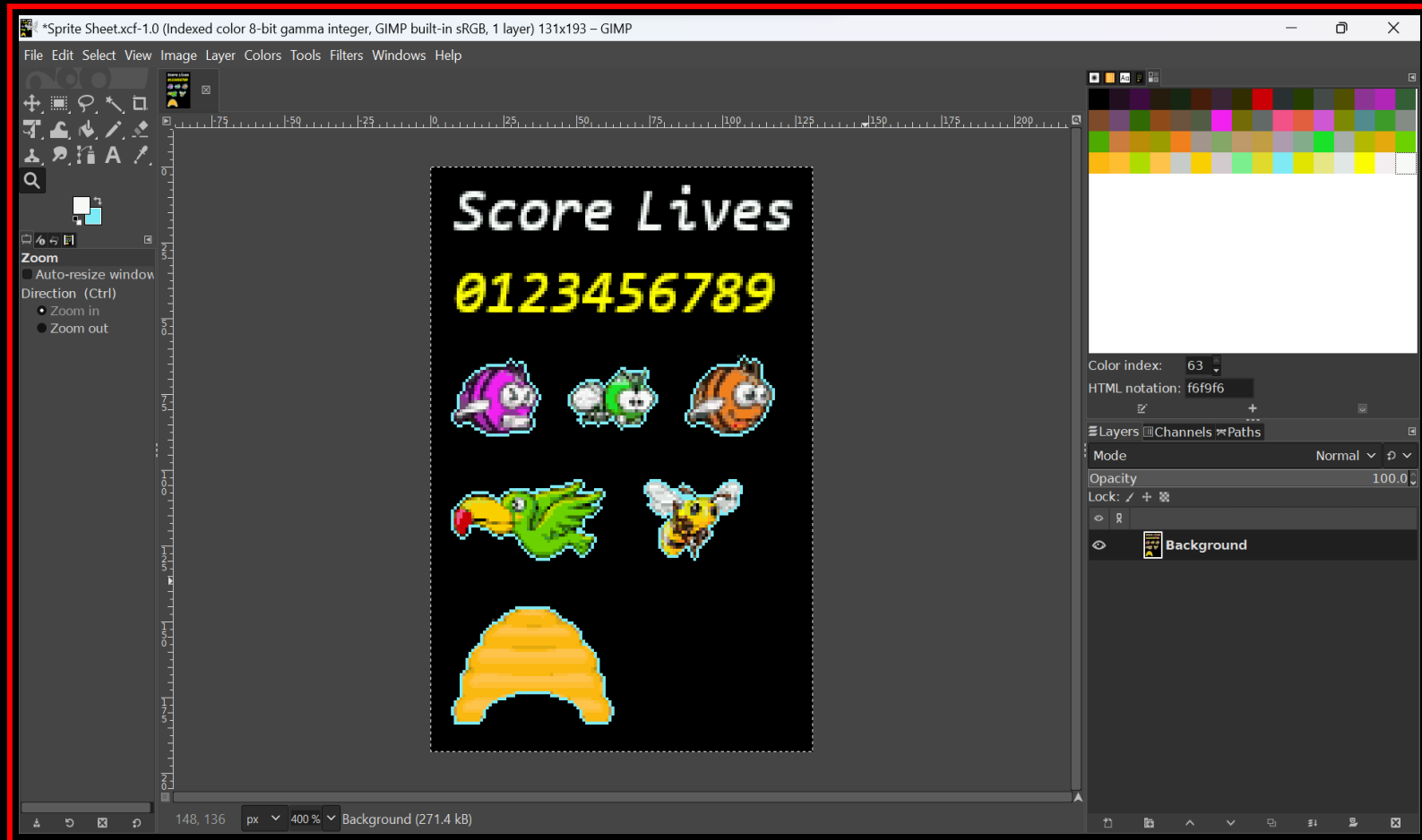
The sprite sheet needs to be converted to 64 colours by selecting: Image → Mode → Indexed

Set the maximum number of colours to 64

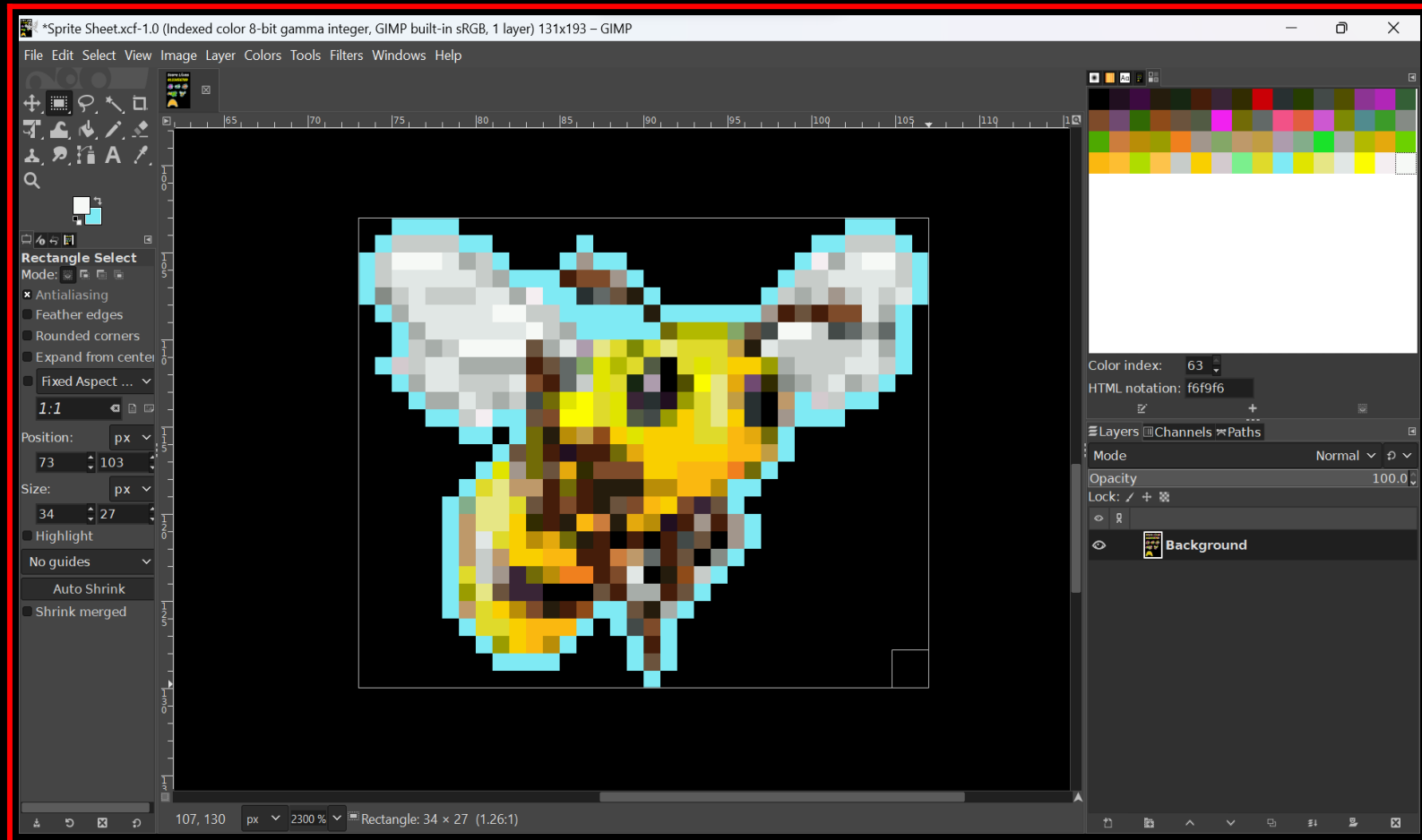
Please make sure that Remove unused and duplicate colors from colormap is NOT selected and click on Convert (as shown below)



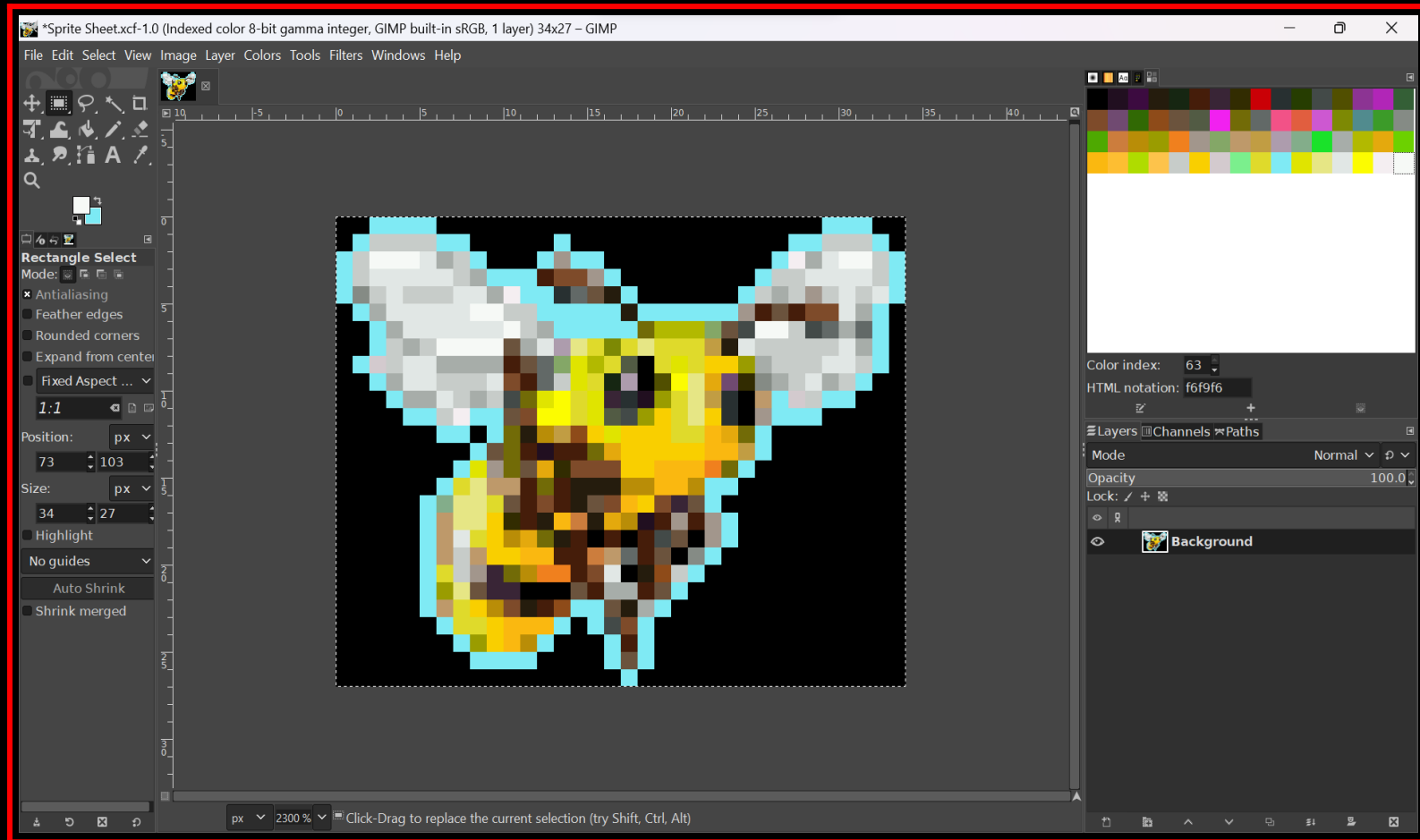
04 Select: Windows → Dockable Dialogs → Colormap, to show the colour palette created (0 - 63 different colours have been obtained from the Sprite Sheet)



05 Zoom in on the Bee character and using the "rectangle select tool" select around the bee (this should be a rectangle 34 x 27 pixels)

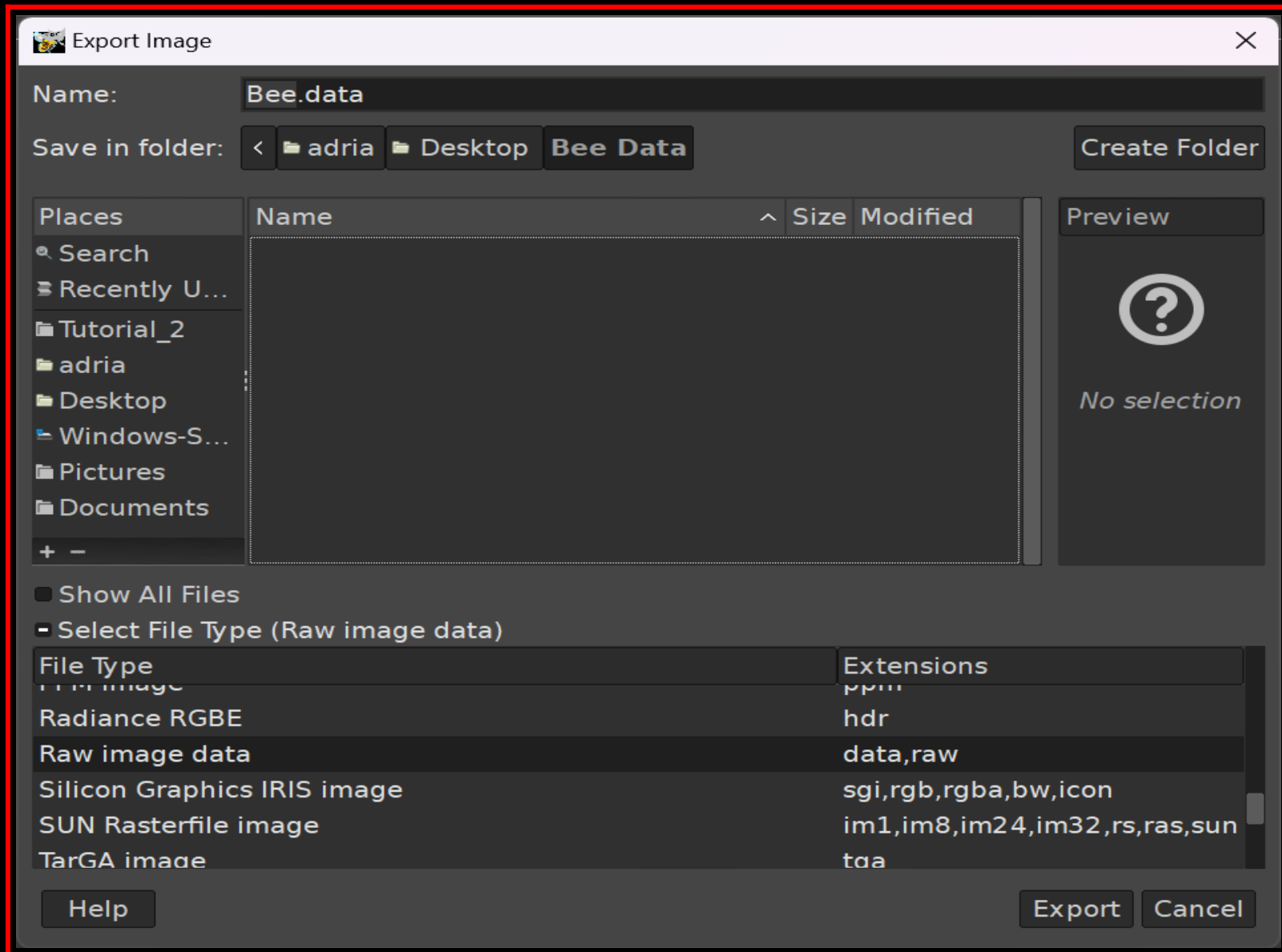


06 Now select: Image → Crop to Selection



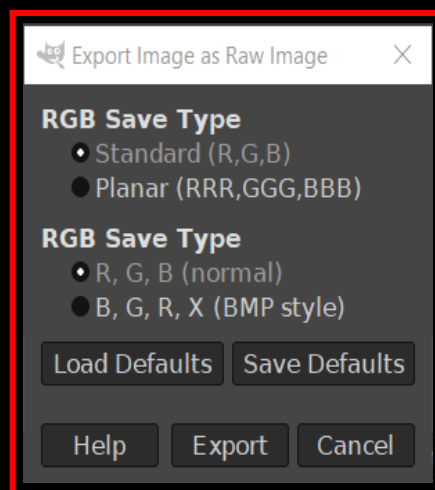
07

The image needs to be saved as a Raw Data File, do this using File → Export As → Raw image data. Call the file "Bee.data", point "Save in folder" to somewhere on your computer and click "Export"

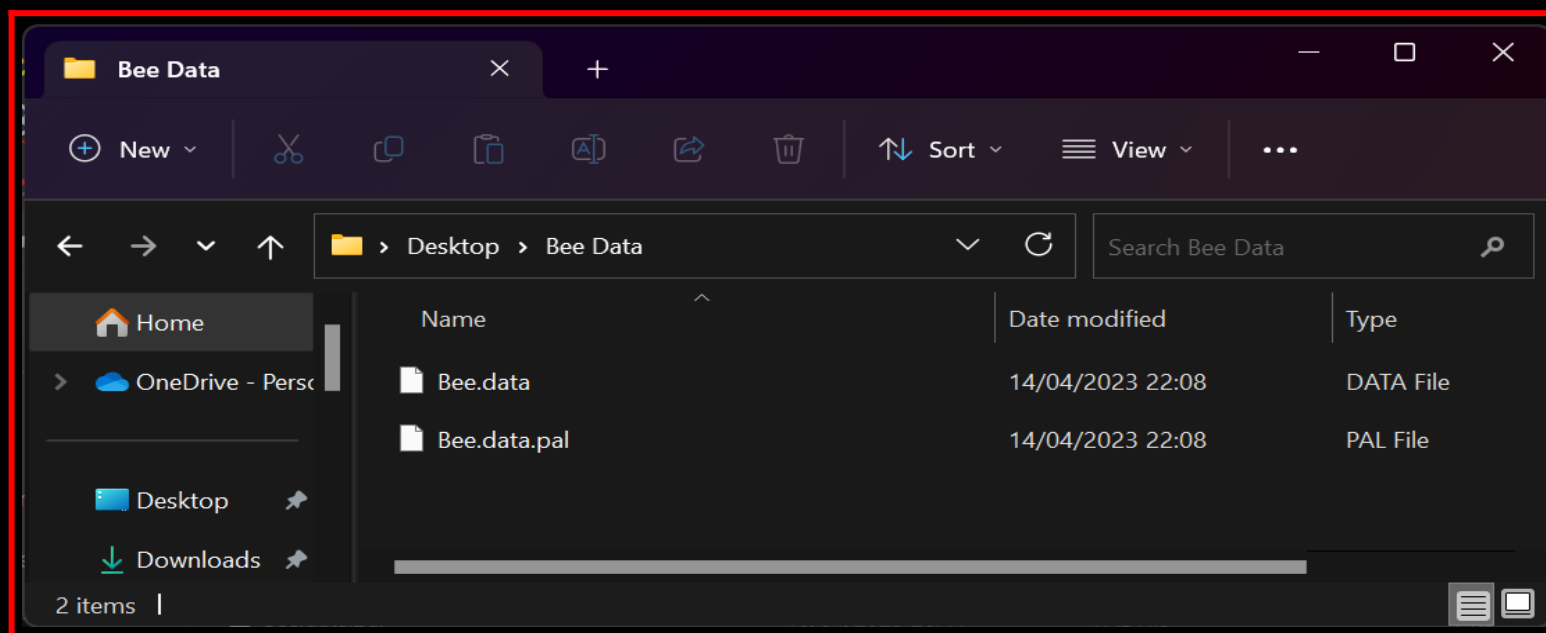


08

When the below screen appears click "Export"

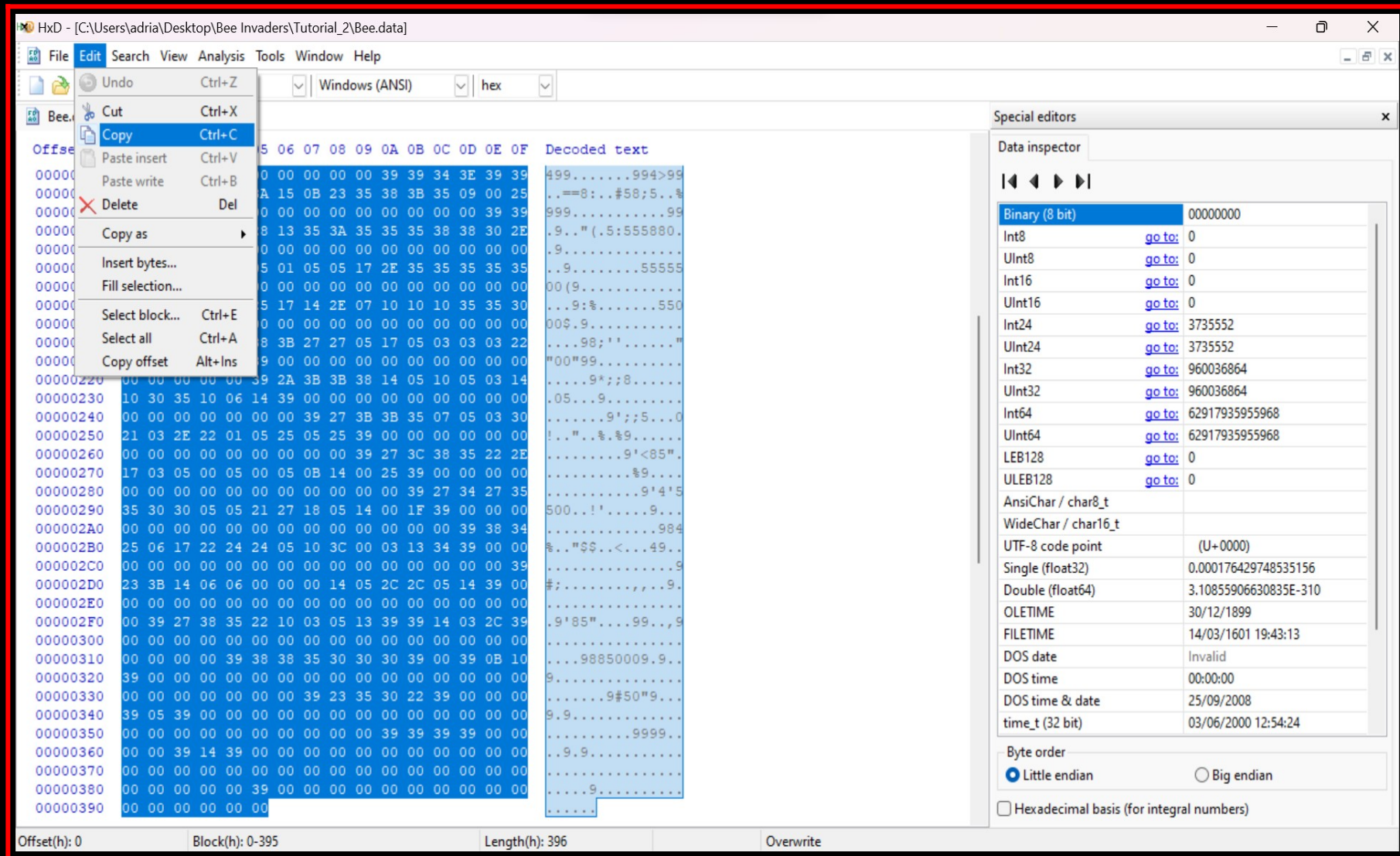


If you navigate to the folder where you exported the Bee character, you should now have 2 files in the folder (Bee.data and Bee.data.pal)



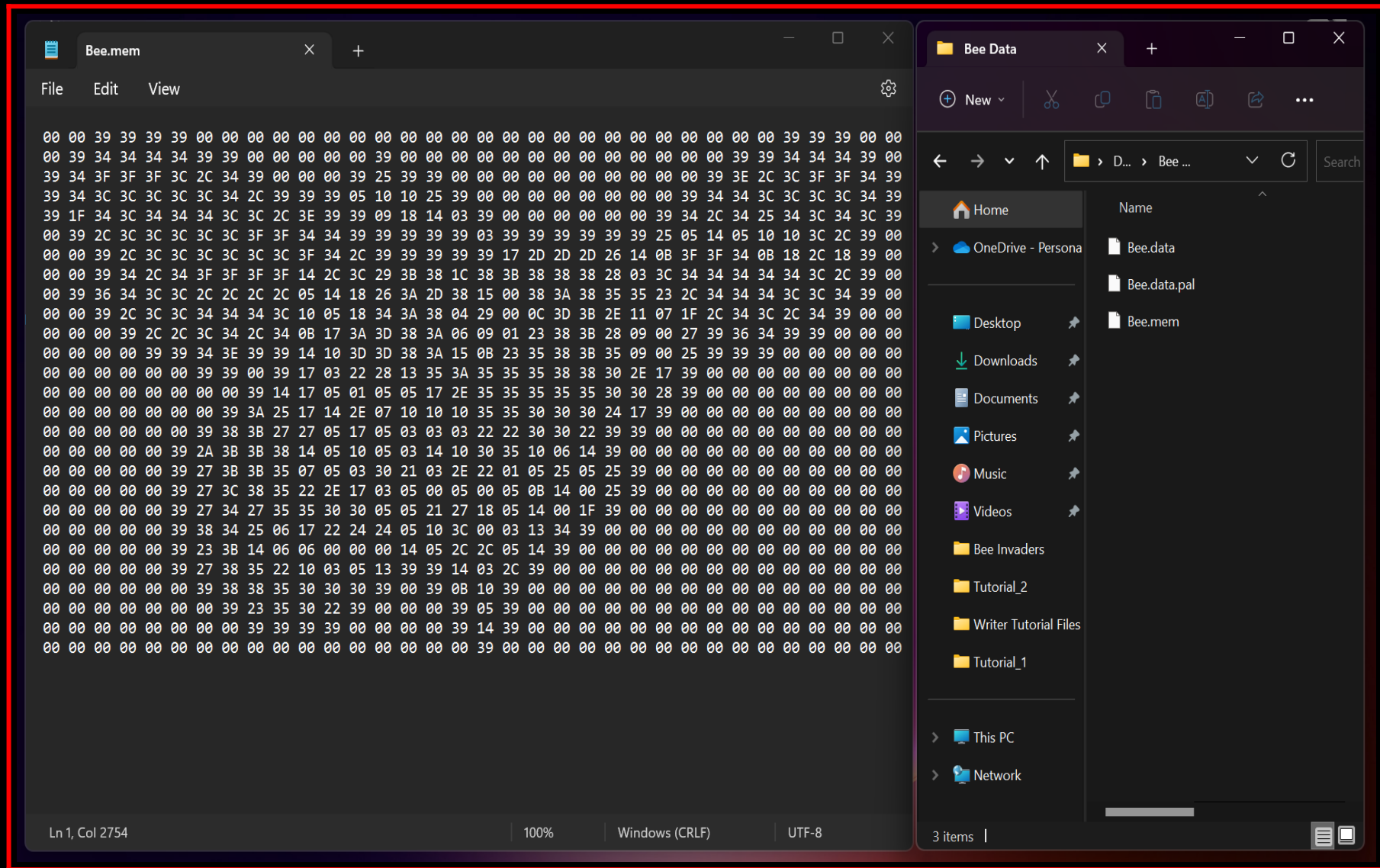
09

Next download, install and run the free program (or similar) called HxD Hex Editor and load the file "Bee.data", select all the data and copy it



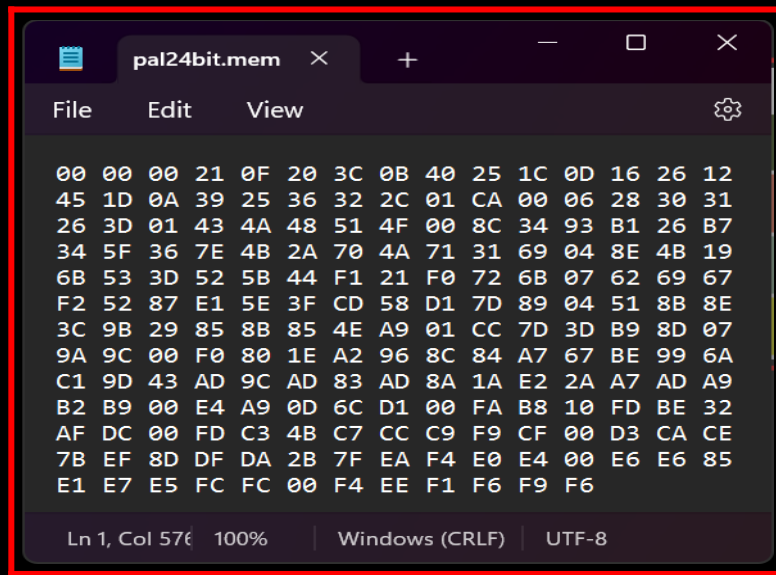
10

Paste the data into a Notepad file and save it as "Bee.mem" in the same folder you exported the Bee character to



11

Do the same with the "Bee.data.pal" file but save the Notepad file as "pal24bit.mem"



```

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



The file contains data for 8 bit Red, 8 bit Green and 8 bit Blue colours (3 sets of 8 bit data by 64 colours = 24bit colour values)

(B) CREATING THE PROJECT IN VIVADO

01

In the folder "Bee Invaders/Tutorial_2" you will find the "Bee.mem" and "Pal24bit.mem", these are the same as the 2 files you created earlier

Follow the instructions in "Tutorial 1" to create a new project in the "Tutorial_2" folder in Vivado but call it "BeeInvaders_WIP". We will use this project name for the remaining tutorials

Add these design sources from the "Tutorial 2" folder:

Top.v
VGA_Timing.v
BeeSprite.v
BeeRom.v
Bee.mem
Pal24bit.mem

Add a constraints file from the "Tutorial 2" folder:

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

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

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

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

With:

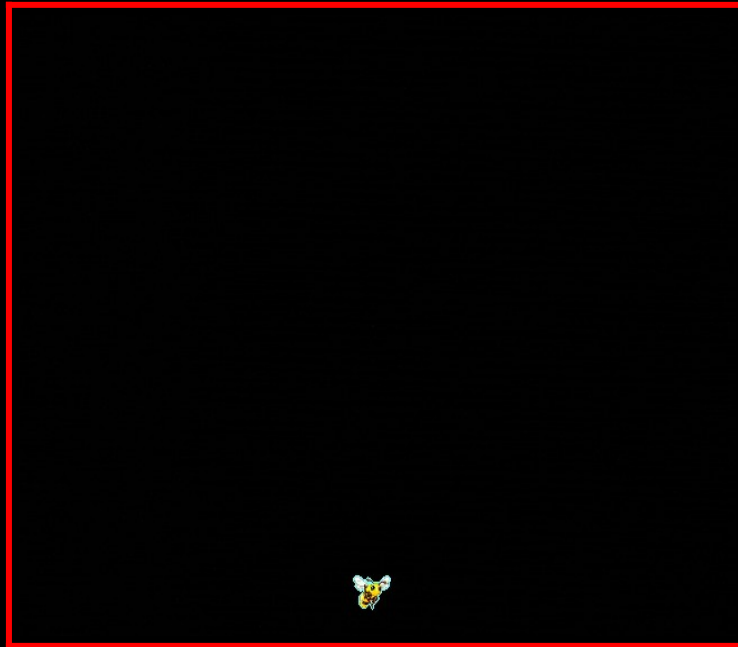
```
.reset(!btn_rst_n),       // reset button is active low
```

02

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

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

You should see our Bee on your VGA monitor, as shown below



Original From Sprite Sheet.xcf



Converted To 64 Colours



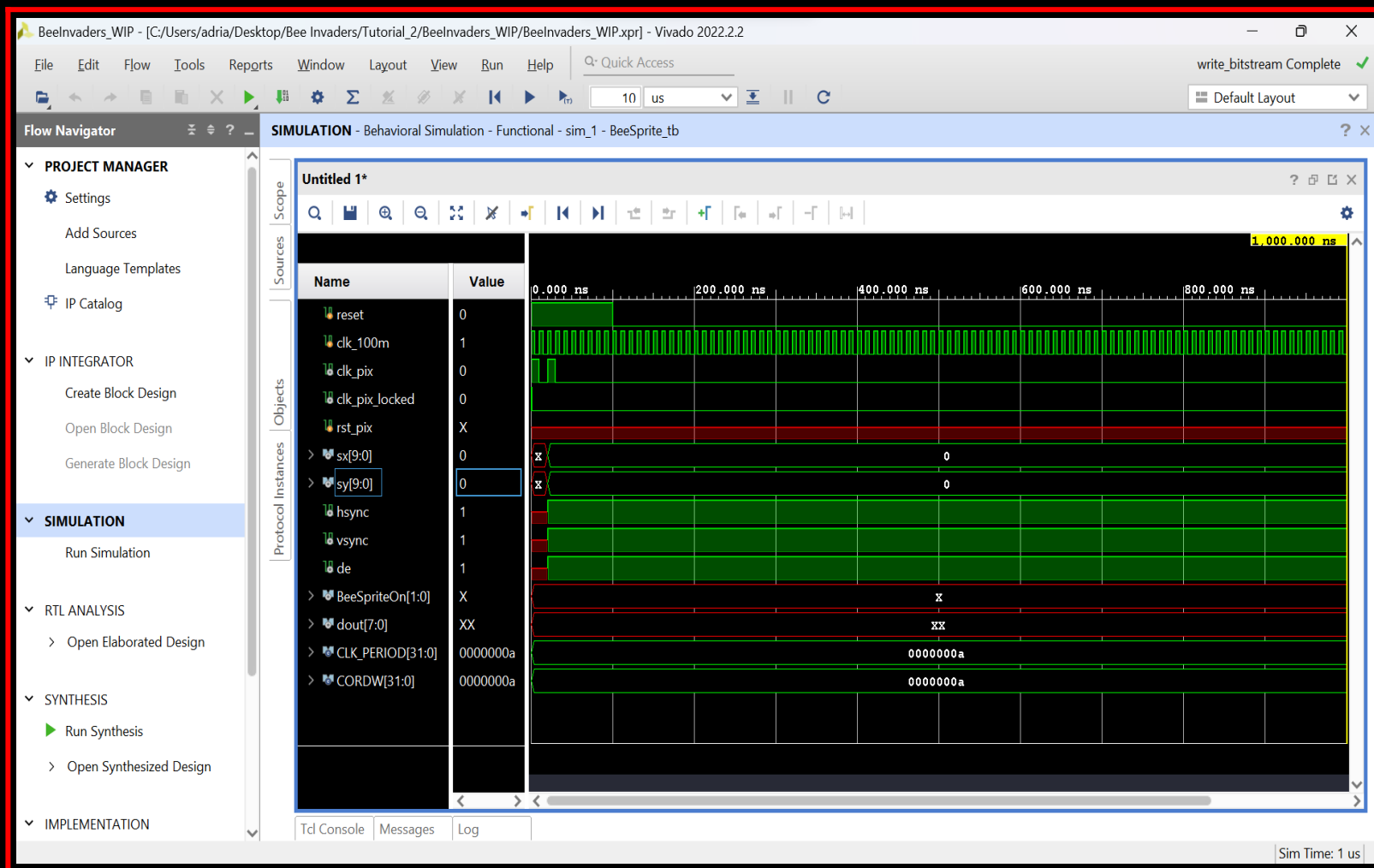
FPGA VGA Generated
Obtained Using Epiphan
VGA2USB LR Frame Grabber



(C) TESTBENCH SHOWING THE DATA LOADING OF THE BEE

01

Add a "Simulation Source" (a testbench called "SpriteBee_tb.v" found in the "Tutorial 2" folder) to the project. Set it as "Set As Top" and then click on "Run Simulation" (refer to "Tutorial 1" on how to add the testbench). Maximize the simulation window, select "Full View" and set the Radix for sx[9:0] and sy[9:0] to Unsigned Decimal



02

Click on the "Run All" button and when "sy[9:0]" reaches around "500" click the "Break" button. Use the "Zoom" buttons and the "Scroll Bar" to position "sx[9:0]" around "297" and "sy[9:0]" equal to "433"

BeelInvaders_WIP - [C:/Users/adria/Desktop/Bee Invaders/Tutorial_2/BeelInvaders_WIP/BeelInvaders_WIP.xpr] - Vivado 2022.2.2

File Edit Flow Tools Reports Window Layout View Run Help Q Quick Access write_bitstream Complete ✓ Default Layout

Flow Navigator

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

SIMULATION - Behavioral Simulation - Functional - sim_1 - BeeSprite_tb

Untitled 1*

Scope Sources Objects Protocol Instances

Name	Value
reset	0
clk_100m	0
clk_pix	1
clk_pix_locked	1
rst_pix	X
> sx[9:0]	273
> sy[9:0]	433
hsync	1
vsync	1
de	1
> BeeSpriteOn[1:0]	0
> dout[7:0]	XX
> CLK_PERIOD[31:0]	0000000a
> CORDW[31:0]	0000000a

13 766 400 000 ns 13 766 600 000 ns 13 766 800 000 ns 13 767 000 000 ns 13 767 200 000 ns

.. 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 .

433

0 1

xx 00 39 00

0000000a 0000000a

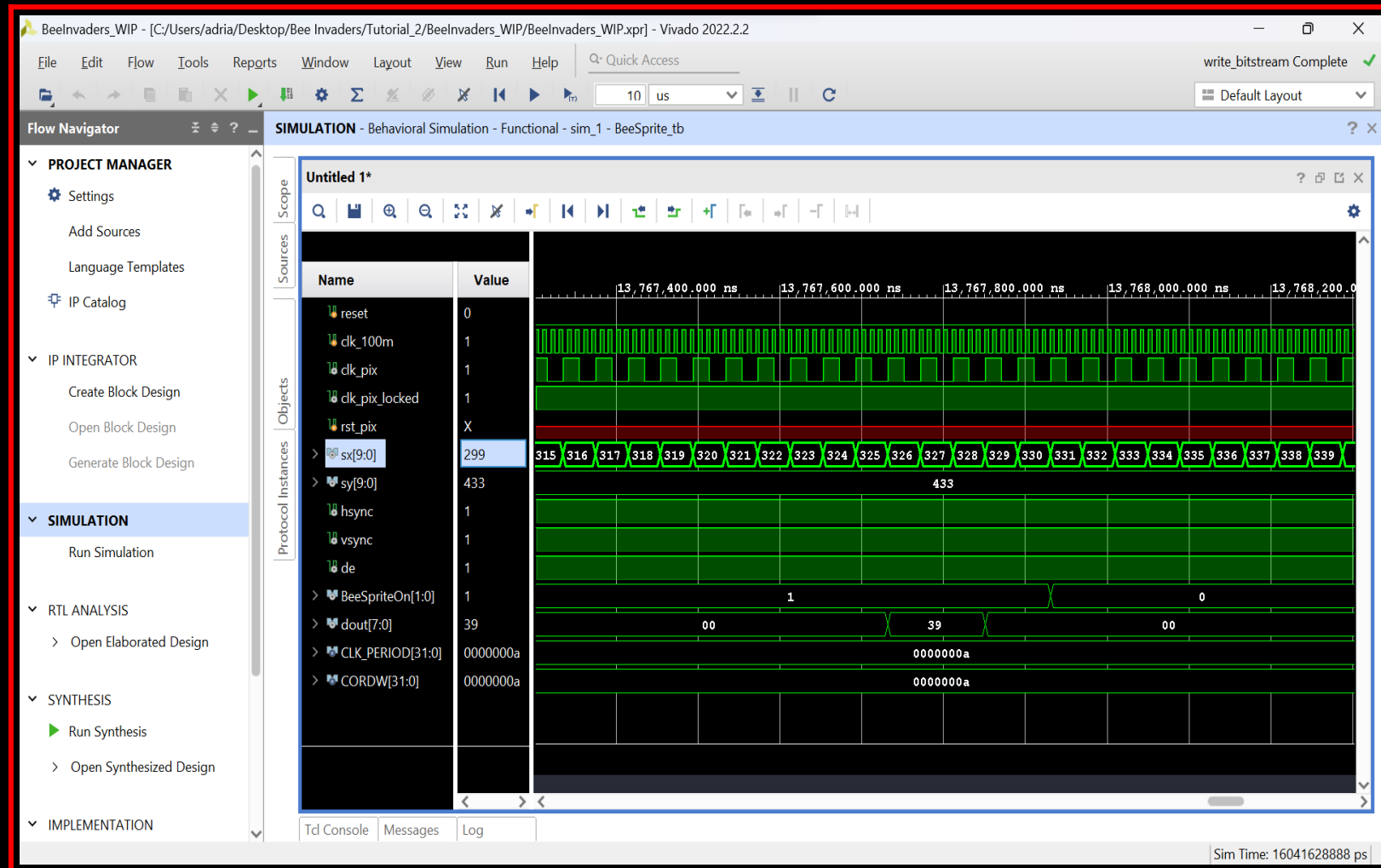
Tcl Console Messages Log

Sim Time: 1604162888 ps

03 When "sx[9:0]" equals "297" notice that "BeeSpriteOn[1:0]" changes to "1" (on)

The can also see the read data: sx[9:0] 297 298 299 300 301 302 303 and so on....
dout[7:0] 00h 00h 39h 39h 39h 39h 00h

Use the "Scroll Bar" to position "sx[9:0]" around "330" and "sy[9:0]" still equal to "433"



When "sx[9:0]" equals "331" notice that "BeeSpriteOn[1:0]" changes to "0" (off)

The can also see the read data:

sx[9:0]	324	325	326	327	328	329	330
dout[7:0]	00h	00h	39h	39h	39h	00h	00h

If you compare this to the first 34 values from "Bee.mem" you will see the data is the same. This is actually the first row of pixels for the Bee sprite

(00 00 39 39 39 39 00) 00 (00 00 39 39 39 00 00)

If you do the same for when "sy[9:0]" equals "434" you should see the read data for the second row of pixels from the Bee sprite

You will see later in this tutorial that the Bee sprite is defined as:

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

You will also see later that there is a latency when reading the data for the Bee and this had to be taken into consideration when writing the code. This testbench came in really helpful and quick when it came to checking that the code was correct

(D) THE CODE FOR THIS TUTORIAL

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

```
//-----
// Top.v module
// Digilent Basys 3
// Bee Invaders Tutorial_2
// 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
);

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

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

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

    // Instantiate BeeSprite
```

```

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

// 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
                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
    end
endmodule

```

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

```
//-----
// VGA_Timing.v module
// Digilent Basys 3
// Bee Invaders Tutorial_2
// 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
```

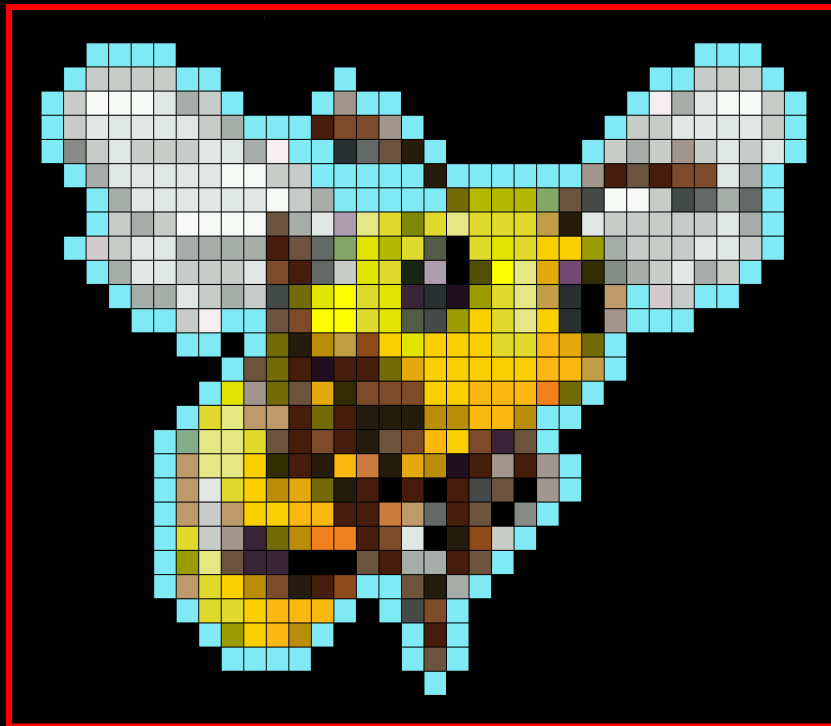
This is the code from the file "BeeSprite.v"

```
//-----  
// BeeSprite.v Module  
// Digilent Basys 3  
// Bee Invaders Tutorial_2  
// Onboard clock 100MHz  
// VGA Resolution: 640x480 @ 60Hz  
// Pixel Clock 25.2MHz  
//-----  
`timescale 1ns / 1ps  
  
// Setup BeeSprite module  
module BeeSprite(  
    input wire clk_pix,  
    input wire [9:0] sx,  
    input wire [9:0] sy,  
    input wire de,  
    output reg [1:0] BeeSpriteOn, // 1=on, 0=off  
    output wire [7:0] dataout  
);  
  
    // instantiate BeeRom code  
    reg [9:0] address; // 2^10 or 1024, need 34 x 27 = 918  
    BeeRom BeeVRom (  
        .address(address),  
        .clk_pix(clk_pix),  
        .dataout(dataout)  
    );  
  
    // setup character positions and sizes  
    reg [9:0] BeeX = 297; // Bee X start position  
    reg [8:0] BeeY = 433; // Bee Y start position  
    localparam BeeWidth = 34; // Bee width in pixels  
    localparam BeeHeight = 27; // Bee height in pixels  
  
    // check if sx,sy are within the confines of the Bee character  
    always @ (posedge clk_pix)  
    begin  
        if(de)  
            begin  
                if(sx==BeeX-2 && sy==BeeY) // sx=295  
                    begin  
                        address <= 0; // 1st Entry: address = 0  
                        BeeSpriteOn <=1;  
                    end  
                if((sx>BeeX-2) && (sx<BeeX+BeeWidth-1) && (sy>BeeY-1) && (sy<BeeY+BeeHeight)) // sx = 296 to 329 = 33 Entries  
                    begin  
                        address <= (sx+2-BeeX) + ((sy-BeeY)*BeeWidth); // 2nd Entry: address = 296 + 2 - 297 = 1  
                        BeeSpriteOn <=1;  
                    end  
                else  
                    BeeSpriteOn <=0;  
            end  
        end  
    end  
endmodule
```


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

```
//-----  
// BeeRom.v Module  
// Single Port ROM  
// Digilent Basys 3  
// Bee Invaders Tutorial_2  
// Onboard clock 100MHz  
// VGA Resolution: 640x480 @ 60Hz  
// Pixel Clock 25.2MHz  
//-----  
`timescale 1ns / 1ps  
  
// Setup BeeRom module  
module BeeRom(  
    input wire [9:0] address, // (9:0) or 2^10 or 1024, need 34 x 27 = 918  
    input wire clk_pix,  
    output reg [7:0] dataout // (7:0) 8 bit pixel value from Bee.mem  
);  
  
    (*ROM_STYLE="block"*) reg [7:0] memory_array [0:917]; // 8 bit values for 918 pixels of Bee (34 x 27)  
  
    initial  
    begin  
        $readmemh("Bee.mem", memory_array);  
    end  
  
    always@(posedge clk_pix)  
        dataout <= memory_array[address];  
endmodule
```

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



This is the data from the file "Pal24bit.mem"

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

This is the code from the file "Basys3.xdc"

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

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

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

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

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

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

## 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]}];
```

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

```
//-----  
// BeeSprite_tb.v module  
// Digilent Basys 3  
// Bee Invaders Tutorial_2  
// Onboard clock 100MHz  
// VGA Resolution: 640x480 @ 60Hz  
// Pixel Clock 25.2MHz  
//-----  
  
`timescale 1ns / 1ps  
  
module BeeSprite_tb();  
  
    parameter CLK_PERIOD = 10; // 10 ns == 100 MHz  
    parameter CORDW = 10;      // screen coordinate width in bits  
  
    // VGA_Clock  
    reg reset;  
    reg clk_100m;  
    wire clk_pix;  
    wire clk_pix_locked;  
    VGA_Clock clock_pix_inst (  
        .clk_100m(clk_100m),  
        .reset(reset),  
        .clk_pix(clk_pix),  
        .clk_pix_locked(clk_pix_locked)  
    );  
  
    // VGA_Timing  
    reg rst_pix;  
    wire [9:0] sx;  
    wire [9:0] sy;  
    wire hsync;  
    wire vsync;  
    wire de;  
    VGA_Timing display_inst (  
        .clk_pix(clk_pix),  
        .rst_pix(!clk_pix_locked),  
        .sx(sx),  
        .sy(sy),  
        .hsync(hsync),  
        .vsync(vsync),  
        .de(de)  
    );  
  
    // BeeSprite  
    wire [1:0] BeeSpriteOn; // 1=on, 0=off  
    wire [7:0] dout;        // pixel value from Bee.mem  
    BeeSprite BeeDisplay (  
        .clk_pix(clk_pix),  
        .sx(sx),  
        .sy(sy),  
        .de(de),  
        .BeeSpriteOn(BeeSpriteOn),  
        .dataout(dout)  
    );  
  
    // generate clock  
    always #(CLK_PERIOD / 2) clk_100m = ~clk_100m;  
    initial
```


(E) THE BLOCK RAM OF THE BASYS3 FPGA BOARD

The Basys3 board has 1,800 Kbits of fast Block RAM (BRAM) which can be configured as single, or dual port RAM, a ROM, or even a FIFO. To store our Bee image we will use a ROM (read-only-memory), as we do not need to modify the data.

In this tutorial we use 2 memory files "Pal24bit.mem" and "Bee.mem". Because the "pal24bit.mem" file is small and we have not specified which type of RAM to use, Vivado has implemented this using "Distributed RAM" (the FPGA logic cells have look up tables (LUTs) that can be configured as memory)

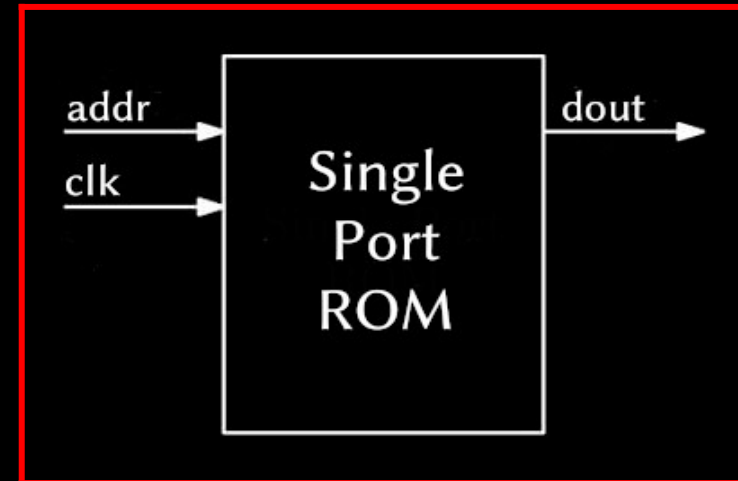
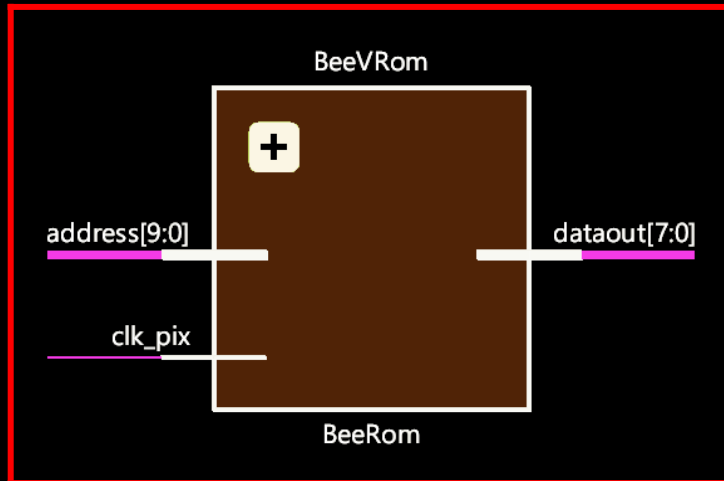
A Memory (.mem) file is a text file that describes contiguous blocks of data. All data values must be the same number of bits wide and must be the same width as expected by the memory model

In this tutorial we have added a register called "memory_array" in the "BeeRom.v" module using (*ROM_STYLE="block"*) reg [7:0] memory_array [0:917]; to represent an 8 bit (7:0) register memory array which can hold 918 entries (0:917)

We have also instructed Vivado to specify which style of RAM to use ("block" ROM). The register was then used to read the pixel data from "Bee.mem" using \$readmemh("Bee.mem", memory_array);

The "BeeRom.v" module used to read the Bee character data from the "Bee.mem" file is a:

Single Port ROM The below diagrams show a "Single Port Rom" with 2 inputs (a clock input and a 10 bit address input) and 1 output (an 8 bit data output)

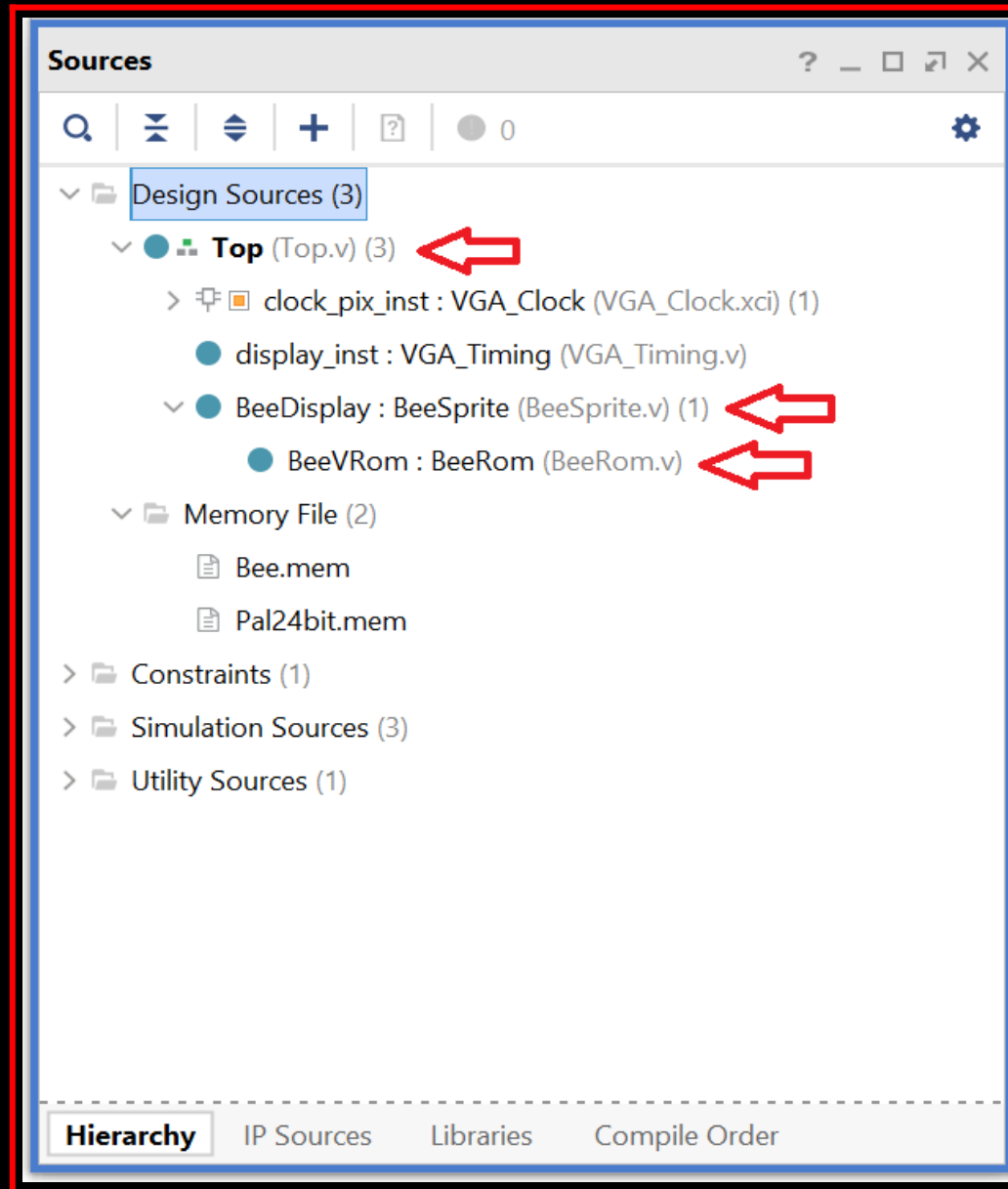


The code we are using to create a Single Port ROM for the Bee character consists of:

`address [9:0]` The Bee character is 34×27 pixels = 918 pixels. [9:0] represents 10 bits, 2^{10} or 1024, the closest number of bits above 918 pixels. "address" is used in the "BeeSprite.v" and "BeeRom.v" module to point to an address in the "memory_array" for the Bee character

`dataout[7:0]` The data found at the address of the "memory_array" is outputted in "dataout"
The Single Port ROM forms part of the "BeeSprite.v" module

(F) EXPLANATION OF THE VERILOG CODE USED



01

Top.v module additional code

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

This links the "Top.v" module to the "BeeSprite.v" module

BeeSpriteOn will equal 1 if the Bee sprite x, y coordinates equal the current "VGA_Timing.v" x, y coordinates, else it will equal 0

dout will contain the hex pixel colour values from the "Bee.mem" file

```
// 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
```

This reads the 8 bit RGB values from "pal24bit.mem" (1xRed + 1xGreen + 1xBlue x 64 = 192 values) and stores the data in the 8 bit register palette

```

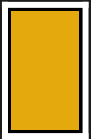
// VGA Output
assign vga_hsync = hsync;
assign vga_vsync = vsync;
always @ (posedge clk_pix)
begin
    if(de)
        begin
            if (BeeSpriteOn==1)
                begin
                    vga_r <= (palette[(dout*3)])>>4; // RED bits(7:4) from colour palette
                    vga_g <= (palette[(dout*3)+1])>>4;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

```

If `de` equals 1 (`VGA_Timing` is in the visible pixel area) and `BeeSprite` is active then the RGB colour value (`dout`) is retrieved from palette and sent to the VGA output

If `de` equals 1 (`VGA_Timing` is in the visible pixel area) and `BeeSprite` is not active then the background colour (reg [7:0] `COL = 0`;) is sent to the VGA output

>>4 changes the 8 bit values retrieved from palette into a 4 bit values:

		Hex	Dec	Binary		Colour 24 Bit
8 Bit Values For Colour 46 From pal24bit.mem	RED	E4	228	1110	0100	
	GREEN	A9	169	1010	1001	
	BLUE	0D	13	0000	1101	

		Hex	Dec	Binary		Colour 12 Bit
8 Bit Binary Values Shifted Right x 4	RED	E	14	0000	1110	
	GREEN	A	10	0000	1010	
	BLUE	0	0	0000	0000	

If de equals 0 (VGA_Timing is outside the visible pixel area, i.e. in the Blanking Time area) then 0 RGB values must be sent to the VGA output

02 BeeSprite.v module

```
//-----  
// BeeSprite.v Module  
// Digilent Basys 3  
// Bee Invaders Tutorial_2  
// Onboard clock 100MHz  
// VGA Resolution: 640x480 @ 60Hz  
// Pixel Clock 25.2MHz  
//-----  
`timescale 1ns / 1ps  
  
// Setup BeeSprite module  
module BeeSprite(  
    input wire clk_pix,  
    input wire [9:0] sx,  
    input wire [9:0] sy,  
    input wire de,  
    output reg [1:0] BeeSpriteOn, // 1=on, 0=off  
    output wire [7:0] dataout  
);
```

This module is linked to the "Top.v" module in order that the character can be drawn on the screen

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

This links to the "BeeRom.v" module (a single port ROM) to load the data for the Bee character. address is used as a counter (0 - 917) to define dataout in "BeeRom.v" (memory_array[address])

```

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

// check if sx,sy are within the confines of the Bee character
always @ (posedge clk_pix)
begin
    if(de)
        begin
            if(sx==BeeX-2 && sy==BeeY) // sx=295
                begin
                    address <= 0; // 1st Entry: address = 0
                    BeeSpriteOn <=1;
                end
            if((sx>BeeX-2) && (sx<BeeX+BeeWidth-1) && (sy>BeeY-1) && (sy<BeeY+BeeHeight)) // sx = 296 to 329 = 33 Entries
                begin
                    address <= (sx+2-BeeX) + ((sy-BeeY)*BeeWidth); // 2nd Entry: address = 296 + 2 - 297 = 1
                    BeeSpriteOn <=1;
                end
            else
                BeeSpriteOn <=0;
        end
    end
endmodule

```

BeeX & BeeY hold the x, y coordinates for the Bee character and BeeWidth & BeeHeight hold the size of the character

When the "VGA_Timing" x, y coordinates are in the visible pixel area (de equals 1) a check is made to see when BeeX -2 and BeeY have been reached

At this point address is set to 0 and the first data value is obtained from "BeeRom.v" (a single port ROM that has a latency of at least 1 clock cycle). This is the reason why we retrieve the "BeeRom.v" data 2 pixels early to compensate for the clock cycle delay. BeeSpriteOn is then set to 1 in order that the "Top.v" module can start to draw the Bee on the screen

Thereafter and as long as the "VGA_Timing" x, y coordinates are within the boundaries of the Bee character, the data is retrieved and linked back to the "Top.v" module via the dataout wire

When the "VGA_Timing" x, y coordinates are outside of the Bee boundaries BeeSpriteOn is set to 0

03 BeeRom.v module

```
//-----  
// BeeRom.v Module  
// Single Port ROM  
// Digilent Basys 3  
// Bee Invaders Tutorial_2  
// Onboard clock 100MHz  
// VGA Resolution: 640x480 @ 60Hz  
// Pixel Clock 25.2MHz  
//-----  
`timescale 1ns / 1ps  
  
// Setup BeeRom module  
module BeeRom(  
    input wire [9:0] address, // (9:0) or 2^10 or 1024, need 34 x 27 = 918  
    input wire clk_pix,  
    output reg [7:0] dataout // (7:0) 8 bit pixel value from Bee.mem  
);
```

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

```
(*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
```


address (a counter) contains the address in the "Bee.mem" file provided by the "BeeSprite.v" module

dataout will contain the hex value obtained from the "Bee.mem" file which is sent back to "BeeSprite"

When the "Bee.mem" file is read the data is stored in memory_array, an 8 bit register which holds the 918 pixel colour data values for the Bee character

Notice this line of code:

```
(*ROM_STYLE="block"*) reg [7:0] memory_array [0:917];
```

This tells Vivado to store the data in the style of Block RAM (BRAM)