Lab 6: Digital Photo Frame and VGA Game

Submission Due Dates:

Demo: 2023/11/14 17:20

Source Code: 2023/11/14 18:30

Report: 2023/11/19 23:59

Objective

1 Getting familiar with the VGA display, block RAM, and other IOs on the FPGA demo board.

Action Items

In Part A, you will implement a digital photo frame with fancy transition effects on the VGA display. In Part B, you will implement a puzzle game with a photo on the VGA display. Pick your favorite image(s) for both parts. However, inappropriate images are forbidden. Refer to the lecture notes on the way to use **PicTrans.exe** to generate your own .coe file from an image**.** We also provide the template and constraint files for both Part A and Part B. **Pictrans.exe** are in **SampleCode_VGA** subfolder.

A. lab6_1.v (30%)

Design a VGA controller that can scroll your image from the right to the left or from the left to the right over time, mirror the image orientation, and enlarge the image size.

a. IO list:
   - Inputs: clk, rst, en, dir, vmir, hmir, enlarge
   - Output: vgaRed, vgaGreen, vgaBlue, hsync, vsync

b. **rst**: the asynchronous positive reset: the VGA display will show the image at the origin position after reset.

c. **en**: only affect the scrolling of the image. **vmir**, **hmir**, **enlarge** should have effect no matter **en** == 1'b0 or **en** == 1'b1.

d. If **en** == 1'b0: hold the image still on the screen.

   If **en** == 1'b1:
   - If **dir**== 1'b0:

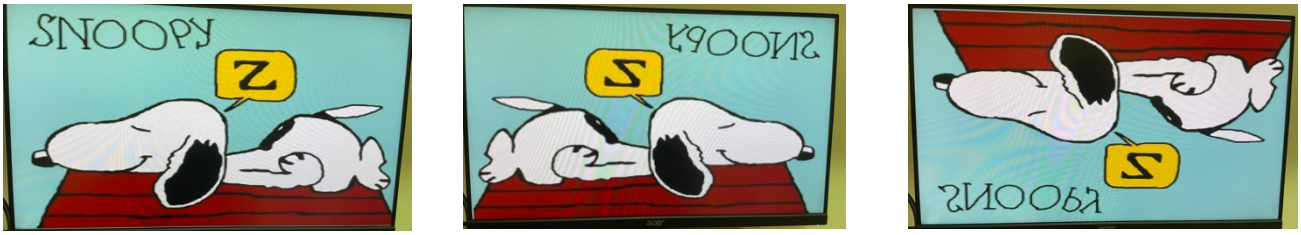
- the image scrolls from the right to the left at the frequency of 100MHz divided by 2^{22}.

- If **dir**== 1'b1:
  - the image scrolls from the left to the right at the frequency of 100MHz divided by 2^{22}.

e. If **vmir**==1'b1: the image is mirrored vertically.

f. If **hmir**==1'b1: the image is mirrored horizontally.

g. If both **vmir** and **hmir** ==1'b1, the image is mirrored **both** vertically and horizontally.



original mirrored horizontally mirrored vertically

h. If both **vmir** and **hmir** ==1'b0, the image remains its original orientation.

i. If **enlarge**==1'b1: the image should be magnified from the center of the screen. You can decide your own magnification scale. The scale has to be large enough to observe. For example, you may scale up the image 2 times in both width and height.

original

magnified

j. When the image is mirrored or enlarged, the scrolling should function continuously, and the scrolling direction should remain the same.

k. You have to use the following template for your design:

```
module lab6_1 (
  input clk,
  input rst,
  input en,
  input dir,
  input vmir,
  input hmir,
  input enlarge,
  output [3:0] vgaRed,
  output [3:0] vgaGreen,
  output [3:0] vgaBlue,
  output hsync,
  output vsync
  );
  // add your design here
endmodule
```

IO Connection:

clk	connected to W5
rst	connected to U18 (BTNC)
en	connected to V17 (SW0)
dir	connected to V16 (SW1)
vmir	connected to W16 (SW2)
hmir	connected to W17 (SW3)
enlarge	connected to W15 (SW4)
vgaRed	connected to pin N19, J19, H19, G19
vgaGreen	connected to pin D17, G17, H17, J17
vgaBlue	connected to pin J18, K18, L18, N18

hsync	connected to pin P19
vsync	connected to pin R19

Demo video:
https://youtu.be/EapIMoURjOI

B. lab6_2.v (70%)

Design a VGA control for the sliding puzzle game. Your image is divided into 4×4 blocks. All blocks are shuffled from the original position. Some of the blocks are rotated 180 degrees. Each block is mapped to a corresponding key as shown below.

You can swap two blocks by pressing their keys at the same time. You can rotate a block by 180 degrees by pressing its corresponding key and the left shift key at the same time.

Example: an initial position

Example: a solved puzzle

a. IO list:

- Inputs: clk, rst
- Inout: PS2_CLK, PS2_DATA
- Output: vgaRed, vgaGreen, vgaBlue, hsync, vsync, pass

b. **rst**: The VGA display will show the puzzle at the initial state after the asynchronous positive reset. You may set the initial position of each block and choose the rotation blocks (**at least 3 blocks** rotated 180 degrees) as you like.
c. When two of the keys are pressed simultaneously, the two corresponding blocks will swap.
d. When pressing the corresponding key and the left shift key, the specific block will rotate 180°.
e. If **hint**==1'b1, the screen will demonstrate the correct puzzle solution. All the keyboard inputs are disabled at this moment.
f. When all the blocks are in the correct position with the correct rotation, the **pass** is set to high, and LED0 lights up. All the blocks can not move any further until a reset.

Demo video:
https://youtu.be/Q6BeFyGENGk

g. You have to use the following template for your design:

```
module lab6_2 (
  input clk,
  input rst,
  inout PS2_CLK,
  inout PS2_DATA,
  output [3:0] vgaRed,
  output [3:0] vgaGreen,
  output [3:0] vgaBlue,
  output hsync,
  output vsync,
  output pass
);
  // add your design here
endmodule
```

IO Connection:

clk	connected to W5
rst	connected to U18 (BTNC)
PS2_CLK	connected to C17

PS2_DATA	Connected to B17
vgaRed	connected to pin N19, J19, H19, G19
vgaGreen	connected to pin D17, G17, H17, J17
vgaBlue	connected to pin J18, K18, L18, N18
hsync	connected to pin P19
vsync	connected to pin R19
pass	connected to U16 (LED0)
hint	connected to V17 (SW0)

Questions and Discussion

Please answer the following questions in your report.

A. If we want to turn the image with a special effect of the negative film (which means each pixel is complemented in color, like the example shown in below, the word 'POPCAT' is original white, but after complemented, it is shown in black), how would you modify your design of lab6_1?

positive film

negative film

B. Suppose we want to create a VGA game with animations, how can you design the frame buffer and let the item on the monitor moves? (hint: you can use two frame buffers)

C. Our FPGA equips with the BRAM of only 1800 Kbits, which a 640×480 image cannot fit in. If we want to implement a video game, apart from storing a smaller image (e.g., 320×240) like we did in this lab, please give at least 2 possible methods to reduce the BRAM usage.

Guidelines for the report

Refer to the guidelines in the report template (or in the previous lab assignments).

Grading policy (subject to change): Part (A): **35%**; Part (B): **50%**; Part (C): **10%**; (D): **5%**

Attention

- ✓ DO NOT include vga_controller, onepulse, debounce into the .v files.
- ✓ You must hand in the file named **lab6_1.v** and **lab6_2.v**. DO NOT hand in any compressed ZIP filed, which will be considered an incorrect format.
- ✓ If you create several modules for your design, merge them all into one Verilog file.
- ✓ You should also hand in your report as **lab6_report_StudentID.pdf** (i.e., lab6_report_111456789.pdf).
- ✓ You should be able to answer questions of this lab from TA during the demo.
- ✓ You need to prepare the bitstream files before the lab demo to make the demo process smooth.
- ✓ Feel free to ask questions about the specification on the EECLASS forum.