Lab 5 Advanced: Digital Photo Frame and VGA Game

Submission Due Dates:

Demo:	2024/11/19 17:20
Source Code:	2024/11/19 18:30
Report:	2024/11/24 23:59

Objective

1 Gaining a working knowledge of the VGA display, block RAM, and other I/O components on the FPGA demo board.

Action Items

In **Part A**, you will implement a digital photo frame with various fancy transition effects on the VGA display. In **Part B**, you will create a puzzle game on the VGA display. Pick your favorite images for both parts; however, ensure all images are appropriate. Refer to the lecture notes for guidance on using **PicTrans.exe** to convert your chosen image into a **.coe** file. Templates and constraint files for both Part A and Part B are provided. **Pictrans.exe** can be found in the **SampleCode_VGA** subfolder.

A. lab5_1.v (30%)

Create a VGA controller that can perform the following functions:

- Scroll an image across the screen, either from the upper-right to the lower-left or from the upper-left to the lower-right, over time.
- Mirror the image orientation.
- Enlarge the image size.

a. IO list:

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

b. IO Connection:

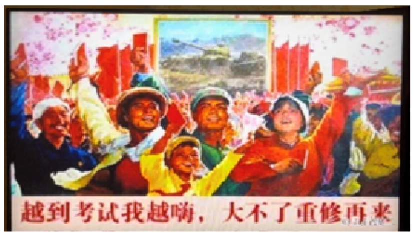
clk	connected to W5
rst	connected to U18 (BTNC)
en	connected to V17 (SW0)
dir	connected to V16 (SW1)
vmir	connected to W16 (SW2)
hmir	connected to W17 (SW3)
enlarge	connected to W15 (SW4)

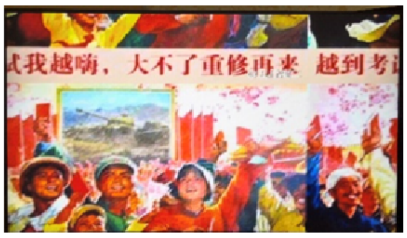
vgaRed	connected to pin N19, J19, H19, G19
vgaGreen	connected to pin D17, G17, H17, J17
vgaBlue	connected to pin J18, K18, L18, N18
hsync	connected to pin P19
vsync	connected to pin R19

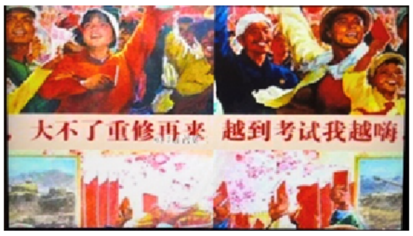
b. Upon reset (**rst**: the asynchronous positive reset), the VGA display will show the image at the origin position.

c. The enable switch (**en**) controls the scrolling of the image.

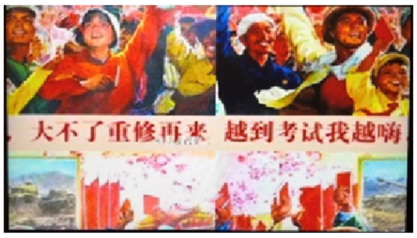
- If **en** == 1'b0: the image holds still on the screen.
- If **en** == 1'b1:
  - If **dir**== 1'b0, the image **scrolls from the upper-right to the lower-left** at the frequency of 100MHz divided by 2^{22}.

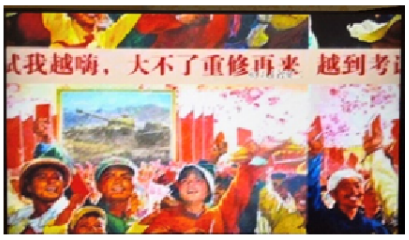


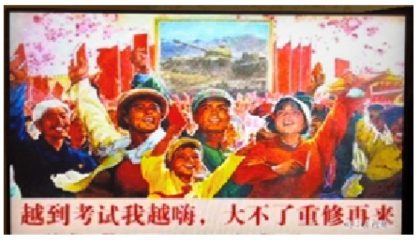




  - If **dir**== 1'b1, the image **scrolls from the lower-left to the upper-right** at the frequency of 100MHz divided by 2^{22}.

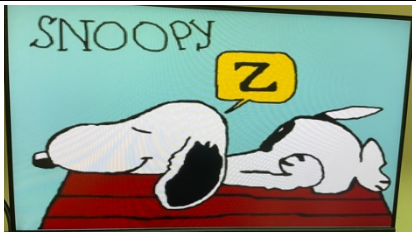




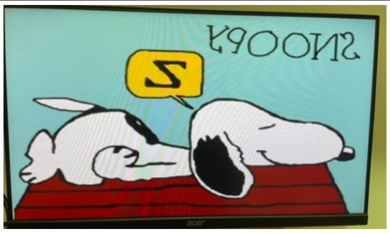


d. The vertical mirroring switch (**vmir**) and horizontally mirroring switch (**hmir**):

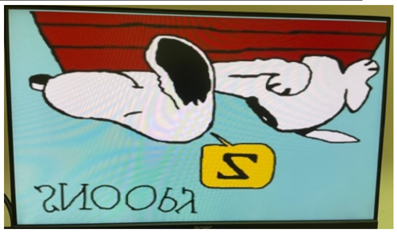
- If **vmir**==1'b1, the image is mirrored vertically.
- If **hmir**==1'b1: the image is mirrored horizontally.
- If both **vmir** and **hmir** are equal to 1'b1, the image is mirrored **both** vertically and horizontally.



original



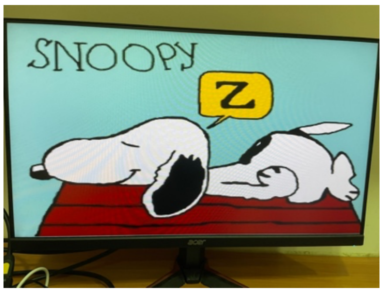
mirrored horizontally



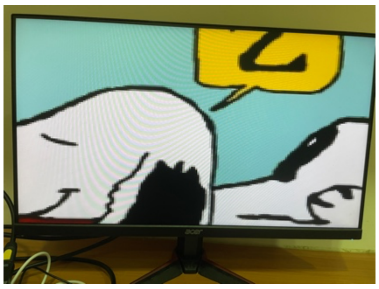
mirrored vertically

- If both **vmir** and **hmir** are 1’b0, the image remains in its original orientation.

e. The enlarging switch (**enlarge**):
   - If **enlarge**==1’b1, the image will be magnified **from the center of the screen**. You can choose your own magnification scale, but it must be large enough to be clearly noticeable. For example, consider scaling the image by a factor of in both width and height.



original


magnified

f. The signals **vmir**, **hmir**, and **enlarge** should take effect regardless of whether **en** is 1’b0 or 1’b1.

g. When the image is mirrored or enlarged, scrolling should function uninterrupted, maintaining the same scrolling direction.

h. Use the provided template for your design:

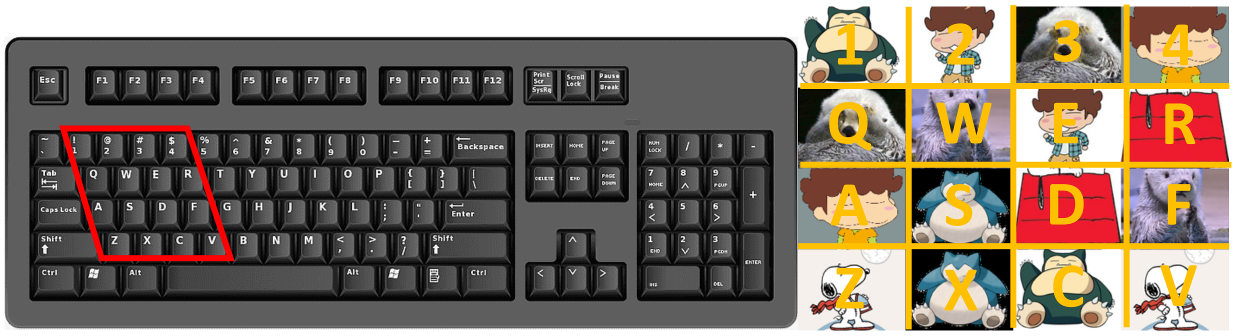
```
module lab5_1 (
  input wire clk,
  input wire rst,
  input wire en,
  input wire dir,
  input wire vmir,
  input wire hmir,
  input wire enlarge,
  output reg [3:0] vgaRed,
  output reg [3:0] vgaGreen,
  output reg [3:0] vgaBlue,
  output reg hsync,
  output reg vsync
  );
  // add your design here
endmodule
```

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

B. lab5_2.v (70%)

Design a VGA controller to implement a memory puzzle game with the following features:

- Game States: The game operates through four states: **Init**, **Show**, **Game**, and **Finish**.
- Display Grid: The screen is divided into a 4×4 equal-sized grid, with each cell containing an image.
- Image pairing: Each image appears twice on the board, with some images mirrored vertically to add variety.



You can flip two blocks at a time by pressing their corresponding keys. If the two flipped blocks reveal matching images with the correct rotation, they will remain visible. However, if the blocks do not match in both image and rotation, they will turn back to black when you press the **Enter** key.

To mirror a block vertically, press its corresponding key along with the **Left Shift** key. This mirrored block will reveal itself and then return to black after pressing the **Enter** key.

a. IO list:
   - Inputs: clk, rst, hint
   - Inout: PS2_CLK, PS2_DATA
   - Output: vgaRed, vgaGreen, vgaBlue, hsync, vsync, pass

b. IO Connection:

clk	connected to W5
rst	connected to U18 (BTNC)
start	Connected to T17 (BTNR)
hint	connected to V17 (SW0)
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)

c. Upon reset (**rst**: the asynchronous positive reset), the VGA display will enter the **Init** state, showing a completely black screen.

b. The **start** button: Press to control the gameplay.

c. You may decide each block's initial position and select which blocks will be mirrored. **At least three blocks** should be mirrored vertically.

d. **Init state**: All blocks are displayed as black. Pressing the **start** button will transition to the **Show** state.

e. **Show state**: The initial block positions and rotations are displayed the screen. Pressing the **start** button will transition to the **Game** state.

f. **Game state**:
   - Pressing any two keys simultaneously will reveal the images in the corresponding blocks. Afterward, pressing the **Enter** key will turn unmatched blocks back to black.
   - Pressing a block’s corresponding key with the **Left Shift** key will mirror that block vertically and display the mirrored image. Afterward, pressing the **Enter** key will turn that block back to black.
   - If the switch **hint == 1'b1**, the screen will display all the images in their correct direction. All the keyboard inputs will be disabled at this moment.
   - Once all the blocks are correctly matched and flipped over, the game will automatically transition to the **Finish** state.

g. **Finish state**:

- The **pass** signal is set to high, and **LED0** lights up.
- All the blocks are locked and cannot be flipped or mirrored.
- Pressing the **start** button will transition to the **Init** state.

h. The gameplay actions for matching and mirroring will always involve pressing two keys simultaneously, and no other key combinations will be considered. The **Enter** key will always be pressed alone.

i. There are two methods for creating a screen with 16 image blocks. The first method involves creating a single composite image that contains all 16 smaller images. The second method stores 8 individual images and implements a converter to display them on the screen in the desired arrangement.

Demo video: https://www.youtube.com/watch?v=P89Zc01M0Xo

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

```
module lab5_2 (
  input wire clk,
  input wire rst,
  input wire start,
  input wire hint,
  inout wire PS2_CLK,
  inout wire PS2_DATA,
  output reg [3:0] vgaRed,
  output reg [3:0] vgaGreen,
  output reg [3:0] vgaBlue,
  output reg hsync,
  output reg vsync,
  output reg pass
);
  // add your design here
endmodule
```

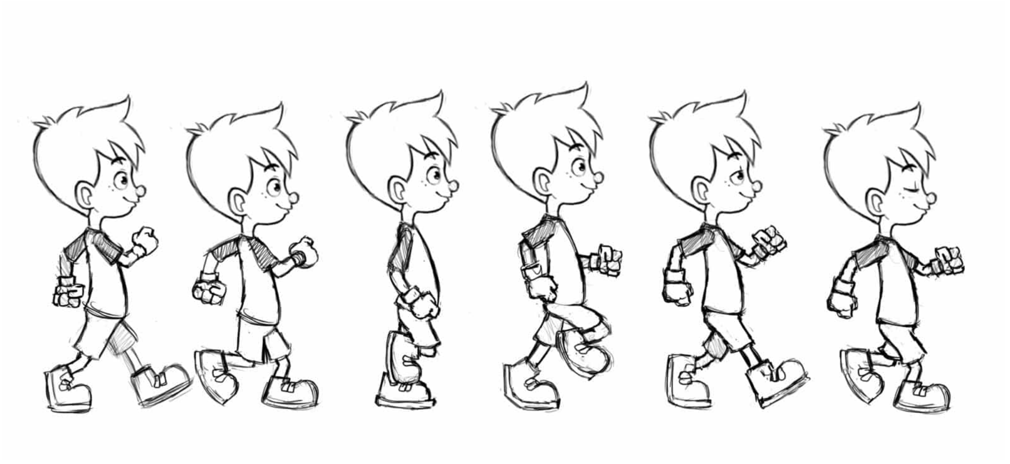
Questions and Discussion

Please answer the following questions in your report.

A. 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) as we did in this lab, please give at least 2 possible methods to reduce the BRAM usage.

B. Verilog supports 2D arrays. Briefly explain how you may use it in your design of lab5_2.

C. Suppose that you are given Picture A below and are asked to make an animation of the little boy walking in the middle of the screen. How should you implement it? Write down your pseudo code or concept with an explanation.


(Picture A)

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

- ✓ Please use the same template keyboard_decoder, KeyboardCtrl, Ps2Interface in lab4.
- ✓ DO NOT include vga_controller, onepulse, debounce, keyboard_decoder, KeyboardCtrl, Ps2Interface into the .v files.
- ✓ You must hand in the file named **lab5_1.v** and **lab5_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 **lab5_report_StudentID.pdf** (i.e., lab5_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.