Lab 7: Digital Photo Frame

Submission Due Dates:

Demo: 2021/12/07 17:20

Source Code: 2021/12/07 18:30

Report: 2021/12/12 23:59

Objective

1 Getting familiar with finite state machines (FSMs) in Verilog.

2 Getting familiar with the control of VGA display, keyboard and other I/Os on the FPGA demo board.

Action Items

In this lab, you will implement a digital photo frame with fancy transition effects on the VGA display. Pick an image you like. Use **PicTrans.exe** to generate your own .coe file**. Make sure it is an appropriate image**.

1 VGA controller

A. lab7_1.v (60%)

Design a VGA controller that makes your image scrolls up or down over time and change color.

a. IO list:

- Inputs: clk, rst, en, dir, nf
- Output: vgaRed, vgaGreen, vgaBlue, hsync, vsync

b. **rst**: the positive-edge-triggered reset, the VGA display will show the image at the origin position after triggered.

c. If **en** == 1'b0: hold the image on the screen unchanged.

d. If **en** == 1'b1:

- If **dir**== 1'b0:
  - the image scrolls up at the frequency of 100MHz divided by 2^{22}.

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

e. If nf==1'b0: show the original colors of the image.

f. If nf==1'b1: change the image into negative film (which means the colors reversed into their complementary colors).

Positive film

negative film

Demo video:
https://drive.google.com/file/d/19ifjoKsAzJ9GGWpDnlOYr8_TI4kDMaEm/view?usp=sharing

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

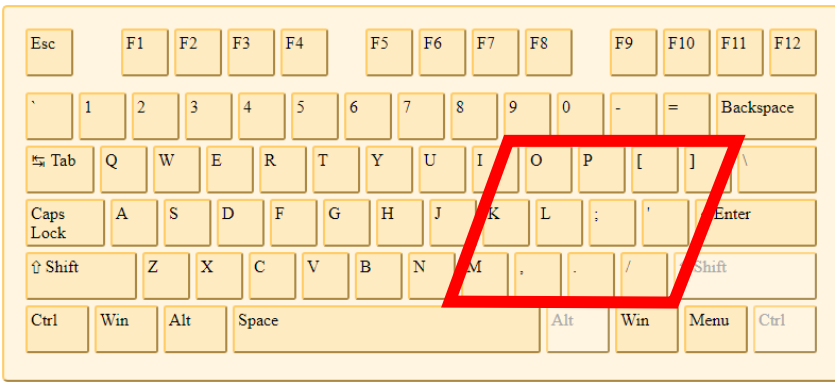
```
module lab7_1 (
  input clk,
  input rst,
  input en,
  input dir,
  input nf,
  output [3:0] vgaRed,
  output [3:0] vgaGreen,
  output [3:0] vgaBlue,
  output hsync,
  output vsync
  );
  // add your design here
endmodule
```

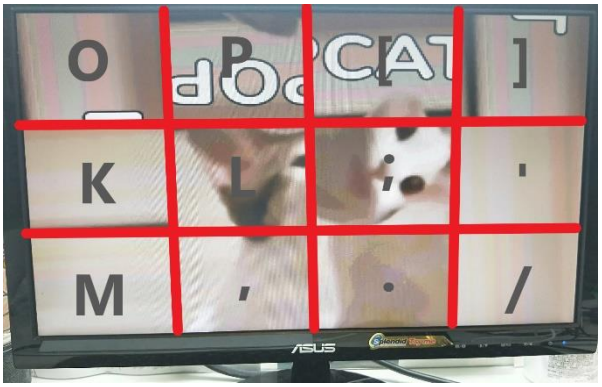
IO Connection:

clk	connected to W5
rst	connected to U18 (btnC)
en	connected to V17
dir	connected to V16
nf	connected to W16
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. lab7_2.v (40%)

Design a VGA game controller that divides your image into 4*3 blocks. You can rotate each block by the keyboard. The block turns 90° when you push the corresponding button. The following figures show the key mapping.





a. IO list:
   - Inputs: clk, rst, en, hold
   - inout: PS2_CLK, PS2_DATA
   - Output: vgaRed, vgaGreen, vgaBlue, hsync, vsync, pass

b. **rst**: The VGA display will show the image at the origin position after the positive-edge-triggered reset. You may set the initial direction of each block as you like.

c. If **hold** == 1'b1: The screen shows the correct image as *the hint mode*. The block can not be rotated in this mode.

d. If **hold** == 1'b0: The screen goes back to *the game mode*.

- Press the key directly:

  Turn the corresponding block 90° clockwise.

- Press the key with the shift button:

  Turn the corresponding block 90° counterclockwise. The left-shift and right-shift buttons should both work.

e. When all the blocks rotate to the correct direction, set the pass signal to 1'b1 to indicate that the mission is accomplished. Also, disable the block rotation so that no block can be rotated further.

Demo video:
https://drive.google.com/file/d/1Hryxc4pkaLheuV73azLBvFz_uY58bB7k/view?usp=sharing

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

```
module lab7_2 (
  input clk,
  input rst,
  input hold,
  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)
hold	connected to T17(btnR)
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

Attention

- DO NOT copy-and-paste code segments from the PDF materials. Occasionally, it will also paste invisible non-ASCII characters and lead to hard-to-debug syntax errors.
- You should hand in the file named **lab7_1.v** and **lab7_2.v**. If you create several modules for your design, merge them all into one Verilog file. Don't include Vga_controller, KeyboardDecoder, Onepulse, Debounce into lab7.v.
- You should also hand in your report as **lab7_report_StudentID.pdf** (i.e., lab7_report_109000000.pdf).
- Please do not hand in any compressed files, which will be considered as an incorrect format
- You should be able to answer questions of this lab from TA during the demo.
- You need to generate bitstream before the demo.