Tel Aviv University

The Iby and Aladar Fleischman Faculty of Engineering

Electronics - Laboratory (3)

Experiment FPGA 2 – I/O Interfaces

Written by: Konstantin Berestizshevsky

& Maayan Tamari

Lab Goals

The purpose of this lab is to cover the following topics:

1. Asynchronous reset – advantage and disadvantages
2. PS2 interface - serial data transfer, transmission-frame structure, not a constantly toggling clock, how to identify an event of keypress.
3. Video Graphics Array (VGA) interface – synchronization signals, color space encoding, graphics representation.

Submission Guidelines

The lab consists of 2 tasks. In each task you'll find what to submit in the preliminary report and what to submit in the final report. The preliminary report is where you will need more effort. Namely, you need to design and test all the sources of all the tasks and to submit them as a preliminary report.

Prepare all tasks and perform synthesis, implementation, and generating a bit file before the lab session. Keep in mind that before creating the bit file, you have to fix all errors, critical warnings and warnings in both synthesis and implementation.

Then, during the lab session, you will only have to deal with debugging according to the instructor's comments. Otherwise, you might not have enough time during the lab session to finish all stages and receive the instructor's approval of your design.

1. Submit a preliminary report (PDF format), as well as the XDC constraints file, and all the *.v files you have edited. All zipped together in a zip file named EE3_PRE_FPGA2 <ID1> <ID2>
2. Submit a final report (PDF file only), the XDC constraints file, and all the *.v and *.xdc files you have edited. All zipped together in a zip file named EE3_POST_FPGA2 <ID1> <ID2>
3. Each waveform must be annotated by drawing arrows that point to important signals and explaining these signals.
4. Each Verilog source/test-bench header must have both of your names in it:

```
1  `timescale 1ns/10ps
2  //////////////////////////////////////////////////////////////////
3  // Company:          Tel Aviv University
4  // Engineer:         Your name
```

5. At home, you can work with the Vivado Web-Pack, which allows you to design and simulate RTL modules. Only during the lab session, the instructor will test your design on an actual FPGA chip.

Project creation

First, create "lab2" project. All the tasks in this lab will be performed under the same project, however you need to use a different constraints-file for each task. To this end, create two *.xdc files in the project, one for each task, with the unused file fully commented out. Set the used file as target constraints file (right-click on the used file).

Task 1 – PS2 interface for keyboard input

In this task, you will implement a PS2 serial interface to allow the FPGA to read input values from an external keyboard. The modules for you to implement are:

1) **Ps2_Interface** – with 3 inputs (PS2Clk, rstn, PS2Data) and 4 outputs (number[3:0], keyPressed, keyReleased, errorStrobe). The module interprets the packets arriving via PS2Clk and PS2Data inputs from the keyboard and outputs the most recently received 8-bit *scancode* interpreted to the number written on the key (0-9) as well as one of the following outputs as a single cycle high pulse:
   a. *keyPressed* – in case a make-code packet arrived
   b. *keyReleased* – in case 2 packets arrived (first one contained 0xF0 and the 2nd one contained the scancode of the key that was release)
   c. *errorStrobe* – in case that the packet contained error (parity check failed)

   All packet containing the extended key qualifier ("0xE0") are to be ignored.
   You can create a sub-module for the interpretation process if you prefer.
2) **Ps2_Display** – with 6 inputs (clk, rstn, keyPressed, keyReleased, errorStrobe, number[3:0]) and 4 outputs (seg[6:0], an[3:0], dp, led[2:0]). The module drives the 7-segment display and the user LED to provide visual feedback of the input inspected by the Ps2_Interface module. The functionality of this module is:
   a. **7-seg outputs (seg[6:0],an[3:0],dp).** Display up to four numbers according to the last keys pressed:
      - After reset is pressed, all four digits should be off.
      - When the first key is pressed, then only the rightmost digit should display a number.
      - When the next key is pressed (assuming no reset in between), it should appear on the rightmost digit, while the first one shifts one digit to the left, and so on.
      - If more than four keys were pressed since the last time reset was pressed, the left shifting should continue so that the last four keypresses are displayed.

      No need to handle pressing down more than one key simultaneously.
      You should adjust the code of the Seg_7_display module from the previous lab to have most of the functionality.
      The display should respond to the keyReleased signal.
   b. **User LED outputs. led[0], led[1], led[2]:** 200 milliseconds-long pulses that starts at the beginning of the received *keyPressed*, *keyReleased* and *errorStrobe* pulses, respectively.
3) **Ps2_Top** – a top level module that instantiates Ps2_Insterface and Ps2_Display and connects to the FPGA pins. Refer to Figure 1 for a high-level diagram of the design. The keys pressed on a keyboard are interpreted by the Ps2_Interface into the keyPressed and keyReleased pulses (indicate the events of pressing or holding

a key) and into the scancode bus (the value of the key being pressed or held). The pulses and the interpretation of the scancodes into numbers are visualized through Ps2_Display module.

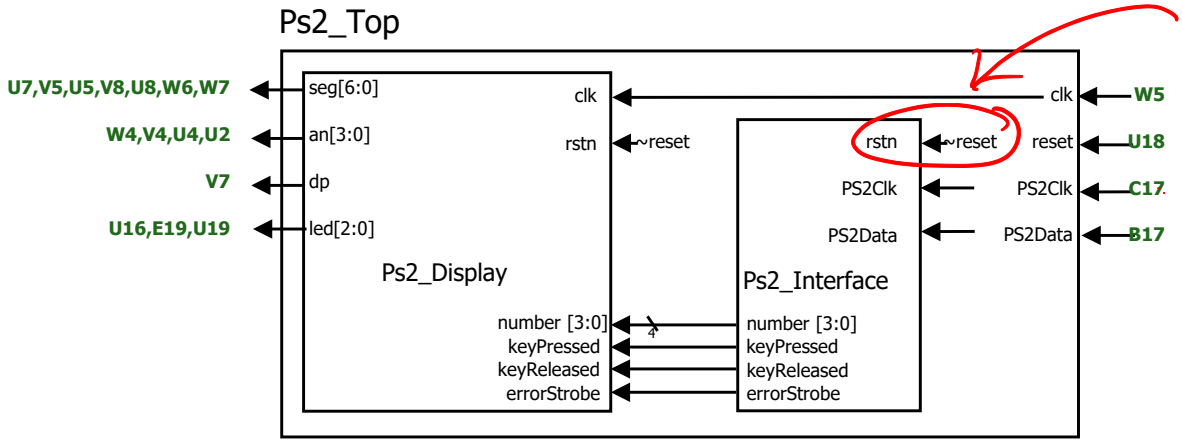

Figure 1 - High level diagram of the design required in task 1. The green labels are the FPGA pin name whereas the black labels are the IO ports of the Verilog sources.

Important Design Notes:

1. There is an easy way of implementing the keyboard interface using a 22-bit shift register in your design.
2. Don't always wait to receive a 22-bit long transmission. Instead, evaluate the received bits every 11 clock cycles.
3. The keyboard-clock is not toggling all the time, but only during the transmission. Hence, waiting until receiving the whole packet will bring you to a non-toggling clock and you won't be able to perform a useful logic because the clock will be idle by this moment. Try analyzing the packet before the last cycle (before the receiving the stop bit).
4. As a part of the preliminary report, you must perform rigorous simulations to ensure a correct timing. Make sure that the "keyPressed", "keyReleased" pulses are generated correctly and that they last only 1 clock cycle.
5. You can base the top module simulation on the interface simulation. The inputs are almost the same. You will need to simulate two clocks, one for the Ps2 clock and one for the display module. It is enough for the purpose of this simulation, to define the clocks as we did in previous simulations, by writing-
   always #5 clk = ~clk;
   And similarly for the Ps2 clock with the proper delay.
6. Make sure to ignore the 0xE0 packets.
7. For this design, you should use the keyboard clock as an input to your module. Disregard what the Basys3 Board Manual says .

During the lab session:

The keyboard that we will use is a small numeric keyboard.

Upload your project to google drive and share the project with the instructor that is present in your online room. The instructor will reply to you with comments on your design. If the instructor tells

you to proceed, then you can proceed. Otherwise, you will be asked to resolve the existing problems and to resend the new version to the instructor.

Submit:

In preliminary report – submit the files:
Ps2_Interface.v, Ps2_Interface_tb.v, Ps2_Display.v, Ps2_Top.v, Ps2_Top_tb.v
task1.xdc.
Provide a screenshot of the simulation and explain its principle signals behavior by pointing with an arrow to these signals' transitions and annotating them.
Note: you need to simulate only your Ps2_Interface.v nothing more.

In the final report – explain which problems appeared during the lab and how did you overcome these problems. Re-submit the corrected files.

Task 2 – VGA interface for screen output

In this part of the lab, you will design a VGA interface to output graphics to the computer monitor connected to the Basys3 board. So far, we were limited to either the 7-segment display or the LEDs. In this lab, we expand this functionality to allow graphical images to be displayed from the FPGA board.

We will examine the functionality of your design by displaying an image of your choice (preferably not a flag). The image you create can be of whatever you like. It has to include the usage of at least two colors. The minimum requirement regarding shapes is simple vertical or horizontal lines. Sophisticated designs including shapes or an interesting usage of colors will receive a bonus.

Figure 2 depicts a high-level diagram of a design required in this task.

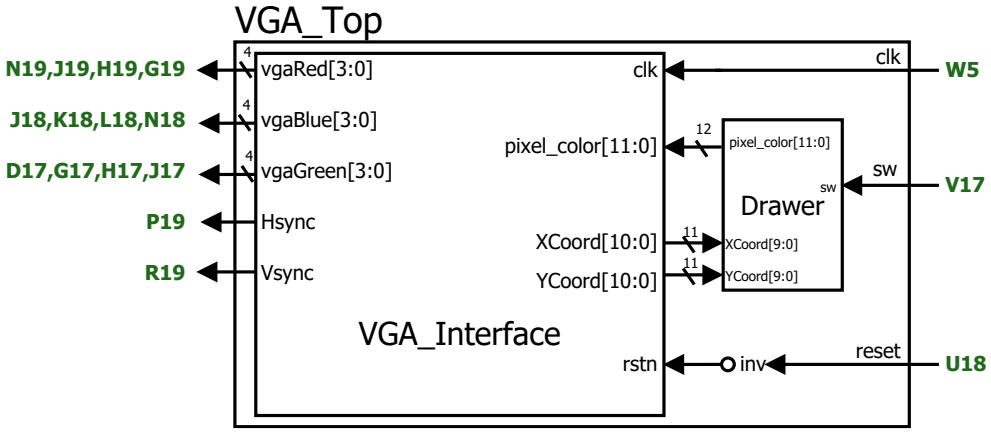

Figure 2 - High level diagram of the design required in task 2. The green labels are the FPGA pin name whereas the black labels are the IO ports of the Verilog sources.

The modules that you are required to implement are:

1) **VGA_interface** – with 3 inputs: clk, rstn, pixel_color; and 7 outputs: vgaRed, vgaGreen, vgaBlue, Hsync, Vsync, XCoord, YCoord. The VGA interface

takes as an input a 12-bit signal pixel_color (which contains 4-red-channel bits, 4-green-channel bits, 4 blue-channel bits) which represents a color of a single pixel, and outputs this color to the VGA-output ports. The VGA interface will generate 50MHz pixel-clock internally. The outputs vga* and *sync of VGA interface must respect the VGA standard for 800x600 resolution (see theoretical background). It will also output the current horizontal coordinate (0-1039) and vertical coordinate (0-665).

2) **Drawer** – 3 inputs: sw (1 bit), XCoord (11 bits), YCoord (11 bits). The output is the 12-bit encoding of the current color to be shown on the screen. This is a combinational module which decides which color is to be sent to the current pixel on the screen. If sw=0, then for every coordinate, the output must encode the black color. If sw=1 then according to the current x,y coordinates, color should be generated **such that an image of your choice will be shown on the monitor. Sophisticated images will receive a bonus.**
3) **VGA_Top** – a top level module comprised of the VGA_Interface and the Drawer modules. It receives clock and reset signals, as well as a single switch (SW0) from the board. It outputs the VGA control signals using VGA_Interface. If SW0 is high then an image should be seen on the VGA monitor, otherwise the monitor should show a solid black picture.

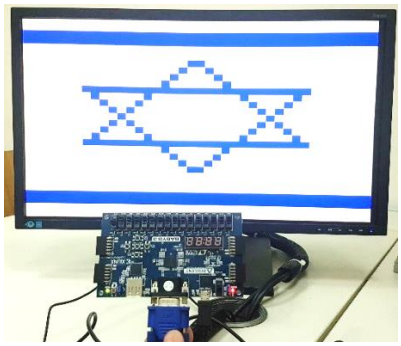

Figure 3 - Example of an image displayed on VGA monitor. The pattern is generated by the Drawer module.

Guidance: The VGA interface can be implemented as follows:

1. Warning: the 72Hz screen refresh rate may not be supported by some of the monitors!
2. Use 2 counters to store the values of *hcount* and *vcount*
3. On the rising edge of the pixel clock, increment the *hcount*. Increment *vcount* when *hcount* has reached the end of the row.
4. Generate the *Hsync* signal based on the value of *Hcount*. *Vsync* is generated in a similar fashion. Refer to the theoretical background for timing diagrams.
5. Generate a signal to determine whether the pixel is in the visible region as illustrated in the regions diagram in theoretical background
6. When in the visible region, output the pixel color value {R, G, B}, otherwise when in the blanking region, output {0, 0, 0}.
7. The output signals {*Hsync*, *Vsync*, vgaRed, vgaGreen, vgaBlue} should be defined as flip-flops to ensure no combinational logic delays will interfere with the output display.
8. A good advise would be to downsample the xcoord and ycoord by a factor of 8, brining the resolution to 100x75, which is easier to handle.

Notes:

1. The 50Mhz pixel clock can be generated from the 100Mhz system clock.
2. Do not turn on user-LEDs in response to turning on the user-switches.

3. Perform a rigorous simulation to ensure precise timing, which must perfectly correspond to the timing diagrams in the theoretical background.

During the lab session:

Upload your project to google drive and share the project with the instructor that is present in your online room.

The instructor will reply to you with comments on your design. If the instructor tells you to proceed, then you can proceed. Otherwise, you will be asked to resolve the existing problems and to resend the new version to the instructor.

Take a photo of a correct functioning of the monitor.

Submit:

In preliminary report –

- submit the files:

VGA_Interface.v, VGA_Interface_tb.v, Drawer.v, VGA_Top.v, task2.xdc.

- Provide a screenshot of the simulation and explain its principle signals behavior by pointing with an arrow to these signals' transitions and annotating them.
- Describe the image you are trying to create, preferably add a simple sketch of the image.

In the final report –

- Explain which problems appeared during the lab and how did you overcome these problems.
- Attach a photo of the correct monitor operation.
- Re-submit the corrected files.