Tel Aviv University

The Iby and Aladar Fleischman Faculty of Engineering

Electronics - Laboratory (3)

Experiment FPGA 2 – I/O Interfaces

Written by: Konstantin Berestizshevsky

Lab Goals (This is also the material for the test)

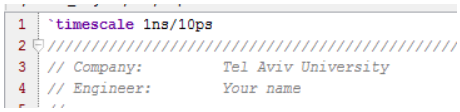
The purpose of this lab is to cover the following topics:

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 key-press
3. Video Graphics Array (VGA) interface – synchronization signals, color space encoding.

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. In the lab, you'll only need to implement the design on FPGA and deal with post-implementation debugging. Otherwise, you will not have enough time in the lab.

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:



5. At home you can work with the Vivado Web-Pack, which allows you to design and simulate RTL modules. Only at the lab you will be able to test you design on an actual FPGA chip.
6. It is your responsibility to either use a USB-flash drive or other cloud solutions to save your project before you leave the laboratory. The laboratory PCs are erased from time to time.

Task 1 – PS2 interface for keyboard input

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 (you can have 2 *.xdc files in the project, one for each task, with the unused file fully commented out). 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 2 outputs (scancode[7:0], keyPressed) The module provides an interface with the keyboard and outputting the most recently pressed 8-bit scan-code as well as outputting a pulse "keyPressed" to notify of a new key pressing at the moment of the first make-code packet reception.
2) **Ps2_Display** – with 4 inputs (clk, rstn, keyPressed, scancode[7:0]) and 4 outputs (seg[6:0], an[3:0], dp, led). The module operates 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 the 2 hex symbols of the scan-code received by the Ps2_Interface on the two right hand side 7-segment displays (recall that there are also extended keys which transmit 4-symbols of scan-code, for them you should indicate only the last 2 symbols and to ignore the 0xE0 prefix). The 7-segment display should show the scan-code of a key until a new key is pressed, at which time it starts to show the scan-code of the new key. You should adjust the code of Seg_7_display module from the previous lab to have the most of the functionality done. To distinguish between D and 0 as well as between B and 8, you should light up the dot (dp output) when showing D and B symbols or simply use lowercase "d" and "b".
   b. **User LED output (led)**. Your module must output a strobe signal called led to indicate the event of a key pressing on the keyboard. A strobe signal is a short pulse on one of the board LEDs, yet long enough to be noticed by the naked eye.
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.

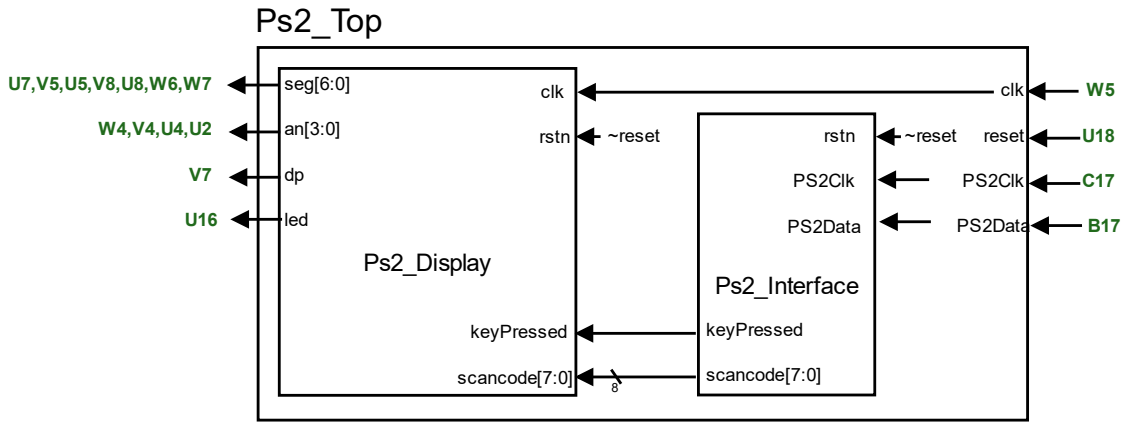


Figure 1 - High level diagram of the design required in task 1. The green labels are the FPGA pin name whereas the black labels are the IO ports of the Verilog sources.

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, 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 a rigorous simulation to ensure a correct timing. Make sure that the "keyPressed" pulse is generated correctly – at the moment of the first pressing on the button. Make sure that despite the numerous retransmissions of the make-code, you do not generate multiple pluses. Make sure to ignore the 0xe0 packets.
5. For this design, you should use the keyboard clock as an input to your module. Disregard what the Basys3 Board Manual says .
6. Logic Analyzer Tip: Logic analyzer is unable to debug nets from different clock domains. In your design there are indeed two clock domains (one works with the slow keyboard clock and another works with the fast 100MHz system clock). Therefore, if you need to debug, first perform a debugging for the keyboard clock related nets, and when you are done with them – remove the debug and add the fast-clock related debug nets. You can also try creating two separate debug cores simultaneously on the same design.

In the Lab:

The keyboard that we'll use is a small numeric keyboard.
Connect the keyboard to the USB port of the BASYS board,
Connect the PC to the BASYS board Jtag-port.
Program the FPGA with your implemented design and call an instructor to show him a correct functioning of your PS2 interface.

Submit:

In preliminary report – submit the files:
Ps2_Interface.v, Ps2_Interface_tb.v, Ps2_Display.v, Ps2_Top.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 [see demo]

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 colors set from the keyboard. Figure 2 depicts a high-level diagram of a design required in this task.

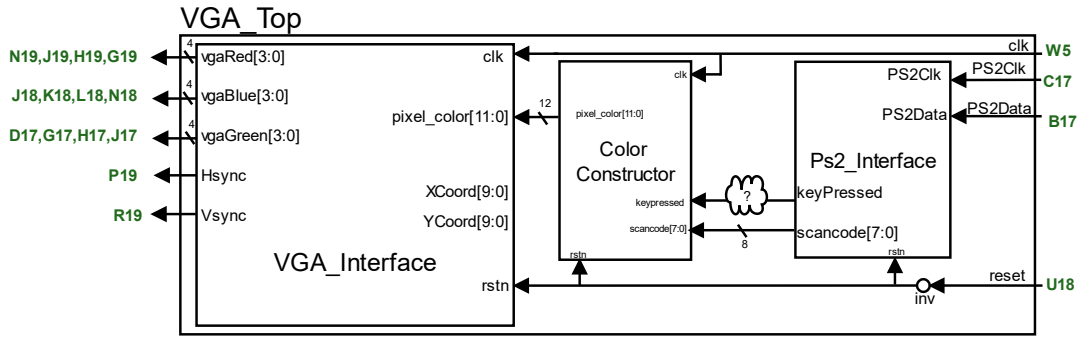


Figure 2 - High level diagram of the design required in task 2. The green labels are the FPGA pin name whereas the black labels are the IO ports of the Verilog sources.

The new 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 25MHz pixel-clock internally. The outputs vga* and *sync of VGA interface must respect the VGA standard for 640x480 resolution (see theoretical background). It will also output the current horizontal coordinate (0-799) and vertical coordinate (0-524).
2) **Color_Constructor** – 4 inputs: clk, rstn, keypressed, scancode (8 bits). The output is the 12-bit encoding of the current color to be shown on the monitor. This is a synchronous module which decides which color is to be sent to the monitor (VGA interface) according to a color code fed by keyboard (incoming from ps2 interface). The initial pixel color should be white (0xfff) and from this moment the user can type 3-tuples of digits terminated by an “Enter” key stroke in order to set the R,G,B channel intensities of the presented color. 0 value is the lowest intensity and 8 is the highest. The color is updated on the screen only after the Enter is pressed. For example pressing 8, then 8, then 0 and then “enter” will result in registering the yellow color.
3) **VGA_Top** – a top level module comprised of the VGA_Interface, PS2_Interface and the Color_Constructor modules. It receives a 100MHz clock and reset signals, as well as PS2’s clock&data from the board. It outputs the VGA control signals using VGA_Interface. The goal is to receive color codes from the keyboard (in a form of R-intensity,G-intensity,B-intensity,”Enter”) and to send the resulting RGB-coded color to the VGA port.

Guidance:

The **VGA interface** can be implemented as follows:

1. Use 2 counters to store the values of *hcount* and *vcount*
2. On the rising edge of the pixel clock, increment the *hcount*. Increment *vcount* when *hcount* has reached the end of the row.
3. 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.
4. Generate a signal to determine whether the pixel is in the visible region as illustrated in the regions diagram in theoretical background
5. When in the visible region, output the pixel color value {R, G, B}, otherwise when in the blanking region, output {0, 0, 0}.
6. 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.
7. The 25Mhz pixel clock can be generated from the 100Mhz system clock.
8. Perform a rigorous simulation to ensure precise timing, which must perfectly correspond to the timing diagrams in the theoretical background.

The **Color_Constructor**:

1. Design can be based on a 3-word shift register, as appeared in the briefing.
2. An additional register should be used to hold the most recently accepted color code, and meanwhile a new color can be fed by the user. The additional register receives the new color upon an "Enter" keystroke.
3. Pay attention to the pulse width of the *keypressed* signal as it is provided by the interface. You should consider ways to shorten this pulse (to a single 10ns pulse) to avoid the effect of "multiple key pressings" on the Color_Constructor's side.
4. When the user presses on a non-numeric key, refer to its numeric value as 0.
5. If the user didn't press all three numeric keys prior to pressing "Enter" – you may implement any behavior you choose. Just explain it in the report.
6. Intensity scaling – the valid intensities of each color channel, fed from the keyboard, are 0…8. However you should scale them to the full 4-bit range {0,2,4,6,8,10,12,14,15}.

In the Lab:

Connect the Jtag cable from PC to FPGA board

Connect the VGA cable from the monitor to the FPGA board.

Program the FPGA with your implemented design and call an instructor to show him a correct functioning of your VGA interface.

Take a photo of a correct functioning of the monitor.

Submit:

In preliminary report – submit the files:
VGA_Interface.v, VGA_Interface_tb.v, Color_Constructor.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.
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 showing a special color of your choice. Re-submit the corrected files.