|  |  |
| --- | --- |
| *Title:* | ***Lab #9: FPGA Pong – Top.vhd*** |
| *Name:* |  |

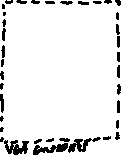
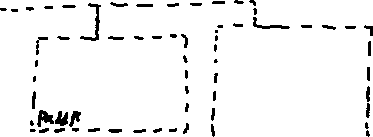
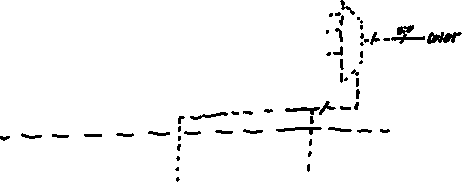
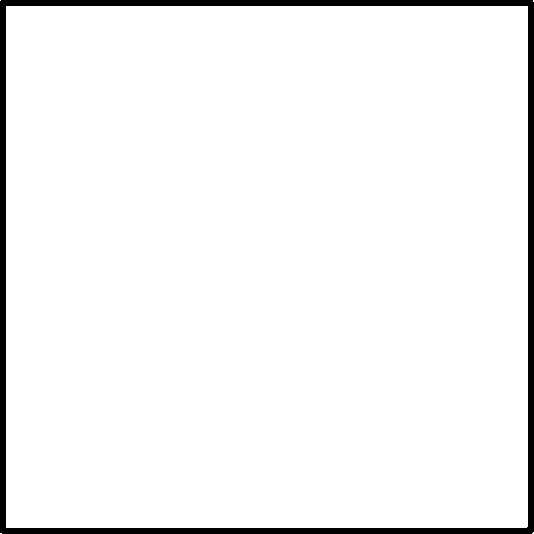
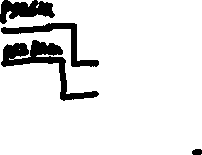
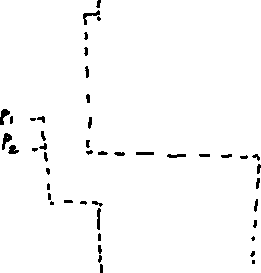
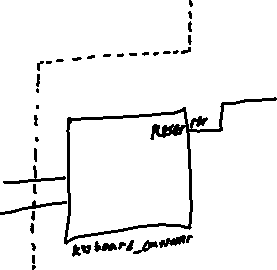
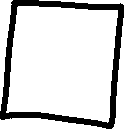
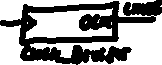
# INTRODUCTION

You need to use Xilinx Vivado.

Very few real-world applications of VHDL and Verilog use just one file. Even in the last lab, we had two separate files for the clock divider and the score keeper. With such modularity in the code, it is common for there to be one component referred to as the top or head component, which is used to instantiate all other components of your system. We will be creating this file today and instantiating our score keeper component along with a keyboard controller.

Read the previous lecture notes for an overview of how to instantiate components in VHDL, Or alternatively look at last week’s test bench.

Here is a quick overview of what we are trying to accomplish:



# Start Xilinx Vivado

1. In the pong project, create a new design source called Top.vhd.
2. Use the following input and output structure for the component:

clk : in STD\_LOGIC;

PS2Clk : in STD\_LOGIC;

PS2Data: in STD\_LOGIC;

Hsync : out STD\_LOGIC;

Vsync : out STD\_LOGIC;

color : out STD\_LOGIC\_VECTOR(11 downto 0);

an : out STD\_LOGIC\_VECTOR(3 downto 0);

seg : out STD\_LOGIC\_VECTOR(6 downto 0)

1. These inputs and outputs control the following:

|  |  |
| --- | --- |
| **Ports** | **Links to** |
| clk | Clock divider |
| PS2Clk, PS2Data | keyboard controller |
| Hsync, Vsync, color | VGA controller |
| an, seg | 7 segment display |

1. Import the keyboard\_controller component into your design sources.
2. Import the “top\_TB.vhd” file into your design sources
3. Right click the top\_TB.vhd component and select “Set as Top”. This means that in simulations and in bitstream generations, it will be the first component read. Otherwise, the simulations might simulate only parts of a project, rather than top\_tb.vhd (which will contain all other components)

**Component In**stant**iation**

To declare a component, we use the following syntax:

component (component\_file\_name)

Port (

X: in STD\_LOGIC;

Y: out STD\_LOGIC;

…;

Z: in STD\_LOGIC\_VECTOR (1 downto 0)

);

end component;

Where component\_file\_name is the name of the component in your files, exactly as written (without the .vhd extension). Component declaration happens within the architecture block of the component, before the ‘begin’ statement. This essentially tells the component “There is another component with this name somewhere in the project, here are its inputs and outputs.” Once the component is declared, inside the begin statement, you can instantiate it. To do so, the following syntax is used:

(alias) : (component\_file\_name)

port map (

(lower\_component\_port) => (upper\_component\_signal),

(l\_port\_2) => (u\_port\_2),

…,

(l\_port\_3) => (u\_port\_3)

);

Where “alias” is whatever name you wish to give the component. Note that there are no begin or end statements in this instantiation. This is because despite the whitespace there is only one command actually being issued, and so we are allowed to omit the begin and end statements. In this scenario, the left side of the port assignments are the names used in the declaration (X, Y, Z in the above example), while the right side is whatever you wish to assign the port to in the top component (ports or internal signals).

# Procedure

1. Declare the Clock Divider, Score Keeper, and Keyboard Controller components in top.vhd (not top\_TB!). You will need to look in the keyboard controller component to see what the names of the ports are. Spend some time reading the comments to get a basic understanding of how the PS/2 Protocol works for peripheral communications between keyboards/mice and the controller.
2. You will need to create signals for all ports in the components listed that do not connect directly to the ports of top.vhd. look back to page 2 to see the table of all of the top.vhd ports and what they link to. For the ports on subcomponents that do not have a matching name in top.vhd, create a signal.
3. Instantiate all of the components in top.vhd.

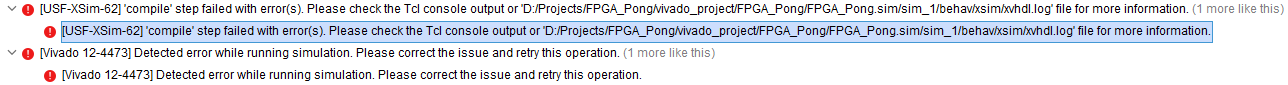
# Compile and test it!

Generate your bitstream and upload it onto the board. If your code works, by pressing 1, 2, 3, 4 you should be able to increment the seven segment display’s 1st, 2nd, 3rd, or 4th digit respectively.

No papers need to be turned in for this lab; have the TA sign off on their sheet that you have finished the work.

# Troubleshooting

It is understandable for the component not to work first try, but if last week’s lab did work, then you have already ruled out all of the problems within the clock\_divider and score\_keep components themselves. If these two components do not work on their own (see lab 8), then troubleshoot them first.

Read the messages log. Usually if something failed, you can get a good hint as to what failed there. Look for any error messages. Usually they provide a line telling you exactly where an error might be, but one of the most common errors is something along the lines of…

In this scenario, it is very likely that a component was instantiated incorrectly. A computer code with text

AI-generated content may be incorrect.

In this case, there was a simple typo in the score\_keep instantiation. Vivado will not pick up on this error by itself, and it will still show the green box. It will, however, find the error if the instantiation and declaration names are different. If the instantiation said “scor\_p1” and the declaration was “score\_p1”, you would get an error along the lines of “no default value”.