Tutorial: Debugging with Vivado ILA Cores

Created by Yu-Wei Lee, Mar. 2015

In this tutorial, we want to add debugging probes in Vivado's Integrated Logic Analyzer (ILA) cores to a BCD counter. *(Note: The ILA debugging core feature is only available on the full Vivado installed on lab computer and not available on the free WEBpack version.)*

Debugging probes allow you to observe values for various signals on your Basys3 board in real-time. The flow is similar to typical Vivado synthesis/implementation flow with a few additional steps. *For typical synthesis/implementation flow, refer to the Vivado tutorial on Canvas.*

Step 1. Creating a Project in Vivado

To create a project, use the New Project wizard to name the project, to add RTL source files and constraints, and to specify the target device, as normal.

> *Note: In the Project Manager, click Project Settings, go to the Synthesis tab and change the -flatten_hierarchy option to none, then click OK. The reason for changing this setting to none is to prevent the synthesis tool from performing any boundary optimizations for this tutorial.*

Step 2. Adding Debug Probes

There are multiple ways to add debug probes in our design. In this tutorial, we will be using the Netlist Insertion method. To add a Vivado ILA 2.0 core to the design, we will take advantage of the integrated flows between the Vivado IDE and Vivado logic analyzer. First synthesize your design. Then, from the Synthesis pull-down, click Open Synthesized Design.

> *Note: Before proceeding, make sure that the Flow Navigator on the left panel is enabled. Use Ctrl-Q to toggle off and on. Secondly, the window layout must be set to Debug. At this point you may get a Synthesis is Out-of-Date warning. Click Open Design.*

1. Click the Debug tab on the bottom if it is not already selected. Select the Netlist tab to expand Nets. *If you don't see the Debug or Netlist tabs, click on Windows->Debug or Windows->Netlist.*

2. Right-click the selected nets and select Mark Debug. (Don't mark all the nets, or else implementation may take a very long time.)

   Note: As shown in Figure 1, in Vivado IDE, you can also see the green bug icon next to each scalar or bus, which indicates that a net has the attribute mark_debug = true. If you see a white bug, you will need to assign a clock domain to that signal in step 3.

For I/O related nets, mark "name_IBUF" or "name_OBUF" for debug.

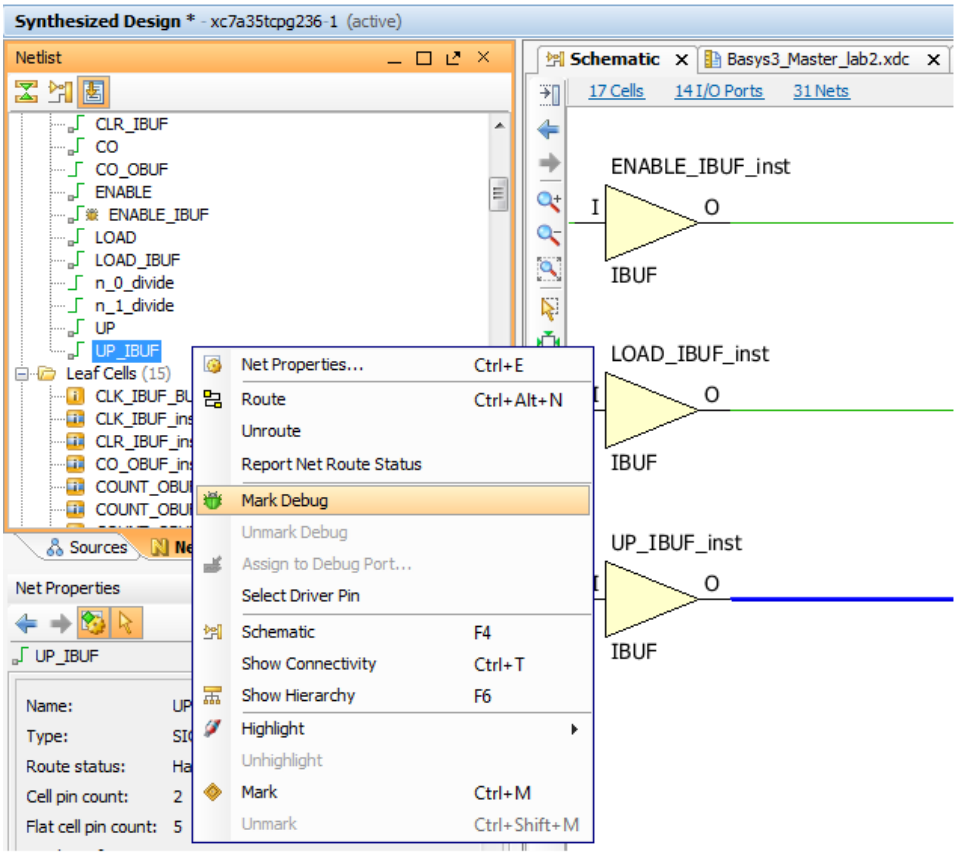

Figure 1. Marking Debug Nets from the Netlist Tab

The following are some other ways to add debug nets using Vivado:

• Add mark_debug attribute to the target XDC file *(Do not use this with pre-synthesis or elaborated design netlists):*

set_property mark_debug true [get_nets sine*]

• Add mark_debug attribute to HDL files

(* mark_debug = "true" *) wire sine;
(* mark_debug = "true" *) wire sineSel;

• Use a Tcl prompt to set the mark_debug attribute.

For example, set mark_debug true [get_nets sine]. This applies the mark_debug on the current, open netlist.*

Step 3: Running the Set Up Debug Wizard

1. From the Debug tab (by clicking on the tab) or Tools menu, select Set Up Debug. The Set Up Debug wizard opens as shown in Fig.2. If the wizard asks to update your .xdc file, click ok. You should see your .xdc file being updated.

Figure 2. Set Up Debug Wizard

2. Click through the wizard to create Vivado logic analyzer debug cores, keeping the default settings. Make sure you assign a clock domain to each signals as shown in Fig 3.

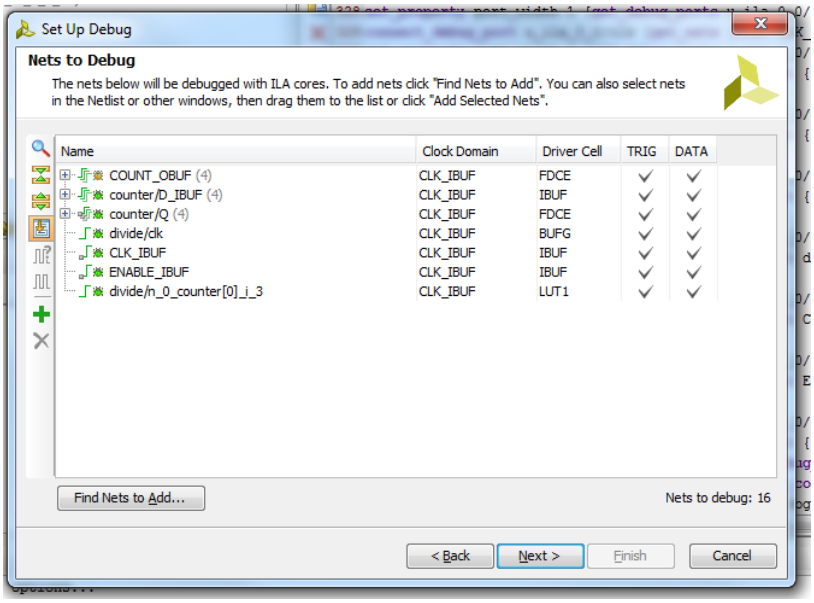

Figure 3. Assigning Clock Domain to Nets

3. Implementing the design and generate the bitstream file.

    - *In the upper right-hand corner, you can click on more info and then Force up-to-date. This step will prevent the tools from re-synthesizing and re-implementing the design.*

Step 4: Loading Bitstream with Vivado Logic Analyzer (ILA)

1. Ensure that the board is plugged in and powered on. Open your target device by open target, program the device using the **.BIT** bitstream file that was created previously. Note that the window layout has changed to look like figure 3, with a new tab named *ILA probes*:

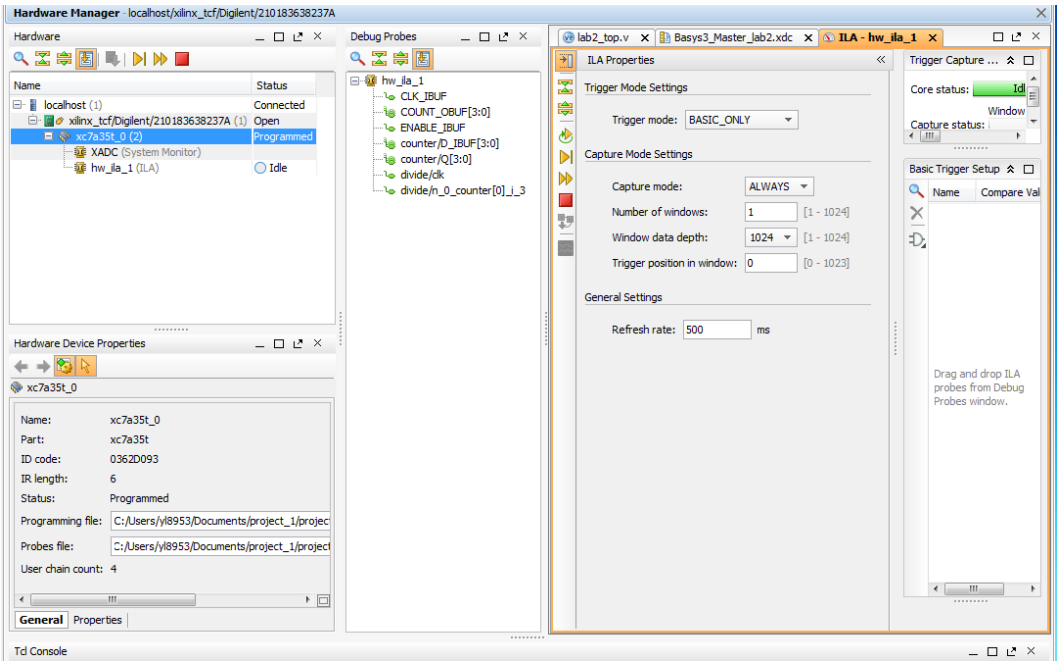

Figure 3. Window Layout with ILA Probes

2. Ensure that an ILA core was detected in the Hardware panel of the Debug view, as shown in Figure 4.

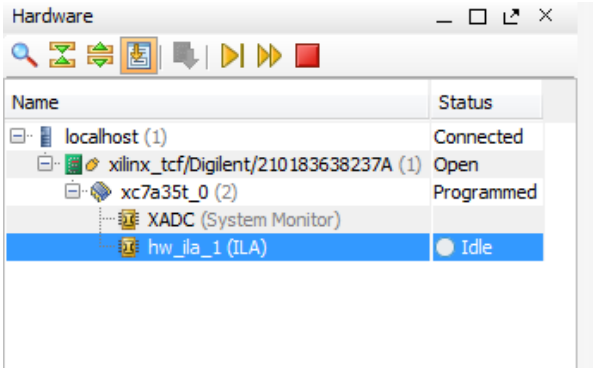

Figure 4. ILA Core Detection

3. Next, select the ILA Probes tab as shown in figure 5 and ensure that all of the debug nets added from previous steps are accounted for.

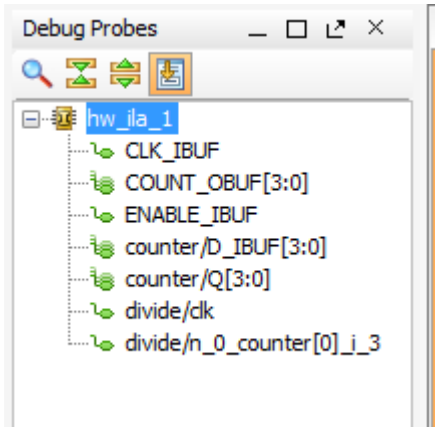

Figure 5. List of Debug Nets

Step 5. Verifying Waveform Activity

1. Click the Run Trigger Immediate button to trigger and capture data immediately as shown in figure 6.

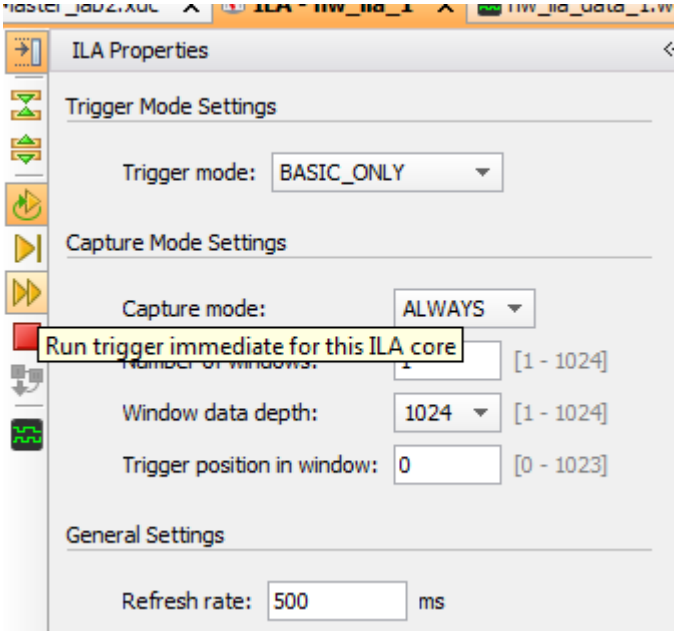

Figure 6. Run Trigger Immediate Button

2. In the waveform window, observe waveform activities. Click on Run Trigger again to update the waveform. Figure 7 shows an example waveform. *You can also toggle auto-retrigger button in figure 6 so it updates automatically.*

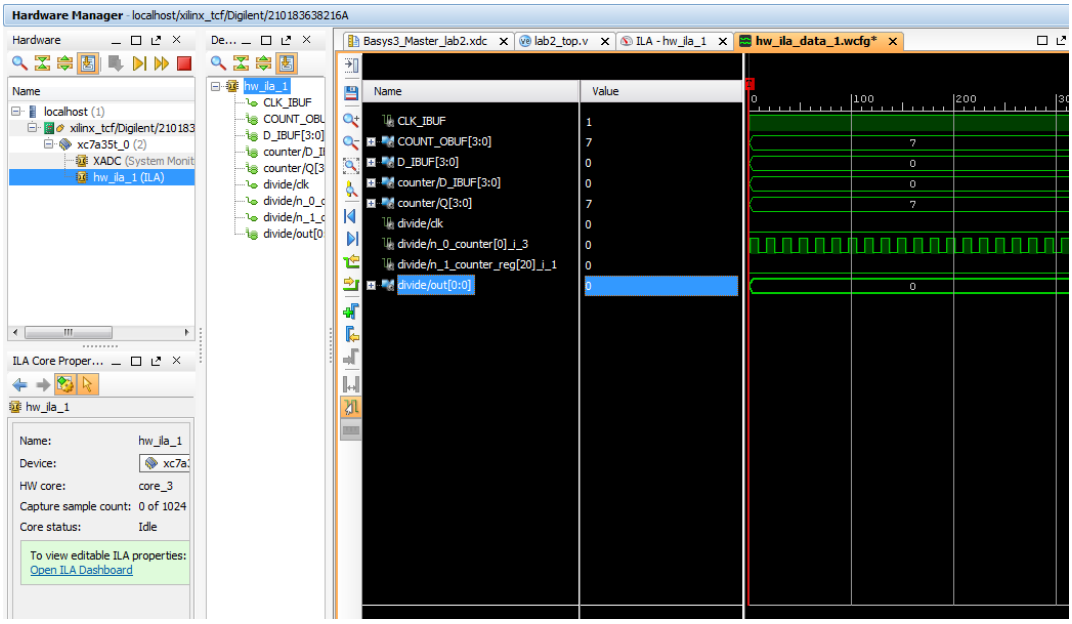

Figure 7. Waveform Window

3. You can also define triggers to wait for a specific event. Before clicking on "Run triggers", in debug probes, right click on the signal you want to monitor, select "Add probes to basic trigger setup." Signal will show up in "Basic Trigger setup."

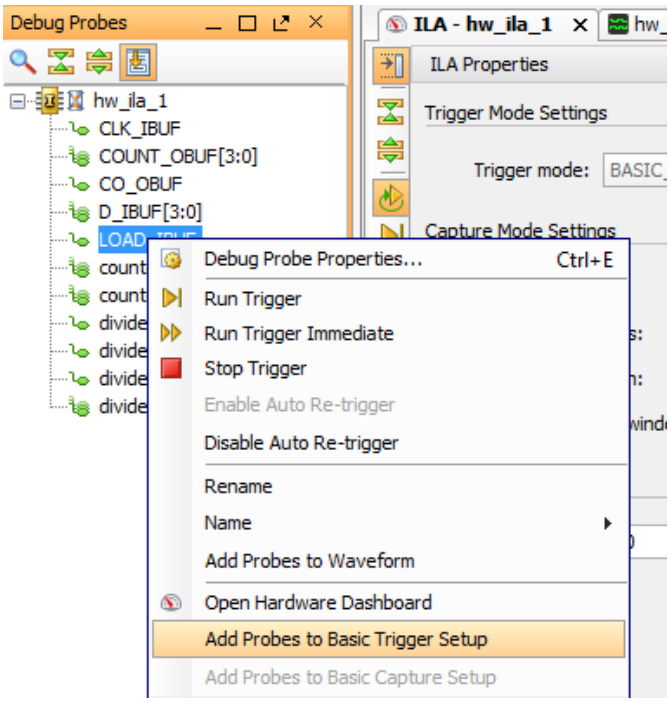

Figure 8. Add Probes to Basic Trigger

4. In Basic Trigger Setup, click on the signal and set compare value to 1/0/X or a rise/fall transition.

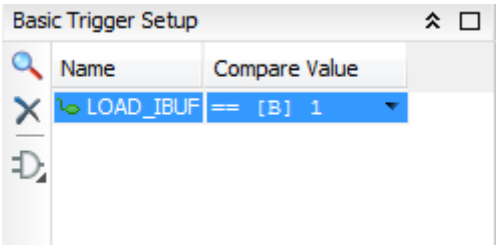

Figure 9. Set Compare Value

5. Click on “Run Trigger”. Waveform will show up when the specified trigger occur. *You can also toggle auto-retrigger so your waveform gets updated whenever the specified event happen.*