**Hardware Debug Instructions**

This document describes how to perform hardware debug using Vivado Integrated Logic Analyzer (ILA) cores. The comprehensive guide to this topic can be found in Xilinx User Guide (UG936):

<https://www.xilinx.com/support/documentation-navigation/see-all-versions.html?xlnxproducttypes=Design%20Tools&xlnxdocumentid=UG936>

This guide uses marking signals for debug in HDL source files.

A second approach is Marking signals and busses for debugging in a Vivado Block Design.

A third approach is also possible where signals for debug can be marked and configured using the post-synthesis netlist, by writing the debug configuration into a XDC constraint file.

**Marking Signals for Debug in HDL Source Files**

1. Before the begin statement in the architecture under signal declarations, add these statements. See avr\_fpga for example.

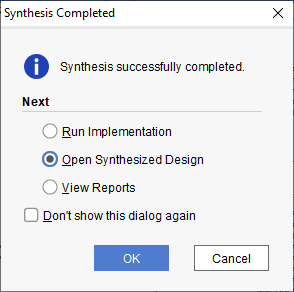
attribute mark\_debug : string;

attribute mark\_debug of signal\_name1 : signal is “true”;

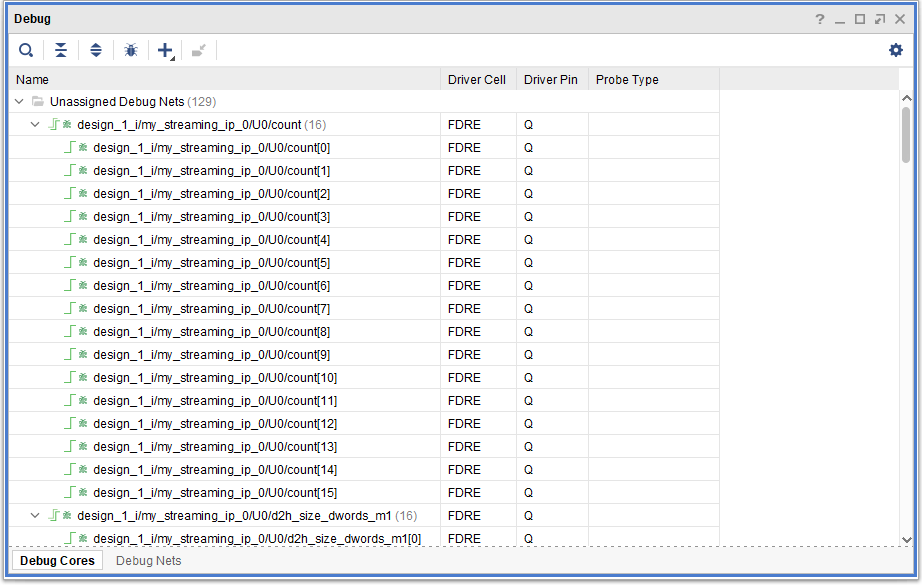
attribute mark\_debug of signal\_name2 : signal is “true”;

Save the file and re-run the synthesis step.

1. After synthesis has completed, select Open Synthesized Design…



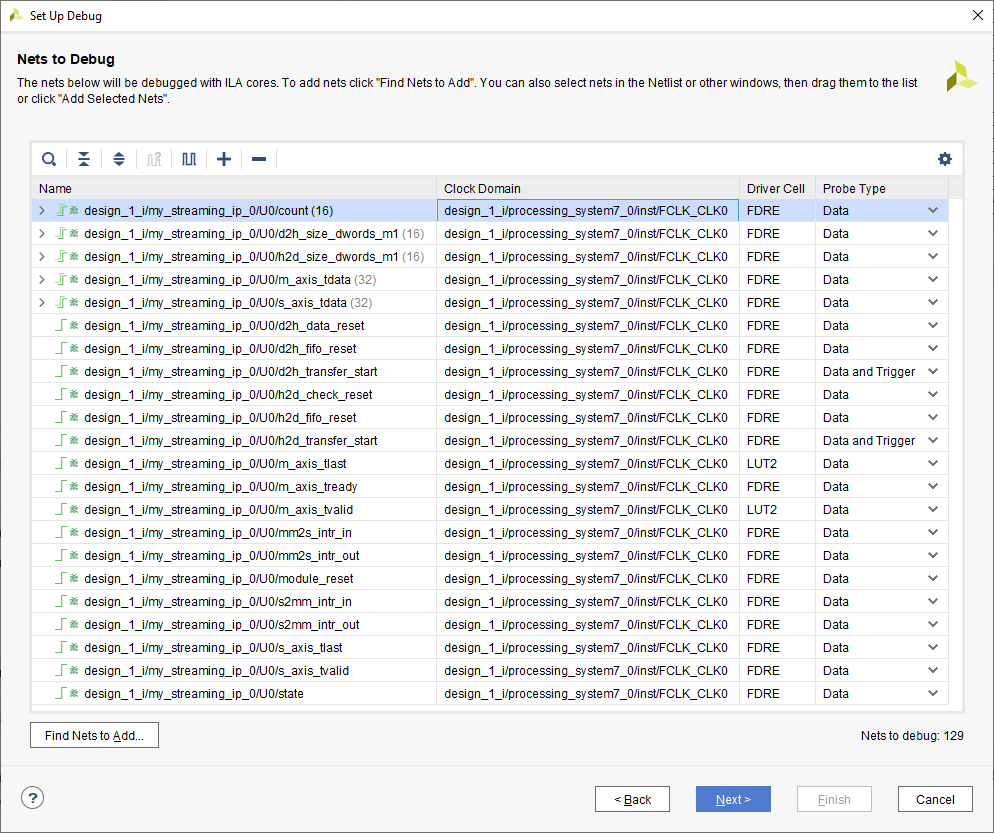
1. In the Debug window, the signals that we marked using VHDL attributes will be visible:



1. Click the ‘bug’ icon to Setup Debug:

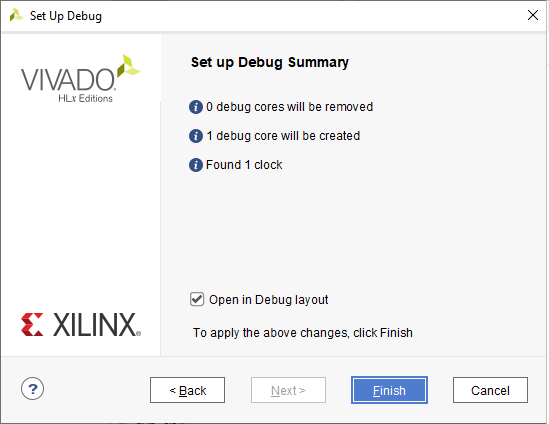
This will bring up the ‘Nets to Debug’ dialog. Highlight the nets that have an undefined clock and select local. Choose the main clock coming out of I\_Buffer.

Since ILA cores use the PL’s FPGA fabric including block RAM for storing the sampled data, only necessary features should be enabled. If not, an overly complex ILA core configuration will interfere with the implementation process leading to long implementation runtimes and/or a design that may fail timing. This can happen due to insufficient routing resources or inefficient routing. Hence, the probe type for other signals should be changed to ‘Data’. This will minimize logic resources used for trigger events. Modify the debug configuration as shown, but with the desired signals.

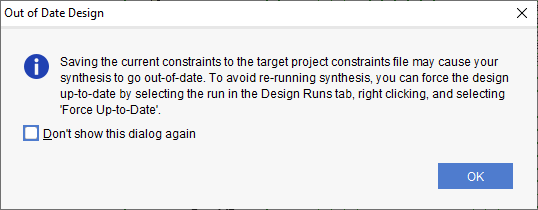


1. The *Sample of data depth* setting determines how much data is captured. Expand this to whatever size needed. 1024 is probably ok. Check advanced trigger.

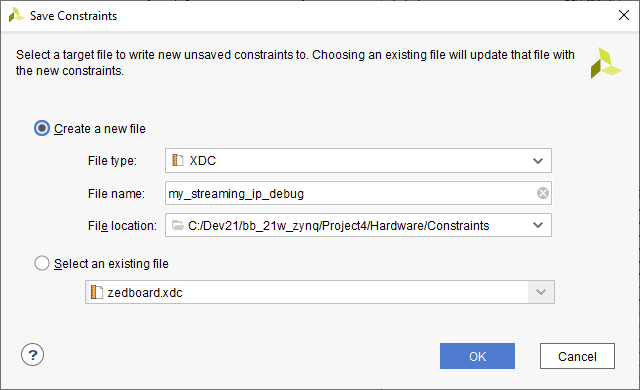
Click through to complete the debug configuration:



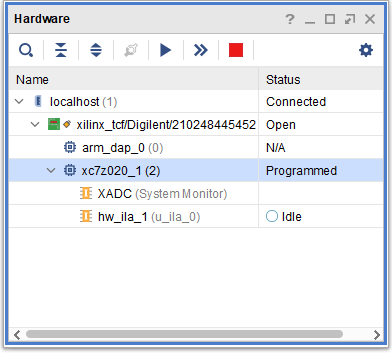
1. Save the configuration.



1. Next, the option is presented as to where the debug configuration should be saved. In general, we want to keep the hardware debug configurations in a separate XDC file than the one used for I/O location constraints. Select the Create New file option and set the path to where we store the project’s constraints files:

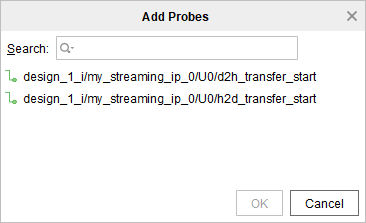


1. Proceed to Generate BitStream… This will re-synthesize the design. Always do this step since it does not take that much time and is the safer option compared to force-updating synthesis.
2. Open the hardware manager and connect. Select Refresh device. Now the hw\_ila\_1 core should be visible:



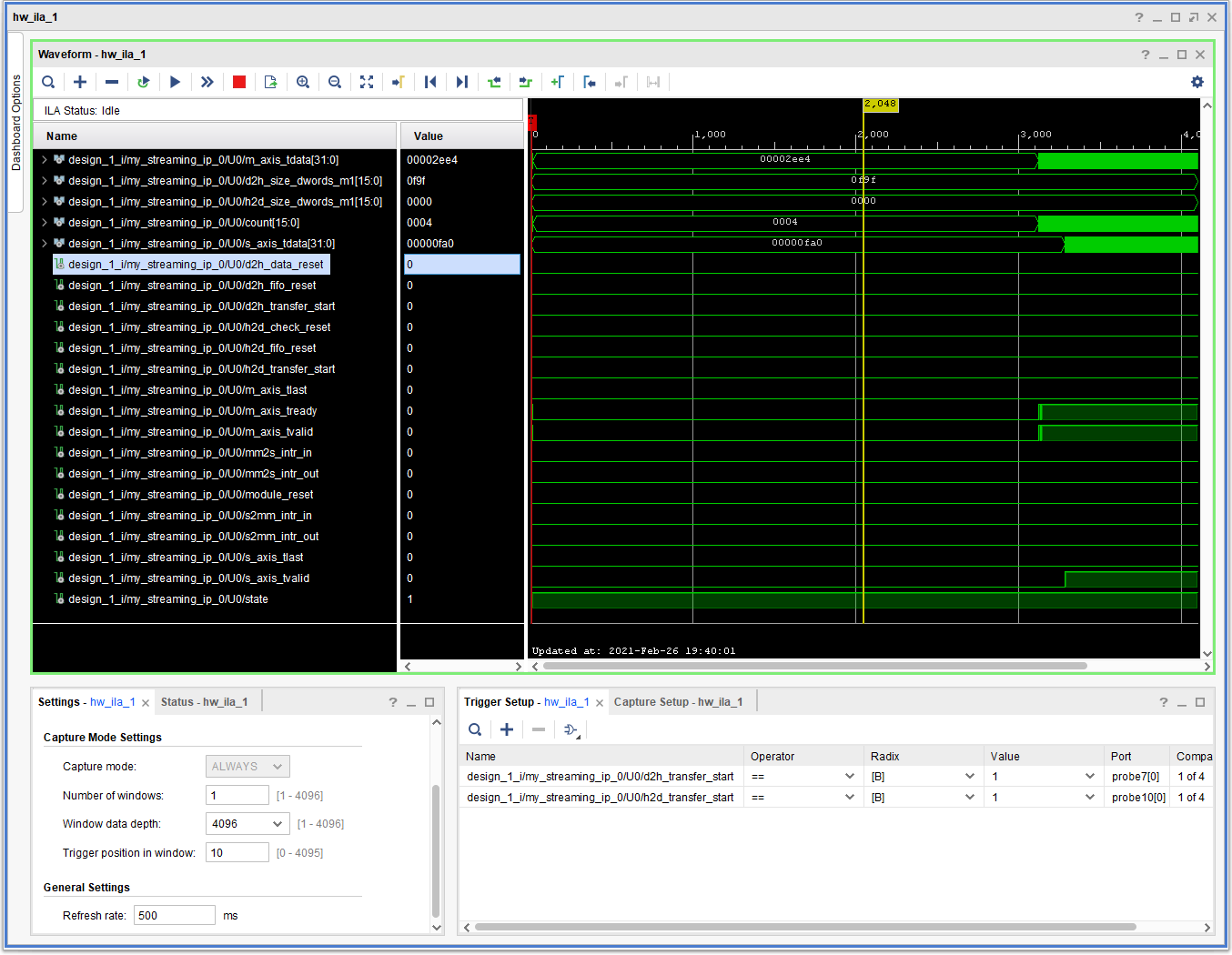
The waveform window should show the signals that we marked for debug.

1. In the Trigger Setup window, click the + button to add trigger signals:



Add both *d2h\_transfer\_start* and *h2d\_transfer\_start* signals into the Trigger Setup

1. Set the Value field 1 (logical 1) for both signals. Then select the gate icon and select Set Trigger Condition to ‘Global OR’. This will cause the core to trigger when either of these signals asserts to logic ‘1’. Set conditions as desired.
2. In the Waveform window, press the ‘>>’ button.
3. In the Waveform window, press the blue ‘Play’ button. This will cause the ILA core to wait for the configured trigger condition. In the Hardware tab, you should see an hourglass icon with a Status saying ‘waiting for trigger’ When the trigger event occurs, the waveform should populate.



1. Configure the switches to perform DMA transfers every second and apply this setting. You should now be able to trigger on these events and observe the data changing. Note that data is being transferred in both directions simultaneously, confirming full-duplex operation:

