**Tutorial 6**

*April 2022*

**ChipScope on SR1**

In this tutorial we see how to view signals on the hardware running at SR1 using an ILA on ChipScope. This can be done remotely as long as one has access to the usual SR1 setup and also pix-sr1-01 (or another dev machine). We will go over the basics of setting up an ILA from the existing project. There is a lot more to ILAs, but the main objective of this tutorial is to show how to use an ILA to spy on signals while doing runs on SR1.

**Modifying the CDC file**

A .cdc file (ChipScope Definition and Connection file) is used to define an ILA and signal connections in the firmware to insert using Core Inserter. It is not the only way to insert an ILA but is recommended (and widely used by Pixel DAQ) over writing an ILA directly into the HDL, as it can avoid errors, and having a cdc file is also useful when viewing signals which will be described later.

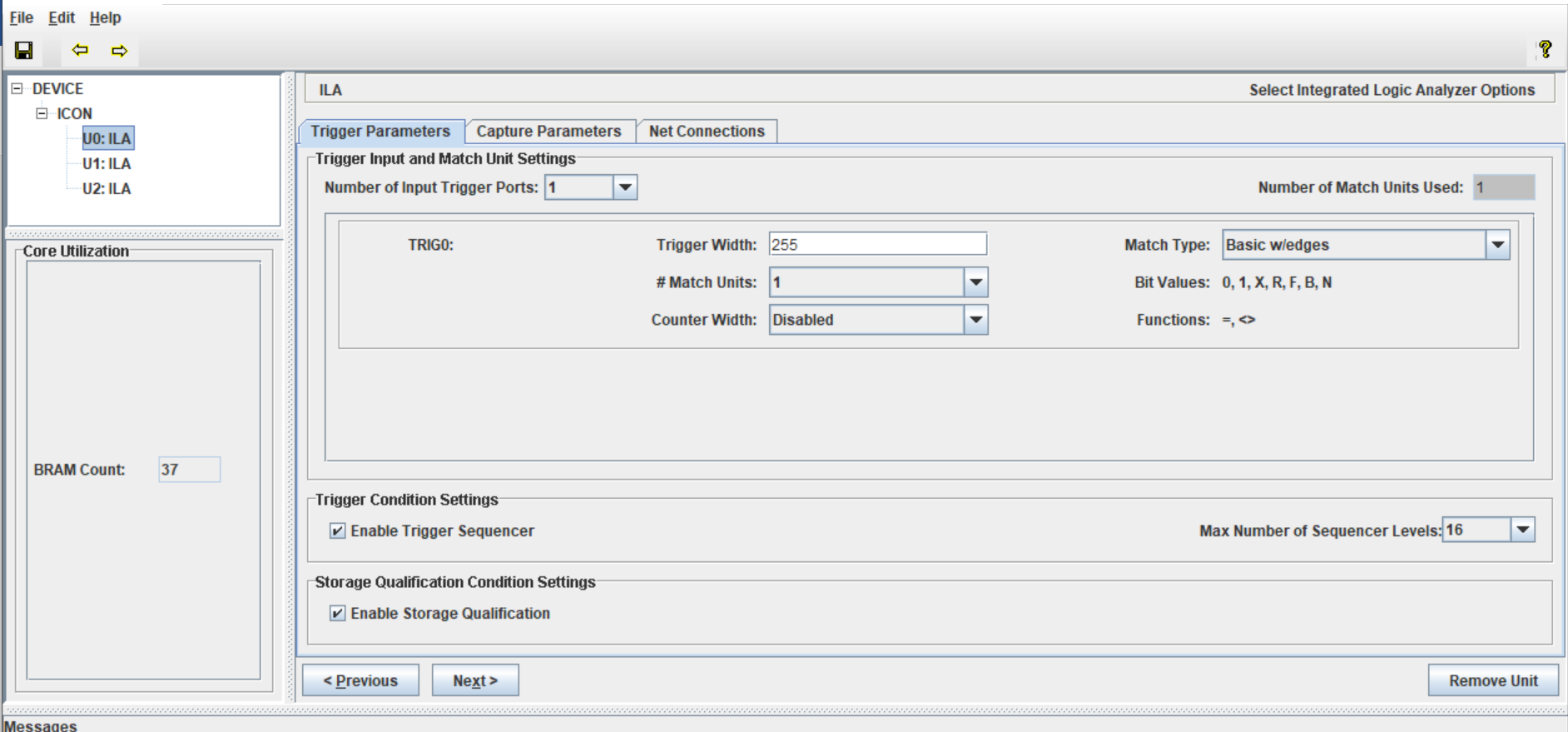
1. Open the ISE project

There should be a .cdc file by default in the design hierarchy. Click on this to invoke Core Inserter. It may go through synthesis if it hasn’t been done already.

If there is no file, create one by adding a new source to the project and selecting cdc.

1. Core Inserter

Once core inserter window pops up, the ILAs present in the file can be seen on the left. In this example we look at ILA0 but the procedure for any ILA is the same.

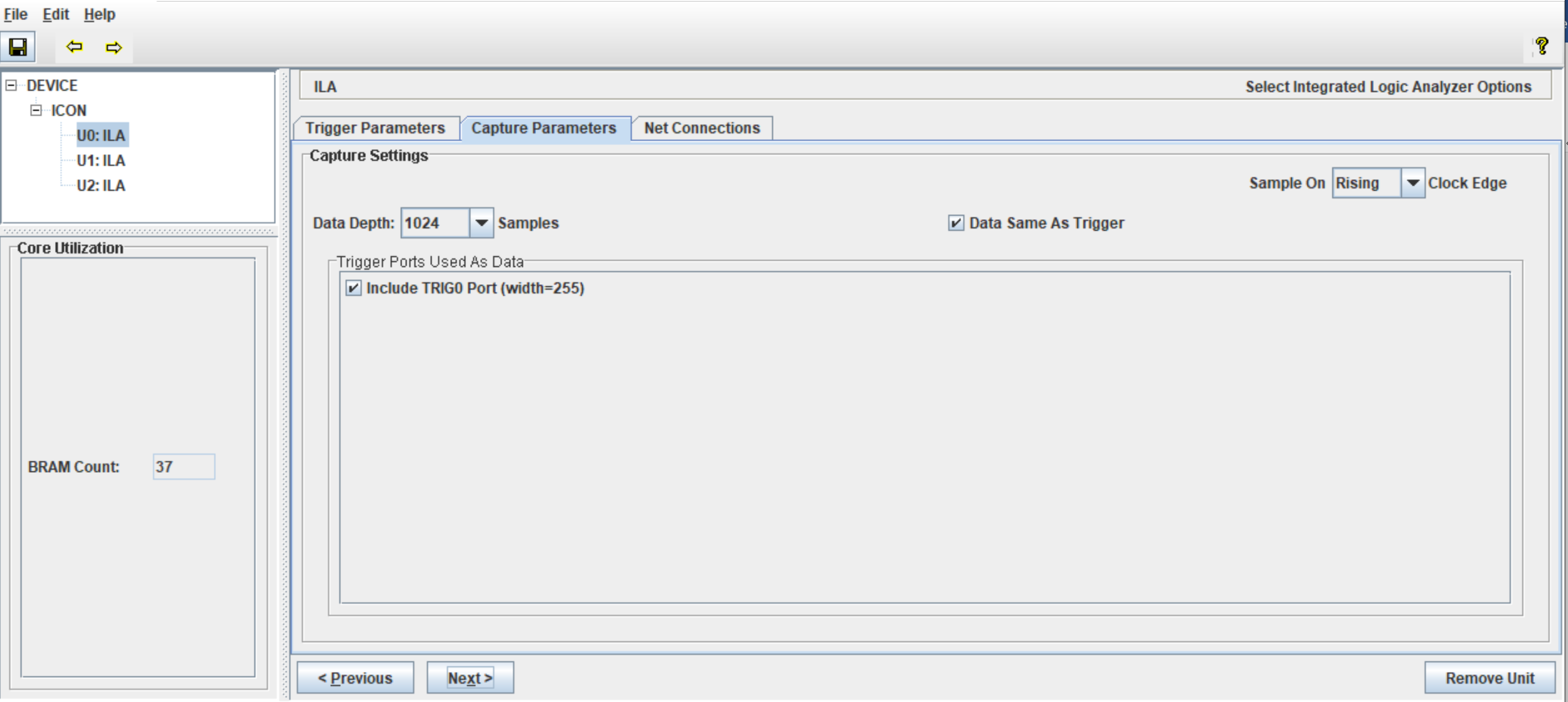


Go to the Trigger parameters tab

Trigger width – number of signals to view, the maximum possible number is 256, so if there are more signals you may want to have more ILAs (if resources allow it)

Capture Parameters

Data Depth – the number of samples that can be collected. One sample is collected in every clock cycle. So, in this ILA, 255 signals can be captured with a width of 1024 samples each.

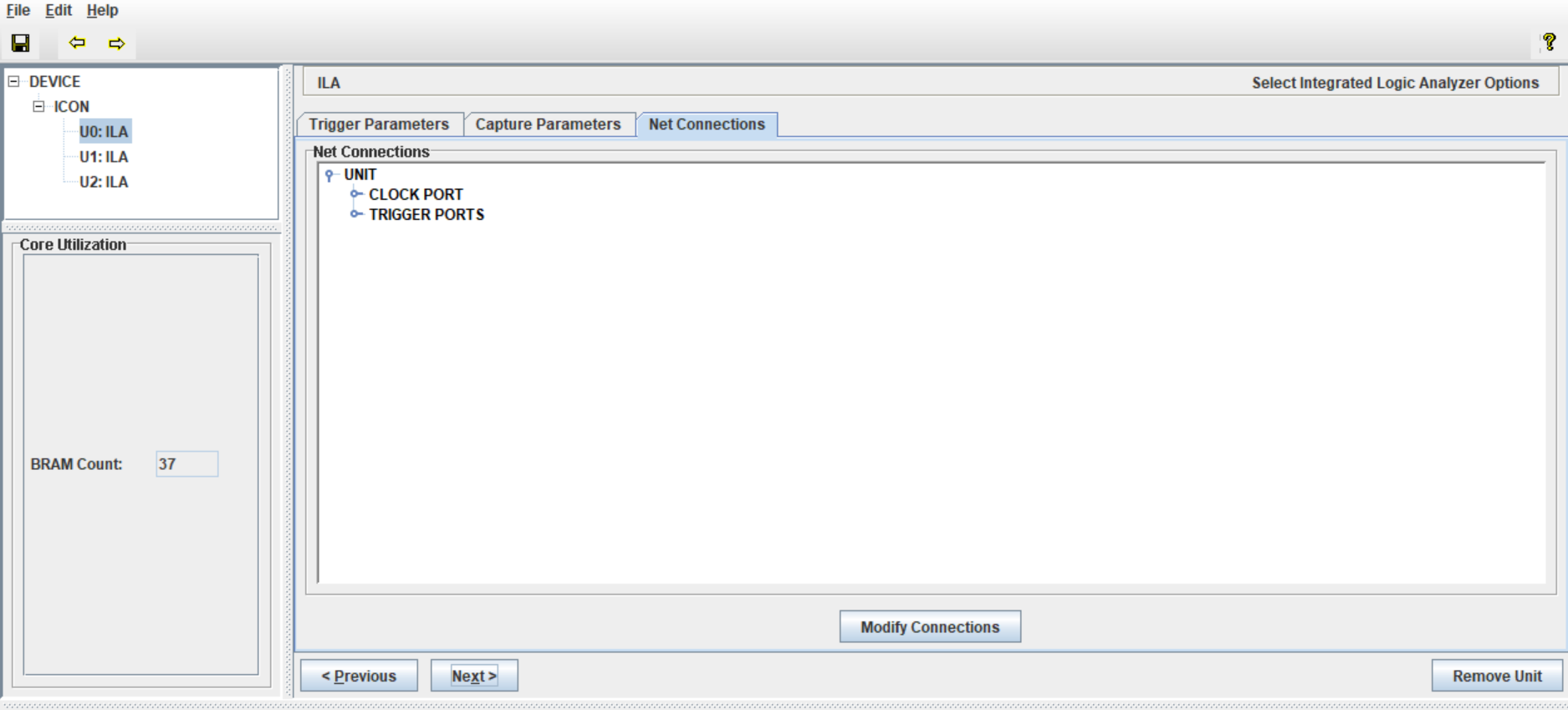


Keep an eye on the BRAM count on the left while deciding trigger width and depth. Since an FPGA has limited resources and most of it likely will be used by the actual design, it is important that the ILA doesn’t use up too many resources and actually fits in the FPGA (It will fail Implementation if it doesn’t). In this design I found that 37 was the limit and going over that fails implementation.

1. Net Connections

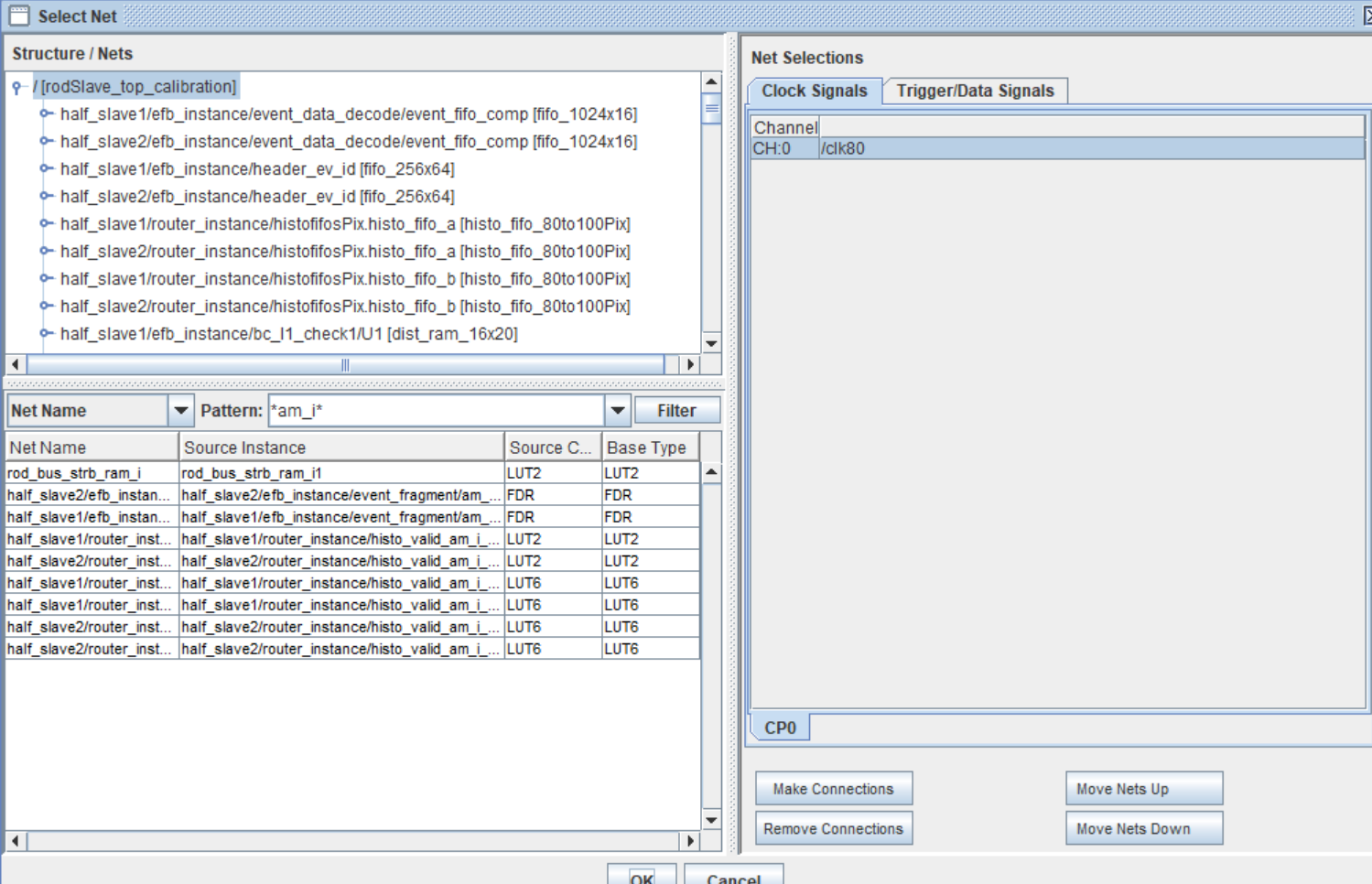
Here we get to define how the ILA will be clocked (select a clock signal) and which signals we want to look at.

Click on Modify Connections

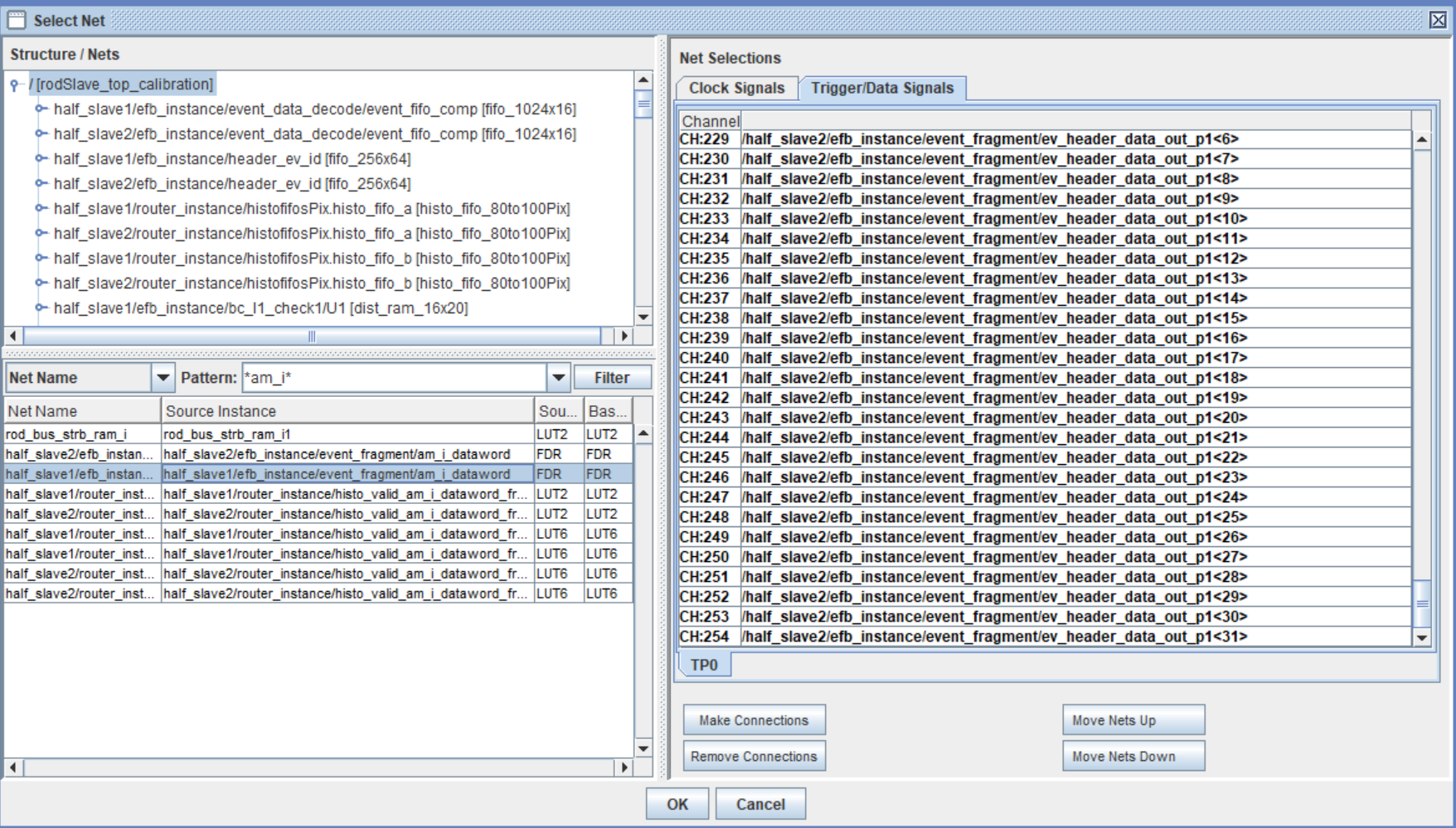


Select clock signals – search for the clock signal relevant to the signals to be probed. If there are signals across clock domains that need to be probed it is recommend to use separate ILA for them.

Here clk80 is the clock signal



Go to trigger signals – these are the signals you want to view. Search for the signals on the left, select and click make connections. The other buttons can be used to remove and move around signals. The number of signals added has to match the given trigger width else the file will not be complete. If all of them aren’t filled up, go back and edit the trigger width to match.



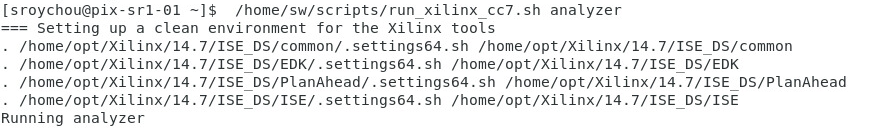
1. When satisfied with the signal selections, close the window and save changes. Generate the bitstream, and load it into the FPGA at SR1(covered in previous tutorials). Also remember to copy the cdc file to you home directory.

**ChipScope on SR1**

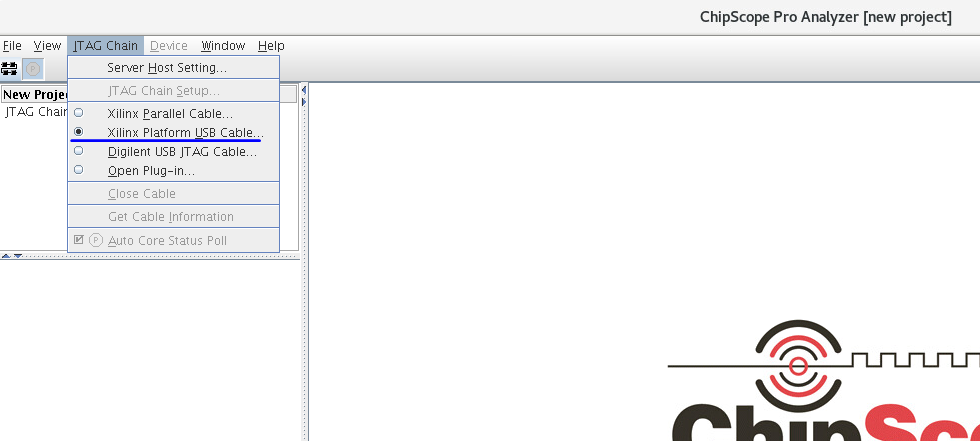
1. ssh into pix-sr1-01 (ssh -XY pix-sr1-01). This machine has the license for Xilinx tools.

Run this command to invoke analyzer

/home/sw/scripts/run\_xilinx\_cc7.sh analyzer

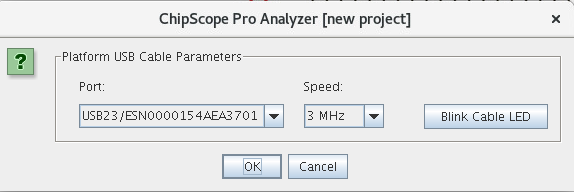


1. A ChipScope analyzer window should pop up click on JTAG chain -> Xilinx Platform USB Cable

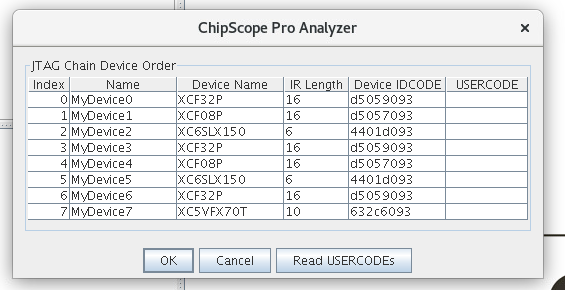


Select the correct cable corresponding to the board in which fw was loaded from list or the module that SR1 runs will be done on (Selected using TagManager).

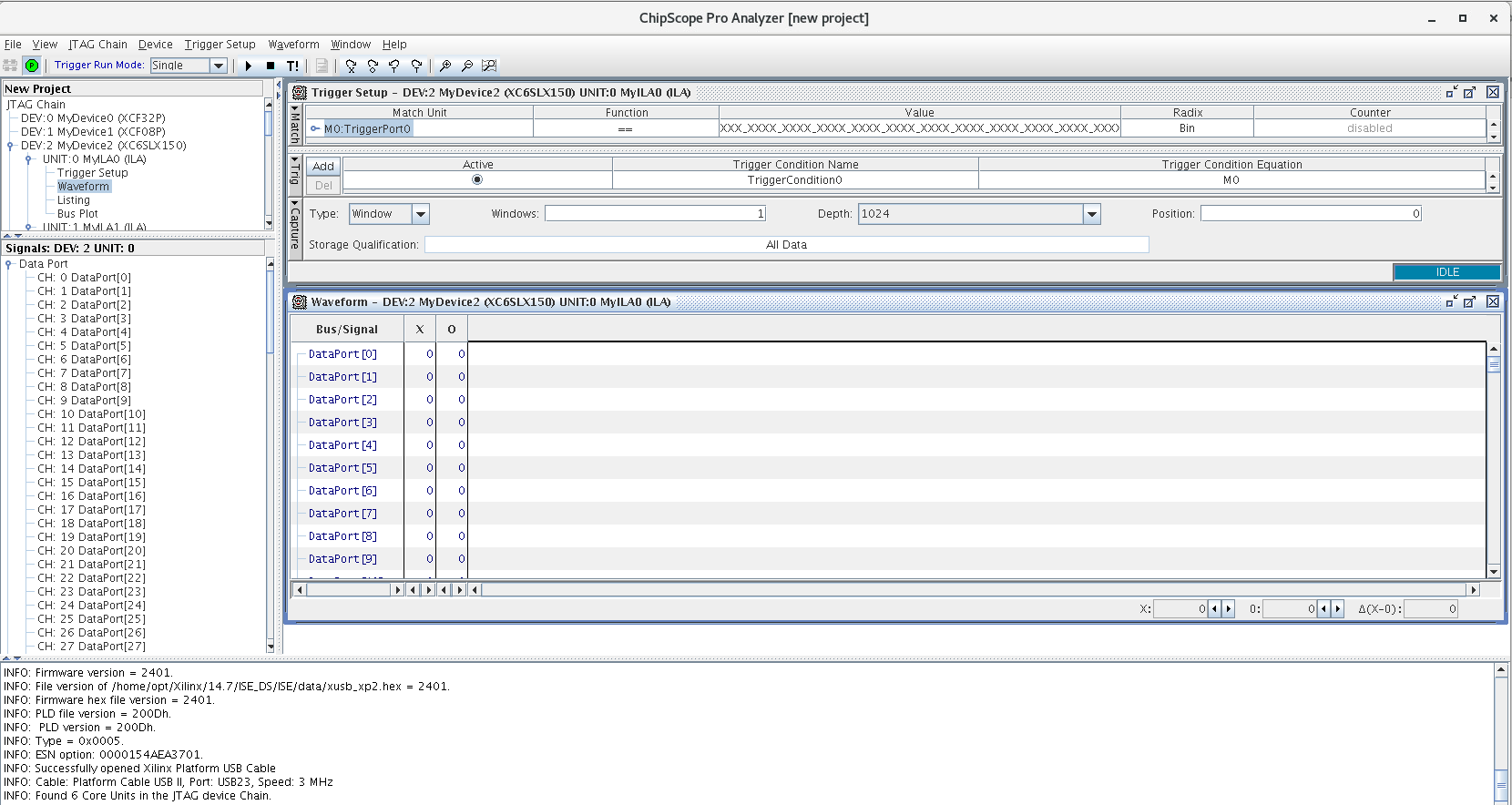
Refer to this spreadsheet to find the cable number- <https://docs.google.com/spreadsheets/d/1s-ktpfiLgkcWlEr5HXKpSdO-Cmn3PE66lwZ5NXgxGcA/edit#gid=1978861311>

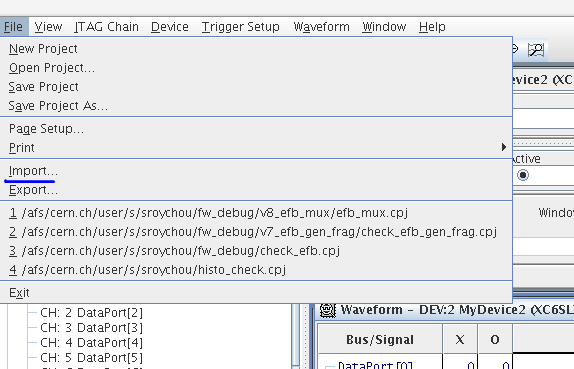


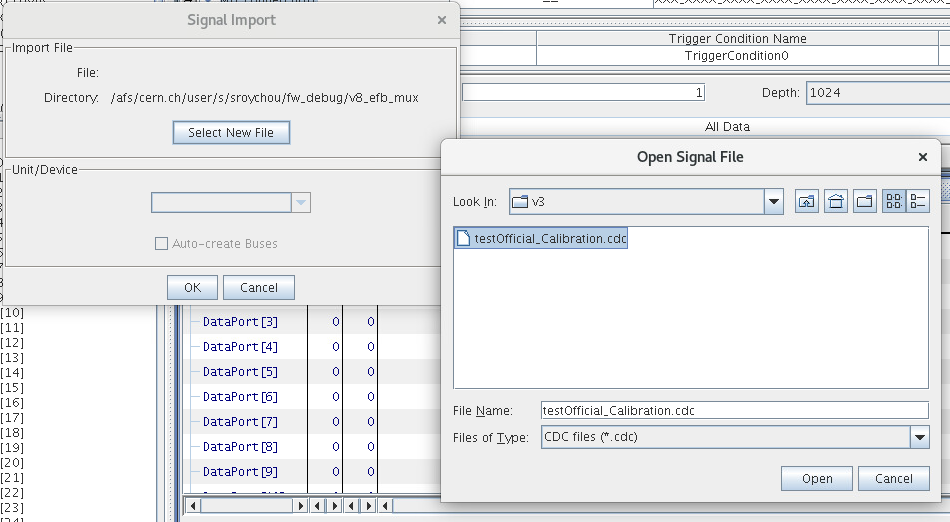
The devices shown here represent all the FPGAs in a ROD system, not just the ROD slave we are testing. Press ok.



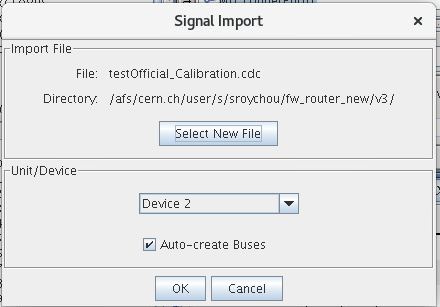
The window will look like this. Note that there are no specific signal names and all of them are named dataport[x]



1. Import CDC. The .cdc file created earlier has to be imported for signal names to show. File->Import, then follow the prompts as shown

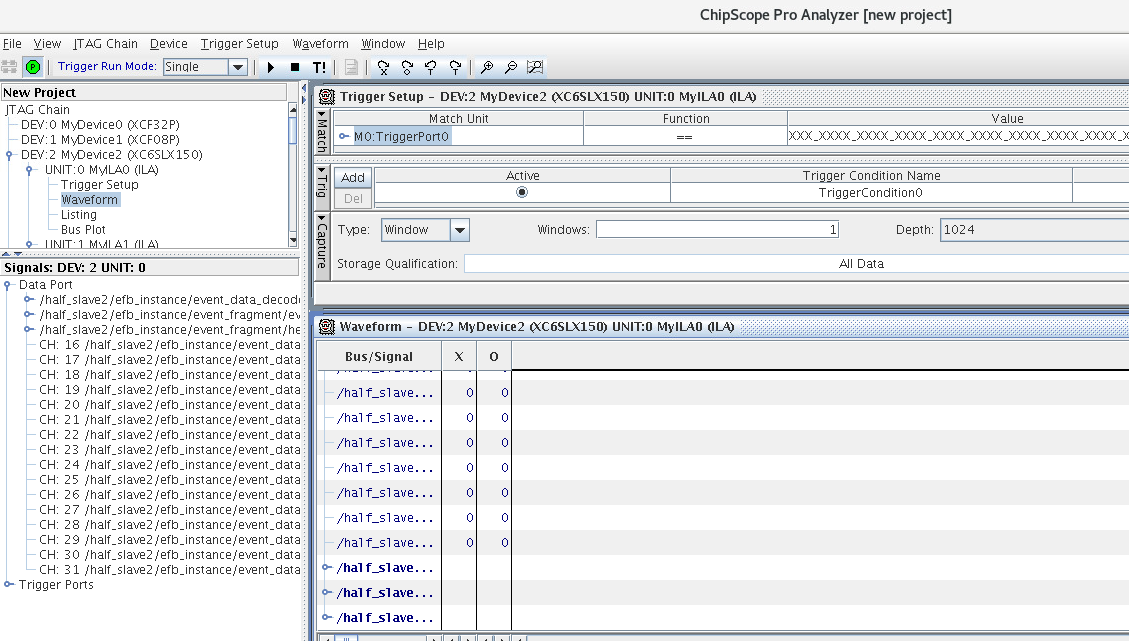


Make sure auto-create buses is selected. This will ensure that buses are grouped together which is easier to manage and see.

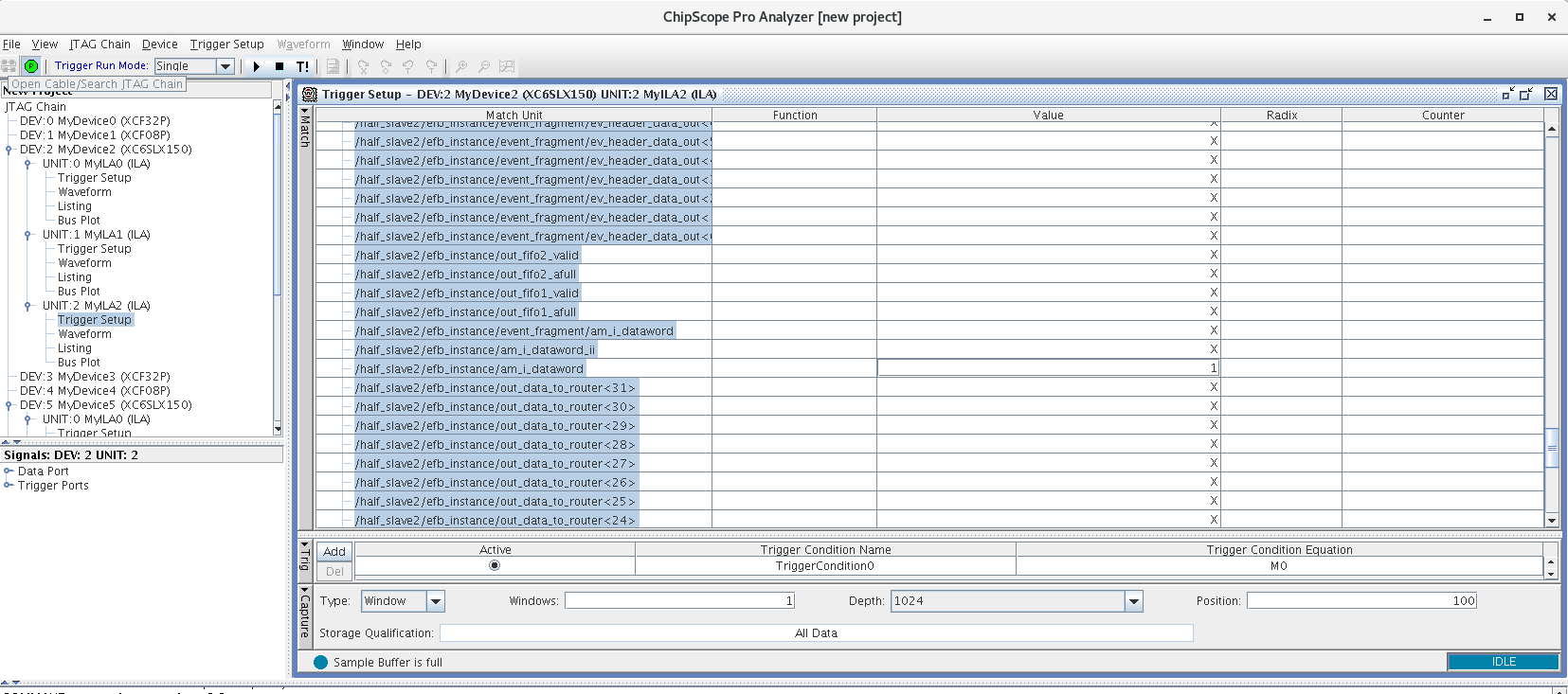


Now the signal names should be visible.

1. Trigger condition. Here Device 2 corresponds to a ROD Slave, so we need to set up trigger conditions from this device



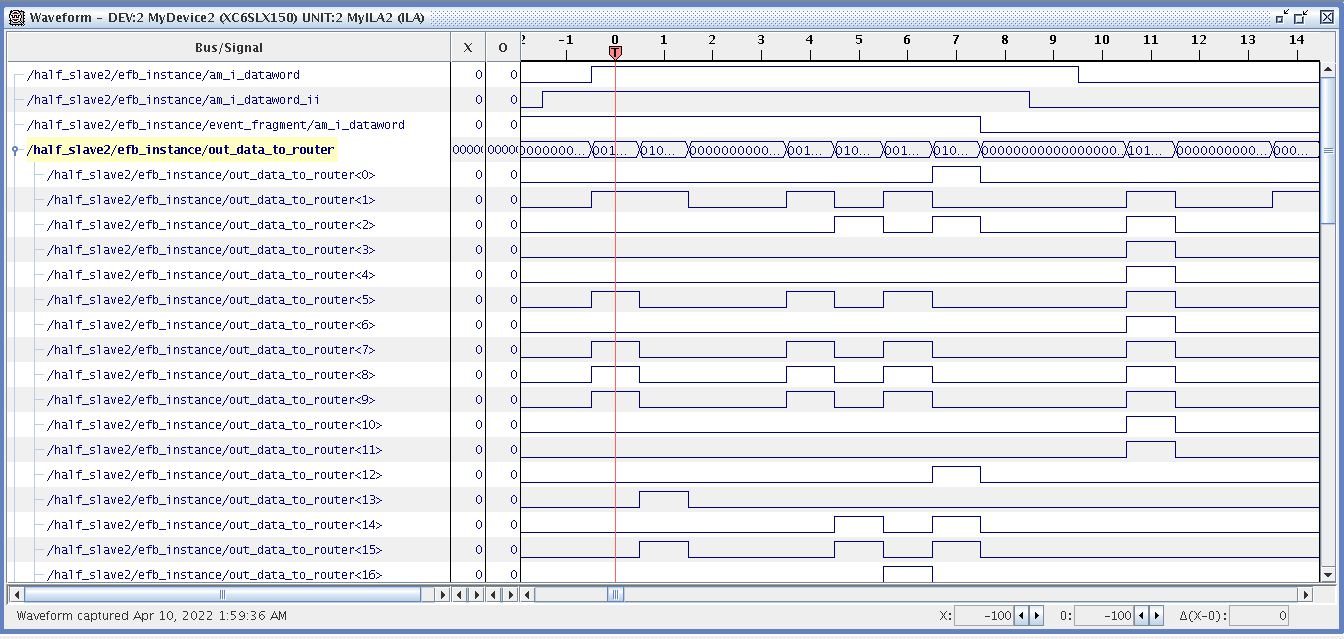
On the left, after expanding Device 2, all the ILAs in the device will show up. Here there are 3. Here we’ll set up a trigger condition in ILA2. Click on trigger setup to see the list of signals available (that would correspond to the cdc file created). Here the condition is that am\_i\_dataword should go to 1. All other values are set to X which means that there is no condition det for them. Possible values are 0, 1, R(rise), F(fall).



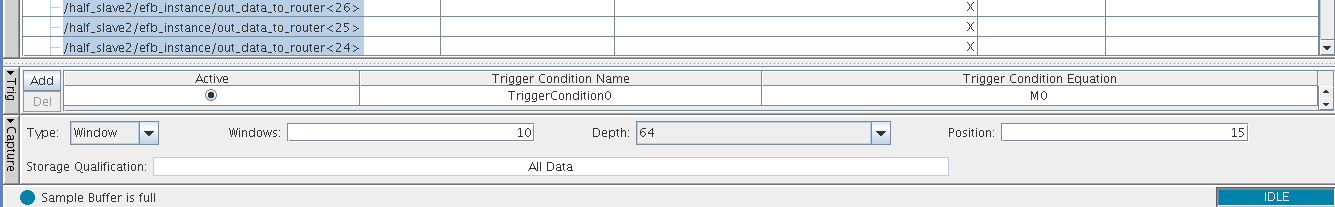
1. Start looking at signals. The play button at the top starts waiting for the trigger and waveforms will show up once the condition is met. The stop button is to stop scanning. T! can be used to take a real time snapshot of all the signals in the ILA irrespective of the trigger condition. If running on DAQSlice for example, press play here and then start the run in DAQSlice.



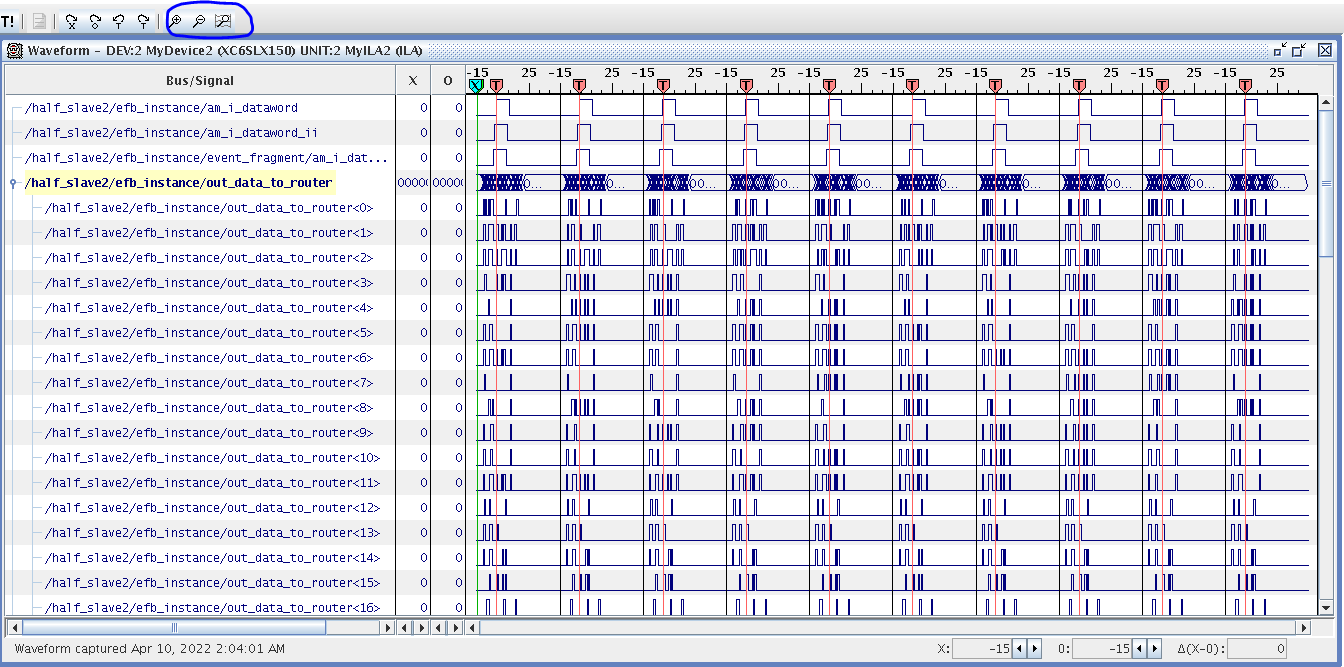
Waveform appears when trigger condition is met



It is also possible to set up windows to capture multiple occurrences of the trigger. The depth is the size of the window, and position is where is the window the trigger would be in the window, so a few cycles before/after the trigger condition can be seen.



Waveform with windows. The zoom in and out buttons can be used to look at signals more clearly.



The project settings can also be saved as a .cpj file.