Version 0.01a For EDK 6.2i, ChipScope Pro Software v6.2i November 30, 2004

Words of Caution

ChipScope allows one to debug signals in a programmed FPGA. Tempting as it may be, ChipScope should be used sparingly and only as a last resort. Ensure that the problem cannot be uncovered using simulation techniques. Debugging with ChipScope can be quite time consuming.

Goals

- Learn one of the several ways to insert a ChipScope module into a Verilog design in the EDK.
- Learn how to use the ChipScope analyzer to view signals.

Preparation

Have a quick look at the introduction in *ChipScope Pro Software and Cores User Manual*. The sections of interest are the *ChipScope Pro Tools Description* and the descriptions of the ICON and ILA cores.

Background

There are several methods that can be used to insert a ChipScope module into a design, both pre and post-synthesis. The ChipScope package also includes several different cores specialized to debug different things. As an introduction, this document focuses on one pre-synthesis method to insert a ChipScope module, and uses the simplest and most general set of cores.

The two cores that will be inserted in the design will be the ICON core and the ILA core.

ICON The ChipScope module communicates with a PC via the JTAG connection. The ICON is the core that provides this connectivity. Each ICON core can be connected to 15 ILA cores.

ILA The ILA is the core connected to the signals to be observed. The ILA will record a predetermined amount of data when the trigger condition is met. The ILA has a resolution of one sample per clock cycle.

Setup

This document assume that you have a EDK project containing the following:

- A custom core written in Verilog within which a ChipScope module is to be inserted.

Step-by-step

Generating the ChipScope cores

1. Create a temporary directory within which the generated cores can be placed and start up the ChipScope Pro Core Generator.
2. The first type of core to generate is the ICON. Select ICON (Integrated Controller) and click Next.
3. Select the appropriate directory that was created earlier to place the Output Netlist. Select the Device Family of the FPGA within which your design is to be placed. The number of control ports corresponds to the number of ILAs that will be connected to the ICON. Select 1. Click Next.

4. By default the generator will produce example HDL files as templates to insert the cores in to your design. Change the HDL language to Verilog and click Generate Core. The ICON core will now be ready in the directory you selected with files starting with "icon". Click Start Over.

5. Select ILA (Integrated Logic Analyzer) and click Next.

6. The appropriate directory, within which you placed the ICON core, should already be selected in the General Options dialog screen. Alter the Clock Settings section to select upon which edge the data should be sampled. For most purposes, sampling on the rising clock edge is most appropriate. Click Next.

7. The trigger selection dialog allows you to create multiple triggers. Triggers can be signals of any bit width. The value of the trigger on which ChipScope begins to record is specified during analysis. The types of allowable matches are determined here. The Basic match type is often sufficient. Other match types allow matching the rising and falling of individual bits and matching the range of the signal value. Select the desired settings. One trigger should be sufficient. Click Next.

8. The next dialog box specifies the bit width of the data that will be recorded by ChipScope as well as the amount of data that will be recorded. Change the data width to the number of bits you which to record. If you which to sample multiple signals from your design, the Data width must be equal to the sum of signal widths. You may also which to sample the signal used to trigger the ILA. If so, add its width to the data width since its values are not recorded automatically.

   The Data Depth specifies the number of samples that the ILA will record. One sample is recorded each clock cycle. The number of Block RAMs required is indicated. Check how many your design occupies and your FPGA datasheet for the number available as this may restrict the maximum number of samples. After selecting the appropriate values click Next.

9. Change the HDL Language to Verilog and click Generate Core. The two necessary cores have now been generated. The ChipScope Core Generator can now be closed.

Inserting the ChipScope module into the EDK

10. The files of interest that ChipScope Core Generator created are the *.v and the *.edn files. Open `icon_xst_example.v` and `ila_xst_example.v`. Within each file you will find an a sample instantiation of the core followed by a declaration of the core. Copy the declaration into a separate file for each core and save them as `icon.v` and `ila.v`.

11. Add the two .v files you created and the generated .edn files to your custom EDK core. Place the `ila.v` and `icon.v` files in the `pcores/yourcustomcore/hdl/verilog` directory. Place the `ila.edn` and `icon.edn` files in the `pcores/yourcustomcore/netlist` directory. The netlist directory may not exist if you do not have any netlist components in your design. If it does not exist, create it.

12. Edit the file in the `pcores/yourcustomcore/data` directory to include the files your added.

    (a) If your design already contained netlist components, the .mpd file does not need to be modified. Otherwise, add `OPTION STYLE = MIX` to the options section of the file. This indicates that the design consists of both HDL and netlist components.

    (b) Edit the .pao file to contain references to `icon.v` and `ila.v` by adding lines similar to

    ```
    lib yourcustomcore icon
    lib yourcustomcore ila
    ```

    at the top of the document (the lines must precede the HDL file that contains their instantiation).

    (c) Edit the .bbd file to contain references to `icon.edn` and `ila.edn`. If your design does not have any netlist components the .bbd file may not exist. In this case the minimum content of the .bbd file is

```
Files
icon.edn,ila.edn
```

13. The cores can now be instantiated within the designs HDL code. Examples of the each cores instantiation can be found in the generate files `icon_xst_example.v` and `ila_xst_example.v`. The signals, for the most part, are self-explanatory. The output `control0` signal from the ICON core must be connected to the input `control` signal of the ILA core. Below is an example instantiation of the ILA core with multiple signals concatenated together to form the data input.

```
ila i_ila
  (
    .control(control0),
    .clk(clk),
    .data({4'b0,datax,datay,dataz}),
    .trig0(reset)
  );
```

As in the example above, extra bits in the data input should be filled. If they are not, an error will result during synthesis.

Removing Potential Conflicts

ChipScope Analyzer communicates with the ICON core via the JTAG connection. Often, other modules in a system also use the JTAG. A common example of this is the opb_mdm core which communicates with the XMD debugger. A system containing both will result in an over-mapping of the BSCAN component of the FPGA. A simple solution is to remove conflicting modules. Below are steps specific to removing the opb_mdm core.

14. Open the system being tested with the Xilinx EDK. From the menu, select Project→Add/Edit Cores.... Within the Peripherals tab, select opb_mdm and click the Delete>> button.

15. From the menu, select Project→Software Platform Settings. Move to the Processor and Driver Parameters Tab. Within the Processor Parameters table, select None as the current value for the xmdstub_peripheral instance. Click OK.

16. Without the debug module, software applications cannot be run in XmdStub mode and must be put in Executable mode. This means that they will begin execution immediately after the system has successfully downloaded to the FPGA. Since you will be unable to control the execution of your software application via XMD, you may wish to alter your software so that it loops. You may also need to disable the initialization of the xmdstub software to the BRAMs.

17. At this point, the system with ChipScope is ready to be downloaded to the FPGA.

Gathering Data

18. With the system downloaded to the FPGA, open ChipScope Pro Analyzer. From the menu, selection JTAG Chain→Xilinx Parallel Cable. A dialog box will open with appropriate default settings. Click OK. A second dialog box will appear listing the devices found on the JTAG connection. Click OK.

    The ChipScope Pro Analyzer screen should fill with two Windows: Trigger Setup and Waveform.

19. From within the Trigger Setup window, set up the trigger condition by manipulating the table. Alter the matching function and value to your preference.

20. The Waveform window displays a wave for each bit of the data input for the ILA by default. This can be changed so that values of multi-bit signals can be examined. To do this, bits are grouped together in "Buses". From within the Waveform window, selecta all the DataPort bits that correspond to a signal. Right-click on the selection and from the pop-up menu, select Add to Bus→New Bus. A new wave corresponding to the signal will be added to the window.

21. After the trigger and the Waveform window have been setup, ChipScope is ready to collect data. From the menu, select Trigger Setup→Run. ChipScope will wait for the trigger condition to be met and soon after, the collected data will appear in the Waveform window ready to be analyzed.