Overview

ChipScope is an embedded, software based logic analyzer. By inserting an “integrated controller core” (icon) and an “integrated logic analyzer” (ila) into your design and connecting them properly, you can monitor any or all of the signals in your design. Even nicer is that ChipScope provides you with a convenient software based interface for controlling the “integrated logic analyzer,” including setting the triggering options and viewing the waveforms. ChipScope has a few disadvantages however. First, because it is synchronous ChipScope cannot be used to examine clock signals. Second, because it uses the SRAM on the Virtex part, it cannot capture very many samples.

There are six main steps to using ChipScope, as detailed below.

1. Setup a CORE Generator project
2. Generate an “integrated controller core” or icon
3. Generate one or maybe more “integrated logic analyzers” or ilas
4. Connect the ilas to the icon and make all of these modules part of your design.
5. Synthesize, and implement your design (including the icon and ilas) as normal.
6. Program the Calinx board
7. Run the ChipScope software to access and use the ilas (the ChipScope software requires the icon to gain access to the ilas)

Detailed Instructions: Step 1 – Creating a new Project

2. Open File ➔ New Project.
   a. Choose where to save project files.
      i. Create a new directory and open it.
      ii. Save the coregen.cgp file in this directory.
      iii. This is where your ICON and ILA cores (and their metadata files) will be generated.
   b. Under the Part tab, select all of the same device settings as you do when you setup a normal Xilinx ISE Project.
   c. Under the Generation tab, change Design Entry to Verilog.
   d. Select Ok to return to the main screen.
3. On the main screen, go to the View By Function tab.
   a. Select the Debug and Verification folder.
b. Select **Chip Scope Pro**.

4. You should now be looking at a list of Chip Scope “cores.” Among them should be ICON, ILA, and VIO. We will be using ICONs and ILAs.

**Detailed Instructions: Step 2 – Generating the ICON**

1. Double-click on ICON (ChipScope Pro – Integrated Controller). A window that will allow you to customize your ICON should have appeared.
   a. **Component Name**: Assign your ICON a name (this is arbitrary).
   b. Select the correct **Number of Control Ports**.
      i. Each ILA requires one control port
      ii. Normally you will only need **1 Control Port**
      iii. **If you generate an ICON with multiple control ports, you must connect every control port to an ILA.** If you have “hanging” control ports in your design, Xilinx ISE will fail during synthesis.
   c. Leave **Disable Boundary Scan** unchecked.
   d. Leave **Enable Unused Boundary Scan Ports** unchecked.
2. The ChipScope Pro Core Generator will now generated the ICON core according to the settings you specified. If you have errors go back and make sure you followed the above instructions.
3. Click **Finish** to Return to the main screen.
4. Your ICON will have been created in the directory you specified to store your project.

**Detailed Instructions: Step 3 – Generating the ILA**

1. Once again on the main screen, double-click on ILA (ChipScope Pro – Integrated Logic Analyzer). A window that will allow you to customize your ILA should have appeared.
   a. **Component Name**: Same as with the ICON, you may set this to whatever you like. As the semester progresses, you will see what an informative ILA naming convention entails.
   b. Normally you will only need **1 Trigger Port**
   c. Leave **Enable Output Trigger Port** unchecked.
   d. Leave **Sample On to Rising** (you will sample on the rising edge).
   e. Select the desired **Sample Data Depth**
      i. This is the number of samples the ILA will capture after it receives the trigger. It will capture one sample per clock cycle until it captures this many samples. You will probably need relatively few for Lab5.
   f. Leave **Data Same as Trigger** unchecked.
      i. The bench logic analyzers in the lab have 16bits of input used for both the data and the trigger. The ILA can have separate data and trigger inputs. **For clarity’s sake this semester, always have different trigger and data ports.** In Lab5, if you use a “detected error” signal then you should connect that to the trigger and connect the counter output to the data port.
g. **Data Port Width** is only available when you are using separate trigger and data ports. Set this to the number of bits you will need to see in the wave window of ChipScope.

h. **(CLICK NEXT TO GO TO THE NEXT PAGE)**

i. Set the **Trigger Port Width** to any amount you desire (probably either 1 bit or 32 bits for Lab5)

j. You will most likely need 1 Match Unit.

k. Leave **Counter Width** to Disabled.

l. Leave **Match Type** set to basic.

2. The ChipScope Pro Core Generator will now generated the ICON core according to the settings you specified. If you have errors go back and make sure you followed the above instructions.

3. Click **Finish** to Return to the main screen.

4. Your ILA will have been created in the directory you specified to store your project.

**Detailed Instructions: Step 4 – Connecting the Cores to Your Design**

1. Declare a control bus for each ILA, similar to the following:
   a. `wire [35:0] ILAControl;
   b. Remember, each ILA you want to add to your design will require a control bus.
   c. You can route control busses as input/outputs if you want to instantiate the ICON somewhere other than where you instantiate the ILA.

2. Instantiate the ICON core
   a. `icon i_icon(.CONTROL0(ILAControl));
   b. Remember to only instantiate **ONE** ICON, your design can never have more than one.

3. Instantiate the ILA core
   a. `ila i_ila (.CLK(Clock), .CONTROL(ILAControl), .TRIG0/**/), .DATA/**/);
   b. You may instantiate ILAs wherever they are necessary, just make sure to route the control bus from the ICON appropriately
   c. You may have one or more ILAs in your design.
   d. **Your ILA may have different ports** be sure to read the verilog example produced by the ChipScope Core Generator.

4. On both the ILA and ICON core instantiations, add **Synthesis no-prune directives**. Since the ICON and ILA do not output anything from your design, Synplify Pro will think that they have no impact on the circuit and will try to “prune” or remove them. Below are some example ICON/ILA instantiations that have the necessary synthesis directives added:

   ```verilog
   icon_1 icon(.CONTROL0(ILAControl)) /* synthesis syn_noprune=1 */;
   ila_1 ila (.CLK(Clock),
              .CONTROL(ILAControl),
              .TRIG0/**/),
              .DATA/**/)) /* synthesis syn_noprune=1 */;
   ```
Note that synthesis directives are normal block comments in Verilog, placed after the instantiation but before its closing semi-colon.

5. **REMEMBER TO LOOK AT THE EXAMPLE VERILOG FILES GENERATED BY CHIPSCOPE CORE GENERATOR!**

**Detailed Instructions: Step 5 – Synthesize and Implement Your Design**

1. Make sure to **add the verilog examples** generated by ChipScope Core Generator to your Xilinx ISE project as **Verilog Design Files**. (this is done through the **Add Source** option that you have been using to add modules to your design up to this point).
2. Make sure you set your **Macro Search Path** so that the search path directory contains all of the metadata files created for your cores.
   a. In the **Sources for** combo box, select **Implementation**.
   b. Click on **FPGA_TOP2** in the **Sources for** box.
   c. In the **Processes for** box, right click on **Implementation**.
   d. Under **Translate Properties**, set the **Macro Search Path** to the directory containing the ICON/ILA core files.
   e. Note that the core files must be **in the exact directory specified**. The Macro Search Path will not search through subdirectories.
3. Synthesize and implement your design as normal.

**Detailed Instructions: Step 6 – Program the CaLinx Board**

1. Make sure that the the **Parallel Cable IV** is connected to the **JTAG** Port on the CaLinx board and that the CaLinx board is on. **Make sure that you connect the Parallel Cable IV to the JTAG port and NOT the SlaveSerial port.**
   a. The little light on the Parallel Cable IV will turn green when the cable detects that it is connected to a powered Xilinx chip.
2. Run **iMPACT** from **Xilinx ISE**
   a. When it asks select **FPGA_TOP2.bit** as before, **Right click** on the picture of the FPGA and select **program**
3. **YOU MUST CLOSE IMPACT OR CHIPSCOPE WILL NOT BE ABLE TO WORK!**

**Detailed Instructions: Step 7 – Run ChipScope**

1. Open **Start** → **Programs** → **Xilinx ISE Design Suite 10.1** → **ChipScope Pro** → **Analyzer**.
2. Make sure that the the **Parallel Cable IV** is connected to the **JTAG** Port on the CaLinx board and that the CaLinx board is programmed, the **DONE** LED will light up green when the board is programmed.
   a. The little light on the Parallel Cable IV will turn green when the cable detects that it is connected to a powered Xilinx chip.
3. Once ChipScope Pro Analyzer is running you must connect to the parallel cable.
   a. Go to the **JTAG Chain** menu
   b. Select **Xilinx Parallel Cable**
c. Select the **Xilinx Parallel Cable IV**

d. Set the **Speed** to 5MHz

e. Make sure the **Port** is set to **LPT1**

f. Click **OK**

g. In the next window you will see two chips listed

i. The **System_ACE-CF** is not used in this class

ii. The **XCV2000E** is the FPGA.

h. Click **OK**

4. The ChipScope Pro Analyzer should now be connected to the FPGA and running.

a. You can move the **Trigger Setup** and **Waveform** windows around as needed to be able to see the information you’re looking for.

b. First you must use the **Trigger Setup** window to set a **trigger function**, just like with the Bench Logic Analyzers

c. When you have a trigger, click the **Run** button in the toolbar to start waiting for that trigger

d. When the **trigger occurs** ChipScope will start downloading data from the FPGA and show it in the **Waveform** window, much like ModelSim.

5. **Please experiment with ChipScope**, it is an invaluable tool for FPGA debugging.

<table>
<thead>
<tr>
<th>RevC – 9/30/2008</th>
<th>Chris Fletcher</th>
<th>Rewrote tutorial to be Xilinx ISE 10.1 compliant; major edits</th>
</tr>
</thead>
<tbody>
<tr>
<td>RevB – 6/30/2004</td>
<td>Greg Gibeling</td>
<td>Added mention of ChipScope’s drawbacks; minor editing</td>
</tr>
<tr>
<td>RevA</td>
<td>Greg Gibeling</td>
<td>Original</td>
</tr>
</tbody>
</table>