Introduction to Xilinx ChipScope Pro Tool

Part 2 – Core Instantiation Flow

EE254L/EE560
Gandhi Puvvada
USC Viterbi School of Engineering
Two flows: Core Generator and Core Inserter

Part 2: ChipScope Pro Core Generator
- Generate...
  - ICON, ILA, IBA/OPB, IBA/PLB, VIO, or ATC2 cores
- Instantiate...
  - cores into HDL source
- Synthesize...
  - design with cores in it
- Connect...
  - busses and internal signals to cores

Part 1: ChipScope Pro Core Inserter
- Synthesize...
  - design without instantiating ChipScope cores
- Insert...
  - ICON, ILA, and/or ATC2 cores into synthesized design (.ngc or EDIF netlist)

ISE
- Implement...
  - design
Create a project and add New Source
Select IP

Create Coregen or Architecture Wizard IP Core.

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>AXI4</th>
<th>AXI4-Stream</th>
<th>AXI4-Lite</th>
<th>Status</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automotive &amp; Industrial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXI Infrastructure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BaseIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Elements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication &amp; Networking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug &amp; Verification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXI Bus Functional Model</td>
<td>2.1</td>
<td></td>
<td></td>
<td></td>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>Debug</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATC2 (ChipScope Pro - Agilent Trace Core 2)</td>
<td>1.05.a</td>
<td></td>
<td></td>
<td></td>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>AXI Chipscope Monitor</td>
<td>3.05.a</td>
<td></td>
<td></td>
<td></td>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>ICON (ChipScope Pro - Integrated Controller)</td>
<td>1.06.a</td>
<td></td>
<td></td>
<td></td>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>ILA (ChipScope Pro - Integrated Logic Analyzer)</td>
<td>1.05.a</td>
<td></td>
<td></td>
<td></td>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>VIO (ChipScope Pro - Virtual Input/Output)</td>
<td>1.05.a</td>
<td></td>
<td></td>
<td></td>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>Digital Signal Processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embedded Processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPGA Features and Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Math Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memories &amp; Storage Elements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:\Xilinx_projects\ee560_eee2011_detour_Chipscope_lab\EE560_detour_Chipscope_ILA_instantiation_lab_sol\pcore_dir
Source Type: IP (CORE Generator & Architecture Wizard)
Source Name: ILA_for_detour.xco
Core Type: ILA (ChipScope Pro - Integrated Logic Analyzer); Version: 1.05.a
ILA Core configuration before generation

ILA (ChipScope Pro - Integrated Logic Analyzer)

- Component Name: ILA_for_detour
- Generate Example Design
- Trigger Port Settings:
  - Number Of Trigger Ports: 6
  - Max Sequence Levels: 6
- Use RPMs
- Enable Trigger Output Port
- Storage Settings:
  - Sample On: Rising
  - Sample Data Depth: 1024
- Enable Storage Qualification
- Data Same As Trigger
- Data Port Width: 16

Minimum option highlighted: Sample Data Depth
ILA configuration continued

We chose
The LogiCORE™ IP ChipScope™ Pro Integrated Logic Analyzer (ILA) core is a customizable logic analyzer core that can be used to monitor any internal signal of your design. The ILA core includes many advanced features of modern logic analyzers, including boolean trigger equations, trigger sequences, and storage qualification. Because the ILA core is synchronous to the design being monitored, all design clock constraints that are applied to your design are also applied to the components inside the ILA core.

This step takes some time!
After the core is configured
HDL Functional model

```vhls
-- This wrapper is used to integrate with Project Navigator and PlanAhead
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ILA_for_detour IS
  port (
    CONTROL: inout std_logic_vector(35 downto 0);
    CLK: in std_logic;
    DATA: in std_logic_vector(15 downto 0);
    TRIG0: in std_logic_vector(3 downto 0);
    TRIG1: in std_logic_vector(0 to 0);
    TRIG2: in std_logic_vector(0 to 0);
    TRIG3: in std_logic_vector(0 to 0);
    TRIG4: in std_logic_vector(0 to 0);
    TRIG5: in std_logic_vector(2 downto 0));
END ILA_for_detour;
ARCHITECTURE ILA_for_detour_a OF ILA_for_detour IS
BEGIN
END ILA_for_detour_a;
```
HDL Instantiation Template

```vhdl
-- Component Identifier: xilinx.com:ip:chipscope_ila:1.05.a
-- The following code must appear in the VHDL architecture header:

--------- Begin Cut here for COMPONENT Declaration -------- COMP_TAG

component ILA_for_detour

PORT (  
    CONTROL : INOUT STD_LOGIC_VECTOR(35 DOWNTO 0);  
    CLK : IN STD_LOGIC;  
    DATA : IN STD_LOGIC_VECTOR(15 DOWNTO 0);  
    TRIG0 : IN STD_LOGIC_VECTOR(0 TO 0);  
    TRIG1 : IN STD_LOGIC_VECTOR(0 TO 0);  
    TRIG2 : IN STD_LOGIC_VECTOR(0 TO 0);  
    TRIG3 : IN STD_LOGIC_VECTOR(0 TO 0);  
    TRIG4 : IN STD_LOGIC_VECTOR(0 TO 0);  
    TRIG5 : IN STD_LOGIC_VECTOR(2 DOWNTO 0));

end component;

-- COMP_TAG_END -------- End COMPONENT Declaration ----------
-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.
--------- Begin Cut here for INSTANTIATION Template ------ INST_TAG

your_instance_name : ILA_for_detour

port map (  
    CONTROL => CONTROL,  
    CLK => CLK,  
    DATA => DATA,  
    TRIG0 => TRIG0,  
    TRIG1 => TRIG1,  
    TRIG2 => TRIG2,  
    TRIG3 => TRIG3,  
    TRIG4 => TRIG4,  
    TRIG5 => TRIG5);
```
Now add another new source
ICON IP
Select IP

Create Coregen or Architecture Wizard IP Core.

View by Function | View by Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>AXI4</th>
<th>AXI4-Stream</th>
<th>AXI4-Lite</th>
<th>Status</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automotive &amp; Industrial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXI Infrastructure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BaseIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Elements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication &amp; Networking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug &amp; Verification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXI Bus Functional Model</td>
<td>2.1</td>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATC2 (ChipScope Pro - Agilent Trace Core 2)</td>
<td>1.05.a</td>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXI Chipscope Monitor</td>
<td>3.05.a</td>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICON (ChipScope Pro - Integrated Controller)</td>
<td>1.06.a</td>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILA (ChipScope Pro - Integrated Logic Analyzer)</td>
<td>1.05.a</td>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIO (ChipScope Pro - Virtual Input/Output)</td>
<td>1.05.a</td>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital Signal Processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embedded Processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPGA Features and Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Math Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memories &amp; Storage Elements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Search IP Catalog: | Clear

All IP versions

Only IP compatible with chosen part

Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:\\linx_projects\EE560_ee2011_detour_Chipscope_lab\EE560_detour_Chipscope_ILA_instantiation_lab_sol\ipcore_dir
Source Type: IP (CORE Generator & Architecture Wizard)

Source Name: ICON_for_detour.xco
Core Type: ICON (ChipScope Pro - Integrated Controller); Version: 1.06.a

Finish
Generally we do not need to do much with the ICON
Introduction

The LogiCORE™ IP ChipScope™ Pro Integrated CONtroller core (ICON) provides an interface between the JTAG Boundary Scan (BSCAN) interface of the FPGA device and the ChipScope Pro cores, including the following types of cores:

- Integrated Logic Analyzer (ILA)
- Virtual Input/Output (VIO)
- Agilent Trace Core 2 (ATC2)
- Integrated Bus Analyzer (IBA)

This interface allows the ChipScope Pro Analyzer software to communicate with these cores through the JTAG port of the device. The ICON core is designed to be easily instantiated and connected to these cores directly in a Verilog or VHDL design. The ICON core can also be added to an embedded processor system design using the Xilinx Embedded Development Kit (EDK) tools.

<table>
<thead>
<tr>
<th>Supported Device Family(1)</th>
<th>Kintex®-7, Virtex®-7, Virtex-6/6L,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Virtex-5, Spartan®-6, Spartan-3E,</td>
</tr>
<tr>
<td></td>
<td>Spartan-3A, Spartan-3A DSP, Spartan-3E,</td>
</tr>
<tr>
<td></td>
<td>Spartan-6, XA Virtex-4, Spartan-6Q/6QL, Virtex-6Q, Virtex-6Q/6QL</td>
</tr>
</tbody>
</table>

| Supported User Interfaces | Not applicable. |

<table>
<thead>
<tr>
<th>Provided with Core</th>
<th>Resources</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration(2)</td>
<td>LUTs</td>
<td>FFs</td>
</tr>
<tr>
<td>Config1</td>
<td>190</td>
<td>198</td>
</tr>
<tr>
<td>Config2</td>
<td>317</td>
<td>335</td>
</tr>
<tr>
<td>Config3</td>
<td>541</td>
<td>559</td>
</tr>
</tbody>
</table>
After the ICON core is generated
HDL Functional Model

```
-- This wrapper is used to integrate with Project Navigator and PlanAhead
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ICON_for_detour IS
  port (CONTROL0: inout std_logic_vector(35 downto 0));
END ICON_for_detour;
ARCHITECTURE ICON_for_detour_a OF ICON_for_detour IS
BEGIN
END ICON_for_detour_a;
```
--- The following code must appear in the VHDL architecture header ---

--------------- Begin Cut here for COMPONENT Declaration ----------

component ICON_for_detour

PORT ( 
  CONTROL0 : INOUT STD_LOGIC_VECTOR(35 DOWNTO 0));

end component;

--------------- End COMPONENT Declaration ---------------------

--- The following code must appear in the VHDL architecture body. Substitute your own instance name and net names. ---

--------------- Begin Cut here for INSTANTIATION Template ------ INST_TAG

your_instance_name : ICON_for_detour

port map ( 
  CONTROL0 => CONTROL0);

--------------- End INSTANTIATION Template ---------------------
detour_top.vhd revisions

```
-- Additional signal declaration to help with the instantiation of an ILA and an ICON.
signal control_bus : std_logic_vector(35 downto 0);
signal data_to_sample : std_logic_vector(15 downto 0);

data_to_sample(0) <= divclk(25); -- the system clock
data_to_sample(1) <= detour_signals(0);
data_to_sample(2) <= detour_signals(2);
data_to_sample(3) <= detour_signals(4);
data_to_sample(4) <= detour_signals(6);
data_to_sample(8 downto 5) <= state_encoded;
data_to_sample(12 downto 9) <= JC_copy;
data_to_sample(13) <= sw1;
data_to_sample(14) <= sw2;
data_to_sample(15) <= sw3;

```

```
component ICON_for_detour
PORT (  
  CONTROL0 : INOUT STD_LOGIC_VECTOR(35 DOWNTO 0));
end component;

component ILA_for_detour
PORT (  
  CONTROL : INOUT STD_LOGIC_VECTOR(35 DOWNTO 0);
  CLK : IN STD_LOGIC;
  DATA : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
  TRIG0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  TRIG1 : IN STD_LOGIC_VECTOR(0 TO 0);
  TRIG2 : IN STD_LOGIC_VECTOR(0 TO 0);
  TRIG3 : IN STD_LOGIC_VECTOR(0 TO 0);
  TRIG4 : IN STD_LOGIC_VECTOR(0 TO 0);
  TRIG5 : IN STD_LOGIC_VECTOR(2 DOWNTO 0));
end component;

ILA_1 : ILA_for_detour
port map (  
  CONTROL => control_bus,
  CLK => divclk(24), -- to ?
  DATA => data_to_sample,
  TRIG0 => JC_Copy,
  TRIG1 => (others => sw1),
  TRIG2 => (others => sw2),
  TRIG3 => (others => sw3),
  TRIG4 => (others => sw4),
  TRIG5 => (sw3, sw2, sw1));
```
Chipscope is not part of the free WebPACK. Do “VPN” to get USC floating license.
Since there is no design-level .cdc file to import signal names from, in this design flow, first time, you need to rename all signals and from appropriate buses. Trigger conditions and storage qualifications are to be set at run-time in both flows. All this can be saved so that you would need to do only incremental changes as needed.
Next set Trigger/Trigger sequence

M1 \rightarrow !M1 \rightarrow M2 \rightarrow !M2 \rightarrow M3 \rightarrow !M3
Next set Capture (Storage) Conditions

Sw4 shall be in up position to allow capturing of data.

39 samples before trigger and 15 (= 64-39) samples after trigger.

Save project.
After signal renaming and setting trigger and storage qualifications
Arm trigger, simulate sequence of events and obtain waveform

But first verify that you can capture waveforms with no trigger (with immediate trigger). Make sure you have satisfied the storage qualifications (set SW4 to on (up) condition).

In our design, the trigger sequence is satisfied by sliding sw1 to up and then bringing it back (M1 -> !M1 ->) then sliding sw2 to up and then bringing it back (M2 -> !M2 ->), and then (M3 -> !M3).

Note: The STORAGE qualification applies during pre-trigger sampling also, when ILA is waiting for the trigger condition. You can try keeping sw4 down and try to arm trigger and it does not gather any samples and hence it does not trigger even if you toggle sw1, sw2, and sw3 as per the above trigger sequence requirement. This seems be a bug in the ILA behavior. We need to investigate further.
Applying trigger sequence and capturing waveforms