Introduction to Xilinx ChipScope Pro Tool

EE254L/EE560
Gandhi Puvvada
USC Viterbi School of Engineering
Topic list

• Basic Introduction

• Demo

• Advanced Topics
Logic Analyzer and On-Chip Logic Analyzer

• Logic Analyzer vs. Oscilloscope
  Digital
  32/64/128/256 channels
  Trigger and Storage qualifications

  Analog
  2 or 4 channels
  Simple trigger

• SoC (System on Chip) and Embedded Processors in SoC
PowerPC-based Embedded Design

- PowerPC 405 Core
- Dedicated Hard IP
- Flexible Soft IP
- RocketIO
- Full system customization to meet performance, functionality, and cost goals

- IBM CoreConnect™ on-chip bus standard PLB, OPB, and DCR

- Off-Chip Memory: ZBT SRAM, DDR SDRAM, SDRAM

From Xilinx EDK course slides
AXI (Advanced eXtensible Interface) in EDK

AXI Interconnect Core Diagrams

Figure 2-6 illustrates a top-level AXI Interconnect.

Figure 2-6: Top-Level AXI Interconnect
References
http://www.xilinx.com/tools/cspro.htm

ChipScope Pro Software and Cores User Guide
(ChipScope Pro Software 14.7)
Elements of ChipScope

• ChipScope Pro software

• ILA Core: Integrated Logic Analyzer
  Gathers and stores internal signals.

• ICON Core: Integrated Control Core
  Communicates with your PC
On-chip cores communicate via JTAG or USB port with the host PC

Parallel Cable or USB Cable
Integrated Logic Analyzer (ILA)

The Integrated Logic Analyzer is a parameterized soft core that you can embed in your designs.

It is used in conjunction with the ChipScope software to provide real-time, on-chip debugging and design verification.
One or more ILAs and a ICON
One ILA and a ICON
Triggering and Capture Qualifications
Triggering and Capture Qualifications

• Simple minded logic analyzer configurations capture all activity on waveforms
Triggering and Capture Qualifications

• Simple minded logic analyzer configurations capture **all** activity on waveforms

• **Triggering:** Start capturing only after a specific event has occurred.
Triggering and Capture Qualifications

• Simple minded logic analyzer configurations capture all activity on waveforms.

• Triggering: Start capturing only after a specific event has occurred.

• Capturing: Capture only what is of most useful for debugging. Example: Only I/O bus cycles associated with printer communication.
Figure 1-2: ChipScope Pro Tools Design Flow

HDL instantiation flow

CORE Generator Tool

Generate...
ICON, ILA, VIO, or ATC2 cores

Instantiate...
cores into HDL source

Synthesize...
design with cores in it

Connect...
buses and internal signals to cores

Netlist insertion flow

Synthesize...
design without instantiating ChipScope cores

PlanAhead Tool or Core Inserter

Insert...
ICON, ILA, and/or ATC2 cores into synthesized design (.ngc or EDIF netlist)

ISE

Implement...design

Select...
bitstream
Set...
trigger
View...
waveform
DEMO

- ILA core insertion
- Implement
- Invoke ChipScope Analyzer
ChipScope Pro requires License
A free WebPACK license is not enough.

For Xilinx tools
One ILA is enough for our design!
Design for Demo

EE201L Detour Lab design (slightly modified)

Instead of 7 states, we have 13 states.

Extra states: IS0, IS1, L123_DIS, L123_REA, R123_DIS, R123_REA
EE201L Detour lab (original)

<table>
<thead>
<tr>
<th>State</th>
<th>GL</th>
<th>G1</th>
<th>G2</th>
<th>GR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle State</td>
<td>◁</td>
<td>◁</td>
<td>◁</td>
<td>◁</td>
</tr>
<tr>
<td>R1 State (G1 is ON)</td>
<td>◁</td>
<td>◁</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>R12 State (G1, G2 are ON)</td>
<td>◁</td>
<td>◁</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>R123 State (G1, G2, GR are ON)</td>
<td>◁</td>
<td>◁</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Idle State</td>
<td>◁</td>
<td>◁</td>
<td>◁</td>
<td>◁</td>
</tr>
<tr>
<td>L1 State (G2 is ON)</td>
<td>◁</td>
<td>◁</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>L12 State (G2, G1 are ON)</td>
<td>◁</td>
<td>◁</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>L123 State (G2, G1, GL are ON)</td>
<td>◁</td>
<td>◁</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
### EE201L Detour lab (modified)

<table>
<thead>
<tr>
<th>State Description</th>
<th>GL</th>
<th>G1</th>
<th>G2</th>
<th>GR</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS0 (Idle Start 0) State</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IS1 (Idle Start 1) State</td>
<td>• • • • • • •</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I (Idle) State</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1 State (G1 is ON)</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R12 State (G1, G2 are ON)</td>
<td>•</td>
<td>• •</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R123 State (G1, G2, GR are ON)</td>
<td>•</td>
<td>• •</td>
<td>• •</td>
<td></td>
</tr>
<tr>
<td>R123_DIS State (all OFF)</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R123_REA State (G1, G2, GR are ON)</td>
<td>•</td>
<td>• • • • • • •</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1 State (G2 is ON)</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L12 State (G2, G1 are ON)</td>
<td>•</td>
<td>• •</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L123 State (G2, G1, GL are ON)</td>
<td>•</td>
<td>• • • • • • •</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L123_DIS State (all OFF)</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L123_REA State (G2, G1, GL are ON)</td>
<td>•</td>
<td>• • • • • • •</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Starting LAMP TEST by** 16 times flashin all lights 8

**Disappear/Reappear sequence 16 times** 4

**Disappear/Reappear sequence 16 times** 4
On Reset and again in Idle state, we clear the J counter to ZERO.

During Reset it remains in S0 state.

After Reset, it does S0 S1 eight times.

Then it does one of the two depending upon L/R.
Final Result!
Add New Source (.cdc file)

ChipScope Definition and Connection Source File
ChipScope Definition and Connection Source File

Select Source Type

- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: detour
Location:

Add to project

Next
Summary

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

- Add to Project: Yes
- Source Directory: C:\Xilinx_projects\ee560_ee201L_detour_Chipscope_labs\EE560_detour_Chipscope_labs
- Source Type: ChipScope Definition and Connection File
- Source Name: detour.cdc
- Association: detour_top

Finish
.cdc file added

ChipScope Definition and Connection Source File
Double-click on the .cdc file
One ICON is already added.
Add one new ILA unit.
Trigger Parameters

- **Number of Input Trigger Ports**: 6
- **Match Units**: 1
- **Trigger Width**: 4
- **Match Type**: Basic
- **Bit Values**: 0, 1, X
- **Functions**: -", <-'

**Trigger Condition Settings**
- Enable Trigger Sequencer
- Max Number of Sequencer Levels: 6

**Storage Qualification Condition Settings**
- Enable Storage Qualification
Capture Parameters

Sample on Rising or Falling edge?
Net Connections

- UNIT
  - CLOCK PORT
  - TRIGGER PORTS
  - DATA PORT

Modify Connections
Select signals for trigger and storage qualification
Choose right sampling clock

We have chosen divclk(24) as the sampling clock. This is 2 times faster than the system clock which is divclk(25). We did so because we wished to display divclk(25) itself by sampling it at double the rate on the negative edges of divclk(24). This is actually not the best way. Experienced designers do not need clock as the samples are at clock ticks! They can mentally visualize the clock.

\[
\text{clk} \leftarrow \text{divclk}(25); \quad -- \quad \text{divclk}(25) = (\sim1.5\text{Hz}) = (100\text{MHz} / 2^{*26})
\]
It is a Logic Analyzer!

• Digital (not Analog)

• Discrete Samples (not continuous)

• One sample per clock

• Waveform ➞ samples connected together
Can you see the clock?

• You can NOT see the sampling clock!

• To see system clock on the waveform, you need to use a trigger clock (sampling clock) with higher frequency (higher than the system clock, at least double).

• Most designers do not need to see the clock!
Choose right data for sampling

Note: We are trying to display the System Clock by sampling at double the rate!
Finish producing .cdc and Start “Analyzing your Design using ChipScope
Finish producing .cdc and Start “Analyzing your Design using ChipScope”

Completed producing the .bit file

Then connect your board to the PC, switch on your board and then Double-Click
Connect your board to the PC and identify the device on your board.

If clicking this does not work, then Expand JTAG chain and select Digilent USB JTAG Cable.
Configure your device
Display data and trigger ports
Display Trigger Setup and Waveform
Match units, Trigger and Capture
First
setup match units
Next set Trigger/Trigger sequence

Trigger Condition: TriggerCondition0

- **Boolean**
  - Number of Levels: 6
  - Use Contiguous Match Events Only: unchecked
  - Level | Match Unit | Negate
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>M1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>M2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>M2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>M3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>M3</td>
<td></td>
</tr>
</tbody>
</table>

- **Sequencer**

Trigger Condition Equation:

M1 -> !M1 -> M2 -> !M2 -> M3 -> !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.
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 \((M1 \rightarrow !M1 \rightarrow)\) then sliding sw2 to up and then bringing it back \((M2 \rightarrow !M2 \rightarrow)\), and then \((M3 \rightarrow !M3)\).

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
List Display and Analysis

All samples are duplicated because of our higher frequency sampling clock which is unnatural.

Drag from the waveform window and Drop into the list window.

OR
Some advanced topics

• Sampling on Rising or Falling edge

• Different Match Units

• Complex Trigger Conditions

• Virtual I/O
Sample on Rising or Falling edge?

• Ideally we should sample once per clock at the significant edge of the clock.

• Same edge sampling ➔ The Sampling process should not introduce delay. You should capture data at or just before the significant edge, **not** just after the significant edge. *If it is just after the edge, you will be capturing data in transition (data in turmoil)!*
Sample on Rising or Falling edge?

- Here we are sampling using high-speed clock (higher speed than the system clock). Actually double the clock rate.

- Here we are running at a low speed and did not constrain our design in terms of time yet.

- So it safer to sample somewhere in the middle of the clock.
Sample on Rising or Falling edge?

Two middles (one-fourth and three-fourth points) of the system clock coincide with the RISING edge of the high-speed clock.
### Different Match Units

**Table 3-1: ILA Trigger Match Unit Types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Bit Values¹</th>
<th>Match Function</th>
<th>Bits Per Slice²</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>0, 1, X</td>
<td>‘=’, ‘&lt;&gt;’</td>
<td>Virtex-5: 19</td>
<td>Can be used for comparing data signals where transition detection is not important. This is the most bitwise economical type of match unit.</td>
</tr>
<tr>
<td>Basic w/edges</td>
<td>0, 1, X, R, F, B</td>
<td>‘=’, ‘&lt;&gt;’</td>
<td>All others: 8</td>
<td></td>
</tr>
<tr>
<td>Extended</td>
<td>0, 1, X</td>
<td>‘=’, ‘&lt;&gt;’, ‘&gt;’, ‘&gt;=’, ‘&lt;’, ‘&lt;=’</td>
<td>Virtex-5: 16</td>
<td>Can be used for comparing address or data signals where magnitude is important.</td>
</tr>
<tr>
<td>Range</td>
<td>0, 1, X</td>
<td>‘=’, ‘&lt;&gt;’, ‘&gt;’, ‘&gt;=’, ‘&lt;’, ‘&lt;=’, ‘in range’, ‘not in range’</td>
<td>Virtex-5: 8</td>
<td>Can be used for comparing address or data signals where a range of values is important.</td>
</tr>
</tbody>
</table>
Multiple Match Units

Figure 1-3: ILA Core Connection Example
Complex Trigger Sequencer

Figure 3-12: Trigger Sequencer Block Diagram with 16 Levels and 16 Match Units
Virtual Input Output

VIO Console Window

IN1 IN2 IN3

OUT1 OUT2

Figure 4-45: The VIO Console Window
ILA (Integrated Logic Analyzer) Core **Inserter** Flow

VIO (Virtual Input and Output) Core **Generator** Flow
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... buses 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