Debugging Techniques Using the Vivado Logic Analyzer

Course Description
As FPGA designs become increasingly more complex, designers continue to look to reduce design and debug time. The powerful, yet easy-to-use Vivado® logic analyzer debug solution helps minimize the amount of time required for verification and debug. This one-day course will not only introduce you to the cores and tools and illustrate how to use the triggers effectively, but also show you effective ways to debug designs—thereby decreasing your overall design development time. This training will provide hands-on labs that demonstrate how the Vivado debug tool can address advanced verification and debugging challenges.

Level: FPGA 2
Training Duration: 1 day

Who Should Attend?
System and logic designers who want to minimize verification and debug time.

Prerequisites:
- Basic language concepts
  - Designing with VHDL or equivalent knowledge of VHDL
  - Designing with Verilog or equivalent knowledge of Verilog
- Basic FPGA skills
  - Essentials of FPGA Design
  - Intermediate FPGA skills
  - Vivado Static Timing Analysis and Xilinx Design Constraints
- Programming and Debug videos recommended

Software Tools:
- Vivado Design or System System Edition 2015.3

Hardware:
- Architecture: N/A*
- Demo board: Kintex®-7 FPGA KC705 board*

Skills Gained: After completing this training, you will be able to:
- Identify each Vivado IDE debug core and explain its purpose
- Effectively utilize the Vivado logic analyzer
- Implement the Vivado IDE debug cores using both the netlist insertion and HDL instantiation tool flows
- Select effective test points in your design
- Optimize design and core performance when debug cores are used
- Execute various techniques for collecting data including
  - File storage
  - Scripting
  - Building custom triggers

Course Outline
1. Introduction to the Vivado Logic Analyzer
2. Demo: JTAG-to-AXI Master Debug IP Transactions
3. Adding the Debug Cores – Netlist Insertion Flow
4. Lab 1: Inserting a Debug Core Using the Netlist Insertion Flow
5. Instantiating the Debug Cores – HDL Instantiation Flow
6. Lab 2: Adding a Debug Core Using the HDL Instantiation Flow
7. Debug Flow in IP Integrator
8. Lab 3: Debugging Flow – IPI Block Design
9. Triggering and Visualizing Data
10. Demo: Using Dashboards in the Vivado Logic Analyzer
11. Demo: Trigger on Startup
12. Tips and Tricks
13. Lab 4: Tips and Tricks
14. Scripting
15. Lab 5: VIO Tcl Scripting
16. Remote Access
17. Lab 6: Remote Access

Lab Description
Lab 1: Inserting a Debug Core Using the Netlist Insertion Flow – Insert ILA cores into an existing synthesized netlist and debug a common problem.
Lab 2: Adding a Debug Core Using the HDL Instantiation Flow – Build upon a provided design to create and instantiate a VIO core and observe its behavior using the Vivado logic analyzer.
Lab 3: Debugging Flow – IPI Block Design – Add an ILA IP core to a provided block design and connect nets to the core. Observe its behavior using the Vivado logic analyzer.
Lab 4: Tips and Tricks – Sample across multiple time domains and use advanced trigger and capture capabilities.
Lab 5: VIO Tcl Scripting – Configure automated analysis.
Lab 6: Remote Access – Use the Vivado logic analyzer to configure an FPGA, set up triggering, and view the sampled data from a remote location.