System Debugging using Logic Debug Cores and XSDK

XILINX.

Objectives

> **After completing this module, you will be able to:**

- List various available debug cores and their functionality
- Describe the process of including the Vivado Design Suite debug tool sampling cores
- State how the Vivado integrated logic analyzer, XSDK Debug perspective, and TCF tools facilitate hardware and software debugging

XILINX.

Outline

> *Introduction*
> Hardware Debugging
> Software Debugging
> System Debugging
> Summary

Introduction

- **Debugging is an integral part of embedded systems development**
- **The debugging process is defined as testing, stabilizing, localizing, and correcting errors**
- **Two methods of debugging:**
  - Hardware debugging via a logic probe, logic analyzer, in-circuit emulator, or background debugger
  - Software debugging via a debugging instrument
    - A software debugging instrument is dedicated hardware and part of the silicon that is accessible via JTAG or dedicated part pins
    - Controls the processor as an intrusive debug unit that is disabled during normal operation
    - Some "hard" processors have this feature permanently available while "soft" processors may have this physically removed from the delivered product
- **Debugging types:**
  - Functional debugging
  - Performance debugging

The Need for On-Chip Debug

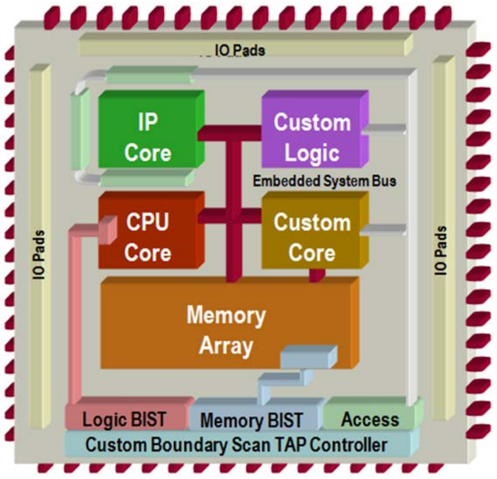
> **FPGAs/SoCs offer limited internal visibility**
  >> How to access the embedded system bus?
  >> How to monitor memory and registers?

> **Non-processor dedicated silicon IP cores**
  >> Cannot obtain internal access

> **Full scan insertion**
  >> Increases overhead

> **Changes late in the design cycle are ENORMOUSLY expensive**

> **Co-verification**
  >> Tools are cumbersome and slow
  >> Modeling issues



XILINX.

Xilinx Solution for Debugging Embedded Designs

> **Hardware debugging tool**
  >> Vivado logic analyzer for hardware; functionally replaces expensive external logic analyzer

> **Software debugging tools**
  >> XSDK Debugging perspective and inexpensive JTAG cable replace ICE

> **Built-in, cross-probing trigger capability between hardware and software**
  >> Vivado logic analyzer invoked through Vivado
  >> XSDK Debugging perspective accessed from within XSDK

> **A single JTAG connection can be used for**
  >> Programming the programmable logic
  >> Downloading application
  >> Hardware and software debugging

XILINX.

Hardware Debugging

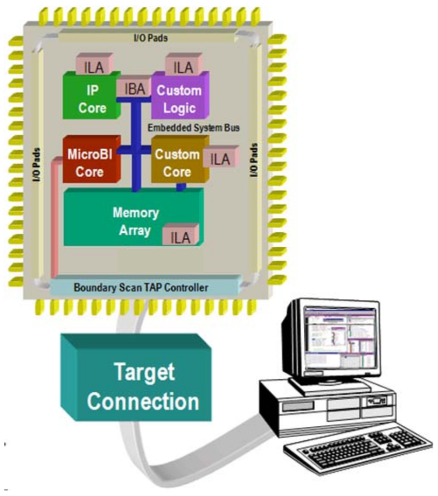
XILINX

Vivado Logic Analyzer System

> **Vivado logic analyzer tool cores provide full internal visibility to all soft IP**
  - Access to hard IP ports
  - Accesses all the internal signals and nodes within the programmable logic (ILA)
  - Stimulus can be applied using the Virtual I/O core (VIO)

> **Debugging occurs at, or near, system speeds**
  - Debug on-chip using the system clock

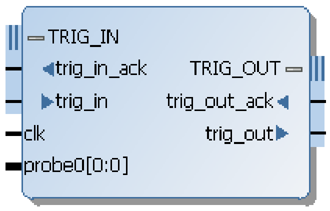
> **Minimize pins needed for debugging**
  - Access via the JTAG interface



XILINX

ILA Core

- Used for monitoring internal programmable logic signals for post-analysis
- Multiple configurable ILA trigger units
  - Configurable trigger input widths and match types for use with different input signals types
- Up to 64 probes through GUI
  - Up to 1024 probes through tcl command
- Sequential triggering
- Storage qualification
- Configurable cross triggering
  - Trigger in and Trigger out interfaces
- Pre- and post-trigger buffering (capture data before, during, and after trigger condition is met)



Core Resources Usage

- Debug tools will use block RAM to store samples
- More block RAM equates to larger (deeper and/or wider) trace buffers
- Use larger devices for debugging prototypes to provide extra block RAM
- If the application is utilizing too much block RAM, temporarily modify hardware applications to free up block RAM for debugging usage—small FIFOs and buffer memory, for example

System ILA Core

> Multiple probe ports, which can be combined into a single trigger condition

> Debugging of any debuggable interface including AXI4-MM and Stream

> User-selectable AXI4-MM channel debug and AXI Data/Address width selection

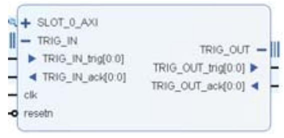
> Data and Trigger probe and interface type selection

> BRAM estimation

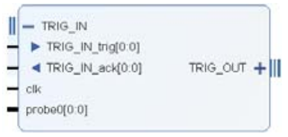
> AXI4-MM and AXI4-Stream Protocol checking

> Allows mixing of native and interface modes

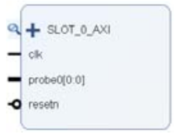
Interface with Cross-triggering



Native with Cross-triggering



Native with no Cross-triggering



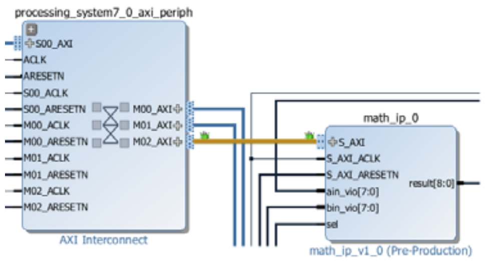
XILINX.

Adding Logic Analyzer Cores

> Instantiate in HDL
  >> Cores added like any other peripheral
    – Found in the IP Catalog under Debug
  >> User makes connections to ILA
    – Debug signals, clocks

> IP Integrator
  >> Add core from IP Catalog
  >> Interactively select signals and mark for debug

> mark_debug
  >> Specify in HDL
  >> Specify through tcl
  >> Specify in xdc
  >> Mark in IP Integrator

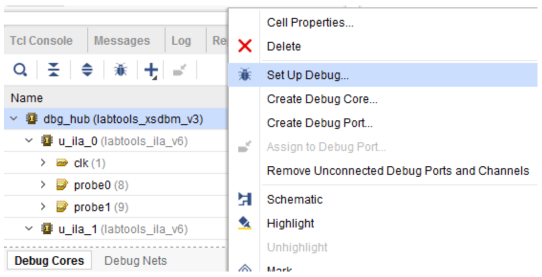


XILINX.

Vivado Debug Tool Access Points

> **Select Tools > Set up Debug to launch the Vivado Debug Wizard**
   - Synthesized design must be open
   - Debug tab is made visible

> **Right-click the debug_hub icon in this window to launch/re-launch the wizard**
   – Add signals
   – View resources
   – Create debug core or port
   – Implement the cores



XILINX.

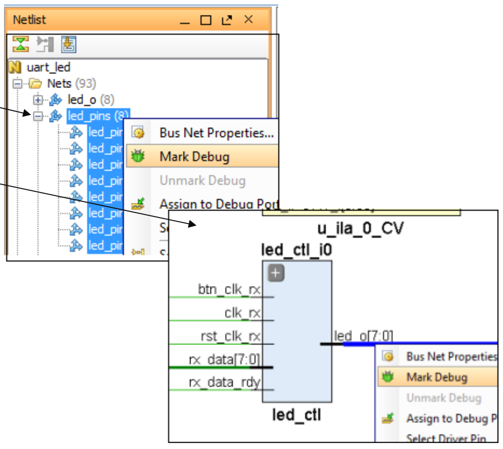
Selecting Signals to Debug

> **Select nets in the Vivado tool by any means**
   - Netlist view (nets folders)
      – Each level of logic hierarchy
   - Schematic
   - Find results

> **Right-click the net and select Mark Debug**

> **Nets added to Unassigned Nets folder in the Debug tab view**
   - Placeholder for probable nets prior to configuring cores

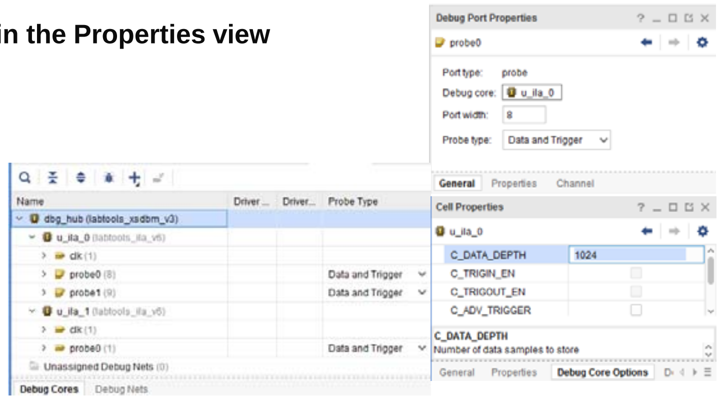
> **Net name search also in the Set Up Debug Wizard**



XILINX.

Debug Tool Configuration

- The Vivado tool view displays core content and configuration
  - CLK, PROBE
  - Signal count
- Set options for cores and signals in the Properties view



Debug Port Properties

probe0

Port type:	probe
Debug core:	u_ila_0
Port width:	8
Probe type:	Data and Trigger

General | Properties | Channel

Name	Driver ...	Driver...	Probe Type
dbg_hub (labtools_xsdbm_v3)			
u_ila_0 (labtools_ila_v6)			
clk (1)			
probe0 (8)			Data and Trigger
probe1 (9)			Data and Trigger
u_ila_1 (labtools_ila_v6)			
clk (1)			
probe0 (1)			Data and Trigger
Unassigned Debug Nets (0)			

Debug Cores | Debug Nets

Cell Properties

u_ila_0

C_DATA_DEPTH	1024
C_TRIGIN_EN	☐
C_TRIGOUT_EN	☐
C_ADV_TRIGGER	☐

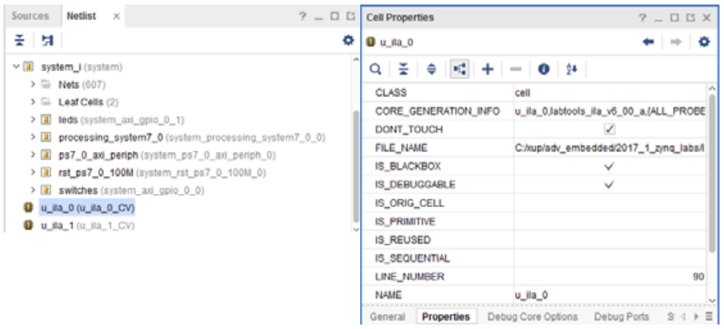
C_DATA_DEPTH
Number of data samples to store

General | Properties | Debug Core Options

XILINX

Exploring Core Logic

- Netlist window populated with implemented core logic
  - debug_core_hub displayed in Netlist view
  - Expandable logic
- Analyze internal core logic
  - Resource statistics
  - Schematic
  - Connectivity
- Floorplan the cores



Sources | Netlist

system_i (system)
Nets (607)
Leaf Cells (2)
leds (system_axi_gpio_0_1)
processing_system7_0 (system_processing_system7_0_0)
ps7_0_axi_periph (system_ps7_0_axi_periph_0)
rst_ps7_0_100M (system_rst_ps7_0_100M_0)
switches (system_axi_gpio_0_0)
u_ila_0 (u_ila_0_CV)
u_ila_1 (u_ila_1_CV)

Cell Properties

u_ila_0

CLASS	cell
CORE_GENERATION_INFO	u_ila_0,labtools_ila_v6_00_a,{ALL_PROBE
DONT_TOUCH	✓
FILE_NAME	C:/xup/adv_embedded/2017_1_zynq_labs/l
IS_BLACKBOX	✓
IS_DEBUGGABLE	✓
IS_ORIG_CELL	
IS_PRIMITIVE	
IS_REUSED	
IS_SEQUENTIAL	
LINE_NUMBER	90
NAME	u_ila_0

General | Properties | Debug Core Options | Debug Ports

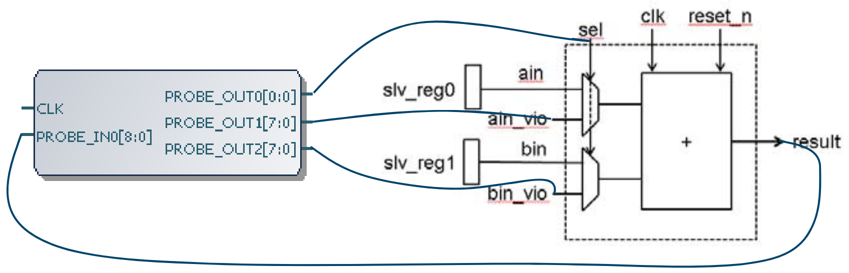
XILINX

VIO Core

> **Support for monitoring and driving internal programmable logic signals in real time**

> **Probe input unit**

> **Probe output unit**

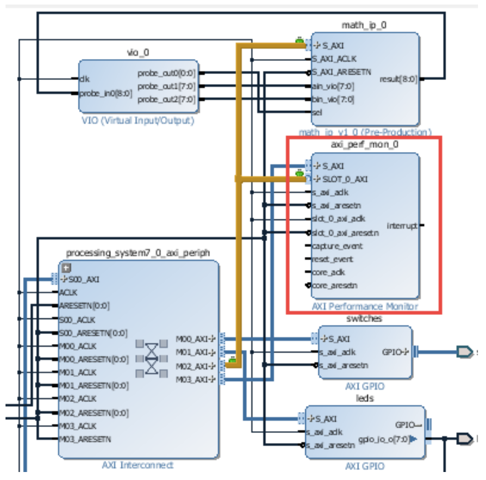


XILINX.

AXI Performance Monitor Core

> **Enables AXI system performance measurement for multiple slots**
>> AXI4 and AXI4-Stream

> **AXI Performance Monitor supports analyzing system behavior on AXI interfaces**
>> Event logging
– Captures AXI events and external events
– Time stamp between two successive events into streaming FIFO
>> Event counting
– Measure events on AXI4/AXI4-Stream monitor slots or external event ports



XILINX.

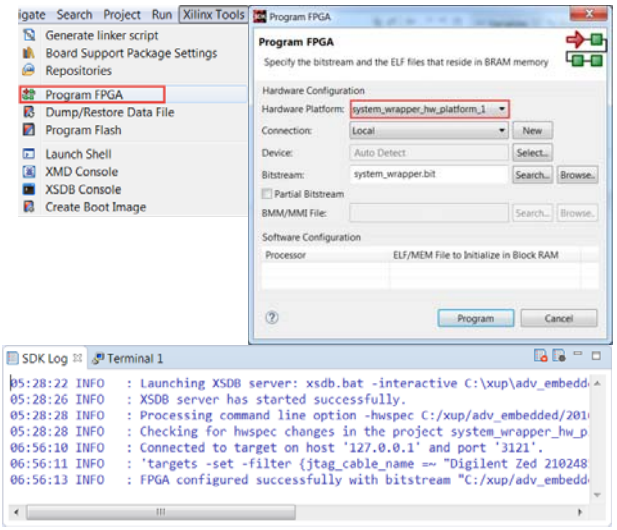
Software Debugging

Software Debugging Support in XSDK

> **XSDK supports software debugging via:**
  - Zynq Programmable SoC contains own internal trace port
    - Performs same general tasks as MDM for MicroBlaze
  - TCF(Target Communication Framework) debugger over digilent cable for ARM
    - Open source
    - Supports system level debugging
    - Improved performance

Configure the Programmable Logic

- **Before debugging session can be launched, the target programmable logic must be configured**
  - PL would have logic analyzer core(s) for hardware debugging
- **Design must contain a hardware debug connection**
  - CoreSight – built into the Zynq ARM Cortex-A9 processor
  - MDM – add in the MicroBlaze based system
- **Verify in the XSDK Log tab that download succeeded**
  - Done pin goes High



Program FPGA

Specify the bitstream and the ELF files that reside in BRAM memory

Hardware Configuration

Hardware Platform: system_wrapper_hw_platform_1

Connection: Local

Device: Auto Detect

Bitstream: system_wrapper.bit

Partial Bitstream

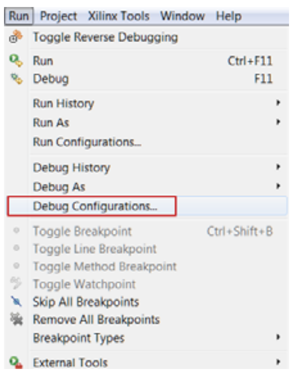
BMM/MMI File:

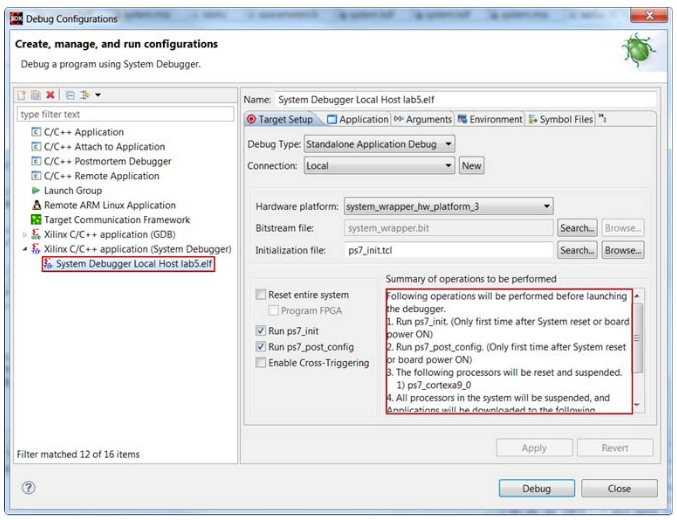
Software Configuration

Processor | ELF/MEM File to Initialize in Block RAM

```
05:28:22 INFO   : Launching XSDB server: xsdb.bat -interactive C:\xup\adv_embedd
05:28:26 INFO   : XSDB server has started successfully.
05:28:28 INFO   : Processing command line option -hwspec C:/xup/adv_embedded/201
05:28:28 INFO   : Checking for hwspec changes in the project system_wrapper_hw_p
06:56:10 INFO   : Connected to target on host '127.0.0.1' and port '3121'.
06:56:11 INFO   : 'targets -set -filter {jtag_cable_name =~ "Digilent Zed 210248
06:56:13 INFO   : FPGA configured successfully with bitstream "C:/xup/adv_embedd
```

Create Debug Configuration




Debug Configurations
Create, manage, and run configurations
Debug a program using System Debugger.
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
Launch Group
Remote ARM Linux Application
Target Communication Framework
Xilinx C/C++ application (GDB)
Xilinx C/C++ application (System Debugger)
System Debugger Local Host lab5.elf
Filter matched 12 of 16 items
Target Setup
Application
Arguments
Environment
Symbol Files
Apply Revert
Debug Close

Name: System Debugger Local Host lab5.elf

Debug Type: Standalone Application Debug

Connection: Local

Hardware platform: system_wrapper_hw_platform_3

Bitstream file: system_wrapper.bit

Initialization file: ps7_init.tcl

☐ Reset entire system

☐ Program FPGA

☑ Run ps7_init

☑ Run ps7_post_config

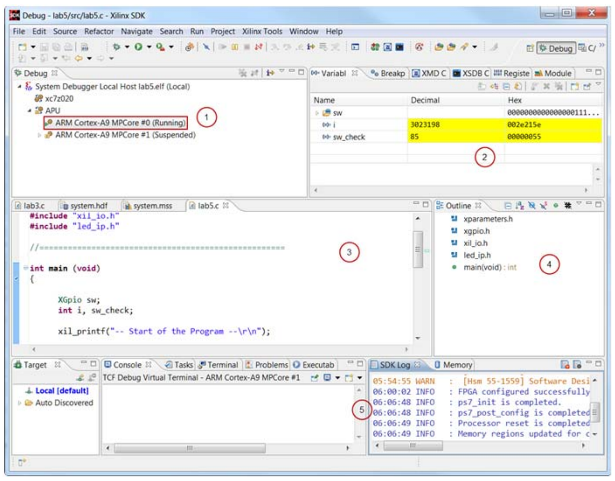
☐ Enable Cross-Triggering

Summary of operations to be performed

Following operations will be performed before launching the debugger.
1. Run ps7_init. (Only first time after System reset or board power ON)
2. Run ps7_post_config. (Only first time after System reset or board power ON)
3. The following processors will be reset and suspended.
   1) ps7_cortexa9_0
4. All processors in the system will be suspended, and Applications will be downloaded to the following

XSDK Debug Perspective

- Stack frame for target threads
- Variables, breakpoints, and registers views
- C/C++ editor
- Code outline
  - Disassembly view can be added using Window > Show View > Disassembly
- Console, XSDK Log, and Memory views


Debug - lab5/src/lab5.c - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
Debug
C/
Debug
System Debugger Local Host lab5.elf (Local)
xc7z020
APU
ARM Cortex-A9 MPCore #0 (Running)
ARM Cortex-A9 MPCore #1 (Suspended)
1
Variabl
Breakp
XMD C
XSDB C
Registe
Module
2
lab3.c
system.hdf
system.mss
lab5.c
3
Outline
xparameters.h
xgpio.h
xil_io.h
led_ip.h
main(void) : int
4
Target
Local [default]
Auto Discovered
Console
Tasks
Terminal
Problems
Executab
TCF Debug Virtual Terminal - ARM Cortex-A9 MPCore #1
5
SDK Log
Memory

Name	Decimal	Hex
sw		0000000000000000111...
i	3023198	002e215e
sw_check	85	00000055

```c
#include "xil_io.h"
#include "led_ip.h"

//====================================================

int main (void)
{

	XGpio sw;
	int i, sw_check;

	xil_printf("-- Start of the Program --\r\n");
```

```
05:54:55 WARN   : [Hsm 55-1559] Software Desi
06:00:02 INFO   : FPGA configured successfully
06:06:48 INFO   : ps7_init is completed.
06:06:48 INFO   : ps7_post_config is completed
06:06:49 INFO   : Processor reset is completed
06:06:49 INFO   : Memory regions updated for c
```

XILINX

System Debugging

XILINX

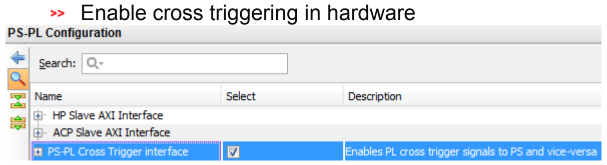
What is System Debugging?

- System debugging is determining where the bug is:
  - In hardware?
  - In software?
- How is system debugging possible?
  - Set breakpoint: when hit → triggers the Vivado logic analyzer
  - Set trigger in Vivado logic analyzer: when hit → halts CPU and debugger stops

XILINX.

System Debug: Simultaneous Hardware/Software Debugging (1)

- **TCF supports simultaneous access over Xilinx download cables**
- **ILA core in the target cores**
- **Software condition triggers hardware:**
  - Software breakpoint asserts the P2F_trig signal and if the ILA is sensitive to this signal, it will capture
- **Hardware condition triggers software:**
  - Match in the ILA asserts the F2P_Trig signal, which halts the processor
- **Vivado logic analyzer running on the host**
  - Enable cross triggering in hardware

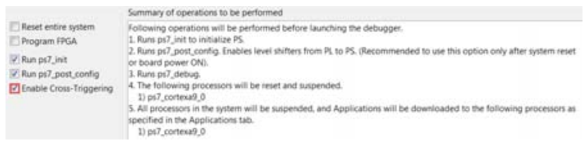


PS-PL Configuration

Search:

Name	Select	Description
HP Slave AXI Interface		
ACP Slave AXI Interface		
PS-PL Cross Trigger interface	☑	Enables PL cross trigger signals to PS and vice-versa

- **Debugger on the host**
  - Enable cross triggering in software



☐ Reset entire system
☐ Program FPGA
☑ Run ps7_init
☑ Run ps7_post_config
☑ Enable Cross-Triggering

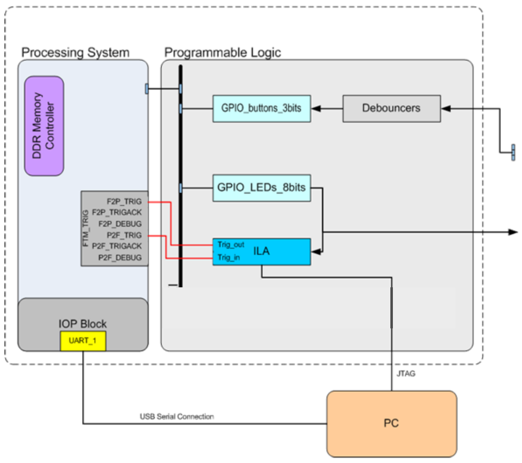
Summary of operations to be performed

Following operations will be performed before launching the debugger.
1. Runs ps7_init to initialize PS.
2. Runs ps7_post_config. Enables level shifters from PL to PS. (Recommended to use this option only after system reset or board power ON).
3. Runs ps7_debug.
4. The following processors will be reset and suspended.
   1) ps7_cortexa9_0
5. All processors in the system will be suspended, and Applications will be downloaded to the following processors as specified in the Applications tab.
   1) ps7_cortexa9_0

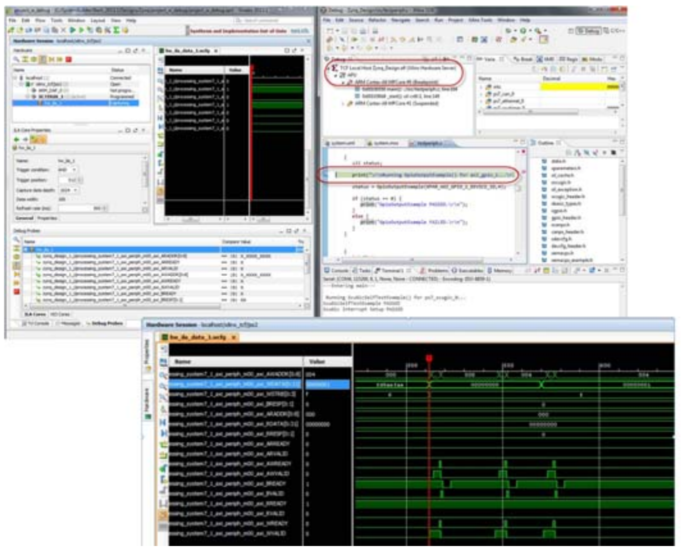
XILINX.

System Debug: Simultaneous Hardware/Software Debugging (2)

- Cross triggering connections between Zynq processor and ILA
- TCF support simultaneous access over Xilinx download cables
  - ILA instantiations
  - Treated like the peripheral cores
- Set breakpoint in the XSDK Debugging perspective: when hit, triggers the logic analyzer
- Set trigger in logic analyzer: when hit, halts CPU and debugger stops



System Debug: Simultaneous Hardware/Software Debugging (3)



Summary

XILINX

Summary

> Debugging is an integral part of embedded systems development

> Vivado and XSDK provides tools to facilitate hardware and software debugging
>> Hardware debugging is done through using Vivado logic analyzer cores
>> Software debugging is performed using system debugger

> XSDK provides environment, perspective, and underlying tools to enable seamless software debugging

> A significant amount of hardware overhead may occur depending on the type and number of cores and sample depth

> The Debug Configuration wizard simplifies hardware connections and the cores inclusion

> With software debugging and the Vivado logic analyzer supporting cross-probing enables simultaneous hardware and software debugging
>> Use it to find and fix embedded system bugs faster

XILINX

Adaptable.
Intelligent.

XILINX