Enabling success from the center of technology™

Xilinx Embedded Processor Debugging

Goals

- **Demonstrate solid debug strategies using the Xilinx Software Development Kit and ChipScope Pro tools**
- **Learn how to achieve a functional hardware and software design in the shortest amount of time**
- **Demonstrate unique Xilinx debug resources and strategies**

Agenda

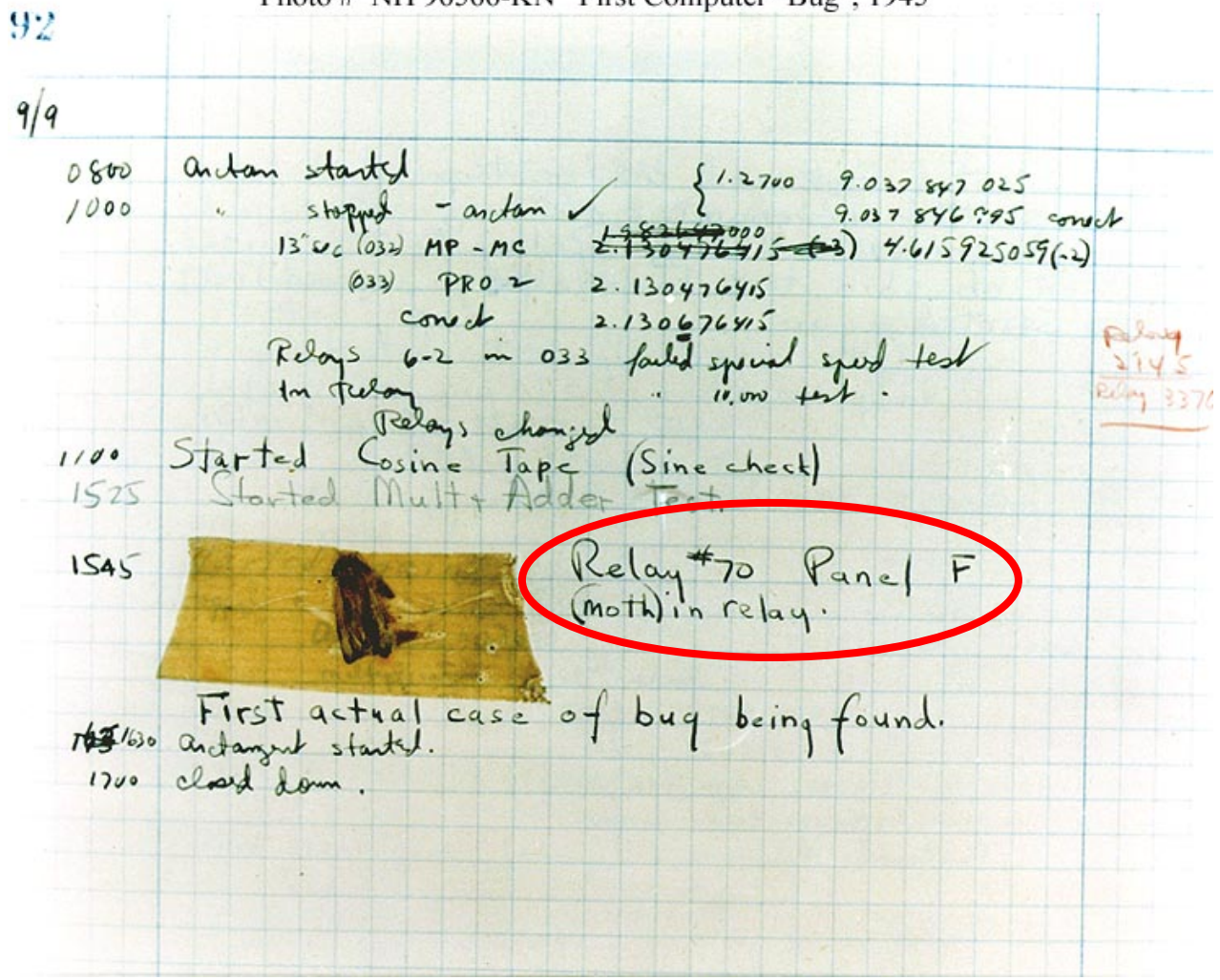
- **Define debugging**
- **Describe tool flow**
- **Tour of Software Development Kit**
- **SDK demo**
- **Platform debug - cross probing with ChipScope Pro**
- **ChipScope Pro demo**
- **Advanced debugging**
- **Lauterbach demo**

Agenda

- **Define debugging**
- Describe tool flow
- Tour of Software Development Kit
- SDK demo
- Platform debug - cross probing with ChipScope Pro
- ChipScope Pro demo
- Advanced debugging
- Lauterbach demo

Origin of Debug

Photo # NH 96566-KN First Computer "Bug", 1945



Admiral Hopper's Bug
September 9, 1945

Define Debugging

- **Debugging is an integral part of embedded systems development**
- **The debugging process is defined as testing, stabilizing, localizing, and correcting errors**
- **Historically software was the debug point**
  - Hardware was viewed as a “static” platform
- **New realm – hardware is just as malleable as software**
  - FPGAs can be changed during development AND in the field just like software

Define Debugging

- **Two methods of debugging**
  - Hardware debugging via a logic probe, logic analyzer, in-circuit emulator, or background debugger
  - Software debugging via a debugging instrument or source level debugger
    - A software debugging instrument is source code that is added to the program for the purpose of debugging

- **Debugging types**
  - Functional debugging
  - Performance debugging

Xilinx Solution

- **Xilinx offers integration of both the hardware and software debug methods**
- **All necessary tools are included in the Embedded Development Kit (EDK)**
- **EDK is a bundle of**
  - Tools
  - IP
  - Licenses
  - Documentation

- **Used to design and develop PowerPC® or MicroBlaze® processors into Xilinx FPGAs**

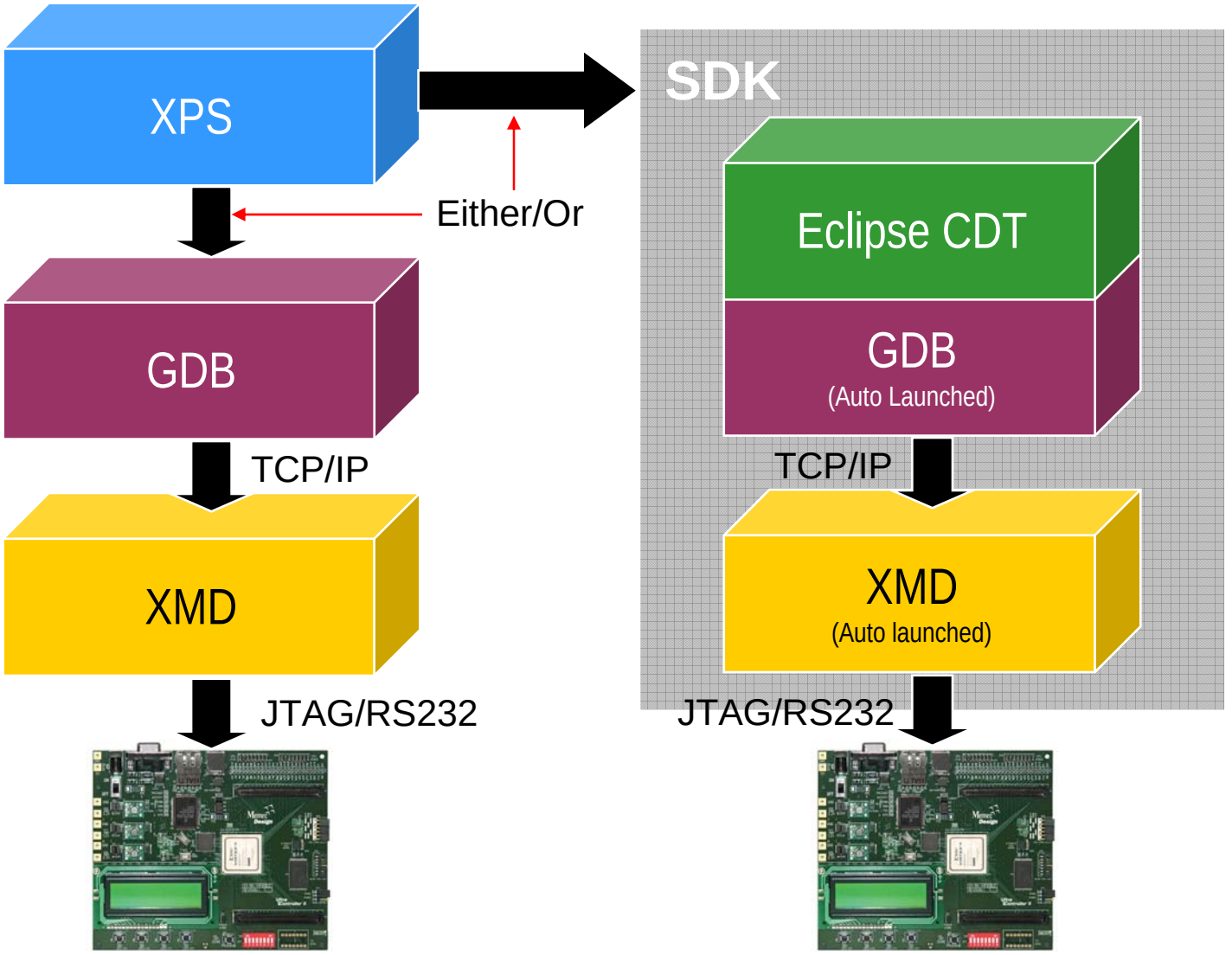
Agenda

- Define debugging
- **Describe tool flow**
- Tour of Software Development Kit
- SDK demo
- Platform debug - cross probing with ChipScope Pro
- ChipScope Pro demo
- Advanced debugging
- Lauterbach demo

Necessary Tools for Debug

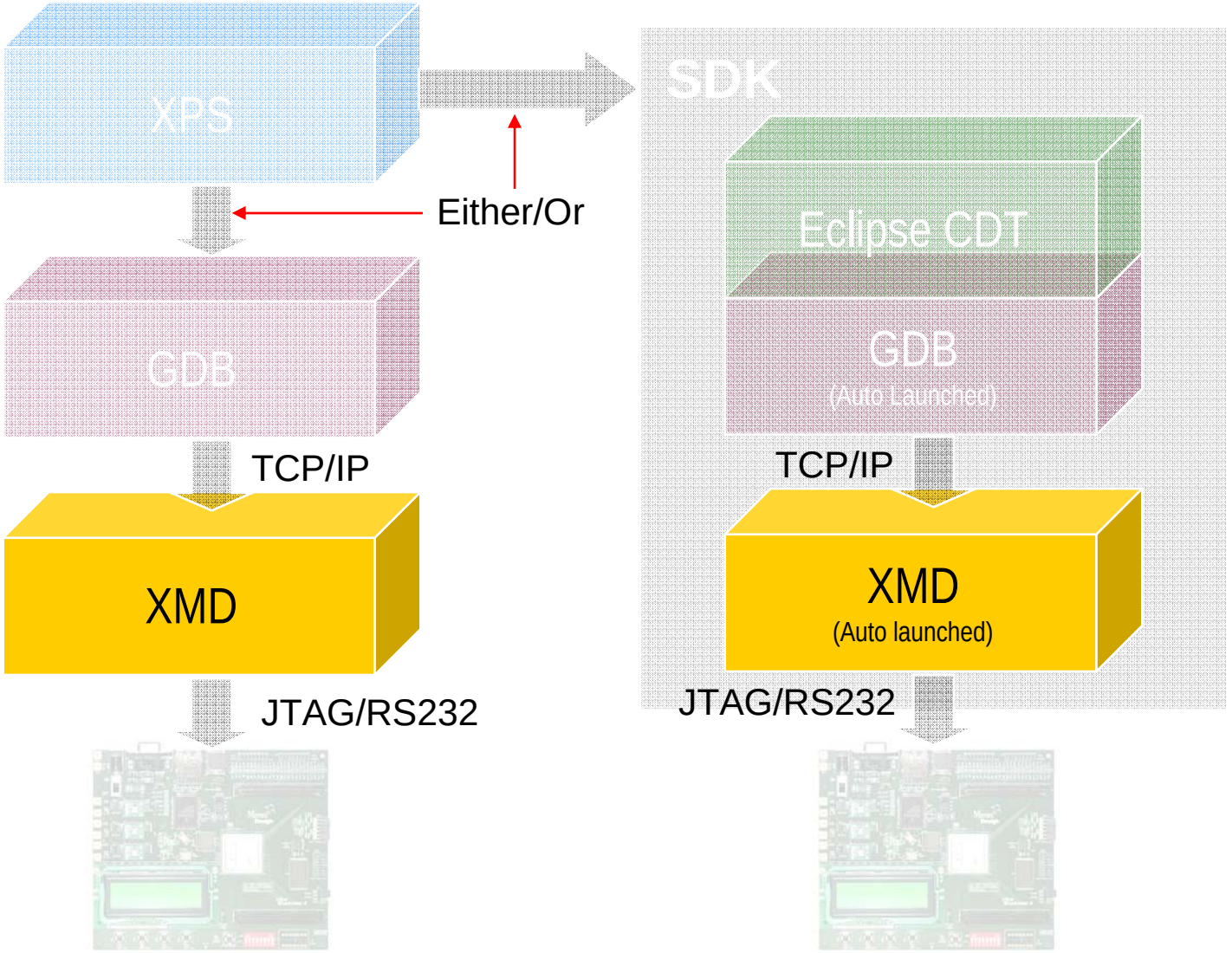
- **Xilinx Microprocessor Debug Engine (XMD)**
- **GNU Debugger (GDB)**
- **Xilinx Platform Studio (XPS)**
- **Platform Studio Software Development Kit (SDK)**
  - Includes Eclipse Code Development Tool (CDT)
- **All above are included in EDK**

Debug Flow



Debug Flow - XMD

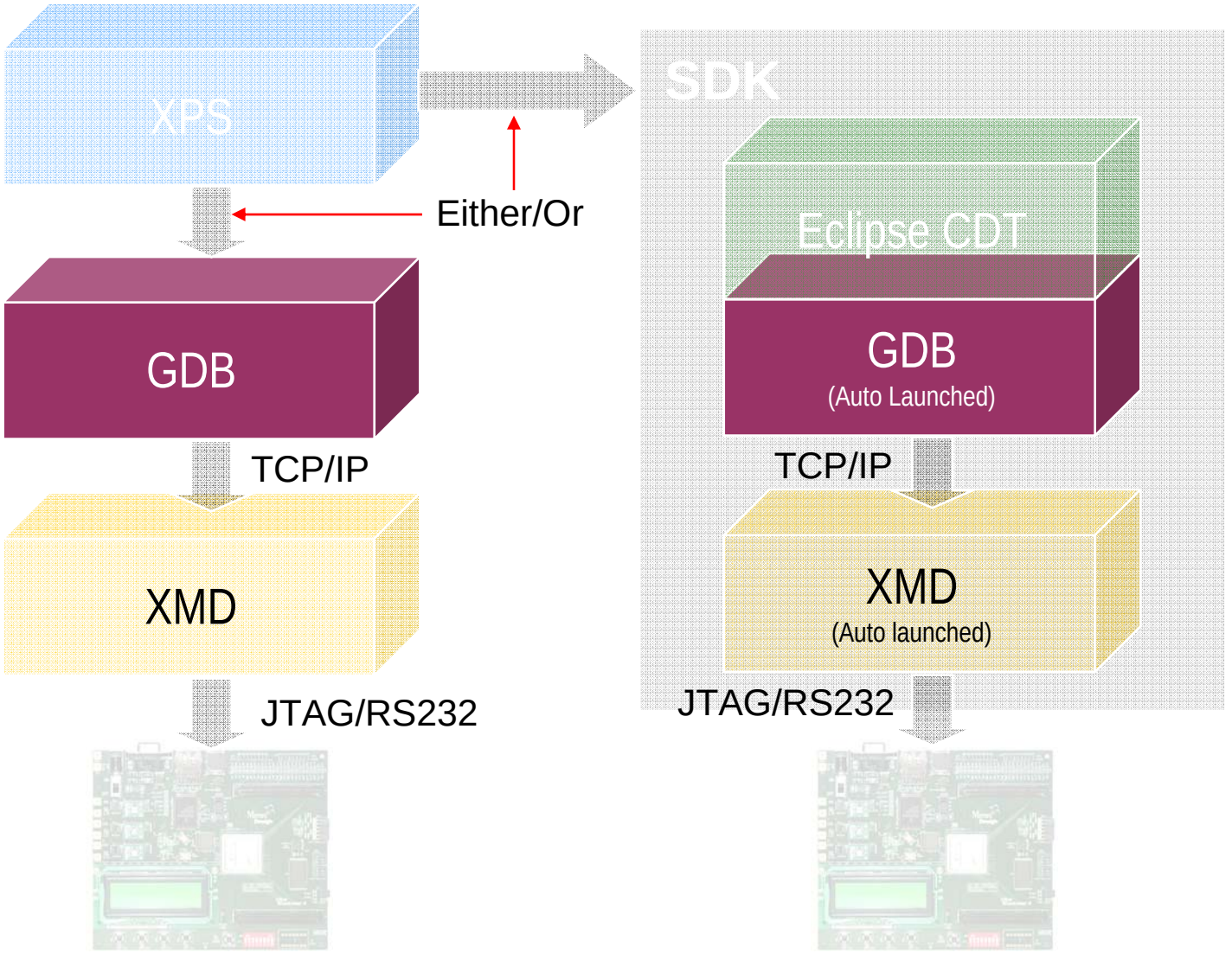
Enabling success from the center of technology™



XMD Functionality

- **Host based application that facilitates the GNU debugger interface**
- **Local interface to control target hardware over JTAG**
- **Debuggers communicate with XMD by using TCP/IP**
- **Tool Command Language (tcl) interface**
- **XMD supports debugging user programs on different targets**
  - Cycle-accurate MicroBlaze instruction set simulator
  - MicroBlaze systems running xmdstub on a hardware board
  - MicroBlaze systems using the MDM peripheral
  - PowerPC systems on a hardware board

Debug Flow - GDB



GDB Functionality

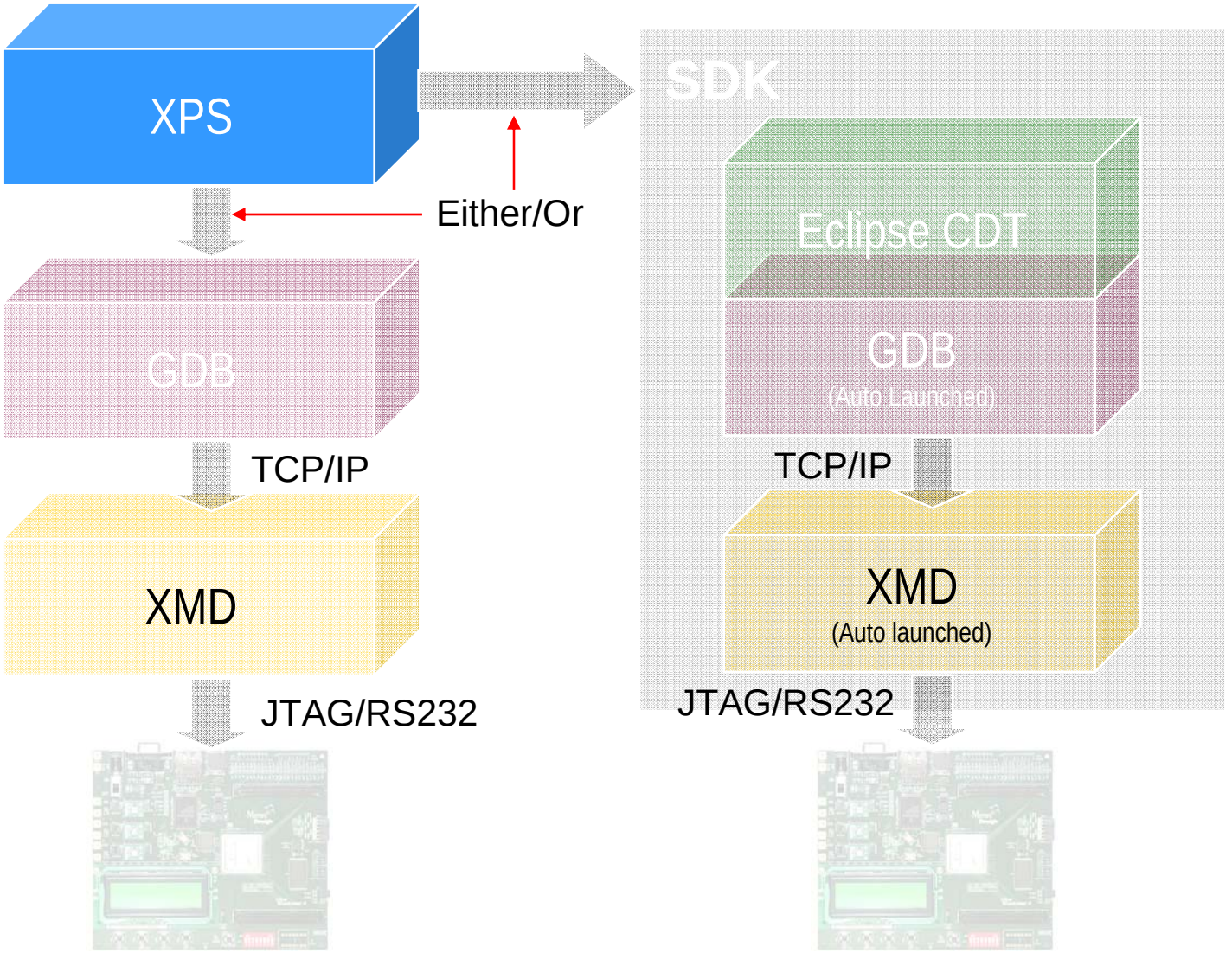
- **GDB is a source and assembly-level debugger that helps you**
  - Start your program
  - Set breakpoints
  - Examine what has happened when your program encounters breakpoints
    - Registers
    - Memory
    - Stack
    - Variables
    - Expressions
  - Change elements of your program so that you can experiment with correcting the effects of one bug and go on to another

GDB Functionality

- **Must generate debug symbols when the program is compiled**
  - Information is part of the object file
  - Describes the data type of each variable or function
  - Correlates between source line numbers and addresses in the executable code

- **Use GDB to debug programs written in C and C++**

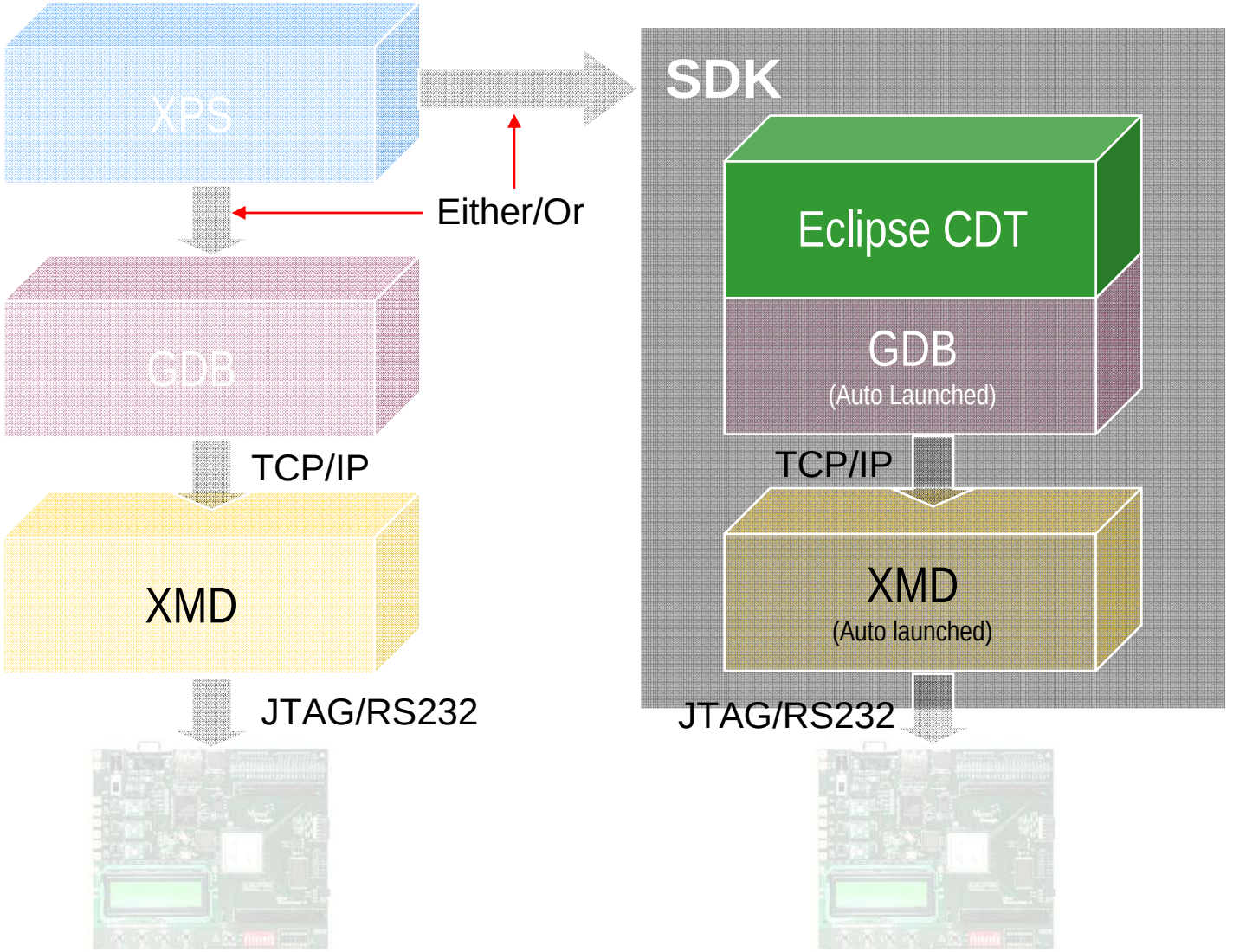
Debug Flow – XPS



XPS

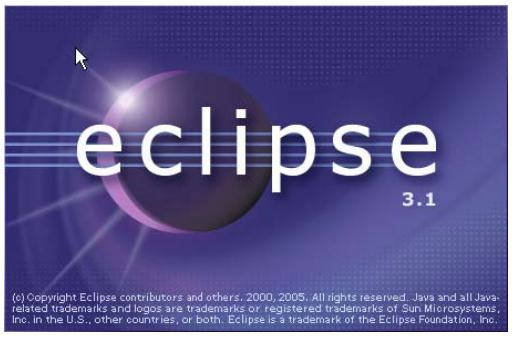
- **The “hardware” tool**
- **Integrated development environment and tool suite used to define, configure, and generate a processor design**
- **Generates all necessary files for hardware implementation and software development**

Debug Flow - SDK

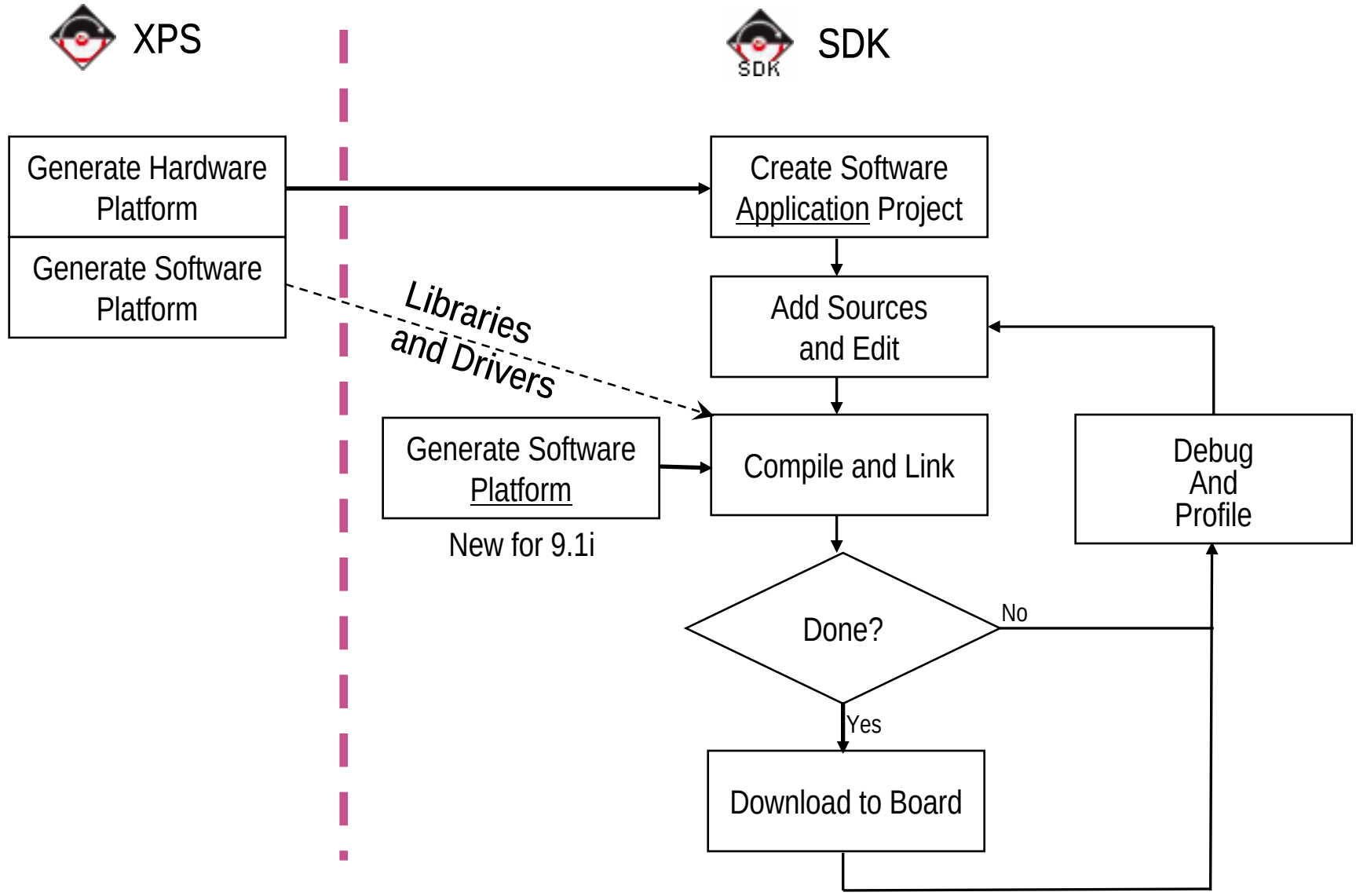


Xilinx Platform Studio SDK

- **The "software" tool**
- **Can be launched from XPS or independently**
  - Software application hand-off from XPS to SDK
  - Software platform generation
  - Linker script generation
  - Software interface document generation
  - Download FPGA bitstream
  - Flash programmer
  - Improved Ease of Use
    - Project setup wizard
- **Enhanced C/C++ editor support includes**
  - Code folding of functions
  - Methods
  - Classes, structures, and macros
- **Eclipse Based Platform version v3.1**



XPS to SDK Software Development Flow



Xilinx Extensions to Eclipse

- **SDK leverage on Eclipse plus Code Development Tool (CDT)**
  - Project management
  - Makefile builder
  - Code editor, error navigation
  - Debug
  - Search

- **SDK value add to CDT**
  - Debug integration using XMD
  - Xilinx custom compiler settings for the PowerPC™ and MicroBlaze™ processors
  - Profiling flow and visualization
  - Productization

Workspaces and Perspectives

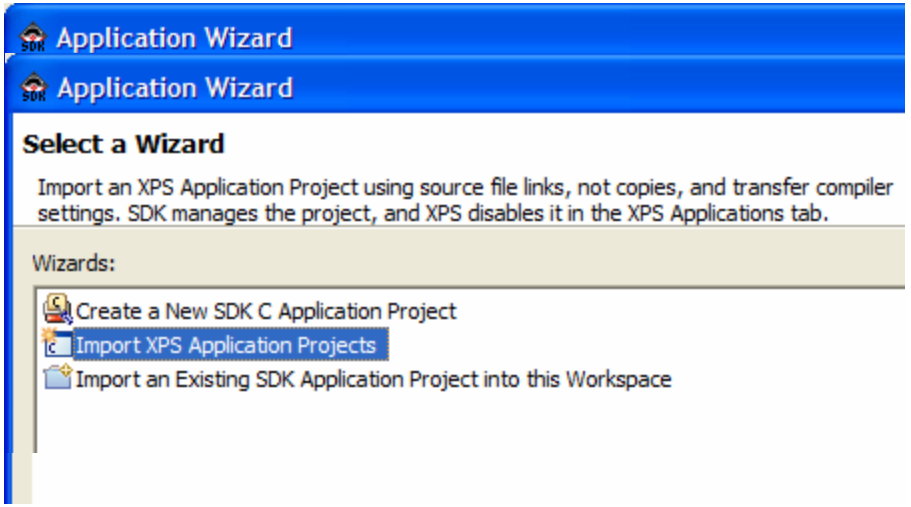
- **Workspace**
  - Location to store preferences and internal project information
  - Transparent to SDK users
  - Source files are not stored under the workspace
- **Views and editors**
  - Basic user interface element
- **Perspectives**
  - Collection of functionally related views
  - Layout of views in a perspective can be customized

Agenda

- Define debugging
- Describe tool flow
- **Tour of Software Development Kit**
- SDK demo
- Platform debug - cross probing with ChipScope Pro
- ChipScope Pro demo
- Advanced debugging
- Lauterbach demo

Xilinx Platform Studio SDK

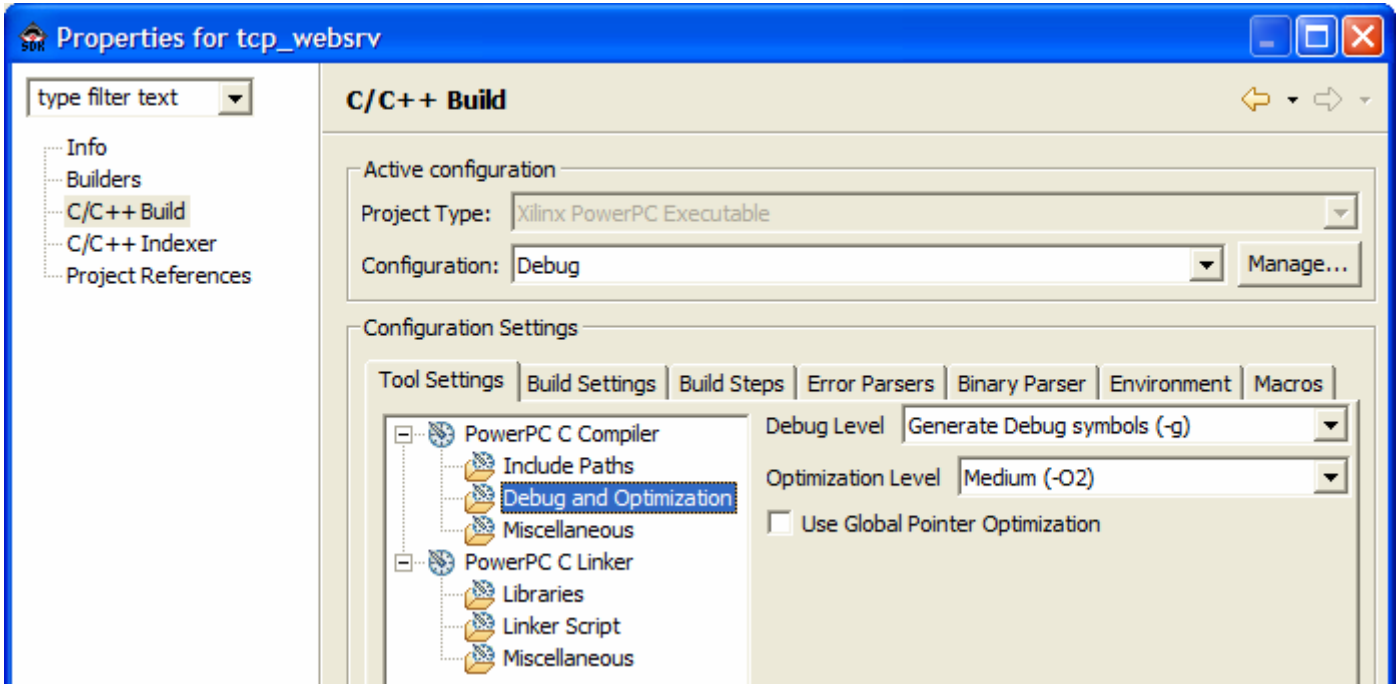
- **Application Wizard assists in the migration from XPS to SDK**
- **Lists the SW applications it knows about**
- **Imports the software application and platform from XPS into SDK**



Setup wizard enables simple migration to full featured software application development and debug environment

Xilinx Platform Studio SDK

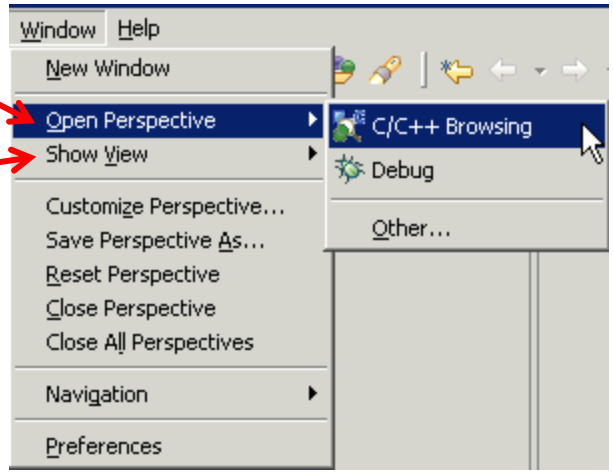
- **Compiler & Linker options**
  - Include paths, debug settings
  - Link libraries, linker script



Opening Perspectives and Views

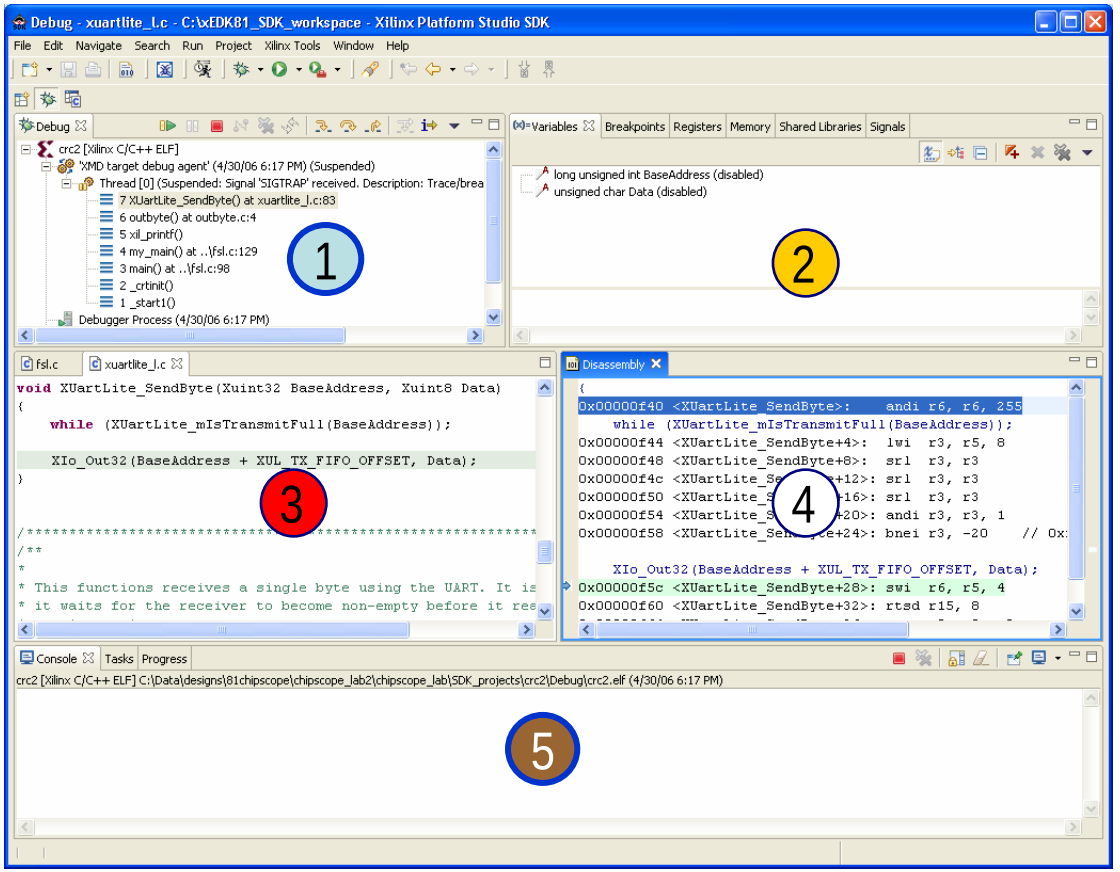
- **To open a perspective, select**
  Window → Open Perspective

- **To open a view, select**
  Window → Show View
  - If the view is already present in the current perspective, the view is highlighted

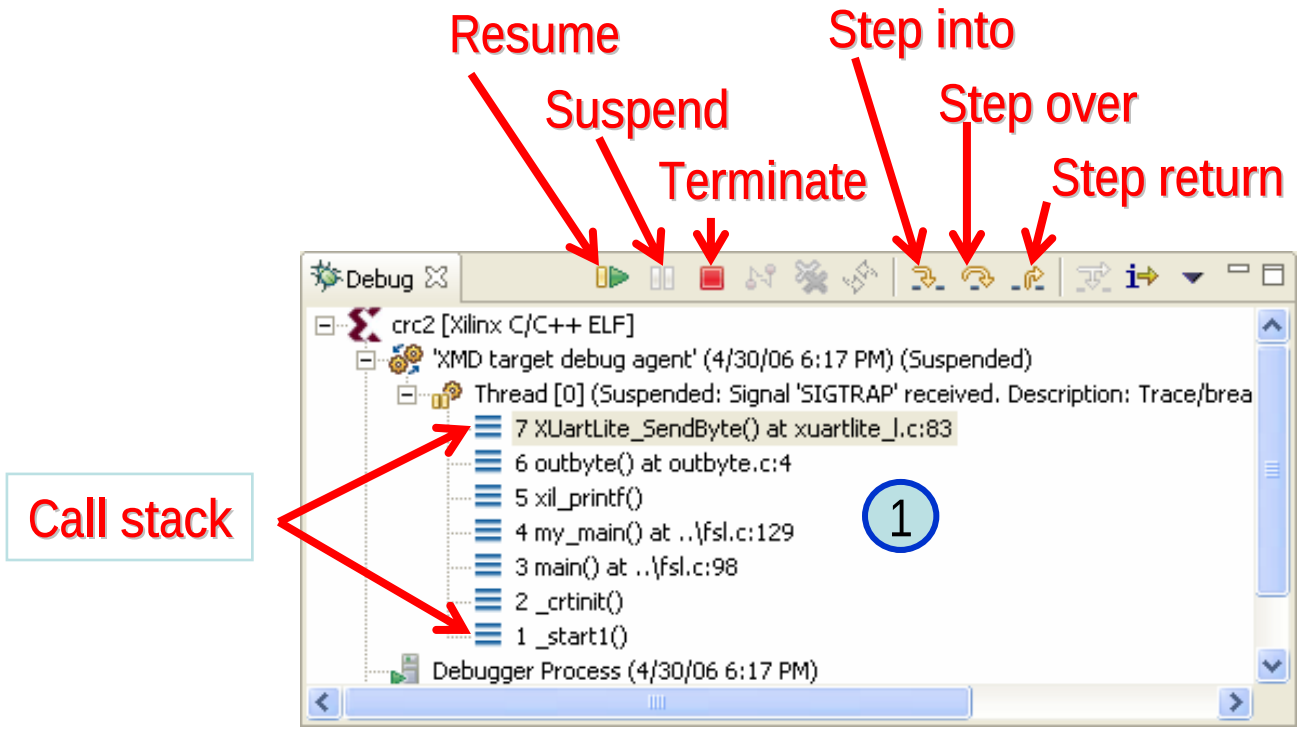


Debug Perspective

1. **Stack & debug control**
2. **Views**
3. **High level source**
4. **Assembly**
5. **Console**

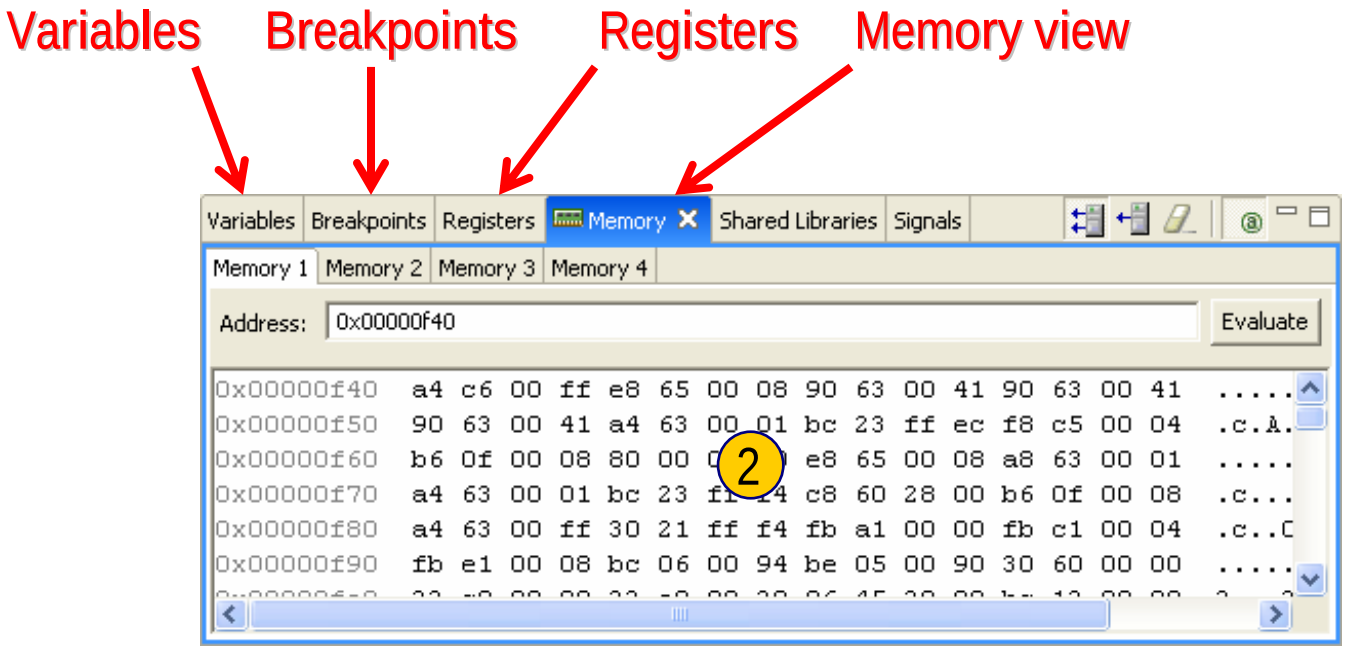


Debugging Control



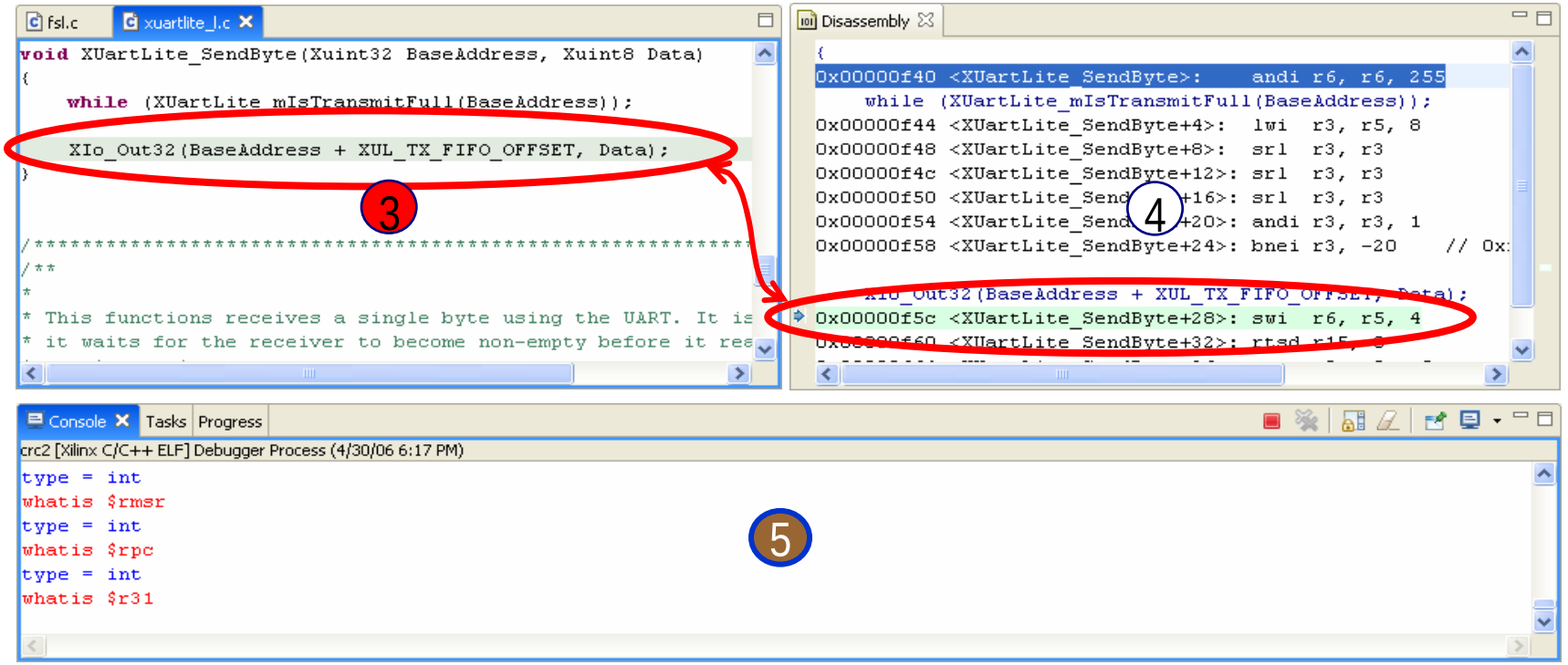
- **Control application execution with buttons**
- **Each thread in your program is represented as a node in the tree**

Views



- **Variables, Breakpoints, Registers, and Memory views allow for viewing and real-time interaction with the contents for more powerful debugging potential**

Sources, Disassembly, and Console



- C/C++ editor highlights the location of the execution pointer, along with allowing the setting of breakpoints
- Code outline and disassembly view provide compiler-level insight to what is occurring in the running source
- Console view lists output information

Performance Question

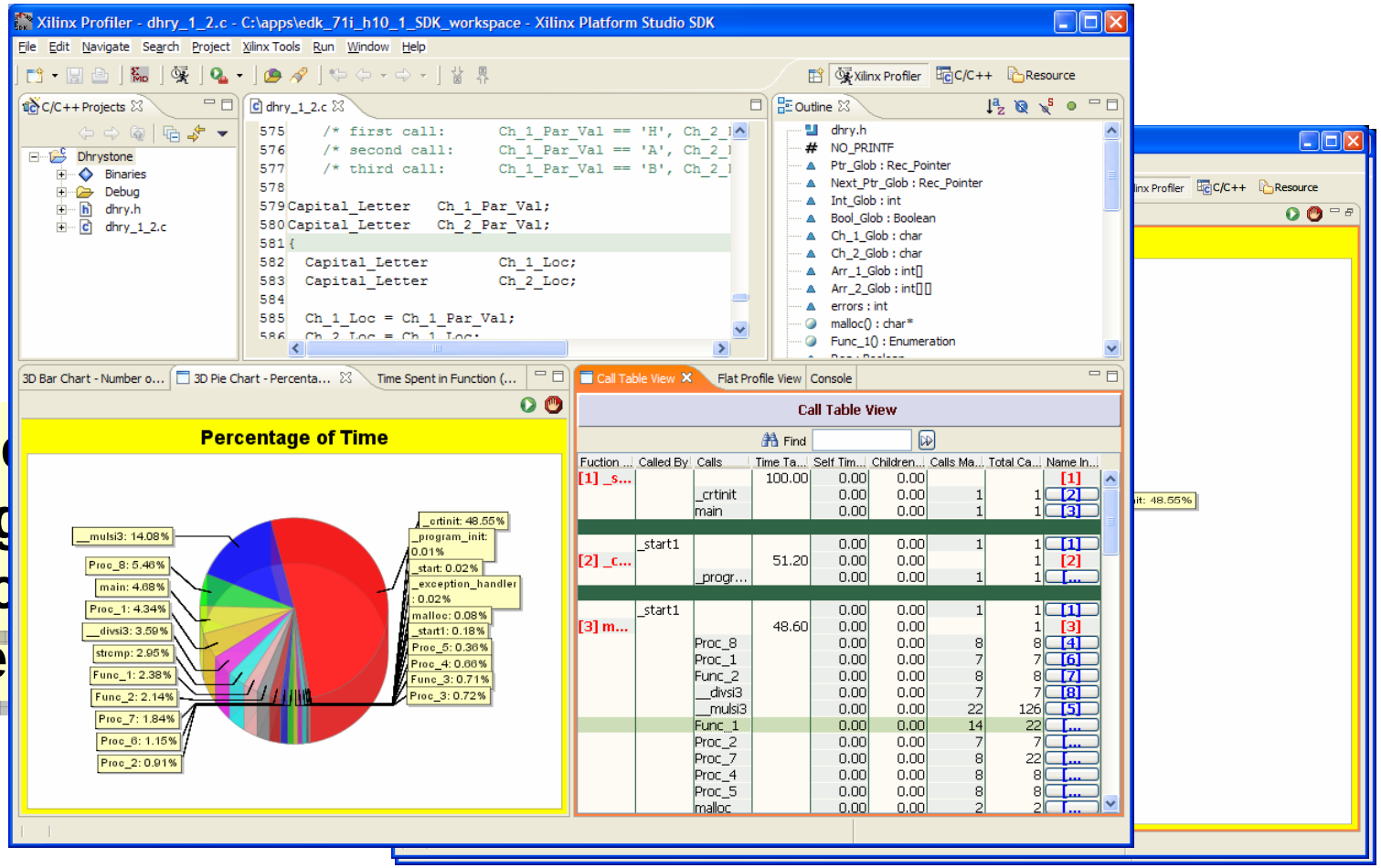
Q: Given an application with 20,000 lines of source made of many functions, where do you start optimizing?

A: Start with the function that uses the most processing time.

- How do you determine this??
- Use Profiling!

SDK Profiling

Enabling success from the center of technology™



Determin
percentag
each func
was calle

All fully integrated into the Platform Studio SDK environment

Agenda

- Define debugging
- Describe tool flow
- Tour of Software Development Kit
- **SDK demo**
- Platform debug - cross probing with ChipScope Pro
- ChipScope Pro demo
- Advanced debugging
- Lauterbach demo

SDK Demo

- **XPS project to SDK project migration**
  - Show application in XPS then in SDK
- **SDK Debug**
  - Debug Perspective
    - Set Breakpoints
    - View/Modify Variables, Memory, etc.
- **Profiling**
  - Profile Perspective
    - Pie Chart
    - Bar Graph
    - Tables

Agenda

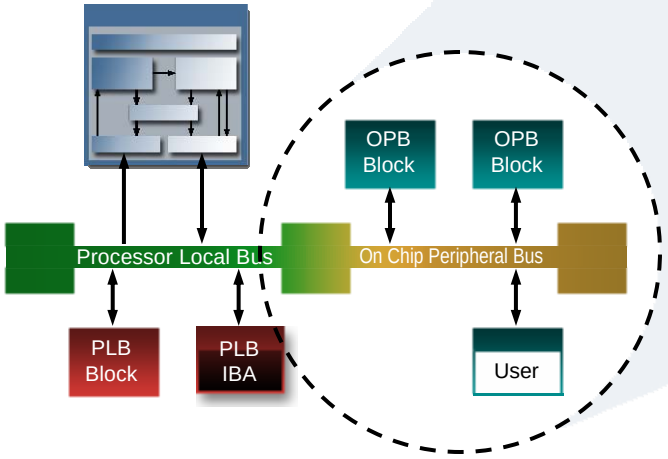
- Define debugging
- Describe tool flow
- Tour of Software Development Kit
- SDK demo
- **Platform debug - cross probing with ChipScope Pro**
- ChipScope Pro demo
- Advanced debugging
- Lauterbach demo

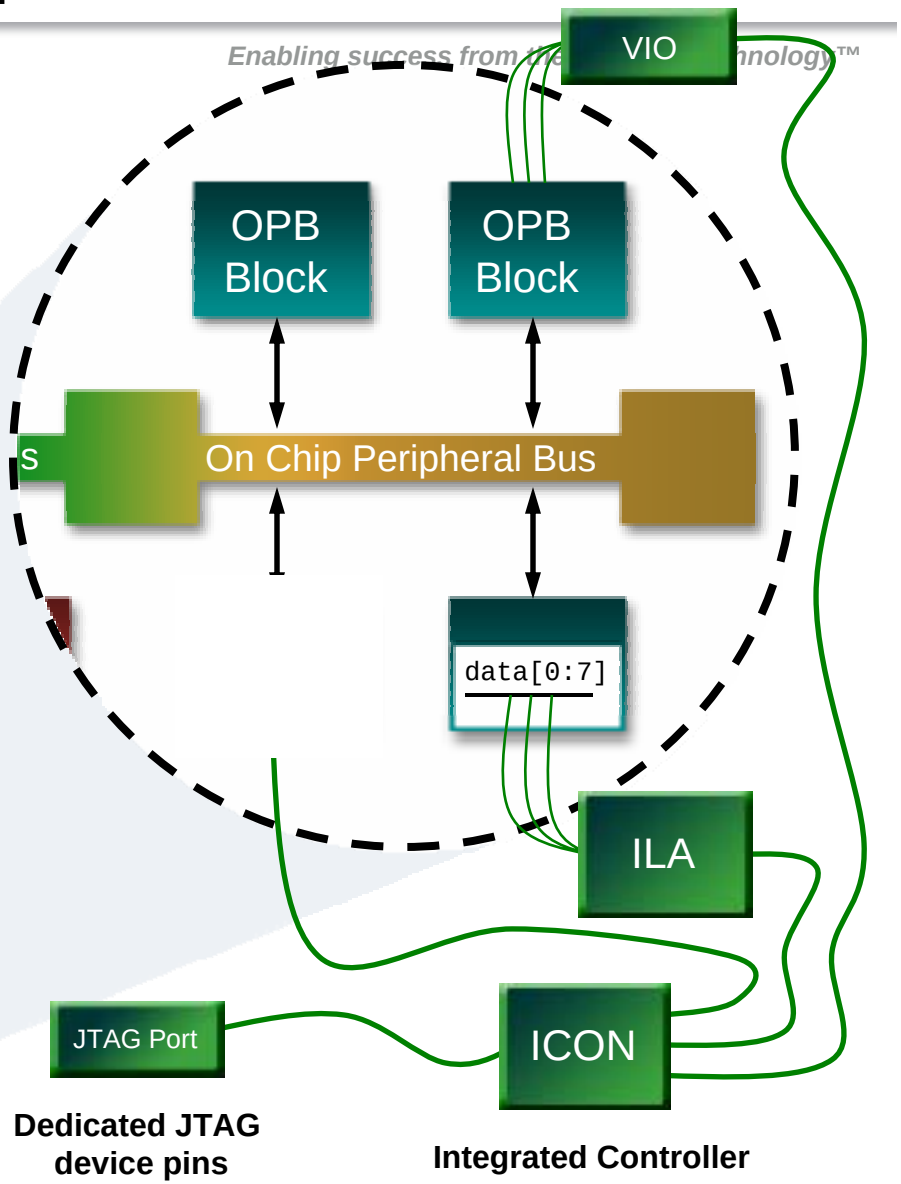
Platform Debug

- **The ability to debug and analyze both the hardware and software platforms simultaneously**
- **Software debug via integrated GNU debugger**
  - Differentiate critical versus typical accesses using software breakpoints
- **Hardware debug using ChipScope Pro**
  - Capture unexpected system issues and exceptions using hardware triggers
- **Synchronous cross triggering between the hardware and software**

ChipScope Pro

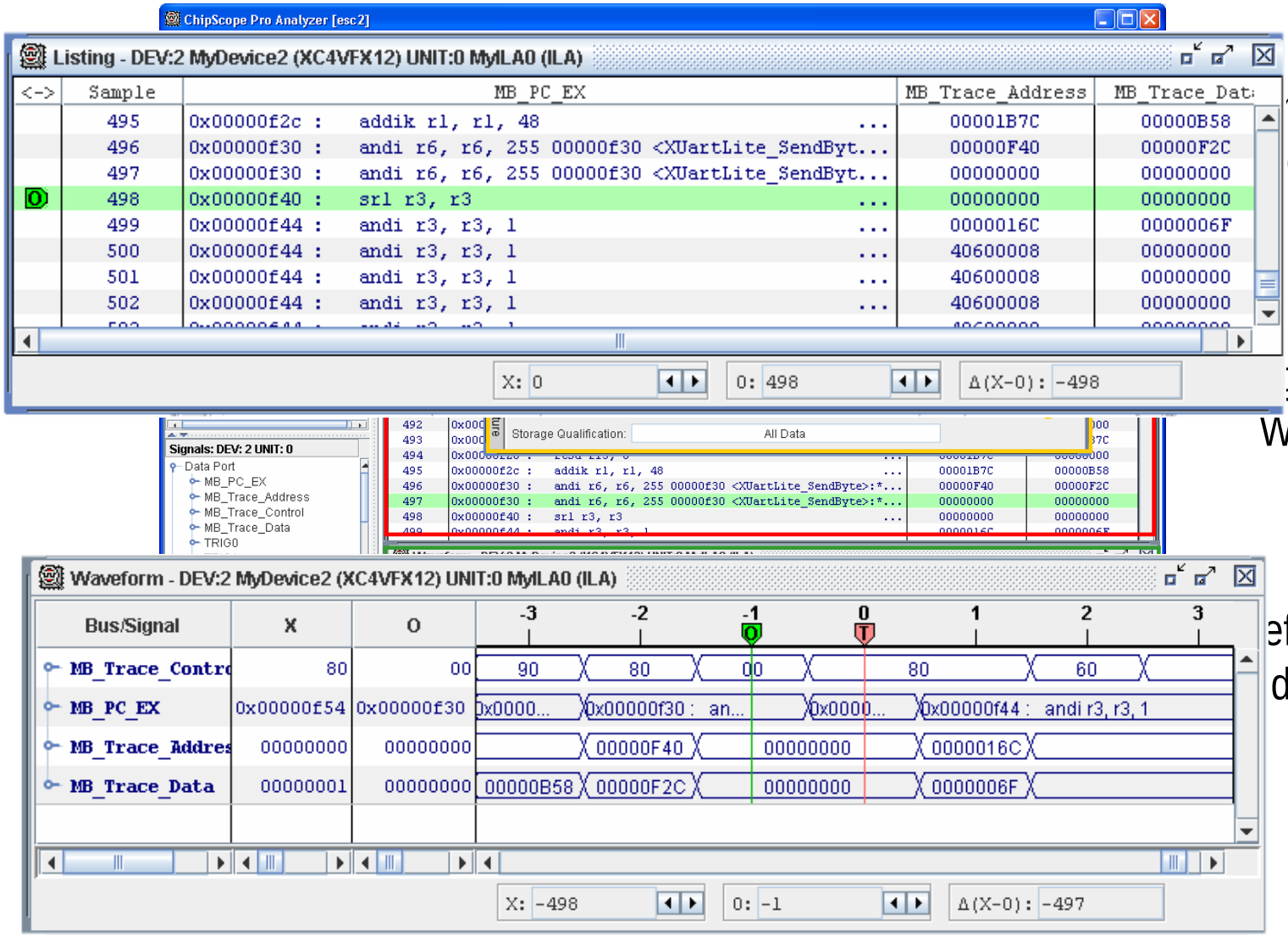
- **Integrated Bus Analyzer (IBA) provides bus transaction analysis (PLB and/or OPB)**
- **Integrated Logic Analyzer (ILA) provides internal logic analysis**
- **Virtual I/O (VIO) provides input stimulus/output analysis for resets and other signals**





ChipScope Pro Analyzer

Enabling success from the center of technology™

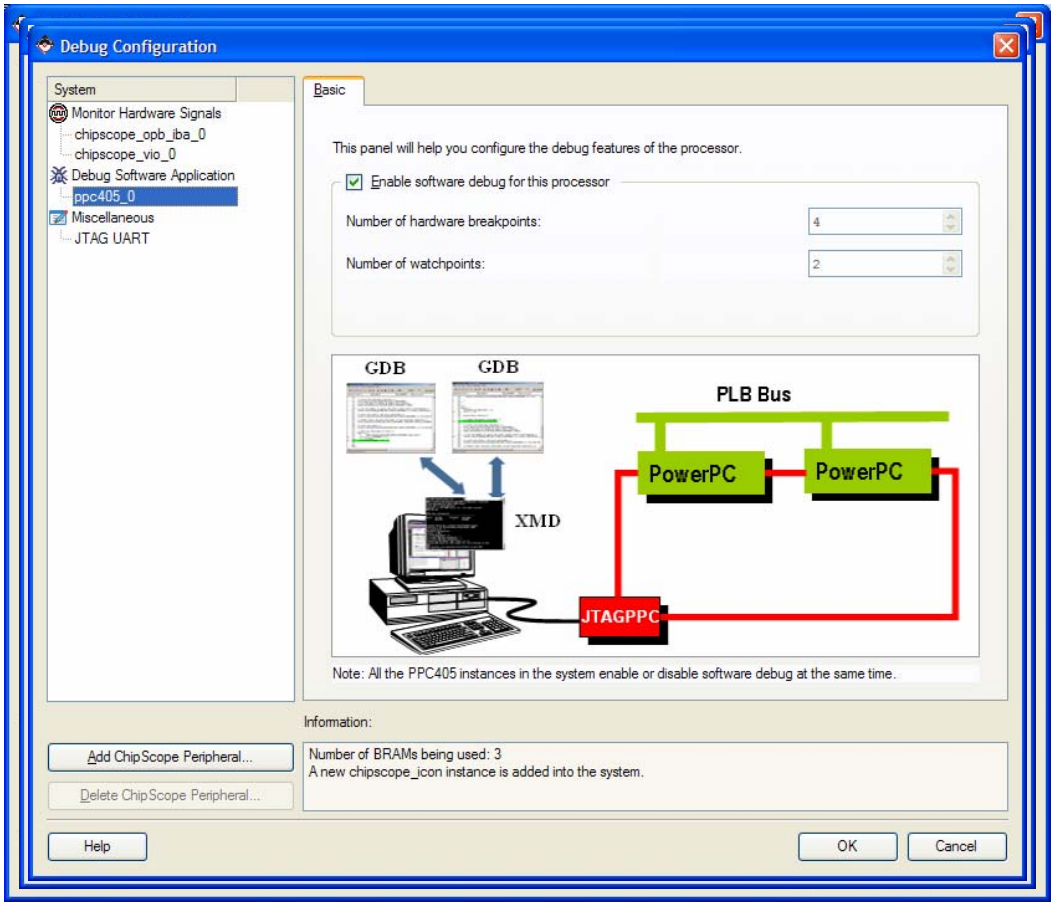


ng
ow

ger Setup
Window

eform
dow

Debug Configuration Wizard



- **Integrated into XPS**
  - Simplifies hardware debug setup
  - Select between various ChipScope cores to insert
  - Setup and configure the selected ChipScope core
  - Configure processor debug features

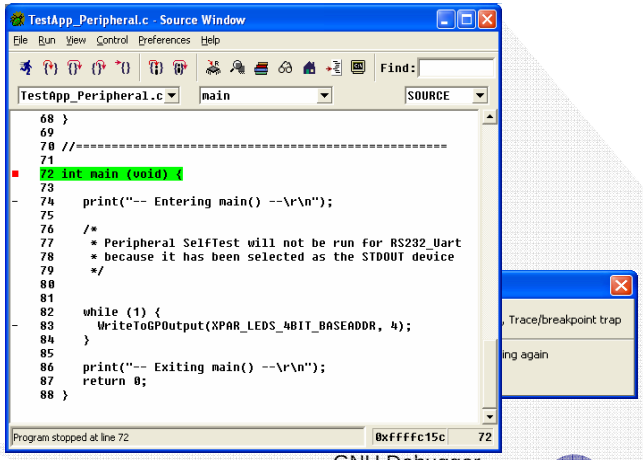
Simultaneous Hardware/Software Debug

- **ChipScope Pro cores in target hardware**
- **ChipScope Pro Analyzer on host**
- **GDB debugger on host**
- **XMD supports simultaneous access over Xilinx cables**

Minimal "*skid-by*" as cross-triggering is done on chip between IBA cores and PPC/MicroBlaze debug interfaces

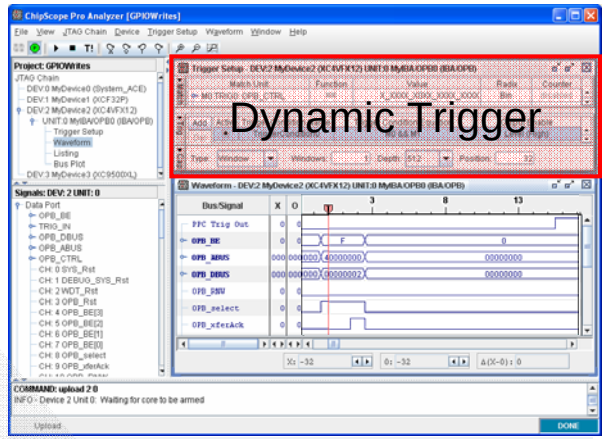
Cross Triggering

Enabling success from the center of technology™



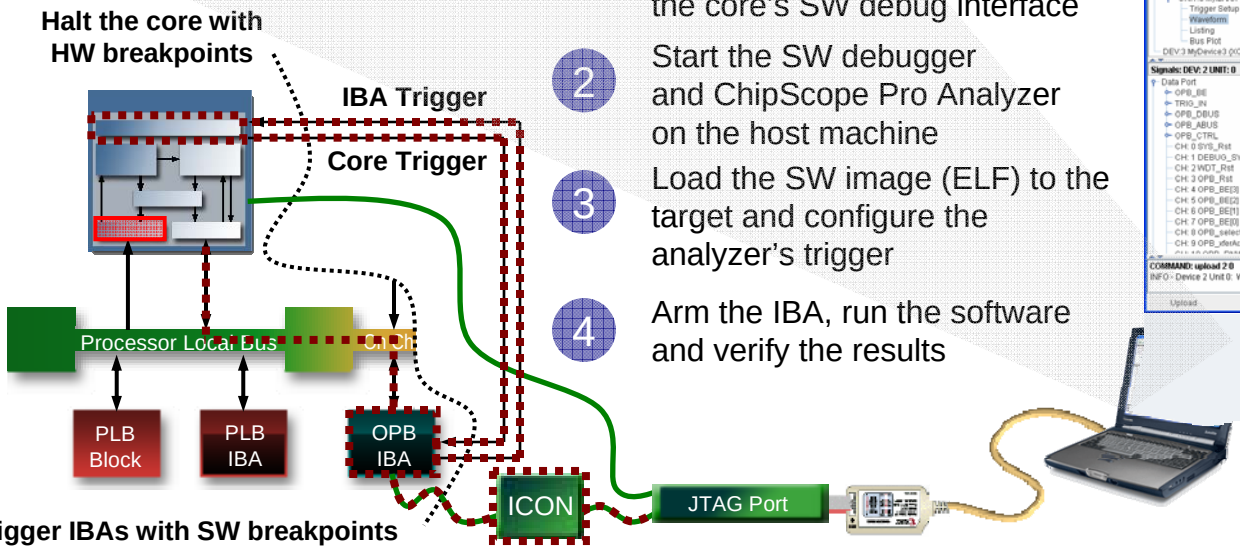
GNU Debugger

- **Set breakpoint in GDB**
  - Triggers ChipScope Pro Analyzer
- **Set trigger in ChipScope Pro**
  - Halts CPU and debugger stops



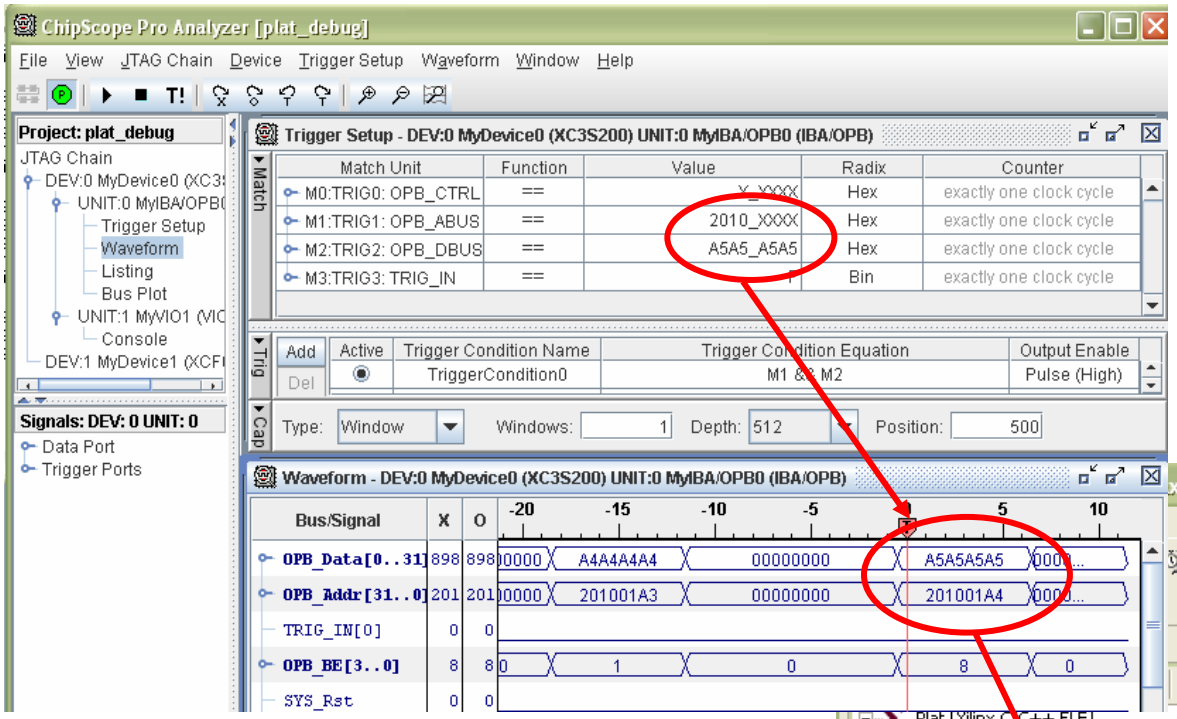
ChipScope Pro Analyzer

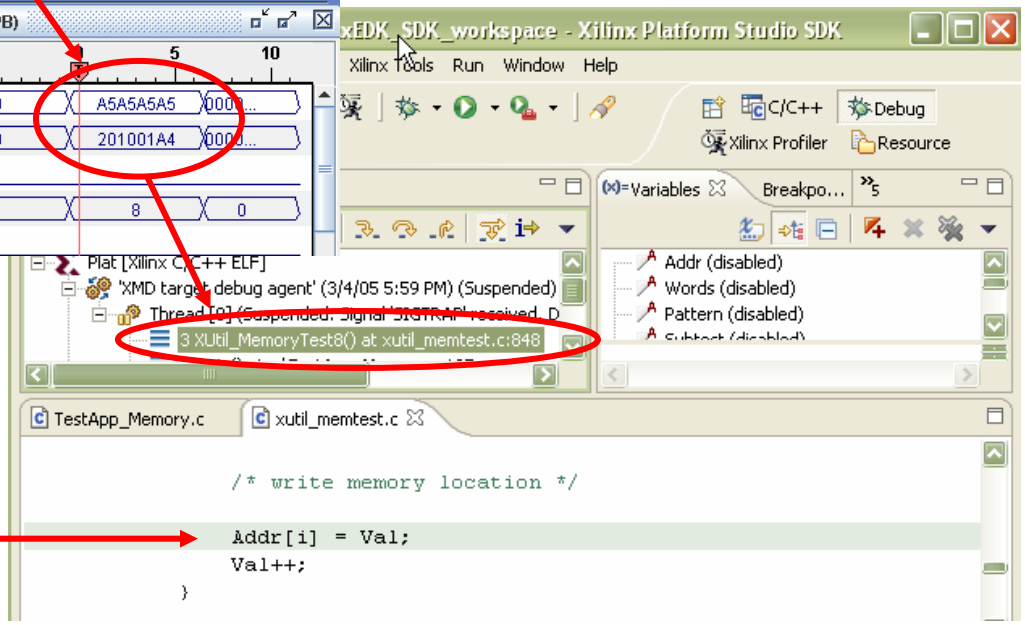
1. Instantiate an Integrated Bus Analyzer (IBA) and connect to the core's SW debug interface
2. Start the SW debugger and ChipScope Pro Analyzer on the host machine
3. Load the SW image (ELF) to the target and configure the analyzer's trigger
4. Arm the IBA, run the software and verify the results



Cross Triggering

Enabling success from the center of technology™





- ChipScope Pro triggering debugger example
- Complex trigger condition detects address and data value simultaneously
- Suspends software routine within a few clock cycles

- Enabling better insight into the HW / SW code dynamics

Agenda

- Define debugging
- Describe tool flow
- Tour of Software Development Kit
- SDK demo
- Platform debug - cross probing with ChipScope Pro
- **ChipScope Pro demo**
- Advanced debugging
- Lauterbach demo

ChipScope Pro Demo

- **Show IBA/ICON cores in XPS**
- **Launch SDK in debug perspective**
- **Launch ChipScope Pro Analyzer**
- **Cross trigger**
  - Set ChipScope Pro trigger condition to break software
  - Set software breakpoint to trigger ChipScope Pro
- **Show memory corruption detection**

Agenda

- Define debugging
- Describe tool flow
- Tour of Software Development Kit
- SDK demo
- Platform debug - cross probing with ChipScope Pro
- CS Pro demo
- **Advanced debugging**
- Lauterbach demo

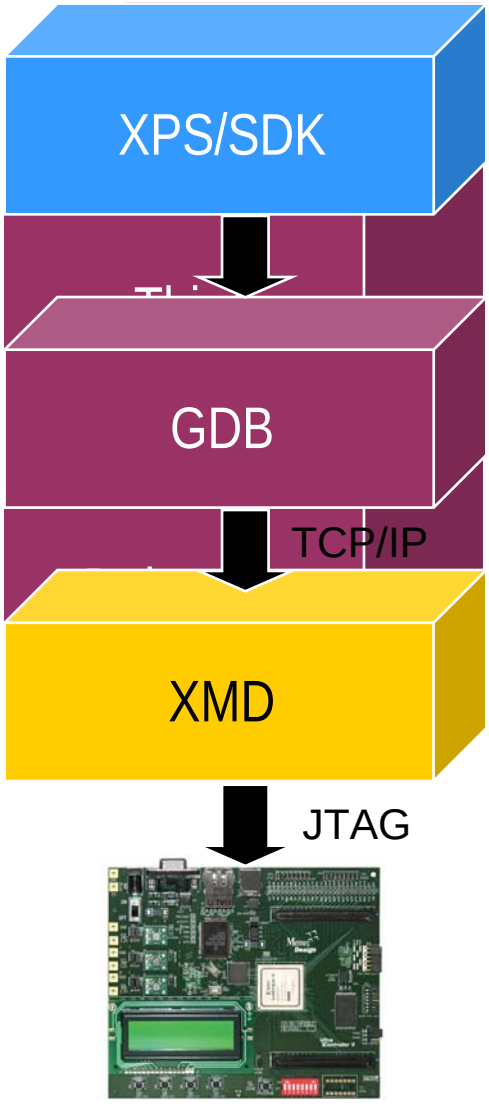
Advanced Debugging

- **Advanced debugging?**
  - Multi-threaded applications/operating systems
  - Memory issues such as stack and heap
  - Interrupt service routines
  - Many, many other issues…
    - Code Coverage
    - Performance analysis
    - Memory leaks
    - Too much to cover in 90 minutes!

Multi-threaded and Operating Systems

- **Multi-threaded applications and operating systems task switch**
  - Also referred to as context switch
- **Active task is dependant on**
  - External stimulus
  - Task priorities
  - System tick
- **Traditional source level debuggers "step" to next instruction**
  - Did an interrupt occur?
  - Did the scheduler request a context switch?
- **Need a debugger that is "aware" of the OS context switch**

Third Party Debuggers



Company	Product	Processor
Corelis	CodeRunner	PPC
GreenHills	MULTI	PPC
IBM	RISCWatch	PPC
Mentor	EDGE	PPC/MB
Lauterbach	TRACE32	PPC/MB
LynuxWorks	Totalview	PPC/MB
WindRiver	ICE/Trace	PPC

- Supported OS details can be found at:

  http://www.xilinx.com/ise/embedded/epartners

Common Memory Issues

- **Memory problems are insidious**
  - Leaks
    - Program allocates more and more memory over time
    - System eventually runs out of memory and fails
      - Hidden problem is that failing code has nothing to do with leak
      - Lead to random failures that field just attribute to needing a reboot
  - Fragmentation
    - Allocated memory that is distributed into smaller blocks
    - Inefficient block usage causes memory allocation error over time
  - Corruption
    - Stack & Heap - Not enough space allocated
    - Any code with pointers can corrupt memory
      - Writing off the end of an array
      - Writing to freed memory
      - Dangling pointers
    - Corrupted systems are completely unpredictable

ISRs

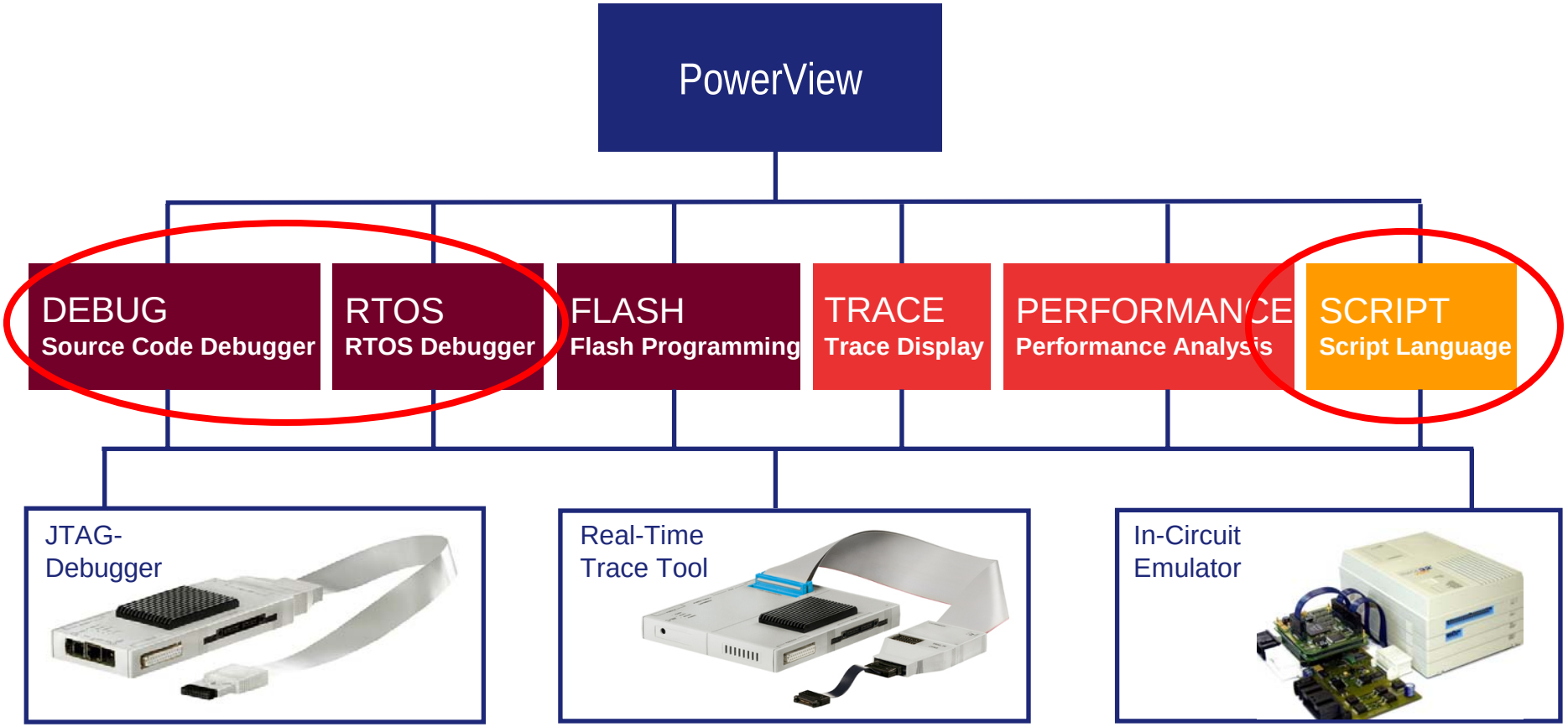
- **Unfortunately there is no “correct answer” on debugging them**
- **Some tips**
  - Keep the ISR short
  - Create an ISR “map”
    - Frequency
    - Max execution time
  - Prioritize
  - Avoid loops
  - Don’t process data, queue it for another process
- **Simple solution in some cases is to simply instrument the mainline code**
  - Writing to console or file may incur long delays
  - Set breakpoints on each line of non-ISR code and continue to next breakpoint to avoid missing an interrupt
  - Write to global variable in a known location and use ChipScope to trap on access

Agenda

- Define debugging
- Describe tool flow
- Tour of Software Development Kit
- SDK demo
- Platform debug - cross probing with ChipScope Pro
- ChipScope Pro demo
- Advanced debugging
- **Lauterbach demo**

Lauterbach TRACE32

Enabling success from the center of technology™



- **PowerView – The uniform GUI for all products**

Lauterbach Demo

- **Launch script to run through a uClinux debug session**
  - Kernel awareness
  - Stack usage
  - Source and assembly debug views of standard Linux application “uname”
  - Function trace chart
  - File system awareness
- **Other debugger features**

What's Next?

- **Contact your local FAE**
- **Get Xilinx tools**
  - ISE WebPack can be downloaded free
  - EDK is often bundled with Avnet development boards during Avnet Speedway promotions
- **Get a development board**
- **Create your own embedded processor design!**
  - Attend Avnet Speedway workshop for a quick start