Introduction to MicroBlaze Hardware Development

Featuring ISE Embedded Design Suite 13

Accelerating Your Success™

Course Objectives

- **Understand the MicroBlaze development flow and how to use the Xilinx embedded systems tools**
- **Introduce the new AXI Interface Specification**
  - Explore AXI Plug-and-Play IP
- **Utilize the Xilinx embedded systems tools to**
  - Design a MicroBlaze System
  - Develop a Software Application
  - Simulate and Debug an Embedded System
  - Create a bootloader and program serial flash

Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

AVNET electronics marketing

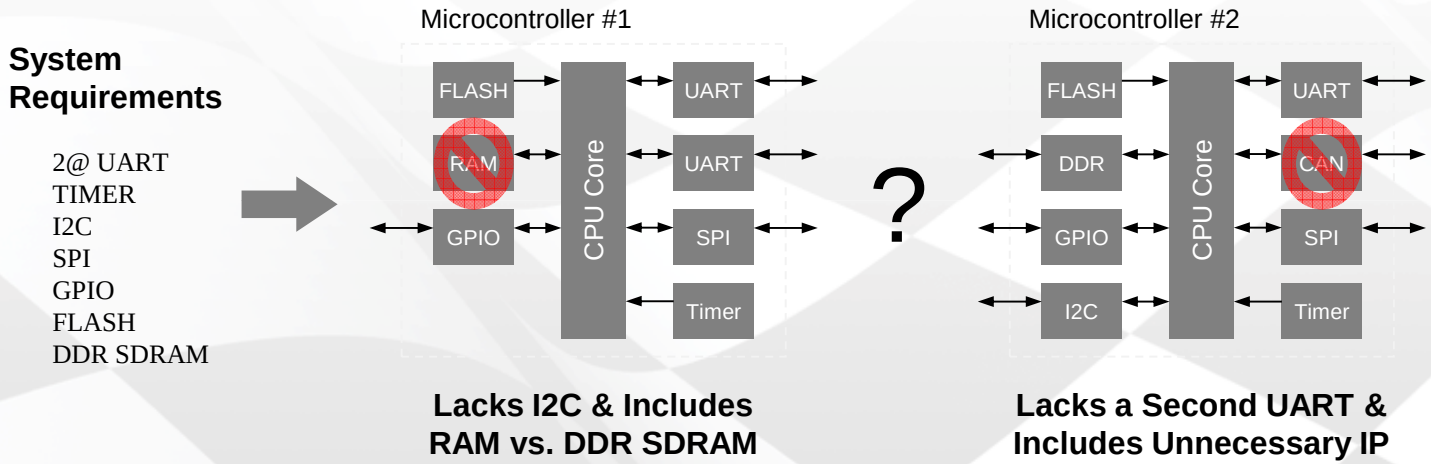
SPEEDWAY DESIGN WORKSHOPS

Why integrate a processor on an FPGA ?

Ideal mix of peripherals…

1

- **Difficult to Find the Required Mix of Peripherals in Off the Shelf (OTS) Microcontroller Solutions**
  - Even with variants, compromises must still be made



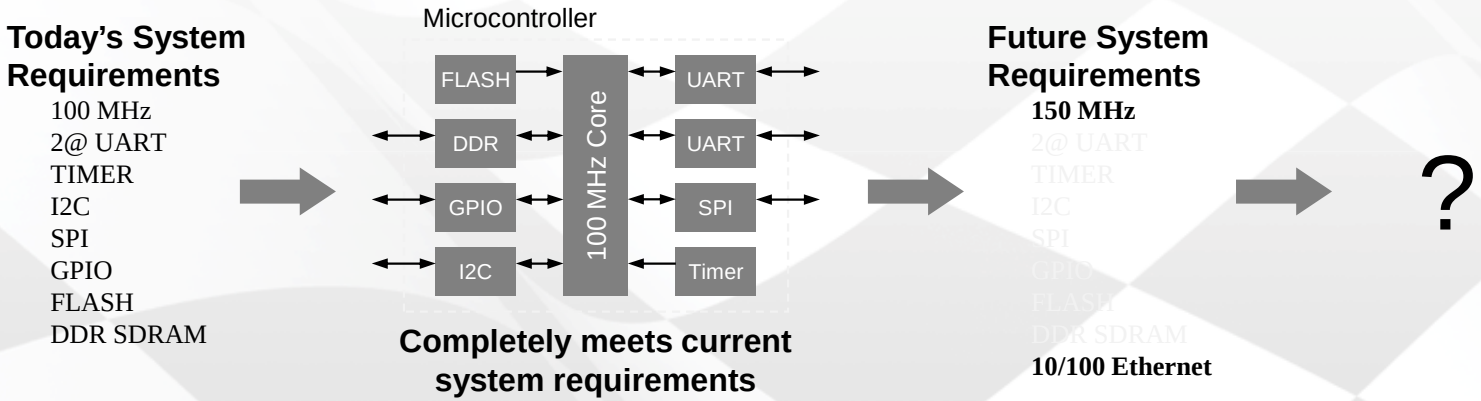
  - Must decide between inventory impact of multiple configurations or device cost of comprehensive IP

FPGAs flexibility allows ideal mix of peripherals for YOUR application

Why integrate a processor on an FPGA ?

Changing Requirements…

- **Selecting a Single discrete Processor Core with Long Term Solution Viability is Difficult at Best**
  - Future proofing system requirements is difficult at best



  - Changing discrete processor cores to accommodate new requirements consumes valuable design resources

Xilinx's processor cores offer – Performance scaling YOU control!

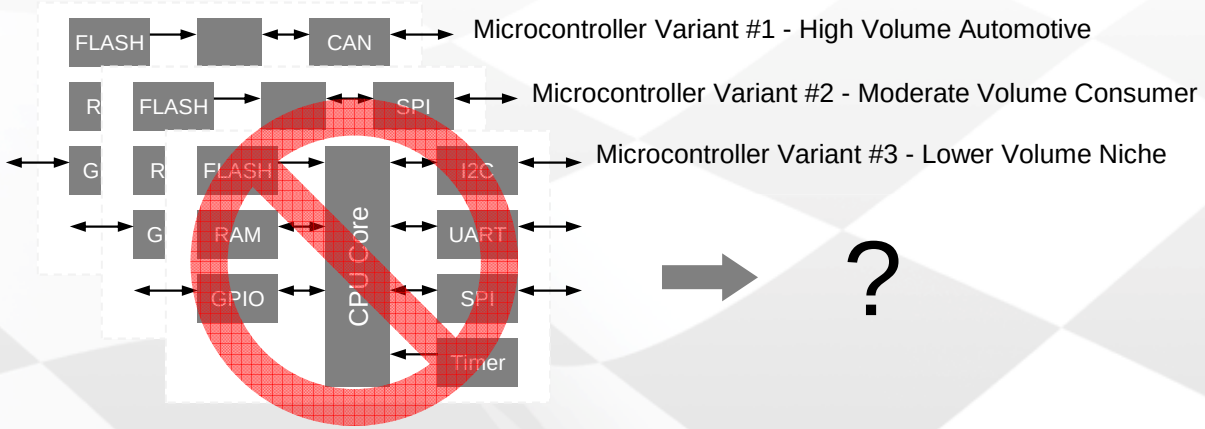
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Why integrate a processor on an FPGA ?

Avoiding Obsolescence…

- **Without Direct Ownership of the Processing Solution, Obsolescence is Always a Concern**
  - A single core is used to create a family of µCs



  - The sheer number of µC configurations can lead to obsolescence of lower volume configurations/variants

Longevity of FPGAs Approaches Longest Available Microcontrollers in the Market

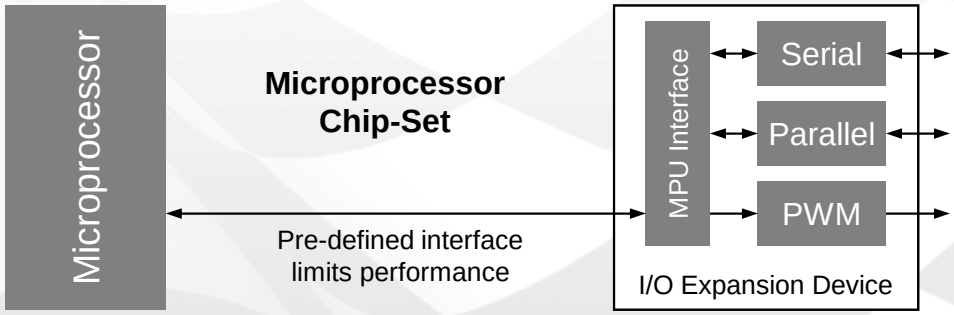
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Why integrate a processor on an FPGA ?

Reducing cost, board complexity…

- **Many Microprocessor Based Solutions Provide Limited On-Chip Peripheral Support**
  - Manufacturers create I/O devices to extend the functionality of the base processor core

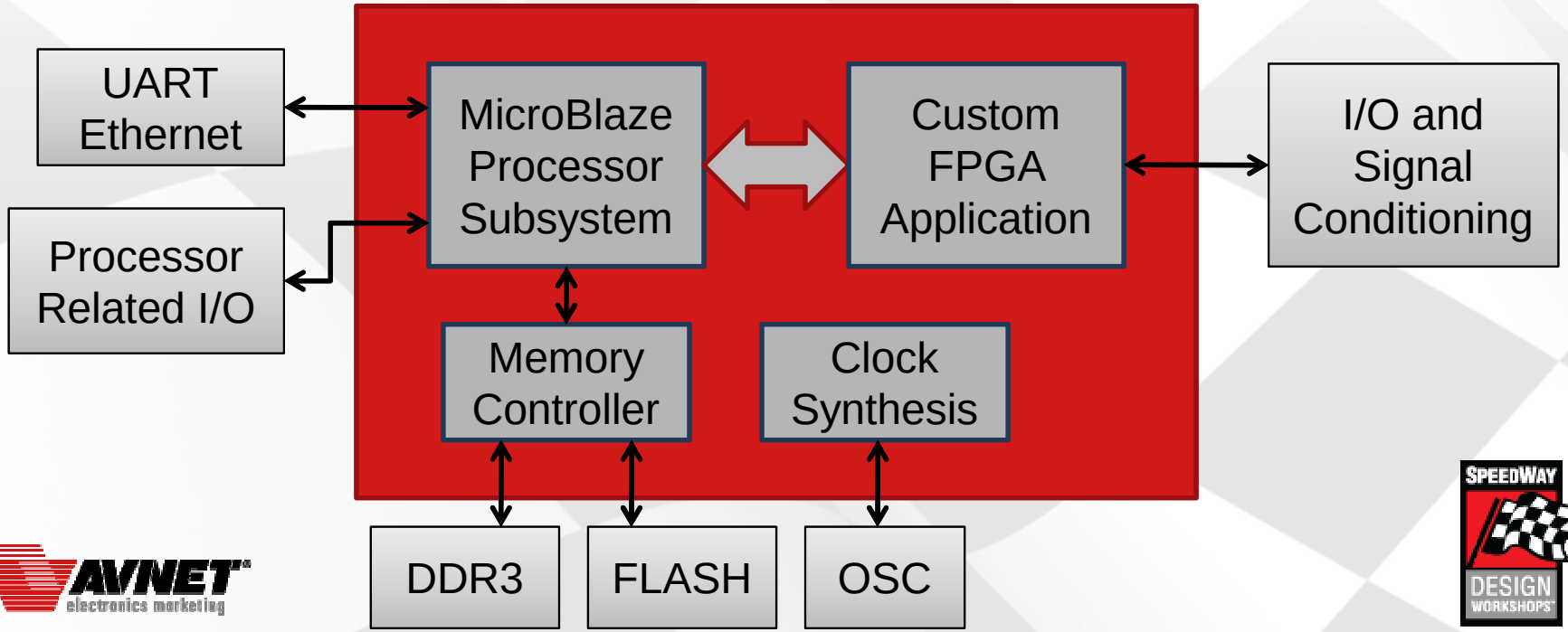


  - An external, pre-defined interface between a µP and its support device limits overall system performance

FPGA Integration is next logical step…

Xilinx System on a Chip

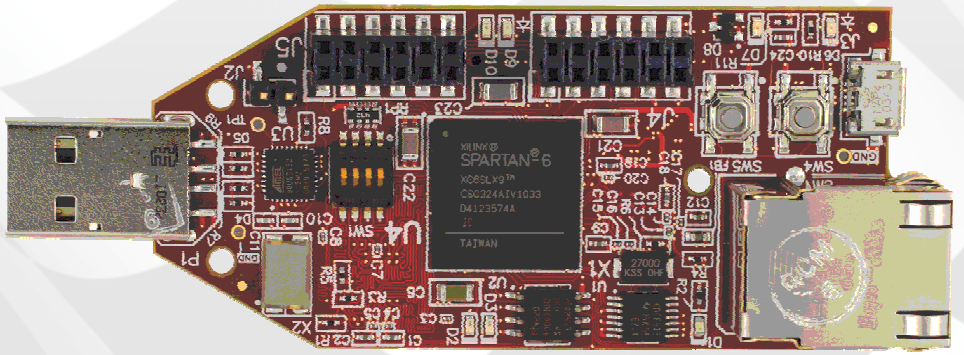
- Xilinx embedded processor innovation
  - Processor sub-system
  - Optional off-chip memory
  - Application-specific FPGA logic
  - Interface to the real world
  - System Synchronization



Xilinx® Spartan®-6 LX9 MicroBoard

Compact Embedded Platform

- **USB-stick form factor card**
- **Many reference design systems and tutorials available**
- **Software development and ChipScope licenses included**
- **Expansion Interfaces**

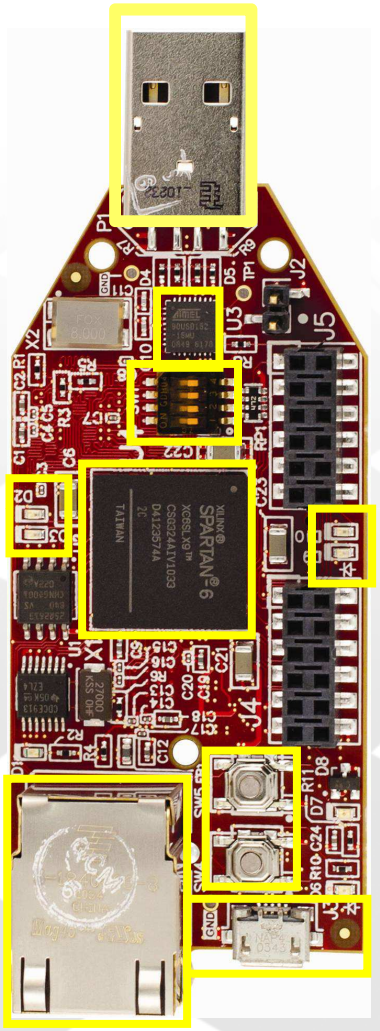


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

MicroBoard Peripherals

- **Xilinx Spartan-6 FPGA**
  - XC6SLX9-2CSG324
- **USB JTAG Circuitry**
- **10/100 Ethernet**
- **Micro USB-UART port**
- **Pushbuttons and LED's**
- **4-bit DIP Switches**

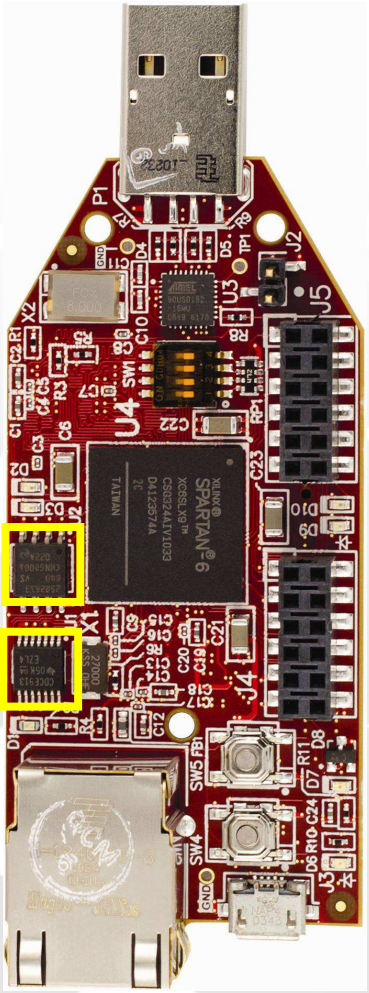


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

MicroBoard Clock Generation and Memory

- **Programmable clock chip**
  - Texas Instruments CDCE913
  - 3 outputs, upto 230MHz
- **64-MB LPDDR SDRAM (on back)**
  - Micron MT46H32M16
- **128 Mb Multi-I/O SPI Flash**
  - Micron N25Q128

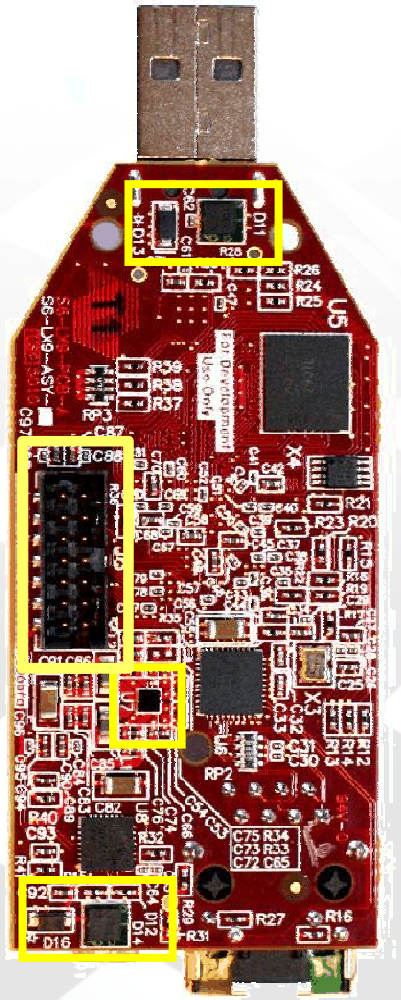


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

MicroBoard Power and Protection

- **Very compact 3-rail Power**
  - 5x5 mm
- **USB over-voltage and ESD protection**
- **Hi-Speed JTAG Access**
  - Requires External Cable



AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

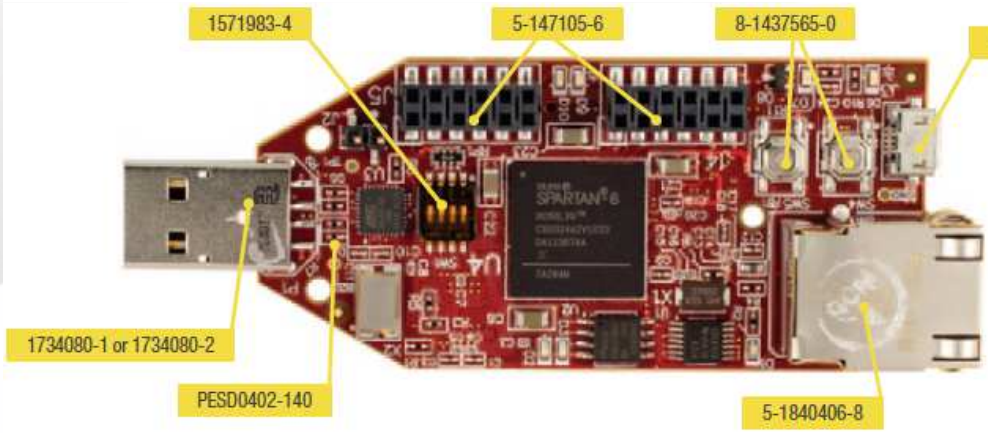
TE Interconnect Solutions for the Xilinx® Spartan®-6 FPGA LX9 MicroBoard

TE connectivity

Authorized Distributor

TE Connectivity designs, manufactures and markets engineered electronic components for a broad array of industries and applications.

These solutions include circuit protection, switches, cables and board-to-board and integrated magnetics I/O connectors – as shown on the LX9.



TE PART NUMBER	CIRCUIT FUNCTION ON SPARTAN-6 FPGA LX9 MICROBOARD
ZEN056V130A24LS	Provides overcurrent and overvoltage protection for USB ports
PESD0402-140	Provides ESD protection for USB ports
1734080-1 or 1734080-2	Connects PCB to PC or Laptop via USB port
1571983-4	User-defined PCB configuration and I/O test
5-147105-6	Allows for expansion ports off of FPGA
8-1437565-0	User-defined switches (PCB power, reset or LED blinking)
1981668-1 or 1981584-1	Connects PCB to PC or Laptop via USB cable
5-1840406-8	Connects PCB to Ethernet cable
2032807-1	USB cable that connects to PCB to PC or Laptop

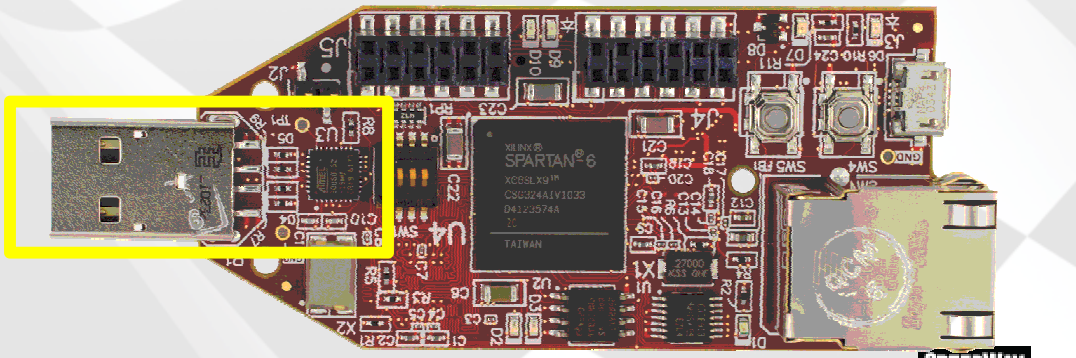
Note: ZEN056V130A24LS and 2032807-1 are also included in the kit but not shown.

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

USB JTAG Circuit

- **No external programming cable required.**
- **Compatible with iMPACT, ChipScope, SDK Debugger**
- **Full-speed**
- **Available for industry adoption**
  - Module and chip solutions available

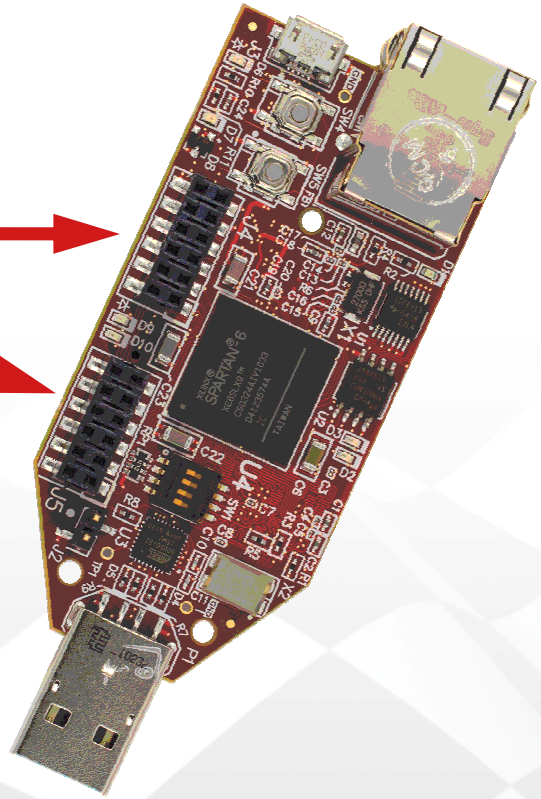


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Expansion Boards

- **Two PMOD sockets**
- **Many low-cost add-on peripheral modules, as low as $10**
  - ADC
  - Wireless/Bluetooth
  - SD Card
  - LCD

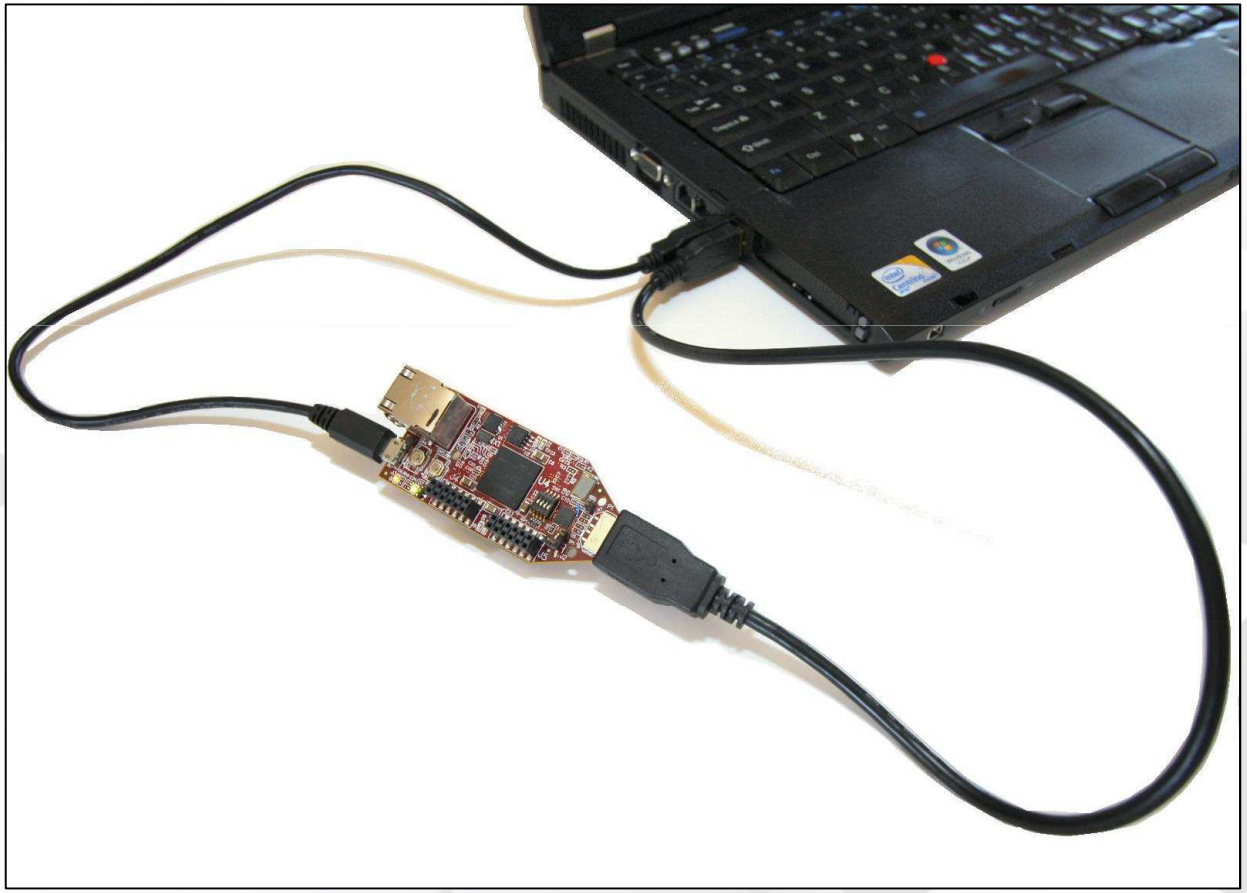


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Lab Equipment and Setup

- **UART interface through MicroUSB cable**
- **JTAG interface through USB extension cable**

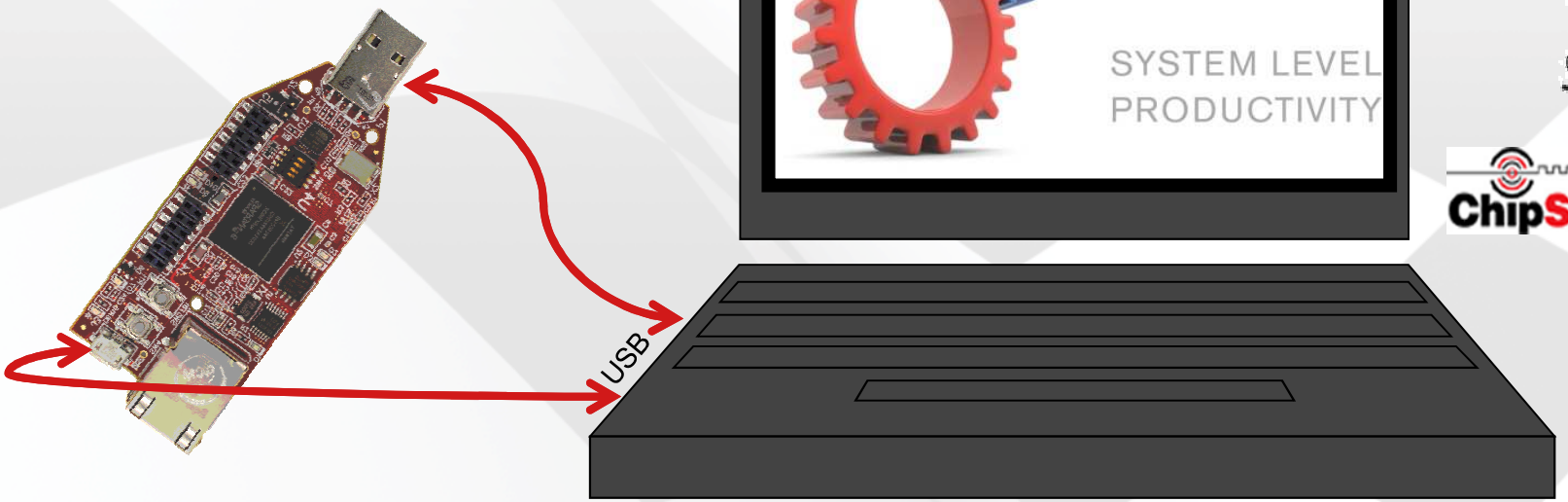


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Lab Setup and Design Tools

- **Xilinx ISE Design tools**
  - ISE
  - EDK
    - XPS
    - SDK
  - ChipScope
  - ISim



Platform Studio

SDK

ChipScope Pro

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

ISE Design Suite Software Products

Included in Spartan-6 LX9 MicroBoard Kit

ISE WebPACK

Limited Devices

PlanAhead

ISE Foundation
ISim Light

Logic Edition

ChipScopePRO
Serial I/O Toolkit

PlanAhead

ISE Foundation
ISim Full

Embedded Edition

EDK

Logic Edition

DSP Edition

System Generator

Logic Edition

System Edition

EDK

System Generator

Logic Edition

Targeted Standalone Products

Software Developers	Debug	Embedded Developers	DSP Developers
SDK	ChipScope	EDK	System Generator

SDK

ChipScope Pro

Platform Studio

SDK

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Operating System Support

Many OS options including Windows and Linux

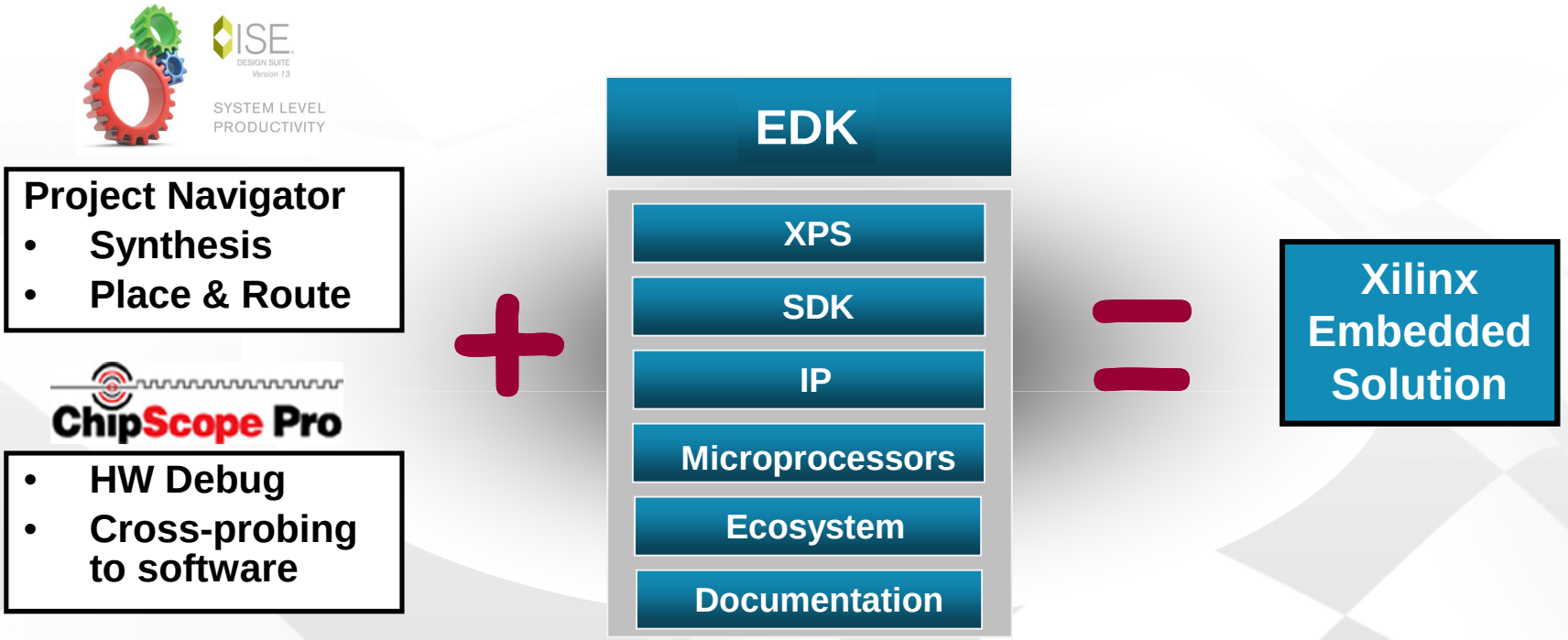
ISE Design Suite 13.x Operating Support	Windows XP Pro 32/64	Windows 7 Pro 32/64	Windows Server 2008	Red Hat Linux Enterprise 4 WS32/64	Red Hat Linux Enterprise 5 WS32/64	SUSE Linux Enterprise 11 32/64
ISE Design Suite	✓	✓	✓	✓	✓	✓
ISIM	✓	✓	✓	✓	✓	✓
ISE WebPack	✓	✓	✓	✓	✓	✓
Chipscope Pro	✓	✓	✓	✓	✓	✓
EDK	✓	✓	✓	✓	✓	✓
SDK	✓	✓	✓	✓	✓	✓
SYSGEN	✓	✓	✓	✓	✓	✓

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Design Tool Flow

*Must have **EDK** and **ISE** for embedded systems development. ChipScope adds debugging capability*



Value of using Xilinx embedded systems

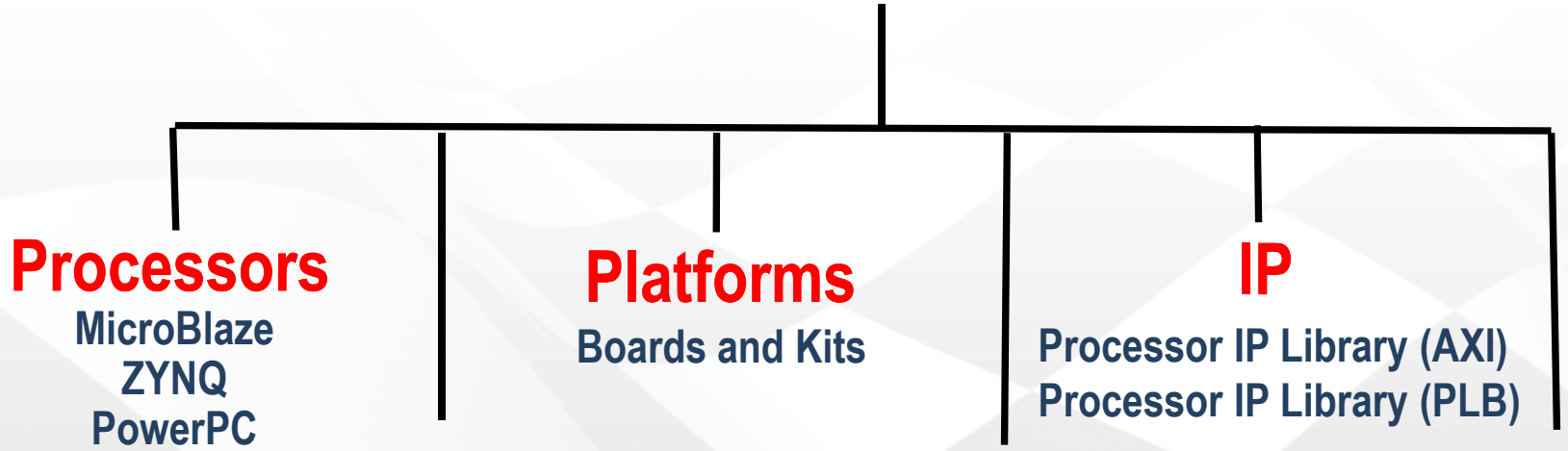
- One tool chain for hard and soft microprocessors
- Reduces board complexity and cost
- Allows users focus on ***their*** intellectual property
- Software Developers only need SDK

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

What's in the Embedded Development Kit?

Xilinx Embedded Edition /
Embedded Development Kit (EDK)



Processors
MicroBlaze
ZYNQ
PowerPC

Platforms
Boards and Kits

IP
Processor IP Library (AXI)
Processor IP Library (PLB)

Tools Suite
HW – Xilinx Platform Studio (XPS)
SW – Software Development Kit (SDK)

EDK includes both tools
SDK can be purchased separately

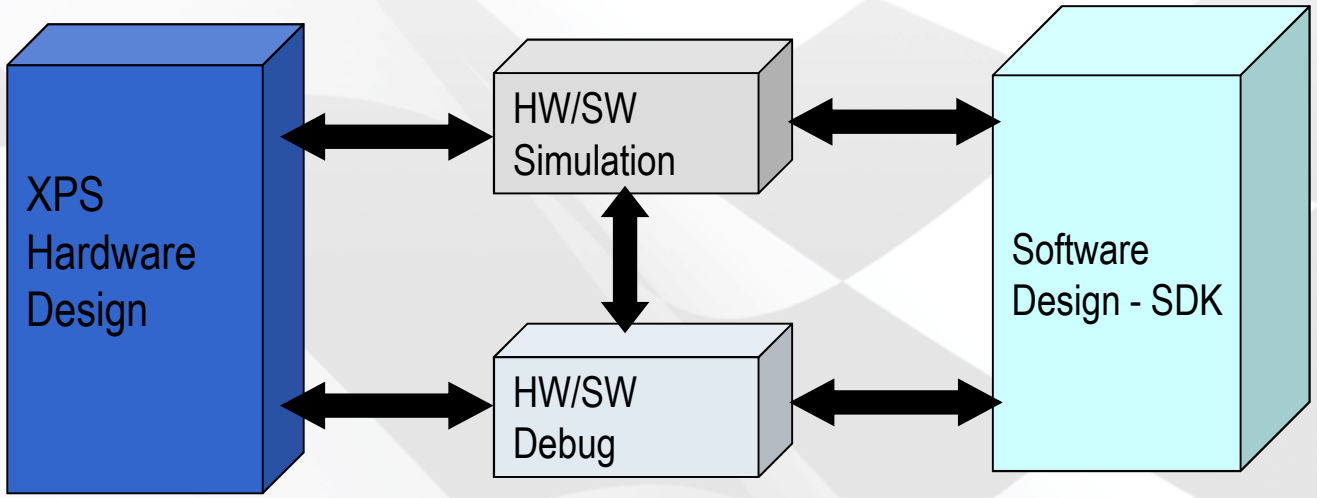
Documentation
Reference Designs
Examples
Documentation
Support

Ecosystem
RTOS
3rd party debuggers
3rd party trace

EDK Provides Comprehensive Support

EDK Tool Suite

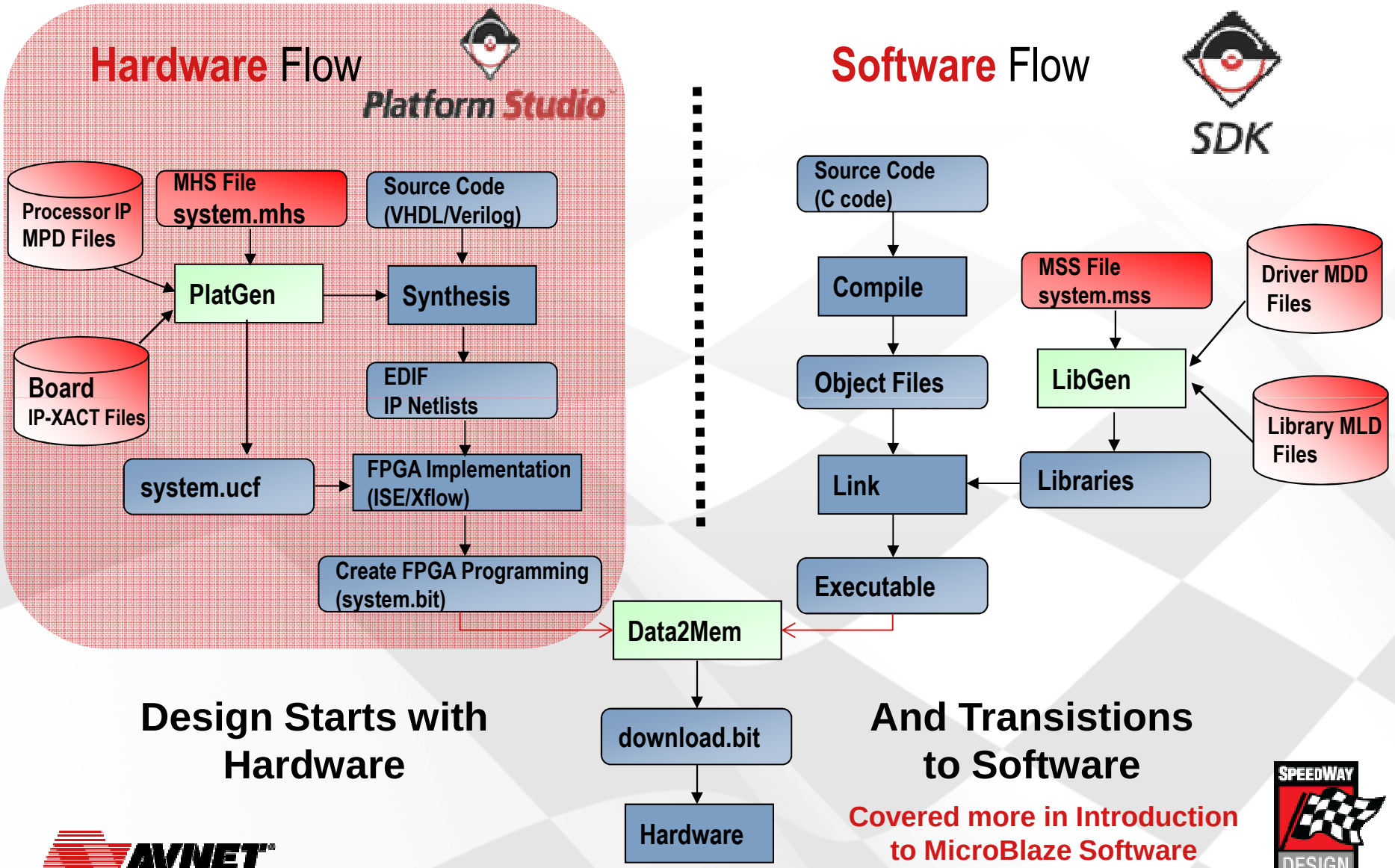
- **Xilinx Platform Studio** (**XPS**)
  - Design environment for processor subsystem
  - Xilinx Microprocessor Project (**XMP**) file
  - Microprocessor Hardware Specification (**MHS**) file
  - Bus Functional Model (**BFM**) Simulation
  - ChipScope Pro logic analyzer integration
- **Software Development Kit** (**SDK**)
  - Project workspace
  - Hardware platform definition (**XML**)
  - Board Support Package (**BSP**)
  - Software application
  - Software debugging



AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Detailed EDK Design Flow



Design Starts with Hardware

And Transistions to Software

Covered more in Introduction to MicroBlaze Software Development Speedway

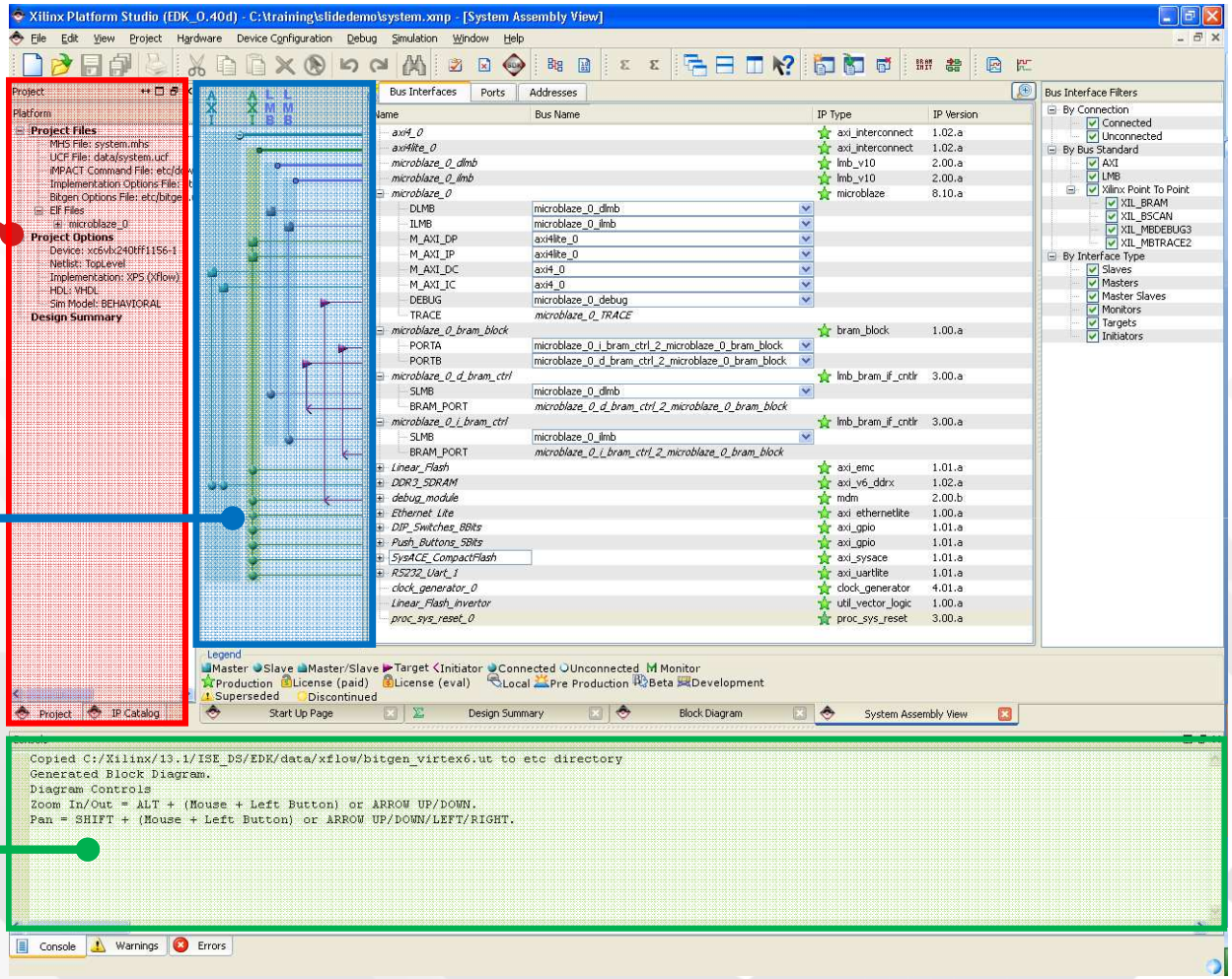
Xilinx Platform Studio (XPS)

System Assembly View – Bus Interface Tab

Project Information Area
- Lists project files
- Tabs select Project files and IP Catalog

Connectivity Panel
- Shows interface connections

Console
- Shows Messages and TCL commands



AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

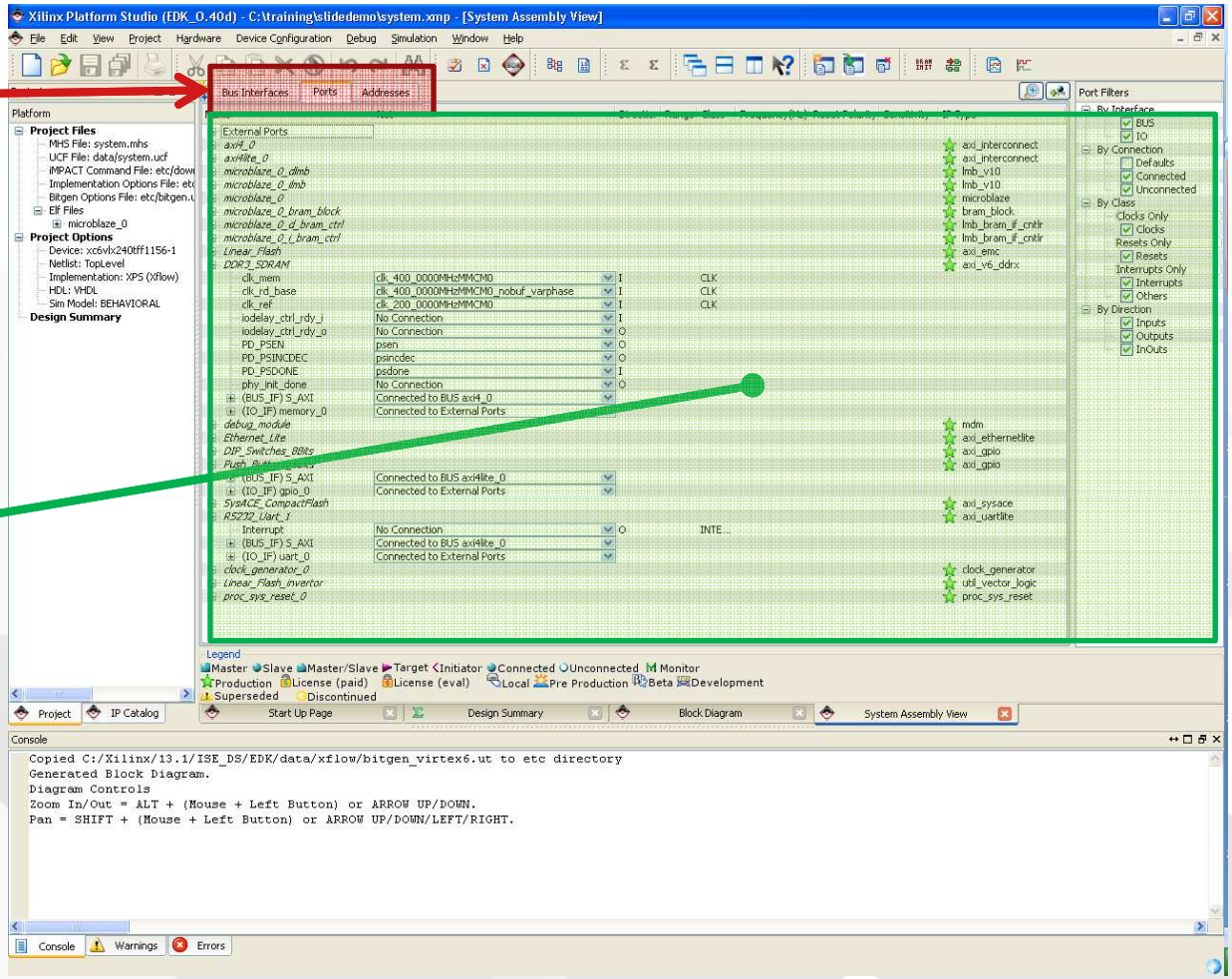
Xilinx Platform Studio (XPS)

System Assembly View – Ports Tab

Tabs:

- Bus Interfaces
- Ports
- Address

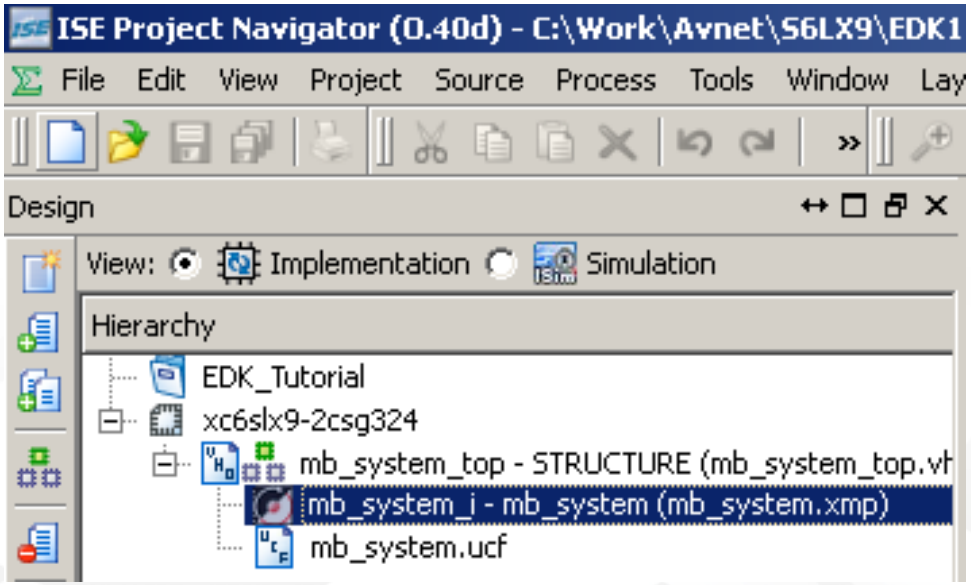
System Assembly View



AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

XMP File



- **The XMP file is the project support file for XPS**
- **Takes on the name of the project, *<project>.xmp***
  - In the labs, mb_system.xmp is used
- **Contains and controls**
  - Files that make up the XPS project; that is, the MHS file
  - Tools settings
  - GUI settings
- **The XMP file is typically the embedded source file that contains the XPS component in Project Navigator**
  - Is the file type that identifies the embedded processor component source
  - Alternative to using *system_stub.vhd/v*

Checkpoint!

- **What EDK tool is used to create and define the MicroBlaze Processor?**

  XPS – Hardware

- **What EDK tool is used to create MicroBlaze Software?**

  SDK – Software

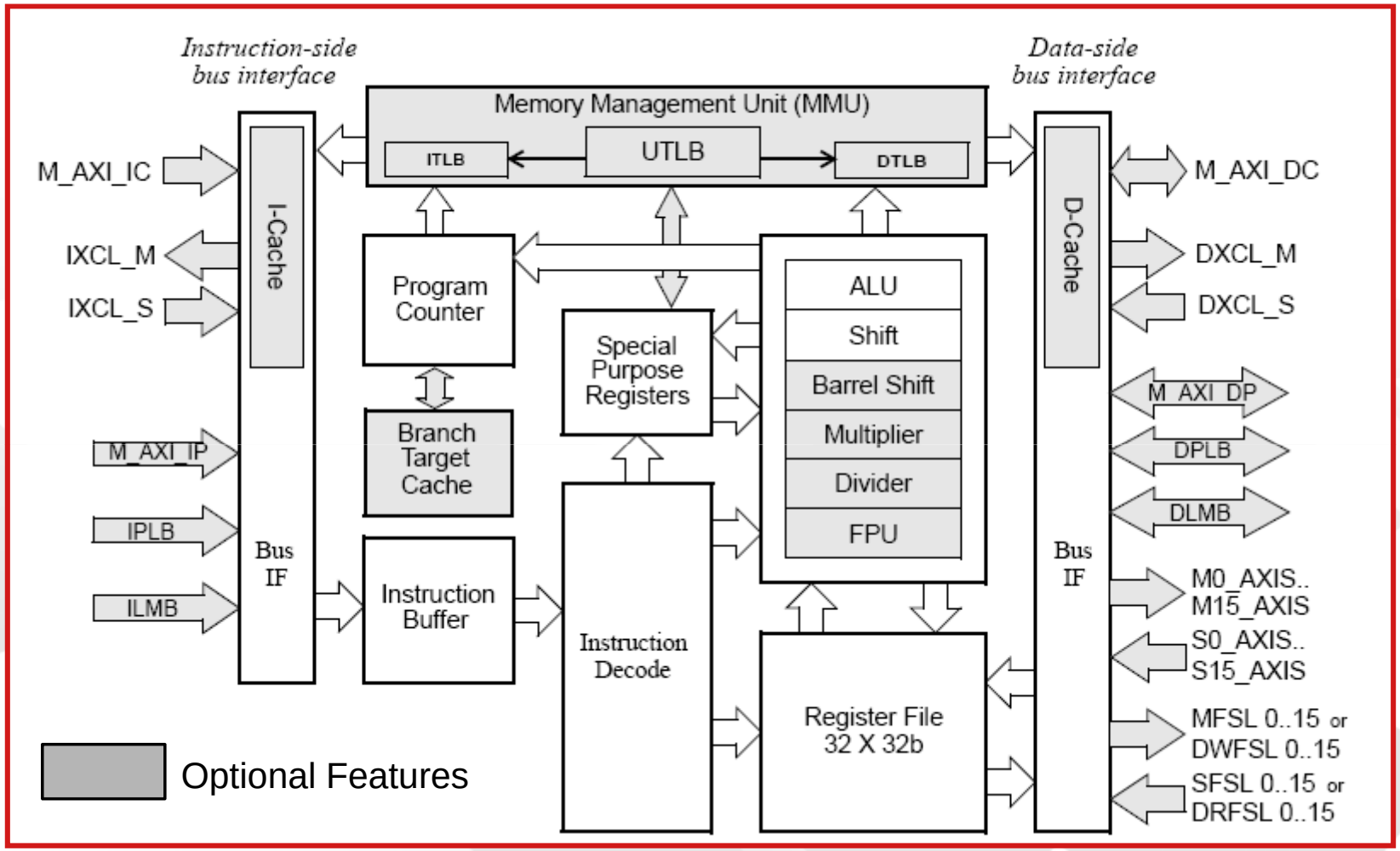
- **What is the XMP File?**

  Contains files that make up XPS Project including tool settings

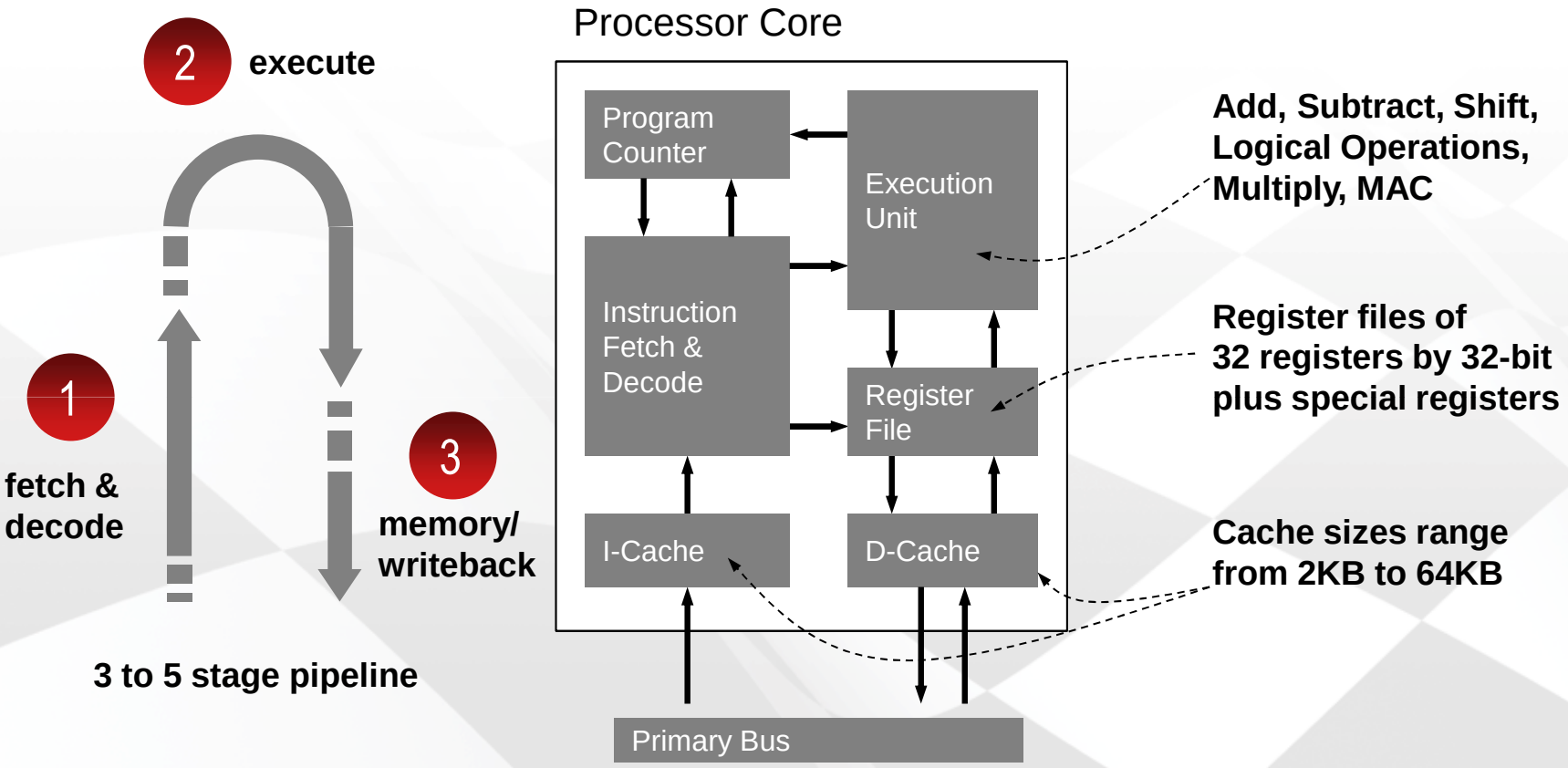
Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

MicroBlaze Processor Block Diagram



The Processor Core



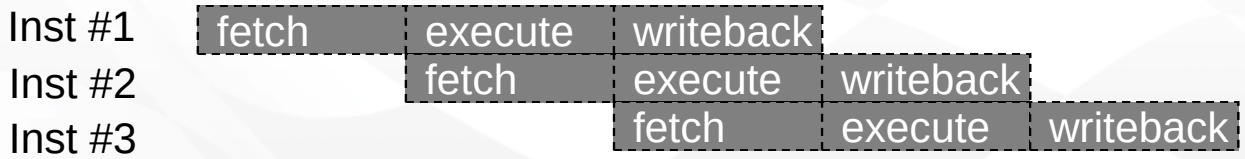
*Typical pipeline stages, functions, quantities & sizes

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Instruction Pipelining

- **Instruction pipelining overlaps multiple instructions as they are executed**



- **Improves the utility of the various functional blocks in the processor core - fetch/decode, register file, ALU**
- **Dramatically Improves the performance of the core**
- **Increases the complexity of the processor core**
- **Incorrect prediction requires pipeline flushing**

MicroBlaze Processor Basic Architecture

- **Embedded soft RISC processor (version 8.10a)**
  - 32-bit address and data buses
  - 32-bit instruction word (three operands and two addressing modes)
  - 32 registers (32-bit wide)
  - Three or five pipeline stages (three stages if area optimization is selected)
  - Big-endian format
- **Buses**
  - Full Harvard architecture
  - AXI4 interface
  - LMB for connecting to local block RAM (faster), instruction, and data (user selectable)
  - AXI4 streaming interface: dedicated, unidirectional point-to-point data streaming interfaces; support for up to 16 streaming interfaces
  - Dedicated master AXI ports for instruction and data caching with four (eight)-word cache line size and critical word-first access capability

MicroBlaze Processor Features

- **ALU**
  - Hardware multipliers/DSP48
  - Barrel shifter
- **Floating Point Unit (FPU)**
  - Implements IEEE 754 single-precision, floating-point standards
  - Supports addition, subtraction, multiplication, division, and comparison
- **Program counter**
- **Instruction decode**
- **Instruction cache**
  - Direct mapped
  - Configurable caching with CacheLink
  - Configurable size: 2 kB, 4 kB, 8 kB, 16 kB, 32 kB, 64 kB

MicroBlaze Processor v8.10a Performance

Microcontroller Configuration MicroBlaze with local memory and debugger, external memory controller, UART, FSL link, 2K I-Cache, 8K D-Cache			
With Performance-Optimized MicroBlaze (1.19 DMIPs/MHz) * *Note (1)*			
Virtex-6 FPGA (-3)			
223 MHz			
5788 LUTs			
Spartan-6 FPGA (-3)			
129 MHz			
3157 LUTs			

(1): Adding Branch Optimizations to Performance Optimized MicroBlaze increases performance to 1.30 DMIPs/MHz)

New MicroBlaze Processor v8 Features

- **High-performance AXI4 interface and AXI4 peripherals**
- **Memory Management Unit (MMU) implements virtual memory management**
  - PPC405 processor MMU compatible
  - Virtual memory management provides greater control over memory protection, which is especially useful with applications that can use an RTOS
- **Processing improvements**
  - New float-integer conversion and float-square root instructions
  - Speeds up
    - FP → Int conversion
    - Int → FP conversion
    - FP square root

```
#include <math.h>
…
float x=-1.0F;
…
x = sqrt(x); /* uses double precision */
```

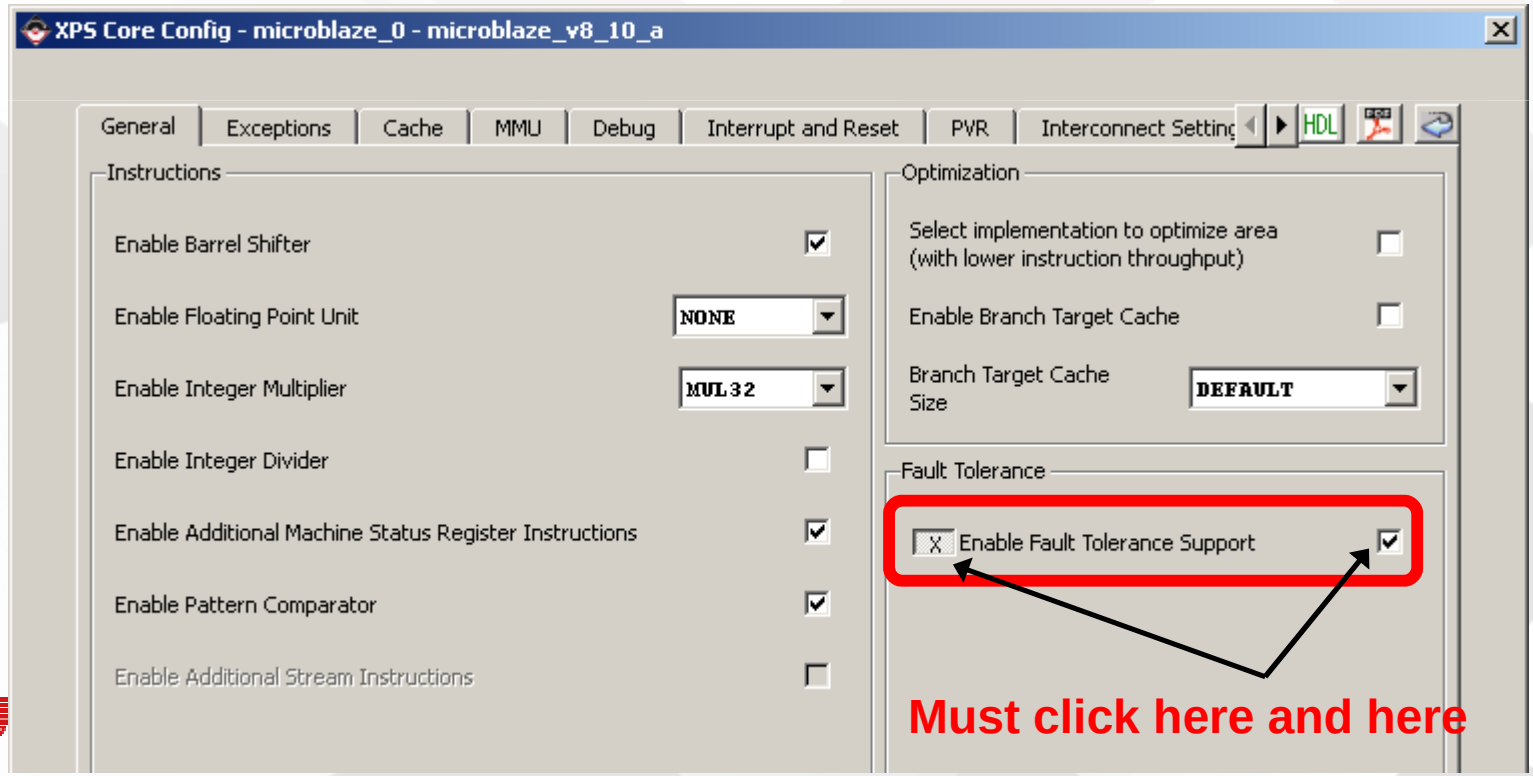
- **Enhanced XMD support**

AVNET electronics marketing

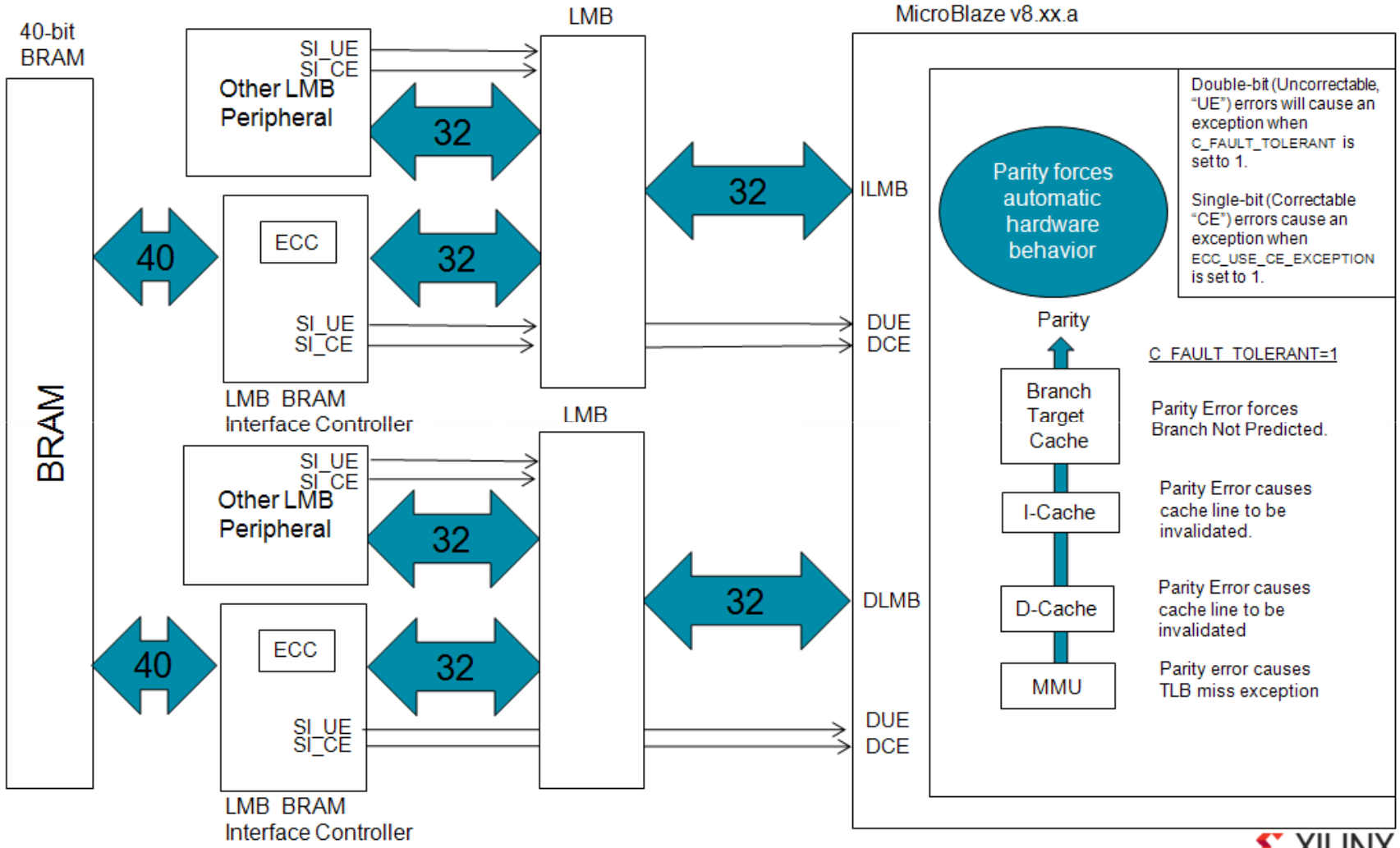
SPEEDWAY DESIGN WORKSHOPS

New MicroBlaze Processor v8 Features

- **MicroBlaze offers Fault Tolerant features**
  - ECC (Error Correction Code) support on LMB BRAMs
  - Parity support on internal BRAMs
  - Detects and correct soft errors in BRAMs
  - Significantly reduces overall failure susceptibility



MicroBlaze Fault Tolerance in 13.1 (continued)



XILINX

AVNET electronics marketing

See AR# 40863 for more details

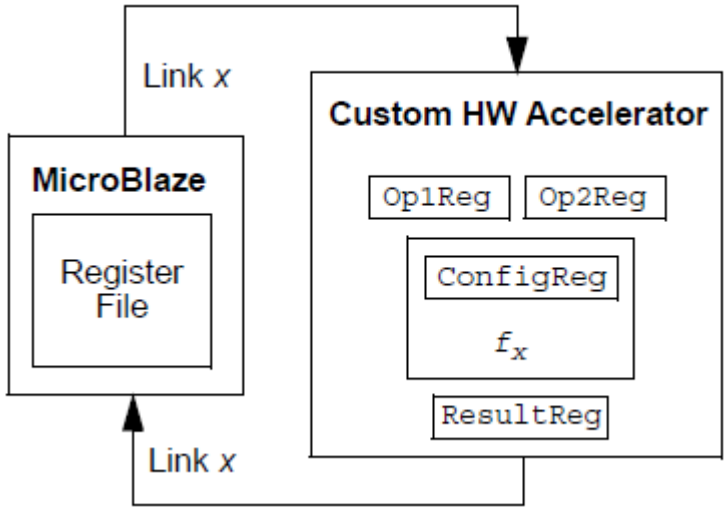
DESIGN WORKSHOPS

AXI Stream Link Interfaces

- **AXI4 streaming interface to/from MicroBlaze processor**
  - Was FSL link.
- **Unidirectional, point-to-point FIFO-based communication**
- **Built-in programmable depth FIFO**
- **Simplex connection—processor to/from FIFO**
- **Direct connection to CPU core, internal pipeline**
  - Native to MicroBlaze processor hardware
    - Enable up to 16 in/out pairs (channels)
    - Dedicated register to/from FIFO assembler instructions

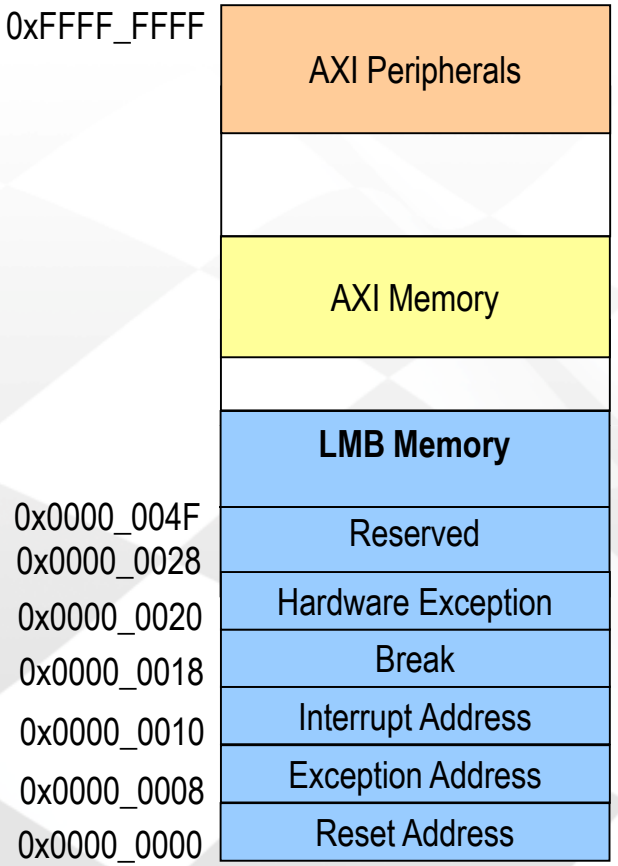
Example code:

```
// Configure fx
cput Rc,RFSLx
// Store operands
put Ra, RFSLx // op 1
put Rb, RFSLx // op 2
// Load result
get Rt, RFSLx
```



MicroBlaze Processor Memory Space

- **Memory and peripherals**
  - The MicroBlaze processor uses 32-bit addresses
- **Special addresses**
  - MicroBlaze processors must have user-writable memory from 0x00000000 through 0x0000004F
  - Each vector consists of two instructions IMM followed by a BRAI instruction to address full memory range
- **All instructions take one clock cycle, except the following**
  - Load and store (two clock cycles)
  - Multiply (two clock cycles)
  - Branches (three clock cycles, can be one clock cycle)



Checkpoint!

- **Name a few optional features of MicroBlaze.**

AXI Support , Fault Tolerance, MMU/VPU, FPU improvements…

- **What is typical performance for MicroBlaze in Spartan-6 (in MHz)?**

110-130 MHz

- **Where do the reset, interrupt and exception vectors reside in the MicroBlaze processor?**

0x00000000 through 0x00000020

- **What is the starting address of the Local Memory (LMB)?**

Immediately above the reserved address space, 0x0000004F

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Summary

- **The MicroBlaze processor balances execution performance against implementation size**
  - The MicroBlaze v8 processor supports the AXI4 interface standard and offers an MMU
  - Being a soft core processor, many features are implemented on demand
  - Supports up to 16 channel AXI4 stream
    - Simplex; up to 16 in and 16 out
    - AXI stream interface on fabric side
  - Fault Tolerance significantly reduces overall failure susceptibility

PIT STOP!

(Take a Break)

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Agenda

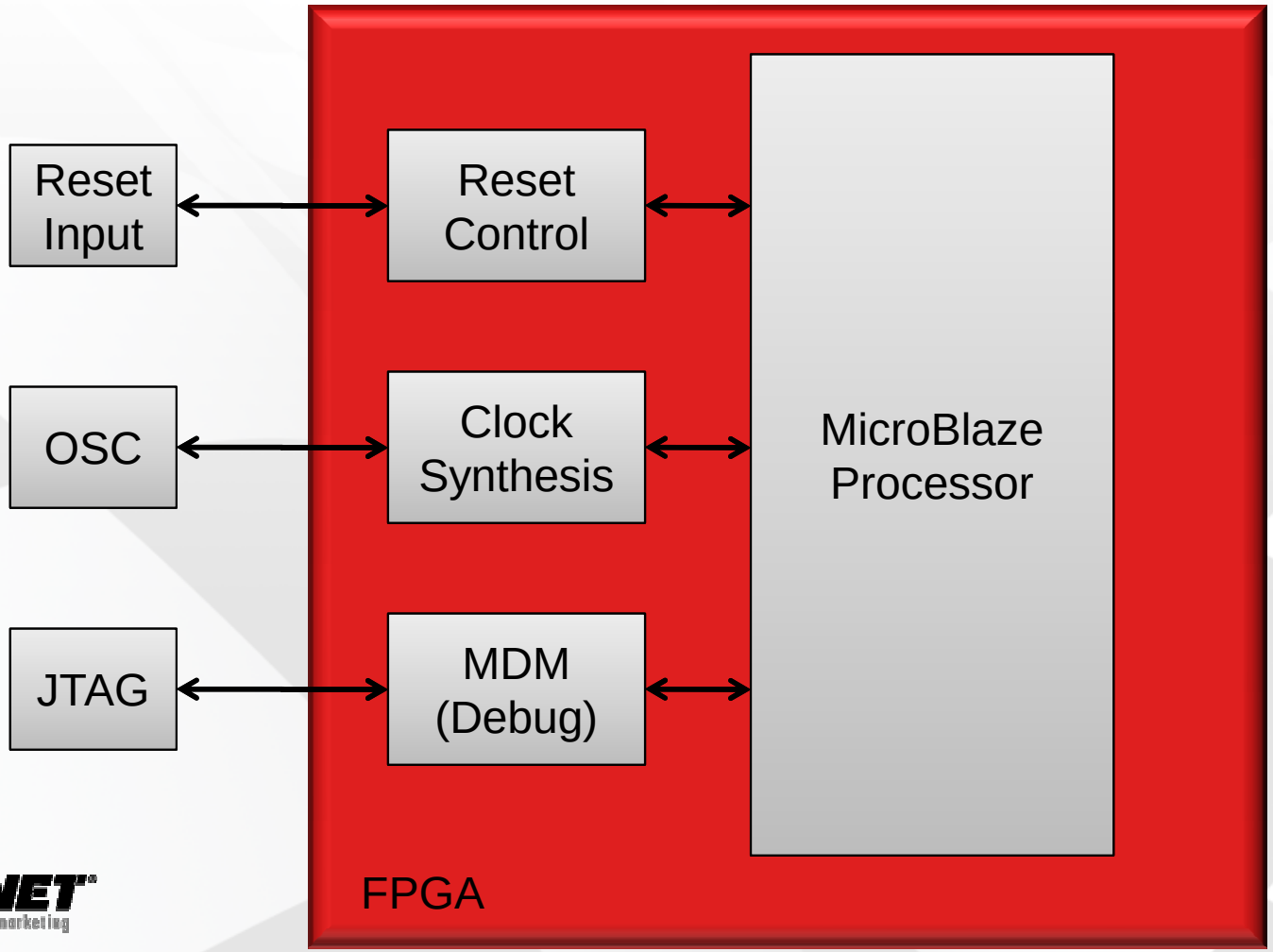
- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

System Architecture Considerations

- **At minimum, MicroBlaze requires 2-3 interfaces**
  - JTAG optional but used in nearly every design.



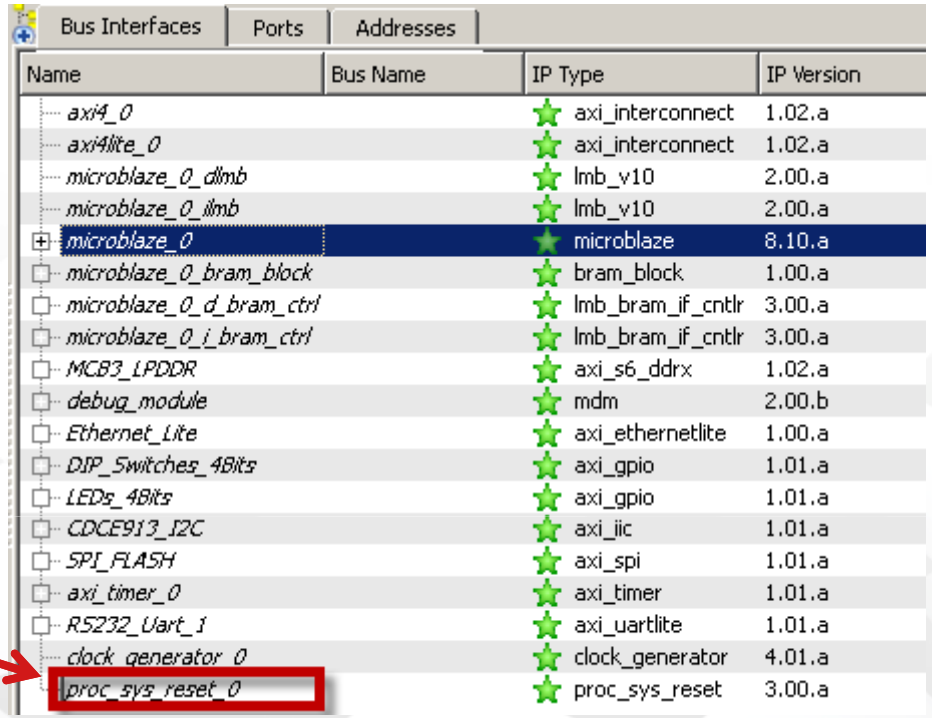
System Architecture Considerations

- **Choice of single or dual processor system**
  - MicroBlaze processor (soft core)
  - ZYNQ (ARM-based hard core processor)
  - PowerPC (hard core processor, Virtex-5)
- **System reset**
- **Clocking circuitry**
- **JTAG connectivity for hardware/software debug**
- **XPS bus interfaces**
  - AXI-4
  - AXI-4 Streaming
  - AXI-4 Lite
  - PLB

Reset Circuitry

RESET: PROC_SYS_RESET Component

- **A reset control circuit is needed to coordinate reset behavior between the system reset inputs, processor, bus, and peripheral resets.**
- **A Processor System Reset Module is a required component in an embedded system to accomplish this task.**
- **Xilinx Processor System Reset module design allows you to tailor the design to suit your application by setting certain parameters to generate the desired reset**

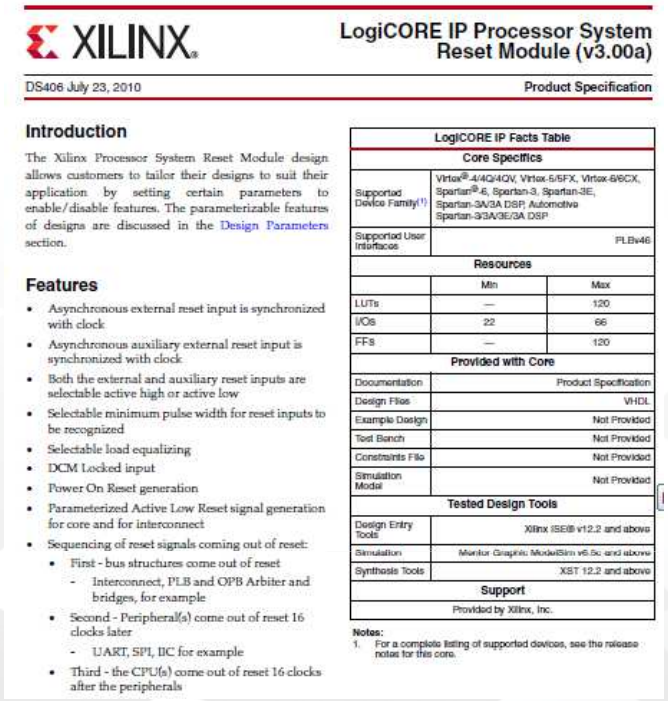


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

PROC_SYS_RESET Component

- **Asynchronous external reset input is synchronized with the clock**
- **Asynchronous auxiliary external reset input is synchronized with the clock**
- **Both the external and auxiliary reset inputs are selectable active high or active low, default = high.**
- **Selectable minimum pulse width for reset inputs to be recognized**
- **Selectable load equalizing**
- **DCM-locked input**
- **Power-on reset generation**
- **Sequencing of reset signals coming out of reset**
  - First: Bus structures come out of reset
  - Second: Peripherals come out of reset 16 clocks later
    - UART, SPI, and IIC, for example
  - Third: The CPUs come out of reset 16 clocks after the peripherals



XILINX

LogiCORE IP Processor System Reset Module (v3.00a)

DS406 July 23, 2010 — Product Specification

Introduction

The Xilinx Processor System Reset Module design allows customers to tailor their designs to suit their application by setting certain parameters to enable/disable features. The parameterizable features of designs are discussed in the Design Parameters section.

Features

- Asynchronous external reset input is synchronized with clock
- Asynchronous auxiliary external reset input is synchronized with clock
- Both the external and auxiliary reset inputs are selectable active high or active low
- Selectable minimum pulse width for reset inputs to be recognized
- Selectable load equalizing
- DCM Locked input
- Power On Reset generation
- Parameterized Active Low Reset signal generation for core and for interconnect
- Sequencing of reset signals coming out of reset:
  - First - bus structures come out of reset
    - Interconnect, PLB and OPB Arbiter and bridges, for example
  - Second - Peripheral(s) come out of reset 16 clocks later
    - UART, SPI, IIC for example
  - Third - the CPU(s) come out of reset 16 clocks after the peripherals

LogiCORE IP Facts Table		
Core Specifics		
Supported Device Family[1]	Virtex®-4/4Q/4QV, Virtex-5/5FX, Virtex-6/6CX, Spartan®-6, Spartan-3, Spartan-3E, Spartan-3A/3A DSP, Automotive Spartan-3/3A/3E/3A DSP	
Supported User Interfaces	PLBv46	
Resources		
	Min	Max
LUTs	—	120
I/Os	22	66
FFs	—	120
Provided with Core		
Documentation	Product Specification	
Design Files	VHDL	
Example Design	Not Provided	
Test Bench	Not Provided	
Constraints File	Not Provided	
Simulation Model	Not Provided	
Tested Design Tools		
Design Entry Tools	Xilinx ISE® v12.2 and above	
Simulation	Mentor Graphic ModelSim v6.5c and above	
Synthesis Tools	XST 12.2 and above	
Support		
Provided by Xilinx, Inc.		

Notes:
1. For a complete listing of supported devices, see the release notes for this core.

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

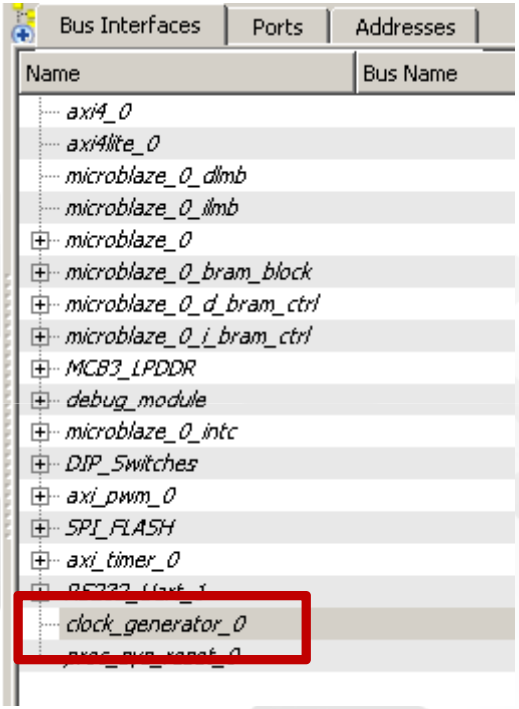
PROC_SYS_RESET Block Ports

Name	Instructions
Reset Block Inputs	
Slowest_sync_clk	Slowest Synchronous Clock - Typically AXI clock
Ext_Reset_In	External Reset Input - Active high or low based upon the generic C_EXT_RESET_HIGH
Aux_Reset_In	Auxiliary Reset Input - Active high or low based upon the generic C_AUX_RESET_HIGH
MB_Debug_Sys_Rst	Reset input generated by the MicroBlaze processor Debug Module (MDM)
Dcm_locked	DCM locked will cause all outputs to remain active until cm_locked goes high which will cause the resets to sequence to their inactive state.
Reset Block Output	
MB_Reset	Resets the MicroBlaze Processor

Clock Circuitry

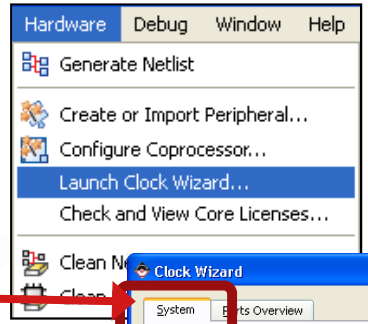
Clock Generator Component - clock_generator

- **The Clock Generator helps you configure DCM and PLL components for your embedded system**
  - One input reference clock
  - One feedback clock for clock deskew
  - Up to 16 output clocks, whose frequency range refer to DCM and PLL primitives
  - Fixed phase shift (0 to 359 degrees) for output clocks, Variable phase shift on Virtex-6 devices
  - Grouping output clocks for reducing skew among these clocks. The grouped clocks will be the output of the same DCM, PLL, or PLL with deskew adjust
  - Input and output clock frequency range check according to device family
  - Supports both active high and active low external reset



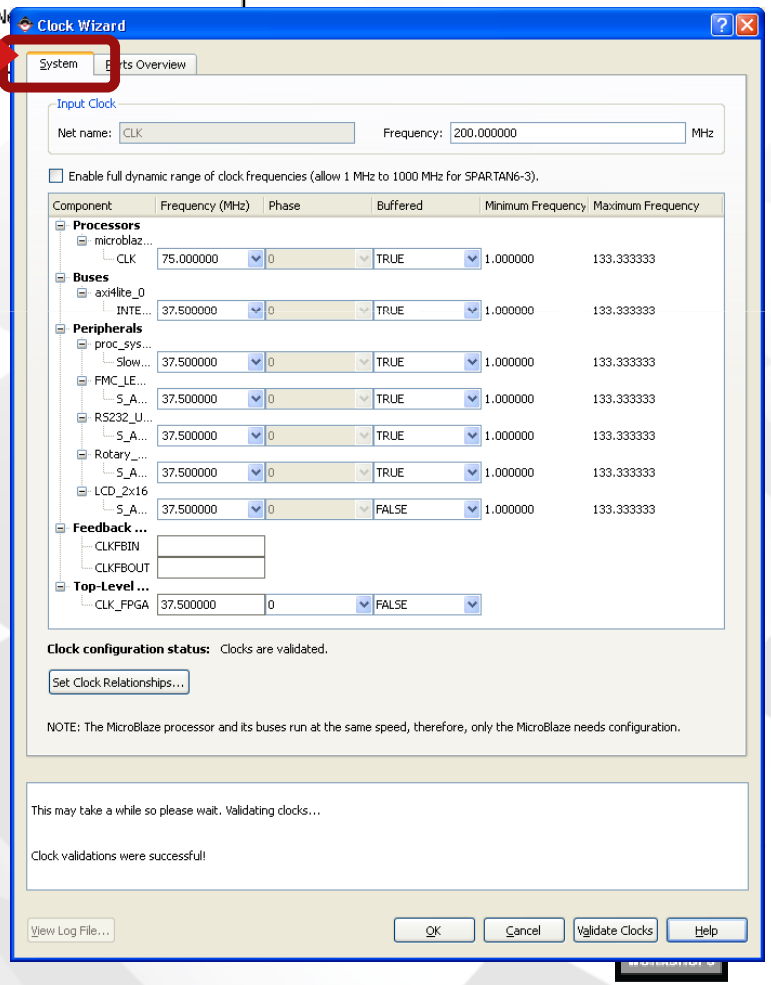
Clock Wizard

- **Use Clock Wizard to configure Clock Generator**



System Tab

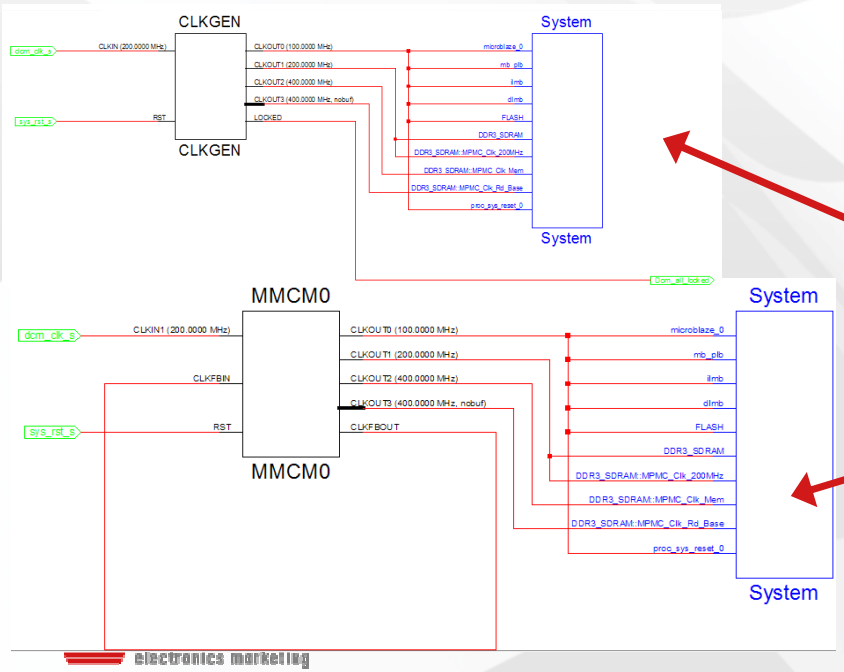
- Use this tab to enter values for clock configuration
- Processors, buses, and peripherals needing clocks are automatically added
- Individual clock signals are automatically identified on a per component basis
- Selectable output frequencies
- Selectable buffer insertion
- Automatic DRC via Validate Clocks button and when OK is selected verifies that selections can be legally generated

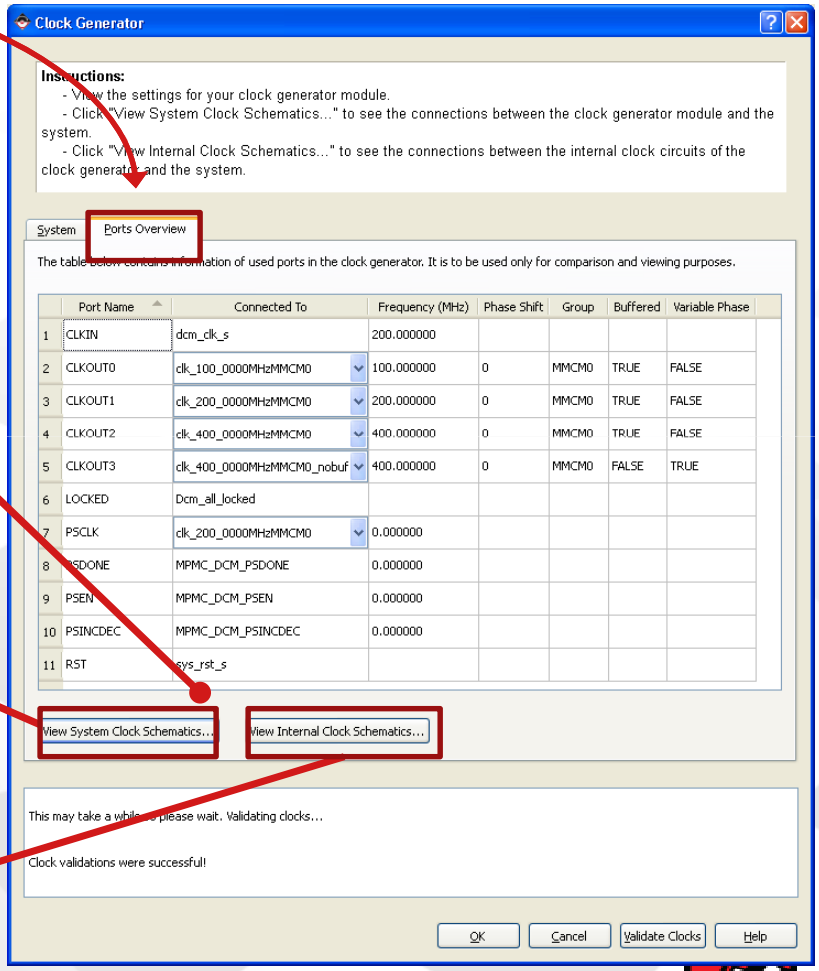


Using the Clocking Wizard

Ports Overview Tab

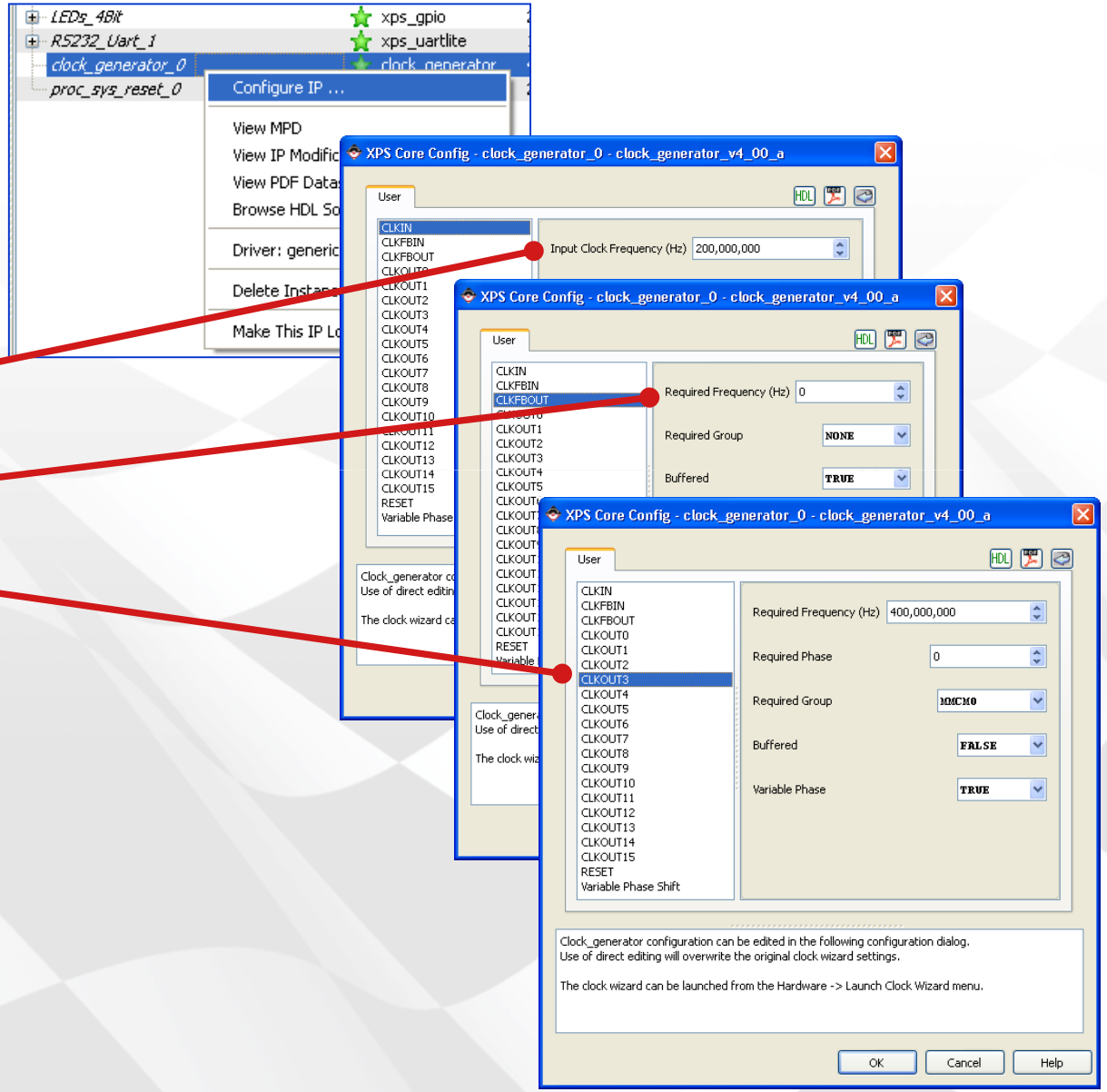
- **Use this tab to view a table of the current clock configuration**
- **Informational only, no editing**
- **Generated clocking component schematic view available**





Using the Clock Generator - Configure IP

- **Accessed via Configure IP to modify the initial Clock Generator settings**
  - Input clock frequency
  - Clock feedback in
  - Clock feedback out
  - Output frequency, phase, and buffering of up to 16 clocks
  - Reset polarity
  - Variable phase shift



JTAG Configurations

JTAG TAP Options

- **At design time, you have control over whether the JTAG TAP of each processor (in the case of multiple processors) is incorporated into the FPGA JTAG TAP chain after FPGA configuration or whether it remains a separate chain**
- **This is accomplished by instantiating or not instantiating the dedicated JTAGPPC block**
- **For the MicroBlaze processor, the MDM (MicroBlaze Debug Module), when instantiated, always attaches to the internal BSCAN component to gain accessibility to the FPGA JTAG pins**
- **When both a MicroBlaze and PowerPC processor are incorporated in the same design, an internal JTAG daisy chain can be automatically created by using and sharing the BSCAN component**

JTAG Configuration/Wakeup

With Combined JTAG Chains

- **Combined JTAG chain designs allow configuration of FPGA, CPU, and external memory all at once**
- **Via JTAG, system configuration can occur as follows**
  - Configure the FPGA
  - Identify processors present via their debug module
  - Configure the CPU, CPU caches, and any external memory that is accessible by the CPU
  - Download code into memory, BRAM or external
  - Execute the code
- **JTAG commands can exist as Serial Vector Format (SVF) in configuration memory to facilitate use with third-party software and download cables**

Checkpoint!

- **What is the purpose of Proc_sys_reset?**

  Coordinates reset behavior between the system reset inputs, processor, bus, and peripheral resets.

- **How does the Clock Generator module work?**

  Determines DCM/PLL configuration, examines the system processors, buses, and peripherals and then builds an appropriate clocking module with the required number of clock outputs. The clock outputs can be configured by the user for frequency, phase, and clock buffer insertion.

- **Name an example of a peripheral whose instantiation is benefited by the presence of the Clock Generator module**

  Memory Controllers as they require multiple phase locked clocks.

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Summary

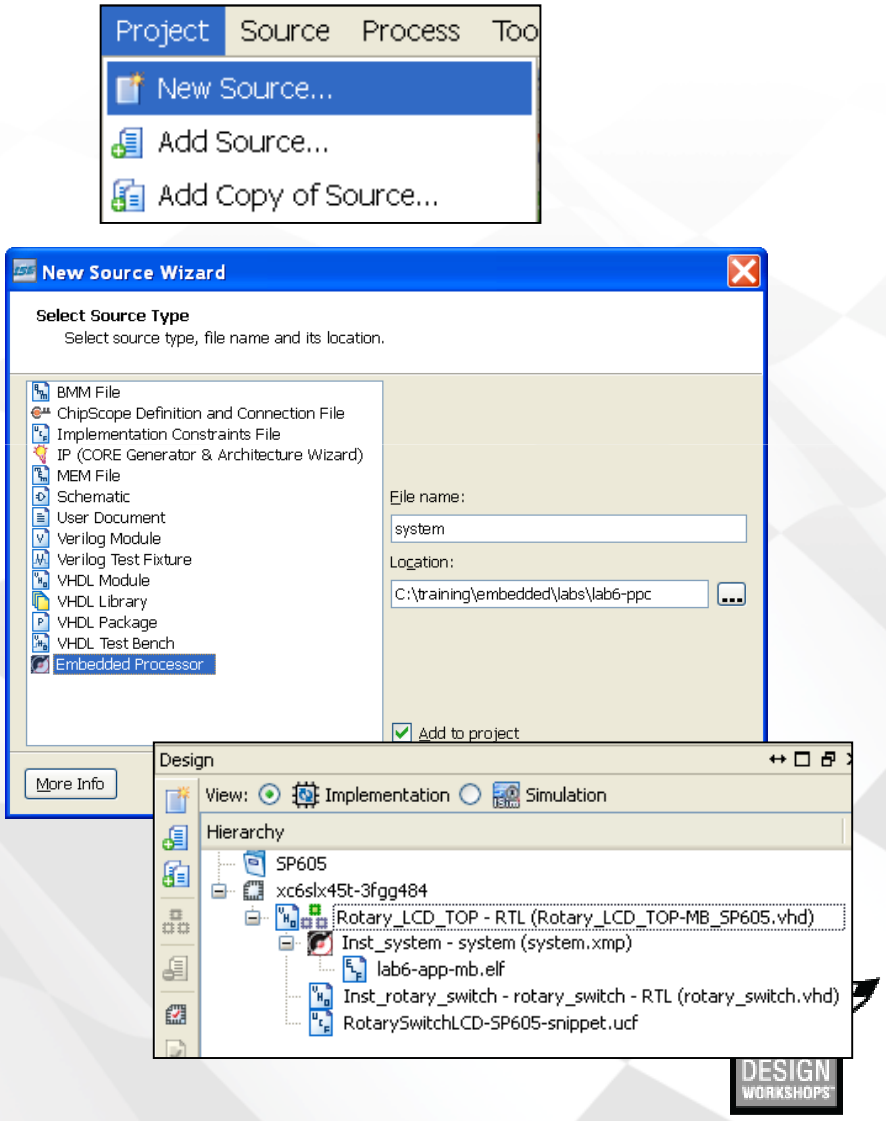
- **The Processor System Reset component provides control over the reset of the entire embedded system**
- **The Clock Generator component automatically inserts configurable clocks for processors, memory controllers, and peripherals.**
- **A combined JTAG chain enables the linking of the FPGA and MDM JTAG chains**
  - This allows the use of the ChipScope Pro logic analyzer and the GDB software debugger

Embedded Development Design Flow

1. Create a new EDK project as a new source in Project Navigator
2. Use the Base System Builder (BSB) to construct your basic embedded design
3. Run PlatGen (Hardware > Generate Netlist) to make your HDL instantiation files and netlist for each component in your embedded design. Alternatively, Export Design to SDK.
4. Close XPS and return to Project Navigator
5. Export processor hardware launch SDK and create a new workspace
6. Create a new software application project in the SDK workspace
7. Compile your software with the GNU cross-compiler using SDK
8. In Project Navigator, add an SDK-created ELF object to the project and generate the programming file
9. Implement the embedded design with the ISE software
10. Download your FPGA's completed bitstream using iMPACT

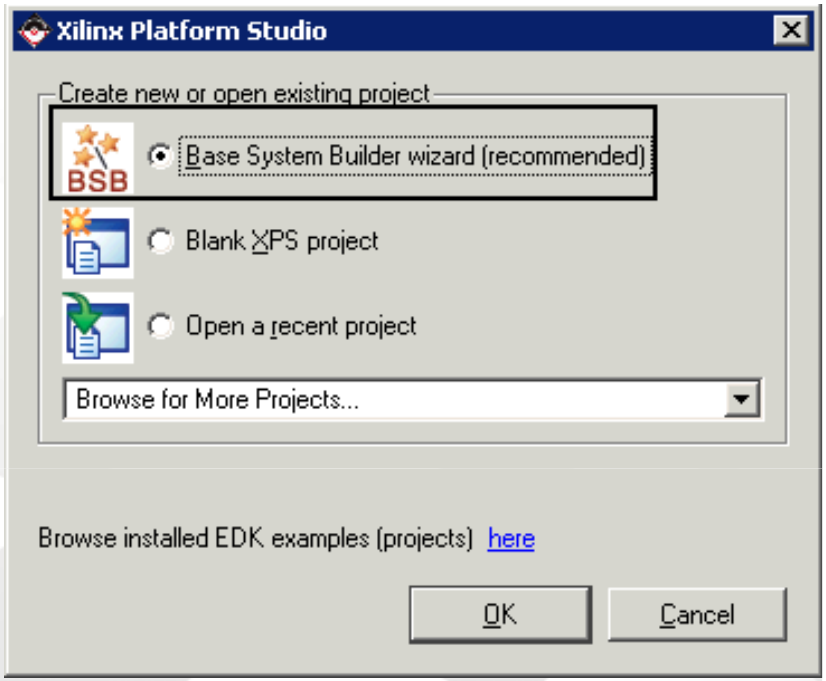
Launching a New XPS Project from the ISE Software – Project Navigator

- **It is recommended that XPS processor projects be launched from Project Navigator in the ISE software suite**
  - Easy to integrate a processor sub-system with other FPGA logic
  - Access to more Xilinx point tools
  - Easy software integration
- **The processor sub-system can be place anywhere in the design hierarchy**



New Project Creation Using the Base System Builder (BSB) Option

- **Select a target board or custom design**
- **Select a processor**
- **Configure the processor**
- **Select and configure board specific interfaces**
- **Add internal peripherals**
- **Generate example software applications**
  - Peripheral and memory tests
- **Generate design files**
  - Generated files include the following
    - *system.mhs*
    - *system.xmp*
    - *data/system.ucf*
    - *pcore directory (empty)*

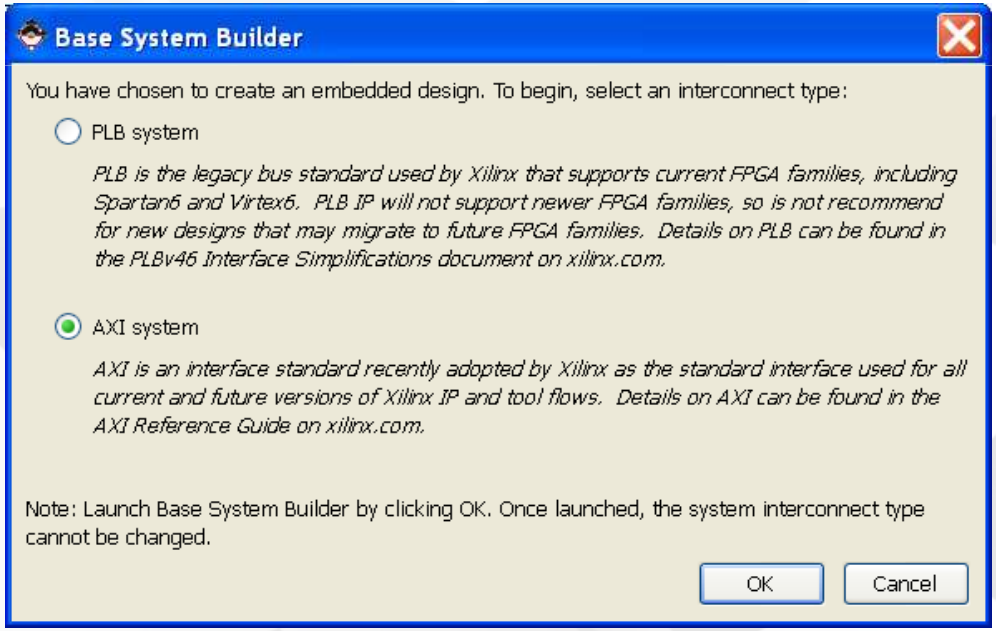


BSB – A simple wizard for creating embedded systems

- **Every processor system requires certain modules that are highly configurable and can be complex to integrate**
- **Processor system module design overhead**
  - Processor
  - Processor bus (AXI4, LMB, AXI4 Streaming)
  - Reset Block
  - Clocking Module
  - Debug Module
- **Base System Builder is an ideal tool to accomplish this design integration**
- **Copying and pasting from an existing and working design MHS file is another alternative**

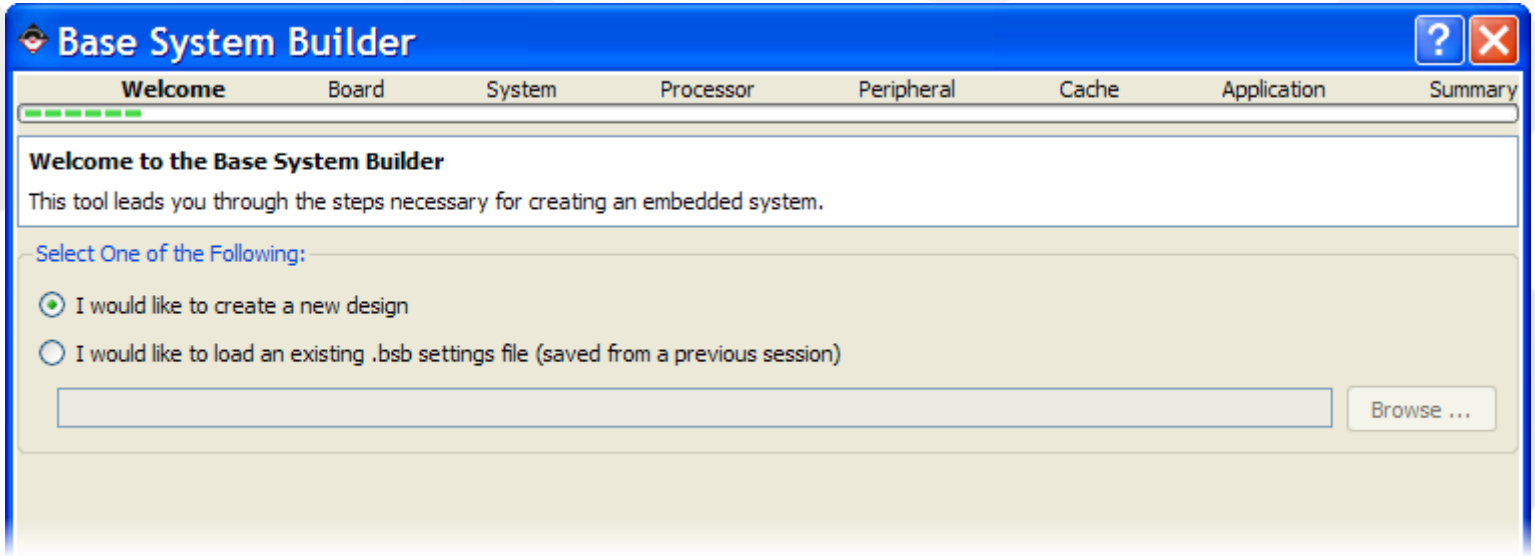
Choose an Interconnect System

- **Traditional PLB v46 or AXI**
- **Will determine the remaining BSB wizard selections**
  - AXI only supported for the MicroBlaze™ processor
  - PLB v46 is traditional support for both the MicroBlaze and PowerPC® processors



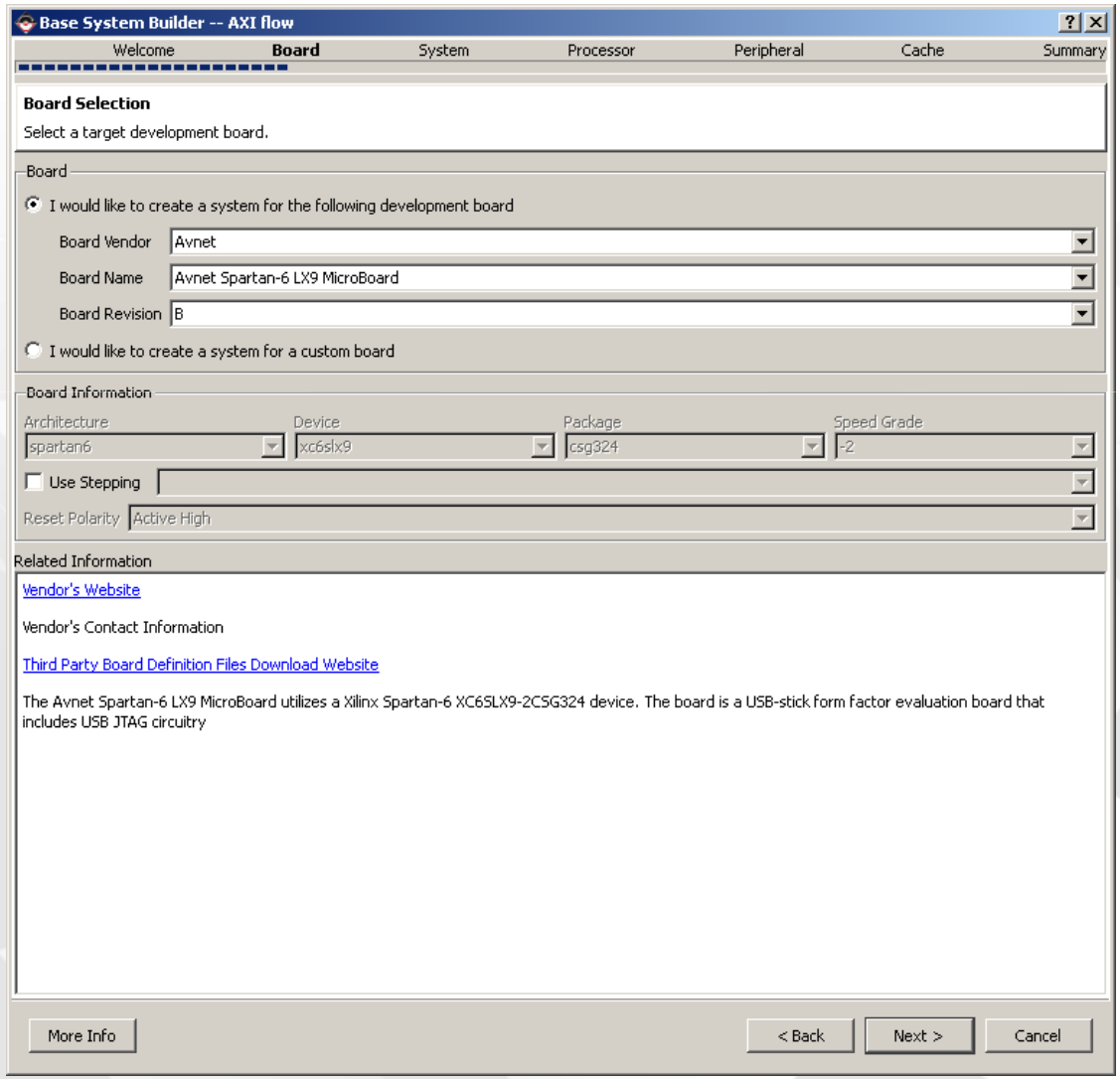
Welcome

- **Note the progress thermometer across the top**
- **You can load an existing settings file (.*bsb* extension)**
  - Useful if you have already customized the BSB file for a custom demo board
  - Check the More Info button for help information about XPS



Selecting a Board

- **Xilinx and its distribution partners sell demo boards with a wide range of added components**
  - This dialog box allows you to quickly learn more about all available demo boards
  - If you want to target a demo from another vendor, their XBD files must first be installed
  - Note that you can also create your own BSB file for a custom board



BSB-Supported Platforms

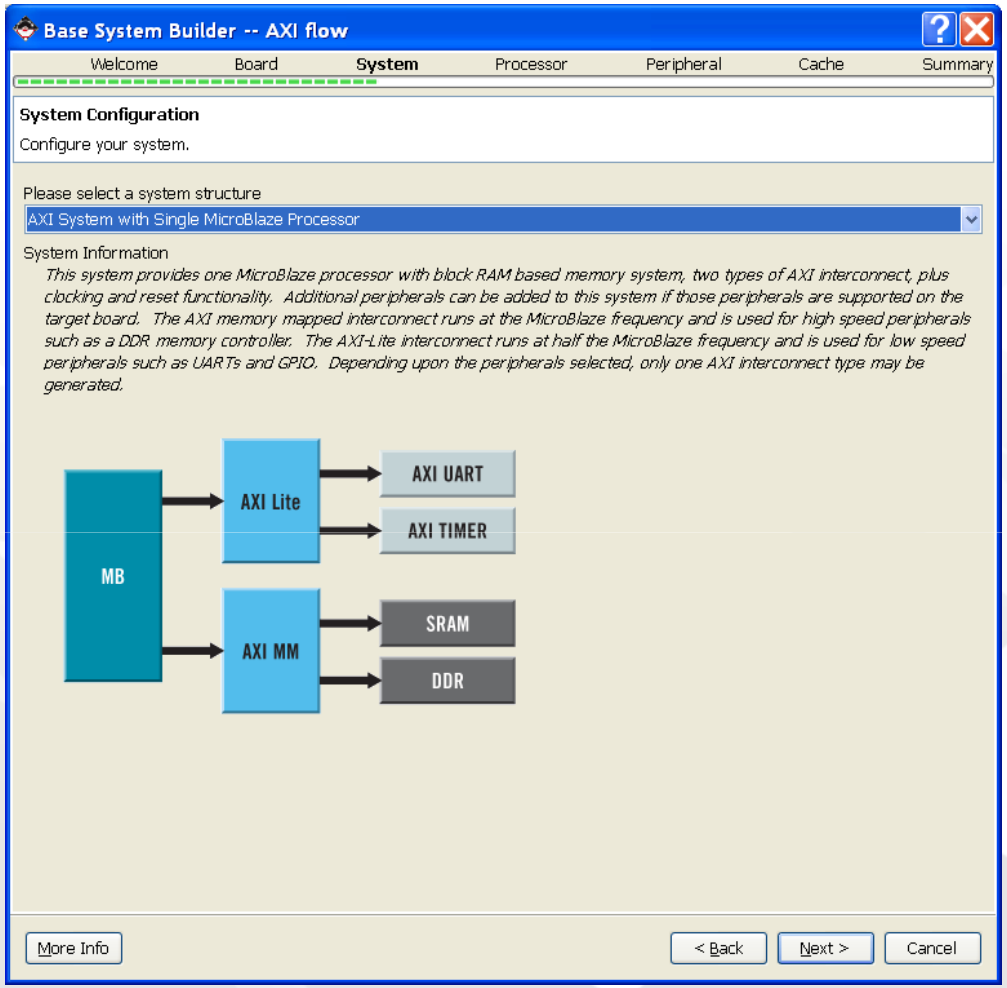
- **AXI-supported Xilinx hardware boards**
  - Spartan-6 FPGA SP601/605 boards
  - Virtex-6 FPGA ML605 boards
  - All Avnet Spartan-6 and Virtex-6 Evaluation Boards
  - Custom board
- **Board definition files are supported in the IPXACT format**
  - New industry standard
- **Board definition files (XBD, XBD2(IPXACT)) for third-party boards can be downloaded from the website of the board vendor**
  - Links from the BSB wizard and Xilinx embedded Web page

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

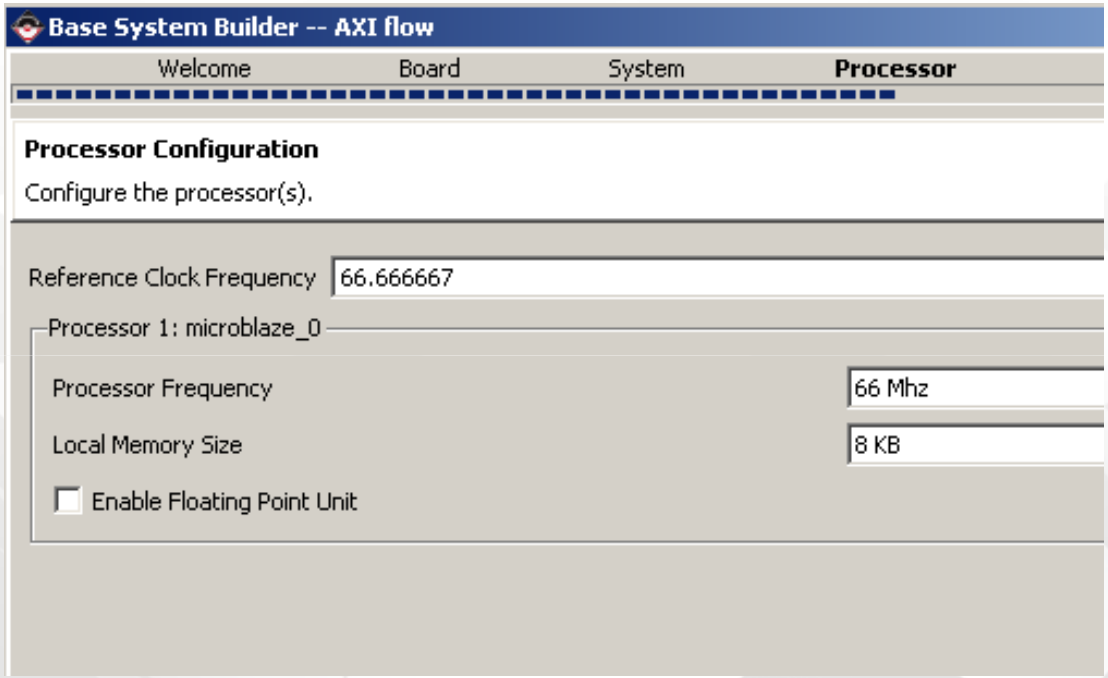
Selecting Single or Dual Processors

- **BSB AXI-based systems presently support only a single processor-based system**
- **More processors can be added later in the System Assembly View**



Configuring the Processor – Reference Clock

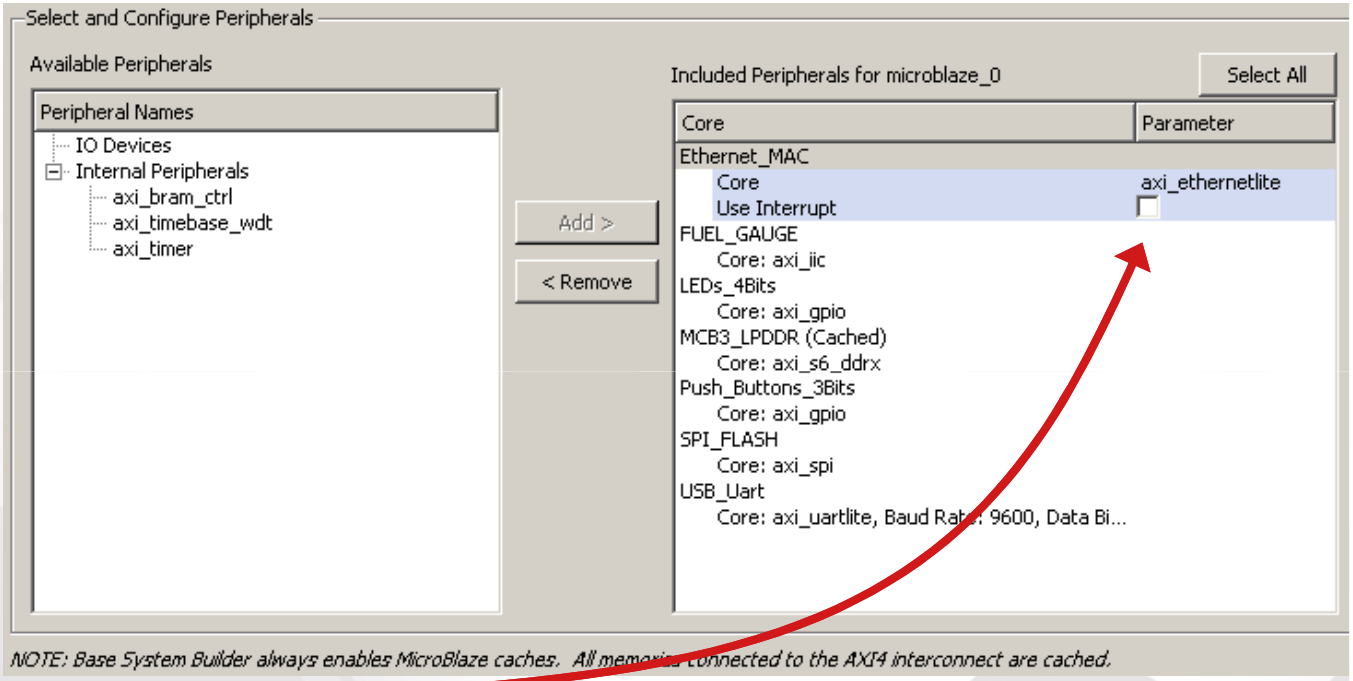
- **Reference clock is board oscillator frequency**
- **Processor clock frequency is the clock rate connected directly to the processor**
- **Bus clock frequency is the clock rate of all bus peripherals in the system**
- **These selections will automatically customize the clock generator module**
- **The appropriate debug interface is added automatically**



Configuring the I/O Interfaces

- **Choose the peripherals you need from those available for your demo board**

- **Peripherals can be added or removed**
- **Most peripherals are customizable via drop-down lists when selected**
- **Many peripherals support the use of interrupts**



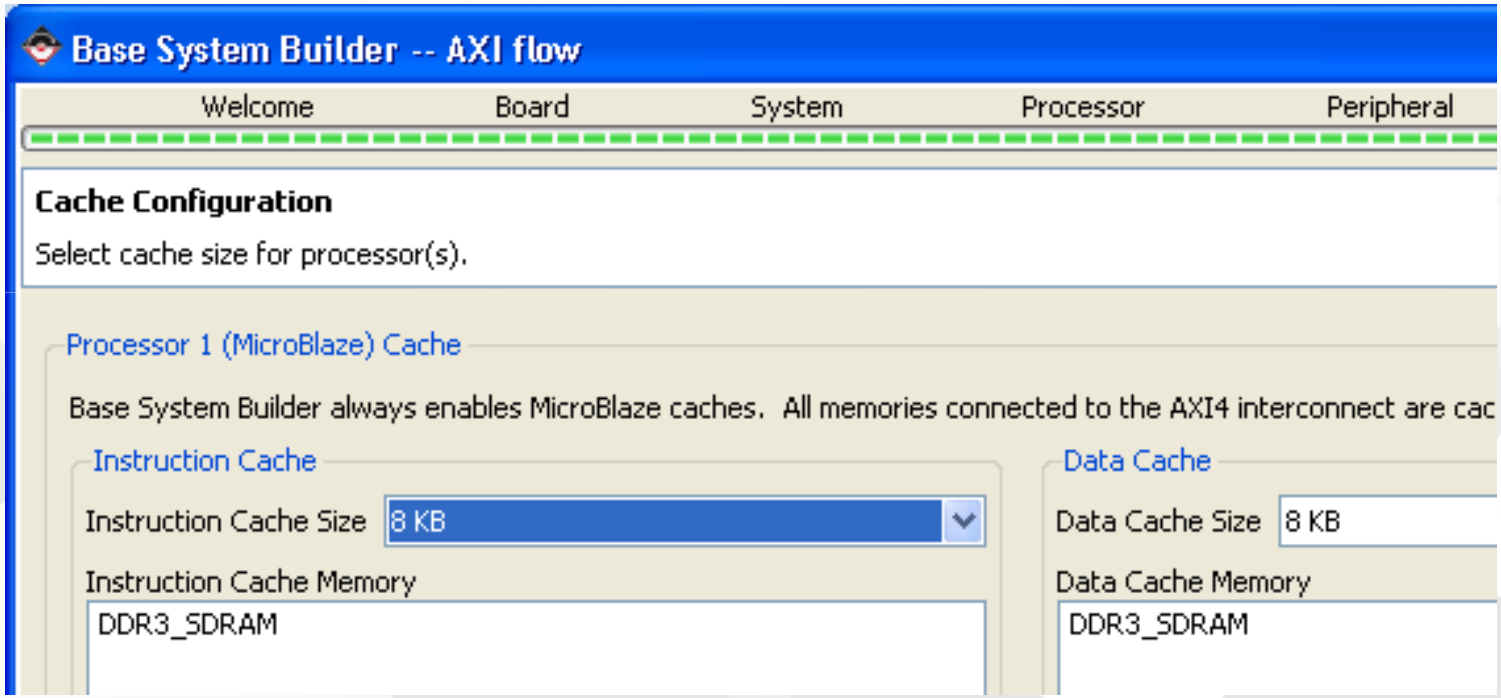
- **Internal peripherals exist for all board hardware configurations**

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

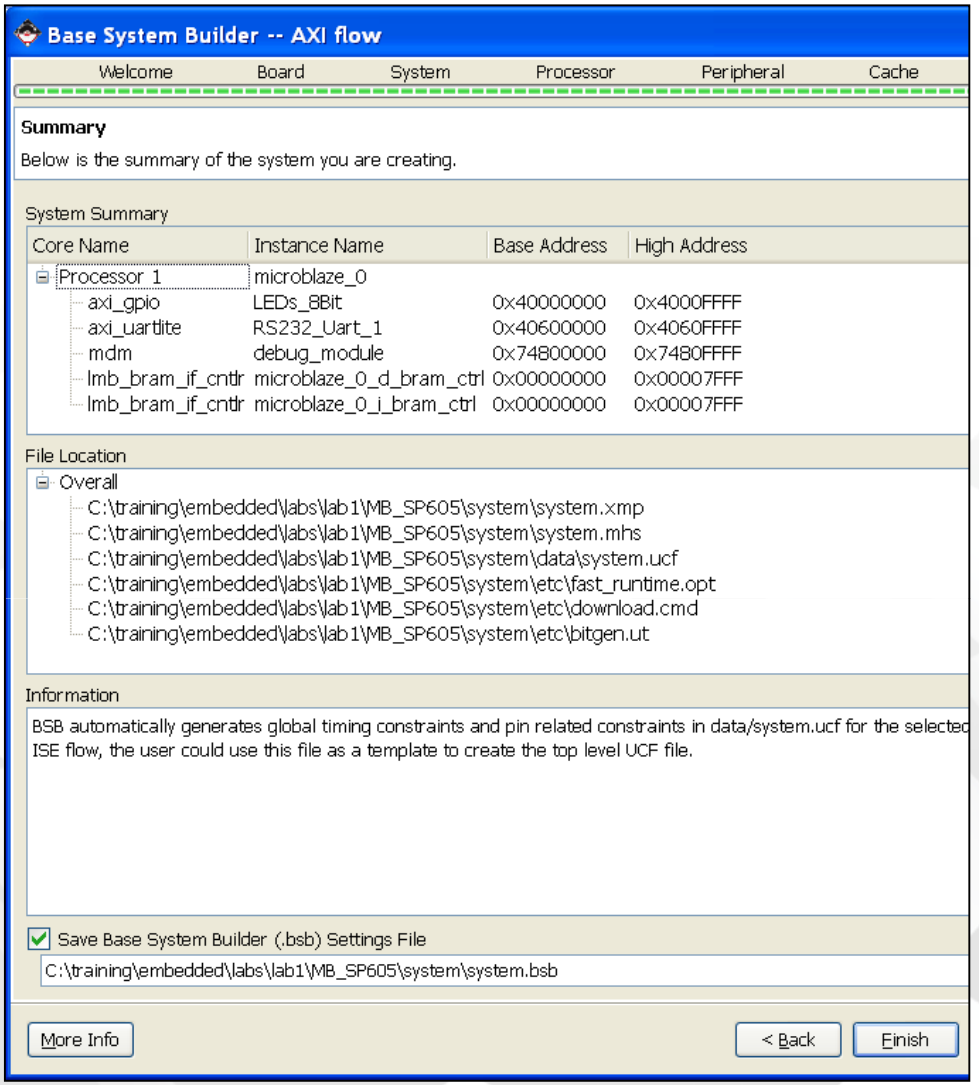
Cache Configuration

- **Cache is automatically enabled for off FPGA memory**
- **Cache size is selectable**



Creating the System

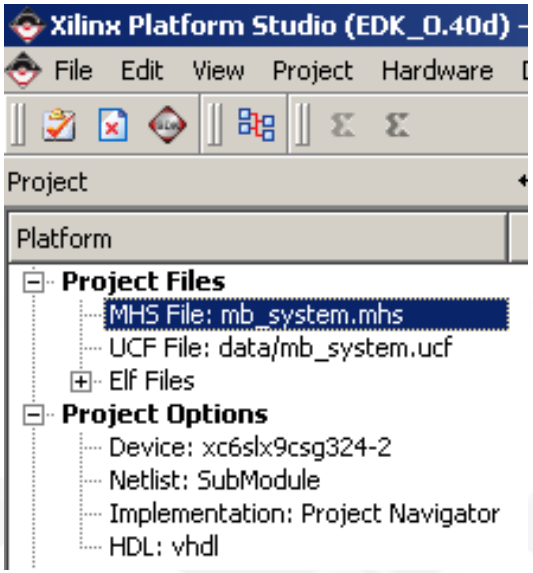
- **BSB creates a MHS File for your system**
- **The MHS references every processor, bus, peripheral, and memory in your embedded system**
  - It contains connection information and parameter settings for every peripheral
- **Each peripheral has a default parameter setting stored in an MPD file**
  - MHS settings override MPD default settings
- **Memory mapping information follows the nature of the chosen processor for your system**



AVNET electronics marketing

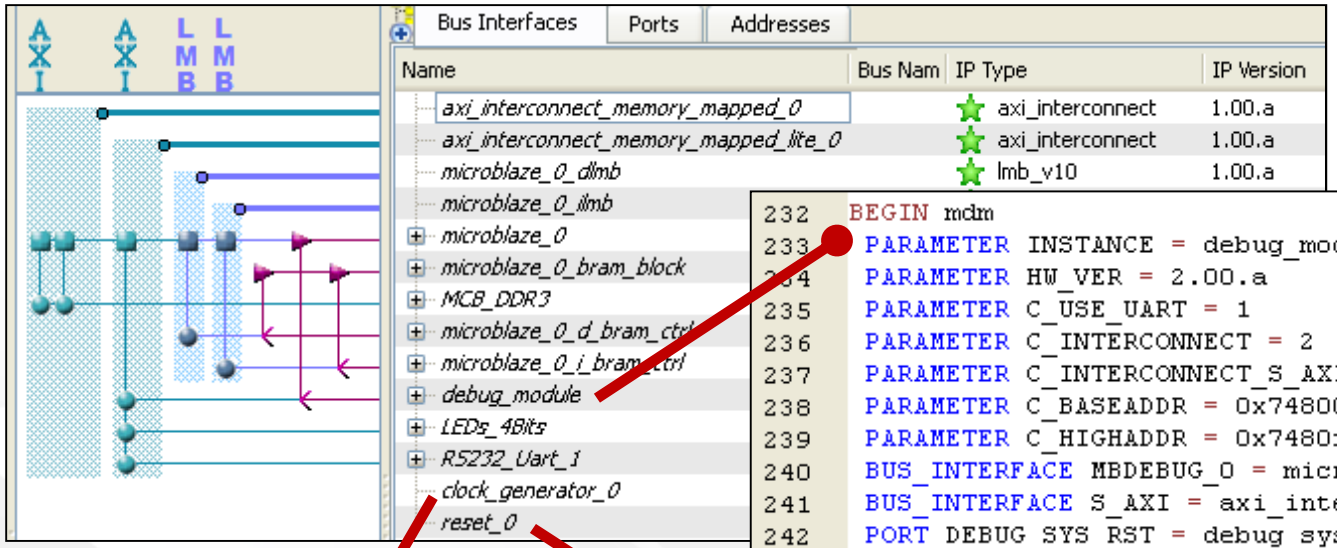
SPEEDWAY DESIGN WORKSHOPS

MHS File



- **MicroBlaze Hardware Specification**
- **Hardware netlist of the processor sub-system**
  - Main source file of the processor sub-system
  - Component instances are from IP Catalog
  - Shows component instances and their connectivity
  - Stored as a text file
- **Named after the project, *<project>.mhs***
- **Fully defines the embedded system hardware**
  - Does not include tool setting options (these are part of the ISE Project file, assuming ISE is top level project.)
- **The System Assembly View is the IDE that builds the MHS file**
  - Hand editing is also allowed
- **XPS will not open a project whose XPS file has an error**
  - Usually happens when hand editing mistakes are made
  - May require fixing in Notepad, because XPS will not re-open

Leveraging BSB for the MicroBlaze Processor



MHS File

```
232  BEGIN mdm
233   PARAMETER INSTANCE = debug_module
234   PARAMETER HW_VER = 2.00.a
235   PARAMETER C_USE_UART = 1
236   PARAMETER C_INTERCONNECT = 2
237   PARAMETER C_INTERCONNECT_S_AXI_MASTERS = microblaze_0.M_AXI_DP
238   PARAMETER C_BASEADDR = 0x74800000
239   PARAMETER C_HIGHADDR = 0x7480ffff
240   BUS_INTERFACE MBDEBUG_0 = microblaze_0_debug
241   BUS_INTERFACE S_AXI = axi_interconnect_memory_mapped_lite_0
242   PORT DEBUG_SYS_RST = debug_sys_rst
243   PORT S_AXI_ACLK = clk_37_5000MHzPLL0
244  END
```

```
123  BEGIN proc_sys_reset
124   PARAMETER INSTANCE = reset_0
125   PARAMETER HW_VER = 3.00.a
126   PARAMETER C_EXT_RESET_HIGH = 1
127   PORT Ext_Reset_In = RESET
128   PORT MB_Reset = microblaze_0_Reset_reset_0_Reset_0_adhoc
129   PORT Slowest_sync_clk = clk_37_5000MHzPLL0
130   PORT Interconnect_aresetn = connectnetwork_0_reset_reset_0_Reset_3_adhoc
131   PORT Dcm_locked = reset_0_Lock_clock_0_Lock_adhoc
132   PORT MB_DEBUG_SYS_RST = debug_sys_rst
133   PORT BUS_STRUCT_RESET = sys_bus_reset
134  END
```

```
136  BEGIN clock_generator
137   PARAMETER INSTANCE = clock_generator_0
138   PARAMETER HW_VER = 4.00.a
139   PARAMETER C_CLKIN_FREQ = 200000000
140   PARAMETER C_CLKOUT0_BUF = FALSE
141   PARAMETER C_CLKOUT0_FREQ = 600000000
142   PARAMETER C_CLKOUT0_GROUP = PLL0
143   PARAMETER C_CLKOUT1_BUF = FALSE
144   PARAMETER C_CLKOUT1_FREQ = 600000000
145   PARAMETER C_CLKOUT1_GROUP = PLL0
146   PARAMETER C_CLKOUT1_PHASE = 180
147   PARAMETER C_CLKOUT2_FREQ = 37500000
148   PARAMETER C_CLKOUT2_GROUP = PLL0
149   PARAMETER C_CLKOUT3_FREQ = 75000000
150   PARAMETER C_CLKOUT3_GROUP = PLL0
151   PORT RST = RESET
152   PORT CLKIN = CLK
153   PORT LOCKED = reset_0_Lock_clock_0_Lock_adhoc
154   PORT CLKOUT0 = clk_600_0000MHzPLL0_nobuf
155   PORT CLKOUT1 = clk_600_0000MHz180PLL0_nobuf
156   PORT CLKOUT2 = clk_37_5000MHzPLL0
157   PORT CLKOUT3 = clk_75_0000MHzPLL0
158  END
```

SPEEDWAY DESIGN WORKSHOPS

MHS File Quiz

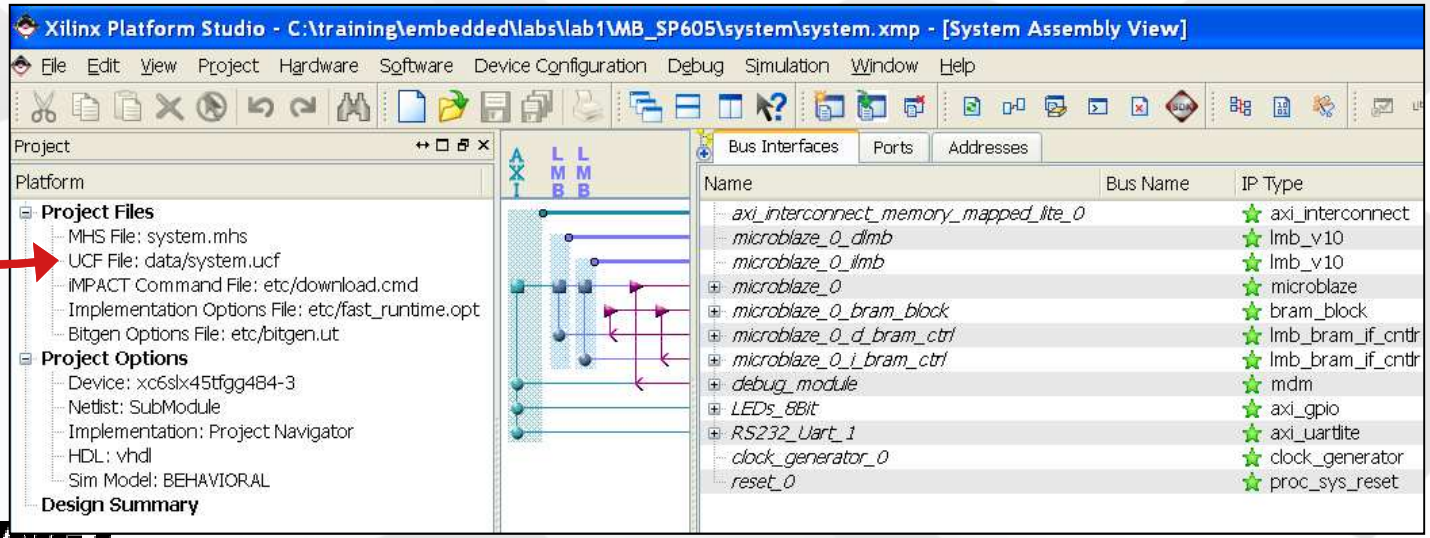
- To the right is a snippet of the MHS file that was generated by BSB
- What do the MHS snippets instantiate?
- Can you identify the addresses of the components?
- What is the difference between parameters and ports?
- Be careful when editing the MHS
  - Any mistakes will be reflected in the GUI
    - Some mistakes will not allow you to re-open the **System Assembly View**
    - Must use text editor to fix

```
BEGIN axi_uartlite
 PARAMETER INSTANCE = RS232_Uart_1
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BAUDRATE = 9600
 PARAMETER C_DATA_BITS = 8
 PARAMETER C_USE_PARITY = 0
 PARAMETER C_ODD_PARITY = 1
 PARAMETER C_BASEADDR = 0x40600000
 PARAMETER C_HIGHADDR = 0x4060ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT TX = RS232_Uart_1_sout
 PORT RX = RS232_Uart_1_sin
 PORT S_AXI_ACLK = clk_33_3333MHzPLL0
END

BEGIN axi_gpio
 PARAMETER INSTANCE = LEDs_4Bits
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_GPIO_WIDTH = 4
 PARAMETER C_ALL_INPUTS = 0
 PARAMETER C_INTERRUPT_PRESENT = 0
 PARAMETER C_IS_DUAL = 0
 PARAMETER C_BASEADDR = 0x40000000
 PARAMETER C_HIGHADDR = 0x4000ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT GPIO_IO_O = LEDs_4Bits_TRI_O
 PORT S_AXI_ACLK = clk_33_3333MHzPLL0
END
```

Continuing with XPS

- **The BSB creates a completed design of your system as specified**
- **A User Constraint File (UCF) is created that contains**
  - Pin assignments
    - For any MicroBlaze peripherals that require external FPGA I/O
  - Timing constraints
    - Based on the hardware board selected
    - Based on the clock frequency specifications
- **After completing BSB, start using XPS to add more components or modify the design and build the hardware netlist**



UCF File

- A UCF file has been generated by BSB, based on your peripheral selections
- The UCF mainly consists of location, timing, and area constraints
- It will not be modified when deleting or adding components to the design
  - **Changes must be done by hand**
- Why has the TIG been placed?
- What pin assignments or attributes do you see here?

```
## System level constraints

Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;

Net sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 10000 ps;
Net sys_rst_pin TIG;

#### Module LEDs_8Bit constraints

Net fpga_0_LEDs_8Bit_GPIO_IO_pin<0> LOC = AE24;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<0> IOSTANDARD=LVCMOS18;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<0> PULLDOWN;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<0> SLEW=SLOW;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<0> DRIVE=2;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<1> LOC = AD24;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<1> IOSTANDARD=LVCMOS18;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<1> PULLDOWN;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<1> SLEW=SLOW;
Net fpga_0_LEDs_8Bit_GPIO_IO_pin<1> DRIVE=2;
```

Checkpoint!

- **Name some of the components that are common to every design.**

  Processor, Reset block, Clocking module, Debug module, AXI (or PLB) bus

- **What is an MHS File?**

  Microprocessor Hardware Specification file
  Hardware netlist of the embedded processor system
  Text format and is typically edited graphically in XPS

- **When does a BSB UCF file need to be updated?**

  UCF file is generated once when BSB finishes a design for a specified hardware development board
  If peripherals with external FPGA pins are added or deleted to or from the design, the UCF file must be updated manually.
  Also, if any of the clock module timing is changed, this may require a modification to related timing constraints in the UCF.

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Summary

- **Creating processor designs is easy with Base System Builder**
- **Modifying a BSB-created hardware design is easily done with XPS**
- **MHS File defines hardware platform of processing system**
- **The System Assembly view is a graphical editor of the MHS file**

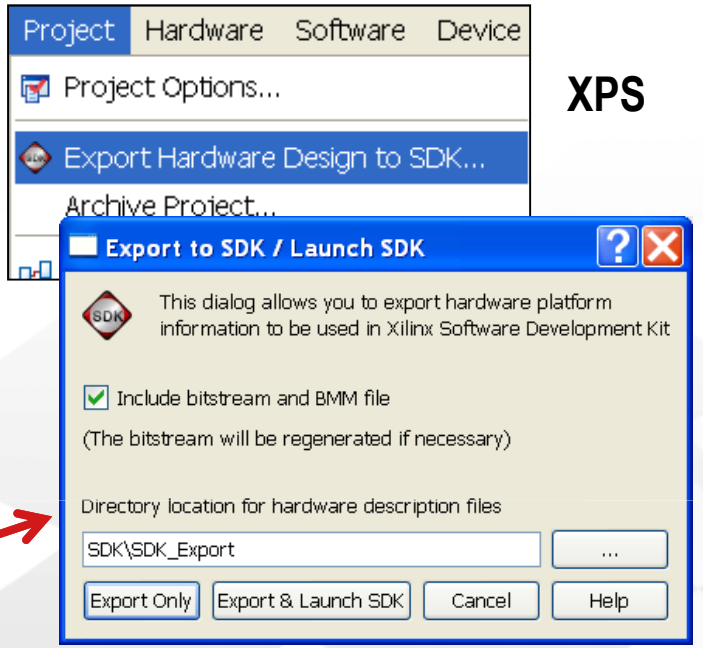
Software Development Kit

AVNET electronics marketing

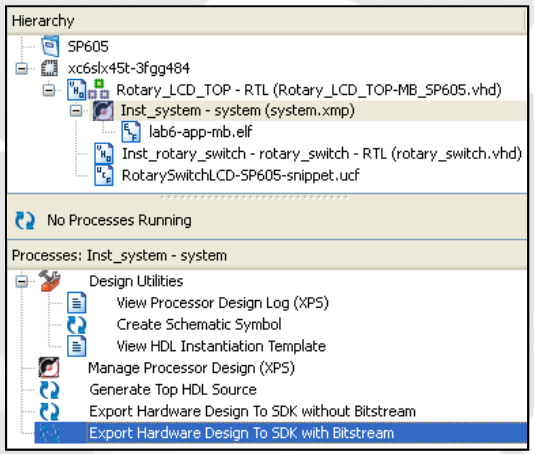
SPEEDWAY DESIGN WORKSHOPS

Software Flow - Exporting Hardware Platform to SDK

- **Software development is performed with the Xilinx Software Development Kit (SDK)**
- **A hardware image XML file, generated by XPS, defines the hardware platform for which the software application will be developed**
  - XML exported from XPS or Project Navigator
- **The SDK software tools will then associate the software project to the hardware**



XPS



Project Navigator

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

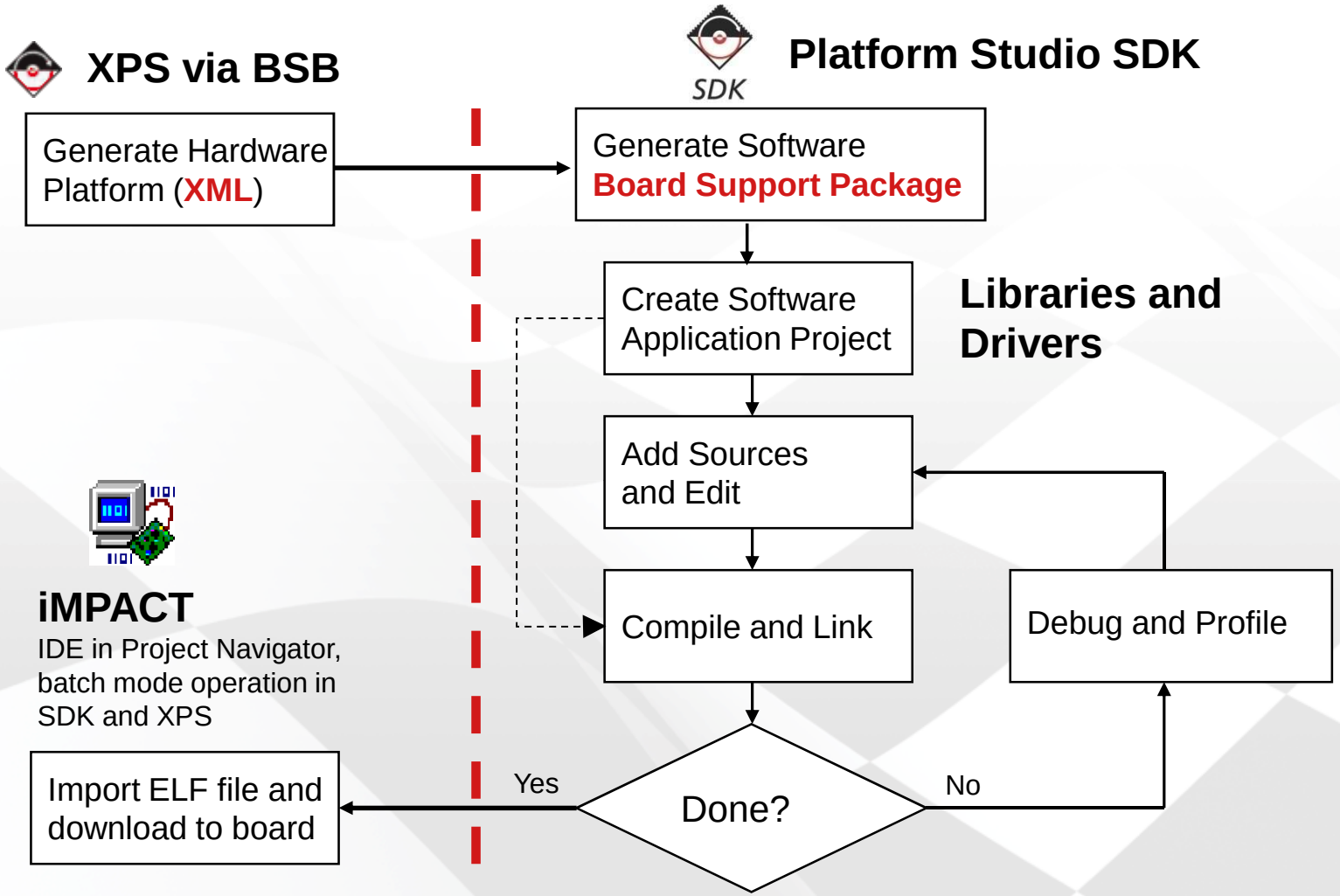
Software Development Kit (SDK)

- **Full-featured software design environment**
- **Separate tool from Project Navigator and XPS**
- **Base on popular Eclipse open-source IDE**
- **Used for software applications only; hardware design and modifications are done in XPS**
- **Well-integrated environment for seamless debugging of embedded targets**
- **Sophisticated software design environment with many options and features with support for**
  - Multiple processor platforms
  - Multiple software Board Support Packages
  - Multiple software applications
- **Superior C/C++ code editor and error navigator**

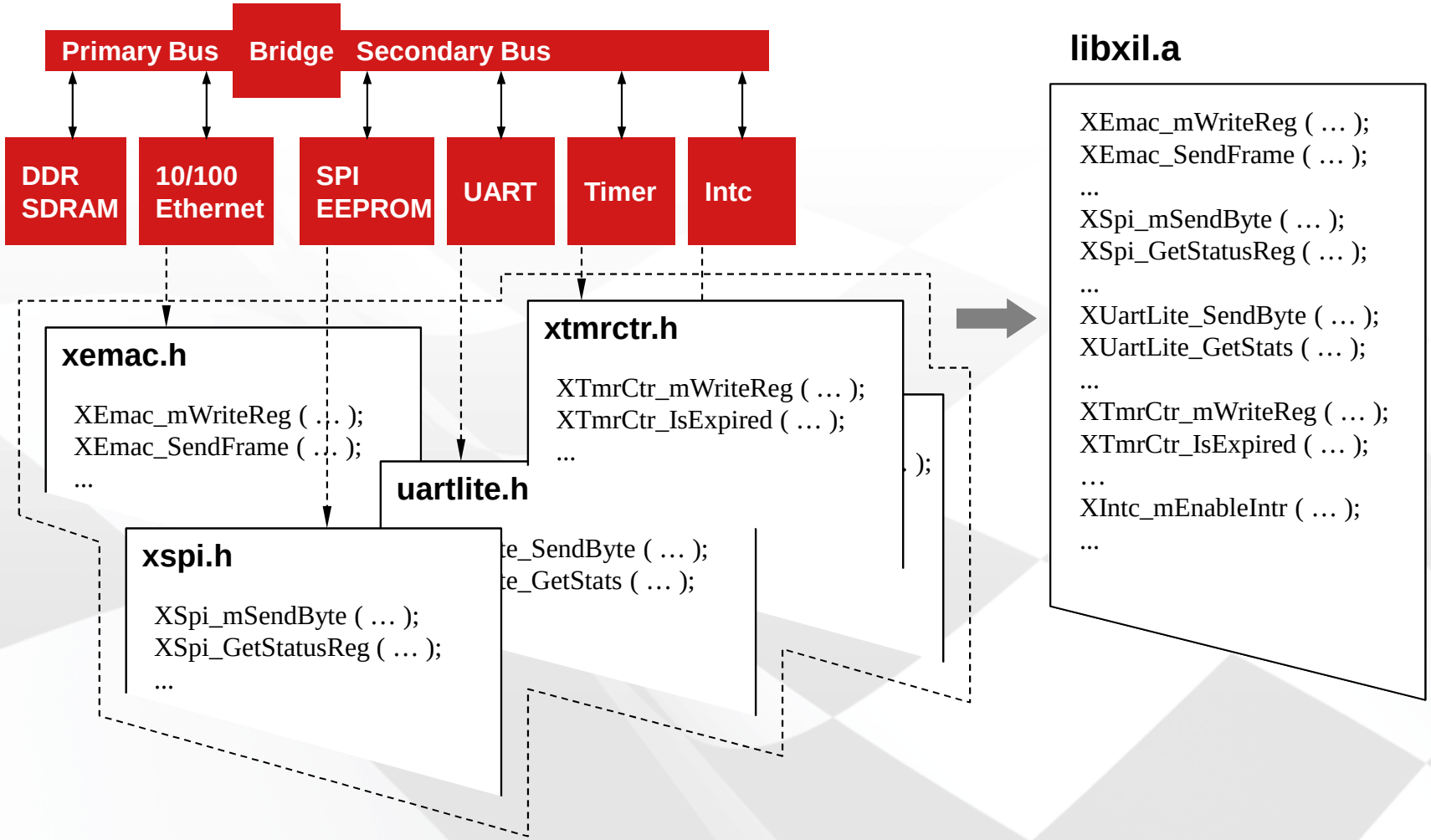
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

SDK Application Development Flow



Board Support Package



Board Support Packages (BSPs) are collections of parameterized drivers for a specific processor system

SDK Workbench Views

C/C++ Perspective

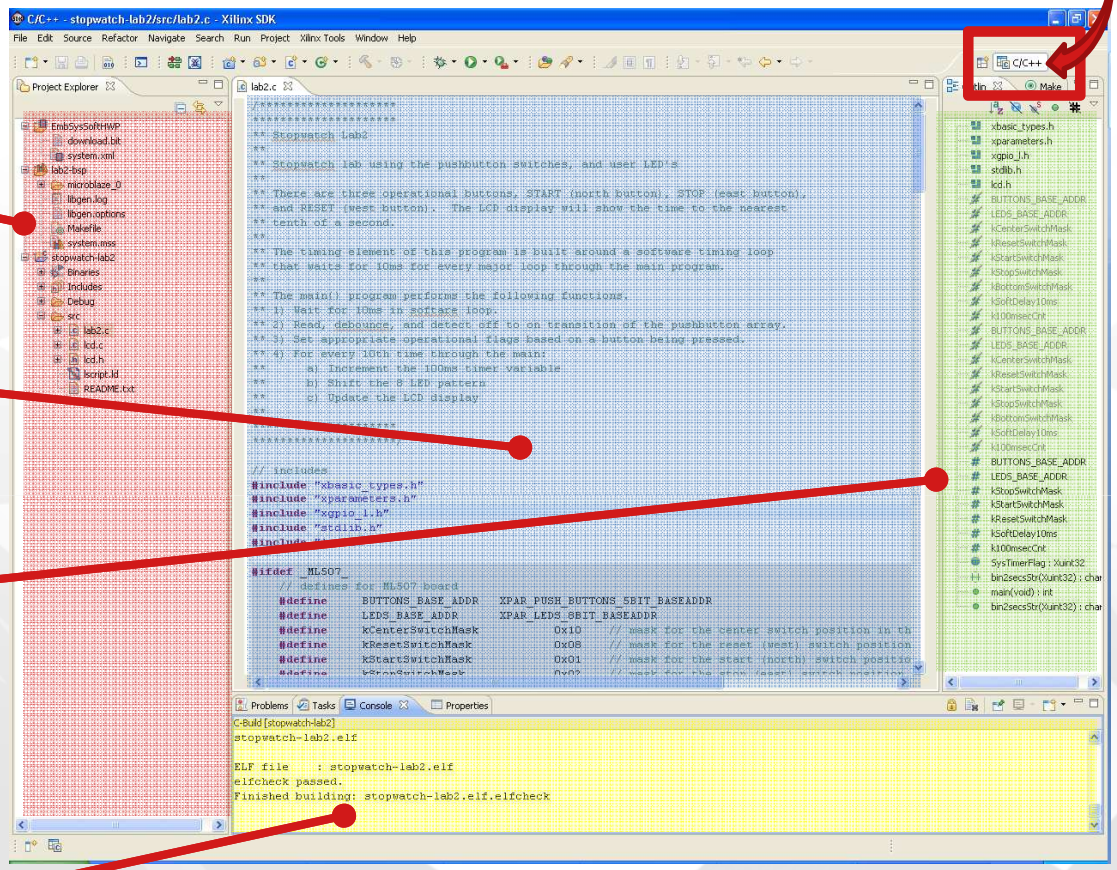
C/C++ project outline displays the elements of a project with file decorators (icons) for easy identification

C/C++ editor for integrated software creation

Code outline displays elements of the software file under development with file decorators (icons) for easy identification

Problems, Console, Properties views list output information associated with the software development flow

Active Perspective indicated here



AVNET electronics marketing

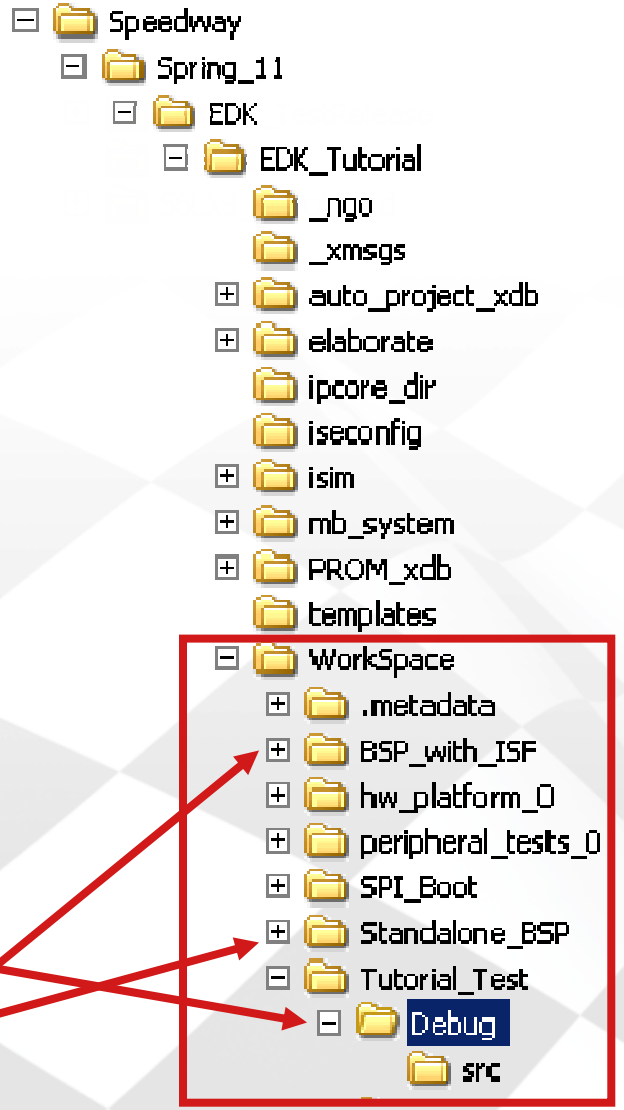
SPEEDWAY DESIGN WORKSHOPS

Building the Software Application

- **The software toolchain consists of the following GNU tools**
  - Pre-processor (optional)
  - Compiler
  - Assembler
  - Linker
- **A software build executes all of the tools in sequence and produces an ELF file**
- **A software build happens when**
  - Saving a resource file
  - Selecting **Project** > **Build All**

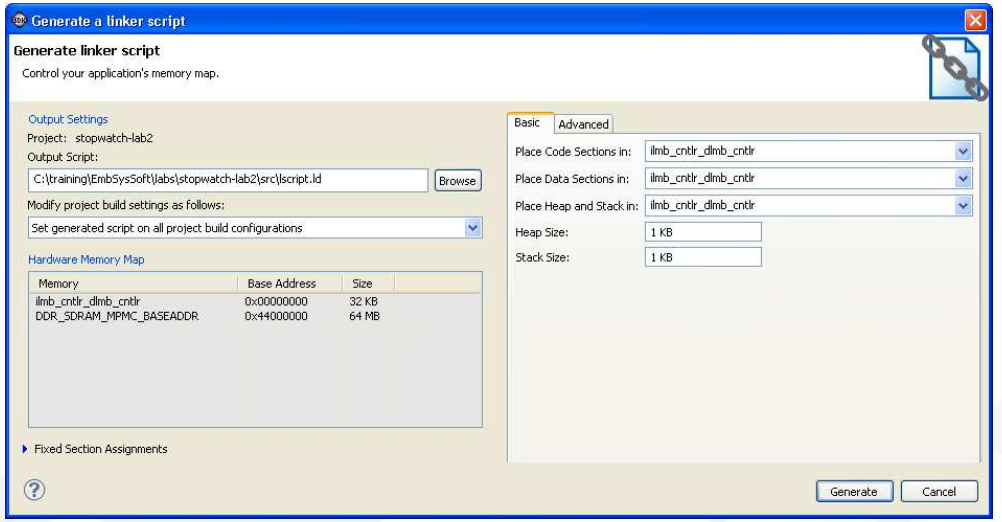
Directory Structure

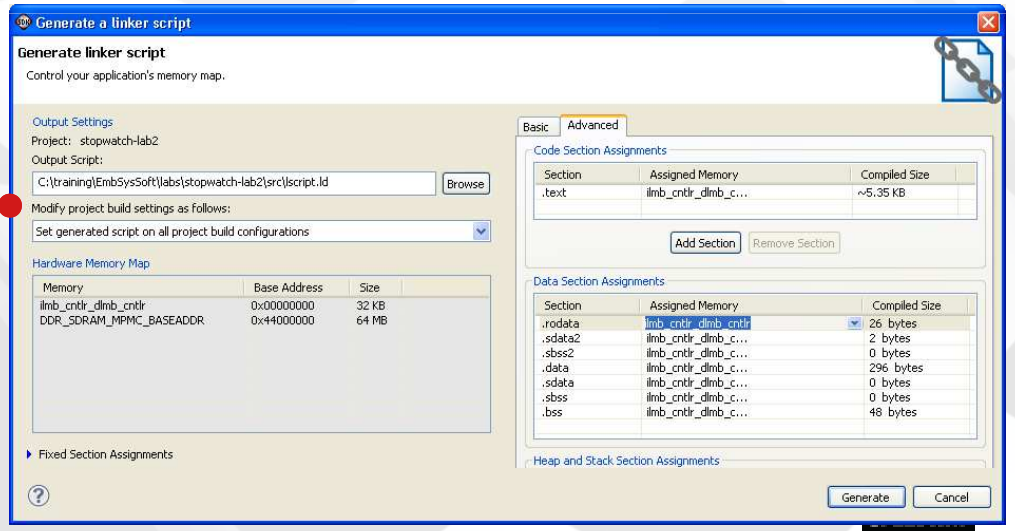
- **SDK projects are place in the application directory that was specified when SDK was launched – WorkSpace is commonly used.**
- **Each project will have multiple directories for system files and configurations**
- **Configurations are property tool option permutations of the software application.**
- **Each configuration has project properties set depending on needs. An ELF file is generated for each.**
  - Release configuration
  - Debug configuration
  - Profile configuration
- **A Debug configuration is created by default**
- **Board Support Packages associated with each software application**



Software Application Linker Script

- **A linker script is necessary for every software application**
- **Dictates where software is to be placed in memory**
- **Select Xilinx Tools > Generate linker script**
- **Use default location for linker script file for automatic insertion as linker option**

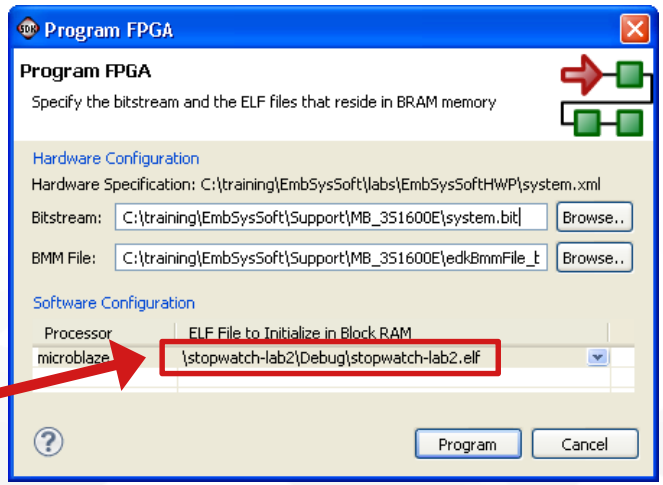




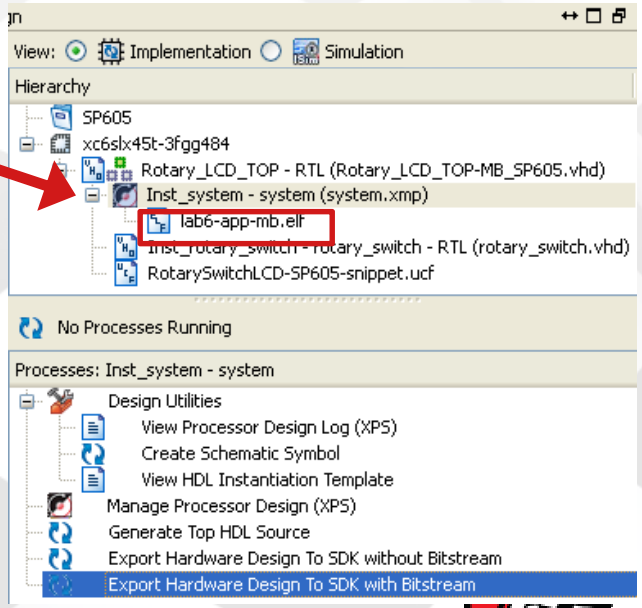
Merging Hardware and Software Flows

- **Software Application needs to be merged with Hardware so MicroBlaze has code to execute upon bootup.**
  - Only for application software that resides in block RAM
- **Can be done in SDK or ISE**
  - In ISE, add the software ELF object file as a project source
- **Create a bitstream file**
  - Generate bitstream
  - Data2MEM is automatically executed as part of the Generate Bitstream command
  - Software application is now pre-loaded into the block RAM

SDK

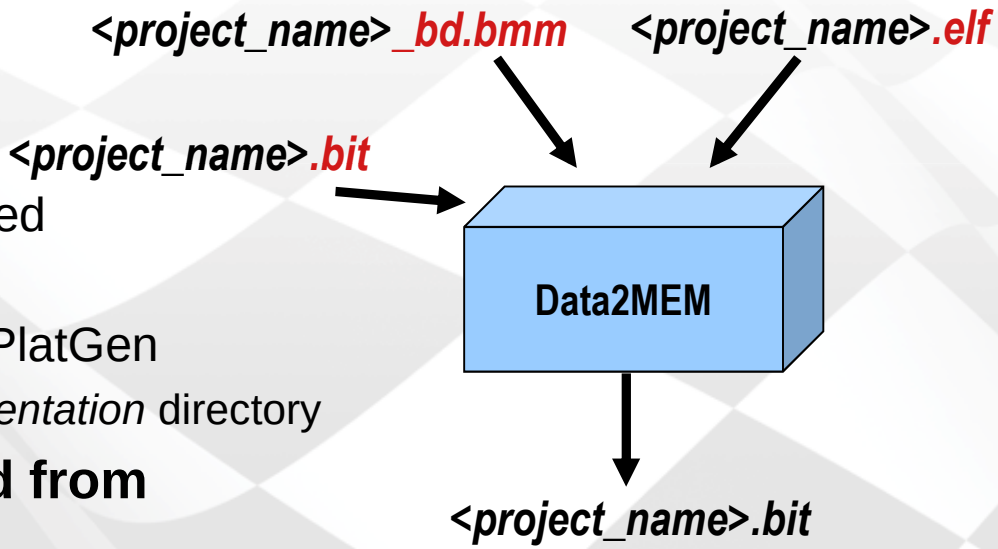


ISE Tool



Data2Mem - Memory Initialization

- **Block RAM memory must be initialized with program code so that the processor has something to execute after FPGA configuration**
- **Each bit of FPGA block RAM is represented as a bit in the bit file, you need**
  - The compiled executable
    - *executable.elf*
  - Hardware design implemented
    - *<project_name>.bit*
  - The BMM file generated by PlatGen
    - *<project_directory>/implementation* directory
- **Data2MEM can be launched from**
  - Project Navigator
  - SDK
  - XPS



Checkpoint!

- **What file is required from XPS to load project into SDK? How is it created?**

XML File
Must be exported from XPS tool

- **What does the Linker Script do?**

Dictates where software applications reside in memory space

- **What tool merges software application with hardware bitstream? How is it invoked?**

DATA2MEM
ISE – Add ELF file to Project, Generate Bitstream
SDK – Program FPGA, select ELF File to initialize BRAM

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Summary

- **SDK is a comprehensive software development environment for simple software and firmware and for complex applications**
- **Based on the Eclipse open-source IDE**
  - WorkSpace
  - Projects
  - Perspectives
  - Views
- **Full tool and environment option control**
- **Multiple project and processor support**
- **Easy access to BSP and linker script generation tools**

Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - **Lab 1- Adding a Processor to a ISE Design**
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

Lab Instructions

- **Labs are self guided, you will find accompanying step-by-step directions and illustrated figures that provide detail for performing the lab.**
- **Labs build on one-another. So labs must be completed before starting next lab.**

- **Directory structure is under**

  C:\SpeedWay\Spring_11\EDK

  - *EDK_Tutorial* – your work space
  - *Solutions* – files containing the completed lab

Lab 1: Hardware Construction with the BSB

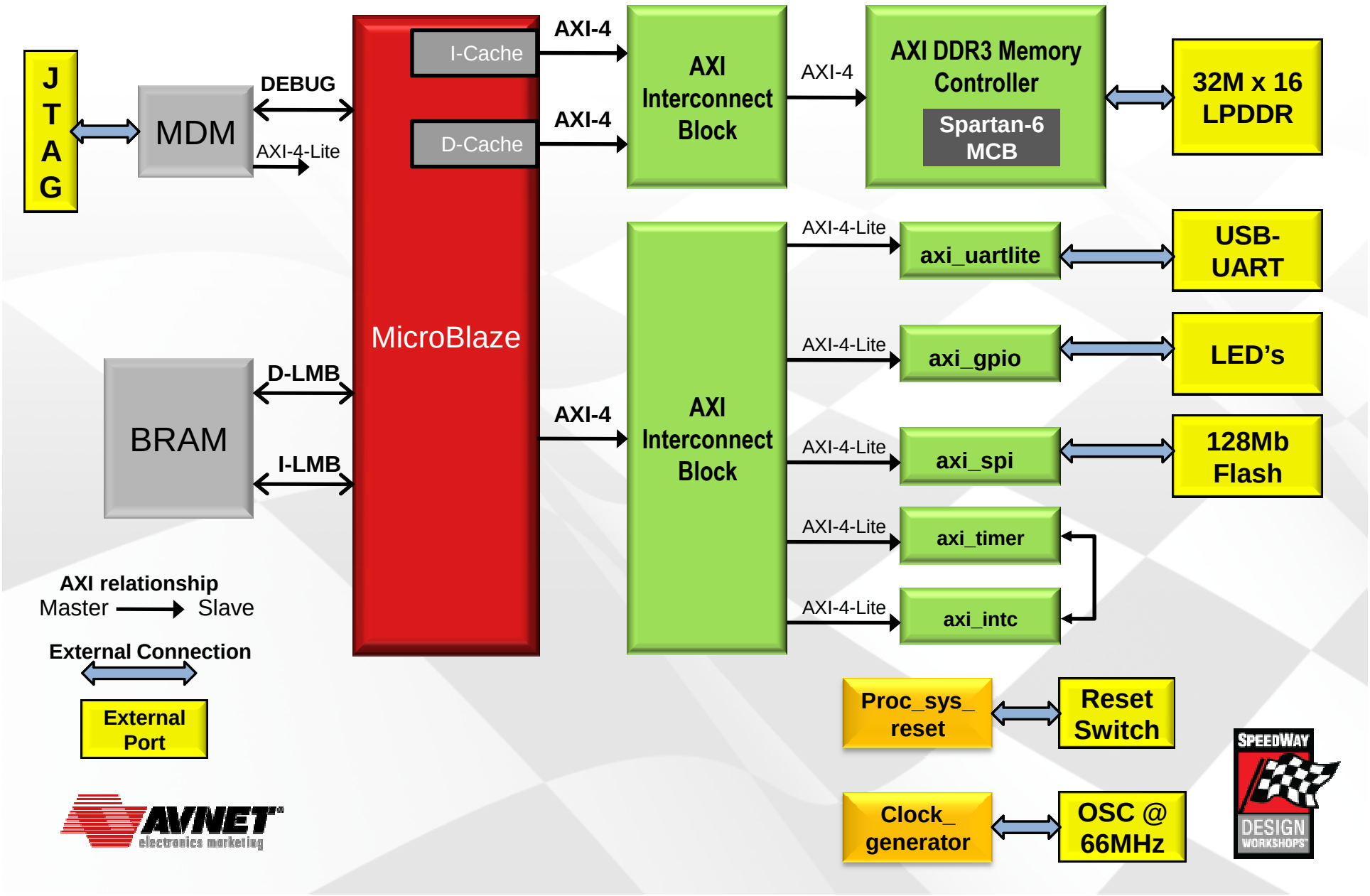
Introduction

- **This lab guides you through the process of using the Xilinx Project Navigator (the ISE® tool) and Platform Studio (XPS) to create a simple processor system. In the process, you will use BSB to create an MHS file, import design to ISE, and generate a bitstream**

Objectives

- Create an XPS processor system inside of an ISE project
- Build an embedded processor system using Base System Builder
- Generate a top-level wrapper for the embedded processor component in ISE Project Navigator
- Implement the project in the ISE tool
- Create a sample C application and download it with SDK

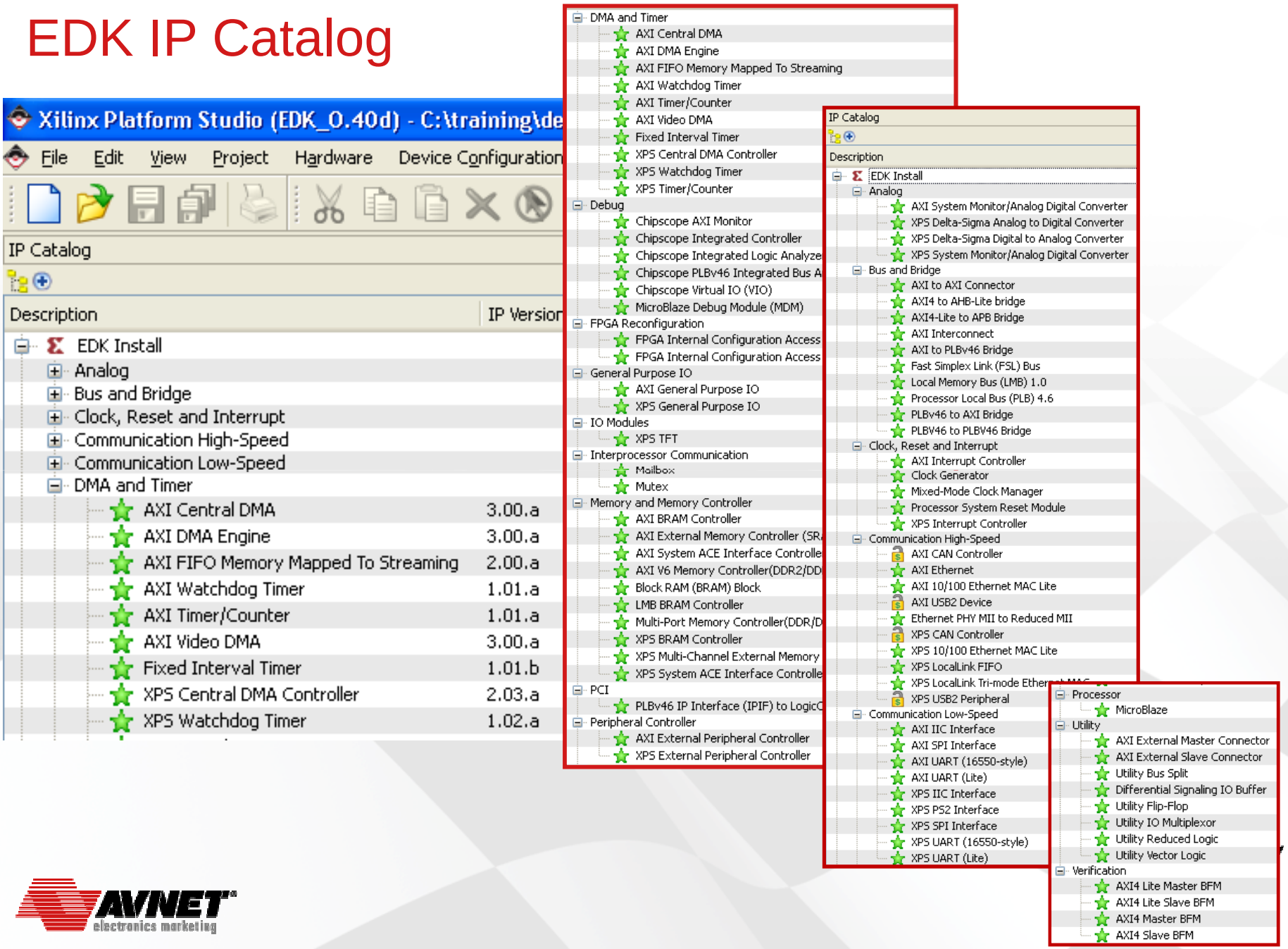
Lab #1 - MicroBlaze Platform



Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

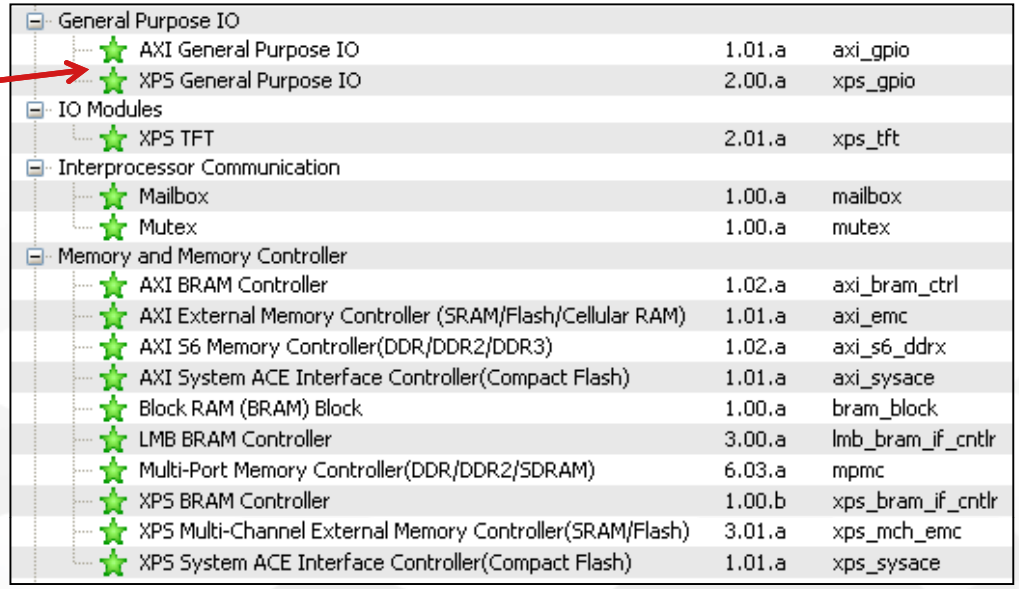
EDK IP Catalog

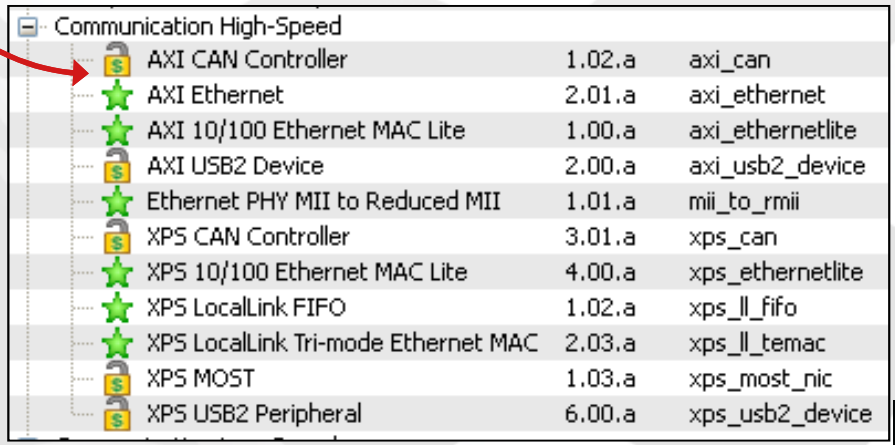


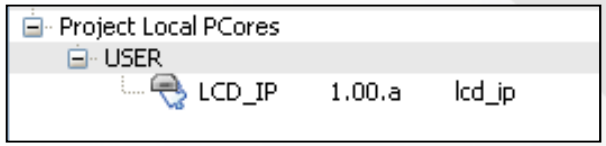
AVNET electronics marketing

IP Peripherals

- **Most IP cores are free to use**
- **The more exotic IP cores are licensed $$**
  - Xilinx developed, delivered, and supported
  - Evaluation IP installs with a 90-day evaluation license
- **You can also add your own IP cores to the catalog**







AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

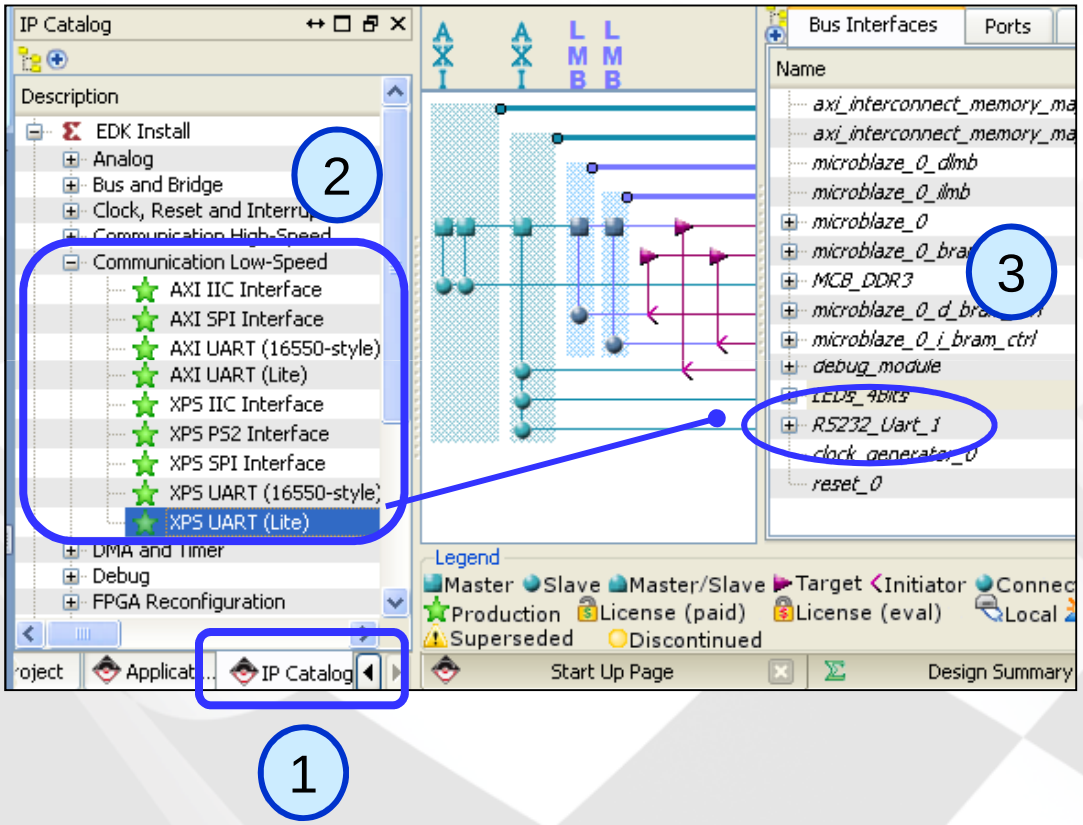
Core Sizes

- **The size of each core is available in the data sheet**
- **For example, the XPS LL TEMAC soft Ethernet core data sheet contains the following table**

Parameter Values					Device Resources				f_{MAX} (MHz)
C_TEMAC1_ENABLED	C_TEMAC1_TXFIFO C_TEMAC1_RXFIFO C_TEMAC0_TXFIFO C_TEMAC0_RXFIFO	C_TEMAC0_TXCSUM C_TEMAC0_RXCSUM	C_TEMAC1_TXCSUM C_TEMAC1_RXCSUM	C_TEMAC_TYPE	Slices	Flip- Flops	BRAMS	4-input LUTs	f_{MAX}
0	2048	0	0	2	1291	2399	8	2277	
0	2048	0	0	**0**	501	920	6	748	
0	**16384**	0	0	**2**	1235	2444	22	2428	
1	16384	0	0	2	2515	4571	44	4625	
1	16384	**1**	**1**	2	2731	5061	44	5167	
0	16384	1	**0**	2	1422	2689	22	2699	
0	**32768**	**0**	0	2	1422	2465	38	2495	

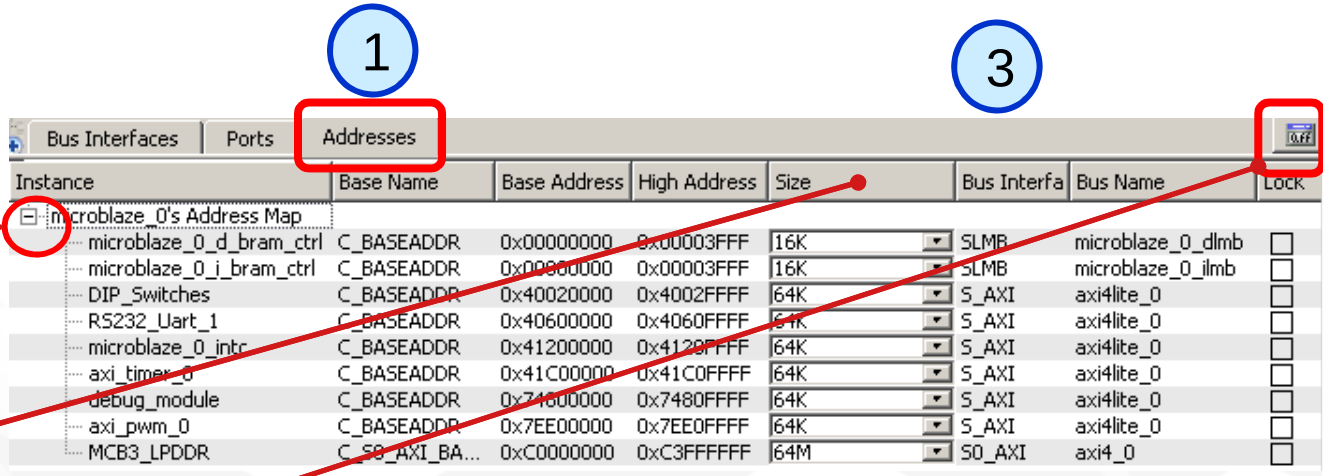
Adding IP to the Design

1. To add hardware in a new, empty project, or to an existing project, select the IP Catalog tab in XPS
2. Expand the group(s) of IP in the left window
3. Select an IP and drag it to the System Assembly View window
   - Will automatically be added to the system MHS file

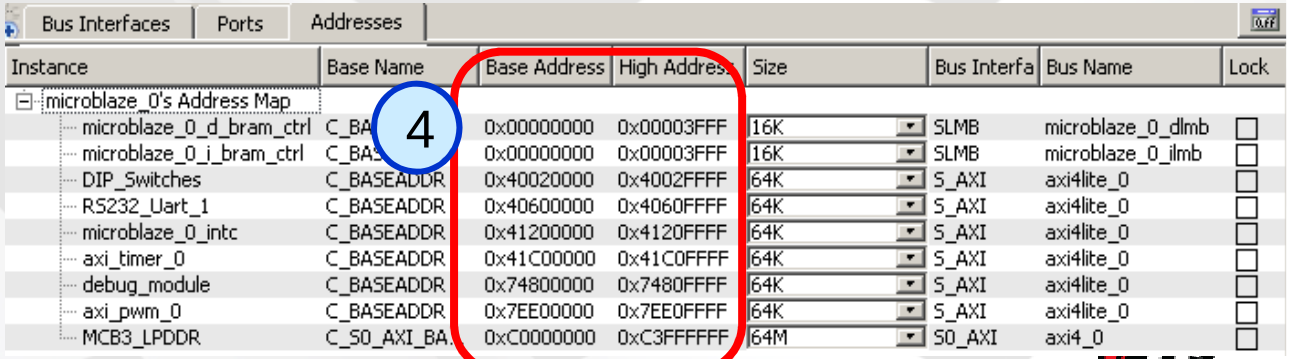


Assigning Addresses

1. Select the **Addresses** tab
2. Expand (**+**) to see addresses
3. Click in the Size column and select the desired size or click **Generate Addresses**
4. Optionally, Enter the base address
   - Default address will automatically be generated
   - XPS calculates the high address from the base address and size entries



Mapped New Components

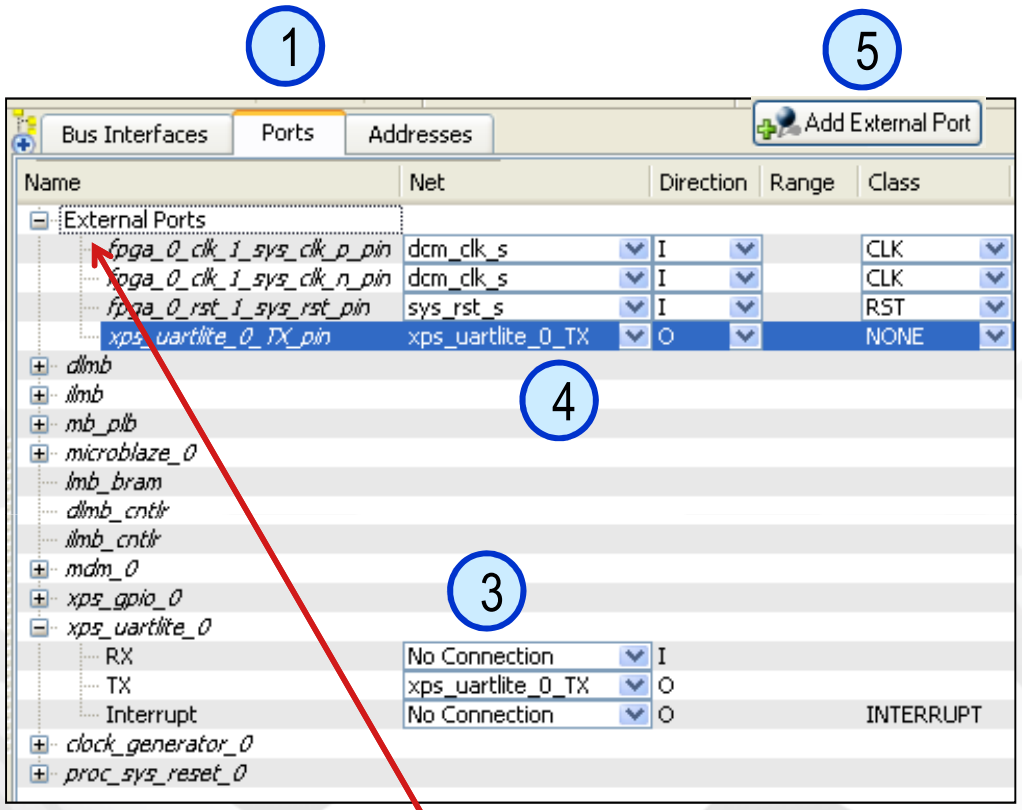


AVNET electronics marketing

DESIGN WORKSHOPS

Connecting Ports

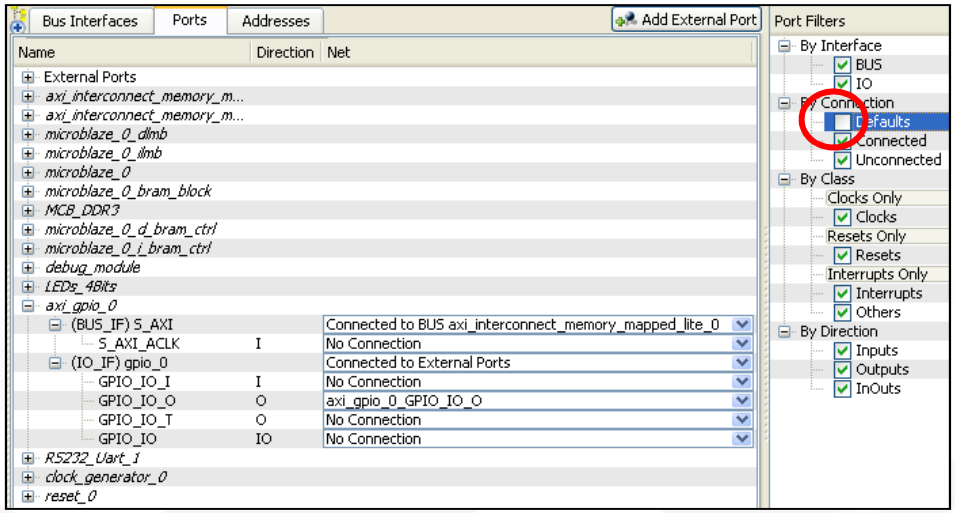
1. Select the **Ports** tab
2. Expand ports listing
3. Click under the Net column and select the appropriate signal
4. If the port is external in the design, make it external. A default name will be created
5. For global ports, click **Add External Port** and assign a name

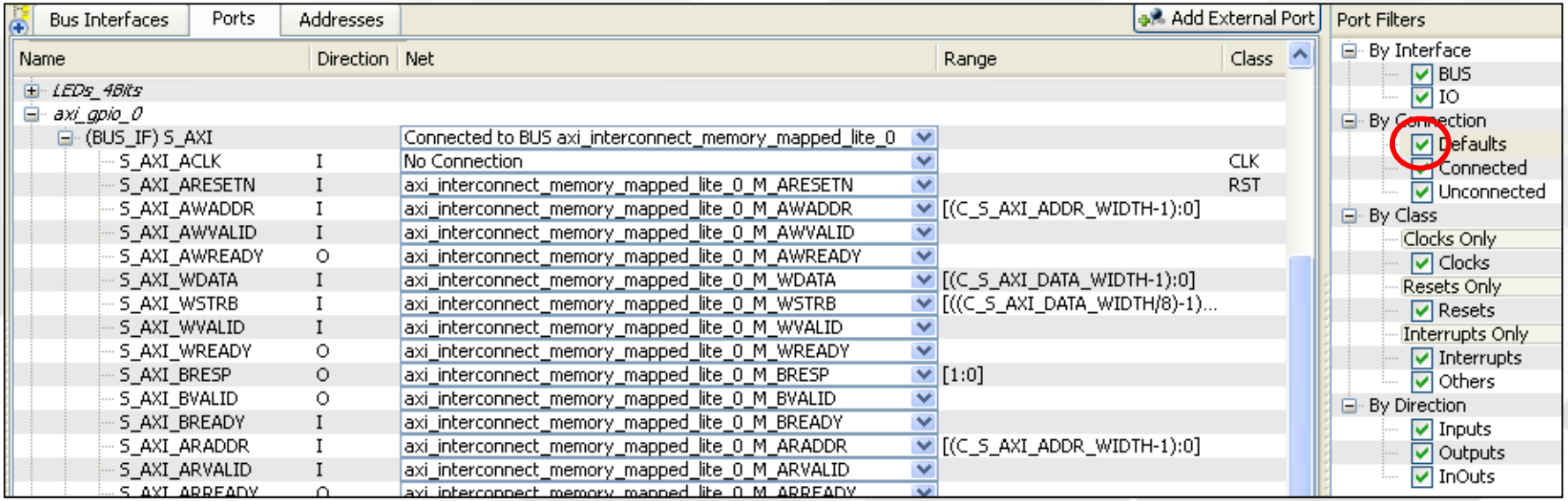


All External Ports will be displayed in the External Ports List

Port Filters

- Port filters remove undesired congestion in the net column by sorting on ports of interest





AVNET electronics marketing

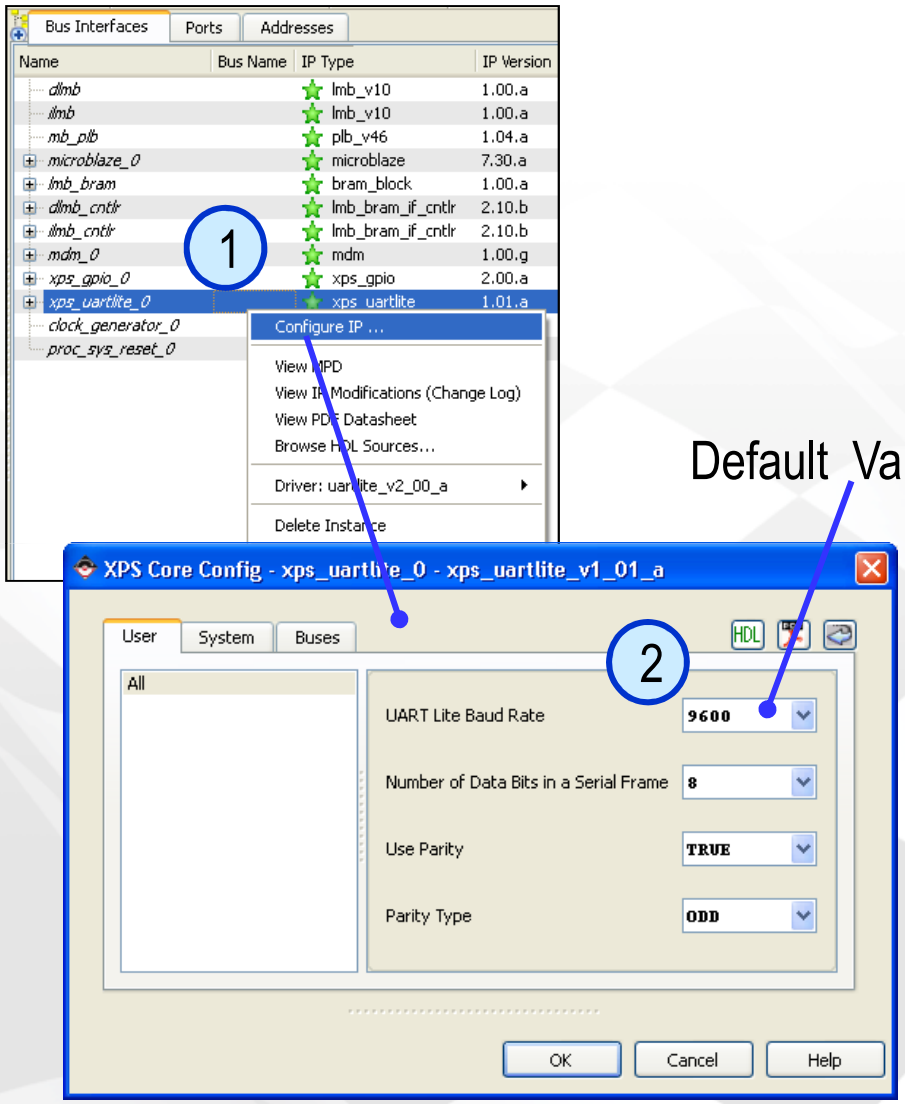
SPEEDWAY DESIGN WORKSHOPS

Parameterize IP Instances

1. Double-click or right click the instance and select **Configure IP** to open the configurable parameters dialog box (refer to the datasheet if needed)

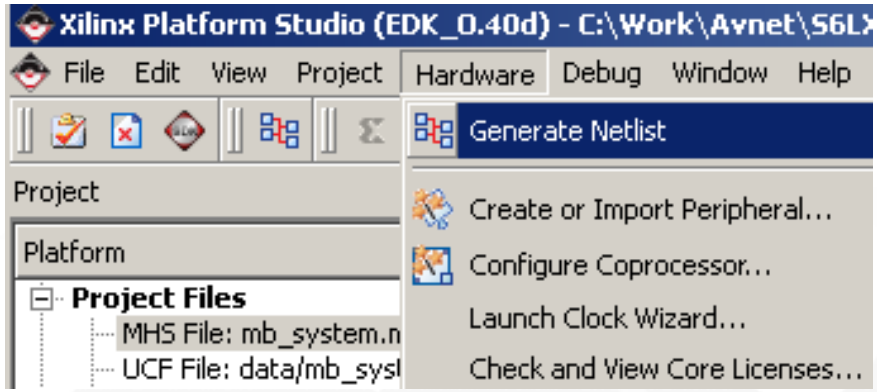
2. Default values are shown. Customize the parameters that you want

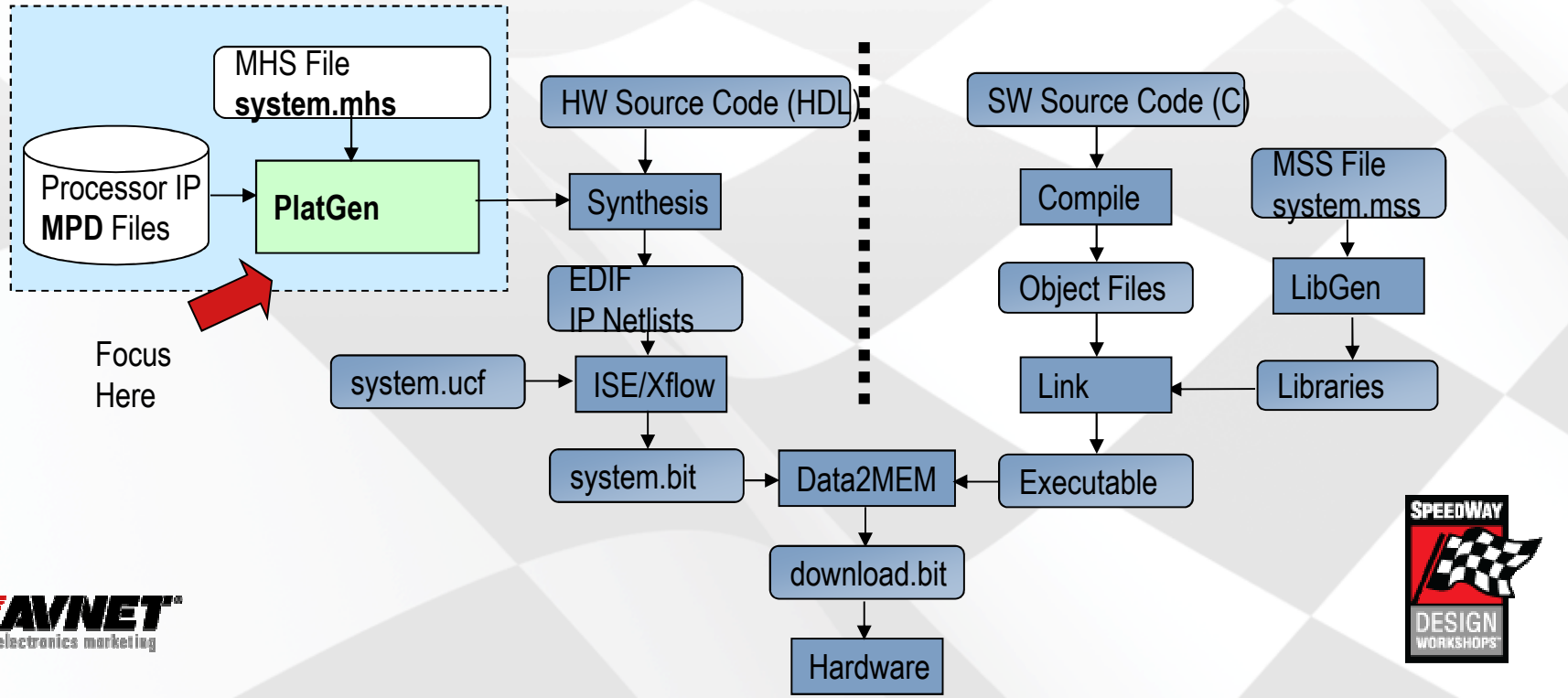
MHS File will be Updated



XPS – Platform Generator (PlatGen)

- **After defining the system, hardware and connectivity, the next step is to create hardware netlists with the Platform Generator (PlatGen)**
  - **Alternatively,** run Synthesis in ISE Project Navigator (Lab)





Hardware Design – Generate Netlist (PlatGen)

- **PlatGen inputs the following files**
  - Microprocessor Hardware Specification (**MHS**) file
  - Microprocessor Peripheral Definitions (**MPD**) file
- **PlatGen constructs the embedded processor system in the form of hardware netlists for each component (HDL wrappers and implementation netlist files)**
- **MHS file parameters override MPD parameters**

MHS File

```
BEGIN xps_uartlite
 PARAMETER INSTANCE = RS232_Uart_1
 PARAMETER C_FAMILY = virtex5
 PARAMETER C_BAUDRATE = 115200
 PARAMETER C_DATA_BITS = 8
 PARAMETER C_USE_PARITY = 0
 PARAMETER C_ODD_PARITY = 0
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BASEADDR = 0x84000000
 PARAMETER C_HIGHADDR = 0x8400ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT RX = fpga_0_RS232_Uart_1_RX_pin
 PORT TX = fpga_0_RS232_Uart_1_TX_pin
END
```

Microprocessor Peripheral Definitions (MPD) File

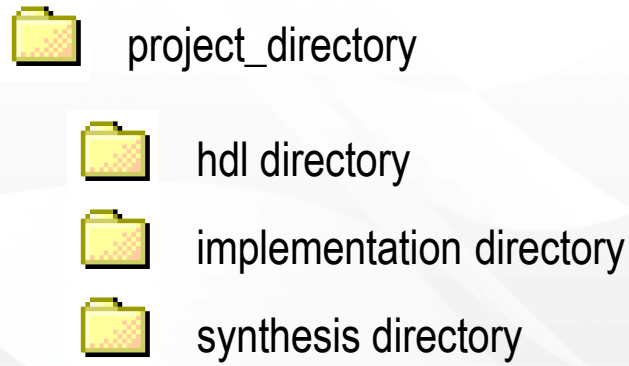
```
## Generics for VHDL or Parameters for Verilog
PARAMETER C_FAMILY = virtex5, DT = STRING
PARAMETER C_SPLB_CLK_FREQ_HZ = 100000000, DT = INTEGER, BUS = SP
PARAMETER C_BASEADDR = 0xffffffff, DT = std_logic_vector(0 to 31
PARAMETER C_HIGHADDR = 0x00000000, DT = std_logic_vector(0 to 31
PARAMETER C_SPLB_AWIDTH = 32, DT = INTEGER, BUS = SPLB, ASSIGNME
PARAMETER C_SPLB_DWIDTH = 32, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_P2P = 0, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_MID_WIDTH = 1, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_NUM_MASTERS = 1, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_SUPPORT_BURSTS = 0, DT = INTEGER, BUS = SPLB, A
PARAMETER C_SPLB_NATIVE_DWIDTH = 32, DT = INTEGER, BUS = SPLB, A
PARAMETER C_BAUDRATE = 9600, DT = INTEGER, DESC = Baudrate, PERM
PARAMETER C_DATA_BITS = 8, DT = INTEGER, RANGE = (5:8), DESC = N
PARAMETER C_USE_PARITY = 1, DT = INTEGER, RANGE = (0,1), DESC =
PARAMETER C_ODD_PARITY = 1, DT = INTEGER, RANGE = (0,1), DESC =
```

MPD contains all of the defaults

PlatGen

Use these if importing to PlanAhead

PlatGen-Generated Directories



- *hdl* directory
  - *system.[vhd|v]* file (if top level)
  - *system_stub.[vhd|v]* file (if submodule)
  - *peripheral_wrapper.[vhd|v]* files
- *implementation* directory
  - *peripheral_wrapper.ngc* files
  - *system.ngc* file
  - *system.bmm* file
- *synthesis* directory
  - *peripheral_wrapper.[prj|scr]* files
  - *system.[prj|scr]* files

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

PlatGen Memory Generation

- **Memory generation**
  - Platform Generator generates memory using block RAMs based on the memory sized allocated to the memory controller
  - A **BMM** file is also generated.
    - Defines the architecture of the memory built
    - Placement information is later added to the BMM file by PAR
    - Used by Data2MEM, to load ELF software object into block RAMs, by indicating the appropriate BRAM bits in the BIT file to populate
- **Current block RAM controllers include the following**
  - XPS block RAM controller (xps_bram_if_cntlr)
  - AXI block RAM controller (axi_bram_cntlr)
  - LMB block RAM Controller (lmb_bram_if_cntlr)

System.bmm File

- Generated by PlatGen when it instantiates BRAM
  - Netlist indicating the number of block RAMs used and their connection topology.

```
ADDRESS_SPACE plb_bram_if_cntlr_2_bram_combined COMBINED [0xfffe0000:0xfffe3fff]
        ADDRESS_RANGE RAMB32
        BUS_BLOCK
                plb_bram_if_cntlr_2_bram/plb_bram_if_cntlr_2_bram/ramb36_0 [31:24] ;
                plb_bram_if_cntlr_2_bram/plb_bram_if_cntlr_2_bram/ramb36_1 [23:16] ;
                plb_bram_if_cntlr_2_bram/plb_bram_if_cntlr_2_bram/ramb36_2 [15:8] ;
                plb_bram_if_cntlr_2_bram/plb_bram_if_cntlr_2_bram/ramb36_3 [7:0] ;
        END_BUS_BLOCK;
        END_ADDRESS_RANGE;
END_ADDRESS_SPACE;
ADDRESS_SPACE xps_bram_if_cntlr_1_bram_combined COMBINED [0xffffc000:0xffffffff]
        ADDRESS_RANGE RAMB32
        BUS_BLOCK
                xps_bram_if_cntlr_1_bram/xps_bram_if_cntlr_1_bram/ramb36_0 [63:48] ;
                xps_bram_if_cntlr_1_bram/xps_bram_if_cntlr_1_bram/ramb36_1 [47:32] ;
                xps_bram_if_cntlr_1_bram/xps_bram_if_cntlr_1_bram/ramb36_2 [31:16] ;
                xps_bram_if_cntlr_1_bram/xps_bram_if_cntlr_1_bram/ramb36_3 [15:0] ;
        END_BUS_BLOCK;
        END_ADDRESS_RANGE;
END_ADDRESS_SPACE;
```

PlatGen BMM Memory Sizes

- **Memory size must be aligned on a 2^n address boundary**
- **For non-2^n boundaries of contiguous memory, instantiate multiple block RAM memory controllers**
  - You may have to manually edit the controller base addresses to be contiguous

FPGA Architecture	Memory Size (kB) 32-bit LMB or PLBv46 Bus	Memory Size (kB) 64-bit PLBv46 Bus
Spartan®-3(E)	8, 16, 32, 64	16, 32, 64, 128
Spartan-3A(DSP)	2, 4, 8, 16, 32, 64	4, 8, 16, 32, 64, 128
Spartan-6	2, 4, 8, 16, 32, 64	4, 8, 16, 32, 64, 128
Virtex®-4	2, 4, 8, 16, 32, 64, 128	4, 8, 16, 32, 64, 128, 256
Virtex-5	4, 8, 16, 32, 64, 128, 256	8, 16, 32, 64, 128, 256, 512
Virtex-6	4, 8, 16, 32, 64, 128, 256	8, 16, 32, 64, 128, 256, 512

Checkpoint!

- **What is the smallest memory size that PlatGen can generate for a Spartan-6 FPGA on a 32-bit LMB bus?**

  2 KB

- **What will the BAUDRATE for this peripheral be?**
  - If the MPD file has the following parameter: C_BAUDRATE = 9600
  - If the MHS file has the following parameter: C_BAUDRATE = 115200

  The MHS file overrides MPD, thus 115200.

- **What tool is invoked to generate a XPS netlist?**

  PlatGen

- **What are PlatGen's inputs?**

  MHS and all MPD files

- **What are PlatGen's outputs?**

  NGC Netlist, HDL wrapper files for all cores, and BMM files

Summary

- **Adding EDK IP is easy**
- **The primary output of PlatGen is a netlist for each component of your embedded hardware platform**
  - Each bus, peripheral, and processor in your hardware system will have its own NGC netlist (stored in the implementation directory)
  - PlatGen also makes a synthesis and an HDL directory that contains all the necessary scripts and wrappers to synthesize your design
- **PlatGen automatically generates a memory array structure**
  - Only for block RAM controllers: axi_bram_cntlr, xps_bram_if_cntlr, and lmb_bram_if_cntlr
  - Size is determined by the address space allocated to the memory controller component
  - BMM file defines the memory subsystem structure
  - Generated memory size is always on 2^n boundaries

Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - **Lab 2 - Adding EDK IP**
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Lab 2: Adding EDK IP

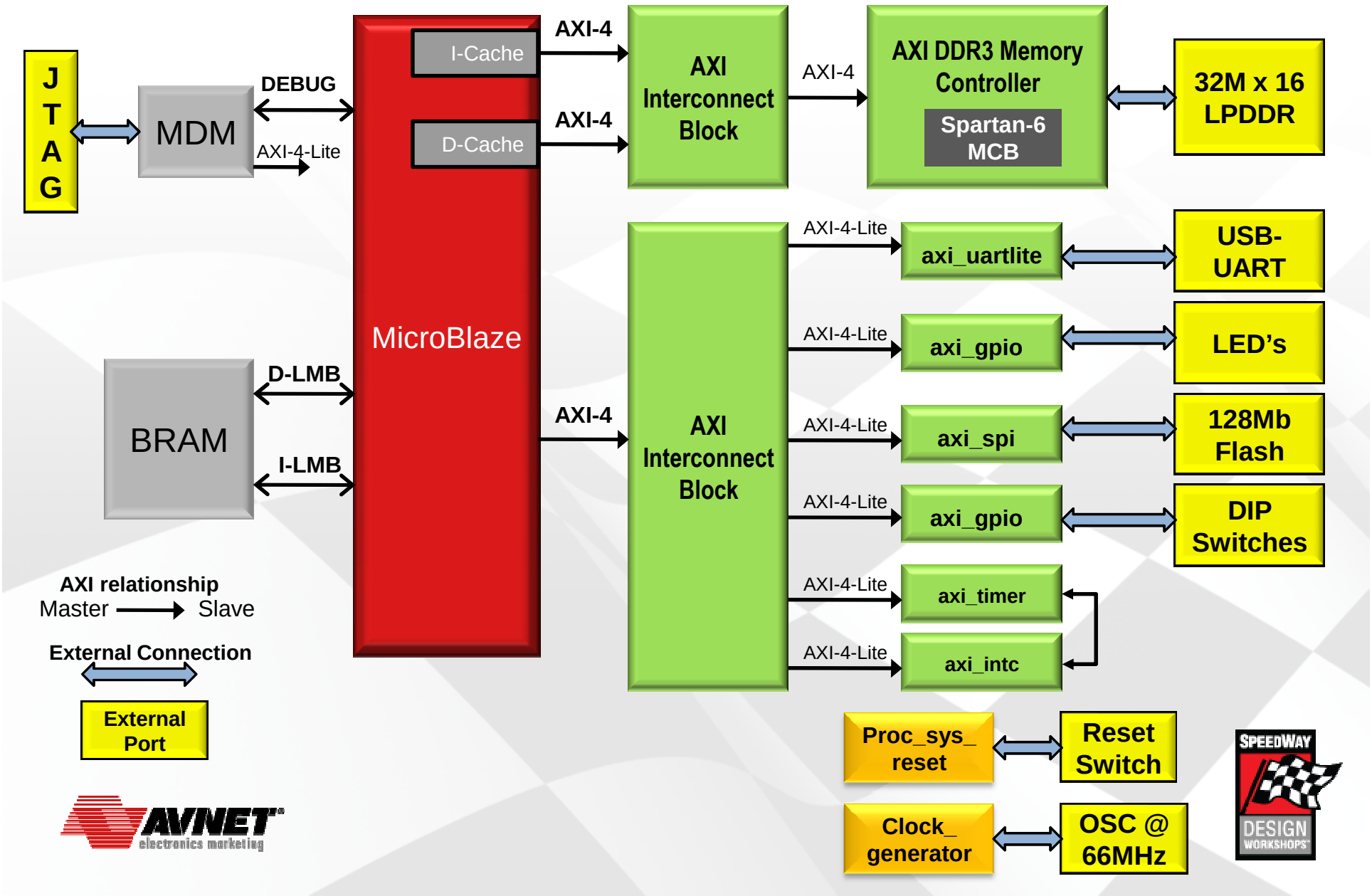
Introduction

- **This tutorial demonstrates how to add and modify peripherals to an existing MicroBlaze system using Xilinx Platform Studio (XPS). The system from the previous tutorial will be used as the starting point**

Objectives

- How to add an EDK peripheral
- How to connect the new peripheral to the existing system
- How to modify the peripheral options
- How to add constraints for the new peripheral
- How to add a software application to support the new peripheral

Lab #2 - MicroBlaze Platform



Agenda

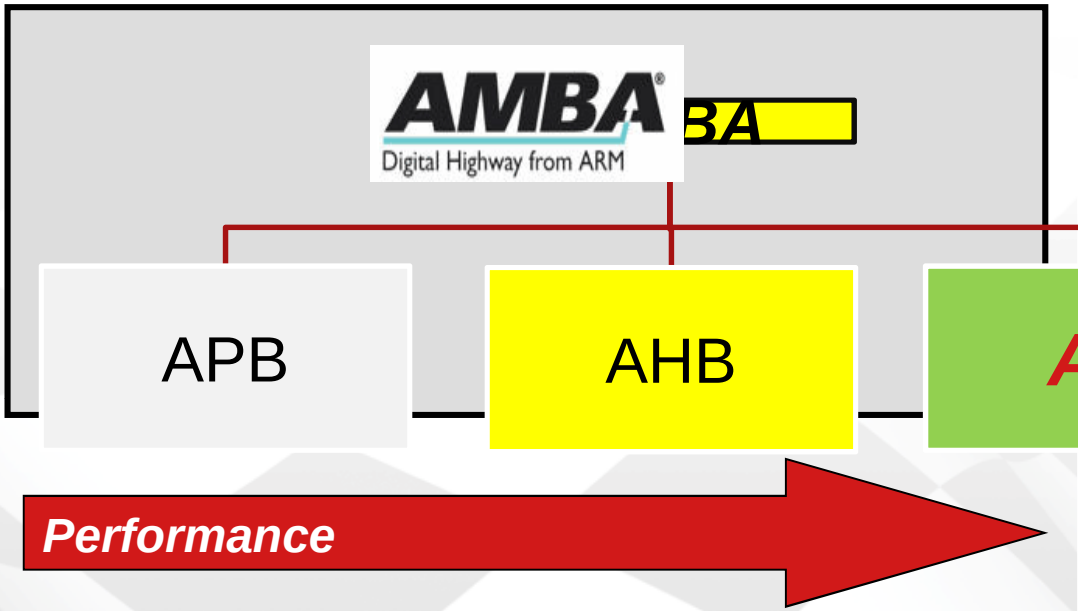
- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

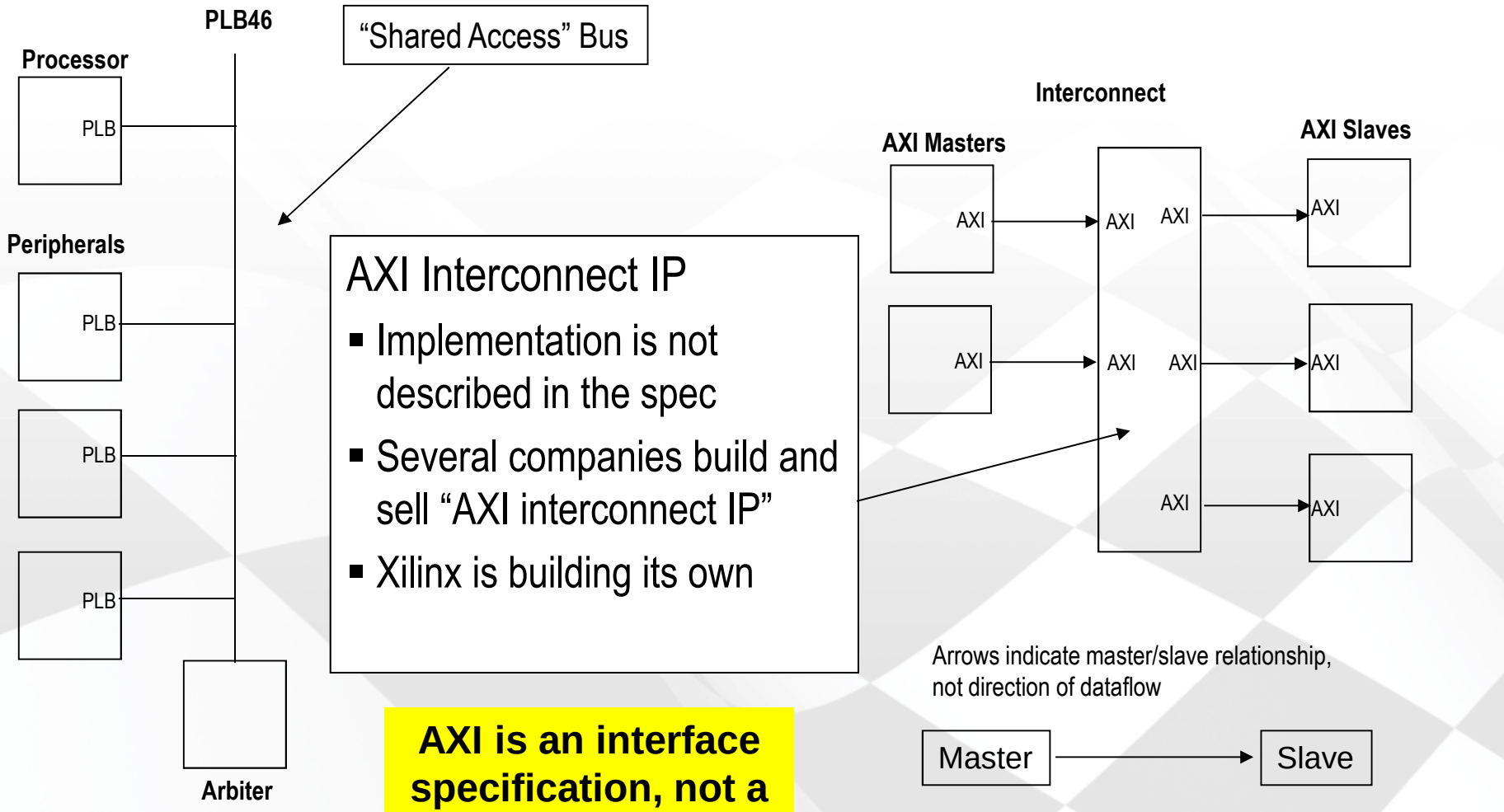
AXI is Part of AMBA:
Advanced Microcontroller Bus Architecture

- **AXI Is. . .**
  - An interface and protocol definition
  - Widely used industry standard
- **AXI Is not. . .**
  - A bus



The AXI specification describes an interface on a piece of IP.
It does not specify how systems of IP will be connected.

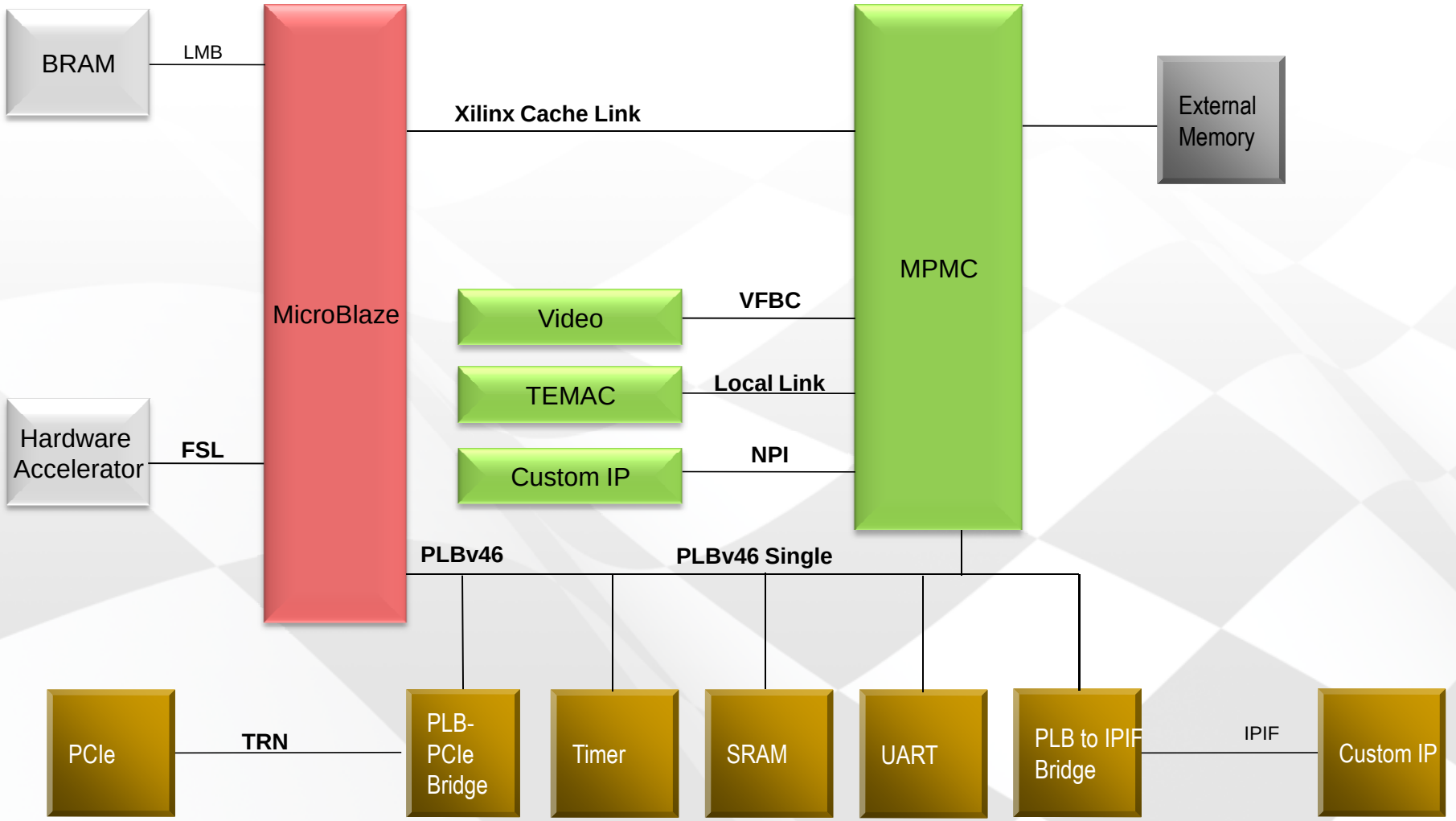
PLB is a Bus Spec / AXI is an Interface Spec



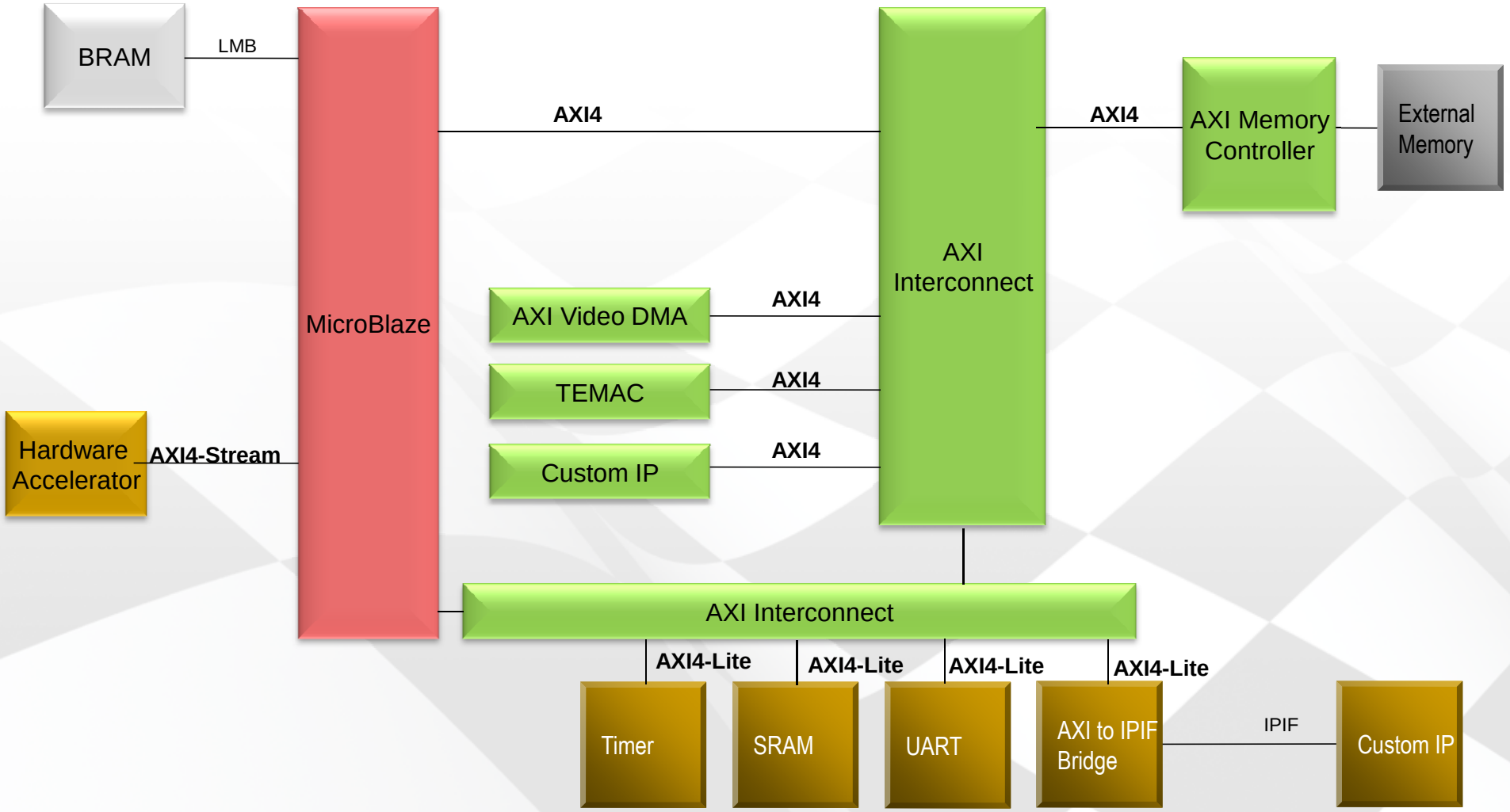
Goals of Transition to AXI

- **Higher performance**
  - AXI allows systems to be optimized for highest Fmax, maximum throughput, lower latency or some combination of those attributes. This flexibility enables you to build the most optimized products for your markets
- **Easier to use**
  - By consolidating a broad array of interfaces into AXI, you only need to know one family of interfaces, regardless of whether they are embedded, DSP or logic users. This makes it easier to integrate IP from different domains, as well as developing your own IP
- **Enable ecosystem**
  - Partners are embracing the move to AXI: an open, widely adopted interface standard. Many of them are already creating IP targeting AXI and other AMBA® interfaces. This gives you a greater catalog of IP, ultimately leading to faster time to market

Previous MicroBlaze Design - *Many interfaces*



Now -*Standardize on AXI4*

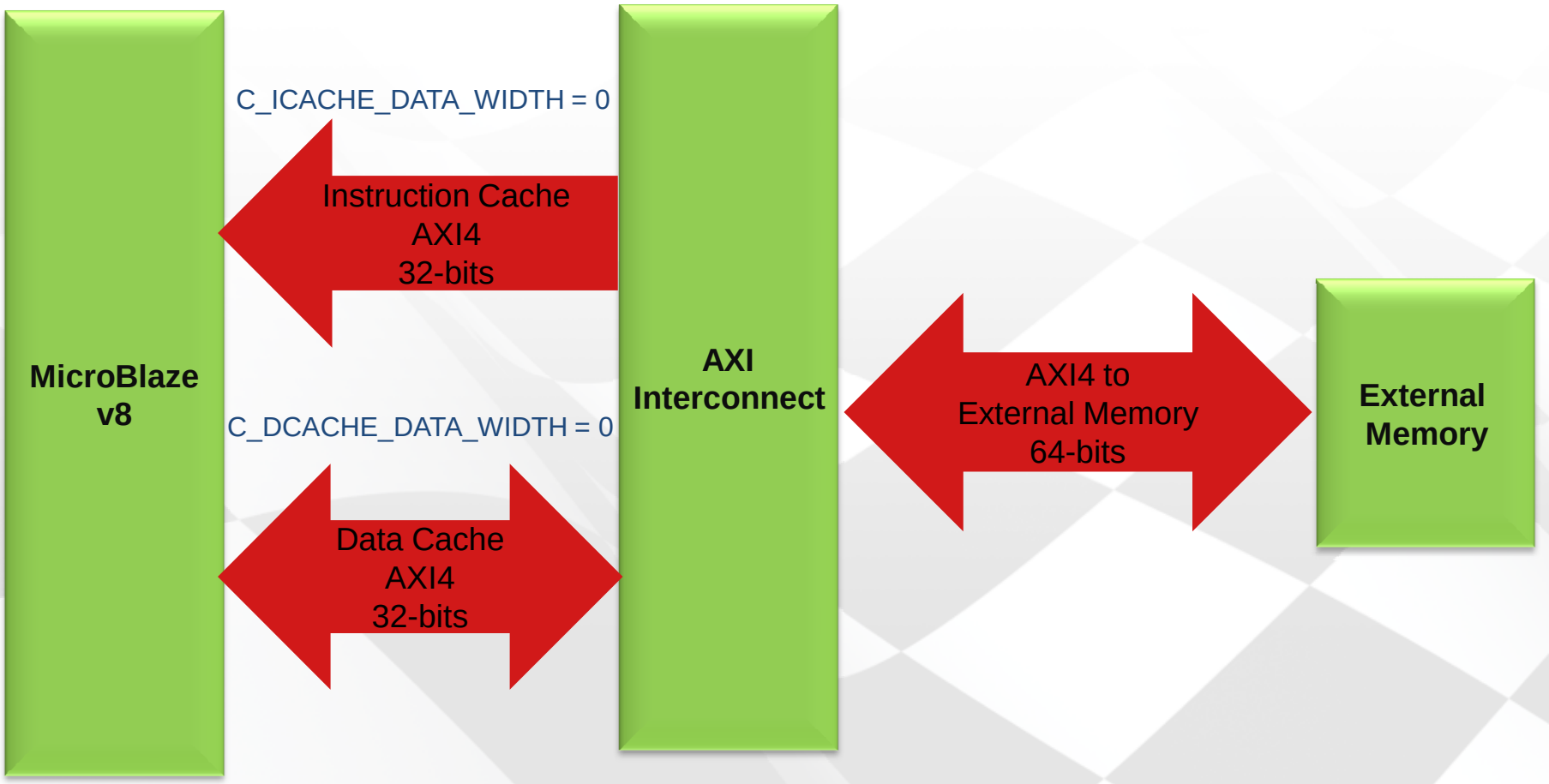


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

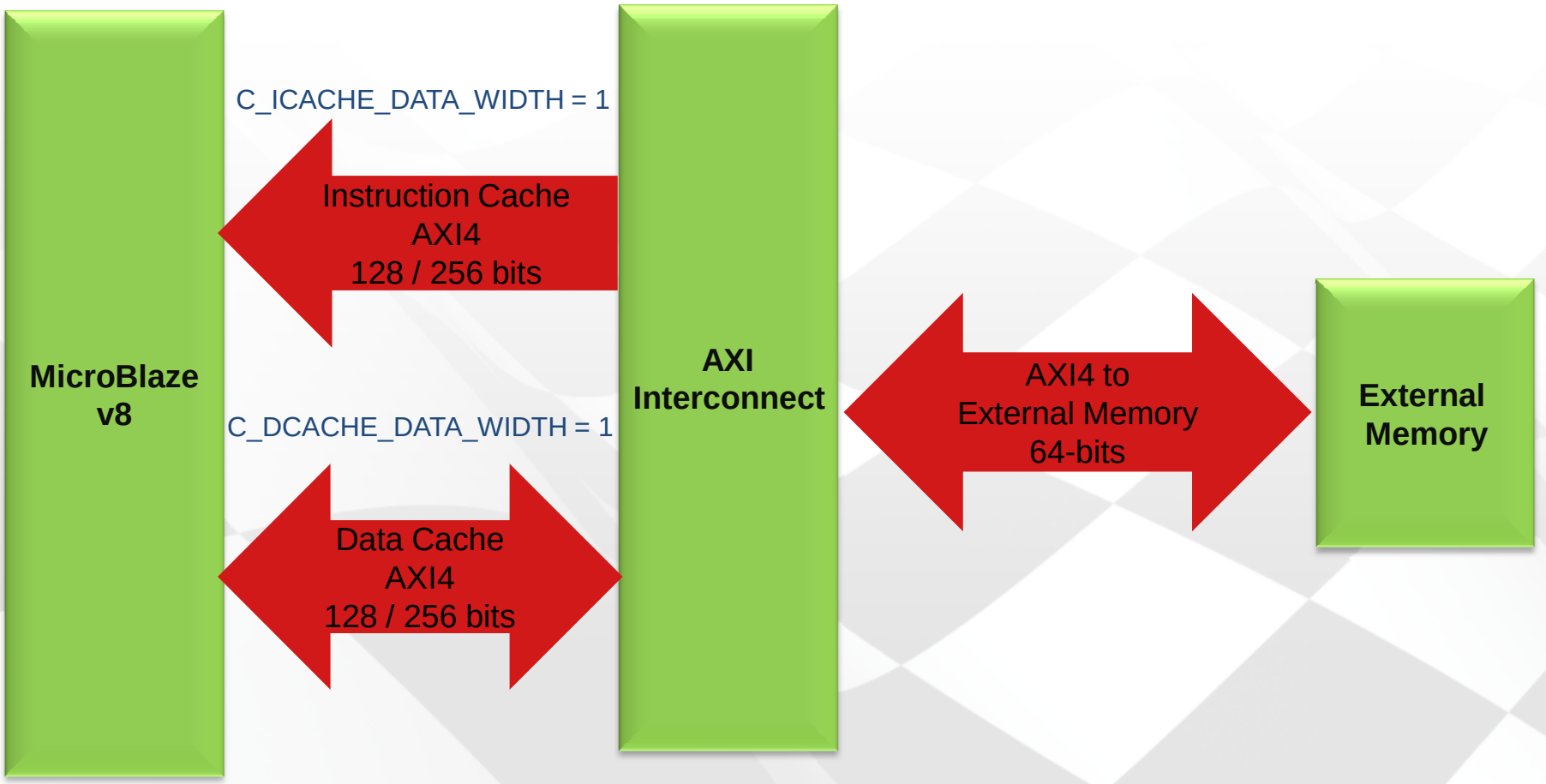
MicroBlaze Cache to External Memory Datapath

Default ,Configuration, same as pre-AXI



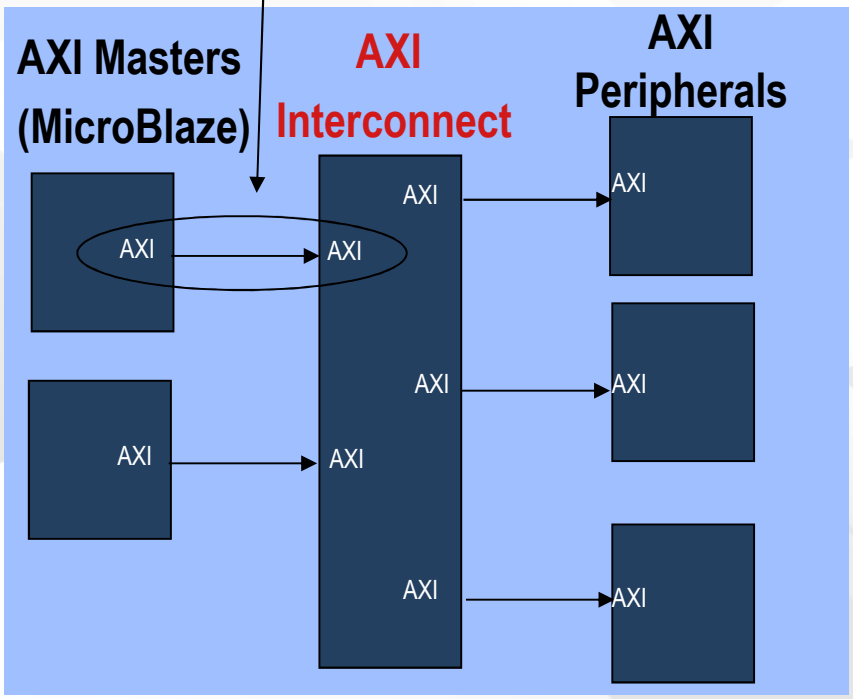
MicroBlaze Cache to External Memory Datapath

Highest Performance, MicroBlaze port widths match AXI cache line width



AXI Interconnect Block
Connects Multiple Master / Slave Pairs

AXI defines a point to point, master/slave interface



- Up to **16** masters and **16** slaves per interconnect
- 32 to 256 bit data widths per endpoint
- Built-in AXI4, AXI4-Lite and AXI3 protocol conversion
- Each master / slave pair has own clock domain
- Pipeline registers per channel to boost timing

To use AXI Interconnect (and other "embedded" AXI IP) outside of Xilinx Platform Studio, consult Answer Record 37856

Plug and Play IP

Achieved via adoption of industry standards

AMBA AXI4	IP-XACT	IEEE P1735
Standard Interconnect	Standard Packaging	Standard Security
• IP Verification with AXI4 BFM within Xilinx Platform Studio • Up to 20% higher system bandwidth and up to 50% better area for the interconnect block	• IP Repository with common “look & feel” for Xilinx and partner IP (50 IP-XACT cores) • IP Packager available to Alliance Program Members	• Support for AXI BFM for simulation • Fully functional encryption flow for partners (13.3)

Now Easier to Access, Integrate and Protect IP

AXI4 IP for 6 and 7 Series FPGAs

AXI4 - Advanced Extensible Interface

Standard Overview

- **AXI4**
  - Three flavors: AXI4, AXI4-Lite, AXI4-Stream
  - All three share same terminology, signal naming, …

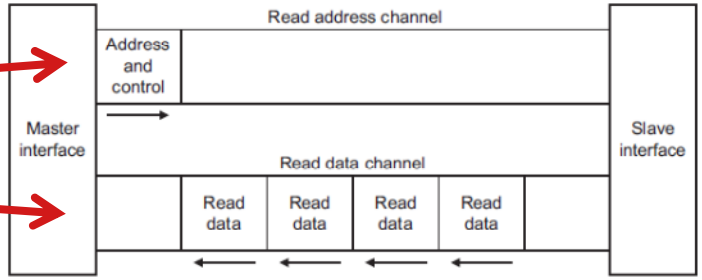
	AXI4	AXI4-Lite	AXI4-Stream
Dedicated for	high-performance and memory mapped systems	register-style interfaces (area efficient implementation)	non-address based IP
Burst** (data beat)	up to 256	1	unlimited
Data Width	32 to 1024 bits	32 or 64 bits	any number of bytes
Applications (examples)	Embedded, memory	Small footprint control logic	DSP, video, communication

***Burst based - only a start address required to initiate a transaction*

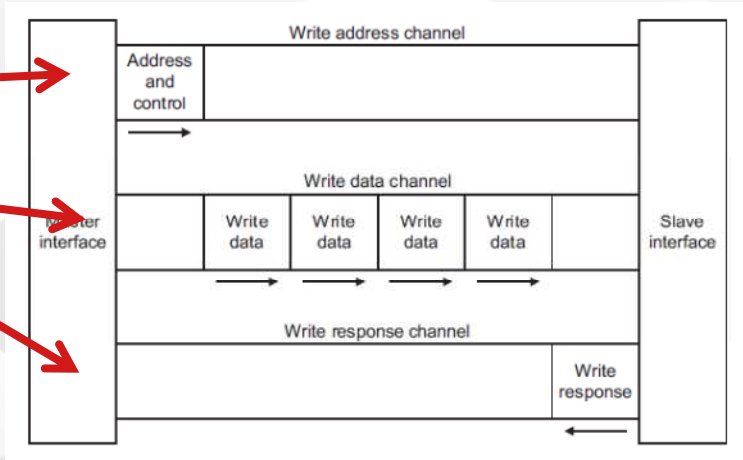
Basic AXI Signaling - AXI-4

Five Channels

1. **Read Address Channel**
2. **Read Data Channel**

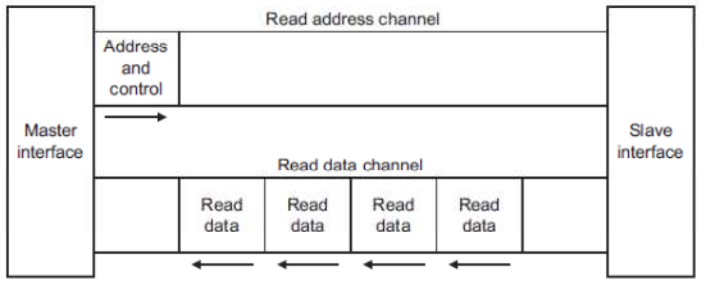


3. **Write Address Channel**
4. **Write Data Channel**
5. **Write Response Channel**
   - Posted write model: there will always be a “Write Response”



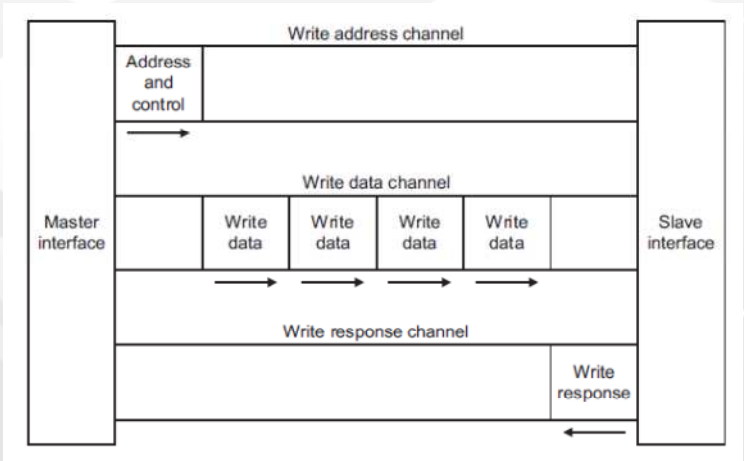
The AXI Interface - AXI4

- **Also called:**
  - Full AXI
  - AXI Memory Mapped



AXI4 Read

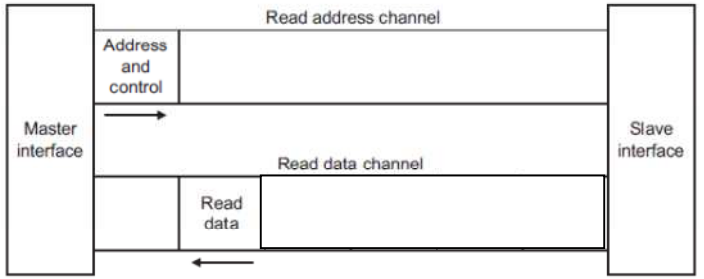
- **Single address multiple data**
- **Burst up to 256 data beats**
- **Data widths 32-1024 bits wide**



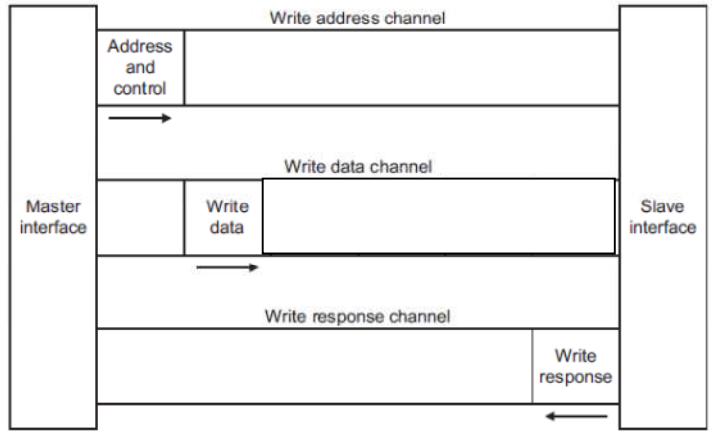
AXI4 Write

AXI Interface: Lite

- **No burst**
- **Data width 32 or 64 only**
  - Xilinx IP will only support 32 bits
- **Simple “logic shim” to connect AXI4 master to AXI4-Lite slave**
  - Reflect master’s transaction ID



AXI4-Lite Read

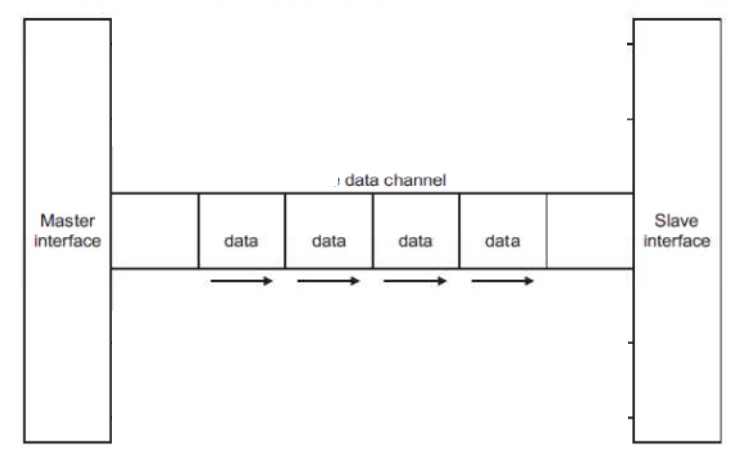


AXI4-Lite Write

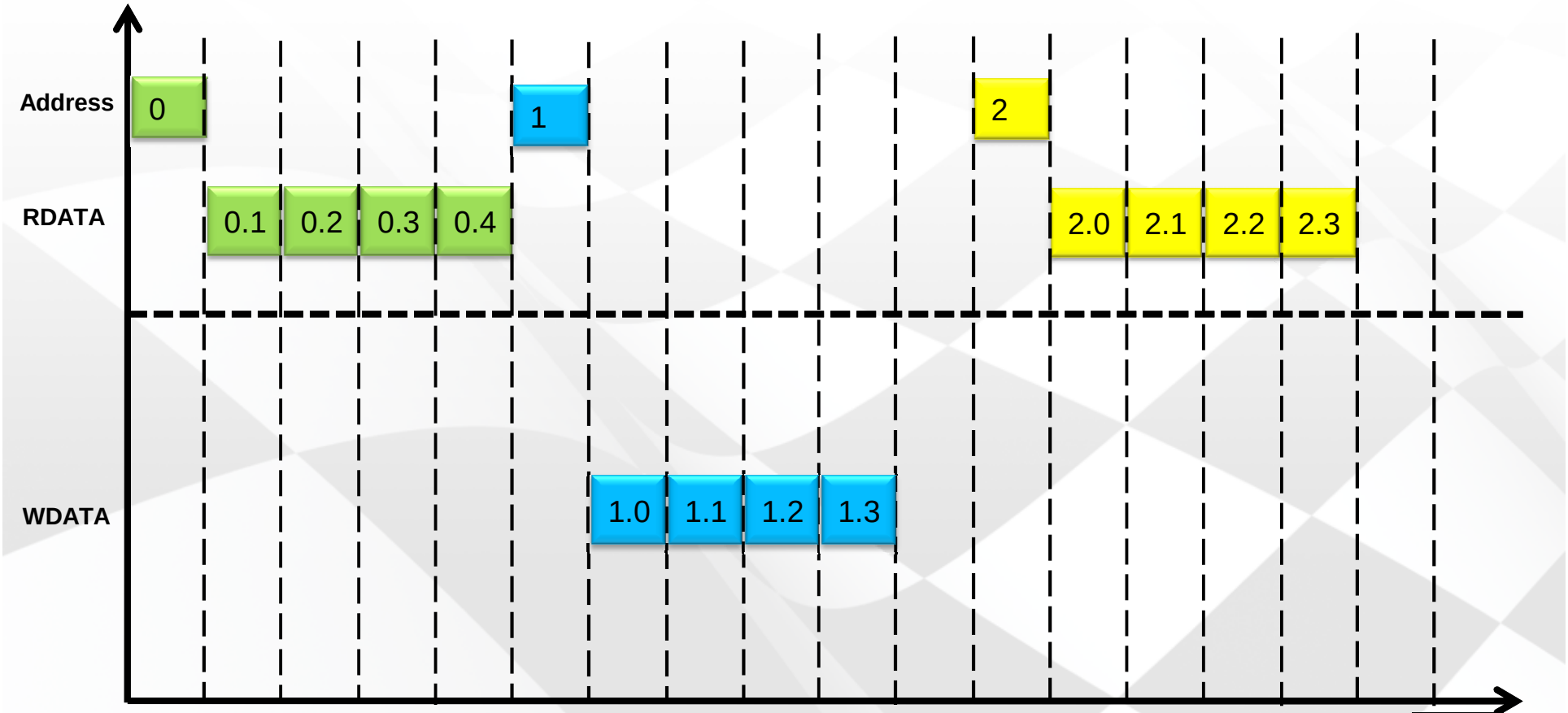
The AXI Interface - Streaming

- **No address channel**
- **Not read and write, always just master to slave**
- **Unlimited burst length**

AXI4-Streaming Transfer

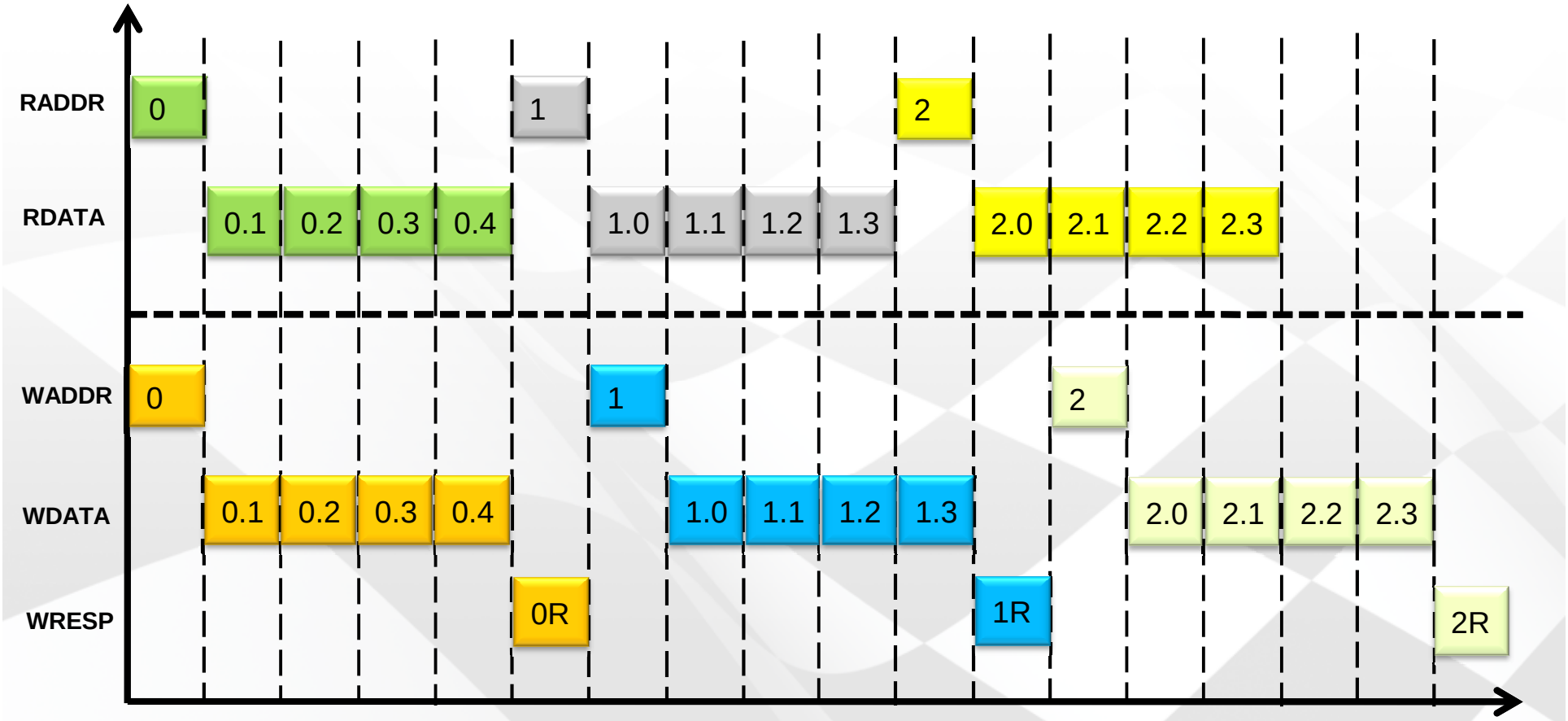


AXI vs. PLB v46 Comparison: Three Back-to-Back PLB Transactions



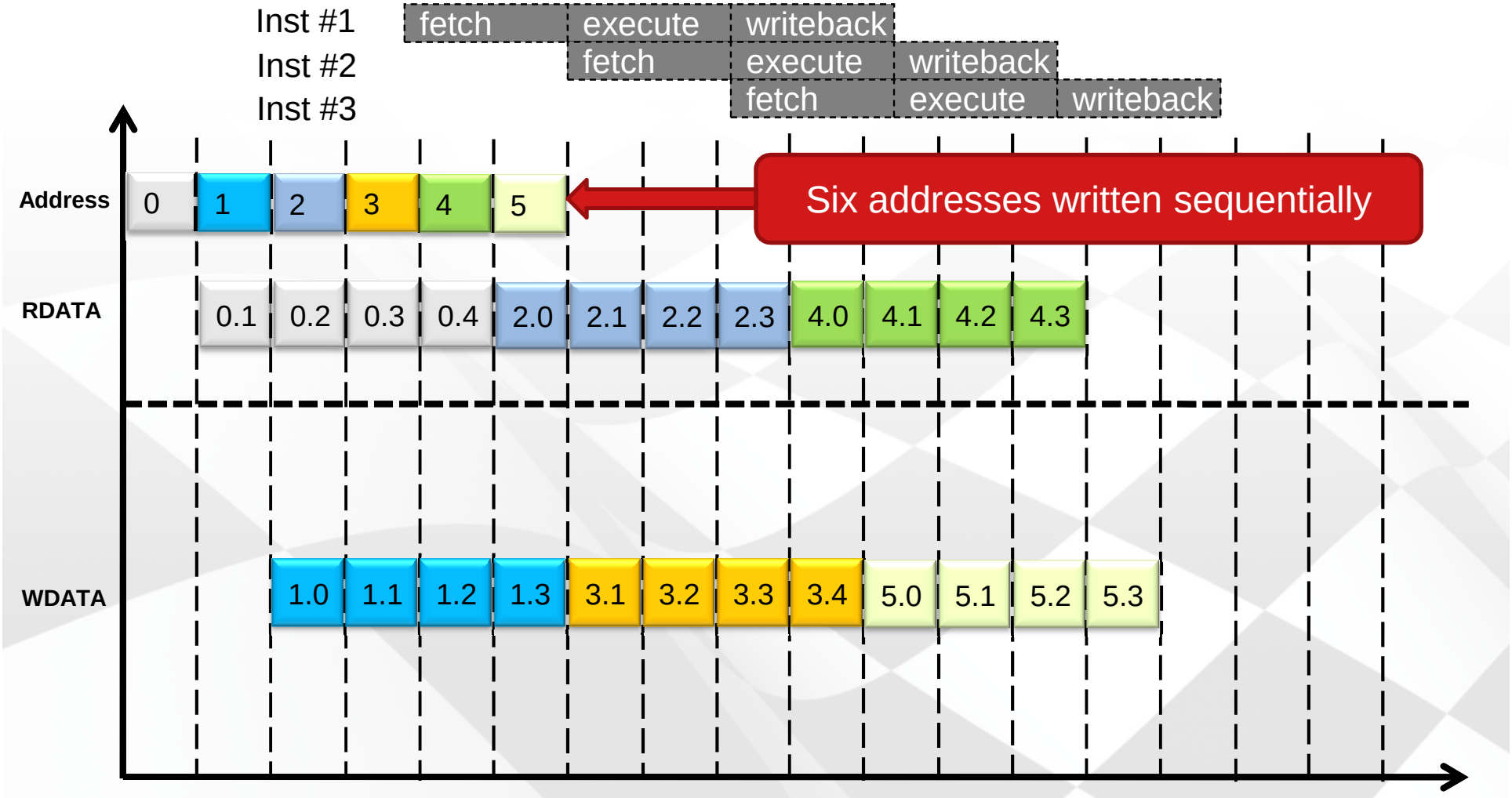
PLBv46 has *single address*, separate data channels

AXI Has Separate Read and Write Address Channels



Read and write transactions can be initiated simultaneously

Address Pipelining PLB v46

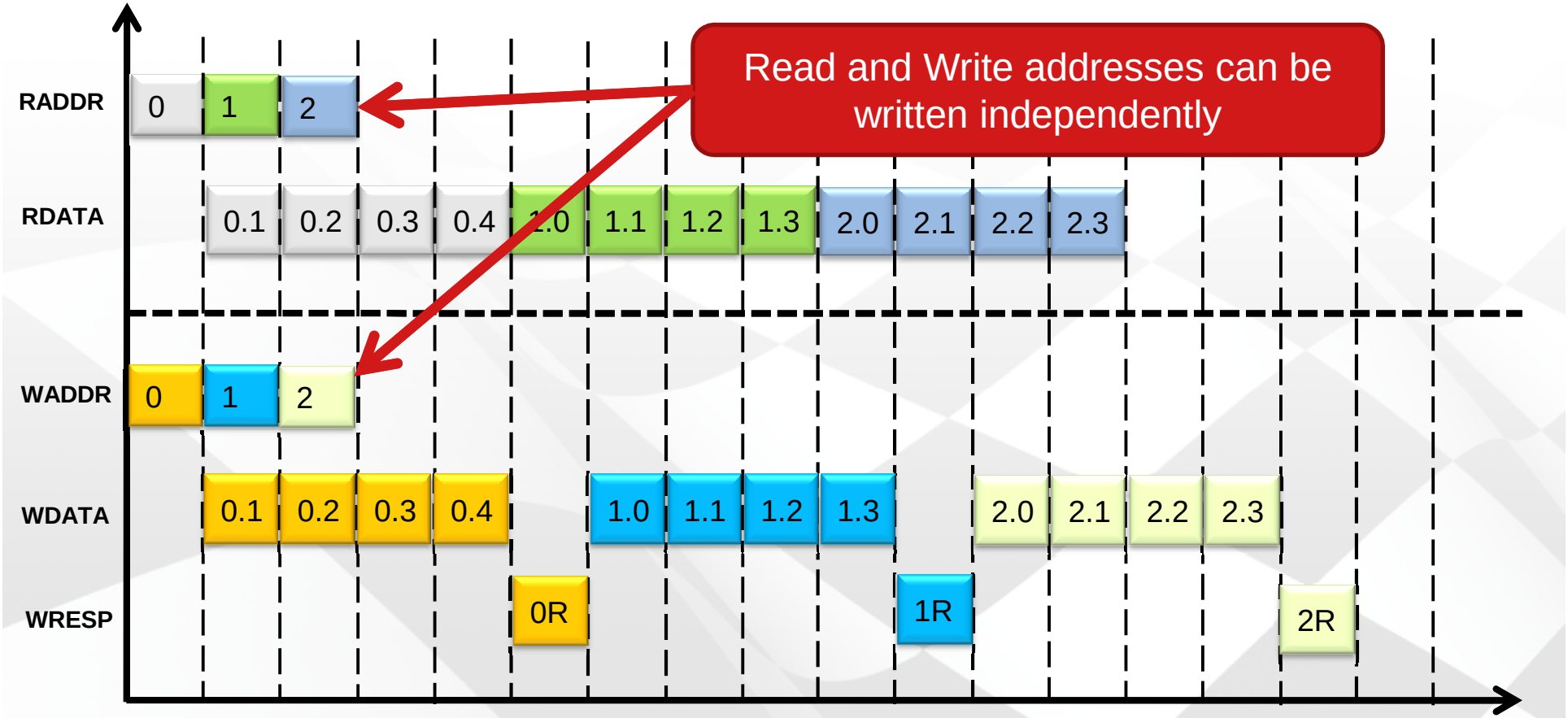


Address pipelining allows initiation of transactions sooner (sometimes referred to as overlapping)

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Address Pipelining Helps AXI as Well

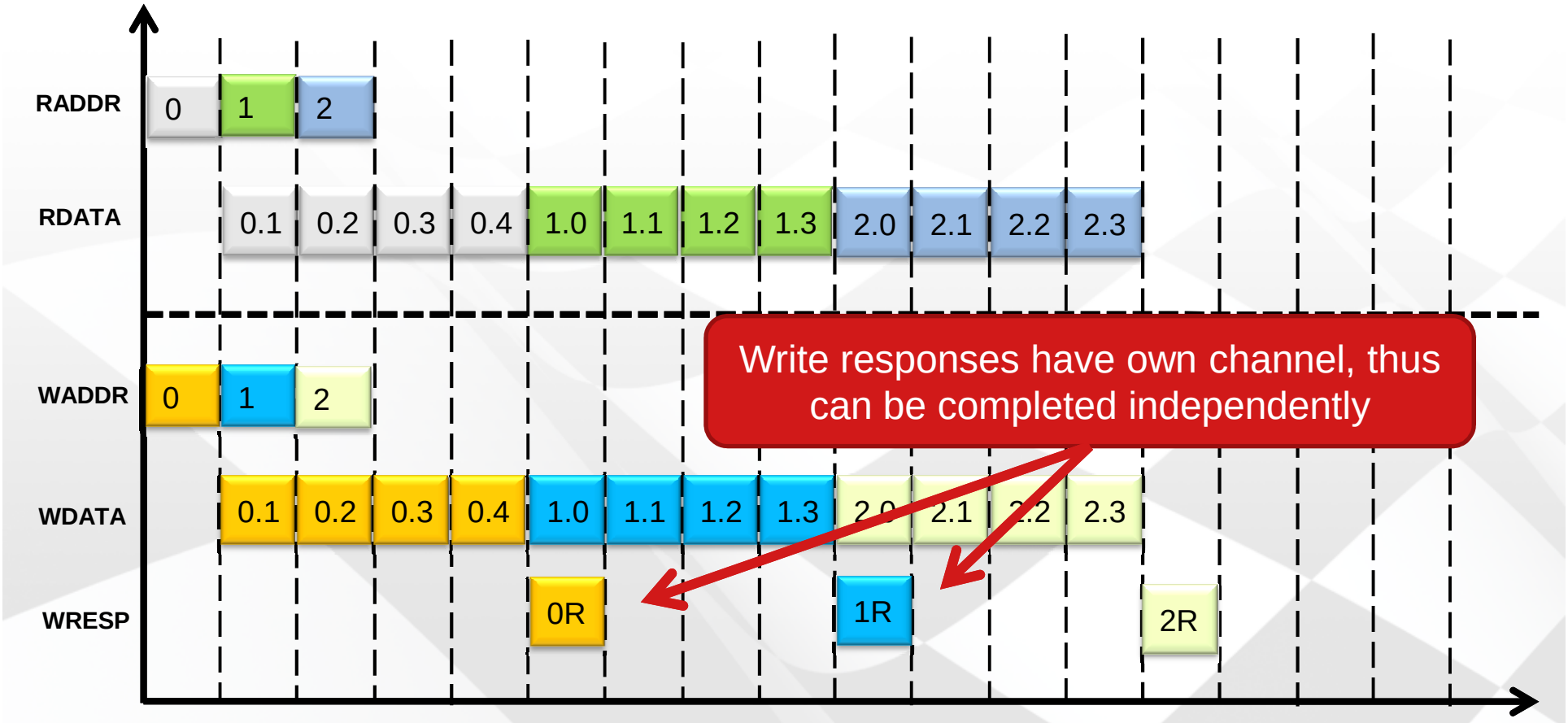


Separate address buses helps pipelining further

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Add Overlapping Transactions for More Efficiencies

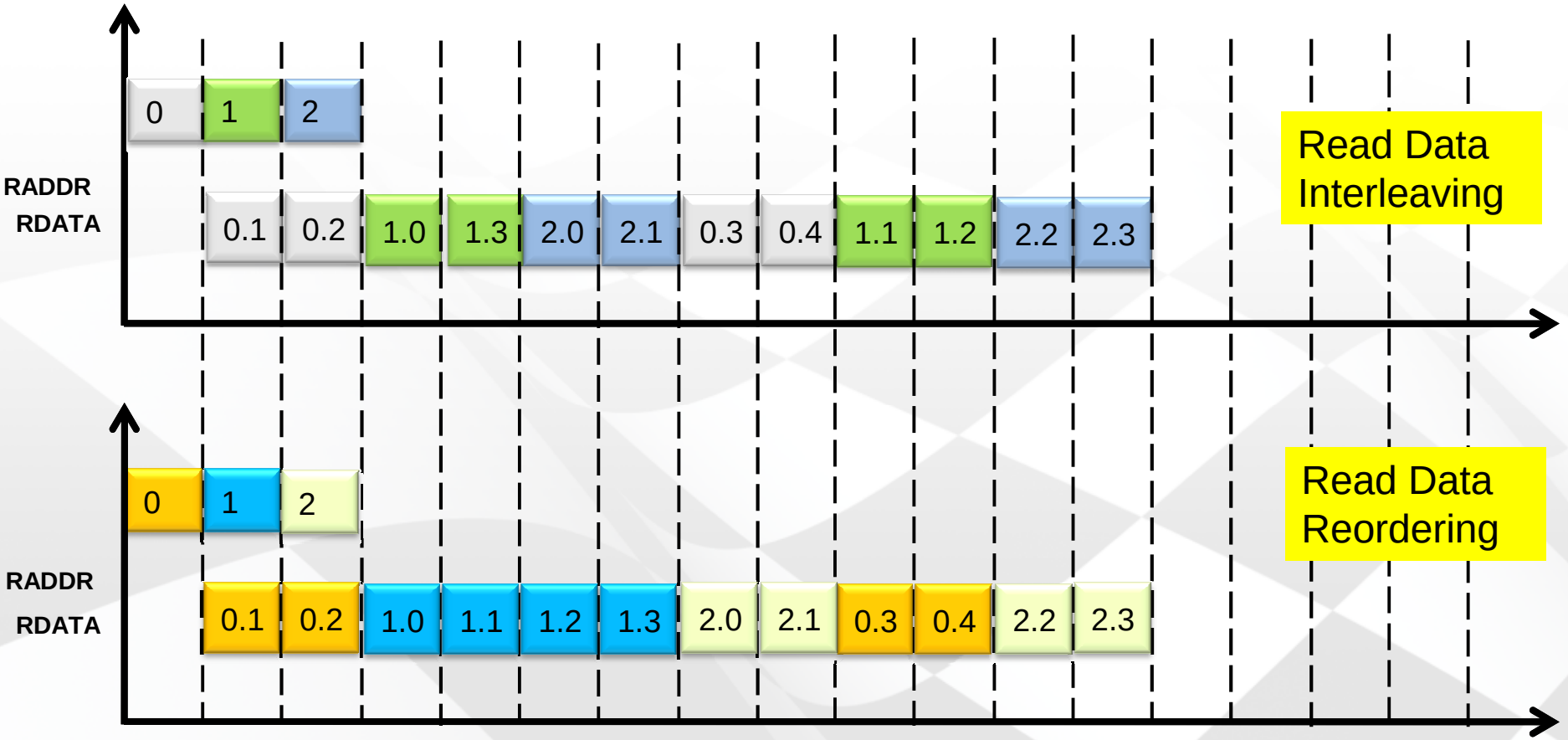


Masters, slaves, and interconnect keep track of transactions using ID bits

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

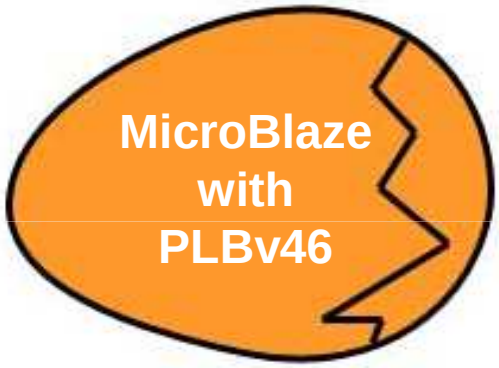
Other “AXI Tricks” for Even More Bandwidth



Interleaving and re-ordering minimize the effects of slow slaves

MicroBlaze is Both Endian
AXI:Little Endian, PLBv46 Big Endian

Little Endian means the little end (the least significant byte) of the multibyte word goes into memory first. First here means the lower memory address. Big Endian is just the opposite.



BIG ENDIAN - The way people always broke their eggs in the Lilliput land

LITTLE ENDIAN - The way the king then ordered the people to break their eggs

Endian Byte Ordering Example

	High Address			Low Address
Address	3	2	1	0
Little-endian (AXI)	Byte 3	Byte 2	Byte 1	Byte 0
Big-endian (PLB)	Byte 0	Byte 1	Byte 2	Byte 3
Memory Contents	0x44	0x33	0x22	0x11

AXI
32-bit value in little-endian
0x44332211

PLB
32-bit value in big-endian
0x11223344

Checkpoint!

- **List the various AXI-based system architectural models?**

  AXI4, AXI Lite, AXI Streaming

- **What are the five AXI channels?**

  Read address channel, Read data channel, Write address channel, Write data channel, Write response channel

- **What are the advantages of the AXI protocol over a shared bus model??**

  Flexibility – Selectable interface type (Full, Lite, Streaming)

  Performance – No Arbitration, AXI peripherals do not share a bus

  Performance – AXI has separate Read and Write Channels

  Performance – AXI has configurable Interface widths up to 1024 bits wide

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Summary

- **AXI is a signal interface protocol, not a bus**
- **AXI has separate, independent read and write interfaces implemented with channels**
- **Each AXI channel supports a valid/ready acknowledgement handshake**
- **AXI supports bursts and overlapped transactions**
- **The AXI4 interface offers improvements over AXI3 and defines**
  - Full AXI memory mapped
  - AXI Lite
  - AXI Streaming
- **The definition of the AXI interconnect is independent of the AXI interface protocol and determined by the system architecture design**

Adding Custom IP via IPIC

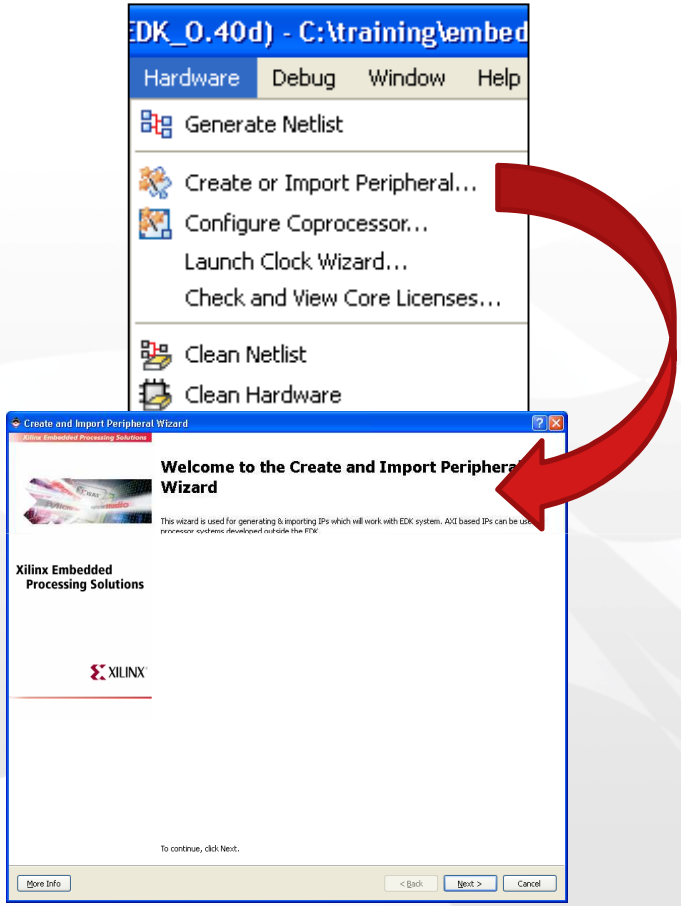
- **IPIC – IP Interface Controller**
- **The IPIC is generated from a set of VHDL sources**
  - Master and slave attachments that have already been described
  - Read FIFO, write FIFO, and interrupt logic that provide additional functions
- **The VHDL is parameterized via generics for conditional code generation and various bus widths and features**
  - Numerous parameters
  - Complex VHDL environment; source code is supplied
- **More signals defined than you may need for your design**

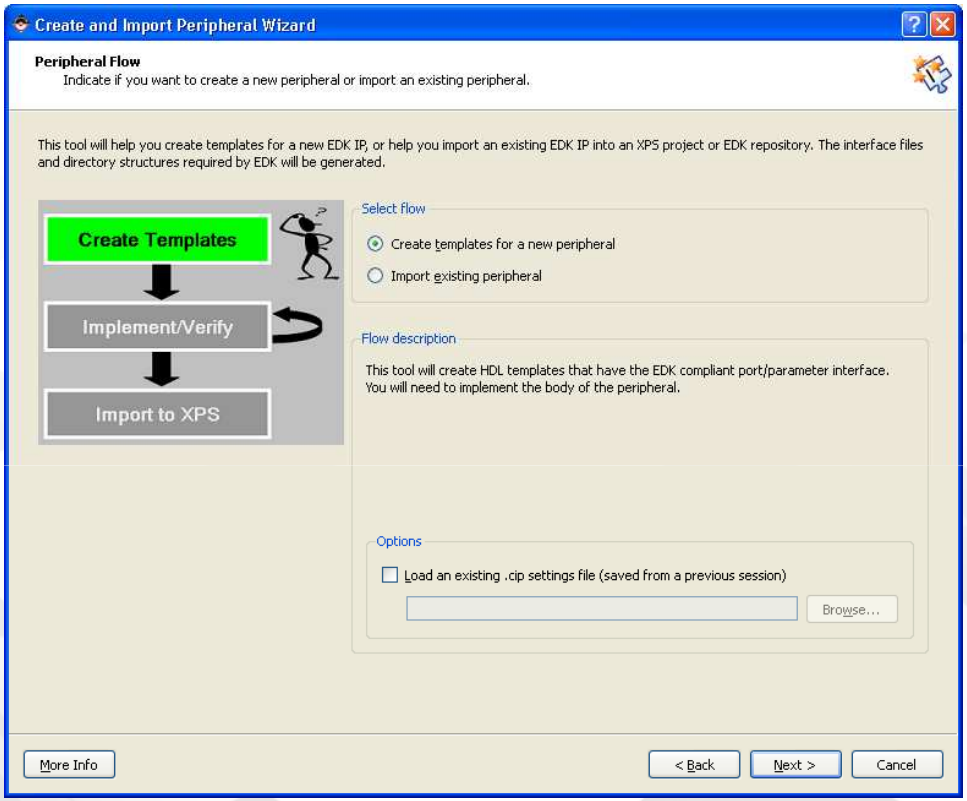
- **All of the above is abstracted away by the Create or Import Wizard**
  - Reduces the time to configure the IPIC to minutes

Create or Import Wizard

- **The wizard helps you create your own peripheral and then import it into your design**
- **The wizard generates the necessary core description files into the user-selected directory**
- **You can start the wizard after creating a new project or opening an existing project in XPS**
- **An existing peripheral can be imported directly through the wizard by skipping the creation option**
  - Ensure that the peripheral complies with the Xilinx implementation of the IBM CoreConnect bus architecture standard
  - Used when updating peripherals from earlier tool versions

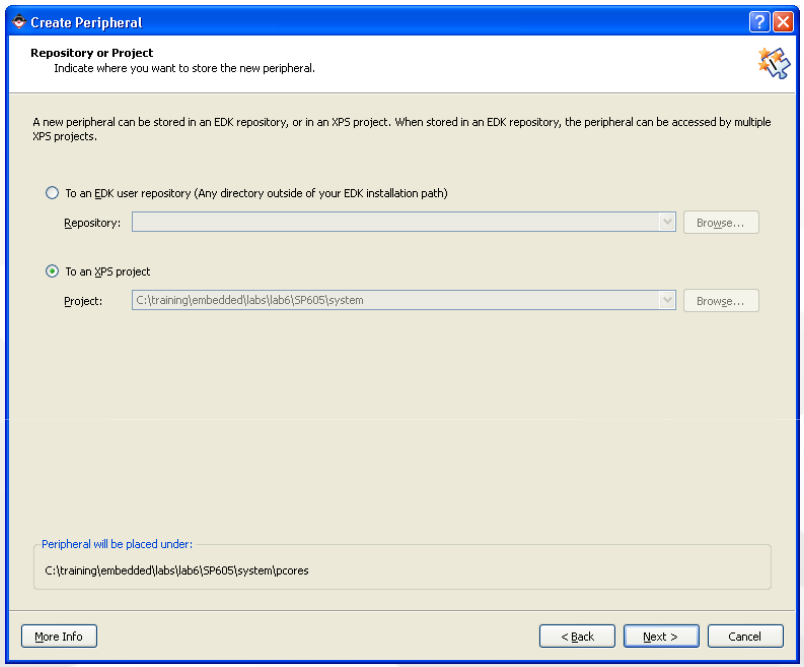
Create or Import Wizard

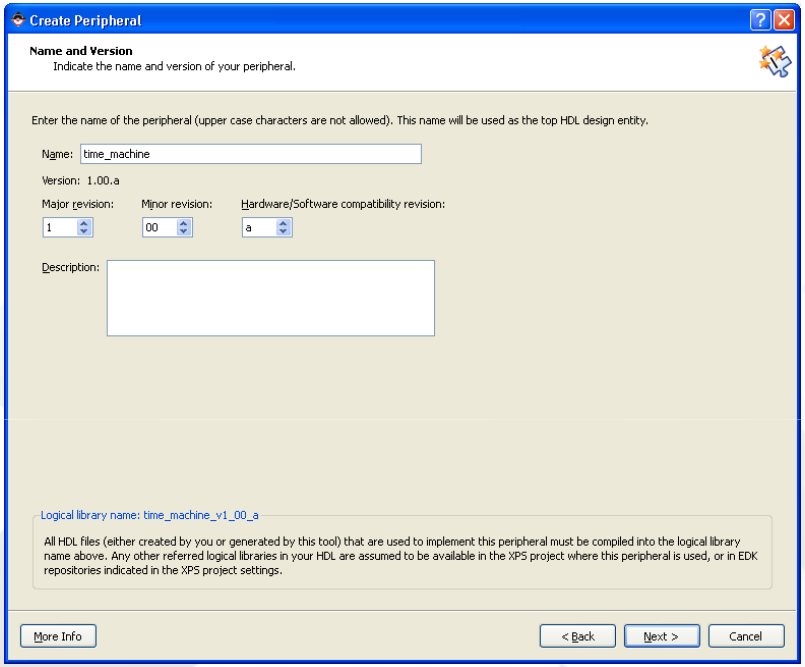




- **Two wizards in one, let's look at the Create IPIC Template Wizard**

Create or Import Wizard





Select to add to existing project or centralized repository

Enter peripheral name, version information, and brief description to be used by XPS tools

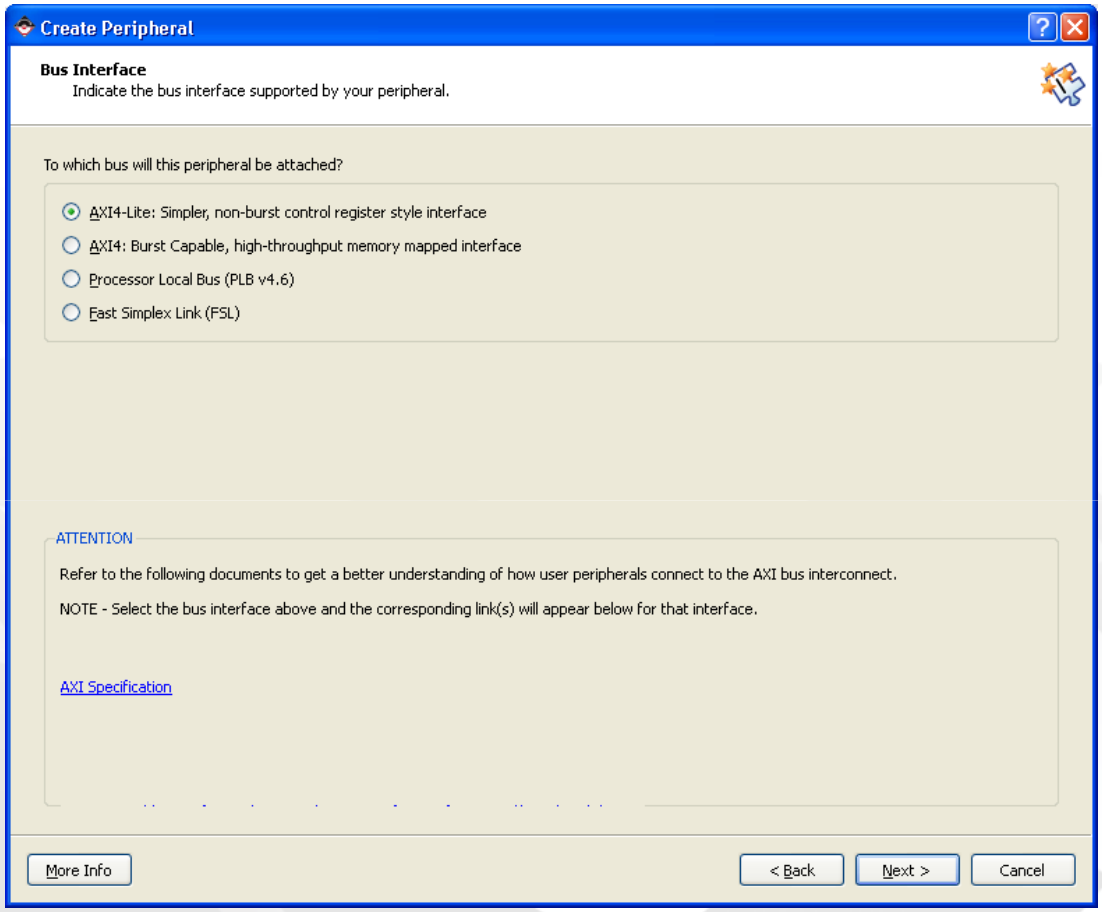
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Create or Import Wizard

Select between AXI , PLB v46, or Fast Simplex Link (FSL)

Hyperlinks to related attachment specifications

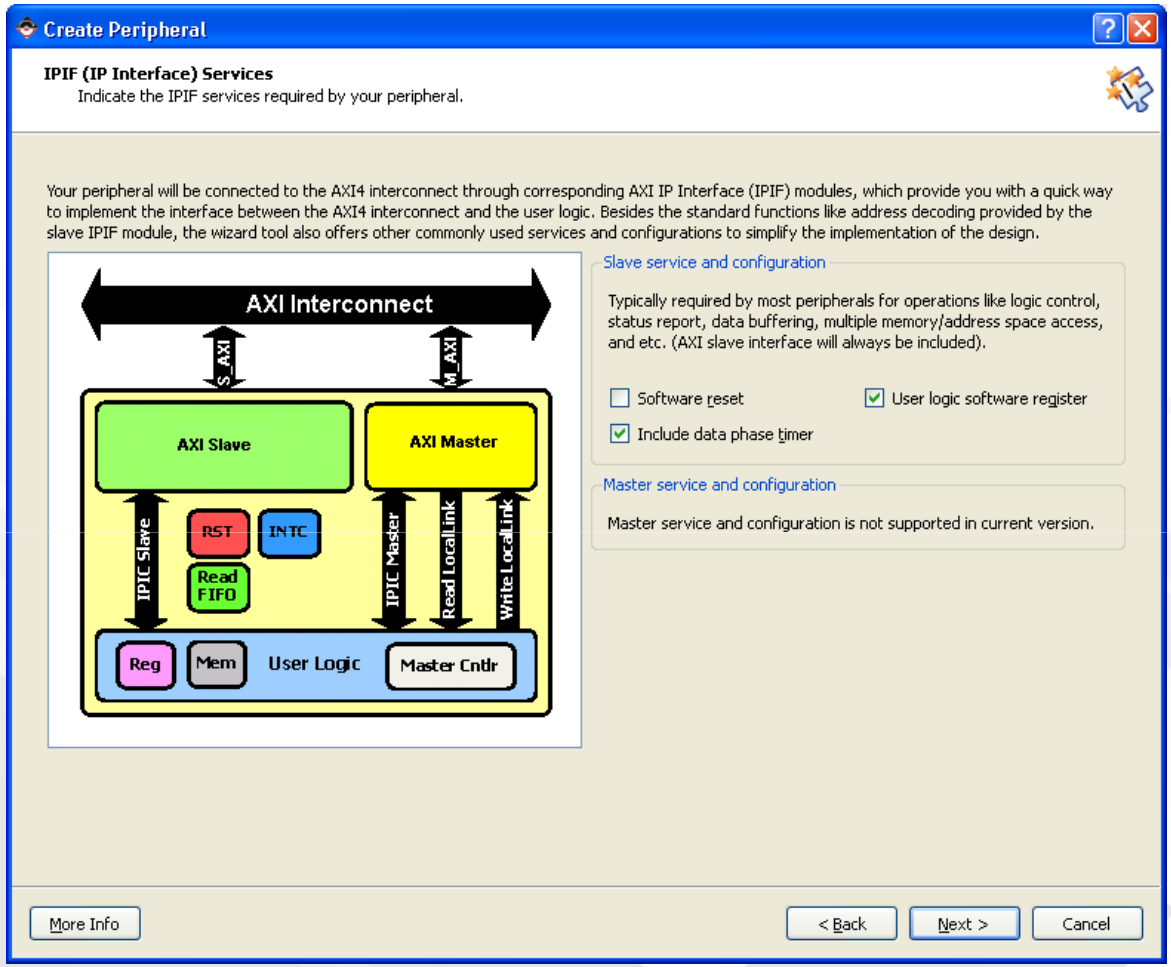


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Create or Import Wizard

Select which optional services to include into peripheral

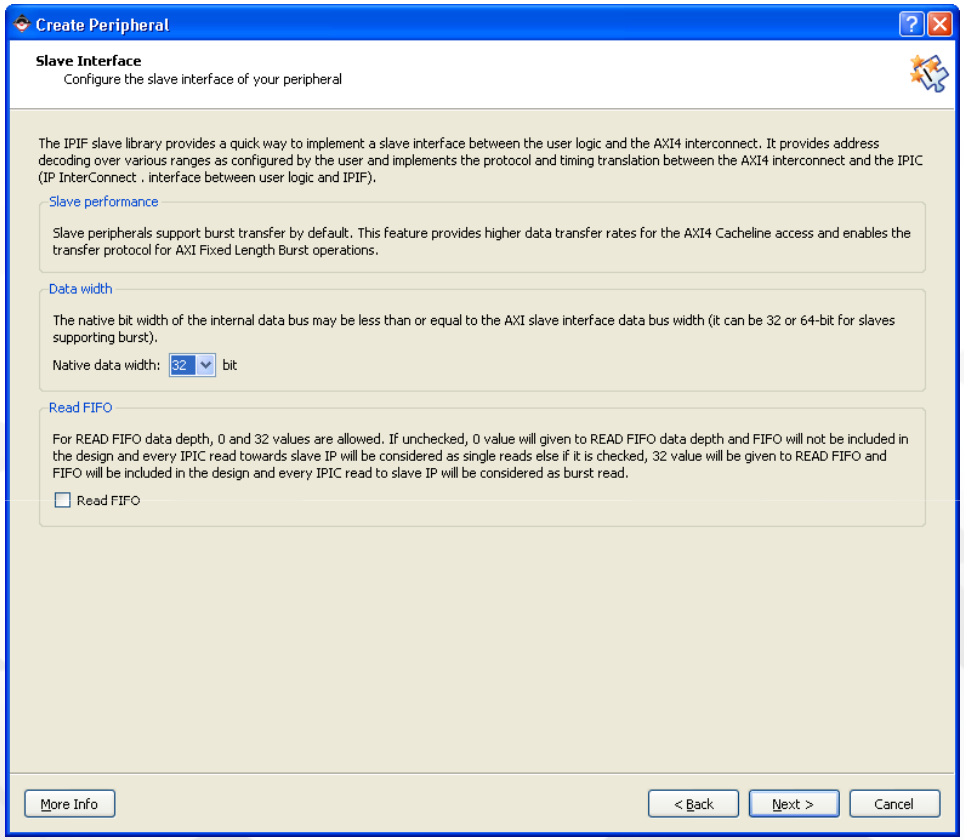


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

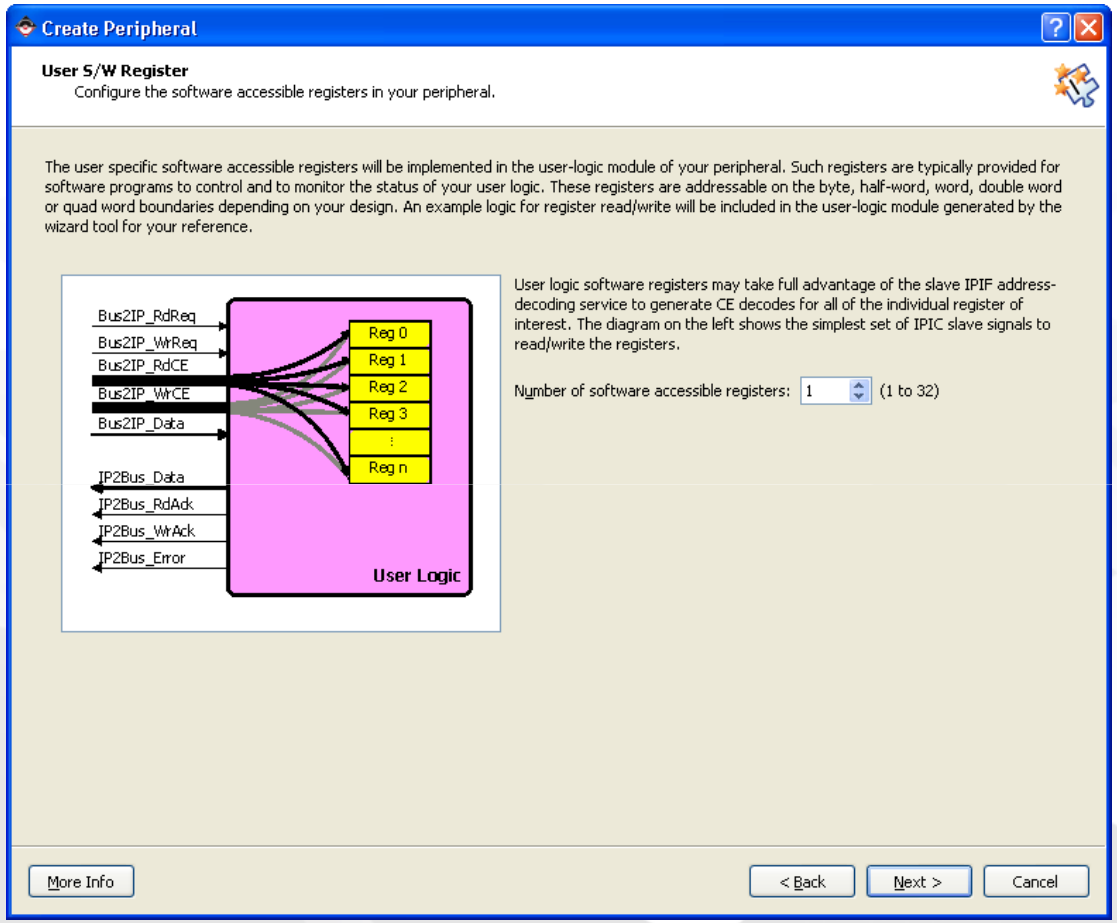
Create or Import Wizard

- **This choice is not seen for the AXI Lite single data phase IPIC**
  - Enable FIFO services



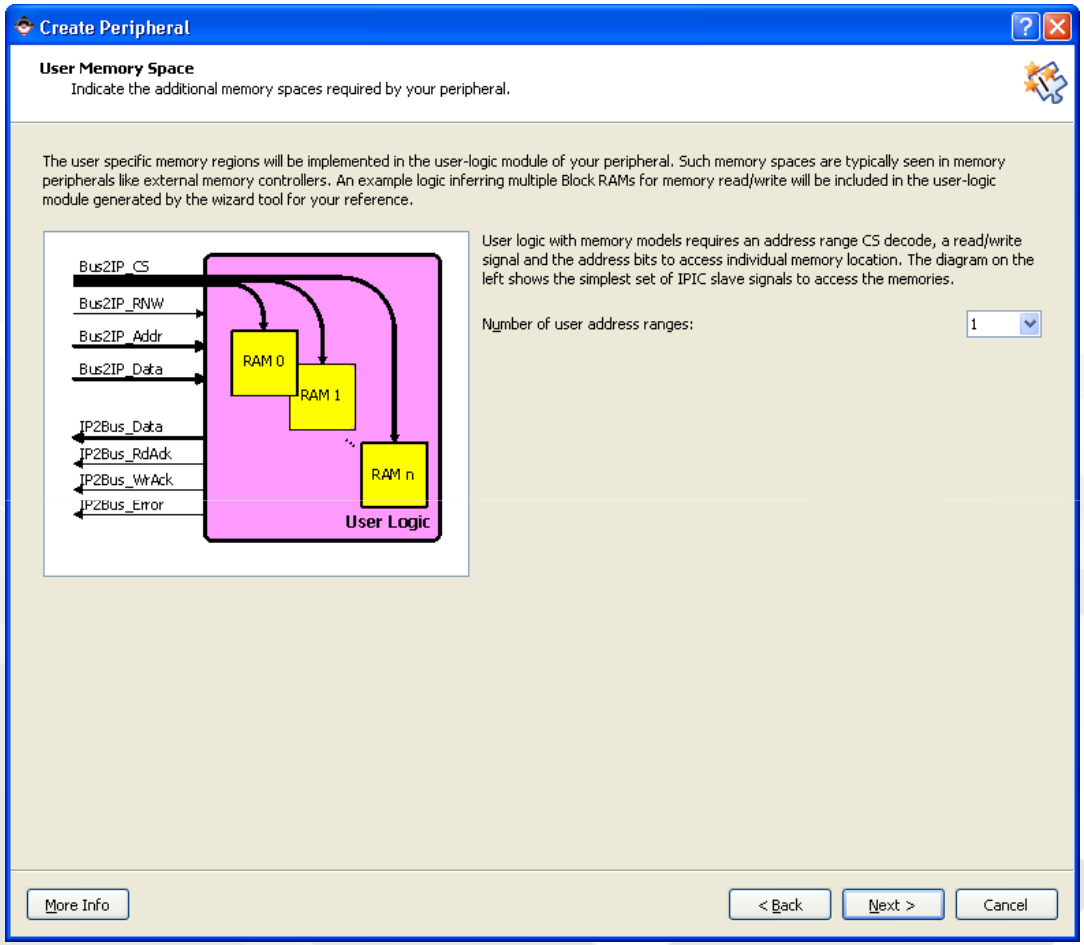
Create or Import Wizard

- **Select number of register enables to provide**

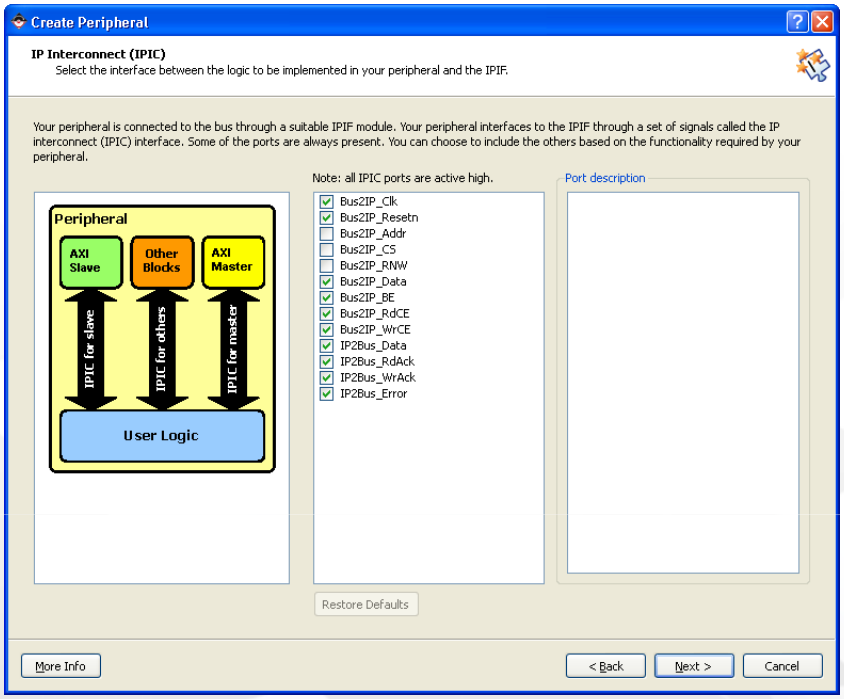


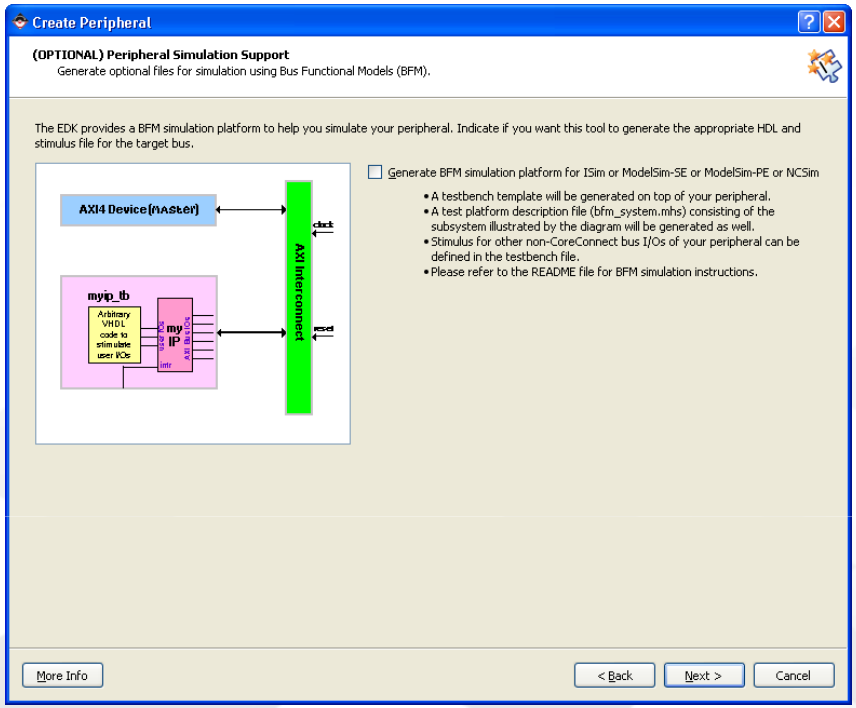
Create or Import Wizard

- **Select number of address ranges to provide**



Create or Import Wizard





Displays which IPIC signals will be provided to the backend user logic

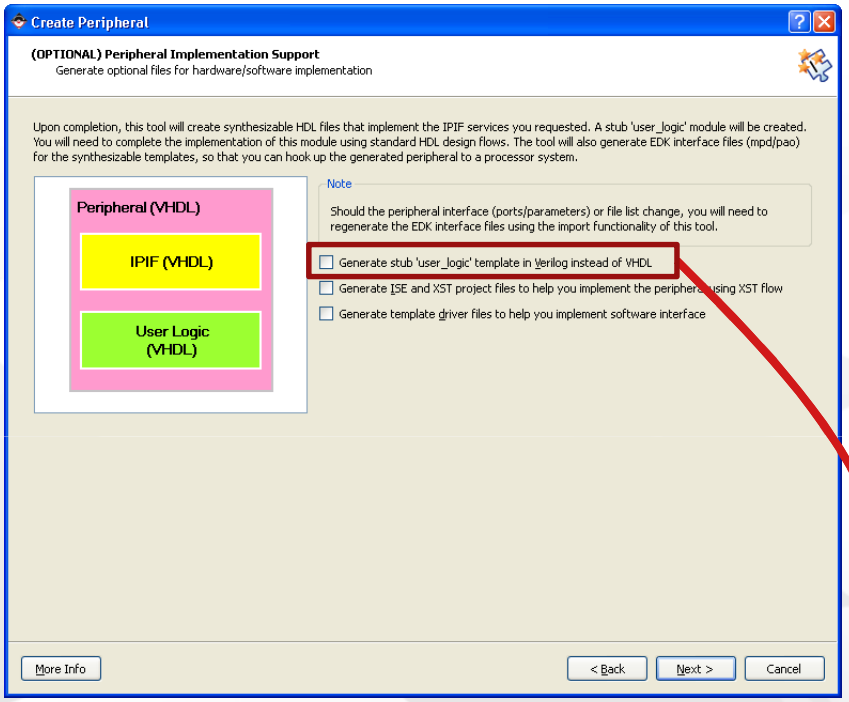
Option to auto-generate a BFM simulation environment

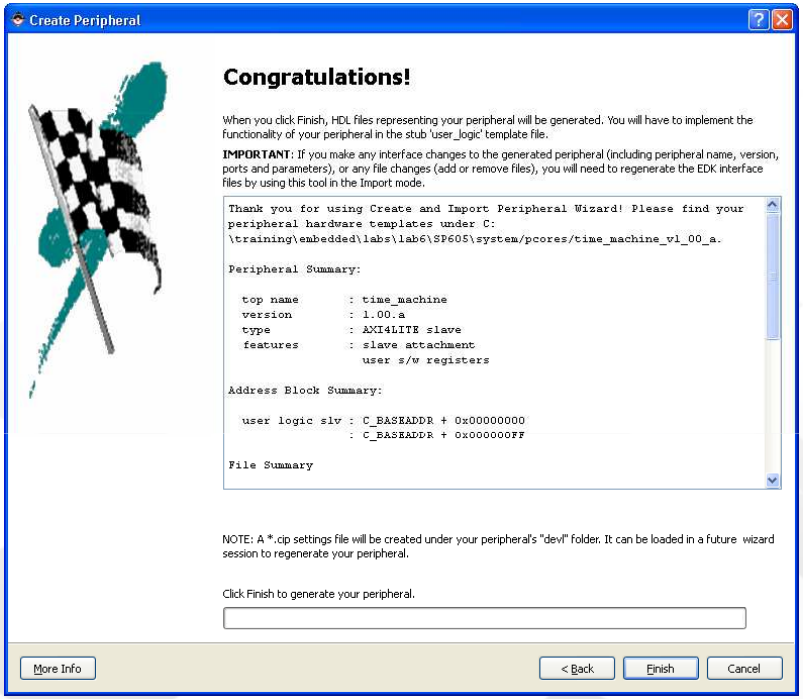
Requires a separate tool license

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Create or Import Wizard





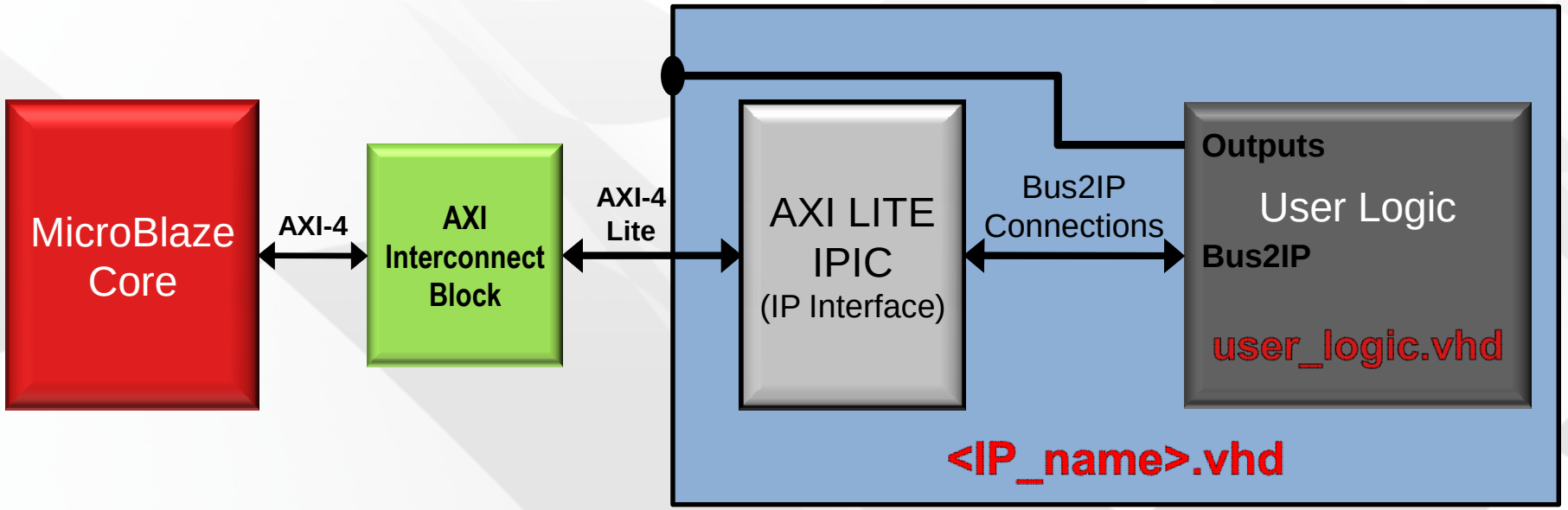
Additional options for development environment, driver template generation, and language options

All done!

Select here for Verilog HDL Template for User_logic

IPIC Connections

- **HDL files created by Wizard**
  - **<IP_NAME>.vhd** is wrapper file containing IPIC and User Logic
    - IPIC abstracts AXI to Bus2IP protocol
    - Requires editing when exporting external ports
  - **User_logic.vhd** is where custom HDL code goes



Inside user_logic.vhd

- **Entity statement**
  - Add custom port signals
  - Add custom generics and parameters
- **Instantiate the rest of the design as a component**
- **Review the sample code provided for each option**
  - Registers implemented for *Bus2IP_WrCE* and *Bus2IP_RdCE* selects
  - Block RAM memory implemented for *Bus2IP_CS*
  - Example code to generate interrupts
  - Example code to transfer data between read/write FIFO
- **Modify/delete code to accommodate your application**
- **Only the needed IPIC signals will appear in *user_logic.v***

Summary

- **The IPIC attachments make it a simple task to interface to the AXI interconnect**
- **Complex IPIC services are configurable via parameters and VHDL generics**
- **The Create or Import Wizard makes it a simple task to configure and create a specific instance of an IPIF attachment**
- **IPIC attachments are supplied as separate master and slave modules**
- **IPIC slave services**
  - Single and burst data phase
  - Address range (CS) and single address (CE) decodes
  - Burst read FIFO
  - Reset and MIR register
  - Data phase timeout timer

Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - **Lab 3 - Adding Custom AXI IP**
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

Lab 3: Adding Custom IP

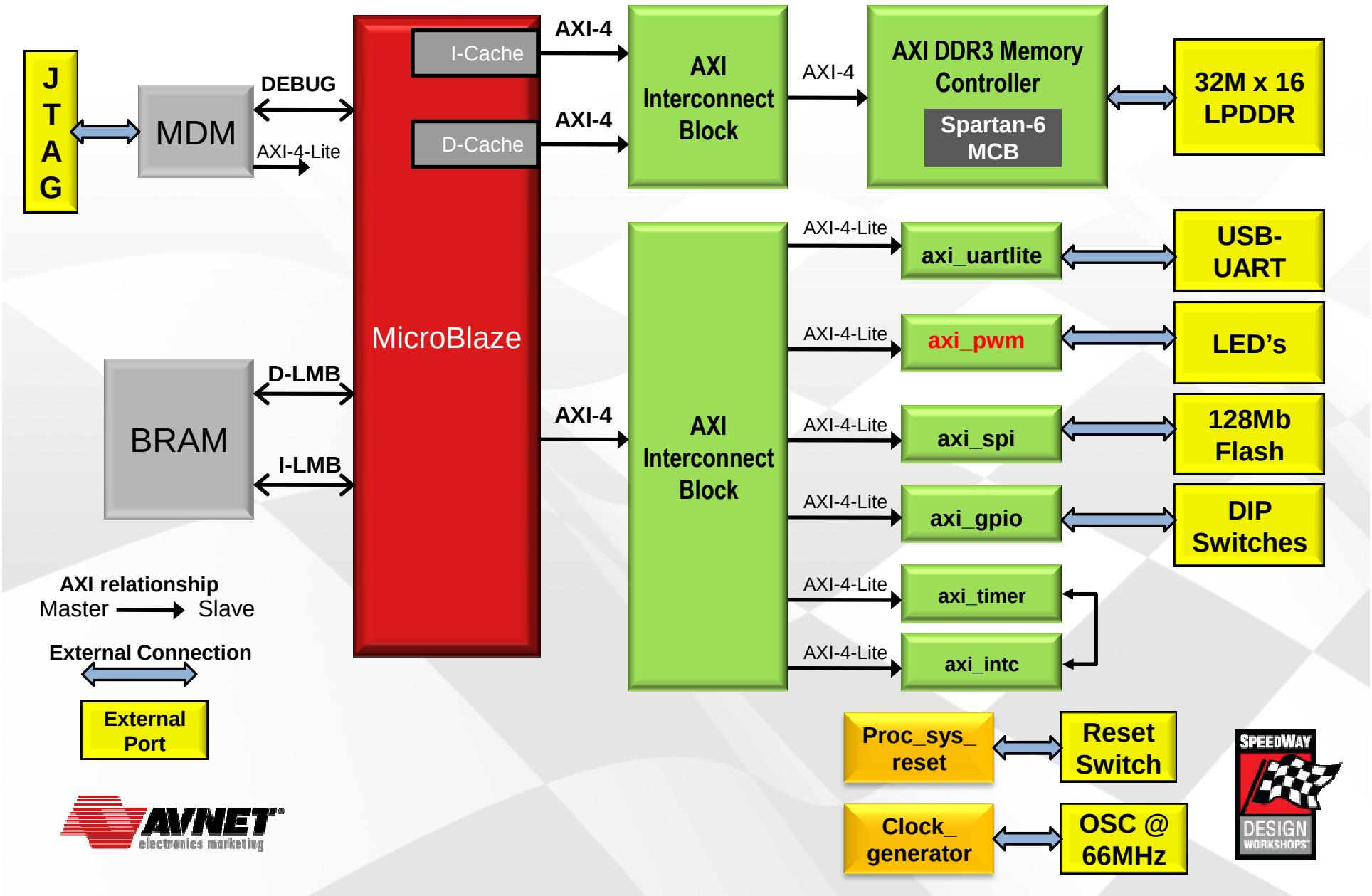
Introduction

- **This tutorial demonstrates how to create and add custom IP to an existing MicroBlaze system using the Xilinx Platform Studio (XPS) Create/Import Peripheral Wizard. The system from the previous tutorial will be used as the starting point**

Objectives

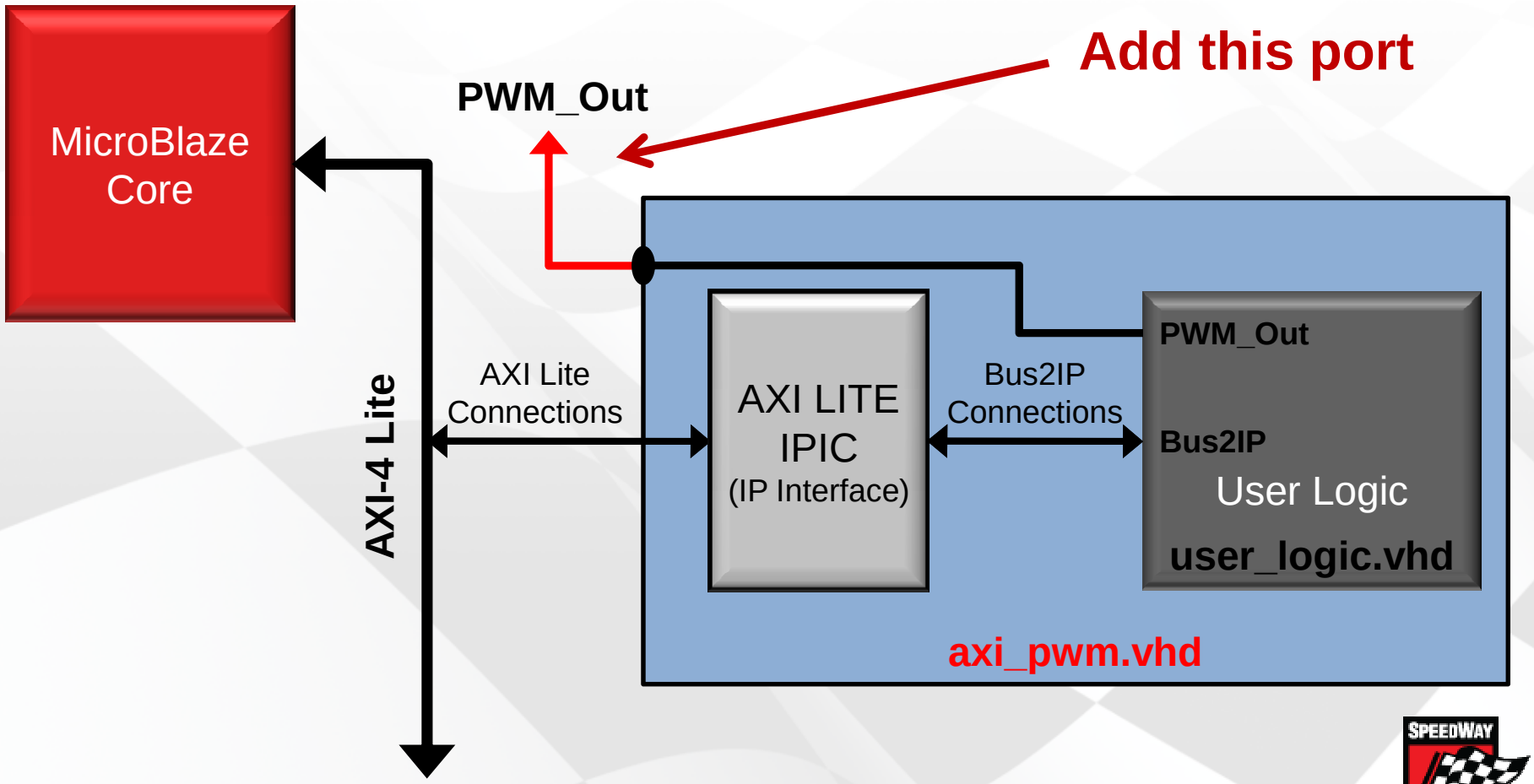
- How to create a custom AXI IP using the wizard
- How to customize the peripheral
- How to add the IP core to the project
- How to write code for the new IP

Lab #3 - MicroBlaze Platform – Adding Custom IP



IPIC Connections

- **HDL files created by IPIF**



Agenda

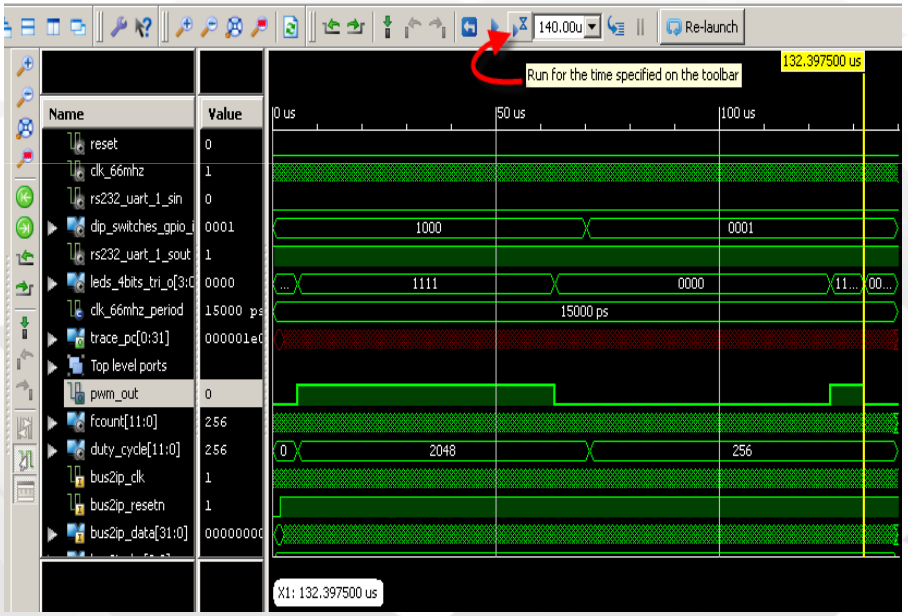
- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

Simulating an Embedded System

- **Your FAE's favorite question???**

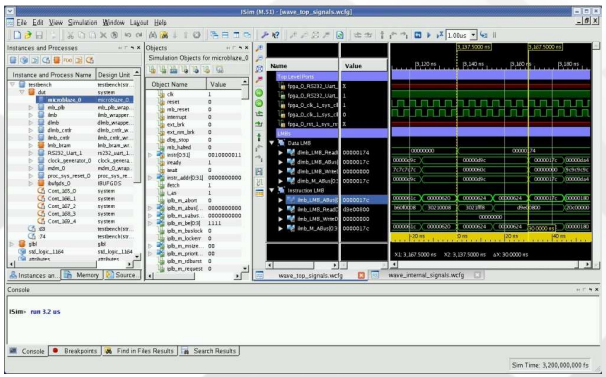
Did you simulate your design first?

- **Simulation can be used to validate:**
  - IP connectivity
  - IP functionality
  - AXI Bus transactions
  - Software execution



ISim – Powerful Mixed-Language HDL Simulator

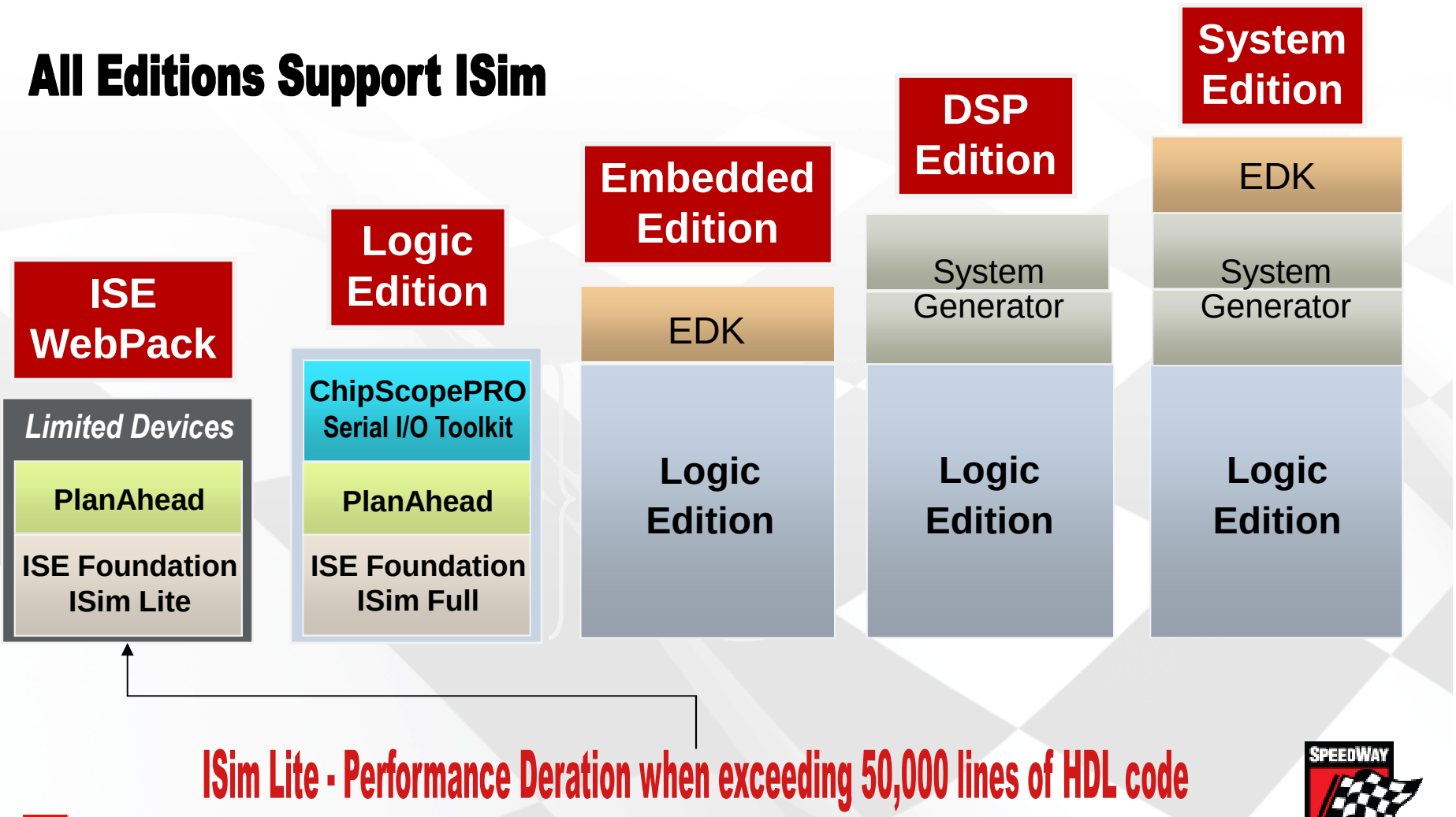
- **Strong compliance to standards**
  - VHDL (IEEE 1076-1993) Verilog (IEEE 1364-2001)
  - Mixed VHDL/Verilog simulation
  - SDF, Tcl, VCD, SAIF
- **Tight integration**
  - Supports all Xilinx devices inluding all Hard IP blocks
  - ISE, EDK, SysGen
- **High productivity**
  - HDL source code debugging
  - Multiple waveform windows
  - Searching, filtering, cross-probing
  - Automatic memory elements detection for editing
  - Recompile and re-launch simulation



AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

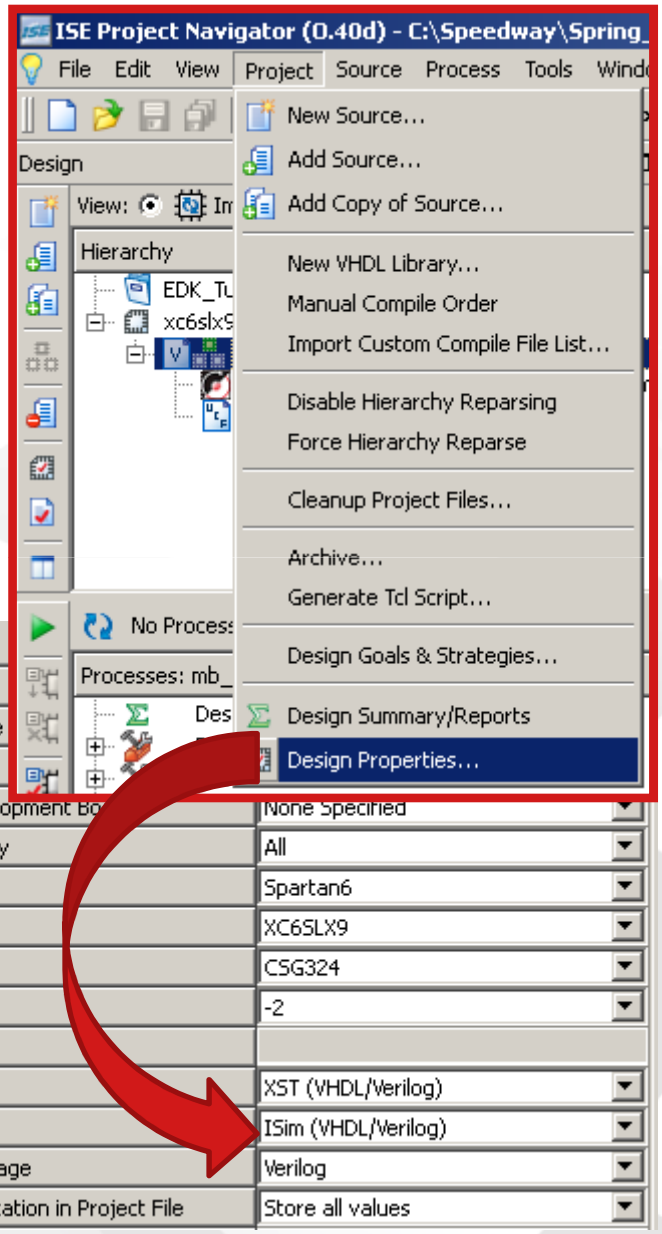
Accessing ISim



Setting up Simulation Environment

- **All simulation preferences set in ISE Project Navigator**
- **Select Preferred Simulator, available:**
  - ISIM
  - Modelsim
  - Questa
  - For Aldec, see:

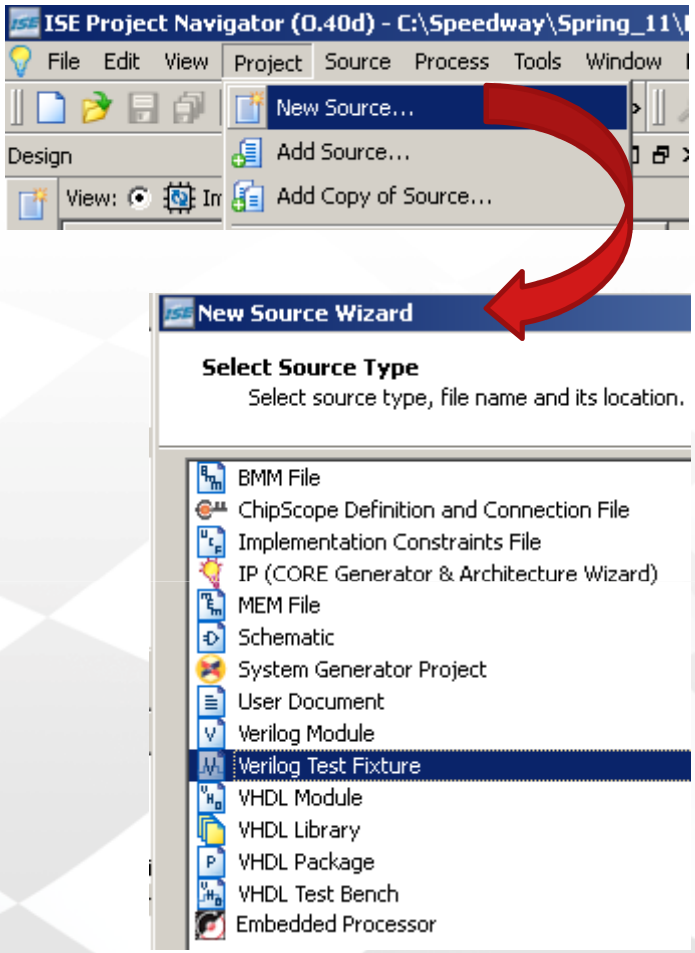
http://support.aldec.com/SupportArchive/PDFs/000820_SimulatingXilinxEDKMicroBlazedesigninActive-HDL.pdf



Creating a Stimulus File

- **First step is to create a stimulus file**
  - Text Fixture (Verilog)
  - Test Bench (VHDL)

- **Project Navigator will automatically create a template**



AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

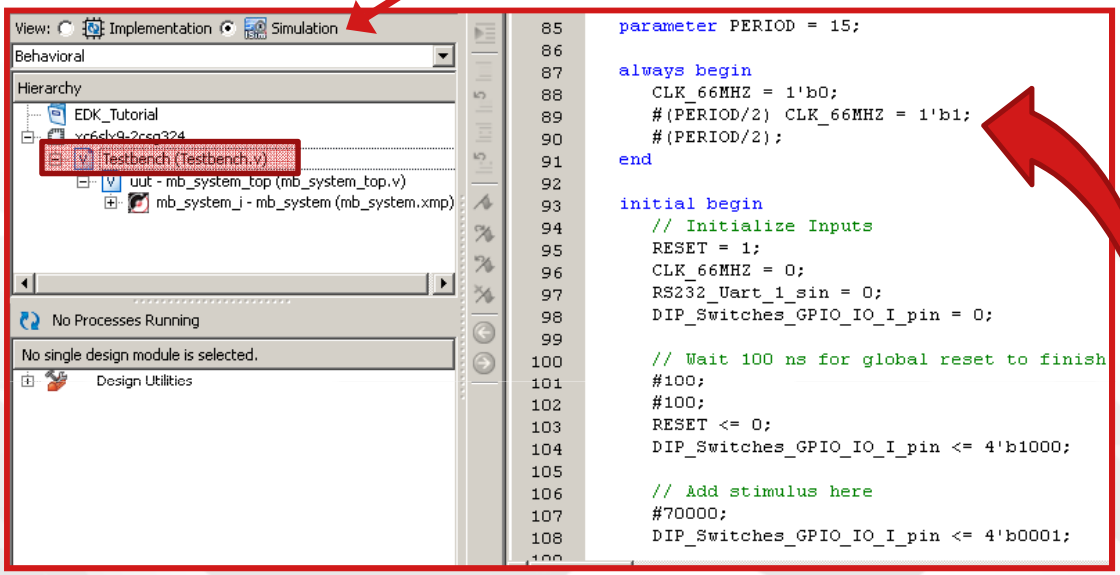
Modifying the Stimulus File

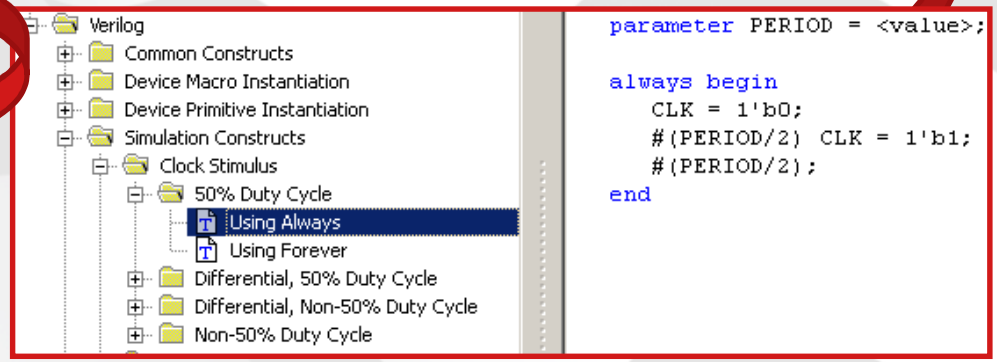
- **MicroBlaze designs only require two inputs… Do you remember what they are?**
  - **Reset**
  - **Clock**

- **Other stimulus includes all system inputs**

- **Templates available for creating stimulus**

In ProjNav, switch View to Simulation

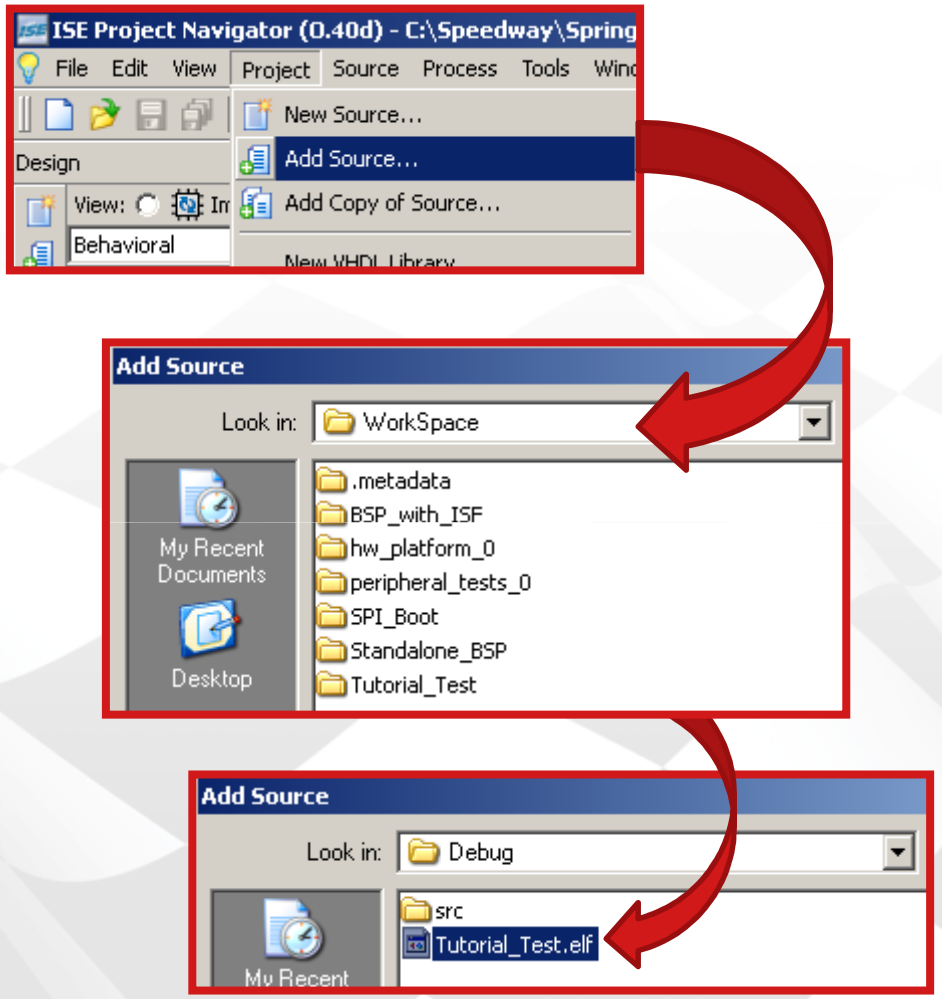




AVNET electronics marketing

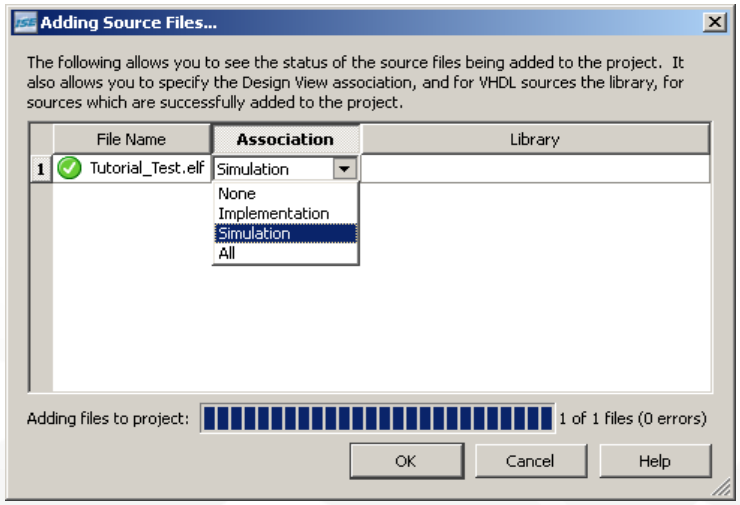
Adding Software Application for Simulation

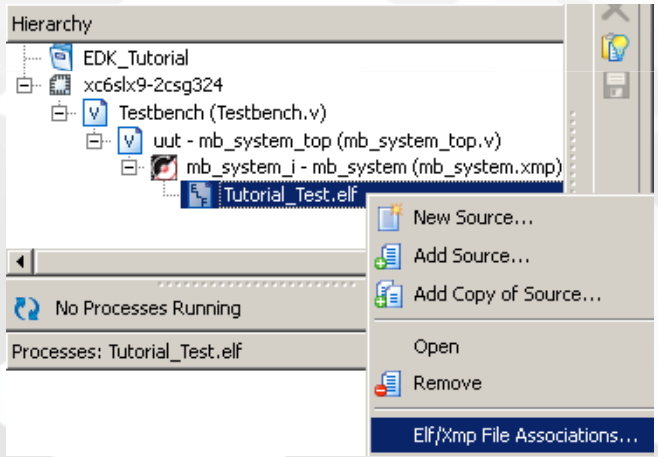
- **ELF files provide embedded code for simulation**
  - ELF file must be built in SDK and **mapped to BRAMs**
  - ELF files exist in SDK WorkSpace directory
- **Add ELF File to project**



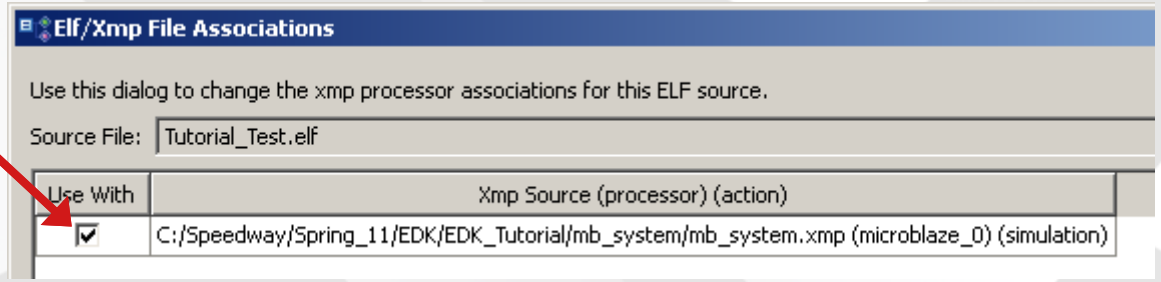
Setting ELF File Associations

- **Associations must be set for simulation**
  - Can be associated with simulation and implementation
    - Simulation uses ELF to execute code in MicroBlaze
    - Implementation embeds ELF file into Bitstream





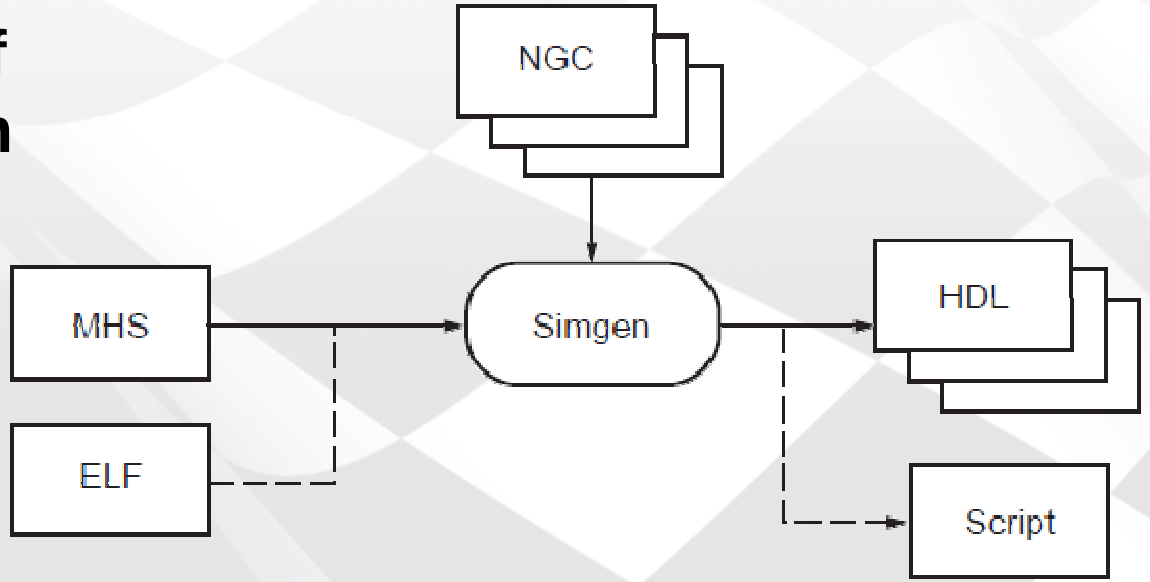
- **Must check Use With box**



Simgen – Simulation Model Generator

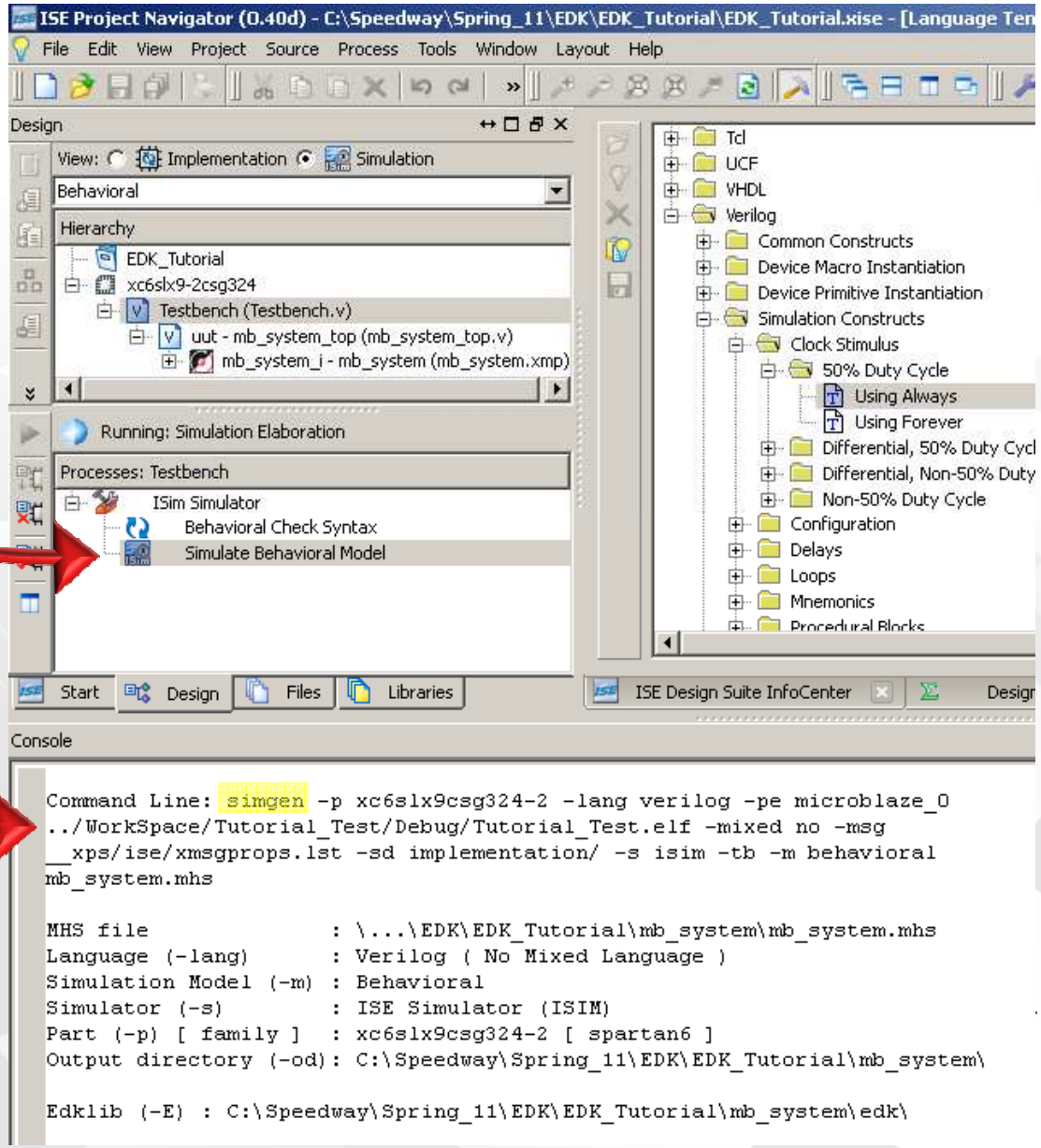
- **Creates structural simulation models used to simulate embedded systems**

- **Inputs MHS and ELF files to create model of embedded system**



Invoking SimGen – Launching ISim

- **Select Testbench**
- **Run Simulate Behavioral Model**
- **Simgen is called**



AVNET electronics marketing

Agenda

- Overview and Xilinx Development Tools Review
- Xilinx MicroBlaze Architecture Overview
- Creating an Embedded Design
  - Lab 1- Adding a Processor to a ISE Design
- Exploring EDK IP Catalog
  - Lab 2 - Adding EDK IP
- AXI Interface Introduction
  - Lab 3 - Adding Custom AXI IP
- Embedded Simulation
  - **Lab 4 – Simulating a MicroBlaze Design**
- Debugging AXI peripherals with ChipScope
  - Lab 5 – ChipScope Debugging
- Using SPI Flash
  - Lab 6 – SPI Programming

Lab 4: Embedded System Simulation

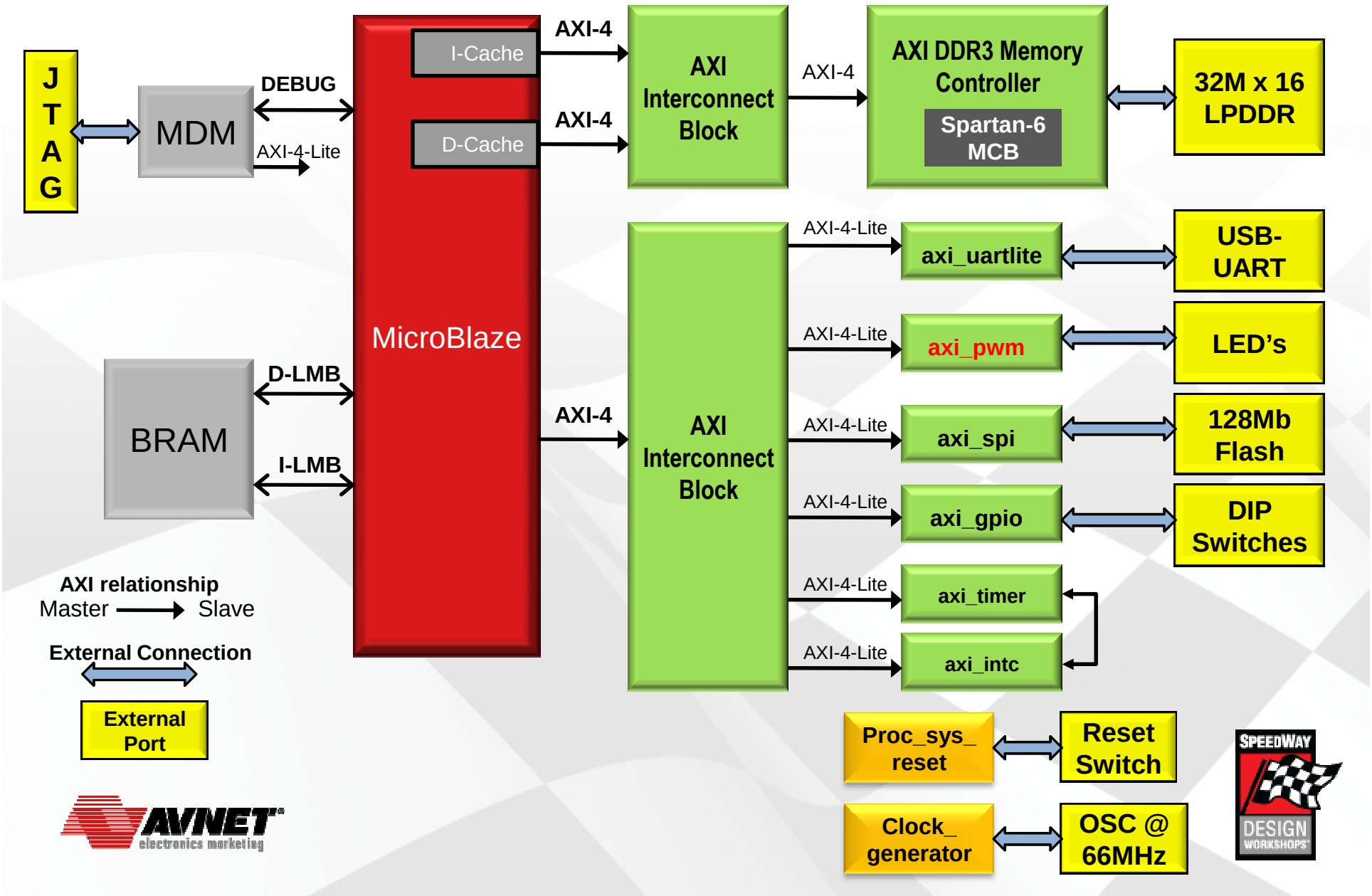
Introduction

- **This tutorial demonstrates how to simulate the embedded system using ISim**

Objectives

- How to setup the simulation properties
- How to add a VHDL testbench
- How to simulate a cycle accurate MicroBlaze design
- How to view MicroBlaze specific signals and AXI bus transactions

Lab #4 - MicroBlaze Platform - Simulation



Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

Debugging your system in Hardware

- **Simulation looked good, but what does it look like in Hardware?**

- **ChipScope™ Pro tool inserts logic analyzer probes and virtual I/O low-profile software cores directly into your design, allowing you to view any internal signal or node, including embedded hard or soft processors.**

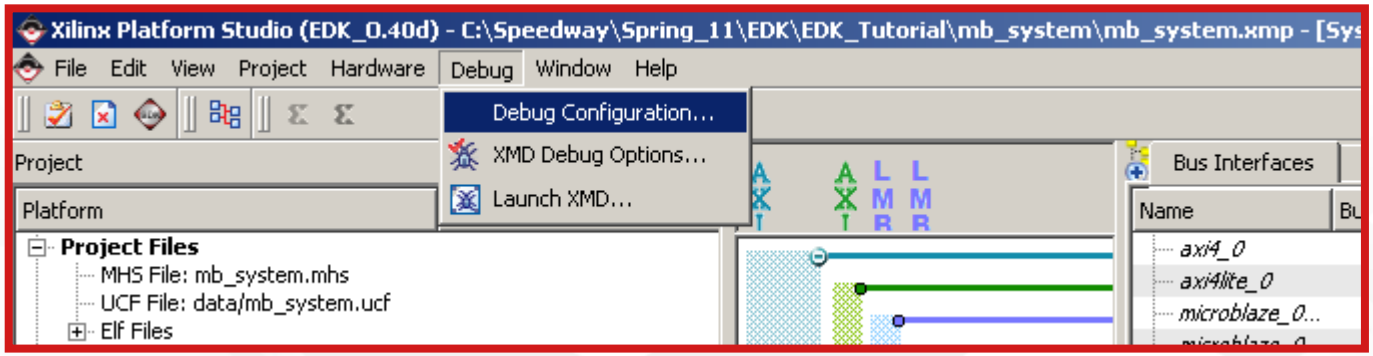
- **Captured signals are then displayed and analyzed using the ChipScope Pro Analyzer tool.**

AVNET electronics marketing

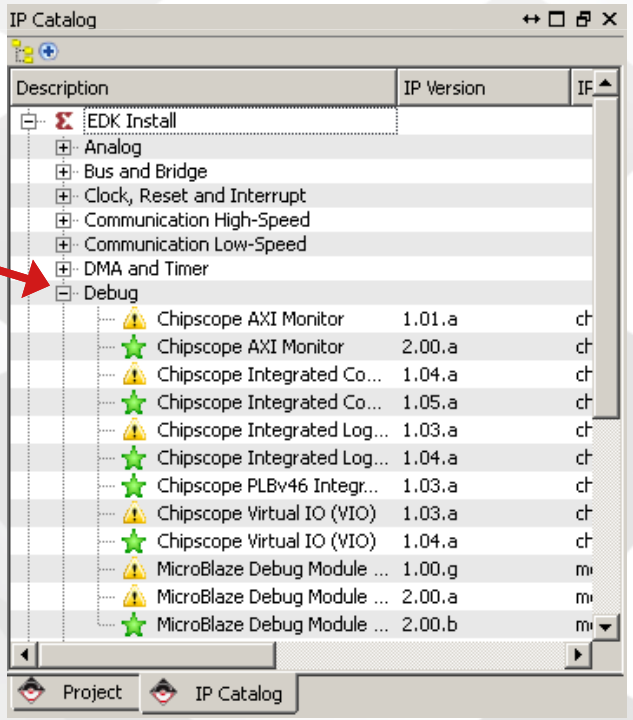
SPEEDWAY DESIGN WORKSHOPS

Inserting ChipScope Pro Cores with XPS

- **Use Debug Wizard – EASY to use!**

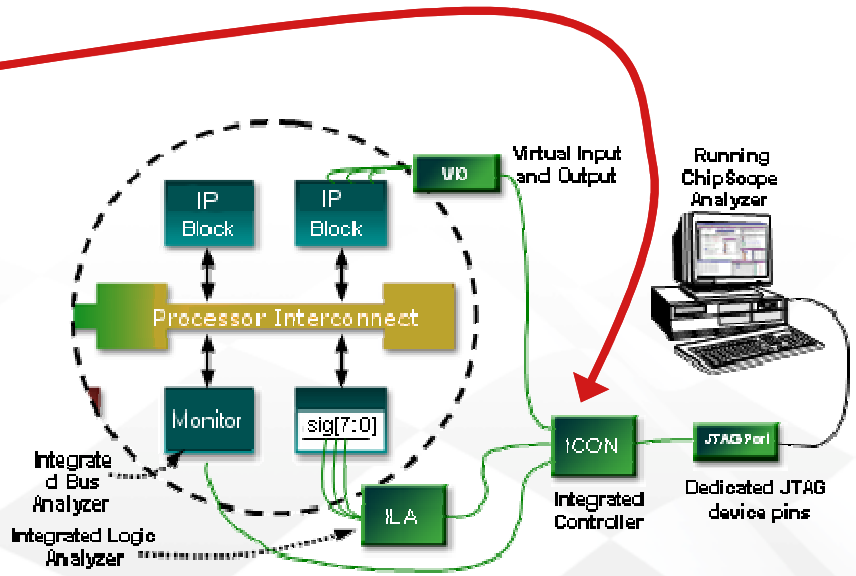


- **Alternatively, Add ChipScope cores via IP Catalog**
  - Will require adding all necessary components
    - **Integrated Controller (ICON)**



ChipScope Core Options

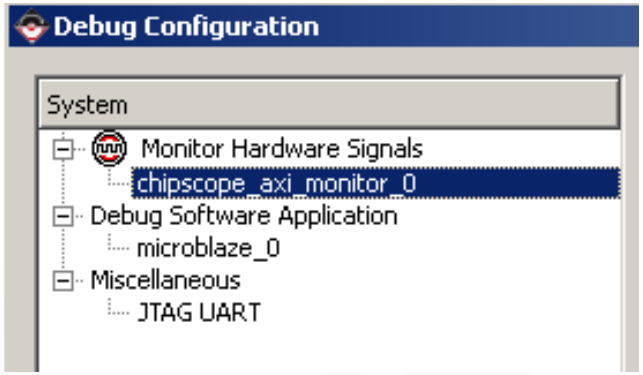
- **ChipScope_ICON**
  - Provides communication between all cores to JTAG
    - Required by all cores
- **ChipScope_ILA**
  - Used to monitor any signals in the design



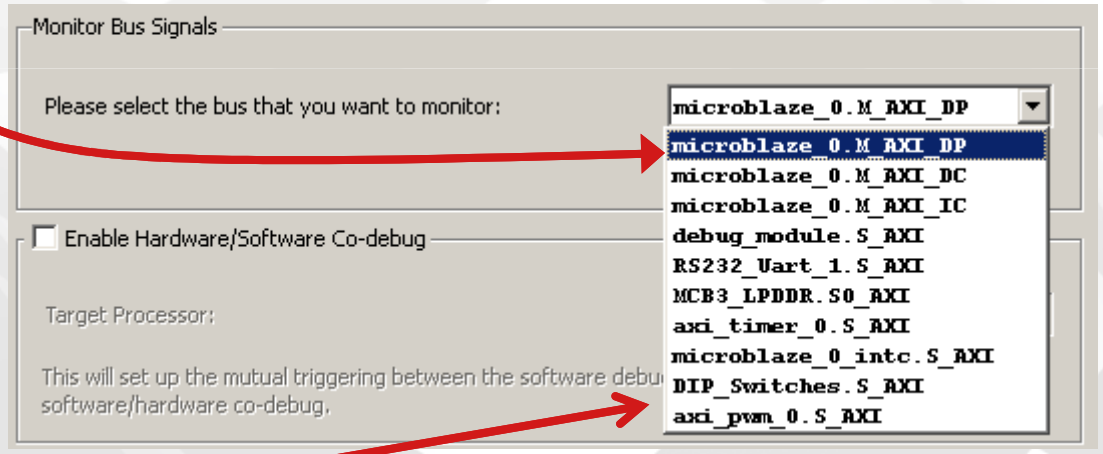
- **ChipScope_VIO**
  - Facilitates use of Virtual I/O, directly reads and writes logic signals at runtime.
- **ChipScope_AXI_Monitor**
  - Monitors AXI Interconnect transactions.
  - Used in today's lab

ChipScope AXI Monitor

- **ChipScope Integrated Bus Analyzer monitor transactions on AXI Interconnect**

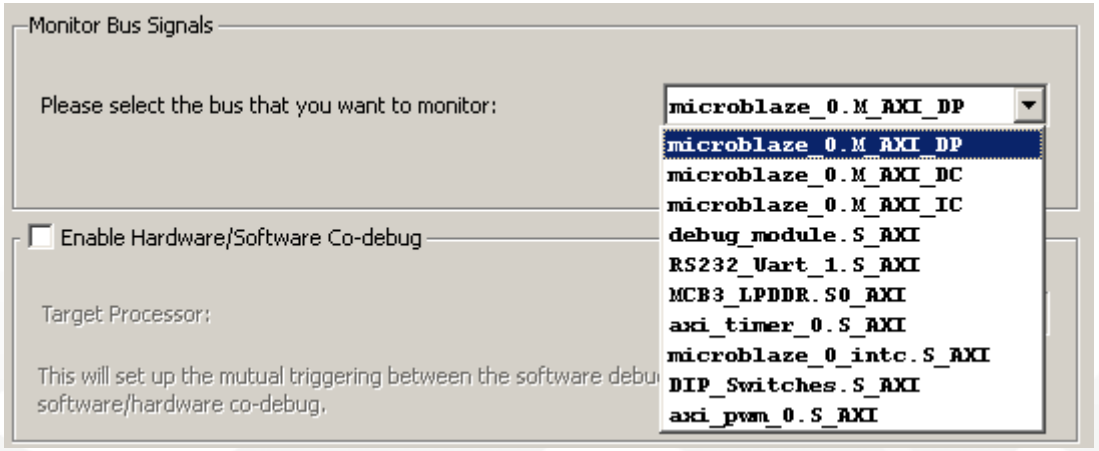


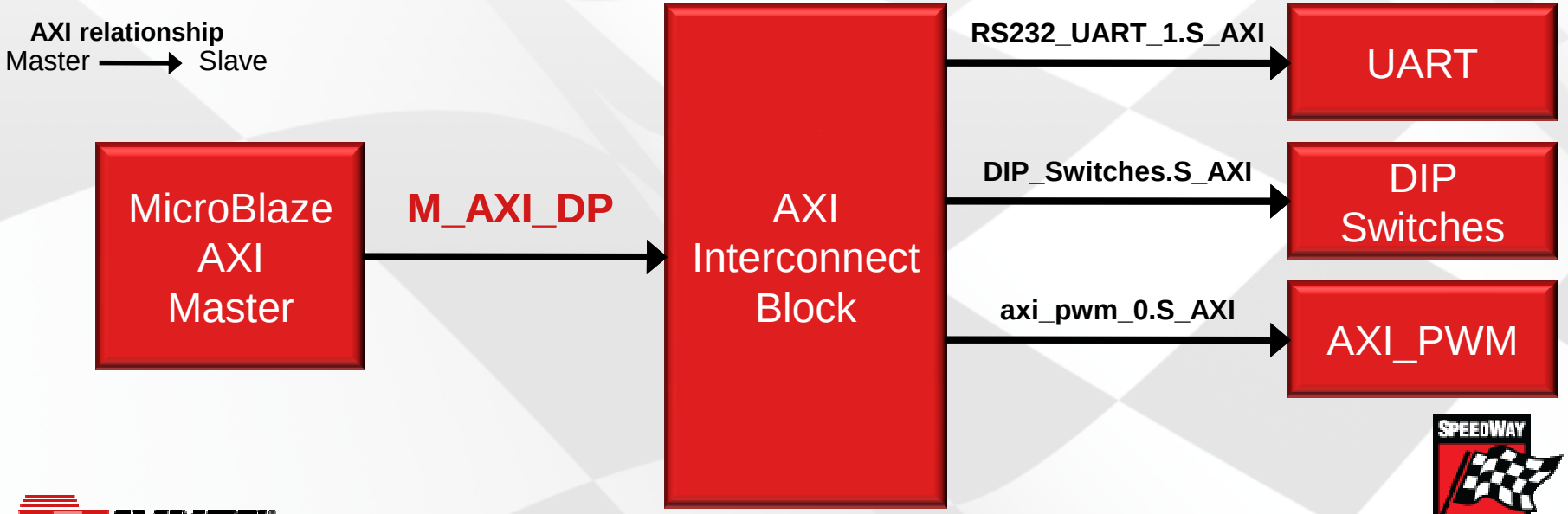
- **Can monitor transactions between all Master/Slave interfaces**
  - **M_AXI_DP** Interface between MicroBlaze and Peripheral AXI Interconnect Block
  - Including signals between Interconnect Block and each peripheral.



AXI Monitor Interfaces

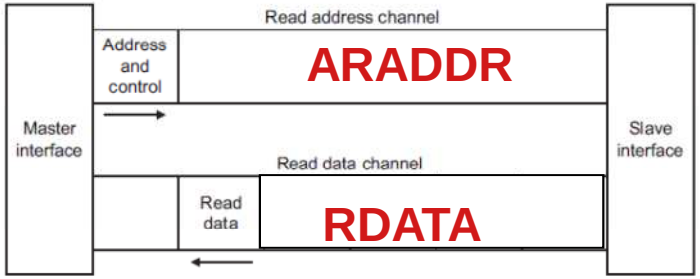
- **AXI has independent connections for each Master/Slave pair**
- **All slaves will have their own interface**



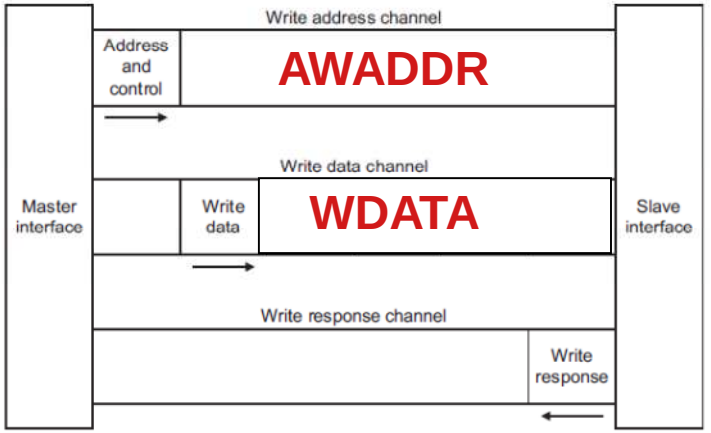


AXI-4 Lite Bus Review

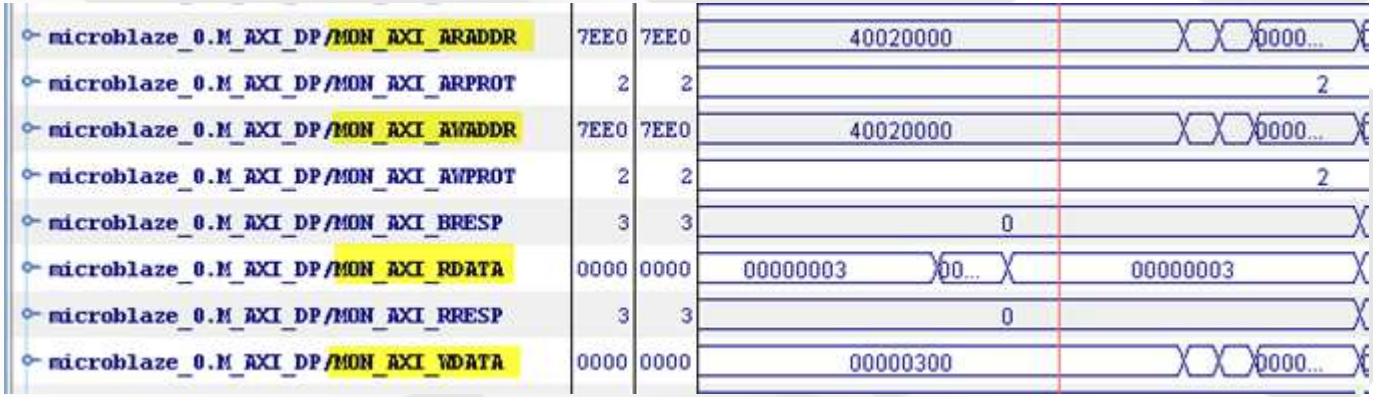
- **AXI has separate read and write channels**
- **Transactions only pass one word of data**



AXI4-Lite Read



AXI4-Lite Write

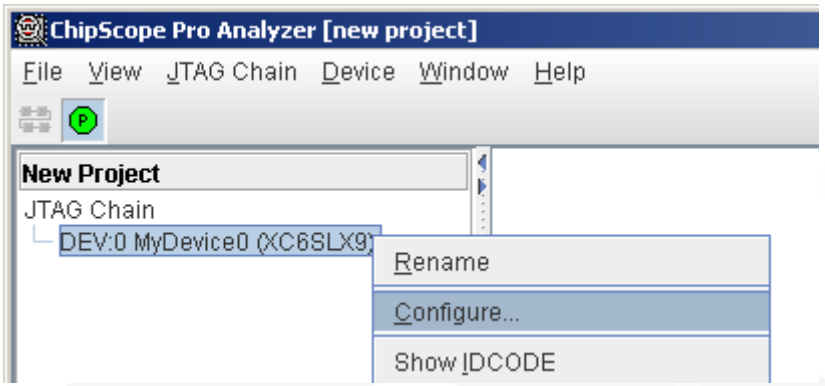


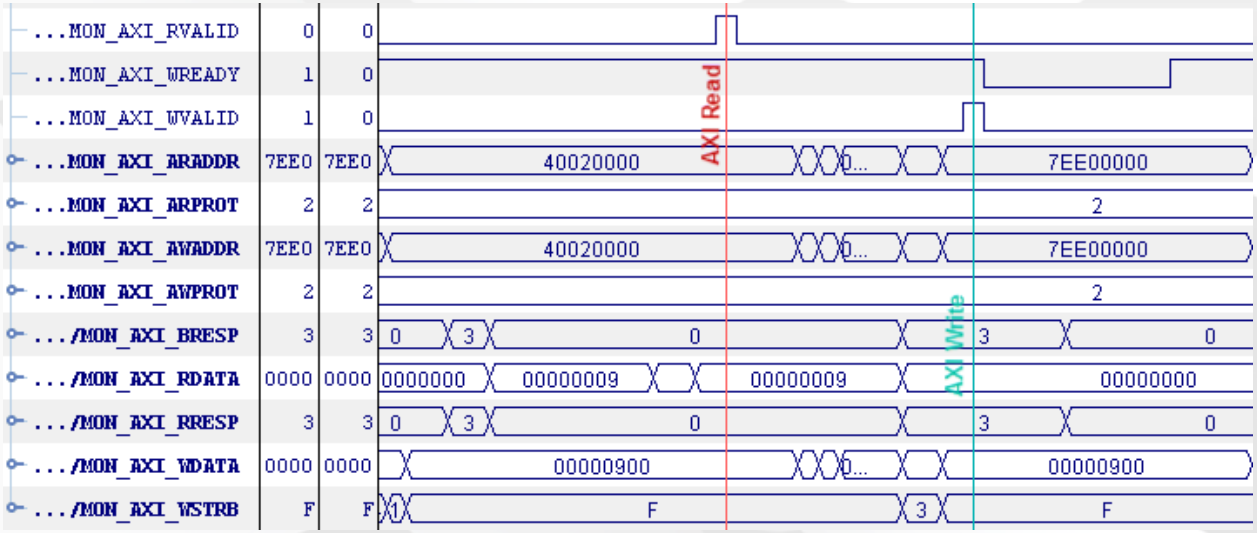
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

ChipScope Pro Analyzer

- **Analyzer can program FPGA**
- **Functions similar to Logic Analyzer with**
  - Trigger Setup
  - Waveform Viewer
  - Bus Plots





Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - **Lab 5 – ChipScope Debugging**
- **Using SPI Flash**
  - Lab 6 – SPI Programming

Lab 5: Embedded ChipScope Debugging

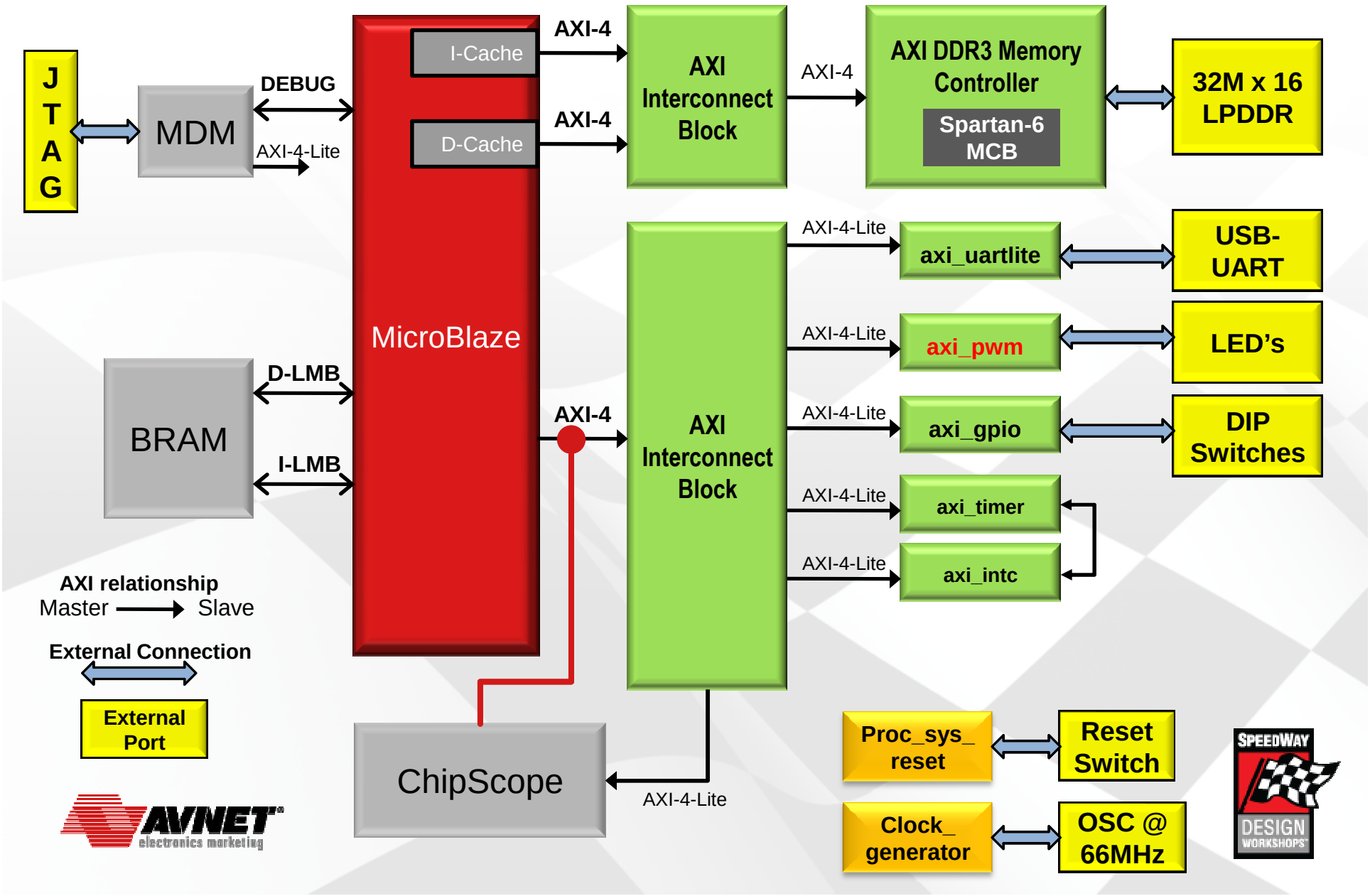
Introduction

- **This tutorial demonstrates how to debug and investigate AXI transactions in the embedded system using ChipScope Pro**

Objectives

- How to add a ChipScope AXI Bus Monitor to our embedded processor
- Generate a FPGA Bitstream with embedded ChipScope core and SDK software application
- How to view and interpret AXI Bus transactions using ChipScope Analyzer

Lab #5 - MicroBlaze Platform - ChipScope

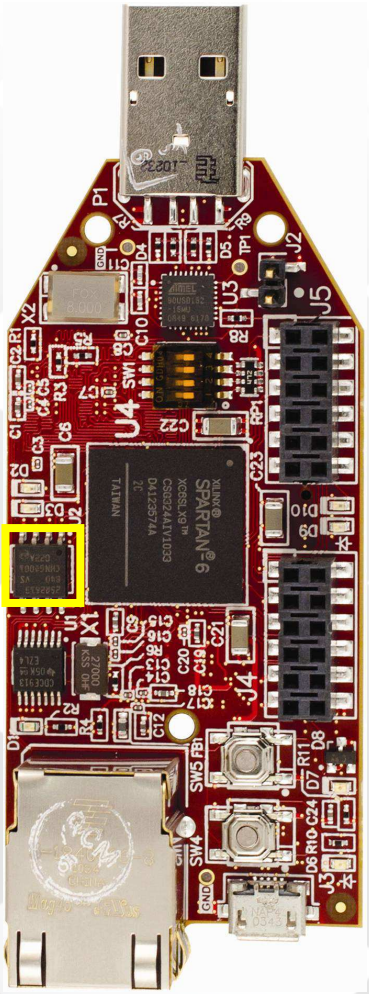


Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - Lab 6 – SPI Programming

Spartan-6 LX9 MicroBoard Peripherals

- **Micron 128Mb Multi-I/O S-P-I Flash**
  - N25Q128A
  - Serial NOR Flash
- **Stores FPGA bitstream with abundant space left-over for:**
  - Additional bitstreams
  - MicroBlaze code
  - User Data



Serial Flash Memory Structure

- **64KB sectors**
  - 256 total sectors
- **New data sections start on sector boundaries**
- **LX9 bitstream = 2.7Mb**
  - Thus take 6 sectors for LX9 configuration memory.
  - Configuration memory starts at base address 0x0

Table 11. Memory organization (uniform) (page 8 of 8)

Sector	Address range	
11	B0000	BFFFF
10	A0000	AFFFF
9	90000	9FFFF
8	80000	8FFFF
7	70000	7FFFF
6	60000	6FFFF
5	50000	5FFFF
4	40000	4FFFF
3	30000	3FFFF
2	20000	2FFFF
1	10000	1FFFF
0	0	FFFF

SW Application

LX9 Config Space

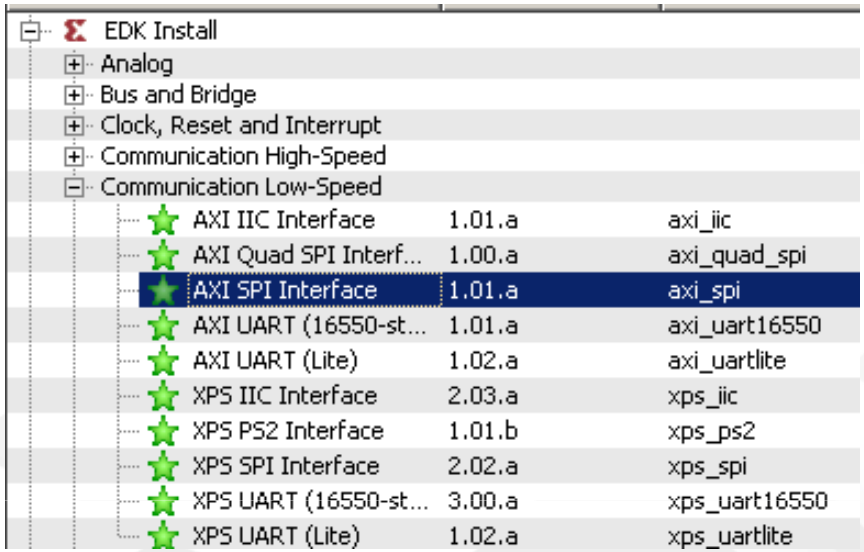
- **2.7Mbits/8-bits = 337500 Bytes**
- **337.5 KB / 64KB = 5.2 sectors**

AVNET electronics marketing

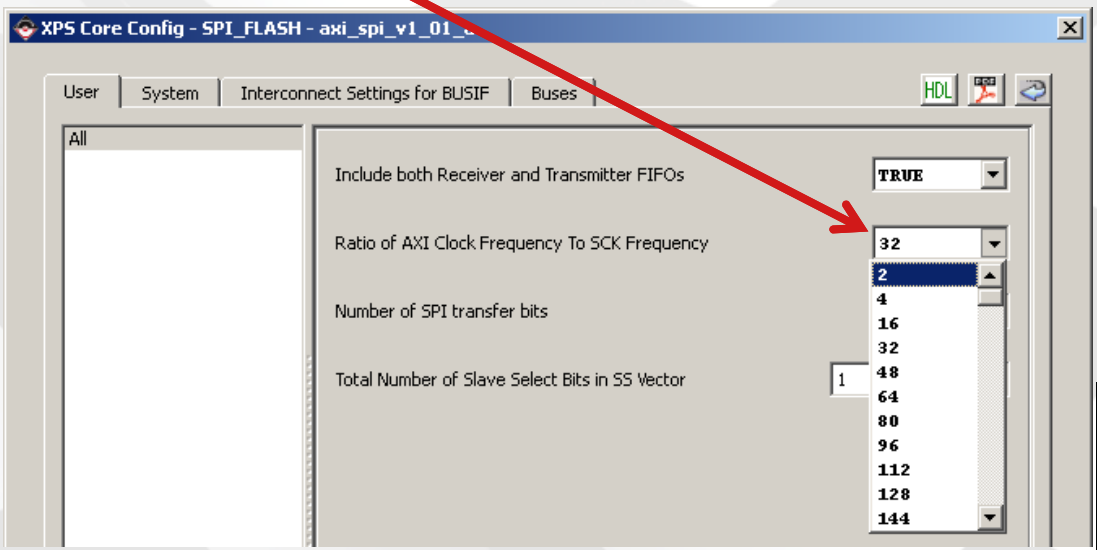
SPEEDWAY DESIGN WORKSHOPS

XPS Interfacing to SPI Flash

- **XPS connects to SPI Flash through axi_spi core**



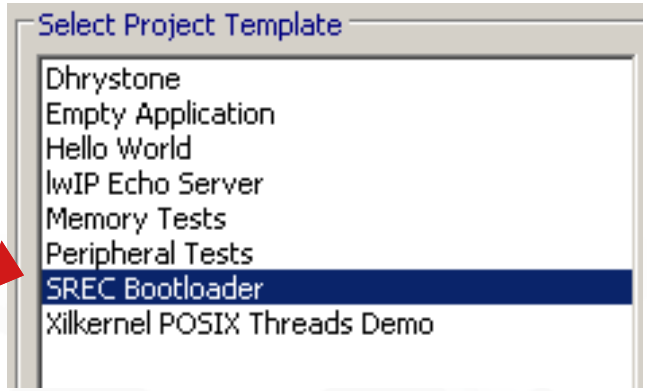
- **Set SPI Flash clock frequency, as ratio to AXI-4-Lite clock**
  - By Default, it divides by 32
  - SPI Flash runs at 100+MHz
  - If AXI4_Lite bus is 66MHz, Divisor of 2 runs SPI Flash at 33MHz



AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

SDK Flash Bootloader



- **SDK has a pre-coded, simple bootloader**
  - Works with parallel flash
    - Must be updated to support serial SPI Flash
- **Bootloader Assumptions:**
  - Image must be SREC format and programmed into known Flash address
    - Modify blconfig.h to set physical address of SREC image
  - Flash image is a software application for MicroBlaze and does not overlap with bootloader
- **Bootloader must be initialized for BRAM so it bootloads the SREC image upon FPGA power-up.**

SDK has SREC image creation tool

Agenda

- **Overview and Xilinx Development Tools Review**
- **Xilinx MicroBlaze Architecture Overview**
- **Creating an Embedded Design**
  - Lab 1- Adding a Processor to a ISE Design
- **Exploring EDK IP Catalog**
  - Lab 2 - Adding EDK IP
- **AXI Interface Introduction**
  - Lab 3 - Adding Custom AXI IP
- **Embedded Simulation**
  - Lab 4 – Simulating a MicroBlaze Design
- **Debugging AXI peripherals with ChipScope**
  - Lab 5 – ChipScope Debugging
- **Using SPI Flash**
  - **Lab 6 – SPI Programming**

Lab 6: Creating a Bootloader & SPI Flash Programming

Introduction

- **This tutorial demonstrates how to create an application to be stored into SPI Flash as well as the boot loader required to copy the application from Flash to main memory for execution**

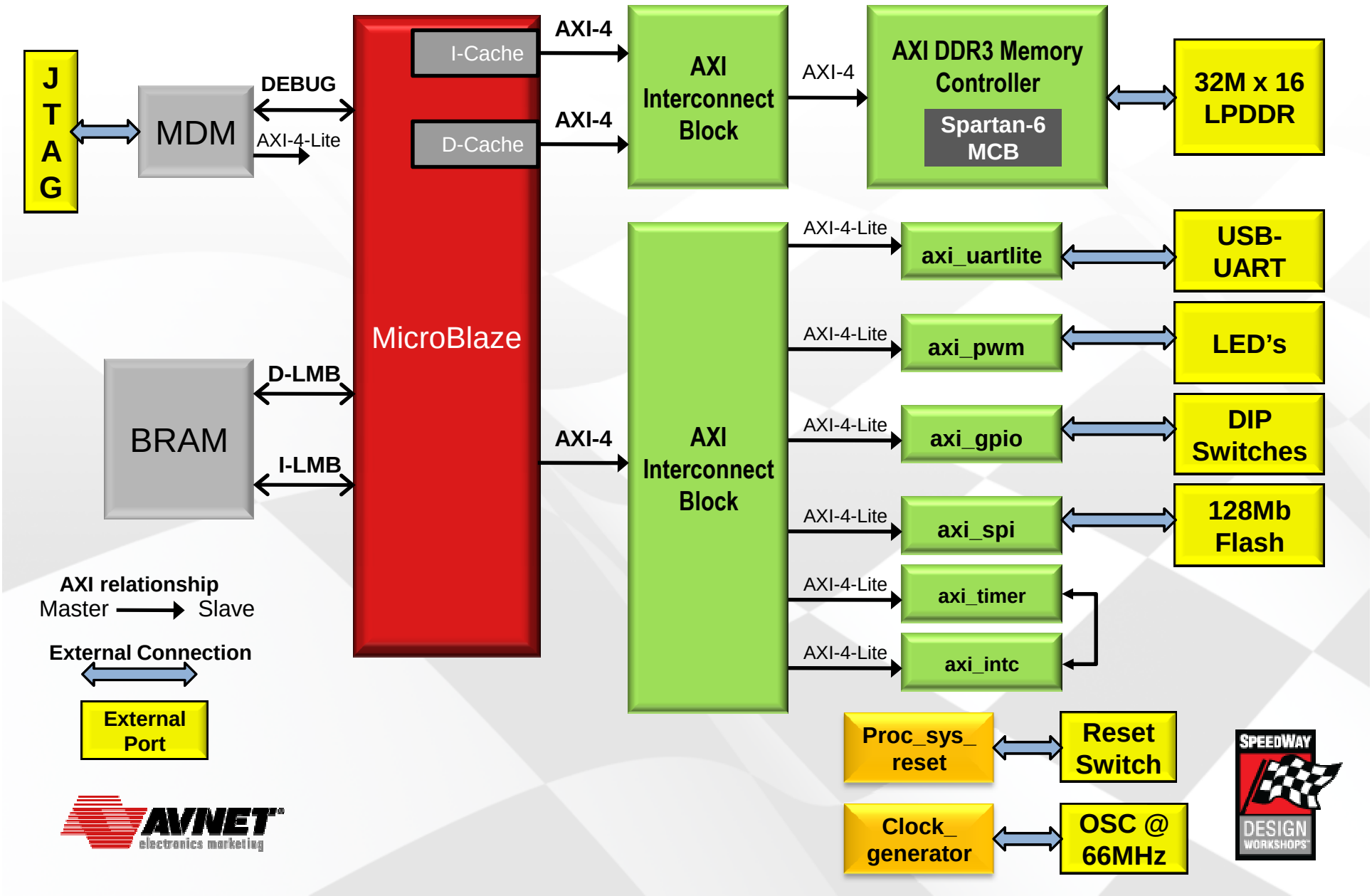
Objectives

- Update the SPI Flash Clock Rate
- Move application to run out of LPDDR and create an SREC Flash Image
- Create a Serial Flash bootloader
- Program the SPI Flash with bootloader and software application using iMPACT

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Lab #6 - MicroBlaze Platform



Course Summary

- **Understand the MicroBlaze development flow and how to use the Xilinx embedded systems tools**
- **Introduce the new AXI Interface Specification**
  - Explore AXI Plug-and-Play IP
- **Utilize the Xilinx embedded systems tools to**
  - Design a MicroBlaze System
  - Develop a Software Application
  - Simulate and Debug an Embedded System
  - Create a bootloader and program serial flash

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Getting Help and Support

- **Evaluation Kit home page with Documentation and Reference Designs**
  http://em.avnet.com/s6microboard

- **Avnet Spartan-6 LX9 MicroBoard forum:**
  http://community.em.avnet.com/t5/Spartan-6-LX9-MicroBoard/bd-p/Spartan-6LX9MicroBoard

- **For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at www.support.xilinx.com. On this site you will also find the following resources for assistance:**
  - Software, IP, and Documentation Updates
  - Access to Technical Support Web Tools
  - Searchable Answer Database with Over 4,000 Solutions
  - User Forums
  - Training - Select instructor-led classes and recorded e-learning options

- **Contact Avnet Support for any questions regarding the Spartan-6 LX9 MicroBoard reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.**
- **http://www.em.avnet.com/techsupport**

- **You can also contact your local Avnet/Silica FAE.**

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS