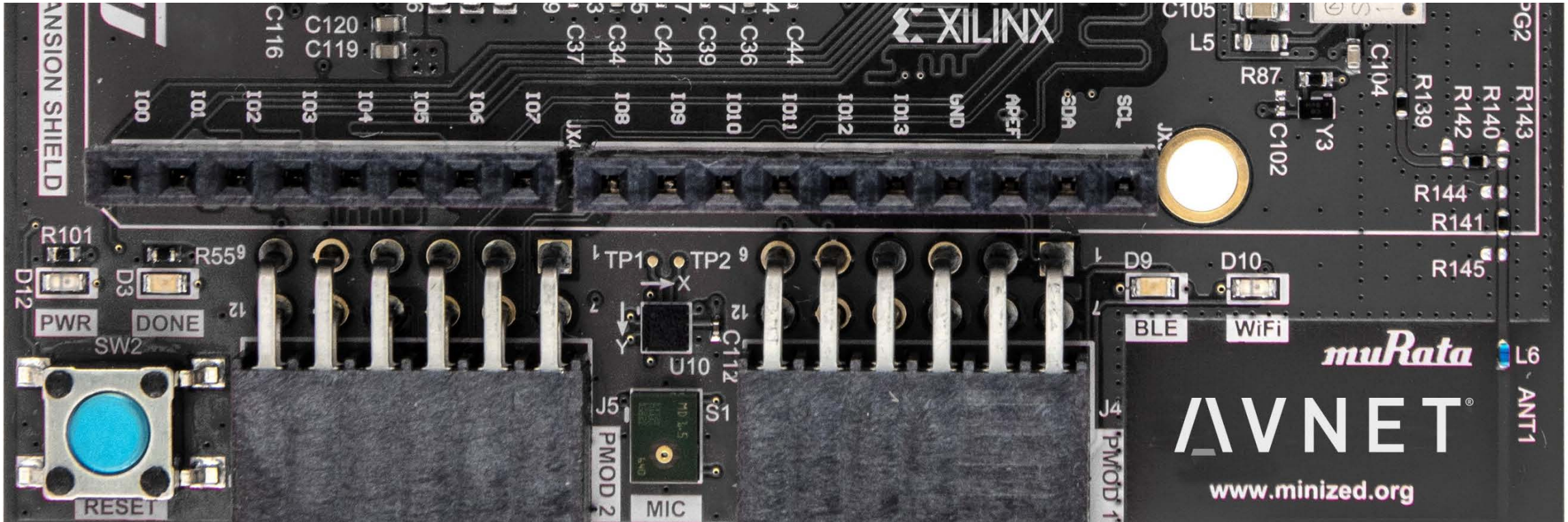
Developing Zynq Software

With Xilinx Software Development Kit 2017.4

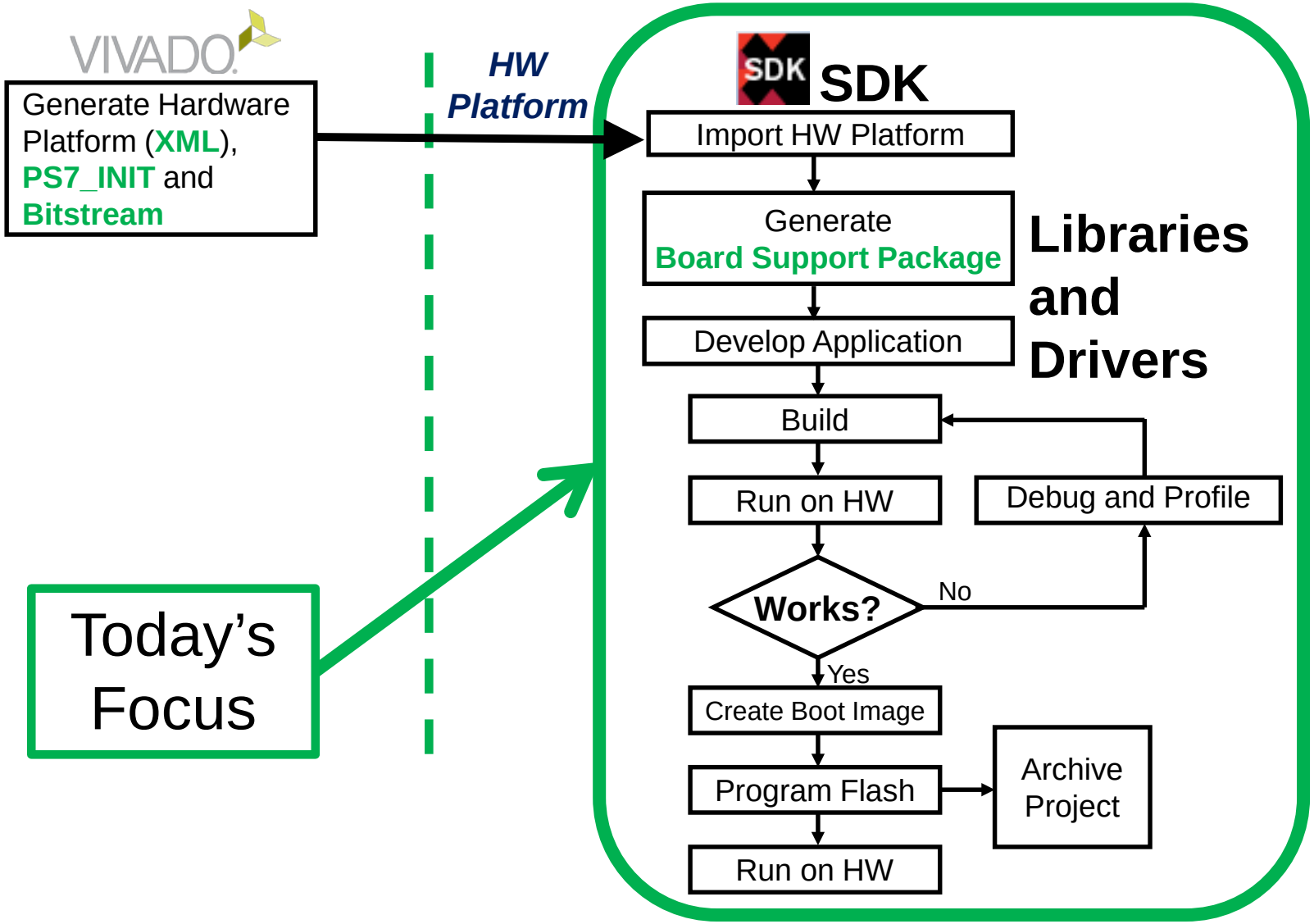
SPEEDWAY

AVNET
Reach Further™

15 February 2018



SDK Application Development Flow



AVNET

Objectives

- Introduce developers to Xilinx SDK
- Explore how SDK makes your job easier
- Connect SDK to hardware for execution and debug
- Utilize a peripheral interrupt to show real-time software response
- Show a basic example of how to use an external sensor module

Agenda

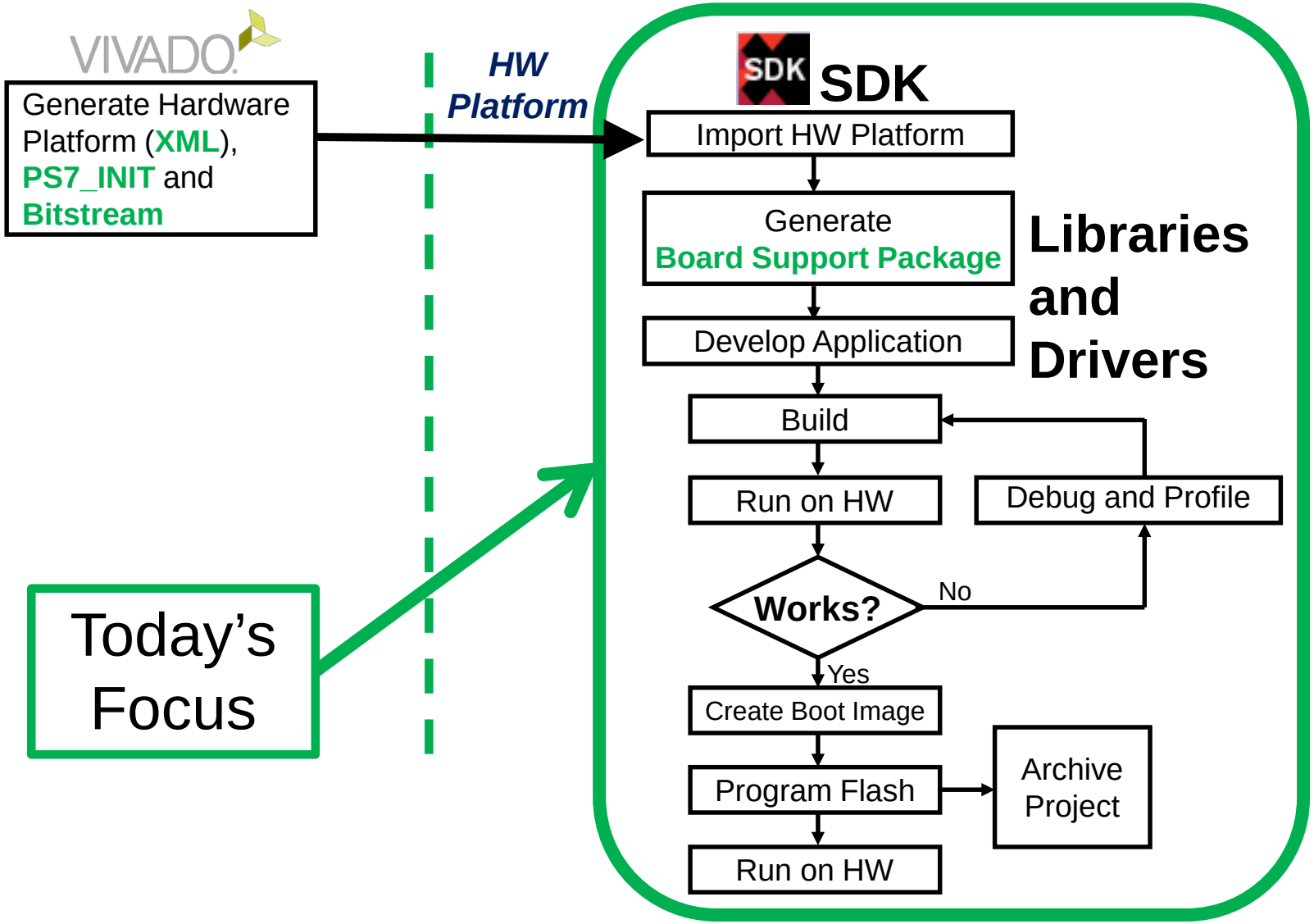
Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

SDK Application Development Flow



Introducing Zynq-7000S Devices

Application Processors

ARM Cortex A9

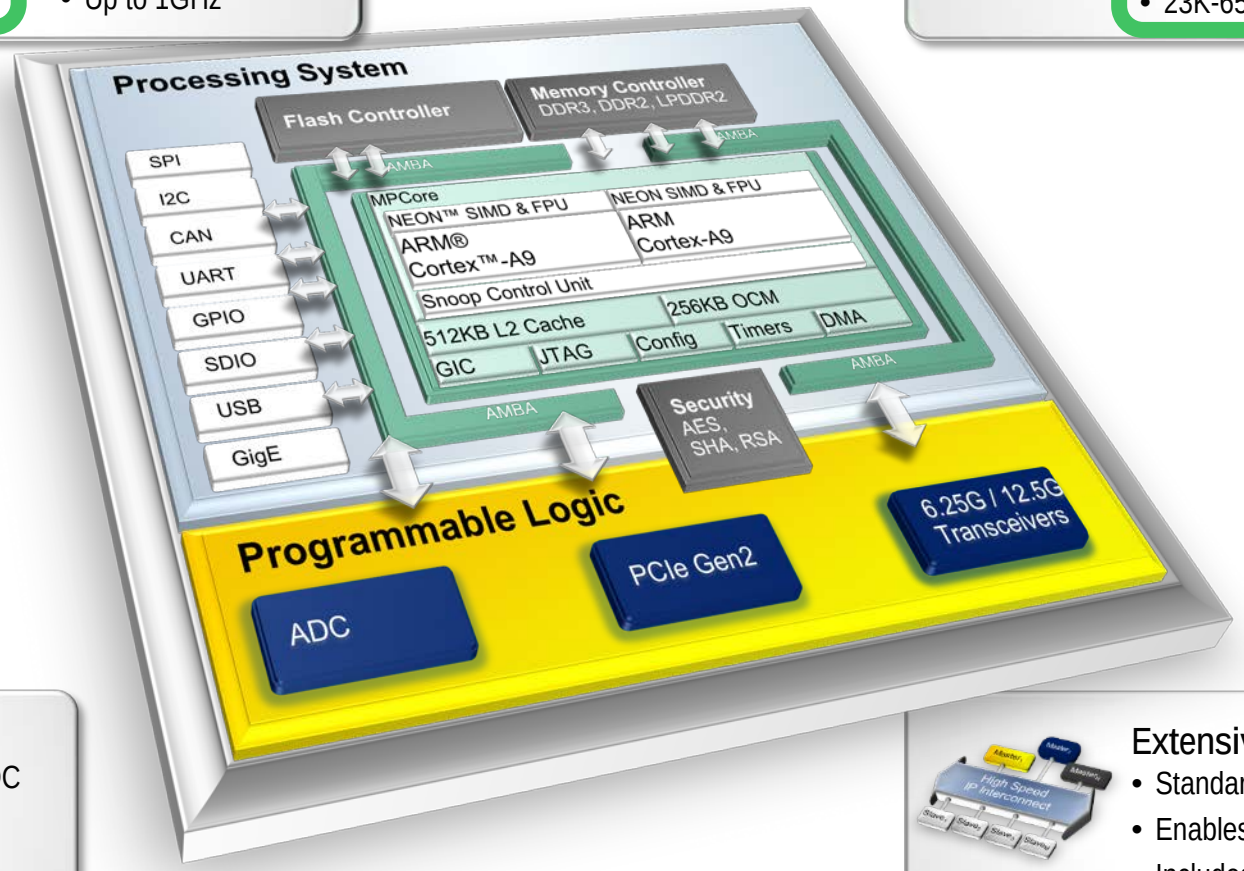
Zynq-7000S
- Single-Core
- Up to 766MHz

Zynq-7000
- Dual-Core
- Up to 1GHz

Programmable Logic

Zynq-7000S
- Artix-7 Series FPGA
- 23K-65K Logic Cells

Zynq-7000
- 7 Series FPGA
- 28K-440K Logic Cells



High Bandwidth Memory

- L1/L2 CPU Caches
- Dedicated On-Chip Memory (OCM)
- DDR3, DDR2, LPDDR2 w/ ECC

Integrated Memory Mapped Peripherals

- e.g. USB2.0, GigE

Tightly Coupled Domains

- 3000+ PS/PL interconnects
- Low Latency
- Up to 100Gb of bandwidth

Integrated Analog

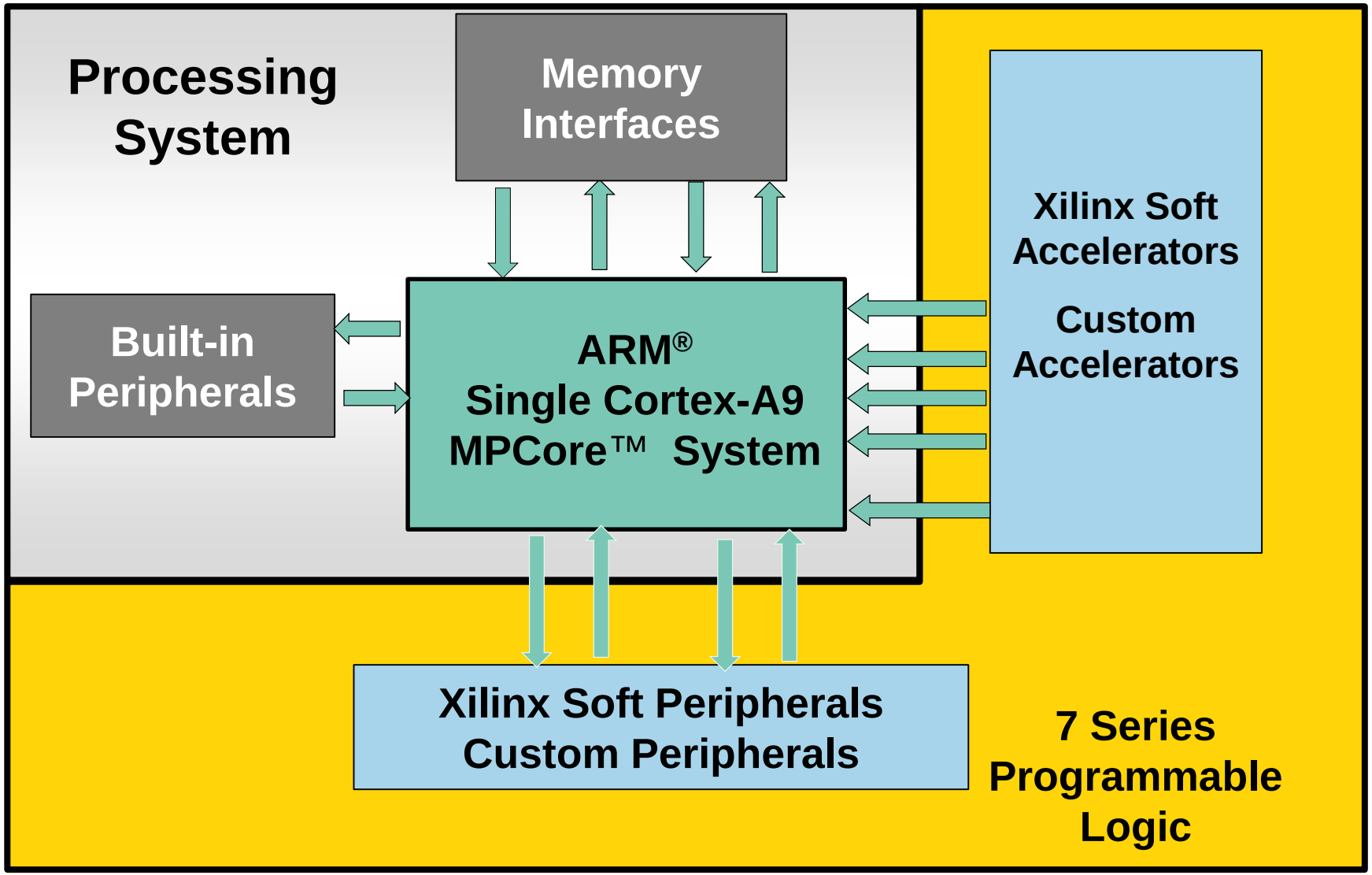
- Dual multi-channel 12-bit ADC
- Up to 1Msps
- Temp & Voltage sensors

Extensive IP Portfolio

- Standardized AXI4 interfaces
- Enables peripheral expansion
- Includes software drivers

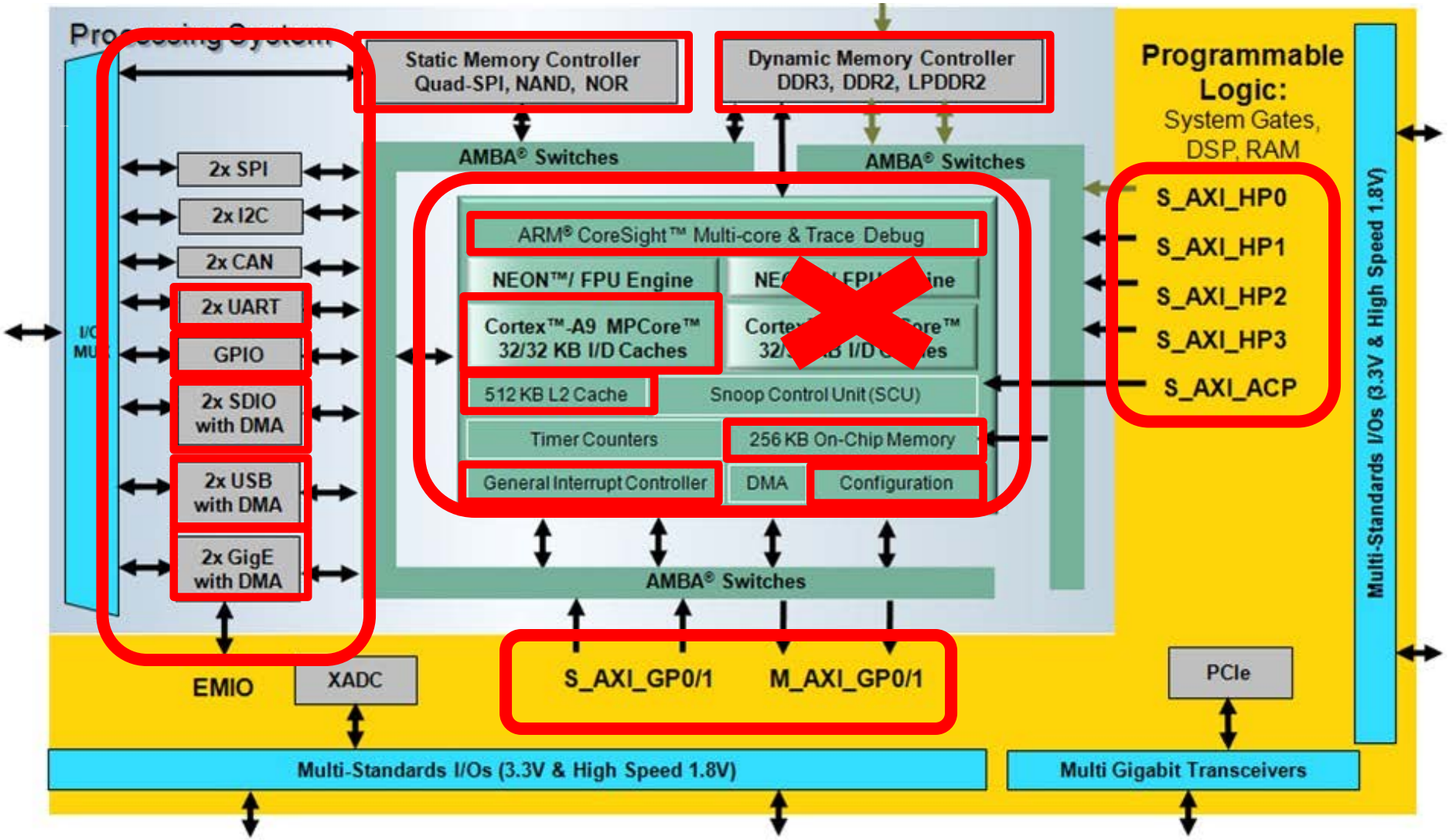
AVNET

Zynq-7000S AP SoC Basic Architecture



AVNET

Zynq-7000S AP SoC Block Diagram



AVNET

Zynq-7000 Device Portfolio Summary

Scalable platform offers easy migration between devices

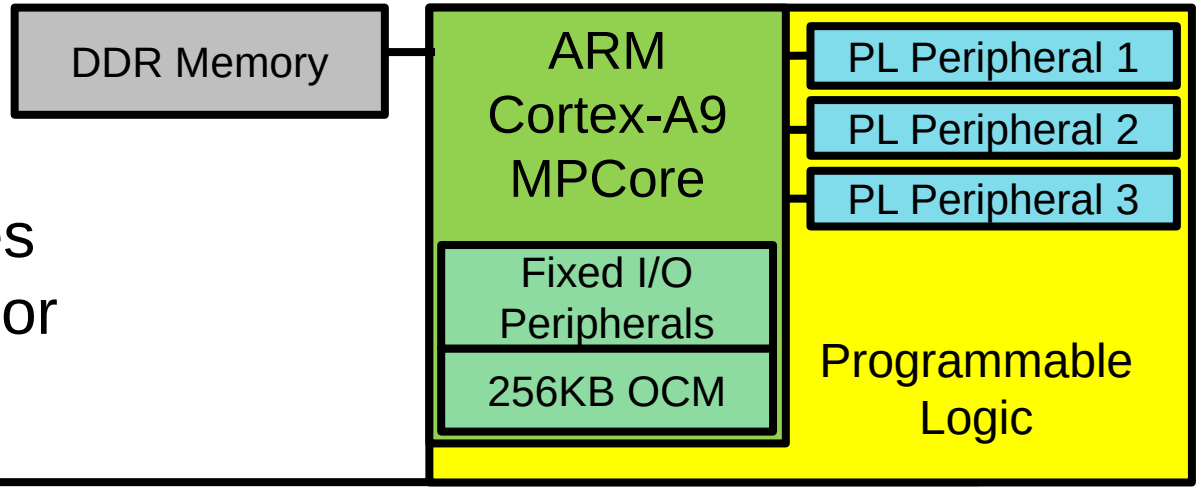
		Cost-Optimized Devices						Mid-Range Devices			
	Device Name	Z-7007S	Z-7012S	Z-7014S	Z-7010	Z-7015	Z-7020	Z-7030	Z-7035	Z-7045	Z-7100
	Part Number	XC7Z007S	XC7Z012S	XC7Z014S	XC7Z010	XC7Z015	XC7Z020	XC7Z030	XC7Z035	XC7Z045	XC7Z100
Processing System (PS)	Processor Core	ARM® Cortex™-A9 MPCore™ Up to 766MHz Single-Core			ARM Cortex-A9 MPCore Up to 866MHz Dual-Core			ARM Cortex-A9 MPCore Up to 1GHz[1] Dual-Core			
	Processor Extensions	NEON™ SIMD Engine and Single/Double Precision Floating Point Unit per processor									
	L1 Cache	32KB Instruction, 32KB Data per processor									
	L2 Cache	512KB									
	On-Chip Memory	256KB									
	External Memory Support[2]	DDR3, DDR3L, DDR2, LPDDR2									
	External Static Memory Support[2]	2x Quad-SPI, NAND, NOR									
	DMA Channels	8 (4 dedicated to PL)									
	Peripherals	2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO									
	Peripherals w/ built-in DMA[2]	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO									
	Security[3]	RSA Authentication of First Stage Boot Loader, AES and SHA 256b Decryption and Authentication for Secure Boot									
	Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master, 2x AXI 32b Slave 4x AXI 64b/32b Memory AXI 64b ACP 16 Interrupts									
Programmable Logic (PL)	7 Series PL Equivalent	Artix®-7	Artix-7	Artix-7	Artix-7	Artix-7	Artix-7	Kintex®-7	Kintex-7	Kintex-7	Kintex-7
	Logic Cells	23K	55K	65K	28K	74K	85K	125K	275K	350K	444K
	Look-Up Tables (LUTs)	14,400	34,400	40,600	17,600	46,200	53,200	78,600	171,900	218,600	277,400
	Flip-Flops	28,800	68,800	81,200	35,200	92,400	106,400	157,200	343,800	437,200	554,800
	Total Block RAM	1.8Mb	2.5Mb	3.8Mb	2.1Mb	3.3Mb	4.9Mb	9.3Mb	17.6Mb	19.1Mb	26.5Mb
	(# 36Kb Blocks)	(50)	(72)	(107)	(60)	(95)	(140)	(265)	(500)	(545)	(755)
	DSP Slices	60	120	170	80	160	220	400	900	900	2,020
	PCI Express®	—	Gen2 x4	—	—	Gen2 x4	—	Gen2 x4	Gen2 x8	Gen2 x8	Gen2 x8
	Analog Mixed Signal (AMS) / XADC[2]	2x 12 bit, MSPS ADCs with up to 17 Differential Inputs									
	Security[3]	AES & SHA 256b Decryption & Authentication for Secure Programmable Logic Config									
	Speed Grades: Commercial		-1		-1			-1			-1
	Speed Grades: Extended		-2		-2,-3			-2,-3			-2
	Speed Grades: Industrial		-1, -2		-1, -2, -1L			-1, -2, -2L			-1, -2, -2L

MiniZed

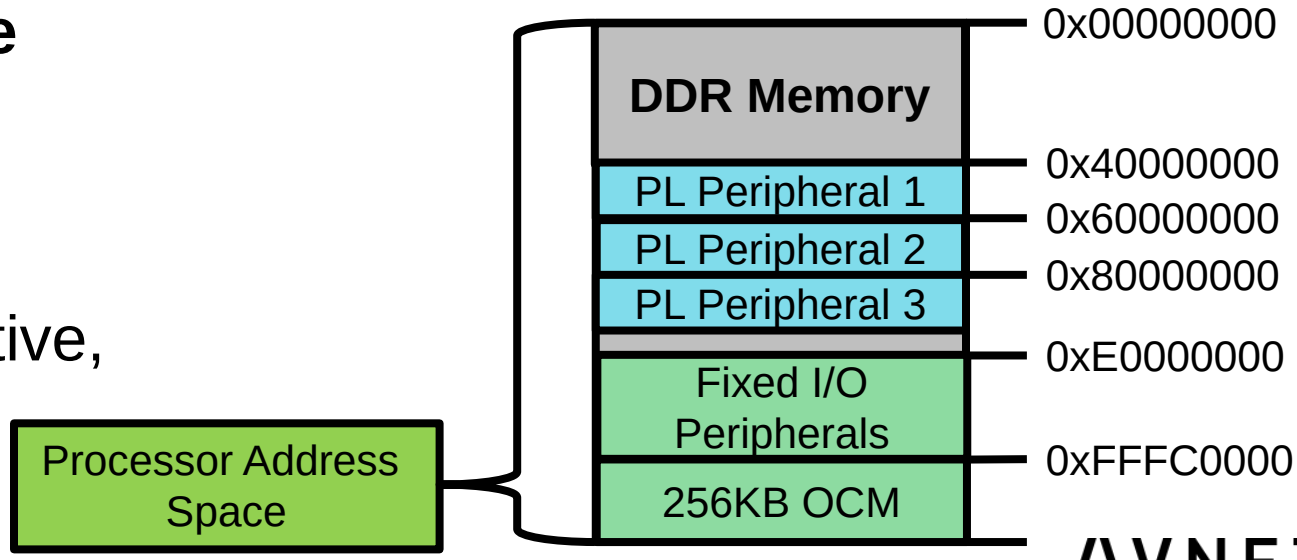
AVNET

Connecting HW and SW

- **Hardware Designer Perspective**
  - Peripheral blocks can be standard core offerings or custom logic cores
  - Logic controls exposed to Processor System via register interface



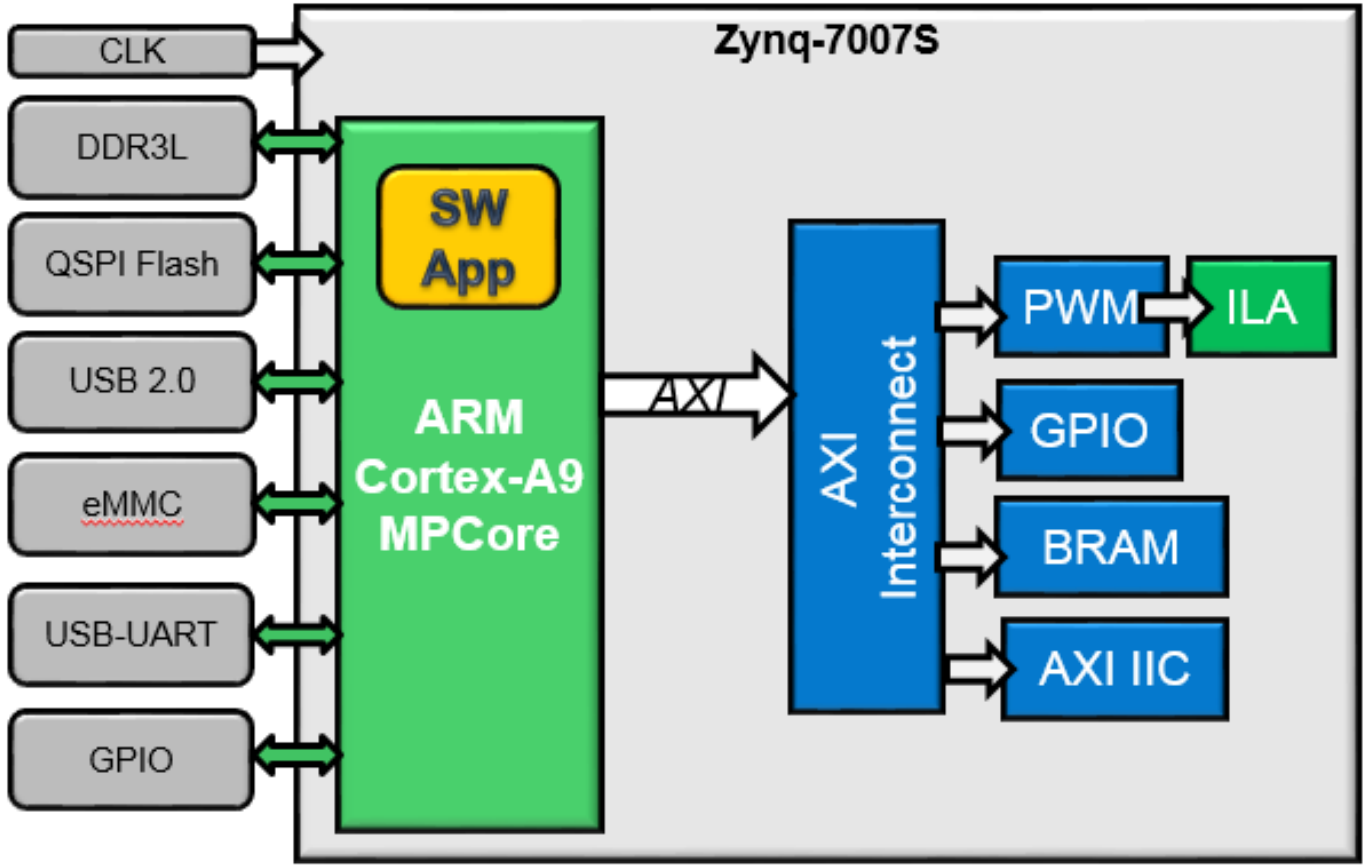
- **Software Developer Perspective**
  - Processor System controls Programmable Logic blocks via exposed register interface
  - From ARM processor's perspective, everything is memory mapped



AVNET

The Zynq Hardware Platform

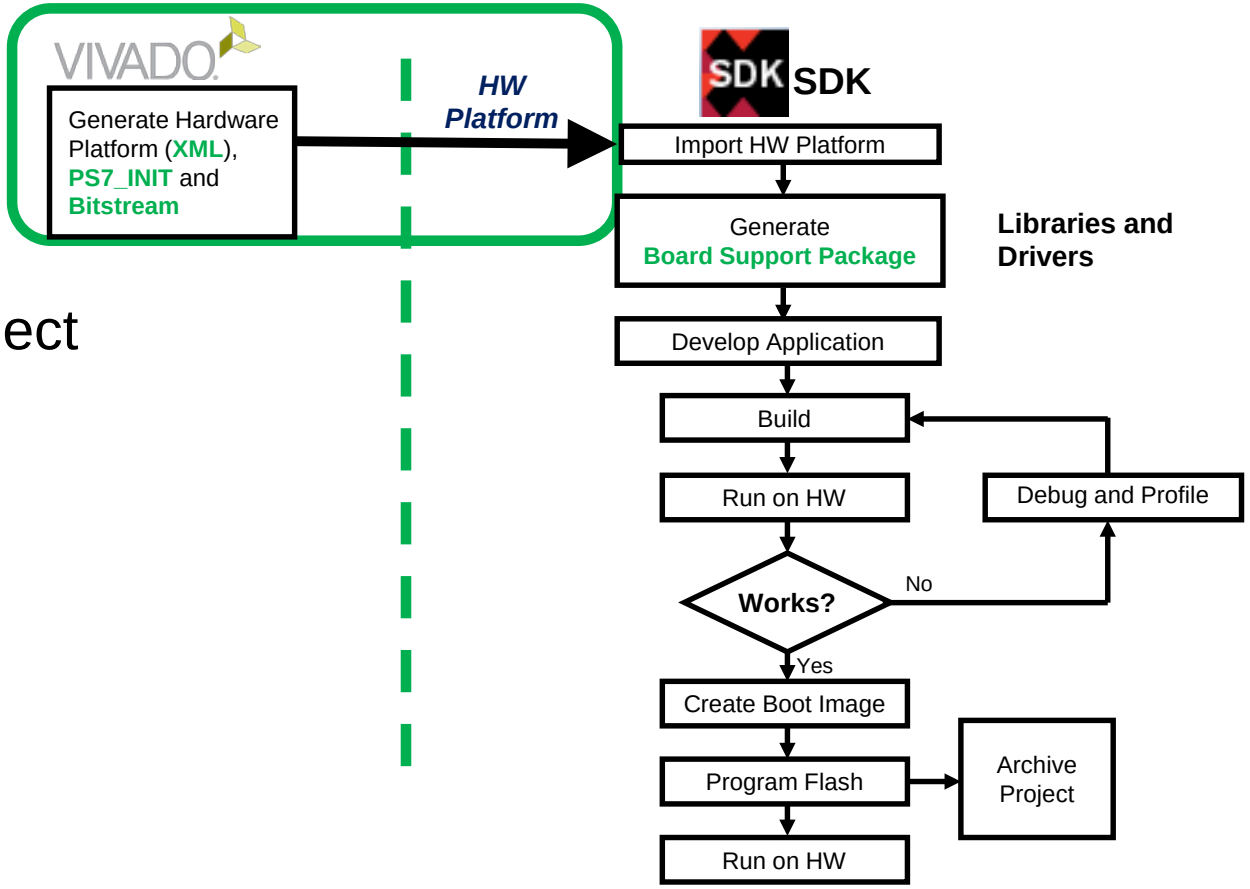
- Output from Vivado
  - Hardware description in XML format
  - FPGA bitstream
  - Block RAM Memory Map (BMM) file
- Includes
  - Zynq processor configuration
  - PS Peripheral and MIO assignments
  - PL design and pinout
- Pre-built for this course
- To learn how to create hardware platforms, look at the *Developing Zynq Hardware* course



AVNET

Lab 1 – Explore an Exported Hardware Platform

- Created in Vivado
- Pre-built
- This lab includes the complete Vivado project
  - Not necessary to have the full project
  - The export directory alone is enough
- No Xilinx tools required for this lab



AVNET

Questions

- What tool suite creates the Zynq hardware platform archive?
  - Vivado is the currently recommended tool for creating Zynq hardware platforms
- Which file in the hardware platform archive is most comparable to a ‘datasheet’ for the hardware platform?
  - ps7_init.html
- Which file contains the information that SDK will later use to build a BSP?
  - The Hardware Handoff File (HWH in XML format) file – Z_system.hwh
- Which file will later be used to initialize the Zynq PS during a debug session?
  - The TCL file – ps7_init.tcl

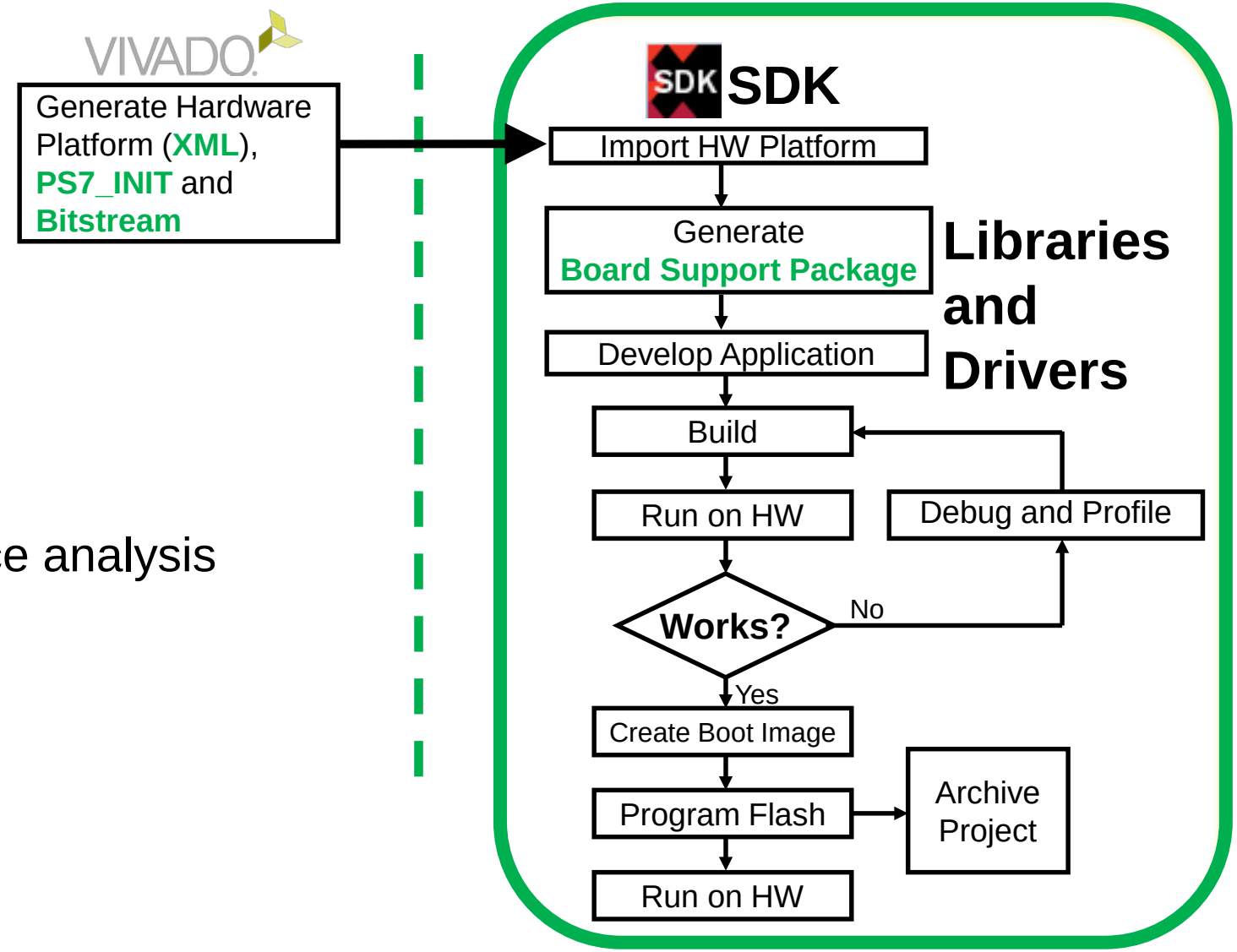
Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

SDK Application Development Flow

- Automatic BSP generation
- GNU cross-compiler toolchain
- Common application templates
- Automated linker script generation
- TCF-based multi-core debugger
- Hardware and software download
- Hardware/Software cross-probing
- Multi-processor debugging
- Application profiling and performance analysis
- Boot image creation
- Flash memory programming



Basic Eclipse Concepts

- Workbench
  - Refers to the Eclipse IDE
- Workspace
  - Keeps Workbench session information
  - Stores Eclipse preferences
  - Only one active Workspace at a time per Workbench
  - Manages C/C++ projects
- Perspectives
  - Collection of functionally related views, editors, menu items, and toolbar buttons
  - Layout of views in perspective can be customized
- Views
  - Basic user interface element
  - Provides information, control, or navigation

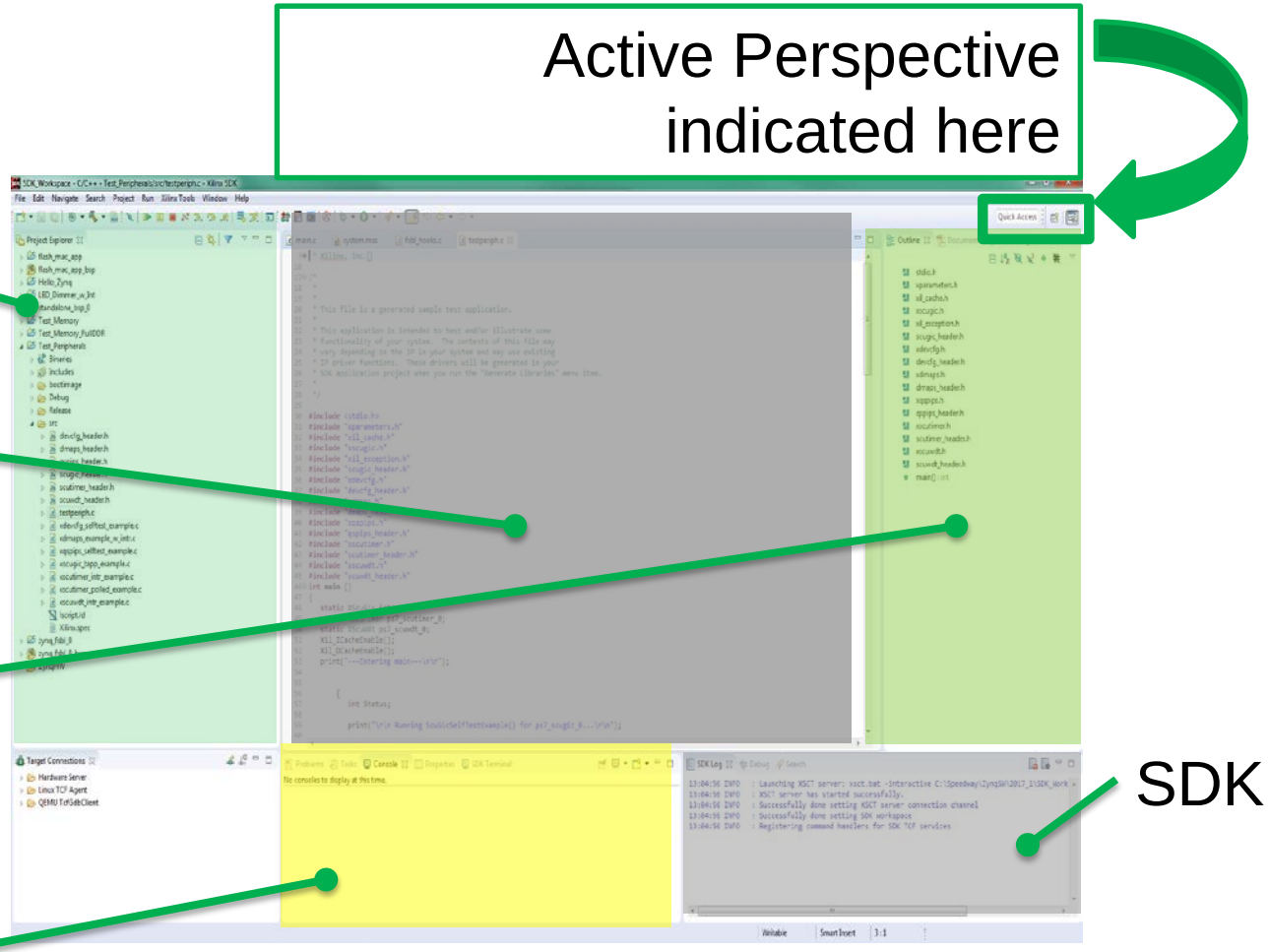
SDK Workbench Views – C/C++ Perspective

Project Explorer displays the projects with icons for easy identification

C/C++ editor for integrated software creation

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

Problems, Console, and Properties views list output information

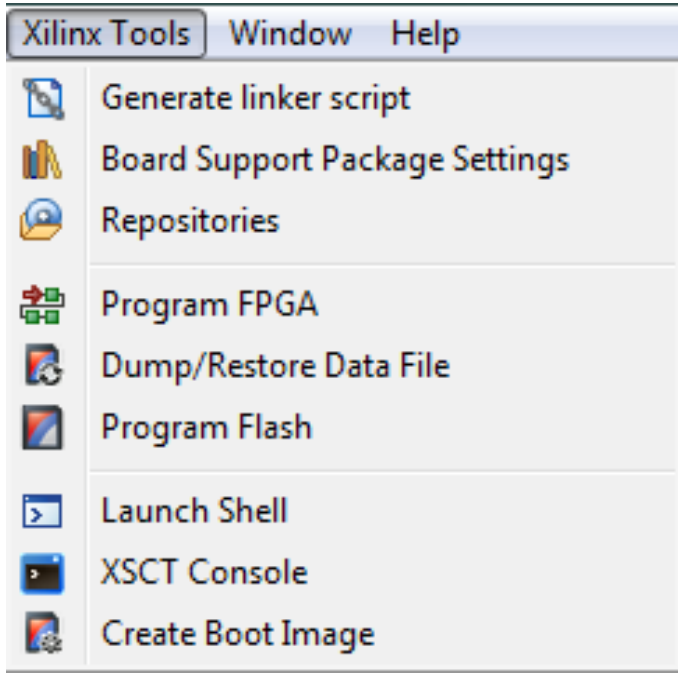


AVNET

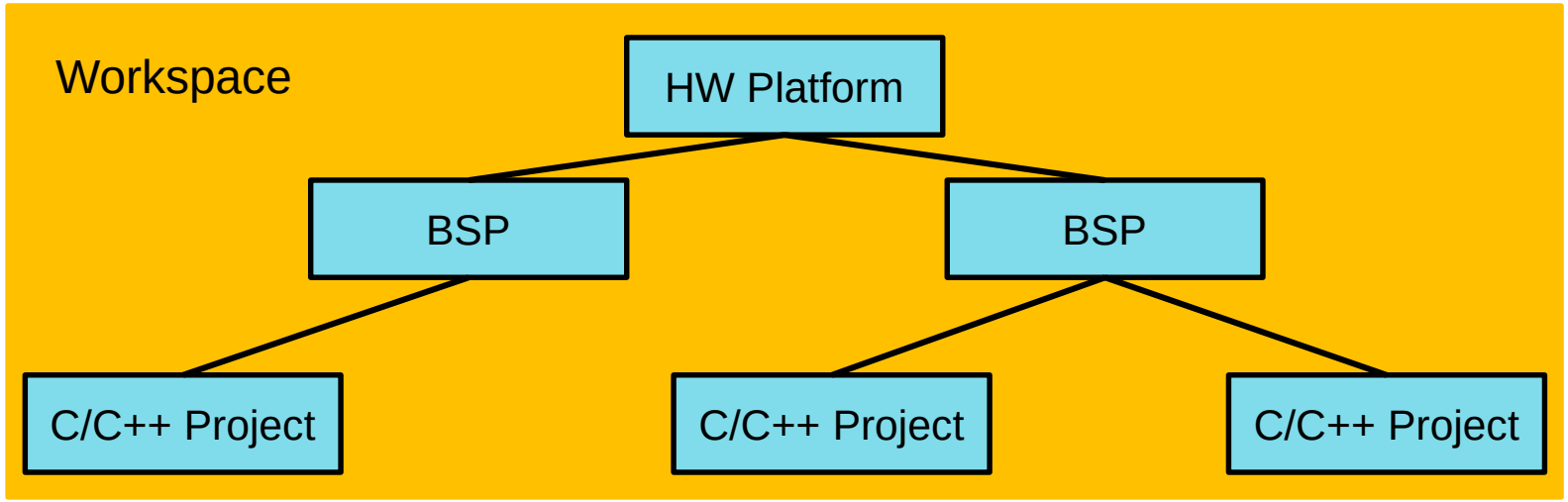
Xilinx Plug-ins

- Plug-ins provide a way to customize Eclipse

- Xilinx plug-ins include:
  - Project creation wizard
  - Xilinx specific managed build
  - Xilinx specific toolchain and utilities
  - Xilinx specific menu items
  - Debug integration
  - Linker script generator
  - Boot image creation
  - Flash programmer
  - Xilinx documentation



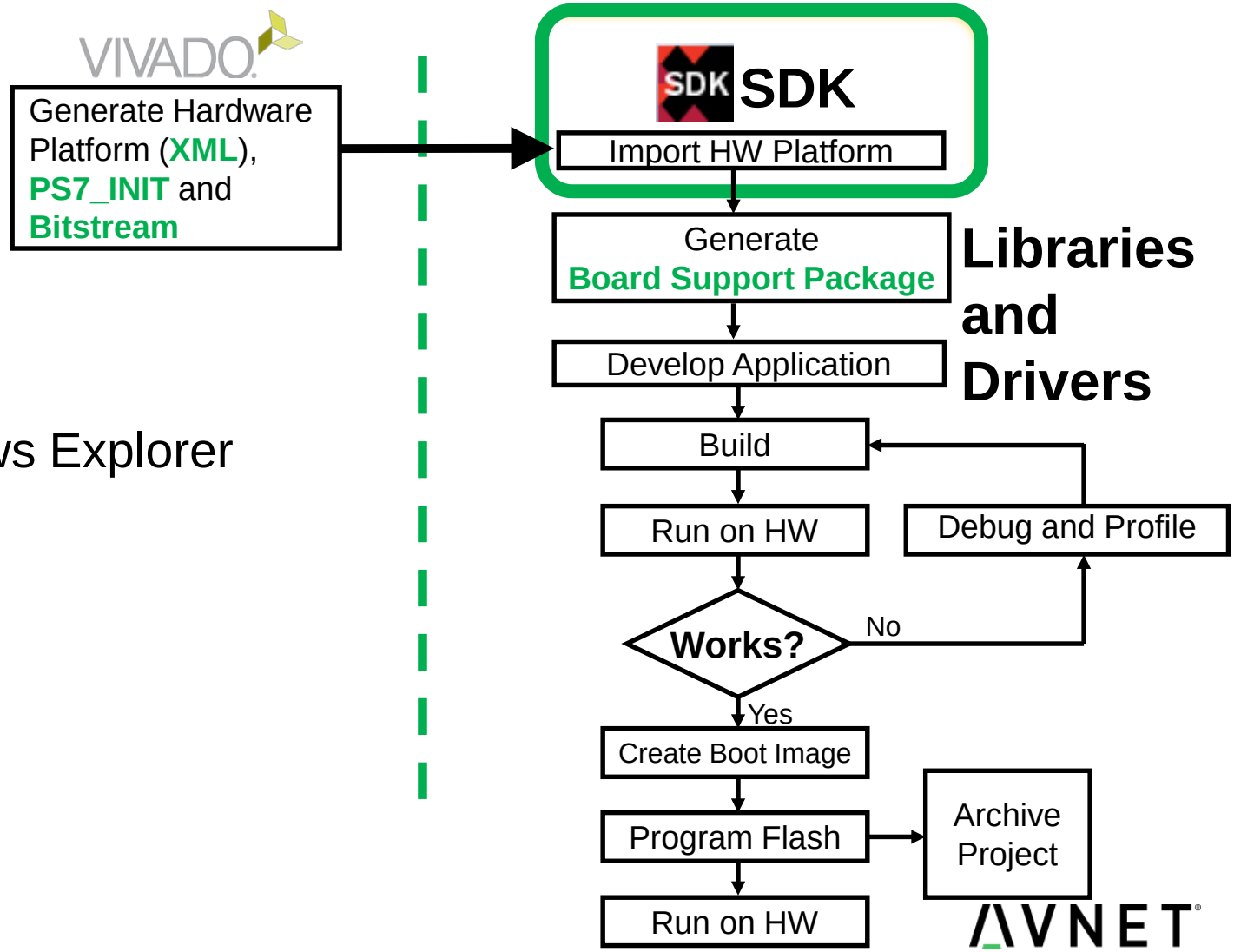
Application Dependencies



- Every workspace has one Hardware Platform
  - Specified through an XML file
  - XML file is generated in Vivado
- Multiple Board Support Packages (BSPs) can reference a single Hardware Platform
  - Xilinx provides a Standalone BSP
- Multiple C/C++ projects can reference a single BSP

Lab 2 – SDK Workspace and Hardware Platform

- Launch SDK
- Create a workspace
- Import Hardware Platform Definition
- Examine results in SDK and Windows Explorer



Questions

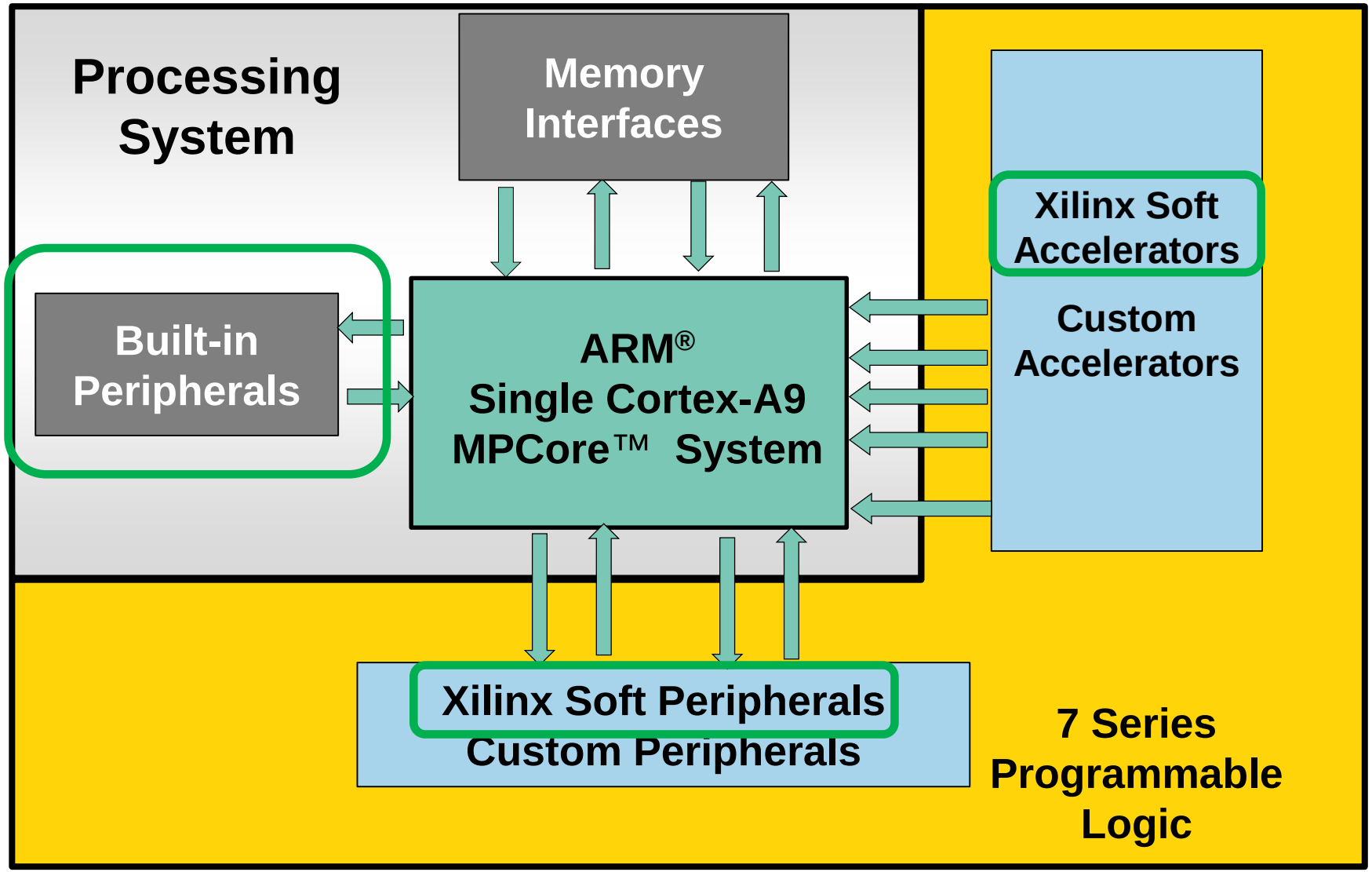
- What is the purpose of the SDK Workspace?
  - The Workspace holds a collection of projects related to your software application development. It will typically contain one hardware platform and one or more BSPs and software applications. The workspace contains your SDK settings, software sources, and logs.
- When you need to share an SDK Workspace, is it a good idea to simply zip it up and send it over?
  - NO! In Lab 8, we'll show you the proper way to archive your workspace
- What is address range for the DDR3 (ps7_ddr_0)?
  - 0x00100000 to 0x1FFFFFFF (MiniZed)
- How large is the QSPI Flash in the system?
  - 0xFCFFFFFF – 0xFC000000 + 1 = 0x1000000 = 16 MB = 128 Mb addressable space
- Where will you find the hardware datasheet for the ps7* peripherals?
  - The Zynq-7000 Technical Reference Manual (TRM) – UG585

Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

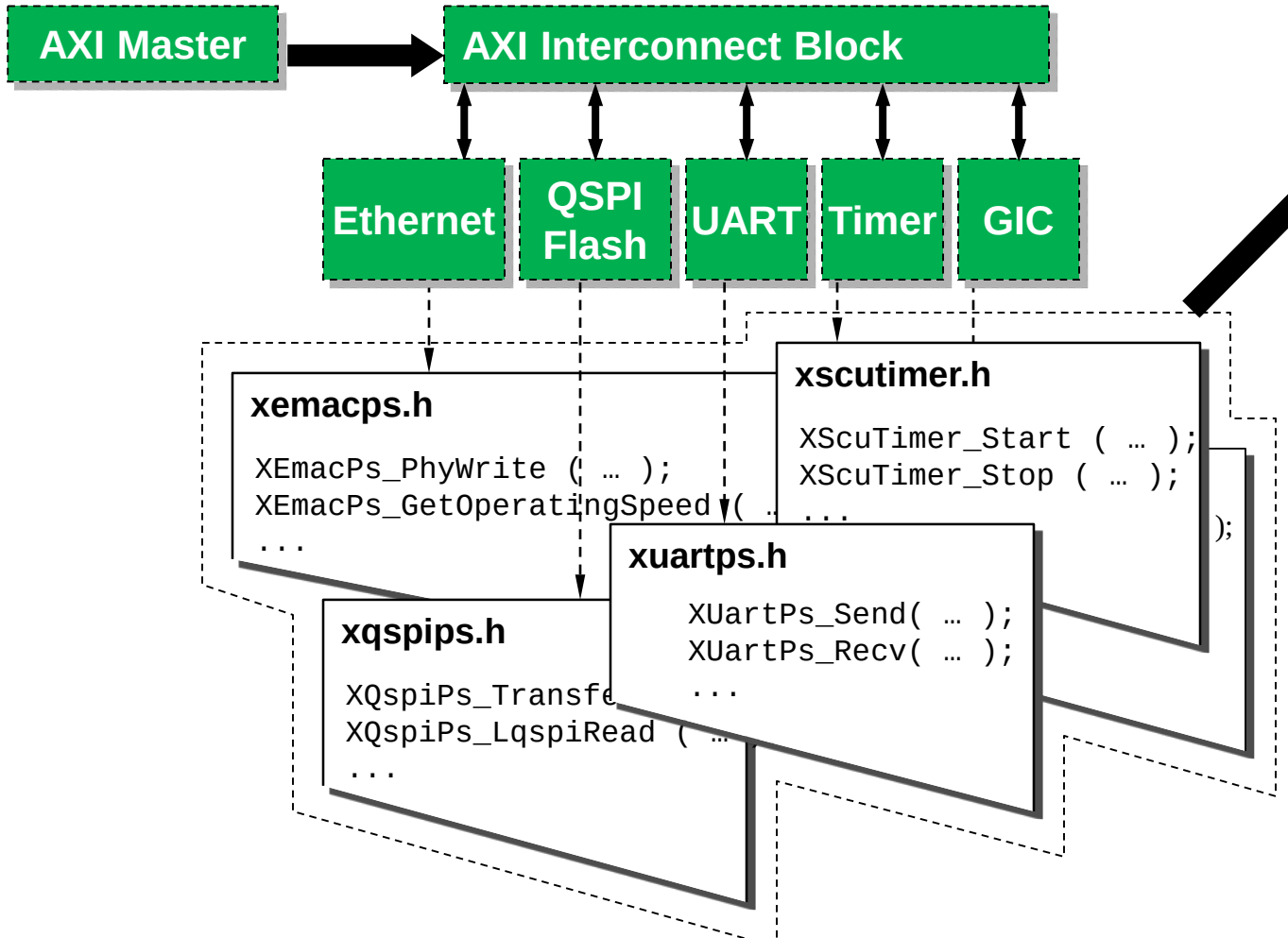
Xilinx Provides Drivers and Services

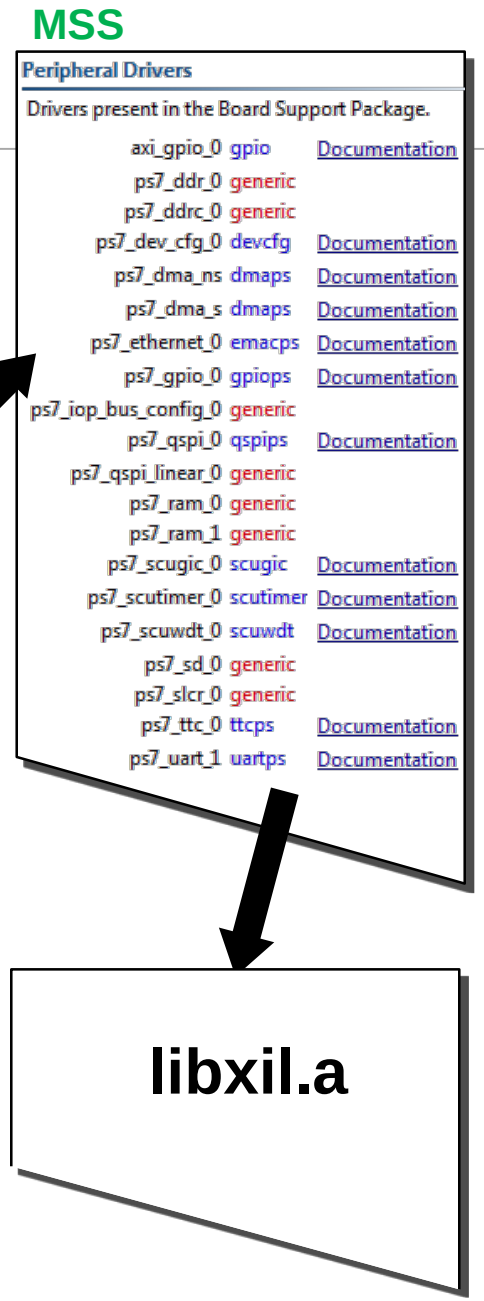


AVNET

Board Support Package (BSP)

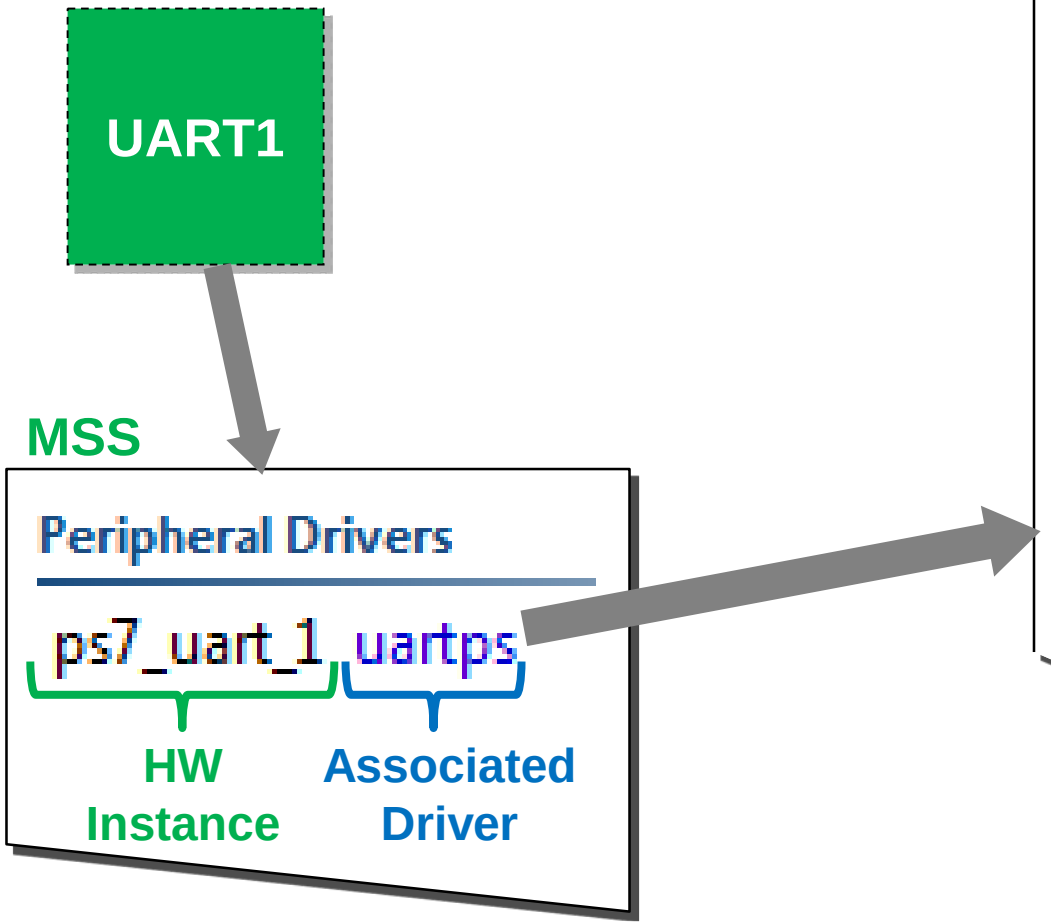
Board Support Packages are collections of parameterized drivers for a specific microprocessing system





AVNET

UART Device Driver Example



xuartps.h

Structures

XUartPS – Stores information about UART

Initialization

```
XUartPs_CfgInitialize(…);
XUartPs_EnableUart(…);
XUartPs_DisableUart(…);
```

Send/Receive

```
XUartPs_Send( … );
XUartPs_Recv( … );
XUartPs_IsTransmitEmpty(…);
```

AVNET

GCC Low-Level Runtime Library

- Zynq uses standard C libraries from libgcc.a

- Included and statically linked from toolchain install folder

`C:\Xilinx\SDK\2017.4\gnu\aarch32\nt\gcc-arm-none-eabi\lib\gcc\arm-none-eabi\6.2.1`

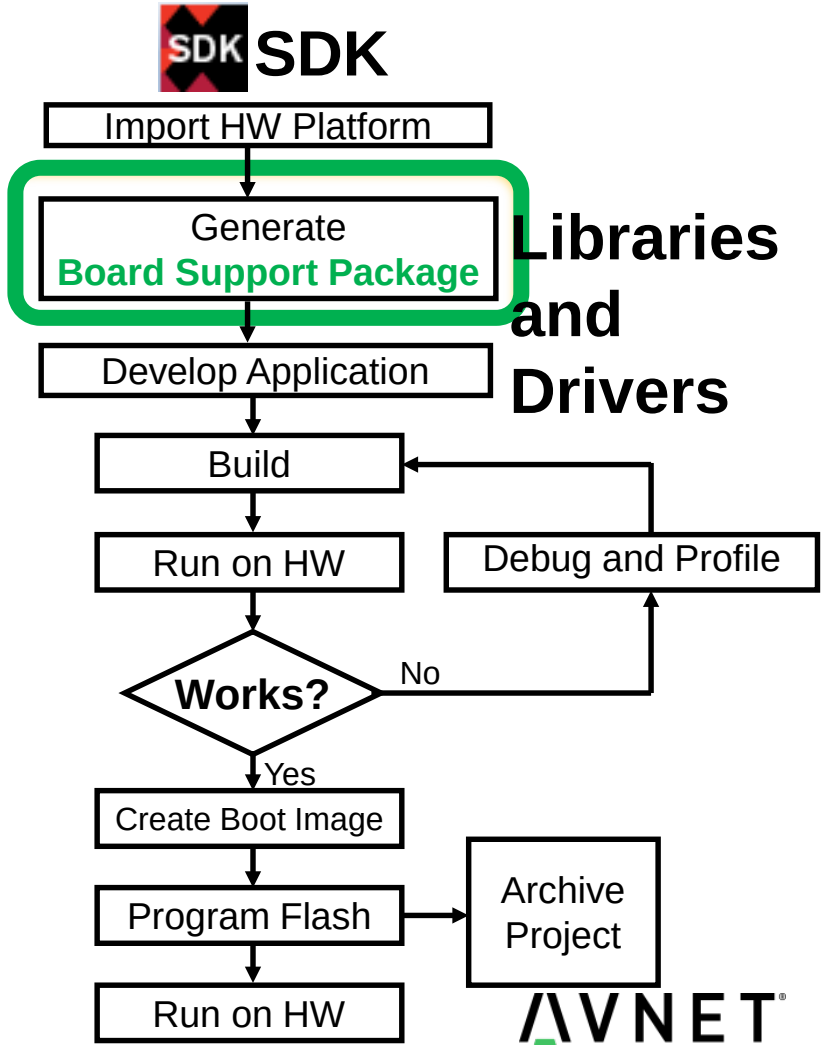
- Routines provided
  - Integer library routines
  - Soft float library routines
  - Decimal float library routines
  - Fixed-point fractional library routines
  - Exception handling routines
  - Miscellaneous routines

- Look at Xilinx ***OS and Libraries Document Collection*** (UG645) for more information

AVNET

Lab 3 – Board Support Package

- Create the BSP
- Examine what was created
- Explore a built-in device driver



Questions

- What are the base addresses for the OCM segments, XPAR_PS7_RAM_0_S_AXI_BASEADDR and XPAR_PS7_RAM_1_S_AXI_BASEADDR?
  - 0x00000000 and 0xFFFC0000

- What size is PS7_RAM_0 ?
  - 0x0003FFFF – 0x00000000 + 1 = 0x40000 = 262144 bytes= 256 KB

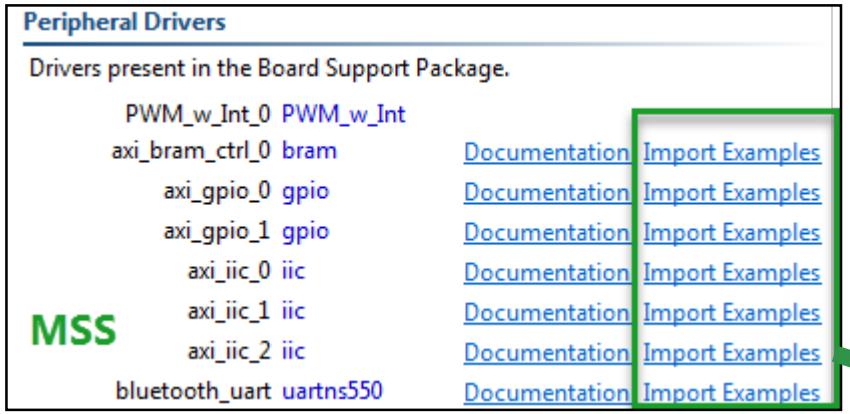
- What parameter name would you access for the PWM peripheral interrupt?
  - XPAR_FABRIC_PWM_W_INT_0_INTERRUPT_OUT_INTR, which happens to be set to interrupt #61

Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

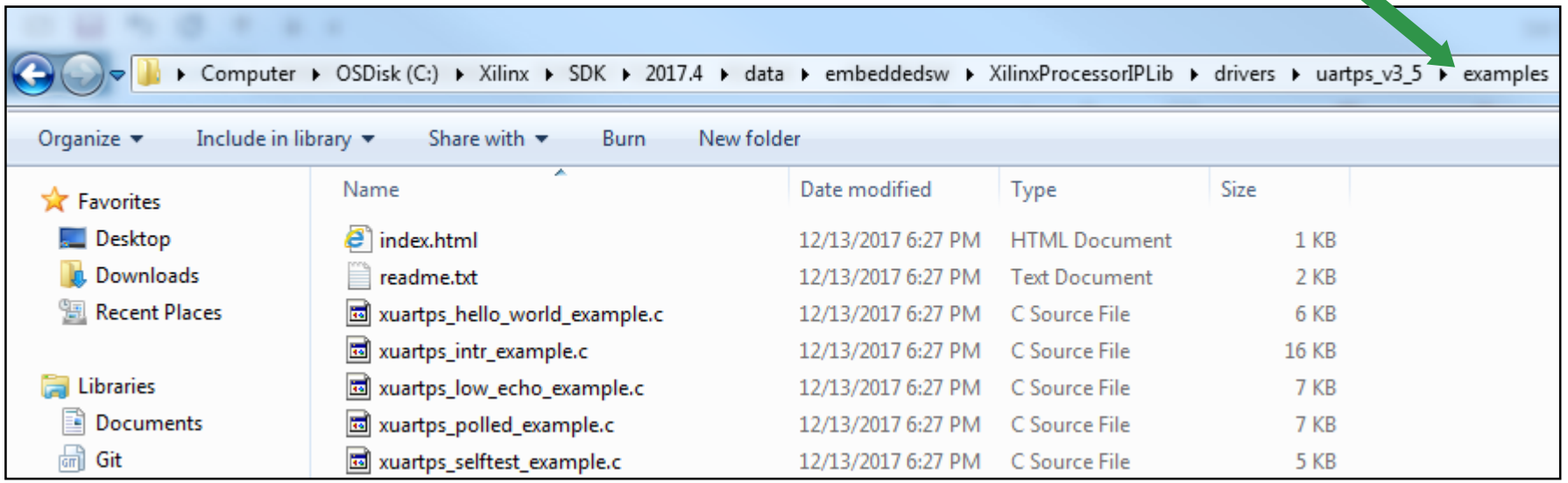
Example Code



Xilinx provides example code for nearly every included peripheral driver

Example Applications for the driver uartps_v2_2

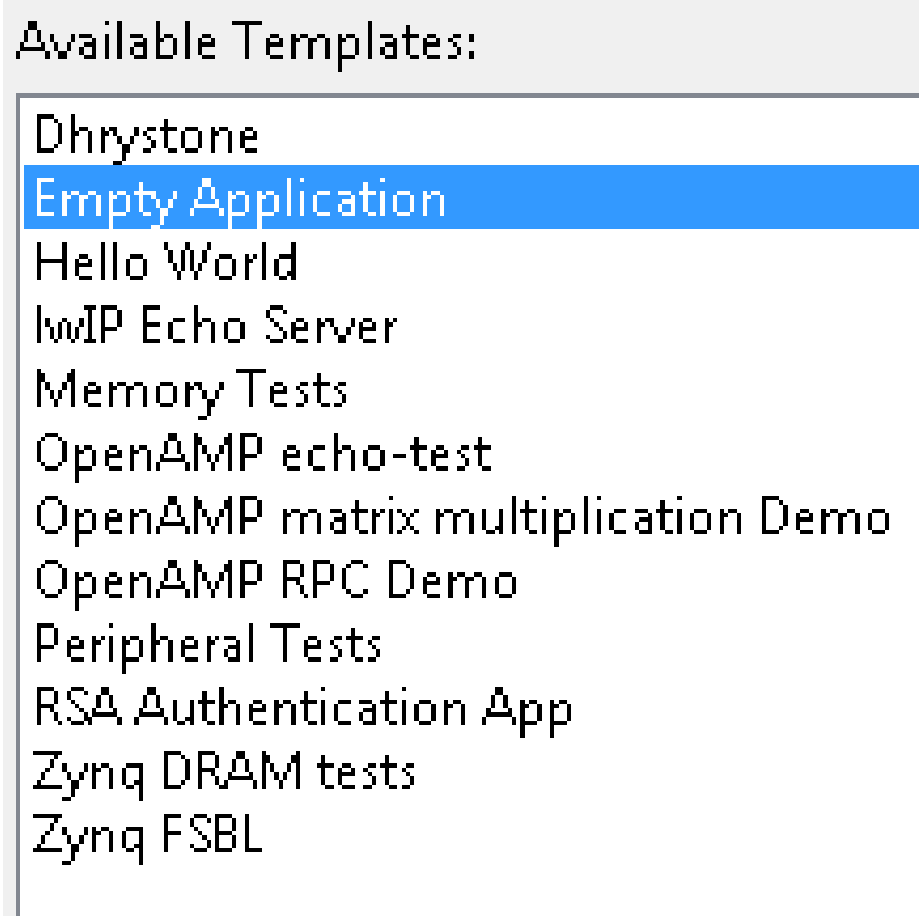
- xuartps_hello_world_example.c (source)
- xuartps_intr_example.c (source)
- xuartps_low_echo_example.c (source)
- xuartps_polled_example.c (source)
- xuartps_selftest_example.c (source)



AVNET

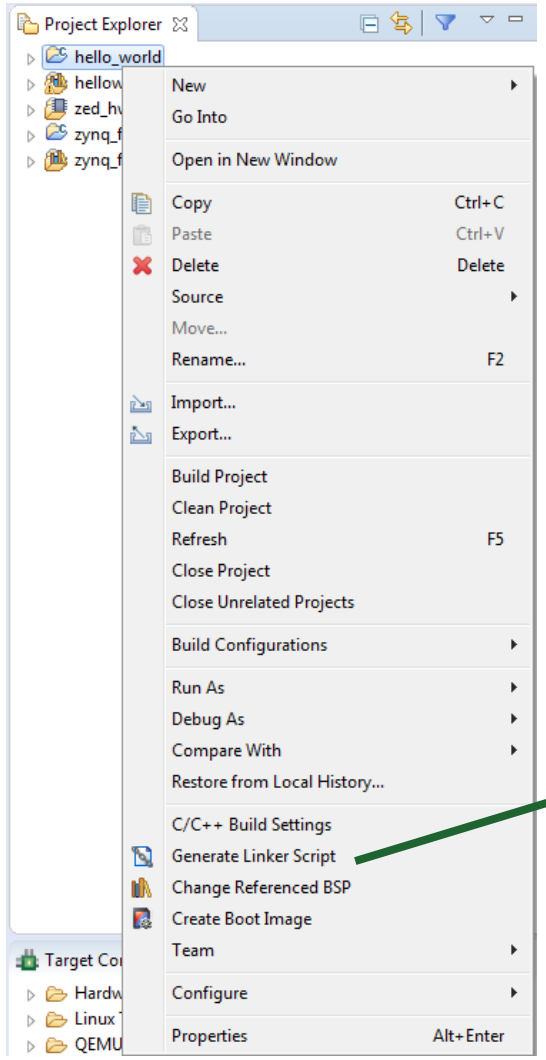
New C Project Sample Applications

- Empty Application
  - Will setup the project references
  - Can be used to import an existing application
- Hello World
  - Directs stdout output to PS UART
- lwIP Echo Server
  - Simple example using the lwIP Ethernet stack
  - Not useful for MiniZed, works with wired Ethernet
- Memory Tests
  - Tests all available memories
- Peripheral Tests
  - Basic peripheral tests
- Zynq FSBL
  - First stage bootloader

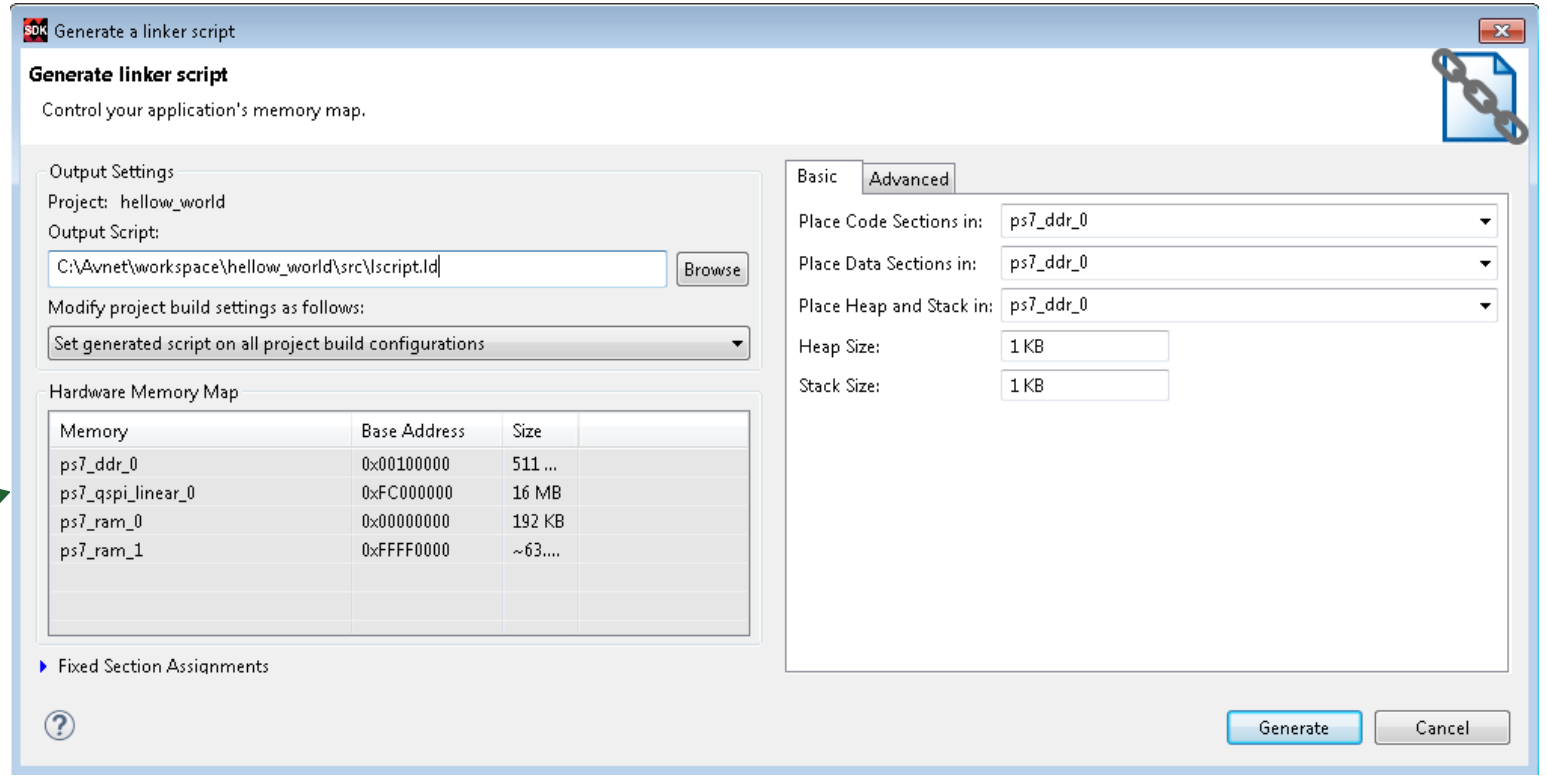


AVNET

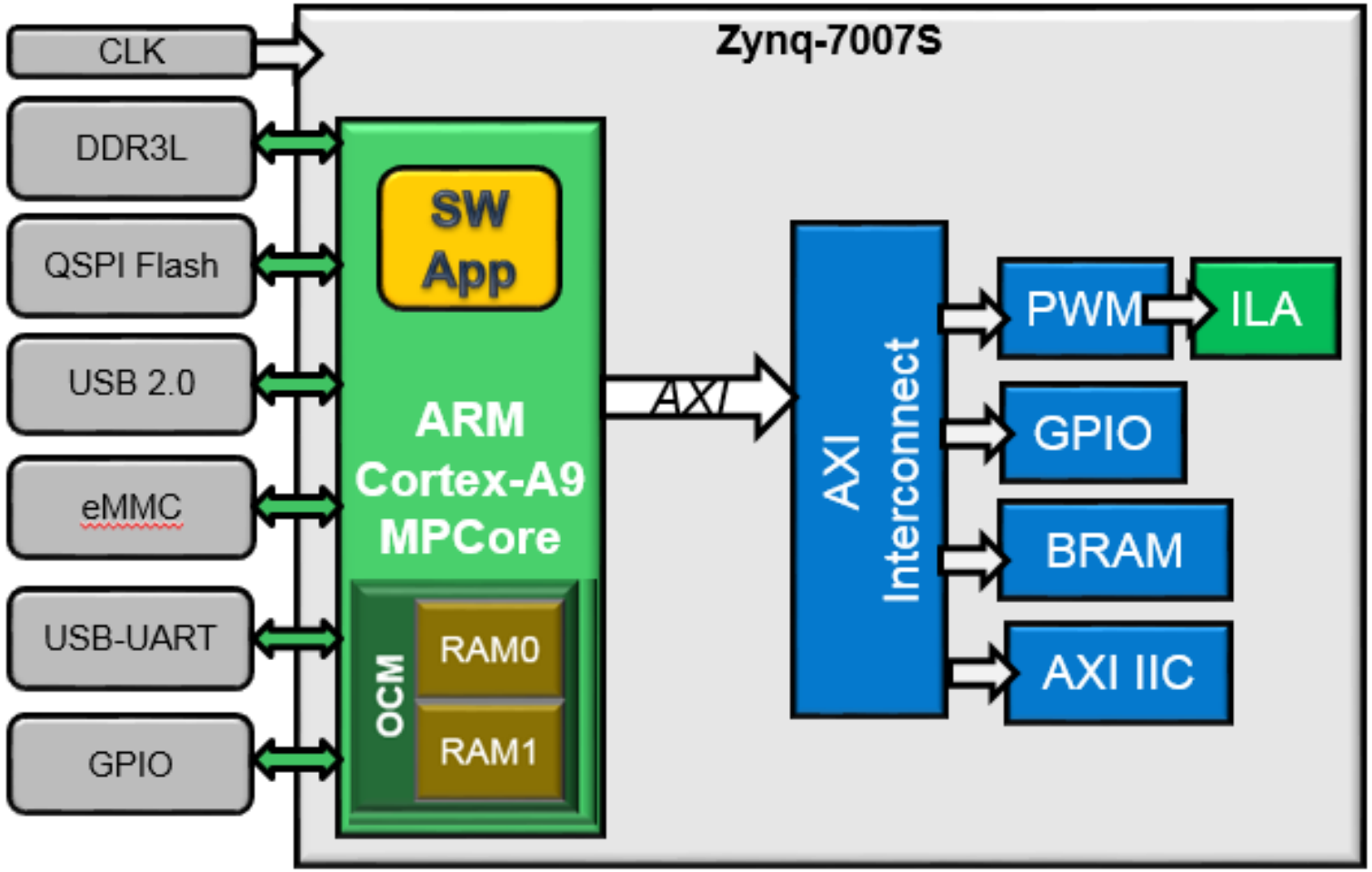
Linker Script Generator



- Provides an easy way to modify the linker script
  - Advanced tab allows individual section assignments
  - Modify the Heap and Stack sizes



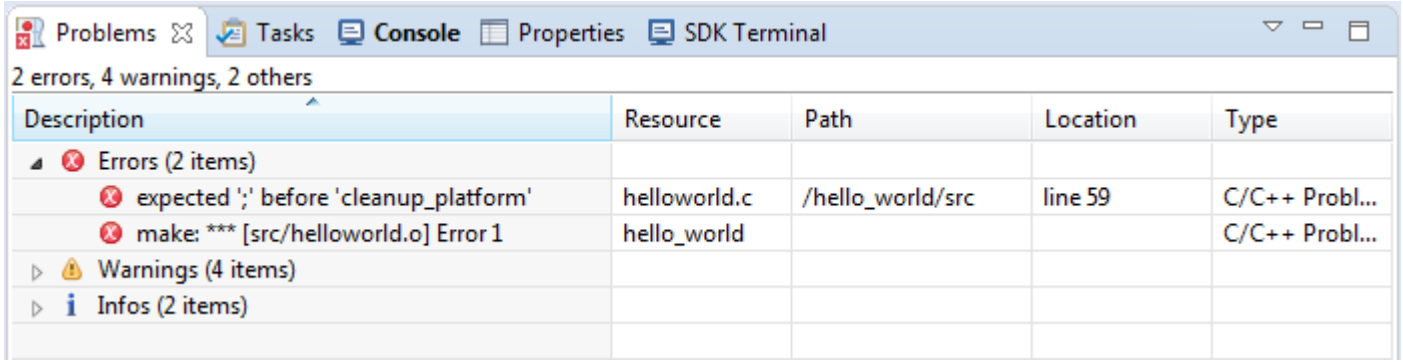
Memories in the Lab Hardware Platform



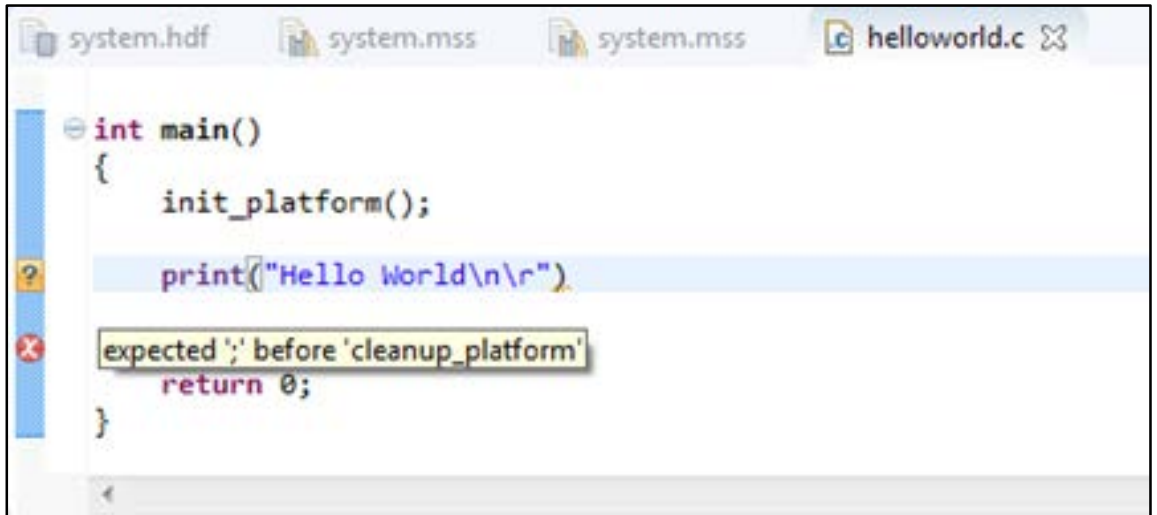
AVNET

Build Errors and Warnings

- Errors and Warning are displayed
  - In the Problems View, double clicking problem takes you to Editor View line of code

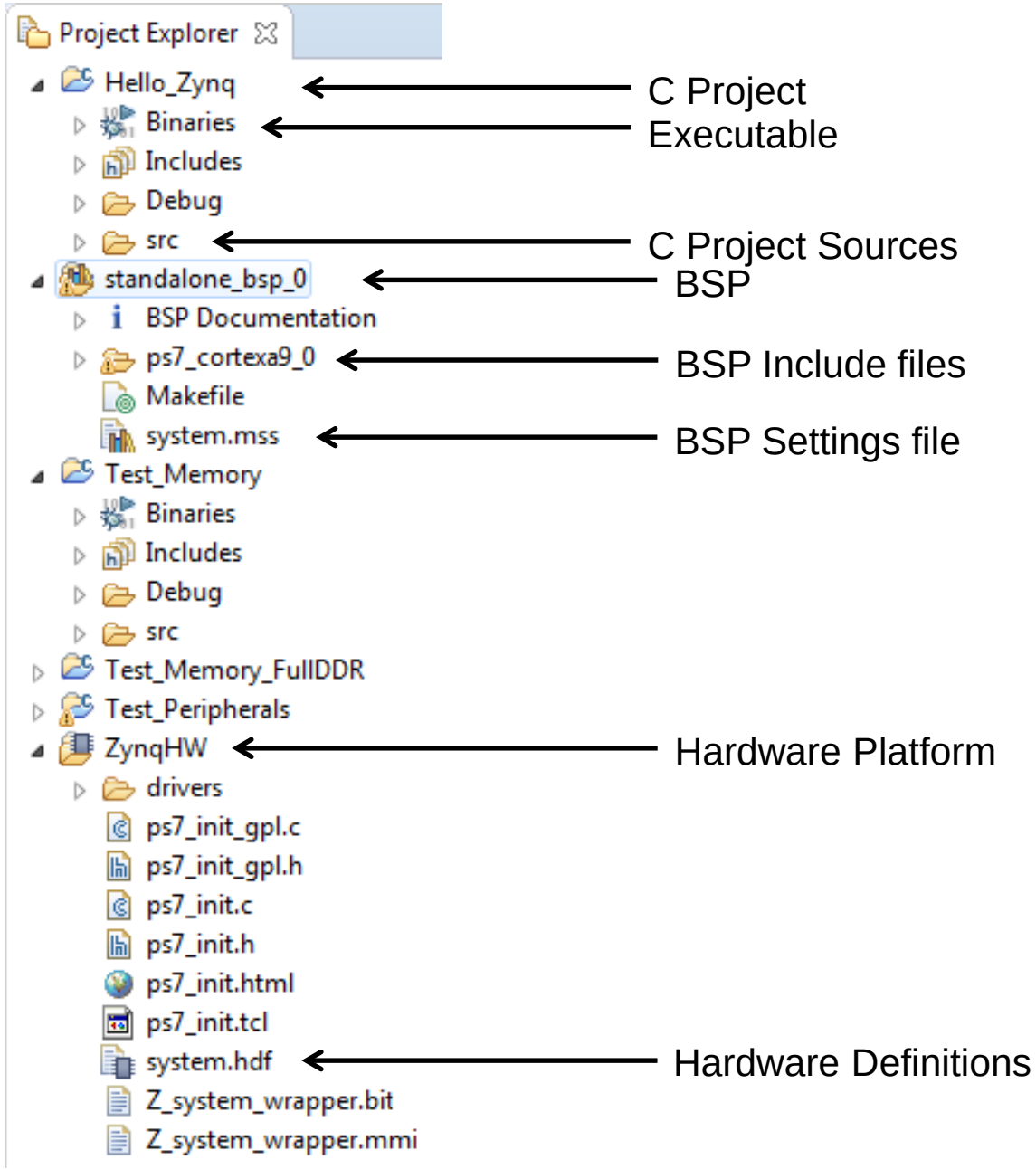


  - In the Editor margin, hover over marker to see the related warning or error



AVNET

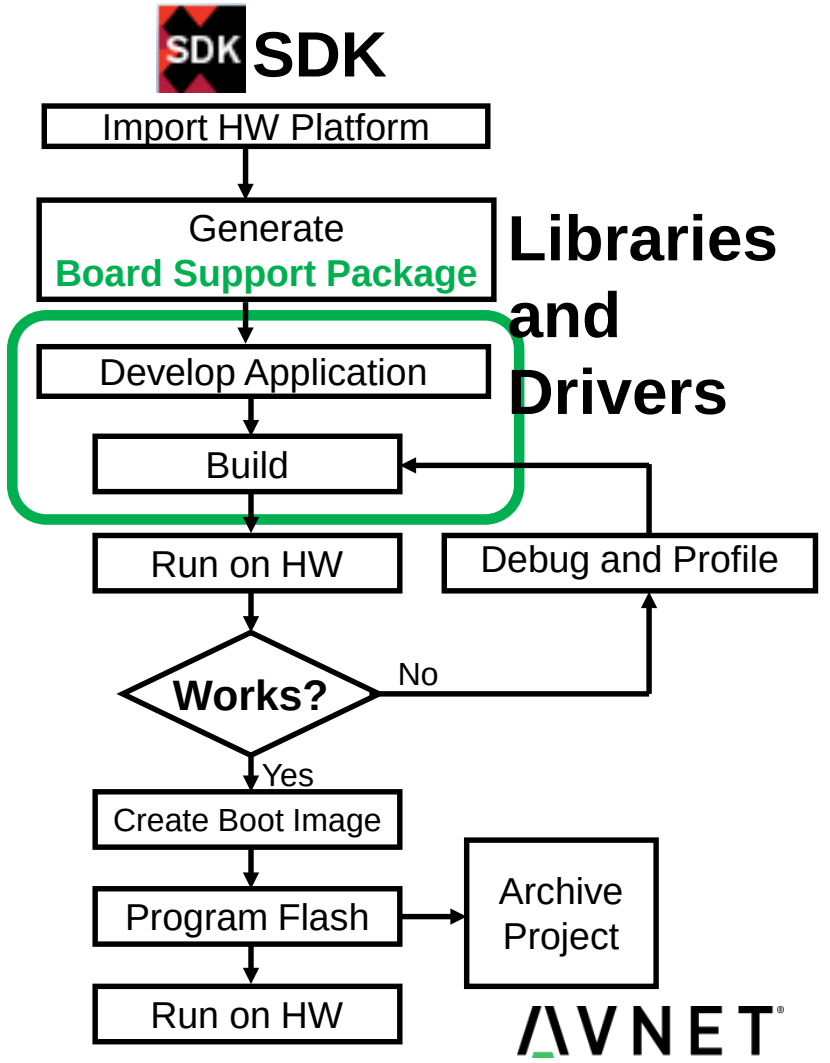
Project Explorer View



- Lists all the projects alphabetically
- One Hardware Platform Definition Project
- Multiple BSPs allowed
  - With different settings or libraries
- Multiple C projects
- All the files can be opened with the editor
  - Hardware specification file shows the memory map, core used, and contains links to the datasheets
  - BSP Settings file lists the device drivers used and contains links to the drivers documentation

Lab 4 – Developing Applications

- Create a new Xilinx C/C++ application
- Use Example Code
  - Hello World
- Use a built-in template
  - Memory Test Application
  - Peripheral Test Application
- Explore the code
- Use the Generate Linker Script tool to modify a linker script



Questions

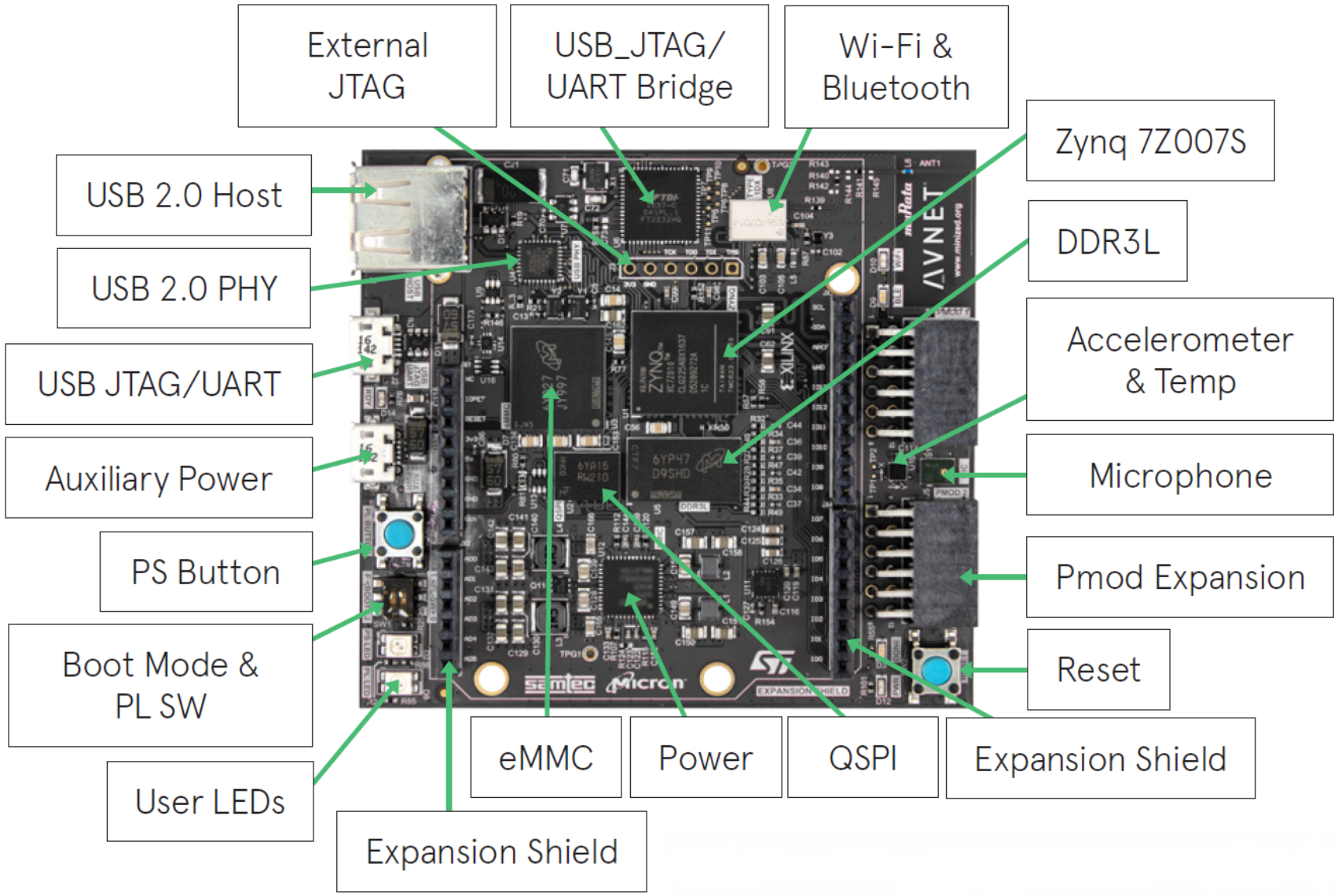
- You've been assigned a task to develop code to test reading and writing to the PL BRAM (peripheral axi_bram_ctrl_0 in this hardware platform). What do you do?
  - When starting to work with a new peripheral and its associated driver, the best place to start is the example code provided by Xilinx. Go to the system.mss Overview. Find the BRAM peripheral. Click on the Import Examples link to get to the example code.
- To what memory region(s) is the Test_Peripherals application targeted?
  - DDR3 for everything – Code, Data, heap, and stack
- To what memory region(s) is the Test_Memory application targeted?
  - ps7_ram_0_S_AXI_BASEADDR for the Code, Data, heap, and stack sections
- How much memory is tested by default?
  - 4096 bytes
- How many memories are tested? Which ones?
  - Three memories: DDR3, axi_bram, and ram_1
  - ram_0 is not actually tested as this is where the code, data, heap and stack reside.

Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

Lab Hardware - MiniZed



MINIZED™

www.minized.org

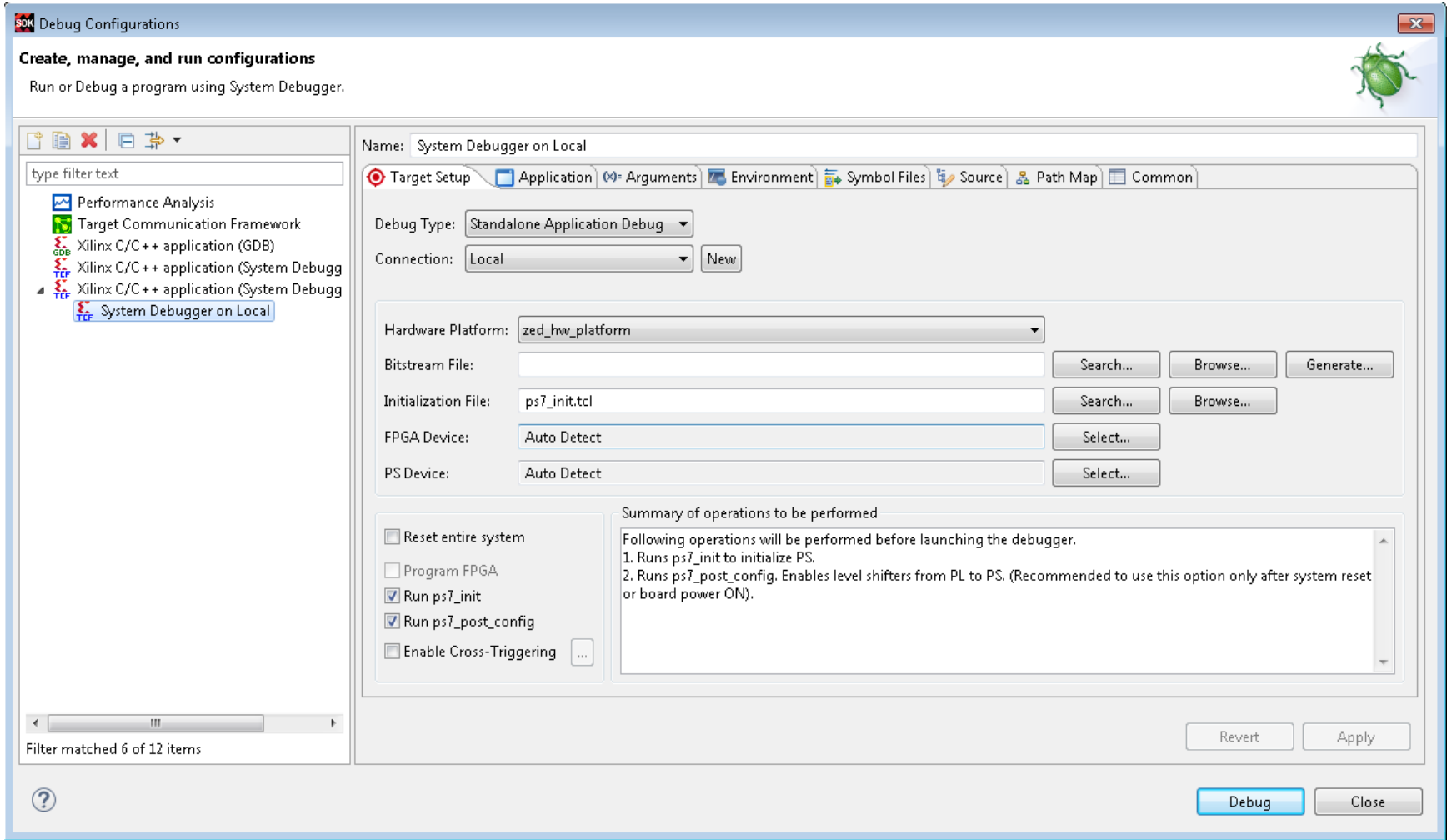
AVNET

Configurations

- Configuration = settings associated with running or debugging within SDK

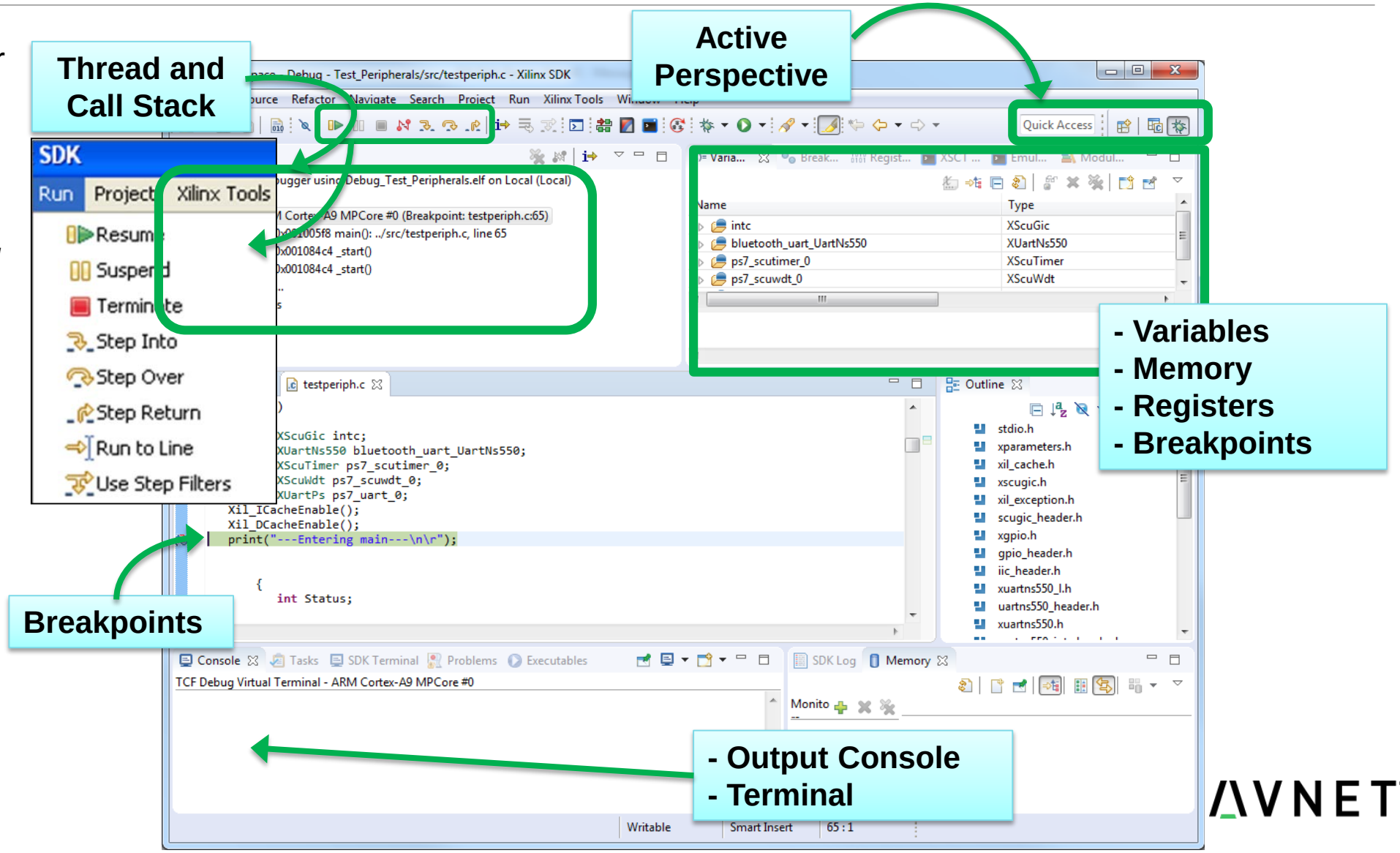
- Run and Debug options

- Set up for
  - Processor reset
  - Processor initialization
  - Download bitstream
  - Data download
  - Terminal
  - Profiling
  - Remote debug
  - Passing arguments



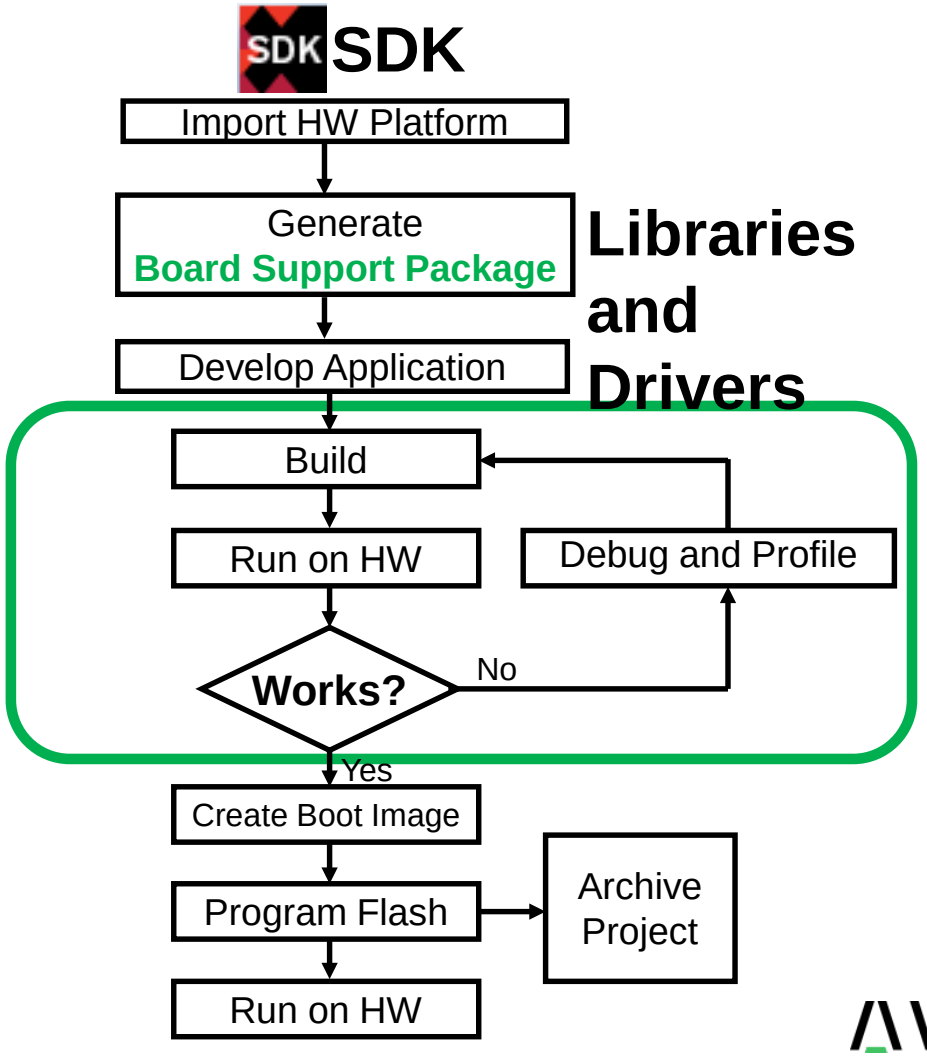
Application Debugging

- System Debugger for source and assembly
  - 3rd Party trace tools available – ***Debugging ARM Processor Systems*** online training



Lab 5 – Connecting Hardware

- Connect the hardware
- Download an application over JTAG
- Debug an application over JTAG



AVNET

Questions

- For what can the JTAG interface be used?
  - Read and write ARM registers
  - Configure the PL with a bitstream
  - Program attached QSPI Flash
  - Upload application code to OCM or DDR3L
  - Application debug
  - Performance analysis and profiling
- Under what conditions must the hardware platform first be downloaded into the PL?
  - If the application only uses the Zynq PS, then programming the PL is not necessary. If the application makes use of something in the PL, then the PL must be programmed first.
- How does the ARM get initialized when running an application from SDK?
  - ps7_init.tcl that was provided with the hardware platform
- How much memory is tested by default?
  - 4 KB

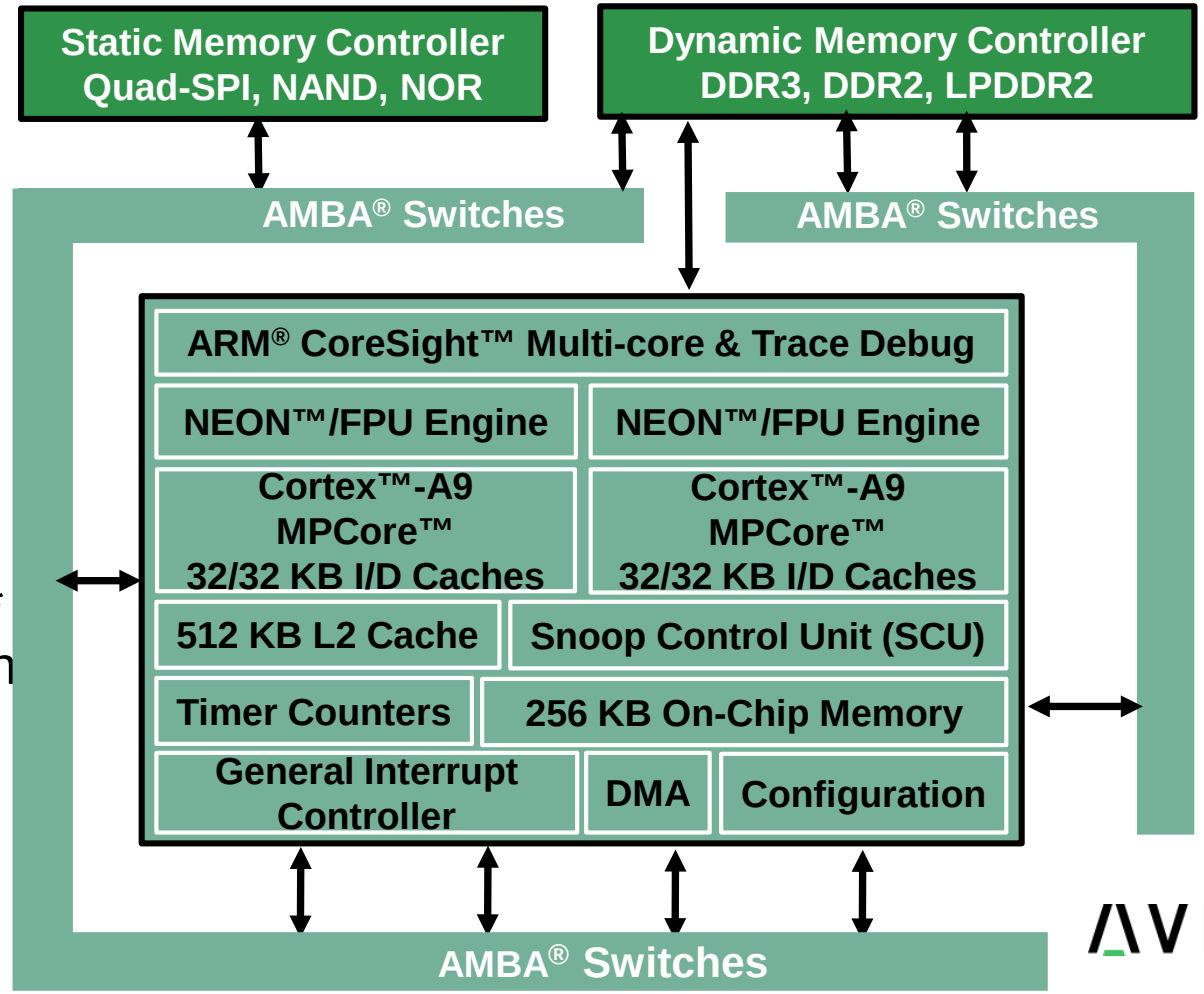
Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

Zynq-7000 Configuration and Boot

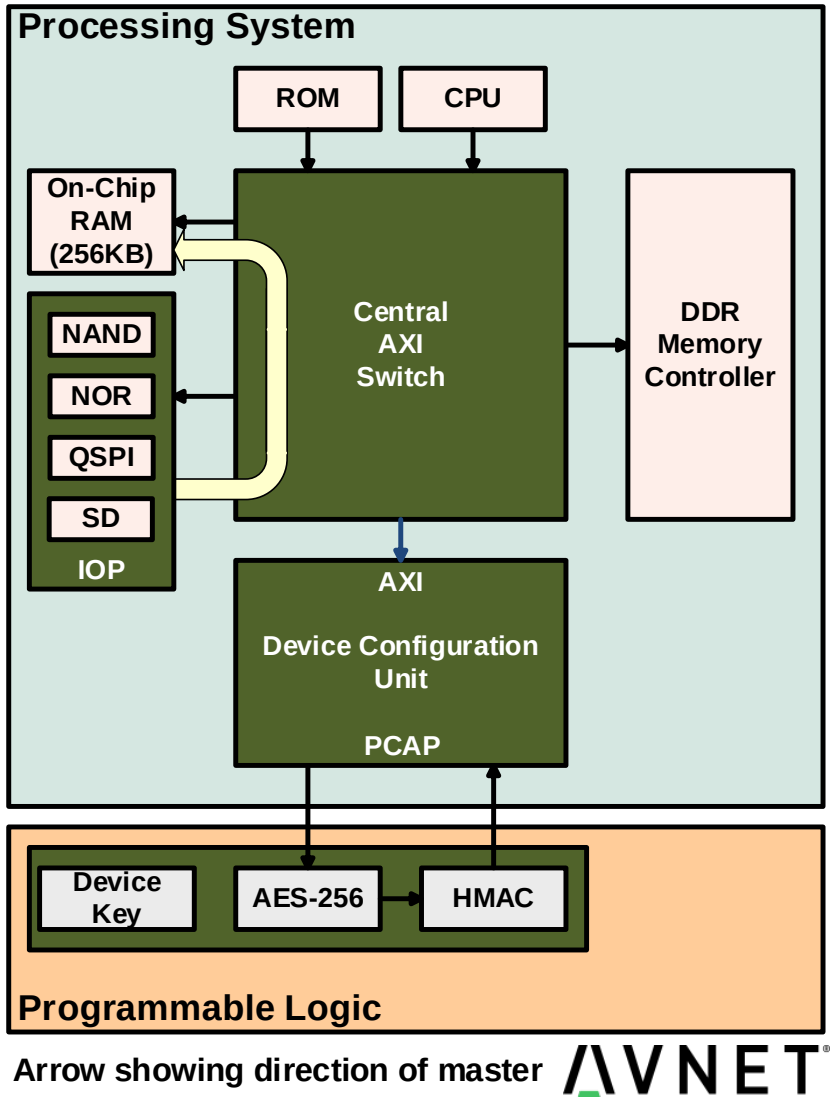
- Processor First! CPU configures the PS and PL
  - Standalone PL configuration (without PS configuration) is not supported
  - Configuration under external host control is also possible via JTAG
- Two boot modes
  - Secure boot
  - Non-secure boot
- Four master boot methods (secure or non-secure boot)
  - QSPI (16MB, 50MB/Sec)*
  - NOR (64MB, 20MB/Sec)*
  - NAND (tested up to 1GB, 10MB/Sec)*
    - * cannot be used in the same design
  - SD (Up to 32GB)
- One slave boot method (non-secure)
  - JTAG for debug and development



AVNET

Non-Secure Boot – Stage 0

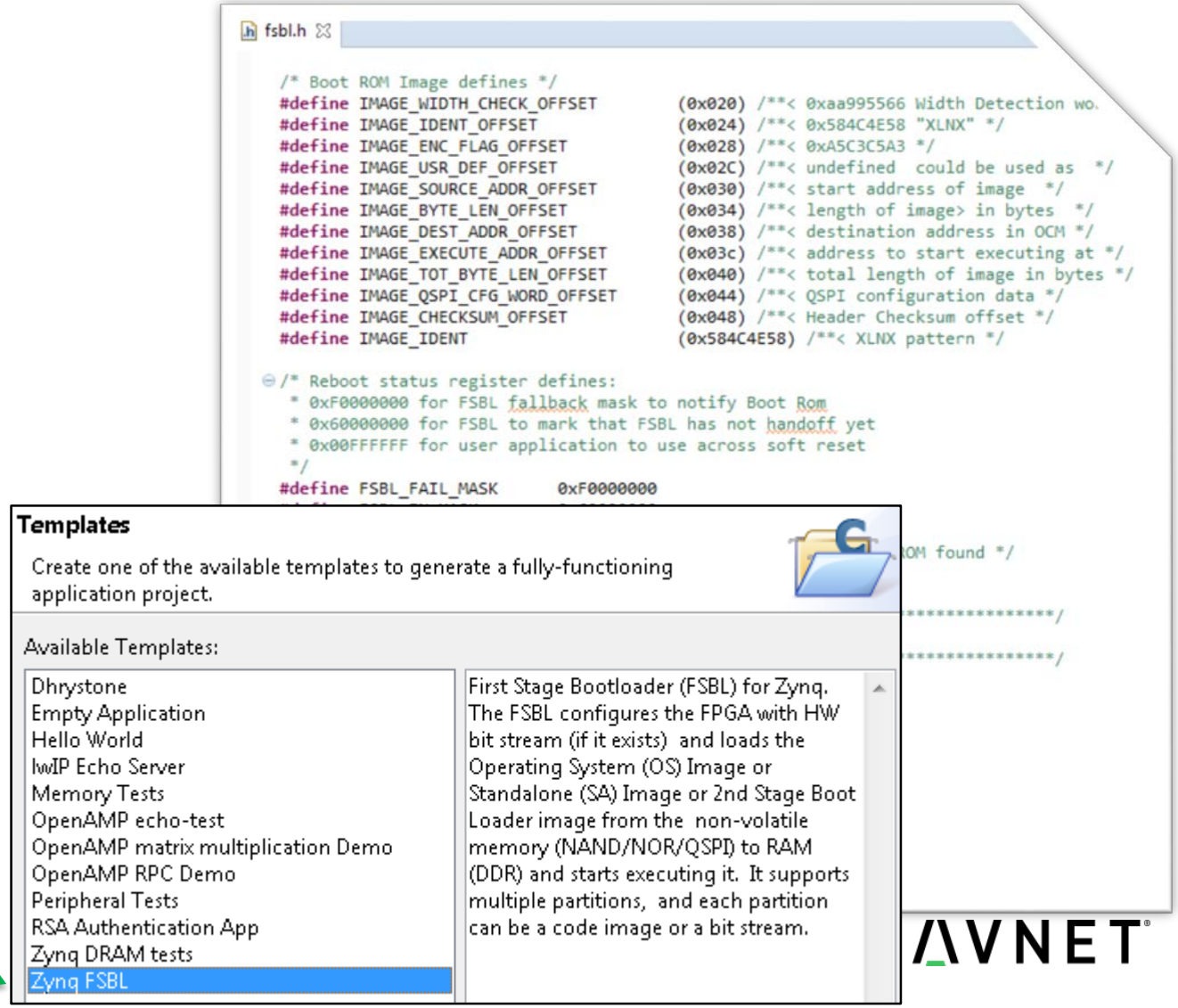
- CPU starts executing code from ROM
  1. Initializes the Cortex-A9 CPU 0
  2. Checks CRC on ROM code
  3. Reads the **boot mode pins** to determine Stage 1 boot mode
  - Can boot from QSPI, NAND, NOR, SD card, or JTAG
  4. Typically, a **first stage boot loader** is read from external non-volatile memory and copied into the OCM (192KB max)
  - If the Execute In Place (XIP) feature is enabled, first stage boot executed directly from QSPI or NOR
  - Stage 1 boot header specifies the use of XIP feature



First Stage Boot Loader (FSBL)

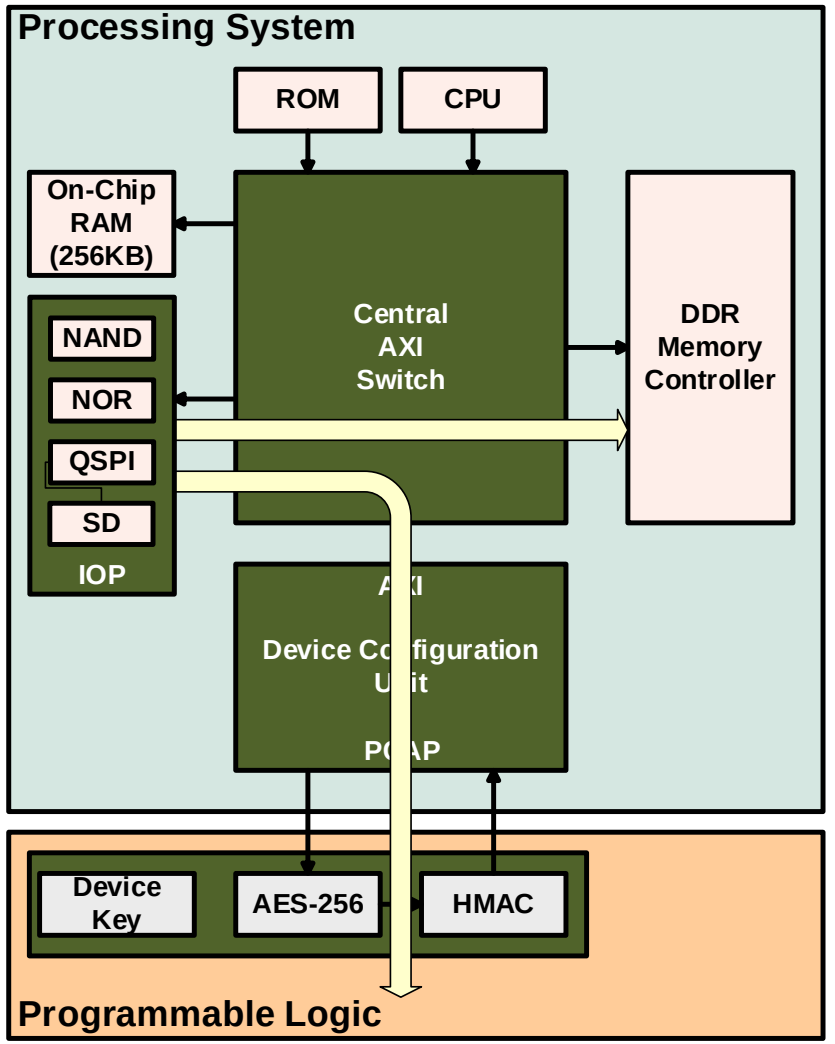
- First Stage Boot Loader Functions
  - Initialize Processing System blocks
    - PLL
    - External memory controller
    - MIO
  - Configure Programmable Logic with Bitstream (optional)
  - Provides for secure boot option
  - Execute application code

- Application created automatically directly from the SDK FSBL new project template



Non-Secure Boot – Stage 1

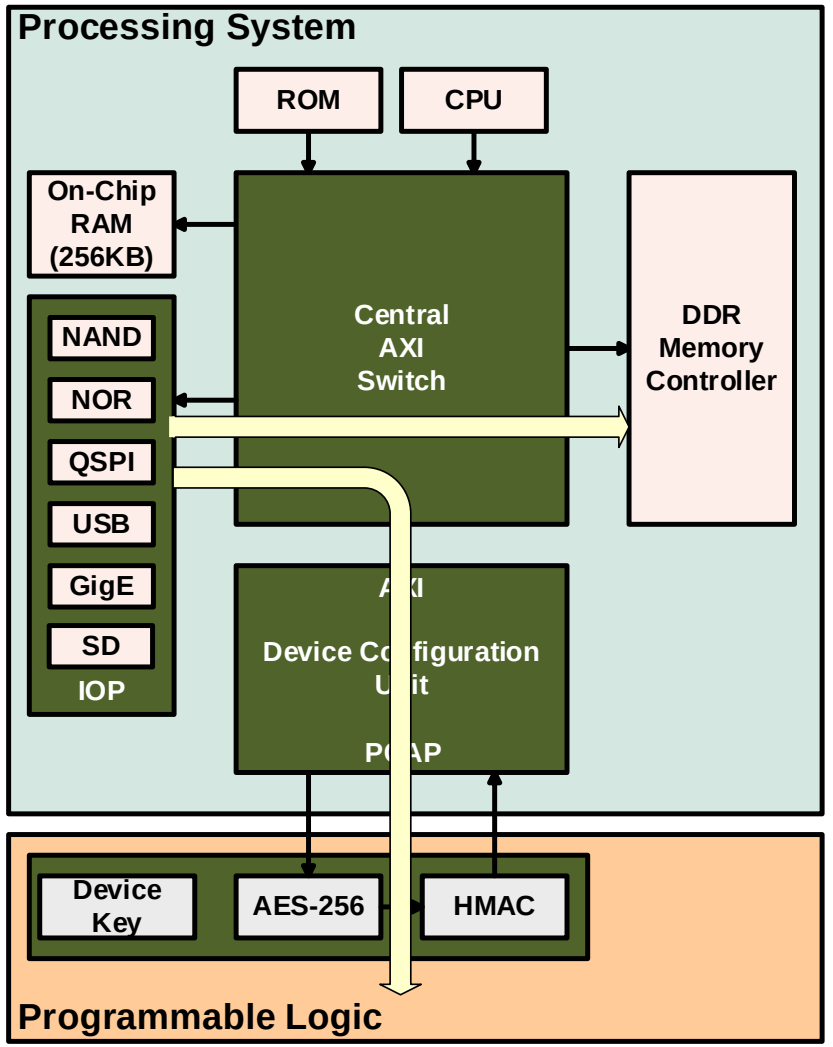
- CPU starts executing code from OCM (FSBL)
  - Code execution can also be performed directly from QSPI or NOR if the Execute In Place (XIP) feature is used
    - 192KB code size limit is removed
    1. DDR controller, clock generation module, and selected peripherals are initialized
    2. PS application and/or second stage boot is loaded into the DDR memory
    3. Optionally, bitstream is loaded from non-volatile memory and PL is configured
    4. Second stage boot is optionally enabled



Stage 2 Boot

- Stage 2 boot loader such as U-Boot runs from the DDR memory
  - Responsible for loading OS kernel
  - OS image can be sourced from any of the following peripherals
    - NAND
    - NOR
    - QSPI
    - USB
    - GigE
    - SD
  - Optionally configure PL in stage 2

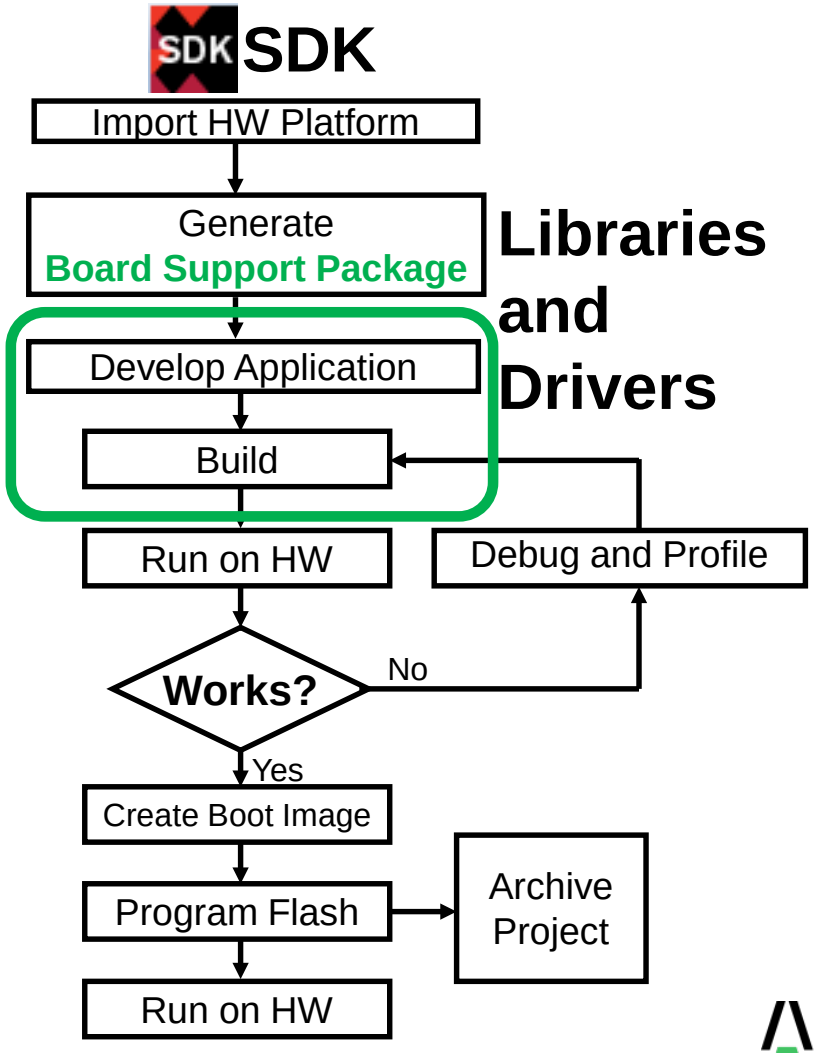
Get U-Boot source code from Xilinx's Github Repository
Learn more from http://wiki.xilinx.com/zynq-uboot



AVNET

Lab 6 – FSBL

- Create the FSBL
- Analyze the code



AVNET

Questions

- What is the size of the FSBL application with the Debug configuration?
  - ~ 159596 bytes
- What is the target memory for the FSBL?
  - ps7_ram_0_S_AXI_BASEADDR, which is the 192 KB on-chip RAM
- What is the size of the FSBL application with the Release configuration?
  - 153788 bytes => 5808 bytes (or 3.9%) smaller
- In what file was the FSBL main() function?
  - main.c
- Which files included all of the ARM register settings? Where did they originate (prior to FSBL generation)?
  - ps7_init.c and ps7_init.h files
  - The imported Hardware Platform Definition. Remember Lab 1?

Agenda

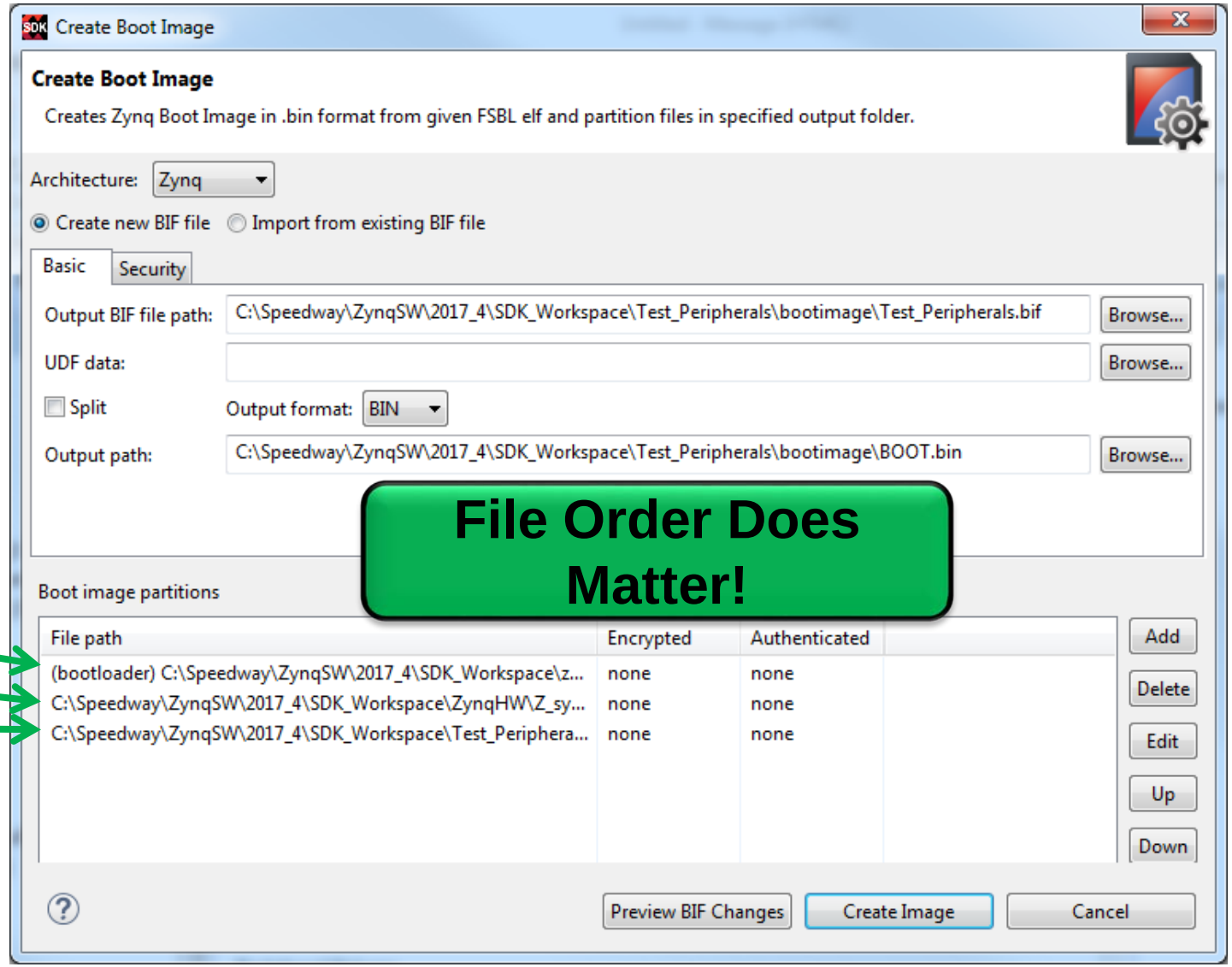
Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

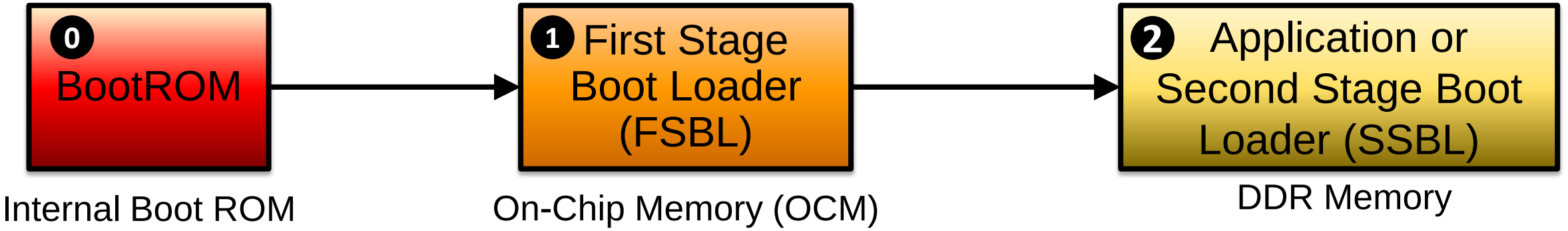
Create Zynq Boot Image in SDK

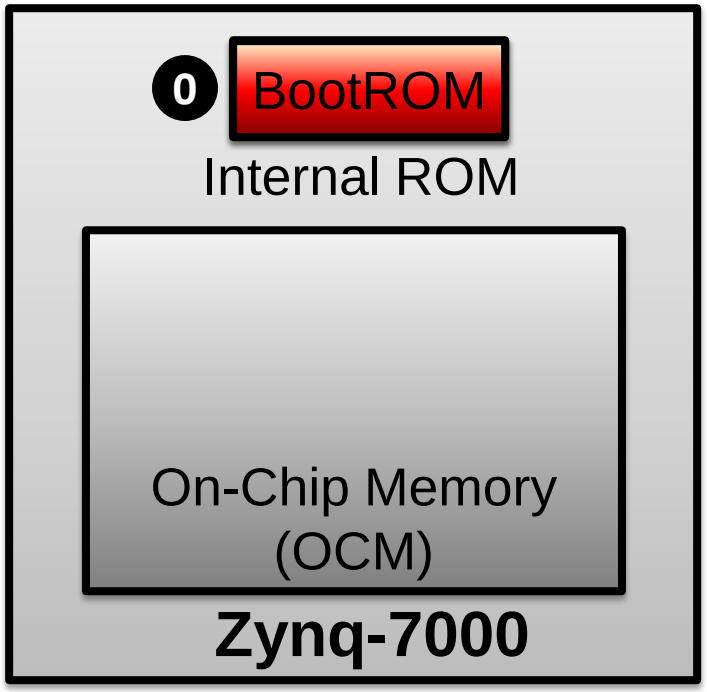
- **Graphical front end to command line bootgen**

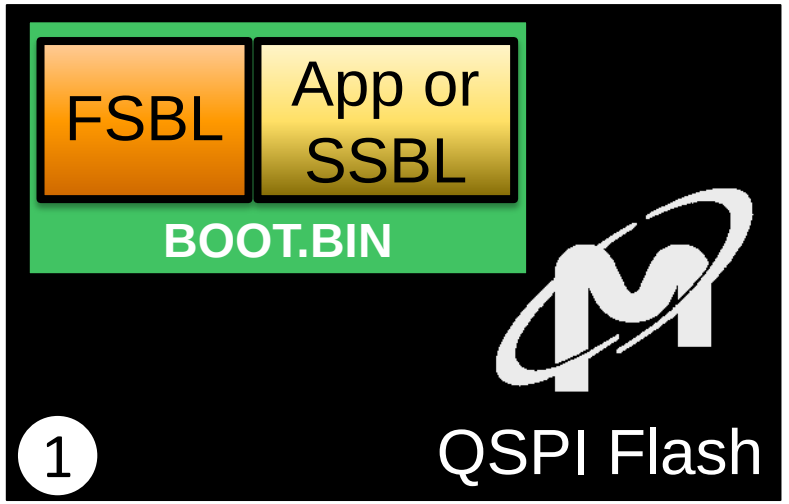
- **Partitions**
  1. **FSBL**
  2. **Bitstream**
  3. **Application**

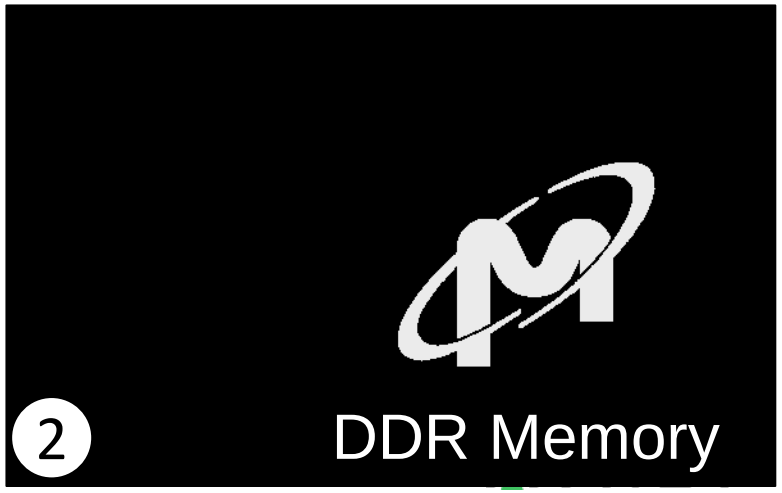


Boot Process Overview for Zynq



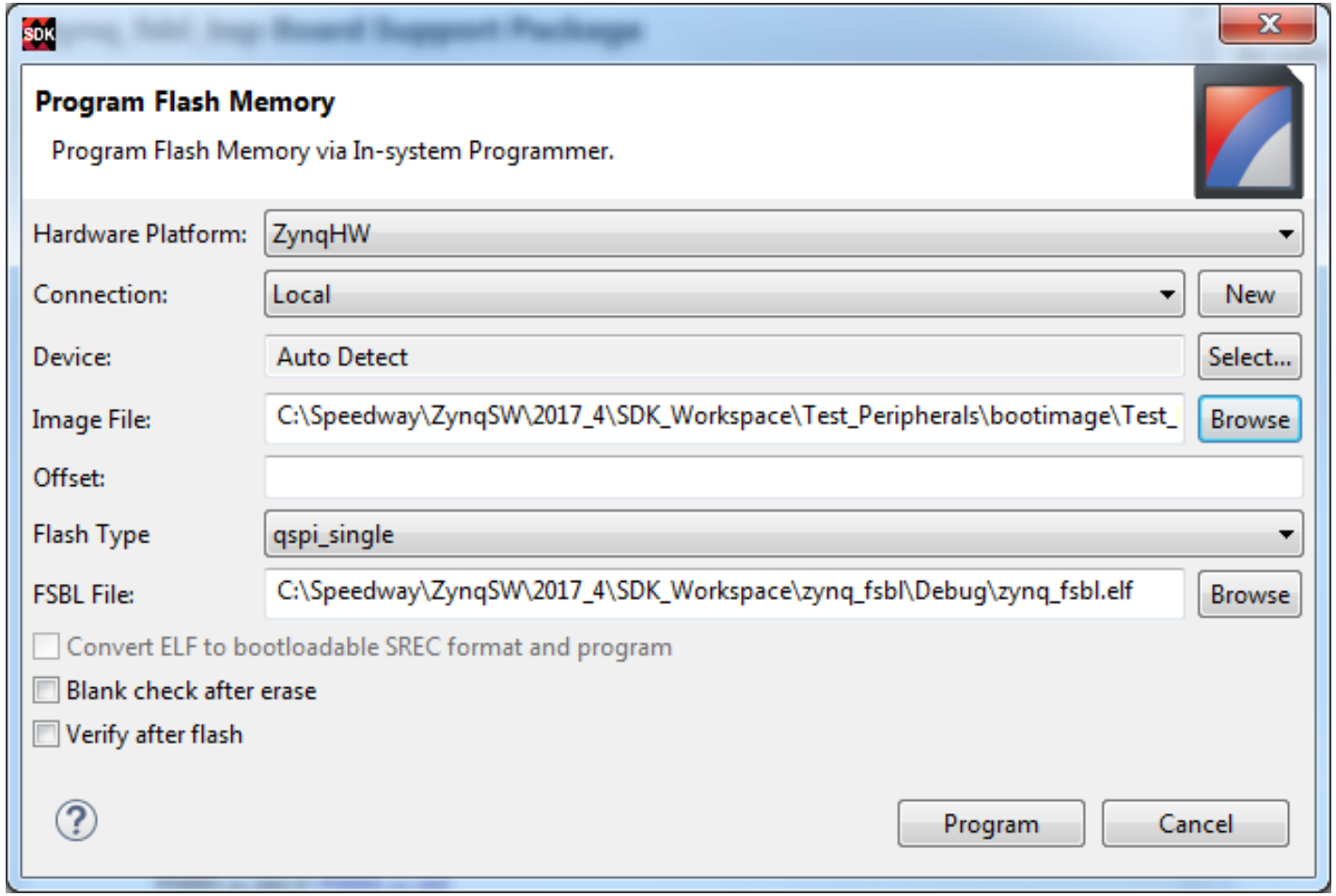






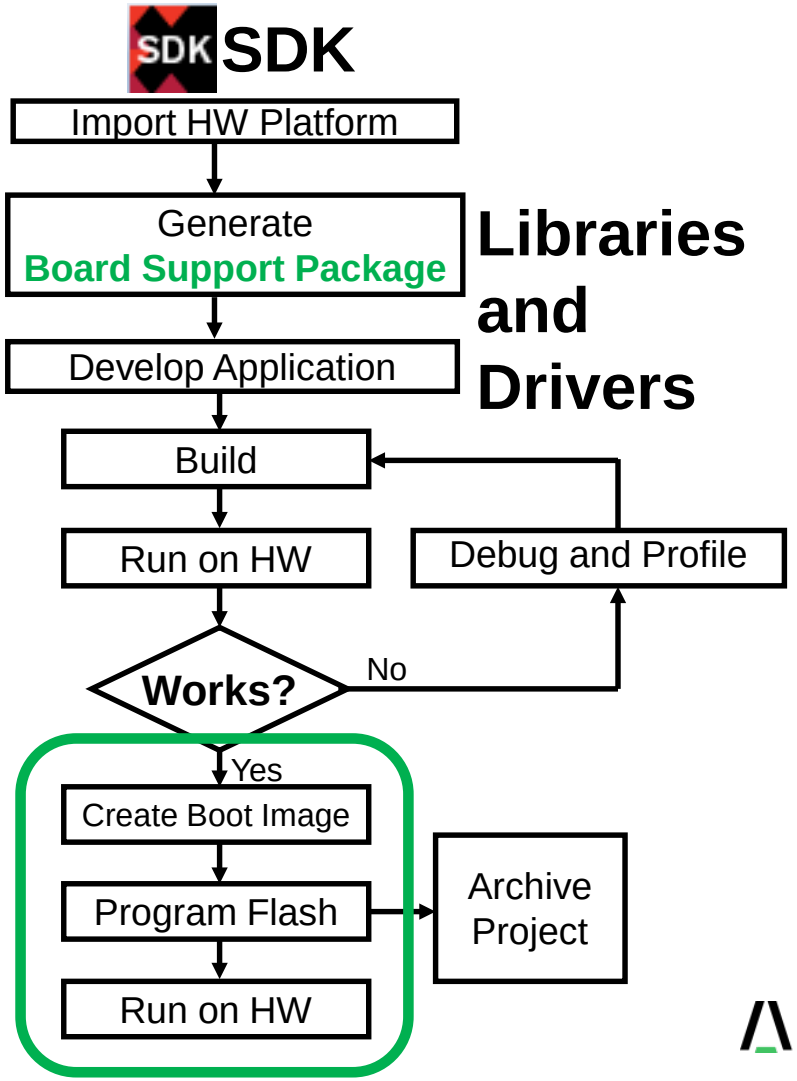
SDK Flash Programming Utility

- Supports multiple non-volatile types
  - Single QSPI will be used in the labs
- Interprets BIN and MCS format files
- Capable of programming at offset
- Erase check and write verify options
- FSBL must be specified



Lab 7 – Flash Programming and Configuration

- Create the boot images
- Copy to QSPI
- Boot from QSPI



AVNET

Questions

- Is the order of the boot images critical? If so, list the order.
  - YES! FSBL ELF first, then bitstream, then application ELF

- True or False? The Create Zynq Boot Image tool creates a boot image for the QSPI Flash.
  - True. SDK uses a utility called *bootgen* to generate a boot container file that contains the FSBL, the Programmable Logic bitstream (optional), and the Standalone user application.

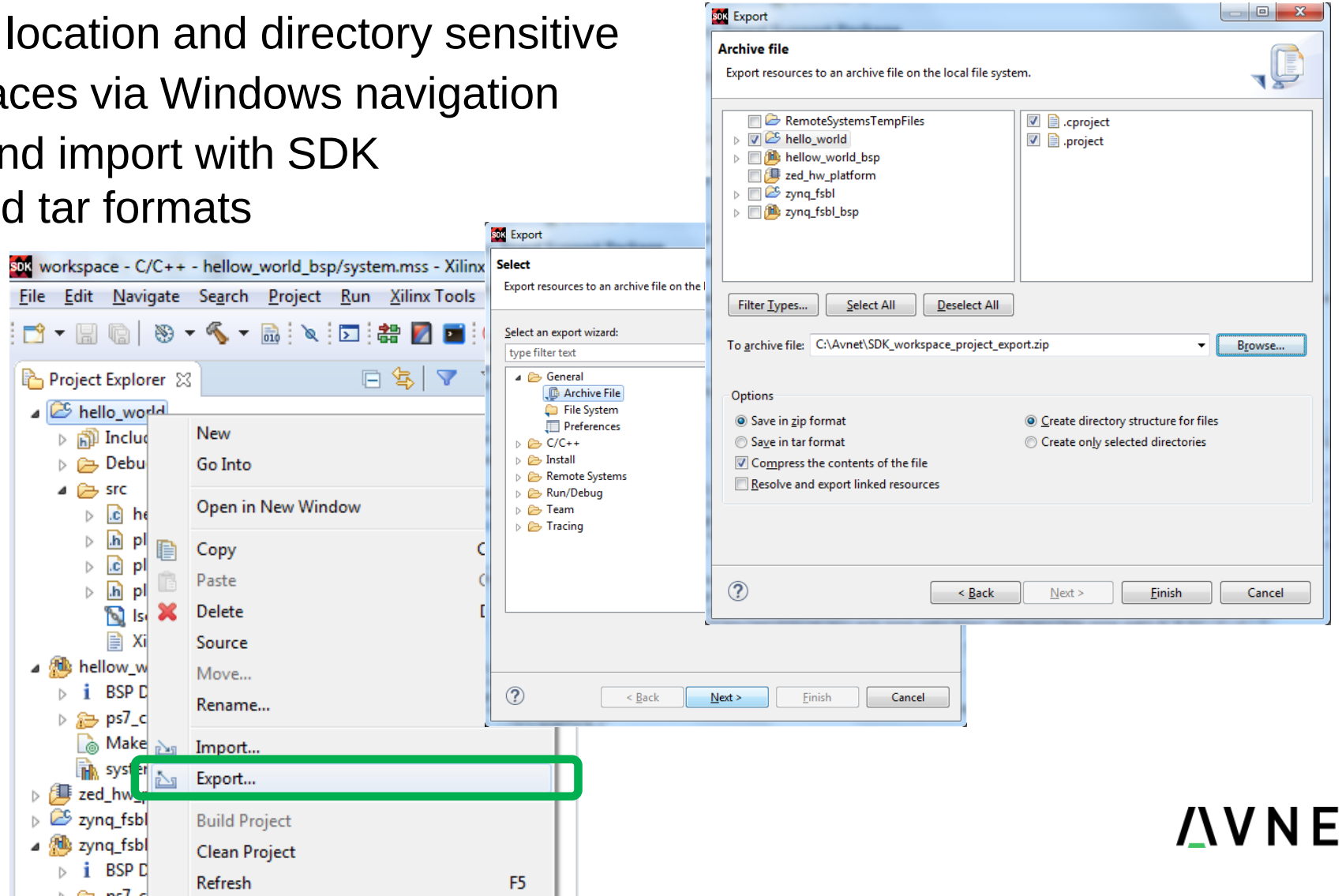
Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

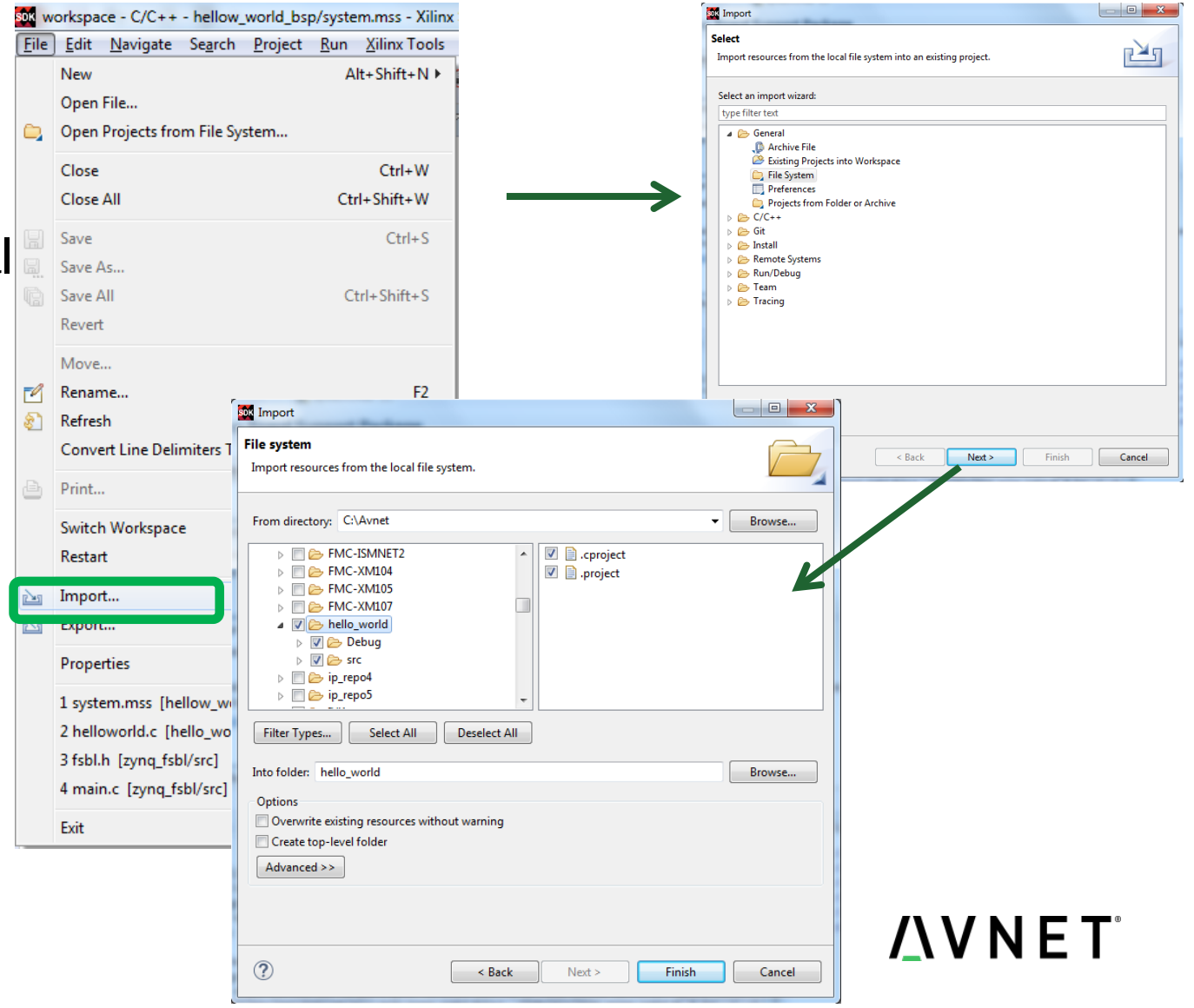
Sharing SDK Workspaces and Projects

- SDK workspaces are location and directory sensitive
- *Do not* move workspaces via Windows navigation
- Two ways to export and import with SDK
  - Archive file – zip and tar formats
  - Directory structure

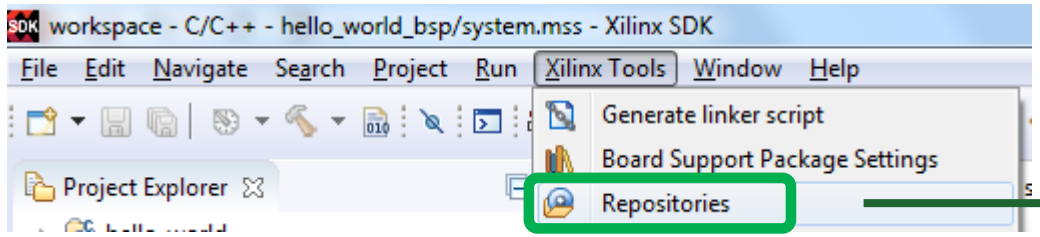


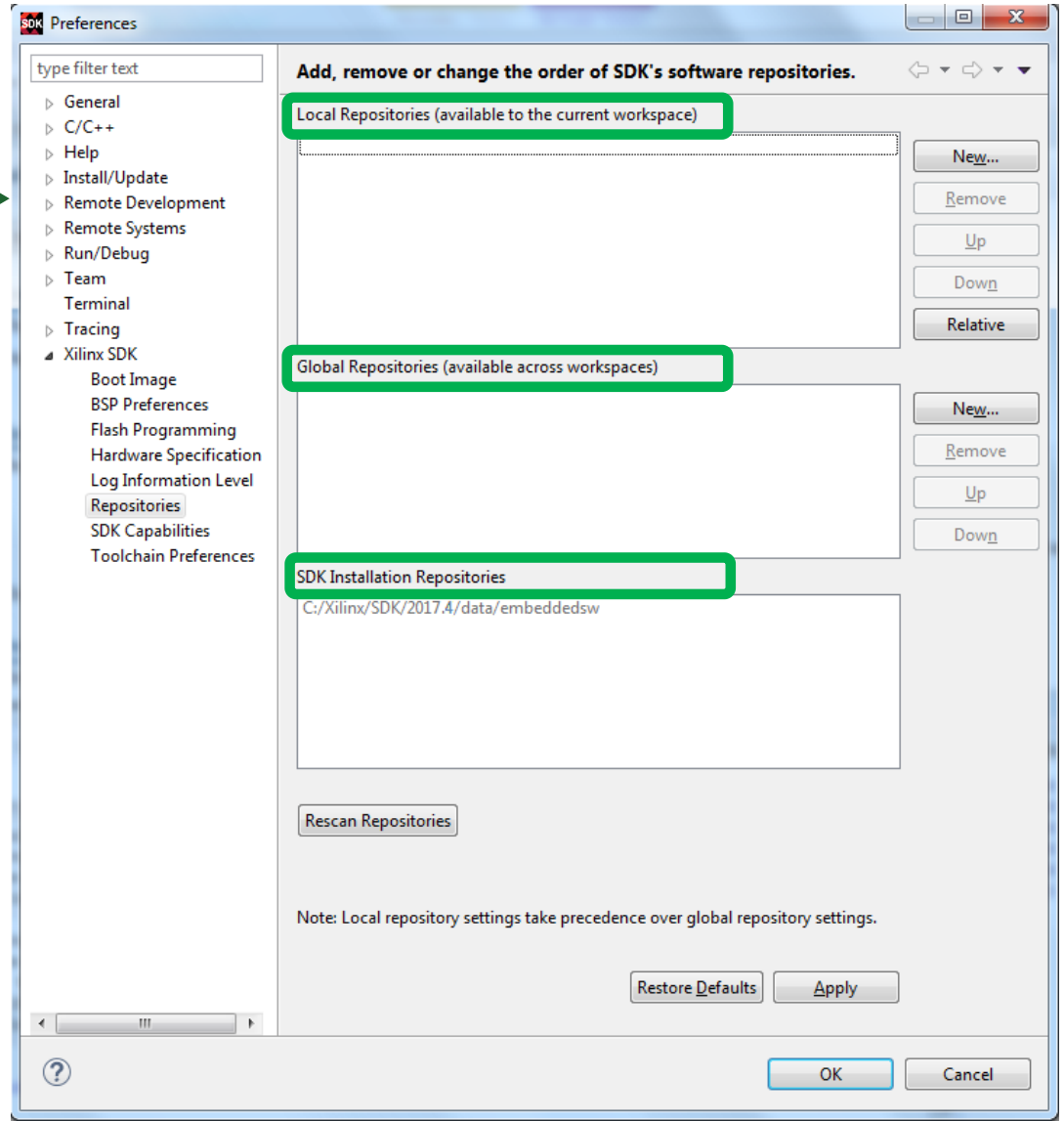
Importing C/C++ Source Files

- Create an Empty Application project
  - Set HW platform and BSP as needed
- Select Import in the File Menu
- Select File System option under General import source type
- Select the C/C++ project
  - Directories or individual files



Software Repositories

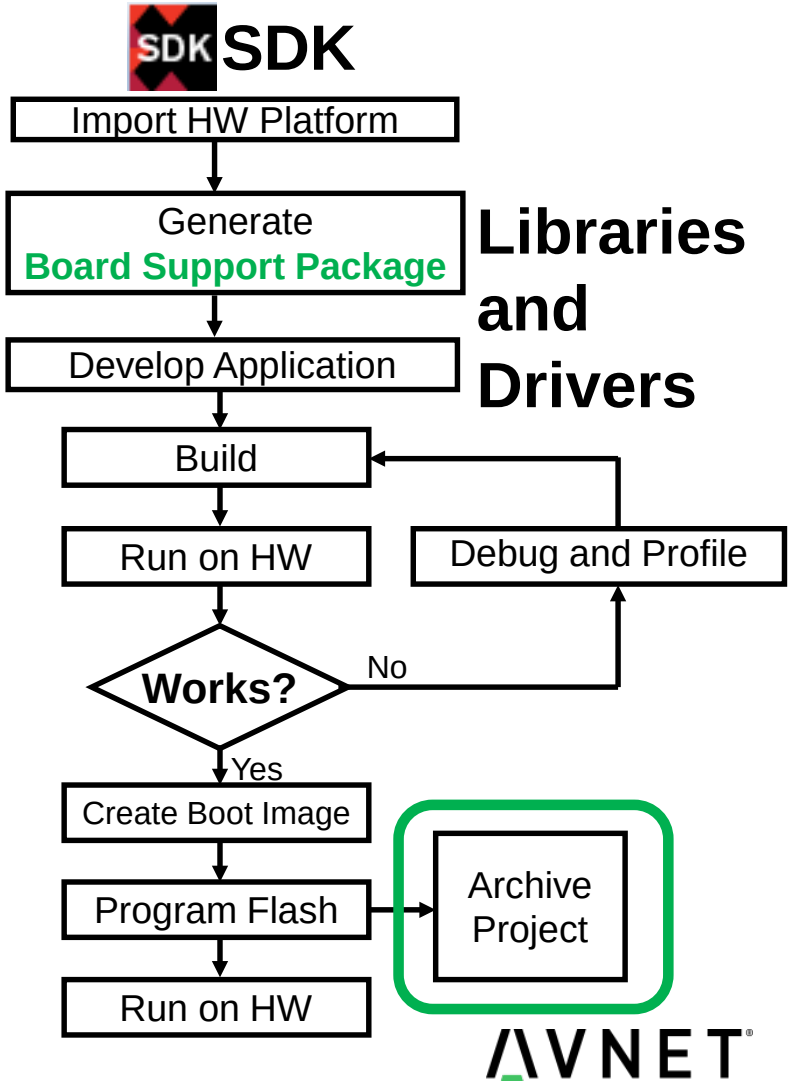




- May include
  - Custom Drivers
  - Custom Libraries
  - Custom Board Support Packages
  - Could be a modified Xilinx driver or library
  - Repository sub-directories
    - bsp
    - drivers
    - sw_services
- Prioritized

Lab 8 – Project Management

- Export your complete workspace
  - Hardware Platform
  - BSP
  - All applications
- Create a new Workspace
- Re-create your previous Workspace in the new Workspace



Questions

- What is the advantage of exporting your Workspace items as opposed to simply zipping the workspace?
  - If you zip and share your SDK workspace, there is a very good chance that it will not work when opened again. The SDK workspace is full of absolute paths, so unless the recipient unzips the SDK workspace to the exact same location, it won't fully work. It might appear to work initially, but it is likely not going to build properly.
  - If you export the workspace, it is fully transportable.

Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

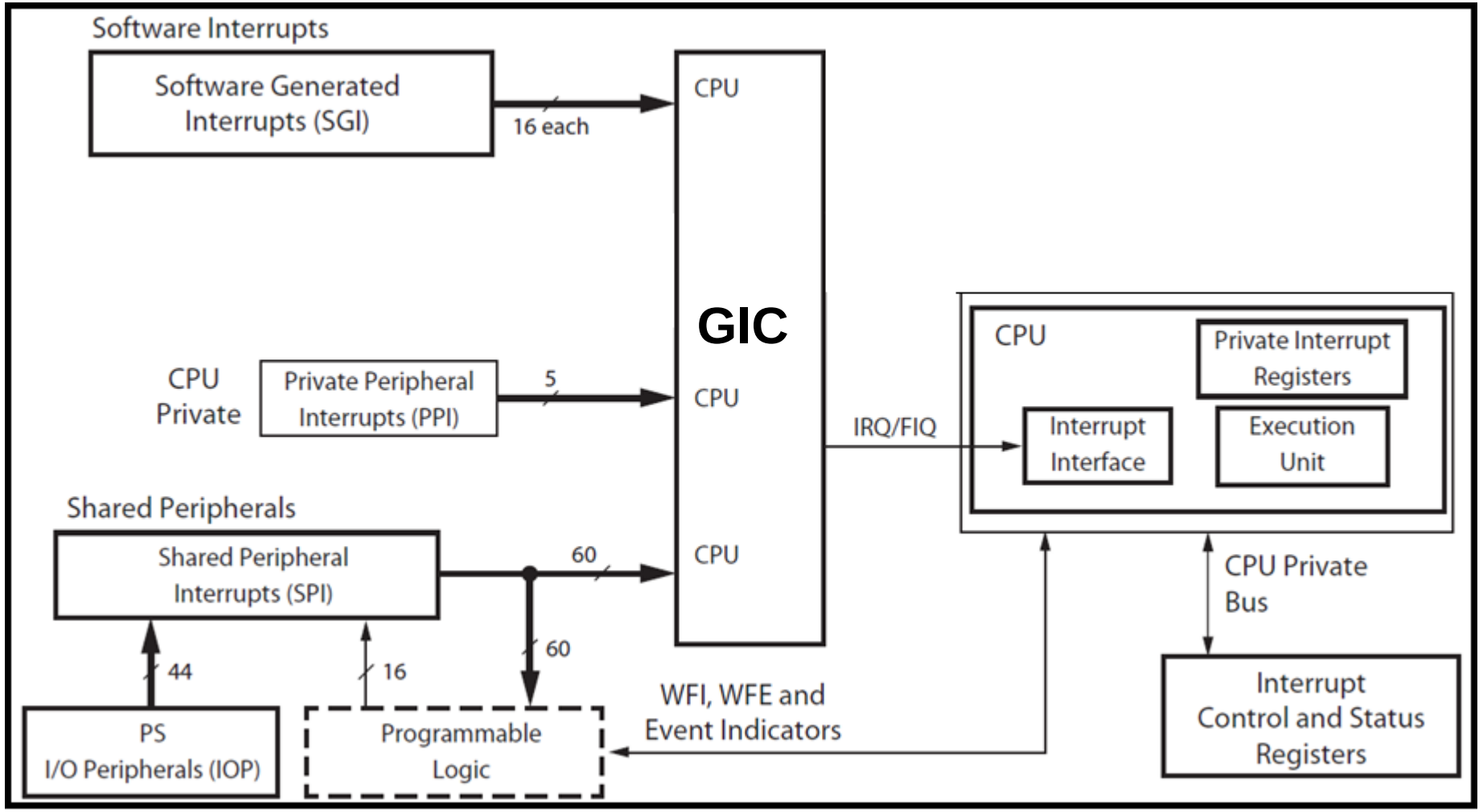
Interrupts and Exceptions

- Hardware interrupt
  - An asynchronous signal from hardware, originating outside the processor core, indicating a peripheral's need for attention

- Software Interrupt
  - A synchronous event in software indicating the need for a change in execution, also referred to as an exception

- Occurring events which need immediate responses

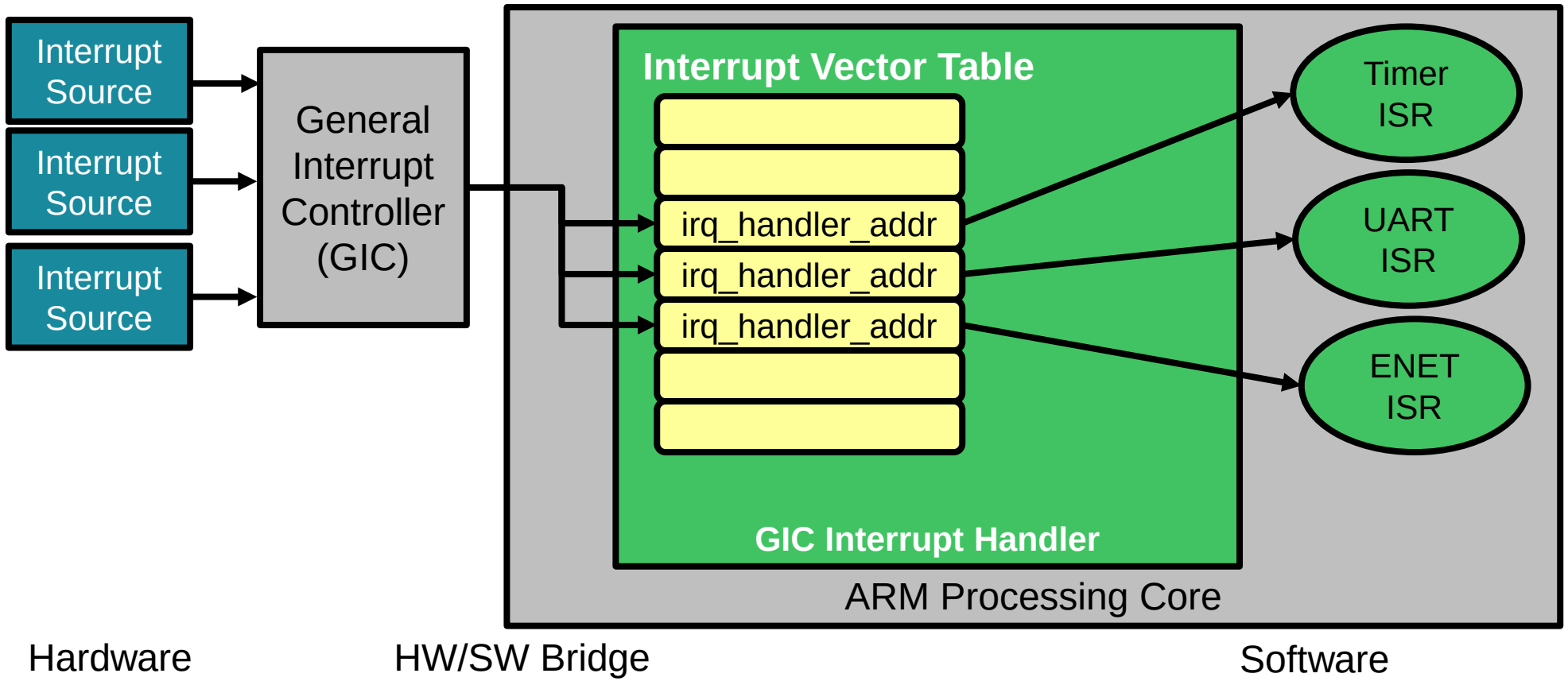
Zynq Interrupts - Generic Interrupt Controller (GIC)



AVNET

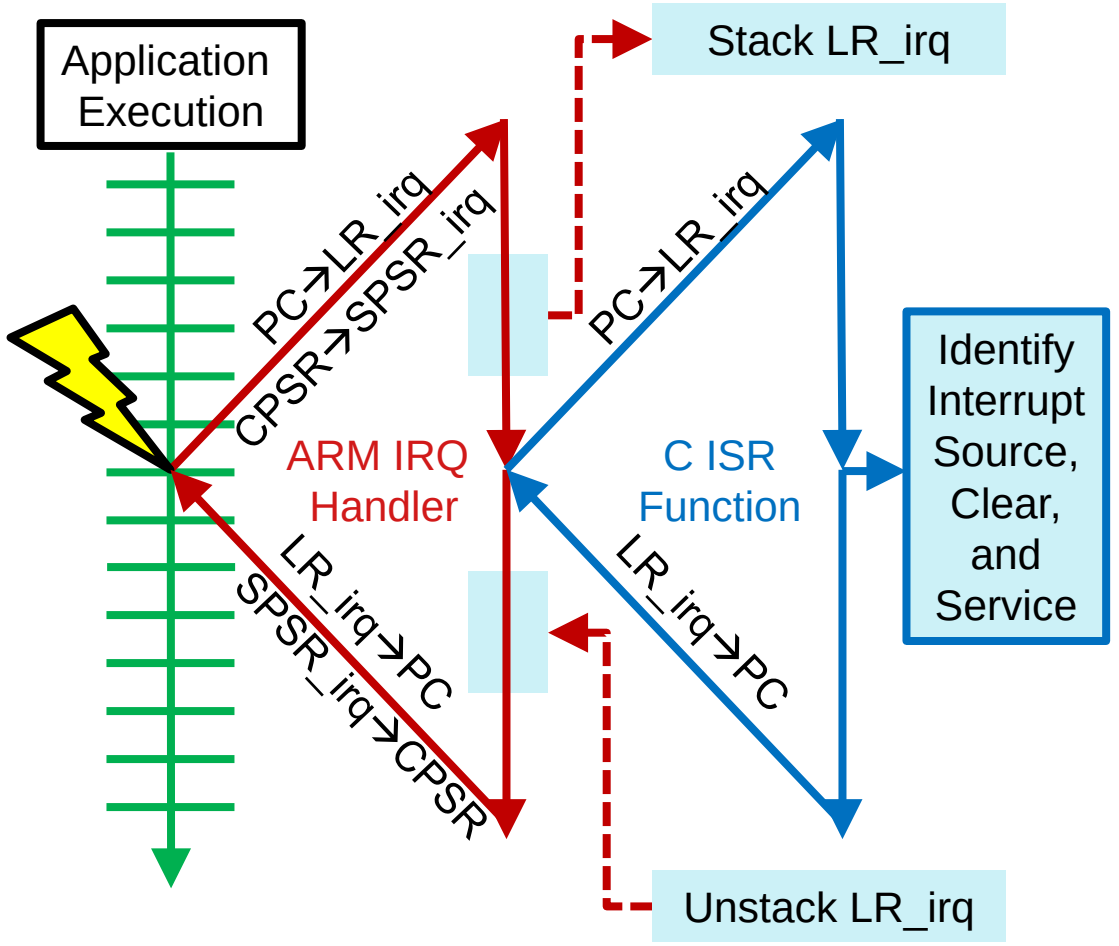
Interrupt Handling

- When an interrupt occurs, an Interrupt Handler should respond to the event by calling the appropriate ISR



Process of Handling an Interrupt

1. Interrupt is received, and current executing instruction completes
2. Save processor status
   - Copies CPSR into SPSR_irq
   - Stores the return address in LR_irq
3. Change processor status for exception
   - Mode field bits
   - Interrupt disable bits (if appropriate)
   - Sets PC to vector address
4. Execute top-level exception handler
   - Top-level handler branches to the appropriate registered ISR
5. Return to main application
   - Restore CPSR from SPSR_irq
   - Restore PC from LR_irq
   - When re-enabling interrupts change to system mode (CPS)



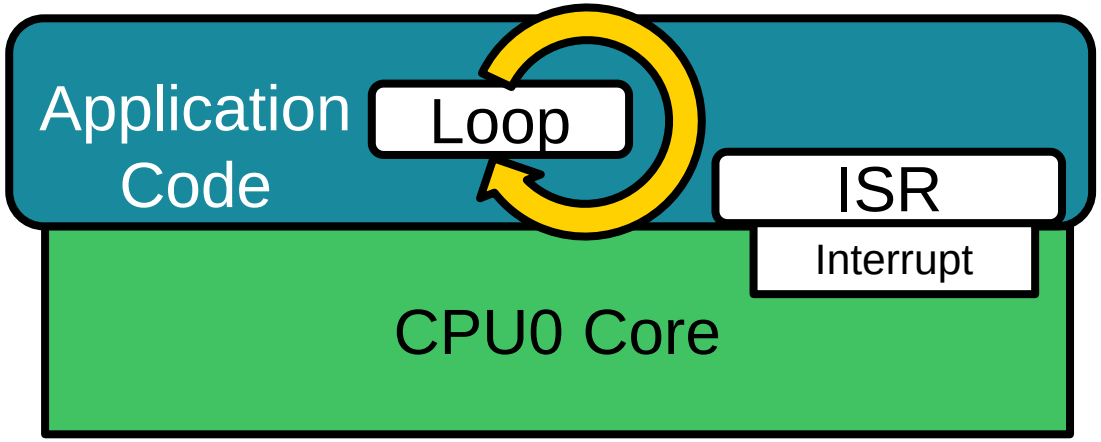
AVNET

ISR Tips and Tricks

- Keep the code short and simple; ISRs can be difficult to debug
- Do not allow servicing of other interrupts while in the ISR
- Time is of the essence!
  - Spend as little time as possible in the ISR
  - Do not perform tasks that can be done in the background
  - Use flags to signal background functions
- Make use of the callback argument (or parameter) passed in ISR registration
- Make use of the provided interrupt support functions when using IP drivers
- Be sure to enable interrupts when leaving the registered handler or ISR

Lab 9 – Interrupts

- Add an ISR to an existing application
- Register the ISR and enable the related interrupt source



AVNET

Questions

- What happens when a value outside of the requested range of 0-9 is entered?
  - The value is accepted, multiplied by the brightness clock multiplier and write that value into the PWM hardware block control registers. However, the custom IP block detects the out of range value and triggers an interrupt.
- Are interrupts currently enabled in the example application provided?
  - Hardware interrupts are currently provided from the PWM hardware block which are then routed to the GIC in the hardware platform. However, the software application at this point does not configure the GIC to allow the application execution to be interrupted so the software end of the interrupt is not currently enabled.
- When does the function PWMIsr() get called?
  - The PWMIsr() is the interrupt service routine callback function. It is registered with the interrupt subsystem as the callback interrupt handler through the call to XScuGic_Connect() during our interrupt setup.
- GROUP DISCUSSION QUESTION – We handled the event handling within the ISR, what is another way of doing this?
  - Alternatively, we could have the ISR just set a flag and then wait for the main (background execution thread) function process the flag and write a message to the console.

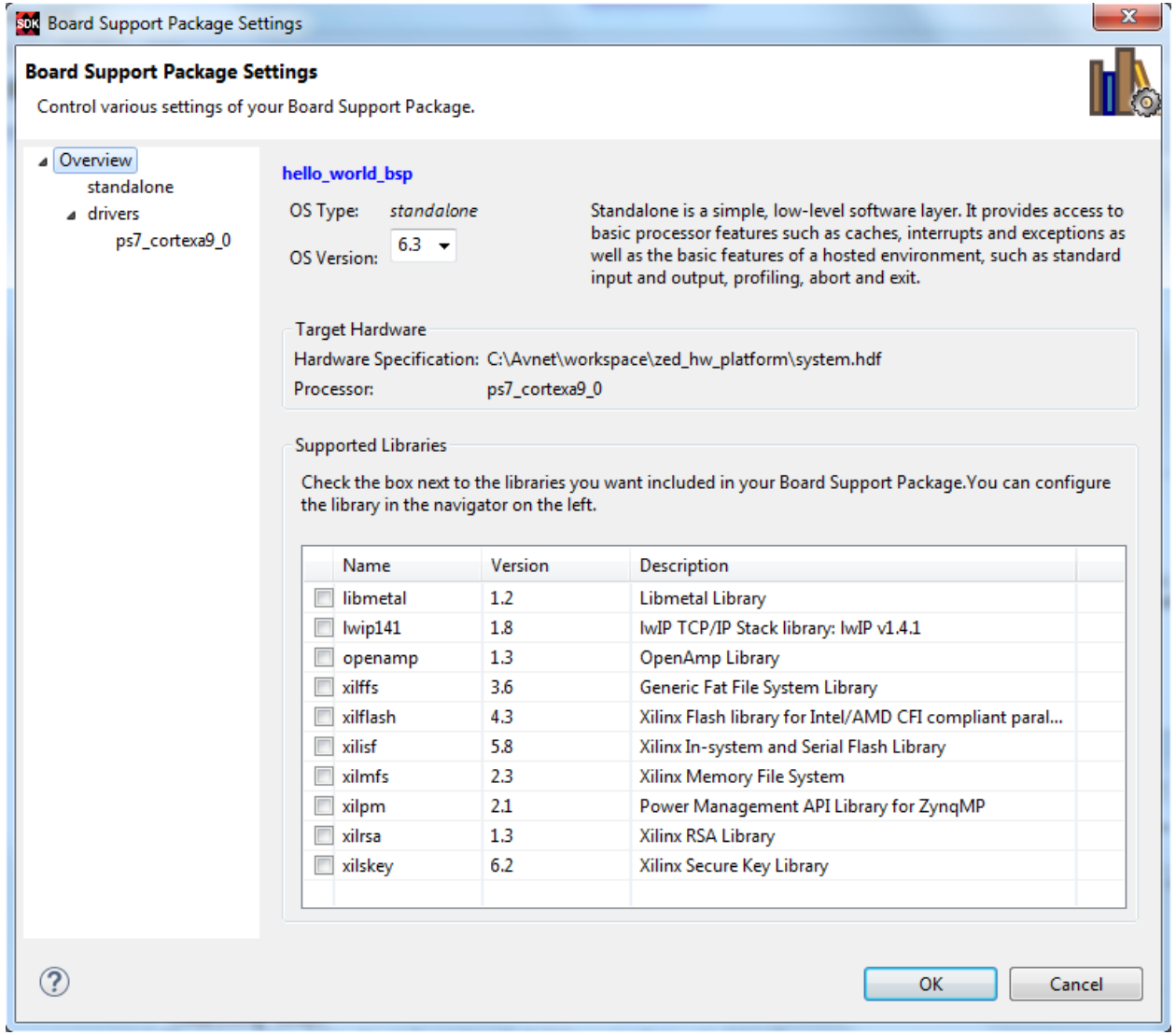
Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

SDK BSP - Supported Libraries

- BSP settings allow for system access library options



System Access Libraries

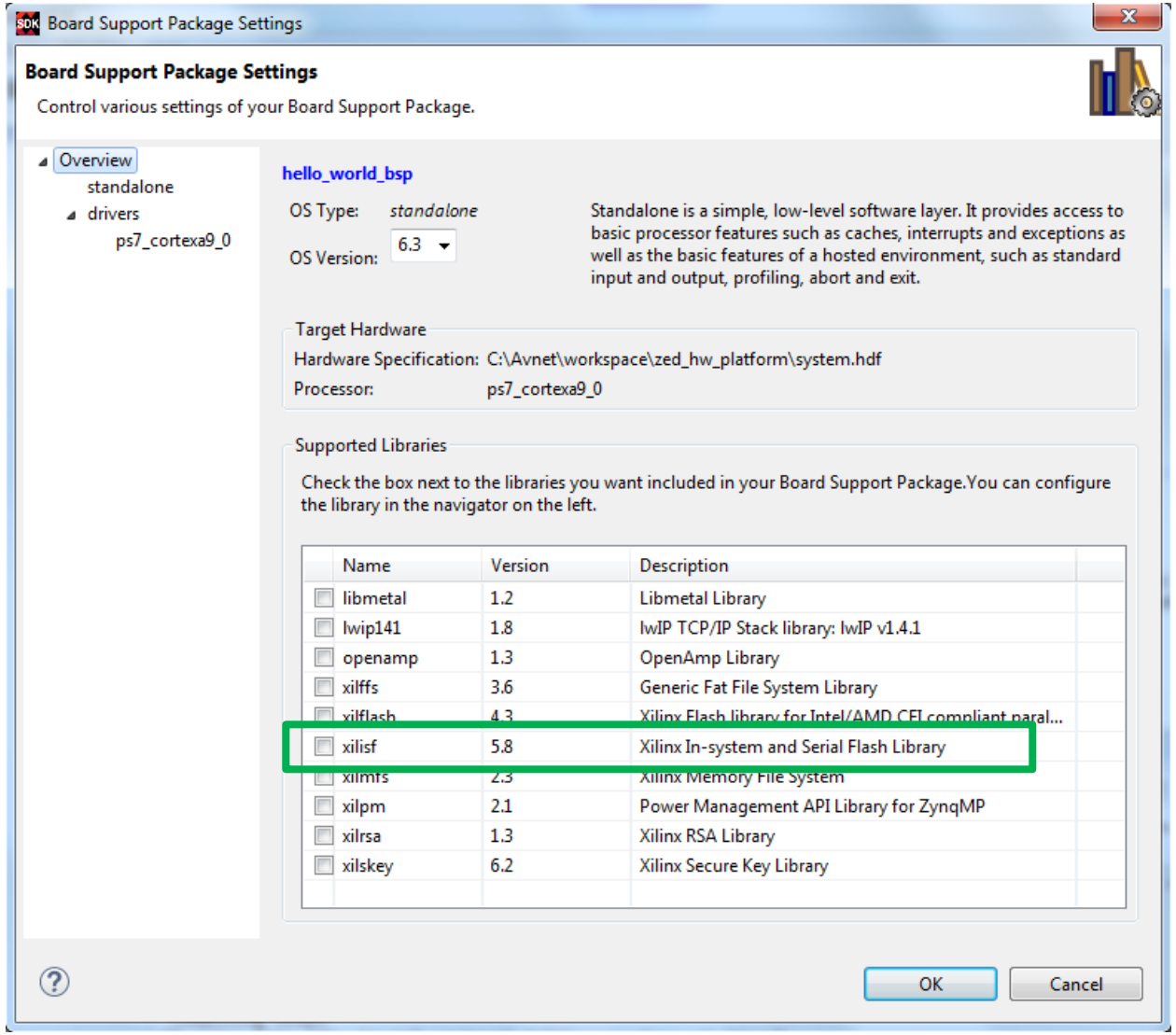
Xilinx provides several important system access libraries

- Generic FAT file system: **xilffs**
- Open source lightweight TCP/IP stack support: **lwip141**
- Access the RSA 2048 and SHA2 algorithms for performing boot authentication: **xilrsa**
- Can be used for secure boot programming of eFUSE and BBRAM: **xilskey**
- In-system serial flash support: **xilisf**
  - Support for Micron QSPI flash device on MiniZed

AVNET

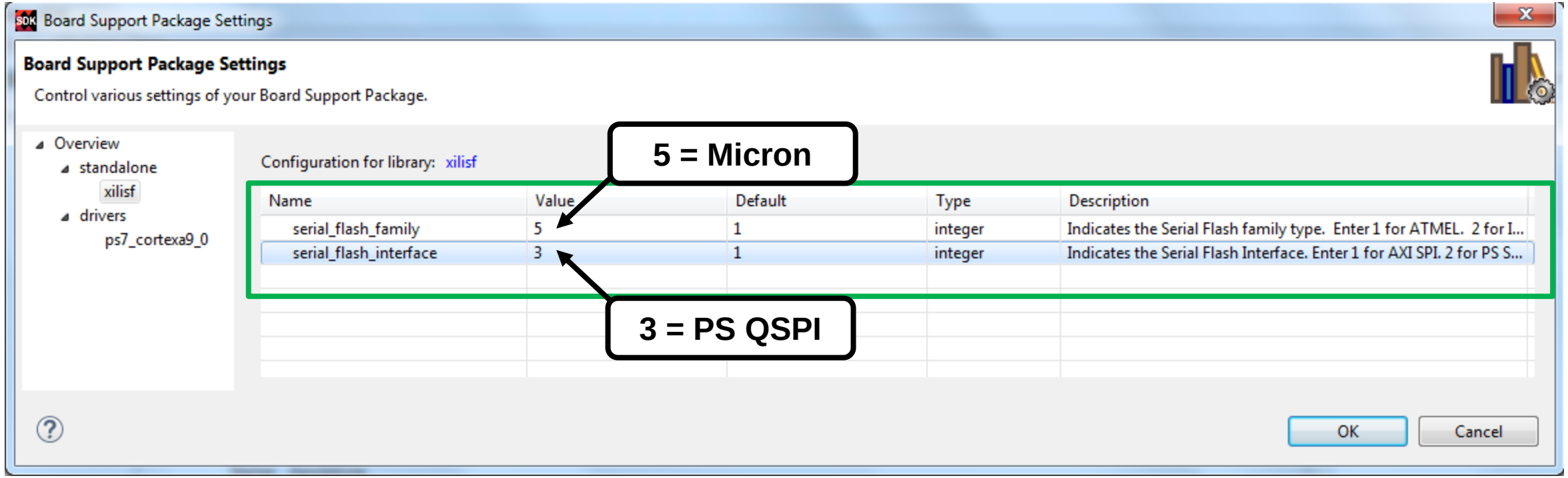
Adding xilisf Library Support to a BSP

- Select xilisf library within BSP settings



Configuring the xilisf Library

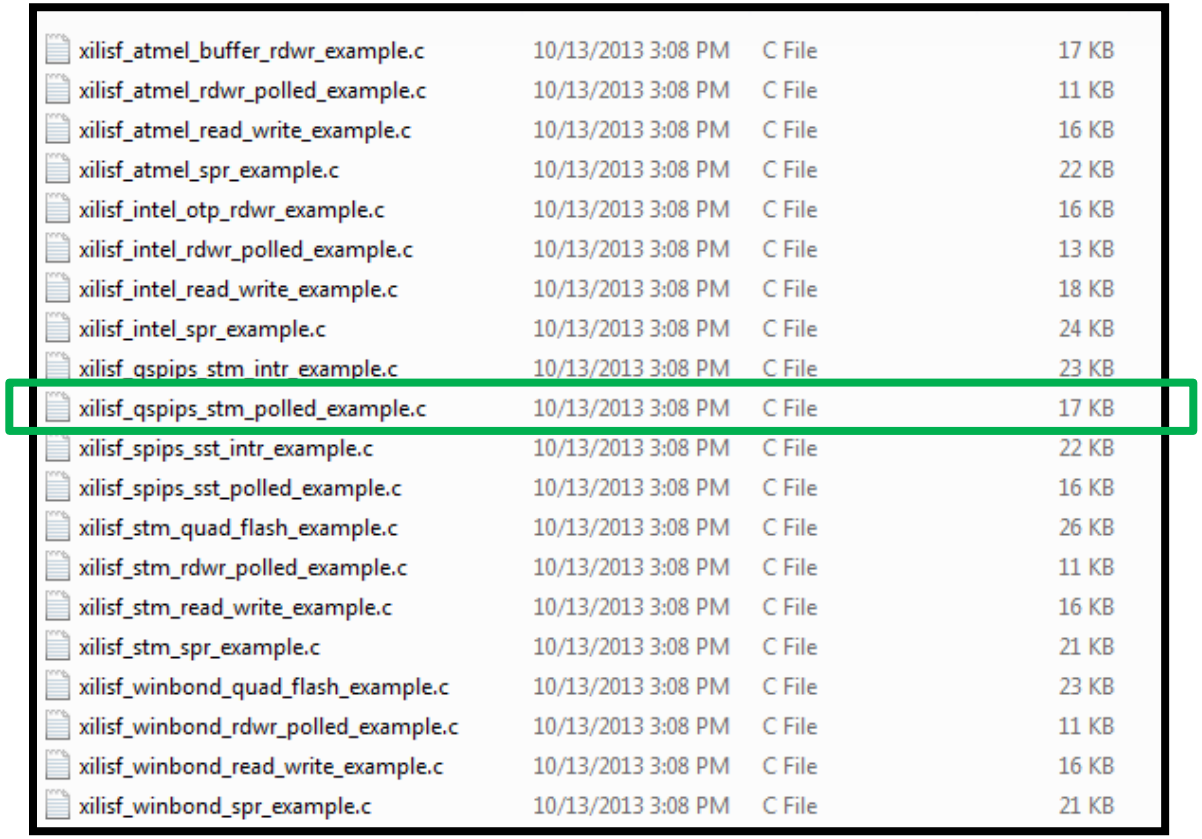
- Initialize xilisf parameter types
  - Flash family type
  - Flash interface type



Leverage Example Code for Libraries

- Example code found within the Xilinx xilisf library folder

```
C:\Xilinx\SDK\2017.4\data\embedded\lib\sw_services\xilisf_v5_9\examples\
```

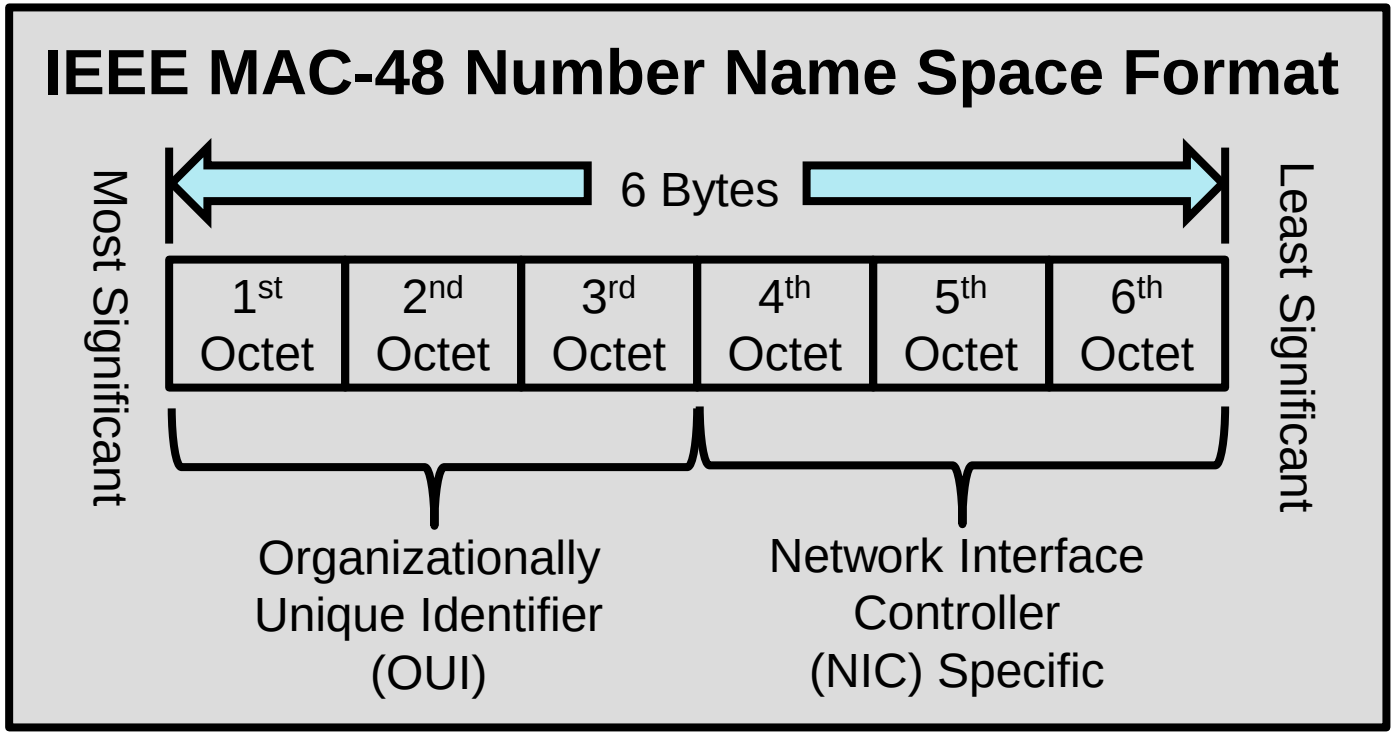


xilisf_atmel_buffer_rdwr_example.c	10/13/2013 3:08 PM	C File	17 KB
xilisf_atmel_rdwr_polled_example.c	10/13/2013 3:08 PM	C File	11 KB
xilisf_atmel_read_write_example.c	10/13/2013 3:08 PM	C File	16 KB
xilisf_atmel_spr_example.c	10/13/2013 3:08 PM	C File	22 KB
xilisf_intel_otp_rdwr_example.c	10/13/2013 3:08 PM	C File	16 KB
xilisf_intel_rdwr_polled_example.c	10/13/2013 3:08 PM	C File	13 KB
xilisf_intel_read_write_example.c	10/13/2013 3:08 PM	C File	18 KB
xilisf_intel_spr_example.c	10/13/2013 3:08 PM	C File	24 KB
xilisf_qspips_stm_intr_example.c	10/13/2013 3:08 PM	C File	23 KB
xilisf_qspips_stm_polled_example.c	10/13/2013 3:08 PM	C File	17 KB
xilisf_spips_sst_intr_example.c	10/13/2013 3:08 PM	C File	22 KB
xilisf_spips_sst_polled_example.c	10/13/2013 3:08 PM	C File	16 KB
xilisf_stm_quad_flash_example.c	10/13/2013 3:08 PM	C File	26 KB
xilisf_stm_rdwr_polled_example.c	10/13/2013 3:08 PM	C File	11 KB
xilisf_stm_read_write_example.c	10/13/2013 3:08 PM	C File	16 KB
xilisf_stm_spr_example.c	10/13/2013 3:08 PM	C File	21 KB
xilisf_winbond_quad_flash_example.c	10/13/2013 3:08 PM	C File	23 KB
xilisf_winbond_rdwr_polled_example.c	10/13/2013 3:08 PM	C File	11 KB
xilisf_winbond_read_write_example.c	10/13/2013 3:08 PM	C File	16 KB
xilisf_winbond_spr_example.c	10/13/2013 3:08 PM	C File	21 KB

AVNET

Lab 10 – Serial Flash Library

- Use xilisf library to access flash memory to store a system configuration parameter
  - Erase a page of flash memory
  - Write data to a piece of memory
  - Read data back from that piece of memory



AVNET

Questions

- What if you wanted to reuse part of this application on your own Zynq design but you prefer to use Microchip Flash instead of Micron Flash?
  - Avnet also distributes Microchip (Atmel) devices and the only part of this software application that would need to be changed is the serial_flash_family field from Experiment 1, Step 10.
- Why is a separate BSP created for this application? Why not reuse the standalone_bsp_0 from earlier lab activities?
  - A separate BSP was created since the Xilinx xilisf library was to be added specifically to support this MAC programming application. We could have instead modified the standalone_bsp_0 to add the xilisf library but that would cause the library code to built into the BSP each time the project is cleaned and rebuilt. This would unnecessarily increase build time for the full workspace which might not be desirable. However, if xilisf were built into the BSP but not called by our other applications, it would NOT affect their executable code size.
- How were functions created within the application code which accesses the flash? Were these functions written entirely from scratch?
  - Not really, the flash access functions found within the xilisf_qspips_flash_polled.c source file are adapted from the example code found within the Xilinx xilisf library folder

    C:\Xilinx\SDK\2017.4\sw\lib\sw_services\xilisf_v5_9\examples

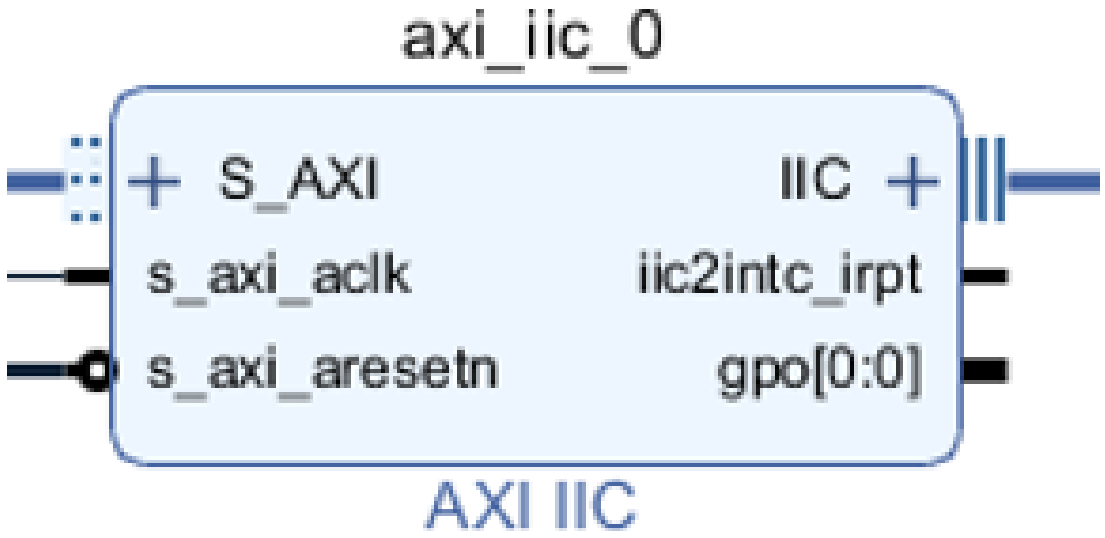
AVNET

Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

AVNET

AXI IIC IP Block



Xilinx Standalone iic_v3_4 Driver Documentation

XILINX
ALL PROGRAMMABLE™

iic_v3_4

Xilinx SDK Drivers API Documentation

Overview | Data Structures | APIs | File List

iic_v3_4 Documentation

Xlic is the driver for an IIC master or slave device.In order to reduce the memory requirements of the driver the c
multimaster features of the driver, the user must call functions (Xlic_SlaveInclude and Xlic_MultiMasterInclude) t

Two sets of higher level API's are available in the **Xlic** driver that can be used for Transmission/Reception in Ma

- **Xlic_MasterSend()/ Xlic_MasterRecv()** which is used in normal mode.
- **Xlic_DynMasterSend()/ Xlic_DynMasterRecv()** which is used in Dynamic mode.

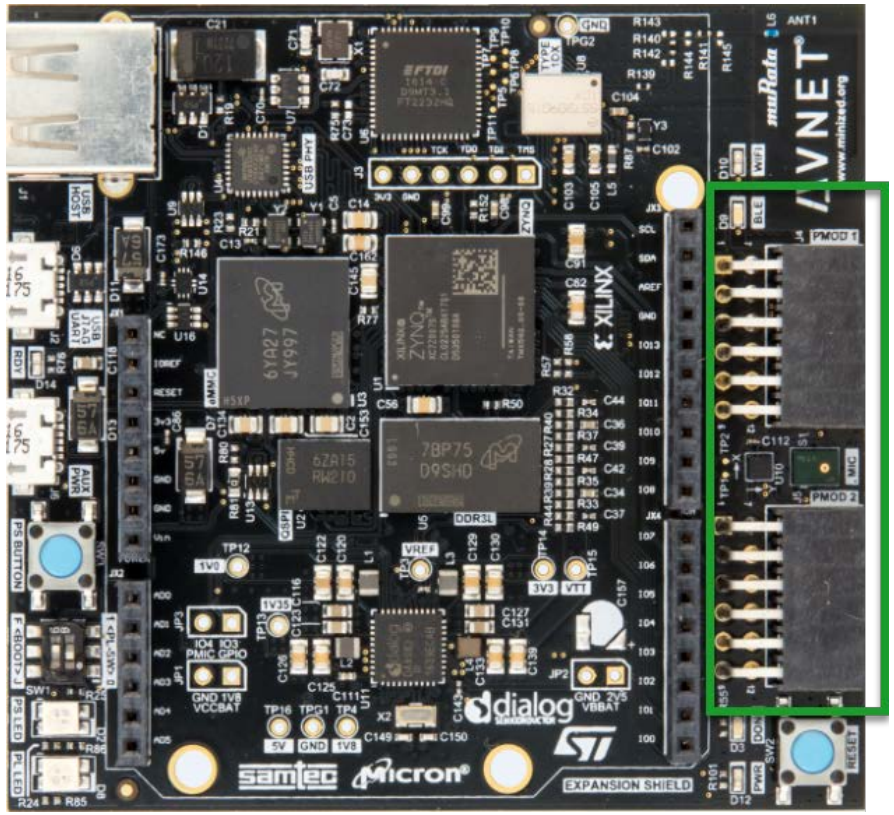
Similarly two sets of lower level API's are available in **Xlic** driver that can be used for Transmission/Reception in

- **Xlic_Send()/ Xlic_Recv()** which is used in normal mode

AVNET

Expansion Through Pmod Interface

- Two 2x6 Pmods provide expansion using 16 PL I/Os



HTU21D

HTU21D Peripheral Module	Pmod PCB	HTU21D	Humidity Temperature	1.5 to 3.6V	I²C	

AVNET

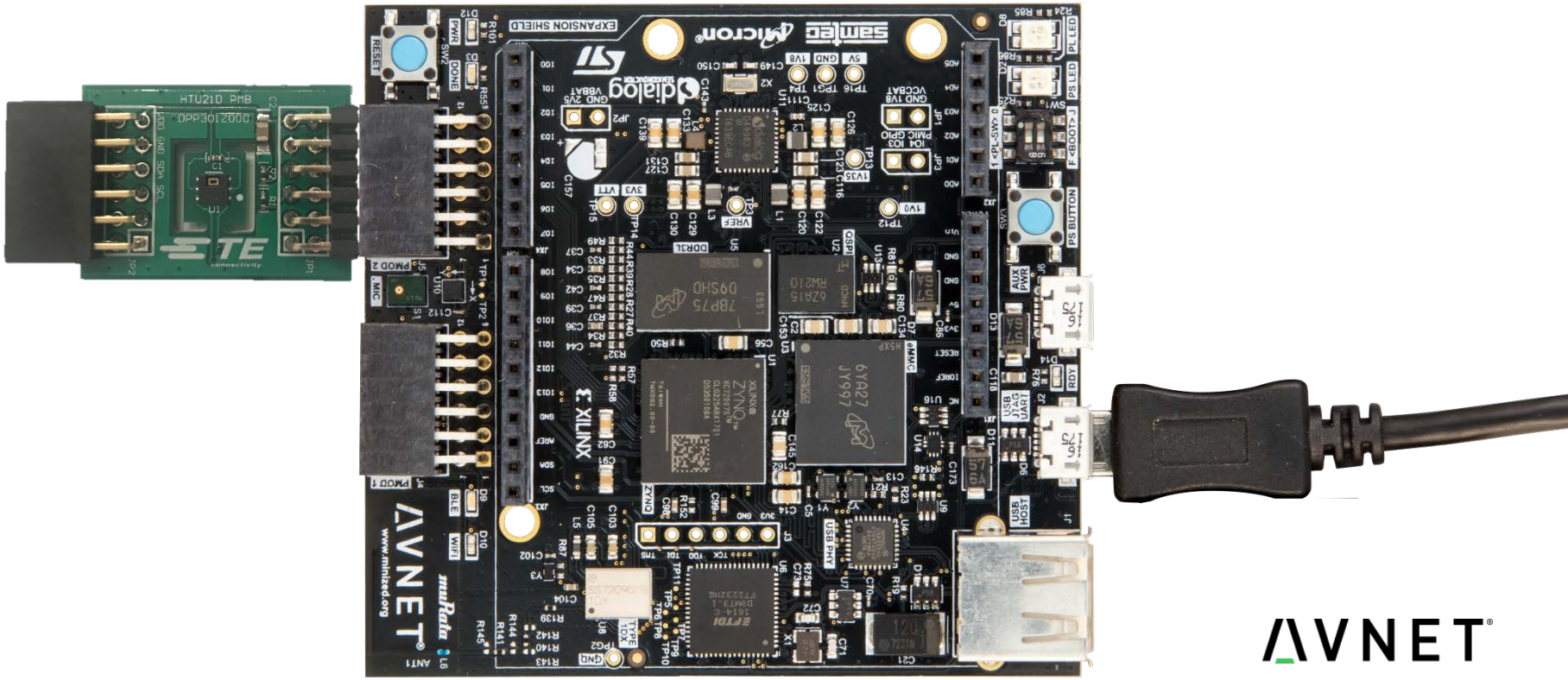
Available Pmods

Model	Package	Sensor Type	Properties Measured	Power	Interface	Image
TSYS01 Peripheral Module	Pmod PCB	TSYS01	Temperature	2.2 to 3.6V	I²C	
TSYS02 Peripheral Module	Pmod PCB	TSYS02	Temperature	1.5 to 3.6V	I²C	
MS5637 Peripheral Module	Pmod PCB	MS5637	Pressure Temperature	1.5 to 3.6V	I²C	
HTU21D Peripheral Module	Pmod PCB	HTU21D	Humidity Temperature	1.5 to 3.6V	I²C	
KMA36 Peripheral Module	Pmod PCB	KMA36	Angular Position	3.0 to 5.5V	I²C	
MS8607 Peripheral Module	Pmod PCB	MS8607	Pressure Temperature Humidity	1.5 to 3.6V	I²C	

AVNET

Lab 11 – TE Sensor Pmod

- Import example application code into a new project
- Connect TE Connectivity sensor module to MiniZed expansion port labeled Pmod 2
- Capture HTU21D humidity and temperature sensor data to the serial console



AVNET

Questions

- What peripheral does the iic_v3_4 driver support? (Hint : Refer to the MDD file)
  - Looking in the MDD file located at C:\Xilinx\SDK\2017.4\data\embeddedsw\XilinxProcessorIPLib\drivers\iic_v3_4\data it states **OPTION supported_peripherals = (axi_iic);** indicating it supports the AXI IIC IP Block

Agenda

Lecture	Description	Labs
Chapter 1	Zynq System Architecture Basics	Lab 1
Chapter 2	Xilinx SDK Overview	Lab 2
Chapter 3	Standalone Board Support Package	Lab 3
Chapter 4	Developing Applications	Lab 4
Chapter 5	Connecting Hardware and Debugging	Lab 5
Chapter 6	First Stage Boot Loader	Lab 6
Chapter 7	Flash Programming and Boot-up	Lab 7
Chapter 8	SDK Project Management	Lab 8
Chapter 9	Interrupts	Lab 9
Chapter 10	Xilinx Libraries	Lab 10
Chapter 11	TE Sensor Pmod	Lab 11
Chapter 12	Next Steps	

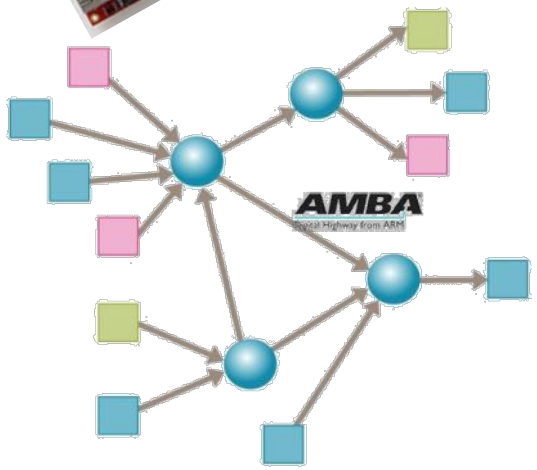
AVNET

Comprehensive Software Ecosystem Support

https://www.xilinx.com/products/design-tools/software-zone/embedded-computing.html#os

AVNET

Largest Portfolio of IP, Design Kits and Ref. Designs

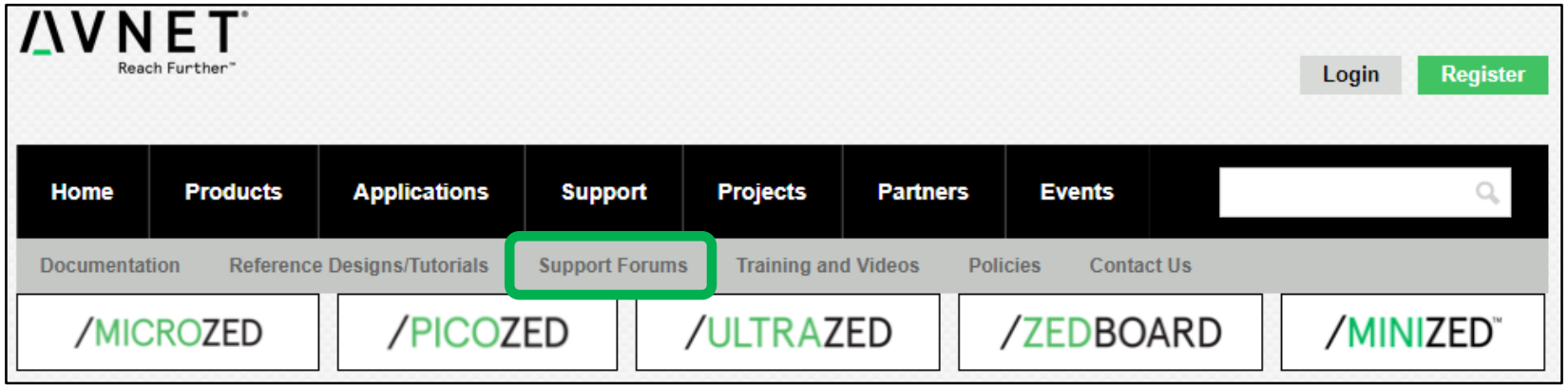


- Widest Selection of Development Platforms and SOMs
  - Over 90 boards and SOM from Xilinx and partners
  - Embedded, intelligent control, video signal processing
  - 200+ FMC daughter cards
- Ready to Use Segment Solutions
  - Boards, IPs, reference designs, app notes, OS and drivers
  - Automotive, A&D, broadcast & pro A/V, consumer electronics, data center, emulation & prototyping, high performance computing, industrial, medical, test & measurement, wired and wireless communications
- Large Selection of AXI-based IPs
  - Xilinx IPs standardized on AXI
  - Large availability of Xilinx and Partner IPs
  - Enhanced portability between FPGAs and SoCs

AVNET

Getting Help and Support

- Visit MiniZed.org Support Forums:
  www.minized.org/forums/zed-english-forum



- For Xilinx technical support, you may contact your local Avnet FAE or Xilinx Support site at
  https://www.xilinx.com/support.html

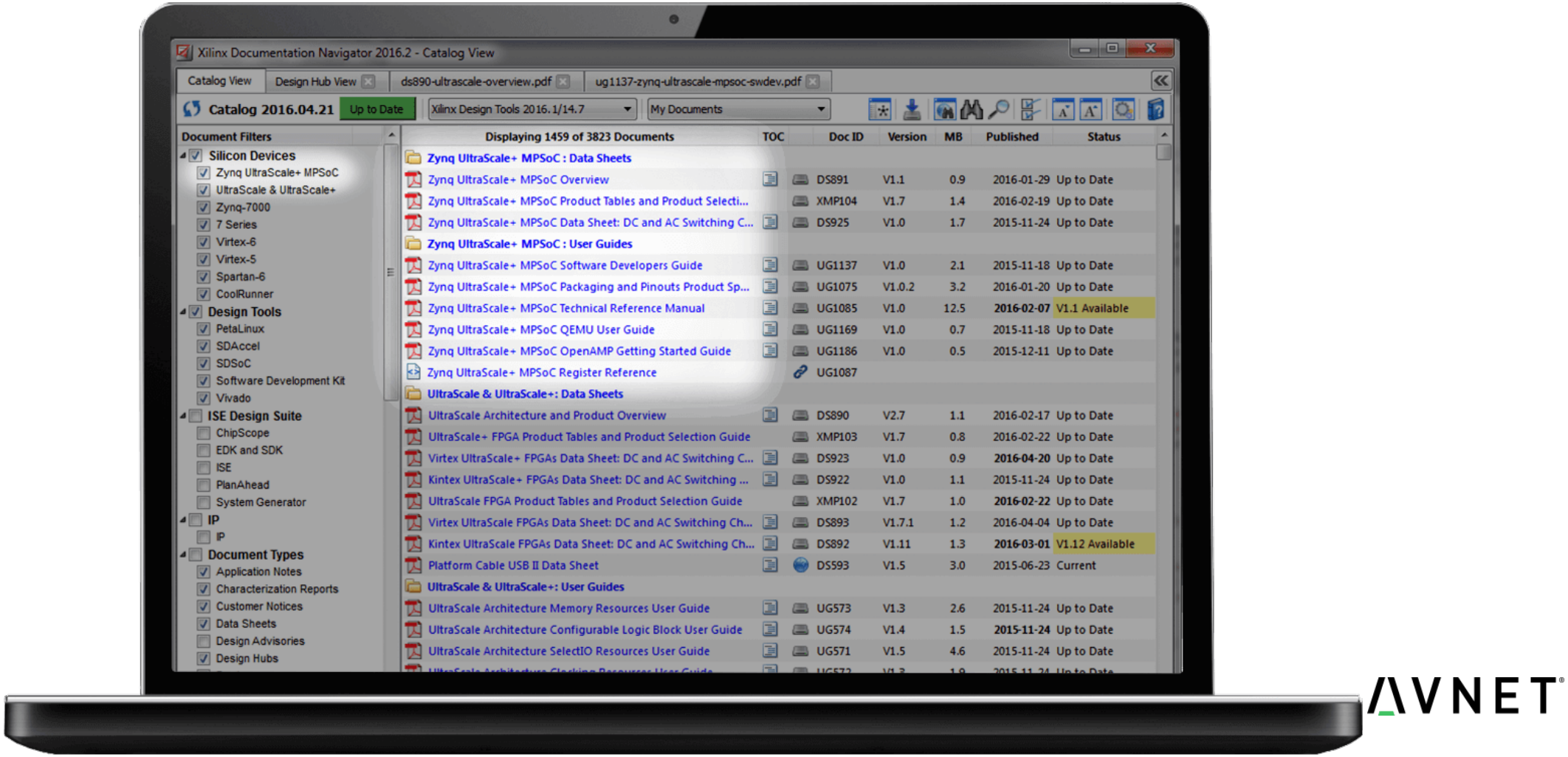
- On Xilinx site you will also find the following resources for assistance:
  - Documentation for Devices, Development Tools, Intellectual Property, Boards, Kits and Modules
  - Searchable Answer Database with Over 4,000 Solutions
  - User Forums

- Contact Avnet Support for any questions regarding MiniZed reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design

AVNET

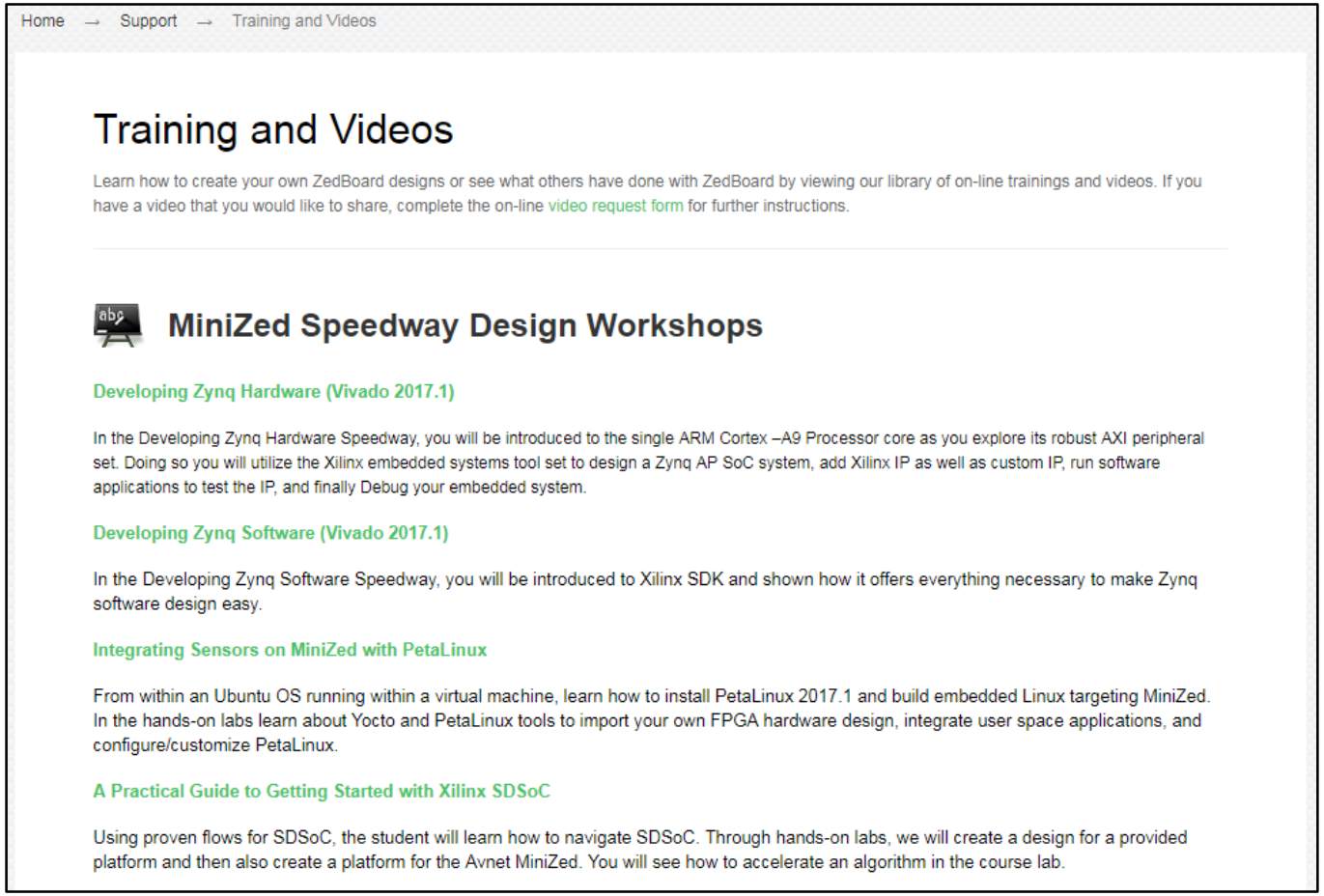
Built-In Documentation - Xilinx DocNav

https://www.xilinx.com/support/documentation-navigation/overview.html



Additional Training

- MiniZed.org Training and Videos Page
minized.org/support/trainings-and-videos



- Xilinx Training – ATP offered courses, self-paced tutorials, and online training videos
https://www.xilinx.com/training.html

AVNET

Objectives

- Introduce developers to Xilinx SDK
- Explore how SDK makes your job easier
- Connect SDK to hardware for execution and debug
- Utilize a peripheral interrupt to show real-time software response
- Show a basic example of how to use an external sensor module