Developing Zynq AP SoC Hardware

With Xilinx Vivado 2013.3

Accelerating Your Success™

Course Objectives

- **Understand the Zynq-7000 All Programmable SoC development flow with Vivado's IP Integrator**
- **Introduce the Extensible Dual ARM Cortex™-A9 Processors Cores**
  - Explore Robust AXI Peripheral Set
- **Utilize the Xilinx embedded systems tools to**
  - Design a Zynq AP SoC System
  - Add Xilinx IP as well as custom IP
  - Run Software Applications to test IP
  - Debug an Embedded System

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

Agenda

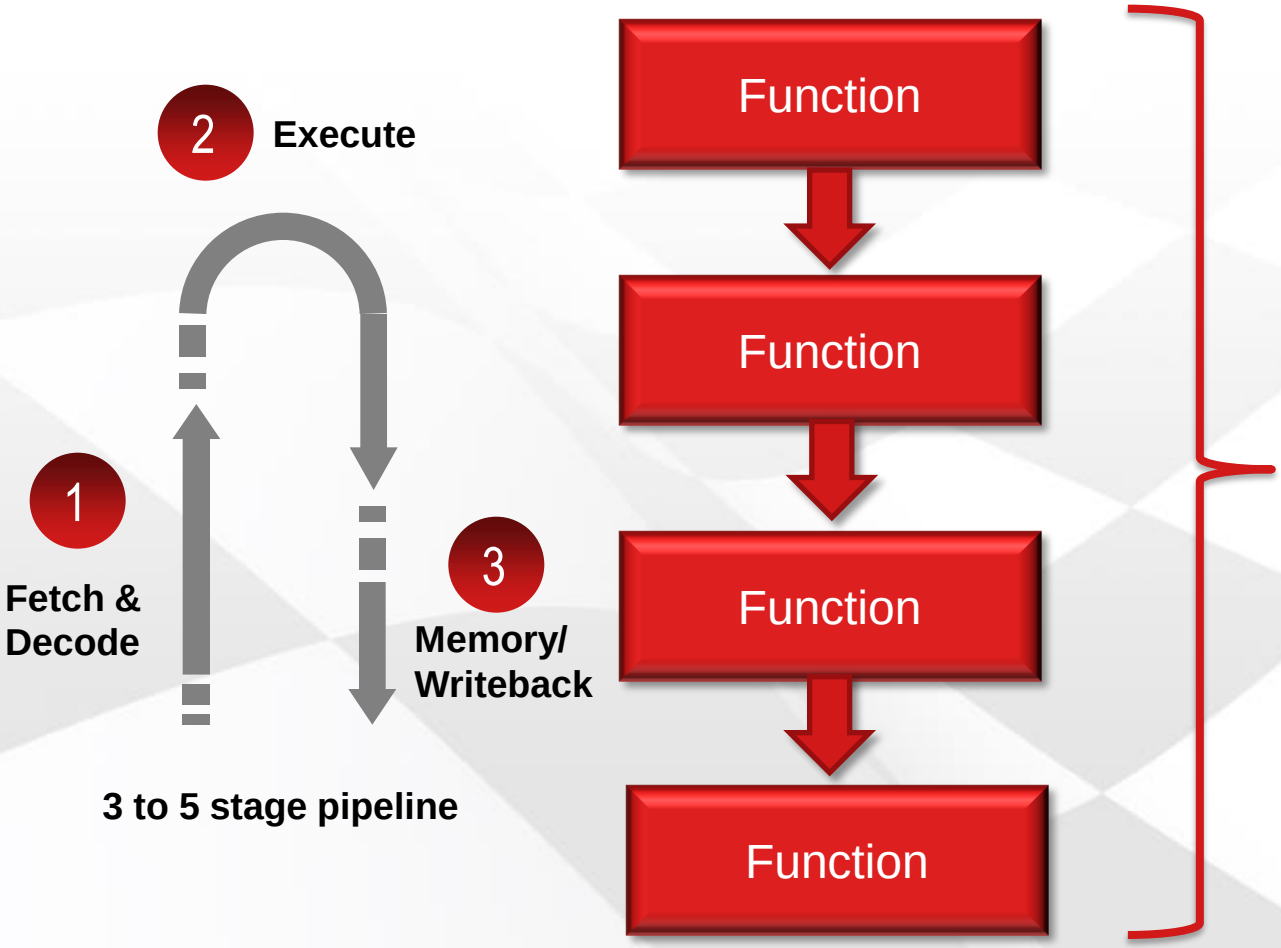
Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

System Bottlenecked?

- **Profiling indicates processor activity**
  - Is processor utilization exceeding 90%?
  - Processor Queue Length > 2?
  - If multiprocessor system, processor time > 50%?
  - Intensive reoccurring tasks?

- **Effects of overburdened Processors include:**
  - Increased Data Latency
  - Delayed Interrupt Handling
  - Lowered Data Throughput

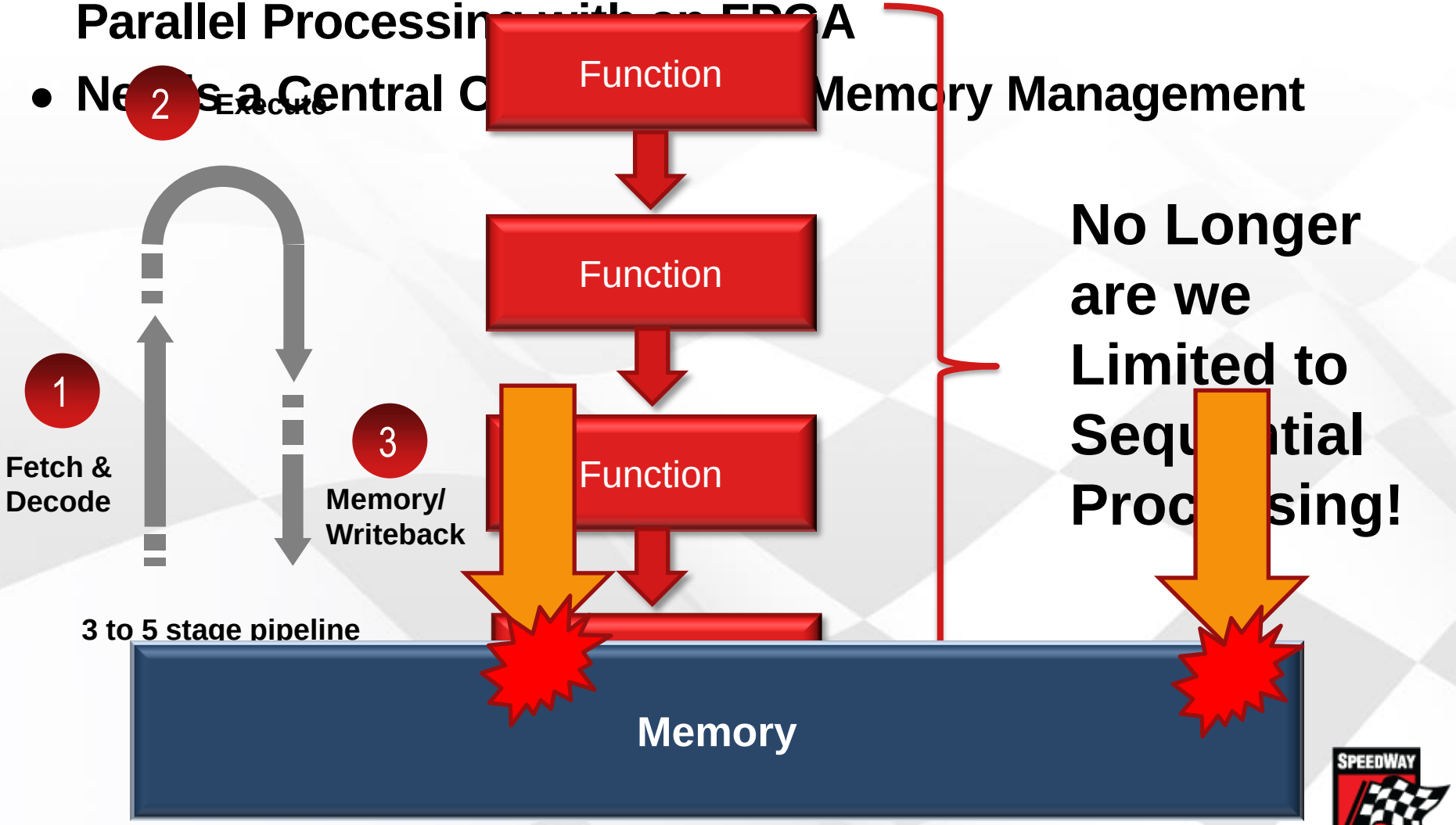
Software Engineers: Stuck in a Sequential World



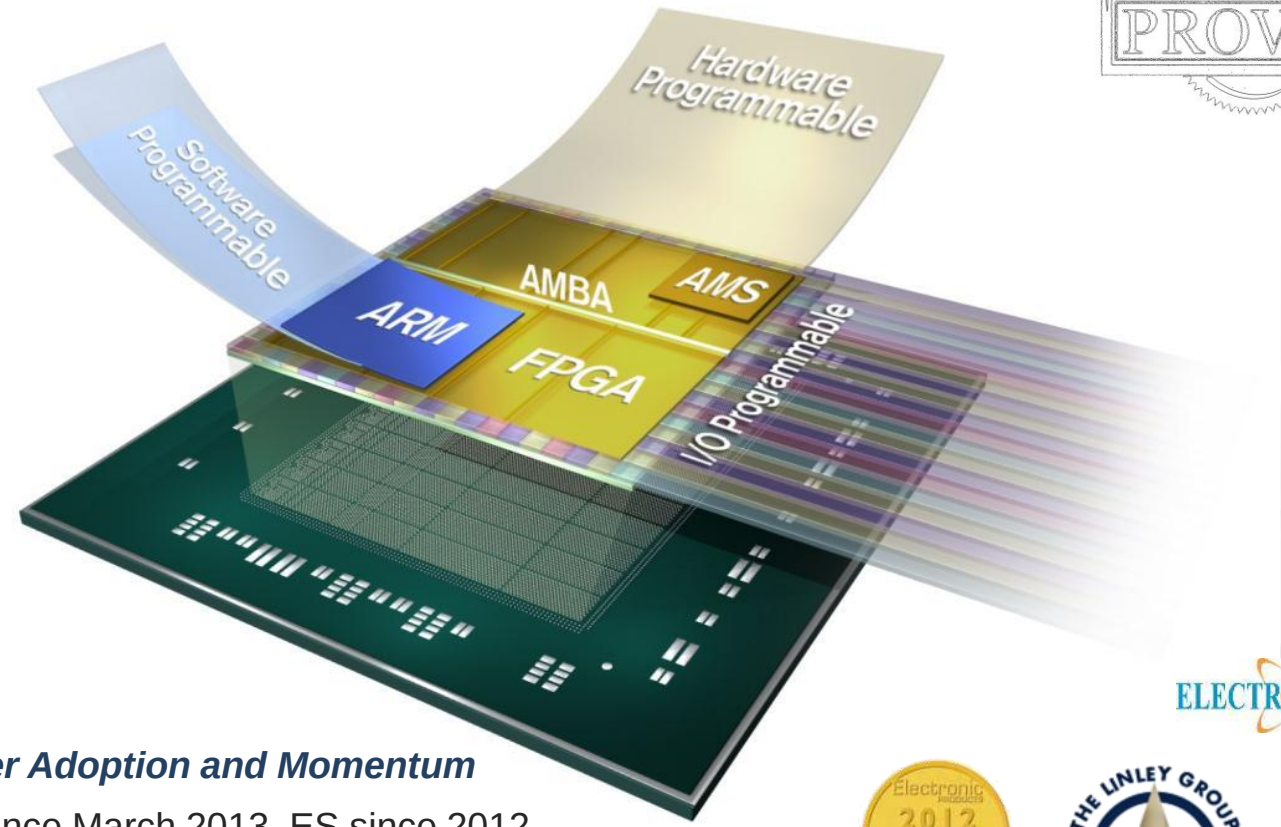
Sequential Processing limits performance

Parallel Processing is Efficient…

- **For critical systems and performance, Engineers utilize Parallel Processing with an FPGA**
- **Needs a Central C… Memory Management**



The First All Programmable SoC



ZYNQ

Significant Customer Adoption and Momentum

- ✓ In full production since March 2013, ES since 2012
- ✓ 500+ unique customers actively designing
- ✓ 100+ Zynq specific partners
- ✓ All major OSs supported and in use
- ✓ 20+ different development boards and SOMs

PROVEN

ELECTRON D'OR

Electronic Products 2012 PRODUCT of the YEAR AWARD

THE LINLEY GROUP 2012 ANALYSTS' CHOICE

ACE AWARDS EETimes EDN

EDN CHINA INNOVATION AWARD 创新奖 2011 优秀产品奖

BEST OF SHOW AWARD ESC 2010 SAN JOSE

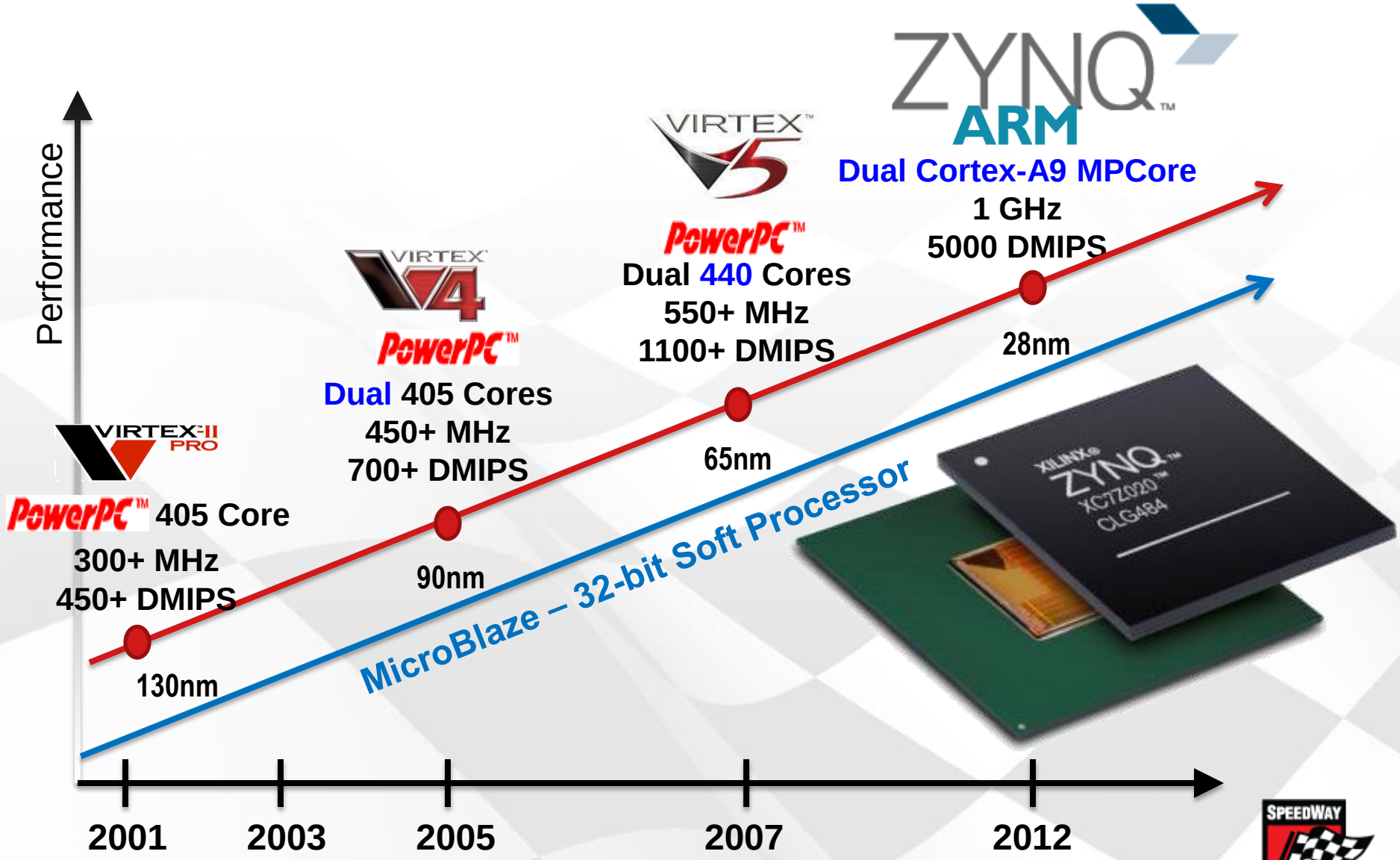
Elektra 2011 European Electronics Industry Awards WINNER

IET AWARDS The Institution of Engineering and Technology

SPEEDWAY DESIGN WORKSHOPS

AVNET electronics marketing

Xilinx Processing Heritage



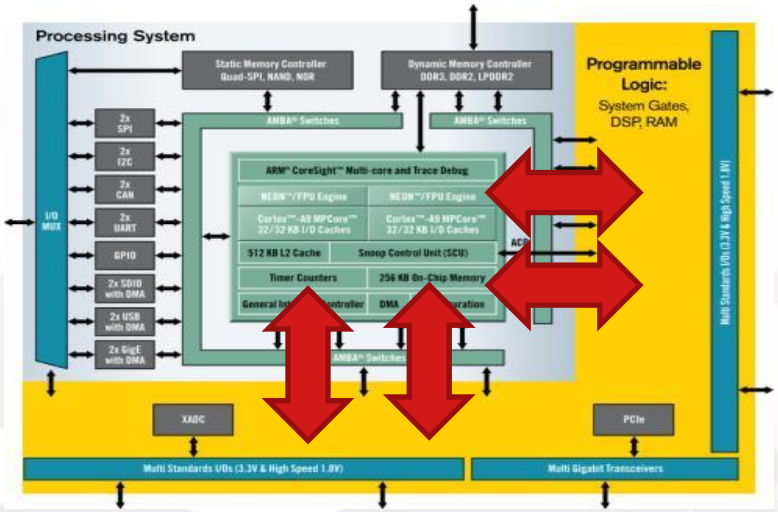
AVNET electronics marketing

10+ years, 4 Generations

SPEEDWAY DESIGN WORKSHOPS

Increased System Performance

- **Increasing SW Processing Performance with:**
  - Programmable Logic
  - Massive DSP processing
  - High throughput AXI
    - Over 3000 PS to PL direct connections
  - High performance I/Os
  - Gigabit transceivers



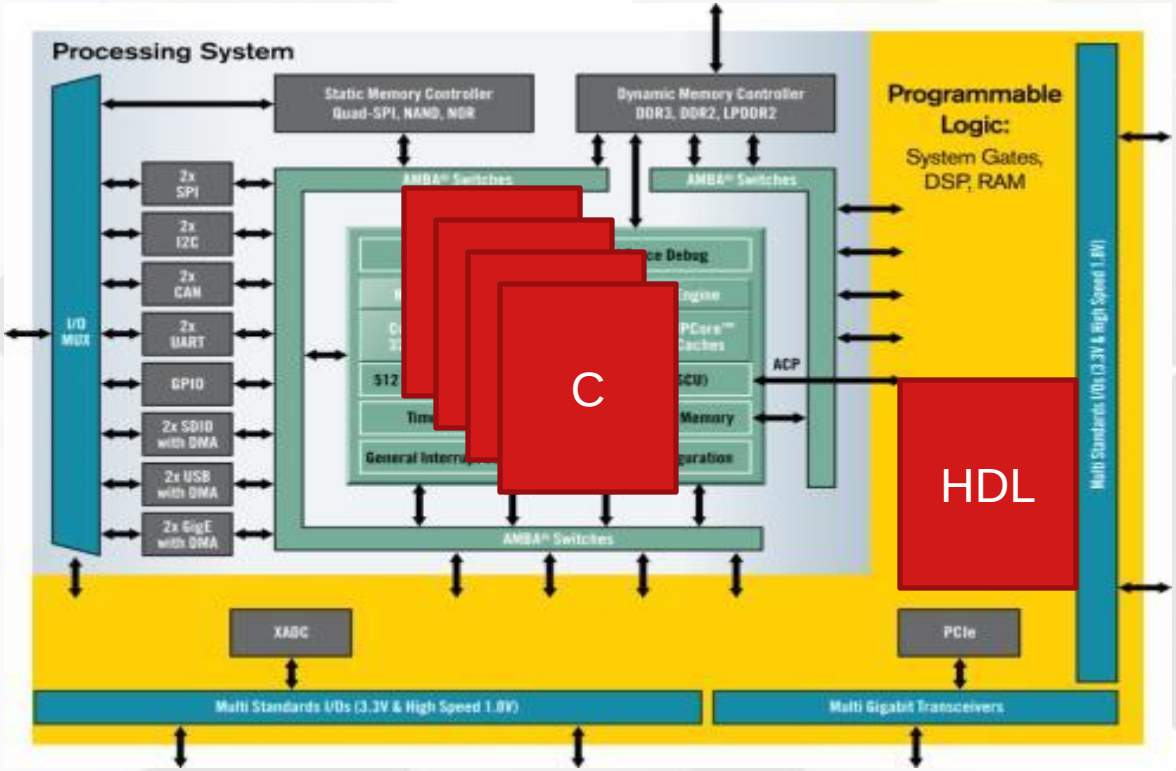
Elements	Performance (up to)
Processors (each)	1 GHz
PL Fabric/ DSP Fmax	741 MHz
DSP (aggregate)	1080 GMACs
Transceivers (each)	12.5Gbps

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Increased System Performance

- **Optimized & Simplified HW/SW Partitioning**
  - HW acceleration enables scaling SW performance to address many applications
  - Low latency interfacing for efficient co-processor implementation and high throughput data transfers

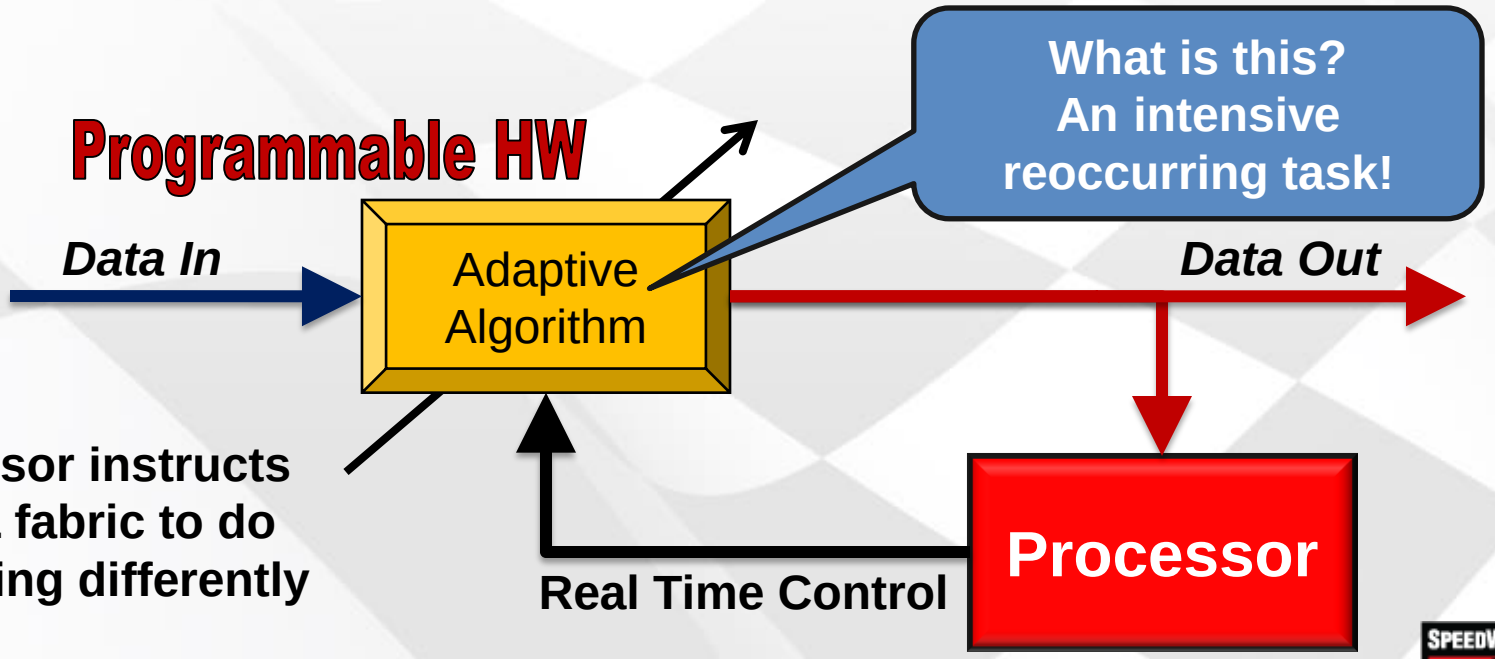


AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

More Than Just Parallel Processing

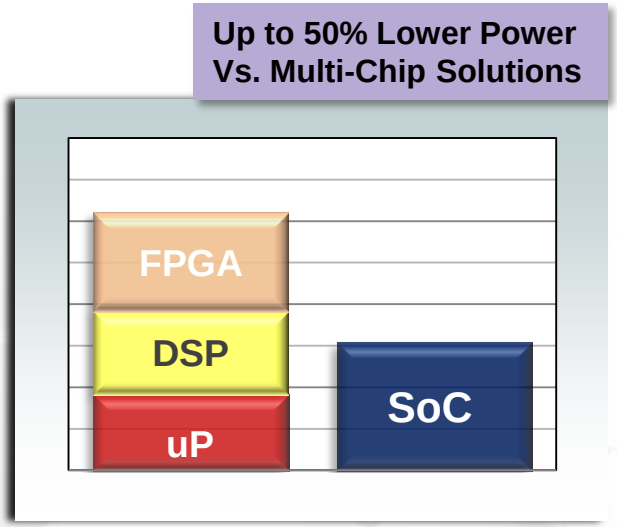
- **HW Runs Heavy Data Path Algorithms**
  - Take advantage of HW parallelism
  - Maintain high bandwidth throughput (High Frequency)
- **SW Controls and Updates Elements of Algorithms**
  - Easier to process complex input criteria
  - Typical lower data rate then actual data path (Off-load HW)



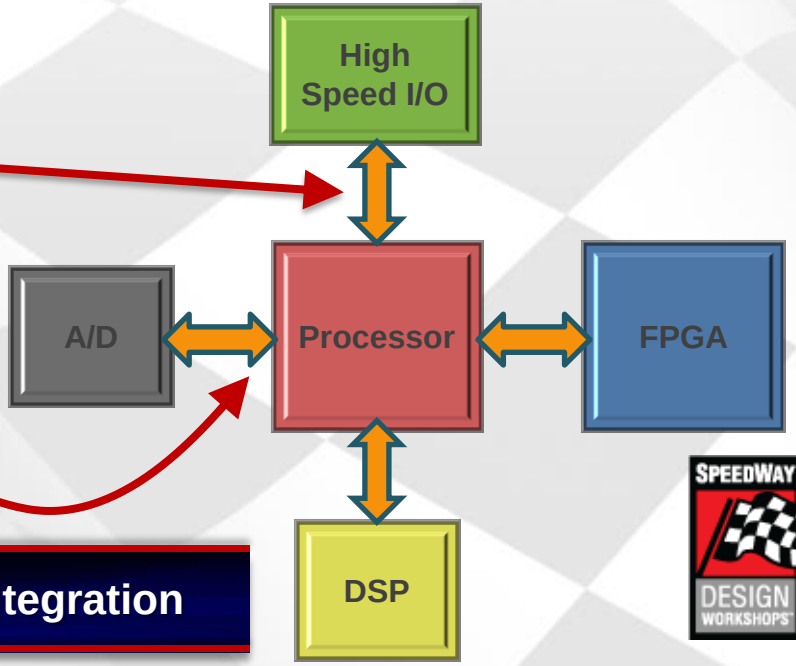
SDR, Video, Encryption, Data Processing

Total Power Reduction

- **Integration = Power Reduction**
  - Static Power
  - I/O Power
- **Flexible/Tunable Power Envelope**
  - Adjusted frequencies
  - Clock gating
  - ARM low Power States
- **EMI Reduced**



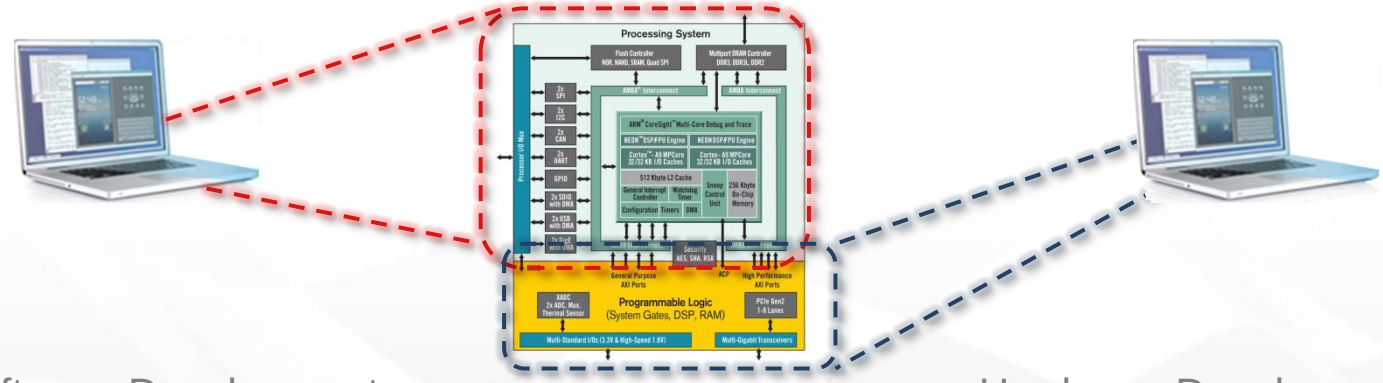
Significant Power Dissipation in I/Os



Power reduction at the system level through integration

SPEEDWAY DESIGN WORKSHOPS

Parallel Developments of your AP SoC Based Application



Software Development

- **Processor boots first like any ARM based SoC.**
- **SW developers can use their favorite SW tool to load / debug SW code over JTAG**
- **Programmable Logic can be left unconfigured while developing on real hardware**

Hardware Development

- **Reference SW boots processor first leaving PL up and ready to be programmed through JTAG**
- **Vivado Probe connects to Programmable Logic like to any other FPGA**
- **FPGA developer can start loading / debug like for any FPGA**

SPEEDWAY

SW developments like any other ARM based SoC

HW developments like any other Xilinx FPGA

Highest Levels of Security - The Xilinx Advantage

- **No snooping of PS to PL data transfers**
- **Fully secured single source boot**
  - 2 boot methods = back door into system
- **Large OCM (256kB) for BootLoader, TrustZone and sensitive SW tasks**
- **Comprehensive security offering including hardware blocks, IP, OS, software, …**

Hardware

Discretix trust • embedded

XILINX

Elliptic

Secure System Boot

ARM

XILINX

Software Execution

sierraware

Discretix trust • embedded

ARM

open virtualization

WIND RIVER

Open Kernel Labs

Green Hills SOFTWARE

SYSGO EMBEDDING INNOVATIONS

XILINX

Levels of security beyond multi-chip solutions

SPEEDWAY DESIGN WORKSHOPS

Extensive OS, Middleware & Stack Ecosystem

Middleware and Stacks

Operating Systems


Open Source Linux & Commercial Linux

Hypervisors

Zynq-7000 Device Portfolio Summary

Scalable platform offers easy migration between devices

	Zynq-7000 AP SoC Devices	Z-7010	Z-7015	Z-7020	Z-7030	Z-7045	Z-7100
Processing System	Processor Core	Dual ARM® Cortex™-A9 MPCore™					
	Processor Extensions	NEON™ & Single / Double Precision Floating Point					
	Max Frequency	866 MHz			Up to 1 GHz		
	Memory	L1 Cache 32KB I / D, L2 Cache 512KB, on-chip Memory 256KB					
	External Memory Support	DDR3, DDR3L, DDR2, LPDDR2,2x QSPI, NAND, NOR					
	Peripherals	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO					
Programmable Logic	Approximate ASIC Gates	~430K (28k LC)	~1.1M (74k LC)	~1.3M (85k LC)	~1.9M (125k LC)	~5.2M (350k LC)	~6.6M (444kLC)
	Block RAM	240KB	380KB	560KB	1,060KB	2,180KB	3,020KB
	Peak DSP Performance (Symmetric FIR)	100 GMACS	200GMACS	276 GMACS	593 GMACS	1334 GMACS	2662 GMACS
	PCI Express® (Root Complex or Endpoint)	-	Gen2 x4	-	Gen2 x4	Gen2 x8	
	Agile Mixed Signal (XADC)	2x 12bit 1Msps A/D Converter					
I/O	Processor System IO	130					
	Multi Standards 3.3V IO	100	150	200	100	212	250
	Multi Standards High Performance 1.8V IO	-	-	-	150	150	150
	Multi Gigabit Transceivers	-	4	-	4	16	16

MicroZed

ZedBoard

AVNET electronics marketing

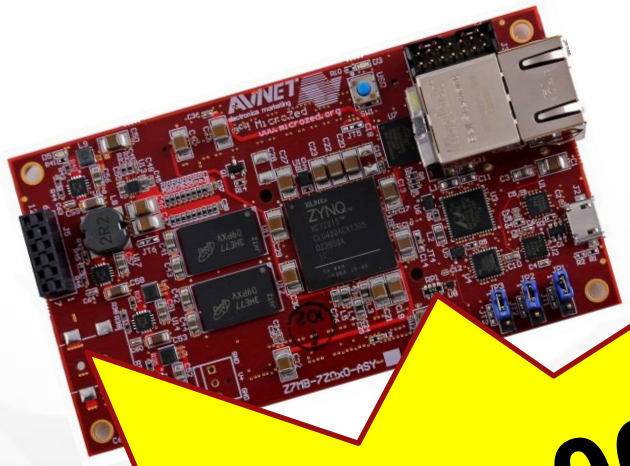
DESIGN WORKSHOPS

Zynq Prototyping and Production Solutions

We've got you covered!

- **ZedBoard**
- **MicroZed**
- **ZC702**
- **ZC706**
- **Zynq MMP**
- **Zynq Mini-ITX**

- **IVK**
- **SDR**



Over 12,000 Boards Sold!

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

Xilinx Embedded Tool Flow

- **Vivado Design Suite WebPACK Edition**
  - **FREE!**
  - Supports **four** Zynq devices: 7010, 7015, 7020 and 7030

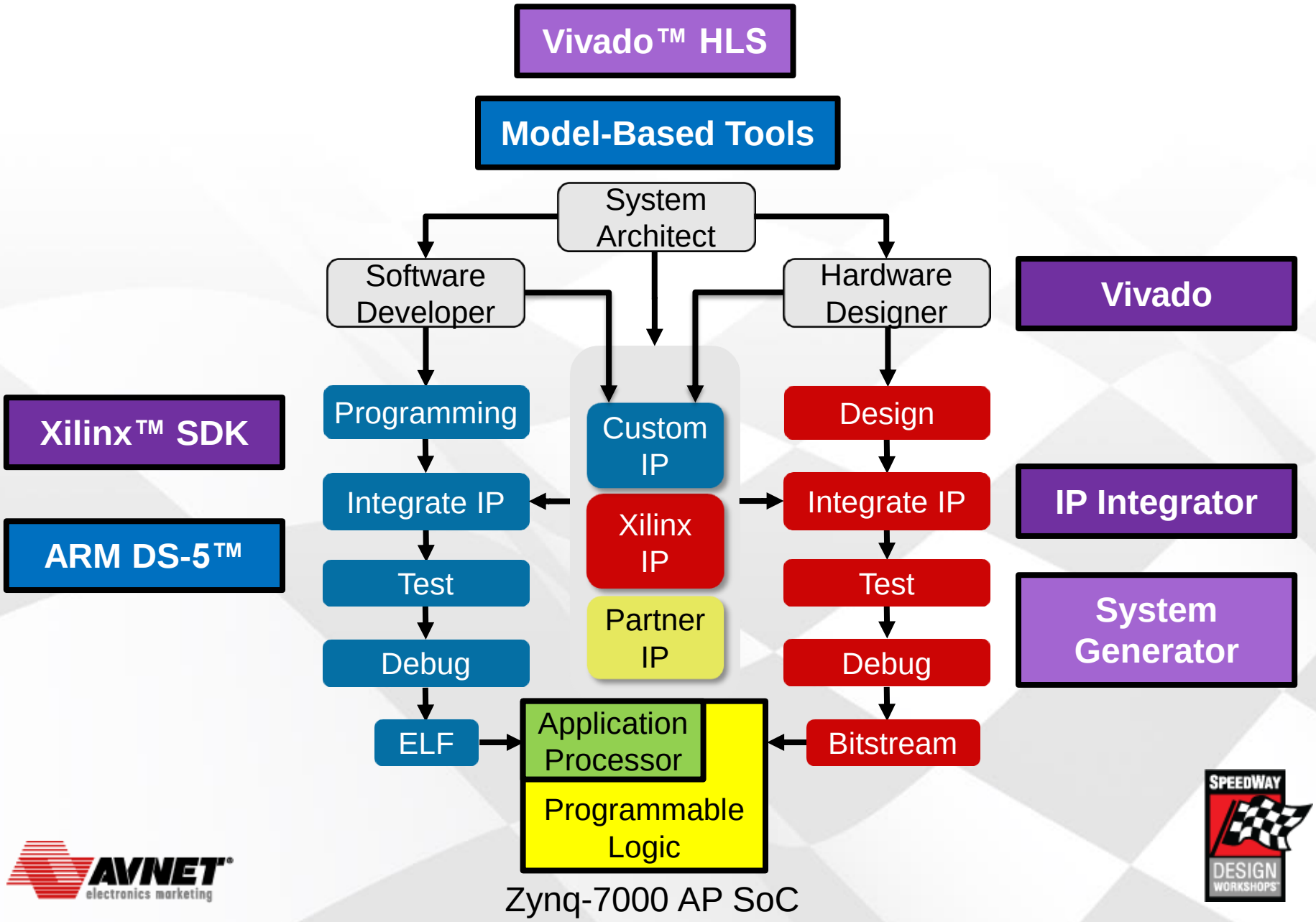
Pillars of Productivity	Features	WebPACK	Design Edition	System Edition	Free 30-day Eval
IP Integration and Implementation	Integrated Design Environment	√	√	√	√
	Software Development Kit (SDK)	√	√	√	√
Verification and Debug	Vivado Simulator	√	√	√	√
	Vivado Logic Analyzer		√	√	√
	Vivado Serial Analyzer		√	√	√
Design Exploration and IP Generation	High-Level Synthesis			√	√
	System Generator for DSP			√	√

- **MicroZed and ZedBoard include license for Vivado Analyzer**

AVNET electronics marketing

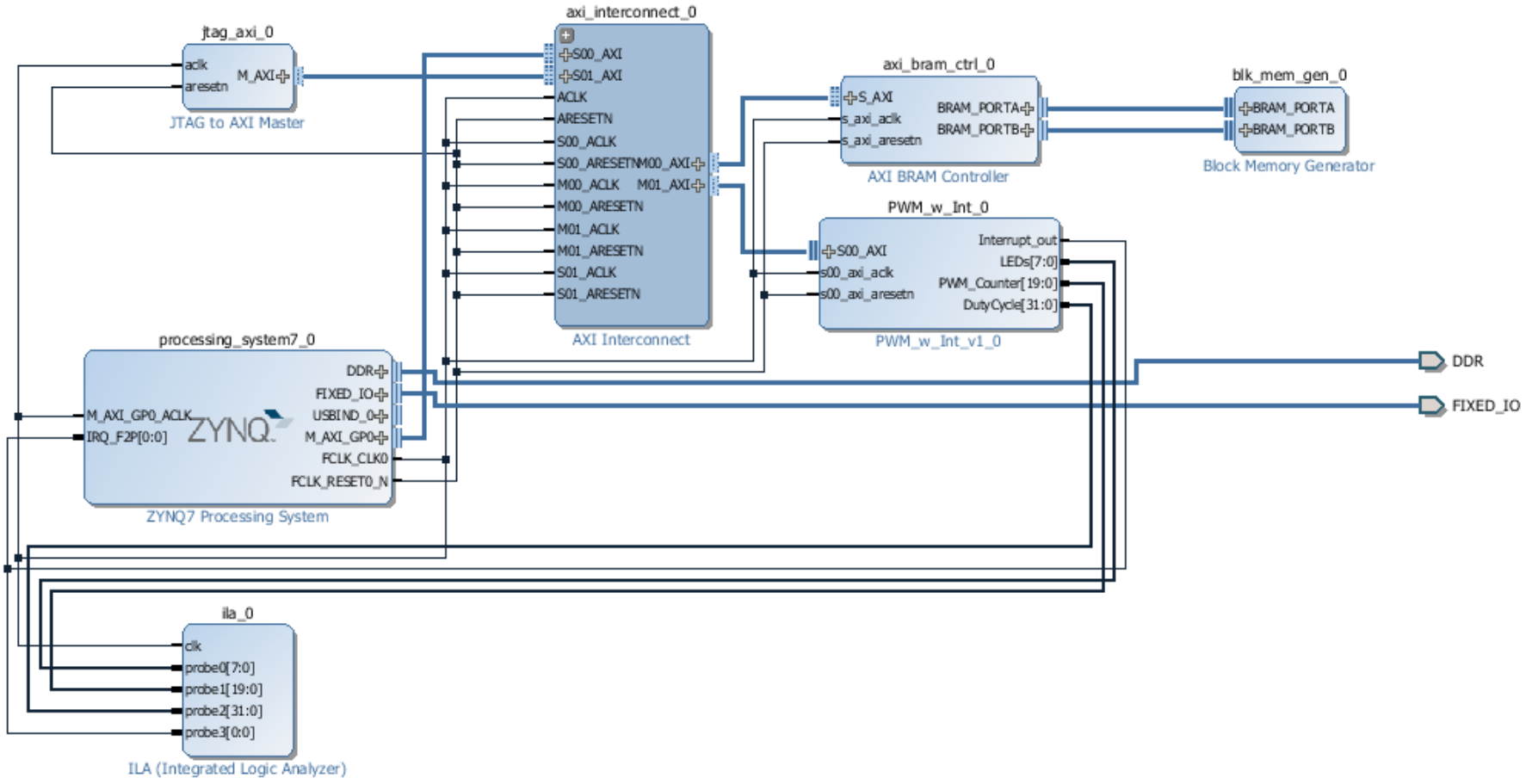
SPEEDWAY DESIGN WORKSHOPS

Zynq-7000 AP SoC Embedded Design Flow



Introducing Vivado IP Integrator (IPI)

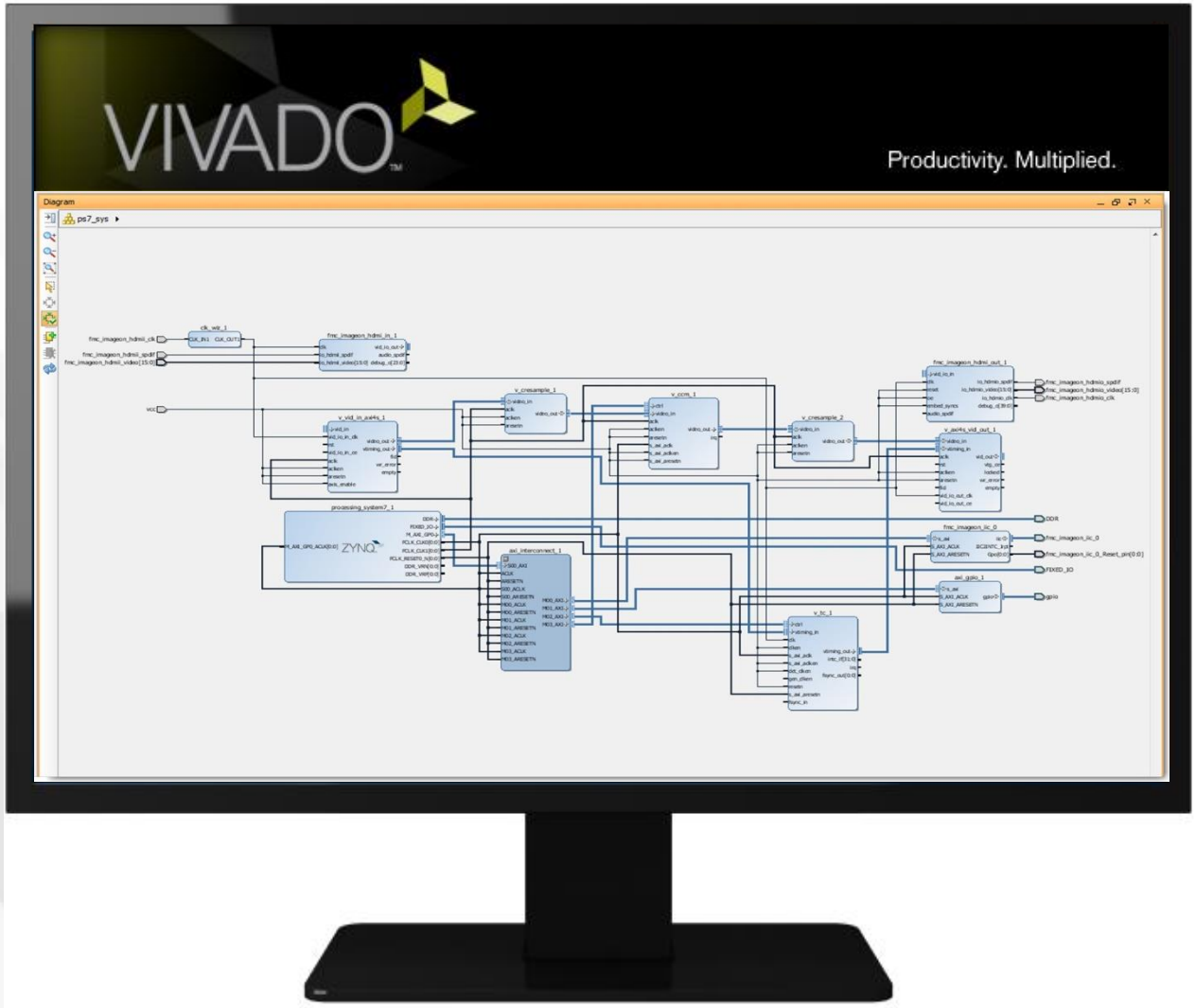
- **You were doing schematics 20 years ago, and we're doing them again!**
  - Why? We thought designs were complex then… But now…



Introducing Vivado IP Integrator

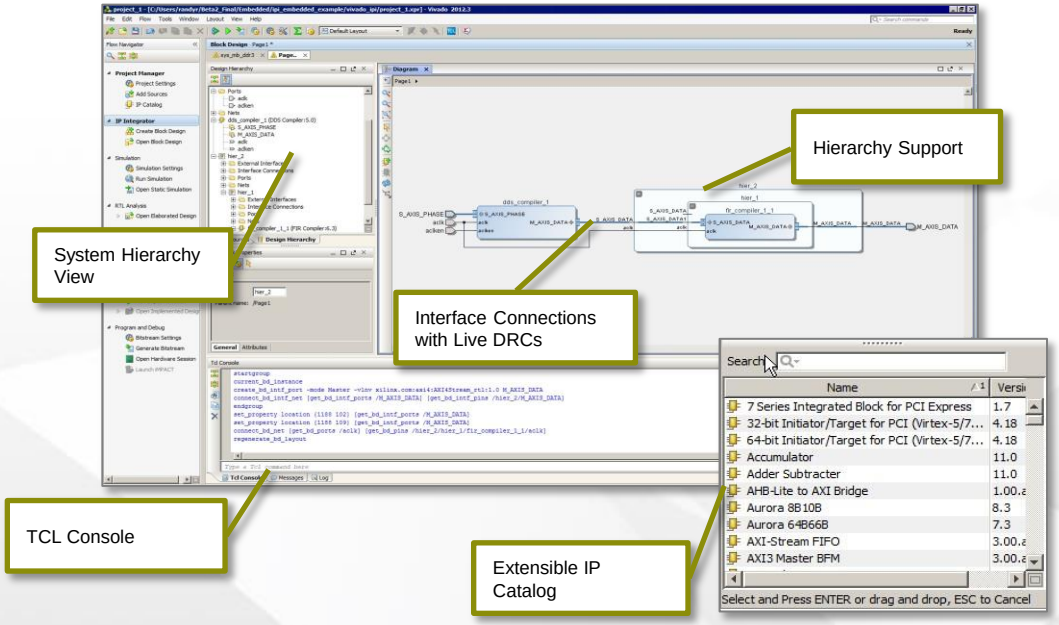
Enabling an IP Centric Design Flow

- **Plug-and-play IP**
- **Vast IP catalog**
- **Accelerates**
  - Integration
  - Productivity
- **Smarter systems**
  - Embedded
  - DSP
  - Video
  - Analog
  - Networking



AVNET electronics marketing

Vivado IP Integrator: Intelligent IP Integration



Correct-by-construction

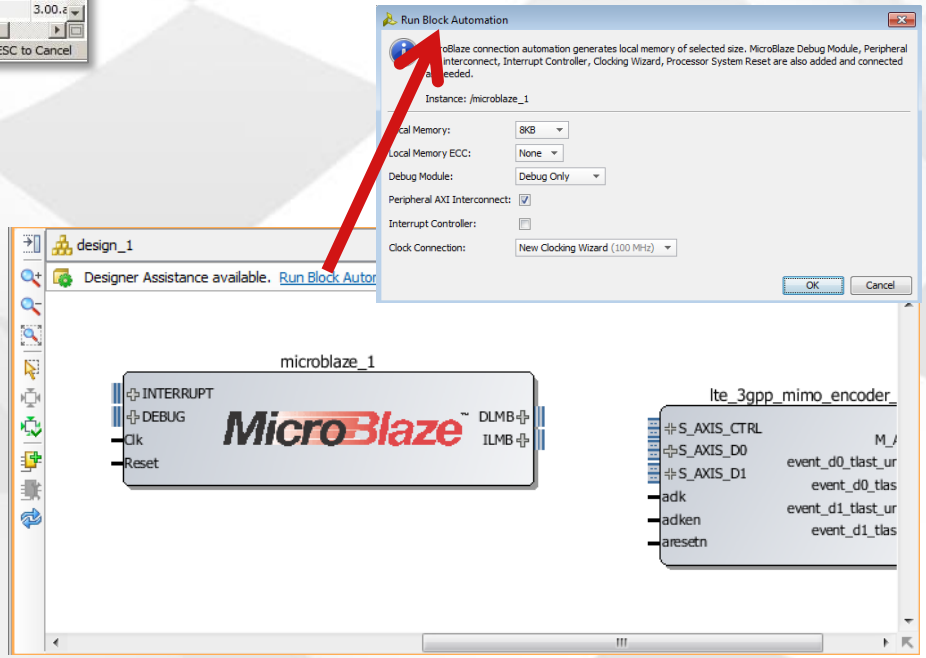
- Interface level connections
- Extensible IP repository
- Real-time DRCs and parameter propagation / resolution
- Designer Assistance

Automated IP Subsystems

- Block automation for rapid design creation
- One click IP customization

Board Aware

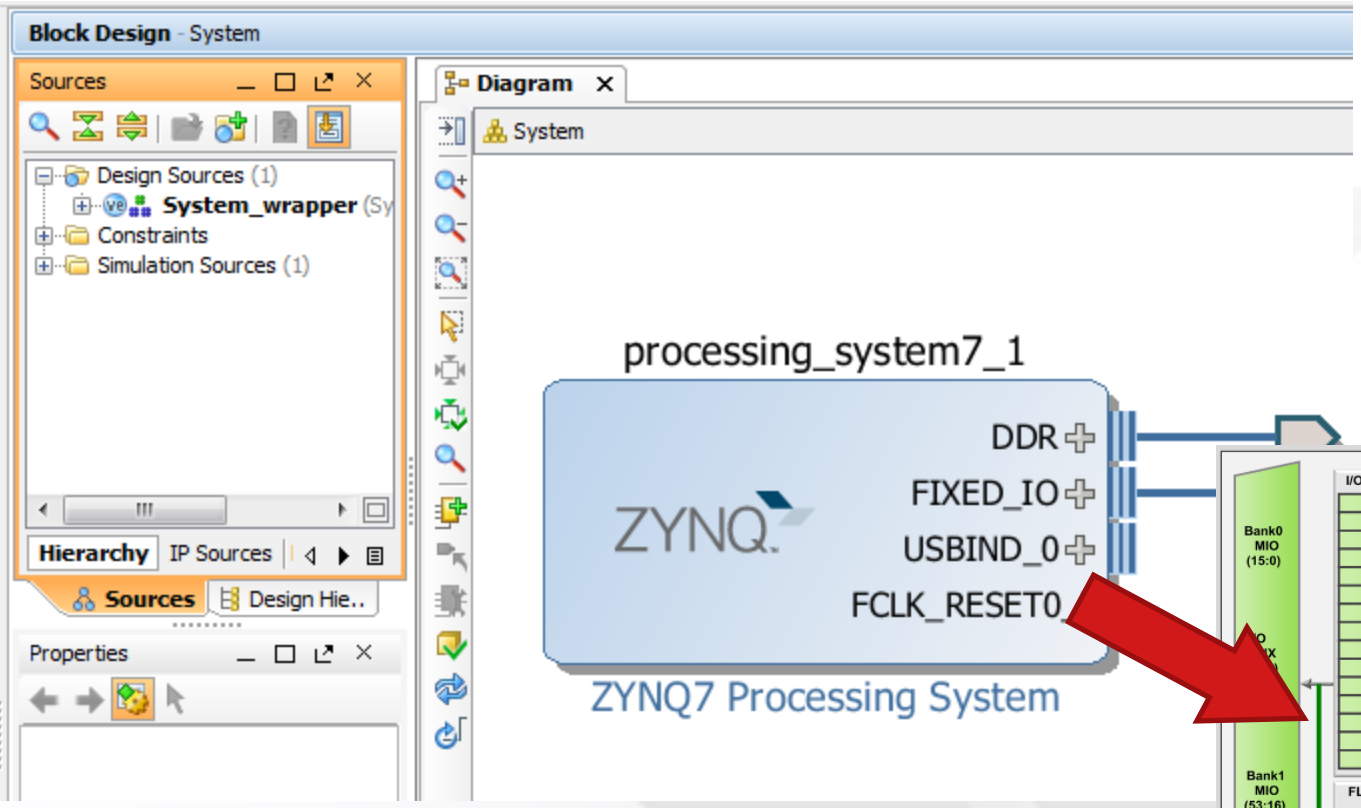
- Support all 7 series and Zynq Platforms

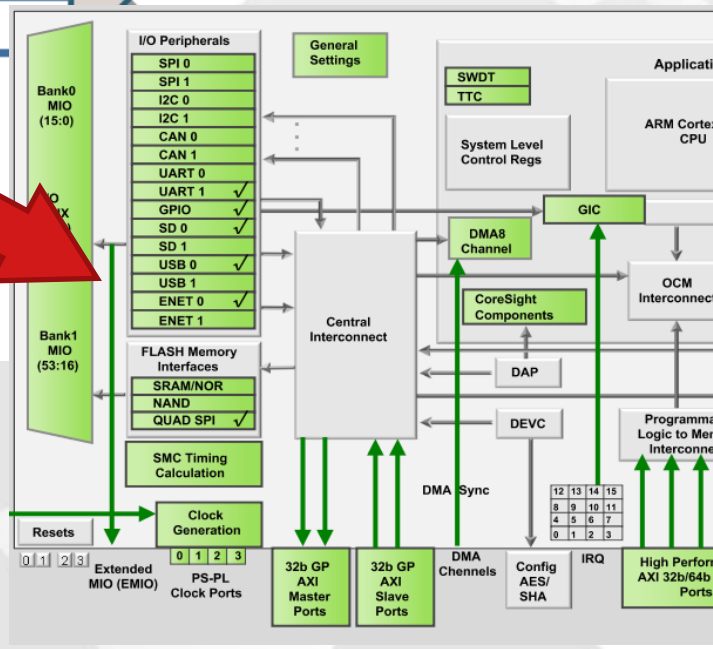


AVNET electronics marketing

Getting Started: Adding an Embedded Source

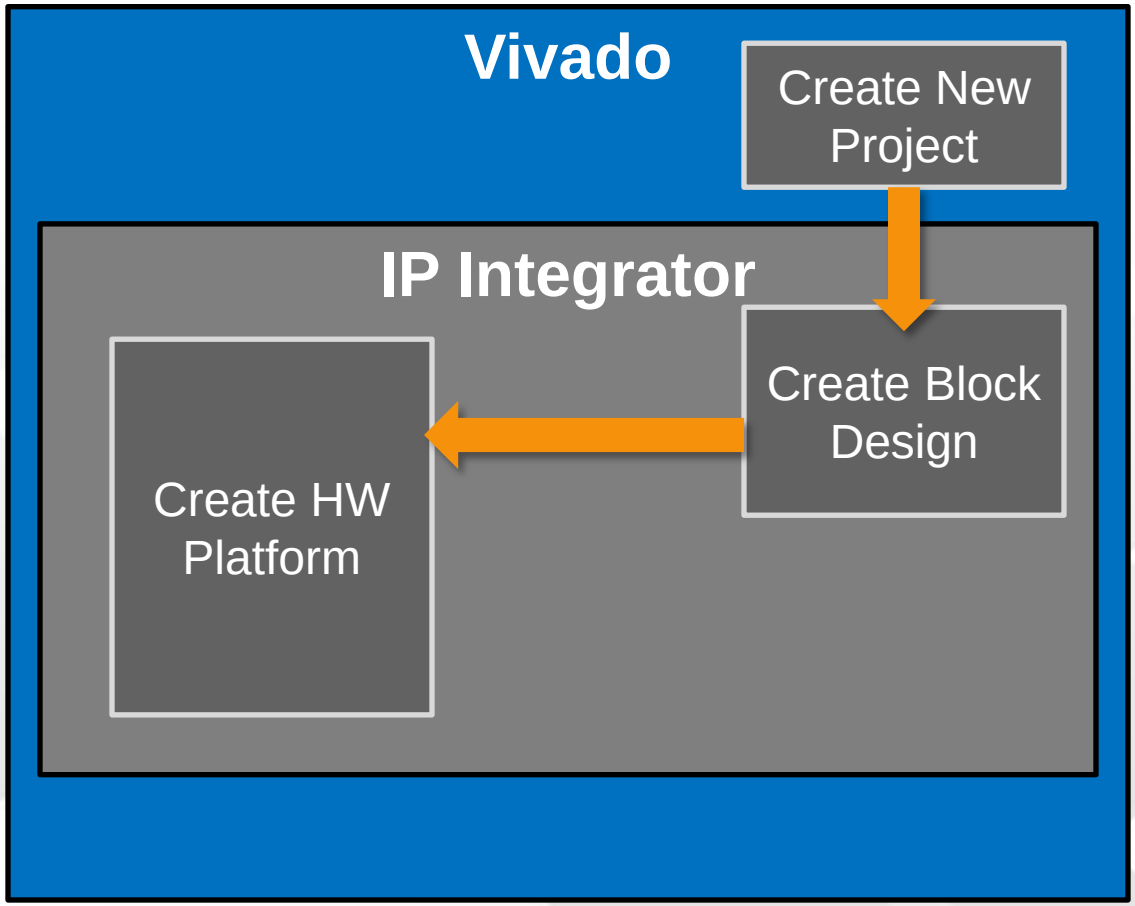
1. **Create Block Design in IP Integrator**
2. **Customize Zynq Processor**

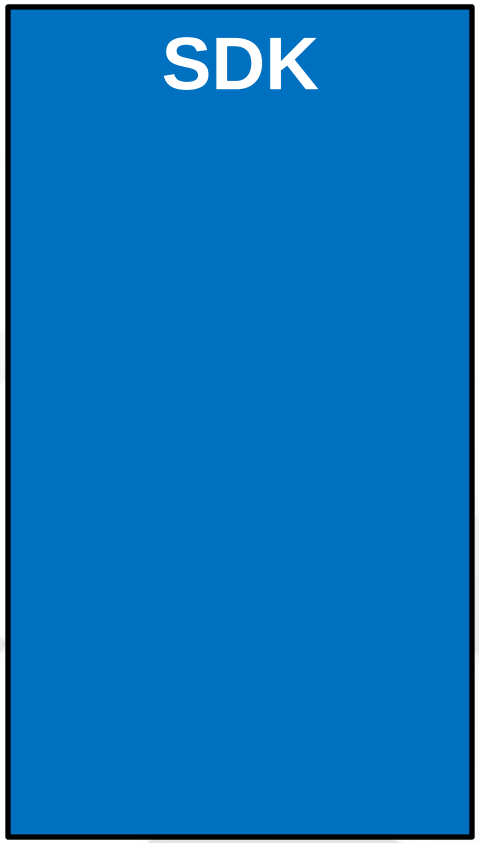




AVNET electronics marketing

Lab 1 – Create Basic System



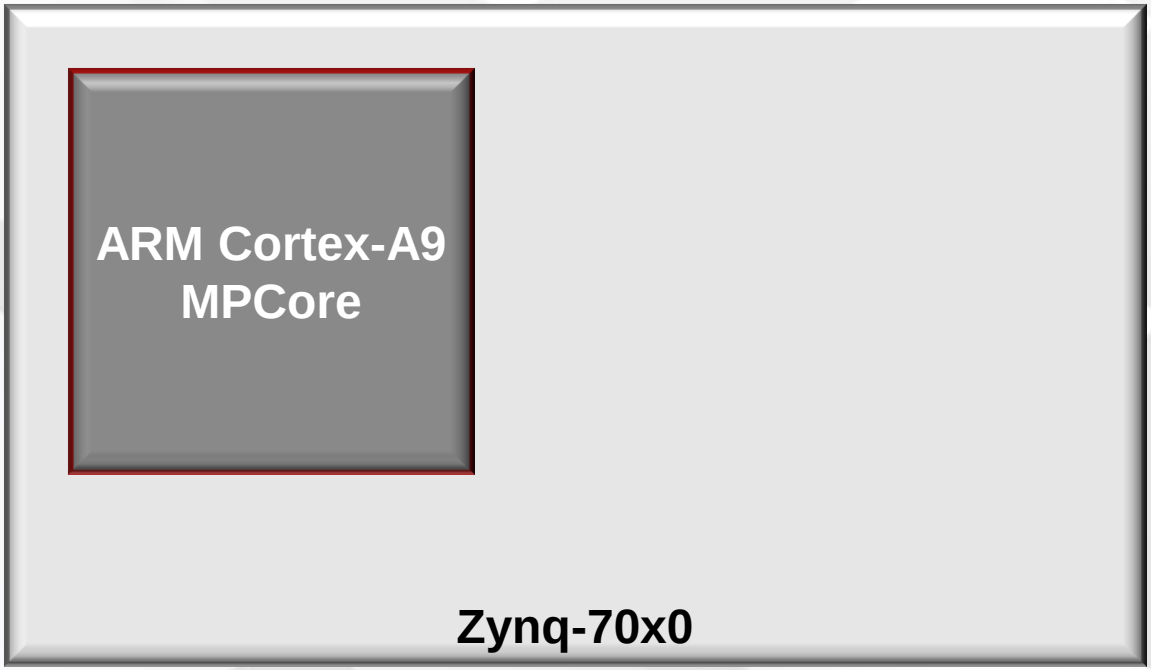


AVNET electronics marketing

Lab 1 – Create a Basic System

- **Create Vivado project**
- **Create a block design**
- **Add an embedded source**

C:/Speedway/ZynqHW/2013_3



Questions

- **What device is selected? What is the Top Module Name?**
  - For MicroZed it is xc7z010clg400-1. For ZedBoard it is xc7z020clg484.
  - The top module name has not been defined yet. But when created, will appear under *Design Sources*.
- **What are the Vivado Synthesis and Implementation Strategies?**
  - Both are set to Default
- **What other information is available in the Project Summary?**
  - Synthesis and Implementation status, DRC violations, timing results, device utilization statistics and power information.
- **List some of configurable components in the Zynq PS?**
  - I/O Peripherals, General Settings, Flash Memory Interfaces, Clock Generation, AXI Ports, DDR Memory Controller, and more.
- **The I/O Peripherals are connected to external I/O through the I/O Mux (MIO), how many I/O are available in the MIO?**
  - 54 Total. 16 in MIO Bank 0 and 38 in MIO Bank 1

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

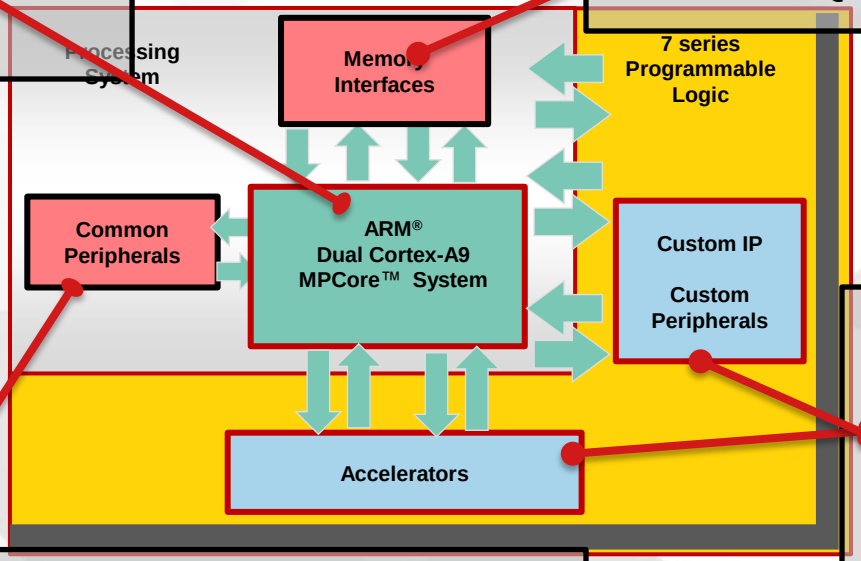
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Complete ARM-Based System

Processing Core Complex
- Dual ARM Cortex A9
- NEON Extensions
- FPU
- Up to 1GHz

High Bandwidth Memory
- 256KB OCM
- 32KB L1 / 512KB L2 Cache
- Memory Controllers
  - DDR3/L, DDR2, LPDDR2
  - 2xQSPI, NOR, NAND

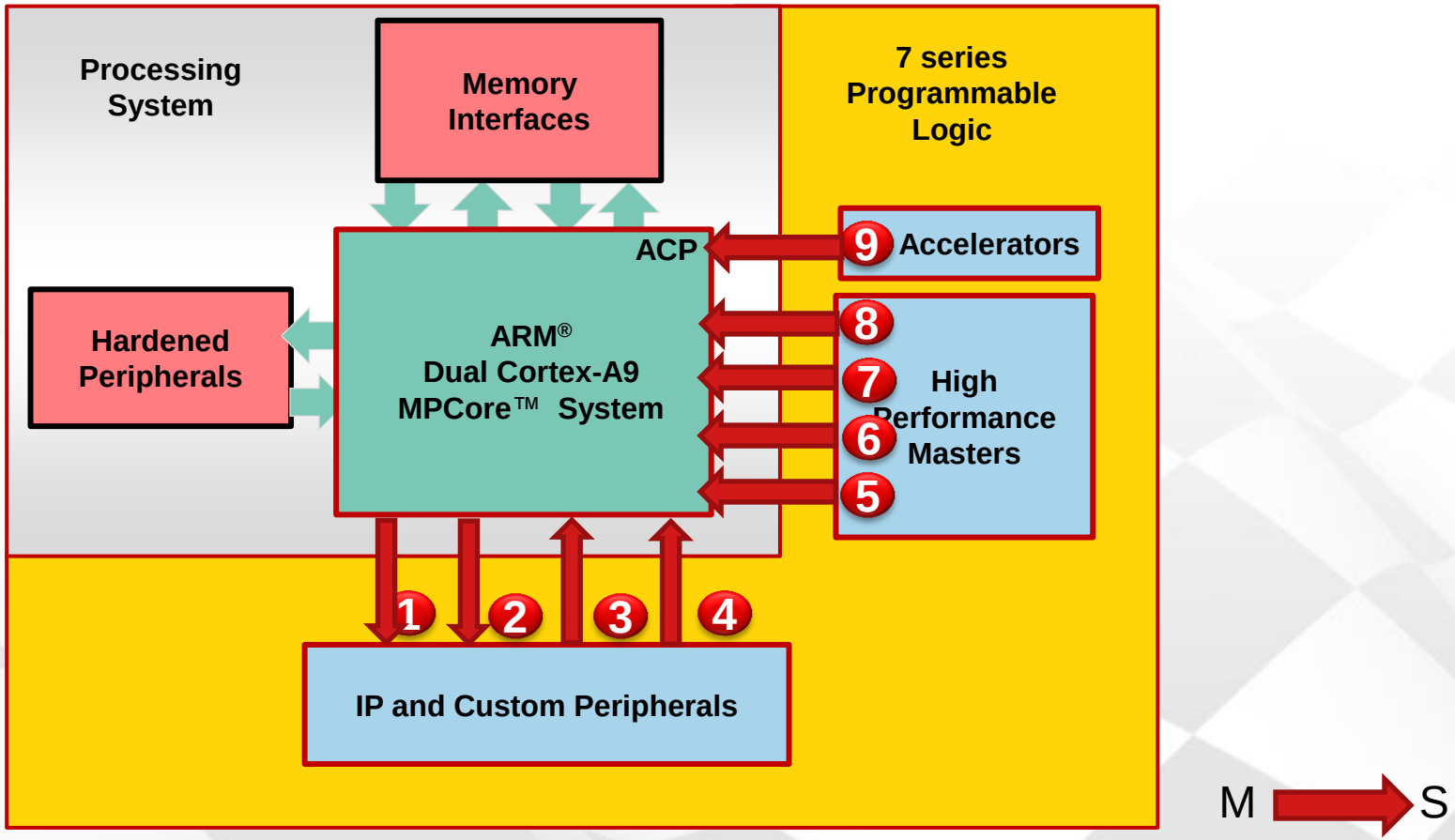


AMBA AXI
- High Bandwidth between PS-PL
- ACP Port
  - Hardware Acceleration

Integrated Memory Mapped Peripherals
- 2x USB 2.0 OTG w/DMA
- 2x TEMAC w/DMA
- 2x SD/SDIO w/DMA
- 2x UART, 2x CAN, 2xI2C, 2x SPI, GPIO

SPEEDWAY DESIGN WORKSHOPS

Zynq-7000 AP SoC Architecture Overview



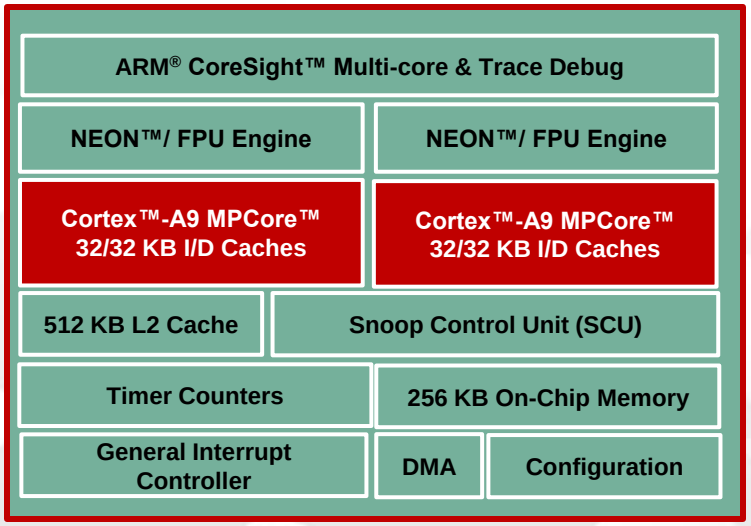
9 Independent PS-to-PL Interface
~100Gbps of Bandwidth

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

ARM Cortex-A9 Processor and L1 Caches

- **ARM Cortex-A9 processor key features**
  - 2.5 DMIPS/MHz or 1128 CoreMark™ per core (@667MHz)
  - Up to 1GHz operation
  - Harvard architecture, independent
    - 64-bit data
    - 64-bit instruction
  - Little endian support for both instruction and data



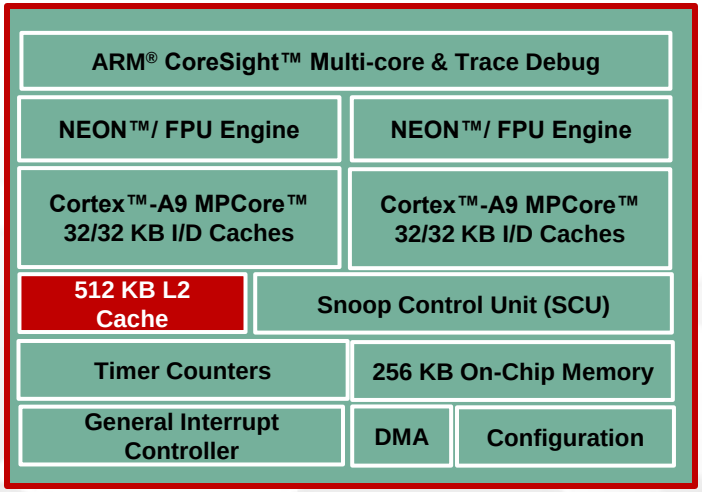
- **ARM Cortex-A9 processor L1 cache key features**
  - 32KB Instruction and 32KB Data cache
  - The cache line length is eight words (32 bytes)
  - All L1 caches support parity
  - 4-way set associative, write-back

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

L2 Cache Controller

- **High performance L2 cache controller**
  - 512KB of cache
  - Supports lockdown by line for routines that might need low latency
  - 8-way set associative
  - Parity support
  - Write-through and write-back support



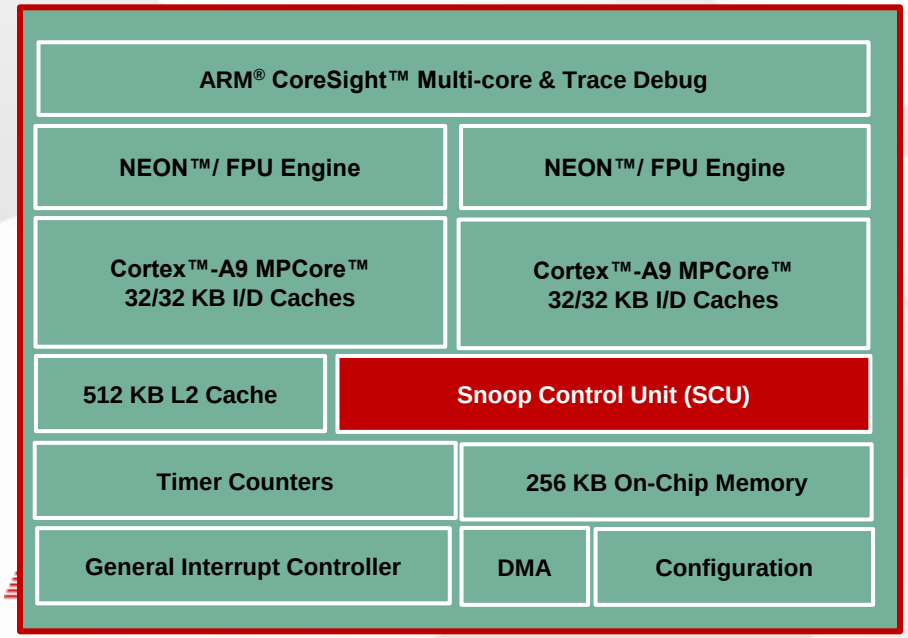
- **L2 cache controller AXI interfaces**
  - One AXI master (M) interface for the DDR controller
  - One AXI master interface for all slave devices in the PL and PS
  - One AXI slave (S) interface for the Snoop Control Unit

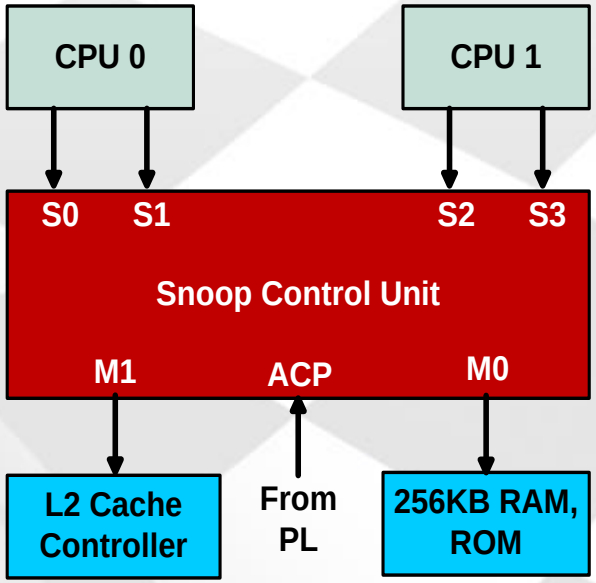
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Snoop Control Unit (SCU)

- **The Snoop Control Unit (SCU) connects two Cortex-A9 processors to the system memory/peripherals via AXI interfaces in Zynq-7000 AP SoC**
  - Provides L1 data cache coherency between the Cortex-A9 processors and the Accelerator Coherence Port (ACP)
  - Arbitrates between Cortex-A9 processors requesting L2 cache accesses
  - Provides access to the on-chip ROM and RAM
  - Manages Accelerator Coherence Port (ACP) accesses

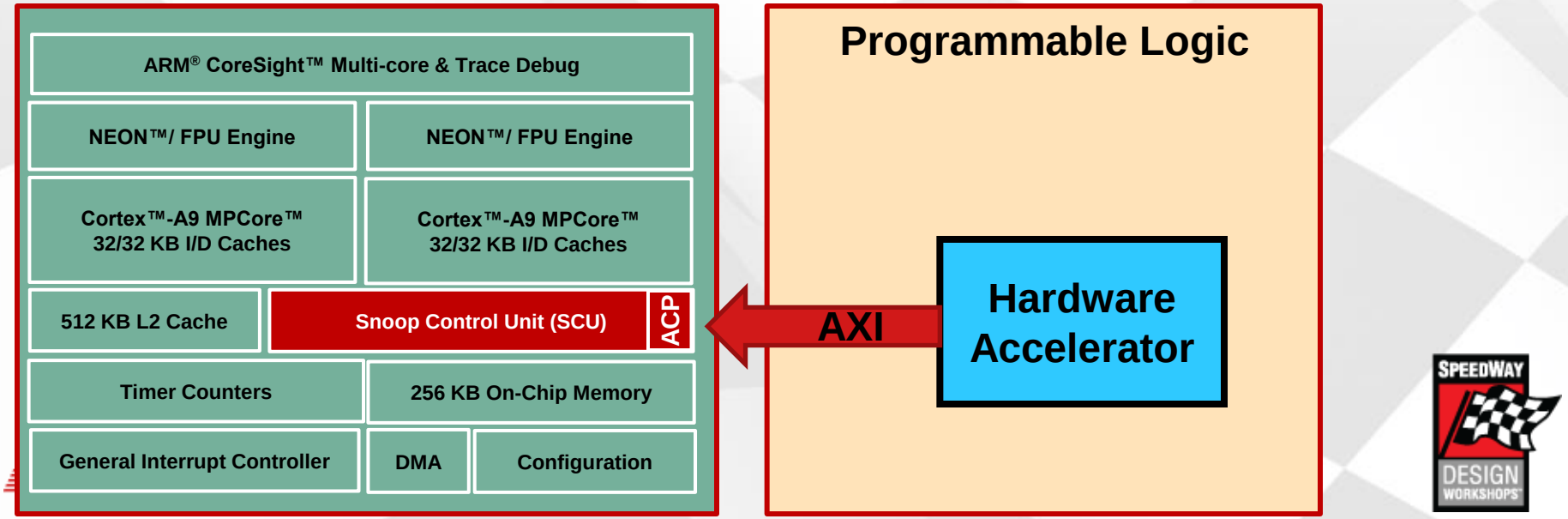




Arrow shows direction of the Master

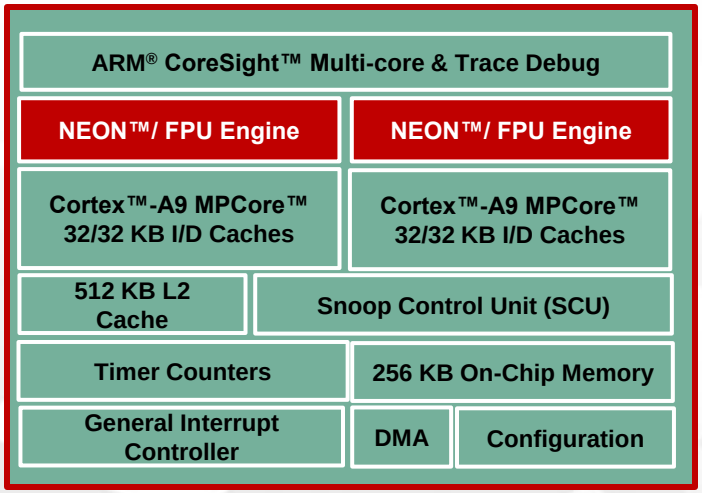
Accelerating Software with Hardware

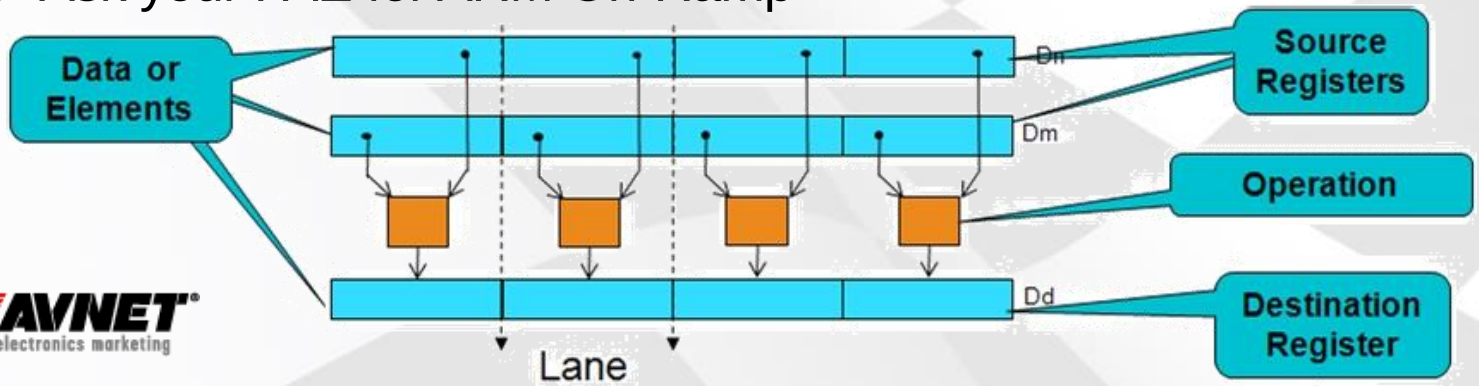
- **Accelerator Coherency Port (ACP)**
  - The Accelerator Coherency Port is a slave AXI 64-bit port on the SCU connected to the PL (150MHz in a -1 device)
  - ACP port can be used by accelerators in the PL to access the ARM L1/L2 caches and maintain coherency
  - Cache accesses could be used as a means to share data between the A9 CPUs and hardware accelerator in the PL with low latency



Cortex-A9 NEON™ Processing

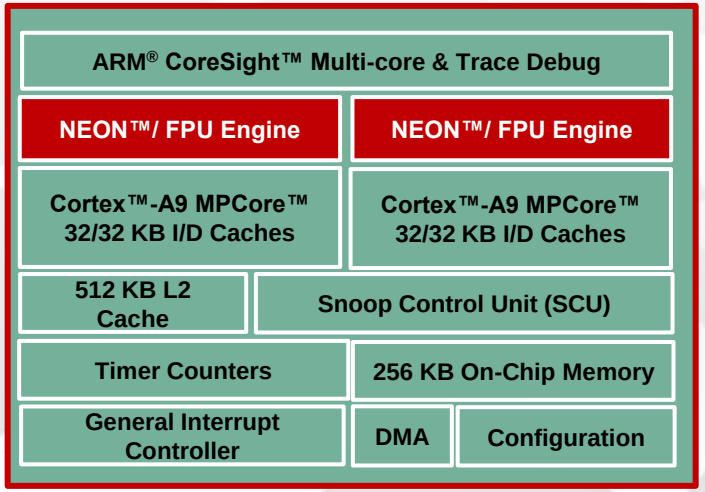
- **A co-processor that extends the ARM instruction set targeted at audio, video, 3-D graphics, image and speech processing applications**
- **NEON SIMD instruction supports multiple**
  - 8, 16, 32, and 64-bit integer data
  - 32-bit floating-point data (via **FPU**)
- **Instructions need fewer cycles than ARM**
  - NEON typically half
- **Disabled at reset, enabled via software**
- **Requires Linaro or ARM compiler, or specialized Library**
  - Ask your FAE for ARM On-Ramp





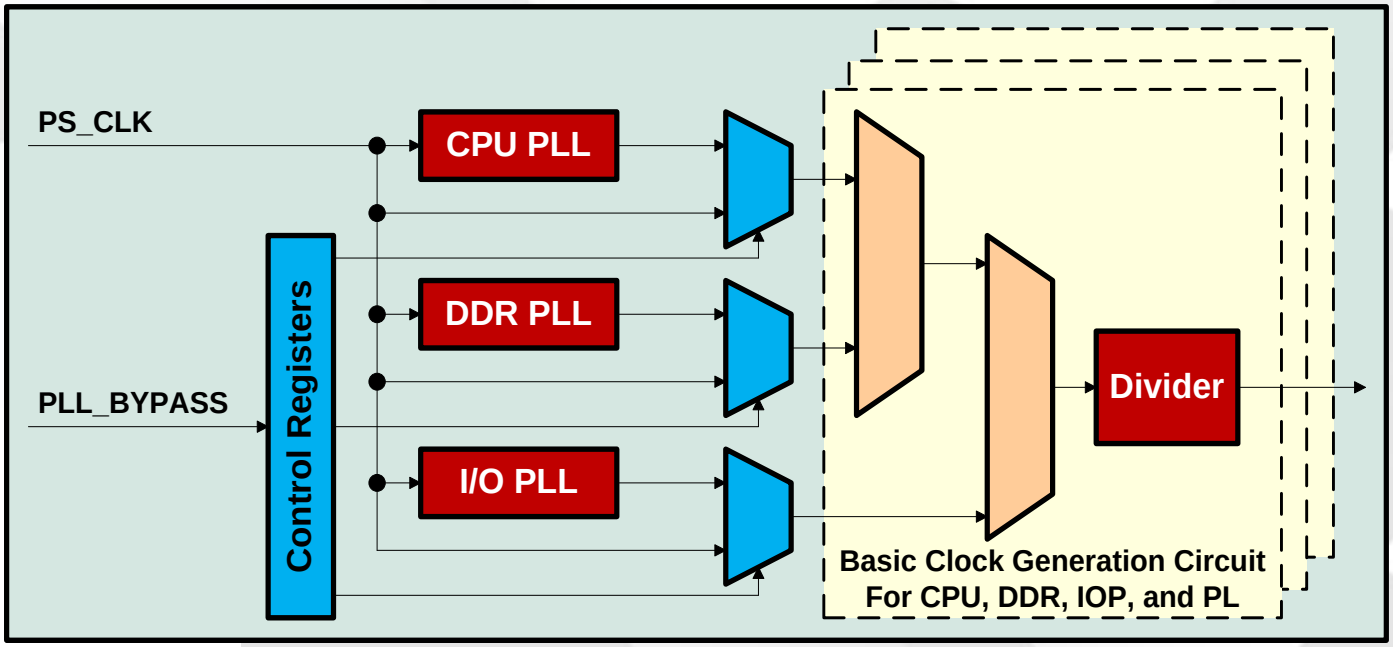
FPU Engine

- **Extension to NEON sub-processor (not independent component)**
- **High performance single/double precision floating-point operations**
  - VFP v3 FPU (no vector feature)
  - New half-precision conversions (FP16)
- **Compliant with IEEE-754 standard with noted exceptions**
  - Refer to the UG585 for more information
- **Register set shared with NEON**
- **2 MFLOPS/MHz performance**



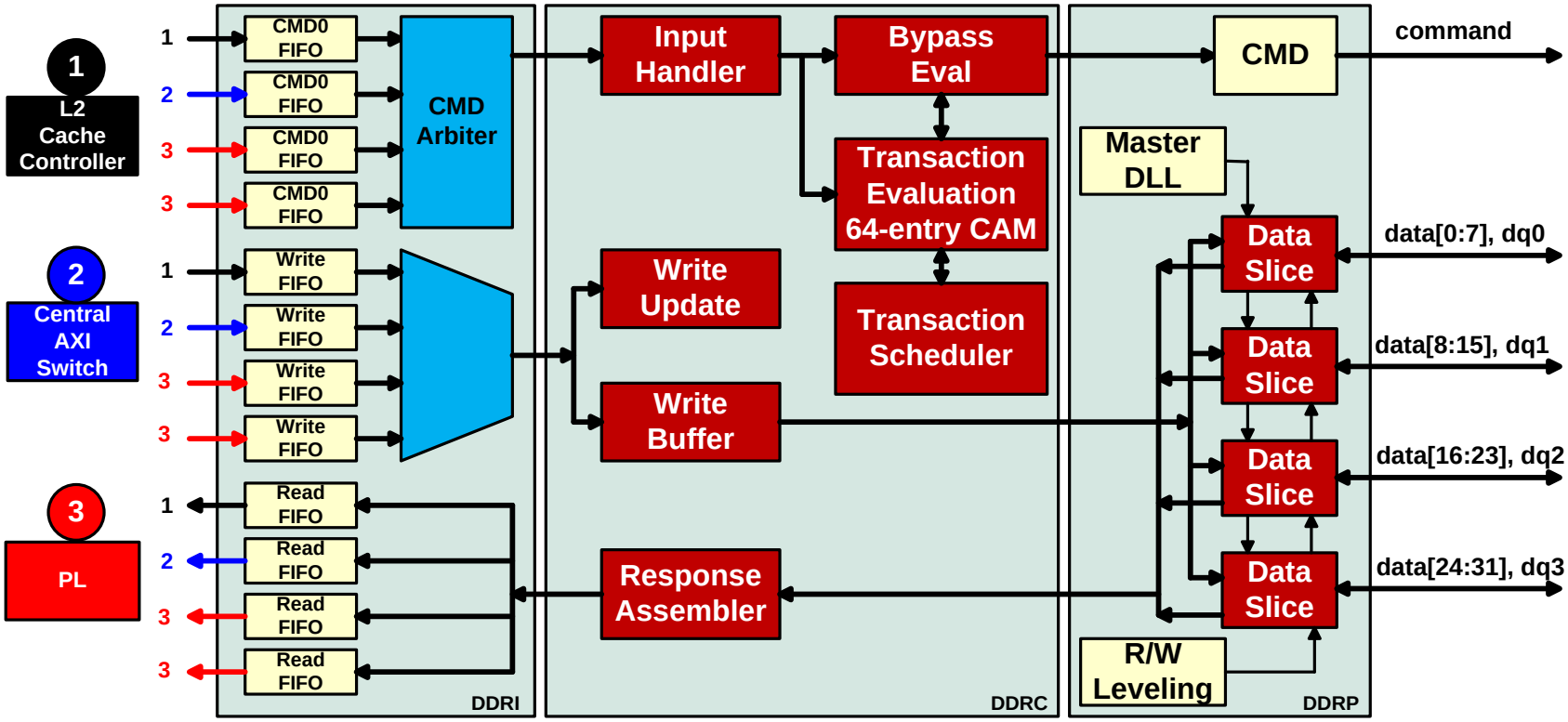
Zynq-7000 AP SoC Clock Generation Module

- **Generates clocks for the CPU, DDR, PL, and all common peripherals**
  - PS_CLK is the PS reference clock input and is required to be 30-60 MHz
    - A 33.333 MHz reference frequency is recommended
  - Each clock generation circuit has a PLL source selection multiplexer followed by a programmable divider
  - Four general-purpose clock outputs are generated for the PL
  - Xilinx tools provide means to set the clock frequencies



DDR Memory Interface

- **DDR memory interface features consist of**
  - Support for LPDDR2, DDR2, and DDR3/L (400, 400, 533MHz in a -1 device)
  - Configurable 16-bit and 32-bit external DDR data bus width
  - ECC with one bit error correction and two bit error detection (16-bit only)
  - One external chip select, no DIMM or parity support



Checkpoint!

- **What is the cost of the tools to develop a Zynq design?**

  Nothing, Tools are FREE for 7010, 7015 , 7020 and 7030 devices

- **How many AXI interfaces exist between the PS and PL?**

  Nine – 4 GP, 4 HP and 1 ACP

- **What is the max CPU frequency?**

  1 GHz

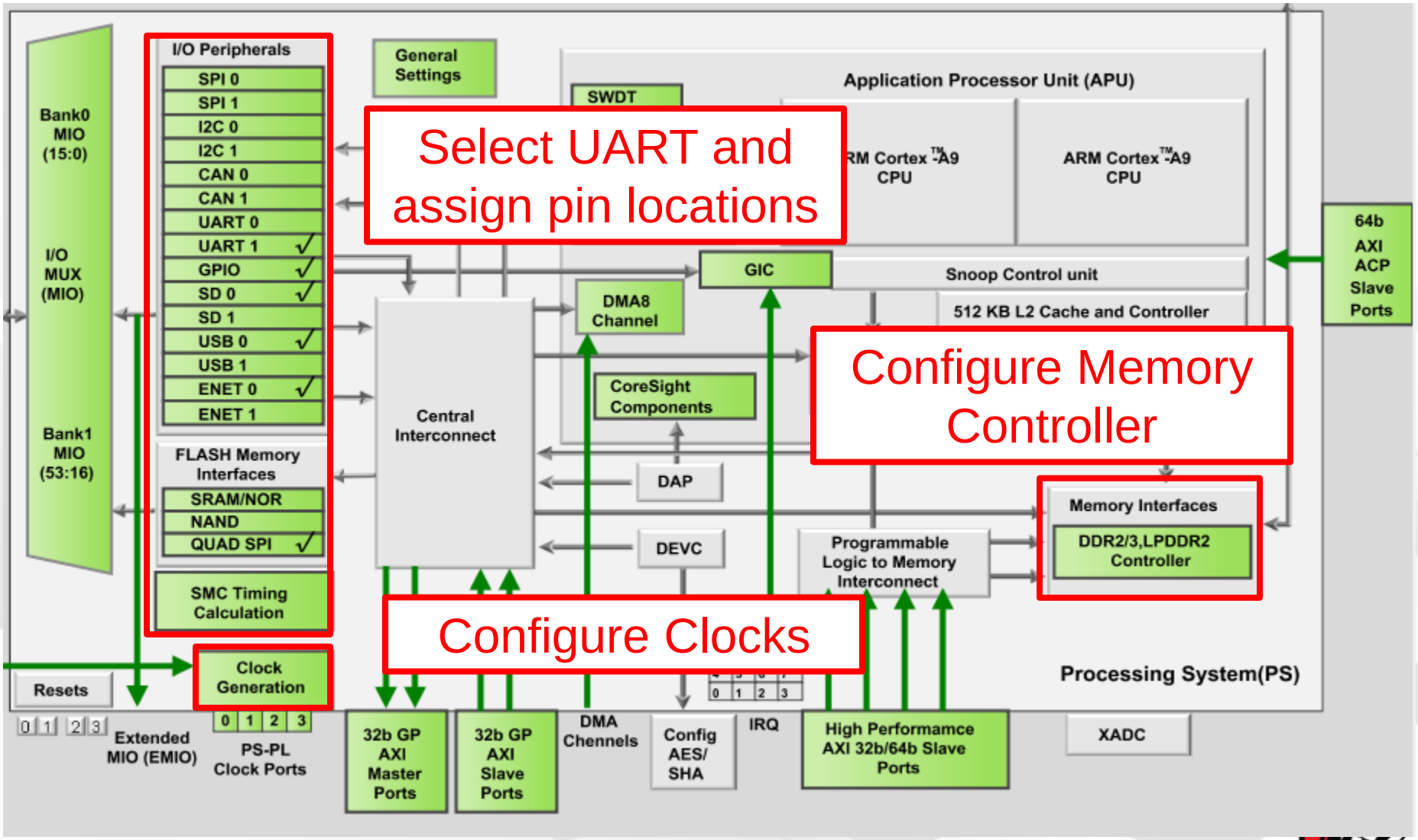
- **What is the recommended input clock frequency?**

  33.333 MHz

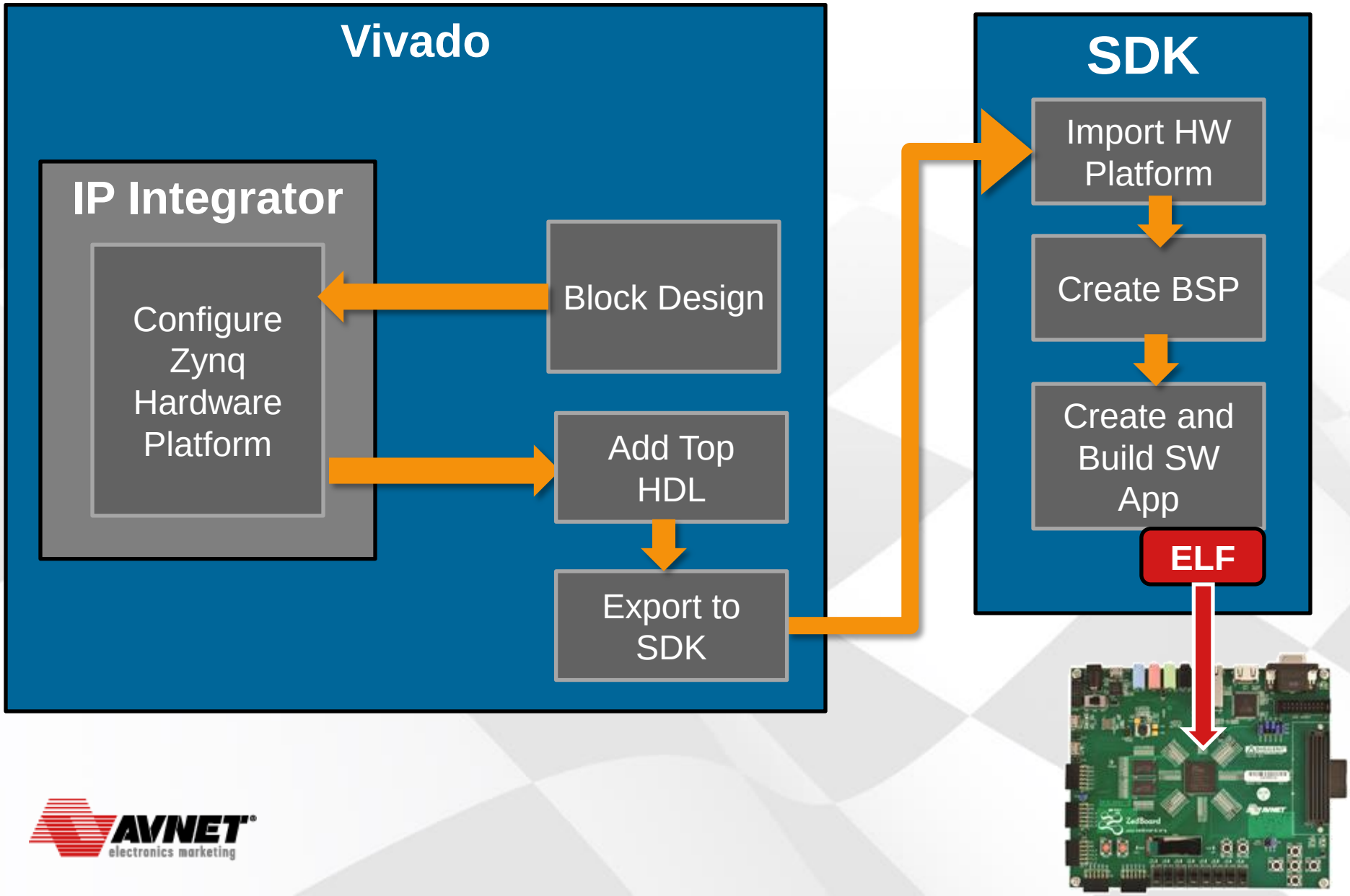
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Lab 2 - PS Configuration

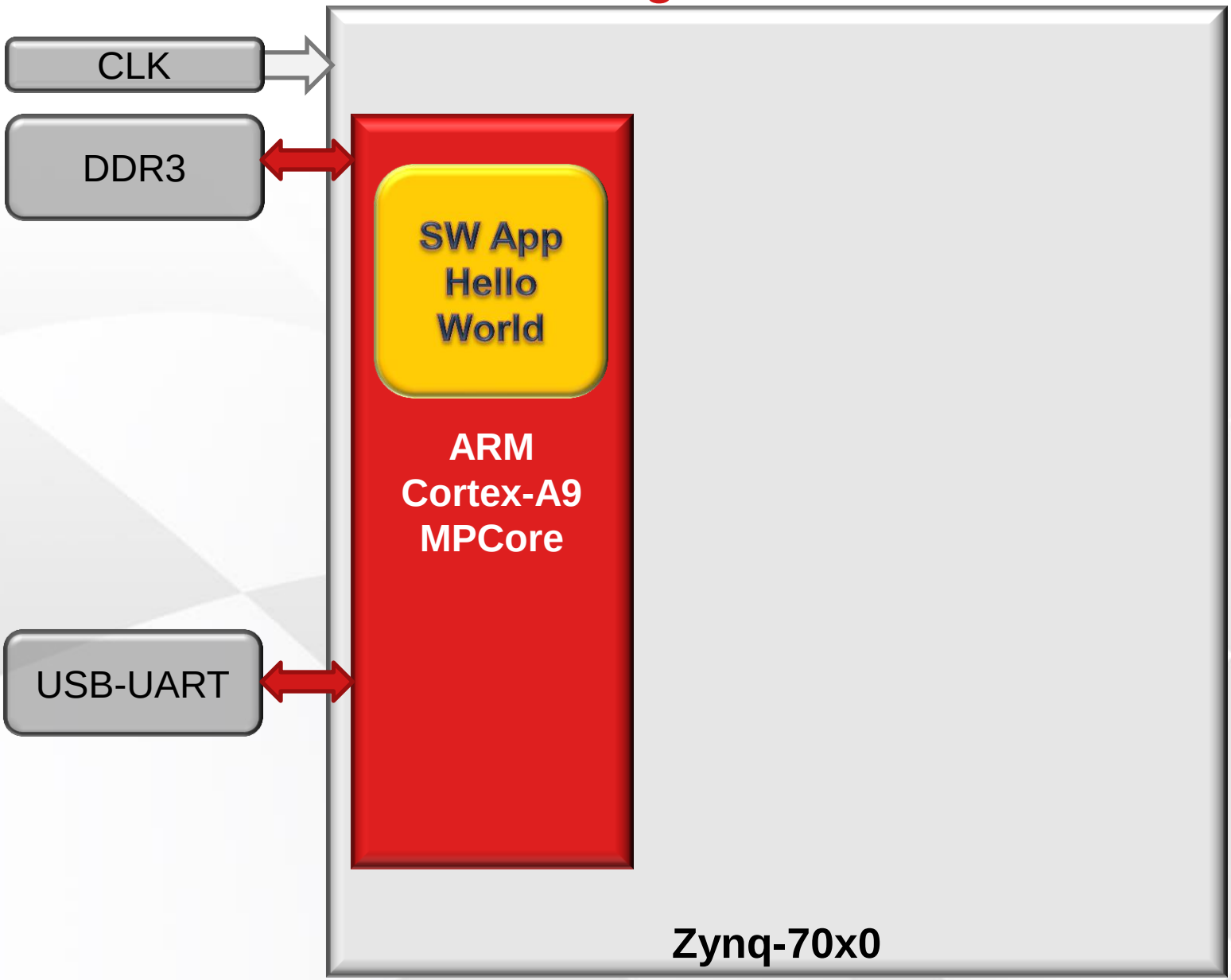


Lab 2 – Basic PS Configuration



AVNET electronics marketing

Lab 2 – Basic PS Configuration



Questions

- ***What do you think is the purpose of EMIO?***
  - EMIO provide PL and PL I/O access to PS peripherals.
- ***Why are the Peripherals not listed alphabetically in the I/O Peripherals Configuration tool?***
  - The peripherals are listed in order of priority. Peripherals on the top of the list generally need to be connected first and less MIO options.
- ***Extra Credit: If the Modem Signals are used with one of the UART peripherals, where must they be mapped?***
  - EMIO
- ***Where can the DDR interface speed be set?***
  - In the Clock Configuration window or the DDR Configuration window.
- ***Where did Vivado get the Memory Part Configuration Settings? Where would you get them for a custom part?***
  - Vivado has a preset library of memory part details. With each release of Vivado, Xilinx adds new memory devices to this library.
  - If you use a custom part, these values must be extracted from the specific DDR datasheet.

Questions

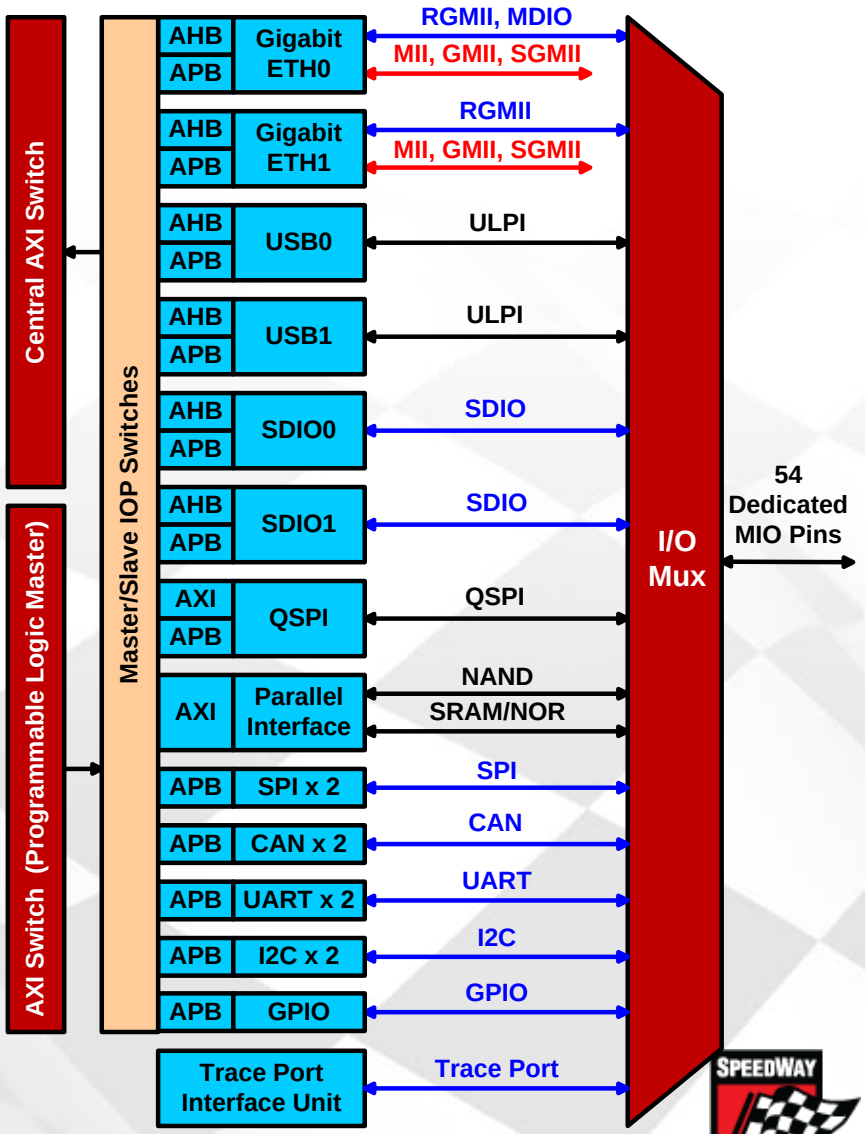
- ***What is the maximum speed the DDR3 interface can run at? Extra Credit: What is the slowest?***
  - 533MHz. Slowest? It depends on the DDR3 memory device. Even though Vivado lists the slowest speed as 10MHz, the DDR3 memory device's internal PLL may not be able to run that slow. When a known memory device is selected, Vivado prevents users from setting a speed to slow. In this case, 303MHz.
- ***Does the Hello World C source include Zynq initialization?***
  - No
- ***How did the Zynq get initialized in this Hello World experiment?***
  - When you created the Run Configuration, there was a Device Initialization tab. Here there is a field to point to an initialization TCL file. This is set by default to the ps7_init.tcl file that was created as part of the Export to SDK. Inside this TCL, you will find a number of XMD commands that initialize all the registers exactly how you specified in XPS.

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

Zynq-7000 AP SoC Common Peripherals

- **The PS common peripherals**
  - Gigabit Ethernet w/DMA (2 ports)
  - USB 2.0 w/DMA (2 ports)
  - SD/SDIO w/DMA (2 ports)
  - CAN (2 ports)
  - I2C (2 ports)
  - UART (2 ports)
  - SPI (2 ports)
  - NAND (x8/x16)
  - NOR/SRAM (x8)
  - Quad SPI
  - Up to 118 GPIO ports (54 MIO + 64 EMIO)
- **Peripheral I/O can be routed to the PS MIO or PL Extended MIO (EMIO)**

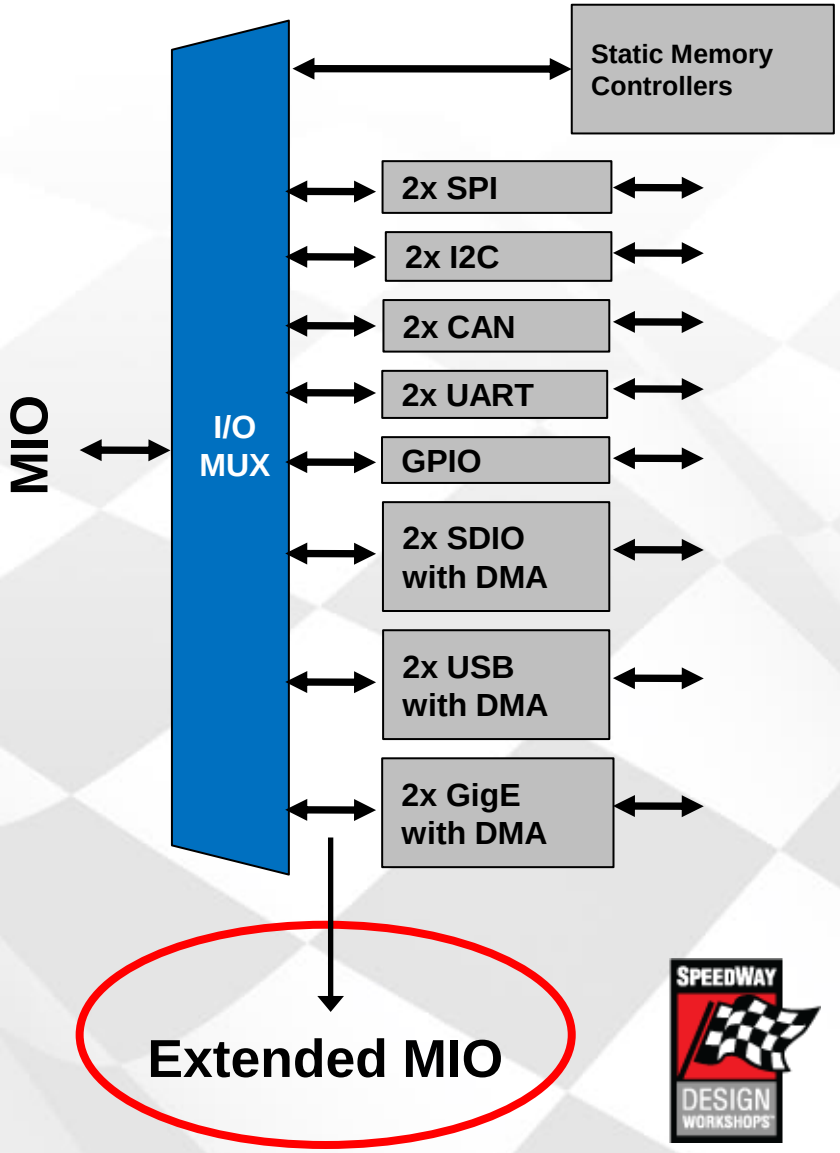


AVNET electronics marketing

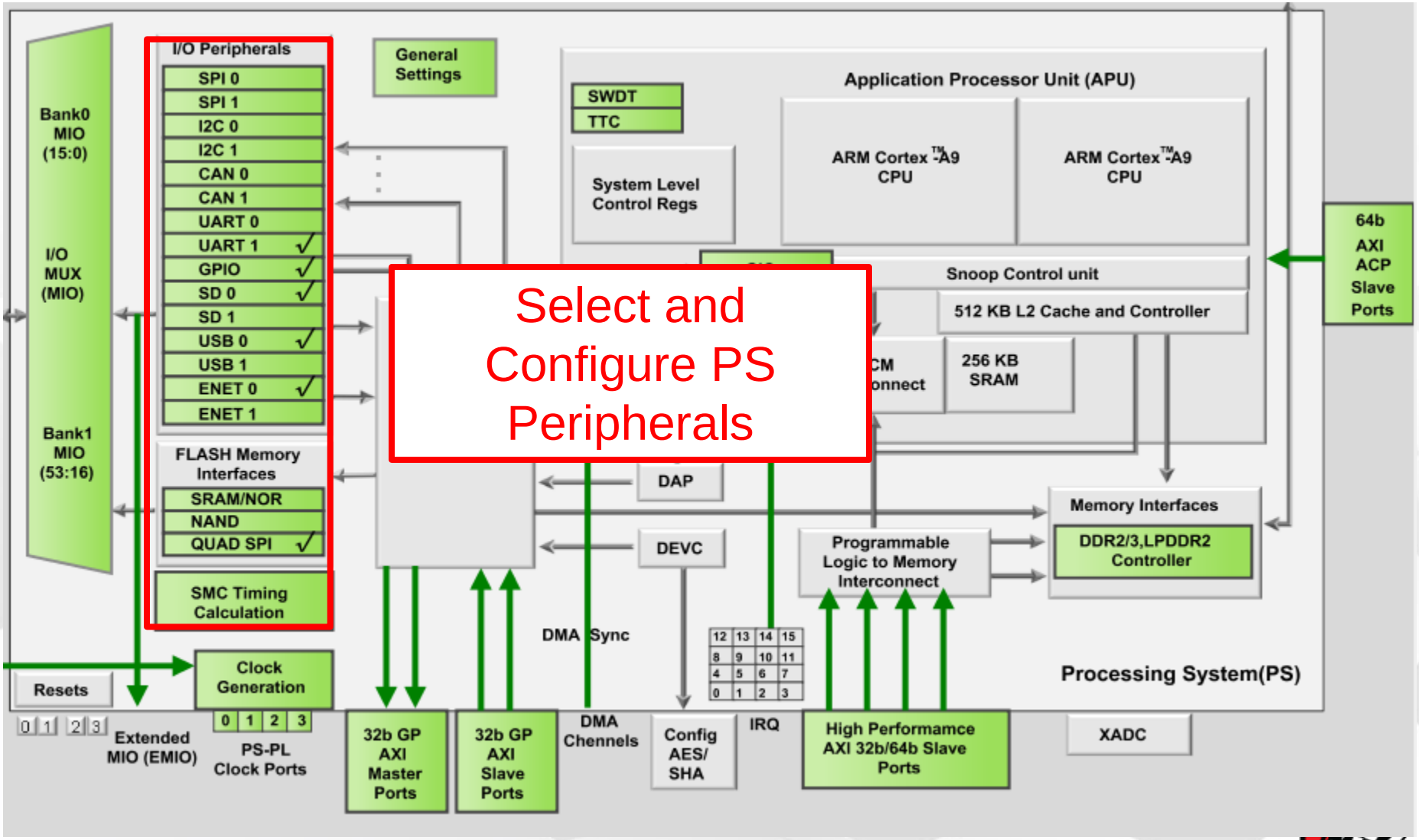
SPEEDWAY DESIGN WORKSHOPS

Zynq-7000 AP SoC Common Peripherals I/O Pins

- **Peripheral I/O connected to MIO only**
  - USB0 and USB1 ULPI
  - QSPI
  - NAND Flash
  - SRAM/NOR Flash
  - ETH0 or ETH1 via RGMII
- **Peripheral I/O connected to MIO or EMIO**
  - SDIO0 and SDIO1
  - CAN
  - I2C
  - UART
  - SPI
  - GPIO
  - Trace Port
  - Processor JTAG
- **Peripheral I/O connected to EMIO only**
  - ETH0 and ETH1 MII, GMII, SGMII
    - RGMII with PL-shim
  - UART Modem Signals

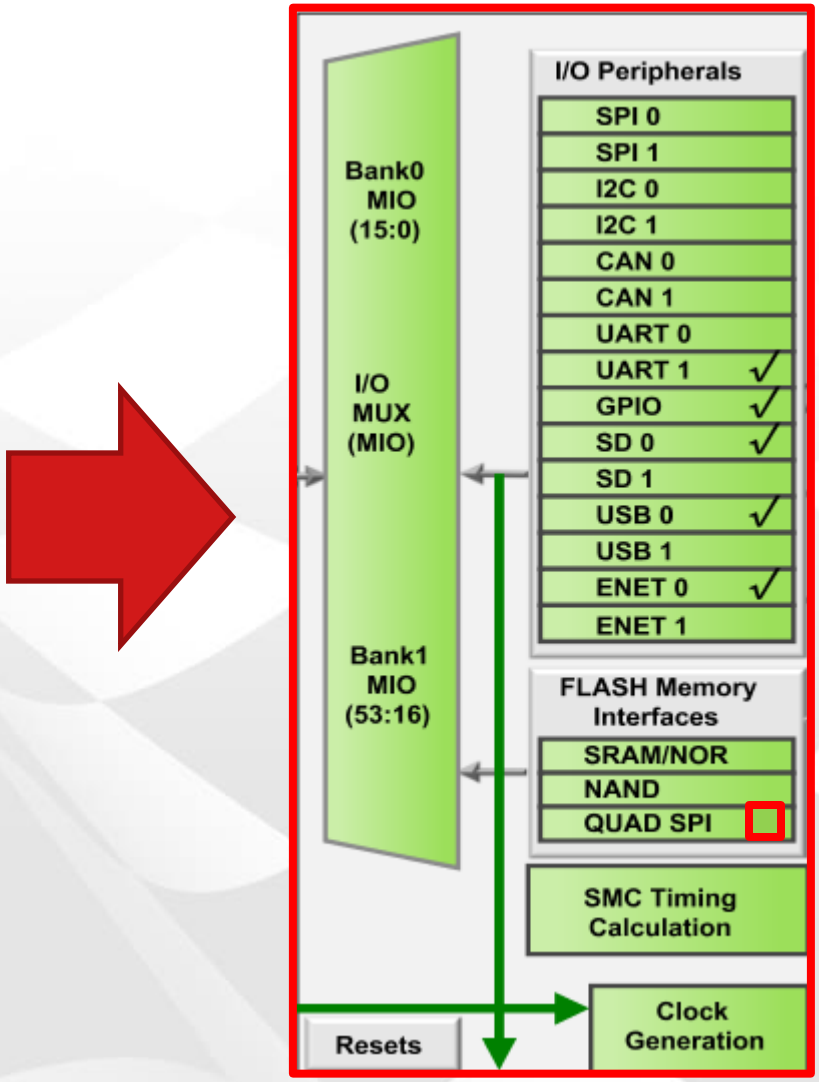


PS MIO Configuration



Select Peripherals

Peripheral	IO
Memory Interfaces	
Quad SPI Flash	MIO 1 .. 6
SRAM/NOR Flash	
NAND Flash	
I/O Peripherals	
ENET 0	MIO 16 .. 27
ENET 1	
USB 0	MIO 28 .. 39
USB 1	
SD 0	MIO 40 .. 45
SD 1	
UART 0	
UART 1	MIO 48 .. 49
I2C 0	
I2C 1	
SPI 0	
SPI 1	
CAN 0	
CAN 1	
GPIO	



Zynq Address Map

Address Range	CPUs & ACP	AXI_HP	Other Bus Masters	Description
0000_0000 to 000F_FFFF	DDR/OCM	OCM	OCM	SCU and OCM
0010_0000 to 3FFF_FFFF	DDR	DDR	DDR	DDR, All interconnect masters
4000_0000 to 7FFF_FFFF	PL		PL	General Purpose Port #0 to the PL, M_AXI_GP0
8000_0000 to BFFF_FFFF	PL		PL	General Purpose Port #1 to the PL, M_AXI_GP1
E000_0000 to E02F_FFFF	IOP		IOP	MIO Peripheral registers
E100_0000 to E5FF_FFFF	SMC		SMC	Static Memory Controller (NAND, NOR, SRAM)
F800_0000 to F800_0BFF	SLCR		SLCR	System Level Control Registers
F800_1000 to F880_FFFF	PS		PS	PS System registers
F890_0000 to F8F0_2FFF	CPU			CPU Private Registers
FC00_0000 to FDFF_FFFF	QSPI		QSPI	Quad-SPI linear address for linear mode
FFFC_0000 to FFFF_FFFF	OCM	OCM	OCM	OCM Memory (Mapped High)

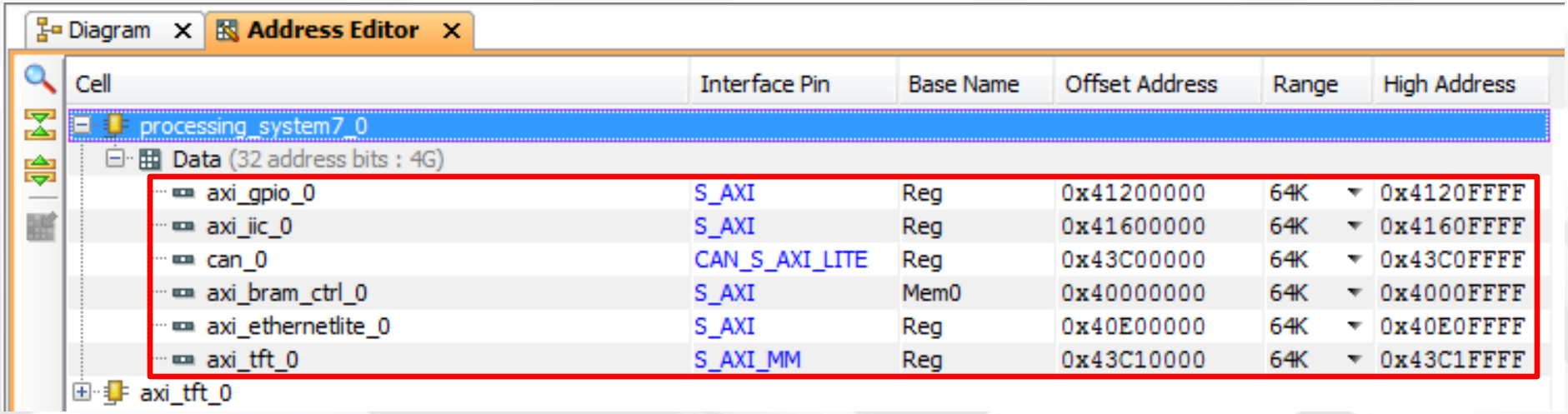
Zynq Address Map

Address Range	CPUs & ACP	AXI_HP	Other Bus Masters	Description
0000_0000 to 0				
0010_0000 to 3				
4000_0000 to 7				
8000_0000 to B				
E000_0000 to E				
E100_0000 to E				
F800_0000 to F				
F800_1000 to F				
F890_0000 to F				
FC00_0000 to F				
FFFC_0000 to F				

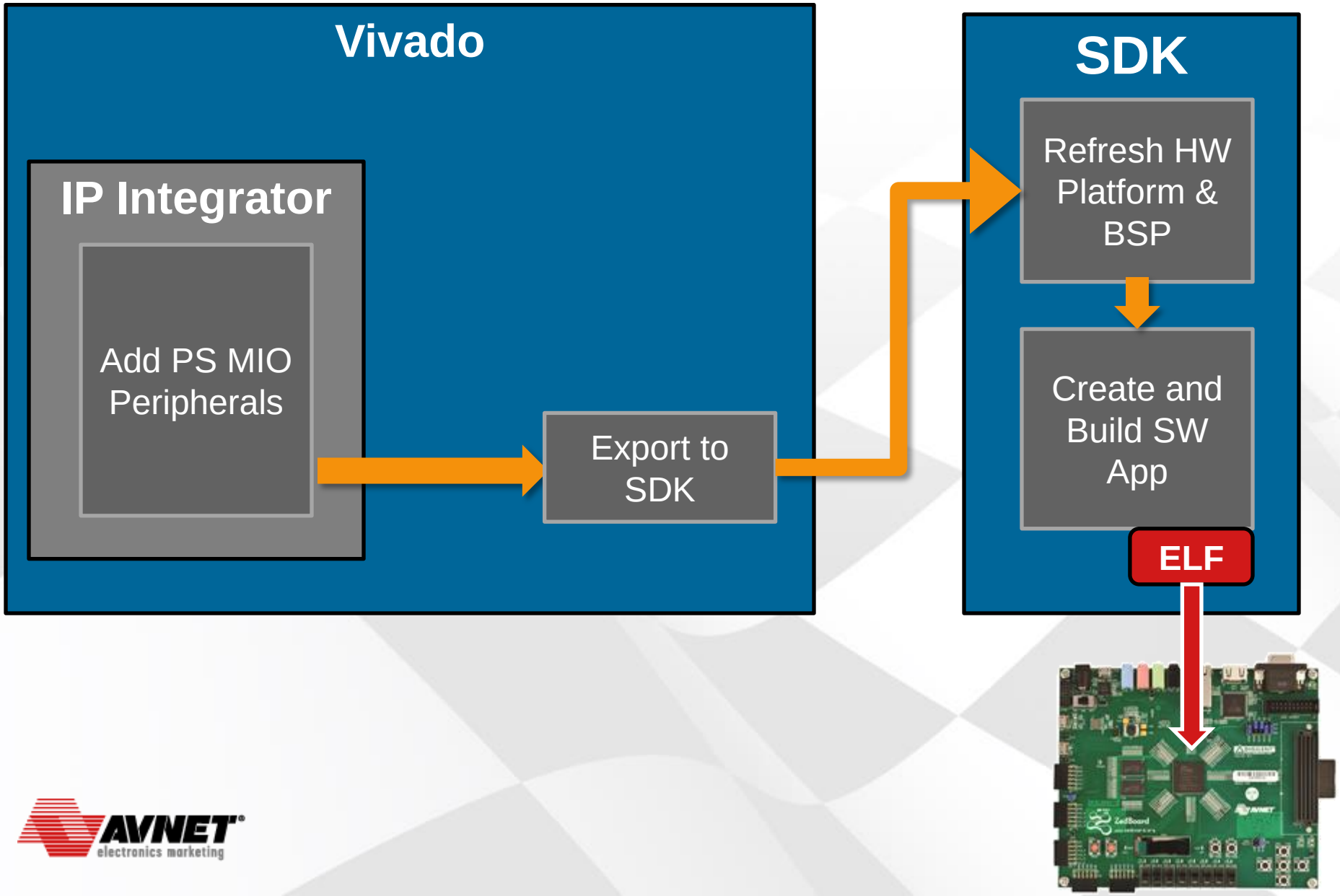
Address Base	MIO Peripheral
E000_0000, E000_1000	UART Controllers 0, 1
E000_2000, E000_3000	USB Controllers 0, 1
E000_4000, E000_5000	I2C Controllers 0, 1
E000_6000, E000_7000	SPI Controllers 0, 1
E000_8000, E000_9000	CAN Controllers 0, 1
E000_A000	GPIO Controller
E000_B000, E000_C000	Ethernet Controllers 0, 1
E000_D000	Quad-SPI Controller
E000_E000	Static Memory Controller (SMC)
E010_0000, E010_1000	SDIO Controllers 0, 1

Address Mapping Sub-system View

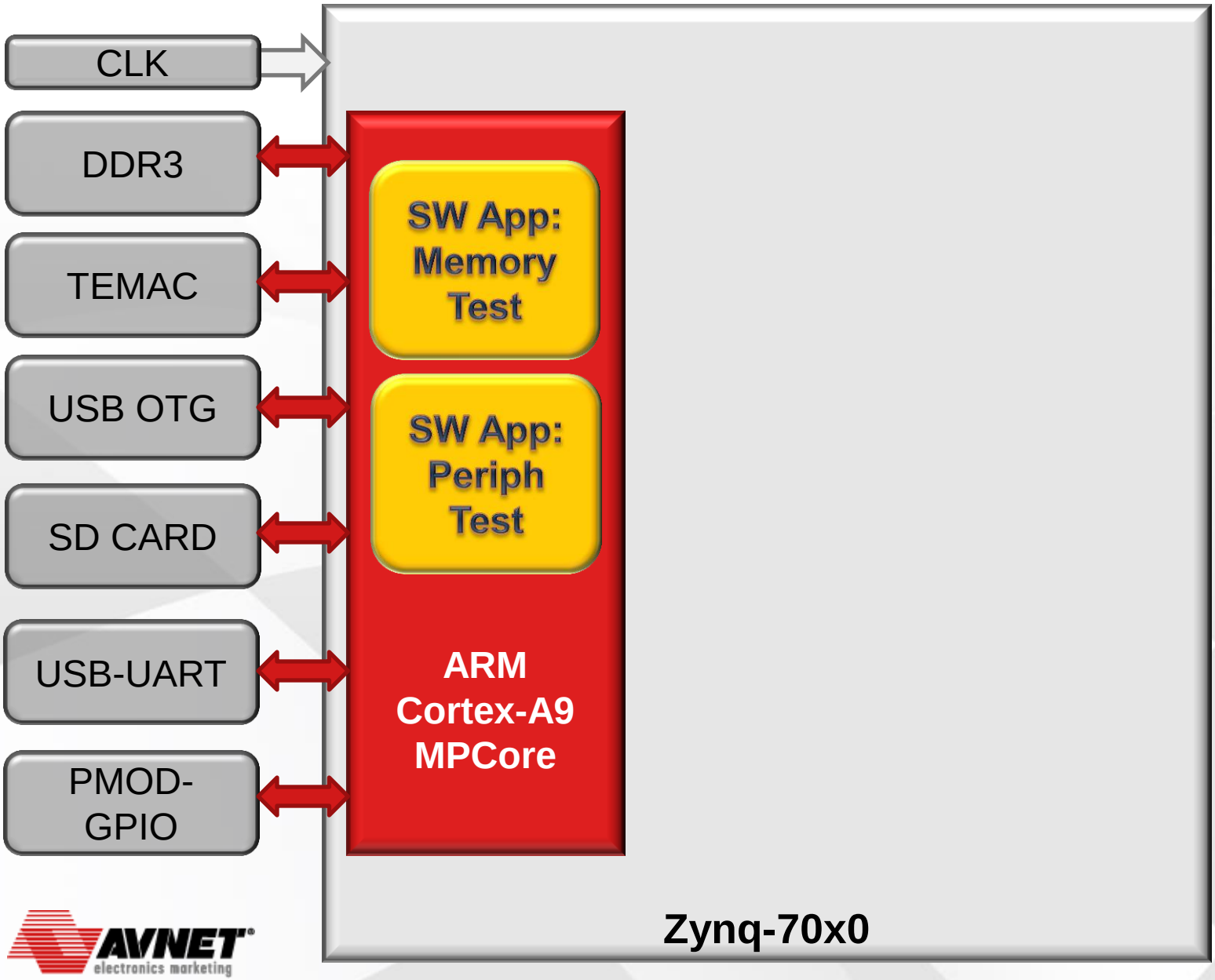
- **Vivado automatically assigns addresses for PL-based peripherals**



Lab 3 – Add PS MIO Peripherals



Lab 3 – Add MIO Peripheral & Create Test App



Questions

- ***Why is EMIO not an option for connecting USB 0?***
  - Due to the ULPI interface standard for the USB peripherals, timing could not be met by going through the EMIO. Therefore, USB peripherals MUST be connected to MIO or they are lost.
- ***Which other peripherals would you expect not to have EMIO options? Why?***
  - The 3 Flash devices. All other peripherals can be mapped to EMIO. The Flash cannot be mapped to EMIO because we depend on the Flash to be the boot device.
- ***What is the Power pin for on the SD peripheral? (Hint: use the TRM)***
  - This is a control pin output to enable/disable power to the SD Card slot.
- ***12 peripherals can be mapped to PS Pmod. Find one that cannot?***
  - SP0 and Trace (clk, ctl, and d[3:0]; d[15:4] would have to go to EMIO). These peripherals can be mapped to the PS Pmod: Dual Quad, SPI, SD 1, UART 0, I2C 0, I2C 1, SPI 1, CAN 0, CAN 1, Watchdog, PJTAG, GPIO.
- ***Is Processor JTAG available on ZedBoard? If so, how do I access it?***
  - Yes, it is available. You can access ARM JTAG in cascaded JTAG mode through the shared JTAG. Independent Processor JTAG is also available through the PS Pmod. To do this will require the following steps:
  - o Set boot_mode[3] to 1 for Independent JTAG before power up.
  - o A small piece of configuration code must execute to configure MIO[10-13] as PJTAG.
  - o Use the PS Pmod to access PJTAG. Avnet will offer a small Pmod with the correct mechanical connections to do this.
- ***Name one scenario where having independent access to PJTAG would be useful.***
  - Hardware/software simultaneous debug. The standard JTAG port is used for ChipScope while the PS Pmod PJTAG is used with an ARM DS-5 or DSTREAM for processor debug.

Questions

- ***What is special about MIO[7] and MIO[8]?***
  - They are output only.
- ***Notice that the FCLK_CLK1 Actual Frequency doesn't match the Requested Frequency. Why is this?***
  - The PLLs have a limited number of M/N ratios to generate the various frequencies. With the ENET set to 1000 Mbps, we have a hard requirement for a 125 MHz output clock. Starting with a 33.3333 MHz reference clock, the PLL gets set to 1 GHz, then divided by 8 to generate 125 MHz. With the PLL set to 1 GHz, you are limited now by available dividers. When 150 MHz is requested, the closest divider is 7. 1 GHz / 7 = 142.857 MHz.
- ***Look again at Figure 20 – Memory Test Results. What is the Base Address for the memory test?***
  - 0x00100000
- ***Why doesn't the DDR start at address 0x0?***
  - This is where OCM resides. You can remap DDR to reside at address 0x0, but this is typically done during a 2nd-stage bootloader. See Section 29.4.1 of the TRM.

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

What is Tcl?

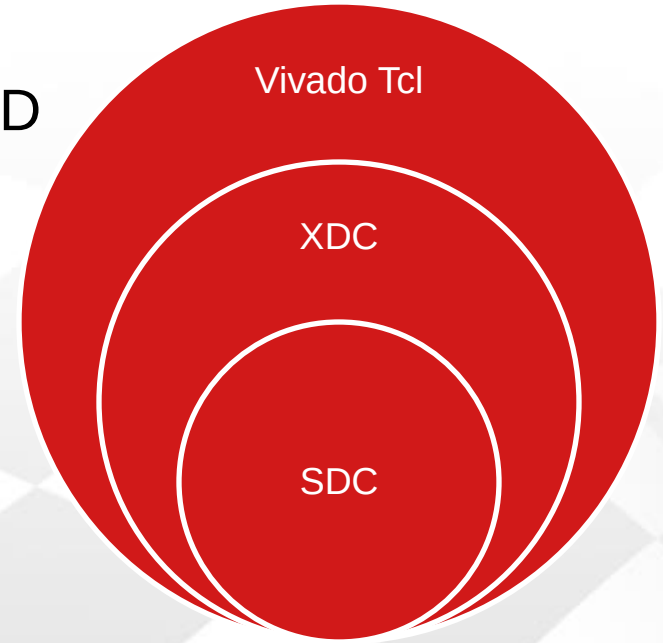
- **History**
  - Invented by John Ousterhout at UC Berkeley in 1988
    - Intended to unify command languages in EDA software
    - Provides generic facilities such as variables, control structures and procedures
    - Embeddable and extensible
  - Grew popular while improved by Sun & Scriptics/Ajula in the 1990s
  - Currently Open Source from ActiveState (www.activestate.com)
- **Adopted by the "big guys"**
  - Synopsys moved from dcsh to dctcl 10 years ago
  - Mentor: ModelSim/Precision/Calibre/Eldo…
  - Cadence: Virtuoso/Encounter…
  - Even the "other" guys use it!!

Introduction to General Tcl

- **Interactive and Interpreted Language**
  - No compilation – send commands to interpreter
  - Everything can be treated as a string
- **Very Simple, or very complex as you need**
  - Loosely typed
  - Full featured language
  - Variables, control, error handling, file IO, etc
- **Main utilization**
  - Rapid prototyping
  - Scripted applications

Vivado Tcl Defined

- **SDC - Synopsys Design Constraints**
- **XDC - Xilinx Design Constraints**
  - Tcl Equivalent to UCF/PCF/XCF/BCD
- **Vivado Tcl**
  - SDC + XDC
  - Xilinx commands
    - Project, compilation, report,…
  - Xilinx objects
    - Netlist, device, timing, project,…
  - General Tcl (8.5)
    - List-related commands are overloaded to support primary objects, unlike Synopsys and the competition
    - Primary objects can directly be used with general commands, unlike Synopsys and the competition



Benefits of XDC and Vivado Tcl –

Why you should care

- **Future constraint language from Synthesis to P&R**
  - Vivado-only for sign-off static timing analysis (STA)
- **Powerful debug and analysis**
  - Fast custom timing reports
  - What-if Analysis with incremental Static Timing Analysis
  - Extendable
- **Industry standard Tool Control**
  - Synplify, Precision, and all ASIC Synthesis/P&R 3rd Party EDA tools use same interface
- **Cross-platform Scripting (Linux and Windows)**
  - Just beware those DOS backslashes! "\"
    - Use "/" or {}

Invoking Vivado via Tcl

- **Vivado IDE startup:**
  - *vivado -source setup.tcl*
  - Type commands directly in Tcl Console
  - Tools → Run Tcl Script
- **Batch Mode – No IDE:**
  - *vivado -mode batch -source my.tcl*
- **Interactive Shell – No IDE:**
  - *vivado -mode tcl*
  - Can start the IDE at any time via **`start_gui`** command

Vivado Command Categories Examples

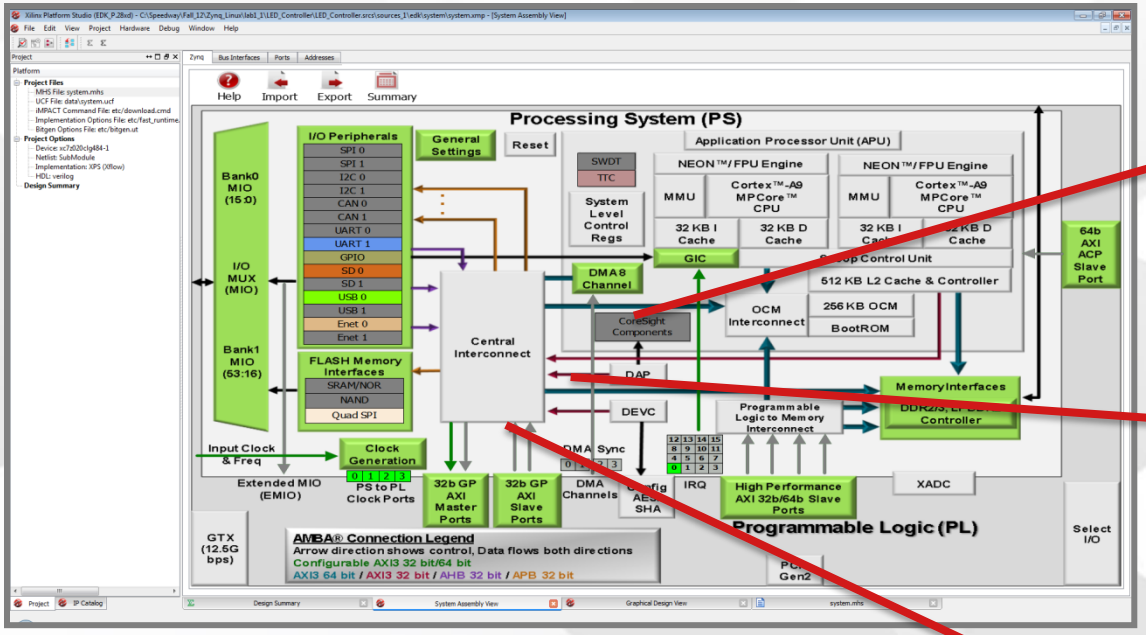
- **Project - Project Management**
  - create_project
  - add_files
  - import_files
  - import_ip
- **Object - Object Access**
  - get_cells
  - get_property
  - set_property
- **Report**
  - report_timing
  - report_power
  - report_drc
- **Project - Runs, Synthesis and Implementation Control**
  - create_run
  - launch_runs
  - wait_on_run
- **FileIO**
  - read_verilog
  - read_xdc
  - read_csv
  - write_verilog
- **GUIControl**
  - start_gui
  - stop_gui

Tcl References

- **Tcl Built-in Command Reference:**
  - http://www.tcl.tk/man/tcl8.5/TclCmd/contents.htm
- **General Tcl Tutorial**
  - http://www.tcl.tk/man/tcl/tutorial/tcltutorial.htm
- **Books (Search on Amazon.com):**
  - Brent Welch:
    - http://www.amazon.com/Practical-Programming-Tcl-Tk-4th/dp/0130385603/ref=sr_1_2?ie=UTF8&s=books&qid=1271867857&sr=8-2
  - John Ousterhout:
    - http://www.amazon.com/Tcl-Toolkit-2nd-John-Ousterhout/dp/032133633X/ref=sr_1_1?ie=UTF8&s=books&qid=1271867857&sr=8-1

Files Created by Vivado Block Design

- **Block design saved as a .bd file**
- **Everything created in a block can be recreated by a TCL script…**
  - More later!



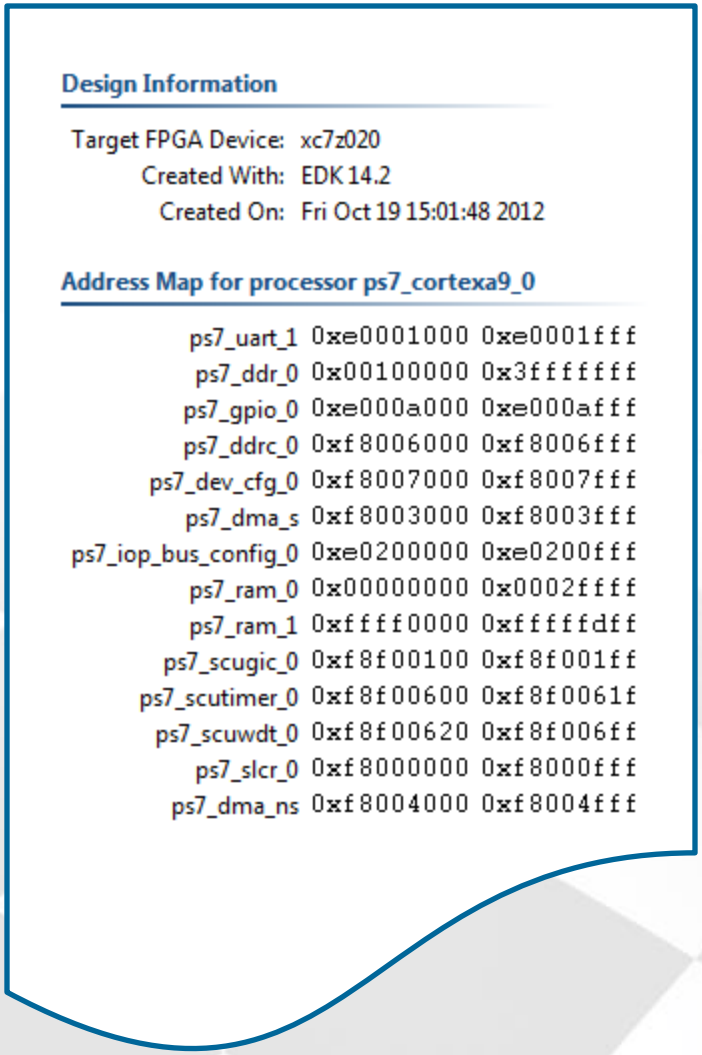
.BD

XML

PS7_INIT

XML File

- **Hardware platform description for FSBL and BSP generation**
- **Automatically created by Vivado, Exported to SDK**
- **Text File, but do not edit, use IP Integrator, or TCL commands**
- **Contains**
  - Processor instantiation
  - Peripheral instances and addresses



Design Information

Target FPGA Device: xc7z020
Created With: EDK 14.2
Created On: Fri Oct 19 15:01:48 2012

Address Map for processor ps7_cortexa9_0

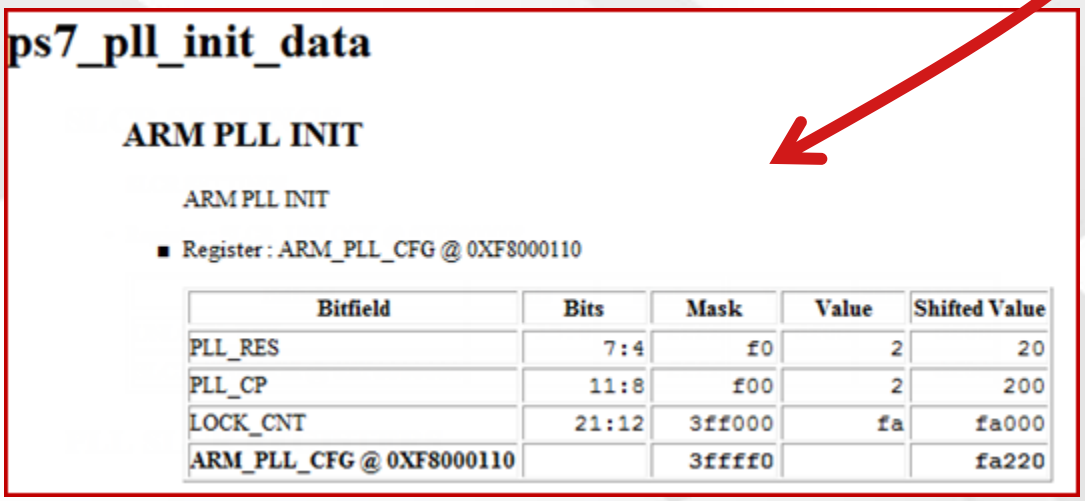
ps7_uart_1	0xe0001000	0xe0001fff
ps7_ddr_0	0x00100000	0x3fffffff
ps7_gpio_0	0xe000a000	0xe000afff
ps7_ddrc_0	0xf8006000	0xf8006fff
ps7_dev_cfg_0	0xf8007000	0xf8007fff
ps7_dma_s	0xf8003000	0xf8003fff
ps7_iop_bus_config_0	0xe0200000	0xe0200fff
ps7_ram_0	0x00000000	0x0002ffff
ps7_ram_1	0xffff0000	0xffffdfff
ps7_scugic_0	0xf8f00100	0xf8f001ff
ps7_scutimer_0	0xf8f00600	0xf8f0061f
ps7_scuwdt_0	0xf8f00620	0xf8f006ff
ps7_slcr_0	0xf8000000	0xf8000fff
ps7_dma_ns	0xf8004000	0xf8004fff

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Processor Initialization – PS7_Init

- **PS7_Init file describes processor configuration**
  - DDR Controller Specifications
  - PLL Configuration
  - JTAG Modes
  - Zynq Peripheral Configuration
- **C, TCL and HTML files created**



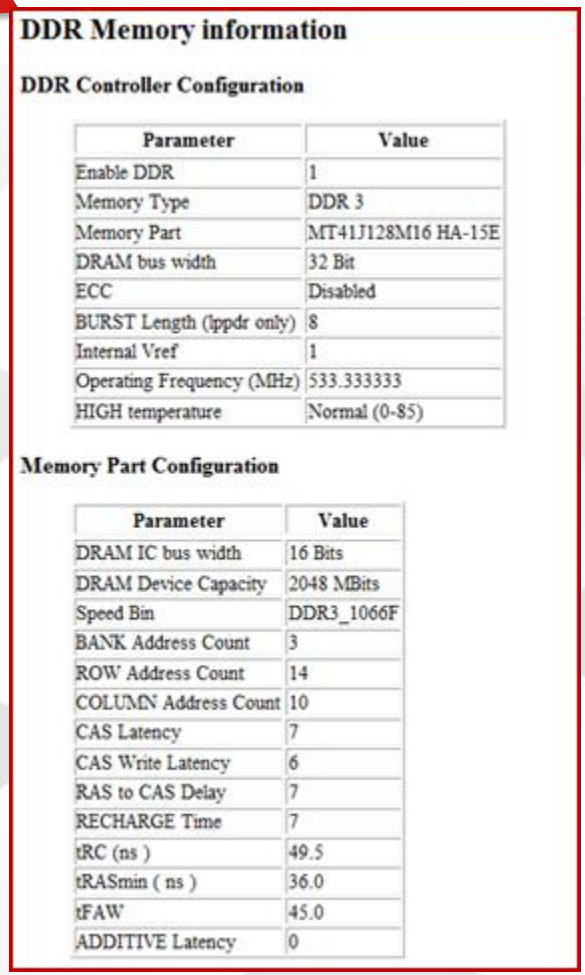
ps7_pll_init_data

ARM PLL INIT

ARM PLL INIT

- Register : ARM_PLL_CFG @ 0XF8000110

Bitfield	Bits	Mask	Value	Shifted Value
PLL_RES	7:4	f0	2	20
PLL_CP	11:8	f00	2	200
LOCK_CNT	21:12	3ff000	fa	fa000
ARM_PLL_CFG @ 0XF8000110		3ffff0		fa220



DDR Memory information

DDR Controller Configuration

Parameter	Value
Enable DDR	1
Memory Type	DDR 3
Memory Part	MT41J128M16 HA-15E
DRAM bus width	32 Bit
ECC	Disabled
BURST Length (lppdr only)	8
Internal Vref	1
Operating Frequency (MHz)	533.333333
HIGH temperature	Normal (0-85)

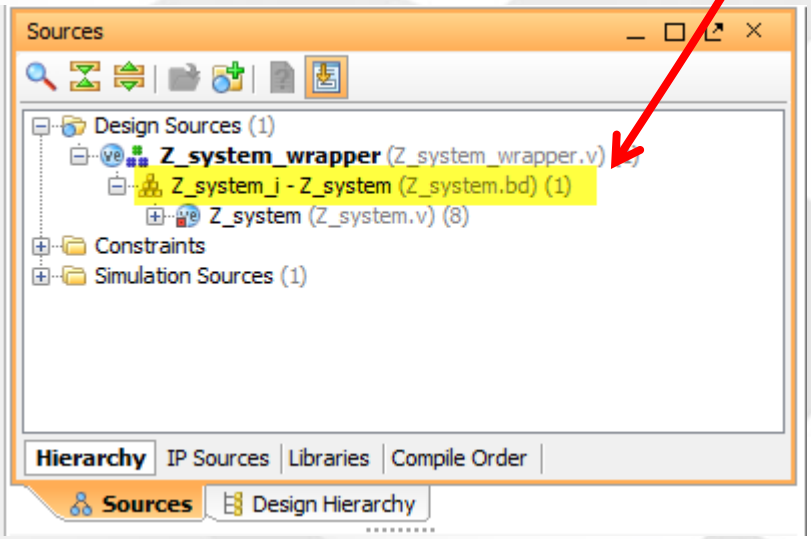
Memory Part Configuration

Parameter	Value
DRAM IC bus width	16 Bits
DRAM Device Capacity	2048 MBits
Speed Bin	DDR3_1066F
BANK Address Count	3
ROW Address Count	14
COLUMN Address Count	10
CAS Latency	7
CAS Write Latency	6
RAS to CAS Delay	7
RECHARGE Time	7
tRC (ns)	49.5
tRASmin (ns)	36.0
tFAW	45.0
ADDITIVE Latency	0

AVNET electronics marketing

BD File

- **Exists in Vivado**
- **The BD file is the block design file for IP Integrator**
- **Takes on the name of the project, *<project>.bd***
  - In the labs, Z_system.bd is used
- **Contains and controls**
  - Details that make up the block design
    - XML format
  - Previously, MHS file
  - DON'T EDIT outside Vivado



Checkpoint!

- **What tool is used to create and define the Zynq Processors? Files?**

IP Integrator

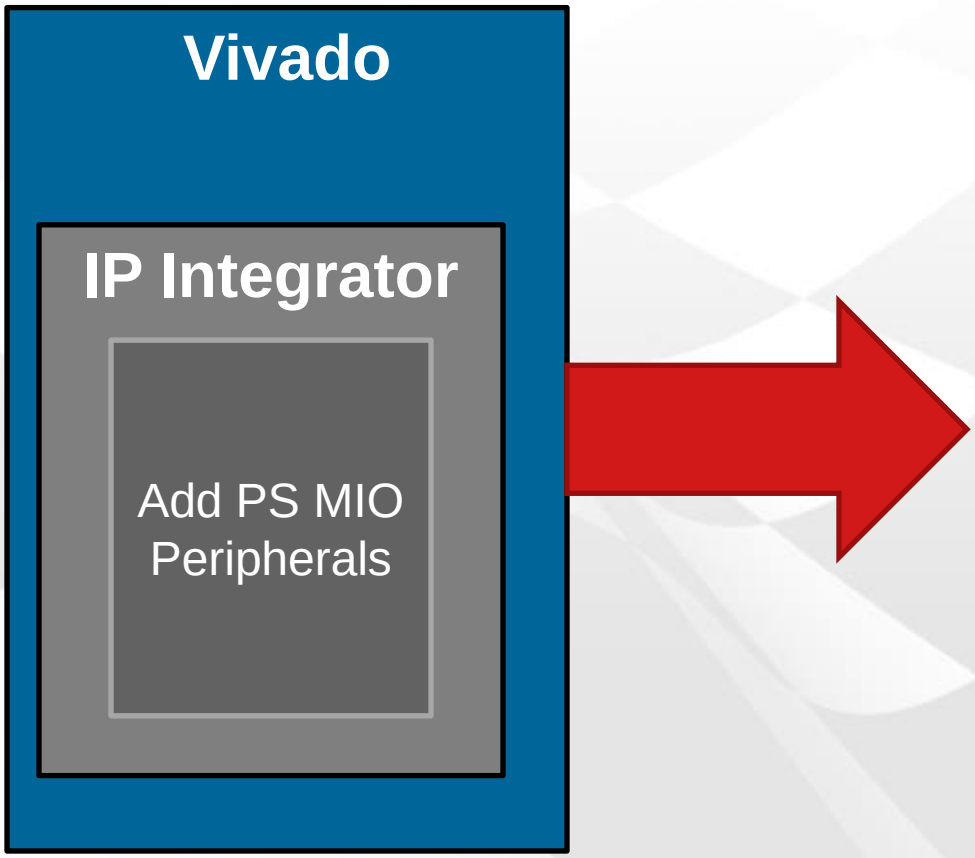
PS7_Init, XML, BD

- **What tool is used to create Zynq Software?**

SDK – Software

Lab 4 – TCL Scripting

- **Create TCL scripts that rebuild project**



TCL Scripts

```
# Set the original project directo
set orig_proj_dir "C:/Speedway/Zyn

# Create project
create_project ZynqDesign ./ZynqDe

# Set the directory path for the n
set proj_dir [get_property directo

# Set
set ob
set_pr
set_pr
```

```
# Create interface ports
set DDR [ create_bd_intf_port -mo
set FIXED_IO [ create_bd_intf_por

# Create ports

# Create instance: processing_sys
set processing_system7_0 [ create
set_property -dict [ list CONFIG.
CONFIG.PCW_EN_CLK0_PORT {0} CONFI
} CONFIG.PCW_QSPI_PERIPHERAL_ENAB
CONFIG.PCW_SD0_GRP_WP_IO {MIO 46}
 {1} CONFIG.PCW_UIPARAM_DDR_BOARD
CONFIG.PCW_UIPARAM_DDR_BOARD_DELA
CONFIG.PCW_UIPARAM_DDR_DQS_TO_CLK
HA-15E} CONFIG.PCW_UIPARAM_DDR_TR
CONFIG.PCW_UIPARAM_DDR_USE_INTERN
```

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Questions

- ***What is the project name?***
  - # CHANGE DESIGN NAME HERE
    set design_name Z_system
- ***How many interface ports do you see? What are they?***
  - 2 – DDR & FIXED_IO
- ***What is the MIO Bank1 Voltage set to?***
  - CONFIG.PCW_PRESET_BANK1_VOLTAGE {LVCMOS 1.8V}
- ***In what directory will the project be created?***
  - This is a tough question. The first TCL command in this script sets the original project location, but that is only used to import source files. The second TCL command creates the project and does it in whatever directory the TCL script was sourced from. So before running this script, make sure you are in the directory you want the project created.
- ***What part is selected?***
  - set_property "part" "xc7z010clg400-1" $obj
- ***Does the TCL file load the block design?***
  - Yes it does:
    # Import local files from the original project
    set files [list \
    "C:/Speedway/ZynqHW/2013_3/ZynqDesign/ZynqDesign.srcs/sources_1/bd/Z_system/Z_system.bd"\
  - But it gets this source from our original project directory. So if run on another PC, the project will not get recreated correctly.
- ***Does the TCL file reload the synthesis and implementation settings?***
  - Yes, see Create 'synth_1' run and Create 'impl_1' run

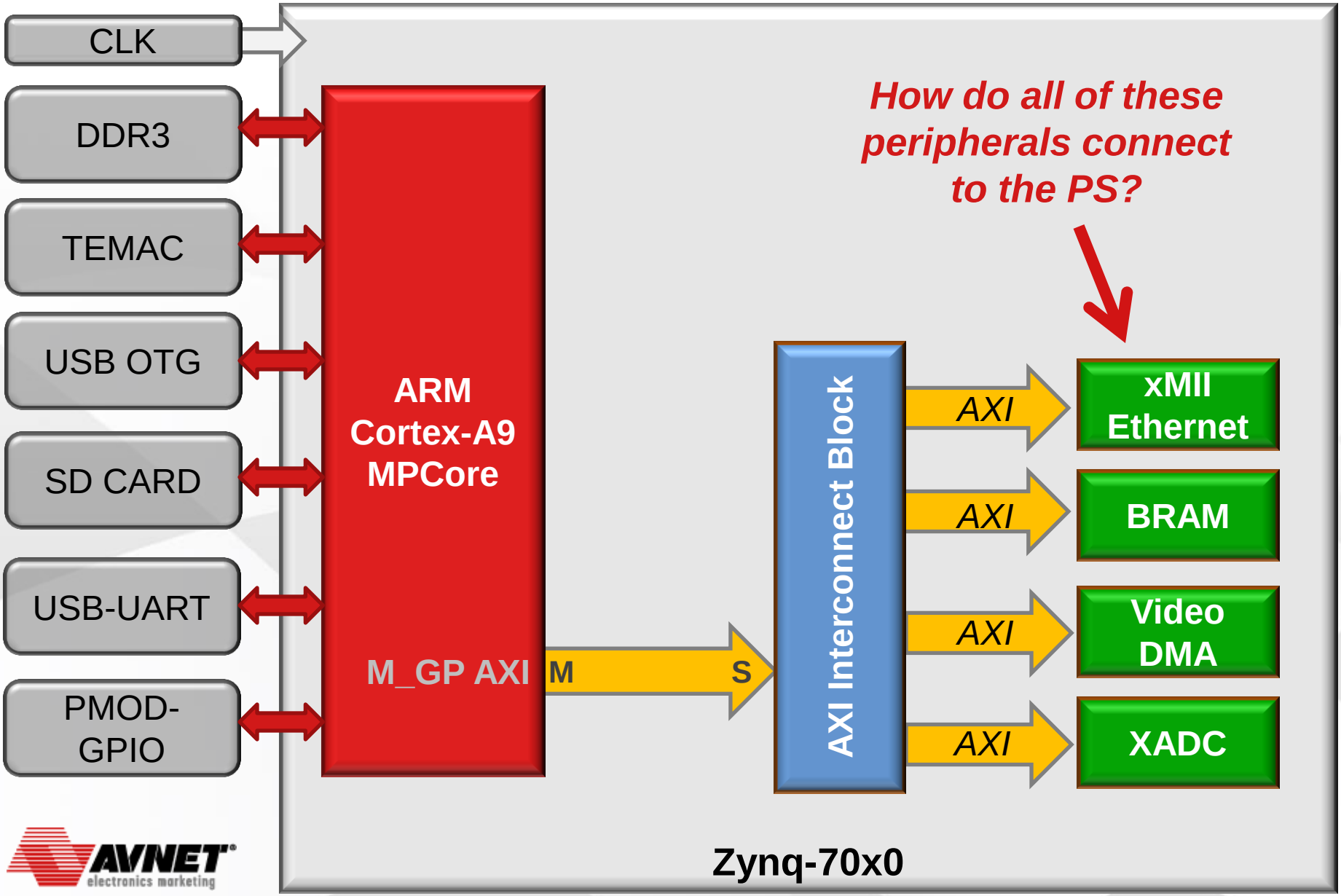
Questions

- ***In the project_setup.tcl script, what files does the script need to recreate the project?***
  - ZynqDesign/ZynqDesign.srcs/sources_1/bd/Z_system/Z_system.bd"\
    ZynqDesign/ZynqDesign.srcs/sources_1/bd/Z_system/hdl/Z_system_wrapper.v"\
- ***Does the Zynq IP block look the same? Are all I/O peripherals mapped the same?***
  - Yes, it is the same
- ***Could both of the TCL files used to recreate this project be combined into one TCL file?***
  - Yes, they could simply be combined, but the project_setup.tcl would need to remove importing the block design source. Otherwise when the basic_design.tcl is run, it will already see a block design with the same name and quit running.
- ***What source files were needed to recreate this entire project? In other words, what do you need to archive for revision control at this point?***
  - None, only the two TCL scripts are required. Per the above question, one script could be created that performs the entire project creation, block design creation, and Zynq IP Customization. That's quite powerful. Thus only the TCL file would need to be checked into a revision control system.

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

Adding Soft, PL IP Peripherals

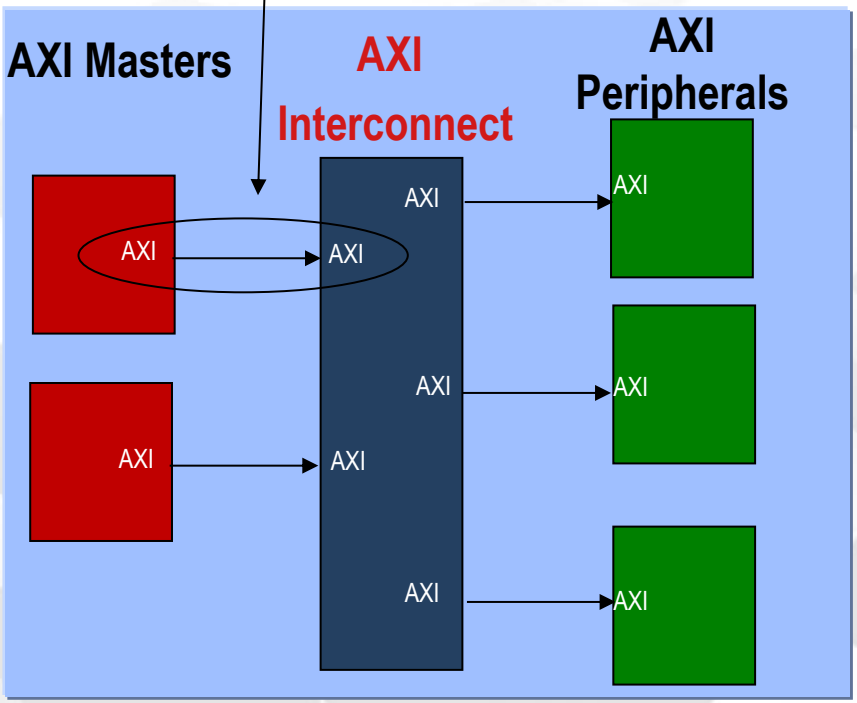


AXI Interconnect Block

Connects Multiple Master / Slave Pairs

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

AXI defines a point to point, master/slave interface



Plug and Play IP

Achieved via adoption of industry standards

- **Now Easier to Access, Integrate and Protect IP**
- **IP Packager available**

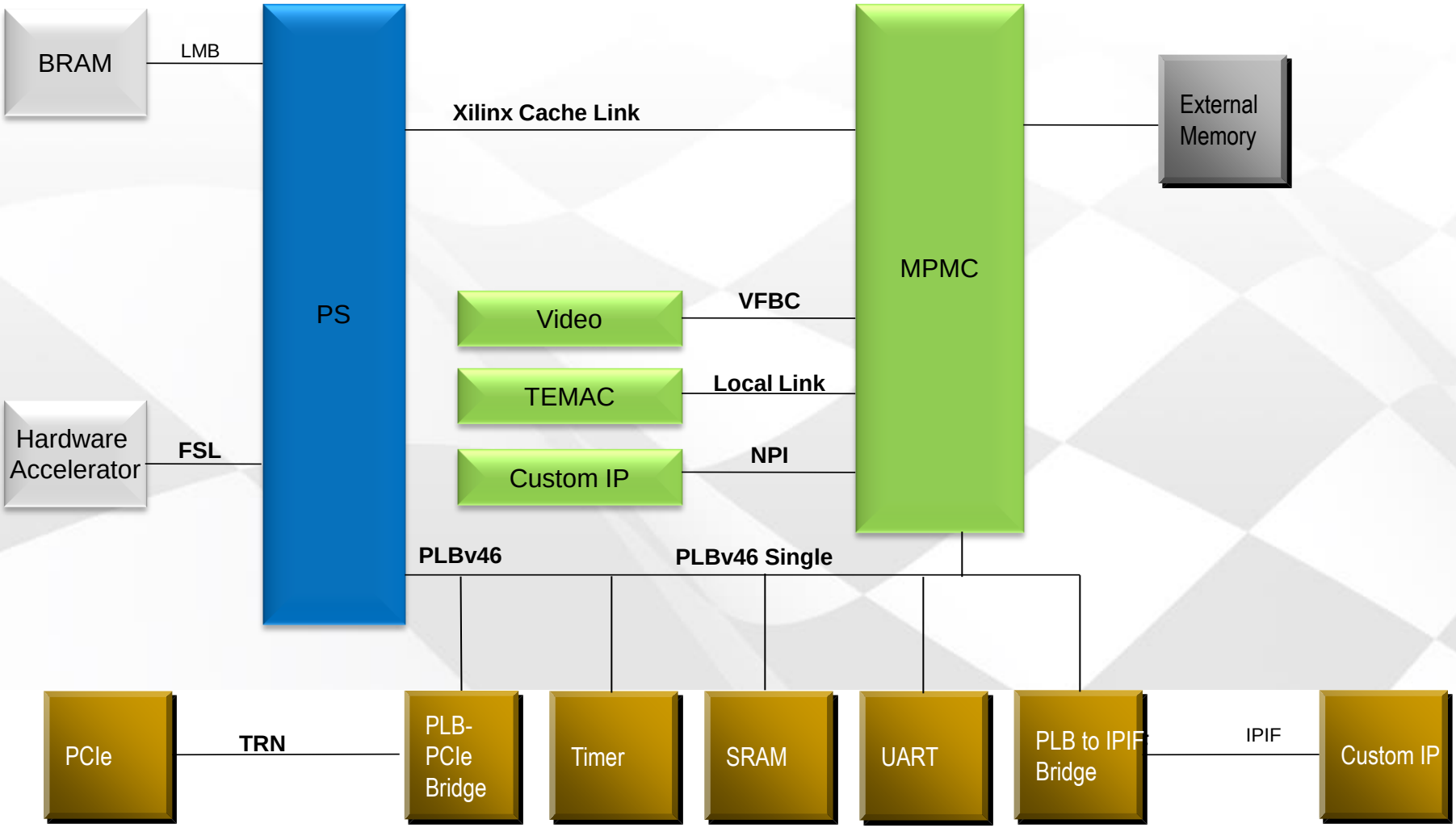
AMBA AXI4	IP-XACT
Standard Interconnect	Standard Packaging
• IP Verification with AXI4 BFM within Vivado • Requires license • Up to 20% higher system bandwidth and up to 50% better area for the interconnect block	• IP Repository with common “look & feel” for Xilinx and partner IP

AVNET electronics marketing

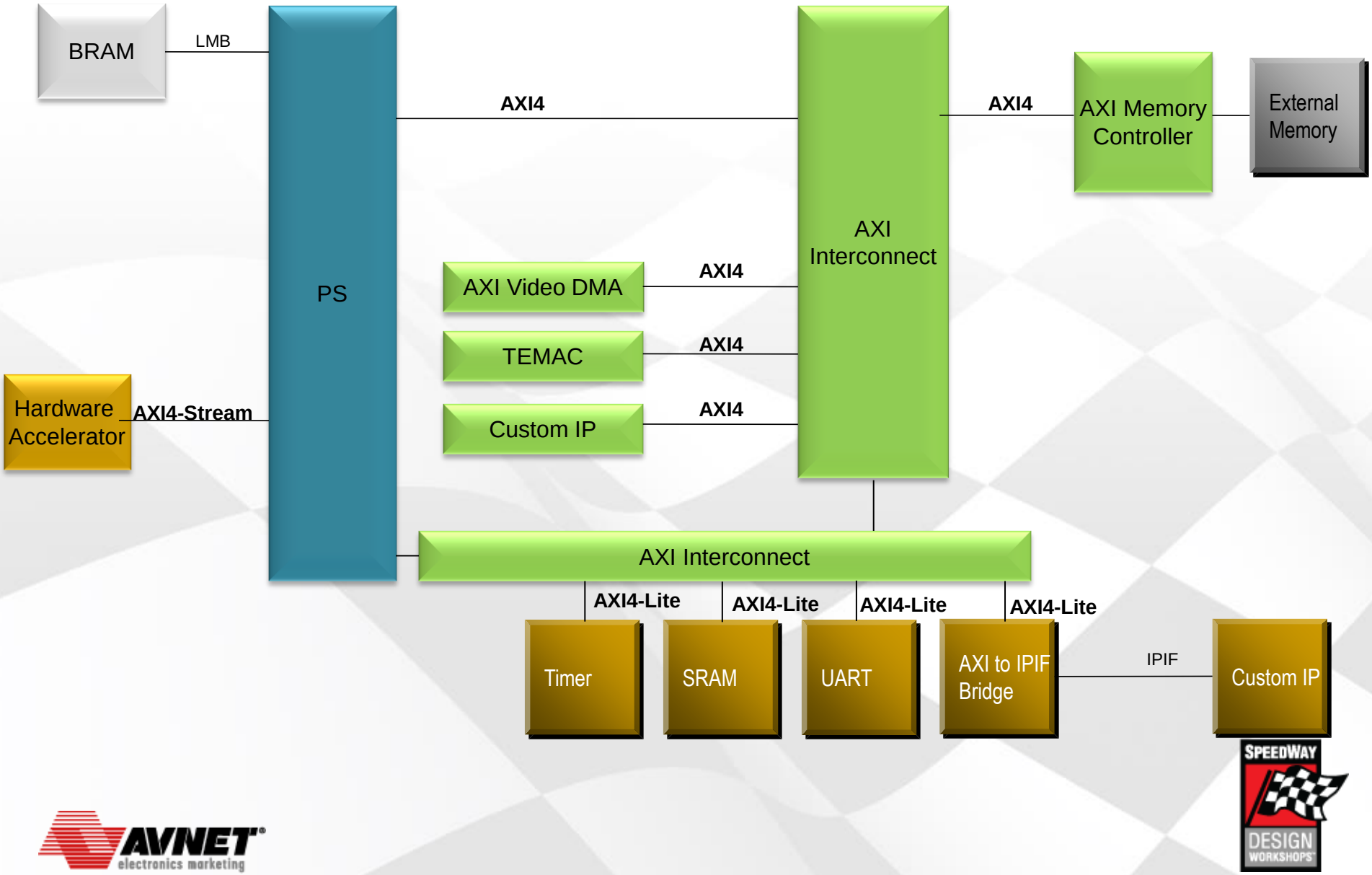
SPEEDWAY DESIGN WORKSHOPS

Why are IP Standards So Important?

- **Do you remember this?**



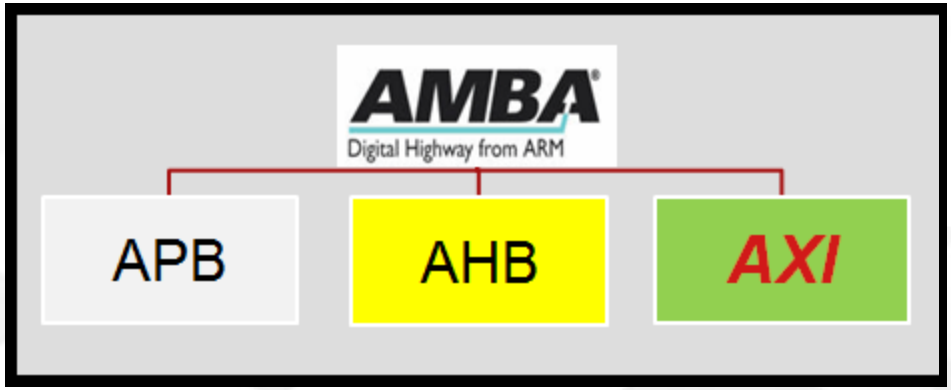
Now -Standardize on AXI4



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



Advanced eXtensible Interface

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

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

AXI4 - Advanced Extensible Interface

Standard Overview

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

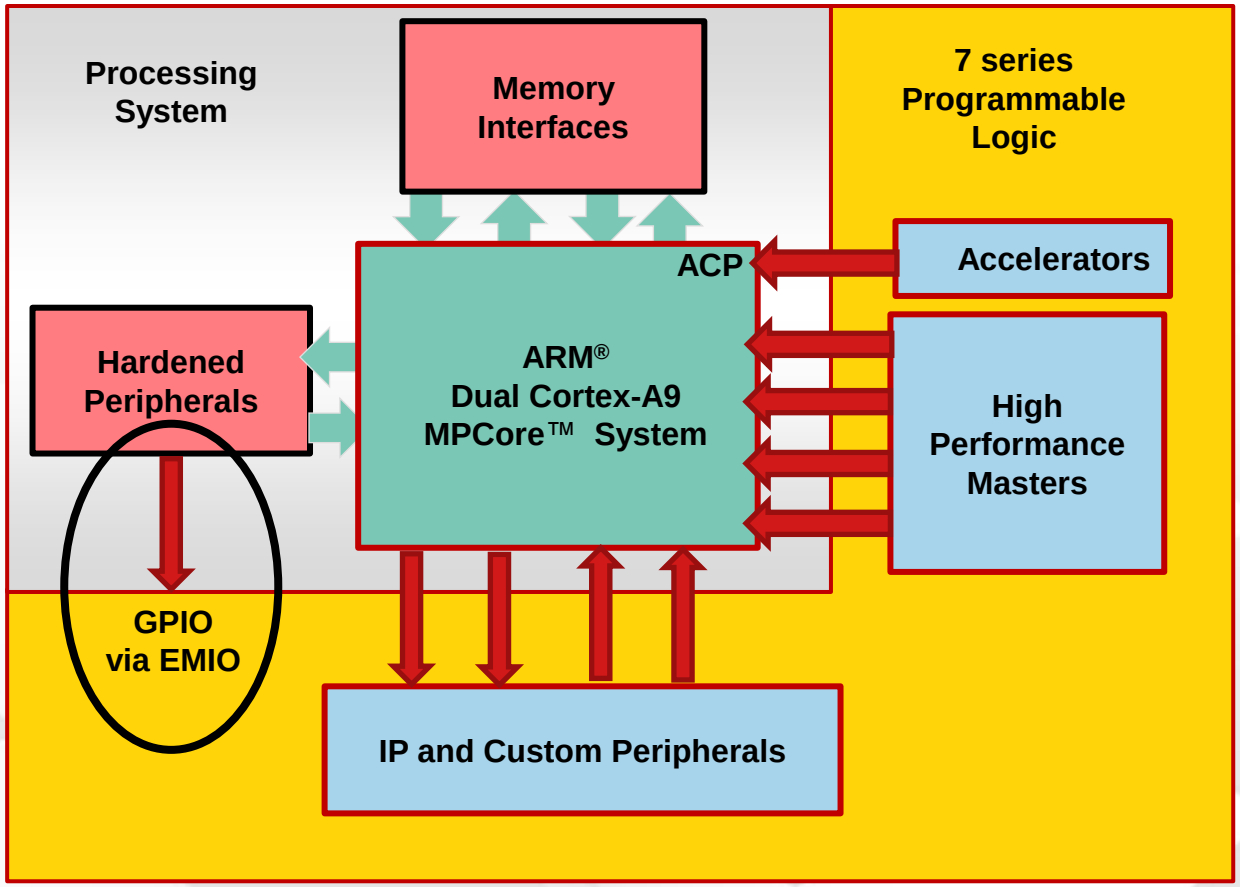
	AXI4	AXI4-Lite	AXI4-Stream
Dedicated for	high-performance and memory mapped systems	register-style interfaces (area efficient implementation)	non-address based IP
Burst** (data beat)	up to 256	1	unlimited
Data Width	32 to 1024 bits	32 or 64 bits	any number of bytes
Applications (examples)	Embedded, memory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

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

AVNET electronics marketing

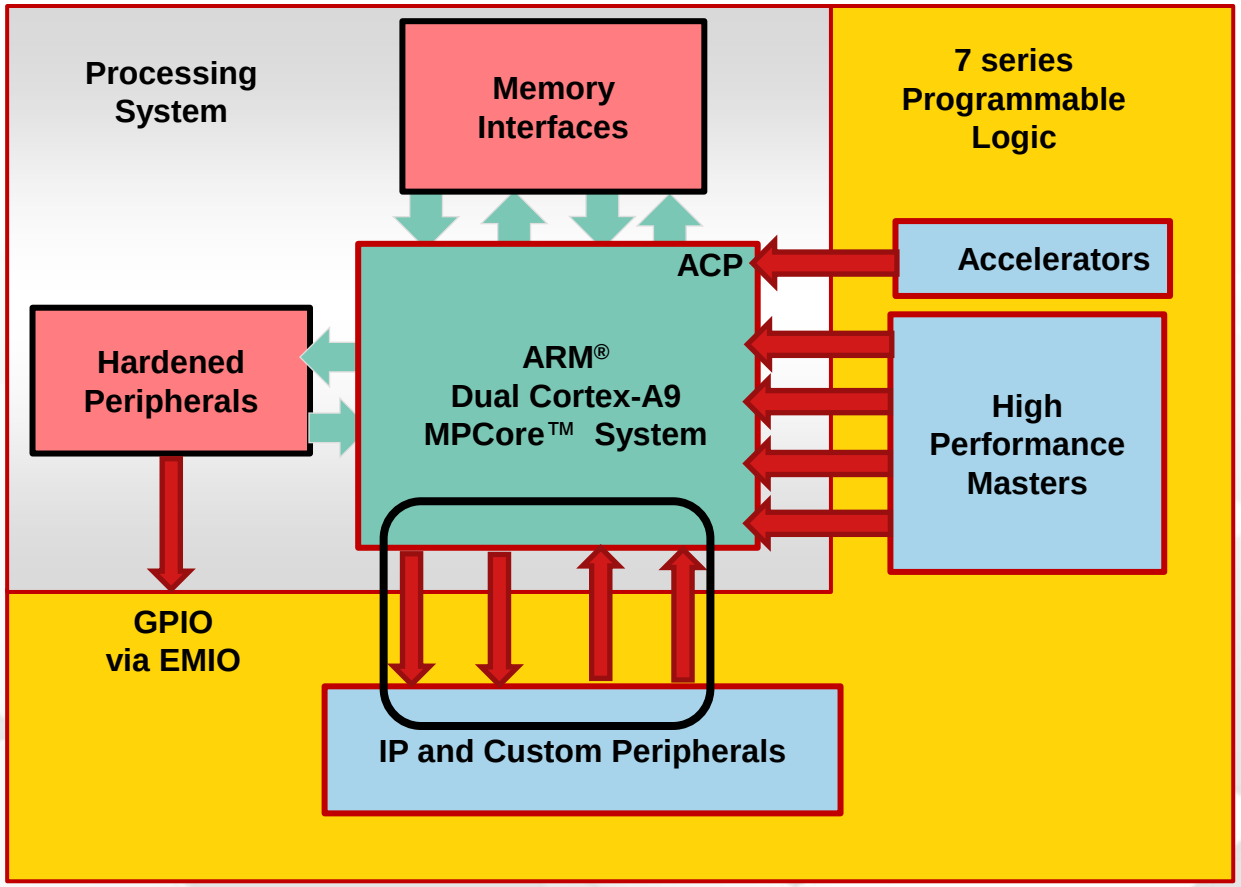
SPEEDWAY DESIGN WORKSHOPS

PL Interface Bandwidth - GPIO



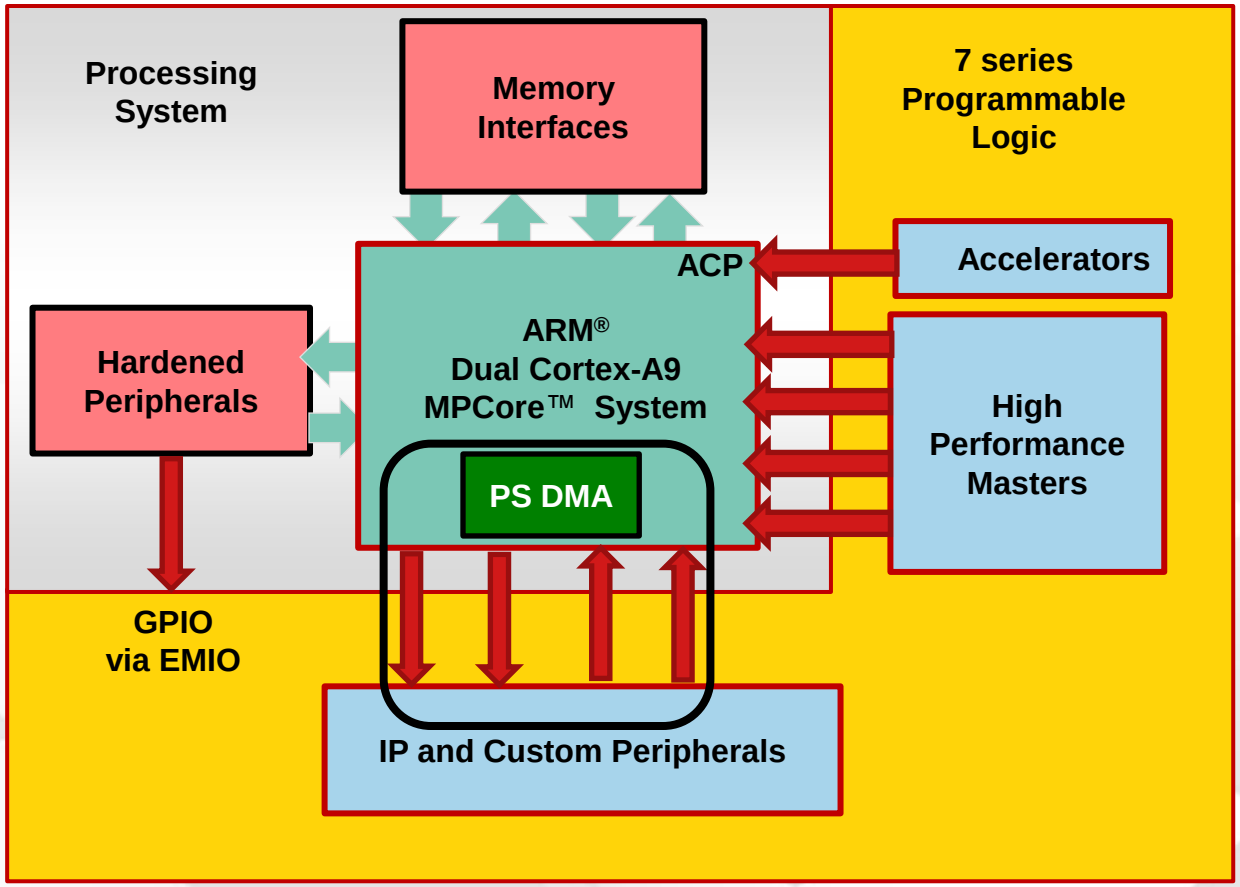
Method	Benefits	Drawbacks	Suggested Uses	Estimated Throughput
CPU Programmed I/O	Simple Software, Least PL Resources, Simple PL Slaves	Lowest Throughput	Control Functions	**<25 MB/s**

PL Interface Bandwidth – PL General Purpose AXI



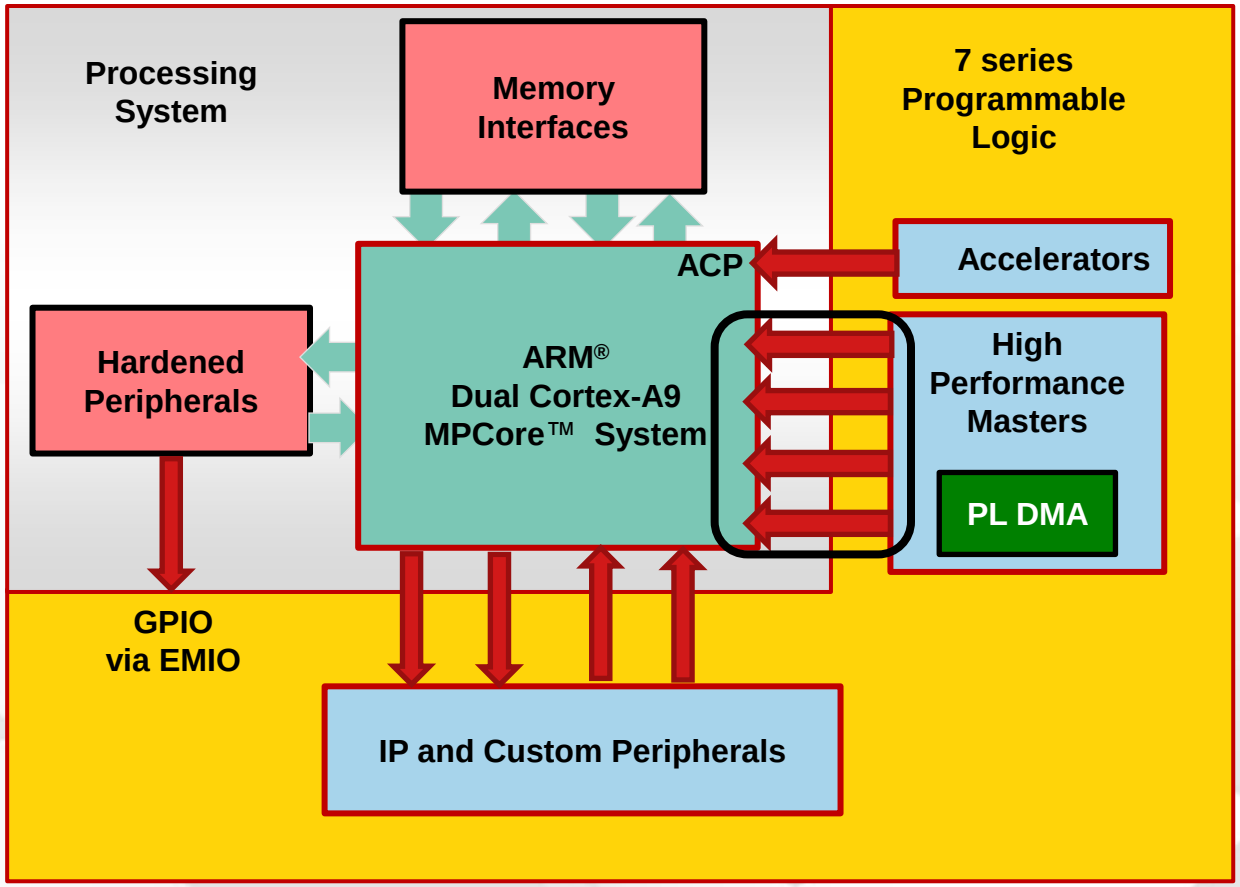
Method	Benefits	Drawbacks	Suggested Uses	Estimated Throughput
PL AXI_GP	Medium Throughput	More complex PL Master design	PL to PS Control Functions, PS I/O Peripheral Access	**600 MB/s (per interface)**

PL Interface Bandwidth – GP AXI Utilizing PS DMAC



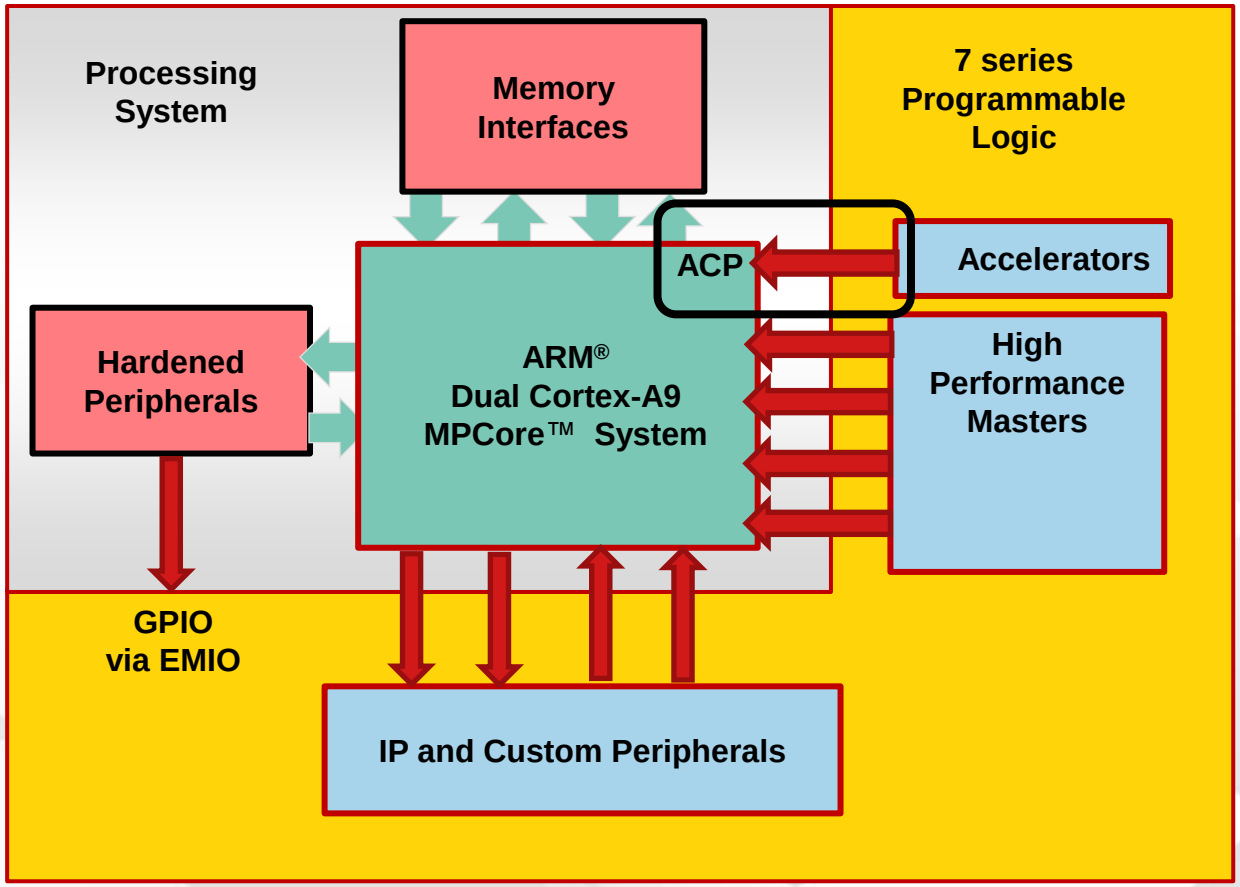
Method	Benefits	Drawbacks	Suggested Uses	Estimated Throughput
PS DMAC	Same as before but, DMAC handles traffic, not Processor!	Somewhat complex DMA programming	Limited PL Resource DMAs	**600 MB/s (per interface)**

PL Interface Bandwidth – High Performance w/ DMA



Method	Benefits	Drawbacks	Suggested Uses	Estimated Throughput
PL AXI_HP DMA	Multiple Interfaces, Command/Data FIFOs	OCM/DDR access only More complex PL Master design	High Performance DMA for large datasets	**1,200 MB/s (per interface)**

PL Interface Bandwidth – ACP w/ DMA



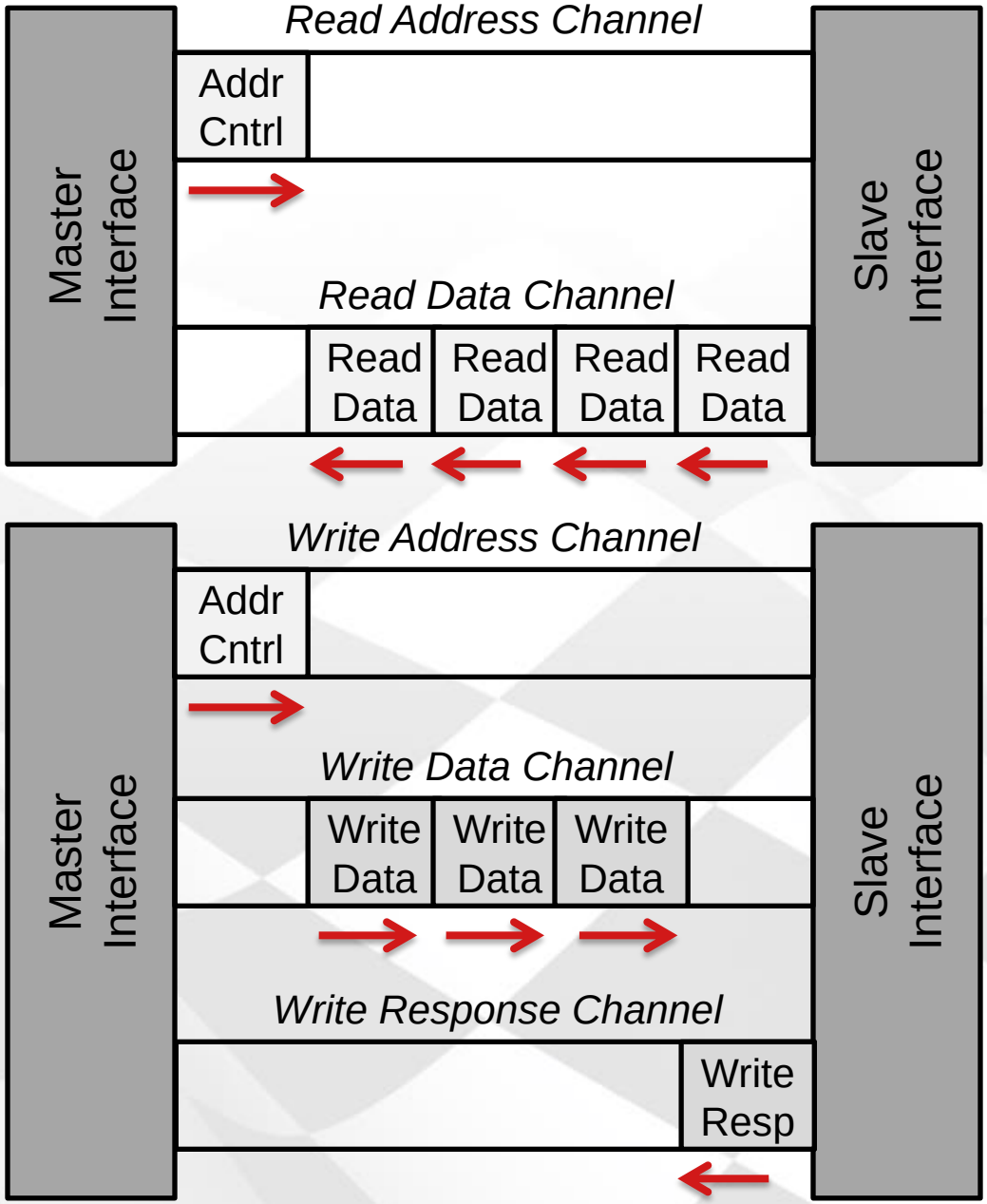
Method	Benefits	Drawbacks	Suggested Uses	Estimated Throughput
PL AXI_ACP	Lowest Latency, Optional Cache Coherency	Large burst may = cache thrashing, Shares CPU Interconnect BW, More complex PL	High Performance for smaller, coherent datasets, Medium granularity CPU Offload	**1,200 MB/s**

GP AXI Signaling

Five Channels

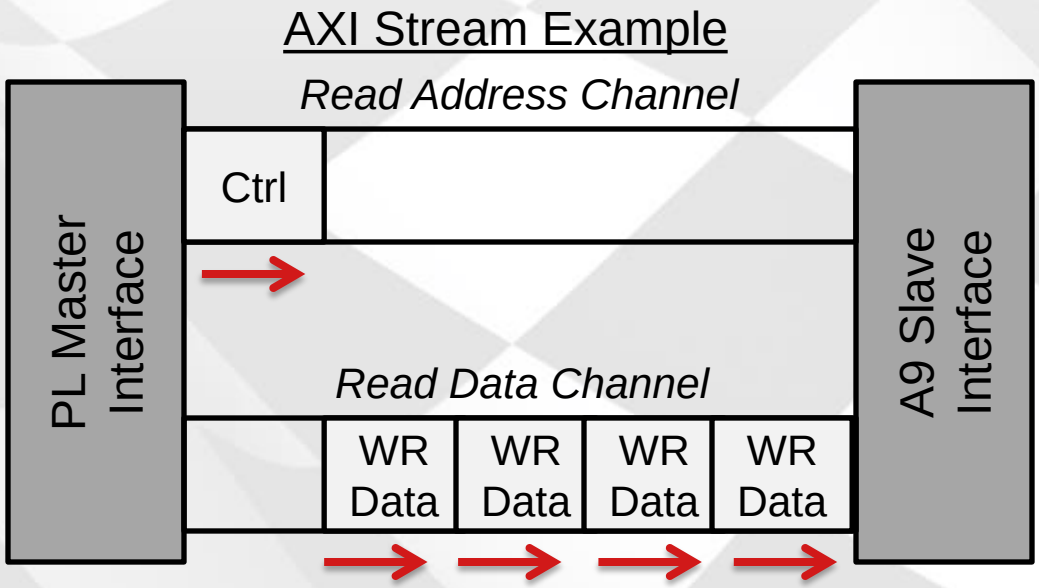
1. **Read Address Channel**
2. **Read Data Channel**
3. **Write Address Channel**
4. **Write Data Channel**
5. **Write Response Channel**
   - Non-Posted write model: there will always be a “Write Response”

AXI-4 (Full) Example



AXI High-Performance Interface

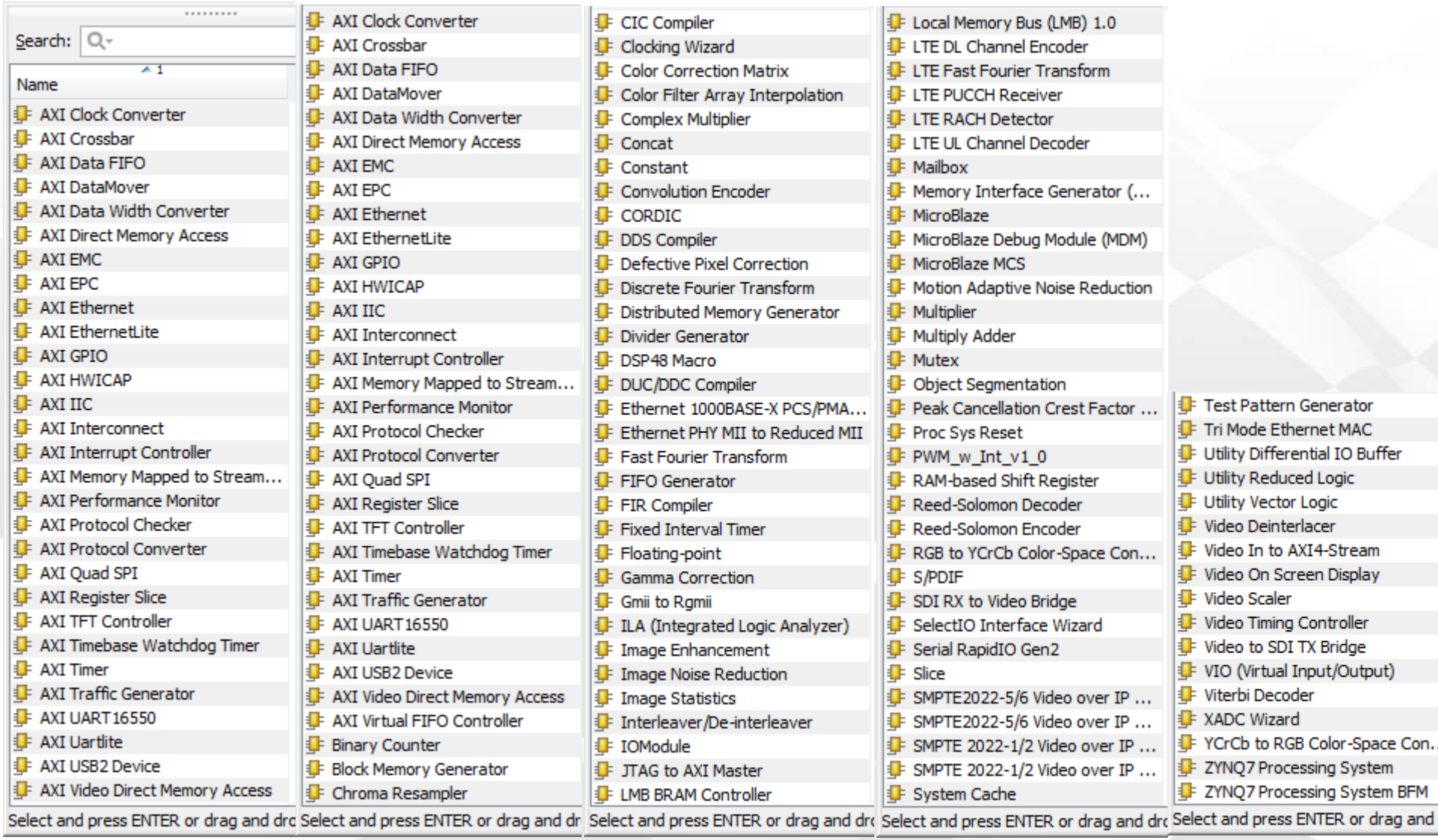
- **High-bandwidth interfaces from PL Master to OCM and DDR memories**
  - AXI_HP provide the greatest aggregate interface bandwidth @ 1,200 MB/s per interface
- **Four, 64-bit wide interfaces (-1 = 150MHz)**
- **Saves PL resources by reducing the need to a PL AXI interconnect**
  - Contains control and 128x64b data FIFOs
- **Ideal for workloads such as video frame buffering in DDR**



AVNET electronics marketing

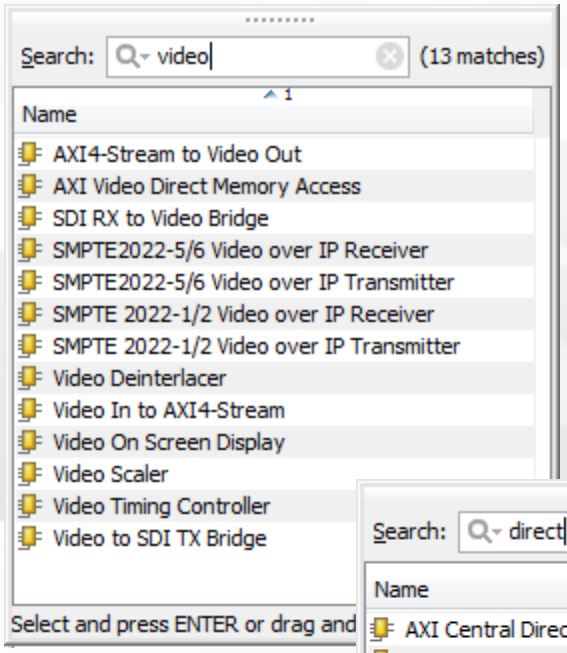
What IP Core's are Available?

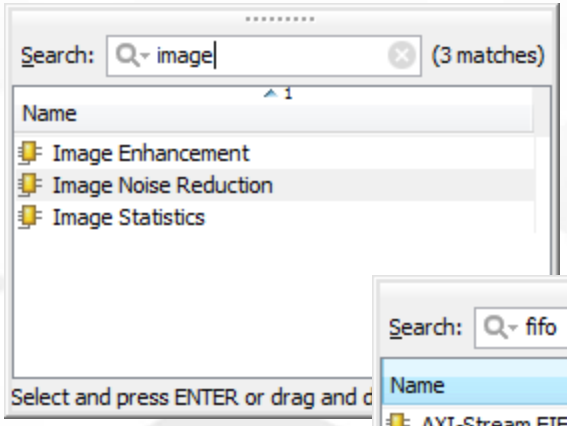
- **Over 150 cores available in the IP Catalog!**

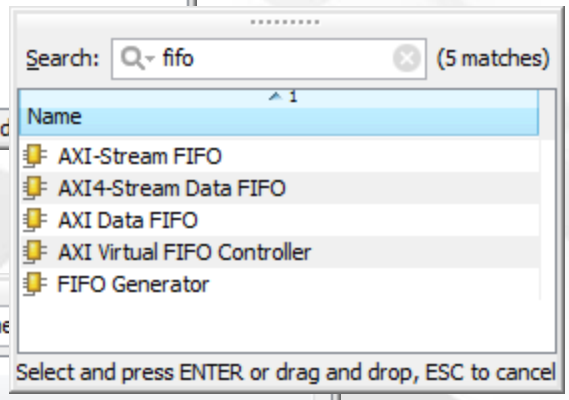


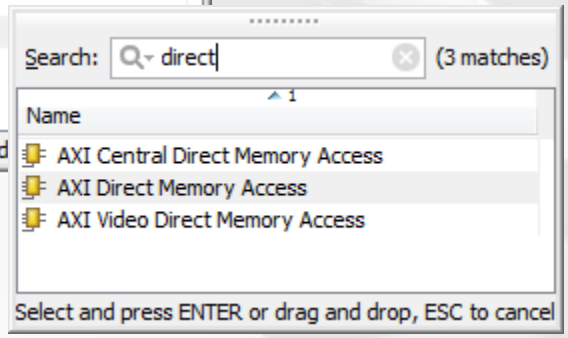
That's a lot of IP, How Do I Find it?

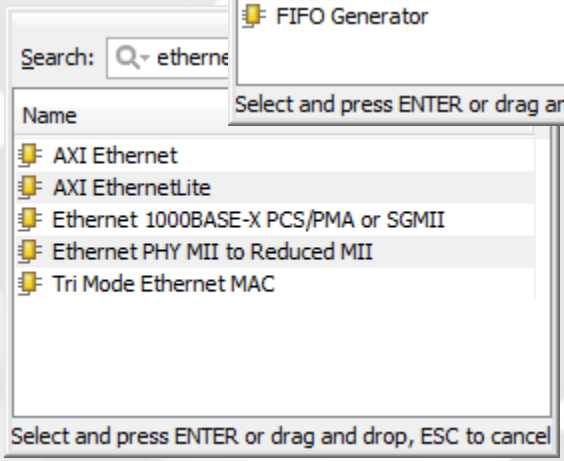
- **Search IP Catalog to refine search**
  - Can't find what you're looking for?
  - Use search function









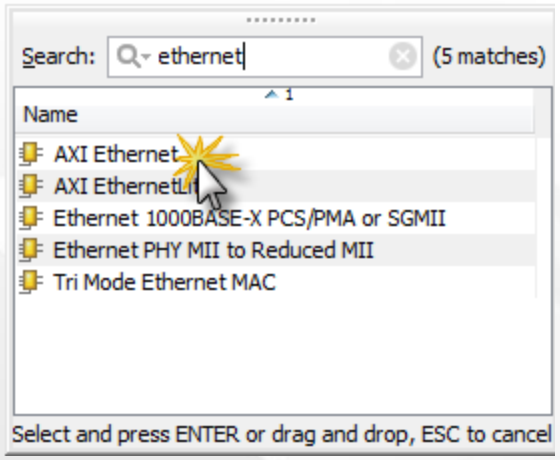


AVNET electronics marketing

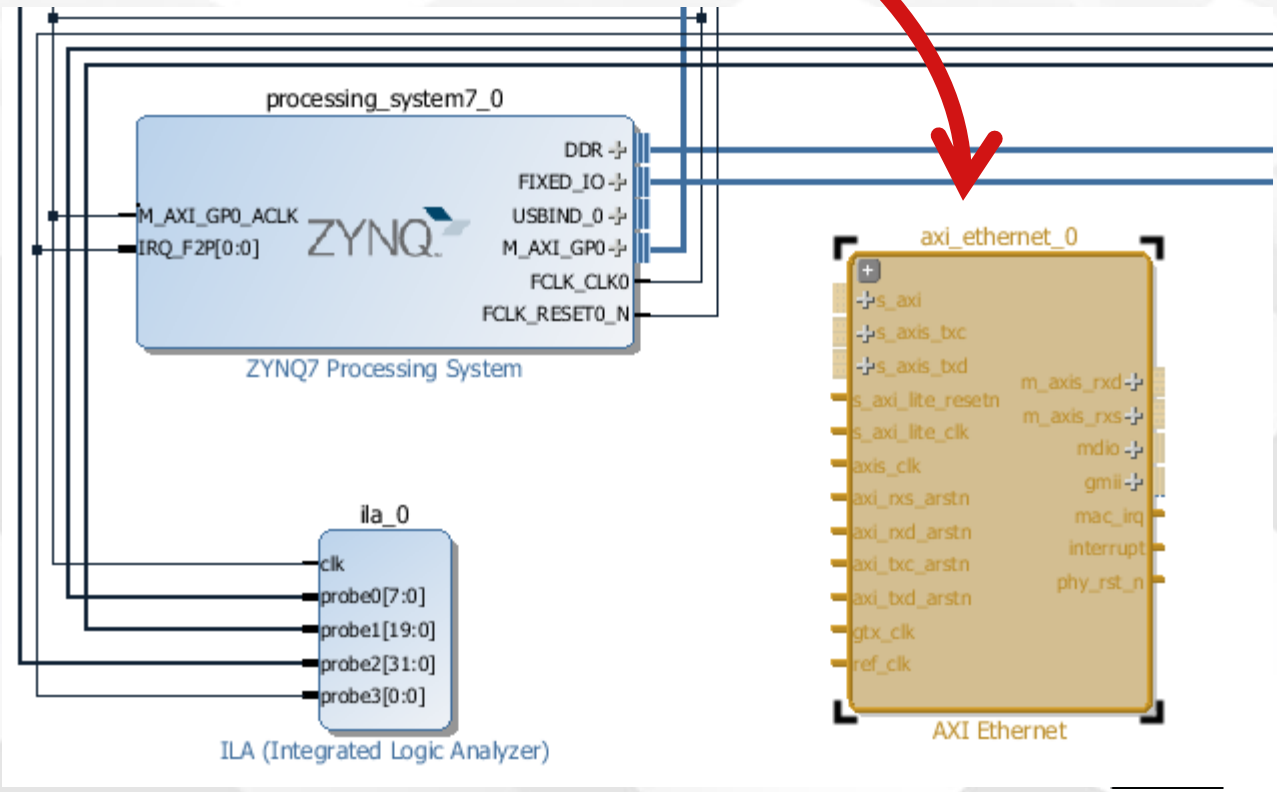
SPEEDWAY DESIGN WORKSHOPS

Adding PL IP to the Design

1 Double-Click IP



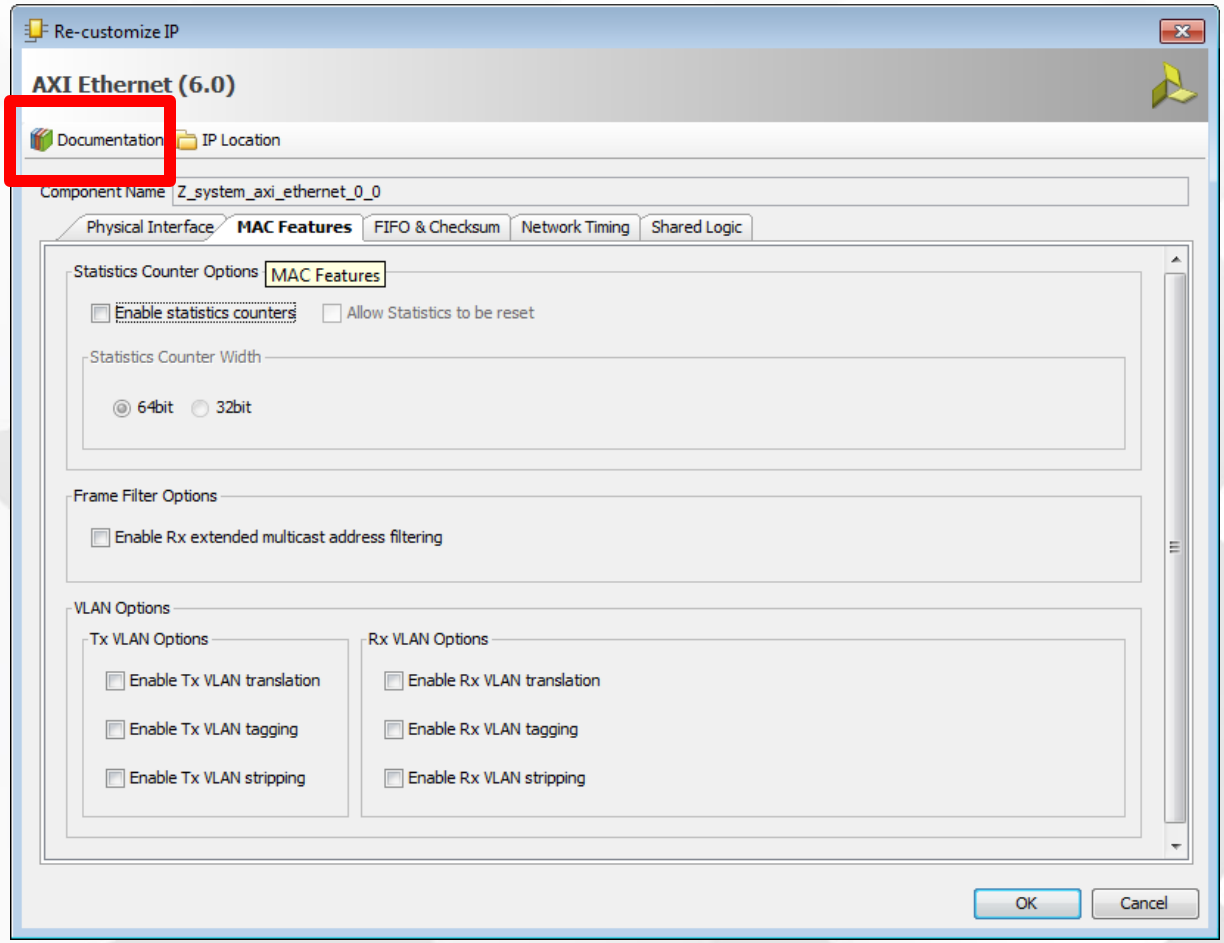
2 IP Appears in Block Design



AVNET electronics marketing

Parameterize IP Instances

- **Double-click or right-click the instance and select Configure Block to open the configurable parameters**
  - Default values are shown



Core Sizes

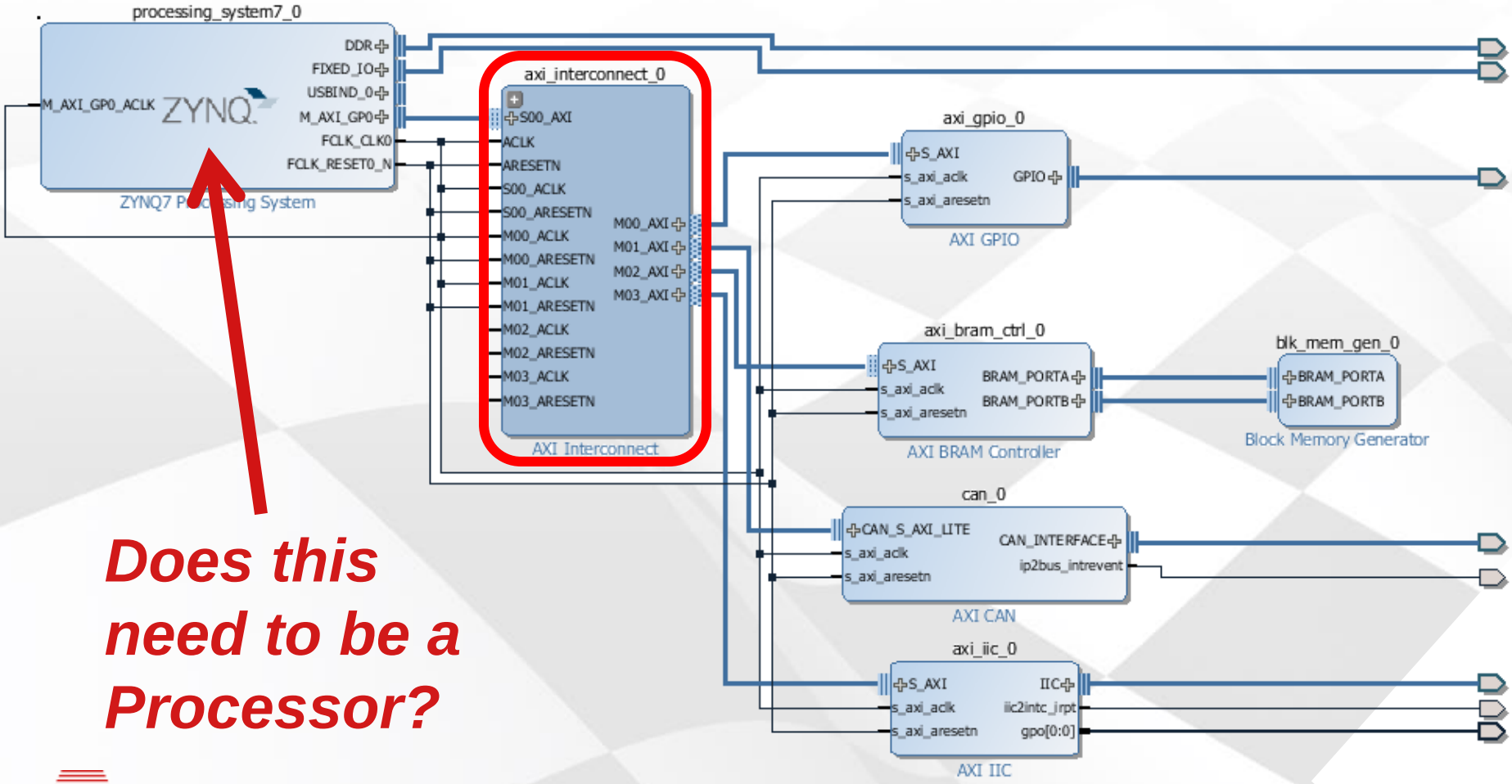
- **The size of each core is available in the data sheet**
  - For example, the LogiCORE IP AXI Ethernet data sheet contains the following table

Kintex-7 FPGA Performance and Resource Utilization Benchmarks

Parameter Values									Device Resources							
C_(T,R)XMEM	C_(T,R)XCSUM	C_TXVLAN_*	C_RXVLAN_*	C_MCAST_EXTEND	C_STATS	C_AVB	C_TYPE	C_PHY_TYPE	Slices	Flip- Flops	block RAMs (RAMB36E1/ FIFO36E1)	LUTs	BUFGs	BUFIOs	BUFRs	BUFGP
32768	0	0	1	1	1	0	1	1	2214	4877	26	4371	2	1	1	0
32768	0	1	1	1	1	0	1	1	2374	5254	26	4774	2	1	1	0
32768	0	0	0	0	0	1	1	1	2247	5392	26	4770	3	1	1	1
32768	0	0	0	0	1	1	1	1	2818	6242	26	5353	3	1	1	1
32768	1	0	0	0	1	1	1	1	3047	6478	26	5735	3	1	1	1
32768	2	0	0	0	1	1	1	1	3047	6850	26	6518	3	1	1	1
32768	0	0	0	0	1	1	1	4	3075	7117	26	6166	6	1	0	2
32768	2	0	0	0	1	1	1	4	3312	7489	26	6927	6	1	0	2
32768	0	0	0	0	1	1	1	5	2949	7031	26	6038	5	1	0	2
32768	0	0	0	0	1	1	1	5	3401	7389	26	6730	5	1	0	2

IP Interconnect

- **As noted earlier, IP connects to processor via AXI Interconnect Block, but…**



Does this need to be a Processor?

AVNET electronics marketing

Endian Byte Ordering Example

- **AXI is Little-Endian and that's a Big Deal!**

	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

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Checkpoint!

- **Where do you find typical device utilization statistics for soft IP cores?**

  IP Datasheet, which can be linked to from IP Catalog

- **What is the estimated throughput of a High Performance DMA AXI interface?**

  64 bits (8 bytes) * 150 MHz = 1.200 GB/s per interface

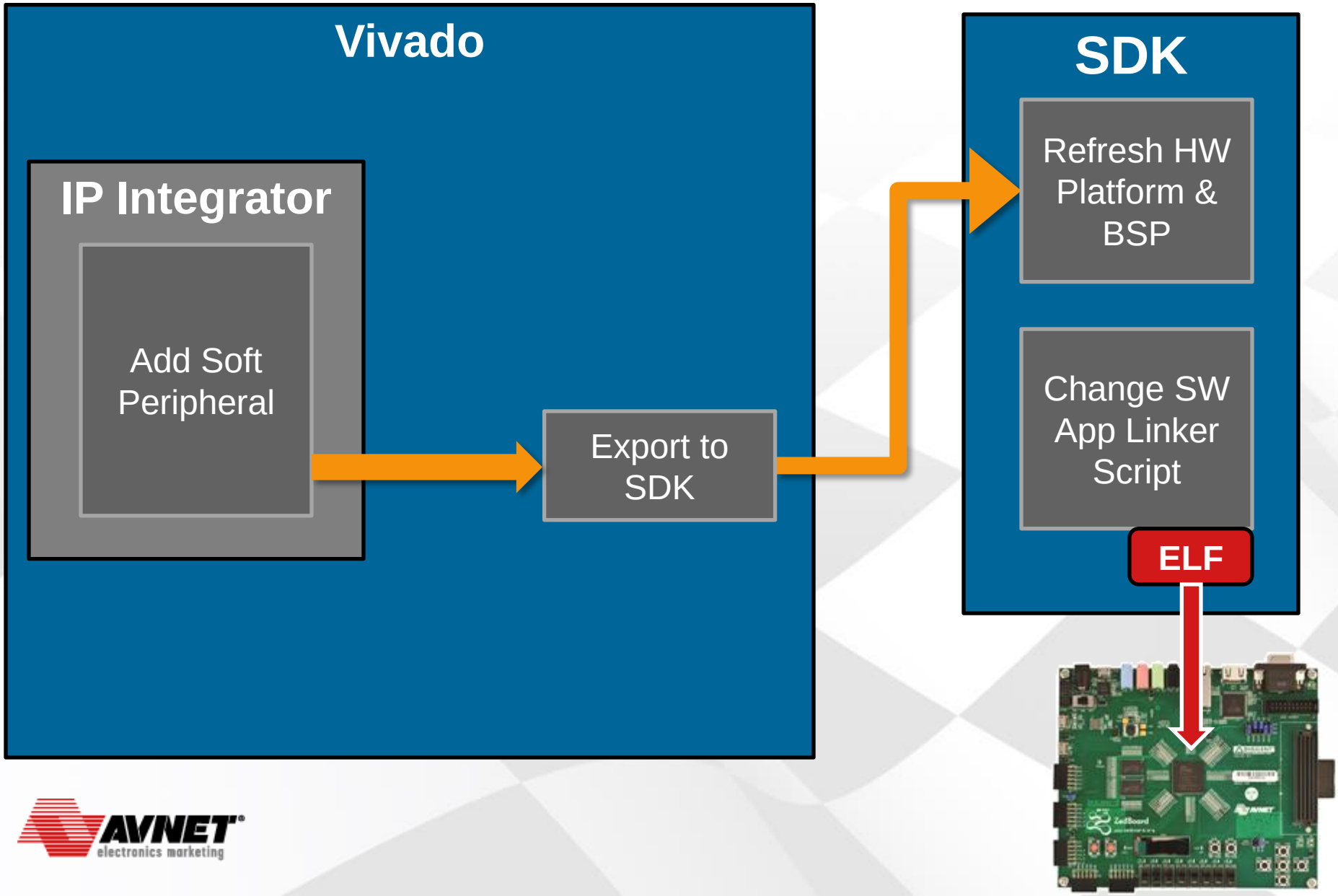
- **How many interfaces can an AXI Interconnect Block support?**

  16 Masters and 16 Slaves

- **How many slave peripherals can the AXI GP ports support?**

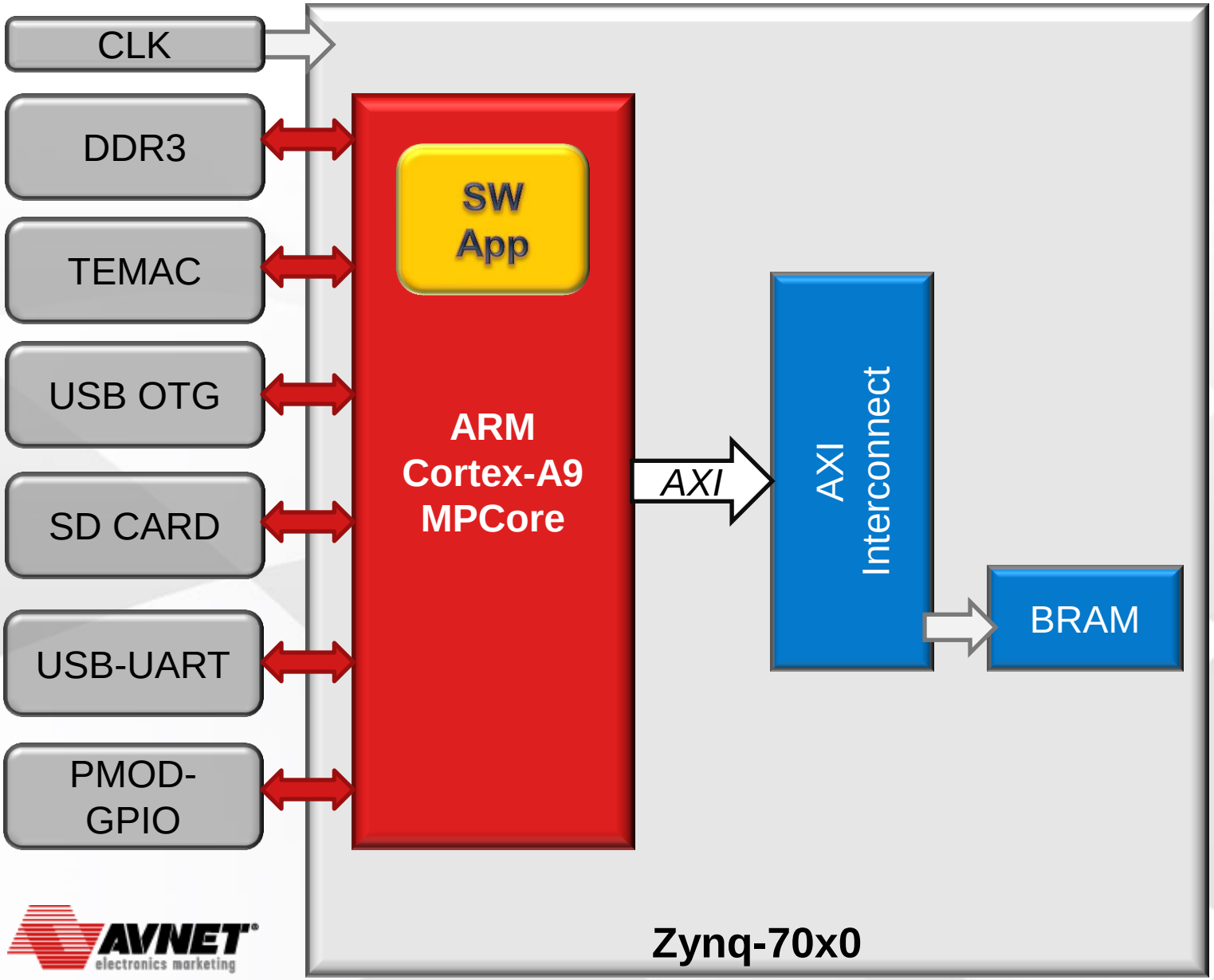
  32 as there are two AXI GP Master ports

Lab 5 – Add PL Peripherals



AVNET electronics marketing

Lab 5 – Add PL Peripheral



Questions

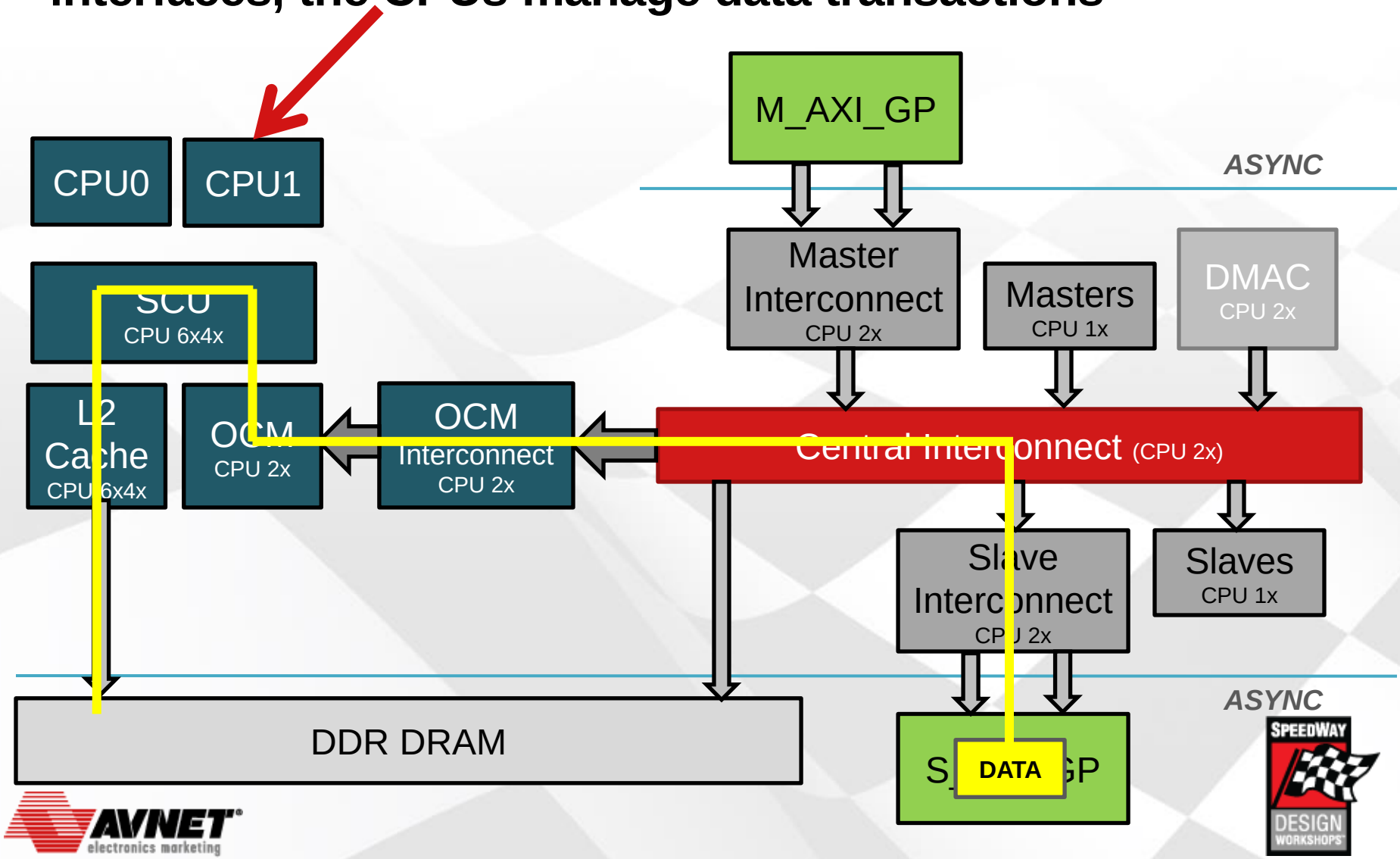
- **How many BRAM's are consumed by the Block Memory Generator?**
  - 16
- **What is the base address of the BRAM? Why is it mapped here?**
  - 0x40000000, because this is starting address space for M_AXI_GP0 (refer to the Zynq All Programmable SoC User Guide).
- **If more IP peripherals were connected, where would they connect?**
  - Add Master AXI ports to the AXI Interconnect Block for additional slaves to connect into.

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

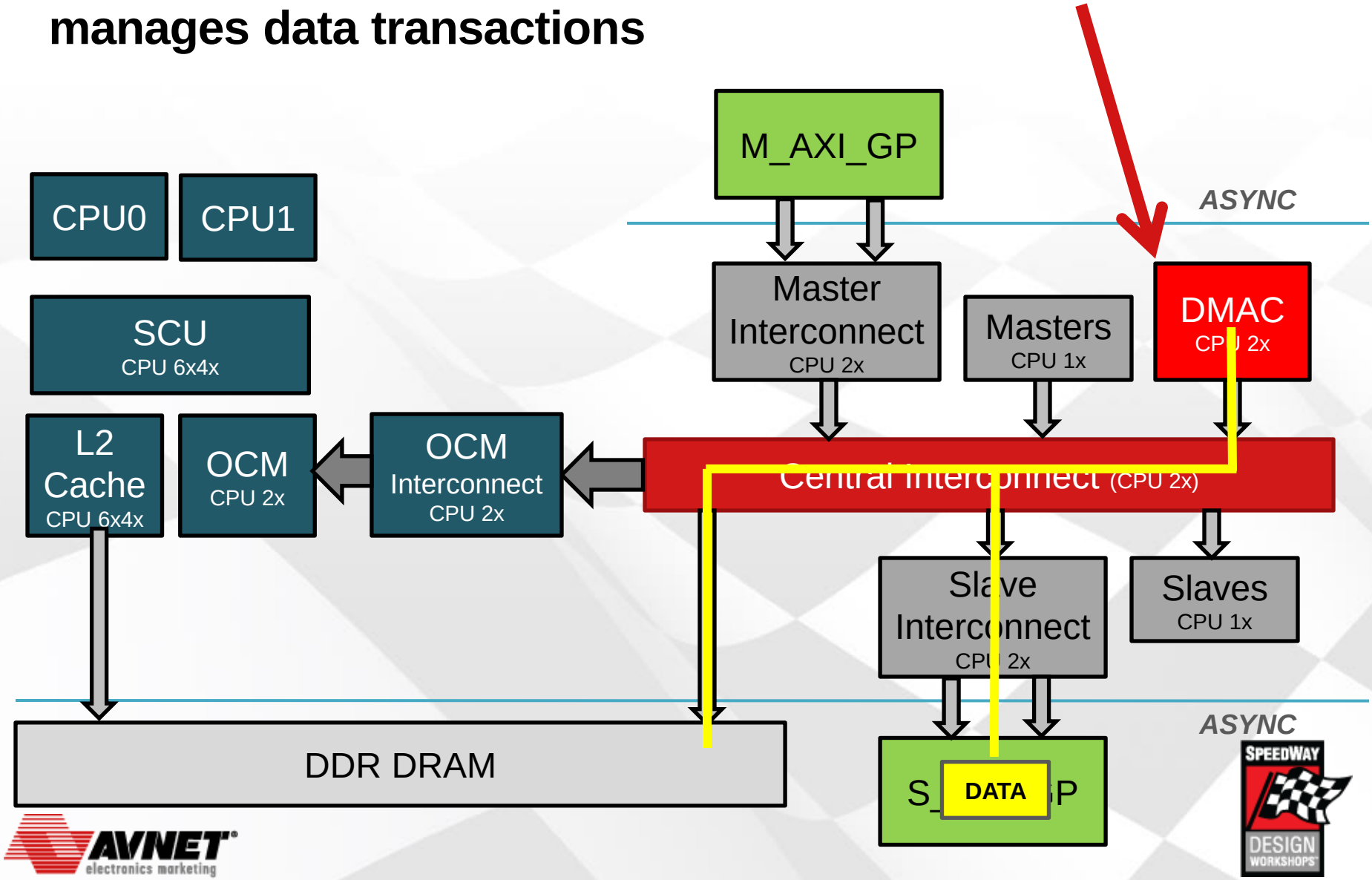
PS-PL Data Transfers

- **When data passes between the PS and PL via AXI GP interfaces, the CPUs manage data transactions**



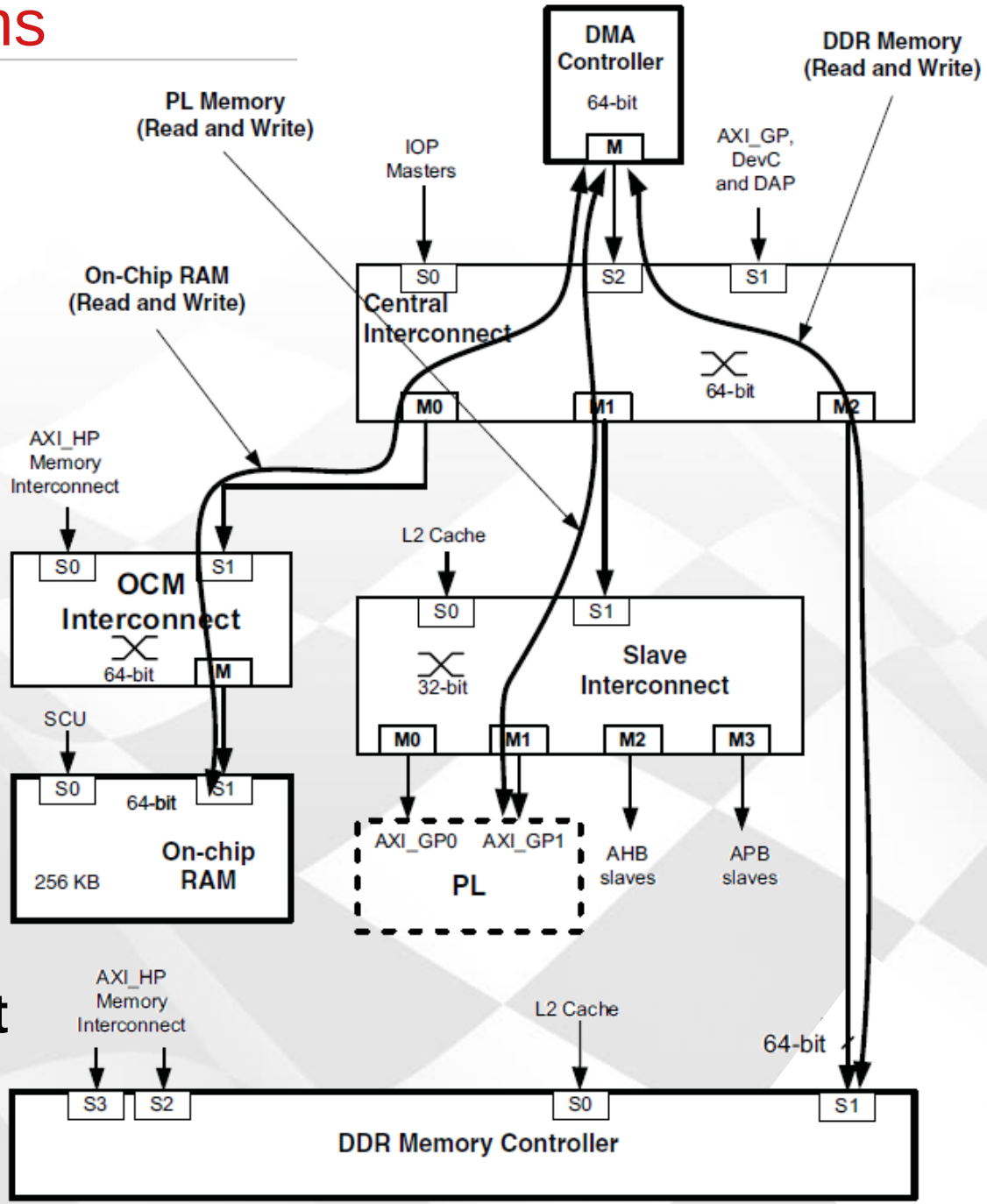
PS-PL DMA Data Transfers

- **When utilizing a DMA Controller (DMAC), the DMAC manages data transactions**



PS DMAC R/W Options

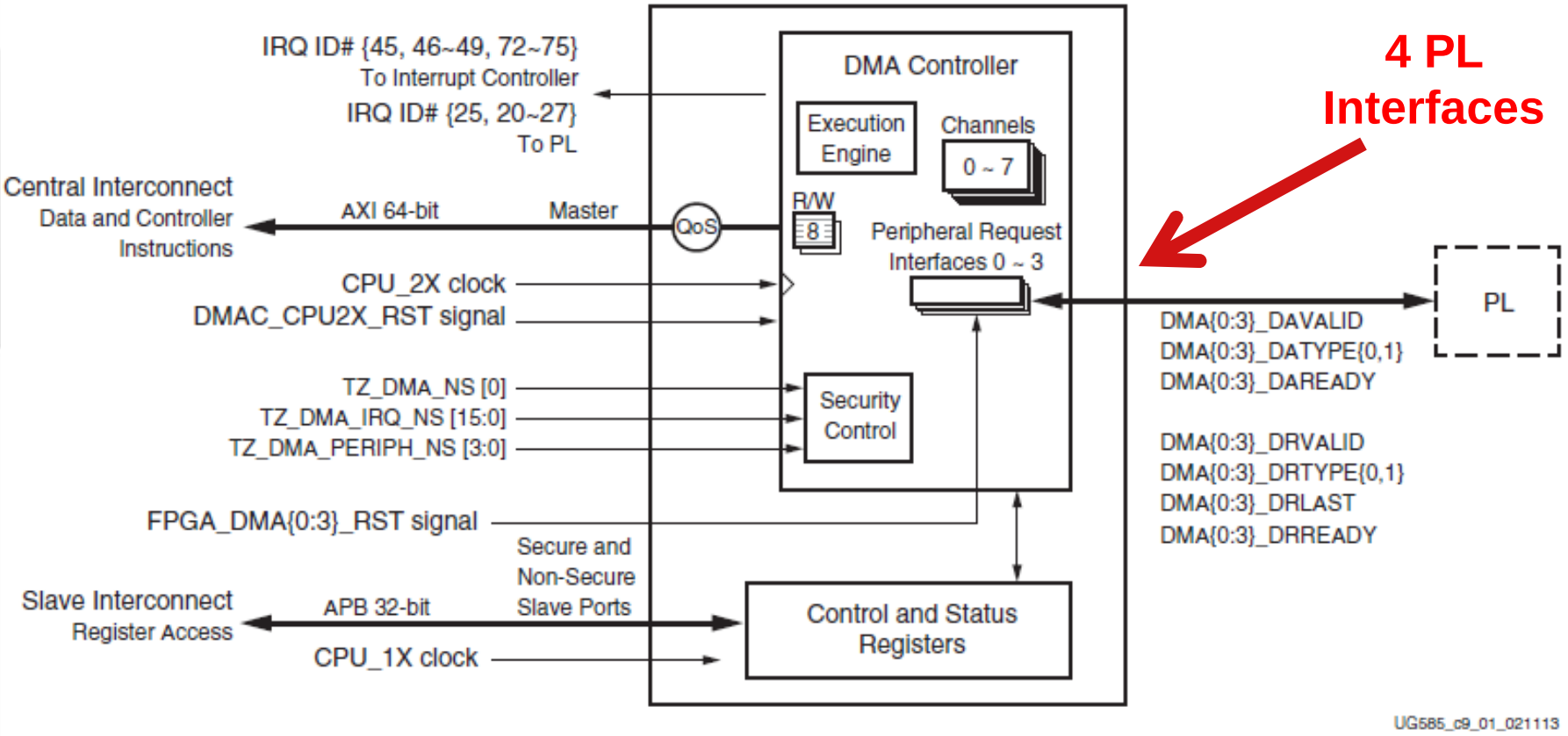
- **The PS DMAC can move data between:**
  - On-chip Memory
  - DDR Memory
  - Slave PL Peripherals
- **Each path can be read or write**
- **Common Transactions:**
  - Memory to Memory
  - DDR Memory to/from PL Peripheral
- **Scatter Gather support**



AVNET electronics marketing

DMA Controller System Viewpoint

- **Eight concurrent DMA channels**
  - Four for the programmable logic
  - Four for the processing system
- **Eight interrupt lines**
- **64 deep Multi-channel FIFO**
- **Support for both 32-bit and 64-bit transfers**

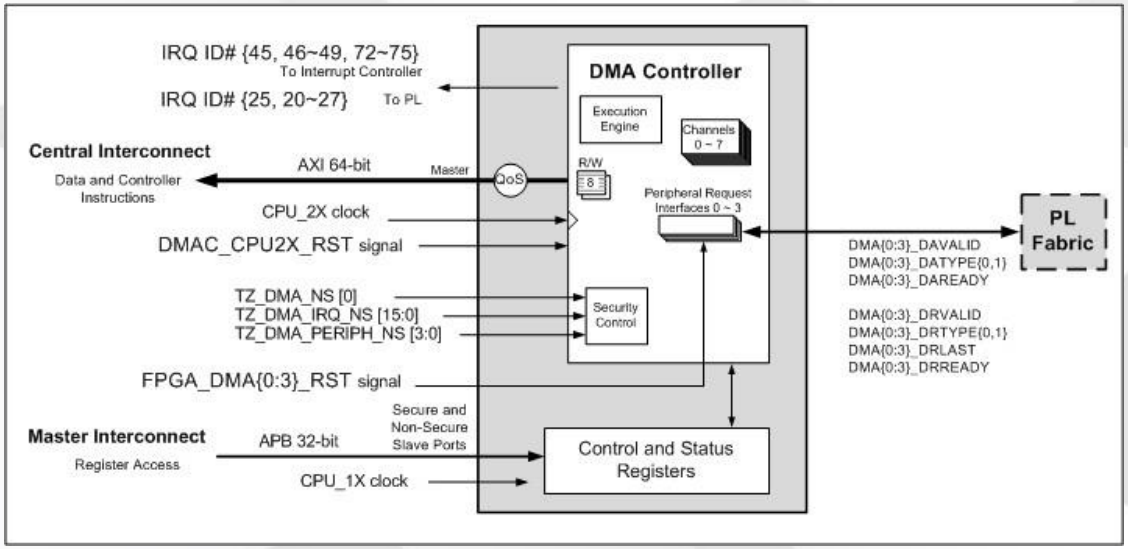


DMA Instruction Execution Engine

- **Contains an instruction processing block that enables it to process program code that controls a DMA transfer**
- **Maintains a separate state machine for each thread**
- **Channel arbitration**
  - Round-robin scheme to service the active DMA channels
  - Services the DMA manager prior to servicing the next DMA channel
  - Changes to the arbitration process are not supported
- **Channel prioritization**
  - Responds to all active DMA channels with equal priority
  - Changes to the priority of a DMA channel over any other DMA channels is not supported

Programming the PS DMA

- **DMAs require a fair bit of programming**
- **Similar to an assembly language**
  - Variable-length instructions 1-6 bytes
  - Separate program counter for each channel
- **Documented in the Zynq-7000 All Programmable SoC Technical Reference Manual (UG585)**
  - 67 pages



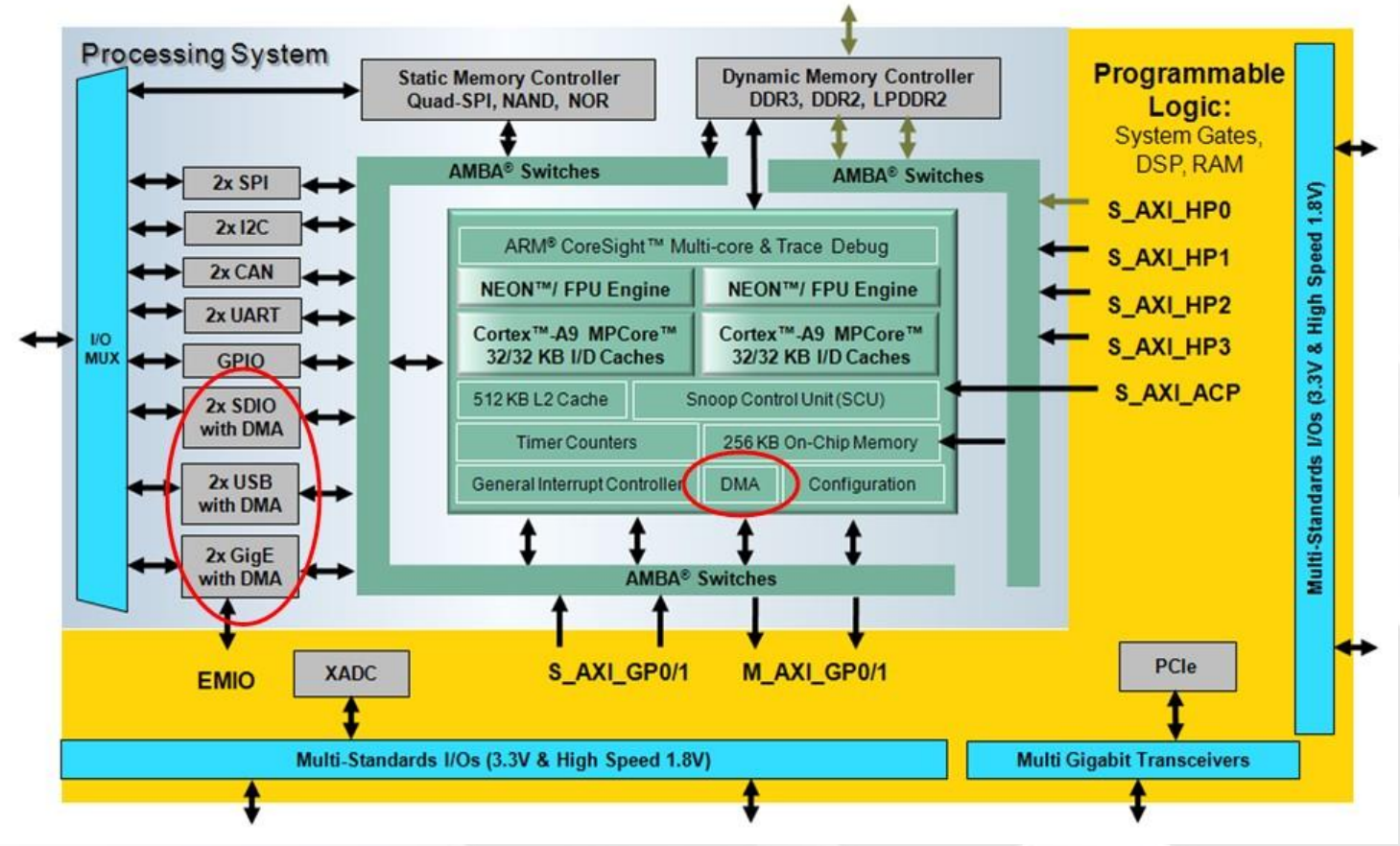
Programming Guide for DMA Controller

- **Startup**
  - Example: Start-up Controller
    1. Configure Clocks
    2. Configure Security State
    3. Reset the Controller
    4. Create Interrupt Service Routine
    5. Execute DMA Transfers
- **Execute a DMA Transfer**
    1. Write Microcode into Memory for DMA Transfer
        a. Create a program for the DMA channel.
        b. Store the program in a region of system memory.
    2. Start the DMA Channel Thread
- **Create ISR**

See UG585 for more details

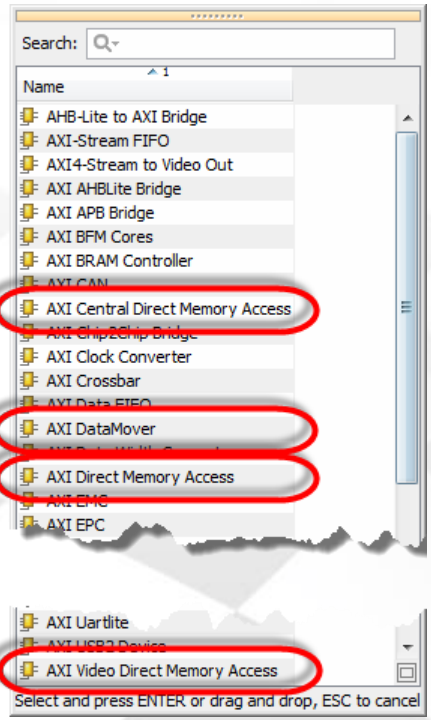
Additional PS DMA Controllers

- **DMA is intrinsically supported in USB, Ethernet, and SD-SDIO**



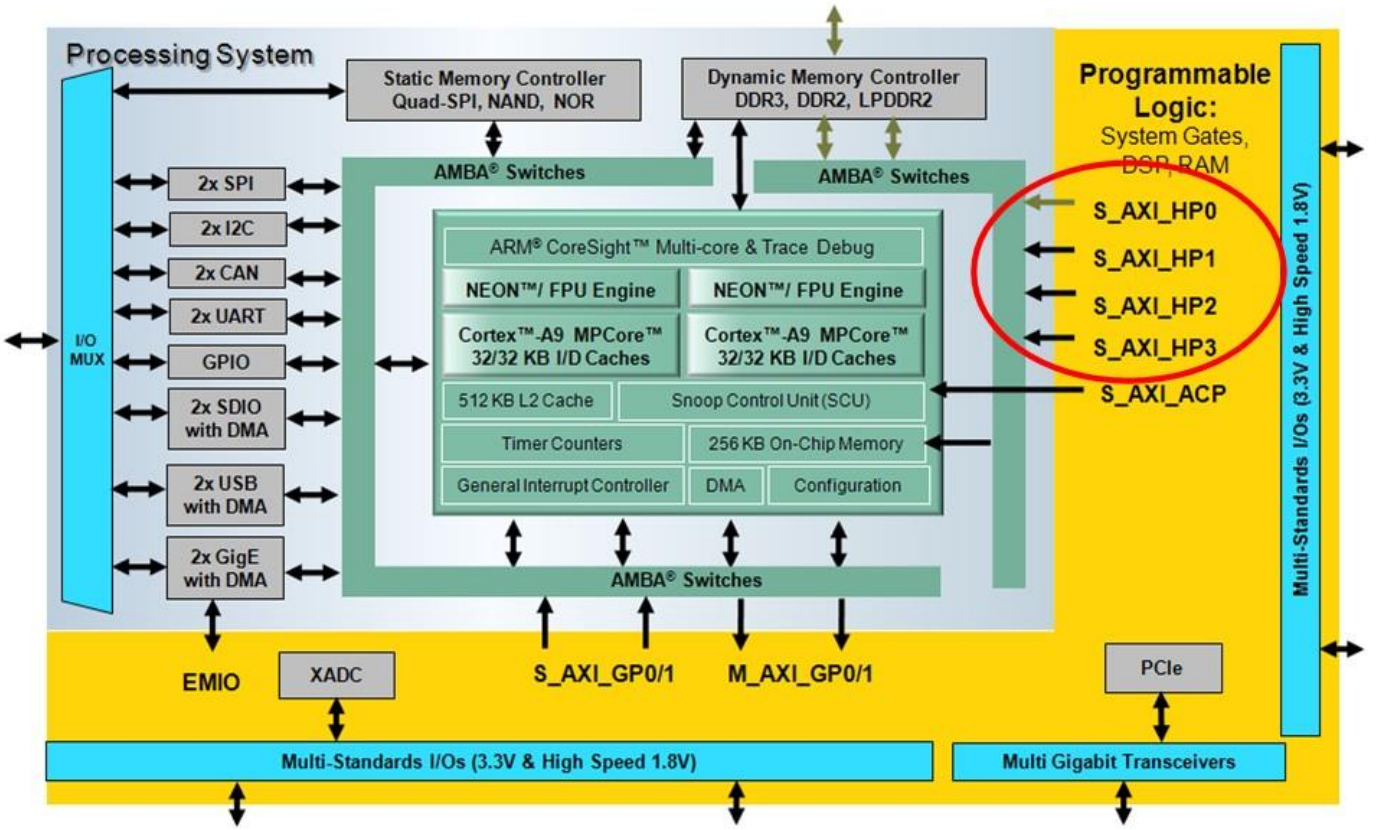
AXI Direct Memory Access

- **Four PL-based DMACs**
  - AXI Central DMA (CDMA)
    - Memory-to-memory operations
  - AXI DMA
    - High Bandwidth DMA
    - Memory to/from AXI Stream-type peripherals
  - AXI DataMover
    - FIFO Memory Mapped to Streaming
    - Streaming AXI interface alternative to traditional DMA, no Scatter Gather
  - AXI Video DMA
    - Optimized for streaming video application to/from memory



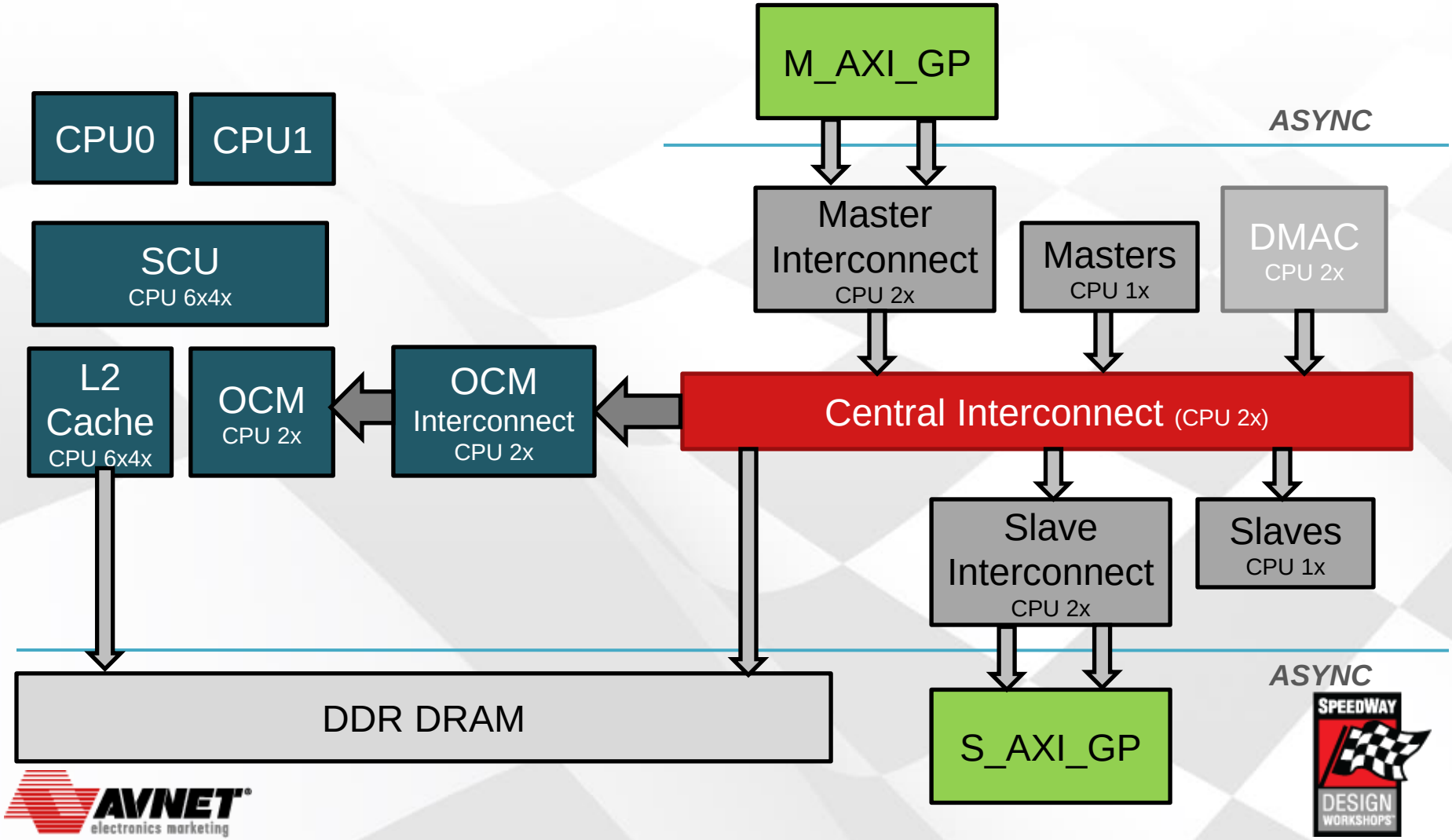
Support for HP PL DMA

- **PL-based DMA engines are used to transfer data from PL components to DDR and OCM**



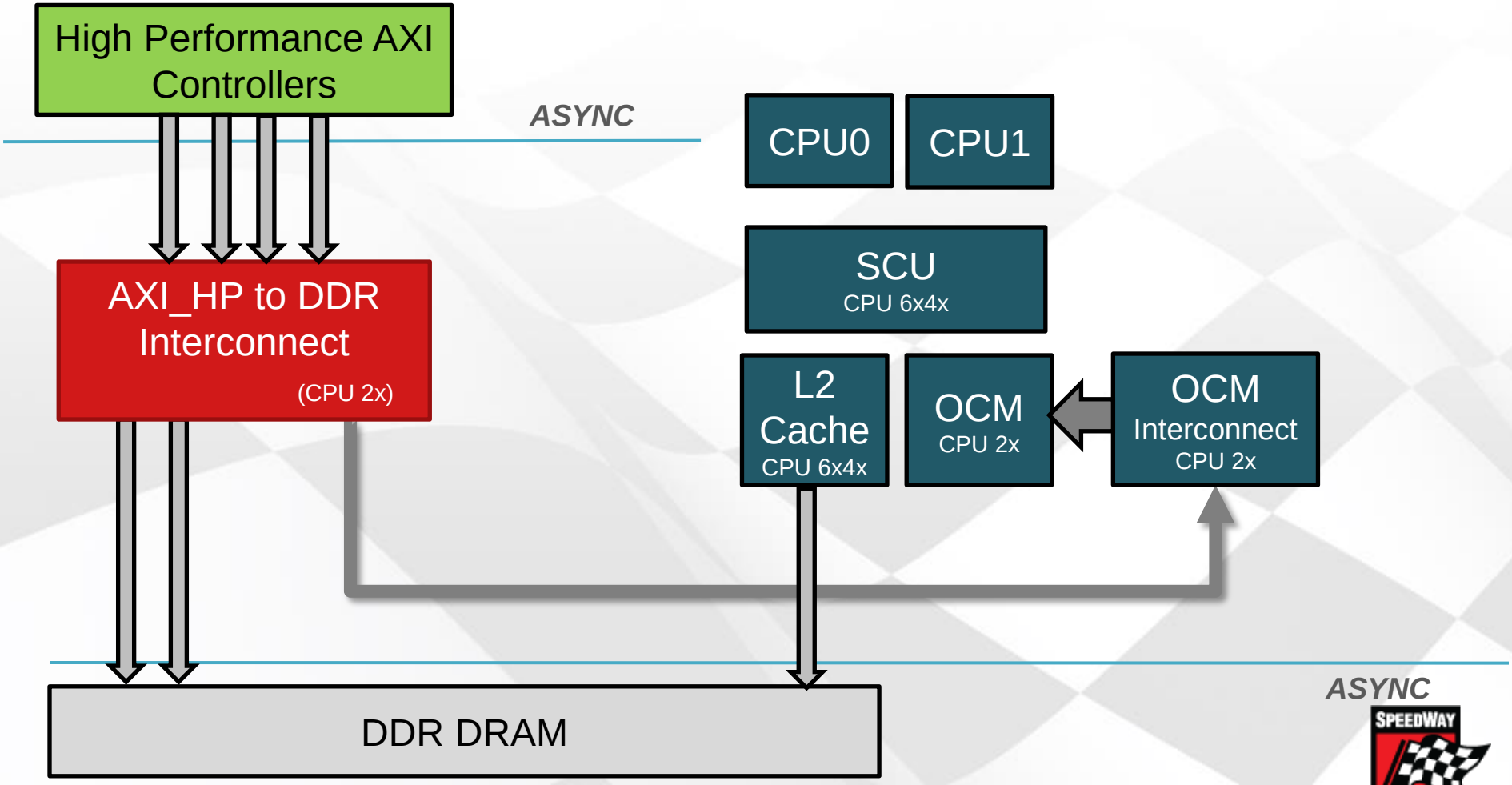
PS-PL Data Transfers

- **When data passes between the PS and PL via AXI HP interfaces, data can pass directly to Memory**

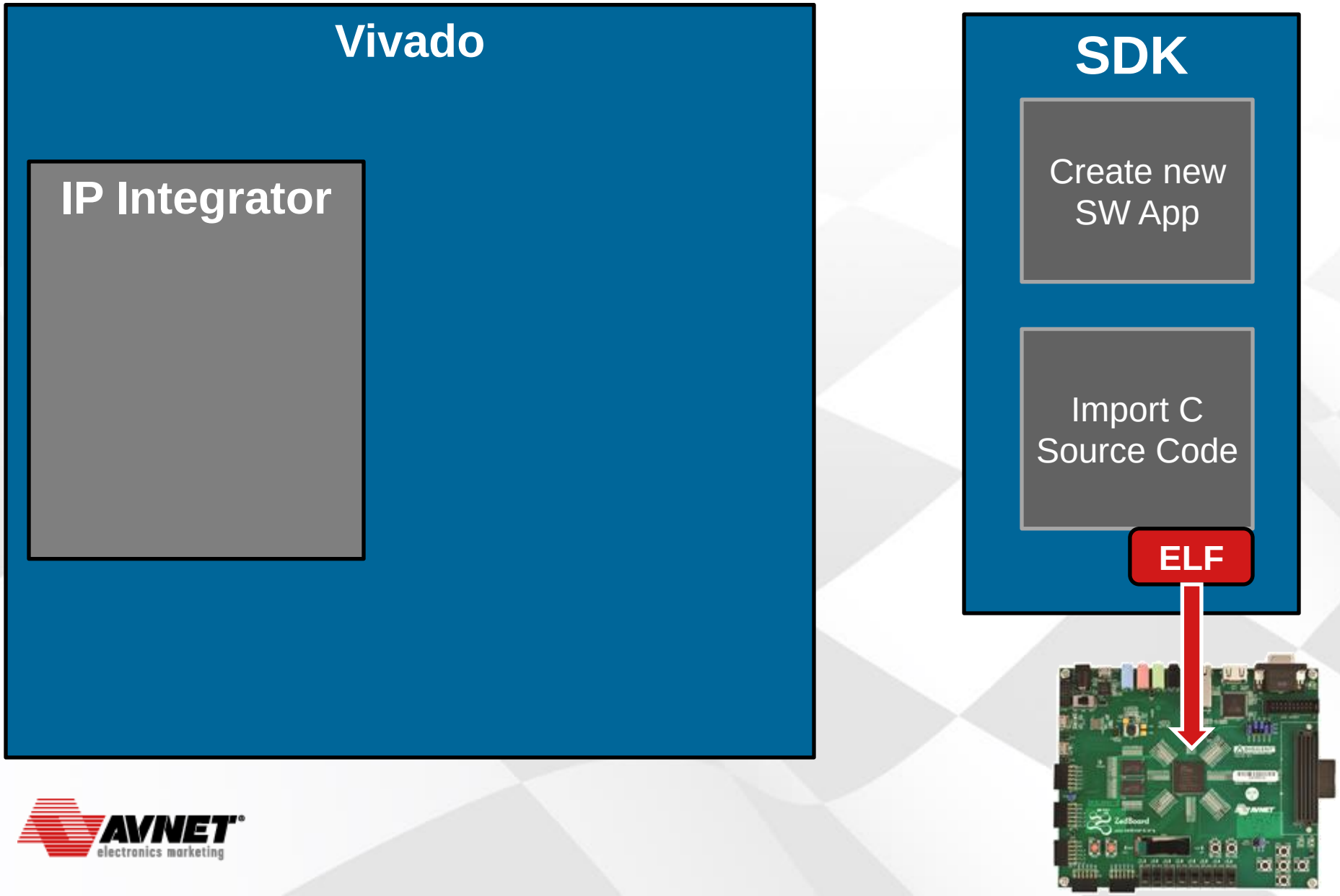


HP PS-PL Data Transfers

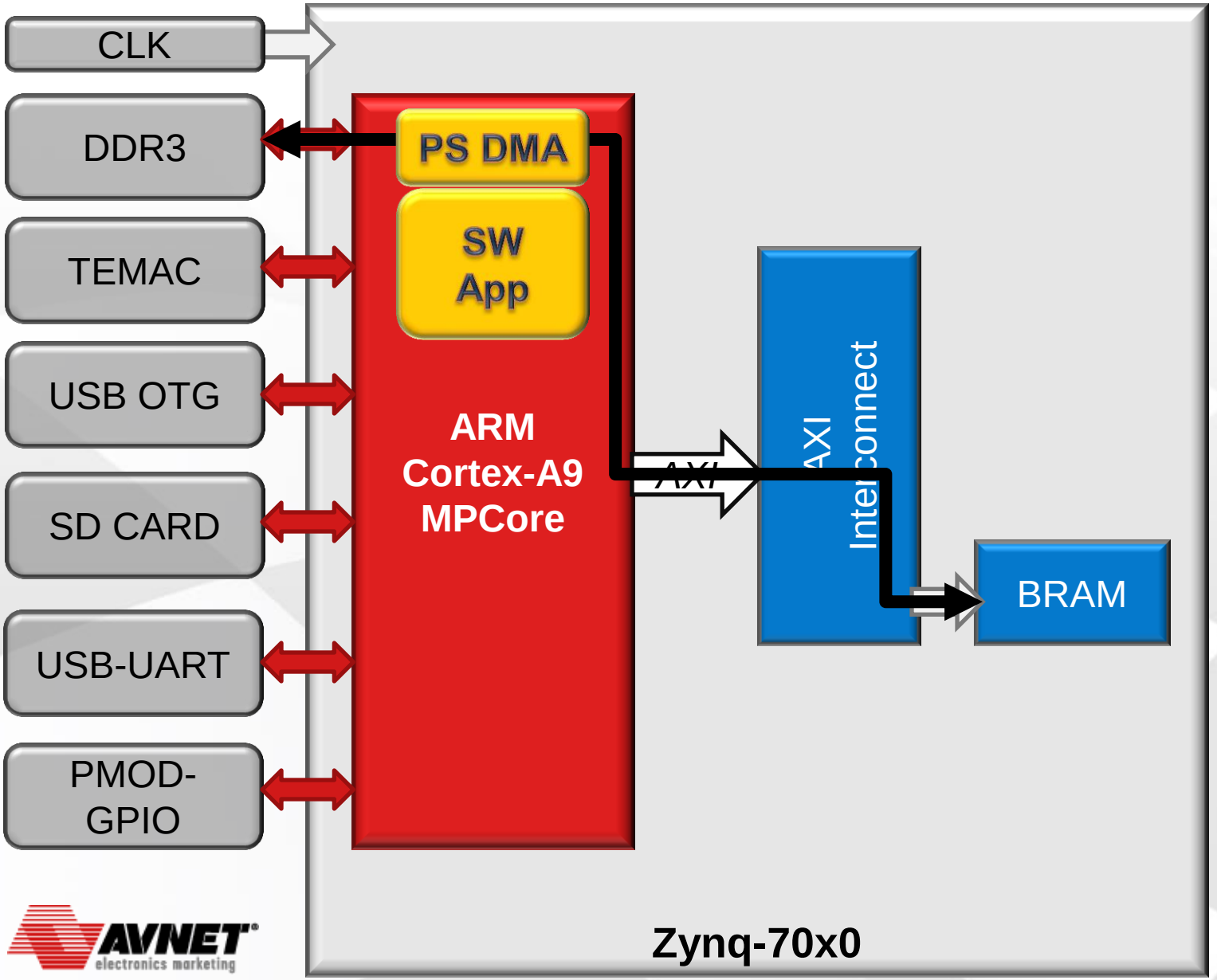
- **When data passes between the PS and PL via AXI HP interfaces, data can pass directly to Memory**



Lab 6 – Access PS DMA Controller



Lab 6 – Access PS DMA



Questions

- ***Was a new BSP required for this application?***
  - No. We could have used the BSP created for Hello_World_BRAM in the past lab. That BSP included drivers for the PL BRAM. However, it's not necessarily bad practice to generate a new BSP for each new HW platform.
- ***What is the performance increase when transferring 8192 bytes from BRAM to DDR3? Try again from DDR3 to DDR3? BRAM to BRAM?***
  - 11x, 3x, 20x (Numbers may vary)
- ***What is the performance increase when transferring 256 bytes from BRAM to DDR3? Try again from DDR3 to DDR3? BRAM to BRAM?***
  - 10x, 1x, 18x (Numbers may vary)

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

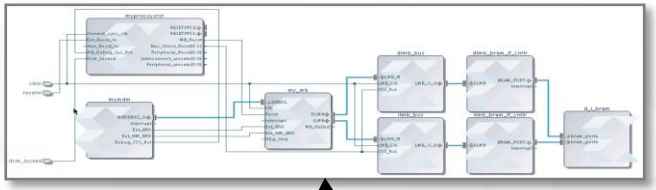
AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

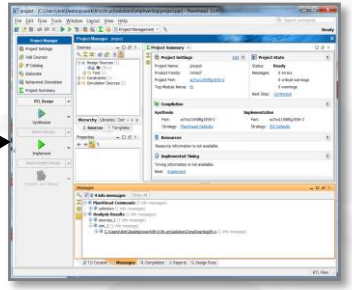
Expanding the IP Catalog

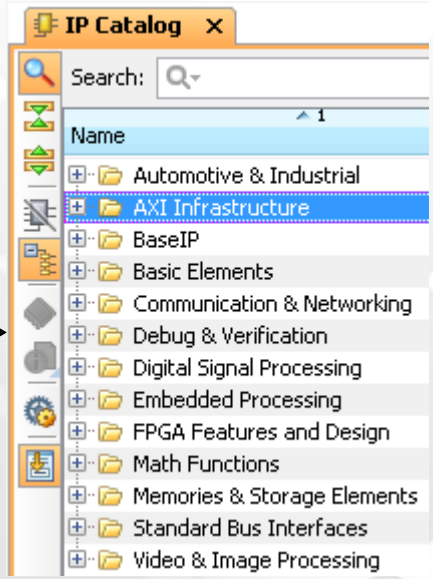
- **Wouldn't it be nice to have your own IP in the Vivado IPI Catalog?**

Vivado IP Integrator



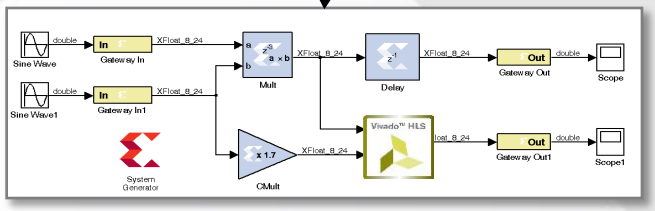
Vivado RTL Integration





Vivado IP Catalog

Vivado HLS



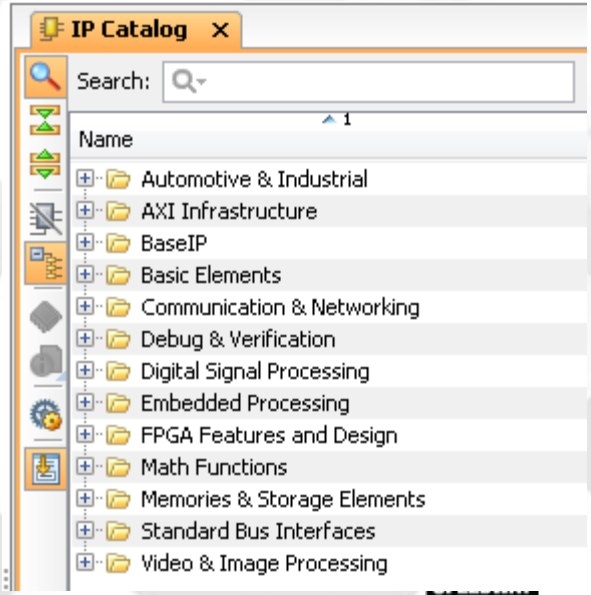
System Generator for DSP

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Vivado IP Overview

- **Vivado has a new and unified catalog with all IP in one place**
  - The Vivado IP catalog uses IP-XACT to store metadata
- **IP can be added to the catalog by users (IP Packager)**
- **All IP deliver IEEE P1735 simulation models as an output product**
  - Xilinxcorelib only required for support of legacy/non-updated IP cores
- **Streamlined versioning of Xilinx IP**



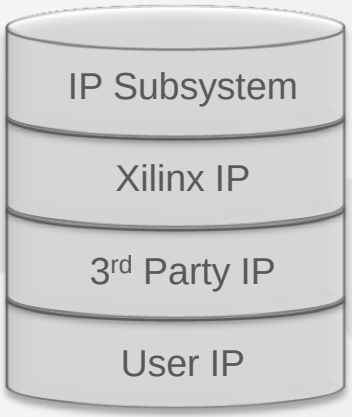
IP Integrator – IP Packager

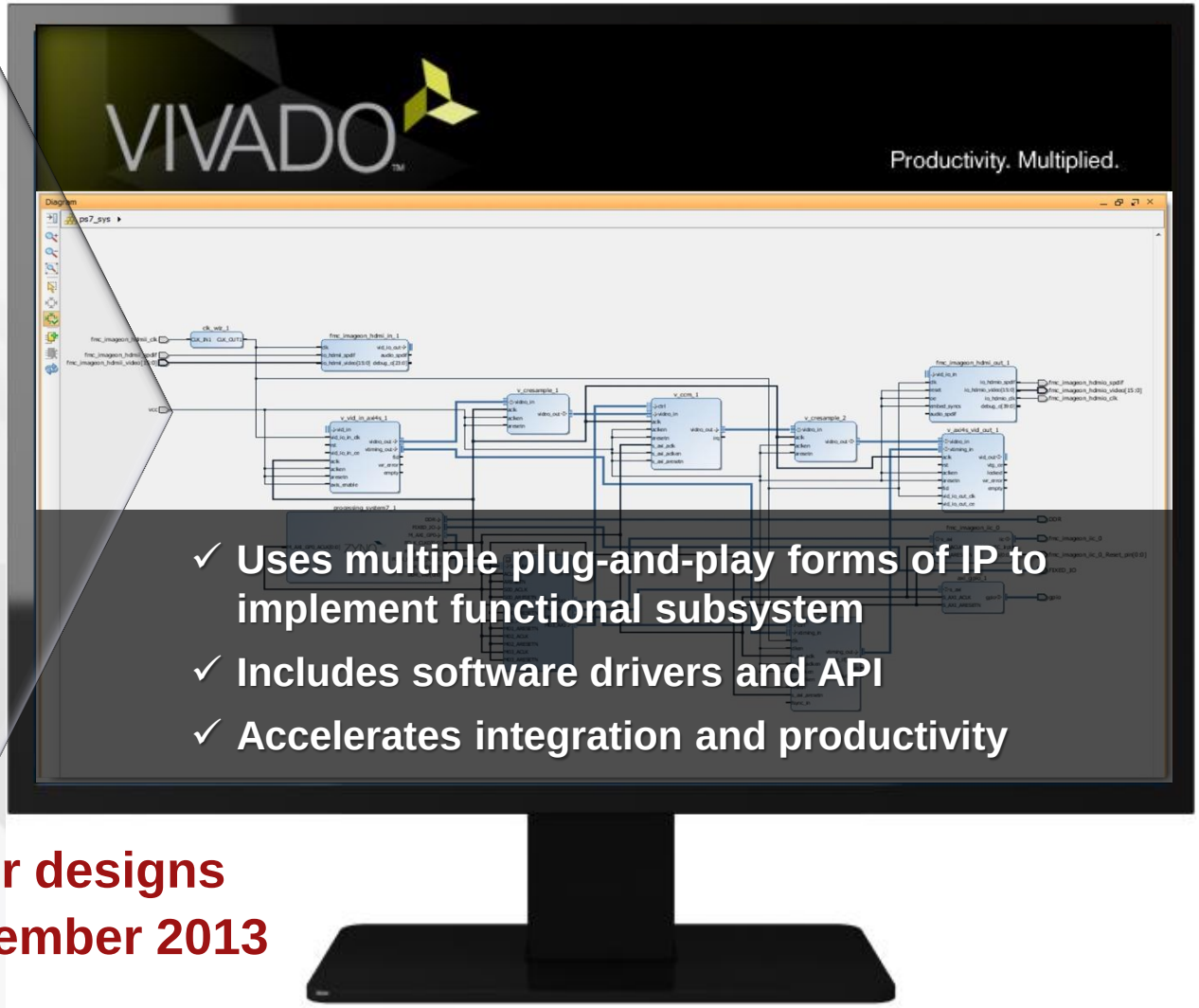
Enabling Reuse and Delivering Fully Functional IP Subsystems

IP Packager

- Source (C, RTL, IP)
- Simulation models
- Documentation
- Example Designs
- Test bench

Standardized IP-XACT



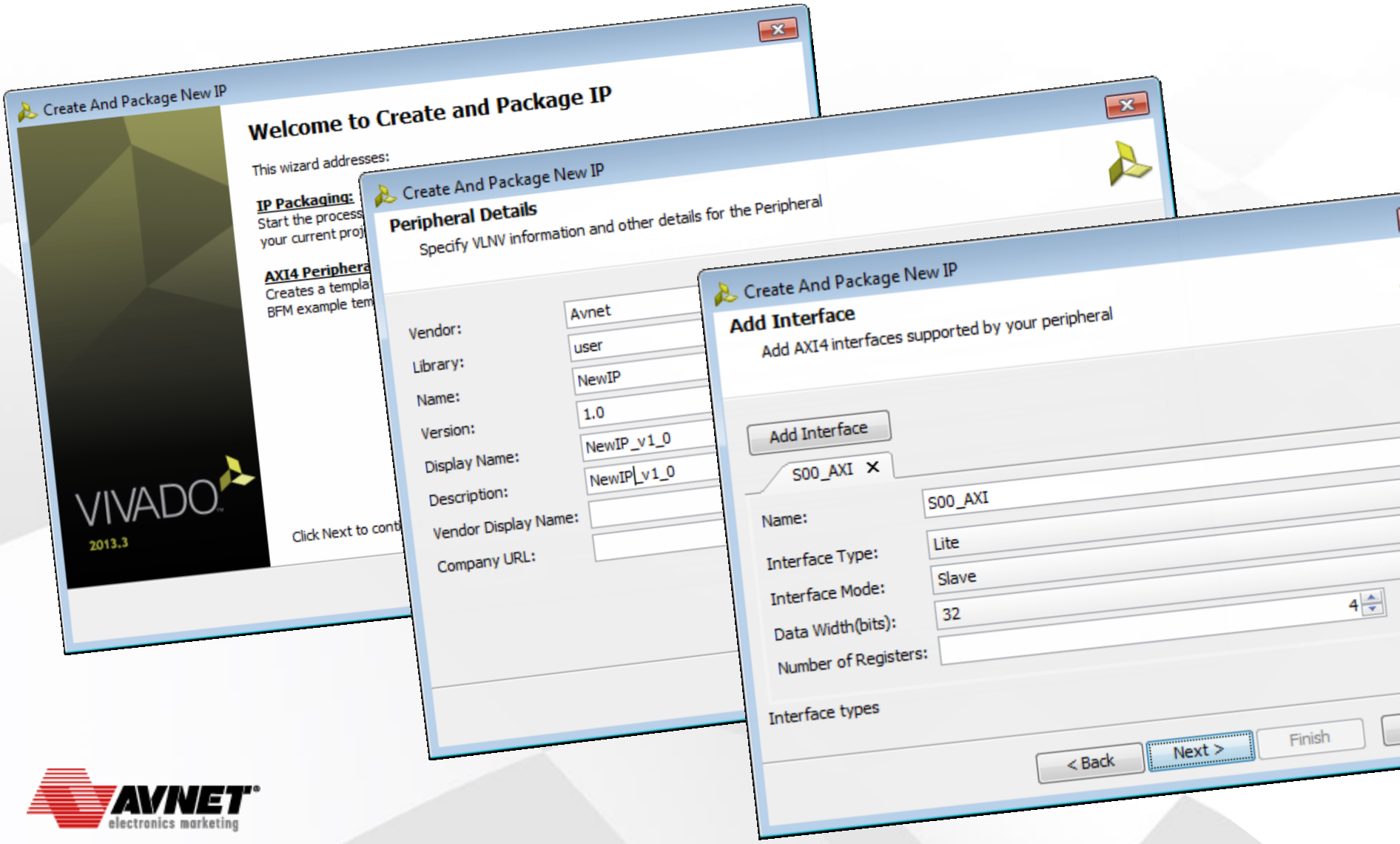


Over 1,000 IPI customer designs generated as of September 2013

AVNET electronics marketing

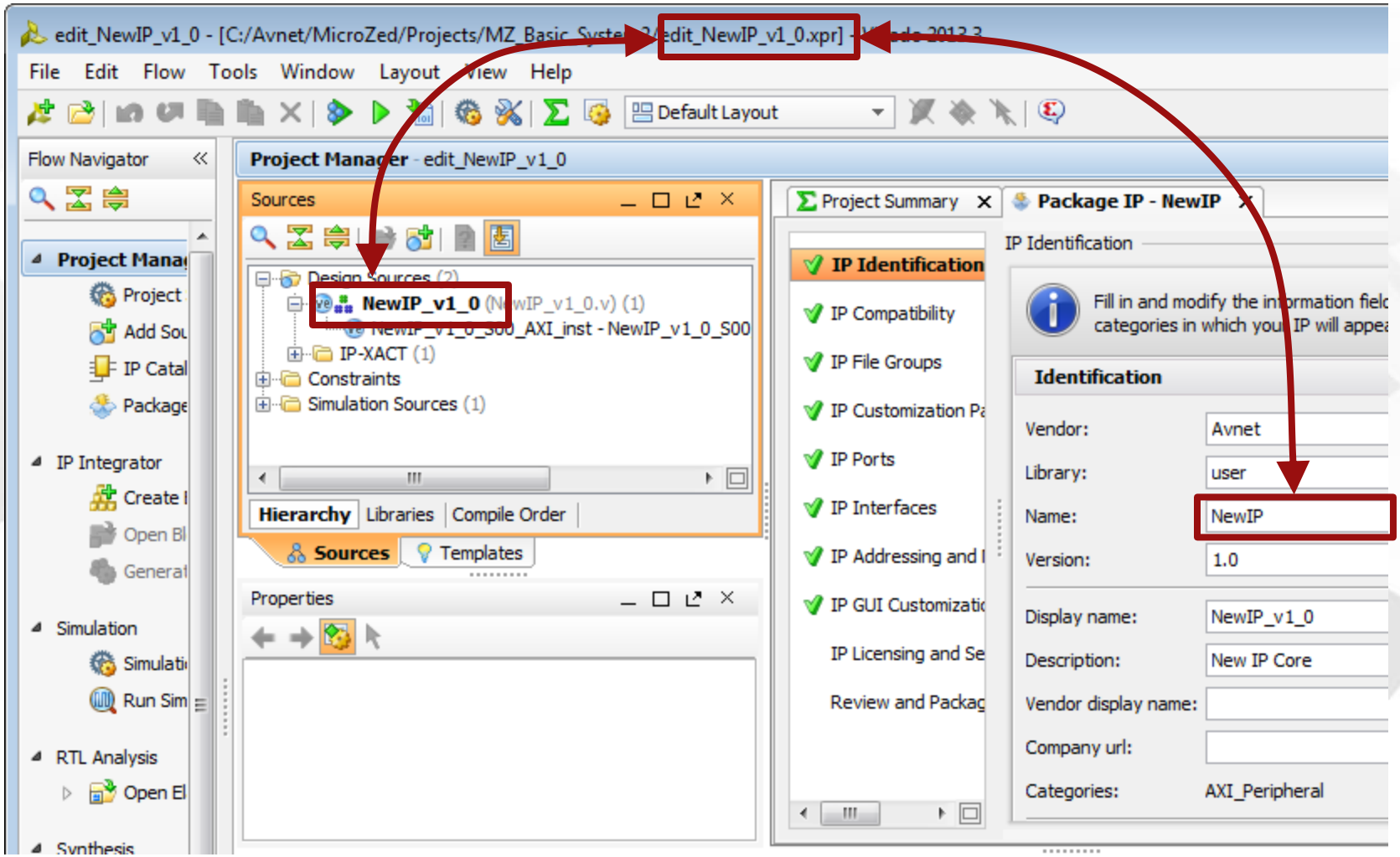
Creating New IP with IP Packager

- **Vivado provides wizard to create and package IP**

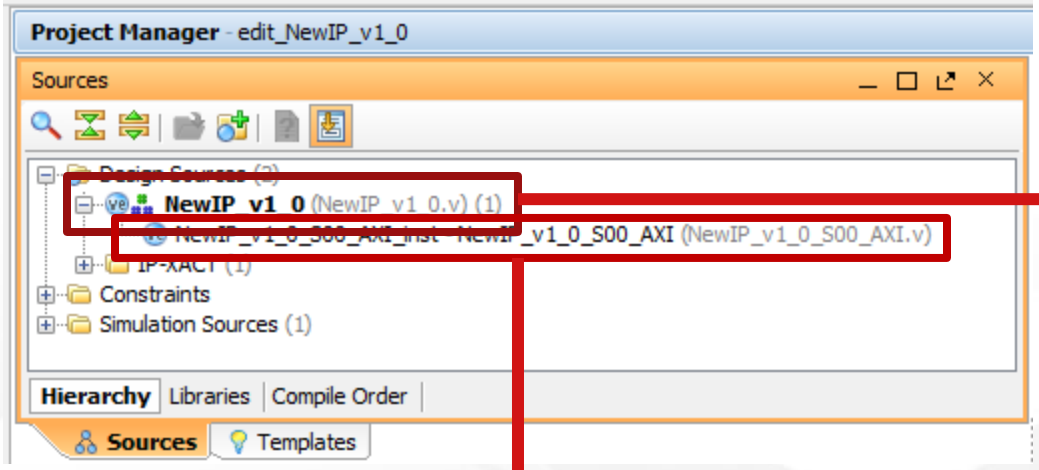


IP Packager creates a new Vivado Project

- **Inherits name of the IP, edit_<ip_name>_v1_0.xpr**
  - **Appends version number**



IP Packager Creates HDL Sources



Top-Level HDL Source

```
module NewIP_v1_0 #
(
    // Users to add ports here
    // Ports of Axi Slave Bus Interface S00_AXI
    input wire  s00_axi_aclk,
    input wire  s00_axi_aresetn,
    input wire [C_S00_AXI_ADDR_WIDTH-1 : 0] s00_axi_awaddr,
    input wire [2 : 0] s00_axi_awprot,
    input wire  s00_axi_awvalid,
    output wire  s00_axi_awready,
    input wire [C_S00_AXI_DATA_WIDTH-1 : 0] s00_axi_wdata,
    input wire [(C_S00_AXI_DATA_WIDTH/8)-1 : 0] s00_axi_wstrb,
    input wire  s00_axi_wvalid,
    output wire  s00_axi_wready,
    output wire [1 : 0] s00_axi_bresp,
    output wire  s00_axi_bvalid,
// Instantiation of Axi Bus Interface S00_AXI
    NewIP_v1_0_S00_AXI # (
        .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),
        .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)
    ) NewIP_v1_0_S00_AXI_inst (
        .S_AXI_ACLK(s00_axi_aclk),
        .S_AXI_ARESETN(s00_axi_aresetn),
        .S_AXI_AWADDR(s00_axi_awaddr),
        .S_AXI_AWPROT(s00_axi_awprot),
        .S_AXI_AWVALID(s00_axi_awvalid),
        .S_AXI_AWREADY(s00_axi_awready),
        .S_AXI_WDATA(s00_axi_wdata),
        .S_AXI_WSTRB(s00_axi_wstrb),
        .S_AXI_WVALID(s00_axi_wvalid),
        .S_AXI_WREADY(s00_axi_wready),
        .S_AXI_BRESP(s00_axi_bresp),
        .S_AXI_BVALID(s00_axi_bvalid),
        .S_AXI_BREADY(s00_axi_bready),
        .S_AXI_ARADDR(s00_axi_araddr),
        .S_AXI_ARPROT(s00_axi_arprot),
        .S_AXI_ARVALID(s00_axi_arvalid),
        .S_AXI_ARREADY(s00_axi_arready),
        .S_AXI_RDATA(s00_axi_rdata),
        .S_AXI_RRESP(s00_axi_rresp),
        .S_AXI_RVALID(s00_axi_rvalid),
```

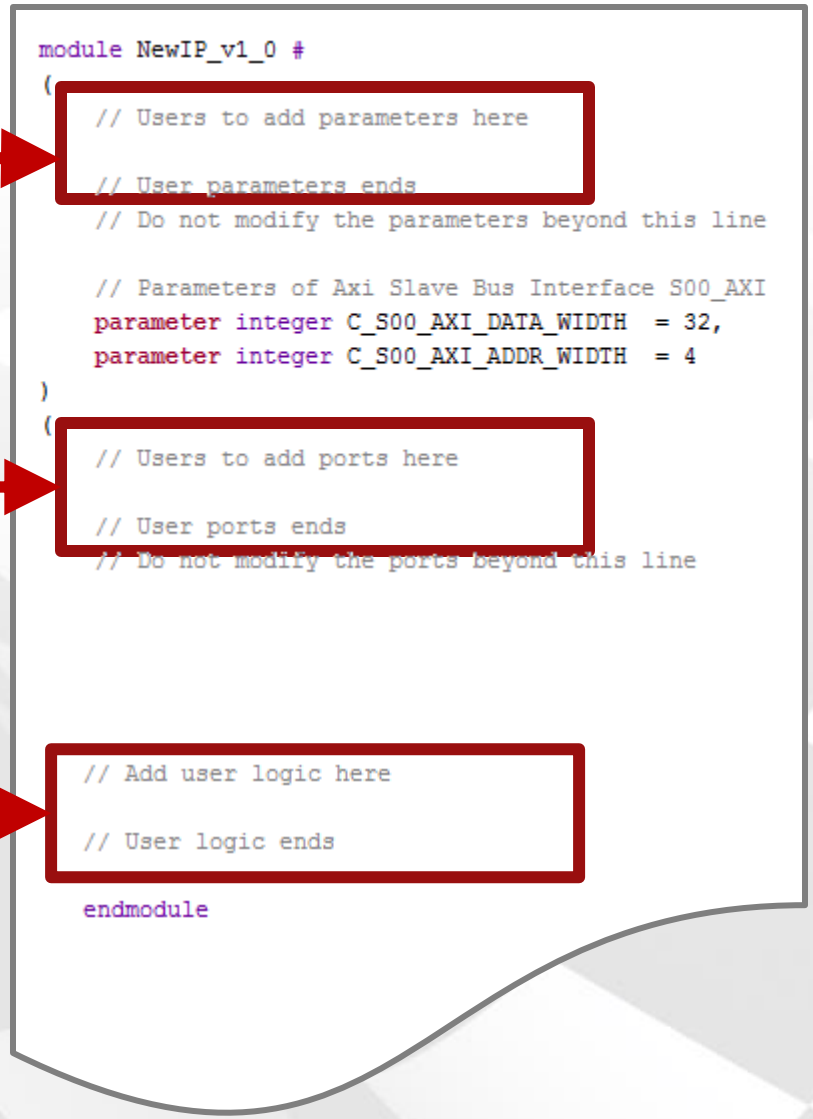
Slave AXI-4-Lite Interface

```
module NewIP_v1_0_S00_AXI #
(
    // Users to add parameters here

    // User parameters ends
    // Do not modify the parameters beyond this line

    // Width of S_AXI data bus
    parameter integer C_S_AXI_DATA_WIDTH    = 32,
    // Width of S_AXI address bus
    parameter integer C_S_AXI_ADDR_WIDTH    = 4
)
(
    // Users to add ports here

    // User ports ends
    // Do not modify the ports beyond this line
```

Instantiation

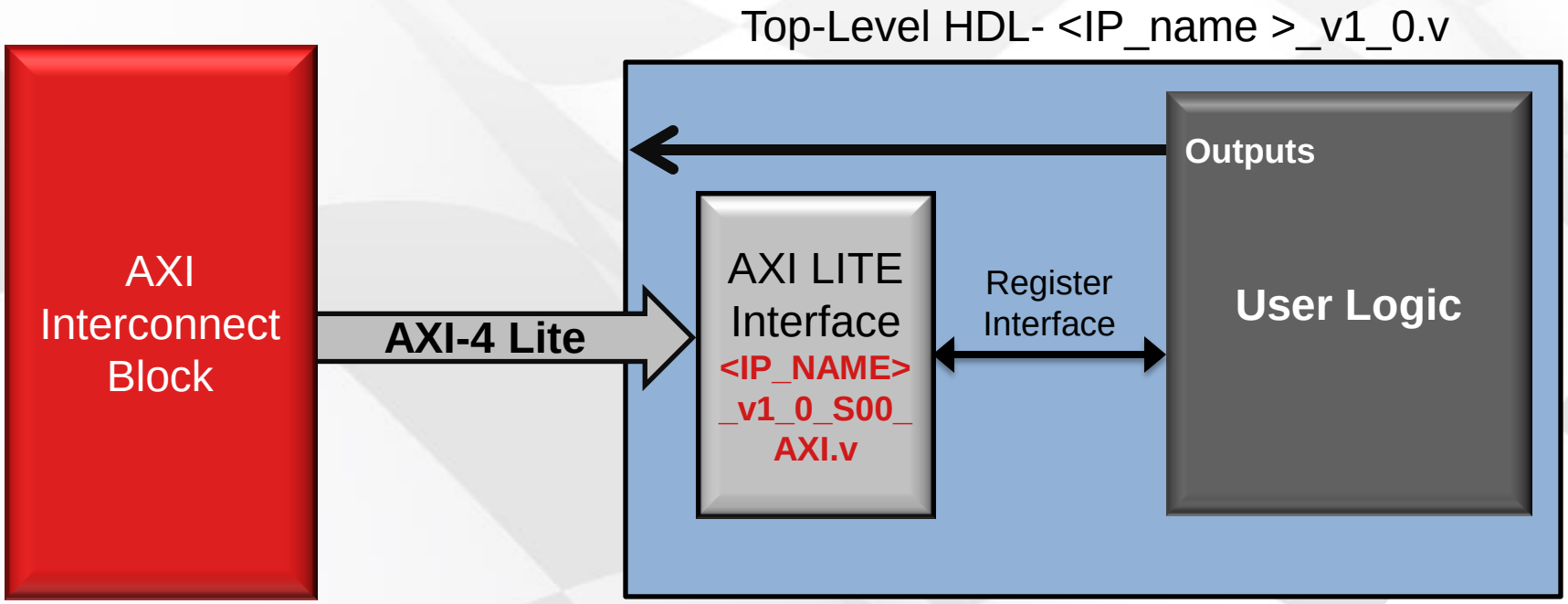
Top-Level HDL

- **Top-Level HDL has placeholders to add:**
  - External Parameters
  - External Ports
  - User Logic

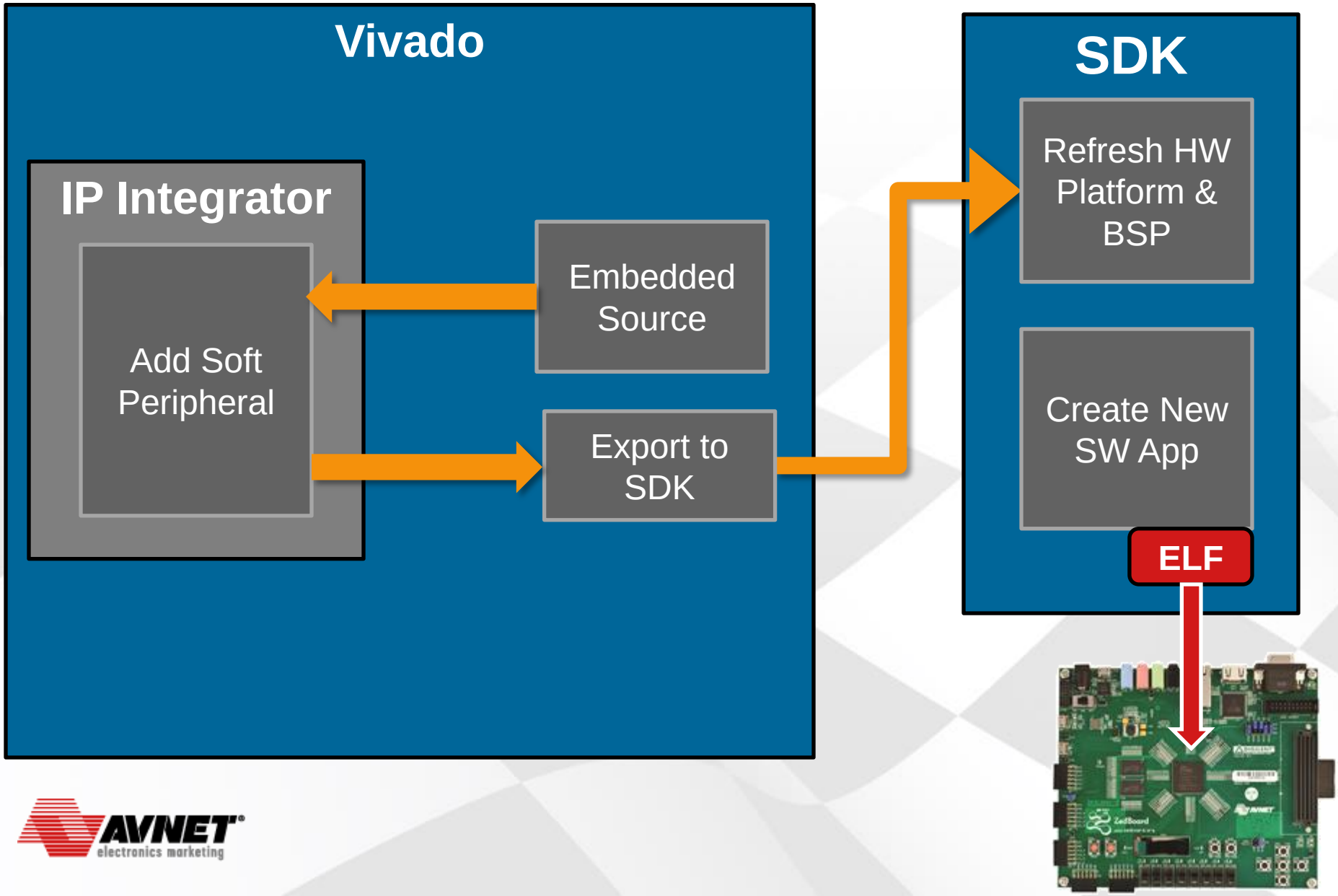


Block Diagram View of Project

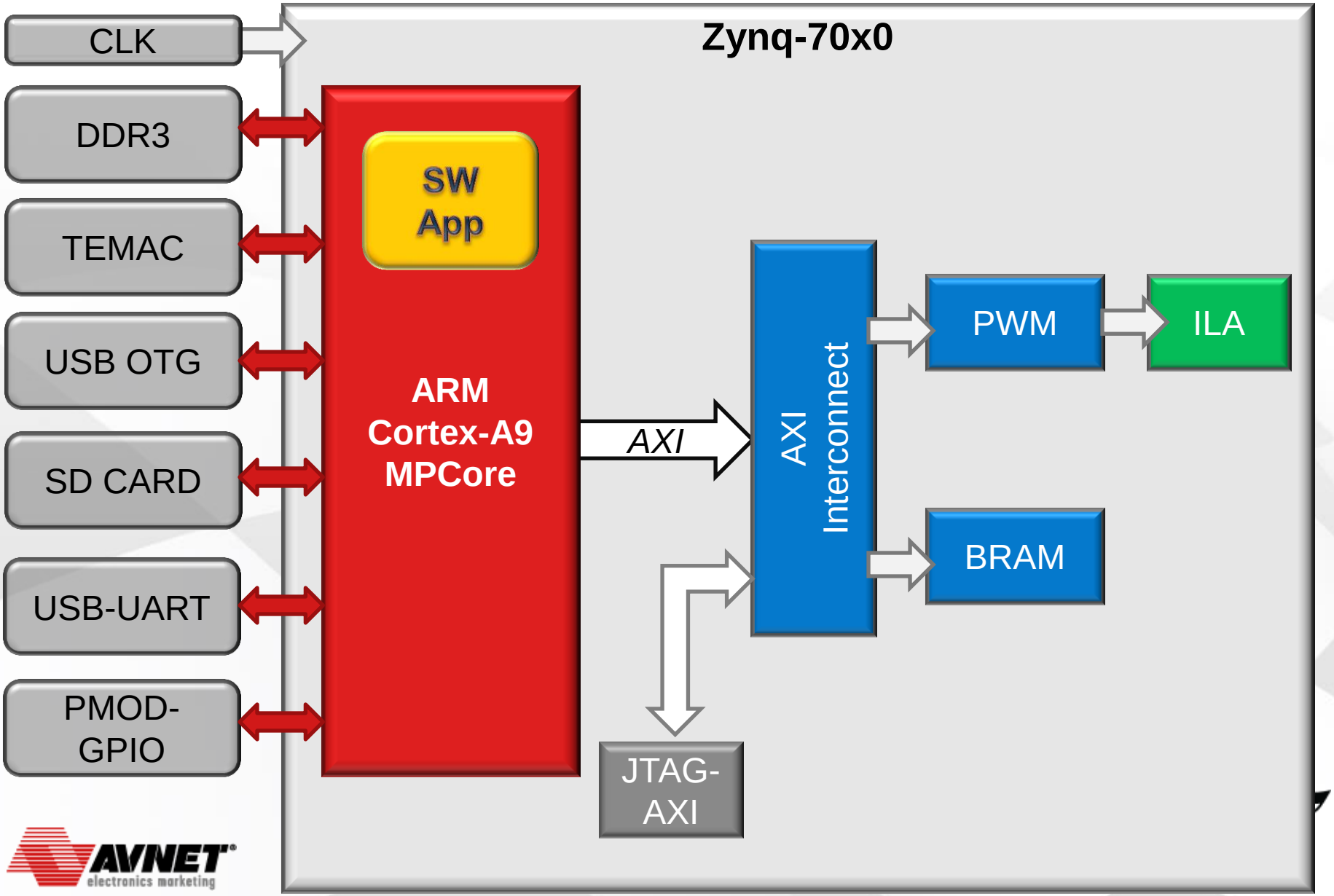
- **<IP_NAME>_v1_0_S00_AXI.v is wrapper file containing AXI interface**
  - Abstracts AXI interface to simple registers
  - Requires editing when exporting external ports
- **User_logic.v is where custom HDL code goes**



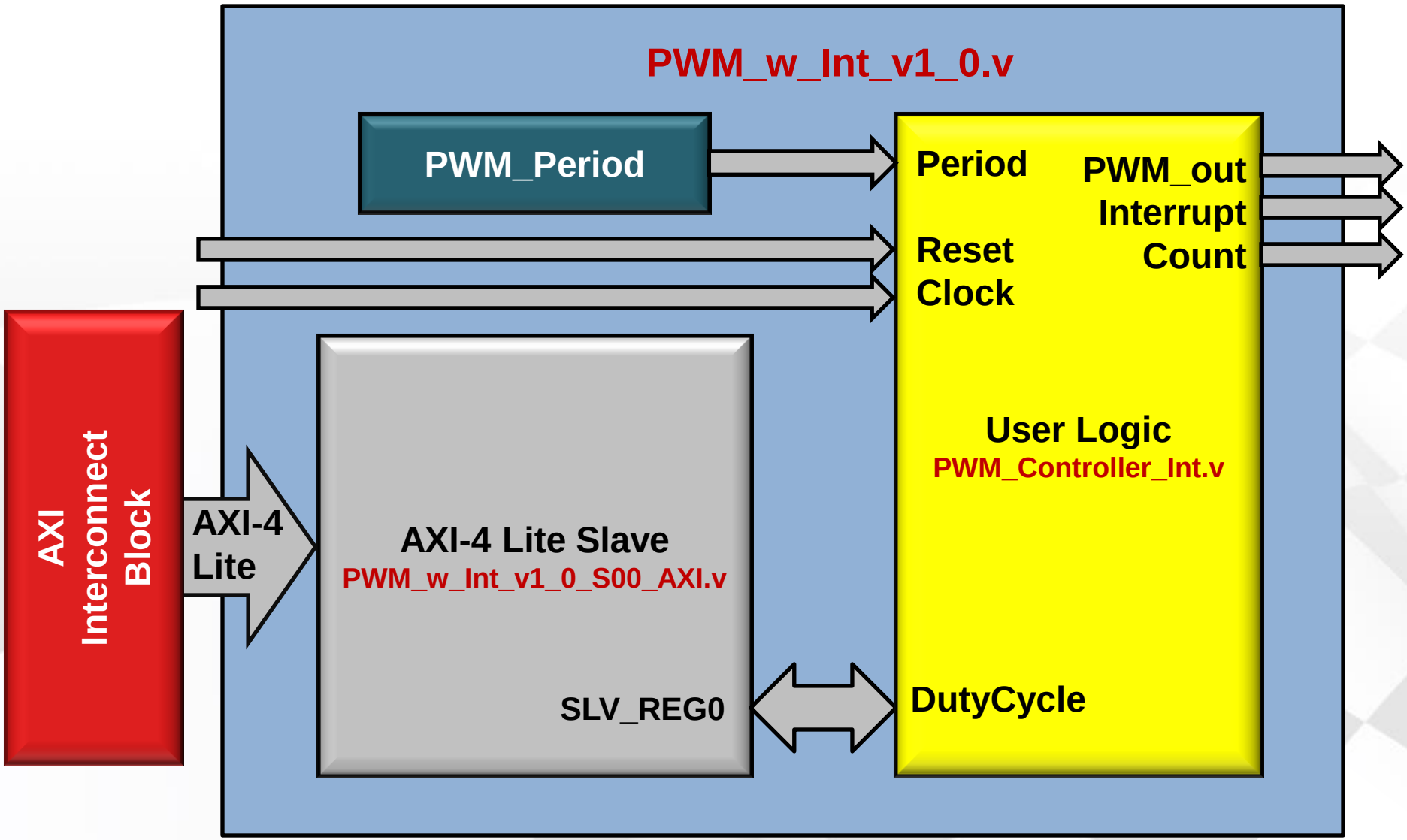
Lab 7 – Add Custom PL Peripheral



Lab 7 – Add Custom PL Peripheral



IP Core Finished



Questions

- ***Where is the IP project located?***
  - In the same directory as our current project:

    C:/Speedway/ZynqHW/2013_3/ZynqDesign
- ***How many AXI Masters and Slaves can be added to the AXI Interconnect?***
  - *16 and 16*
- ***How many interrupts can be generated from fabric resources?***
  - *8 per M_AXI_GP = 16*
  - *Plus nFIQ & nIRQ per core = 4*
  - *20 Total*

Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Recommended Debug Methodology

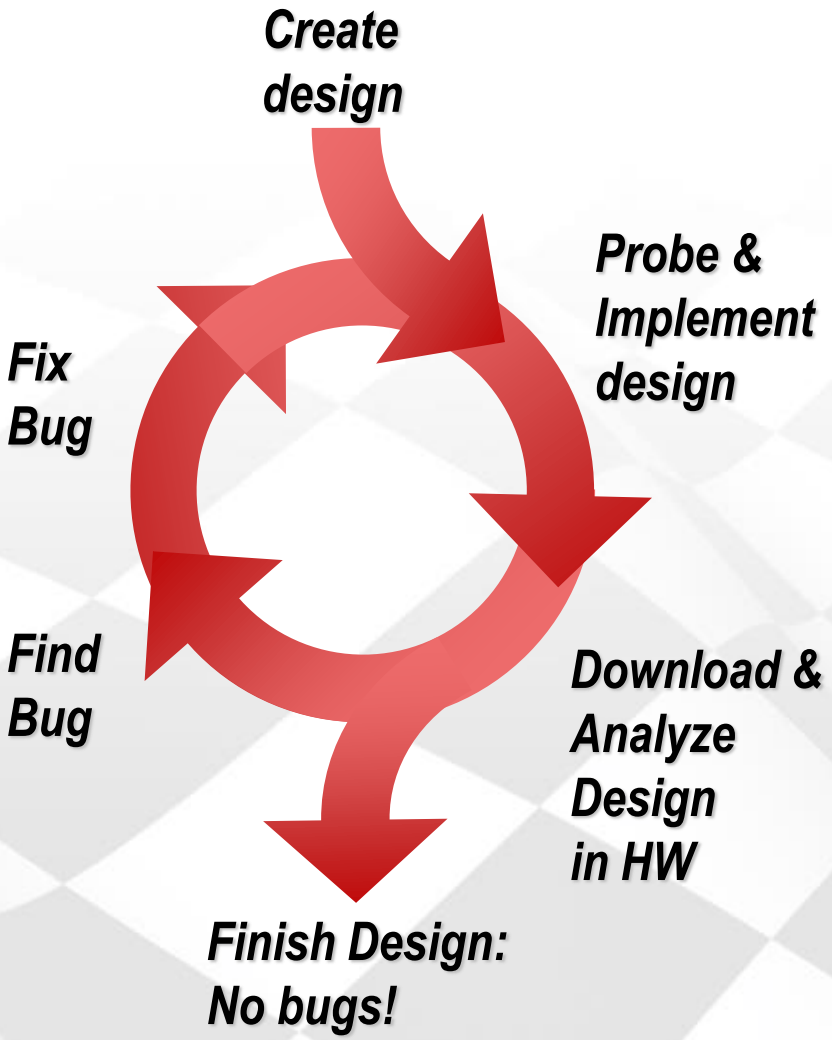
- **Engineers are trained to solve problems logically**
  - Break a problem into smaller parts
  - Simplify by reducing variables & variation
  - Make a prediction, verify the results
  - Plan how and where to debug early in the design cycles
- **FPGA Design is an Iterative Process**
- **Debugging is an iterative process**

  1) Probe: Adding/modify debug probes

  2) Implement: Compile design w/probes

  3) Analyze: Look for bugs using probes

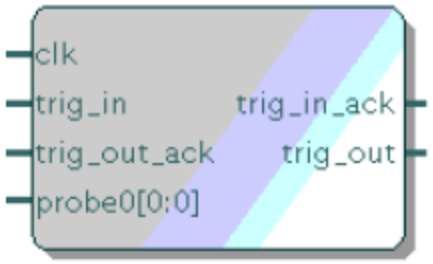
  4) Fix: Fix any bugs, repeat as necessary

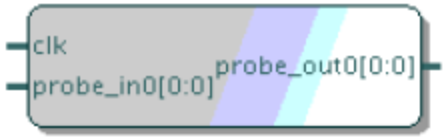


The reconfigurable nature of FPGAs enables the iterative debug process

Vivado Logic Debug IP

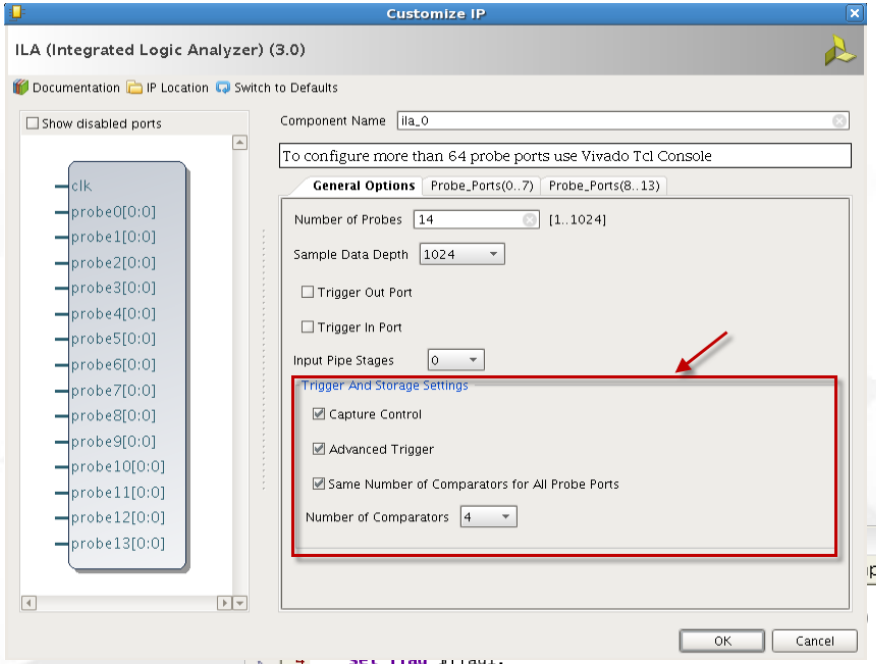
- **ILA 3.0**
  - Integrated Logic Analyzer debug IP core
  - Netlist insertion support
  - HDL instantiation support
  - Added support for advanced trigger and capture features
- **VIO 3.0**
  - Vivado native Virtual Input / Output
  - HDL instantiation support
- **JTAG2AXI 1.0**
  - JTAG to AXI Master debug IP core
  - HDL instantiation support
  - IP Integrator block designs and HDL designs





ILA 3.0 Advanced Trigger Features

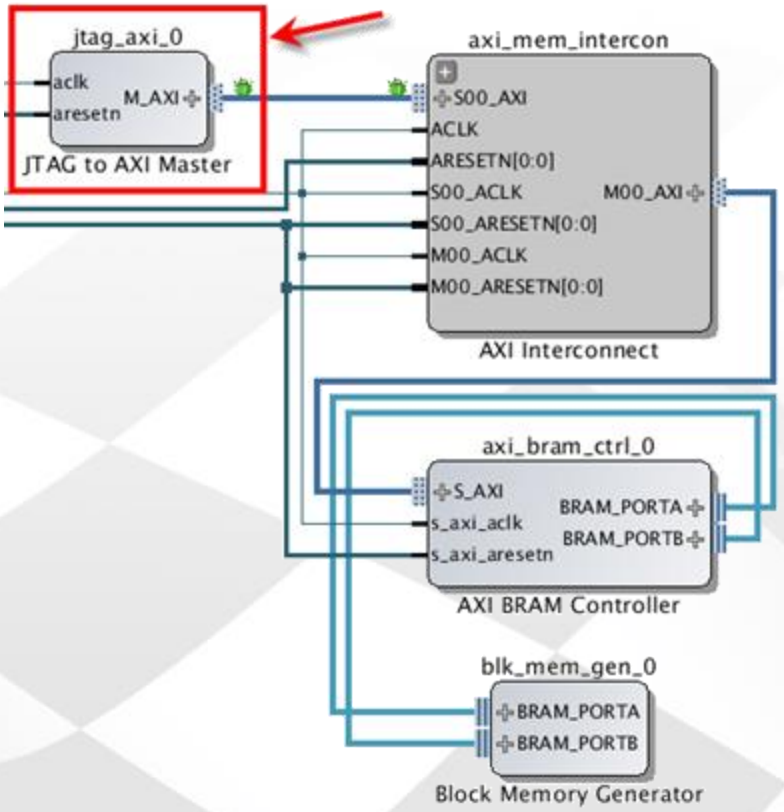
- **Advanced trigger state machine:**
  - Fully programmable at run-time
  - 3-way branching per state
  - Up to 16 states
  - Up to four comparators per PROBE input
  - Four programmable counters (reset, increment, conditional compare)
  - Four programmable flags (set, clear, monitor in GUI)




```
    goto wait_for_second_button_push;
  elseif (vio_slice1_fb == 1'bR) then
    increment_counter $counter0;
    goto wait_for_first_button_push_3_times;
  else
    goto wait_for_first_button_push_3_times;
  endif

state wait_for_second_button_push:
  if (vio_slice1a_fb == 1'bR) then
    set_flag $flag2;
    goto trigger_on_counter_value;
  else
    goto wait_for_second_button_push;
  endif

state trigger_on_counter_value:
  if (fast_cnt_count == 32'hffff_ffff) then
    trigger;
  else
    goto trigger_on_counter_value;
  endif
```

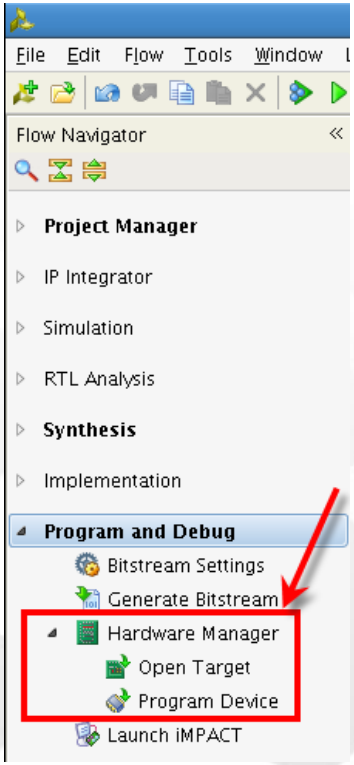
JTAG to AXI Master

- **Interact with AXI-based system without writing microprocessor code**
- **Connects to AXI or AXI-Lite interfaces**
- **Can be used in both IP Integrator block designs and HDL designs**
- **Vivado run-time Tcl commands to create and run AXI transactions**
  - help *hw_axi*



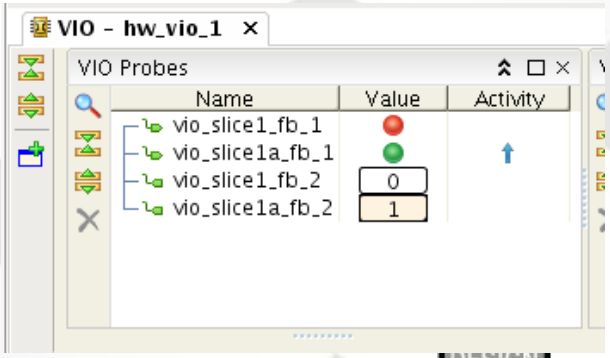
Hardware Manager & GUI Ease of Use

- **Hardware Manager and related tasks in Flow Navigator improve flow and highlight next steps**
  - Not another tool that needs to be opened



- **Simplified Debug Probes window for ease of finding probe information**

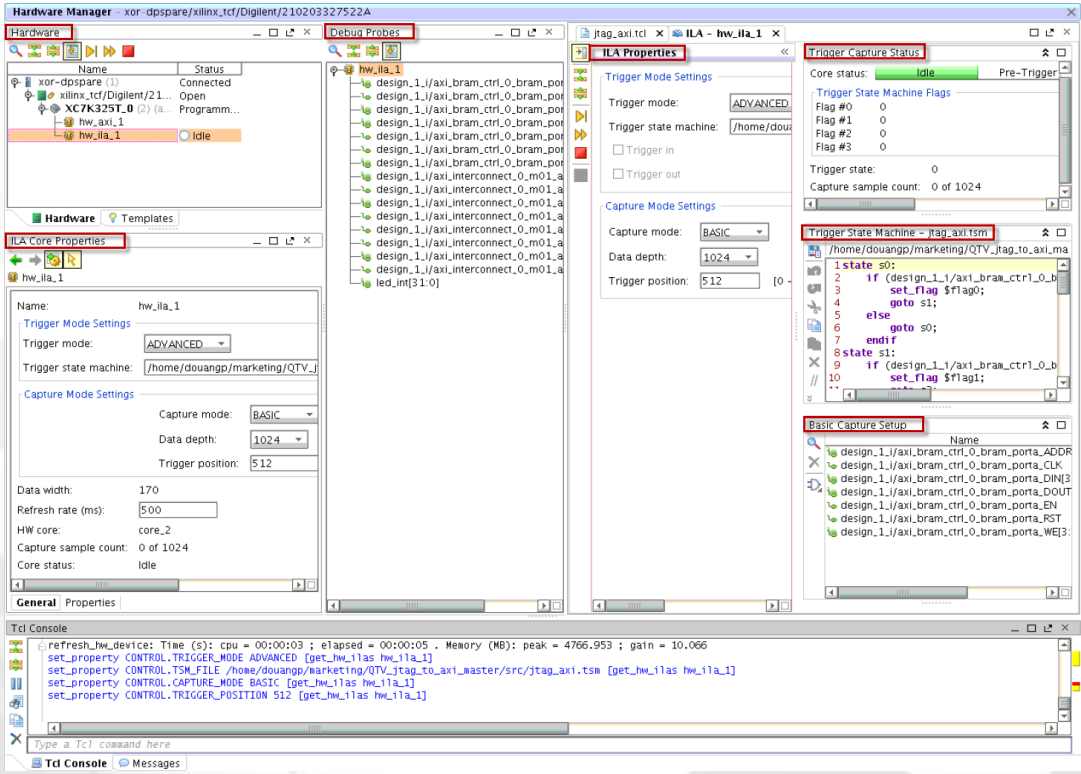
- **ILA and VIO "dashboards" in GUI provide better organization and customization**



Benefits of Vivado Logic Debug

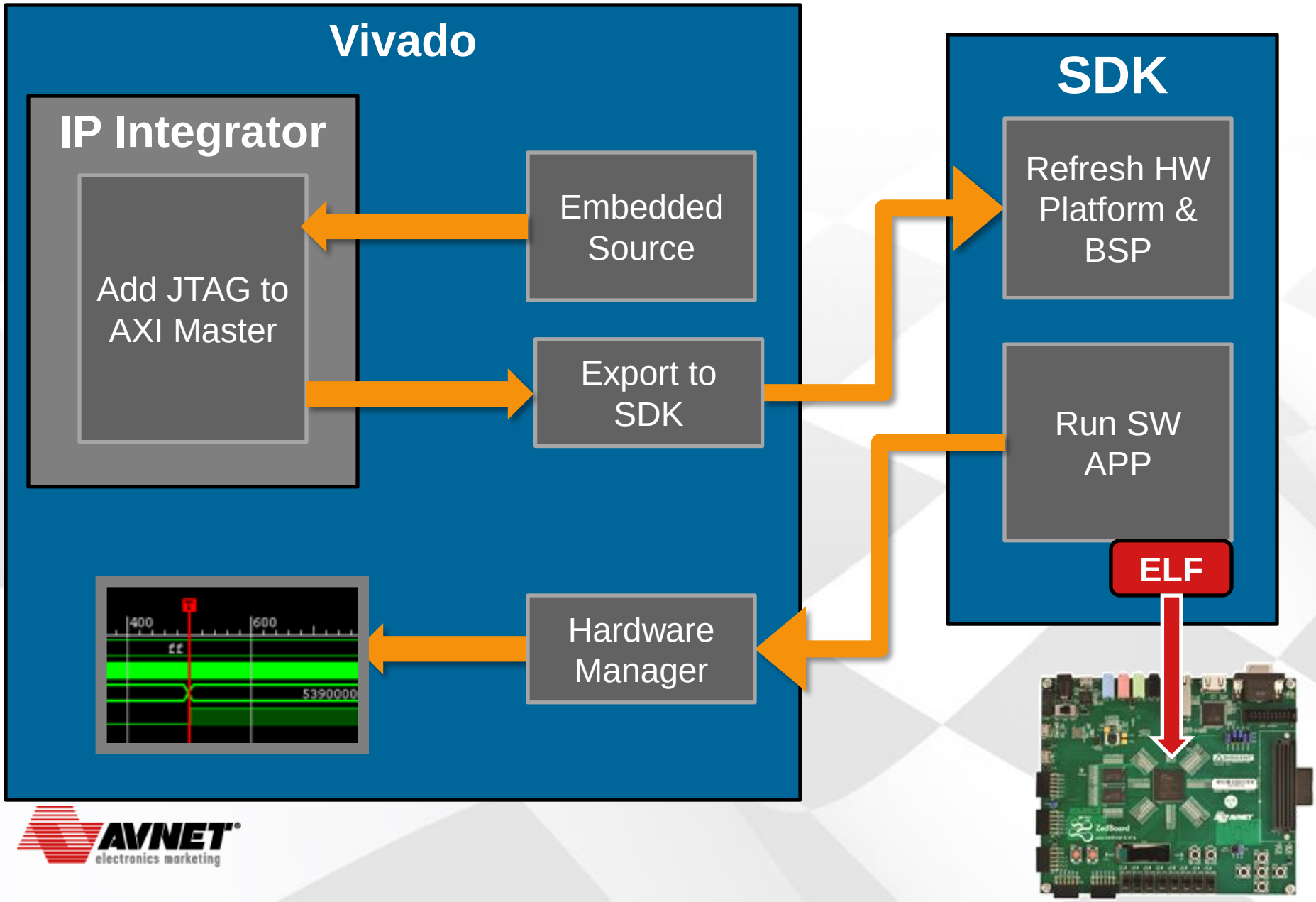
Key Benefits Summary

- **Runtime software interface for interacting with debug IP cores**
- **Robust Core insertion flow**
- **Advanced trigger and capture control**
  - Detect complex trigger events
  - Control what data to capture by ILA core
- **Added runtime access interface for JTAG to AXI Master IP**
  - Provides quick read and write transactions to an AXI-based system
- **Hardware Manager & GUI Ease of Use**
- **Complete Tcl scripting of run-time functions**

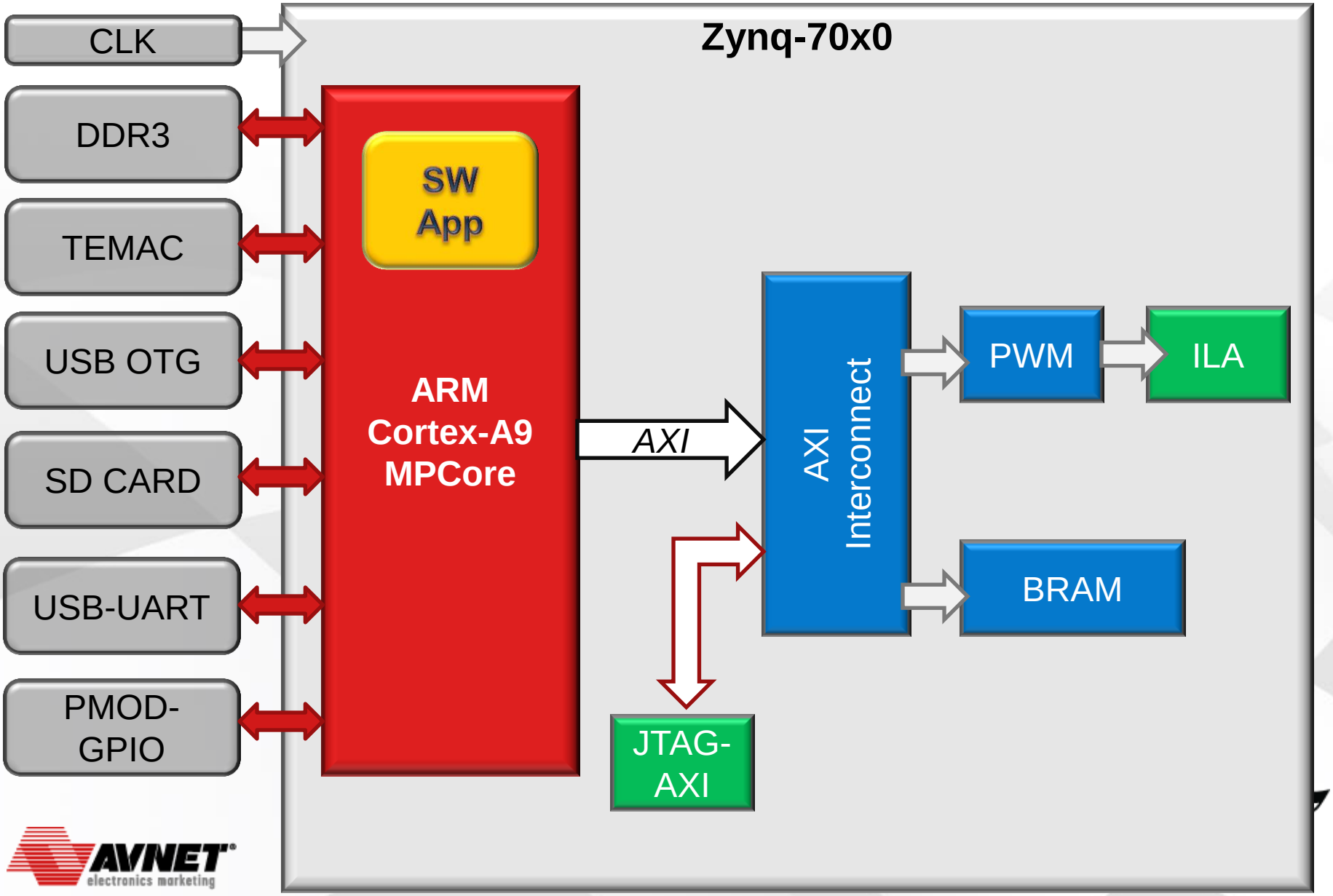


UG936 – Vivado Design Suite Tutorial Programming and Debugging

Lab 8 – Add PL Peripherals



Lab 8 – Add JTAG to AXI Master



Questions

- ***What function is called when the Interrupt is detected?***
  - PWMIsr

    /* Connect the interrupt handler that will be called when an
    * interrupt occurs for the device. */
    result = XScuGic_Connect(IntcInstancePtr, INTC_PWM_INTERRUPT_ID,
    (Xil_ExceptionHandler) **PWMIsr**, 0);
- ***What does the PWMIsr function do?***
  - Resets the Brightness value and writes it out.
- ***In the PWMIsr function, what does it do to the brightness value?***
  - Zero
- ***What is the INTC_PWM_INTERRUPT_ID? Where did this number come from?***
  - *This is defined as:*
    PAR_FABRIC_PWM_W_INT_0_INTERRUPT_OUT_INTR
  - *This is assigned to 91 in xparameters.*h?

    /* Definitions for Fabric interrupts connected to ps7_scugic_0 */
    #define XPAR_FABRIC_PWM_W_INT_0_INTERRUPT_OUT_INTR 91

Questions

- ***Can you detect how long it takes the processor to handle the interrupt?***
  - Technically yes, but not with this design. Why? Because our capture depth is only 1024 clock cycles, and the trigger does not reset in that time. If the capture depth was increased then it could be captured.

- ***How would you delete one of the AXI transaction commands?***
  - delete_hw_axi_txn [get_hw_axi_txns write_txn]

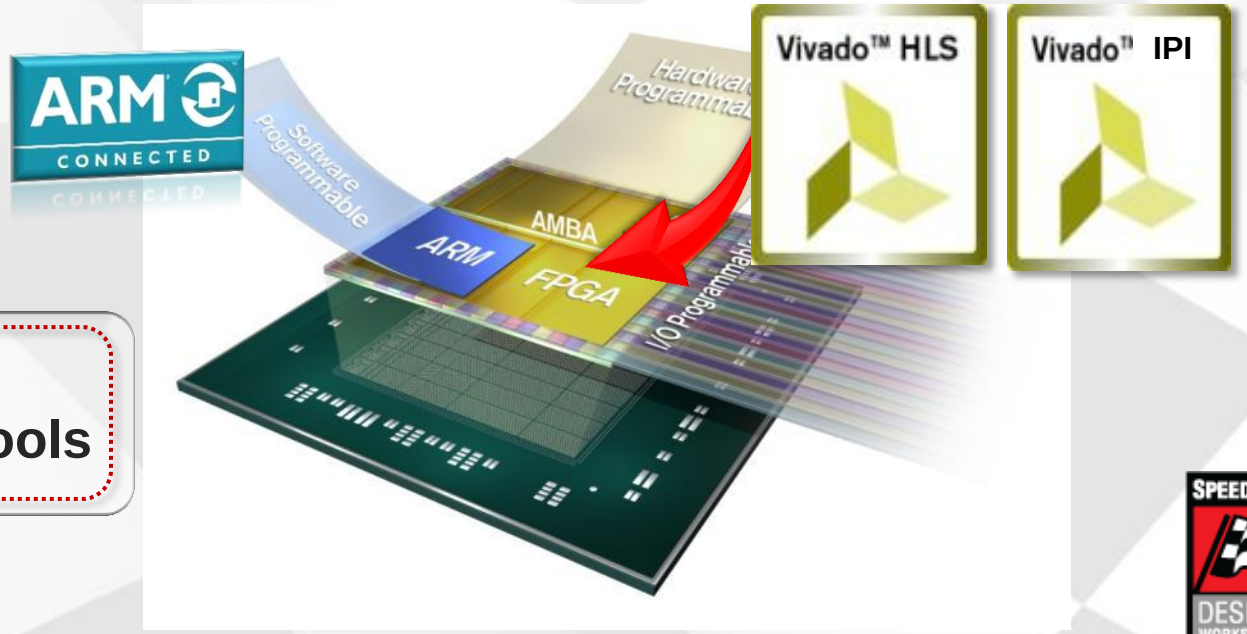
Agenda

Lecture	Description	Labs
Chapter 1	Zynq Overview	
Chapter 2	Xilinx Embedded Tool Flow	Lab 1
Chapter 3	Zynq Processor Overview	Lab 2
Chapter 4	Zynq Peripherals	Lab 3
Chapter 5	The Power of TCL	Lab 4
Chapter 6	Adding Programmable Logic Peripherals	Lab 5
Chapter 7	Zynq Processing System DMA Controller	Lab 6
Chapter 8	Creating Custom IP	Lab 7
Chapter 9	Vivado Hardware Manager	Lab 8
Chapter 10	Next Steps	

All Programmable Abstractions & Automation

Abstraction

Automation



Next Generation
High-level Design Tools

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Widest Selection of Software Tools

Software Development Tools

XILINX SDK

ARM

YOKOGAWA

LAUTERBACH DEVELOPMENT TOOLS

KMC Kyoto Microcomputer Co.,Ltd.

mentor embedded

Microsoft Visual Studio

abatron

Computex

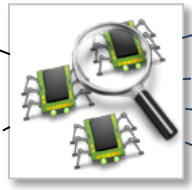
No Cost

Commercial Offerings

Heterogeneous Multi-Core Debugger

Cortex-A9

Cortex-A9



MicroBlaze 32

MicroBlaze 32

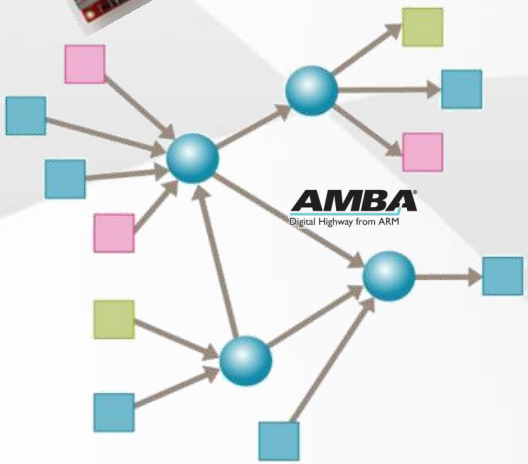
MicroBlaze 32

MicroBlaze 32

Scalable offering from Xilinx and partners to best fit your needs

SPEEDWAY DESIGN WORKSHOPS

Largest Portfolio of IP, Design Kits and Ref. Designs



- **Widest Selection of Development Platforms and SOMs**
  - Over 20 boards and SOM from Xilinx and partners
  - Embedded, intelligent control, video signal processing
  - 100+ FMC daughter cards
- **Ready to Use Segment Solutions**
  - Boards, IPs, reference designs, app notes, OS and drivers
  - Automotive, A&D, broadcast, consumer, industrial, medical, wired and wireless
- **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 electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Where to go for additional training?

Avnet Speedway Courses

- **PetaLinux for the Zynq-7000 AP SoC**
  - Hands-on experience in building the Linux environment
  - Introduces embedded Linux components, use of open-source components, environment configurations, network components, and debugging/profiling options
- **Designing Accelerators for the Zynq-7000 AP SoC**
  - Create custom hardware accelerators in the PL
  - Learn profiling techniques to identify acceleration opportunities
  - Driver and software application development
- **Introduction to Vivado**
  - Create new projects, create and import IP cores
  - Set basic clocking and I/O timing constraints
  - Analyze implementation results through Vivado Analysis features and reports

www.em.avnet.com/xilinxspeedways

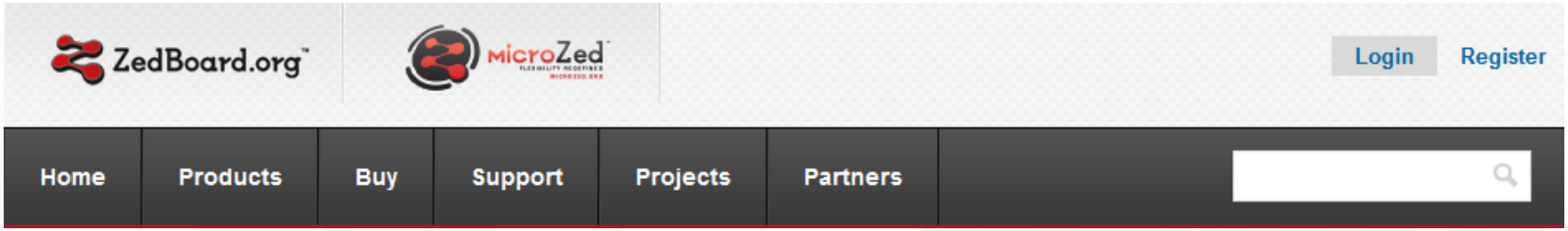
Even More Training…

Zynq-7000 All Programmable SoC Product Training		
Essentials of Microprocessors	**Level: 1**	**Duration: 2 days**
Learn what makes microprocessors tick! This class offers insights into all major aspects of microprocessors, from registers through coprocessors and everything in between. Differences between RISC and CISC architectures are explored as well as the concept of interrupts. A generic microprocessor is programmed and run in simulation to reinforce the principles learned in the lecture modules. The student will leave the class well prepared for the Xilinx Zynq-7000 All Programmable SoC training curriculum.		
C-Programming with Xilinx SDK	**Level: 1**	**Duration: 2 days**
This course is broken into a day of C language review, including variable naming, usage, and modifiers as well as an introduction to the Software Development Kit (SDK) environment, an explanation of the use of the preprocessors, program control, and proper use of functions. The second day consists of common issues and techniques employed by embedded programmers in the Xilinx SDK environment.		
How to Design Xilinx Embedded Systems in One Day	**Level: 2**	**Duration: 1 day**
The workshop introduces you to fundamental embedded design concepts and techniques for implementation in Xilinx FPGAs. The focus is on fundamental aspects of Xilinx embedded tools, IP, and the Embedded Targeted Reference Design (TRD). Design examples and labs are drawn from the Embedded TRD.		
Introduction to the Zynq-7000 All Programmable SoC Architecture	**Level: 3**	**Duration: 1 day**
This course provides hardware and firmware engineers with the knowledge to effectively utilize a Zynq-7000 All Programmable SoC. It covers the architecture of the ARM® Cortex™-A9 processor-based processing system (PS) and the integration of programmable logic (PL).		
Zynq-7000 All Programmable SoC Architecture	**Level: 3**	**Duration: 2 days**
The Xilinx Zynq-7000 All Programmable SoC provides a new level of system design capabilities. This course provides experienced system architects with the knowledge to effectively architect a Zynq-7000 All Programmable SoC. This course presents the features and benefits of the Zynq-7000 All Programmable SoC architecture for making decisions on architecting a Zynq-7000 All Programmable SoC project. It covers the architecture of the ARM® Cortex™-A9 processor-based processing system (PS) and the integration of programmable logic (PL) at a sufficiently deep level that a system designer can successfully and effectively utilize the Zynq-7000 All Programmable SoC.		
Embedded Systems Development	**Level: 3**	**Duration: 2 days**
The Xilinx Zynq-7000 All Programmable SoC provides a new level of system design capabilities. This course brings experienced FPGA designers up to speed on developing embedded systems using the Embedded Development Kit (EDK). The features and capabilities of the Zynq-7000 All Programmable SoC as well as concepts, tools, and techniques are included in the lectures and labs. The hands-on labs provide students with experience designing, expanding, and modifying an embedded system, including adding and simulating a custom AXI-based peripheral. Additionally, the features and capabilities of the Xilinx MicroBlaze™ soft processor are also included in the lectures and labs.		
Embedded Systems Software Development	**Level: 3**	**Duration: 2 days**
This two-day course introduces you to software design and development for the Xilinx Zynq-7000 All Programmable SoC using the Xilinx Software Development Kit (SDK). You will learn the concepts, tools, and techniques required for the software phase of the design cycle. Topics are comprehensive, covering the design and implementation of the board support package (BSP) for resource access and management of the Xilinx Standalone library. Major topics include device driver use and custom development and user application debugging and integration. Practical implementation tips and best practices are also provided throughout to enable you to make good design decisions and keep your design cycles to a minimum. You will have enough practical information to start developing software applications for the ARM® Cortex™-A9 and MicroBlaze™ processors.		
Advanced Features and Techniques of Embedded Systems Design	**Level: 4**	**Duration: 2 days**
Advanced Features and Techniques of Embedded Systems Design provides embedded systems developers the necessary skills to develop complex embedded systems and enables them to improve their designs by using the tools available in the Embedded Development Kit (EDK). This course also helps developers understand and utilize advanced components of embedded systems design for architecting a complex system in the Zynq-7000 All Programmable SoC or Microblaze™ soft processor. This course builds on the skills gained in the Embedded Systems Design course. Labs provide hands-on experience with developing, debugging, and simulating an embedded system. Utilizing memory resources and implementing high-performance DMA are also covered. Labs use demo boards in which designs are downloaded and verified.		
Advanced Embedded Systems Software	**Level: 4**	**Duration: 1 day**
This course will help software engineers fully utilize the components available in the Zynq-7000 All Programmable SoC. This course covers advanced Zynq-7000 All Programmable SoC topics for the software engineer, including advanced boot methodology, the NEON co-processor, programming PS system-level function control registers, the general interrupt controller, the DMA, Ethernet, and USB controllers, and the various low-speed peripherals included in the Zynq-7000 All Programmable SoC processing system.		
Embedded Design with PetaLinux SDK	**Level: 4**	**Duration: 2 days**
This intermediate-level, two-day course provides embedded systems developers with experience in creating an embedded PetaLinux SDK operating system on a Xilinx MicroBlaze processor development board. The course offers students hands-on experience on building the environment and booting the system using a basic, single-processor System on Chip (SoC) design with PetaLinux SDK on the MicroBlaze processor. This course also introduces embedded Linux components, use of open-source components, environment configurations, network components, and debugging/profiling options for embedded Linux platforms. The primary focus is on embedded Linux development in conjunction with the Xilinx tool flow.		

Getting Help and Support

- **Visit ZedBoard.org or MicroZed.org**

  http://www.zedboard.org | http://www.microzed.org



- www.support.xilinx.com
  - 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 Engineering Services if you are interested in customizing a kit or SOM**

  customize@avnet.com

- **Contact your local Avnet/Silica FAE**

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Course Objectives

- **Understand the Zynq-7000 All Programmable SoC development flow with Vivado's IP Integrator**
- **Introduce the new Extensible Dual ARM Cortex™-A9 Processors Cores**
  - Explore Robust AXI Peripheral Set
- **Utilize the Xilinx embedded systems tools to**
  - Design a Zynq AP SoC System
  - Add Xilinx IP as well as custom IP
  - Run Software Applications to test IP
  - Debug an Embedded System

AVNET electronics marketing

SPEEDWAY DESIGN WORKSHOPS

Thank-You!

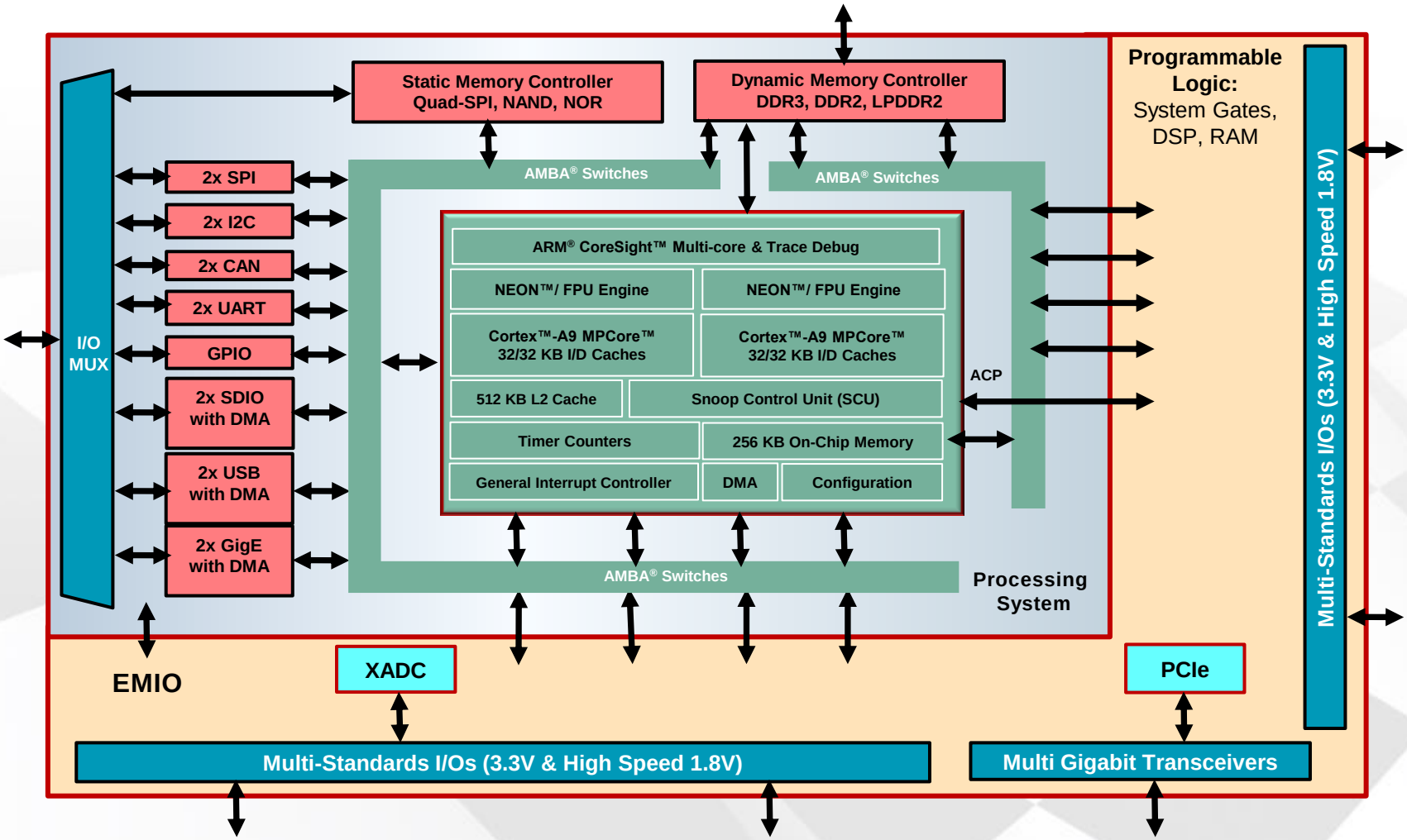
- **Please Complete the Digital Survey**

  www.em.avnet.com/speedwaysurvey

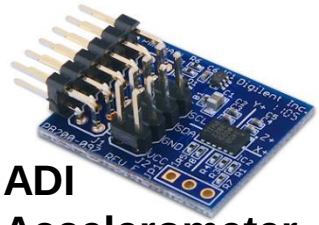
Appendix

Miscellaneous

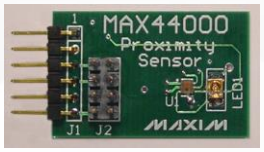
Zynq-7000 AP SoC Architecture Overview



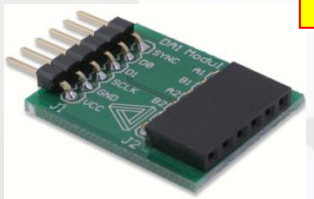
Pmod Expansion on ZedBoard



ADI Accelerometer

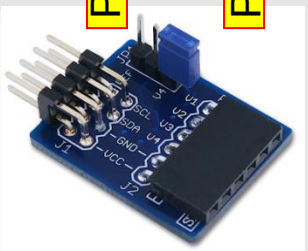


Maxim Proximity Detector



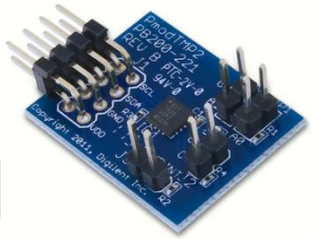
ADI Quad 8-bit DAC

Maxim Real-time Clock



ADI 4-channel 12-bit ADC

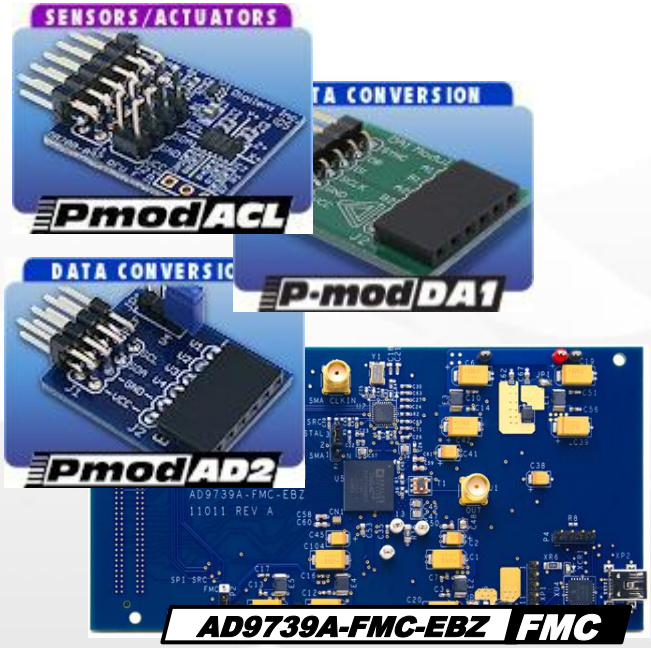
Maxim 16-bit Delta-Sigma ADC



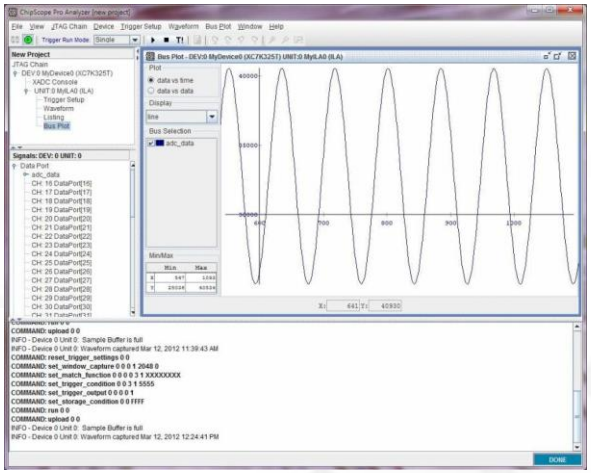
ADI 16-bit Temp Sensor

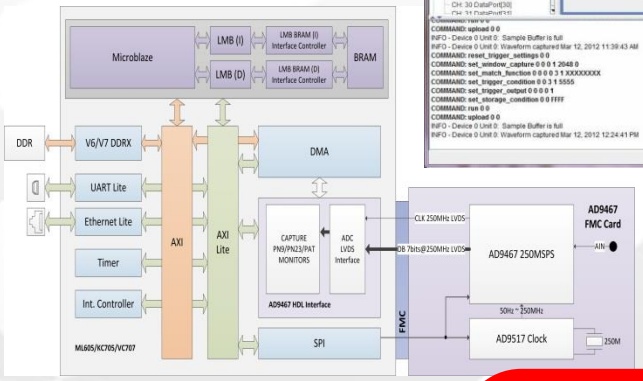
Maxim 16-bit DAC

Analog Devices Evaluation Boards



Reference Designs with Immediate Results





ANALOG DEVICES

Pmods

- 12-, 16-, and 24-bit ADCs
- 8-, 12-, and 16-bit DAC's
- 3-axis Accelerometer
- Gyroscope
- Digital Potentiometer
- I2C Temp Sensor

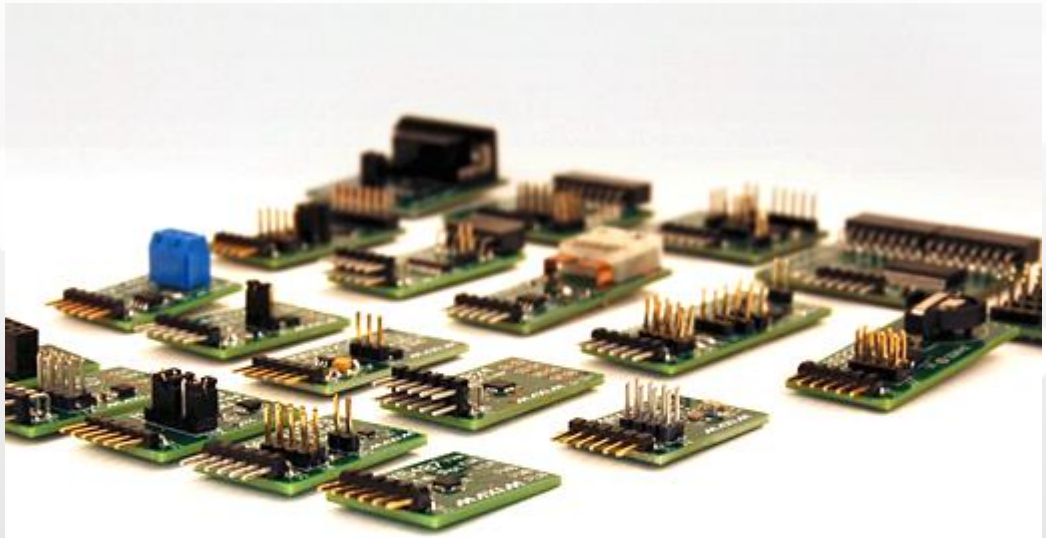
FMCs

- Hi-speed 14-bit DAC @ 2.5GSPS
- High-speed 16-bit ADC @ 250MSPS
- RF Transmit and Receive channels
- Interposer for SDP-B

AMS

- Instrumentation Amplifier Board
- RF RMS Detector
- RF LogAmp Detector

MAXIM Plug-In Peripheral Modules



Pmod™ compatible

- **Over 15 Popular Functions**
  - Data conversion, Isolation, IO Expansion, Current Sense, Current Limiting, Temp Sense, Light Sense, RS232, RS485, and more
- **Complete Software, FPGA code and documentation available on-line**
- **All modules follow Digilent's Pmod pin-out conventions**

$89.95

Pmod™ is a trademark of Digilent, Inc