HETEROGENEOUS COMPUTING ARCHITECTURE

DR. SHARAD SINHA

COMPUTER SCIENCE AND ENGINEERING,

INDIAN INSTITUTE OF TECHNOLOGY GOA (IIT GOA)

SHARAD@IITGOA.AC.IN

COMPUTER ARCHITECTURE WINTER SCHOOL (CAWS)

PES UNIVERSITY, BANGALORE

23 DECEMBER 2020

OUTLINE

- What is Heterogeneous Computing?
- Heterogeneous Computing Architectures
  - Heterogeneous Multi-Core
  - CPU-FPGA Systems
- Conclusion

WHAT IS HETEROGENEOUS COMPUTING?

- Let's first understand **Homogeneous Computing**
- Any computing system makes use of "**Processing Elements (PEs)**"
  - Processing elements could be:
    - Microprocessors (CPU)
    - Graphics Processing Units (GPU)
    - Field Programmable Gate Array (FPGA)
    - Accelerators (Custom Computing Elements)

WHAT IS HETEROGENEOUS COMPUTING?

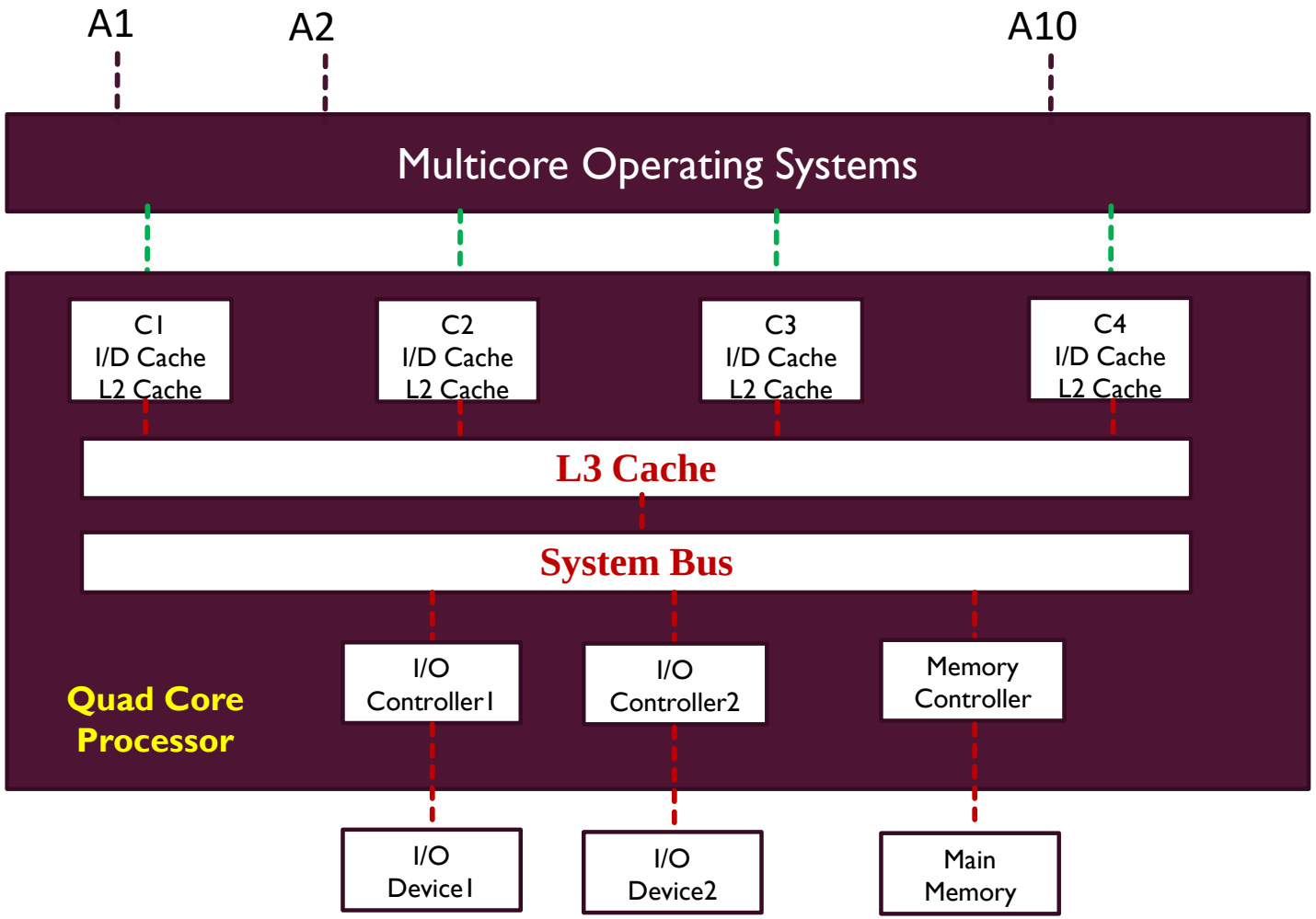
- When a computing system makes use of **ONLY ONE TYPE OF PE**, i.e. all PEs are identical in characteristics, the system is called **Homogeneous Computing System**
- When a computing system makes use of **MULTIPLE TYPES OF PE** i.e. PEs are not identical in characteristics, the system in called **Heterogeneous Computing System**

EXAMPLE: HOMOGENOUS MULTICORE PROCESSOR

- A **Homogeneous Multicore Processor** has all processor cores of the same type i.e. identical in Instruction Set Architecture
- Also called Symmetric Multicore Processor
- Homogeneous Multicore Processors typically run a single multicore operating system

HOMOGENEOUS MULTICORE PROCESSOR

- Conceptual Diagram
- 10 applications running on a quad-core homogeneous multicore processor



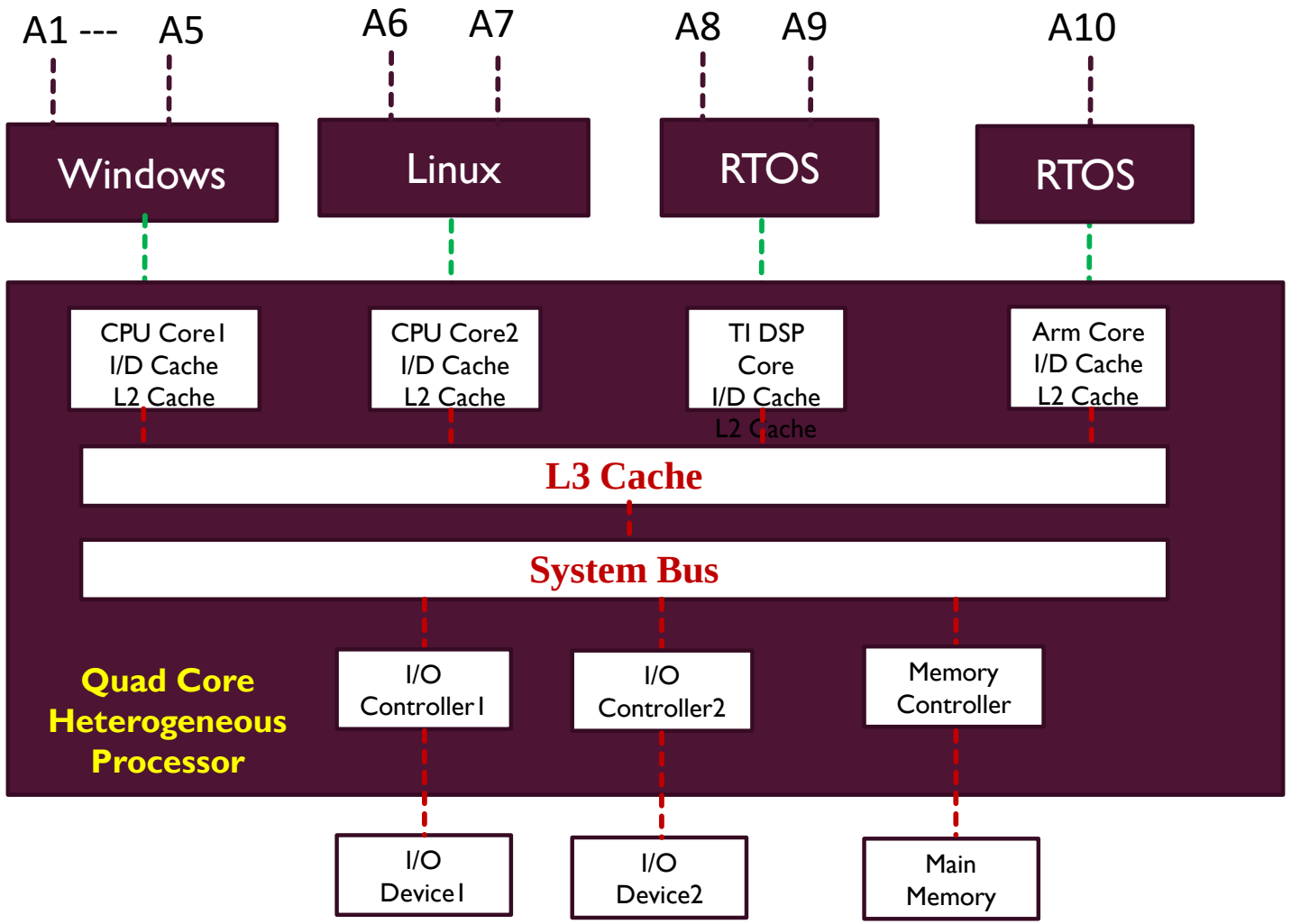
HOMOGENEOUS MULTICORE PROCESSOR

- Example of homogeneous multicore processors:
  - Intel Core i3, i5, i7, i9 etc.
  - IBM POWER4, 5, 6, 7, 8, 9 etc.
  - AMD Opteron 2, 4, 6, 8 etc.

- Example of homogeneous multicore operating system:
  - Most modern operating systems support multicore processors (e.g: Windows and Linux on your laptop, desktop)

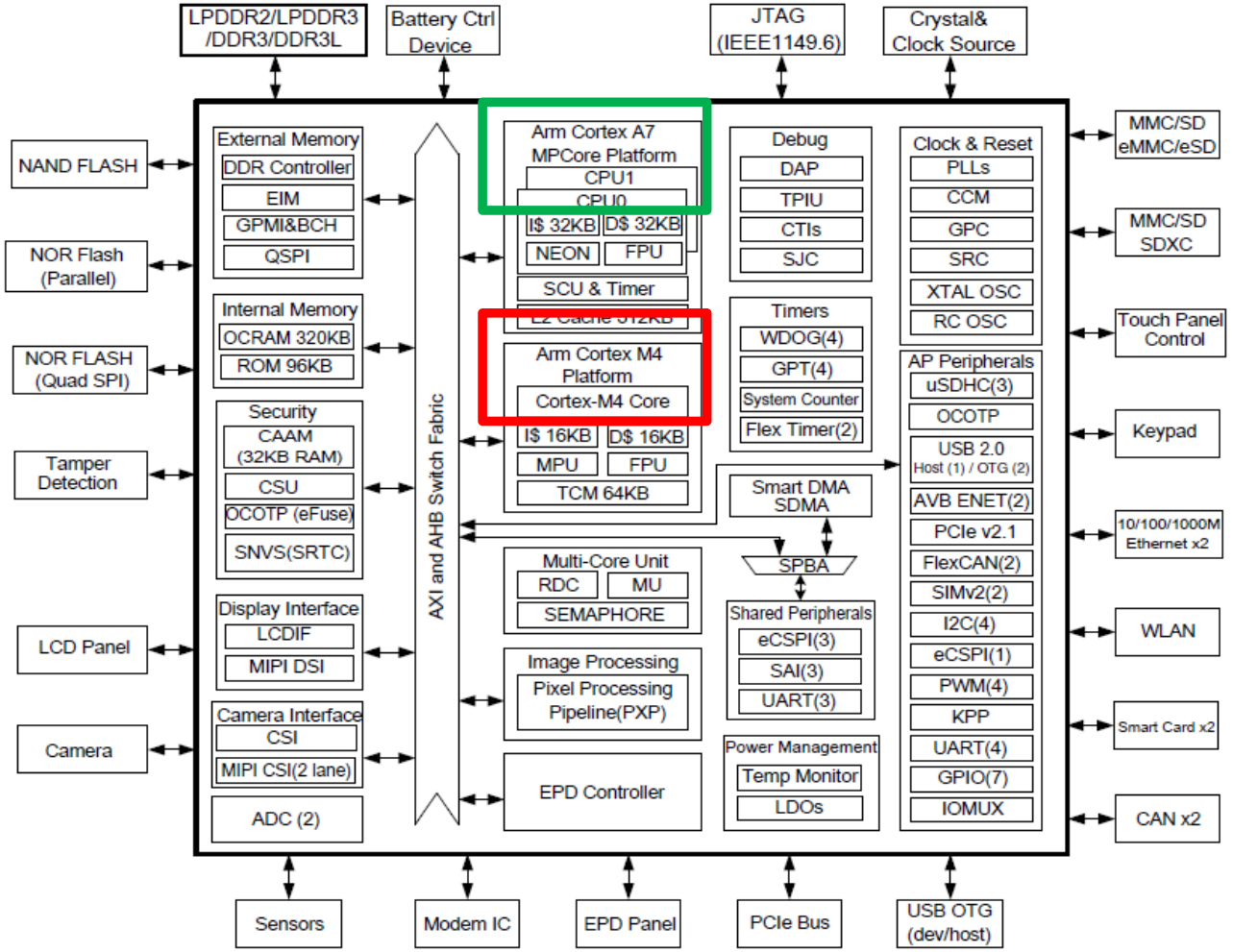
HETEROGENEOUS MULTICORE PROCESSOR

- Conceptual Diagram
- 10 Applications distributed across 4 different processor cores
- CPU Core 1 and Core 2 have different ISA
- TI DSP is a Digital Signal Processor from Texas Instruments
- Arm Core is an Arm processor core



HETEROGENEOUS MULTICORE PROCESSOR

- Example of heterogeneous multicore processors:
  - i.MX7 Dual Application Processor – NXP Semiconductors
- Different cores can run different OS



CPU-FPGA Heterogeneous Computing System

DR. SHARAD SINHA, IIT GOA, 22 DECEMBER 2020

FPGA AND ITS ROLE IN HETEROGENEOUS COMPUTING

- CPU-FPGA systems have existed for quite a few years with increasing growth in the last decade
- FPGA can be used as "**Hardware Accelerator**" i.e.:
  - Offload the compute intensive part of an algorithm to FPGA to reduce execution time of the application

EARLY CPU-FPGA SYSTEMS

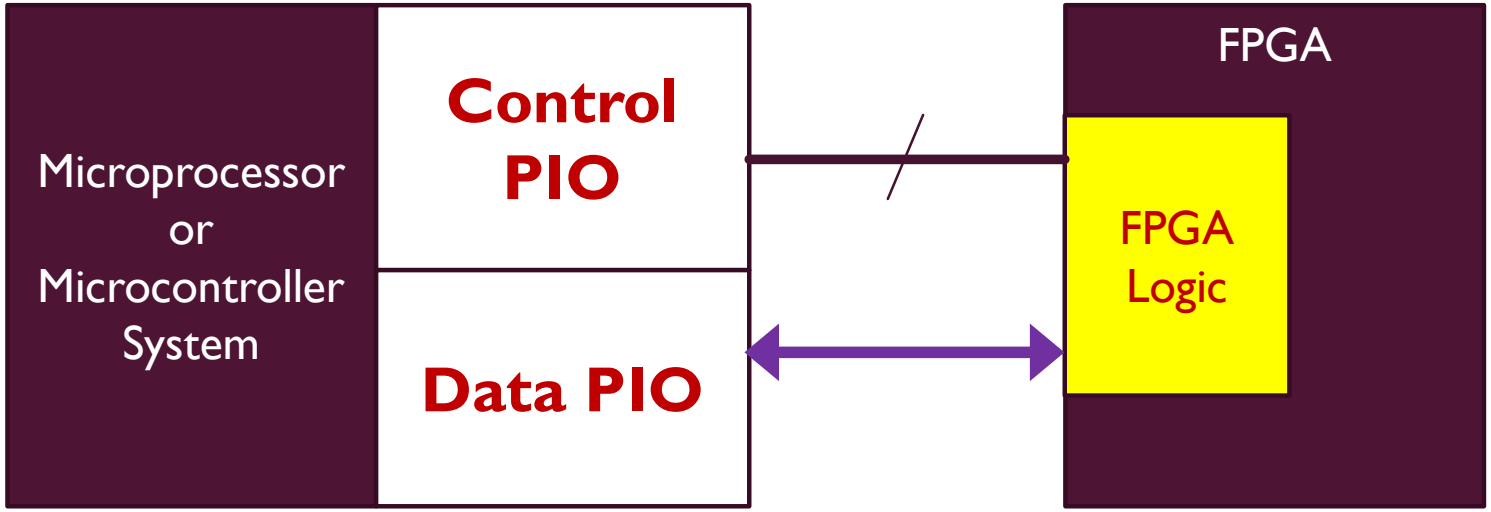
- Early CPU-FPGA systems consisted of separate CPU and FPGA chip on a PCB connected to each other i.e. they were external to each other.
- These days, one can find integrated CPU-FPGA systems, also called "SoC FPGA" or "System on Chip FPGA"

DISCRETE CPU-FPGA SYSTEMS

- The main goal is to ensure proper data transfer between CPU and FPGA
- Several schemes exist:
  - Programmable IO (PIO) Interface (means IO port can be programmed as Input or Output)
  - External Bus Interface (EBI)
  - Dedicated Bus Interface built in CPU for Interfacing with FPGA

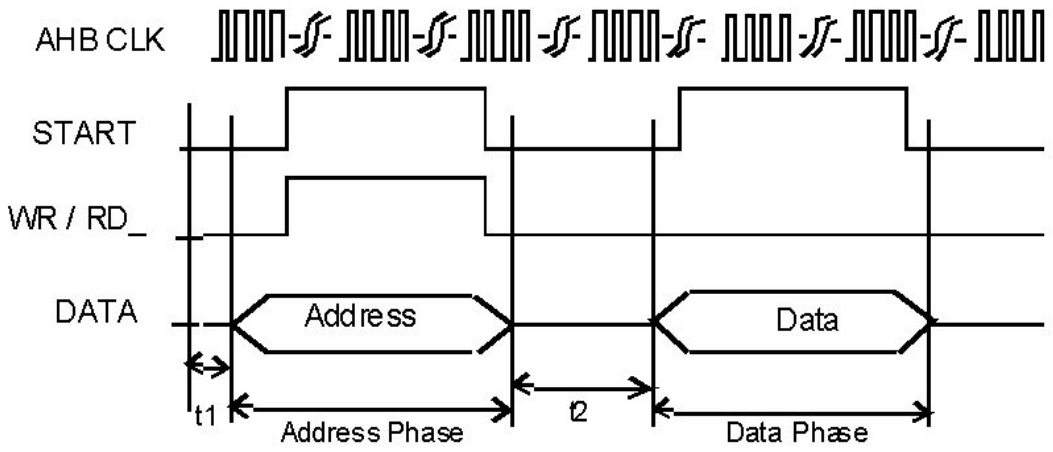
DISCRETE CPU-FPGA SYSTEMS (PIO INTERFACE)

- Programmable IO Interface available in CPU/Microcontroller

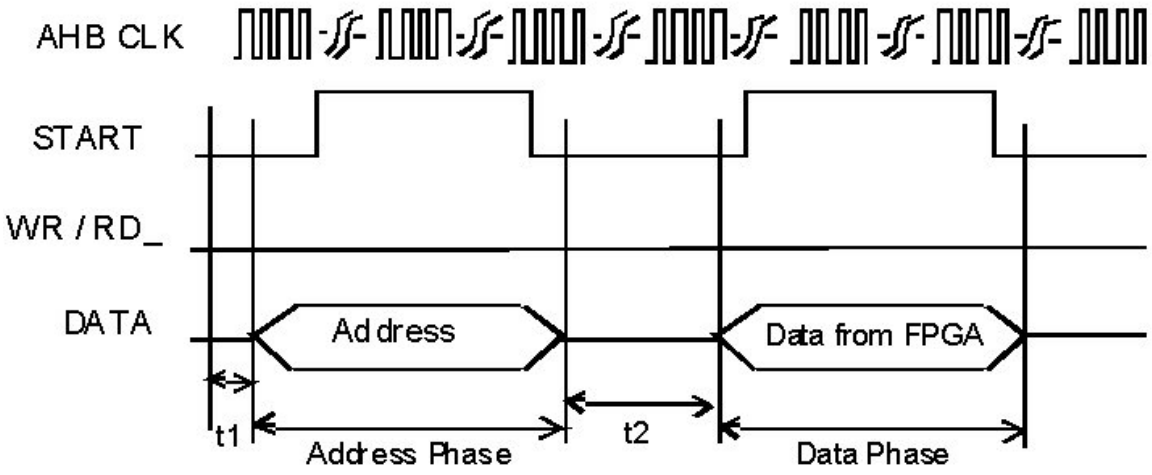


- Control Signals like from Control PIO. Each signal is unidirectional:
  - RD/WR
  - Start
- Signals from/to Data PIO:
  - Address
  - Data

CPU-FPGA (PIO INTERFACE)



CPU to FPGA Write

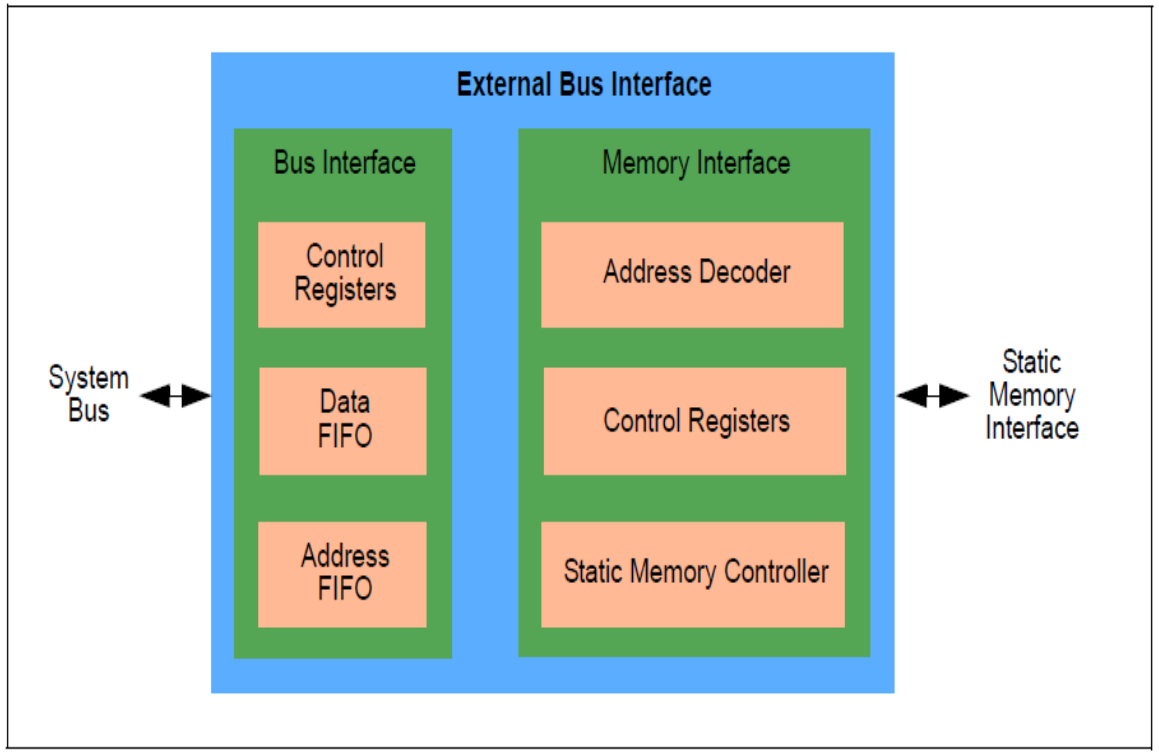


Read from FPGA

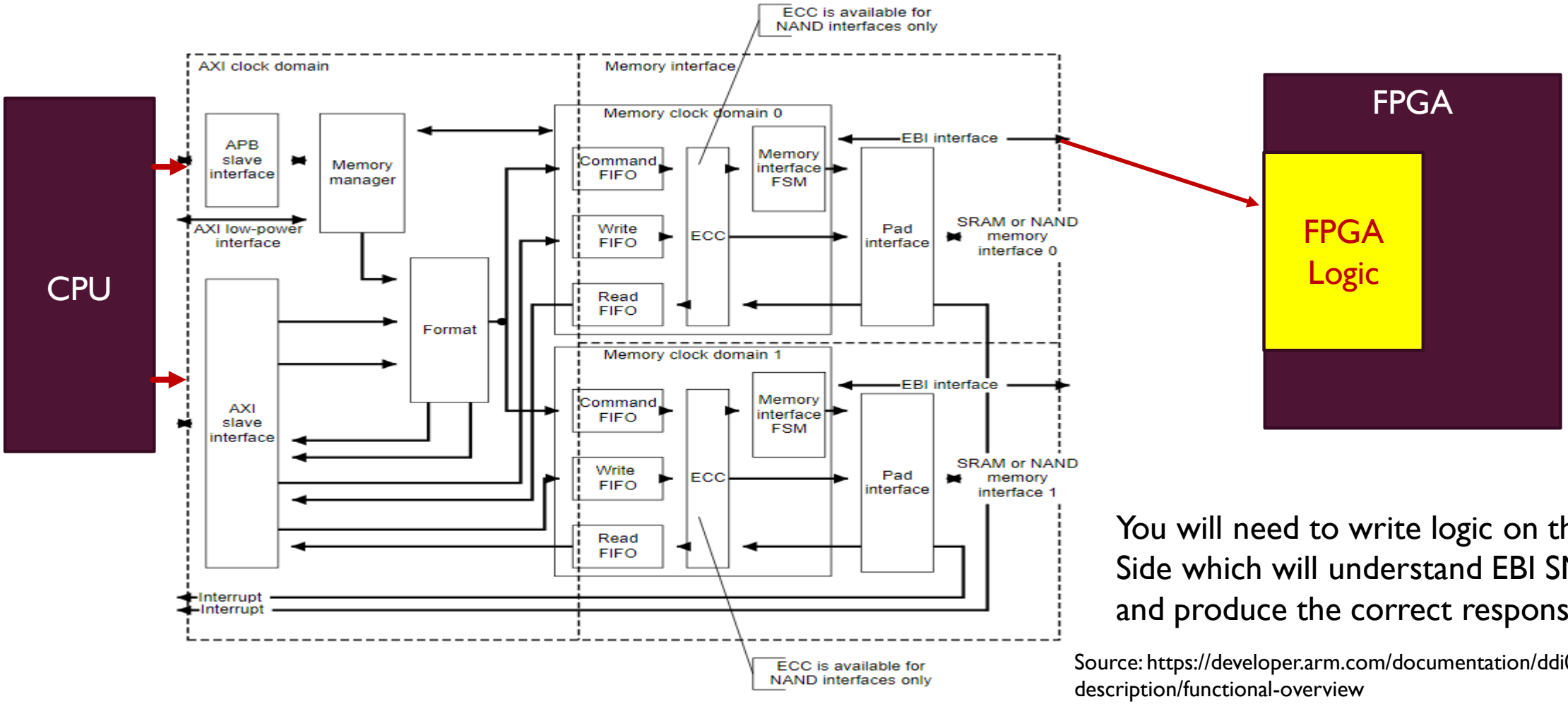
Fig. source: https://www.eetimes.com/how-to-interface-fpgas-to-microcontrollers/

DISCRETE CPU-FPGA SYSTEMS (EBI INTERFACE)

- Many microcontrollers/CPUs have EBI module for transfer of data between external devices and microcontrollers/microprocessors
- Example of an EBI Interface from PIC microcontroller. The static memory interface is used to connect to FPGA
- EBIs are common on Arm processors



ARM EBI (SMC) FOR ARM MCUS



You will need to write logic on the FPGA Side which will understand EBI SMC timing and produce the correct response

Source: https://developer.arm.com/documentation/ddi0380/h/functional-description/functional-overview

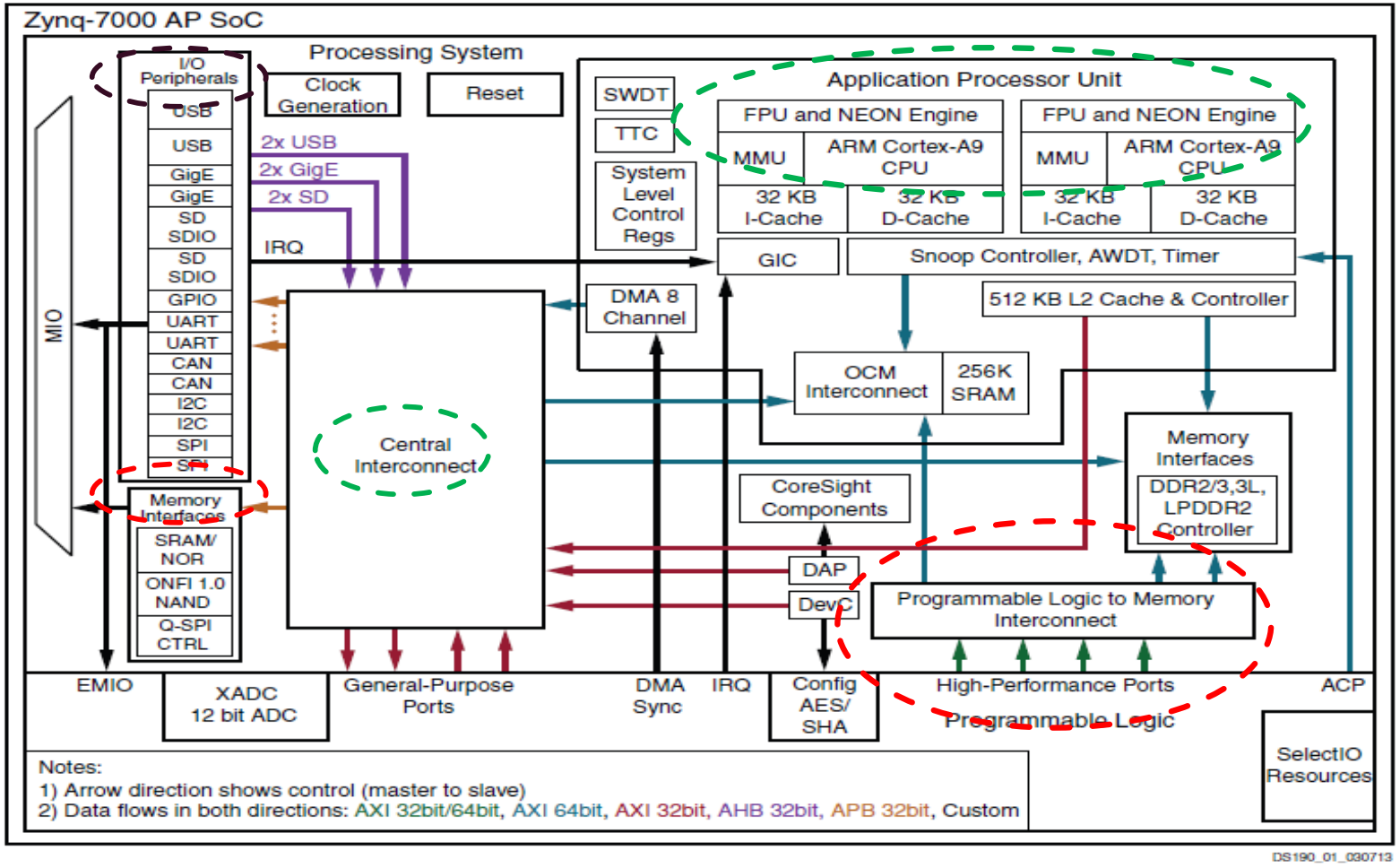
SoC FPGA

DR. SHARAD SINHA, IIT GOA, 22 DECEMBER 2020

SYSTEM ON CHIP FPGA

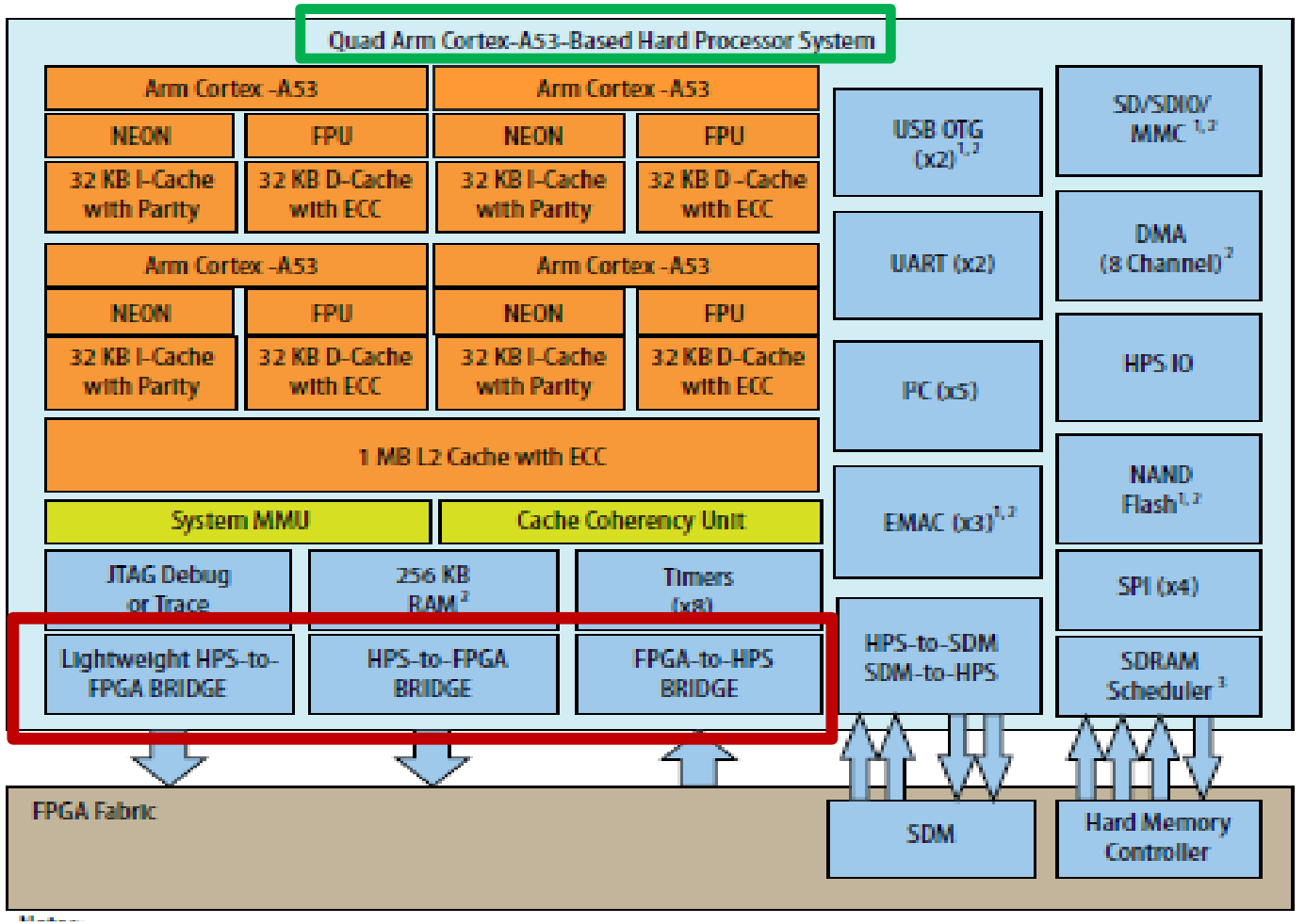
- These are FPGAs with inbuilt hard processor system and programmable logic connected to each other through various buses
- Thus, these are on-die integrated processor and programmable logic (*no more discrete systems*)
- Examples:
  - Xilinx Zynq-7000 series FPGAs
  - Intel Stratix 10 SoC FPGA, Arria 10 SoC FPGA

ZYNQ 7000 SOC OVERVIEW



INTEL STRATIX 10 SOC FPGA

- Quad Core Arm Cortex – A53
- HPS to FPGA and FPGA to HPS Bridges
- Source: Intel Stratix 10 Device Overview, 2020.09.28



ZYNQ-7000 FAMILY

Processing System (PS)

Programmable Logic (PL)

	Z-7010	Z-7015	Z-7020	Z-7030	Z-7045	Z-7100
Processor	Dual core ARM Cortex-A9 with NEON and FPU extensions					
Max. processor clock frequency	866MHz			1GHz		
Programmable Logic	Artix-7			Kintex-7		
No. of FlipFlops	35,200	96,400	106,400	157,200	437,200	554,800
No. of 6-input LUTs	17,600	46,200	53,200	78,600	218,600	277,400
No. of 36Kb Block RAMs	60	95	140	265	545	755
No. of DSP48 slices (18x25 bit)	80	160	220	400	900	2020
No. of SelectIO Input/Output Blocks[a]	HR: 100 HP: 0	HR: 150 HP: 0	HR: 200 HP: 0	HR: 100 HP: 150	HR: 212 HP: 150	HR: 250 HP: 150
No. of PCI Express Blocks	-	4	-	4	8	8
No. of serial transceivers	-	4	-	4	8 or 16[b]	16
Serial transceivers maximum rate	-	6.25Gbps	-	6.6Gbps / 12.5Gbps [c]	6.6Gbps / 12.5Gbps [b]	10.3Gbps

a. Depends on the package; maximum numbers are shown here. HR = High Range, HP = High Performance.
b. Depends on package chosen.
c. Depends on the speed grade of the device.

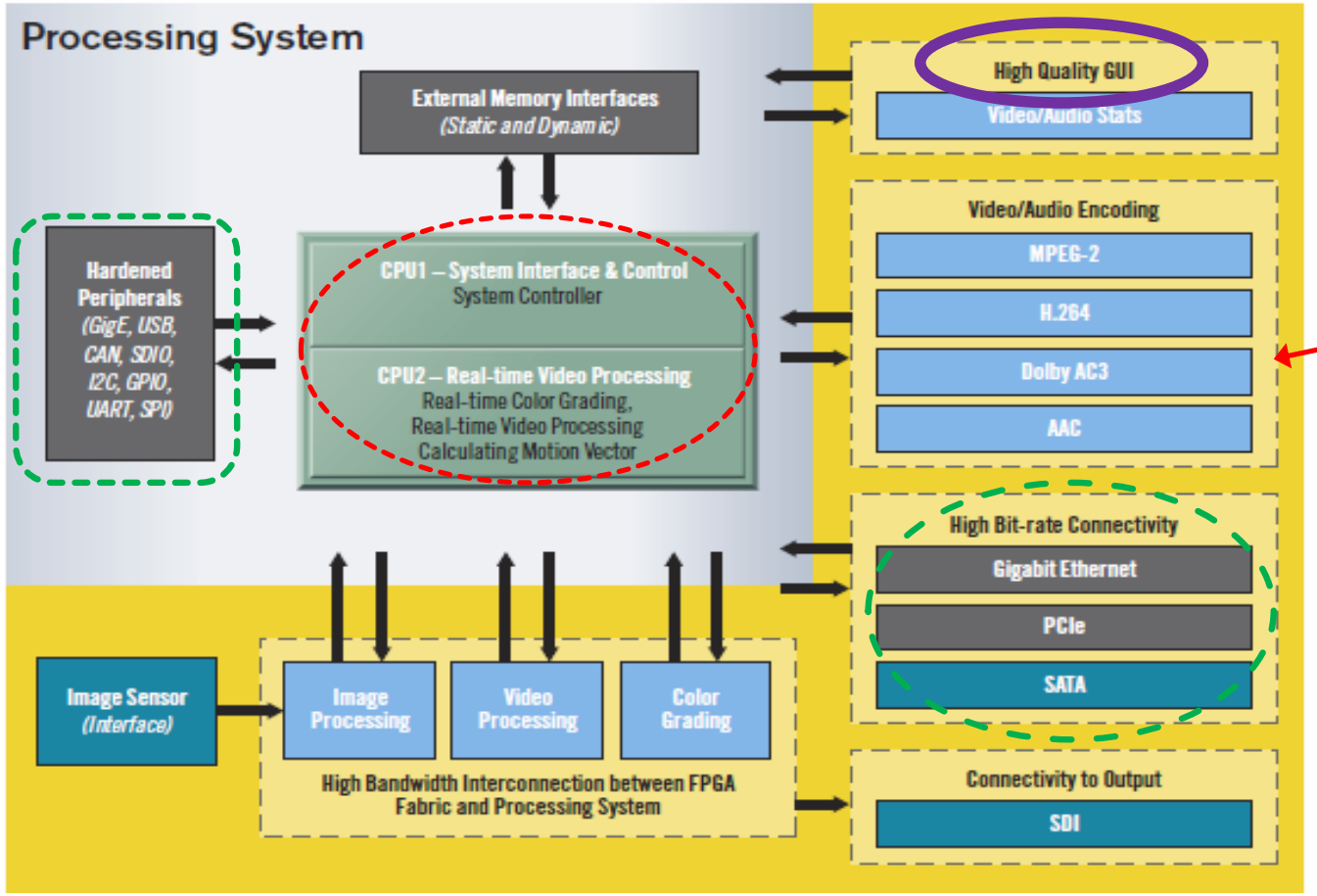
Note that the main difference is on the PL side

EXAMPLE MAPPING OF APPLICATIONS TO DEVICES

CLUSTER	MARKET	KEY APPLICATIONS	SAME PROCESSING SYSTEM Different Programmable Logic Densities ● Optimal for application			
			Z-7010	Z-7020	Z-7030	Z-7045
Intelligent Video	Auto	Driver Assistance, Driver Information, Infotainment	●	●		
	Consumer	Business-class Multi-function Printers	●	●	●	
	ISM	IP and Smart Cameras	●	●	●	
		Medical Diagnostics, Monitoring and Therapy	●	●		
		Medical Imaging	●		●	●
	Broadcast	Prosumer/Studio Cameras, Transcoders		●	●	●
	A&D	Video/Night Vision Equipment	●	●		
Comms	A&D	Milcomms, Cockpit & Instrumentation		●	●	●
	Wireless	LTE Radio, Basband, Enterprise Femto		●	●	●
	Wired	Routers, Switches, Multiplexers, Edge Cards			●	●
Control	ISM	Motor Control, Programmable Logic Controller (PLC)	●	●	●	
	A&D	Missiles, Smart Munitions			●	●
Bridging	Broadcast	Edge QAMs, Routers, Switchers, Encoders/Decoders			●	●
	ISM	Industrial Networking	●	●	●	

Source: Zynq Product Brief

BROADCAST CAMERA APPLICATION EXAMPLE



PL Side supports:

1) Connectivity
2) Encoder blocks
3) Image processing blocks

PS side supports:

1) Control
2) Interface with peripherals
3) Memory interface

Source: Zynq Product Brief

MAIN COMPONENTS OF A SOC FPGA (ZYNQ AS AN EXAMPLE)

COMPUTE	Programming System (PS) (ARM Cores)
	Programmable Logic (PL) (LUT, DSP, FF)
COMMUNICATE	UART, I2C, Select IO etc.
MEMORY	DDR 2/3/ Memory Controller
	On-chip Memory (OCM)
	On Chip SRAM
INTERCONNECT	Central Interconnect
	PS to PL Interconnect
	OCM Interconnect
CONTROL	PL Configuration Related Blocks

ZYNQ 7000 SOC OVERVIEW

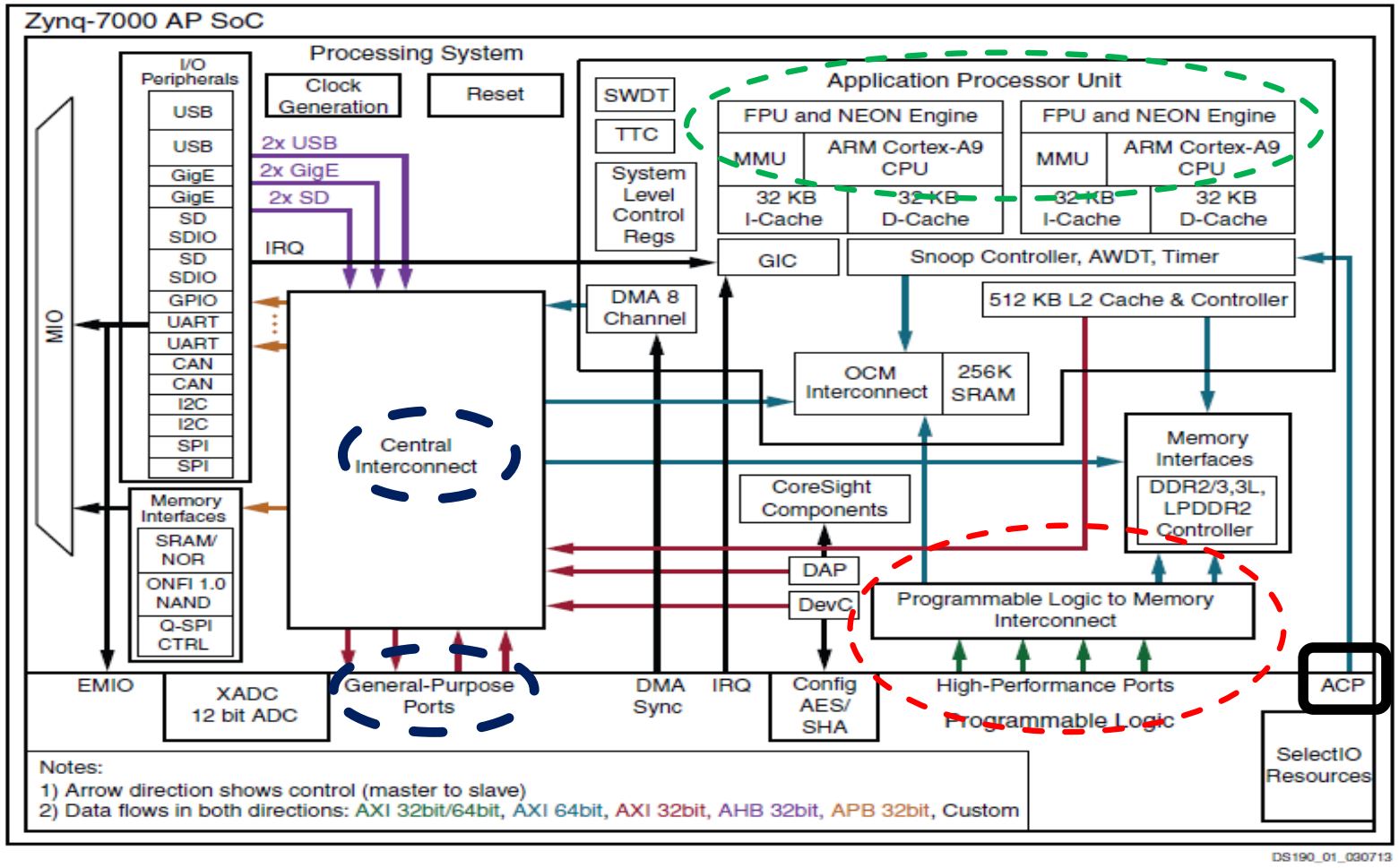
FPGA – CPU Communication:

General Purpose (GP) Ports

High Performance (HP) Ports

Accelerator Coherency Port (ACP)

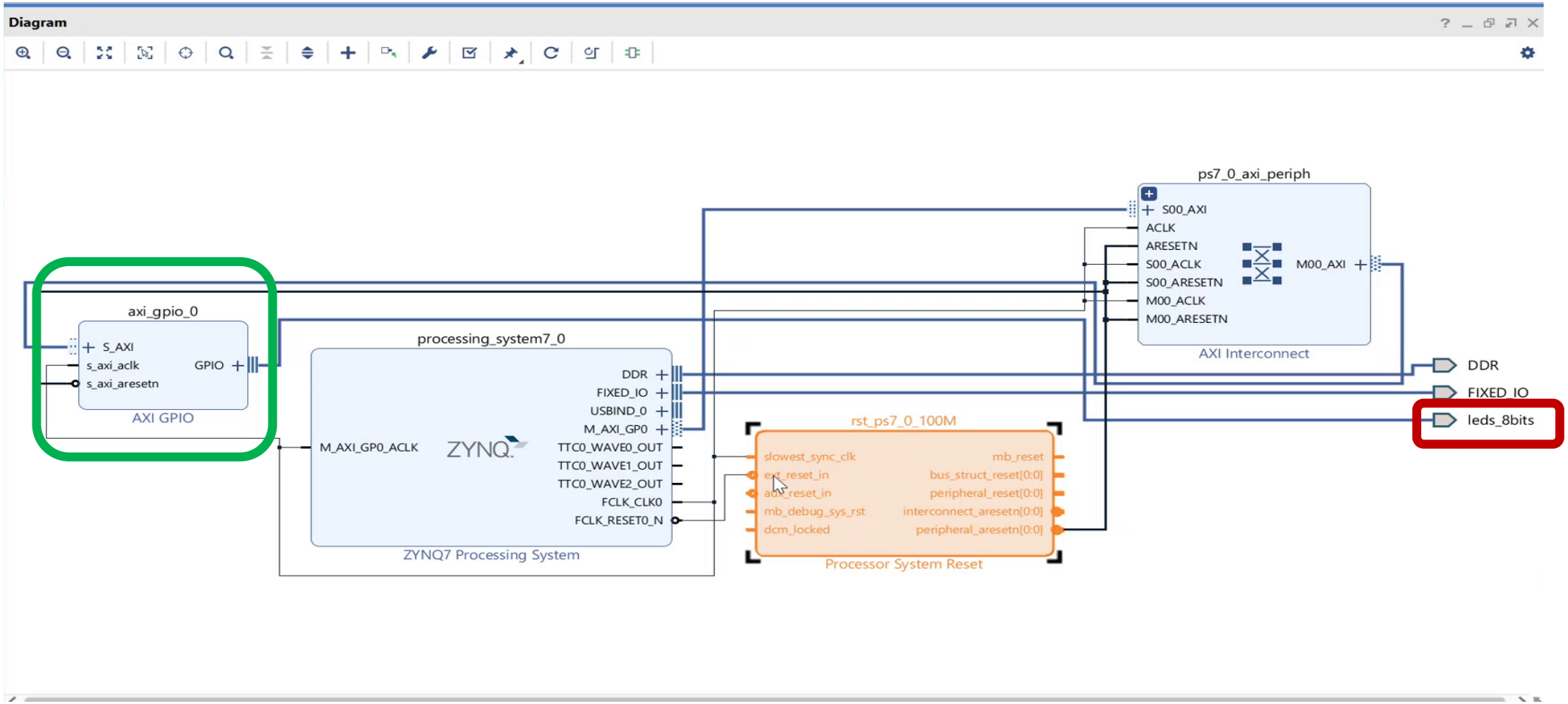
These are Arm eXtensible Interface (AXI) ports



HOW DO YOU BUILD A DESIGN ON SOC FPGA?

- Main Goal: Split your application between CPU and Programmable Logic (PL) side
- Typical Steps (for Xilinx SoC FPGA):
  - 1. Create a Block Diagram Design (BDD) in Vivado
    - Any user added IP block is treated as a module implemented on the PL side
  - 2. Synthesize and generate bit-stream in Vivado
  - 3. Export generated hardware to Vivado SDK
  - 4. Write the software to run on Arm processor
    - User added IP blocks are accessed through hardware device drivers (automatically generated by Vivado)

AN EXAMPLE BLOCK DIAGRAM



WHY THIS EMPHASIS ON HETEROGENEOUS COMPUTING?

- Use PEs for tasks that they are best suited for – that is the mantra
- For example:
  - Processors – good at control tasks, user interface, running an OS, providing multitasking etc.
  - FPGA – good at hardware acceleration of algorithms (because of high spatial parallelism), low power density (watt/operation)

WHAT TO PUT ON THE PL SIDE?

- Or, in other words, which function(s) to "Accelerate in Hardware"
- Hardware Acceleration refers to:
  - Accelerating the execution of a function/task/application by running it on custom hardware like FPGA compared to software execution (on a processor)
  - If software execution time is t_{sw} , hardware execution time is t_{hw} , we define a ratio $R_{acceleration}$ as:

$$R_{acceleration} = \frac{t_{sw}}{t_{hw}}$$

  - $R_{acceleration} > 1$ is the goal
  - In the design community, this is referred to as "Software Offload" and the unit processing this in hardware is called "Hardware Accelerator or Offload Engine"
  - You must remember that t_{hw} should be a combination of actual computation time in hardware and the time spent in reading data needed by the hardware accelerator and writing results

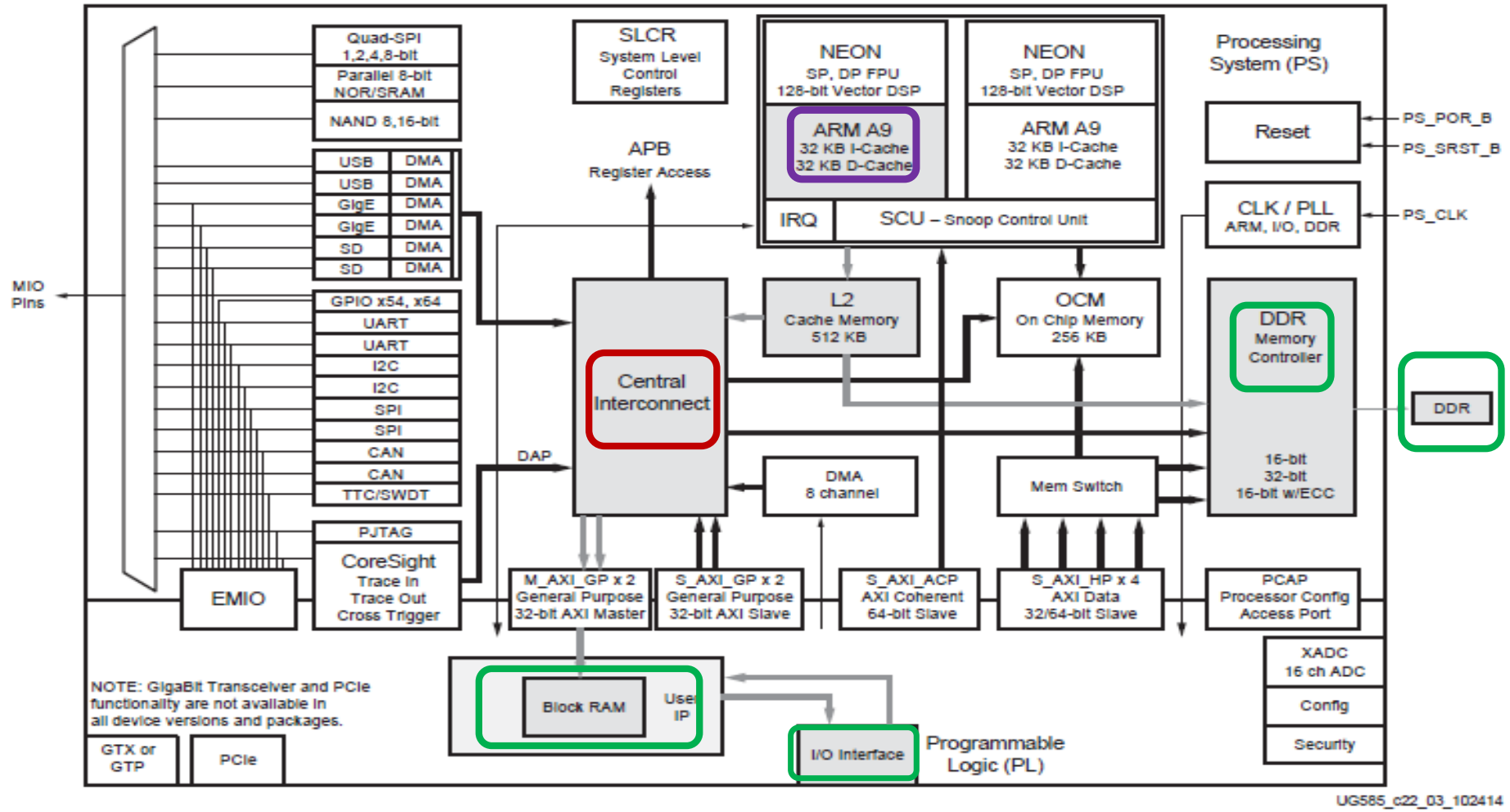
WHICH FUNCTION/TASK TO ACCELERATE?

- Identify "Hot Spots" in an application
  - Use software/code profiling
- Software/code Profiling
  - Basically tells you which functions, instructions etc. are consuming what percentage of the total execution time reported
- Dataflow Consideration
  - It is important to consider the dataflow to the accelerator.
  - In many cases, it is easier to design the hardware unit for the algorithm than to establish an efficient dataflow. The aim is to "keep the accelerator fed with data at all times or as quickly as possible"
  - In Zynq SoC, dataflow would mean data flowing between accelerator in PL and external DRAM or between accelerator in PL and the ARM processor or a combination of these.

WHICH FUNCTION/TASK TO ACCELERATE?

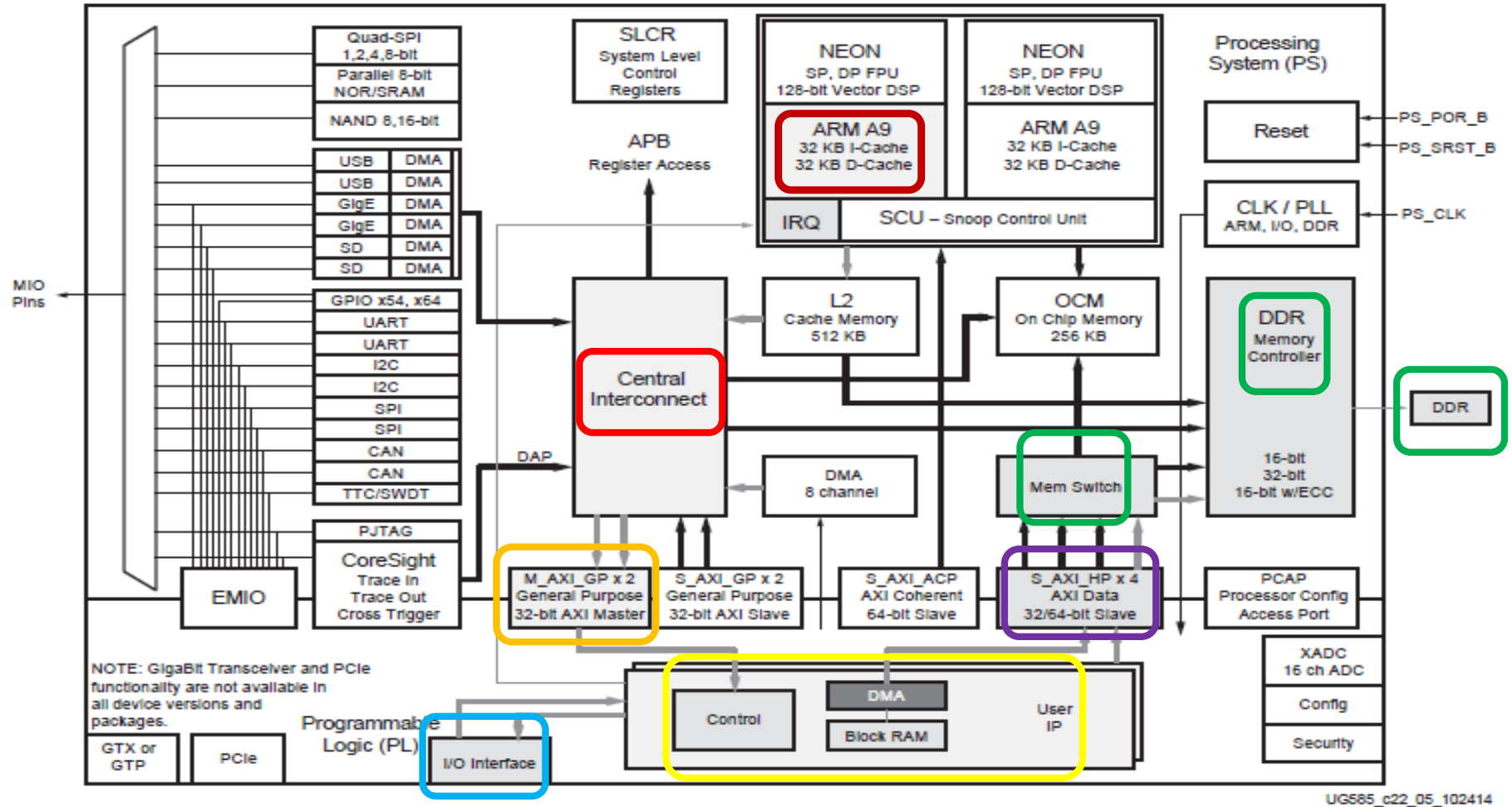
- Some general guidelines:
  - Lots of computation
  - Function/task can be parallelized (parallelizability)
  - Mostly data dependence among operations
  - Less control dependence among operations

EXAMPLE OF DATA MOVEMENT IN ZYNQ



Source: Zynq Technical Reference Manual; UG585

ANOTHER EXAMPLE OF DATA MOVEMENT IN ZYNQ



LET'S TAKE AN APPLICATION EXAMPLE FOR ZYNQ (MATRIX MULTIPLICATION)

- Consider the multiplication of two matrices **A** and **B** where both are 100x100 in size and each matrix element is a 16 bit integer
- Hence, each matrix has 10,000 of 16-bit elements = 1,60,000 bits = 20,000 bytes
- Let's say that a user needs to input all these matrix elements through a Graphical User Interface (GUI)
- Question: how will you implement this matrix multiplication on Zynq?

LET'S TAKE AN APPLICATION EXAMPLE FOR ZYNQ (MATRIX MULTIPLICATION)

- $\begin{bmatrix} a1 & b1 & c1 \\ d1 & e1 & f1 \\ g1 & h1 & i1 \end{bmatrix} \times \begin{bmatrix} a2 & b2 & c2 \\ d2 & e2 & f2 \\ g2 & h2 & i2 \end{bmatrix} = \begin{bmatrix} a3 & b3 & c3 \\ d3 & e3 & f3 \\ g3 & h3 & i3 \end{bmatrix}$
- Then, a3 = a1*a2 + b1*d2 + c1*g2

  b3 = a1*b2 + b1*e2 + c1*h2
- The calculation of a3 is independent of that of b3, except for (a1,b1,c1) being common to both

LET'S TAKE AN APPLICATION EXAMPLE FOR ZYNQ (MATRIX MULTIPLICATION)

- Subject to the resource constraints on the PL side, you can calculate as many product elements in parallel as possible.
- This means that respective multiplier and multiplicand elements needs to be made available to a **hardware block** on PL that implements:

$$a3 = a1*a2 + b1*d2 + c1*g2$$

- You can implement multiple such blocks in PL side to calculate multiple product elements in parallel

LET'S TAKE AN APPLICATION EXAMPLE FOR ZYNQ (MATRIX MULTIPLICATION)

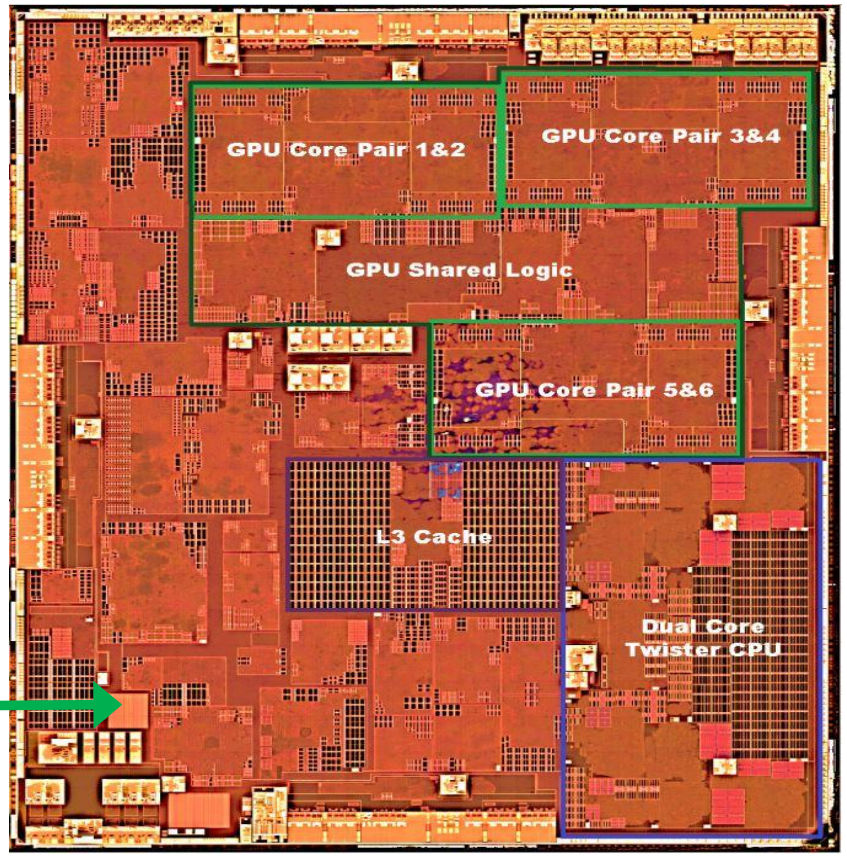
- It is much easier to take user inputs through the GUI when the GUI is running on the processor
- The processor can then store the matrix on the OCM or stream it to PL side or write to DDR (depending on storage required and dataflow model adopted)
- You will find that you will get faster results when you calculate multiple elements in parallel in PL side

CONCLUSION

- Heterogeneous architecture focuses on using PEs for tasks that they are good at
- Application development on such architectures may require knowledge of both hardware and software
- Can we make it easier to develop applications for such systems?
  - You may read Intel OneAPI specification

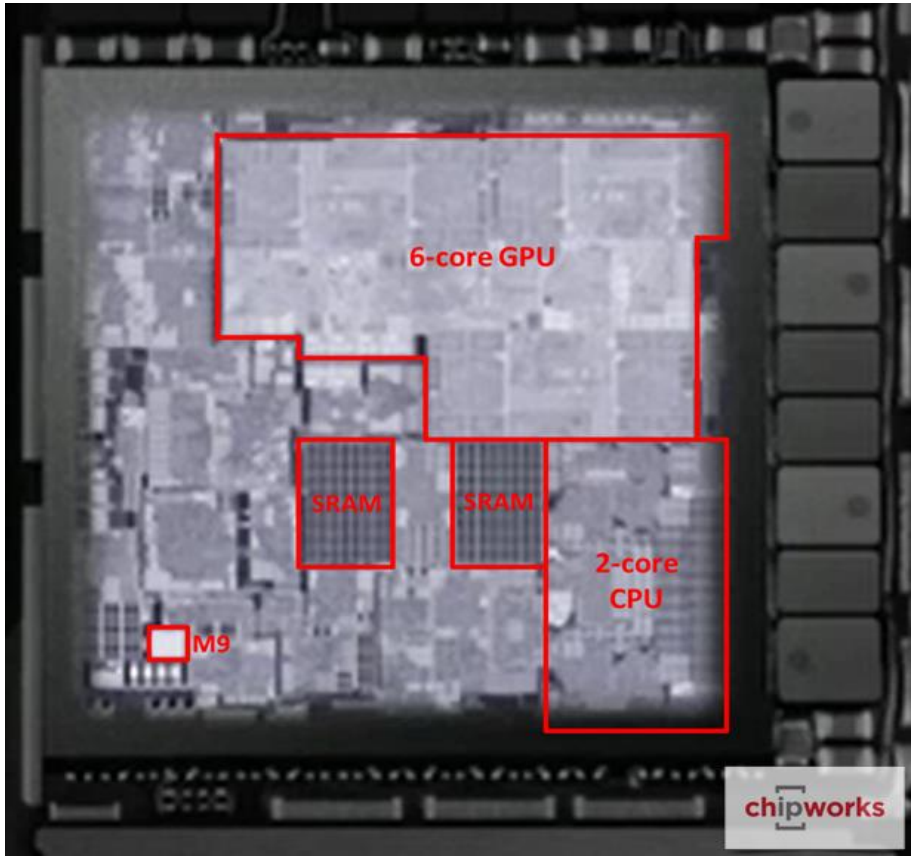
(https://spec.oneapi.com/versions/latest/oneAPI-spec.pdf)

AN INTERESTING IMAGE – HETEROGENEOUS SYSTEM



Motion
Co-processor M9

Picture courtesy: Anandtech



Initial Layout Analysis courtesy of Chipworks

FURTHER READING

- UG585 – Zynq Technical Reference Manual, Xilinx
- UG1165 – Zynq Embedded Design Tutorial, Xilinx
- UG904 – Vivado User Guide, Xilinx
- Advanced FPGA Design, Steve Kilts, Wiley
- FPGA Based Implementation of Signal Processing Systems, Roger Woods et al., Wiley
- Zynq Book: http://www.zynqbook.com/ (freely available)
- Heterogeneous Systems Architecture: https://www.youtube.com/watch?v=In8JpfaLvbM

Thank you

Q&A