GETTING THE MESSAGE ACROSS

"DDR communication through AXI-IPs"

Prakhar Sharma

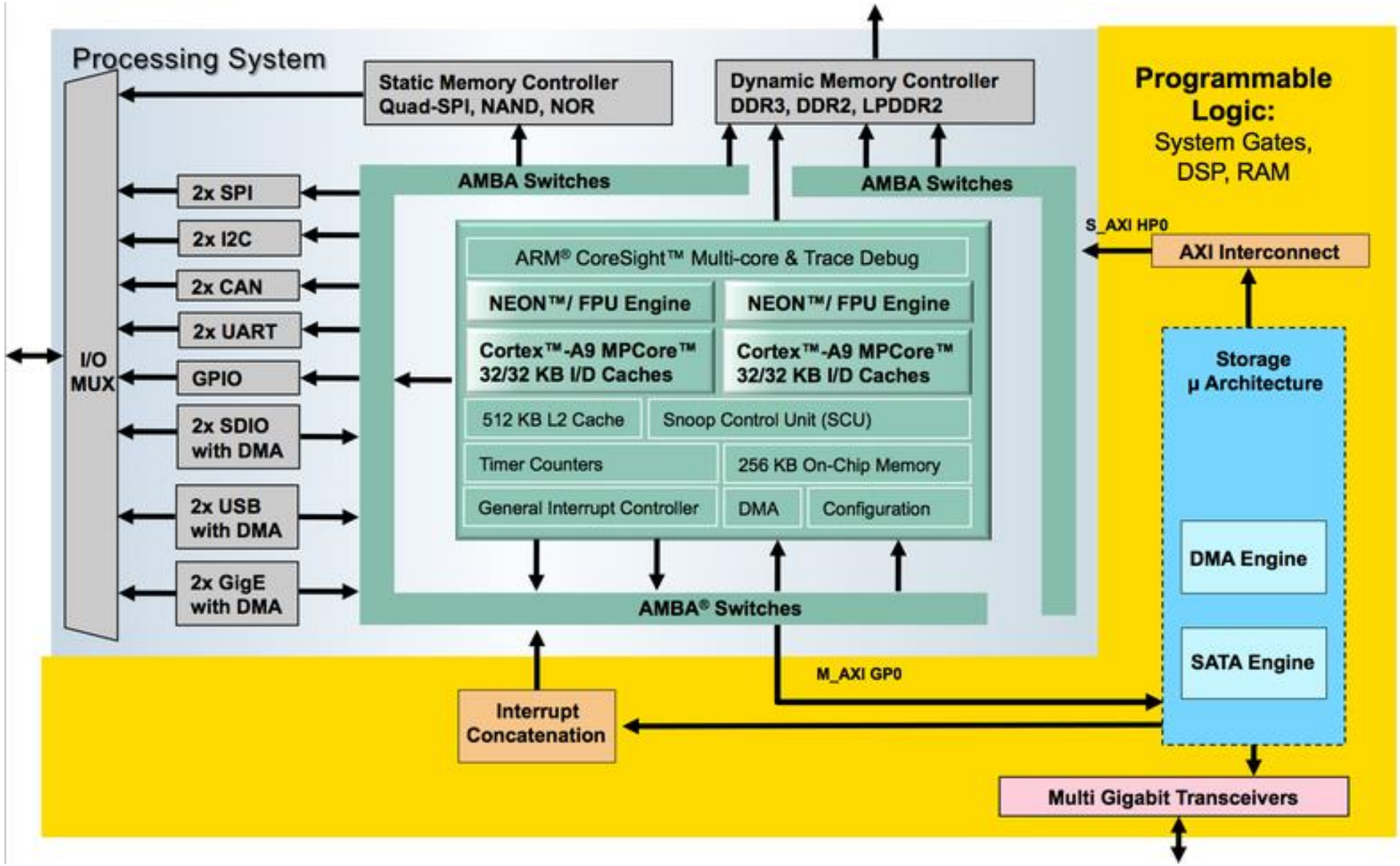
Green-IC group, NUS

http://sharmaprakhar.github.io/

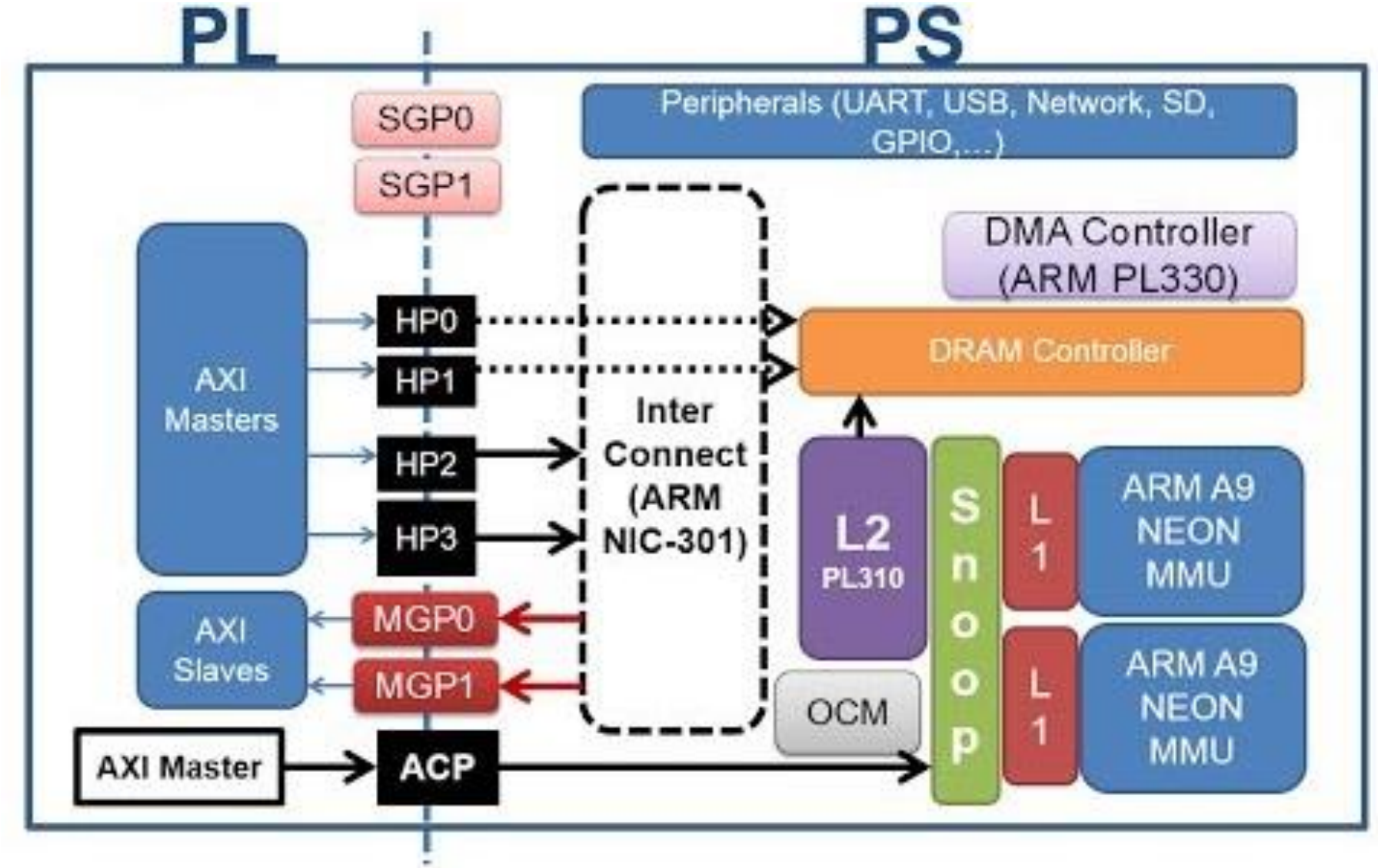
OVERVIEW

- Zynq-7000 Architecture (PS, PL, DDR and ports)
- Xilinx toolchain for embedded development (Vivado, SDK, XMD)
- What is DDR and what is all the fuss about?
- Talking to the DDR – the DMA approach
- Talking to the DDR – the AXI memory mapped approach
- Talking to the DDR – the MIG approach
- The hidden IP – AXI burst IPIF
- Language and customs of the AXI world (AXI burst perspective)
- Simulating your design
- Hardware simulation on board

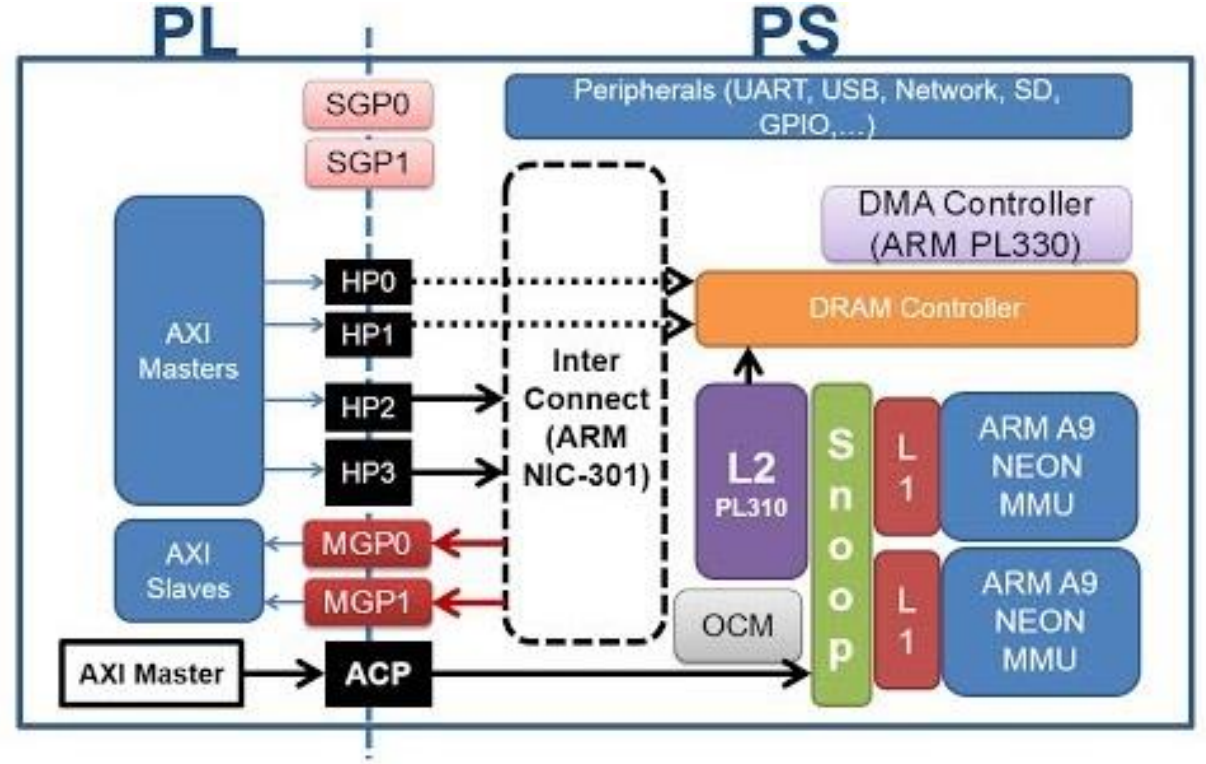
Zynq-7000 Architecture (PS, PL, DDR and ports)



Zynq-7000 Architecture (PS, PL, DDR and ports)



Xilinx toolchain for embedded development (Vivado, SDK, XMD)



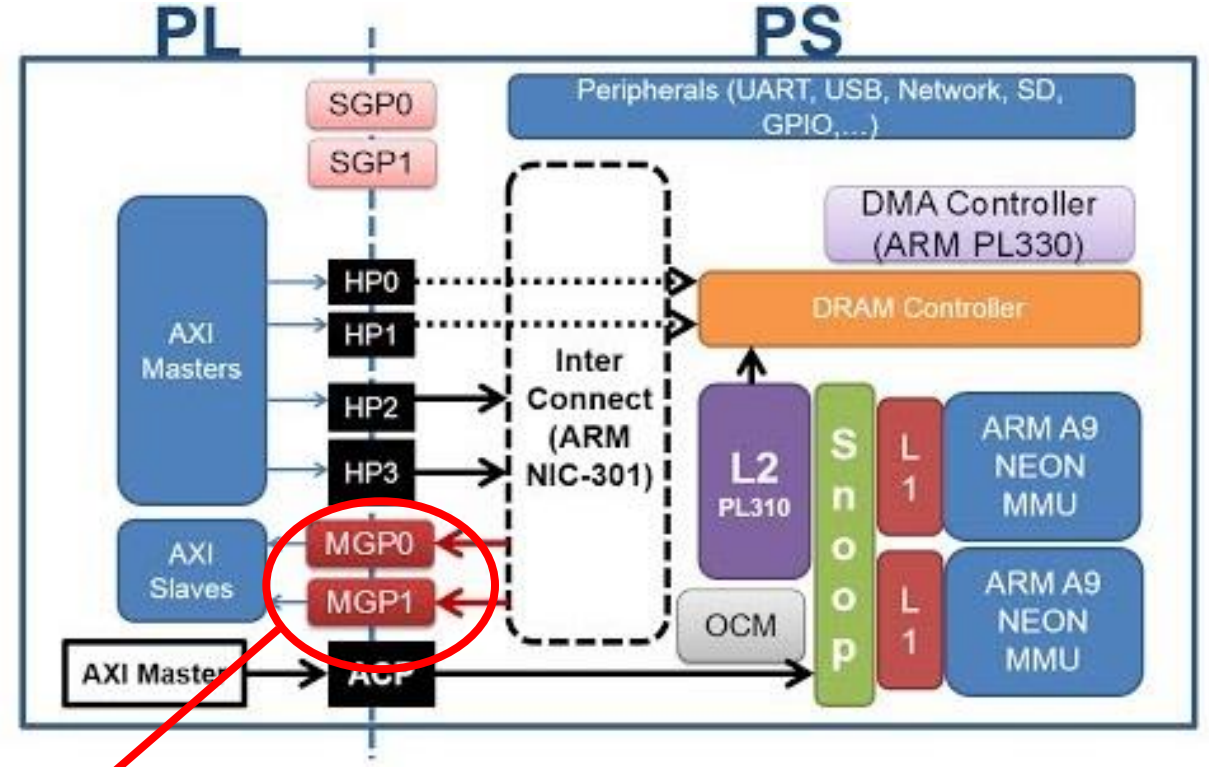
SDK, XMD

- Program ARM core
- Debug ARM core

Vivado, custom IP integrator, IP packager

Xilinx toolchain for embedded development (Vivado, SDK, XMD)

Vivado, custom IP integrator, IP packager



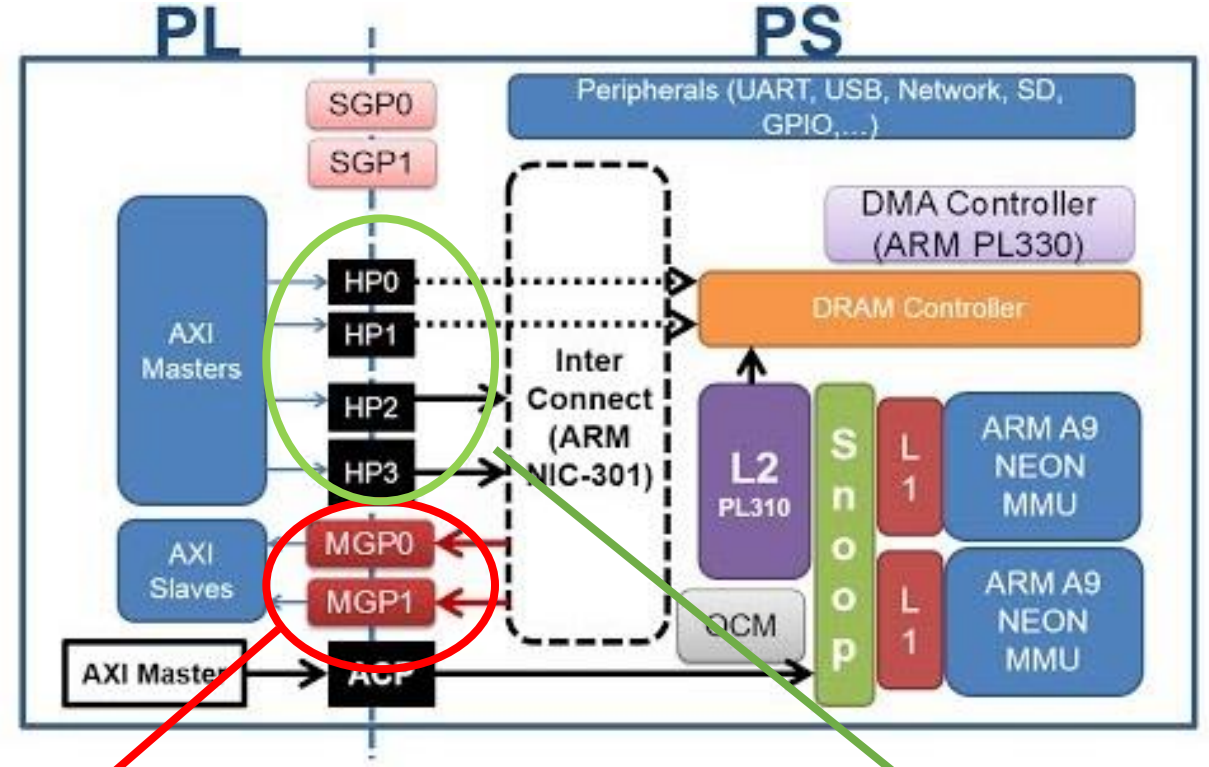
SDK, XMD

- Program ARM core
- Debug ARM core

- ARM core ports – master
- Used to Read/Write to the PL
- Data transfer happens through the SDK (C code)

Xilinx toolchain for embedded development (Vivado, SDK, XMD)

Vivado, custom IP integrator, IP packager



SDK, XMD

- Program ARM core
- Debug ARM core

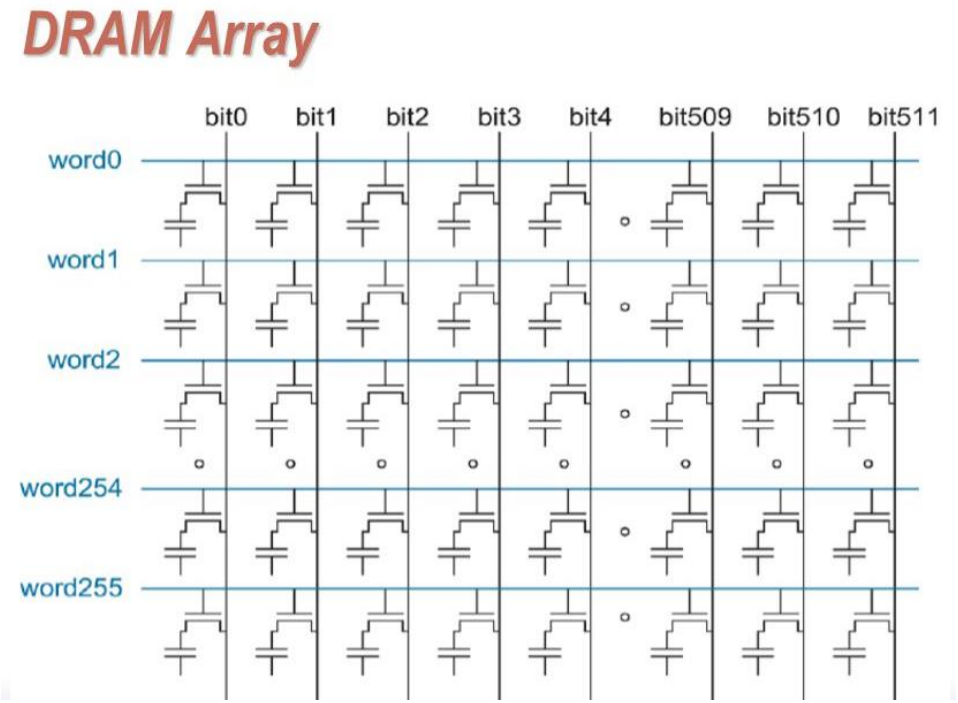
- ARM core ports – master
- Used to Read/Write to the PL
- Data transfer happens through the SDK (C code)

- High performance ports on the PL-PS boundary
- Used for data acquisition from the DDR (Read/Write)

What is DDR

- Basically a DRAM array
- Volatile storage – so data is lost when not powered on
- Data width and data address – two most important parameters
- All other parameters are a property of the data communication protocol
- In the case of Zynq 7000 – AXI4 is the protocol
- Read and Write have different protocol handshaking signals
- Every horizontal location is a physical address in hexadecimal representation e.g. 0x0400_0000
- Transfer data on both rising and falling clock edges
- Higher transfer rates as opposed to SDRAMs
- Tighter clocking constraints (more in an

Advanced memory technology class)



TALKING TO THE DDR – Central DMA

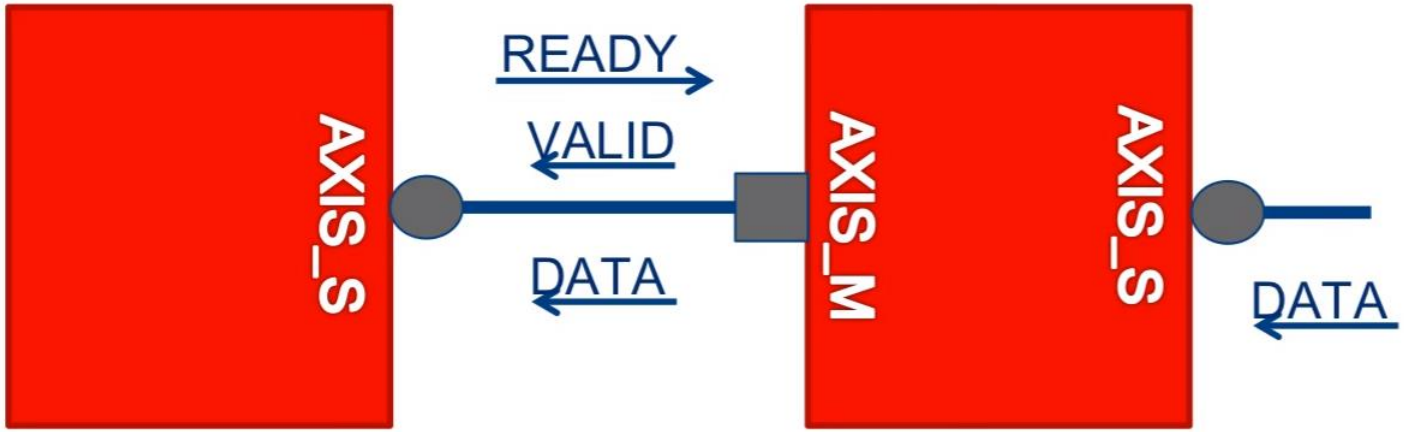
- DMA is programmed through SDK in C or C++
- Standard functions available in the xaxidma_hw.h header file available from Xilinx
- Refer to AXI DMA documentation chapter “Designing with the core” for all the details needed to design with the core. Look for heading “programming sequence”
- Simple DMA mode – Scatter gather mode
- DMA is programmed by writing values to physical registers

- Especially useful to write the AXI Stream data to the DDR – S2MM and MM2S

- Low control over timing and read/write sequence
- Hardware acceleration affected when the processor is engaged for every transfer

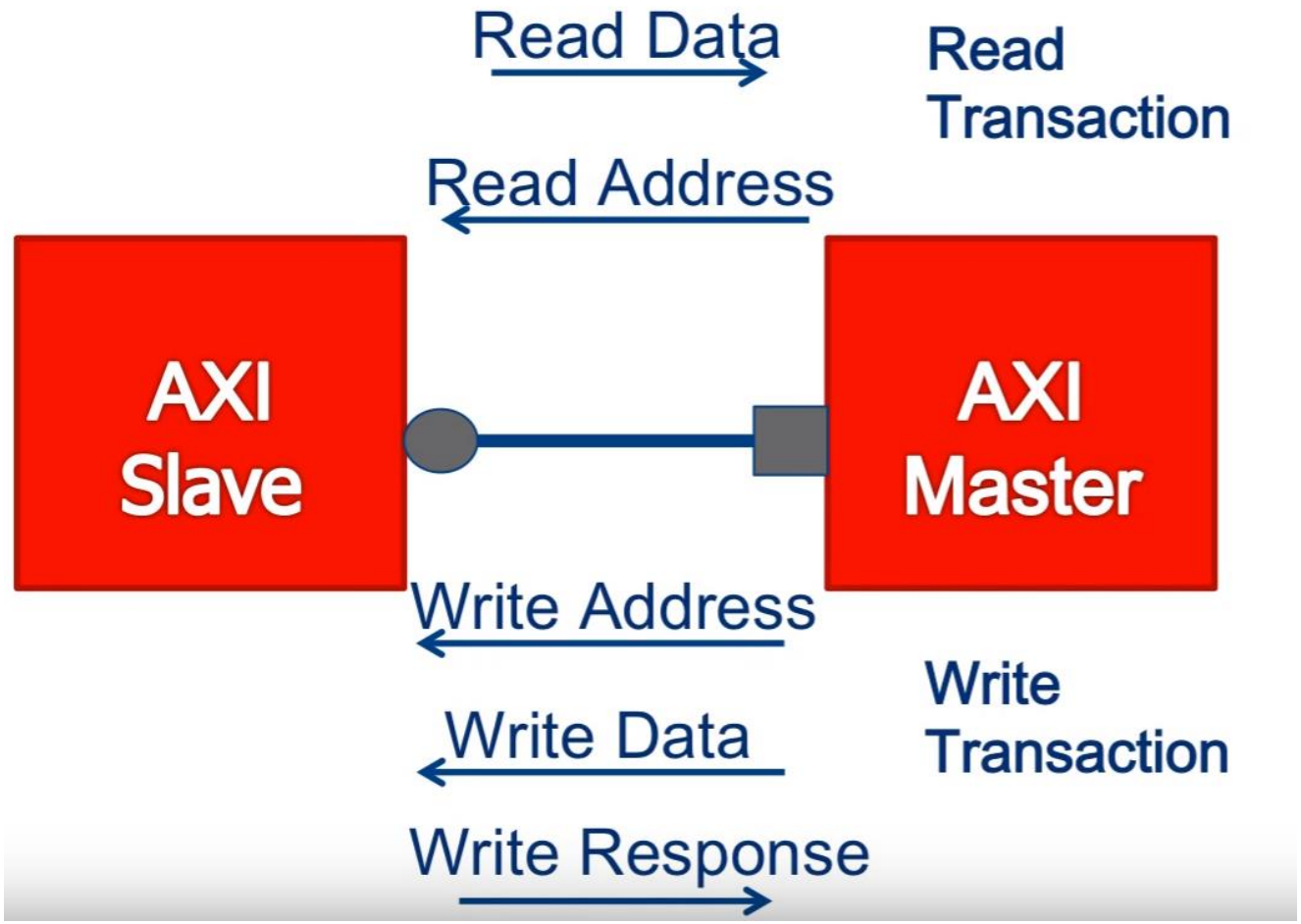
- What if we could offload the processing task of reading and writing to physical addresses to the co-processor??

TALKING TO THE DDR – FMM/MML/STREAM



AXI stream interface
(No addresses)
Especially good for image processing, DSP

TALKING TO THE DDR – FMM/MML



Lite does not support burst transfers but low complexity
Full MM has a complex setup of signals – tough to write from scratch
Refer to the Xilinx documentation for MM full/lite for the complete list of signals

TALKING TO THE DDR – MIG

- Memory Interface Generator directly lets the PL talk to the DDR through a design realized on the PL
- Does not need the HP ports
- Possible only on the ZC-706 board as zedboard does not have a PL side DDR memory bank
- IP customizable in the IP catalog in vivado
- Beyond our scope (read I have little idea about this IP)

If you figure MIG out please let me know!!

TALKING TO THE DDR – AXI burst IPIF

- Supports data burst (duh!)
- Smaller set of well defined signals (but still pretty complicated)
- Has no gui – hidden IP provided with the Xilinx IP catalog
- Implemented as raw VHDL files (Bad news Verilog designers)
- Needs a stable reset signal from the ARM processor to initialize
- User IP has to be “interwoven” with the VHDL files – needs considerable RTL coding experience
- A lot of signals to debug on the ILA core (no JTAG debugging now, we are talking hardware – need actual hardware probes e.g. chipscope)

- Lightning fast – hardware is always faster on logic than software (sort of the whole point of making co-processors)
- Tractable set of signals
- Easy to understand
- Has extensive documentation ☺ (always capitalize on resources)

TALKING TO THE DDR – AXI burst IPIF

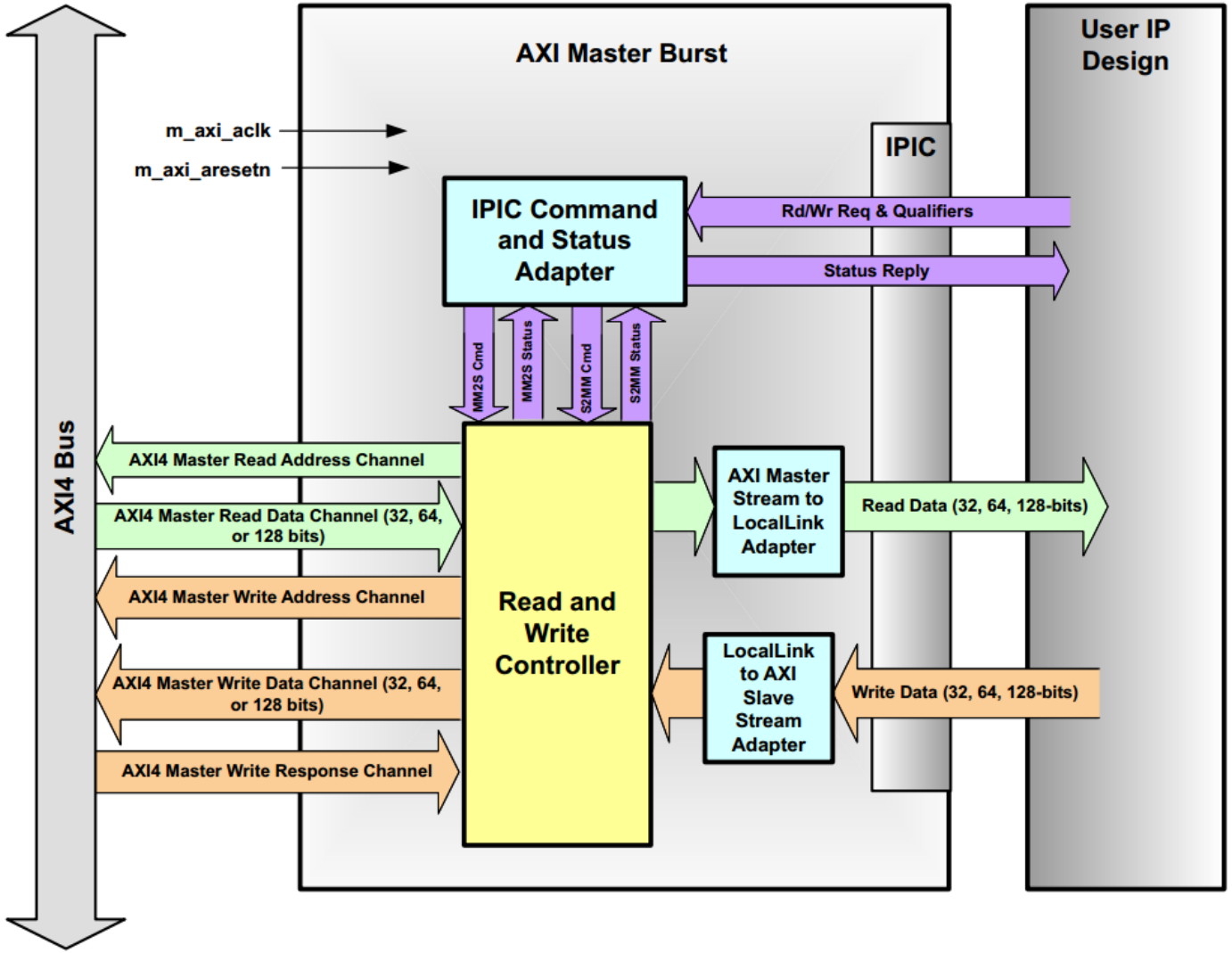
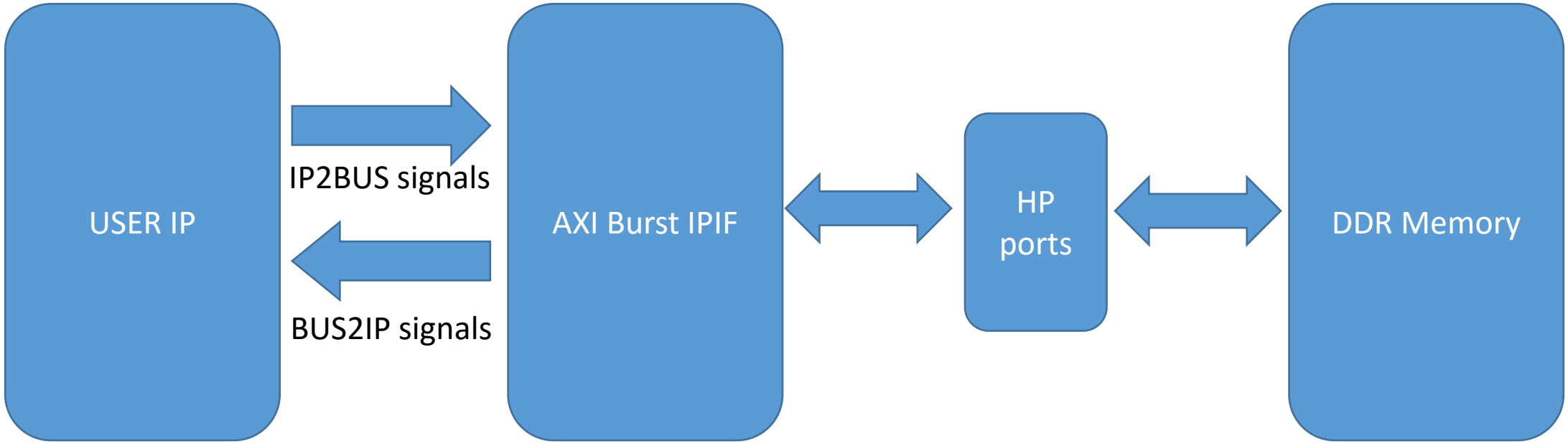


Figure 1: **AXI Master Burst Block Diagram**

http://japan.xilinx.com/support/documentation/ip_documentation/axi_master_burst/v2_0/pg162-axi-master-burst.pdf

TALKING TO THE DDR – AXI burst IPIF

Control IP2BUS signals here



BUS2IP signals dictate how to control IP2BUS signals

IPIF picks data from the memory and gets data to the memory

http://japan.xilinx.com/support/documentation/ip_documentation/axi_master_burst/v2_0/pg162-axi-master-burst.pdf

SIMULATING YOUR DESIGN

Requirement

- Debugging on the hardware should be to fine tune the design
- Pre bitstream design simulation is necessary for complex Ips
- ILA cores have limited memory to store and display probe data and waveforms

Problems

- AXI world BFM not available for cheap
- Zynq BFM simulation models available from Xilinx and Cadence – not cheap
- Extensive simulation setup required for user IP simulation with AXI world and Zynq
- Generally an RTL design team needed to carry out a full fledged emulation task for complex Ips

Need for simple simulation models that could behave like how IPIF behaves and replicates the signal sequence
Could verify simple IPs for glaring faults
Assume that Zynq and AXI4 world IPs are verified and tested by Xilinx

Note: clock cycle latency might be different than what you actually see on the board

https://github.com/sharmaprakhar/zynq_ddr_design_ex