Welcome to the Webinar Offered by:

National Microelectronics Security Training (MEST) Center

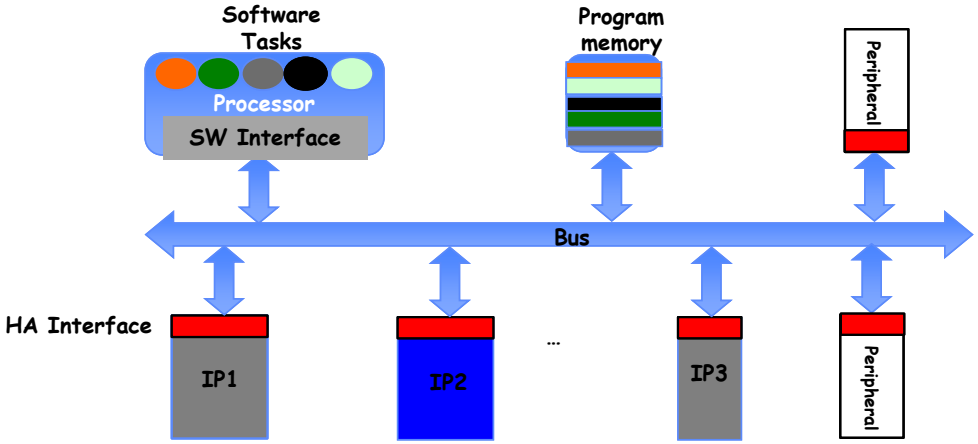
Sponsors:

Introduction to System on Chip

System Integration

Abigail Butka, Kawser Ahmed Muhammed, Christophe Bobda

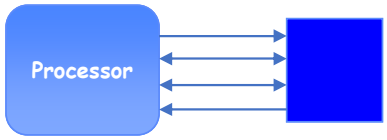
SoC- Architecture



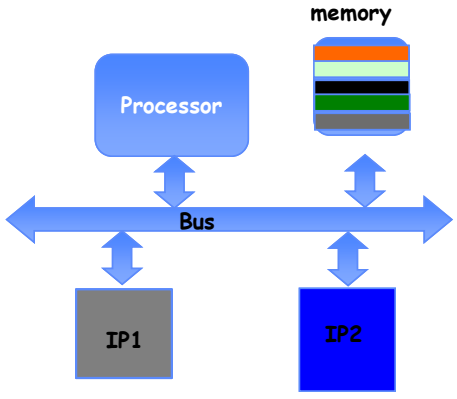
System Integration

Interfacing: I/O addressing

- On a PCB a microprocessor communicates with other IP using some of its pins.
- On SoCs, ports are used for communication.
- Two interfacing approaches
  - Port-based I/O (parallel I/O)
    - Processor has one or more N-bit ports
    - Processor's software reads and writes a port just like a register
    - E.g., P0 = 0xFF; v = P1.2; -- P0 and P1 are 8-bit ports

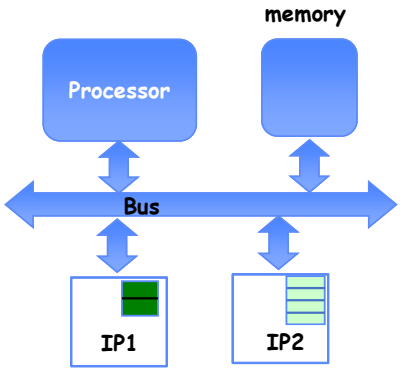


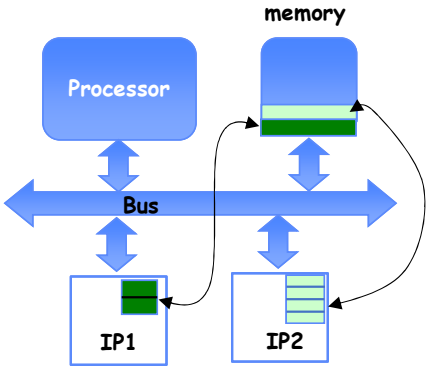
  - Bus-based I/O
    - Processor has address, data and control ports that form a single bus
    - Communication protocol is built into the processor
    - A single instruction carries out the read or write protocol on the bus



Bus-based I/O: Memory-mapped and standard I/O

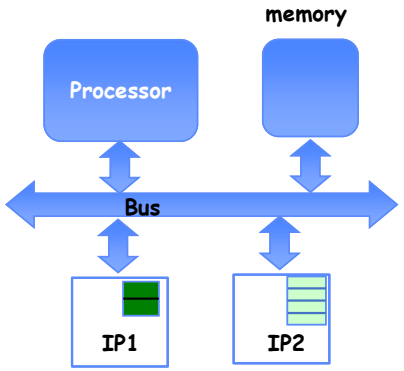
- Processor talks to both memory and peripherals using same bus – two ways to talk to peripherals
  - Standard I/O (I/O-mapped I/O)
    - Additional pin (M/IO) on bus indicates whether a memory or peripheral access
    - e.g., Bus has 16-bit address
    - All 64K addresses correspond to memory when M/IO set to 0
    - All 64K addresses correspond to peripherals when M/IO set to 1
  - Memory-mapped I/O
    - Peripheral registers occupy addresses in same address space as memory
    - e.g., Bus has 16-bit address
    - Lower 32K addresses may correspond to memory
    - Upper 32k addresses may correspond to peripherals



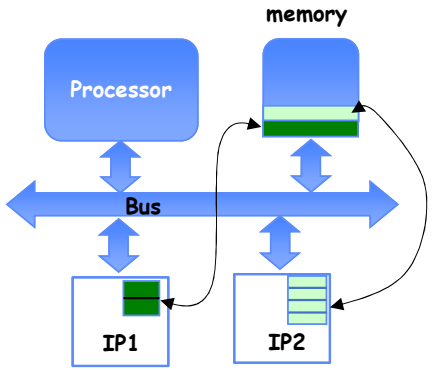


Bus-based I/O: Memory-mapped and standard I/O

- Standard I/O
  - No loss of memory addresses to peripherals
  - Simpler address decoding logic in peripherals possible
  - When number of peripherals much smaller than address space then high-order address bits can be ignored
  - Smaller and/or faster comparators



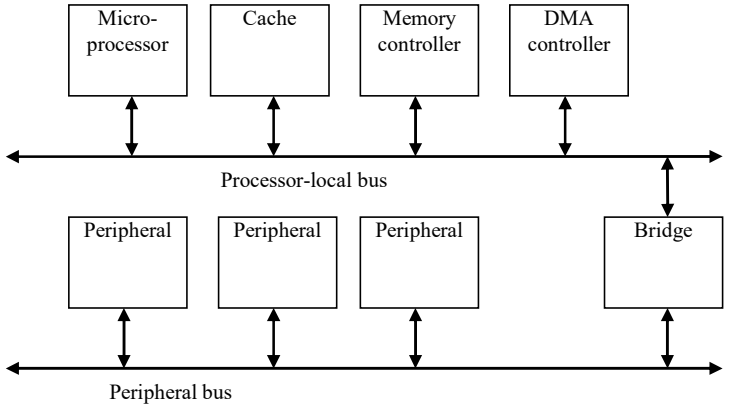
- Memory-mapped I/O
  - Requires no special instructions
  - Assembly instructions involving memory like MOV and ADD work with peripherals as well
  - Standard I/O requires special instructions (e.g., IN, OUT) to move data between peripheral registers and memory



Interconnect

SoC Interconnect: Multilevel bus architectures

- Processor-local bus
  - High speed, wide, most frequent communication
  - Connects microprocessor, cache, memory controllers, etc.
- Peripheral bus
  - Lower speed, narrower, less frequent communication
  - Typically, industry standard bus (ISA, PCI) for portability
- Bridge
  - Single-purpose processor converts communication between busses



SoC Interconnect: Multilevel bus architectures - ARM's AMBA

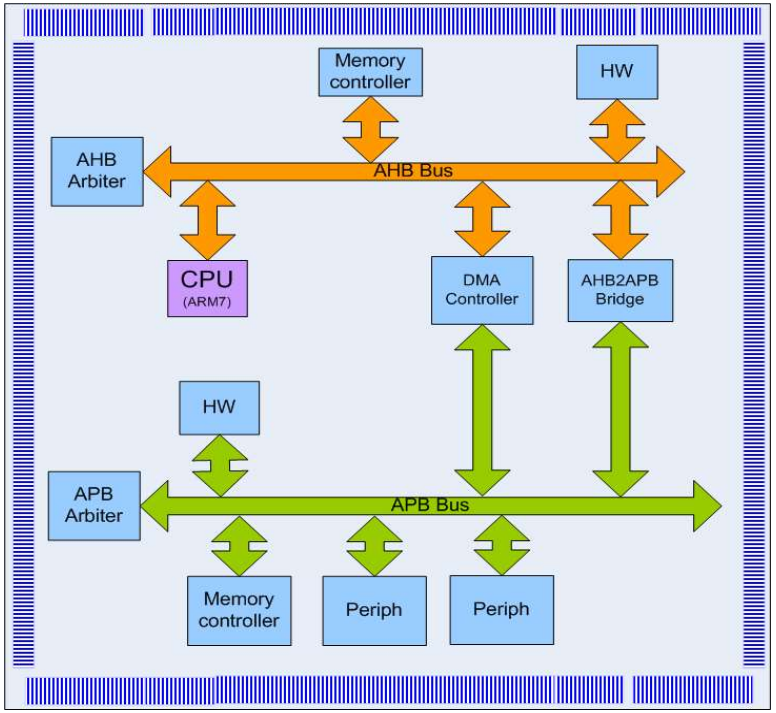
- Advanced Microcontroller Bus Architecture (Introduced by ARM in 96)
- Open standard, on-chip interconnect specification for the connection and management of functional blocks in a System-on-Chip (SoC)
- 32-bit addressing

- Early SoC Architectures
  - high-performance system interconnect
    - Advanced System Bus (ASB): Version 1
    - Advance High-Speed Bus (AHB): Version 2
  - Low-speed peripheral bus:
    - Advance Peripheral Bus (APB): Version 1 & 2
  - Cross Communication via a bridge

- 2003: 3rd generation, including AXI for connection of memory mapped components, with Advanced Trace Bus (ATB)

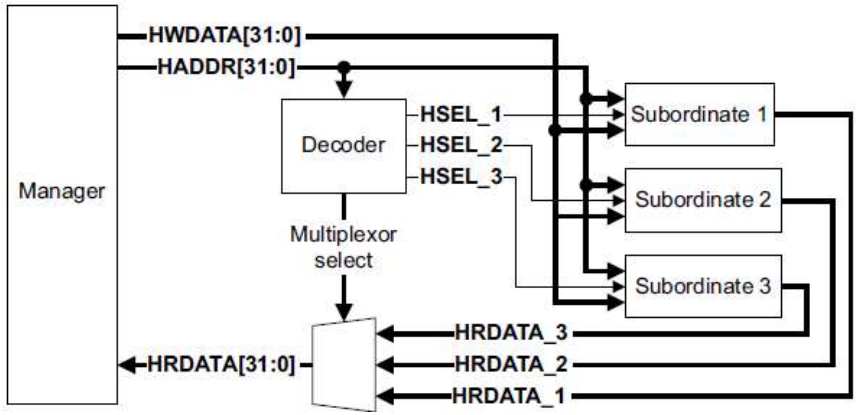
- 2010: 4th generation, AMBA4, incl AXI4 2011: 5th generation, AMBA5

- 2013: 5th generation AMBA5 with CHI (Coherent Hub Interface)



SoC Interconnect: Multilevel bus architectures - ARM's AMBA

- Defines the interface between
  - Managers (Master)
  - interconnects (Medium, Protocol)
  - Subordinates (Slave)
    - APB integration through a bridge

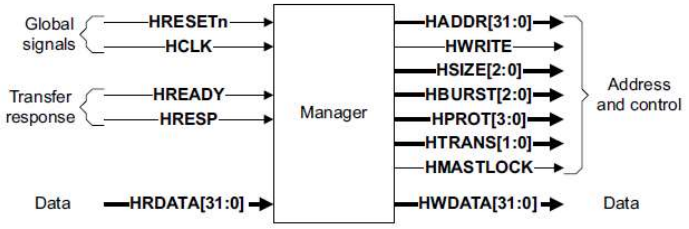


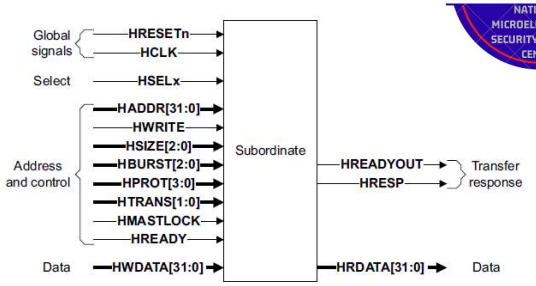
- Operation
  - Managers place read and write request and subordinates respond
  - Transfers can be single or burst
  - The write data bus moves data from the Manager to a Subordinate, and the read data bus moves data from a Subordinate to the Manager

- Every transfer consists of Address phase and Data phase
  - Strobe protocol using 4 clock cycle
  - Wait cycle can be introduced by slave
  - Subordinate HRESP to indicate the success or failure of a transfer

SoC Interconnect: Multilevel bus architectures - ARM's AMBA - AHB

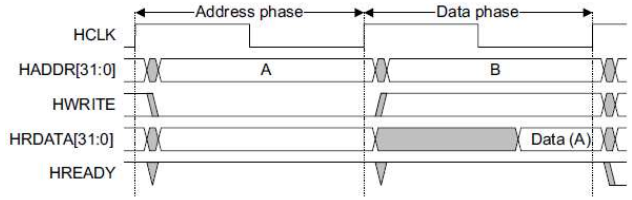
- Manager
  - Initiate and control read and write operations
- Subordinate
  - A Subordinate responds to transfers initiated by Managers
  - HSELx selects subordinate
  - Subordinate signals back to the Manager:
  - The completion or extension of the bus transfer
  - The success or failure of the bus transfer.



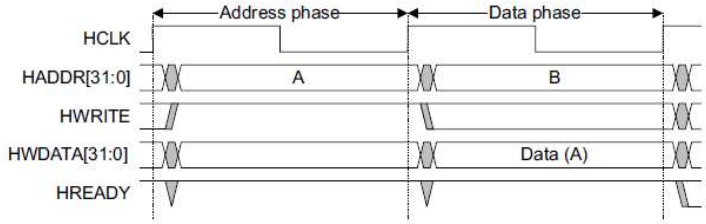


- Basic transfers
  - HWRITE = 1 for write transfer: Manager Place data on the bus
  - HWRITE = 1 = 0 for read transfers:
  - Selected subordinate places data on the bus
  - The simplest transfer is one with no wait states
  - one address cycle and one data cycle

Read Transfer

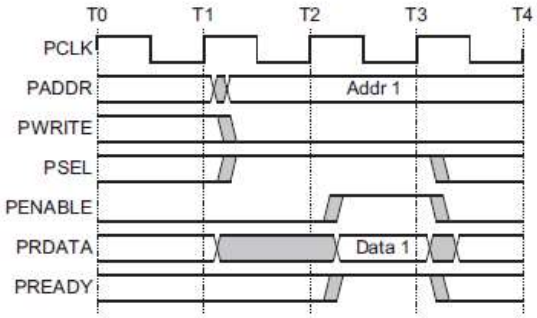


Write Transfer

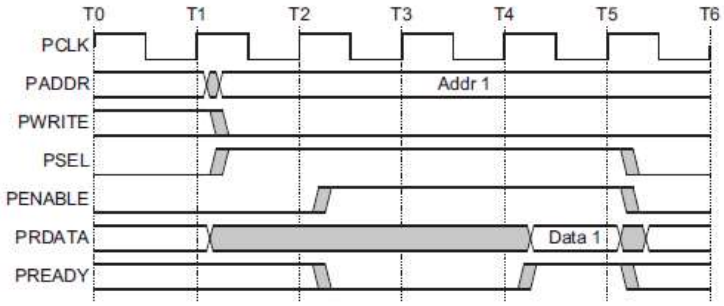


SoC Interconnect: Multilevel bus architectures - ARM's AMBA - APB

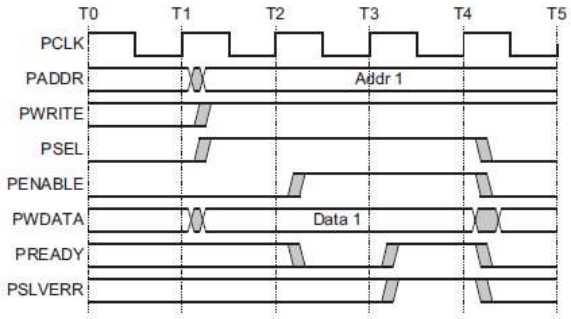
- Transfer controlled by a master (bridge) and a slave
- Structure like AHB
- Strobe transfer on 4 cycles



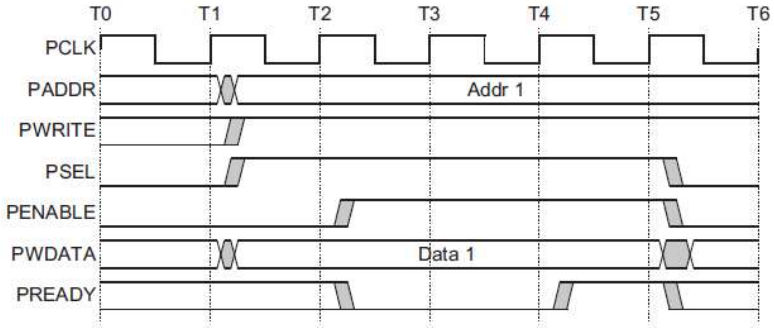
Read transfer, no wait Cycles



Read transfer with 1 wait Cycles



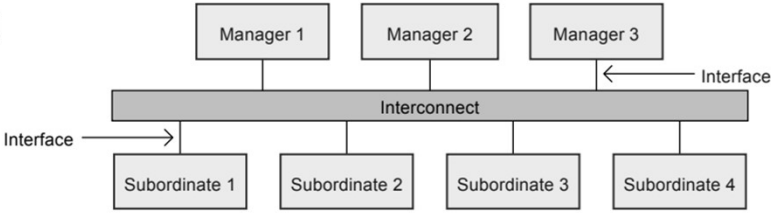
Write transfer, no wait Cycles

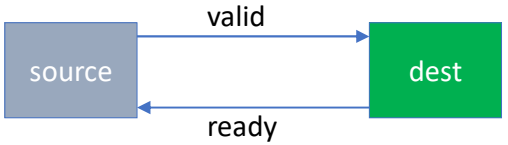


write transfer with 1 wait Cycles

SoC Interconnect: Multilevel bus architectures - ARM's AMBA - AXI

- Burst-based protocol with five independent transactio
  - All channels share the same handshaking mechanism
    - Read address, which has signal names beginning with AR.
    - Read data, which has signal names beginning with R.
    - Write address, which has signal names beginning with AW.
    - Write data, which has signal names beginning with W.
    - Write response, which has signal names beginning with B.
- Address channel carries control information that describes the nature of the data to be transferred
- Data transferred between Manager and Subordinate using either:
  - A write data channel to transfer data from the Manager to the Subordinate. Subordinate responds on uses the write response
  - A read data channel to transfer data from the Subordinate to the Manager



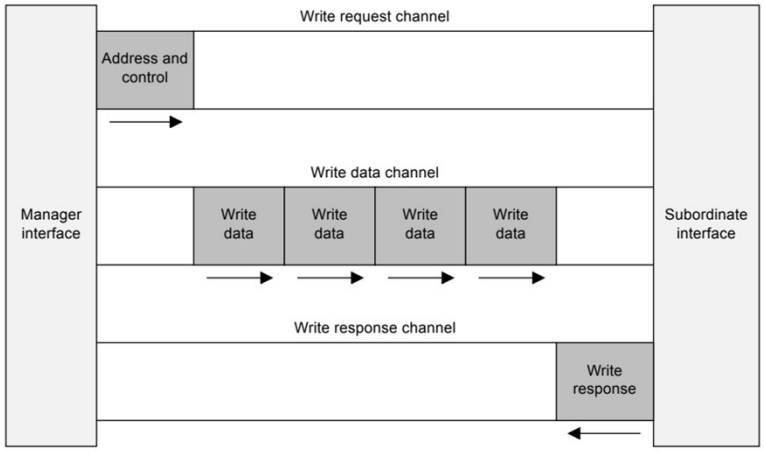


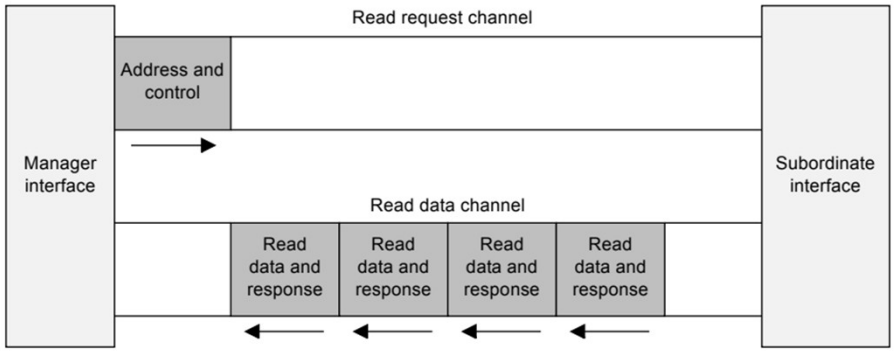
AXI Channel

SoC Interconnect: Multilevel bus architectures - ARM's AMBA - AXI

- Address information can be issued ahead of the actual data transfer
- Supports multiple outstanding transactions
- Supports out-of-order completion of transactions

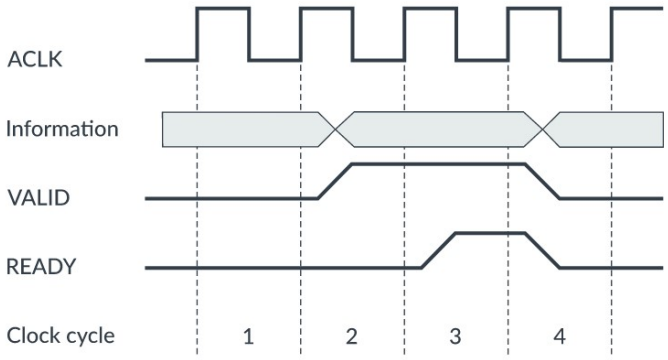
- Transfers and transactions
  - A transfer is a single exchange of information, with one VALID and READY handshake
  - A transaction is an entire burst of transfers, containing an address transfer, one or more data transfers, and, for write sequences, a response transfer



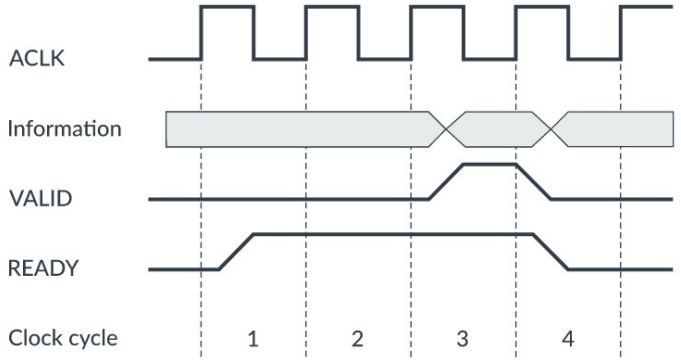


SoC Interconnect: Multilevel bus architectures - ARM's AMBA - AXI

Transfer Examples



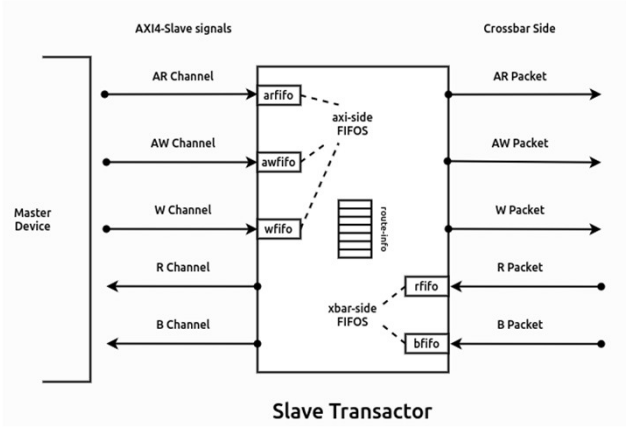
1. In clock cycle 2, the VALID signal is asserted, indicating that the data on the information channel is valid.
2. In clock cycle 3, the following clock cycle, the READY signal is asserted.
3. The handshake completes on the rising edge of clock cycle 4, because both READY and VALID signals are asserted.



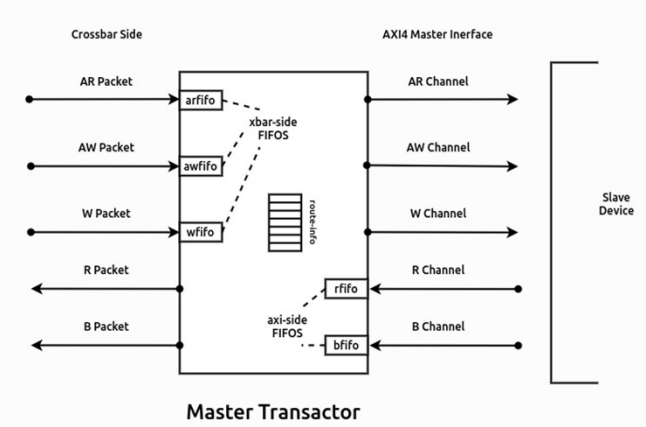
1. In clock cycle 1, the READY signal is asserted.
2. The VALID signal is not asserted until clock cycle 3.
3. The handshake completes on the rising edge of clock cycle 4, when both VALID and READY are asserted.

SoC Interconnect: Multilevel bus architectures - ARM's AMBA - AXI

- Crossbar switch: internally instantiates
  - M slave transactors and S master transactors
- A transactors provide
  - an AXI4 interface on one side (driving the external signals of the cross-bar) and a FIFO like interface on the other side.
  - The slave transactors provide an AXI4 slave interface externally to connect master devices (DMA masters, cache masters, etc.)
  - The master transactors provide an AXI4 master interface externally, to connected slave devices (UART, SPI, Memory controllers, etc.)

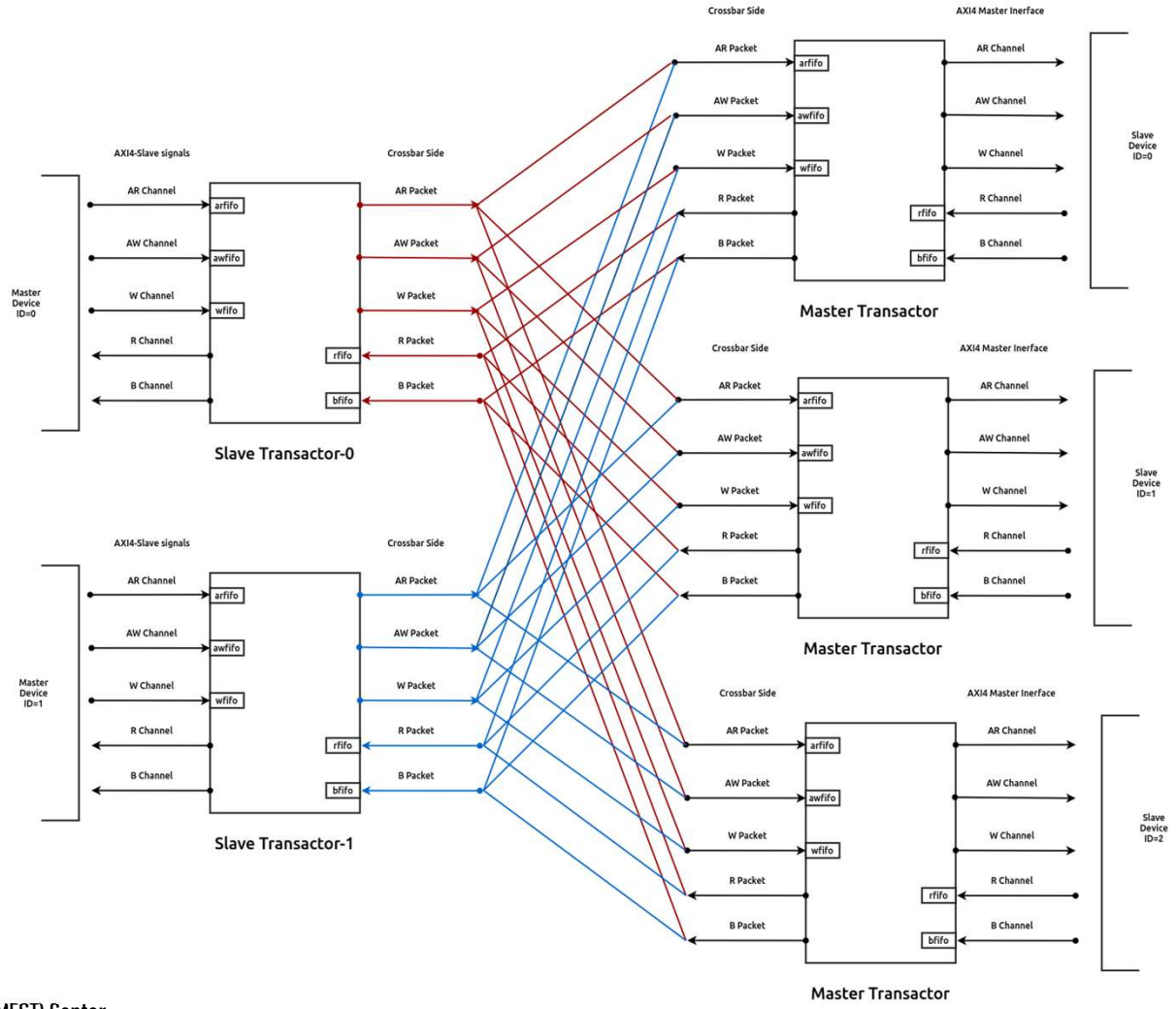


Slave Transactor



Master Transactor

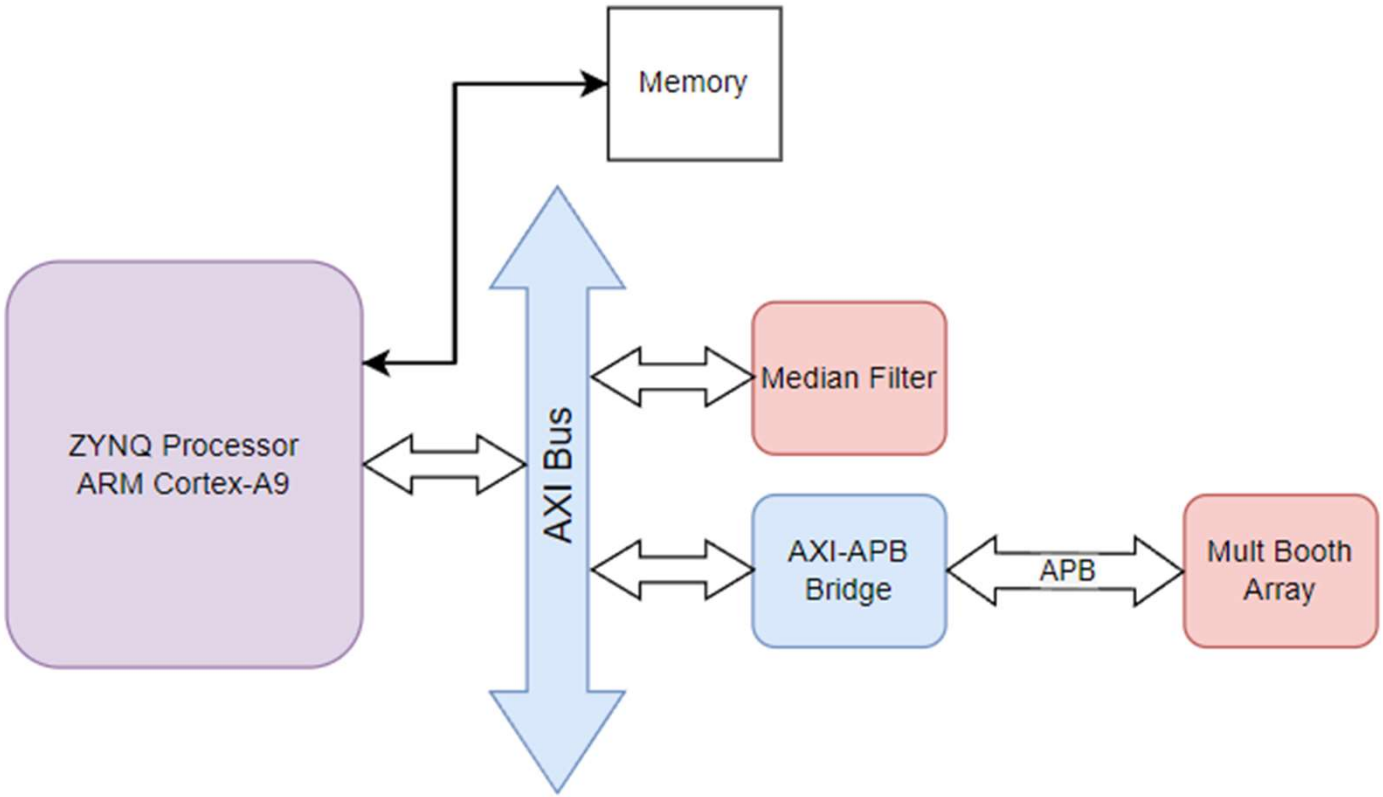
SoC Interconnect: Multilevel bus architectures - ARM's AMBA - AXI



Description of Module Tasks

Abigail Butka, Kawser Ahmed Muhammed, Christophe Bobda

Module Architecture



Module Tasks

- Task 1: Learn how create a custom AXI4 peripheral.
  - How to generate and customize an AXI4 peripheral.
  - Implementation and usage of Memory-Mapped I/O.
  - Write data to slave registers in peripheral, then read the results.

- Task 2: Learn how to create a custom APB peripheral.
  - How to generate and customize an APB peripheral.
  - Learn the differences between APB and AXI.
  - Write data to slave registers in peripheral, then read the results.

- Task 3: Use the custom peripherals created in Task 1 and Task 2 in the same design.
  - Learn how to set the addresses of Memory-Mapped I/O peripherals.
  - Learn how to refer to multiple peripherals in Vitis.