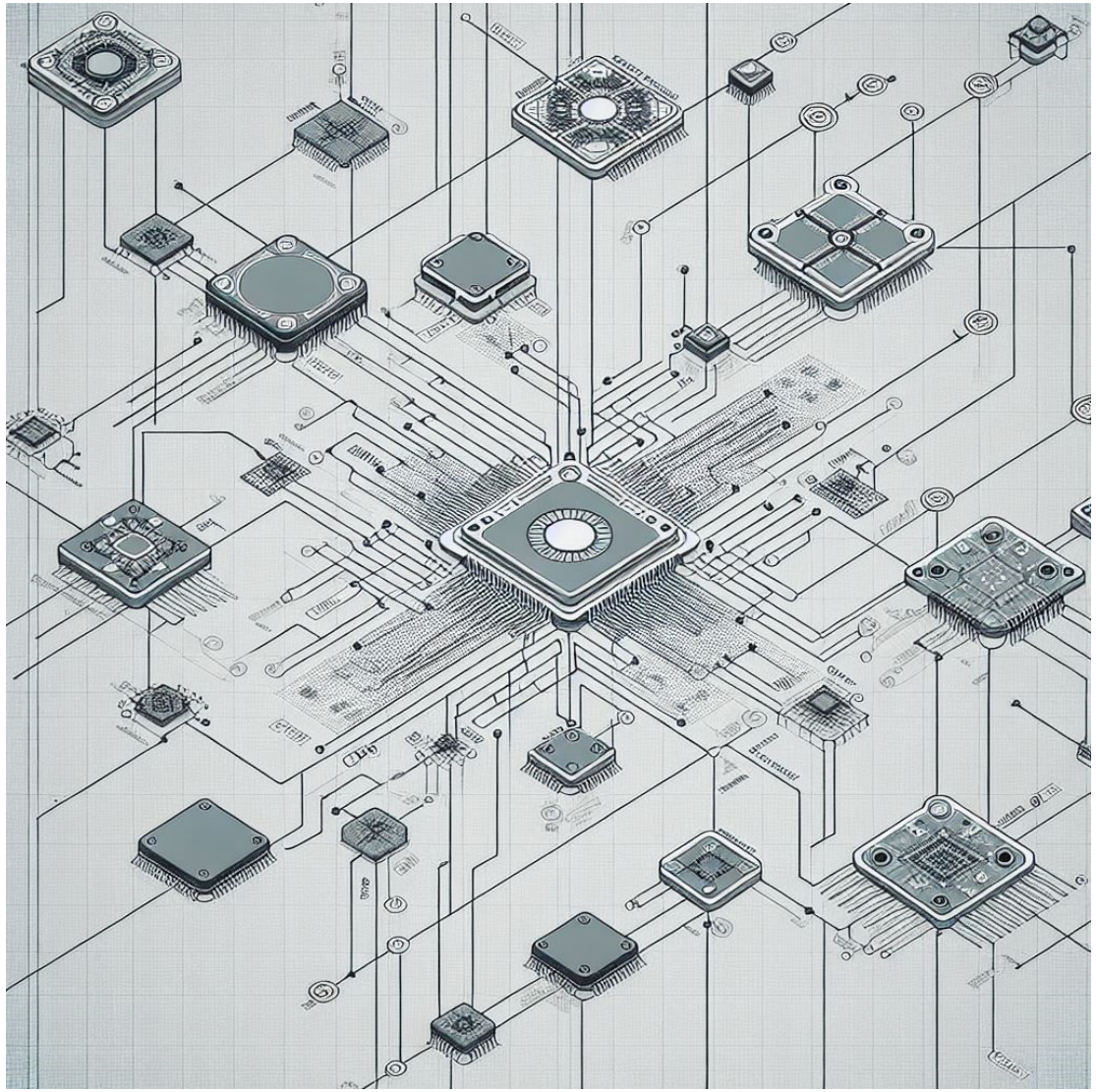
Interconnect Design

SoC Design – Comp.CE-250

Antti Rautakoura

Network-on-Chip topology block diagram drawn by CoPilot



Agenda

- Motivation
- Interconnect basics
- Network-on-Chips
- Interconnect design
- Interconnect verification

Focus of the lecture

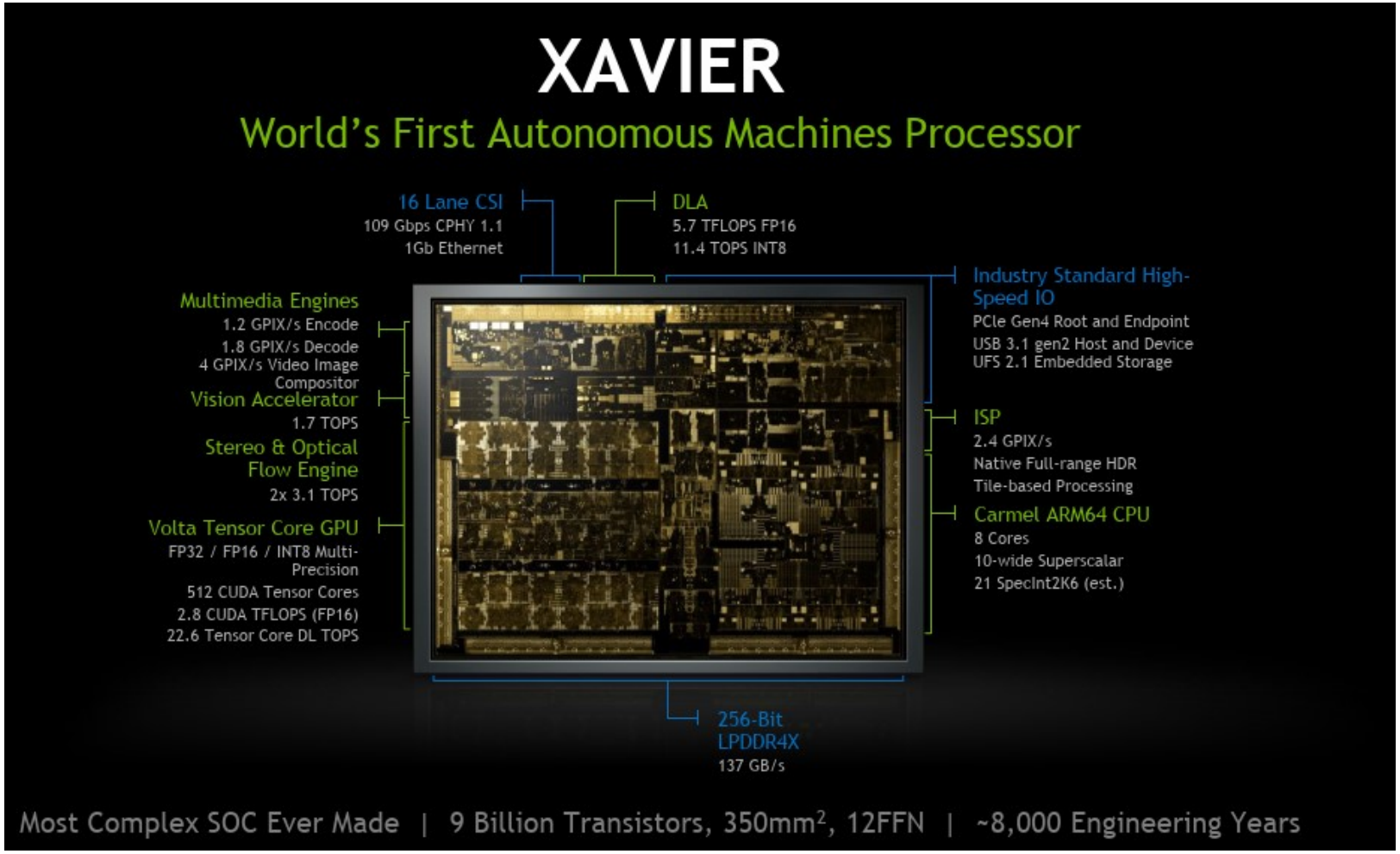
- Interconnects on chips. Just couple of slides about chiplets
- Practical approach
- During the lecture we build basic understanding so that you could participate to interconnect design and verification
- I have practical experience on designing bus and cross-bar based interconnect, but I do not have experience about network-on-chips

Short terminology

- **Interconnect**: The communication infrastructure within an SoC that connects various components, allowing them to communicate and share data.
- **Network-on-Chip (NoC)**: A scalable interconnect architecture that uses a network of routers and links to connect components. Typically employs multi-layer protocol stack bit similar (but simpler) to TCP/IP.
- **Interface**: Component ports related to interconnection
- **Topology**: The arrangement of interconnect in a network. Common topologies include shared bus, cross-bar mesh, and ring.
- **Bus/interconnect protocol**: A set of rules and standards that define how data is transmitted and received over the interconnect. E.g. ARM AHB (Advanced High-performance Bus)
- **Bus**: group of signals/wires e.g. SRAM CS, A, D, WE, OE. Can be used interchangeable with *Interface*
- **Bus**: Interconnect topology. Shared access medium

CoPilot was used as an aid in formulating the terminology

Motivation



https://old.hotchips.org/hc30/1conf/1.12_Nvidia_XavierHotchips2018Final_814.pdf

History: Early days

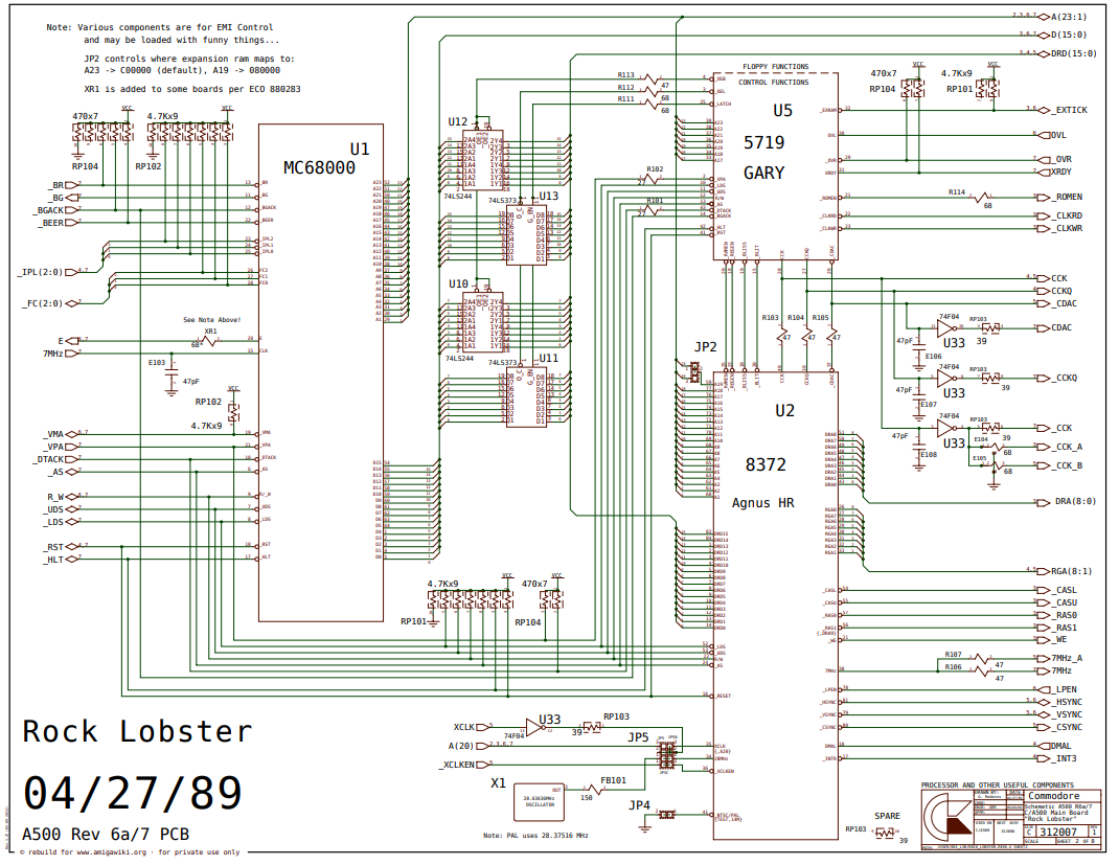
- Interconnect: Wires between separate sub- components on different PCBs

Some of the 50 miles of wiring

https://en.wikipedia.org/wiki/Cray-1

History: Circa ’80 – ‘90

- Design specific interfaces inside chip
- IC did single tasks so there was need to interconnections between sub-systems with different functionality
- Off-chip buses. Example bus between CPU and DMA
- Networks between computers like ethernet

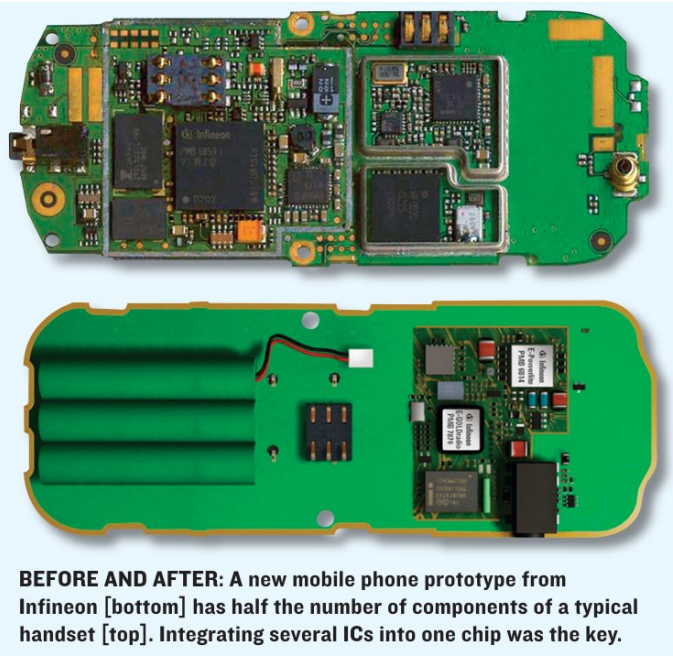


https://www.amigawiki.org/dnl/schematics/A500_R6.pdf

Tampere University

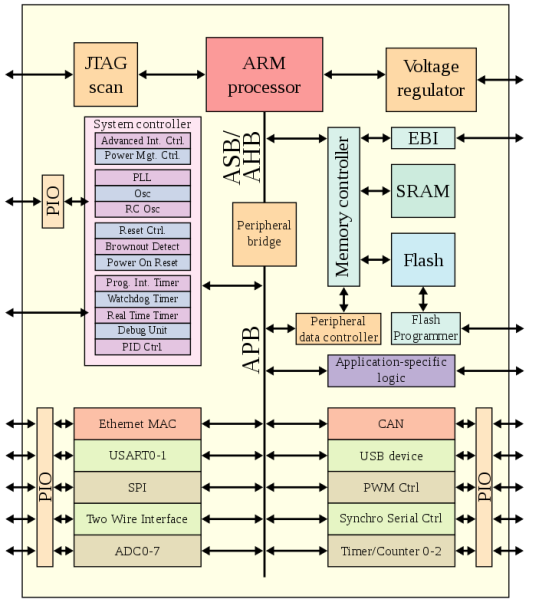
History: early 2000

- SoC designs for single-core systems with integrated peripherals
- Bus topology
- IP reuse and IP business -> Protocol standards
  - ARM Advanced Microcontroller Bus Architecture (AMBA) protocols
  - Open Core Protocol (OCP)
- ~2000: NoCs became hot research topic



BEFORE AND AFTER: A new mobile phone prototype from Infineon [bottom] has half the number of components of a typical handset [top]. Integrating several ICs into one chip was the key.

J. Blau, "Talk Is Cheap," in *IEEE Spectrum*, vol. 43, no. 10, pp 14-15, Oct. 2006, doi:

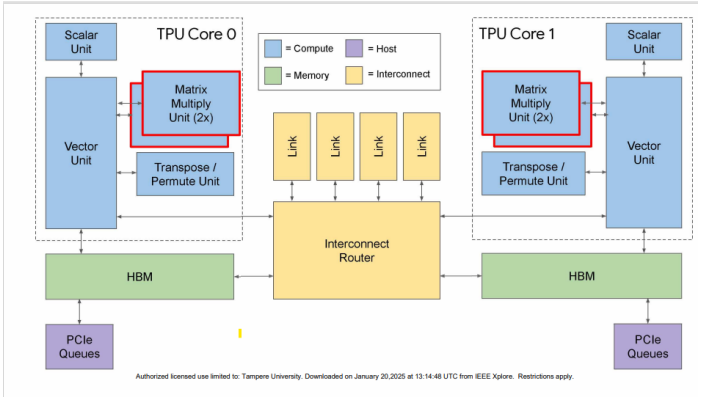


https://en.wikipedia.org/wiki/ARM_architecture

Present

- ARM based systems have huge market share in SoC designs. AMBA protocol family has become de-facto

- Different product categories benefit from different kind of solutions
  - Microcontrollers: Bus topology
  - Heterogeneous multi-core cache coherent SoCs: NoCs, Cross-bar, Bus topologies
  - CPUs: Chiplets -> Die-to-Dies interconnects
  - Datacenters: Chiplets, Cluster interconnects



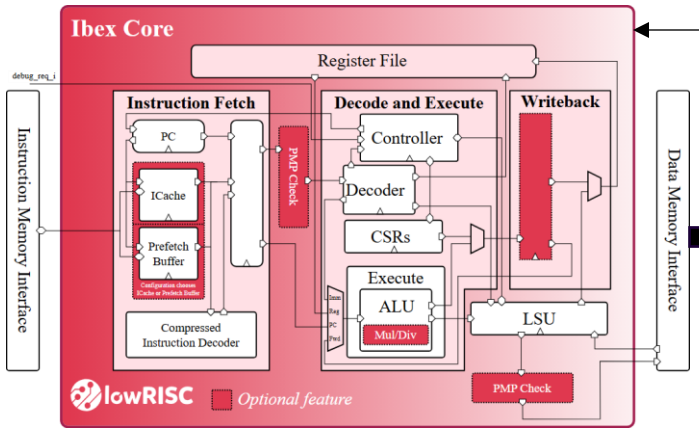
Google's Training Chips Revealed: TPUv2 and TPUv3

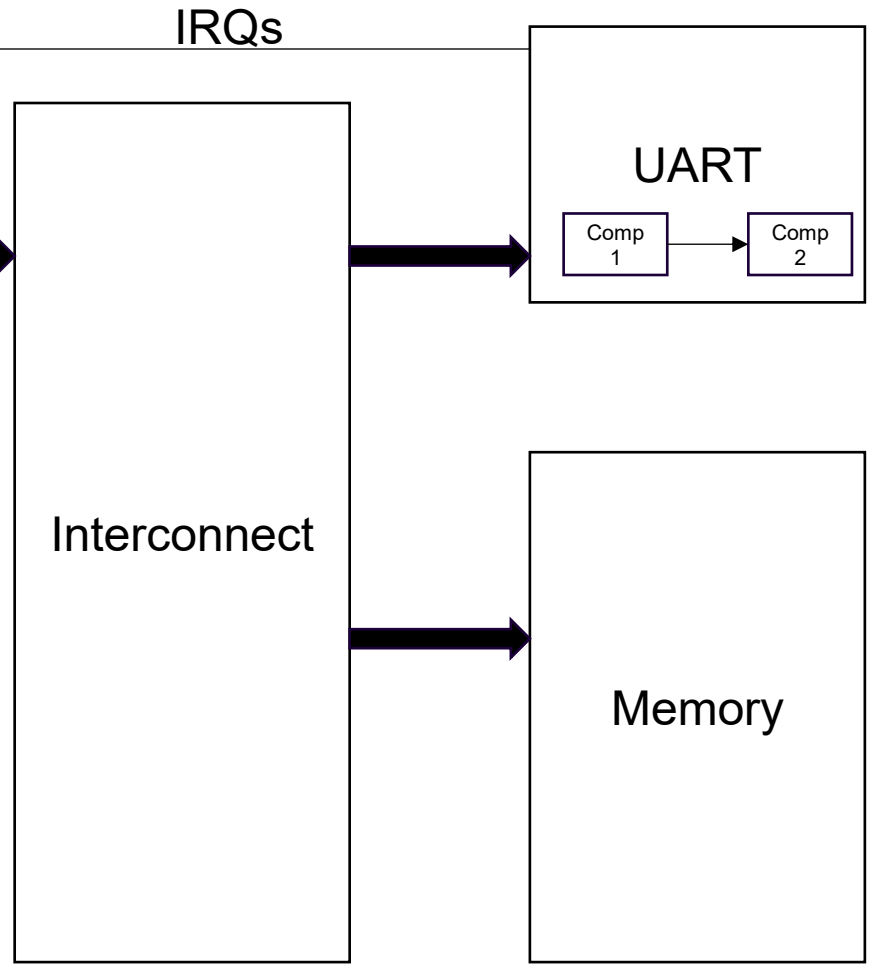
https://www.datamation.com/data-center/what-is-data-center/

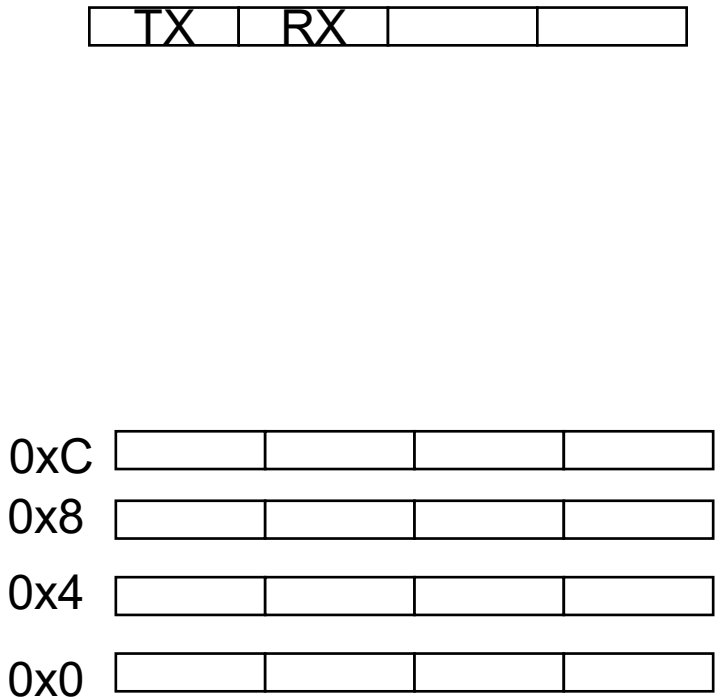
Interconnect basics

Tampere University

What we are interconnecting?

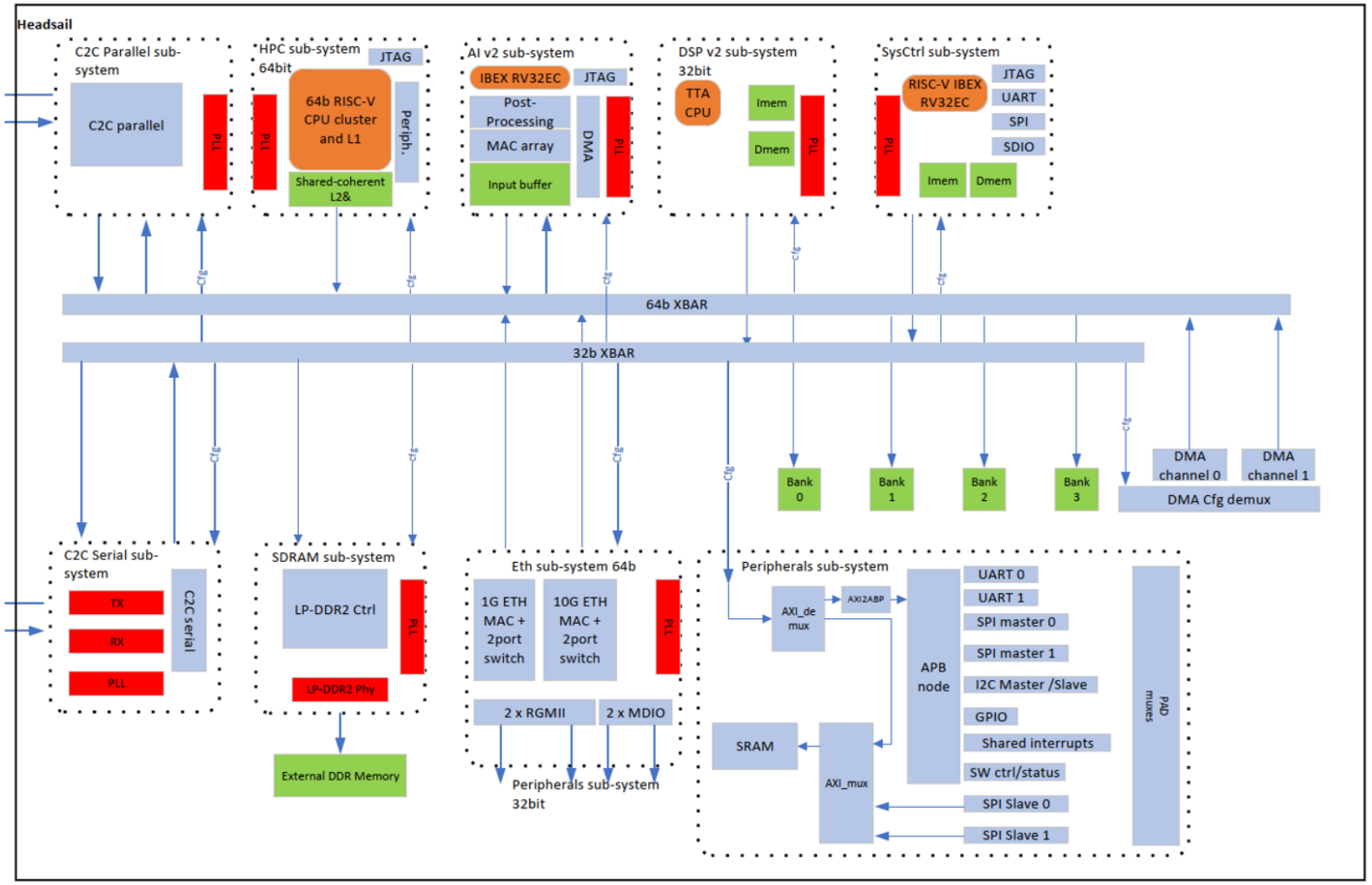






Sources and Destinations

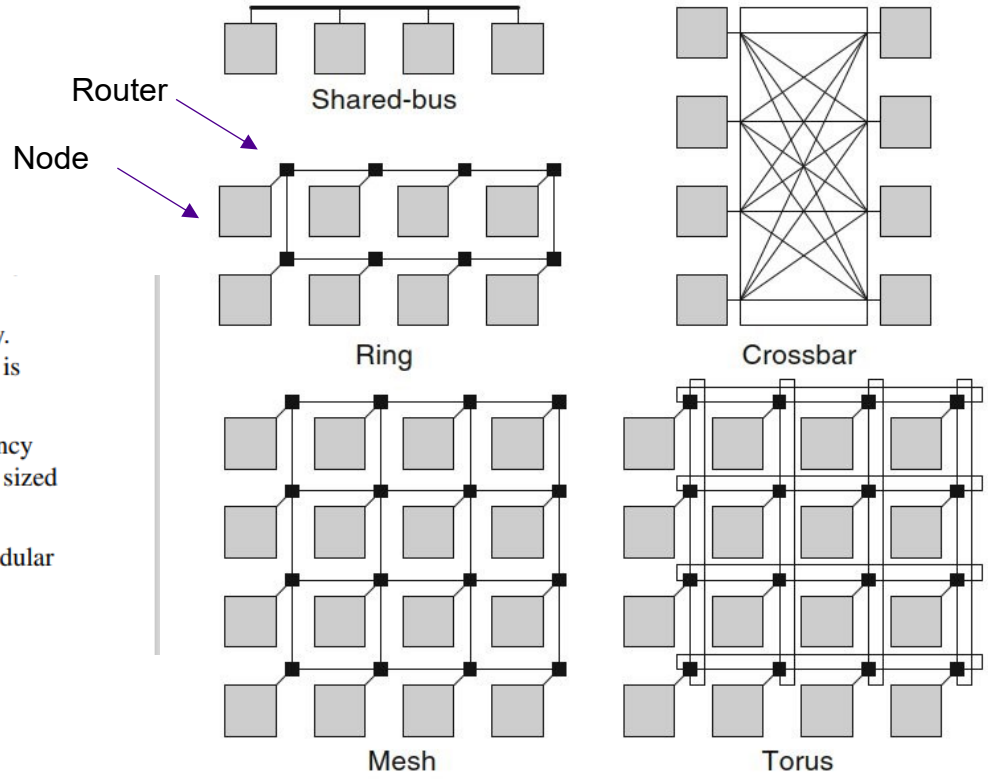
- SoC IPs discussing through interconnect can be divided to *masters* and *slaves*
- *Master is a device which initiates requests. Example writes and reads performed by CPU or DMA*
- Slave is a device which responds to requests. Example by returning the read data.
- One IP can have both type of interfaces.
- Naming conventions vary, but the idea is same. Manager – subordinate, Source – Sink, Initiator - Responder



SoC-hub Headsail SoC

Interconnect topology examples

- Design considerations
  - Bandwidth
  - Latency
  - Scalability
  - Complexity



Crossbar Crossbar topology is simple to build and naturally provides an ordered network with low latency. Crossbar topology is suitable where the wire counts are still relatively small. Crossbar topology is suitable for an interconnect with a small number of nodes.

Ring Ring topology provides a trade-off between interconnect wiring efficiency and latency. The latency increases linearly with the number of nodes on the ring. Ring topology is suitable for a medium sized interconnect.

Mesh Mesh topology provides greater bandwidth at the cost of more wires. Mesh topology is very modular and can be easily scaled to larger systems by adding more rows and columns of switches. Mesh topology is suitable for a larger scale interconnect.

https://developer.arm.com/documentation/ihi0050/latest

Protocols

- Why we need specified protocols?
  - To abstract lower-level behaviour. Buch of logic and wires -> Transactions e.g. writes and reads
  - Agreement of desired behaviour to help communication.
  - Input artifact for design and verification
  - Reusability. Enables architecture, vendor and technology free implementations
  - Specifications.
  - Collaboration. Different teams can work on different sub-components which eventually interplay

- Purpose in this slideset is not to cover different protocol features completely. Examples are given to understand similar concepts you can find from different protocols

Protocols

Examples of free and open protocols

- AMBA protocol family by ARM. CHI, AXI, AHB, APB and many more.
  - De-facto in commercial SoCs due to huge market share of the ARM architecture based SoCs.
  - Open-source interconnect component library available. At least for APB & AXI
- Open Bus Interface by OpenHW Group
  - Driven by RISC-V community. May become popular due to increasing popularity of the RISC-V CPUs
  - Open-source interconnect component library available.
- TileLink
  - Originates from research by UC Berkeley. Currently maintained by SiFive company (spin-off from UC Berkeley)
- Open Core Protocol by Accellera.
  - 494 pages!
  - Haven't seen open-source implementations.
  - Maybe have been faded away
- Wishbone by OpenCores
  - Was developed to enable research on SoC interconnects
  - One of the oldest open specifications. From early 2000
  - Open-source interconnect component library available
- CoreConnect by IBM
  - Have been faded away

APB example

- Every transfer take at least two cycles to complete

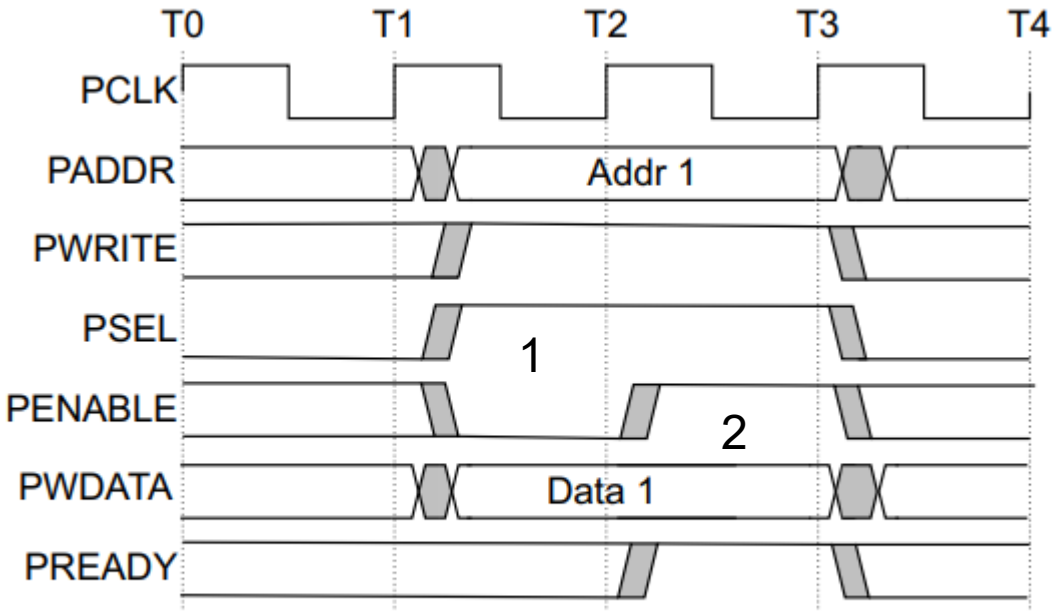


Figure 3-1 Write transfer with no wait states

https://developer.arm.com/documentation/ihi0024/latest/

APB example

- This 2-way handshaking implements flow control on the bus
  - Master: PSEL & PENABLE
  - Slave PREADY
- Slave can use PREADY to control Master to wait

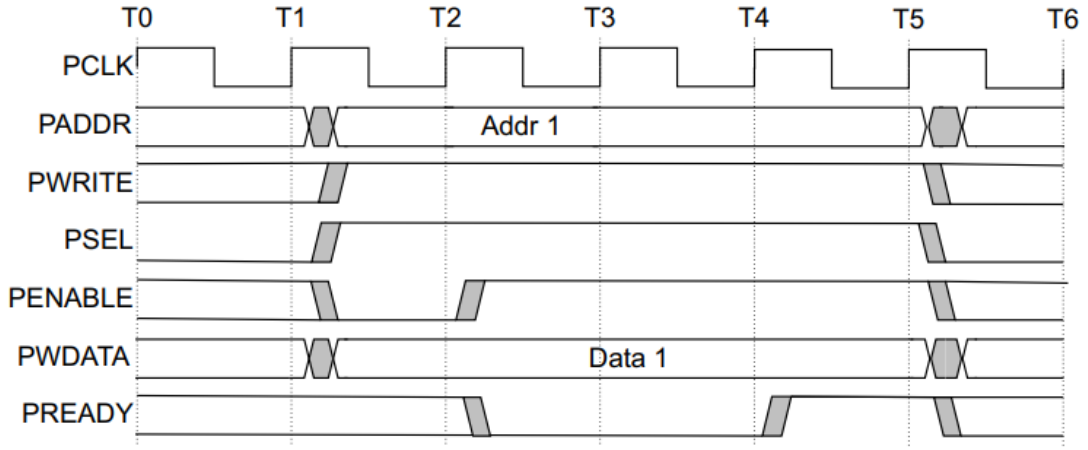


Figure 3-2 Write transfer with wait states

https://developer.arm.com/documentation/ihi0024/latest/

APB Example

- PSTRB indicates which byte lanes to update during a write transfer. There is one write strobe for each 8 bits of the write data bus.
- This way example CPU can perform byte (8bit) and half-word (16bit) operations
- Many other protocols have similar concept. In OBI it's called Byte Enable (BE)

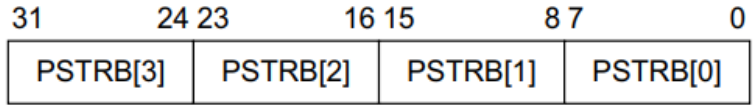


Figure 3-3 Byte lane mapping

https://developer.arm.com/documentation/ihi0024/latest/

Tampere University

AXI

- Targeted at high performance, high clock frequency system designs
- Permits address information to be issued ahead of the actual data transfer.
- Supports multiple outstanding transactions.
- Supports out-of-order completion of transactions.
- Supports bursts transfers
- + Many other advanced features
- Each channel has own 2-way handshaking (Valid – Ready)

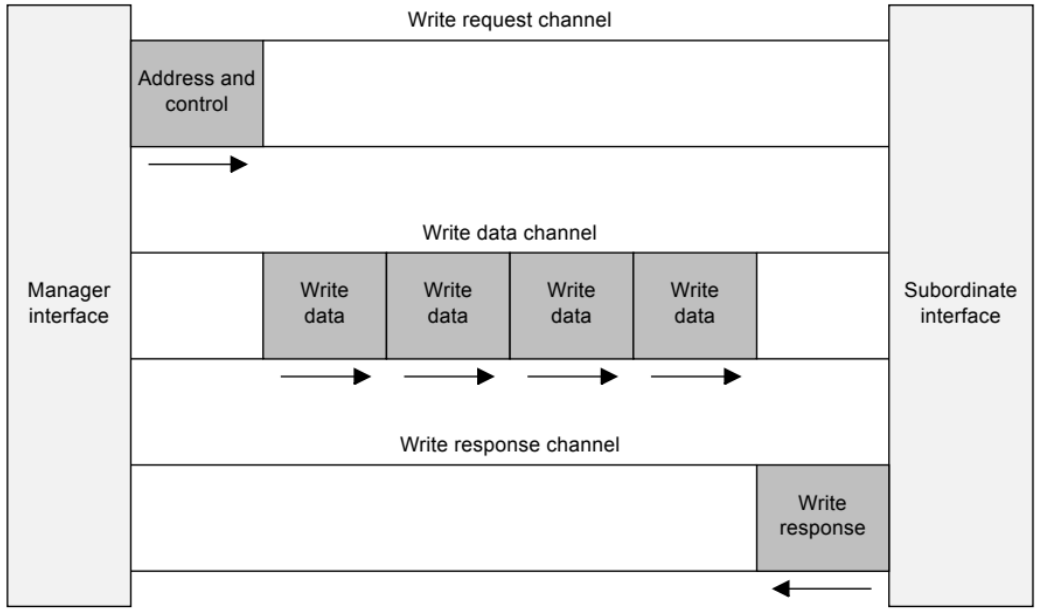


Figure A1.1: Channel architecture of writes

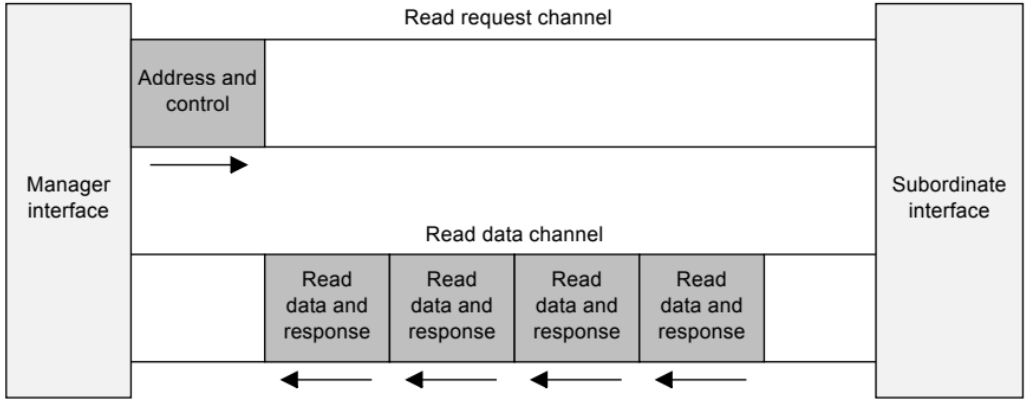
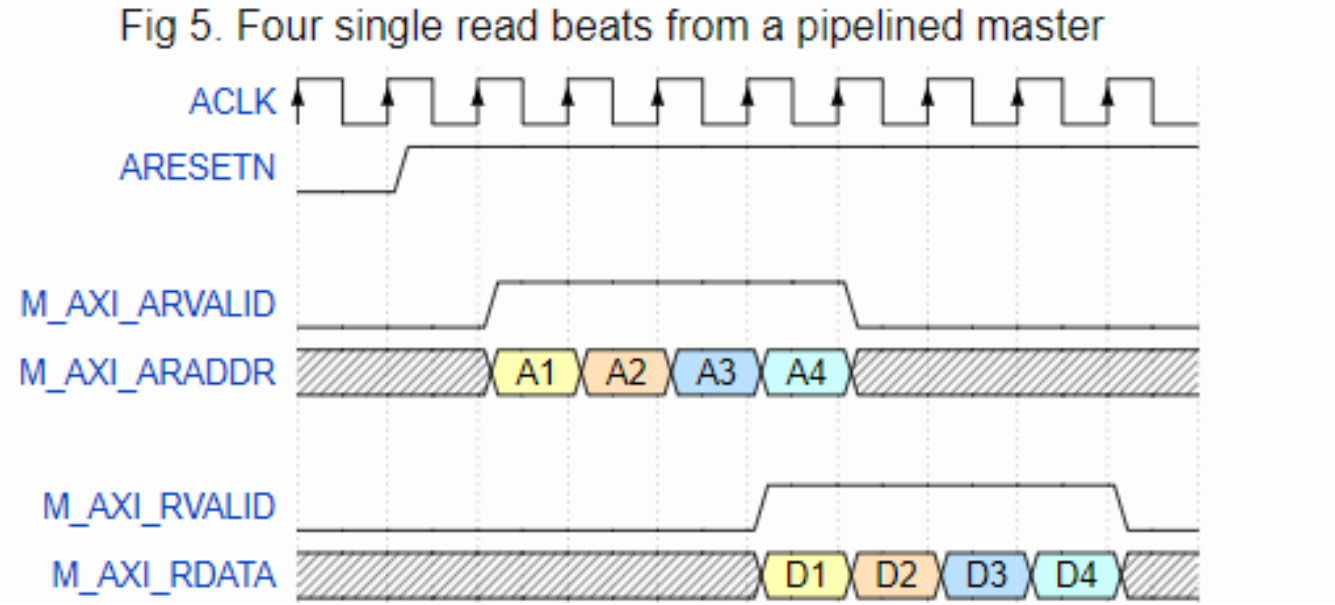


Figure A1.2: Channel architecture of reads

https://developer.arm.com/documentation/ihi0022/latest

AXI Pipelined non-burst operation

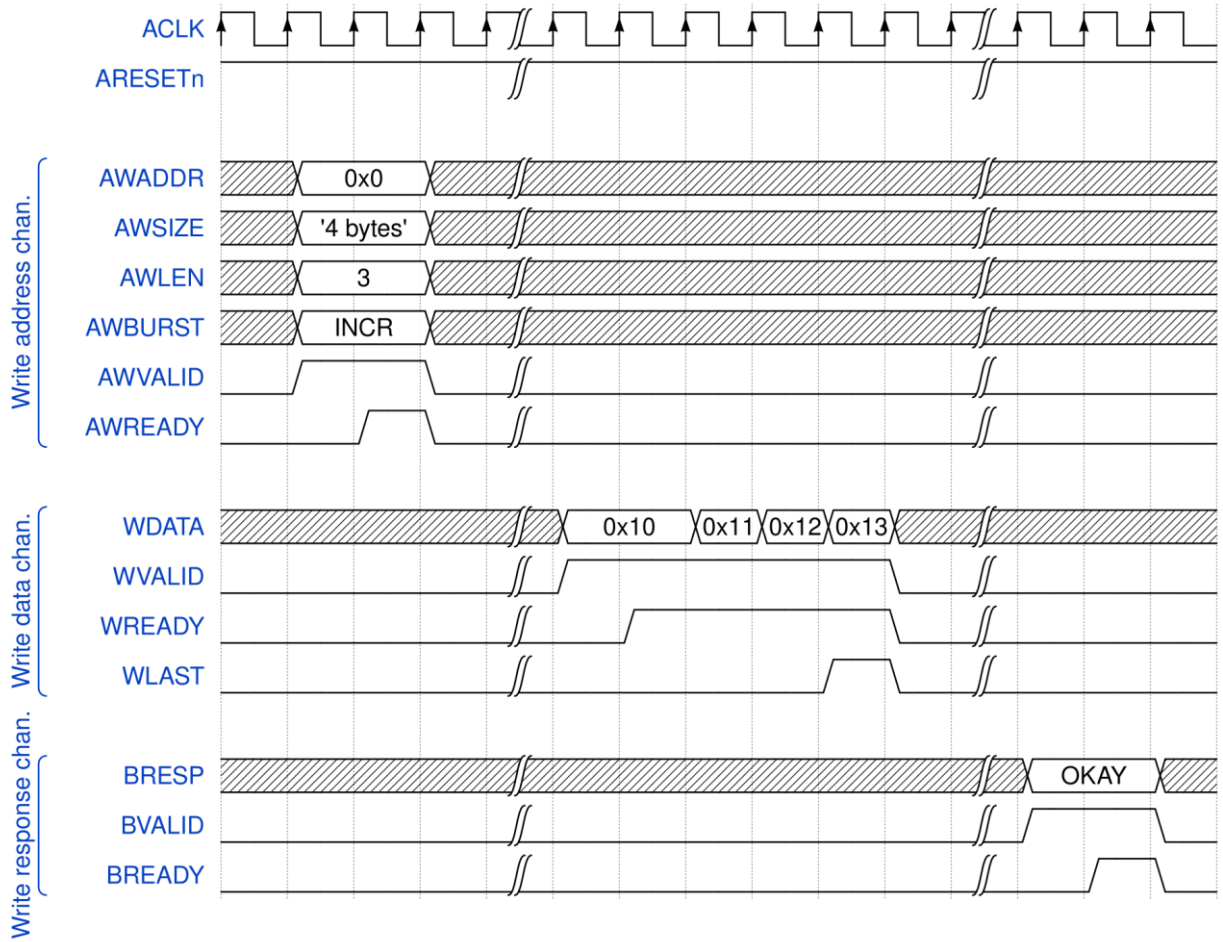
- Master doesn’t need to wait earlier operation to finish to issue next operation.
- This leads to better throughput.
- However, potential issue is that interconnect may arbitrate operations with operation coming form the other master which causes increased and unpredictable latency



https://zipcpu.com/blog/2020/03/23/wbm2axisp.html

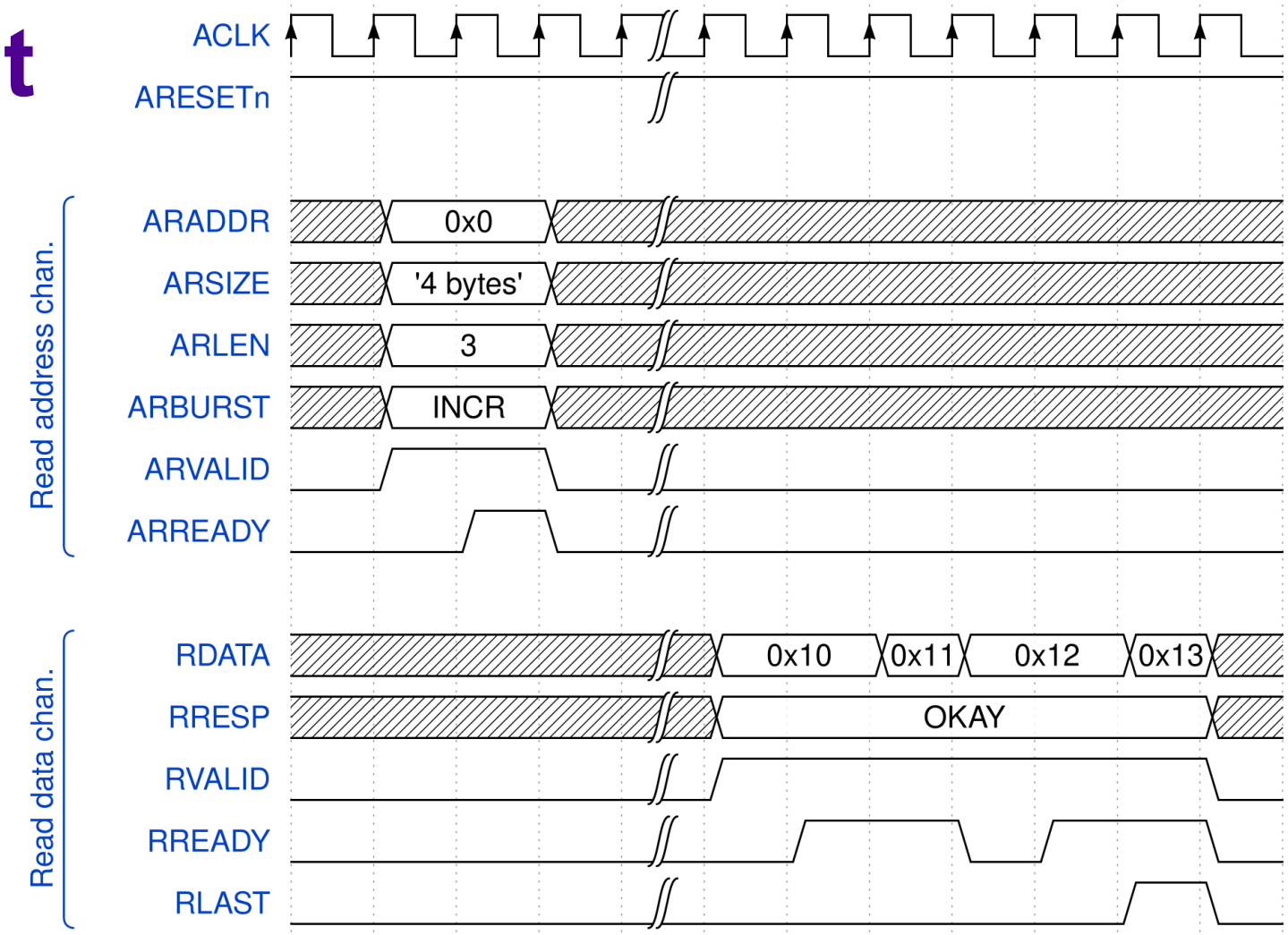
AXI Write Bursts

- Purpose of the burst is to increase bandwidth. Less time is spent on handshaking.
- In AXI there is 4 signals related to write bursts
  - AWSIZE:
    - Indicates that how many data bytes are active in transfer.
    - Increment of the address within burst.
  - AWLEN: Length of the burst
  - AWBURST: Burst type. Fixed, Incremental, Wrapping
  - WLAST: Indicated last beat of the burst



https://en.wikipedia.org/wiki/Advanced_eXtensible_Interface

AXI Read Burst



https://en.wikipedia.org/wiki/Advanced_eXtensible_Interface

Network on chip basics

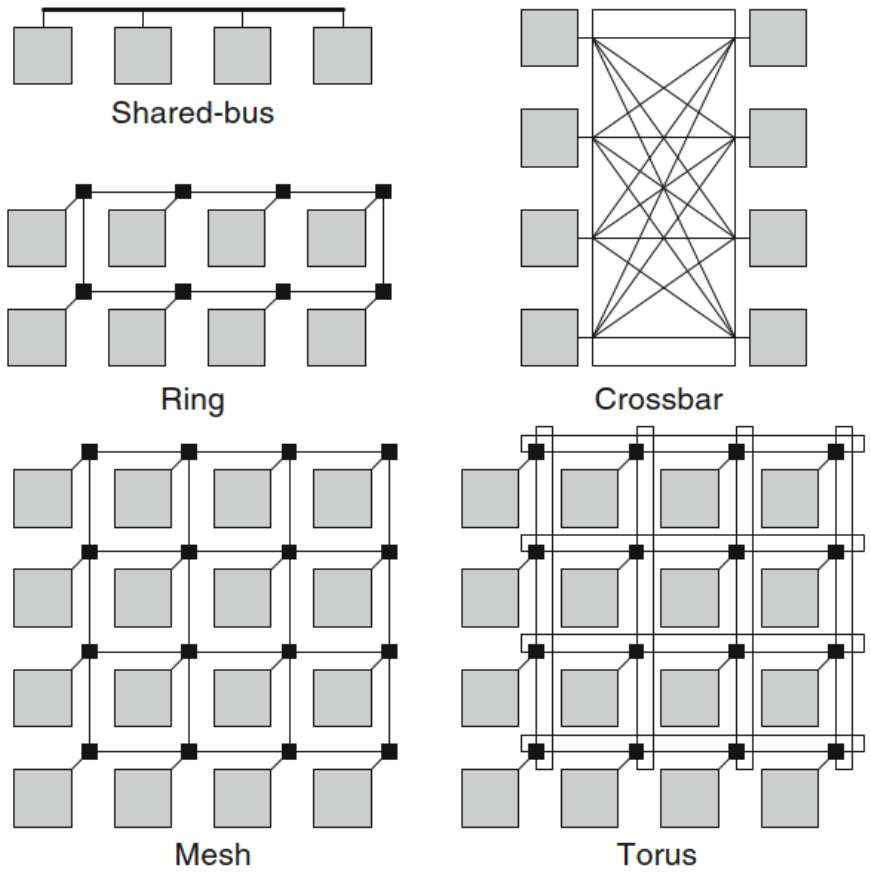
Networks on Chips (NoC)

- The need for NoCs arise from parallel multi-processor communication.
- Holistic NoC approach tries to offer interfaces, protocols and network implementation to carry all these efficiently through unified interconnect architecture.
- NoCs are packet based. To reduce number of needed wires packets are typically split to flits.

- What is the data we are transferring inside SoC? [1]
  - *Signaling:* Interrupts, Status, Control
  - *Data streams:* Real-time, high bandwidth such as fast ADC data for radio data, video etc.
  - *Memory and IO access:* Write and Read operations to manipulate memory mapped devices. Cache operations.
  - *Block transfers:* DMA transfers, Cache operations

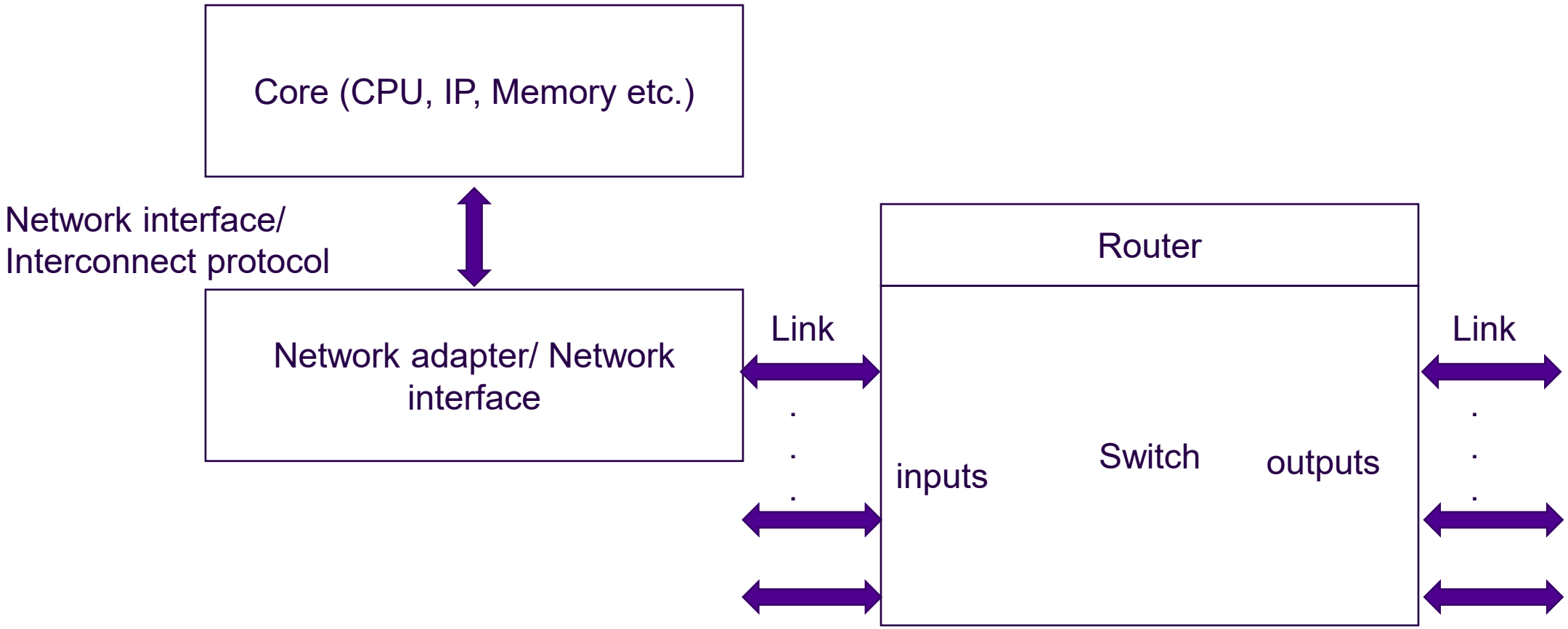
[1] Benini, Luca, and Giovanni De Micheli. "Networks on Chips."

NoC topologies



- Each of the topology has limitations and other considerations
- Hybrid solutions are also common. Complex, but efficient
- The component and layering concepts allows separation of NoC design topology from the network component design.

Conceptual components of NoC



Network layering: The concept

PROCEEDINGS OF THE IEEE, VOL. 71, NO. 12, DECEMBER 1983

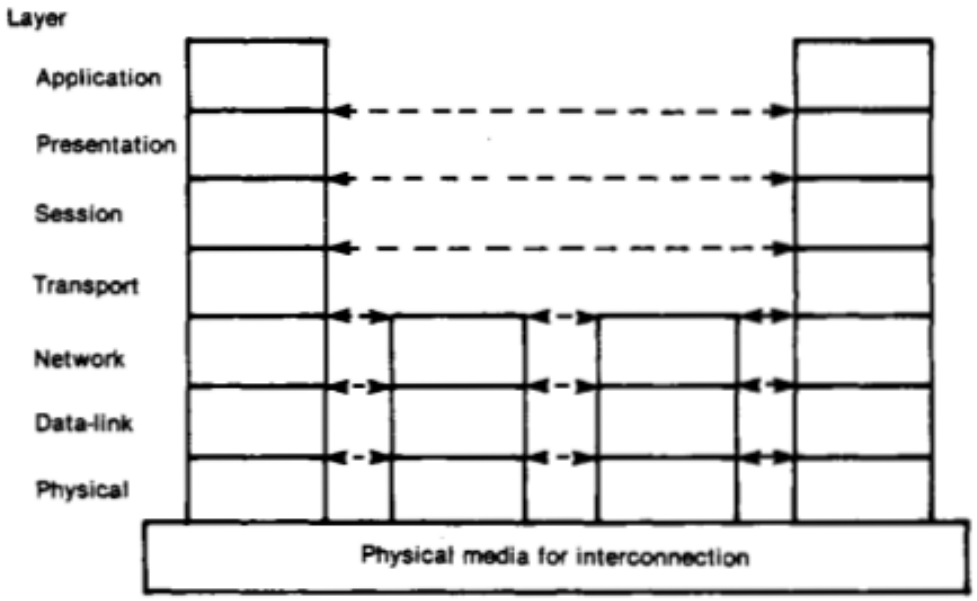


Fig. 9. The seven-layer OSI architecture.

[1] NoC stack fig 1.10

Software
Application
Operating system

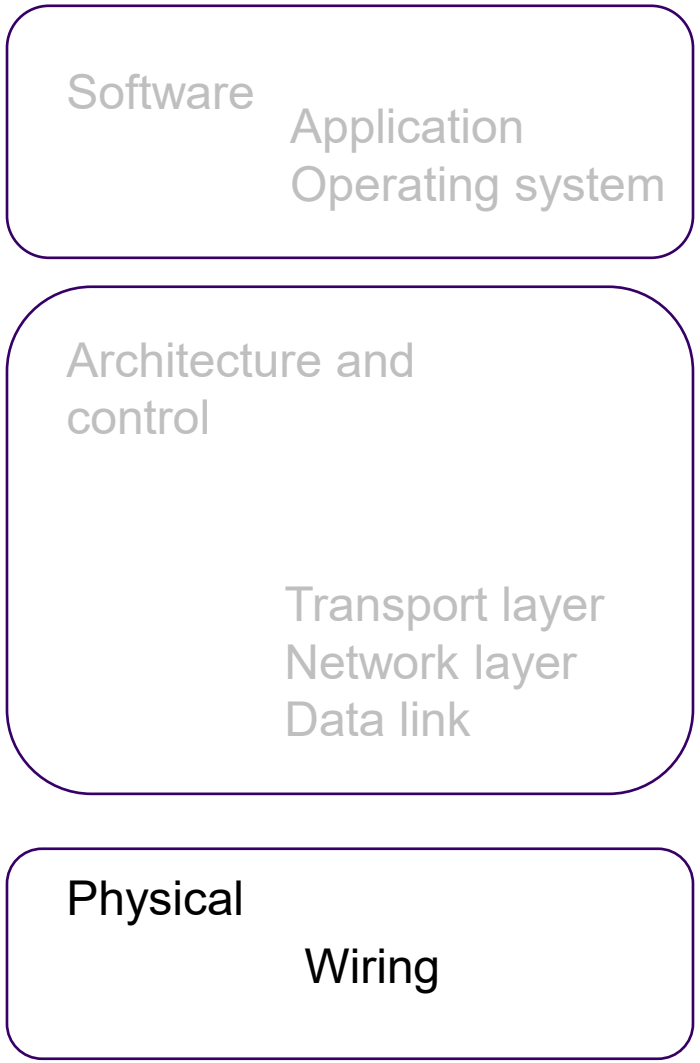
Architecture and control
Transport layer
Network layer
Data link

Physical
Wiring

Network layering: Physical

- Physical Layer: With digital IC design almost fully automated by ASIC tools
  - Unidirectional wires
  - Standard cells e.g AND, OR, Inverter, Buffer as a source/tx and sink/rx

- Despite automation naturally phenomenons on physical layer causes the actual desing limitations
  - Performance
  - Power
  - Area



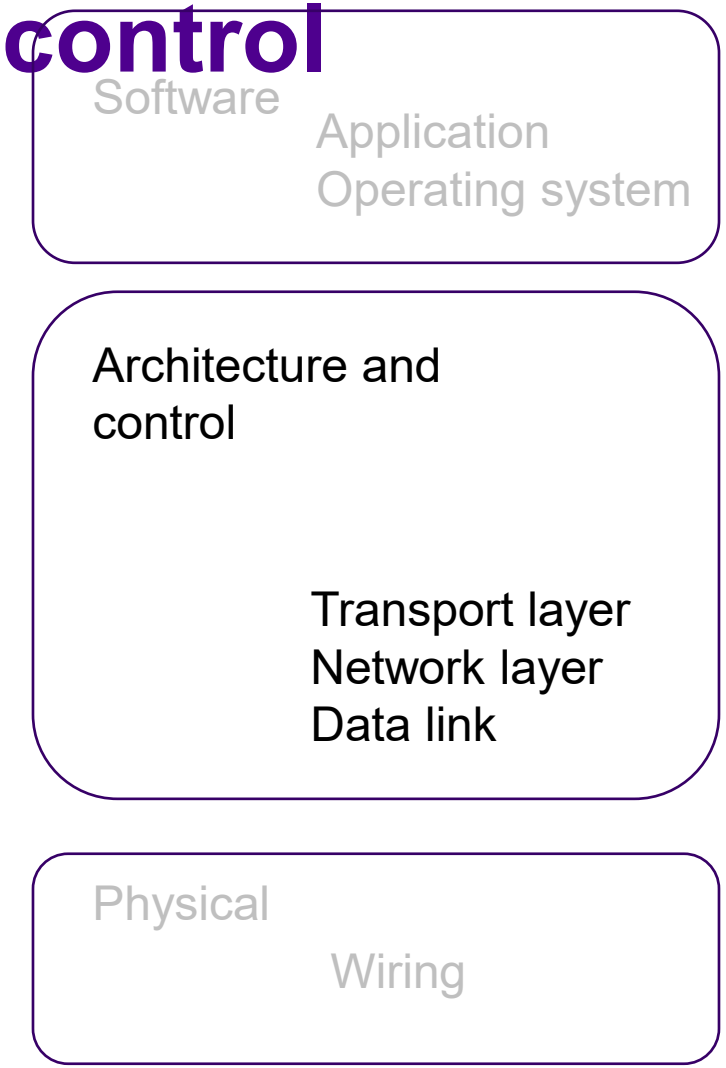
[1] fig 1.10

Network layering: Architecture and control

- Data-link layer
  - Gives the definition for *Link* e.g datawidth of the link
  - Regulates access to shared medium (Medium Access Control)
    - Arbitration and handshaking,
  - Error detection and correction (Data-Link Control)

- Network layer
  - Switching
  - Routing

- Transport layer
  - Defines transactions/messages
  - Decomposition of the transactions to packets

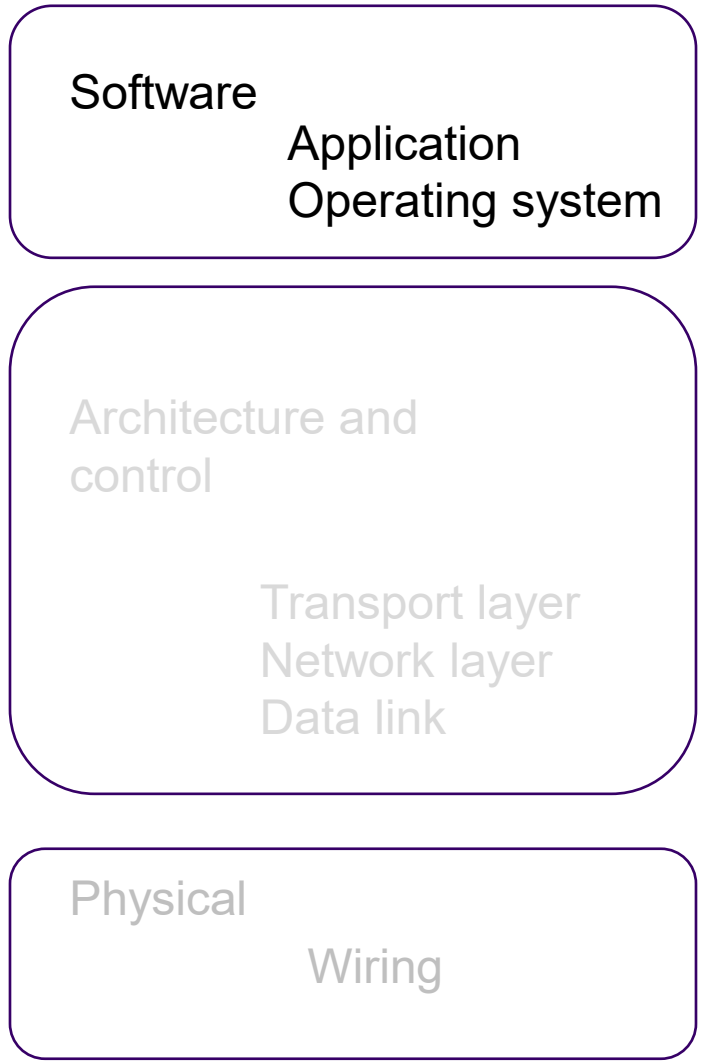


[1] fig 1.10

Network layering: Software

- Operating system
  - Hardware Abstraction layer (HAL), Programming API
  - Software related to NoC configuration
  - Software related to memory management

- Application
  - Although purpose of this layer is to hide the details of the underlaying system the application is essential when defining NoC specification (SW/HW Codesign).
  - Application -> Latancy, bandwidth and power requirements
  - Requirements -> NoC topology and protocol configurations

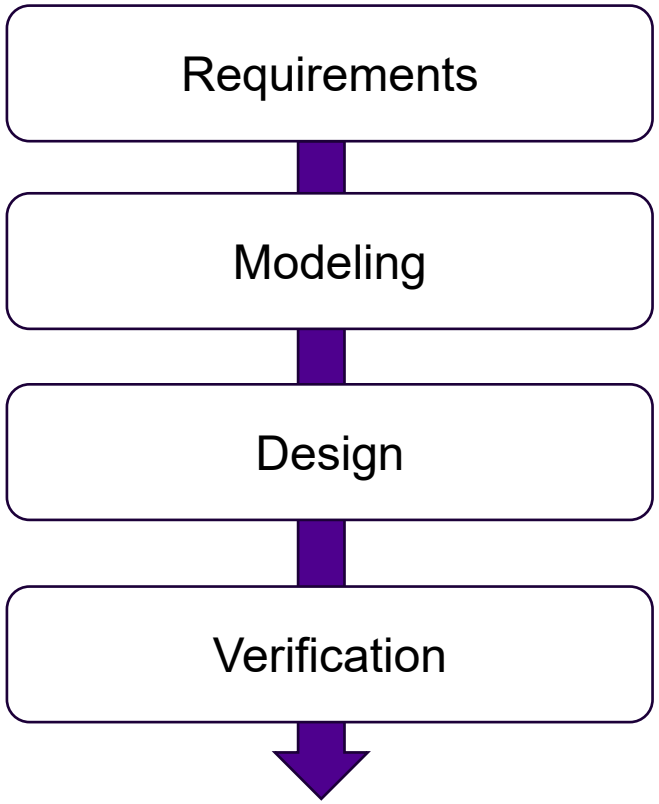


[1] fig 1.10

Interconnect design

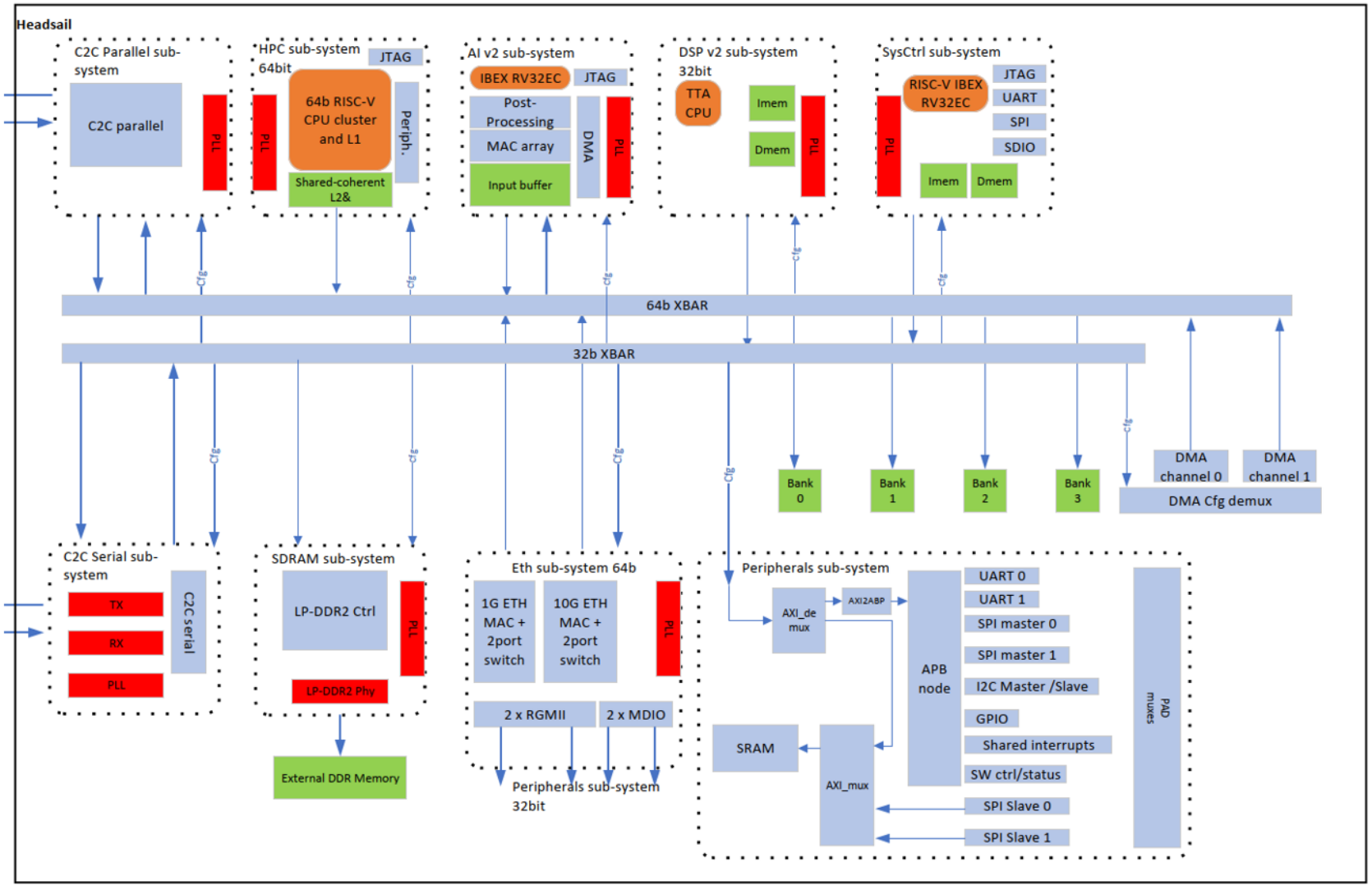
Requirements and modeling

- Interconnect architecting goes hand in hand with system architecting
  - Number of CPUs, Accelerators, Memories etc.
  - Performance (Bandwidth, latency), Power & Area requirements
- In the best case there is a executable high-level model (SystemC, C++ etc) to explorate design space
- Outcome of the modeling is a architectural specification for interconnect design.
  - Interconnect topology, protocols, datawidths, clock-domains, target clock-frequencies…



Tampere University

Architectural specification

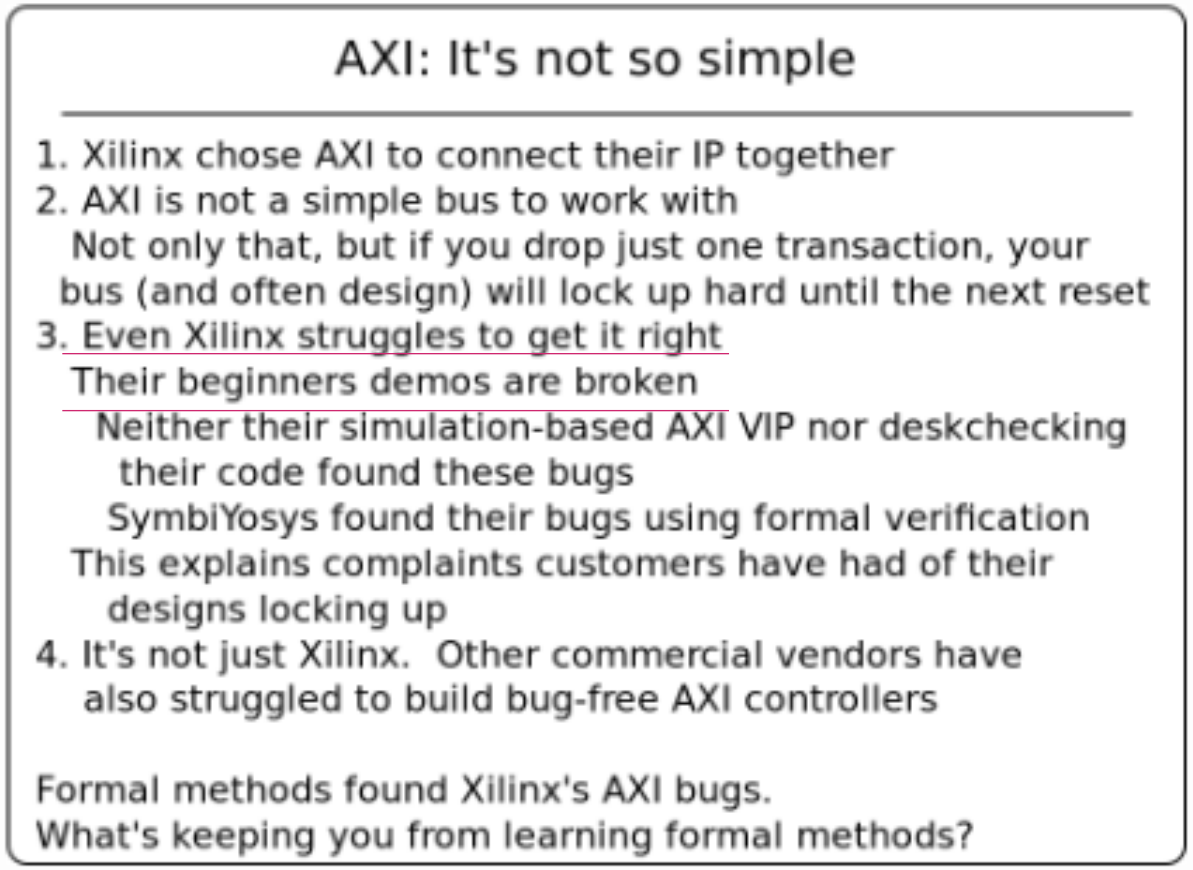


SoC-Hub Headsail SoC

Design: Reuse

- Fundamental components needed to build interconnects are fairly similar
- Don’t try to re-invent the wheel
- Component libraries
  - Synopsys Designware, commercial
  - ARM AMBA components, commercial
  - PULP AXI, open-source
- Typically these comes with reference testbenches

Fig 1. AXI is not a simple protocol

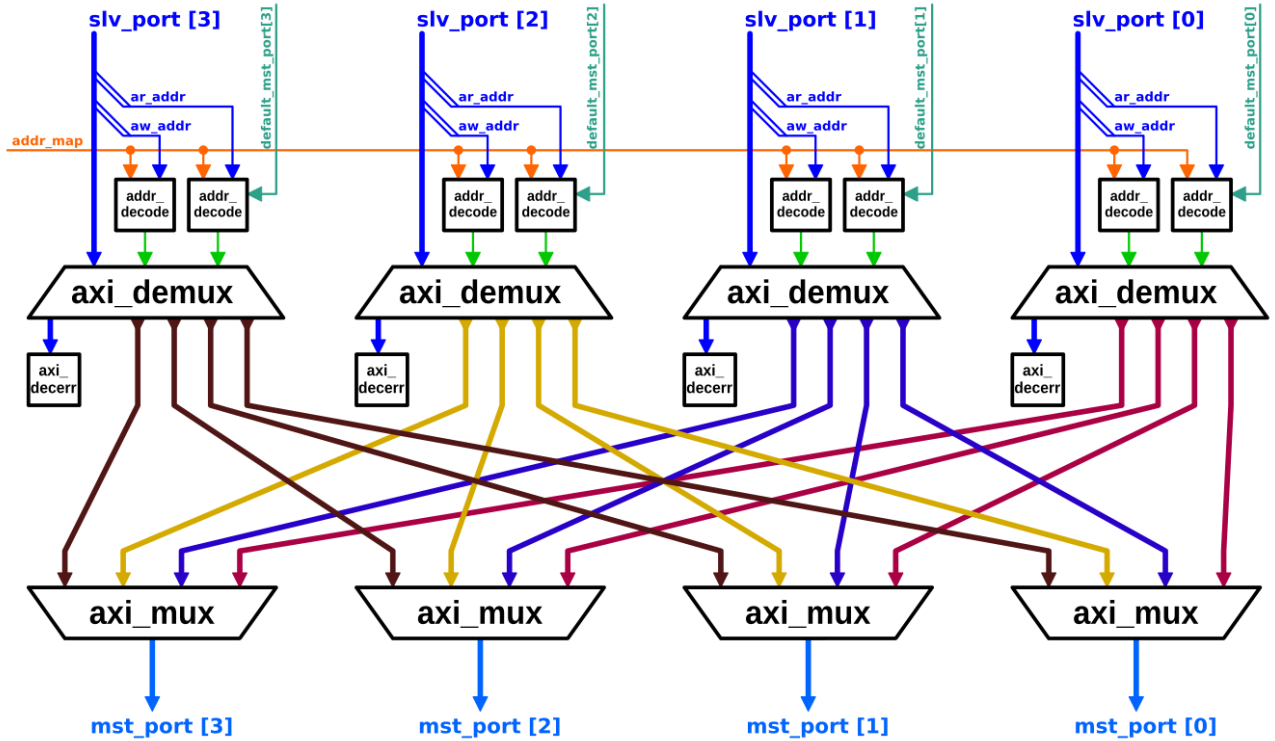


https://zipcpu.com/blog/2020/03/23/wbm2axisp.htm

Interconnect components

- General
  - Muxes
  - Demuxes
  - Cross-bars
  - Datawidth converters
  - Protocol converters
  - Clock-domain-crossing (CDC)

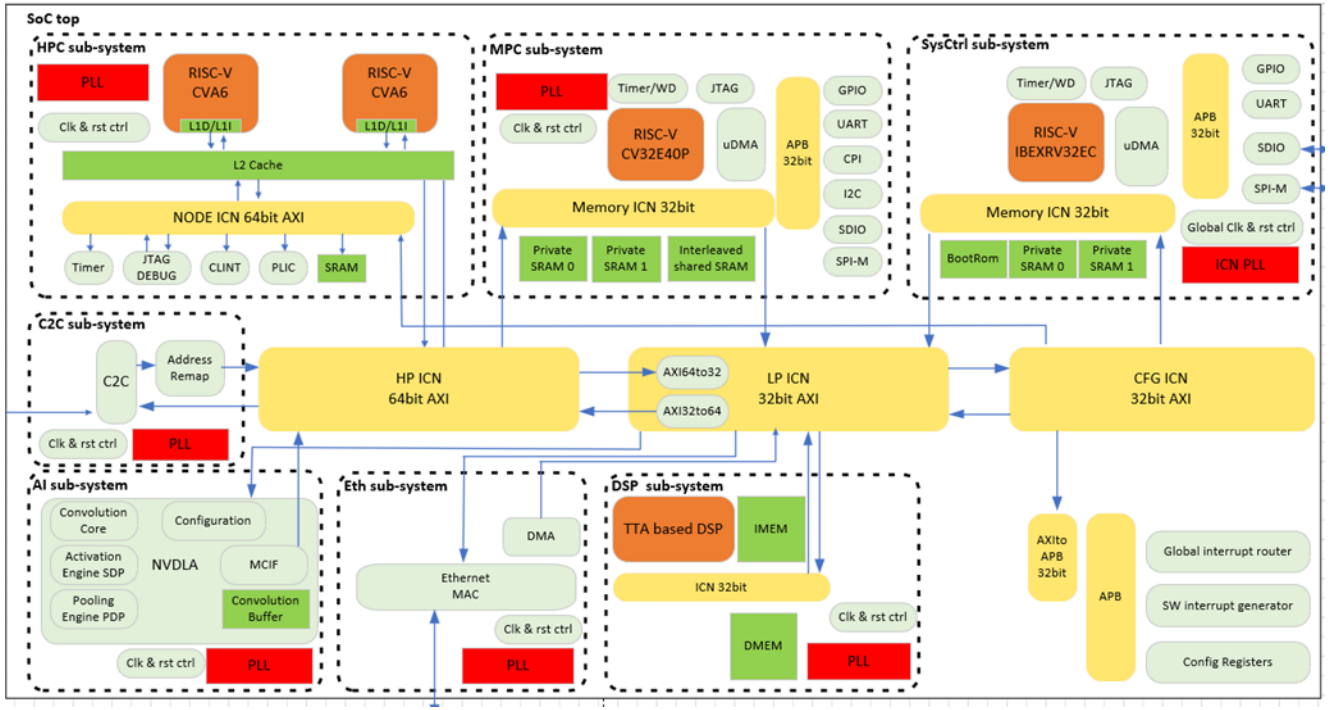
- NoC related
  - Routers
  - Switches



Interconnect verification

Hiearchical design and verification

- Sub-systems - IPs - Components.
- Component example: UART with APB interface.
- IP example. RISC-V processor. Adaptation to sub-system interconnect.
- Sub-system example: Interfaces toward top-level interconnect.
- Top level interconnect. Interconnections between sub-systems, arbitration.
- In Globally Asyncronous, Locally Syncronous (GALS) systems Clock Domain Crossing (CDC) is important part of the interconnect.

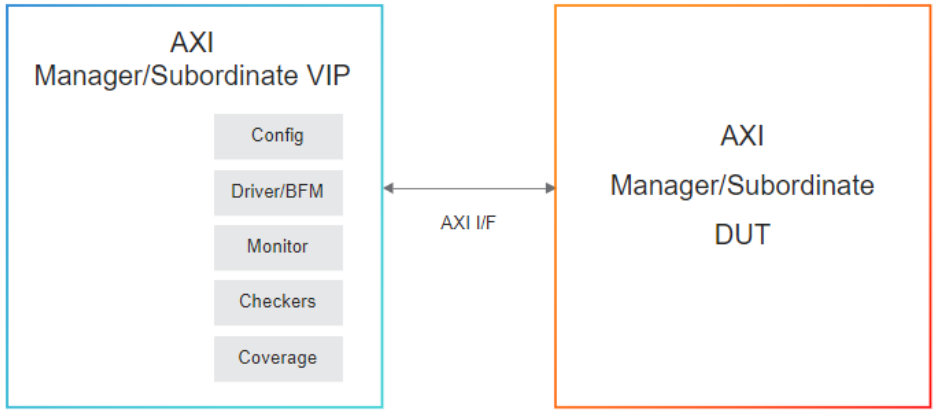


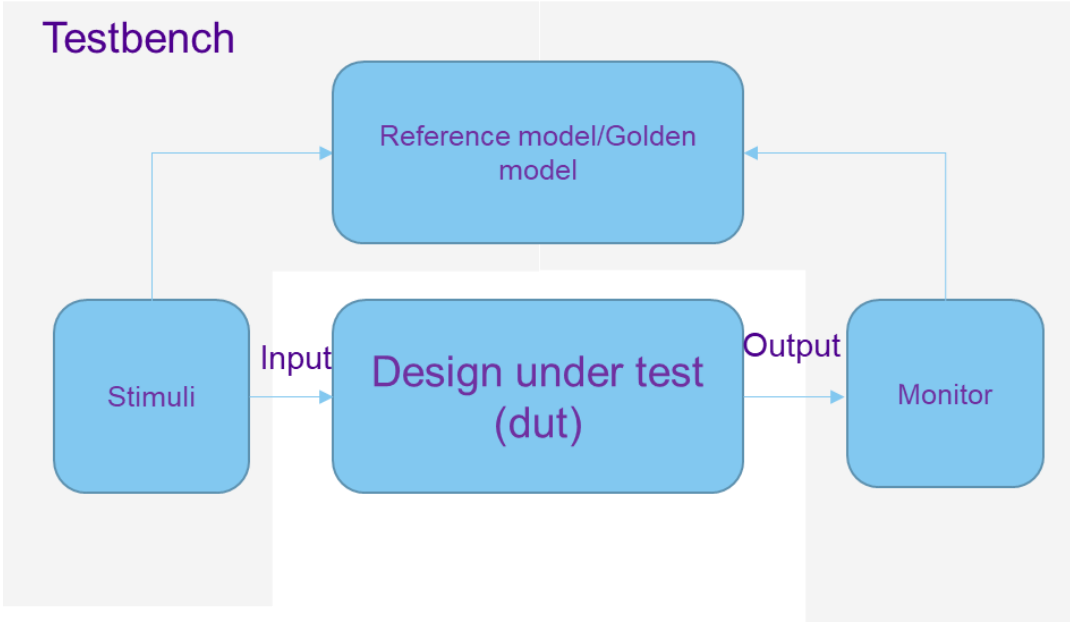
Verification planning

- In short the main questions are:
  - Component verification:
    - Does my implementation fullfil protocol specification?
    - Does it offer desired performance?
    - Does it fullfill above in all corner cases? For example what happens when component's internal FIFO comes full?
    - Does if fullfill above under all valid parameters (parameterizable component)
  - Interconnect verification
    - Does my implementation fullfil protocol specification? (If you reuse verified interconnect components there is a better change that it will)
    - Can all specified master connect all specified slaves (remember the purpose of the interconnect)
    - Does it offer desired performance?
    - Corner cases. Example CDC with different frequencies, what happens if FIFO becomes full on receiving side (Does handshaking interplay)
  - And many others which I forgot to mention ☺

Interconnect verification components

- Protocol VIP (Verification Intellectual Property)
- Modern VIPs are implemented with SystemVerilog UVM





https://www.cadence.com/en_US/home/tools/system-design-and-verification/verification-ip/simulation-vip/amba/amba-axi.html

Verification methodologies

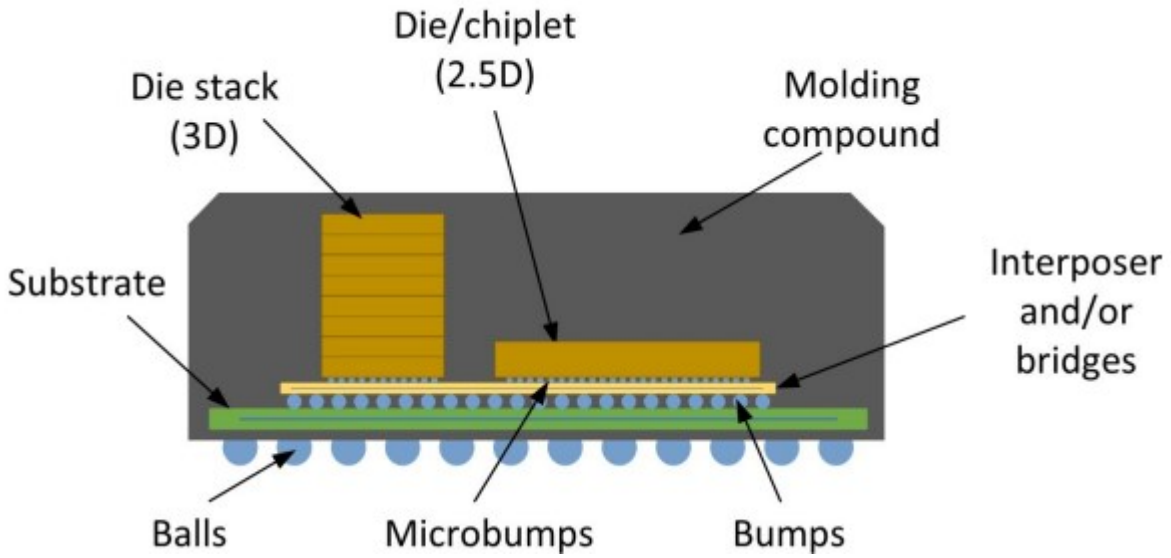
- Directed tests
- Constraint random tests
- Coverage-driven verification
- Formal verification

Special topic

Chiplets

Chiplets

- Different components are produced with the technology node most optimal for them.
- Interconnect design becomes (multi-chip) system problem.
- Substrate is a “PCB” inside a package. Routes die pads to package Balls (or pins)
- Passive silicon interposer is a die which contains only interconnections



Source: Advanced-Packaging-Fundamentals-ebook-2025

Chiplet protocol examples

- Universal Chiplet Interconnect Express (UCIe)
- Bunch of Wires (BoW)
- HBM3 (JEDEC)
- Advanced Interface Bus (AIB, Intel)
- Infinity Fabric (AMD)

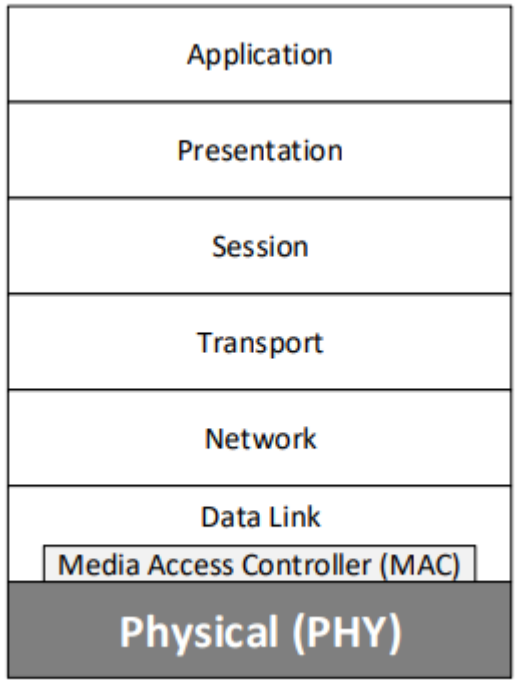


Figure 1. AIB in the OSI Reference Model

Summary

Summary

- Poor interconnect design can ruin system performance. Pay attention to system design prior implementation.
- Protocols are complex and hard to verify.
- Reuse design and verification components from trusted sources.
- Pay attention to verification planning.