Computer Systems Design

Lesson 16

Bus architectures. Networks on chip. Open-source NoCs

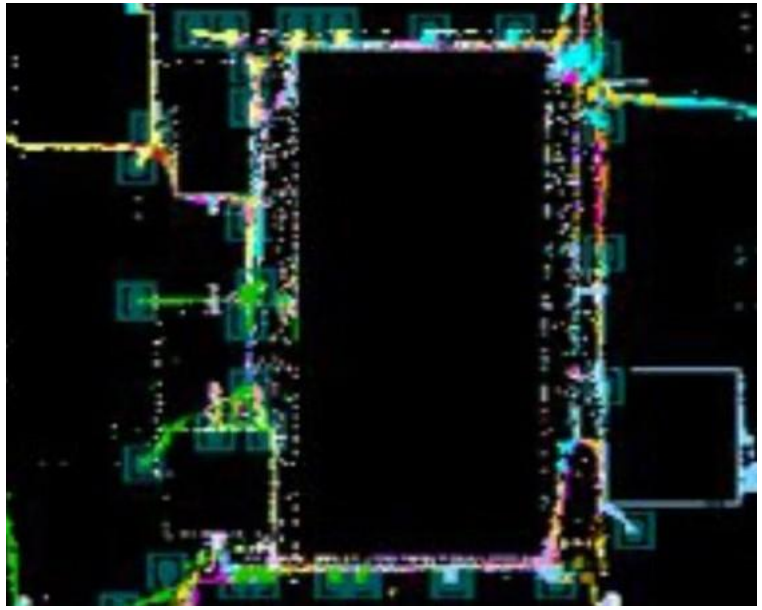
Alexander Antonov, Assoc. Prof., ITMO University

Hangzhou, 2025

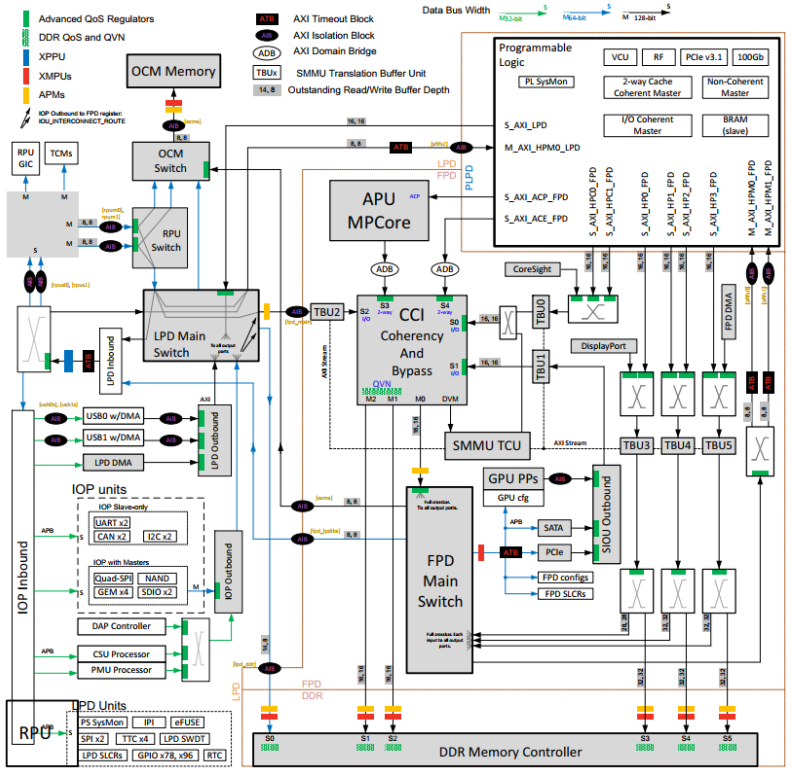
ITMO

Outline the lesson

- Classic bus architectures
- Networks on chip concept
- Common NoC topologies
- Switching
- Flow control
- Routing
- Examples



Example: Xilinx Zynq UltraScale+ MPSoC



Blocks list:

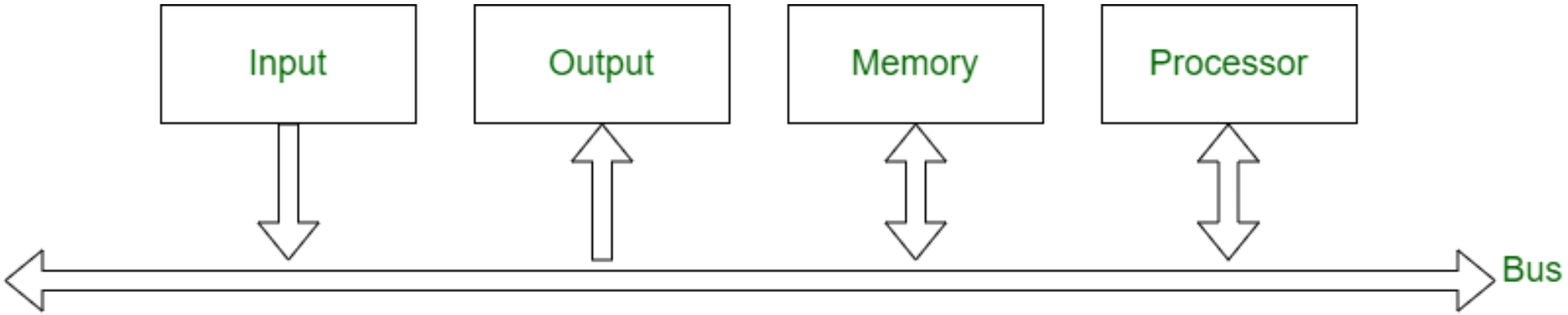
- Cortex-A53 APU (4xCPUs)
- Cortex-R5F (2xCPU)
- Mali-400 GPU
- FPGA (>10 ports)
- DDR controller
- DMA
- High-speed interface controllers: PCIe, SATA, DisplayPort, 2xUSB, 4xEthernet
- Low-speed interfaces: UART, SPI, I2C, CAN, NAND, SD, GPIO, …
- OCM, timers, control registers, …

Dozens/hundred of cores on chip need to exchange data at high speed

UG1085 - Zynq UltraScale+ Device TRM (v2.2), 2020

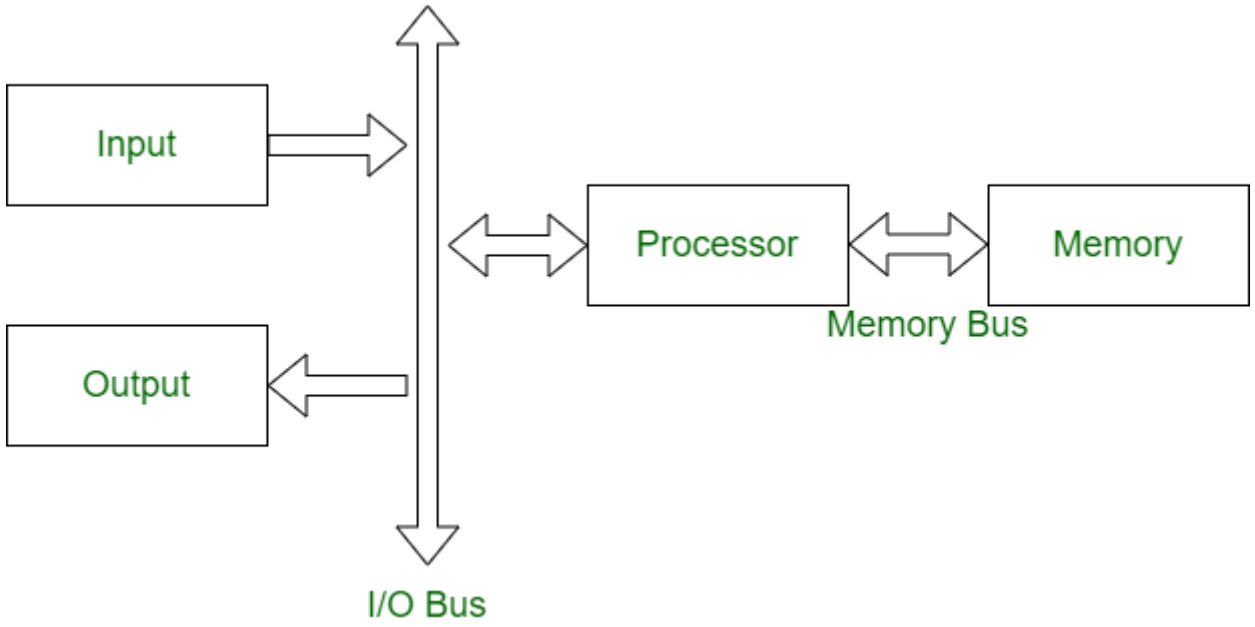
Classic shared bus interconnects

- Uses single physical media for all transfers
  transfer speed limited by the slowest endpoint
- Implements circuit switching
  direct transfers, no intermediate storage of transferred data
  transfer between two active endpoints blocked transfers of all other endpoints
  poor scalability
- Simple protocols, trivial deployment



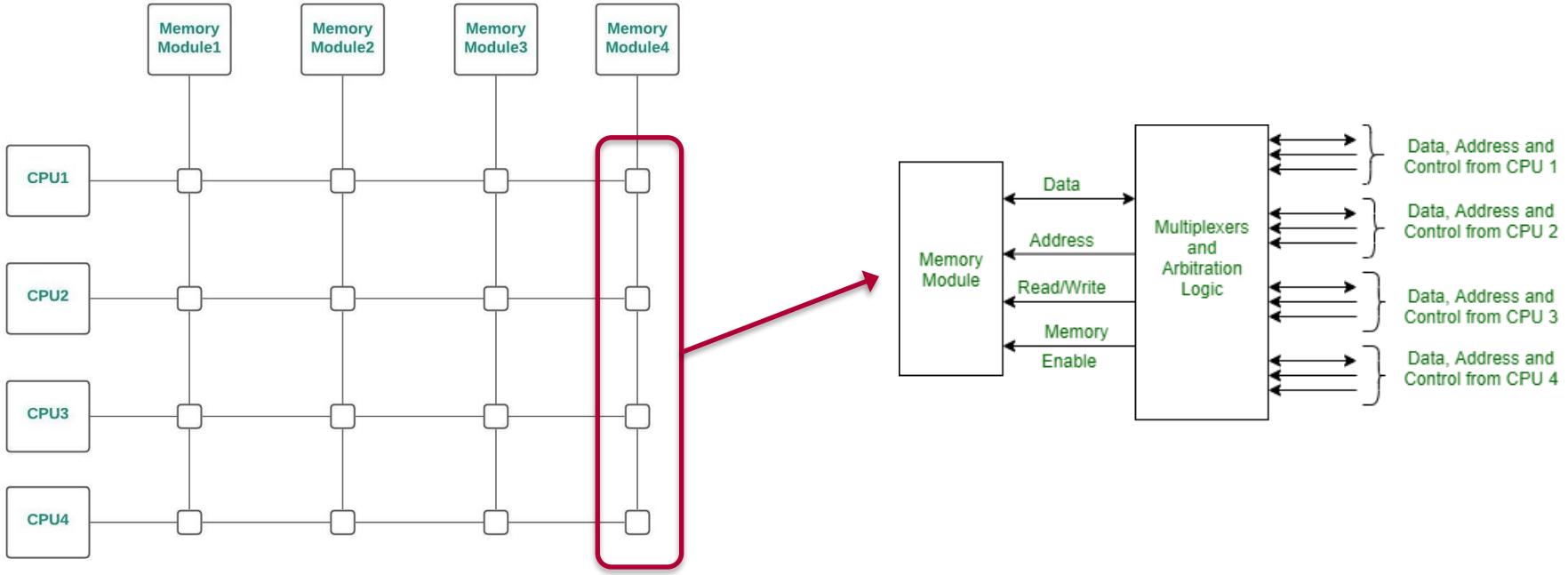
Hierarchical bus interconnects

- Multiple busses for endpoints with multiple speed
  slow endpoints don't limit high-performance endpoints
- Other restrictions remain
  same physical media for each bus, bus locking, limited scalability

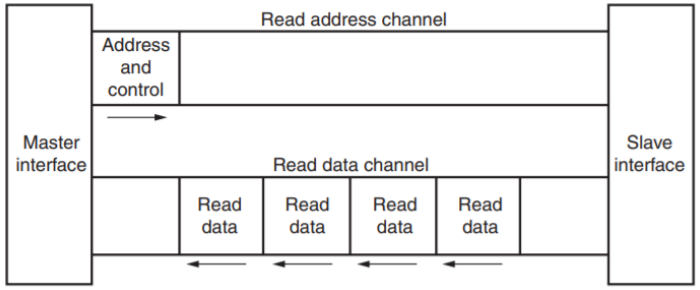


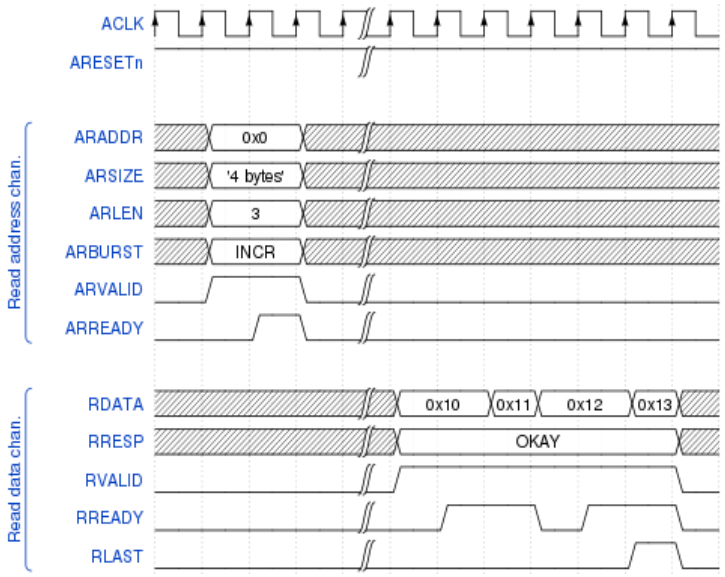
Crossbar switches

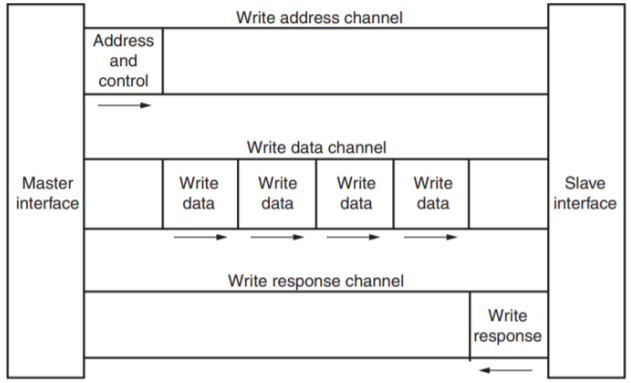
- Multiple point-to-point connections between endpoints
  low capacitance, higher performance
- Each crosspoint controls connection between specific master and slave
  different** endpoints can transfer data **in parallel

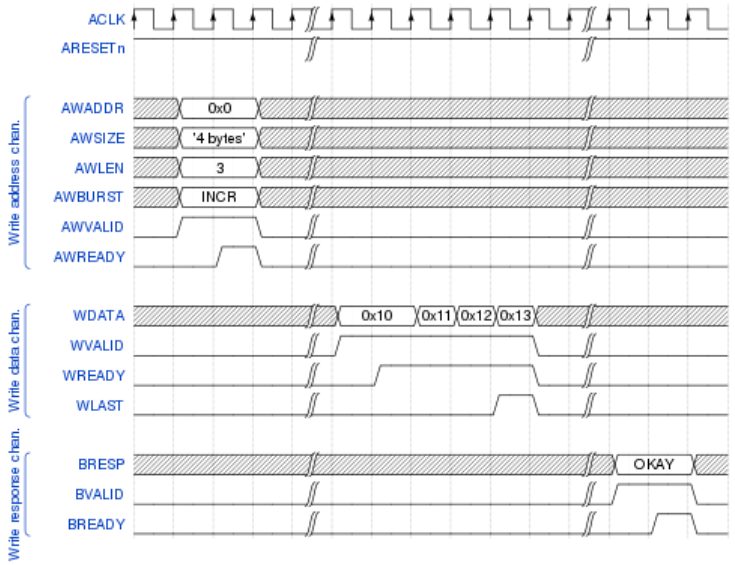


Example of industrial on-chip communication protocol: AMBA AXI4



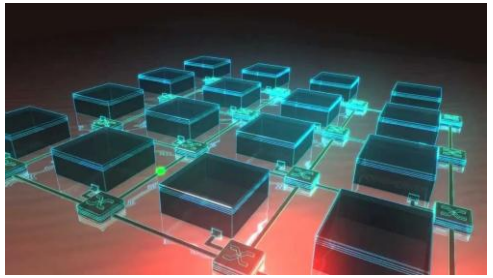






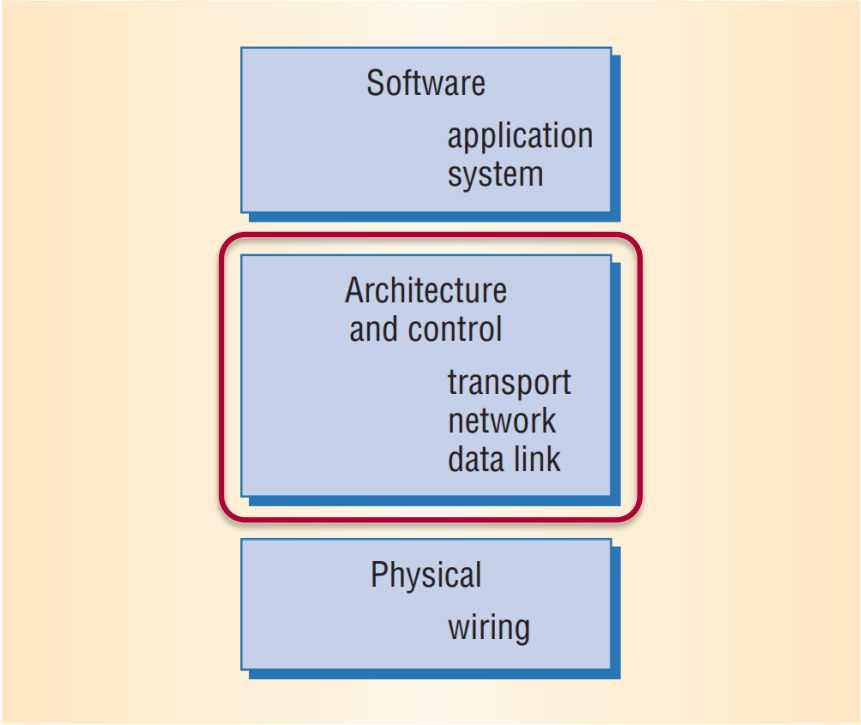
ITMO

Networks on chip (NoC) *



Distinctive features:

- Packet switching
- Point-to-point connections
  No shared bus bottlenecks
- Distributed data transfers
  Scales with increasing network size
- Distributed routing
  Scales with increasing network size
- Multi-level protocols
  ***Deadlock avoidance**, QoS, error correction functions*



** Luca Benini and Giovanni De Micheli. Networks on Chips: A New SoC Paradigm. Computer 35, 2002, pp. 70–78.*

ITMO

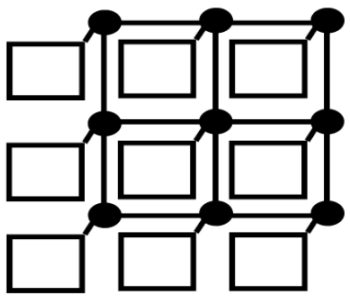
Busses vs. NoCs

Buses			NoCs
Each item will add a parasitic capacitance, hence the degradation of electrical performance increase with the number.	-	+	The elements are connected by point to point interconnections for all sizes of networks, local performance is not degraded.
The time management of the bus is difficult.	-	+	The data transfer can be accomplished by delayed transitions because the connections are point to point.
Delays caused by arbitration at the bus can cause blockages, especially if the number of masters is important.	-	+	Routing decisions are distributed.
The bandwidth is limited and shared by all elements of the bus.	-	+	The bandwidth increases with the size of the network.
The bus tests are long and problematic.	-	+	The dedicated BIST (Built In Self Test) are locals, complete and fast.
The bus latency is the speed of a connection control circuit if the bus is granted.	+	-	Internal decisions making can add delays.
The concept is simple and easy to understand.	+	-	Designers need upgrades in order to exploit new concepts.

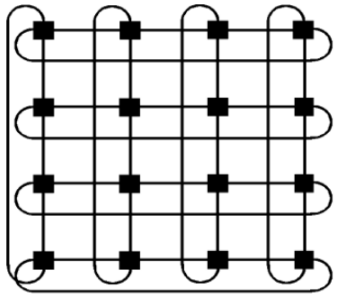
A.B. Achballah, S.B. Saoud. A Survey of Network-On-Chip Tools. International Journal of Advanced Computer Science and Applications, Vol. 4, No. 9, 2013

ITMO

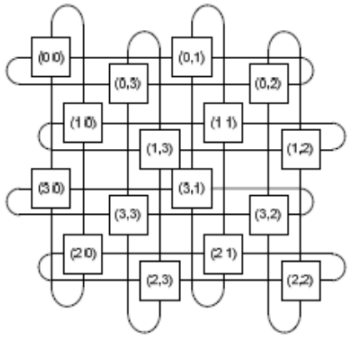
Common regular NoC topologies



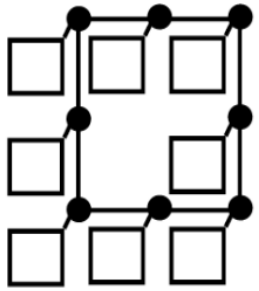
Mesh



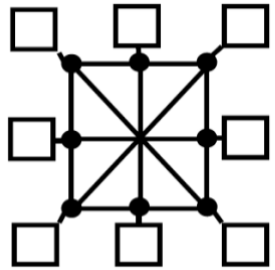
Torus



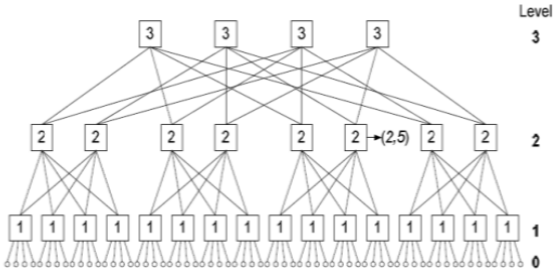
Folded torus



Ring



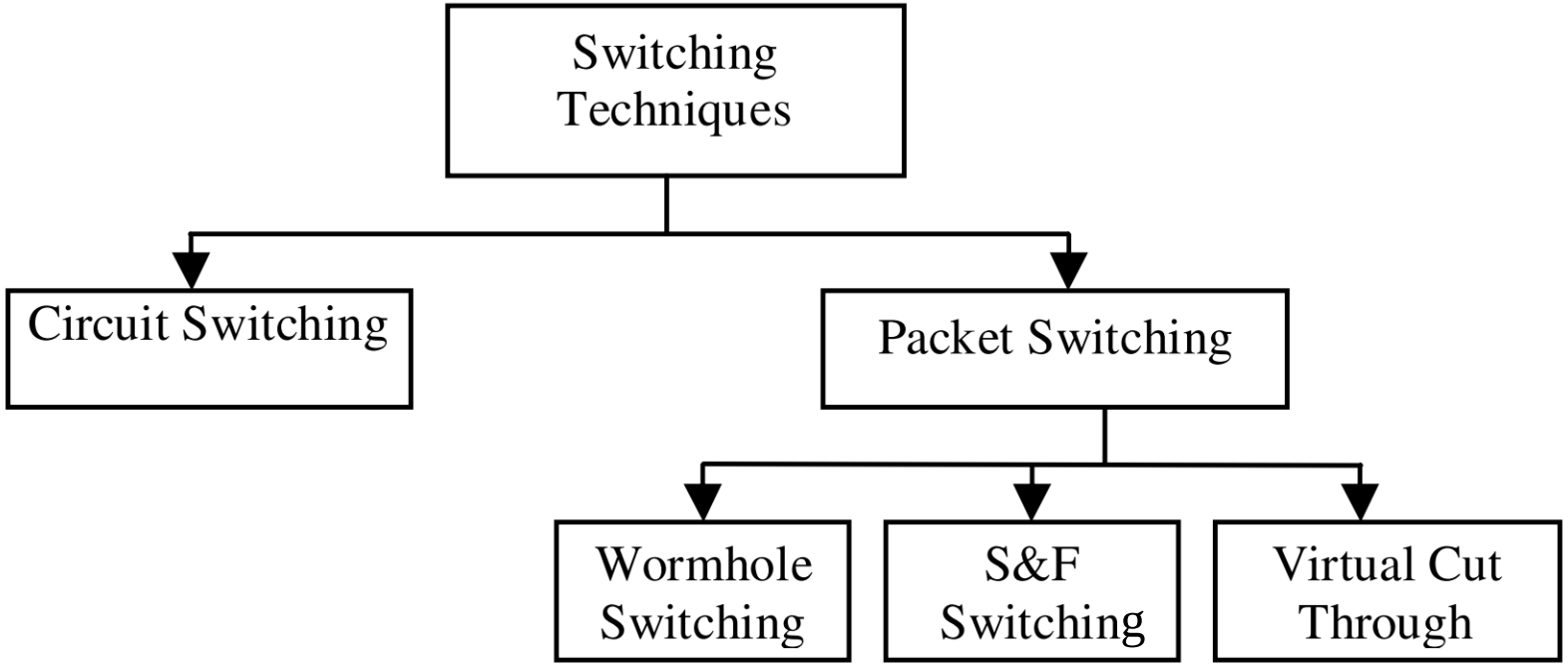
Spidergon



Fat tree

Scalable to hundreds (thousands) cores on chip

NoC: switching



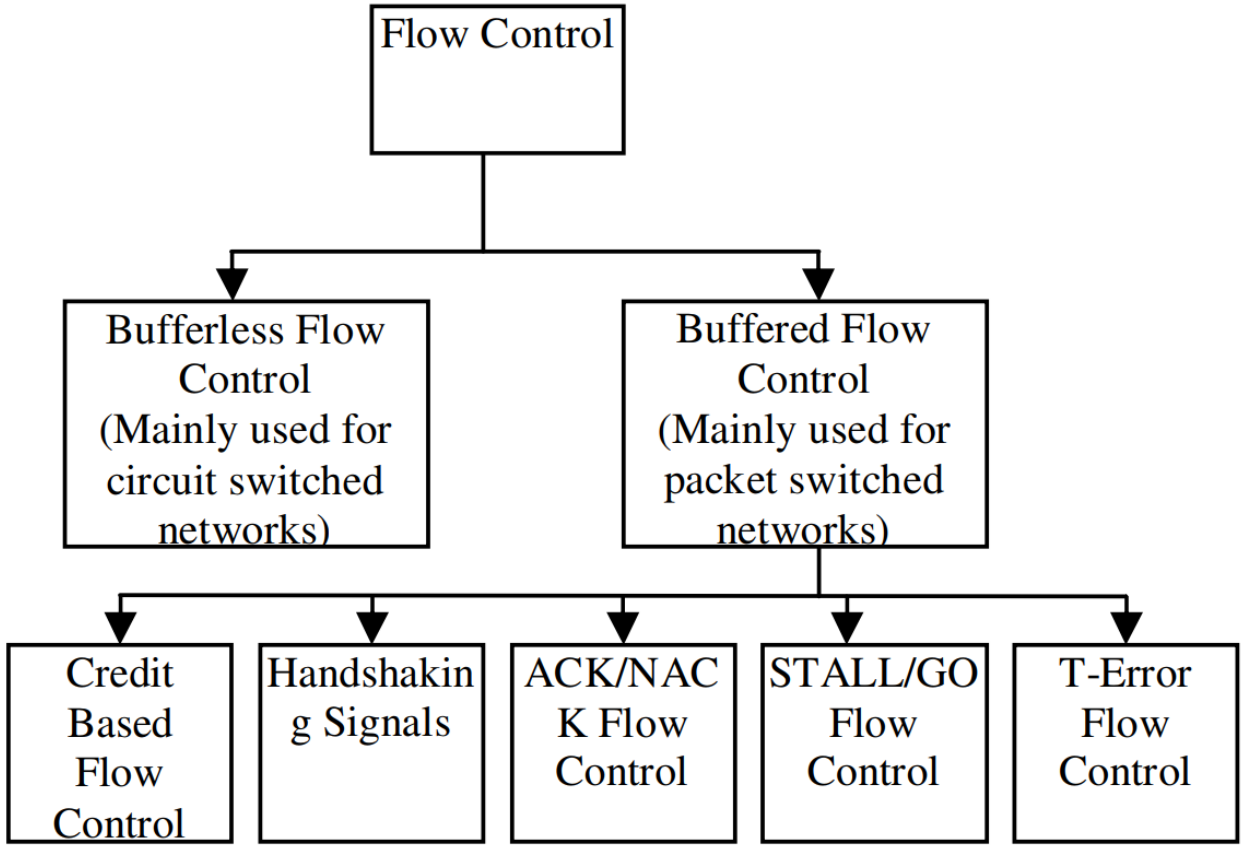
A. Agarwal, R. Shankar Survey of Network on Chip (NoC) Architectures & Contributions. Journal of Engineering, Computing and Architecture, 2009

ITMO

Summary of NoC switching schemes

Scheme	Explanation
Circuit switching	No splitting in packets done, the same physical media used for transfers for multiple endpoints.
Packet switching: wormhole	Transfers are split in packets, but no intermediate buffering is implemented in routers. Stall of head flit causes entire transfer to stall.
Packet switching: Store & Forward	Transfers are split in packets, buffer storage for packets is reserved in routers before transmission, re-transmission begins only after the entire packet is received.
Packet switching: Virtual Cut Through	Transfers are split in packets, buffer storage for packets is reserved in routers before transmission, re-transmission begins once output port is ready (maybe before the entire packet is received).

NoC: flow control



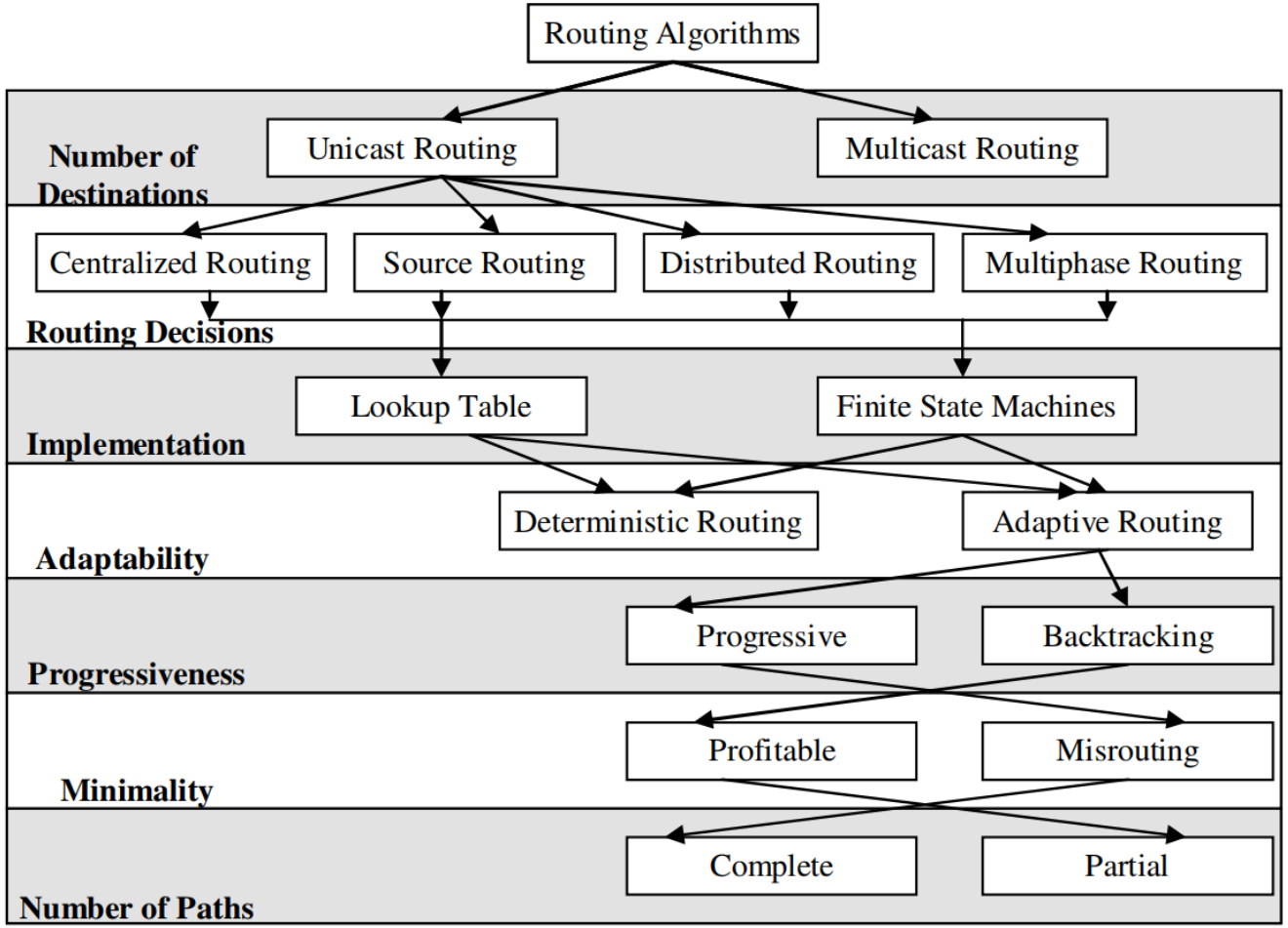
Actual not only for NoCs but for hardware microarchitectures in general

ITMO

Summary of NoC flow control schemes

Scheme	Explanation
Handshaking	Flow control is managed by two signals: valid (driven by sender) and ready (driven by receiver). Transaction completes only when both ready and valid are active.
Stall/Go	Flow control managed by receiver, receiver signals if transmission has to be paused or continued.
ACK/NACK	Copy of transmitted data is stored by sender. Once receiver confirms successful receive, sender deletes the data. If not, sender re-sends the data.
Credit based	Sender manages the counter with guaranteed amount of receivable data. Once data is sent, the counter is decreased. Once receiver finished processing received data, confirmation is sent to sender, and sender increases the counter.
T-Error	Does not guarantee reliable transmission, packets can be discarded. Reliability should be ensured by higher-level protocols.

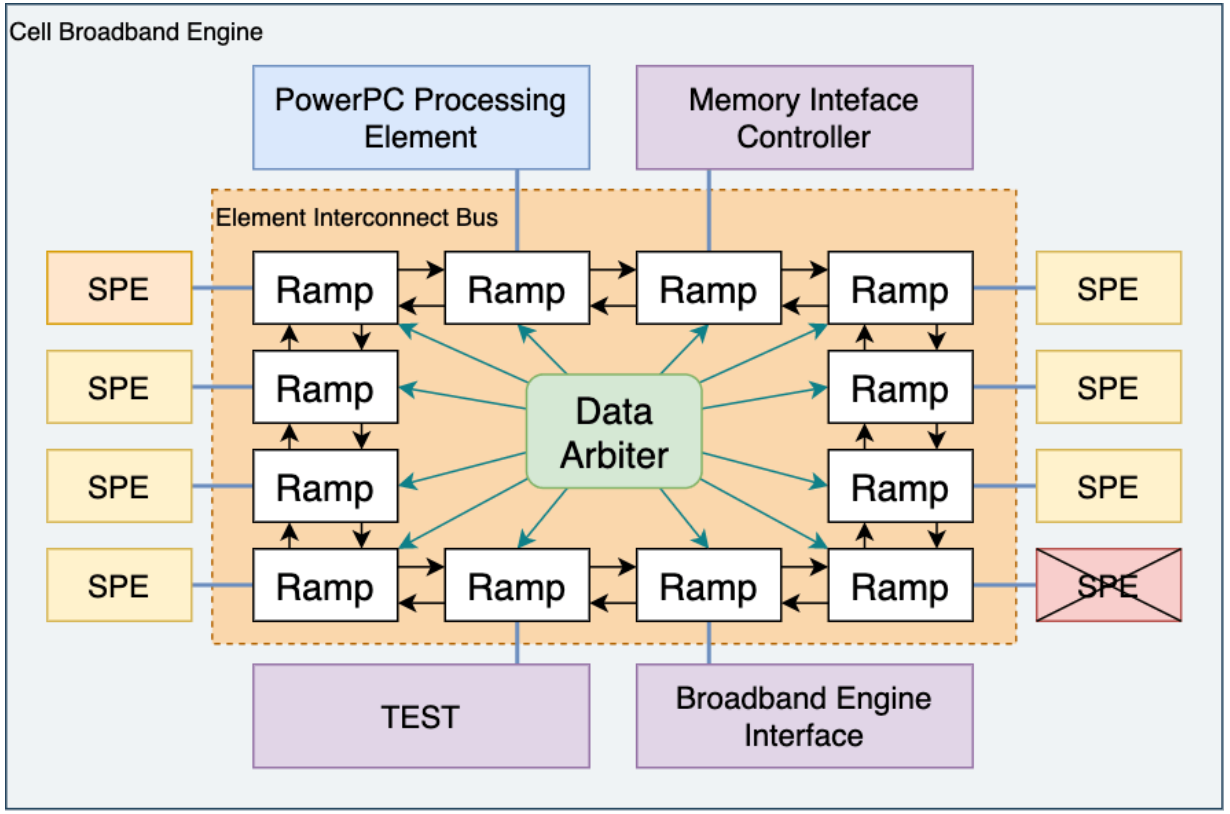
NoC: routing


Routing Algorithms
Number of Destinations
Unicast Routing
Multicast Routing
Centralized Routing
Source Routing
Distributed Routing
Multiphase Routing
Routing Decisions
Lookup Table
Finite State Machines
Implementation
Deterministic Routing
Adaptive Routing
Adaptability
Progressive
Backtracking
Progressiveness
Profitable
Misrouting
Minimality
Complete
Partial
Number of Paths

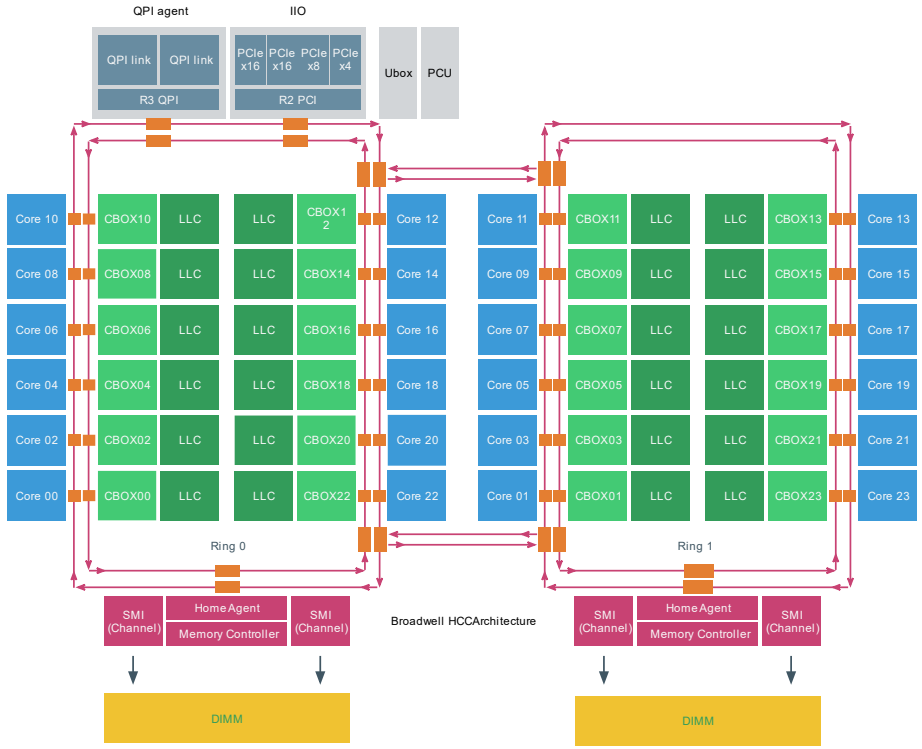
ITMO

Commercial example: NoC in Cell microprocessor

- Topology: **ring**
  good scaling but relatively high latencies
- Point-to-point connections
  *between **Ramps***
- 2x 128-bit buses
  going in opposite directions
- Credit-based flow control
  command credits reserved by
  Data Arbiter

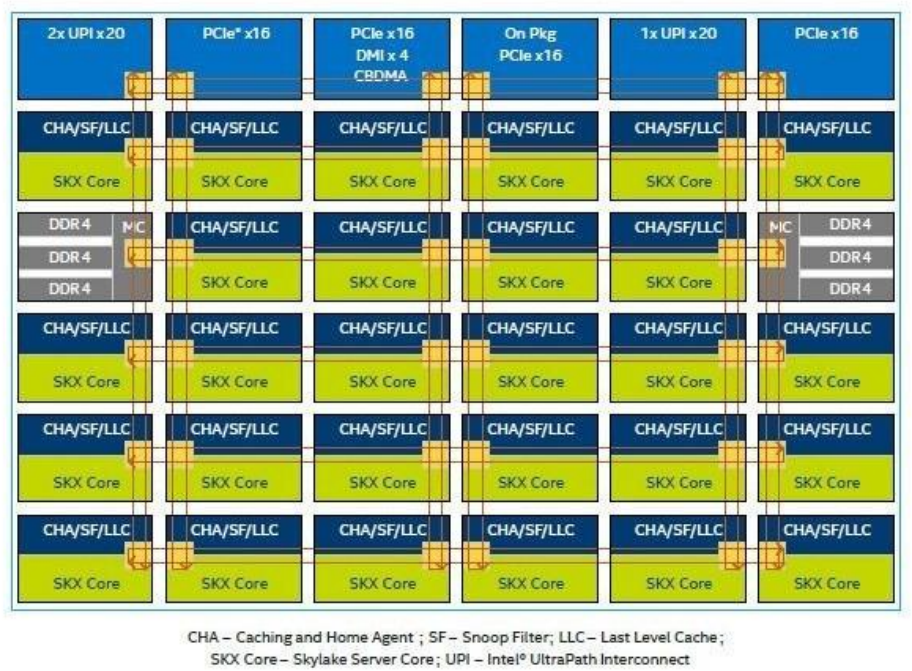


Commercial example: interconnects in modern Intel CPUs



Intel Broadwell EX (2016)

Topology: ***multiple rings***



Intel Skylake-SP (2017)

Topology: ***mesh***

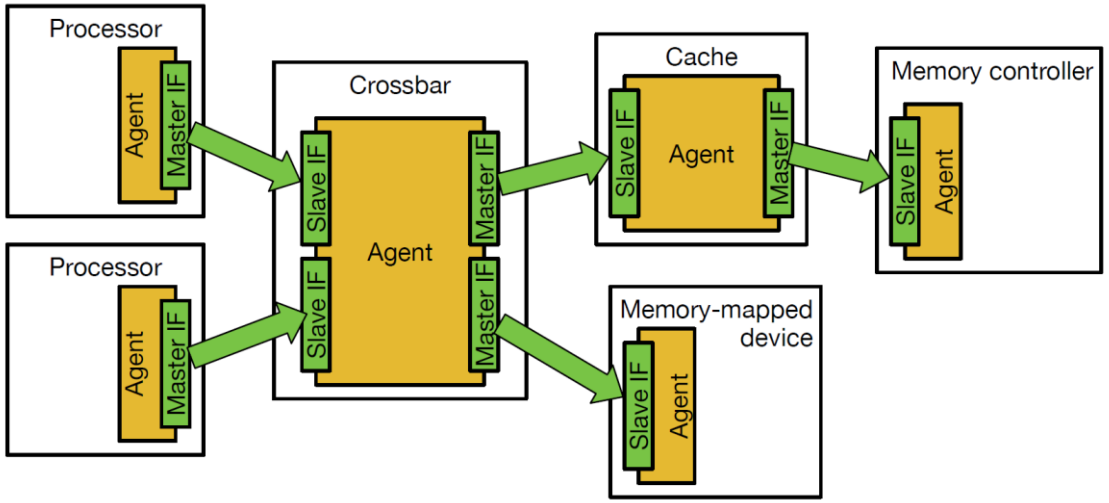
ITMO

Open-source NoCs

ITMO

Open source interconnect generator: Diplomacy (Berkeley)

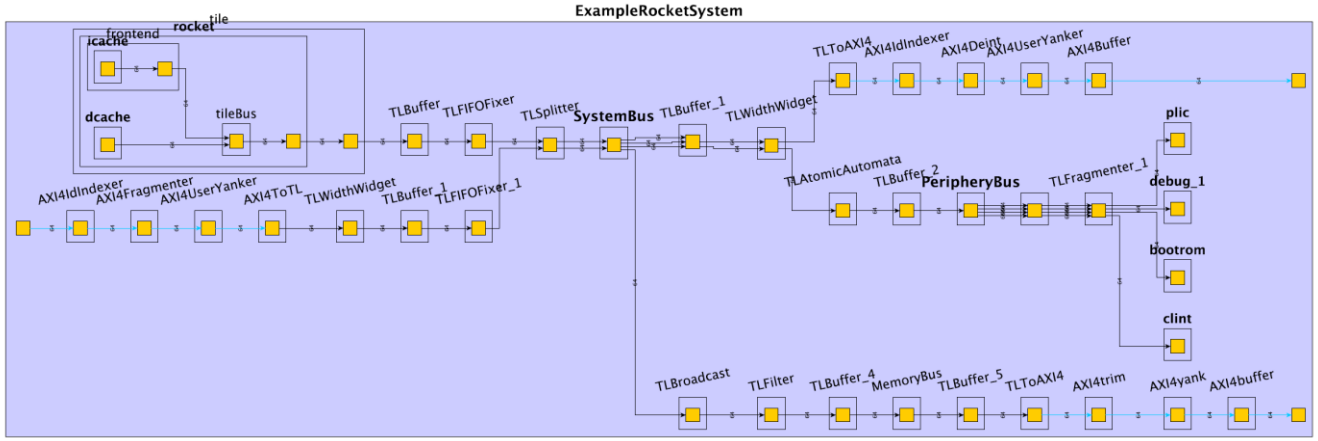
- Framework for negotiating parameters of xbar components
  provides deadlock avoidance, enables forward progress, matches operational requirements
- Based on custom TileLink protocol
- Written in Chisel HCL
- Integrated in Rocket Chip SoC generator



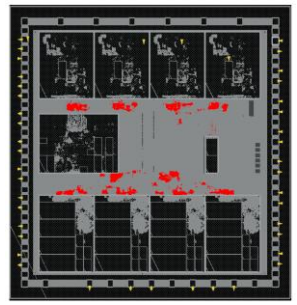
H. Cook, W. Terpstra, and Y. Lee, "Diplomatic Design Patterns: A TileLink Case Study," in 1st Workshop on Computer Architecture Research with RISC-V, 2017.

Open source interconnect generator: Diplomacy (Berkeley)

- Input data (key abstraction): directed graph of interconnected nodes
- Works in 2 phases: (1) parameter negotiation; (2) HW generation
- Manages multiple parameters of interconnect blocks
  address spaces, data widths, allowed operations, ordering requirements, cacheability, …
- Facilitates “correct-by-construction” designing firing errors early instead of long simulations

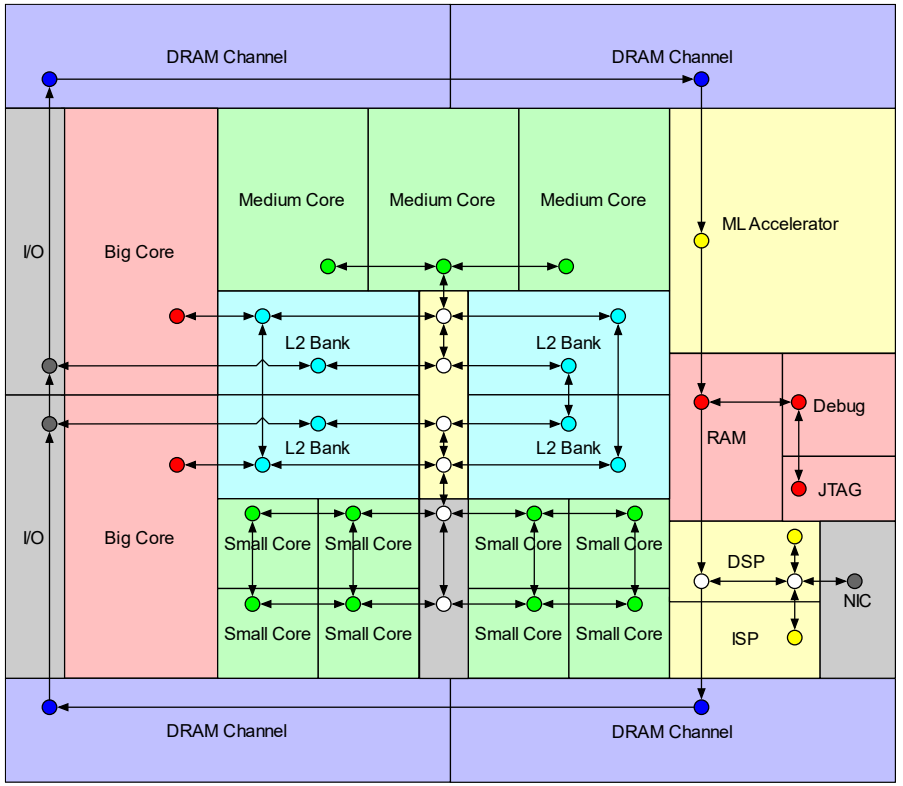


ITMO



Open source NoC generator: Constellation (Berkeley)

- Focuses on custom irregular topologies
- Implements wormhole routing with virtual channels (transaction streams non-blocking each other)
- Written in Chisel HCL
- Integrated into Chipyard SoC design framework



J. Zhao et al., "Constellation: An Open-Source SoC-Capable NoC Generator," in NoCArc, 2022.

ITMO

Wrapping everything up…

ITMO

Wrapping everything up…

- Goal of the course: learn basic and advanced construction principles of optimized computer systems, including HW, SW, and their interplay
- Course structure:
  - **Section 1. Structure and types of computer architecture**
    - **Lecture 1. Basic terms. Von Neumann architecture**
    - **Lecture 2. Non-vN architectures. DSA and SoCs. SW vs. HW design**
    - **Lecture 3. Abstraction levels of programmable computer systems. Hardware and software implementations**
    - **Lecture 4. Instruction set architectures.**
  - **Section 2. Processor architecture. Low-level software development**
    - **Lecture 5. Software toolchain structure**
    - **Lecture 6. Basics of C programming**
    - **Lecture 7. Basics of ASM programming (RISC-V ISA)**
    - **Lecture 8. Operating systems. Memory virtualization**
  - **Section 3. Hardware microarchitecture: optimization of control and data flows**
    - **Lecture 9. Basic microarchitectural templates. Multi-cycle and pipelined hardware**
    - **Lecture 10. Data flow optimization**
    - **Lecture 11. Control flow optimization**
    - **Lecture 12. Dynamic branch prediction. Open-source superscalar OoO CPU cores**
  - **Section 4. Hardware microarchitecture: optimization of communications and memory**
    - **Lecture 13. Basic memory optimizations**
    - **Lecture 14. Caching**
    - **Lecture 15. Basic communications optimization. Shared memory integration. Cache coherence**
    - **Lecture 16. Bus architectures. Networks-on-chip. Open-source NoCs**

Thank you for the lesson!

Alexander Antonov, Assoc. Prof., antonov@itmo.ru

Hangzhou, 2025