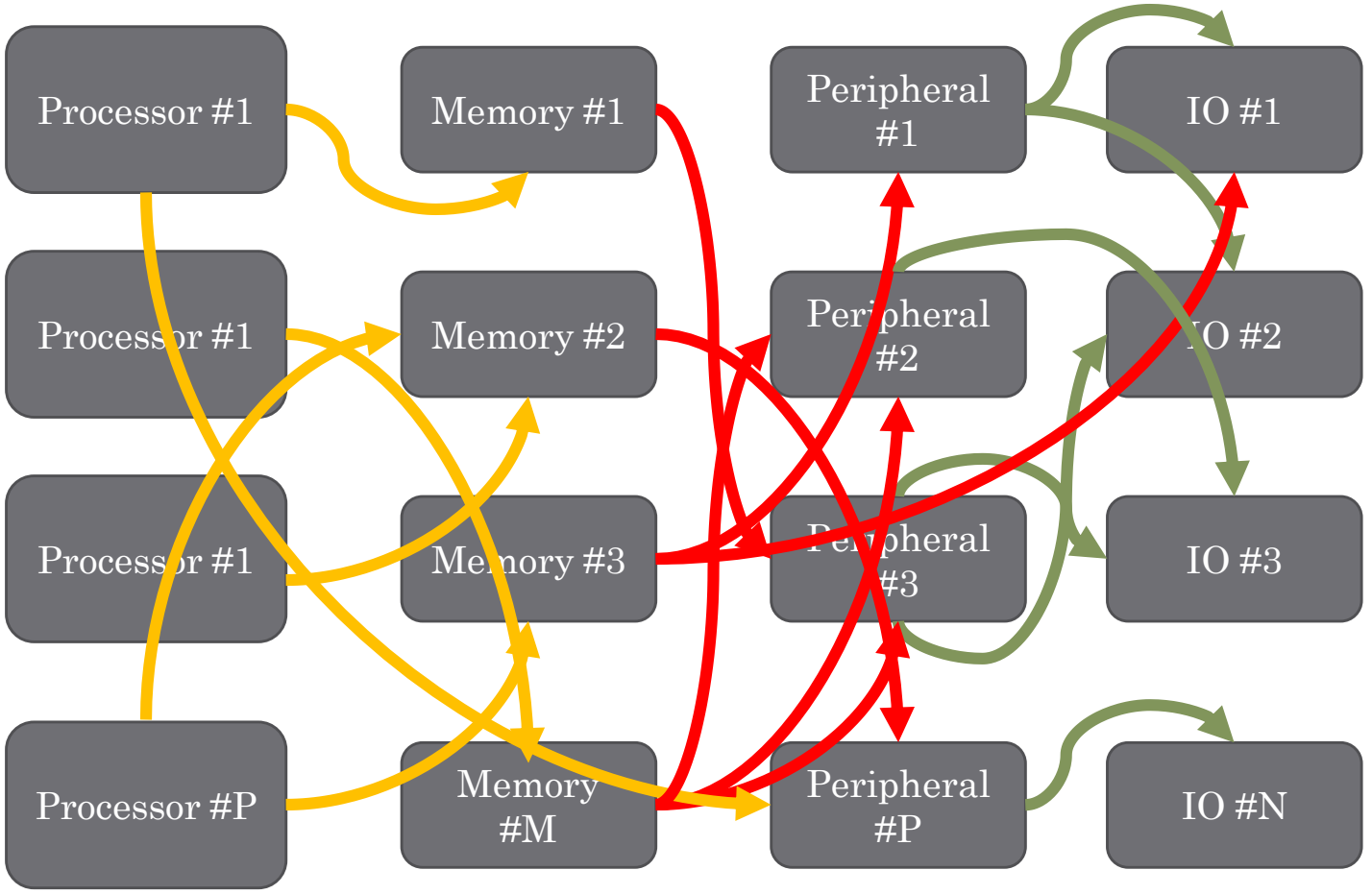
1.2: Computer Interconnects

What are we learning today?

- What are interconnects?
- Impact of interconnects on computing systems
- Types and Topologies of Interconnects
  - Shared bus
  - Network on Chip
  - Point to point
- Interconnect protocols

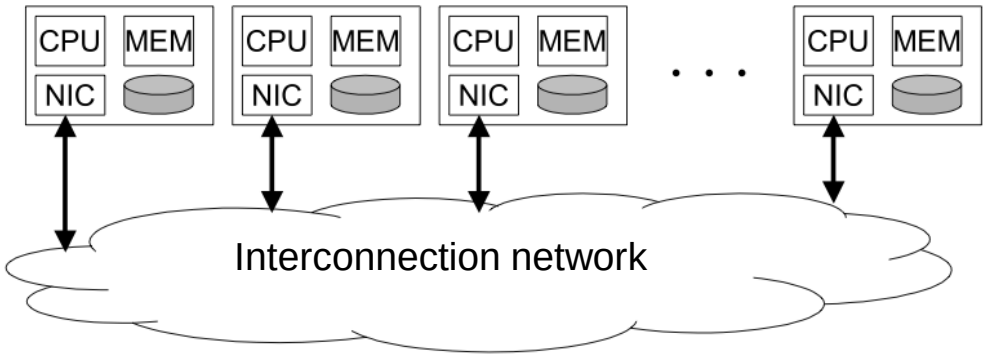
--- Let's play mash up ---



Computer Interconnect

- Interconnect is a critical component in modern computer systems as it defines the underlying performance of the systems

- Interconnects are used to connect various components in the system
  - Processor to processor
  - Processors to memories (banks)
  - Processors to caches (banks)
  - Caches to caches
  - I/O devices



The Importance of Interconnects

- Affects performance and energy efficiency
  - How fast can processors, caches, and memory communicate?
  - How long are the latencies to memory?
  - How much energy is spent on communication?

- Affects the scalability of the system
  - How parallel/large of a system can you build?
  - How easily can you add more processors?
  - How easily can more memory/IOs be integrated in the system

Managing Interconnects

* Driving a interconnect is hard work for devices
  - Lots of buffers and wiring

* Every system has an interconnect topology and protocol to manage how interconnects are
  - **accessed** (IO pin counts, and handshakes)
  - **shared** (i.e. interconnect arbitration)
  - **power managed**: for example IO transceivers disconnect interconnects from the main interconnect when idle

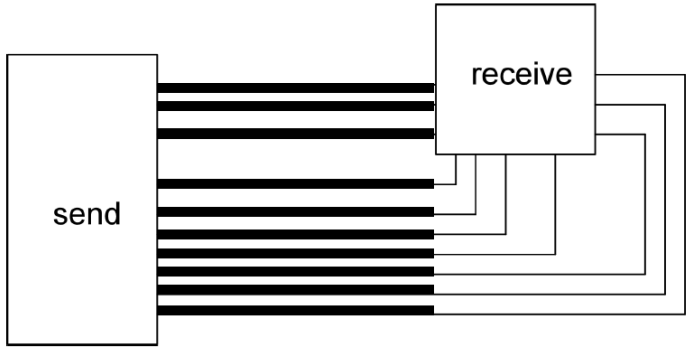
* Different vendors like different bus architectures and protocols

  ARM: AMBA (AHB, AXI), Freescale(NXP): VME, Intel: QuickPath, Open Source: Wishbone, Open Core Protocol

Interconnect Skew

All interconnects suffer from **interconnect or bus skew**: difference in propagation delays on the wires of the bus, because of differences in wire lengths.

Consider the following inter



Uneven wire lengths between send and receive nodes

Bound to cause skew between them

Designer need to design protocols addressing worst case skew times

Interconnect Types

There are two fundamentally different bus protocols:

Synchronous :

- Includes a clock in the control lines
- A fixed protocol for communication relative to the clock
- Advantage: involves very little logic and can run fast
- Disadvantages: every device on the bus must run at the same clock rate to avoid clock skew, it cannot be very long.

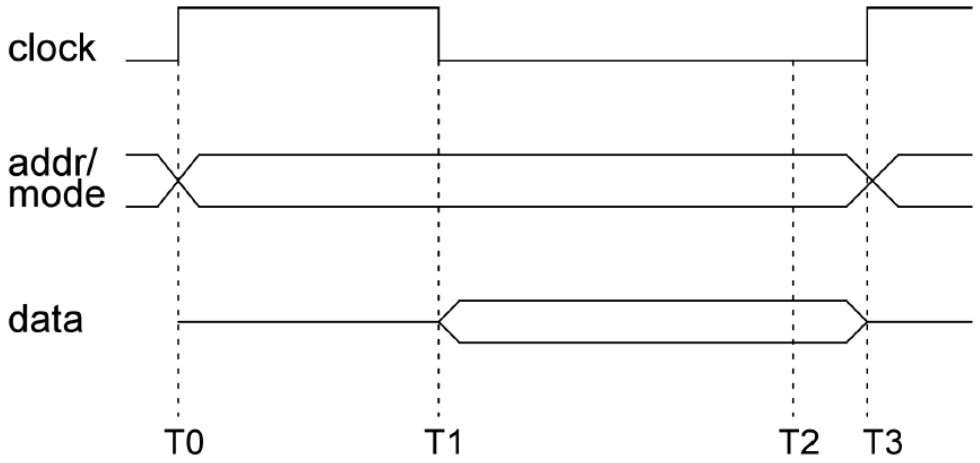
Asynchronous :

- It is not clocked
- It can accommodate a wide range of devices
- It can be lengthened without worrying about clock skew
- It requires a (rather complicated) handshaking protocol with overheads.

Synchronous interconnects

Have a clock line, which drives the interconnect transactions based on cycles.
Interconnect cycles = several CPU cycles (typically 4 to 10).

Example timing diagram: CPU reads data from device.



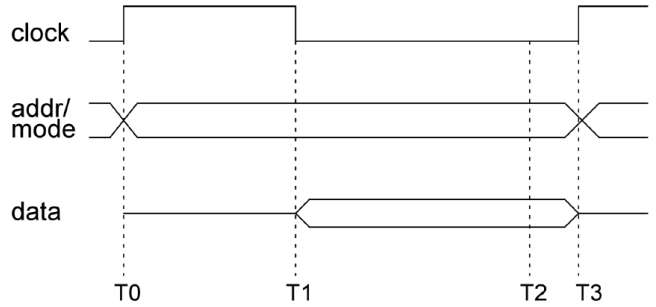
- Slanted lines mean "possible change".

Synchronous interconnects – *contd.*

At *T0*, the CPU sets the address lines, and sets the mode bits to "read".

The clock pulse width (T1-T0) must be long enough to:

- accommodate propagation delay
- allow the receiver to read the address



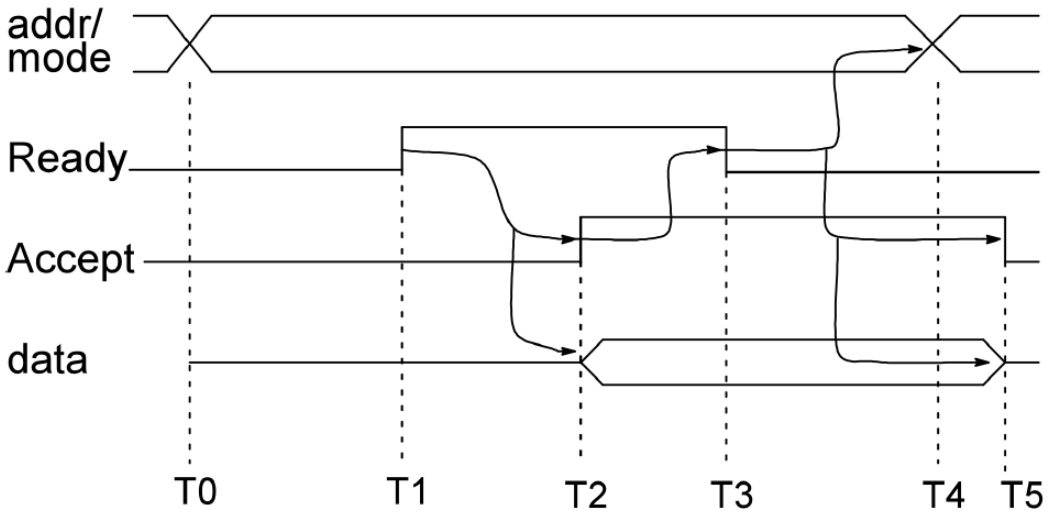
The receiver is ready to respond at T1.
It sets the data lines.

The data is read into the interconnect. At T2, the CPU reads the data. All lines are cleared for a new bus cycle at T3.

Asynchronous interconnects

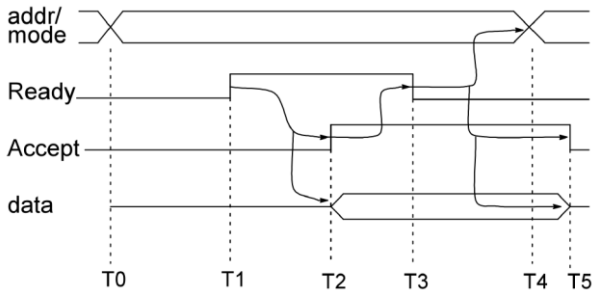
Do not have a global/master clock. Bus cycles can have any length, depending upon the master/slave.

Transfer is achieved by means of **handshaking**.
Input (i.e Read) operation:



- The arrows mean "causes".

Asynchronous interconnects – *contd.*



1. (T5 - T0) is one interconnect cycle. At T0, the CPU sets the address and mode lines.

2. The CPU allows until T1 for bus skew. It then sets the Ready signal. The addressed device receives the Ready signal shortly before T2. It sets the data lines, and raises the Accept line to **acknowledge** receipt.

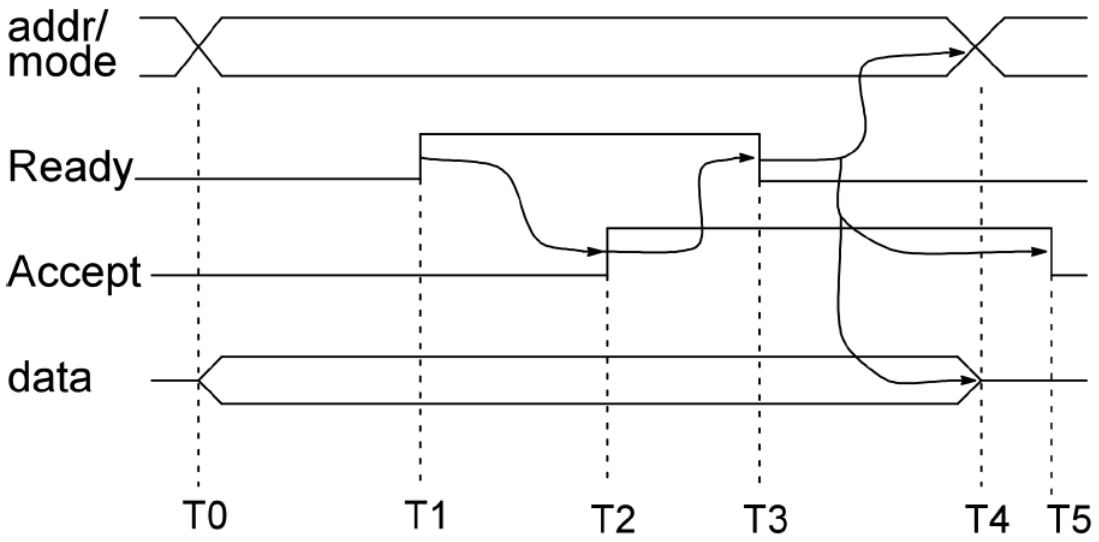
3. When the CPU sees the Accept, it allows for skew. At T3, it lowers Ready, and reads the data. After a bus skew delay, the CPU removes the address at T4.

4. At T5, the addressed device sees Ready going down. It lowers Accept and removes the data.

5. The bus cycle is then finished.

Asynchronous interconnects – *contd.*

- Output (Write) operation from CPU to device:



- Explanation: The CPU provides the data, so this is made available immediately.

  The CPU removes the data at T4.

Interconnect Topology

- Topology: Specifies the way interconnects are wired
  - Affects performance, connectivity and scalability

- Different topologies:
  - **Shared bus:**
    - Simplest, State-of-the-art
    - Simpler arbitration protocols
    - Scalability issues
  - **Point-to-point**
    - ideal but highly costly
    - lots of wires!
    - Poor scalability
  - **Cross bar or Network-on-Chip**
    - emerging modular architectures (good scalability!)
    - Packet switching based, requires complicated network interface and arbitration protocols
  - **Irregular interconnects**
    - Highly customised for a given application
    - not good for scalability

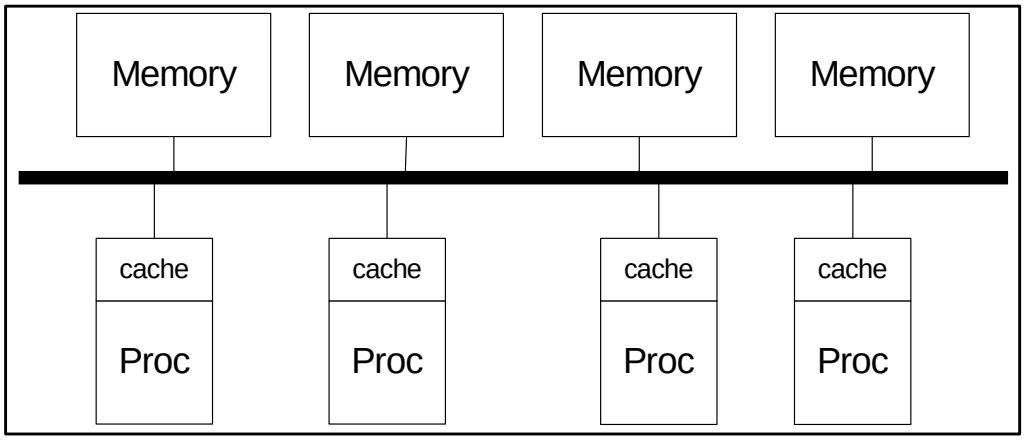
Shared Bus

+ Simple and Cost effective for a small number of nodes

+ Easy to implement coherence (snooping and serialization)

- Not scalable to large number of nodes
  - limited bandwidth
  - electrical loading
  - reduced frequency

- High contention → fast saturation



Shared Bus Transactions

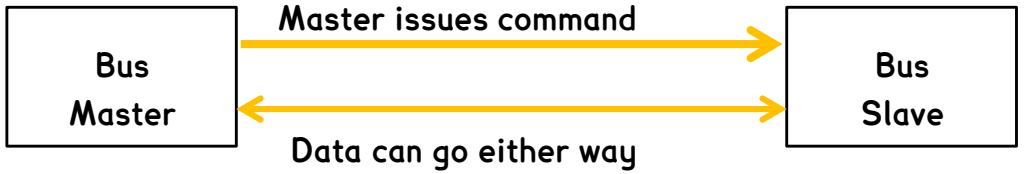
A **bus transaction** (or **bus cycle**) includes two parts:

- Issuing the command and address and transferring the data

The **master** starts the bus transaction through command & address

The **slave** is the one who responds to the address by:

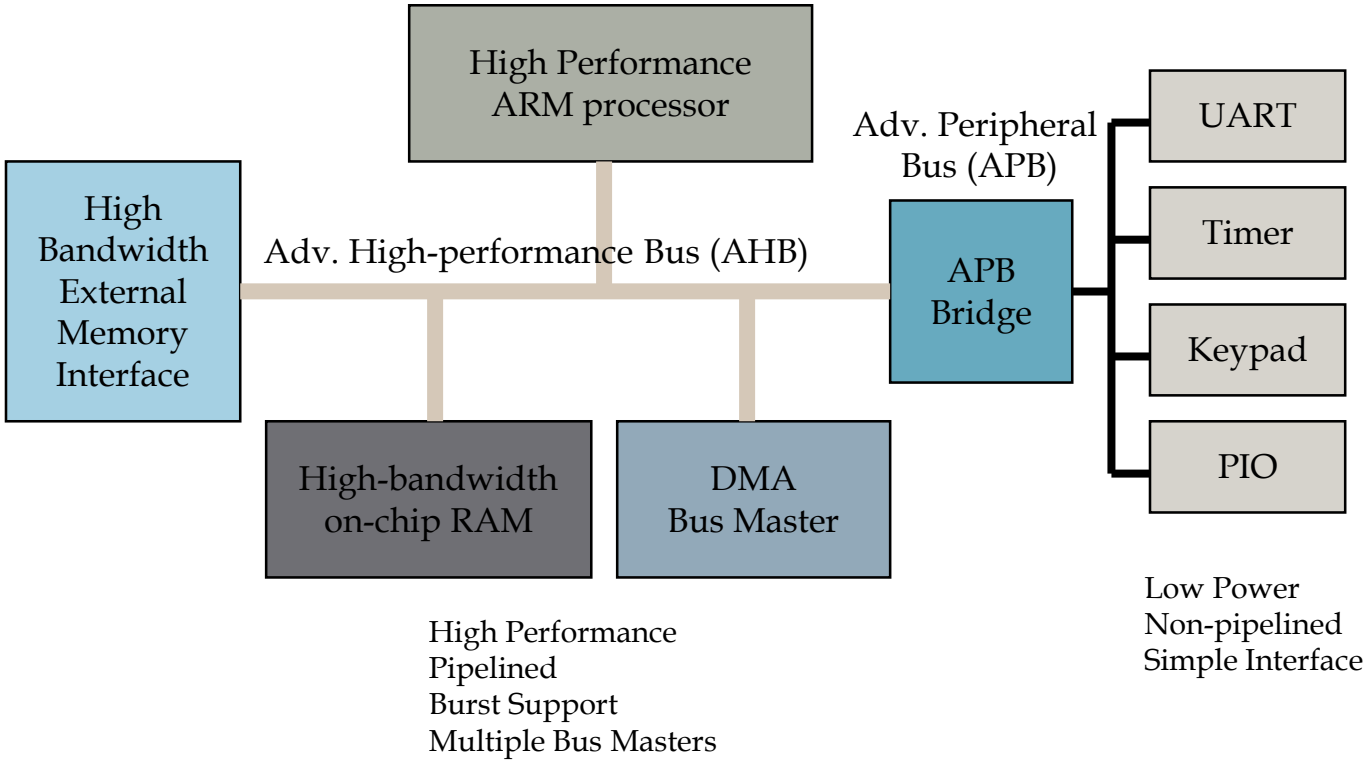
- Sending data to the master upon request
- Receiving data from the master



master	slave	description
CPU	memory	Fetching code or data
CPU	I/O	Data transfer
CPU	Co-processor	Graphics operations
I/O	memory	Data transfer
Co-processor	memory	Data transfer

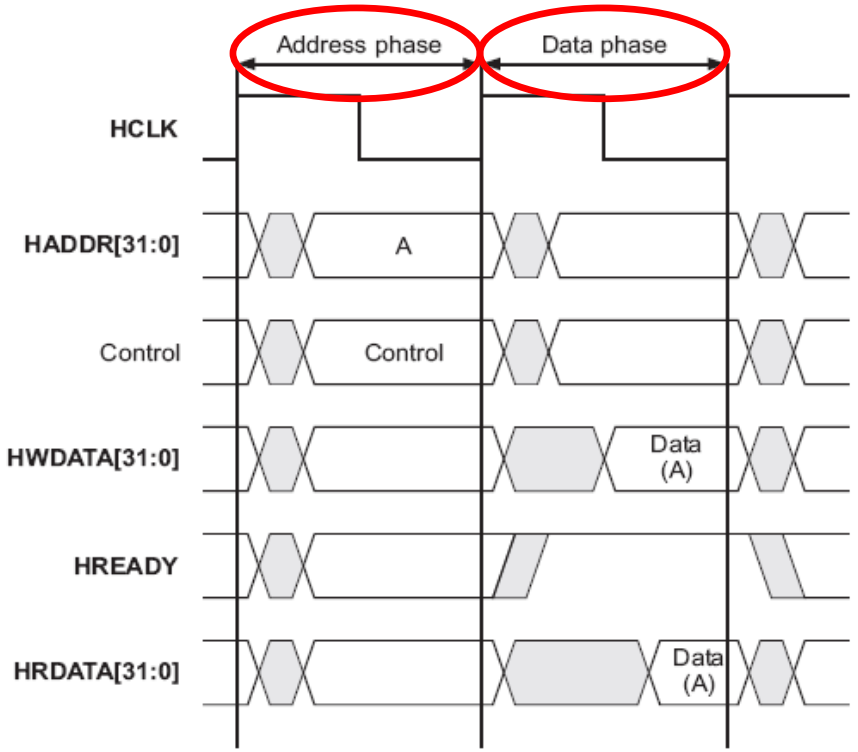
Shared Bus Example: ARM AMBA

AMBA: Advanced Microprocessor Bus Architecture
- Family of ARM's interconnect systems



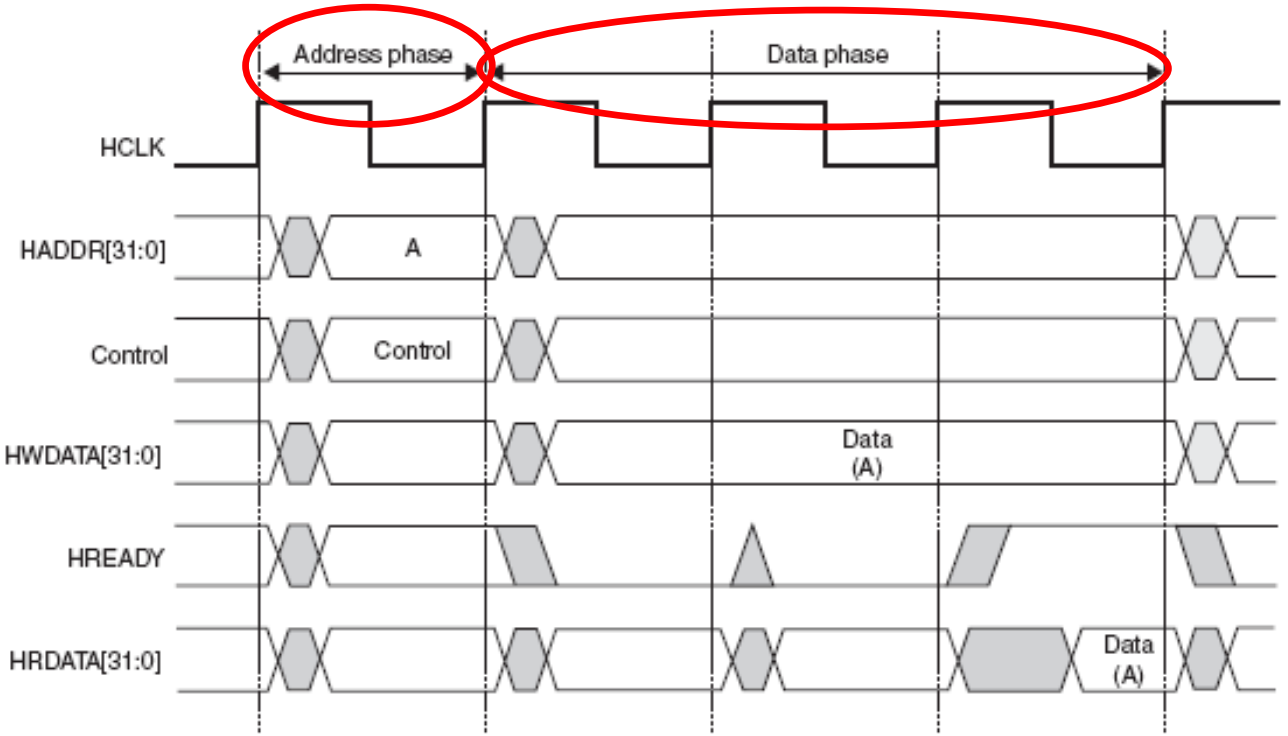
AHB Basic Transfer

- Split ownership of Address and Data bus
- One cycle for address, and the other for data (read/write)



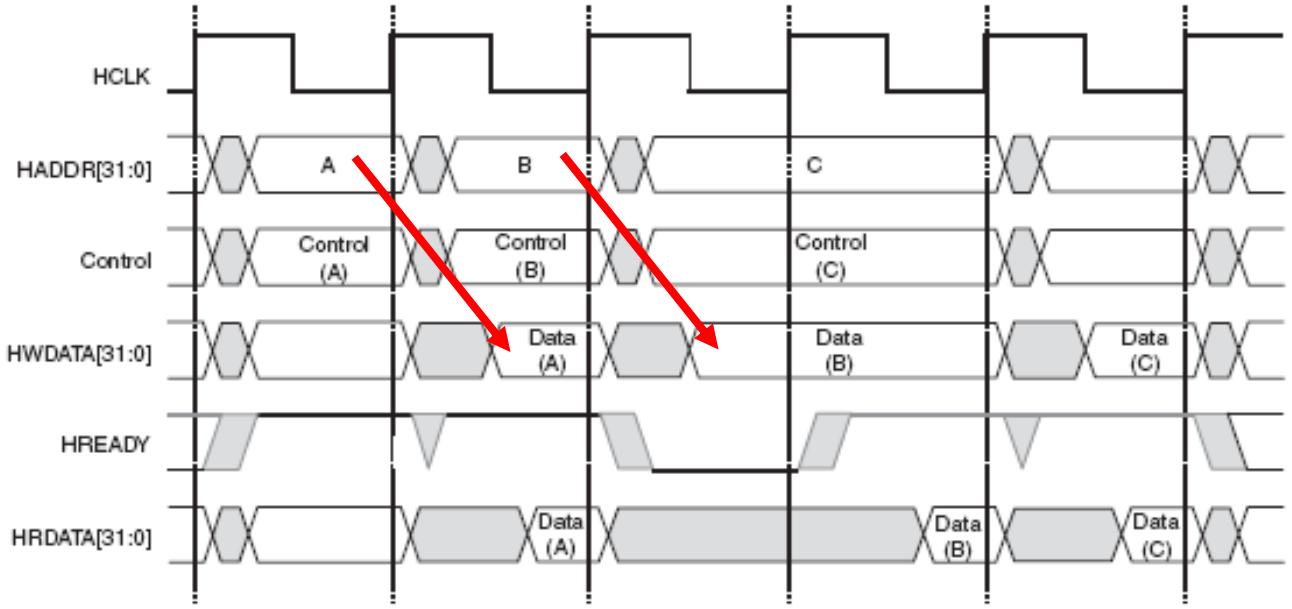
AHB Basic Transfer

- Data transfer with slave wait states
- Longer data phase wait states can slow down pipelined transactions
- Ideally, if every phase is completed in one clock cycle, one transaction per cycle is possible.

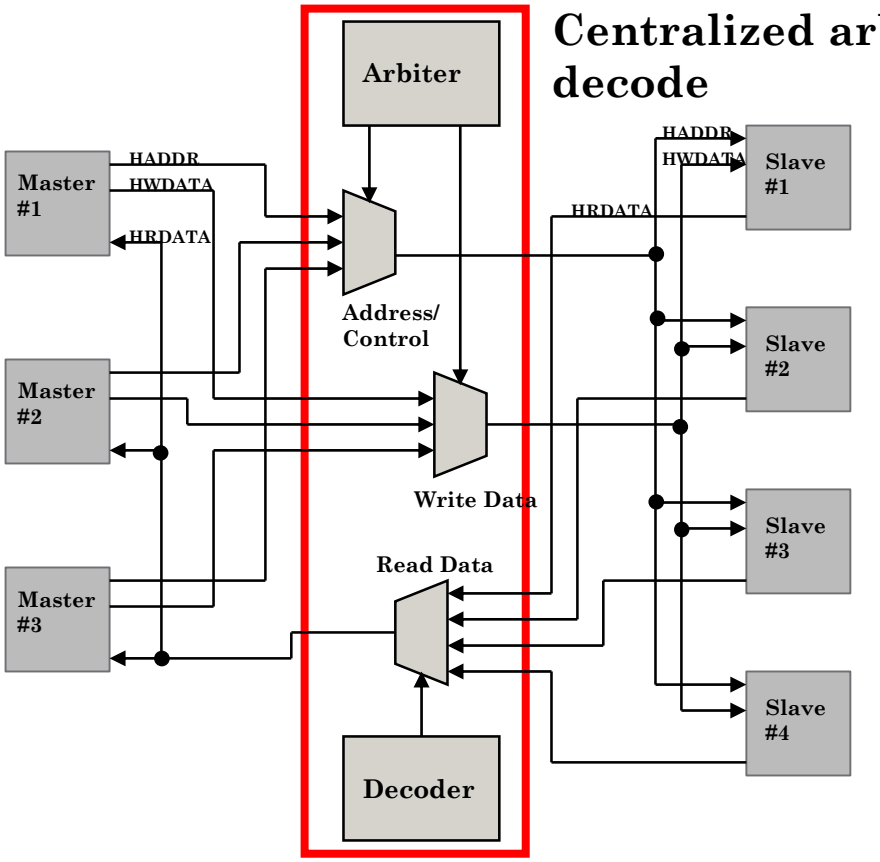


AHB Pipelining

- Transaction pipelining increases bus bandwidth
- This is a two-stage pipelined transaction on the bus: address/command and data phase
- Can you think of any other phase in a more complex interconnect system?



AHB Architecture

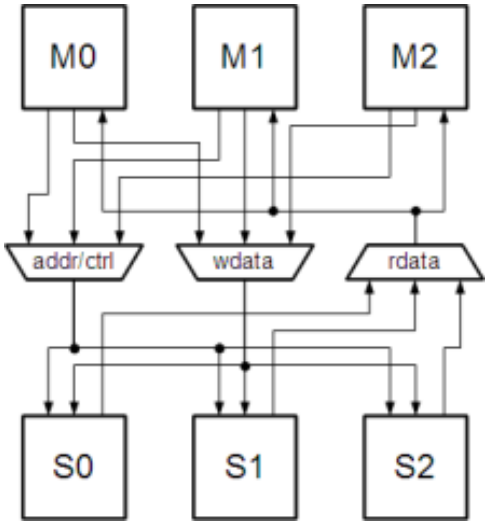


- Arbiter listens to all address/control lines from masters
- Arbiter ensures enable signal to only one write mux channel
- 1 unidirectional address bus (HADDR)
- 2 unidirectional data buses (HWDATA, HRDATA)
- At any time only 1 active data bus
- Decoder decodes the HRDATA and broadcasts to masters via Read Data mux; only one master acknowledges/receives.

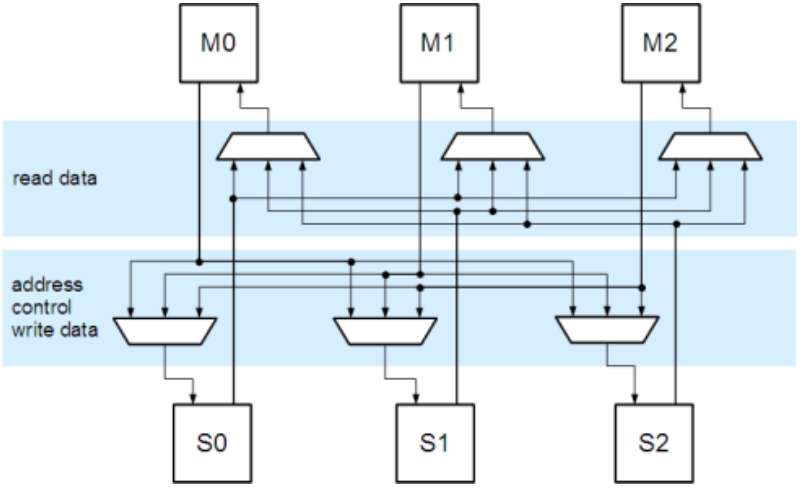
- **What happens when all masters request for write at once?**

Shared Bus: Multi-core Dilemma

- Achieving parallelism using a shared global bus is hard with multi-core (i.e. multiple masters)



Multiplexed masters



Multi-layered masters

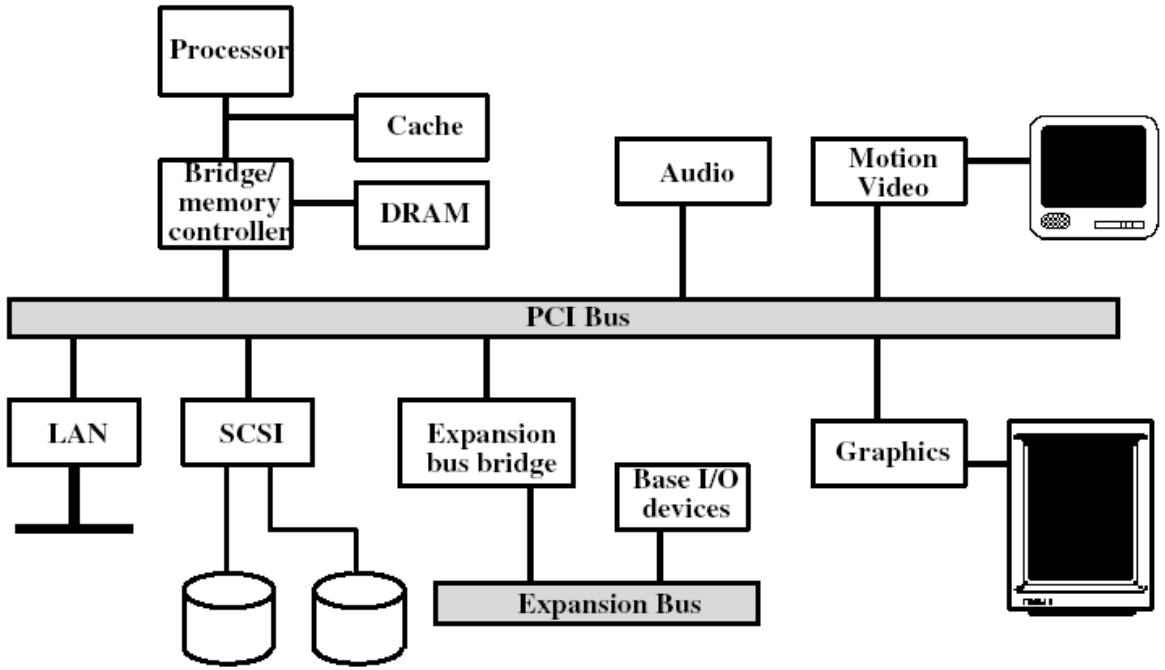
New AXI4 AMBA standards feature
- multi-layering, simpler handshaking and burst trans.
- Higher performance
- Almost like point to point, but with organised interconnect architecture

- **Question: what is the overhead in AXI compared to AHB?**

Shared Bus Example: Intel PCI

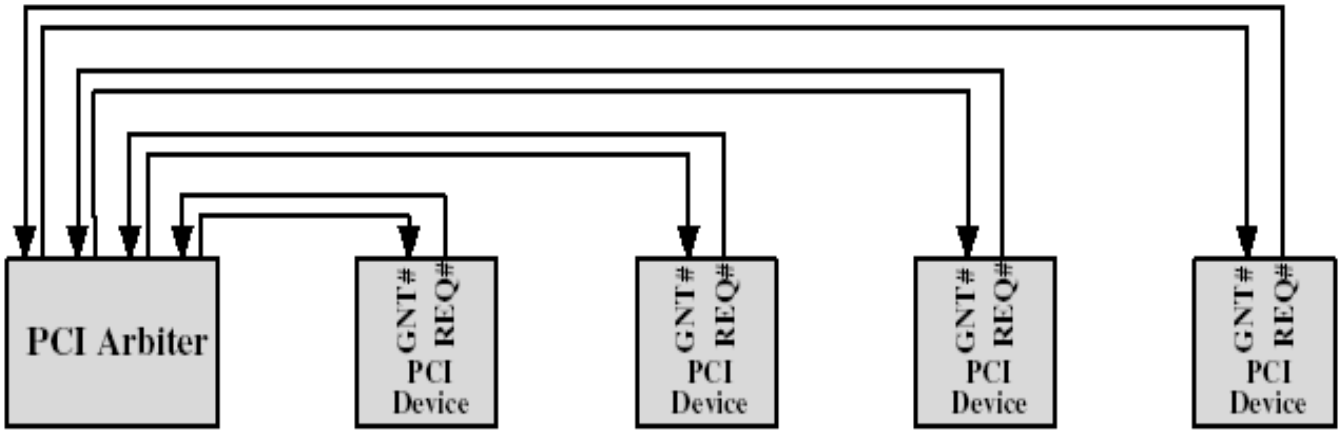
PCI: Peripheral Component Interconnection

- High-speed shared bus
- Intel released PCI in public domain in 1990s
- Bit/speed width depends on the Peripherals (eg: 32, or 64 bits at 66MHz)
- Each port has it's own adaptor to ensure the common bus can be used
- Often separate PCIs are used for input and output to increase IO bandwidth



PCI Arbitration

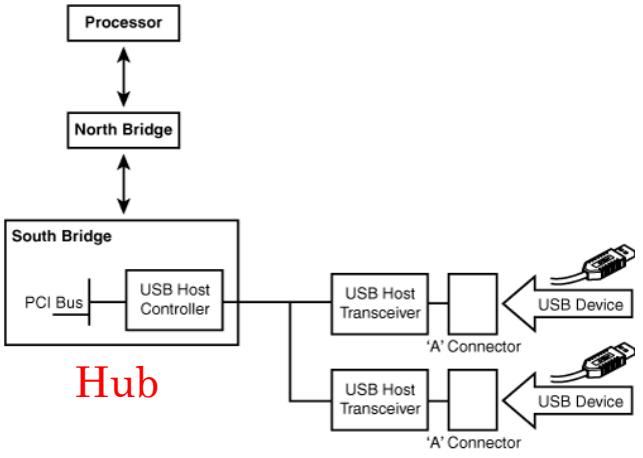
- PCI uses a centralized, synchronous arbitration scheme
  - Each device has its own unique GNT (grant) & REQ (request) lines
  - Simple request-grant handshake is used to grant bus access
  - Question: if PCI can only allow one channel, what will be the bandwidth of the following system when all PCI devices are requesting PCI access at the same time?



More examples

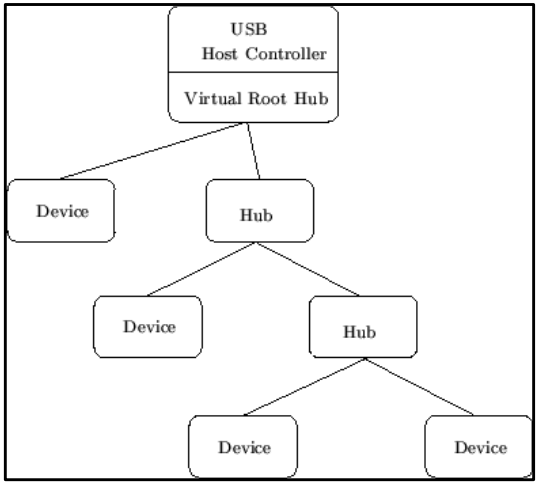
USB: Universal Serial Bus

- Shared bus originally meant for low-speed I/O Devices
- Expandable
  - Through other busses
  - USB allows up to 127 devices (0 for root, 1-127 for others)
  - Simple configuration
  - Simple design
    - Single cable design, also supplying power
    - Support for real-time devices and device classes
    - Simple to manufacture
- Max speed=usb2: 480Mbit/s usb3: 4.8Gbit/s



USB - hardware

- Has a root hub connected to the main bus
  - Further hubs can be connected to this hub, forming a tree topology
- Cable contains four wires
  - 2 data lines
  - 1 power (+5 volts) & 1 ground
- Data transmitted as:
  - "0" is transmitted as a voltage transition
  - "1" as the absence of a transition
  - sequence of "0"s forms a regular pulse stream



- USB root hub multiplexes data from devices and transmits data through regular polling
  - For example USB keyboards poll every 50ms to transmit any input data
  - Research: Find out about the modern mouse devices

Point-to-Point

Every node connected to every other

+ Lowest contention

+ Potentially lowest latency

+ Ideal for small systems, if cost is not an issue

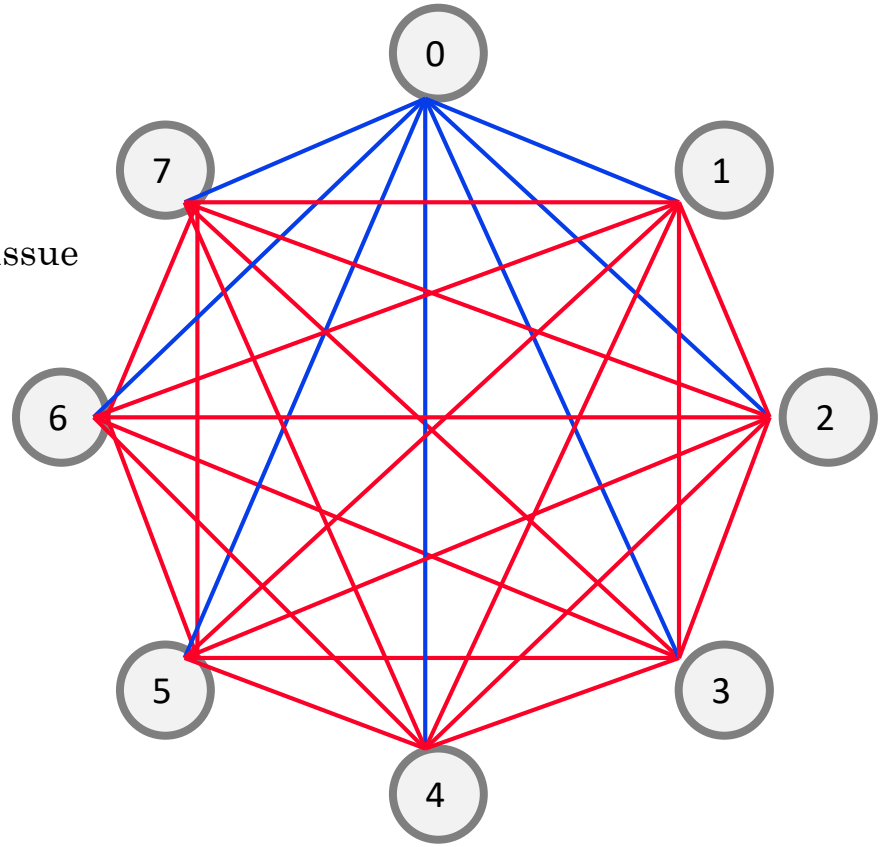
-- Highest cost for large systems

O(N) connections/ports per node

$O(N^2)$ links

-- Not scalable

-- How to lay out on chip?



Example: Nvidia's larger GPU systems have P2P connections between them

Crossbar

- Every node connected to every other (non-blocking) except one can be using the connection at any given time
- Enables concurrent sends to non-conflicting destinations
- Good for small number of nodes

+ Low latency and high throughput

- Expensive

- Not scalable → $O(N^2)$ cost

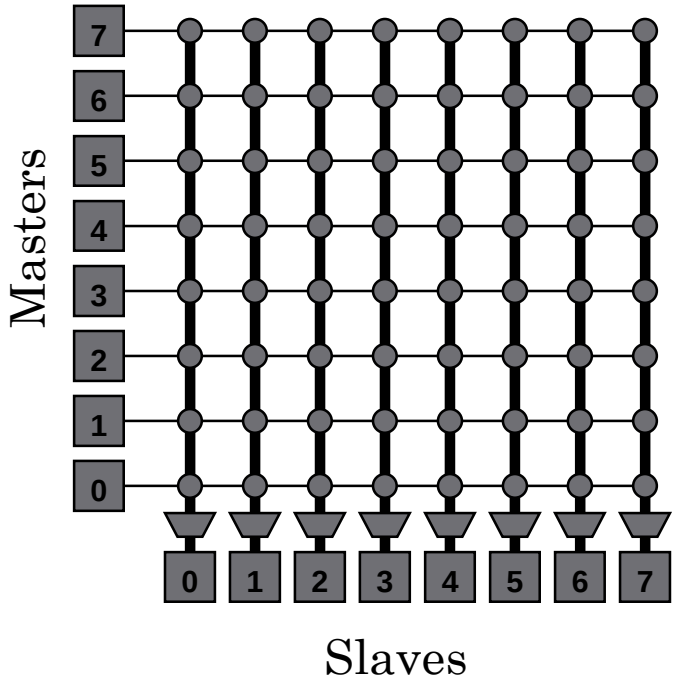
- Difficult to arbitrate as N increases

Used in core-to-cache-bank

networks in

- IBM POWER5

- Sun Niagara I/II



Network-on-Chip

- Tile-based scalable architecture:
  - Each intellectual property core is connected via a network interface, router (+switch)

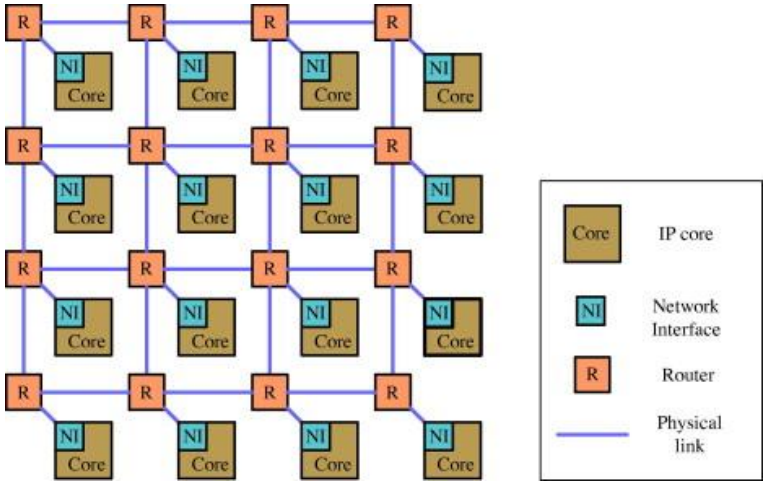
+ Enables concurrent sends to non-conflicting destinations

+ Low latency and high throughput

- Expensive, lots of wiring

- Complicated routing/arbitration

+ Scalable, better electrical properties

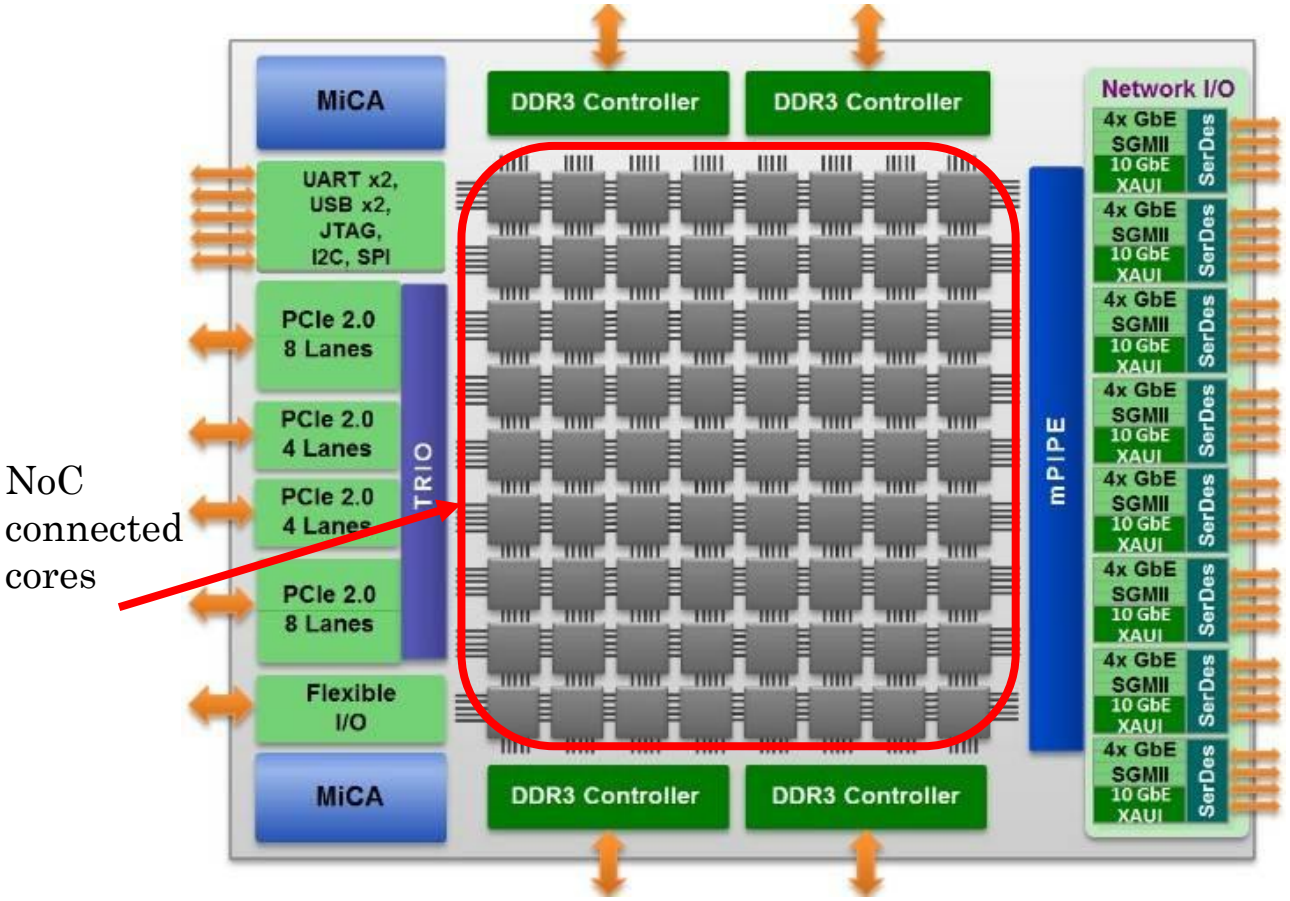


Emerging systems. Examples

- Tilera Co-processors (8x8 NoC)

- Recore Xentium DSP architectures

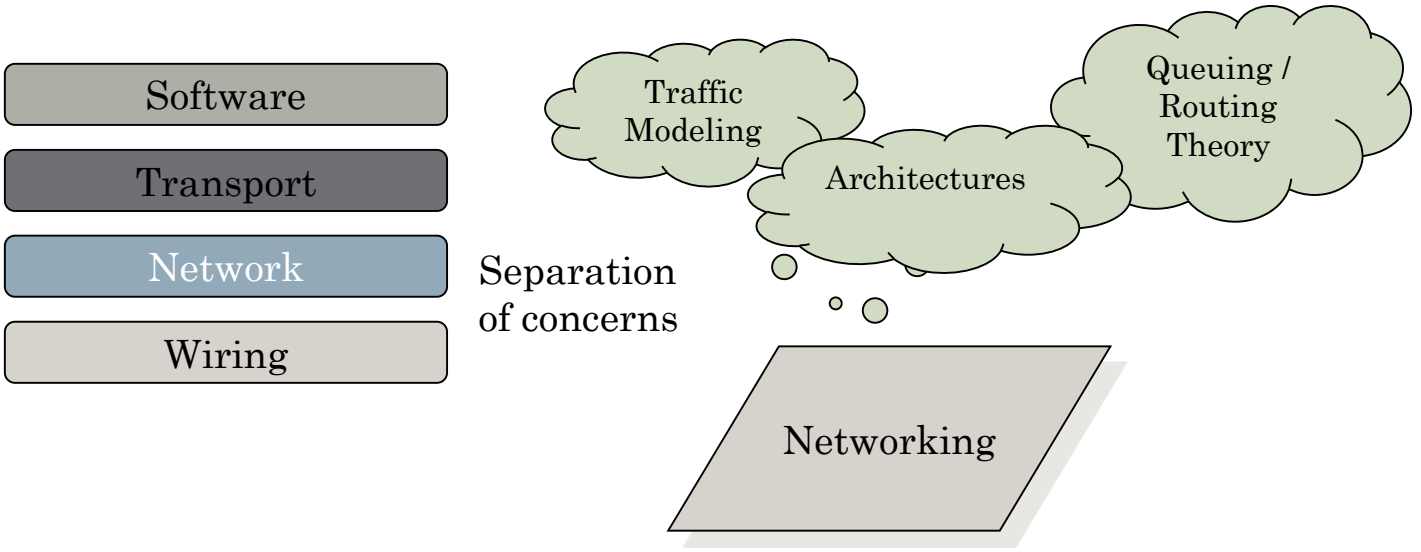
Network-on-Chip Example: Tilera



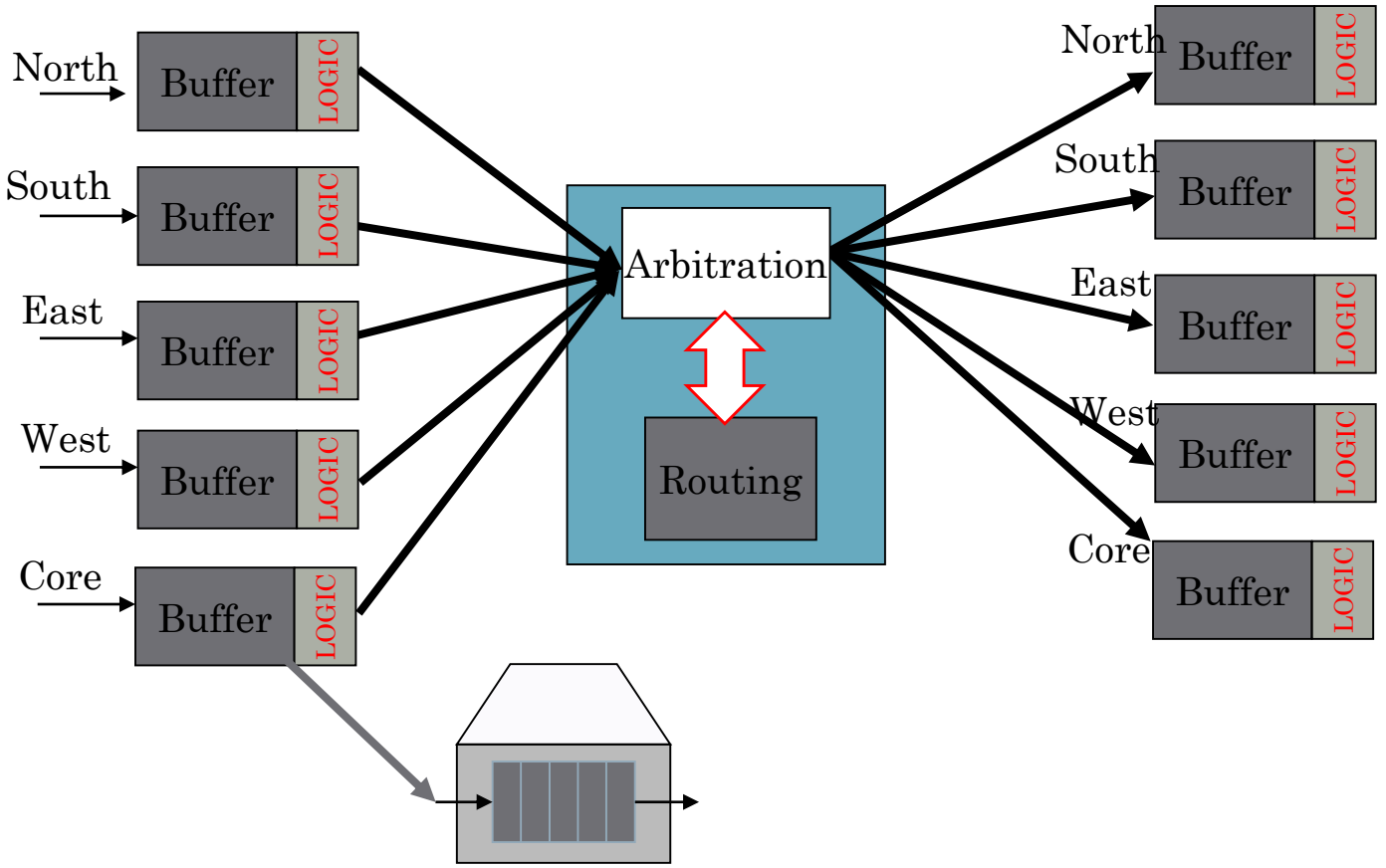
Network-on-Chip

Can borrow much from computer network practices

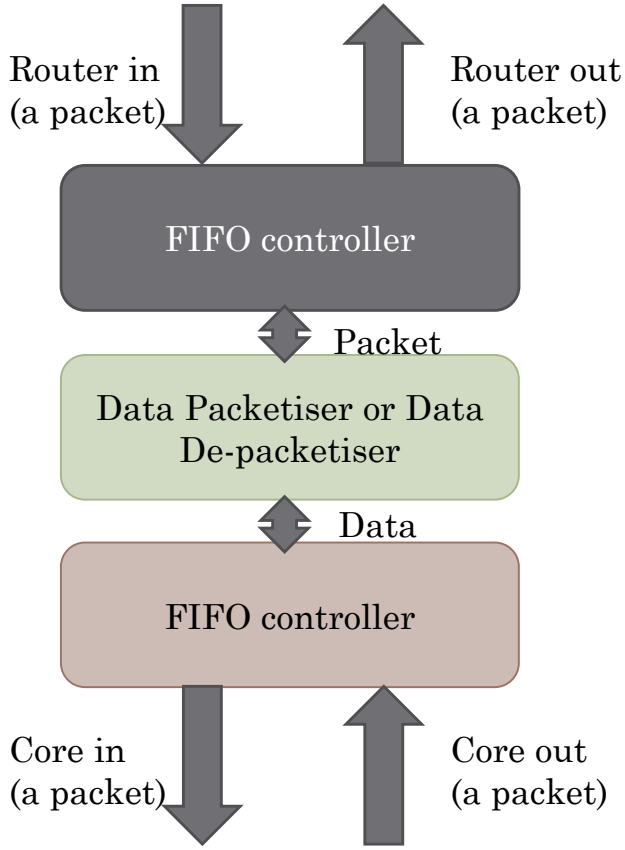
- Layered Approach
- Simpler routing or arbitration is the best



NoC Router (simplified)



NoC Network Interface (simplified)

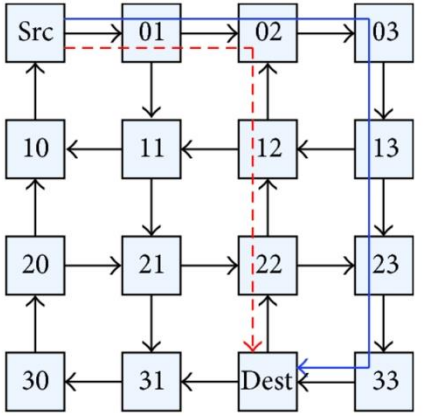


NoC routing

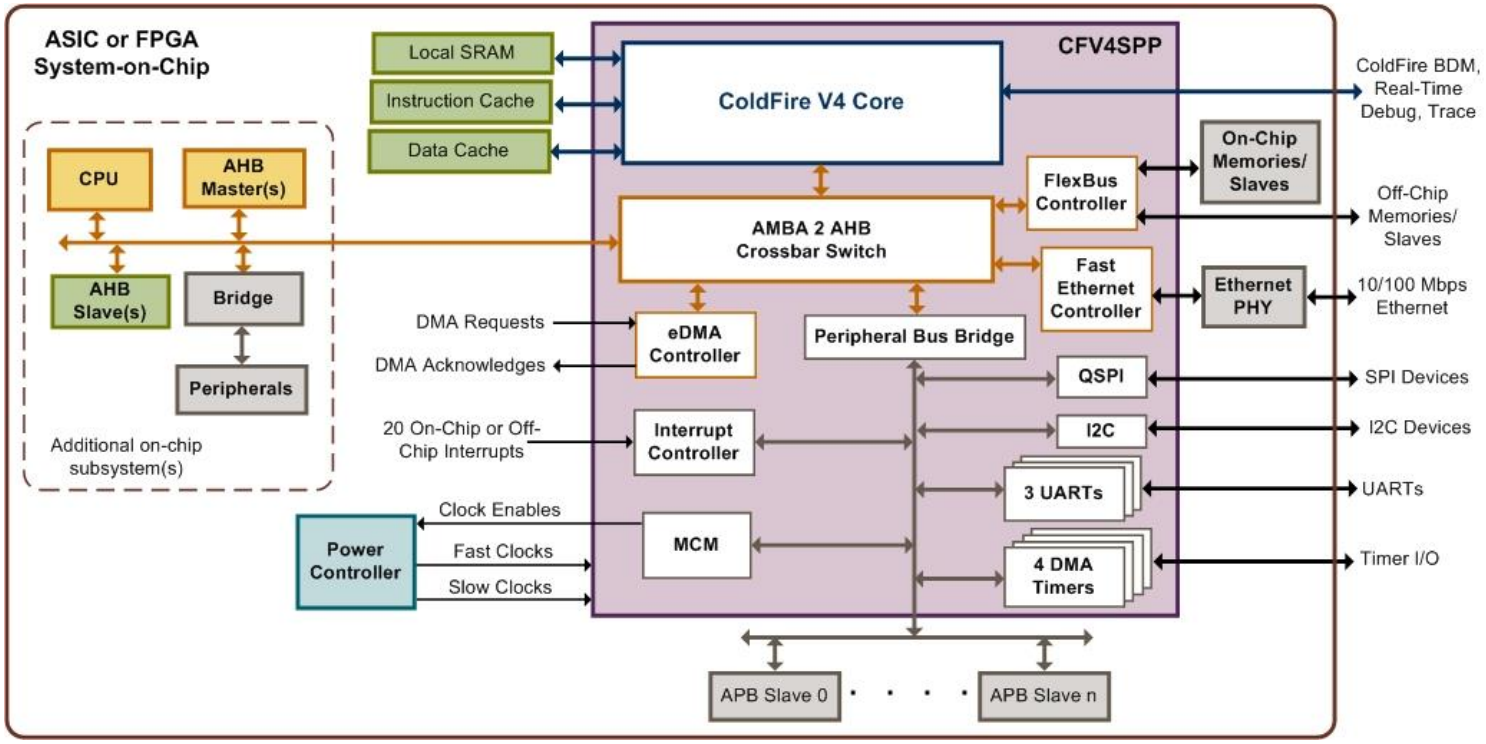
- Affects performance and communication scalability
- Example routing algorithms on-chip
  - Source routing
    - Source tile tells the packet specifically how to reach the destination
    - Simple, but not scalable as sources can be congestion unaware
  - XY routing
    - Go as far close to destination in X direction first; then go in the Y direction
    - Adaptive to NoC congestion, called adaptive XY routing

Example:

- Red line: no contention; go in X direction, and then in Y direction
- Blue line: contention in 02; go as far as X to 03, and then in Y direction; and then go in X direction again. A bit like XYX.



Modern System-on-Chip



Question:

1. what motivated the designer to choose AHB crossbar switch?
2. How do you decide for the right kind of interconnects?

1.2: Exercises / Questions

1. What is interconnect skew? How does it affect interconnect protocols?
2. Explain with examples how shared bus AHB pipelining works.
3. Distinguish between different interconnect topologies.
4. How does NoC work? What are the impacts of NoC routing algorithms?
5. For scalability considerations, which interconnect is the best?
   i. shared bus with multi-layer and burst transactions
   ii. NoC with source-based routing or point to point interconnect
6. You are asked to design the interconnect for a 16-core system. The cores are organized in four groups (with 4 cores each), each with a shared L3 cache and main memory. What type of interconnect would suit such system? Explain with appropriate reasoning.