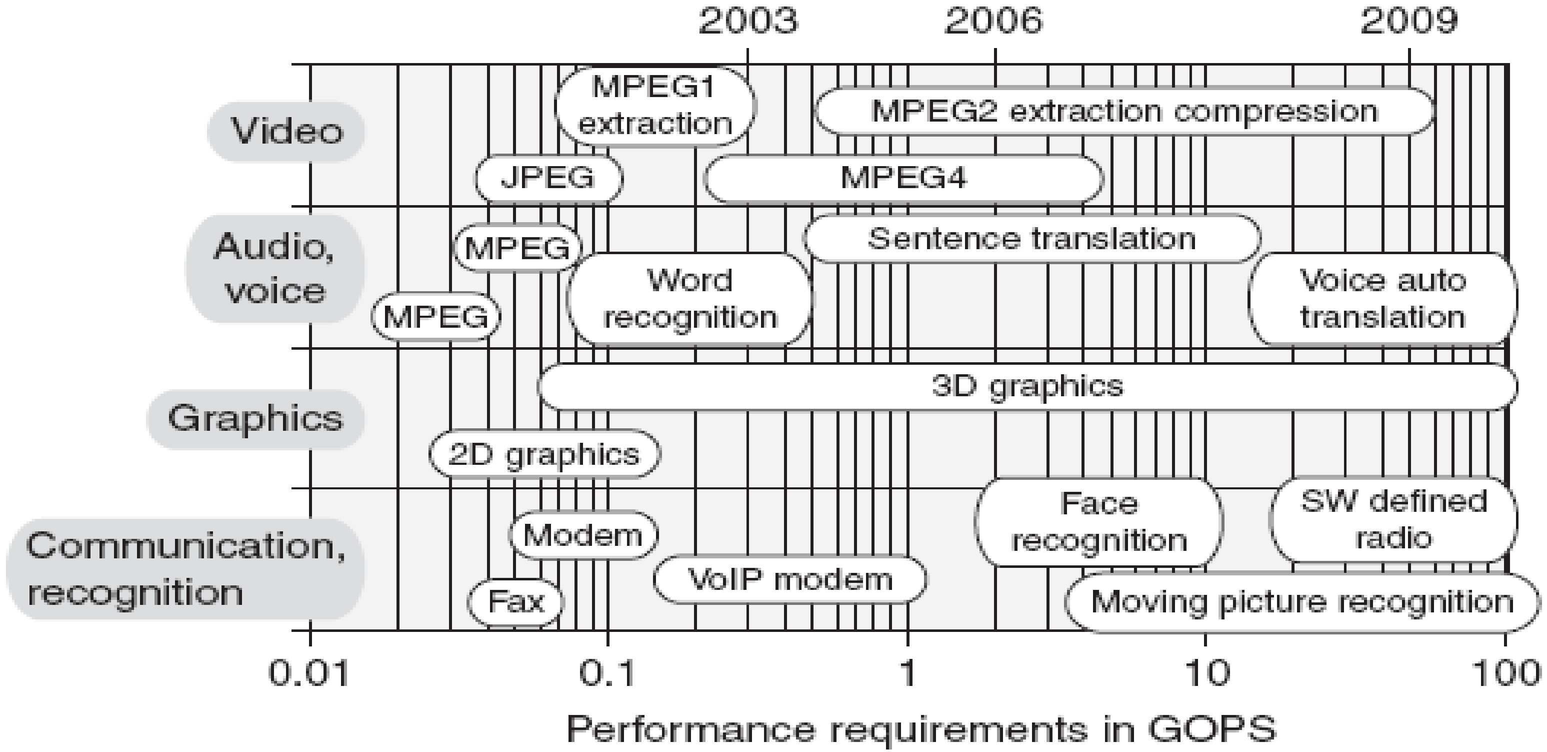
Introduction to Bus Communications

Nishith N. Chakraborty

February, 2025

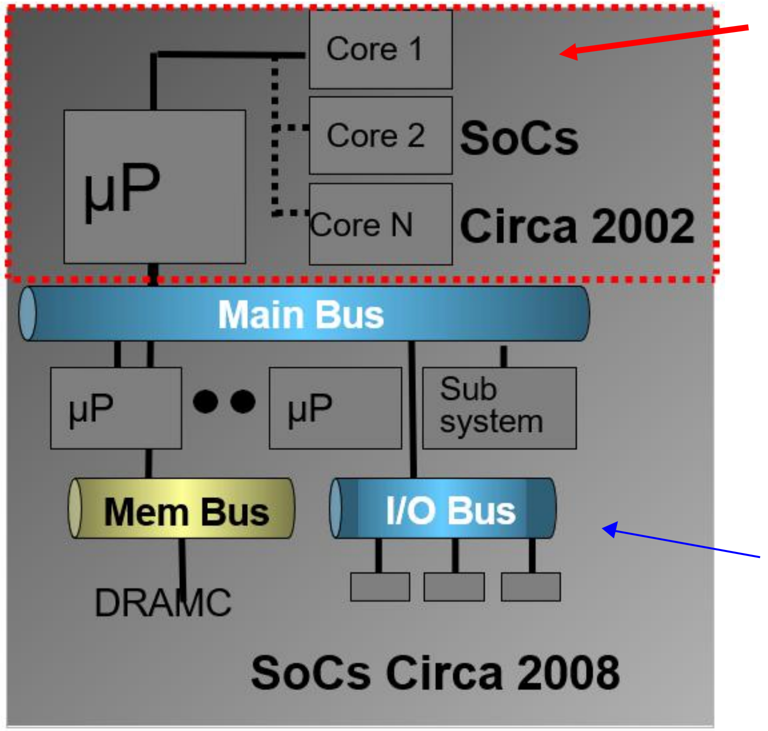
EMERGING APPLICATIONS REQUIREMENTS



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

CAL POLY
SAN LUIS OBISPO

DATA FLOW VS. PROCESSING



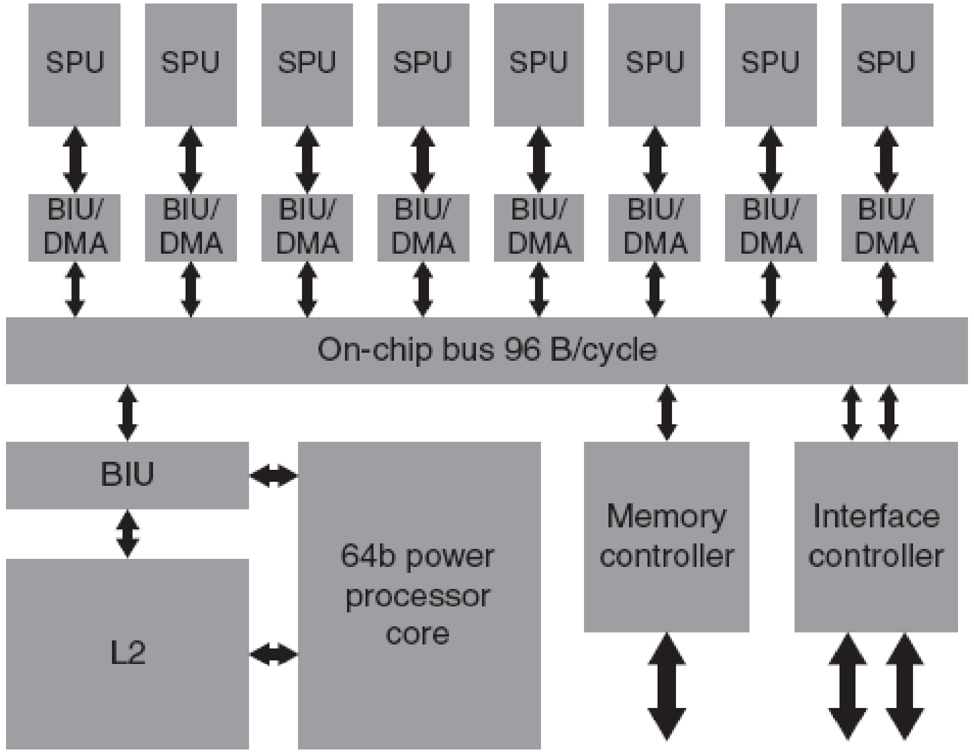
Critical Decision Was uP Choice

- Data flow replacing data processing: major design challenge
- Exploding core counts requiring more advanced interconnects
- EDA cannot solve this architectural problem easily
- Complexity too high to hand craft (and verify!)

Critical Decision Is Interconnect Choice

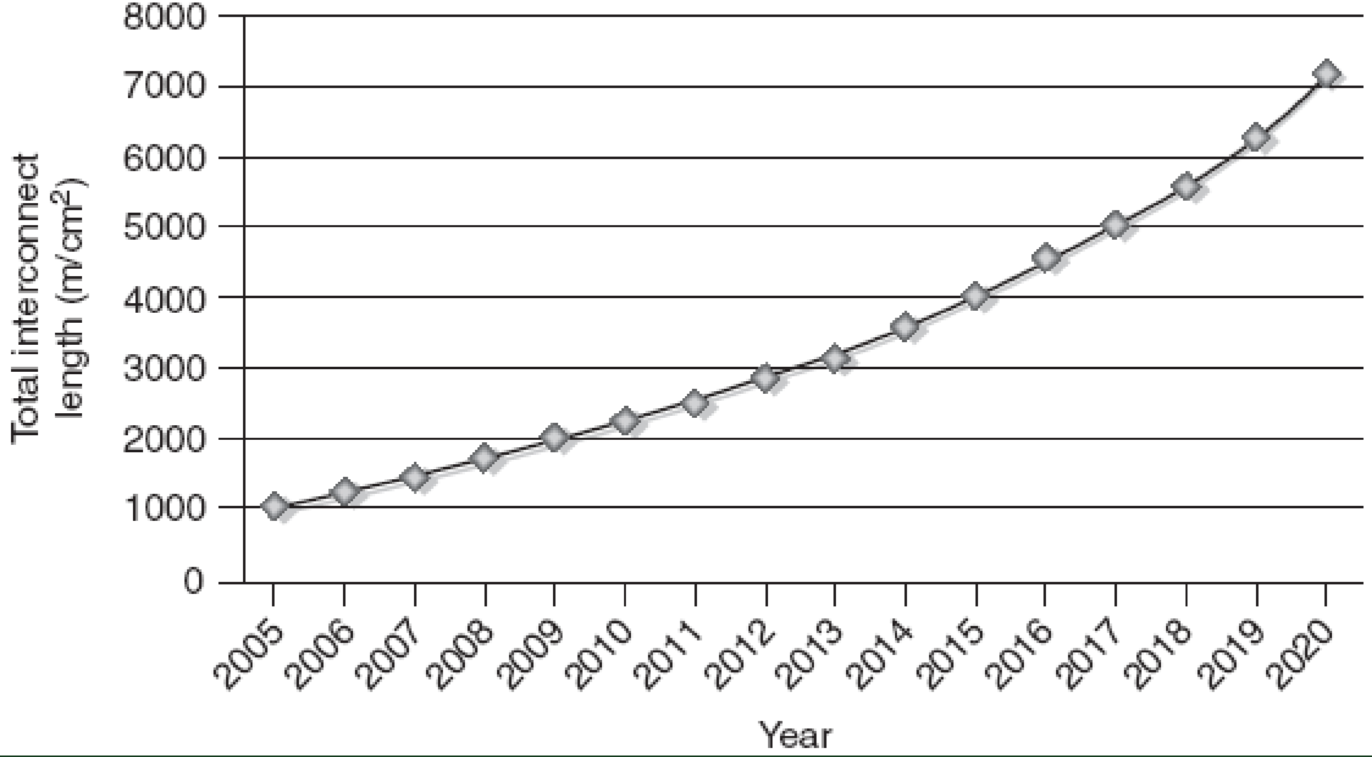
Communication Architecture Design and Verification becoming Highest Priority in Contemporary SoC Design!

EXAMPLE: IBM CELL RING BUS



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

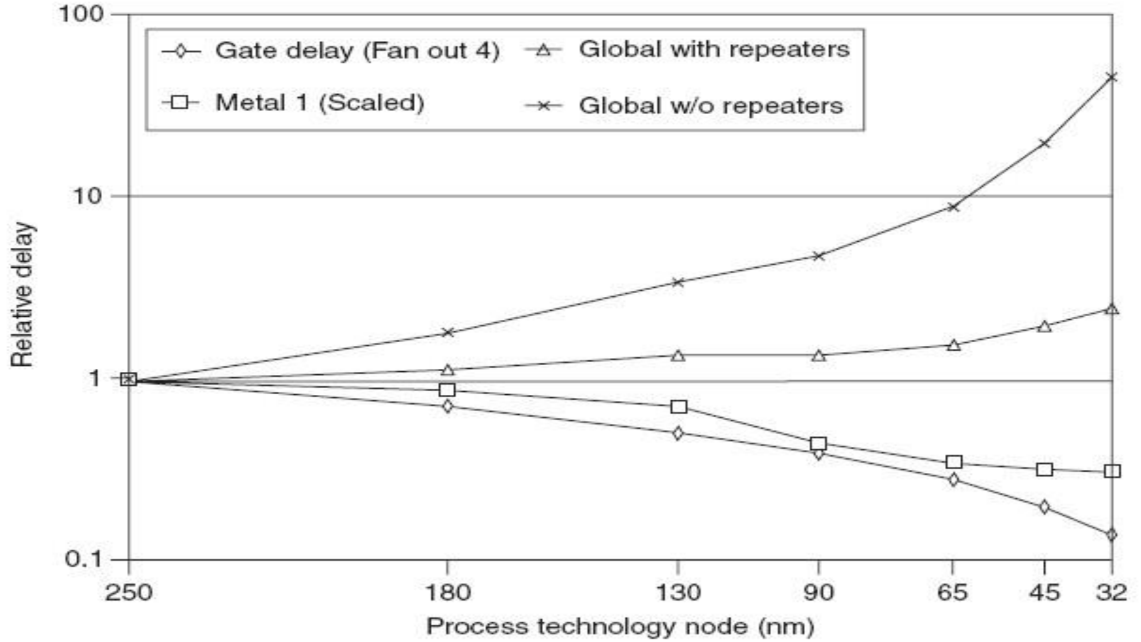
TECH SCALING TRENDS: ON CHIP INTERCONNECT LENGTH



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

TECH SCALING TRENDS: INTERCONNECT PERFORMA

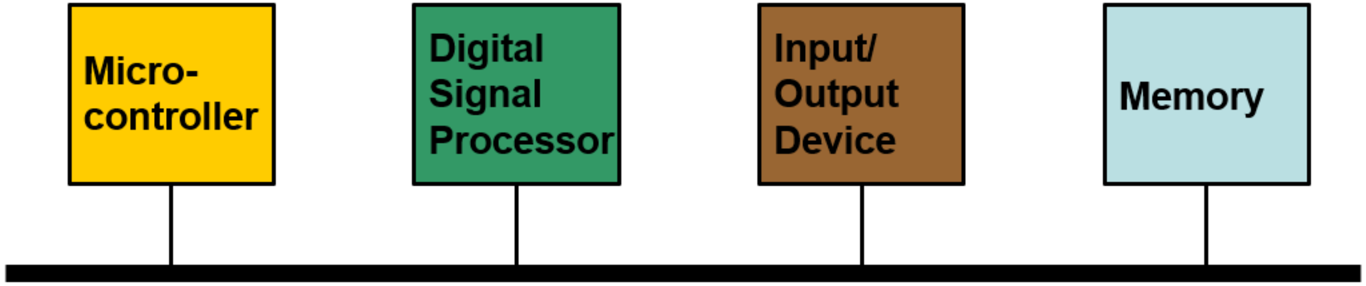
- Increasing wire delay limits achievable performance



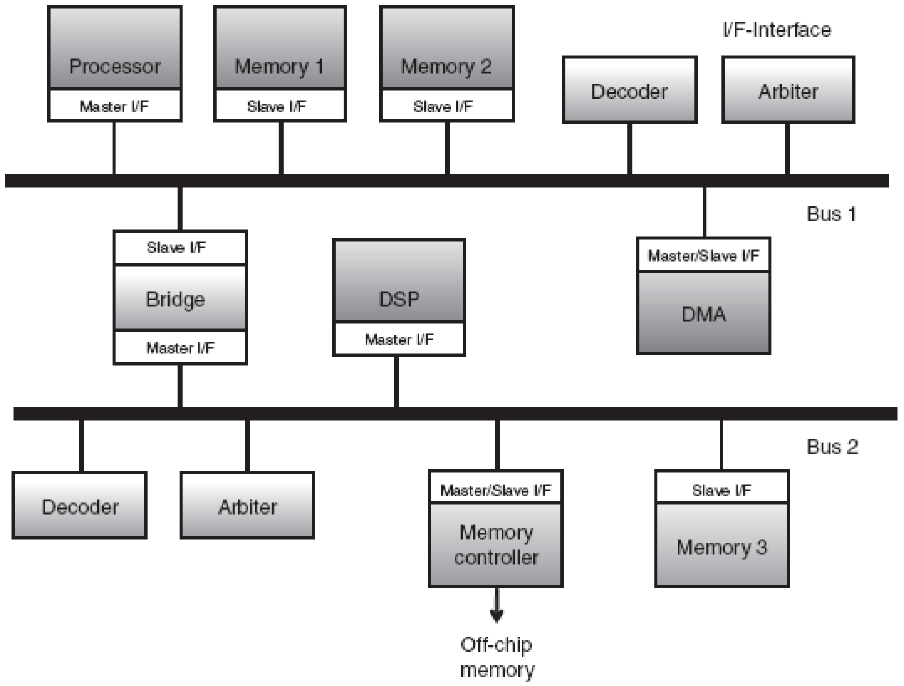
Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS BASED COMMUNICATION ARCHITECTURES

- Buses are the simplest and most widely used SoC interconnection networks
- Bus – collection of signals (wires) with one or more IP components connected
- Only one IP component can transfer data on the shared bus at any given time



BUS TERMINOLOGY



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS TERMINOLOGY

- Master (or Initiator)
  - IP component that initiates a read or write data transfer
- Slave (or Target)
  - IP component that does not initiate transfers and only responds to incoming transfer requests
- Arbiter
  - Controls access to the shared bus
  - Uses arbitration scheme to select master to grant access to bus

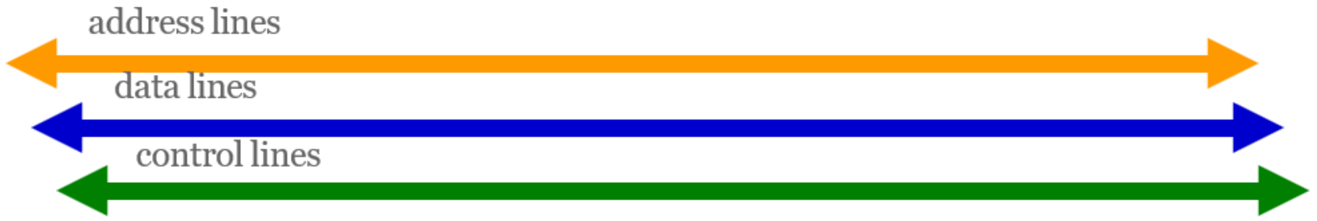
BUS TERMINOLOGY

- Decoder
  - ➢ Determines which component a transfer is intended for
- Bridge
  - ➢ Connects two busses
  - ➢ Acts as slave on one side and master on the other

Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

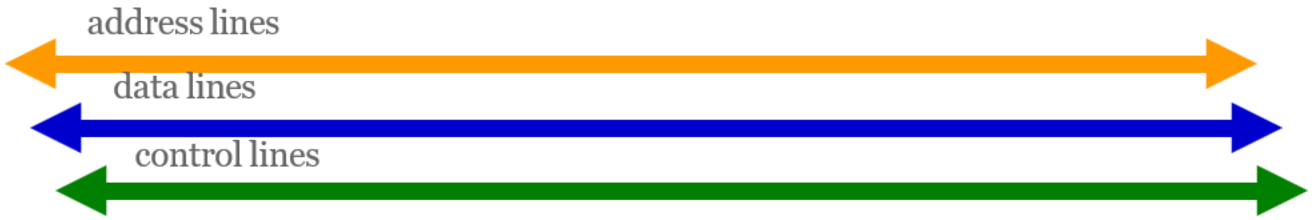
BUS SIGNAL LINES

- Address
  - ➢ Carry address of destination for which transfer is initiated
  - ➢ Can be shared or separate for read, write data
- Data
  - ➢ Carry information between source and destination components
  - ➢ Can be shared or separate for read, write data
  - ➢ Choice of data width critical for application performance

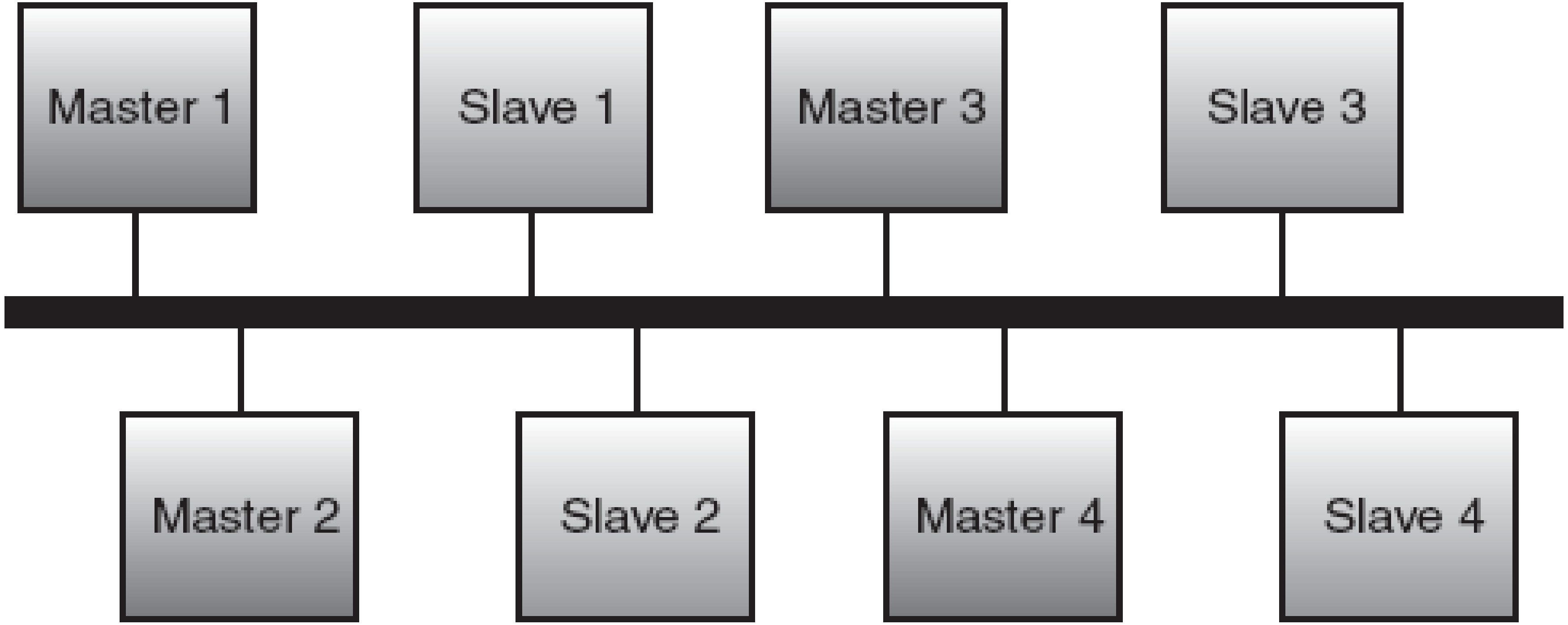


BUS SIGNAL LINES

- Control (Requests and acknowledgements)
  - Specify more information about type of data transfer
    - Byte enable, burst size, cacheable/bufferable, write-back/through



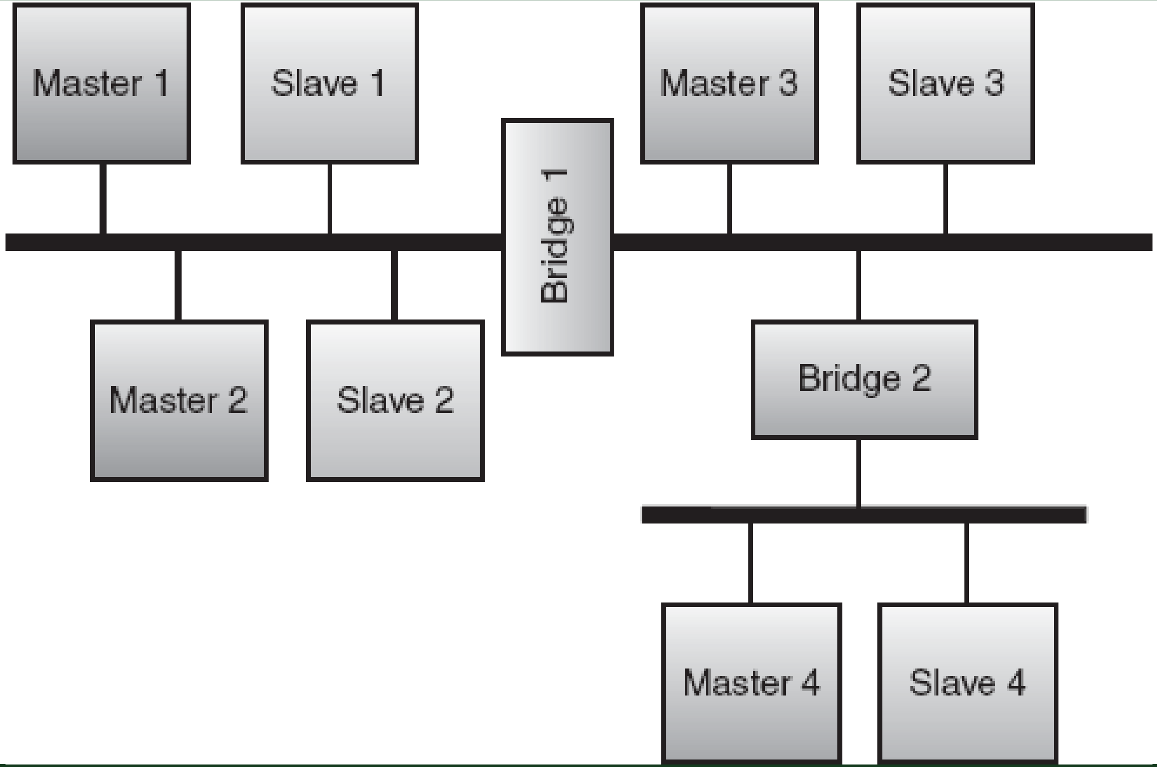
BUS TOPOLOGIES: SHARED



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS TOPOLOGIES: HIERARCHICAL

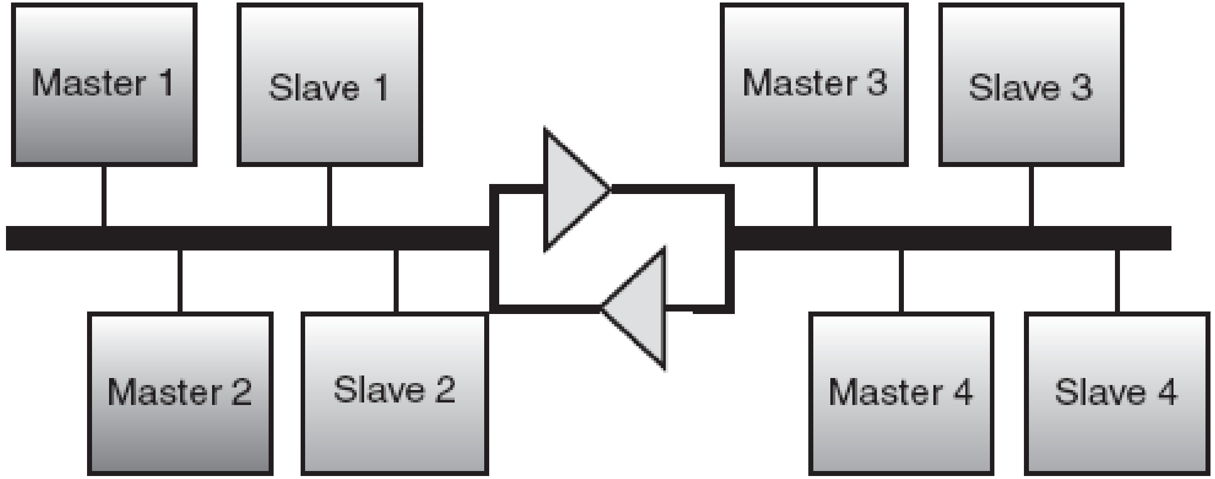
- Improves system throughput
- Multiple ongoing transfers on different buses



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

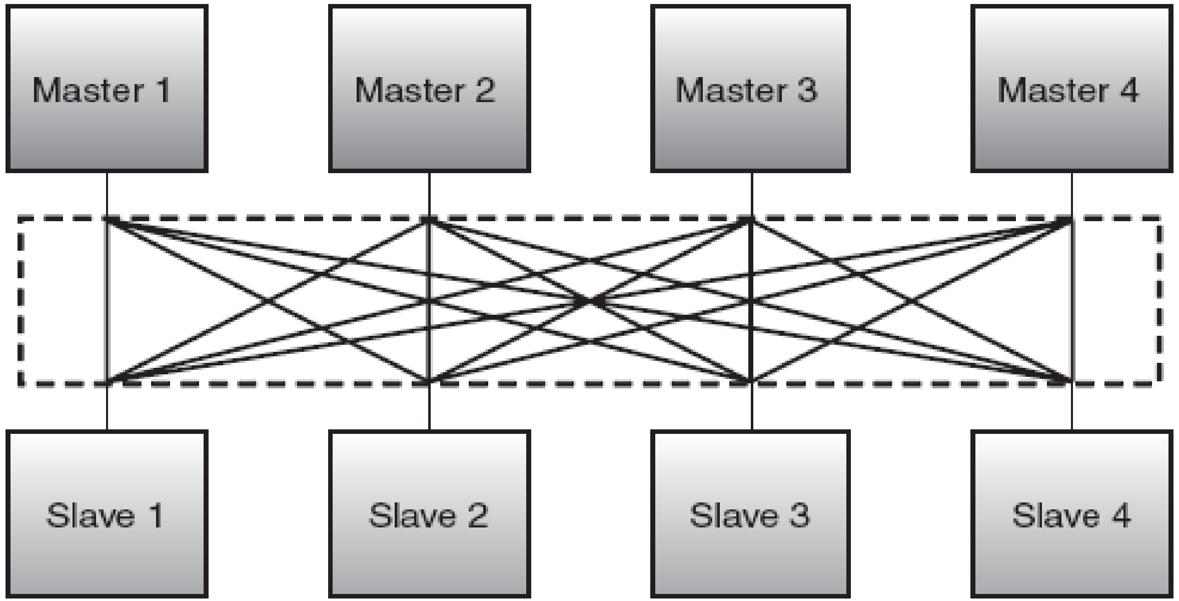
BUS TOPOLOGIES: HIERARCHICAL

- Reduces impact of capacitance across two segments
- Reduces contention and energy



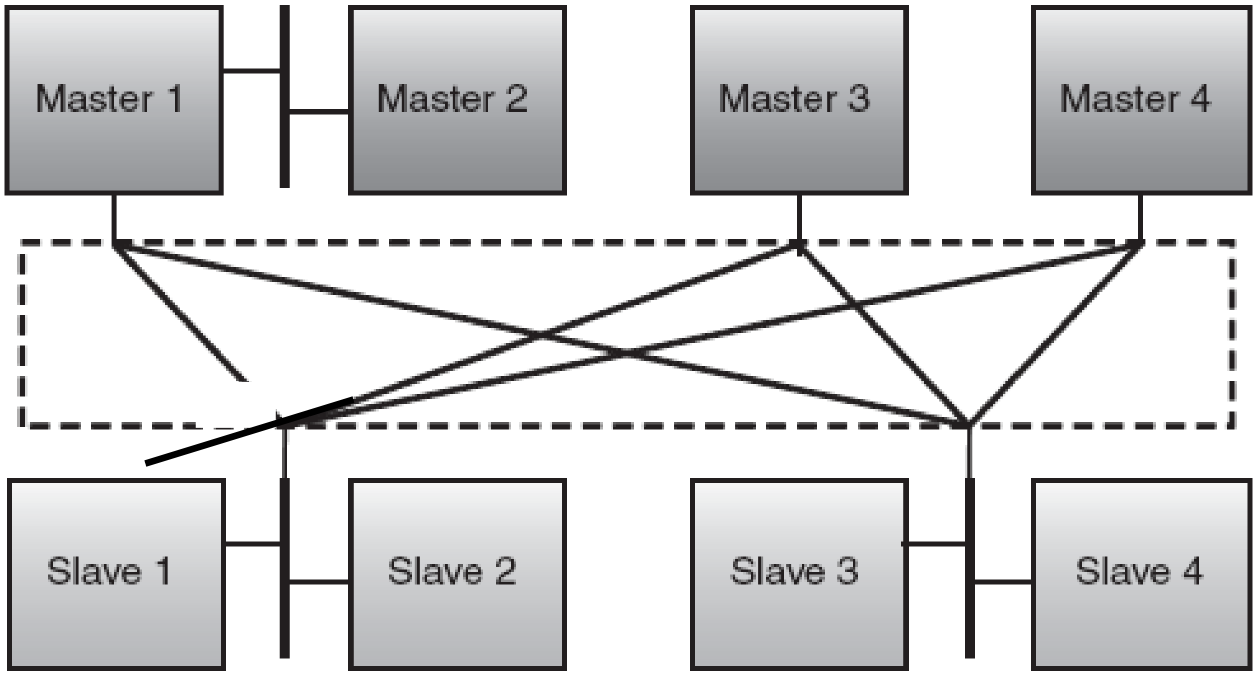
Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS TOPOLOGIES: FULL MATRIX/CROSSBAR

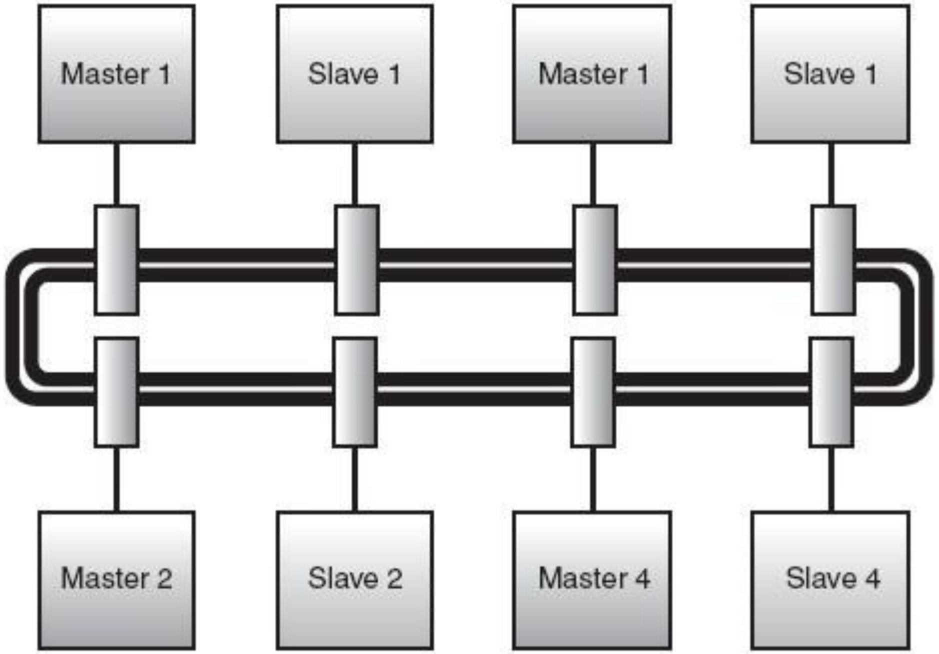


Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS TOPOLOGIES: FULL MATRIX/CROSSBAR

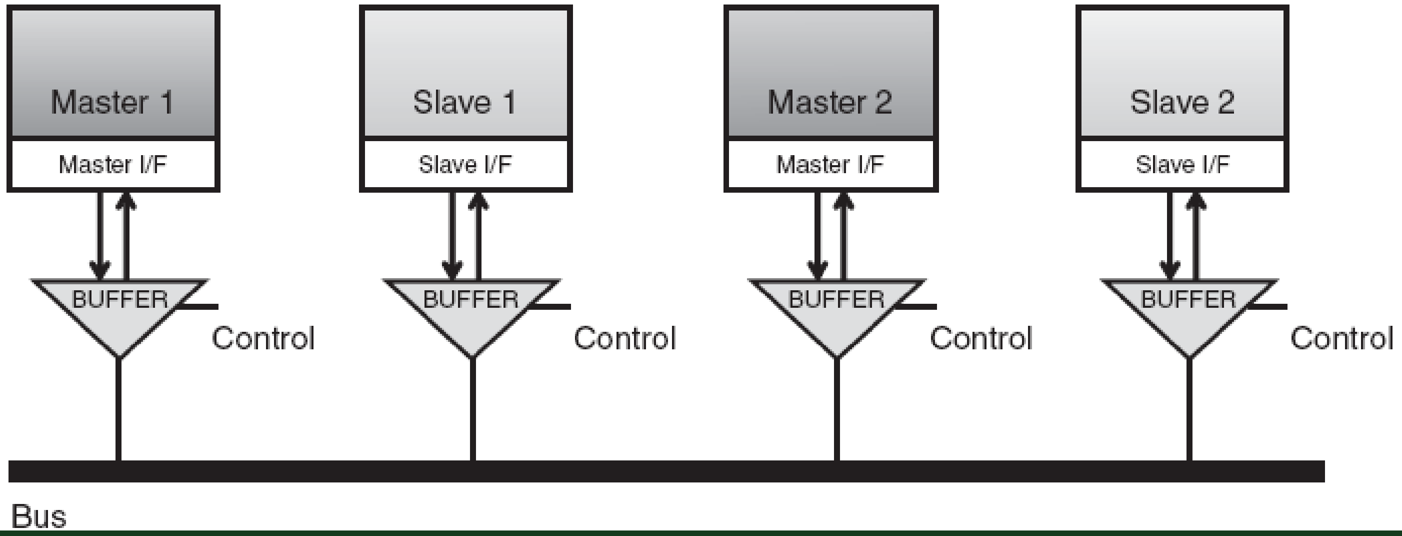


BUS TOPOLOGIES: RING BUS



BUS PHYSICAL STRUCTURE: TRI-STATE BUFFER BASED

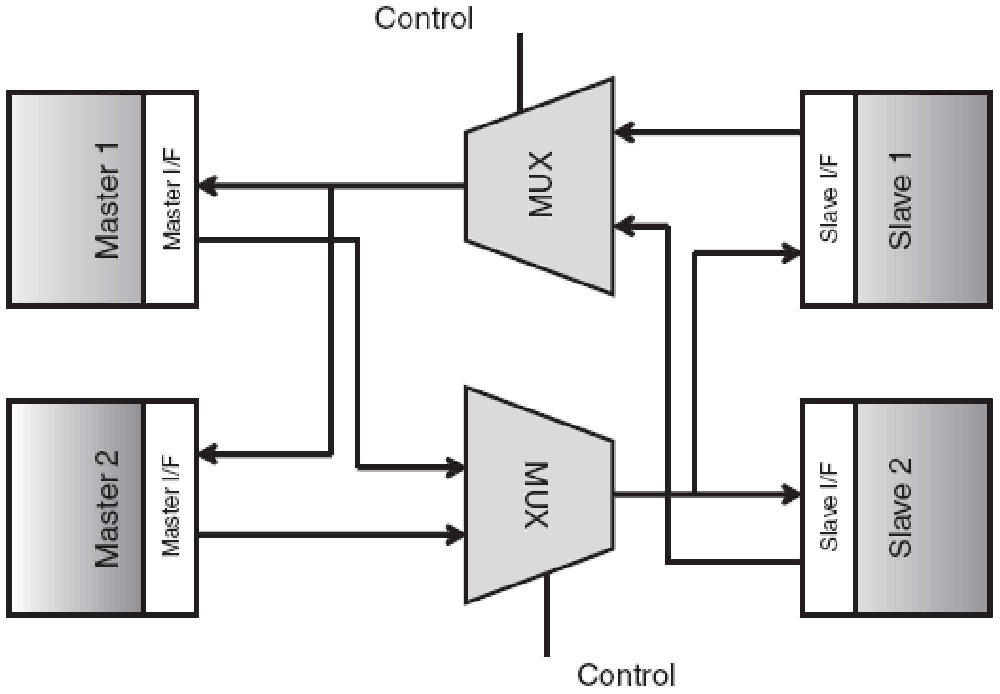
- Commonly used in off-chip/backplane buses
  - Pro: take up fewer wires, smaller area footprint
  - Con: higher power consumption & delay, hard to debug



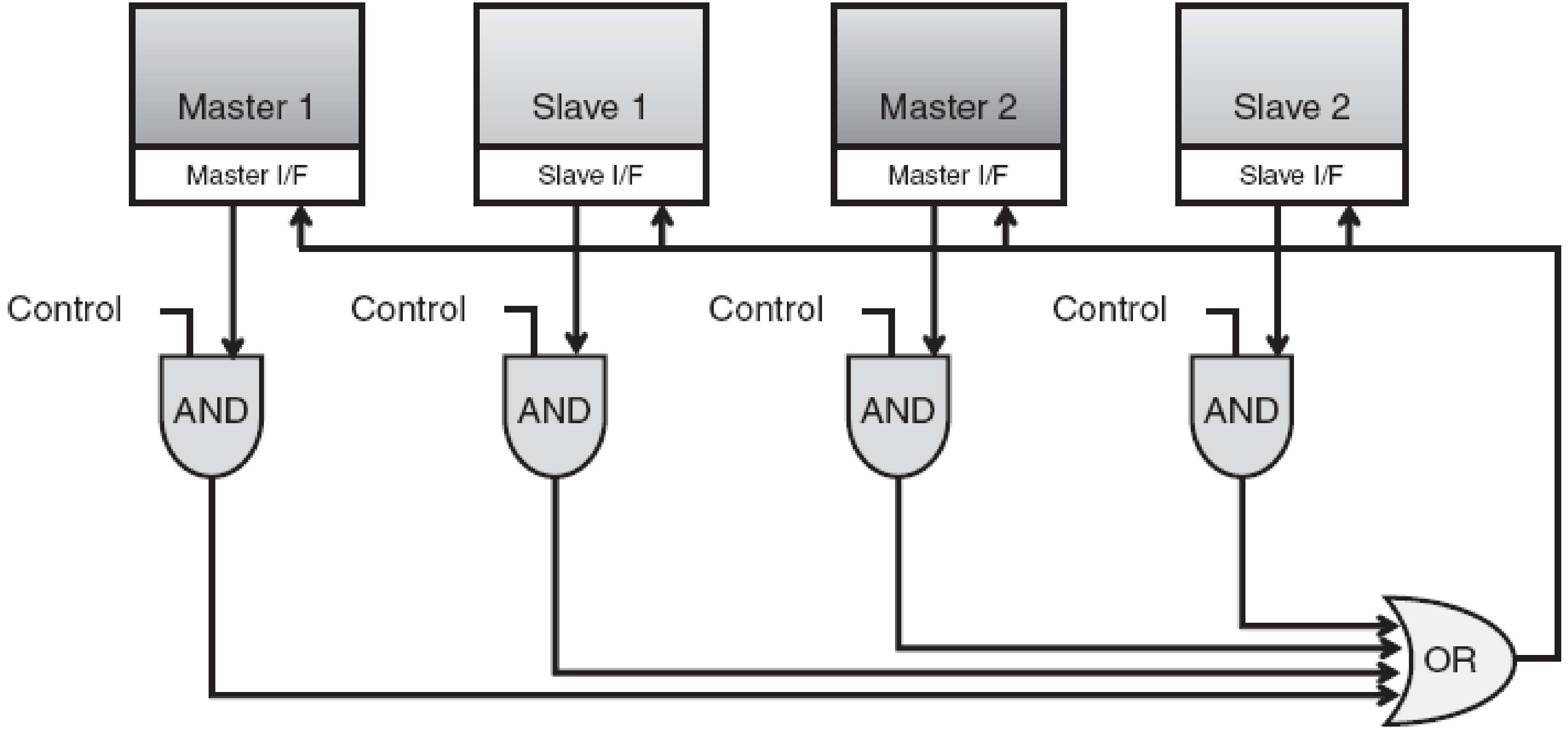
Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS PHYSICAL STRUCTURE: MUX BASED

- Separate read, write channels



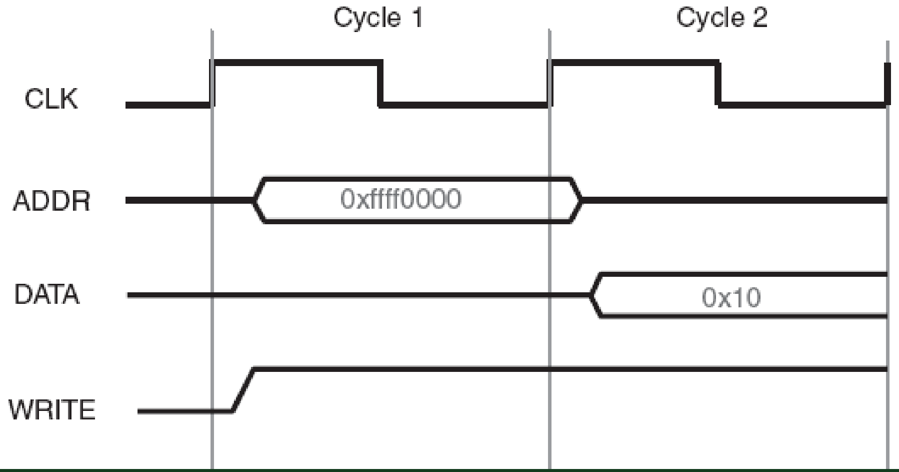
BUS PHYSICAL STRUCTURE: AND-OR BASED



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS CLOCKING: SYNCHRONOUS

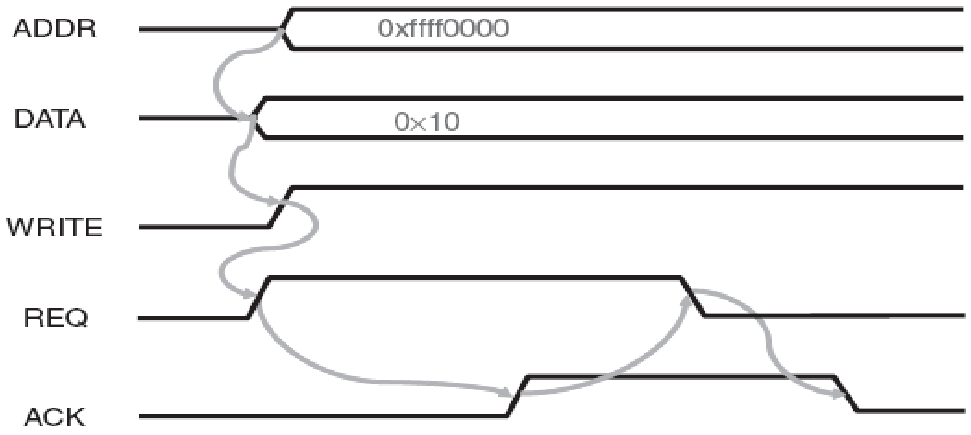
- Includes a clock in control lines
- Fixed protocol for communication that is relative to clock
- Involves very little logic and can run very fast
- Require frequency converters across frequency domains



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS: ASYNCHRONOUS

- Not clocked
- Requires a handshaking protocol
  - ➢ Performance not as good as that of synchronous bus
  - ➢ No need for frequency converters, but does need extra lines
- Does not suffer from clock skew like the synchronous bus



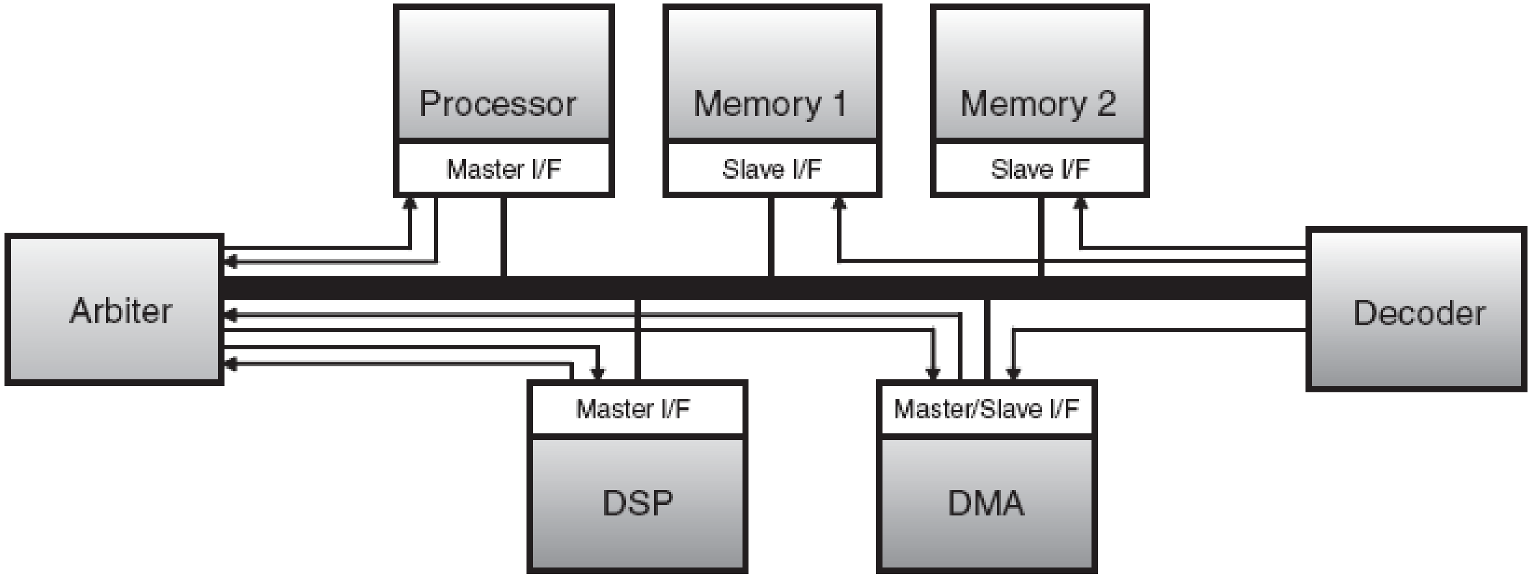
Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

DECODING AND ARBITRATION

- Decoding
  - Determines the target for any transfer initiated by a master
- Arbitration
  - Decides which master can use the shared bus if more than one master request bus access simultaneously
- Decoding and Arbitration can be:
  - Centralized
  - Distributed

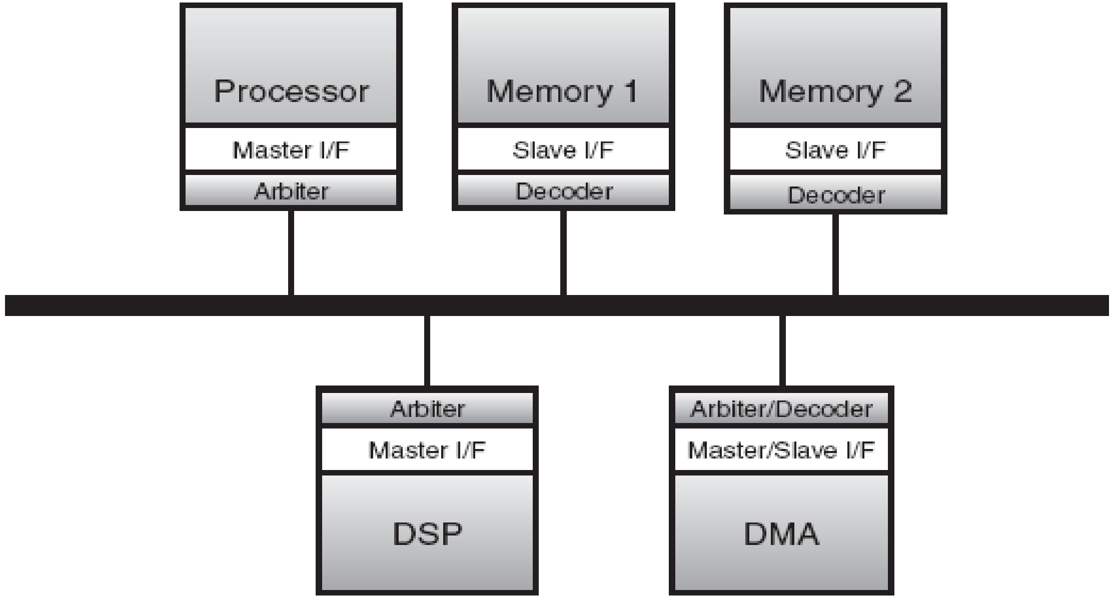
CENTRALIZED DECODING AND ARBITRATION

- Minimal change is required if new components are added to the system



DISTRIBUTED DECODING AND ARBITRATION

- Pro: requires fewer signals compared to the centralized approach
- Con: more hardware duplication, more logic/area, less scalable



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

ARBITRATION SCHEMES

- Random
  - Randomly select master to grant bus access to
- Static priority
  - Masters assigned static priorities
  - Higher priority master request always serviced first
  - Can be pre-emptive (AMBA2) or non-preemptive (AMBA3)
  - May lead to starvation of low priority masters
- Round Robin (RR)
  - Masters allowed to access bus in a round-robin manner
  - No starvation – every master guaranteed bus access
  - Inefficient if masters have vastly different data injection rates
  - High latency for critical data streams

CAL POLY
SAN LUIS OBISPO

ARBITRATION SCHEMES

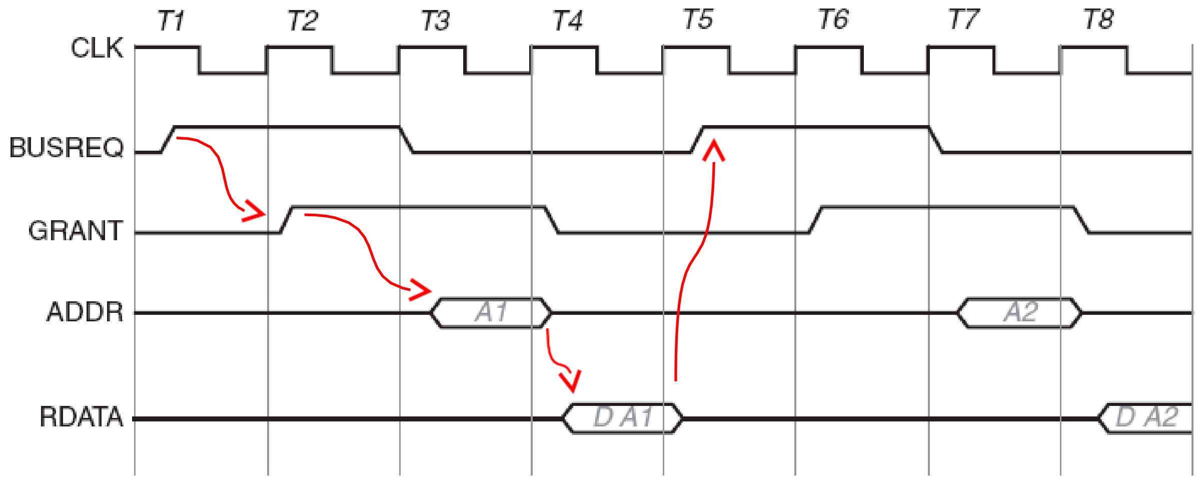
- TDMA
  - Time division multiple access
  - Assign slots to masters based on BW requirements
  - If a master does not have anything to read/write during its time slots, leads to low performance
  - Choice of time slot length and number critical
- TDMA/RR
  - Two-level scheme
  - If master does not need to utilize its time slot, second level RR scheme grants access to another waiting master
  - Better bus utilization
  - Higher implementation cost for scheme (more logic, area)

ARBITRATION SCHEMES

- Dynamic priority
  - Dynamically vary priority of master during application execution
  - Gives masters with higher injection rates a higher priority
  - Requires additional logic to analyze traffic at runtime
  - Adapts to changing data traffic profiles
  - High implementation cost (several registers to track priorities and traffic profiles)
- Programmable priority
  - Simpler variant of dynamic priority scheme
  - Programmable register in arbiter allows software to change priority

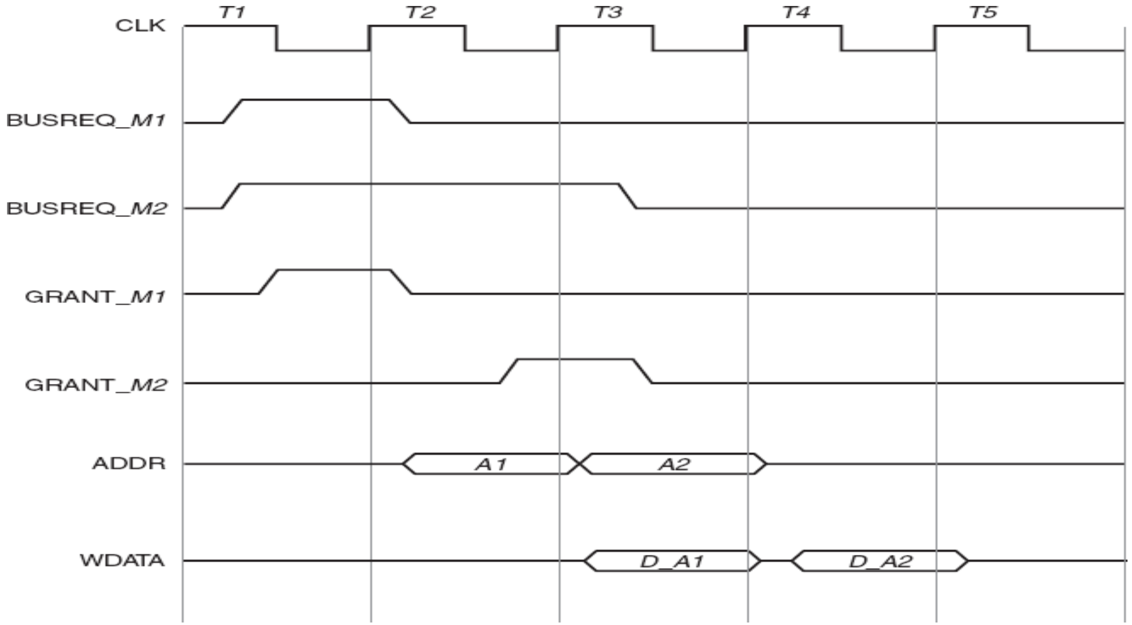
BUS DATA TRANSFER MODES: SINGLE NON-PIPELINED

- Simplest transfer mode
- First request for access to bus from arbiter
- On being granted access, set address and control signals
- Send/receive data in subsequent cycles



BUS DATA TRANSFER MODES: PIPELINED

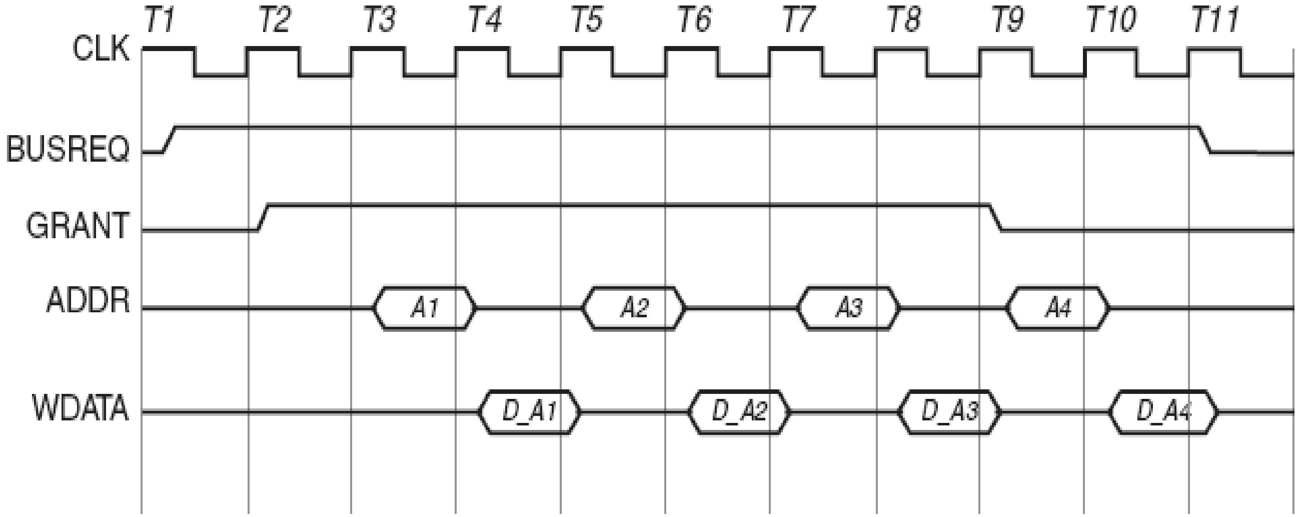
- Overlap address and data phases
- Only works if separate address and data buses are present



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

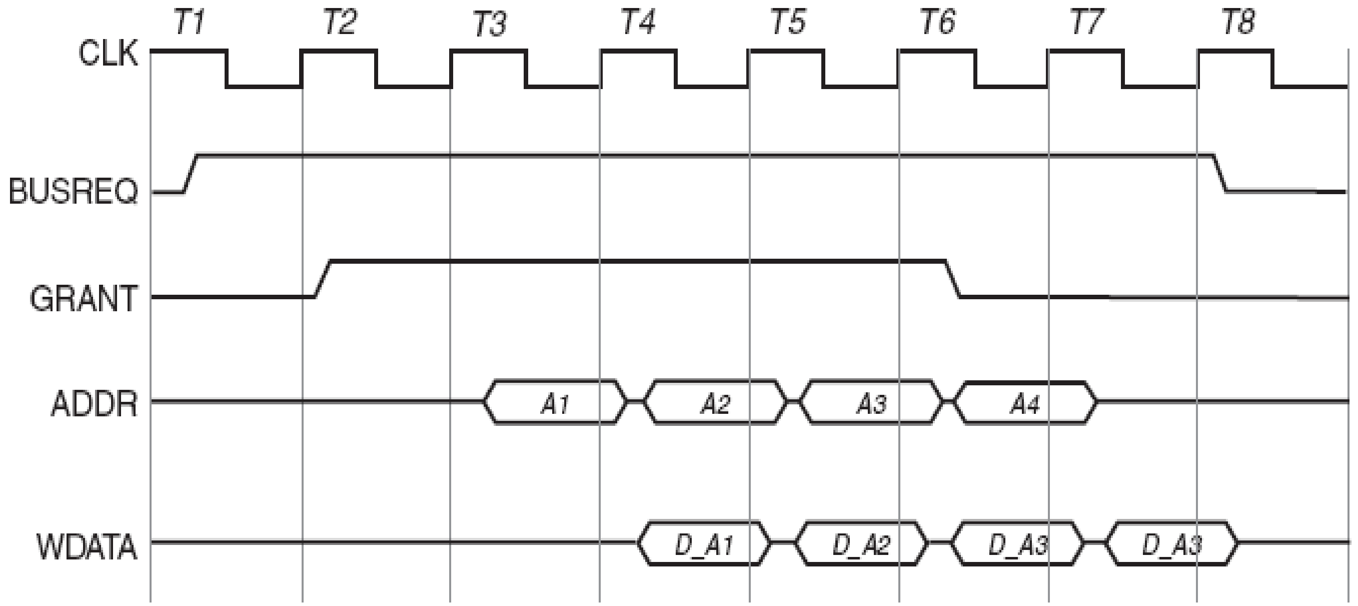
BUS DATA TRANSFER MODES: NON-PIPELINED BURST

- Send multiple data, arbitrate once for entire transaction
- Master indicates to arbiter intention to perform burst transfer
- Saves time spent requesting for arbitration



BUS DATA TRANSFER MODES: PIPELINED BURST

- Useful when separate address and data buses available
- Reduces data transfer latency



Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

BUS DATA TRANSFER MODES: SPLIT TRANSFER

- If slaves take a long time to read/write data, it can prevent other masters from using the bus
- Split transfers improve performance by 'splitting' a transaction
    - ➢ Master sends read request to slave
    - ➢ Slave relinquishes control of bus as it prepares data
        - ▪ Arbiter can grant bus access to another waiting master
        - ▪ Allows utilizing otherwise idle cycles on the bus
    - ➢ When slave is ready, it requests bus access from arbiter
    - ➢ On being granted access, it sends data to master
- Explicit support for split transfers required from slaves and arbiters (additional signals, logic)

BUS DATA TRANSFER MODES: OUT-OF-ORDER TRANSFER

- Multiple transfers from different masters, or even from the same master, are SPLIT by slave and are in progress simultaneously on single bus
- Masters can initiate transfers without waiting for earlier transfers to complete
- Allows better parallelism, performance in buses
- Additional signals needed to transmit IDs for every data transfer in the system
- Master interfaces need to be extended to handle data transfer IDs and be able to reorder received data
- Slave interfaces have out-of-order buffers for reads, writes, to keep track of pending transactions, plus logic for processing IDs
    - ➢ Any application typically has a limited buffer size beyond which performance doesn’t increase

BUS DATA TRANSFER MODES: BROADCAST

- Every time a data item is transmitted over a bus, it is physically broadcast to every component on the bus
- Useful for snooping and cache coherence protocols
- Example: when several components on bus have a private cache fed from a single memory, a problem arises when the memory is updated
  - ➢ When a cache line is written to memory by a component
- It is essential that private caches of the components on the bus invalidate (or update) their cache entries
  - ➢ To prevent reading incorrect values
- Broadcasting allows address of the memory location (or cache line) being updated to be transmitted to all the components on the bus, so they can invalidate (or update) their local copies

Thank you!