EE382M.20: System-on-Chip (SoC) Design

Lecture 9 – SoC Communication Architectures

Source:
Sudeep Pasricha (Colorado State), Nikil Dutt (UC Irvine)
"On-Chip Communication Architectures", Morgan Kaufmann, 2008

Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
`gerstl@ece.utexas.edu`

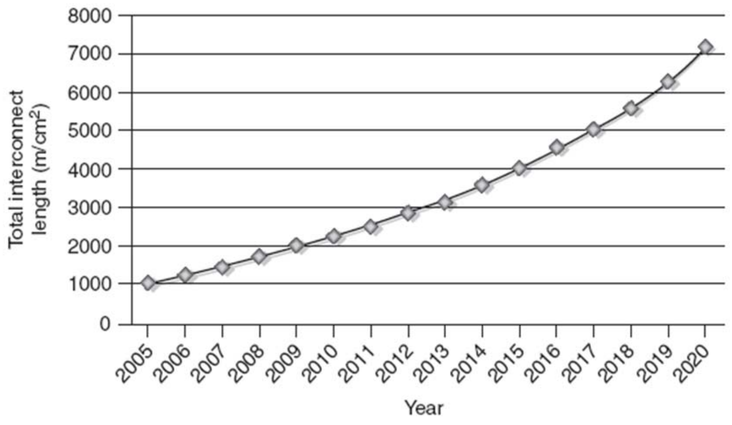
The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Lecture 9: Outline

- **Introduction**
  - Communication-centric design
- **Bus-based architectures**
  - Topologies and structures
  - Decoding, arbitration, transfer modes
- **On-chip communication standards**
  - AMBA and AXI
- **Networks-on-Chip (NoCs)**
  - Topologies, switching, routing

Technology Scaling Trends (1)

- Total Interconnect Length on a Chip

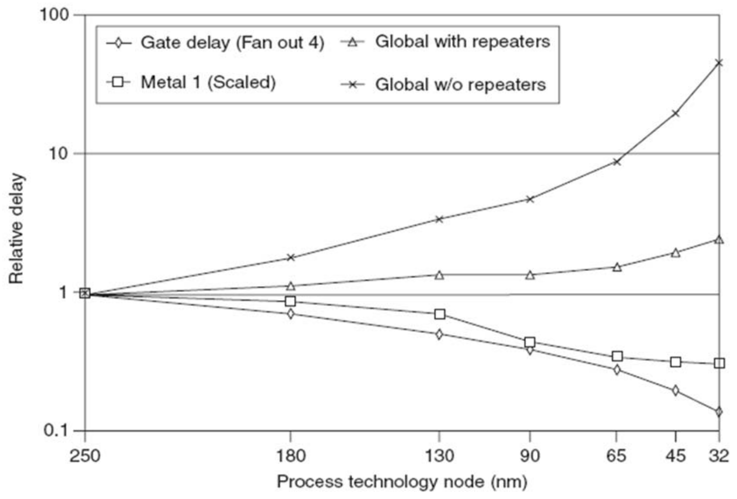


➢ Highlights importance of interconnect design in future technologies

EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 3

Technology Scaling Trends (2)

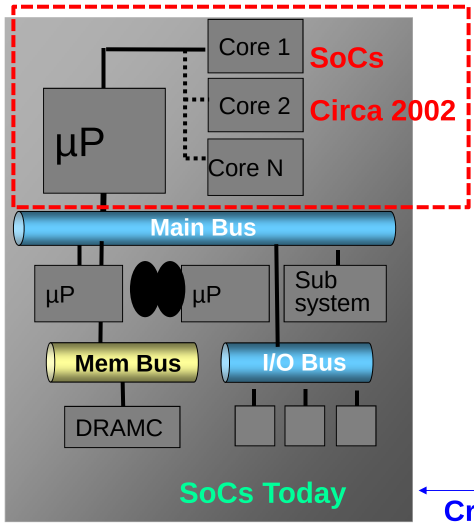
- Relative delay comparison of wires vs. process technology



➢ Increasing wire delay limits achievable performance

EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 4

Communication Trumps Computation



Critical Decision Was uP Choice

- Exploding core counts requiring more advanced Interconnects
- EDA cannot solve this architectural problem easily
- Complexity too high to hand craft (and verify!)

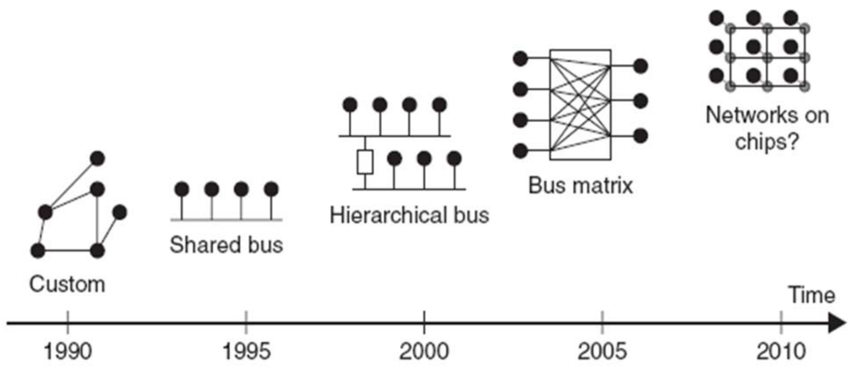
Critical Decision Is Interconnect Choice

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

Source: SONICS Inc.

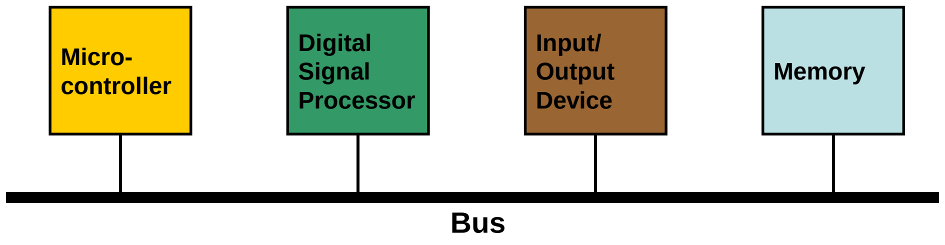
On-Chip Communication Trends

- Evolution of on-chip communication architectures

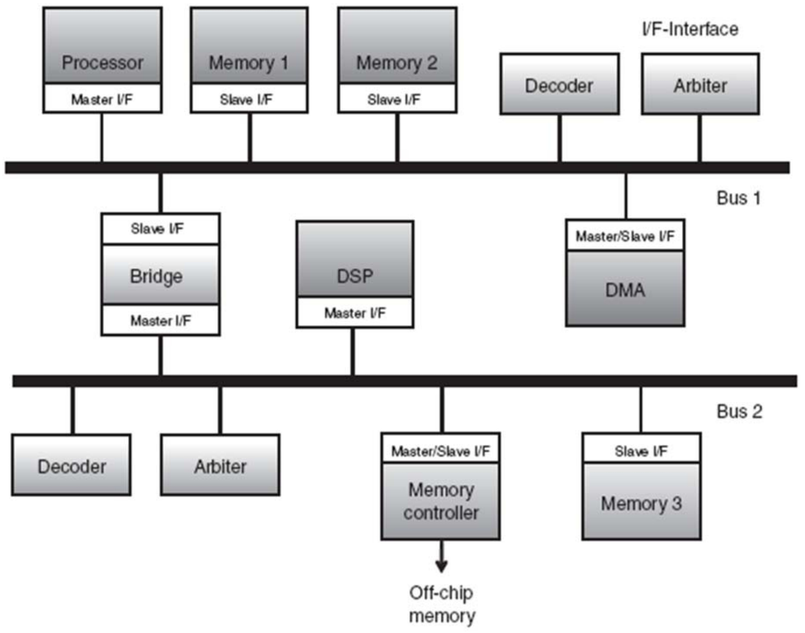


Bus-Based Architectures

- **Buses are the simplest and most widely used SoC interconnection networks**
  - Bus: a collection of signals (wires) to which one or more IP components (which need to communicate data with each other) are connected
- **Only one component can transfer data on the shared bus at any given time**



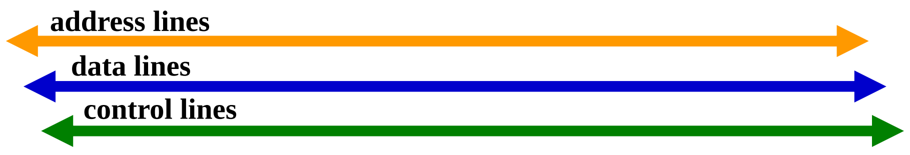
Bus Terminology



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

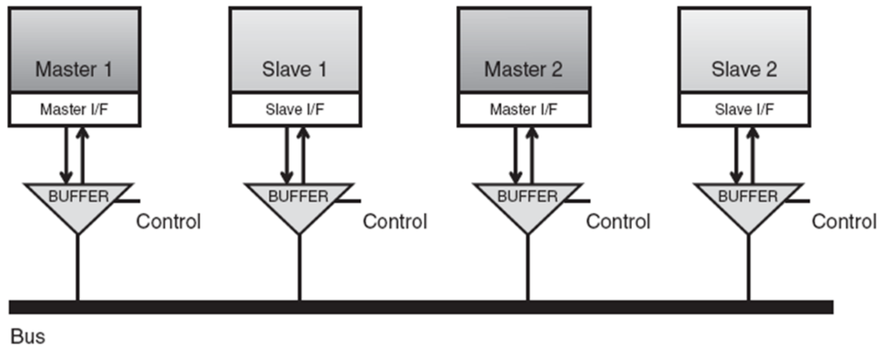
Bus Signal Lines



- **A bus typically consists of three types of 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
  - Control
    - Requests and acknowledgements
    - Specify more information about type of data transfer
    - Byte enable, burst size, cacheable/bufferable, write-back/through, …

Bus Physical Structure (1)

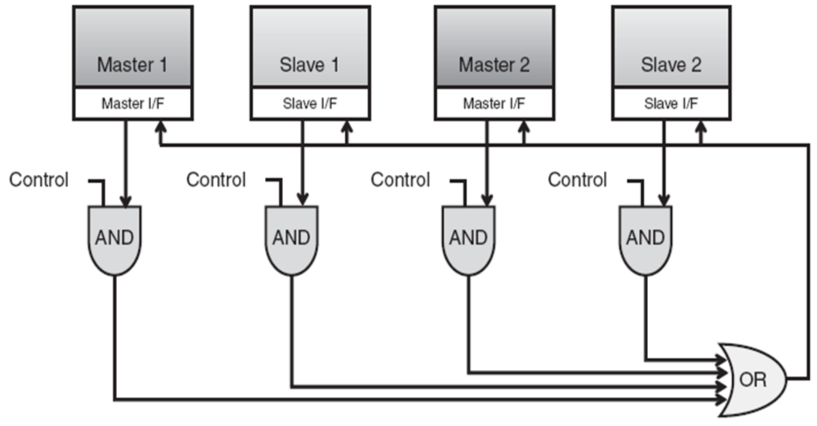
- Tri-state buffer based bidirectional signals



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

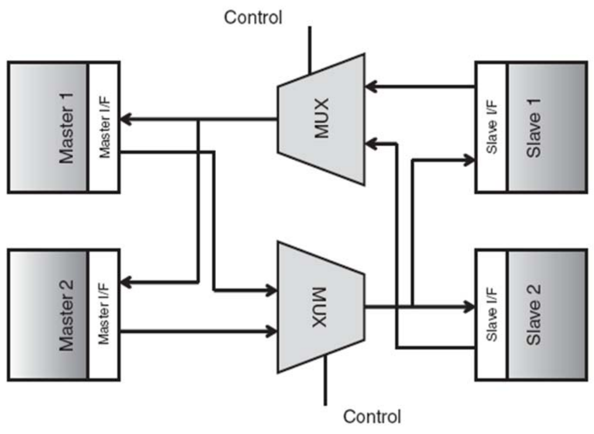
Bus Physical Structure (2)

- AND-OR based signals



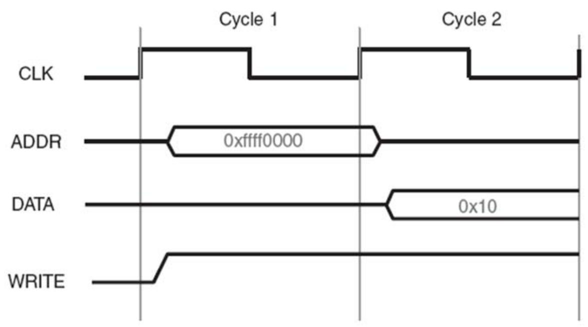
Bus Physical Structure (3)

- **MUX based signals**
  - Separate read, write channels



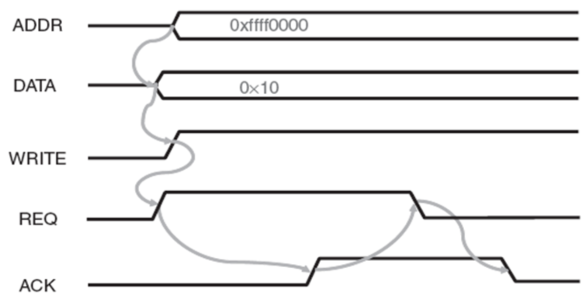
Bus Clocking

- **Synchronous Bus**
  - 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



Bus Clocking

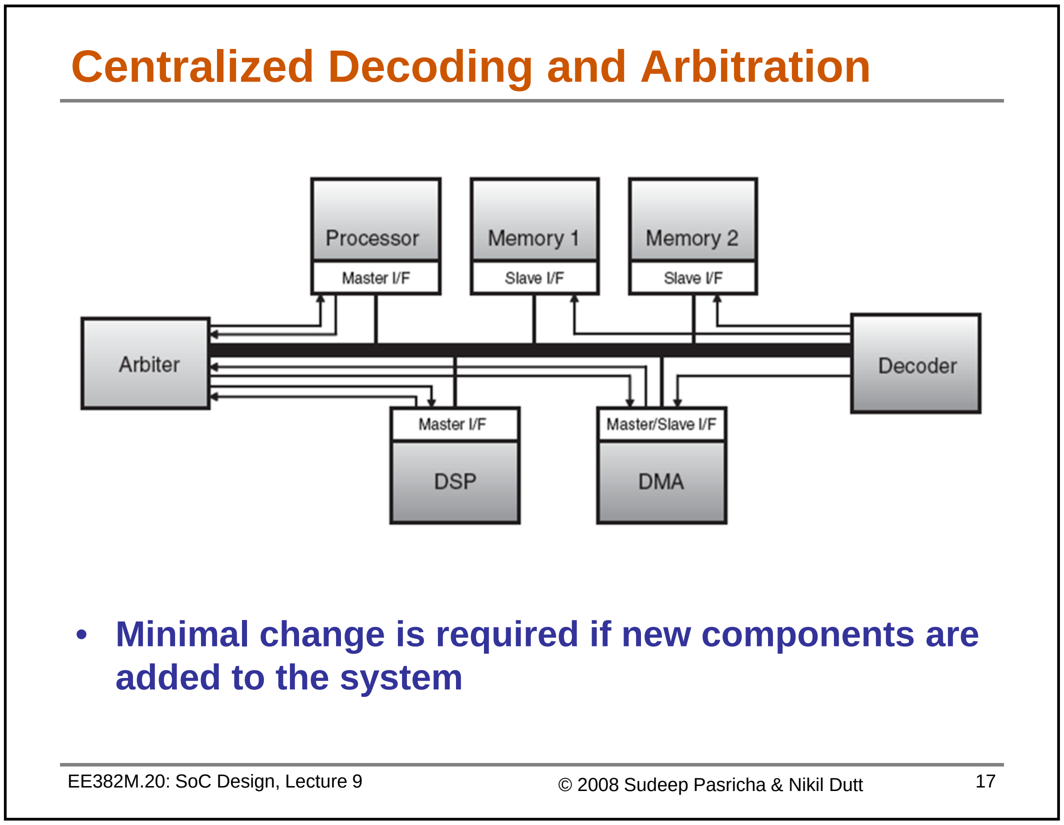
- **Asynchronous Bus**
  - Not clocked
  - Requires a handshaking protocol
    - – performance not as good as that of synchronous bus
    - – No need for frequency converters, but does need extra lines
  - Does not suffer from clock skew like the synchronous bus



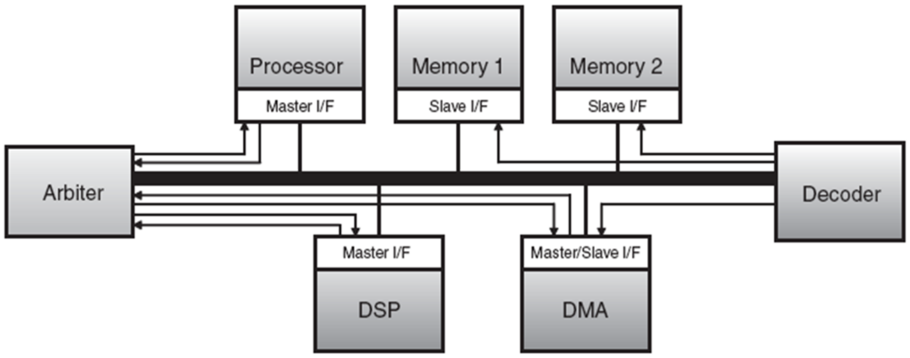
Decoding and Arbitration

- **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

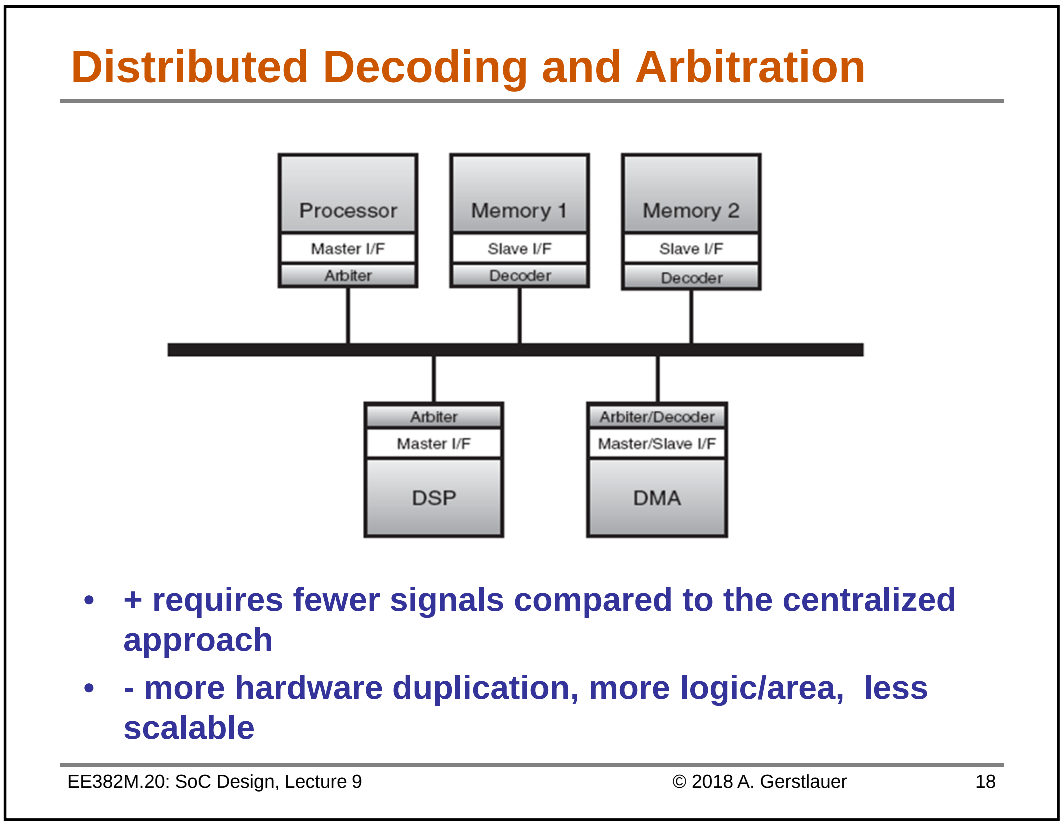
➢ **Decoding and Arbitration can either be**
  - Centralized
  - Distributed



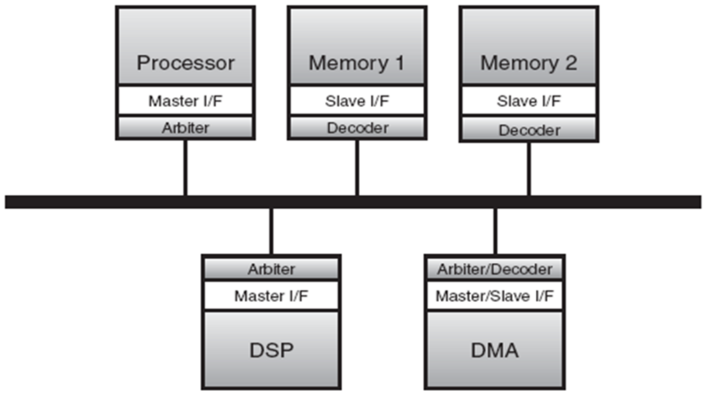
Centralized Decoding and Arbitration



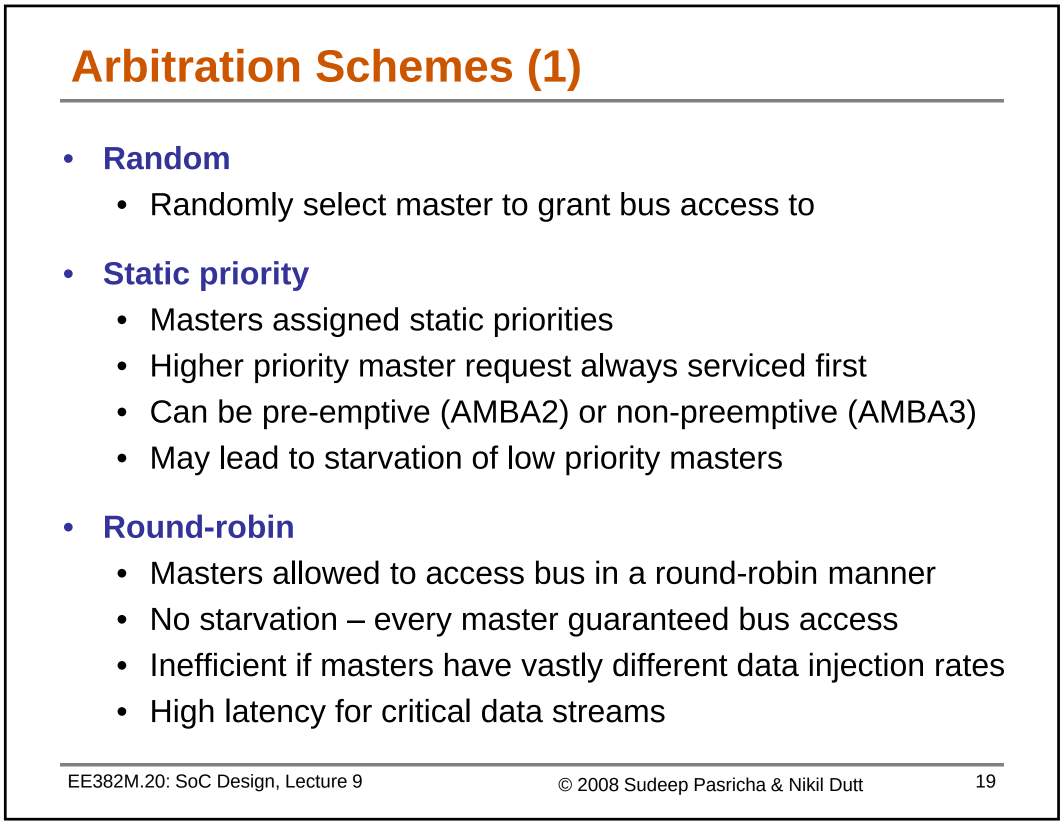
- Minimal change is required if new components are added to the system



Distributed Decoding and Arbitration

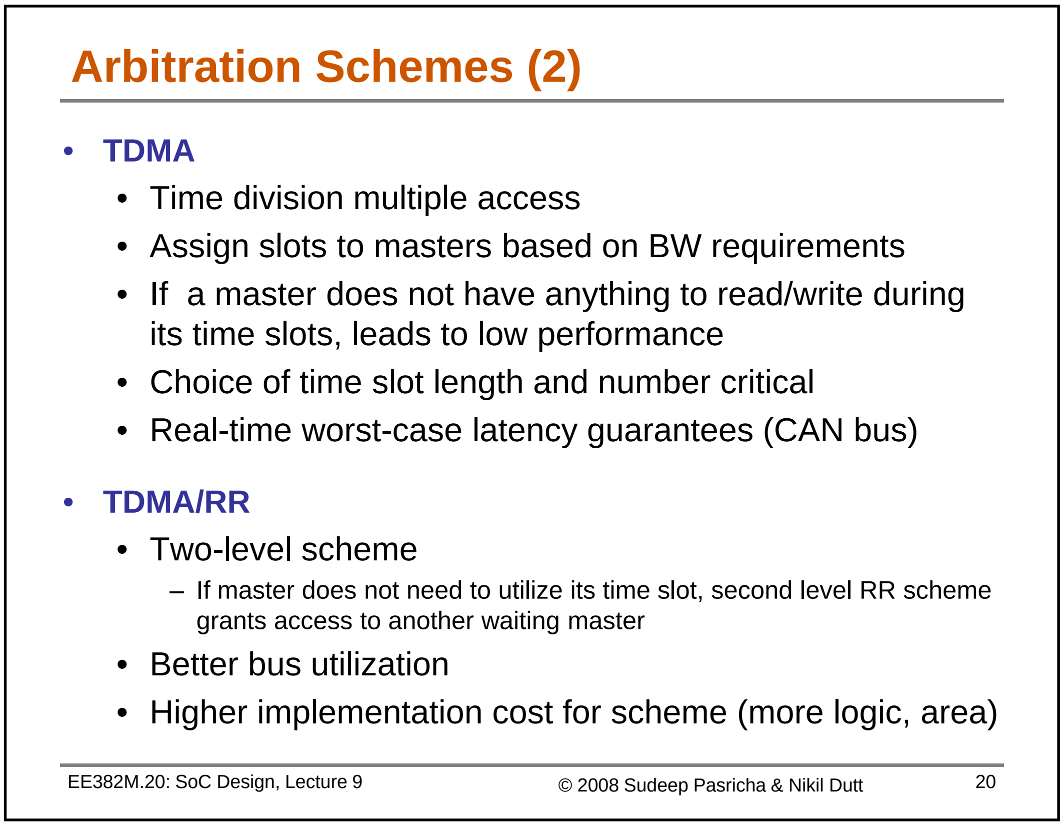


- + requires fewer signals compared to the centralized approach
- - more hardware duplication, more logic/area, less scalable



Arbitration Schemes (1)

- **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**
  - 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



Arbitration Schemes (2)

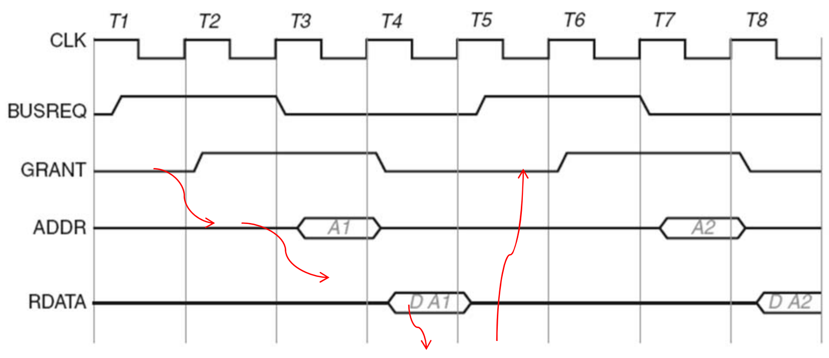
- **TDMA**
  - Time division multiple access
  - Assign slots to masters based on BW requirements
  - If a master does not have anything to read/write during its time slots, leads to low performance
  - Choice of time slot length and number critical
  - Real-time worst-case latency guarantees (CAN bus)
- **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 (3)

- **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

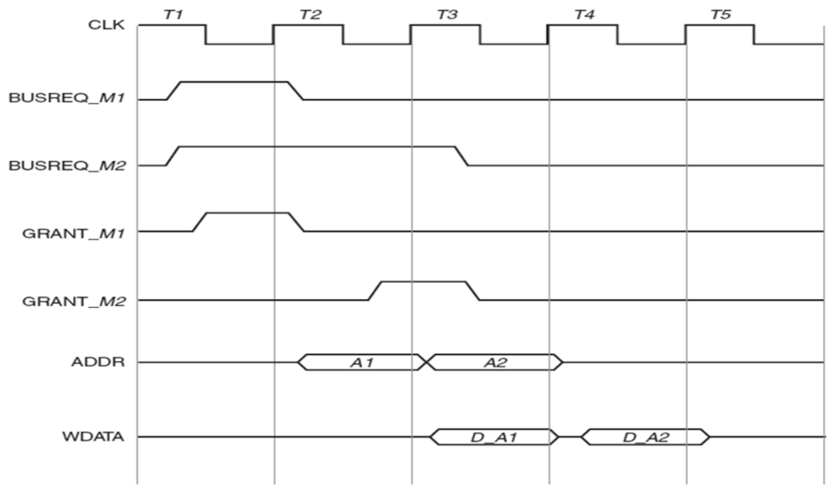
Bus Data Transfer Modes (1)

- **Single non-pipelined transfer**
  - 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



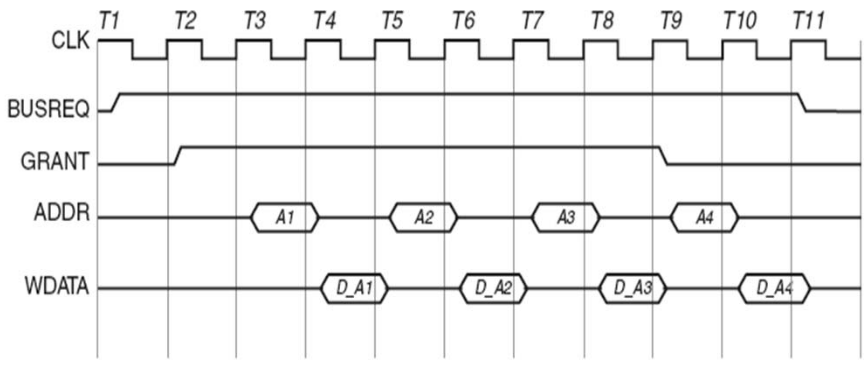
Bus Data Transfer Modes (2)

- **Pipelined transfer**
  - Overlap address and data phases
    - – Only works if separate address and data busses are present



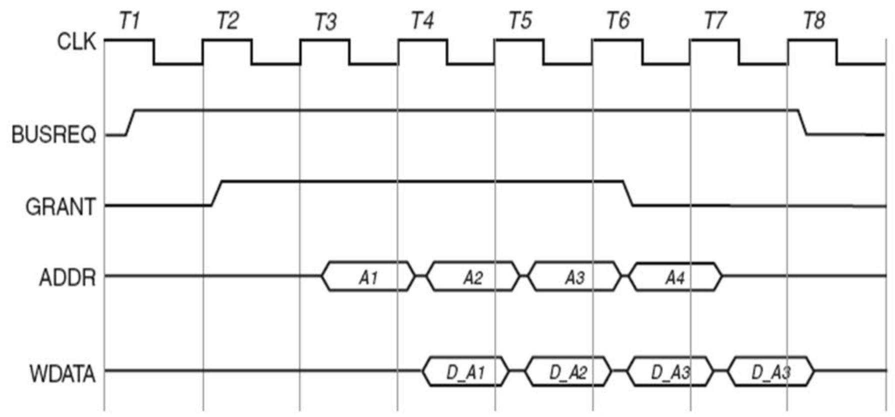
Bus Data Transfer Modes (3)

- **Non-pipelined burst transfer**
  - Send multiple data items, with only a single arbitration for entire transaction
    - – master must indicate to arbiter it intends to perform burst transfer
    - – Saves time spent requesting for arbitration



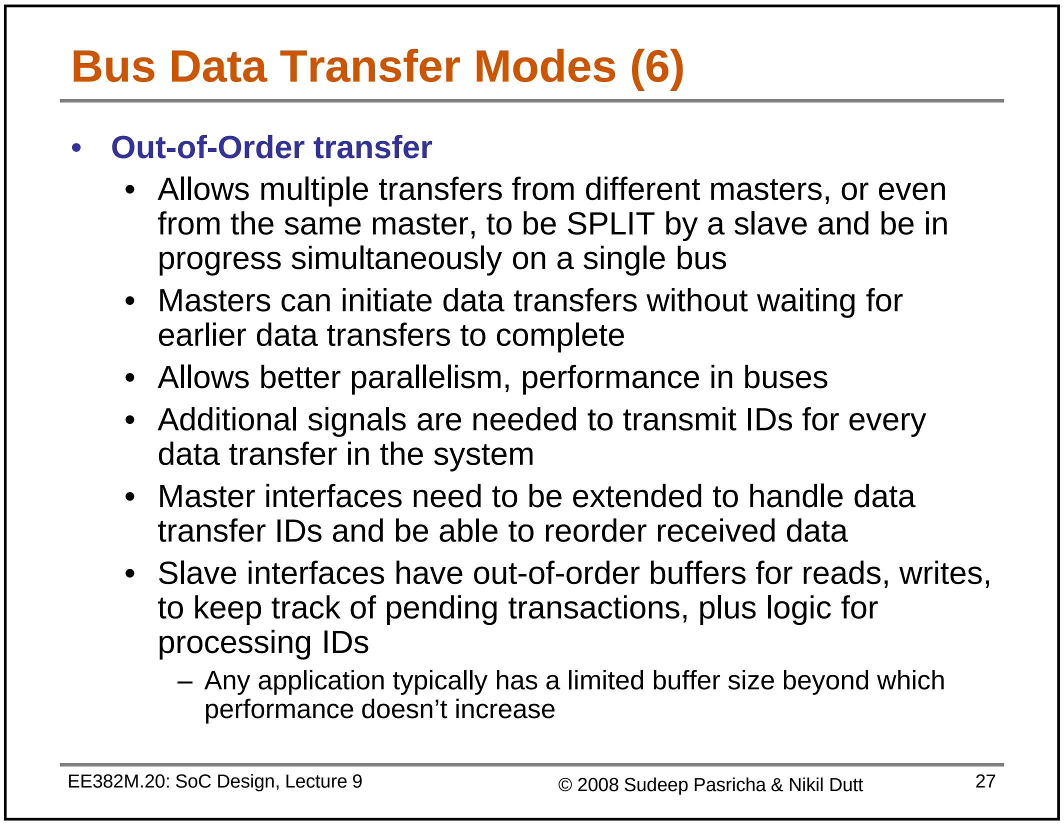
Bus Data Transfer Modes (4)

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



Bus Data Transfer Modes (5)

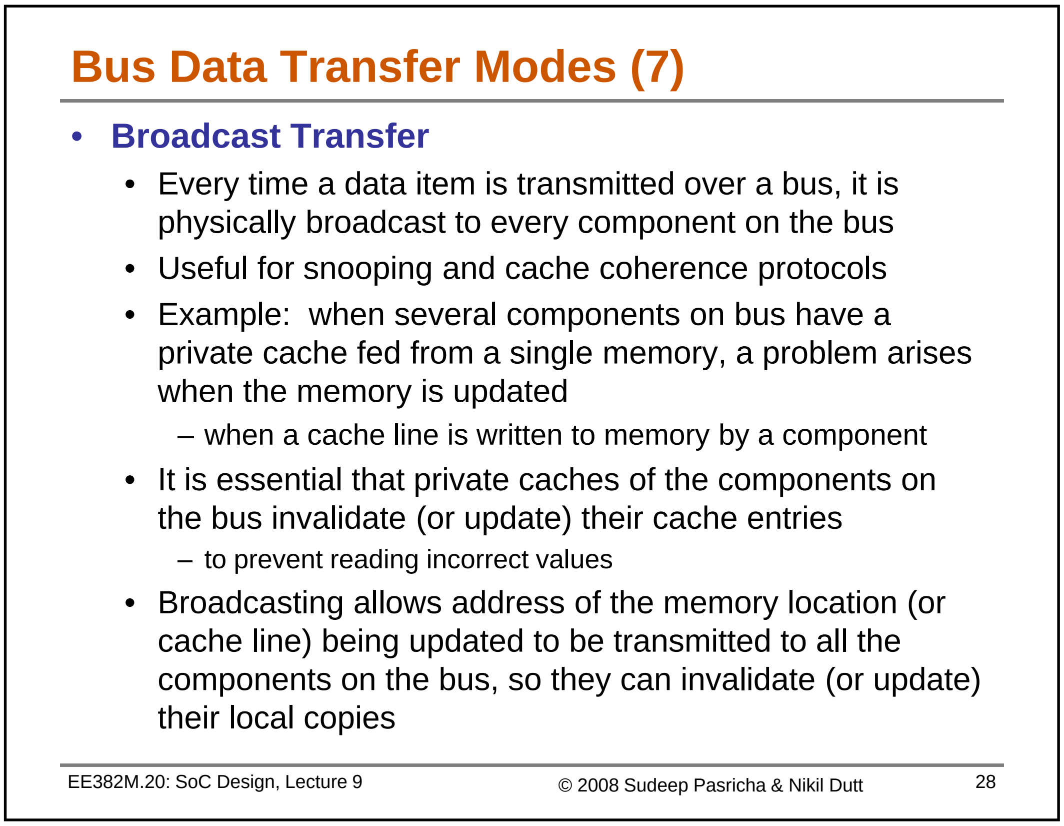
- **Split transfer**
  - If slaves take a long time to read/write data, it can prevent other masters from using the bus
  - Split transfers improve performance by 'splitting' a transaction
    - – Master sends read request to slave
    - – Slave relinquishes control of bus as it prepares data
      - » 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 (6)

- **Out-of-Order transfer**
  - Allows multiple transfers from different masters, or even from the same master, to be SPLIT by a slave and be in progress simultaneously on a single bus
  - Masters can initiate data transfers without waiting for earlier data transfers to complete
  - Allows better parallelism, performance in buses
  - Additional signals are 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

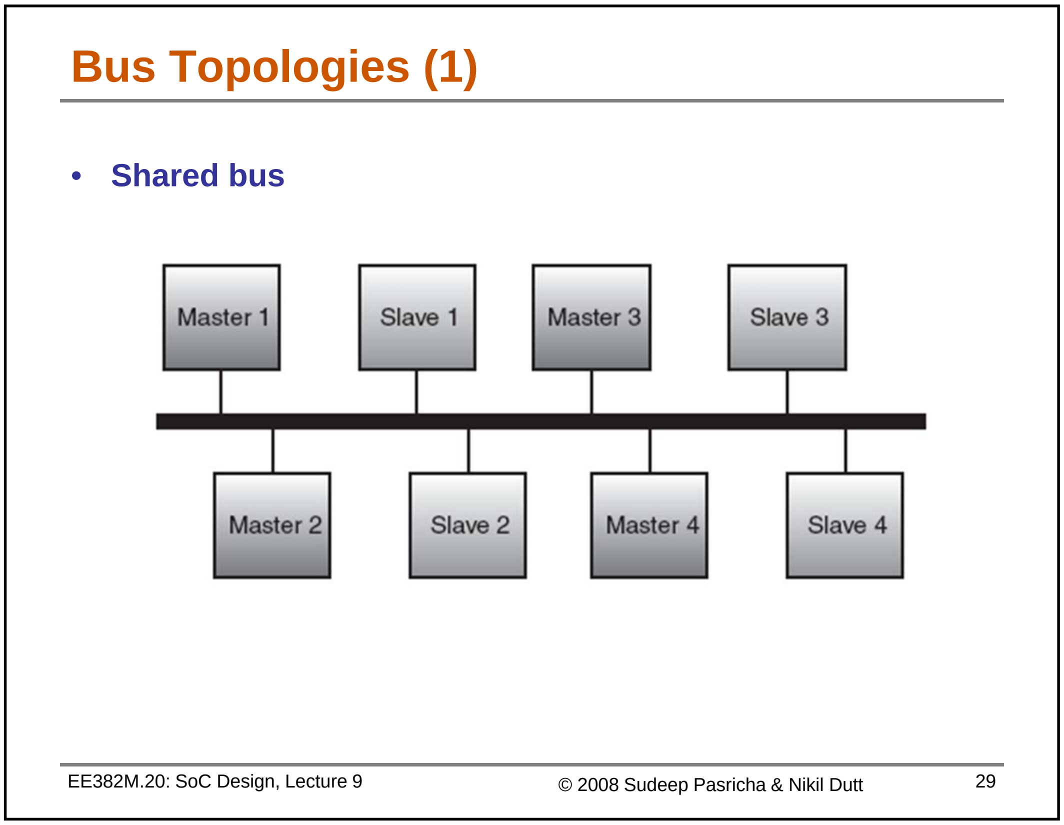
EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 27



Bus Data Transfer Modes (7)

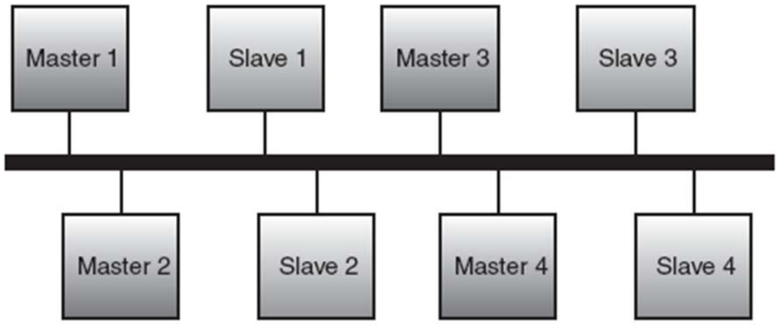
- **Broadcast Transfer**
  - Every time a data item is transmitted over a bus, it is physically broadcast to every component on the bus
  - Useful for snooping and cache coherence protocols
  - Example: when several components on bus have a private cache fed from a single memory, a problem arises when 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

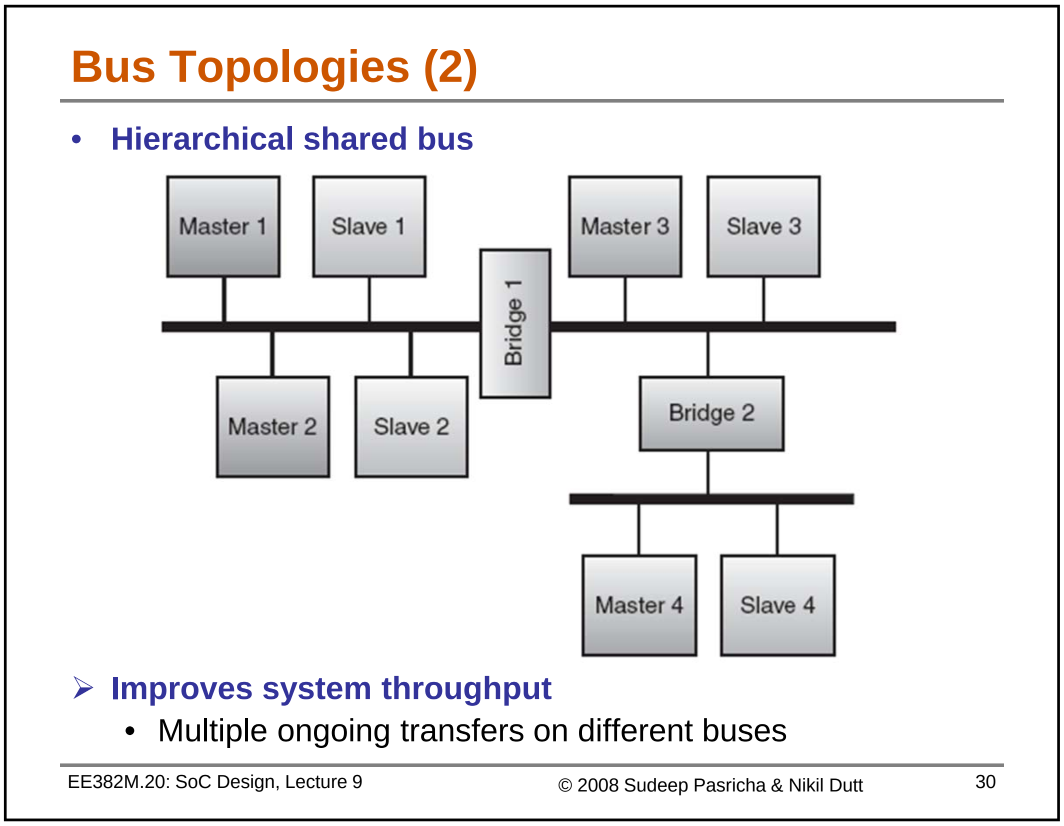
EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 28



Bus Topologies (1)

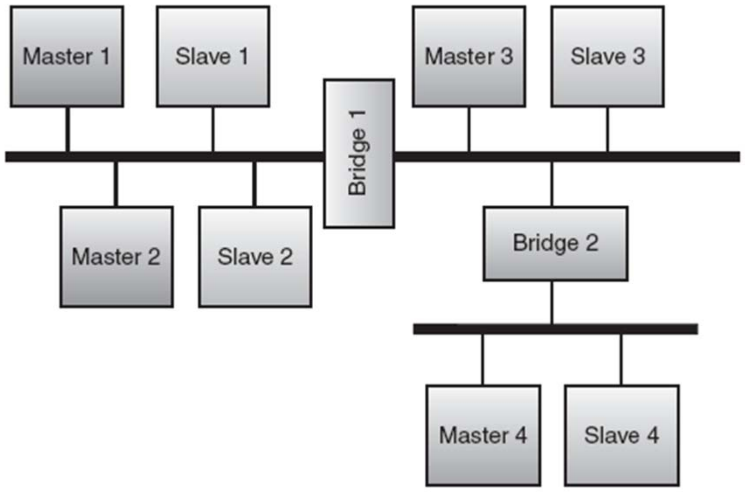
- **Shared bus**





Bus Topologies (2)

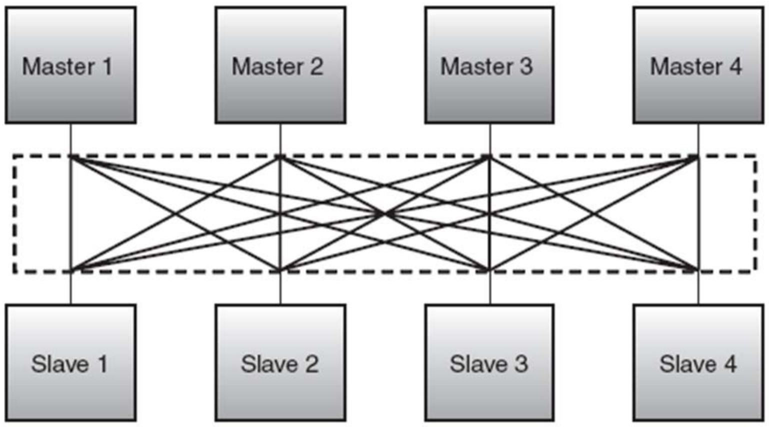
- **Hierarchical shared bus**



➢ **Improves system throughput**
  - Multiple ongoing transfers on different buses

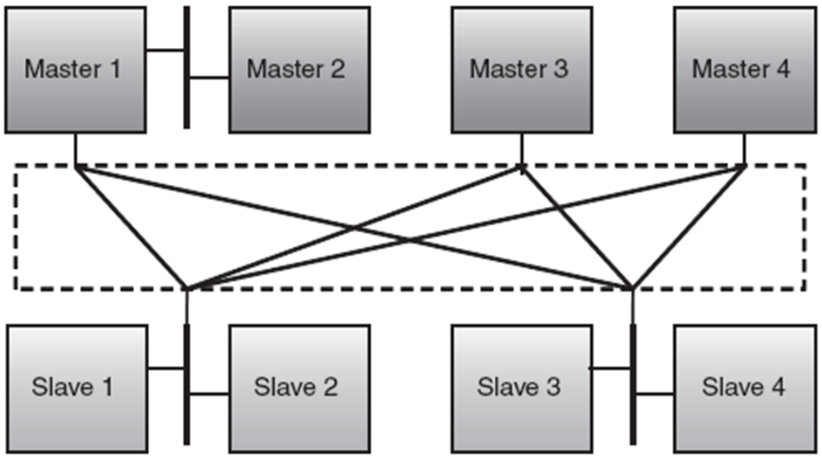
Bus Topologies (3)

- **Full crossbar/matrix bus (point to point)**



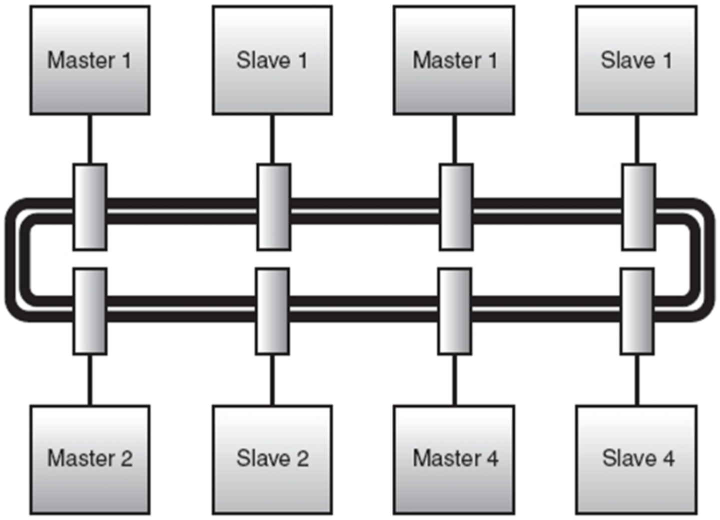
Bus Topologies (4)

- **Partial crossbar/matrix bus**



Bus Topologies (5)

- **Ring bus**



Lecture 9: Outline

✓ **Introduction**
  - ✓ Communication-centric design

✓ **Bus-based architectures**
  - ✓ Topologies and structures
  - ✓ Decoding, arbitration, transfer modes

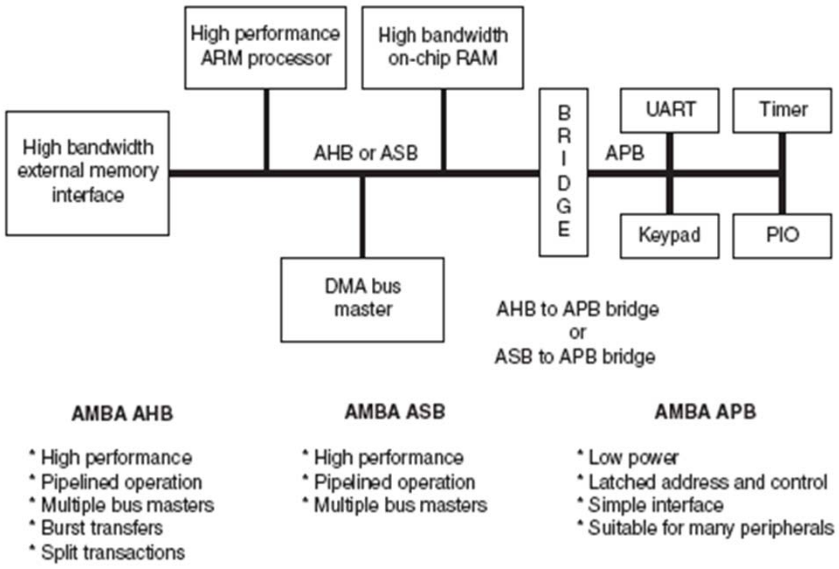
- **On-chip communication standards**
  - AMBA and AXI
- **Networks-on-Chip (NoCs)**
  - Topologies, switching, routing

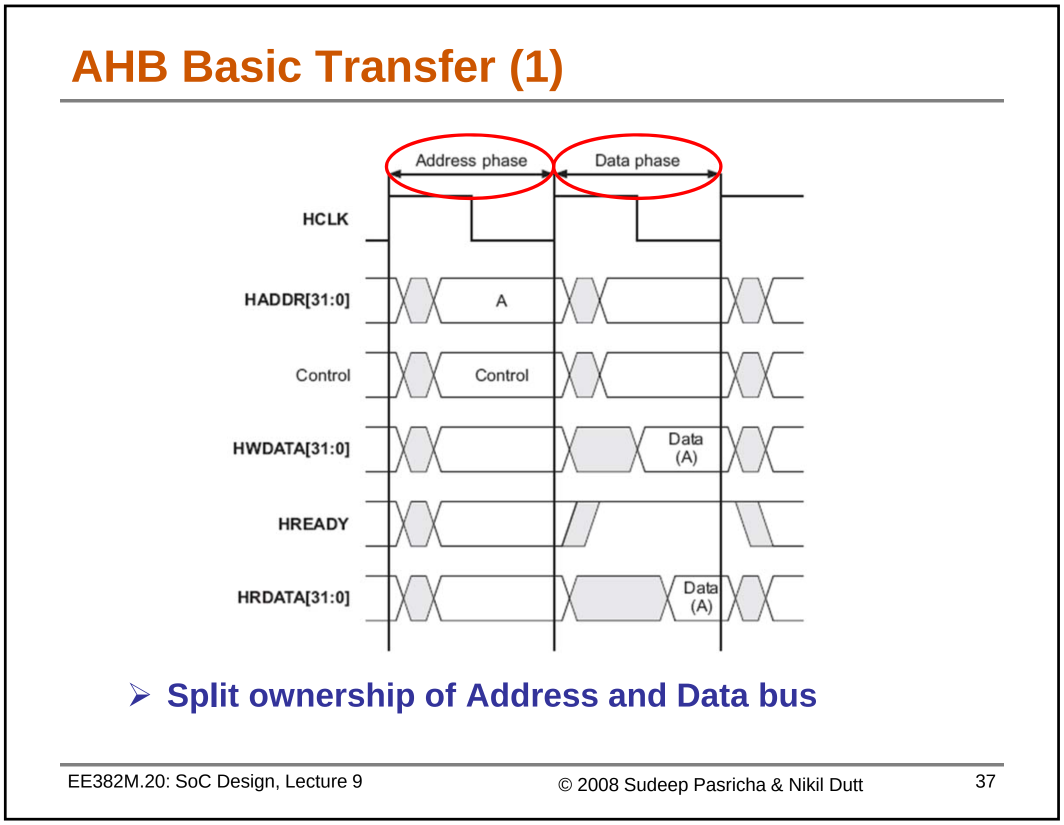
Standard Bus Architectures

- AMBA 2.0, 3.0 (ARM)
- CoreConnect (IBM)
- Sonics Smart Interconnect (Sonics)
- STBus (STMicroelectronics)
- Wishbone (Opencores)
- Avalon (Altera)
- PI Bus (OMI)
- MARBLE (Univ. of Manchester)
- CoreFrame (PalmChip)
- …

$\left.\begin{array}{l}\text{AMBA}\\\text{CoreConnect}\\\text{Sonics}\\\text{STBus}\\\text{Wishbone}\end{array}\right\}$ widely used

AMBA 2.0

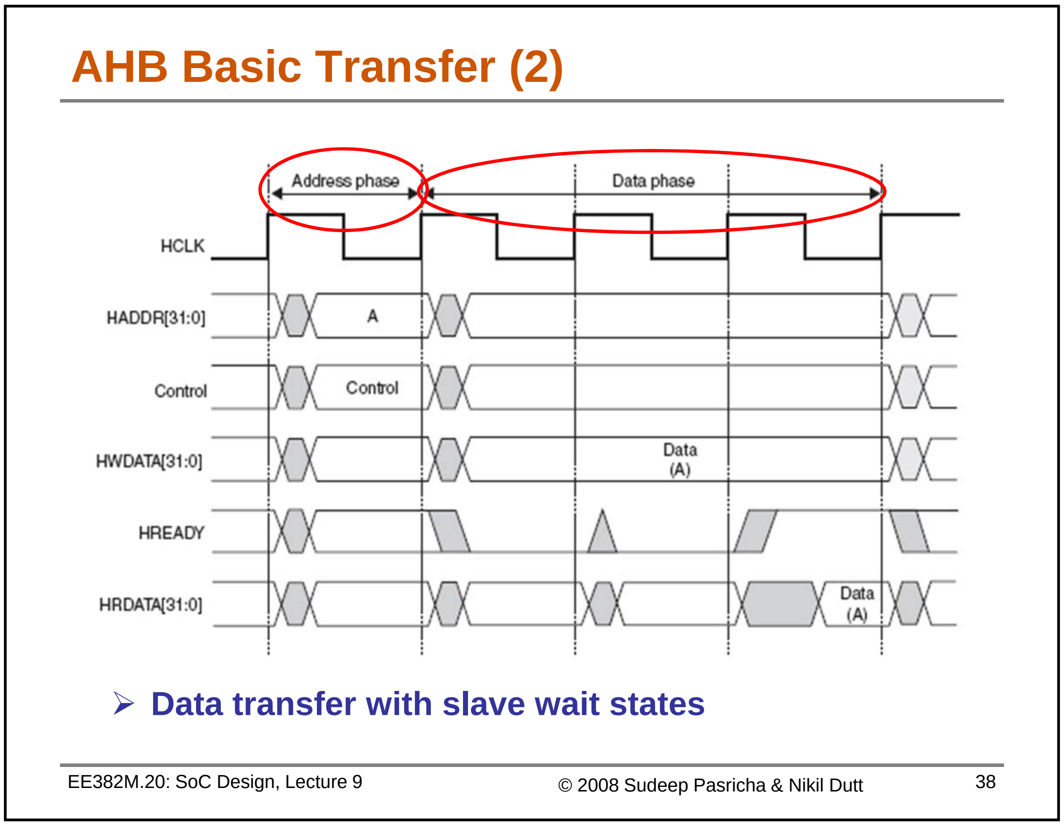




AHB Basic Transfer (1)

- Split ownership of Address and Data bus

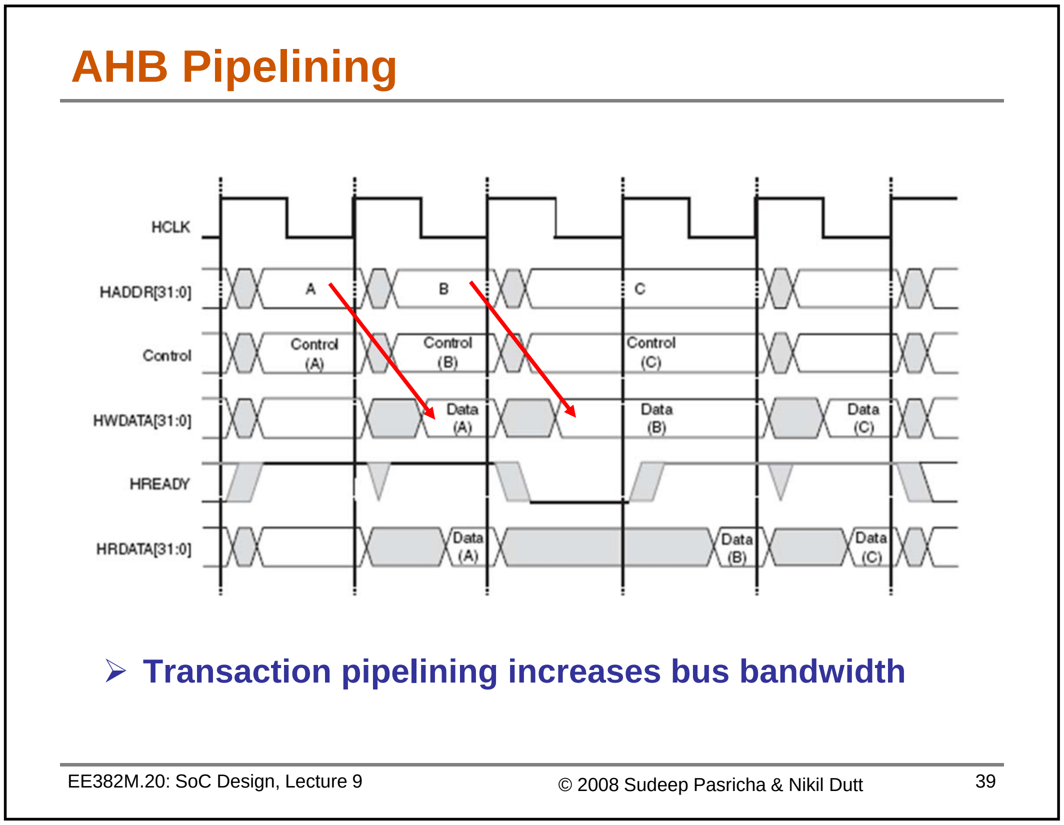
EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 37



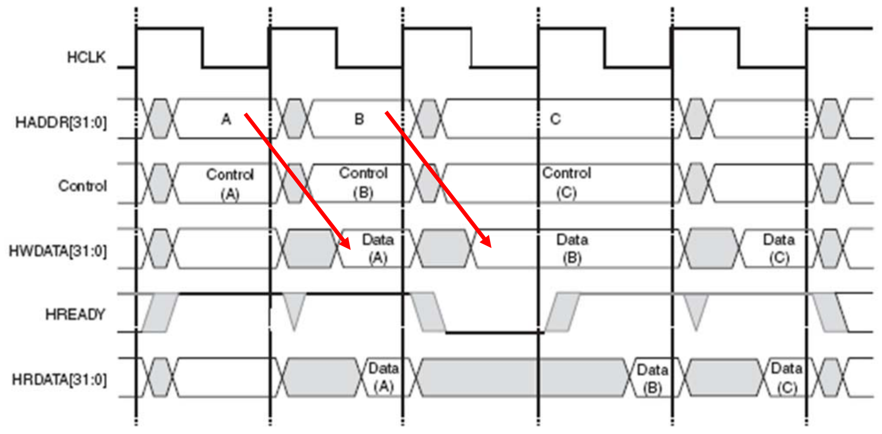
AHB Basic Transfer (2)

- Data transfer with slave wait states

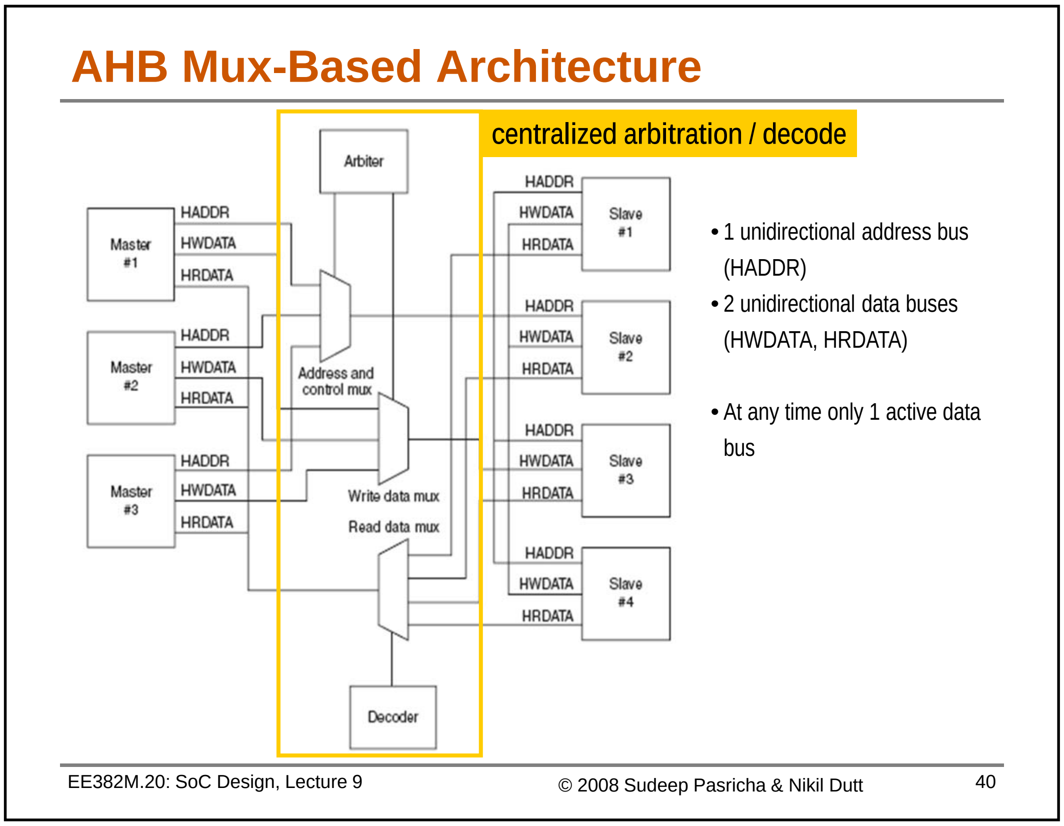
EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 38



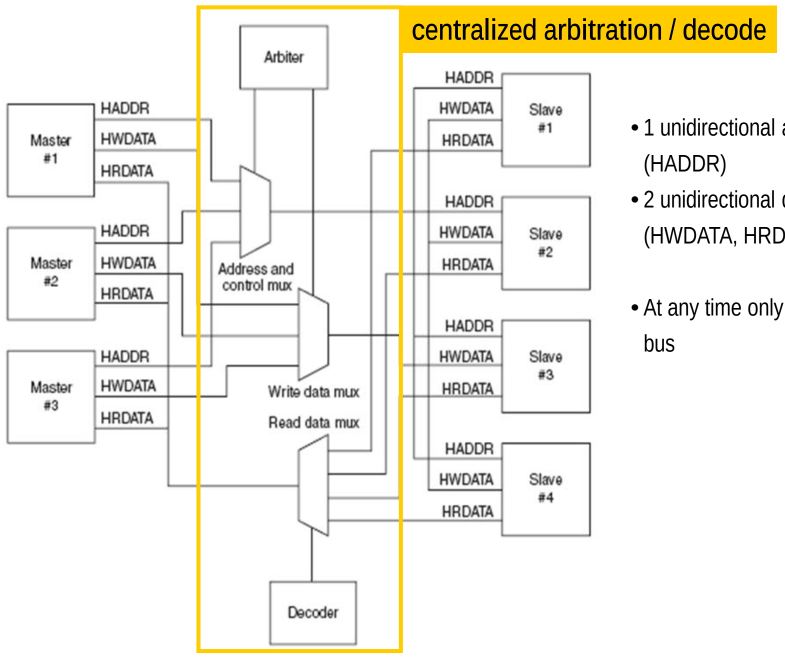
AHB Pipelining



- ➢ **Transaction pipelining increases bus bandwidth**

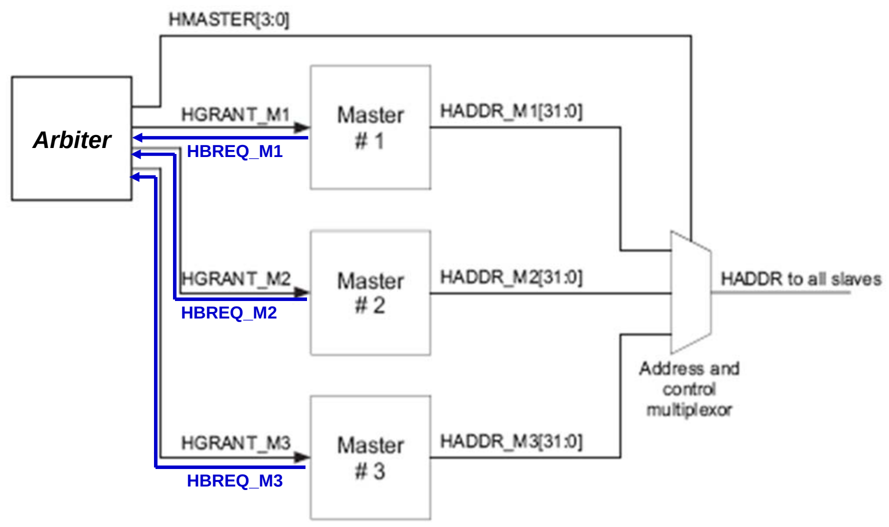


AHB Mux-Based Architecture



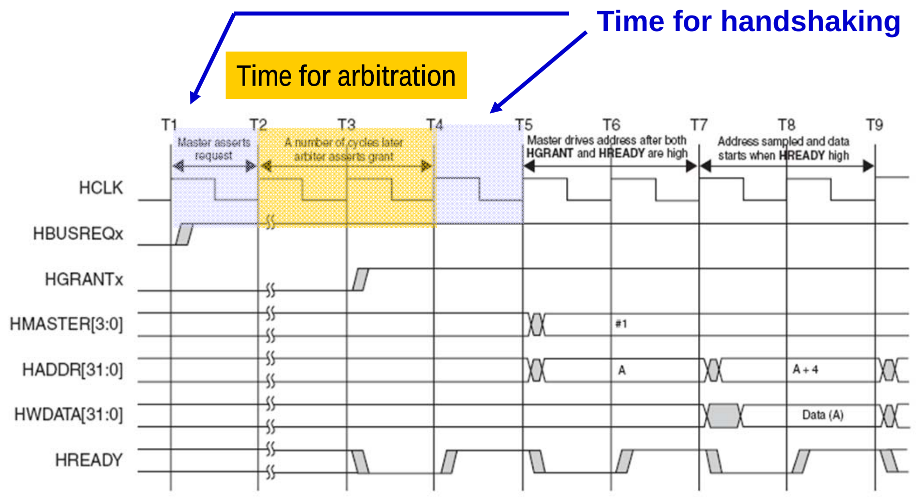
- 1 unidirectional address bus (HADDR)
- 2 unidirectional data buses (HWDATA, HRDATA)
- At any time only 1 active data bus

AHB Arbitration

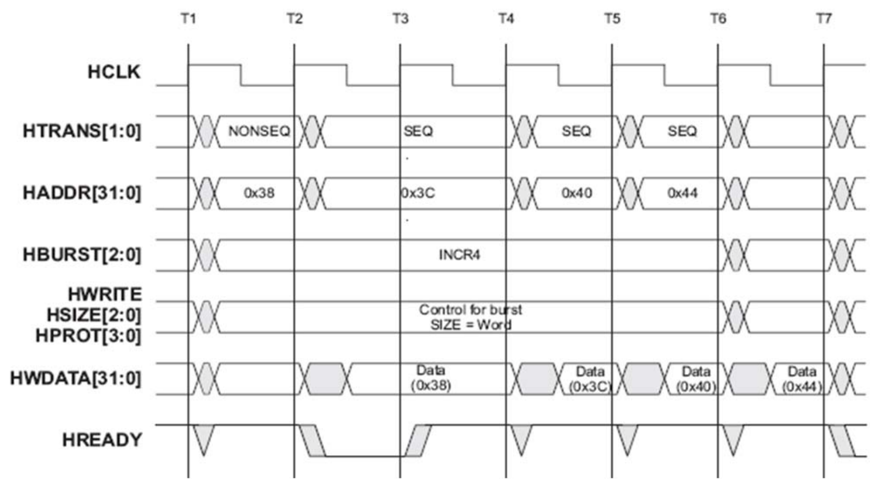


- **Arbitration protocol is specified, but not the policy**

AHB Arbitration Timing



AHB Pipelined Burst Transfers



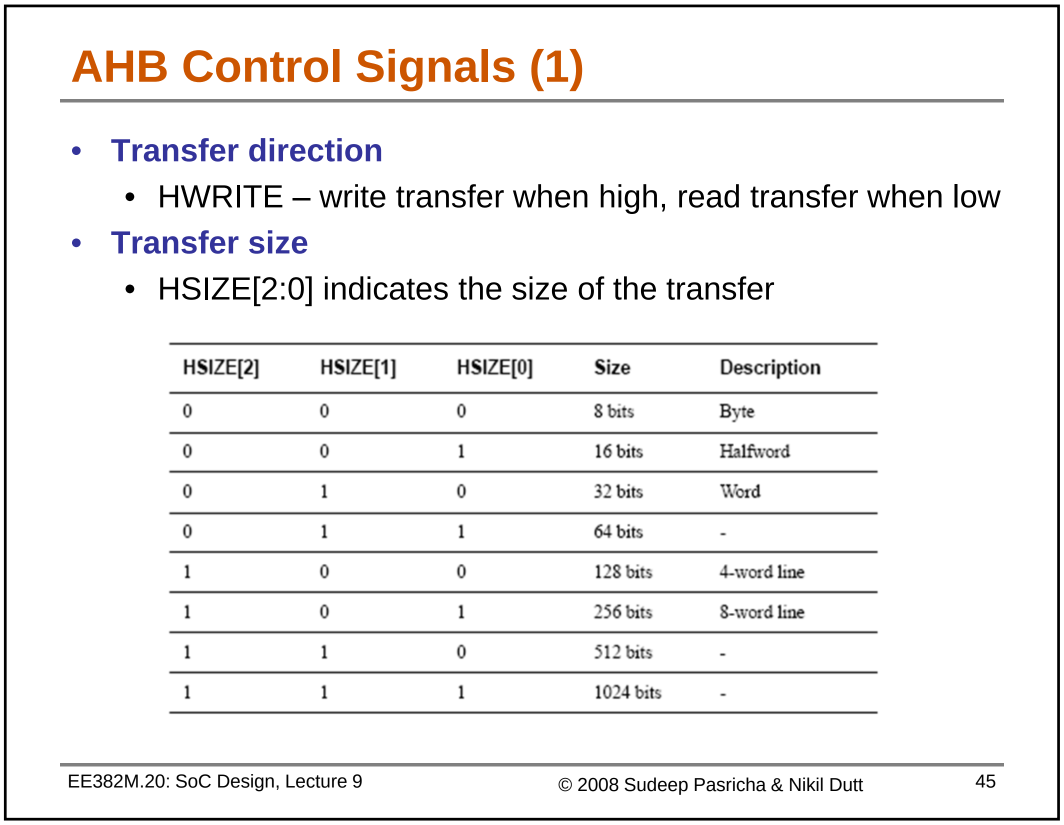
- **Bursts cut down on arbitration, handshaking time**
  - Improving performance

AHB Burst Types

HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

Fixed length bursts

- **Incremental bursts access sequential locations**
  - e.g. 0x64, 0x68, 0x6C, 0x70 for INCR4, transferring 4 byte data
- **Wrapping bursts "wrap around" address if starting address is not aligned to total no. of bytes in transfer**
  - e.g. 0x64, 0x68, 0x6C, 0x60 for WRAP4, transferring 4 byte data

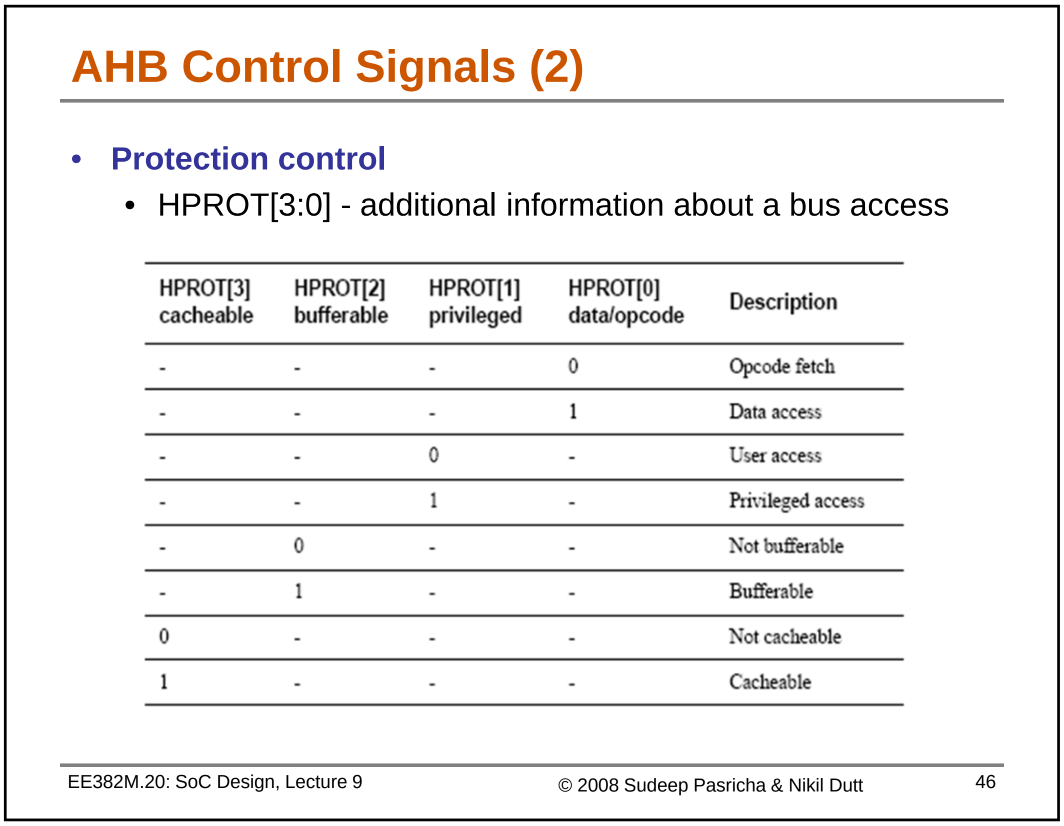


AHB Control Signals (1)

- **Transfer direction**
  - HWRITE – write transfer when high, read transfer when low
- **Transfer size**
  - HSIZE[2:0] indicates the size of the transfer

HSIZE[2]	HSIZE[1]	HSIZE[0]	Size	Description
0	0	0	8 bits	Byte
0	0	1	16 bits	Halfword
0	1	0	32 bits	Word
0	1	1	64 bits	-
1	0	0	128 bits	4-word line
1	0	1	256 bits	8-word line
1	1	0	512 bits	-
1	1	1	1024 bits	-

EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 45



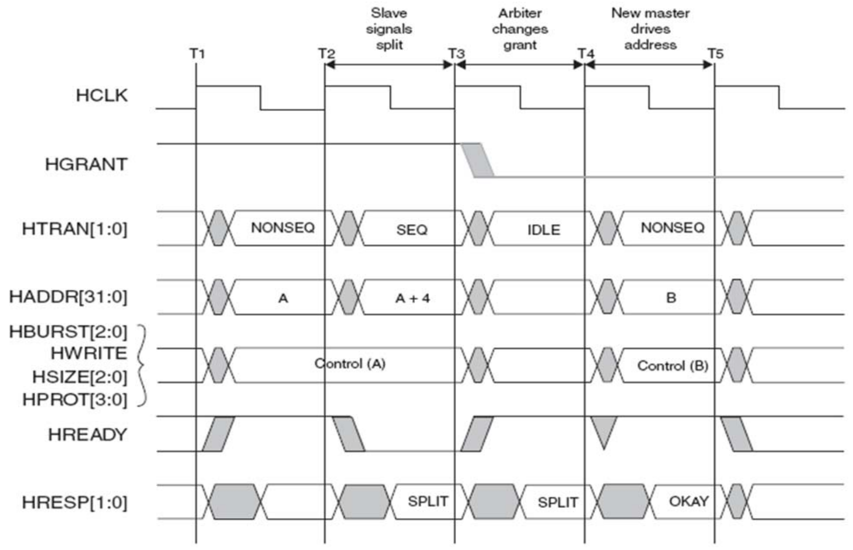
AHB Control Signals (2)

- **Protection control**
  - HPROT[3:0] - additional information about a bus access

HPROT[3] cacheable	HPROT[2] bufferable	HPROT[1] privileged	HPROT[0] data/opcode	Description
-	-	-	0	Opcode fetch
-	-	-	1	Data access
-	-	0	-	User access
-	-	1	-	Privileged access
-	0	-	-	Not bufferable
-	1	-	-	Bufferable
0	-	-	-	Not cacheable
1	-	-	-	Cacheable

EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 46

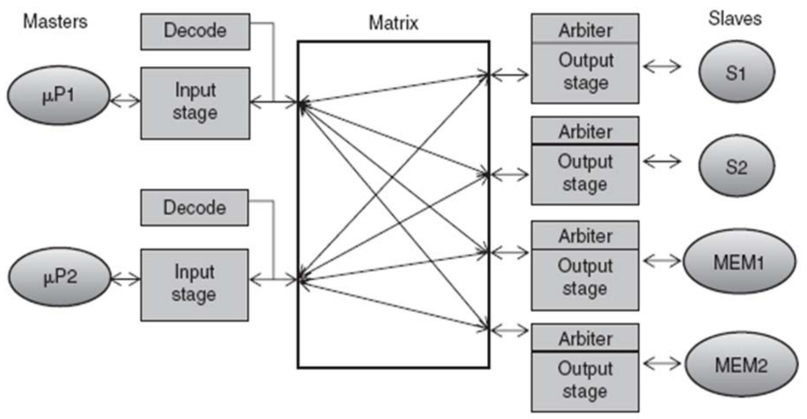
AHB Split Transfers



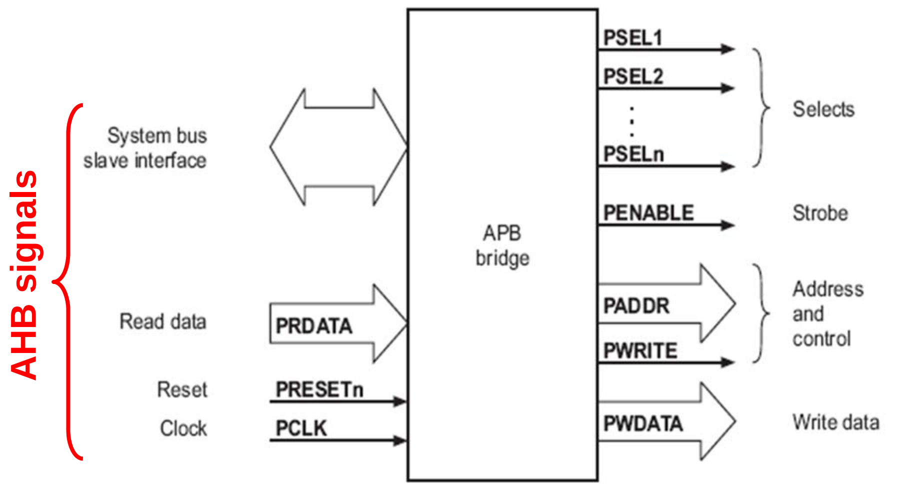
- Improves bus utilization
- May cause deadlocks if not carefully implemented

AHB Bus Matrix Topology

- In addition to shared bus and hierarchical bus, AHB can be implemented as a bus matrix



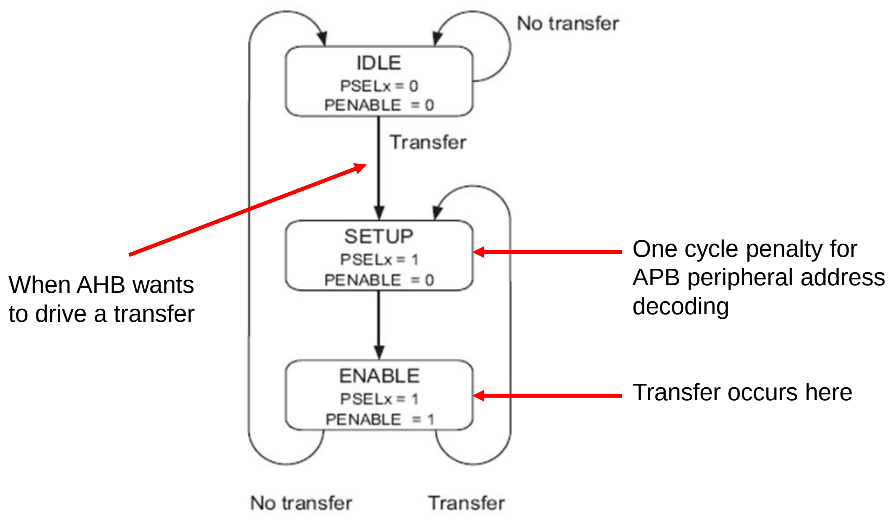
AHB-APB Bridge



High performance

Low power
(and performance)

APB State Diagram



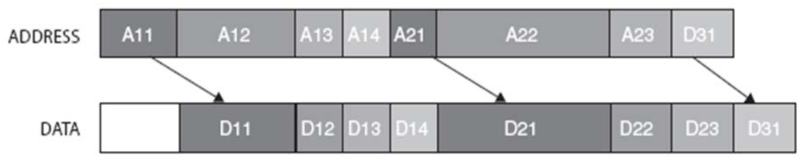
- No (multi-cycle) bursts, pipelined transfers

AMBA 3.0

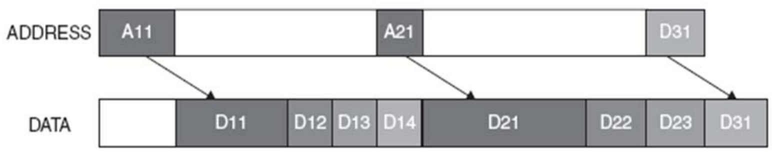
- **Introduces AXI high performance protocol**
  - Support for separate read address, write address, read data, write data, write response channels
  - Out of order (OO) transaction completion
  - Fixed mode burst support
    - Useful for I/O peripherals
  - Advanced system cache support
    - Specify if transaction is cacheable/bufferable
    - Specify attributes such as write-back/write-through
  - Enhanced protection support
    - Secure/non-secure transaction specification
  - Exclusive access (for semaphore operations)
  - Register slice support for high frequency operation

AHB vs. AXI Burst (1)

- **AHB Burst**
  - Address and Data are locked together (single pipeline stage)
  - HREADY controls intervals of address and data

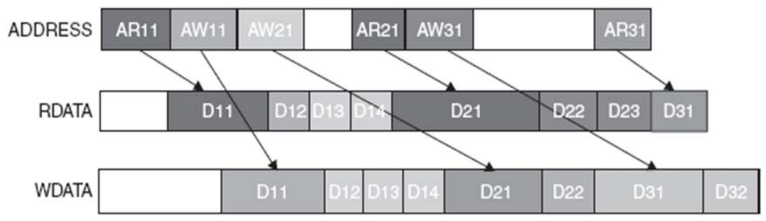


- **AXI Burst**
  - One Address for entire burst



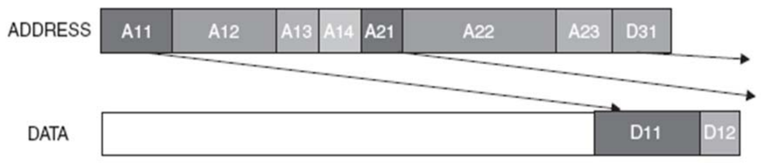
AHB vs. AXI Burst (2)

- **AXI Burst**
  - Simultaneous read, write transactions
  - Better bus utilization

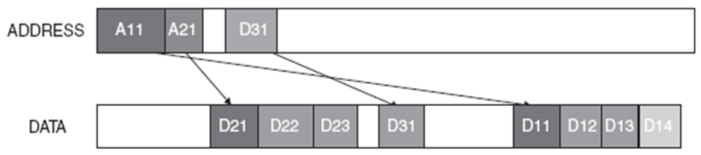


AXI Out of Order Completion

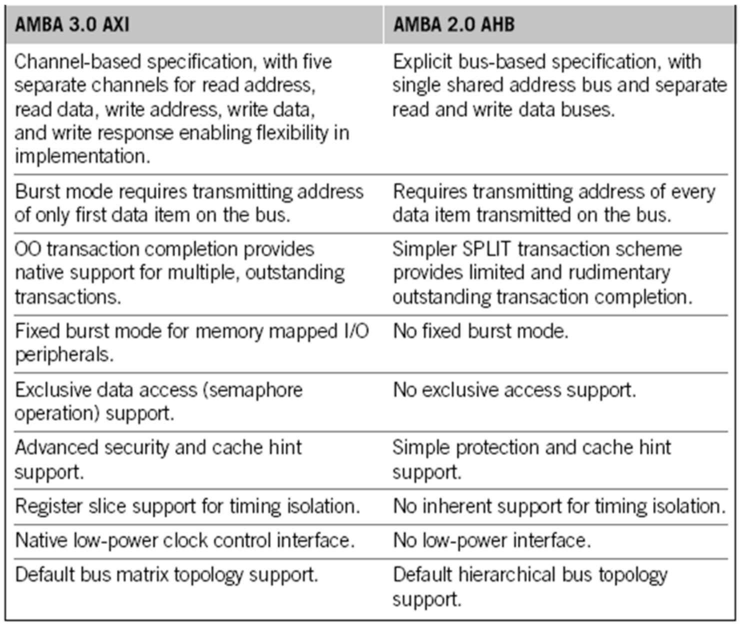
- **With AHB**
  - If one slave is very slow, all data is held up
  - SPLIT transactions provide very limited improvement



- **With AXI Burst**
  - Multiple outstanding addresses
    - Out of order (OO) completion allowed
  - Fast slaves may return data ahead of slow slaves

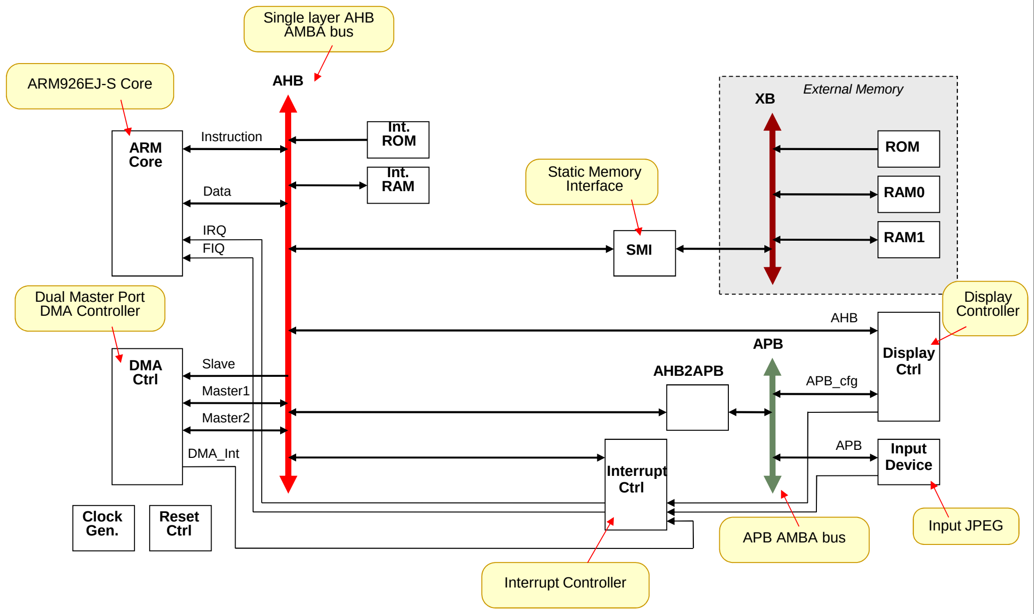


Summary: AHB vs. AXI

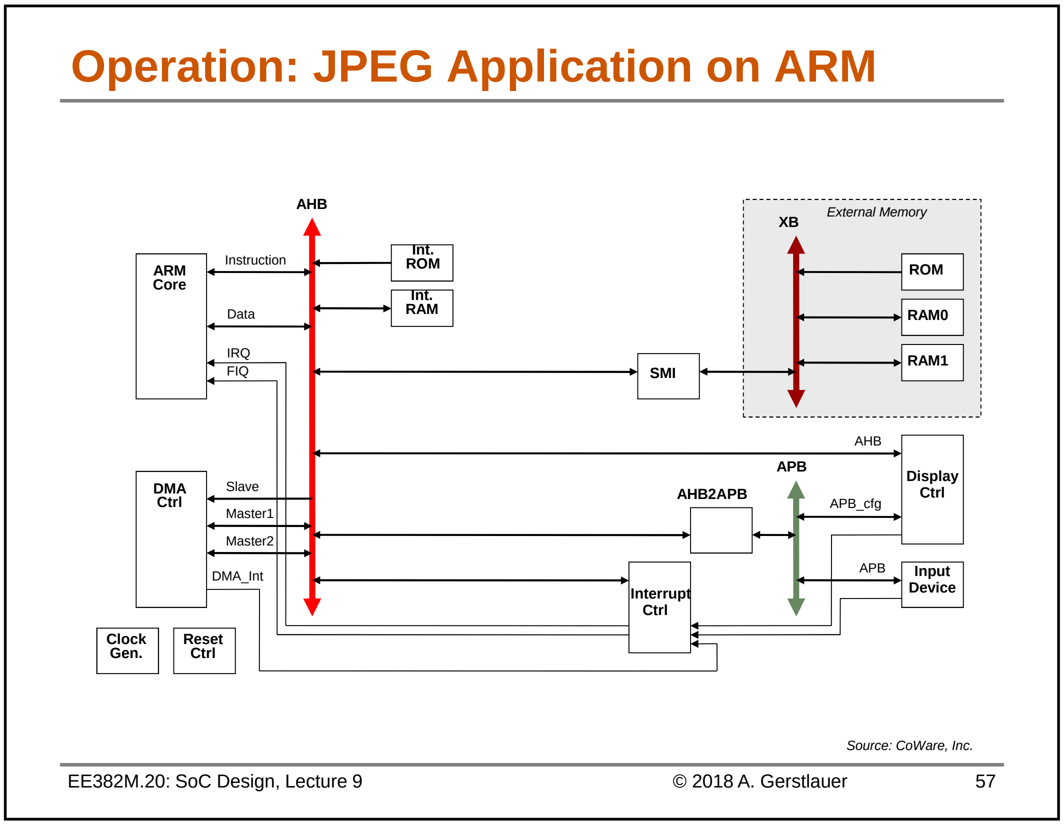


AMBA 3.0 AXI	AMBA 2.0 AHB
Channel-based specification, with five separate channels for read address, read data, write address, write data, and write response enabling flexibility in implementation.	Explicit bus-based specification, with single shared address bus and separate read and write data buses.
Burst mode requires transmitting address of only first data item on the bus.	Requires transmitting address of every data item transmitted on the bus.
OO transaction completion provides native support for multiple, outstanding transactions.	Simpler SPLIT transaction scheme provides limited and rudimentary outstanding transaction completion.
Fixed burst mode for memory mapped I/O peripherals.	No fixed burst mode.
Exclusive data access (semaphore operation) support.	No exclusive access support.
Advanced security and cache hint support.	Simple protection and cache hint support.
Register slice support for timing isolation.	No inherent support for timing isolation.
Native low-power clock control interface.	No low-power interface.
Default bus matrix topology support.	Default hierarchical bus topology support.

JPEG Decoder Case Study



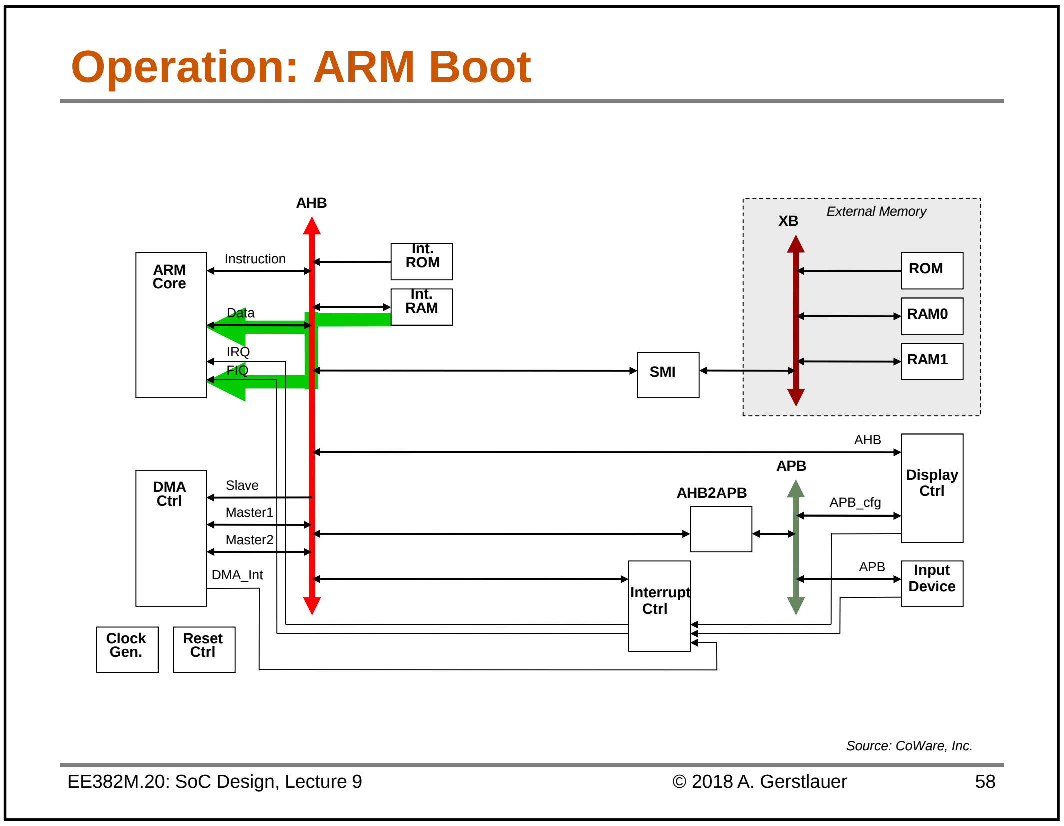
Source: CoWare, Inc.



Operation: JPEG Application on ARM

Source: CoWare, Inc.

EE382M.20: SoC Design, Lecture 9 © 2018 A. Gerstlauer 57

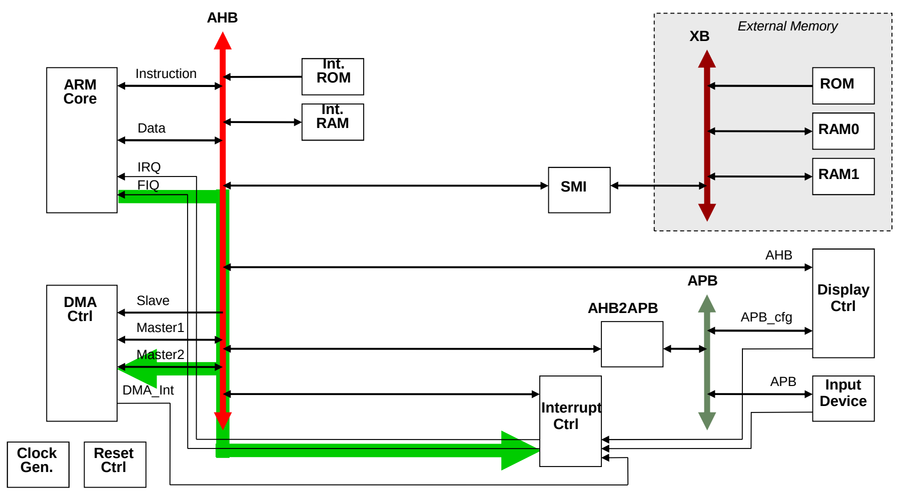


Operation: ARM Boot

Source: CoWare, Inc.

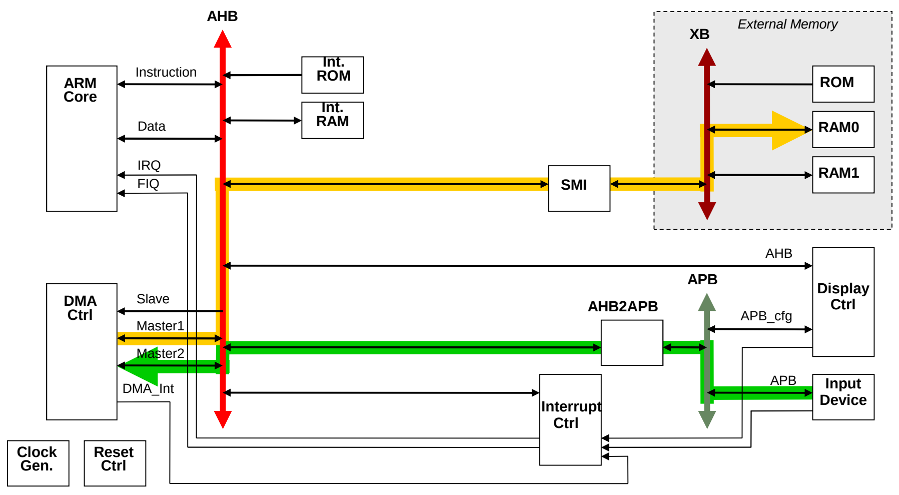
EE382M.20: SoC Design, Lecture 9 © 2018 A. Gerstlauer 58

Operation: DMA and IC Initialization



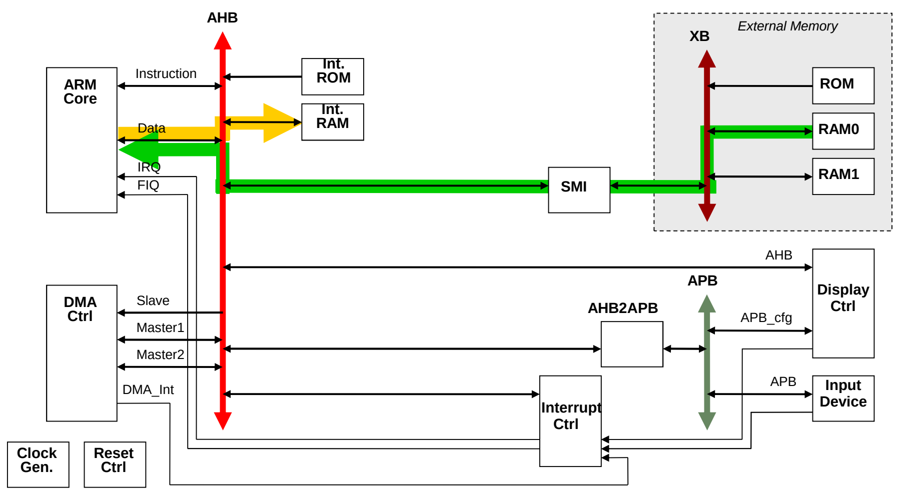
Source: CoWare, Inc.

Operation: DMA Image Transfer



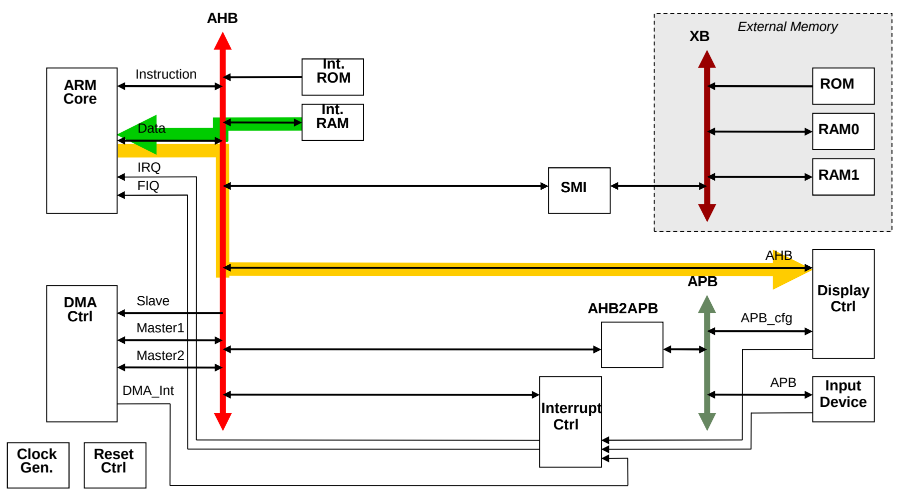
Source: CoWare, Inc.

Operation: Huffman Decoding



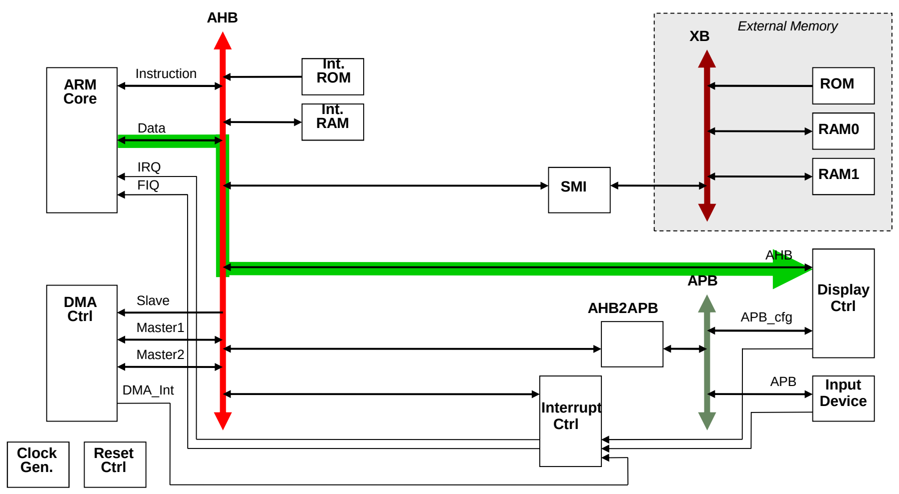
Source: CoWare, Inc.

Operation: IDCT



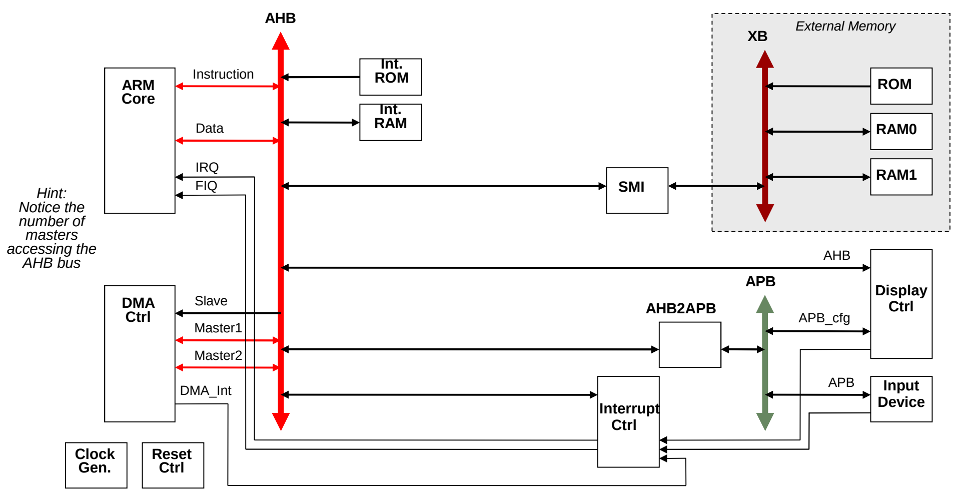
Source: CoWare, Inc.

Operation: Display



Source: CoWare, Inc.

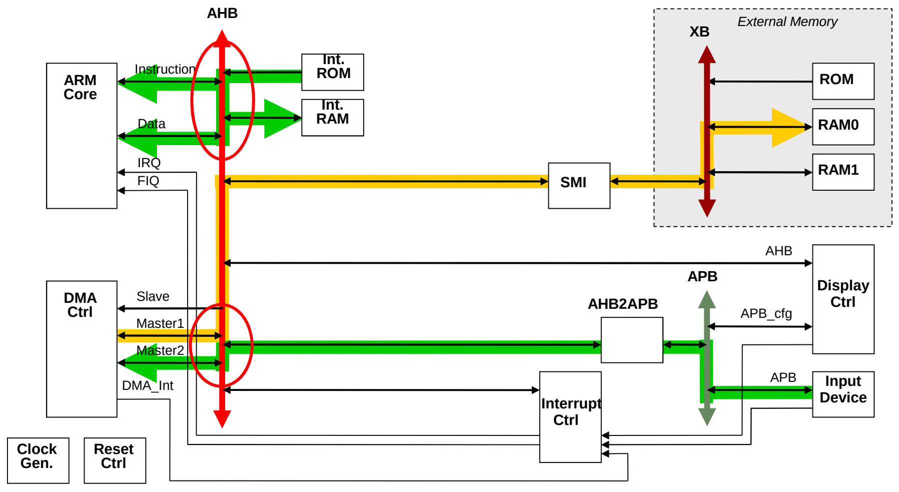
Bus Contention?



Source: CoWare, Inc.

Architecture 1

- **Contention and utilization problems due to**
  - ARM core and dual DMA activity

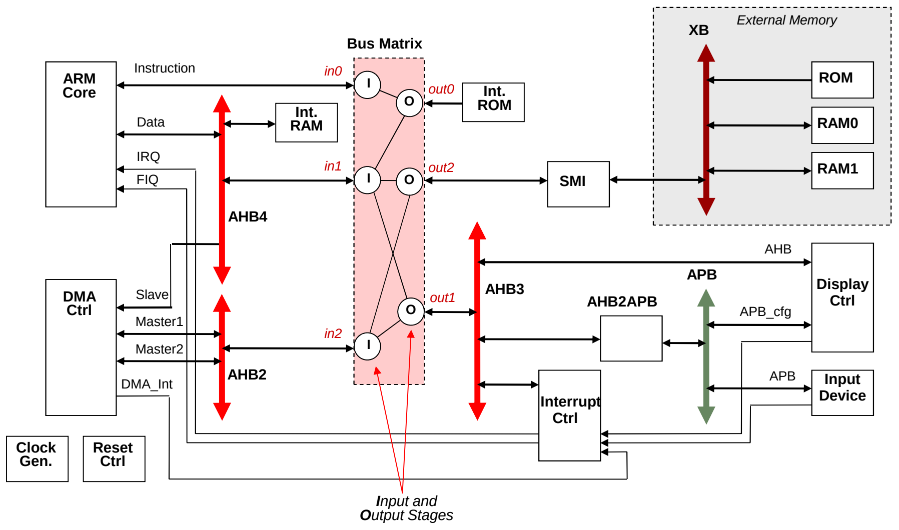


Source: CoWare, Inc.

EE382M.20: SoC Design, Lecture 9 © 2018 A. Gerstlauer 65

Architecture 2

- **Multi-layer architecture**
  - Multiple AHB busses

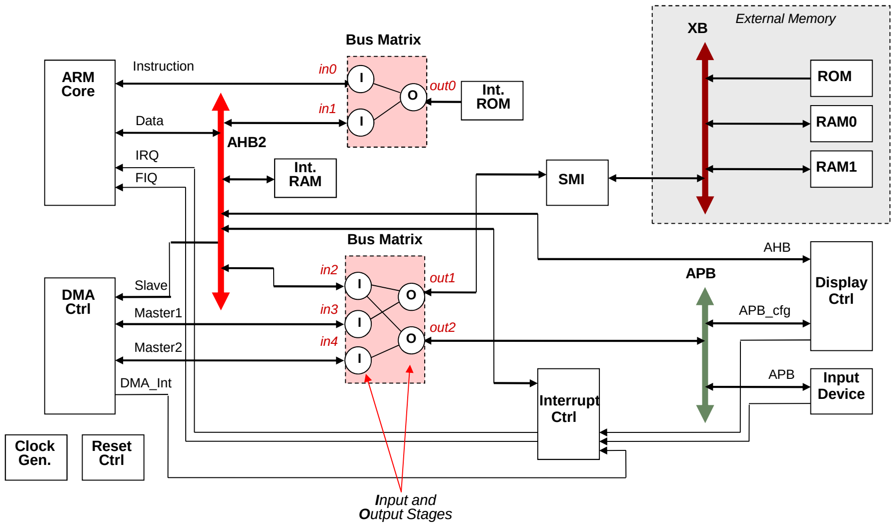


Source: CoWare, Inc.

EE382M.20: SoC Design, Lecture 9 © 2018 A. Gerstlauer 66

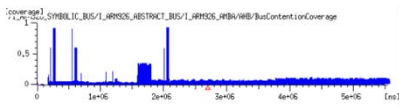
Architecture 3

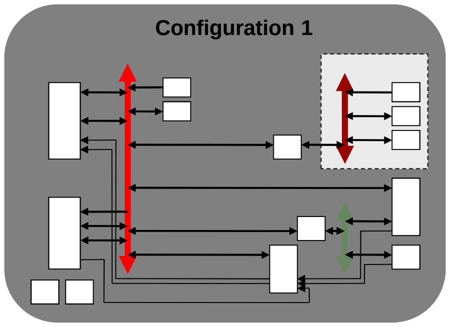
- **Dual multi-layer architecture**
  - Single AHB bus

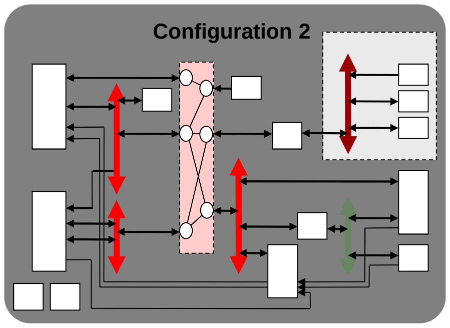


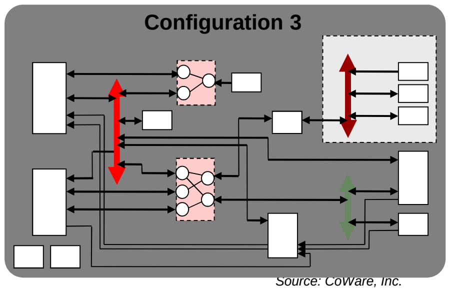
Source: CoWare, Inc.

Minimal Bus Contention?



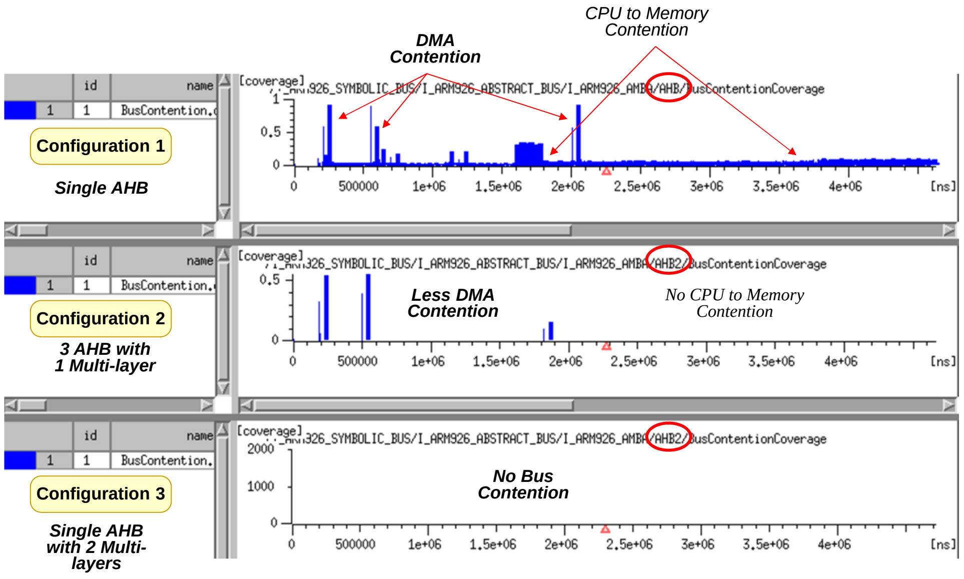






Source: CoWare, Inc.

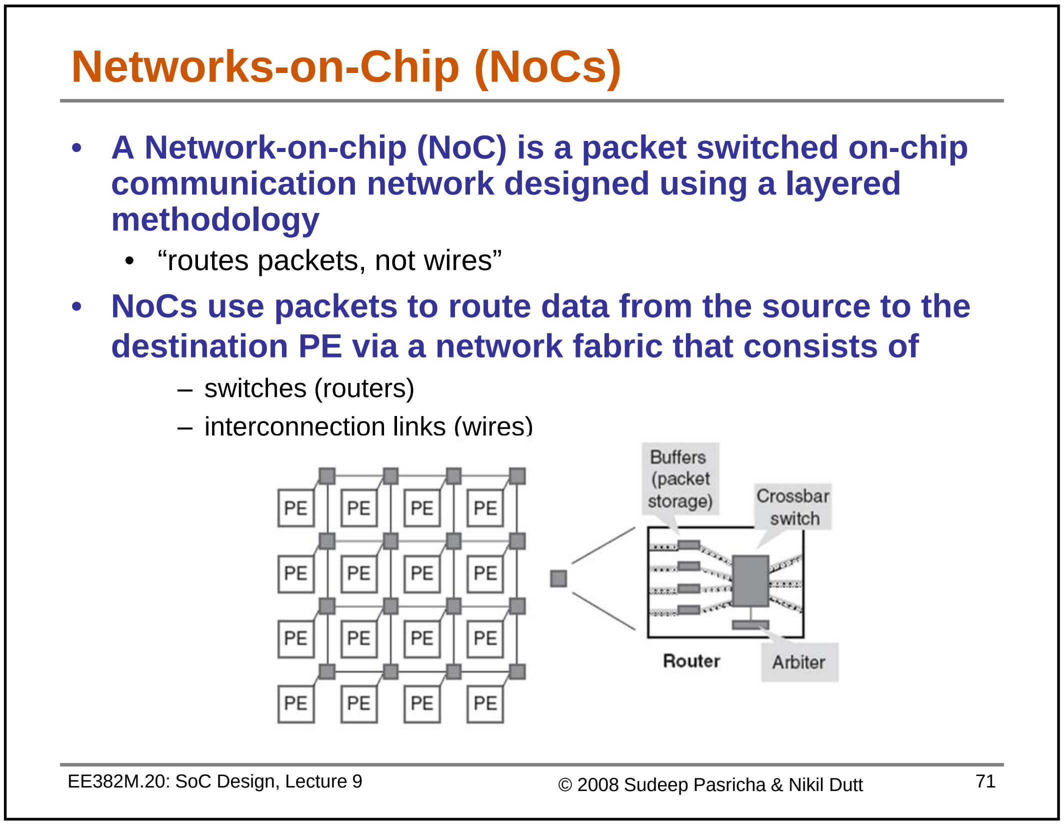
TLM Simulation Results



Source: CoWare, Inc.

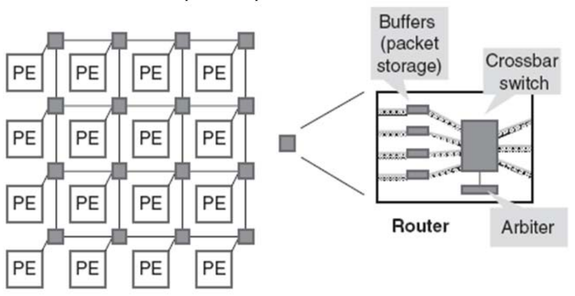
Lecture 9: Outline

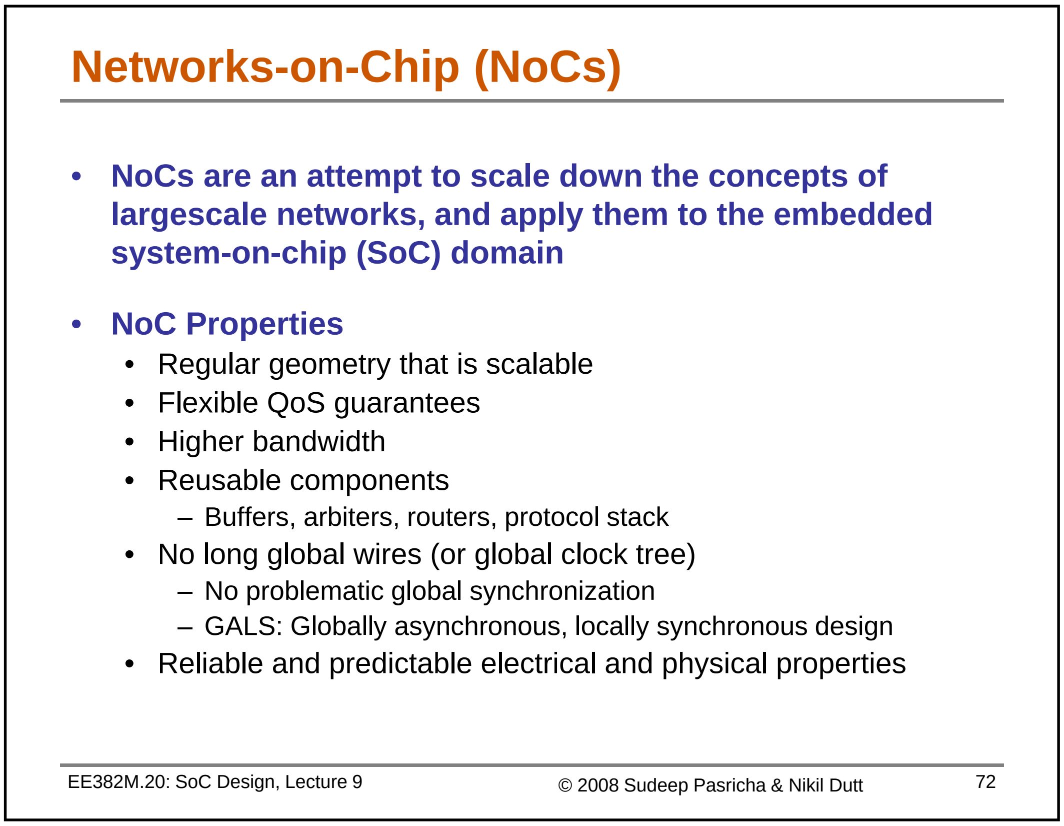
- ✓ Introduction
  - ✓ Communication-centric design
- ✓ Bus-based architectures
  - ✓ Topologies and structures
  - ✓ Decoding, arbitration, transfer modes
- ✓ On-chip communication standards
  - ✓ AMBA and AXI
- **Networks-on-Chip (NoCs)**
  - Topologies, switching, routing



Networks-on-Chip (NoCs)

- **A Network-on-chip (NoC) is a packet switched on-chip communication network designed using a layered methodology**
  - "routes packets, not wires"
- **NoCs use packets to route data from the source to the destination PE via a network fabric that consists of**
  - – switches (routers)
  - – interconnection links (wires)



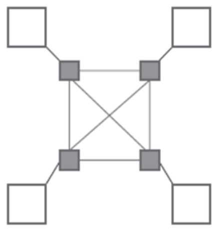


Networks-on-Chip (NoCs)

- **NoCs are an attempt to scale down the concepts of largescale networks, and apply them to the embedded system-on-chip (SoC) domain**
- **NoC Properties**
  - Regular geometry that is scalable
  - Flexible QoS guarantees
  - Higher bandwidth
  - Reusable components
    - – Buffers, arbiters, routers, protocol stack
  - No long global wires (or global clock tree)
    - – No problematic global synchronization
    - – GALS: Globally asynchronous, locally synchronous design
  - Reliable and predictable electrical and physical properties

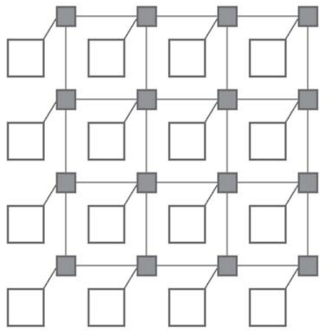
NoC Topology (1)

- **Direct Topologies**
  - Each node has direct point-to-point link to a subset of other nodes in the system called neighboring nodes
    - E.g. Nostrum, SOCBUS, Proteo, Octagon
  - Nodes consist of computational blocks and/or memories, as well as a NI block that acts as a router
  - As the number of nodes in the system increases, the total available communication bandwidth also increases
  - Fundamental trade-off is between connectivity and cost



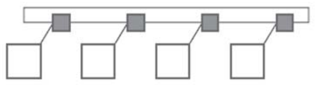
NoC Topology (2)

- **Most direct network topologies have an orthogonal implementation, where nodes can be arranged in an *n*-dimensional orthogonal space**
  - Routing for such networks is fairly simple
    - E.g. n-dimensional mesh, torus, folded torus, hypercube, and octagon
- **2D mesh is most popular topology**
  - All links have the same length
    - Eases physical design
  - Area grows linearly with the number of nodes
  - Must be designed in such a way as to avoid traffic accumulating in the center of the mesh

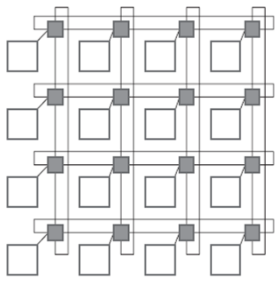


NoC Topology (3)

- **Torus topology, also called a *k*-ary *n*-cube, is an *n*-dimensional grid with *k* nodes in each dimension**
  - *k*-ary 1-cube (1-D torus) is essentially a ring network with k nodes
    - Limited scalability as performance decreases when more nodes

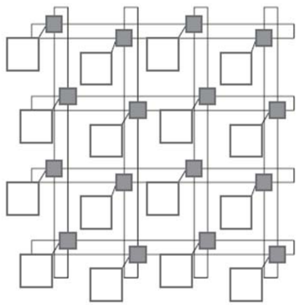


  - *k*-ary 2-cube (i.e., 2-D torus) topology is similar to a regular mesh
    - Except that nodes at the edges are connected to switches at the opposite edge via wrap-around channels
    - Long end-around connections can, however, lead to excessive delays

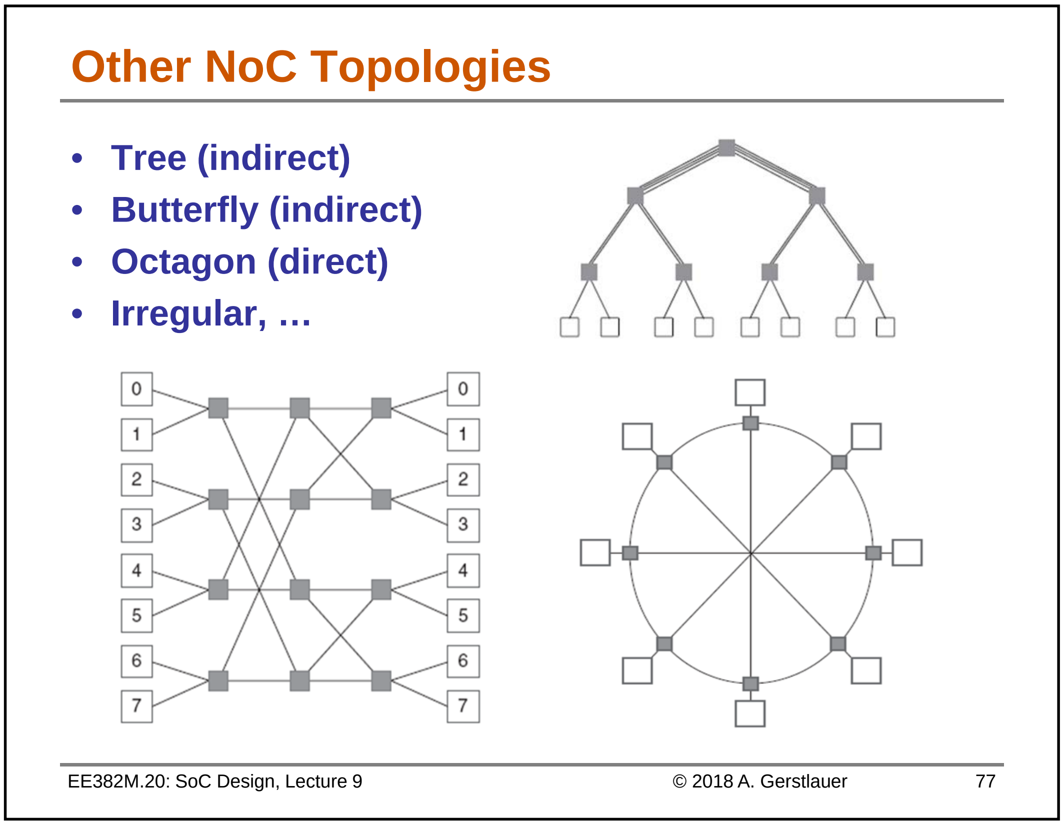


NoC Topology (4)

- **Folding torus topology overcomes the long link limitation of a 2-D torus**
  - Links have the same size

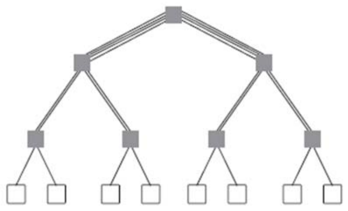


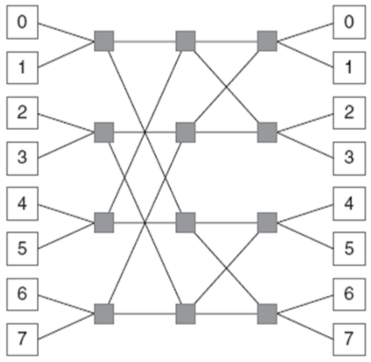
- **Meshes and tori can be extended by adding bypass links to increase performance at the cost of higher area**

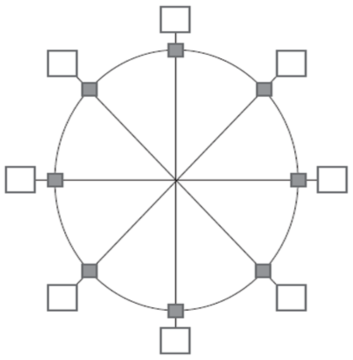


Other NoC Topologies

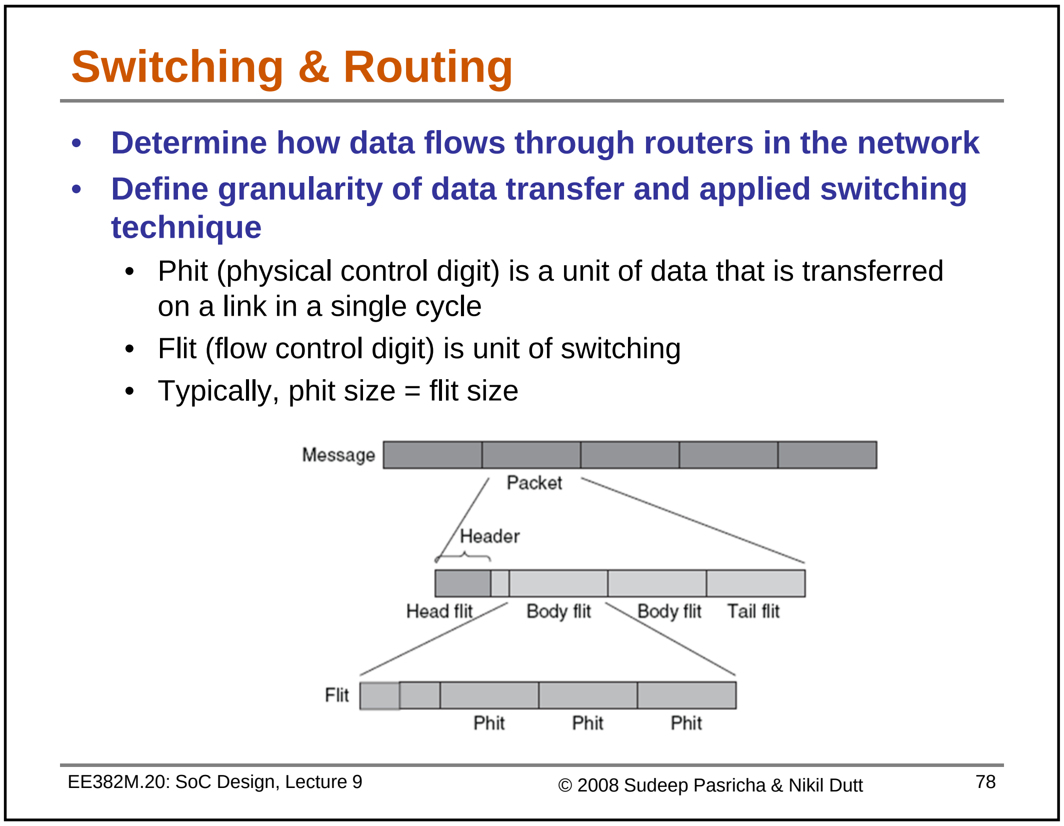
- Tree (indirect)
- Butterfly (indirect)
- Octagon (direct)
- Irregular, …





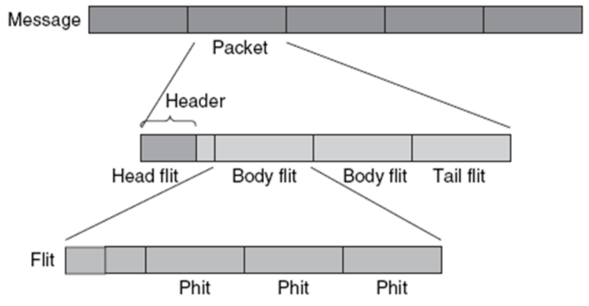


EE382M.20: SoC Design, Lecture 9 © 2018 A. Gerstlauer 77



Switching & Routing

- Determine how data flows through routers in the network
- Define granularity of data transfer and applied switching technique
  - Phit (physical control digit) is a unit of data that is transferred on a link in a single cycle
  - Flit (flow control digit) is unit of switching
  - Typically, phit size = flit size



EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 78

Switching Strategies (1)

- **Two main modes of transporting flits in a NoC are circuit switching and packet switching**
- **Circuit switching**
  - Physical path between the source and the destination is reserved prior to the transmission of data
  - Message header flit traverses the network from the source to the destination, reserving links along the way
  - Advantage: low latency transfers, once path is reserved
  - Disadvantage: pure circuit switching does not scale well with NoC size
    - Several links are occupied for the duration of the transmitted data, even when no data is being transmitted
      - » For instance in the setup and tear down phases

Switching Strategies (2)

- **Virtual circuit switching**
  - Creates virtual circuits that are multiplexed on links
  - Number of virtual links (or virtual channels (VCs)) that can be supported by a physical link depends on buffers allocated to link
  - Allocating one buffer per virtual link
    - Depends on how virtual circuits are spatially distributed in the NoC, routers can have a different number of buffers
    - Can be expensive due to the large number of shared buffers
    - Multiplexing virtual circuits on a single link also requires scheduling at each router and link (end-to-end schedule)
    - Conflicts between different schedules can make it difficult to achieve bandwidth and latency guarantees
  - Allocating one buffer per physical link
    - Virtual circuits are time multiplexed with a single buffer per link
    - Uses time division multiplexing (TDM) to statically schedule the usage of links among virtual circuits
    - Flits are typically buffered at the NIs and sent into the NoC according to the TDM schedule
    - Global scheduling with TDM makes it easier to achieve end-to-end bandwidth and latency guarantees
    - Less expensive router implementation, with fewer buffers

Switching Strategies (3)

- **Packet Switching**
  - Packets are transmitted from source and make their way independently to receiver
    - Possibly along different routes and with different delays
  - Zero start up time, followed by a variable delay due to contention in routers along packet path
  - QoS guarantees are harder to make in packet switching than in circuit switching
  - Three main packet switching scheme variants

1. **Store-and-forward (SAF) packet switching**
   - Packet is sent from one router to the next only if the receiving router has buffer space for entire packet
   - Buffer size in the router is at least equal to the size of a packet
   - Disadvantage: excessive buffer requirements

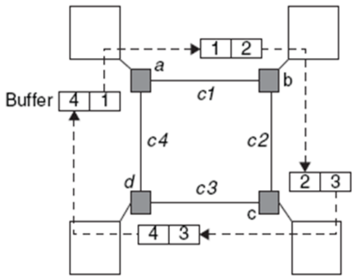
Switching Strategies (4)

2. **Virtual cut through (VCT) packet switching**
   - Reduces router latency over SAF switching by forwarding first flit of a packet as soon as space for the entire packet is available in the next router
   - If no space is available in receiving buffer, no flits are sent, and the entire packet is buffered
   - Same buffering requirements as SAF switching
3. **Wormhole (WH) packet switching**
   - Flit from a packet is forwarded to receiving router if space exists for that flit
   - Parts of the packet can be distributed among two or more routers
   - Buffer requirements are reduced to one flit, instead of an entire packet
   - More susceptible to deadlocks due to usage dependencies between links

Routing

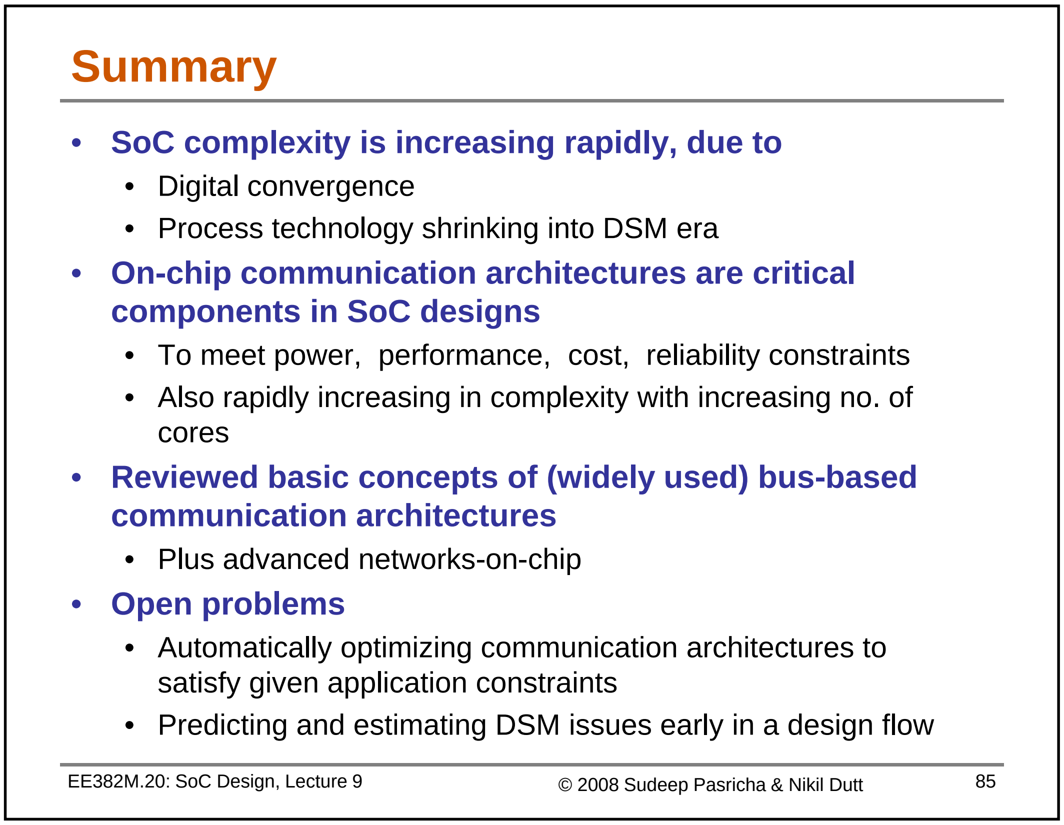
- **Static vs. dynamic routing**
  - Fixed vs. adaptive source-destination paths
- **Distributed vs. source routing**
  - Packets carry destination only or complete route
- **Minimal vs. non-minimal routing**
  - Always shortest path or deviations allowed

- ➢ **Deadlocks?**
  - ➢ Cyclic resource dependency
- ➢ **Livelocks?**
  - ➢ "Hot potato"
- ➢ **Starvation?**
  - ➢ Low-priority traffic fairness



Flow Control

- **Goal of flow control is to allocate network resources for packets traversing a NoC**
  - Can also be viewed as a problem of resolving contention during packet traversal
- **At the data link-layer level, when transmission errors occur, recovery from the error depends on the support provided by the flow control mechanism**
  - E.g. if a corrupted packet needs to be retransmitted, flow of packets from the sender must be stopped, and request signaling must be performed to reallocate buffer and bandwidth resources
- **Most flow control techniques can manage link congestion**
- **But not all schemes can (by themselves) reallocate all the resources required for retransmission when errors occur**
  - Either error correction or a scheme to handle reliable transfers must be implemented at a higher layer



Summary

- **SoC complexity is increasing rapidly, due to**
  - Digital convergence
  - Process technology shrinking into DSM era
- **On-chip communication architectures are critical components in SoC designs**
  - To meet power, performance, cost, reliability constraints
  - Also rapidly increasing in complexity with increasing no. of cores
- **Reviewed basic concepts of (widely used) bus-based communication architectures**
  - Plus advanced networks-on-chip
- **Open problems**
  - Automatically optimizing communication architectures to satisfy given application constraints
  - Predicting and estimating DSM issues early in a design flow

EE382M.20: SoC Design, Lecture 9 © 2008 Sudeep Pasricha & Nikil Dutt 85