ECE 551
System on Chip Design

Bus Standards (AMBA Example)

Garrett S. Rose
Fall 2018

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Bus Standards

- SoC components (IPs) interface to other components via a set of pins
  - responsible for sending/receiving addresses, data, control
- Number and functionality of pins must adhere to a specific interface standard
- Important for seamless integration of SoC IPs – avoid integration mismatches
  - e.g. 1 - connecting IP with 32 data pins to a 30 bit data bus
  - e.g. 2 - connecting IP supporting data bursts to a bus with no burst support
- Mismatches require “logic wrappers” at IP interfaces
  - to ensure correct data transfers
  - time consuming to create, reduce performance, take up area

Bus Standards

- Interface standards define a specific data transfer protocol
  - decide number and functionality of pins at IP interfaces
  - make it easy to connect diverse IPs quickly
- Two categories of standards for SoC communication:
  - Standard bus architectures
    - define interface between IPs and bus architecture
    - define (at least some) specifics of bus architecture that implements data transfer protocol
  - Socket based bus interface standards
    - define interface between IPs and bus architecture
    - freedom w.r.t choice and implementation of bus architecture
- Ideally, designers want one standard to interconnect all IPs
- In reality, several competing standards have emerged

Standard Bus Architectures

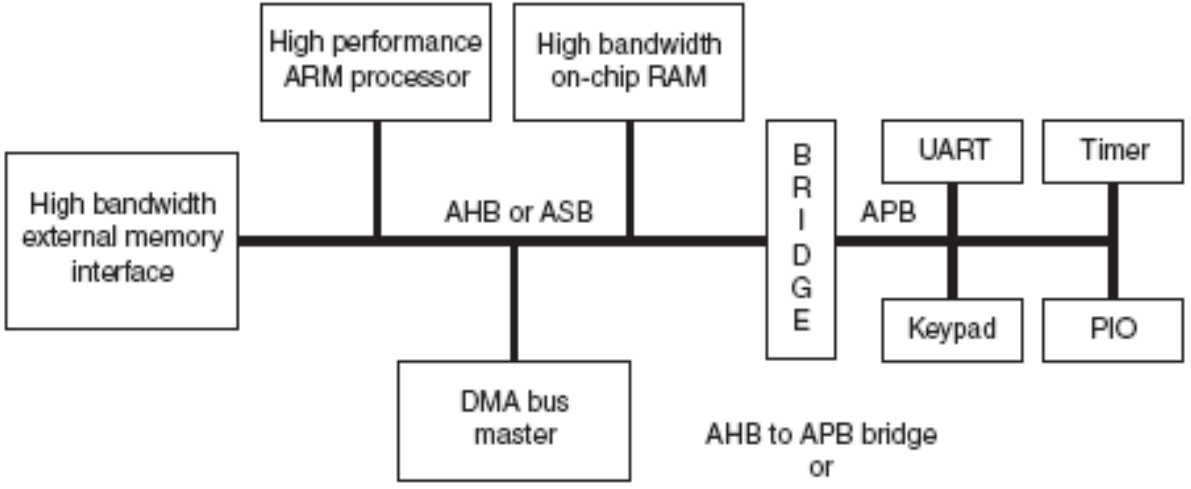
- AMBA 2.0, 3.0 (ARM)
- CoreConnect (IBM)
- Sonics Smart Interconnect (Sonics)
- STBus (STMicroelectronics)
- Wishbone (Opencores)
- Avalon (Altera)
- PI Bus (OMI)
- MARBLE (Univ. of Manchester)
- CoreFrame (PalmChip)
- …

widely used

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)
- …

AMBA 2.0



AMBA AHB

* High performance
* Pipelined operation
* Multiple bus masters
* Burst transfers
* Split transactions

AMBA ASB

* High performance
* Pipelined operation
* Multiple bus masters

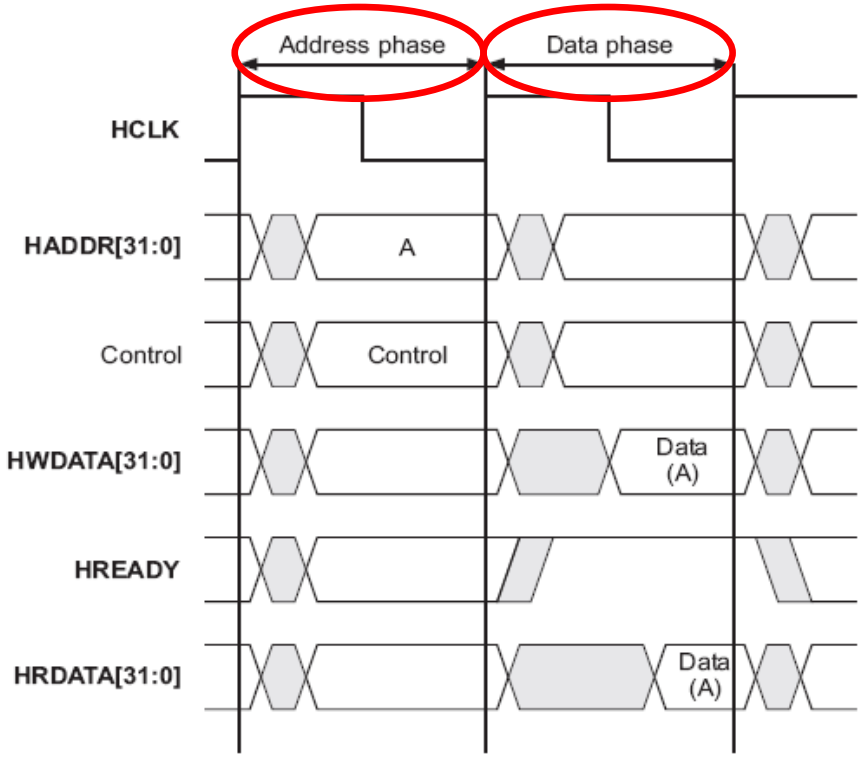
AMBA APB

* Low power
* Latched address and control
* Simple interface
* Suitable for many peripherals

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

AHB Basic Transfer

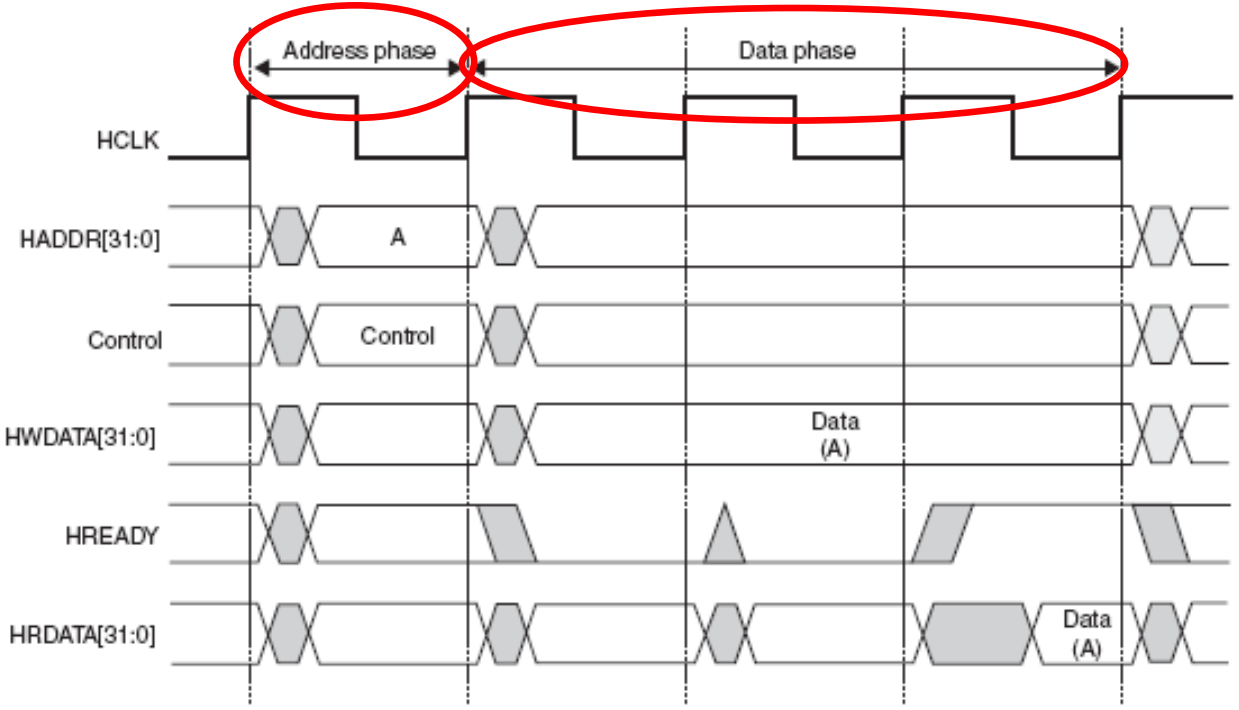
Split ownership of Address and Data bus



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

AHB Basic Transfer

Data transfer with slave wait states

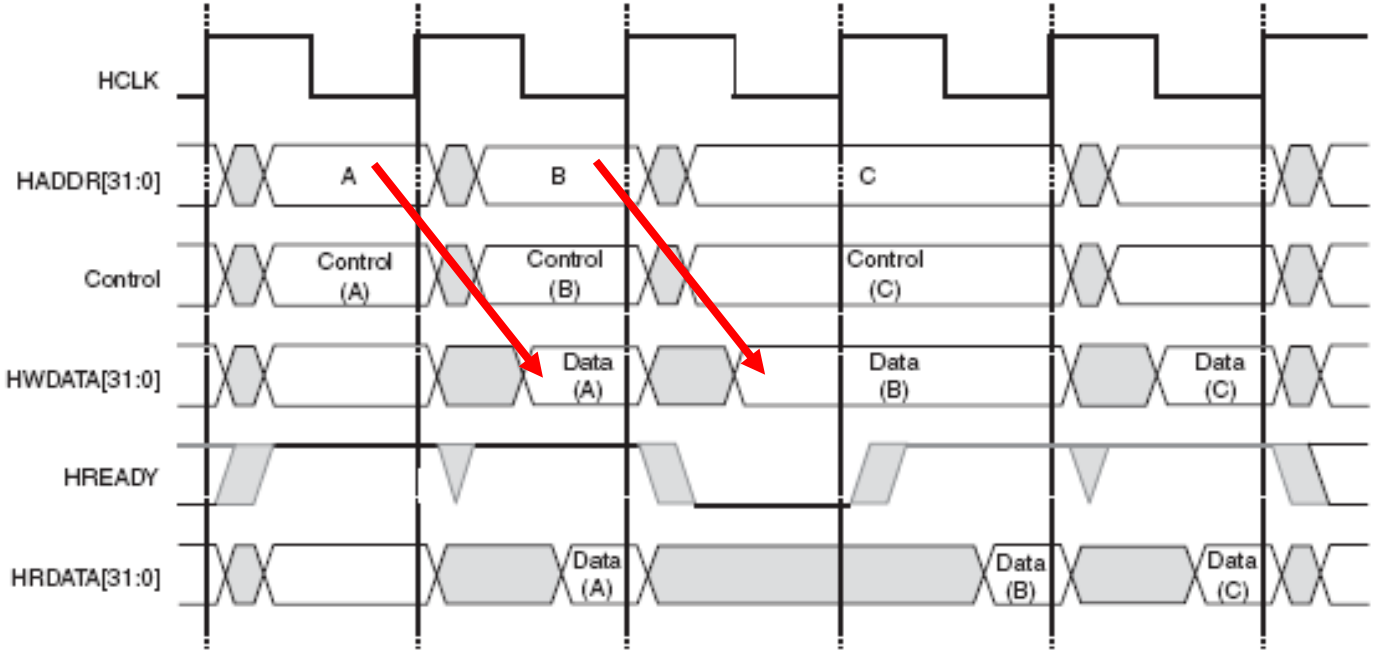


THE UNIVERSITY OF TENNESSEE KNOXVILLE

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

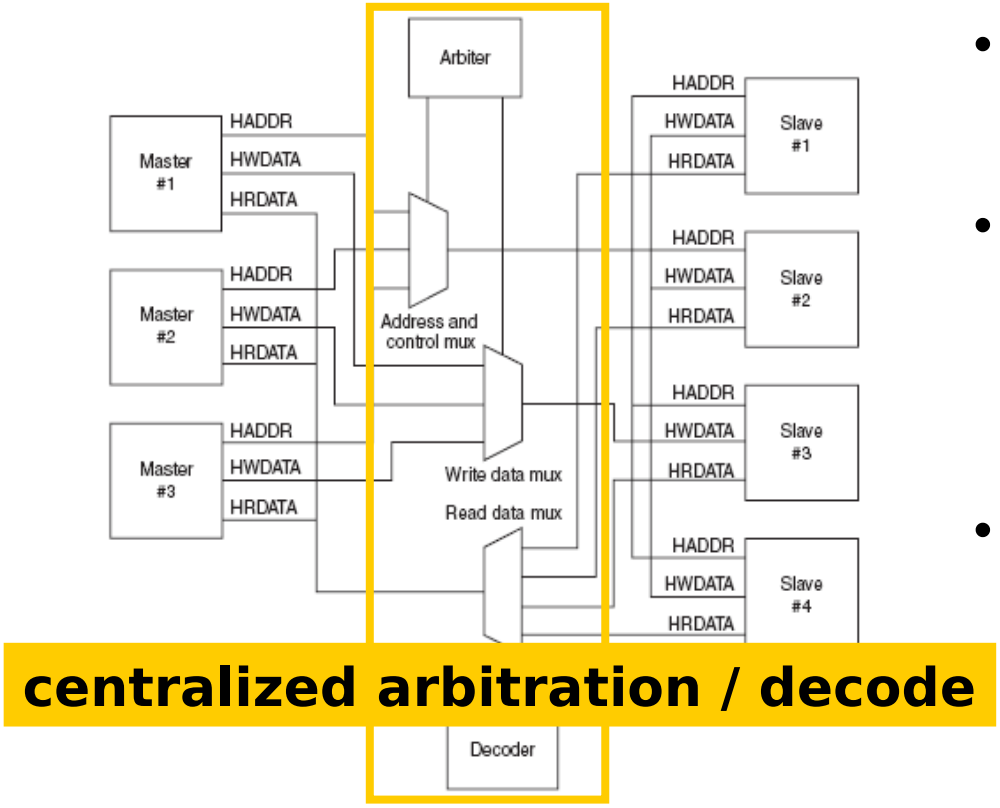
AHB Pipelining

Transaction pipelining increases bus bandwidth



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

AHB Architecture

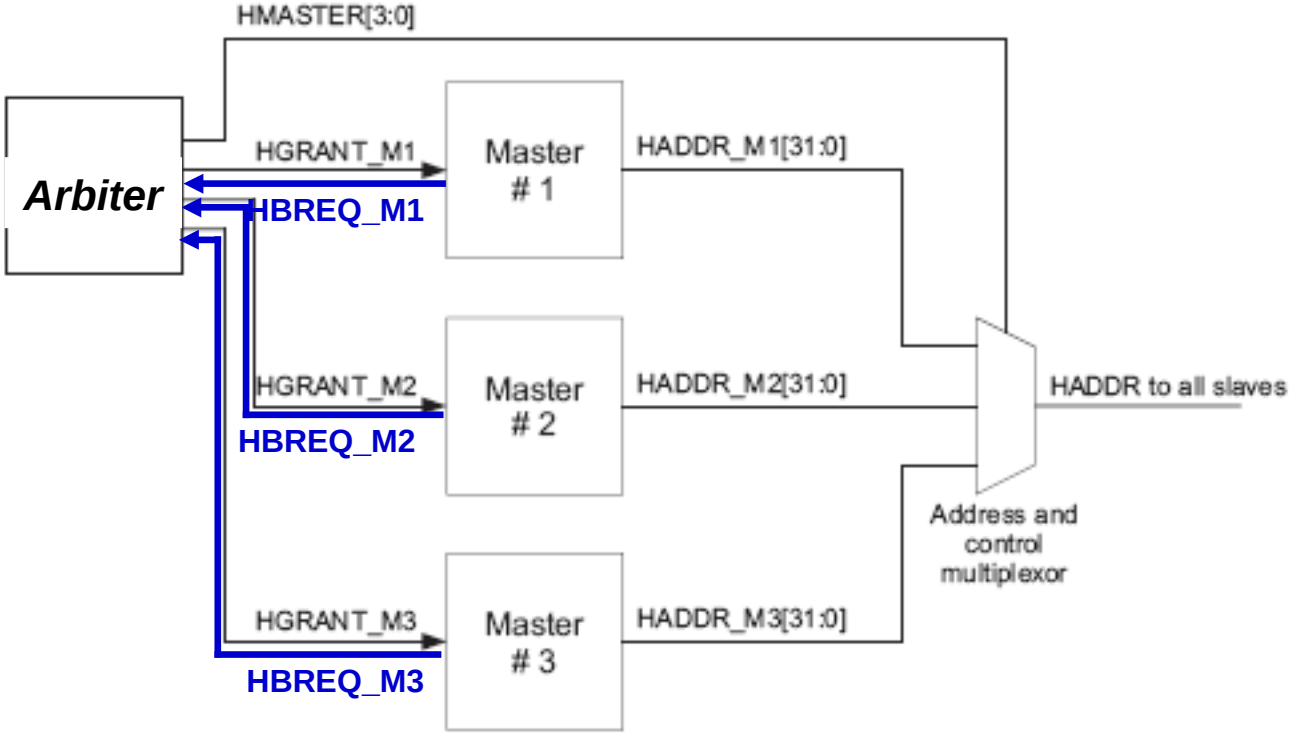


- 1 unidirectional address bus (HADDR)
- 2 unidirectional data buses (HWDATA, HRDATA)
- At any time only 1 active data bus

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

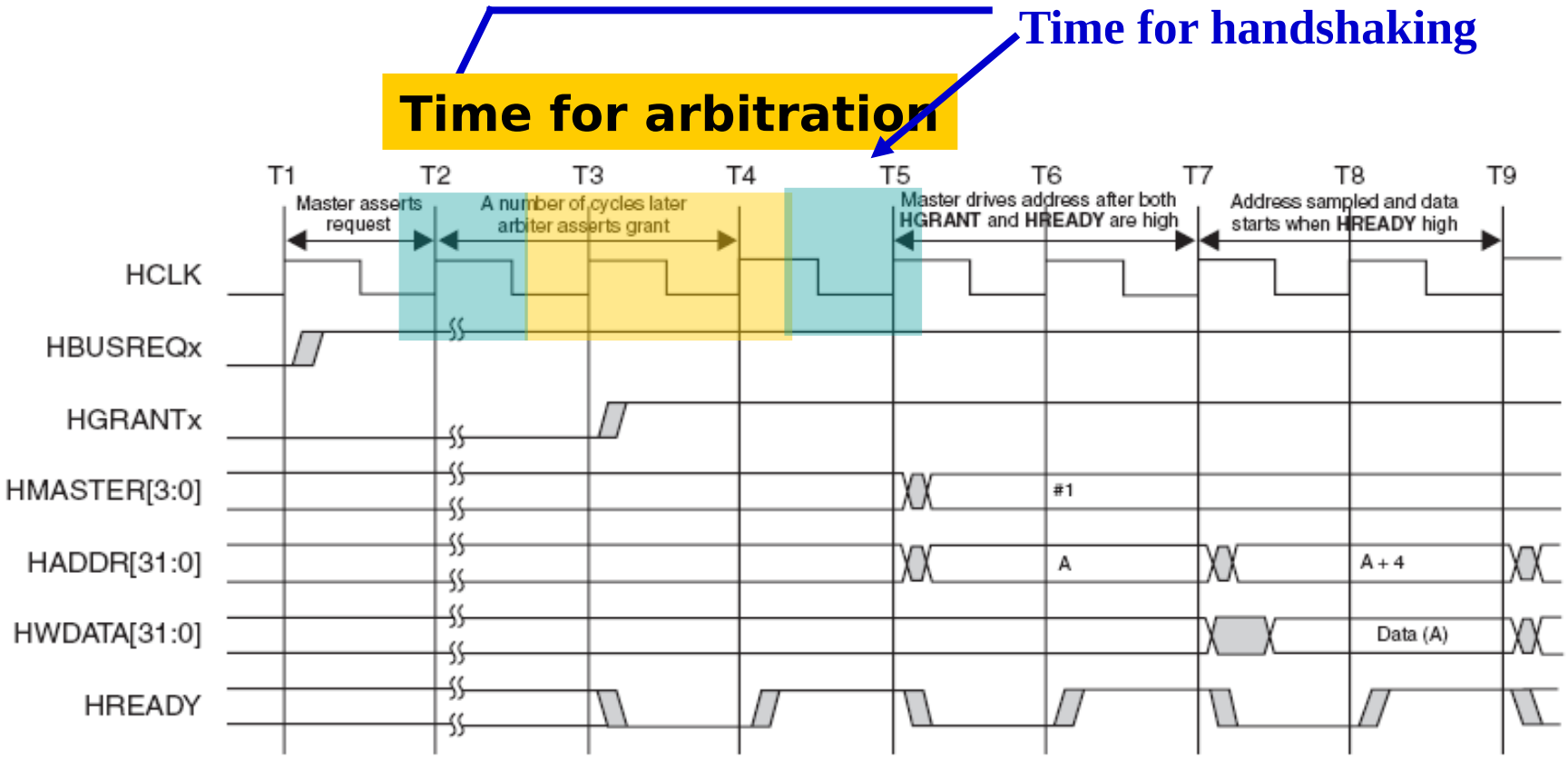
AHB Arbitration

Arbitration protocol is specified, but not the arbitration policy



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

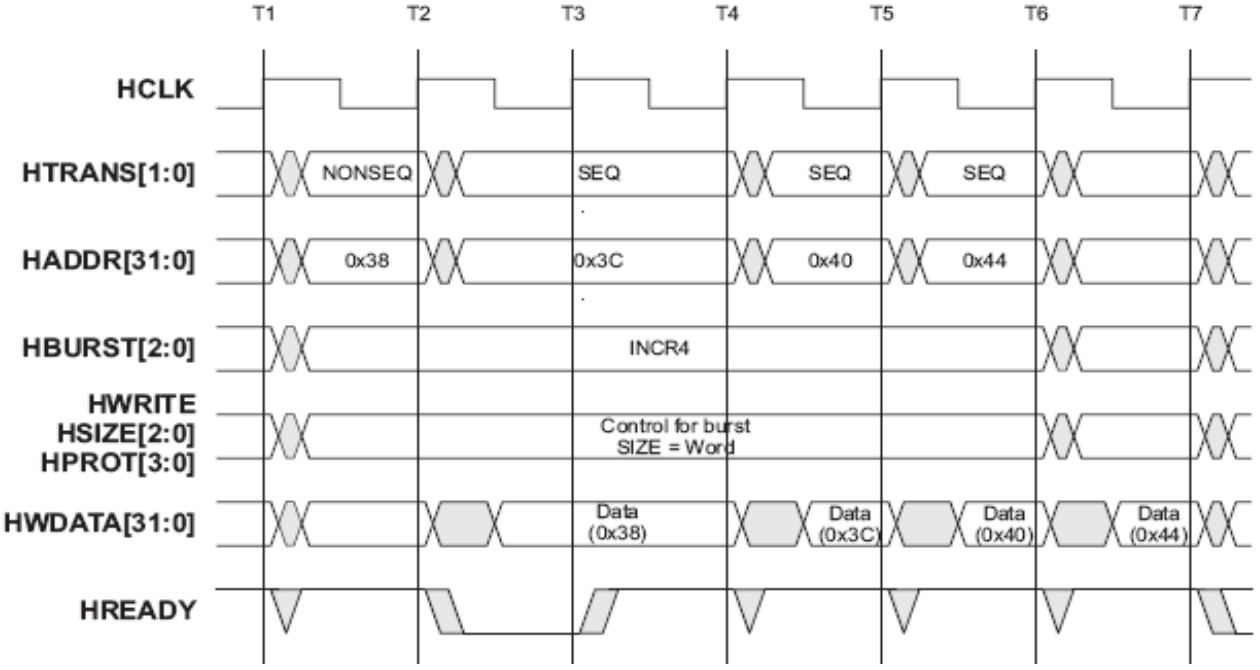
Cost of Arbitration in AHB



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

AHB Pipelined Burst

Bursts cut down arbitration and handshaking time
-- improves performance



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

AHB Burst Types

- 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

Fixed length bursts

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

AHB Control Signals

- 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	-

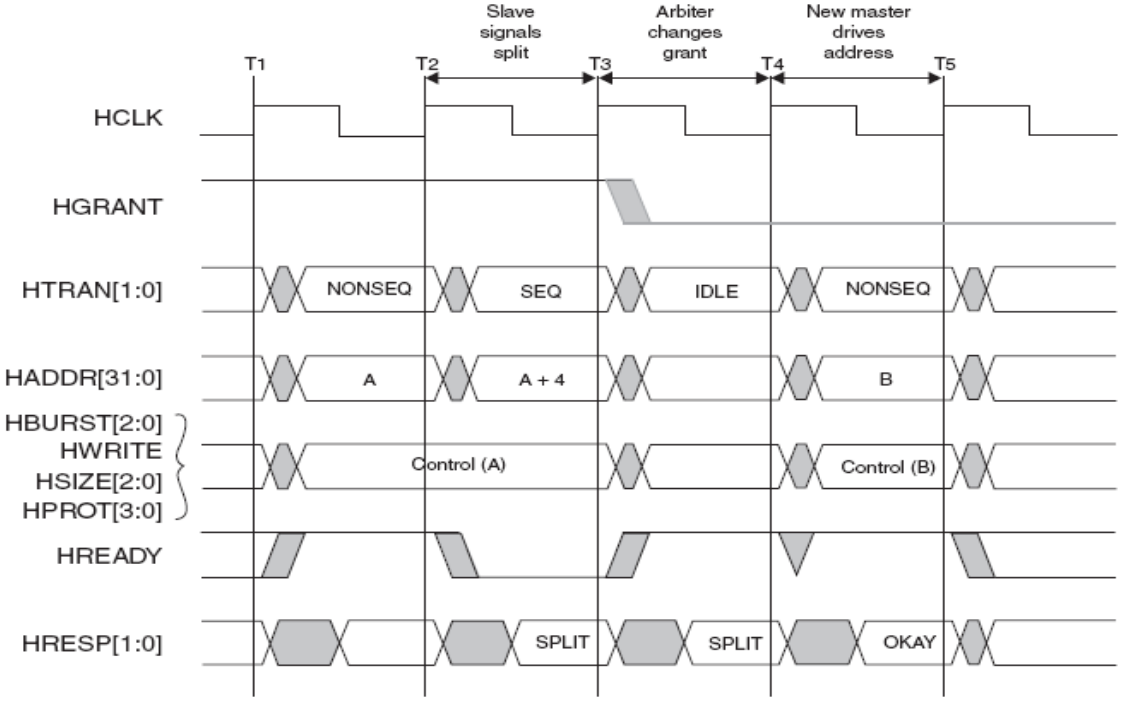
AHB Control Signals

- Protection control
  - HPROT[3:0], provide 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

AHB Split Transfer

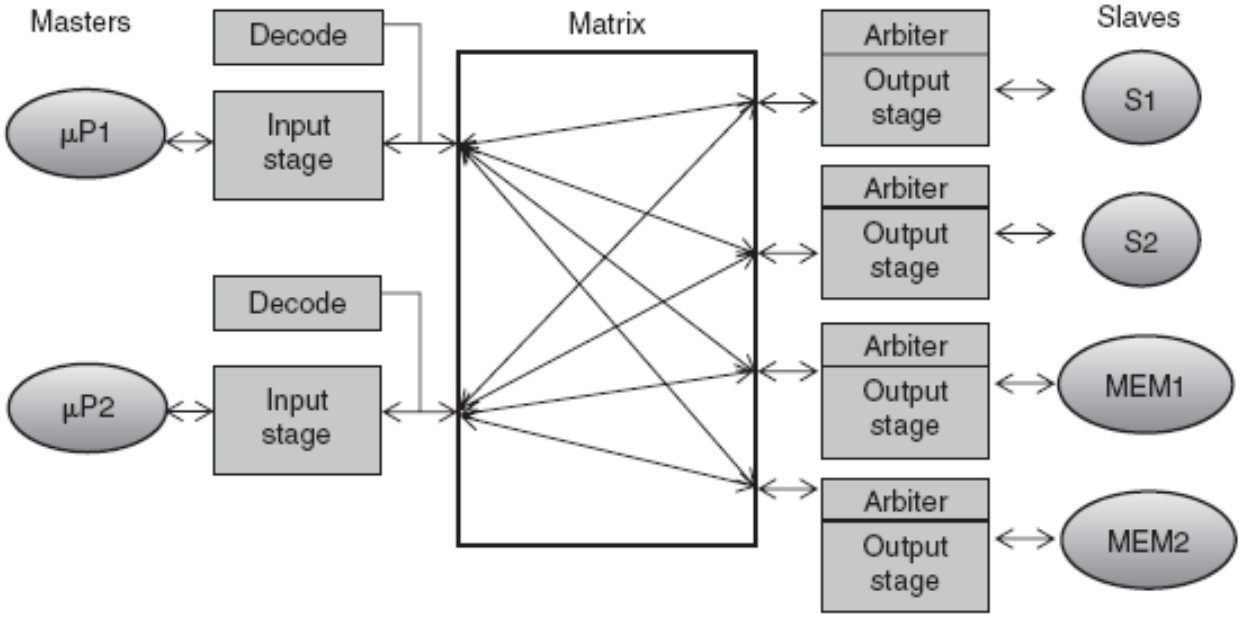
- Improves bus utilization
- May cause deadlocks if not carefully implemented



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

AHB Bus Matrix Topology

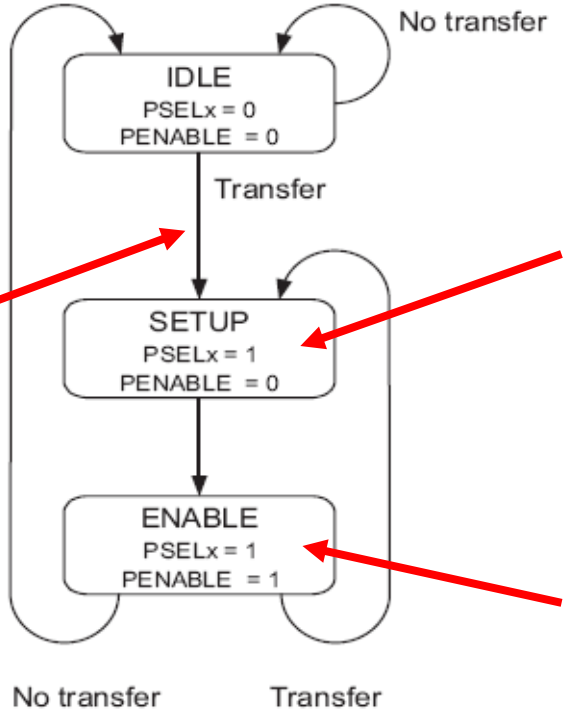
In addition to shared bus and hierarchical bus, AHB can be implemented as a bus matrix



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

APB State Diagram

APB has no (multi-cycle) bursts or pipelined transfers



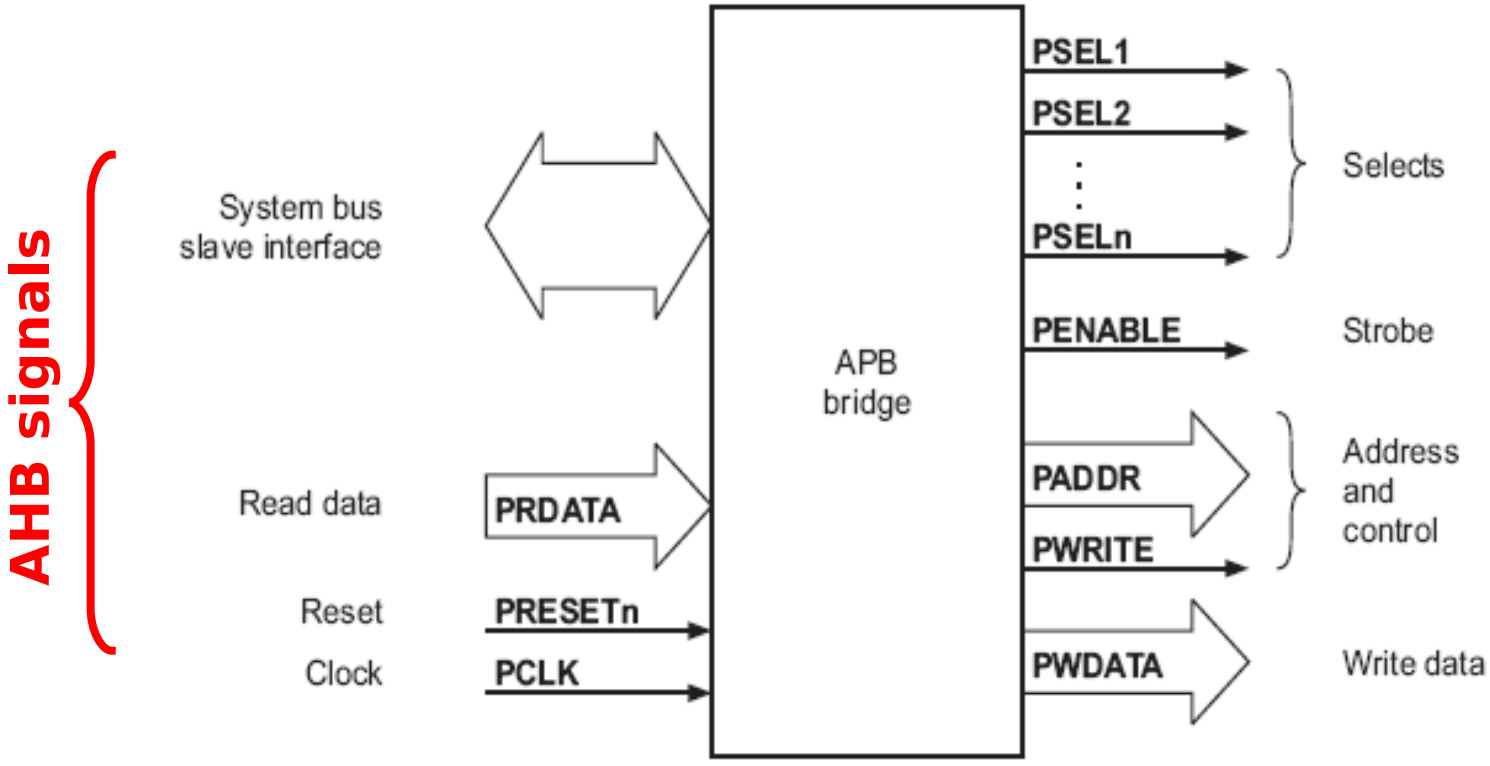
When AHB wants to drive a transfer

One cycle penalty for APB peripheral address decoding

Transfer occurs here

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

AHB-APB Bridge



High performance **Low power (and performance)**

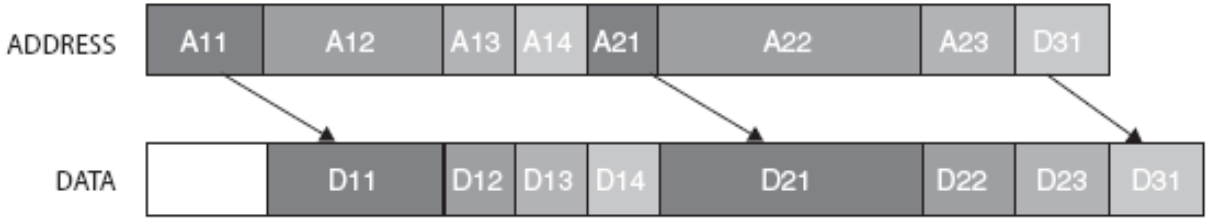
Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

AMBA 3.0

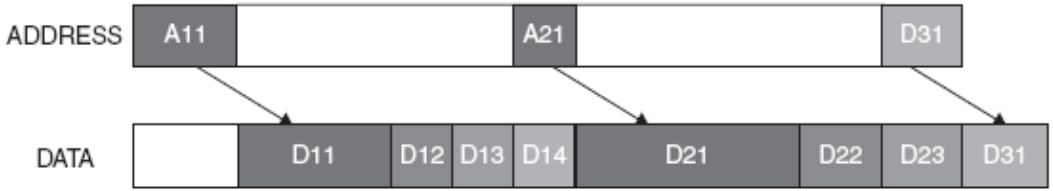
- Introduces AXI high performance protocol
- Support for separate read address, write address, read data, write data, write response channels
- Out of order (OO) transaction completion
- Fixed mode burst support
  - Useful for I/O peripherals
- Advanced system cache support
  - Specify if transaction 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

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



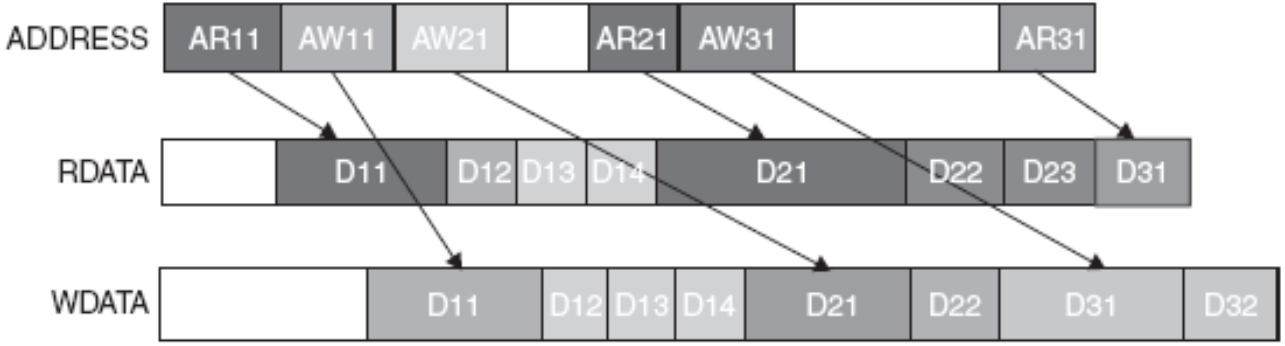
- AXI Burst
  - One Address for entire burst



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

AHB vs. AXI Burst

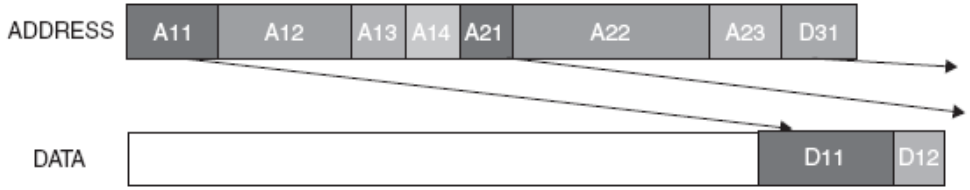
- AXI Burst
  - Simultaneous read, write transactions
  - Better bus utilization



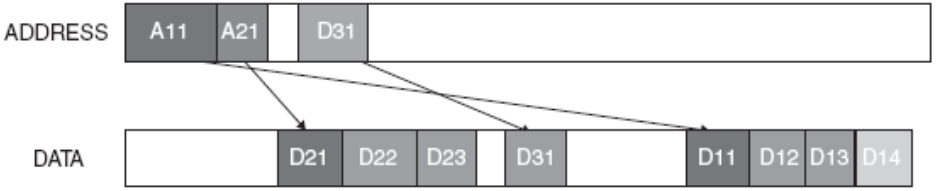
Source: S. Pasricha and N. Dutt, On-Chip Comm. Arch. – SOC Interconnect, Morgan Kaufmann, 2005.

AXI Out of Order Completion

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



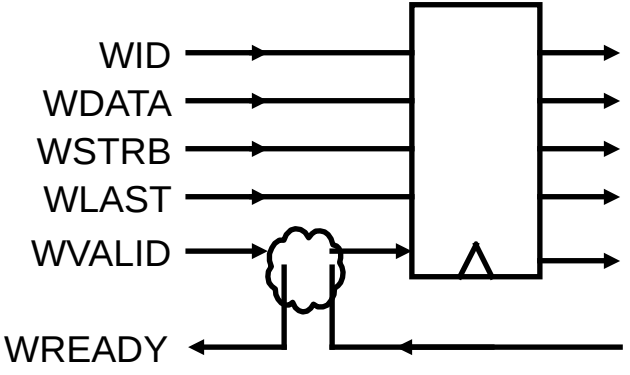
- With AXI Burst
  - Multiple outstanding addresses, out of order (OO) completion allowed
  - Fast slaves may return data ahead of slow slaves



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

Register Slices for Max Frequency

- Register slices can be applied across any channel
- Allows maximum frequency of operation by matching channel latency to channel delay
- Allows system topology to be matched to performance requirements



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.

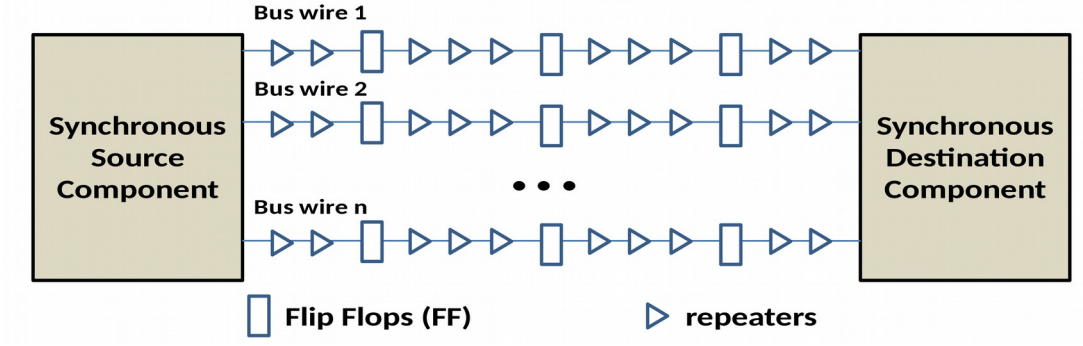
Physical Implementation Issues for Bus Wires

- Bus wires are implemented as long metal lines on a silicon wafer
  - transmitting data using electromagnetic waves (finite speed limit)
- As application performance requirements increase, clock frequencies are also increasing
  - Greater bus clock frequency = shorter bus clock period

    100 MHz = 10 ns ; 500 MHz = 2 ns
- Time allowed for a signal on a bus to travel from source to destination in a single bus clock cycle is decreasing
- Can take multiple cycles to send a signal across a chip
  - 6-10 bus clock cycles @ 50 nm
  - unpredictability in signal propagation time has serious consequences for performance and correct functioning of synchronous digital circuits

Physical Implementation Issues for Bus Wires

- Partition long bus wires into shorter ones
  - Hierarchical or split bus communication architectures
  - Register slices or buffers to pipeline long bus wires
    - enable signal to traverse a segment in a single clock cycle



- Asynchronous buses
  - No clock signal
- Low level techniques
  - Inserting repeaters or using fat wires

Deep Sub-µm (DSM) Buses

- With CMOS process technology scaling below 90 nm, SoCs have entered the deep submicron (DSM) era
  - High levels of component integration
  - High clock frequencies
  - Low signal voltages
- Buses significantly impacted by DSM effects – signal integrity issues
  - Scenario where the received signal at the destination is different from the transmitted signal at the source driver
  - Noise caused due to following factors
    - crosstalk
    - external electromagnetic interference
    - transmission line effects
    - soft errors

DSM Effects: Crosstalk

- Phenomenon of noise being caused on signal A due to coupling with another signal B
  - due to the close proximity of bus wires
  - near-field electromagnetic coupling causes inductive and capacitive crosstalk on bus signals
- Even when wires are far apart, crosstalk can still occur
  - due to coupling facilitated by
    - common substrate,
    - shared power supply or ground, or
    - a shared signal return path
- As wires become narrower with scaling and clock frequencies increase, fringing field effects and inductance effects become larger for wires
  - higher inductive and capacitive crosstalk

DSM Effects: Electromagnetic Interference (EMI)

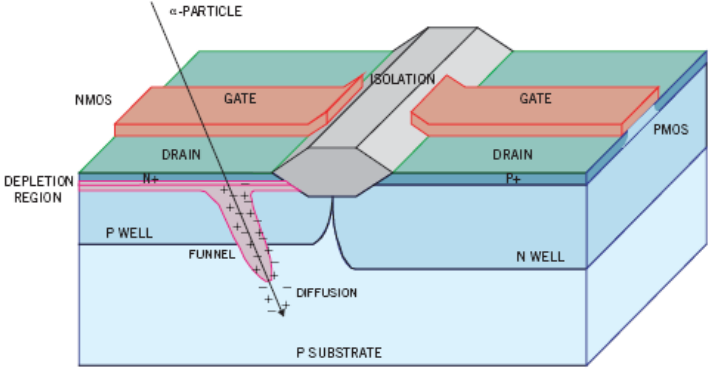
- Phenomenon of large external electric and magnetic fields coupling into circuits and creating unwanted noise
- EMI due to external and internal coupling is expected to increase with evolving technology
  - As highly integrated, portable wireless communication SoCs increasingly consist of analog, RF, and digital circuits
- Long on-chip buses in particular will be the sources and receptors of EMI noise

DSM Effects: Transmission Line Effects

- When a wire is longer than 1/10 of the wavelength of the signal frequency component that is transmitted, the wave nature of the propagated signal must be modeled
  - otherwise significant errors may result.
- wires will thus have to be modeled as transmission lines to avoid errors during signal analysis
- Discontinuities in these transmission lines can result in impedence mismatches
  - due to various factors such as capacitive loads, vias, wire bends, package pins, crossover wires, and non-ideal receivers
- Such mismatches will create noise as a result of signal reflections at the discontinuities

DSM Effects: Soft Errors

- Phenomenon of spurious pulses and interference with signals on buses
- Caused by
  - collision of thermal neutrons
    - produced by the decay of cosmic ray showers
  - alpha particles
    - produced by impurities in the substrate
- Highly integrated SoCs will be particularly susceptible to soft errors



DSM Effects

- Harder to guarantee error-free data transfers on buses
- Reduced signal swings in DSM technologies will result in a further reduction of voltage noise margins
  - Greater probability of transmission errors in the presence of even the smallest sources of noise
- Many other factors will further complicate bus design
  - increasing wire resistance due to skin effect at high frequencies
  - increasing number of metal layers that increase cross-layer coupling
  - timing errors due to jitters
- Emerging tools and methodologies for on-chip communication architecture design must predict and address DSM issues early in the design flow
  - instead of finding and fixing the problems in post-layout

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
- Open Problems
  - Designing communication architectures to satisfy diverse and complex application constraints
  - Predicting and estimating DSM issues early in a design flow