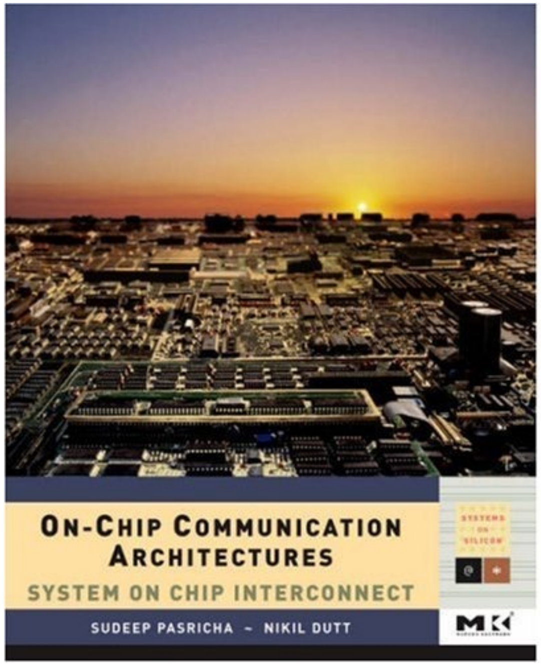
06/março/2022

On-Chip Communication Architectures


ON-CHIP COMMUNICATION ARCHITECTURES
SYSTEM ON CHIP INTERCONNECT
SUDEEP PASRICHA - NIKIL DUTT
MK

Introduction

ICS 295

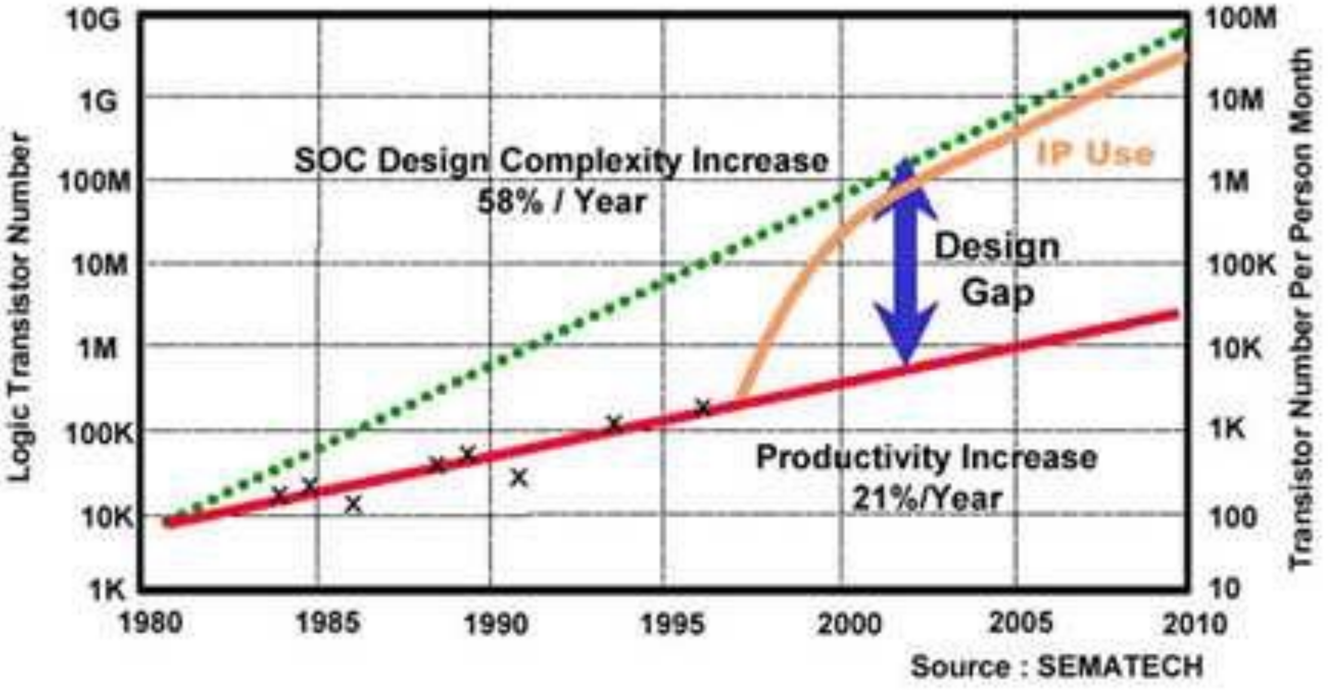
Sudeep Pasricha and Nikil Dutt

Slides based on book chapters 1, 2

Outline

- Introduction to SoC Design Trends
- Significance of on-chip communication architectures
- Bus-based communication architectures
  - Terminology
  - Physical structure
  - Clocking
  - Arbitration and decoding
  - Topology types
  - Data transfer modes
  - Physical implementation issues
  - DSM effects

Designer Productivity Gap

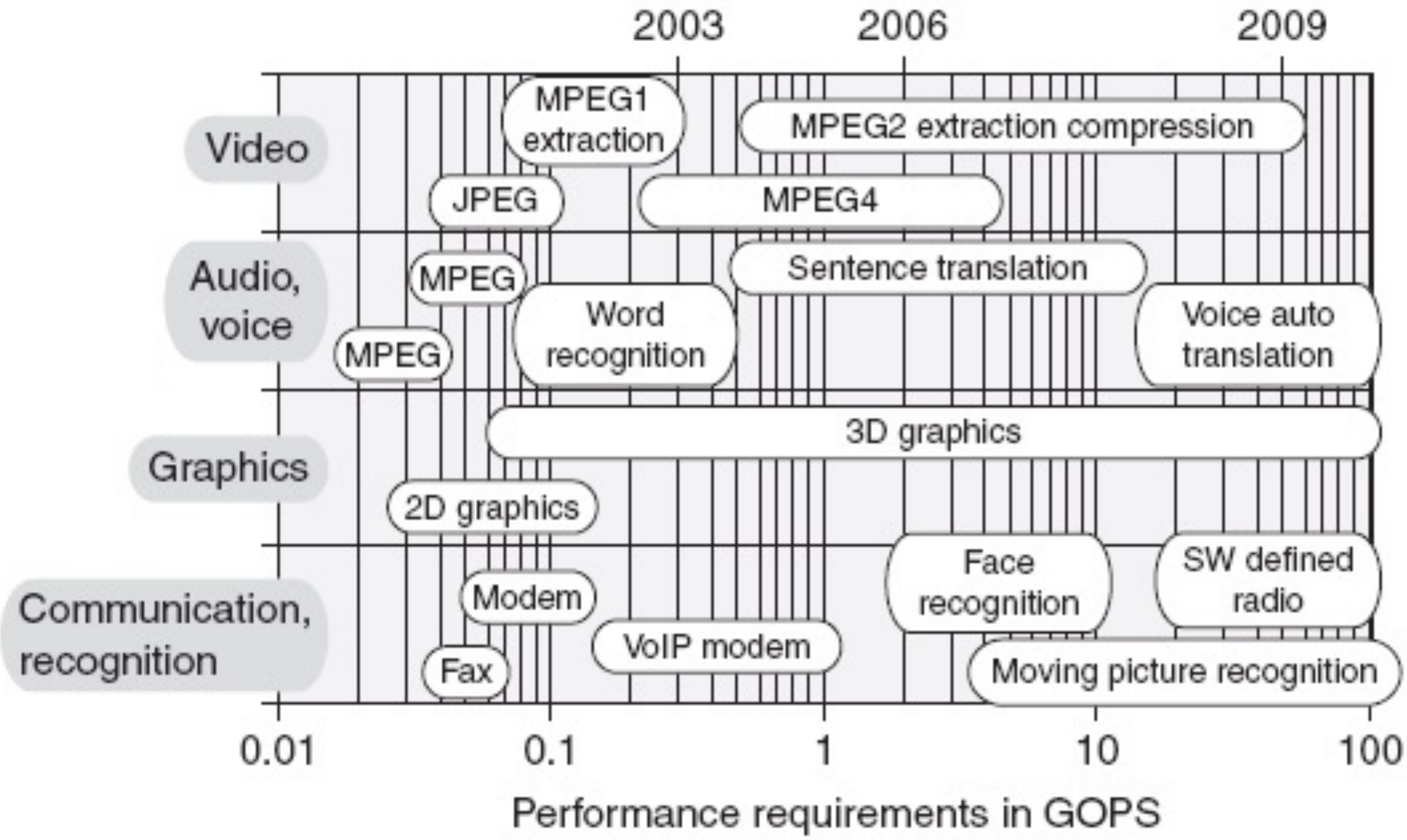


Source : SEMATECH

https://m.eet.com/media/1119378/figure%202.jpg

SoC designs today are complex, characterized by more and more IPs being integrated on a single chip, and a shrinking time-to-market

Emerging Application Requirements

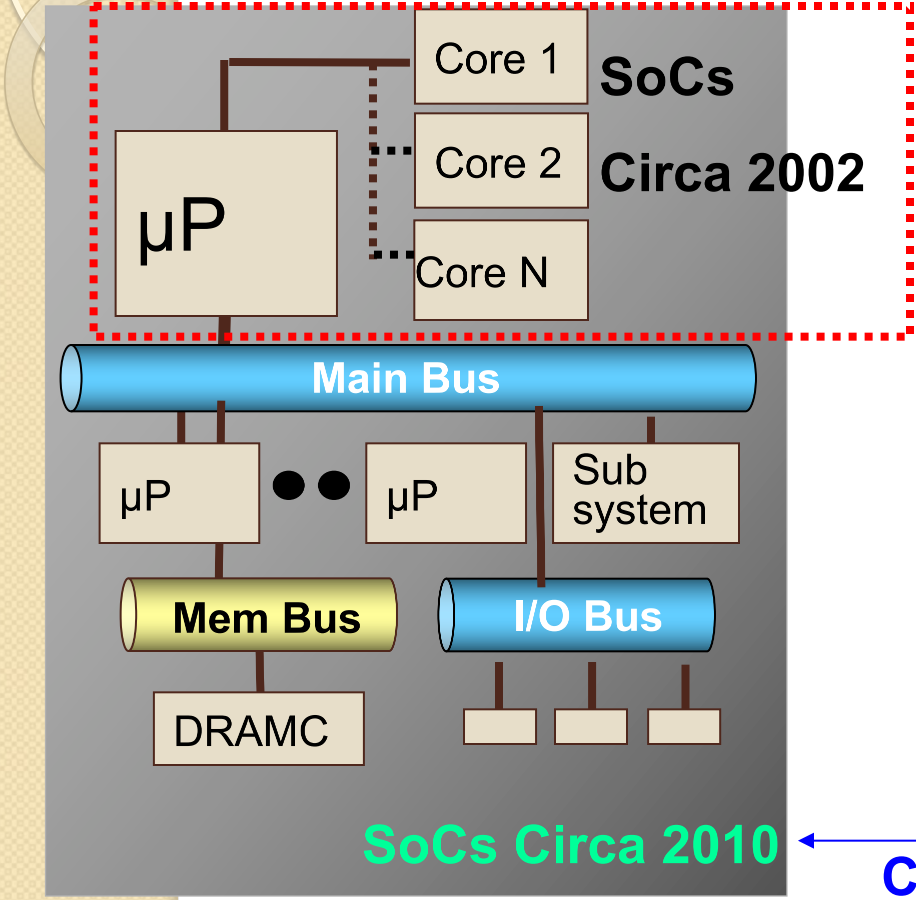


Coping with SoC Complexity

- Practicing IP based Design and Reuse (3PIP)
  - Raising the reuse factor from standard cells to IP blocks
    - e.g. predesigned hardware IPs for processors (ARM, PowerPC), communication (AMBA, CoreConnect), memories (Samsung SDRAMs, Denali SRAMs), I/O (UART, USB) etc.
  - IPs not just for hardware, but for software (device drivers, OS) too
  - Substantial reduction in SoC design and verification time
  - Requires initial investment to create reusable cores
    - but productivity improves with reuse

- IP Interfacing Standards
  - IP based design needs to handle incompatible IP interfaces
  - Assembling heterogeneous IPs for SoC design can take months!!!
  - Need for unified standard to quickly connect IPs
    - e.g. OCP-IP, VSIA VCI etc.

Data Flow Replacing Data Processing As Major SoC Design Challenge



Critical Decision was uP Choice

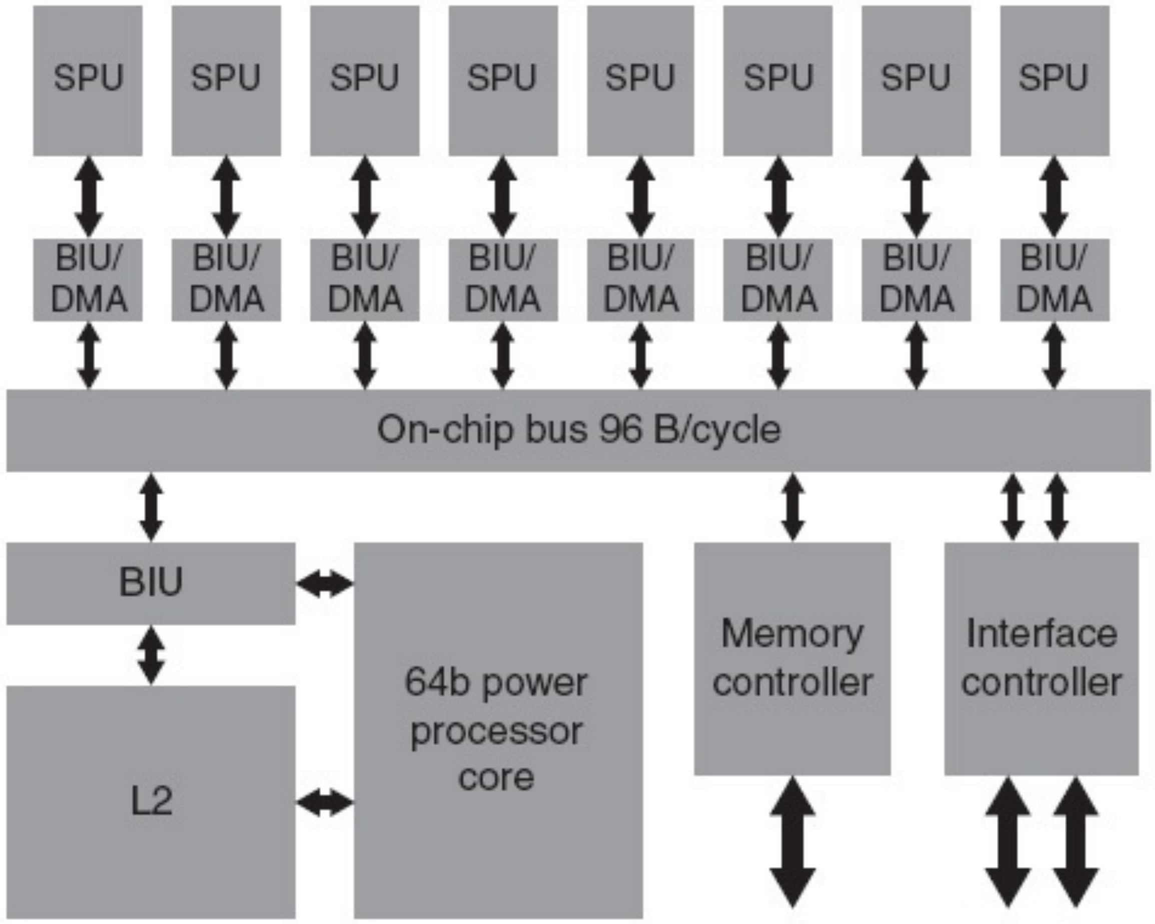
- ❑ Exploding core counts requiring more advanced Interconnects
- ❑ EDA cannot solve this architectural problem easily
- ❑ Complexity too high to hand craft (and verify!)

Critical Decision Is Interconnect Choice

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

Source: SONICS Inc.

Examples of On-chip Communication Architectures



- IBM Cell ring bus communication architecture

© 2008 Sudeep Pasricha & Nikil Dutt

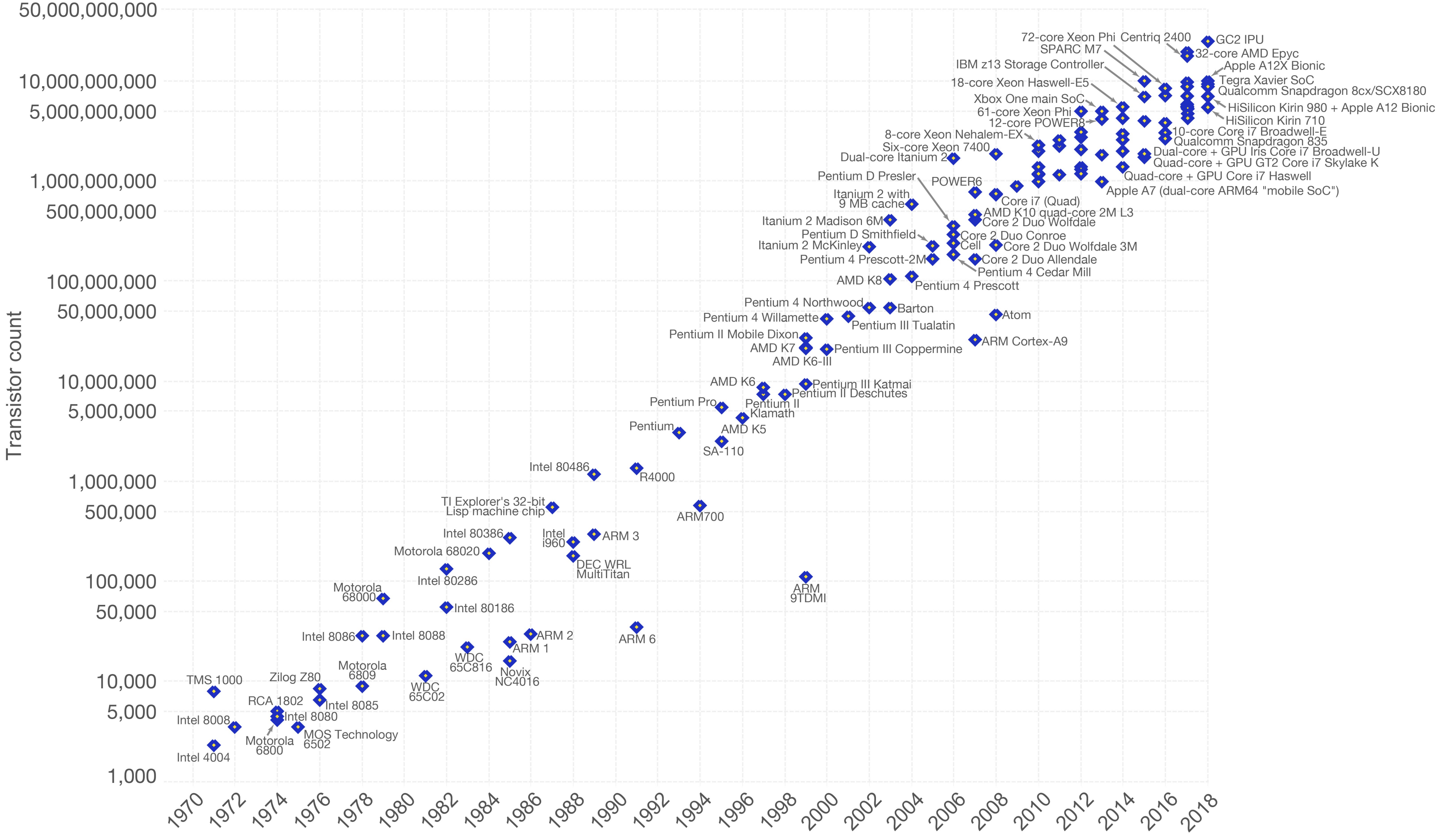
Need for Communication-centric Design Flow

- Communication is THE most critical aspect affecting system performance
- Communication architecture consumes upto 50% of total on-chip power
- Ever increasing number of wires, repeaters, bus components (arbiters, bridges, decoders etc.) increases system cost
- Communication architecture design, customization, exploration, verification and implementation takes up the largest chunk of a design cycle

Communication Architectures in today's complex systems **significantly** affect performance, power, cost and time-to-market!

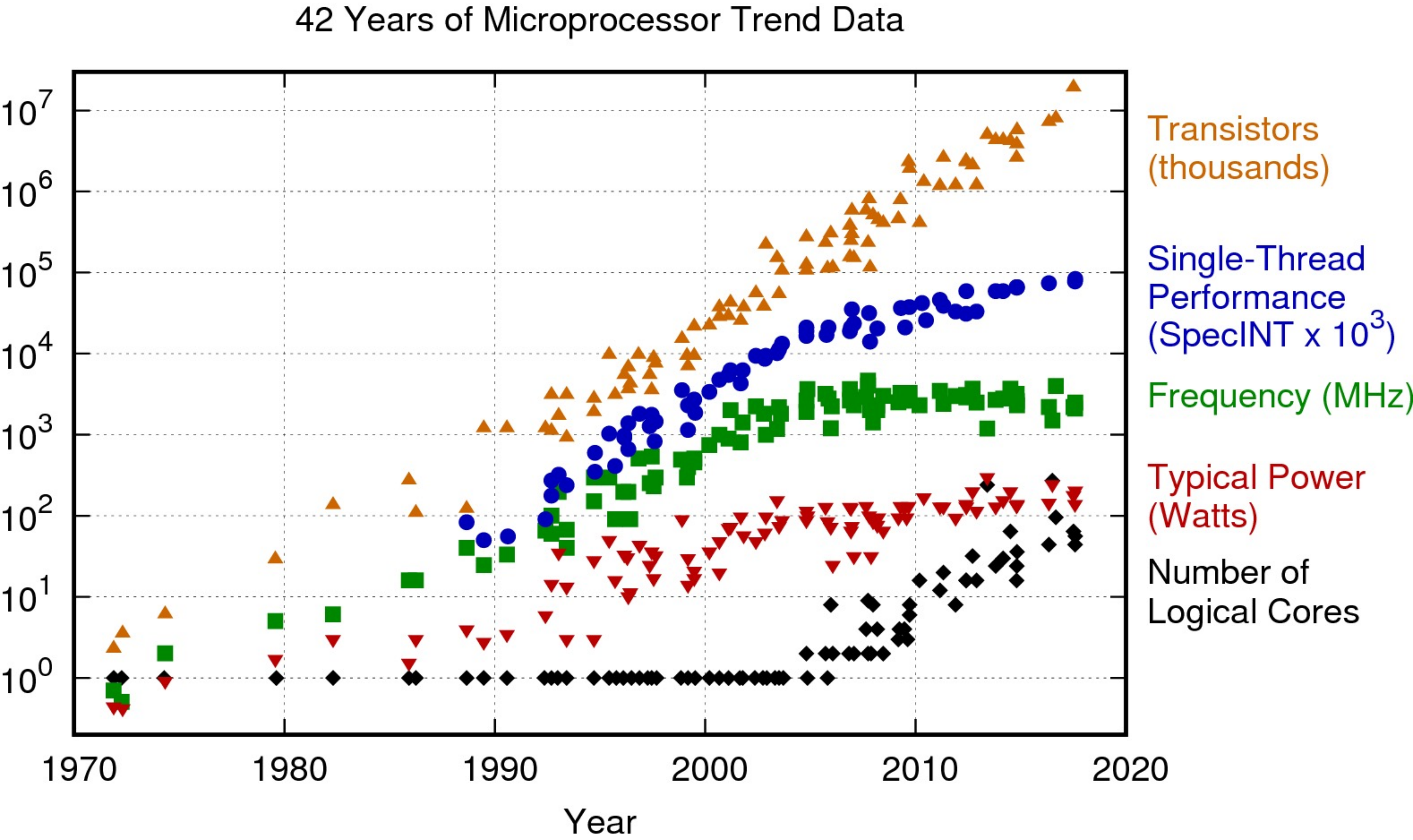
Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

https://ourworldindata.org/uploads/2019/05/Transistor-Count-over-time-to-2018.png



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

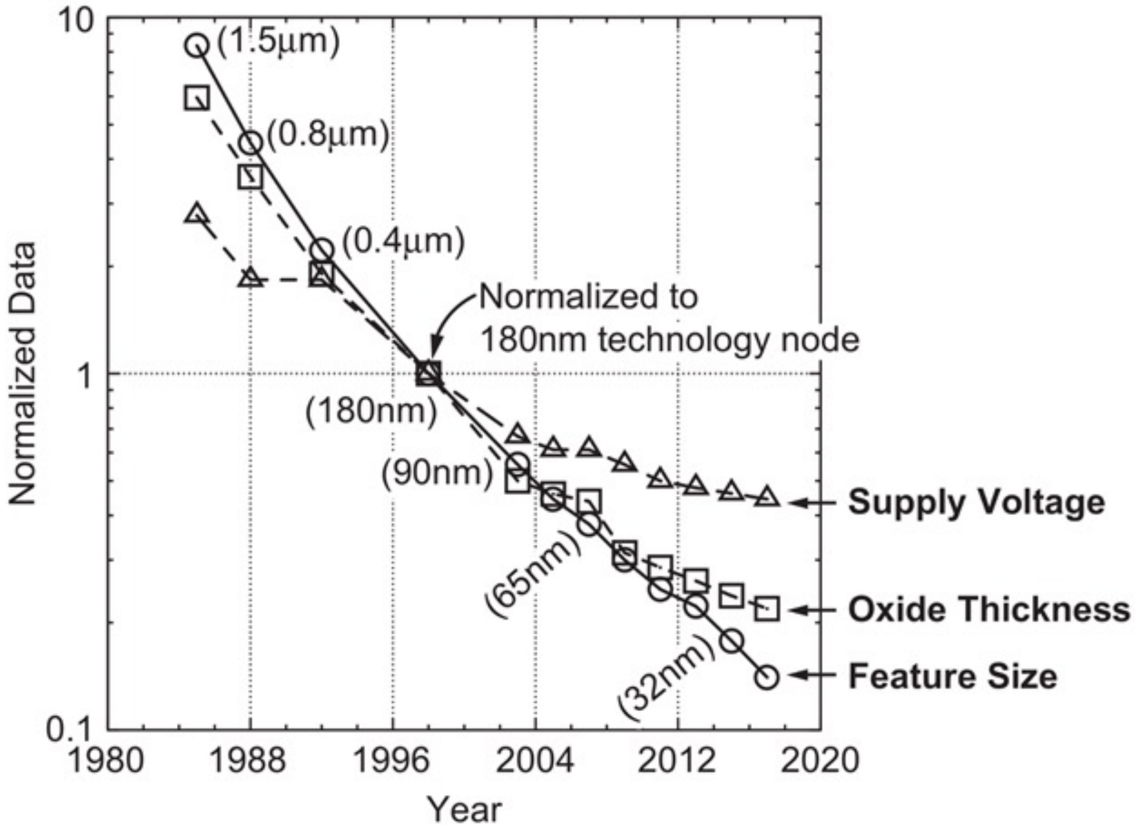
https://www.karlrupp.net/wp-content/uploads/2018/02/42-years-processor-trend.png

Dennard Scaling

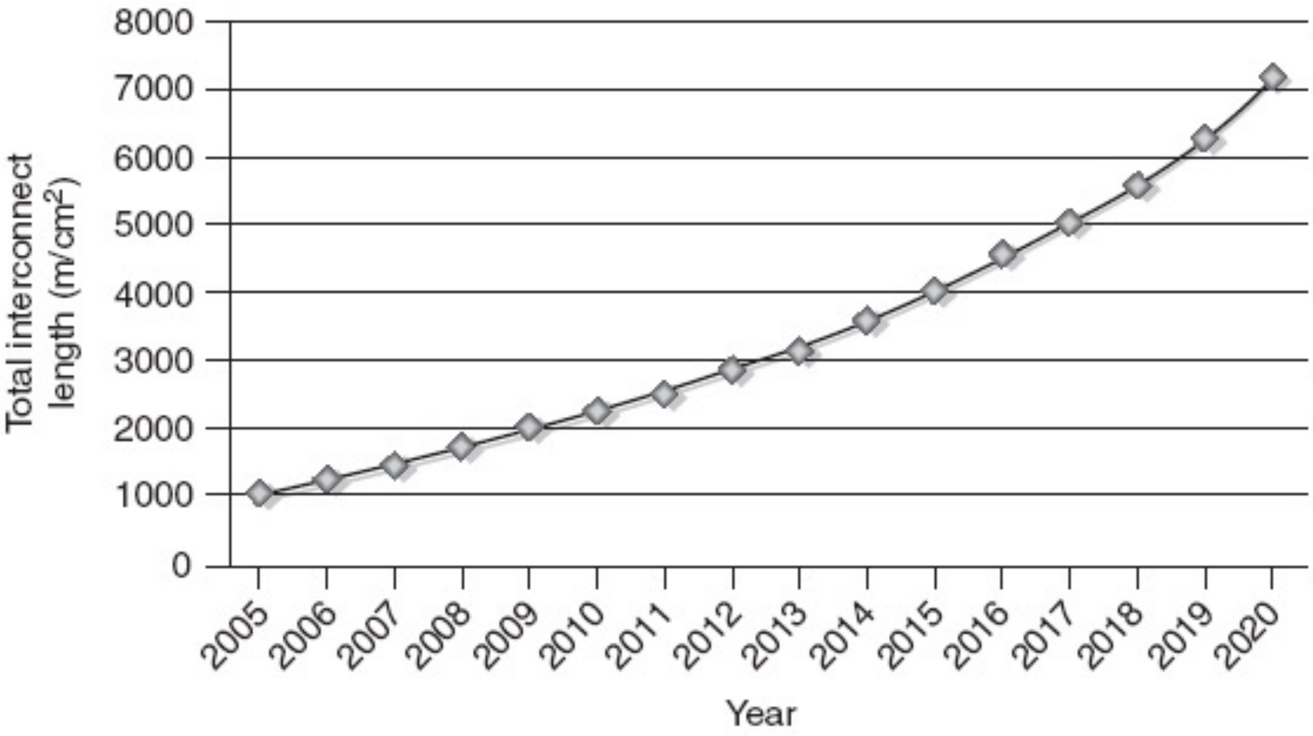
- Dennard (1974) observou que a tensão e a corrente devem ser proporcionais às dimensões lineares de um transistor
  - Assim, conforme os transistores encolheram, também diminuíram a tensão e a corrente necessárias; a potência é proporcional à área do transistor.
- Final da “lei de Dennard”

$$P = \alpha \cdot \ C \ \cdot \ F \ \cdot \ V^2$$

α - percent time switching
C - capacitance
F – frequency
V - voltage

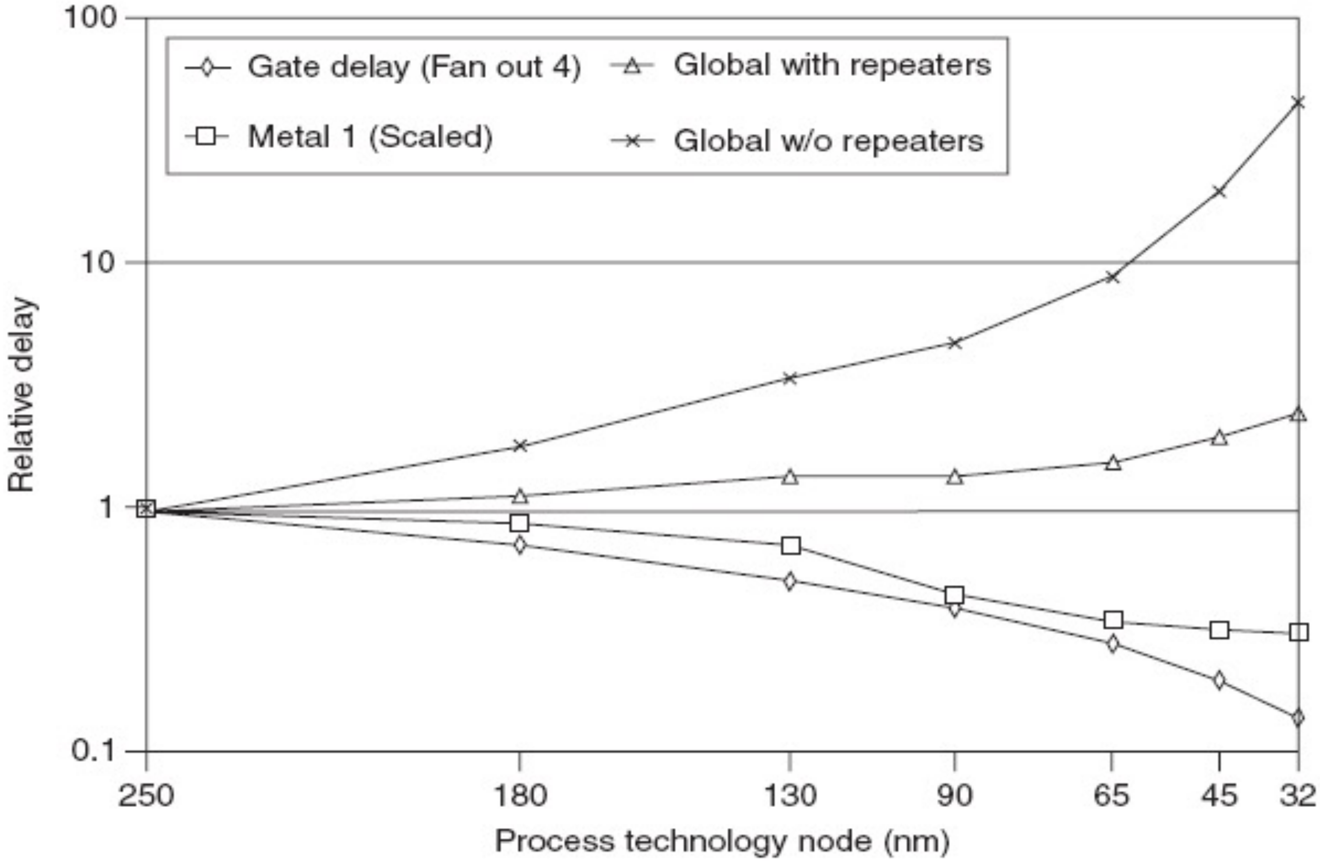


Technology Scaling Trends: Total Interconnect Length on a Chip



- Highlights importance of interconnect design in future technologies

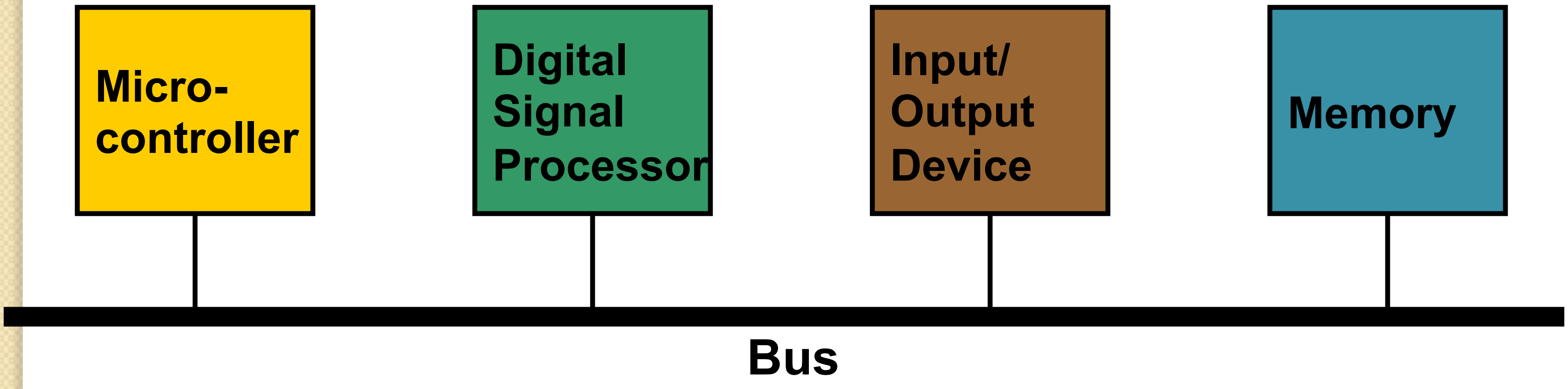
Technology Scaling Trends: Interconnect Performance



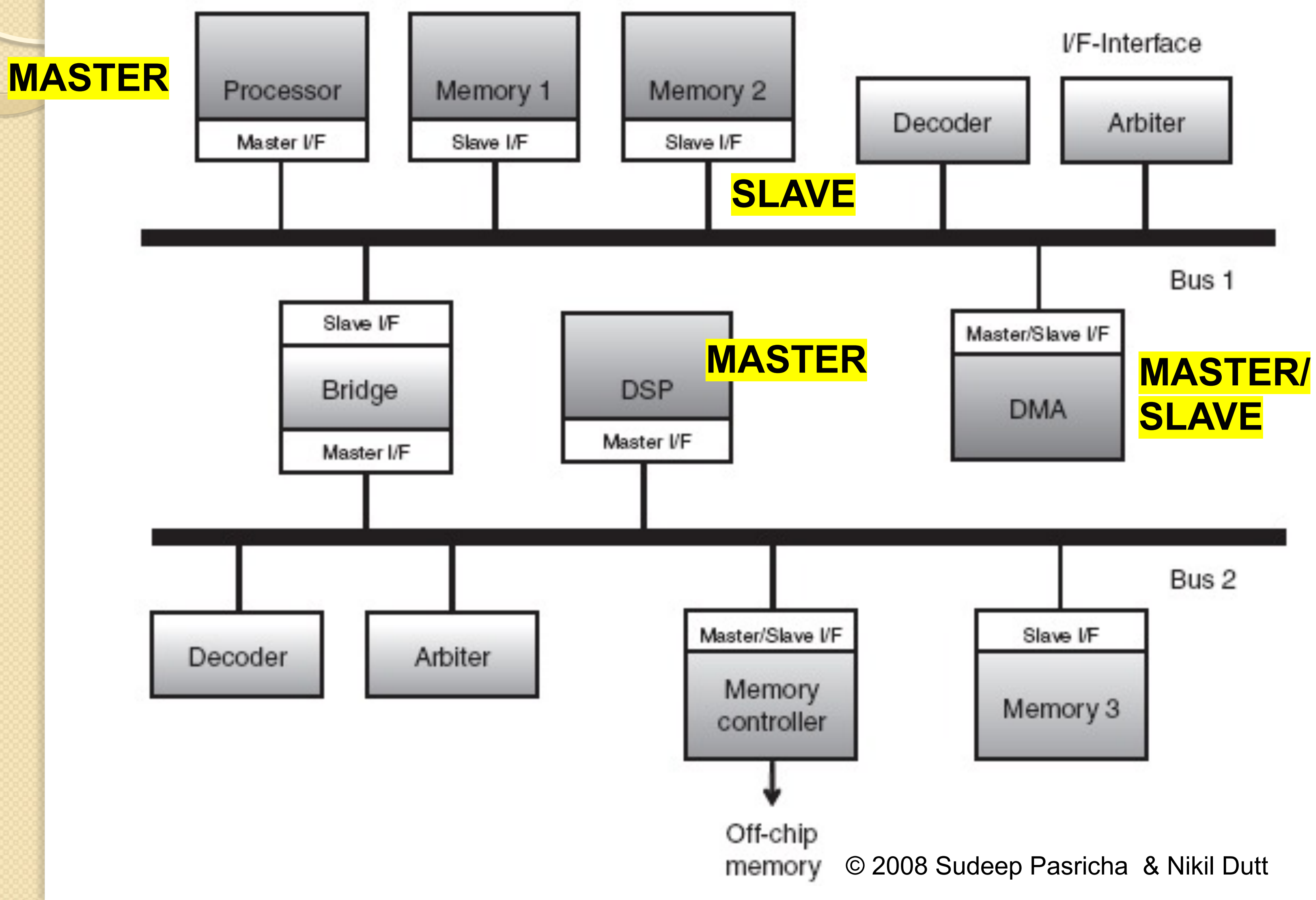
- Relative delay comparison of wires vs. process technology
- Increasing wire delay limits achievable performance

Bus based On-Chip Communication 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 IP component can transfer data on the shared bus at any given time



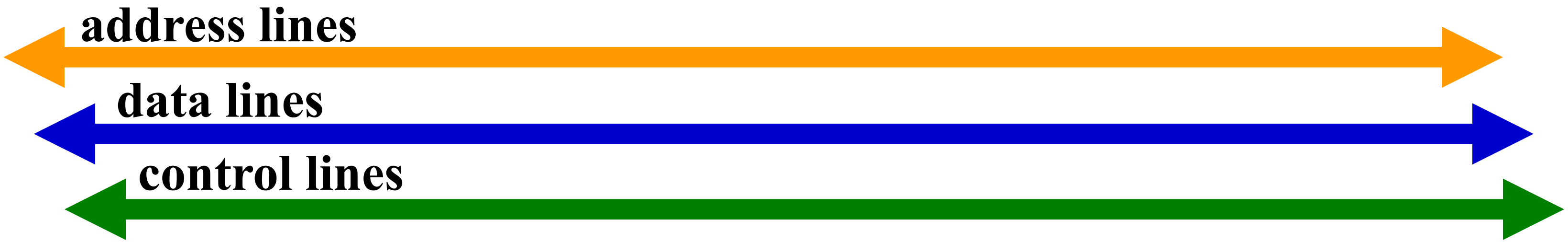
Bus Terminology



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

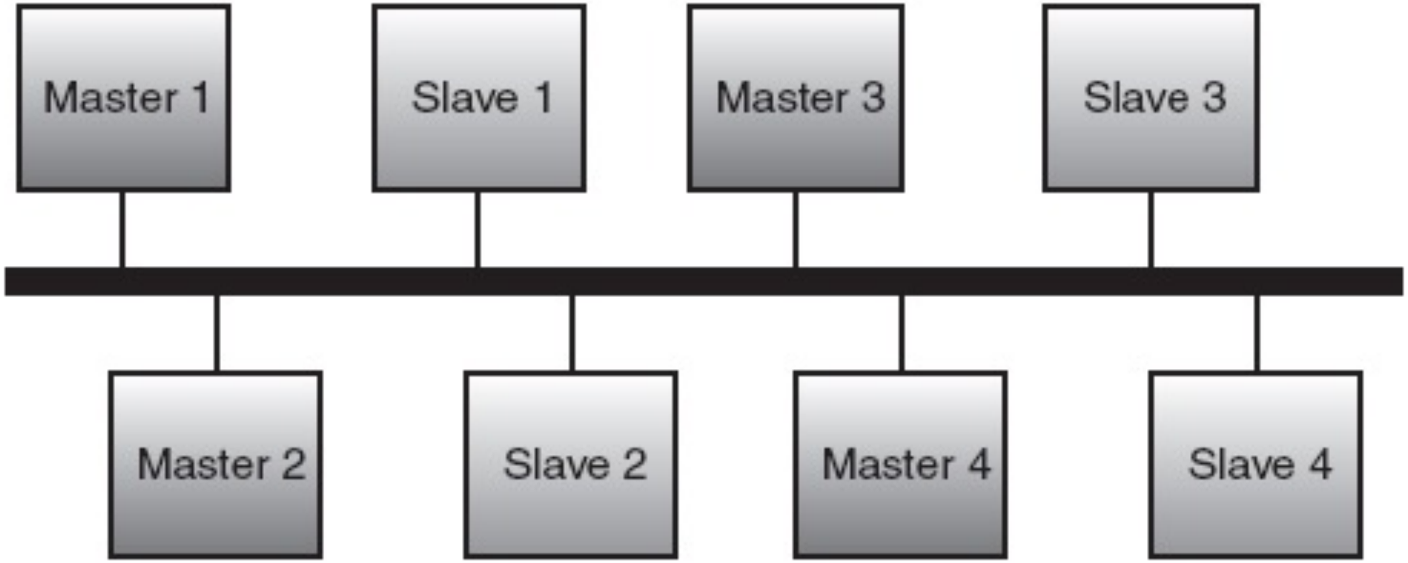
Bus signal lines



- 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, …

Types of Bus Topologies

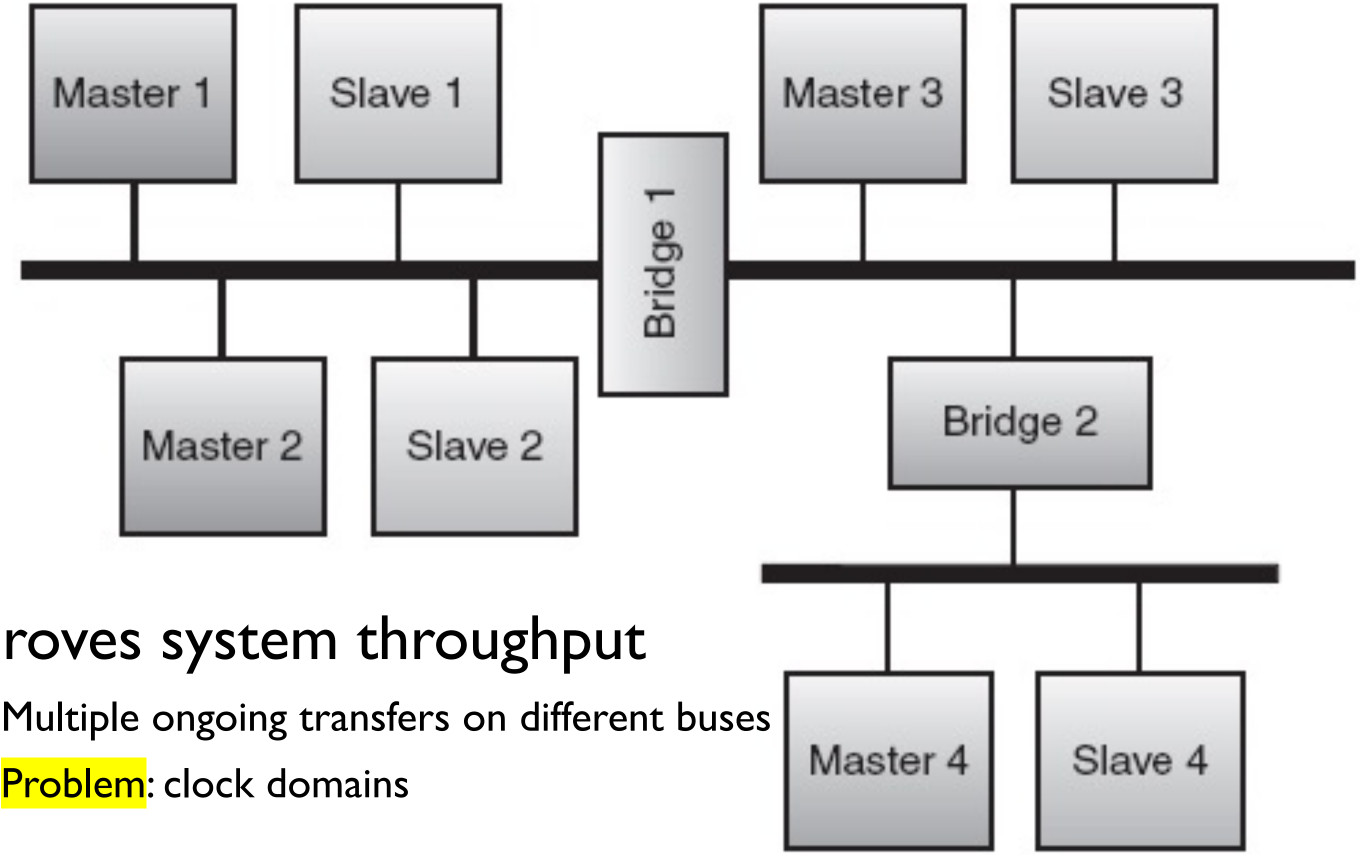
- Shared bus



Example: DaVinci family of digital video processing (TEXAS)

Types of Bus Topologies

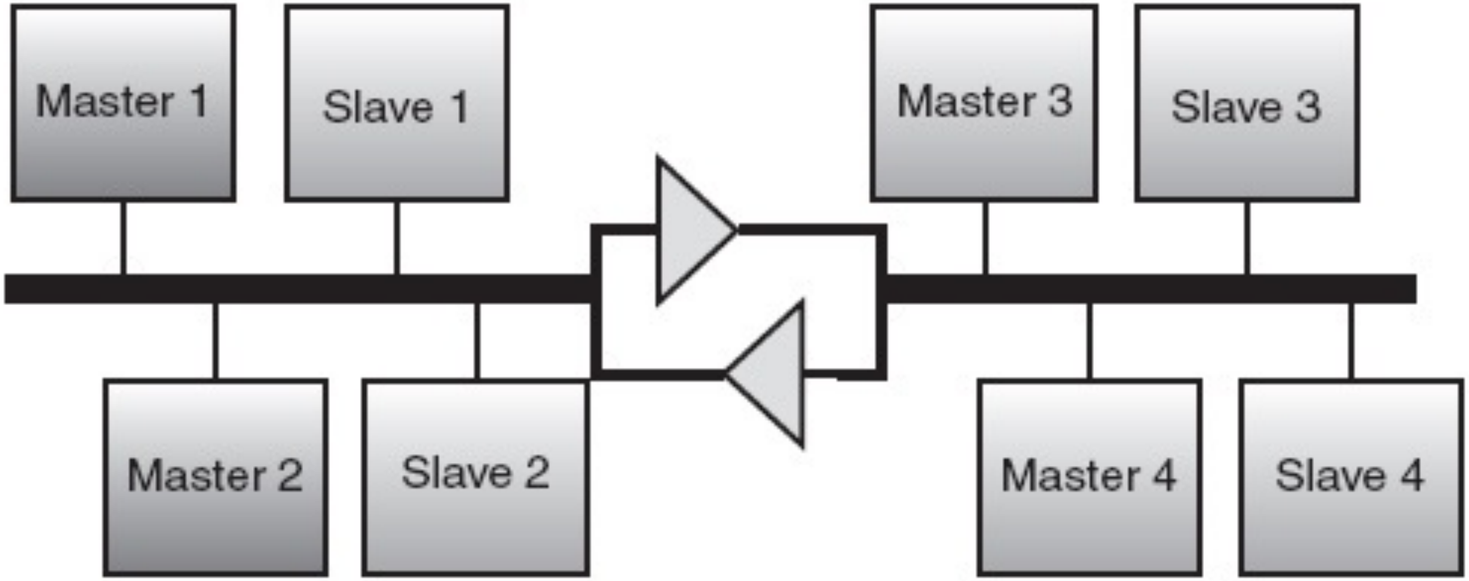
- Hierarchical shared bus



- Improves system throughput
  - Multiple ongoing transfers on different buses
  - Problem: clock domains

Types of Bus Topologies

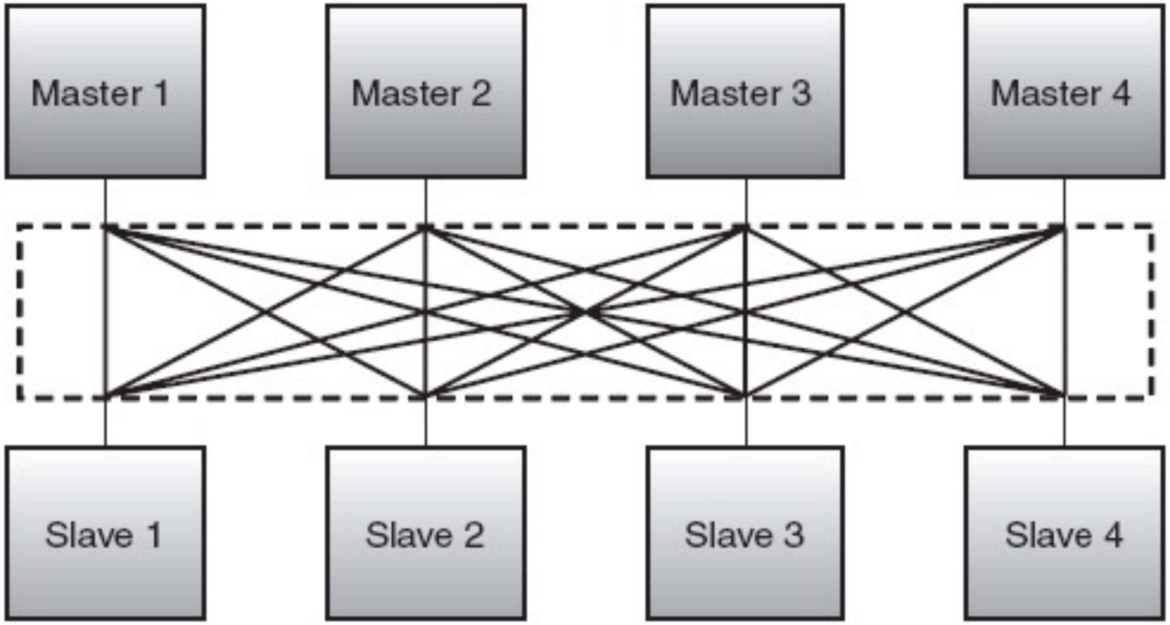
- Split bus



- Reduces impact of capacitance across two segments
- Reduces contention and energy
- Simpler than hierarchical bus

Types of Bus Topologies

- Full crossbar/matrix bus (point to point)

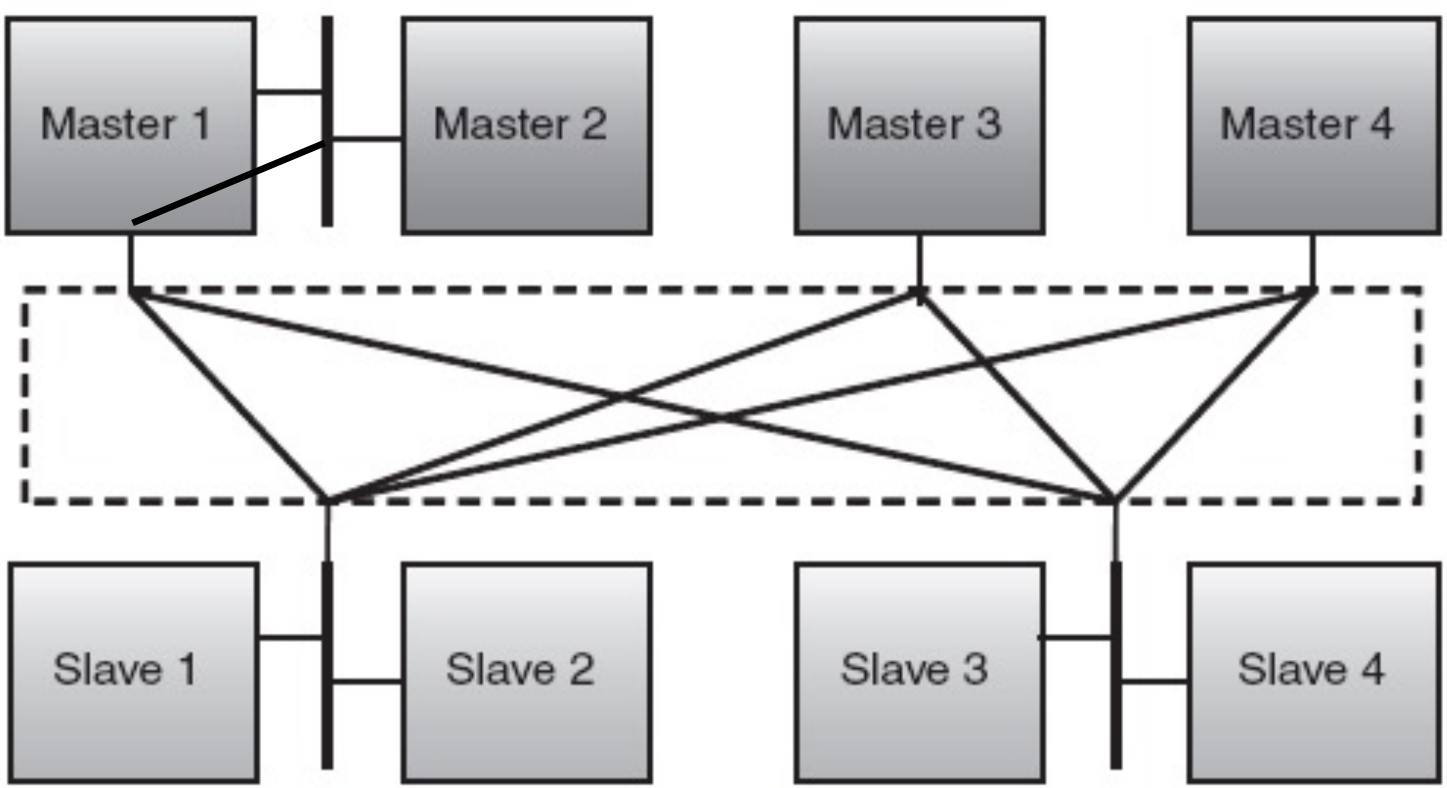


PROBLEM: 1 arbiter/decoder per slave

Example: Niagara MPSoC from SUN – 8 Sparc / 2 L2 / IO bridge / FPU

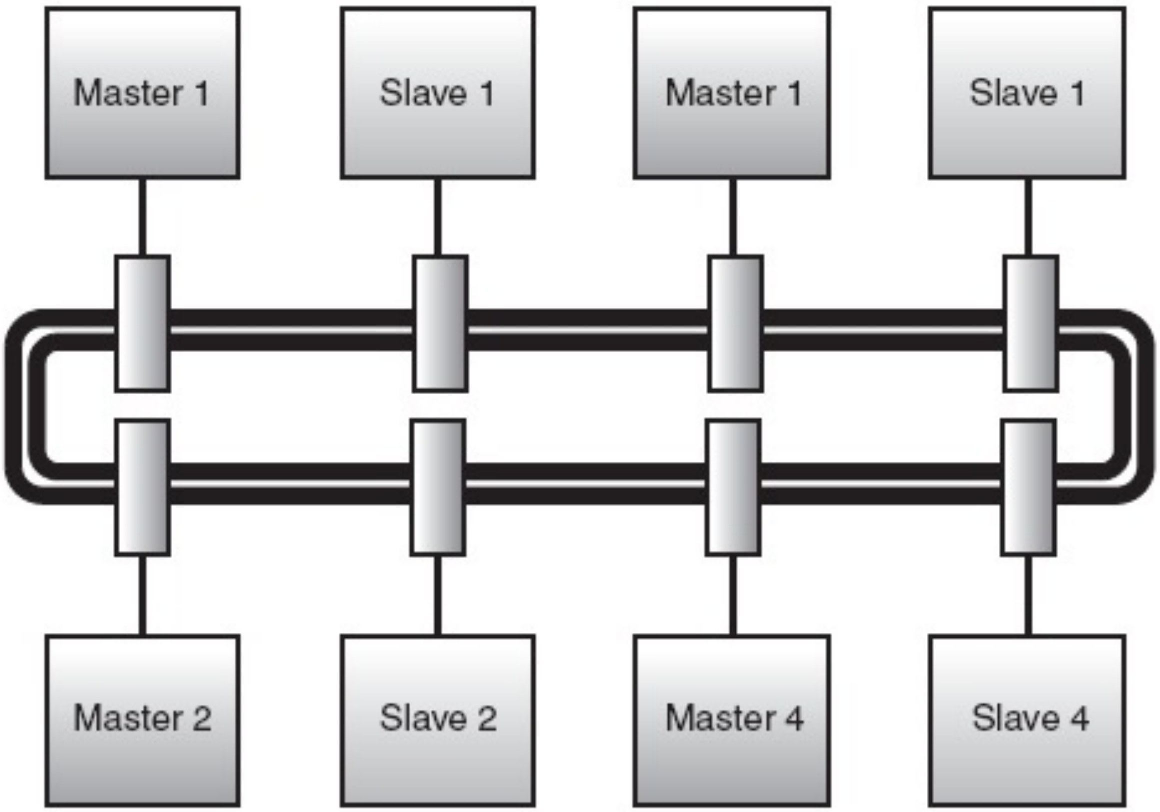
Types of Bus Topologies

- Partial crossbar/matrix bus
  - Clustering of components
  - Reduce complexity of the full crossbar



Types of Bus Topologies

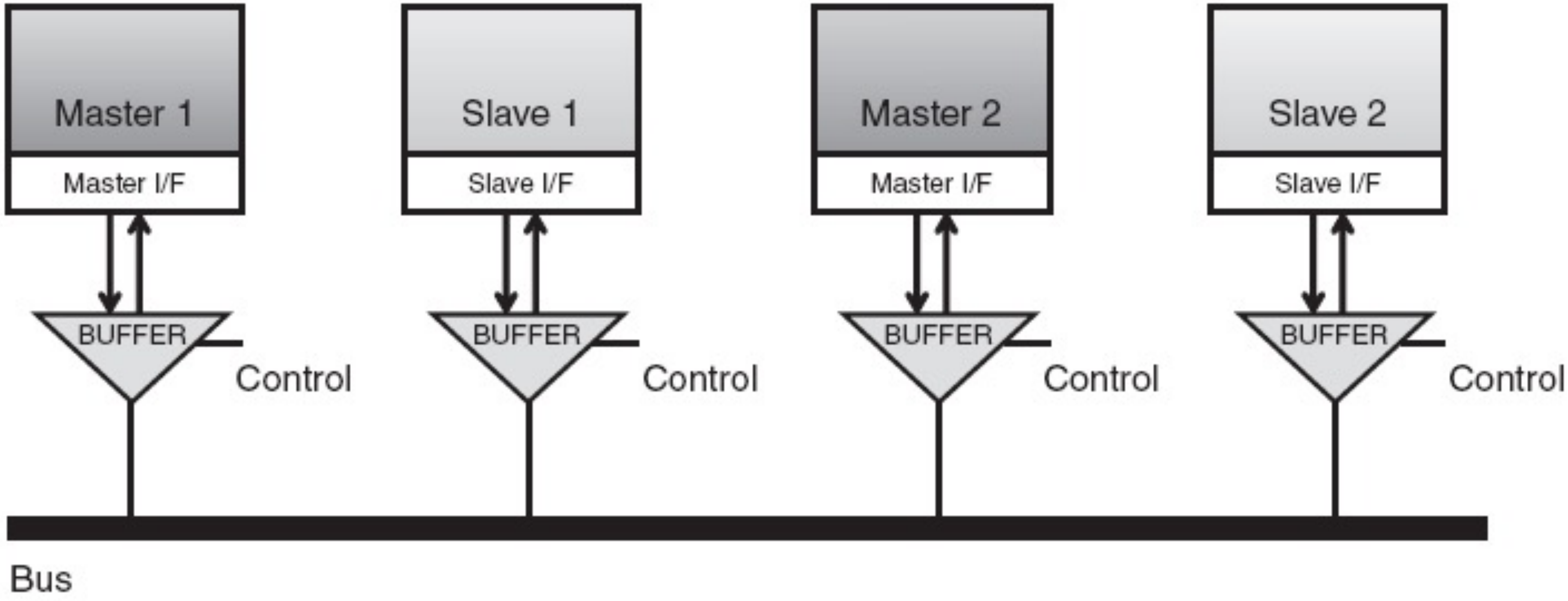
- Ring bus



Example: IBM Cell processor

Bus Physical Structure

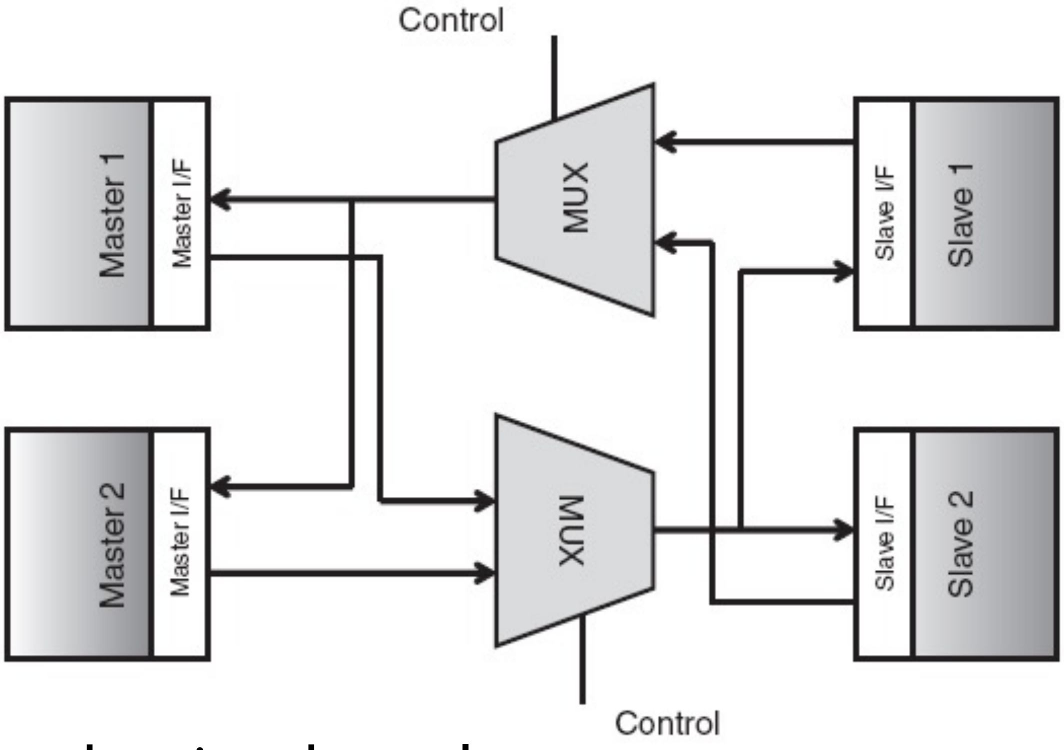
- tri-state buffer based bidirectional signals



- 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

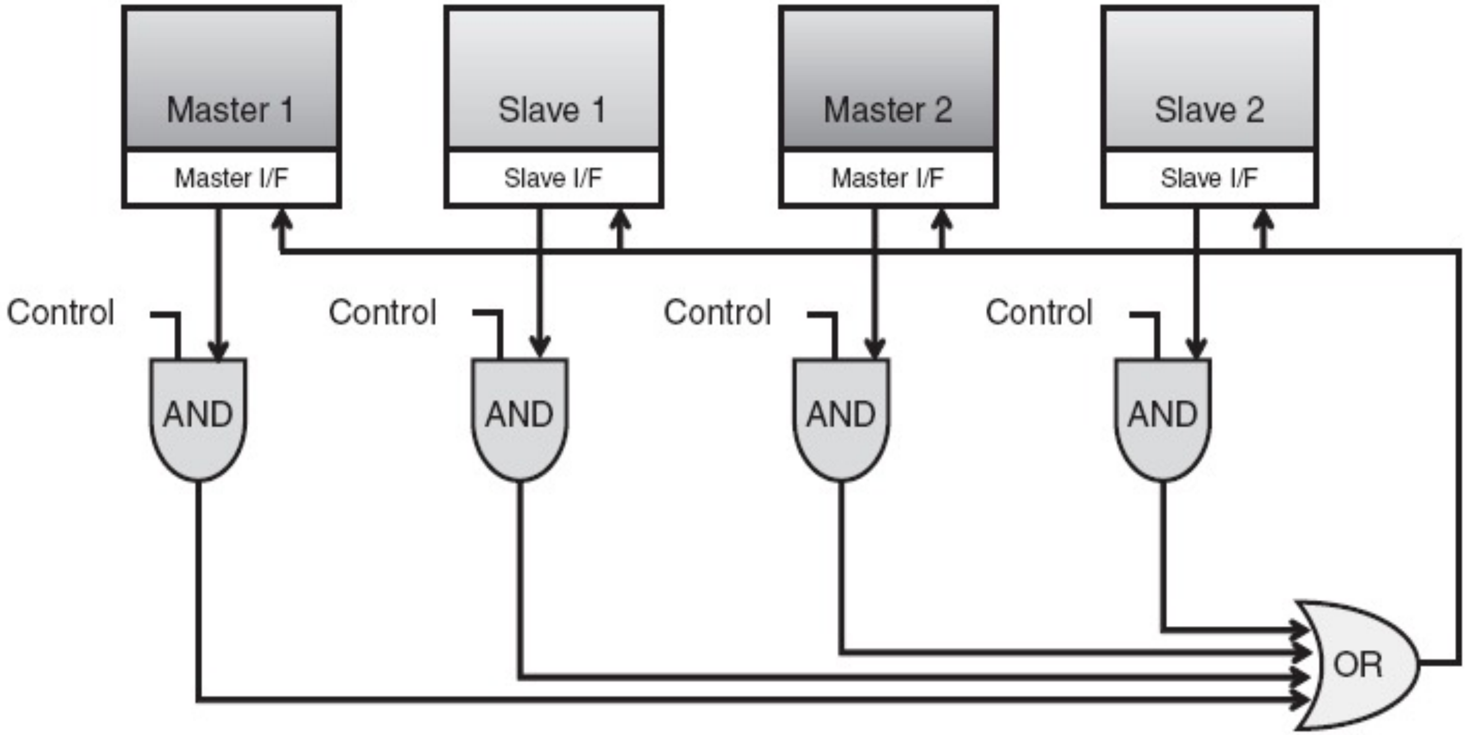
- MUX based signals



- Separate read, write channels

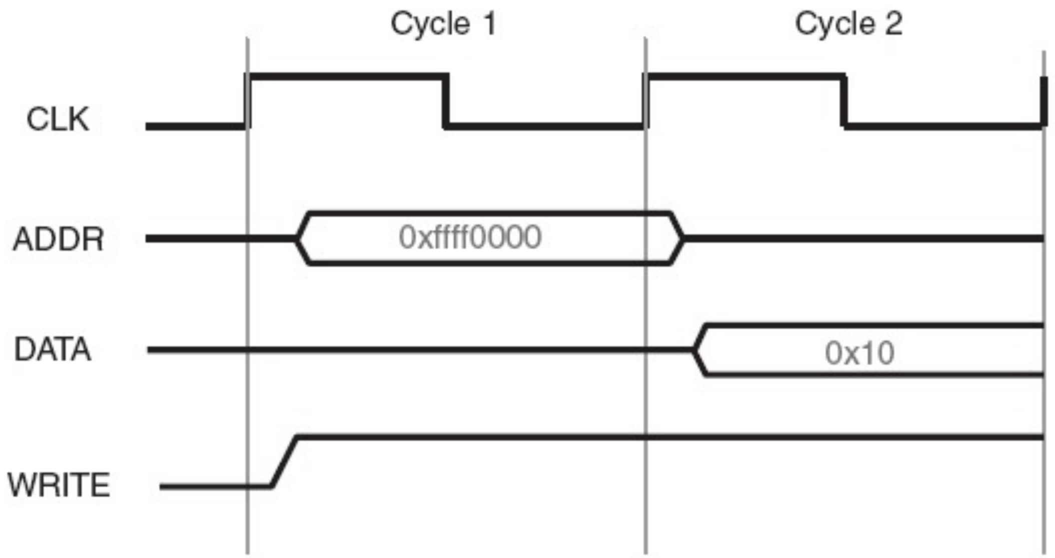
Bus Physical Structure

- AND-OR based signals



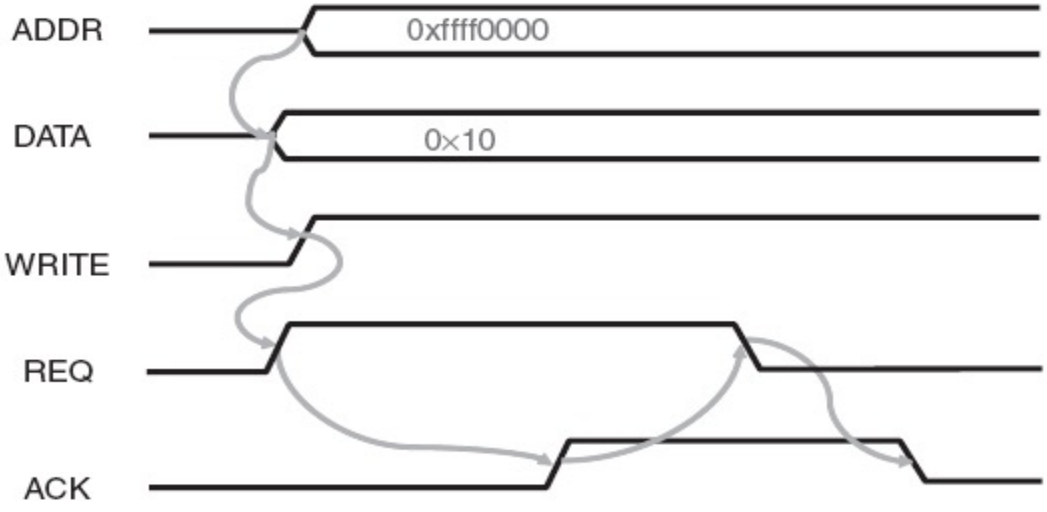
Bus Clocking

- 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
    - Processors are faster than busses



Bus Clocking

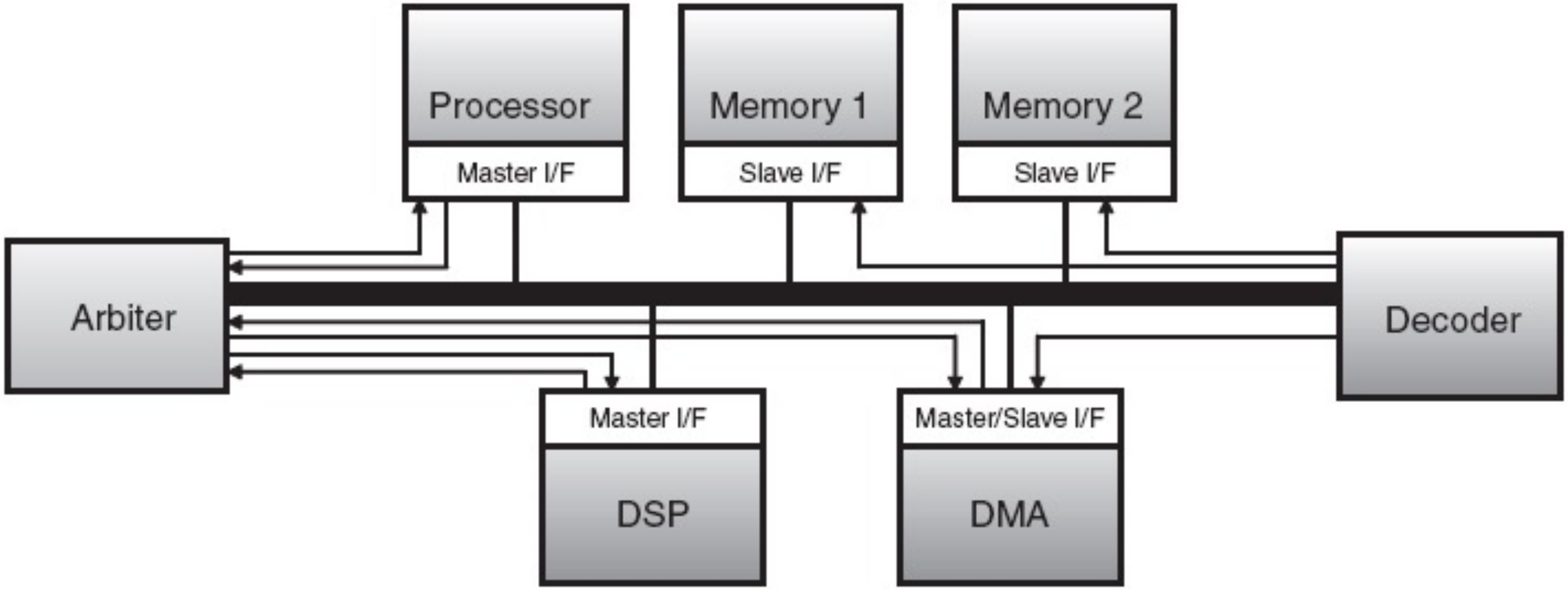
- Asynchronous Bus
  - Not clocked
  - Requires a handshaking protocol
    - performance not as good as that of synchronous bus
    - no need for frequency converters, but does need extra lines
  - Does not suffer from clock skew like the synchronous bus



Decoding and Arbitration

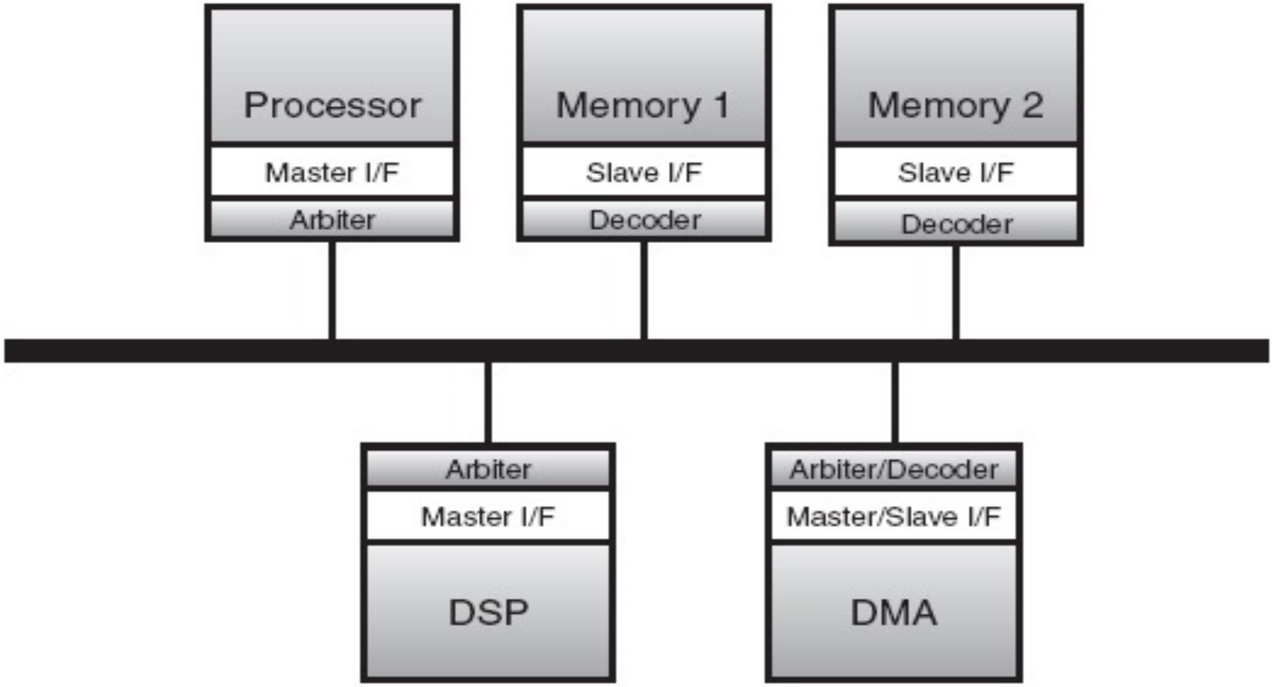
- Decoding
  - determines the target for any transfer initiated by a master
- Arbitration
  - decides which master can use the shared bus if more than one master request bus access simultaneously
- Decoding and Arbitration can either be
  - *centralized*
  - *distributed*

Centralized Decoding and Arbitration



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

Distributed Decoding and Arbitration



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

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
- 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

Arbitration Schemes

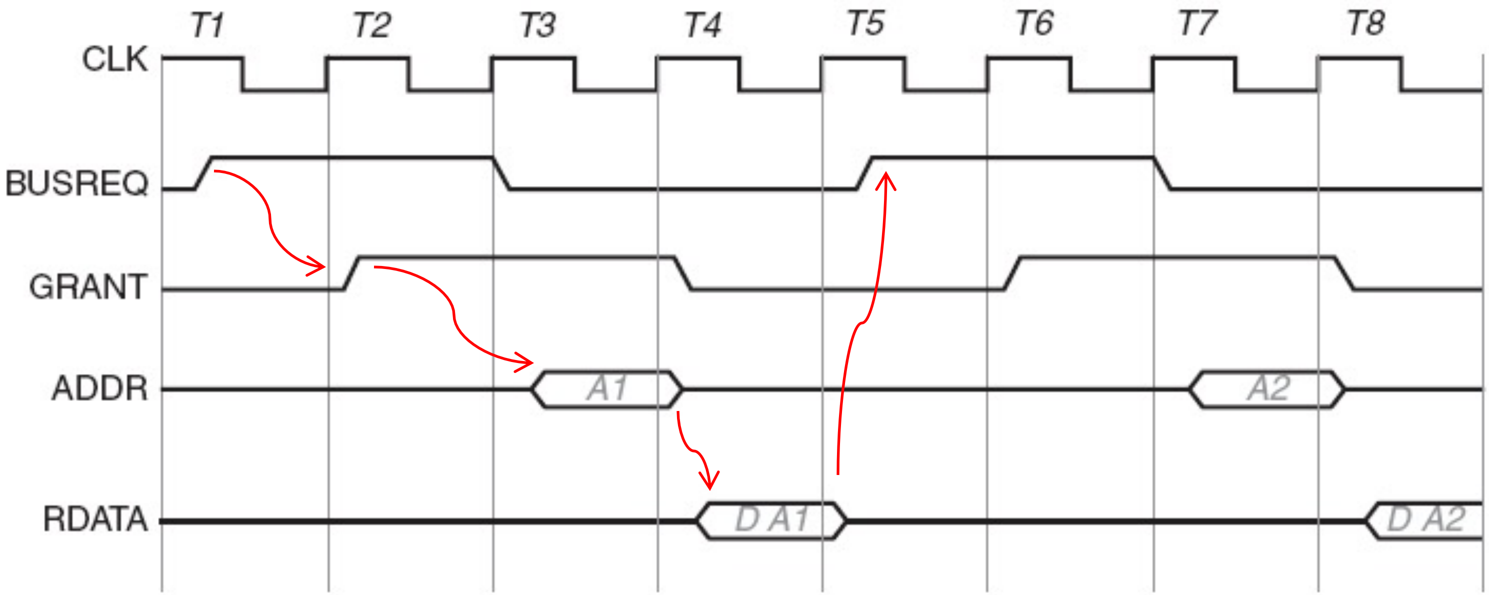
- TDMA
  - Time division multiple access
  - Assign slots to masters based on BW requirements
  - If a master does not have anything to read/write during its time slots, leads to low performance
  - Choice of time slot length and number critical
- TDMA/RR
  - Two-level scheme
  - If master does not need 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

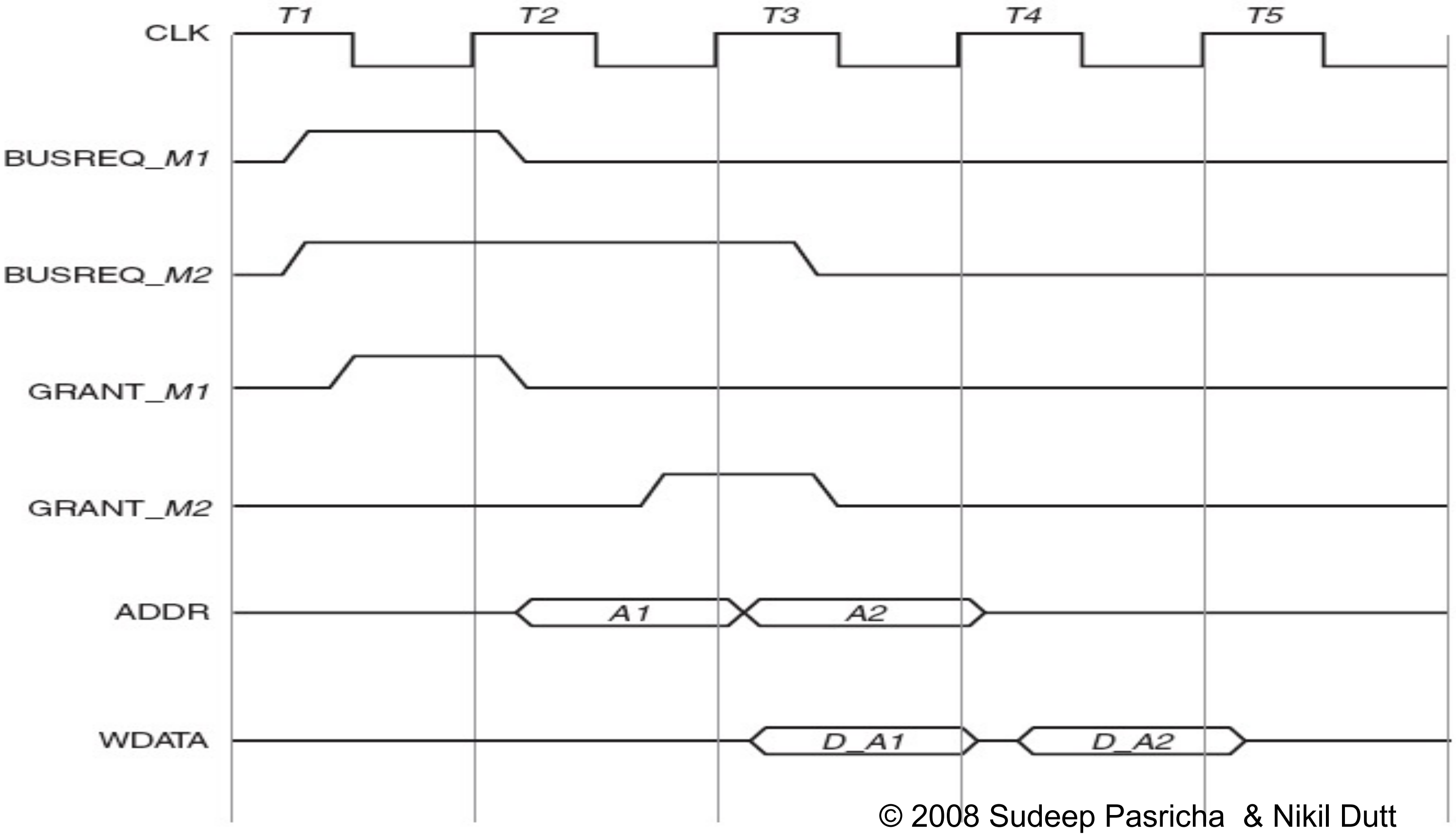
Bus Data Transfer Modes

- Single Non-pipelined 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



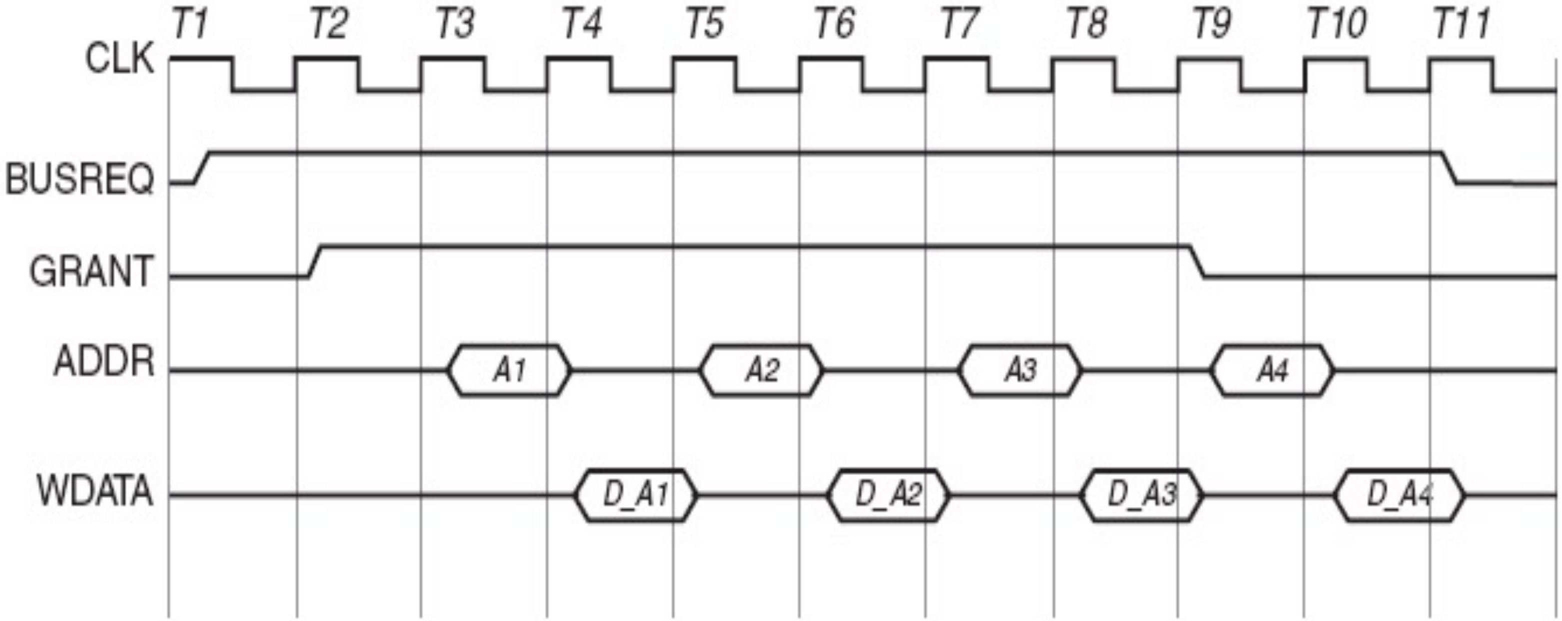
Bus Data Transfer Modes

- Pipelined Transfer
  - Overlap address and data phases
    - Only works if separate address and data busses are present



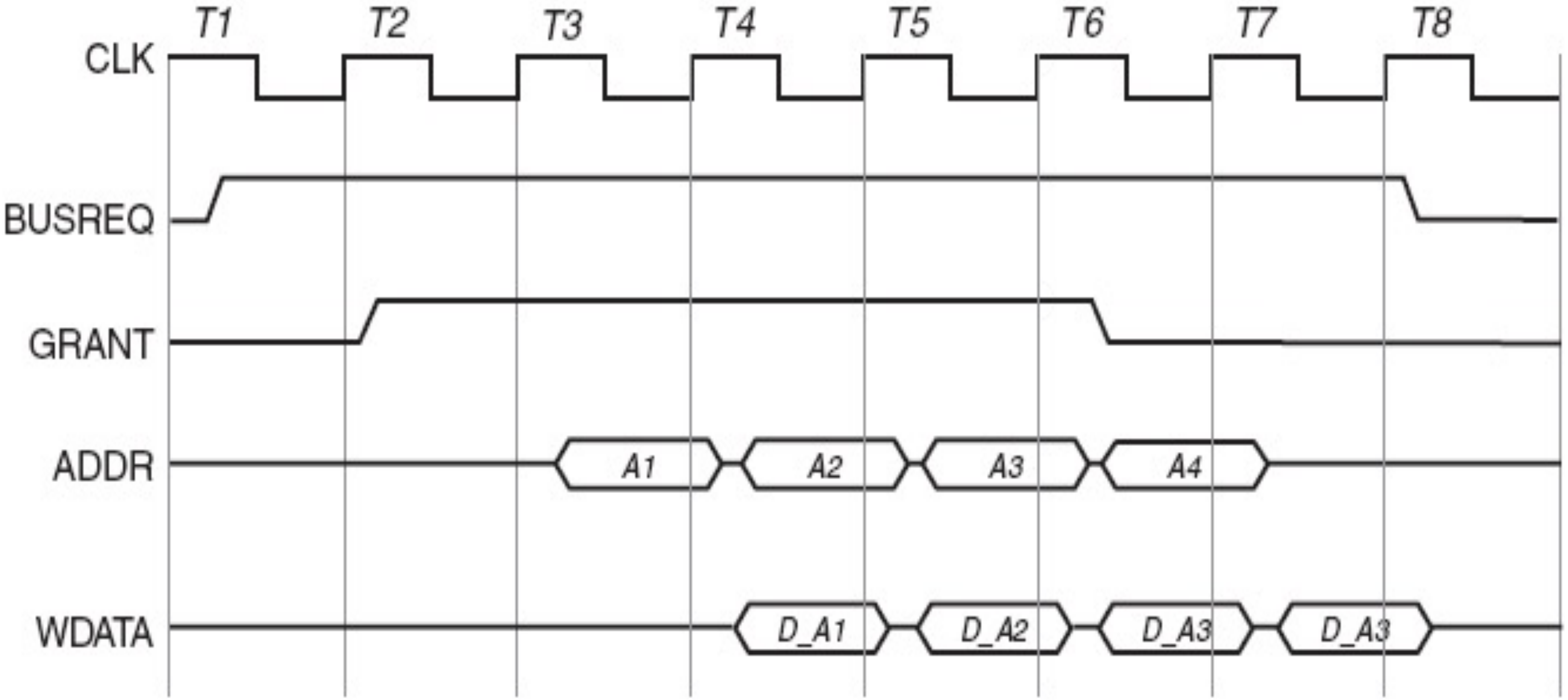
Bus Data Transfer Modes

- 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

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



© 2008 Sudeep Pasricha & Nikil Dutt

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
  - 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

Bus Data Transfer Modes

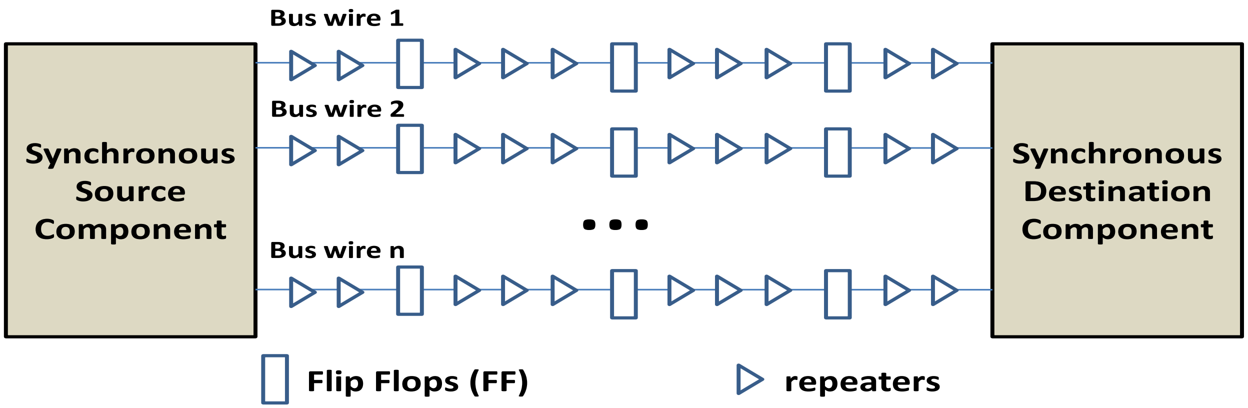
- Broadcast Transfer
  - Every time a data item is transmitted over a bus, it is physically broadcast to every component on the bus
  - Useful for snooping and cache coherence protocols
  - Example: when several components on bus have a private cache fed from a single memory, a problem arises when the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

Physical implementation issues for bus wires

- Throughput = bus width * clock frequency
- 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



Bus wire 1
Bus wire 2
Bus wire n
Synchronous Source Component
Synchronous Destination Component
Flip Flops (FF)
repeaters

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

Buses in the DSM era

- With CMOS process technology scaling below 90 nm, SoCs have entered the 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

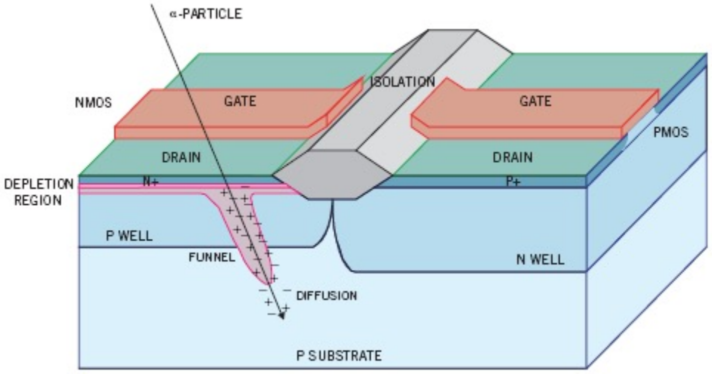
- Crosstalk
  - Phenomenon of noise being caused on a signal A due to the coupling with another signal B
    - due to the close proximity of bus wires
    - near-field electromagnetic coupling causes inductive and capacitive crosstalk on bus signals
  - Even when wires are far apart, crosstalk can still 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

- 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

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