CSCU9V4 Systems

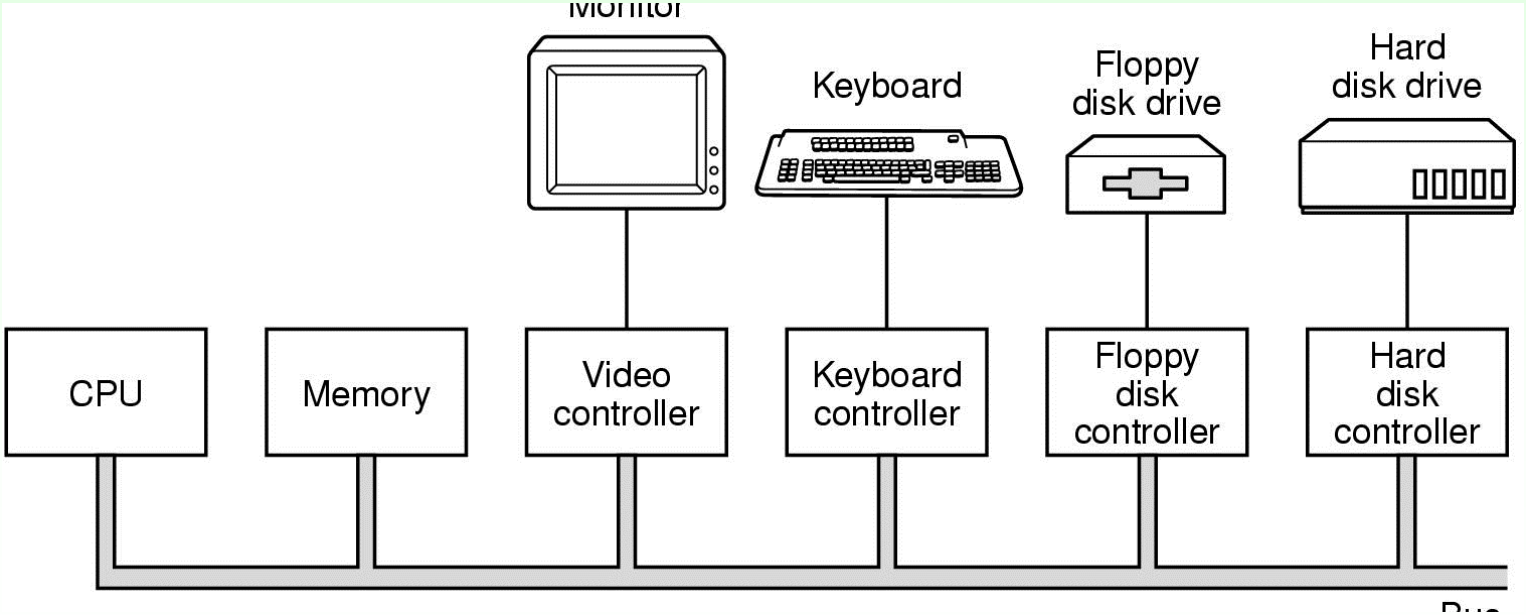
Systems Lecture 8

Computer Organisation 3

Buses

Buses

- Origin: same as omnibus - meaning 'for all' (Latin)
- High-speed data link between all the devices attached to the bus.
- Connect I/O controllers to the memory
- Used by the CPU to access memory
- Possible conflict if two (or more) devices and/or the CPU want to access the bus - requires Bus Arbitration to choose Bus Master

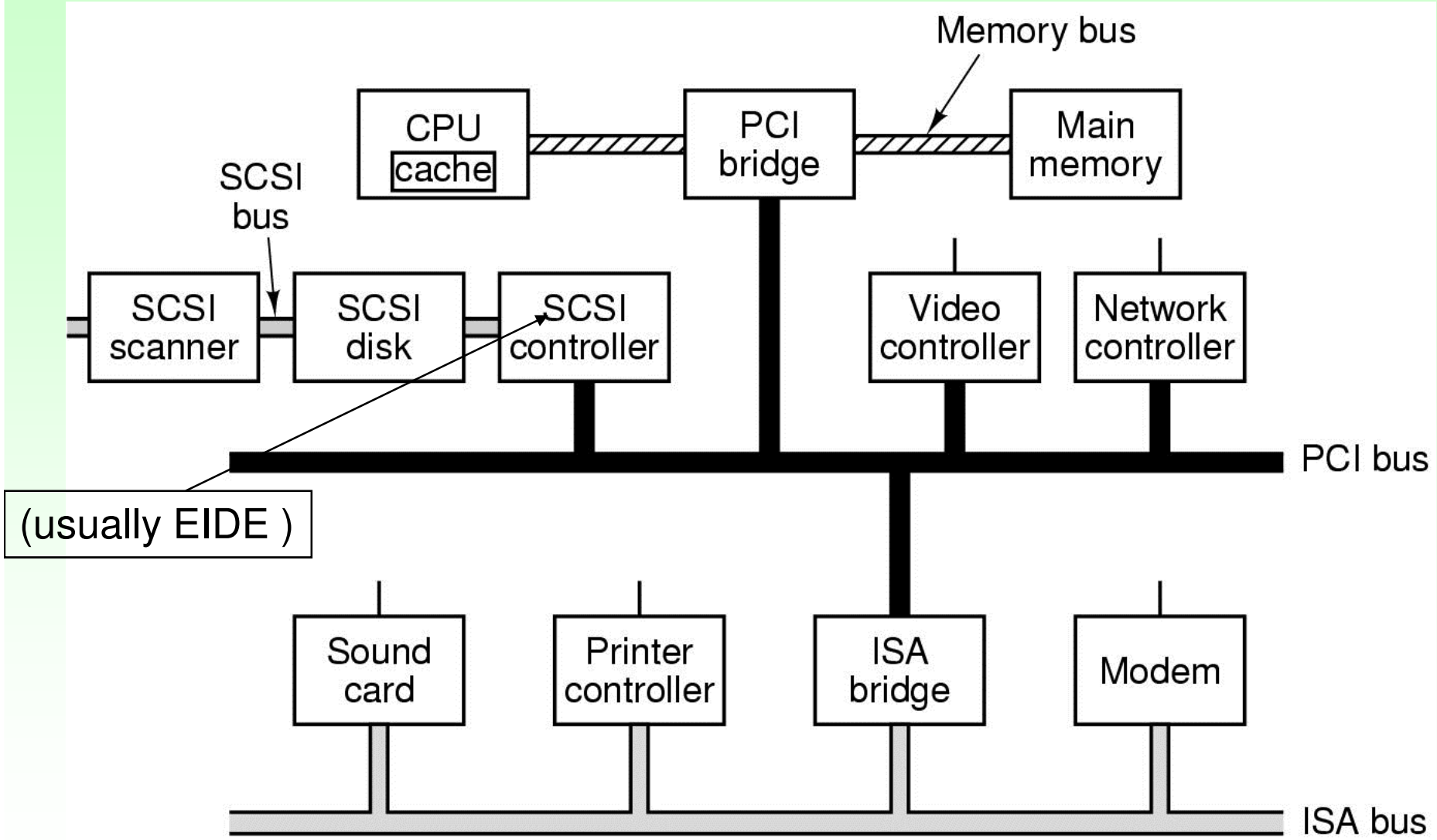


Development of Buses

- Single bus design was common for early computers and worked fine
- Usually a faster bus was designed for the next system
- Developing a new bus for every new PC model is impractical
- Old IBM PC bus is called ISA (Industry Standard Architecture)

- Machines with multiple buses were developed
  - CPU is often connected to memory by a dedicated high-speed bus
    - So that CPU traffic does not affect other buses
  - PCI (Peripheral Component Interconnect) used for most other devices
    - High-bandwidth peripherals are connected to the PCI bus
  - ISA or EISA bus can be used by older devices (may be absent entirely in new machines)
    - PCI has a connection to the ISA bus if it is present
  - Modern machines use SATA and PCIe.

Example Architecture (old and simple)



Different Buses

- Buses are inside a shielded case to minimise electrical interference
  - Transmission is error-free
- Buses inside the CPU may be of any kind (and are not studied here!)
  - Do not interface with the outside world
- External buses must adhere to well-defined rules on how the bus works
  - Allow 3rd parties to develop peripherals and controllers
  - Rules are called the bus protocol
    - Additionally there are electrical and mechanical specifications
- Some example buses:
  - Multibus (8086)
  - IBM PC bus (PC/XT)
  - ISA bus (PC/AT)
  - EISA bus (80386)
  - Microchannel (PS/2)
  - SCSI (PCs and workstations)
  - Nubus (Macintosh)
  - Universal Serial Bus (PCs)
  - FireWire (consumer electronics)
  - VME bus (physics lab equipment)
  - PCI bus (PCs)
  - PCI Express bus (newer PCs)

Master and Slave

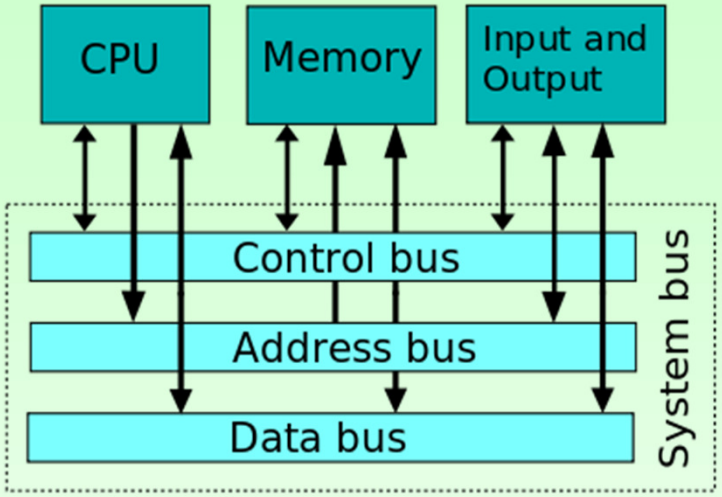
- Some devices attached to buses are active and can initiate transfers
- Other devices are passive and wait for requests
- Active devices = Masters
- Passive devices = Slaves
- Example:
  - CPU is bus master, and orders the disc controller to read a block
  - Disc controller becomes bus master, and transfers the block from disc to memory
- Master and slave associations may change and depend on particular activities, however, memory can never be master

Master	Slave	Example
CPU	Memory	Fetching instructions and data
CPU	I/O device	Initiating data transfer
CPU	Coprocessor	CPU handing instruction to Coprocessor
I/O device	Memory	Transfer data to memory
Coprocessor	CPU	Coprocessor fetching operands from CPU

Bus Outline

- The bus consists of a number of parallel conductors

- These are used to specify
  - Memory address (or peripheral being accessed)
  - Direction the transfer is running in
  - Timing of the transfer
  - Data to be transferred



From Wikipedia

- Often separate sets of conductors are used for these various purposes:

Line Name	Symbol	Function
Address lines	$A_0..A_{NA}$	Specifying the cell being accessed (same width as MAR)
Data lines	$D_0..D_{ND}$	The data being transferred (same width as MDR)
Read/Write	R/W	Direction of transfer (wrt CPU)
Clock	f	Timing of the transfer

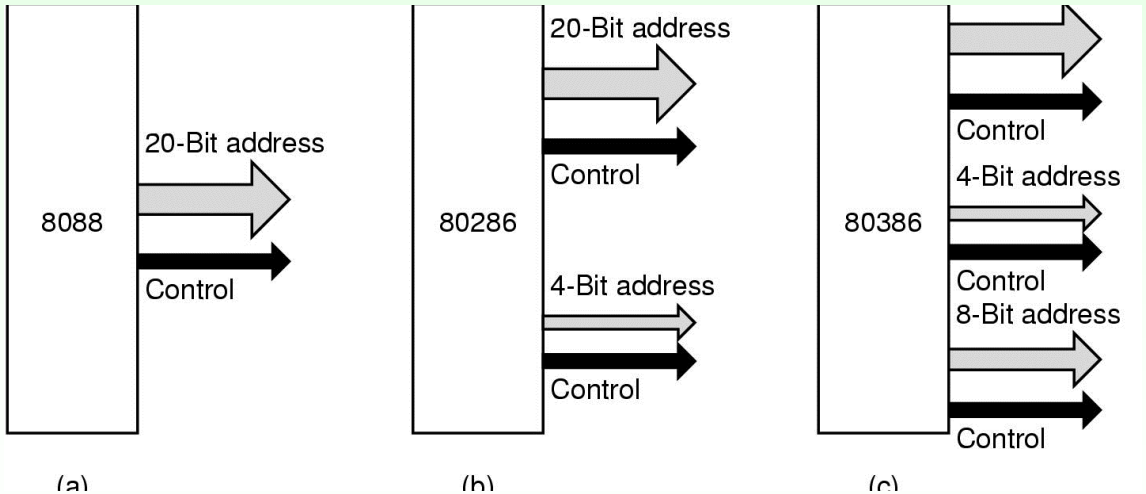
Bus Outline

- Bus design is a complex problem
- We will provide an overview here
- Principal bus design issues include
  - Bus width
  - Bus clocking
  - Bus arbitration
  - Bus operations

- Look at them in turn …

Bus Width – Address lines

- The more address lines are available the more memory can be addressed directly
- Bus with n address lines → CPU can address 2^n different memory cells
- Large memories → many address lines
  - 32 bits: 4Gcells (usually 4Gbytes)
- Wide buses require more wires and also take up more space
- More expensive! → memory size vs. cost
- Needs to be considered if the memory is to be extendible later



Bus Width – Data lines

- How to increase the data bandwidth of a bus
  - Use more data lines
  - Decrease the bus cycle time (more transfers/sec)
  - Increase the data bus width (more bits/transfer)
- Speeding up the bus is difficult (signals travel at different speeds on different lines)
  - Electrical problems increase with increasing bus speed
  - Old devices can't work at higher speeds
- Hence, usual approach to improve performance is to increase data width
  - Add more data lines
- Another option: multiplexed busses (use signal lines for more than one purpose)
  - Cost vs speed

Mode of Transfer (Control Lines)

- Read or Write
  - R/W line

- There may be additional lines specifying the size of the transfer:
  - the bus can transfer 1 word (i.e. using all data lines) at a time, but sometimes it may transfer less:

8 bits	Byte
16 bits	Short
32 bits	Word
64 bits	Long word

- Some buses also have 'burst mode' transfers:
  - for example, several contiguous words of memory may be transferred, one after the other from memory to CPU
- Interrupts…see later

Timing of Transfer

The control signals specify the times ...

... when the processor (or memory or peripheral devices)

- may place data on the bus,

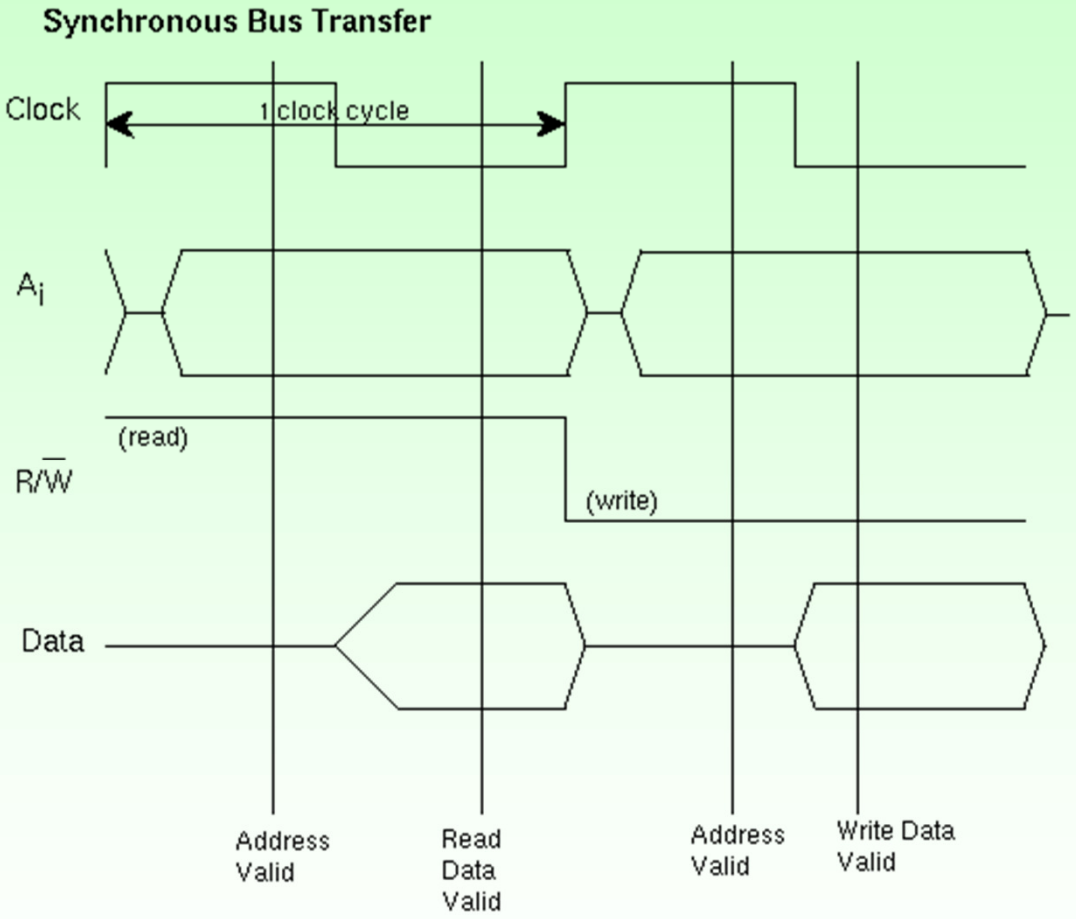
... or

- when the data on the bus may be assumed to be valid.

- Buses can be grouped into two main categories depending on clocking technique
- **Synchronous buses** have a "master clock"
  - Special line driven by a crystal oscillator (very precise timing)
  - Signal on the line is a square wave between 5 and 1GHz
  - All bus activities take a (fixed) integral number of these cycles (bus cycles)
- **Asynchronous buses** have no "master clock"
  - Bus cycles can be of any length
  - May differ between various pairs of devices
  - (bus still has a clock line)

Synchronous Bus Transfer

- Figure shows one read and one write cycle.
- Each takes place in exactly one clock cycle.



A more complex example (Tanenbaum)

- 40MHz clock, - memory read: 40nsec
- bus cycle of 25 nsec,

CPU done

Can start again

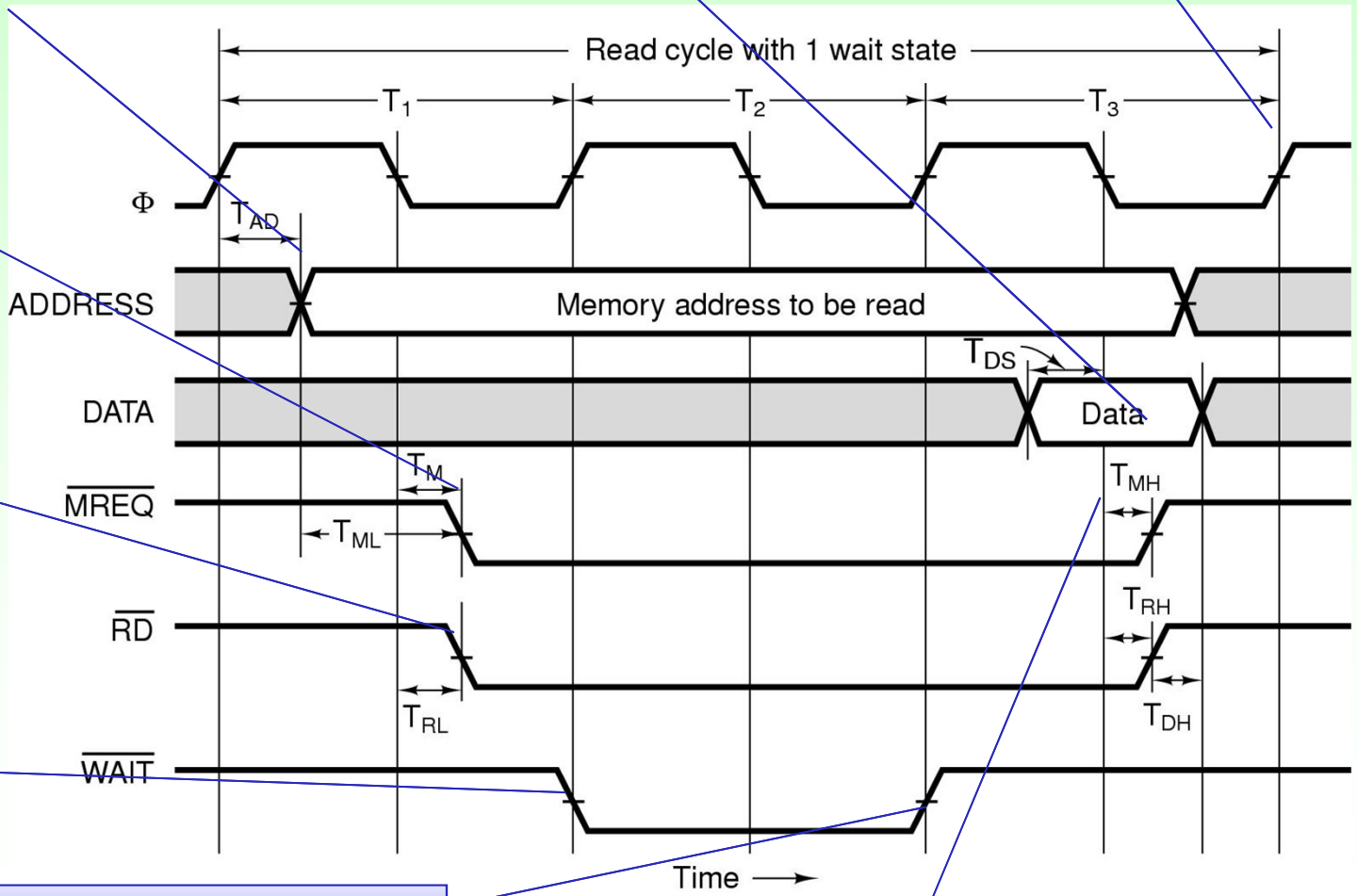
After start of T1 CPU puts address on address bus

Once address lines settled, CPU asserts for memory access.

Once address lines settled, CPU asserts for memory *read*.

Memory not ready - tell CPU

Memory ready on *next* -ve edge - tell CPU



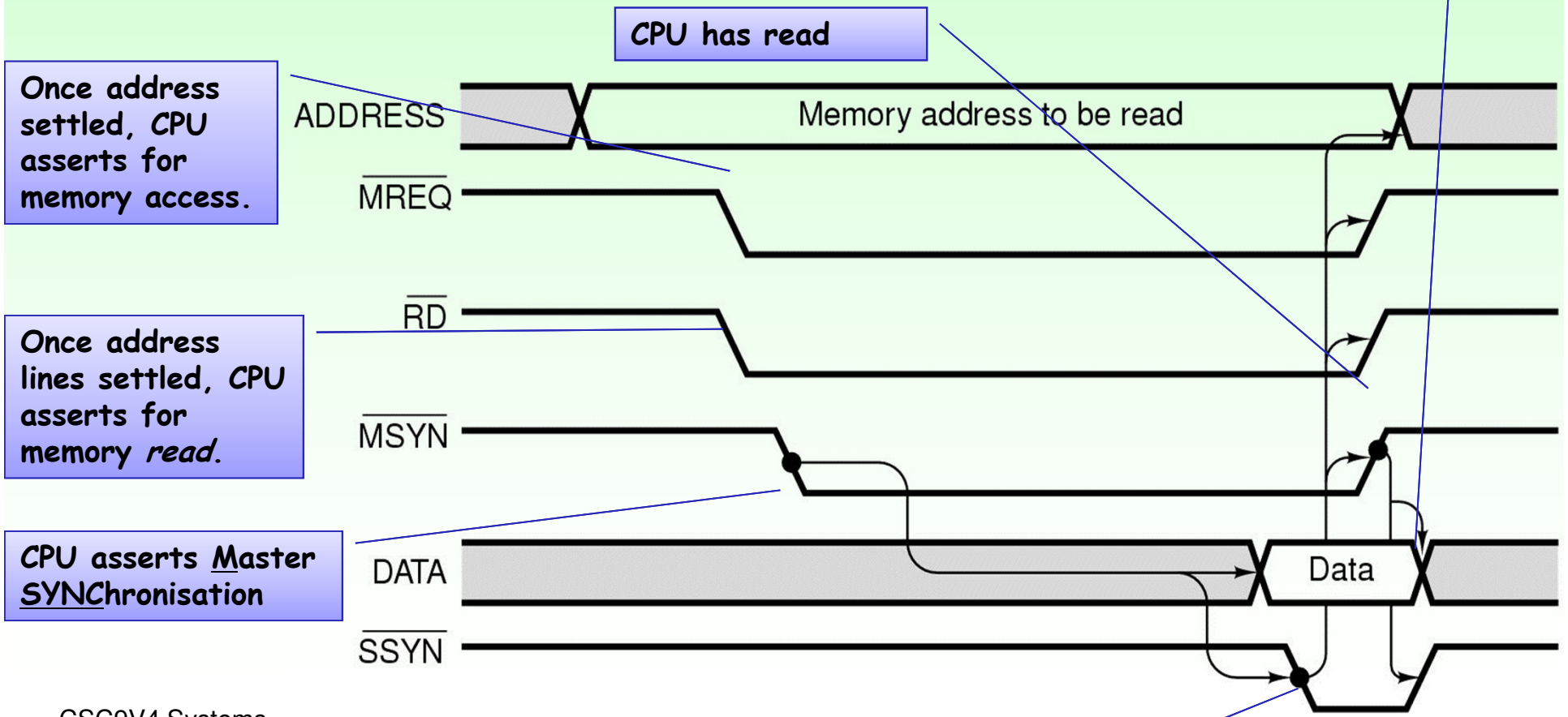
CPU reads data

Problems of Synchronous Buses

- All devices on the bus must be capable of operating at the bus clock speed
- The bus needs to run at the speed of the slowest component on the bus
  - Implies wait states
- Making the whole system run rapidly then entails using fast components for everything on the bus.
- Everything works in multiples of the bus clock
  - If CPU and memory are capable of completing a transfer in 3.1 cycles, it is stretched to 4 cycles → no fractional cycles
  - Difficult to take advantage of future improvements of technology
    - What if new memory with an access time of 20 nsec becomes available?
    - Or with 10 nsec?
    - No good handling of heterogeneous collection of devices
  - Good PC hardware designers account for this with memory and device selection to maxmimise throughput

Asynchronous Buses

- No master clock!
  - additional signals to control the transfer of information
- In this way, different bus interfaces can be run at different speeds.
- MSYN and SSYN lines



Full Handshake

- A set of signals that interlocks as MSYN and SSYN
  → Full Handshake

- MSYN is asserted
- SSYN is asserted in response to MSYN
- MSYN is negated in response to SSYN
- SSYN is negated in response to negation of MSYN

- Full handshakes are timing independent
  - Each event is caused by a prior event
  - No clock pulses involved
  - If a master-slave pair is slow, other master-slave pairs are not affected

Why Asynchronous Buses?

- Asynchronous bus transfer allows devices which run at different speeds to share one bus without a speed penalty.
- The slave can spend a variable amount of time
  - accessing/retrieving data
- Similarly, the master can spend a variable length of time
- Actions need not occur at every clock pulse.

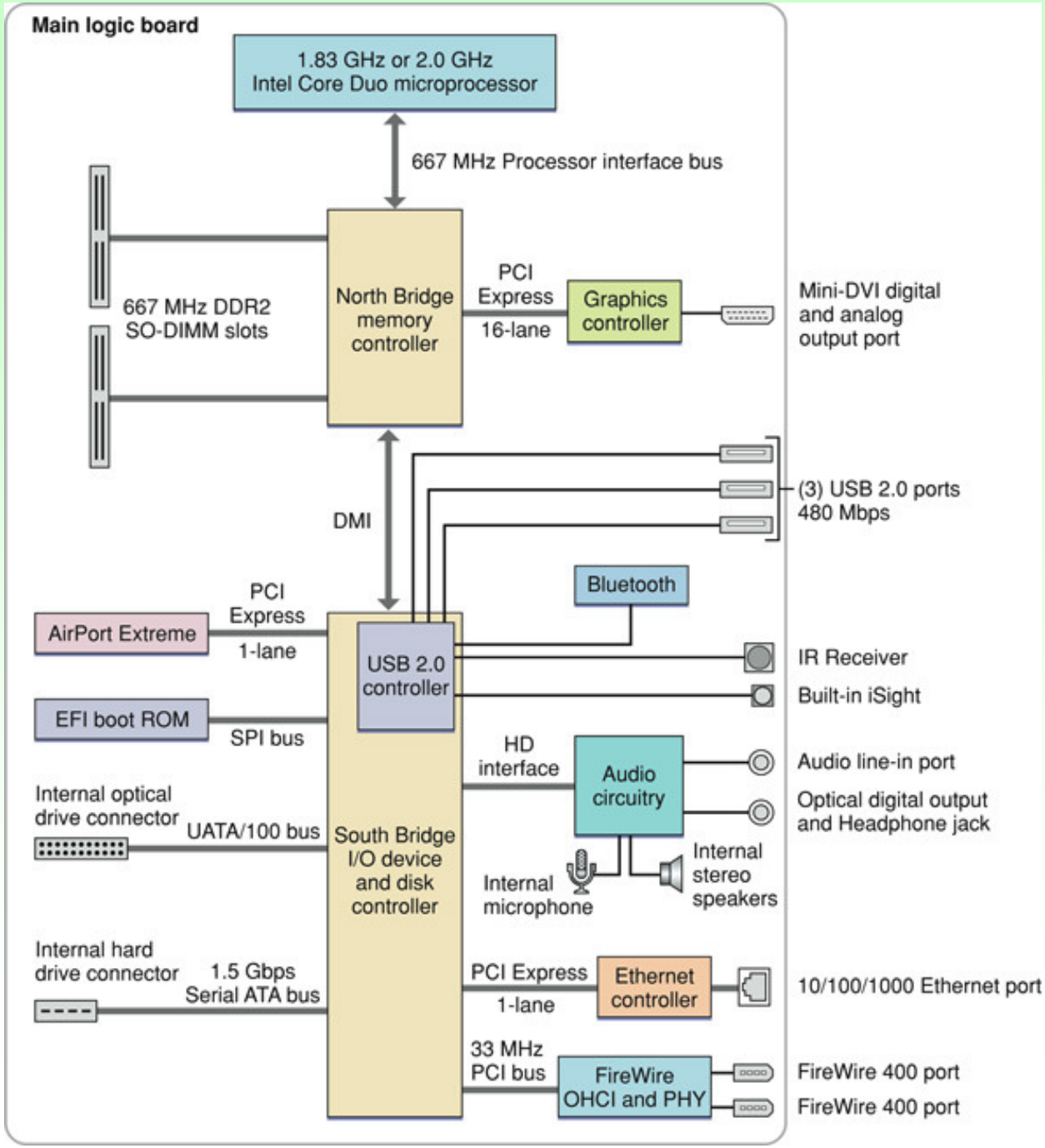
- However, most buses are synchronous buses
  - Easier to build; effective if components are chosen properly
  - Can use multiple busses to cope with fast and slow subsystems
  - Lot of investment in synchronous bus technology

Buses in Modern PCs

- Requirements
  - very fast transfer between CPU and Memory
  - fast transfer between fast peripherals and CPU/Main Memory
  - slower transfer for slower peripherals
  - slow and inexpensive transfers for slow peripherals
- CPU/Memory
  - usually a proprietary bus: very high speed

- PCI bus: High speed bus, wider and faster than ISA
- PCIe bus: high speed serial link
- USB bus: Generic 'one size fits all' (mice, keyboards, printers)
- ISA bus (out of date): Compatibility reasons

- Modern bus architechture



The PCI Bus

- Need for high speed - consider displaying a video: huge data rates required

- PCI = Peripheral Component Interconnect
- designed by Intel in 1990, open access: public domain (used by Apple as well)
- PCI transfers 32-bit or 64 bit data at 33 MHz or 66 MHz or 132 MHz

- Provides a high bandwidth link to fast peripherals.
- Bus is synchronous (mostly…)
- Data and address lines are multiplexed

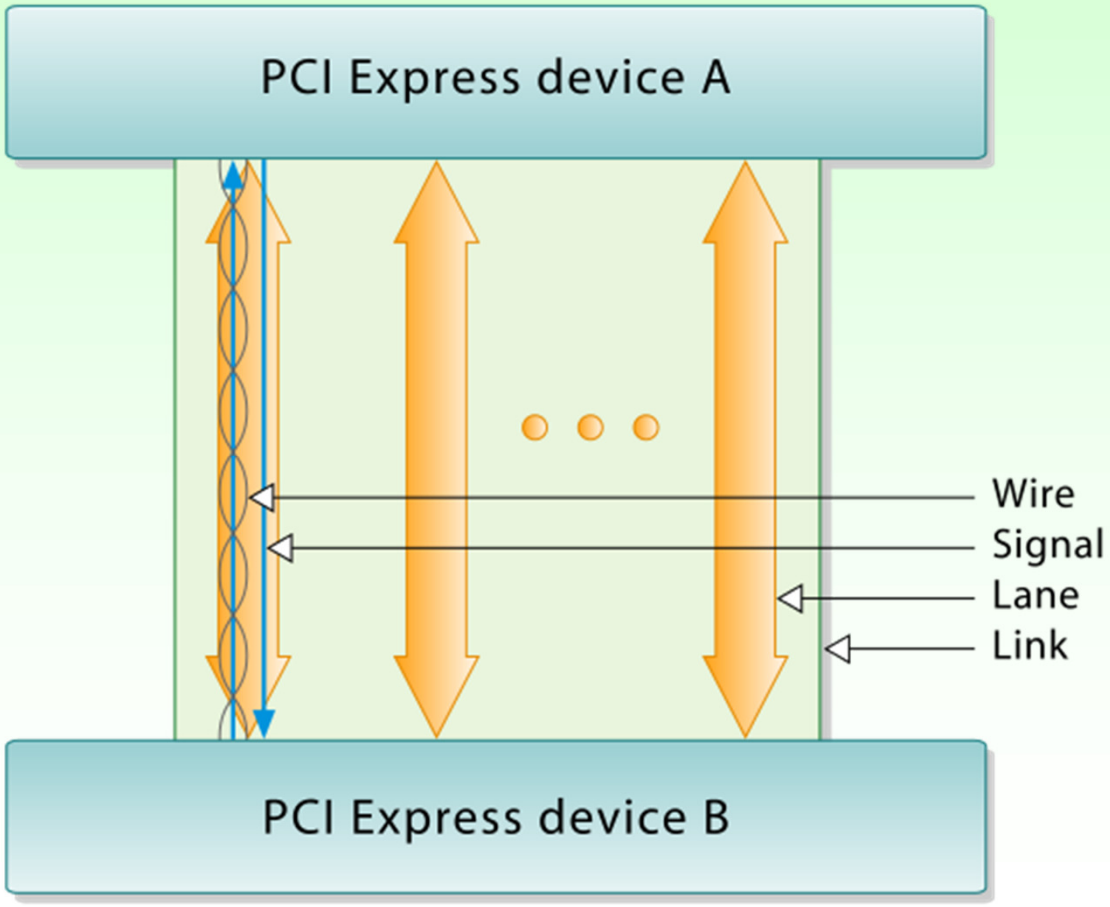
- Bus has facilities for transferring bus mastership in each cycle
  - usually requires 1 cycle between reads and writes
- Bus has also facilities for burst mode transfers
- Bus also has interrupt lines

Other buses

- PCI Express: replacement for PCI, but serial, faster and full duplex
- AGP: bus used purely for graphics devices (usually just 1 graphics card)
- Buses for discs
  - SCSI
  - EIDE
  - ATA, SATA
- Serial buses
  - FireWire
  - USB
  - Thunderbolt

PCI Express

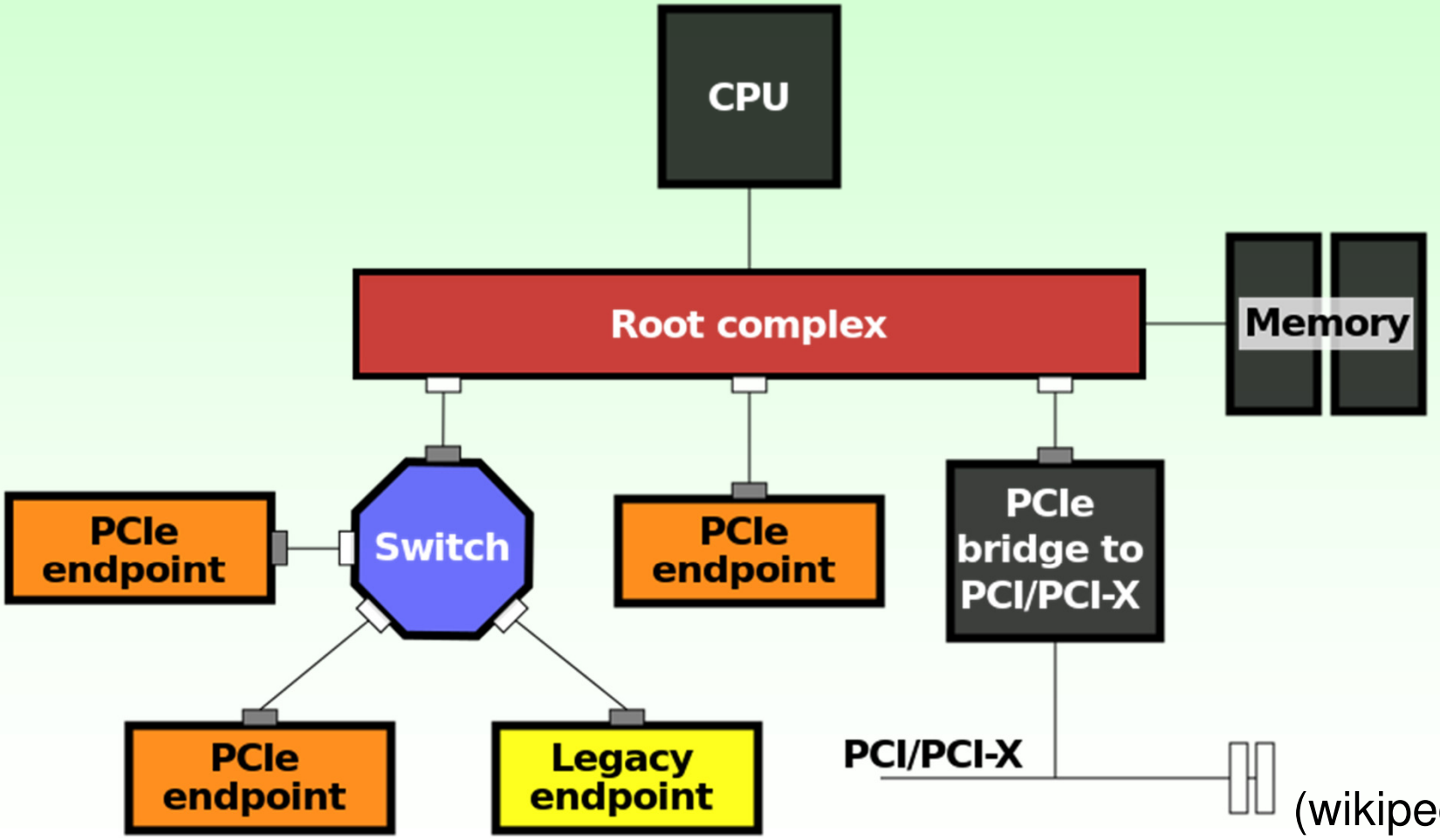
- PCIe is a serial bus that has a smaller footprint and is faster than PCI



(wikipedia)

PCI Express

- Point-to-point connections with packet-switching (like internet)



(wikipedia)

End of Lecture