System Buses
Ch 3

Computer Function

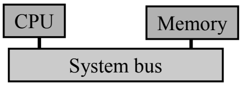
Interconnection
Structures

Bus Interconnection

PCI Bus

Computer Function

- von Neumann architecture
  - memory contains both instruction and data



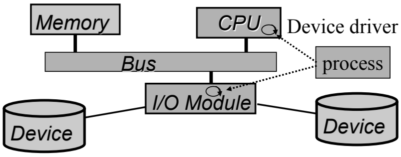
- Fetch-Execute Cycle

(käskyn nouto ja suoritus sykli)

Figs 3.3, 3.9

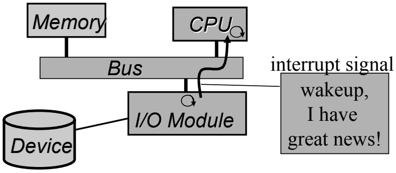
I/O control

- CPU executes instructions and with those instructions guides I/O modules
  - control and data registers in I/O modules
  - I/O modules give feedback to CPU with control and data registers, but only when CPU is reading them!



I/O Control

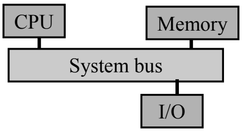
- Interrrupts allow I/O modules to give feedback to CPU even when CPU is doing something else



- DMA allows I/O modules to access memory without CPU's help

von Neumann Bottleneck

(von Neumann pullonkaula)



- All components communicate via system bus
- Each component has its own inputs/outputs

Fig. 3.15

  - System bus must support them all

Fig. 3.16

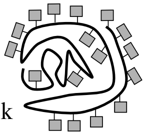
System Bus

- 50-100 lines (wires)
  - address
  - data
  - control
  - other: power, ground, clock
- Performance
  - bandwidth, how many bits per sec?
  - propagation delay?

(väyläkapasiteetti)

(päästä päähän viive)

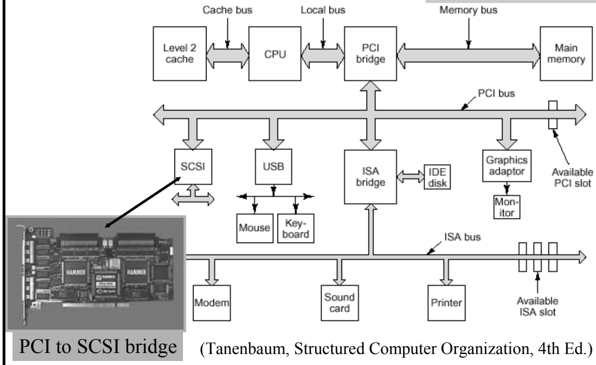
Bus Configurations



- One bus alone
  - might be very long
    - large end-to-end signal time
  - serious von Neumann bottleneck
  - all devices use similar speeds
  - slowest device determines speed used
- Hierarchy of buses
  - can maximize speed for limited access
    - closer to CPU
  - lower speed general access I/O
    - further away from CPU

Hierarchy of Buses

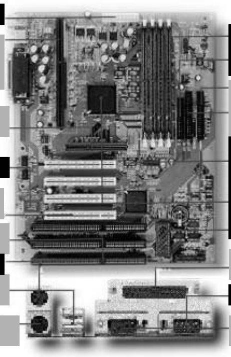
Typical Pentium II system



PCI to SCSI bridge

(Tanenbaum, Structured Computer Organization, 4th Ed.)

[LX6] - Pentium®II Processor Based Motherboard



[redacted]

CPU

AGP

[redacted]

PCI

ISA

[redacted]

USB

keyboard/mouse

[redacted]

IDE

[redacted]

parallel

[redacted]

serial

http://www.abit-usa.com/english/product/index.htm

Bus Design Features (3)

- Bus type
  - dedicated
    - address wires and data wires
  - multiplexed (aikavuorottelu)
    - address & data on same wires
- Arbitration method (vuoronvalinta)
  centralised, distributed (keskitetty, hajautettu)
  bus controller, arbiter (vuoronantaja)
- Timing
  synchronous (asynkrooninen,
  asynchronous epäsynkrooninen)

Synchronous timing

- All same speed devices
- All synchronized with a clock signal
- Slowest device determines speed
- Can make assumptions on when some other device will do something, or how fast it will do it *”Do this in next cycle!”*
  - 1 or 2 clock cycles to do it?
  - data will be ready to read in 1 cycle Fig. 3.19
  - written data stored in 1 cycle (Fig. 3.19 (a) [Stal99])

How to read
timing diagrams: Fig. 3.27 (Fig. 3.26 [Stal99])

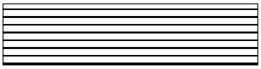
Asynchronous timing

- No need to have same speed devices
- Timing determined with change of signal levels
  - signaling may happen only at predetermined times
- Synchronize with signals (wires) *”Do this in when you can, please. Let me know, when you are done”*
  - ”read”, ”write”, ”ack”, …
- Speed determined by devices in action
  - not by all devices that are or could be connected!
- Can ***not*** make assumptions on when some other device will do something

Fig. 3.20 (Fig. 3.19 (b) [Stal99])

Bus Design Features (cont)

- Bus width
  - address, data



- Data transfer types
  - read, write

    Fig. 3.21 (Fig. 3.20 [Stal99])
    - multiplexed & non-multiplexed operations
  - read-modify-write
    - E.g., for indivisible increments (concurrency control method for multiprocessor environments)
  - read-after-write
    - E.g., for check that write succeeds (multiproc. env.)
  - block
    - long delay for interrupt handling?

Example Bus: Industry Standard Architecture (ISA, or PC-AT)

- Bus type: dedicated
- Arbitration method: single bus master
- Timing: asynchronous
  - own 8.33 MHz clock,
  - 15.9 MBps max data rate, 5.3 MBps in practice
- Bus width: address 32, data 16
- Data transfer type
  - read, write, read block, write block

Example: Peripheral Component Interconnect (PCI) Bus

- Bus type: multiplexed
- Arbitration method: centralised arbiter
- Timing: synchronous, own 33 MHz clock
  - 2.122 Gbps (265 MBps) max data rate
- Bus width: address/data 32 (64), signal 17
- Data transfer type
  - read, write, read block, write block
- Max 16 slots (devices)

PCI Configurations

- Hierarchy

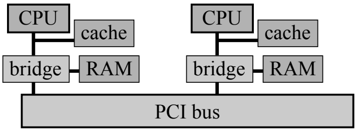
  Fig. 3.22 (Fig. 3.21 [Stal99])
- Bridge to internal/system bus allows them to be faster (with different bus protocol)
- Bridge to expansion buses allows them to slower (with different bus protocol)

PCI card: 49 Mandatory Signals (6)

- 32 pins for address/data, time multiplexed
  - 1 parity pin
- 4 pins for command type/byte enable
  - E.g., 0110/1111 = memory read/all 4 bytes
- 2 system pins: clock, reset
- 6 transaction timing & coordination pins
- 2 arbitration pins (not shared with other devices!) to PCI bus arbiter: REQ, GNT
- 2 error pins: parity, system

PCI Bus 51 Optional Signals (4)

- Request interrupt pins (4 pins for each dev)
- Cache support pins (2) for snoopy cache protocols



- 32 pins for additional multiplexed address/data
  - plus 7 control/parity pins
- 5 test pins
- 1 pin for locking the bus for multiple transactions
- (may have extra pin to select 66/33 MHz clock)

PCI Bus Transaction (4)

- Bus activity is in separate *transactions*
- Each transaction preceded by *arbitration*

Fig. 3.24 (Fig. 3.23 [Stal99])

  - central arbiter (e.g., First-In-First-Out)
  - determines initiator/master for transaction
- Transaction is executed
- Bus is marked “ready” for next transaction

PCI Transaction Types (5)

- Interrupt Acknowledge
  - READ interrupt parameter (e.g., subtype) for interrupt handler
- Special Cycle
  - broadcast message to many targets
- Configuration Read/Write
  - Read/Update (Write) device configuration data
- Dual Address Cycle
  - use 64 bit addresses in this transaction
- I/O or memory read/write
  - one or multiple words, cache line

PCI Read Transaction (no anim)

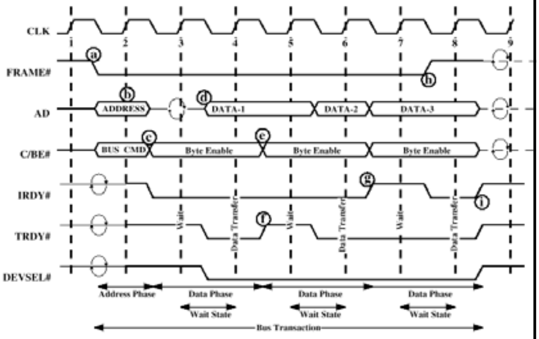


Fig. 3.23 (Fig. 3.22 [Stal99])

a) start trans frame, set addr, set trans. type

d) ack address, set data, indicate valid data

set & indicate data

data ready, read

b) recognise address, find data

data ready, read

set & indicate data

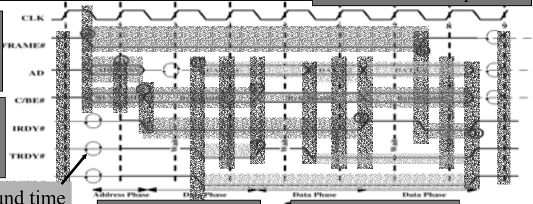
e) sel next bytes

g) not ready: hold

c) select bytes, indicate ready to receiv

f) need more time, indicate not valid data

h) ready for last block: end frame and stop hold



Initiator CPU action

Target memory action

turnaround time

data ready, read

get ready for next

get ready for next

All ready for new transaction

All ready for new transaction

Arbitration: A and B want bus

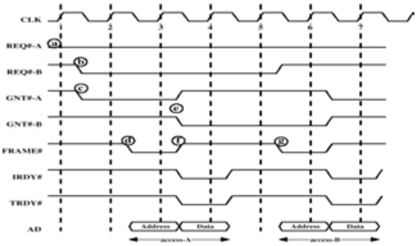


Fig. 3.25

Mostly just arbitration signals shown here (Fig. 3.24 [Stal99])

a) A wants bus

b) B wants bus

c) A granted bus

d) starts frame, requests also for next transaction

e) Grants bus to B for next trans.

g) starts frame, no more req.

knows that it has bus and bus is available

Sees that both still want it

f) marks last frame transfer, marks data ready

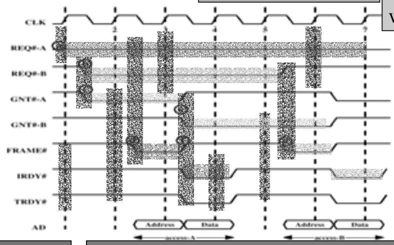
A’s target reads data

Sees that only **A** wants it

A action

B action

Arbitrator action



All ready for new trans

All ready for new trans, granted for B, B knows that it has bus

PCI Express New Bus to Replace PCI

- Code name ”Arapahoe” or 3GIO
- Prevent bus bottleneck between fast CPU and memory of the future
- Arapahoe Work Group
  - Compaq, Dell, IBM, Intel, Microsoft, ….
- Will replace PCI as industry standard
  - late 2003? low-end 2004? high-end 2005?
- PCI devices will work with PCI Express
- Speedup (E.g.) 64x as compared to std PCI
  - 16 GB/s with 32 *lanes* vs. 264 MB/s
- Scalable capacity per device (pin count, speed)

http://www.pcisig.com

-- End of Chapter 3: System Buses --

(PCI card - connectors also on other side, some pins not used by this card)