



HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Bus (*Väylä*)

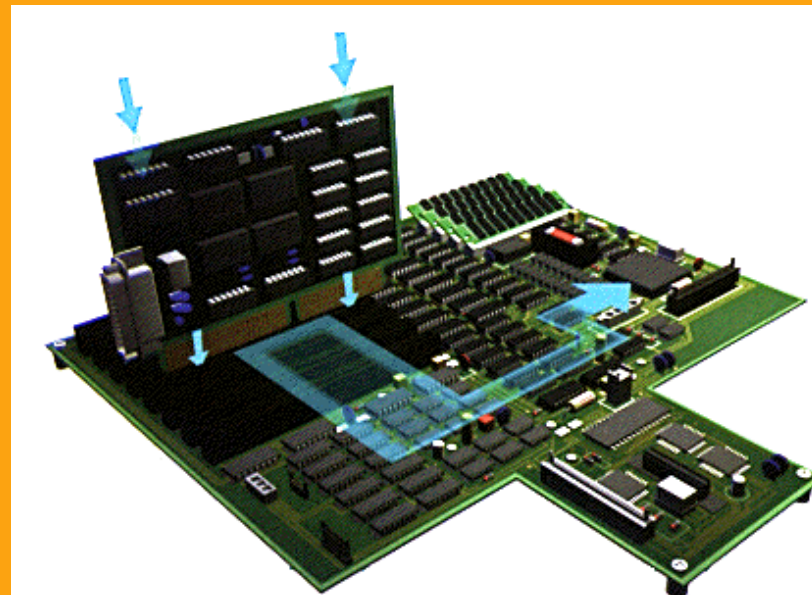
Stallings: Ch 3

What moves on Bus?

Bus characteristics

PCI -bus

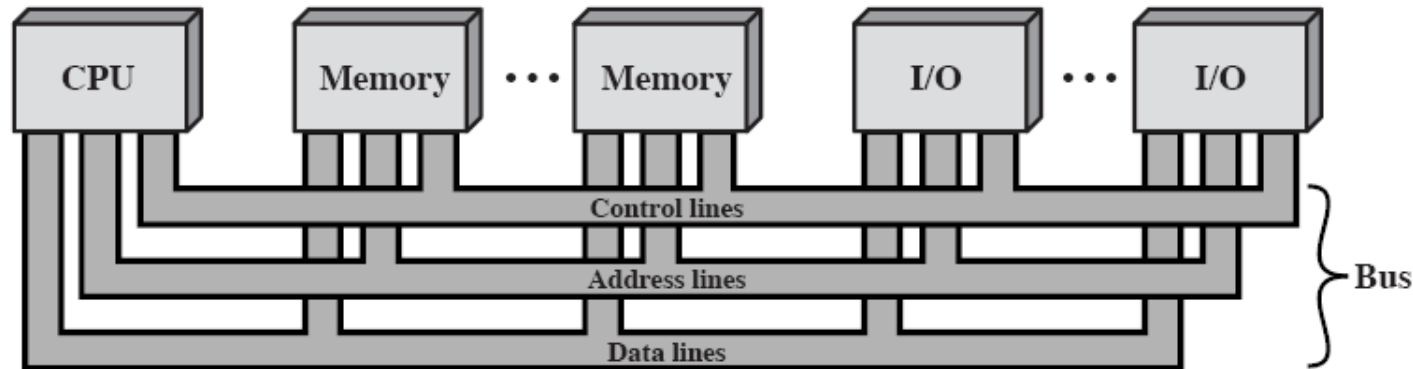
PCI Express





Bus

(Sta06 Fig 3.16)

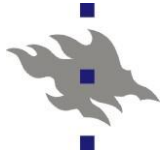


- For communication with and between devices
- Broadcast (*yleislähetys*): most common
 - Everybody hear everything
 - React to messages/signals to itself only
- Each device has its own control and status information
 - Device driver (OS) moves control data to device controller's registers
 - ~ memory address, device address, how much, direction
 - Device driver reads the status from the controller's status register
 - Ready? Operation successful? ...



Bus structure

- Control lines (*Ohjausväylä (~ johtimet)*)
 - Control and timing information
 - Operations: like memory read, memory write, I/O read
 - Interrupt request
 - Clock
- Address lines (*Osoiteväylä*)
 - Source and destination ids
 - Memory address, device address (module, port)
 - For transfer source and destination
 - Width (number of parallel lines) determines the memory address space (*osoiteavaruuden koko*)
 - For example: 32 b \Rightarrow 4 GB

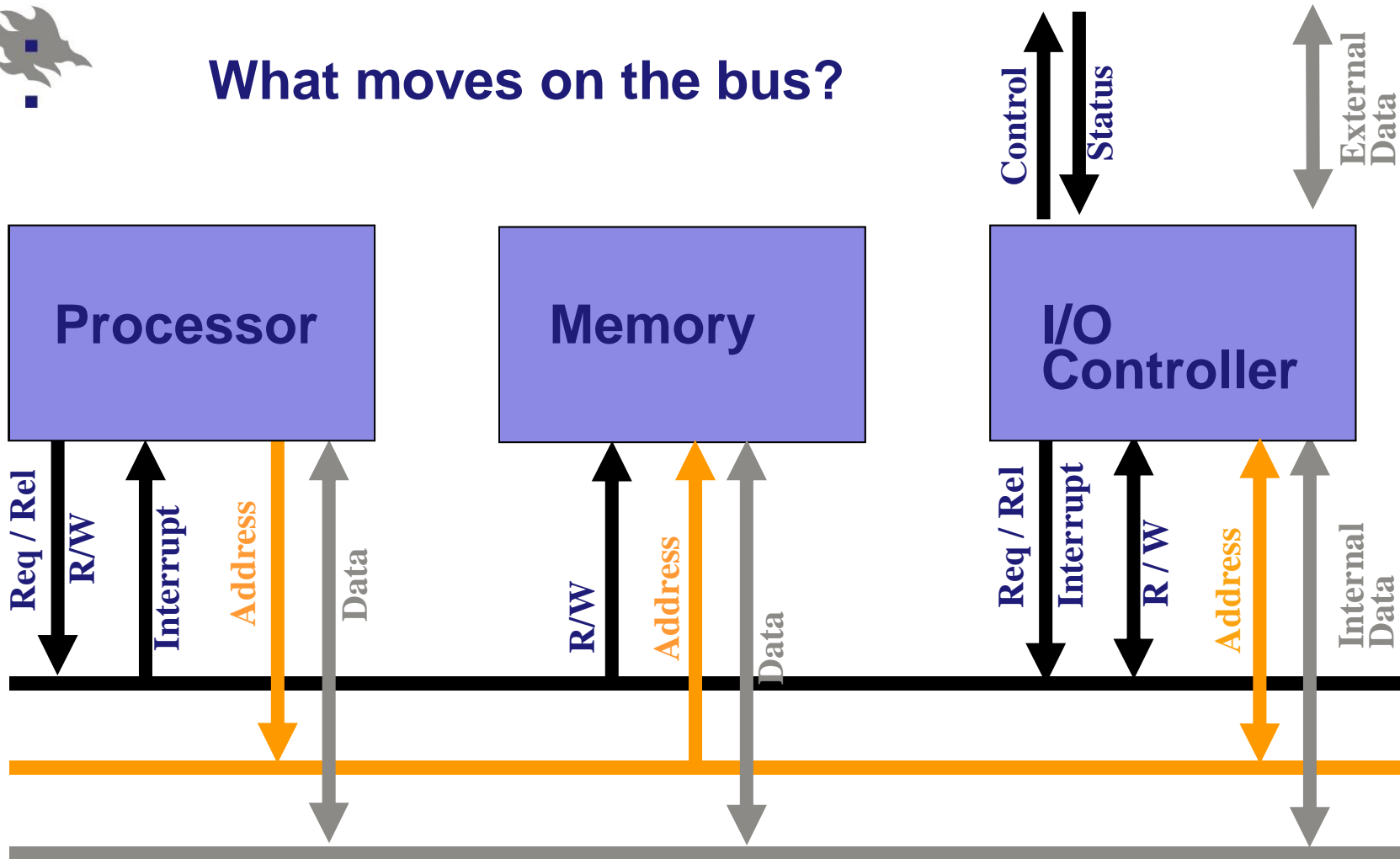


Bus structure

- Data lines (*Dataväylä*)
 - All processing information:
 - Instructions
 - Data
 - DMA –transfer contents
 - Width determines the maximum number of bits that can be transferred at the same time
 - For example 38b wide line allows 32 bits data plus 6 Hamming-coded parity bits (*tarkistusbitti*)



What moves on the bus?

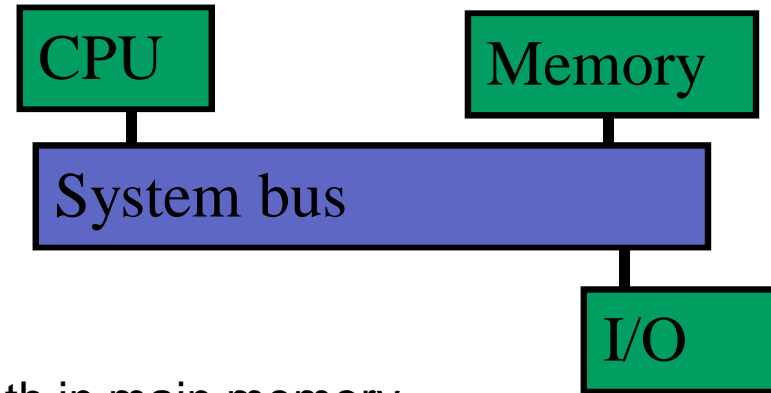


- **Timing**

- **Memory-mapped I/O**
- **DMA**



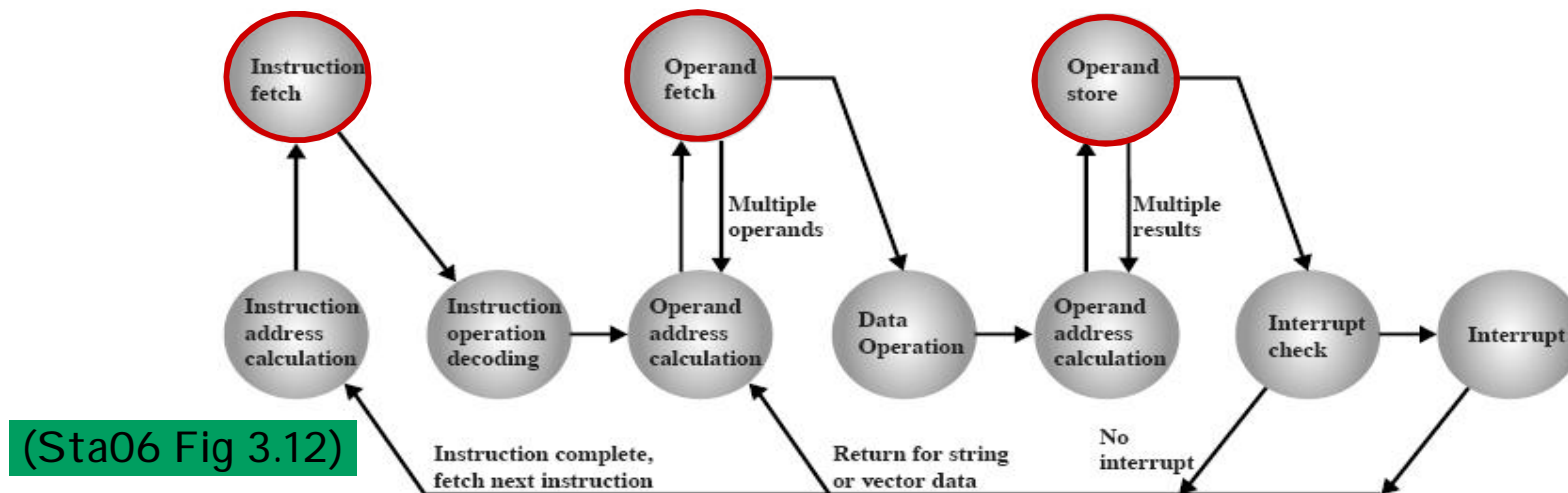
Bus = Bottleneck?

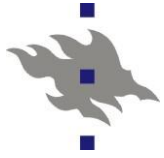


■ von Neumann architecture

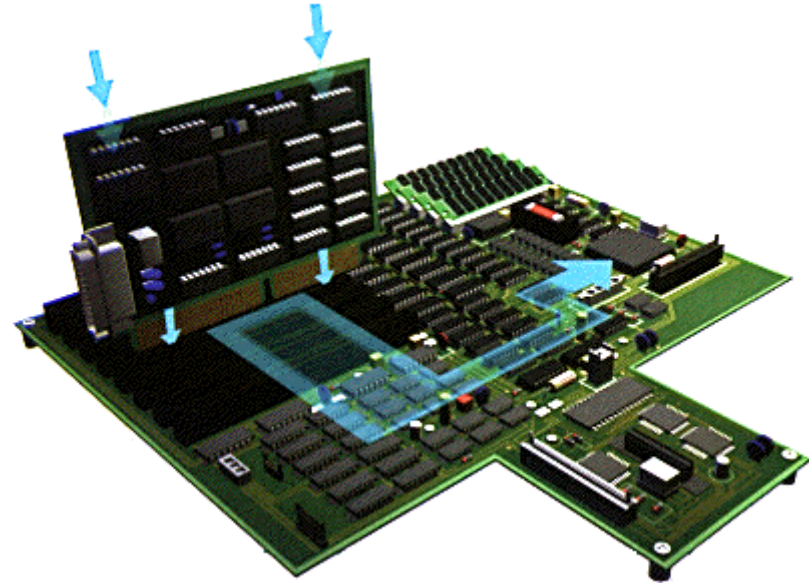
- Instructions and the data both in main memory
- All memory content referred using address
- Sequentially ordered instructions executed sequentially unless order changed explicitly (jumps, branches)

■ Fetch-Execute Cycle





Computer Organization II

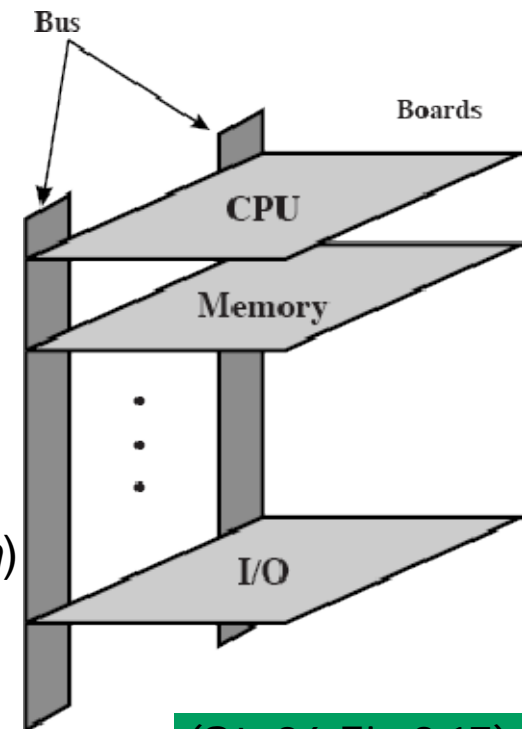


Bus characteristics



Bus characteristics

- Width
 - ~ 50 – 100 lines (*johdinta*) – mother board, cable, connectors
- Bus type
 - Dedicated, non-multiplexed (*dedikoitu*)
 - Address and data – separate lines
 - Time multiplexed (*aikavuoroteltu*)
 - Address and data share lines
 - Address valid / data valid -line
- Arbitration (*käyttövuoron varaus*)
 - Centralized
 - One bus controller, arbiter (*väyläohjain*)
 - Distributed
 - Controllers have necessary logic

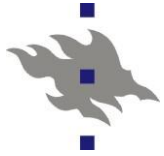


(Sta06 Fig 3.17)



Bus characteristics

- Timing (*ajoitus*, tahdistus)
 - Synchronous (*tahdistettu*)
 - Regular clock cycle (*kellopulssi*) – sequence of 0s and 1s
 - Asynchronous
 - Separate signals when needed
 - Shared traffic rules
everyone knows what is going to happen next
- Efficiency (*tehokkuus*)
 - Bandwidth (*kaistanleveys*)
 - How many bits per second



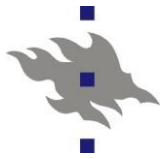
Synchronous timing

- Based on clock
 - Control line has clock pulse (cycle 1-0)
 - All devices "hear" the same pulse
- Event takes one cycle (commonly)
 - Start at the begin of the cycle (leading edge)
 - For example, reading data takes one cycle
- All devices in the bus work at the same pace
 - Slowest determines the speed of all
 - Each device knows the speed of the others
 - ⇒ knows, when it is ready for next event
- "Do this during the next cycle"
 - ⇒ Device can count on the other one to do it!



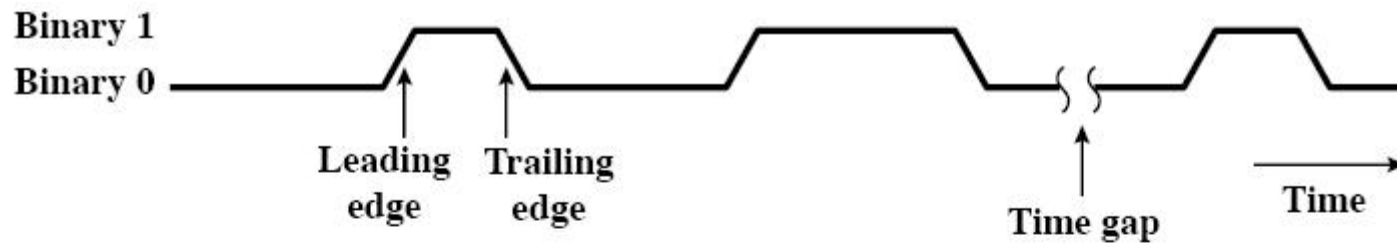
Asynchronous timing

- Devices can use arbitrary speeds (variation allowed)
 - Processing time depends on the device
 - Device can determine, when the other one is ready
 - How long is the event going to last to perform?
- Synchronization using a special signal
 - Send synchronization signal, when work done and ready
 - Address and data on bus \Rightarrow send signal "write"
(for example: change "write"-line to 1)
 - Data stored to memory \Rightarrow send signal "ack"
 - Time of the next event depends on the finish of the previous
- "Do this when you have time, inform me when ready"



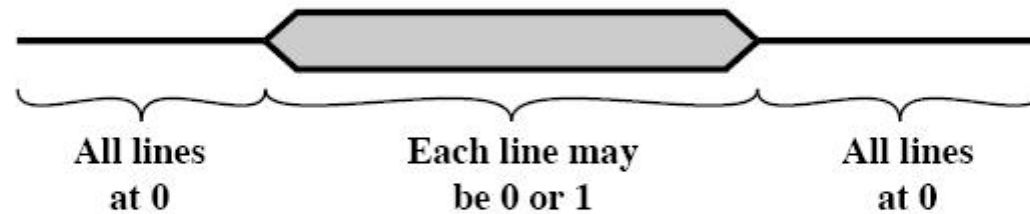
Timing diagrams (*ajoituskaavio*)

- See Appendix 3a [Sta06, Ch 3]

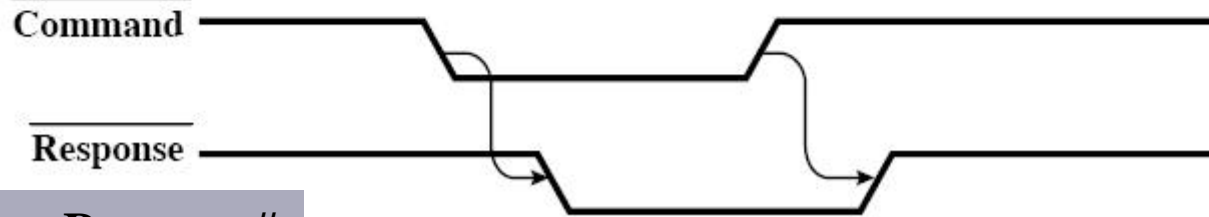


(a) Signal as a function of time

“assert” or
“active”
= 0-level



(b) Groups of lines

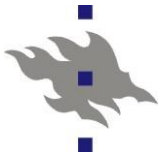


(c) Cause-and-effect dependencies

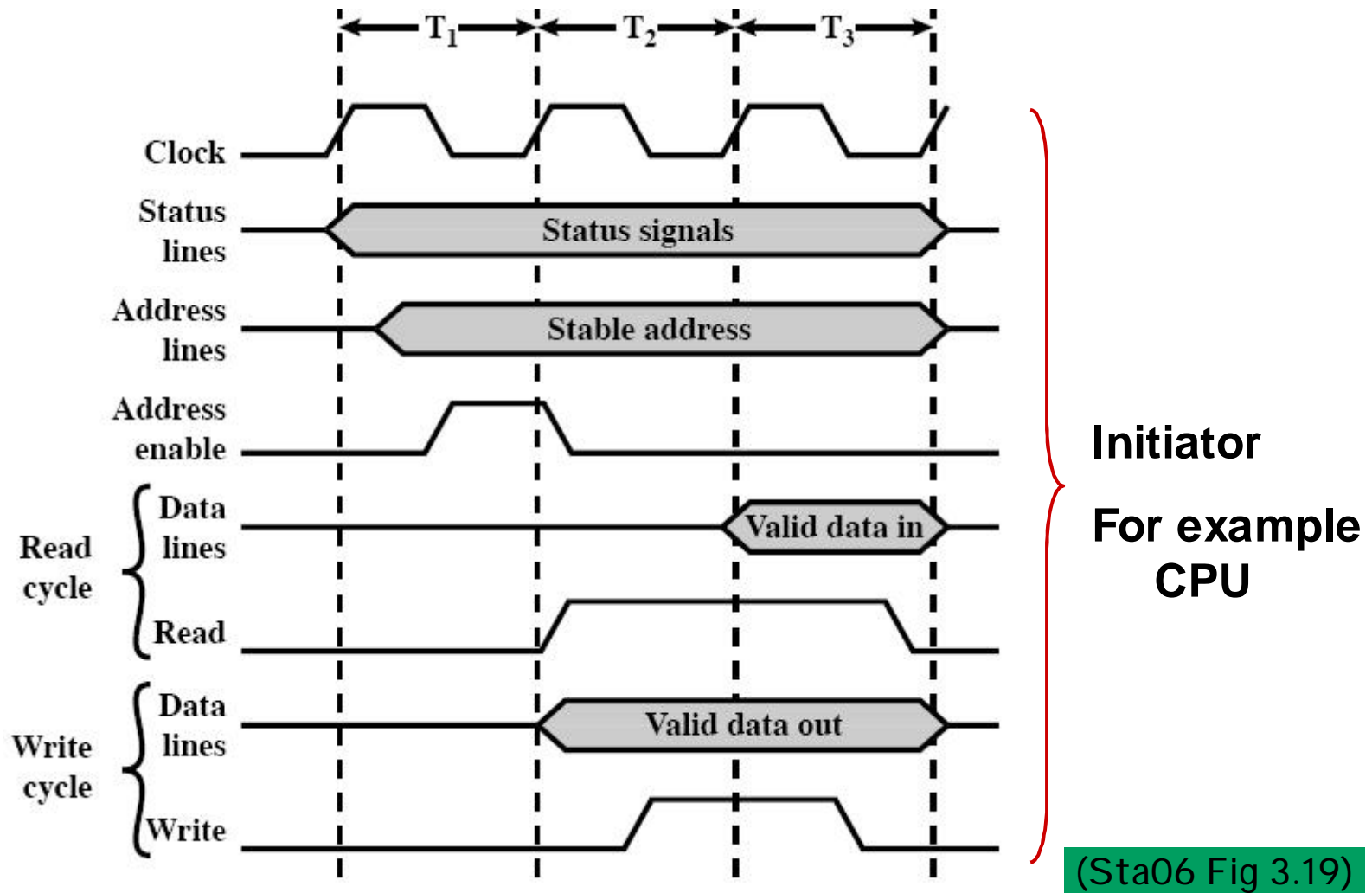
Response or Response#

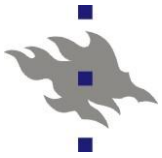
Asserted on 0; asserted on 1

(Sta06 Fig 3.27)

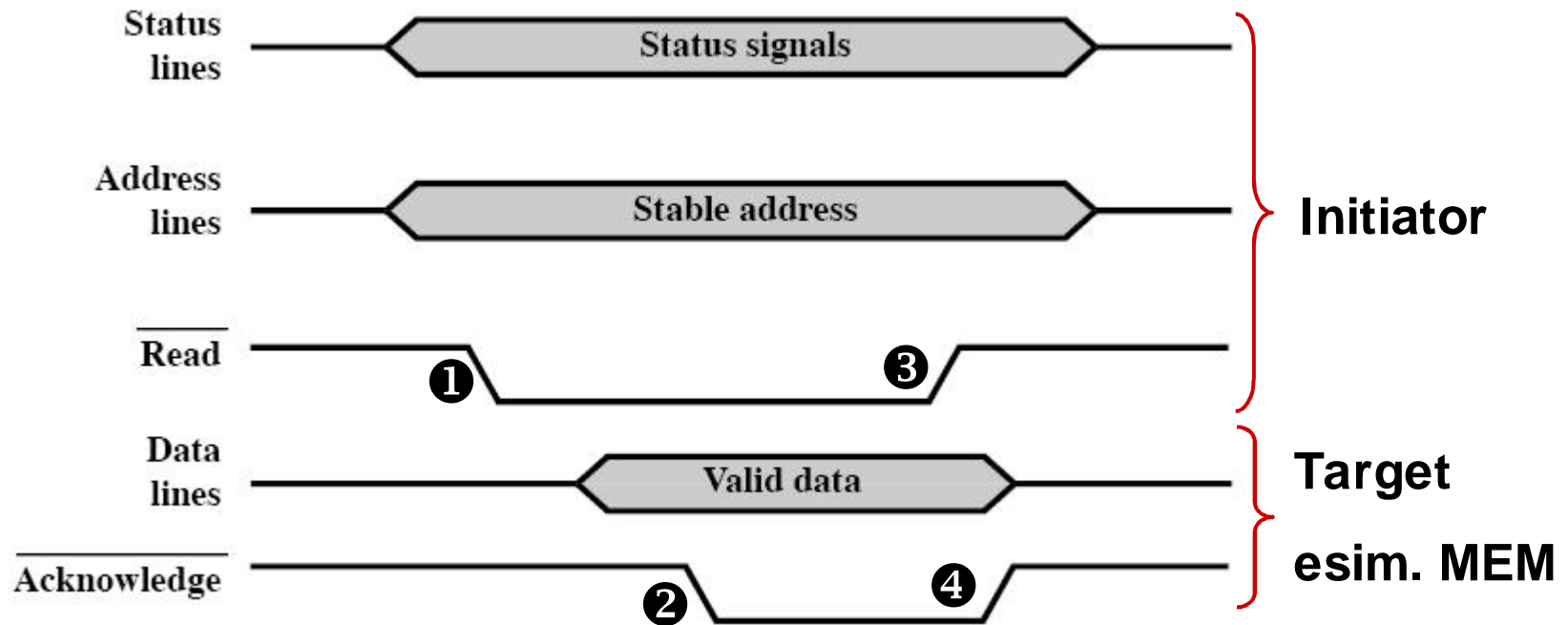


Synchronous timing

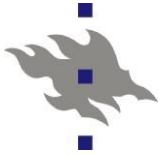




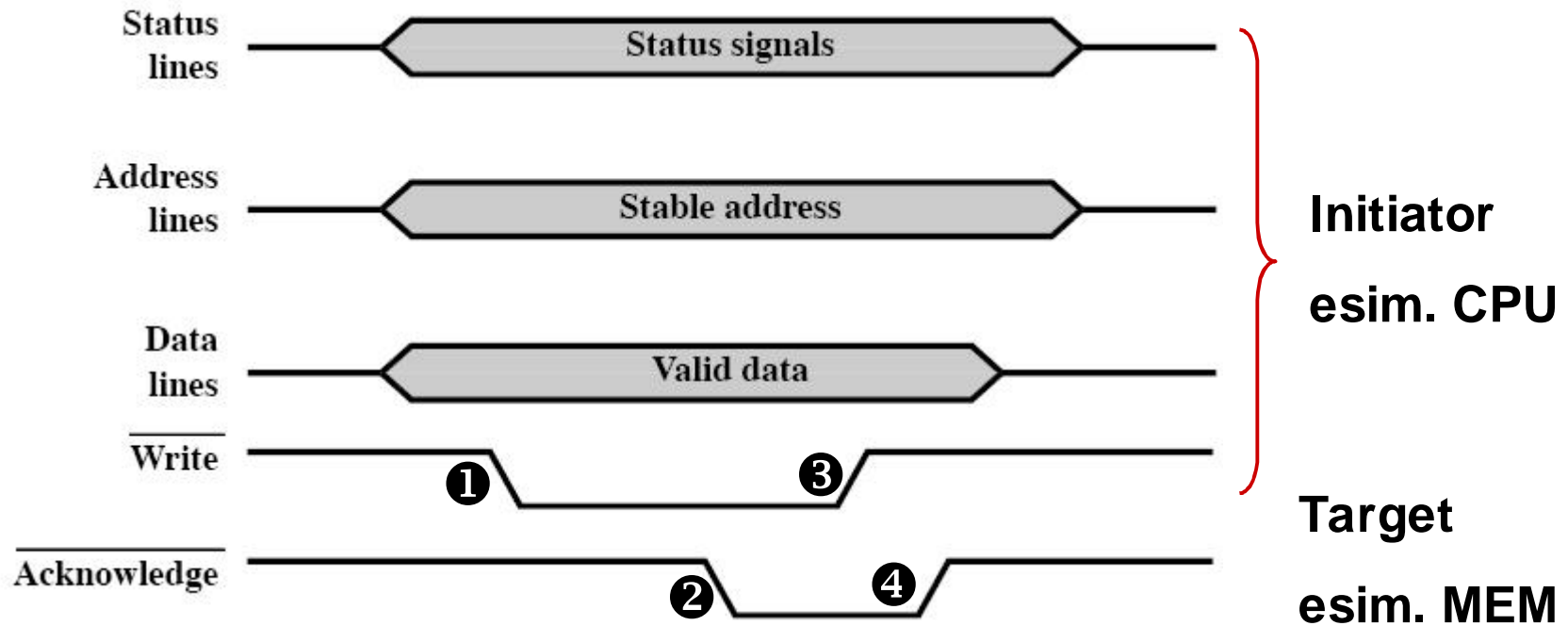
Asynchronous timing - Read



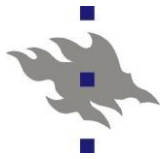
(Sta06 Fig 3.20a)



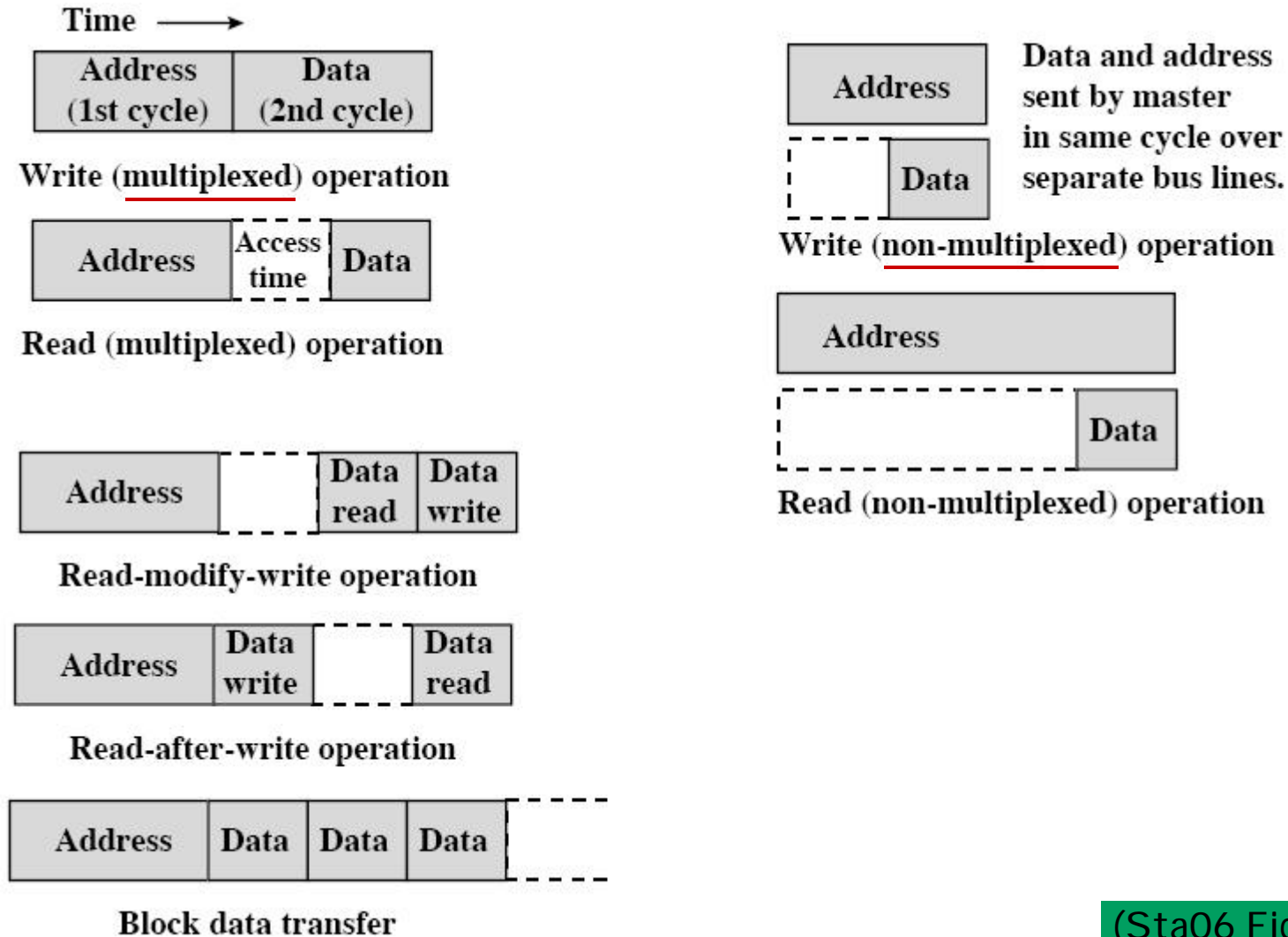
Asynchronous timing - Write



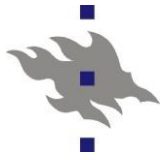
(Sta06 Fig 3.20b)



Events on bus (väylätapahtumia)

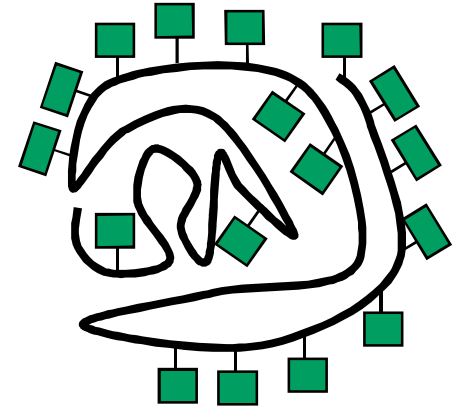


(Sta06 Fig 3.21)



Bus configuration

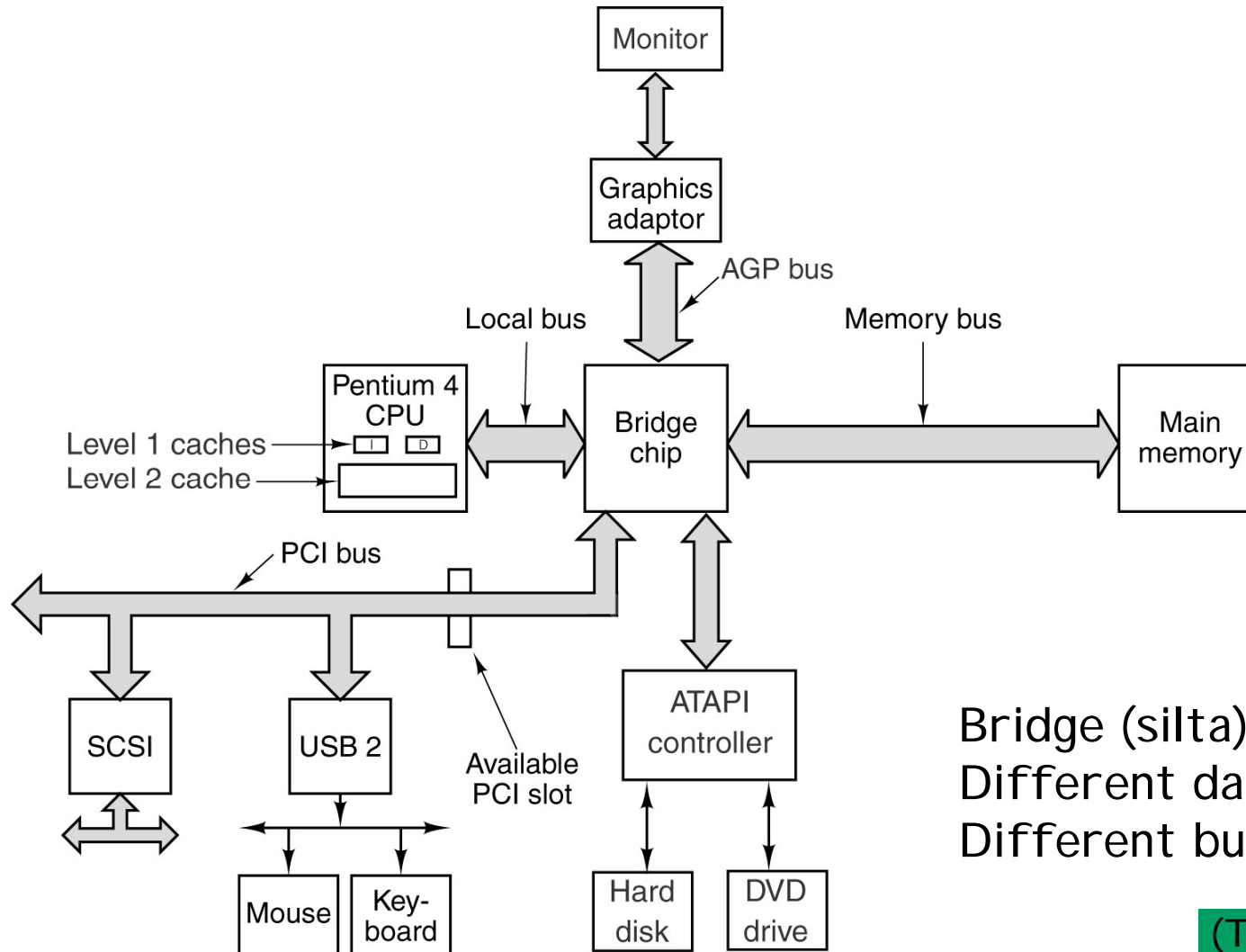
- All devices on one bus?
 - All must use the same technique
 - Long bus \Rightarrow propagation delay (*etenemisviive*)
 - Combined data rates of the devices may exceed the capacity of the bus
 - Collisions on the arbitration, extra wait
 - Synchronous? \Rightarrow slowest determines the speed of all
- Bus hierarchy
 - Isolate independent traffic from each other
 - Maximize the most important transfer pace, CPU \Leftrightarrow MEM
 - I/O can manage with lower speed



Bottleneck!

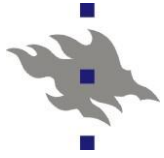


Bus hierarchy, typical Pentium 4



Bridge (silta)
Different data rates
Different bus protocols

(Tan06 Fig 3-53)



Computer Organization II

PCI-bus

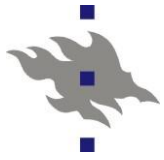
[Sta06, Ch 3.5]

<http://www.soe.ucsc.edu/classes/cmpe003/Spring02/motherboard.gif>



PCI: Peripheral Component Interconnect

- Time-based; 49 mandatory (+51 optional) signal lines
 - Address data: 32b mandatory (optional allows 64b)
 - Other signals: 17 mandatory (+ 19 optional)
- Centralized arbiter (*keskitetty väylän varaus*)
- Synchronous timing (*Synkroninen tahdistus*)
 - own 33 or 66 MHz clock (PCI-X: 133/156/533 Mhz)
 - Transfer rate 133, 266, 532 MB/s (PCI-X: 1 GB/s,4 GB/s)
- Events on the bus
 - read, write, read block, write block (multiplexed)
- Max 16 devices

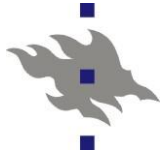


49 mandatory signal lines (*pakollista johdinta*)

Sta06 Table 3.3

- AD[32]: address or data, multiplexed (*aikavuorottelu*)
 - + 1 parity
- C/BE[4]: bus command tai byte enable, multiplexed
 - For example: 0110/1111 = memory read/all 4 Bytes
- CLK, RST#: clock, reset
- 6 for interface control
 - FRAME#, IRDY#, TRDY#, STOP#, IDSEL, DEVSEL#
- 2 for arbitration (*väylän varaus*)
 - REQ# requires, GNT# granted
 - Dedicated lines for devices
- 2 error reporting pins (lines)
 - PERR# parity, SERR# system





51 optional signal lines (valinnaista johdinta tai signaalia)

(ks. Sta06 Table 3.4)

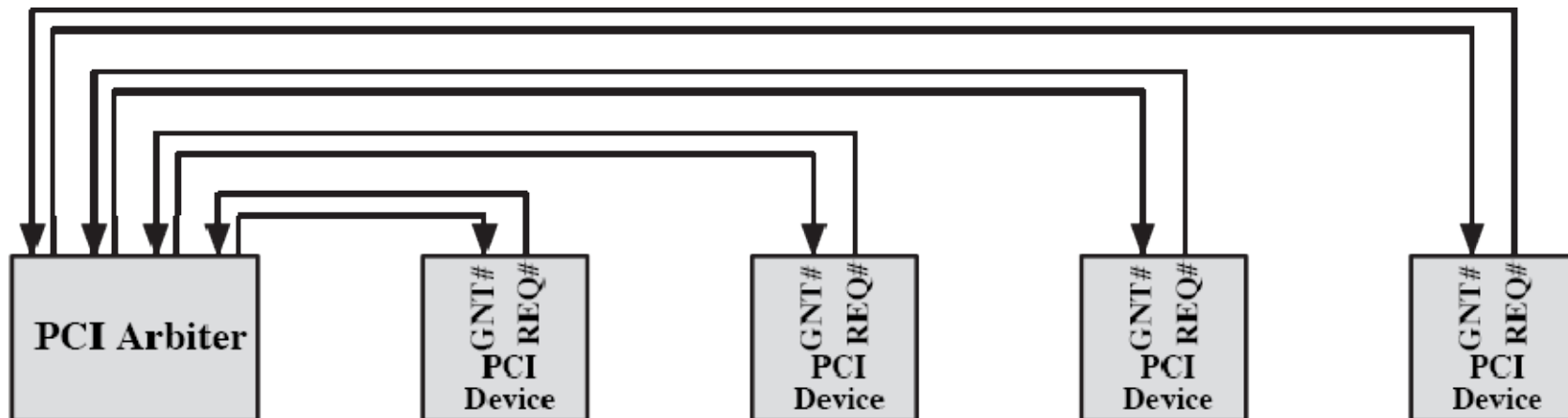
- 4 lines for interrupt requests (*keskeytyspyyntö*)
 - Each device has its own dedicated line(s)
- 2 lines for cache support (on CPU or other devices)
 - snoopy cache
- 32 A/D extra lines
 - 32 mandatory + 32 optional => 64 bit address/data lines
- 4 additional lines for C/BE bus command tai byte enable
- 2 lines to negotiate 64b transfer
- 1 extra parity line
- 5 lines for testing

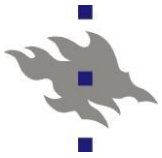


PCI: transactions

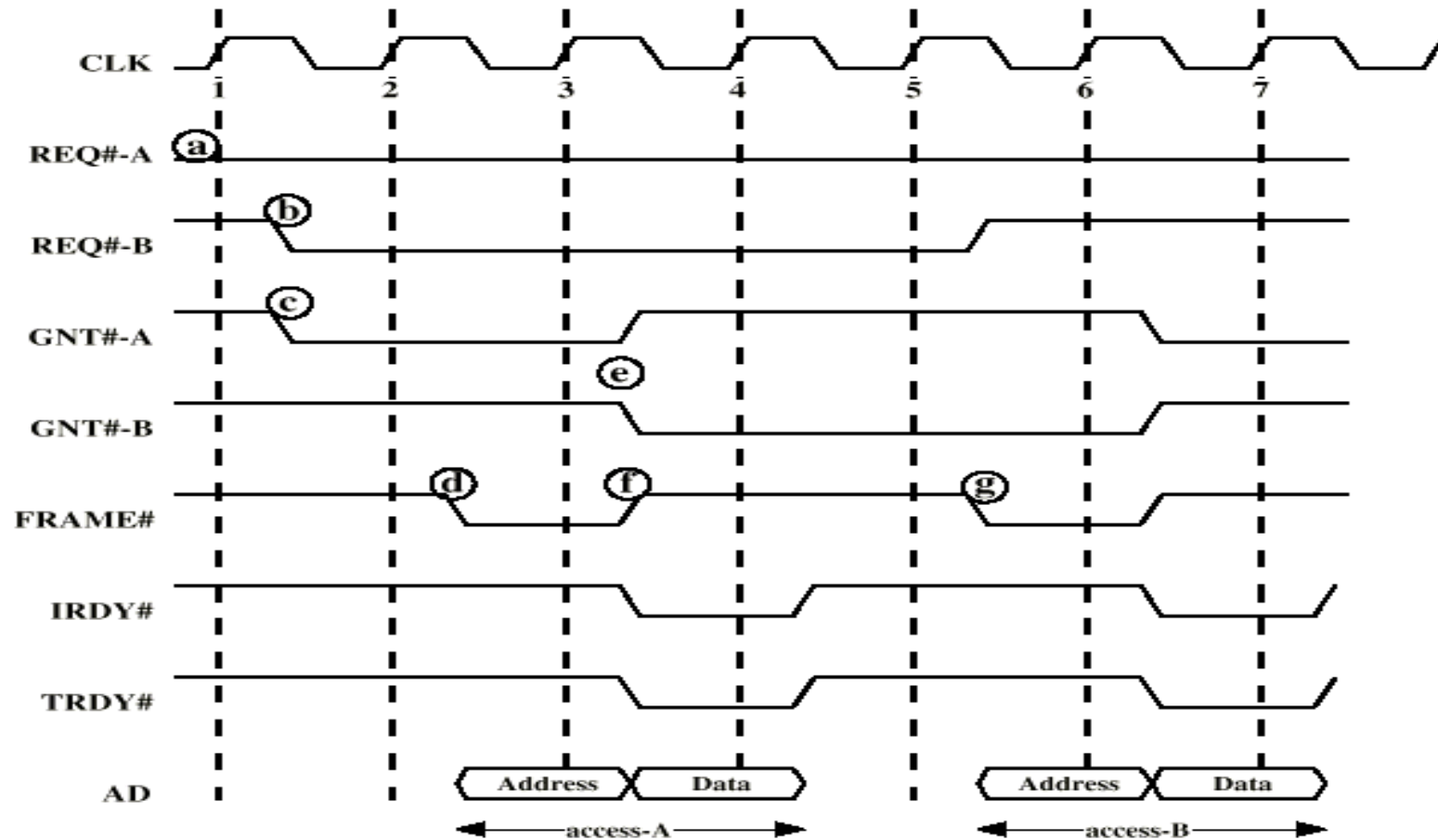
- Bus activity as transactions
 - New bus request for each new transaction
- First reservation
 - Central arbiter
 - send REQ, wait for GNT
- Then transaction
 - Initiator or master (device who reserved the bus)
 - Begin by asserting FRAME (reserve of bus)
 - Stop by releasing FRAME (indicate free bus)

(Sta06 Fig 3.24)





Bus arbitration : A and B want bus



(Sta06 Fig 3.25)

a) A wants bus

b) B wants bus

c) A granted bus

knows that it has bus and bus is available

d) starts frame, requests also for next transaction

Sees that both still want it

e) Grants bus to B for next trans.

f) marks last frame transfer, marks data ready

A's target reads data

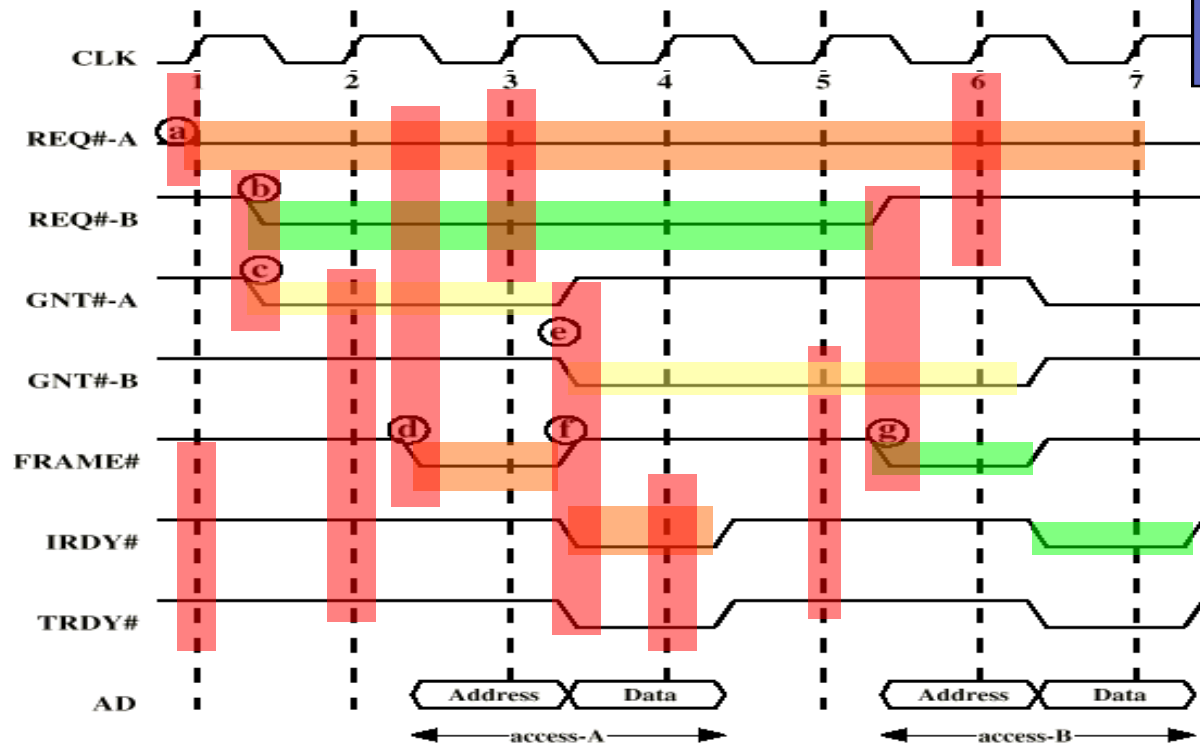
g) starts frame, no more req.

Sees that only A wants it

A action

B action

Arbiter action



All ready for new trans

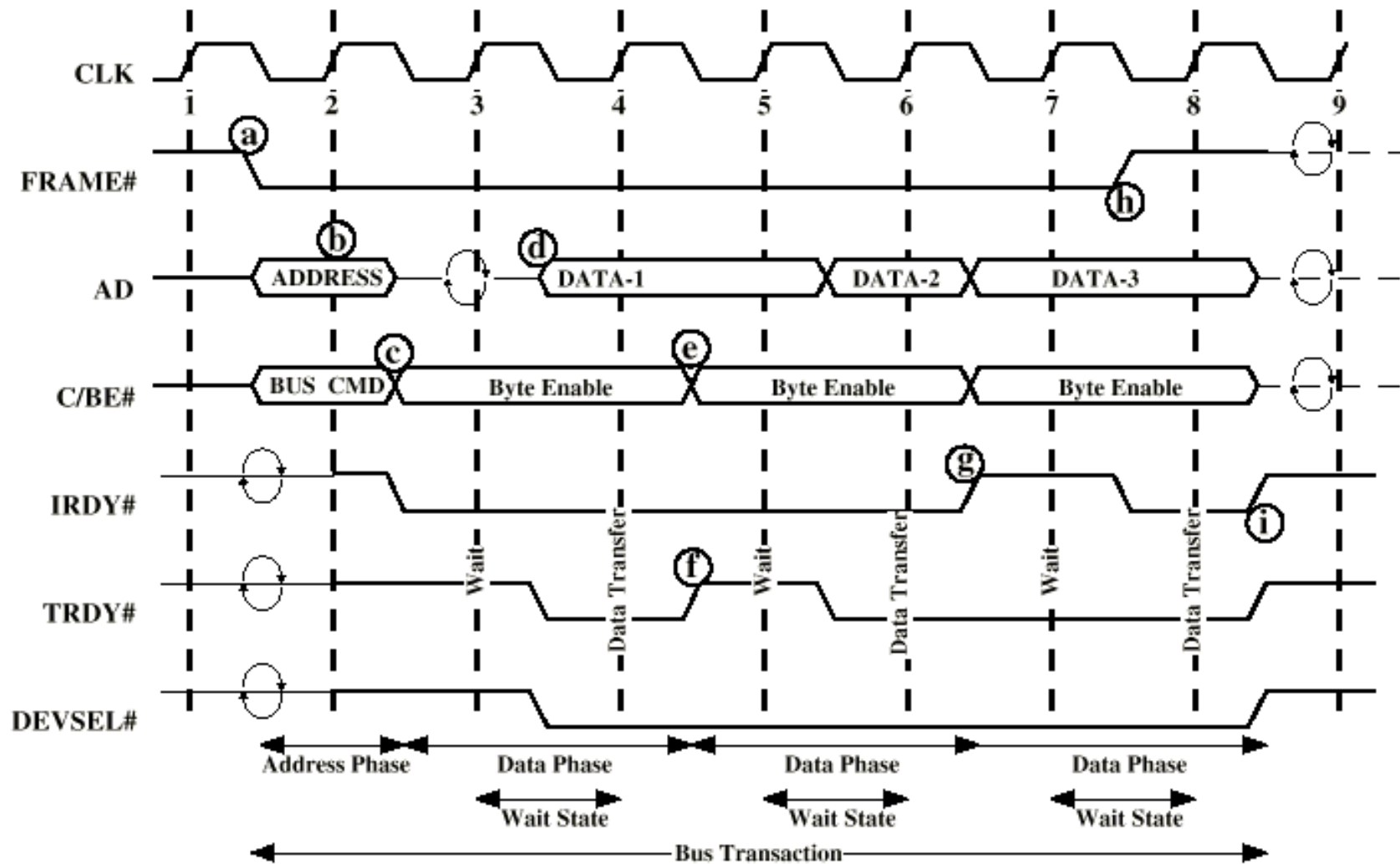
All ready for new trans, granted for B, B knows that it has bus



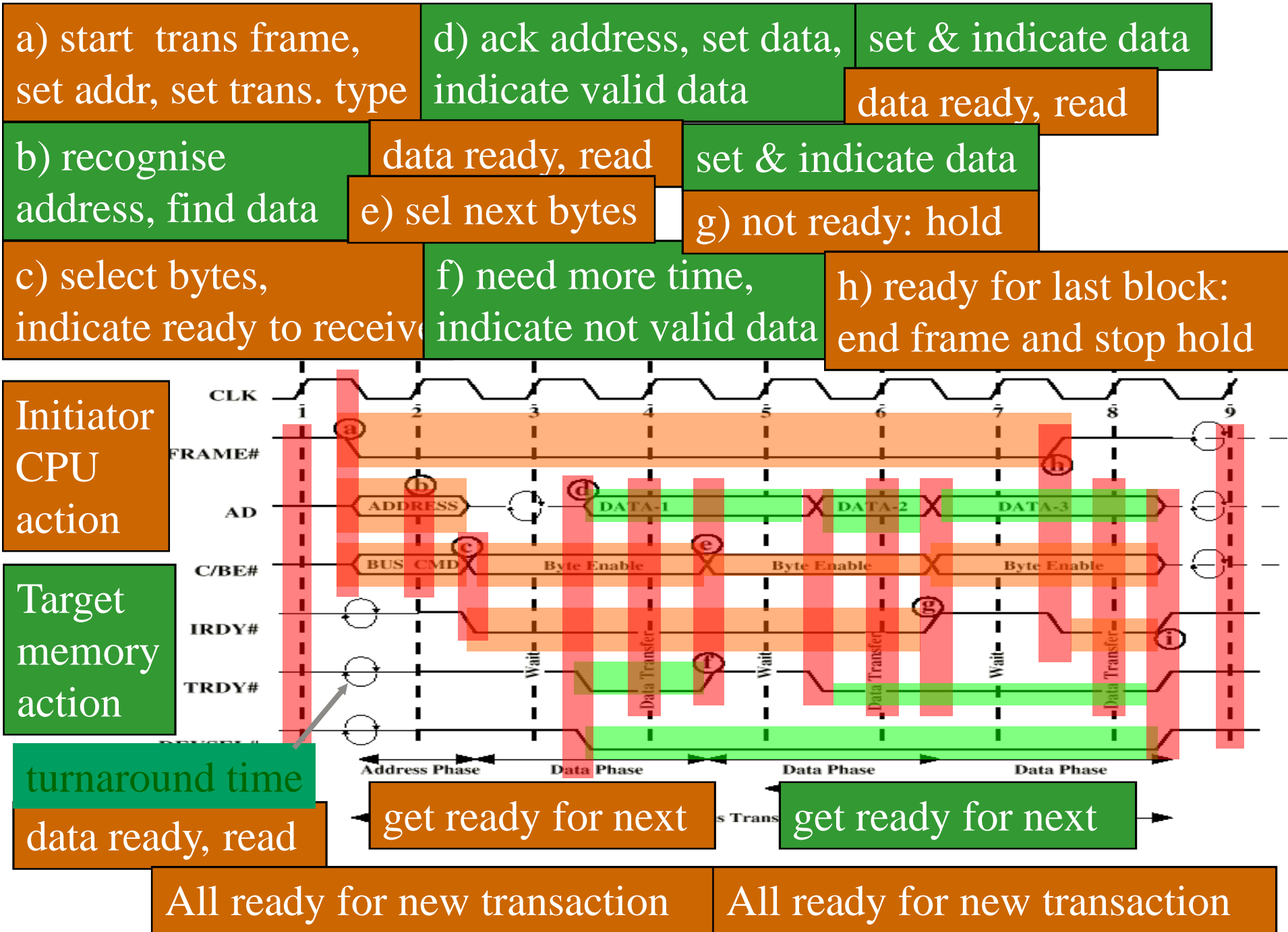
PCI: transactions

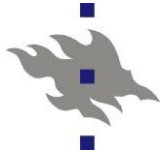
- Memory or I/O Read/Write [Line | Multiple]
 - Transfer one or more words (alternatively: cache line or block)
- Memory Write and Invalidate
 - Guarantees that at least one cache line written to memory (*Takaa, että tieto siirtyy välimuistista muistiin*)
- Configuration Read/Write
 - Access to configuration parameters on the device (256B)
 - Plug-and-Play, PnP
- Interrupt Acknowledge
 - Interrupt controller collect more interrupt information from the device (to create interrupt vector for interrupt handler)
- Special Cycle
 - Broadcast (*yleislähetys*) to one or more targets
- Dual Address Cycle
 - Indication of using 64 bit address

PCI Memory Read



(Sta06 Fig 3.23)





Tietokoneen rakenne

PCI Express

[Tan06, s.212]

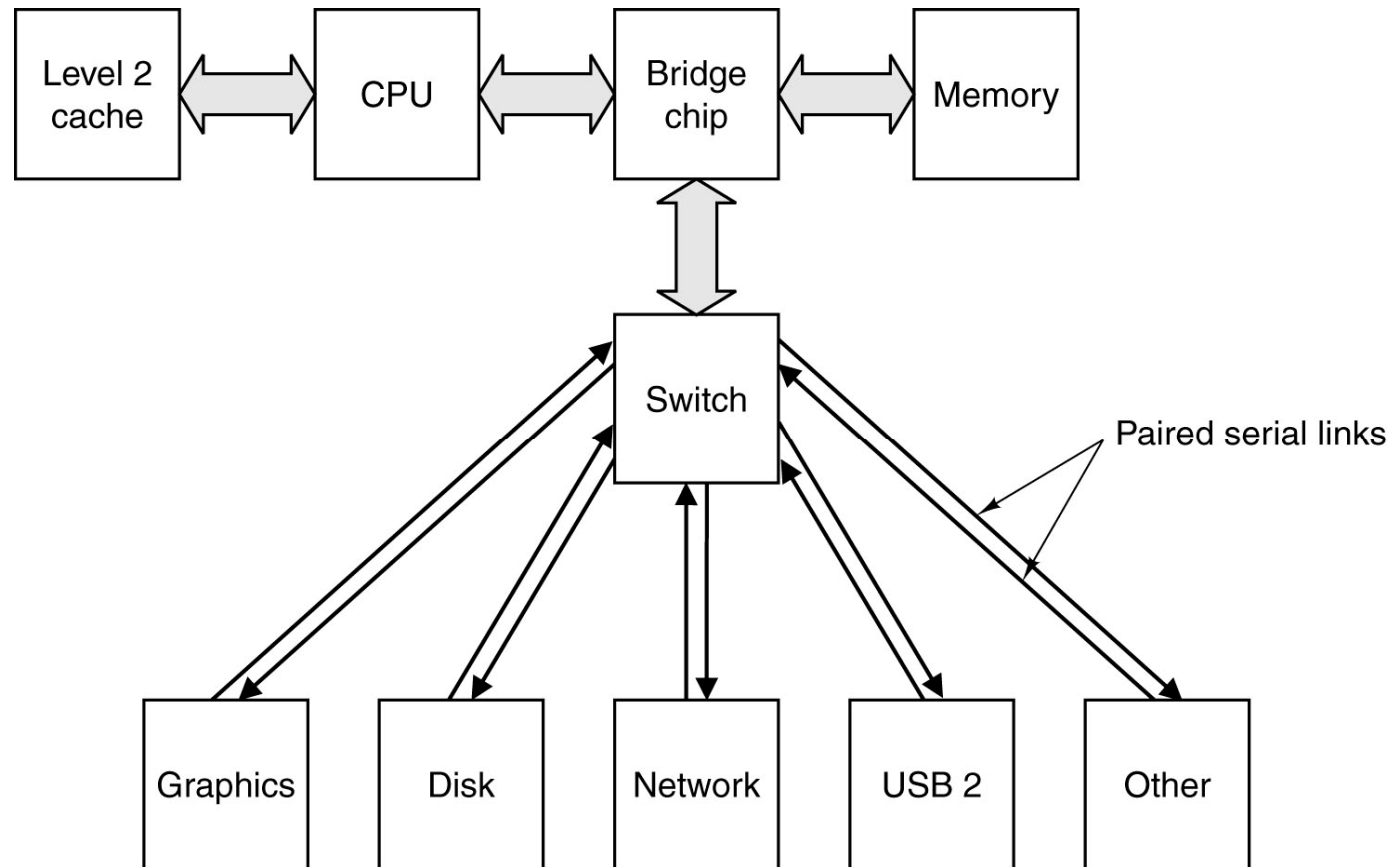


Packet-switched PCI Express (PCIe, PCI-E)

- PCI bus is too slow for some devices
- Replaces PCI bus (and possibly other I/O-bus)
 - Already available on new computers
- Hub on motherboard acting as a crossbar switch (*kytkin*)
- Based on point-to-point connections (*kaksipisteyhteys*)
 - Full-duplex, one lane has two lines (one send, one receive)
 - One device can use one or more (2,4,8,16,32) lanes
- Data stream (serial transfer)
 - Small packets (header + payload), bits in sequence
- No reservation, no control signals.
 - Each device may send at any time, when it wishes
 - Packet header contains the control information (like target)
- Data rate on one lane 250MB/s (future 3rd gen: 1GB/s)



Typical PCI Express system



(Tan06, Fig 3-57)



PCI Express advantages

- Each packet contains error-detection code
 - CRC – cyclic redundancy check
 - More reliable than parity bit on PCI bus
- Devices can be further from each other (partitioning)
 - For example, hard disk inside the monitor covers
 - PCI allowed max 50 cm
- Expandability PCI Express: max not determined
 - A device can be a switch
- Allows hot-swap Plug-and-Play
 - Device can be connected /disconnected while running, PnP
- Physically smaller connectors
 - Computers and devices can be smaller



Review Questions

- Main differences between synchronous and asynchronous timing?
- Benefits of bus hierarchy?
- Main differences of PCI Express and PCI ?

- See course book for more review questions



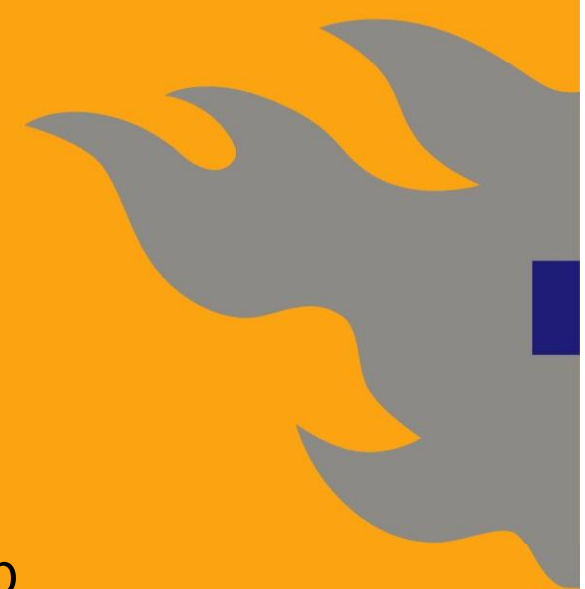
Stallings Online Chapter 20

Self-study!!

Combinatorial Circuits, Sequential Circuits

Building blocks for more complex circuits

- ◆ Multiplexer
- ◆ Encoders/decoder
- ◆ Read-Only-Memory
- ◆ Adder
- ◆ Flip-Flop
- ◆ S-R Latch
- ◆ Registers
- ◆ Counters





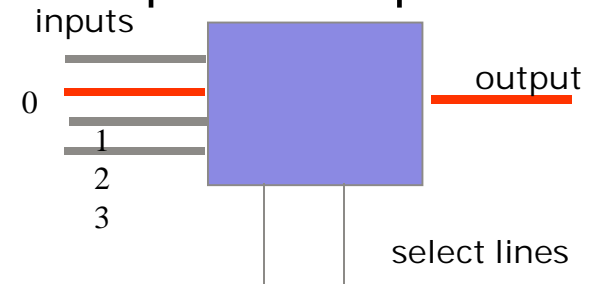
Multiplexers

limitin

Sta06 Fig B.12

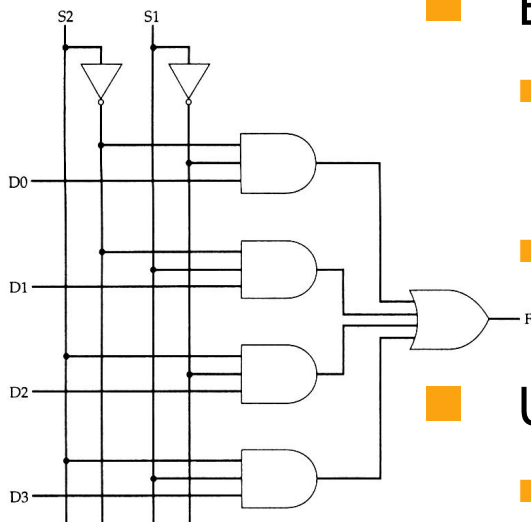
- Select one of many possible inputs to output

- black box
- truth table
- implementation



Sta06 Table B.7

Sta06 Fig B.13



- Each input/output “line” can be many parallel lines

- select one of three 16 bit values
 - $C_{0..15}$, $IR_{0..15}$, $ALU_{0..15}$
- simple extension to one line selection
 - lots of wires, plenty of gates ...

Sta06 Fig B.14

- Used to control signal and data routing
- Example: loading the value of PC



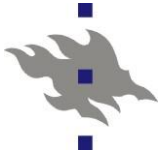
Encoders/Decoders

- Exactly one of many Encoder input or Decoder output lines is 1
- Encode that line number as output
 - hopefully less pins (wires) needed this way
 - optimise for space, not for time space-time tradeoff
 - Example:
 - encode 8 input wires with 3 output pins
 - route 3 wires around the board
 - decode 3 wires back to 8 wires at target

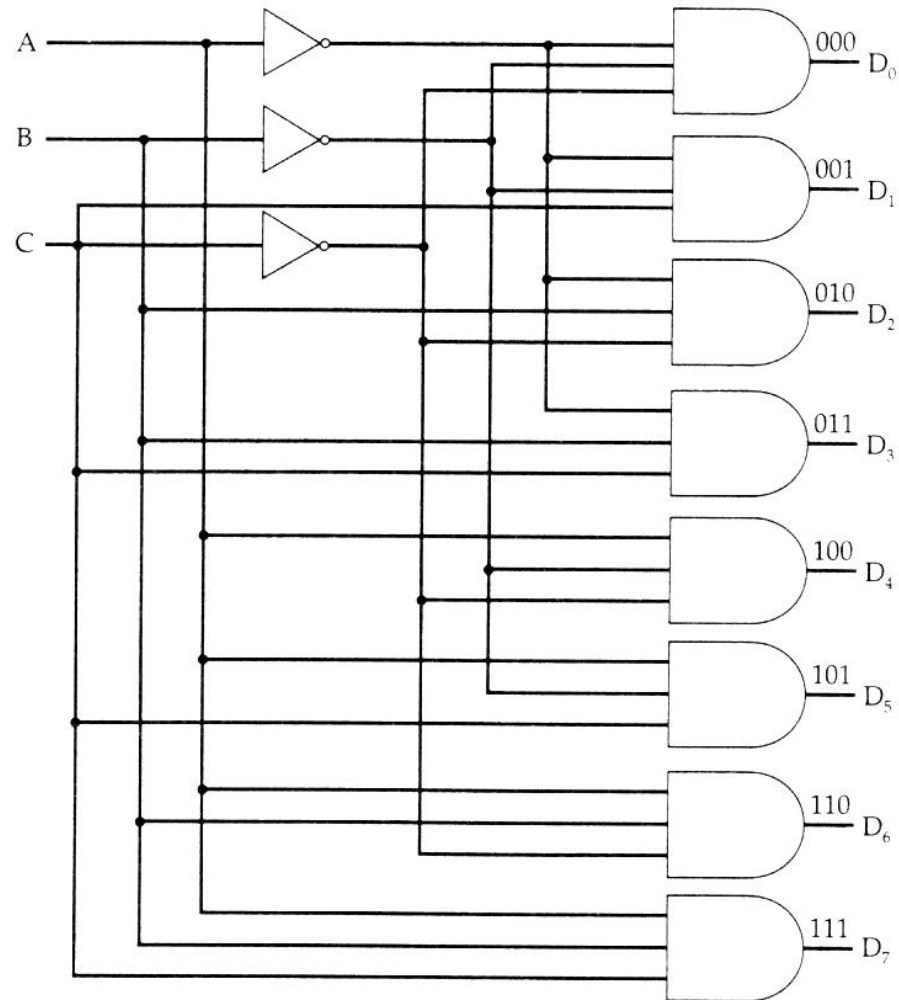
Sta06 Fig B.15

Ex. Choosing the right memory chip from the address bits.





Decoder



Sta06 Fig B.15



Read-Only-Memory (ROM) (5)

- Given input values, get output value
 - Like multiplexer, but with **fixed data**
- Consider input as address, output as contents of memory location

- Example Sta06 Table B.8
 - Truth tables for a ROM
 - 64 bit ROM
 - 16 words, each 4 bits wide

 - Implementation with decoder & or gates Sta06 Fig B.20



ROM - truth table

address

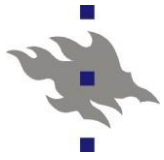
value

Sta06 Table B.8

Mem (7) = 4

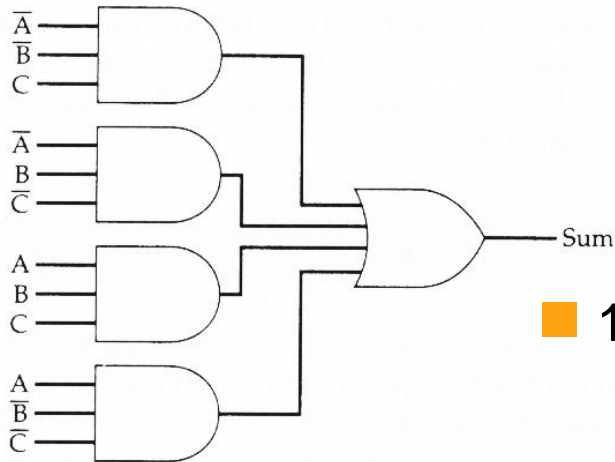
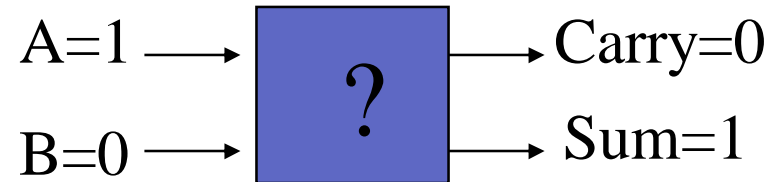
Mem (11) = 14

Input				Output			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

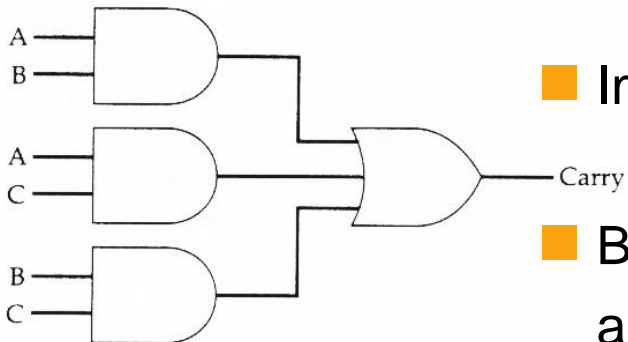
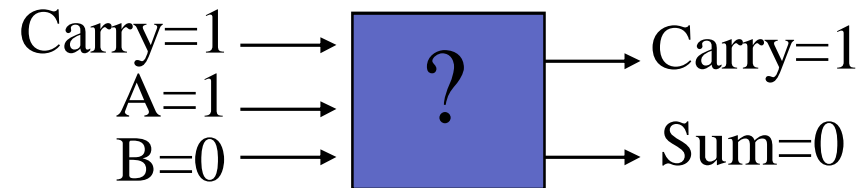


Adders

■ 1-bit adder



■ 1-bit adder with carry



■ Implementation

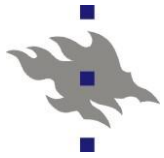
■ Build a 4-bit adder from four 1-bit adders

See Sta06 Table B.9, Fig B.22

Compare to ROM?

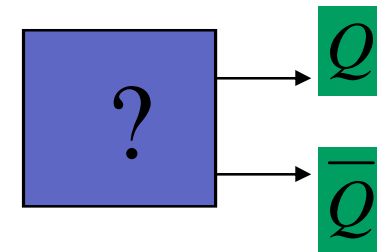
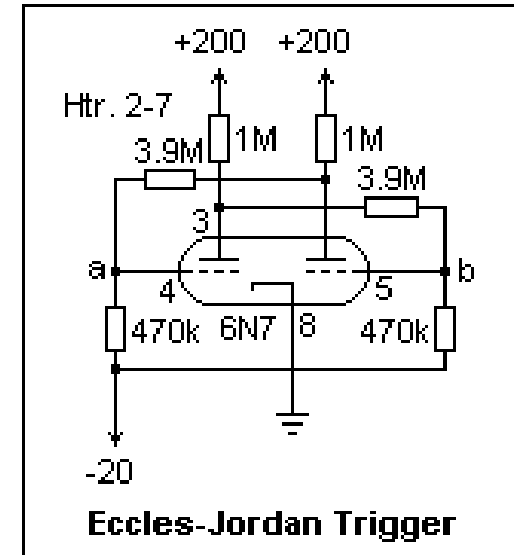
See Sta06 Fig B.21

Sta06 Fig B.22



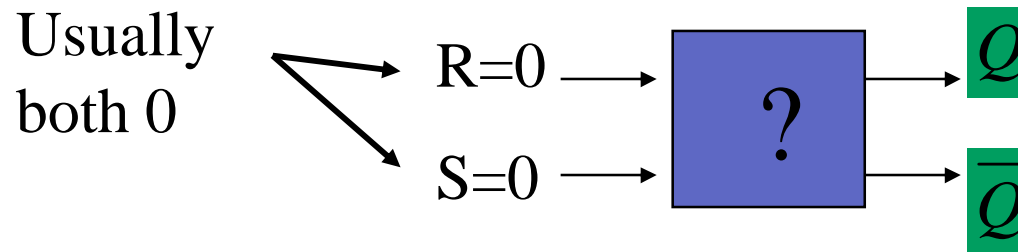
Flip-Flop (*kiikku*)

- William Eccles & F.W. Jordan
 - with vacuum tubes, 1919
- 2 states for Q (0 or 1, true or false)
 - 1-bit memory
 - **Maintains state when input absent**
- 2 outputs
 - complement values
 - both always available on different pins
- Need to be able to change the state (Q)





S-R Flip-Flop or S-R Latch (*salpa*)

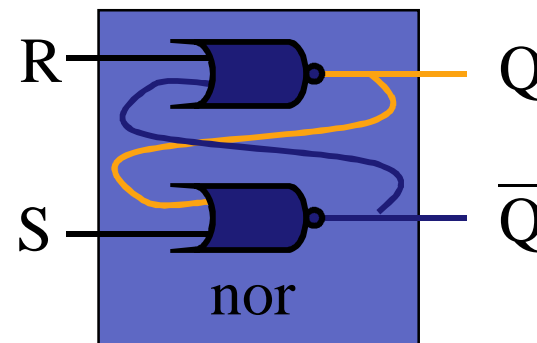


S = “SET” = “Write 1” = “set S=1 for a short time”

R = “RESET” = “Write 0” = “set R=1 for a short time”

Use NOR gates

nor (0, 0) = 1
nor (0, 1) = 0
nor (1, 0) = 0
nor (1, 1) = 0



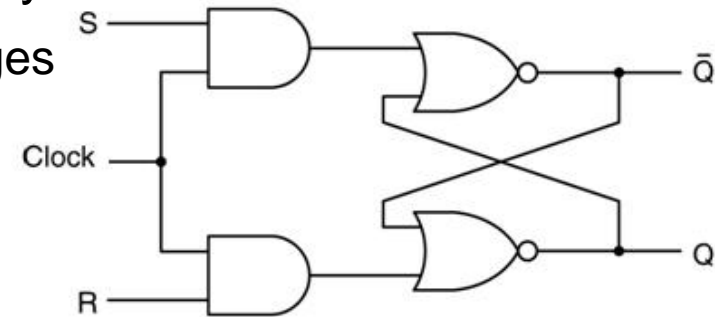


Clocked Flip-Flops

- State change can happen only when clock is 1

- more control on state changes

- Clocked S-R Flip-Flop



- D Flip-Flop Sta06 Fig B.27

(Sta06 Fig B.26)

- only one input D

- D = 1 and CLOCK → write 1

- D = 0 and CLOCK → write 0

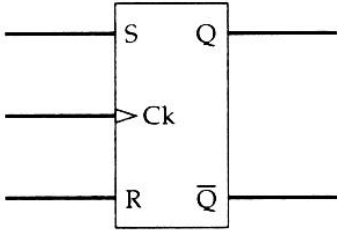
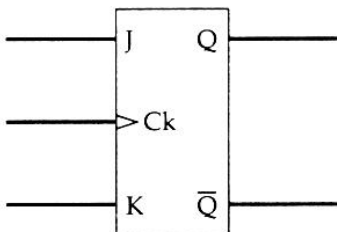
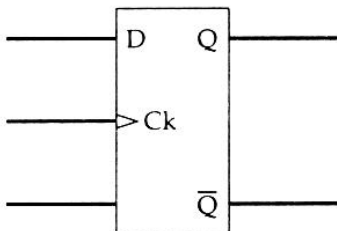
- J-K Flip-Flop Sta06 Fig B.28

- Toggle Q when J=K=1

Sta06 Fig B.29



Basic Clocked Flip-flops

Name	Graphic Symbol	Characteristic Table															
S-R		<table border="1"> <thead> <tr> <th>S</th> <th>R</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q_n</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>-</td> </tr> </tbody> </table>	S	R	Q_{n+1}	0	0	Q_n	0	1	0	1	0	1	1	1	-
S	R	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	1															
1	1	-															
J-K		<table border="1"> <thead> <tr> <th>J</th> <th>K</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Q_n</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>$\overline{Q_n}$</td> </tr> </tbody> </table>	J	K	Q_{n+1}	0	0	Q_n	0	1	0	1	0	1	1	1	$\overline{Q_n}$
J	K	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	1															
1	1	$\overline{Q_n}$															
D		<table border="1"> <thead> <tr> <th>D</th> <th>Q_{n+1}</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </tbody> </table>	D	Q_{n+1}	0	0	1	1									
D	Q_{n+1}																
0	0																
1	1																

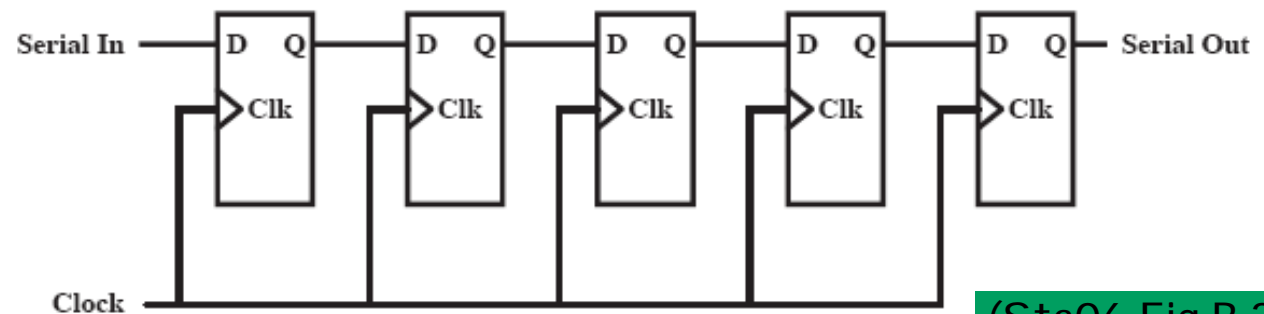


Registers

- Parallel registers
 - read/write
 - CPU user registers
 - additional internal registers

- Shift Registers
 - shifts data 1 bit to the right
 - serial to parallel?
 - ALU ops?
 - rotate?

Sta06 Fig B.30



(Sta06 Fig B.31)

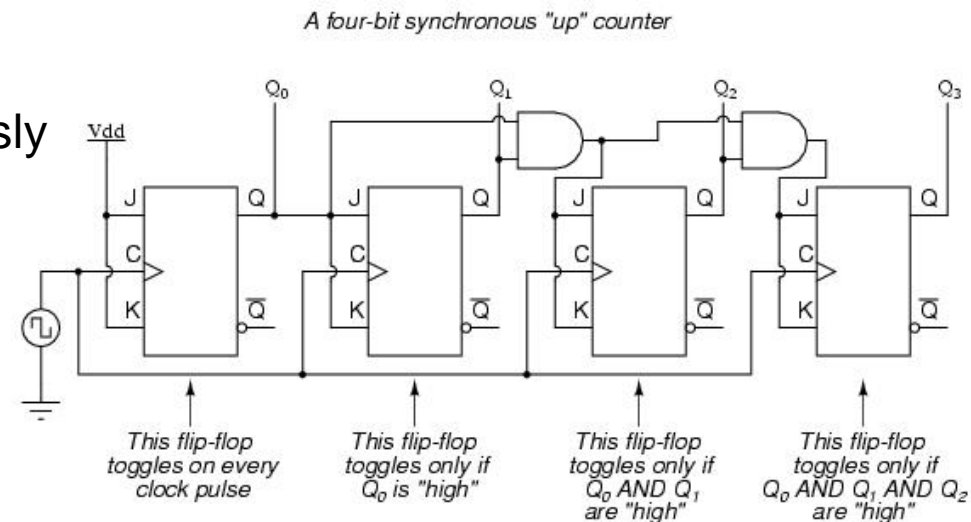


Counters

- Add 1 to stored counter value
- Counter
 - parallel register plus increment circuits
- Ripple counter (aalto, viive)
 - asynchronous
 - increment least significant bit,
 - and handle “carry” bit as far as needed
- Synchronous counter
 - modify all counter flip-flops simultaneously
 - faster, more complex, more expensive

space-time tradeoff

Sta06 Fig B.32



(<http://www.allaboutcircuits.com>)



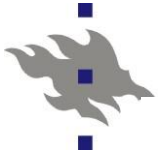
Summary

- Boolean Algebra → Gates → Circuits
 - can implement all with NANDs or NORs
 - simplify circuits (not on this course!)

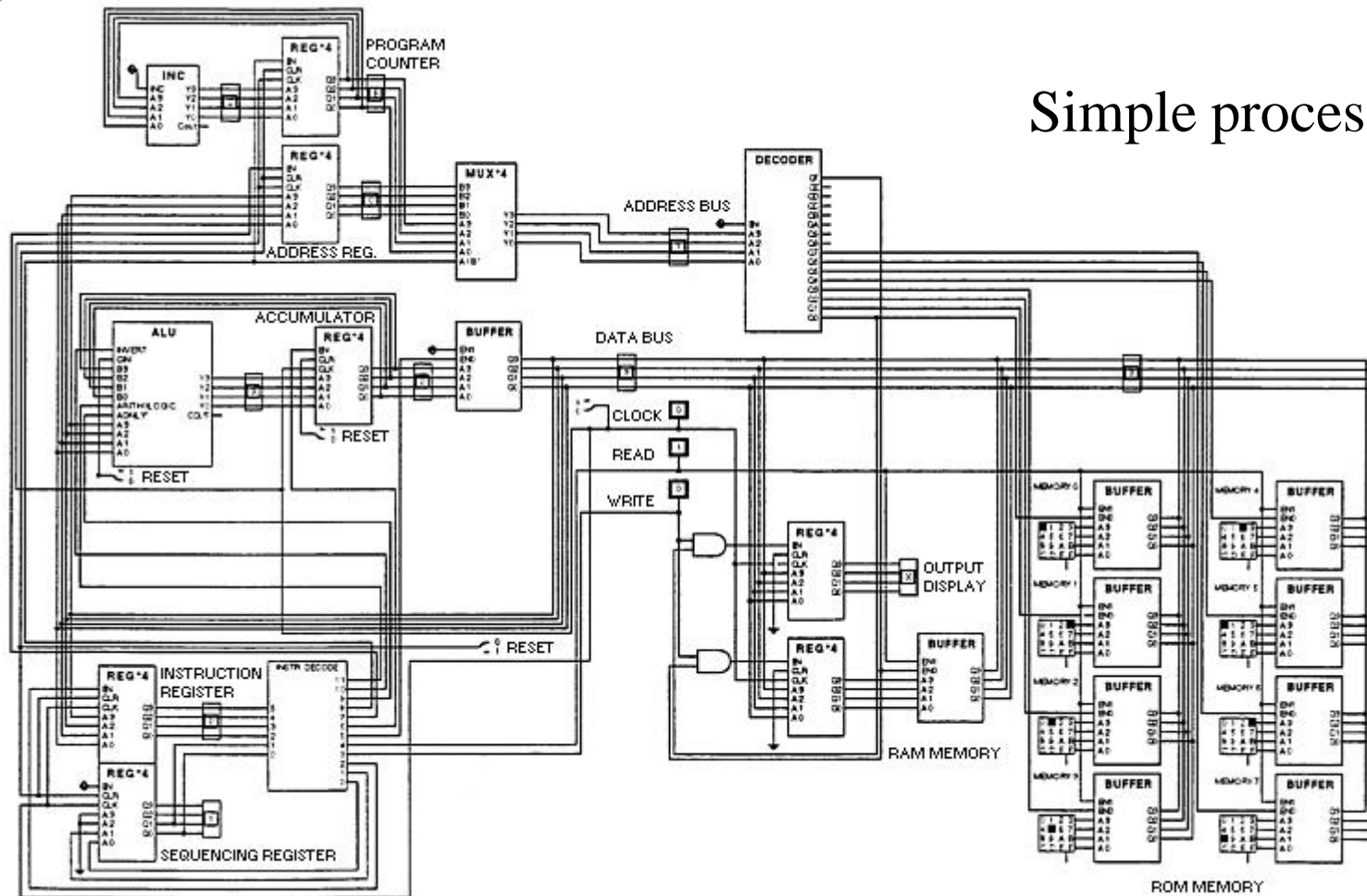
- Components for CPU design
 - ROM, adder
 - multiplexer, encoder/decoder
 - flip-flop, register, shift register, counter

Simulations of gates and circuits:

Hades Simulation Framework: <http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/index.html>



Simple processor



http://www.gamezero.com/team-0/articles/math_magic/micro/stage4.html