TSEA83 : Datorkonstruktion Fö6

Minnen + Bussar

LINKÖPINGS UNIVERSITET

Fö6 : Agenda

- Cache-minnen, BPT
  -Repetition
- Minnen allmänt
  -ROM, RAM, DRAM, SDRAM, DDR
- MMU – Memory Management
- Minnen Nexys3
  -Block-RAM, distributed RAM, DDR3
- Bussar och I/O
  -Vad kan man göra med Nexys3?
  -RS232, I2C, I2S, SPI, Ethernet, USB
  -Parallell kommunikation
- Planering
  -Den närmaste framtiden

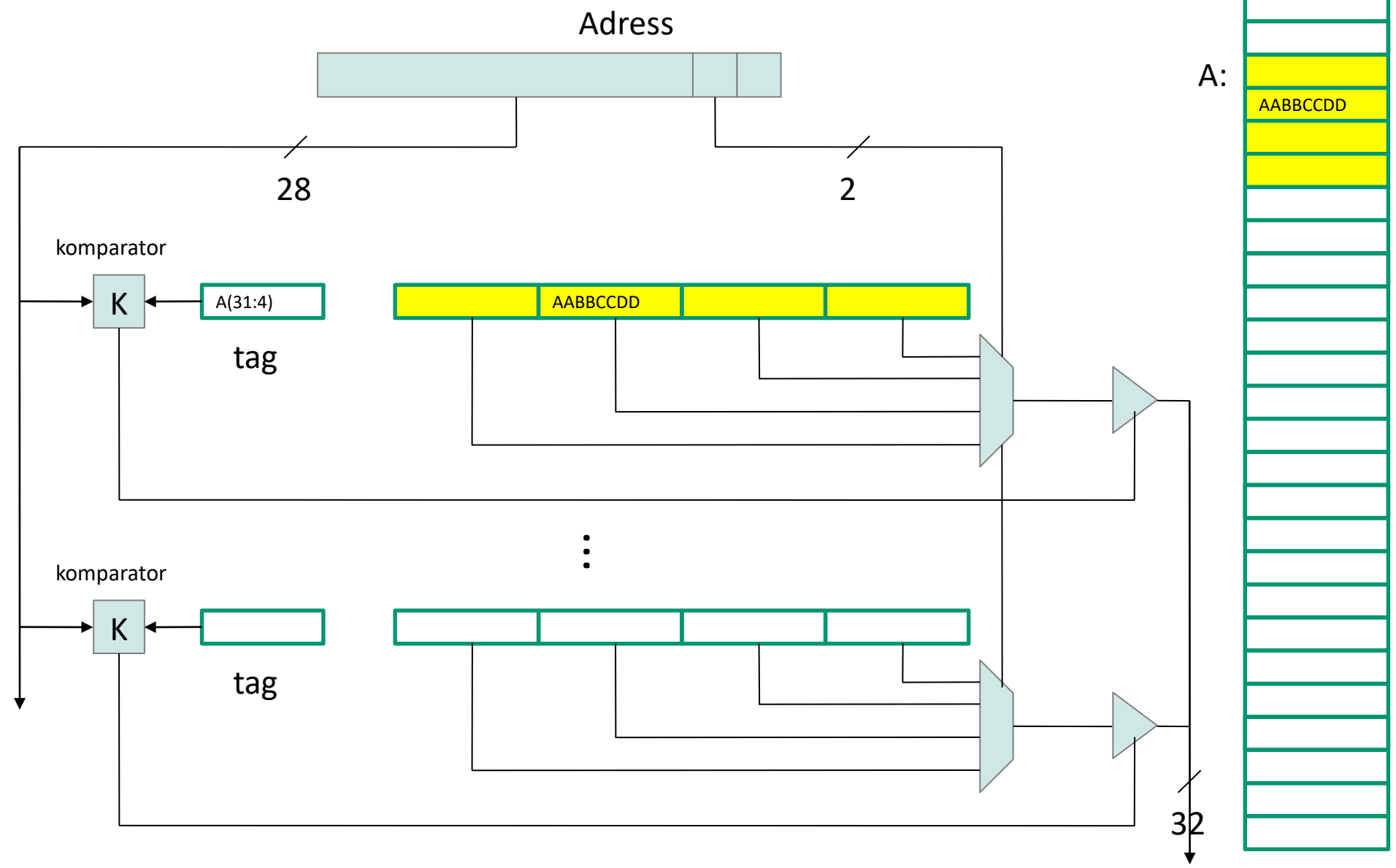
Cache-minnen, BPT

Repetition

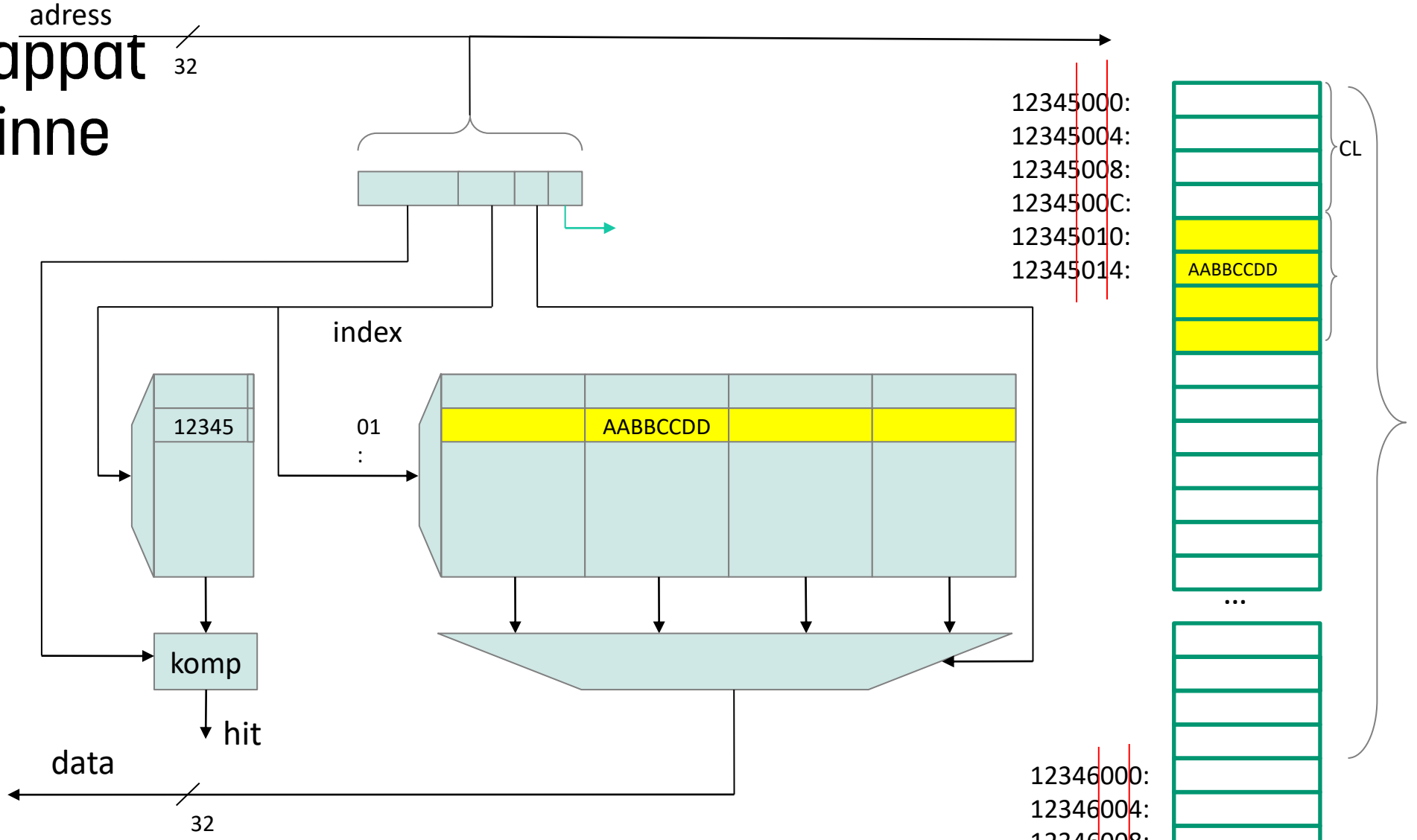
Associativt cacheminne

A= adress till PM/CM
CL = cachelinens storlek
CM = cacheminnets storlek

Tag = A/CL = A/2^4 = A(31:4)

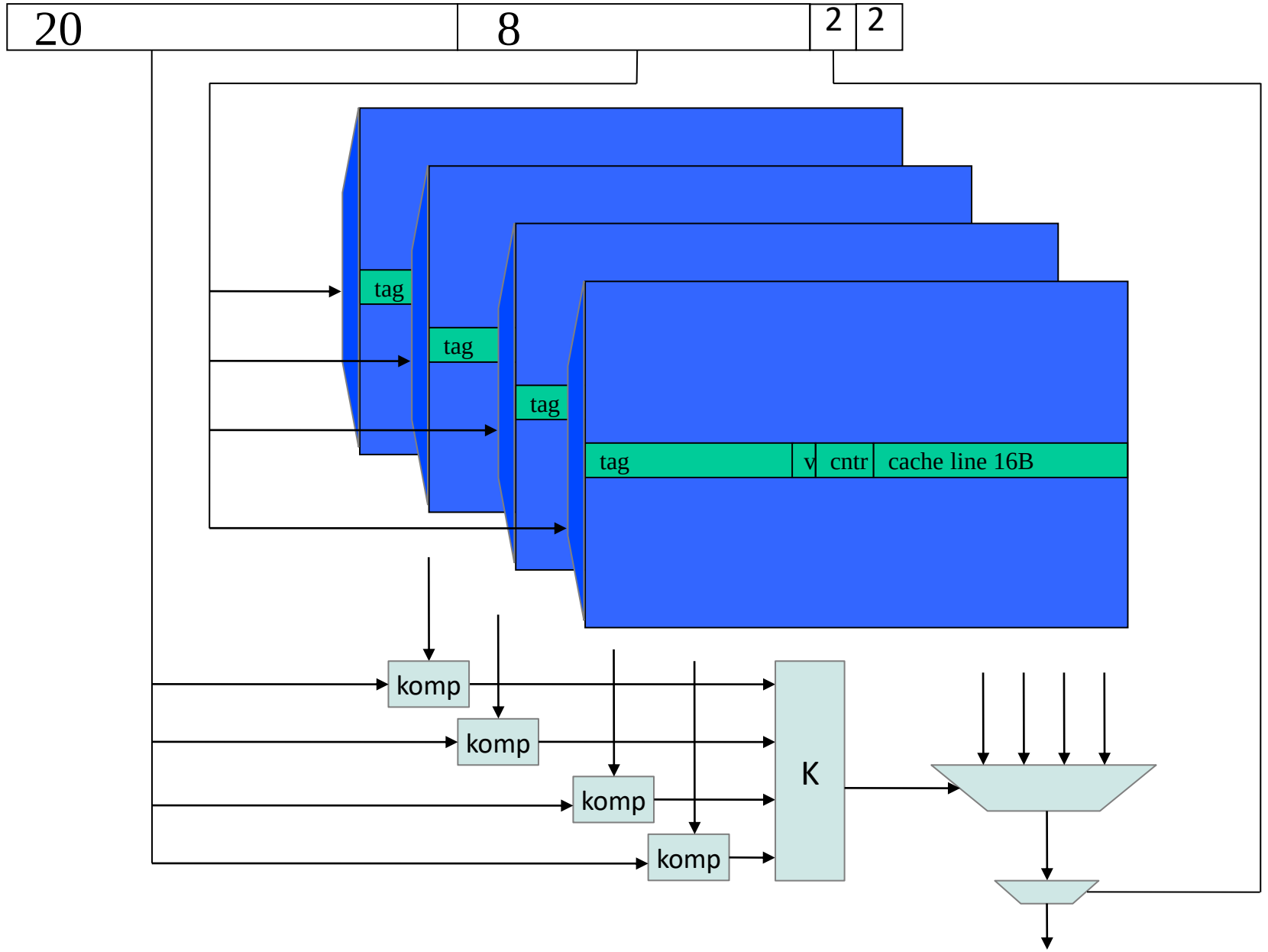


Direktmappat cacheminne



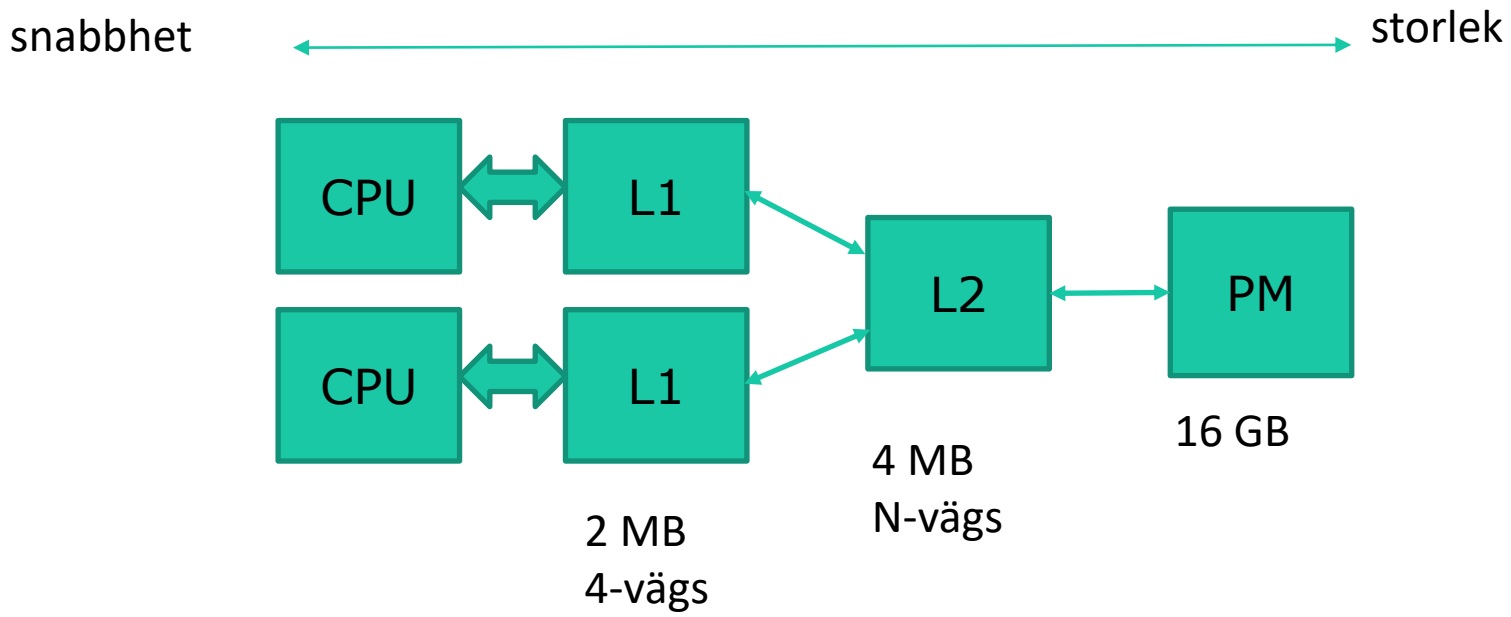
Storlek = 4kB
CL = 4
Index= 12-2-2=8 bitar
Tag=32-12=20 bitar

4-vägs cache 16kB



Cachehierarki

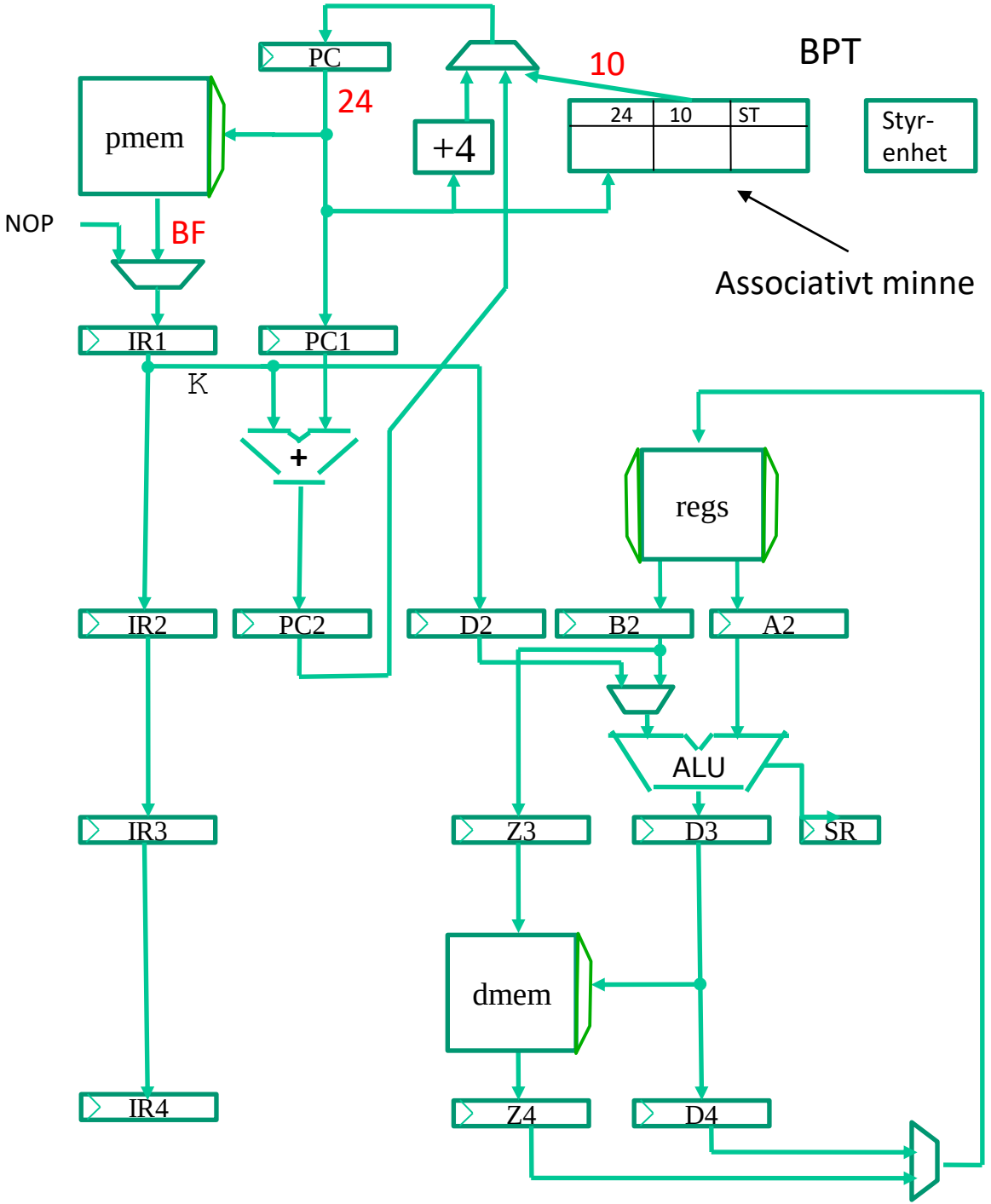
- När PM växer måste också CM växa för att hålla träffkvoten uppe
- Simuleringar visar att det lönar sig att att införa cacheminne i flera nivåer istf att bygga en större L1-cache



BPT

Branch Prediction Table.
Vi väntar inte på F, vi chansar mha BPT!

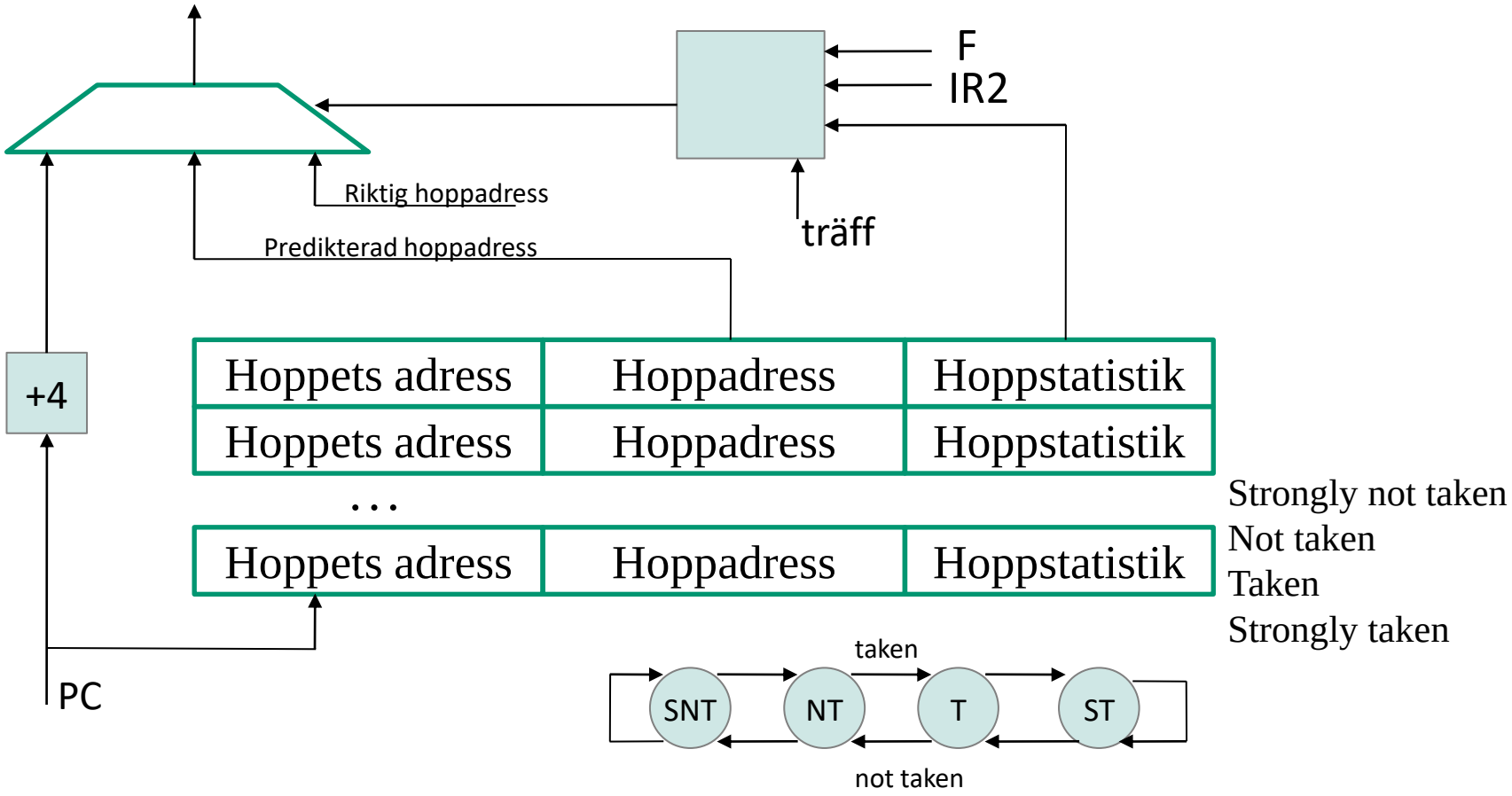
```
LOOP:
10: ZZZ
      ...
1C: ADDI R1,R1,1
20: SFEQ R1,R2
24: BF LOOP
28: XXX
2C: YYY
```



Ett försök att få bort den där NOP-en vid hopp

Skiss: Varje gång ett hopp påträffas

1) Finns inte i BPT: stoppa in det och uppdatera hoppstatistiken
2) Finns i BPT: läs ut hoppadressen



BPT

Pipelinediagram

PC	IR1	IR2	IR3	IR4
24	SFEQ	ADDI	...	...
10	BF	SFEQ	ADDI	...
14	ZZZ	BF	SFEQ	ADDI
28	NOP	NOP	BF	SFEQ

← BPT->PC, 28->SPC

← F=0 => SPC->PC

Rätt predikterat : Vi sparar 2 klockcykler, XXX går aldrig in i pipelinen

Den där NOP-en försvinner

Fel predikterat: Vi förlorar 2 klockcykler

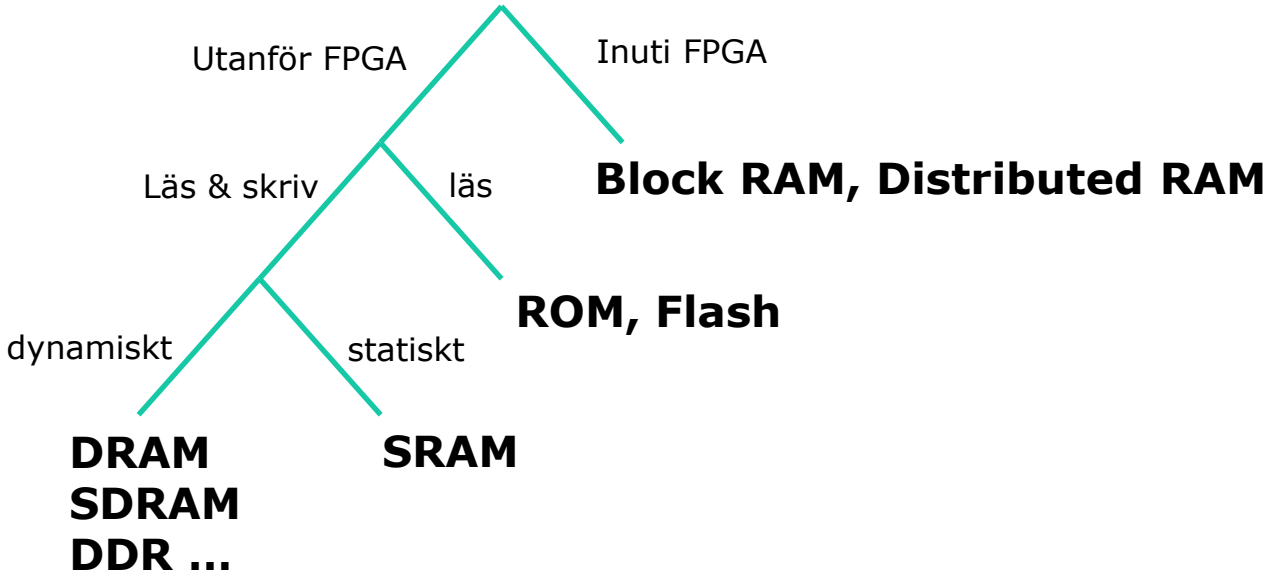
Minnen, allmänt

ROM, RAM, DRAM, SDRAM, DDR

Minnen allmänt

Några förkortningar:

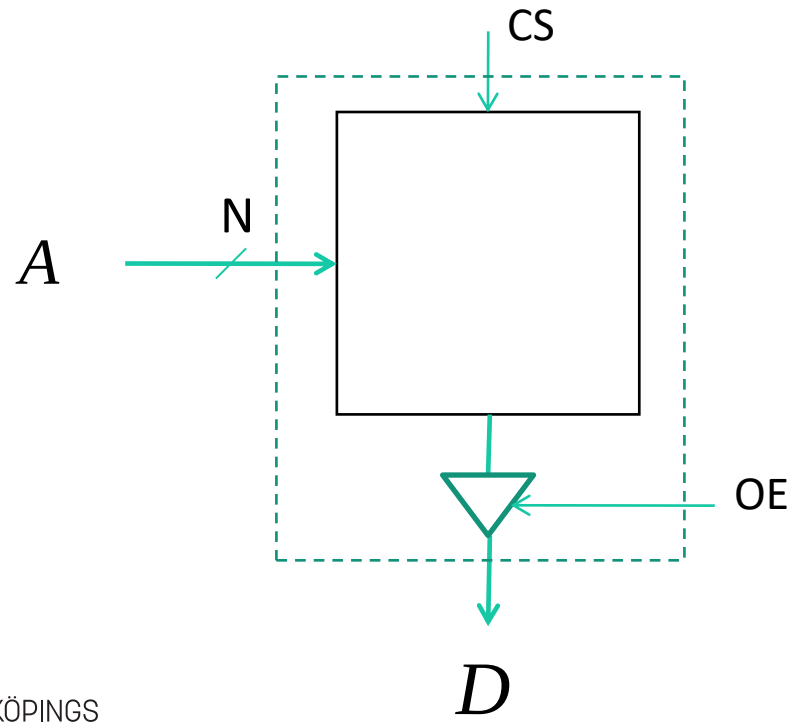
ROM = read only memory
RAM = random access memory
SRAM = static RAM
DRAM = dynamic RAM
SDRAM = synchronous DRAM
DDR = double data rate DRAM



Minnen ROM

Läsminnen

ROM = Read Only Memory
PROM = Programmable ROM
UV EPROM = UV Erasable PROM
Flash = Electrically EPROM = EEPROM

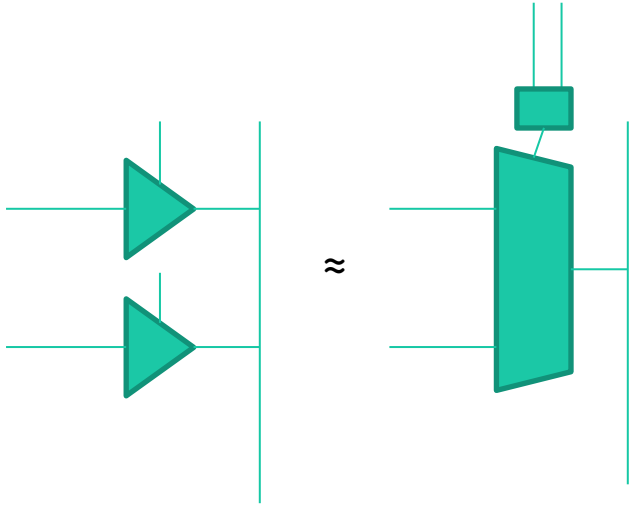


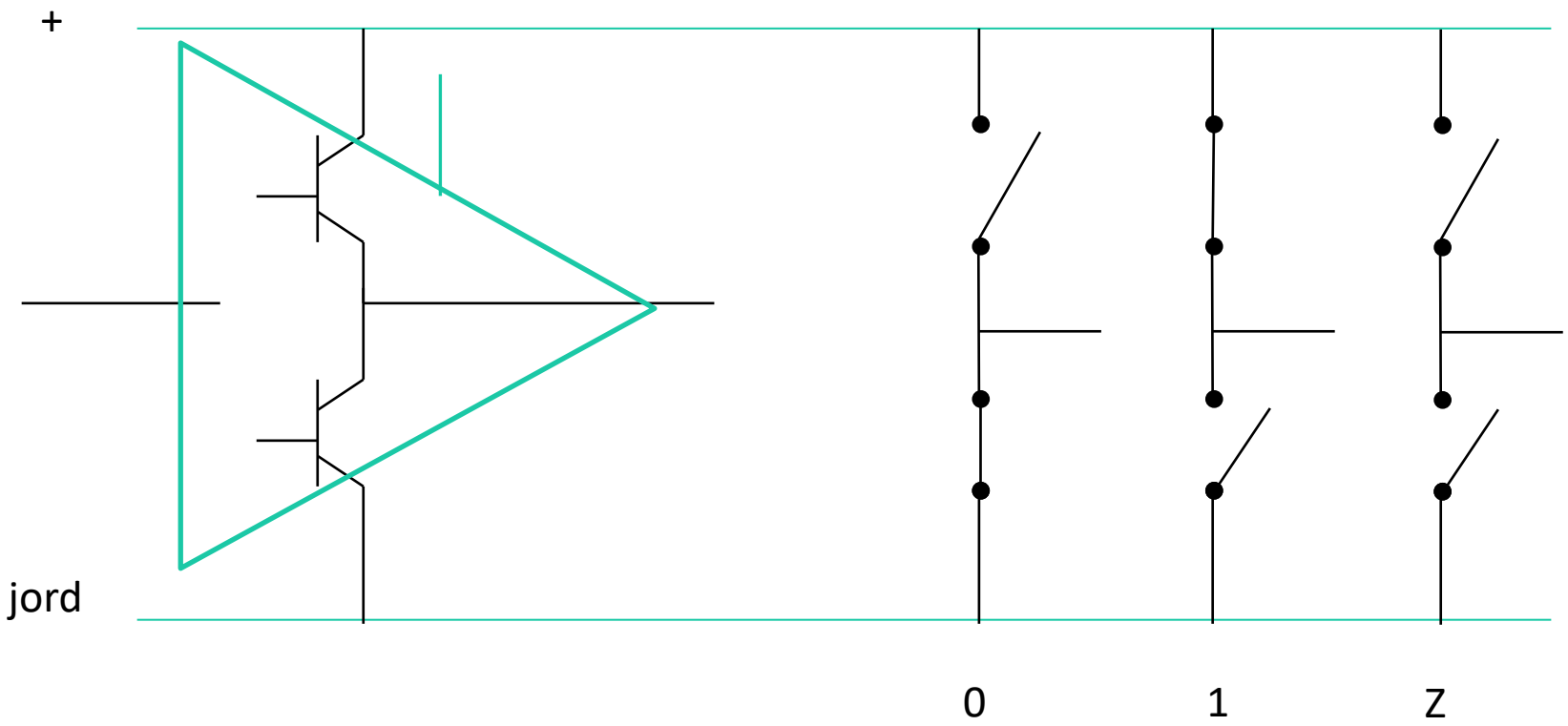
N st adressingångar
M st dataingångar
1 st chip select-ingång (aktivt låg)
1 st output enable-ingång (aktivt låg)

Organisation: 2^N ord à M bitar

Tristate-buffer

1) Förbättrar signalen
2) Möjlighet till tristate

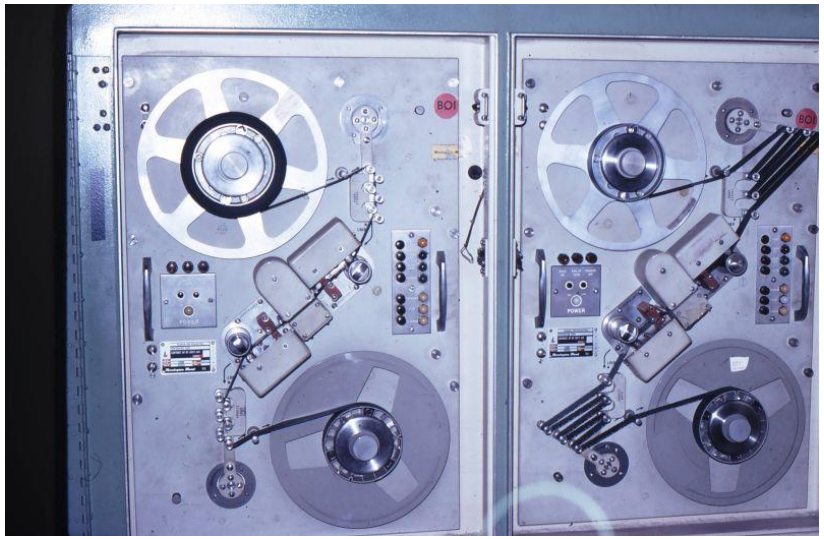




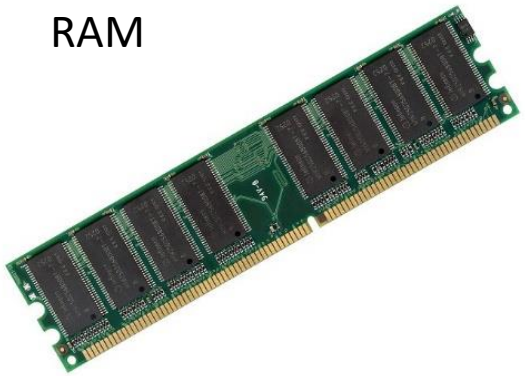
RAM – Random Access Memory

Vaddå random?

Tape memory



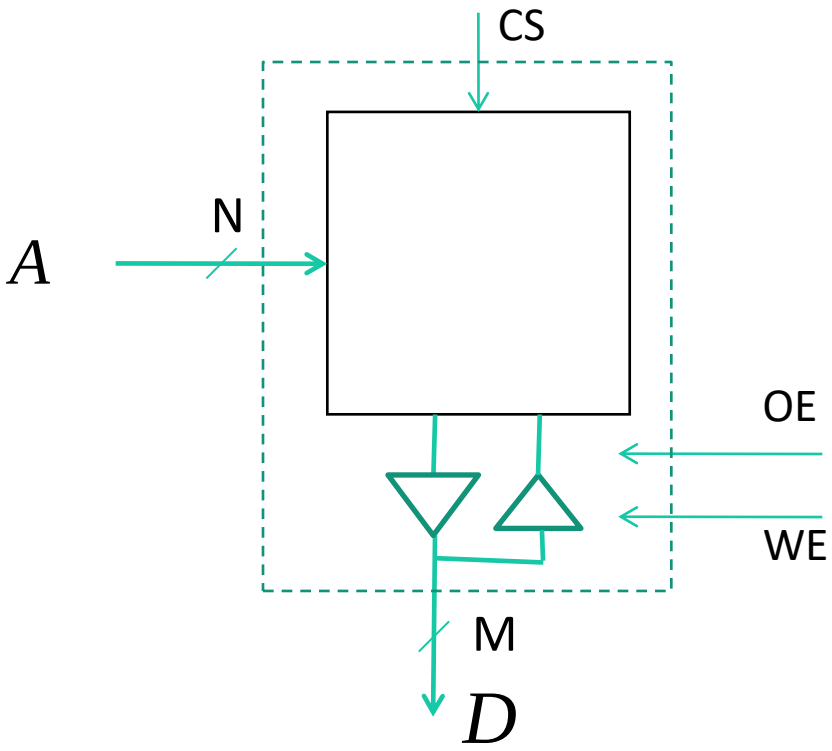
RAM



För att nå ett visst data på magnetbandet måste det spolas fram till rätt plats, vilket gör att acesstiden för ett godtyckligt data blir att betrakta som slumpmässig.

SRAM – Static RAM

Asynkron => ingen klocka
Varken kombinatorik eller synkront sekvensnät



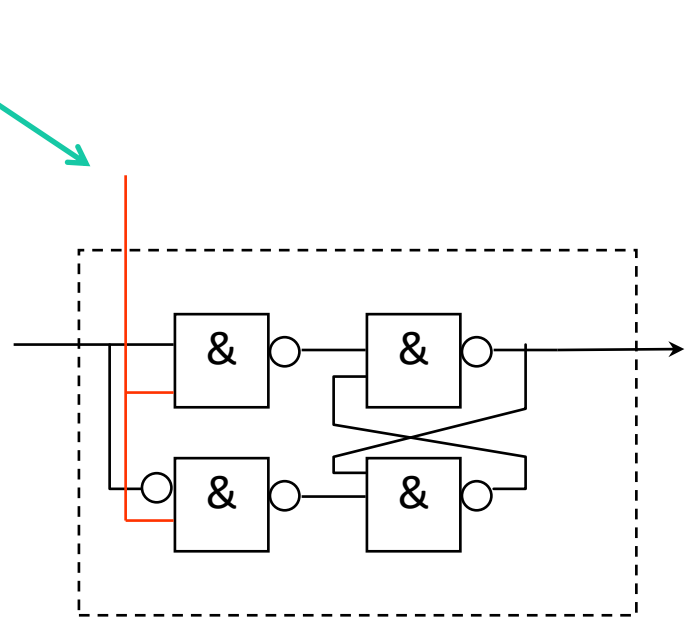
N st address-ingångar
M st data-in/utgångar
1 st chip select-ingång (aktivt låg)
1 st output enable (aktivt låg)
1 st write enable (aktivt låg)

Organisation: 2^N ord à M bitar

SRAM – Static RAM

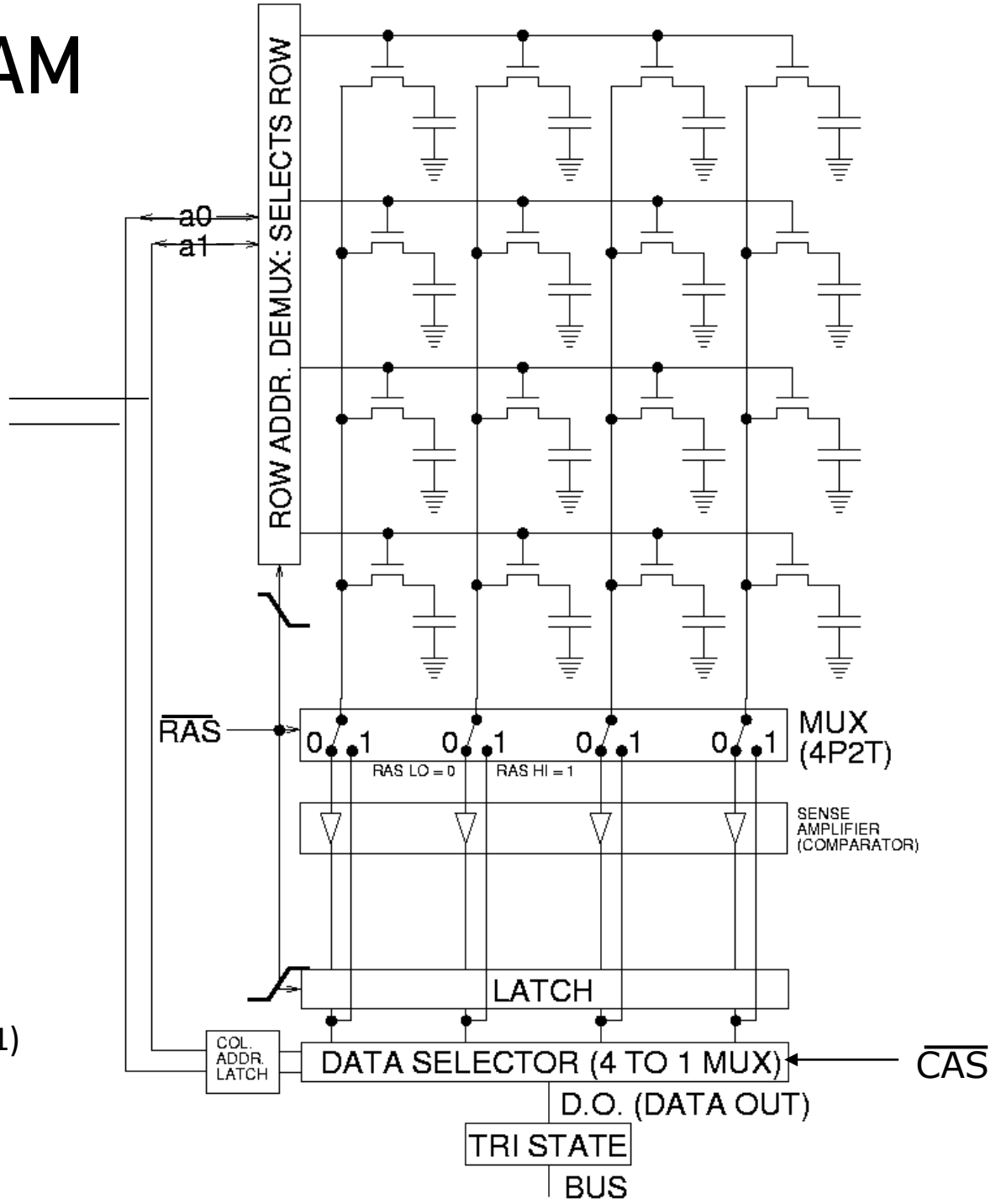
Läs/skriv , minnet försvinner utan spänning

ME = latch, 6 transistorer
skrivning sker hela tiden
då denna signal är hög



DRAM – Dynamic RAM

KL = kolumnledning
KA= kolumnadress
RL = radledning
RA = radadress
SA = sense amplifier

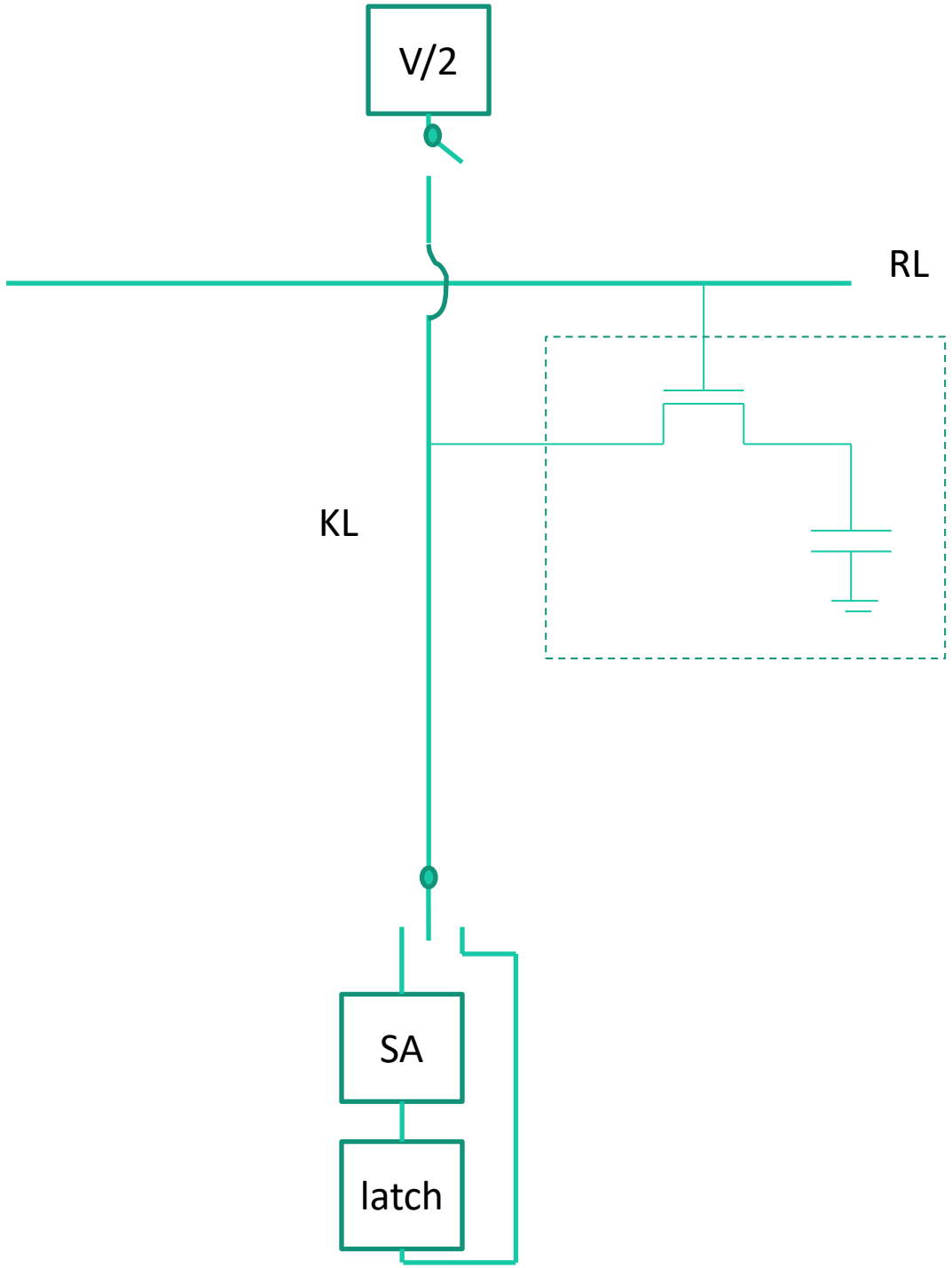


Läscykel

1. Förladda KL till V/2
2. Låt KL flyta fritt
3. Driv radledningen till V
4. Koppla in SA till KL
5. Håll KL i latchen
6. Återställ laddning (omkopplaren=1)
7. Släpp ut värde på bussen

DRAM – Dynamic RAM

KL = kolumnledning
KA= kolumnadress
RL = radledning
RA = radadress
SA = sense amplifier

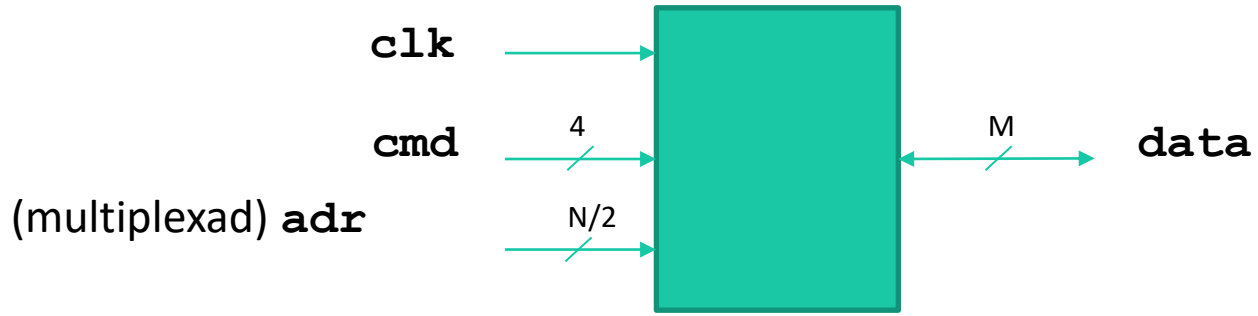


Läscykel

1. Förladda KL till V/2
2. Låt KL flyta fritt
3. Driv radledningen till V
4. Koppla in SA till KL
5. Håll KL i latchen
6. Återställ laddning (omkopplaren=1)
7. Släpp ut värde på bussen

SDRAM – Synchronous DRAM

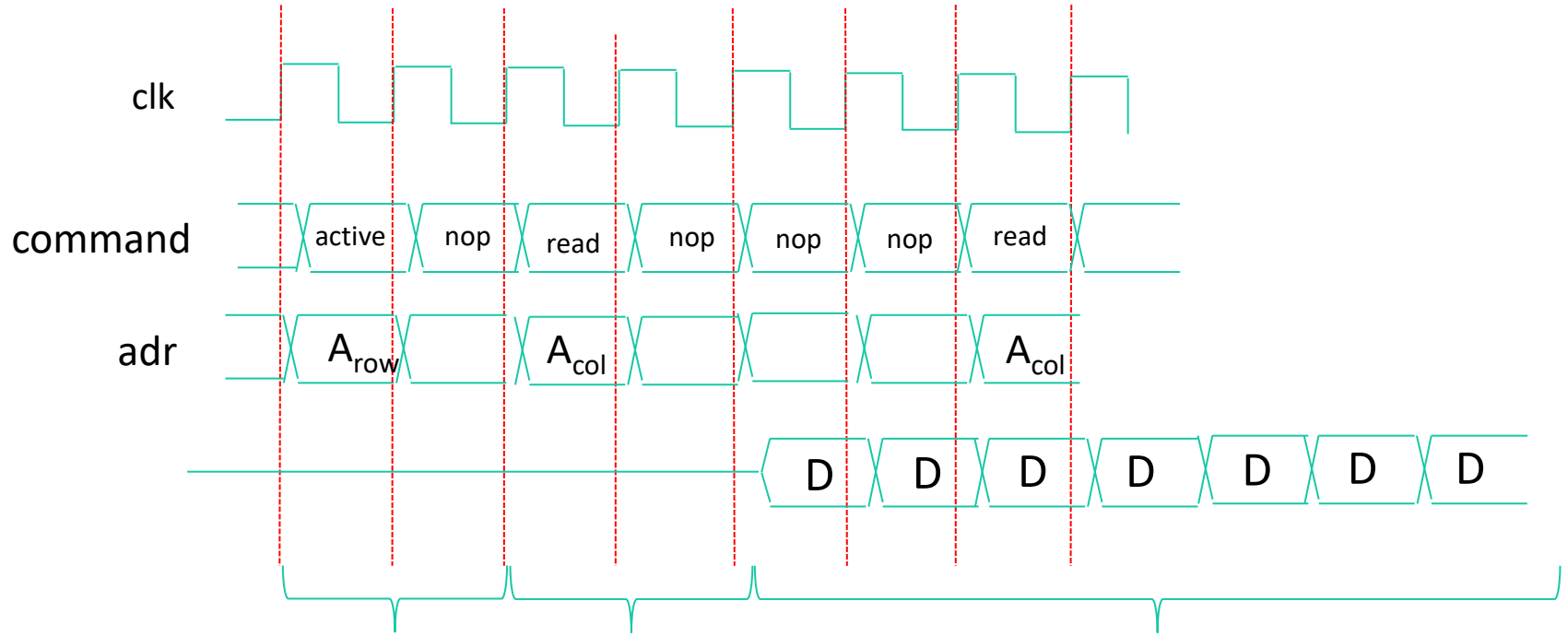
- Dominerar helt
- 1-transistorcellen => 1 läs/skriv-transistor + 1 kapacitans
- Synkron komponent



- Multiplexad adress
- Pipelinead och burstorienterad =>
  en serie av **cmd** och **adr** måste skickas in för
  att få ut första datat. Därefter kan data fås på varje klockflank.

SDRAM – Synchronous DRAM

Read burst



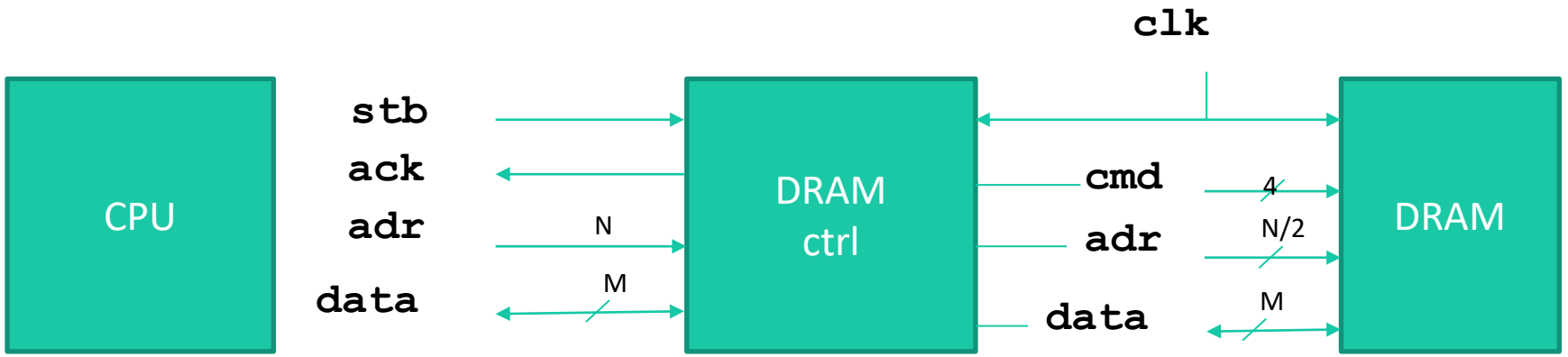
Läs en hel rad

Muxa ut bitar

Så länge vi är på samma rad är allt bra!

SDRAM – Synchronous DRAM

- Kan knappast användas utan DRAM-controller. Numera finns den i CPU-n
- Eftersom kapacitanserna läcker, måste periodiska sk refresh-cykler köras hela tiden (dummy read från en rad)

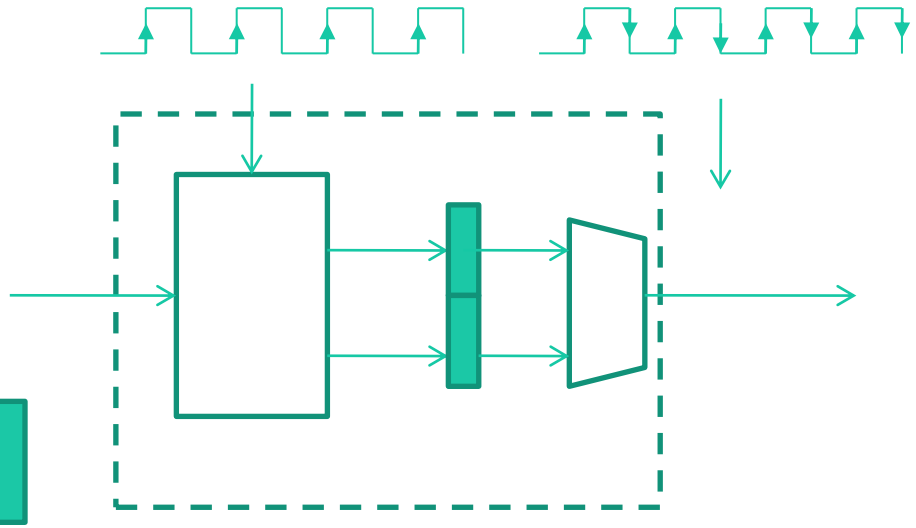


Refreshcykler har högst prioritet,
CPU måste ibland vänta genom att **ack** fördröjs

DRAM-utveckling

- SDRAM = synkront DRAM
  - klockat,
  - controller på chipet, programmerbart,
  - burst-orienterat, pipelineat
- DDR (double data rate) SDRAM
- DDR2 SDRAM (4x)
- DDR3 SDRAM (8x)
- DDR4 SDRAM (16x)

  …



MMU – Memory Management Unit

MMU – Memory Management Unit

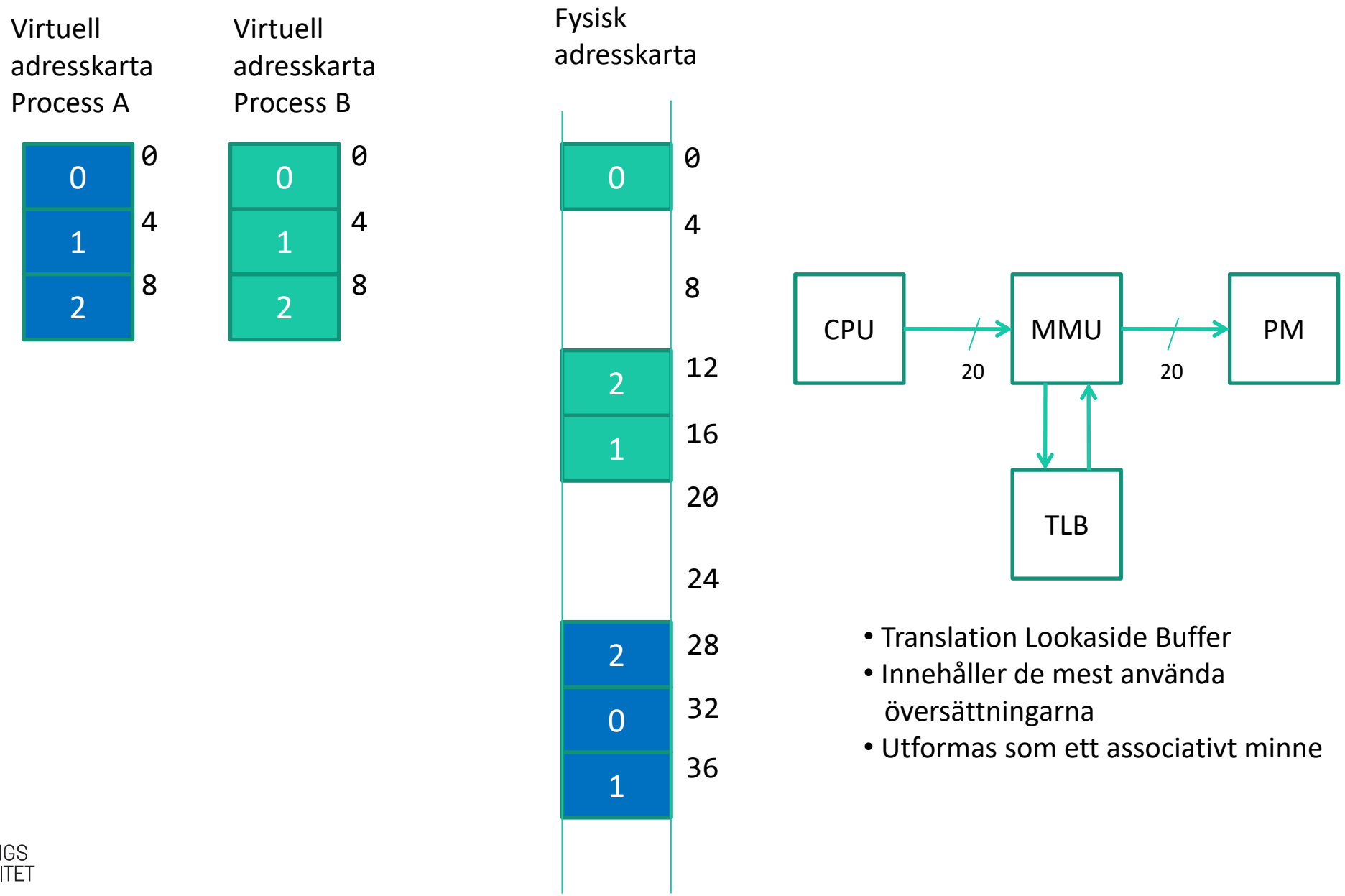
- Om vi ska köra ett OS (tex Linux) på vår processor, så behövs memory management!
- I PM finns då OS och flera processer
- Vi behöver:
  - minnesskydd
  - adressöversättning (virtuell/logisk => fysisk)

    Processer skapas/försvinner

    Processer gör malloc/free → minnesfragmentering

- Vanligast är att dela in PM i sidor (pages) av fix storlek. Tex 4kB
- OS tilldelar sig själv och varje process ett antal sidor.
- Detta administreras av OS i sk page tables, som finns i PM.
- Här lagras översättningen, skrivskydd, exekveringsskydd, ...
- I detta exempel är OS, mikroprogram, hårdvara inblandat.

MMU – Memory Management Unit, 4kB pages

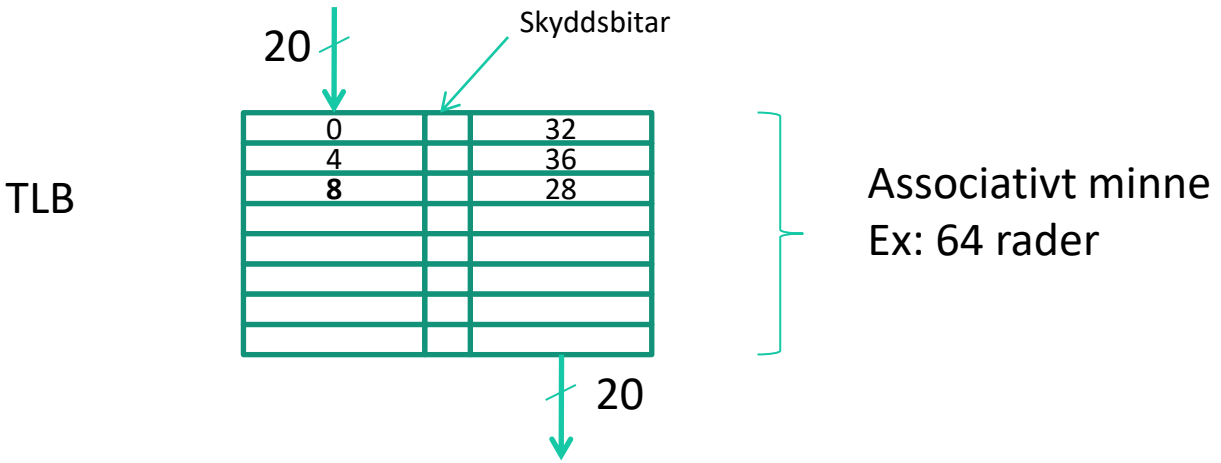


- Translation Lookaside Buffer
- Innehåller de mest använda översättningarna
- Utformas som ett associativt minne

MMU – Memory Management Unit, 4kB pages

Process A har alltså fått 3 sidor.
Process A kan inte komma åt några andra sidor.

Två processer kan dela en sida. Man kan då skrivskydda sidan för den ena processen.

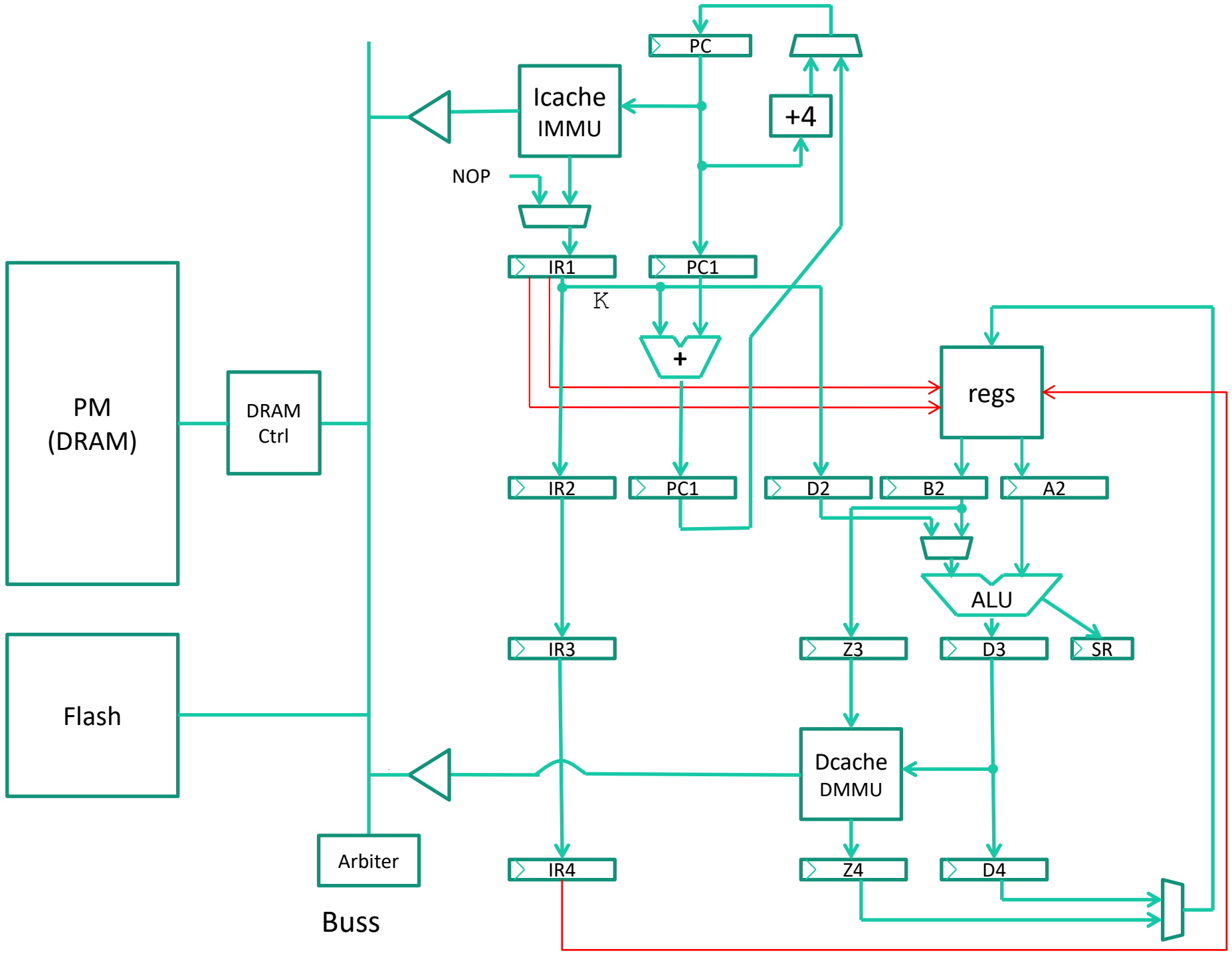


MMU – Memory Management Unit, 4kB pages

Exempel: Istället för programminnet stoppar vi in föregående konstruktion i vår 5-stegs pipeline:

- PC innehåller nu en virtuell adress
- vid cache hit returneras direkt sökt instruktion
- vid cache miss hämtar styrenheten en cacheline från PM och fyller på rätt rad i cachen. Vår pipeline stallar ...
- vid TLB miss hämtar styrenheten en adressöversättning från ”page tables” i PM. Vår pipeline stallar ...

MMU


PC
Icache
IMMU
+4
NOP
IR1
PC1
K
+
regs
PM
(DRAM)
DRAM
Ctrl
IR2
PC1
D2
B2
A2
ALU
IR3
Z3
D3
SR
Flash
Dcache
DMMU
Arbiter
IR4
Z4
D4
Buss

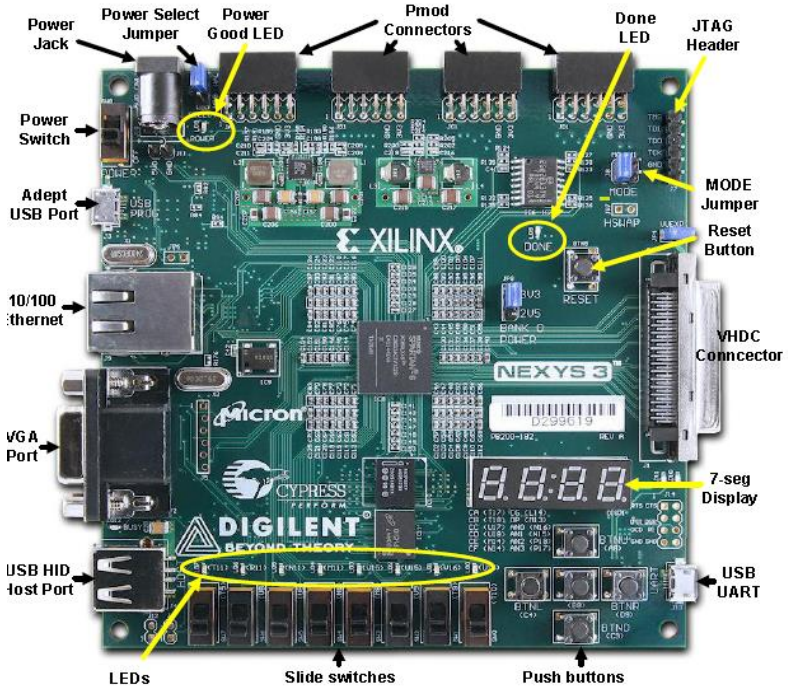
Minnen, Nexys3

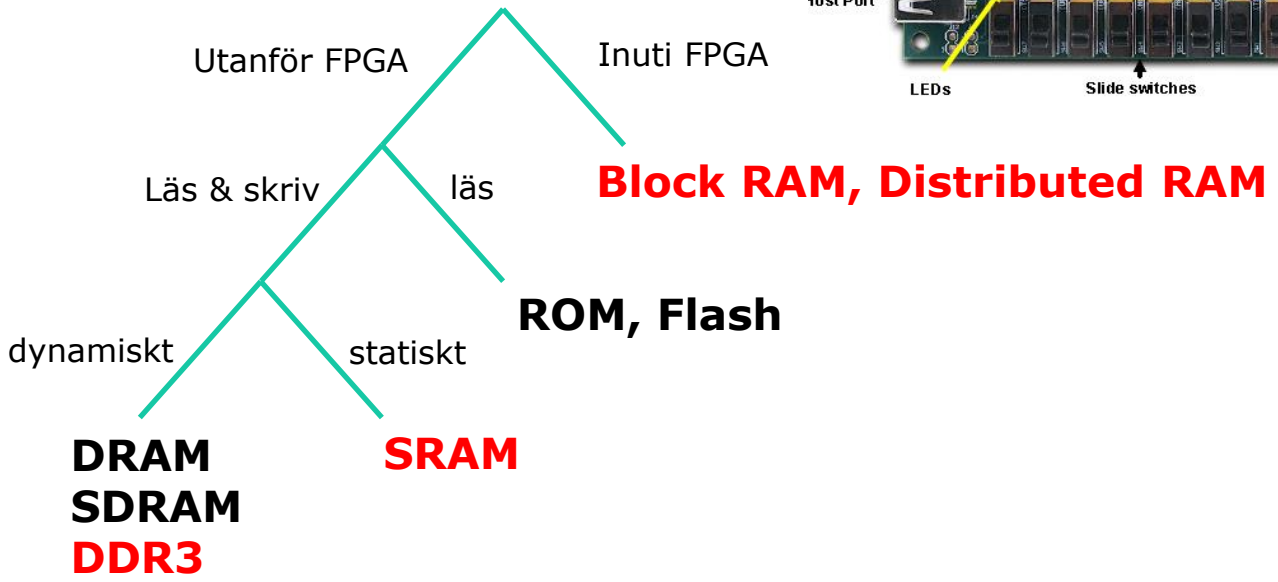
Block-RAM, distributed RAM, DDR3

Minnen med ankn. Till Nexys3

Några förkortningar:

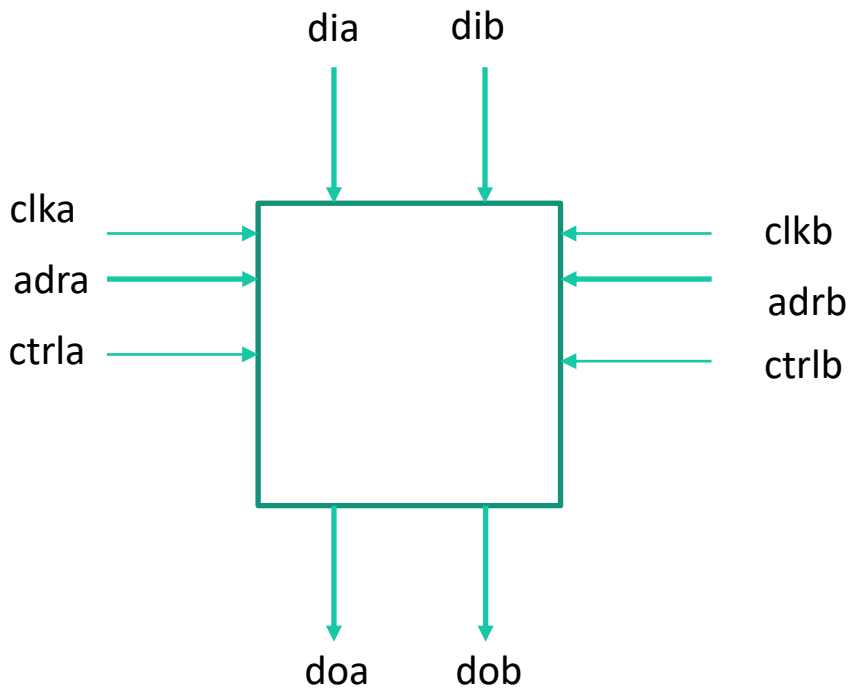
ROM = read only memory
RAM = random access memory
SRAM = static RAM
DRAM = dynamic RAM
SDRAM = synchronous DRAM
DDR = double data rate DRAM





Block-RAM

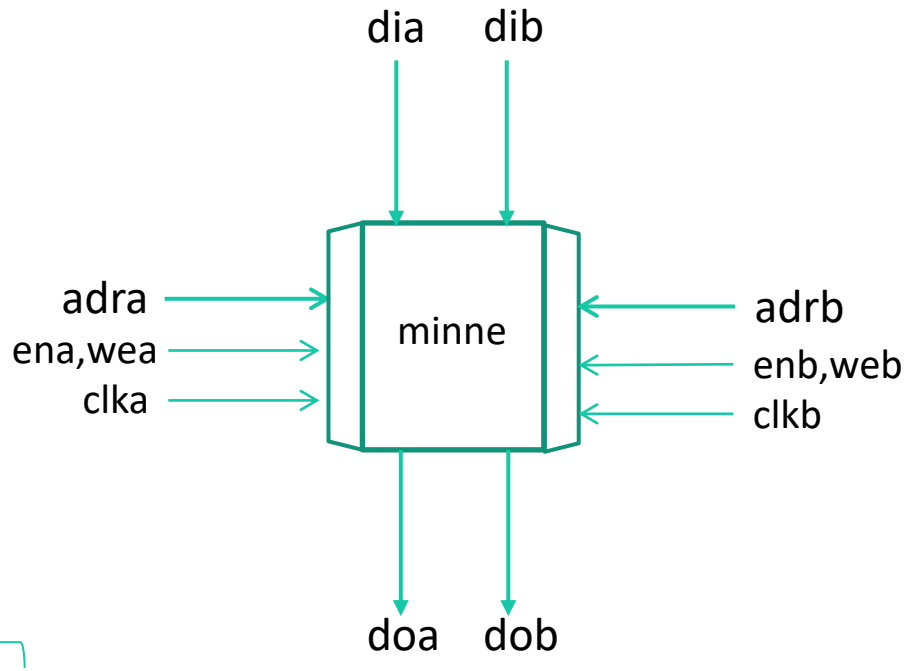
- I FPGA:n finns 32 st synkrona RAM à 2 kB
- Jag rekommenderar: beskriv minnet på hög nivå i VHDL (som en array). Syntesverktyget bygger då ditt minne av befintliga komponenter.

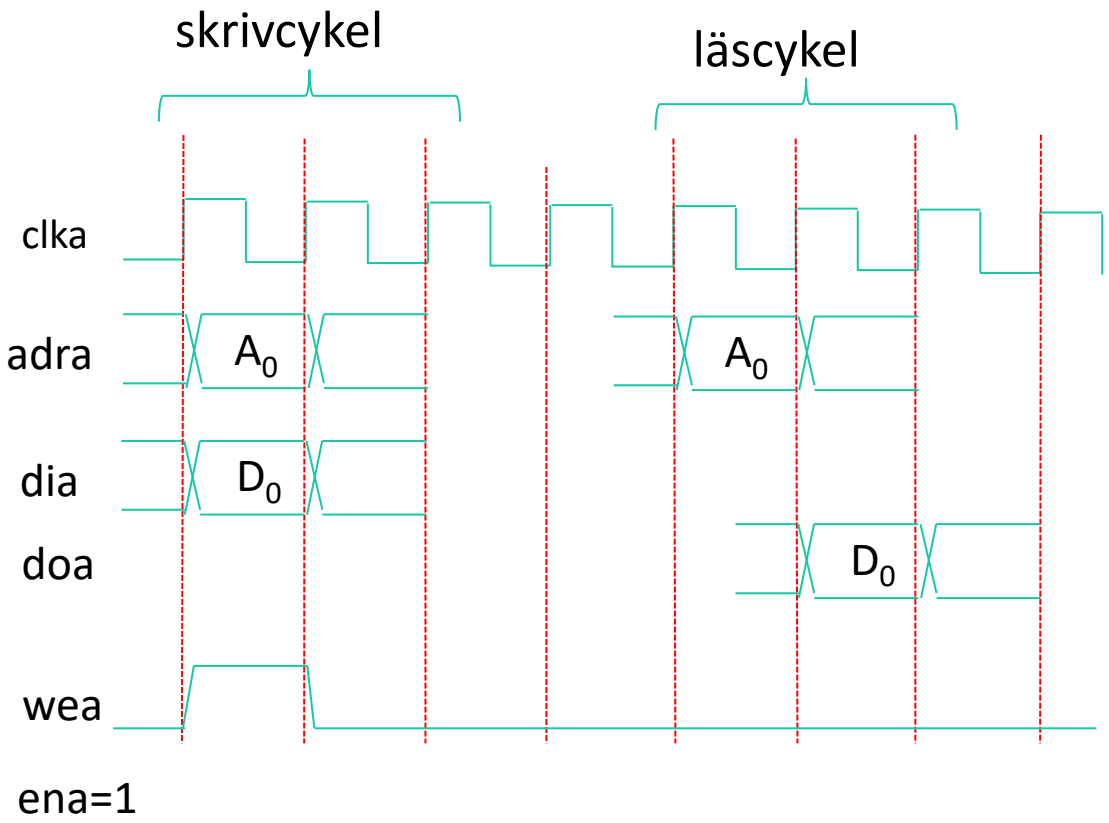


BRAM är tvåportsminne.
A och B är helt oberoende.
Vi kan skriva på A-sidan och läsa på B-sidan samtidigt.

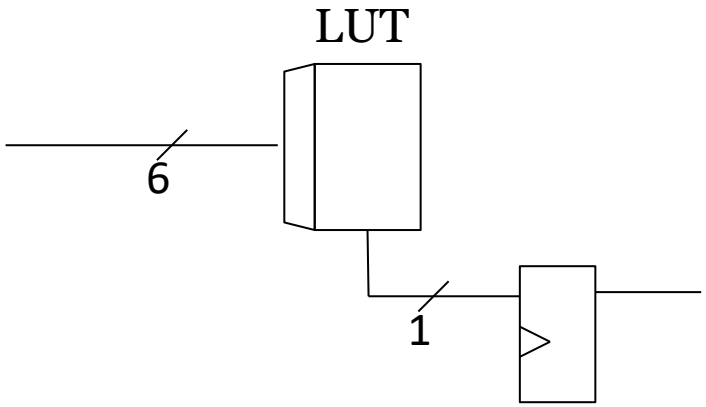
Block-RAM

- Både läscykel och skrivcykel är synkrona!



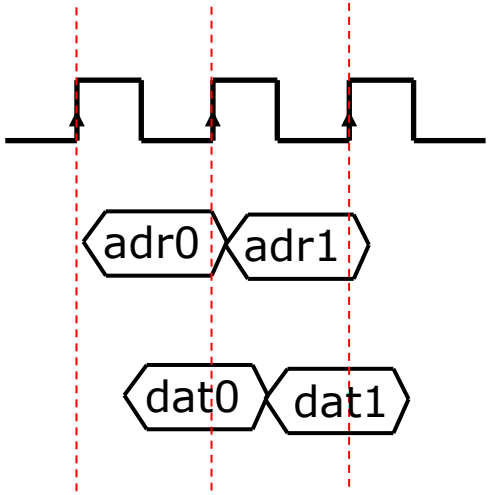


Distributed RAM

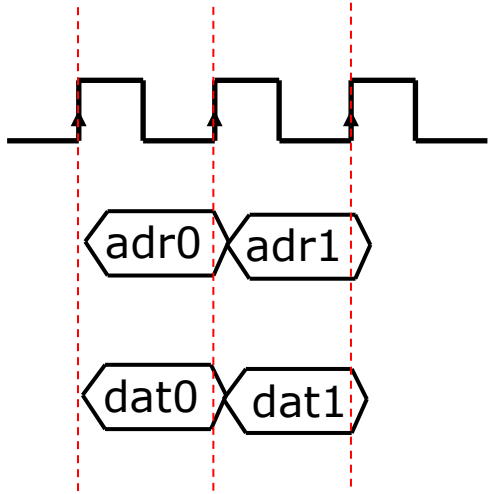


Logikblocken består av vippor och LUT-ar.
Syntesverktyget kan bygga små minnen av LUT-arna.

Kombinatorisk läsning

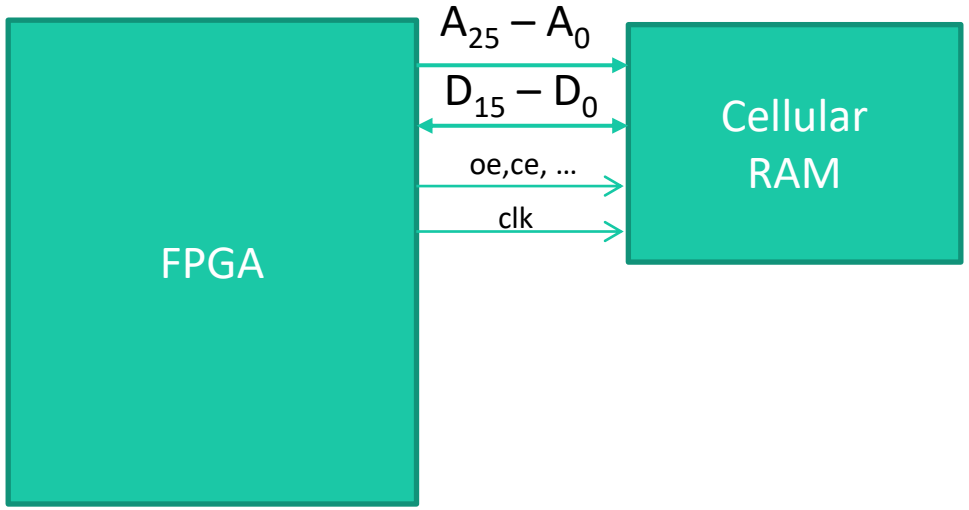


Synkron skrivning



FPGA-kortet

- Har 16 MB Cellular RAM
- DRAM med Controller, ser ut som SRAM
- Cycle time 70 ns => 7 CK med 100 MHz klocka

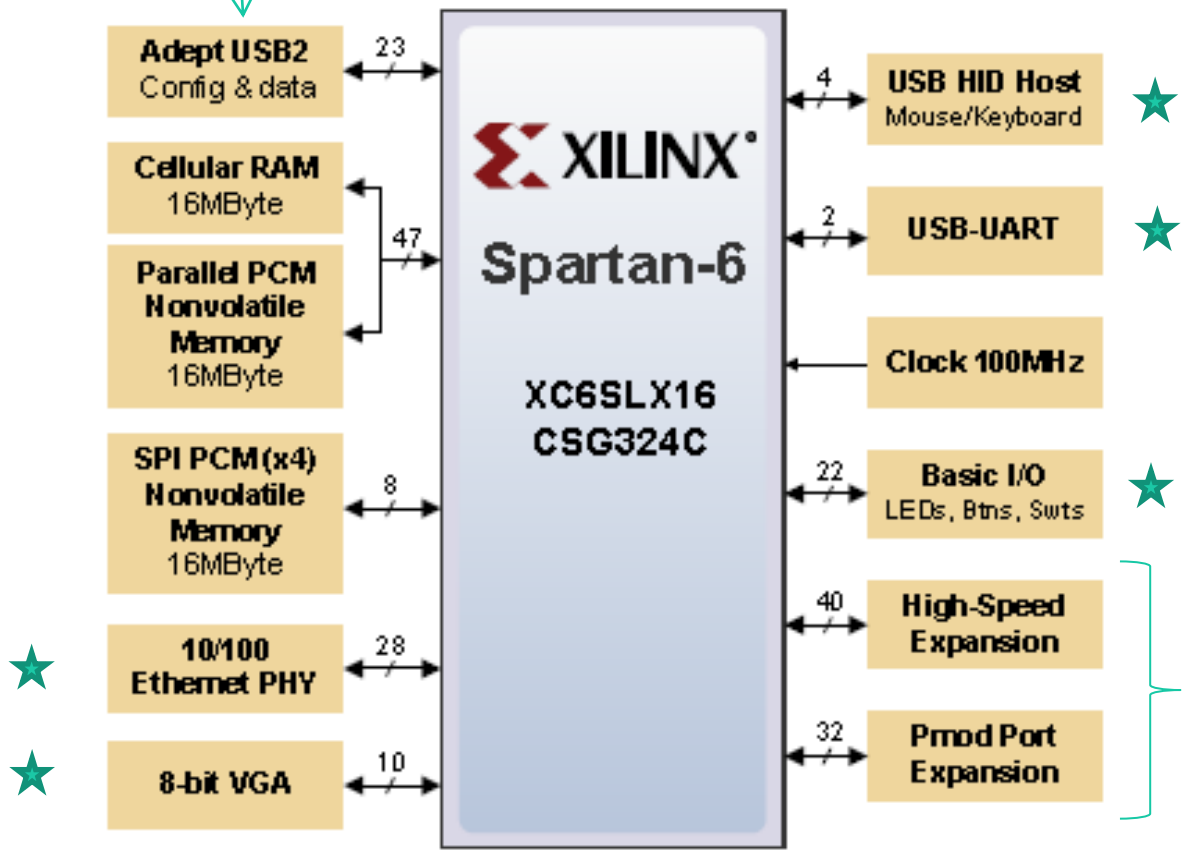


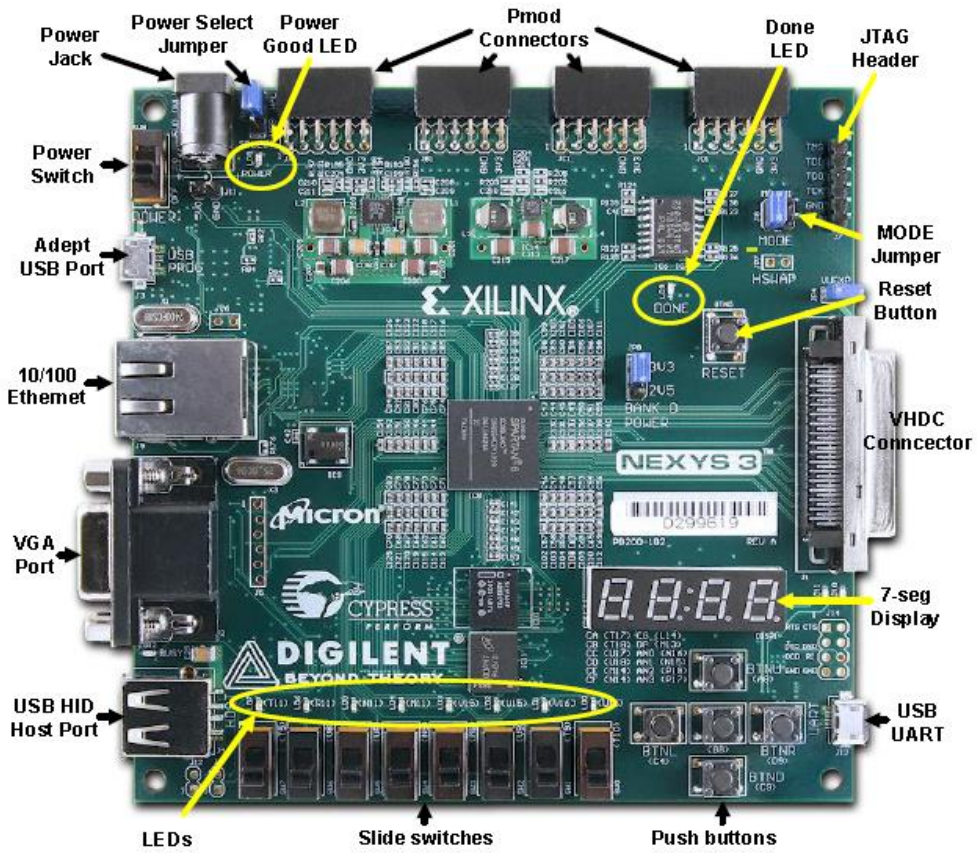
Bussar och I/O, vad kan man göra med Nexys3?

RS232, I2C, I2S, SPI, Ethernet, USB
Parallell kommunikation

Nexus3

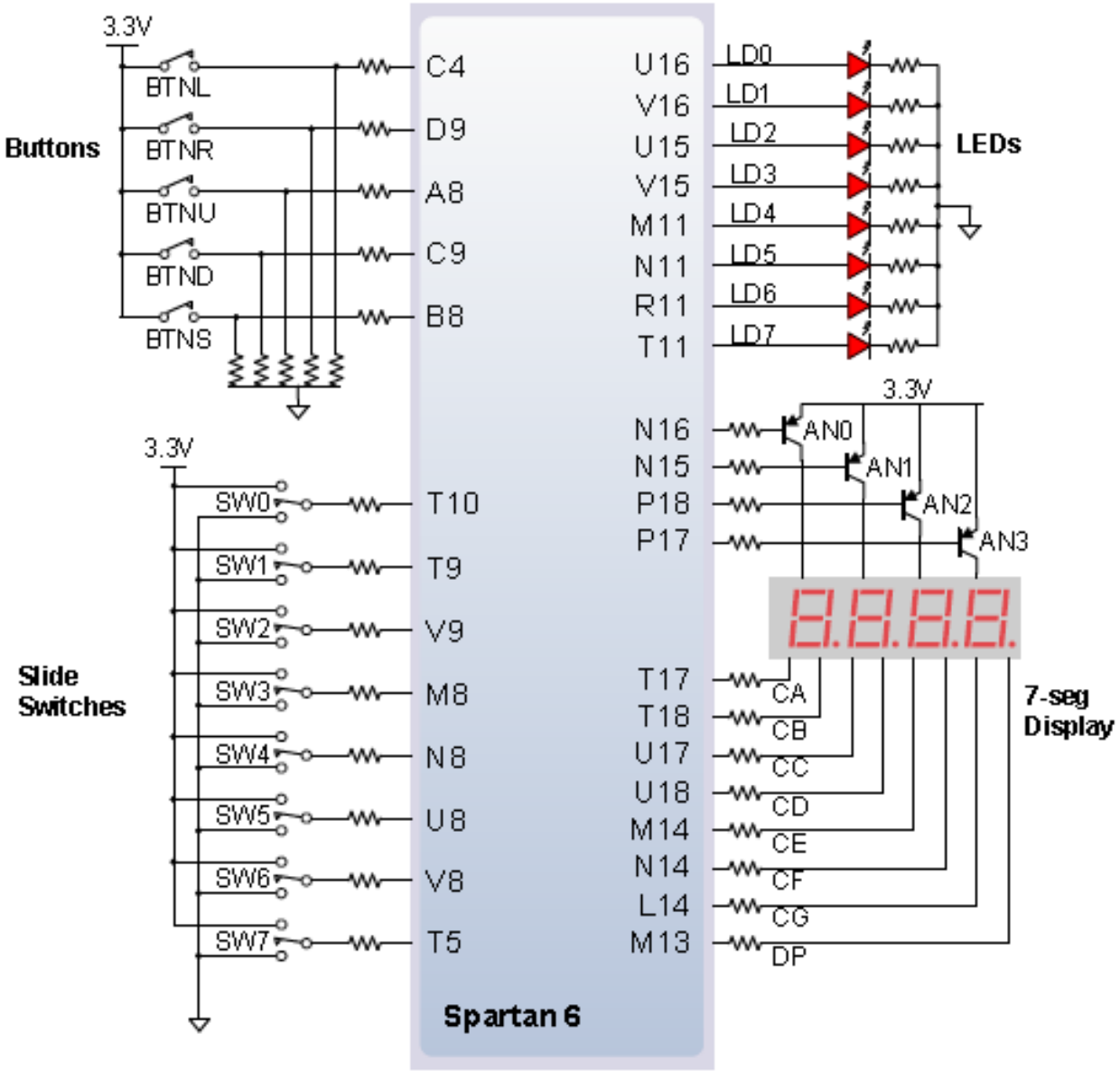
Prog + spänningsmatning
mha USB





Expansions-
kort

Basic I/O


3.3V
BTNL
BTNR
BTNU
BTND
BTNS
Buttons
C4
D9
A8
C9
B8
LD0
LD1
LD2
LD3
LD4
LD5
LD6
LD7
U16
V16
U15
V15
M11
N11
R11
T11
LEDs
3.3V
SW0
SW1
SW2
SW3
SW4
SW5
SW6
SW7
Slide Switches
T10
T9
V9
M8
N8
U8
V8
T5
3.3V
AN0
AN1
AN2
AN3
N16
N15
P18
P17
CA
CB
CC
CD
CE
CF
CG
DP
T17
T18
U17
U18
M14
N14
L14
M13
7-seg Display
Spartan 6

Master.ucf (User Constraints File)

```
## 7 segment display
Net "seg<0>" LOC = T17 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L51P_M1DQ12, Sch name = CA
Net "seg<1>" LOC = T18 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L51N_M1DQ13, Sch name = CB
Net "seg<2>" LOC = U17 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L52P_M1DQ14, Sch name = CC
Net "seg<3>" LOC = U18 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L52N_M1DQ15, Sch name = CD
Net "seg<4>" LOC = M14 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L53P, Sch name = CE
Net "seg<5>" LOC = N14 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L53N_VREF, Sch name = CF
Net "seg<6>" LOC = L14 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L61P, Sch name = CG
Net "seg<7>" LOC = M13 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L61N, Sch name = DP

Net "an<0>" LOC = N16 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L50N_M1UDQSN, Sch name = AN0
Net "an<1>" LOC = N15 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L50P_M1UDQS, Sch name = AN1
Net "an<2>" LOC = P18 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L49N_M1DQ11, Sch name = AN2
Net "an<3>" LOC = P17 | IOSTANDARD = LVCMOS33; #Bank = 1, pin name = IO_L49P_M1DQ10, Sch name = AN3
```

I VHDL-koden

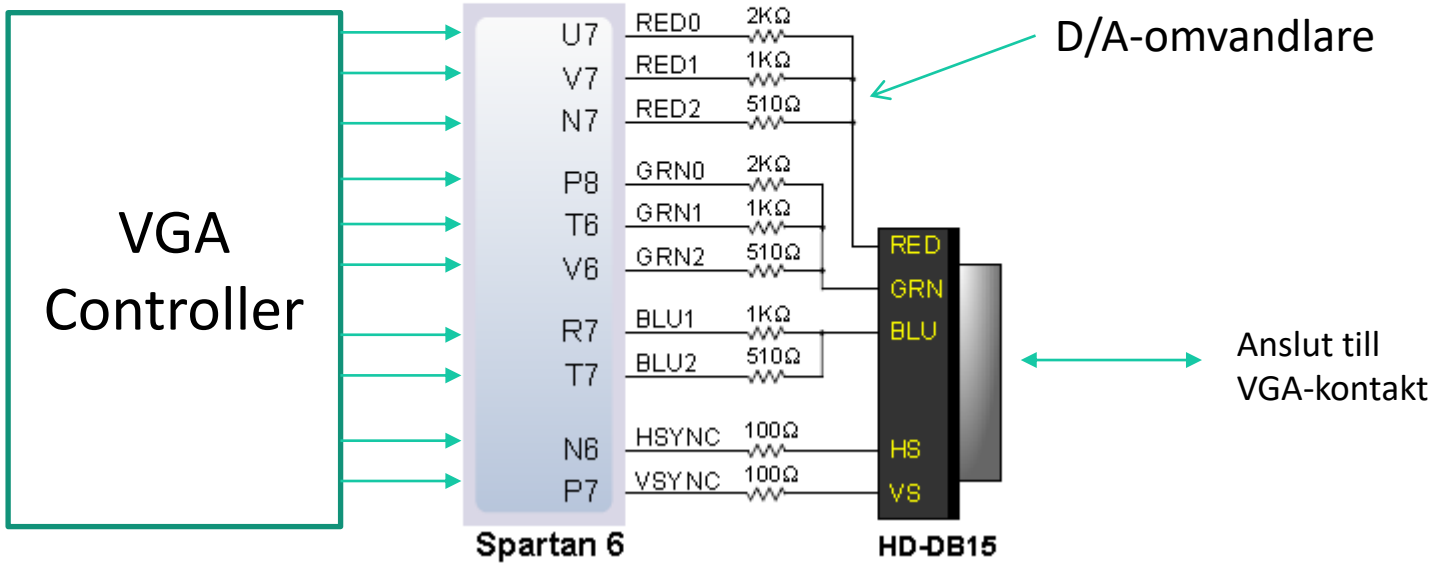
```
seg <= ”10110000”; -- siffran 3
an <= ”1110”;      -- längst till vänster
```

VGA port

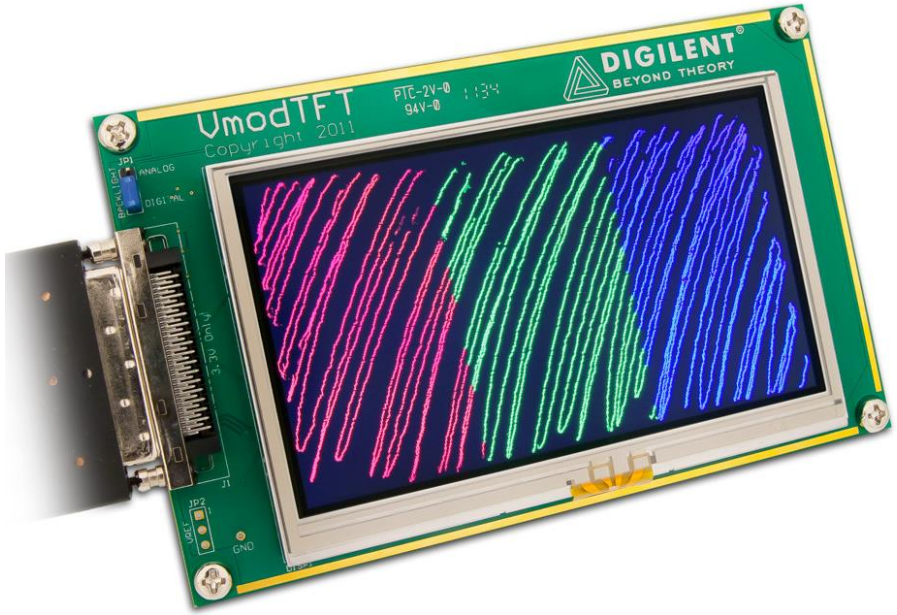
I manualen finns siffror för 640*480-bild

Komplexitet:

- Två räknare + lite avkodning s k VGA-motor
- …
- Avancerad GPU som kan programmeras



En Vmod

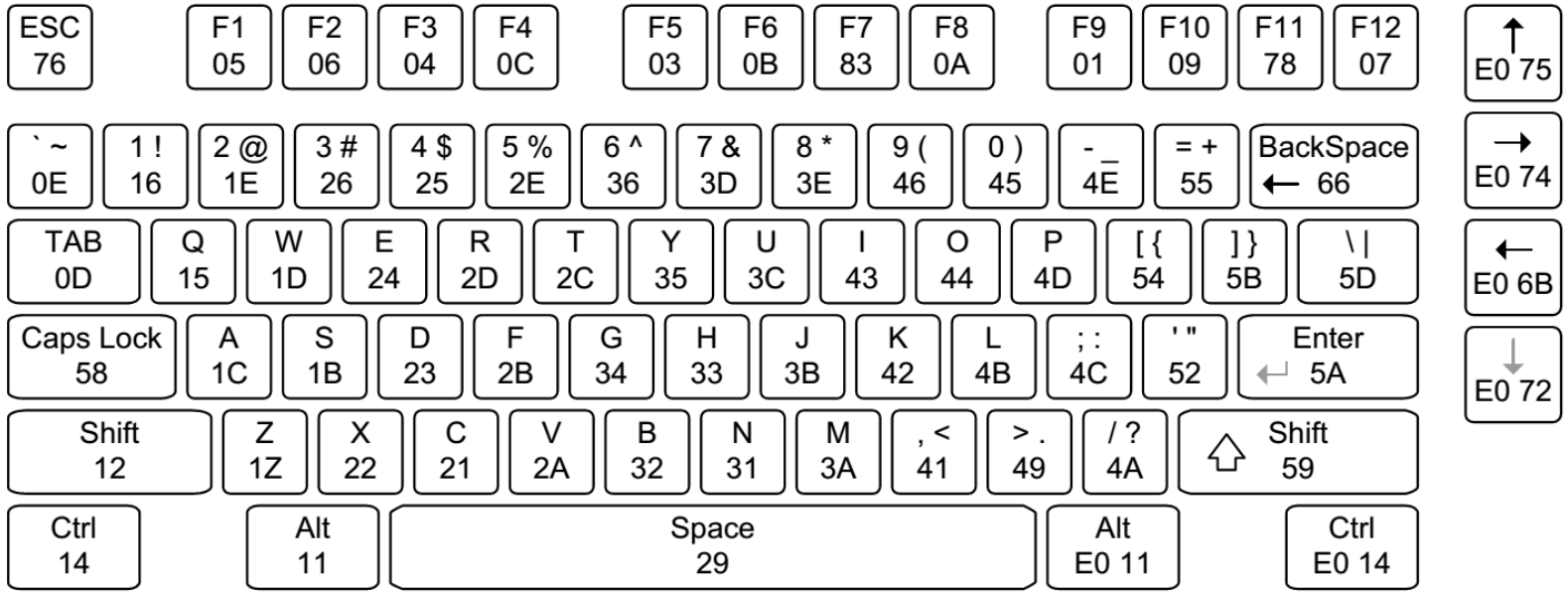


- LCD 480*272
- Resistive touchscreen
- Gränsnitt: se datablad!
- Styrs på ung. samma sätt som en bildskärm, dvs med synksignaler

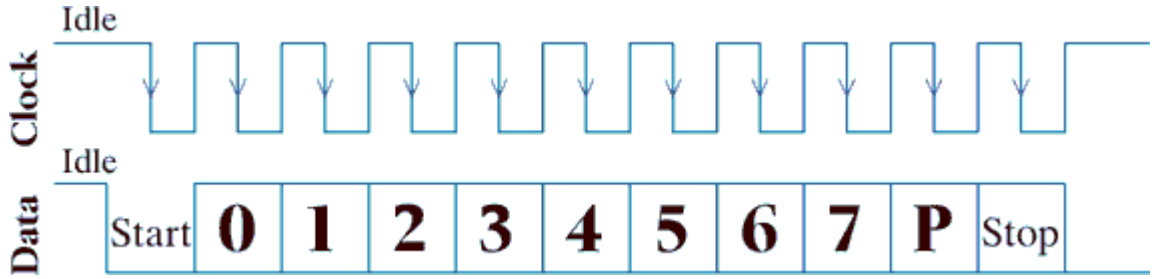
PC-Tangentbord

- Ansluts via USB
- Kommunicerar med PS/2
- Tangentkombinationer kan användas
- Avkodas med tillståndsmaskin

PC-Tangentbord



PS/2 Keyboard Scan Codes



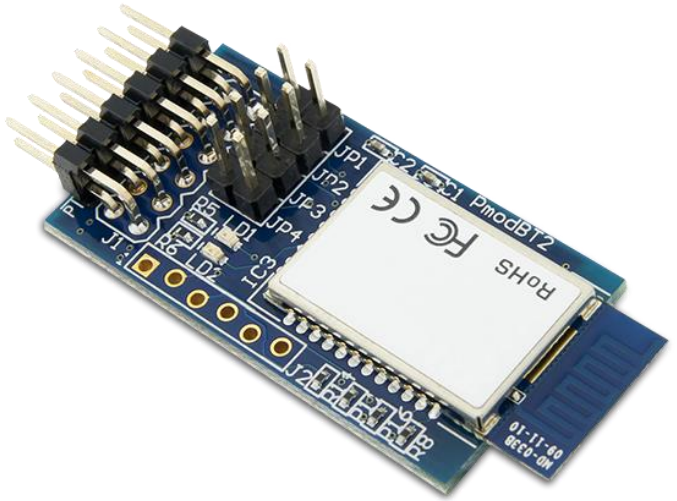
Tryck A ➔ 1C (make)
Tryck R ➔ 2D (make)
Släpp R ➔ F0,2D (break)
Släpp A ➔ F0,1C (break)

Några Pmods

HEX-tangentbord

- 16 tangenter, 0-F
- Saknar inbyggd kontroller, behöver ”scannas”
  drivrutin finns

BT2



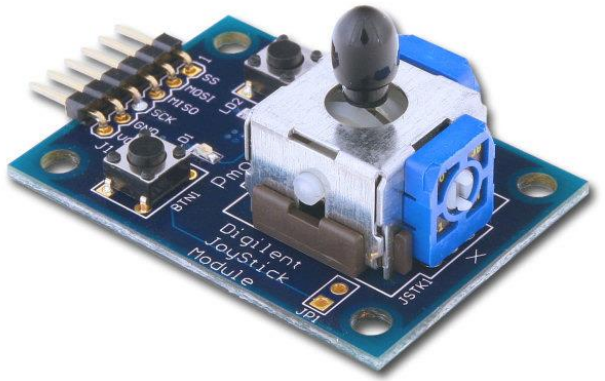
- Blåtandsmodul
- UART-kommunikation

Några Pmods

OLED

- Monokrom
- Text / grafik 128*32
- Controller
- SPI

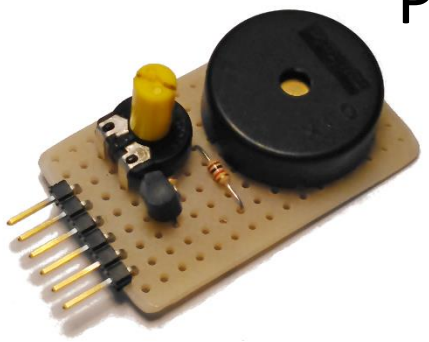
JSTK



- Two axis joystick
- Controller
- SPI

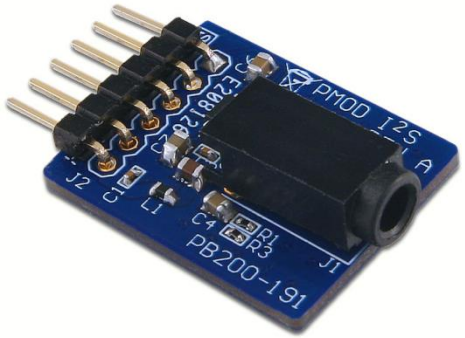
Några Pmods

Piezo



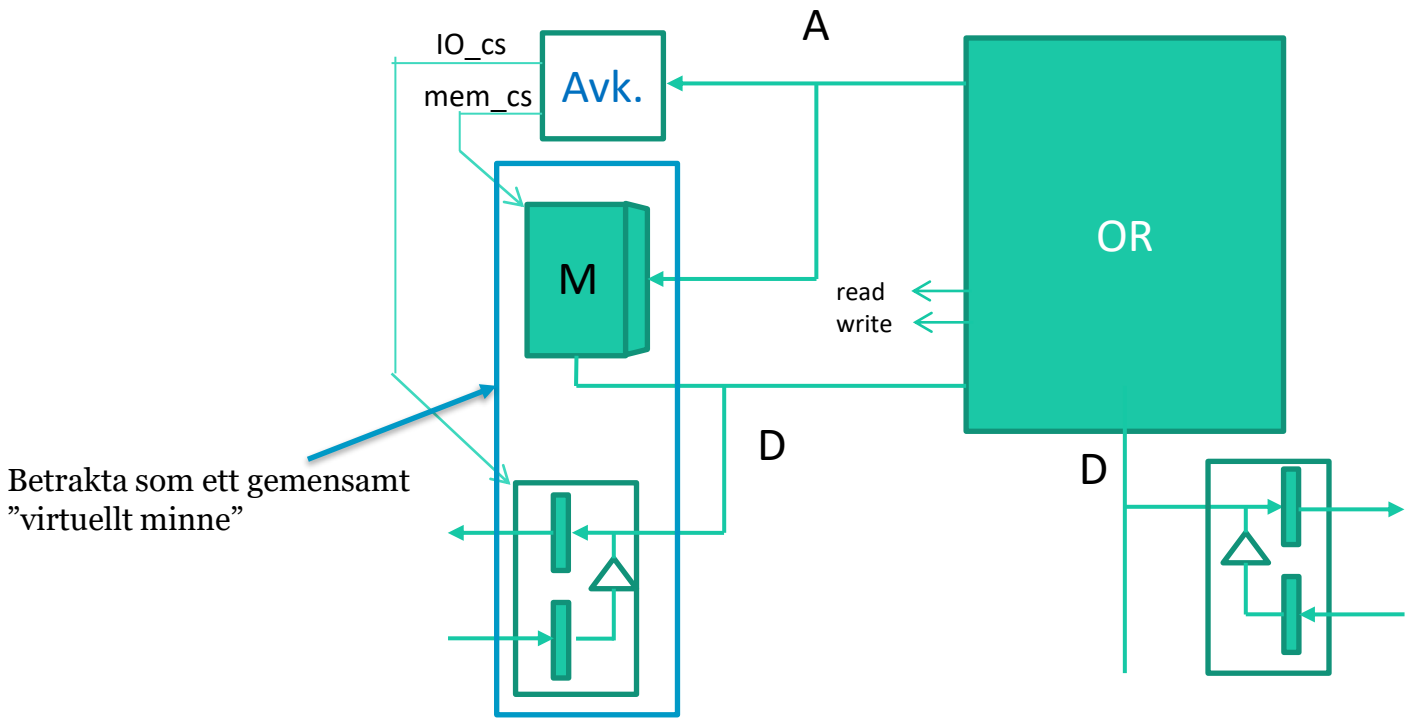
- Monohögtalare (Piezo)
- Bit banging

I²S



- Stereo”högtalare”
- I²S (seriellt)

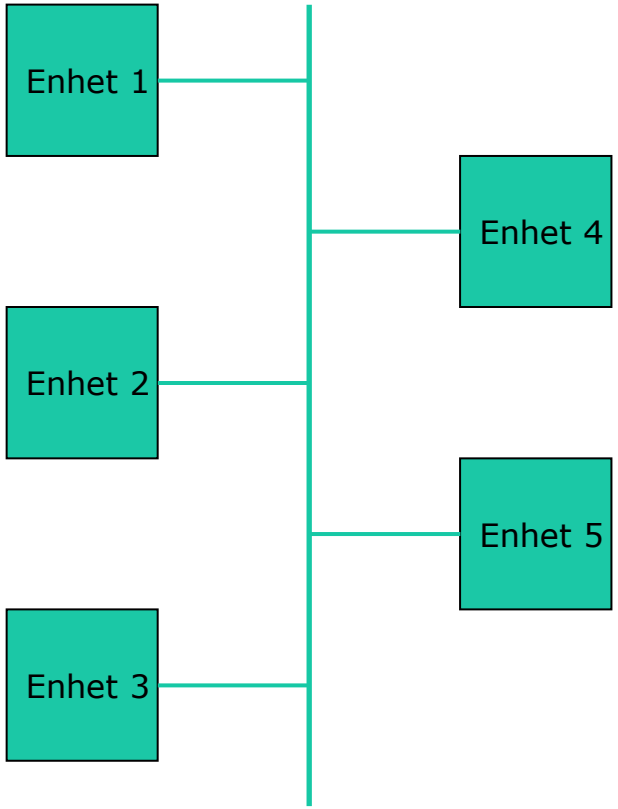
Två sätt att koppla in I/O-kretsar



Memory-mapped I/O
- På den yttre databussen
- **I/O-registren som minnesceller**
- Vanliga instruktioner (LDA,STA)
- Adressavkodning måste fixas mm

I/O-mapped I/O
- Direkt på interna databussen
- **I/O-registren som interna register**
- Speciella instruktioner,
- Som kan vänta på att någonting hänt (fördelen med mikroprog. dator)

Seriella och parallella bussar



Buss = gemensamma ledningar
för kommunikation mellan enheter.
Endast en överföring åt gången.

Sändare/Mottagare = som det låter

Master/Slave = Master kan starta
en överföring

Arbitrering = skiljedom,
behövs för flera masters

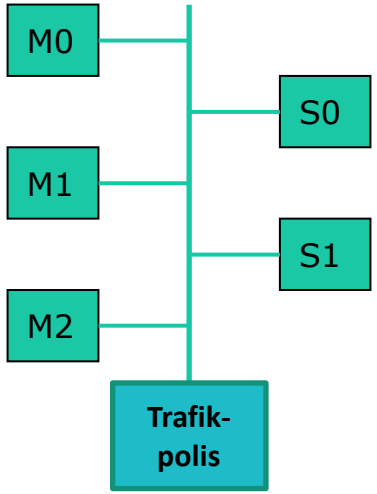
On chip -> multiplexer
Off chip -> tristate

Dum <-> Smart

Seriell <-> parallell

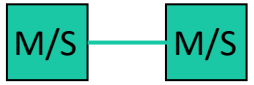
Exempel på enheter:
CPU
Minne
I/O – parallellport, UART, …
I/O med DMA

Några varianter

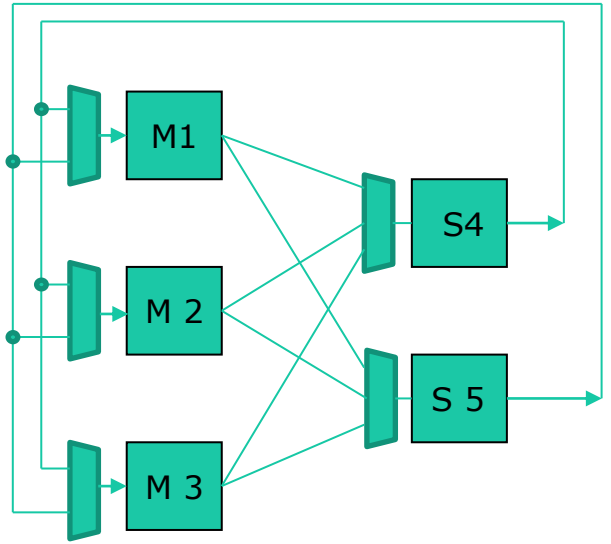


Multimaster-buss,
dubbelriktad databuss, tristate,
endast ett ”samtal” åt gången
(Jfr OR-datorn, mprog styr trafiken på bussen)

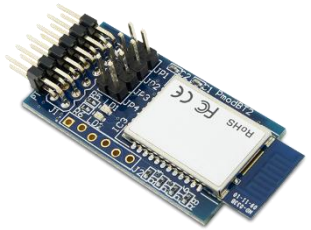
Punkt till punkt



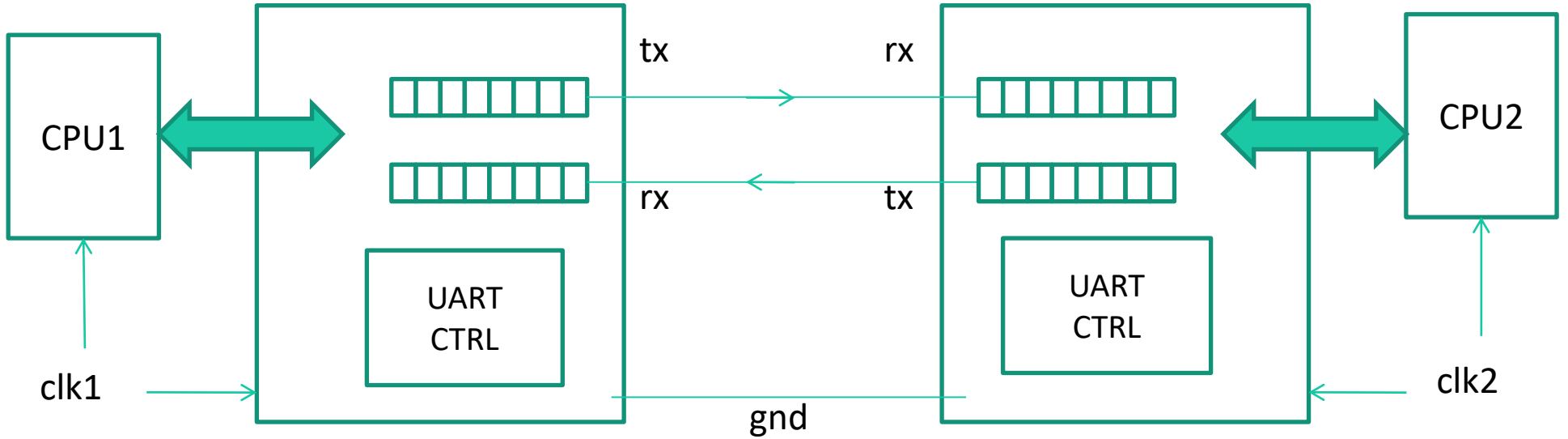
Korskoppling, crossbar
enkelriktade databussar



RS232 UART



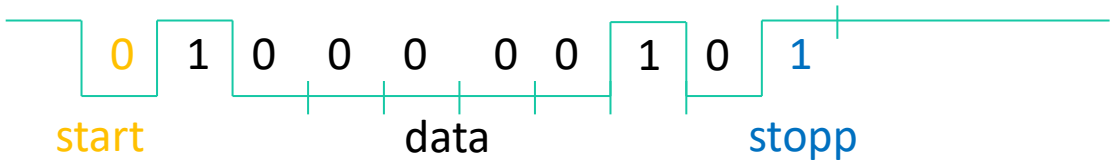
Kommunikation mellan datorer



- Seriell
- Punkt till punkt
- Problem 1: UART1 och UART2 har inte samma klocka
  => synkronisering nödvändig
- Problem 2: hur vet CPU2 att ett tecken kommit in?
  hur vet CPU1 att ett tecken har sänts?
  => handskakning nödvändig

UART

Vår klocka = 100 MHz



Synkronisering: UART <-> UART

bithastighet: 115200 bit/s
1 bit = 868 CK, 1 tecken = 8680 CK, då CLK=100MHz
båda sidor har en räknare, som räknare 16 ggr / bit
S : skifta ut var 16:e CK
M: vänta på startbit, starta räknare,
skifta in mitt i bitarna

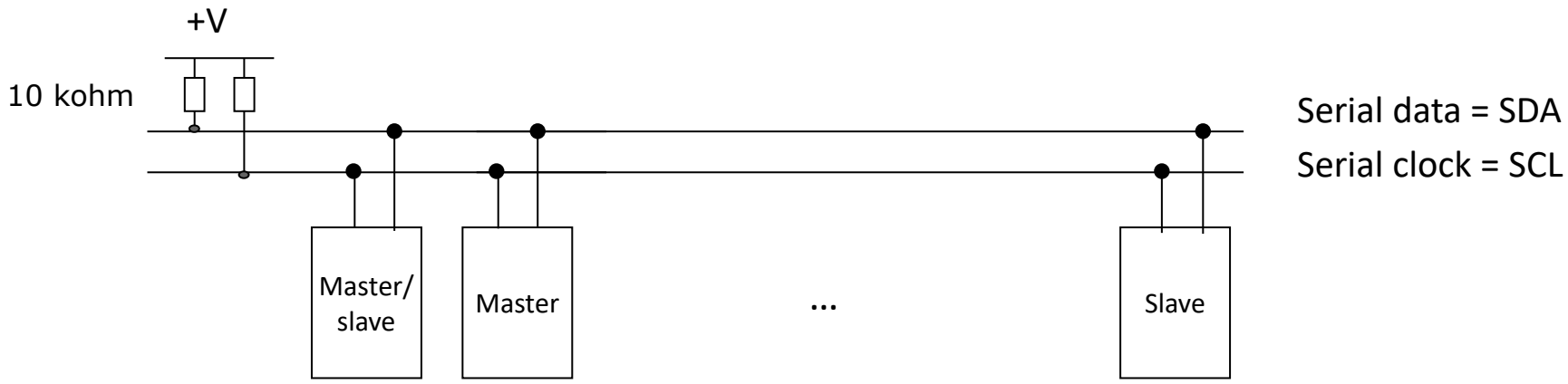
Handskakning: UART <-> CPU

flaggor (=bitar i kontrollregister)

rxfull = tecken har kommit in!	1: när tecknet kommit in
	0: när CPU:n läst tecknet
txempty = tecken har sänts!	1: när tecknet sänts
	0: när CPU:n skrivit tecken

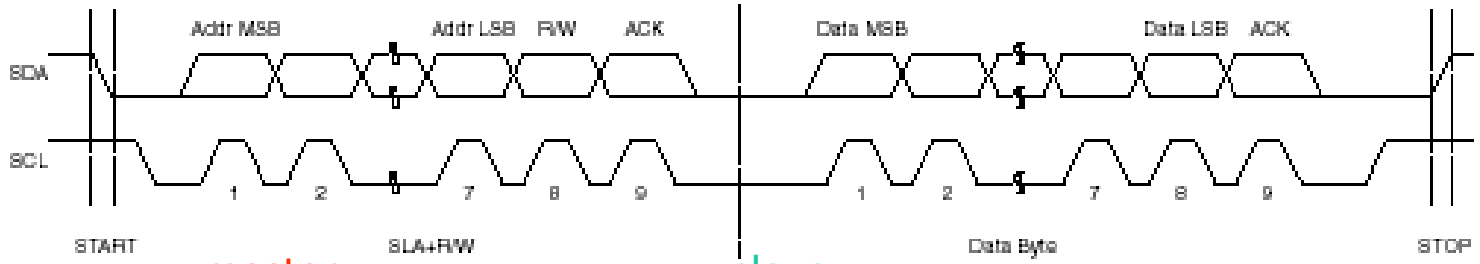
I²C = Inter-Integrated-Circuit

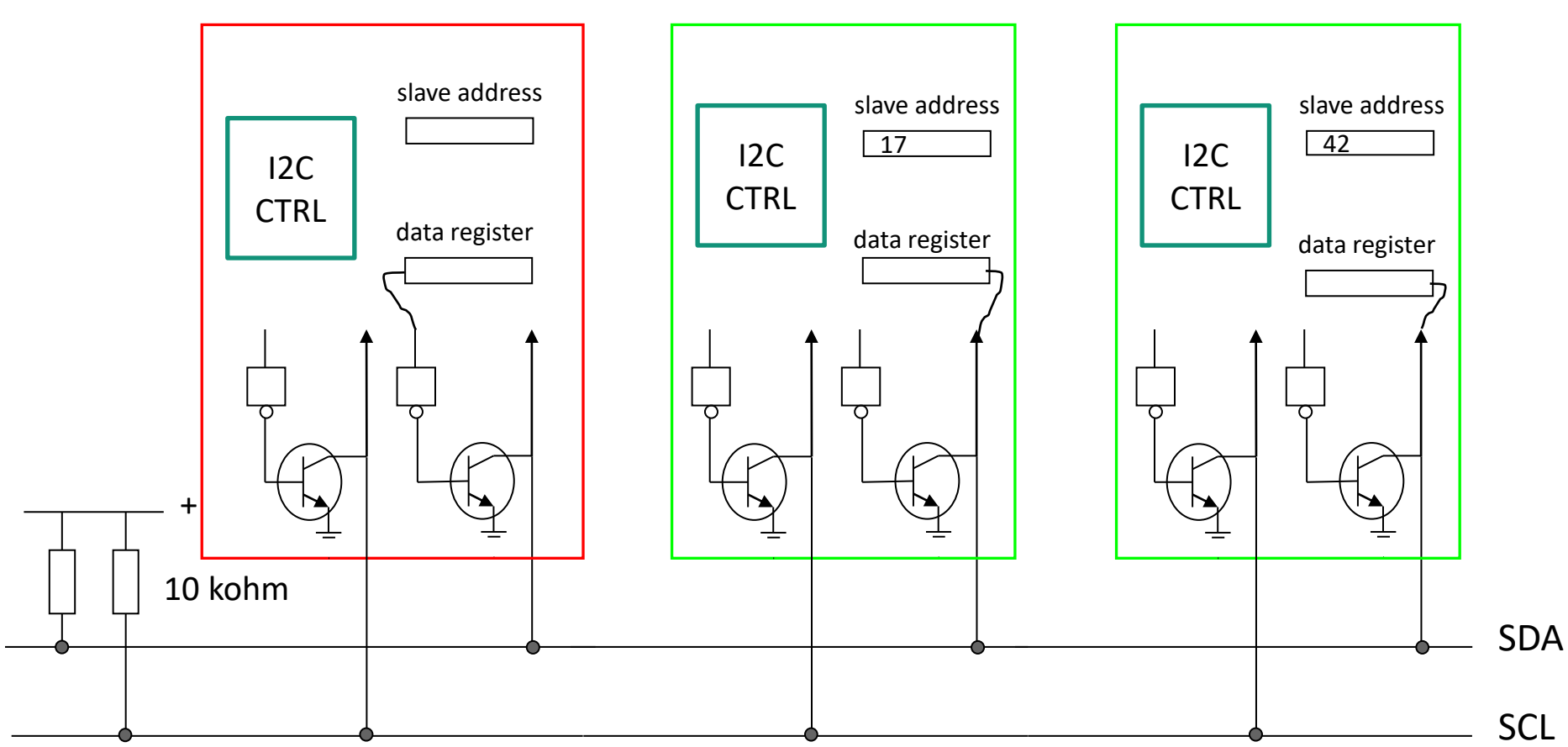
- Seriell buss
- Philips
- Konsumentelektronik
- Kommunikation inom ett kretskort
- 2 trådar, ganska låg hastighet, billigt
- Finns i PC: läs av CPU temp, fläktvarvtal, minnestyp, …
- Upp till 127 enheter



I²C = Inter-Integrated-Circuit

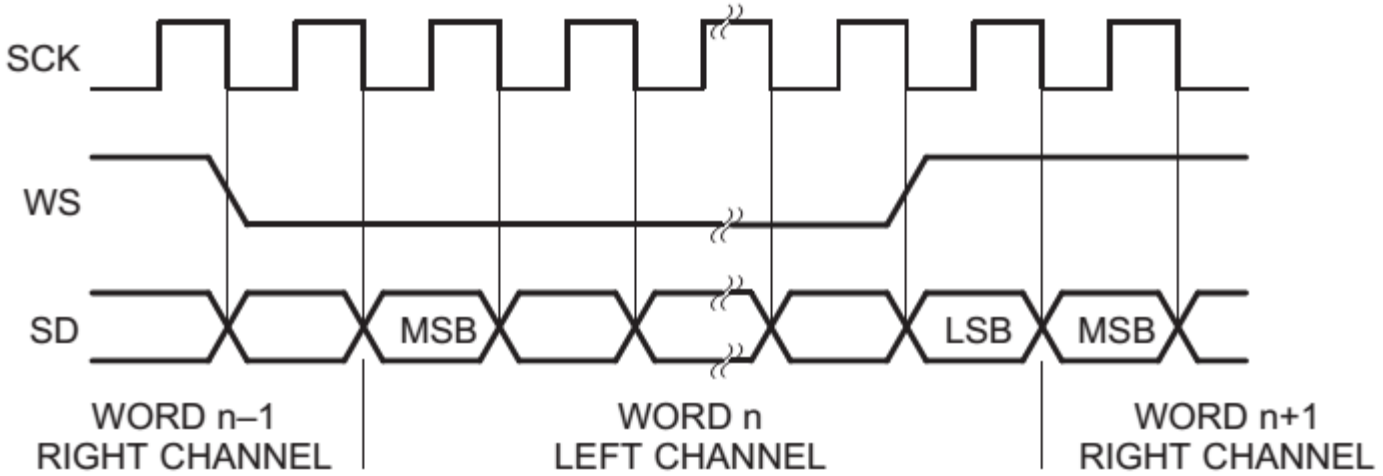
Figure 81. Typical Data Transmission





I^2S = Inter-IC Sound

- Seriell buss
- Philips
- Stereo , multiplexad höger/vänster-kanal
- Tvåkomplementskodat data med oberoende ordlängd
- T ex CD-ljud 44.1 kHz x 16 x 2 => 1.4112 MHz SCK



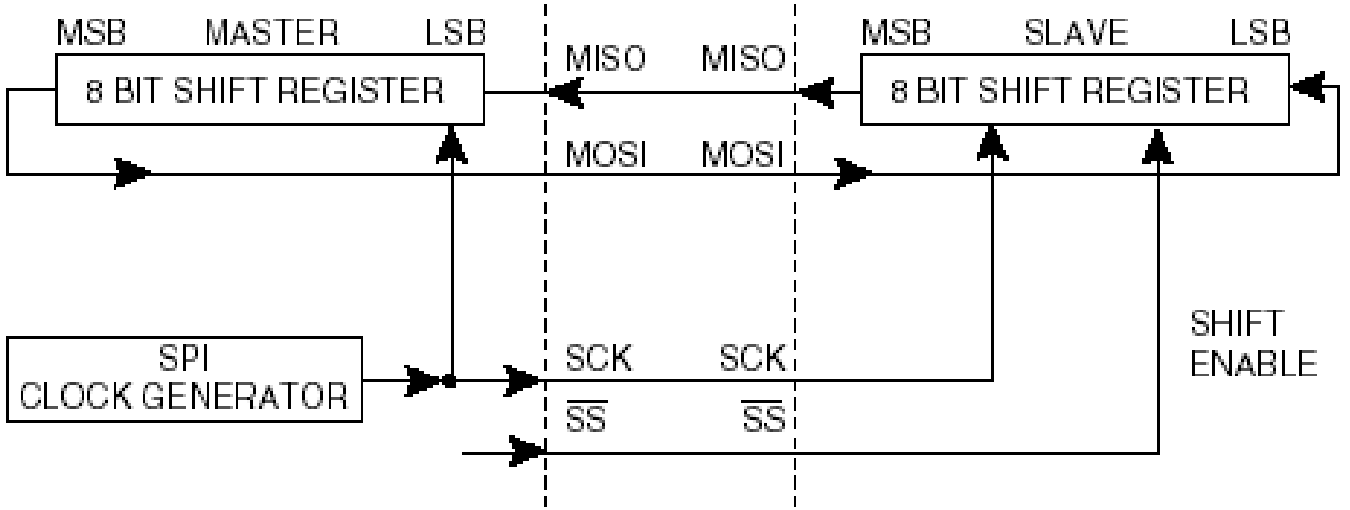
Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega16 and peripheral devices or between several AVR devices. The ATmega16 SPI includes the following features:

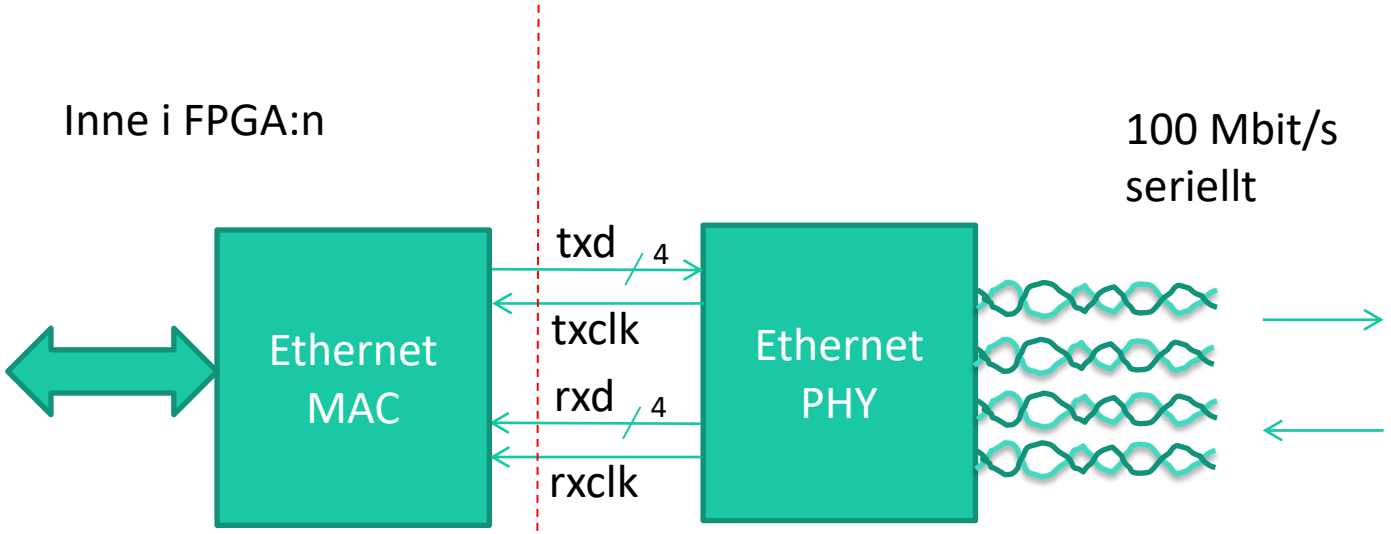
- **Full-duplex, Three-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **LSB First or MSB First Data Transfer**
- **Seven Programmable Bit Rates**
- **End of Transmission Interrupt Flag**
- **Write Collision Flag Protection**
- **Wake-up from Idle Mode**
- **Double Speed (CK/2) Master SPI Mode**

f_{max} = 8/4 MHz

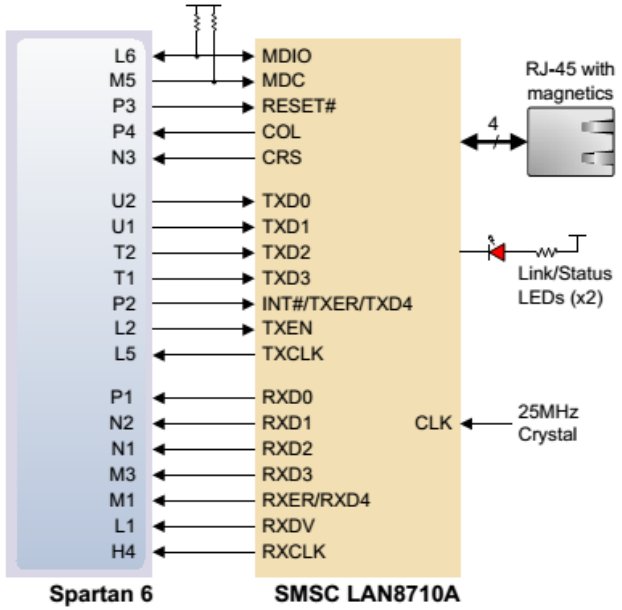
Figure 66. SPI Master-slave Interconnection



Ethernet PHY?

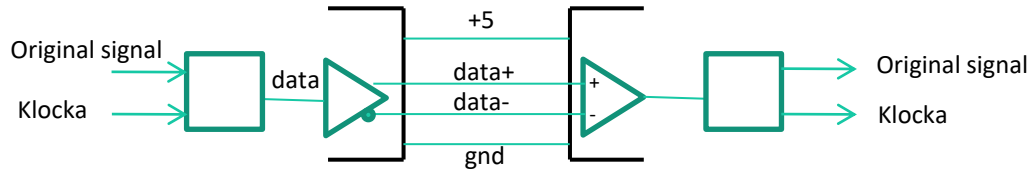


Media Access Control
”bygger paket av nibbles”
Nibble = halv byte, dvs 4 bitar



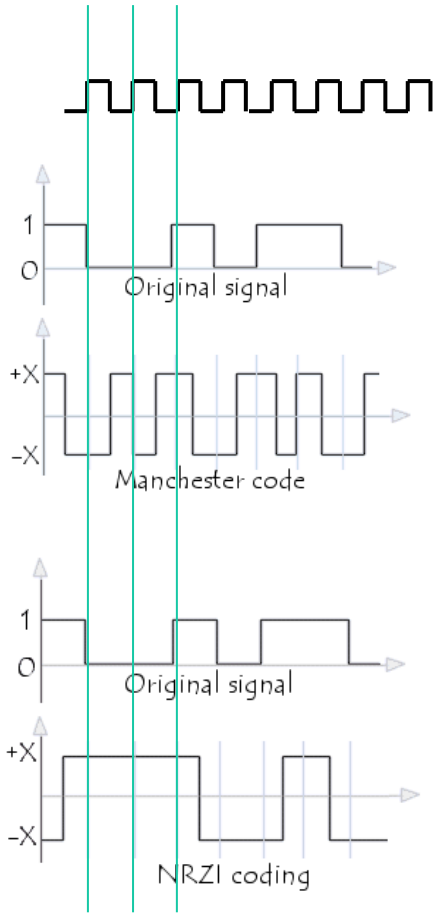
USB = Universal Serial Bus

1)



USB har bara dataledning

2)



0: upp
1: ner

0: behåll
1: ändra
bitstuffing

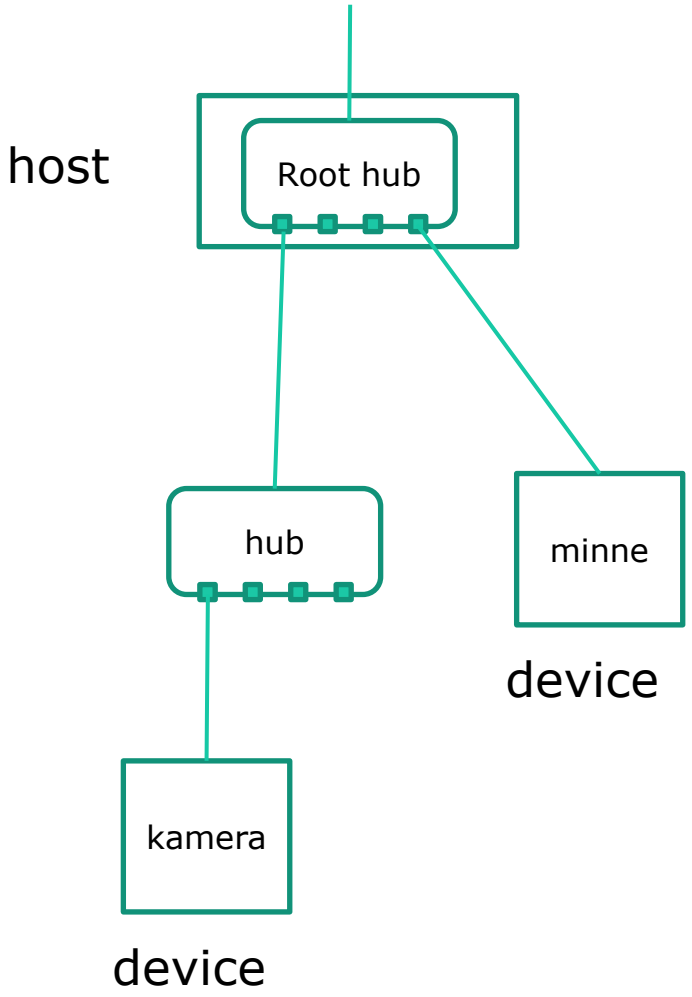
Low Speed – upp till 1,5 Mbit/s
Full Speed – upp till 12 Mbit/s
Hi-Speed – upp till 480 Mbit/s
Superspeed - upp till 5Gbit/s

3)

Kommunikationen är paketbaserad
Exempel:

Det finns många sorters paket!

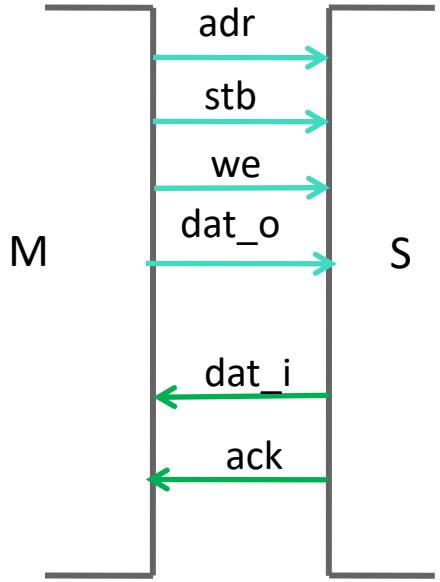
USB = Universal Serial Bus

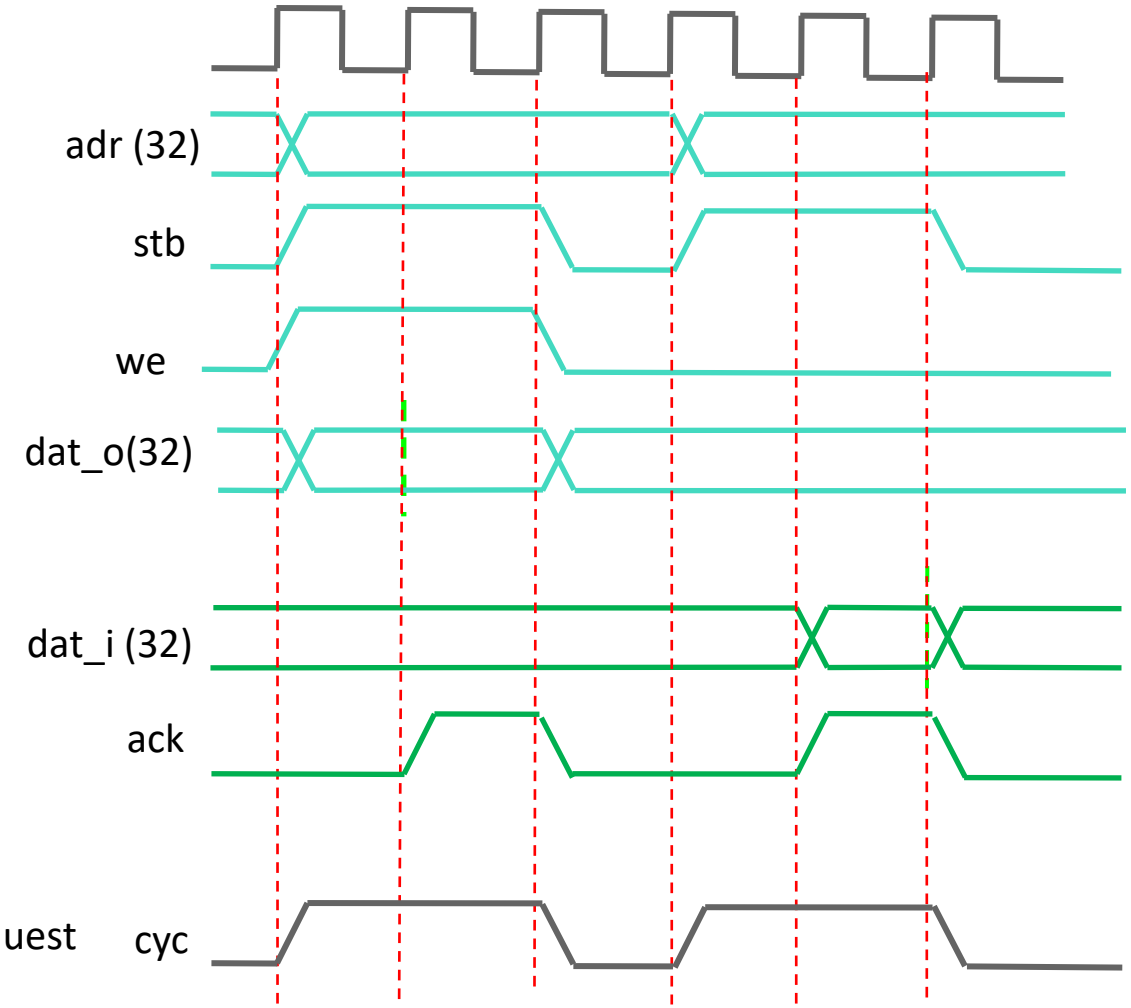


- 1 host
- upp till 127 devices
- Host kontrollerar all trafik
- Inga avbrott
- Each device sees all traffic from host
- A device does not see traffic from other devices
- A driver does not know topology of the network

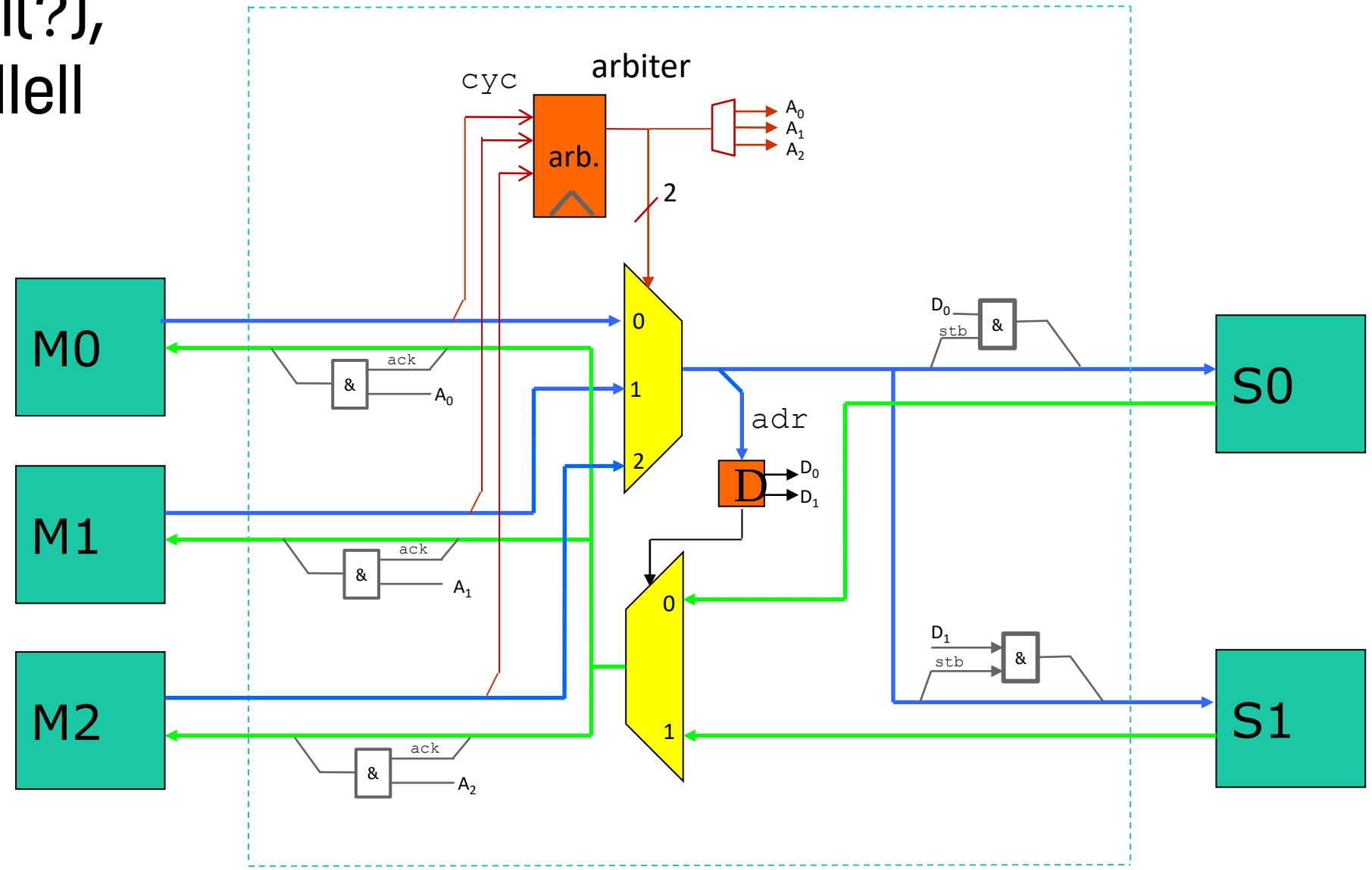
Enkel(?), parallell buss

Wishbone: Open-Source-buss för FPGA, ASIC
on-chip bus, dubbla databussar





Enkel(?), parallell buss



adr[31:0], dat_o[31:0], stb, cyc, we, …

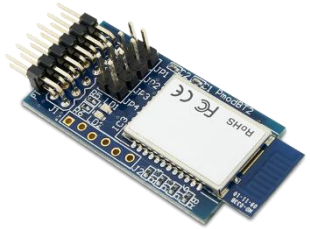
dat_i[31:0], ack, …

Sammanfattning

In-enheter

PC-tangentbord
PS/2

HEX-tangentbord
Scannas

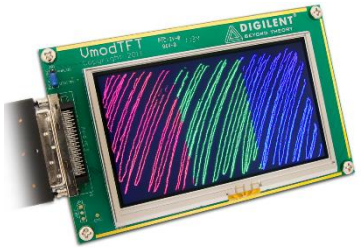


Blåtandsmodul
UART

Joystick
SPI

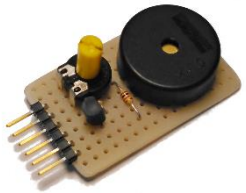
Ut-enheter

VGA-monitor
Synksignaler



LCD med touch
Synksignaler

OLED-display
SPI



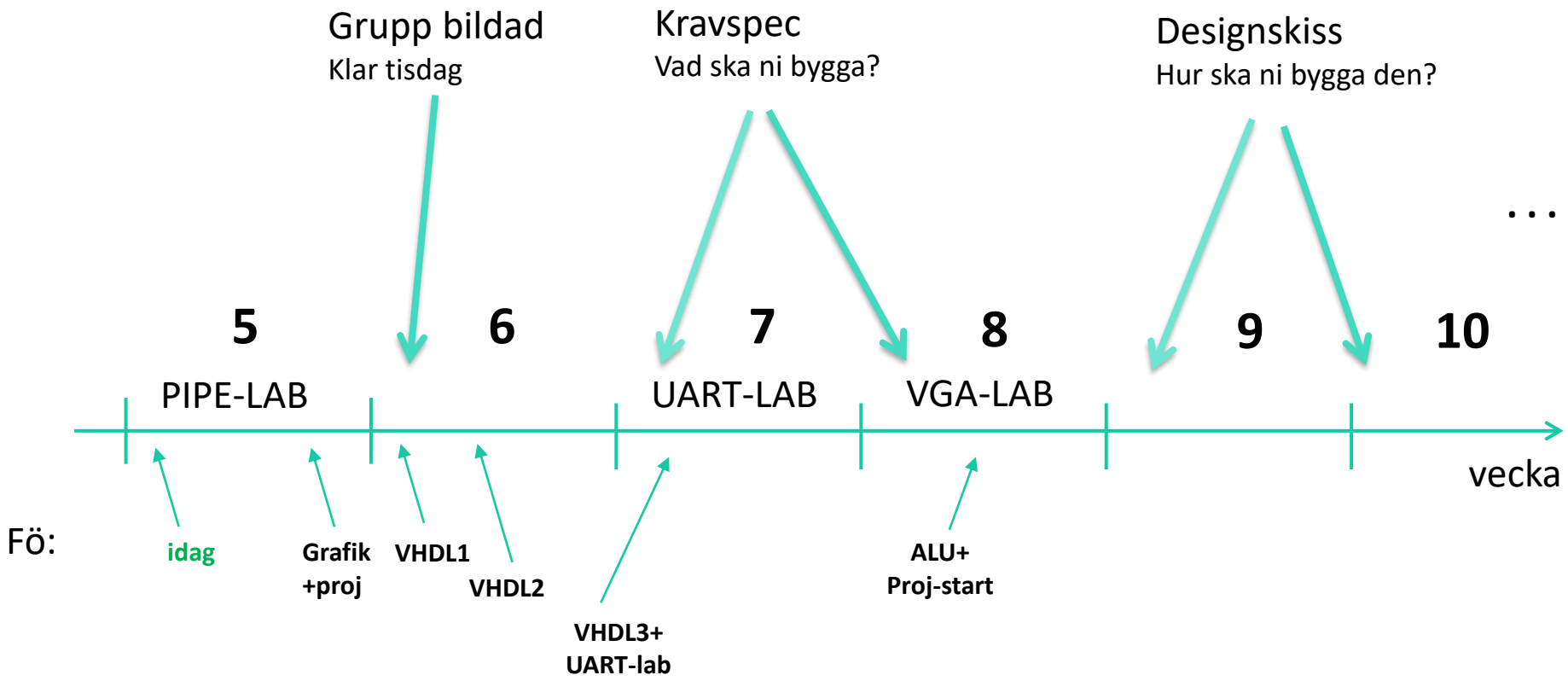
Monohögtalare
Bit banging

Stereo”högtalare”
I²S

Planering

Den närmaste framtiden

Den närmaste framtiden



Anders Nilsson

www.liu.se

LINKÖPINGS UNIVERSITET