6. Minnen

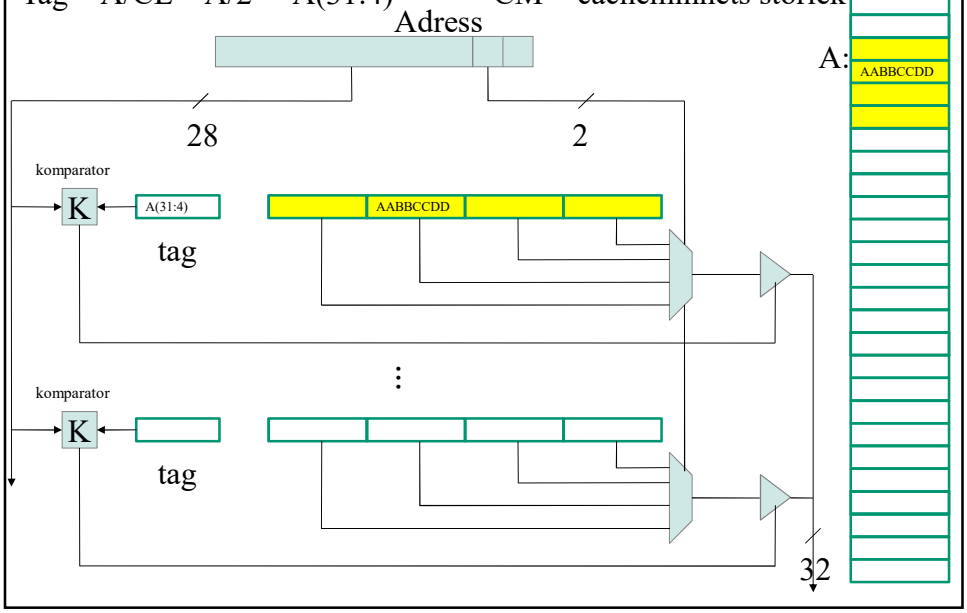
- Repetition cache-minnen, BPT
- Minnen allmänt
  ROM, RAM, DRAM, SDRAM, DDR
- MMU - Memory management
- Minnen NEXYS3
  Block RAM, distributed RAM, DDR3

1

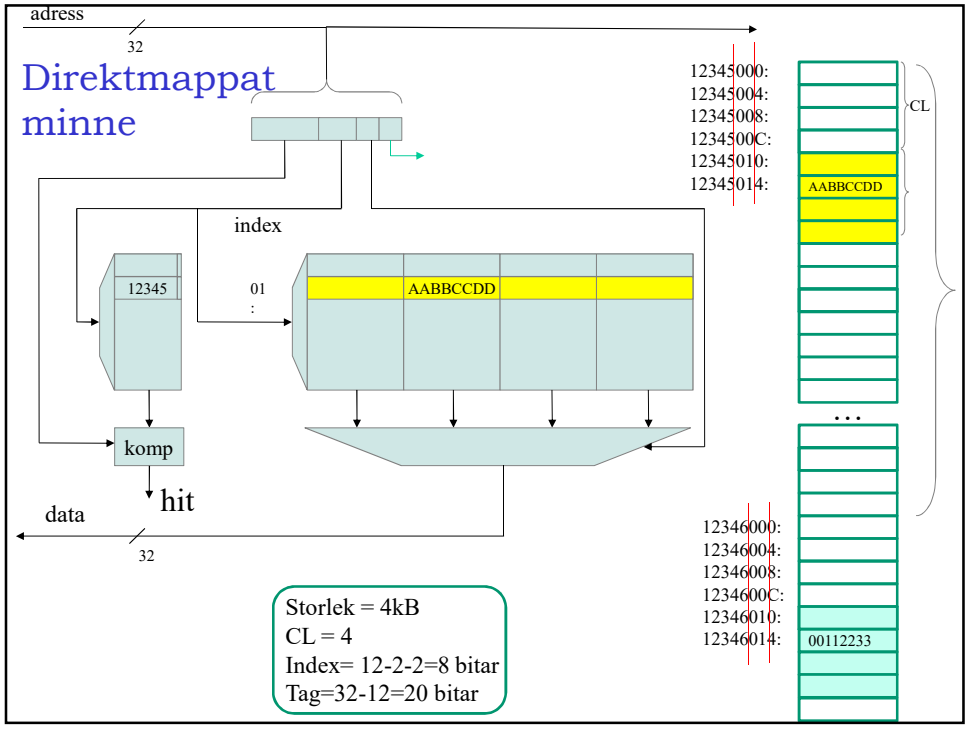
Associativt minne

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

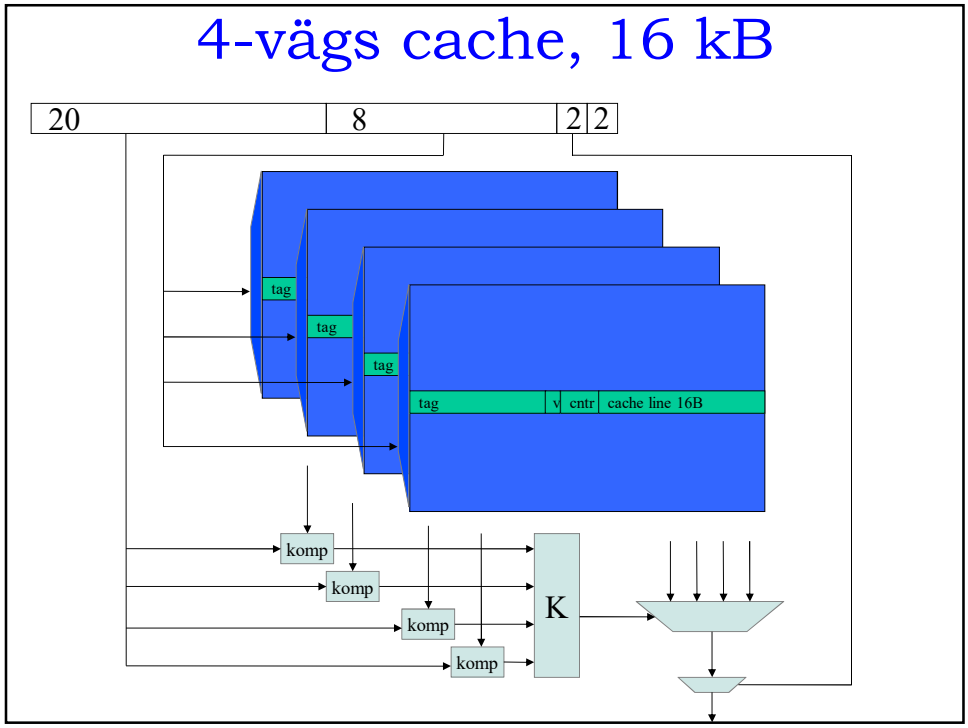
Tag = A/CL = A/2^4 = A(31:4)



2


adress
32
Direktmappat minne
index
12345
01
:
AABBCCDD
komp
hit
data
32
Storlek = 4kB
CL = 4
Index= 12-2-2=8 bitar
Tag=32-12=20 bitar
12345000:
12345004:
12345008:
1234500C:
12345010:
12345014:
AABBCCDD
CL
12346000:
12346004:
12346008:
1234600C:
12346010:
12346014:
00112233

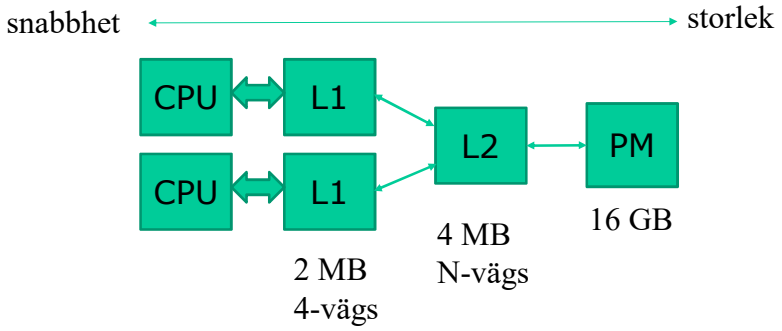
3


4-vägs cache, 16 kB
20
8
2
2
tag
tag
tag
tag
v
cntr
cache line 16B
komp
komp
komp
komp
K

4

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

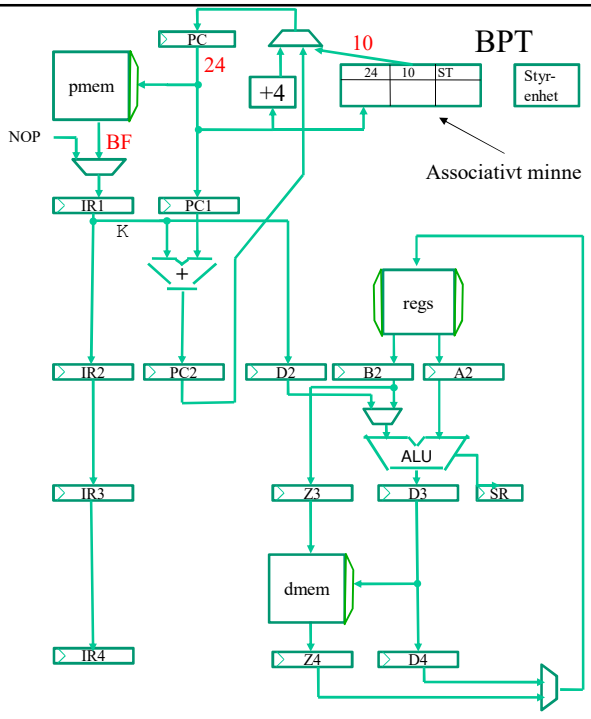


5

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
```



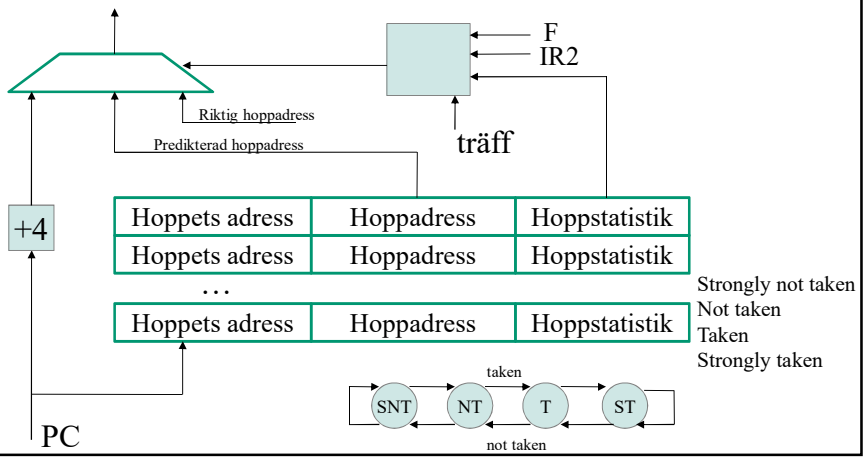
6

BPT = branch prediction table

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



7

BPT = branch prediction table

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

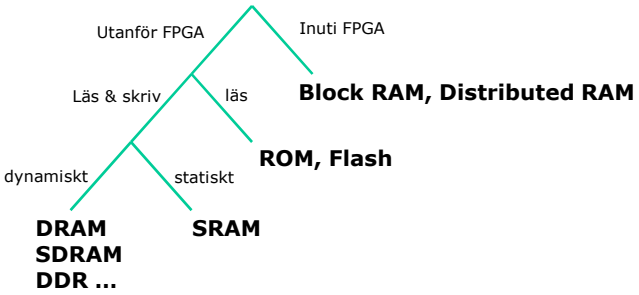
Rätt predikterat : Vi sparar 2 klockcykler, XXX går aldrig in i pipelinen
Den där NOP-en försvinner

Fel predikterat: Vi förlorar 2 klockcykler

8

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

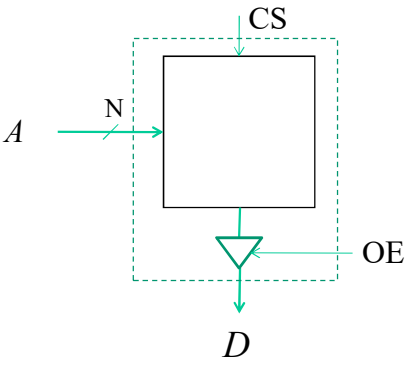


10

Minnen ROM

Läsminnen

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



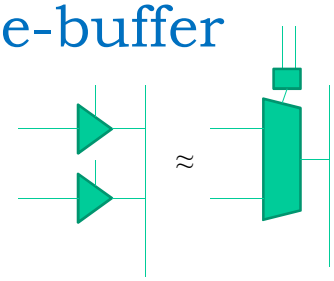
N st adressingångar
M st dataingångar
1 st chip select-ingång (aktivt låg)
1 st output enable-ingång (aktivt låg)

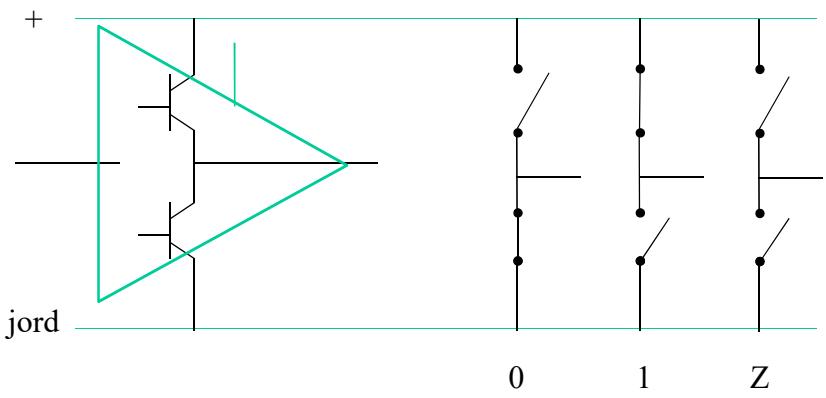
Organisation: 2^N ord à M bitar

11

Tristate-buffer

1) Förbättrar signalen
2) Möjlighet till tristate



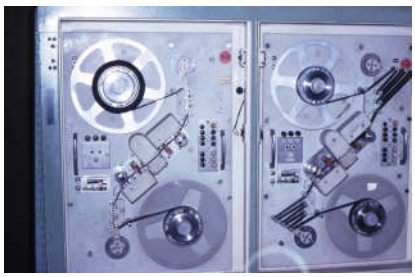


12

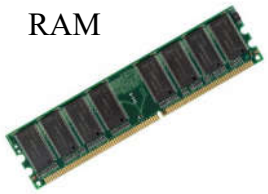
RAM – Random Access Memory

Vaddå random?

Tape memory



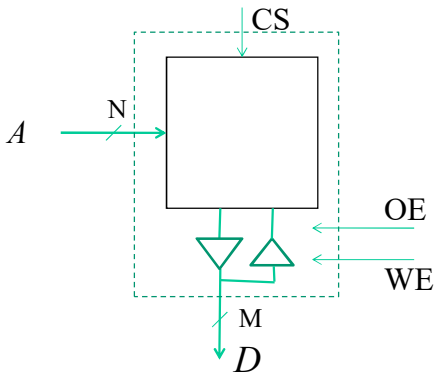
RAM



13

Static RAM = SRAM

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)

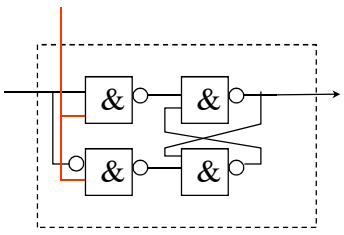
Organisation: 2^N ord à M bitar

14

Static RAM = SRAM

Läs/skriv , minnet försvinner utan spänning

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



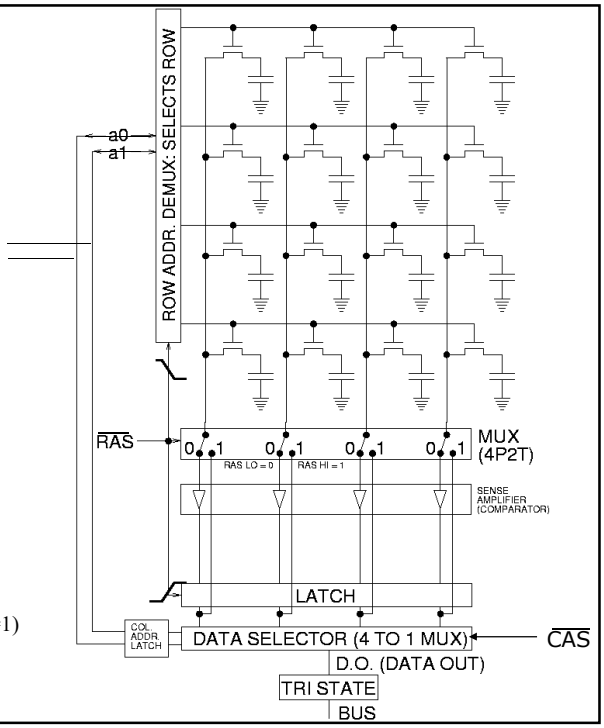
15

DRAM (princip)

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



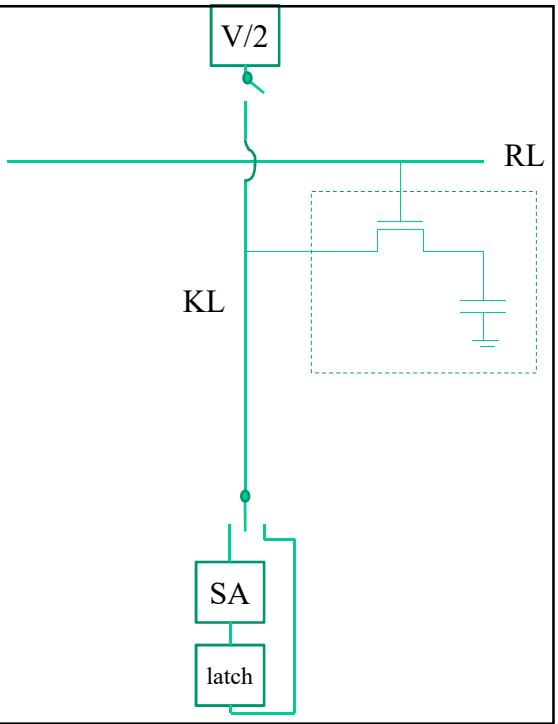
18

DRAM (princip)

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

Läscykel

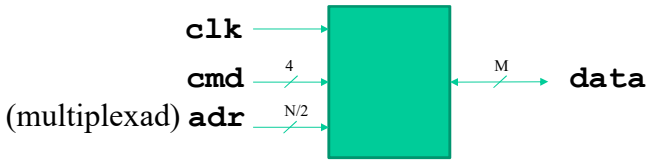
1. Förladda KL till V/2
2. Låt KL flyta fritt
3. Driv radledningen till V
4. Koppla in SA till KL
5. Håll KL i latchen
6. Återställ laddning (omkopplaren=1)
7. Släpp ut värde på bussen



19

Synchronous Dynamic RAM = SDRAM

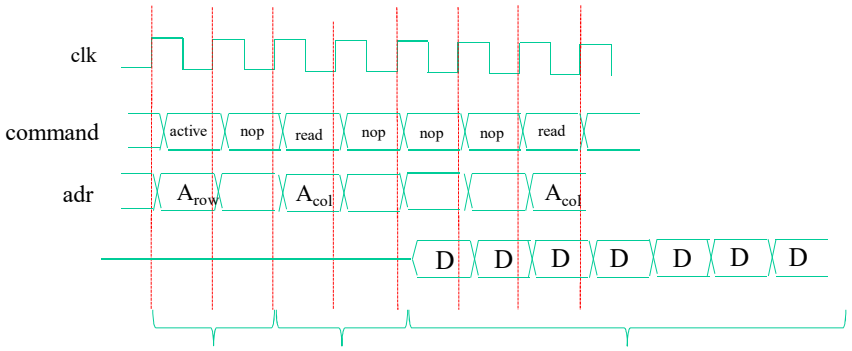
- Dominerar helt
- 1-transistorcellen => 1 läs/skriv-transistor + 1 kapacitans
- Synkron komponent



- 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.

23

Read burst

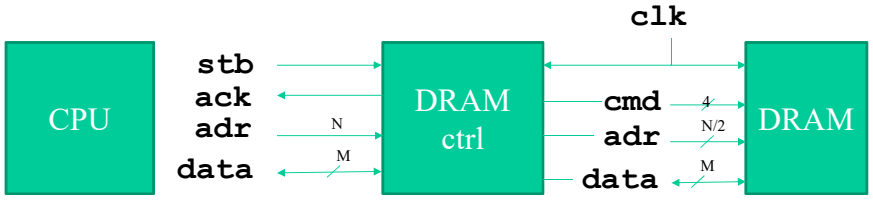


Läs en hel rad | Muxa ut bitar | Så länge vi är på samma rad är allt bra!

24

Synchronous Dynamic RAM = SDRAM

- 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)

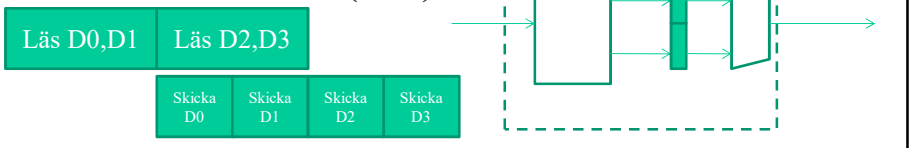


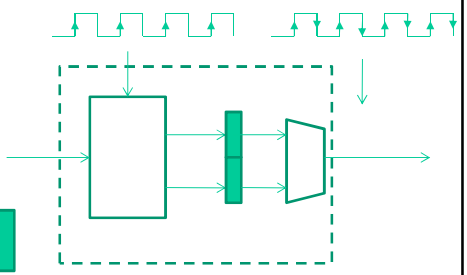
Refreshcykler har högst prioritet,
CPU måste ibland vänta genom att **ack** fördröjs

25

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)





26

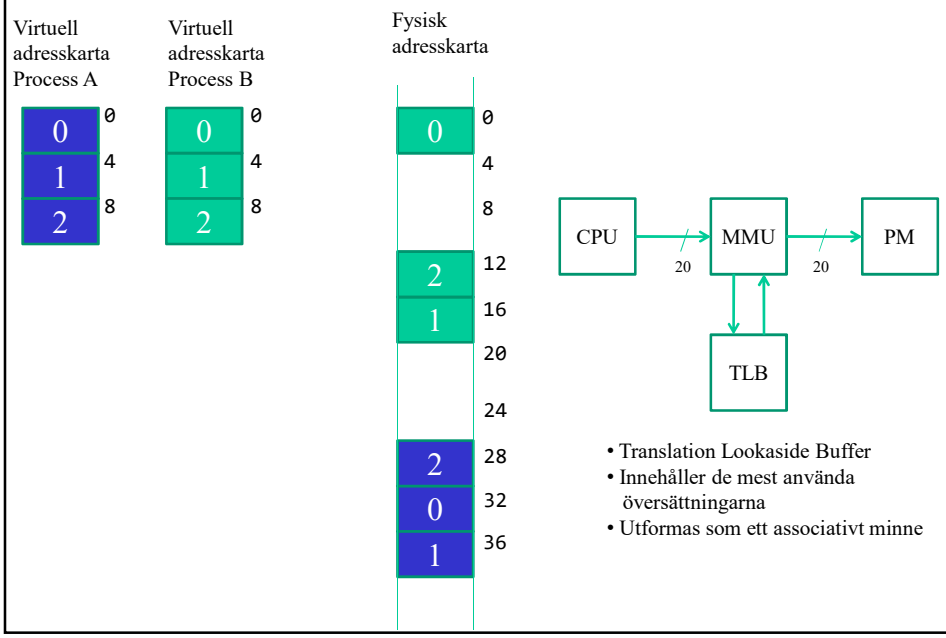
Memory management

- 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.

37

Memory management, 4kB pages



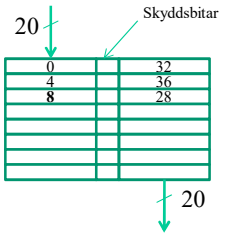
38

Memory management, 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.

TLB



Associativt minne
Ex: 64 rader

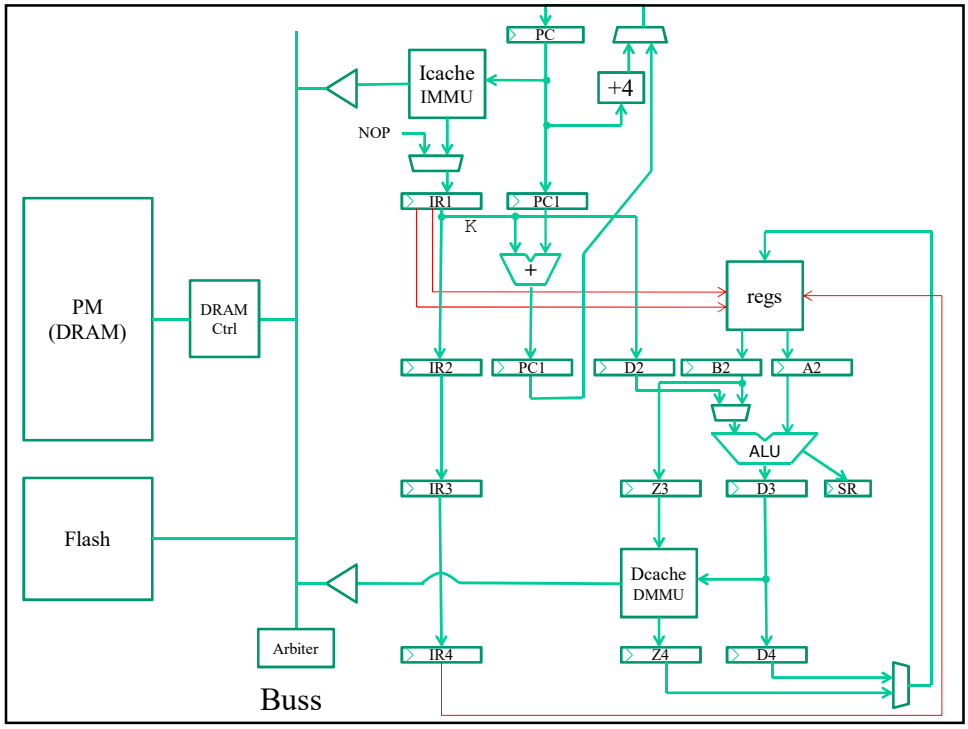
39

Memory management

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 ...

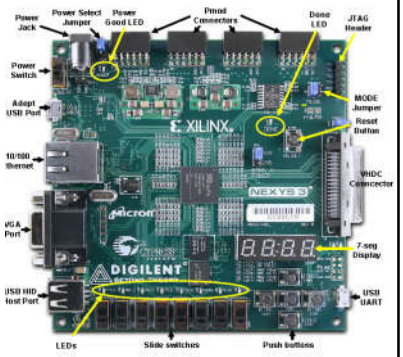
41

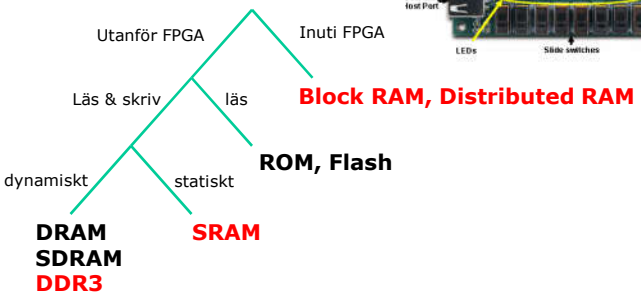


42

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

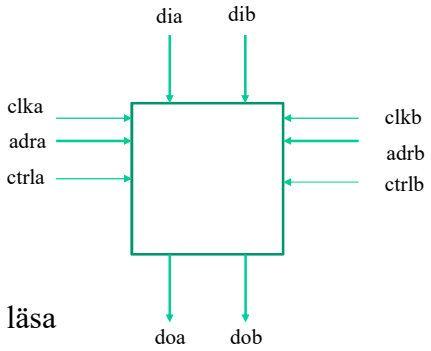




43

Block RAM

- I arrayen 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.

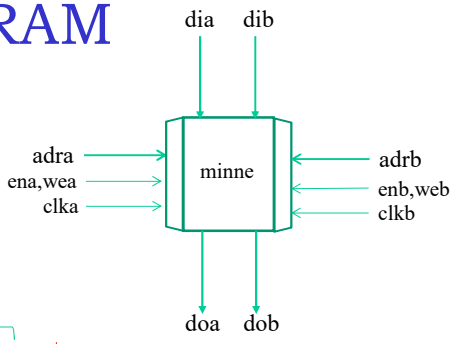


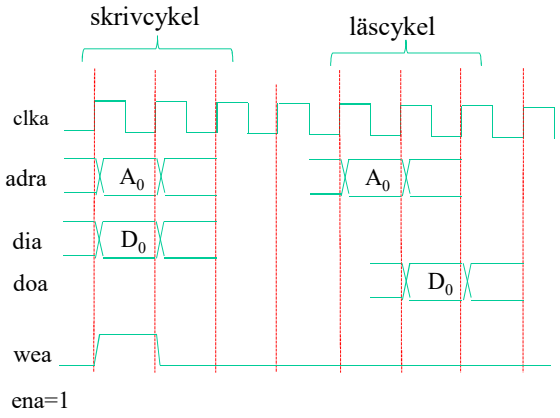
BRAM är tvåportsminne.
A och B är helt oberoende.
Vi kan skriva på A-sidan och läsa på B-sidan samtidigt.

45

Block RAM

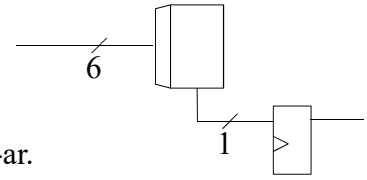
- Både läscykel och skrivcykel är synkrona!





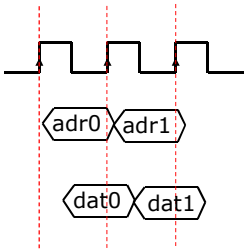
46

Distributed RAM

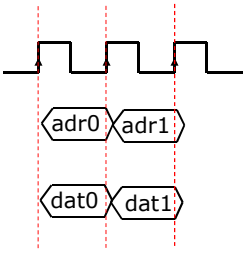


Logikblocken består av vippor och LUT-ar. Syntesverktyget kan bygga små minnen av LUT-arna.

Kombinatorisk läsning



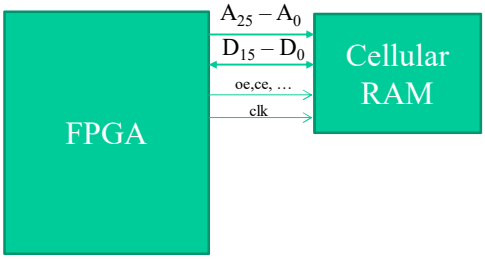
Synkron skrivning



47

FPGA-kortet

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



48