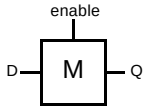
F10: Minneselement

- **Innehåll:**
  - **Latchar**
  - **Flip-Flops**
  - **Register**
  - **Läs- och skrivminne (Random-Access Memory RAM)**
  - **Läsminne (Read Only Memory ROM)**

Ett minneselements egenskaper

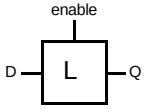
- **Generellt sett så kan följande operationer utföras på ett minneselement**
  - **Skriv:**
    Värdet på signalen som ligger på ingången D skrivs in i minnet.
  - **Lagra:**
    Värdet som skrevs in i minneselementet hålls kvar även om värdet på insignalen D förändras.
  - **Läs:**
    Läs-operationen tillhandahåller det lagrade värdet på utgången Q.



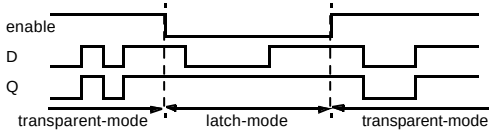
enable = 1: skriv D i minnet
enable = 0: håll värdet i minnet

Latch

- **En latch är ett minneselement med följande egenskaper:**
  - **Den kan vara transparent:**
    d.v.s utgången Q följer ingången D
  - **Den kan låsa ett värde (latch):**
    d.v.s minnes ett binärt värde (Q=0 eller Q=1) genom att använda en *bistabil krets*.
- **Generell modell:**

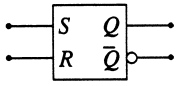


- **Tidsdiagram:**

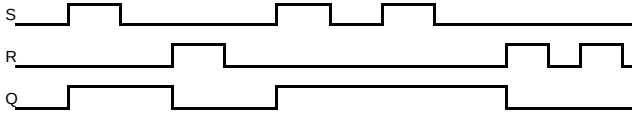


SR-latch

- **Funktion:**
  - **SR-latchen har två ingångar som aktiverar latchens två grundläggande operationer:**
    Set (S):
    Latchens innehåll tvingas till värdet 1 (Q=1)
    Reset (R):
    Latchens innehåll tvingas till värdet 0 (Q=0)
- **Symbol:**

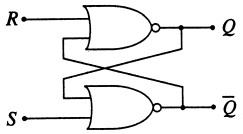


- **Tidsdiagram:**



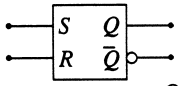
SR-latch baserad på NOR-grindar

- **Kretsschema och funktion:**



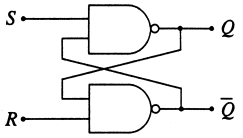
S	R	Q	$\bar{Q}$	Operation
0	0	Q	$\bar{Q}$	Hold
⎍	0	1	0	Set (Q → 1)
0	⎍	0	1	Reset (Q → 0)
1	1	0	0	Not used

- **Symbol:**



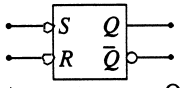
SR-latch baserad på NAND-grindar

- **Kretsschema och funktion:**



S	R	Q	$\bar{Q}$	Operation
0	0	0	0	Not used
⎍	1	1	0	Set ($Q \rightarrow 1$)
1	⎍	0	1	Reset ($Q \rightarrow 0$)
1	1	Q	$\bar{Q}$	Hold

- **Symbol:**

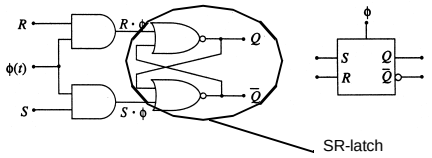


Inputs Outputs

Klockad SR-latch

- **Funktion:**
  - **Synkronisering av set- och resetsignalerna med hjälp av en klocksignal.**

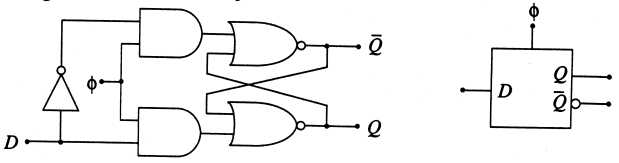
- **Logiskt schema och symbol**



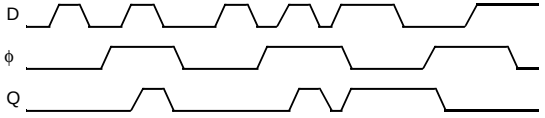
Klockad D-latch

- **Funktion:**
  - **Skrivning av data till latchen synkroniseras av en klocksignal**

- **Logiskt schema och symbol:**

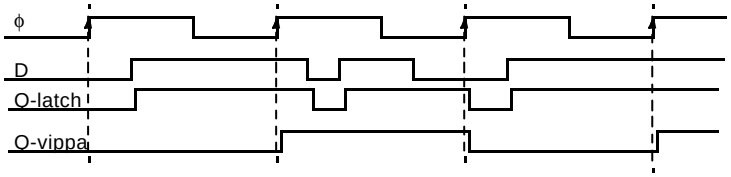


- **Tidsdiagram:**



Flank-triggade vippor

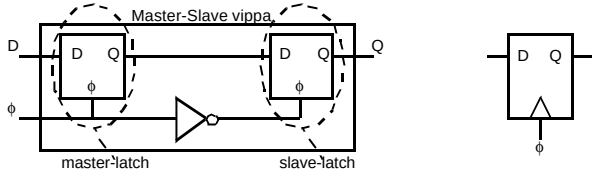
- **Klockad D-latch**
  - **Nivån på klocksignalen avgör när nytt data ska skrivas in till latchen.**
- **Klockad D-vippa (D-flip-flop)**
  - **Nytt data skrivs in i vippan på flanken av klocksignalen**
- **Jämförelse:**



- **Polaritet**
  - **En positivt flanktriggad vippa läser in nytt data då ϕ gör övergången 0 → 1.**
  - **En negativt flanktriggad vippa läser in nytt data då ϕ gör övergången 1 → 0.**

Master-Slave konfiguration för flank-triggad vippa

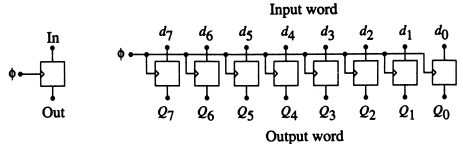
- **Master-slave konfiguration:**



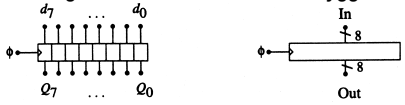
- **Funktion:**
  - **Då ϕ = 0:**
    master-latchen är stängd och slave-latchen transparent.
    Vippans värde ligger i master-latchen.
  - **Då ϕ = 1:**
    master-latchen är transparent och slave-latchen stängd.
    Vippans värde ligger i slave-latchen. Master-latchen är öppen och nytt värde kan skrivas in
  - **Då ϕ 1 → 0:** Master-latchens värde förs över via slave-latchen till utgången.

Register

- **Funktion:**
  - **Lagra n-bitars binära tal**
- **Konstruktion:**
  - **Parallellkoppling av n stycken minneselement**
    Samtliga element kan läsas och skrivas samtidigt.
  - **Sammankoppling av individuella vippor till ett 8-bitars register**

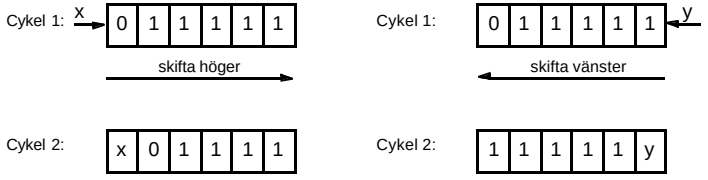


  - **Ett register kan ses som ett enda byggblock**



Skiftregister

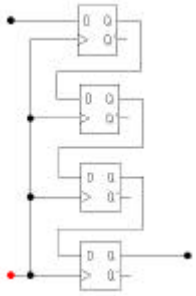
- **Funktion:**
  - **Ett skiftregister flyttar det lagrade binära talet i "sidled" vänster eller höger.**
  - **Exempel:**



- **Exempel på användning:**
  - **Multiplicera med 2 (skifta vänster)**
  - **Dividera med 2 (skifta höger)**
  - **Omvandla data från parallell form till seriell form**

Skiftregister för Seriellt in och Seriellt ut

- **Namn:**
  - **Serial-In Serial-Out (SISO) register**
- **Exempel:**



serin: 1-bits seriell ingång
serout: 1-bits seriell utgång

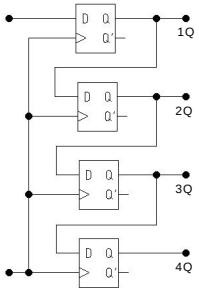
serout = serin fördröjd n klockperioder i ett n-bitars SISO-register

För ett 4-bitars SISO register

serin	1	0	1	1	0	0	1	1	1	0
serout	-	-	-	-	1	0	1	1	0	0
clk	¦	¦	¦	¦	¦	¦	¦	¦	¦	¦

Skiftregister för Seriellt in och Parallellt ut

- **Namn:**
  - **Serial-In Parallel-Out (SIPO) register**
- **Exempel:**



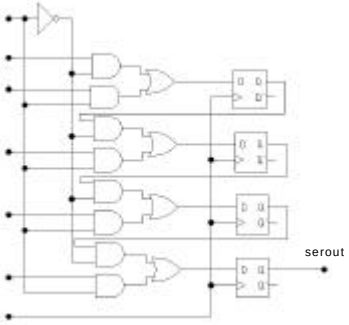
Seriell till parallel konvertering

För ett 4-bitars SIPO register:

serin	1	0	1	1	0	0	1	1	1	0
1Q	-	1	0	1	1	0	0	1	1	1
2Q	-	-	1	0	1	1	0	0	1	1
3Q	-	-	-	1	0	1	1	0	0	1
4Q	-	-	-	-	1	0	1	1	0	0
clk	¦	¦	¦	¦	¦	¦	¦	¦	¦	¦

Skiftregister för Parallellt in och Seriellt ut

- **Namn:**
  - **Parallel-In Serial-Out (PISO) register**
  - **Exempel:**



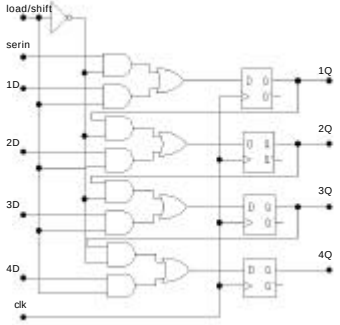
Parallell till seriell konvertering

Load/Shift = 1 : $D_i \rightarrow Q_i$

Load/Shift = 0 : $Q_i \rightarrow Q_{i+1}$

Skiftregister för Parallellt in och Parallellt ut

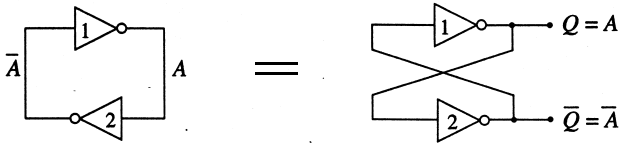
- **Namn:**
  - **Parallel-In Parallel-Out (PIPO) register**
- **Exempel:**



Generellt, kan användas för att göra vilken typ av skiftregister.

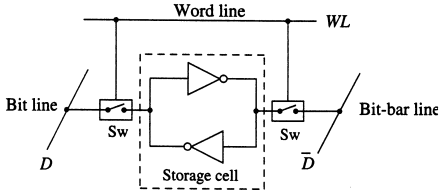
Statisk RAM cell

- **Funktion:**
  - **Håller 1 bits information statiskt, d.v.s data ligger kvar ända tills nytt data skrivs in.**
- **Minneselement:**
  - **Konstrueras med två sammankopplade inverterare:**



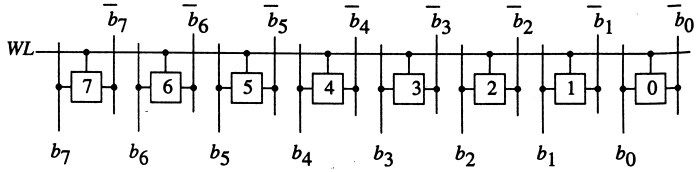
Adressering av minnescell

- **Adresseringsteknik:**
  - **Minneselementet kommer man åt via två omkopplare (switchar)**
  - **Switcharna kontrolleras av en enable-signal som kallas "word line" (WL)**
    - **WL = 0** så håller minnescellen sitt värde
    - **WL = 1** så tillåts antingen läs- eller skrivoperation
  - **Om skriv- eller läsoperation ska göras bestäms av speciella kretsar (som inte visas här)**



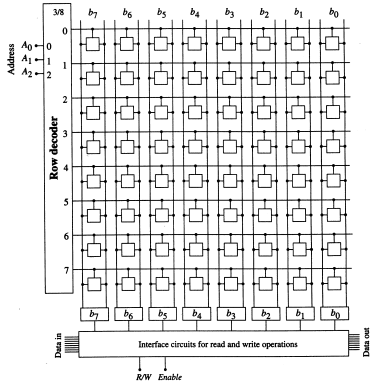
Matriser med SRAM celler

- **Exempel:**
  - **1 x 8 bitars minne, d.v.s 1 binärt tal (ord) som består av 8 bitar kan lagras.**
  - **Funktion:**
    WL = 0: minnescellerna isoleras från in- och utgångarna (b_i)
    WL = 1: samtliga minnesceller i ordet kan kommas åt antingen för skrivning eller läsning.



Matriser med SRAM celler

- **Exempel:**
  - **8 x 8 bitars minne, d.v.s 8 binära tal (ord) som består av 8 bitar kan lagras.**



Ingångar:
$A_2A_1A_0$ adresserar ett ord

R/W styr om skrivning eller läsning ska göras på det adresserade ordet.

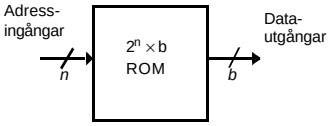
Data in är data som ska skrivas till minnet.

*Enable*aktiverar hela minnesmatrisen.

Utgångar:
Data out är data som läses från minnet.

Läsminne (Read Only Memory - ROM)

- **Funktion:**
  - **En kombinatorisk krets med *n* ingångar och *b* utgångar.**
  - **Innehållet i minnet bestäms av användaren och programmeras en enda gång.**
  - **Minnet är "icke-flyktigt", d.v.s. innehållet finns kvar även utan matningsspänning.**



ROM

- **Olika synsätt:**
  - **Ett ROM lagrar 2^n ord med vardera *b* bitar.**
  - **Ett ROM largrar en logisk funktion som motsvarar en sanningstabell med *n* ingångar och *b* utgångar.**

- **Exempel:**

n = 2 b = 4

A1	A0	D3	D2	D1	D0
0	0	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	1	0	0	0

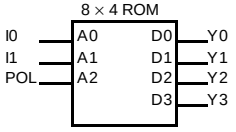
Lagrar 4 stycken 4-bitars ord eller lagrar 4 funktioner med 2 ingångsvariabler.

Exempel: Kombinatorisk logik med ROM

- **Konstruktionsuppgift:**
  Konstruera en krets med 3 ingångar och 4 utgångar med den logiska funktionen specificerad i sanningstabellen nedan. Det är en 2-4 avkodare med "polaritetskontroll".

A2	A1	A0	D3	D2	D1	D0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

aktivt höga utgångar

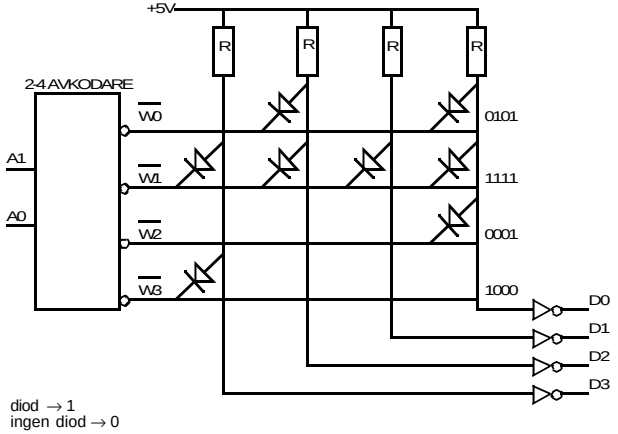


A2 = Polaritet, 0 = aktiv låg, 1 = aktiv hög

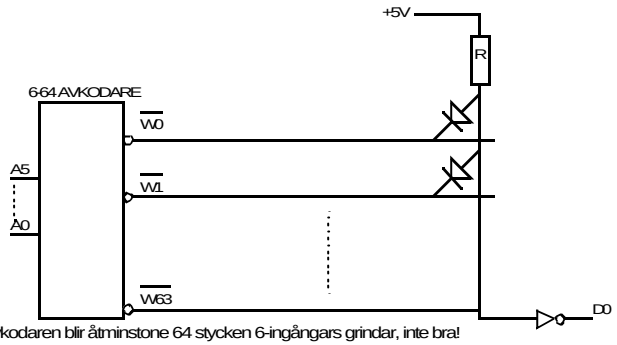
A0, **A1** = I0, I1, två-bitars ingång till avkodaren

D0, ... , **D3**, fyra-bitars avkodad utgång

Intern struktur för en 4 x 4 bitars diod-ROM



Minnesorganisation: Exempel 64 x 1 ROM

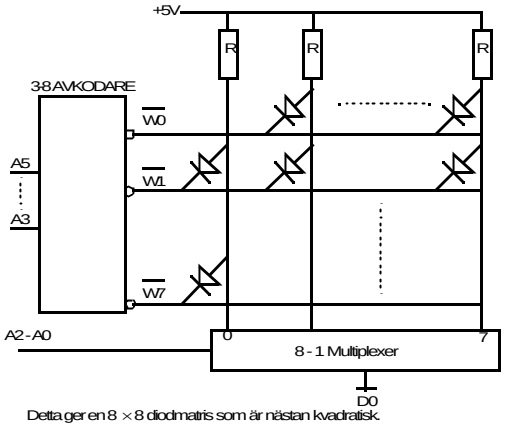


Avkodaren blir åtminstone 64 stycken 6-ingångars grindar, inte bra!

Kretsen blir väldigt hög och smal, inte bra. Man eftersträvar så kvadratiska kretsar som möjligt.

Minnet måste organiseras på ett annat sätt.

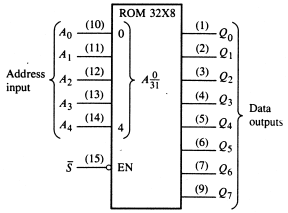
Minnesorganisation: 64 x 1 ROM med 2-dimensionell avkodning



Detta ger en 8×8 diodmatris som är nästan kvadratisk.

ROM som komponent

- **Symbol:**



- **Gränssnitt:**

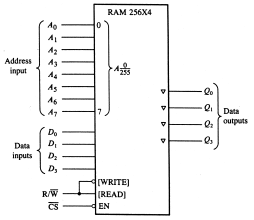
A4 - A0: Adresser

EN: Enable-signal som lägger ut addresserat data på utgångarna (Q7-Q0)

Q7-Q0: Datautgångar

RAM som komponent

- **Symbol:**



- **Gränssnitt:**

A4 - A0: Adresser

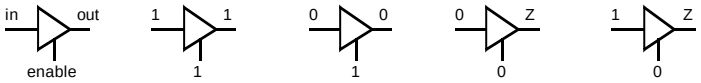
D4 - D0: Data som ska skrivas i minnet på positionen given av adressen.

R/W: Anger om minnet ska vara i skriv

Q7-Q0: Datautgångar som visar innehållet i minnet som pekas ut av adressen

Minnens gränssnitt mot buss

- **Bakgrund:**
  I många situationer så kopplas flera minnens utgångar samman på en gemensam buss. För att göra detta möjligt inför man "three-state logik":
- **Three-state buffer:**



- **Exempel:**

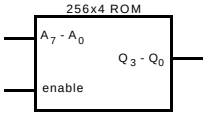
Endast den vänstra kretsen driver nod A

Båda kretsarna driver samtidigt nod A

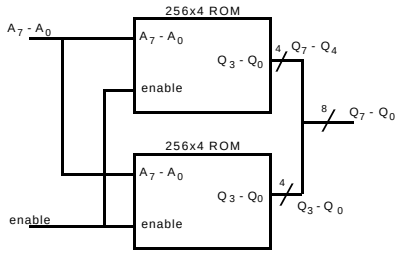
Minnesexpansion

- **Utökning av antal bitar i varje ord i minnet:**
  - **Exempel: Ta fram ett minne med 256 ord där varje ord är 8 bitar utifrån en ROM komponent med 256 ord á 4 bitar.**

ROM komponent:



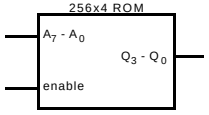
Utökning av minnet:



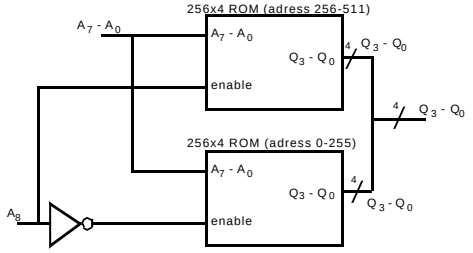
Minnesexpansion

- **Utökning av antalet ord i minnet:**
  - **Exempel: Ta fram ett minne med 512 ord där varje ord är 4 bitar utifrån en ROM komponent med 256 ord á 4 bitar.**

ROM komponent:



Utökning av minnet:



Minnesexpansion

- **Exempel: Ta fram ett minne med 1024 ord där varje ord är 4 bitar utifrån en ROM komponent med 256 ord á 4 bitar.**