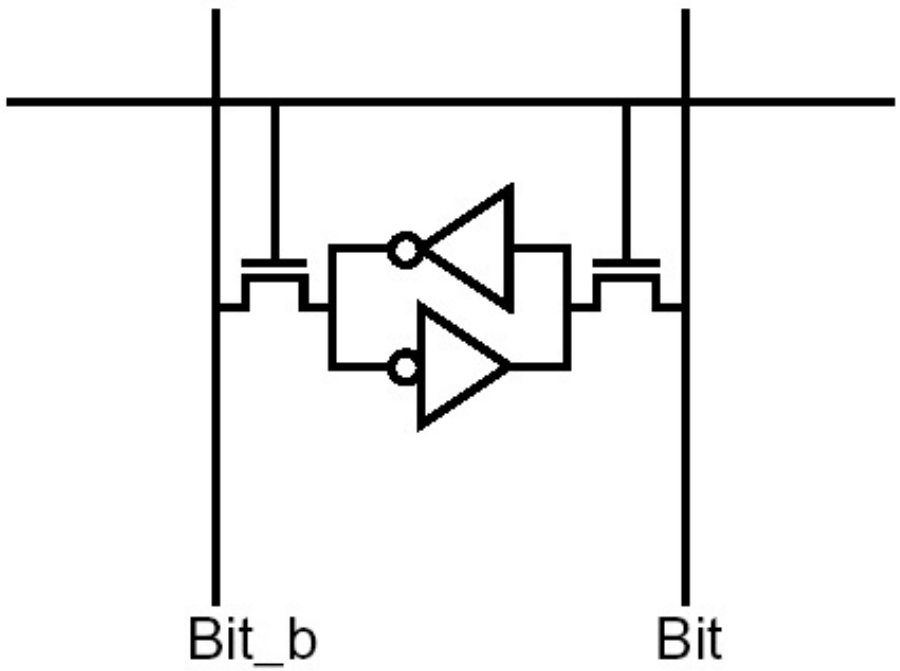
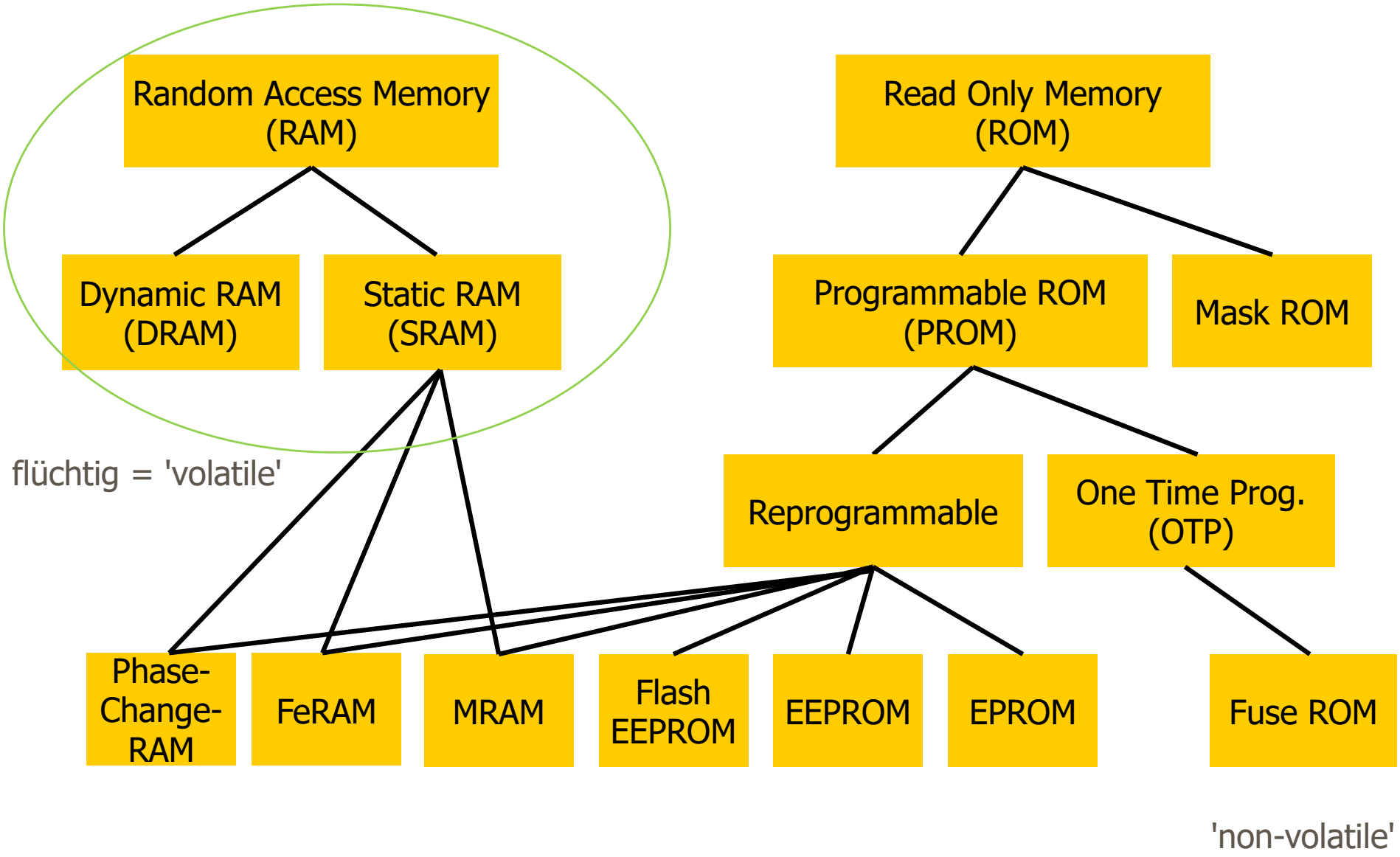
SPEICHERBAUELEMENTE

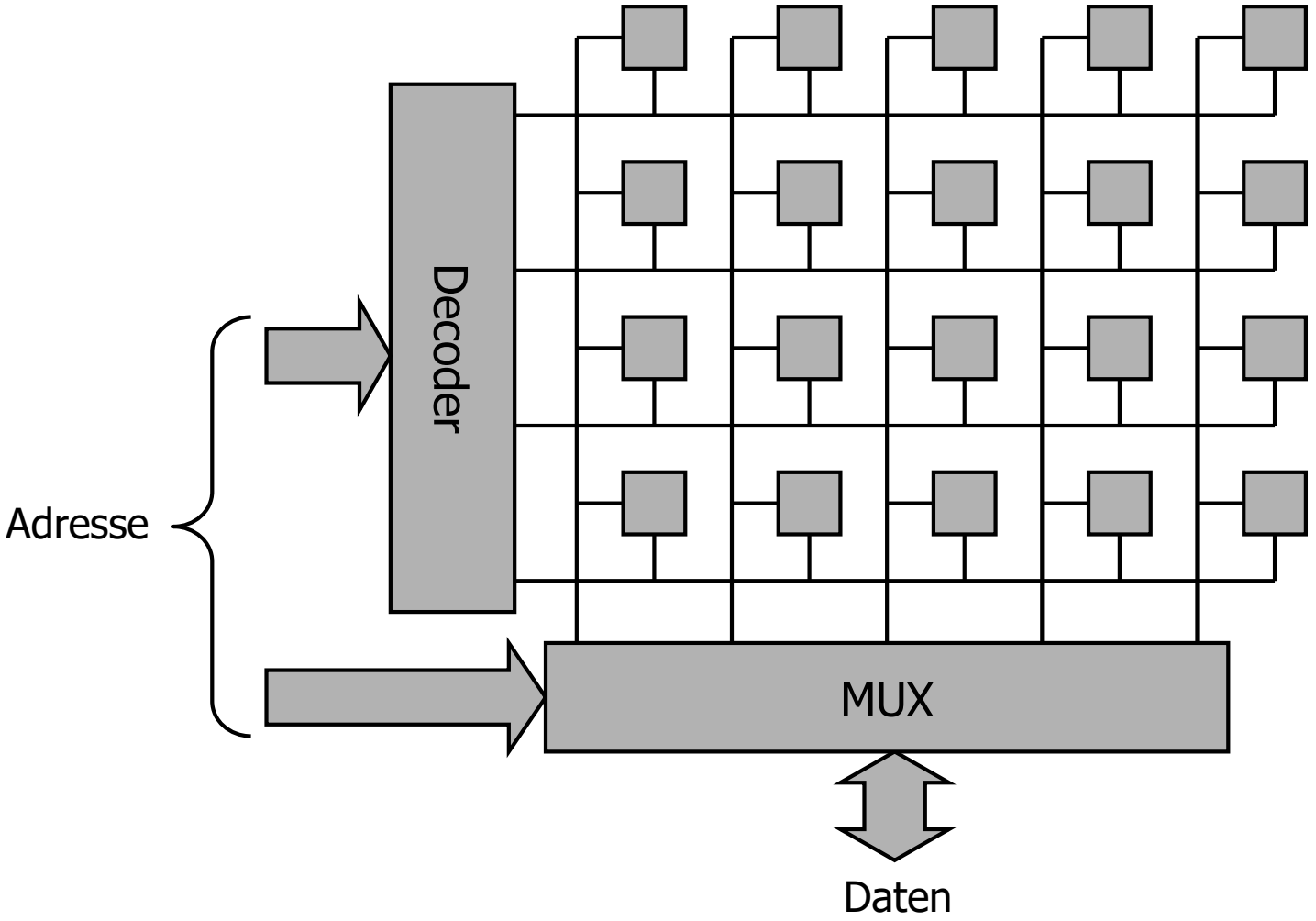


Speichertypen



Allgemeiner Aufbau – 'Organisation'

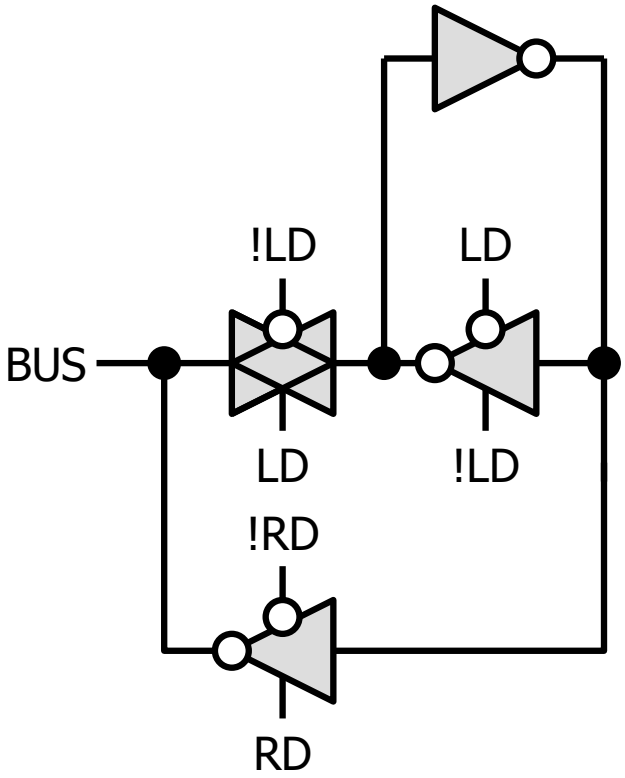
- Hohe Dichte erfordert regelmäßiges Layout
- Vereinfachung der Ansteuerung durch (möglichst) quadratische Matrix:
- N Zeilen und M Spalten bedienen N x M Speicherzellen

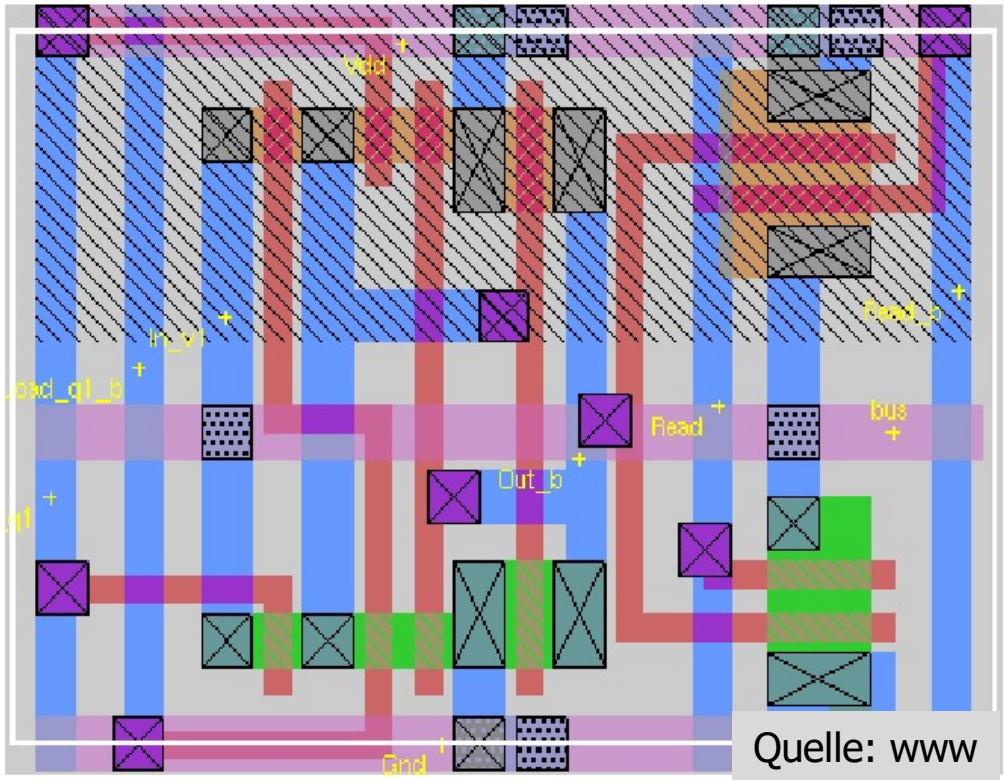


SRAM

Eine Möglichkeit: Latch als Speicherzelle

- Man könnte ein Latch z.B. aus Gated Invertern benutzen
- Diese Zelle benötigt 4 Steuerleitungen (LD, RD und Inverse), einen Bus, 2 Versorgungen
- Viele Leitungen und viele (12) Transistoren, sehr sicheres Design



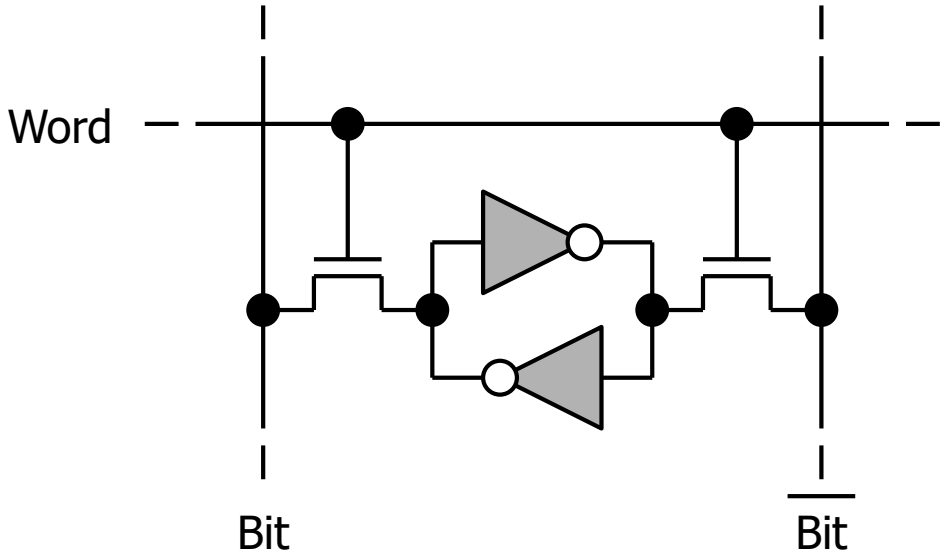


Quelle: www

- Layoutbeispiel (www., mit dem Programm 'MAGIC'):
- 78 x 55 Einheiten groß. Limitiert durch Verdrahtung!

Standard '6 Transistor – Zelle'

- 6 Transistoren: Zwei rückgekoppelte Inverter (2 NMOS, 2 PMOS) und zwei Schreib-Lese-Schalter (2 NMOS)
- Lesen und Schreiben erfolgen mit den selben Leitungen !
- Horizontale **Wort-Leitung** (wordline) aktiviert die Schreib/Lese-Schalter, die **Bit-Leitungen** ('bitlines': Bit und !Bit) verlaufen vertikal
- Zelle ist klein, da nur wenige Leitungen benötigt werden (word, bit, !bit, gnd, vdd)



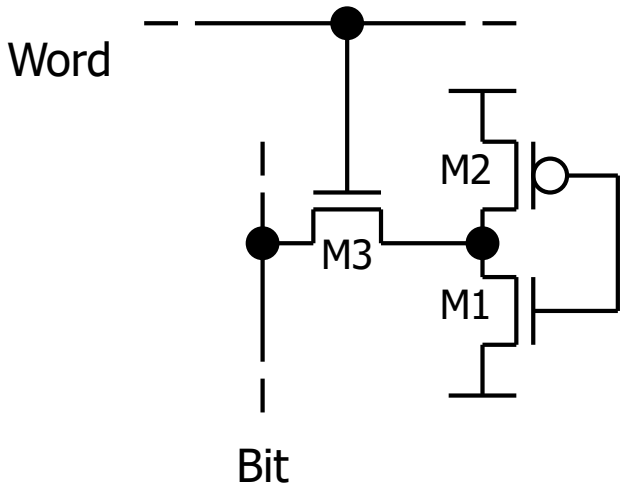
- Zum **Schreiben** werden Bit und !Bit auf 1/0 oder 0/1 gehalten und Word aktiviert
- Beim **Lesen** darf die Zelle nicht umkippen, wenn sie an den Bit/!Bit Bus geschaltet wird.
  Die Bit-Leitungen werden daher z.B. auf VDD vorgeladen (**'precharge'**)

Dimensionierung der Transistoren im 6T SRAM

- Es werden **nur Nullen geschrieben** (Einsen werden geschrieben, indem an der 'anderen Seite' eine Null geschrieben wird)
- Liegt eine Eins an der Bit-Leitung an wenn M3 durchschaltet, so darf die Zelle nicht geschrieben werden, sonst würde das Lesen nicht funktionieren!

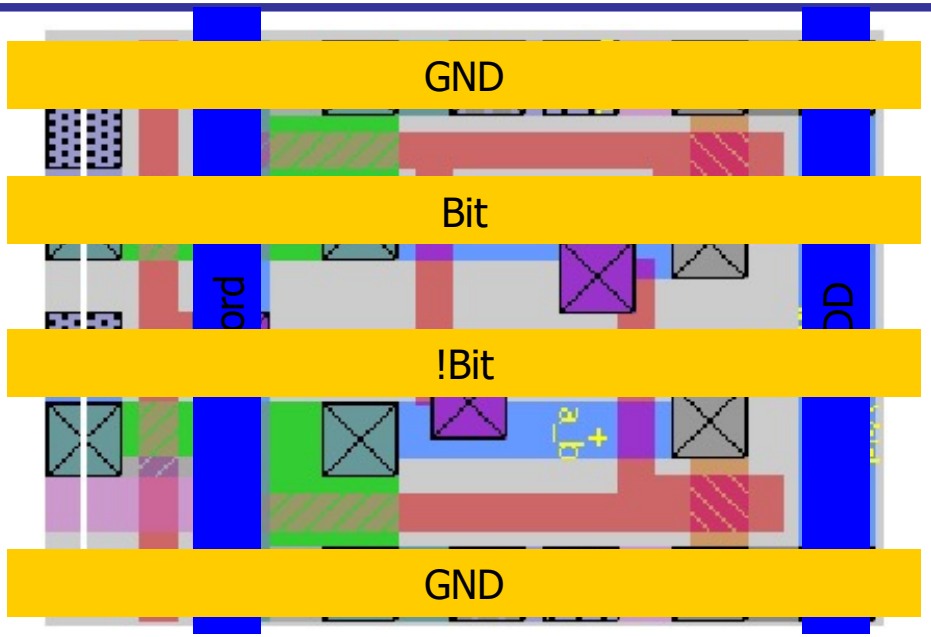
Dimensionierung der Transistoren (sehr verkürzt):

- Beim **Schreiben der Null** muß NMOS M3 stärker sein als PMOS M2.
  Da K_N = 2-3 K_P ist: $\mathbf{(W/L)_{M3}=1..1.5\ x\ (W/L)_{M2}}$
- Beim **Lesen einer Null**, wenn **auf dem Bus eine Eins** liegt:
  M1 muß stärker als M3 sein, damit beim Schließen von M3 die Spannung am Speicherknoten nicht über die Schwelle des Latches ansteigt. (Leider ist M1 im linearen Bereich, M3 in Sättigung.)
  Hier hilft der Substrateffekt von M3. Trotzdem: $\mathbf{(W/L)_{M1}=1.5..2\ x\ (W/L)_{M3}}$

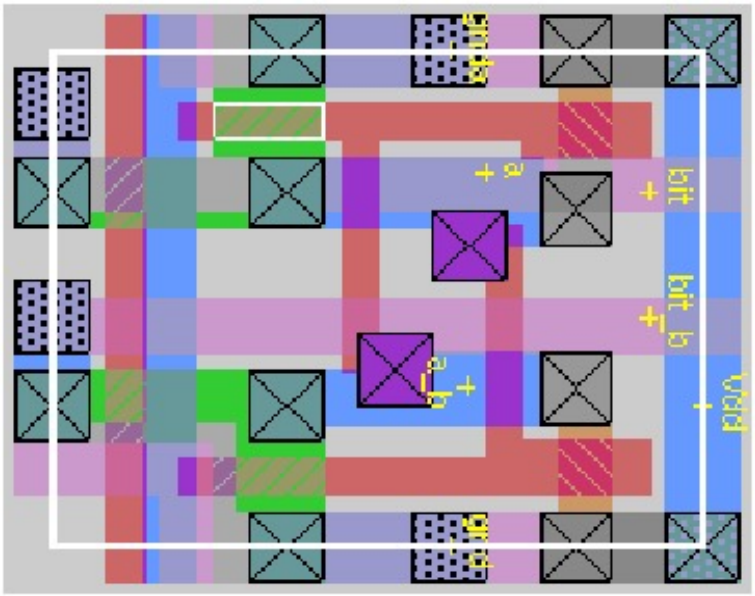


Layouts 6T SRAM

- 'Konservativ':
  - Substrat und Wannenkontakt in jeder Zelle
  - Wort-Leitung in Metall mit Kontakt in jeder Zelle
  - PMOS sehr schwach (3/3)
  - NMOS pulldown sehr stark (8:2)
  - 41 x 28 Einheiten (~1/4 der Latch Zelle)



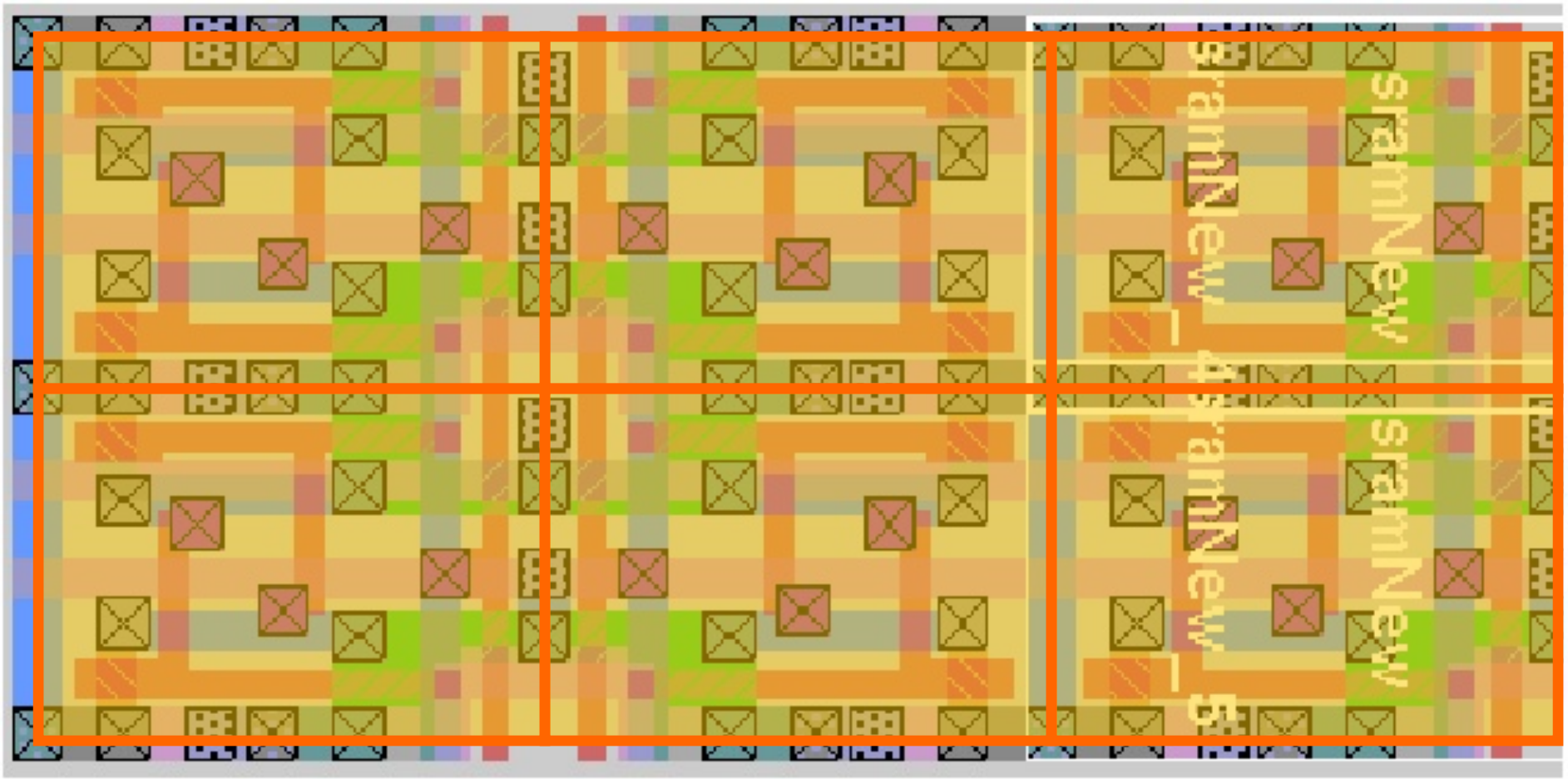
- Etwas kleiner:
  - Nur nwell Kontakt
    (Wannenkontakt wird nach jeweils N Zellen eingefügt)
  - NMOS pulldown ist etwas schwächer (6:2)
  - 36 x 28 Einheiten



- Die weiße Linie ist die effektive Zellgröße
- Kontakte und GND/VDD werden mit Nachbarn geteilt.

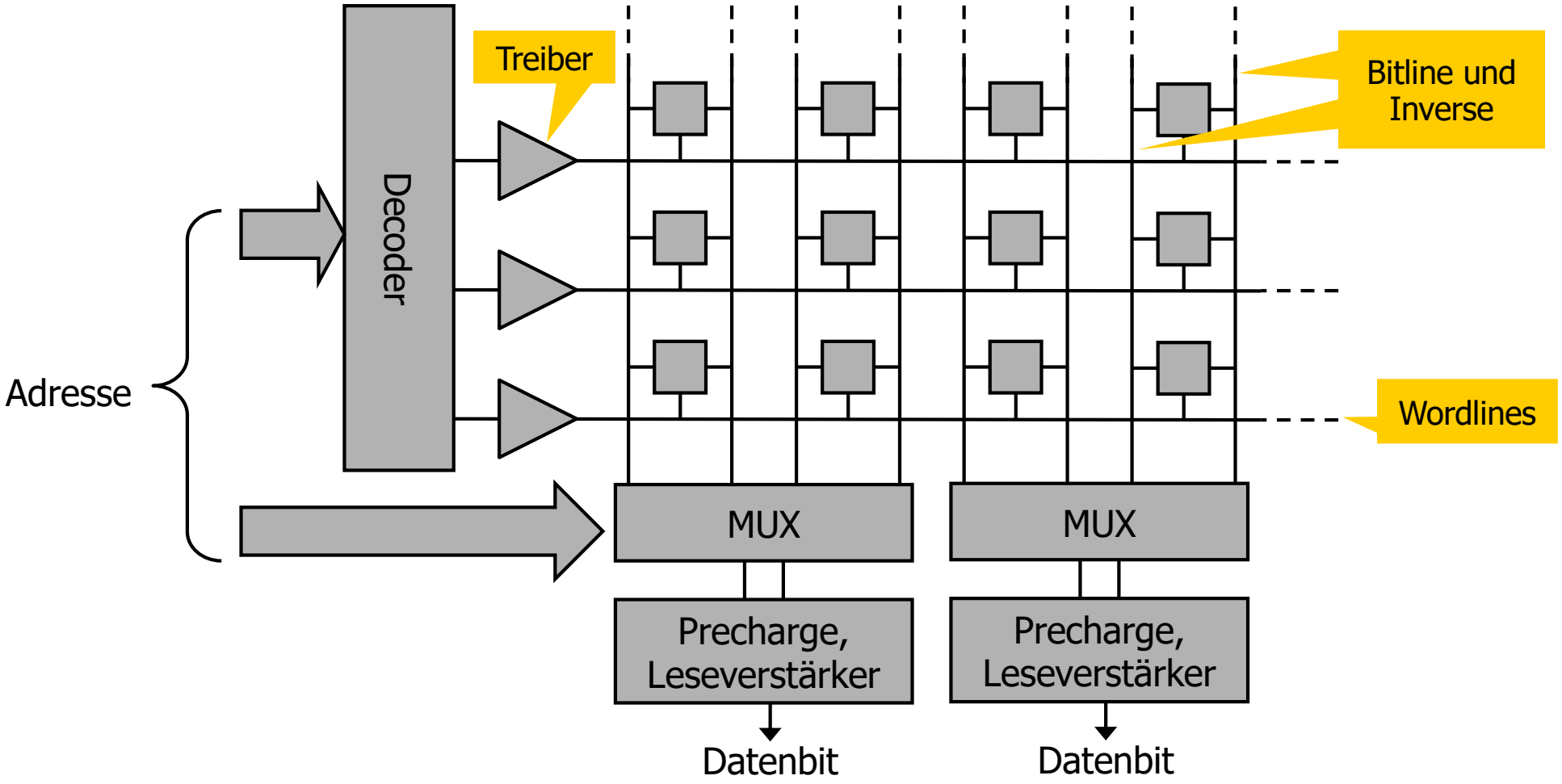
SRAM Matrix

- Hier 3 Zellen breit und 2 Zellen hoch
- Benachbarte Zellen sind jeweils horizontal / vertikal gespiegelt
- Kontakte werden in X- und Y- zwischen den Zellen geteilt



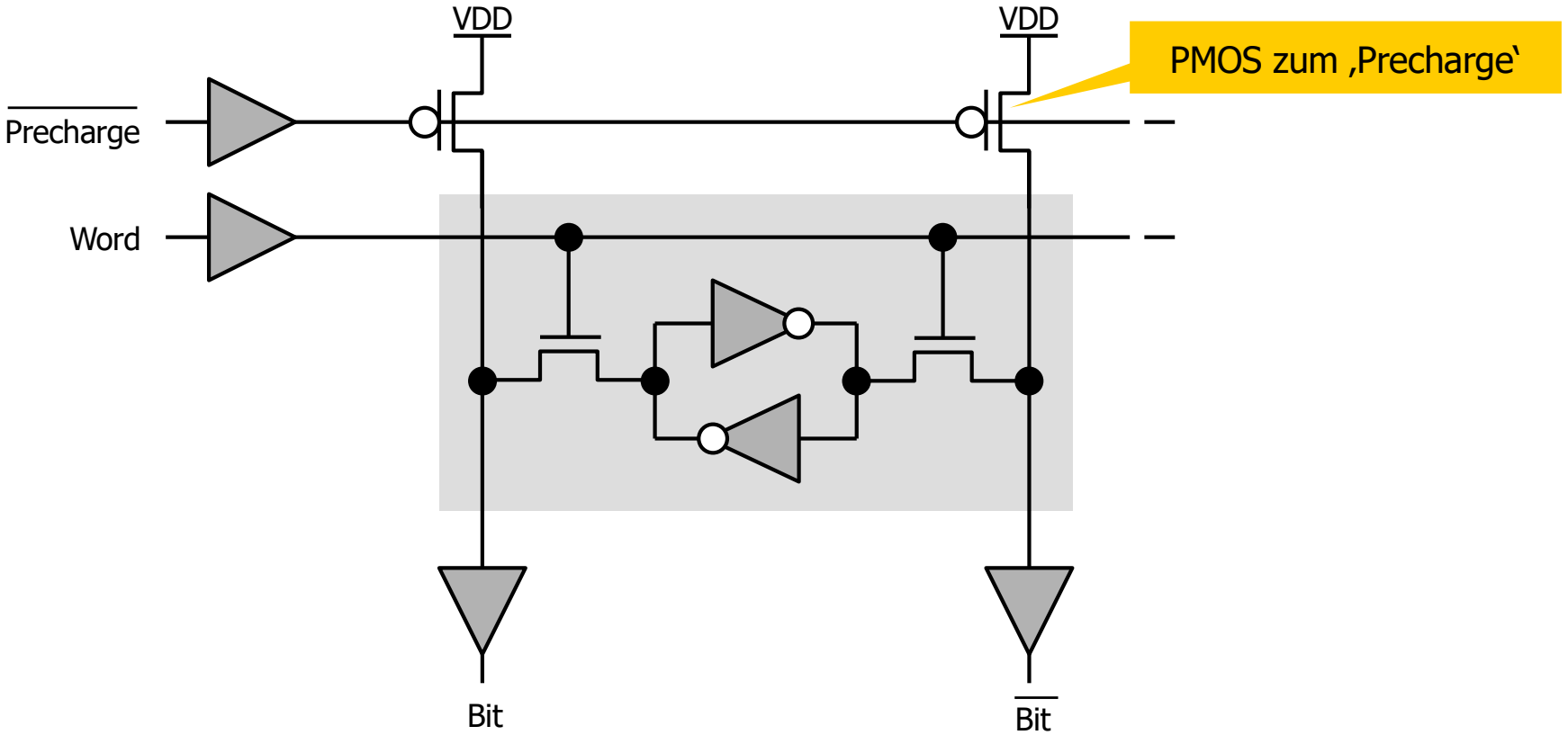
Betrieb der Matrix

- Ein Decoder wählt eine der Wort-Leitungen aus. Die **ganze Zeile** in der Matrix wird aktiviert.
- Wird nicht die ganze Zeile benötigt, so können **Multiplexer** an den Bit-Leitungen **Teile** auswählen
- In speziellen Speicherarchitekturen nutzt man das aus, um die nachfolgenden Zugriffe in eine Zeile schneller zu machen (‚Burst Mode')



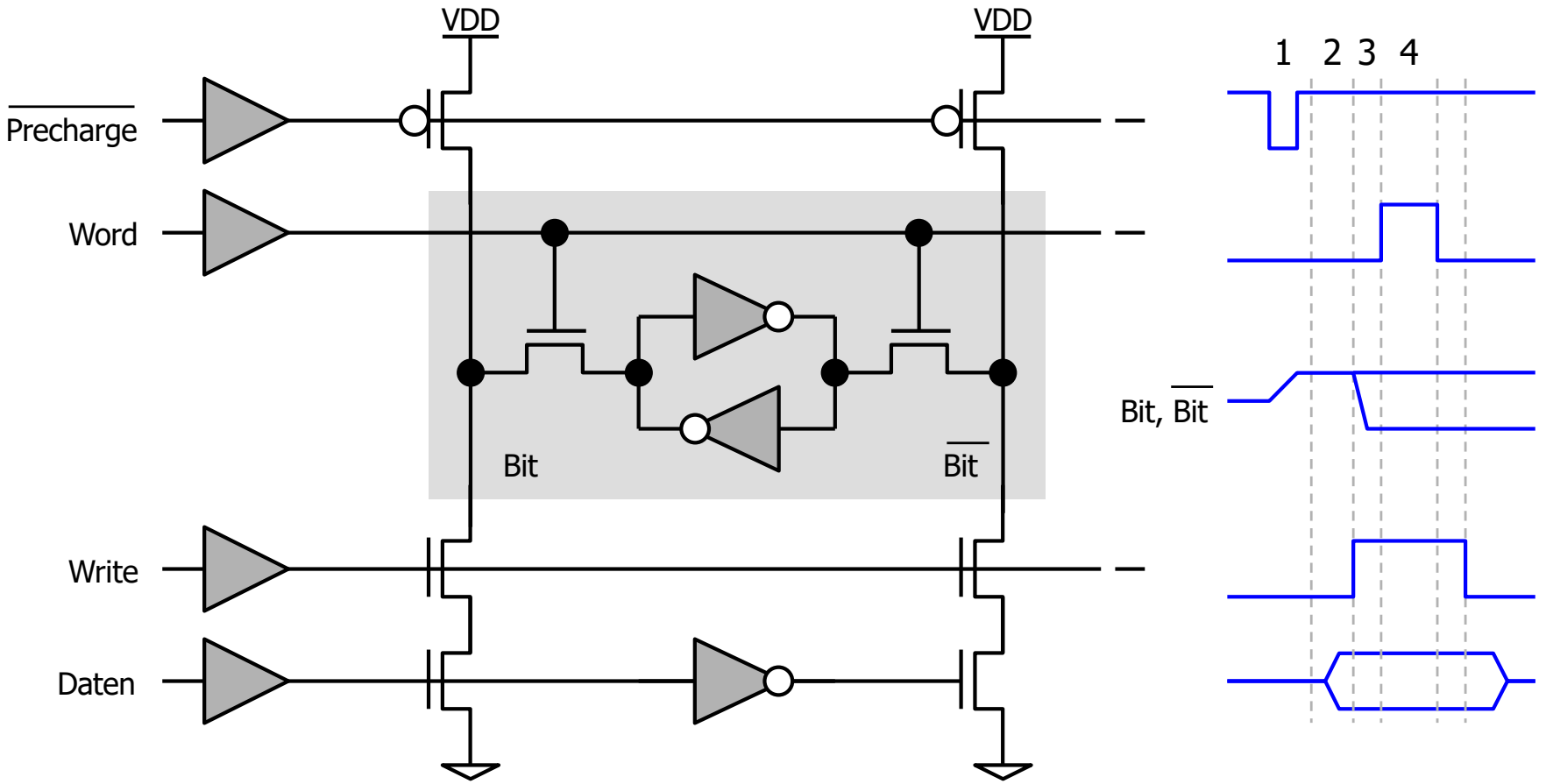
Lesen der 6T-Zelle

1. Die Bit-Leitungen werden auf VDD (oder auch etwas weniger) vorgeladen - 'precharge'
   (Alternativ kann man eine Pseudo-NMOS Last benutzen: PMOS als Stromquelle - langsamer!)
2. Die Wort-Leitungen werden aktiviert
3. Die SRAM-Zelle zieht eine der beiden Bitlinien auf Masse. Nach genügend langer Zeit stellt sich ein korrektes CMOS Signal ein!



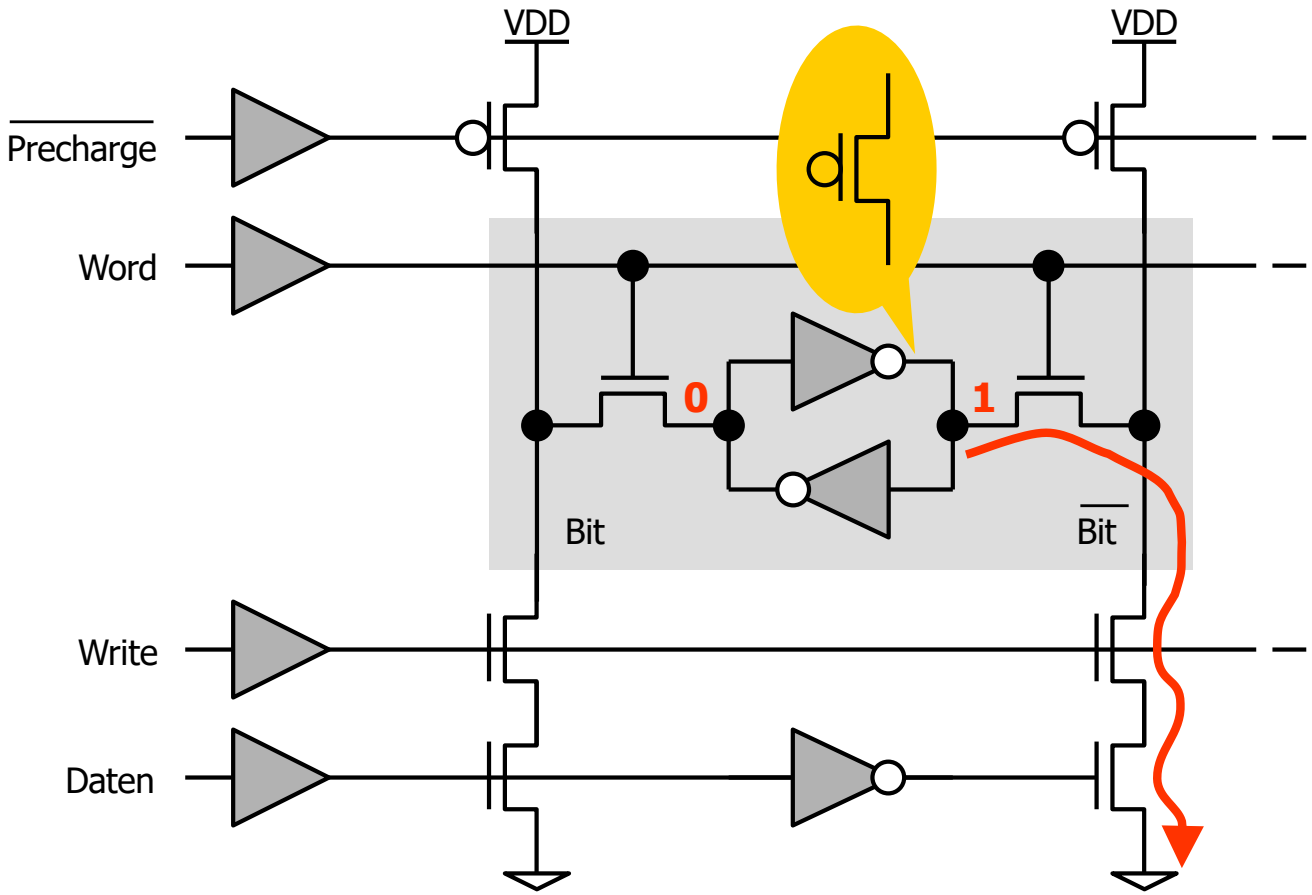
Schreiben der 6T-Zelle (Beispiel)

1. Die Bit-Leitungen werden durch einen Puls an PrechargeB auf VDD vorgeladen
2. Die Daten werden angelegt (das kann natürlich auch schon früher geschehen)
3. **Write** wird aktiviert. Dies zieht eine der Bit-Leitungen nach Masse
4. Die Wort-Leitung wird aktiviert. Dadurch wird die Zelle geschrieben.



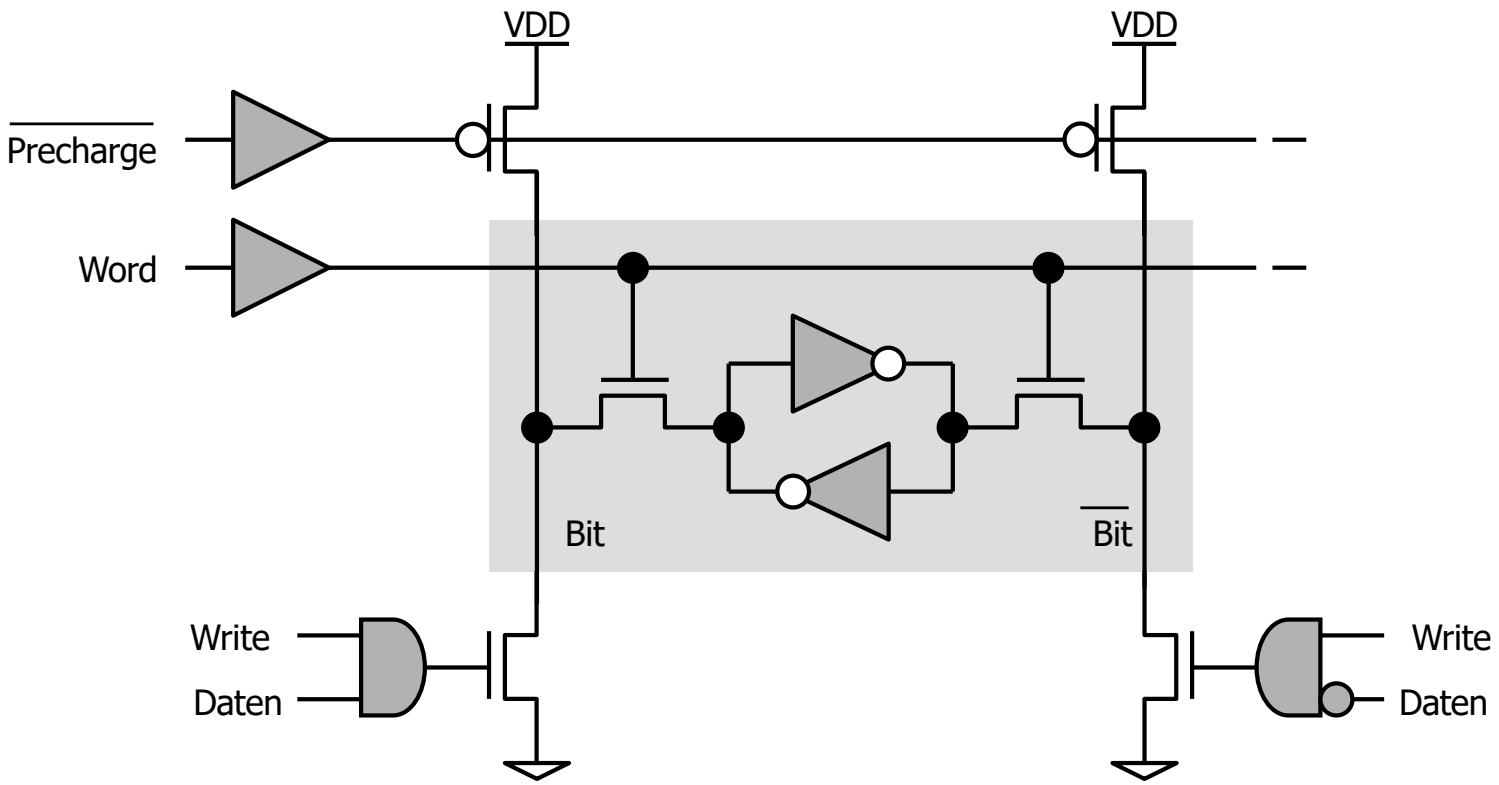
Write - Treiber

- Achtung: Die Null auf der Bitleitung muss 'gut' sein, d.h. das W/L der beiden NMOS, die sie auf den Bus treiben, muss groß sein. Sonst wird die SRAM - Zelle nicht umgeworfen!
- **Drei** NMOS in Serie treiben gegen den PMOS in der Zelle !



Verbesserte Schaltung für Write - Treiber

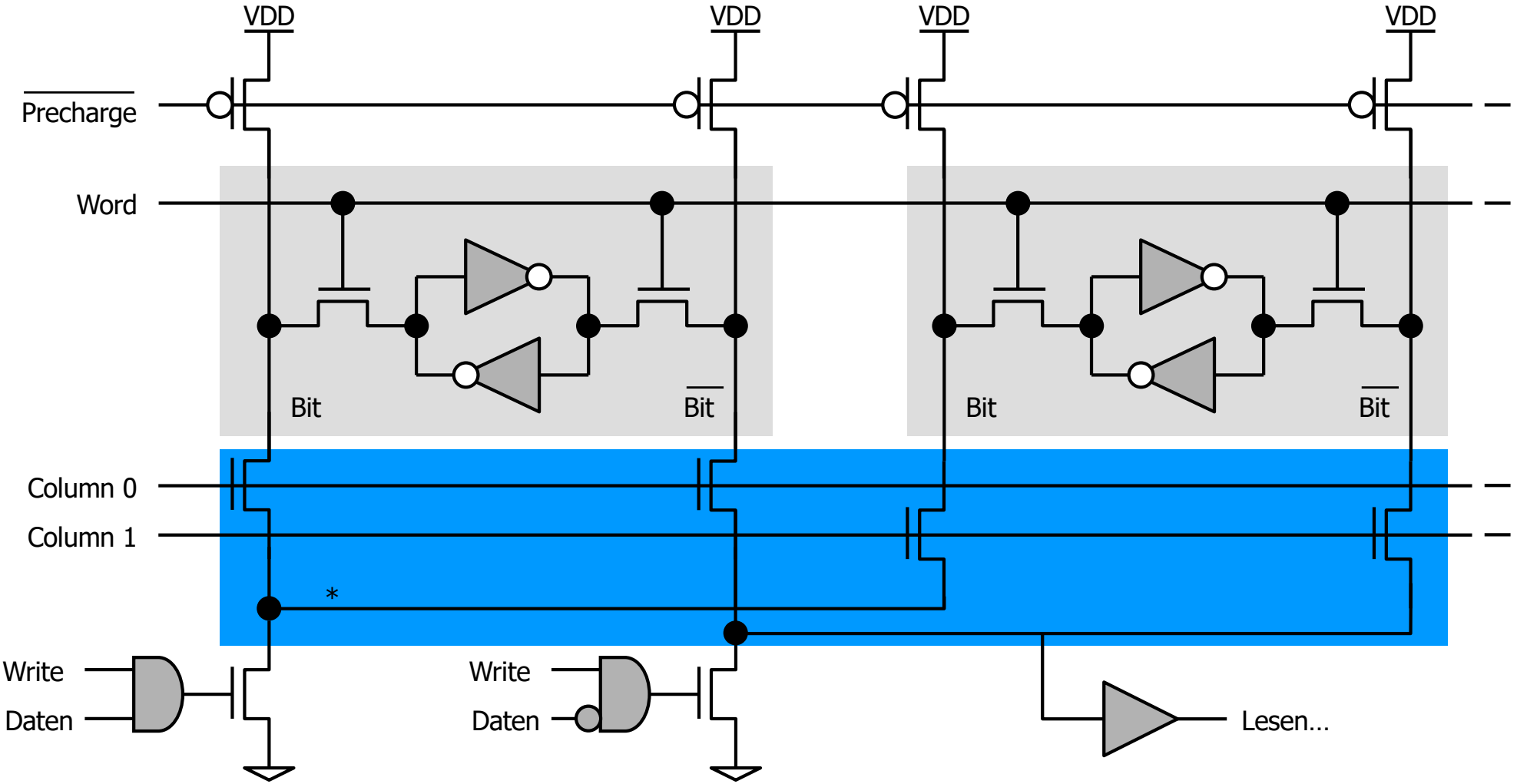
- Hier treibt nur ***ein*** NMOS nach Masse



- (Alternativ kann auch ein Treiber mit Enable (z.B. Gated Inverter) die Bit-Leitungen aktiv treiben, d.h. auch das High-Level auf den Bit-Leitungen aktiv erzeugen)

Bitline Multiplexing

- Der Pitch (räuml. Periode) der Bit-Leitungen ist sehr klein. Die Schreib/Lese Schaltung benötigt mehr Platz.
- Man benutzt daher oft *Multiplexer* um eine von mehreren Bit-Leitungen auszuwählen.
- Die folgende Lösung fügt wieder einen NMOS hinzu, der beim Schreiben beachtet werden muss.
  Ein Precharge am Ausgang des MUX (*) würde das high-Level verbessern.



Leseverstärker (sense amplifiers)

- Die Buskapazitäten sind bei großen Arrays sehr hoch.
- Sie entstehen durch
  - Die Diffusionkapazität der Schreib/Lese-Transistoren
  - Die Gate-Überlapp-Kapazität zwischen Bitleitung und Wortleitung
  - (Leitungskapazitäten)
- Gespiegeltes Layouts (gemeinsame Benutzung der Diffusion in Nachbarzellen) reduziert die Kapazität

- Es dauert daher sehr lange, bis die 'schwache' RAM-Zelle eine Bitleitung auf Masse gezogen hat:

$$\frac{\Delta U}{\Delta T} = \frac{I}{C} \quad \Leftrightarrow \quad \Delta T = \frac{C}{I} \cdot \Delta U$$

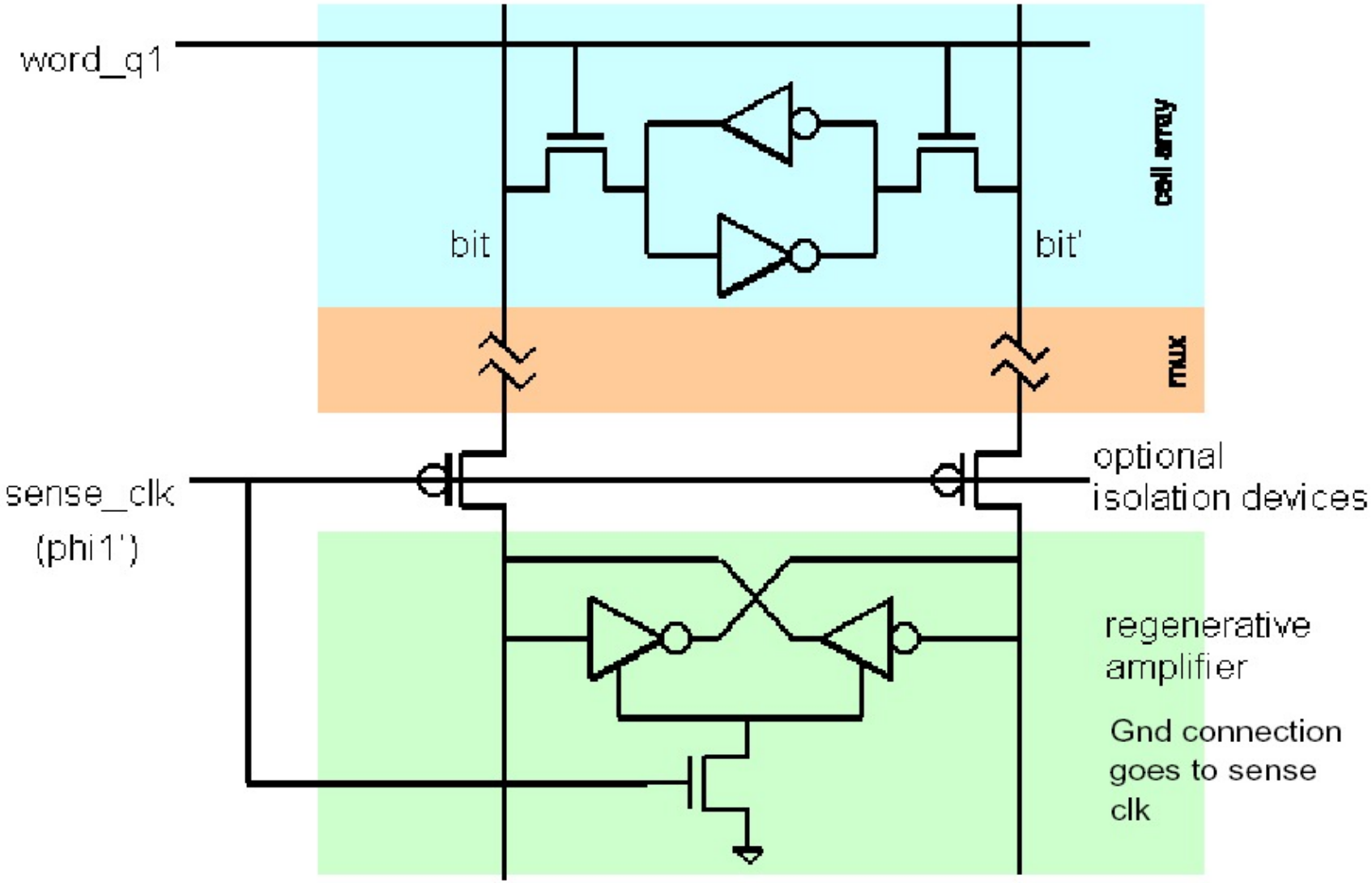
gegeben

- Verbesserung: ΔU kleiner machen, indem an die Bitleitung ein **Leseverstärker** angeschlossen wird

- Für die Implementierung dieser 'Sense-Amplifier' gibt es viele Möglichkeiten, z.B.
  - 'echte' Differenzverstärker (insbesondere f. DRAMs)
  - regenerative Latches

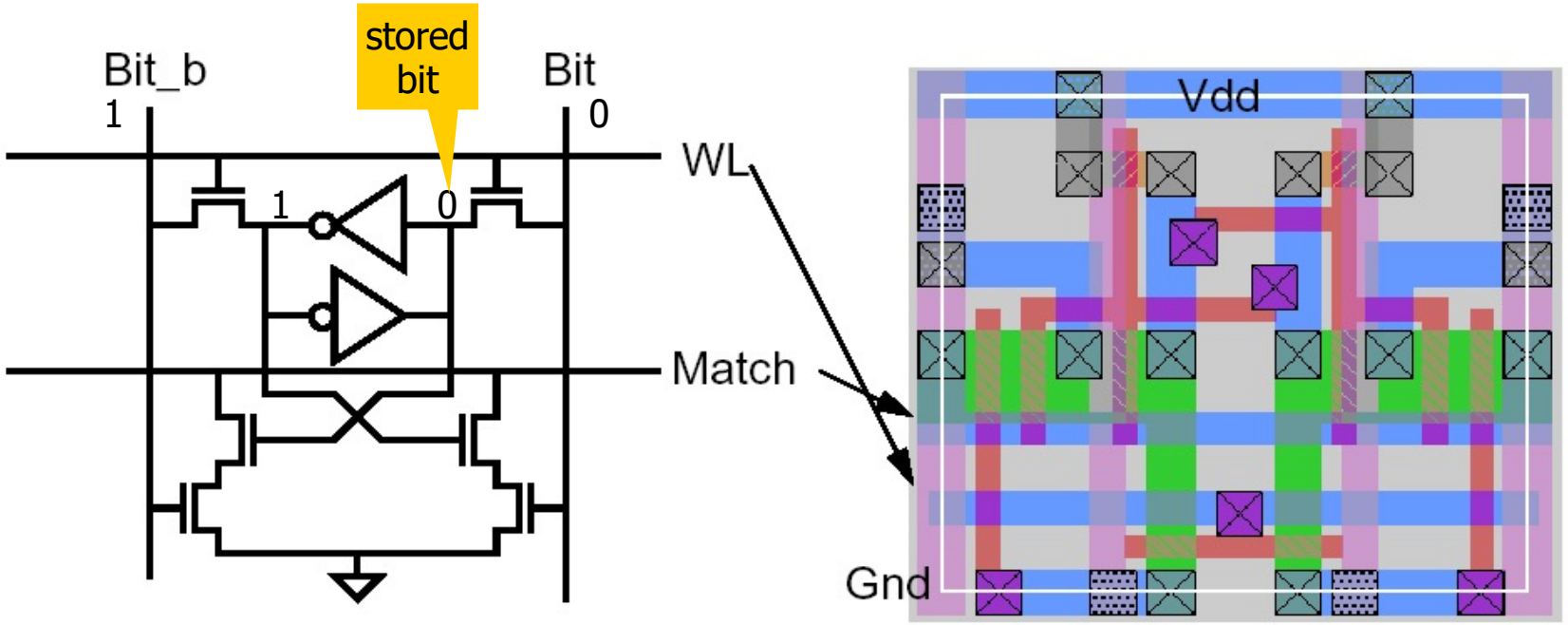
Leseverstärker

- Hier sehr einfach: Regeneratives Latch mit positivem Feedback
- Wird nur beim Lesen aktiviert



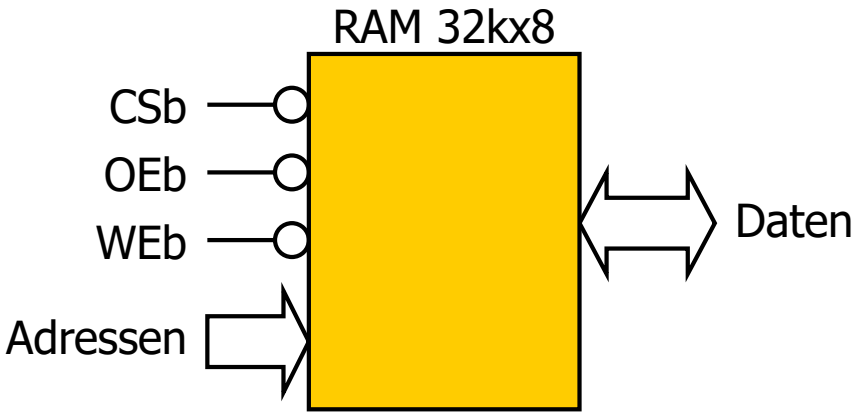
Content Addressable Memory (CAM)

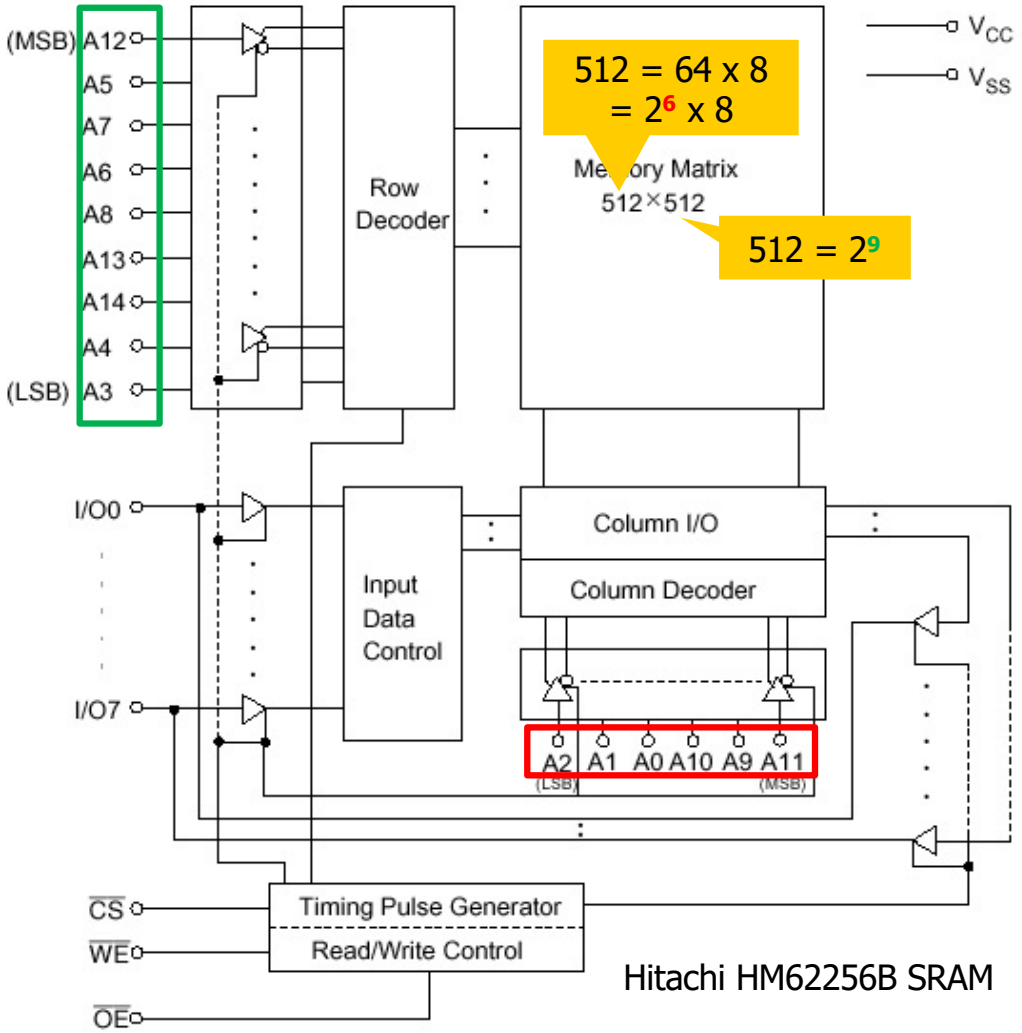
- Die Zelle vergleicht, ob ihr Inhalt mit einem gegebenen Wert übereinstimmt
- 'Adressierung über den Inhalt der Zelle' – Content Adressable Memory (CAM)
- Jede Zelle enthält ein XOR, das den Speicherwert mit den Bitleitungen vergleicht.
- Die Implementierung des XOR ist einfach, da die Inversen beider Eingangssignale vorliegen
- Die Zelle zieht die 'Match' – Leitung nach **Masse**, wenn die Werte **ungleich** sind (precharge!)
- Der Vergleich kann mit **vielen Bits gleichzeitig** durchgeführt werden, die Match Leitung bildet dann das Wired-OR der Vergleichsergebnisse. Nur wenn ALLE Bits übereinstimmen bleibt Match auf 1.
- Irgendeine Form des Precharge / Pullup für Match ist nötig.
- Einzelne Bits können sogar aus dem Vergleich ausgeschlossen werden, indem beide Bit-Leitungen auf Masse gezogen werden.



RAM Block Diagramm

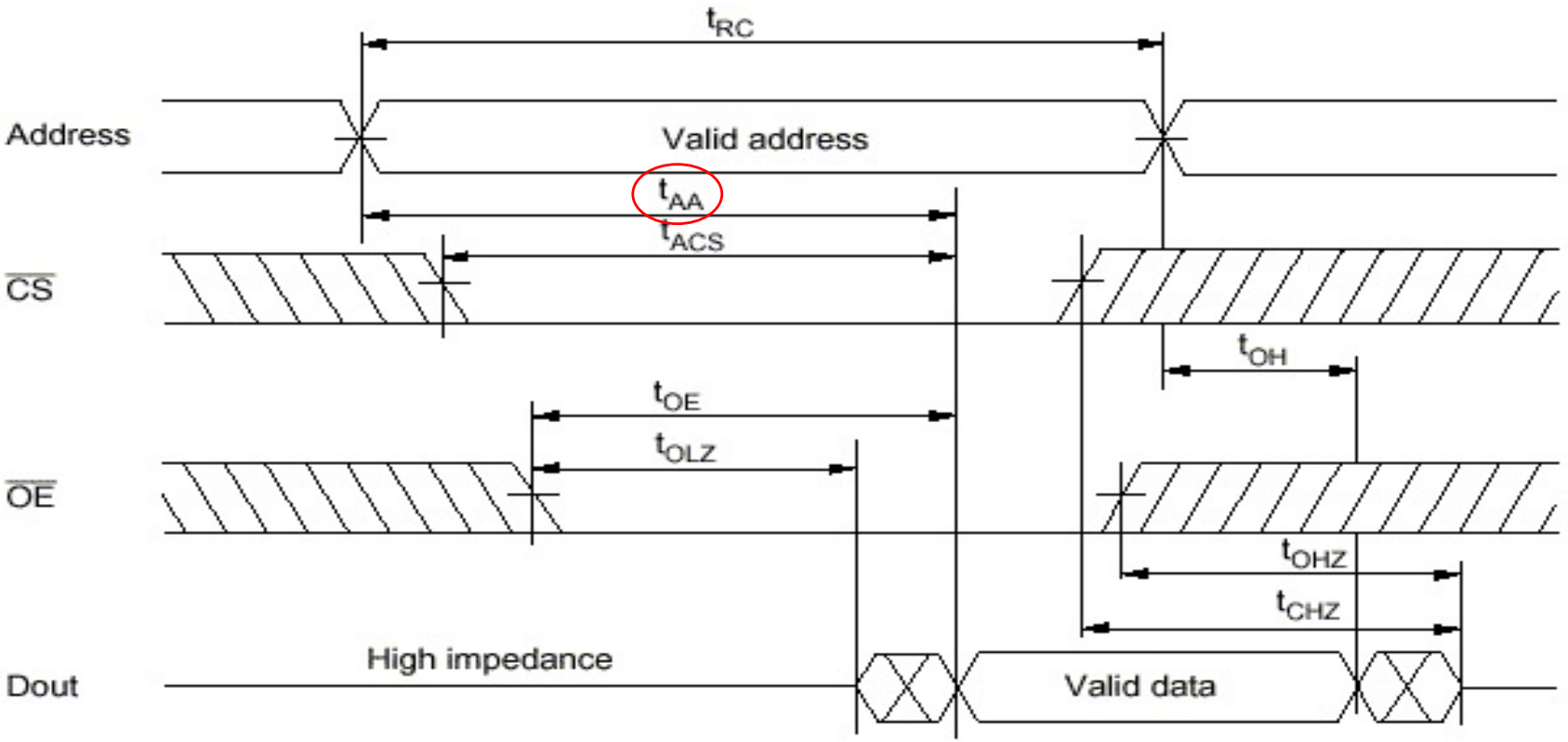
- Organisation meist in 8 (4) Bit breiten Worten
- Ansteuerung über
  - Adressen
  - Chip-Select Leitung (meist Active low)
  - Output-Enable Leitung
  - Write-Enable Leitung
- Beispiel: 32k x 8 RAM
  - 32k = 2^{15}, also 15 Adress- Bits
  - 9 Adressen für 512 Zeilen
  - 6 Adressen für 64 Gruppen von 8 Spalten





Hitachi HM62256B SRAM

SRAM Lesezugriff

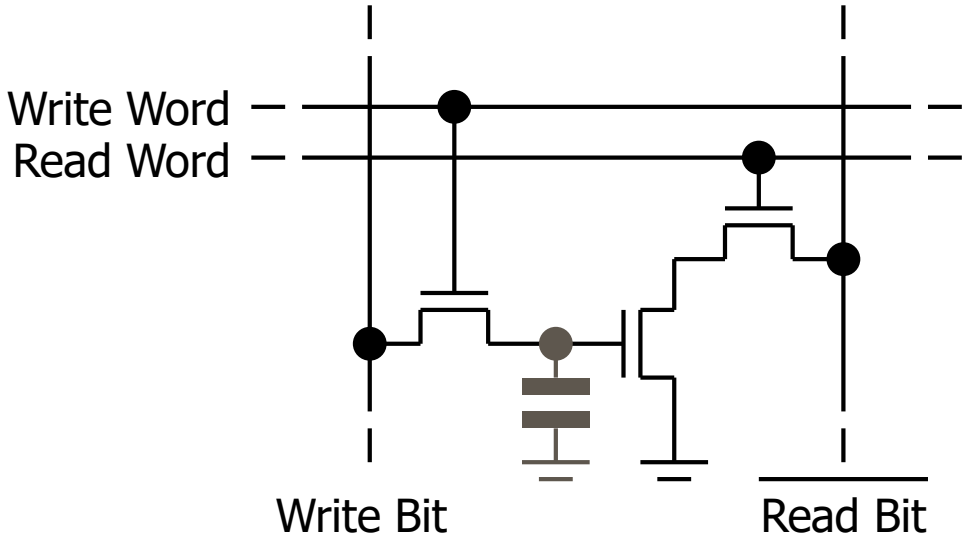


- Address Access Time – t_{AA} Zeit bis Daten nach Anlegen der Adressen gültig sind
- CS to Access Time – t_{ACS} Zeit bis Daten nach Auswahl des Chips gültig sind
- OE to Output Valid – t_{OE} Zeit bis Ausgänge aus dem Tri-State Zustand aufwachen und Daten gültig sind
- Read Cycle Time – t_{RC} Bestimmt maximale Lesefrequenz

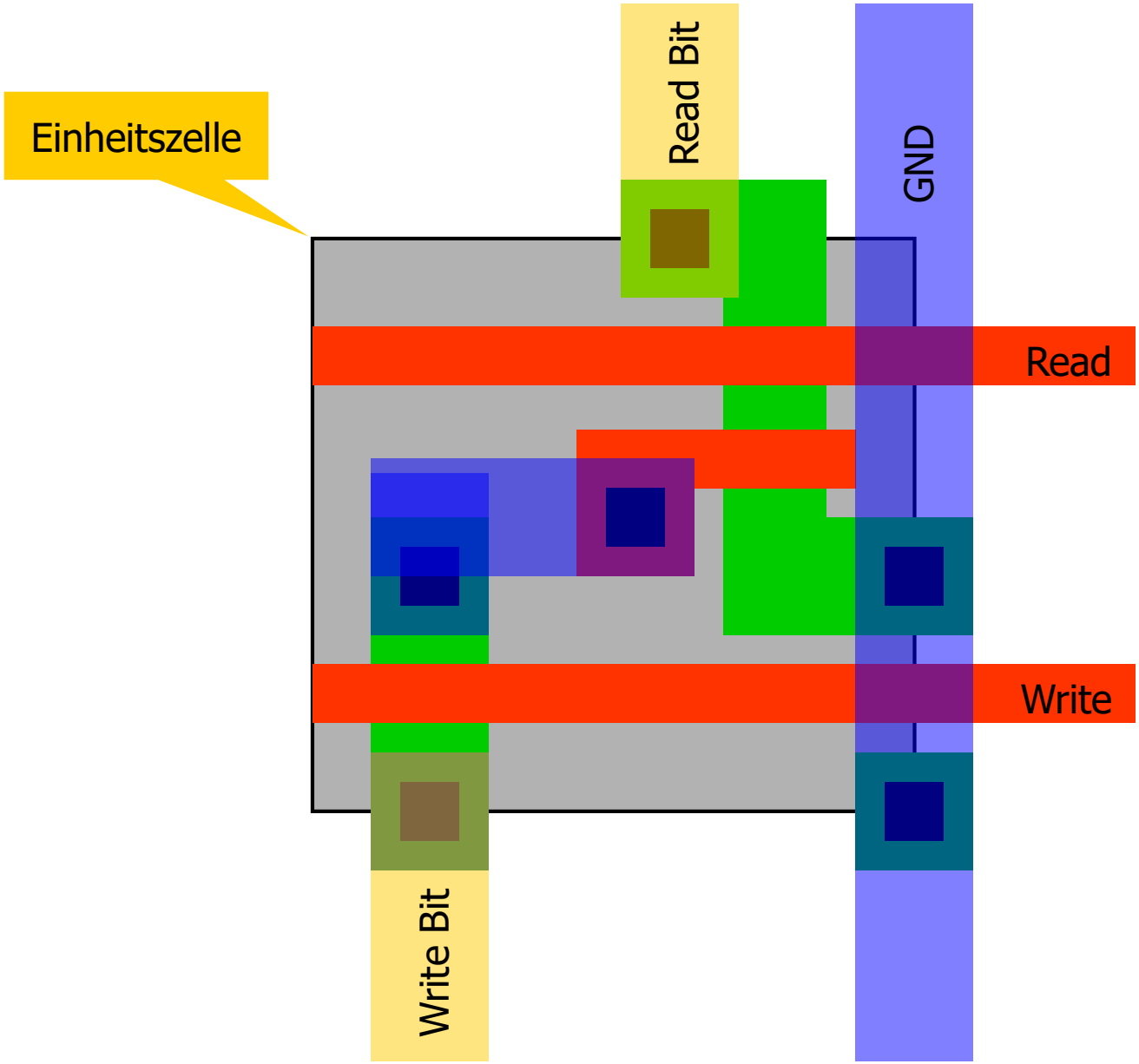
DRAM

3T DRAM

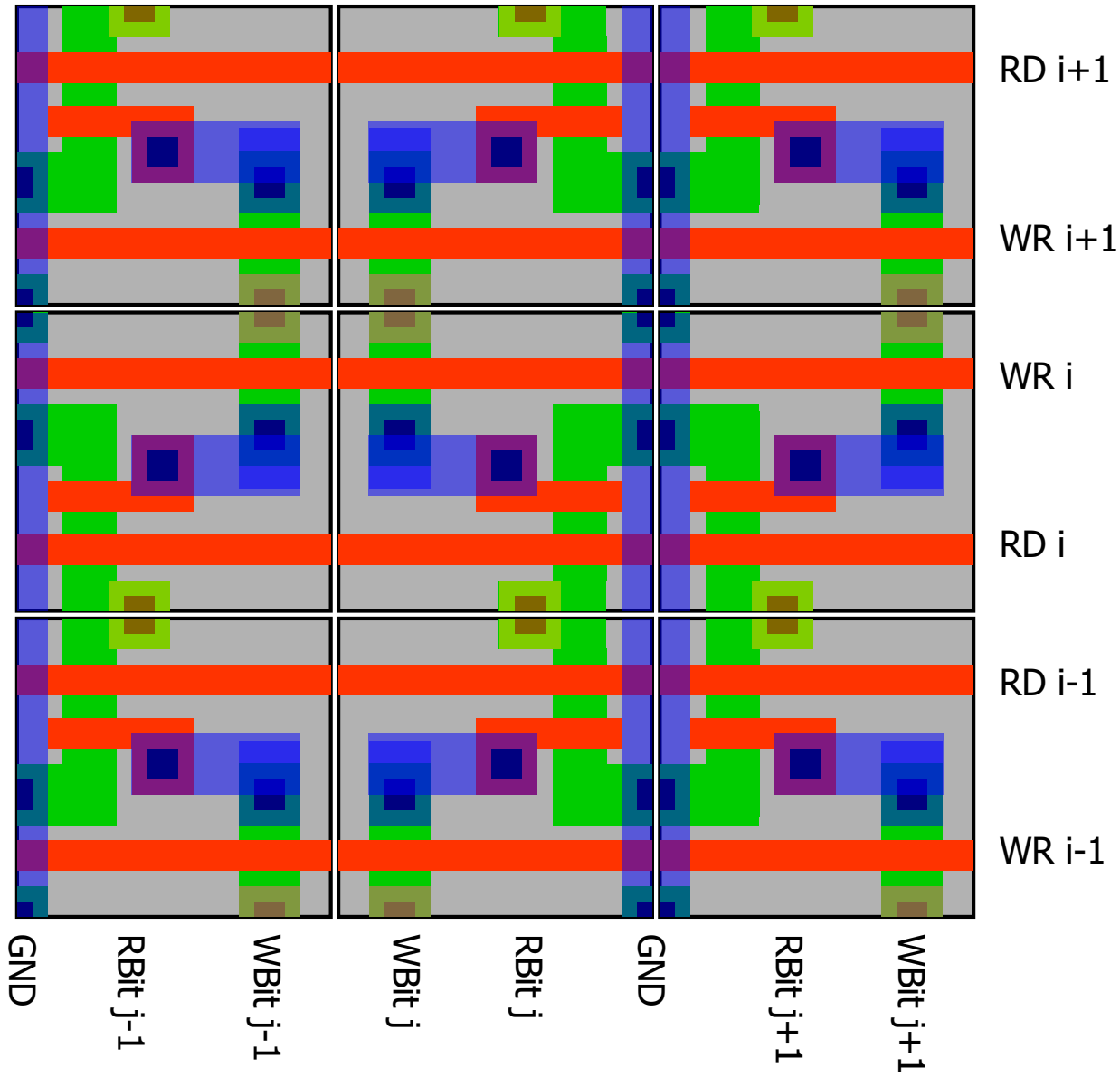
- 3 Transistor-Zelle benötigt:
  2 Wort-Leitungen (Write/Read) und 2 Bit-Leitungen (Read/Write) und Ground (kein VDD!)
- Die Bit-Leitungen für Wite/Read können auch verbunden werden.
- Es werden nur NMOS Transistoren benötigt!
- Speicherknoten wird beim Schreiben einer 1 nur auf $V_{WriteWord} - V_{TN}$ aufgeladen. Der Strom im Speichertransistor beim Lesen ist daher geringer
  Die Spannung auf der Schreibe-Wortleitung wird daher meist *höher als VDD* gemacht, um eine gute 1 zu schreiben
- Flächengewinn nicht signifikant, da viele Leitungen
- Schreiben und Lesen sind gleichzeitig möglich. Lesen zerstört die gespeicherten Daten nicht.



Ein mögliches 3T DRAM - Layout

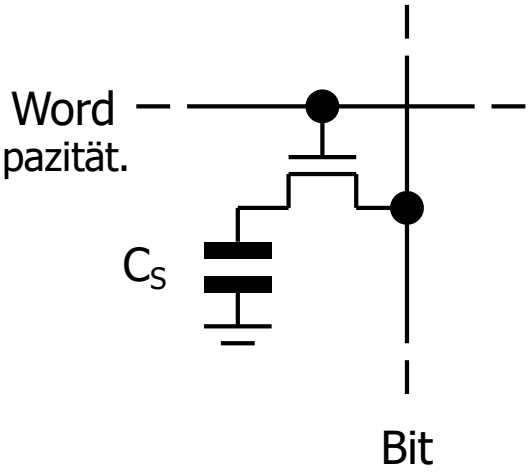


Array von Einheitszellen



1T DRAM

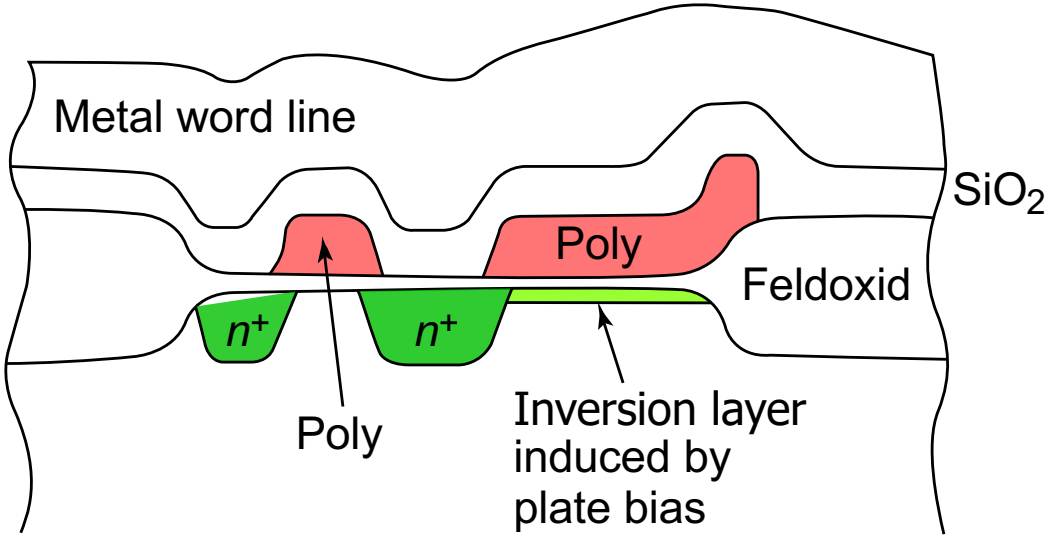
- Nutze **separate Kapazität** zur Speicherung
- Diese wird durch **spezielle Prozeßschritte** vergrößert.
- Benötigt nur 1 Transistor
- Eine Wort-Leitung, eine Bit-Leitung, Ground ⇒ nur 3 Leitungen!
- Sehr kompakt !



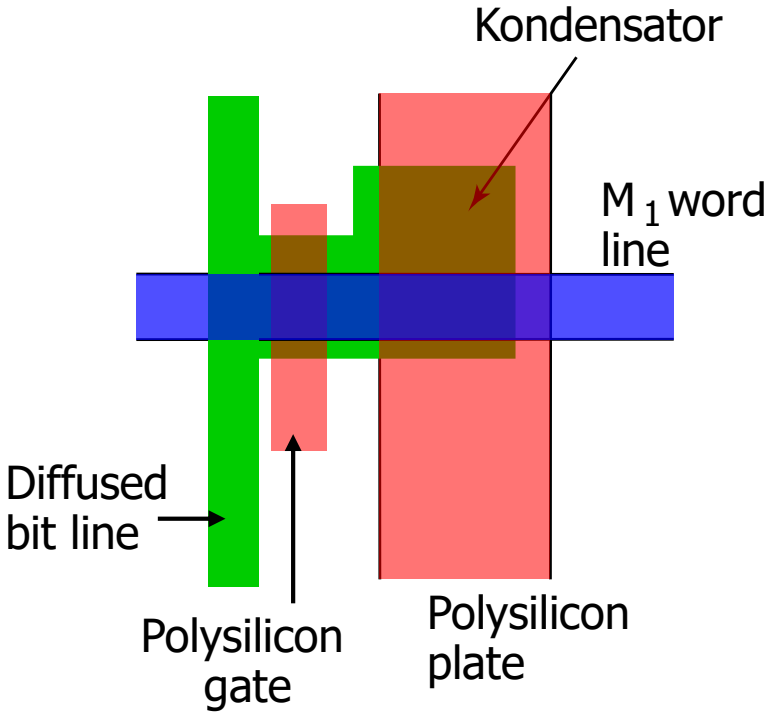
- **Lesen schwierig**: Die kleine Ladung auf der Speicherkapazität verteilt sich beim Schließen des Schalttransistors auf die gesamte Buskapazität. Die Spannungsänderung ist daher nur sehr klein (<~200mV):
  $\Delta V = (V_{bit} - V_{Cap}) \times C_S / (C_S + C_{bus})$
- Daher sind gute 'Leseverstärker' nötig.
  Z.B. Ladungsverstärker, die die Spannung des Busses konstant halten.
  Die Bit-Leitung wird vor dem Lesen z.B. auf VDD/2 vorgeladen.
- **Lesen ist destruktiv**.
  Inhalt muss nach dem Lesen erneut geschrieben werden.
- Dies ist sowieso regelmäßig zum **Refresh** notwendig (auffrischen der Daten, die ja durch die dynamische Speicherung nur einige ms lang erhalten bleiben)
- Der Widerstand des Schalters wird erniedrigt, indem die Wort-**Steuerspannung > VDD** gemacht wird.
  ('Ladungspumpe' auf dem Chip)

1-T DRAM Zelle: Standard Technologie

- In Standard Technologie wird als Kondensator ein Gate-Oxid benutzt
- Fläche relativ groß

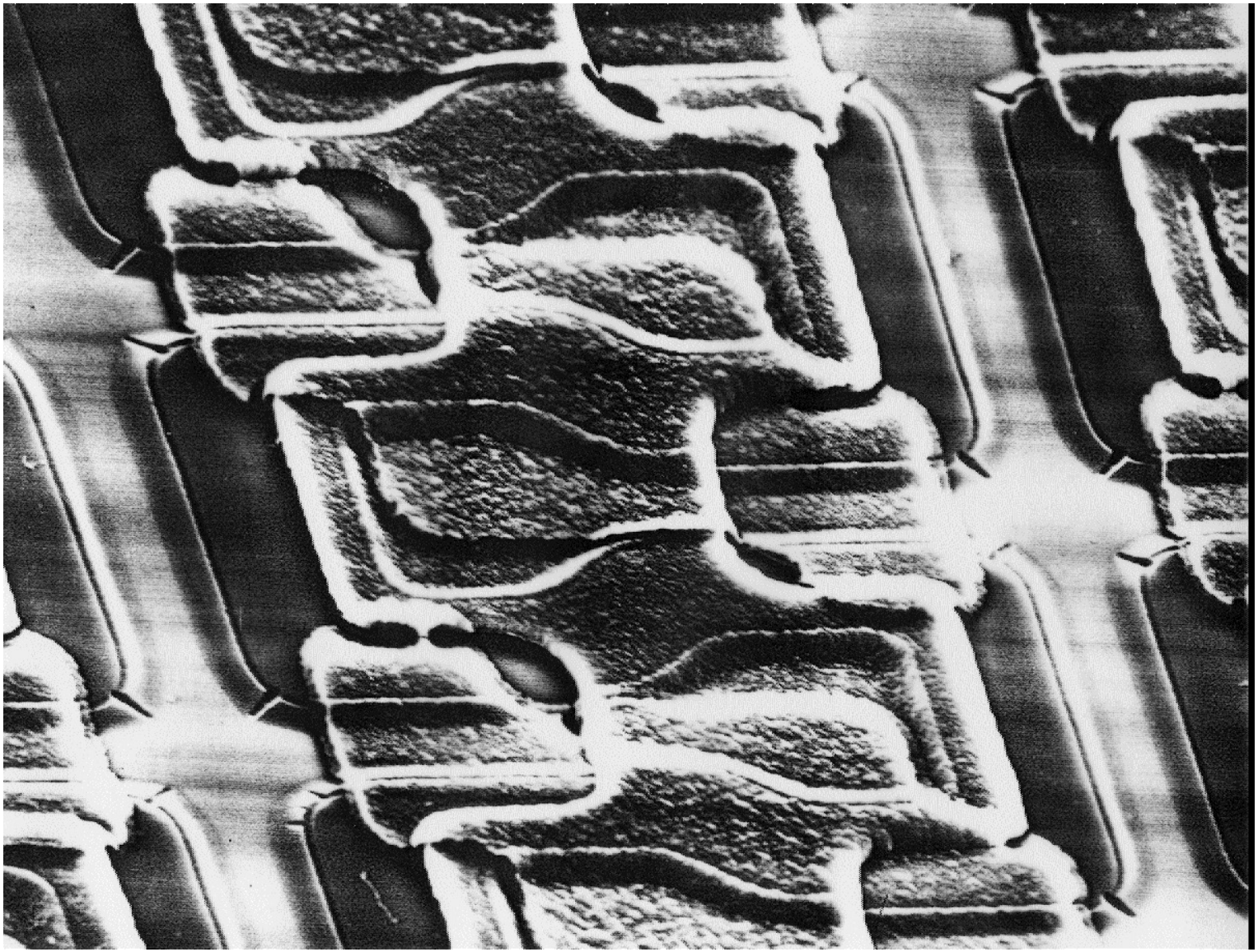


Querschnitt



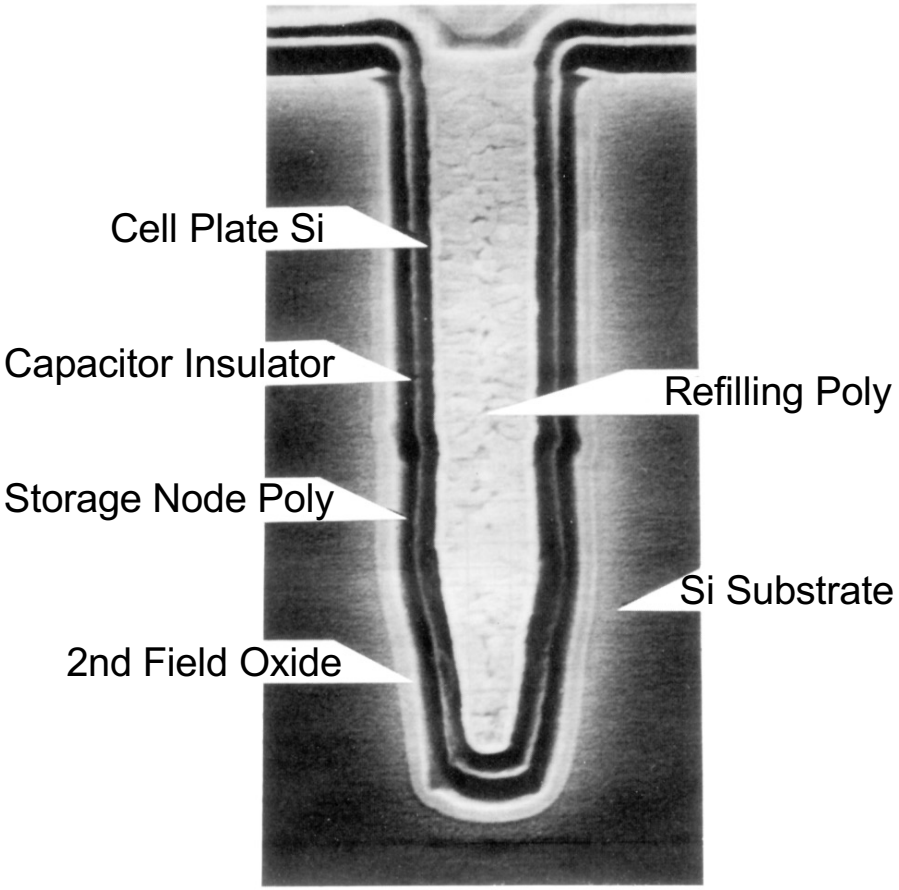
Layout

REM Aufnahme poly-diffusion capacitor 1T-DRAM

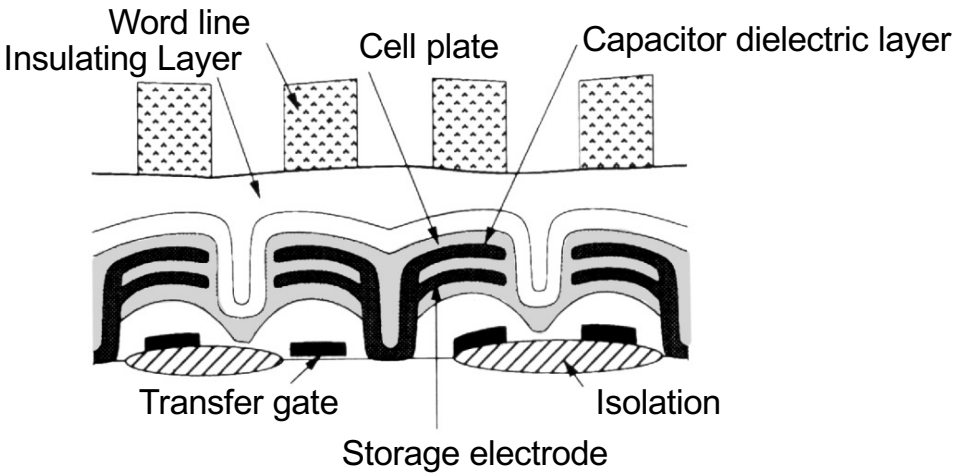


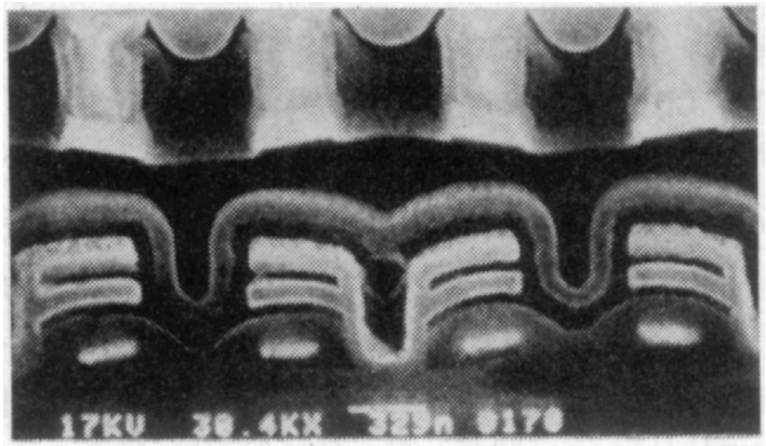
Fortgeschrittene 1T DRAM Zellen

- Ziel ist die Erzeugung **großer Kapazitäten** auf **kleiner Fläche** ⇒ 'Gräben' oder 'Pilze,
- Diese effizienten Tricks erfordern spezielle Technologien! DRAM in 'normalem' CMOS ist daher schwierig!



'Trench Cell'



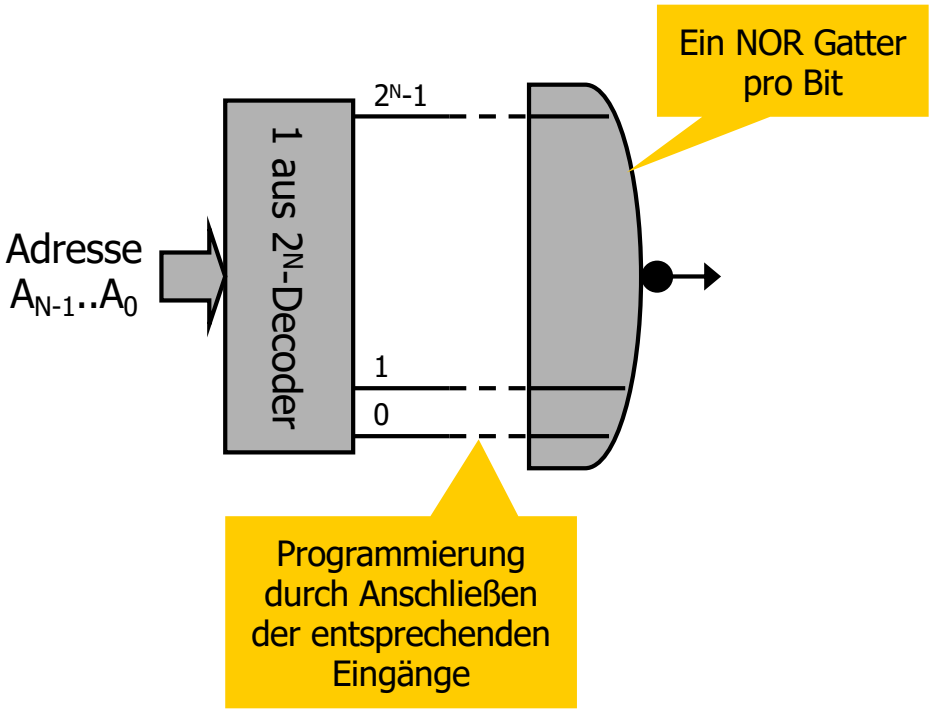


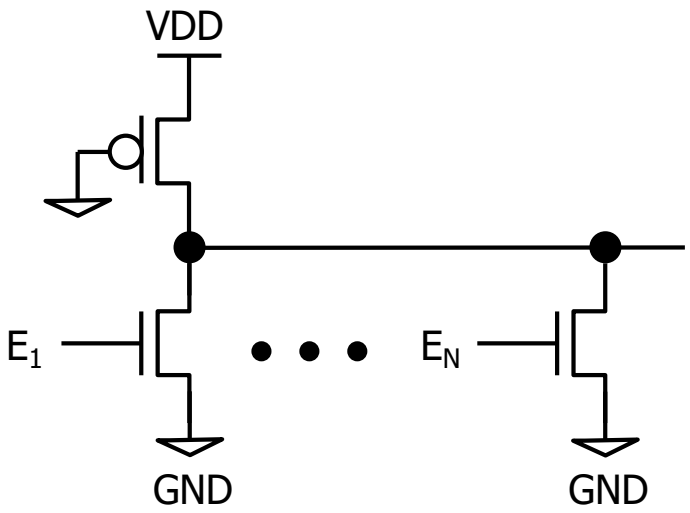
'Stacked-capacitor Cell'

ROM

NOR - ROM

- Basiert auf NOR – Gattern mit maximal 2^N Eingängen
- Ein Decoder liefert 2^N Steuerleitungen, die normalerweise **Null** sind
- Das Muster (der gespeicherte Wert) für jedes Datenbit wird definiert, indem die entsprechenden Steuerleitungen ans NOR angeschlossen werden (bzw. indem die MOS *weggelassen* werden)

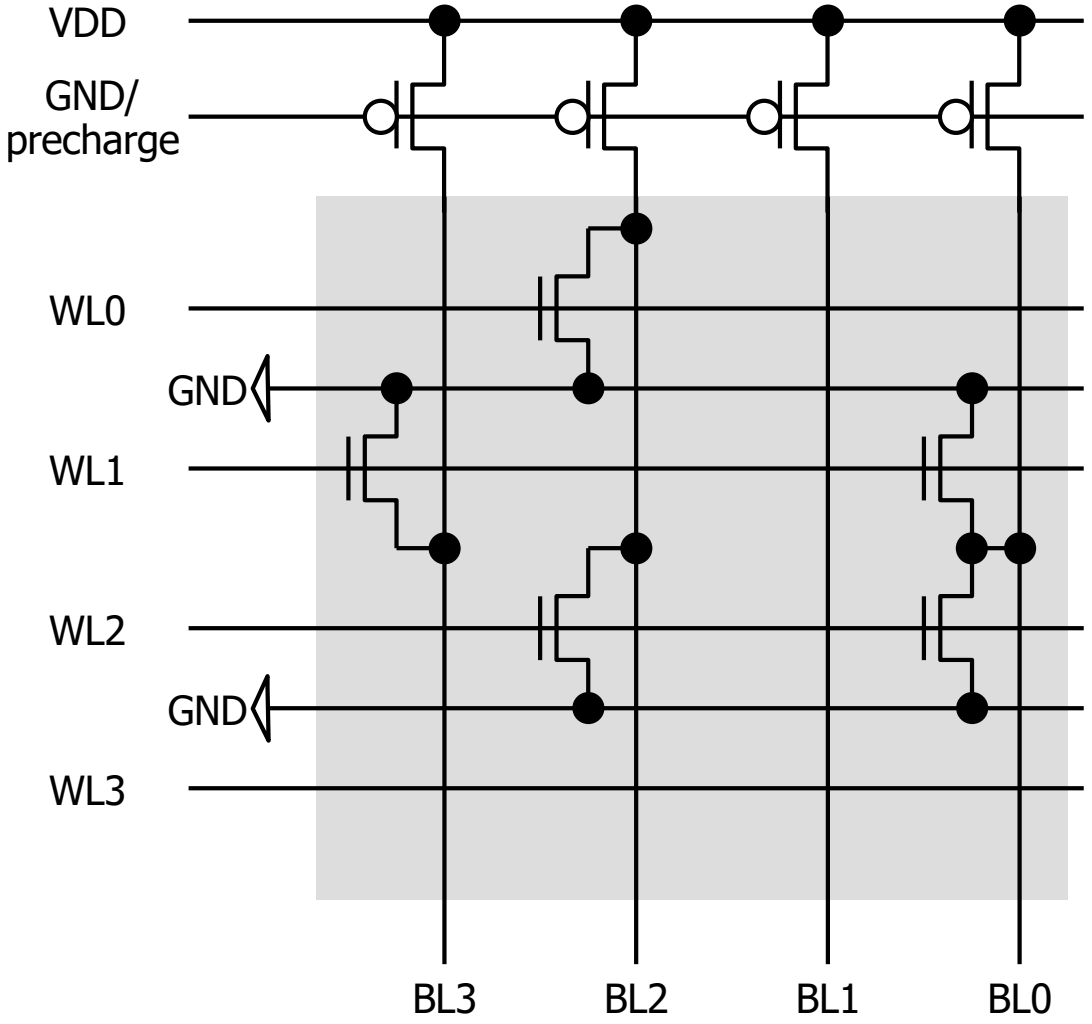




(P-Last) **NOR**

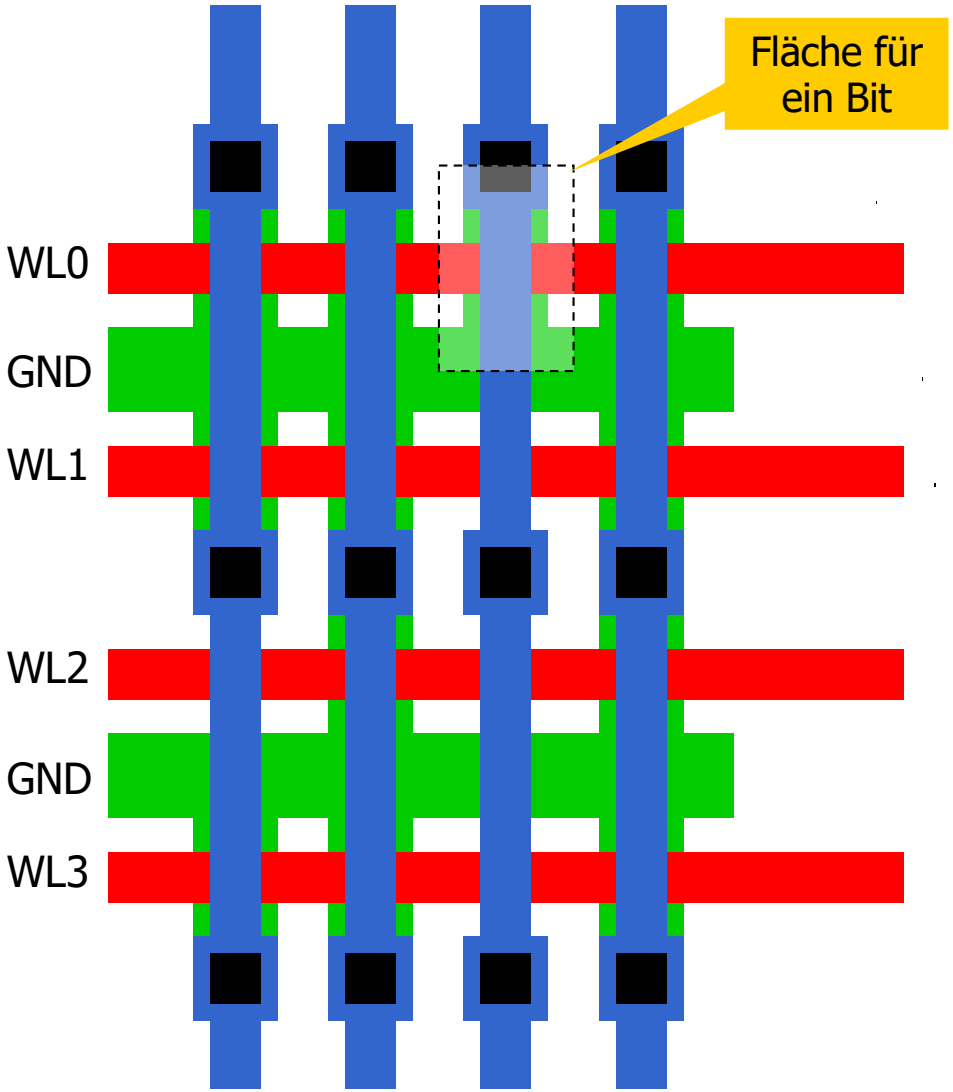
NOR ROM

- NOR mit einem Eingang pro Wort-Leitung
- Zur Vereinfachung der Matrix:
  - Pseudo-NMOS (statische PMOS Last)
  - Precharge Logik
    (schneller, keine statische Power)
- In beiden Fällen hat man im Array nur NMOS Transistoren
  $\Rightarrow$ sehr kompaktes Layout!
- NMOS sind parallel geschaltet



Adresse	WL<3:0>	Daten BL<3:0>
0	0001	1**0**11
1	0010	**0**11**0**
2	0100	1**0**1**0**
3	1000	1111

NOR ROM Layout



Programmierung durch Hinzufügen/Entfernen der Active-Lage

(andere Varianten möglich!)

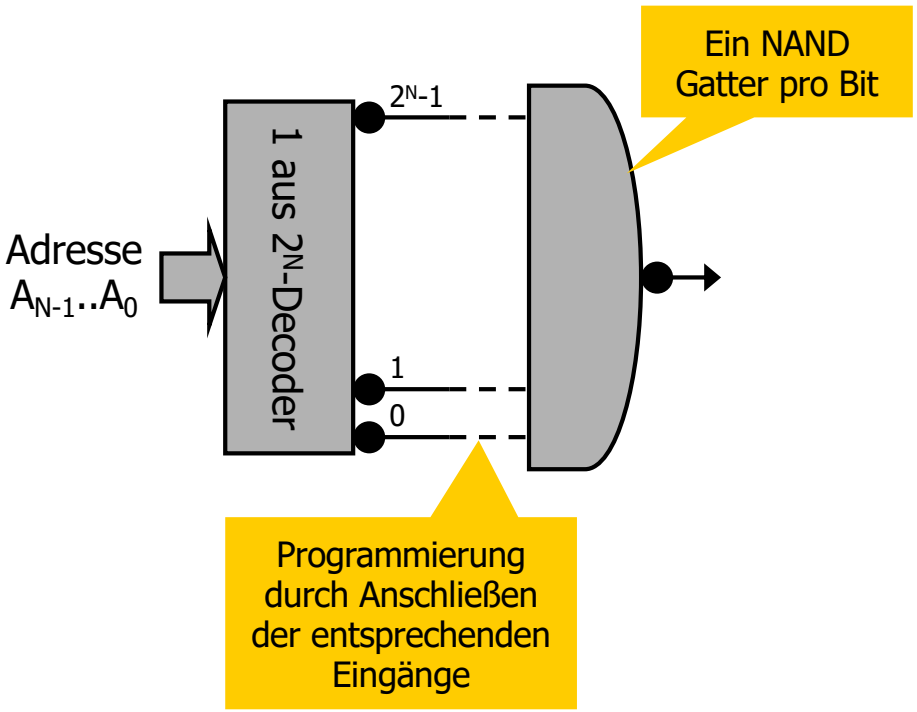
Polysilicon

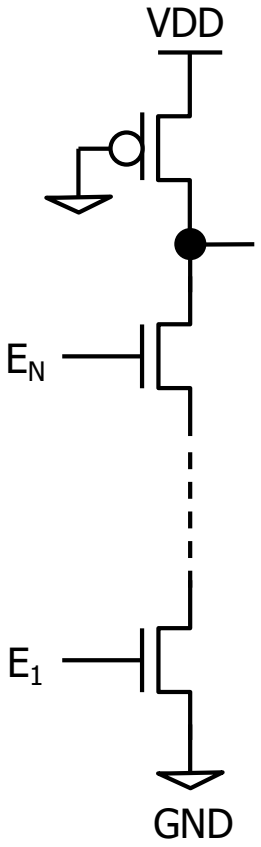
Metal1

Diffusion

NAND - ROM

- Basiert auf NAND – Gattern mit maximal 2^N Eingängen
- Ein Decoder liefert 2^N Steuerleitungen, die normalerweise **Eins** sind
- Das Muster (der gespeicherte Wert) für jedes Datenbit wird definiert, indem die entsprechenden Steuerleitungen ans NAND angeschlossen werden (bzw. indem die MOS *kurzgeschlossen* werden)

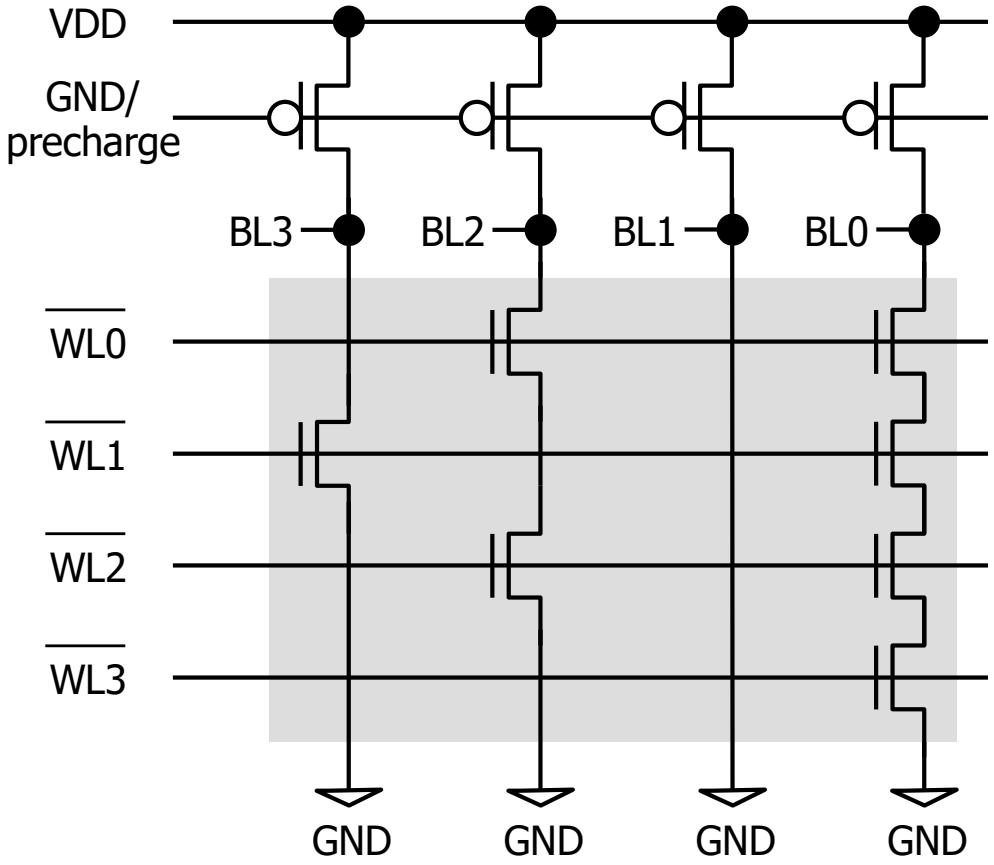




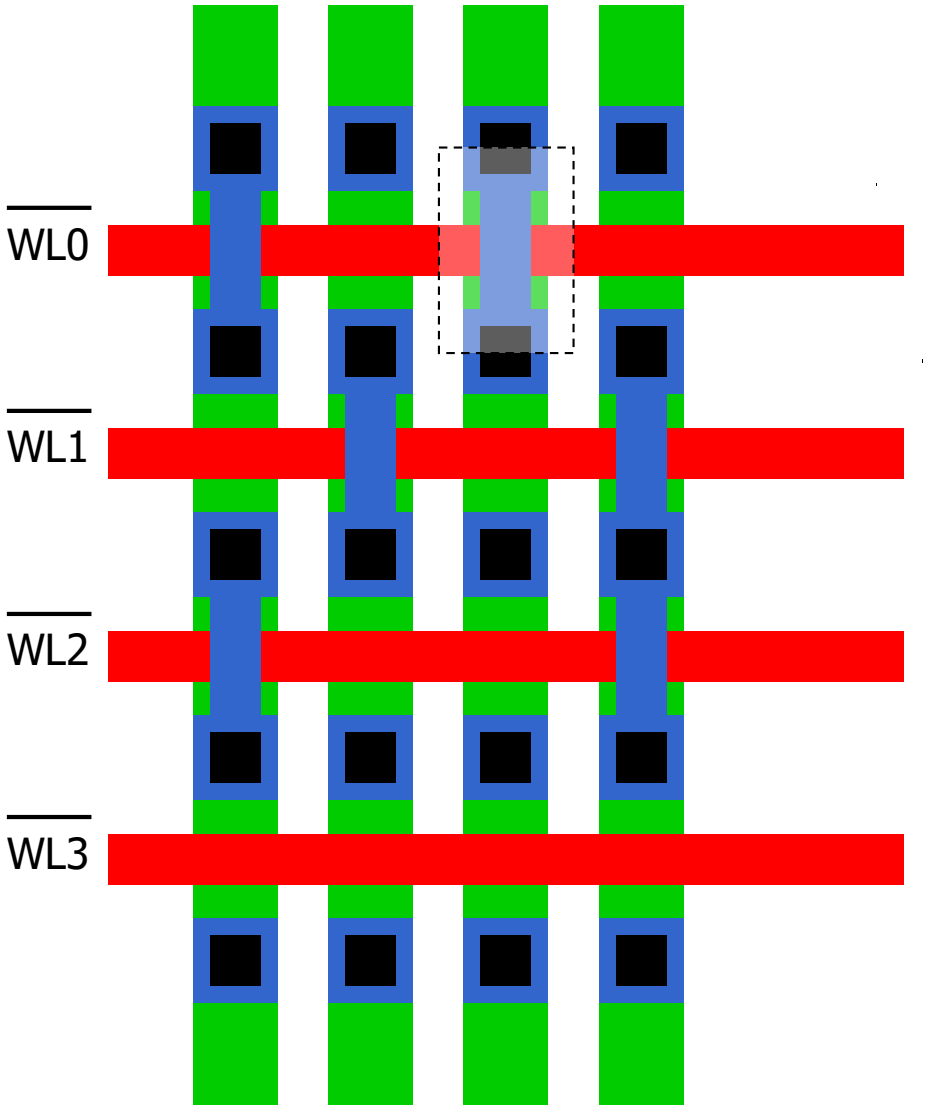
(P-Last) **NAND**

NAND ROM

- NMOS hier in Serie. Wort-Leitungen sind normalerweise 1, nur die selektierte Zeile ist Null
- Weniger Kontakte nötig, keine Masse in der Matrix, daher kleinere Zelle
- Aber *langsamer* wegen der *Serienschaltung* der MOS



NAND ROM Layout



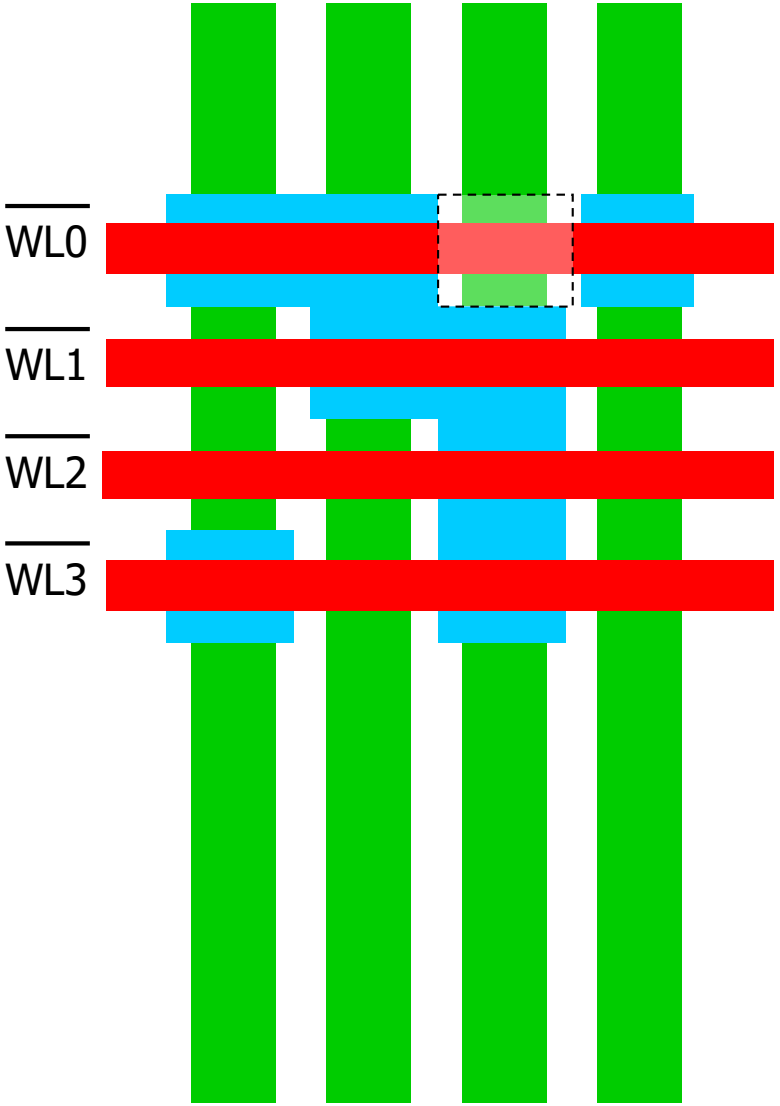
Programmierung durch Hinzufügen/Entfernen der Metallkurzschlüsse

Polysilicon

Metal1

Diffusion

Alternatives NAND ROM Layout



Programmierung
durch eine Implantation,
die die Schwelle absenkt

Sehr kompakt!

Polysilicon

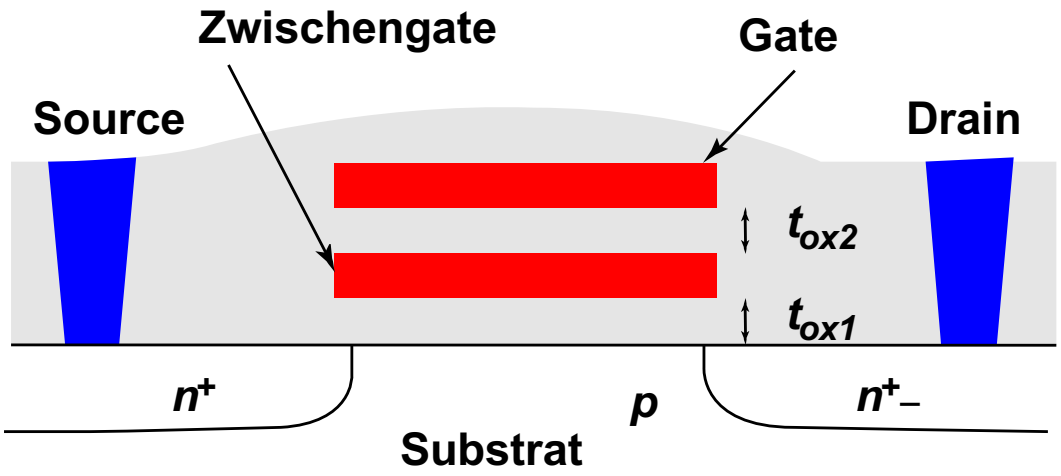
Implantation zur
Verschiebung der Schwelle

Diffusion

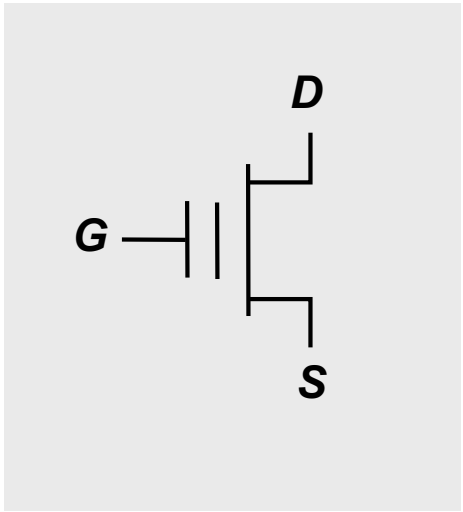
EPROM / EEPROM
Flash-Memory

Grundelement: Der Floating-Gate Transistor

- Transistoren mit einem **Zwischengate,** auf dem Ladung gespeichert werden kann
- Es gibt **keine Verbindung** vom Zwischengate nach außen ⇒ Ladung bliebt sehr lange gespeichert
- Ladung auf dem Zwischengate verschiebt die Schwellenspannung des Transistors
- Ladung auf dem Zwischengate wird ‚mit Gewalt', z.B. mit hohen Spannungen verändert
- Das Schreiben/Löschen des Zwischengates belastet das Oxid, daher ist die Anzahl Zyklen begrenzt

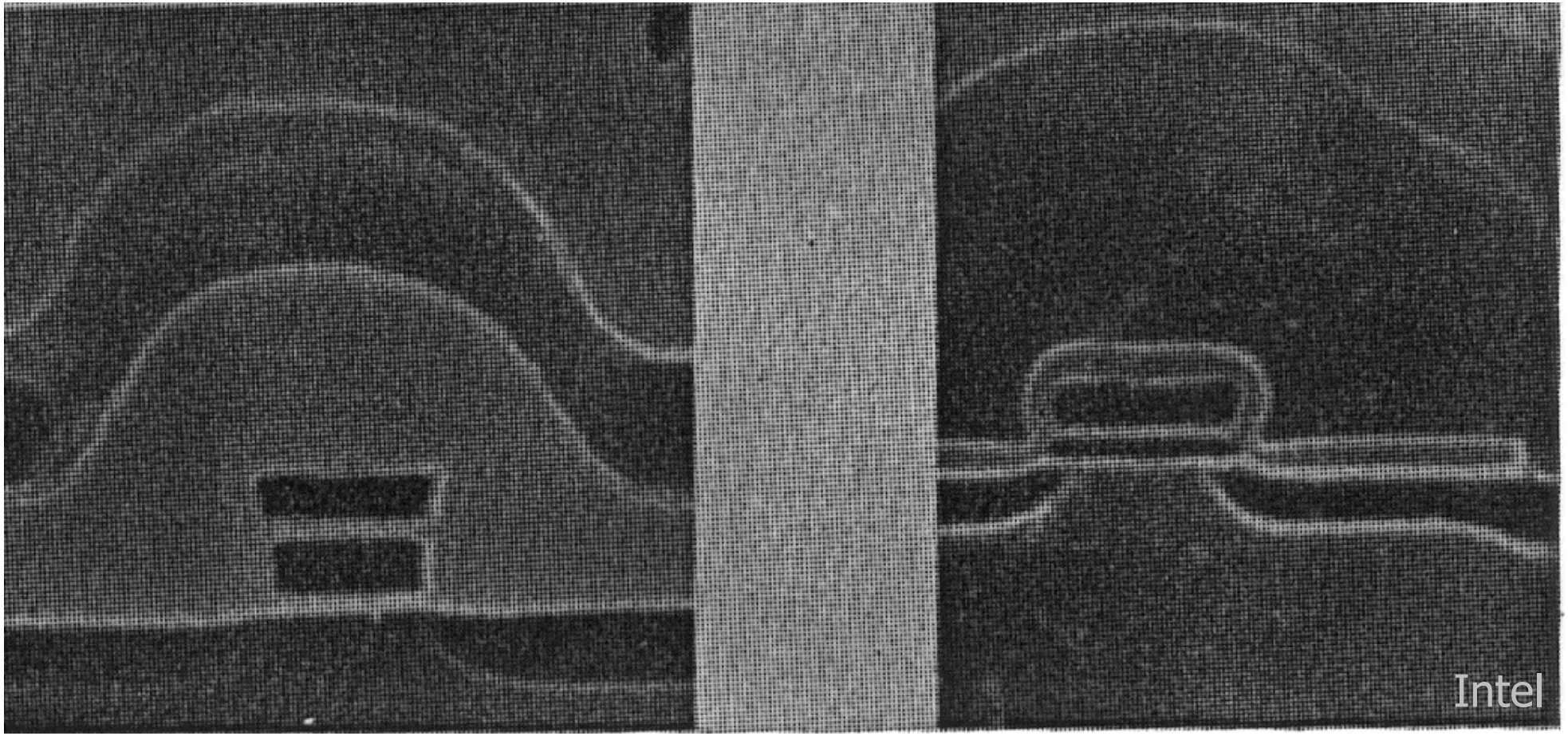


Querschnitt



Symbol

Querschnitt EPROM



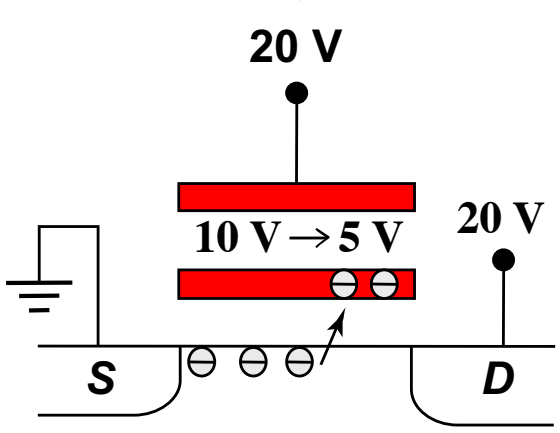
EPROM

Flash

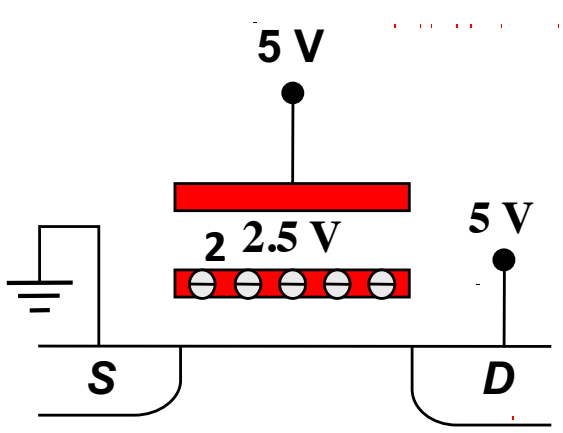
Programmierung des Floating-Gate Transistors

- Schreiben (bzw. Löschen) von Ladung auf das interne Gate durch unterschiedliche physikalische Mechanismen:

- Durch **UV-Licht** werden Elektron-Loch-Paare im Oxid erzeugt, die im Feld getrennt werden. Sie entladen/laden das Zwischengate (langsam, Chips mit UV-Glasfenster)
- **Fowler-Nordheim Tunneln**:
  Bei genügend hoher Spannung können Ladungsträger durch das Gate-Oxid tunneln.
- **Hot-Electron Injection**:
  Bei einem hohen Strom im Kanal können schnelle (‚heiße') Elektronen an der Drain-Seite einen Lawinendurchbruch herbeiführen. Die dabei entstehenden Ladungsträger können durch das Gateoxid zum Zwischengate gelangen

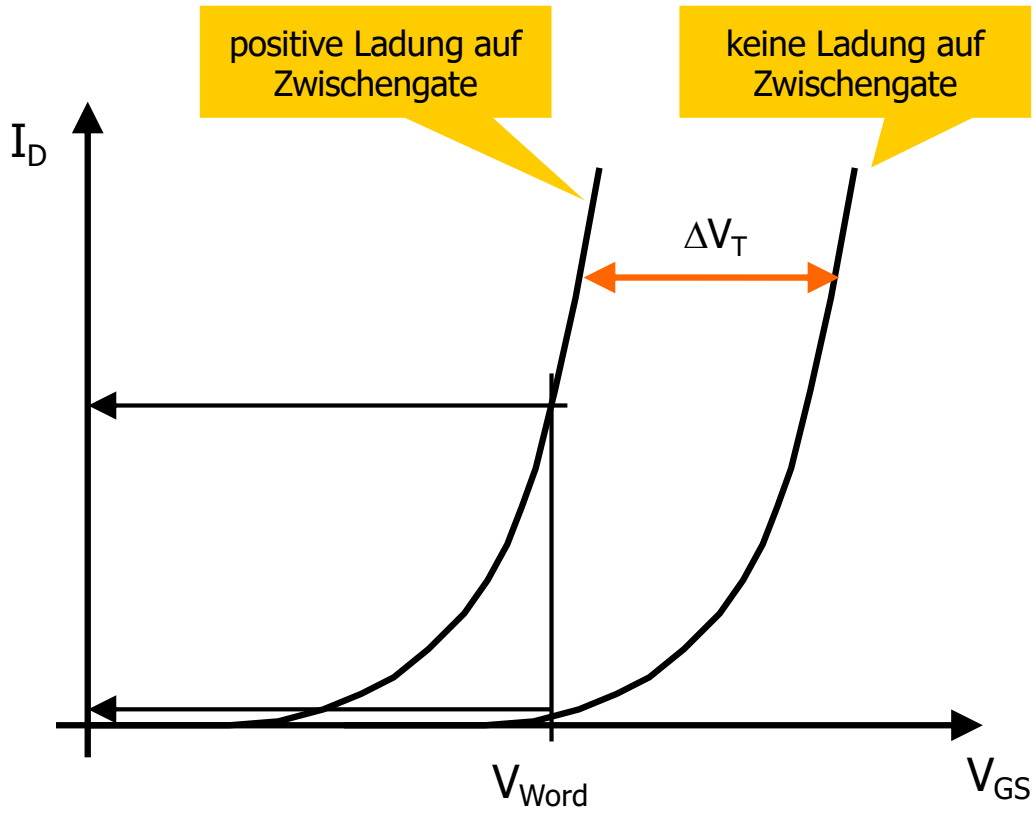


Lawineninjektion



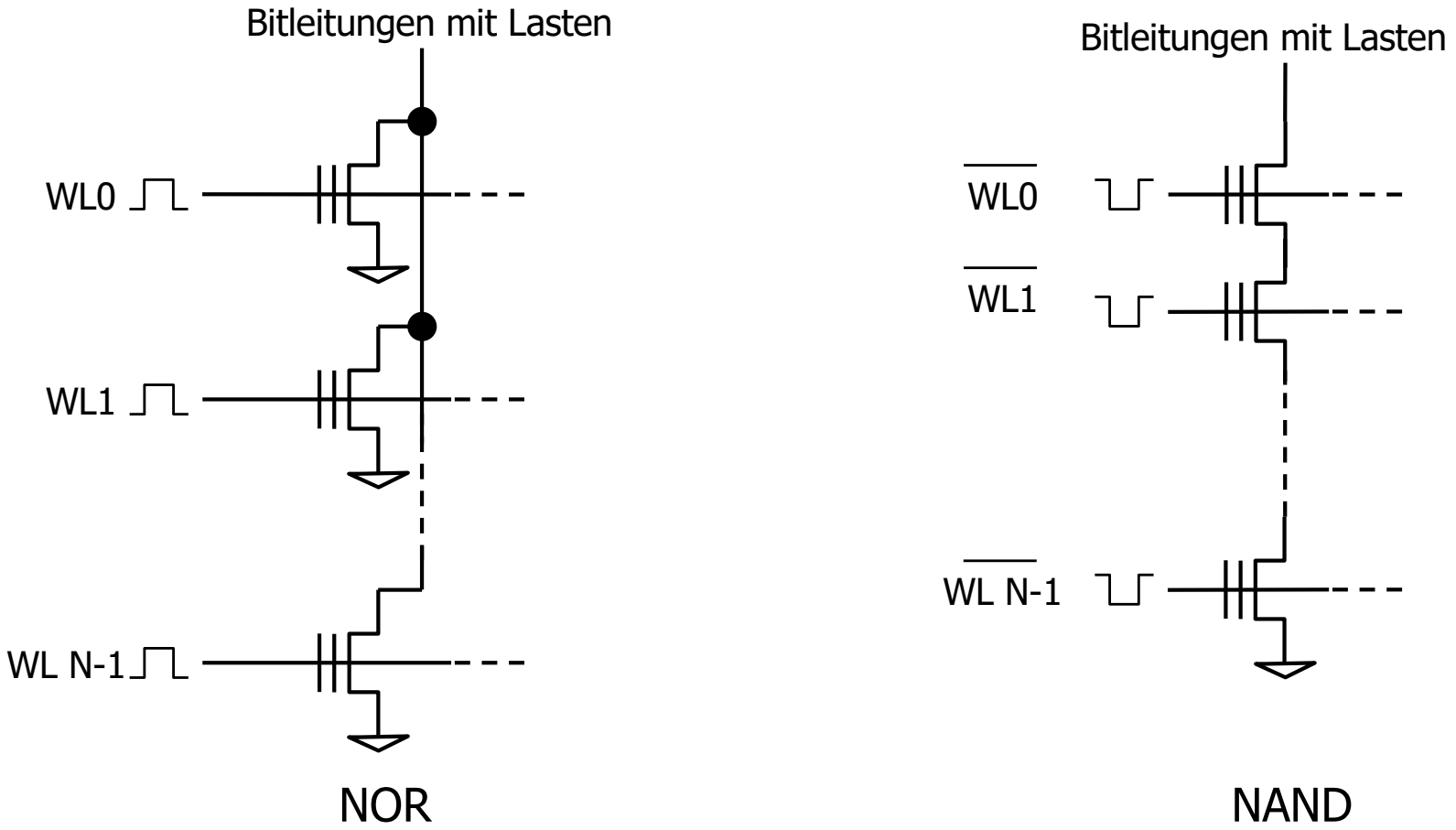
Durch Elektronen auf dem Zwischengate wird V_T höher

Übertragungskennlinie mit gespeicherter 1 oder 0



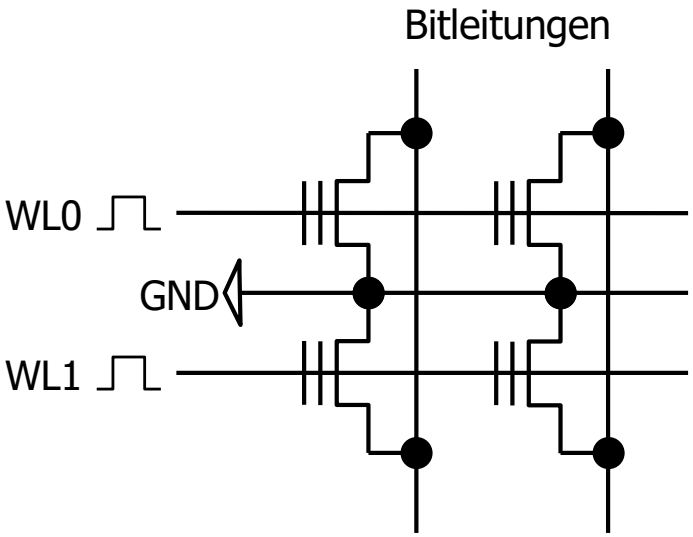
NOR und NAND Flash Memory

- Architektur wie ROM
- Datenmuster wir eingestellt über
  - NAND: ‚Kurzschließen' von MOS durch niedrige Schwellen
  - NOR: ‚Entfernen' von MOS durch hohe Schwellen

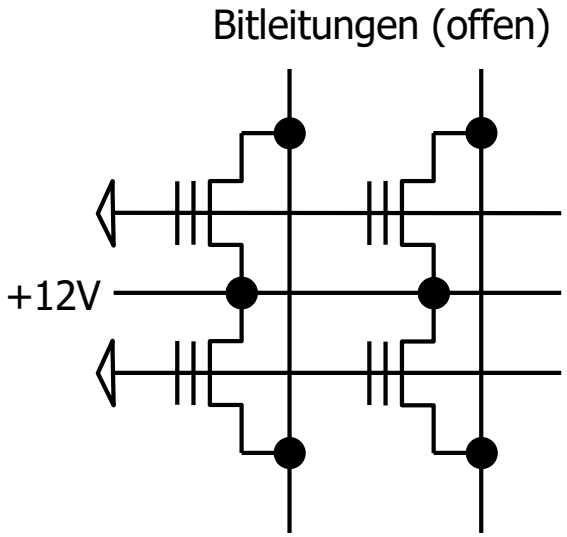


Löschen

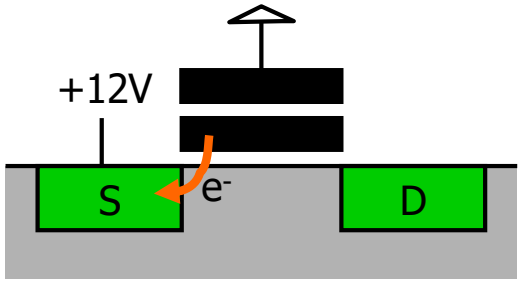
- Ladungen müssen in **2 Richtungen** auf die Floating Gates gebracht werden
- Dies ist für jedes Bit **einzeln nicht** möglich (bei kompakter Architektur)
- Lösung: Eine Richtung für **alle** Zellen, die andere Richtung für **adressierte** Zellen



Betrieb



Globales Löschen
(hier: Elektronen entfernen)



- Das Schreiben / Löschen schädigt mit der Zeit das Gate Oxid, so dass nur eine begrenzte Anzahl Schreibzyklen (viele 1000) möglich ist. (Lesen unbegrenzt!).

Multi-Level - Zellen

- Durch sorgfältiges Schreiben und Lesen kann man mehr als 2 Werte (0/1) in einer Zelle speichern und so mehr also nur 1 Bit unterbringen!
- Inzwischen bis zu 4 Bit pro Zelle! (‚MirrorBit Quad': http://www.heise.de/newsticker/meldung/78683): Dies erfordert die Erkennung von 16 (!) unterschiedlichen Levels!

- SLC = Single-Level-Cell, MLC = Multi-Level-Cell, ...

- Flash Speicher wird in Solid State Disks (SSD) genutzt.

- Um eine schnelle Abnutzung viel benutzter Blöcke (Directory – 'tracks') zu vermeiden, sorgen spezielle Betriebssysteme für eine gleichmäßige Auslastung aller Zellen ('wear leveling')

- Speicherung ANALOGER Werte ebenfalls möglich
  - Offset Trim in Verstärkern
  - Gewichtsfaktoren in Neuronalen Netzen (KIP)