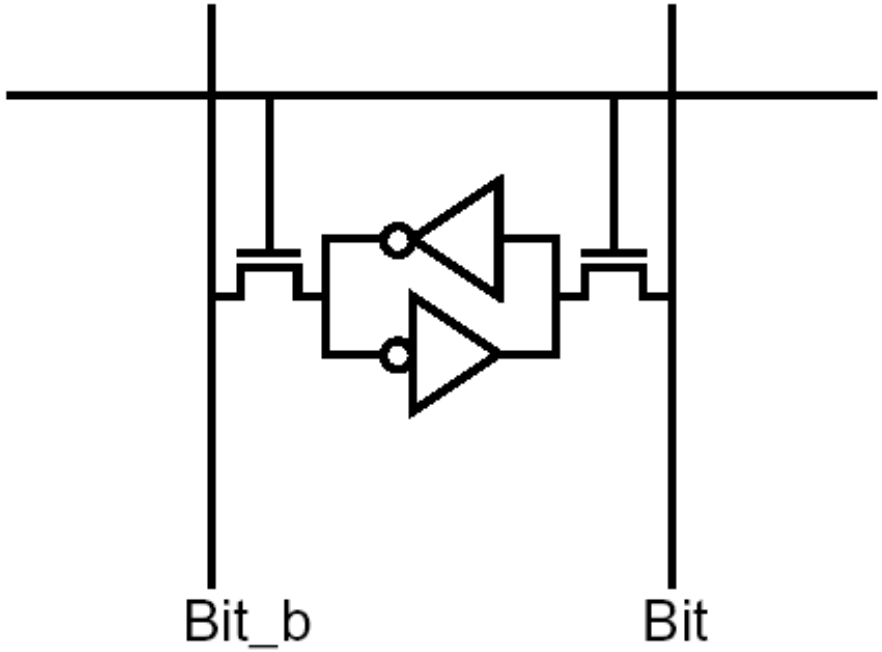
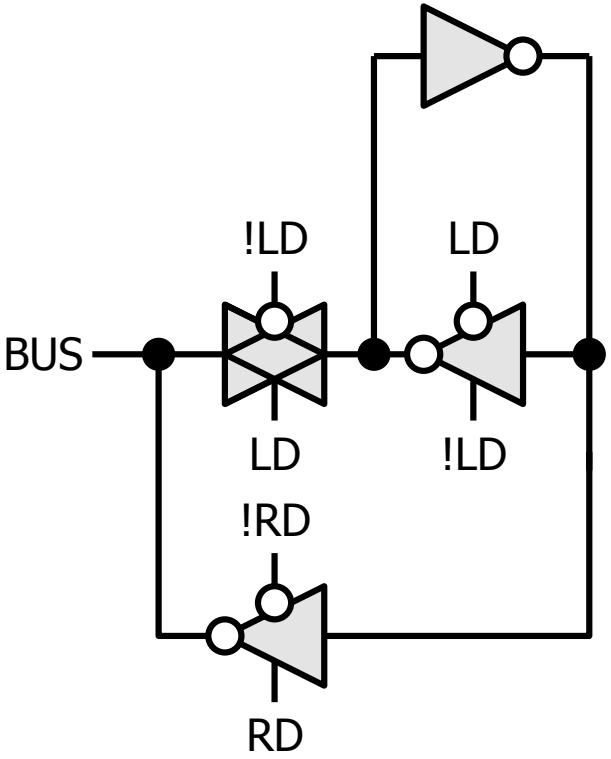
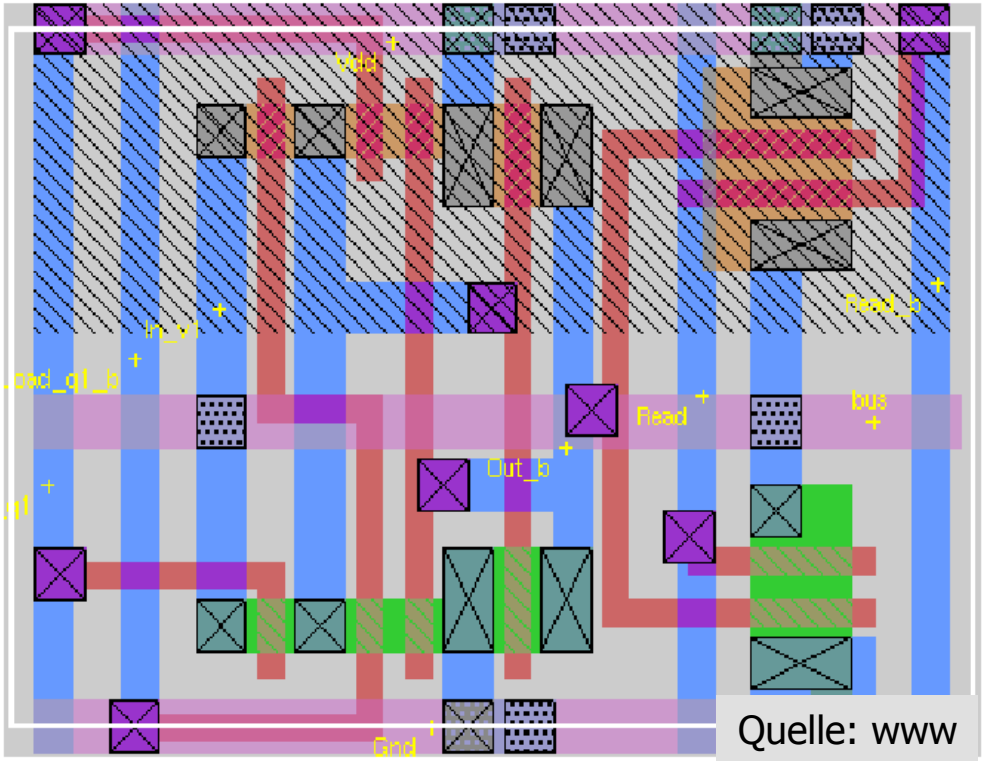
SRAM



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



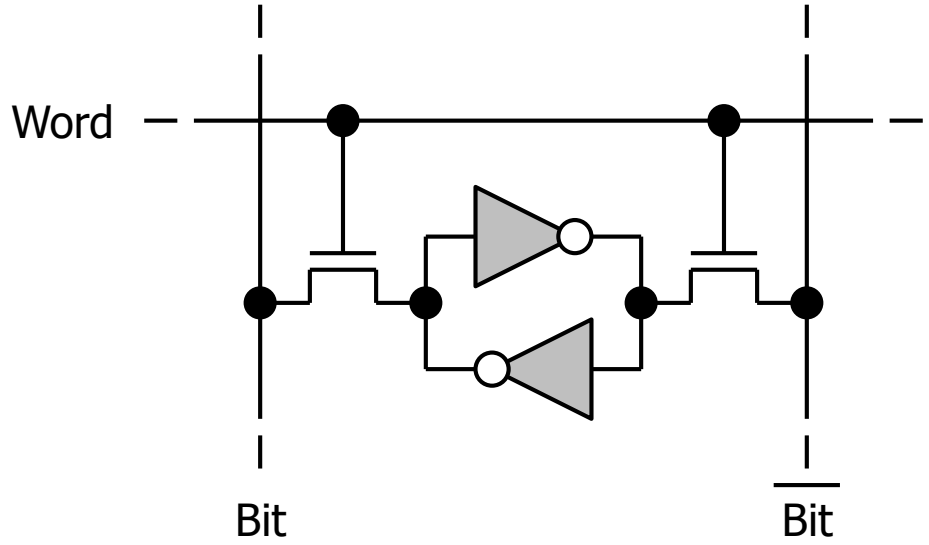


Quelle: www

- 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)



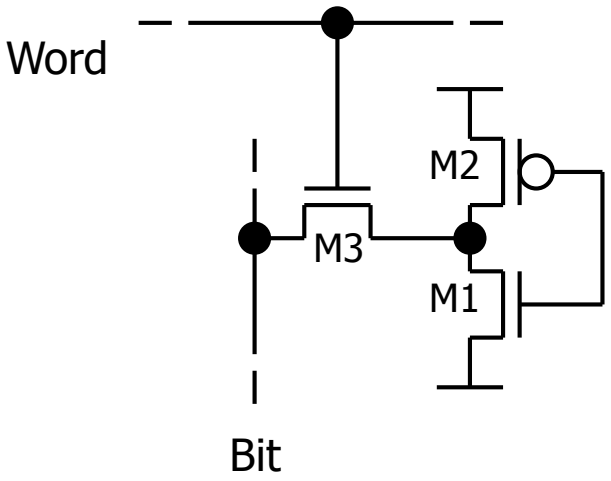
- Zum **Schreiben** werden Bit und !Bit auf 1/0 oder 0/1 gehalten und Word aktiviert
- Beim **Lesen** darf die Zelle nicht umkippen, wenn sie an den Bit/!Bit Bus geschaltet wird.
  Die Bit-Leitungen werden daher z.B. auf VDD vorgeladen (**'precharge'**)

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!

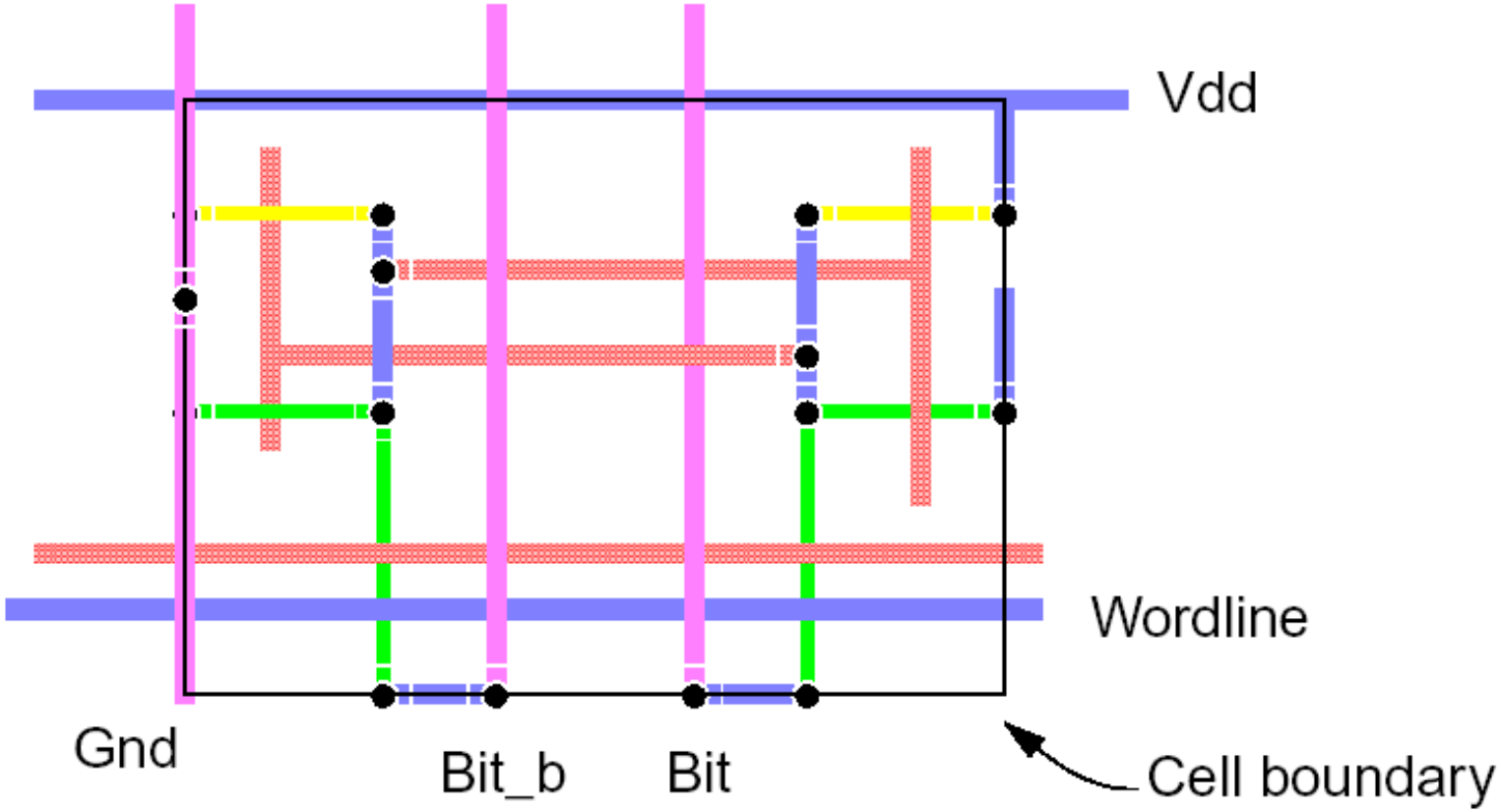
Dimensionierung der Transistoren (sehr verkürzt):

- 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: $\mathbf{(W/L)_{M1}=1.5..2\ x\ (W/L)_{M3}}$



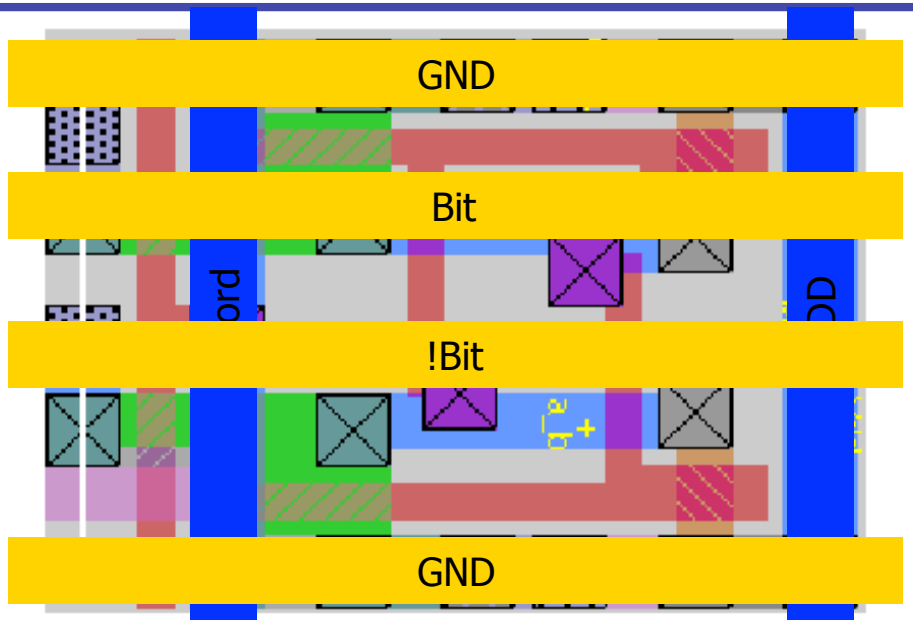
Stick Diagramm 6T SRAM

- Ein mögliches Layout der 6T Zelle
- Die 5 Leitungen (Word, bit, !bit, gnd, vdd) werden gleichmäßig horizontal/vertikal verteilt
- Sie werden in benachbarten Zellen mitbenutzt (Layout der Nachbarzellen horizontal/vertikal gespiegelt).

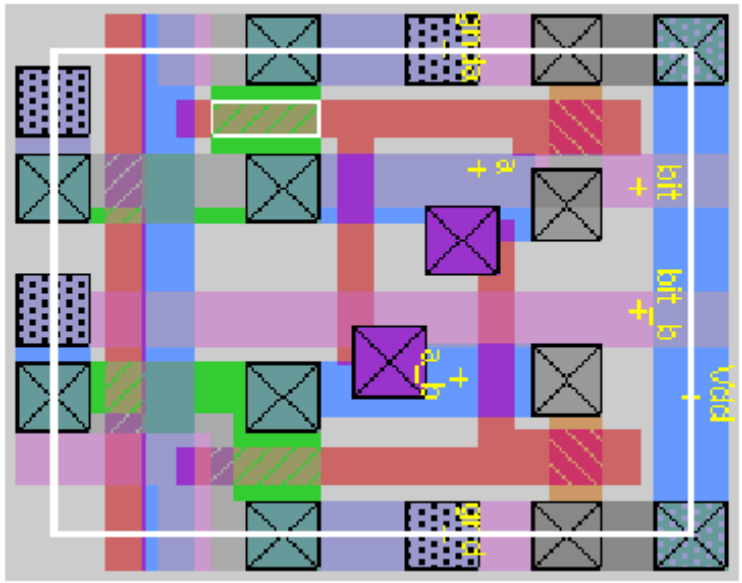


Layouts 6T SRAM

- '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)



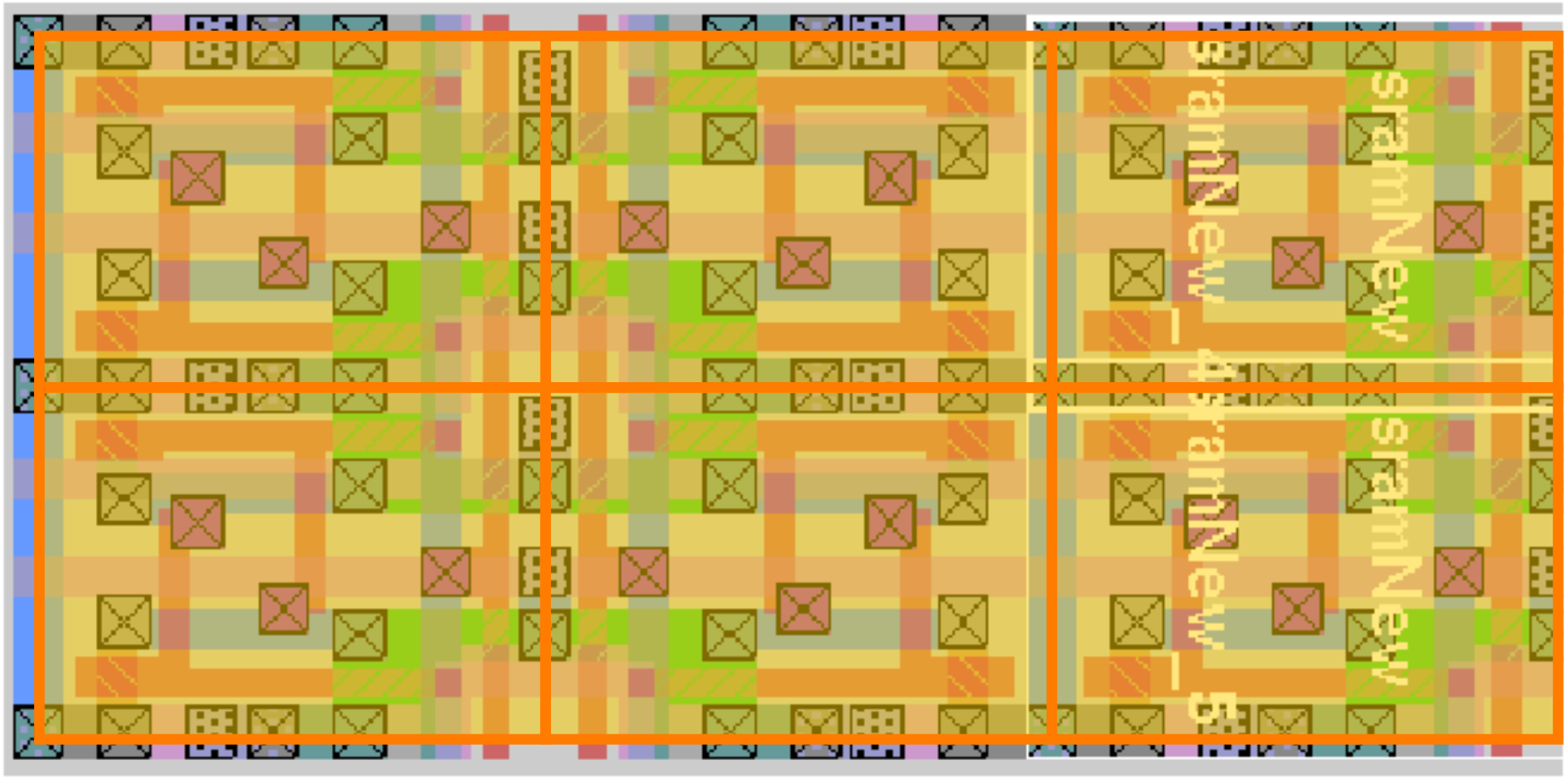
- Etwas kleiner:
  - Nur nwell Kontakt
    (Wannenkontakt wird nach jeweils N Zellen eingefügt)
  - NMOS pulldown ist etwas schwächer (6:2)
  - 36 x 28 Einheiten



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

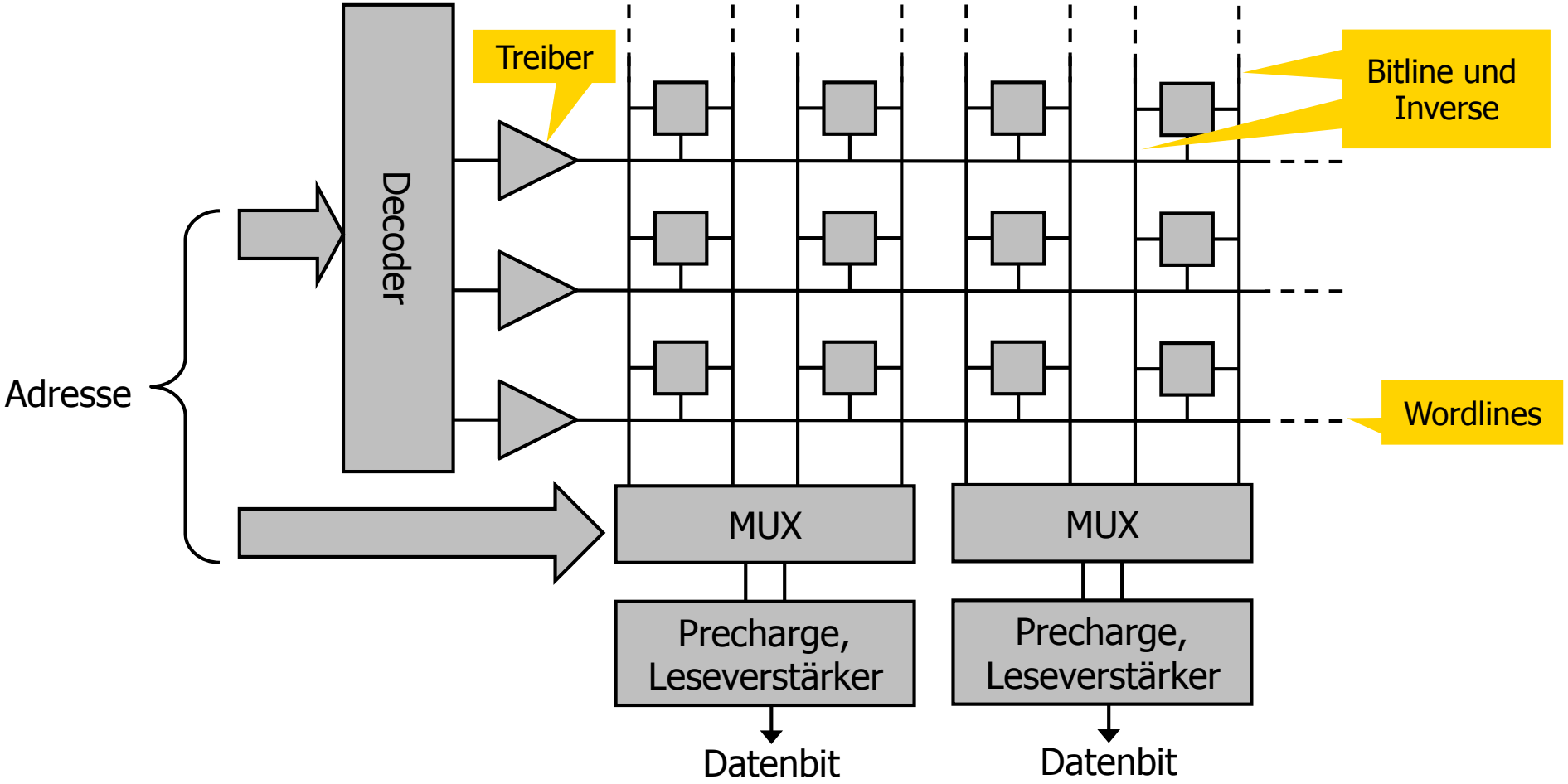
SRAM Matrix

- 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



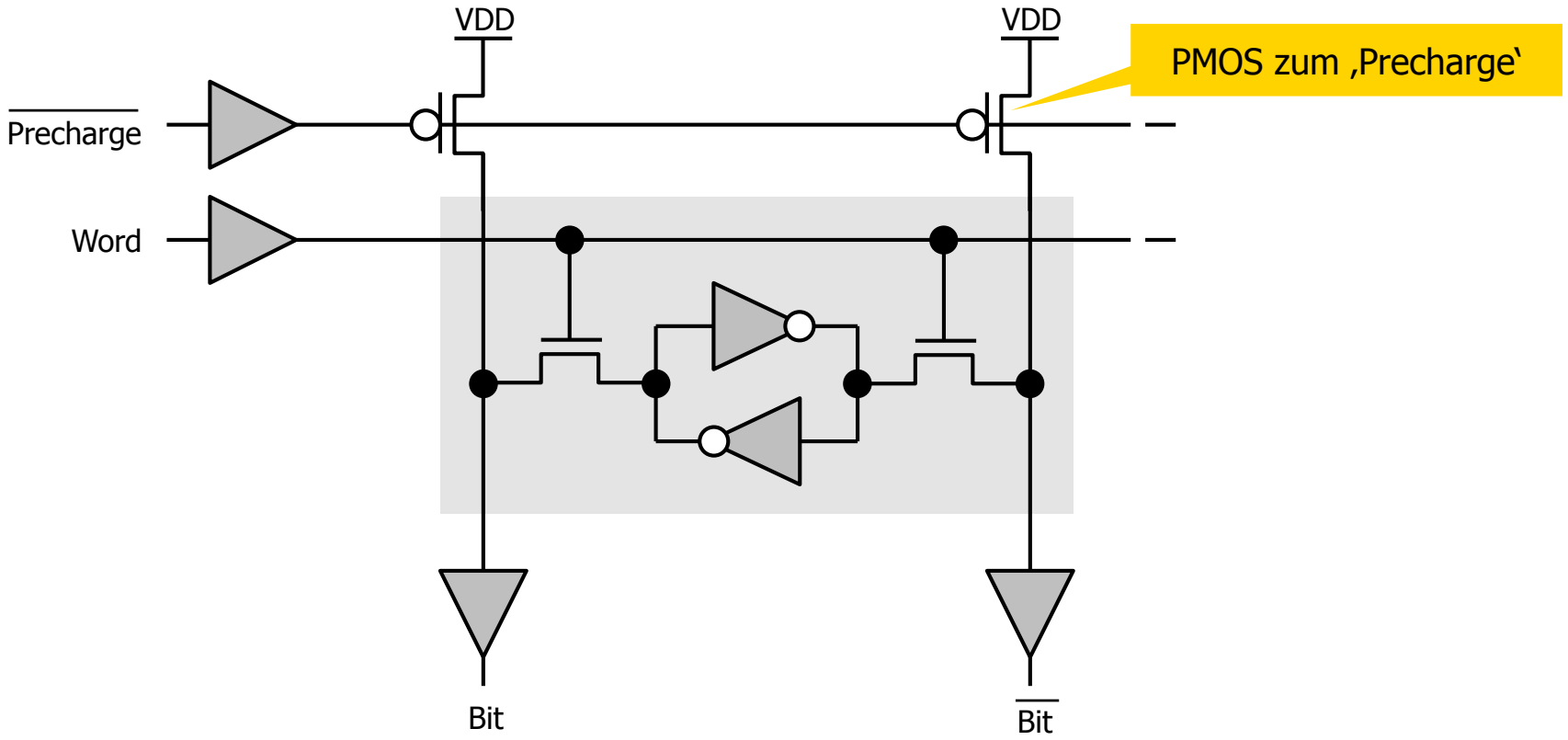
Betrieb der Matrix

- 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')



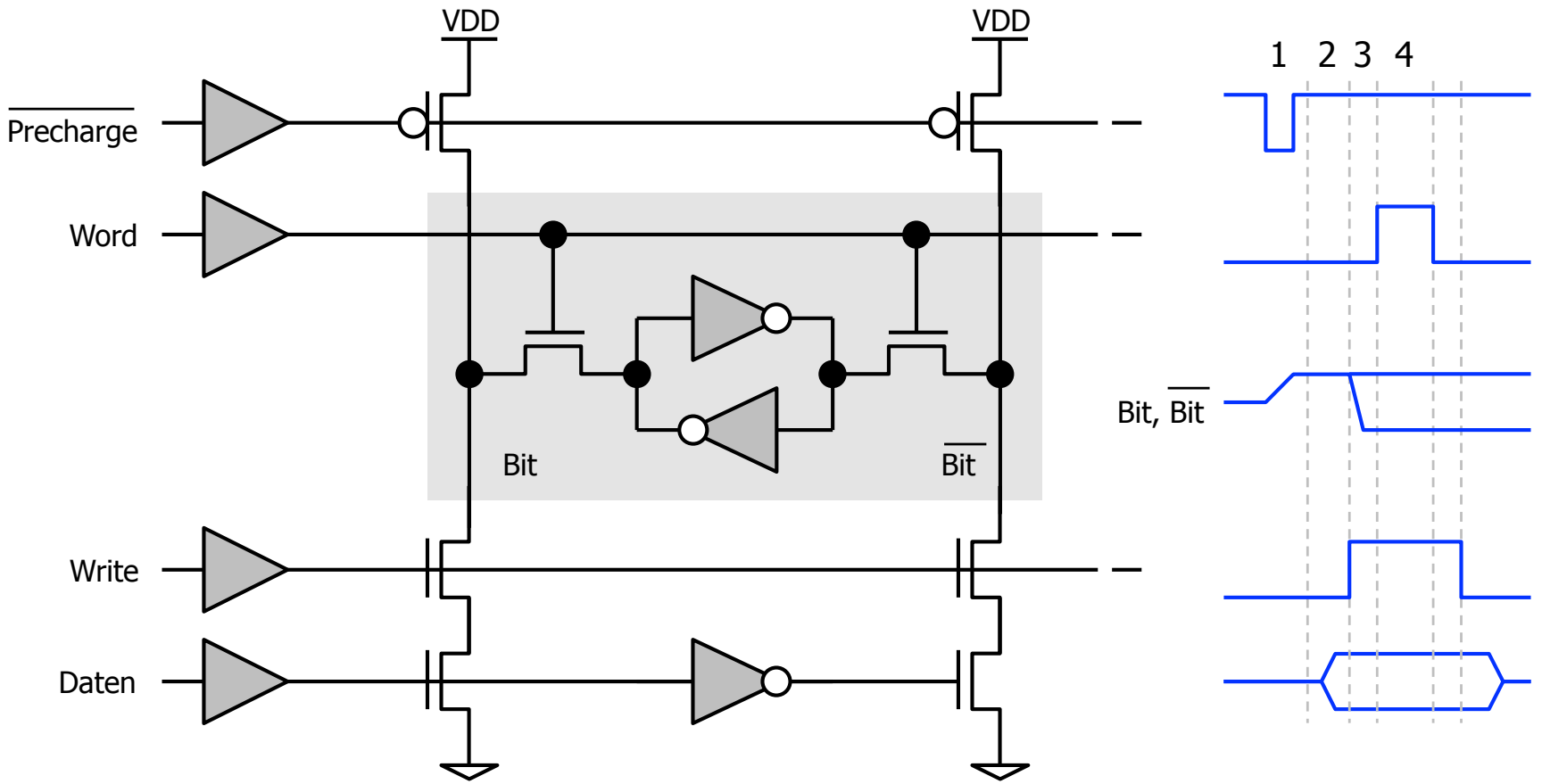
Lesen der 6T-Zelle

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!



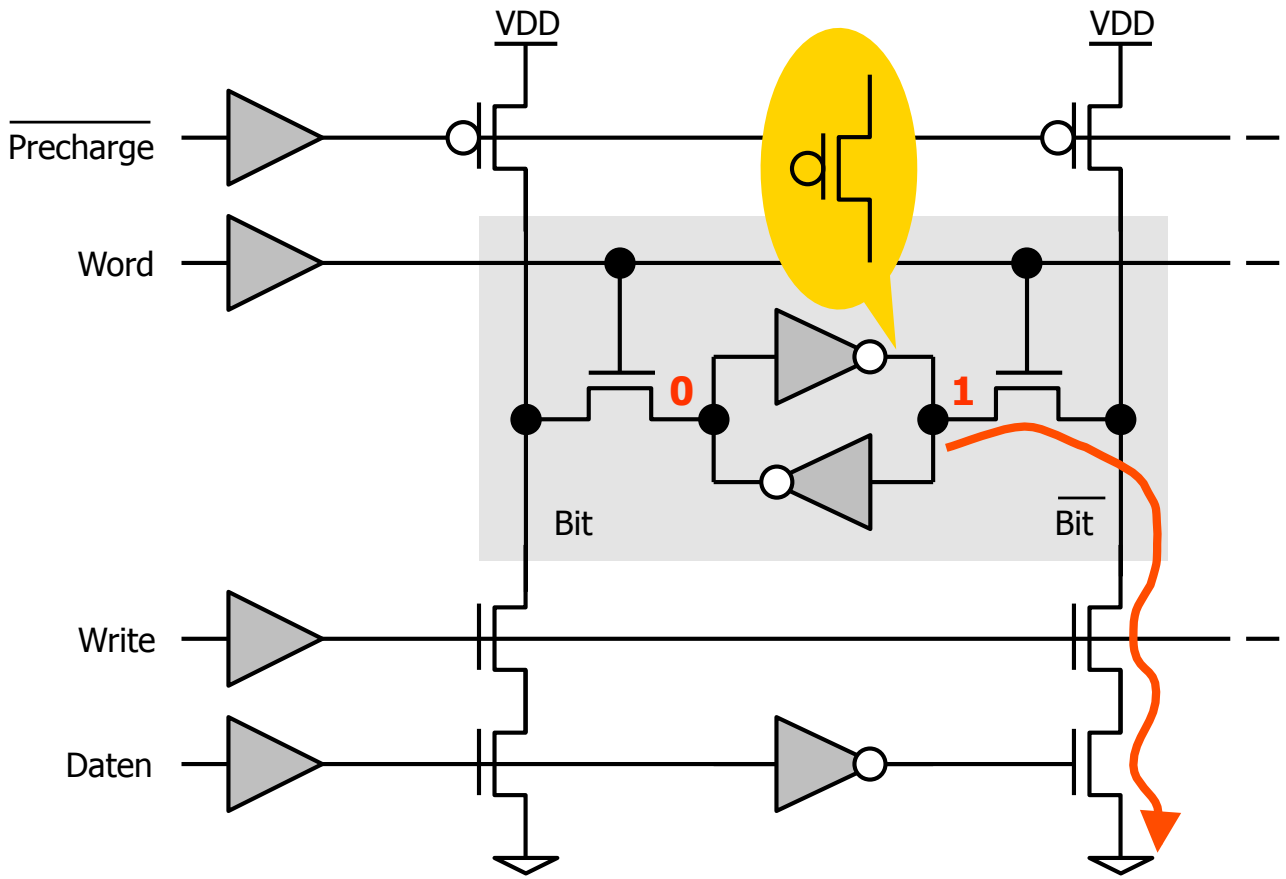
Schreiben der 6T-Zelle (Beispiel)

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.



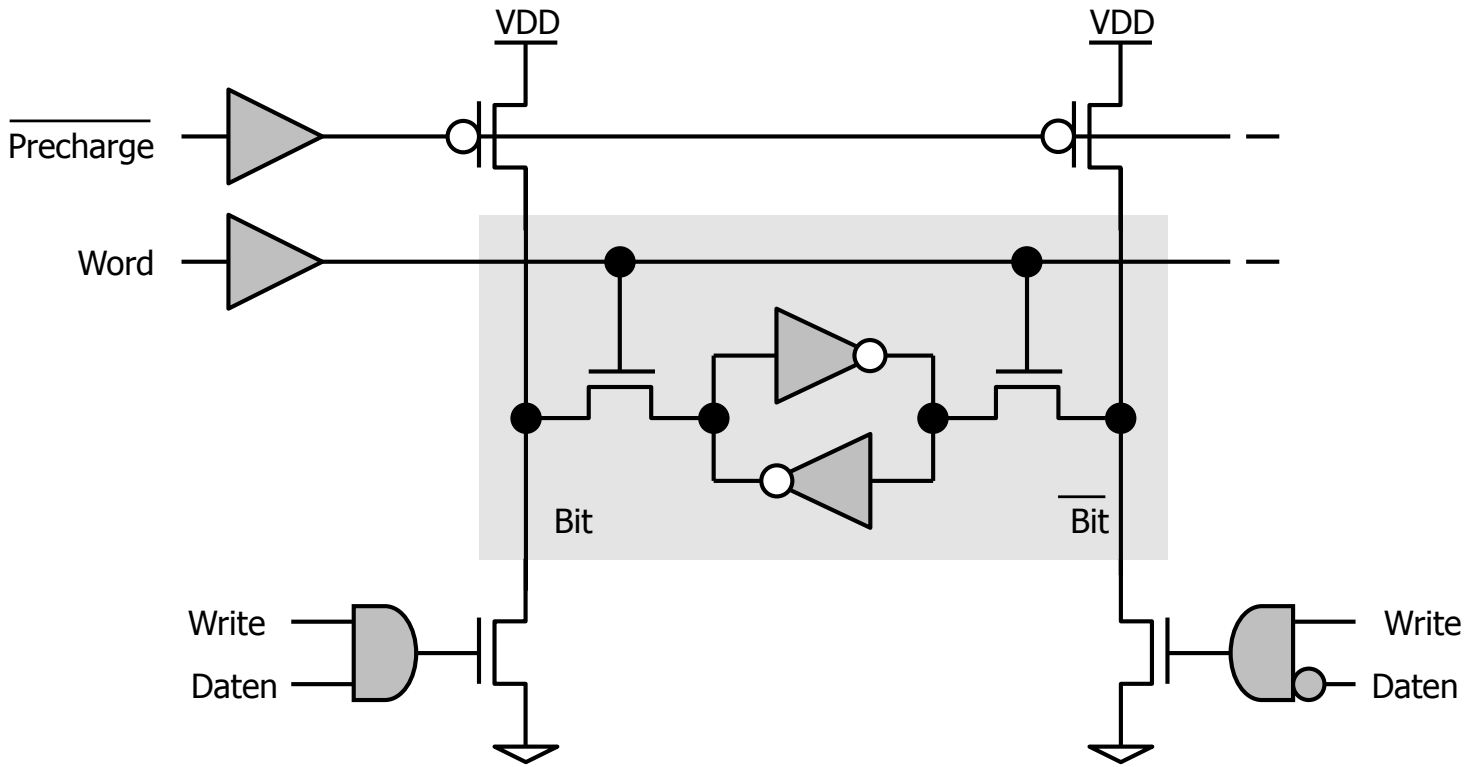
Write - Treiber

- 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

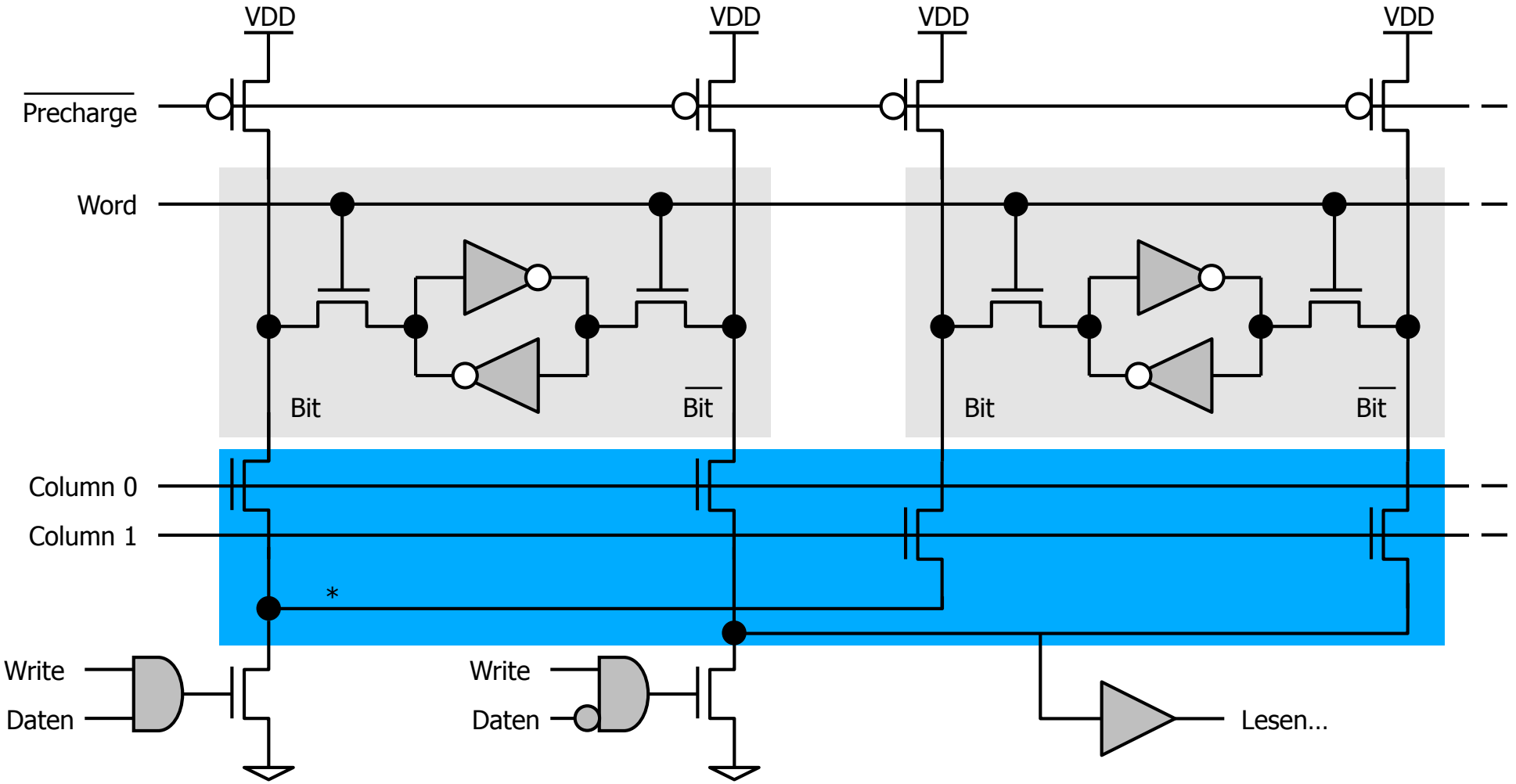
- Hier treibt nur ***ein*** NMOS nach Masse



- (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)

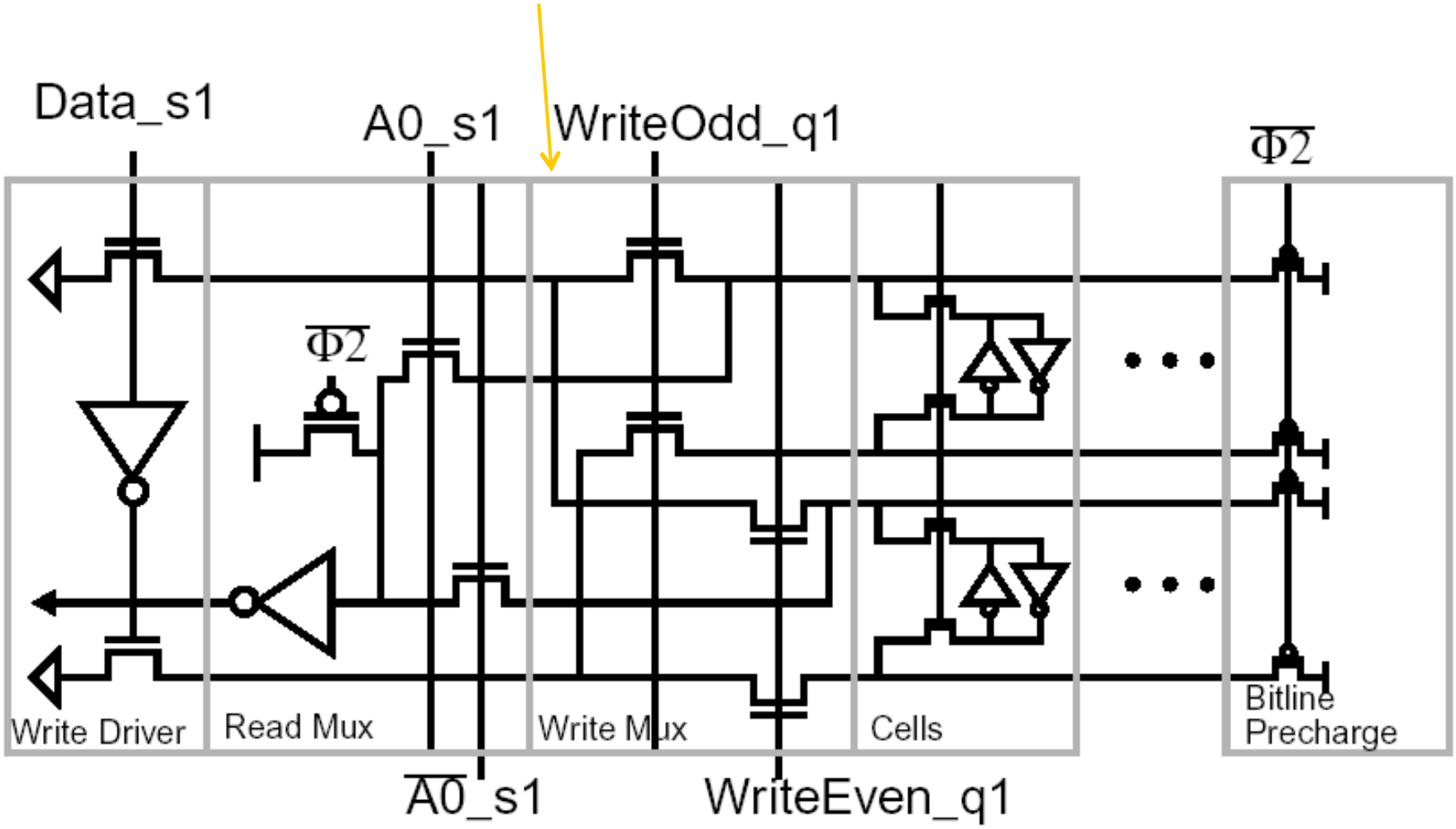
Bitline Multiplexing

- 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.



Bitline Multiplexing - Variante

- Getrennte MUX für Schreiben und Lesen
- Lese-MUX hat Precharge
- Nur ein NMOS für WRITE_ENABLE und für MUX



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:

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

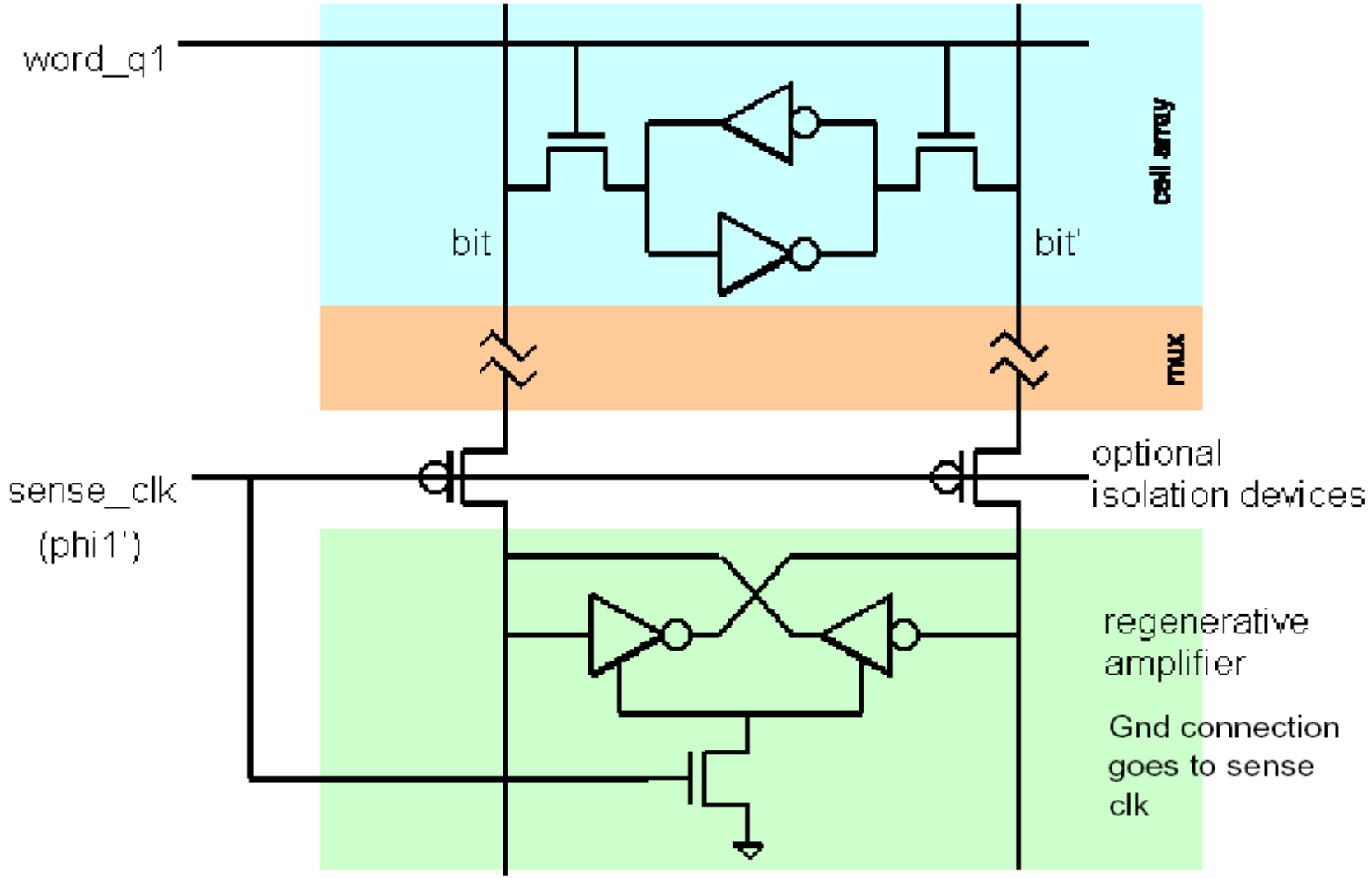
gegeben

- 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

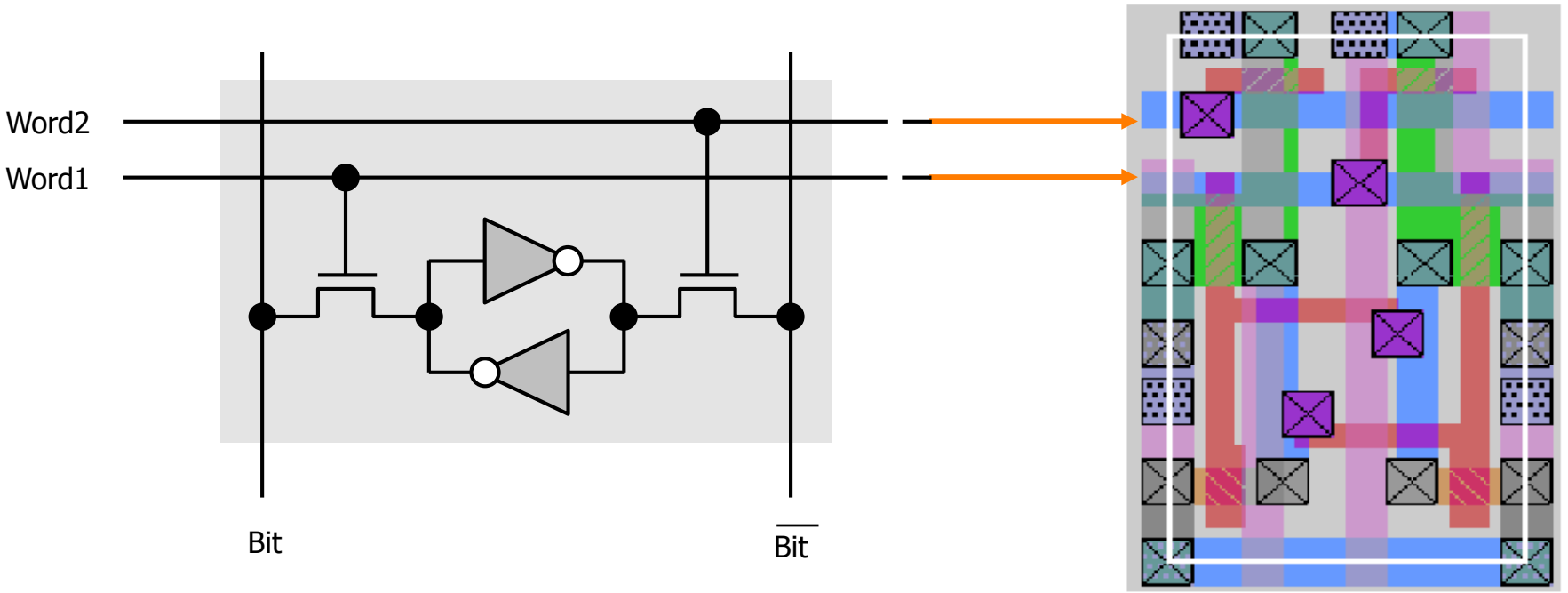
Leseverstärker

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



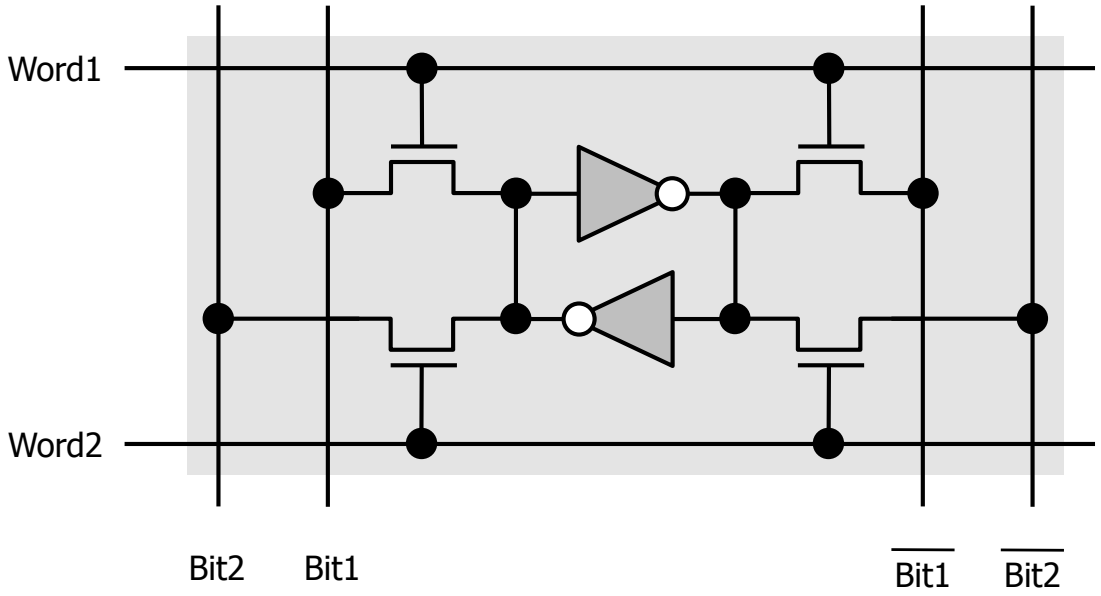
Dual Ported RAM

- Zwei Wort-Leitungen (eine pro Pass-Transistor)
- Zelle nur geringfügig größer
- Man kann jetzt **zwei Zeilen gleichzeitig lesen**:
  - Die Zeile, die mit Word1 aktiviert ist, gibt Daten auf Bit_b aus,
  - Die andere Zeile gibt Daten auf Bit aus.
- Beim **Schreiben** benötigt man immer noch beide Bitleitungen, kann also nur 1 Zeile schreiben



Multiported RAM

- Durch Hinzufügen von Bitleitungen bekommt man echt unabhängigen Schreib/Lese-Zugriff
- Hier: 'Echtes' Dual ported RAM
- Mehr Bit-Leitungen, daher größeres Layout



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.