Digitaltechnik

6 Speicherelemente

Revision 1.4

Übersicht

Adressen

Read-Only Memory – ROM

Random Access Memory – RAM

Datenbusse

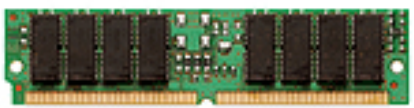
Caches

Speicher – Memory

- ROM: *read-only memory*
- RAM: *random-access memory*
  (besser wäre *read and write memory*)
- SRAM: *static RAM*
  Zustand bleibt solange Strom anliegt
- DRAM: *dynamic RAM*
  implementiert mit Kondensatoren
  Zustand verliert sich ohne regelmässiges Auffrischen (engl. refresh)

RAM in PCs

30 pin SIMM



72 pin SIMM

MicroDIMM

184 pin RAMBus RIMM

100 pin DIMM

72 pin SODIMM

144 pin SDRAM SODIMM

200 pin DDR SODIMM

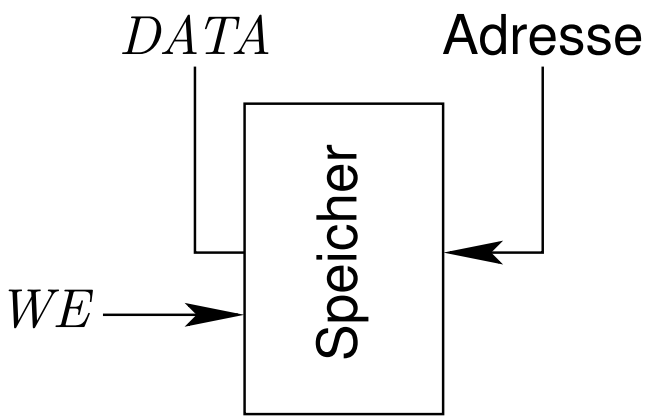
200 pin DDR-2 SODIMM

168 pin SDRAM DIMM

184 pin DDR DIMM

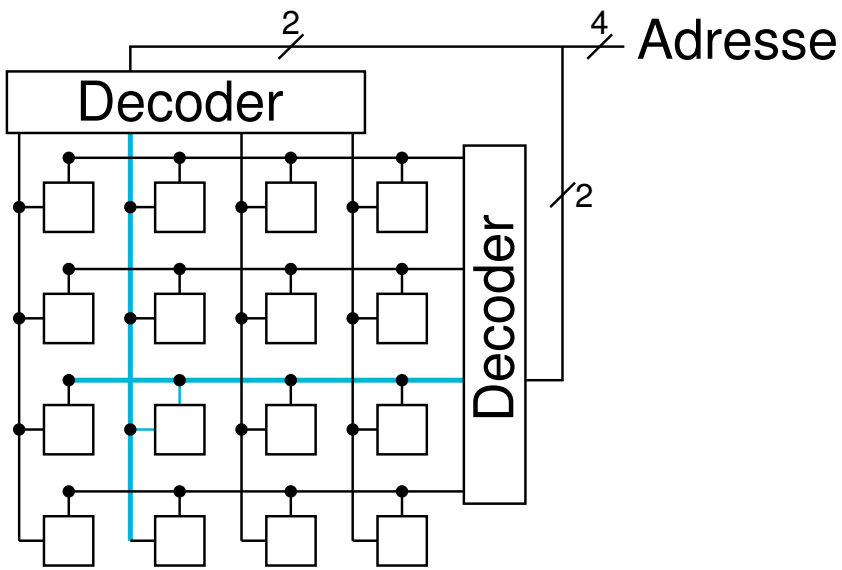
240 pin DDR-2 DIMM

Adressen



- RAM/ROM-Chips speichern viele (Milliarden) Bits
- Diese Bits werden durch Adressen unterschieden
- Die Adresse wird binärcodiert übergeben
- WE: Lesen/Schreiben
- Die Datenleitungen werden sowohl zum Lesen als auch zum Schreiben verwendet

Aufbau



- RAM/ROM-Chips sind als 2D-Matrix aufgebaut
- Die Adresse wird in Zeilen- und Spaltenadresse aufgeteilt (engl. *row* und *column*)
- Die Binärcodierung wird durch einen Decoder in eine unäre Codierung umgewandelt

Spezifikation n-Bit Decoder:

$$dec : \{0,1\}^n \longrightarrow \{0,1\}^{2^n}$$

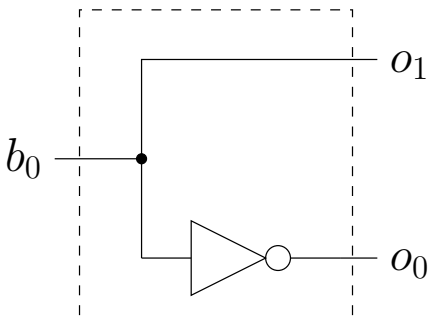
mit

$$\forall i \in \{0, \ldots 2^n - 1\}.\, dec(x)_i \iff \langle x \rangle = i$$

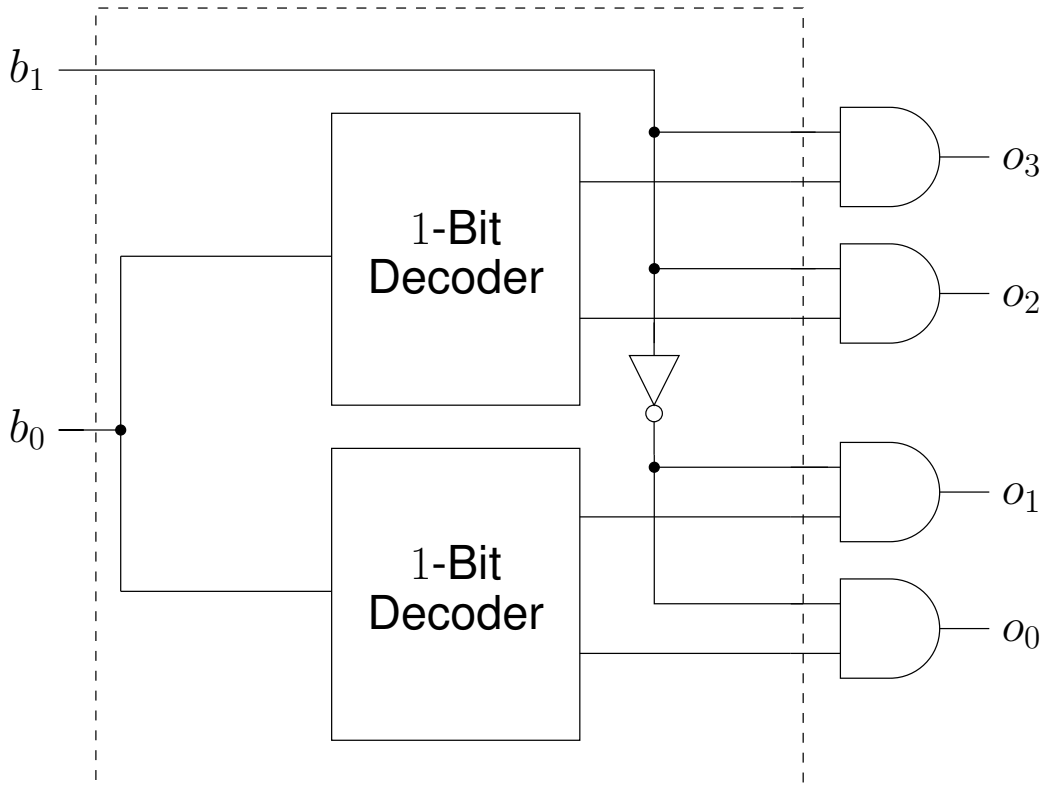
Beispiel:

$$\begin{aligned} dec(000) &= 00000001 \\ dec(001) &= 00000010 \\ dec(111) &= 10000000 \end{aligned}$$

Implementierung:

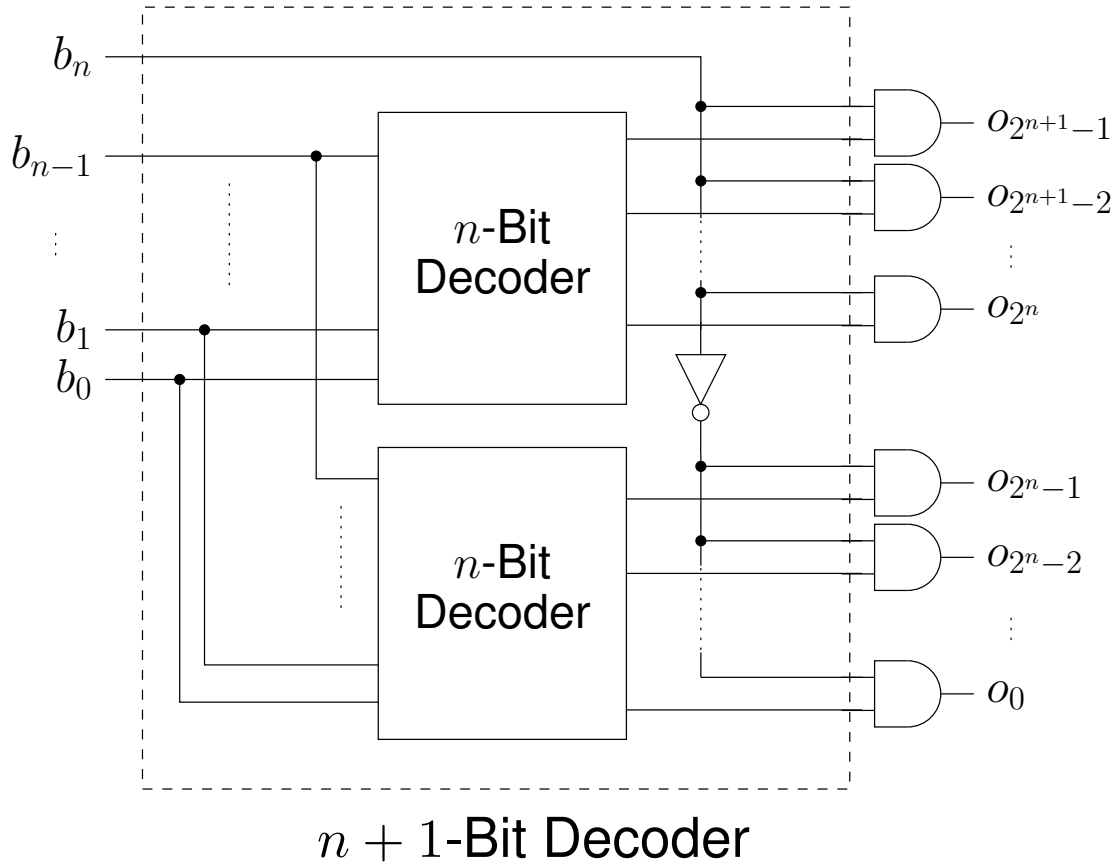


1-Bit Decoder



2-Bit Decoder

Rekursionsschema:



Read-Only Memory – ROM

- Früher: Diodenmatrizen – manuell verdrahtet
- PROM: *programmable* ROM
- EPROM: löschbar durch ultraviolettes Licht
- EEPROM: „elektronisch“ löschbar
- Flash-EEPROM: USB-Sticks, Digitalkameras, IPODs, ...

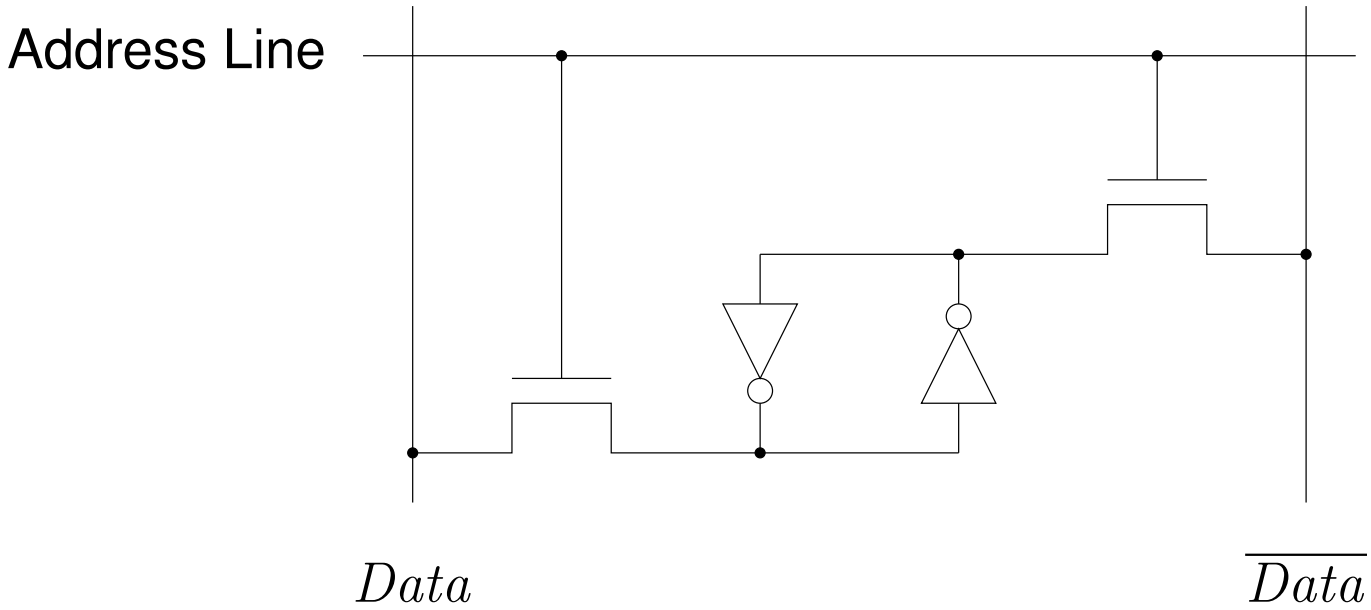
ROM Modell in Verilog

```
module my_rom16x7(input [3:0] address,
                  output [6:0] data);

  reg [6:0] rom [3:0];

 initial begin
   rom[0]='b1110111;
   // ... 14 aehnliche Zeilen ...
   rom[15]='b1101101;
  end

  assign data=rom[address];

endmodule
```

Ad: Rom Modell in Verilog

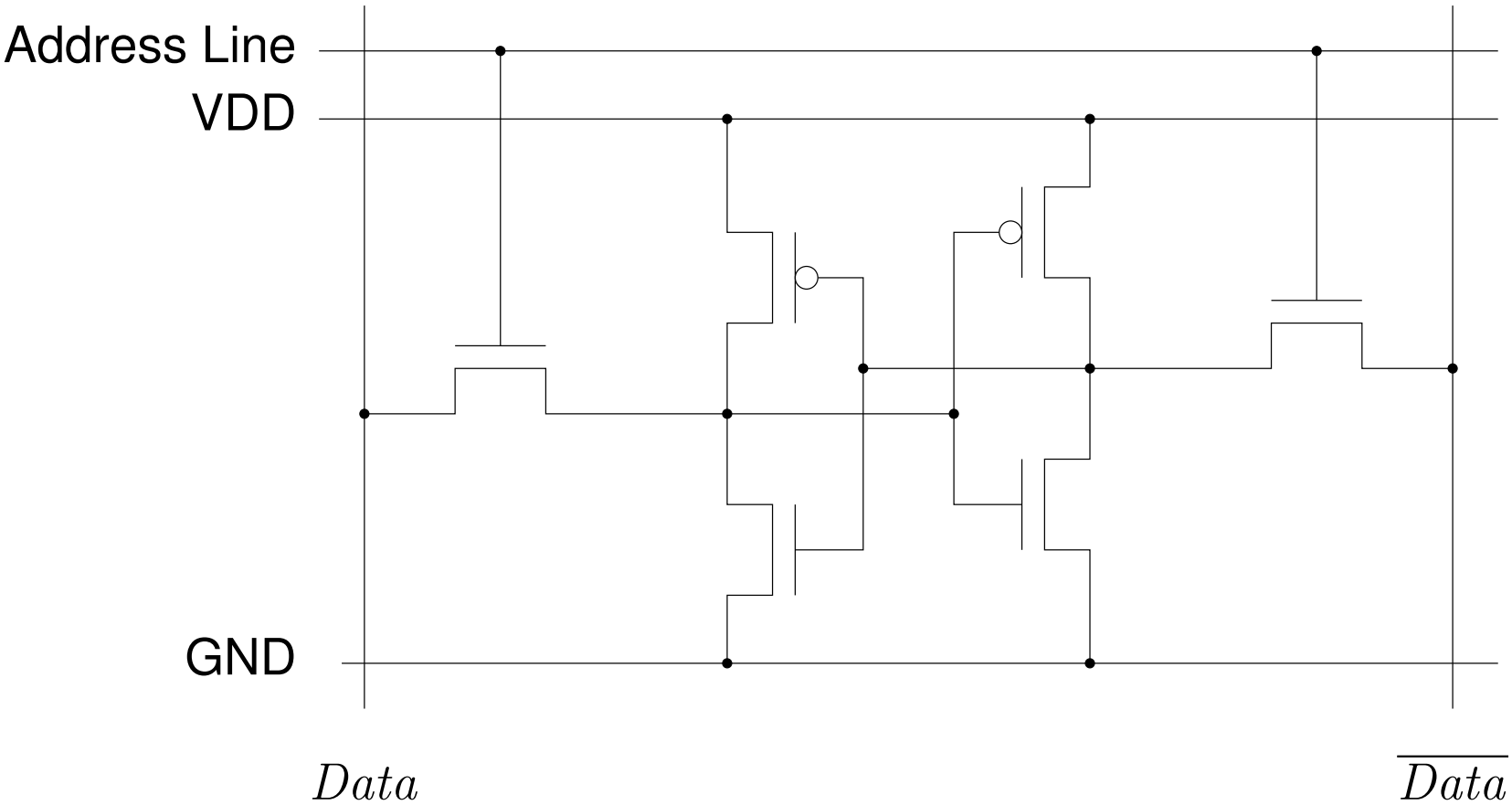
- Eigentlich nur Simulations-Modell, Implementierung des ROMs separat
- Timing mag interessant sein: `assign #5 data = ...`
- Einfache kombinatorische Logik lässt sich als ROM implementieren
- EPROM/Flash:
  Programmierung durch Anlegen hoher Spannung (z.B. -12 V)
  Programmierbarkeit wird normalerweise nicht modelliert

SRAM Zelle mit zwei Invertern



- Schreib- und lesbar
- Address Line wählt Zelle aus
- Zustand wird von den gekoppelten Invertern (Latch) gehalten
- Auslesen durch Vergleich von $Data$ und $\overline{Data}$

SRAM Zelle in CMOS

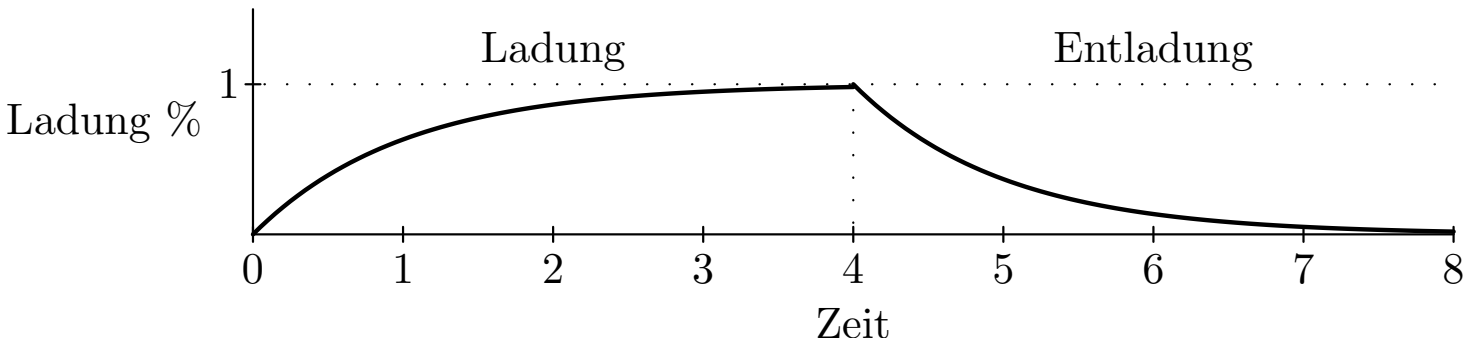


DRAM

- DRAM verwendet die Kapazität von Kondensatoren
- einfacherer und regelmäßigerer Aufbau als SRAM
- ✔ hohe Integrationsdichte, günstige Fertigungskosten
- Aber: langsamerer

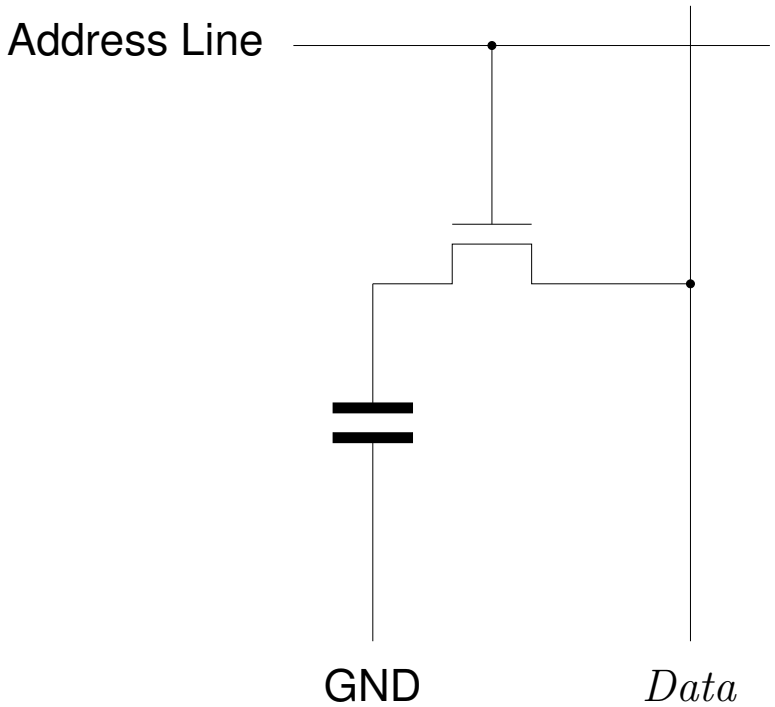
- schnelles aber teures SRAM für Cache Speicher (später)
- langsames aber billiges DRAM für Hauptspeicher

Erinnerung: Kondensatoren



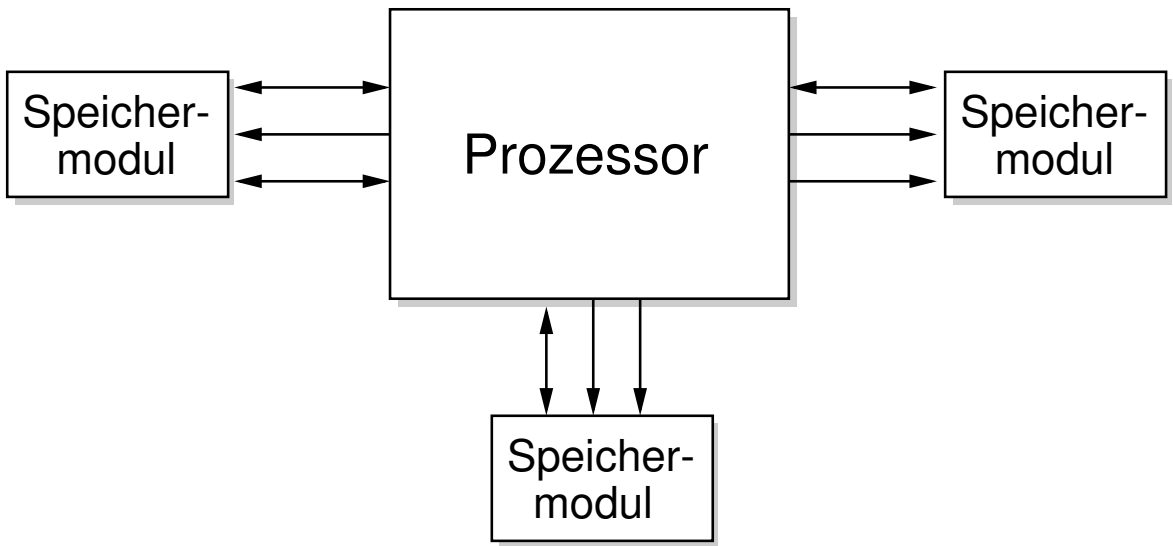
Speichern elektrische Ladung – für begrenzte Zeit

DRAM Zelle



Bit wird als Kapazität gespeichert und muss ständig aufgefrischt werden

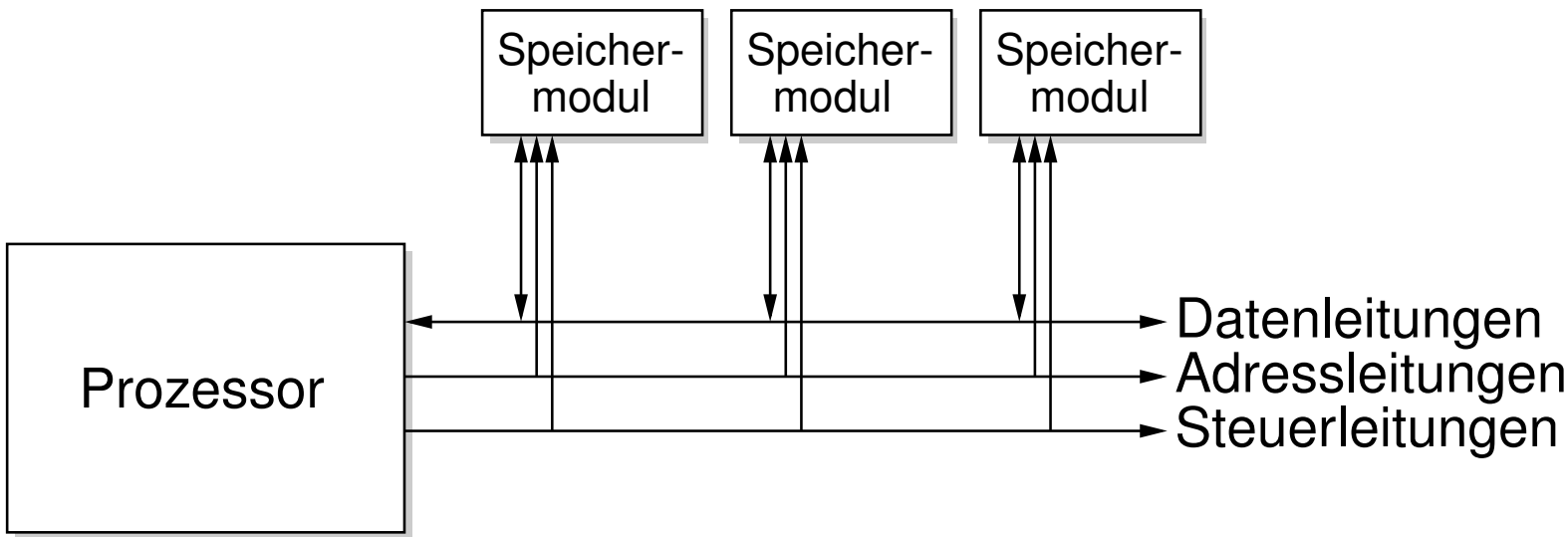
Anbindung mehrerer Spreicherchips:



Nein: I/O Pins sind teuer!

Datenbusse

- Ziel: effektive Nutzung der teuren Verbindungsleitungen
- Idee: Daten- und Adressleitungen teilen!



Interface RAM-Chips

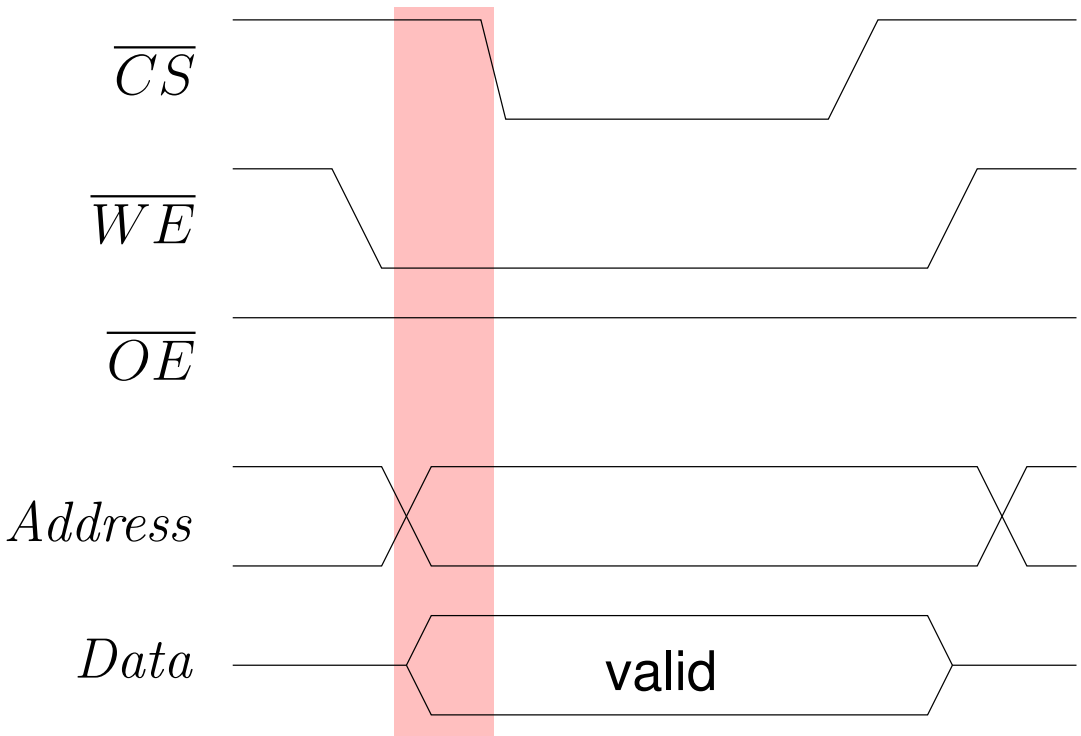
- Kontrollsignale:
  - $\overline{CS}$ (Chip Select) – aktiviert einen bestimmten RAM-Chip
  - $\overline{WE}$ (Write Enable)
  - $\overline{OE}$ (Output Enable)

- unaktiviert sind Ausgaben eines RAM-Chips im Tristate ($\mathtt{Z}$)

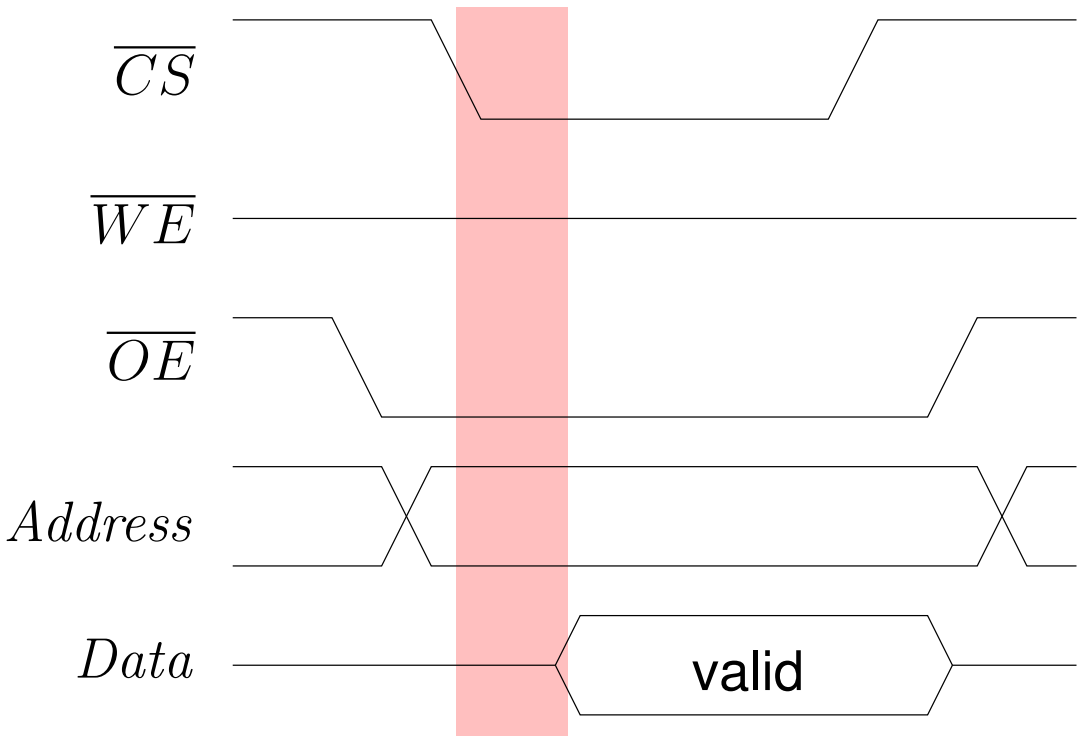
- Schreiben durch Setzen von $\overline{WE}$, Lesen durch Setzen von $\overline{OE}$

- Interface Constraint: $\overline{OE}$ und $\overline{WE}$ sind nie gleichzeitig gesetzt

Schreibzyklus



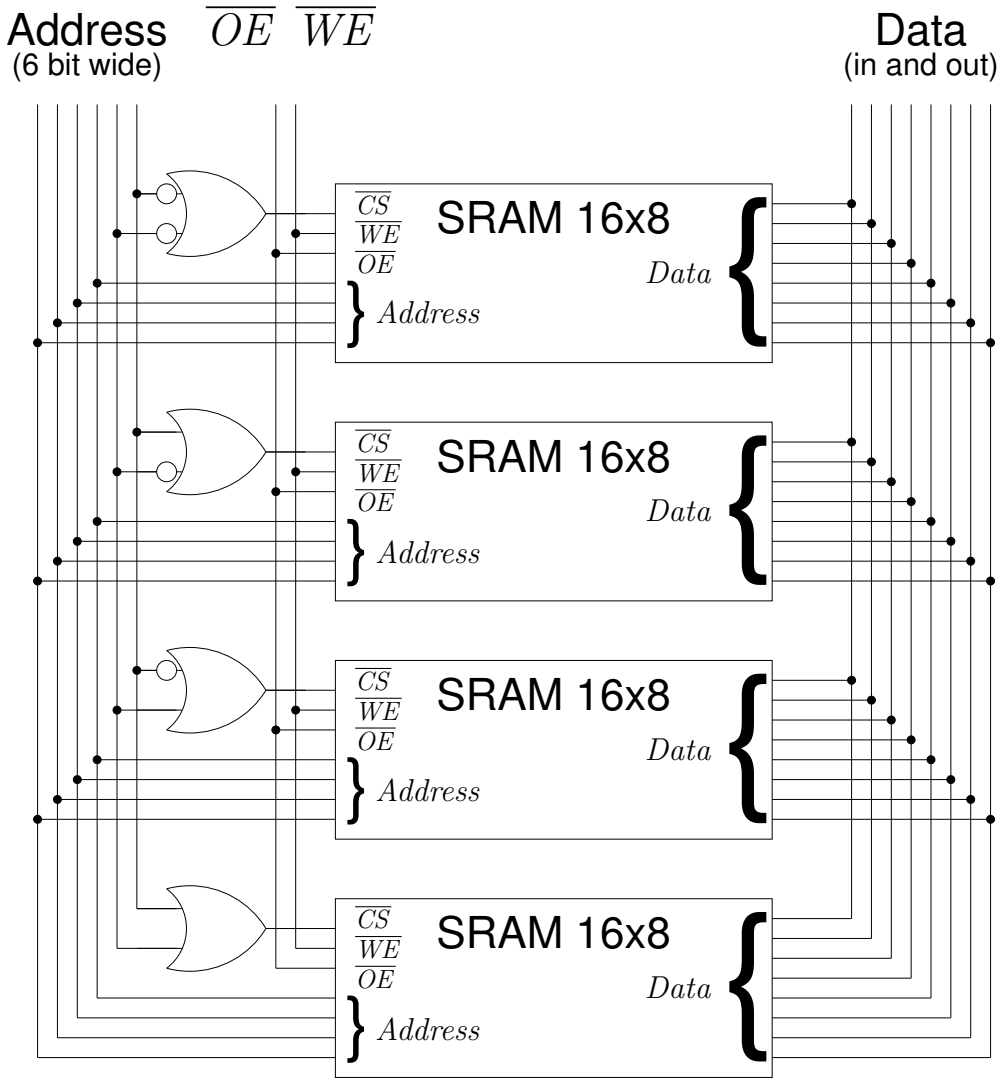
Lesezyklus



RAM-Modell in Verilog

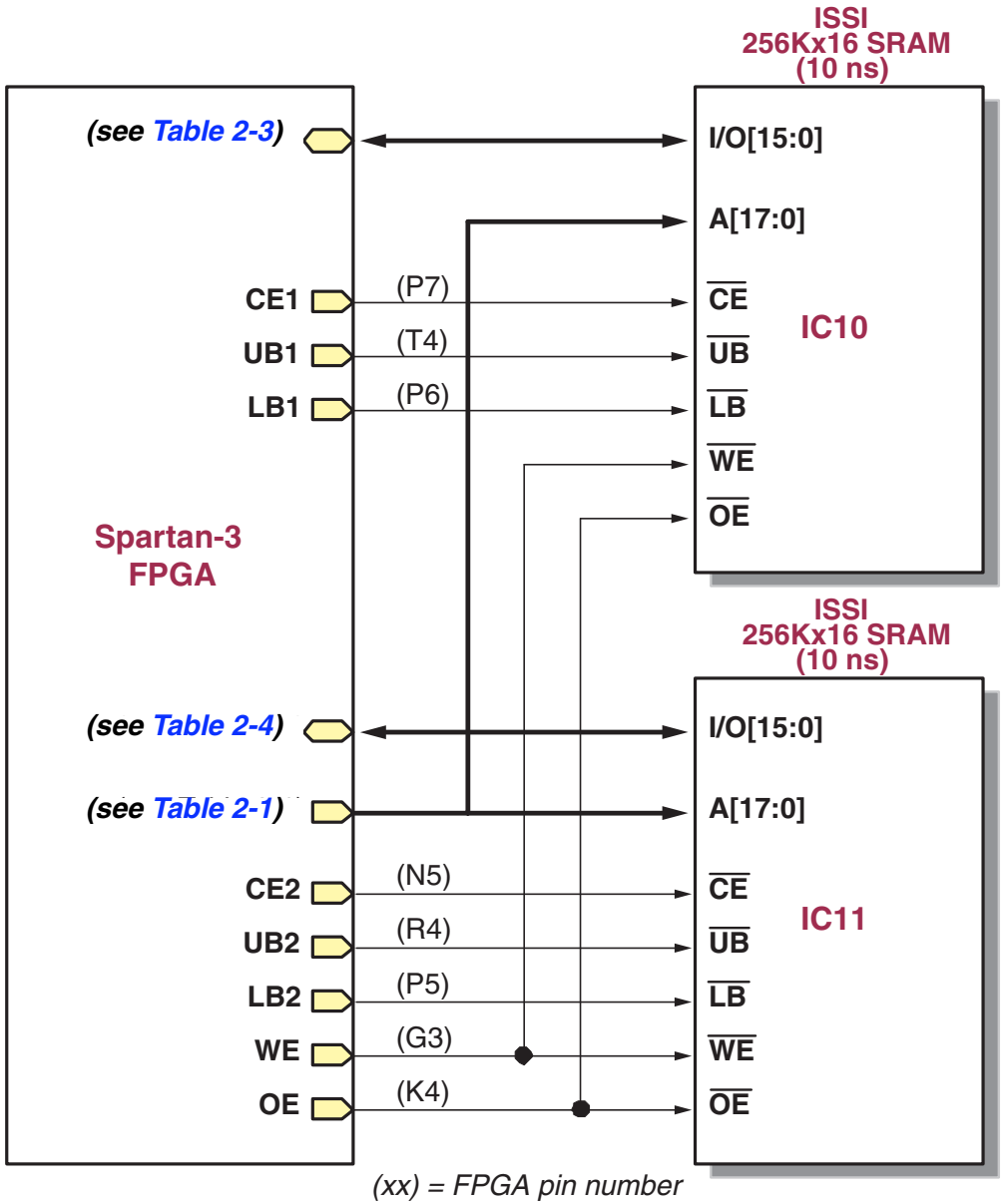
```
module my_ram16x8(input [3:0] address,
                  inout  [7:0]  data,
                  input CS, WE, OE);

  reg  [7:0]  ram  [3:0];

  assign data = (!CS && !OE) ? ram[address] : 'bzzzzzzzz;

  always @(address, CS, WE)
    if (!CS && !WE)
      ram[address] <= data;

endmodule
```

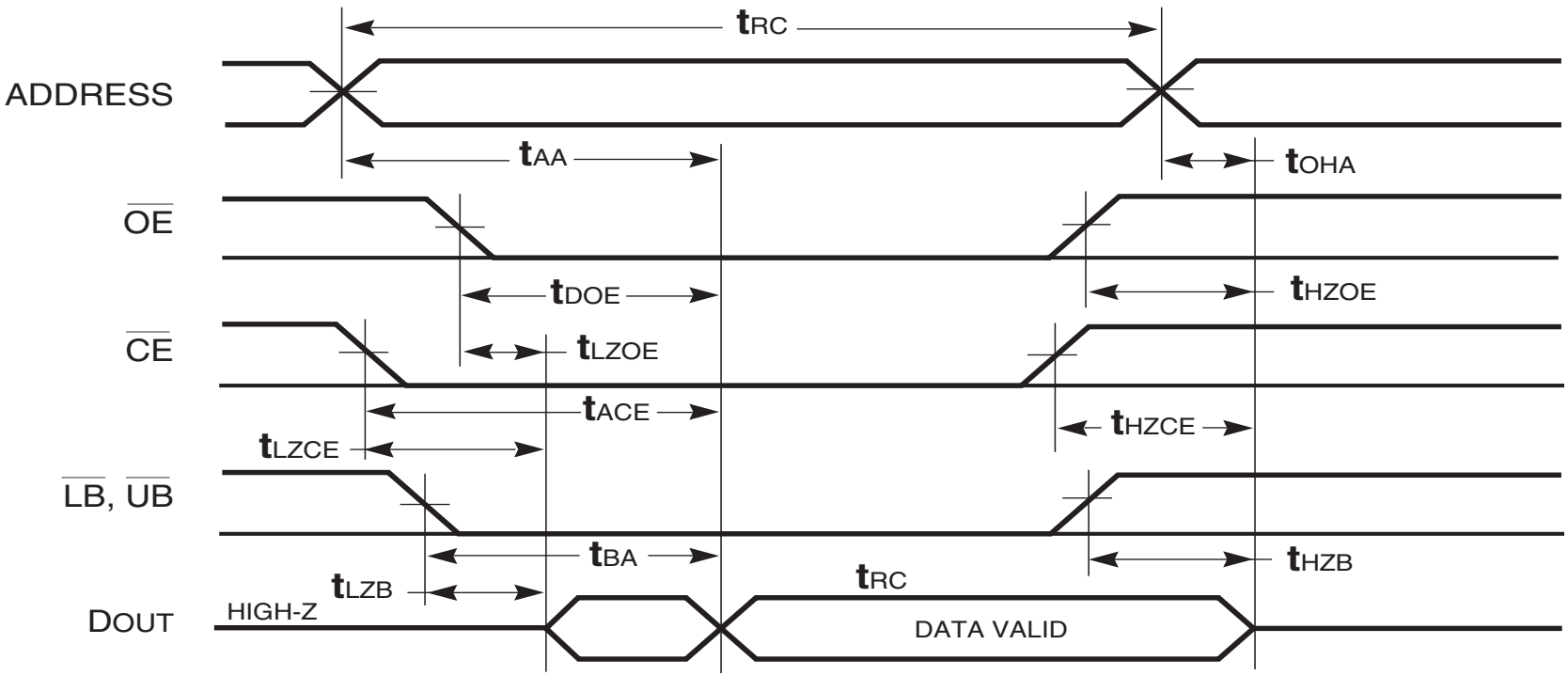
Kaskadierung von RAM-Chips



Achtung: $\overline{OE}$ und $\overline{WE}$ sind active-low!

Das SRAM auf dem FPGA-Board



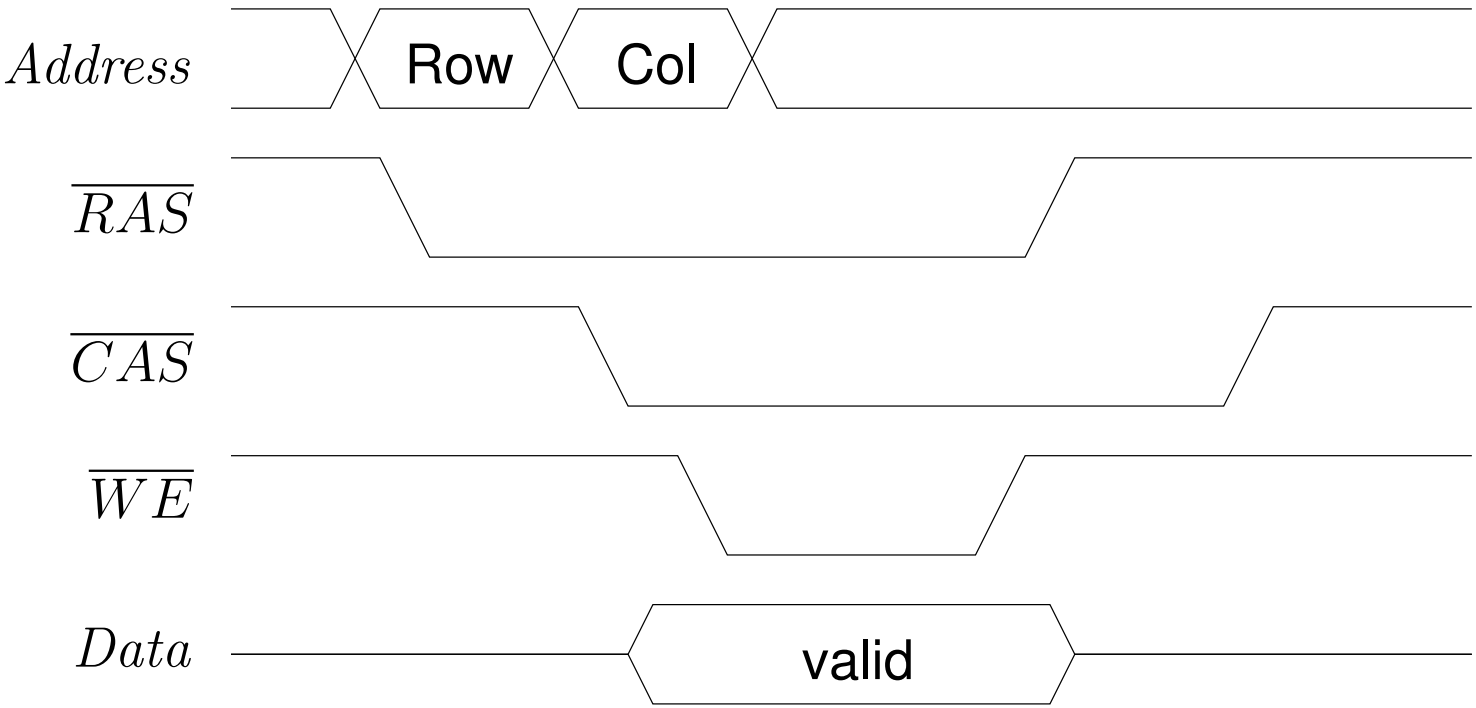


Lesezyklus des IS61LV25616AL von ISSI

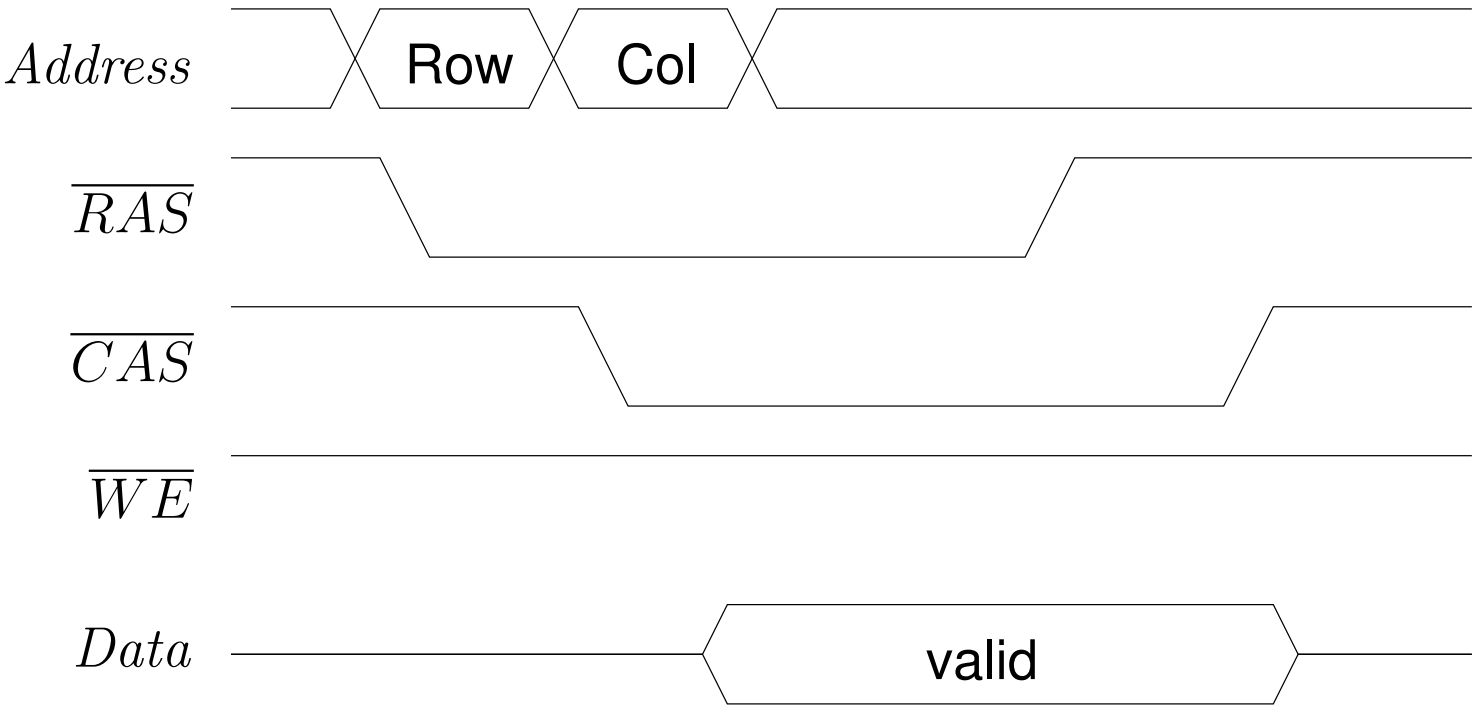
Row- und Column-Address-Strobes

- Idee: Leitungen sparen durch stückweise Übertragung der Adresse
- Beispiel: Zeile und Spalte werden getrennt übertragen
- $\overline{RAS}$: *Row Address Strobe*,
  $\overline{CAS}$: *Column Address Strobe*

RAS/CAS Schreibzyklus



RAS/CAS Lesezyklus



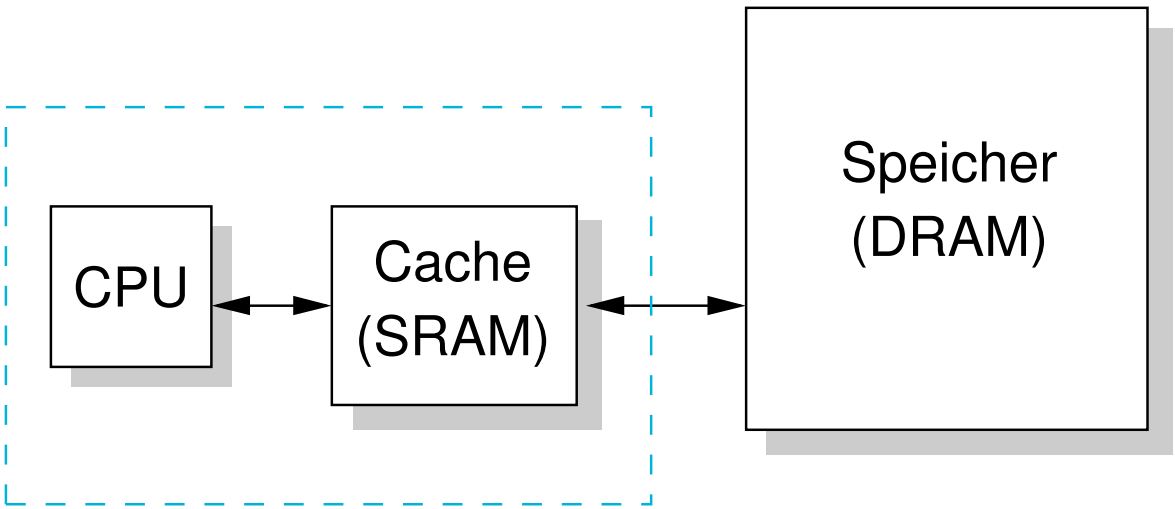
RAM-Modell mit RAS/CAS in Verilog

```
module DRAM8MBIT(input [9:0] address, inout [7:0] data,
                 input RAS, CAS, WE, OE);

  reg [9:0]  row_addr;
  reg [19:0]  mem_addr;

  reg [7:0]  mem [0:(1<<19)-1];

  always @(negedge RAS)
    row_addr <= address;

  always @(negedge CAS) begin
    mem_addr = (row_addr<<10) + address;
    if (WE==0)
      mem[mem_addr] <= data;
  end

  assign data = !OE ? mem[mem_addr] : 'bzzzzzzzz;

endmodule
```

Caches

- Erinnerung: DRAM langsam/billig, SRAM schnell/teuer
- Idee: SRAM als schnellen Zwischenspeicher (Cache) für DRAM verwenden
- „Versteckt“ die Latenz des langsamen DRAMs
- Oft gute Trefferraten: $> 90\,\%$

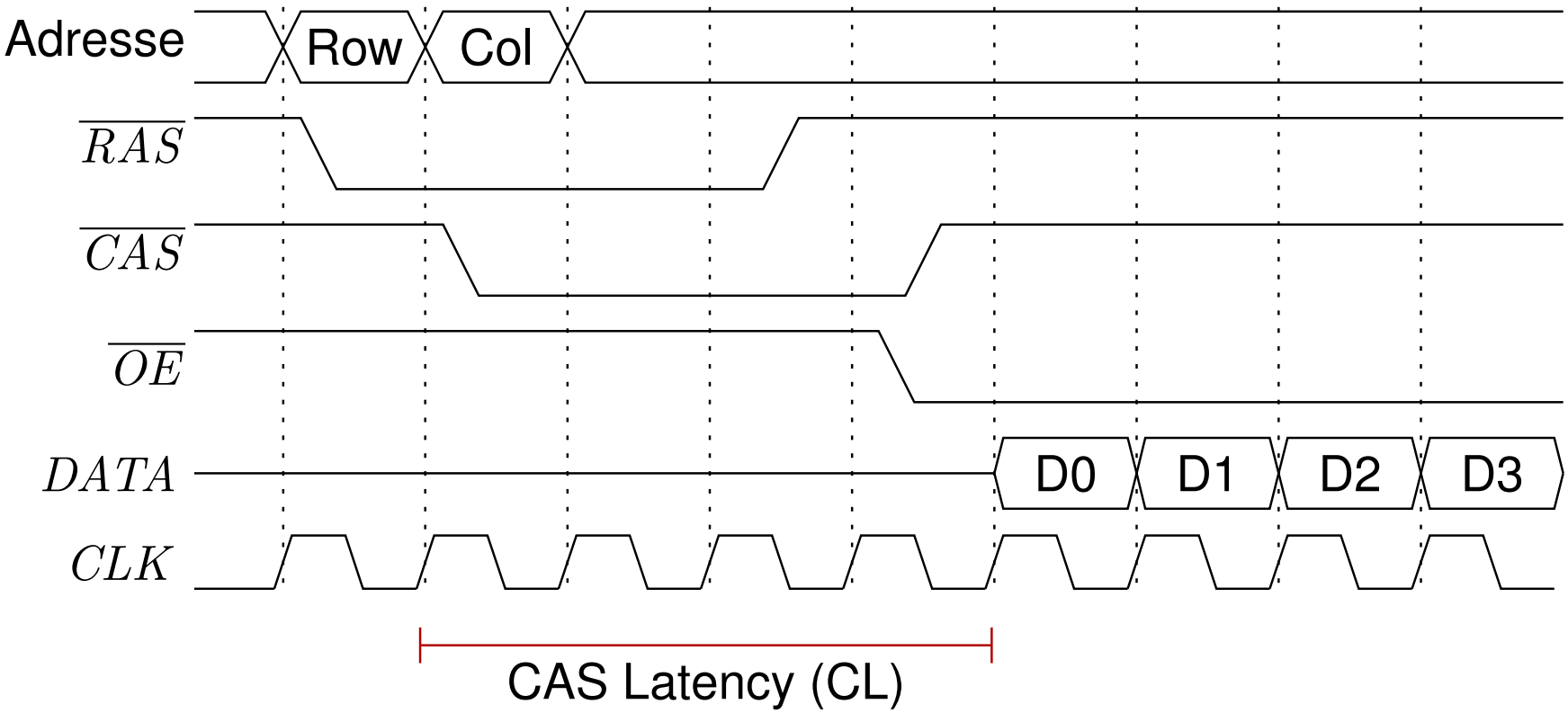
Caches



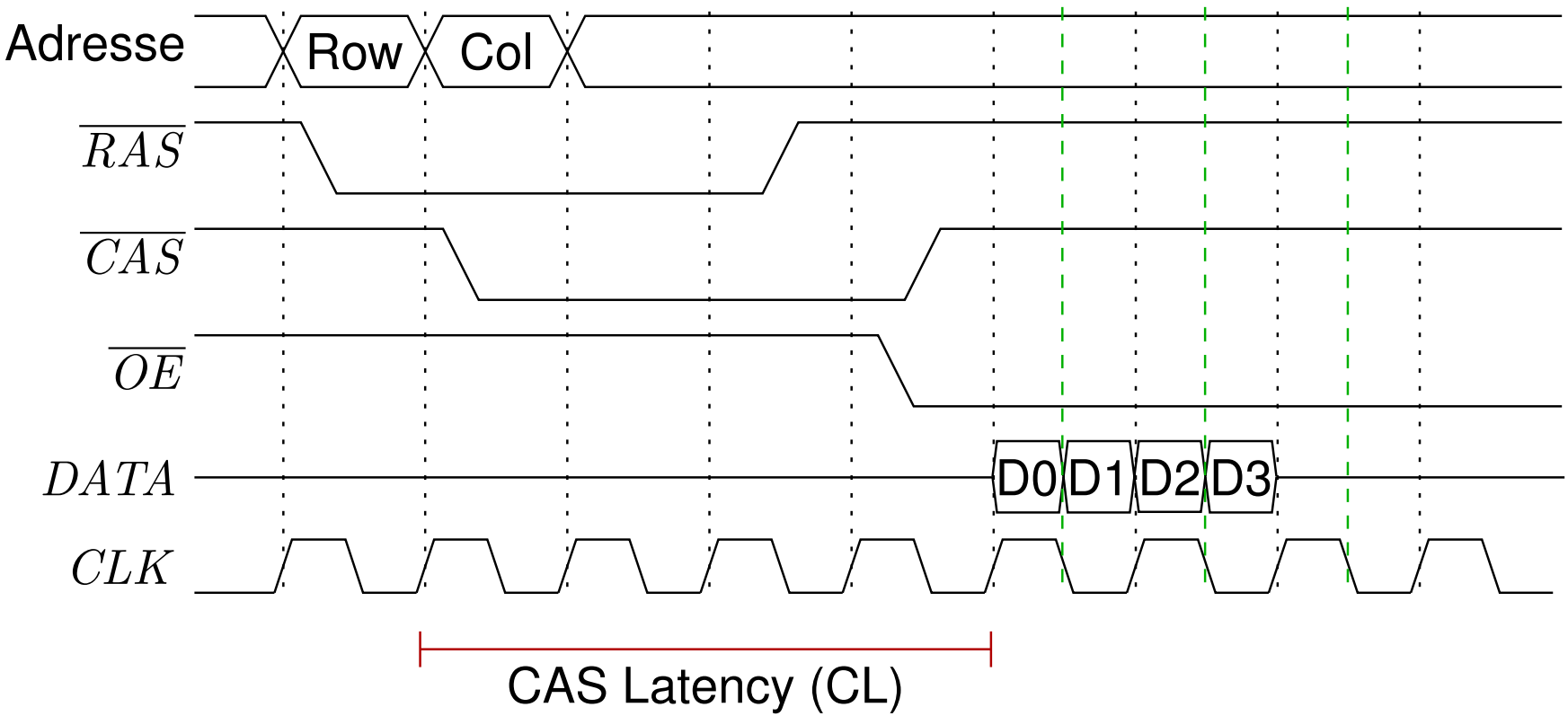
Bus-Bursts

- RAM wird oft sequentiell gelesen
- Caches sind daher in Zeilen organisiert:
  aufeinanderfolgende Adressen (z.B. 256 Bytes)
- Bus-Bursts: effiziente Übertragung einer ganzen Zeile
  (ggf. mit Interleaving)

Bus-Bursts


Adresse
Row
Col
$\overline{RAS}$
$\overline{CAS}$
$\overline{OE}$
$DATA$
CLK
D0
D1
D2
D3
CAS Latency (CL)

Double Data Rate (DDR) RAM



Timings

4GB 800MHz DDR2 Non-ECC CL5 (5-5-5-15) DIMM (Kit of 2)	KHX6400D2K2/4G
4GB 800MHz DDR2 Non-ECC Low-Latency CL4 (4-4-4-12) DIMM (Kit of 2)	KHX6400D2LLK2/4G
4GB 1066MHz DDR2 Non-ECC CL5 (5-5-5-15) DIMM (Kit of 2)	KHX8500D2K2/4G
4GB 1066MHz DDR2 Non-ECC CL5 (5-5-5-15) DIMM (Kit of 2) Tall HS	KHX8500D2T1K2/4G
4GB 1066MHz DDR2 Non-ECC CL5 (5-5-5-15) DIMM (Kit of 4)	KHX8500D2K4/4G
8GB 800MHz DDR2 Non-ECC Low-Latency CL4 (4-4-4-12) DIMM (Kit of 4)	KHX6400D2LLK4/8G
HyperX DDR3 1375MHz, 1600MHZ, 1625MHz, 1800MHz, 1866MHz and 2000MHz	
Description	**Part Number**
1GB 1375MHz DDR3 Non-ECC Low-Latency CL7 (7-7-7-20) DIMM	KHX11000D3LL/1G
1GB 1600MHz DDR3 Non-ECC CL9 (9-9-9-27) DIMM	KHX12800D3/1G
1GB 1625MHz DDR3 Non-ECC Low-Latency CL7 (7-7-7-20) DIMM	KHX13000D3LL/1G
1GB 1625MHz DDR3 Non-ECC Low-Latency CL7 (7-7-7-20) DIMM	KHX13000AD3LL/1G
1GB 1800MHz DDR3 Non-ECC CL8 (8-8-8-24) DIMM	KHX14400D3/1G
1GB 1800MHz DDR3 Non-ECC CL8 (8-8-8-24) DIMM	KHX14400AD3/1G
2GB 1375MHz DDR3 Non-ECC CL9 (9-9-9) DIMM (Kit of 2)	KHX11000D3K2/2G
2GB 1375MHz DDR3 Non-ECC Low-Latency CL7 (7-7-7-20) DIMM	KHX11000D3LL/2G
2GB 1375MHz DDR3 Non-ECC CL7 (7-7-7-20) DIMM (Kit of 2)	KHX11000D3LLK2/2G
2GB 1375MHz DDR3 Non-ECC CL7 (7-7-7-20) DIMM (Kit of 2) Intel XMP	KHX11000D3LLK2/2GX
2GB 1600MHz DDR3 Non-ECC CL9 (9-9-9-27) DIMM	KHX12800D3/2G
2GB 1600MHz DDR3 Non-ECC CL9 (9-9-9-27) DIMM (Kit of 2)	KHX12800D3K2/2G
2GB 1625MHz DDR3 Non-ECC Low-Latency CL7 (7-7-7-20) DIMM	KHX13000D3LL/2G
2GB 1625MHz DDR3 Non-ECC Low-Latency CL7 (7-7-7-20) DIMM (Kit of 2)	KHX13000D3LLK2/2G
2GB 1625MHz DDR3 Low Latency CL8 (8-7-7-20) DIMM (Kit of 2) NVIDIA SLI	KHX13000D3LLK2/2GN

Timings

Beispiel: 2-2-2-5

Aktueller Standard:

1. CAS Latency
2. RAS-to-CAS Delay
3. RAS Precharge
4. Act-to-Precharge Delay