

Digitaltechnik

Wintersemester 2025/2026

Vorlesung 12



TECHNISCHE
UNIVERSITÄT
DARMSTADT



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

1 Speicherfelder

2 Logikfelder



Harris 2016
Kap. 5.5, 5.6

Anwendungs- software	>"hello world!"	Programme
Betriebs- systeme		Gerätereiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



Abgabefrist für Hausaufgabe E zu
Vorlesungen 09 und 10 **diese Woche**
Freitag 23:59!

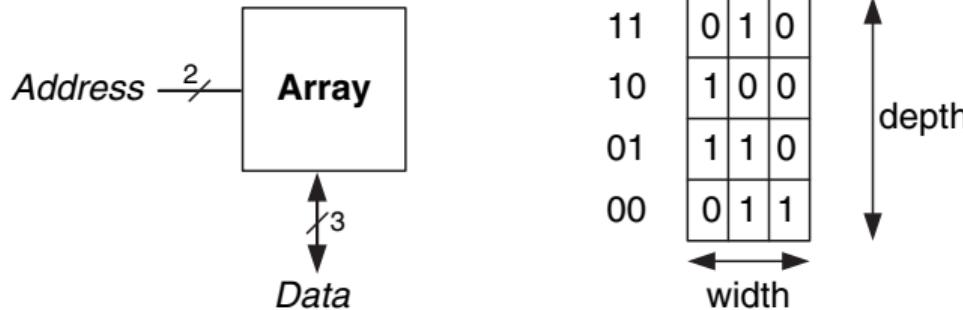
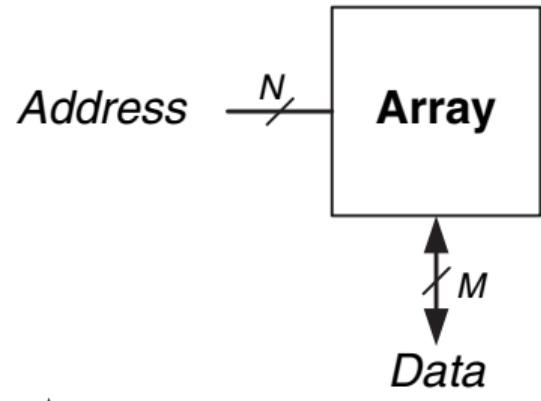
Wöchentliches Moodle-Quiz nicht vergessen!

1 Speicherfelder

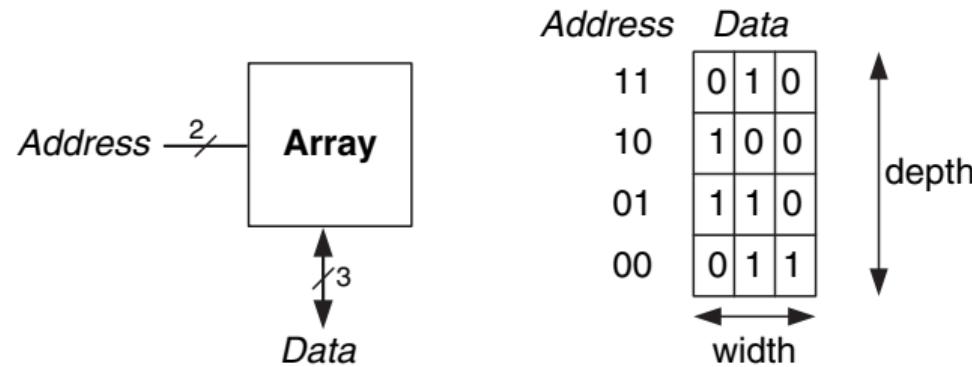
2 Logikfelder

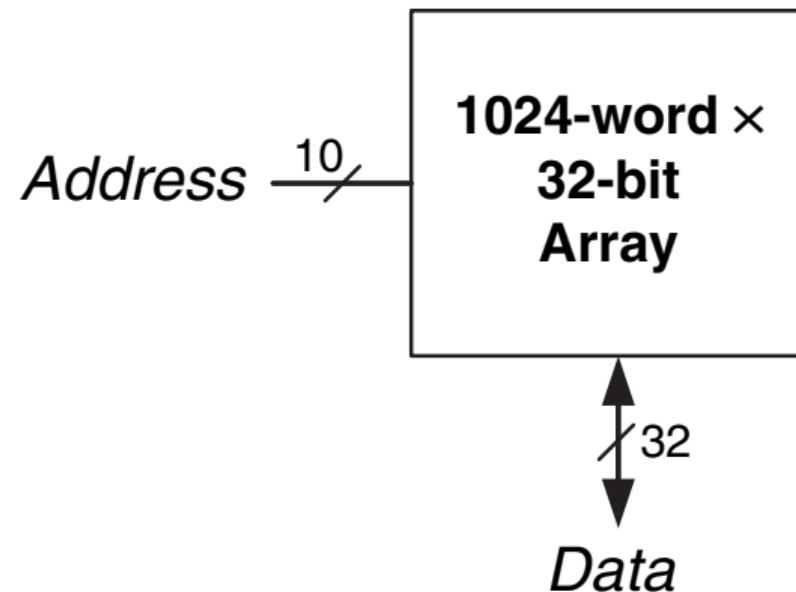
Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätereiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

- 2-dimensionales Array von Bitzellen
- Jede Bitzelle speichert ein Bit
- N Adressbits und M Datenbits:
 - 2^N Zeilen und M Spalten
 - **Tiefe:** Anzahl der Zeilen (Anzahl der Wörter)
 - **Breite:** Anzahl der Spalten (Wortbreite)
 - **Größe:** Tiefe \times Breite = $2^N \times M$ Bits



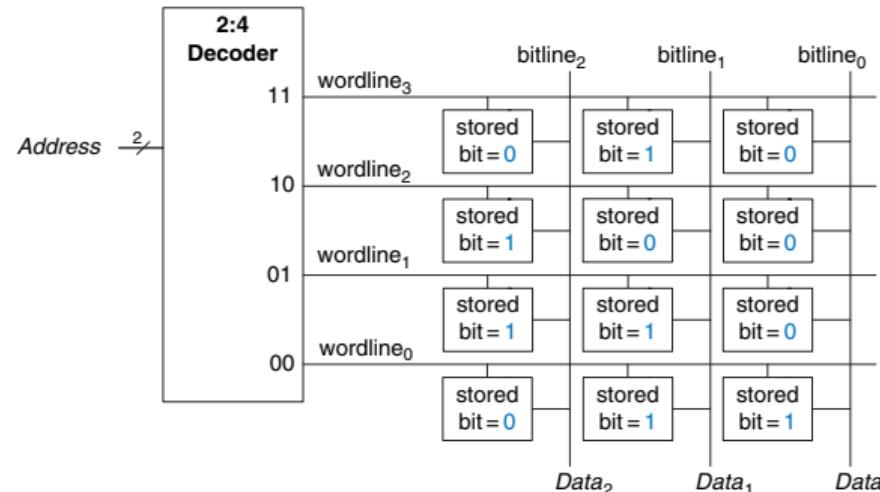
- $2^2 \times 3$ -Bit Array
- Anzahl der Wörter: 4
- Wortbreite: 3 Bits
- Beispiel: das 3-Bit Wort, das an der Adresse 10 gespeichert ist, lautet 100

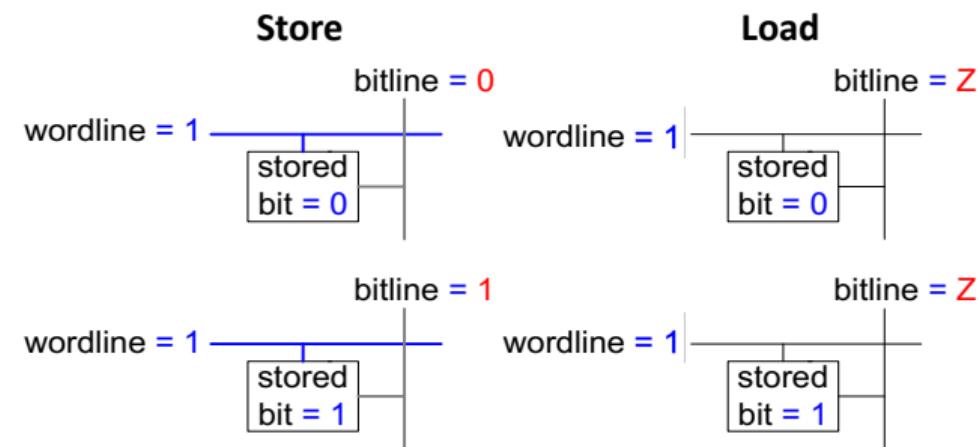
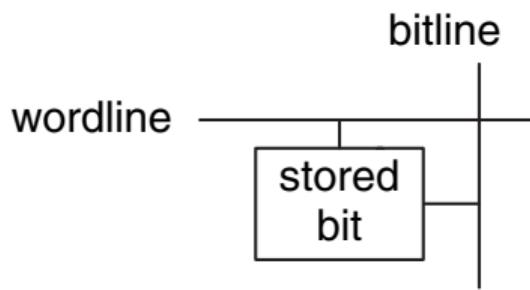




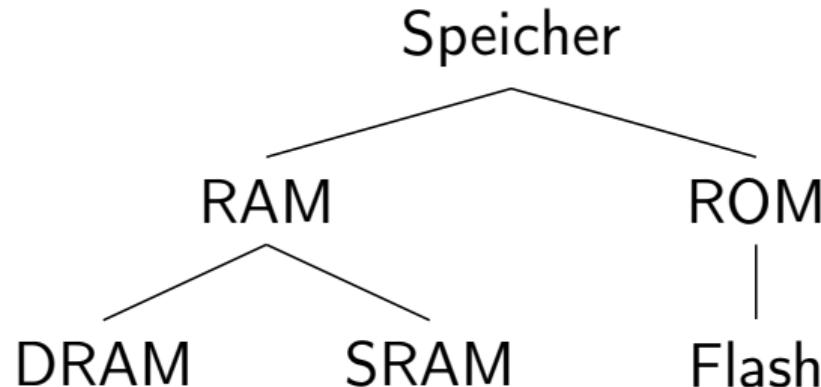
■ Wordline:

- Vergleichbar zu *ENABLE* Signal
- Einzelne Zeile im Speicherfeld wird gelesen/geschrieben
- Entspricht einer eindeutigen Adresse
- Maximal eine Wordline kann HIGH sein





- Direktzugriffsspeicher (*random access memory*, RAM): **flüchtig**
 - Dynamic RAM (DRAM)
 - Static RAM (SRAM)
- Festwertspeicher (*read-only memory*, ROM): **nicht flüchtig**
 - Flash



- **Flüchtig:** verliert Daten beim Ausschalten
- Schnelles Lesen und Schreiben
- Der Hauptspeicher Ihres Computers ist RAM (meist DRAM)

Historisch als Direktzugriffsspeicher bezeichnet, da auf jedes Datenwort gleich schnell / direkt zugegriffen werden kann (im Gegensatz zu sequentiellen Zugriffsspeichern wie Audiokassette oder Bandlaufwerk)

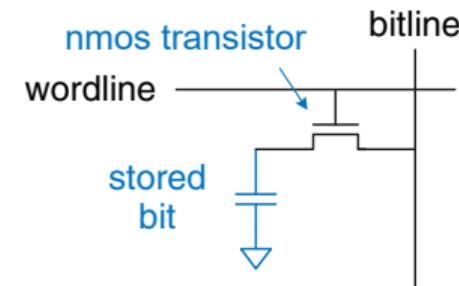
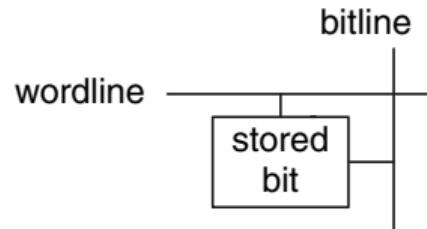
- **Nicht flüchtig:** Daten bleiben beim Ausschalten erhalten
- Schnelles Lesen, aber Schreiben ist unmöglich oder langsam
- Flash-Speicher in Digitalkameras, USB-Sticks und SSDs sind alles ROMs

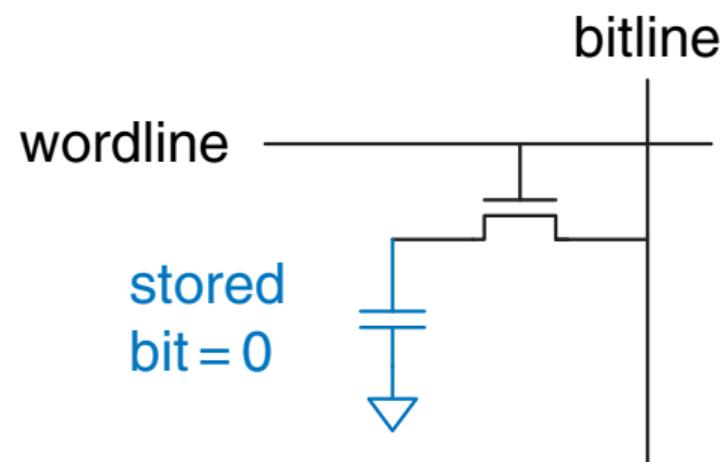
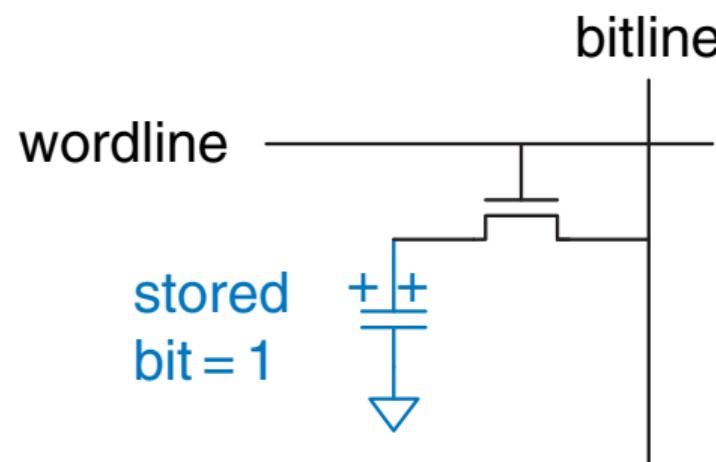
Historisch als Festwertspeicher bezeichnet, da ROMs zum Zeitpunkt der Herstellung oder durch Brennen von Sicherungen einmalig beschrieben wurden.

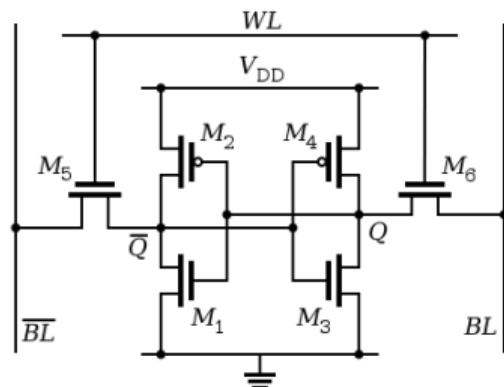
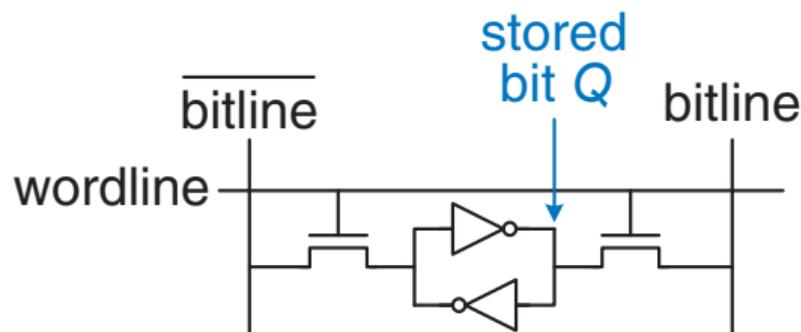
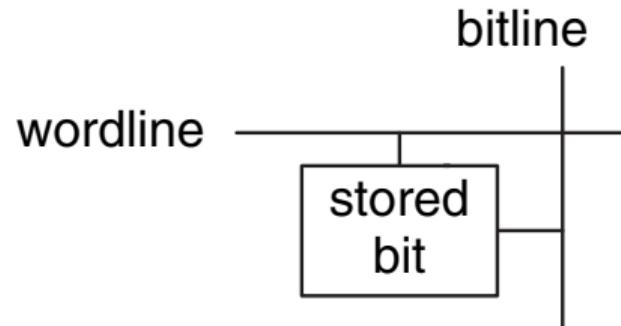
Sobald das ROM konfiguriert wurde, konnte es nicht erneut beschrieben werden.
Neuere Arten von ROMs wie Flash-Speicher können mehrmals beschrieben werden.

- **DRAM** (*dynamic random access memory*)
 - DRAM verwendet Kondensator zur Datenspeicherung
- **SRAM** (*static random access memory*)
 - verwendet Inverter mit Rückkopplung zur Datenspeicherung

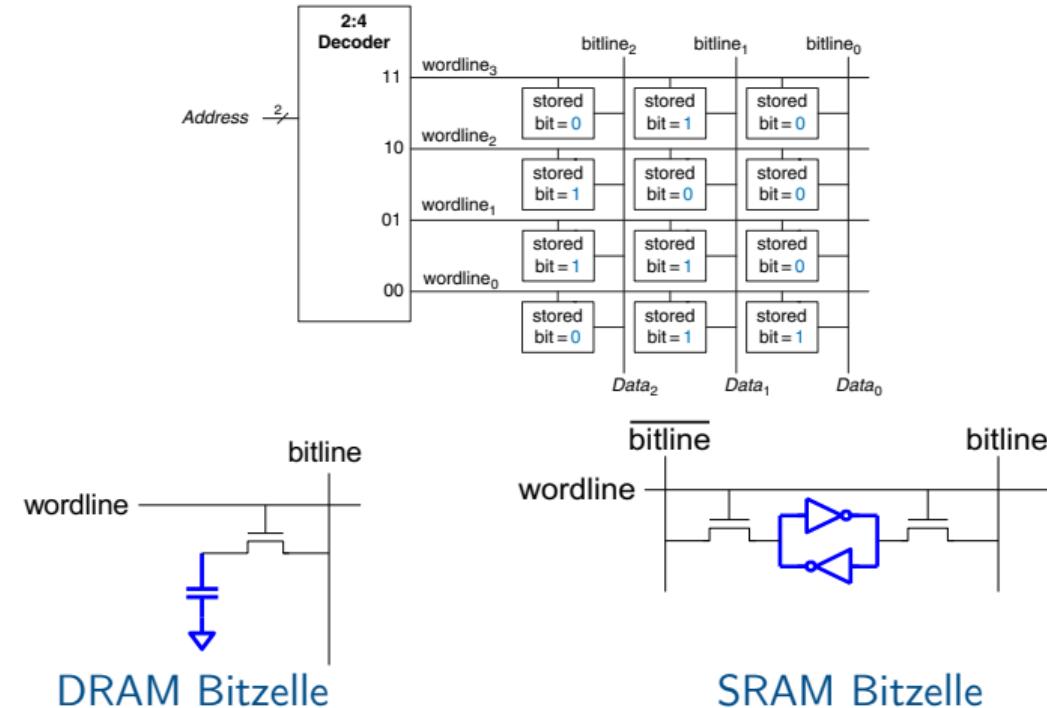
- Datenbits werden in Kondensator gespeichert
- *Dynamisch*, weil der Wert regelmäßig und nach dem Lesen aktualisiert (neu geschrieben) werden muss:
 - Ladungsverlust des Kondensators verschlechtert den Wert mit der Zeit ($1 \rightsquigarrow 0$)
 - Lesen zerstört den gespeicherten Wert







RAM Speicherfelder im Überblick



```
vectors_vs_arrays.sv
1 logic [3:0] A;           // 4 bit Vektor
2 logic A [1:0];          // Array mit 2 bits
3 logic [3:0] A [1:0];    // Array mit 2 4 bit Vektoren
```

- SystemVerilog unterstützt Vektoren und Arrays
- Arrays unterstützen jedoch weniger Operationen, beispielsweise keine bitweisen oder arithmetischen Operationen
- ⇒ Aber: Kombination aus beiden zum Beispiel für Beschreibung von RAM sehr nützlich

SystemVerilog RAM



ram.sv

```
1 // 2**N-Wort x M-bit RAM
2 module ram #(parameter N=6, M=32) // N Adressbits, M Datenbits
3     (input logic             CLK,
4      input logic             WE, // write enable
5      input logic [N-1:0]      ADR,
6      input logic [M-1:0]      D_IN,
7      output logic [M-1:0]     D_OUT);
8
9 logic [M-1:0] mem [2**N-1:0]; // 2**N Vektoren mit je M bits
10
11 // Schreiben
12 always_ff @(posedge CLK) begin
13     if (WE) begin
14         mem [ADR] <= D_IN;
15     end
16 end
17
18 // Lesen
19 assign D_OUT = mem [ADR];
20 endmodule
```



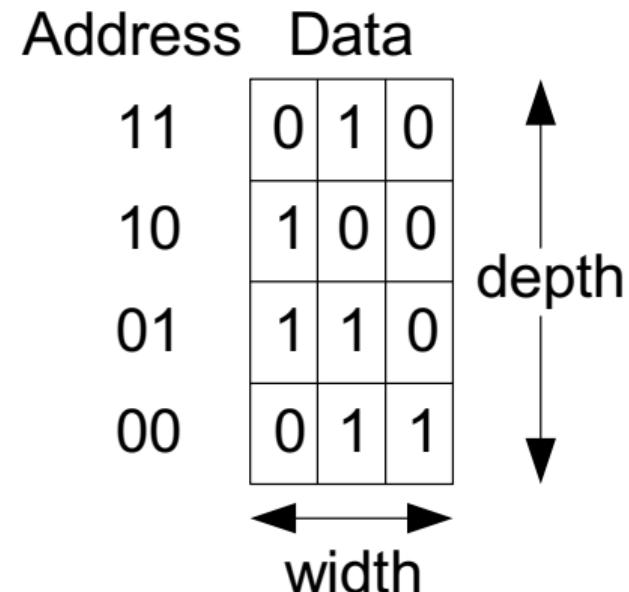
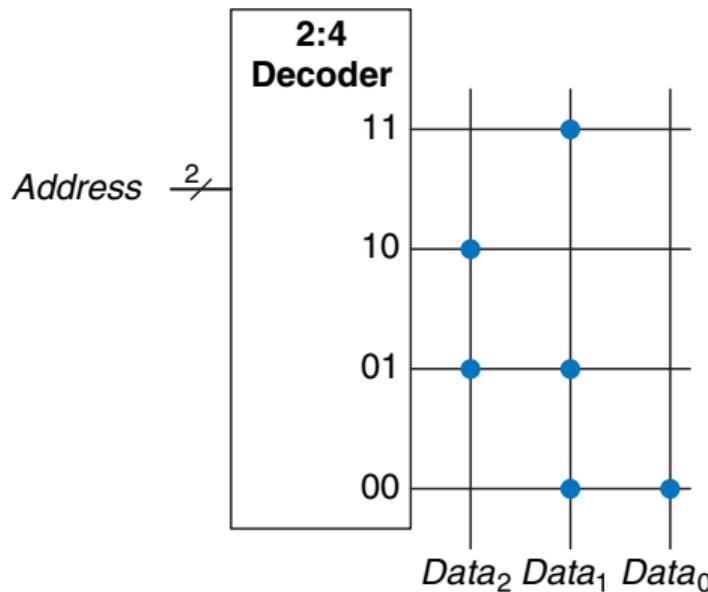
SystemVerilog RAM

Schreiben und Lesen



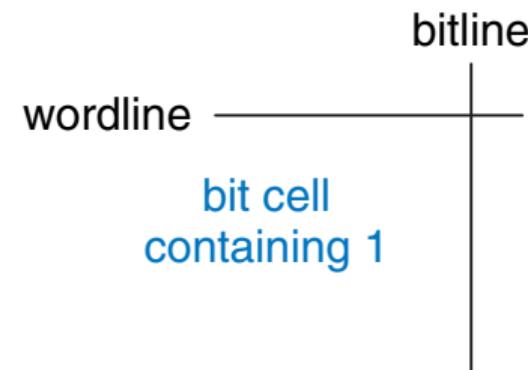
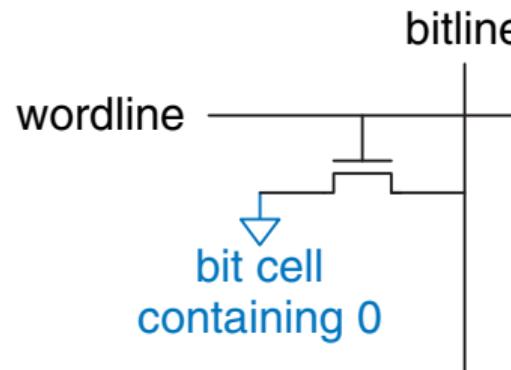
```
ram #(N(4), M(32)) uut(CLK, WE, ADR, D_IN, D_OUT);
initial begin
    $dumpfile("ram_tb.vcd"); $dumpvars;
    WE = 1;
    for (int i = 0; i < 16; i++) begin
        ADR = i[3:0];
        D_IN = 10 * ADR;
        #1;
    end
    WE = 0;
    #1;
    for (int i = 0; i < 16; i++) begin
        ADR = i[3:0];
        #1;
        $display("Read value %d at location %d", D_OUT, ADR);
    end
    $display("FINISHED ram_tb"); $finish;
end
```

```
VCD info: dumpfile ram_tb.vcd opened for output
VCD warning: $dumpvars: Package ($unit) is not
Read value      0 at location 0
Read value      10 at location 1
Read value      20 at location 2
Read value      30 at location 3
Read value      40 at location 4
Read value      50 at location 5
Read value      60 at location 6
Read value      70 at location 7
Read value      80 at location 8
Read value      90 at location 9
Read value     100 at location 10
Read value     110 at location 11
Read value     120 at location 12
Read value     130 at location 13
Read value     140 at location 14
Read value     150 at location 15
FINISHED ram_tb
ram_tb.sv:27: $finish called at 3300 (10ps)
```



Lesen: Bitline auf weak high und danach wordline auf 1 setzen.

Wenn Transistor vorhanden, zieht dieser die bitline auf 0, sonst bleibt diese auf 1.

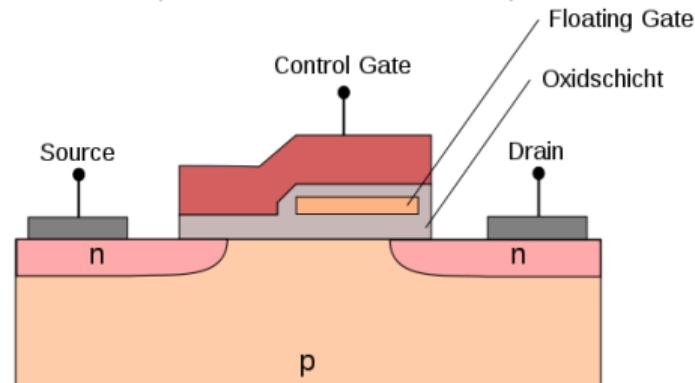


rom.sv

```
1 // 4-Wort x 3-bit ROM
2 module rom(input logic [1:0] ADR,
3             output logic [2:0] D_OUT);
4
5   always_comb begin
6     case(ADR)
7       2'b11: D_OUT = 3'b010;
8       2'b10: D_OUT = 3'b100;
9       2'b01: D_OUT = 3'b110;
10      2'b00: D_OUT = 3'b011;
11    endcase
12  end
13 endmodule
```

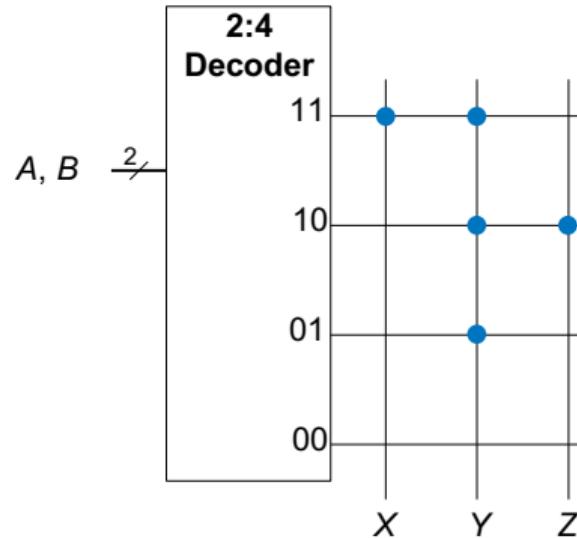


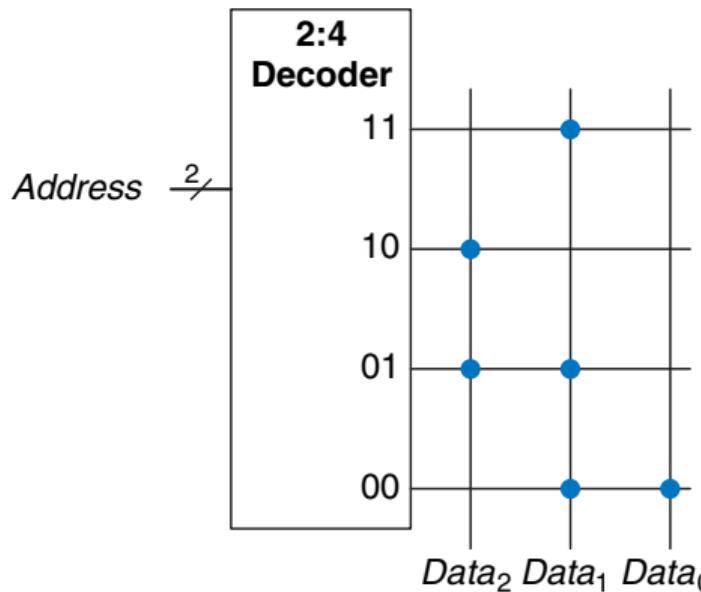
- Ladung auf Floating Gate beeinflusst Leitfähigkeit des Feldeffekttransistors von Source nach Drain.
- Floating Gate kann durch Anlegen von hoher Spannung geladen / entladen werden.
 - Laden : Source = 0 V, Control Gate = Drain = 12 V
 - Entladen: Source = offen, Control Gate = 0 V, Drain = 12 V



Implementieren Sie die folgenden Logikfunktionen mit einem $2^2 \times 3$ -Bit ROM:

- $X = A \cdot B$
- $Y = A + B$
- $Z = A \cdot \overline{B}$



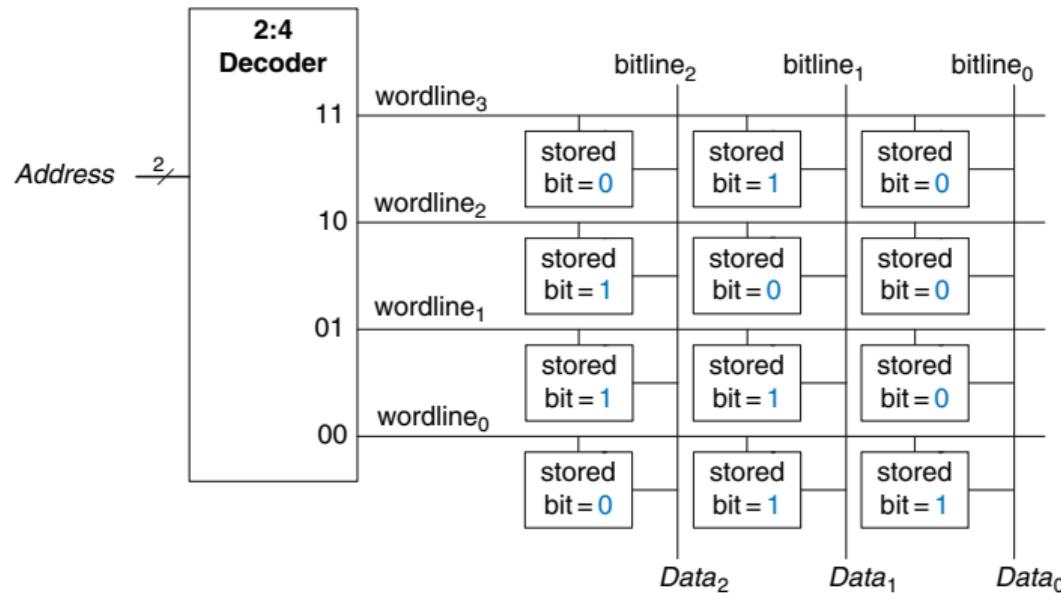


$$Address = A_1, A_0$$

$$Data_2 = A_1 \oplus A_0$$

$$Data_1 = \overline{A_1} + A_0$$

$$Data_0 = \overline{A_1} \overline{A_0}$$



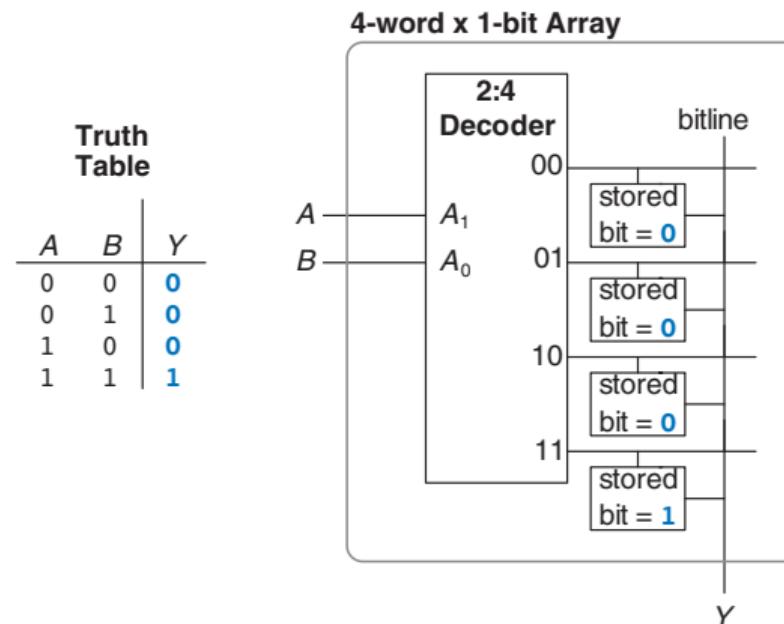
$$Address = A_1, A_0$$

$$Data_2 = A_1 \oplus A_0$$

$$Data_1 = \overline{A_1} + A_0$$

$$Data_0 = \overline{A_1} \overline{A_0}$$

Ausgabe bei jeder Eingangskombination (Adresse) nachschlagen

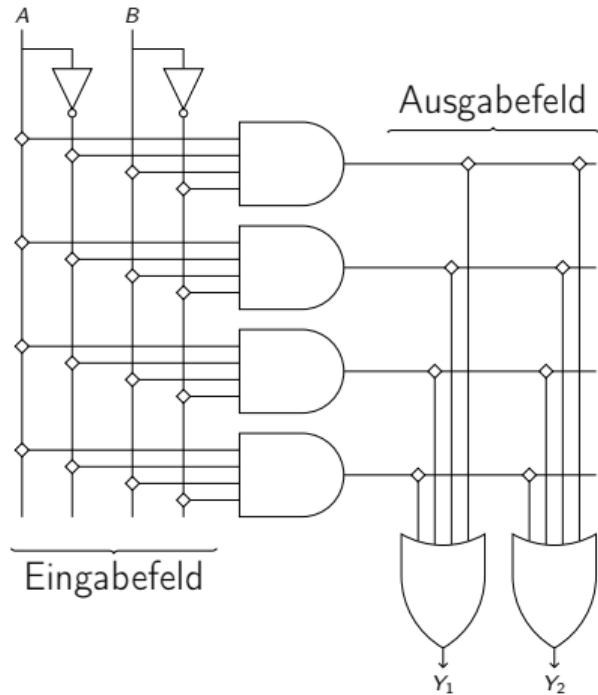


1 Speicherfelder

2 Logikfelder

Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätereiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

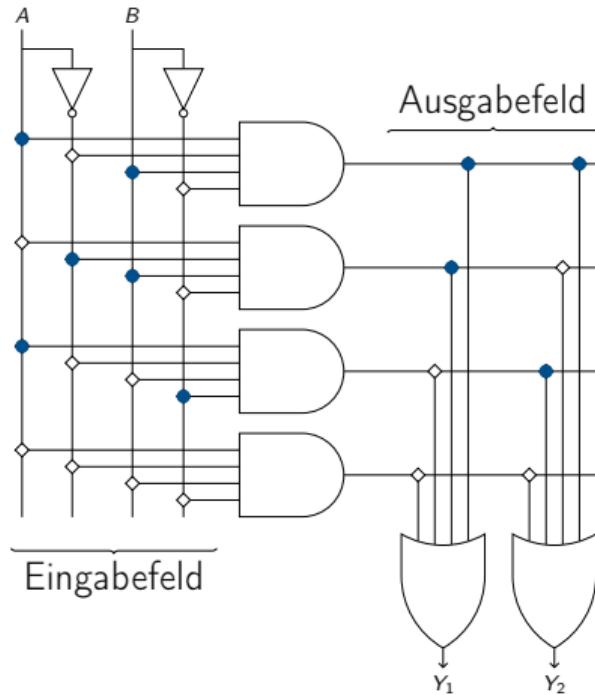
- realisiert einfache kombinatorische Logik via Sum-of-Products Form (DNF)
- zweistufige Logik mit programmierbaren Schaltern in Eingabefeld (links) und Ausgabefeld (rechts)
- Günstigere Varianten:
 - Programmable ROM:
nur Ausgabefeld programmierbar
 - Programmable Array Logic (PAL):
nur Eingabefeld programmierbar



Programmable Logic Array (PLA)

Beispiel

- $Y_1 = AB + \bar{A}\bar{B}$
- $Y_2 = A\bar{B} + A\bar{B}$



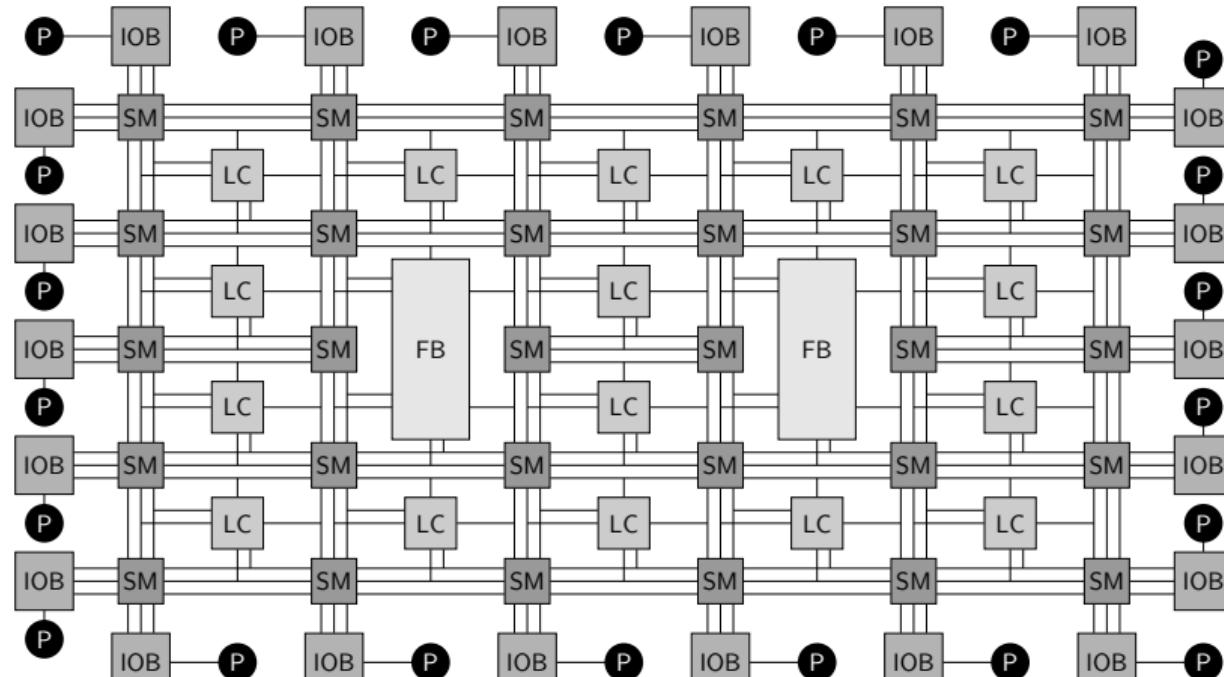
- Anwendungsspezifische integrierte Schaltung (ASIC, application-specific integrated circuit)
 - führt *für eine Anwendung* optimierte (parallele) Datenpfade aus
 - Basisgatterschaltungen (bspw. als CMOS) durch optische/chemische Prozesse auf Silizium-Wafer realisiert
 - ⇒ zur Laufzeit nicht an neue Anwendung anpassbar
- Software-Prozessor
 - führt generische Instruktionen sequentiell aus
 - nur generische (Mikro-)Architektur in Hardware realisiert
 - ⇒ zur Laufzeit durch Austauschen der Instruktionssequenz an neue Anwendung anpassbar
- ⇒ Field Programmable Gate Array (FPGA) vereint
 - Flexibilität von Software-Prozessoren („im Feld programmierbar“)
 - mit Performanz von ASICs (optimierte „Basisgatter-Schaltungen“)

- FPGAs verwenden feingranulare (bitweise) Konfigurationsspeicher statt wortweisen Instruktionsspeichern
- Konfigurationsspeicher realisiert mit verschiedenen Speicher-Technologien:
 - volatil (bspw. SRAM): schnell beschreibbar, benötigt aber permanente Spannungsversorgung (statische Leistungsaufnahme), oder
 - nicht-volatile (bspw. Flash): aufwendiger Schreibzugriff, aber Zustand bleibt auch ohne Spannungsversorgung erhalten

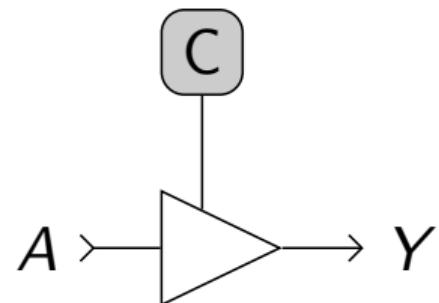
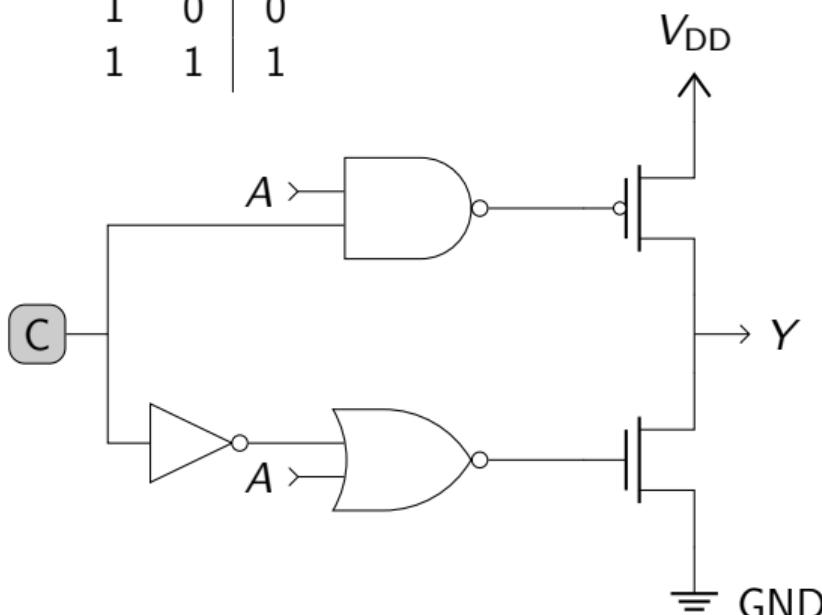
Field Programmable Gate Array (FPGA)



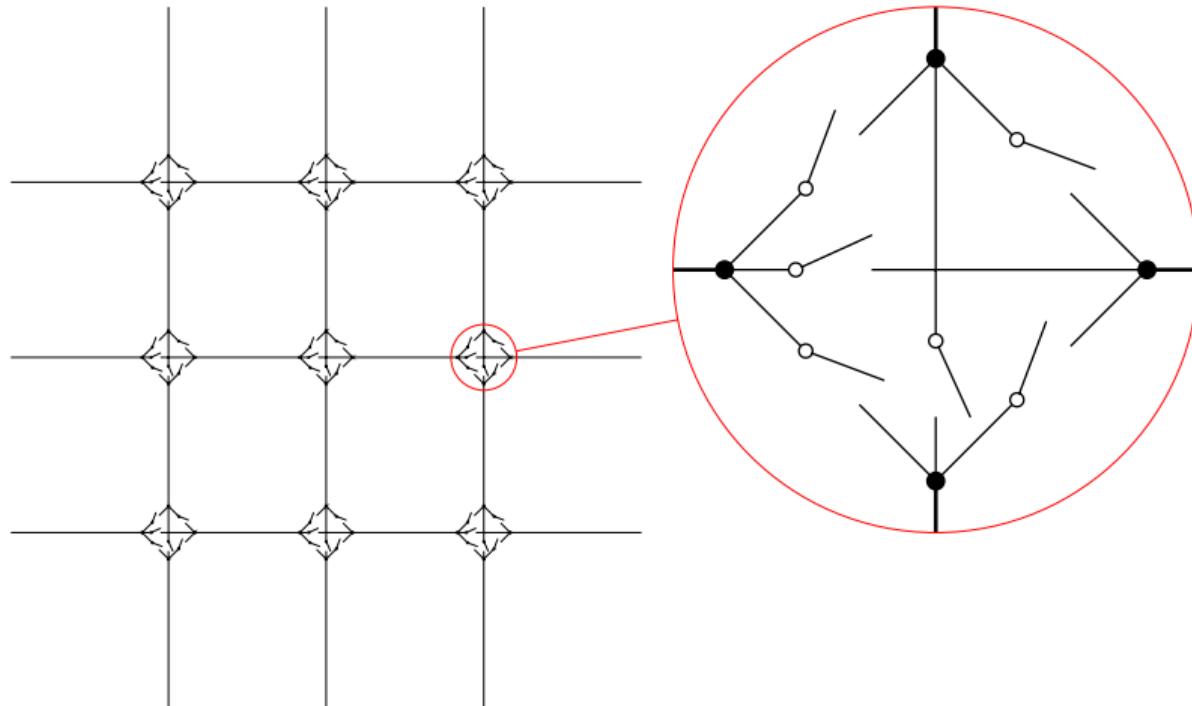
P: Pin, IOB: I/O Block, SM: Switch Matrix, LC: Logic Cell, FB: Function Block



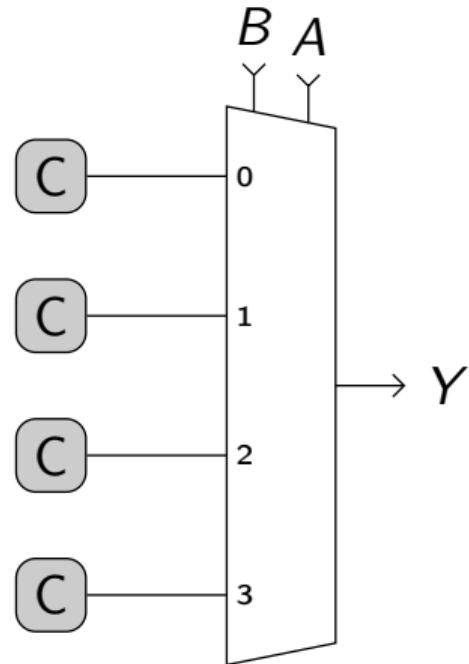
C	A	Y
0	*	Z
1	0	0
1	1	1



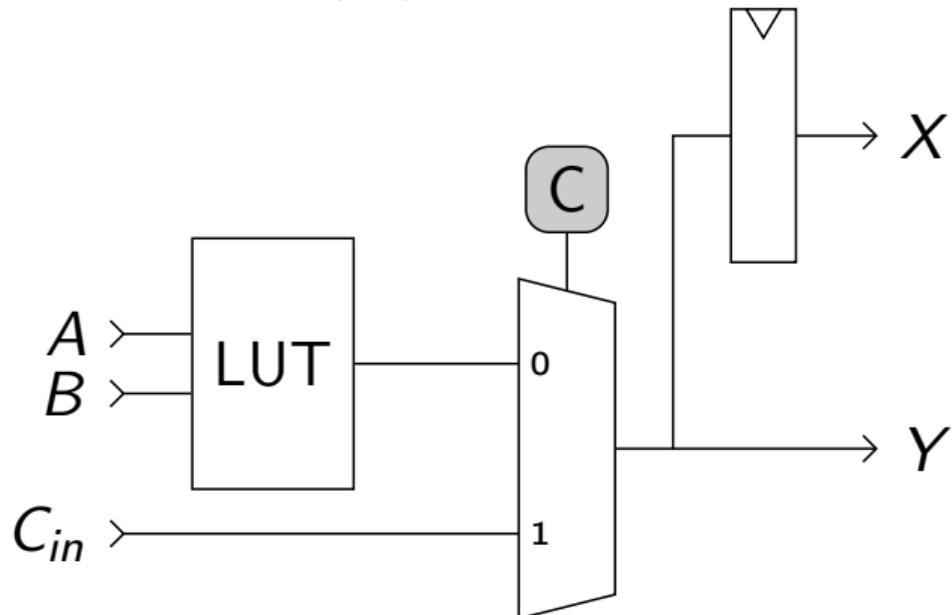
Realisiert durch programmierbare Schalter



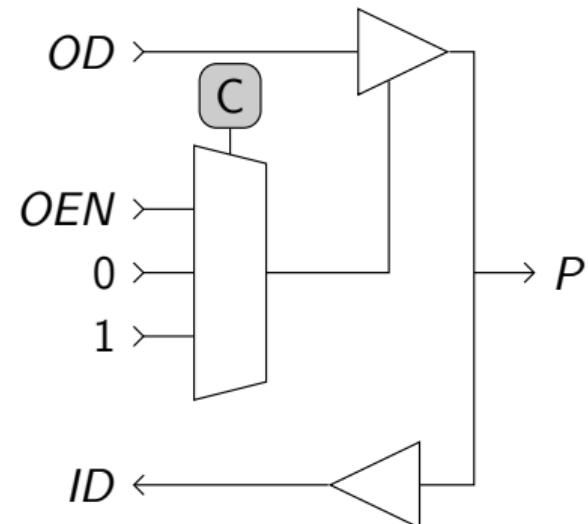
- realisiert kombinatorische Logik
- 2 bis 6 Eingänge (im Beispiel 2: B, A)
- häufig auch aufteilbar in kleinere LUTs
- Multiplexer (MUX)



- kann als kombinatorische Logik (Y) und/oder Speicher (X) verwendet werden
- häufig auch spezielle Carry In (C_{in}) für schnelle Arithmetik



- P wird mit physikalischen Pins verbunden
- Ausgabetreiber (OD) und Eingabetreiber (ID)
- Ausgabetreiber kann permanent oder zur Laufzeit steuerbar (durch OEN) aktiviert werden
- häufig auch weitere Konfigurationsmöglichkeiten:
 - Spannungs-Level
 - maximale Stromstärke



- häufig verwendete Logikbausteine als begrenzte Ressourcen verfügbar
 - Block RAM (BRAM): kleine SRAM Speicher (wenige Kilobit)
 - Digitale Signalverarbeitung (DSP): Multiplizierer, MAC
 - Phase-Locked Loop (PLL): Taktmodifikation
 - Kommunikations-Treiber (USART, USB, Ethernet)
 - kleine Prozessoren
 - ...

1 Speicherfelder

2 Logikfelder

nächste Vorlesung beinhaltet

- SystemVerilog Abschluss
- Vorzeichenbehaftete Binärzahlen: Vorzeichen und Betrag
- reelle Zahlen: Festkomma- und Gleitkommadarstellung

**Hausaufgabe E zu Vorlesungen 09 und 10 muss bis
diese Woche Freitag 23:59 abgegeben werden.
Wöchentliches Moodle-Quiz nicht vergessen!**

Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätertreiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen