Robert Limberger

Analogtechnik

- Kontinuierliche Signale

+ Multiplikation und Addition leicht zu realisieren
+ kleiner Flächenbedarf
+ sehr schnell
- temperaturabhängig
- nichtlineare Bauteile

↳ Sensoren, Messtechnik, HiFi-Technik, Rundfunktechnik

Digitaltechnik

- diskrete Signale

- feste Anzahl von k möglichen diskreten Zuständen
- $k=2$ → binär

+ unempfindlich gegenüber Störungen
+ einfacher Entwurf
+ beliebig hohe Präzision
- hoher Flächenbedarf
- höherer Energieverbrauch

↳ Computer, Bussysteme, Datenübertragung

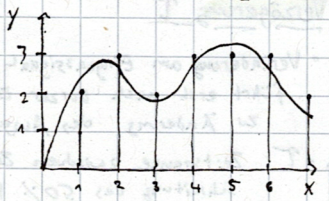
Analog-Digital-Wandlung: Analog → Digital

1) Quantisierung in der Zeit
- Abtasten in festen zeitlichen Abständen t
→ Abtastfrequenz $f = \frac{1}{T}$ muss mindestens das Doppelte der Signalfrequenz sein (Nyquist)

2) Wertmäßige Quantisierung
- Abbildung auf nächsten zulässigen Wert

→ Quantisierungsfehler sinkt bei Erhöhung der Zahl der möglichen diskreten Zustände.

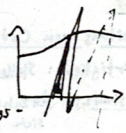
- Kodierung der diskreten Werte in bestimmter Wortbreite n
- Wortbreite n wird in Bit angegeben
- Bit: 0 oder 1
- Breite n Bit → 2^n verschiedene diskrete Werte



- Byte: $n=16$
- Halbwort: $n=32$
- Wort: $n=64$

Sägezahnverfahren

- startet im negativen Bereich
- Zähler startet bei 0V
- stoppt bei Eingangsspannung
- Abstand wird berechnet



Beispiel Hexadezimalzahl

$FE01_{16} = 15 \cdot 16^3 + 14 \cdot 16^2 + 1 \cdot 16^0 = 65040_{10}$

Umrechnung Zahlensysteme

Divisionsmethode

Bsp.: $x = 29_{10}$ → ins Binärsystem

29/2 = 14	Rest	1
14/2 = 7	Rest	0
7/2 = 3	Rest	1
3/2 = 1	Rest	1
1/2 = 0	Rest	1

Konvertierte Zahl sind die Reste von unten nach oben

$29_{10} = 11101_2$

Zahlen mit Komma

feste Zahl von k Nachkommastellen im Binärsystem → Dezimal

Bsp. $0110\,1,11\,0_2$
$= 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2}$
$= 13,75_{10}$

Dezimal → anderes System

Bsp. $13,75$

→ Zahl vor Komma wie gewohnt umformen

→ Nachkommazahl: Multiplikationsmethode

0,75	· 2	
1,5	→ 1	wieder Nachkommazahl mal zwei
0,5	· 2	
1	→ 1	

- Die Nachkommazahl immer mit zwei multiplizieren
- Die Zahl vor dem Komma ist dann die binäre Ziffer

↳ $13,75_{10} = 1101,11_2$

Einerkomplement (Negative Zahlen)

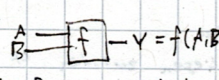
Bsp negative Zahlen 8-Bit

kleinste Zahl: $10000000_2 = -127_{10}$
größte negative Zahl: $11111110_2 = -1_{10}$
größte Zahl: $01111111_2 = 127_{10}$

- bei den negativen Zahlen:
→ startet mit 1
→ die Nullen sind die "Einsen"
- bei positiven Zahlen:
→ startet mit 0
→ die Einsen werden ganz normal als Einsen interpretiert

Gatter

- "Black Box"
- beliebig viele Eingänge
- genau ein Ausgang
- realisiert Funktion, z.B. $Y = f(A,B,(...)$



- Die Funktion wird durch eine Wahrheitstabelle festgelegt

Häufige Gatter

und-Gatter:	&	∧
oder-Gatter:	≥1	∨
nicht-Gatter:	1	¬
NAND-Gatter:	&	
NOR-Gatter:	≥1	
XOR-Gatter:	=1	

A B	Y
0 0	1
0 1	1
1 0	1
1 1	0

A B	Y
0 0	1
0 1	0
1 0	0
1 1	0

A B	Y
0 0	0
0 1	1
1 0	1
1 1	0

DNF und KNF

DNF:
- es werden die Felder mit (1) betrachtet

z.B.

A	B	C	Y
0	1	0	1

→ in DNF: $Y = \bar{A} \wedge B \wedge \bar{C}$

Verknüpfung der Variablen mit ∧

KNF:
- es werden Felder mit (0) betrachtet
- anders als bei DNF → Felder bei einzelnen Variablen mit 0
↳ Die Variable ist 'true'

z.B.

A	B	C	Y
0	1	0	0

→ in KNF: $A \vee \bar{B} \vee C$

Variablenverknüpfung durch ∨

Minterm

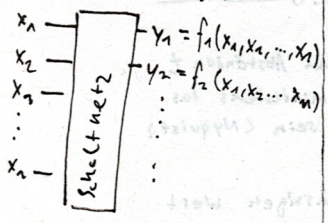
Term, in dem jede Variable einer boolschen Funktion genau einmal vorkommt, verbunden mit ∧
z.B. $x \wedge \bar{y} \wedge z$ von $f(x,y,z)$

Maxterm

→ das selbe aber mit ∨
z.B. $x \vee \bar{y} \vee z$ von $f(x,y,z)$

Schaltnetze

- kombinatorische Logik



Multiplexer

- mehrere Eingangssignale auf einzigen Ausgang
- z.B. Auswahl einer Datenquelle

Demultiplexer

- einzelner Eingang aber mehrere Ausgänge
- z.B Auswahl Datensenke (z.B. Speicherzeile)

Codes

Bewertung von Codes

- Wertigkeit: Stellenwert für jede Position?
- Distanz: Anzahl Bitstellen, in denen sich zwei Codewörter unterscheiden
- Hamming-Distanz: Minimum aller Abstände zwischen Wörtern innerhalb eines Codes
- Stetigkeit: Distanz zwischen benachbarten Codewörtern ist konstant
- Redundanz: Anteil (in Bit) einer Nachricht, die keine Informationen enthält

$R = \log_2(N/M)$

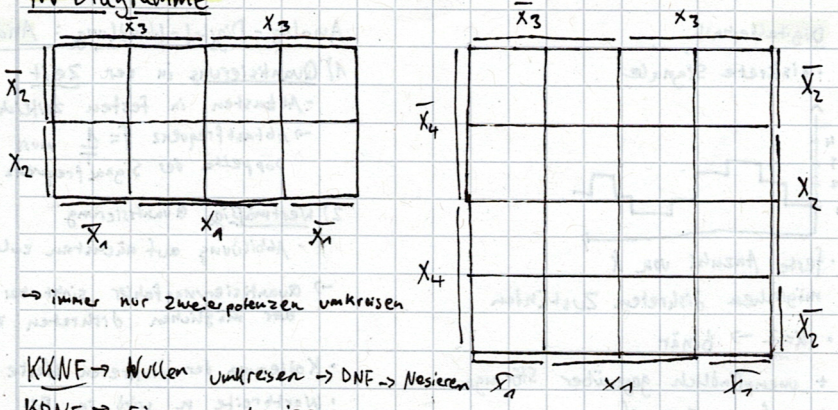
N: mögliche Anzahl mit n Bit kodierbare Zeichen

M: tatsächliche Anzahl der verwendeten Zeichen

Beispiel: Bewertung Dual-Code (Binär)

- Wertigkeit: Stelle i hat Wichtung 2^i
- Abstand: minimal 1, maximal n
- Stetigkeit: nicht stetig
- Hamming-Distanz: H = 1
- Redundanz: R = 0 Bit

KV-Diagramme



→ immer nur Zweierpotenzen umkreisen

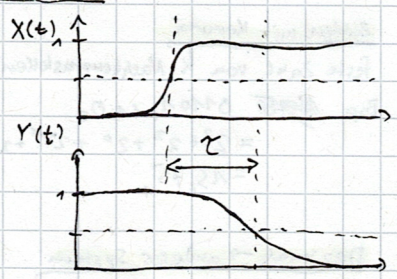
KKNF → Nullen umkreisen → DNF → Negieren

KDNF → Einsen umkreisen

Verzögerung τ

- Veränderung am Eingangssignal führt erst nach kurzer Zeit zu Änderung des Ausgangssignals
- τ: Zeitspanne zwischen Zeitpunkten der Überschreitung des 50% Pegels des realen Signals

Bsp. Inverter



Strukturhazard: durch Verzögerung im Gatter können kurzzeitig falsche Ausgangssignale auftreten

↳ Schaltung ändern, dass:

- alle Pfade im Schaltnetz gleich lang sind
- ~~Entfernen~~ Einfügen zusätzlicher Logikgatter

BCD-Code

- Jede Ziffer wird binär kodiert
  - 4 Bit je Ziffer nötig
  - Beispiel: $9128_{10} = 1001\ 0001\ 0010\ 1000_{BCD}$
  - übrige Codes (Pseudotetraden) kommen nach 9 (1010, 1011, ...)

Aiken-Code

- Wertigkeit der ~~Bits~~ 4 Bits: 2 4 2 1
- z.B. $5_{10} = 1011_{Aiken}$
- Die 6 Pseudotetraden liegen zwischen 4 und 5
- Taschenrechner und Digitaluhren
- Beispiel: $9128_{10} = 1111\ 0001\ 0010\ 1110_{Aiken}$

Gray-Code

- Distanz benachbarter Codewörter = 1

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Fehlererkennende Codes

- z.B. im Dual-Code ein weiteres Bit
  - einzelne Bitfehler bei Übertragung erkennen
  - zwei Bitfehler können nicht erkannt werden

Beispiel: Ergänzung Dual-Code um Bit, das jedes Codewort auf gerade Anzahl von Einsen ergänzt

0	0000	0
1	0001	1
2	0010	1
3	0011	0
⋮		

- ergänzes Bit: Paritätsbit
- Hamming Distanz: H = 2

- es können auch mehrere Prüfbits hinzugefügt werden:
  - z.B. jeweils Ergänzung von Bitgruppen zu gerader / ungerader Parität
    - Hammins Distanz erhöht
    - Erkennung mehrerer Bitfehler
    - Bitfehler ggf. sogar korrigiert

1. Code mit Hamming Distanz h kann (h-1) Bitfehler erkennen
2. kann (h-1)/2 Bitfehler korrigieren

Digitaltechnik — Sequentielle Logik

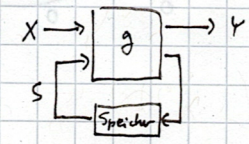
Vergleich

Kombinatorische Logik
- Gatter werden als Verzögerungsfrei angenommen
- Schaltungen werden als Schaltnetze betrachtet
- Rückkopplungen nicht gestattet
- Schaltungen: gerichtet azyklischer Graph

Sequentielle Logik
- Zustandsspeicher erforderlich
- Flipflops → Grundelemente
- Rückkopplungen gestattet
- Schaltungen als Schaltwerke betrachtet
- Schaltungen: gerichtet zyklischer Graph

Schaltnetz
- realisiert Schaltfunktion

Schaltwerk
- Ausgabe hängt zusätzlich von Zustand S ab

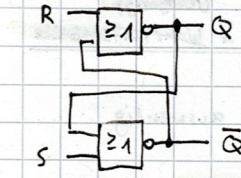


Asynchrones Schaltwerk
- gesteuert durch Veränderung der Eingangssignale
- aufwendig intuitiver Entwurf
- sehr schnelle Arbeitsweise möglich

Synchrones Schaltwerk
- gesteuert durch zentralen Takt
- Übernahme der Änderungen am Eingangssignal erfolgt nur zu festen Zeitpunkten
- einfacher systematischer Entwurf (Automaten)
- kritischer Pfad bestimmt maximale Taktfrequenz

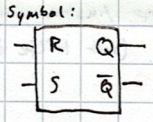
Flip-Flops

RS Flip-Flop
mit NOR-Gattern:



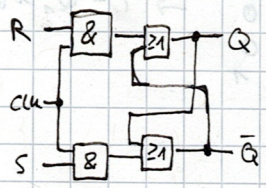
R	S	Q' (Folgezustand)
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

Symbol:



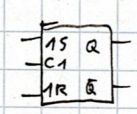
- "R" = "Reset"
- "S" = "Set"

Getaktetes RS Flip-Flop
- R und S werden nur übernommen, wenn Clk = 1

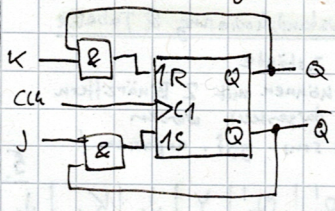


R	S	Clk	Q'
d	d	0	Q
0	0	1	Q
0	1	1	1
1	0	1	0
1	1	1	nicht erlaubt

Symbol:

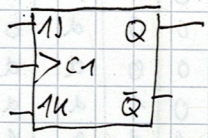


JK Flip-Flop
- 1,1 für Invertierung von Q (Toggle)

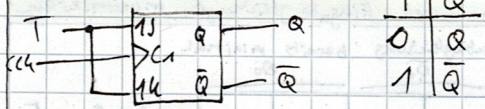


J	K	Q'
0	0	Q
0	1	0
1	0	1
1	1	Q̄

Symbol (positive Flankentriggerung):

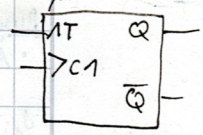


T Flip-Flop
- Modifikation JK Flipflop; J = T = S = K
- T = 1 ⇒ Toggle, wechselt in jedem Takt den Zustand

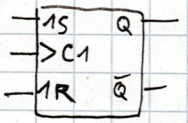


T	Q
0	Q
1	Q̄

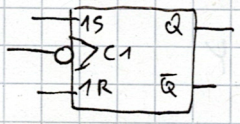
Symbol:



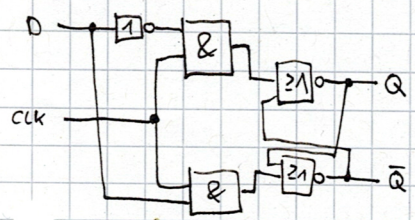
Flankengetriggerte Flip-Flops
- bei steigender Flanke (positiv)



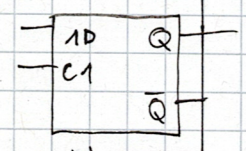
- bei fallender Flanke (negativ)



D Flip-Flop
- bei Clk = 1: S = D R = D̄
- unerlaubter Zustand wird vermieden

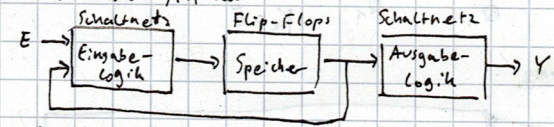


D	Clk	Q'
0	0	Q
0	1	0
1	0	Q
1	1	1



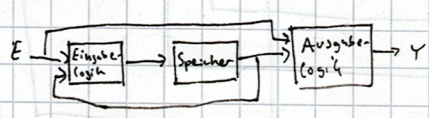
- wird meist flankengetriggert verwendet

Moore-Automat
- allgemeiner Aufbau



- Eingabelogik: bestimmt Zustandsübergänge:
  - abhängig von Eingabesignalen und
  - aktuellem Zustand
- Ausgabelogik:
  - hängt nur vom aktuellen Zustand ab

Mealy-Automat



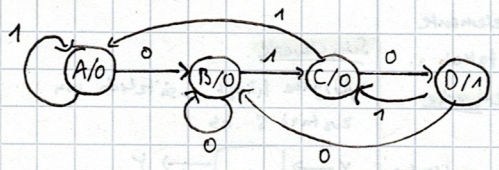
- Ausgabelogik hängt auch vom Eingabesignal E ab
- kann schneller auf Eingabeänderungen reagieren

Vorgehensweise Entwurf eines Schaltwerks
1. Zustandsdiagram
2. Zustandstabelle
3. Auswahl binäre Zustandskodierung & binäre Zustandstabelle
4. Auswahl Flip-Flop & Flip-Flop Ansteuerungen
5. Ausgabegleichung
6. Minimierung Ansteuer- und Ausgabegleichung
7. Realisierung

Entwicklung eines Automaten am Beispiel Moore-Automat

Aufgabe: Sequenz 010 erkennen

1. Zustandsdiagramm



2. Zustandstabelle

S	E	S'	Y
A	0	B	0
A	1	A	0
B	0	B	0
B	1	C	0
C	0	D	0
C	1	A	0
D	0	B	1
D	1	C	1

- für jeden Zustand S (A,B,C...) wird geklärt was ein Folgezustand kommt (für alle möglichen Eingaben (0,1))
- + zugehörige Ausgabe Y

3. Auswahl binäre Zustandskodierung & Tabelle

- in dem Fall: 4 Zustände
  - ↳ können mit 2 Binärziffern dargestellt werden
  - ↳ sonst ggf. anpassen!

	Q_1	Q_0	E	Q_1'	Q_0'	Y	J_1	K_1	J_0	K_0
A	0	0	0	0	1	0	0	d	1	d
	0	0	1	0	0	0	0	d	0	d
B	0	1	0	0	1	0	0	d	d	0
	0	1	1	1	0	0	1	d	d	1
C	1	0	0	1	1	0	d	0	1	d
	1	0	1	0	0	0	d	1	0	d
D	1	1	0	0	1	1	d	1	d	0
	1	1	1	1	0	1	d	0	d	1

4. Auswahl Flipflop (immer JK) & Übergangs-Wahrheitstabelle

Q	Q'	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

→ auswendig

5. Ermittlung Flip-Flop Ansteuerungen

(mit Hilfe der Übergangstabelle)

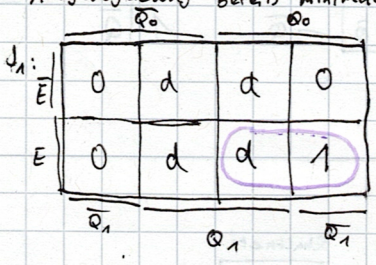
6. Bestimmen Ausgabegleichung

Q_1	Q_0	Y
0	0	0
0	1	0
1	0	0
1	1	1

→ $Y = Q_1 \wedge Q_0$

Ausgabe- und Eingabegleichungen minimieren

- Ausgabegleichung bereits minimal



→ $J_1 = Q_0 \wedge E$

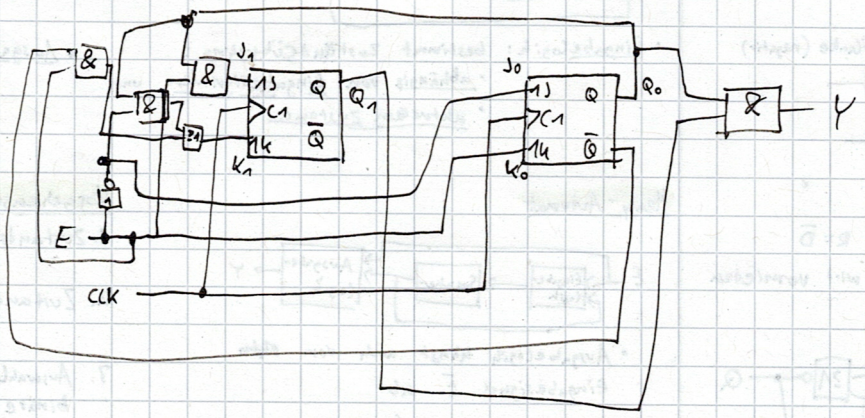
- selbes Verfahren für alle J und K

$K_1 = (Q_0 \wedge \bar{E}) \vee (\bar{Q}_0 \wedge E)$

$J_0 = \bar{E}$

$K_0 = \bar{E}$

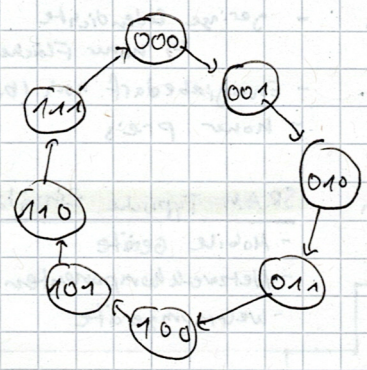
7. Realisierung



Zähler

Synchroner 3-Bit Binärzähler

1. Zustandsdiagramm mit allen möglichen Zuständen & Zustandsübergängen



2. abgeleitete Übergangstabelle für JK-FF

Q	Q'	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

→ 3 Binärziffern, also auch 3 Q und 3 Flip-Flops

3. Zustandsübergangstabelle

Q_2	Q_1	Q_0	Q_2'	Q_1'	Q_0'	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	d	0	d	1	d
0	0	1	0	1	0	0	d	1	d	d	1
⋮											
⋮	etc.										

→ aktueller Zustand stellt auch Ausgabe dar

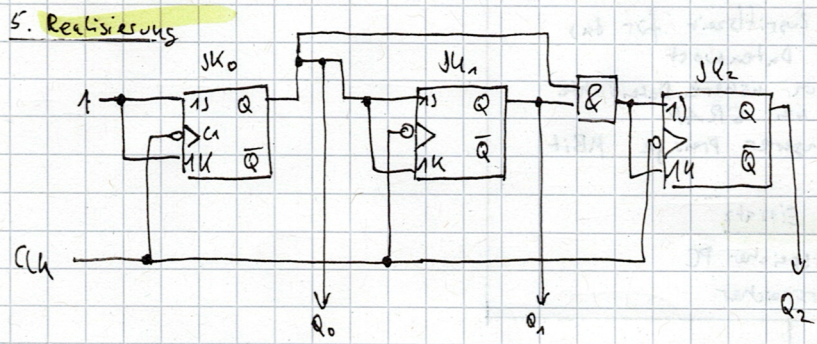
4. Gleichungen für $J_2, K_2, J_1, \dots$ erstellen

- Mit KV-Diagramm oder ablesen

Beispielhaft: $J_2 = Q_1 \wedge Q_0$ $J_1 = Q_0$ $J_0 = 1$

$K_2 = Q_1 \wedge Q_0$ $K_1 = Q_0$ $K_0 = 1$

5. Realisierung

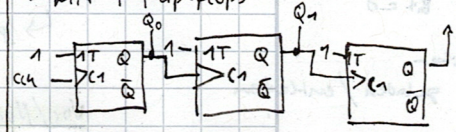


Asynchroner Zähler

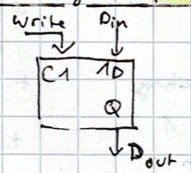
- kein synchroner Takt
- statt dessen Flip-Flop Ausgänge als Takt

Frequenzteiler

- Ausgangssignale $Q_0, Q_1, Q_2, \dots$ mit 1/2, 1/4, 1/8, ... der Taktfrequenz von Clk
- mit T-Flip-Flops



Grundlegende Idee RAM



- bei Write = 1 wird Information von Din gespeichert und steht in Dout zur Verfügung
- Information bleibt gespeichert, auch wenn bei Write = 0 Din geändert wird

RAM

- flüchtiger Speicher:
  - Informationen gehen nach Ausschalten der Versorgungsspannung verloren

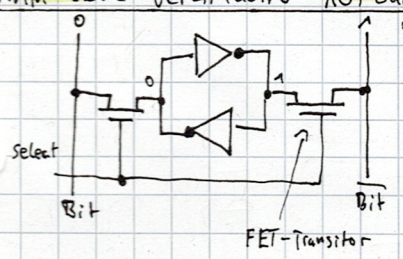
- **statischer Speicher**
  - Speicherung durch 4–6 Transistoren pro Bit
  - kein Refresh notwendig
  - Bausteine: SRAM
  - Zugriffs- und Zykluszeit: ca. 10ns

- **dynamischer Speicher**
  - Speicherung durch Kondensator und Transistor je Bit
  - Refresh notwendig (Kondensator verliert Ladung)
  - hohe Datendichte möglich
  - Bausteine: DRAM, SDRAM, DDR-SDRAM
  - Zugriffszeit: ca. 40ns ; Zykluszeit: ca. 60ns

- **Zugriffszeit t_{ac}**
  - Zeitspanne von Anlegen einer Adresse bis zur Gültigkeit der ausgelesenen Daten

- **Zykluszeit t_{cycle}**
  - Zeitspanne vom Anlegen einer Adresse bis zum möglichen Anlegen der nächsten Adresse
  - ↳ Zykluszeit oft größer als Zugriffszeit

SRAM-Zelle vereinfachter Aufbau



← hier wurde 0 für Bit und 1 für $\overline{Bit}$ angelegt:
- zum Schreiben einer 0
- Zustand bleibt erhalten

- bei Select = 1 → FETs leiten → bistabile Kippstufe
  Select = 0 → FETs sperren

SRAM Auslesen

- Bitleitungen: Bit = 1 ; $\overline{Bit}$ = 1 (precharging)
- Select: kurzer Impuls
- geringer Spannungsabfall auf Bit oder $\overline{Bit}$ wird erkannt
- Verstärker gibt entsprechende Ausgabe

RAM-Fortsetzung

SRAM Organisation

- Wortorganisierter Speicher (je Speicherzeile ein n-Bit Wort)
- m x n Speichermatrix (m Speicherzeilen mit je n SRAM-Zellen)
- Auswahl einer der Zeilen

SRAM Steuerlogik

- $\overline{CS}$ (Chip Select):
  - Auswahl und Aktivierung eines SRAM Bausteins
- $\overline{WE}$ (Write Enable):
  - Zum Speichern eines Wertes
- $\overline{OE}$ (Output Enable):
  - Lesen und Freischalten der Ausgänge

Vor/Nachteile SRAM

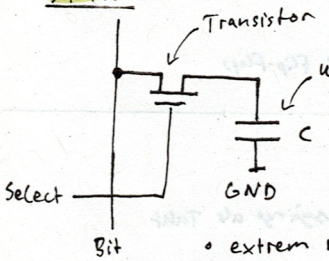
+ schneller Zugriff
+ unempfindlich gegen elektromag. Strahlung
- geringe Datendichte
  ↳ hoher Flächenbedarf
- Energiebedarf hoch (bei vielen Zugriffen)
- hoher Preis

SRAM Typische Einsätze

- Mobile Geräte
- Netzwerkkomponenten
- Weltraumgeräte

DRAM

Aufbau



- extrem hohe Speicherkapazität durch geringen Flächenbedarf

- Speichermatrix ist bitorganisiert
- Kapazität C des Kondensators einer DRAM-Zelle ist sehr gering
- Ladungsverlust:
  - beim Lesen
  - mit der Zeit
  - durch elektromag. Strahlung

  ↳ periodischer **Refresh**

Schreiben

- Bit = 1 oder Bit = 0
- Select = 1
- Transistor leitet
- Kondensator geladen / entladen

Auslesen

- Select = 1
- Transistor leitet
- falls Kondensator geladen war
  ↳ kurzer Impuls auf Bit-Leitung
- Leseverstärker erzeugt logische 1
- Kondensator wird beim Auslesen entladen

  ↳ muss neu beschrieben werden

Vor/Nachteile

- periodischer Refresh notwendig
  ↳ hoher Energie
- hohe Zugriffszeit für das erste Datenwort
+ 4 fach höhere Datendichte als bei SRAM
+ geringerer Preis je MBit

Typischer Einsatz

- Hauptspeicher PC
- Pufferspeicher

Diode

- Strom kann nur in eine Richtung fließen

Transistor

- leitet nur wenn Basisstrom fließt

Wandlung Digital → Analog

- nur möglich, wenn jede Bitstelle feste Wertigkeit hat (Binärcode √, Gray-code ✗)
- 1 Stufe entspricht 1 LSB (Least Significant Bit)
- ggf. anschließende Glättung durch Filter

→ gestuftes Widerstandsnetzwerk