

6 DIGITALE SCHALTUNGSTECHNIK

Um Daten zu verarbeiten, verwenden Computer als grundlegende Größen logische Variablen, die genau zwei Zustände annehmen können, nämlich den Wert „0“ und den Wert „1“. Man kann dies auch mit den Werten „ja“ und „nein“ oder „wahr“ und „falsch“ oder „ein“ und „aus“ interpretieren. In jedem Fall beinhaltet diese Variable eine so genannte „Elementarentscheidung“, die genau einen der beiden oben genannten Werte annehmen kann. Eine solche Elementarentscheidung wird als Bit (engl. Abkürzung von **binary digit**) bezeichnet.

Die technische Umsetzung dieser beiden logischen Werte „0“ und „1“ erfolgt in elektrischen Bauteilen durch elektrische Spannungen U . Dies geschieht zunächst dadurch, dass man die Pegel $U_{L(\text{ow})}$ und $U_{H(\text{igh})}$ festlegt, wobei gilt $U_L < U_H$.

Gilt nun für eine Spannung $U < U_L$ dann befindet sich U im Zustand L (low), gilt $U > U_H$, dann befindet sich U im Zustand H (high). Nun ordnet man den beiden Zuständen L und H eine der logischen Variablen „0“ (im Folgenden auch mit log.0 abgekürzt) und „1“ (log.1) zu. Es gibt zwei Möglichkeiten:

log.	Positive Logik	Negative Logik
0	L	H
1	H	L

Alle Aktionen die ein Mikroprozessor ausführt lassen sich auf diese beiden Elementarzustände und auf die drei logischen Funktionen **NOT**, **AND** und **OR** zurückführen. Tabelle 6-1 zeigt Wahrheitstabellen für die 3 genannten logischen Funktionen:

Tabelle 6-1 Wahrheitstabellen für NOT, AND und OR-Verknüpfung.

NOT		AND			OR		
In	Out	In1	In2	Out	In1	In2	Out
0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1
		1	0	0	1	0	1
		1	1	1	1	1	1

Diese drei Elementarfunktionen können mit elektronischen Bauteilen realisiert und auch miteinander kombiniert werden. Sehr häufig findet man folgende Logikgatter: **Inverter** (=NOT-Schaltung), **NAND-Gatter** (=NOT AND) und **NOR-Gatter** (=NOT OR).

Je nach Art der Bauteile die man hierfür verwendet, kann man verschiedene **Logikfamilien** definieren. Gängige Logikfamilien sind:

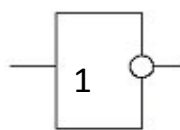
- RTL-Logik: Resistor-Transistor-Logic. Aus Widerständen und Transistoren aufgebaut.
- DTL-Logik: Diode-Transistor-Logic. Aus Dioden und Transistoren aufgebaut
- TTL-Logik: Transistor-Transistor-Logic. Dioden aus DTL werden durch Transistoren ersetzt. Es werden bipolare Transistoren verwendet.

- CMOS-Logik: Complementary MOS-Logik: wird aus komplementären n-MOS und p-MOS-FETs aufgebaut.

Heute wird fast ausschließlich die moderne CMOS-Technologie verwendet – insbesondere bei integrierten Schaltungen, jedoch findet auch die ältere TTL-Logik noch Anwendung. Wir betrachten nun exemplarisch den Aufbau des Inverters – NAND und NOR-Gatter sind in optionalen Kapiteln beschrieben.

6.1 DER INVERTER

Der Inverter ist die technische Realisierung einer NOT-Verknüpfung. Er ist die wahrscheinlich wichtigste Schaltung der Digitaltechnik und dient dazu einen Schaltzustand zu invertieren. Der Inverter hat einen Eingang (In) für das Eingangssignal und einen Ausgang (Out) für das Ausgangssignal. Abbildung 6-1 zeigt das Schaltzeichen und die Wahrheitstabelle des Inverters:



In	Out
0	1
1	0

Abbildung 6-1 Schaltzeichen (links) und Wahrheitstabelle (rechts) des Inverters.

Wir wollen nun näher betrachten, wie man ein solches Logikgatter technisch realisieren kann. Wie schon beschrieben realisiert man die logischen Werte 0 und 1 durch Spannungen, die in diesem Fall als Pegel bezeichnet werden. Nehmen wir an wir ordnen der logischen 0 den (idealen) Pegel 0V und der logischen 1 den (idealen) Pegel 5 V zu. Da man gewisse Toleranzen zulassen muss, können wir weiterhin z. B. definieren, dass Spannungen $> U_H=2,0V$ als logisch 1 zu betrachten sind und Spannungen $< U_L=0,8 V$ als logisch 0 zu betrachten sind. Eine Möglichkeit einen Inverter zu realisieren ist nun die in

Abbildung 6-2 dargestellte:

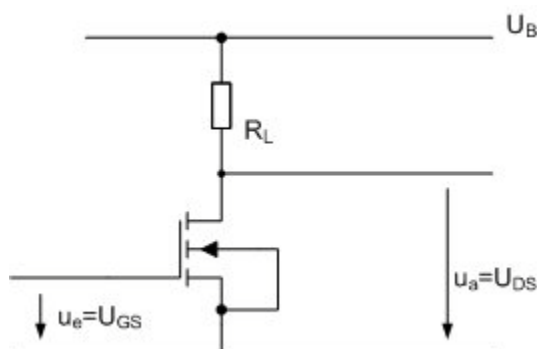


Abbildung 6-2: Grundschaltung eines Inverters mit n-MOS-Transistor

Die Schaltung wird mit der Betriebsspannung U_B (z. B. $U_B=5V$) betrieben. Es wird ein n-MOS-FET vom Anreicherungstyp verwendet und der Lastwiderstand R_L dient u. a. zur Strombegrenzung. Nun wird eine Eingangsspannung u_e an das Gate des n-MOS-FET angelegt (U_{GS}). Ist die Spannung $u_e=0$, bzw. sehr klein ($U_{GS}<U_{th}$), was einer logischen 0 entspricht, dann sperrt der Transistor (es fließt kein Strom) zwischen Source und Drain. Daher beträgt der Spannungsabfall am Lastwiderstand 0V, somit erhält man $u_a=5V$, was in der Tat logisch 1 entspricht. Etwas problematischer liegt der Fall allerdings, wenn man an das Gate eine

Spannung $u_e > 2,0V$ anlegt (logisch 1). Dann wird der Transistor durchlässig, d. h. sein Widerstand wird sehr gering und damit wird auch die Spannung u_a auf jeden Fall kleiner als 5,0 V. Allerdings ist die Spannung nun von der Stromstärke I_D und dem Lastwiderstand R_L abhängig. Es gilt:

$$U_{DS} = U_B - I_D \cdot R_L$$

Es ist jedoch nicht ganz trivial den Lastwiderstand so zu konfigurieren, dass der Spannungsabfall $U_{DS} < U_L$ ist. Daher ist dies zwar eine Möglichkeit einen Inverter zu realisieren, jedoch nicht die beste. Ein weiterer Grund dafür, dass diese Schaltung nicht ideal ist, liegt im Strom I_D selbst, denn sobald ein Strom fließt erhält man natürlich eine Verlustleistung $P = U_B \cdot I_D$, die zur Erwärmung der Schaltung führt. Abgesehen davon ist der benötigte Lastwiderstand von seiner Ausdehnung her fast 30 Mal so groß, wie der Transistor, was sich extrem ungünstig in der Realisierung von integrierten Schaltungen in Chips auswirkt.

Ideal wäre es also, wenn – auch wenn der Transistor durchschaltet – kein Strom in der Schaltung fließt und außerdem der Widerstand eliminiert werden könnte.

Dies erreicht man durch **die CMOS(Complimentary MOS)-Technik**. Man ersetzt hier einfach den Lastwiderstand R_L durch einen komplementären MOS-FET. Komplementär bedeutet hier: wenn man in der ursprünglichen Schaltung einen n-MOS-FET verwendet hat, ersetzt man den Widerstand durch einen p-MOS-FET, bzw. umgekehrt. Unser Inverter sieht dann so aus:

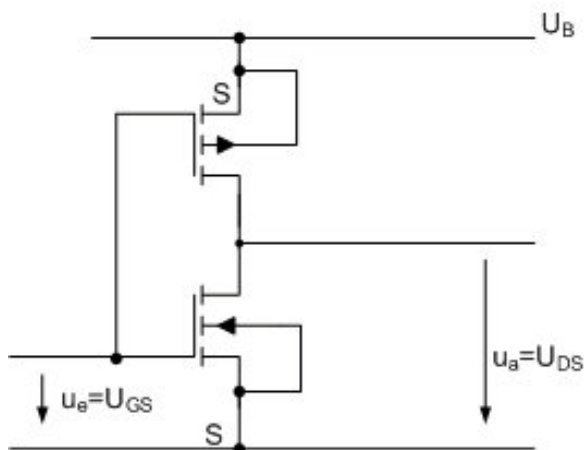


Abbildung 6-3 Inverter in CMOS-Technologie

Der obere FET ist ein p-FET und ersetzt den Lastwiderstand. Zu beachten ist hierbei, dass die Source (S) des p-FETs an die Betriebsspannung U_B angeschlossen ist, und das Gate auf dem Potential u_e liegt. Dies hat folgende Auswirkungen, die hier tabellarisch aufgelistet sind ($U_B = 5V$):

u_e	U_{GS}		Schaltzustand MOS		U_{DS}		u_a
	n-MOS	p-MOS	n-MOS	p-MOS	n-MOS	p-MOS	
0V (log. 0)	0V	-5V	sperrt	durchlässig	5V	0V	5V (log.1)
5V (log. 1)	+5V	0V	durchlässig	sperrt	0V	5V	0V (log.0)

Da immer einer der Transistoren sperrt kann in beiden Schaltzuständen kein Strom fließen und es entsteht keine Verlustleistung. Lediglich während des Umschaltvorgangs fließt kurzzeitig ein Strom, weil hier beide Transistoren gleichzeitig leiten. Außerdem hat jeder MOS-FET eine Kapazität (ähnlich wie ein Kondensator) C_S , und beim Auf- und Entladen fließt ein Strom.

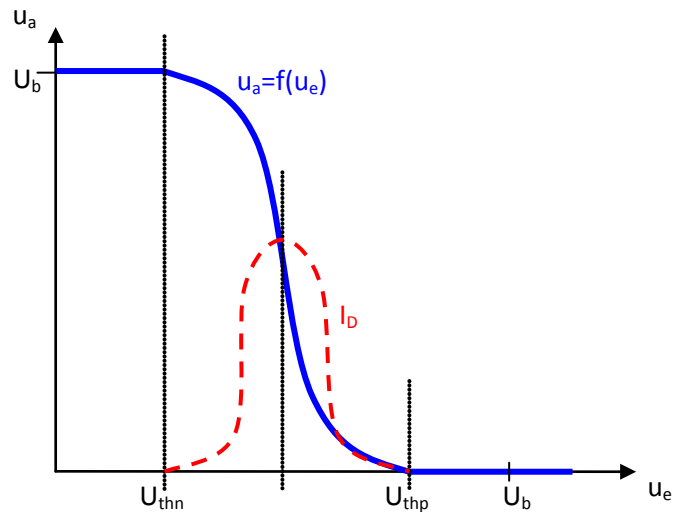


Abbildung 6-4 Übertragungskennlinie des CMOS-Inverters

Die Verlustleistung der CMOS-Schaltung berechnet sich wie folgt:

Bei jedem Schaltvorgang wird der Energiebeitrag

$$w = Q \cdot U = C_S \cdot U_B^2 \quad (6.1-1)$$

in Wärme umgesetzt. Die Verlustleistung P_V ist diese Energie pro Zeiteinheit. So erhält man:

$$P_V = \frac{w}{T} = w \cdot f = C_S \cdot U_B^2 \cdot f \quad (6.1-2)$$

mit $f = \frac{1}{T}$ der Taktfrequenz des Systems in dem der Inverter arbeitet.

Beispiel: der CMOS-Inverter 74HC04 hat eine Verlustleistung von $P_V = 0,5 \frac{\mu W}{kHz} \cdot f$.

6.2 NAND- UND NOR-GATTER

Neben dem Inverter gibt es noch zwei wichtige Logikschaltungen, nämlich das NAND- und das NOR-Gatter. Hierbei hat das NAND-Gatter eine etwas größere Bedeutung, das es sich technisch günstiger Produzieren lässt und man ein NOR-Gatter aus NAND-Gattern aufbauen kann. Im Folgenden werden beide Typen kurz beschrieben.

Das NAND-Gatter:

Abbildung 6-5 zeigt das Schaltsymbol und die Wahrheitstabelle des NAND-Gatters:

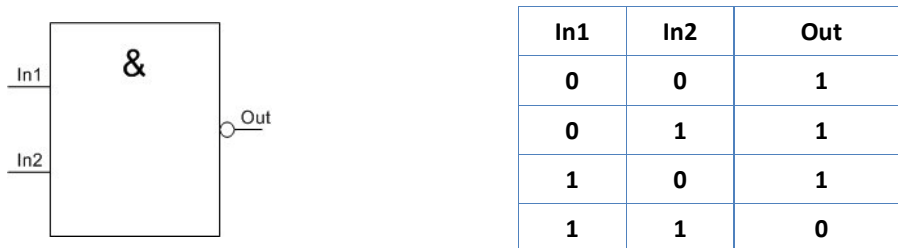


Abbildung 6-5 Schaltsymbol (links) und Wahrheitstabelle (rechts) eines NAND-Gatters

Man kann ein solches Gatter folgenderweise aus CMOS-Transistoren aufbauen:

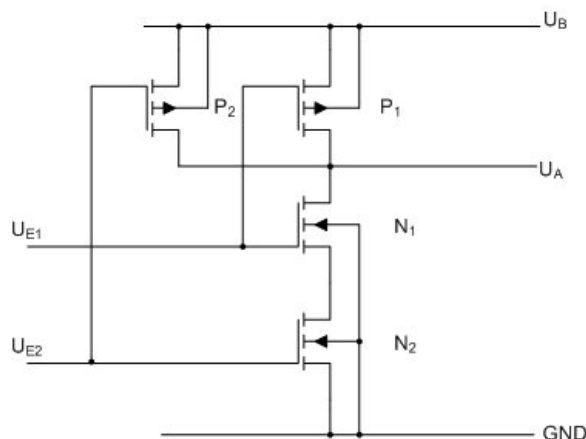


Abbildung 6-6 Aufbau eines NAND-Gatters in CMOS-Technologie

Es besteht aus den beiden n-MOS-Transistoren N_1 und N_2 und den beiden p-MOS-Transistoren P_1 und P_2 . Die Eingänge sind mit U_{E1} und U_{E2} bezeichnet, das Ausgangssignal ist die Spannung U_A . U_B ist die Betriebsspannung des Gatters. Das Verhalten der Transistoren und des gesamten Gatters ist wiederum tabellarisch aufgelistet. Als Betriebsspannung ist hier $U_B=5V$ angelegt.

U_{E1}	U_{E2}	U_{GS}				Schaltzustand				U_A
Logische Werte		N_1	N_2	P_1	P_2	N_1	N_2	P_1	P_2	Log. We
0	0	0V	0V	-5V	-5V	sperrt	sperrt	durchl.	durchl.	1
0	1	0V	5V	-5V	0V	sperrt	durchl.	durchl.	sperrt	1
1	0	5V	0V	0V	-5V	durchl.	sperrt	sperrt	durchl.	1
1	1	5V	5V	0V	0V	durchl.	durchl.	sperrt	sperrt	0

Anmerkung: AND-Logikgatter erhält man nun aus einen NAND-Gatter und einem Inverter.

Das NOR-Gatter:

Abbildung 6-7 zeigt das Schaltsymbol und die Wahrheitstabelle des NOR-Gatters:

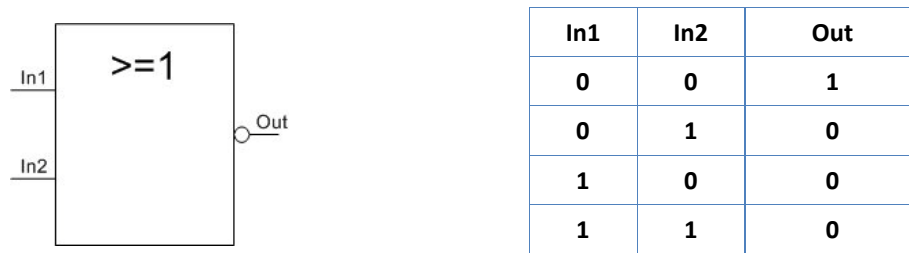


Abbildung 6-7 Schaltsymbol (links) und Wahrheitstabelle (rechts) eines NOR-Gatters

Man kann ein solches Gatter folgenderweise aus CMOS-Transistoren aufbauen:

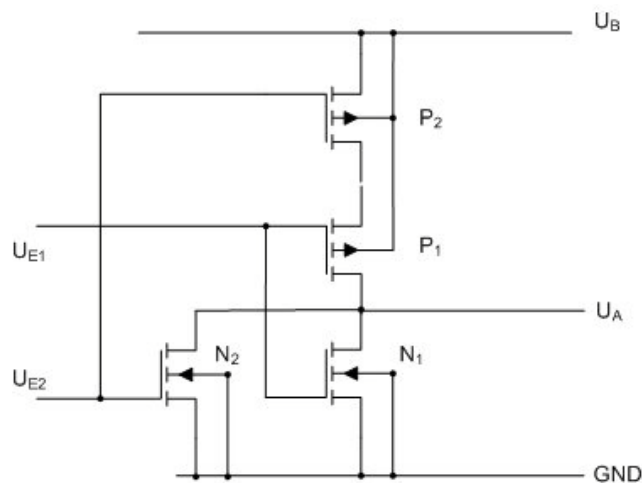


Abbildung 6-8 Aufbau eines NOR-Gatters in CMOS-Technologie

Es besteht aus den beiden n-MOS-Transistoren N_1 und N_2 und den beiden p-MOS-Transistoren P_1 und P_2 . Die Eingänge sind mit U_{E1} und U_{E2} bezeichnet, das Ausgangssignal ist die Spannung U_A . U_B ist die Betriebsspannung des Gatters. Das Verhalten der Transistoren und des gesamten Gatters ist wiederum tabellarisch aufgelistet. Als Betriebsspannung ist hier $U_B=5V$ angelegt.

U_{E1}	U_{E2}	U_{GS}				Schaltzustand				U_A (Log.Werte)
		N_1	N_2	P_1	P_2	N_1	N_2	P_1	P_2	
0	0	0V	0V	-5V	-5V	sperrt	sperrt	durchl.	durchl.	1
0	1	0V	5V	-5V	0V	sperrt	durchl.	durchl.	sperrt	0
1	0	5V	0V	0V	-5V	durchl.	sperrt	sperrt	durchl.	0
1	1	5V	5V	0V	0V	durchl.	durchl.	sperrt	sperrt	0

Anmerkung: OR-Logikgatter erhält man nun aus einen NOR-Gatter und einem Inverter.

6.3 DATEN-SPEICHER

Digitale Schaltungen bzw. logische Gatter können verwendet werden um Datenspeicher (Hauptspeicher, Arbeitsspeicher) zu realisieren. Ein solcher Datenspeicher besteht aus vielen einzelnen Speicherzellen, wobei jede Speicherzelle genau ein Bit (also den Wert logisch 0 oder logisch 1) abspeichern kann. Daher gibt es 2 Hauptaufgaben, die hier bewältigt werden müssen:

1. Das Ansteuern von Speicherzellen
2. Das Abspeichern von Werten in der Speicherzelle.

Für die erste Aufgabe werden so genannte Dekoder verwendet, für die zweite Aufgabe wird die Flip-Flop-Technik eingesetzt. Zunächst wollen wir uns aber ansehen, wie der Datenspeicher prinzipiell aufgebaut und organisiert ist.

6.3.1 SPEICHERORGANISATION

Der Speicher besteht zunächst aus einzelnen Speicherzellen. Jede Zelle kann genau ein Bit abspeichern. Eine Speicherzelle kann schematisch folgendermaßen dargestellt werden:

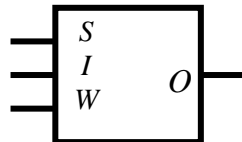


Abbildung 6-9 Schematische Darstellung einer Speicherzelle

Die Speicherzelle verfügt über den eigentlichen Speicherbereich, der hier noch nicht genauer betrachtet wird, und insgesamt 4 Anschlüssen. Davon ist der Anschluss O (Output) eine Ausgangs-Leitung, d. h. an diesem Anschluss kann der Wert der in der Speicherzelle abgelegt wurde ausgelesen werden, während die Anschlüsse S, I und W Eingangs-Leitungen sind. Der Anschluss I (Input) dient dazu Werte in den Speicher zu schreiben. Dies ist aber nur dann erlaubt, wenn am Anschluss W (Write) eine logische 1 anliegt. Liegt hier eine logische 0 dann kann nur lesend auf die Zelle zugegriffen werden. W gibt also an, ob in die Zelle geschrieben werden soll, oder ob ein Wert aus der Zelle ausgelesen werden soll. Der Zugriff auf die Zelle (egal ob lesend oder schreibend) ist aber generell nur dann möglich, wenn am Eingang S (Select) eine 1 anliegt. Liegt hier eine 0, kann weder in die Zelle geschrieben werden, noch ein Wert ausgelesen werden.

Zumeist werden Daten nicht bitweise, sondern byteweise oder wortweise verarbeitet. Man greift also nicht auf einzelne Speicherzellen zu, sondern auf ein **Register**. Ein Register ist ein Zusammenschluss von mehreren Speicherzellen. Dies sind üblicherweise 8 Speicherzellen (Byte-Register) oder 16 Speicherzellen (Wort-Register). Der Übersicht halber ist in Abbildung 6-10 ein Beispiel eines Registers mit nur 4 Speicherzellen dargestellt. Wie man sieht besitzt das Register für alle 4 Speicherzellen eine gemeinsame S- und W-Leitung. Man kann also alle Speicherzellen des Registers nur gemeinsam zum lesen oder schreiben anwählen. Die I- und O-Leitungen sind separat nach Außen geführt. So kann man quasi parallel alle 4 Bits zur gleichen Zeit auslesen.

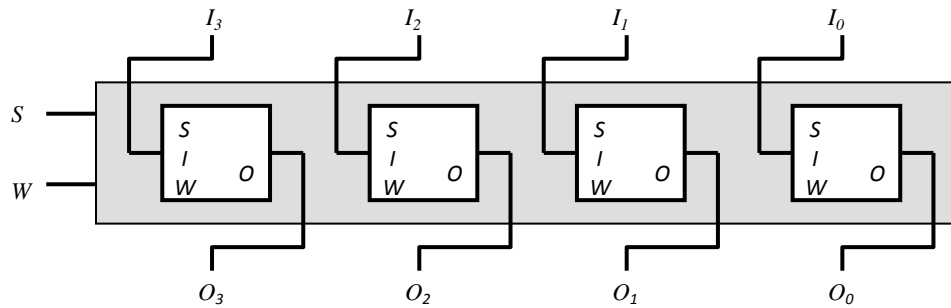


Abbildung 6-10 Schematischer Aufbau eines Registers mit 4 Speicherzellen.

Ein Datenspeicher ist nun aus vielen solchen Registern aufgebaut. Abbildung 6-11 zeigt dies beispielhaft mit 4 Registern à 4 Speicherzellen.

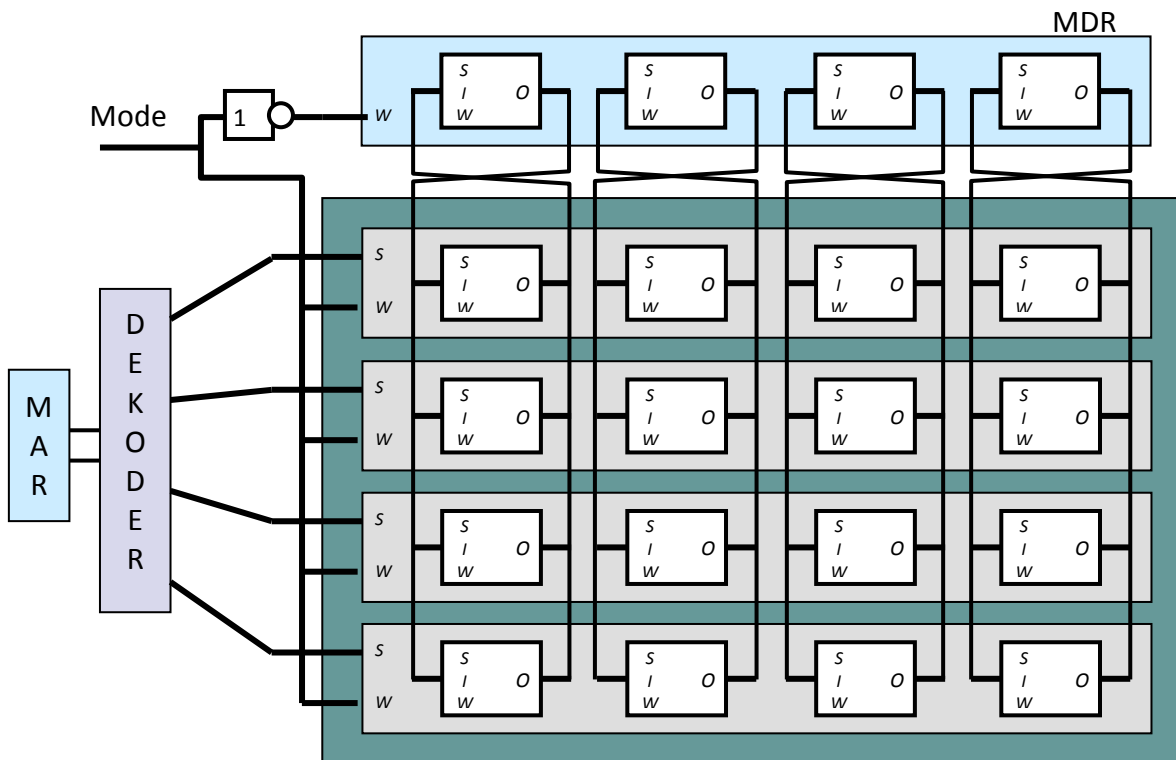


Abbildung 6-11 Schematischer Aufbau eines Datenspeichers. MAR: Memory Address Register, MDR: Memory Data Register.

Um Daten in den Speicher schreiben zu können, werden diese zunächst im **Memory Data Register (MDR)** zwischengespeichert. Dann wird an der **Mode**-Leitung eine logische 1 angelegt. Da die Mode-Leitung mit dem W-Input aller Register verbunden ist, werden damit alle Register des Datenspeichers auf „Schreiben“ gesetzt. Dann wird über das **Memory Address Register (MAR)** ein Register des Datenspeichers ausgewählt, in dem die Daten abgelegt werden sollen. Hierzu hat jedes Register im Datenspeicher eine eindeutige **Adresse** (ganze Zahl ≥ 0) zugeordnet bekommen. Ein **Dekoder** wählt anhand der Adresse aus dem MAR die richtige Select-Leitung aus und ermöglicht es somit in ein definiertes Register zu schreiben. Außerdem ist die Mode-Leitung über eine NOT-Verknüpfung mit dem MDR verbunden, was dazu führt, dass wenn der Datenspeicher auf „Schreiben“ gesetzt wird, das MDR im Mode „Lesen“ ist. Nur so ist es möglich, die Werte vom MDR in den Datenspeicher zu übertragen.

Möchte man Daten aus dem Datenspeicher auslesen, wird zunächst das entsprechende Register über das MAR angesprochen und über den Dekoder die entsprechende Select-Leitung auf 1 gesetzt. Nun wird die Mode-Leitung auf 0 gelegt und damit einerseits alle Register in den Modus „Lesen“ versetzt, und gleichzeitig das MDR in den Modus schreiben. Dadurch wird der Wert aus dem Datenspeicher in das MDR übertragen und kann von dort aus weiterverarbeitet werden.

Möchte man auf den Speicher nicht nur wort- bzw. byteweise zugreifen, sondern bitweise ist in diesem Fall ist noch ein weiterer Dekoder zur Auswahl des jeweiligen Bits (bzw. der Spalte) erforderlich. Nun werfen wir noch einen Blick ins „Innere“ von Dekodern und Speicherzellen.

6.3.2 DER DEKODER

Dekoder werden dafür verwendet die im Datenspeicher ein ausgewähltes Register anzusprechen. Hierfür muss der Dekoder einen Adresswert analysieren und die zu diesem Adresswert gehörende Select-Leitung ansprechen. Die Adresse selbst wird dem Dekoder vom Memory-Address-Register übermittelt und zwar in Form einer binären Codierung. Der größte Adresswert A_{\max} der dem Dekoder übergeben werden kann hängt von der Anzahl der Verbindungsleitungen k zwischen MAR und Dekoder ab. Es gilt

$$A_{\max} = 2^k - 1 \quad (6.3-1)$$

und es kann jede beliebige ganze Zahl A mit

$$0 \leq A \leq A_{\max}$$

als Adresse verwendet werden.

Der Dekoder hat also k Eingänge. Um nun alle A_{\max} Adressen auswählen zu können, muss er außerdem über

$$n = 2^k \quad (6.3-2)$$

Ausgangsleitungen verfügen.

Wir betrachten nun exemplarisch einen 1-aus-4-Dekoder, d. h. es kann eine aus 4 Ausgangleitungen selektiert werden. Nach Gl. (4.3-2) benötigt der Dekoder dazu 2 Eingangsleitungen.

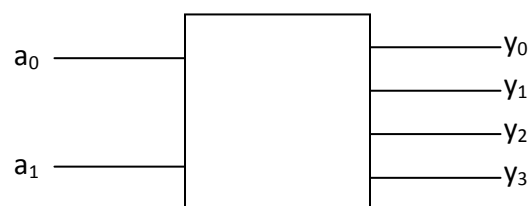


Abbildung 6-12 Prinzipieller Aufbau eines 1-aus-4 Dekoders.

Man erhält dann folgende Wahrheitstabelle:

Adresse A	Eingangsleitungen		Ausgangsleitungen			
	a₁	a₀	Y₃	Y₂	Y₁	Y₀
0	0	0	0	0	0	1
1	0	1	0	0	1	0
2	1	0	0	1	0	0
3	1	1	1	0	0	0

Um das in der Wahrheitstabelle notierte Verhalten technisch zu realisieren kann ein Dekoder folgendermaßen aufgebaut werden:

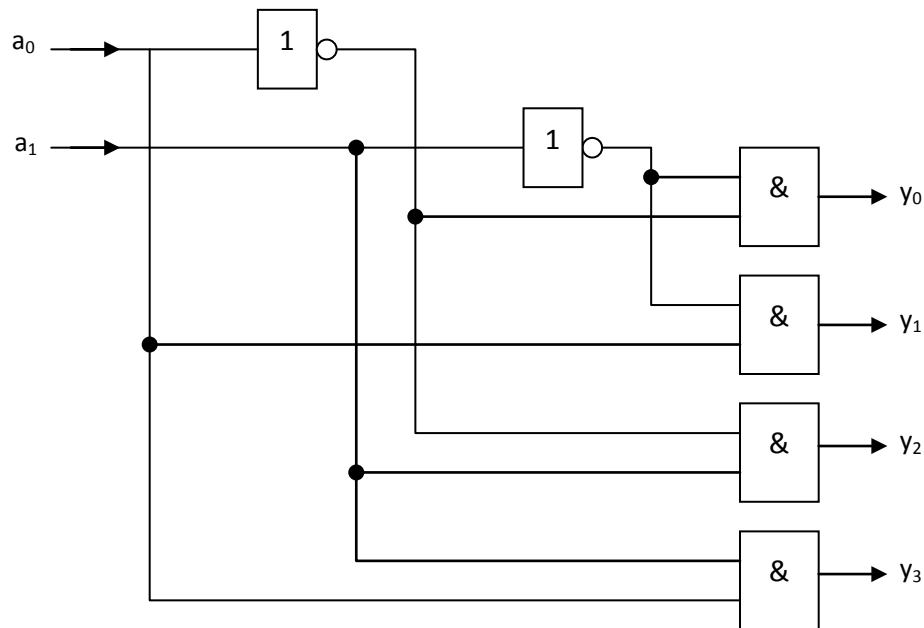


Abbildung 6-13 Technische Realisierung eines 1-aus-4-Dekoders.

6.3.3 FLIP-FLOPS

Nun werfen wir einen Blick ins innere der Speicherzelle. Die Speicherzelle muss einen Wert, der ihr über eine Input-Leitung übermittelt wird abspeichern, was bedeutet, dass der Wert auch dann noch innerhalb der Speicherzelle vorhanden sein muss, wenn der Pegel an der Input-Leitung zurückgesetzt wird. Andererseits soll der Wert nicht fest „eingebrennt“ werden, dass man ihn nicht mehr ändern kann, es muss also eine Möglichkeit gegeben sein, durch einen erneuten Schreibbefehl den gespeicherten Wert durch einen anderen zu ersetzen. Um dies technisch zu realisieren wer

den so genannte **Flip-Flops** eingesetzt. Ein Flip-Flop hat genau zwei stabile Zustände auf das es sich „einpendeln“ kann (H:Highpegel, L: Lowpegel). Man bezeichnet ein Flip-Flop daher auch als Bipolare Kippstufe. Wir betrachten zunächst die einfachste Ausführung eines Flip-Flops – das RS-Flip-Flop.

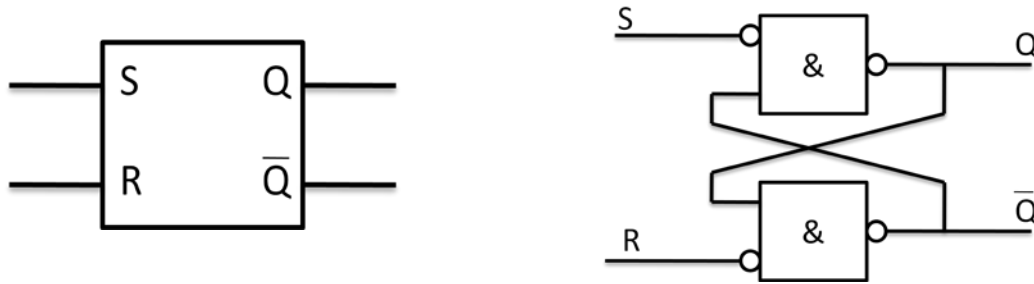


Abbildung 6-14 Schaltzeichen und logischer Aufbau eines RS-Flip-Flops.

Tabelle 6-2 zeigt wie das RS-Flip-Flop beschaltet werden kann und welche Auswirkungen dies auf den gespeicherten Wert Q_{n+1} hat. Hierbei ist Q_n der vorher im Flip-Flop gespeicherte Wert.

Tabelle 6-2 Schaltzustand des RS-Flip-Flops nach setzen von R und S.

R	S	Q_{n+1}
0	0	Q_n
1	0	0
0	1	1
1	1	verboten

Der Zustand des RS-Flip-Flops wird also in dem Moment geändert in dem entweder an R oder an S eine 1 angelegt wird. Solange R und S auf 0 liegen bleibt der im Flip-Flop vorher gesetzte Wert erhalten. Wie man aus der Tabelle sehen kann, darf niemals R und S gleichzeitig auf 1 gesetzt werden. Dies würde einen instabilen Zustand des RS-Flip-Flops erzeugen.

Zwei Verbesserungen kann man an einem RS-Flip-Flop noch durchführen. Erstens kann man dafür sorgen, dass das Flip-Flop nur zu ganz bestimmten Zeiten umgeschaltet werden kann – dies erfolgt durch hinzufügen einer Taktleitung – und zweitens kann man den R und den S-Eingang über einen Inverter miteinander koppeln, so dass sichergestellt ist, dass niemals R und S gleichzeitig auf 1 gesetzt sind. Dies bezeichnet man dann als **D-Flip-Flop**, wobei der Eingang D zum Beschalten des Flip-Flops der R-Leitung des RS-Flip-Flops entspricht.

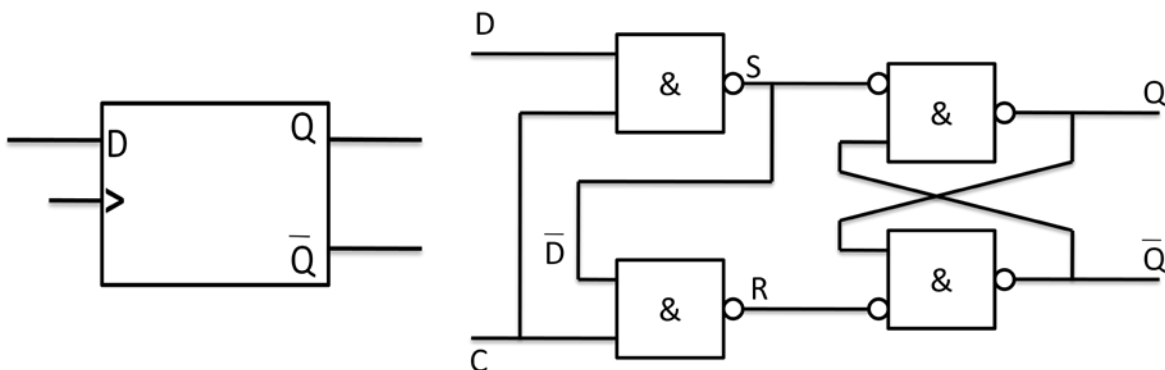


Abbildung 6-15 Schaltzeichen und logischer Aufbau eines D-Flip-Flops.

Technische Realisierung von Flip-Flops

Eine Möglichkeit Flip-Flops technisch zu realisieren, ist über die CMOS-Technologie (

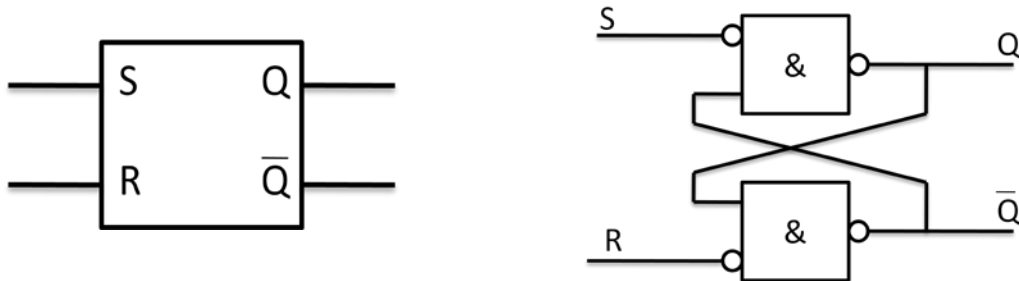


Abbildung 6-14, rechts). Alternativ kann auch die NMOS-Technologie verwendet werden, bei der man die Schaltung ausschließlich aus n-MOS-FETs aufbaut.

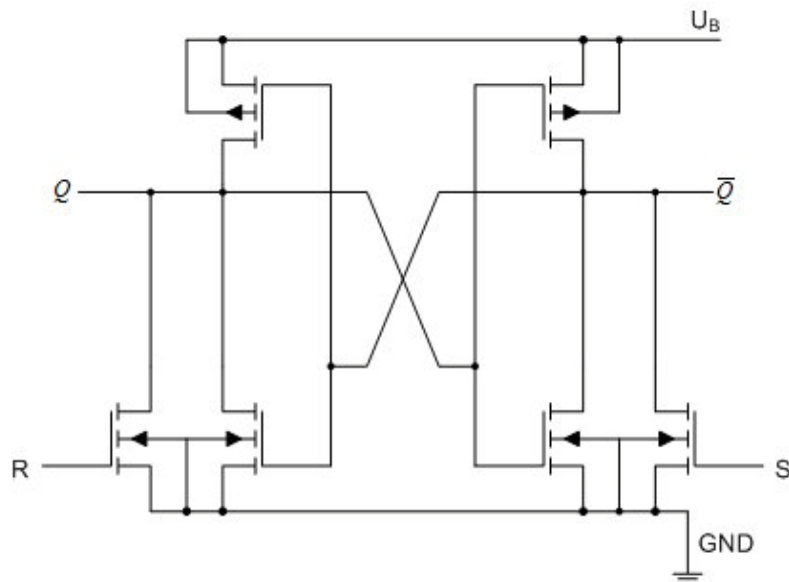


Abbildung 6-16 RS-Flip-Flop in CMOS-Technik.

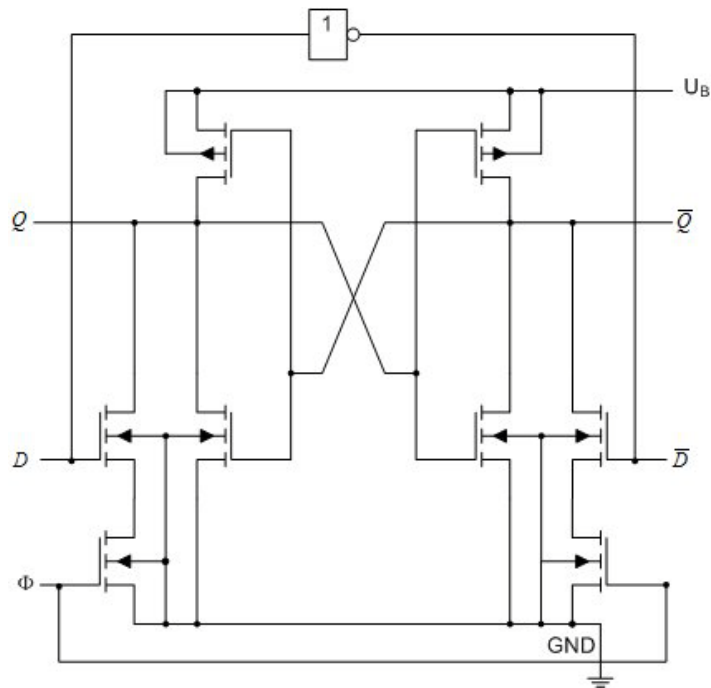


Abbildung 6-17 Getaktetes D-Flip-Flop. Nur wenn Φ auf 1 gesetzt ist, dann der Schaltzustand über die Leitung D geändert werden.

6.3.4 RAM UND ROM

RAM (Random Access Memory) ist ein Schreib-/Lesespeicher. Im RAM werden die hineingeschriebenen Werte nur so lange gespeichert, wie eine Betriebsspannung U_B vorhanden ist. Schaltet man die Spannungsversorgung aus, gehen alle gespeicherten Werte verloren. RAM wird daher zum als Arbeits- bzw. Hauptspeicher verwendet und dient u. a. zum Ablegen von gerade auszuführendem Programmcode oder von Zwischenergebnissen. Man unterscheidet zwischen **statischem RAM** und **dynamischem RAM**. Im statischen RAM bleibt ein einmal geschriebener Wert bis zum Ausschalten der Betriebsspannung erhalten, beim dynamischen RAM müssen die zu speichernden Werte ständig (alle 10 ms) neu „aufgefrischt“ werden. Der Aufbau einer Speicherzelle des statischen RAM ist in Abbildung 6-18 dargestellt.

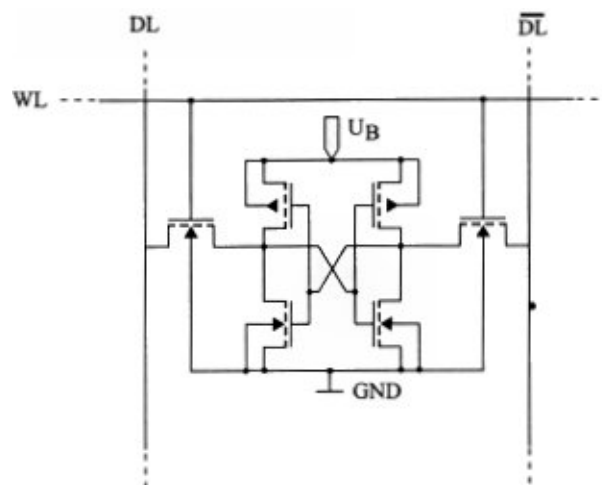


Abbildung 6-18 Speicherzelle des statischen RAM (aus [6]).

Sie besteht im Wesentlichen aus einem RS-Flip-Flop. Pro Zelle werden 6 Transistoren benötigt (Sechs-Transistorzelle). Vorteil dieser Technik sind sehr kurze Zugriffszeiten auf eine Zelle, aufgrund der aufwendigen Schaltungstechnik ist die Speichergröße (in Byte) aber begrenzt. Typische Werte sind 256-512 kByte. Diese Art von Speicher eignet sich also für den Cache-Speicher eines Prozessors.

Beim dynamischen RAM erfolgt die Speicherung durch einen CMOS-Kondensator, der über einen Transistorschalter adressiert wird (Abbildung 6-19).

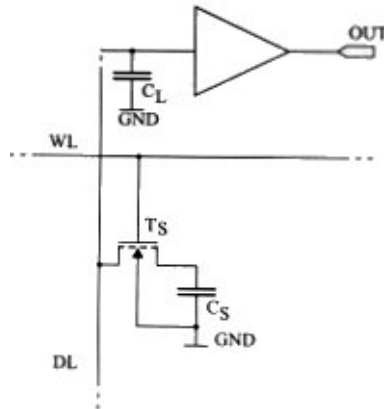


Abbildung 6-19 Speicherzelle des dynamischen RAM (aus [6]).

Durch diesen kompakten Aufbau (1 Transistor/Zelle) ist eine sehr hohe Speicherdichte möglich, d. h. es können höhere Speichergrößen (in Byte) erreicht werden. Nachteil ist jedoch, dass sich die Kondensatoren durch Leckströme entladen. Daher müssen die Werte alle 10 ms aufgefrischt werden, was zeitaufwändig ist. Daher ist der dynamische Speicher langsamer als der statische.

Beim ROM (Read Only Memory) handelt es sich um einen Speicher der nicht einfach beschrieben werden können. Sie sind nur zum Auslesen gedacht und können z. B. dazu verwendet werden Firmware für einen Mikroprozessor dauerhaft zu speichern. Hier gehen die gespeicherten Werte auch nicht verloren, wenn die Betriebsspannung abgeschaltet wird. Auch der ROM besteht aus Speicherzellen. Diese sind sehr kompakt aufgebaut und bestehen z. B. aus einem einzigen MOS-FET (Abbildung 6-20).

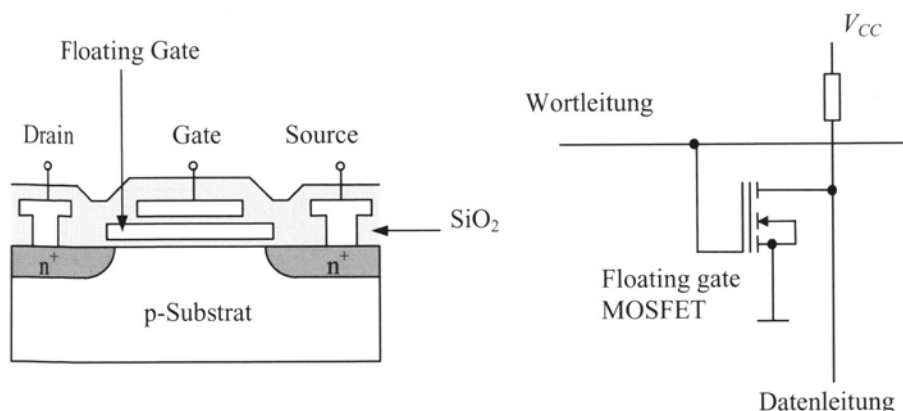


Abbildung 6-20 Floating-Gate MOSFET und Speicherzelle eines EPROMs (aus [13]).

Bei den maskenprogrammierbaren ROMs werden bereits beim Herstellungsprozess ausgewählte Leitungen aufgetrennt, oder die Eigenschaften von ausgewählten Transistoren verändert. So werden Werte dauerhaft abgespeichert.

Die elektrisch programmierbaren ROMs (PROM) können nach dem Herstellungsprozess genau einmal beschrieben werden, indem beim „brennen“ des ROMs an definierten Stellen Kurzschlüsse erzeugt werden. Weiterhin gibt es EPROMs und EEPROMs die mehrfach beschrieben (gebrannt) werden können. Beide Typen bestehen aus Floating-Gate MOSFETs (siehe Abbildung 6-20). Diese verfügen über ein zusätzliches Gate (Floating Gate) was vollständig isoliert in der Siliziumoxid-Schicht liegt. Legt man eine sehr erhöhte Spannung ($U_{GS} \approx 24V$ und $U_{DS} \approx 17V$) an können Elektronen durch den Isolator Tunneln und geraten so auf das Floating-Gate. Somit wird dort dauerhaft (ca. 10 Jahre) eine negative Ladung gespeichert, die verhindert, dass beim Anlegen einer normal großen positiven Spannung U_{GS} an das Gate der MOSFET nicht leitend wird, also dauerhaft sperrt. So kann permanenter High-Pegel erzeugt werden und damit eine logische 1 abgespeichert werden. Bei EPROMs kann das Isoliermaterial durch UV-Licht ionisiert werden. Dadurch wird es leitfähig und die Elektronen auf dem Floating-Gate können so wieder in das Substrat wandern. Nach ca 20 Minuten Bestrahlung mit UV-Licht kann man das EPROM wieder löschen. Bei EEPROM erfolgt dieser Löschvorgang nicht durch UV-Licht, sondern auch durch Anlegen einer Spannung U_{GS} .