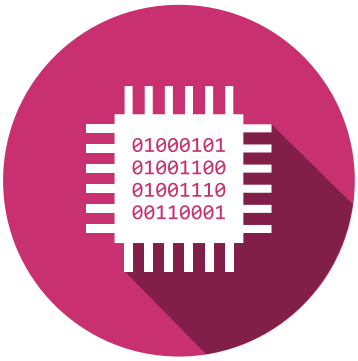
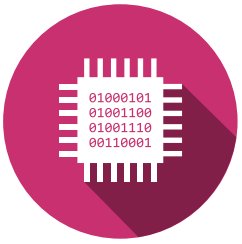


Festwertspeicher (ROM)

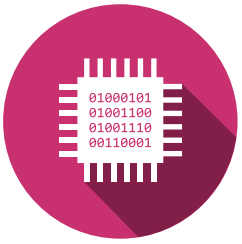
Vorlesung Digitales Design

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Orientierung: Informatik und Kommunikationssysteme (ISC)
Kurs: Digitales Design (DiD)
Verfasser: Christophe Bianchi, François Corthay, Pierre Pompili, Silvan Zahno
Datum: 25. August 2022
Version: v2.1



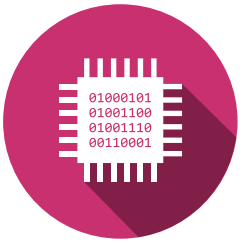
Inhaltsverzeichnis



1 Einführung

Mit den Festwertspeichern können mittels Programmierung von Sicherungselementen kombinatorische Logikfunktionen realisiert werden. Sie werden in erster Linie benutzt, um das Programm eines Mikroprozessors abzuspeichern. Hierfür sind sie gleich wie die LeseSchreib-Speicher an den Mikroprozessor angeschlossen. Heutzutage können sogar wiederbeschreibbare Speicher realisiert werden. Diese Operation ist jedoch langsamer als das Schreiben in einen LeseSchreib-Speicher.

Dieses Kapitel befasst sich mit der Struktur der Festwertspeicher, den Anschlussmethoden für eine Erhöhung der Kapazität, verschiedenen Arten von Festwertspeichern und gewissen typischen Schaltungen.



2 Universelle Logikfunktion

2.1 Struktur der ROMs

Ein Demultiplexer, dessen Dateneingang bei '1' ist, decodiert einen Steuerungseingang bei n Bit und bringt 2^n Ausgänge hervor, die allen möglichen Kombinationen der Eingänge entsprechen. Die Tabelle 1 stellt die Wahrheitstabelle eines Demultiplexers mit 2 Steuerungseingängen dar und zeigt, wie man durch die Kombination dieser Ausgänge mittels ODER-Gatter verschiedene Funktionen realisieren kann.

a_1	a_0	w_0	w_1	w_2	w_3	d_3	d_2	d_1	d_0
0	0	**1**	0	0	0	1	0	0	1
0	1	0	**1**	0	0	0	1	1	1
1	0	0	0	**1**	0	1	1	1	1
1	1	0	0	0	**1**	0	1	0	0

Tabelle 1: Realisierung von Funktionen mit Hilfe eines Demultiplexers

Für die in der Tabelle 1 dargestellten Funktionen gelten folgende Gleichungen 2.1.

$$\begin{cases} d_3 = \overline{a_0} \\ d_2 = a_0 + a_1 \\ d_1 = a_0 \oplus a_1 \\ d_0 = \overline{a_0 \cdot a_1} \end{cases} \tag{1}$$

Das entsprechende Schema ist in der Abbildung 1 dargestellt.

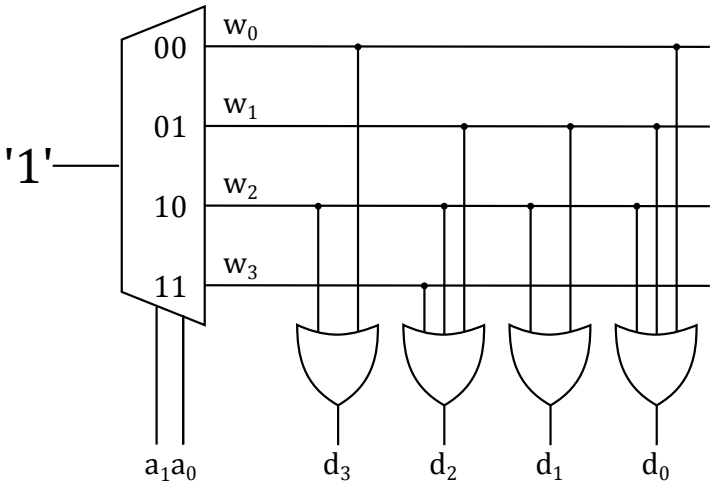
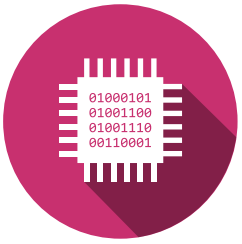


Abbildung 1: Struktur eines ROM

Es ist offensichtlich, dass mit der Kombination eines Demultiplexers und eines ODER-Gatters jede beliebige kombinatorische Logikfunktion realisiert werden kann. Solche Schaltungen existieren in Form von integrierten Schaltungen und werden Nur-Lese-Speicher (**R**ead **O**nly **M**emory (ROM)) oder Festwertspeicher genannt. Die Bezeichnung Speicher beruht darauf, dass ihre wichtigste Aufgabe die Speicherung von Computerprogrammen ist.



Wenn ein ROM als kombinatorische Logikfunktion benutzt wird, entspricht er einer Wahrheitstabelle:

- die Ausgangsleitungen des Demultiplexers entsprechen den Linien der Tabelle,
- für n Eingänge gibt es 2^n Linien,
- eine Verbindung mit einem ODER-Gatter entspricht einer '1' in der Tabelle, keine Verbindung entspricht einer '0'.

2.2 Realisierung der programmierbaren ODER-Funktion

Um die Grösse der Schaltung zu verringern und die Regelmässigkeit zu verbessern wird die ODER-Funktion verdrahtet realisiert. In der Abbildung 2 ist eine mögliche Realisierung dieser ODER-Funktion in Form einer NOR-Funktion gefolgt von einem Inverter dargestellt. In diesem Schema wird das verdrahtete NOR mit Hilfe von MOS-Transistoren realisiert, die wie Schalter arbeiten. Sie sind leitend, wenn der Steuerungseingang bei '1' ist und schliessen dann die Leitung, mit der sie verbunden sind, zum Potential der logischen '0', V_{SS}, kurz. Wenn kein Transistor die Linie auf '0' zwingt, wird diese von einem Widerstand auf dem Potential der logische '1' gezogen, V_{DD}.

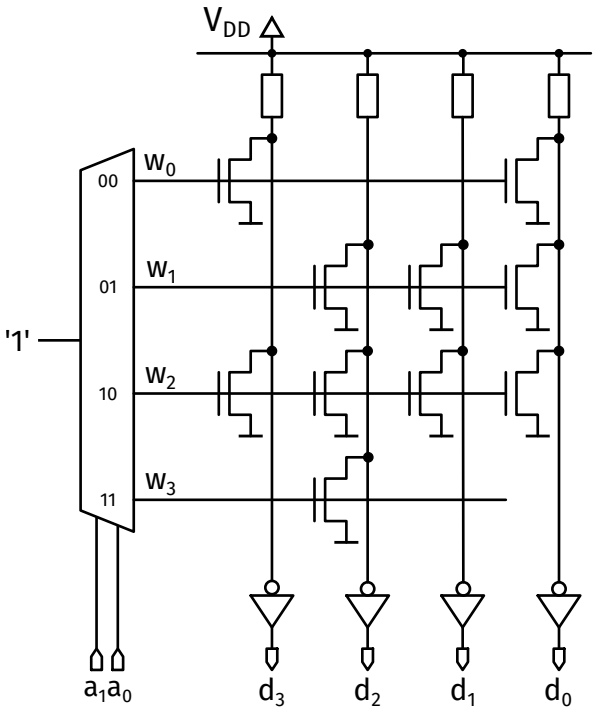
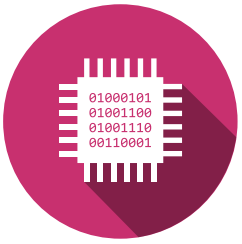


Abbildung 2: Festwertspeicher des Typs NOR

Eine '0' in der Wahrheitstabelle entspricht einem nicht vorhandenen Transistor und eine '1' einem vorhandenen Transistor.



3 Zusammensetzung von Speicherschaltungen

3.1 Kapazität eines Speichers

Die Kapazität eines Speichers wird mit der Anzahl Bit der entsprechenden Wahrheitstabelle angegeben. Diese Kapazität ist das Produkt aus der Anzahl Ausgangsleitungen des Multiplexers, n_w, und der Anzahl Ausgangs- oder Datenleitungen , n_d des Speichers (Gleichungen 3.1).

$$C = n_w \cdot n_d = 2^{n_a} \cdot n_d \tag{2}$$

Wir haben bereits gesehen, dass die Anzahl Ausgangsleitungen des Multiplexers, n_w, gleich 2 hoch die Anzahl Eingangs- oder Adressleitungen des Speichers, n_a, ist (Gleichungen 3.1).

$$n_w = 2^{n_a} \tag{3}$$

3.1.1 Beispiele

Der Speicher mit 2 Eingangs- und 4 Ausgangsleitungen in der Abbildung 2 hat eine Kapazität von 4 x 4 Bit.

Ein Speicher mit 10 Eingangs- und 8 Ausgangsleitungen ist ein Speicher mit 1024 x 8 Bit. Da 1024 = 1k und 8 Bit gleich 1 *Byte (byte, B)* sind, hat dieser Speicher eine Kapazität von 1 kB.

Ein Speicher mit 16 Eingangs- und 8 Ausgangsleitungen ist ein Speicher mit 65536 x 8 Bit, d.h. 64kB.

3.2 Speichererweiterung

Die Entwicklung der Mikroprozessoren und der steigende Bedarf an immer grösseren Speichern für immer komplexere Programme hält die Benutzer zur Erweiterung ihrer Speicherkapazitäten an. Die Kapazität eines Speichers kann vergrössert werden, indem man entweder die Anzahl Adressen und somit auch die Zahl der Linien in der Wahrheitstabelle oder die Anzahl Datenleitungen erhöht.

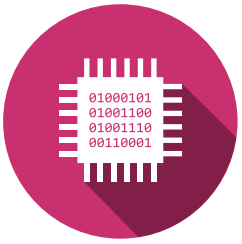
3.3 Serienschaltung

Die Kapazität eines Speichers kann mit gleichbleibender Anzahl Datenleitungen erhöht werden, wenn man die Schaltungen in Serie schaltet.

Hierfür werden die Ausgänge der Schaltungen untereinander verbunden, und eine Decodierschaltung wählt eine der Schaltungen aus, deren Aufgabe es sein wird, ihre Informationen auf die Datenleitungen zu übertragen. Es wird immer nur eine Schaltung gleichzeitig ausgewählt. Um Interferenzen zu vermeiden, sind die Ausgänge der anderen Schaltungen hochohmig. Dies ist in der Abbildung 3 dargestellt.

3.4 Decodierung

Für Speicherschaltungen gleicher Kapazität wird für die Decodierung ein Demultiplexer benutzt. Er wird mit den werthöchsten Bits der Adresse gesteuert und die wertniedrigen Bit werden direkt



auf die Speicher übertragen, damit sie von ihren internen Demultiplexern benutzt werden können.

Für Speicherschaltungen unterschiedlicher Kapazität wird für die Decodierung eine komplexere Schaltung benötigt, die in der Regel einem Baum von Demultiplexern entspricht, welcher der Grösse der Demultiplexer der Speicher entspricht.

Abbildung 3: Serienschaltung von ROM

3.5 Speicherbelegungsplan

Bei der Realisierung von Prozessorsystemen mit verschiedenen Speicherschaltungen muss ein *Speicherbelegungsplan (memory map)* ausgearbeitet werden, der angibt, welche Schaltung bei welcher Adresse antwortet.

Dieser Plan gibt die Funktionsweise der Decodierschaltung an, die eine Speicherschaltung in Abhängigkeit der vom Mikroprozessor erstellten Adressen auswählt.

3.5.1 Beispiel

Die Abbildung 4 zeigt ein Beispiel einer Speicherbelegung eines Mikroprozessors mit 16 Adressleitungen.

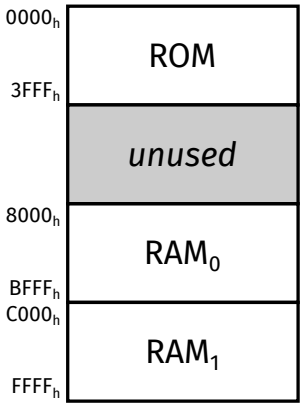


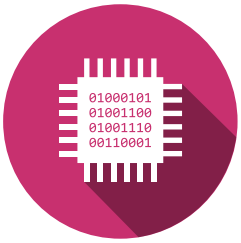
Abbildung 4: Beispiel für die Speicherbelegung eines Mikroprozessors

Mit dieser Speicherbelegung antwortet der ROM solange sich die Adresse zwischen 0000_h und $3FFF_h$ befindet. Wenn die Adresse zwischen 8000_h und $BFFF_h$ ist, wird der RAM_0 gewählt. Mit einer Adresse zwischen 4000_h und $7FFF_h$ reagiert keine Schaltung und die Datenleitungen bleiben hochohmig.

Diese Decodierung ist mit einem Demultiplexer 1 zu 4 möglich, der mit den 2 werthöchsten Bit der Adressen gesteuert wird.

3.5.2 Kommentar

Da die Mikroprozessoren für die Kommunikation mit den Speichern einen Adress- oder einen Datenbus benutzen, bieten die Hersteller E/A-Schaltungen an, z.B. serielle oder parallele Schnittstellen, die den gleichen Kommunikationsmodus benutzen. Der Mikroprozessor betrachtet diese Schaltungen als kleine Speicher und beim Lesen der Daten einer solchen Schaltung hat der Prozessor Zugriff auf die über diesen Port übertragenen Daten. In diesem Fall spricht man von *speicherzugeordneten E/A (memory-mapped I/O)*.



3.6 Parallelschaltung

Die Kapazität eines Speichers kann mit gleichbleibender Anzahl Adressleitungen erhöht werden, wenn man die Schaltungen parallelschaltet (figure 5).

Diese Montage ist typisch für Mikroprozessoren, die mit 16, 32 oder 64 Datenbit arbeiten, während die gängigen Speicher 1, 4 oder 8 Bit liefern.

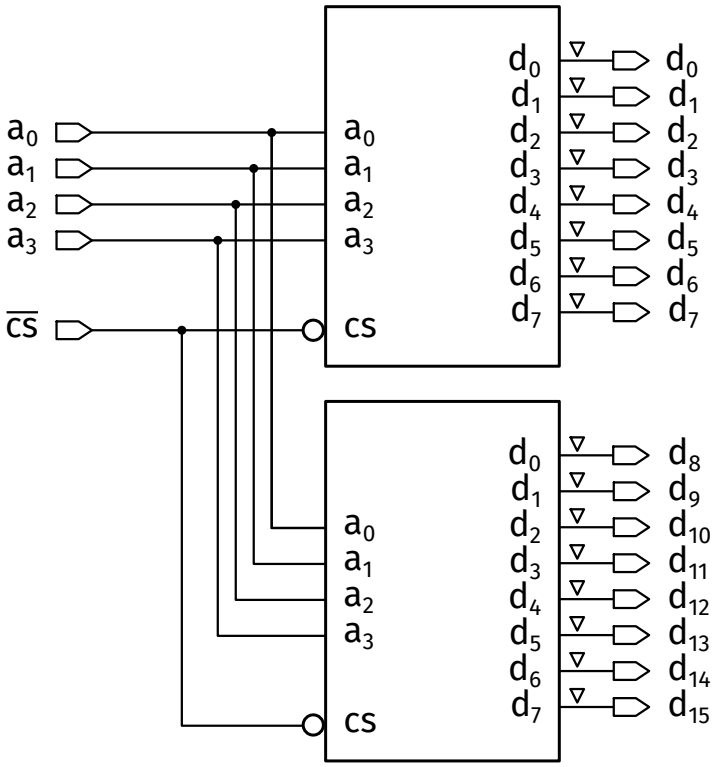
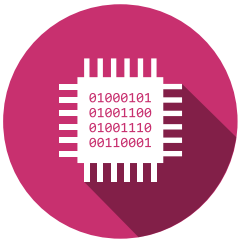


Abbildung 5: Parallelgeschaltete ROM



4 Arten von Festwertspeichern

4.1 ROM

Die Funktion eines ROM hängt davon ab, ob Transistoren vorhanden sind oder nicht. Die Transistoren werden in der Fabrik, d.h. bei der Herstellung der Schaltung realisiert.

Die Herstellung eines ROM kostet einige Tausend Franken und dauert einige Wochen. Ein ROM wird in der Regel nur bei grossen Serien gewählt, d.h. ab tausend Stück.

4.2 PROM

Für kleinere Serien wird ein programmierbarer Festwertspeicher (PROM) gewählt. Für die Verbindungen zwischen den Transistoren und den Linien werden Sicherungen benutzt. Der ROM wird programmiert, indem man die Sicherungen durchbrennt, die mit den unerwünschten Transistoren verbunden sind. In der Abbildung 6 ist ein solcher PROM mit den nicht durchgebrannten Sicherungen dargestellt.

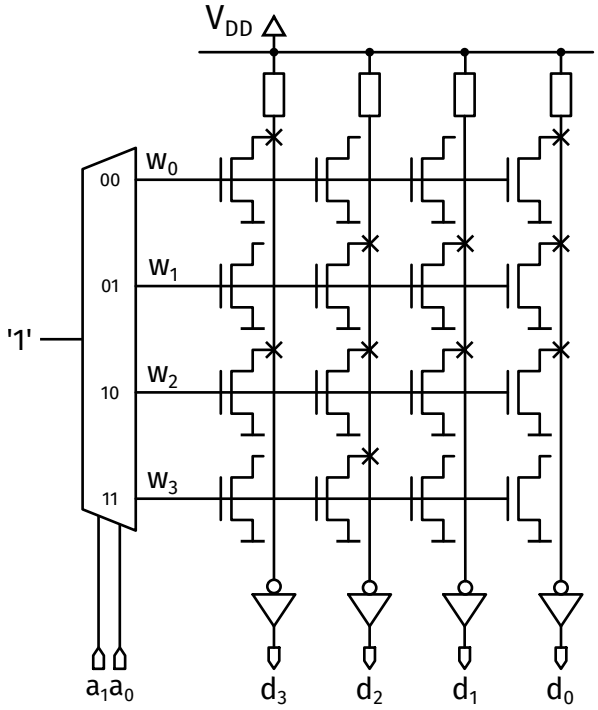
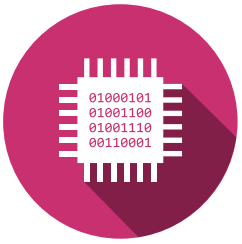


Abbildung 6: PROM

Für die Programmierung eines PROMs wird ein Ad-hoc-Gerät benötigt: ein Speicherprogrammiergerät. Mit unprogrammierten Schaltungen und einem Programmiergerät ist die Herstellungsfrist fast gleich Null.

4.2.1 Kommentar

Für die Programmierung des verdrahteten ODER brennt man Sicherungen durch; eine '0' in der Wahrheitstabelle entspricht einem nicht vorhandenen Transistor und somit einer durchgebrannten Sicherung. Wenn der Inhalt eines PROM nicht vollständig benutzt wird, sollten alle Bit des freien Teils bei '1' gelassen werden, damit die Programmierungszeit verkürzt und die Funktionalität des Systems, falls notwendig, ausgeweitet werden können.



4.3 Programmierungsdateien

Es gibt verschiedene Formate zur PROM-programmierungsdateien. Diese wurden im wesentlichen von Mikroprozessorenherstellern bestimmt.

Der folgende Listing 4.3 gibt den Inhalt einer Sinustabelle im Format Intel Hex.

```
:020000020000FC
:10000000000D1925313C47515B636A71767A7E7F1A
:100010007F7F7E7A76716A635B51473C3125190D8B
:1000200000F3E7DBCFC4B9AFA59D968F8A868281A6
:10003000808182868A8F969DA5AFB9C4CFDBE7F316
:00000001FF
```

Listing 1: Intel HEX RAM Data

Dies Linien sind aufgebaut mit dem Charakter ":", gefolgt von einem Code, welcher die Länge der zu kommenden Information angibt, dann von der Startsdresse, von einem Code, welcher die Art der Information angibt, von den Daten selber und von einem Checksum.

Der folgende Listing 4.3 gibt den Inhalt einer Sinustabelle im Format Motorola EXORciser

```
S00B000044415441204 92f4fF3
S1130000000D1925313C47515B636A71767A7E7F1A
S11300107F7F7E7A76716A635B51473C3125190D8B
S113002000F3E7DBCFC4B9AFA59D968F8A868281A6
S11300308081828 68A8F969DA5AFB9C4CFDBE7F316
S9030000FC
```

Listing 2: Motorola EXORciser RAM Data

Dies Linien sind aufgebaut mit dem Charakter "S", gefolgt von einem Code, welcher die Art der zu kommenden Information angibt, dann von einem Code für die Länge, von der Startsdresse, von den Daten selber und von einem Checksum.

4.4 EPROM

Für die Entwicklung von Schaltungen auf der Basis von ROM können mit Hilfe der PROMs schnell Prototypen hergestellt werden. Jeder neue Prototyp bedingt jedoch das Durchbrennen eines neuen PROM. Für diese Art Anwendungen bieten die Hersteller lösch- und programmierbare Festwertspeicher (**E**rasable **P**rogrammable **R**ead **O**nly **M**emory (EPROM)) an.

In diesen Schaltungen sind die Paare Transistor-Sicherung durch einen Transistor mit einem floating (schwebenden) Gate ersetzt, wie in Abbildung 7 dargestellt. Im Rahmen der Programmierung wird eine Ladung auf das floating Gate angewandt, die zur Steuerspannung des Transistors eine konstante Spannung hinzufügt. Wenn das Gate richtig geladen ist, leitet der Transistor nicht.

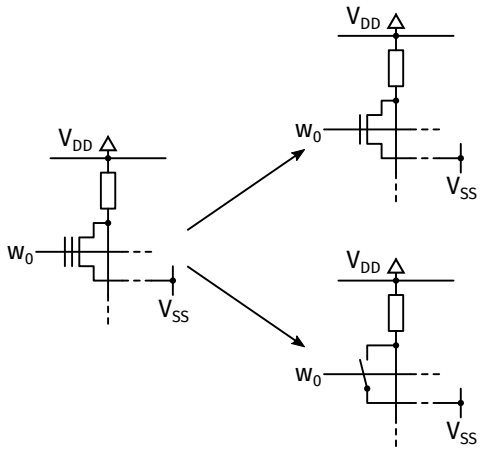


Abbildung 7: Transistor mit floating Gate

Ein programmierter EPROM kann gelöscht werden, indem man die integrierte Schaltung UV-Strahlen aussetzt. Um diesen Löschvorgang zu ermöglichen, haben EPROMs auf der Gebäudeoberseite ein Fenster. Diese Beleuchtung verleiht den sich auf dem floating Gate befindenden Ladungen genug Energie, um sich davon zu lösen.

4.4.1 Anmerkung

Da das Umgebungslicht ebenfalls ultraviolette Strahlen aufweist, können sich die EPROM nach und nach löschen. Um unbeabsichtigtes Löschen zu verhindern, empfiehlt es sich daher, nach dem Programmieren, das Fenster mit einem kleinen Klebeetikett abzudecken.

4.5 OTP-ROM

Da Gehäuse mit einem Fenster teurer sind als normale Gehäuse, bieten die Hersteller auch EPROMs ohne Fenster an, die aber nur einmal programmiert werden können. Diese PROMs werden einmal programmierbare ROM (**O**ne **T**ime **P**rogrammable **R**ead **O**nly **M**emory (OTP-ROM)) genannt.

Mit diesen billigeren Gehäusen können die integrierten Schaltungen der EPROM, die in grossen Serien hergestellt werden, auch zu günstigeren Preisen verkauft werden.

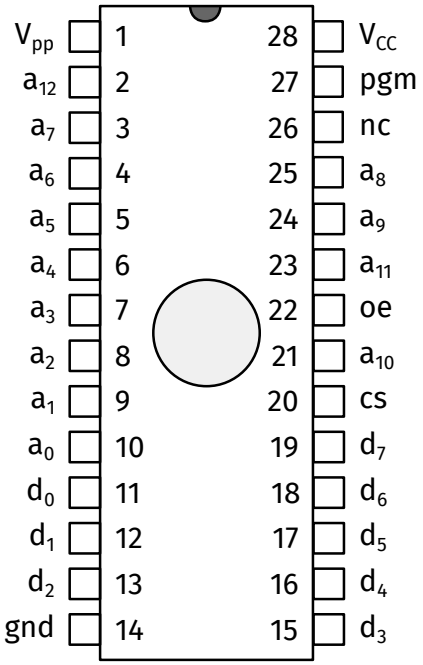


Abbildung 8: EPROM mit Fenster

4.6 EEPROM

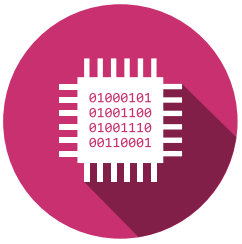
Dank neuen Erkenntnissen bezüglich der physikalischen Phänomene im Zusammenhang mit dem Laden und Entladen der floating Gates können heute Schaltungen realisiert werden, wo die floating Gates elektrisch und ohne UV-Strahlen entladen werden. Schaltungen, für die der Speicherinhalt elektrisch gelöscht werden kann, sind die sog. elektrisch löschbaren Festwertspeicher (**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory (EEPROM)).

Obwohl sie gleichermassen gelesen und beschrieben werden können, sind die EEPROMs trotzdem Speicher, die vor allem gelesen werden, da das Löschen und Programmieren ziemlich viel Zeit beansprucht und die Anzahl Neuprogrammierungen beschränkt sind.

4.7 Flash-Speicher

Um das für das Löschen der Bit der EEPROMs notwendige Material zu verringern, bieten die Hersteller Schaltungen an, für die das Löschen und das anschliessende Beschreiben blockweise und nicht mehr Wort für Wort vorgenommen wird. Diese Speicher werden Flash-Speicher genannt.

Mit dieser Art Speicher muss für das Schreiben eines neuen Werts eines Bit eine ganze Speicherseite gelöscht werden.



5 Typische Schaltung

5.1 Zusätzliche Signale

Neben den Adress- und Datenleitungen besitzen die ROMs auch zusätzliche Kontrollsignale. Diese Signale können in zwei Kategorien eingeteilt werden:

- Programmiersignale: Schutz gegen eine ungewollte Programmierung, besondere Programmierungsspannung, ...
- Kontrollsignale: Wahl der Schaltung, Wahl des Zugriffsmodus, ...

5.1.1 Beispiel: PROM 16 x 8

Die Abbildung 9 zeigt ein typisches Beispiel für die Signale eines PROMs.

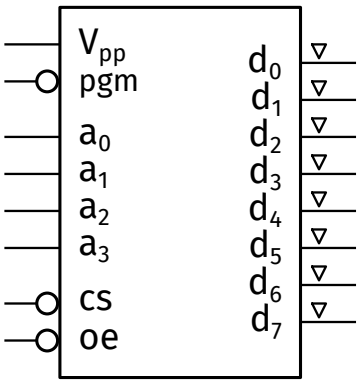


Abbildung 9: Steuersignale eines PROMs

Die Programmierungssignale werden nur zum Zeitpunkt der Programmierung des PROMs benutzt. Im normalen Betrieb müssen diese Eingänge auf das vom Hersteller der Schaltung angegebene Speisepotential gebracht werden.

In der Abbildung 10 ist eine mögliche zeitliche Abfolge der für das Lesen dieser ROM notwendigen Befehle dargestellt. Man stellt fest, dass die beiden Signale $\overline{oe}$ und $\overline{cs}$ aktiv sein müssen, damit der Bestandteil die Datenleitungen aktiviert.

5.2 ROMs mit seriellem Zugriff

Um die Anzahl Stifte und die Komplexität des Anschlusses der ROMs zu vereinfachen, bieten gewisse Hersteller Speicher mit seriellem Zugriff an: die Daten und die Adressen werden sequentiell auf die gleiche Linie übertragen, und ein Taktsignal koordiniert die Informationen.

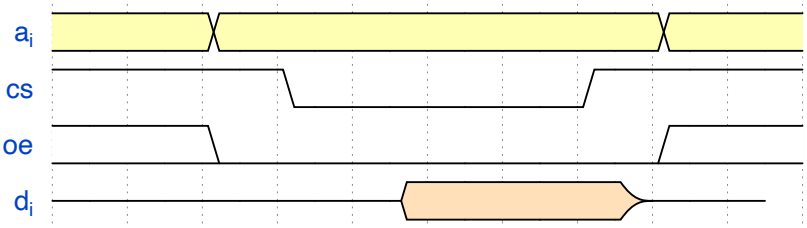
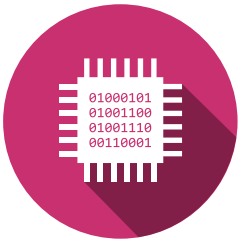


Abbildung 10: Lesezyklus



5.2.1 Beispiel: EEPROM I2C

In der Abbildung 11 sind die Signale für einen seriellen EEPROM dargestellt. Für den Zugriff auf den Speicher benutzt man die Leitungen **S**erial **Cl**ock (SCL) und **S**erial **Da**ta (SDA). 8 identische Komponenten können an denselben seriellen Bus angeschlossen werden, wenn die 3 Stifte (Pin) A2 .. A0 für sie verschiedene Adressen spezifi- zieren können. Das Kontrollsignal **W**rite **P**rotect (WP) verhindert das Schreiben in der Schaltung.

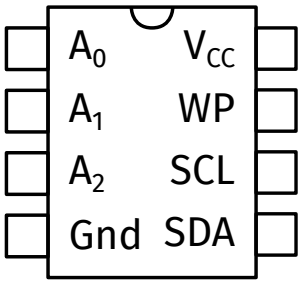


Abbildung 11: EEPROM-Gehäuse mit seriellem Anschluss

Die Abbildung 12 zeigt die Sequenz der Signale SCL und SDA für den Zugriff auf ein Speicherwort.

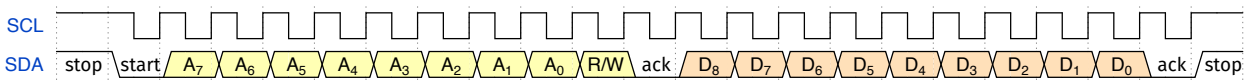
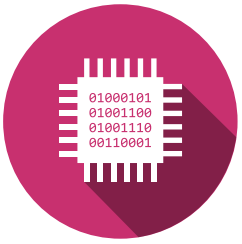


Abbildung 12: Zyklus eines seriellen Zugriffs

5.3 Anwendung der ROMs

Die ROMs werden in erster Linie für die Speicherung der Programme von Mikroprozessoren benutzt: Startcode für Computer oder Anwendungsprogramme für Mikroprozessorsysteme. Hierfür eignen sich in erster Linie die sog. Flash-Speicher.

Die ROMs werden zudem benutzt, um komplexe Funktionstafeln (Sinus, Code für die Anzeige von Zeichen...) zu speichern.



Literatur

[1] Suhail Almani. *Electronic Logic Systems*. second edition. New-Jersey: Prentice-Hall, 1989.

[2] Jean Michel Bernard und Jean Hugon. *Pratique Des Circuits Logiques*. quatrième édition. Paris: Eyrolles, 1987.

[3] Michael D. Ciletti und M. Morris Mano. *Digital Design*. second edition. New-Jersey: Prentice-Hall, 2007.

[4] Clive Maxfield. *Bebop to the Boolean Boogie*. Elsevier, 2009. ISBN: 978-1-85617-507-4. DOI: 10.1016/B978-1-85617-507-4.X0001-0. URL: https://linkinghub.elsevier.com/retrieve/pii/B9781856175074X00010 (besucht am 27. 05. 2021).

[5] Betty Prince. *Semiconductor Memories: A Handbook of Design, Manufacture and Application*. Wiley, 1991. 830 S. ISBN: 978-0-471-92465-4. Google Books: VNsmAQAAMAAJ.

[6] Ronald J. Tocci und André Lebel. *Circuits Numériques: Théorie et Applications*. deuxième édition. Ottawa: Editions Reynald Goulet inc. / Dunod, 1996.

[7] John F. Wakerly. *Digital Design: Principles And Practices*. 3rd edition. Prentice-Hall, 2008. ISBN: 0-13-082599-9.

[8] John F. Wakerly. *Digital Design: Principles and Practices*. 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2000. 949 S. ISBN: 978-0-13-769191-3.

Akronyme

EEPROM **E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory. 11, 13

EPROM **E**rasable **P**rogrammable **R**ead **O**nly **M**emory. 9–11

OTP-ROM **O**ne **T**ime **P**rogrammable **R**ead **O**nly **M**emory. 10

PROM **P**rogrammable **R**ead **O**nly **M**emory. 8–10, 12

ROM **R**ead **O**nly **M**emory. 3, 4, 6–10, 12, 13

SCL **S**erial **Cl**ock. 13

SDA **S**erial **Da**ta. 13

UV **U**ltra–**V**iolet. 10, 11

WP **W**rite **P**rotect. 13