Flash-Speichermedien

Physikalische Grundlagen

18.11.2008

Dennis Runz
(dennis.runz@gmx.de)

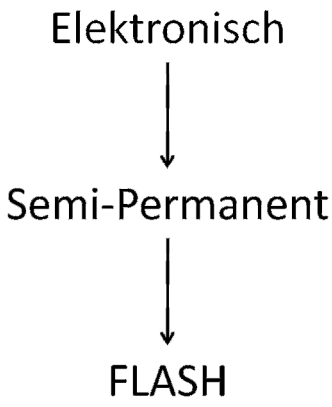
Seminar Speichermedien WS0809
Universität Heidelberg
Betreuung: Olga Mordvinova, Julian M. Kunkel

Gliederung

1. Einleitung
2. Grundlagen
   - MOSFET / Floating Gate MOSFET
   - EPROM / EEPROM / Flash-EEPROM
   - SLC MLC / Aufbau
3. NOR Flash
4. NAND Flash
   - Wear-Leveling
5. NAND vs. NOR

Einleitung

Einordnung Taxonomie



Einsatzbereiche

Mass Storage

Code Execution

Einsatzbereiche

Mass Storage

Code Execution

Einsatzbereiche

Mass Storage

Code Execution

Geschichte

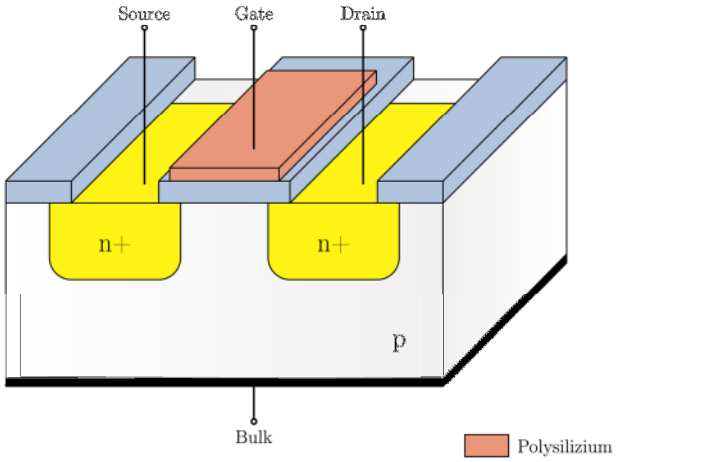
- NAND- und NOR-Flash erfunden ca. 1980 von Toshiba
- Erster kommerzielle NOR-Flash Speicher 1988 von Intel
- Erste NAND-Flash Speicherkarte (SmartMedia) 1995 von Toshiba

Grundlagen

Allgemeine elektronische Speicher
und Flash-Grundlagen

MOSFET

- „Metal Oxide Semiconductor Field Effect Transistor“



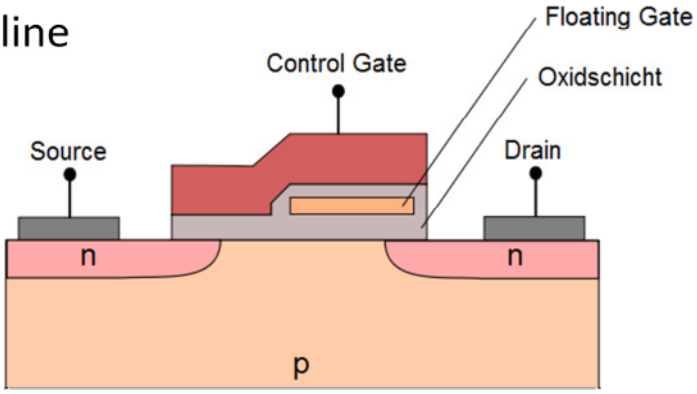
Dennis Runz 10 / 48

Beim Anlegen von Spannung am Gate entsteht ein elektrisches Feld, was eine Anreicherung von Minoritätsladungsträgern unter dem Gate bewrikt. Ab einer bestimmten Schwellenspannung wird die Source-Drain-Strecke unter dem Gate n-leitend, es kann ein Strom fließen.

„Metall-Oxid-Halbleiter-FeldEffektTransistor“: Der Name stammt noch aus der Zeit, in der Aluminiumoxid als Gate-Material verwendet wurde. Heute sind Materialien wie Polysilizium gängig.

Floating-Gate MOSFET

- Grundlage für alle hier behandelten el. Speicher
- Control Gate -> Wordline
- Drain -> Bitline



Dennis Runz 11 / 48

Zusätzlich gibt es hier das sog. Floating Gate. Der Name kommt von der Tatsache, dass es komplett mit einer isolierenden Oxidschicht umgeben ist, also quasi darin schwimmt. Das Floating Gate wird hierbei als Ladungsspeicher verwendet, wodurch sich auch bei abgeschalteter Stromzufuhr Informationen halten lassen (nicht flüchtig).
Die Ladung im Floating Gate kodiert hier direkt die bzw. das gespeicherte Bit. Befindet sich Ladung im FG, wirkt diese der angelegten Spannung am Control Gate entgegen und der Transistor bleibt geschlossen.
Ist hingegen keine Ladung im FG, wird durch anlegen von Spannung am Control Gate die sog. Source-Drain Strecke leitend und es kann ein Strom fließen. Der Transistor ist geöffnet.

Im folgenden werden wir immer annehmen, dass Ladung im FG eine logische 0 bedeutet (Transistor geschlossen) und keine Ladung eine logische 1 (Transistor geöffnet).
Es ist allerdings nicht zwingend bei jedem realen Speicher so.

Das Control Gate wird mit der sog. Wordline verbunden. An sie wird die dekodierte Adresse angelegt, um so z.B. einzelne Zellen zu aktivieren bzw. auszuwählen.
Das Drain wird mit der sog. Bitline verbunden. Sie stellt den Datenbus dar.
Später sind ja mehrere Zellen zusammen auf einem Speicher. Hat man z.B. einen 8 Bit breiten Datenbus, bilden 8 FG-MOSFETs ein Byte. Werden diese 8 Zellen dann durch die dekodierte Adresse an der Wordline ausgewählt, liegt das durch diese 8 Zellen gespeicherte Byte dann an der Bitline an.

Kenngrößen

- Endurance
  - Maximale Anzahl an Schreibzyklen pro Speicherzelle
  - Flash-Speicher hat also eine begrenzte Lebensdauer!
- Retention
  - Minimale Zeit, in der eine gespeicherte Information erhalten bleibt.

Typische Werte für Endurance bei nichtflüchtigen Speichern, 10.000 bis 1 Mio Schreibzyklen (pro Block).
Typische Werte für Retention, ca. 20 Jahre.

Prinzipielle Eigenschaften von Flash

- Schreiben in 2 Vorgänge unterteilt:
  - Programmieren: 1 -> 0 (logisch)
  - Löschen: 0 -> 1 (logisch, nur Blockweise)
- NAND und NOR Flash verwenden untersch. Techniken beim Programmieren/Löschen
  - Resultiert in unterschiedlichen Schreibgeschwindigkeiten.

Bei einem Programmiervorgang (einer Speicherzelle) wird das gespeicherte Bit auf 0 gesetzt (Ladung in das Floating Gate „befördern“).
Beim Löschvorgang auf 1 (Ladung aus dem Floating Gate „entfernen“).

Lösch-/Programmierverfahren

Hot Carrier Injection

- Hot Electron Injection
  - n-type auf p-Substrat
- Hot Hole Injection (selten)
  - p-type auf n-Substrat
  - Sehr langsam
- Lawinendurchbruch (ineffizienter)

Fowler-Nordheim Tunneling

- Uniform Tunneling
  - Langsamer
- Drain Side Tunneling
  - Schneller aber beschädigt Oxidschicht stärker (Endurance)
- Nutzt Tunneleffekt aus (effizienter)

Dennis Runz 14 / 48

Es gibt zwei hauptsächlich Verwendete Techniken zum Programmieren bzw. Löschen von elektronischen Speichern. Die Folie bezieht sich hauptsächlich auf Flash-Speicher. Allerdings werden die Techniken auch bei normalen EEPROMs verwendet. Bei (UV-)EPROMs wird die bereits erwähnte UV-Emission verwendet.Generell kann Tunneling sowohl zum Programmieren (Tunnel Injection) als auch zum Löschen (Tunnel Release) verwendet werden. Die Hot Carrier Injection wird z.B. bei NOR Flash zum Programmieren verwendet.

Ein Verfahren ist die sog. **„Hot Carrier Injection“** oder auch „Channel Hot Carrier Injection“ (CHCI). Hierbei wird dem Ladungsträger (abhängig von der verwendeten Halbleiter Bauweise – NMOS: Elektronen, PMOS: Defektelektronen/Löcher) durch eine Hohe Spannung relativ viel Energie zugefügt, d.h. man bekommt hier schnelle (heiße / hot) Elektronen welche an der Drain Seite sozusagen auf die Oxidschicht Richtung Floating Gate geschossen werden. Die Elektronen dringen hierbei in die Oxidschicht ein, gelangen aber nicht in das Floating Gate. Sind genügend Elektronen vorhanden können diese einen sog. Lawinendurchbruch auf der Drain-Seite herbeiführen, was bewirkt dass die Ladung ins Floating Gate gelangt.

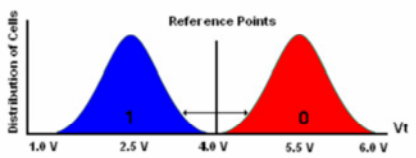
Ein weiteres Verfahren ist das sog. „**Fowler-Nordheim Tunneling“**. Hier wird ein quantenmechanischer Effekt, der sog. Tunneleffekt ausgenutzt. Daher benötig man hier nicht so hohe Spannungen; die Ladungsträger springen oder tunneln quasi in das Floating Gate -> höhere Effizienz. Es gibt hier wieder 2 konkrete Verfahren. Bei „Uniform Tunneling“ werden die Ladungsträger vom Substrat aus in das Floating Gate befördert. Bei „Drain Side“ Tunneling ähnlich wie bei der Carrier Injection von der Drain Seite aus.

Bei allen Verfahren degeneriert sich die Oxidschicht bei jedem Löschvorgang. Da sich beim Drain Side Tunneling der Elektronenbeschuss auf eine kleine Fläche der Oxidschicht konzentriert, degeneriert hier die Oxidschicht schneller, was zu einer niedrigeren Endurance führt. Vorteil ist aber eine kürzere Zeit zum Löschen der Zelle.

Hauptgrund warum heutige Halbleiter als n-type auf p-Substrat gefertigt werden ist, weil Hot Hole Injection aufgrund der Masse von Löchern sehr ineffizient bzw. langsam ist.

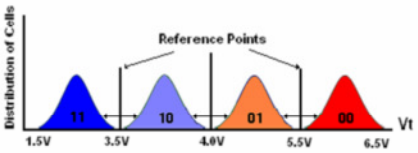
SLC / MLC

SLC (Single Level Cell)



- Speicher nur ein Bit Information
- Schneller, Verbraucht weniger Energie, höhere Endurance
- Industrieller Einsatz (Wärme)

MLC (Multi Level Cell)



- Speichert mehr als ein Bit (höhere Datendichte)
- Wahrscheinlichkeit für Fehler erhöht (Fehlerkorrekturalgorithmen benötigt: BCH codes)
- Langsamer, Verbraucht mehr Energie, niedrigere Endurance
- Kommerzieller Einsatz (Wärme)
- Präzise Ladungsinjektion und -Sensorik Notwendig

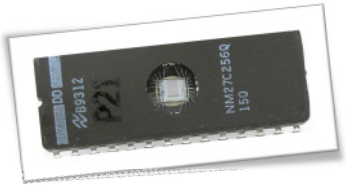
Dennis Runz 15 / 48

MLC: Hier wird nicht nur ein “Level of Charge” (daher der Name) im Floating Gate einer einzelnen Speicherzelle verwendet, sondern mehrere. Beispielsweise bei MLC NAND Speicher und 4 „Levels of Charge“ könnte 2 Bit an Information gespeichert werden (2^2 = 4).

Heute gängig sind MLC Chips mit 2 Bit pro Zelle. In Zukunft werden NAND MLC Chips 3 bzw. 4 Bit speichern können.

Bose-Chaudhuri-Hocquenghem (BCH)

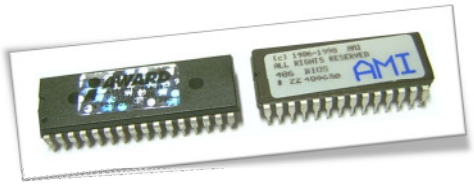
(UV-)EPROM



- „UltraViolet Erasable Programmable Read-Only Memory “
- Einfache Massenproduktion
- Matrix aus Floating-Gate MOSFETs
- Chip wird „parallel gelöscht“
  - Aber sehr langsam (mehrere Sekunden)
- Löschverfahren: UV Emission (UV-Belichtung an Fenster, max. 100 mal möglich)
- Programmierverfahren: Hot Carrier Injection

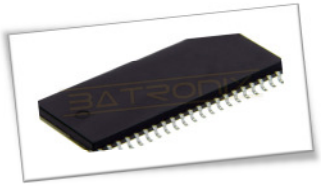
Durch UV-Licht werden Elektron-Loch-Paare im Oxid erzeugt, die im Feld getrennt werden. Sie entladen/laden das Zwischengate (langsam)

EEPROM



- „Electrically Erasable Programmable Read-Only Memory“
- Teure Produktion
- Matrix aus Floating-Gate MOSFETs
- Byteweise beschreib- und löschbar
  - Aber langsam! (~10 ms)
- Lösch-/Programmierverfahren: Fowler-Nordheim Tunneling
- Wird langfristig durch Flash-Speicher ersetzt

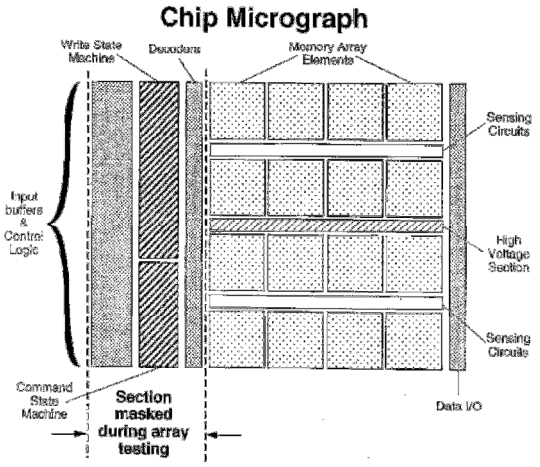
Flash-EEPROM



- Etabliert und billig
- NAND und NOR Technologie
- Kompletter Chip kann „parallel“ gelöscht werden
- Schneller beschreibbar als reine EEPROMs
- Im Prinzip gleich EPROM. Unterschiede im Löschvorgang.
- Lösch-/Programmierverfahren: Hot Carrier Injection, Fowler Nordheim Tunneling

Flash-Speicher: Aufbau

- Interne „Charge Pumps“ zum erzeugen der hohen Spannungen für Schreiben/Löschen
- Array aus Speicherzellen
- Ansteuerungslogik/Controller
- Datenbus bzw. I/O-Interface

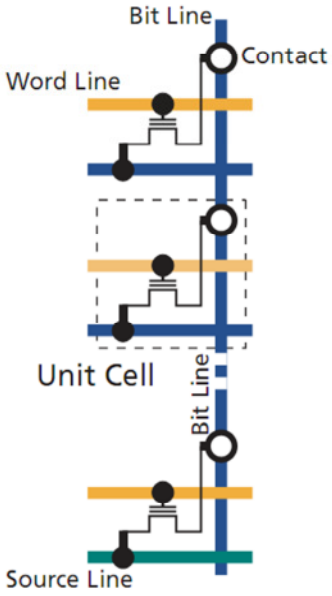


NOR

Die NOR-Flash Architektur

NOR: Aufbau

- Bytes sind zu Blöcken zusammengefasst
  - Lese- und Schreibzugriff auf Bytes wahlfrei
  - Beim Schreiben jedoch vorheriges Löschen des Blockes notwendig (Flash-Besonderheit)
- Bytes parallel geschaltet (separate Adress- und Datenleitungen für jedes Byte)

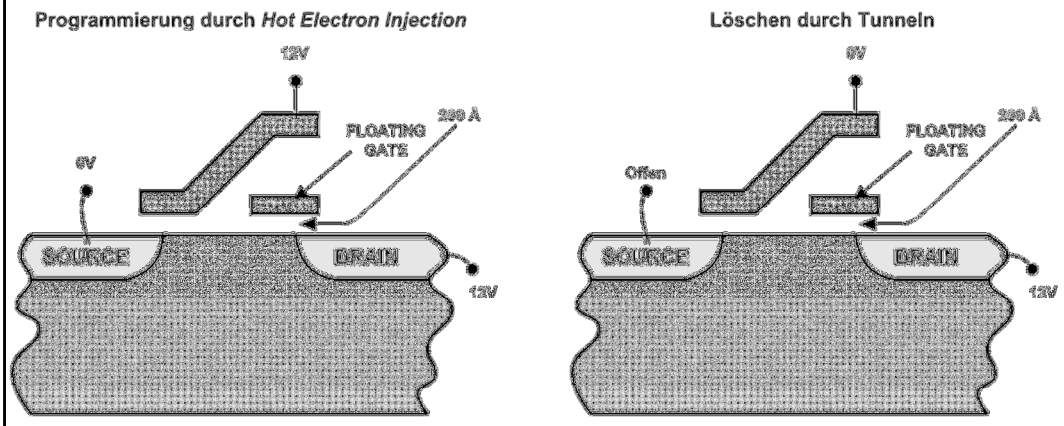


Ein Flash-Speicherchip kann bei einem Programmiervorgang nur einen Übergang von einer logischen 1 auf eine logische 0 bewerkstelligen. Soll also z.B. ein Byte innerhalb eines Blockes geschrieben werden, so muss aus genanntem Grund erst ein Löschen des gesamten Blockes erfolgen (ein Löschen bei Flash-Speichern kann architekturbedingt nur auf einem gesamten Block angewendet werden), wobei wieder alle Bits auf 1 gesetzt werden um dann die notwendigen Bits wieder auf 0 setzten zu können (abhängig vom zu schreibenden Wort).

NOR: Programmieren / Löschen

- NOR Programmieren
  - „Hot Electron Injection“
  - Energieverbrauch steigt mit zunehmender Zahl Speicherzellen
  - Nur Byte- bzw. Wortweise programmierbar -> Schreibzeiten pro Byte länger
- NOR Löschen
  - „Fowler-Nordheim tunneling“

NOR: Programmieren / Löschen



Dennis Runz 23 / 48

NOR Schreibvorgang: Es wird das sog. “Channel Hot Electron Injection” Verfahren verwendet. Man verleiht den Elektronen durch den erzeugten Strom zwischen Source und Drain genügend Energie, sodass einige von Ihnen durch die Oxidschicht in das Floating Gate kommen und so dort eine negative Ladung verursachen. Diese Ladung wirkt der Spannung am Control Gate entgegen und es kann kein Strom mehr zwischen Source und Drain fließen, der Transistor ist geschlossen (logische 0).

NOR-Löschen: Durch die beim Löschvorgang erzeugte positive Ladung im Floating Gate (überwiegende Defektelektronen bzw. Löcher) kann bei einem Lesevorgang wieder Strom zwischen Source und Drain fließen. Der Transistor ist geöffnet (logische 1).

NOR: Zugriff

- Keine „GlueLogic“ erforderlich sofern Daten- und Adressbus richtig geschaltet
- XIP (eXecute In Place) möglich

- Lesen und Schreiben Byteweise (Random Access)
- Löschen Blockweise

NAND

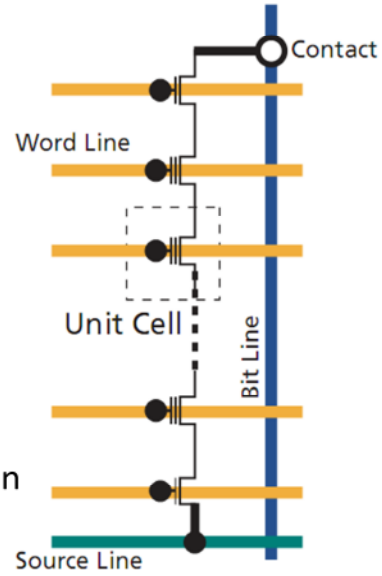
Die NAND-Flash Architektur

NAND: Intention

- Höhere Speicherdichte
  - Aber: Verlust von „Random Access“
  - Nochmalige Erhöhung durch MLC
- Geringere Kosten
  - Kleinere Zellengröße, dadurch kleinere Chipgröße und somit geringere Kosten pro Bit

NAND: Aufbau

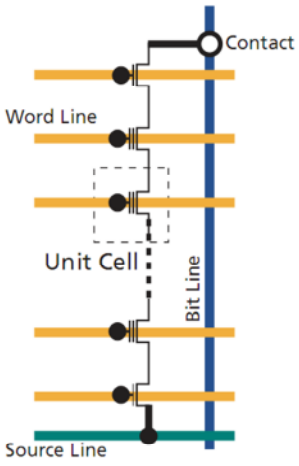
- Mehrere Speicherzellen zu Page zusammengefasst (512B – 4KB)
- Mehrere Pages zu Block zusammengefasst (16KB – 512KB)
- Pages nur einmal beschreibbar. Danach wieder Löschvorgang auf Block notwendig.
  - Aber: Erweiterte Schreibalgorithmen können optimieren (1111->1011->1000->0000 # 1111 …)



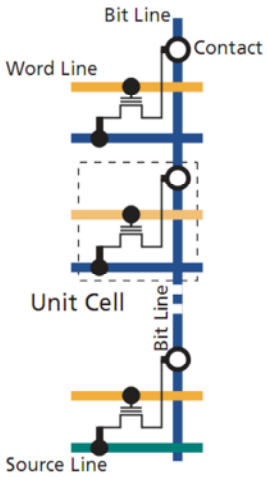
Typische Blockgrößen:
32 Pages zu je 512 bytes -> Blockgröße von 16 kb
64 Pages zu je 2048 bytes -> Blockgröße von 128 kb
64 Pages zu je 4096 bytes -> Blockgröße von 256 kb
128 Pages zu je 4096 bytes -> Blockgröße von 512 kb

Erinnerung

NAND



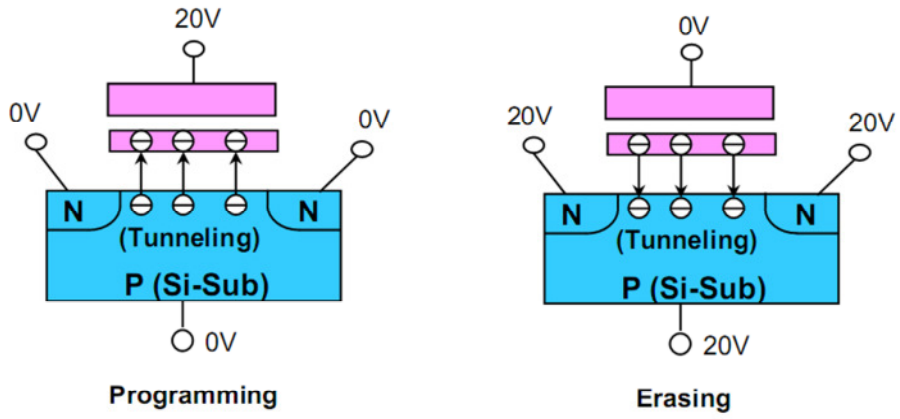
NOR



NAND: Programmieren / Löschen

- NAND Programmieren
  - „Fowler-Nordheim tunneling“ „Tunnel injection“
  - Energieverbrauch steigt nicht signifikant mit zunehmender Zahl Speicherzellen
  - Viele NAND-Flashes parallel beschreibbar -> Schreibzeit kurz
- NAND Löschen
  - „Fowler-Nordheim tunneling“ „Tunnel Release“

NAND: Programmieren / Löschen



NAND: Zugriff

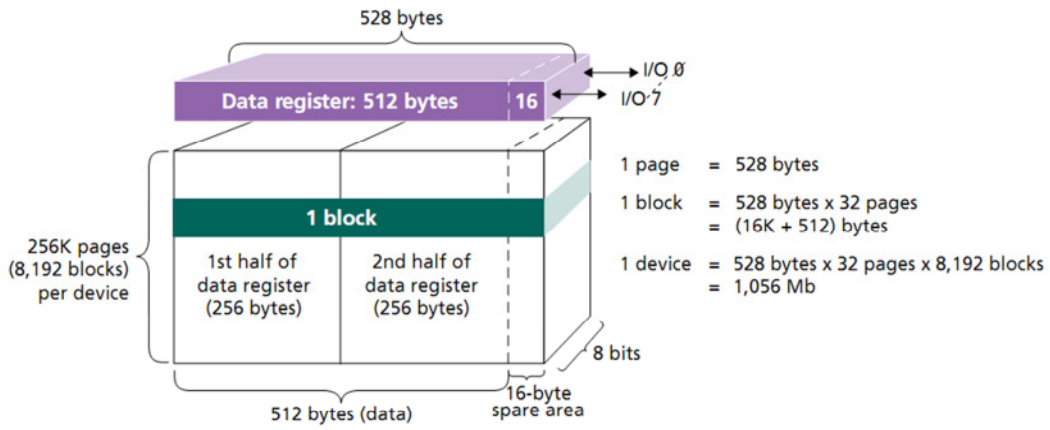
- Zugriff ähnlich zu Block-Devices aber „Glue-Logic“ erforderlich
- Kein XIP (eXecute In Place) möglich, Daten müssen vorher in RAM geladen werden
- Generell höherer Aufwand im Vgl. zu NOR

- Lesen und Schreiben Seitenweise
- Löschen Blockweise

Will man Software bzw. Code auf einem NAND-Flash ausführen, so muss dieser erst in den Hauptspeicher geladen werden, da kein Wahlfreier Zugriff möglich ist. Hierzu werden Virtuelle Speicherverwaltungstechnologien verwendet (Paging). Der Code wird dann direkt auf dem Hauptspeicher ausgeführt.
Manchmal werden auch Kombinationen von NOR und NAND Speicher verwendet, wobei der NOR Speicher für Code und der NAND Speicher für Daten, Dateisystem etc. verwendet wird.

Technologien wie „oneNAND“ vereinen NAND/NOR Eigenschaften.

NAND: Aufbau 1GB Array

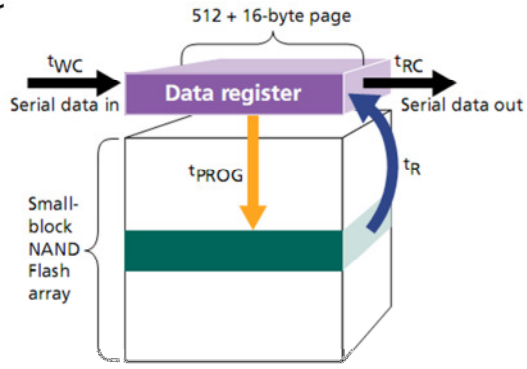


Es gibt Small Block NAND Flashes und Large Block NAND Flashes.
Small Block: Jeder Block besteht aus 32 Pages zu je (512 + 16) Byte. Macht bei 1GB 8192 Blöcke. [hier betrachtet]
Large Block: Jeder Block besteht aus 64 Pages zu je (2048 + 64) Byte. Macht bei 1GB 1024 Blöcke.

Generell:
Small Block: <= 1GB Speicherdichte
Large Block: >= 1GB Speicherdichte

NAND: Page Read

1. Command phase
2. Address phase
3. Data Transfer phase
4. Read Out phase



Dennis Runz 33 / 48

Command phase:
„read“ Command-Byte wir an das I/O Interface des NAND-Speichern angelegt -> Versetzt den Speicher in den Lese-Modus (Command-Byte wird im Command-Register abgelegt).
Bei Small Block NAND Flashes wird hier zwischen mehreren „read commands“ unterschieden. Hierbei steht jedes Command für einen anderen „read mode“.

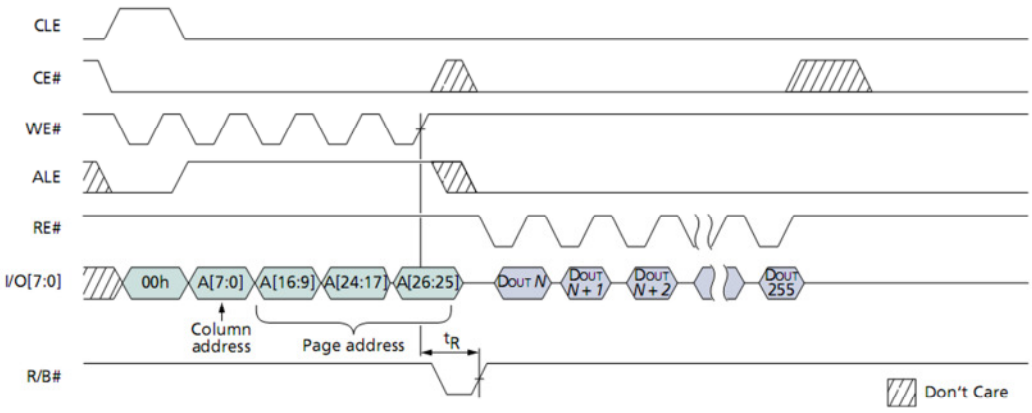
Hat man z.B. eine Pagegröße von 528 Bytes, so werden folgende read modes verwendet (Toshiba Speicher):
„read mode 1“: Adressierung der Bytes 0 bis 255 (command 00h)
„read mode 2“: Adressierung der Bytes 256 bis 511 (command 01h)
„read mode 3“: Adressierung der Bytes 512 bis 527 (command 50h - Spare Bereich)

Address phase:
Die Spalten und Seitenadresse wird an das I/O Interface angelegt.
Die jeweilige komplette Page (in der die angeforderten Daten liegen) wird in das Output Register geladen.

Data Transfer phase:
Die angeforderten Daten werden vom Speicher-Array in das Daten-Register transferiert.

Read Out phase:
Daten liegen nun im Daten-Register zum Lesen bereit. Zum sequentiellen byteweisen Lesen, wird nun mit jedem Impuls am ReadEnable (RE) Pin ein Byte gelesen.
Wird das Ende der Seite erreicht wird automatisch zur nächsten gesprungen.

NAND: Page Read



NAND Flashes haben keine dedizierten Adresspins. Die Adresse wird an einem Shared I/O Bus angelegt, an dem auch Commands und Daten angelegt werden.

CLE (Command Latch Enable)
CE (Chip Enable)
WE (Write Enable)
ALE (Address Latch Enable)
RE (Read Enable)
R/B (Ready/Busy)

NAND: Page Read

Process	Repetitions	Cycle Time	Total Time	
Command latch (00h and/or 01)	2	50	100	ns
Address latch	4	50	200	ns
Command latch (30h or 50h)	1	50	50	ns
R/B# LOW (tR)	1	15	15	µs
Data output cycles	528	50		ns
Data output cycles total time			26.4	µs
Total time to read a page			41.75	µs
Data rate			12.65	MB/s

$$\frac{512 + 16\,Byte}{41{,}75\,\mu s} = \frac{528 * 10^{-6} MB}{41{,}75 * 10^{-6} s} = 12.65\ MB/s$$

NAND: Endurance Defektmanagement

- Ursache für Endurance: Hohe Spannung am Control-Gate beim Schreiben/Löschen -> Oxidschicht reduziert sich
- Defektmanagement in (Hardware und Software):
  - Fehlertolerante Codes (z.B. Hamming Codes)
  - Zusätzliche Bits für jeden Block
  - Bad Block Management
  - Wear-Leveling

Beim Programmieren einer Flash-Speicherzelle werden relativ hohe Spannungen am Control Gate benötigt, um Ladungsträger durch die Oxidschicht in das Floating Gate befördern zu können. Dabei degeneriert die isolierende Oxidschicht um das Floating Gate. Ist diese Schicht zu sehr degeneriert ist die Zelle unbrauchbar. Der Flash-Speicher an sich ist jedoch weiterhin normal benutzbar, wenn auch mit reduzierter Speicherkapazität. Dazu sind Defektmanagementmechanismen notwendig.

Bei NAND-Flashes ist es nicht unüblich, dass schon frisch produzierte Chips fehlerhafte Blöcke (Bad Blocks) enthalten. Diese werden dann schon von Beginn an als defekt markiert. Aufgrund von begrenzten Schreib/Löschzyklen (Endurance) bei Flash-Speichern ist es ebenso notwendig Bad Block Detection während des Betriebes zu betreiben. In diesem Fall wird ein Block für gewöhnlich als defekt markiert, sobald ein Fehler beim Löschen eines Blockes oder beim Programmieren einer Seite auftritt. Um solche Fehler zu detektieren gibt es ein sog. „Status-Command“, also ein Befehl mit dem man ermitteln kann ob die zuletzt ausgeführte Lösch- bzw. Programmieroperation erfolgreich war oder nicht. Solche Operationen werden automatisch vom Chip nach jeder Durchführung verifziert. Nach dem Löschen wird also geprüft ob alle Bits im Block 1 sind und beim Schreiben ob alle zu programmierenden Nullen korrekt programmiert wurden (Einsen werden ignoriert).

Ein Nebeneffekt der bei zunehmender Anzahl von Schreib und Löschzyklen auftritt, ist dass die Dauer der Programmierung zunehmend abnimmt. (Reduktion der Oxidschicht)

Die Bad Block Tabelle wird entweder in einem funktionstüchtigen Block auf dem Flash-Chip selbst abgelegt, oder extern, z.B. im RAM (Systemabhängig).
Außerdem sei erwähnt, dass ein defekter Block die funktionierenden Blöcke nicht beeinträchtigt, da jeder Block von den Bit-Lines durch eigene „Block-Selection“ Transistoren isoliert werden kann.

Unterkategorie NAND

NAND WEAR-LEVELING

Dennis Runz

WL: Klarstellung

- Gleichmäßige Verteilung von Daten, sodass alle Blöcke gleichmäßig belastet werden

Beispiel: Flash-Speicher mit 4096 Blöcken. 3 Dateien über 200 Blöcke verteilt. Alle 10 Minuten wird eines der Files neu überschrieben.

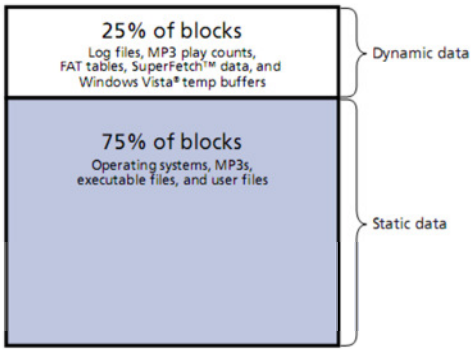
- Ohne Wear-Leveling (200 Blöcke verwendet)
  - Wear-Out schon in unter einem Jahr!
- Mit Wear-Leveling (Alle Blöcke verwendet)
  - Wear-Out erst in mehr als 15 Jahren.

„Wear-Out“ bedeutet hier nicht dass der komplette Speicher unbrauchbar wird, sondern nur bestimmte Blöcke unbrauchbar werden. Im ersten Fall sind das 200 Blöcke, d.h. nach einem Jahr hat der Flash-Speicher 200 Blöcke (Speicherplatz) weniger zur Verfügung, wo hingegen im zweiten Fall die komplette Kapazität über mehr als 15 Jahre erhalten bleibt.

Da Flash-Speicherzellen irgendwann kaputt gehen sind Wear-Leveling Verfahren notwendig um eine gleichmäßige Belastung der Speicherblöcke zu erzielen. Ein praktisches Beispiel für die Problematik bei Dateisystemen ist z.B. FAT. Hier wird z.B. bei jeder Änderung einer Datei eines Ordner etc. die FAT (File Allocation Table) bzw. der Bereich an dem die Ordner verwaltet werden aktualisiert. Diese Bereiche würden sehr viel stärker belastet werden als alle anderen Bereiche auf dem Flash-Chip. Daher implementieren viele NAND-Speicher Wear-Leveling-Algorithmen bereits auf dem Controller. Hierbei werden die logischen Adressen sukzessive auf unterschiedliche Physikalische Adressen gemapped um eben eine solche Gleichverteilung der Belastung zu erreichen. Es gibt auch Dateisysteme, die Wear-Leveling bereits integriert haben, z.B. JFFS2 (Journaling Flash File System 2) und YAFFS (Yet Another Flash File System).

WL: Dynamic Wear-Leveling

- Wear-Leveling nur auf Dynamischen Daten
- Block mit dem kleinsten „Erase-Count“ wird für nächste Schreiboperation ausgewählt



WL: Static Wear-Leveling

- Auch statische Daten berücksichtigt
- Verschiebt statische Daten bei Bedarf in Bereiche mit hohem Wear-Out um für dynamische Daten „Platz“ zu schaffen
- Erreicht gleichmäßige Belastung des gesamten Chips

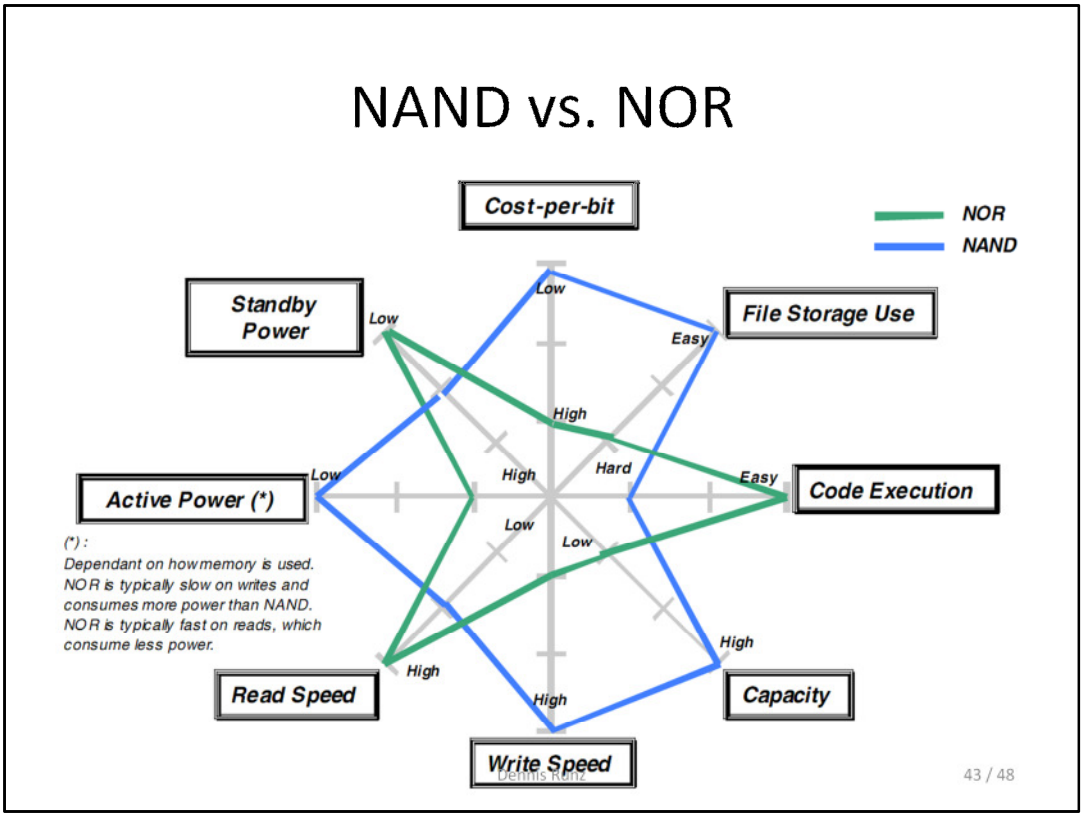
Hier werden auch statische Daten berücksichtigt. Blöcke mit statischen Daten und geringem Erase-Count (im Vergleich zu anderen Blöcken) werden in Bereiche mit hohem Erase-Count verschoben, damit dynamische Daten wieder Blöcke mit geringerem Erase-Count zugeteilt werden können. So erreicht man eine gleichmäßige Auslastung des gesamten Chips, und nicht nur dem Bereich mit dynamischen Daten, wie es beim Dynamic Wear-Leveling der Fall ist.

WL: Dynamic vs. Static

	Vorteile	Nachteile
Static	• Maximale Lebensdauer • Sehr robust • Effiziente Nutzung des Speicher-Arrays	• Mehr Controller-Overhead • Kann Schreibvorgänge verlangsamen • Höherer Energieverbrauch • Komplizierter umzusetzen
Dynamic	• Besser als kein Wear-Leveling • Leichter umzusetzen • Kein Einfluss auf Performance	• Manchmal keine Auswirkung auf Lebensdauer

NAND vs. NOR

Direkter Vergleich der Technologien



NAND: FN Tunneling für Programmieren und paralleles schreiben -> geringer Energieverbrauch
NOR: Hot Electron Injection für Programmieren und langsames schreiben -> hoher Energieverbrauch.

NAND vs. NOR

	SLC NAND Flash (x8)	MLC NAND Flash (x8)		MLC NOR Flash (x16)	
Density	512Mbit to 4Gbit	1Gbit to 16Gbit	+	16Mbit to 1Gbit	-
Read Speed	24 MB/s	18.6 MB/s	-	103 MB/s	+
Write Speed	8.0 MB/s	2.4 MB/s	+	0.47 MB/s	-
Erase Time	2.0 ms	2.0 ms	+	900 ms	-
Costs	Billig	Sehr billig	+	Teuer	-
Chipsize			+		-
Endurance	1 Mio	100.000 - 1 Mio	+	10.000 - 100.000	-
Interface	I/O – indirect access	I/O – indirect access		Random access	
Application	Program/Data mass storage	Program/Data mass storage		eXecuteInPlace	

NAND- NOR-Flash Spezifikationen (Toshiba 2006)

Die Angaben bei Endurance bedeuten z.B. für NAND: Maximal 1 Mio. Löschvorgänge pro Block.

Zusammenfassung

- NAND und NOR Architektur mit unterschiedlichen Eigenschaften, Aufbau und Einsatzgebiet
- Floating-Gate Transistor speichert 1-Bit Information (Speicherzelle)
- MLC und SLC Speicherzellen
- Endurance und Retention sind maßgebliche Einschränkungen und erfordern Defektmanagement und Wear-Leveling

Aussicht Vortrag Flash2

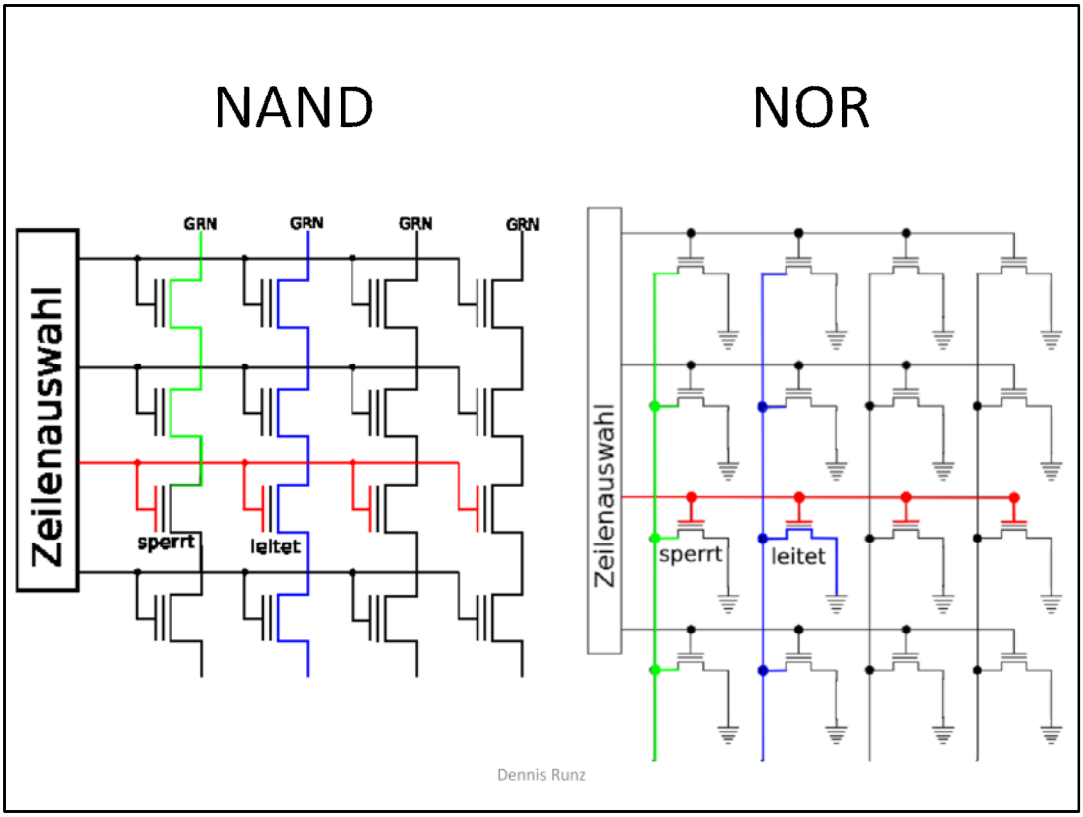
- SSD (Solid State Drives)
- Speicherkarten
- USB Stick
- Dateisysteme
- Benchmarks

Quellen:

- http://de.wikipedia.org/wiki/EEPROM
- http://de.wikipedia.org/wiki/Feldeffekttransistor
- http://de.wikipedia.org/wiki/Floating_Gate
- http://de.wikipedia.org/wiki/CMOS
- http://de.wikipedia.org/wiki/Flash-Speicher
- http://de.wikipedia.org/wiki/NAND-Flash
- http://de.wikipedia.org/wiki/NOR-Flash
- http://en.wikipedia.org/wiki/Hot_carrier_injection
- http://en.wikipedia.org/wiki/Single-level_cell
- http://en.wikipedia.org/wiki/Multi-level_cell
- http://www.toshiba.com/taec/components/Generic/Memory_Resources/NANDvsNOR.pdf
- http://www.dataio.com/pdf/NAND/Toshiba/NandDesignGuide.pdf.pdf
- http://klabs.org/richcontent/MAPLDCon98/Papers/c4_miyahira.doc
- http://www.samsung.com/global/business/semiconductor/products/flash/FlashApplicationNote.html
- http://www2.electronicproducts.com/NAND_vs_NOR_flash_technology-article-FEBMSY1-FEB2002.aspx
- http://gunn.winterwolf.co.uk/archive/hot_e_inj
- http://download.micron.com/pdf/technotes/nand/tn2907.pdf
- http://download.micron.com/pdf/technotes/nand/tn2942_nand_wear_leveling.pdf
- http://download.micron.com/pdf/technotes/nand/tn2919.pdf

Quellen:

- http://aplawrence.com/Makwana/nonvolmem.html (Programming Methods, Endurance, Retention Physics)
- http://sus.ziti.uni-heidelberg.de/Lehre/DSTVorlesung03/DST_Speicher.pdf
- http://www.supertalent.com/datasheets/SLC_vs_MLC%20whitepaper.pdf
- http://www.hitequest.com/Kiss/Flash_terms.htm
- http://www.virtualuniversity.ch/elektronik/digital/dvs/28.html


NAND
NOR
Zeilenauswahl
GRN
GRN
GRN
GRN
sperrt
leitet
Zeilenauswahl
sperrt
leitet
Dennis Runz