Springer-Lehrbuch

Digitaltechnik

Ein Lehr- und Übungsbuch

von
Roland Woitowitz, Klaus Urbanski

erweitert, überarbeitet

Springer 2007

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 540 73672 1

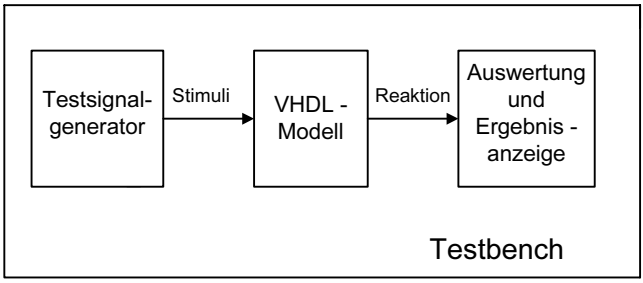
Zu Inhaltsverzeichnis

en von VHDL-Modellen

ationstechniken

hwerpunkt lag bisher auf VHDL-Modellen, die mit Hilfe programmierbarer synthetisiert werden. In erster Linie wurde VHDL als Entwurfssprache einge- m Folgenden soll die VHDL-Syntax zur Bildung von Testumgebungen ver- t werden. Hierzu werden nichtsynthesefähige VHDL-Modelle aufgestellt, mit Hilfe VHDL-Komponenten gestestet werden können. Der zu testende VHDL- rf wird in eine Testbench eingebunden. Der Name Testbench (Werkbank) wird logie zu einer realen Werkbank gewählt, auf der ein Werkstück getestet wer- m. In VHDL werden die Eingangssignale mit Signalgeneratoren stimuliert und aktion der Ausgangssignale kontrolliert.

einfachste Form einer Simulation ist die interaktive. VHDL-Compiler enthal- ols, um die Porteingänge eines Modells zu stimulieren und die Ergebnisse in enform oder als Signalzeitdiagramm am Bildschirm und/oder Drucker aus- n. Diese Methode eignet sich gut, um einen Überblick über das Verhalten des rfs zu erlangen. Im allgemeinen Fall muss die Eingabe bei einer Änderung holt werden, was zeitaufwendig und fehlerträchtig ist.



: Die Testbench dient zur Erprobung des VHDL-Modells.

estbench hat gegenüber der interaktiven Methode deutliche Vorteile:

e Dokumentation der Testvektoren für Ein- und Ausgänge
chte Wiederverwendbarkeit bei Entwurfsänderungen
tematische Vorgehensweise
nelle Fehlererkennung
gleiche Testbench lässt sich sowohl für das Quellcode-VHDL-Modell als
für das Postlayout-VHDL-Modell verwenden

olgenden werden zwei unterschiedliche Entwürfe für Testbenches vorgestellt:

estvektoren werden innerhalb der Testbench erzeugt
estbench mit Ein- und Ausgabedatei

2
tbench mit Testvektoren

Entity ist sehr einfach aufgebaut, insbesondere enthält sie keine Port- oder Gene-
Anweisungen:

```
y testbench_name is      -- Entity der Testumgebung
estbench_name;
```

Architektur enthält ein Testvektorfeld, das sowohl die stimulierenden Werte für
Eingabeports als auch die am Ausgang erwarteten Logikzustände enthält. Das zu
nde VHDL-Modell muss als Komponente vorliegen. Die compilierte Komponen-
in einem Package in der Work-Library oder in einer benutzerdefinierten Libra-
espeichert. Bei der Ausführung werden in einer For-Schleife die Stimuli eines
ors (hier: vektor.x) den Porteingängen (hier: x) zugewiesen. Nach einer Verzöge-
(hier: 30 ns), die die Durchlaufzeit des Schaltkreises simuliert, wird die Reakti-
es getesteten VHDL-Modells (device under test) am Portausgang (hier: y) vergli-
mit dem Wert des Testvektors (hier: vektor.y). Mit Hilfe der Assertion- und Re-
Anweisung (Kap. 4.7.1) lässt sich im Fehlerfall während der Ausführung der
bench (Run-Kommando) ein Report am Monitor ausgeben. Weiterhin kann über
Variable Fehler ein abschließender Report am Ende des Tests ausgegeben wer-

las schon bekannte VHDL-Modell zu Tab. 2.11 (Kap. 4.5.4.8) ist in einem Bei-
eine Testbench angegeben. In dem Package "tabelle_pack.vhd" wird die Kompo-
e tab2_11a deklariert. Nach der Compilierung wird sie in der Work-Library abge-
und gespeichert. Es kann nun über die Use-Anweisung auf die Komponente
_11a zugegriffen werden.

```
elle_pack.vhd
y ieee;
eee.std_logic_1164.all;

ge tabelle_pack is  -- Package tabelle_pack.vhd
onent tab2_11a      -- Component-Deklaration der Tabelle 2.11
ort (
: in std_logic_vector (1 to 4);
: out std_logic_vector (1 to 2));
omponent;
```

```
.std_logic_1164.all;
ab2_11a is port (
n std_logic_vector (1 to 4);
ut std_logic_vector (1 to 2));
2_11a;

ture sequent_verhalten of tab2_11a is

: process(x) begin      -- Anordnung entspricht der Wahrheitstabelle
x is
en "0000" => y <= "11";
en "0001" => y <= "11";
en "0010" => y <= "11";
en "0011" => y <= "0-";
en "1000" => y <= "0-";
en "1001" => y <= "0-";
en "1010" => y <= "11";
en "1011" => y <= "01";          -- Fehler "01" statt "11"
en "1100" => y <= "0-";
en "1110" => y <= "0-";
en others => y <= "00";          -- Kombinationen, die "00" ergeben
ase;
ocess tabelle; -- am Prozessende erhält y den neuen Wert
uent_verhalten;

bench_tab.vhd
eee;
.std_logic_1164.all;
k.tabelle_pack.all;     -- erlaubt Zugriff auf die Komponente tab_2_11a

stbench is
bench;

ture auto_test of testbench is
al x: std_logic_vector(1 to 4);
al y: std_logic_vector(1 to 2);
t_vektor is record      -- Stimuli und Erwartungswerte → Record
d_logic_vector(1 to 4);
d_logic_vector(1 to 2);
ord;
t_vektor_array is array (natural range <>) of test_vektor;
t test_feld: test_vektor_array:=(
> "0000", y => "11"),
> "0001", y => "11"),
> "0010", y => "11"),
> "0011", y => "0-"),
> "0100", y => "00"),
> "0101", y => "00"),
> "0110", y => "00"),
```

```
x => "1011", y => "11"),
x => "1100", y => "0-"),
x => "1101", y => "00"),
x => "1110", y => "0-"),
x => "1111", y => "00")


     -- instanziieren der Component tab2_11a
ab2_11a port map (x, y);
: process       -- Testvektor zuweisen und pruefen
ariable vektor: test_vektor;
ariable fehler: boolean := false;

n test_feld'range loop
ektor := test_feld(i);
 <= vektor.x;
ait for 30 ns;  -- Verzoegerungszeit abwarten
 y /= vektor.y then        -- Ergebnis ueberpruefen
     assert false
     report "Ergebnis falsch";
     fehler := true;
nd if;
oop;

ssert not fehler              -- Ausgabe eines Reports
eport "Test ist fehlerhaft"
everity note;
ssert fehler
eport "Test ist o.K"
everity note;
ait;
rocess testen;
uto_test;
```

der Ausführung der Testbench mit ModelSim Xilinx [http://www.xilinx.com] en folgende Schritte durchgeführt:

Compilieren des Package tabelle_pack.vhd. Die compilierte Datei wird automatisch in der Work-Library gespeichert und steht für weitere Anwendungen zur Verfügung.

Compilieren der Testbench test_bench_tab.vhd.

Laden der einzelnen Komponenten

Ausführen der Simulation mit dem Kommando "run –all"

zug der Monitorausgabe bei der Ausführung mit ModelSim:

```
work.testbench
```

```
ng work.testbench(auto_test)
ng work.tab2_11a(sequent_verhalten)

ror: Ergebnis falsch
e: 360 ns  Iteration: 0  Instance: /testbench
te: Test ist fehlerhaft
e: 480 ns  Iteration: 0  Instance: /testbench
```

liesem Beispiel ein Fehler in dem zu testenden VHDL-Modell tab2_11a vor-
n ist, werden bei der Ausführung der Testbench mit ModelSim Xilinx Fehler-
ngen ausgegeben.

ench mit Ein- und Ausgabedatei

Verwendung von Testbenches lässt sich die Vorgabe der Stimulationswerte
Ausgabe der Ergebnisse noch verbessern, indem man Textdateien für die Ein-
sgabe einsetzt. Die Stimuli werden jetzt als Zeichenfolge in einer Eingabeda-
elegt und während der Ausführung über standardisierte Prozeduren eingelesen.
h können die Werte für die Stimulation leicht geändert werden, ohne dass eine
ompilierung der Testbench erfolgen muss. Entsprechend werden die Ergebnis-
rmatierter Form über Ausgabeprozeduren als Text in einer Ausgabedatei ge-
ert, die dann vom Anwender leichter ausgewertet werden kann.

rwendung der Ein- und Ausgabeprozeduren ist mit größerem Aufwand ver-
, da die Standard-Lese- und Schreibprozeduren nicht für den Datentyp
gic bzw. std_logic_vector gelten. Im Folgenden werden erweiterte Ein- und
beprozeduren [26] vorgestellt, die mit Hilfe von Overloading den Datentyp
gic berücksichtigen. Man kann das Package "text_io_pack.vhd" anwenden, oh-
Einzelheiten der VHDL-Beschreibung zu kennen. Es kann in ähnlicher Weise
werden wie ein Package mit einer VHDL-Deklaration, das eine Addition von
en (Kap. 6.2.2.2) erlaubt. Das unten beschriebene Package "text_io_pack .vhd"
r alle Aufgaben, die mit den Datentypen "std_logic" und "std_logic_vector" in
d Ausgabedateien arbeiten, verwendet werden.

genden Beispiel wird eine Testbench vorgestellt, die für die gleiche Aufgaben-
g mit der Wahrheitstabelle (Kap. 4.9.2) eingesetzt wird. Es wird vorausgesetzt,
ch das VHDL-Modell tab2_11a als compilierte Komponente in dem Package
abelle_pack.vhd befindet (s. Kap. 4.9.2).

```
o_pack.vhd
eee;
.std_logic_1164.all;
textio.all;
```

```
rocedure read (L: inout Line; wert: out std_logic_vector; gut: out boolean);
rocedure read (L: inout Line; wert: out std_logic_vector);
rocedure write (L: inout Line; wert: in std_logic; justified:in side := right; field: in
idth := 0);
rocedure write (L: inout Line; wert: in std_logic_vector; justified:in side := right;  field: in
idth := 0);
ype std_logic_chars is array (character) of std_logic;
onstant to_stdlogic: std_logic_chars:=
U' => 'U','X' => 'X','0' => '0','1' => '1','Z' => 'Z',
V' => 'W','L' => 'L','H' => 'H','-' => '-',others => 'X');
ype character_chars is array (std_logic) of character;
onstant to_character: character_chars :=
U' => 'U','X' => 'X','0' => '0','1' => '1','Z' => 'Z',
V' => 'W','L' => 'L','H' => 'H','-' => '-');
ext_io_pack;

ge body text_io_pack is
rocedure read (L: inout Line; wert: out std_logic; gut: out boolean) is
ariable temp: character;
      variable gut_character: boolean;
egin
      read (L, temp, gut_character);
      if gut_character = true then
                  gut := true;
                  wert := to_stdlogic(temp);
      else        gut := false;
      end if;
nd read;
rocedure read (L: inout Line; wert: out std_logic) is
      variable temp: character;
      variable gut_character: boolean;
egin
      read (L, temp, gut_character);
      if gut_character = true then  wert := to_stdlogic (temp);
      end if;
nd read;
rocedure read (L: inout Line; wert: out std_logic_vector; gut: out boolean) is
      variable temp: string(wert'range);
      variable gut_string: boolean;
egin
      read (L, temp, gut_string);
      if gut_string = true then
                  gut := true;
                  for i in temp'range loop
                    wert(i) := to_stdlogic(temp(i));
                  end loop;
      else gut := false;
      end if;
nd read;
```

```
in
   read (L, temp, gut_string);
   if gut_string = true then
                for i in temp'range loop
                  wert(i) := to_stdlogic(temp(i));
                end loop;
   end if;
 read;
cedure write (L: inout Line; wert: in std_logic; justified:in side := right; field: in
th := 0) is
   variable write_wert: character;
in
   write_wert := to_character(wert);
   write(L,write_wert, justified, field);
 write;

cedure write (L: inout Line; wert: in std_logic_vector; justified:in side := right;
   field: in width := 0) is
   variable write_wert: string(wert'range);
in
   for i in wert'range loop
     write_wert(i) := to_character(wert(i));
    end loop;
   write(L,write_wert, justified, field);
 write;
_io_pack;

ench mit Ein- und Ausgabedatei:  tb_textio_tab.vhd
eee;
.std_logic_1164.all;
textio.all;
k.text_io_pack.all;
k.tabelle_pack.all;

_textio is
extio;

ture auto_test of tb_textio is
: std_logic_vector(1 to 4);
: std_logic_vector(1 to 2);

2_11a port map (x => x, y => y);
process
eingabe_datei: text is in "tab_io_test.txt";            -- Eingabedatei: tab_io_test.txt
ausgabe_datei: text is out "ergebnis_aus.txt";      -- Ausgabedatei: ergebnis_aus.txt
able zeile_ein, zeile_aus: line;
able aus_y: std_logic_vector(1 to 2);
able v_x: std_logic_vector(1 to 4);
able v_y: std_logic_vector(1 to 2);
```

```
onstant abstand_2: string(1 to 2) := "  ";
onstant abstand_3: string(1 to 3) := "   ";
onstant ueber: string(1 to 21) := "  X   Y  Ysoll Fehler"; -- Ueberschrift Ausgabedatei

rite(zeile_aus, ueber);   -- Ueberschrift
riteline(ausgabe_datei, zeile_aus); -- Ausgabe der Zeile
riteline(ausgabe_datei, zeile_aus); -- Ausgabe einer Leerzeile

_loop: while not endfile(eingabe_datei) loop
eadline (eingabe_datei,zeile_ein); -- Zeile einlesen
ead (zeile_ein,char,gut);
berspringe Zeile, falls Zeichen kein Tabulator s. Tab. 4.10
 not gut or char /= HT then next;
nd if;
ssert gut
      report "Fehler beim Lesen"
everity note;
ead (zeile_ein,v_x,gut);
ext when not gut;
ead (zeile_ein,char);
ead (zeile_ein,v_y);
ait for 20 ns;
 <= v_x;        -- Typ-Konvertierung: variable → signal
ait for 30 ns;
berpruefen des Ergebnisses
us_y := y;
 aus_y /= v_y then
      assert false
      report "y falsch";
      fehler := true;
      fehler_aus := " ja ";
nd if;
rmatierter Ausgabereport
rite(zeile_aus, v_x);                -- Stimuli x
rite(zeile_aus, abstand_2);          -- 2 Leerzeichen
rite(zeile_aus, aus_y);              -- Ergebniswert y
rite(zeile_aus, abstand_2);          -- 2 Leerzeichen
rite(zeile_aus, v_y);                -- Erwartungswert für y
rite(zeile_aus, abstand_3);          -- 3 Leerzeichen
rite(zeile_aus, fehler_aus);         -- Fehler: ja oder nein
riteline(ausgabe_datei, zeile_aus);
fehler_aus := "nein";                -- Defaultwert für Fehler
oop zeile_loop;
 not fehler
eport "Test ist fehlerhaft"
everity note;
ssert fehler
eport "Test ist o.K"
everity note;
```

10: Ein- und Ausgabedatei, die innerhalb der Testbench verwendet werden

e_datei: tab_io_test.txt	ausgabe_datei: ergebnis_aus.txt
heitstabelle 2.11	X Y Ysoll Fehler
0000 11	0000 11 11 nein
0001 11	0001 11 11 nein
0010 11	0010 11 11 nein
0011 0-	0011 0- 0- nein
0100 00	0100 00 00 nein
0101 00	0101 00 00 nein
0110 00	0110 00 00 nein
0111 00	0111 00 00 nein
1000 0-	1000 0- 0- nein
1001 0-	1001 0- 0- nein
1010 11	1010 11 11 nein
1011 11	1011 01 11 ja
1100 0-	1100 0- 0- nein
1101 00	1101 00 00 nein
1110 0-	1110 0- 0- nein
1111 00	1111 00 00 nein

b. 4.10 enthält auf der linken Seite die Eingabedatei und auf der rechten Seite sgabedatei mit der Fehlermeldung. Das erste Zeichen einer relevanten Eingazeile ist der Tabulator. Dadurch können in der Eingabedatei auch Überschrifl Kommentarzeilen verwendet werden, die beim Einlesen der Werte über die eprozedur ignoriert werden.

tur zu Kap. 4: [9, 21, 65, 70, 78, 97, 120, 127]

RAM

den vorangehenden Kapiteln dargestellt wurde, sind heute leistungsfähige iterspeicher hoher Kapazität mit geringen Zugriffszeiten auf DRAM-Basis am verfügbar. Auch im Bereich der nichtflüchtigen Speicher gibt es ein weitgefä- Angebot, wie EPROMS, EEPROMs und Flash-EPROMs. Beide Speichertyfriedigen jedoch nicht Ansprüche, wie sie zunehmend im Bereich mobiler Angen gestellt werden. Insbesondere hier besteht Bedarf an Halbleiterspeichern, nstige Eigenschaften beider Gruppen in sich vereinen.

ca. 20 Jahren wird ein Speichermedium erforscht, das diese Forderung erfülnte, nämlich der ferroelektrische Speicher. Die Bezeichnung FRAM wurde r Fa. Ramtron geschützt. Andere Hersteller nennen diesen Speichertyp daher M. Nach Einschätzung von Fachleuten werden ferroelektrische und magnetorepeicher (MRAMs, s.Kap. 7.1.15) die Speichertechnologie in nächster Zukunft eren. Ein qualitativer Vergleich wesentlicher Eigenschaften von Halbleiterrn macht dieses deutlich (Tab. 7.2). Es werden FRAMs angestrebt, die als Ulspeicher bisherige Speichertypen ersetzen sollen (All-in-One-Solution).

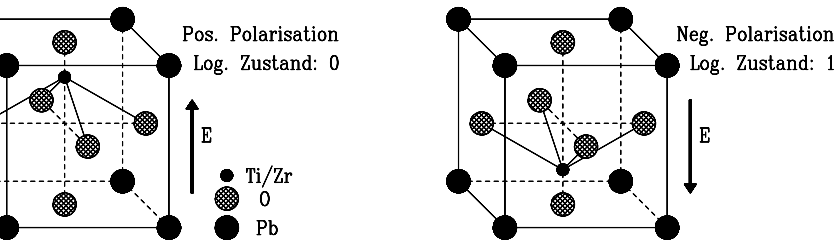
2: Wesentliche Eigenschaften unterschiedlicher Speichertechnologien

chaften	**SRAM**	**DRAM**	**EEPROM**	**FLASH**	**FRAM MRAM**
üchtig	nein	nein	ja	ja	ja
Zellenmaße	nein	ja	nein	ja	ja
eise les- und beschreibbar	ja	ja	ja	nein	ja
er Leistungsbedarf	ja	ja	nein	nein	ja
ler Schreibzugriff	ja	ja	nein	nein	ja
chreibzyklen	ja	ja	nein	nein	ja
günstig	nein	ja	nein	ja	ja

d an einen Kondensator eine elektrische Spannung angelegt, nimmt er eine La- auf, und zwischen seinen Belägen entsteht ein elektrisches Feld. Dieses führt r Ladungsverschiebung im Dielektrikum, die man als dielektrische Polarisatieichnet. Wird der Kondensator entladen, verschwindet auch die Polarisation.

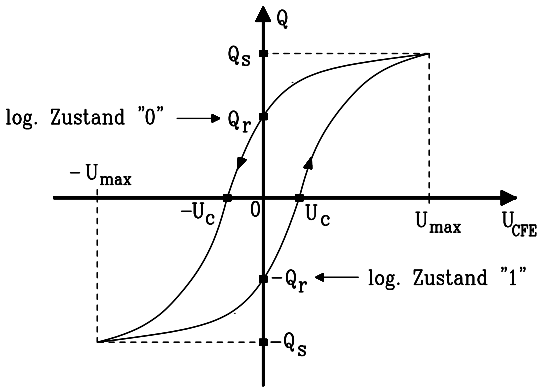
er dem *ferroelektrischen Effekt* versteht man die Eigenschaft einiger Isolierdurch Anlegen geeigneter elektrischer Felder spontan einen von zwei unterichen Polarisationszuständen einnehmen zu können, und diesen Zustand auch ntfernen der Spannung als *remanente Polarisation* beizubehalten. Diese Ei-

utzt man diese Stoffe als Dielektrikum in einem Kondensator, erhält man ein biles Element, das zum Speichern digitaler Information geeignet ist und *ferro-rischer Kondensator* (C_{FE}) heißt. Das Speicherprinzip ist in Bild 7.29 gezeigt. Im rum des Kristalls befindet sich in Abhängigkeit von der Legierung ein Titan- Zirkoniumatom. Die unterschiedliche bistabile Position dieses Atoms bestimmt olarisationsrichtung des Kristalls.



7.29: Ferroelektrische Perovskit-Kristalle (PZT), die durch elektrische Feldstärken E in unterschiedliche stabile Polarisationsrichtungen versetzt worden sind

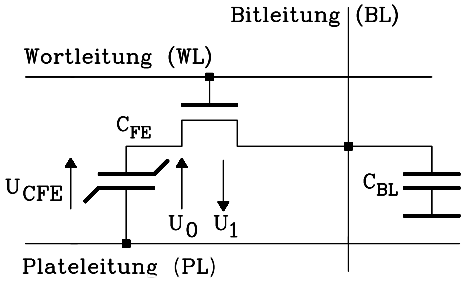
er funktionelle Zusammenhang zwischen der angelegten Spannung U und der enommenen Ladung Q zeigt eine Hysterese, deren Form der Magnetisierungs- e ferromagnetischer Stoffe vergleichbar ist, wie Bild 7.30 zeigt.



7.30: Zusammenhang zwischen Spannung U_{CFE} und aufgenommener Ladung Q bei einem ferroelektrischen Kondensator C_{Fe}. Die Abkürzungen bedeuten: Q_s:Sättigungsladg., Q_r: Remanentladg, U_{CFE}: Kondensator- und U_c: Koerzitivspannung

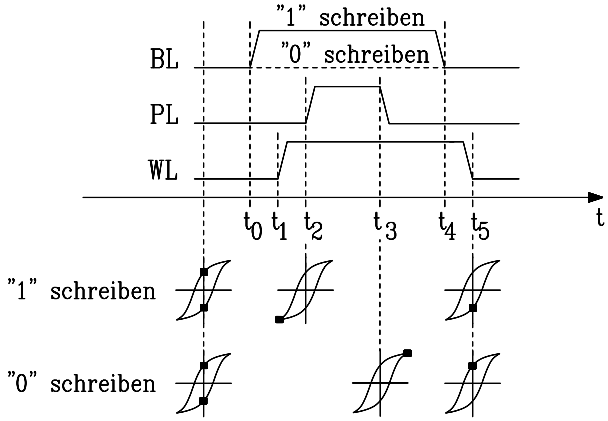
ird der Kondensator an eine Spannung $+U_{max}$ gelegt und dann die Spannungs- e entfernt, wird der stabile Arbeitspunkt Q_r angenommen. Damit ist die log. "0"

Organisation der einzelnen Speicherzellen in einem FRAM entspricht prinzi-
r bei DRAMs verwendeten Technik: Jede ferroelektrische Zelle wird über ei-
Kanal-Enhancement-Transistor angesteuert. Daraus entsteht die sog. 1T-1C-
1 Transistor, 1 Kapazität, Bild 7.31). Sie enthält als Speicherelement den ferro-
schen Kondensator C_{Fe} . Die Zelle wird angesprochen durch einen ausreichend
H-Pegel an der Wortleitung WL. Dadurch wird der Transistor leitfähig, und
n beschrieben oder gelesen werden. C_{BL} repräsentiert die parasitäre Kapazität
leitung.



31: Ferroelektrische 1T-1C-Speicherzelle, bestehend aus Zugangstransistor, Speicherkondensator C_{Fe}, und der parasitären Kapazität der Bitleitung C_{BL}

bvorgang: Das Liniendiagramm ist in Bild 7.32 gezeigt. Die Speicherzelle be-
ich für $t < t_0$ unselektiert in einem der beiden Zustände "0" oder "1".



32: Schreibvorgang bei einer FRAM-Speicherzelle. Wichtige Zustände des ferromagnetischen Kondensators sind an der Hystereseschleife verdeutlicht.
Die Abkürzungen bedeuten: BL: Bitleitung PL: Plateleitung WL: Wortleitung

$\mathbf{t_0}$: Die Bitleitung (BL) geht vorbereitend auf H-Pegel.
$\mathbf{t_1}$: Die Wortleitung (WL) geht auf H-Pegel. Der Transistor leitet, und C_{Fe} wird über den ON-Widerstand des Transistors auf H-Pegel geladen. Damit befindet sich die Zelle in der Hystereseschleife im Punkt $-Q_S$.
Zum Zeitpunkt $\mathbf{t_2}$ wird ein kurzer positiver Impuls auf die Plateleitung PL gegeben, der hier keine Bedeutung hat.
Nach dessen Ende bei $\mathbf{t_3}$ steht an C_{Fe} wieder H-Pegel, da WL und BL noch gesetzt sind. Damit gilt: $U_{CFE} = -U_1 = -U_{max}$.
Zum Zeitpunkt $\mathbf{t_4}$ geht BL auf "0" bei leitendem Transistor, d.h. $U_{CFE} = 0$ und C_{Fe} geht in den Zustand remanenter Polarisation Q_r, speichert also das Bit "1".
Bei $\mathbf{t_5}$ wird die Speicherzelle durch WL deselektiert und der Zustand "1" bleibt ohne weitere Energiezufuhr erhalten.

eiben einer "0": BL bleibt stets auf L-Pegel.

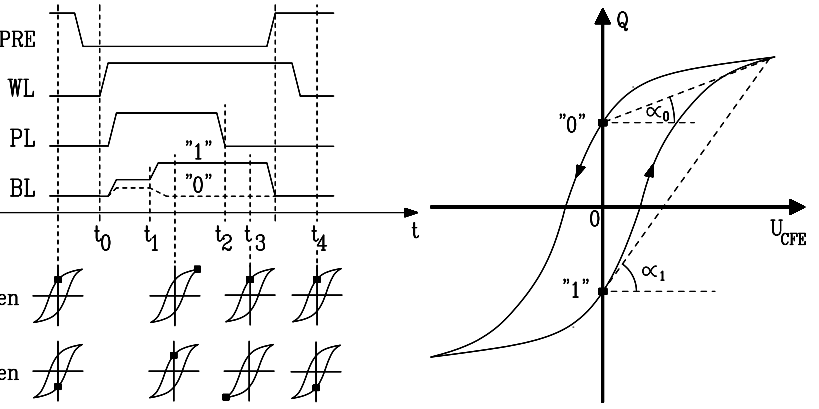
Bei $\mathbf{t_1}$ nimmt WL H-Pegel an, der Transistor leitet. Der Kondensator CFe liegt an der Spannung UCFE = 0V, d.h. sein logischer Zustand bleibt unverändert.
Im Intervall $\mathbf{t_2}$ bis $\mathbf{t_3}$ geht PL auf "1", dadurch wird UCFE = U0 = +Umax, d.h. CFe befindet sich auf der Hysteresekurve im 1. Quadranten bei QS .
Im Intervall $\mathbf{t_3}$ bis $\mathbf{t_4}$ beträgt UCFE = 0 und CFe befindet sich im Zustand remanenter Polarisation im Punkt +Qr, hat also das Bit "0" gespeichert.
Bei $\mathbf{t_5}$ wird die Speicherzelle durch WL deselektiert und der Zustand "0" bleibt ohne weitere Energiezufuhr erhalten.

vorgang: Der Lesevorgang an einer FRAM-Speicherzelle besteht prinzipiell dadass der ferromagnetische Kondensator C_{FE} bei hochohmig geschalteter Bitleieinen Teil seiner Ladung auf die parasitäre Kapazität C_{BL} überträgt. Die dabei portierte Ladung und daher auch die an C_{BL} entstehende Spannung hängen davon b vorher eine "0" oder eine "1" gespeichert war. Ein Sense-Verstärker führt diese vertung durch. Daraus folgt, dass es sich um ein zerstörendes Leseverfahren elt. Der ursprüngliche Wert muss anschließend wieder rückgespeichert werden. 7.33 zeigt einen Lesevorgang nach der Methode des *Step-Sensing-Approach.* er Vorgang hat sehr große Ähnlichkeit mit dem Leseverfahren an einem DRAM.

ie FRAM-Speicherzelle befinde sich für $t << \mathbf{t_0}$ unselektiert in einem der beiden ände "0" oder "1". Dann folgen:

$t < \mathbf{t_0}$: Eine Leseoperation beginnt mit einem Precharge-Vorgang, der zunächst die Bitleitung hochohmig schaltet und anschließend die parasitäre Kapazität CBL entlädt. Der Speicherinhalt verändert sich dabei nicht. Zum Zeitpunkt $\mathbf{t_0}$ ist dieses abgeschlossen.
Bei $\mathbf{t_0}$ geht die Wortleitung auf H-Pegel und schaltet den Transistor in den leitenden Zustand. Dadurch sind C_{FE} und CBL zwischen PL und Masse in Serie ge-

. Für die Kapazität gilt: $C_{FE} = \Delta U_{CFE}/\Delta Q$, und dieser Quotient wird vom Be-
ebspunkt auf der Hystereseschleife während des Ladeprozesses bestimmt (s.
ld 7.33 (rechts).



33: Lesevorgang bei einer FRAM-Speicherzelle mit Zuständen des ferromagnetischen Kondensators in der Hystereseschleife und Näherungen für die Kapazitäten von C_{FE} : tan $\alpha 1 = C_1$ und tan $\alpha 0 = C_0$. Die Abkürzungen bedeuten:
PRE: Prechargesignal BL: Bitleitung PL: Plateleitung WL: Wortleitung

pproximiert man die beiden Kapazitäten als Geraden, ergibt sich überschlägig für C_0 = tan $\alpha 0$ und für C_1 = tan $\alpha 1$ mit $C_1 > C_0$. An der Bitleitung liegt dann die Spannung U_{CBL} und es gilt $U_{CBL}(C_{FE} = C_1) > U_{CBL}(C_{FE} = C_0)$. Diese Verhältnisse sind an der Stelle $t = \mathbf{t_1}$ Bild 7.33 erkennbar.

$$_{CBL} = U_{CC} \cdot \frac{C_{FE}}{C_{FE} + C_{BL}}$$

um Zeitpunkt $t = \mathbf{t_1}$ wertet der Sense-Verstärker die Spannung U_{CBL} aus und kennt den Wert des gelesenen Bits. War es "1", legt er die Spannung U_{CC} an e Bitleitung, ansonsten 0V.

a WL weiterhin auf H-Pegel liegt und zum Zeitpunkt $t = \mathbf{t_3}$ L-Pegel an PL gt, wird der gelesene Bitwert in den Speicher zurückgeschrieben.

er Lesezyklus endet bei $\mathbf{t_4}$ durch WL = L-Pegel und der gelesene Zustand eibt ohne weitere Energiezufuhr in der Speicherzelle erhalten.

ximale Datenhaltung (data retention) für FRAMs wird heute mit 10 Jahren be- Gründe für die Begrenzung sind im Wesentlichen:

it zunehmendem Alter tritt eine Depolarisation auf. Sie äußert sich dadurch, ss der remanente Ladungsbetrag $|Q|$ für beide Betriebspunkte abnimmt (Fa-

Wenn FRAM-Zellen überwiegend einen festen log. Zustand speichern, passt sich die Hystereseschleife diesem Zustand an, indem der remanente Ladungsbetrag $|Q_r|$ des gegenüberliegenden Arbeitspunktes abnimmt (Imprint).

h Weiterentwicklung der Schaltungskonzepte für Speicherzellen und Sensetärker wird versucht, die Spannungsunterschiede zwischen den Lesesignalen für nd "1" zu vergrößern, um die Wahrscheinlichkeit von Lesefehlern zu reduzieren. oben beschriebene 1T-1C-Zelle wird sich künftig voraussichtlich gegenüber der ängerer Zeit auf dem Markt befindlichen 2T-2C-Zelle durchsetzen, da sie weniPlatz benötigt und Kosten spart.

esentliche Betriebsdaten von FRAMs sind in der Tab. 7.3 denen anderer Speikonzepte gegenübergestellt.

7.3: Gegenüberstellung wesentlicher Daten unterschiedlicher Speichertypen

nschaften	SRAM	DRAM	NAND-FLASH	NOR-FLASH	MRAM (FET)	FRAM
engröße/µm²	100	8	1,3	2,5	>8	4-20
ebsspannung /V	2,5	2,5	1,8	3,3	1,8-5	3-5
nhaltung/a	<<1 mit Batterie	flüchtig	10	10	10	10
zeit/ns random	2	60	10^4	60-90	10-50	60
eibzeit/ns random am/Erase Speed	2	60	2,1/5,3 MB/s	0,2/0,08MB/s	10-40	60
zyklenzahl	$> 10^{15}$	$> 10^{15}$	$> 10^{15}$	$> 10^{15}$	$> 10^{15}$	10^{12}-10^{15}
eibzyklenzahl	$> 10^{15}$	$> 10^{15}$	10^5	10^5	$> 10^{15}$	10^{10}-10^{15}

15
MRAM

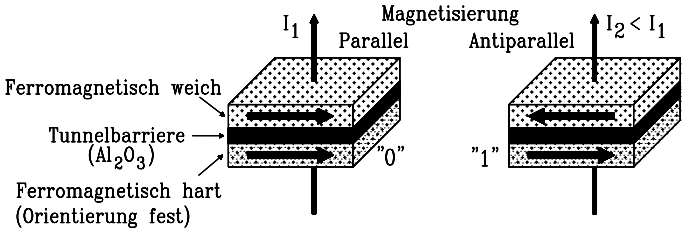
Kapitel 7.1.14 wird mit dem FRAM ein neuer Speichertyp vorgestellt, der ähnli-Eigenschaften wie ein DRAM besitzt, aber nichtflüchtig ist und daher neue Mögteiten im Bereich mobiler Anwendungen und als Arbeitsspeicher in der Compuchnik bietet.

uf diese Anwendungsbereiche zielt auch ein anderer nichtflüchtiger Speichertyp, *Magnetoresistive RAM* (MRAM). Es hat gegenüber dem FRAM einen weiteren eil, denn es lässt sich zerstörungsfrei lesen. Damit entfällt das Rückspeichern der senen Information, welches Zeit und Energie benötigt. Insgesamt hält damit in Halbleiterindustrie eine neue Entwicklung Einzug, die *Magnetoelektronik*, welche oderne Halbleiterprozesstechnologie mit der Technologie ferromagnetischer

weiterer, mit GMR verwandter magnetoelektrischer Effekt beruht darauf, dass r etwa vier Atomlagen dünne dielektrische Schicht (Al_2O_3) zwischen zwei fernetischen Metallbelägen einen Tunnelstrom führen kann, der sich durch die erung der magnetischen Diplomomente in den Metallbelägen verändern lässt. erartige Zelle heißt *Magnetische Tunnelbarriere* (Magnetic Tunnel Junction, und wird vorwiegend als Speicherzelle verwendet.

Entwicklungsstand von Speicherbauelementen, die dieses Prinzip nutzen, twa drei Jahre hinter dem von FRAMs her. Aufgrund einer weltweit vollzogenzentration der Entwicklungsarbeiten sind aber seit 2007 marktreife Produkte par.

und Funktion einer MTJ-Speicherzelle: Sie besteht aus einem Stapel zweier agnetischer Schichten, die durch ein dünnes Dielektrikum voneinander gesind. Wird an diese Zelle eine Spannung gelegt, fließt ein Tunnelstrom, dessen davon abhängt, ob die Orientierung des magnetischen Feldes in den ferromagen Schichten parallel oder antiparallel ist (Bild 7.34).

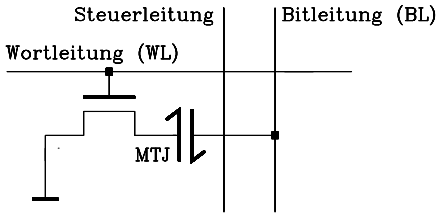


34: Funktionsweise eines MTJ-Speicherelements (Magnetic Tunnel Junction). Die Richtung des magnetischen Feldes in der ferromagnetisch weichen Schicht verändert den Tunnelstrom. Daher besitzt das Element zwei stabile Zustände, denen die log. Zustände "0" bzw. "1" zugeordnet sind.

ursacht wird dieses Verhalten durch eine Spin-Polarisation der Leitungselekin den ebenfalls nur wenige Atomlagen dicken ferromagnetischen Elektroden, der Orientierung des Magnetfeldes bestimmt wird. Sind etwa Leitungselekach dem Passieren der ersten Elektrode in einer Richtung 1 polarisiert, wird chgang durch die zweite Elektrode behindert, wenn diese für die andere Polasrichtung 2 eingestellt ist. Daher bewirkt die parallele magnetische Feldorien- in beiden Leitern gegenüber antiparalleler einen um bis zu 50% geringeren stand. Diesen Effekt nennt man *Magnetoresistanz*.

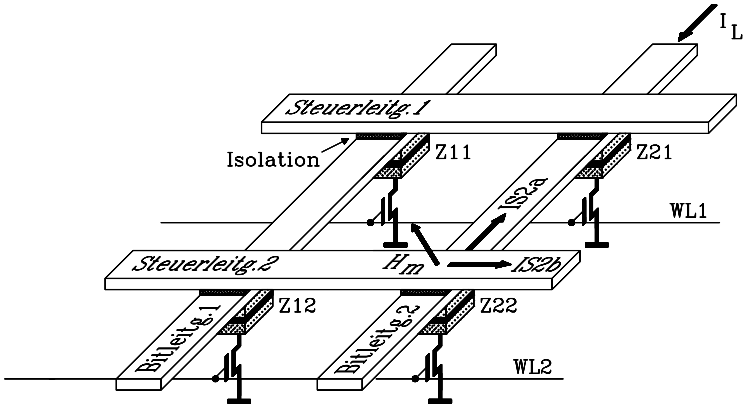
Speicherzelle hat also infolge der Remanenz im magetisch weichen Leiter

ie Einordnung in ein Speicherarray ist beim MRAM prinzipiell möglich, indem eine Elektrode der Zelle mit der Bitleitung und die andere mit der Wortleitung t verbindet (Kreuzpunkt-Zelle). Zur Reduzierung parasitärer Leckströme müsie Zellen dann aber sehr hochohmig gefertigt werden. Dadurch sinkt wegen der sitären Kapazitäten des Systems die Geschwindigkeit um etwa drei Zehnerpoten-Sind kleine Zugriffszeiten nötig, schließt man jede Zelle über einen Transistor an Array an und erhält damit eine 1T-1MTJ-Zelle (Bild 7.35).



7.35: Magnetoresistive 1T-1MTJ-Speicherzelle. MTJ bedeutet: Magnetic Tunnel Junction. Die Steuerleitung ist für Schreiboperationen erforderlich.

ur Erläuterung der Funktionsweise eines MTJ-Speichers ist in Bild 7.36 ein verchter Ausschnitt aus der räumlichen Anordnung eines Speicher-Arrays darge-. Es besteht aus zwei Wort- und zwei Bitleitungen und enthält die vier Speichern Z11, Z12, Z21 und Z22 mit ihren Zugangstransistoren. Zwei zusätzliche Steutungen sind für den Schreibvorgang nötig. Bit- und Steuerleitungen sind voneinr isoliert. Die Zugangstransistoren sind nur bei Leseoperationen durchgeschaltet.



7.36: Ausschnitt aus einem 1T-1MTJ-Array mit 4 Speicherzellen. Die Abkürzungen bedeuten: MTJ: Magnetic Tunnel Junction WLi: Wortleitungen Zii: MTJ-Zellen

22 durchtunnelt. Die Spannung an der Bitleitung hängt nun vom log. Zustand, r Leitfähigkeit des MTJ-Elements ab, sie wird durch einen Sense-Verstärker vertet und liefert das gespeicherte Bit. Das MTJ-Element ändert während des rgangs seinen Zustand nicht, daher handelt es sich um einen nicht-zerstö- Leseprozess, ein besonderer Vorteil dieses Speichertyps.

bvorgang: Alle Zugangstransistoren in den Wortleitungen sind gesperrt. Es die Speicherzelle Z22 beschrieben. Dazu fließen Ströme in den beiden isoliert ander angeordneten Leitungen: In Bitleitung 2 fließt der Strom IS2a und in der eitung der Strom IS2b. Beide Ströme verursachen Magnetfelder, die sich an eren, ferromagnetisch weichen Elektrode der MTJ-Zelle vektoriell zur Feld- H_m addieren. Die Ströme sind so bemessen, dass H_m die Elektrode in magneti- ttigung bringt. Nach Wegnahme der Ströme verbleibt die Elektrode im Rema- nkt. Die Zelle enthält damit z. B. den log. Zustand "0" und dieser ist ohne Zu- n Energie stabil, es handelt sich also um einen nichtflüchtigen Speicher. Zum ben einer "1" ist die Richtung des Stroms IS2a der Bitleitung umzukehren.

Kap. 7.1.14 ist in Tab. 7.3 ein Vergleich wesentlicher Eigenschaften unter- icher Speicherkonzepte dargestellt. Daraus geht hervor, dass *Magnetoresistive* (MRAMs) gegenüber *Ferromagnetischen RAMs* (FRAMs) Vorteile bezüglich öheren Schreib-/Lesezyklenzahl und geringerer Zugriffszeiten haben.

wird damit gerechnet, dass die neuen nichtflüchtigen Speicher MRAMs und s die künftige Speicherlandschaft revolutionieren. Beispielsweise wird heute zepten gearbeitet, diese Speichertypen in Computern künftig nicht nur als Ar- sondern auch als Massenspeicher einzusetzen. Damit würden der Bootprozess ssig und Massenspeicherzugriffe erheblich beschleunigt.

wertspeicher (ROM)

fbau eines Festwertspeichers (Nur-Lese-Speicher, ROM) entspricht hinsicht- r Matrixanordnung seiner Speicherzellen und der Adressverwaltung prinzipiell nigen eines RAMs (Bild 7.1). Allerdings fehlen die Eingänge D_{in} und ¬WE, da stwertspeicher während des Betriebes ausschließlich gelesen werden kann. ndet werden ROMs zum Abspeichern unveränderlicher Daten, wie etwa Ma- nprogrammen in der Mikroprozessortechnik.

enprogrammiertes ROM

7.37 ist der prinzipielle Aufbau eines maskenprogrammierten ROMs darge-