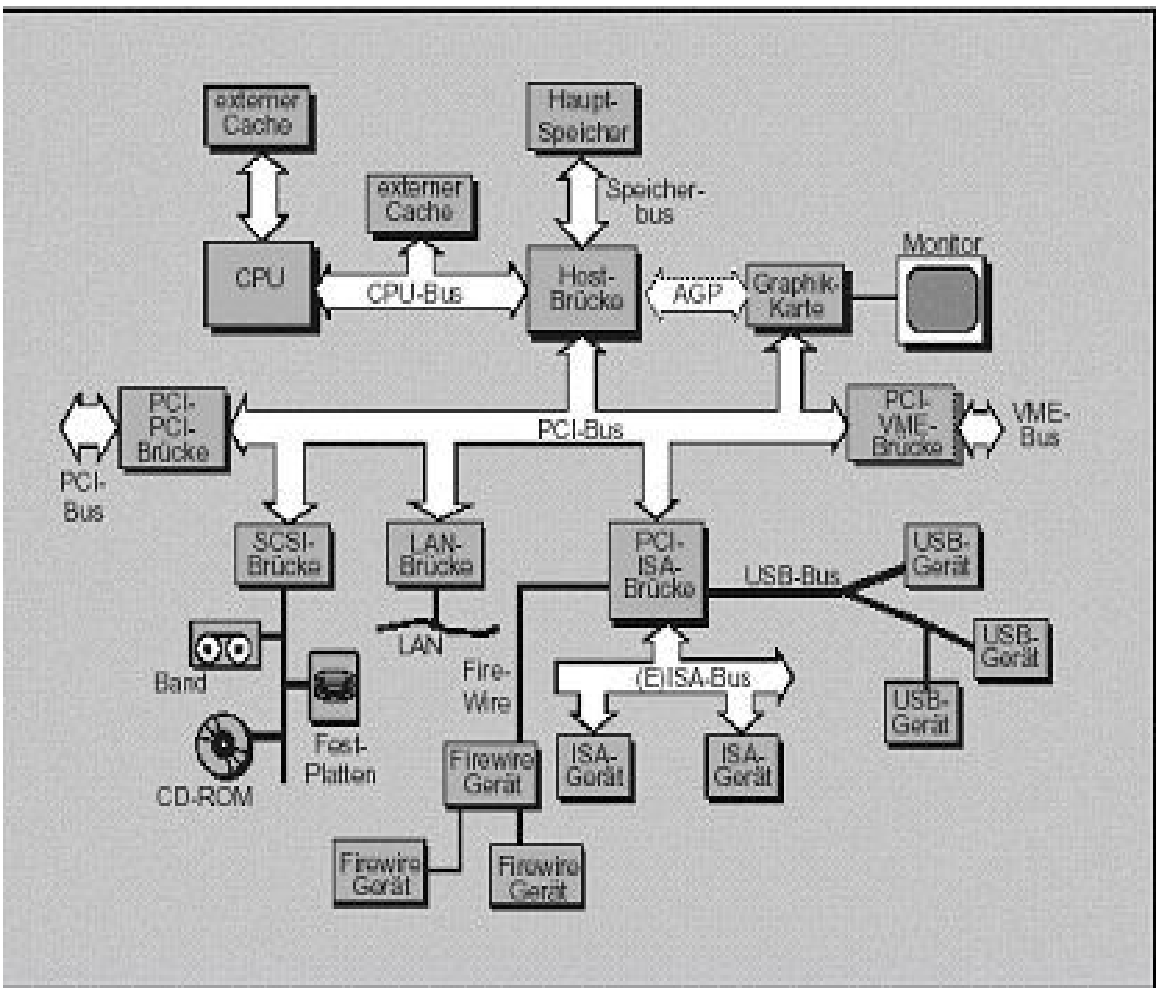
Kurs 1708

Technische Informatik II

Klausur- und Prüfungsvorbereitungsunterlagen
Kurseinheiten 1 – 7

Autor: Annerose Heim



Kurseinheit 1:

Systembusschnittstelle

Besteht aus mindestens folgenden Registern;
- Datenbuspuffer
- Adressbuspuffer
- Lese/Schreib-FIFO

Funktion:

Datenbuspuffer:
Speichert alle Daten, die von der Außenwelt in den Prozessor gelesen oder umgekehrt den Prozessor verlassen
Bei Prozessoren mit höherer Verarbeitungsgeschwindigkeit hält er auszugebende Daten solange fest, bis diese vom Speicher übernommen werden

Adreßbuspuffer
Enthält stets die Adresse der Speicherzelle, auf die im nächsten Buszyklus zugegriffen werden soll

Lese/Schreib-FIFO:
Warteschlange mit wenigstens 4 Speicherplätzen

Zeitverhalten der Systembus-Signale:

- **Synchroner Systembus**

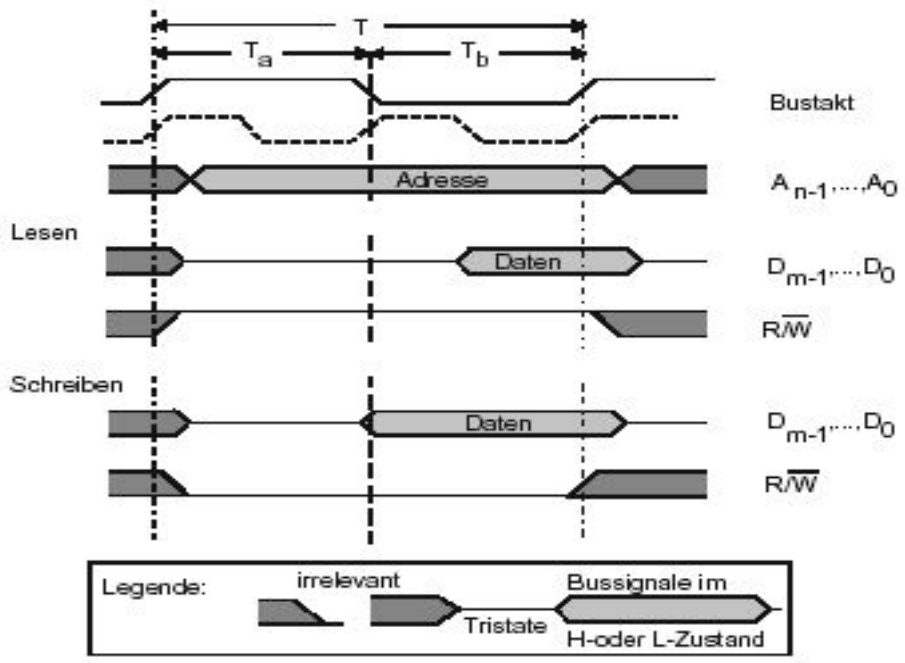


Bild 0.2-1: Zeitliche Abläufe auf einem synchronen Systembus

- beim synchronen Systembus laufen alle Vorgänge synchron zum Takt in einem starren Muster ohne Benutzung weiterer Steuersignale ab
- Übergabe der Daten geschieht zu festgelegten Taktflanken

- Nachteil: alle am Bus angeschlossenen Komponenten müssen dieselben strengen Zeitvorgaben erfüllen

- **Semi-synchroner Systembus**

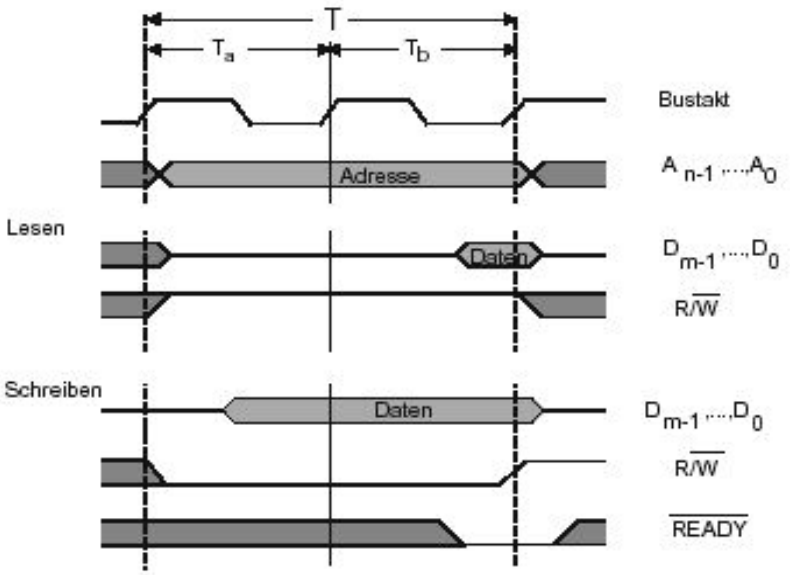


Bild 0.2-2: Zeitliche Abläufe auf einem (semi-)synchronen Systembus

- im Gegensatz zum reinen synchronen Systembus werden hier weitere Steuer- und Meldesignale zur Synchronisation verwendet
- dadurch werden die Nachteile des synchronen Systembusses vermieden

- **Asynchroner Systembus**
  - Handshake-Verfahren:
  - Austausch AS ! und ACK!-Signal
  - Es können beliebige Wartezyklen eingebaut werden

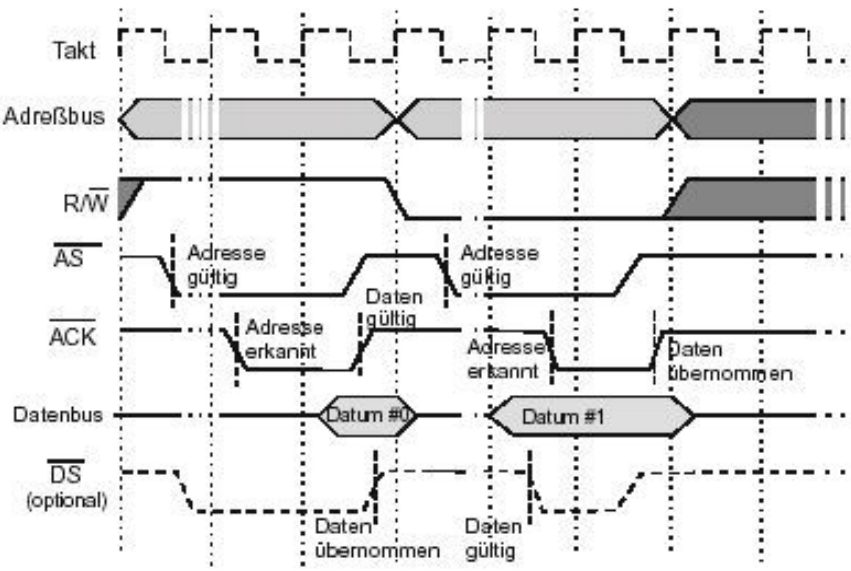


Bild 0.2-5: Zeitliche Abläufe auf einem asynchronen Systembus

Vergleich zwischen synchronen und asynchronen Bussen

- bei semi-synchronen und asynchronen Bussen können auch Komponenten mit unterschiedlichen Geschwindigkeitsanforderungen angeschlossen werden
- semi-synchrone Busse benötigen höheren Steueraufwand als asynchrone
- bei asynchronen Systembussen spielt Systemtakt fast keine Rolle für die Synchronisation der Übertragung

Systembusse mit überlappender Adressierung

- mit gesplittetem Adreß- und Datenzugriff
- mit Blockübertragungen

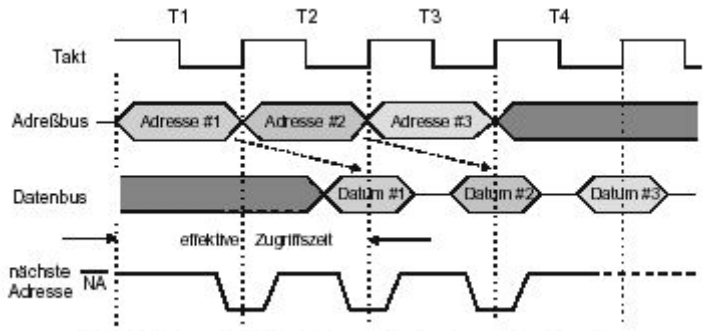


Bild 0.2-6: Protokoll eines Busses mit überlappender Adressierung

Adressierung von Peripheriebausteinen

- Speicherbezogene Adressierung
- Isolierte Adressierung

Multiplexbus

- wurde eingeführt, um die Anzahl der Anschlußstifte am Prozessorgehäuse und die Signalleitungen zu begrenzen
- multiplexen findet statt zwischen
  - Adressen und Daten
  - Höher- und niederwertigen Adreßbits
  - In gemischter Form zwischen niederwertigen Adreßbits und Daten

Steuerung der Datenbusbreite

- 32 bit Prozessor kann mit Hilfe von jeweils 4 Bytes großen Datenpaketen über einen 8 bit-Schnittstellenbaustein Daten ausgeben
- Einsatz eines 8-Bit Festwertspeichers in einem 32-bit-System, der eine Initialisierungsroutine enthält, die nur nach dem Einschalten des Systems ausgeführt wird

Weitere Funktionen der Systembusschnittstelle

- **Erzeugen der folgenden Steuer- und Meldesignale**
  - Speichersteuersignal
    - RAS (Row Address Strobe)
    - CAS (Column Address Strobe)
  - Unterbrechungssignale
    - HALT
    - RESET
    - ERROR
    - Schalten in Sparstrommodus (Power-Down)
  - Test & Debug-Signale
    - Test der Ein-/Ausgangsschaltungen der Bausteine und Verbindungsleitungen durch JTAG (Join Test Action Group)
  - Signale zur Modusauswahl, Identifikation und Benutzer-Ein-/Ausgabesignale
- **Busarbitrierungssignale**
  - BR (Bus Request)
  - BG (Bus Grant)
  - BGACK (Bus Grant Acknowledge)

Programmierbarer SystembusController

Besteht aus zwei Hauptkomponenten:

Chip Select Logik
Bussteuerung

Chip Select Logik

- für jeden Speicherbereich ein Register
- enthält Basisadresse des Speicherbereichs und Länge
- Größe: wenige KB bis 1 MB

Bussteuerung

- für jeden Speicherbereich ein Steuerregister
- Informationen über Art und Arbeitsweise des Systembusses
- Typische Bitfelder im Steuerregister:
  - BW (Bus Width) => Datenbusbreite
  - MUX => Auswahl ob Multiplex oder nicht
  - WS (Wait States) => Anzahl Wartezyklen
  - R/W (Read/Write) => Einsatz von Bausteinen verschiedener Hersteller

Hardware- / Software-Schnittstelle

Datentyp: spezifiziert den Wertebereich, den eine Konstante, Variable, Ausdruck oder Funktion annehmen kann

Datenformat: bestimmt den Datentyp, Anzahl der Bits und seine Bedeutung

Befehlssatz: Menge der Maschinenbefehle

Adressierungsarten: Möglichkeiten, aus Adressen , Adressdistanzen und Register-inhalten Operandenadressen zu berechnen

Datentypen und –formate 8-Bit-Prozessoren

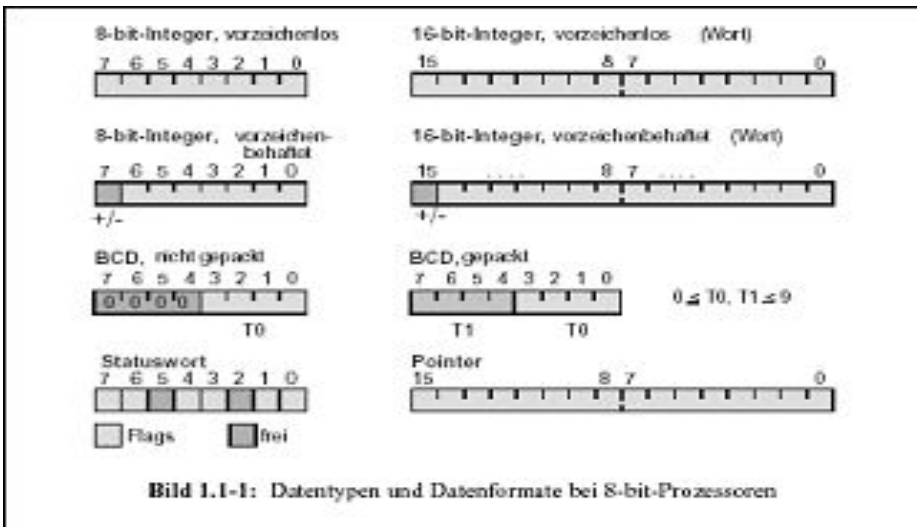


Bild 1.1-1: Datentypen und Datenformate bei 8-bit-Prozessoren

Tabelle 1.1-1: Bit-Wertigkeiten und Zahlenbereiche der Integer-Zahlen

Format	vorzeichenlos		vorzeichenbehaftet	
	Bit-Wertigkeiten	Zahlenbereich	Bit-Wertigkeiten	Zahlenbereich
8 bit (Byte)	$2^7,\ldots,2^0$	$0 \le Z \le 255$	$-2^7, 2^6,\ldots,2^0$	$-128 \le Z \le 127$
16 bit (Wort)	$2^{15},\ldots,2^0$	$0 \le Z \le 65535$	$-2^{15}, 2^{14}\ldots,2^0$	$-32768 \le Z \le 32767$

BCD-Zahl

Binary Coded Decimal

z.B. 1357 = 0001 0011 0101 0111
1 3 5 7

Vorsicht: BCD-Zahl ist keine HEX-Zahl !!!!!

Pointer

Adresse und somit vorzeichenlose ganze Zahl, die auf einen Operanden im Speicher zeigt

Statuswort

Inhalt des Statusregisters
Verkettung von einzelnen Bits

Boolesche Daten

Logisch falsch = 0
Logisch wahre = 1

Rechnen mit Zahlen

Addition mit Wrap around

	DW1	DW2
R1:	6AE303D2	835A7F34
R2:	12F0A234	AF061707
	1 1	1 1 1
	7DD3A606	13260963B

wird abgeschnitten

ð Ergebnis: 7DD3A606 3260963B

Addition vorzeichenlos mit Sättigung

	B7	B6	B5	B4	B3	B2	B1	B0
R1:	6A	E3	03	D2	83	5A	7F	34
R2:	12	F0	A2	34	AF	06	17	07
					1	1	1	
	7C	1D3	A5	106	132	60	96	3B
		ß		ß	ß			
		FF		FF	FF			

ð Ergebnis: 7C FF A5 FF FF 60 96 3B

Addition vorzeichenbehaftet mit Sättigung

	W3	W2	W1	W0
R1:	6AE3	03D2	835A	7F34
R2:	12F0	A234	AF06	1707
	1	1	1 1	1
	7DD3	A606	13260	963B
	negativ		ß	ß positiv
			8000	7FFF

ð Ergebnis: 7DD3 A606 8000 7FFF

Datentypen bei 16/32-bit-Prozessoren

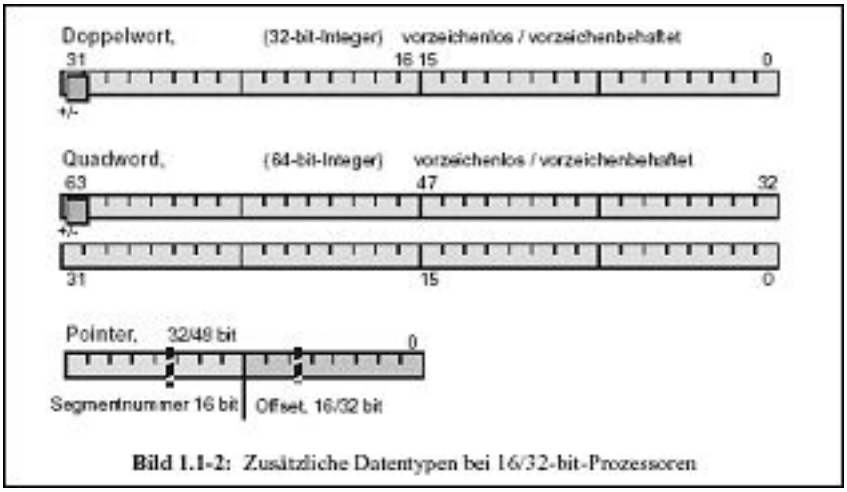


Bild 1.1-2: Zusätzliche Datentypen bei 16/32-bit-Prozessoren

Gleitpunktzahlen IEEE-754 Standard

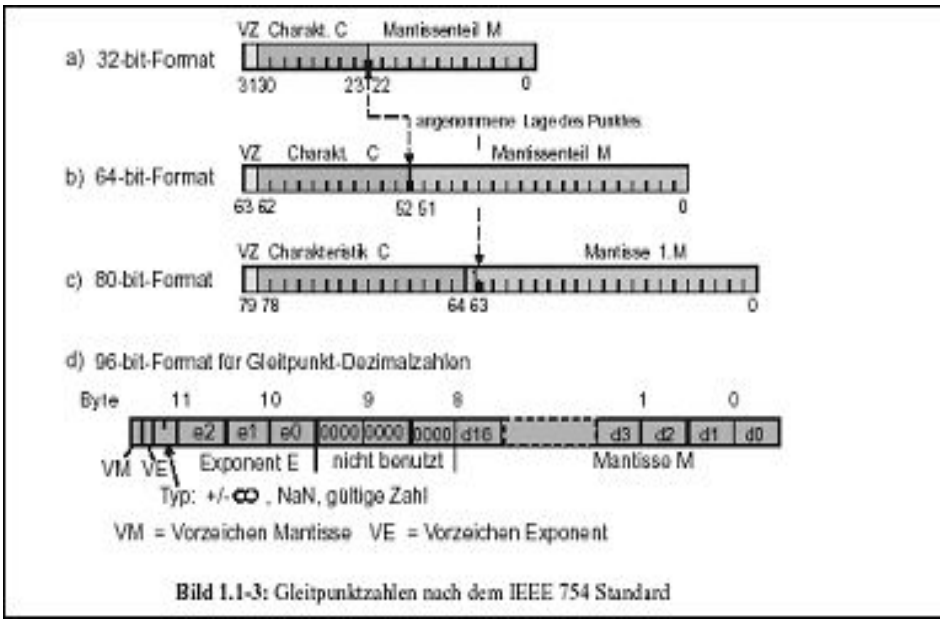


Bild 1.1-3: Gleitpunktzahlen nach dem IEEE 754 Standard

16-Bit Festpunktrechenwerke

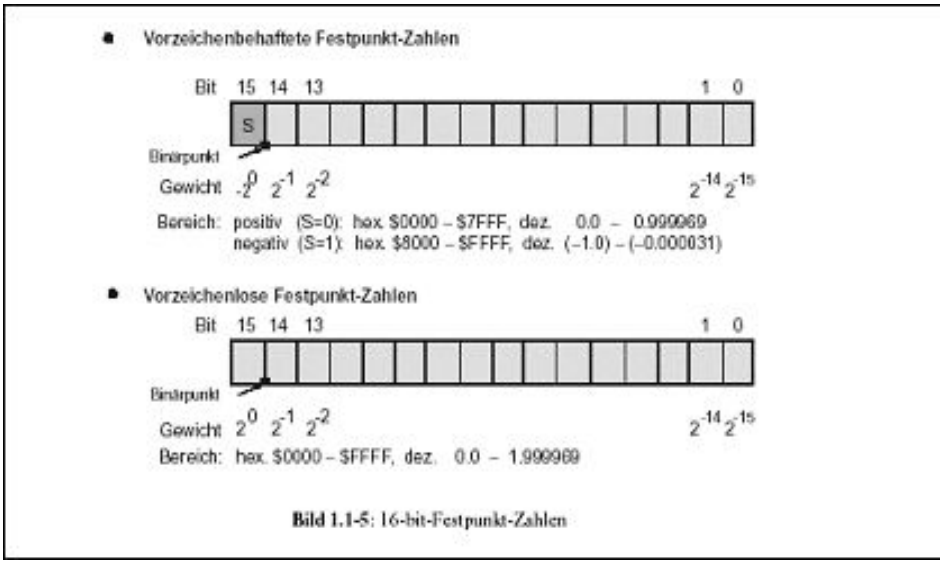


Bild 1.1-5: 16-bit-Festpunkt-Zahlen

Gewicht:

$-2^0 = -1$
$2^{-1} = ½$
$2^{-2} = 1/4$
$2^{-3} = 1/8$
:
:

Rechenbeispiele:

$7000 => 0.111 0000 0000 0000
ð + ½ + ¼ + 1/8 = 0,5 + 0,25 + 0,125 = 0,875 *10*

$5B00 => 0.101 1011 0000 0000
ð ½ + 1/8 + 1/16 + 1/64 + 1/128
= 64/128 + 16/128 + 8/128 + 2/128 + 1/128
= 91/128 = 0.7109375 *10*

$9000 => 1.001 0000 0000 0000
ð -1 + 1/8 = -1 + 0,125 = -0,875 *10*

Multimedia-Erweiterungen

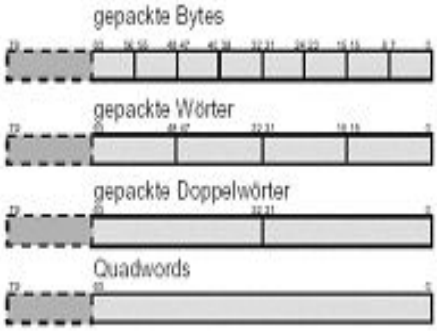


Bild 1.1-6: MMX-Datentypen

Im einzelnen werden die folgenden Möglichkeiten unterschieden, Daten zu einer Datenstruktur zusammen zu fassen und parallel durch jeweils denselben Befehl verarbeiten zu lassen:

- gepackte Bytes (*packed Bytes – B*): acht Bytes B7,..,B0,
- gepackte Wörter (*packed Words – W*): vier 16-bit-Wörter W3,..,W0,
- gepackte Doppelwörter (*packed Doublewords – D*): zwei 32-bit-Doppelwörter D1, D0,
- Quadword (*Q*): ein 64-bit-Wort Q0.

Die Teildaten (B, W, D, Q) können als vorzeichenbehaftete oder vorzeichenlose ganze Zahlen aufgefaßt werden.

Befehlssätze

Befehl = Operationscode + Daten = Operationscode + Operanden = Zeiger

Befehlssatz = Menge der von einem Prozessor zur Verfügung gestellten Befehle

Programmcounter = Adressierung des augenblicklich auszuführenden Befehls

Adreßpuffer = Adressierung der Operanden

Zwei Arten von Befehlssätzen:

- Orthogonal : jeder Befehl läßt jede Adressierungsart zu
- Symmetrisch : alle für einen Befehl relevanten Datentypen, Datenformate und Adressierungsarten gelten sowohl für Quell- als auch Zieloperanden

Operationen:

- **monadische:**
  A:= Op B
- **dyadische:**
  A:= C op B
  - Drei-Adreß-Format:
    A,B,C sind verschiedene Variablen
  - Zwei-Adreß-Format
    A:= A op B
    Ergebnisvariable stimmt mit einer Eingangsvariable überein

  - Einadreß-Format
    AC:= AC op B
    Ergebnisvariable A steht im Akkumulator AC (implizite Adressierung)

Little – Endian Format

<Op-Code> <L-Adresse> < H-Adresse>

Big-Endian Format

<OpCode> <H-Adresse> <L-Adresse>

H = höherwertig
L = niederwertig

Beispiel:

ADD $48, $32

- Little Endian Format: => addiert $3248 hinzu
- Big Endian Format: => addiert $4832 hinzu

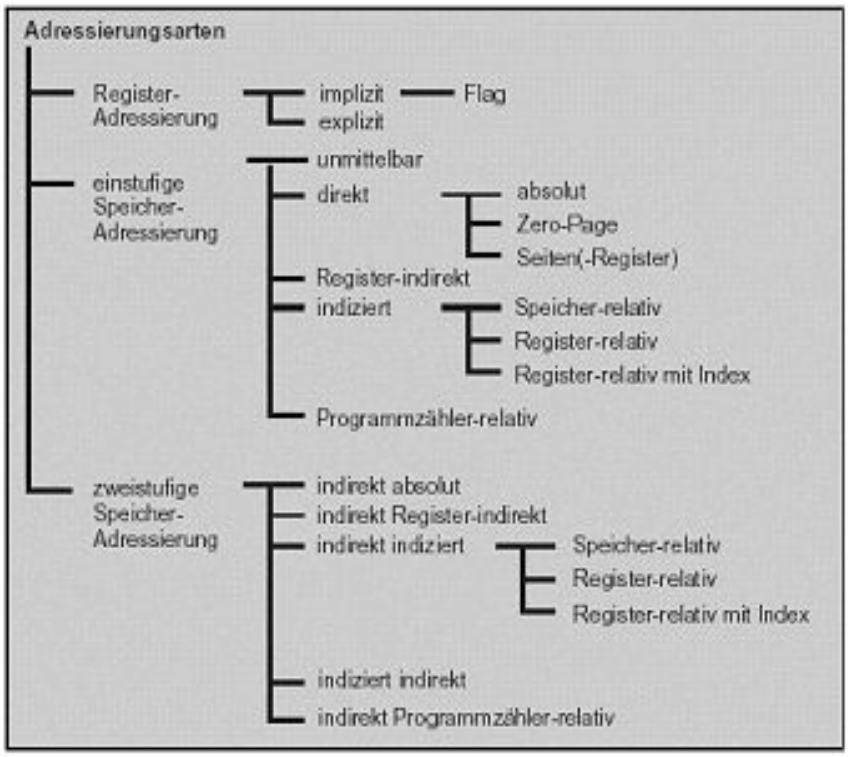
Befehlsgruppen

Tabelle 1.2-1: Die unterscheidbaren Befehlsgruppen

Abschnitt	Befehlsgruppen
	Basis-Befehlsgruppen der Integer-Rechenwerke
1.2.3.1	Arithmetische Befehle
1.2.3.2	Logische Befehle
1.2.3.3	Schiebe- und Rotationsbefehle
1.2.3.4	Flag- und Bit-Manipulationsbefehle
1.2.3.5	Zeichenketten- und Blockbefehle
	Basis-Befehlsgruppen des Steuerwerks
1.2.3.6	Datentransportbefehle
1.2.3.7	Ein-/Ausgabe-Befehle
1.2.3.8	Sprung- und Verzweigungsbefehle
1.2.3.9	Unterprogramm- und Rücksprünge, Software-Interrupts
0	Systembefehle
1.2.3.11	Zusammengesetzte Befehle
	Befehlsgruppen der Gleitpunkteinheiten
1.2.3.12	Arithmetische Befehle, Vergleichsbefehle, Transferbefehle,
	Befehlsgruppen der Multimedia-Rechenwerke
1.2.3.13	Integer-Multimedia-Befehle
1.2.3.14	Gleitpunkt-Multimedia-Befehle

Adressierungsarten

Tabelle 1.3-1: Übersicht über die gebräuchlichsten Adressierungsarten



Registeradressierung

- Operand steht im Register des Prozessors
- Zugriff auf Speicher nicht nötig

Implizite Adressierung

- Effektive Adresse ist im OpCode codiert enthalten
- Ansprechen von nur einem oder wenigen Registern

Assemblerschreibweise: <Mnemo> A
Effektive Adresse: EA ist codiert im Opcode enthalten
Beispiel: LSR A

Flag Adressierung

- Effektive Adresse ist im OpCode codiert enthalten
- Nur ein einzelnes Bit wird angesprochen, nicht ein ganzes Register

Assemblerschreibweise: SE <flag>
CL <flag>
Effektive Adresse: EA ist codiert im Opcode enthalten
Beispiele: SEI, SEC
CLI, CLC

Explizite Adressierung

- steht Operand in allen oder mehreren Registern, wird Nummer des Registers im OpCode angegeben

Assemblerschreibweise: Ri
Effektive Adresse: EA = i
Beispiel: DEC R0

Einstufige Speicher-Adressierung

- für EA nur eine Adreßberechnung notwendig
- EA stets im Adreßpufferregister oder im Programmzähler (PC)

Unmittelbare Adressierung

- Befehl enthält Operanden selbst
- Operand wird durch (PC)+1 adressiert

Assemblerschreibweise: #<Operand>
Effektive Adresse: EA = (PC)+1
Beispiel: LDA #$A3

Direkte Adressierung

- Befehl enthält im Speicherwort nach dem OpCode die (logische) Adresse des Operanden

Assemblerschreibweise: <Adresse>
Effektive Adresse: EA = ((PC)+1)

Absolute Adressierung

- eine Art der direkten Adressierung
- Befehl enthält die vollständige Adresse des Operanden
-

Beispiel: LDA $07FE

Seiten Adressierung

- eine Art der direkten Adressierung
- im Befehl steht als Kurzadresse nur der niederwertige Teil der Operandenadresse
- höherwertiger Teil wird entweder durch Zero-Page Adressierung oder durch Seitenregister-Adressierung ermittelt
- Zero-Page Adressierung
    - Höherwertiger Teil der Adresse wird durch entsprechende Anzahl von '0'-Bits ersetzt
- Seitenregister-Adressierung
    - Höherwertiger Adreßteil steht in einem Register

Beispiel: INC $7F , INC $007F Zero-page
LD R0, >$7F absolute Adresse
LD R0, <$7F Direct page

Register-indirekte Adressierung

- Adressregister enthält die Adresse des Operanden
- Runde Klammern geben an, dass nicht der Inhalt des Registers, sondern der Inhalt der durch das Register adressierten Speicherzelle benutzt wird

Assemblerschreibweise: (Ri)
Effektive Adresse: EA = (Ri)

Postinkrement

- Inhalt des Registers wird nach der Ausführung des Befehl erhöht

Assemblerschreibweise: (Ri)+
Effektive Adresse: EA = (Ri)

Predekrement

- Inhalt des Register wird vor der Ausführung des Befehls erniedrigt

Assemblerschreibweise: -(Ri)
Effektive Adresse: EA = (Ri)-1

Beispiele:
LD R1, (A0)
INC (R0)+
CLR –(R0)

ST Ri, -(SP) Store
LD Ri, (SP)+ Load

Indizierte Adressierung

- Berechnung der EA durch die Addition des Inhalts eines Registers zu einem angegebenen Basiswert

Speicher-relative Adressierung

- Basiswert wird als absolute Adresse im Befehl vorgegeben

Assemblerschreibweise: <Adresse>(Ii)
Effektive Adresse: EA = ((PC)+1)+(Ii)
Beispiel: ST R1, $A704 (R0)

Register-relative Adressierung

- Basiswert befindet sich in einem Basisregister
- Im Befehl wird Offset angegeben, der zum Inhalt des Basisregisters addiert wird

Assemblerschreibweise: <Offset> (Bi)
Effektive Adresse: EA = (Bi) +((PC)+1)
Beispiel: CLR $A7 (B0)

Register-relative Adressierung mit Index

- Basiswert wird in einem Basisregister übergeben
- Dazu wird Inhalt eines Indexregisters addiert

Assemblerschreibweise: <Offset> (Bi) (Ij)
Effektive Adresse: EA = (Bi) + (Ij) + ((PC)+1)
Beispiel: DEC $A7 (B0) (I0)+

Programmzähler-relative Adressierung

- EA entsteht durch die Addition eines im Befehl angegebenen Offsets zum aktuellen Programmzählerstand

Assemblerschreibweise: <Offset> (PC)
<Offset>
Effektive Adresse: EA = (PC) +2+((PC)+1)
Beispiel: BNE $7F
LBRA $7FFF

Zweistufige Speicher-Adressierung

- Zur Bestimmung der EA sind mehrere sequentiell auszuführende Adreßberechnungen und Speicherzugriffe notwendig
- 1. Berechnung liefert Adresse eines Speicherwortes, in dem wiederum eine Adresse für die folgende Adreßberechnung zu finden ist

Indirekte absolute Adressierung

- Befehl enthält eine absolsute Adresse, die auf ein Speicherwort zeigt.
- Inhalt des Speicherwortes ist die EA des gesuchten Operanden

Assemblerschreibweise: (<Adresse>)
Effektive Adresse: EA = (((PC)+1))
Beispiel: LDA ($A347)

Indirekte Register-indirekte Adressierung

- Das im Registerfeld des OpCodes ausgewählte Register enthält die Adresse eines Speicherwortes, in dem seinerseits die EA des Operanden steht

Assemblerschreibweise: ((Ri))
((Ri)+)
(-(Ri))
Effektive Adresse: EA = ((Ri))
EA = ((Ri))
EA = ((Ri)-1)

Beispiel: LD R0, (-(R1))

Indirekte indizierte Adressierung

- Nach dem Indizierungsverfahren zur Gewinnung einer indizierten Adressierung wird eine effektive Zwischenadresse gebildet. Diese EA zeigt auf ein Speicherwort, das die EA des gesuchten Ooeranden enthält.

Assemblerschreibweise: (<Adresse>(Ii)) indirekt Speicher-relativ
(<Offset>(Bi)) indirekt Register-relativ
(<Offset>(Bi)(Ii)) indirekt Register-relativ mit Index

Effektive Adresse: EA = ((Ii)+((PC)+1))
EA = ((Bi) + ((PC)+1))
EA = ((Bi) + (Ii) + ((PC)+1))

Indizierte indirekte Adressierung

- Aus dem aktuellen Inhalt des Programmzählers und einem Offset wird eine Zwischenadresse gebildet. Diese zeigt auf ein Speicherwort, das die EA des Operanden enthält

Assemblerschreibweise: <2.Offset>(<1.Offset>(Bi))(Ij)
Effektive Adresse: EA = ((Bi) + ((PC)+1))+(Ij)+((PC)+2)
Beispiel: INC $A7($10(B0))(I2)

Indirekte Programmzähler-relative Adressierung

Assemblerschreibweise: (<Offset>(PC))
Effektive Adresse: EA = ((PC)+2+((PC)+1))
Beispiel: JMP ($A7(PC))

Kurseinheit 2:

Befehlsverarbeitung

Phasen der Befehlsbearbeitung

Holphase
Befehlsregister ist zu Warteschlange (Prefetch Queue) erweitert: Abarbeitung nach FIFO
Prefetch Queue wird durch Prefetcher der Ablaufsteuerung gefüllt

Decodierphase
Befehle werden aus der Prefetch Queue in die Decodier-Einheit gebracht und decodiert

Ausführungsphase
Ausführung der Befehle erfolgt durch die Steuerungen des Operationswerks
Verarbeitungseinheiten erhalten Operanden aus den Registern

Rückschreibphase
Ergebnisse werden in Registern abgespeichert

Fließbandverarbeitung (Pipeline)

3-stufige Pipeline

Holphase (Instruction Fetch Phase)
Decodierphase (Decode Phase)
Ausführungsphase (Execution Phase) (enthält Write Back)

4-stufige Pipeline

Holphase
Decodierphase
Ausführungsphase
Rückschreibphase

5-stufige Pipeline

Holphase
Decodierphase
Ausführungsphase
Speicherzugriffsphase (Load / Store)
Rückschreibphase

Pipeline-Hemmnisse

- Betriebsmittelabhängigkeit (Resource Dependency)
- Datenabhängigkeit (Data Dependency)
- Kontrollflußabhängigkeit (Control Dependency)

Betriebsmittelabhängigkeit

- tritt auf, wenn zwei Befehle auf dieselbe, nur einmal vorhandene Komponente des Prozessors zugreifen wollen (z.B. Bus, Operationswerk, Register)

Möglichkeiten für Betriebsmittelabhängigkeit

- kleinere Zykluszeit des Prozessors als der Arbeitsspeicher, daher Anhalten der Pipeline um mindestens einen Takt (Hardware Interlocking) bis Speicher Datum bereitgestellt hat, da bei den meisten Prozessoren kein Overlapping möglich (Parallelbearbeitung)

- Operationswerke mit unabhängigen Pipelines unterschiedlicher Länge, aber demselben Registersatz

Pipeline 1	**Pipeline 2**
Befehl 1	Befehl 4
Befehl 2	Befehl 5
Befehl 3	

Befehl 4 wäre vor Befehl 3 abgearbeitet, da in der kürzeren Pipeline; Registerwert würde durch Ergebnis von Befehl 4 überschrieben ohne Befehl 3 bearbeitet zu haben (Fehler !!)

- Register(namen)abhängigkeit
- Abhilfe: Anhalten der kürzeren Pipeline durch Hardware Interlocking oder softwaremäßig durch Auswählen eines anderen Ausgaberegisters für Befehl 4

Datenabhängigkeiten

- Das durch einen Load-Befehl aus dem cache gelesene Datum steht zu spät zur Verfügung, als dass es noch im folgenden Befehl verarbeitet werden kann
- Datum, das in einem Befehl gebraucht wird , wurde vorher noch nicht berechnet

Möglichkeiten der Abhilfe:

- Anhalten der Pipeline durch Hardware Interlocking
- Bypass Bus
  => Datum wird auch auf ALU-Eingänge gelegt beim Übertragen auf Registersatz

- Software
  - Einfügen von NOP-Befehlen, bis Datum zur Verfügung steht

  - Umplatzieren von nicht datenkritischen Befehlen (z.B. Sub vor Mult wenn nicht gleichzeitig auf R1 zugegriffen wird) bei optimierendem Compiler

    Datenabhängigkeit, wenn Ergebnis einer Operation von einem folgenden Befehl als Eingabewert benötigt wird:

Kontrollflußabhängigkeiten

- tritt im Zusammenhang mit bedingten Verzweigungen auf, wenn die vorgegebene Bedingung nicht rechtzeitig ausgewertet wird

Lösung des Problems:

- Einfügen von NOP-Befehlen

- Bei RISC-Prozessoren oft nur Einfügen eines Befehls(NOP) hinter dem Verzweigungsbefehl notwendig (Delayed Branch)
  Positionen hinter den Verzweigungsbefehlen => Branch Delay Slots

- Optimierende Compiler ersetzen NOPs durch Befehle, die unabhängig sind und auf jeden Fall durchgeführt werden müssen

Superskalarität

Parallel-Pipelines

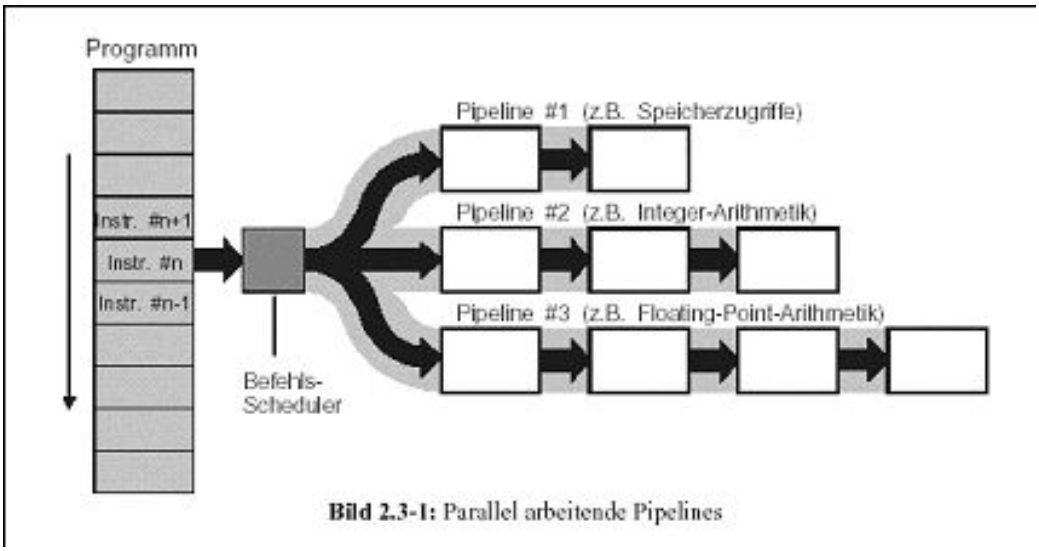


Bild 2.3-1: Parallel arbeitende Pipelines

Parallele Pipelines werden jeweils einem der Operationswerke zugewiesen (z.B. bei CISC-Prozessoren)

Probleme bei Parallel-Pipelines:

- wegen unterschiedlicher Längen der Pipelines können sich Befehle in verschiedenen Pipelines überholen
- Datenabhängigkeiten entstehen nun auch zwischen den einzelnen Pipelines
- Im Mittel kann höchstens 1 Befehl pro Taktzyklus fertig gestellt werden: IPC <= 1
  Vergrößerung des Wertes über 1 nur möglich, wenn in jedem Takt mehr als ein Befehl aus dem Befehlsstrom entnommen und initialisiert werden kann

Prozessoren, deren Scheduler pro Taktzyklus mehrere Befehle gleichzetig initialisieren kann, nennt man superskalare Prozessoren. Anzahl der maximal pro Takt zu initialisierenden Befehle nennt man Superskalaritätsgrad. Übliche Werte sind zwischen 2 und 6.

Statische Befehlszuteilung

Befehlszuteilung erfolgt hier durch den Compiler.
Scheduler entnimmt hier jeweils ein Paar von Befehlen und übergibt diese an zwei Decoder:

- erster Befehl an den „unteren“ Decoder
- zweiter Befehl an den „oberen“ Decoder

Jeder Decoder kann widerum mit mehreren Pipelines verbunden sein.

Dynamische Befehlszuteilung

Befehlszuteilung erfolgt zur Laufzeit, nicht schon beim Compiler.
Scheduler hat folgende Aufgaben:

- Befehle holen
- Befehle an die mehrfach ausgelegten Decoder übergeben
- Operanden aus den Registern holen
- Decodierte Befehle mit ihren Operanden einer freien Pipeline zuweisen

Zuteilung der Befehle zu den Pipelines erfolgt nach prozessor-spezifischen Strategien.

Mögliches auftretendes Problem: Wartezeiten, wegen Betriebsmittelabhängigkeit (es steht keine freie Verarbeitungseinheit zur Verfügung)

Verzweigungsvorhersage

Weshalb macht man Verzweigungsvorhersagen?

Problem: Bei der Fließbandverarbeitung geschieht die Auswertung der Statusbits, von denen die Ausführung der Verzweigungsbefehle abhängt, erst in späteren Pipelinestufen. Dies führt zu Verzögerungen.

Um dieses Problem zu lösen, bedient man sich der Verzweigungsvorhersage:

- zwei Befehlswarteschlangen unterhalten
- Sprungzielvorhersage
    - Statische Vorhersage
    - Dynamische Vorhersage

Statische Vorhersagen

- alle Verzweigungsbefehle werden gleich vorhergesagt, d.h. alle als „durchgeführt“ oder „nicht durchgeführt“ gekennzeichnet
- bei komplexeren Verfahren hat jeder Verzweigungsbefehl eine Grundeinstellung, die durch zusätzliches Steuerbit im OpCode vorgegeben ist: z.B. Rückwärtsverzweigungen werden durchgeführt, Vorwärtsverzweigungen nicht

Statische Vorhersagen sagen in der Regel nur einen Sprung vorher. Bei weiteren Verzweigungsbefehlen innerhalb eines solchen Zyklusses, wird das vorzeitige Laden der Befehlswarteschlange angehalten.

Dynamische Vorhersagen

Es werden die jeweils zuletzt ausgeführten Verzweigungsbefehle „beobachtet“ und zu jedem Branch-Befehl in wenigen Bits die Vorgeschichte aufgezeichnet. Bei einer erneuten Ausführung eines Branch-Befehls, trifft die Hardwarelogik dann anhand der Vorhersagebits ihre Entscheidung. Nachdem feststeht, ob die Verzweigung richtig vorhergesagt wurde oder nicht, werden die Vorhersagebits ggf. modifiziert.

Als Ausgangszustand werden die Historybits mit bestimmten Werten vorbelegt:

Speicherung der Vorhersagebits

Es gibt 3 Möglichkeiten:

- Vorhersagebits im Befehls-Cache
- Branch History Table und Branch Target Cache
- Branch Target Buffer

Spekulative und ungeordnete Befehlsausführung

Spekulative Befehlsausführung

Bei der Sprungzielvorhersage wird davon ausgegangen, dass auch die Befehle an der vorhergesagten Zieladresse bereits ausgeführt werden können, bevor die Sprungvorhersage in späteren Pipelinestufen aufgelöst wird. Diese Ausführung auf Verdacht nennt man spekulative Befehlsausführung.

War die Verzweigung falsch vorhergesagt, so müssen die Ergebnisse der spekulativ ausgeführten Befehle rückgängig gemacht werden.

Ungeordnete Befehlsausführung

Um Wartezeiten bei der Befehlsausführung zu vermeiden, versuchen moderne superskalare Mikroprozessoren, ihre Befehle ungeordnet, d.h. außerhalb der durch das Programm vorgegebenen Reihenfolge, den Verarbeitungseinheiten zuzuweisen und dort ausführen zu lassen.

Cache-Speicher

Lage des Cache im Mikrorechner

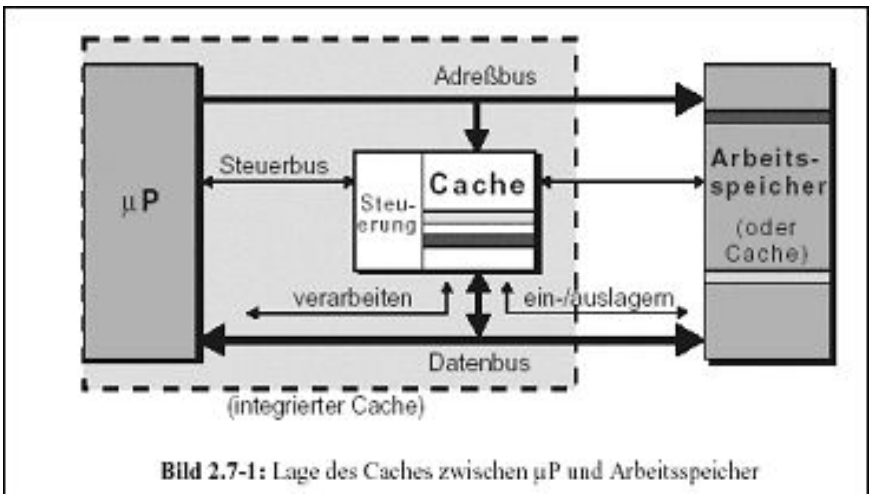


Bild 2.7-1: Lage des Caches zwischen µP und Arbeitsspeicher

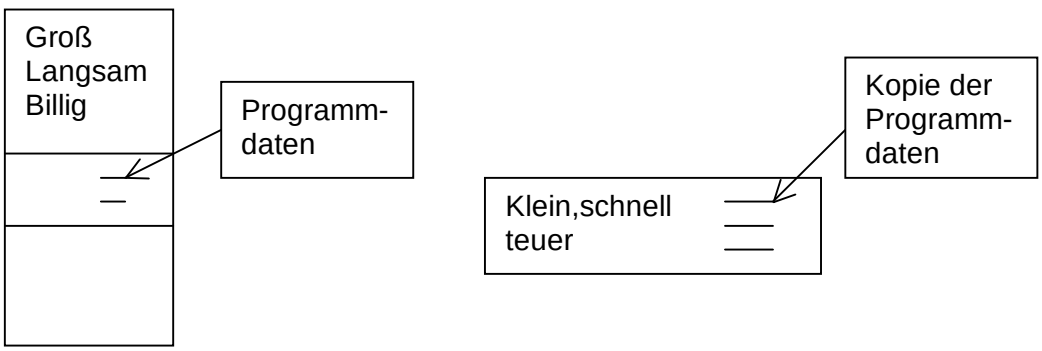
Caching:

Idee eines Caches:
Cache gibt es überall und häufig verwendet

Es gibt 2 Sorten von Speichern:

a) groß, langsam, billig
b) klein, schnell, teuer

Nachteil großer Speicher: unübersichtlich



Vorteil Cache: Ersparnis von Zugriffszeit

Vermutung: Man geht davon aus, dass Daten geordnet vorliegen, in der Hoffnung, dass man die richtigen Daten erwischt hat.

Wo ist der Gewinn: wenn das Datum tatsächlich im Cache gefunden wird
Treffer (Hits) im Cache sind daher möglichst hoch zu halten
Bei Miss wird ein Verlust an Zugriffszeit gemacht

Idee: man macht den Cache ziemlich groß
Nachteil: Cache ist äußerst teuer

486-er hatte als erster Prozessor einen internen Cache von 8 K und lief mit Prozessortakt

Hauptspeicherzugriffszeit: 16 ns

Aufbaustruktur eines Caches:
HS + 2nd Level Cache + 1st Level Cache

Datencache + Befehlscache

2 x 64 KB first Level Cache das erste Mal bei Pentium PC verwendet

aus Kostengründen wird der first Level Cache manchmal auch weggelassen

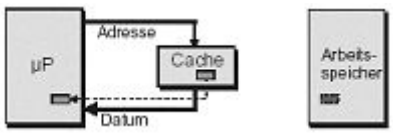
Hauptspeicher: besteht aus dynamischen RAMs
Cache: besteht aus statischen RAMs

Arbeitsweise eines Cache:

Lesen:
Die CPU schaut im Cache nach, ob Datum darin steht.

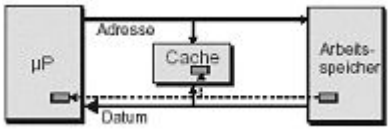
Read Hit:

- **è** Datum gefunden:
  Datum wird aus dem Cache gelesen



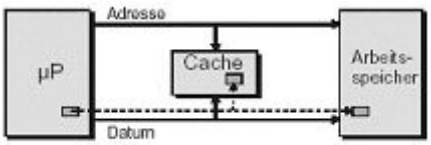
Read Miss:

- **è** Datum nicht gefunden:
  Es wird im Hauptspeicher nach dem Datum gesucht
  Datum wird in den Cache geladen und erst dann aus dem Cache gelesen



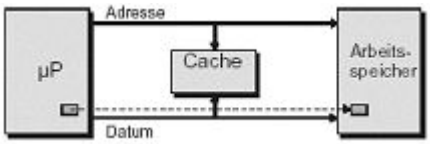
Schreiben:

Write Hit:

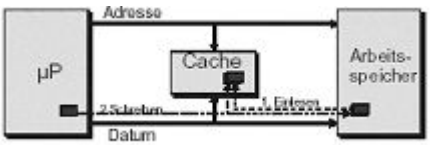


Write Miss:

Ohne Allocation:



Mit Allocation:



CPU legt das Datum im Cache ab
Datum , das im Cache geändert wurde, muß aus dem Cache in den Hauptspeicher geschrieben werden (**Write-through**)
Steht das Datum nicht im Cache, so muß nicht auf den Cache geschrieben werden. Das Datum wird direkt in den Hauptspeicher geschrieben (nur bei Miss, nicht bei Hit)

Rückschreibeverfahren:

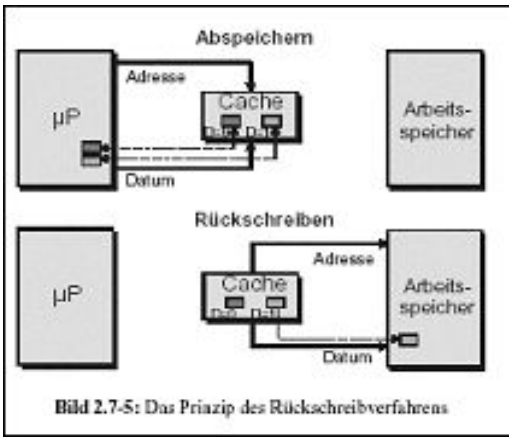


Bild 2.7-5: Das Prinzip des Rückschreibverfahrens

Wenn Datum immer nur im Cache geändert wird, und erst dann in den Hauptspeicher geschrieben wird, wenn Cache voll ist und geräumt werden muß **(Write-back)**

Beim Write-back entsteht eine Diskrepanz zwischen Datum und Hauptspeicher. Um dies zu umgehen wird das sogenannte „dirty“- Bit gesetzt beim Datum im Hauptspeicher.
Die Schreibzyklen erfolgen aber immer nur auf dem Cache.

Nachteil: mehr Aufwand an Verwaltung, da hier zusätzlich Speicher für das Dirty-Bit zur Verfügung gestellt werden muß

Bei 4-Wege-Speicher ist Wahrscheinlichkeit größer, dass Datum im Cache gefunden wird.

Der Direct Mapped Cache

2.7.3.1 Der *Direct Mapped Cache*

Wegen der erwähnten Probleme, vollassoziative Caches großer Kapazität zu bauen, wird bei den Cache-Speichern für Mikrorechner-Systeme eine Mischform aus inhalts- und ortsadressierten Speichern eingesetzt, wie sie im Bild 2.7-7 schematisch dargestellt ist.

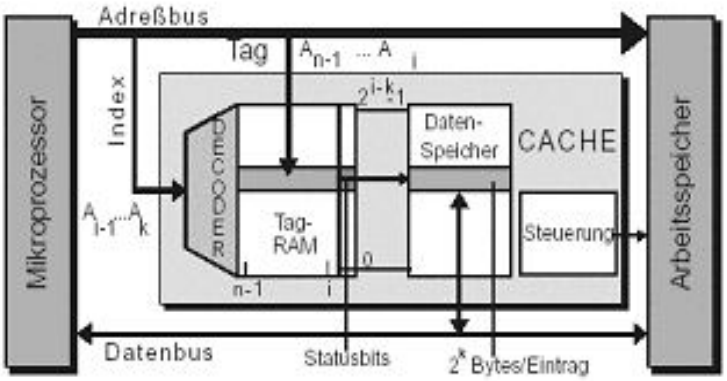


Bild 2.7-7: Aufbau eines Cache-Speichers für Mikrorechner-Systeme

Aufteilung einer Adresse beim direct mapped cache:

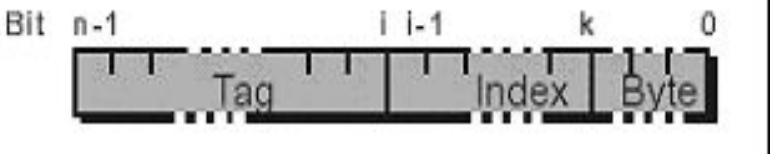


Bild 2.7-8: Aufteilung einer Adresse

- Paralleler Adreßvergleich wird reduziert auf den Vergleich zwischen den höherwertigen Bits der angelegten Adresse und demjenigen Tag, der durch die Adreßbits $A_{i-1} \ldots A_k$ selektiert wird.

Der n-Way Set Associative Cache

- mehrere Direct Mapped Caches werden verwendet und durch eine Auswahllogik wird entschieden, in welchen der (Teil-)Caches das Datum bei einem Read Miss eingetragen wird.
- Die Mengen der Einträge, die jeweils zu den verschiedenen Indizes gehören, bezeichnet man als Sets. Daher der Name n-Way Set Associative Cache.
- Übliche Werte für n sind 2 bis 8

- Beim 2-Way Set Associative Cache geschieht die Auswahl des Caches durch ein zusätzliches Bit (SB), das den Weg anzeigt, in den das Datum geschrieben wird.

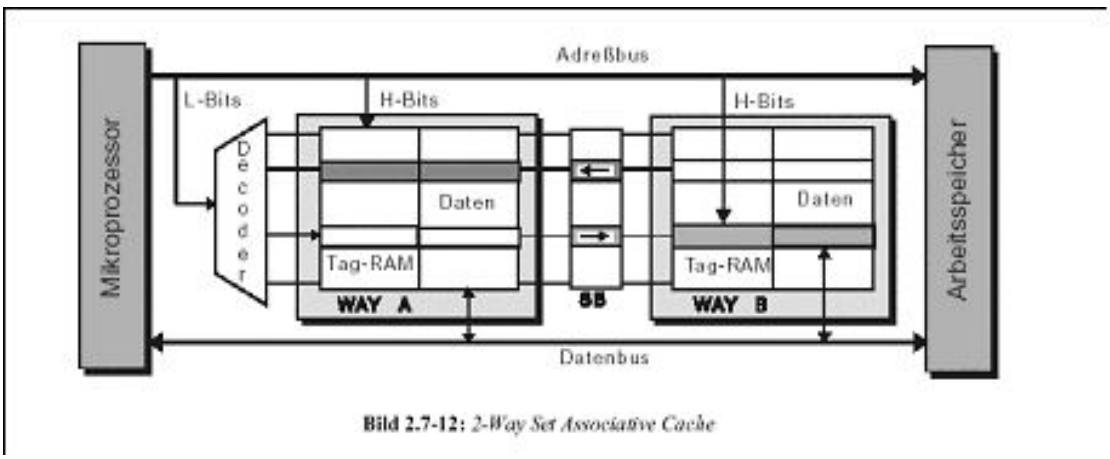


Bild 2.7-12: *2-Way Set Associative Cache*

Verdrängungsstrategien

- **FIFO**
  Cache-Eintrag, der zuerst eingelagert wurde, wird auch zuerst verdrängt.

- **Random**
  Nächster zu überschreibender Eintrag wird durch eine Pseudo-Zufallsgenerator ausgewählt

- **LRU-Strategie**
  Der Eintrag, auf den am längsten nicht zugegriffen wurde, wird überschrieben

- **Pseudo-LRU**
  Controller hält in wenigen Bits pro Index fest, welcher der Einträge mit gleichem Index am längsten nicht mehr angesprochen wurde. Dieser ist dann zu verdrängen.

Typische Cache-Eigenschaften

L1-Cache

- Kapazität zwischen 16 kByte und 64 kByte
- Meist n-Way Set Associative Cache mit n=2,4 oder 8
- Gebräuchlichste Verdrängungsstrategie: Pseudo- LRU

L2-Cache

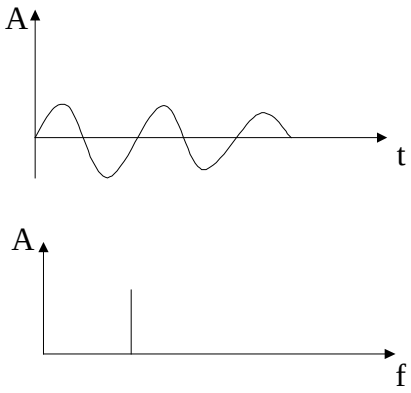
- Direct Mapped Cache oder n-WaySet Associative Cache mit n=2,3,4 oder 8
- Enthält in der Regel Befehle und Daten gemeinsam
- Kapazitäten Datenspeicher externer L2-Caches: einige Megabyte (1 – 16)
- Gebräuchlichste Form des Schreibzugriffs: Durchschreibverfahren

Kurseinheit 3:

Digitale Signalprozessoren

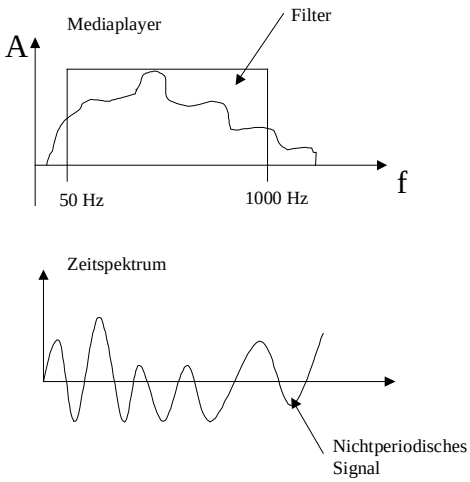
Fourier-Transformation

Fourier-Transformation



Fourier-Transformation: vom Zeit- in den Frequenzbereich kommen

Fourier-Transformation



FFT: Fast-Fourier-Transformation
Butterfly-Diagramm elementarer Bestandteil der FFT

Anmerkung: FFT besitzt keine nennenswerten Zeitverzögerungen und ist möglichst schnell.

Umwandlung Analog-Digital-Signal

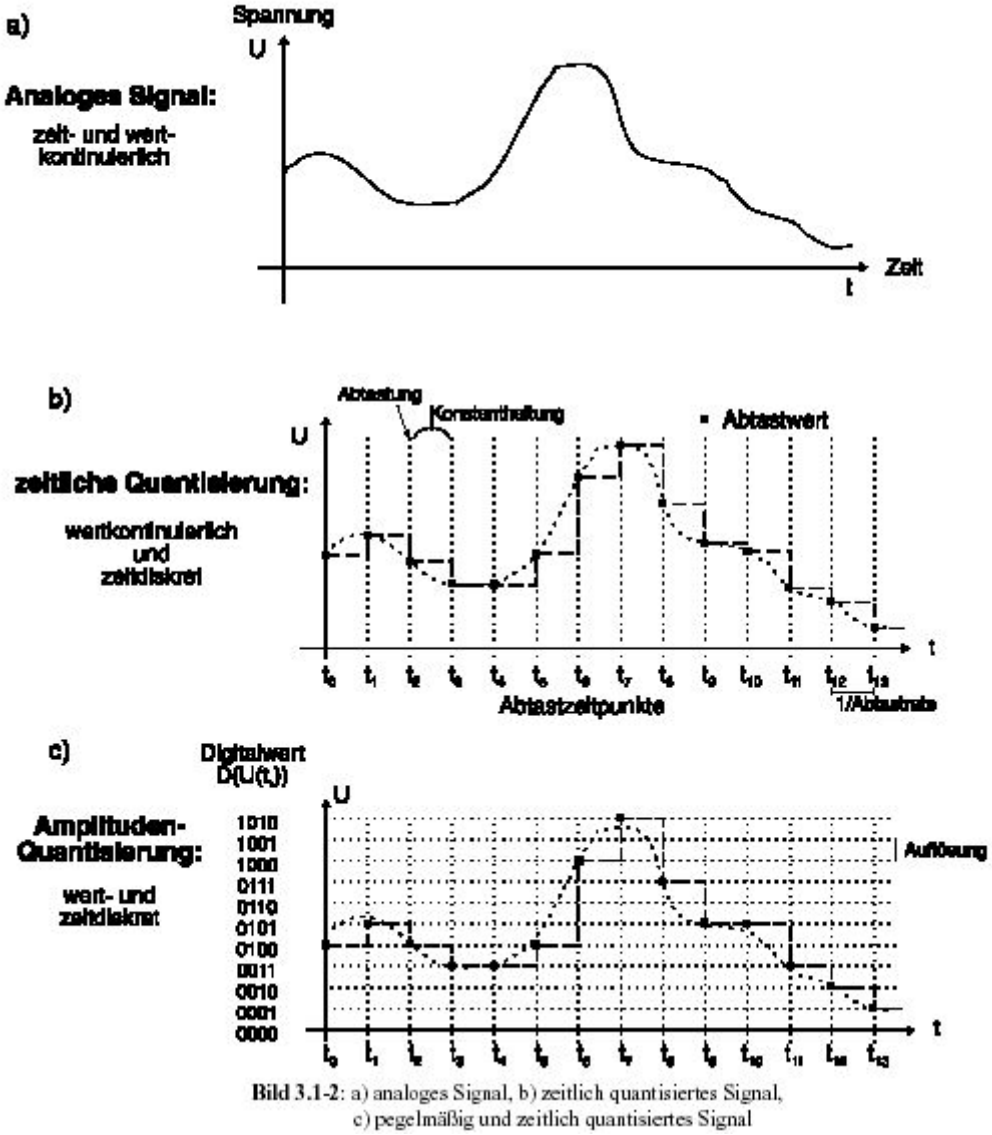


Bild 3.1-2: a) analoges Signal, b) zeitlich quantisiertes Signal, c) pegelmäßig und zeitlich quantisiertes Signal

Basisarchitektur Digitaler Signalprozessoren

- Digitale Signalprozessoren sind in Harvard-Architektur realisiert
- Durch gleichzeitige Adressierung der Speicher wird Einsatz von getrennten Adreß-Werken gefordert, daher kommt Von-Neumann-Architektur nicht in Frage
- Oftmals auch so realisiert, dass die internen Speicher zwar voneinander getrennt sind, jedoch nach außen nur ein gemeinsames Bussystem geführt wird
- Leistungsfähigster DSP mit 1 Programmbussystem und 3 Datenbussysteme

DSP-Einsatzbereiche

- Telekommunikation und Nachrichtentechnik
- Konsumerbereich
- Datenverarbeitung, Computer
- Multimediabereich
- Automobilbereich
- Industrie
- Medizintechnik
- Militär
- Meßtechnik

Bitreverse Adressierung

Dezimal	Binär	Bitrevers	Dezimal
1	00000001	10000000	128
7	00000111	11100000	224
27	00011011	11011000	216
217	11011001	10011011	155

ADSP-218x

Die Zahlenformate des ADSP-218x

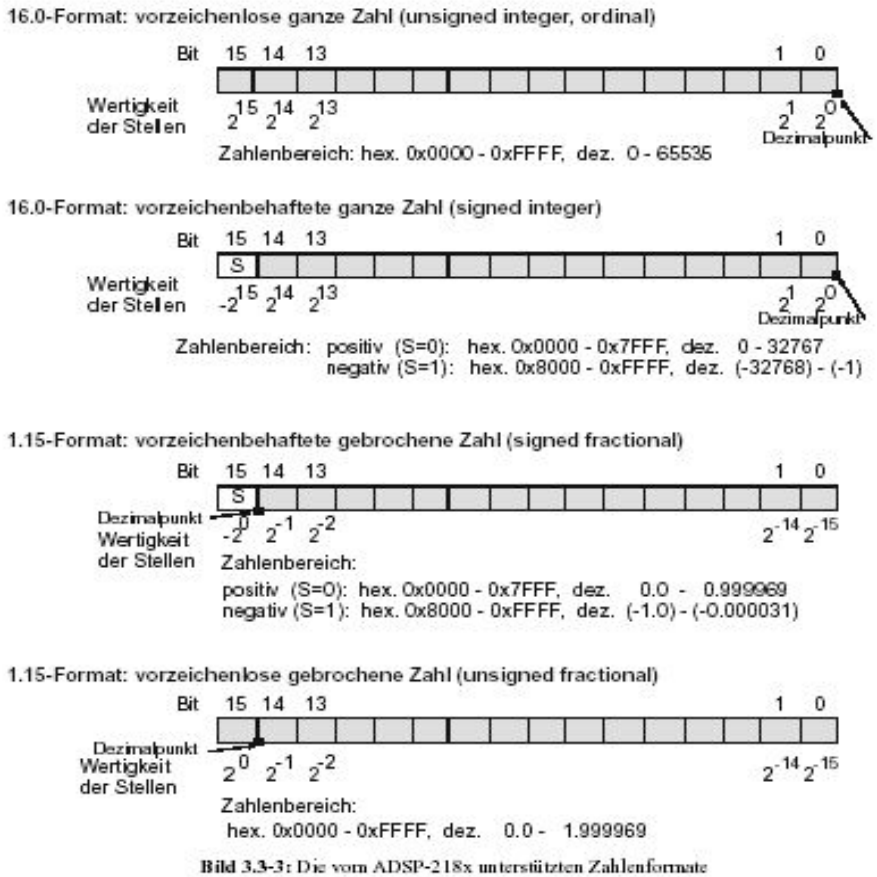
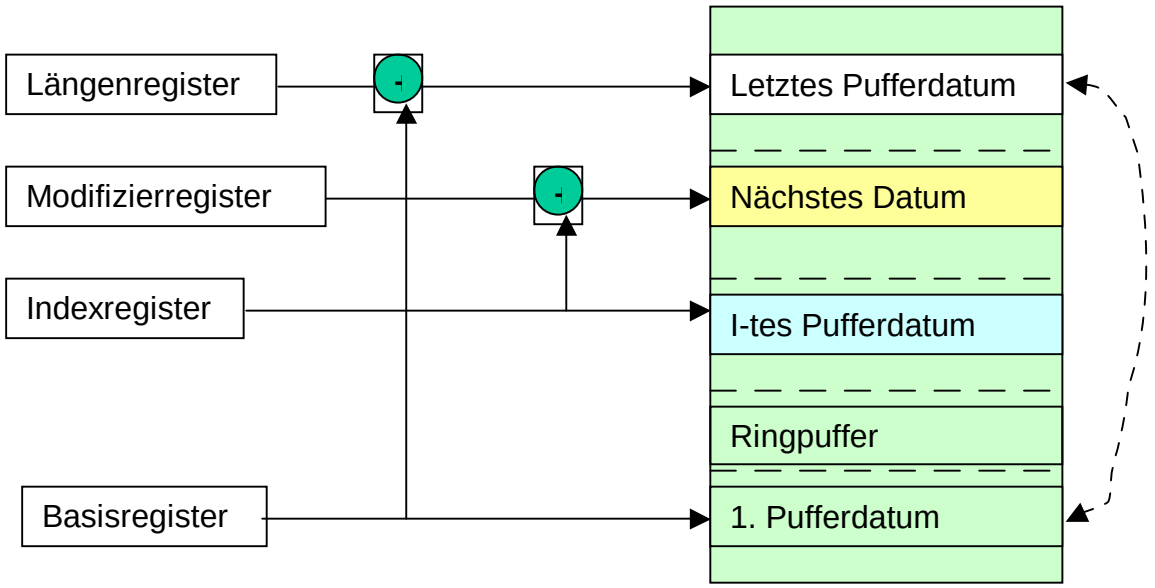


Bild 3.3-3: Die vom ADSP-218x unterstützten Zahlenformate

Ringpuffer-Adressierung



Ringpuffer-Adressierung ADSP-218x

I1

M1

L1: Länge des Ringpuffers

0	0	0	0	0	0	0	0	0	0	0	1	0	1

Ohne Post-Inkrement:

I1

M1

0	0	0	0	0	0	0	0	0	0	0	0	0	0

L1: Länge des Ringpuffers

0	0	0	0	0	0	0	0	0	0	0	1	0	1

Mit Post-Inkrement: |M| < L

I1

M1

0	0	0	0	0	0	0	0	0	0	0	0	1	0

L1: Länge des Ringpuffers

0	0	0	0	0	0	0	0	0	0	0	1	0	1

Beispiel:

I1: $0033

0	0	0	0	0	0	0	0	1	1	0	0	1	1

M1: $2

0	0	0	0	0	0	0	0	0	0	0	0	1	0

L1: Länge des Ringpuffers : $5

0	0	0	0	0	0	0	0	0	0	0	1	0	1

Basis-Adresse:

$B = n * 2^{k}$

B = $ 10

I = (Ia + M – B) mod L + B

I = ($33 + $2 - $10) mod $5 + $10 = $25 mod $10 = $2

Ergebnis:

I1: $0002

0	0	0	0	0	0	0	0	0	0	0	0	1	0

M1: $2

0	0	0	0	0	0	0	0	0	0	0	0	1	0

L1: Länge des Ringpuffers : $5

0	0	0	0	0	0	0	0	0	0	0	1	0	1

Adressierung mit Bitumkehr:

Basis-Adresse: $0033
Bitreverse Form:
00000000110011 ->11001100000000

I1: $0033 Bitrevers

1	1	0	0	1	1	0	0	0	0	0	0	0	0

M1: $2

0	0	0	0	0	0	0	0	0	0	0	0	1	0

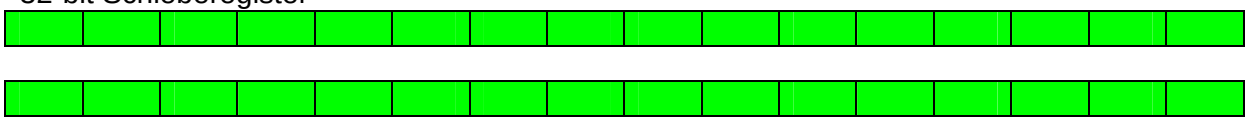
L1: Länge des Ringpuffers : $5

0	0	0	0	0	0	0	0	0	0	0	0	0	0

Schiebe/Normalisiereinheit des ADSP-218x

Shifter Array:

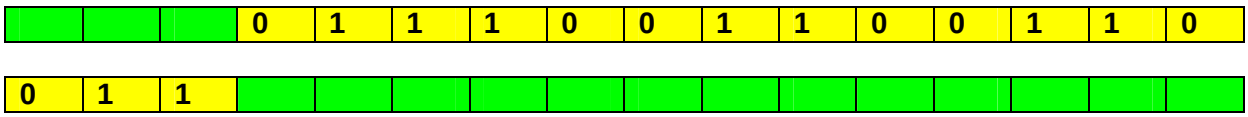
32-bit Schieberegister



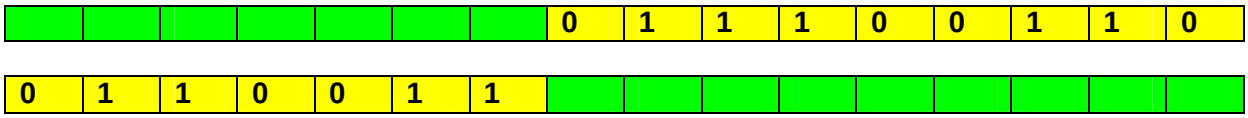
16-bit Eingabeoperand:

0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1

Platzierung des Eingabeoperands im 32-bit-Register:



Oder:



Rein theoretisch ist also eine beliebige Ablage des Operanden im Schieberegister möglich.

Die Platzierung wird durch den Zustand des HI/LO-Signals am R-Eingang des Shifter Arrays:
SR1:

SR0:

Ist HI gesetzt, so ergibt sich folgender Zustand:
SR1:

0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1

SR0:

Ist LO gesetzt, so ergibt sich folgender Zustand:
SR1:

SR0:

0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1

Schieben der Registerinhalte in der Schiebe/Normalisiereinheit des ADSP-218x

Registerinhalt:

1	1	1	0	0	1	1	0

Arithmetisches Rechtsschieben:

1	1	1	1	0	0	1	1

^

|____ hier wird das Vorzeichen nachgezogen

logisches Rechtsschieben:

0	1	1	1	0	0	1	1

^

|____ hier wird eine Null nachgezogen

Linksschieben:

1	1	0	0	1	1	0	0

^

hier wird eine Null nachgezogen ____|

Daten-Adressgeneratoren:

DAG1 / DAG2

Registerbank 1: **Indexregister**

I0:

I1:

I2:

I3:

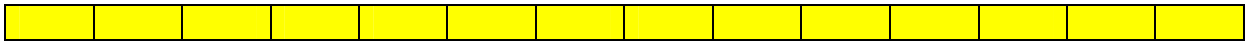
Registerbank 2:**Modifizierregister**

M0:

M1:

M2:

M3:



Registerbank 3:**Längenregister**

L0:

L1:

L2:

L3:

Adressierungsarten:

Direkte Adressierung:

Operandenadresse:

0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Wert: 0x3800
Befehl: AX0 = DM(0x3800)

Register-indirekte Adressierung

Allgemein:

I1

M1

L1

0	0	0	0	0	0	0	0	0	0	0	0	0	0

Ohne Post-Inkrementierung:

I1

M1

0	0	0	0	0	0	0	0	0	0	0	0	0	0

L1

0	0	0	0	0	0	0	0	0	0	0	0	0	0

Mit Post-Inkrementierung:

I1

M1: gibt Wert an, um den I1 nach dem Zugriff **erhöht** wird

0	0	0	0	0	0	0	0	0	0	0	0	1	0

L1

0	0	0	0	0	0	0	0	0	0	0	0	0	0

Oder:

I1

M1: gibt Wert an, um den I1 nach dem Zugriff **erniedrigt** wird

1	0	0	0	0	0	0	0	0	0	0	0	1	0

L1

0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kurseinheit 4

PCI-Bus

- synchroner Parallelbus mit 33 MHz oder 66 MHz getaktet
- Bustakt muß mit Takt der CPU synchronisiert werden
- 32 oder 64 Adreß-/Datenleitungen
- Übertragung der Daten im Multiplexbetrieb

Vorteil: hohe Übertragungsrate
Nachteil: bietet keine Multiprozessor-Unterstützung
Relativ hohe Anforderungen an Betriebssystem zur Konfiguration des Rechnersystems

Busankopplung: über Treiber mit Tristate- Ausgängen

Koppeleinheiten:
Hostbrücke
Standardbus-Brücke
PCI/PCI-Brücken

Synchronisations- und Übertragungsverfahren:
- Einzelwort-Transfer
- Bustakt über die Leitung CLK (Clock) übertragen
- Zwischen zwei Busaktionen muß ein Ruhezyklus bzw. Turn-around Cycle eingelegt werden

PCI-Kommandos:
- Schreib-/Lesezugriffe auf die 3 Adreßräume
- Befehle zur Übertragung von Cache-Blöcken
- Interrupt Anforderungen
- Senden von Runspruch-Nachrichten an alle Busteilnehmer

Steuerung der Datenbusbreite: Automatische Anpassung der Datenbreite auf dem PCI-Bus

Adressierung der Busteilnehmer:
3 Adreßräume:
- Speicher-Adreßraum
- Ein-/Ausgabe-Adreßraum
- Konfigurationsbereich

Speicheradreßraum:
- 2^32 oder 2^64 Adressen
- Auf 32-bit-Bus Übertragung von 64-Bit-Adressen möglich
- Auf 64-bit-Bus Übertragung von 32-Bit-Adressen möglich

Ein-/Ausgabe-Adreßraum:
- 2^32 Adressen
- höherwertige 30 Bits bestimmen eine Doppelwortadresse
- niederwertige Bits werden zur Erzeugung der Signale BE[3::0] benutzt

Konfigurationsbereich:

- in jeder PCI-Einheit bis zu 64 Doppelwortregister (=256 Bytes)
- Zugriffsarten:
  - Typ 1 ermöglicht Adressierung in Systemen mit mehrfachen PCI-Bussen
  - Typ 0 regelt Adressierung in einfachen PCI-Bus-Systemen
  - Typ 2 ist für x86-Systeme reserviert

AGP-Port

- Ansteuerung von Graphikkarten
- PCI-Bus „Flaschenhals“ für graphische Anwendungen
- Hostbrücke wurde um einen speziellen Datenpfad für Graphikdaten erweitert
  ð Accelerated Graphics Port (AGP)
- AGP ist 32 bit breit und mit 66 MHz getaktet

Vorteil: Schneller Verbindungsweg zwischen Graphik-Controller und Hauptspeicher

SCSI-Bus

- geräteunabhängiger Parallelbus für den Anschluß verschiedenster Peripheriegeräte
- einfache Realisierung hat 8 Signalleitungen
- Wide-Realisierung hat 16 Signalleitungen
- Übertrageung erfolgt entweder über einfache Signalleitungen oder differentiell über Leitungspaare
- Maximale Länge Bus: 25 m
- Befehle und Nachrichten werden stets asynchron übermittelt
- Verdoppelung der maximalen Übertragungsrate durch übertragen eines Datums bei jeder Taktflanke
- SCSI-Bus ermöglicht Einsatz von mehreren Initiatoren im Bus; jede Komponente hat bis zu 8 logische
  Einheiten (LUNs)

Anschluß des SCSI-Bus

- Flachbandkabel, das durch alle Knoten geschleift wird
- Kabel muß an beiden Enden abgeschlossen (terminiert) sein
- Kabel umfaßt 8 oder 16 Datenleitungen oder Datenleitungspaare
- Anschluß der Geräte über SCSI-Adapter

Synchronistations- und Übertragungsverfahren

- bei 8-bit-Version 50 Leitungen
- bei 16-bit-Version 68 Leitungen
- 16-bit-Bus besitzt für je 8 Datenleitungen ein eigenes Handshake-Signalpaar REQ/ACK bzw. REQB/ACKB
- alle Signale im Low-Pegel aktiv
- es gibt 2 Arten von Übertragungen:
  - asynchrone Datenübertragung
  - synchrone Datenübertragung

- Steuerung der Datenübertragung geht vom Target und nicht vom Initiator aus

Asynchrone Übertragung:
- Busbreite stets 8 bit
- Transferrate von 5 MHz

Synchrone Übertragung:
- Busbreite muß abgesprochen werden
- Transferrate bis zu 80 MHz

Adressierung der Busteilnehmer
- Adresse des SCSI-Adapters wird durch Brücken (Jumper) fest auf der Hauptplatine eingestellt
- Fast-SCSI Adressen 7 ... 0
- Wide-SCSI Adressen 15 .. . 0

Busanforderung und –Zuteilung
- jeder Teilnehmer hat eindeutige Priorität
- höchste Priorität Adapter 7
- Prioritäten in 8-er-Gruppen absteigend sortiert: 7 ... 0 , 15 ... 8

Ablauf einer Befehlsausführung

8 Phasen der Befehlsausführung:

1	Bus Free	Freigabe des Busses
2	Arbitration	Zuteilung des Busses
3	(Re-)Selection	Selektion des Kommunikationspartners
4	Command	Übertragung des Befehls
5	Data in/out	Ein-/Ausgabe Daten
6	Status	Übermittlung des Zustands
7	Message in/out	Nachrichtenübertragung
8	Bus Free	Freigabe des Busses

Aufbau SCSI-Nachrichten
Nachrichtentypen:
- Ein-Byte-Nachrichten
- Zwei-Byte-Nachrichten
- Erweiterte Nachrichten

Ein-Byte-Nachrichten
- informieren über
  - Abschluß eines Befehls
  - Erkannte Paritätsfehler
  - Unterbrechung oder Abbruch des Befehls
  - Rücksetzen der Komponente
  - Löschen der Befehlswarteschlange

Zwei-Byte-Nachrichten
- Verwalten von Befehlswarteschlangen der LUNs

Erweiterte Nachrichten

- Übertragung von Parametern
- Festlegung Form der Übertragung (asynchron oder synchron)
- Festlegung Datenbusbreite (8 oder 16 bit)
- Nur bei synchroner Übertragung: Festlegung des Offsets

Aufbau Statusbyte

Codes:

$00	Good
$02	Check Condition
$04	Condition Met
$08	Busy
$10	Intermediate
$14	Intermediate Condition Met
$18	Reservation Conflict
$22	Command Terminated
$28	Queue Full

Aufbau der SCSI-Befehle

- Länge 6, 10 oder 12 Byte
- 1. Byte jedes Befehls enthält Gruppencode
- anschließend folgt 5-bit Befehlscode
- 2. Byte enthält Numer der LUN im Target
- 21-Bit-Adresse des ersten Datenblocks folgt

USB (Universal Serial Bus)

- Ermöglichen von Hinzufügen und Entfernen von Komponenten und Geräten ohne Neustart des Rechners im laufenden Betrieb
- Anschluß der Geräte über kostengünstige Schnittstelle mit billigen Steckern
- Übertragung läuft seriell über abgeschirmtes 4-Draht-Kabel
- USB 1.1: Standard-Übertragungsrate 12 Mbit/s; Kabel mit verdrillten Signalleitungen; Länge max. 5m
- Neuere USBs: hohe Übertragungsrate von 480 Mbit/s
- Anschluß von 127 Geräten möglich

Aufbau

- Baumstruktur mit Hubs als Zwischenknoten und angeschlossenen Geräten als „Blätter“
- Wurzelknoten (Root Hub) ist stets im Rechner eingebaut, der den Bus verwaltet
- Hubs können in den Geräten integriert oder eigene Komponenten sein
- Maximal 7 Schichten
- Maximale Länge: 5m
- Größte Entfernung 35 m zwischen Host und Gerät

Host: PC oder Workstation mit spezieller Hard- und Software
Hub:

- Spezielles Gerät zur baumförmigen Verzweigung der „Bus-Stränge
- 4-8 Downsteam-Ports

- erkennt selbständig, ob Daten mit langsamer (1,5 Mbit/s) oder Standardrate (12 Mbit/s) übertragen werden

Endgeräte:
- Funktionseinheiten
- Sind direkt oder über Hubs mit Host verbunden
- Bestehen aus mehreren logischen Endpunkten

Synchronisations- und Übertragunsverfahren
- Übertragung erfolgt in Form von Paketen
- Übertragung in Zeitrahmen mit konstanter Länge von 1 ms aufgeteilt
- Übertragungen im Halbduplex-Verfahren (von Host zu Funktionseinheit oder umgekehrt)
- Jeder Rahmen beginnt mit Start-Paket aus 10 bit langer Rahmennummer
- Pro Paket zusätzliches 8-bit-Synchronzeichen SYNC am Anfang

Adressierung der Busteilnehmer
- zweistufig durch 11-bit-Adresse
- höherwertige 7 Bits = Geräteadresse
- untere 4 Bits selektieren logischen Endpunkt im angesprochenen Gerät
- Adressen werden in Token-Paketen übertragen
- Nach Einschalten oder Rücksetzen des USB-Systems hat jedes Gerät Funktionsadresse 0
- Host vergibt eindeutige Nummern an alle Funktionseinheiten
- Adresse 0 für Initialisierung neuer Geräte reserviert

Busanforderung und –Zuteilung
- Zuteilung des Busses durch Polling-Verfahren
- Host fragt in gewissen Zeitabständen alle Teilnehmer nach Zugriffswünschen ab

Übertragungsarten
- isochrone Übertragung
- Übertragung von Interruptdaten
- Übertragung mit langsamer Geschwindigkeit
- Übertragung von Steuerdaten
- Bulk-Übertragung

Isochrone Übertragung
- in jedem Rahmen wird gleichbleibende Zeitdauer für Daten reserviert
- Vorteil: konstante Datenrate verhindert „untragbare“ Verzögerungen
- Nachteil: keine Fehlerkorrektur
- Übertragungsrage 12 Mbit/s
- Maximal 1023 Bytes pro Rahmen
- Maximale Busbreite 8,2 Mbit/s

Übertragung von Interruptdaten
- nicht periodisch, spontan auftretende Datenmengen
- Fehlererkennung und Wiederholungsversuche
- Nachteil: kein konkretes Zeitverhalten
- Übertragunsrate: 8, 16, 32 oder 64 Bytes pro Paket
- Maximal 19 x 64 Bytes
- Bandbreite 9,7 Mbit/s

Übertragung mit langsamer Geschwindigkeit
- maximal 1 Paket pro Rahmen

Übertragung von Steuerdaten

- wenigstens 10 % des Rahmens für Steuerdaten reserviert
- Pakete zu 8, 16, 32 oder 64 Bytes
- Maximal 13 x 64 Bytes
- Bandbreite 6,6 Mbit/s

Bulk-Übertragung

- Übertragung von Massendaten
- Fehlererkennung durch CRC-Prüfsummen (PID)
- Automatische Wiederholungsversuche
- Für Bulk-Übertragung steht nur nicht belegter Rest des Rahmens zur Verfügung
- Maximal 19 Pakete mit 64 Bytes
- Bandbreite 9,7 Mbit/s

Paketformate

- Start-of-Frame-Paket
- Token-Paket
- Datenpaket
- Handschake-Paket

Start-of-Frame-Paket

- Kennzeichnung eines neuen Zeitrahmens im Abstand von 1 ms
- Enthält
  – Pakettyp
  – 11 bit Rahmennummer
  – 5 bit Prüfzeichen

Token-Paket

- Übertragung der USB-Adresse durch Host
- Enthält
  – 5-bit-Prüfzeichen
  – Typ der Kommunikation und Datenrichtung
- Unterstützt folgende Typen
  – OUT Datenübertragung Host -> Gerät
  – IN Datenübertragung Gerät -> Host
  – SETUP Initialisierung des Kontrollendpunktes 0 im Gerät
  senden und Empfangen allgemeiner Datenpakete an bzw. vom USB-Gerät

Datenpaket

- Übertragung von bis zu 1023 Datenbytes
- LSB zuerst gesendet
- Datenwörter im Little-Endian-Format
- 16-bit-Prüfzeichen
- PID-Kennzeichen unterscheidet DATA0 und DATA1-Pakete
- Beginn der Übertragung stets mit DATA0
- Bei Übertragung zweier gleicher Pakete an Empfänger wird zweites Paket als Wiederholungspaket verworfen

Handshake-Paket

- 3 Möglichkeiten zur Quittung
  - mit PID-Kennung ACK : fehlerfreier Empfang
  - durch NAK: Senden und Empfangen nicht möglich
  - durch STALL: adressierter Endpunkt außer Betrieb

Ablauf der Paketübertragungen

Ausgabe von Datenblöcken (Bulk-Übertragung)

- Token OUT Paket
- DATA0 ... DATA1
- Quittung durch ACK oder Abbruch durch NAK/STALL

Einlesen von Datenpaketen (Bulk-Übertragung)

- Token IN Paket
- DATA0/DATA1 alternierend
- Quittung durch ACK
- Bei Fehler keine Quittung
- Bei nicht empfangsbereit NAK/STALL

Übertragung im isochronen Modus

- wie bei Bulk-Übertragung, jedoch ohne Quittungspakete

Einlesen von Interrupt-Daten

- Token IN
- DATA0-Paket
- Quittung mit ACK
- Bei Fehler keine Quittung
- Keine Interruptdatenlieferung beenden mit NAK/STALL

Übertragung Steuerinformationen

- Token SETUP
- DATA0
- Quittung mit ACK
- Erfolgloses SETUP: keine Quittung
- Übertragung zum Host:
    - Statusphase: Token OUT
    - DATA1 (evtl. leer)
    - Quittung mit ACK

Struktur USB-Software

- Kommunikation Host -> Gerät über logische Kanäle (Pipes) zwischen Anwendungssoftware und Interface des Geräts
- Pipes verbinden festgelegte Pufferbereiche im Arbeitsspeicher des Hosts mit Endpunkten eines Interfaces
- Pro Endgerät mindestens Pipe zu Endpunkt 0
- 2 Kommunikationsarten:
  - Nachrichten
    - Übertragen Daten mit festgelegter Struktur
    - Bidirektionale Übertragung möglich
  - Stromdaten
    - Übertragen Daten ohne Struktur
    - Isochrone, Bulk- oder Interrupt-Datenübertragung
    - Unidirektionaler FIFO-Transfer

- 3 Schichten
  - Businterface Schicht
    regelt physikalischen Kommunikationsfluß auf den USB-Leitungen
  - Geräteschicht
    Kommunikation Host mit Gerät
  - Funktionsschicht
    Datenaustausch Anwendungen -> USB-Funktionseinheiten

Hot Plug & Play

- Einfügen und Entfernen von Geräten während des laufenden Betriebs

Fire Wire

- 2 Versionen
  - Rückwandbus in Einschubgehäusen
  - Kabelverbindung zwischen eigenständigen Geräten

- Unterstützung von asynchroner und isochroner Übertragung

- Unterstützung von Datenraten von 100, 200 oder 400 Mbit/s

Topologie

- fast beliebige Netzstrukturen mit Verzweigungen und Verkettungen, jedoch keine geschlossenen Schleifen
- alle Verbindungen sind Punkt-zu-Punkt (Peer-to-Peer)
- Verzweigungen könen in den Endgeräten selbst vorgenommen werden
- Jeder Knoten kann bis zu 27 Ports zum Anschluß von weiteren Knoten besitzen
- Pro Bus bis zu 63 Geräte erlaubt
- Bis zu 1023 Busse über Brücken gekoppelt möglich
- Maximal 64449 Geräte anschließbar
- Bussystem ist selbstkonfigurierend
  - Nach Einschalten ermitteln die Knoten, welcher von ihnen die Funktion des Wurzelknotens wahrnimmt
- Bei kleineren Systemen findet automatische Optimierung des Protokoll-Zeitverhaltens statt
- Zwischen zwei Kommunikationspartnern dürfen bis zu 16 Kabelabschnitte liegen
- Low-Cost-Kabel
  - Max. 4,5 m Länge der Abschnitte
  - Max. Entfernung zwischen 2 Knoten: 72 m
  - Kabel besteht aus zwei einzeln abgeschirmten Signalleitungspaaren:
    - 1 Paar für Übertragung von Daten
    - zweites Paar für Strobesignal zur Bussteuerung
- POF-Kabel
  - Abschnitte zwischen 50 und 100 m lang
  - Maximale Entfernung zwischen 2 Knoten: 1600 m

Synchronisations- und Übertragungsverfahren

- Daten- und Strobesignal werden derart übertragen, daß in jedem Taktintervall genau eines der beiden Signale seinen Zustand ändert
- Empfänger kann aus den empfangenen Signalen durch Antivalenz-Verknüpfung (XOR) den Übertragungstakt regenerieren
- Signalübertragung in differentieller Form über je eines der verdrillten Signalleitungspaare
- Übertragung in Form von Paketen
- Zur Synchronisation wird Übertragung durch 8kHz-Impulse des Wurzelknotens in Zeitrahmen zu konstanter Länge von 125 µs aufgeteilt

Adressierung der Buskomponenten

- 64-bit-Fire-Wire-Adresse
- niederwertige 48 Adreßbits erlauben in jedem Gerät die Adressierung von bis zu 256 Tbytes in internen Speichermodulen oder Registersätzen

Busanforderung und –Zuteilung

- beliebiger Knoten kann zu jeder Zeit einen Zugriffswunsch an den Host stellen, indem er Anforderung an den „Vater“ schickt
- „Vater“ reicht Anforderung an eigenen „Vaterknoten“ weiter und blockiert Anforderungen der Söhne
- Host entscheidet nach fester Strategie, wer den Zugriff erhält

Asynchrone Übertragung

- von allen Fire Wire-Geräten unterstützt
- „sichere“ Übertragung mit Quittierung jedes Datenpakets und automatischer Wiederholung eines fehlerhaften Pakets
- für jede Paketübertragung muß sich Sender um Bus bewerben
- bestimmte Kennungen geben Paketanfang und Paketende an
- Unterstützung der Ausgabe von Blockierpaketen, die zwischenzeitlichen Zugriff durch andere Knoten auf die gesperrten Speicherbereiche im Empfänger der Nachricht verhindern
- Zwischen einzelnen Busaktionen Lücken mit definierter Länge eingefügt
- Es müssen (kürzere) Lücken zwischen den Paketen und ihren Quittungspaketen eingehalten werden
- Beschleunigung durch Weglassen der Lücken
- Austausch der Nachrichten in 2 Phasen
  - Anforderungsphase
  - Antwortphase

Isochrone Übertragung

- Übertragung in bis zu 63 Kanälen pro 125 µs-Zyklus
- Ausführen von Broadcast-Meldungen möglich
- Knoten kann auch Daten von mehreren Sendern empfangen
- Garantiert geforderte Bandbreite und gewisse Echtzeit-Übertragung
- Keine automatische Wiederholung eines fehlerhaften Pakets
- Besonders geeignet für Austausch von Multimedia-, Audio- und Video-Daten
- Zuteilung der Bandbreite erfolgt in Bandwidth Allocation Units
- Übertragungszeit eines 32-bit-Quadlets bei 1600 Mbit/s
- Bei Übertragungsrate von 400 Mbit/s bis zu 4000 Bytes pro Kanal in einem Zyklus übertragbar
- Beschleunigung durch Weglassen der Daten-Ende-Kennung und der Lücke zwischen den Kanälen

Struktur der FireWire-Software

4-Schichten:

- **Physikalische Schicht**
  - Stecker, Kabel und analoge Signale
  - Spezifiziert den Paketrahmen mit Codierung des Datenpräfix und die Daten-Ende-Kennung
  - Kümmert sich um den Buszugriff
  - Liefert alle Mechanismen zum Hot Plug & Play

- **Verbindungsschicht**
  - Übernimmt Aufgaben der Prüfsummen-Erzeugung und –Überprüfung
  - Interpretation der Paketkopf-Informationen
  - Erkennung der Empfängeradresse
  - Verwaltet asynchrone und isochrone Kommunikationsverbindungen
  - Veranlaßt bei asynchroner Übertragung senden der Quittungen(ACK)

- **Transaktionsschicht**
  - Nur bei asynchroner Übertragung
  - Bietet Dienste zur Ausführung der Steuer- und Transportoperationen

- **Treiberschicht**
  - Schnittstelle zu Programmen der Anwendungsschicht
  - Genormte Dienste für Anwendungsprogrammierer zur Kommunikation über den FireWire

Register und Speicher der Buskomponenten

- **Kernregister**
  - Speichern Informationen zur Unterstützung der Implementierung von (Treiber-) Software
  - Hardwarerealisierungen
  - Fehlererkennung und Fehlerbehandlung

- **PHY-Register**
  - Unterstützt Funktionen der physikalischen Schicht z.B. Adresse der Komponente, max. Übertragungsgeschwindigkeit, Anzahl der Ports

- **Configuration ROM**
  - Herstellerangaben
  - Informationen zu Komponenten
  - Treiber- und Diagnosesoftware-Informationen

Hot Plug & Play

- Businitialisierung
- Identifikation der Baumtopologie
- Komponentenidentifikation

Controller Area Network – CAN

- Bussystem, das zur Kopplung von Mikrocontrollern untereinander und mit ihren Peripheriekomponenten (Sender, Stellglieder) benutzt wird

Physikalische Eigenschaften

- hohe Übertragungsrate von 100 kbit/s bis zu 1 Mbit/s
- Übertragungsgeschwindigkeit durch Leistungsfähigkeit der Knoten bestimmt
- Bei Übertragungsrate von 1 Mbit/s Übertragungsweiten bis zu 40 m möglich
- Keine Beschränkung in Anzahl der anzuschließenden Knoten
- Selbstsynchronisierende Bitcodierung (im Knoten eigenständige Verlängerung oder Verkürzung der Bitlänge)
- Übertragungsbeginn eines Knotens unbekannt
- Entscheidung über den Buszugriff erfolgt bitweise
- Beginn des Sendevorgangs bei unbelegtem Bus durch jeden Knoten möglich
- Informationsübertragung in Form von kurzen Botschaften mit 0 bis 8 Bytes Länge

Übertragungsverfahren

- Systemknoten arbeiten vollständig unabhängig voneinander
- Vergeben von Prioritäten an Botschaften, die durch Identifikationsfeld angezeigt werden
- Für Nachrichten mit hoher Priorität garantierte Wartezeiten auf den Buszugriff
- Verschiedene Verfahren und Methoden zur Fehlererkennung eingesetzt

Anschluß des CAN-Busses

- Übertragung der Signale differentiell über ein Leitungspaar
- Verbindungskabel enthält 2 getrennte Masseleitungen
- Stecker ist Sub-D-Stecker mit 9 Anschlüssen

Struktur der CAN-Software

4 Schichten:

- Anwendungsschicht
- Objektschicht
- Übertragungsschicht
- Physikalische Schicht

Anwendungsschicht

- zu übertragende Daten als Botschaften bereitgestellt
- Botschaften mit Kennung versehen
- Durch Wahl der Kennung werden Botschaften mit Prioritäten versehen

Objektschicht

- Hauptaufgaben:
  - Botschaftenverwaltung
  - Zustandsverwaltung
- entscheidet, welche Botschaften momentan übertragen werden

Übertragungsschicht

- Abarbeitung des spezifizierten Protokolls
- Erzeugung eines Übertragungsrahmens
- Anforderung des Busses
- Evtl. Durch- und Weiterführung des Zugriffs
- Fehlererkennung und Fehleranzeige
- Fehlereindämmung

Physikalische Schicht

- Übertragung der einzelnen Bits einer Botschaft
- Definition des Übertragungsmediums und der Signalpegel

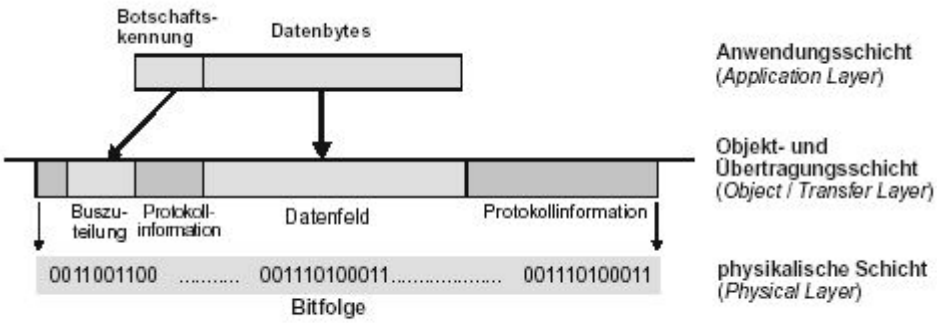


Bild 4.8-4: Umwandlung der Nachrichten durch die Protokollschichten

Buszuteilung

- Ankopplung der Knoten über die Signalleitungen CAN-H und CAN-L

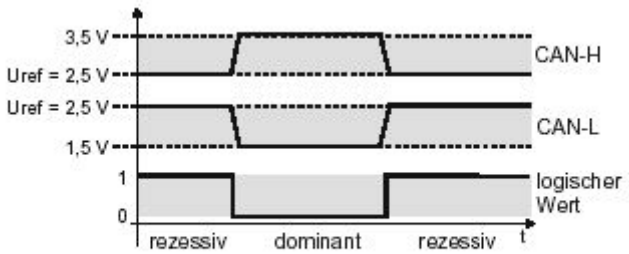


Bild 4.8-5: Ankopplung der Knoten an den CAN-Bus

- Zwei unterschiedliche Spannungszustände: dominanter Zustand
- Beide Ausgangssignale auf Uref: rezessiver Zustand
- Bitweise Arbitration

Botschaftenformate

4 Rahmentypen:

- Datenrahmen
- Daten-Anforderungsrahmen
- Fehlerrahmen
- Abstandsrahmen (Überlastrahmen, Rahmenpause)

Datenrahmen bzw. Daten-Anforderungsrahmen

- Startbit
- Identifikationsfeld
- Kontrollfeld
- Datenfeld
- Sicherungsfeld
- Quittungsfeld
- Übertragungsende

Fehlerrahmen

- Datenrahmen
- Fehlermarkierung
- Überlagerungsbits
- Fehler-Begrenzung

Überlastrahmen

- wie Fehlerrahmen
- Unterschied: Überlastmarkierung besteht aus 6 dominanten Bits

Rahmenpause

- Trennung der Rahmen durch Minimalabstand
- Besteht aus 3 rezessiven Bits

Sicherheit im CAN

Fehlererkennungsmöglichkeiten

- Busbeobachtung
- Rahmenformat-Überprüfung
- Zyklische Redundanzprüfung
- Bit-Stuffing

Fehlerbehandlung

- erneutes Aussenden von fehlerhaften Botschaften
- Interruptanforderungen an den Mikrocontroller
- Unterscheidung aktives und passives Fehlerverhalten
- Abschalten von Knoten nach Überschreiten einer oberen Grenze von Fehlern

CAN-Buscontroller

Komponenten des CAN-Buscontroller

- Steuereinheit
- Systembusschnittstelle
- Registersatz
- Bitstrom-Prozessor
- Sende-Steuereinheit
- Empfangs-Steuereinheit
- CRC-Checker/Generator
- Fehler-Behandlungslogik

Registersatz im CAN-Controller

- Systemregister
  - Steuerregister
  - Befehlsregister
  - Statusregister
  - Interruptregister
  - Akzeptanz
  - Bussynchronisation
  - Sendesteuerung

- Register zur Fehlerbehandlung

- Sendepuffer

- Empfangspuffer

Kurseinheit 5

Aufbau und Organisation des Arbeitsspeichers

Grundbegriffe

Arbeitsspeicher
- „Gedächtnis“ einer Rechenanlage
- Ort an dem Programme und Daten entweder permanent oder vorübergehend abgelegt werden

Speicherelement
- Schaltung, die genau einen binären Wert aufnehmen und für gewisse Zeit speichern kann

Speicherzelle
- umfaßt feste Anzahl von Speicherelementen, die durch eine einzige Adresse ausgewählt werden
- Einzelnes Element = Bit
- Enthält 8, 16, 32 oder 64 Bits

Speicherwort
- besteht aus der maximalen Anzahl von Speicherelementen, deren Inhalt in einem einzigen Buszyklus übertragen werden

Organisation des Arbeitsspeichers
- wird durch die Anzahl n der Speicherzellen und die Anzahl m der Speicherelemente definiert
- Form n x m bit

Kapazität
- Maß für die Informationsmenge, die maximal in einem Speicherbaustein untergebracht werden kann
- Einheit Bit
- Zahlenwert berechnet sich aus n * m

Arbeitsgeschwindigkeit
- Zugriffszeit
  - Maximale Zeitdauer, die vom Anlegen einer Adresse an den Baustein bis zur Ausgabe des gewünschten Datums an den Ausgängen vergeht
- Zykluszeit
  - Minimale Zeitdauer, die zwischen zwei hintereinander folgenden Aufschaltungen von Adressen an den Baustein vergehen muß
  - Idealfall: Zykluszeit = Zugriffszeit
  - Realität: Zykluszeit überschreitet Zugriffszeit um bis zu 80 %
    Grund: bei einigen Speicherarten wird die Information beim Auslesen zerstört und muß erst neu eingeschrieben werden

Klassifizierung von Halbleiterspeichern

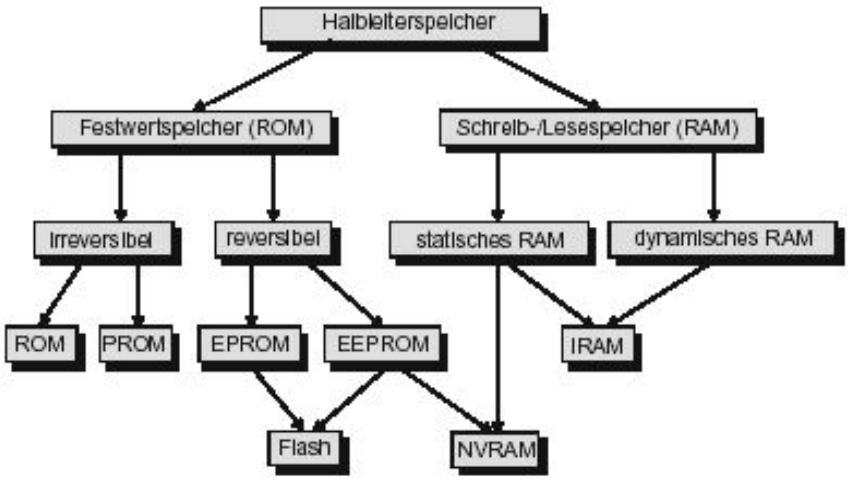


Bild 5.1-3: Die wichtigsten Typen der Halbleiterspeicher

Festwertspeicher

- Inhalt wird vom Mikroprozessor während des normalen Betriebs nur gelesen, nicht aber verändert
- Bezeichnung ROM (Read only Memory)
- Inhalt ist nicht flüchtig und bleibt nach dem Ausschalten des Rechners erhalten
- Dienen hauptsächlich zur Aufnahme von Programmen und Daten, die zur Funktionsfähigkeit des Systems dauernd und unverändert zur Verfügung stehen müssen:
  - Teile des Betriebssystems
  - Systemtabellen
  - Systemkonstanten
- in ihnen abgelegte Software = Firmware
- Programmierung geschieht außerhalb des Systems
- Zwei Typen von ROMs mit irreversibler Information:
  - Maskenprogrammierten Festwertspeicher (ROM)
  - Programmierbaren Festwertspeicher (PROM)
- Festwertspeicher mit reversibler Information:
  - (UV-)löschbare Festwertspeicher (EPROM)
  - elektrisch löschbare Festwertspeicher (EEPROM)

  Information kann zwar geändert werden, jedoch nicht während des laufenden Betriebs oder durch den Prozessor selbst

Schreib-/Lese-Speicher

- Bezeichnung Random Access Memory (RAM)
- Inhalt ist in der Regel flüchtig und geht nach dem Ausschalten des Rechners verloren
- Nur zum kurzzeitigen Aufbewahren von Programmen und Daten benutzt
- 2 Gruppen von RAMs:
  - statische RAM-Bausteine
    - speichern Informationen in Zellen, die aus FlipFlops bestehen
    - halten Information solange, bis sie durch einen erneuten Speichervorgang geändert wird
  - dynamische RAMs
    - Information wird als elektrische Ladung im Kondensator gespeichert
    - Lesen der Speicherzelle bedingt Entladen des Kondensators
    - Gelesener Wert muß anschließend wieder eingeschrieben werden
    - Ladung geht durch unvermeidbare Leckströme kontinuierlich verloren, daher muß in regelmäßigen Abständen aufgefrischt werden
    - Gesonderte Steuerlogik notwendig für das Auffrischen der Zellen

Nicht flüchtige RAMs

- Kombination aus Festwertspeicher und Schreib-/Lese-Speicher
- Jede Speicherzelle ist doppelt ausgelegt:
  - Einerseits als statisches RAM
  - Andererseits als EEPROM

- im normalen Betrieb wie ein RAM benutzt

- Steuerschaltung erlaubt, den gesamten RAM-Inhalt in das EEPROM zu übertragen

Prinzipieller Aufbau eines Speicherbausteins

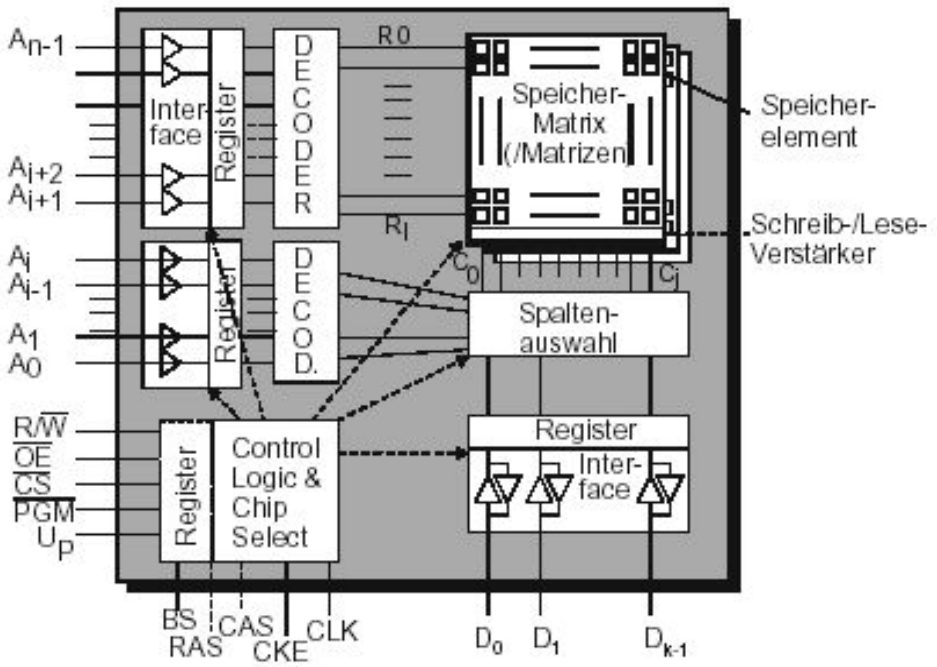


Bild 5.2-1: Prinzipieller Aufbau eines Speicherbausteins

- **Kern des Bausteins ist die Speichermatrix**
  - Speicherelemente sind reihen- und spaltenweise untergebracht
  - Jedes Element liegt im Schnittpunkt einer Zeilen-Auswahlleitung mit einer Spalten-Auswahlleitung
  - Zur minimierung der Anzahl der Auswahlleitungen wird Speichermatrix möglichst quadratisch gebaut
  - Bei hochintegrierten modernen Bausteinen wird Matrix in Teilmatrizen aufgeteilt

- **Adressierung der Speicherzellen über die Eingänge An-1 ... A0**
  - Adreßeingänge werden auf spezielle Schnittstellen-Schaltungen gegeben
  - In einfachster Form nur ausreichende Treiberleistung zur Verfügung gestellt
  - Einige arbeiten intern mit anderen Spannungspegeln als der µP
  - Dynamische RAMs haben Speicheradresse in zwei Teilen (höher- und niederwertige Bits)
  - Wegen rasanter Steigerung des Prozessortaktes in den letzten Jahren Übergang zu synchronen Speicherbausteinen

- **Steuerlogik und Bausteinauswahl**

  - Durch CS#-Signal auswahl des Bausteins aus der Menge aller Systembausteine

  - R/W#-Signal nur bei Speichrtypen, deren Inhalt geändert werden kann. Schaltet die Richtung der Treiber in der Datenbus-Schnittstelle um

  - Signal OE# aktiviert die Tristate-Ausgänge der Treiber in der Datenbus-Schnittstelle. Gibt die aus der Speichermatrix ausgelesenen Informationen auf den Datenbus

  - Signal PWM# muß aktiviert werden, wenn Baustein neu programmiert werden soll

Festwertspeicher

Maskenprogrammierbare ROMs

ROM-Speicherzellen

- 3 Möglichkeiten der Realisierung
  - mit Diode
  - mit bipolarem Transistor
  - mit MOS-Transistor

ROM-Bausteine

- in der Regel byteweise Organisation
- Spalten-Auswahllogik aus 8 n-auf-1-Multiplexern
- Werden durch CE#-Signal aktiviert

Programmierbare ROMs

PROM-Speicherzellen

- 2 Verfahren
  - Koppelelement aus einer feinen Widerstandsleiterbahn aus Nickel-Chrom-Legierung
    - erste Variante mit Diode
    - zweite Variante mit bipolarem Transistor
  - AIM-Verfahren
    - Umfunktionierung von hohen Stromimpulsen einer Halbleiterdiode in einen niederohmigen Widerstand
    - Erste Variante mit zwei gegeneinander geschalteten Dioden
    - Zweite Variante mit bipolaren Transistoren

PROM-Bausteine

- in der Regel byteweise organisiert
- interner Aufbau komplexer als bei ROM-Bausteinen
- Steuerung der Datenbustreiber durch CE# - und OE#-Signale
- Verknüpfung mit PGM#-Signal , das zur Programmierung aktiviert werden muß

EPROMs

EPROM-Speicherzellen

- Realisierung durch MOS-Transistor
- Dieser Transistor besitzt zweite Steuerelektrode

Programmierung

- Lesen
  - Auswahlleitung A wird auf +5 V gesetzt
  - Bitleitung B# wird auf +5 V Betriebsspannung gehalten, wenn Ladungsträger auf Floating Gate gespeichert sind
  - Leitung B# auf Massepotential, wenn keine Ladungsträger vorhanden sind

- Löschen
  - Transistor wird der Schaltung entnommen und in einem speziellen Löschgerät einer starken UV-Strahlung ausgesetzt
  - Löschen geschieht gemeinsam für alle Zellen des Bausteins

EPROM-Bausteine

- 3 Formen von Gehäusen
  - Plastik – oder Keramikgehäuse mit 2 Reihen von 28 bis 40 Anschlüssen an den Längsseiten
  - Plastik- oder Keramikgehäuse ohne Fenster
  - Quadratisches PLCC-Gehäuse

EEPROMs

EEPROM-Speicherzellen

- Schaltung enthält zusätzlich zum Speichertransistor noch einen Schalttransistor, der die Selektion der Zelle ermöglicht

Lesen

- Auswahlleitung A sowie Steuerelektrode des Speichertransistors werden auf Potential +5 V gelegt
- Schalttransistor wird leitend
- Speichertransistor leitet genau dann, wenn auf Floating Gate keine Ladungsträger gespeichert sind

Löschen

- Bitleitung B wird auf Massepotential gelegt und Auswahlleitung A sowie Steuerelektrode des Speichertransistors auf hohes Potential (z.B. +21 V)
- Beim Lesen einer gelöschten Zelle ergibt sich positives Potential auf Bitleitung B und logischer Wert „1“ wird ausgegeben

Programmierung

- EEPROM-Zelle muß vor Programmierung grundsätzlich gelöscht werden (es müssen Ladungsträger auf dem Floating Gate gespeichert sein)
- Auswahlleitung muß auf +21 V gehalten werden
- Steuerelektrode des Speichertransistors wird auf Massepotential 0 V gezogen
- Bitleitung B auf 0 führt zu gelesener logischen „1“
- Bitleitung B auf positivem Potential(+19 V) führt zu gelesener „0“

EEPROM-Bausteine

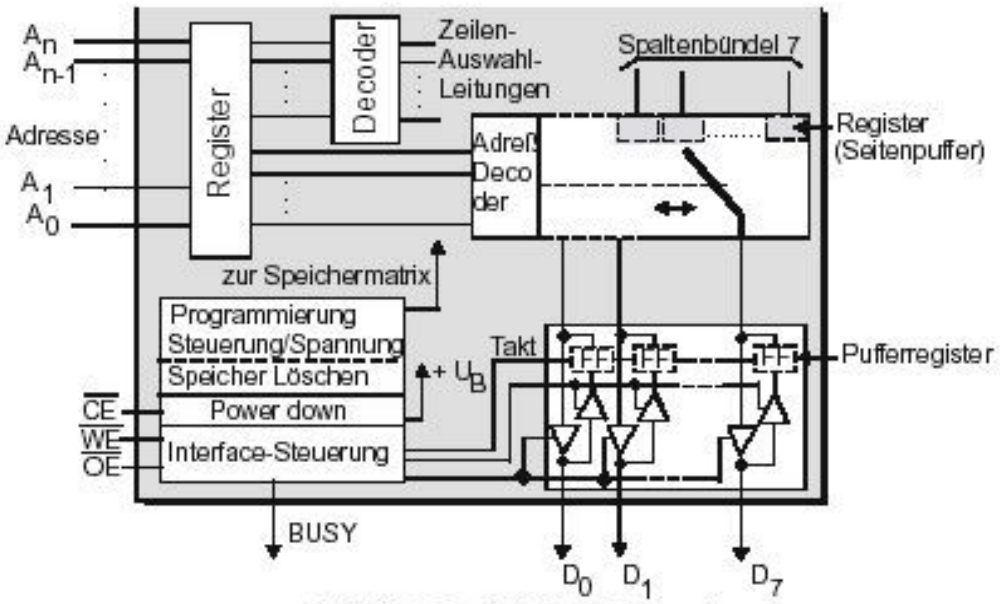


Bild 5.3-13: EEPROM-Baustein

- Unterschied zu Schreib-/Lese-Baustein: Vor dem Einschreiben eines neuen Datums muß Speicherzelle gelöscht werden
- Steuerlogik um einen Teil erweitert, der das Löschen einer einzelnen Speicherzelle oder aber auch der gesamten Speichermatrix erlaubt
- Löschen geschieht in der Regel automatisch vor dem Einschreiben eines neuen Wertes
- Unterschied zu EPROM: Spannung zum Löschen und Programmieren muß nicht extern zugeführt werden, sondern wird im Baustein selbst erzeugt
- Während des Einbrennens müssen Daten- und Adressinformationen im Pufferregister festgehalten werden

Nachteil EEPROM: begrenzte Lebensdauer, da bei jedem Löschen oder Programmieren einige Ladungsträger in der Isolierschicht zwischen Floating Gate und Drain-Zone hängen bleiben und so die Schwellspannung des Transistors allmählich immer weiter verschoben wird.

Vorteil gegenüber EPROM: gespeicherte Information liegt mindestens 10 Jahre lang unverändert vor.

FLASH-Speicher

FLASH-Speicherzellen

- vereinigen die Vorteile von EPROM und EEPROM in einem Speicherbaustein
- bei ihm können alle Speicherzellen auf einmal gelöscht werden
- Source-Anschlüsse über Schalter wahlweise mit Masse oder positiver Betriebsspannung verbunden
- Programmierung geschieht wie beim EPROM
- Löschen geschieht wie beim EEPROM

FLASH-Bausteine

- 2 Realisierungsformen
  - NOR-Flash-Bausteine
    - Anordnung wie bisher zwischen den Zeilen- und Spalten-Auswahlleitungen
  - NAND-Flash-Bausteine
    - Anordnung der Transistoren in Ketten

Schreib-/Lese-Speicher

Statische RAM-Speicher

Bipolare Speicherzellen

- mögliche Realisierung eines FlipFlops aus zwei rückgekoppelten Transistorschaltungen

Lesen

- beide Bitleitungen auf niedriges Potential gelegt

Schreiben

- es wird gezielt einer der beiden Transistoren in den gesperrten Zustand versetzt

MOS-Speicherzellen

- besteht aus kreuzweise rückgekoppelten CMOS-Invertern mit zwei Transistorpaaren
- Vorteil CMOS-Zelle: Nur im Umschaltzeitpunkt fließt ein nennenswerter Strom, daher wesentlich geringere Leistungsaufnahme

SRAM-Bausteine

Asynchrone SRAM-Bausteine
- arbeitet ohne Taktsignal
- Treiber in der Datenbus-Schnittstelle sind bidirektional ausgelegt
- Aktivierung der Treiber durch CE#-Signal
- Auswahl der Übertragungsrichtung durch Schreibsignal WE#

Synchrone SRAM-Bausteine
- arbeitet mit Taktsignal
- hauptsächlich zum Aufbau von Cache-Speichern benutzt
- Unterschied zu asynchronen Bausteinen:
  - Adresse, Ein- und Ausgabedaten und Steuersignale werden in Registern aufgefangen
  - Register werden durch das Taktsignal CLK getriggert
  - durch die Register zweistufige Fließbandverarbeitung (Pipelining)

NVRAM-Speicher

NVRAM-Speicherzellen
- nicht flüchtige RAMs
- größter Schaltungsaufwand von allen Speicherbausteinen
- besteht aus einer statischen 6-Transistor-RAM-Zelle und einer EEPROM-Zelle
- beide Zellen durch MOS-Transistor verbunden
- Vor Übertragung von Informationen muß EEPROM-Zelle gelöscht werden
- Im normalen Betrieb nur RAM-Zelle angesprochen
- Nur bei besonderen Situation Übertragung von RAM-Inhalt in EEPROM-Zelle (z.B. Spannungsausfall oder Einschalten des Gerätes)

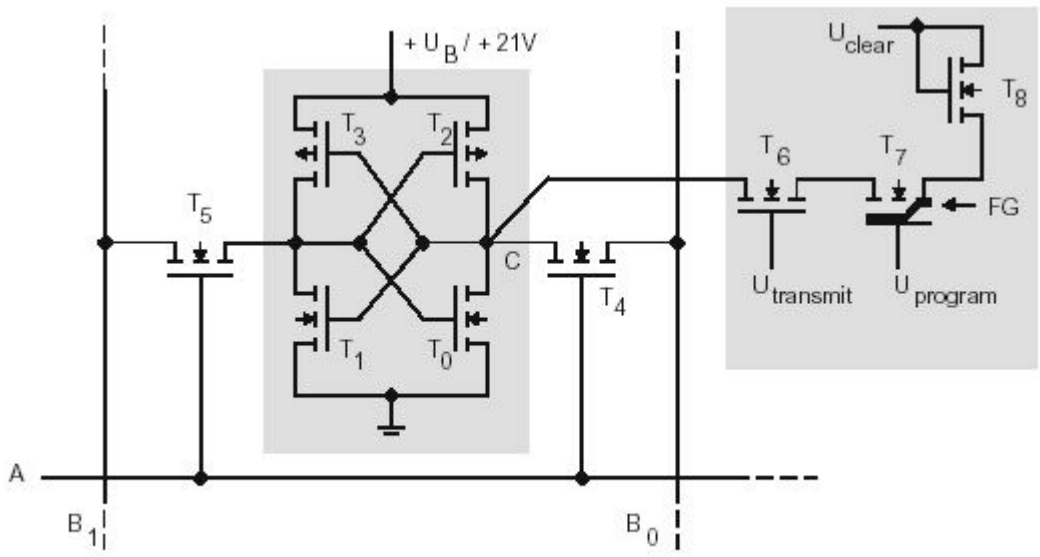


Bild 5.4-7: Aufbau einer NVRAM-Speicherzelle

Löschen
- Uclear wird auf 21 V gesetzt, alle anderen Eingänge auf 0 V
- Koppeltransistor sperrt und EEPROM-Zelle wird vom RAM abgetrennt

Speichern
- Wert ‚1' in RAM-Zelle, dann bleibt T7 selbstleitend im logischen Zustand ‚1'
- Wert ‚0' in RAM-Zelle, dann bildet T7 leitenden Kanal und logischer Zustand ‚0' wird ausgegeben

Auslesen
- vor dem Auslesen der EEPROM-Zelle in die RAM-Zelle, muß diese gelöscht werden

NVRAM-Bausteine
- verdoppelte Speichermatrix
- hinter Speichermatrix der statischen RAM-Zellen befindet sich als „Schattenspeicher" die Matrix der EEPROM-Zellen

Dynamische RAM-Speicher

Dynamische RAM-Bausteine
- ursprünglich meist bitweise organisiert
- neuere Bausteine mit Datenbreite bis zu 36 bit

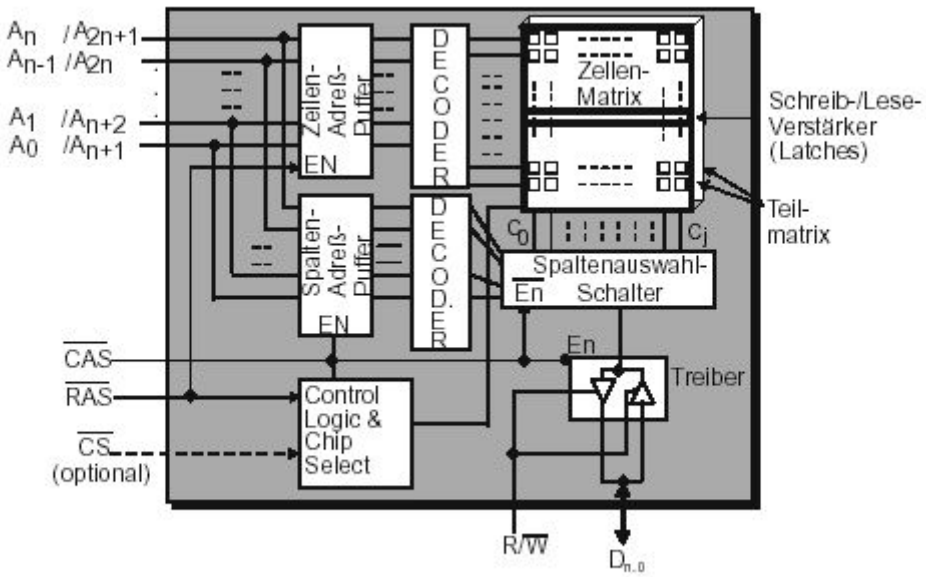


Bild 5.4-10: Dynamischer RAM-Baustein

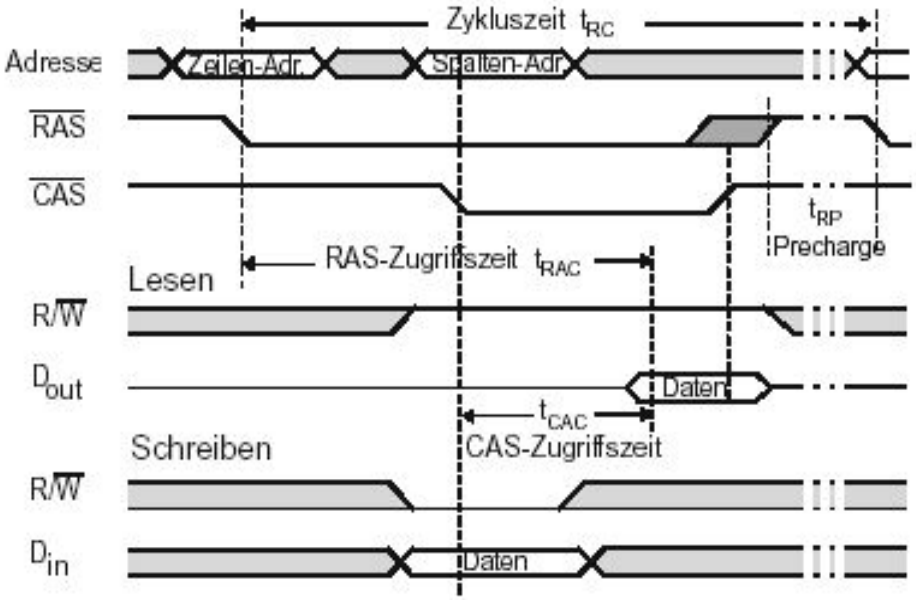


Bild 5.4-11: Adressierung eines dynamischen RAM-Bausteins

Lesen

- Datum erscheint eine gewisse Zeit nach der negativen Flanke von CAS# am Ausgang

Schreiben

- gleichzeitig mit der Spaltenadresse muß das einzuschreibende Datum an Dateneingang anliegen
- Übernahme in den Speicher geschieht mit der negativen Flanke des CAS#-Signals

Auffrischen dynamischer RAMs

- Inhalt jeder Speicherzelle muß periodisch nach einem Zeitintervall im ms-Bereich aufgefrischt werden
- Auffrischen geschieht Zeile für Zeile der Speichermatrix

Pseudo-statische RAMs

- dynamische RAMs, die integriert auf dem Chip die gesamte Auffrischschaltung enthalten
- wird heute nur noch äußerst selten verwendet, da mittlerweile jeder DRAM-Baustein eine solche Auffrischschaltung enthält

Weiterentwicklungen der DRAM-Bausteine

Bankadressierung

- Beschleunigung durch die Einteilung von DRAM-Bausteinen in verschiedene Speicherbänke
- Auswahl zwischen den Bänken geschieht durch ein geeignetes Adreßsignal

(Fast) Page Mode DRAMs

- enthält als zentrale Schaltung eine statische MOS-Speicherzelle (FlipFlop)
- Bitwert wird in ihr solange gespeichert, bis er durch einen neuen Wert überschrieben wird
- Faßt man die Speicherzellen aller Schreib-/Leseverstärker zusammen, erhält man ein Pufferregister, das eine ganze Zeile der Speichermatrix aufnehmen kann
- Erste DRAM-Bausteine, die zur Realisierung von externen Cache-Speichern eingesetzt wurden
- Nachteil: Verkürzung der CAS—Zeit führt unmittelbar zu einer Verkürzung der Dauer mit der die Daten auf dem Datenbus gültig sind

Hyper-Page Mode DRAMs (EDO-RAMs)

- Nachteil der FPM-RAMs durch einfache Schaltungserweiterung vermieden
- Ausgabedaten werden im Datenbusinterface in Auffangflipflops zwischengespeichert
- Vorteil EDO-RAM: CAS-Erholzeit und die Datenausgabe überlappen sich, dadurch Verkürzung der CAS-Zykluszeiten
- Vorteil nur bei Seiten-Lesezugriffen
- Nachteil EDO-RAM: zusätzliche Tristate-Treiber müssen in Speichern mit mehreren Speicherbänken dafür sorgen, daß beim Zugriff auf eine neue Speicherbank nicht die zuvor angesprochene Speicherbank noch den Datenbus belegt

Burst Hyper-Page Mode DRAMs

- Auffangflipflop in der Datenbusschnittstelle der EDO-RAMs wird durch ein getaktetes Register ersetzt
- Unterschiede zu EDO-RAM:
  - Speichercontroller muß nur die erste Spaltenadresse eines Datenblocks liefern
  - Fließbandausgabe von Spaltenadressen während eines Seitenzugriffs möglich

Synchrones DRAM (SDRAM)

- speichert die Adreßeingaben, Steuerinformation und Ein-/Ausgabedaten in Registern zwischen und synchronisiert den Zugriff durch den Bustakt
- unterstützen Blockmodus (Burst Mode) durch integrierten Burst-Spaltenzähler
- Verfügen über die gesamte Logik zum Auffrischen der DRAM-Zellen
- unterstützen die Fließbandausgabe von Block-Startadressen

DRAMs mit Zweiflanken-Übertragung

- Vorteil: Verdoppelung der Übertragungsrate ohne die Taktrate zu erhöhen
- Bei jeder Flanke des Bustaktes wird ein Datum übertragen
- Bezeichnung auch DDR-RAM
- Geschwindigkeit der Datenübertragung benötigt spezielle Komponenten zur Erzeugung von Strobe- und Synchronisiersignalen
- Takt – und Synchronisiersignale müssen zur Unterdrückung von Störungen differentiell übertragen werden

Rambus

- spezieller paralleler, synchroner Speicherbus, auf dem Adressen und Daten in kleine Blöcken transportiert und ausgewertet werden
- bis zu 32 Speicherbausteine anschließbar
- Steuerung durch speziellen Rambus-Controller:
  - Einzelbaustein
  - In PCI-Hostbrücke integriert
- Übertragungen erfolgen nach dem Zweiflankensystem

Direct RDRAM-Baustein

- Kapazität von 256 Mbit
- In 32 Speicherbänke aufgeteilt
- 32 Bausteine dieses Typs an einem Rambus-Kanal angeschlossen
- intern zwei 72 bit breite Datenpfade, die durch Multiplexer/Demultiplexer mit den beiden externen 9-bit-Datenbussen verbunden sind
- Eingaberegister zum Einschreiben der Daten

Aufbau und Funktion von DRAM-Controllern

- komplexe Schaltung , die das Multiplexen der Zeilen- und Spaltenadressen sowie die zeitgerechte Erzeugung der Steuersignale RAS und CAS und das Auffrischen der Zellen vornehmen muß

Auffrischlogik

- Zähler, der zyklisch alle Zeilenadressen erzeugt
- Länge von 8 bis 12 bit
- Auffrischen von n Zellen benötigt n*15,6 µs

Auffrischsteuerung

- RAS-only Refresh
- Nur die Zeilenadresse wird an den Baustein gelegt und druch negative Flanke des RAS#-Signals in das Register übernommen
- Ausgangstreiber des Bausteins nicht aktiviert
- CAS-before-RAS Refresh
- Speicherbaustein verfügt über integrierte Auffrischschaltung
- Zuerst wird CAS#-Signal aktiviert, erst dann RAS#-Signal

Auffrischvarianten

- verteiltes Auffrischen
- Auffrischzyklen gleichmäßig über das 2 bis 64 ms Intervall verteilt
- Prozessor über READY-Leitung zum Einfügen von Wartezyklen veranlaßt, bis Speicherzugriff wieder frei ist

- verdecktes Auffrischen
  - innerhalb eines Zeitintervalls wird genau eine Matrixzeile aufgefrischt
- Verzögerung bei diesem Verfahren geringer als bei allen anderen

- stoßweises Auffrischen
- im vorgegebenem Intervall werden alle Auffrischzyklen direkt hintereinander durchgeführt

- externes Auffrischen
- Auffrischzyklen werden vom Mikroprozessor zu beliebigen Zeitpunkten über den Eingang RFRQ# angefordert

Steuerlogik

- zentrale Steuerlogik übernimmt die Synchronisation aller Bausteinkomponenten
- Schalten der Multiplexer
- Steuerung der Auffrischschaltung
- Signalerzeugung zur Ansteuerung der DRAM-Bausteine

Speicherverschränkung

- bei älteren DRAM-Bausteinen ohne Seitenzugriff hängt die Auswahl der Adreßsignale von der Breite des Arbeitsspeichers sowie der Anzahl der Speicherbänke ab
- bei modernen DRAM-Bausteinen mit Seitenzugriff soll möglichst nur dann auf die andere Bank umgeschaltet werden, wenn ein Seitenwechsel nötig ist; Adreßsignale werden durch Anzahl der Speicherbänke, Seitengröße und Breite der Speicherwörter bestimmt

ECC-Logik

- erlaubt die Erkennung zweier Bitfehler bzw. die Korrektur eines Einzelbitfehlers
- jedes Speicherwort wird durch ein Codewort zur Fehlererkennung ergänzt
- Beim Einschreiben des Datums wird ein ECC-Code berechnet
- Beim Lesen wird ECC-Code erneut ermittelt und mit Code vom Einschreiben verglichen
- Bei Abweichungen der beiden Codes, wird ein Fehler erkannt
- Überprüfung des Speichers auf „schlummernde Fehler“ (Verfälschungen der Speicherzellen, auf die momentan nicht zugegriffen wird)
- ECC-Logik verfügt hierzu über weiteren Auffrischzähler

Programmierung des DRAM-Controllers

- Betriebsart in der Regel durch Hardware fest vorgegeben
- Programmierung nur nach dem Einschalten des Geräts notwendig
- bei Steuerung von asynchronen DRAMs nur wenige Parameter zu setzen
- kompliziertere Programmierung bei universellen Speichercontrollern z.B. in Brückenbausteinen implementiert
  - Parameter legen fest
    - Typ der eingesetzten DRAMs
    - Art der Auffrischung
    - Wartezyklen
    - Größe und Lage der Speicherbereiche
    - Größe der Seiten in den Speichermatrizen
    - Taktfrequenz der SDRAM-Bausteine
    - Wert der CAS-Verzögerung
    - Anzahl der Bänke in einem SDRAM-Speicherbereich
    - Aktivieren des Stromsparmodus

Organisation des Arbeitsspeichers

Speicherbelegungsplan

- legt fest, durch welche Art von Speicherbausteinen die einzelnen Bereiche des Arbeitsspeichers realisiert werden

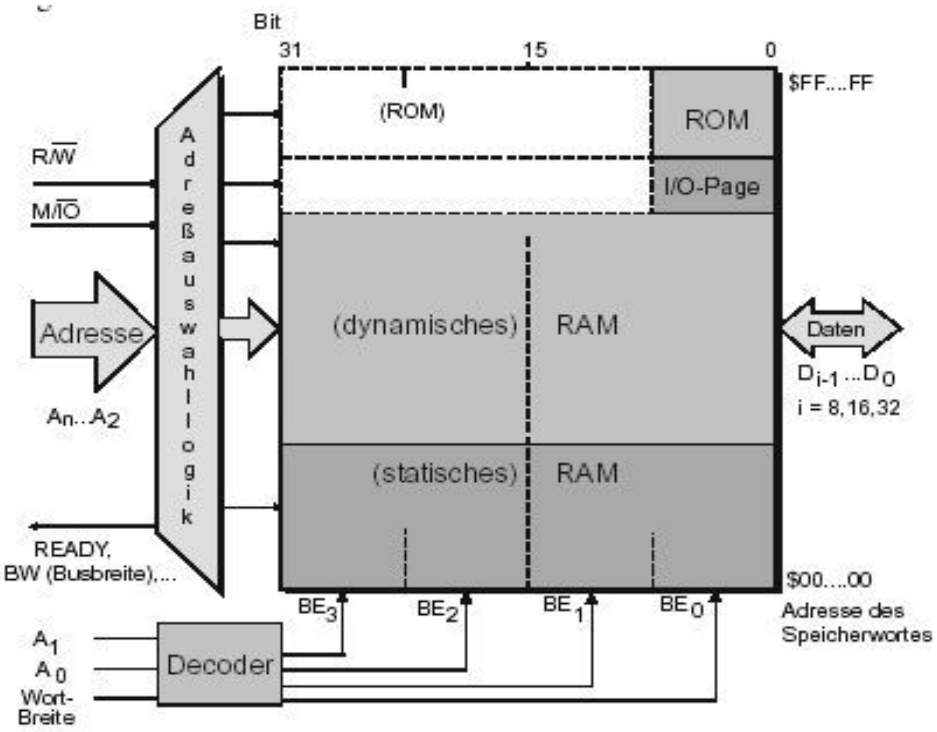


Bild 5.7-1: Beispiel für einen Speicherbelegungsplan

- größter Teil der Arbeitsspeicher aus relativ kostengünstigen langsamen dynamischen RAMs
- kleiner Teil aus teuren statischen RAMs

Adreßauswahl

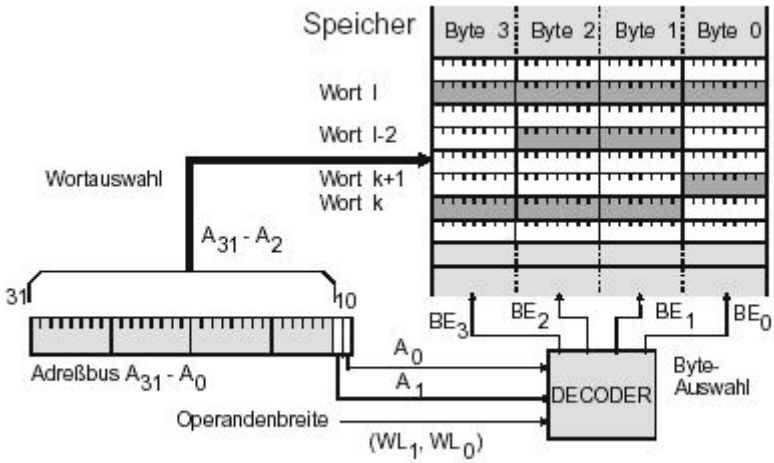


Bild 5.7-2: Zur Auswahl eines Bytes in einem Speicherwort

Kurseinheit 6

Systemsteuerbausteine

Nicht programmierbare Systemsteuerbausteine
- Taktgeneratoren
- Bussteuerbausteine (Bus Controller)
- Bus Arbiter
- Dynamic RAM Controller

Programmierbare Systemsteuerbausteine
- DMA Controller
- Cache Controller
- Speicherverwaltungsbausteine (MMU)
- Timer
- Echtzeit-Uhren
- Interrupt Controller

Schnittstellenbausteine (I/O-Controller)

Systembausteine

Bild einscannen von S. 214 im Buch

Adressierungsarten
- speicherbezogene Adressierung
  - Adreßblock zusammen mit allen Speicheradressen in einem gemeinsamen Adreßraum untergebracht
- isolierte Adressierung
  - zwei getrennte Adreßräume, zwischen denen er durch ein M/IO-Signal auswählt
  - in der Regel sind beide Adreßräume unterschiedlich groß

Aufbau der Systembausteine

Steuerung
- sorgt für die zeitgerechte Erzeugung der Signale, mit denen die internen Komponenten aktiviert und geschaltet werden
- bei komplexen Bausteinen ist Steuerung ein synchrones Schaltwerk, das vom µP mit Systemtakt versorgt wird
- Quelle und Ziel eines Datentransport sind die internen Register
  - Statusregister
    - Informationen, die den Zustand des Bausteins beschreiben, z.B. zuletzt ausgeführte Funktion, Betriebsart, ...
  - Steuerregister
    - Bestimmt Betriebsart, in der der Baustein arbeiten soll
  - Befehlsregister
    - Informationen, welche die Ausführung der aktuell gewünschten Operation verlangt

Ausführungseinheit
- der Teil des Bausteins, der die spezifischen Bausteinfunktionen zur Verfügung stellt

Schnittstelle zum Peripheriegerät
- Ausführungseinheiten besitzen eine mehr oder weniger große Anzahl von Ein- oder Ausgangsleitungen zur Kommunikation mit einem Peripheriegerät
  - Bündel bidirektionaler Leitungen zum Austausch paralleler Daten
  - Leitungen zur Übertragung serieller Signale
  - Leitungen zur Übertragung analoger Signale
  - Anzahl von Steuerleitungen zur Ausgabe von Steuerinformationen
  - Menge von Meldeleitungen zum Erhalten von Informationen über den Zustand der angeschlossenen Komponenten

Ein-/Ausgabe-Verfahren

- **interruptgesteuerte Ein-/Ausgabe**
  - Transfer eines Datums oder Datenblocks vom Schnittstellenbaustein über die INT-Leitung angefordert
  - Vorteil: Prozessor kann zwischen den einzelnen Datentransfers andere Operationen ausführen
  - Nachteil: zusätzlicher Zeitaufwand für die Programmumschaltung zur Interruptroutine
- **programmierte Ein-/Ausgabe**
  - Datenübertragung durch die Ausführung von Schreib-/Lese-Zugriffen auf das Datenregister des Baustein an geeigneten Stellen des Programms
  - Prozessor fragt in gewissen Abständen Zustand des Statusregisters ab
  - Busy-Waiting: Prozessor führt in einer Schleife ausschließlich die Abfrage des Statusregisters durch
- **DMA-Übertragung**
  - Vorteil: große Übertragungsrate
  - Prozessor von der Übertragung derjenigen Daten befreit, die für die Programmausführung augenblicklich keine Bedeutung haben

Synchronisation der Datenübertragung
- mehrere Alternativen möglich z.B. Hardwarelösung

Busarbiter

- Busmaster: Komponenten, die selbständig auf den Systembus zugreifen können z.B. Coprozessoren, DMA-Controller
- Anforderung des Systembusses durch das HOLD-Signal
- Quittung über HOLDA-Signal
- regelt den Zugriff zum Systembus, indem er die Tristate-Treiber zwischen lokalen Bus und Systembus an- oder abschaltet

Systembus-Zuteilungsverfahren

- unabhängige Anforderung
- Daisy-Chain-Verfahren

Prinzipieller Aufbau eines Busarbiters

- besteht im wesentlichen aus zwei synchronen Schaltwerken, die unabhängig voneinander vom Takt des lokalen Busses bzw. des Systembusses getrieben werden
- je nach Verfahren Zugriff zum Systembus über folgende Signale:
  - BRQ0# bzw. BGR# bei Independent Request Verfahren
  - BGR0# bzw. BGR# bei Daisy-Chain Verfahren
- bei Feststellung der Freigabe des Busses über BUSY#-Signal werden die Treiber des lokalen Busses über Enable aktiviert

Weitere spezielle Arbiter-Funktionen

- durch IOB-Eingang kann wahlweise zwischen der Behandlung von Zugriffen zum Speicher und zu den Ein-/Ausgabe-Schnittstellen unterschieden werden
- der ANYRQ-Eingang ermöglicht , die von der externen Prioritätsschaltung festgelegte Reihenfolge der µR für einzelne Arbiter außer Kraft zu setzen
- über CRQEN-Anschluß kann die beschriebene Aufhebung der Priorität für einzelne Arbiter zeitweise unwirksam gemacht werden

Interrupt-Controller

- 2 Verfahren
  - Polling Verfahren
    Prozessor fragt der Reihe nach in allen Bausteinen das Interrupt Flag ab
  - Daisy-Chain-Verfahren
    alle Interruptquellen miteinander verkettet
- programmierbarer Interrupt-Controller: Spezialbaustein zur Verwaltung mehrerer Interruptquellen

Prinzipieller Aufbau eines Interrupt-Controllers

- Baustein besitzt meist 8 Eingänge IR7 ... IR0, über die die Systemkomponenten ihre Unterbrechungswünsche anmelden können
- Unterbrechungswünsche werden im Interrupt Request Register zwischengespeichert
- Eingänge IR i werden als Interrupt-Channels bezeichnet
- Interrupt Mask Register kann während einer Programmausführung mit einem beliebigen 0-1 Muster geladen werden
- Von im Interrupt Request Register angezeigte Unterbrechungswünsche werden nur diejenigen an den Prioritätsdecoder weitergereicht, deren zugeordnetes Maskenregister eine ‚0' enthält.

- Prioritätsdecoder:
  - aus gleichzeitig vorliegenden, nicht maskierten Unterbrechungen, diejenige herauszusuchen, die die höchste Priorität hat
  - feststellen, ob augenblicklich durchgeführte Anforderung durch höher priorisierte Anforderung unterbrochen werden darf

Einsatz mehrerer Interrupt-Controller

- Daisy-Chaining
- Kaskadierung
  1 Controller übernimmt Rolle des Masters
  Interruptquellen oder weiterer Controller als Slave angeschlossen

Programmiermodell eines Interrupt-Controllers

- umfaßt Status-, Steuer- und Befehlsregister
- Prozessor kann über CS#-Signale jeden Controller gezielt ansprechen

Steuerregister

- es können verschiedene Arbeitsmodi des Controllers festgelegt werden

Befehlsregister

- dient dazu, während des Betriebes des Systems in Abhängigkeit vom augenblicklich ausgeführten Programm verschiedene Arbeitsweisen des Controllers aufzurufen

Statusregister

- Anforderungsregister
- Maskenregister
- Bedienungsregister
- 4. Statusregister

Direkter Speicherzugriff (Direct Memory Access - DMA)

- DMA-Controller führen Datenübertragung hardwaremäßig , also ohne sequentielle Abarbeitung eines im Arbeitsspeicher liegenden Programms aus
- Sind viel schneller als Ausführung durch µP
- Speicherzugriff einer Systemkomponente ohne Einsatz des µP

Prinzip des direkten Speicherzugriffs

- zu Beginn der Datenübertragung wird DMA-Controlle von CPU mit den benötigten Informationen zur Datenübertragung versorgt
- Datenübertragung wird selbständig vom DMA-Controller vorgenommen
- 2 verschiedene Verfahren
  - Explicit Addressing
    DMA-Baustein adressiert zunächst den Speicher und lädt Datum in ein internes Register
    Danach wird Peripherie-Baustein angesprochen
    Insgesamt nur 2 Buszugriffe für Datentransfer nötig
  - Implicit Addressing
    Umweg des Datums über internes Pufferregister im DMA-Baustein wird vermieden
    DMA-Controller legt nur die Adresse des Operanden an den Speicher
    Gleichzeitig wird Schnittstelle selektiert
    Für jeden Datentransfer nur ein Buszugriff notwendig

Aufbau eines DMA-Controllers

- besteht aus Steuerwerk und Ausführungseinheit

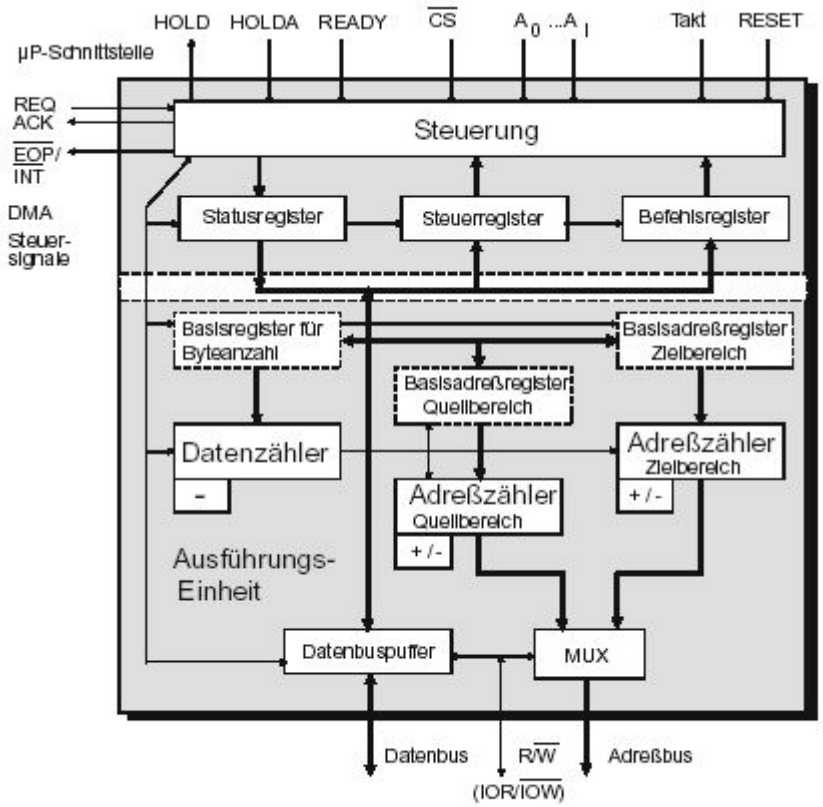


Bild 6.4-3: Prinzipieller Aufbau eines DMA-Controllers

Steuerwerk

- kontrolliert alle Komponenten des Bausteins
- erzeugt alle benötigten Takt- und Auswahlsignale
- über Ein- oder Ausgangsleitungen mit dem µP verbunden
- zusätzlich HOLD und HOLDA-Signale
- Zusätzlich noch DMA Request, DMA Acknowledge und End of Prozeß – Signale
- µP kann über REQ-Eingang Anforderung stellen, dazu wird EOP-Ausgang mit dem IRQ-Eingang des Prozessors verbunden
- Zur Programmierung und Überwachung seiner Arbeit dienen 3 Register des Steuerwerks:
  - Steuerregister
  - Befehlsregister
  - Statusregister

Ausführungseinheit

- besteht aus 3 Zählern, die vom µP vor dem Beginn der Datenübertragung geladen werden:
  - Datenzähler
  - Adreßzähler des Quellbereichs
  - Adreßzähler des Zielbereichs
- Datenzähler: enthält die Anzahl der zu übertragenden Daten
- Adreßzähler des Quellbereichs: enthält die Adresse des ersten zu transferierenden Datums
- Adreßzähler des Zielbereichs: enthalt die Adresse, unter der das erste übertragene Datum abgespeichert werden soll

DMA-Übertragungsarten

Einzel-Datentransfer (Cycle Stealing)

- es wird jeweils genau ein Datum übertragen
- danach wieder Freigabe des Systembusses für den Prozessor

Blocktransfer (Burst Mode)

- Controller überträgt ohne zwischenzeitliche Freigabe des Systembusses alle Daten eines Blockes
- Um Informationsverlust im DRAM-Speicher zu verhindern, kann diese Form der DMA-Übertragung durch den DRAM-Controller zum Auffrischen des Speichers unterbrochen werden

Transfer auf Anforderung (Demand Transfer Mode)

- Mittelstellung zwischen Cycle Stealing und Burst Mode
- Datenübertragung solange ohne Unterbrechung durch µP-Buszyklen ausgeführt, wie dies vom Requester gewünscht
- Übertragungswunsch teilt Requester durch REQ-Signal an den Contoller mit

Unterschiedliche Datenbreite in Target und Requester

- Adreßzähler müssen nach jedem Transfer um 1,2,4 oder sogar 8 erhöht werden, je nachdem, ob ein Byte, Wort , Doppelwort oder Quadword übertragen wurde
- Datenbreite Quellbereich größer als Zielbereich, muß Baustein das eingelesene Datum vor dem Weitertransport zerlegen
- Non-aligned-Datum muß erkannt werden
- Im Fly-by-Betrieb müssen Requester und Target gleiche Datenbusbreite haben

Die Register des Steuerwerks

- Statusregister
- Befehlsregister
- Steuerregister

Zeitgeber-/Zähler-Bausteine

- Erzeugung und Messung von Zeitfunktionen unterschiedlichster Art
- Zählen bestimmter Ereignisse
- Einsatzgebiete:
  - Impulsgenerator
  - Taktgenerator
  - Ereigniszähler
  - Zeit-Meßschaltung zur Ermittlung der Schwingungsdauer oder Impulsdauer externer digitaler Signale

Prinzipieller Aufbau eines Zeitgeber-/Zähler-Bausteins

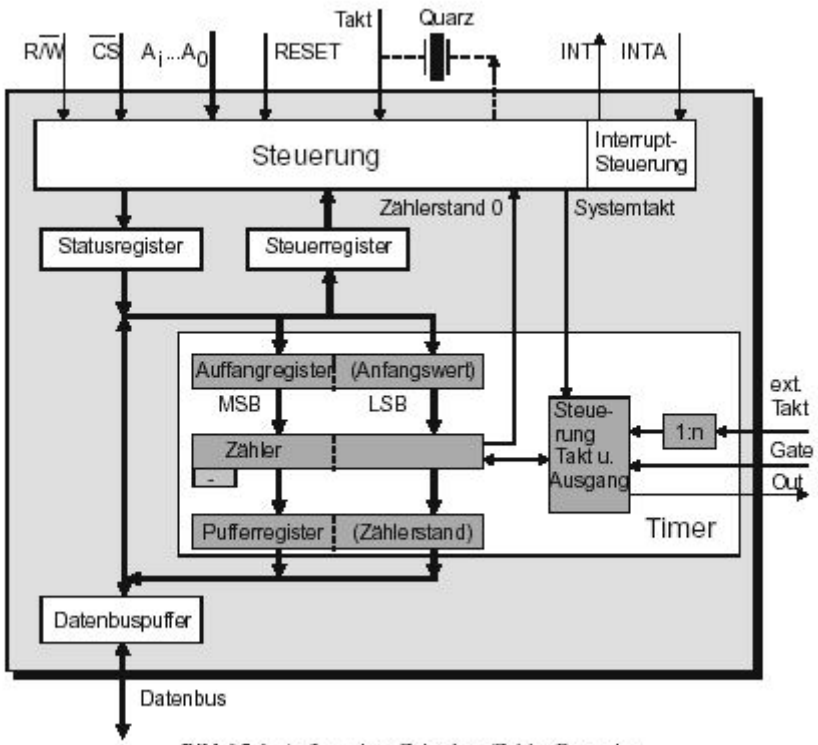


Bild 6.5-1: Aufbau eines Zeitgeber-/Zähler-Bausteins

- Ansteuerung sowohl mit Systemtakt als auch mit eigenem quarz-stabilisiertem Takt
- Hauptkomponente von einem synchronen Binärzähler gebildet
- Länge zwischen 16 und 32 Bit
- Zählvorgang vom Anfangswert bis zum Endwert 0 wird als Zählzyklus bezeichnet
- Anfangswert wird in ein Auffangregister übertragen und später daraus in den Zähler geladen (Initialisierung)
- Prozessor kann zu jedem Zeitpunkt Zählerstand abfragen

Zählmodi

- normaler 16-Bit Zählmodus
- 2 x 8 bit Zählmodus
  - niederwertige Bytes und höherwertige Bytes als zwei getrennte 8-bit-Zähler betrieben

- Zähler einiger Bausteine entweder als Dualzähler oder als BCD-Zähler in Betrieb

Programmiermodell

- ein Statusregister
- ein Steuerregister

Steuerregister

- um alle Arbeitsweisen zu berücksichtigen hat gezeichnetes Steuerregister 10 bits.
- Reale Bausteine besitzen 8 bits
- Bits des Steuerregister
  - Reset
  - Interrupt Enable
  - Output Enable
  - Mode
  - Counter Mode
  - Clock
  - Prescaled

Statusregister

- Interrupt Flag
- OUT-bit
- IN-bit
- Bits 4 – 0 z.T. Bits des Steuerregisters wiedergegeben

Timer- Funktionen

Initialisierung

- Hardware-Triggerung
- Software-Triggerung
- Initialisierung durch RE-Bit

Timer als Impulsgenerator

- 3 verschiedene Einzelimpulsformen
  - Monoflop (16-bit Mode) Single Shot
  - Monoflop (2 x 8 bit Mode)
  - Strobe
- Anwendungsbeispiel :
  - Watch Dog als Überwachung der zeitgerechten Abarbeitung eines Programmes durch den Prozessor

Timer als Taktgenerator
- Erzeugt periodisch digitale Zeitsignale
- 3 Möglichkeiten
  - beliebiges Impuls-Pausen-Verhältnis
  - Square-Wave
    gleich große Impuls- und Pausenlängen
  - Trigger bzw. Interrupt konstanter Rate
    kurzer negativer Impuls, wenn Zähler im Nullzustand ist

Timer als Ereigniszähler
- zu zählende Ereignisse treten sporadisch und aperiodisch auf
- Timer wird durch ansteigende Flanke des Gate-Signals nicht getriggert

Timer als Zeitmesser
- zu vermessendes Signal muß an den Steuereingang Gate angelegt werden
- Zählzyklus muß länger dauern als zu ermittelnde Zeit
- 2 Arten:
  - Frequenzvergleich
  - Impulsbreitenvergleich

Echtzeit-Uhren-Baustein HD146818
- universell einsetzbarer Zeitgeber-/Zählerbaustein
- Zusatzfunktionen
  - Erzeugung eines 1-Hz-Taktsignals
  - Umrechnung der Uhrzeit in Stunden, Minuten und Sekunden
  - Vergleich der aktuellen Uhrzeit mit einer einprogrammierten Weckzeit und Auslösen des Alarms
  - Fortschreibung des Datums

Kurseinheit 7

Schnittstellenbausteine

Bausteine für parallele Schnittstellen

- englische Bezeichnungen:
  - Peripheral Interface (Adapter)
  - Programmable Peripheral Interface
  - Parallel I/O Circuit
  - General Purpose I/O

Prinzipieller Aufbau

- erscheinen wie ein normales Register, das sowohl vom Prozessor als auch vom Peripherie-Gerät gelesen und beschrieben werden kann
- Realisierung einer parallelen Schnittstelle durch 8 Datenleitungen und 2 Steuerleitungen
- Programmierung für alle Datenleitungen gemeinsam und einheitlich
- Bei vielen Bausteinen Festlegung der Übertragungsrichtung individuell für jede einzelne Datenleitung
- Ausführungseinheit mit Daten- und Steuerleitungen wird als Port bezeichnet
- Reale Bausteine besitzen meistens 2 bis 3 unabhängig voneinander zu betreibende Ports
- Funktion der Steuerleitungen sehr vielfältig und kann durch Manipulation bestimmter Bits im Steuerregister festgelegt werden
- Port-Bausteine kommen ohne spezielles Befehlsregister aus, da die Programmierung darauf beschränkt ist, die Funktion der Steuerleitungen und die Richtung der Datenleitungen festzulegen

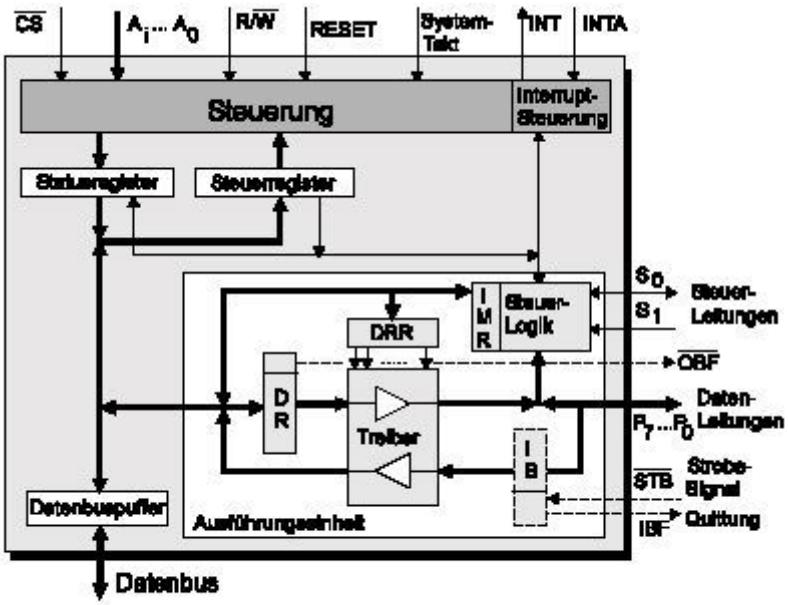


Bild 7.1-1: Baustein für parallele Schnittstellen

Aufbau der Ausführungseinheit

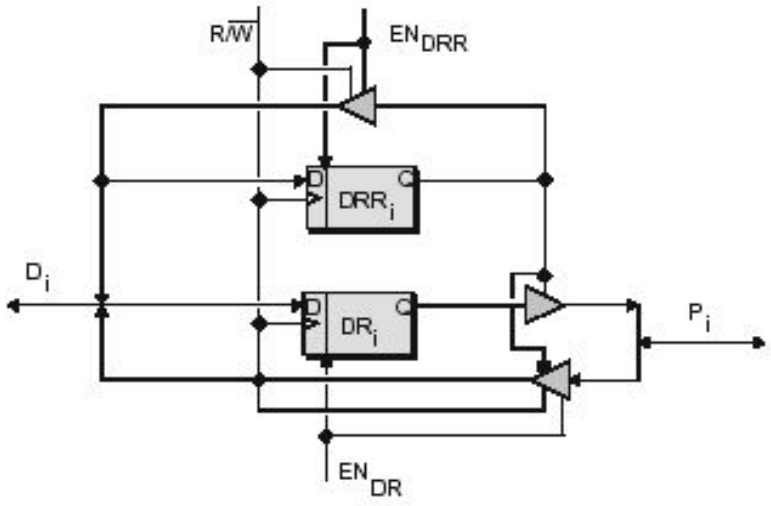


Bild 7.1-2: Übertragungslogik für ein einzelnes Datenbit

Datenausgabe

- auszugebende Daten werden vom Prozessor über den Datenbus in das Datenregister DR geschrieben
- Adreßdecoder erzeugt das Signal EN
- Nach kurzer Verzögerung erscheinen alle Datenbits an den Portleitungen die als Ausgänge definiert sind

Dateneingabe

- Aktivierung der Treiber der als Eingänge definierten Leitungen
- Leitungen ohne Zwischenspeicherung auf den Datenbus geschaltet
- Aktivierung der Treiber kann auch durch EN-Signal erfolgen
- Durch Handshake-Verfahren wird verhindert, dass das anliegende Datum während des Einlesens verändert wird

Variante

- einige Port-Bausteine erlauben auch den Einsatz der Portleitungen als Interrupteingänge
- besonders häufig bei den parallelen Schnittstellen von Mikrocontrollern

Fallstudie: PPI-Baustein 8255 von Intel

Modus 0

- Port PC wird in zwei Teilports PCH und PCL mit je 4 Leitungen unterteilt
- Jeder Port dient als paralleler Ein-/Ausgabeport, dessen Leitungen stets dieselbe Übertragungsrichtung besitzen
- Eingabedaten werden unmittel (ohne Zwischenspeicherung) in den Prozessor geladen

Modus 1

- Je 3 Leitungen des Ports PC werden den Ports PA und PB als Handshake-Leitungen zur Synchronisation der Datenübertragung zugeordnet
- Auf den 3 Leitungen werden die Signale STB#, ACK# und INT# übertragen
- Restliche 2 Leitungen von Port PC gehören zur Portgruppe A

Dateneingabe

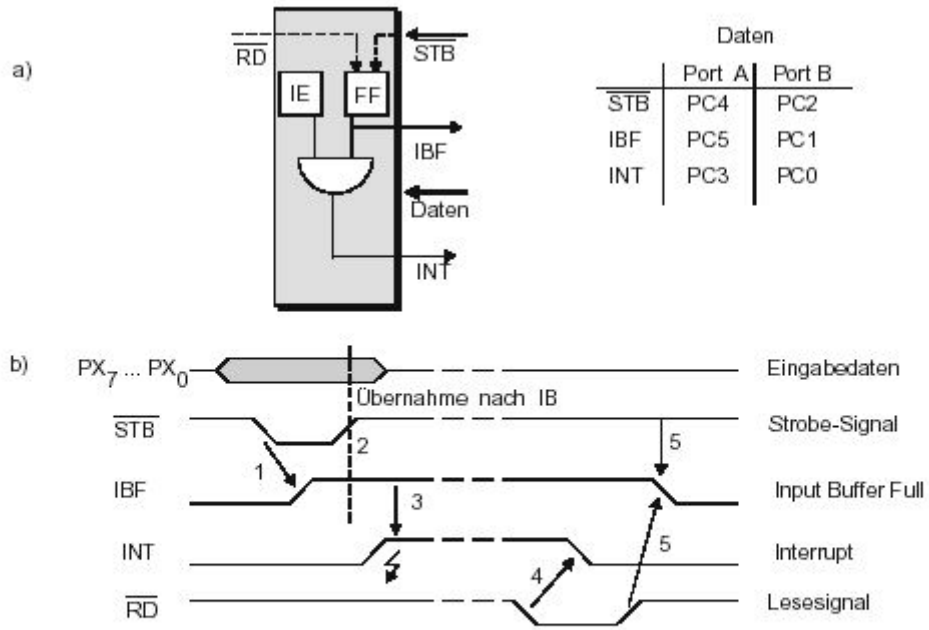


Bild 7.1-6: Synchronisation der Dateneingabe im Modus 1
a) Beschaltung der Handshake-Leitungen,
b) zeitlicher Ablauf der Übertragung

Datenausgabe

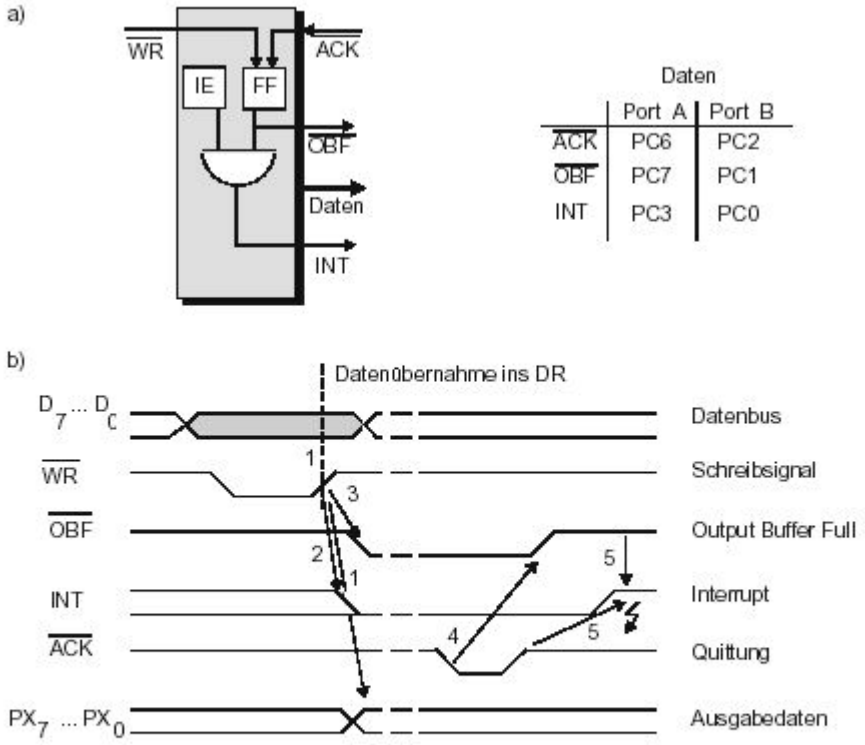


Bild 7.1-7: Synchronisation der Datenausgabe im Modus 1
a) Beschaltung der Handshake-Leitungen,
b) zeitlicher Ablauf der Übertragung

Statusregister im Modus 1

- ist in 2 Bitfelder eingeteilt, die den Portgruppen A und B zugeordnet sind
- Bedeutung der Bitfelder vom gewählten Modus abhängig

Modus 2

- von Port PC werden 3 Leitungen Port PB zugeordnet
- Port PB kann entweder als Eingangs- oder Ausgangsport geschaltet werden
- Die 3 Leitungen von Port PC können wahlweise als Ein-/Ausgabe-Leitungen oder als Handshake-Leitungen benutzt werden
- Restliche 5 Leitungen von Port PC werden als Handshake-Leitungen benutzt

Statusregister im Modus 2

- niederwertige 3 Bits sind wieder der Portgruppe B zugeordnet
- Belegung hängt vom gewählten Modus und der Übertragungsrichtung ab
- Statusbits der Gruppe A enthalten IE-bits und zeigen den Zustand der Ausgangsleitungen an

Centronics-Schnittstelle

- überträgt 8-bit-Zeichen parallel
- benutzt zur Synchronisation 3 Handshake-Leitungen
- Zeichen werden im ASCII-Code ausgetauscht
- Erste beiden Zeilen ASCII-Code = Steuerzeichen
- Zur Codierung aller Zeichen werden 8 bit benötigt

Nachteile von parallelen Schnittstellen

- maximale Länge der Übertragungsstrecke ist auf wenige Meter beschränkt
- große Anzahl von Übertragungs- und Synchronisierleitungen verursacht unverhältnismäßig hohe Kosten für den Übertragungsweg

Bausteine für asynchrone, serielle Schnittstellen

Vorteil serielle Übertragung
- Daten werden häufig nur über 2 bis 3 Leitungen übertragen

Probleme bei serieller Übertragung
- Bit-Synchronisation beim Empfänger
- Zeichen-Synchronisation, durch die der Beginn und das Ende eines Zeichens ermittelt werden
- Block-Synchronisation, die den Beginn und das Ende eines Datenblocks festlegt

Lösung der Probleme
- Bit-Synchronisation durch zusätzliche Leitung, auf der für Sender und Empfänger gemeinsamer Übertakt zur Verfügung gestellt wird
- Zeichen-Synchronisation: 2 zusätzliche Handshake-Leitungen
- Block-Synchronisation: besondere Steuerzeichen kennzeichnen Beginn und Ende des Blocks

Synchronisationsverfahren

- synchrone Übertragung
- asynchrone Übertragung

Synchrone Übertragung
- Synchronisation zwischen Sender und Empfänger nur einmal zu Beginn der Übertragung eines Datenblockes vorgenommen
- Daten werden danach in einem fest vorgegebenen zeitraster übertragen
- Einsatzgebiet: öffentliche Datennetze und schnelle LANs

Asynchrone Übertragung

- Sender und Empfänger synchronisieren für jedes einzelne Zeichen eines Datentransfers erneut
- Zwischen den Zeichen eines Blockes können beliebig große Zeitabstände liegen
- Einsatzgebiet: Verbindung zwischen Mikrorechner und langsamen Peripheriegeräten
- Vorteil asynchrone Übertragung: geringe Anforderungen an den Gleichlauf der Taktgeneratoren
- Nachteil asynchrone Übertragung: für jedes Zeichen wiederholte Synchronisation benötigt relativ viel Zeit

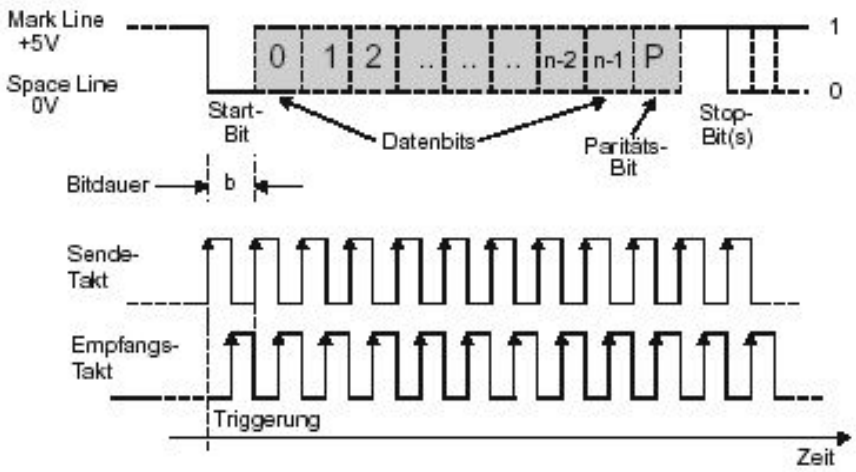


Bild 7.2-1: Zeichenrahmen bei der asynchronen Übertragung

Aufbau eines Bausteins für asynchrone Schnittstellen

- Unterteilung in 2 Einheiten
  - Sender (Transmitter)
  - Empfänger (Receiver)

Sender

- µP schreibt auszugebendes Datum in das Sende-Datenregister TDR des Senders
- vom TDR wird das Datum automatisch in das Sende-Schieberegister TSR übertragen
- Im TSR findet Umsetzung des Datums statt
- Zuerst wird Startbit übertragen
- Dann jedes Bit des Datums
- Parallel wird Parity-Bit ermittelt
- Nach Übertragen des Datums wird Parity-Bit übertragen
- Zum Schluß folgen 1 bis 2 Stoppbits

Empfänger

- arbeitet analog zum Sender
- Daten durchlaufen ihn in entgegensetzter Richtung wie den Sender
- Parallel zur Übertragung wird ein Parity-Bit ermittelt
- Am Ende der Übertragung wird Parity-Bit mit übertragenem Parity-Bit verglichen
- Bei Unterschieden wird Paritätsfehler an das Statusregister gemeldet und Unterbrechungsanforderung zum Prozessor ausgegeben
- Bei fehlenden Stoppbits wird ebenfalls Fehler an das Statusregister gemeldet = Rahmenfehler

Modem-Steuerung

- erzeugt Steuer- und Meldesignale, die der Synchronisation der Datenübertragung mit dem angeschlossenen Gerät dienen
- Modem-Steuersignale werden an den Bausteinaus- bzw. Eingängen invertiert
- Funktionen der Steuersignale
  - DTR# (Data Terminal Ready) Daten-End-Einrichtung betriebsbereit
  - DSR# (Data Set Ready) Daten-Übertragungs-Einrichtung betriebsbereit
  - RTS# (Request to Send) Sendeteil einschalten
  - CTS# (Clear to Send) Daten-Übertragungseinrichtung sendebereit
  - DCD# (Data Carrier Detect) Trägersignal erkannt

Die V.24 Schnittstelle

- dient zum Anschließen eines Modems an Geräte verschiedener Hersteller
- Signalpegel:
  - H-Pegel: 3 bis 15 V
  - L-Pegel: -15 bis –3 V
- Übertragung Datenbits: in negativer Logik
- Übertragung Steuerinformationen: in positiver Logik
- Aus Kostengründen z.T. auch heute noch direkte Ausführung der V.24-Schnittstelle zwischen µP und Peripheriegeräten

Programmiermodell des ACIA-Bausteins

- Steuerregister
- Befehlsregister
- Statusregister

Bausteine für synchrone, serielle Schnittstellen

- Übertragung von Datenblöcken nach festgelegtem Protokoll
  - Zeichenorientiert
  - Bitorientiert
- Übertragung von Einzeldaten ohne Protokoll

Zeichenorientierte Übertragung

Bild 7.3-1: Block der zeichenorientierten Übertragung

- SYNC Synchronisierzeichen
- STX Start of Text
- ETX End of Text
- BCC Prüfzeichen zur Erkennung von Übertragungsfehlern

Bausteine für zeichenorientierte Übertragung

- Unterschied zu asynchronen Bausteinen:
  - Datenregister von Sender und Empfänger sind durch FIFO-Registersatz ersetzt
  - Vorteil: größere Flexibilität beim Einschreiben und Auslesen der Übertragungsdaten
- Zeichenlänge von 5 bis 8 Bit
- Bei Zeichenlänge unter 8 Bit werden höherwertige Registerzellen auf ‚0' gesetzt
- STX, ETX und BCC werden wie normale Daten übertragen und vom Schnittstellenbaustein nicht ausgewertet
- Häufig Bausteine ohne Baudraten-Generator
- SYNC kann durch Bit im Steuerregister ausgeschaltet werden, dann erfolgt die Synchronisation durch den Empfängertakt RCLK

Programmiermodell zeichenorientierter Bausteine

- Statusregister
  - Neu ist UE-Bit (underrun Error)
    wird immer dann gesetzt, wenn während einer Blockübertragung das TSR-Schieberegister leer wird, im FIFO aber noch kein weiteres Zeichen zum Übertragen bereitsteht
  - Baustein überträgt dann ein Füllzeichen

- Steuerregister

Bitorientierte Übertragung

- Daten in einem Rahmen mit festem Format eingebunden
- Nur die Position der Bits bestimmt, ob Datum oder Steuerinformation übertragen wird
- Steuerinformationen werden von Schnittstellenbausteinen interpretiert
- Datenbits werden nicht ausgewertet
- am häufigsten eingesetztes Übertragungsverfahren: High Level Data Link Control (HDLC)

Bild 7.3-4: Übertragungsrahmen der HDLC-Prozedur

- Beginn und Endezeichen ‚01111110' – Flag
- Damit keine Verwechslung mit Daten passiert, wird nach jeweils 5- ‚1'-Bits eine ‚0' eingefügt (Bit-Stuffing) beim Übertragen
- Beim Empfänger wird diese ‚0' dann nach jeweils 5 ‚1'-Bits wieder gelöscht

Bausteine für bitorientierte Übertragung

- Sender- und Empfängerregister durch FIFO-Register ersetzt
- Zeichen zwischen 5 und 8 bit
- Bei Zeichen kürzer als 8 bit analog zur zeichenorientierten Übertragung
- Zeichen kürzer als 5 bit werden zu Blöcken mit mindestens 5 bit zusammengefaßt
- mit ‚FLAG-Erzeugung' bezeichnete Komponente gibt vor der Übertragung des ersten Zeichen die Anfangs-Zeichen auf den Datenausgang
- die ‚FLAG-Erkennung' vergleicht mit Flag-Kennzeichen und synchronisiert den Empfangstakt mit dem Sendetakt auf
- Flag wird nicht in Empfangsschieberegister übernommen

Programmiermodell bitorientierter Bausteine

- Statusregister
- Steuerregister

LAN-Controller

- häufig synchrone bitorientierte Datenübertragung in WAN oder LAN
- hochkomplexe Schaltungen als Bausteine

Synchrone, serielle Schnittstellen zur Bausteinkopplung

Kopplung über die SPORTs

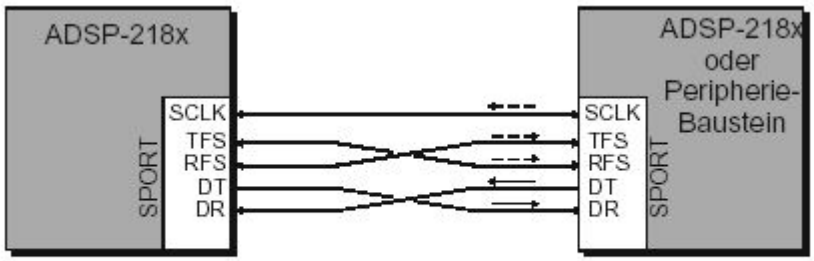


Bild 7.3-7: Kopplung über die seriellen Schnittstellen

- SCLK Taktsignal zur Steuerung der synchronen Übertragung
- DT Ausgabe der seriellen Daten
- DR Empfang der seriellen Daten
- TFS Synchronisiersignal, das Beginn des Datums oder Datenblocks anzeigt beim Aussenden
- RFS Synchronisiersignal, das Beginn des Datums anzeigt beim Empfangen

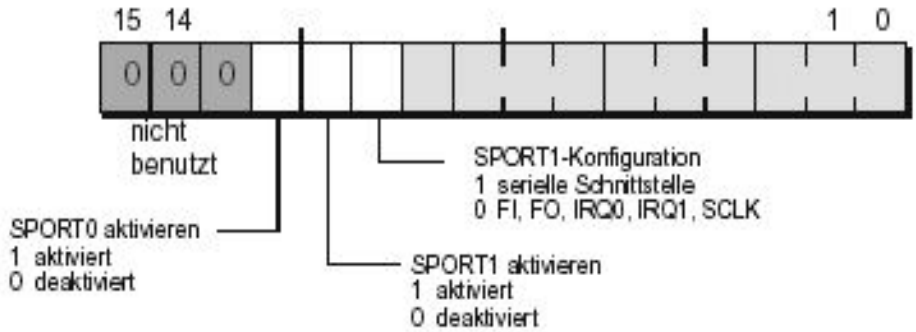


Bild 7.3-8: Das Systemsteuerregister

Interner Aufbau der SPORTs

- Frequenz des Schiebetaktes SCLK:
  - Frequenz von SCLK = 0,5*(Frequenz von CLKOUT)/(SCLKDIV +1)

- Frequenz von RFS Divide:
  - Frequenz von RFS = (Frequenz von SCLK)/(RFSDIV +1)

Synchronisationsverfahren

- **normale Rahmensynchronisation**
  - ein Zyklus vor dem Beginn der Übertragung wird für eine halbe Taktdauer ein Synchronisationssignal ausgegeben
  - Signal wird vom Kommunikationspartner während der fallenden Taktflanke erkannt
  - Beim Schreiben wird das Signal auf der Leitung TFS ausgegeben mit jeder positiven Flanke des Taktes wird danach ein Bit des Datums gesendet
  - Beim Lesen wird das Signal auf der Leitung RFS erzeugt mit jeder folgenden negativen Flanke wird danach ein Bit des Datums empfangen

- **Alternierende Rahmensynchronisation**
  - Synchronisiersignal wird zu Beginn der Übertragung ausgegeben und erst nach dem Ende der Übertragung des Datums zurückgenommen
  - Gleichzeitig mit der Aktivierung wird das erste Bit des Datums übertragen
  - Die Bits RFSW und TFSW müssen den Wert ‚1' haben

- **ungerahmte Datenübertragung**
  - wird eingesetzt, wenn große Blöcke von Daten möglichst schnell und ohne Pausen übertragen werden sollen
  - Sender und Empfänger müssen sich vor der Übertragung über die Länge des Blockes verständigen
  - Synchronisierungssignal nur vor dem Beginn der Übertragung eines Blocks
  - Erst Übertragung des nächsten Blocks wird wieder durch ein Rahmensignal angezeigt

Übertragung mit automatischer Pufferung

- im sendenden Kommunikationspartner kann Datenblock in einem Ringpuffer abgelegt werden und dann durch Hardware der seriellen Schnittstelle automatisch übertragen werden
- im empfangenden Partner kann der gesamte Datenblock automatisch in einem Ringpuffer gesammelt werden, bevor der DSP-Kern aufgefordert wird, den Block zu verarbeiten

Mehrkanal-Übertragung

- SPORT0 kann auch im Zeit-Multiplexverfahren betrieben werden
- Übertragung wird in mehrere Zeitkanäle eingeteilt, die sich zyklisch wiederholen
- Jeder Zyklus wird durch Synchronisiersignal eingerahmt
- Wählbar sind 24 oder 32 Kanäle
- Vor Beginn der Übertragung wird festgelegt, in welchen Kanälen der DSP senden und in welchen er empfangen darf
- Länge der Datenworte muß in allen beteiligten SPORTs auf denselben Wert eingestellt werden
- Zuordnung der Kanäle zu den einzelnen Kommunikationspartnern erfolgt durch die vier Register Multichannel Word Enables

Bausteine zur Analog/Digital- und Digital/Analog-Wandlung

Digital/Analog Wandlung

- Bereich der n-bit-Dualzahlen 0 <=D<= 2^n-1 auf einen vorgegebenen Spannungsbereich Umin <= U < Umax linear abbilden
- Spannungsbereich wird in 2^n gleich große Abschnitte unterteilt, deren Länge man LSB nennt

  LSB = (Umax – Umin)/ 2^n

- Jeder Dualzahl wird die Spannung

  U = D * LSB + Umin

  Zugewiesen

Pulsweiten-Modulation

- Zwischenstufe zur D/A-Wandlung
- Wird ohne weiteren Aufwand durch digitale Schaltungen realisiert
- Digitaler Ausgangswert D wird in eine periodische, digitale Zeitfunktion umgewandelt, deren Impulsdauer proportional zum Wert D ist

Analog/Digital Wandlung

- analoges Signal nimmt Werte aus dem Bereich Umin <= U <= Umax an
- Bereich in 2^n gleich große Abschnitte der Länge LSB unterteilt
- durch zweite Ordinaten-Skala wird Unterteilung in hexadezimale Zahlenwerte vorgenommen

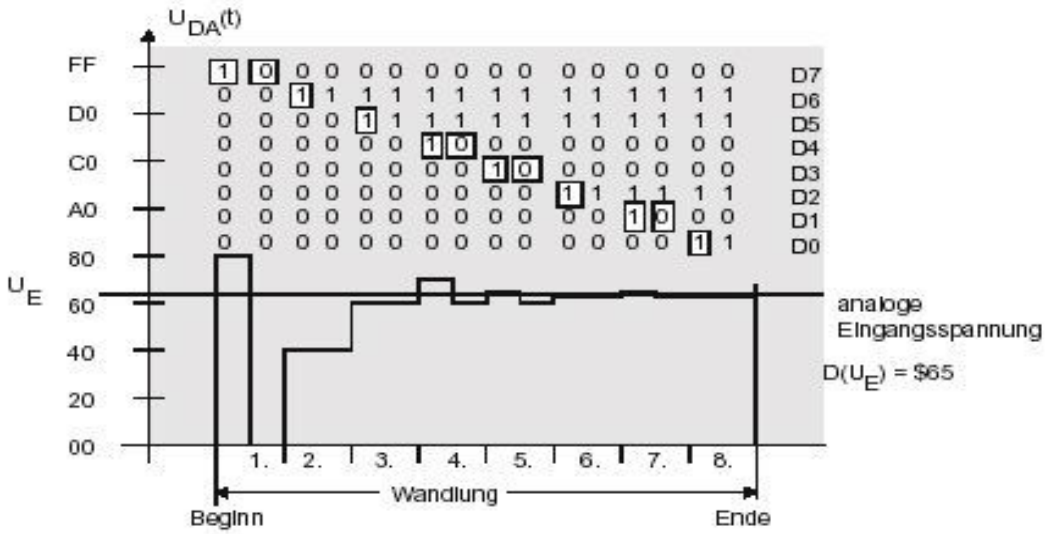


Bild 7.4-4: Beispiel zur sukzessiven Approximation