Prozessorarchitektur

Busse

M. Schölzel

Inhalt

- Grundprinzipien
  - Busarten und Verwendung (seriell/parallel, synchron/asynchron)
  - Busarbitration
  - Busprotokolle
- Beispiele
  - SPI
  - UART
  - PCI
  - PCI-Express

Busse

Ein Bus ist ein gemeinsamer elektrischer Weg zwischen mehreren Geräten.

Tanenbaum, Austin: Rechnerarchitektur

Einteilung von Bussen nach

- Funktion
  - Prozessor-interne Busse
  - Prozessor-externe Busse
- Taktschema
  - synchron
  - asynchron
- Übertragungsart
  - seriell/parallel

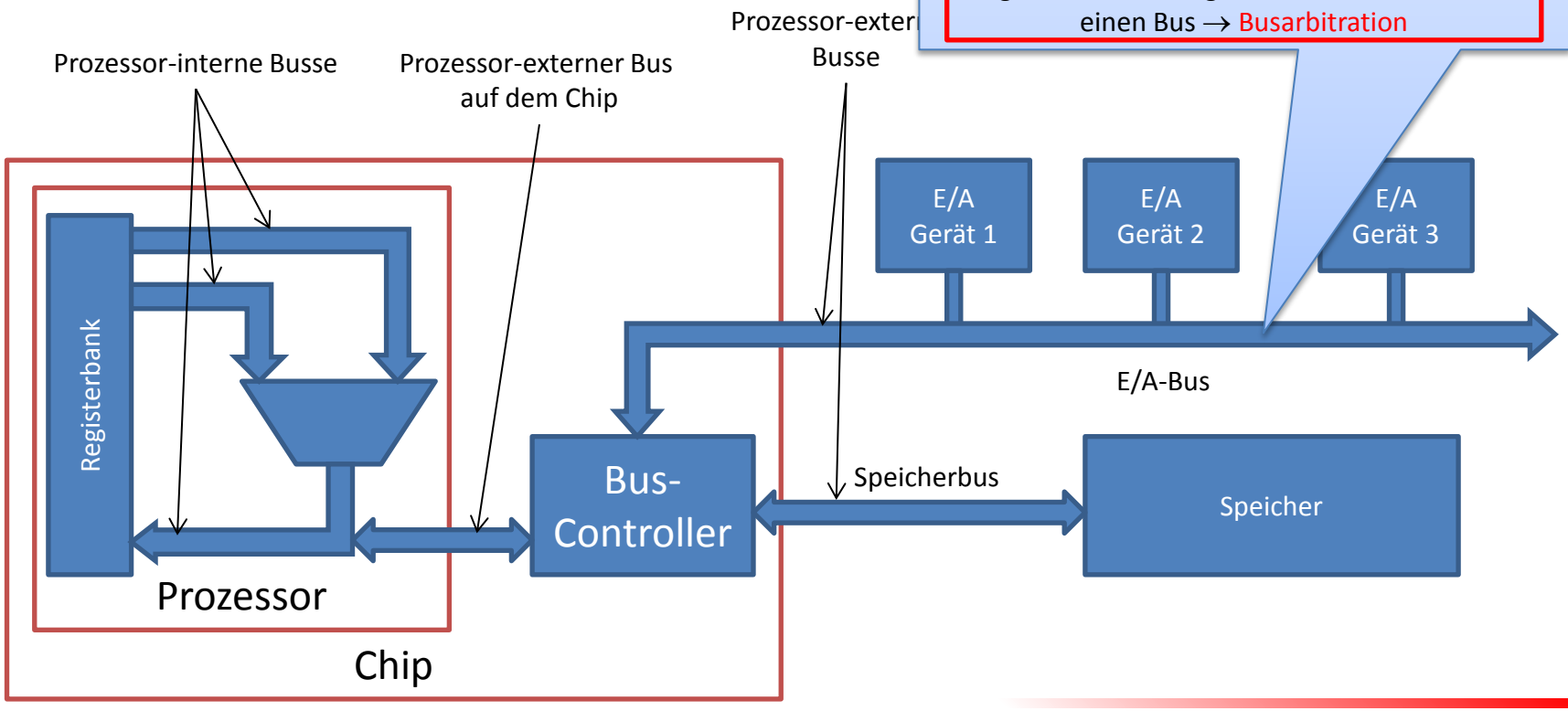
Funktionen von Bussen

- Prozessor-interne Busse
  - für Datenaustausch zwischen Komponenten im Prozessor (A
- Prozessor-externe Busse
  - Datenaustausch zwischen zwei Kommunikationspartnern (i

Organisation der Kommunikation zweier Partner → Busprotokoll regelt den Ablauf

Einer der Partner (Master) startet die Kommunikation; der andere Partner (Slave) reagiert

Organisation des Zugriffs mehrerer Master auf einen Bus → Busarbitration



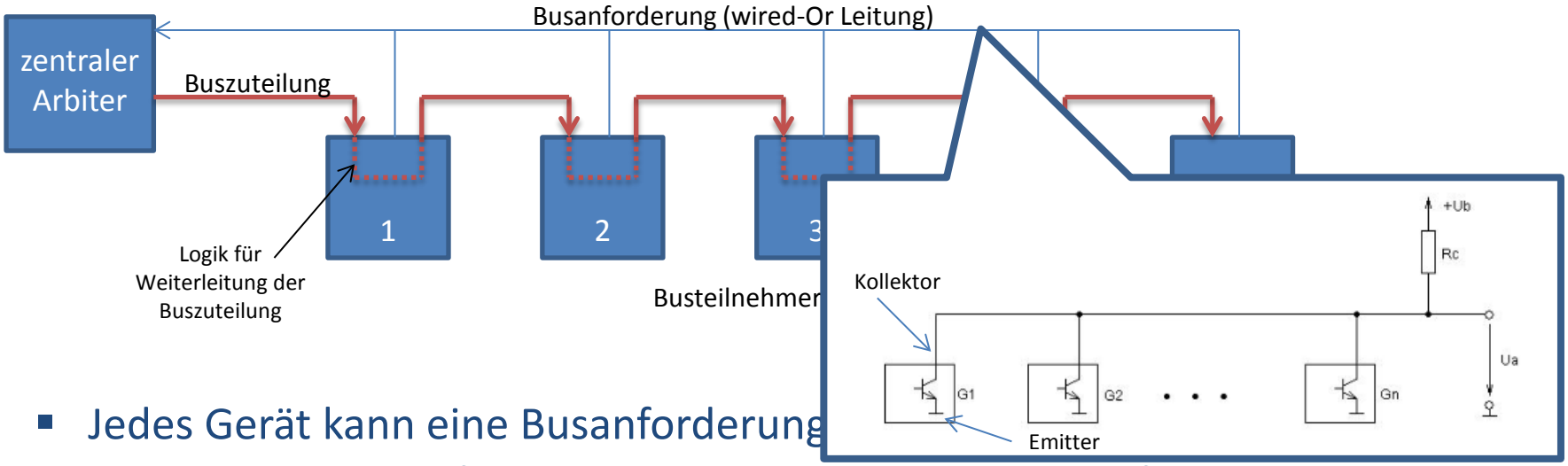
Busarbitration

- Problem: Verschiedene Geräte am Bus wollen gleichzeitig eine Kommunikation mit einem anderen Gerät starten
  - z.B.: mehrere E/A-Geräte wollen mit dem Speicher kommunizieren

- Busarbitration:
  - Auswahl eines Busteilnehmers, der als Master den Bus für seine Kommunikation nutzen darf
  - anschließend wird der Bus wieder frei gegeben
  - Priorisierung der Busteilnehmer sinnvoll
    - E/A-Geräte müssen Daten übermitteln, weil sie sonst verloren gehen (bekommen hohe Priorität, CPU oft niedrige Priorität)

- Methoden:
  - zentralisierte Mechanismen:
    - Es gibt einen Arbiter der eingehende Anforderungen priorisiert und den Zugriff zuteilt
  - dezentralisierte Mechanismen
    - es gibt keinen zentralisierten Arbiter; Zuteilung des Busses geschieht verteilt

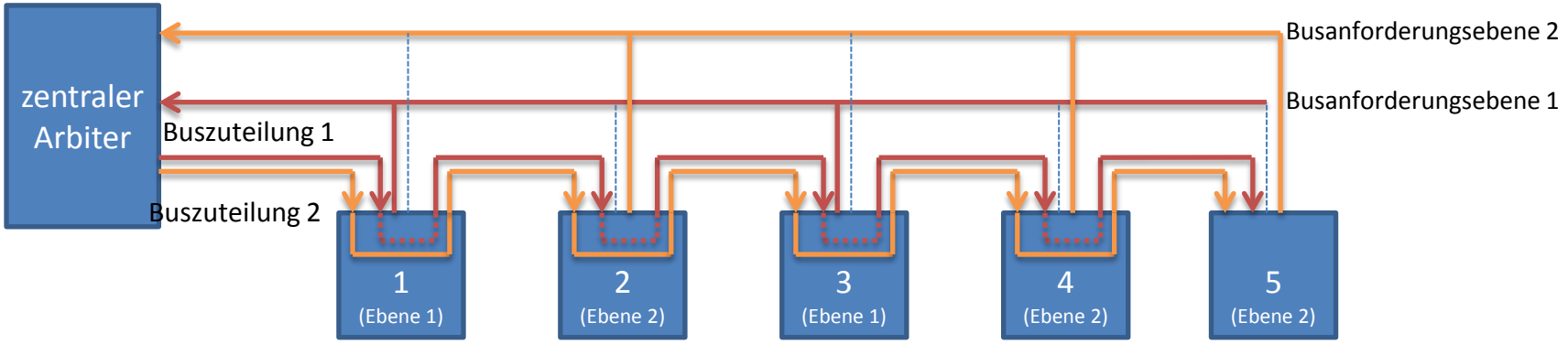
Zentrale Busarbitration mit Daisy-Chain (Reihenschaltung)



- Jedes Gerät kann eine Busanforderung
  - Arbiter kann nicht feststellen wie viele Geräte genau eine Anforderung gestellt haben

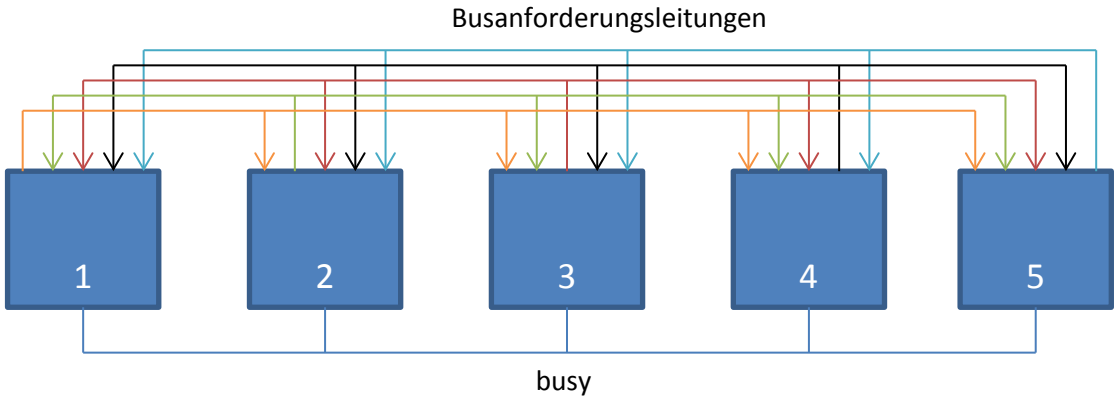
- Arbiter gibt eine Zuteilung aus (über Buszuteilungsleitung)
  - Jedes Geräte kann das Signal für Buszuteilung weiterleiten (wenn keine eigene Anforderung vorliegt) oder löschen (wenn eine eigene Anforderung vorliegt
  - durch die Reihenfolge der Geräte in der Daisy-Chain ist eine Priorisierung bei der Zuteilung festgelegt

Daisy-Chain mit mehreren Ebenen



- Es gibt mehrere Busanforderungs- und Buszuteilungsleitungen (Ebenen)
- Jedes Gerät ist einer Busanforderungsebene zugeordnet
- Arbiter gibt eine Zuteilung priorisiert an Ebenen aus
  - Innerhalb einer Ebene erfolgt die Zuteilung und Priorisierung wie in der Daisy-Chain
  - Typisch sind 4 bis 16 Ebenen

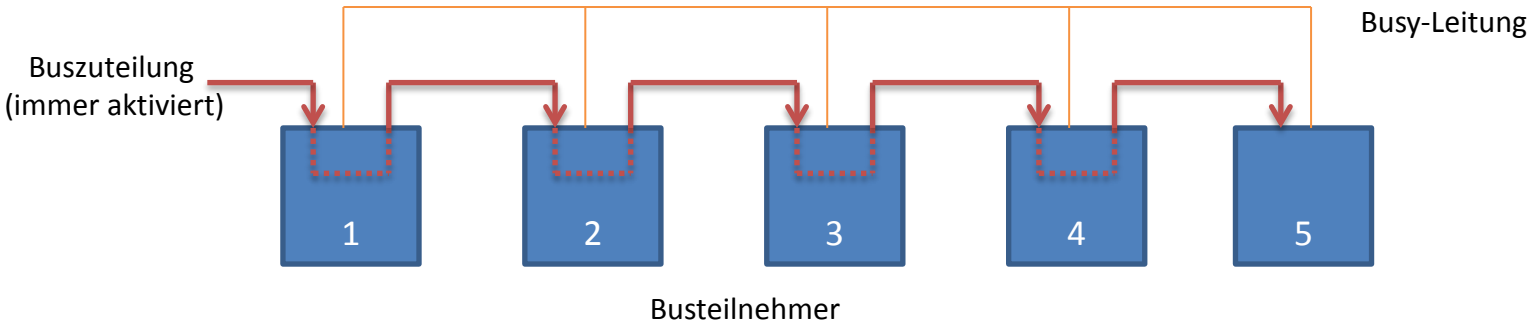
Dezentrale Busarbitration mit vollständiger Verdrahtung



- Jedem Gerät ist eine Anforderungsleitung zugeordnet
- Jedes Gerät überwacht die Busanforderungsleitungen der anderen Geräte
- Für einen Buszugriff
  - eigene Anforderungsleitung aktivieren
  - wenn es keine Anforderung eines Geräts mit höherer Priorität gibt und der Bus ist frei (nicht busy)
    - dann kann der Bus verwendet werden (busy setzen)
    - sonst warten
  - eigene Anforderung und busy wieder deaktivieren

- Anzahl Leitungen wächst mit der Anzahl der Geräte linear

Dezentrale Busarbitration mit Daisy-Chain



- ist der Bus nicht benutzt, dann
  - wird Buszuteilungsleitung von allen Geräten durchgeleitet
  - Busy-Leitung ist deaktiviert

- Damit Gerät *i* Busmaster werden kann, muss
  - Busy-Leitung deaktiviert sein (laufende Kommunikation darf nicht unterbrochen werden) und
  - Buszuteilung für *i* aktiviert sein
    - ausgehende Buszuteilung wird von i deaktiviert
    - i wird Master
    - i aktiviert Busy-Leitung und ausgehende Buszuteilung

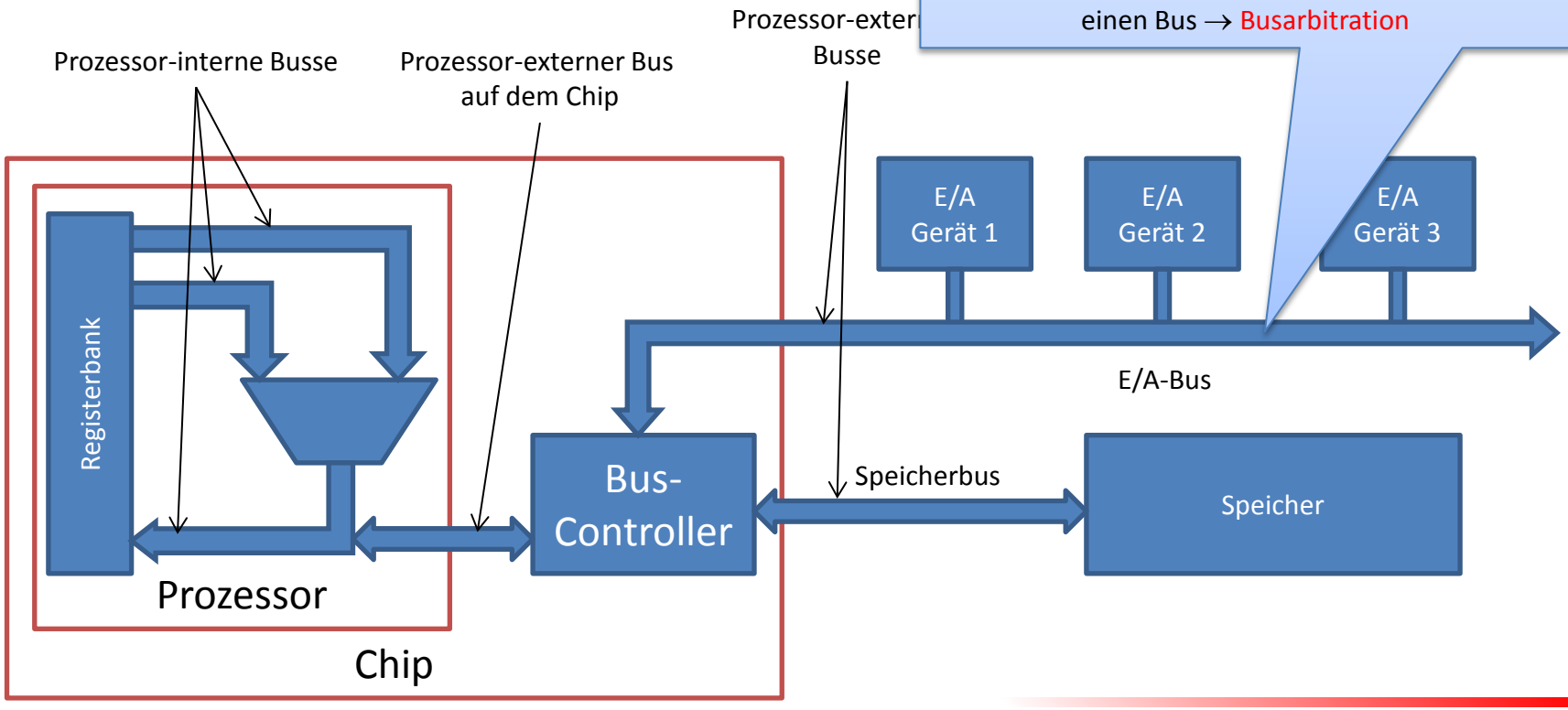
Funktionen von Bussen

- Prozessor-interne Busse
  - für Datenaustausch zwischen Komponenten im Prozessor (A
- Prozessor-externe Busse
  - Datenaustausch zwischen zwei Kommunikationspartnern (i

Organisation der Kommunikation zweier Partner → Busprotokoll regelt den Ablauf

Einer der Partner (Master) startet die Kommunikation; der andere Partner (Slave) reagiert

Organisation des Zugriffs mehrerer Master auf einen Bus → Busarbitration



Buszyklen

Ein Buszyklus ist eine festgelegte Abfolge von Signalen auf den Busleitungen, um eine Transaktion zwischen dem Master und dem Slave durchzuführen.

Ein Busprotokoll spezifiziert die verfügbaren Buszyklen.

Typische Transaktionen:

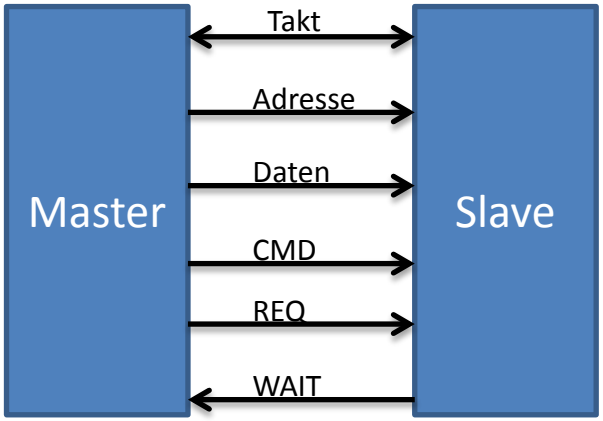
- Leseanforderung
  - Master übermittelt eine Adresse an den Slave und erwartet vom Slave das gelesene Datum von dieser Adresse
- Schreibanforderung des Masters an den Slave
  - Master übermittelt eine Adresse und ein Datum an den Slave und erwartet, dass der Slave das Datum an diese Adresse schreibt
- Blockübertragungen
  - Lese- oder Schreibanforderungen an aufeinanderfolgende Adressen
- Read-Modify-Write:
  - Master kann ein Datum (speziell im Speicher) testen und setzen ohne den Bus dazwischen freizugeben
  - Erforderlich für atomares test-and-set in Multiprozessorsystemen
- Signalisierung von Interrupts

Synchroner Bus

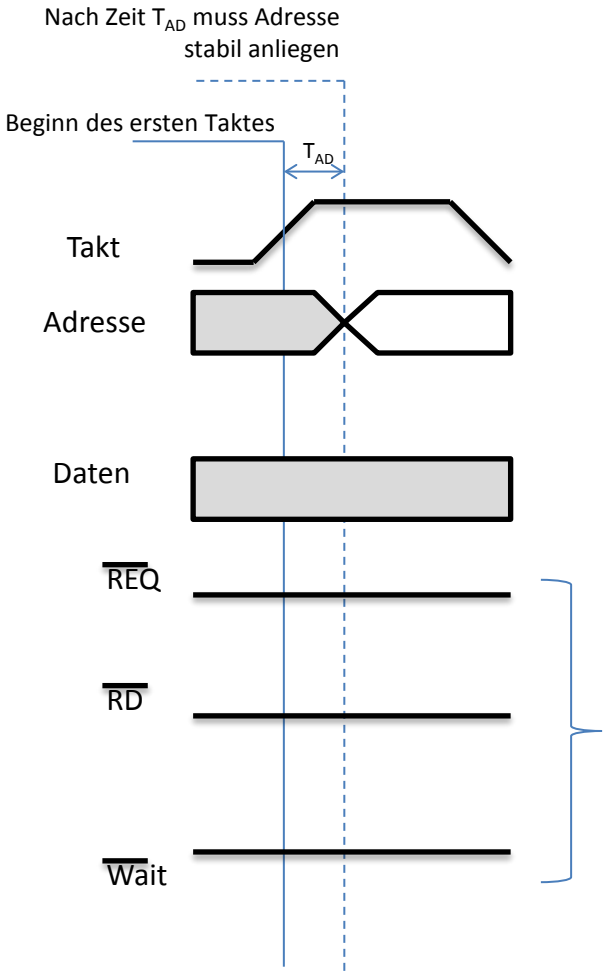
- Gemeinsame Taktleitung für alle Busteilnehmer

- Weitere Leitungen:
  - Adresse: Adresse von der/an die der Slave Daten sendet/schreibt
  - Daten: gelesene/zu schreibende Daten
  - CMD: Kommando (1 = Master will Daten lesen; 0 = Master will Daten schreiben)
  - REQ: Slave auswählen
  - Wait: Slave kann anzeigen, dass die Verarbeitung des Kommandos noch andauert

- alle Buszyklen benötigen ein ganzzahliges Vielfaches des Bustaktes



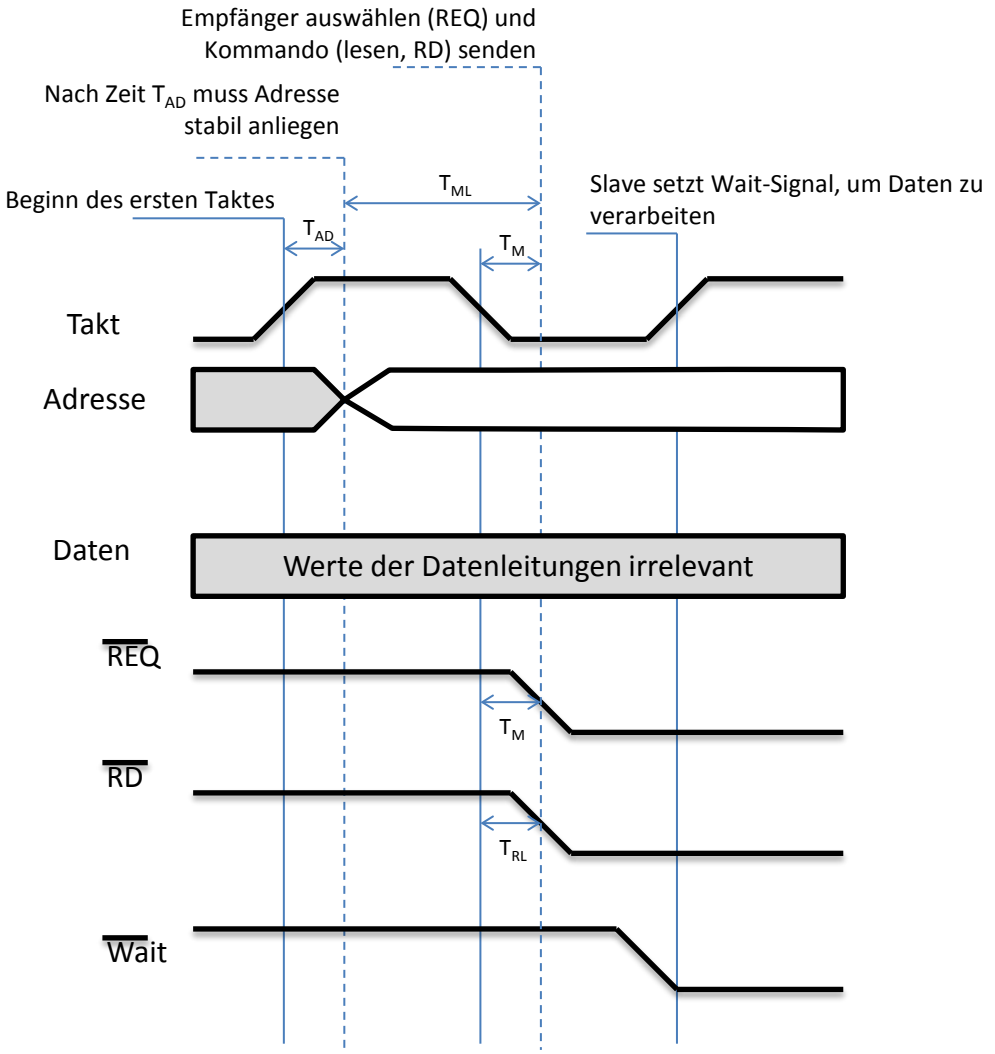
Beispiel
Ablauf einer Leseoperation



- Master setzt zu Beginn der steigenden Taktflanke Addressbits
- das muss spätestens nach Zeit T_{AD} passieren
- Daten sind noch ungültig
- REQ, RD, WAIT sind low-aktiv

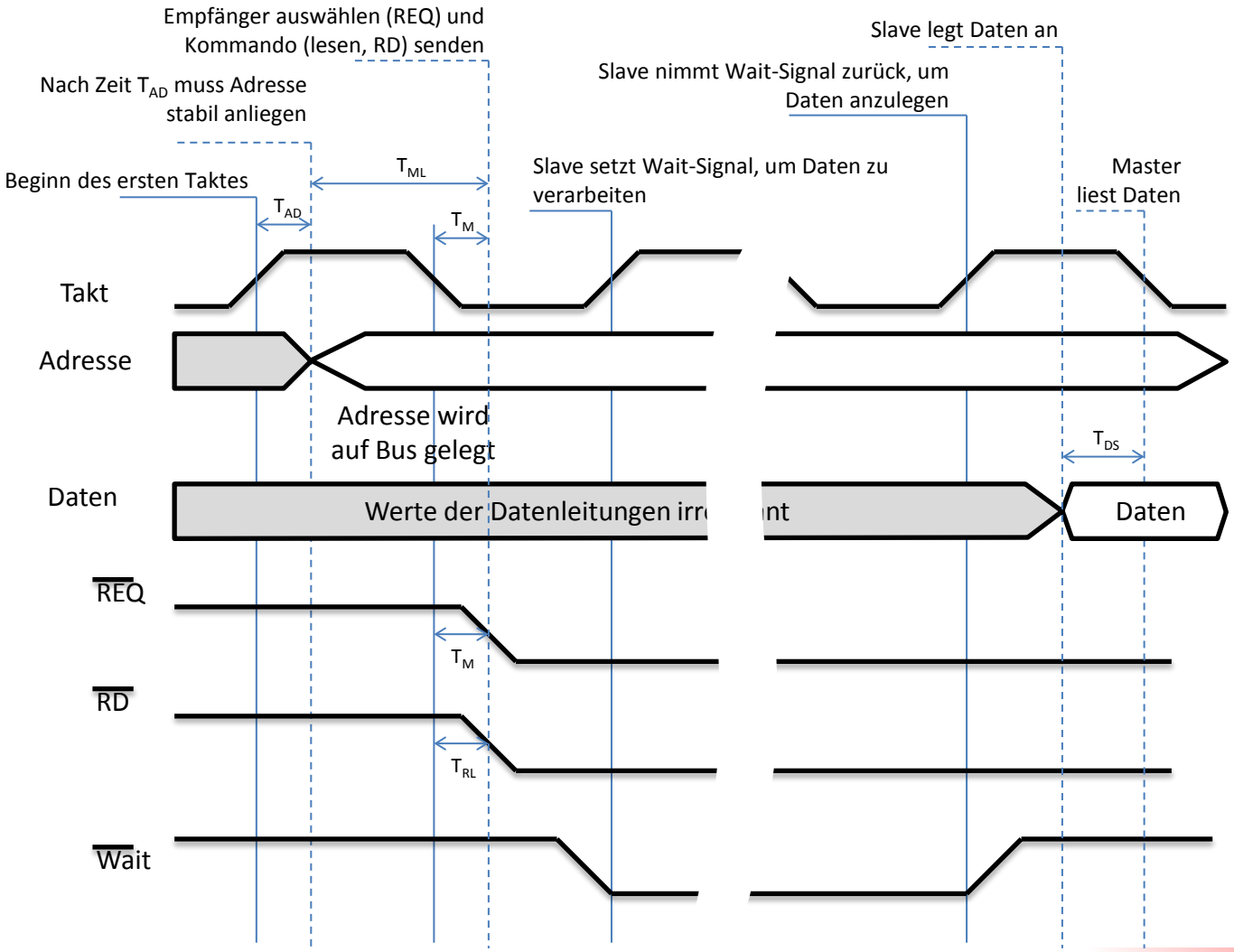
Alle logisch 1 (noch kein REQ, kein Kommando, kein Wait)

Beispiel
Ablauf einer Leseoperation



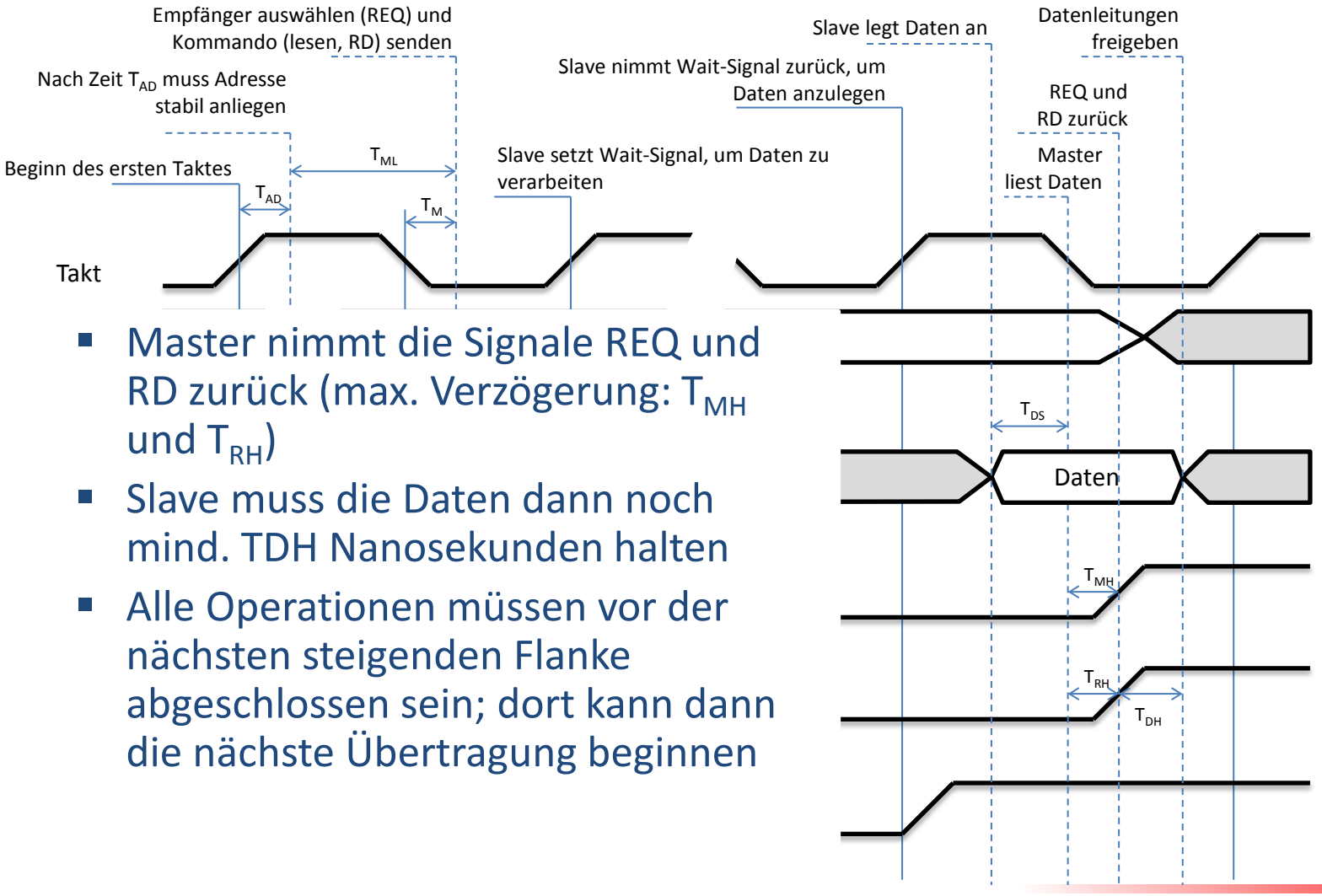
- In der zweiten Takthälfte werden Steuerinformationen an den Slave gesendet:
  - mit REQ Slave auswählen
  - mit RD Kommando sende (0 = lesen)
- Durch WAIT-Signal kann der Slave den Master beliebig lange, mind. aber einen Takt warten lassen

Beispiel
Ablauf einer Leseoperation



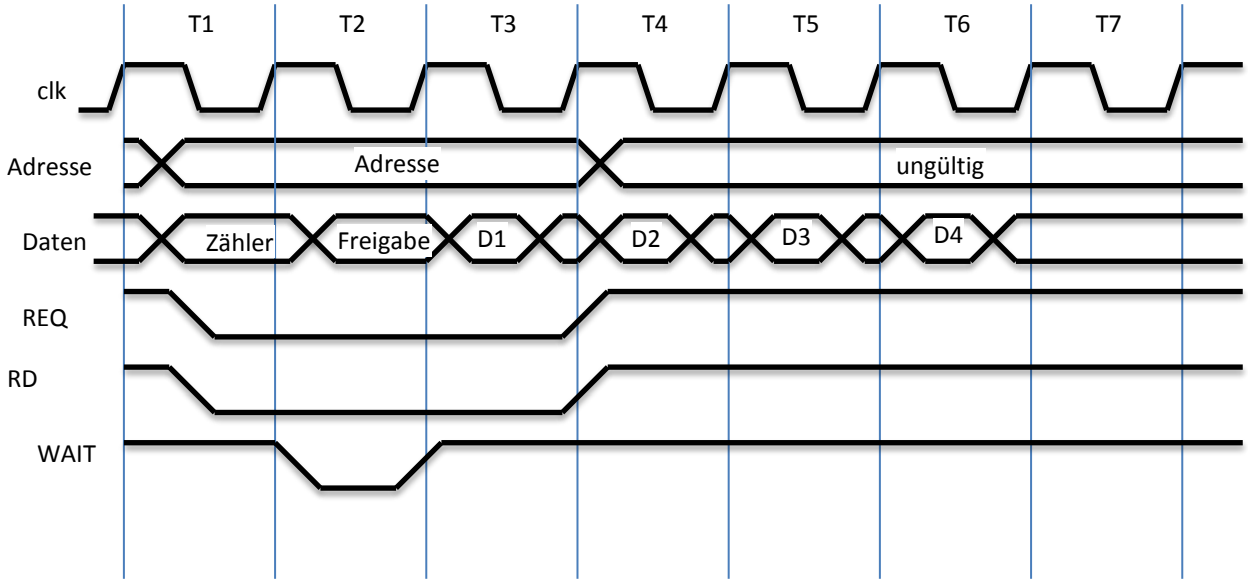
- Slave nimmt WAIT-Signal in erster Takthälfte zurück
- setzt Daten auf einen gültigen Wert mind. Zeit T_{DS} vor der fallenden Flanke
- Master liest die Daten bei der fallenden Flanke

Beispiel
Ablauf einer Leseoperation



- Master nimmt die Signale REQ und RD zurück (max. Verzögerung: T_{MH} und T_{RH})
- Slave muss die Daten dann noch mind. TDH Nanosekunden halten
- Alle Operationen müssen vor der nächsten steigenden Flanke abgeschlossen sein; dort kann dann die nächste Übertragung beginnen

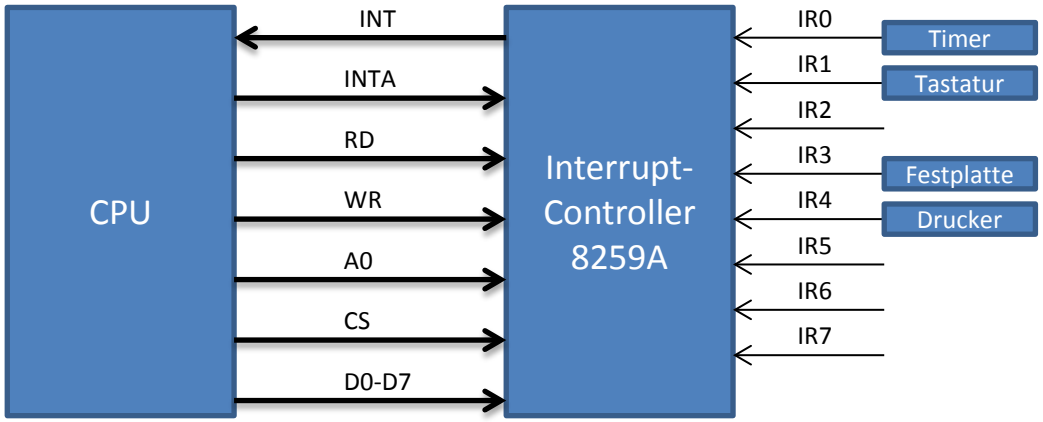
Blockübertragung



- Zähler übermittelt erwartete Anzahl der Worte in der Blockübertragung
- 4 Worte werden in 6 statt 12 Takten übertragen

Buszyklus für Interrupts

- Peripheriegeräte senden Interrupts an CPU
  - Priorisierung notwendig, falls Interrupts gleichzeitig auftreten
- Interruptcontroller übernimmt
  - Interruptverwaltung der Geräte
  - Signalisierung von Interrupts über Buskommunikation an CPU
- Buskommunikation für 8259A:
  - sendet IRQ-Anfrage über INT
  - CPU antwortet mit INTA
  - 8259A legt dann Nummer des Eingangs IRi auf Datenbus
  - CPU führt IRQ-Behandlung aus
  - CPU signalisiert Bereitschaft für weitere Interrupts durch schreiben eines internen Registers im 8259A
  - Konfiguration des 8259A über interne Register (können durch entsprechende Buszyklen gelesen (RD) und geschrieben (WR) werden



Weitere Aspekte beim Bus Design

Steigerung des Durchsatzes durch

- Höheren Takt bei synchronen Bussen
  - Probleme
    - Bus Skew: Signale breiten sich unterschiedlich schnell auf den Leitungen aus;
    - Abwärtskompatibilät nicht mehr gegeben
- Mehr Datenleitungen
  - Problem:
    - Mehr Platz
  - Umgehen durch gemultiplexten Bus
    - Dann aber mehr Buszyklen erforderlich

Zusammenfassung Synchroner Bus

Vorteile

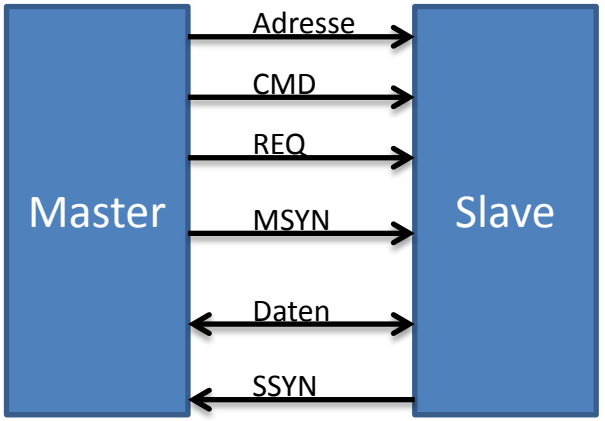
- relativ einfache Organisation der Kommunikation

Nachteile

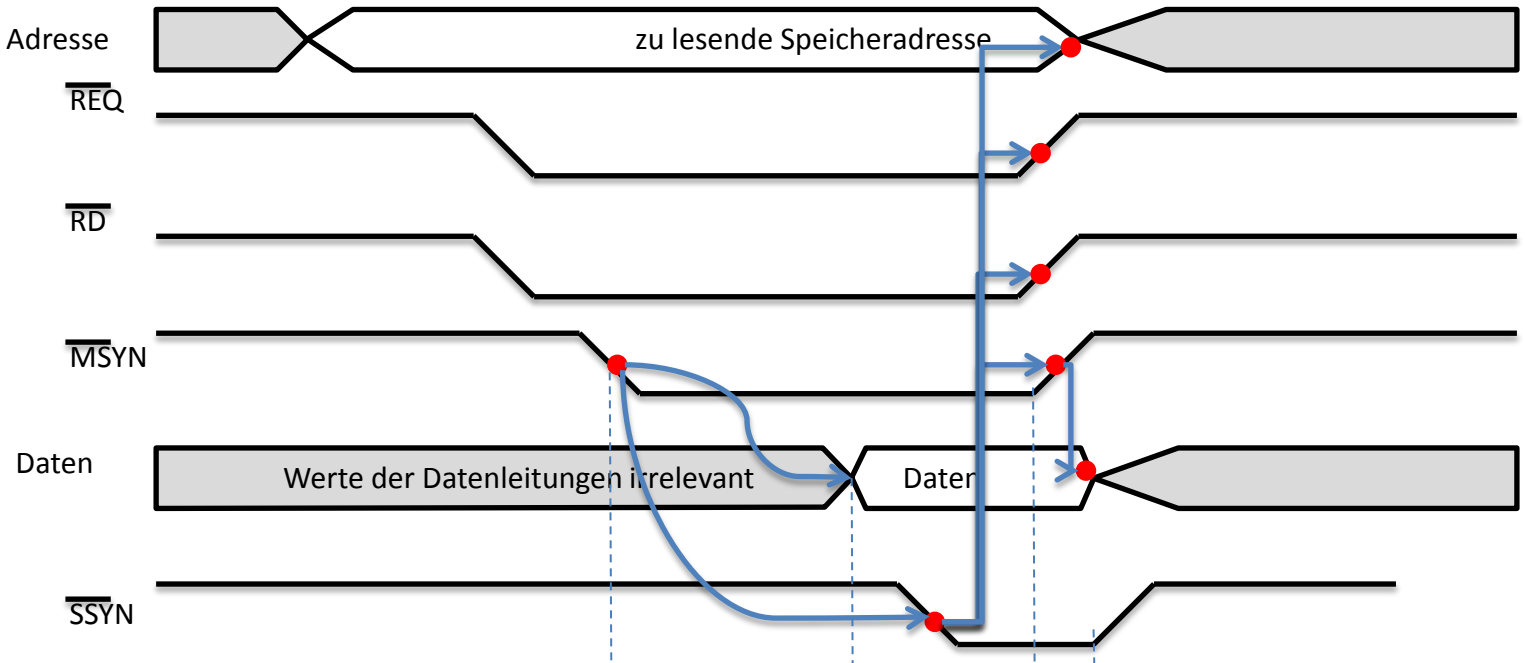
- Zeitverlust durch taktbasierte Diskretisierung
  - Ist eine Operation bereits nach 3,1 Bustakten abgeschlossen, dauert sie trotzdem 4 Takte
  - Vorteile durch Geschwindigkeitsgewinne bei künftigen Technologien (z.B. schnellere Speicher und dadurch Wartezeiten deutlich unter einem Bustakt) können nicht genutzt werden, um Kompatibilität mit älteren Geräten und Karten zu erhalten
- Bustakt muss sich an langsamsten Geräten orientieren, schnellere Geräte können kaum profotieren -> Lösung: asynchroner Bus

Arbeitsweise asynchroner Bus

- Es gibt kein Taktsignal
- Handshake zwischen Master und Slave üblich:
  - Wenn der Master alle relevanten Signale gesetzt hat, dann signalisiert er das über ein MSYN-Signal (Master SYNchronization)
  - Empfängt der Slave das Signal, führt er die erforderlichen Operationen möglichst schnell aus und zeigt deren Beendigung durch Setzen SSYN-Signals (Slave SYNchronization) an
  - Hat der Master noch Operationen auszuführen zeigt er deren Beendigung durch zurücknehmen von MSYN an
  - der Slave nimmt auch das SSYN-Signal zurück



Beispiel
Leseoperation asynchroner Bus



Master hat alle Signale gesetzt
und Slave kann mit Verarbeitung beginnen

Slave ist mit Bearbeitung fertig und
Master kann mit Verarbeitung beginnen

Slave damit auch

Master ist mit Verarbeitung fertig

Inhalt

- Grundprinzipien
  - Busarten und Verwendung (seriell/parallel, synchron/asynchron)
  - Busarbitration
  - Busprotokolle
- Beispiele
  - SPI
  - UART
  - PCI
  - PCI-Express

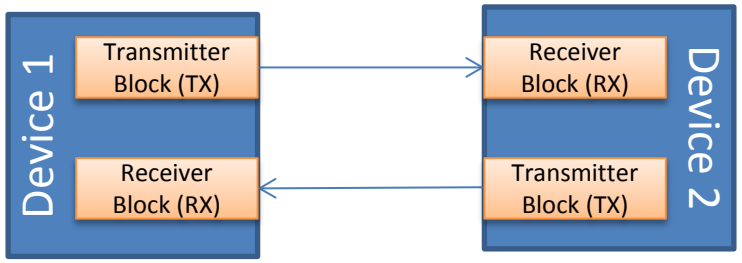
Überblick über Beispielbusse

Name	Funktion	Taktschema	Übertragungsart
UART	Chip-extern	asynchron	seriell
SPI	Chip-extern	synchron	seriell
PCI	Chip-intern/-extern	synchron	parallel
PCI-Express	Chip-intern/-extern	asynchron	seriell

UART

UART (Universal Asynchronous Transceiver and Transmitter)

- Punkt-zu-Punkt Verbindung zwischen zwei Geräten
- Bidirektional, full duplex
- Nur zwei Leitungen
- Keine Taktsignal (asynchronous)



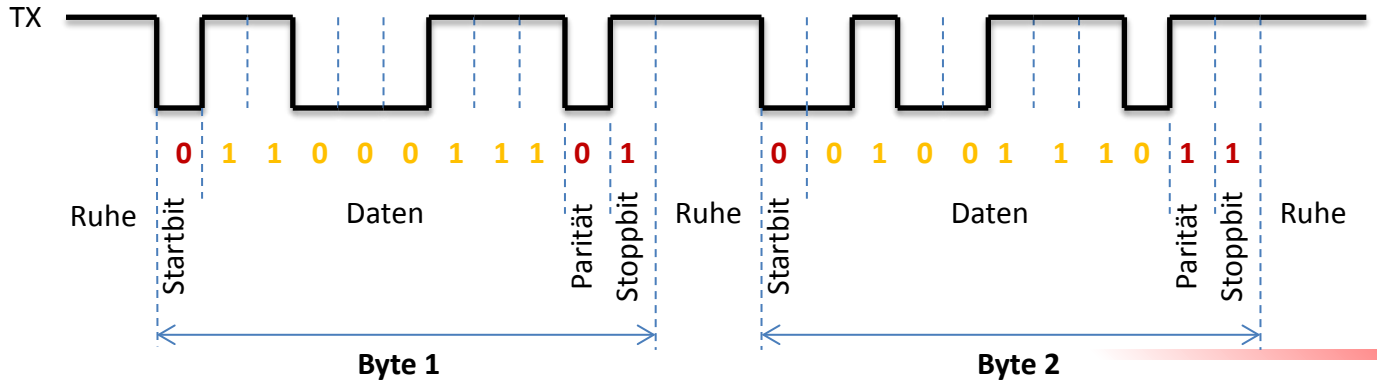
UART Datenübertragung

- Zu übertragende Daten werden
  - in einen Rahmen verpackt
  - und dann der Rahmen bitweise übertragen

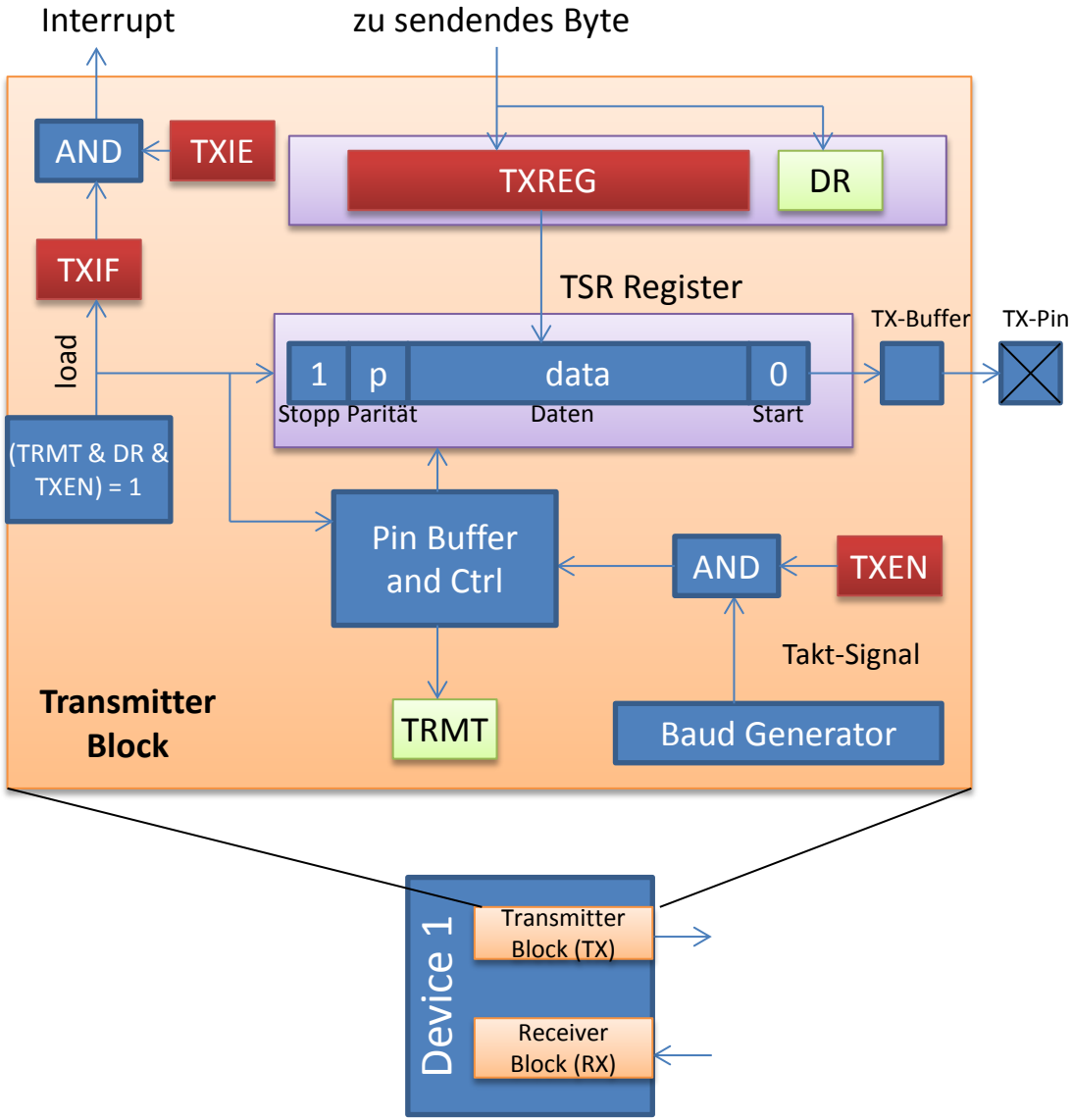
- Transmitter und Receiver müssen in folgenden Parametern übereinstimmen:
  - Baudrate: Übertragene Zeichen (hier Bits) pro Sekunde (9600 Baud bedeutet 0,00010417 Sekunden pro Bit)
  - Stoppbits: 1, 1.5 oder 2
  - Datenbits: 5,6,7, oder 8
  - Parität: gerade oder ungerade (ergänzt die Anzahl der 1en in den Datenbits zu einer geraden/ungeraden Anzahl)

- Rahmenformat:

- Beispiel für Datenübertragung:

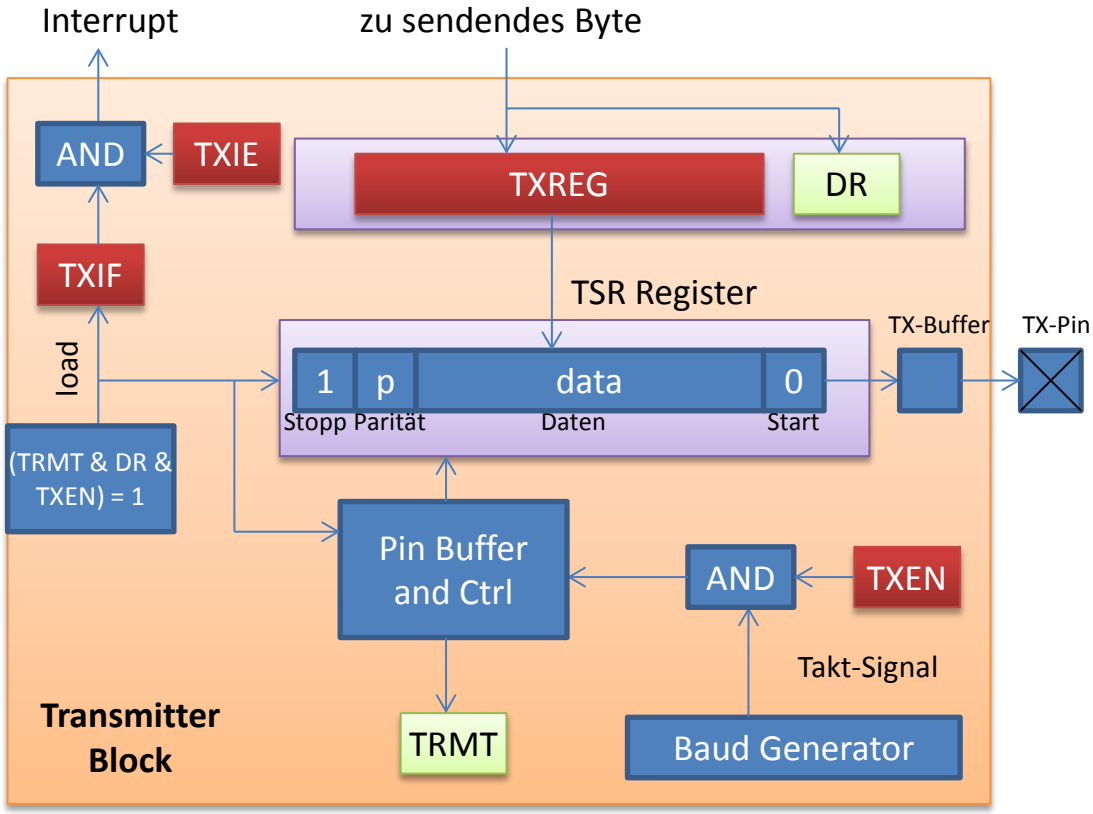


Aufbau des Transmitters



- TXEN aktiviert das Modul (setzt auch TRMT auf 1)
- TRMT = 1, wenn die Übertragung eines Bytes abgeschlossen ist
- in TXREG wird das zu übertragende Byte geschrieben (setzt DR auf 1)
- Wert aus TXREG wird in das Schieberegister für die Übertragung kopiert, wenn (TRMT & DR & TXEN) = 1
- TXIF = 1, wenn Daten in das Schieberegister übertragen wurden (wird zurückgesetzt, wenn Daten in TXREG geschrieben werden
- Ctrl steuert dann das Herausschieben der Bits aus dem TSR-Register

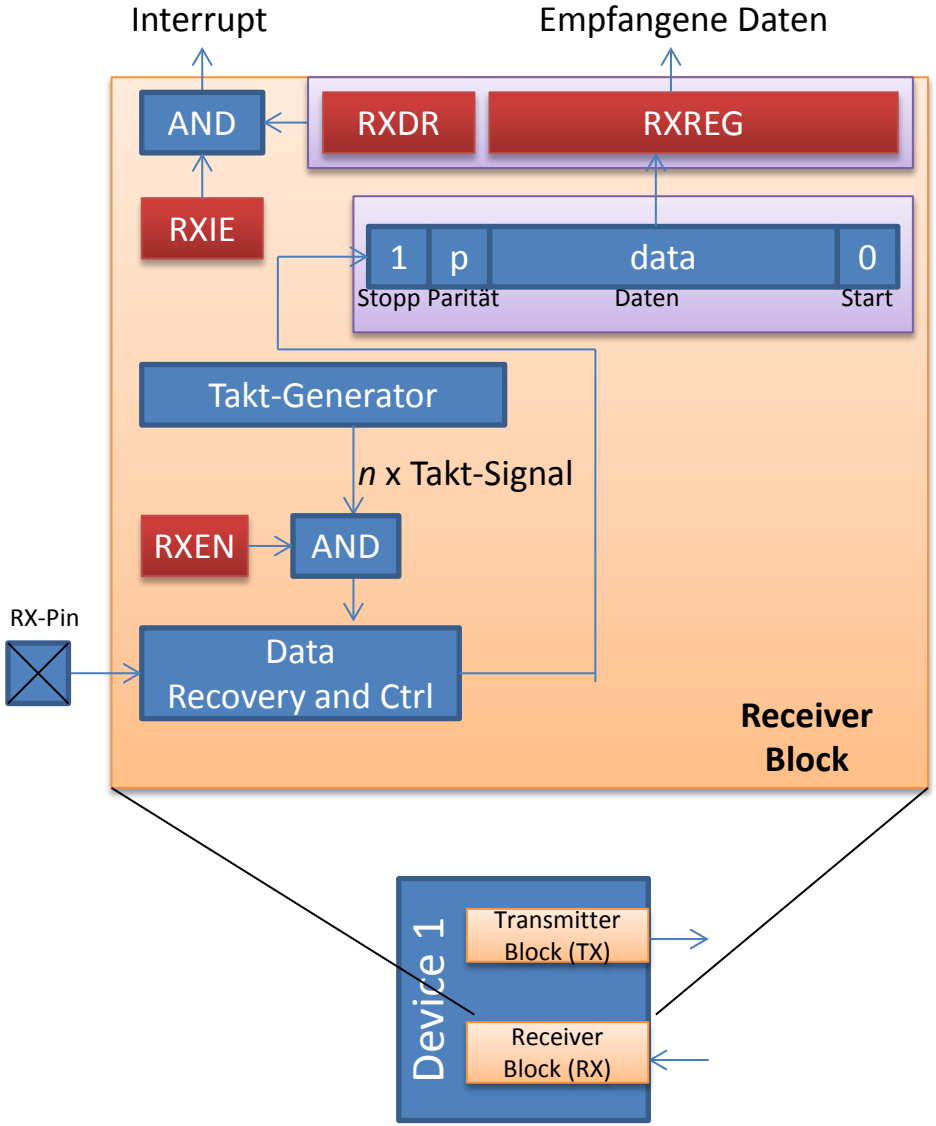
Senden von Daten in Software



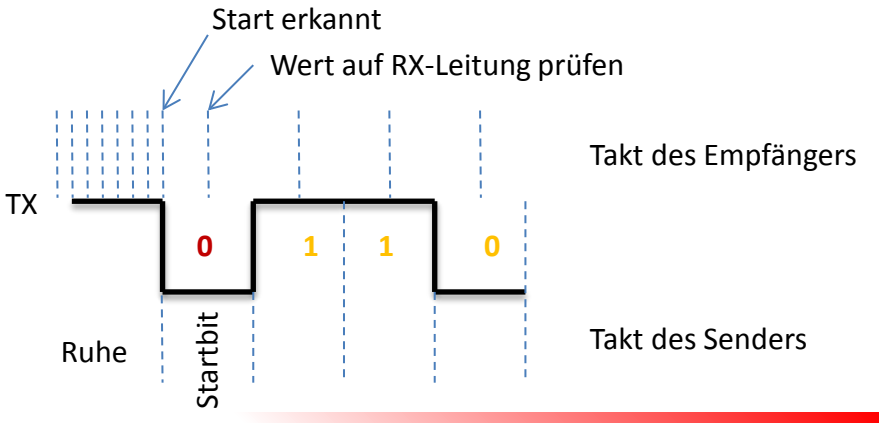

```
TXIE = 0; TXEN = 1
for(i = 0; i < 10; i++)
{
  TXREG = data[i];
  while(!TXIF);
}
```

- TXEN aktiviert das Modul (setzt auch TRMT auf 1)
- TRMT = 1, wenn die Übertragung eines Bytes abgeschlossen ist
- in TXREG wird das zu übertragende Byte geschrieben (setzt DR auf 1)
- Wert aus TXREG wird in das Schieberegister für die Übertragung kopiert, wenn (TRMT & DR & TXEN) = 1
- TXIF = 1, wenn Daten in das Schieberegister übertragen wurden (wird zurückgesetzt, wenn Daten in TXREG geschrieben werden
- Ctrl steuert dann das Herausschieben der Bits aus dem TSR-Register

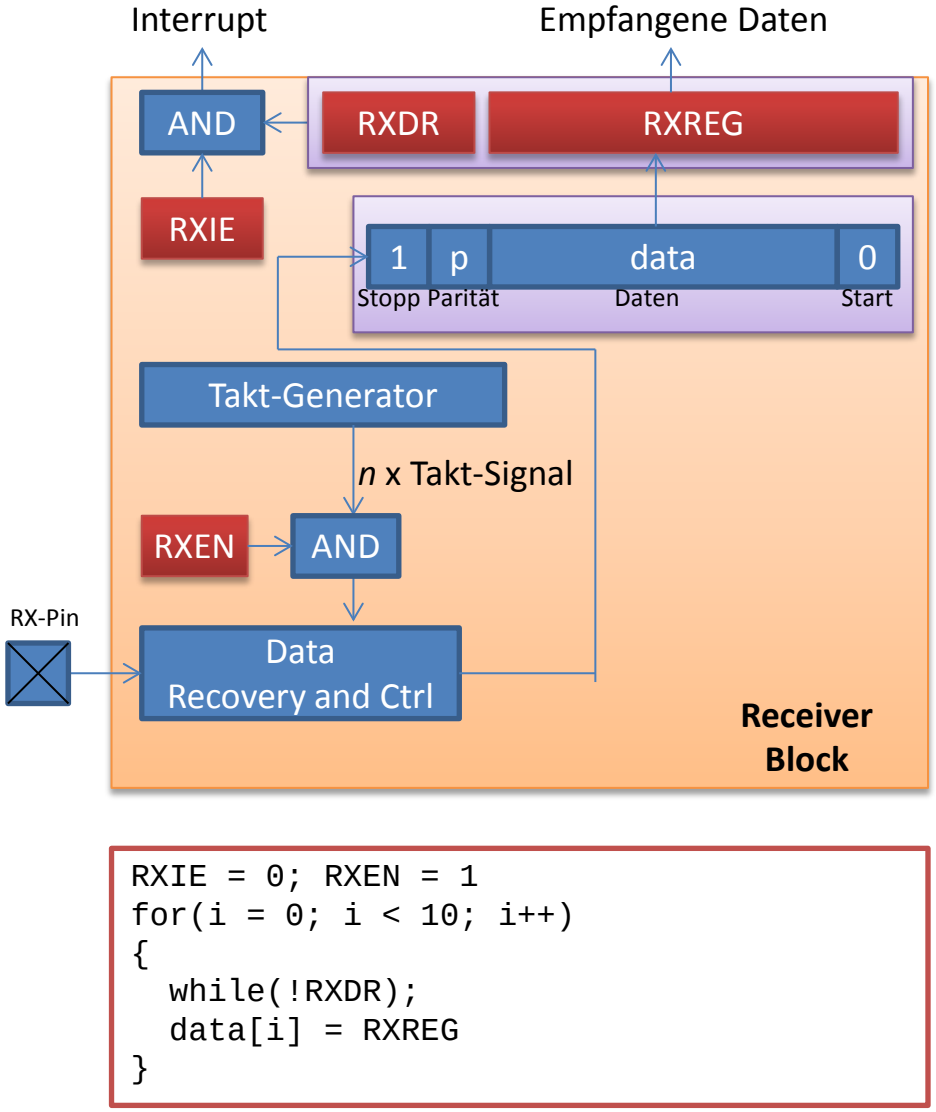
Aufbau des Receivers



- Takt des Empfängers ist *n* Mal höher als der des Senders
- Data Recovery prüft RX-Leitung auf Übergang auf 0 (Startbit) mit *n*-fach höherer Taktrate
- Receiver wartet dann *n*/2 Takte und sampelt dann alle *n* Takte die nächsten Bits, die in das TSR-Register geschoben werden
- Wurde das Stoppbit ins TSR-Register geschoben, dann wird der Datenwert in das RXREG kopiert und RXDR auf 1 gesetzt

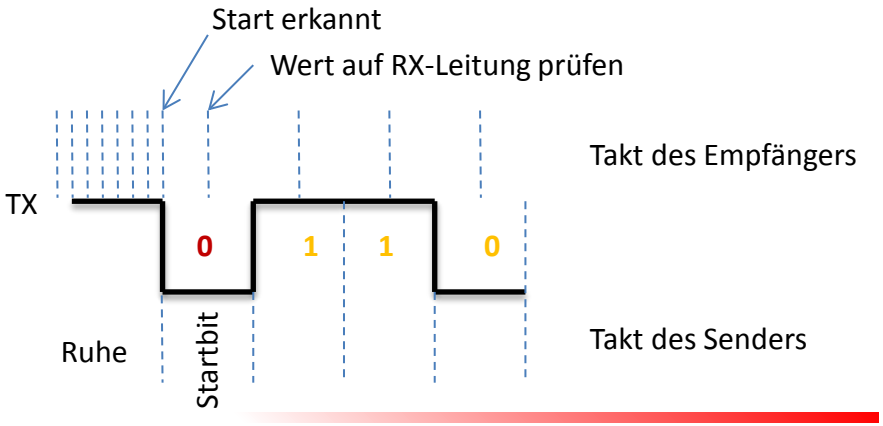


Empfangen von Daten in Software



- Takt des Empfängers ist *n* Mal höher als der des Senders
- Data Recovery prüft RX-Leitung auf Übergang auf 0 (Startbit) mit *n*-fach höherer Taktrate
- Receiver wartet dann *n*/2 Takte und sampelt dann alle *n* Takte die nächsten Bits, die in das TSR-Register geschoben werden
- Wurde das Stoppbit ins TSR-Register geschoben, dann wird der Datenwert in das RXREG kopiert und RXDR auf 1 gesetzt
- RXDR wird durch Auslesen von RXREG auf 0 gesetzt

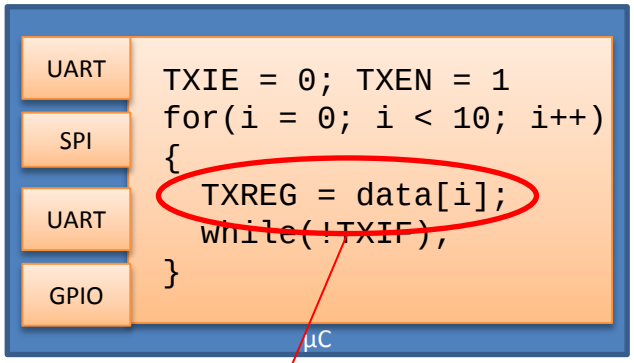
```
RXIE = 0; RXEN = 1
for(i = 0; i < 10; i++)
{
  while(!RXDR);
  data[i] = RXREG
}
```



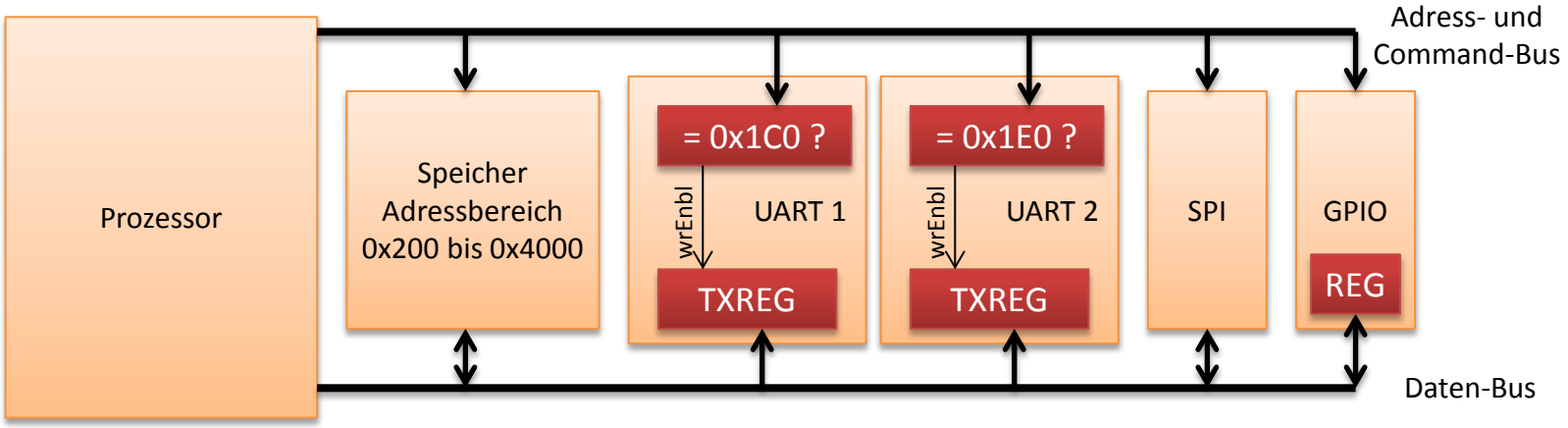
Integration in den Prozessor

Wie kann die Software auf die UART-Register zugreifen?

- Peripherie-Blöcke (Speicher, UART, SPI, GPIO) bekommen Adressbereiche zugewiesen (Memory-Mapped I/O)
- Lese-/Schreibzugriffe des Prozessors auf diese Adressbereiche werden vom entsprechenden Peripherie-Block behandelt
- Beispiel
  - TXREG von UART1 ist auf Adresse 0x1C0 abgebildet



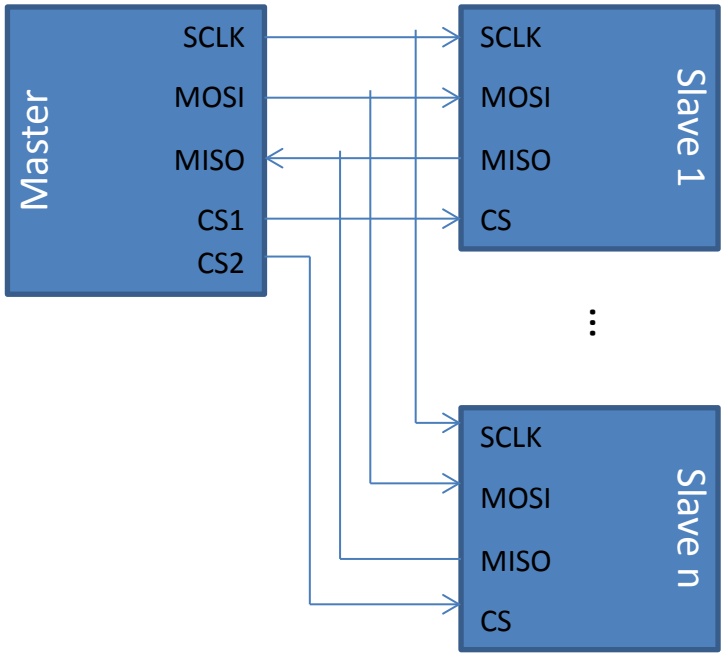

```
uint8* p = 0x1c0;
*p = data[i];
```



SPI

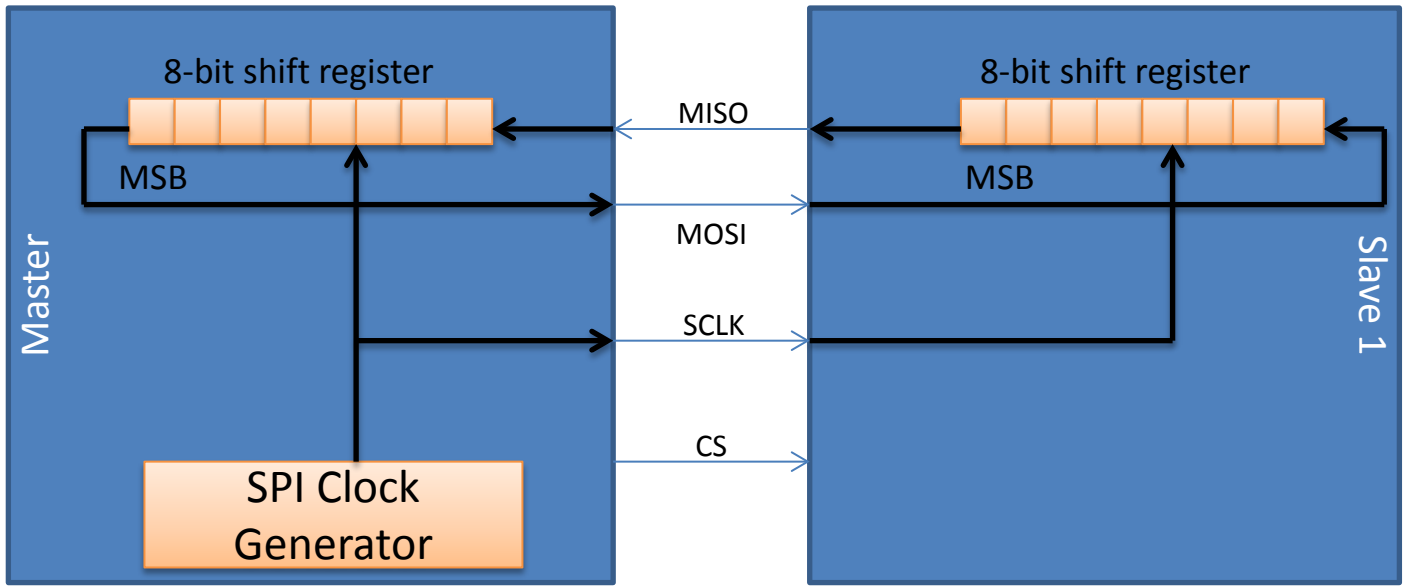
SPI (Serial Peripheral Interface)

- Serieller Datenaustausch zwischen einem Master und mehreren Slaves
  - Master wählt über CS*i* Slave *i* aus
- bidirektional, vollduplex
- mind. 4 Leitungen
- Taktsignal wird verwendet (synchron)
  - Master legt den Takt fest
  - Master und Slave senden gleichzeitig Daten



SPI Implementation

Implementation of sender/receiver module

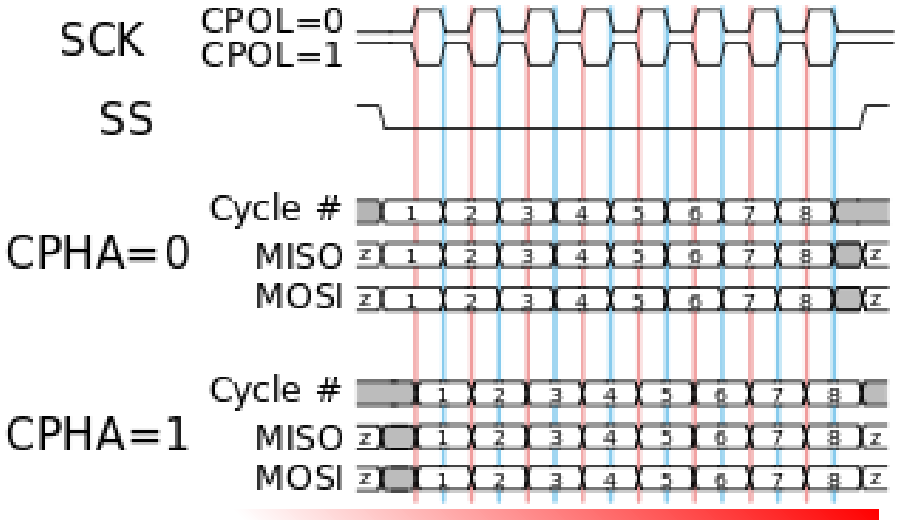


SPI Bus Protocol (1)

Possible Configurations

- Clock Polarity (CPOL): determines the leading and falling edge
  - CPOL = 0: leading edge = raising, trailing edge = falling
  - CPOL = 1: leading edge = falling, trailing edge = raising

- Clock Phase (CPHA = 1)
  - data is written on the leading edge of the current clock cycle
  - data is captured on the trailing edge of the current clock cycle
  - data is hold valid until the leading edge of the following clock cycle

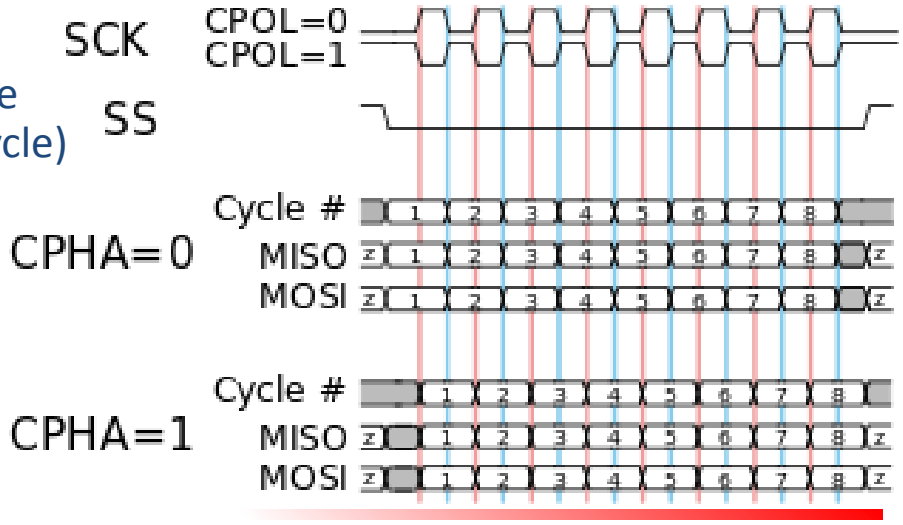


SPI Bus Protocol (2)

Possible Configurations

- Clock Polarity (CPOL): determines the leading and falling edge
  - CPOL = 0: leading edge = raising, trailing edge = falling
  - CPOL = 1: leading edge = falling, trailing edge = raising

- Clock Phase (CPHA = 0)
  - data is written on the trailing edge of the current clock cycle
  - data is captured on the next leading edge (i.e. at the beginning of the next clock cycle)
  - data is hold valid until the next trailing edge (of the next clock cycle)
  - data must be valid before the first leading edge



PCI-Bus

- Synchron
- Master = Initiator
- Slave = Target
- Adressen und Daten werden gemultiplext
- damit nur 64 Pins erforderlich, obwohl 64-Bit Daten und Adressen unterstützt werden

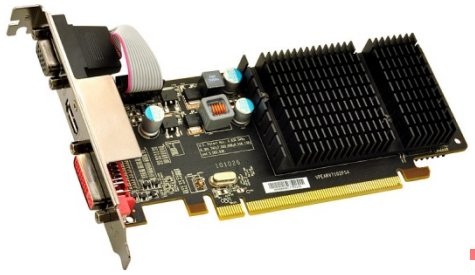
Vergleich PCI- und ISA-Bus

ISA-Bus und EISA-Bus

- Vorgänger des PCI-Busses in PC-Systemen
- Takt:
  - ISA: 8,33 MHz
  - EISA: 8,33 MHz
- Übertragung pro Zyklus
  - ISA: 2 Byte
  - EISA: 4 Byte
- Bandbreite
  - ISA: 16,7 MB/s
  - EISA: 33,3 MB/s

PCI-Bus

- Seit den 1990ern in Pentium-Systemen verwendeter Bus
- Takt:
  - PCI 1.0: 33 MHz
  - PCI 2.1: 66 MHz
- Übertragung pro Zyklus
  - PCI 1.0: 4 Byte
  - PCI 2.1: 8 Byte
- Bandbreite:
  - PCI 1.0: 133 MB/s
  - PCI 2.1: 528 MB/s



Anforderungen an Busse

Video

- 30 Bilder pro Sekunde
- 1024 x 768 Punkte
- 3 Byte je Punkt
- 67,5 MB/s

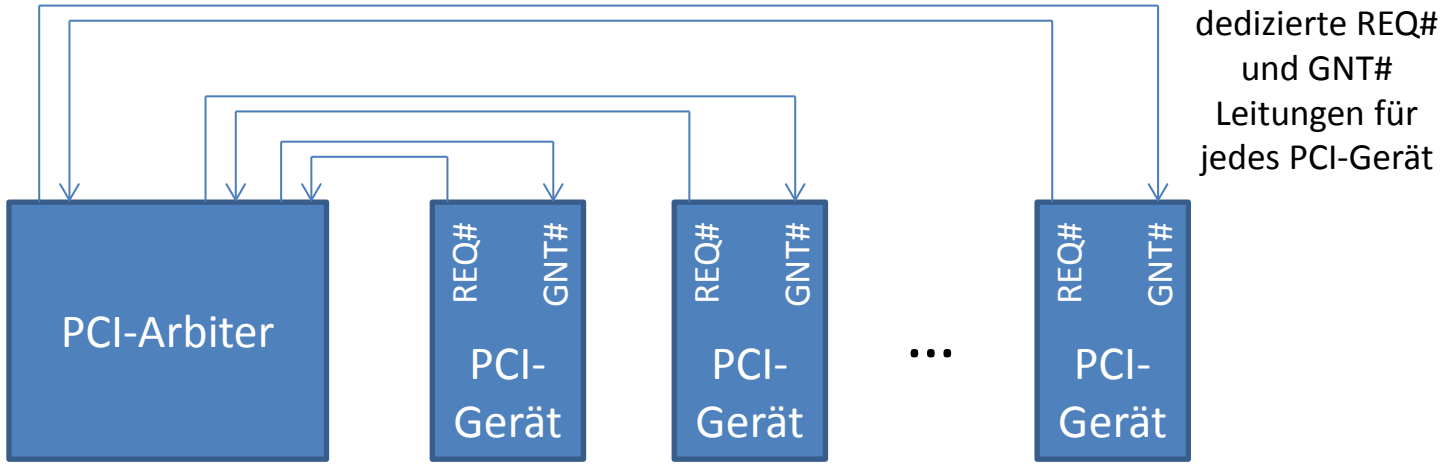
Full HD Video

- 30 Bilder pro Sekunde
- 1920 x 1080 Punkte
- 3 Byte je Punkt
- 155 MB/s

In der Realität müssen diese Daten mehrfach über den Bus gehen (HDD -> Mem -> Videokarte)

PCI-Bus Arbitration

- Arbiter kontrolliert den Zugriff auf den Bus
- Arbiter ist in PCI-Systemen in der Regel im Brückenchip untergebracht
- Arbitration:
  - Für Anforderung des Busses setzt ein Gerät REQ# auf high
  - Setzt der Arbiter GNT# auf 1, dann kann das Gerät im nächsten Zyklus den Bus benutzen



Busüberlassung

- Busüberlassung gilt für eine Transaktion; diese kann theoretisch beliebig lange dauern
- Ein Master kann mehrere Transaktionen hintereinander durchführen; muss dann einen Leerlaufzyklus zwischen den Transaktionen einschieben
- Um sehr lange Transaktionen unterbrechen zu können:
  - GNT#-Signal wird vom Arbiter auf high gehalten, solange der Master den Bus benutzen kann
  - Wird das Signal vom Arbiter zurückgenommen, muss der Master den Bus im nächsten Zyklus freigeben
  - das erlaubt lange effiziente Transaktionen, aber dennoch kurze Reaktionszeiten, wenn andere Geräte den Bus benutzen möchten

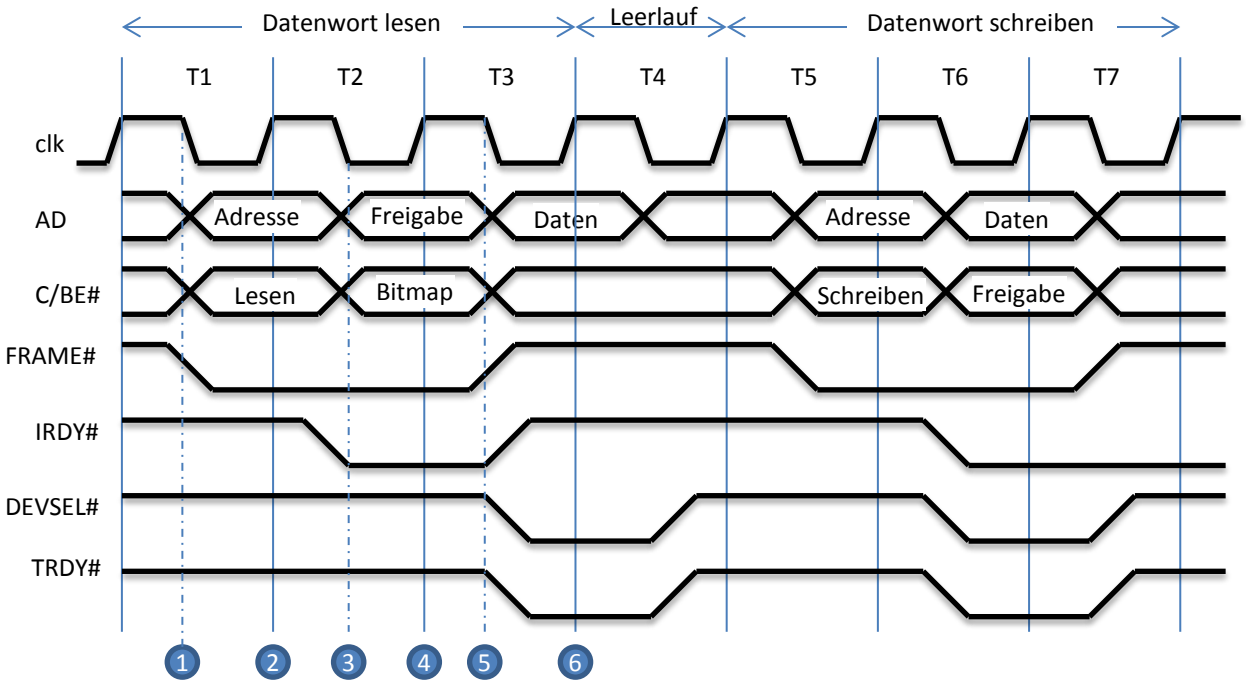
PCI-Bussignale (32-Bit Bus)

Signal	Leitungen	Master	Slave	Beschreibung
CLK	1			Taktsignal (33 oder 66 MHz); eine PCI-Bus-Transaktion beginnt in der Mitte eines Taktes (bei der fallenden Flanke)
AD	32	x	x	Gemultiplexte Adress- und Datenleitungen (typisch: Adresse in Zyklus 1; Daten in Zyklus 3)
PAR	1	x		Adress- oder Datenparitätsbit
C/BE	4	x		Busbefehl (Zyklus 1)/Bitmap für Byteempfang (zeigt gültige Datenbytes an, üblicherweise in Zyklus 2)
FRAME#	1	x		Master zeigt dem Slave den Start einer Transaktion an, d.h. dass AD und C/BE gültig sind
IRDY#	1	x		Lesen: Master ist breit Daten zu akzeptieren, wird gleichzeitig mit FRAME# aktiviert Schreiben: Daten liegen an, wird dann in Zyklus 3 aktiviert
IDSEL	1	x		Wählt Konfigurationsraum statt Speicher; jedes PCI-gerät besitzt 256 Byte für Konfigurationsdaten; diese können von anderen Geräten ausgelesen werden; So können Geräte am Bus erkannt werden
DEVSEL#	1		x	Slave hat seine Adresse auf den AD-Leitungen erkannt und ist in Bereitschaft; Master besitzt ein Timeout, falls der gewünschte Slave nicht antwortet
TRDY#	1		x	Lesen: Daten vom Slave liegen an Schreiben: Slave ist bereit Daten zu akzeptieren
STOP#	1		x	Slave möchte Transaktion sofort wegen eines Fehlers abbrechen
PERR#	1			Empfänger hat Paritätsfehler in voriger Transaktion erkannt
SERR#	1			Adressparitätsfehler oder Systemfehler erkannt
REQ#	1			Anforderung des Busses
GNT#	1			Zuteilung des Busses
RST#	1			Setzt das System und alle Geräte zurück

Zusätzliche PCI-Bussignale (64-Bit Bus)

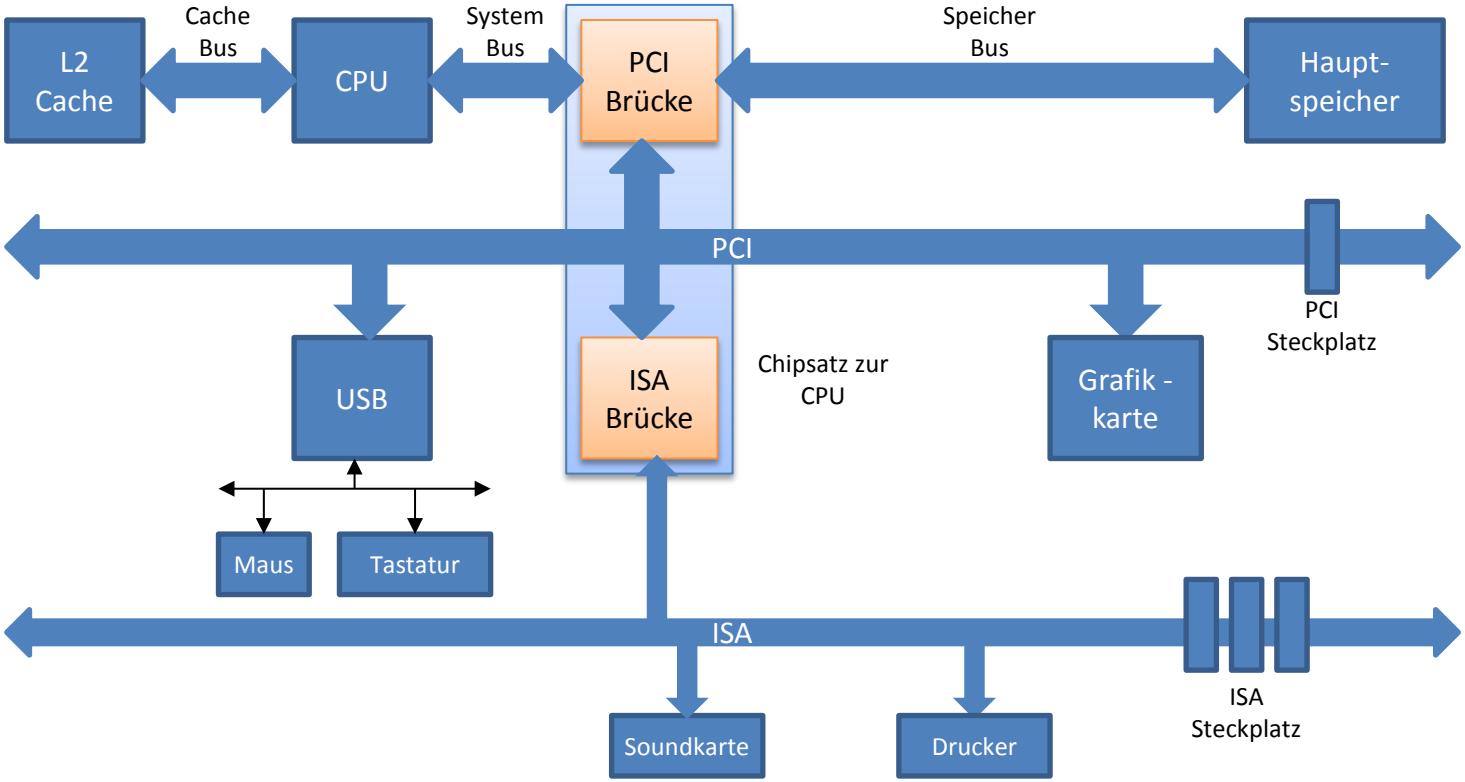
Signal	Leitungen	Master	Slave	Beschreibung
REQ64#	1	x		Anforderung zur Ausführung einer 64-Bit –Transaktion
ACK64#	1		x	Berechtigung für eine 64-Bit Transaktion wird gewährt
AD	32	x		Zusätzliche 32 Bits für Adressen oder Daten
PAR64	1	x		Parität für zusätzliche 32 Daten-/Adressleitungen
C/BE#	4	x		Weitere 4 Bits für Byte Enable (zeigt gültige Datenbytes an)
LOCK	1	x		Bus sperren, um mehrere Transaktionen zu erlauben
SBO#	1			Treffer in einem entfernten Cache (bei einem Mehrprozessorsystem)
SDONE	1			Snooping ausgeführt (bei einem Mehrprozessorsystem)
INTx	4			Anforderung eines Interrupts; bis zu vier logische Geräte sind pro PCI-Karte möglich, jedes davon mit einem eigenen Interrupt
JTAG	5			JTAG-Testsignal nach IEEE 1149.1
M66EN	1			An Betriebsspannung oder Masse gelegt (wählt 66 oder 33 MHz aus)

PCI-Bustransaktionen



1. Master setzt Adress- und Kommandoleitung und zeigt dem Slave über FRAME# den Beginn einer Transaktion an
2. Slave kann Adresse und Kommando lesen
3. Master schaltet Adressleitungen hochohmig (vgl. Tristate) und übermittelt über C/BE# Bitmap für gültige Bytes
4. Slave kann Bitmap lesen
5. Slave bestätigt mit DEVSEL# erhalt der Adresse; Slave legt Daten auf die AD-Leitungen und zeigt deren Gültigkeit mit TRDY# an (das kann verzögert passieren)
6. Master liest die Daten vom Bus

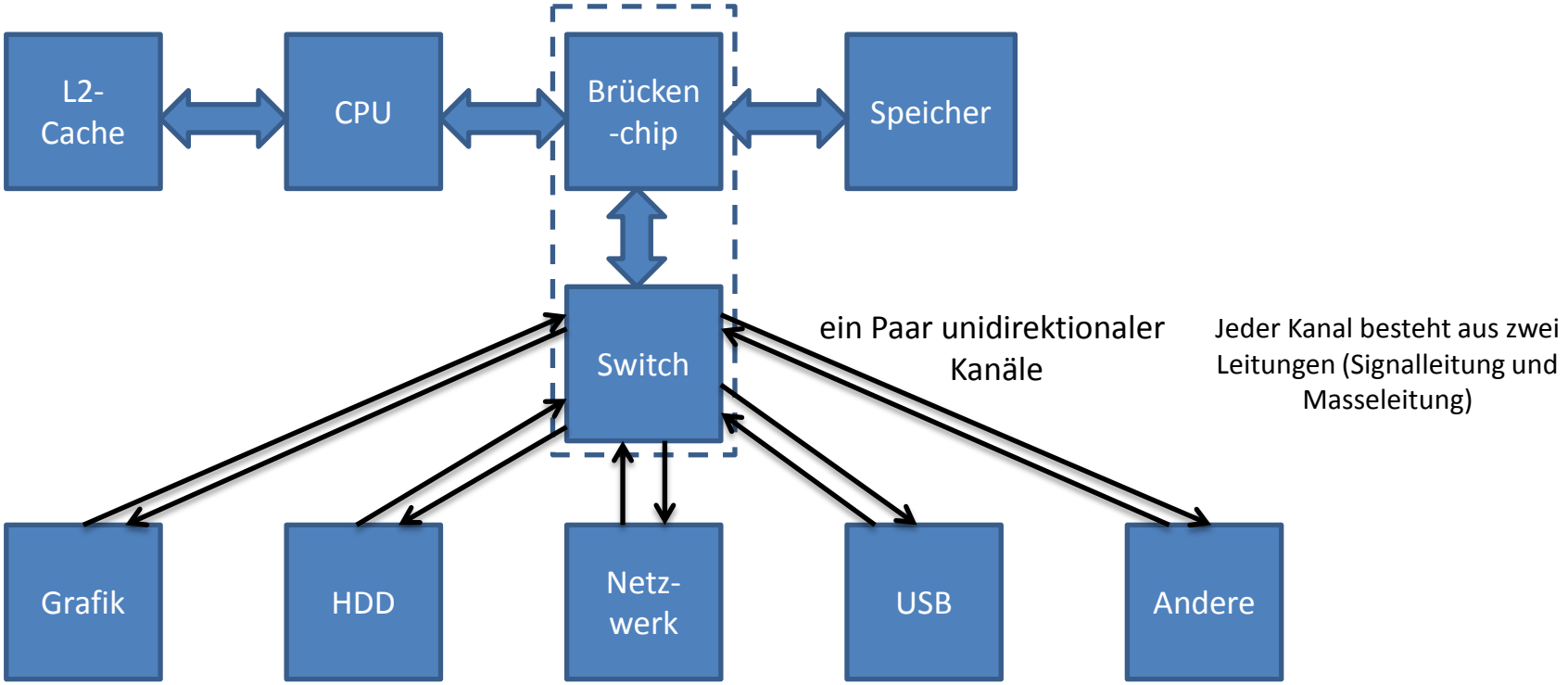
Architektur eines Bussystems im Pentium mit PCI



- Funktionen PCI-Brücke:
  - Verbindet verschiedene Busse (Speicherbus und PCI-Bus) mit dem Systembus (direkt an CPU)
  - enthält den Arbiter für PCI-Bus
  - schnelle Peripherie ist an PCI-Bus angeschlossen
- Funktionen ISA-Brücke:
  - Verbindung zu älterem ISA-Bus
  - langsamere und ältere Peripherie ist an ISA-bus angeschlossen

PCI Express Architektur

- Ersetzen eines parallelen Mehrpunktbusses (PCI) durch mehrere serielle Punkt-zu-Punkt Verbindungen
- Paketorientierte Kommunikation statt Master-Slave-Kommunikation
  - dadurch weniger Steuerleitungen

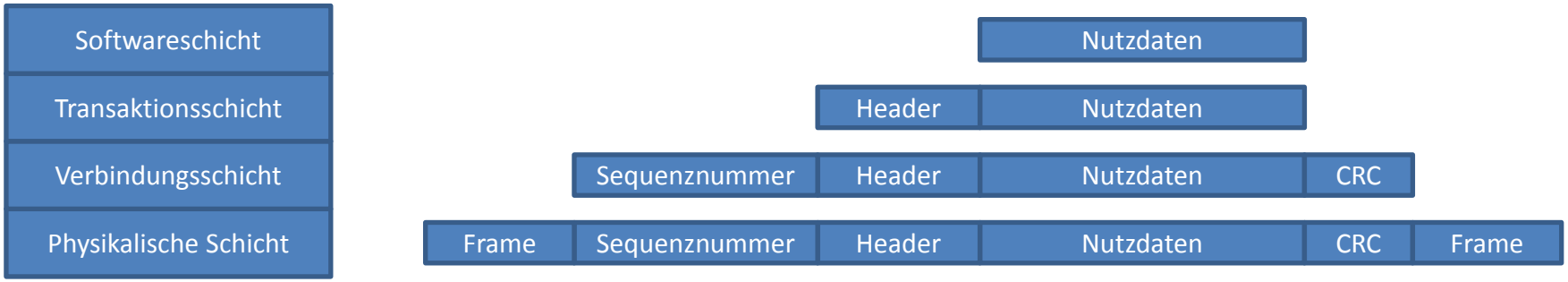


Eigenschaften PCI-Express

- serielle Punkt-zu-Punkt-Verbindung zwischen PCI-Express-Geräten mit zentralem Switch
- Ein Switch kann selbst ein PCI-Express-Gerät sein, damit baumartiger Aufbau eines Netzwerkes möglich
- Paketorientierte Kommunikation
- Pakete enthalten Fehlerkorrekturcodes
  - höhere Zuverlässigkeit als bei PCI
- Verbindung zwischen Switch und einem Chip kann bis zu 50 cm lang sein
- PCI-Express-Geräte sind Hot-Plugging-fähig

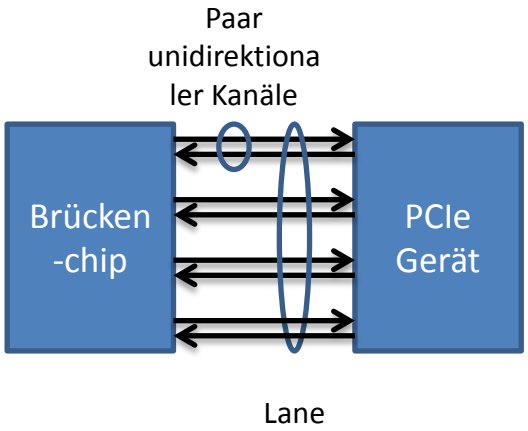
Paketorientierte Kommunikation

- Wenn ein Gerät Daten zu versenden hat, dann werden diese als Nutzdaten in einem Datenpaket versendet
- Datenpaket enthält in einem Header weitere Steuerdaten (z.B. Zieladresse)
- Kommunikation ist in Schichten gegliedert, jede Schicht hat eine bestimmte Aufgabe und unterstützt ein Protokoll



Physical Layer

- Lane: Punkt-zu-Punkt Verbindung aus 1, 2, 4, 8, 16 oder 32 Paaren unidirektionaler Kanäle bestehen
- Jede Richtung muss Datenrate von mind. 2,5 GBit/s unterstützen
- Übertragung in 8b/10b Kodierung (8 Bit werden in 10 Bit kodiert); damit nur 2GBit/s Nettodatenrate in jede Richtung



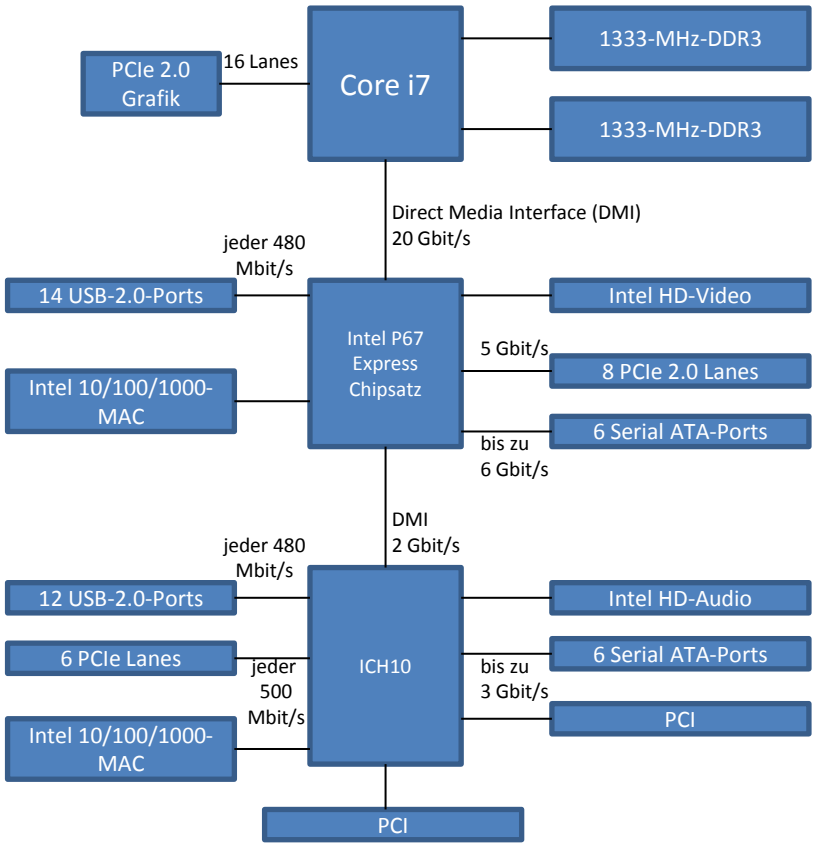
Verbindungsschicht

- Erzeugen einer Sequenznummer und eines CRC für Daten und Header
- Empfänger prüft CRC und sendet Ack-Paket bei Übereinstimmung, sonst fordert er eine Neuübertragung an
- Flussteuerung, um Überfluten eines langsamen Empfängers durch einen schnellen Sender zu verhindern
  - Empfänger und Sender verwenden jeweils einen Ringpuffer derselben Größe für Datenpakete
  - Sender puffert unbestätigte Datenpakete und sendet keine neuen Pakete, wenn sein Puffer voll ist

Transaktionsschicht

- Organisiert Busaktionen zur Ausführung einer Transaktion, z.B.:
  - Prozessor fordert ein Speicherwort aus dem Speicher mit einem Paket an (1. Transaktion)
  - Speicher sendet das Datenwort in einem Paket (2. Transaktion)
- Unterteilung einer Lane in bis zu 8 virtuelle Verbindungen für unterschiedliche Klassen von Datenpaketen
  - Pakete werden entsprechend ihrer Verkehrsklasse markiert (hohe Priorität, niedrige Priorität, Übertragung unabhängig von Reihenfolge, etc.)
  - Switch kann abhängig dieser Tags entscheiden, welche Pakete als nächstes vermittelt werden
- Adressräume für Transaktionen:
  - Speicherraum (für Lese- und Schreiboperationen in den Speicher)
  - E/A-Raum (für die Adressierung von Geräteregistern)
  - Konfigurationsraum (für Plug-and-Play Fähigkeit bei Systeminitialisierung)
  - Nachrichtenraum (für Signalisierung und Interrupts)

Busarchitektur im Core i7



- Separater Speicherbus mit zwei Kanälen
  - jeder Kanal bewältigt
    - 10 GBit/s
    - erlaubt damit 1,333 Mrd Transaktionen/s
- P67 stellt folgende Schnittstellen bereit
  - 8 zusätzliche PCIe-Lanes
  - 1 SATA HDD-Schnittstelle
  - 14 USB-2.0 Schnittstellen
  - 10Gbit-Ethernetanschluss
  - Audioschnittstelle
- ICH10 ist optional; unterstützt ältere Schnittstellen, z.B.:
  - PCI
  - 1Gbit Ethernet

Zusammenfassung

- Grundlegende Prinzipien für Aufbau von Bussen
  - synchron/asynchron, seriell/parallel
- Busprotokoll, Buszyklen, typische Transaktionen
- Umsetzung dieser Prinzipien in realen Bussen
  - UART, SPI, PCI, PCIe