1. Grundlagen der herkömmlichen Bussysteme

1.1 Einführung

Bussysteme wurden entwickelt, um Geräte oder Funktionseinheiten über einheitliche Schnittstellen miteinander zu verbinden. Den ursprünglichen Bemühungen lagen bestimmte Idealvorstellungen zugrunde:

- einheitliche gemeinsame Signalwege. Als Ideallösung wurde ein „Bündel einfacher Drähte" angesehen, an das alle Geräte oder Funktionseinheiten auf gleiche Weise angeschlossen werden (1:1-Verbindung; Abb. 1.1).
- keine Zentralisierung. Alle Einrichtungen sind im Grunde gleichberechtigt.
- alles paßt zusammen. Jede Einrichtung kann an beliebiger Stelle in das System eingefügt werden.
- Einfachheit und Überschaubarkeit der Signalspiele bzw. Busprotokolle. Alle Übertragungvorgänge werden aus wohldefinierten Signalfolgen zusammengesetzt. Es wird eines nach dem anderen erledigt. Grundsatz: ein Übertragungsvorgang (zwischen zwei angeschlossenen Einrichtungen) zu einer Zeit.
- Freizügigkeit der Implementierung. Es sind lediglich die Vorgaben der Schnittstelle (also der Bus- oder Interfacestandard) einzuhalten - *wie* dies geschieht, ist aber gleichgültig. Vor allem soll es möglich sein, Geräte oder Funktionseinheiten nach unterschiedlichen Leistungs- und Kostenvorgaben zu bauen – und im praktischen Betrieb sollte sich auch alles mit allem vertragen (beispielsweise Billiggeräte und Hochleistungsgeräte am selben Interface).

Die Verwirklichung dieser Ideen führte zu zwei typischen Auslegungen:

1. *Geräte* werden über Kabel miteinander verbunden. Das ist die Auslegung der herkömmlichen E-A-Interfaces. Hierbei unterscheidet man zwischen interner und externer Verkabelung:
   - interne Verkabelung: die Einrichtungen befinden sich in einem gemeinsamen Gehäuse (z. B. im PC). Anschluß über ein einziges Kabel mit mehreren Steckverbindern (Abb. 1.2 bis 1.4).
   - externe Verkabelung: die Geräte befinden sich in eigenen Gehäusen. Jedes Gerät hat zwei Steckverbinder: einen für das ankommende und einen für das abgehende Kabel. Die Verbindung wird durch Kabel mit Steckern an beiden Enden hergestellt, die von Gehäuse zu Gehäuse weitergeschleift werden (Daisy-Chain-Prinzip; Abb. 1.5).
2. *Steckkarten* werden in einen Rahmen geschoben, dessen Rückwand durch eine sog. passive Rückverdrahtungsplatine (Backplane) gebildet wird. Diese Platine trägt keine Transistoren und Schaltkreise, sondern nur die Steckverbinder und die Signal- und Versorgungsleitungen (Abbildungen 1.6 bis 1.9). Das ist die herkömmliche Auslegung der Bussysteme, die in der industriellen Steuerungs- und Prozeßleittechnik, in der Meßtechnik, in der Telekommunikationstechnik usw. eingesetzt werden.

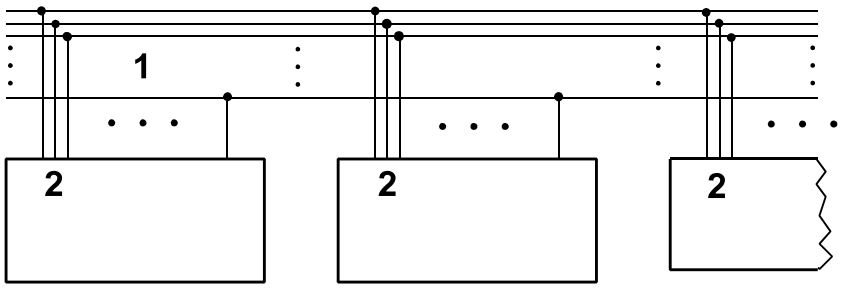


Abb. 1.1 Der Grundgedanke des Bussystems: an durchlaufende Busleitungen 1 sind die einzelnen Einrichtungen 2 auf gleiche Weise angeschlossen (1:1-Verbindung)

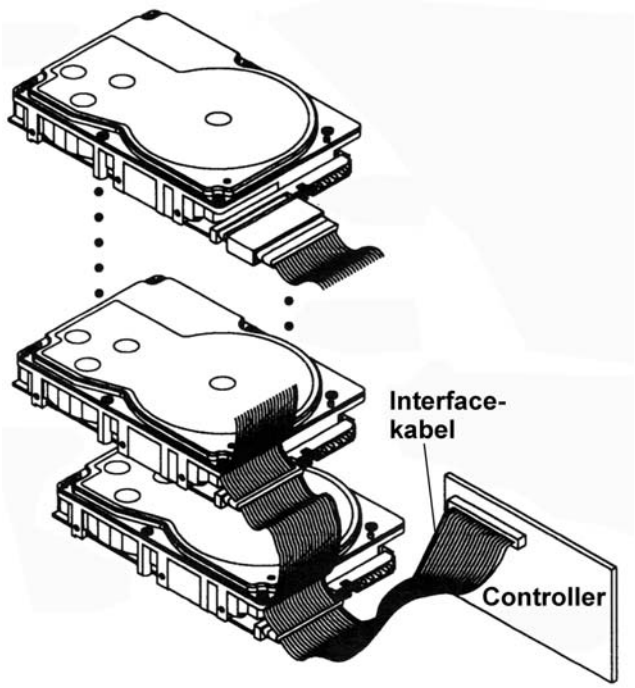
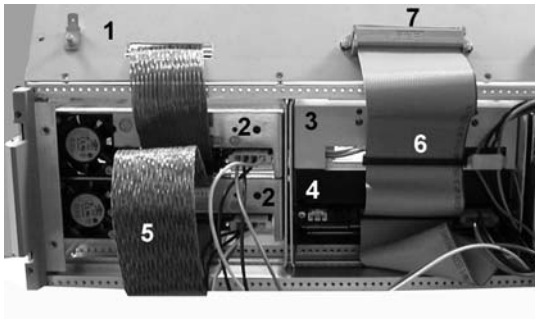


Abb. 1.2 Anschluß von Laufwerken über ein einziges Kabel mit mehreren Steckverbindern (interne Verkabelung). Hier eine SCSI-Konfiguration

Abb. 1.3 SCSI-Kabel für interne Verkabelung



1 - Rückwand (hochgeklappt); 2 - Einschübe für 16-Bit-SCSI-Geräte; 3 - Einschub für 8-Bit-SCSI-Gerät; 4 - DVD-RAM-Laufwerk; 5 - 16-Bit-SCSI-Kabel (Wide Ultra 2 SCSI); 6 - 8-Bit-SCSI-Kabel (SCSI 2); 7 - Steckverbinder für Externanschluß (Verbindung zum PC).

Abb. 1.4 Laufwerksverkabelung in einem 19"-Gehäuse (Beispiel der internen Verkabelung)

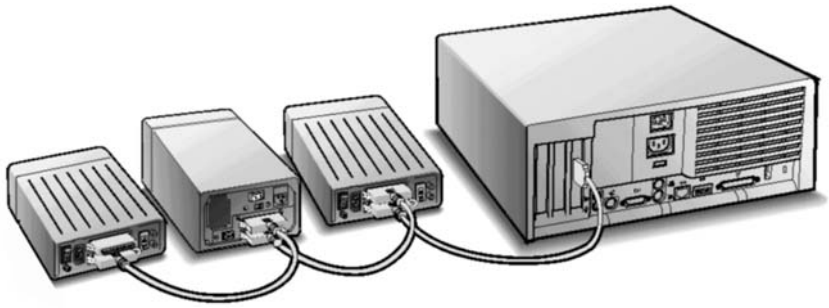


Abb. 1.5 Verkabelung externer Geräte (Adaptec)

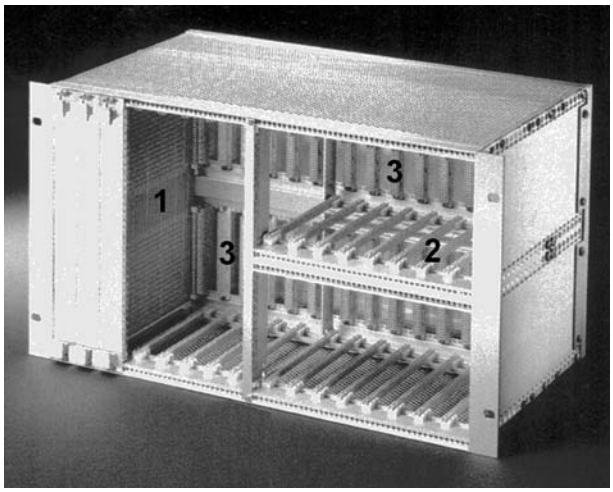


Abb. 1.6 Einbaurahmen mit rückseitiger Busplatine (Schroff). 1 - Einschubkassette mit Steckkarten; 2 - Führungsschienen; 3 - rückseitige Busplatine (Backplane) mit Steckverbindern

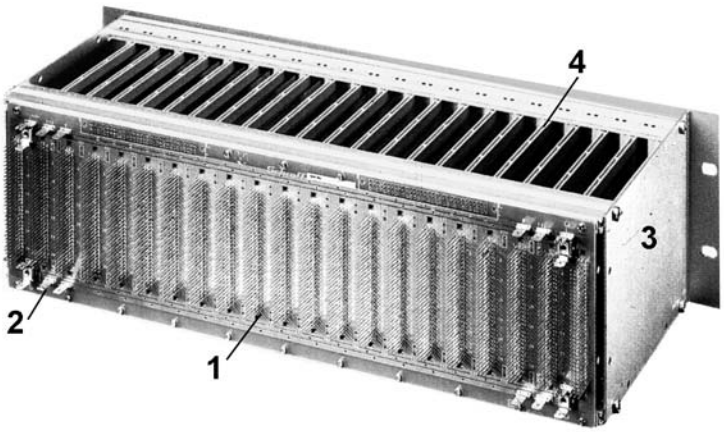


Abb. 1.7 Einbaurahmen mit Busplatine. Ansicht von hinten (Schroff). 1 - Busplatine mit Steckverbindern; 2 - Stromversorgungsanschlüsse; 3 - Seitenwand; 4 - Führungsschienen für Steckkarten oder Einschubkassetten.

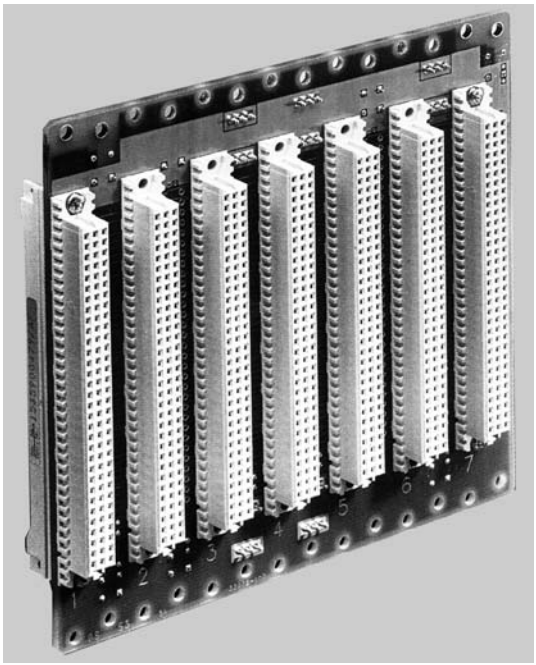


Abb. 1.8 Busplatine (Schroff)

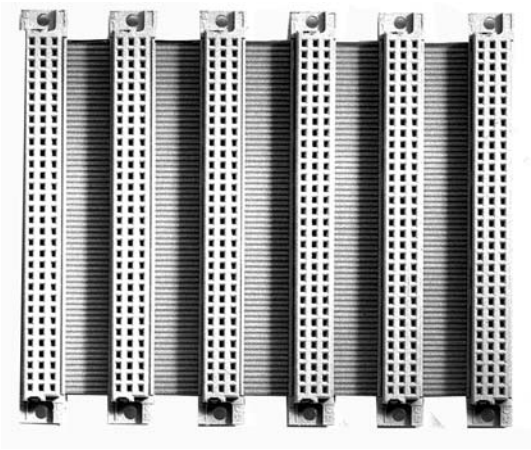


Abb. 1.9 Busverdrahtung über Flachbandkabel (Schroff). Eine „ideale" Busstruktur im Sinne von Abbildung 1.1 macht auch eine solche Lösung möglich. Die über Flachbandkabel untereinander verbundenen Steckverbinder können anstelle einer Busplatine in einen Einbaurahmen montiert werden.

1.2 Grundlagen der Busverbindungen

Ein wesentliches Kennzeichen von Bussystemen ist die Anschaltung mehrerer Einrichtungen an gemeinsam genutzte Verbindungsleitungen (Abb. 1.10). Dafür haben sich zwei Prinzipien durchgesetzt: (1) das Open-Collector-Prinzip und (2) das Tri-State-Prinzip. Die Namen gehen auf Einzelheiten der Schaltungstechnik zurück.

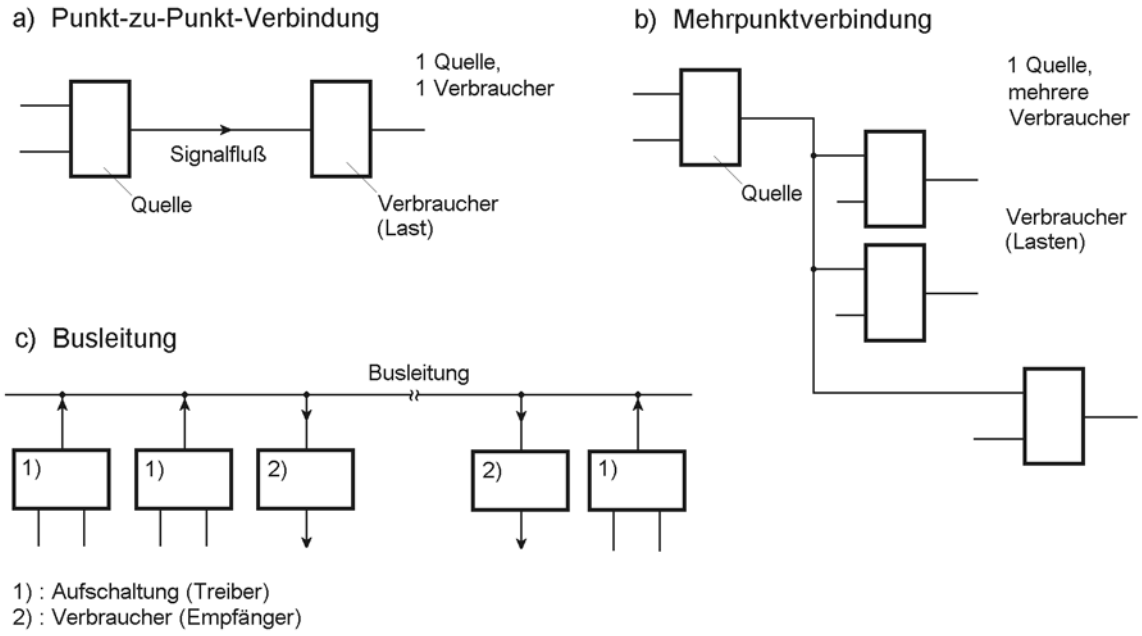


Abb. 1.10 Einzweckverbindungen und Busleitungen

Die Grundsatzfrage: wie bekommt man es hin, daß – in einer Anordnung gemäß Abb. 1.10c – jede Einrichtung die Busleitung belegen kann?

Wenn eine Einrichtung ein binäres Signal auf eine Busleitung aufschalten will, so muß sie diese Leitung entweder auf Low- oder auf High-Potential bringen. Hierfür gibt es zwei Möglichkeiten, die wir uns zunächst anhand von Modellvorstellungen klarmachen wollen (Abb. 1.11):

a) die Leitung führt im „Freizustand“ (es ist keine der angeschlossenen Einrichtungen aktiv) eines der beiden Potentiale, beispielsweise High. Will eine Einrichtung High aufschalten, so tut sie gar nichts. Will sie hingegen Low aufschalten, so stellt sie über einen Schalter eine Verbindung zum Low-Potential (das bedeutet praktisch: zur Masse) her. Würde man als High unmittelbar die Versorgungsspannung (V_{CC}) verwenden, so wäre eine Direktverbindung zwischen High (V_{CC}) und Low (Masse, GND) aus elektrischer Sicht ein Kurzschluß. Man legt deshalb die Busleitung über einen Arbeitswiderstand an die Speisespannung, der bei Durchschaltung zu Low den fließenden Strom auf einen vertretbaren Wert begrenzt. Dies ist das Modell des Open-Collector- bzw. Open-Drain-Prinzips.

b) die Leitung führt im Freizustand (dann, wenn keine der angeschlossenen Einrichtungen aktiv ist) gar kein Signal, sie hängt also elektrisch gleichsam in der Luft. Jede Einrichtung, die die Leitung erregen kann, hat eine Art Wechselschalter, mit dem die Leitung wahlweise auf High- oder auf Low-Potential geschaltet werden kann. Dies ist das Modell des Tri-State-Prinzips.

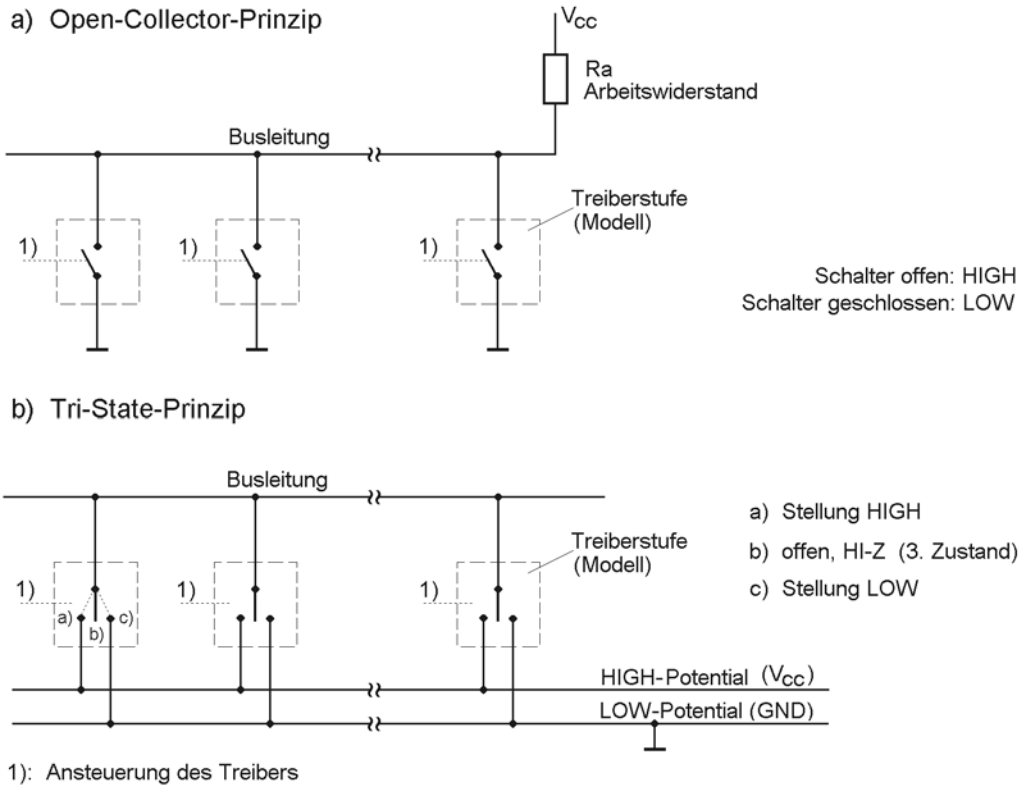


Abb. 1.11 Prinzipien der Busanschaltung (Modellvorstellungen)

1.2.1 Das Open-Collector- bzw. Open-Drain-Prinzip

Abb. 1.12 veranschaulicht eine Busleitung, die gemäß diesem Prinzip beschaltet ist. Die Busleitung selbst ist über einen Arbeitswiderstand (Pull-up-Widerstand) R_a an die Betriebsspannung V_{CC} angeschlossen. Jeder Bustreiber hat einen Transistor, dessen Kollektor mit der Busleitung verbunden ist

(der Name des Prinzips rührt eben daher; der Kollektor des Treiber-Transistors ist ohne weitere Beschaltung auf den Schaltkreisanschluß geführt, liegt also praktisch „offen"). Wird der Transistor aktiviert (durchgesteuert)[1], so zieht er die Busleitung auf Low-Potential.

Open Collector und Open Drain
Beide Begriffe bezeichnen das gleiche Prinzip. Open Collector bezieht sich auf bipolare Transistoren, Open Drain auf Feldeffekttransistoren. Im folgenden werden wir, der Kürze halber, zumeist nur von Open Collector sprechen (meinen damit aber beide Ausführungen).

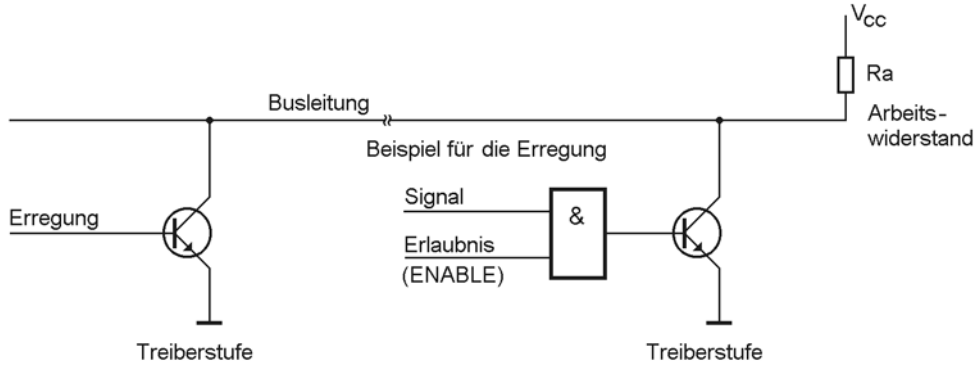


Abb. 1.12 Open-Collector-Prinzip

Was geschieht, wenn mehrere Treiber gleichzeitig aktiv sind? Betrachten Sie die Abbildungen 1.11a und 1.12 genauer: eigentlich nichts! Auch mehrere aktive Bustreiber können nicht mehr tun, als die Busleitung auf Low zu ziehen. Bezüglich des Low-Potentials verhält sich also die Busleitung wie eine ODER-Verknüpfung (bei positiver Logik wie ein NOR – es genügt eine Eins an der Basis eines der Transistoren, um die Busleitung auf Null zu ziehen). Deshalb ist auch die Bezeichnung „Wired OR" (also „verdrahtetes ODER") üblich.

Beim Open-Collector-Prinzip setzen sich die aktiven Bustreiber gleichsam gegen den Arbeitswiderstand durch („Low gewinnt"). Es schadet deshalb nichts, wenn mehrere Einrichtungen gleichzeitig aufschalten.

Hinweis: Es wäre auch möglich, den Arbeitswiderstand R_a mit Masse zu verbinden und mit den Treiber-Transistoren die Busleitung auf High-Pegel zu ziehen. Der Arbeitswiderstand wäre dann an die Emitter der Transistoren angeschlossen (Open-Emitter-Prinzip). Das wurde auch tatsächlich angewendet, nämlich im Rahmen der ECL-Technologie. Bei den allgemein gängigen Logikbaureihen sind die elektrischen Verhältnisse aber so, daß dies mit erheblichen Nachteilen verbunden wäre. Deshalb ist das „auf-Low-Ziehen" allgemein üblich; Open-Collector-Busleitungen wirken somit oftmals *aktiv Low* (mit anderen Worten: sie führen negierte (invertierte) Signale).

1.2.2 Das Tri-State-Prinzip

Abbildung 1.13 veranschaulicht eine Busleitung, die gemäß dem Tri-State-Prinzip beschaltet ist. Der Wechselschalter von Abbildung 1.11b ist eine Anordnung aus zwei Transistoren, die jeweils alternativ angesteuert werden. Es gibt insgesamt drei Zustände (daher der Name):

1. der untere Transistor ist durchgesteuert und zieht somit die Busleitung auf Low-Potential,
2. der obere Transistor ist durchgesteuert und zieht somit die Busleitung auf High-Potential,

---

1 In der Abbildung zum einen (links) durch direkte Erregung und zum anderen (rechts) durch konjunktive Verknüpfung des Datensignals mit einem Erlaubnissignal.

3. beide Transistoren sind gesperrt; damit ist die Busleitung gleichsam freigegeben. Aus elektrischer Sicht ist sie „hochohmig" (dieser Sachverhalt wird in der Literatur oft mit „tristated", „HI-Z", „high impedance" oder „Z = 4" bezeichnet).

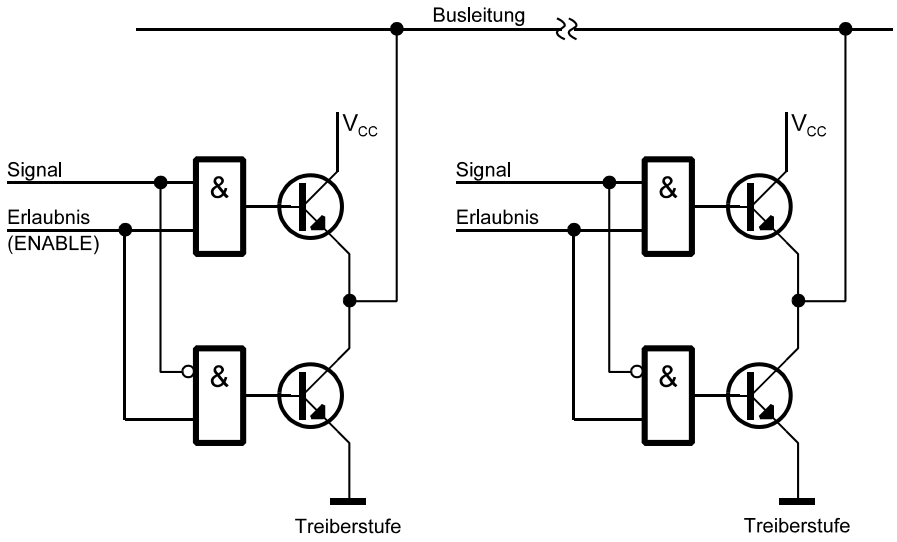


Abb. 1.13 Tri-State-Prinzip

Eine Tri-State-Treiberstufe hat zwei Eingänge:

1. das eigentliche (aufzuschaltende) Datensignal,
2. ein Aufschalterlaubnissignal (Enable-Signal). Ist dieses Signal inaktiv, so befindet sich die Stufe ausganggsseitig im hochohmigen Zustand, ist es aktiv, wird die Datensignalbelegung zur Busleitung durchgeschaltet (Tabelle 1.1).

Erlaubnissignal	Datensignal	Busleitung
0 (inaktiv)	0	4 (hochohmig)
0 (inaktiv)	1	4 (hochohmig)
1 (aktiv)	0	0 (folgt dem Datensignal)
1 (aktiv)	1	1 (folgt dem Datensignal)

Tabelle 1.1 Zur Wirkungsweise einer Tri-State-Stufe

Was geschieht, wenn mehrere Treiber gleichzeitig aktiv sind? Betrachten Sie die Abbildungen 1.11b und 1.13 genauer: das darf einfach nicht vorkommen! Liefern alle aktiven Treiber das gleiche Ausgangspotential (Low oder High), so schadet es nichts, aber wehe, wenn einer auf Low und einer auf High geschaltet ist: Dieser Betriebsfall bedeutet nämlich einen Kurzschluß von V_{CC} nach Masse (Abb. 1.14).

Auf eine nach dem Tri-State-Prinzip ausgelegte Busleitung darf zu einer Zeit nur eine einzige Einrichtung aufschalten. Ansonsten gibt es einen sog. Buskonflikt.

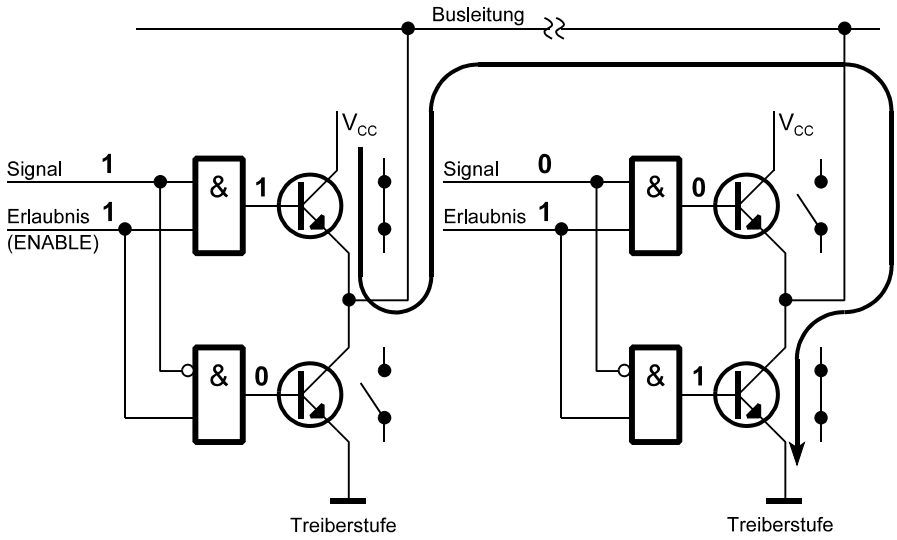


Abb. 1.14 So entsteht ein Buskonflikt ...

1.2.3 Konflikte auf Busleitungen (Bus Contention)

Daß mehrere Bustreiber gleichzeitig aktiv sind und die Busleitung treiben, bezeichnet man in der Fachsprache als Bus Contention (recht zutreffend; Contention bedeutet wörtlich soviel wie Streit oder Zank). Als deutschsprachigen Begriff wollen wir „Buskonflikt", für den Spezialfall des Umschaltens auch „Überschneidung" verwenden.

Bus Contention und Open Collector

Bei Open-Collector-Bustreibern gibt es keine Buskonflikte (Abschnitt 1.2.1.). Busleitungen, bei denen es funktionell notwendig ist, daß mehrere Treiber gleichzeitig aktiv sein können (Wired OR), *muß* man deshalb gemäß dem Open-Collector-Prinzip auslegen. Beispiele: Anforderungs- und Fehlersignalleitungen (Abschnitt 1.4.7.).

Bus Contention und Tri State

Es darf jeweils nur eine Tri-State-Stufe den Bus treiben; mehrere Treiber dürfen nie gleichzeitig aktiv sein. Auch dürfen sich die Aktivierungen mehrerer Treiber am Bus nicht überschneiden (es darf nicht sein, daß ein Treiber aufschaltet, während ein anderer noch nicht in den „hochohmigen" Zustand zurückgeschaltet hat; Abb. 1.15).

Der Konfliktfall ergibt sich, wenn der eine Schaltkreis noch aufgeschaltet hat, der andere Schaltkreis aber schon aufschaltet (Abschalten langsamer als Zuschalten; Abschaltzeit > Aufschaltzeit). Die Dauer des Konfliktfalls entspricht der Differenz von Abschaltzeit und Aufschaltzeit (Konfliktdauer = Abschaltzeit - Aufschaltzeit). Beispiel: Abschaltzeit: 10 ns, Aufschaltzeit: 4 ns, Konfliktdauer = 10 - 4 = 6 ns.

Wann können solche Konflikte vorkommen?
Sehen wir von groben Fehlern aller Art ab: genau dann, wenn es schnell gehen soll. Beispiel: es wird eine neue Speicheradresse geliefert. Die betrifft aber einen anderen Schaltkreis. Also muß der zur Zeit ausgewählte Speicherschaltkreis deaktiviert und der andere aktiviert werden. Wird das auf die einfachste und schnellste Weise erledigt, so kann es zu Überschneidungen kommen.

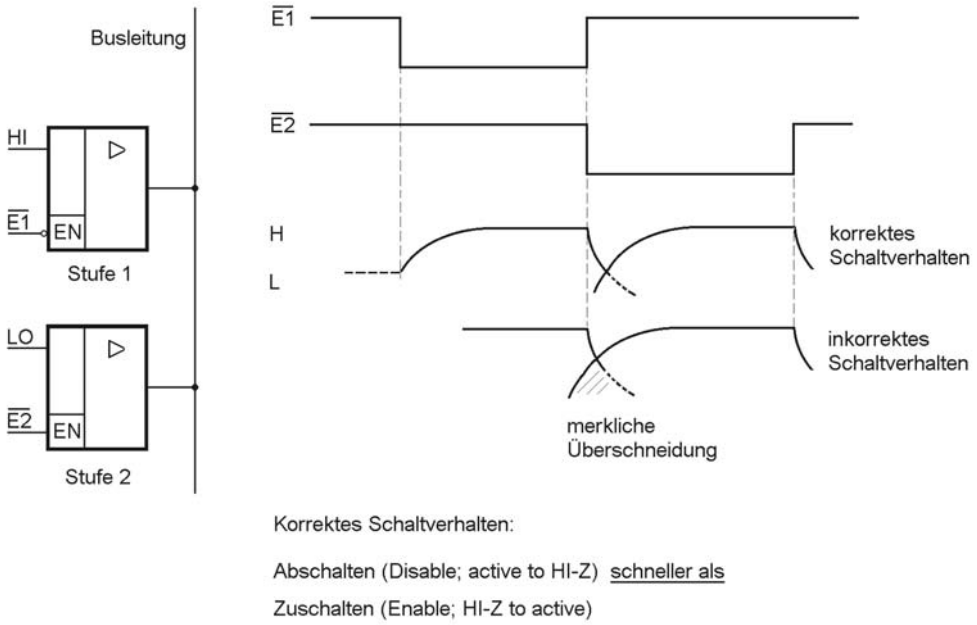


Abb. 1.15 Überschneidung beim Umschalten zwischen zwei Busbelegungen. HI, LO - aufzuschaltende Signalbelegungen; E1, E2 - Aufschalterlaubnissignale (Enable-Signale). Ist ein Aufschalterlaubnissignal aktiv (hier: Low), so wird der betreffende Bustreiber gleichsam scharf und schaltet die jeweilige Signalbelegung zur Busleitung durch.

Sind Buskonflikte auszuhalten?

Es kommt darauf an. Genauer: es hängt von der Auslegung der Schaltkreise, von den fließenden Strömen und von der Dauer des Konfliktes ab.

Herkömmliche (weniger vornehm: altmodische) Tri-State-Schaltkreise (DIL-Gehäuse, höchstens 8 Ausgänge, Treibfähigkeit um die 20 mA) halten auch einen Dauerkurzschluß aus (die integrierten Widerstände wirken strombegrenzend, und die Erwärmung infolge des starken Stromflusses wirkt widerstandserhöhend, so daß der Strom wiederum begrenzt wird). *Das ist aber reine Vorsorge für den Notfall* (wenn, beispielsweise durch einen Fehler an anderer Stelle, tatsächlich zwei Bustreiber gleichzeitig aktiv werden, gehen die Schaltkreise nicht gleich kaputt). Eine solche „ausfallsichere" Auslegung ist aber nicht dazu vorgesehen, Bus Contention als sozusagen offiziellen Betriebszustand einzuführen (Stromfluß und übermäßige Erwärmung über längere Zeit verkürzen die Lebensdauer).

Moderne Bustreiber haben hingegen viele Ausgänge (16...36 sind typisch) und eine geradezu extreme Treibfähigkeit (je Ausgang bis zu 60 mA und mehr). Zudem sind sie in kleinen Gehäusen untergebracht. Und hier können sich Überschneidungen wirklich verheerend auswirken - solche Schaltkreise können buchstäblich „den Deckel abheben".

Wie rechnen die Hersteller?
Der Schaltkreis darf nicht zu warm werden; die Kristalltemperatur sollte auf jeden Fall unter 150° C bleiben. Im Konfliktfall ergibt sich ein Temperaturanstieg um 200° je µs, der sich mit 1 mm je µs ausbreitet. Die Transistorstrukturen haben aber Abmessungen im Bereich weniger µm. Daraus ergibt sich, daß Konflikte gerade noch geduldet werden können, sofern sie wesentlich kürzer sind als 1 µs und vergleichsweise selten auftreten. Richtwert: Überschneidungen von ein paar ns (bis hin zu ca. 10...30 ns) schaden nicht, sofern Periodendauer : Konfliktdauer > 10 : 1. Beispiel: Periodendauer = 100 ns, zulässige Konfliktdauer höchstens 10 ns. Beachten Sie aber, daß bei sehr kurzer Periodendauer (= schnellem Bustakt) auch an sich kurzzeitige Konflikte unzulässig sind (da sie zu oft vorkommen). So dürfte bei einer Periodendauer von 10 ns (× 100 MHz) der Konflikt nicht länger als 1 ns dauern.

Kann man solche (kurzzeitigen) Konflikte in Kauf nehmen?
Auch hier kommt es darauf an. Der Schaltkreishersteller sagt nur, daß bei dieser Betriebsweise sein Schaltkreis nicht kaputtgeht; was sonst noch passiert, ist ihm gleichgültig. Konflikte sind aber stets mit Spitzen im Strombedarf verbunden (Kurzschlußströme). Und die wirken sich als Störungen aus (auch wenn sie nur ns dauern ...). Falls diese Störungen innerhalb des Systems nicht schaden und falls die EMV-Anforderungen ohnehin durch andere Maßnahmen erfüllt werden, könnte man mit solchen kurzzeitigen Überschneidungen leben. Andernfalls muß sich der Entwickler schon Mühe geben, auch den kürzesten Konflikt gar nicht erst entstehen zu lassen.

Konfliktvermeidung

Das Prinzip: erst abschalten, dann aufschalten (Abb. 1.16). Im Fach-Englisch: Break before Make. Dies kann sowohl durch Schaltungsmaßnahmen als auch durch entsprechende Auslegung der Signalfolgen am Bus (der Busprotokolle) erreicht werden. Manche - auf den ersten Blick merkwürdig anmutende - Schaltfolgen sind nur eingeführt worden, um Buskonflikte zu vermeiden (indem man z. B. einem Schaltkreis fast einen ganzen Taktzyklus lang Gelegenheit gibt, sich vom Bus zu verabschieden).

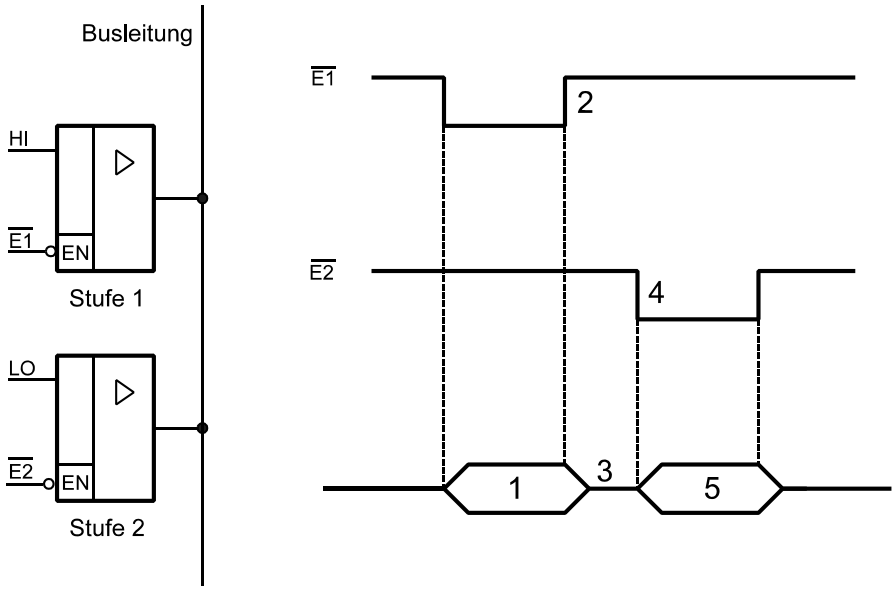


Abb. 1.16 So sollte ein Tri-State-Bus angesteuert werden. 1 - Stufe 1 ist aufgeschaltet; 2 - Stufe 1 wird deaktiviert; 3 - es wird gewartet, bis Stufe 1 den Bus wirklich freigegeben hat (bis also die Busleitung wieder hochohmig geworden ist); 4 - Stufe 2 wird aktiviert; 5 - Busleitung mit Datenbelegung gemäß Stufe 2.

Konfliktvermeidung im Schaltkreis
Manche Schaltkreise gewährleisten von Hause aus, daß es zu keiner Überschneidung kommen kann. Die Voraussetzung dafür: die Ausschaltzeit (Disable Time; Zustandswechsel von aktiv zu hochohmig) muß stets kleiner sein als die Einschaltzeit (Enable Time; von hochohmig zu aktiv). Maßgebliche Kennwerte:

- Abschaltzeit: Turn-off bzw. Disable Time, „Output to High Z",
- Aufschaltzeit: Turn-on bzw. Enable Time, „Output to Low Z".

Meist ist aber nicht zu erkennen, daß derartige Vorkehrungen getroffen wurden (Abb. 1.17).

Symbol	Parameter	71024S12 Min.	Max.
Read Cycle			
tRC	Read Cycle Time	12	—
tAA	Address Access Time	—	12
tACS	Chip Select Access Time	—	12
tCLZ	Chip Select to Output in Low-Z 2	3	—
tCHZ	Chip Deselect to Output in High-Z 1	0	6
tOE	Output Enable to Output Valid	—	6
tOLZ	Output Enable to Output in Low-Z 4	0	—
tOHZ	Output Disable to Output in High-Z 3	0	5

Abb. 1.17 Ein Datenblattauszug (IDT)

Es handelt sich um einen SRAM-Schaltkreis. Uns interessiert das Verhalten des Tri-State-Datenbus. 1 und 3 - Deaktivieren; 2 und 4 - Aktivieren. Schalten wir den Bus über den Chip-Select-Eingang (CS) hochohmig (3), so kann dies bis zu 6 ns dauern (soll heißen: spätestens nach 6 ns ist der Bus frei). Wenn wir aber den Schaltkreis durch Aktivieren von Chip Select auswählen, so belegt er den Bus spätestens nach 3 ns (2). Steuern wir die Datenaufschaltung über Output Enable (OE), so sieht es noch schlimmer aus: beim Deaktivieren (3) kann es bis zu 5 ns dauern, bis der Bus endlich frei wird, beim Aktivieren ist hingegen der Bus sofort (0 ns)[1] belegt. Also dauert das Abschalten sogar länger als das Aufschalten? – Dem ersten Anschein nach ja. Sehen wir aber genauer hin (Pfeil in Abb. 1.17):

- die Abschaltzeiten 1, 3 (High-Z) sind Maximalwerte (soll heißen: es dauert höchstens so lange - kann aber kürzer sein). Maximalwerte gelten für den ungünstigsten Betriebsfall: höchste zulässige Betriebstemperatur und geringste zulässige Betriebsspannung („schlimmer kann es nicht werden").
- die Aufschaltzeiten 2, 4 (Low-Z) sind Minimalwerte (soll heißen: es dauert mindestens so lange, kann aber länger sein). Minimalwerte gelten für den günstigsten Betriebsfall: geringste zulässige Betriebstemperatur und höchste zulässige Betriebsspannung („auch unter besten Bedingungen dauert es garantiert so lange").

Nun werden die gegeneinander kämpfenden Schaltkreise in der Praxis typischerweise unter gleichen Bedingungen betrieben (gleiche Betriebstemperatur, gleiche Versorgungsspannung). Eventuelle Konflikte sind deshalb so kurz (1...2 ns oder weniger), daß sie nicht schaden.

Aber Achtung: das gilt nur, wenn es sich um Schaltkreise gleichen Typs handelt. Ist vorgesehen, an einem solchen Bus Steckkarten, Speichermoduln usw. an sich x-beliebiger Herkunft zu betreiben, so ist Vorsicht geboten.

Konflikte in Fehler- und Sonderfällen

Daß zwei Einrichtungen gleichzeitig den Bus belegen, ist ein vergleichsweise häufiger Fehler, der vielfältige Ursachen haben kann:

---

1 „0 ns" sind eine physikalische Unmöglichkeit. Eine solche Angabe besagt einfach, daß der Hersteller den Wert nicht nachprüft.

Ausfälle der Hardware
Typische Anzeichen: Häufung von Funktionsstörungen, offensichtlich zu warme Schaltkreise. Bei herkömmlicher Hardware könnte man den Fehler im einzelnen suchen und durch Schaltkreistausch beseitigen. Das Auffinden des tatsächlich schuldigen Schaltkreises ist aber ein Kunststück für sich. Auch wirklich gewiefte Praktiker wissen sich oftmals nicht anders zu helfen als auf Verdacht hin zu tauschen.

Inkorrekte Adressen- bzw. Konfigurationseinstellungen
Besonders fehleranfällig: das manuelle Konfigurieren von Steckkarten. Offensichtlich darf es nicht sein, daß zwei Steckkarten den gleichen Adreßbereich zugewiesen bekommen (oder einen Bereich, der bereits anderweitig - z. B. auf dem Motherboard - belegt ist). Grundlage des Handwerks: eine genaue Buchführung über die tatsächliche Konfiguration.

Zu langsame Funktionseinheiten an einem schnellen Bus
An einen schnellen Bus sind Funktionseinheiten mit langsamen Bustreibern angeschlossen. Auch wenn die zeitweiligen Bukonflikte den Schaltkreisen nicht schaden: wir müssen mit *funktionellen* Fehlern rechnen. Nehmen wir an, ein schneller Prozessor arbeitet mit einer E-A-Einrichtung zusammen, die mit ausgesprochen langsamen Bustreibern bestückt ist. Bei einem Lesezugriff zu dieser Einrichtung sind deren Bustreiber aktiv. Ist dieser Zugriff beendet, startet ein schneller Prozessor sofort (z. B. nach 10...20 ns) den nächsten Zyklus. Haben dann die Bustreiber der zuvor angesprochenen Einrichtung den Bus noch nicht freigegeben, stellt sich auf dem Bus eine fehlerhafte Belegung (als „Mischung aus alt und neu") ein.

Ein- und Ausschalten der Betriebsspannung
Betriebsspannungen können nicht in wenigen Nanosekunden ein- oder ausgeschaltet werden. Es handelt sich vielmehr um ein geradezu allmähliches Hochlaufen und Abklingen im Bereich von vielen Millisekunden. Das Problem: in diesen Betriebszuständen können auch jene Schaltungen nicht richtig arbeiten, die an sich vorgesehen sind, Buskonflikte zu vermeiden. Und bei modernen Schaltkreisen können schon Konflikte von wenigen ms ausreichen... Abhilfe: (1) entsprechende Schaltungsauslegung; (2) Vorkehrungen in den Schaltkreisen (um die Ausgänge bei zu niedriger Betriebsspannung hochohmig zu halten).

1.2.4 Open Collector oder Tri State?

Will man bei Tri State Überschneidungen sicher vermeiden, muß man zwischen den einzelnen Buszyklen Lücken lassen (vgl. Abb. 1.16). Das begrenzt die Gesamt-Datenrate (den Durchsatz) des Bussystems. Ist deshalb Open Collector vorzuziehen?

Gibt es bei Tri State wegen der besagten Lücken Leistungsverlust, so bei Open Collector wegen des Arbeitswiderstandes. Wurden die ersten Bussysteme (der 60er und 70er Jahre) vielfach mit Open-Collector-Schaltkreisen verwirklicht, ging man später – insbesondere aus Geschwindigkeitsgründen – auf Tri State über. Derzeit sind die meisten Buskoppelstufen Tri-State-Bauelemente. Nur Busleitungen, bei denen die Wired-Or-Eigenschaft funktionell notwendig ist, werden mit Open-Collector-Koppelstufen angesteuert. Zwischenzeitlich hat man aber, zumindest im obersten Leistungsbereich, die Nachteile der Open-Collector-Ansteuerung durch Weiterentwicklungen der Schaltkreistechnologie überwunden[1], so daß, wenn es um höchste Datenraten geht, dieses Prinzip wieder an Bedeutung gewinnt (Beispiele: Rambus (RSL), Prozessorbussysteme (auf GTL-Grundlage)).Aus Tabelle 1.2 ist das Für und Wider beider Prinzipien ersichtlich.

---

1 Beispiel: der Kollektor (bzw. Drain) wird nicht wirklich offengelassen, sondern mit einem Feldeffekttransistor beschaltet, der als steuerbarer Widerstand betrieben wird und der den eigentlichen Arbeitswiderstand dabei unterstützt, die Signalleitung auf High zu ziehen (AGTL+ (der Bus der modernen Intel-Prozessoren)).

	Open Collector	Tri State
externe Beschaltung	1 Widerstand je Leitung	nicht erforderlich
Strombedarf	groß (niederohmige Dimensionierung für hohe Geschwindigkeit)	vergleichsweise geringer
Geschwindigkeit	nur HI-LO-Flanke an sich schnell; LO-HI-Flanke durch Widerstand bestimmt	beide Flanken gleich schnell
Umschaltung, Überlappung (Buskonflikte)	Überlappung unproblematisch; zwischen zwei Stufen darf daher überlappend umgeschaltet werden	Überlappung (Contention) ist unbedingt zu vermeiden; Umschaltung zwischen zwei Stufen daher langsamer

Tabelle 1.2 Open Collector und Tri State im Vergleich

1.2.5 Der Bus in Ruhe

Was tun, wenn ein Bus nicht genutzt wird? Open-Collector-Signale verharren auf High-Potential; besondere Probleme gibt es hier nicht. Wie verhält sich aber eine Busleitung, die ausschließlich mit Tri-State-Koppelstufen angesteuert wird? Eine solche Leitung hängt gewissermaßen in der Luft. In der Praxis wird sie das jeweils letzte aktive Potential (High oder Low) eine gewisse Zeit noch halten, dann wird sich nach und nach irgendein Zwischenwert einstellen. Eine solche Belegung bezeichnet man als „schwimmendes" (floating) Potential. Sie führt zu folgenden Nachteilen:

- nachgeschaltete Schaltkreise (Empfänger am Bus) werden inkorrekt angesteuert. Es liegt keiner der zulässigen Pegel (Low oder High) an. Das führt insbesondere bei CMOS-Eingängen zu einer übermäßigen Stromaufnahme.
- es ist mit EMV-Problemen (Störstrahlung) zu rechnen.

Die Bussysteme einfacher Mikroprozessorkonfigurationen sind höchstens im Mikrosekundenbereich inaktiv (schließlich muß der Prozessor immer wieder über den Bus auf den Speicher zugreifen, um Befehle zu holen). In einem solchen Fall braucht man nichts zu tun.

Bei Prozessoren mit Caches ist eine pausenlose Aktivität keineswegs mehr zu erwarten, und der Systembus eines Multiprozessorsystems kann unter Umständen lange unbenutzt bleiben (wenn alle Prozessoren mit lokal gespeicherten Daten intensiv rechnen). In solchen Fällen ist es notwendig, durch Zusatzbeschaltung den Bus in einen definierten Zustand zu versetzen. Abb. 1.18 veranschaulicht Problem und Lösungsmöglichkeiten.

Widerstände
Es liegt nahe, ein inaktives Signal mittels Pull-up- oder Pull-down-Widerstand auf High oder auf Low zu ziehen. Manche Bussysteme erfordern ohnehin Widerstände oder Spannungsteiler als Leitungsabschluß.

Widerstände an CMOS-Bussystemen
Man setzt CMOs u. a. deshalb ein, um Strom zu sparen. Durch den Widerstand fließt aber Strom, wenn das Signal auf den jeweils anderen Pegel getrieben wird (z. B. auf Low bei Beschaltung mit Pull-up-Widerstand). Macht man den Widerstand deswegen ausgesprochen hochohmig, so ergeben sich beim Übergang in den hochohmigen Zustand vergleichsweise lange Anstiegszeiten (da die parasitären Kapazitäten nur über den Widerstand umgeladen werden können). Ein Ausweg: das Busprotokoll unterstützt den Widerstand. In einem solchen Fall wird vorgeschrieben, daß vor dem Deaktivieren des

Bustreibers der jeweilige Pegel aktiv einzustellen ist: mit Pull-up-Widerständen beschaltete Signale sind vor dem Deaktivieren auf High zu treiben, mit Pull-down-Widerständen beschaltete Signale auf Low. Der Widerstand hat dann lediglich die Aufgabe, das Signal weiterhin auf dem betreffenden Pegel zu halten. Dieser Trick ermöglicht es, mit hochohmigen Widerständen auszukommen. Beispiel: PCI.

Ein neuerer Trick: die Widerstände werden nur dann angeschaltet, wenn sich das Bussystem in Ruhe befindet. Am arbeitenden Bus sind hingegen keine Widerstände wirksam (somit können auch keine zusätzlichen Ströme fließen). Hierfür gibt es eigens Schaltkreise mit eingebauten Widerständen.

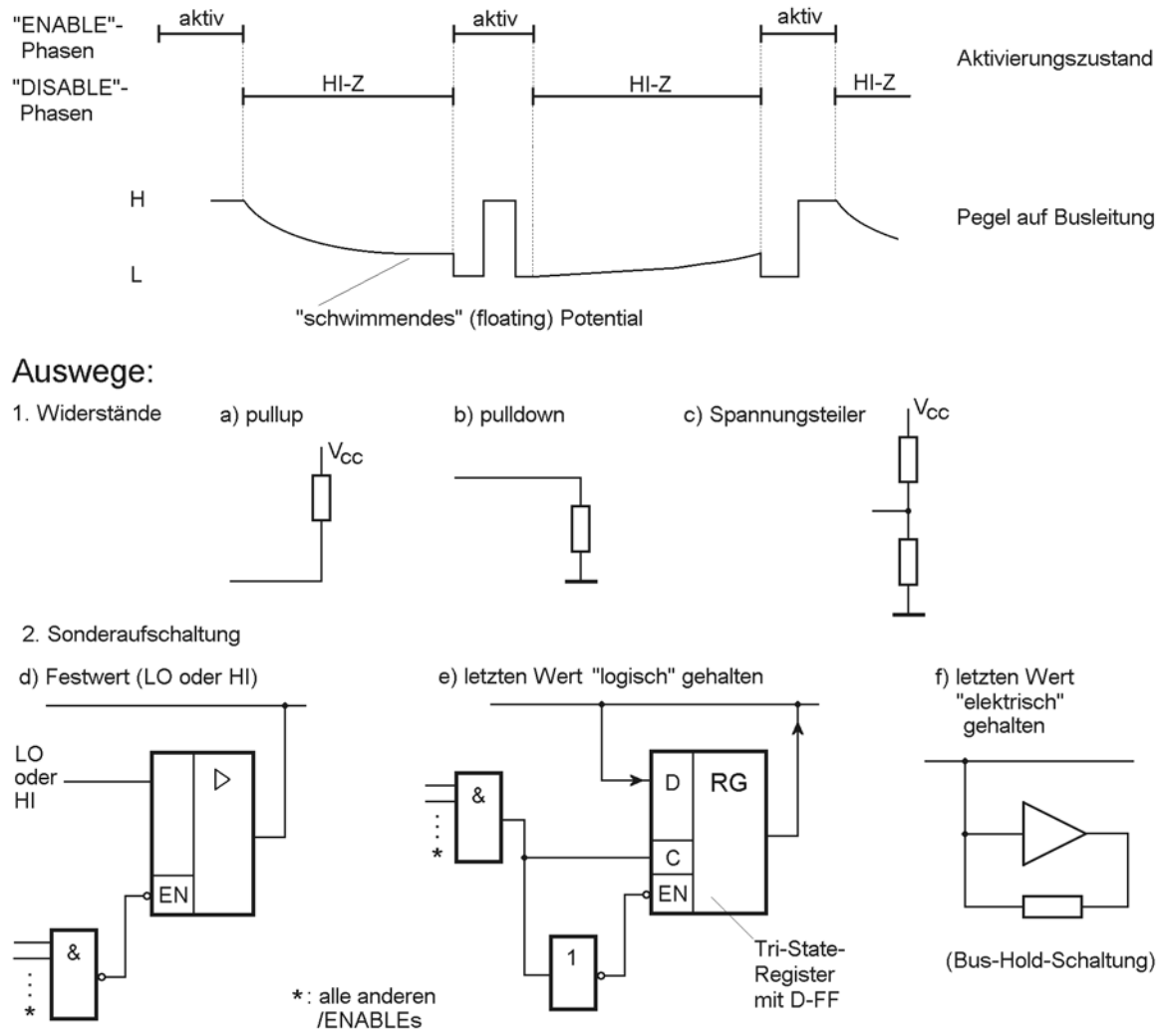


Abb. 1.18 Der Tri-State-Bus in Ruhe: Problem und Lösungen

Aktives Halten (Parken)
Wird der Bus nicht genutzt, so werden die Leitungen mit einem Festwert belegt, oder es wird die jeweils letzte Signalbelegung aktiv gehalten. Beispiel: PCI. Im Ruhezustand muß eine der angeschlossenen Einrichtungen als sog. Park Master wirksam werden.

Halten über Rückführung (Bus-Hold-Schaltung)
Die in Abbildung 1.18f gezeigte Schaltung hält die jeweils letzte Busbelegung (Bus Hold Operation). Viele Schaltkreise haben eingebaute Bushalteschaltungen. Die Vorteile: (1) kein zusätzlicher Strombedarf bei Schaltvorgängen, (2) Busbelegung wird gehalten, ohne daß einschlägige Vorkehrungen in den Busprotokollen erforderlich sind.

Hinweise:

1. Das Halten der jeweils letzten Belegung hat - gegenüber der Belegung mit einem Festwert - den Vorteil, daß Schaltvorgänge entfallen (Stromersparnis, verminderte Störstrahlung).
2. Nicht alle Signale können einfach mit dem letzten Wert belegt bleiben. So müssen Steuersignale, die die einzelnen Buszyklen und Busphasen kennzeichnen, im Ruhezustand inaktiv gehalten werden. (Beispiel: PCI. Die Steuersignale werden über Pull-up-Widerstände inaktiv gehalten, nachdem sie zuvor in den inaktiven Zustand getrieben wurden. Jene Signale, deren Belegung bedeutungslos ist, werden hingegen vom Park Master durch aktives Treiben geparkt.)

1.2.6 Busabschaltung (Quiet Bus Operation)

Hat man mehrere Bussysteme, von denen jeweils nur eines aktiv sein kann, so ergibt sich die Frage, wie man die jeweils unbeschäftigten Bussysteme ansteuern soll. Was also beispielsweise mit einem PCI-Bus tun, wenn gerade zum Arbeitsspeicher zugegriffen wird?

Die herkömmliche Lösung
Datenleitungen, Adreßleitungen usw. der inaktiven Bussysteme werden weiterhin aktiv gelassen. Das heißt, sie schalten gleichsam nebenher mit. Auch die ursprünglichen IBM-PCs waren so ausgelegt. Die Vorteile: (1) geringer Aufwand, (2) Beobachtbarkeit (beispielsweise lassen sich bei einem so aufgebauten PC die Zugriffe zum Speicher auf dem Motherboard über den ISA-Bus beobachten, so daß man dort einen Logikanalysator anschließen oder eine Diagnose-Steckkarte mit Aufzeichnungsfunktionen einsetzen könnte). Dieses Mitschalten funktioniert aber nur dann, wenn alle Bussysteme mit einem gemeinsamen Zeitraster (Takt) arbeiten.

Die moderne Lösung
In modernen PCs wird dafür gesorgt, daß unbenutzte Bussysteme gar nicht erregt werden (Quiet Bus Operation). Die Vorteile:

- geringerer Strombedarf. Die Stromaufnahme der CMOS-Schaltkreise hängt direkt von der Betriebsfrequenz ab. Werden die Eingänge eines solchen Schaltkreises auf festen Pegeln gehalten, so wird nur der - sehr geringe - Ruhestrom aufgenommen.
- Störsicherheit. „Was nicht schwingt, kann auch nicht abstrahlen". Die Energiedichte der hochfrequenten Schwingungen, die jede Impulsschaltung abstrahlt, wird deutlich vermindert. Damit sind die EMV-Anforderungen leichter zu erfüllen.
- jedes Bussystem kann in seinem optimalen Zeitraster betrieben werden (mit anderen Worten: jeder Bus kann eine andere Taktfrequenz haben).

1.2.7 Buskoppelstufen

Koppelstufen, die Busleitungen erregen, also Signale zum Bus senden, heißen *Bustreiber* (Bus Driver bzw. Transmitter) oder kurz Treiber. Koppelstufen, die Busleitungen nachgeschaltet sind, also Signale vom Bus empfangen, heißen *Empfänger* (Receiver). Koppelstufen für beide Signalflußrichtungen (die sowohl senden als auch empfangen können) heißen *bidirektionale* Koppelstufen oder *Transceiver* (Transmitter + Receiver). Es gibt auch universell (wahlweise als Treiber oder Empfänger) einsetzbare Koppelstufen. Abb. 1.19 gibt einen Überblick über die verschiedenen Arten und Ausführungsformen von Buskoppelstufen.

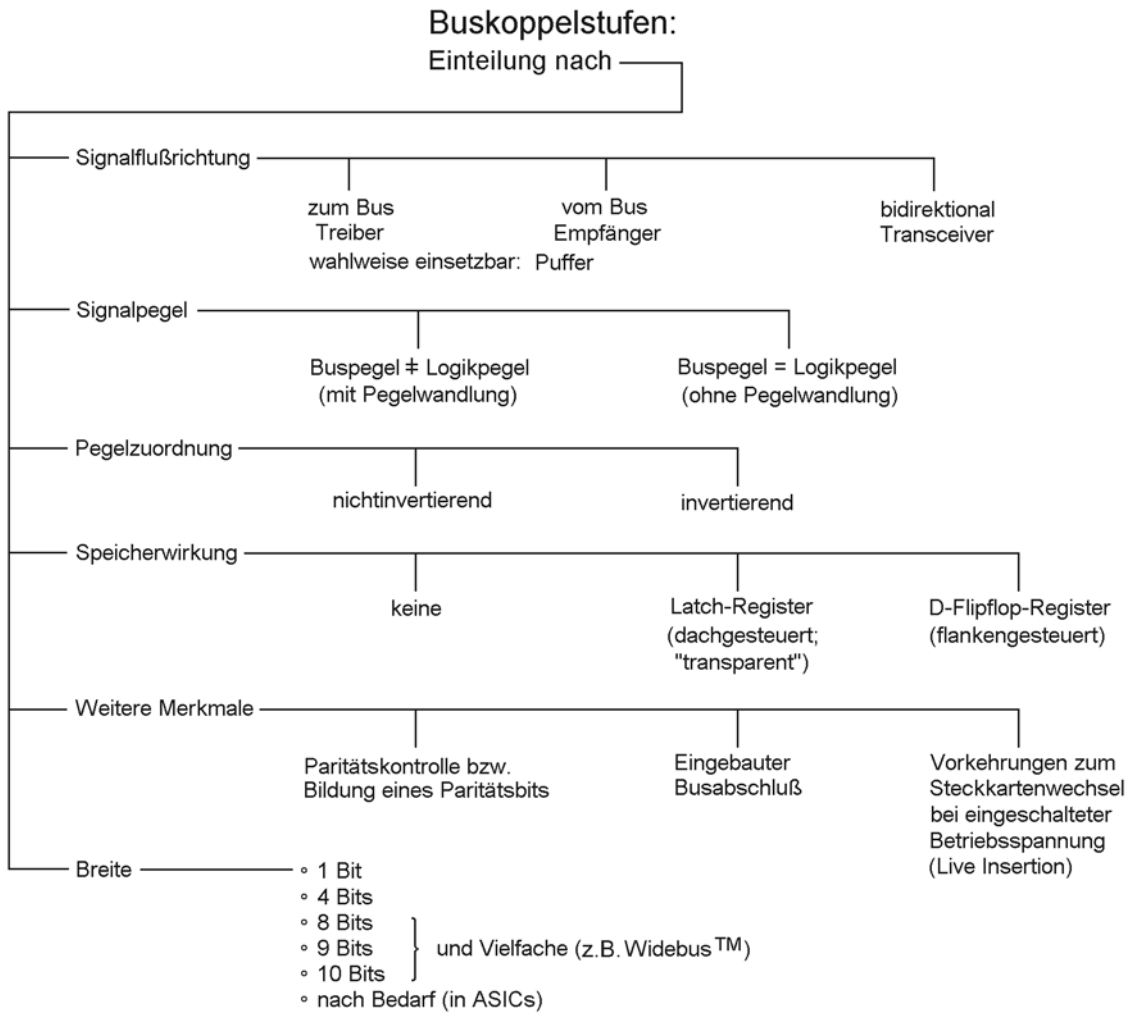


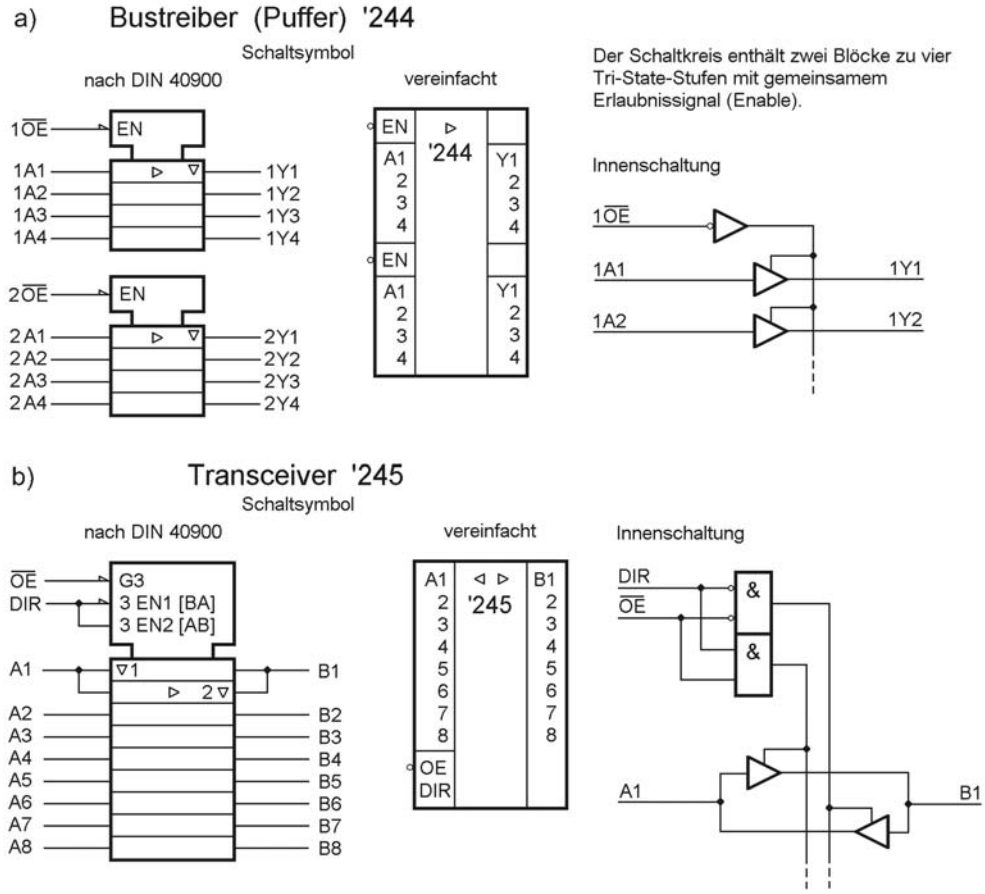
Abb. 1.19 Buskoppelstufen: eine Übersicht

Buskoppelschaltkreise
Buskoppelstufen sind heutzutage meist in hochintegrierte Schaltkreise (Prozessoren, Brücken, Verteiler usw.) eingebaut[1)]. Aber nach wie vor werden Buskoppel*schaltkreise* in großen Stückzahlen eingesetzt[2)]. Buskoppelschaltkreise fassen mehrere gleichartige Buskoppelstufen zusammen. Im Laufe der Zeit wurde eine Vielzahl von Schaltkreistypen entwickelt. Sie lassen sich aber auf nur wenige Grundschaltungen zurückführen. Die Hersteller möchten hohe Stückzahlen fertigen. Und die heutige Technologie ist so weit, daß es auf ein paar Gatter oder Flipflops oder Anschlüsse mehr nicht ankommt. Neuere Entwicklungen sind deshalb vorzugsweise als Universalschaltkreise (Universal Bus Transceivers) ausgelegt. Die Abbildungen 1.20 und 1.21 zeigen Beispiele herkömmlicher und moderner Buskoppelschaltkreise.

Anmerkungen:

1) u. a. wurde der PCI-Bus von Grund auf so entworfen – alle einschlägigen Festlegungen wurden so getroffen, daß sie sich mit Koppelstufen erfüllen lassen, die man in den „funktionellen" Schaltkreisen unterbringen kann.
2) Gründe für den Einsatz besonderer Bukoppelschaltkreise:
   - besondere Forderungen an die Treibfähigkeit,

- besondere Forderungen in Hinsicht auf Verfügbarkeit (Ziehen/Stecken von Einrichtungen während des Betriebs (Hot Plugging), Schutz der funktionellen Schaltkreise, vereinfachte Fehlerbeseitigung (ein über den Bus verursachter Ausfall betrifft nur den (kostengünstigen) Koppelschaltkreis, nicht aber die (teuren) funktionellen Schaltkreise),
- Zusammenfügen verschiedener Technologien, Aufbau der Einrichtung aus verschiedenartigen Bauelementen (wie bei kleinen bis mittleren Stückzahlen allgemein üblich).



a) Bustreiber (Puffer). Eine Signalflußrichtung (von A nach Y). A - Dateneingänge; Y - Ausgänge (Busanschlüsse); OE bzw. EN - Erlaubniseingänge (betreffen je 4 Datensignale). Bei aktivem Erlaubnissignal werden die zugehörigen Datensignale zum Bus durchgeschaltet. Ist das Erlaubnissignal inaktiv, sind die zugehörigen Ausgänge (Y) hochohmig.

b) bidirektionale Koppelstufe (Transceiver). Eine von zwei Signalflußrichtungen wählbar. Beide Seiten A, B, haben Tri-State-Anschlüsse. OE - Erlaubniseingang; DIR - Richtungssteuereingang (Direction Control Input). Zur Wirkungsweise siehe Tabelle 1.3.

Abb. 1.20 Herkömmliche Buskoppelschaltkreise (Beispiele)

DIR	OE	Wirkung
0	0	Richtung von B nach A; A-Ausgänge aktiv
0	1	Richtung von B nach A; A-Ausgänge hochohmig
1	0	Richtung von A nach B; B-Ausgänge aktiv
1	1	Richtung von A nach B; B-Ausgänge hochohmig

Tabelle 1.3 Zur Wirkungsweise der Steuersignale des Transceivers ‘245

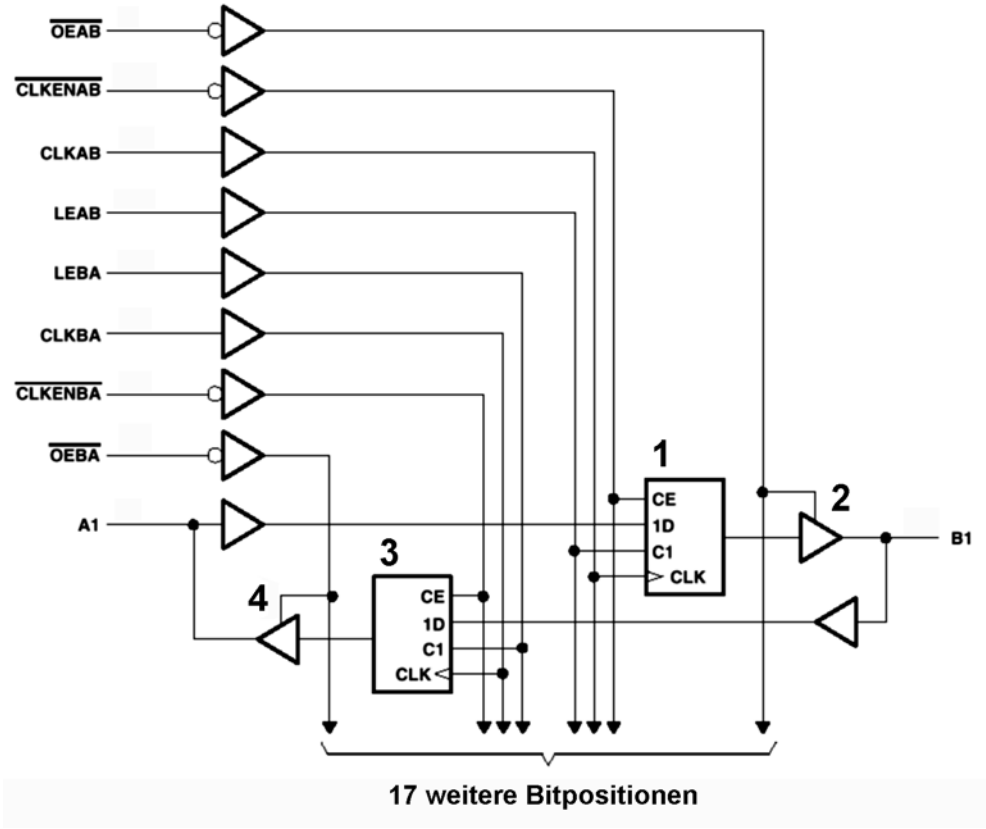


Abb. 1.21 Eine Stufe eines Universal Bus Transceivers (74LVT16601; Texas Instruments)

Ein Schaltkreis des angegebenen Typs enthält insgesamt 18 solcher Stufen. Die Steuereingänge sind allen Stufen gemeinsam. 1 - Datenspeicher für Richtung A - B; 2 - Tri-State-Treiber für B-Anschluß; 3 - Datenspeicher für Richtung B - A; 4 - Tri-State-Treiber für A-Anschluß.

Die Funktionsweise wollen wir zunächst anhand der Signalflußrichtung von links nach rechts (von A nach B) beschreiben (A ist Eingang, B Ausgang). OEAB - Output Enable Richtung A-B; CLKENAB - Clock Enable Richtung A-B; CLKAB - Takt Richtung A-B; LEAB - Latch Enable Richtung A-B. Sinngemäß erklären sich die Steuersignale der Gegenrichtung (B-A).

Aktivieren und Deaktivieren des Ausgangs (B): mittels OEAB (Low: Ausgang aktiv, High: Ausgang hochohmig).

Verhindern, daß der Eingang (A) über den Weg B - A beeinflußt wird: OEBA auf High (deaktiviert Treiber 4).

Steuerung der Betriebsweise des Datenspeichers 1: über LEAB:

- LEAB = Low: Datenspeicher wirkt als D-Flipflop. Übernimmt mit der Low-High-Flanke des Taktes CLKAB die Belegung des A-Eingangs (Registerfunktion). Damit CLKAB wirksam werden kann, muß CLKENAB = Low sein (Übernahmesteuerfunktion).
- LEAB = High: Datenspeicher wirkt als Latch oder als einfache Durchreiche.

Durchreiche (keine Speicherfunktion): Solange LEAB = High ist, wirkt der Datenspeicher als Durchreiche.

Latch-Funktion: Wird LEAB von High auf Low geschaltet, so wirkt der Datenspeicher als Speicherglied und hält die Eingangsbelegung zum Zeitpunkt der High-Low-Flanke von LEAB (Verhalten eines transparenten Latches). In dieser Betriebsart darf CLKAB bei LEAB = Low nicht wirksam werden (sonst: Informationsübernahme mit Low-High-Flanke).

Durch entsprechendes Beschalten der Steuereingänge können viele der gängigen Buskoppelstufen nachgebildet werden. Beispiele:

1. Verhalten ähnlich '244 (einfache Durchreiche mit Tri-State-Ausgängen)
OEAB wirkt als OE. LEAB uund OEBA fest auf High. Restliche Steuersignale auf beliebige Festwerte.

2. Verhalten ähnlich '245 (bidirektionaler Treiber)
OEAB und OEBA wirken als Erlaubnissignale auf der B- bzw. auf der A-Seite. LEAB und LEBA fest auf High. Restliche Steuersignale auf beliebige Festwerte. Wird eine zum '245 kompatible Steuerung mit DIR und OE gewünscht, sind OEAB und OBEA über UND-Gatter gemäß Abb. 1.20b anzusteuern.

3. Einsatz in SDRAM-Speichermoduln mit Registerpufferung
Richtung: von A nach B. Dazu OEAB fest auf Low und OEBA fest auf High. Datenübernahme mittels Takt an Takteingang CLKAB. Übernahmesteuerung muß aktiv sein (dazu CLKENAB fest auf Low). Betriebsartensteuerung über LEAB (mit REGE-Signal des Speichermoduls belegt). Restliche Steuersignale auf beliebige Festwerte.

- Registerfunktion: LEAB = Low,
- Pufferfunktion: LEAB = High.

1.2.8 Punkt-zu-Punkt-Verbindungen in Bussystemen

Manche Bussysteme enthalten neben gemeinsam von allen Funktionseinheiten genutzten, sozusagen echten Busleitungen noch Leitungen, die Punkt-zu-Punkt-Verbindungen oder auch von einer Quelle ausgehende Mehrpunktverbindungen darstellen.

Zentrale Steuerleitungen

In Bussystemen mit zentralisierten Steuerschaltungen (Abschnitt 1.3.1.) werden bestimmte Signale von den zentral angeordneten Schaltmitteln erregt und allen Einrichtungen am Bus zugeleitet. Das betrifft beispielsweise Bustakte und Gültigkeitssignale.

Daisy Chain

„Daisy Chain" ließe sich im Deutschen mit „Kettenschaltung" oder „Durchschleifung" wiedergeben. Es ist zunächst der Allgemeinbegriff für das Weiterreichen von Funktionseinheit zu Funktionseinheit (vgl. Abb. 1.5). Jede Funktionseinheit hat dafür zwei Anschlüsse: einen für das ankommende und einen für das abgehende (weitergeleitete) Signal. Hierbei ist zwischen zwei Anschlußweisen zu unterscheiden:

1. es handelt sich nach wie vor um eine echte Busverbindung zwischen beiden Anschlüssen, die im Gerät nur angezapft wird. Beispiele: DirectRambus), SCSI.
2. es handelt sich um Punkt-zu-Punkt-Verbindungen zwischen jeweils zwei Einrichtungen (die speziellere Bedeutung des Fachbegriffs). Die einzelne Funktionseinheit empfängt das ankommende und bildet das abgehende Signal. Sie kann dabei entscheiden, ob sie ein ankommendes Signal weiterleitet oder nicht (Abb. 1.22). Das wird beispielsweise zu Konfigurations- und Auswahlzwecken verwendet (Zuweisung von Geräteadressen, Bus-Arbitrierung (Abb. 1.25)). Beispiele: IEEE1284 Daisy-Chain-Spezifikation, VME-Bus, das herkömmliche Standard-Interface der IBM-Mainframes.

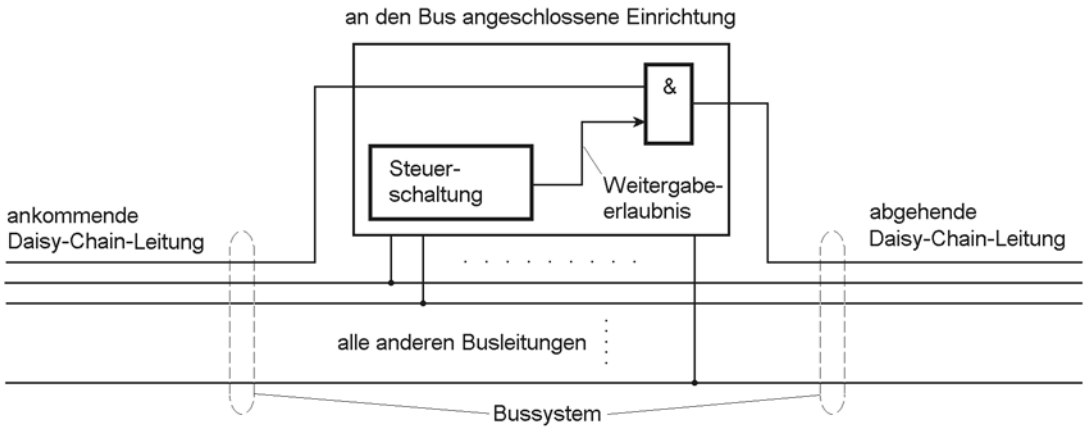


Abb. 1.22 Eine Daisy-Chain-Busleitung

Typischerweise werden nur wenige Signale eines Bussystems (manchmal nur ein einziges) als Daisy-Chain-Signale geführt.

Überbrückung
Wird eine Einrichtung vom Bus entfernt (z. B. eine Steckkarte gezogen), so müssen Daisy-Chain-Signale überbrückt werden.

Handelt es sich um einen Steckkarten-Bus (z. B. um den VME-Bus), so braucht der Servicetechniker eine passende Hilfseinrichtung (Brückenstecker; „blinde" Ersatz-Steckkarte mit Überbrückung[1]). Es gibt aber auch Steckverbinder, die bei gezogener Steckkarte die Überbrückung automatisch (über Kontaktfedern) gewährleisten.

Wird hingegen ein Gerät aus einer Kettenschaltung ähnlich Abb. 1.5 entfernt, so sind die Enden der beiden Kabel miteinander zu verbinden. Manchmal hat man die Steckverbinder so geschickt gewählt, daß es möglich ist, die Kabel einfach zusammenzustecken, manchmal sind Zwischenstecker erforderlich.

Ausgeschaltete Einrichtungen
Ein Problem, wenn die Einrichtungen extern aufgestellte Geräte (mit eigenem Netzanschluß) sind. Die Gerätesteuereinheiten der guten alten Mainframes hatten hierfür eine Relaisschaltung (im stromlosen Zustand wurde der Signalweg durch den Ruhekontakt eines Relais überbrückt).

Daisy Chain als Grundstruktur
Das gesamte Interface beruht darauf, daß jede Einrichtung ankommende Signale empfängt und die zur nächsten Einrichtung führenden Signale treibt. Diese Auslegung ersetzt den Bus im eigentlichen Sinne (Abbildungen 1.1 und 1.10c) durch eine Vielzahl von Punkt-zu-Punkt-Verbindungen. Eine interessante Alternative.

Steckpositionsbezogene Leitungen

Manche Bussysteme haben Signale, die von den Einrichtungen bzw. von den einzelnen Steckplätzen zu zentral angeordneten Schaltmitteln führen (Abbildung 1.23). Beispiel: die Master-Vermittlung (Arbitrierung) des PCI-Bus. Der Vorteil: schnellstmögliche und einfachste Signalisierung ohne besondere Signalspiele bzw. Buszyklen; parallele Auswertung aller von den Einrichtungen kommenden Anforderungen.

---

1 Vgl. auch die Rambus-Blindmoduln (RIMM Continuity Modules).

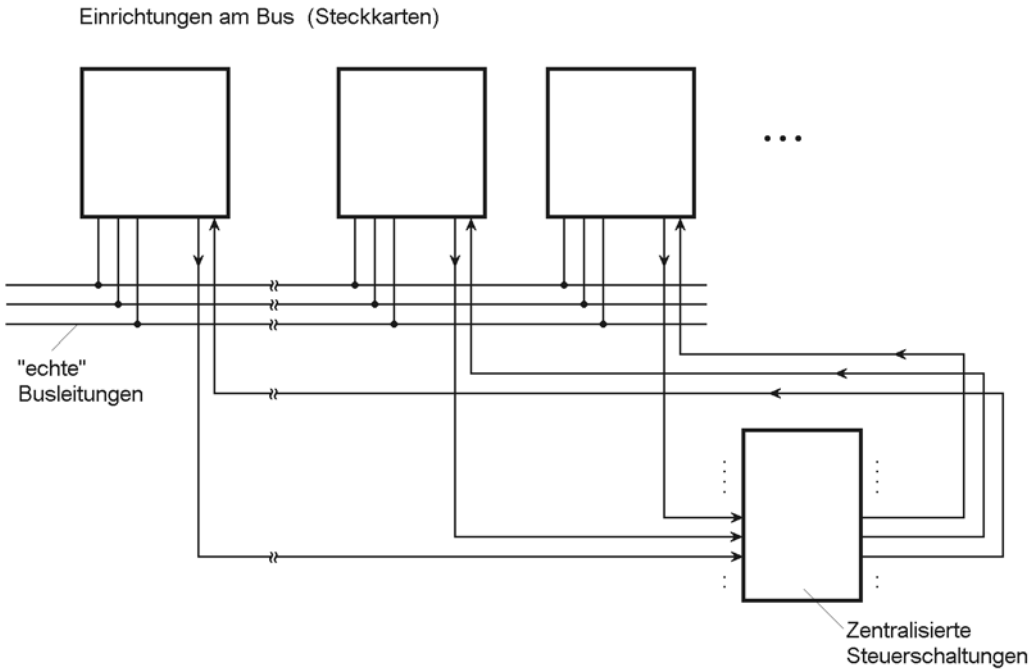


Abb. 1.23 Steckpositionsbezogene Leitungen

Busleitungen als Punkt-zu-Punkt-Verbindungen
Im PC-Bereich hat vor allem der ISA-Bus diese Besonderheit: Leitungen, die auf die Slots geführt sind und wie „echte" Busleitungen aussehen, tatsächlich aber als Punkt-zu-Punkt-Verbindungen betrieben werden. Beim ISA-Bus betrifft dies die Leitungen zur Unterbrechungsanforderung (IRQ) und zur DMA-Steuerung (DRQ, DACK). Wir müssen deshalb beim Konfigurieren der Steckkarten achtgeben: nur jeweils eine Steckkarte darf eine solche Leitung nutzen (zudem müssen DRQ und DACK jeweils einander entsprechen).

1.3 Busstrukturen

1.3.1 Zentralgesteuerte Bussysteme

Ein solches Bussystem (Abb. 1.24) dient nicht zum Verbinden an sich unabhängig arbeitender Funktionseinheiten. Es ist vielmehr eine reguläre Verbindungsstruktur innerhalb einer komplexen Einrichtung. Solche Bussysteme gibt es in Prozessoren, Controllern usw.

Die zentrale Steuerung liefert Aufschaltsignale zu den Bustreibern und (erforderlichenfalls) Übernahmesignale zu den empfangenden Schaltungen (z. B. Taktimpulse zu Registern). Hierbei muß sie dafür sorgen, daß keine Buskonflikte entstehen. Beachten Sie, daß alle Schaltmittel in einem einheitlichen Zeitrahmen arbeiten (bzw. in ein Taktsystem eingebunden sind).

Beispiele:

- Bussysteme in hochintegrierten Schaltkreisen, vor allem in Prozessoren,
- Bussysteme von Mikrocontrollern,
- der X-Bus (auf manchen Motherboards),
- typische Speichersubsysteme.

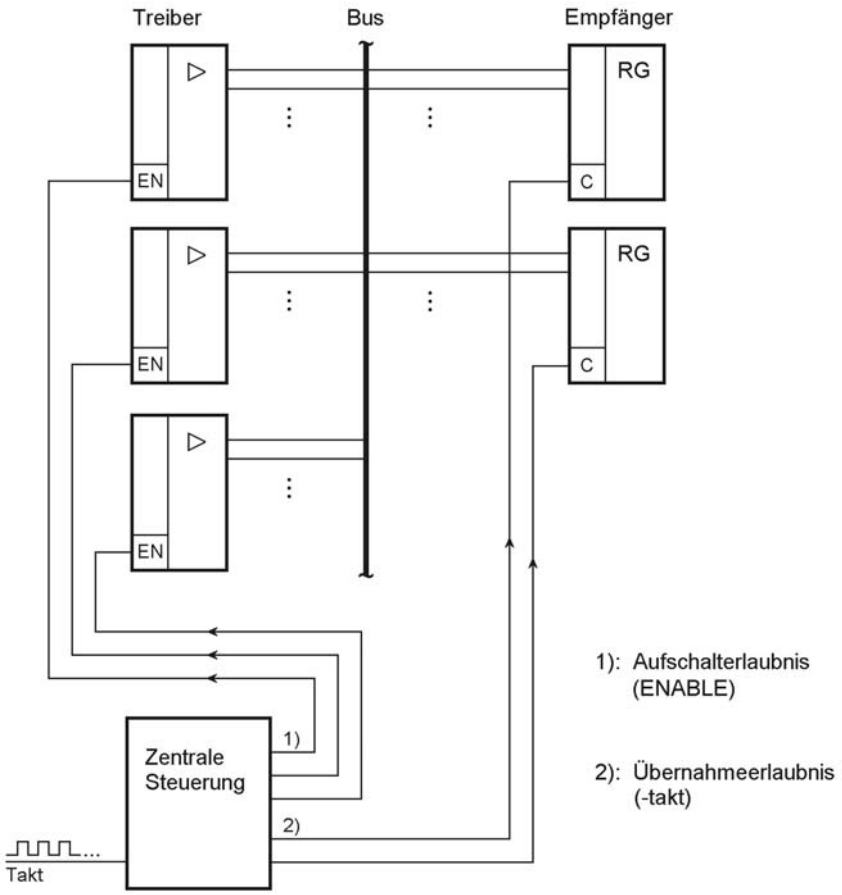


Abb. 1.24 Zentralgesteuertes Bussystem

1.3.2 Single-Master-Bussysteme

In einem solchen Bussystem bestimmt eine einzige Einrichtung (der Master) über die jeweiligen Buszyklen, das heißt, er legt die jeweilige Zugriffsadresse und die Art des Zugriffs (Lesen, Schreiben usw.) fest. Alle anderen Einrichtungen können lediglich als Slaves betrieben werden.

Der Unterschied zum zentralgesteuerten Bus: an den Single-Master-Bus können an sich beliebig komplexe, auch autonom arbeitende Einrichtungen angeschlossen sein, die nicht unbedingt in ein einheitliches Zeitraster bzw. Taktsystem einbezogen sein müssen. Insbesondere kann ein solcher Bus Rückmeldungen vorsehen, so daß Einrichtungen bzw. Funktionseinheiten verschiedener Arbeitsgeschwindigkeit anschließbar sind. Beispiel: die typischen herkömmlichen Mikroprozessor-Bussysteme.

1.3.3 Multi-Master-Bussysteme

Mehrere Einrichtungen sind in der Lage, als Master den Bus zu steuern. Eine davon kann jeweils aktiv sein. Um zu Beginn eines jeden Buszyklus zu bestimmen, welche Funktionseinheit zum steuernden Master wird, braucht man eine „Schiedsrichter"-Funktion (Arbitrierung). In einem wirklich universellen Bussystem kann prinzipiell jede der angeschlossenen Einrichtungen Master werden, d. h. die Kontrolle über den Bus anfordern und zeitweilig übernehmen.

Systeme mit zentralem (primärem) Master

In solchen Systemen hat eine Funktionseinheit eine Sonderstellung, und zwar zumeist sowohl in funktioneller als auch in konstruktiver Hinsicht (z. B. Anordnung auf dem Motherboard). Sie hat als Master unbeschränkten Zugriff auf alle anderen Einrichtungen, kann Sonderfunktionen ausüben (z. B. zwecks Konfigurationssteuerung) und übernimmt die Steuerung der Buszyklen sowie die Master-Auswahl (Arbitrierung). Dies ist die Auslegung der üblichen PC-Bussysteme. Primärer Master ist hier der Prozessor in Verbindung mit speziellen Bussteuerschaltungen.

Systeme mit gleichberechtigten Mastern

Der Bus eines solchen Systems besteht an sich nur aus den Steckverbindern (Slots) und den Leitungen zwischen ihnen (wobei die Verbindungen vorzugsweise als Leiterplatte (Backplane) ausgeführt sind). Eine hervorgehobene Einrichtung gibt es hierbei nicht. (Zwar können den einzelnen Master-Einrichtungen unterschiedliche Prioritäten, Zugriffsrechte usw. zugeordnet sein, aber das ist allein eine Frage der Konfigurations-Einstellung. Grundsätzlich ist ein solcher Bus mit beliebigen Kombinationen von Master-Steckkarten betriebsfähig.) Die meisten Bussysteme für industrielle Anwendungen sind so ausgelegt. Auch das SCSI-Interface wurde ursprünglich so konzipiert (allerdings mit Kabelverbindungen anstelle einer Backplane).

Zentralisierte und dezentrale Bussteuerung

Manche Bussysteme erfordern zentralisierte Schaltmittel zur Ablaufsteuerung, Kontrolle, Takterzeugung, Busvermittlung (Arbitrierung) usw., manche nicht (die Steuermittel sind dann in allen angeschlossenen Einrichtungen vorhanden bzw. über diese verteilt). Zentralisierte Steuerung bedeutet aber nicht unbedingt, daß es einen primären Master im System gibt. Es sind durchaus Bussysteme mit zentralisierten Steuermitteln und gleichberechtigten Master-Einrichtungen denkbar, aber auch solche mit primärem Master und verteilter Steuerung.

Die Bussteuerung in PCs

Man bevorzugt die zentralisierte Steuerung. Die Motherboard-Bauweise legt solche Lösungen ohnehin nahe. Zudem hat diese Auslegung grundsätzliche Vorteile:

- Aufwandsverminderung. Die betreffenden Schaltmittel sind insgesamt nur einmal vorhanden.
- Leistungssteigerung. Entscheidungen sind an einer einzigen, zentralen Stelle schneller zu treffen als in mühseligen Abstimmungsvorgängen. Zudem kann eine zentrale Steuerung *alle* Abläufe übersehen und dementsprechend die Datenflüsse optimieren (z. B. zwischen Prozessor, AGP, Arbeitsspeicher und Peripherie).

Deshalb werden wir uns in den folgenden Erläuterungen auf derartige Prinzipien beschränken.

1.4 Grundlagen der Funktionsweise

1.4.1 Anforderungen an ein universelles Bussystem

Ein universelles Bussystem (mit Multi-Master-Eigenschaften) muß für folgende Funktionen eingerichtet sein:

- Auswahl des aktuellen Masters,
- Auswahl des aktuellen Slaves,
- Steuerung von Funktion und Übertragungsrichtung (Lesen, Schreiben usw.),
- Übertragung von Adressen vom Master zum Slave,
- Übertragen von Daten vom Master zum Slave (beim Schreiben) bzw. vom Slave zum Master (beim Lesen),

- Beenden des Buszyklus,
- erforderlichenfalls Sonderfunktionen, wie Konfigurieren, Prioritätssteuerung, Fehler-signalisierung, Einstellen besonderer Zustände usw.

Um diese Funktionen zu implementieren, stehen jeweils mehrere Prinzipien zur Wahl.

1.4.2 Auswahl des Masters (Arbitrierung)

Eine Einrichtung, die Master werden will, muß die Kontrolle über den Bus (die Busherrschaft) anfordern. Von dieser Anforderung ausgehend muß jeweils eine der anfordernden Einrichtungen diese Kontrolle tatsächlich erhalten; ihr muß der Bus zugesprochen werden. Zugriffskonflikte müssen dabei so gelöst werden, daß das Leistungsvermögen (der Datendurchsatz) des Bussystems möglichst wenig beeinträchtigt wird. Nach der Anordnung der Schaltmittel unterscheiden wir zwischen zentralisierter und dezentraler (verteilter) Busvermittlung bzw. Arbitrierung.

Zentralisierte Bussteuerung
Eine zentrale Funktionseinheit (der Bus-Arbiter) erkennt alle Bus-Anforderungen und teilt den anfordernden Einrichtungen daraufhin den Bus zu.

Dezentrale Bussteuerung
Die Schaltmittel zur Busvermittlung sind auf die einzelnen Einrichtungen verteilt.

Um Bus-Anforderungen zu erkennen und die Busherrschaft zuzuteilen, sind drei verschiedene Prinzipien anwendbar, auch in Kombination untereinander:

- gesteuerte Weiterschaltung (Daisy Chain),
- Abfrage (Polling),
- unabhängige Anforderungen (Independent Requests).

Diese Prinzipien kann man sowohl in zentralisierter als auch in dezentraler Weise verwirklichen.

Gesteuerte Weiterschaltung (Daisy Chain)

Gemäß Abb. 1.25 kann jede Einrichtung durch Belegen der REQUEST-Leitung eine Bus-Anforderung signalisieren (die REQUEST-Leitung ist als Wired OR beschaltet, z. B. mit Open-Collector-Treiberstufen). Erkennt die Bussteuerung, daß REQUEST aktiv ist, so gibt sie ein Signal auf der Auswahlleitung SELECT ab. Reine Slave-Einrichtungen werten SELECT nicht aus. In allen Einrichtungen, die Master werden können, ist SELECT intern aufgetrennt (vgl. Abb. 1.22); es gibt ein ankommendes SELECT und ein weitergegebenes SELECT (SELECT PROPAGATE). Einrichtungen, die keine Anforderungen gestellt haben, leiten SELECT an die jeweils nächste Einrichtung weiter (SELECT IN wird zu SELECT PROPAGATE durchgeschaltet). Trifft der SELECT-Impuls an der ersten Einrichtung ein (von der Bussteuerung aus gesehen), die eine Master-Anforderung gestellt hat, so wird SELECT nicht weitergeleitet (SELECT PROPAGATE bleibt inaktiv). Statt dessen erregt diese Einrichtung die Besetzt-Leitung (BUSY) und zeigt so der zentralen Steuerung an, daß eine Vermittlung stattgefunden hat. Daraufhin kann die betreffende Einrichtung als Master den Bus belegen. Irgendwann während des Buszyklus muß der Master seine Anforderung zurücknehmen. Das Ende des Buszyklus wird durch Deaktivieren der BUSY-Leitung angezeigt.

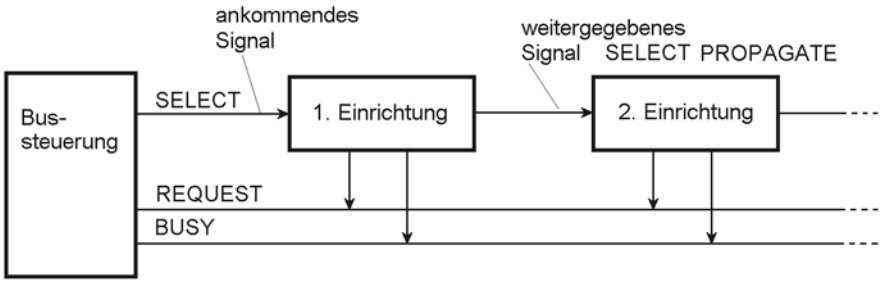


Abb. 1.25 Busvermittlung nach dem Daisy-Chain-Prinzip

Abfrage (Polling)

Die Einrichtungen stellen ihre Anforderungen in der bereits beschriebenen Weise durch Erregen einer REQUEST-Leitung. Dies veranlaßt die zentrale Steuerung, allen Einrichtungen einen Abfragecode (POLL ADRS) anzubieten. Jede Einrichtung hat eine eigene Adresse am Bus, die intern mit dem Abfragecode verglichen wird. Bei Gleichheit wird die BUSY-Leitung aktiviert. Abb. 1.26 veranschaulicht das Prinzip.

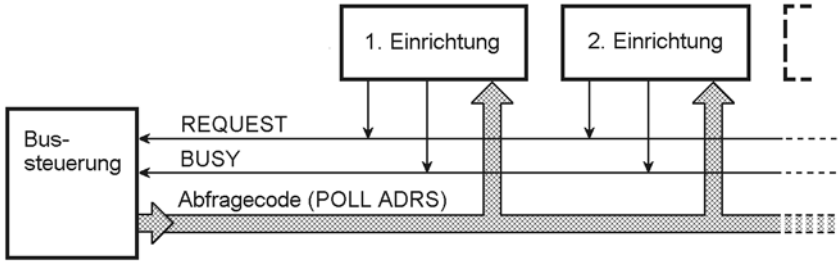


Abb. 1.26 Busvermittlung nach dem Abfrageprinzip

Unabhängige Anforderungen (Independent Requests)

Gemäß Abb. 1.27 hat jede Einrichtung, die Master werden kann, eine gesonderte Anforderungsleitung (REQUEST), die zur zentralen Bussteuerung führt. Dem entspricht je eine Bestätigungsleitung (BUS GRANT). Die Bussteuerung erhält alle Anforderungen gleichzeitig. Aus den vorliegenden Anforderungen bestimmt sie, welche Einrichtung den Bus zugesprochen bekommt. Demgemäß wird die entsprechende BUS GRANT-Leitung aktiviert. Die betreffende Einrichtung antwortet mit Erregung von BUSY, um zu kennzeichnen, daß der Bus von jetzt an besetzt ist[1]. Nach Abfall von BUSY kann der Bus sofort neu vermittelt werden. Beispiele: (1) die Vermittlung der herkömmlichen DMA-Kanäle (DMRQ, DMACK), (2) EISA (MREQ, MACK), (3) PCI (REQ, GNT). (In Klammern stehen jeweils die Bezeichnungen des Anforderungs- und des Bestätigungssignals.)

1 Viele Bussysteme haben keine besonderen Signale, um den Besetztzustand anzuzeigen. Alternativen: (1) entsprechend zeitversetztes Schalten des Anforderungs- und des Bestätigungssignals (DMA, EISA), (2) Beobachten bestimmter Signalspiele, um die Zustandsübergänge zu erkennen (PCI).

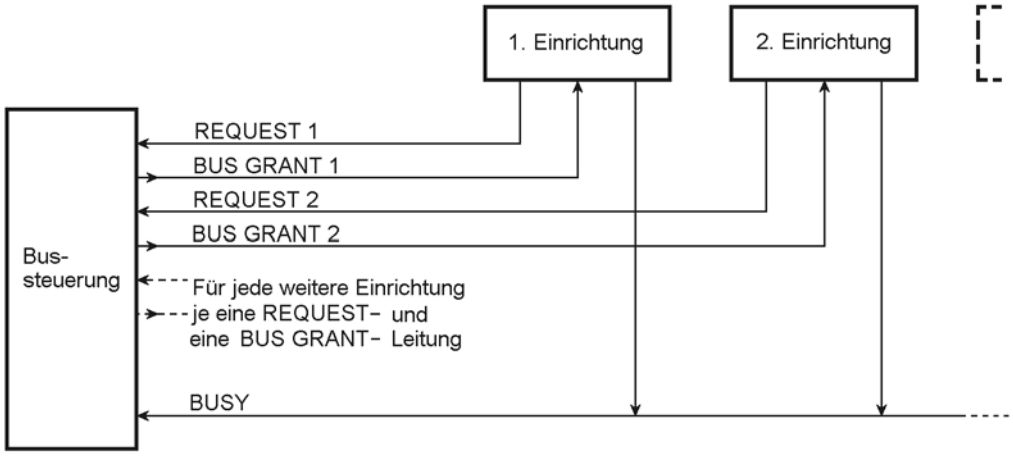


Abb. 1.27 Busvermittlung nach dem Prinzip der unabhängigen Anforderungen

Unabhängige Anforderungen bei dezentraler Vermittlung

Gemäß Abb. 1.28 hat der Bus eine Anforderungsleitung je (Master-) Einrichtung. Jede dieser Einrichtungen treibt ihre eigene Anforderungsleitung und liest alle anderen. Sie kann somit die eigene Priorität mit dem Prioritätszustand des Systems vergleichen. Die Einrichtung, die erkennt, daß ihre eigene Anforderung derzeit die höchste Priorität hat, belegt den Bus durch Erregen von BUS GRANT. Das Ende des Buszyklus wird durch Deaktivieren von BUS GRANT angezeigt, wonach die Vermittlung von neuem beginnt.

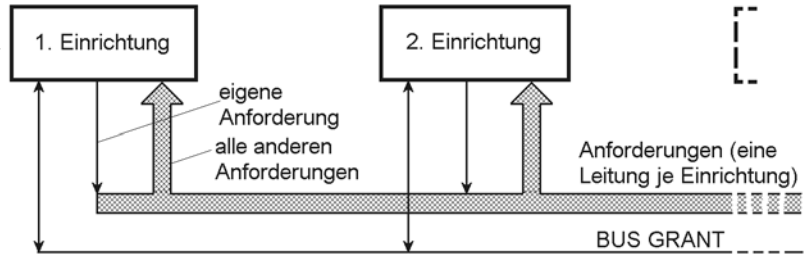


Abb. 1.28 Prinzip der unabhängigen Anforderungen bei dezentraler Bussteuerung

Das Prinzip – jede Einrichtung legt ihre Anforderung auf gemeinsame Leitungen und prüft, ob sie wirklich berechtigt ist, zur Zeit den Bus zu belegen – ist vor allem im Bereich der Peripherie-Interfaces verbreitet (Beispiel: SCSI). Aber auch der FSB der modernen Intel-Prozessoren ist so ausgelegt.

1. Abwandlung: Prioritätscode

Anstelle von n Anforderungsleitungen für n Einrichtungen (1-aus-n-Codierung) sind Leitungen für einen binär codierten *Prioritätscode* vorgesehen. Man braucht dann nur ceil (ld n) Anforderungsleitungen, die Hardware wird aber komplizierter (Beispiel: MCA).

2. Abwandlung: Frei-Besetzt-Signalisierung

Wir begnügen uns mit einer einzigen Leitung. Deren Belegung erlaubt es, zwischen einem Besetzt- und einem Frei-Zustand zu unterscheiden. Finden wir einen Frei-Zustand vor, so versetzen wir den Bus in den Besetzt-Zustand und nehmen uns die Busherrschaft (das geben wir kund, indem wir eine Besetzt-Anzeige auf die Leitung schalten). Erkennen wir aber – anhand der Belegung der Leitung –, daß der Bus bereits besetzt ist, so ziehen wir uns stillschweigend zurück und versuchen es nach einiger Zeit noch einmal (Beispiele: I^2C-Bus, Ethernet).

Ein Grundproblem
Was geschieht, wenn (1) mehrere Einrichtungen zur gleichen Zeit prüfen, ob sie berechtigt sind, im Moment Busmaster zu werden, wenn (2) jede von ihnen befindet, daß der Bus an sich frei ist und wenn deshalb (3) jede für sich beschließt, von nun an die Busherrrschaft auszuüben? Dann haben wir eine Wettlauferscheinung bzw. eine Kollision.

Lösungen:

- zentral definierte Zeitvorgaben (zentrales Timing, Bustakt). Beispiele: MCA, P6-FSB.
- entsprechend ausgeklügelte Vorgaben für das zeitliche Verhalten und für die Leitungslängen. Beispiele: SCSI, Ethernet.
- nach Beanspruchung der Busherrschaft nochmals prüfen, ob es wirklich geklappt hat (Kollisionserkennung). Beispiele: I^2CBus, Ethernet.

Vor- und Nachteile

Daisy Chain
Vorteil: das Prinzip kommt mit einer einzigen Daisy-Chain-Leitung aus, die von Einrichtung zu Einrichtung führt.

Nachteile:

- die Priorität der einzelnen Einrichtungen hängt von deren physischer Anschlußreihenfolge am Bus ab („weiter hinten" (in Abb. 1.25: weiter rechts) angeschlossene Einrichtungen kommen später dran),
- ein Fehler in einer Einrichtung, der die Weitergabe des Daisy-Chain-Signals verhindert, führt zum Ausfall des gesamten Systems,
- die Vermittlungszeit wächst mit der Länge des Bussystems und der Anzahl der angeschlossenen Einrichtungen,
- wird eine Einrichtung vom Bus entfernt, müssen die Daisy-Chain-Anschlüsse überbrückt werden.

Abfrage
Vorteile:

- die Prioritätszuordnung ist auf einfache Weise einstellbar,
- die physische Anschlußreihenfolge hat keinen Einfluß auf die Priorität,
- mit n Leitungen für den Abfragecode (POLL ADRS) kann man bis zu 2^n Einrichtungen abfragen,
- Einrichtungen können ohne weiteres aus dem System entfernt werden.

Nachteil: der Vermittlungsablauf dauert bei n Einrichtungen bis zu n Vermittlungszyklen (im Mittel $^n/_2$). Die Abfrage kann aber parallel zu einem laufenden Buszyklus stattfinden.

Unabhängige Anforderungen
Vorteile:

- maximale Geschwindigkeit = kürzeste Vermittlungszeit (kombinatorische Zuordnung; auch Parallelisierung mit dem laufenden Buszyklus möglich),
- die Prioritätszuordnung ist auf einfache Weise einstellbar,
- die physische Anschlußreihenfolge hat keinen Einfluß auf die Priorität,
- Einrichtungen können ohne weiteres aus dem System entfernt werden.

Nachteil: große Zahl von Leitungen; die Maximalzahl der installierbaren Master-Einrichtungen ist von Anfang an vorgegeben.

Fairneß

Am Multi-Master-Bus ist es wie überall: wenn sich Mehrere um irgend etwas Knappes streiten (hier: um die Busherrschaft), können nicht alle gleichermaßen bedient werden. Es muß also vermittelt werden, und es müssen dabei *Prioritäten* gesetzt werden. Ein Vermittlungsverfahren ist dann „fair", wenn es in einem gewissen Zeitrahmen jeder Einrichtung, die Master werden möchte, auch tatsächlich die Busherrschaft zuspricht. Zentrale Vermittlung über Daisy Chain ist in dieser Hinsicht ausgesprochen unfair, denn die physisch erste Einrichtung kann sich jeden möglichen Buszyklus gleichsam wegschnappen. Man schafft zumeist Abhilfe, indem jene Einrichtungen, die am häufigsten Anforderungen stellen, am Ende des Busses angeordnet werden. Auch bei anderen festen Prioritätszuordnungen geht man sinngemäß vor: die Einrichtung, die den Bus am meisten benötigt, erhält die niedrigste Priorität (bei den PC-Bussystemen ist dies üblicherweise der Prozessor). So wird gewährleistet, daß auch jene Einrichtungen, die vergleichsweise selten Master-Anforderungen stellen, den Bus nach nicht allzu langem Warten zugesprochen bekommen.

Veränderliche (rotierende) Prioritäten

In komplexeren Systemen (z. B. wenn es mehrere Einrichtungen gibt, die den Bus gleich dringend und gleichermaßen häufig benötigen, wie dies etwa in symmetrischen Multiprozessorsystemen der Fall ist) läßt sich Fairneß nur durch genaue Buchführung gewährleisten. So kann die Vermittlungsschaltung auszählen, über wieviele Buszyklen hinweg eine Einrichtung schon vergeblich Anforderungen gestellt hat, und von einer gewissen Anzahl an dieser Einrichtung den Bus bevorzugt vermitteln (ihr also zeitweilig eine höhere Priorität zuweisen). Da hierdurch andere Einrichtungen (mit zuvor höherer Priorität) zurückgestellt werden müssen, bis wiederum eine gewisse Anzahl vergeblicher Anforderungen aufgelaufen ist, ergibt sich der Effekt, daß die Prioritäten zwischen den verschiedenen Einrichtungen gleichsam umlaufen (rotieren). Die einfachste Form dieses Schemas besteht darin, der Einrichtung, die zuletzt als Master aktiv gewesen ist, die niedrigste Priorität zuzuweisen. Damit sind alle potentiellen Master-Einrichtungen praktisch gleichberechtigt; die Buszyklen werden gleichmäßig zwischen ihnen aufgeteilt (zyklisches Weiterschalten, Round-Robin-Strategie). Die Busherrschaft wird in einer festen Reihenfolge zyklisch vergeben. (Z. B. zwischen den Einrichtungen A, B, C, D in der Reihenfolge A 6 B 6 C6 D6 A usw., wobei Einrichtungen, die zur Zeit keine Anforderung gestellt haben, übergangen werden.)

Abbildung 1.29 veranschaulicht ein höherentwickeltes Prioritätsschema. Die einzelnen Anforderungen werden in Gruppen eingeteilt und innerhalb der Gruppen gemäß dem Prinzip der rotierenden Prioritäten vermittelt.

Neuere Busstandards weisen die Verantwortung für die Prioritätsverwaltung gerne den Steuerschaltkreisen zu – sie fordern zwar Fairneß, schreiben aber nichts Konkretes vor und fordern die Entwickler auf, sich etwas einfallen zu lassen.

„Clevere" Vermittlungsalgorithmen

Ein Vermittlungsalgorithmus ist „clever", wenn er nicht nur fair ist, sondern auch die bestmögliche Ausnutzung des Bussystems unterstützt sowie dafür sorgt, daß es keine Verklemmungen (Deadlocks) gibt. Hierzu ist es vor allem wichtig, die Busherrschaft (1) nicht unnötigerweise zu entziehen und sie (2) dann, wenn es sein muß, einer bestimmten Einrichtung vorrangig zuzusprechen.

Begnügen wir uns hier mit einem Beispiel: mit der Auswahl des Park Masters am PCI-Bus. Man nutzt typischerweise die Einrichtung, die zuletzt als Master aktiv gewesen war, als Park Master und beläßt ihr die Busherrschaft, so daß weitere Buszugriffe dieses Masters nicht erst neu vermittelt werden müssen.

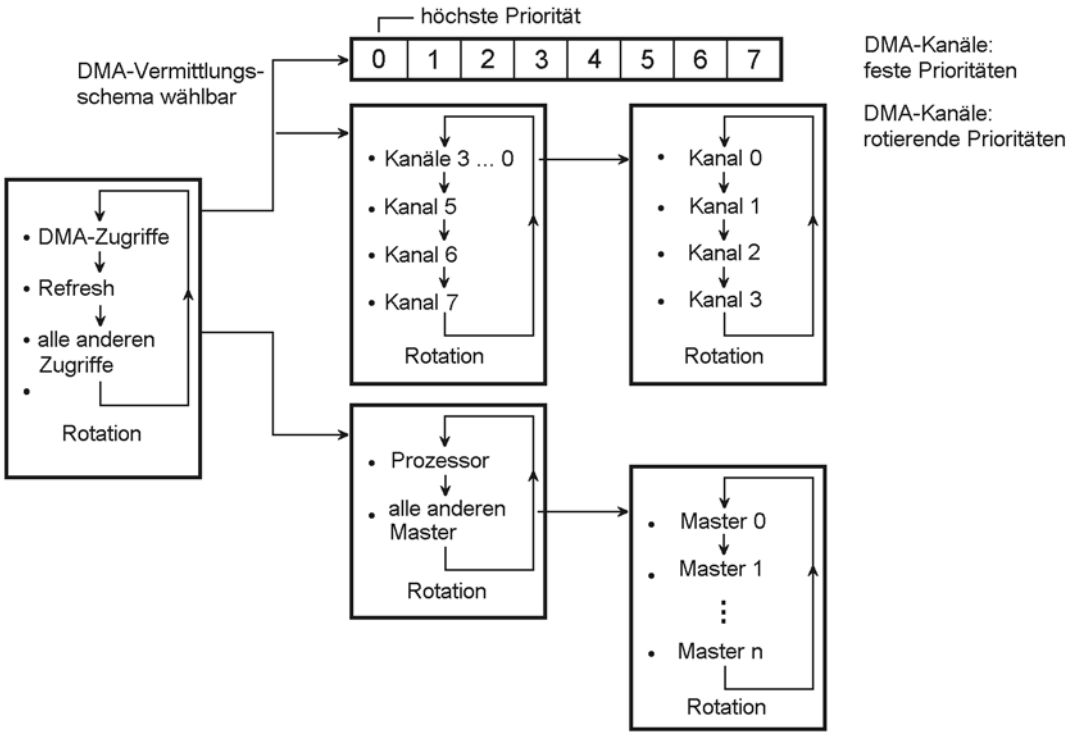


Abb. 1.29 Prioritätsschema auf Grundlage rotierender Prioritäten

Hinweis: Es gibt Schaltkreise, die zusätzliche Einrichtungen zur programmseitigen Optimierung der Busvermittlung enthalten. Beispiel: der sog. Multi Transaction Timer (Intel). Das ist ein Zeitgeber, der bestimmt, wie lange der aktuelle Master den Bus garantiert zugesprochen bekommt. Ist er z. B. auf 32 eingestellt worden, so erhält jeder Master die Busherrschaft für wenigstens 32 PCI-Takte. In dieser Zeit darf er tun, was er will. Der Zweck: der Master bekommt so Gelegenheit, kurze Zugriffsfolgen (Lesen - Schreiben - Lesen usw.) auszuführen, ohne sich für jeden Zugriff erneut um die Busherrschaft bewerben zu müssen (Geschwindigkeitsoptimierung).

1.4.3 Auswahl des Slaves

Hat der Master die Kontrolle über den Bus erhalten, legt er die Zugriffsadresse auf den Bus, um die Slave-Einrichtung auszuwählen.

Die Adressierung beruht auf der Aufteilung des Adreßraums. Einfache Systeme haben eine weitgehend starre Aufteilung, die der Entwickler festlegen muß (z. B. die ersten 64 kBytes als Arbeitsspeicher, die folgenden 64 kBytes als Bildpeicher, dann die RAM-Bereiche auf Steckkarten, dann die BIOS-ROMs usw.).

Zentrale oder verteilte Auswahl

Es wäre möglich, einen zentralen Adreßdecoder vorzusehen, an den die einzelnen Einrichtungen über je ein Erlaubnissignal (Enable) angeschlossen sind – so wie dies in Speichersubsystemen üblich ist. Bei Bussystemen bevorzugt man aber traditionell die dezentrale Auswahl. Das Prinzip: jede Einrichtung erkennt selbst, ob sie im aktuellen Buszyklus als Slave angesprochen wird oder nicht.

Herkömmlicherweise werden einige der höherwertigen Adreßbits zur Slave-Auswahl vorgesehen. Jede Slave-Einrichtung hat einen Adreßdecoder oder einen Vergleicher, der die Gleichheit zwischen den jeweiligen Bits der Zugriffsadresse und der (fest eingestellten) Slave-Busadresse erkennt (Abb. 1.30). Bei Gleichheit nimmt die betreffende Einrichtung als Slave am aktuellen Buszyklus teil.

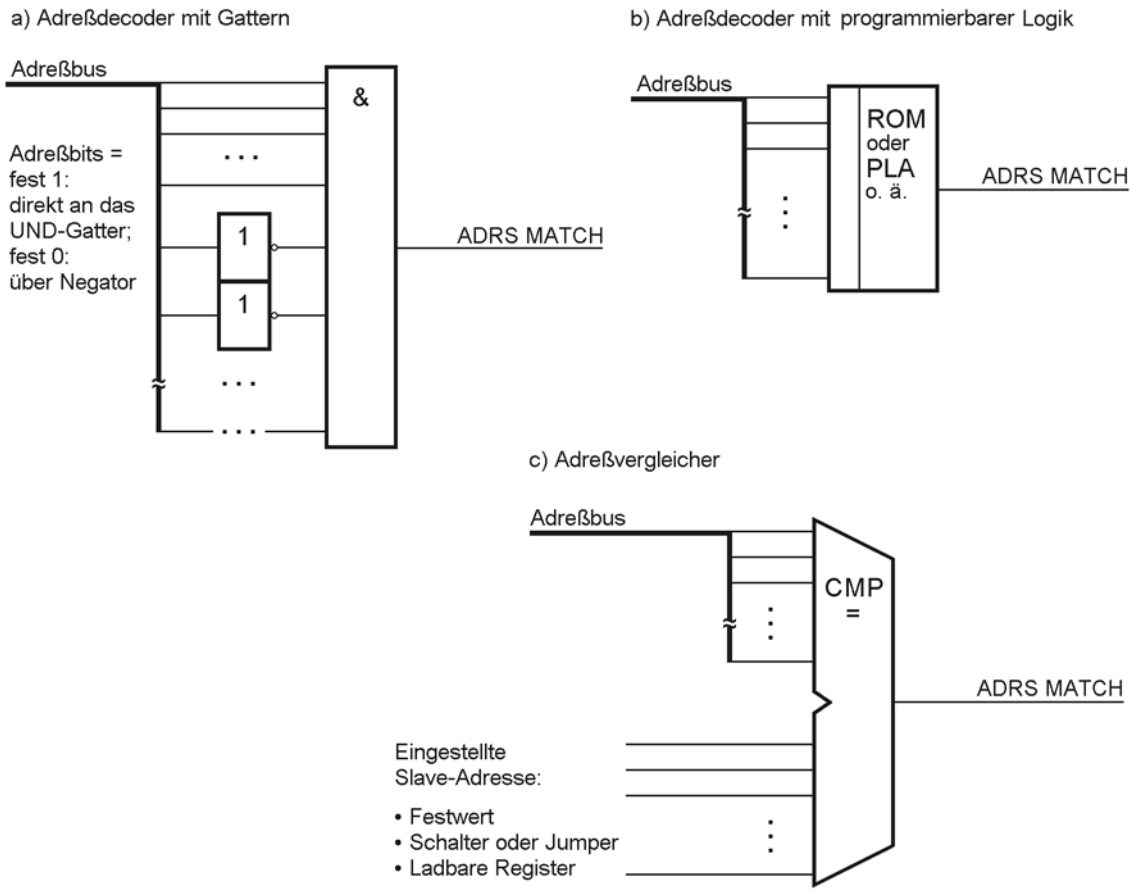


Abb. 1.30 Erkennung der Slave-Adresse

a) einfacher Decoder auf Grundlage eines UND-Gatters. Kann eine feste Adreßbelegung erkennen.
b) Decoder auf Grundlage eines PROMs oder eines programmierbaren Logikschaltkreises. Es können beliebig komplizierte Erkennungsregeln verwirklicht werden (mehrere Adreßbereiche usw.).
c) Adreßerkennung durch Vergleich mit Festwert.

ADRS MATCH wird aktiv, wenn die Einrichtung eine Zugriffsadresse als die ihre erkannt hat.

Die Vorteile gegenüber dem zentralen Adreßdecoder:

- nur so kann man überhaupt ein Busprinzip ohne zentrale Schaltmittel realisieren,
- Narrenfreiheit und Flexibilität. Jede Einrichtung kann für sich entworfen werden, ob sie nur eine einzige Byteadresse benötigt oder einen Bereich von 16 MBytes. Ein zentraler Decoder müßte hingegen an alle vorkommenden Aufteilungen des Adreßraums angepaßt werden (bei den Speichermoduln funktioniert es deshalb, weil eben *nur* Speichermoduln angeschlossen werden – also im Grunde gleichartige Einrichtungen).
- die Adressierung läuft über gleichartige Signalwege; es gibt keine Umwege der Art Adresse - Decoder - Erlaubnisleitung - Einrichtung. Somit haben alle Signale nahezu die gleiche Laufzeit. Deshalb geht man auch bei modernen Speichern von zentral gebildeten Erlaubnissignalen ab und bevorzugt den Adreßvergleich (Multibank-DRAMs, Rambus-DRAMs).

Die Probleme:

- jede Busadresse darf nur einmal vergeben werden. Dabei kann es zu Konflikten kommen.
- Sparlösungen bei der Adreßerkennung führen zu lückenhafter Ausnutzung des Adreßraums oder zu Mehrfachbelegungen (den sog. Alias-Adressen).

Rückmeldung

Viele Bussysteme haben keine. Die betreffende Einrichtung fühlt sich einfach als Slave angesprochen und beteiligt sich so am Buszyklus. Beispiele für Bussysteme *mit* Rückmeldung:

- MCA. Eine sehr komfortable Lösung. Von jeder Einrichtung führt eine Einzelleitung CD SFDBK (Card Selected Feedback) zu den zentralen Steuerschaltungen. Eine Einrichtung, die als Slave adressiert wird, meldet sich über diese Leitung zurück. Die Rückmeldungen sind über ein spezielles Register programmseitig abfragbar.
- PCI. Die jeweilige Slave-Einrichtung (in der PCI-Redeweise: das Target) aktiviert das Signal DEVSEL.

Wenn's schiefgeht

Fehler können u. a. dazu führen, daß sich (1) gar keine Einrichtung angesprochen fühlt, oder daß sich (2) mehrere Einrichtungen gleichzeitig auf den Bus schalten. Manche Fehler kann die Hardware über allgemeine Plausibilitätsprüfungen (vor allem: über Zeitkontrollen) erkennen. Vorkehrungen wie die soeben beschriebenen Rückmeldungen beim MCA ermöglichen es, Testsoftware zu schreiben, die elementare Busfunktionen bis in die Einzelheiten prüft. Grundsätzlich ist aber das genaue Fehlersuchen an einem Bus nicht einfach. Alles kann schuld sein, und da alles an einem Faden (genauer: an den durchgehenden Busleitungen) hängt, ist es schwer, die fehlerhafte Einrichtung aufzufinden.

Adreßeinstellung

Wie wird die (Slave-) Adresse festgelegt? Im Prinzip gibt es 5 Möglichkeiten. Sie sind in Tabelle 1.4 zusammengestellt.

Prinzipien der Adreßaufteilung

Abb. 1.31 veranschaulicht übliche Adreßaufteilungen. In manchen Systemen ist genau vorgeschrieben, welche Adreßbits zur Slave- bzw. Steckkartenauswahl verwendet werden. Dann hat man innerhalb der jeweiligen Steckkarte Narrenfreiheit, d. h. der Entwickler kann den betreffenden Ausschnitt des Adreßraumes so aufteilen, wie er es für zweckmäßig hält. Andere Bussysteme überlassen es dem Steckkarten-Entwickler, sich die Adreßbits, die er decodieren will, selbst herauszusuchen.

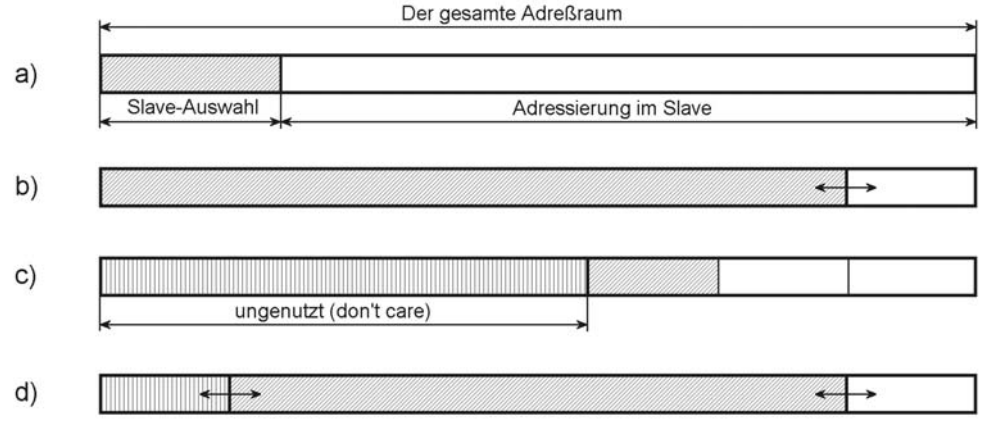


Abb. 1.31 Adreßaufteilungen

a) einige der höchstwertigen Adreßbits sind fest für die Slave-Auswahl vorgesehen - ein einfaches und klares Prinzip, das aber den Adreßraum nicht optimal ausnutzt (so brauchen manche Steckkarten nur einen sehr kleinen Ausschnitt aus dem Adreßraum, manche einen sehr großen; Beispiel: Videoadapter),

b) vom höchstwertigen Adreßbit an werden soviele Bits zur Slave-Auswahl genutzt, wie jeweils verfügbar sind. Beispiel: wir haben eine 24-Bit-Adresse. Die Einrichtung belegt einen Adreßbereich von 8 kBytes. Dann sind ld 8k = 13 Bits zum Adressieren innerhalb der Einrichtung erforderlich. Somit werden 24 - 13 = 11 Bits in den Adreßvergleich einbezogen. Das nutzt den Adreßraum gut aus, erfordert aber höheren Aufwand zur Adreßerkennung.

c) höherwertige Adreßbits bleiben ungenutzt (werden ignoriert); eine feste Zahl von Adreßbits ist zur Slave-Auswahl vorgesehen,

d) je nach Bedarf der jeweiligen Einrichtung wird ein Teil des Adreßraums belegt. Nicht verwendete höchstwertige Bits werden ignoriert.

Ausführung	Änderbarkeit	Vorteile	Nachteile
direkte Verdrahtung (Hard Wired)	nicht änderbar	Einfachheit, Zuverlässigkeit	Probleme bei späterer Erweiterung zu erwarten
Auswahl zwischen mehreren fest vorgegebenen Werten (über Jumper, Schalter o. ä.)	in beschränktem Maße änderbar	flexibler als 1.	keine vollständige Freizügigkeit
gesamter Wertebereich einstellbar (Jumper, DIL-Schalter o. ä.)	beliebig änderbar	hohe Flexibilität	aufwendiger als 2.; Schalter o. dergl. sind Quelle von Unzuverlässigkeit; Buchführung und Handarbeit erforderlich
Realisierung der Adreßdecodierung mit programmierbarer Logik (PROM, PAL o. ä.)	änderbar	hohe Flexibilität bei geringen Kosten (keine Schalter o.dergl.)	meist nicht im System änderbar
gesamter Wertebereich programmseitig ladbar	beliebig änderbar, sogar während des Betriebs	sehr hohe Flexibilität; das automatische Konfigurieren wird auf dieser Grundlage möglich	aufwendig (Bussystem mit Konfigurationsvorkeh-rungen, Konfigurations-software usw.)

Tabelle 1.4 Möglichkeiten zur Einstellung von Slave-Adressen

Alias-Adressen

Alias-Adressen ergeben sich dann, wenn Adreßbits bei der Adreßdecodierung nicht berücksichtigt (ignoriert) werden (Abb. 1.31c, d). Eine so ausgelegte Einrichtung wird auch Adressen erkennen, auf die sie eigentlich nicht reagieren sollte.

Positive und subtraktive Adreßdecodierung

Die Begriffe sind von Bedeutung, wenn es um Systeme geht, in denen mehrere Bussysteme über Brücken gekoppelt sind. Das Problem: manche Adressen werden von Einrichtungen an einem bestimmten Bus genutzt werden, andere hingegen nicht. Es kann aber Einrichtungen an nachgeschalteten Bussystemen geben, die auf diese Adressen ansprechen. Die Brücke muß also derartige Zugriffe weiterreichen. Wir beziehen uns im folgenden – als Lehrbeispiel – auf den PCI-Bus.

Positive Adreßdecodierung
Hiermit bezeichnet man die sozusagen natürliche Form der Slave- bzw. Target-Auswahl: alle Einrichtungen beobachten die Adreßbelegung auf dem Bus und prüfen, ob diese ihren eigenen Adreßbereich betrifft. Ist das der Fall, so „erkennt sich" die betreffende Einrichtung als Target und nimmt demgemäß am Buszugriff teil. Die Adreßerkennung wird durch Erregen der Busleitung DEVSEL angezeigt.

Subtraktive Adreßdecodierung
Eine einzige Einrichtung kann für subtraktive Adreßdecodierung ausgelegt werden. Diese Einrichtung erkennt sich immer dann als adressiert, wenn keine andere Einrichtung die Adreßerkennung gemeldet hat. Mit anderen Worten: der betreffenden (einzigen) Einrichtung wird gleichsam der ungenutzte Rest des Adreßraumes zugewiesen. Es ist üblicherweise eine Brücke, die als einzige Einrichtung für subtraktive Adreßdecodierung ausgelegt ist. Somit können alle Zugriffe auf Adressen, die von den anderen Einrichtungen nicht belegt sind, zum angeschlossenen Bus weitergereicht werden. Eine Einrichtung, die für subtraktive Adreßdecodierung ausgelegt ist, muß DEVSEL beobachten. Wird DEVSEL nicht erregt, so hat keine andere Einrichtung die Adresse erkannt. Somit wird die besagte Einrichtung als Target ausgewählt (sie aktiviert dann ihrerseits DEVSEL, um sich als Target zu erkennen zu geben).

Geographische Adressierung

Daß eine Busadresse an alle Einrichtungen geliefert wird und somit jede Einrichtung für sich entscheiden kann, ob sie als Slave am Buszyklus teilnimmt, ist eine typische Eigenschaft der Bussysteme. Im besonderen ist es gleichgültig, an welchem physischen Anschluß (z. B. in welchem Slot) die Einrichtung angeordnet ist.

Was aber nicht möglich ist:

- herauszufinden, welche Einrichtung sich wo am Bus befindet,
- die gespeicherten Konfigurationsdaten der einzelnen Einrichtungen abzufragen,
- Adreßbereichs- und andere Konfigurationsangaben einzustellen.

Die typische Lösung: eine zusätzliche, alternative Zugriffsweise, wobei die Einrichtungen nicht durch Adreßvergleich, sondern durch direkte Anwahl einer physischen Position (z. B. eines Slots) ausgewählt wird. Diese Betriebsweise wird als "geographische Adressierung" (Geographic Addressing) bezeichnet. Sie ist in allen moderneren PC-Bussystemen (MCA, EISA, PCI) vorgesehen.

Beispiel:
PCI-Einrichtungen haben einen Eingang IDSEL, der ähnlich wirkt wie der Chip-Enable-Eingang eines Speicherschaltkreises. Wird ein Konfigurationszugriff ausgeführt, so wird jene Einrichtung als Slave ausgewählt, deren IDSEL-Eingang aktiv ist. Wie die IDSEL-Eingänge aktiviert werden, ist Sache des Motherboards. Die Vorzugslösung: Nutzung einiger Adreßsignale (AD-Leitungen). Hierbei ist der IDSEL-Eingang jeder Einrichtung an jeweils eine AD-Leitung angeschlossen, z. B. (wie in vielen PCs üblich) Einrichtung 1 an AD16, Einrichtung 2 an AD17 usw. Da die Prozessoren der PCs nur Speicher- und E-A-Zugrifffe ausführen können, nicht aber Konfigurationszugriffe, hat man sog. Konfigurationsmechanismen definiert. Hierbei werden bestimmte E-A-Zugriffe von der Host-to-PCI-Bridge (North Bridge) in Konfigurationszugriffe umgesetzt.

Enumeration

Um herauszufinden, welche Einrichtungen angeschlossen sind, erregt das BIOS nacheinander die einzelnen IDSEL-Eingänge und fragt die gespeicherten Konfigurationsdaten ab. Dieser Vorgang (herauszufinden, welche Einrichtungen im PC überhaupt vorhanden sind) heißt in der (Microsoft-) Fachsprache Enumeration. Jedes Interface hat hierfür seine eigenen Abläufe. Sie bilden die Grundlage der weiteren Konfigurationsvorgänge (Stichwort: Plug and Play).

1.4.4 Funktionsauswahl

Der Master bestimmt die Funktion des aktuellen Buszyklus. Es gibt folgende Auslegungen:

1. besondere Betriebsartensignale. Durch deren Belegung wird der aktuelle Buszyklus eindeutig gekennzeichnet (Schreiben, Lesen, Eingabe, Ausgabe usw.). Zur eigentlichen Ablaufsteuerung ist typischerweise ein Bustakt vorgesehen (Beispiel: PCI).
2. unabhängige Steuersignale für jede Betriebsweise, beispielsweise Strobe-Impulse zum Lesen, zum Schreiben, zur Eingabe, zur Ausgabe usw. (Beispiel: ISA),
3. Kombinationen zwischen 1. und 2.,
4. Übertragung von Kommandos. Diese werden wie Daten übertragen, aber durch besondere Steuerleitungen bzw. Signalfolgen als Kommandobytes gekennzeichnet (Beispiel: SCSI).

1.4.5 Datenübertragung

Um den zeitlichen Ablauf der Datenübertragung zwischen Master und Slave zu steuern, sind besondere Steuerleitungen und Signalspiele vorgesehen. Die typischen Prinzipien werden im folgenden beschrieben.

Zeit- oder taktbezogene Steuerung und Abfrage

Für den Bus ist ein Taktschema (Bustakt) oder ein bestimmtes Zeitraster definiert. Alle Informationsübertragungen werden hierauf bezogen.

Wartezustände
Hierüber wird der Buszugriff an die Arbeitsgeschwindigkeit der ausgewählten Einrichtung angepaßt (Abb. 1.32).

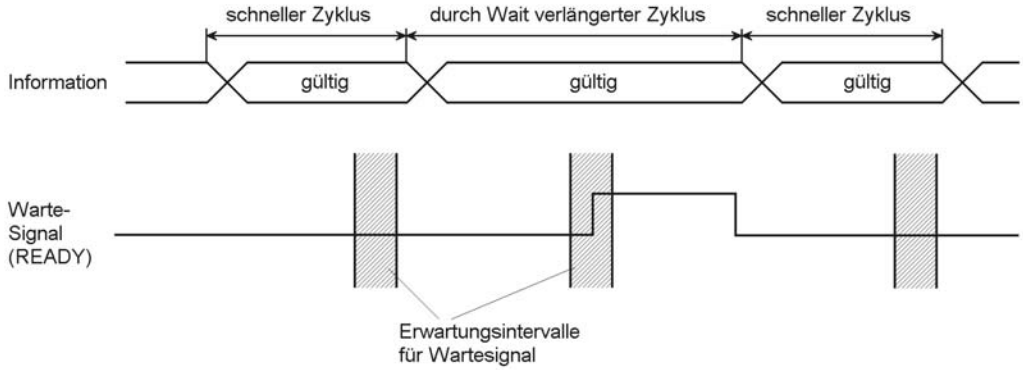


Abb. 1.32 Einfügen von Wartezuständen (Prinzip)

Durch Schalten des Wartesignals kann der einzelne Datenzyklus um weitere Taktperioden (Wartezustände, Wait States) verlängert werden. Zu bestimmten Zeiten wird eine vom Slave anzusteuernde Leitung abgefragt, die üblicherweise mit WAIT oder READY bezeichnet wird. Findet der Master (bzw. die zentrale Bussteuerung) zum Abfragezeitpunkt die READY-Leitung ("Fertigmeldung") als aktiv vor, so wird der laufende Zyklus beendet. Ist hingegen die READY-Leitung zum Abfragezeitpunkt inaktiv, so wird ein Wartezustand eingefügt. In diesem wird READY wiederum abgefragt usw.

Hinweise:

1. Es ist typischerweise nicht notwendig, daß READY impulsförmig erregt wird. Arbeitet die Hardware insgesamt so schnell, daß nie Wartezustände benötigt werden, kann READY auch ständig aktiv gehalten werden.
2. Oft ist das Zeitverhalten (Setup- und Hold-Timing) von READY-Signalen nur zu den Abfragezeitpunkten spezifiziert. Wird ein solches Signal nicht abgefragt, darf es beliebig schalten. Gelegentlich machen die Entwickler von dieser Freiheit Gebrauch.
3. Taktbezogene Steuerung bedeutet nicht zwangsläufig, daß stets ein Bustakt an alle Einrichtungen geliefert wird. In manchen Bussystemen wird kein Bustakt mitgeführt, aber das Schaltverhalten aller Signale ist im Rahmen eines vorgegebenen Taktschemas spezifiziert (das zeitliche Verhalten ist genau vorgegeben), so daß sich der Steckkarten-Entwickler darauf verlassen kann. Beispiel: MCA.

Standardmäßige Wartezustände
Diese Vorkehrung ist beispielsweise am ISA-Bus vorgesehen. Die Bussteuerung fügt, je nach Art des Buszyklus, eine bestimmte Anzahl an Wartezuständen standardmäßig ein. Der Zweck: man möchte den Steckkartenentwicklern die Arbeit erleichtern. Dazu hat man die Buszyklen so gestreckt, daß die zeitlichen Anforderungen mit preisgünstiger Hardware typischerweise erfüllt werden können (mit anderen Worten: wer nur eine Wald- und Wiesen-Karte bauen will, der muß sich um das Einfügen von Wartezuständen gar nicht kümmern). Noch langsamere Karten können die READY-Leitung erregen und damit den Zyklus noch weiter verlängern. Damit aber auch leistungsfähige Karten richtig zur Wirkung kommen können, hat man zusätzlich die Möglichkeit geschaffen, das Einfügen von Wartezuständen zu umgehen (Wait State Override). Eine solche Karte signalisiert zu Beginn des Buszyklus über eine weitere Busleitung (NOWS = No Wait States), daß sie gar keine Wartezustände benötigt.

Hinweis: Die Busschnittstellen verschiedener Mikrocontroller sind ähnlich ausgelegt. Ganz einfache Modelle haben gar kein Rückmeldesignal (Ready), sondern nur programmierbare Wartezustände. Der Entwickler muß also immer genau wissen, wie schnell die Einrichtung ist, auf die er gerade zugreift.

Strobe-Impuls vom Master

Der Master liefert einen Strobe-Impuls, und der Slave reagiert darauf (Abb. 1.33). Da der Slave erst mit dem Eintreffen des Strobe-Impulses wissen kann, was er tun soll, ist die eigentliche Datenübertragung typischerweise mit Bezug auf die Rückflanke des Strobe-Impulses spezifiziert. Das ist die typische Auslegung vieler Mikroprozessor-Bussysteme. Beispiel im PC-Bereich: ISA.

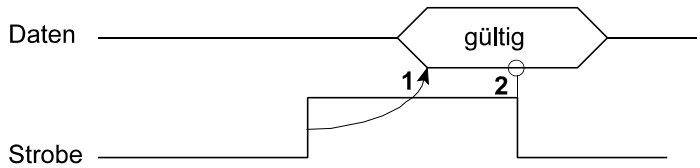


Abb. 1.33 Informationsübertragung mit Strobe-Impuls vom Master

Schreibablauf:

1) Master schaltet Daten auf Bus (nach Empfang des Strobe-Signals hat Slave Gelegenheit, den Datenweg erforderlichenfalls freizugeben),
2) Slave übernimmt die gültigen Schreibdaten.

Leseablauf:

1) Slave führt den eigentlichen Lesezugriff aus und legt die gelesenen Daten auf den Bus (vor dem Senden des Strobe-Signals hat Master den Datenweg ggf. freigegeben),
2) Master übernimmt die gültigen Lesedaten.

Strobe-Impuls von der sendenden Einrichtung

Die sendende Einrichtung (beim Schreiben: der Master; beim Lesen: der Slave) legt die Daten auf den Bus und gibt nach einem gewissen Zeitintervall einen Strobe-Impuls ab, der die Busbelegung als gültig kennzeichnet (Abb. 1.34). Die Verzögerung muß so gewählt werden, daß die Daten an der empfangenden Einrichtung unter Berücksichtigung der Laufzeiten (Busleitungen, Koppelstufen) mit dem eintreffenden Strobe übernommen werden können. Eine naheliegende Forderung: der Strobe-Impuls soll der Mitte des Zeitabschnitts wirksam werden, in dem die jeweilige Datenbelegung gültig ist (so daß sich die (unvermeidlichen) Laufzeitunterschiede zwischen den einzelnen Signalen nicht auf die Datenübernahme auswirken).

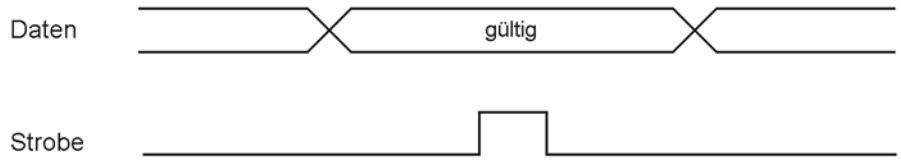


Abb. 1.34 Informationsübertragung mit Strobe-Impuls von der sendenden Einrichtung

Hinweis: Das Verfahren setzt voraus, daß der Slave weiß, was er zu tun hat. Die Art des Zugriffs muß also vorher übermittelt worden sein (z. B. als Zugriffskommando).

Strobe-Impuls mit Antwortsignal (Non-interlocked Handshaking)

Die Übertragung läuft an sich genauso ab wie in Abb. 1.34 dargestellt. Sie wird aber nicht nach einem definierten Zeitintervall beendet, sondern erst, wenn die empfangende Einrichtung einen Quittungsimpuls (Handshake-Signal) abgegeben hat (Abb. 1.35).

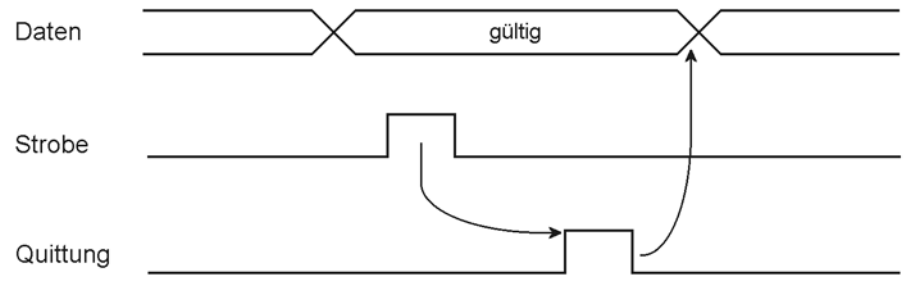


Abb. 1.35 Informationsübertragung mit Strobe- und Quittungs-Impulsen (Non--Interlocked Handshaking)

Teilweise Verriegelung (Half Interlocked Handshaking)

Das Strobe-Signal wird nicht als Impuls definierter Länge gebildet, sondern solange aktiv gehalten, bis der Quittungsimpuls eintrifft (Abb. 1.36); es ist praktisch mit dem Quittungsimpuls verriegelt (interlocked).

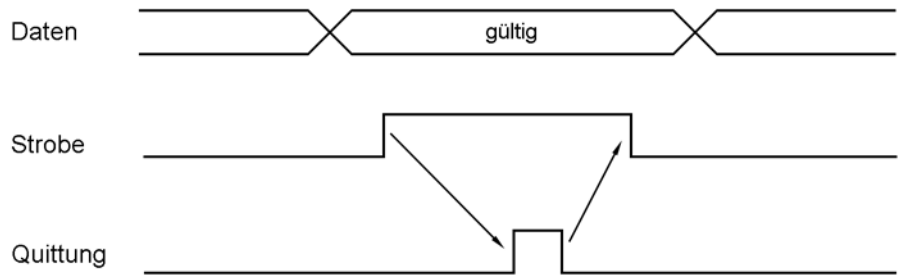


Abb. 1.36 Informationsübertragung mit teilweiser Verriegelung (Half Interlocked Handshaking)

Vollständige Verriegelung (Fully Interlocked Handshaking)

In diesem Fall wird auch das Quittungssignal nicht als Impuls definierter Länge abgegeben, sondern solange aktiv gehalten, bis die gewünschte Wirkung (das Abschalten des Strobesignals) eingetreten ist (Abb. 1.37). In dieser Betriebsart können alle Einrichtungen eine beliebige Arbeitsgeschwindigkeit haben. Ein einzelner Übertragungszyklus braucht aber wenigstens 4 Buslaufzeiten (Treiber – Busleitung – Empfänger).

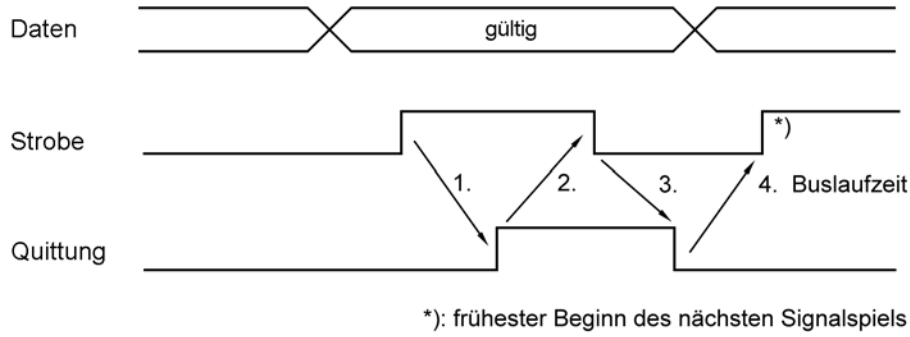


Abb. 1.37 Informationsübertragung mit vollständiger Verriegelung (Fully Interlocked Handshaking)

Mehrere Quittungssignale

In manchen Bussystemen sind mehrere Quittungssignale vorgesehen, damit der Slave außergewöhnliche Bedingungen über den Bus melden kann. Beispiele:

- das Abweisen des Buszyklus (wenn er auf Grund anderweitiger Aktivitäten nicht in der Lage ist, ihn auszuführen),
- Fehlerbedingungen.

Gültigkeit der Daten

Hinsichtlich der Beziehung zwischen Steuersignal (Takt, Strobe-Impuls bzw. Handshake-Signal) und Gültigkeit der Information gibt es folgende Möglichkeiten:

- die Information ist gültig in Bezug auf die Vorderflanke des Steuersignals,
- die Information ist gültig in Bezug auf die Rückflanke des Steuersignals,
- die Information ist gültig in Bezug auf die Dauer des Steuersignals (bzw. der Überlappung zweier verriegelter Steuersignale).

Bustakt oder Handshaking?

Handshaking hat den offensichtlichen Vorteil der Eleganz: man kann die einzelne Einrichtung jeweils so bauen, wie dies gerade zweckmäßig ist; trotzdem wird am Bus alles zusammenspielen. Zudem ist ein Bustakt oder ein eng toleriertes, starr vorgegebenes Zeitraster bei größeren Leitungslängen (mehrere Meter) auch elektrisch nicht unproblematisch. Deshalb wird für E-A-Interfaces traditionell das Handshaking bevorzugt (Beispiel: SCSI).

Weshalb haben aber die typischen Motherboard-Bussysteme einen Bustakt? Damit es (1) noch schneller geht und damit (2) die Hardware auf den Steckkarten einfacher wird.

Die einzelne Einrichtung ist so komplex, daß man intern nicht auf Takte verzichten kann. Jede Schnittstelle zu einem anderen Taktsystem oder zu asynchron schaltenden Signalen (und auch Handshaking-Signale schalten asynchron) erfordert somit eine Synchronisierung (Eintaktierung). Diese kostet im Durchschnitt wenigstens eine halbe Taktperiode. Will man metastabile Zustände sicher vermeiden, braucht man sogar eine weitere Taktperiode. Legt man hingegen die Busschnittstelle von vornherein für einen gemeinsamen Bustakt aus, so hat man es nur mit grundsätzlich synchronen Signalspielen zu tun, so daß eine Eintaktierung mit ihren Zeitverlusten nicht erforderlich ist, zumindest nicht unmittelbar in den Buskoppelschaltungen. Auch läuft ein Übertragungsvorgang naturgemäß schneller ab, wenn man nicht auf Rückmeldungen (Quittungssignale) warten muß. Schließlich erfordern synchron arbeitende Schaltungen zumeist weniger Aufwand.

1.4.6 Beenden des Buszyklus

Der Buszyklus ist beendet, nachdem das letzte Steuersignal inaktiv geworden ist.

Bus- und Datenzyklen

Ein Datenzyklus umfaßt die Signalfolgen, die zur einmaligen parallelen Übertragung von Datenbits (gemäß der jeweiligen Datenwegbreite) ablaufen, ein Buszyklus hingegen alle Signalfolgen für die Informationsübertragung zwischen Master und Slave, von der Erlangung der Busherrschaft bis zur abschließenden Freigabe.

Wann den Bus freigeben?

Wenn eine Einrichtung Master geworden ist, also die Kontrolle über den Bus erlangt hat: wann soll sie ihn wieder freigeben? Hierfür gibt es drei Verfahrensweisen:

1. der herkömmliche Multiplexbetrieb
Der Bus wird am Ende eines jeden Datenzyklus wieder freigegeben und dann erneut vermittelt. Das ist die Betriebsweise der herkömmlichen Standard-Interfaces bei Anschluß mehrerer, vergleichsweise langsamer E-A-Geräte. Der Vorteil: alle Funktionseinheiten können zum Zuge kommen (Fairneß), vorausgesetzt, man hat beim Konfigurieren keinen Fehler gemacht und den Bus nicht durch zuviele Einrichtungen gleichsam überlastet. Der Nachteil: die einzelne Einrichtung kann nur eine bestimmte mittlere Datenrate erreichen, selbst wenn sie längere Zeit den Bus ganz für sich hat, da jeder Datenzyklus einen kompletten Buszyklus (einschließlich Arbitrierung) erfordert. Beim Multiplexbetrieb mehrerer Einrichtungen wird die Übertragungsrate für die einzelne Einrichtung unvorhersagbar (Problem bei Realzeitanwendungen, Datenverlust bei Kommunikation mit Einrichtungen, die gar keine oder zu kleine Zwischenpuffer (FIFOs) haben).

2. der herkömmliche Stoßbetrieb (Burst Mode)
Der Bus wird solange belegt, bis ein bestimmter Datenblock übertragen ist (d. h. für mehrere – zumeist sogar für ziemlich viele – Datenzyklen) und erst dann freigegeben. Das ist die Betriebsweise der herkömmlichen Standard-Interfaces bei Anschluß schneller E-A-Geräte (z. B. Platten- oder Bandlaufwerke). Der Vorteil: eine hohe (und für die einzelne Einrichtung immer gewährleistete)

Datenrate, weil die Datenzyklen ohne vorherige Arbitrierung aufeinanderfolgen. Der Nachteil: Beeinträchtigung des Realzeitverhaltens, weil Einrichtungen, die den Bus benötigen, immer darauf warten müssen, bis ein ganzer Datenblock übertragen worden ist (je länger der einzelne Datenblock (Burst), desto länger die Latenzzeiten der auf die Busherschaft wartenden Einrichtungen).

3. die Verdrängungsstrategie (Preemption)
Dies ist eine Mischung aus beiden Prinzipien: der aktuelle Master kann die Kontrolle über den Bus solange behalten, bis eine andere Einrichtung (höherer Priorität) den Bus ihrerseits anfordert.

Busverriegelung (Bus Lock)

Die Busverriegelung bewirkt, daß die Kontrolle über den Bus über mehrere aufeinanderfolgende Datenzyklen hinweg nicht abgegeben wird. Anwendung: das Ausführen „ungeteilter" Lese-Schreib-Zugriffe („Test-and-Set"-Zugriffe), wie sie u a. zur Ressourcenverwaltung in Multiprozessorsystemen unbedingt notwendig sind. Beispiel: das LOCK-Signal des PCI-Bus.

1.4.7 Signalisierung von Sonderbedingungen

Die meisten Bussysteme haben eigene Leitungen zur Fehlersignalisierung, zum Auslösen von Unterbrechungen, zur Steuerung besonderer Buszyklen sowie zur Anzeige und Steuerung der aktuell genutzten Datenbusbreite und anderer „Nebenumstände" der Datenübertragung.

Fehlersignalisierung

Die typischen PC-Bussyteme sind unter dem Gesichtspunkt geringer Kosten entwickelt worden. Deshalb wurde die Fehlererkennung und -signalisierung ziemlich sparsam ausgelegt. Der Grundgedanke: jene Fehler, die man mittels Hardware erkennen könnte, sind eher selten – und wenn sie auftreten, reicht es, gleichsam das Handtuch zu werfen.

Typische Merkmale:

- Fehlerkontrolle an den Daten- und Adreßwegen (zusätzliche Paritätsbits, Mitführen und Auswerten ohnehin vorhandener Fehlerkorrekturbits),
- gelegentlich eine pauschale Zeitkontrolle der Buszugriffe (Timeout Check),
- Fehlersignalisierung über zusätzliche Leitungen.

Typische Reaktionen auf signalisierte Fehler:

1. Abbrechen der Übertragung,
2. Wiederholen der Übertragung,
3. bedingungslos Kapitulieren (soll heißen: nach Anzeigen einer Fehlermeldung stellt das System den Betrieb ein (bzw. gibt dem Nutzer gar keine andere Möglichkeit, als die Arbeit zu beenden)). Es wird dann ein Rücksetzen erwartet, um die Arbeit wieder aufzunehmen.

Die meisten Reaktionen entsprechen der 3. Art ...

Zur Zugriffswiederholung (Retry, Recovery)
Dies ist ein bewährtes Verfahren in Systemen, die wirklich auf Zuverlässigkeit hin ausgelegt sind. In manchen Spezifikationen des PC-Bereichs wird es als Möglichkeit erwähnt (Beispiel: PCI). Die Implementierung ist aber schwierig. Beispielsweise ist bei E-A-Zugriffen, bei Zugriffen auf Schreibpuffer, auf FIFOs usw. eine Zugriffswiederholung zumeist unmöglich oder sinnlos.

Erkennung von Fehlern in den Abläufen am Bus
Es wird kaum etwas getan. Beispielsweise wird in der PCI-Spezifikation sogar ausdrücklich davon abgeraten, Einrichtungen so auszulegen, daß sie dem Bus gleichsam auf die Finger sehen, d. h. die Bus-Abläufe überwachen und einschlägige Fehler melden (werden solche Fehler doch erkannt, so wird empfohlen, sich einfach vom Bus zu verabschieden (Übergang in den Ruhezustand)). Demgemäß ist kaum damit zu rechnen, daß die Entwickler umfassende Fehlererkennungshardware vorsehen.

Unterbrechungsauslösung

Typiherweise werden die Unterbrechungen von zentralisierten Einrichtungen (z. B. Unterbrechungssteuerschaltkreisen) ausgelöst. Die Unterbrechungsleitungen in den Bussystemen dienen nur dazu, diesen Schaltkreisen die Unterbrechungssignale zuzuführen. Die Unterbrechungssignalisierung steht deshalb traditionell außerhalb der eigentlichen Busprotokolle (wobei es aber Neues gibt - vgl. den folgenden Punkt 3).

1. die flankengesteuerte (Edge sensitive) Unterbrechungssignalisierung
Die mit den ersten IBM-PCs eingeführte herkömmliche Lösung. Für jede Unterbrechungsanforderung gibt es eine besondere Leitung (mit anderen Worten: zwei Einrichtungen dürfen nicht auf dieselbe Unterbrechungssignalleitung (Interruptleitung) geschaltet werden). Einrichtungen, die von Anfang an zum Industriestandard gehören, haben fest zugeordnete Interruptleitungen. Was im Laufe der Zeit hinzugekommen ist (Soundkarten, Netzwerkkarten usw.) muß sich im Rahmen der übriggebliebenen – frei verfügbaren – Leitungen entsprechend einrichten (das tun sie nur dann selbst, wenn die Plug-and-Play-Vorkehrungen richtig funktionieren...). Eine Unterbrechung wird ausgelöst, indem die Interruptleitung vom inaktiven auf den aktiven Pegel geschaltet wird. Es ist also die Signalflanke auf dieser Leitung maßgebend. Beispiel: ISA.

2. die pegelgesteuerte (Level sensitive) Unterbrechungssignalisierung
Mehrere Einrichtungen dürfen an dieselbe Interruptleitung angeschlossen werden (Interrupt Sharing – die Einrichtungen teilen sich gleichsam eine Interruptleitung). Hierzu sind die entsprechenden Koppelstufen als Open-Collector- oder Open-Drain-Treiber ausgelegt. Jede Einrichtung, die eine Unterbrechung auslösen möchte, aktiviert die betreffende Interruptleitung.

Es ist Angelegenheit der Software, herauszufinden, welche Einrichtungen eine Unterbrechung ausgelöst haben (z. B. durch Abfragen entsprechender Anforderungsregister in den Einrichtungen). Eine Einrichtung, die eine Interruptleitung aktiviert hat, muß diese Aktivierung so lange halten, bis sie durch Software zurückgesetzt wird. Abb. 1.38 veranschaulicht den grundsätzlichen Ablauf:

1. die Interruptleitung ist inaktiv,
2. in einzelnen Einrichtungen werden Unterbrechungsbedingungen wirksam. Infolgedessen wird die Interruptleitung aktiviert.
3. das löst einen Interrupt aus. Die behandelnde Software fragt Einrichtung für Einrichtung ab, ob eine Unterbrechungsbedingung anhängig ist oder nicht. Ist eine anhängig, so wird sie behandelt. Dann wird die Bedingung in der jeweiligen Einrichtung zurückgesetzt. Die betreffende Einrichtung gibt somit die Interruptleitung wieder frei.
4. sind alle Einrichtungen abgefragt worden, so wird die Interruptleitung wieder inaktiv.

Es ist also der Signalpegel auf der Interruptleitung maßgebend (Low = aktiv = Interrupt(s) anhängig). Beispiele: MCA, EISA, PCI.

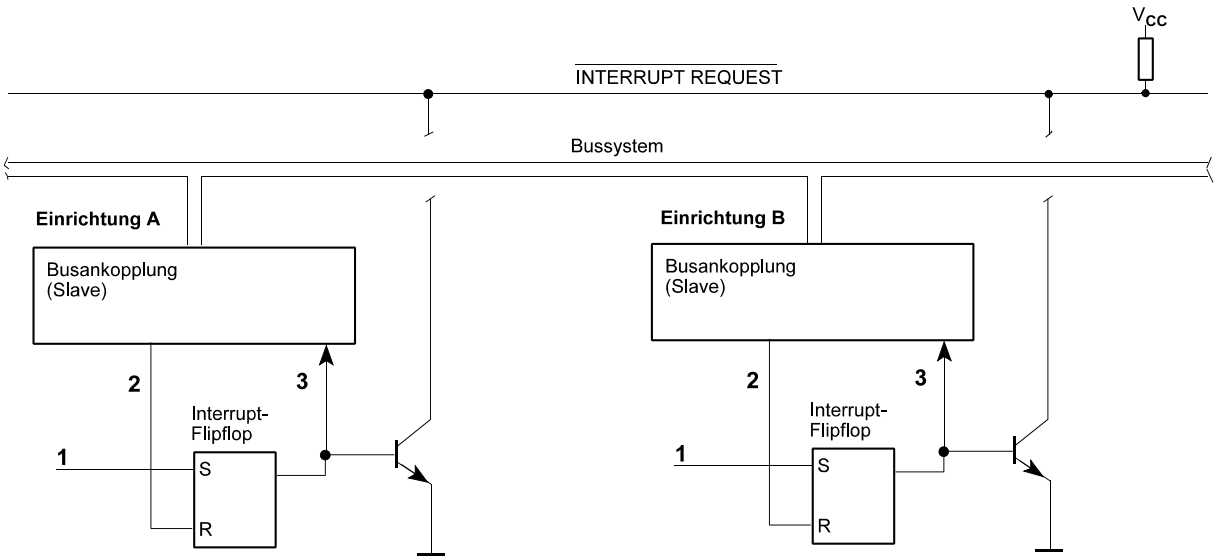


Abb. 1.38 Pegelgesteuerte Unterbrechungssignalisierung (Prinzip). Es sind zwei Einrichtungen A, B dargestellt, die an dieselbe Interruptleitung (INTERRUPT REQUEST) angeschlossen sind. 1 - Setzen der Unterbrechungsanforderung (aus den funktionellen Schaltungen heraus); 2 - Zurücksetzen des Interrupt-Flipflops durch Slave-Zugriff; 3 - Abfragen des Interrupt-Flipflops über Slave-Zugriff.

Hinweis: Das programmseitige Abfragen kostet offensichtlich Zeit. Es ist deshalb sinnvoll, die Unterbrechungsanforderungen zunächst über die vorhandenen Interruptleitungen zu verteilen und mehrere Einrichtungen nur dann auf eine gemeinsame Interruptleitung zu schalten, wenn keine weiteren Interruptleitungen mehr frei sind (Sache der Konfigurationssoftware).

3. Unterbrechungssignalisierung durch Nachrichtenübertragung (Message passing)
Auch wenn mehrere Einrichtungen auf einer Interruptleitung arbeiten dürfen: die richtige Zuordnung (beim Konfigurieren) und die programmseitige Behandlung ist jeweils ein Kunststück für sich – es gibt hierbei ganz spitzfindige Probleme. Der Ausweg: wir gehen vom Prinzip der unabhängigen Interruptleitungen ab. Statt dessen melden wir die Unterbrechungsbedingungen über besondere Buszyklen an den Steuerschaltkreis bzw. an den Prozessor (mit anderen Worten: um einen Interrupt auszulösen, wird eine spezielle Nachricht über den Bus geschickt). Beispiel PCI 2.2 (Message Signaled Interrupts (MSI)).

Steuerung von Einzelheiten der Datenübertragung

Es sind Leitungen vorgesehen, über die Master und Slave die näheren Umstände der Buszyklen signalisieren. Das betrifft:

- die Freigabe des Bus (Burst-Übertragung und Busverriegelung),
- die Ankündigung der aktuellen Übertragungsbreite des Masters,
- die Ankündigung der aktuellen Übertragungsbreite des Slaves,
- die Anzeige der jeweils gültigen Bytepositionen (Byteerlaubnissignale).

2. Alternativen zum Bus

2.1 Punkt-zu-Punkt-Verbindungen

Die (möglichst kurze) Punkt-zu-Punkt-Verbindung hat den Vorteil, daß die Zeitverluste entfallen, die durch den Overhead der Buszyklen und durch die elektrische Auslegung des Bus bedingt sind. Man hat aber über viele Jahre hinweg am Busprinzip festgehalten – die Vorteile waren einfach zu groß (Erweiterbarkeit, Austauschbarkeit von Subsystemen, vergleichsweise wenige Signalleitungen). Mittlerweile kann man sich aber mehr leisten; komplizierte Funktionseinheiten verschwinden gleichsam in hochintegrierten Schaltkreisen, und auch große Gehäuse mit Hunderten von Anschlüssen werden in er Massenfertigung beherrscht.).

Die Hersteller arbeiten daran, die Leistungrenzen der auf herkömmlichen Prinzipien beruhenden Bussysteme zu überwinden. Dabei ersetzt man die herkömmlichen Bussysteme durch Punkt-zu-Punkt-Verbindungen zwischen den hochintegrierten Schaltkreisen. Vom bisherigen Systembus kommt man so nach und nach zum Hochgeschwindigkeitsnetzwerk – die Übergänge sind fließend. Diese Entwicklungen laufen auf verschiedenen Ebenen:

- aufbauend auf den herkömmlichen Bussystemen wird die Entwicklung bis ins Extreme weitergeführt (Beispiele: AGP, DirectRambus, PCI-X),
- Lösungen, die bisher nur in Mainframes, Supercomputern usw. üblich waren, werden im Bereich der PC-Technik verwirklicht. Beispiele solcher Grundsatzlösungen: (1) Daisy-Chain- und Ringstrukturen, (2) Crossbar-Netzwerke und Schaltverteiler (Beispiele: GigaBridge, HyperTransport, RapidIO, StarFabric).
- man entwickelt extrem schnelle serielle Interfaces. Falls die Datenrate eines solchen Signalwegs nicht ausreicht, betreibt man mehrere parallel (Beispiel: InfiniBand. PCI Express).

2.2 Daisy-Chain- und Ringstrukturen

Der Schieberegister-Ringbus

Abb. 2.1 veranschaulicht das Prinzip: jeder angeschlossenen Funktionseinheit ist ein Register zugeordnet, das für jede Busleitung ein Flipflop enthält. Alle diese Register sind hintereinandergeschaltet; die einer Busleitung jeweils zugeordneten Flipflops bilden also ein Schieberegister. So entsteht ein Ring, in dem (gesteuert durch einen zentralen Takt) die Information zirkulieren kann.

Abb. 2.2 zeigt, wie die Funktionseinheiten an diesen Ringbus angeschaltet sind. Zu empfangende Daten können unmittelbar vom jeweiligen Busregister „abgezapft” werden; zu sendende Daten sind jeweils zur rechten Zeit in den Schiebeweg einzuspeisen. Die Adressierung der Funktionseinheiten und die Bestimmung der Zeitpunkte zum Abzapfen bzw. Einspeisen entspricht weitgehend dem Steuerprinzip (Übertragungsprotokoll) eines Token-Ring- Netzwerks (, nur daß die jeweiligen Kennzeichnungsangaben (Token) parallel über gesonderte Ringbusleitungen mitübertragen werden.

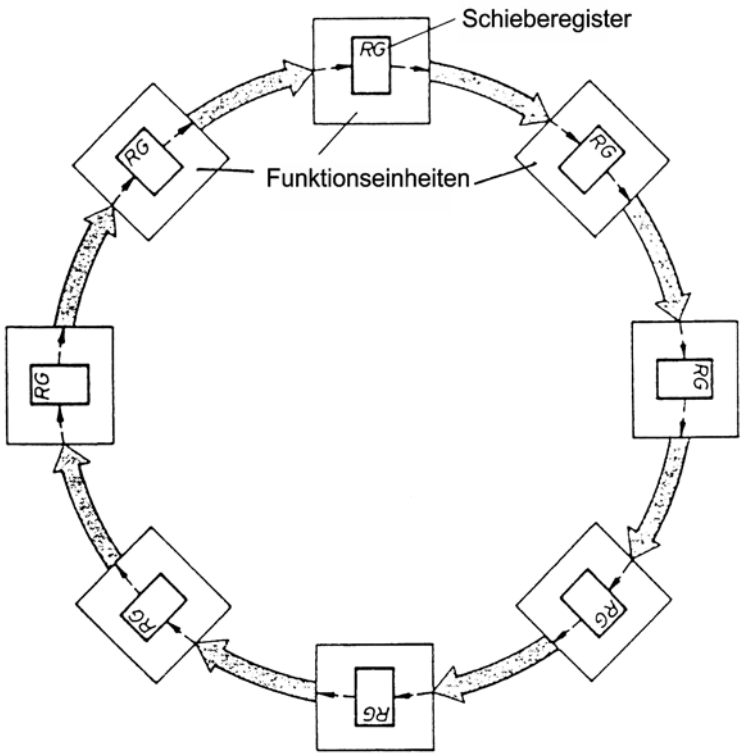


Abb. 2.1 Schieberegister-Ringbus

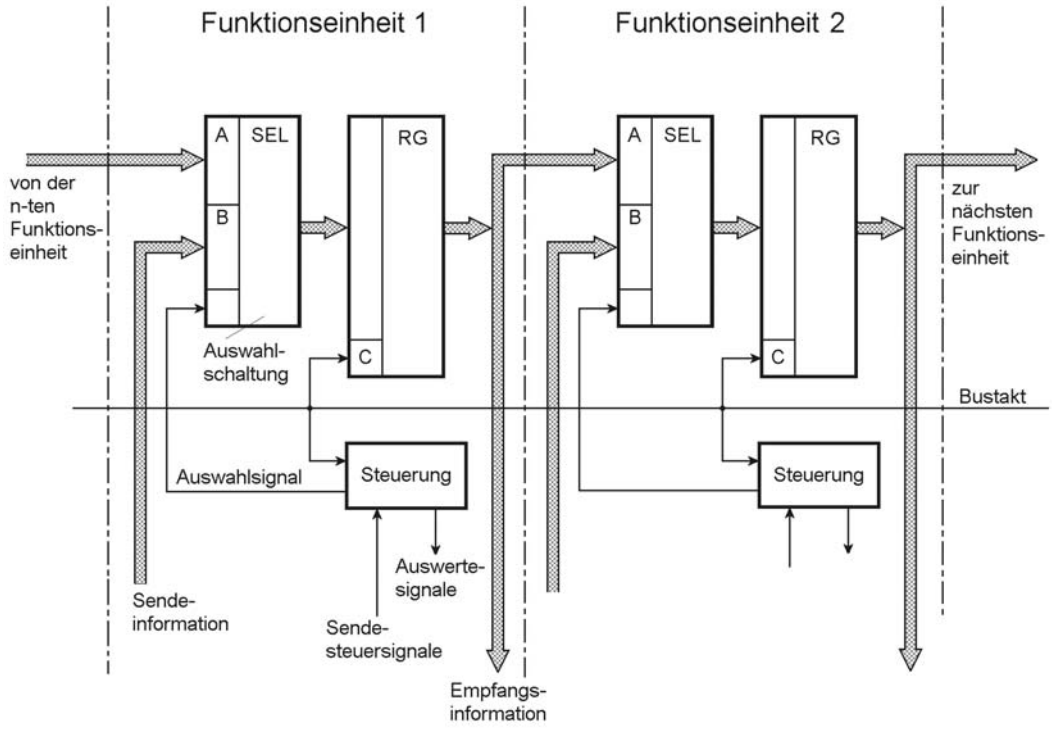


Abb. 2.2 Anschaltung einer Funktionseinheit

Die Vorteile:

- der Schiebe-Ring kann mit höchstmöglicher Taktfrequenz betrieben werden; alle Busleitungen sind nur kurze Punkt-zu-Punkt-Verbindungen mit praktisch vernachlässigbarer Laufzeit,
- bei Übertragung größerer Datenmengen zwischen zwei Funktionseinheiten wird die höchste Datenrate erreicht, die im Rahmen der jeweiligen Technologie und Systemgestaltung überhaupt möglich ist,
- Buskonflikte gibt es nicht,
- es können mehrere Übertragungsvorgänge gleichzeitig stattfinden,
- man kann absolut faire Buszuteilungsstrategien verwirklichen (so daß jede Funktionseinheit, die den Bus braucht, auch in kurzer Zeit zum Zuge kommt),
- Prüfung und Fehlersuche gestalten sich vergleichsweise einfach.

Die Nachteile:

- es ist nicht möglich, auf einfache Weise den Ring zu erweitern bzw. Funktionseinheiten herauszunehmen,
- auch kleinste Fehler im Busanschluß einer Funktionseinheit legen den gesamten Ring lahm,
- der bedeutsamste Nachteil: n Bussignale erfordern 2n Steckkontakte je Funktionseinheit.

Daisy-Chain-Schiebeketten

Der Grundgedanke: wir verzichten darauf, einen Ring zu bilden und betreiben das Interface als einfache Schiebekette. Die Daten werden darin von Einrichtung zu Einrichtung weitergeschoben. Soll diese Auslegung wirklich universell sein, brauchen wir einen zweiten Schiebeweg für den Informationstransport in die jeweilige Gegenrichtung (Abb. 2.3).

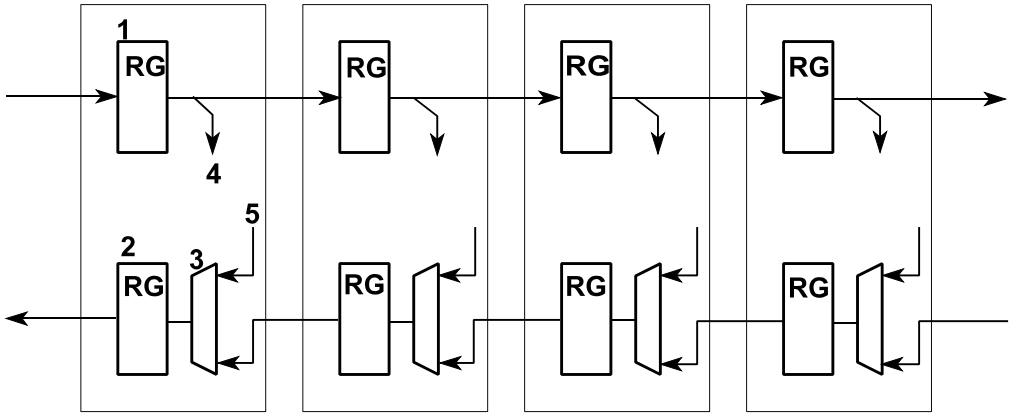


Abb. 2.3 Daisy-Chain-Schiebekette. 1, 2 - Register der jeweiligen Schiebekette; 3 - Auswahlschaltung für zu sendende Daten (vgl. Abb. 2.2); 4 - Abzweigen der empfangenen Daten; 5 - Einspeisen der zu sendenden Daten.

2.3 Crossbar-Netzwerke und Schaltverteiler

Ein Crossbar-Netzwerk ist eine reguläre „Jeder-mit-Jedem"-Verbindung. So etwas kann man mit Auswahlschaltern (Multiplexern) verwirklichen: jeder Eingang wird über einen Multiplexer mit allen Ausgängen verbunden. Abb. 2.4 veranschaulicht dies anhand eines Systems aus 4 Prozessoren und 4 Speichermoduln.

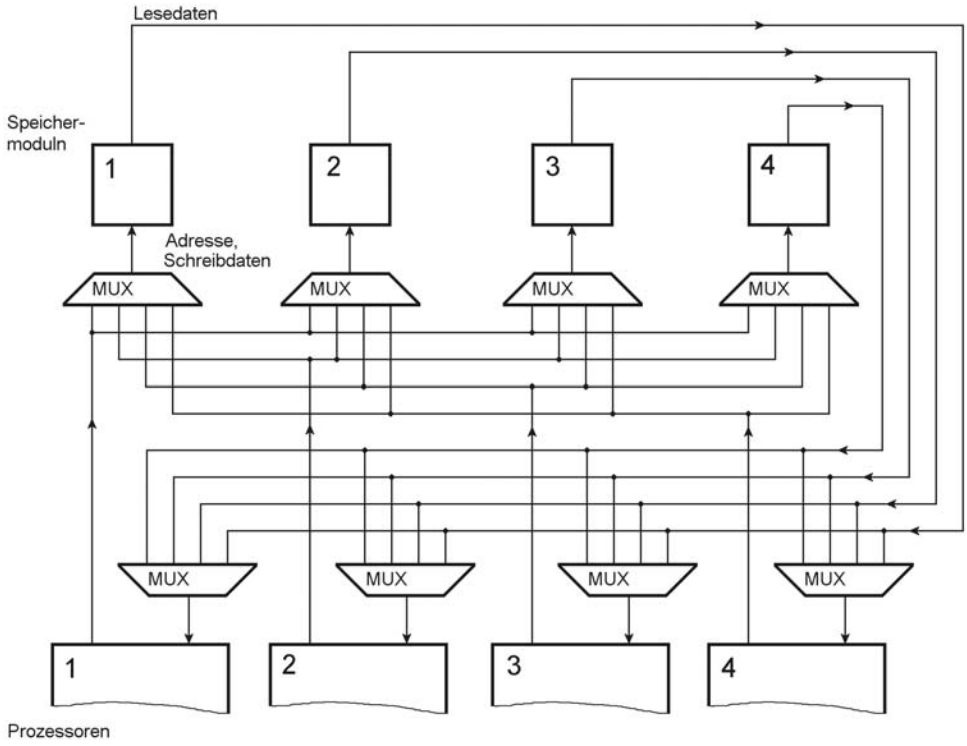


Abb. 2.4 Crossbar-Verbindungen mit Multiplexern

In der Praxis baut man solche Verbindungsstrukturen allerdings kaum mit Multiplexern auf, sondern zumeist mit matrixförmig angeordneten Schaltelementen (Abb. 2.5). Alle Eingänge bilden beispielsweise die Zeilen einer Matrix, als Ausgänge die Spalten. In allen Kreuzungspunkten, die für das Bilden von Verbindungen in Frage kommen, sitzen steuerbare Verbindungen (Schaltelemente). Man verwendet dafür zumeist besondere Halbleiterstrukturen (Transfer Gates).

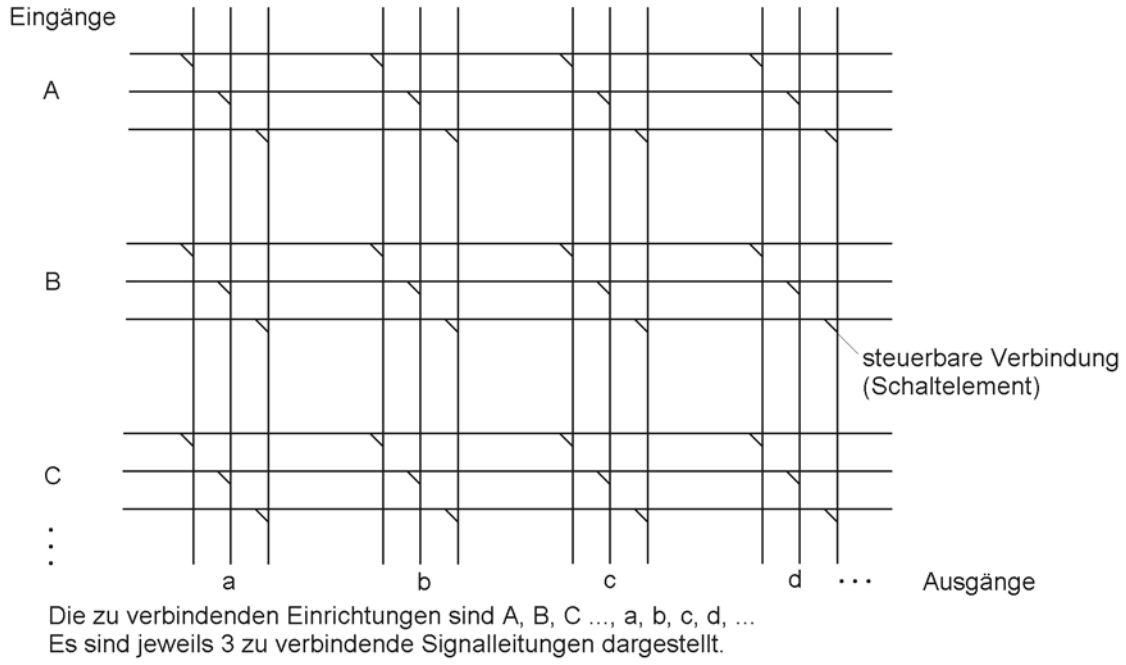


Abb. 2.5 Crossbar-Struktur

Diese Struktur ähnelt dem Barrel Shifter). Zum Aufbau von Crossbar-Strukturen gibt es spezielle Schaltkreise.

Der Vorteil des Crossbar-Netzwerks: geringste Latenzzeiten – es ist eine rein kombinatorische Verbindung ohne Zwischenspeicherung (ist die Verbindung erst einmal geschaltet worden, so beträgt die Durchlaufzeit höchstens einige ns). Zudem wird ein Crossbar-Netzwerk in den meisten Fällen auch dann noch funktionsfähig sein, wenn eine Funktionseinheit defekt ist (Fehlertoleranz).

Der Nachteil (abgesehen vom Schaltungsaufwand): eine Crossbar-Struktur für k Funktionseinheiten mit jeweils n zu schaltenden Signalen erfordert entweder eine „Zentrale" mit k ▯ n Steckkontakten oder (k-1) ▯ n Steckkontakte je Funktionseinheit (jede Einrichtung muß mit allen anderen Einrichtungen über unabhängige Signalwege verbunden werden). Zudem ist ein solches System nicht auf einfache Weise erweiterbar.

Richtwert: Crossbar-Strukturen sind zweckmäßige Lösungen, sofern – größenordnungsmäßig – zwischen 4 und 16 Einrichtungen miteinander zu verbinden sind. Eine typische Anwendung: Multiprozessorsysteme.

Der Ausweg (vor allem dann, wenn es um wesentlich mehr Einrichtungen geht): die gesamte Durchschaltmatrix aus vielen kleinen Schalterelementen aufbauen, wovon jedes z. B. zwei Eingänge mit zwei Ausgängen verbinden kann. Der Fachbegriff: Switch Fabric bzw. Schaltverteiler.

3. Prinzipien der Leistungssteigerung

3.1 Hindernisse auf dem Wege zu extremen Datenraten

3.1.1 Signalflanken und Logikpegel

Signalwechsel brauchen Zeit. Die Anstiegszeit der Signalflanken begrenzt die Impulsfolgefrequenz des einzelnen Signals. Abb. 3.1 soll veranschaulichen, was im günstigsten Fall erreichbar ist.

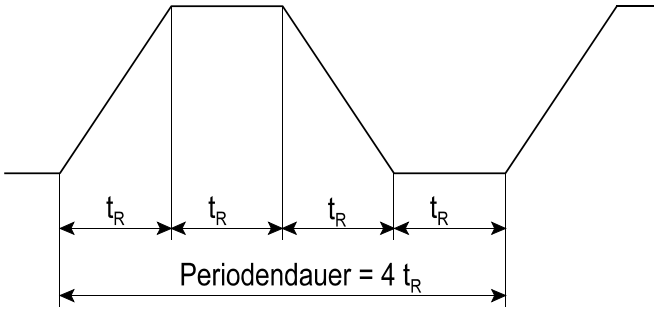


Abb. 3.1 Zur höchstmöglichen Impulsfolgefrequenz - eine plausible Modellvorstellung

Wir idealisieren zunächst den Signalverlauf und nehmen an, daß Anstiegs- und Abfallzeiten gleich sind. Unsere Impulse sollen die jeweils geringstmögliche Anstiegszeit t_r haben und jeweils für eine weitere Dauer von t_r im betreffenden Pegel (High oder Low) verbleiben. Wir kommen somit auf eine Periodendauer von 4 t_r bw. auf eine Impulsfolgefrequenz von $\frac{1}{4\ t_r}$.

Ein solcher Verlauf sieht offensichtlich noch wie eine „vernünftige" Impulsfolge aus; kürzere Impulse würden die Form von Nadeln haben - man würde sie nicht als Signale, sondern als Störungen ansprechen.

Die absolute Obergrenze
Signalleitungen in Form von Leiterzügen und Kabeln können wir näherungsweise als Tiefpässe mit Grenzfrequenzen von 1...> 2 GHz ansehen. Die Grenzfrequenz eines Tiefpasses begrenzt aber die Anstiegszeit der Signalflanken (Eigenanstiegszeit). Tabelle 3.1 nennt einige Anhaltswerte.

Systemumgebung	**Anstiegszeit t_r**	**minimale Periodendauer ($4\ t_r$)**	**Einheitsintervall ($2\ t_r$)[2)]**	**maximale Impulsfolgefrequenz**	**maximale Datenrate je Leitung[3)]**
Signalleitungen, 2 GHz; InfiniBand, GaAs-Logik	170 ps[1)]	680 ps	340 ps	1,5 GHz	3 GBits/s
Signalleitungen, 1 GHz	350 ps[1)]	1,4 ns	700 ps	600 MHz	1,2 GBits/s
RSL, ECL, LVDS	200...600 ps	800 ps...2,4 ns	400...1200 ps	400 MHz...1,2 GHz	0,8...2,4 GBits/s
CMOS, BiMOS, GTL, SSTL	1...2 ns	4...8 ns	2...4 ns	125...250 MHz	250...500 MBits/s
„Wald- und Wiesen-Logik" (TTL, HC)	$ 5 ns	$ 20 ns	$ 10 ns	# 50 MHz	# 100 MBits/s

1): Eigenanstiegszeit; 2): × 1 Bitzelle; 3): Kehrwert des Einheitsintervalls

Tabelle 3.1 Maximale Impulsfolgefrequenzen (Anhaltswerte)

Können solche Grenzwerte in der Praxis auch ausgenutzt werden?
Es kommt darauf an. Was sofort einleuchet: geht es nur um eine einzige Signalleitung, so können wir sie ohne weiteres mit so kurzen Impulsen ansteuern, wie es die technischen Gegebenheiten zulassen (Beschaffenheit der Leitung, Auslegung der Treiber- und Empfängerschaltkreise). Diese Tatsache bildet die Grundlage der verblüffenden Datenraten, die sich mit serieller Informationsübertragung erreichen lassen – die Werte reichen tatsächlich bis in die GBits/s (Beispiel: InfiniBand, PCI Express und SATA mit 2,5 GBits/s). Sollen hingegen mehrere Signalleitungen parallel betrieben werden, so ist darauf Rücksicht zu nehmen, daß jeder einzelne Signalweg seine eigenen Zeitkennwerte hat, daß aber die Zeiten aufeinander bezogen werden müssen. Das erfordert entsprechende Zugaben.

Flanken so steil wie möglich?
Viel hilft nicht immer viel. Sehr steile Flanken bedeuten auch sehr intensive Störungen. Dort, wo es darauf ankommt, bemüht man sich deshalb, bestimmte Mindest-Flankensteilheiten zu gewährleisten (Beispiel: Rambus); die Flanken werden so „langsam" spezifiziert, wie dies im Interesse der Arbeitsfrequenz bzw. Datenrate gerade noch zuträglich ist.

Der Signalhub
Nehmen wir die Anstiegsgeschwindigkeit der Signalflanken als gegeben an, so ist die Umschaltzeit zwischen den beiden Logikpegeln (Low und High) offensichtlich um so kürzer, je geringer der Signalhub ist, das heißt, je näher beide Spannungsbereiche beieinanderliegen (Abb. 3.2). Deshalb bevorzugt man niedrige Signalpegel (GTL, SSTL, RSL usw.). Hat ein Bussystem mehrere Pegelspezifikationen, so werden die höheren Datenraten typischerweise nur in Konfigurationen erreicht, die mit den jeweils niedrigeren Pegeln arbeiten. Beispiele: 3,3-V-PCI und 1,5-V-AGP.

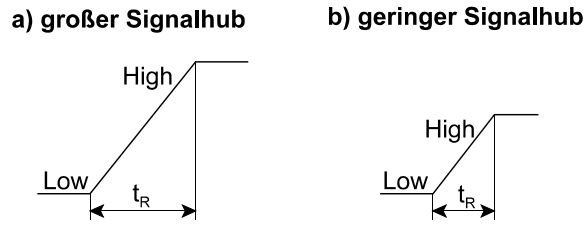


Abb. 3.2 Signalhub und Anstiegszeit

3.1.2 Zeitversatz (Skew)

Der Skew (ist der Zeitversatz zwischen aufeinander zu beziehenden Signalflanken, die eigentlich genau gleichzeitig eintreffen sollen (Abb. 3.3).

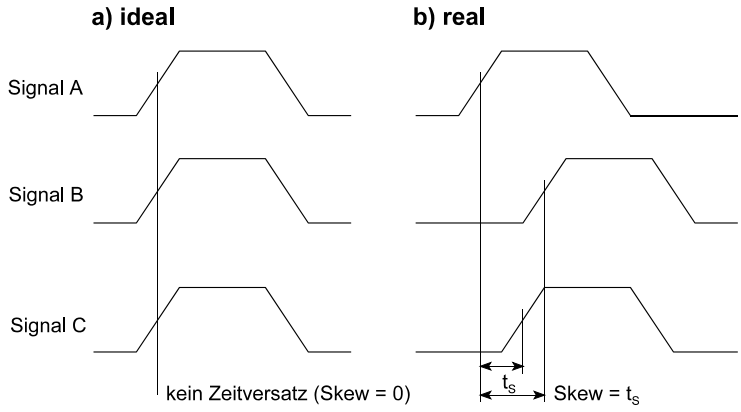


Abb. 3.3 Der Skew

Mehrere Signale sollen gleichzeitig schalten. Die empfangenden Einrichtungen sollen alle Signalpegel (z. B. die Belegung eines Datenwegs) gleichzeitig sehen. Das ist der Idealfall (Skew = 0). Tatsächlich kommt aber jedes Signal zu einer anderen Zeit an. Der größte Zeitunterschied (zwischen dem am frühesten und dem am spätesten ankommenden Signal (hier: zwischen den Signalen A und B)) ist als Skew bei der Festlegung der Zeitkennwerte eines Bussystems oder Interfaces zu berücksichtigen.

Je größer der Skew, desto geringer die höchstmögliche Datenrate. Werden extreme Datenraten angestrebt, ist nur ein sehr geringer Skew tragbar.

Was zum Skew beiträgt:

1. Unterschiede in den Verzögerungszeiten der Logik auf der Senderseite,
2. Unterschiede zwischen den Treiberstufen (Verzögerungszeiten, Anstiegszeiten),
3. unterschiedliche Leitungslängen,
4. unterschiedliche kapazitive Belastung der Signalleitungen,
5. unterschiedliche Schwellwerte der Empfänger,
6. Unterschiede in den Verzögerungszeiten der Logik auf der Empfängerseite.

Die Anteile 1, 2 und 6: unterschiedliche Verzögerungs- und Anstiegszeiten
Schlimmstenfalls wären zum einen die maximalen und zum anderen die minimalen Verzögerungszeiten (gemäß Datenblatt) anzusetzen. Das ist aber eine pessimistische Rechnung, weil diese Maximal- und Minimalwerte für den jeweils ungünstigten Fall der Betriebsbedingungen spezifiziert werden. Je ähnlicher die Signalwege (in Hinsicht auf Betriebsverhältnisse und technische Ausführung), desto geringer der Skew. Werden die Signale von mehreren Schaltkreisen geliefert bzw. empfangen, ist der Skew offensichtlich größer, als wenn alle Signale auf einem einzigen Schaltkreise gebildet und von einem einzigen Schaltkreis empfangen werden. Vom PCI-Bus an hat man deshalb die Bussysteme entsprechend ausgelegt (Prinzip: es wird von Grund auf dafür gesorgt – notfalls mit Kompromissen in anderer Hinsicht – daß man die vollständige Busanschaltung tatsächlich in einem einzigen Schaltkreis unterbekommt).

Die Anteile 3 und 4: Unterschiede in der Leitungslänge und in der kapazitiven Belastung
Sie sind durch entsprechende Dimensionierung und Verlegung der Leitungen so gering wie möglich zu halten. Es geht hier nicht um „so kurz oder sowenig wie möglich", sondern um „alles gleich" - gleiche Leitungslängen, gleiche kapazitive Belastung usw. Notfalls wird nachhaltig für Gleichheit gesorgt (u. a. werden Leiterzüge in Schlaufen geführt oder mit Kondensatoren beschaltet).

Anteil 5: unterschiedliche Schwellwerte
Herkömmliche Eingangsstufen von Logikschaltungen weisen untereinander eine beachtliche Toleranz ihrer Schaltschwellen auf. Infolgedessen schaltet die eine Stufe eher, die andere später (Abb. 3.4). Dementsprechend kommen die einzelnen Signale mit einem merklichen Zeitversatz an der funktionellen Logik des Empfängers an. Dieser Anteil am Skew ist durch die Schaltungstechnik der Eingänge bedingt; er läßt sich auch durch besondere Sorgfalt in der Fertigung nicht entscheidend verringern.

Der Ausweg: wir gehen von den herkömmlichen Eingangstufen ab und verwenden statt dessen Schaltungen, die die ankommenden Signale mit einer festen Referenzspannung (V_{REF}) vergleichen (Abb. 3.5).

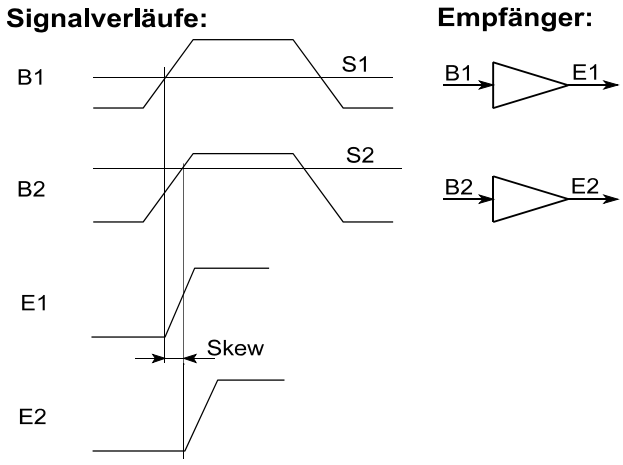


Abb. 3.4 Skew infolge unterschiedliche Schwellwerte. B1, B2 - Bussignale (die hier idealerweise ohne Skew ankommen); E1, E2 - Ausgangssignale der Empfänger; S1, S2 - unterschiedliche Schaltschwellen. Überschreitet das Eingangssignal die Schaltschwelle, so schaltet der Empfängerausgang von Low nach High. Das geschieht desto eher, je niedriger die Schaltschwelle ist.

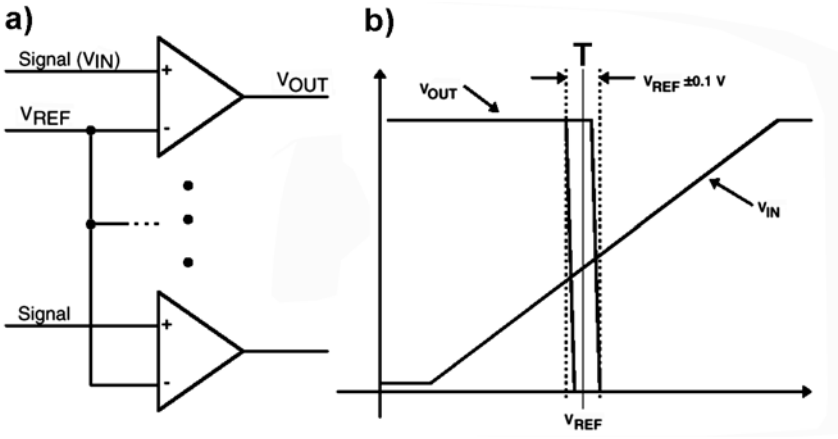


Abb. 3.5 Signalbewertung durch Vergleich mit Referenzspannung am Beispiel des AGP (Intel). a) - Comparatoren als Empfangsstufen; b) - Signalbewertung; V_{IN} - AGP-Signal; V_{OUT} - Signal zur funktionellen Logik; V_{REF} - Referenzspannung; T - Toleranzfeld.

Die Eingangsstufen sind schnelle Comparatorschaltungen. Um die Referenzspannung herum ist ein Toleranzfeld (T) von beispielsweise (AGP) 200 mV (× ± 100 mV) spezifiziert. Der Comparator verhält sich nun folgendermaßen:

- Eingangspegel (V_{IN}) unterhalb des Toleranzfeldes: Ausgangssignal (V_{OUT}) ist garantiert Low,
- Eingangspegel (V_{IN}) oberhalb des Toleranzfeldes: Ausgangssignal (V_{OUT}) ist garantiert High,
- Eingangspegel (V_{IN}) innerhalb des Toleranzfeldes: Ausgangssignal (V_{OUT}) ist entweder Low oder High.

Die Unsicherheit des Umschaltens ist also auf den vergleichsweise schmalen Bereich des Toleranzfeldes beschränkt. Zudem erhalten alle Comparatoren dieselbe Referenzspannung, und sie sind auch auf demselben Schaltkreis angeordnet. Somit wird der gegenseitige Versatz der Schaltschwellen und damit der Skew nur gering sein.

3.1.3 Signallaufzeit

Als Signallaufzeit (Flight Time) bezeichnet man die Zeit, die vergeht, bis eine vom Treiber aufgeschaltete Signalflanke am Empfänger angekommen ist (Abb. 3.6). Dieser Kennwert entscheidet maßgeblich darüber, mit welcher Datenrate ein Bussystem betrieben werden kann. Am Beispiel des PCI-Bus wollen wir etwas näher in die Einzelheiten gehen. Ein PCI-System können wir als ein vom Bustakt CLK gesteuertes Schaltwerk auffassen (Abb. 3.7). Die Grundforderung: die mit einer Taktflanke zu übernehmenden Signale müssen die Vorhalte- und Haltezeit (Setup Time, Hold Time) einhalten, also an den empfangenden Flipflops stabil anliegen, ehe die übernehmende Taktflanke eintrifft, und sie müssen wenigstens bis zum Übernahmezeitpunkt gehalten werden.

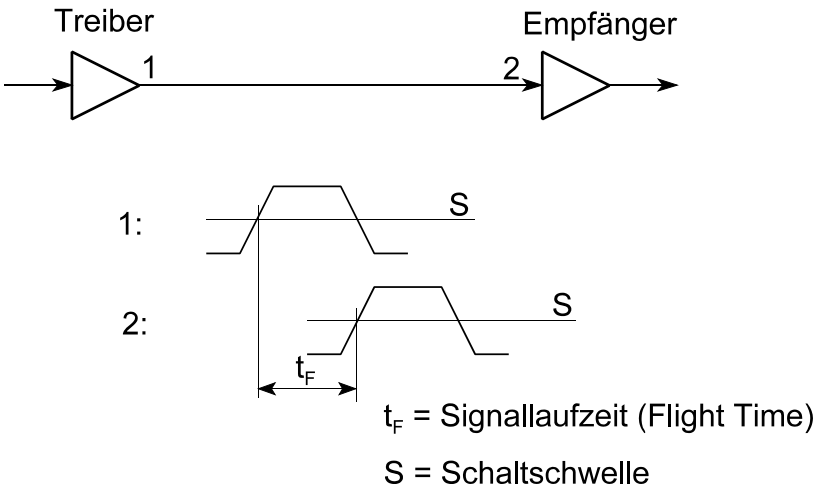


Abb. 3.6 Zur Definition der Signallaufzeit (Flight Time)

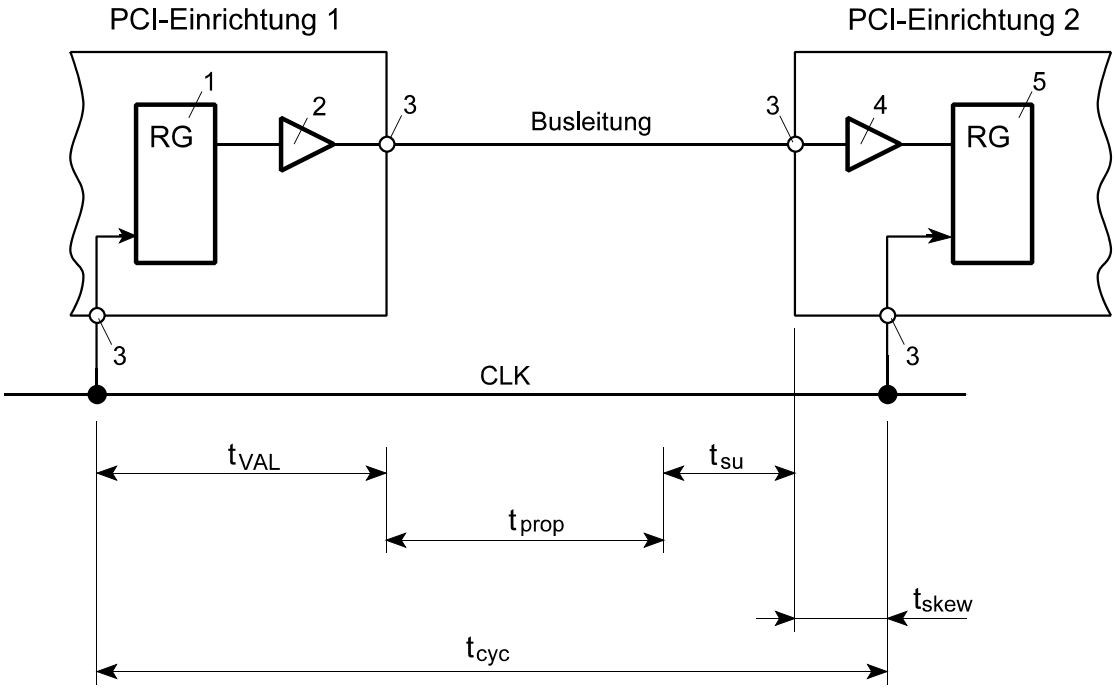


Abb. 3.7 Der PCI-Bus als synchrones Schaltwerk. 1 - sendendes Register bzw. Flipflop; 2 - Ausgangs-Treiber; 3 - Schaltkreisanschluß (Pin); 4 - Eingangsstufe; 5 - empfangendes Register bzw. Flipflop.

Die Abbildung soll veranschaulichen, was sich im Taktzyklus in Hinsicht auf die Signalausbreitung abspielt:

- die Low-High-Flanke des Taktes trifft am sendenden Register (1) ein,
- nach Ablauf der Verzögerungszeit t_{val} erscheint das Signal am Schaltkreisausgang (3),
- das Signal muß sich über die Busleitung ausbreiten,
- am Eingang des empfangenden Schaltkreises (5) muß das Signal wenigstens für die Dauer der Mindest-Vorhaltezeit t_{su} anliegen, bevor die nächste Low-High-Taktflanke erscheinen darf,
- der Zeitversatz der Takte an Sender und Empfänger (Clock Skew t_{skew}) an beiden Registern (1, 5) ist zusätzlich einzurechnen.

All diese Zeitabschnitte müssen in die Taktzykluszeit t_{CYC} hineinpassen. Abb. 3.8 veranschaulicht die Zeitspezifikationen für beide Taktfrequenzen (33 und 66 MHz).

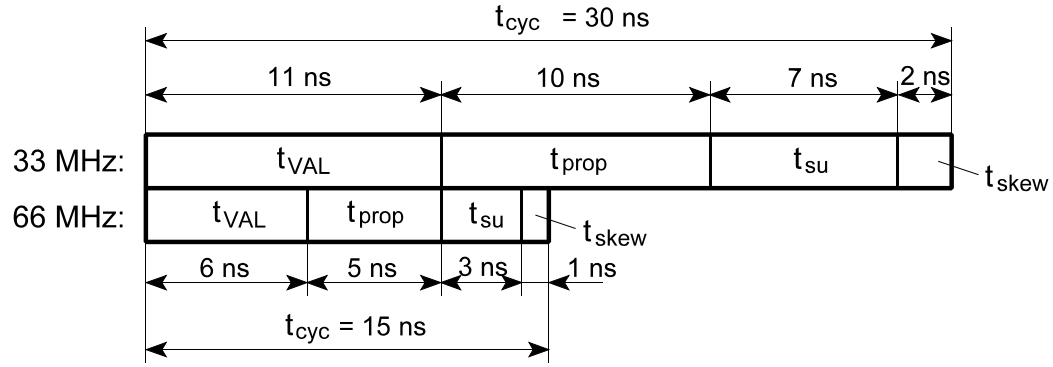


Abb. 3.8 Die einzelnen Zeitanteile in den PCI-Taktzyklen bei 33 und 66 MHz

Die Verzögerungszeit t_{val} und die Vorhaltezeit t_{su} sind Schaltkreis-Kennwerte, die Buslaufzeit t_{prop} und der Taktversatz t_{skew} ergeben sich hingegen aus den Leitungslängen der PCI-Konfiguration. Beachten Sie die extrem knappen Vorgaben für 66 MHz.

3.1.4 Treibfähigkeit

Um eine Leitung auf einen bestimmten Signalpegel zu treiben, muß die Treiberstufe einen bestimmten Stromfluß durch diese Leitung erzwingen. Im Moment des Schaltens sieht der Treiber eine kapazitive Last, oder er sieht zunächst den Wellenwiderstand einer Übertragungsleitung[1]. In beiden Fällen muß er den erforderlichen Strom entweder liefern oder aufnehmen können (um die Kapazität umzuladen oder um eine Wellenfront auszulösen).

Grenzen der Treibfähigkeit

Die Treibfähigkeit kann nicht beliebig erhöht werden, auch nicht bis zu den Grenzen, die an sich technologisch möglich sind

Schaltkreistechnologie
Eine Treiberstufe mit besonders hoher Treibfähigkeit muß vergleichsweise große Abmessungen haben (Stichworte: Leiterquerschnitt, Stromdichte). Solche Stufen kann man nicht in allen Technologien fertigen. Fordert man eine wirklich außergewöhnliche Treibfähigkeit (z. B. 100 mA und mehr), so kann man die betreffenden Stufen typischerweise nicht mehr in den funktionellen Schaltkreisen (Steuerschaltkreisen, Prozessoren usw.) unterbringen. Andererseits ist diese Auslegung - die gesamte Anschlußhardware in einem einzigen Schaltkreis - eine wichtige Forderung, wenn man mit extremen Taktfrequenzen arbeiten will.

Gesamtstrom
Treiben wir beispielsweise durch eine Leitung 100 mA, so ergibt das bei 32 Leitungen 3,2 A – und die müssen irgendwie zurück. Der typische Rückweg: die Masse. Derartige Stromstöße über Masseverbindungen zu führen, hat aber unschöne Folgen (Fachbegriff: Ground Bounce - ein bedeutender Störeffekt).

---

1 Beides sind Modellvorstellungen. Welche jeweils zutrifft, hängt von Leitungslänge und Flankensteilheit ab.

Störstrahlung und Stromverbrauch
Für beides gibt es Anforderungen, die unbedingt einzuhalten sind (EMV-Verordnung, Stromsparrichtlinien)..

Mit geringer Treibfähigkeit auskommen
Das heißt, typischerweise mit ca. 3...5 mA je Signal. Um trotzdem extreme Datenraten zu erreichen, sind verschiedene Prinzipen nutzbar:

- mit geringstmöglichem Signalhub arbeiten (je geringer der erforderliche Spannungshub, desto weniger Strom muß fließen). Ggf. differentielle Signalübertragung, um auch bei sehr niedrigem Signalhub die Übertragungswege unempfindlich gegen Störungen zu machen (Beispiele: LVDS, USB, InfiniBand).
- die rücklaufende Wellenfront ausnutzen (Reflected Wave Switching). Die Signalleitungen *nicht* abschließen. Das bewirkt, daß eine Wellenfront (Signalflanke) am Leitungsende reflektiert wird. Beim Zurücklaufen überlagert diese die Signalbelegung, die vom Treiber aufgeschaltet wird. Infolgedessen bekommen die Empfänger eine überhöhte Spannung zu sehen. Auf diesem Prinzip beruht z. B. der PCI-Bus. Der Trick – mit der reflektierten Wellenfront dem Treiber gleichsam dabei zu helfen, die Signalleitung auf den jeweiligen Pegel zu ziehen – funktioniert natürlich nur dann, wenn die rücklaufende Wellenfront noch während der Signalflanke an den Empfängern ankommt. Die Signalleitungen dürfen deshalb nicht allzu lang sein (PCI: 30...70 cm bei 33 MHz, bei 66 MHz 15...30cm). PCI ist also wirklich auf das Motherboard oder auf eine kurze Busplatine (CompactPCI) beschränkt.
- auf Bussysteme verzichten. Alternative: Punkt-zu-Punkt-Verbindungen.

3.1.5 Latenzzeiten

Latenzzeiten entstehen immer dann, wenn wir von einer Einrichtung etwas wollen, diese aber Zeit braucht, um das Gewünschte auszuführen. Sie lassen sich zwar durchaus verringern, aber nicht in beliebigem Maße.

Latenzzeiten sind nicht zu vermeiden (bei vielen Abläufen kann es keine Latenzzeit Null geben). Latenzzeiten kann man bestenfalls mit anderen Vorgängen überlappen (indem wir z. B. dann, wenn der eine Zugriff im Speicherschaltkreis ausgeführt wird, bereits den nächsten auf den Weg bringen). Damit das aber etwas nützt, müssen einige Voraussetzungen erfüllt sein:

- es müssen – aus der Anwendung heraus – genügend nahezu gleichzeitige unabhängige Anforderungen entstehen. Das ist manchmal der Fall (z. B. beim Durchschleusen von Datenpaketen, beim Umgang mit Multimedia-Datenströmen) und manchmal nicht (nämlich dann nicht, wenn die Ergebnisse vorhergehender Zugriffe benötigt werden, um über die Ausführung nachfolgender Zugriffe zu entscheiden (Stichwort: datenabhängige Algorithmen)).
- es müssen Signalwege und -protokolle vorgesehen werden, um Anforderungen parallel zu (gleichzeitig mit) laufenden Übertragungsvorgängen absetzen zu können,
- die betreffenden Einrichtungen müssen in der Lage sein, verschiedene Anforderungen auch tatsächlich gleichzeitig zu erledigen. Die typische Auslegung: interne Aufteilung in mehrere (kleinere) Funktionseinheiten (vgl. die Speicherbanks der modernen DRAMs – ein Schaltkreis oder Subsystem mit nur einer einzigen Speichermatrix kann zu einer Zeit nur einen Zugriff ausführen).

3.2 Prinzipen der Systemauslegung

3.2.1 Signalwege

Sowenige Signalwege wie möglich

Was ist besser? Ein Bussystem mit 64 Bits breitem Datenweg und einem Zeitraster von 100 ns (× 10 MHz Bustakt) oder eines mit nur 16 Bits breitem Datenweg und einem Zeitraster von 25 ns (× 40 MHz Bustakt)? Beide Auslegungen würden eine maximale Datenrate von 80 MBytes/s ermöglichen (alle 100 ns könnten 8 Bytes übertragen werden). Jeder Signalweg (Treiber - Leitung - Empfänger) belegt Siliziumfläche und Anschlüsse, braucht Strom und verursacht Störungen. Deshalb bevorzugt man neuerdings die Auslegung nach dem Grundsatz „schmal und schnell", also vergleichsweise wenige Signalwege, die aber mit extremen Taktfrequenzen betrieben werden.

Möglichst viele Signalwege zum Transportieren von Daten ausnutzen (Zeitmultiplexprinzip)

Herkömmliche Bussysteme haben unabhängige Signalwege zum Übertragen von Adressen und Daten (Adreßbus, Datenbus). Diese Auslegung führt bei 32 Bits Datenwegbreite und 32-Bit-Adressierung auf 64 Leitungen. Dieser Aufwand ist durchaus vernünftig, wenn es um wahlfreie Zugriffe geht. Aber viele Zugriffe betreffen aufeinanderfolgende Adressen; sie sind regelrechte Blocktransporte. Das gilt z. B. für den Transport kompletter Massenspeicher-Sektoren und für das Füllen von Caches. Sind die 32 Adreßleitungen nicht viel zu schade, um nur die jeweilige Folgeadresse weiterzugeben? Diese Überlegung führte zunächst zur kombinierten Nutzung: bei wahlfreien Zugriffen nutzt man nach wie vor 32 Adreß- und 32 Datenleitungen, bei Blocktransporten überträgt man hingegen nur die Anfangsadresse und sorgt dafür, daß anschließend alle 64 Leitungen zur Datenübertragung verfügbar sind (Verdoppelung der Datenrate). PCI beruht von Anfang an auf einem einzigen Signalweg von 32 bzw. 64 Bits, der zeitmultiplex für Adressen und Daten genutzt wird.

Weshalb hat man überhaupt breite Bussysteme gebaut?
Weil es seinerzeit gar nicht anders ging, und zwar vor allem aus zwei Gründen:

- Geschwindigkeit. Hat man nur vergleichsweise langsame Schaltkreise, z. B. Buskoppelstufen mit Verzögerungszeiten von 10 ns, so bleibt einem Entwickler gar nichts anderes übrig, als entsprechend viele Leitungen vorzusehen, um eine geforderte Datenrate zu erreichen.
- Kompliziertheit. Schmale Bussysteme brauchen kompliziertere Anschlußschaltungen; man muß die zu übertragenden Angaben (Adressen, Daten, Steuerkommandos usw.) auf die Sendeseite serialisieren und auf der Empfängerseite deserialisieren. Stehen dafür nur Schaltkreise geringen Integrationsgrades zur Verfügung, so wird eine solche Lösung viel zu aufwendig. Beispiel: eine 64-Bit-Datenstruktur (z. B. ein Speicherwort) soll über einen 16 Bits breiten Datenweg übertragen werden. Die Serialisierung allein kostet 16 4-zu-1-Multiplexer. also 8 Schaltkreisgehäuse (Steuerschaltungen gar nicht mitgerechnet). Der Aufwand zur Deserialisierung liegt in gleicher Größenordnung.

Beides ist heutzutage weitgehend gegenstandslos. Vor allem kann man sich schaltungstechnische Kompliziertheit wirklich leisten - Schaltungsanordnungen, die früher eine Vielzahl von Schaltkreisen erfordert hätten, bringt man auf modernen hochintegrierten Schaltkreisen buchstäblich am Rande unter.

Seriell oder parallel?

Die „total serielle" Informationsübertragung über eine einzige Leitung hat den Vorteil, daß der Signalweg tatsächlich mit der höchstmöglichen Impulsfolgefrequenz betrieben werden kann - bis in den GHz-Bereich hinein (wenn alles über eine einzige Leitung fließt, so kann es keinen Skew geben). Ganz ohne Probleme ist aber auch dieses Prinzip nicht:

Latenzzeiten
Es muß alles über diesen einen Weg, auch die Adressen, die Steuerkommandos, die Rückmeldungen usw. Beispiel: wir haben eine Datenrate von 1 GBits/s. Die Übertragung einer 32-Bit-Adresse würde dann wenigstens 32 ns erfordern, also etwa ebenso lange dauern wie über den herkömmlichen PCI-Bus.

Taktrückgewinnung
Übertragen wird ein Bitstrom aus Einsen und Nullen. Der Empfänger braucht aber einen Takt, um die Bits übernehmen zu können. Die Lösung: der Takt wird im Empfänger erzeugt, aber mit dem ankommenden Bitstrom synchronisiert. Um diese sog. Taktrückgewinnung zu unterstützen, muß der Bitstrom nach bestimmten Prinzipien aufgebaut werden. Vor allem ist es wichtig, daß der Empfänger in hinreichend kurzen Abständen Einsen (also Impulse) zu sehen bekommt (um die zeitliche Lage seines Taktes überprüfen und ggf. nachregeln zu können). Es muß aber möglich sein, beliebige Nutzdaten zu übertragen (also auch lange Datenblöcke, die nur Nullen enthalten). Diese Forderung führt zu einem gewissen Overhead) im Datenstrom. Eine der gängigen Lösungen: jeweils 8 zu übertragende (Nutz-) Bits werden in ein 10-Bit-Wort umgewandelt, in dem spätestens nach 5 Nullen eine Eins folgt (8B/10B-Codierung)..

Jedes (parallele) Bussystem kann serialisiert werden, ...
... sofern die serielle Übertragung nur schnell genug ist.

Anwendungsbeispiele:

- „Verlängern" von Schnittstellen über Interfacewandler: Serialisierung - serielle Übertragung über größere Entfernungen - Deserialisierung (Abb. 3.9),
- Schaltverteiler (Switch Fabric),
- Verbindungen zwischen Verteilerschaltkreisen, z. B. auf Motherboards (wobei man aber typischerweise ein schmales, aber kein serielles Interface verwendet („serienparallele" Übertragung)).

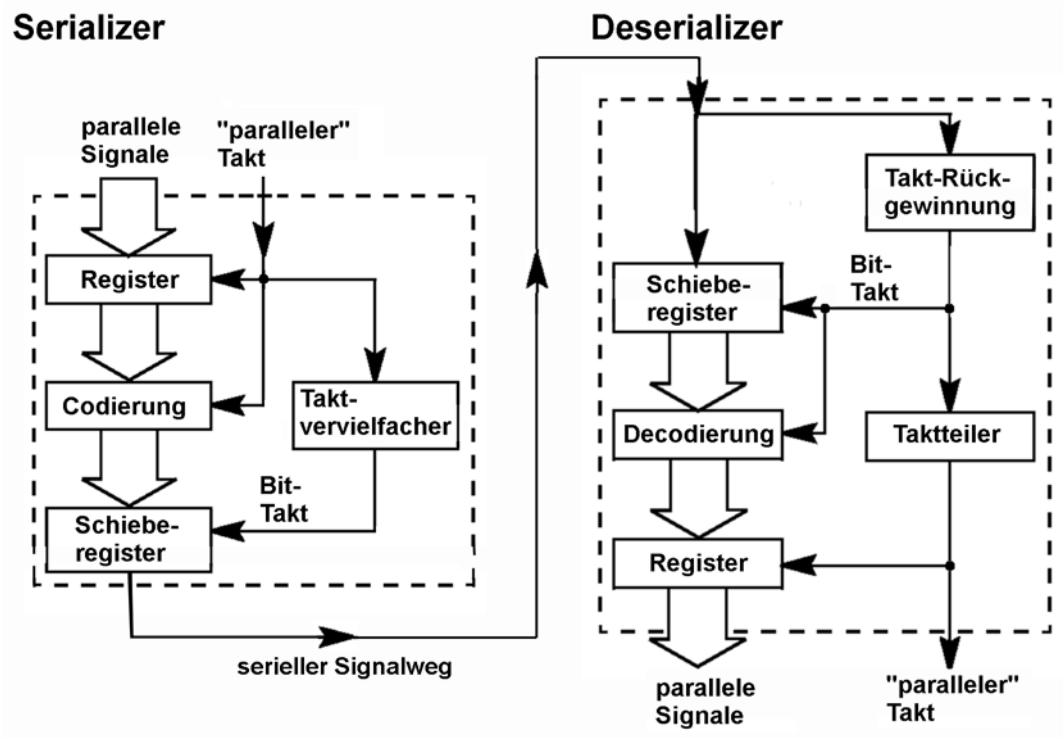


Abb. 3.9 Prinzip einer serialisierten Schnittstelle (Cypress)

Nicht milchmädchenmäßig rechnen. Nehmen wir an, unser (paralleler) Bus hätte 40 Signalleitungen und einen Bustakt von 33 MHz. Um die 40 Signalbelegungen nacheinander zu übertragen, wären 33 MHz ▯ 40 = 1320 MHz erforderlich. Das genügt aber nicht, denn der serielle Datenstrom muß aus verschiedenen Gründen verlängert werden (Overhead): zur Taktrückgewinnung (z. B. mit 2 zusätzlichen Bits je Byte (× 20% Overhead)), zur Synchronisation, Fehlererkennung usw.

Eine Radikallösung
Das Prinzip der seriellen Informationsübertragung wird nicht aufgegeben, sondern bis zu höchsten Datenraten hin entwickelt. Reicht es immer noch nicht, werden mehrere serielle Informationswege vorgesehen, und die zu übertragende Datenmenge wird auf diese Wege aufgeteilt (Beispiel: PCI Express).

Kompromißlösungen
Man verwendet mehr als einen Signalweg und greift dabei tief in die Trickkiste. Da es - um auch die letzte ns auszunutzen - wirklich knapp zugeht, kommt es auf jede Kleinigkeit an. Die Entscheidungen der Entwickler werden von vielen Erwägungen beeinflußt, auch von solchen, die außerhalb des eigentlichen Fachgebietes liegen[1]. Zudem stellen sich die vielfältigsten Nebeneffekte erst im Laufe der Zeit heraus[2]. Solche Schnittstellen können keine total universellen Interfaces sein. Deshalb finden wir verschiedenartige Lösungen, die im einzelnen weit voneinander abweichen, die aber im Grunde alle auf die gleichen Prinzipien zurückgreifen (AGP, PCI-X, Rambus, DDR-SDRAM, RapidIO usw.).

3.2.2 Paket- und Transaktionsorientierung

Paketorientierung
Jeder Übertragungsvorgang erfordert einen gewissen Zeitaufwand für Aktivitäten außerhalb der eigentlichen Datenübermittlung, also einen Overhead (Erlangen der Busherrschaft, Auswahl der Slave-Einrichtung, Kennzeichnung des Zugriffs, Verständigung über das Beenden des Zugriffs, ggf. Übermittlung von Zustandsmeldungen usw.). Es liegt deshalb nahe, größere zusammenhängende Datenblöcke zu übertragen, um den Anteil des Overheads an der gesamten Übertragungszeit zu vermindern (Burst-Betrieb). Lange Bursts haben zwar wenig Overhead, führen aber zu langen Latenzzeiten bei jenen Einrichtungen, die auf die Zuteilung der Busherrschaft warten müssen.

Der Ausweg: die Zerlegung der Datenströme in vergleichsweise kurze Stücke jeweils bekannter Länge – mit anderen Worten: in Pakete. Pakete sind gleichsam Behälter für Nutzdaten (Payload), die um organisatorische und beschreibende Angaben erweitert sind[3]. So kann man die Hardware genau auf diese Paketformate abstimmen.

Herkömmliche Bussysteme: ungeteilte Übertragung
Jeder in sich abgeschlossene Datenübertragungsvorgang zwischen Master und Slave bildet eine Einheit – vom Erlangen der Busherrschaft bis zum Freigeben des Busses. Brauchen die beteiligten Einrichtungen zusätzliche Zeit (um einen Speicher zu adressieren, um Daten abzuspeichern usw.) so verlängert sich der Übertragungsvorgang entsprechend (z. B. durch Einfügen von Wartezuständen oder durch verzögertes Senden von Handshake-Signalen). Der Steuerungsaufwand ist gering, der Bus wird aber während des Wartens blockiert.

---

1 Z. B. die Weiternutzung vorhandener Schaltkreis- und Leiterplattenentwürfe (bei nur geringfügiger Umarbeitung), Prüfgeräte und Fertigungseinrichtungen, das Vermeiden von Lizenzkosten usw.

2 Zu hoher Stromverbrauch, zuviel Wärme (vgl. DirectRambus), schlechte Benchmark-Ratings (weil sich manche Anwendungen doch nicht so verhalten, wie es der Entwickler angenommen hatte) usw.

3 Es gibt auch starr formatierte Pakete, die keine beschreibenden Angaben benötigen. Beispiel: DirectRambus. Dualocts auf dem Datenbus, Zeilenpakete und Spaltenpakete passen jeweils in ein Raster aus 4 Taktimpulsen bzw. 8 Taktflanken.

Transaktionsorientierung, geteilte Übertragung (Split Transactions)

Der Grundgedanke: während des unvermeidlichen Wartens wird der Bus nicht blockiert, sondern anderen Einrichtungen zur Verfügung gestellt. Hierzu wird der Übertragungsvorgang aufgeteilt: die Auftragserteilung (Übermittlung der Adresse und der Steuerangaben, die die Art des Zugriffs bestimmen) wird von der eigentlichen Datenübertragung, von Zustandsmeldungen usw. getrennt. Beispielsweise läuft ein Lesezugriff vom Prozessor auf den Speicher so ab, daß der Prozessor zunächst nur einen Lese*auftrag* absetzt. Dann wird der Bus wieder freigegeben. Hat der Speicher die Daten zur Verfügung, so liefert er diese in einem weiteren Buszugriff an den Prozessor zurück. In der Zwischenzeit kann der Bus anderweitig genutzt werden.

Eine Vorstufe: verzögerte Übertragungen über den PCI-Bus (Delayed Transactions)
Verzögerte Übertragungen sind Bus-Zugriffe, die der Slave (das Target) nicht sofort ausführen kann. *Ablaufschema:*

- der Master erhält den Bus zugesprochen und startet den Zugriff,
- das Target speichert die übertragenen Angaben (Adresse, Kommando, Byteauswahl usw.) und weist den Zugriff mit einer sog. Wiederanlauf-Endebedingung ab,
- das Target erledigt den Zugriff intern (d. h., es bereitet alles soweit vor, bis Lesedaten bereitstehen, Schreibdaten entgegengenommen werden können usw.),
- der Master wiederholt den Zugriff immer wieder,
- ist das Target intern noch nicht bereit, weist es wiederholte Zugriffsversuche des Masters weiterhin mit einer Wiederanlauf-Endebedingung ab,
- ist das Target intern bereit, so weist es den Zugriff nicht mehr ab, sondern führt ihn zu Ende.

Der Master muß abgewiesene Zugriffe zyklisch wiederholen, und zwar so lange, bis sie entweder erfolgreich zu Ende geführt worden sind oder mit einer anderen Endebedingung (kein Wiederanlauf) beendet werden.

PCI wurde an sich als (noch) herkömmliches Bussystem (mit ungeteilten Übertragungen) entwickelt. Man hatte aber die Notwendigkeit erkannt, zu vermeiden, daß einzelne Einrichtungen den Bus allzu lange belegen. Was im Rahmen dieses herkömmlichen Ansatzes noch nicht möglich war: daß eine zunächst als Slave (Target) ausgewählte Einrichtung plötzlich zum Master wird, um den ursprünglichen Master über die Erledigung des Auftrags zu informieren[1].

Deshalb die Kompromißlösung: das Target bleibt Target, hat jedoch die Möglichkeit, Zugriffe abzuweisen. Da es aber keinen Rückweg gibt, muß der (ursprüngliche) Master immer wieder anfragen, ob das Target mit seinem Zugriff endlich zu Stuhle gekommen ist. Das kostet Zeit und belegt unnötigerweise den Bus.

Split Transactions mit strikter Zugriffsreihenfolge - die Einfachlösung
Auftragserteilung und -erledigung sind voneinander getrennt. Die Aufträge müssen aber genau in der Reihenfolge erledigt werden, in der sie erteilt wurden (In-Order Transactions). Alle beteiligten Einrichtungen müssen hierzu die Aufträge mitzählen, deren Durchlauf verfolgen und genau wissen, was sie tun. Beispiele: AGP, DirectRambus, P6-FSB.

Die Vorteile:

- einfache Buchführung; es sind keine Auftragsnummern o. dergl. mitzugeben,

1 Das hätte bedeutet, jedes einschlägige Target zusätzlich mit Master-Eigenschaften auszurüsten. In den 90er Jahren wurde das offensichtlich als zu aufwendig angesehen.

- die gleichsam natürliche (womöglich vom Programmierer beabsichtigte) Zugriffsreihenfolge wird grundsätzlich eingehalten (kein Zusatzaufwand erforderlich, um sie zu überwachen und bei Bedarf zu erzwingen)[1],
- Zum Puffern genügen einfache FIFOs.

Nachteilig ist, daß Einrichtungen, die ihre Aufträge allzu langsam erledigen, den Betrieb aufhalten.

Split Transactions ohne Beschränkung der Zugriffsreihenfolge
Die Erledigung der Aufträge wird signalisiert, ohne auf die Reihenfolge der Erteilung zu achten (Out-of-Order bzw. Deferred Transactions). Es kann durchaus vorkommen, daß ein später erteilter Auftrag eher als erledigt signalisiert wird als ein vorher erteilter. Beispiele: die Bussysteme der neueren AMD-Prozessoren (Athlon, Duron usw.), P6-FSB (Deferred Transactions), PCI-X. Hierdurch wird der Bus nahezu optimal ausgenutzt. Vorausetzung ist aber, daß die Erledigungsreihenfolge tatsächlich keine Rolle spielt (was nicht immer der Fall ist). Zudem ist der Aufwand recht hoch:

- sowohl bei der Auftragserteilung als auch bei der Erledigung müssen Auftragskennzeichen (z. B. laufende Nummern) transportiert werden (Prinzip: „das ist Auftrag Nr. x von Einrichtung y"),
- FIFOs genügen nicht mehr. Vielmehr sind die Zwischenpuffer gemäß den Auftragsnummern zu verwalten (Adressierung über Auftragsnummer, Auffinden erledigter und noch ausstehender Aufträge, Vergabe von Auftragsnummern usw.). Ein solcher Puffer muß ähnlich einem Cache als Verbund von RAM und Assoziativspeicher ausgelegt werden, ist also eine ziemlich komplizierte Angelegenheit. Die Anzahl der Aufträge, die sich gleichzeitig in Bearbeitung befinden dürfen (Outstanding Transactions), ist deshalb beschränkt (auf typischerweise 4...8).

Wie wird dem Auftraggeber mitgeteilt, daß sein Auftrag erledigt wurde?
Beispiel: eine Speichereinrichtung hat einen Leseauftrag erhalten. Nach einiger Zeit sind die Daten endlich verfügbar. Wie bringen wir sie aber zum Auftraggeber? Es gibt folgende Grundsatzlösungen:

- die Einrichtung wird ihrerseit zum Master und fordert die Busherrschaft an. Die typische Lösung für Split Transactions ohne Beschränkung der Zugriffsreihenfolge (Beispiele: PCI-X, Deferred Transactions über den P6-FSB).
- alle Einrichtungen beobachten den Bus und halten sich an Busprotokolle, die gewährleisten, daß eine Einrichtung, die einen erteilten Auftrag erledigen will, dies auch tun kann. Funktioniert nur bei strikter Erledigungsreihenfolge (Beispiel: In-Order Transactions über den P6-FSB).
- die Einrichtung veranlaßt, daß dem ursprünglichem Master (dem Auftraggeber) die Busherrschaft wieder zugesprochen wird. Und der wartet nur darauf, um endlich die Daten zu liefern oder abzuholen (Beispiel: AGP).

3.2.3 Zusatzsignalwege

Zusatzsignalwege ermöglichen es, neue Aufträge zu erteilen, während gleichzeitig Datenübertragungen stattfinden, also neue Kommandos und Adressen parallel zu laufenden Datentransporten zu übertragen. Die Signalwege und Signalprotokolle sind typischerweise so ausgelegt, daß sich im Idealfall die Datenübertragungen lückenlos aneinanderreihen lassen. Prinzip: die Kommando- und Adreßübertragung darf nicht mehr Zeit kosten als die Übertragung eines Datenpakets. Da die Kommando- und Adreßangaben weniger Bits haben als ein Datenpaket, kommt man mit vergleichsweise schmalen Zusatzsignalwegen aus. Beispiele: DirectRambus, AGP.

---

1 Manchmal kommt es darauf an (z. B. bei der Ein- und Ausgabe, beim Speichern und Wieder-Lesen von Verarbeitungsergebnissen oder bei der Datenübergabe zwischen zwei Programmen), manchmal nicht (z. B. beim Schreiben in Bildspeicher).

Überlappte Belegung von Signalwegen
Es gibt Bussysteme, die auf den ersten Blick recht herkömmlich aussehen (sie haben Adreßleitungen, Datenleitungen, Steuersignale, Bestätigungssignale usw.). Die Bussignale sind aber in Gruppen eingeteilt, die jeweils unabhängig voneinander – vor allem aber in zeitlicher Überlappung – an verschiedenen Übertragungsvorgängen (Transaktionen) beteiligt sein können. So ist es möglich, über die Adreßleitungen bereits neue Zugriffe auszulösen, während über die Datenleitungen die Daten eines zuvor ausgelösten Zugriffs übertragen werden. Gleichzeitig können Bestätigungs- und Meldeleitungen Ergebnisse von Fehlerkontrollen und Zugriffsprüfungen wiederum anderer Zugriffe übertragen usw. Das gesamte Bussystem zerfällt also gleichsam in eine Anzahl spezialisierter Busstrukturen, die jeweils unabhängig voneinander Signale führen, die zu verschiedenen Zugriffsvorgängen gehören. Beispiel: P6-FSB.

3.3 Prinzipien der elektrischen Auslegung

3.3.1 Ausnutzung beider Taktflanken

Wenn wir beide Flanken eines Strobe- oder Taktsignals ausnutzen, erreichen wir eine Verdoppelung der Datenrate. Ganz so einfach ist es aber nicht, denn die Anforderungen an die Auslegung der Datenwege und an die Beschaffenheit des Taktsignals verschärfen sich erheblich:

- um eine Datenbelegung ab- und die nächste aufzuschalten, steht weniger als eine halbe Taktperiode zur Verfügung (das sind heutzutage bestenfalls wenige ns!),
- alle Datenleitungen müssen - mit sehr engen Toleranzen - gleiche Signallaufzeiten gewährleisten (alle zulässigen Toleranzen halbieren sich im Vergleich zu einer Auslegung mit einfacher Datenrate)[1],
- die Taktimpulse müssen exakt symmetrisch sein (Impulsdauer = Pausendauer = ½ Taktperiode).

Es gibt verschiedene Möglichkeiten, die Anforderungen zu erfüllen.

Taktsignale als komplementäre Signalpaare
Das Taktsignal wird über zwei Leitungen übertragen, die jeweils invertiert (komplementär) belegt werden. Ist das eine Taktsignal High, so ist das andere Low und umgekehrt. Zur Datenübernahme in Register wird dabei von jedem dieser Taktsignale die Low-High-Flanke ausgenutzt. Der Vorteil: was die Taktierung der Flipflops betrifft, so ergeben sich die gleichen Verhältnisse wie bei der einfachen Datenrate (Abb. 3.10).

Taktsignale als differentielle Signale
Das Taktsignal wird über zwei Leitungen übertragen, die jeweils komplementär belegt sind. Die Übernahmezeitpunkte werden aber nicht von jeweils einer Flanke allein bestimmt. Vielmehr wird ausgewertet, wann sich beide Flanken kreuzen (Abb. 3.11).

1 Und das ist noch eine ziemlich optimistische Annahme. Sollte sich die Setup-Zeit nicht auf weniger als die Hälfte senken lassen, so verringern sich die Zeitreserven für Sicherheitszuschläge noch stärker – es wird also unmöglich, sicherheitshalber einfach ein paar ns zuzugeben ...

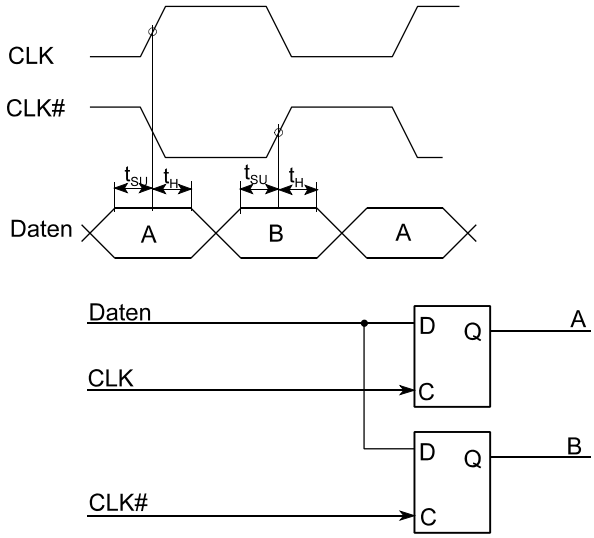


Abb. 3.10 Datenübernahme mit komplementären Taktsignalen. CLK, CLK# - komplementäre Taktsignale; t_{SU} - Setup-Zeit, t_H - Haltezeit. Man kann - von den Flipflops an - vorhandene (für einfache Datenrate vorgesehene) Entwürfe und Technologien weiternutzen, sofern $t_{SU} + t_H$ kürzer ist als eine halbe Taktperiode. Die einzige wirklich neue Anforderung betrifft das Umschalten der Datenbelegung beim Senden.

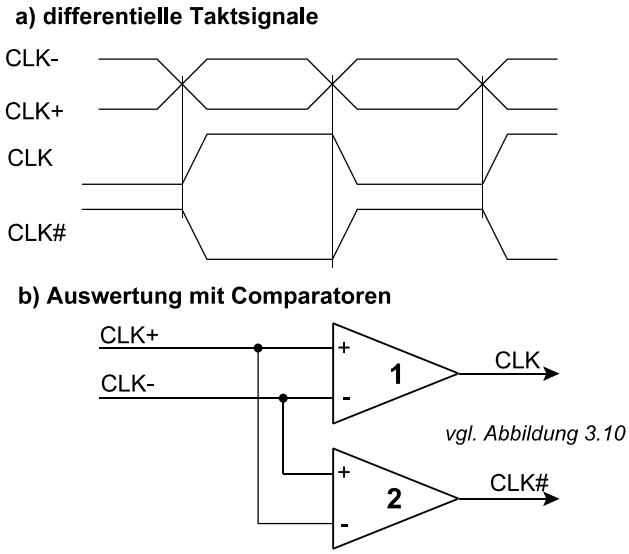


Abb. 3.11 Zur Auswertung differentieller Taktsignale

Die Comparatoren 1, 2 werten die Spannungsdifferenz zwischen beiden Taktsignalen CLK+, CLK- aus:

- Comparator 1 liefert ein High-Signal, solange CLK+ > CLK-,
- Comparator 2 liefert ein High-Signal, solange CLK+ < CLK-.

Die beiden Ausgangssignale CLK, CLK# werden gemäß Abbildung 3.10 zur Datenübernahme genutzt.

Taktregenerierung
Einfache Treiberstufen im Taktsignalweg kann man sich nicht mehr leisten; deren Verzögerung (vor allem: die Laufzeittoleranzen in Bezug auf die Daten und auf andere Taktsignale (Skew) wären viel zu hoch). Der Takt muß deshalb immer wieder neu gebildet werden. Die Grundlage: DLL- und PLL-Schaltungen.

3.3.2 Takt und Daten in gleicher Richtung

Es ist offensichtlich, daß man höhere Datenraten erzielen kann, wenn das Strobe- bzw- Taktsignal von der jeweils sendenden Einrichtung geliefert wird, so daß es zusammen mit den Daten gleichsam von der Quelle zum Empfänger fließt (Source Synchronous Clocking; Abb. 3.13).

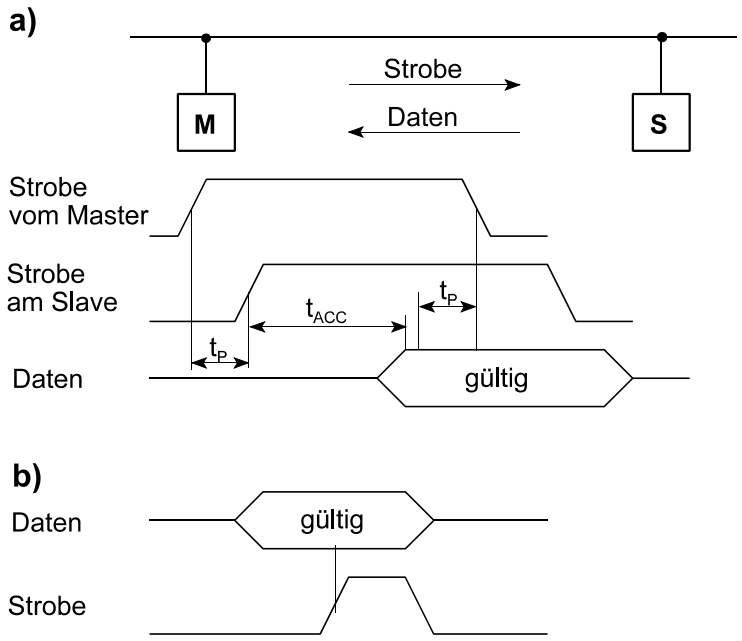


Abb. 3.12 Takt und Daten. M - Master; S - Slave.

a) Takt (Strobe) und Daten haben unterschiedliche Ausbreitungsrichtungen (zentraler Takt, vom Master gebildete Strobe-Impulse usw.). Bei der Festlegung der Zeitspezifikationen ist die Signallaufzeit über das gesamte Bussystem zu berücksichtigen. Der schlimmste Fall: das Lesen mit einem Strobe-Impuls, der vom Master kommt. Zunächst muß der Strobe-Impuls am Slave ankommen (eine Buslaufzeit t_P). Dann muß der Slave die Lesedaten auf den Bus aufschalten (Zugriffszeit t_{ACC}). Schließlich müssen die Daten am Master ankommen (eine weitere Buslaufzeit t_P).

b) Source Synchronous Clocking. Takt (Strobe) und Daten haben gleiche Ausbreitungsrichtung (die jeweils sendende Einrichtung liefert beides). Dann ist lediglich der gegenseitige Zeitversatz (Skew) zu berücksichtigen. Die Leitungslänge an sich spielt gar keine Rolle[1]. Idealerweise tritt die übernehmende Taktflanke genau in der Mitte des Intervalls auf, in dem die Datenbelegung gültig ist.

1 Voraussetzung: alle Leitungen sind nahezu gleich lang und werden gleichartig belastet.

3.3.3 Schnellere Strobes zur Datenübertragung

Bussteuerschaltungen sind im Grunde State Machines, also Flipflop-Anordnungen, die über kombinatorische Netzwerke rückgekoppelt sind. Je komplizierter die zu steuernden Signalspiele und Zustandsübergänge, desto umfangreicher die kombinatorischen Netzwerke, desto länger deren Durchlaufverzögerung. Wer das „Timing" eines Bussystems festlegt, muß auch hierauf Rücksicht nehmen, darf sich also nicht allein auf die Signalwege, die Treiberstufen usw. beziehen. Eine knifflige Sache: wenn man z. B. 100 MHz Taktfrequenz haben möchte, so muß man u. a. auch die Busprotokolle so hintrimmen, daß sich die erforderlichen State Machines tatsächlich mit der entsprechend geringen Durchlaufverzögerung fertigen lassen (mit anderen Worten: was nützt eine extreme Taktfrequenz, wenn die Steuerlogik nicht mitkommen kann?). Wirklich kompliziert sind Busprotokolle aber gleichsam nur am Anfang und Ende (Erlangen der Busherrschaft, Adressierung, Kommandoübertragung, Busfreigabe, Parken usw.). Die fortlaufende Datenübertragung ist hingegen einfach: Datenbelegung für Datenbelegung intern adressieren und auf den Bus legen oder vom Bus übernehmen und intern wegspeichern (z. B. in einen FIFO-Puffer). Solche Überlegungen haben die Entwickler dazu geführt, besondere Takt- oder Strobesignale allein für die Datenübertragung vorzusehen und diese mit entsprechend höheren Taktfrequenzen zu betreiben – nur die eigentliche Datenübertragung ist „richtig schnell", alle anderen Signalfolgen am Bus finden hingegen auf Grundlage vergleichsweise harmloser Taktfrequenzen statt, also zwischen etwa 66 und 133 MHz (Anhaltswerte). Beispiele: DDR-DRAM, AGP 2X und 4X, FSB des Pentium 4 und des Xeon.

Hinweis: Hier kommen die Vorteile der Paketorientierung zur Wirkung:

- ein Paket enthält nicht allzu viele Daten (so daß der gesamte Datenstrom ggf. in einen kostengünstigen FIFO paßt). Ansonsten müßten die Einrichtungen damit rechnen, daß mitten in der Datenübertragung Puffer vollaufen oder leer werden könnten, und sie müßten darauf entsprechend reagieren (z. B. durch Einfügen von Wartezuständen).
- es ist stets bekannt, wie lang das Paket ist (so daß beide Einrichtungen anhand einfacher Zählvorgänge erkennen können, wann die Datenübertragung zu beenden ist). Ist es hingegen so, daß nur eine der beteiligten Einrichtungen die Länge kennt (Beispiel: PCI), so muß die jeweils andere Einrichtung in jedem Datenzyklus damit rechnen, daß eine Endebedingung signalisiert wird (Verkomplizierung der Zustandsübergänge). Deshalb arbeiten moderne Hochgeschwindigkeitssysteme entweder mit festen Paketlängen (Beispiel: DirectRambus), oder beide Einrichtungen verständigen sich vor Beginn der Datenübertragung über die Anzahl der zu übertragenden Bytes (Beispiele: P6-FSB, AGP, PCI-X).

3.3.4 Geringer Signalhub

Es stehen verschiedene Systeme zur Wahl, die sich in den Pegeln und in der Auslegung der Buskoppelstufen unterscheiden (3,3-V-PCI, 1,5-V-AGP, SSTL, RSL, GTL, AGTL usw.). Typische Merkmale im Überblick:

- die Signalpegel werden auf eine Referenzspannung bezogen,
- die Auslegung der Treiberstufen richtet sich nach dem Anwendungsgebiet. Beispiele:
  - Tri-State-Stufen: für Bussysteme auf herkömmlicher Grundlage (PCI), für Punkt-zu-Punkt-Interfaces (AGP) und für Speicherbussysteme. Das obere Ende: SSTL-2 (ermöglicht Taktfrequenzen bis zu 200 MHz). Signalwege mit Stichleitungen (Stubs). Vergleichsweise viele Leitungen. Konfliktfeie Busumschaltung (noch) beherrschbar.
  - Open-Drain-Stufen (RSL, GTL, AGTL+ usw.): Bussysteme für extreme Datenraten (z. B. DirectRambus) und Multimaster-Bussysteme. Längere Stichleitungen werden vermieden. Überschneidungen sind nicht immer zu vermeiden (würde man einschlägig Vorsorge treffen, so wäre das mit längeren Latenzzeiten verbunden); deshalb wählt man Treiberstufen, die Überschneidungen aushalten.

3.3.5 Differentielle Signalübertragung

Je Signalweg werden zwei Leitungen vorgesehen, umd es wird die Spannungsdifferenz ausgewertet, die beide Leitungen gegeneinander führen (Abb. 3.13).

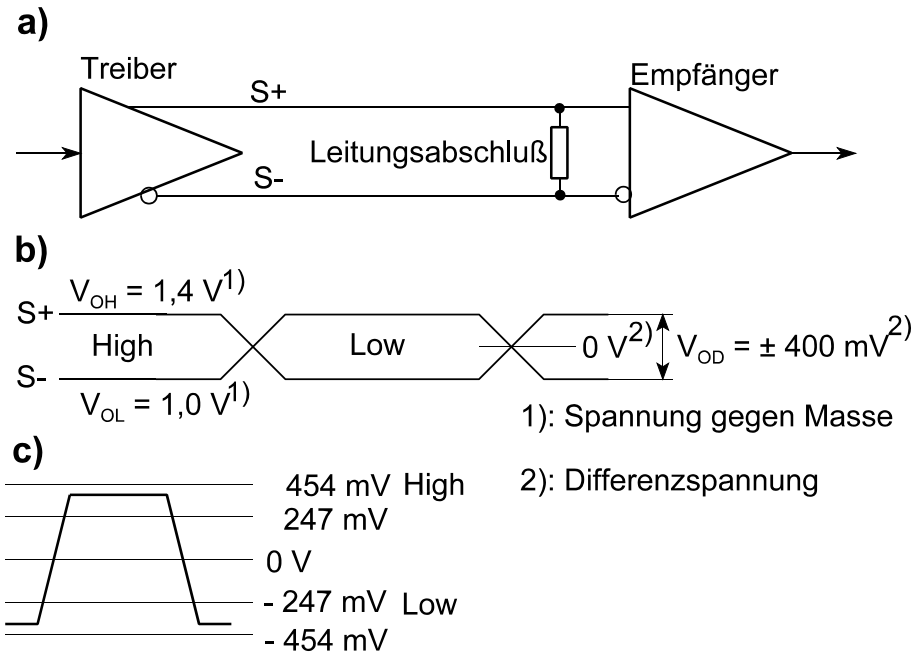


Abb. 3.13 Differentielle Signalübertragung (am Beispiel LVDS). LVDS = Low Voltage Differential Signaling. a) - Signalweg im Überblick; b) - Pegel auf beiden Signalleitungen (an den Treiberausgängen); c) - Logikpegel und Differenzspannung.

Maßgebend ist die Differenzspannung (Differential Voltage), nicht die Spannung gegen Masse (Gleichtaktspannung, Common Mode Voltage). Die Differenzspannung (der Signalhub) zwischen den Leitungen S+ und S- beträgt typischerweise 400 mV. Je nachdem, welche Signalbelegung übertragen werden soll, ist die Differenzspannung zwischen S+ und S+ entweder negativ oder positiv:

- Low-Signal: S+ ist um 247...454 mV negativer als S-,
- High-Signal: S+ ist um 247...454 mV positiver als S-.

Der wichtigste Zweck: Verminderung der Störempfindlichkeit:

- Störungen, die von außen einwirken, betreffen beide Leitungen gleichermaßen. Sie wirken sich nur auf die Gleichtaktspannung aus, die Differenzspannung hingegen wird nicht beeinflußt.
- jedes Signal hat gleichsam seine eigene Rückleitung. Differenzen zwischen den Massepotentialen von Treiber und Empfänger (Ground Shift) haben somit (nahezu) keinen Einfluß auf die Signalübertragung. Auch tragen die Signalrückströme nicht selbst zu solchen Potentialdifferenzen bei.

Der Vorteil: die Signale können auch mit sehr geringem Hub über größere Entfernungen (typisch sind 10...20 m) übertragen werden (Beispiele: USB, DVI, LVDS, SCSI, RapidIO u. a.)).

Der Nachteil: der Aufwand an Signalleitungen. Deshalb werden vor allem externe Interfaces (mit vergleichsweise wenigen Signalwegen) so ausgelegt. Ansonsten werden typischerweise nur besonders kritische Signale (z. B. Takte) differentiell übertragen (vgl. Abb. 3.11).

3.3.6 Mehr als zwei Signalwerte

Könnte man in einer Signalperiode einen von 4 Signalwerten übertragen, so würde das die Datenrate verdoppeln (ld 4 = 2 Bits je Signalperiode). Mit 8 Signalwerten je Signalperiode hätte man die dreifache Datenrate (ld 8 = 3 Bits) usw. Beispiel: QRSL mit 4 Signalwerten.

3.3.7 Möglichst wenige Signaländerungen (Transition Minimized Signaling)

Signaländerungen (= Flanken) bedeuten Stromverbrauch, Störstrahlung und Übersprechen. Deshalb sorgt man gelegentlich dafür, daß möglichst wenige Signale gleichzeitig schalten (Beispiele: (1) das digitale Video-Interface DVI, (2) der FSB des Pentium 4 und des Xeon, (3) AGP 3). Abb. 3.14 und Tabelle 3.2 veranschaulichen das Prinzip.

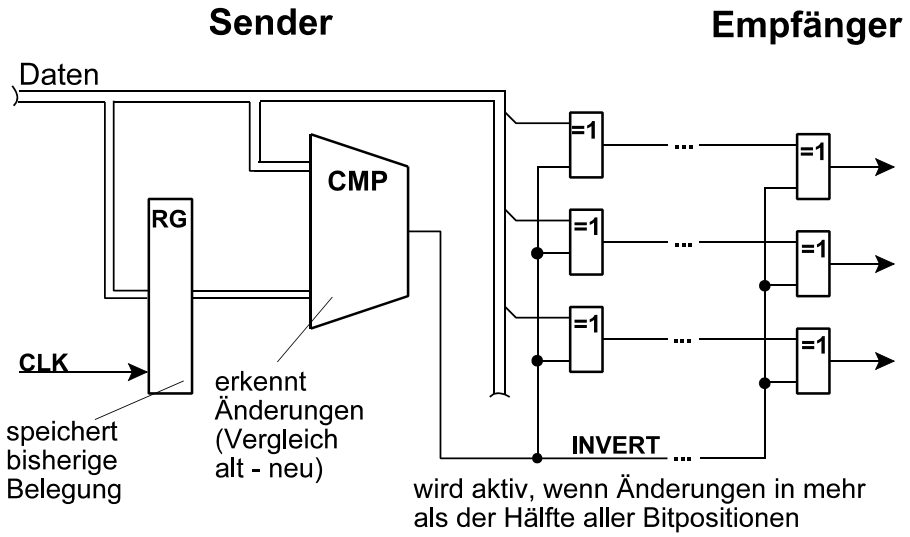


Abb. 3.14 Signalübertragung mit möglichst wenigen Signaländerungen (Transition Minimized Signaling TMS)

Die sendende Einrichtung vergleicht die aufzuschaltende Busbelegung mit der bisherigen Busbelegung. Es kommt darauf an, in wievielen Bitpositionen sich die Belegung ändert (von 0 auf 1 oder von 1 auf 0). Treten in mehr als der Hälfte aller Bitpositionen Änderungen auf, so wird die invertierte Belegung aufgeschaltet. Dies wird über ein zusätzliches Signal (INVERT) angezeigt. Die empfangende Einrichtung ist hierdurch in der Lage, die tatsächliche Belegung ggf. durch nochmaliges Invertieren wiederherzustellen. Die schaltungstechnische Grundlage: Antivalenzgatter (XORs) in den Signalwegen.

3.3.8 Pipelining

Betrachten wir nochmals Abb. 3.7. Eine wichtige Einzelheit tritt dort nicht hervor: es ist nämlich nicht so, daß das empfangene Signal einfach in ein Flipflop übernommen wird. Vielmehr gehören die Flipflops auf der empfangenden Seite zu State Machines, die aus der derzeitigen Belegung (dem aktuellen Zustand) und der ankommenden Busbelegung sowohl den neuen Zustand als auch die Belegung der in die Gegenrichtung führenden Bussignale bestimmen. Die hierzu erforderlichen kombinatorischen Netzwerke liegen im Signalweg vom Bus zum Flipflop (Abb. 3.15).

	Beispiel 1								Beispiel 2							
alt	1	0	0	0	1	1	0	1	1	0	0	0	1	1	0	1
neu	1	1	1	0	1	1	0	1	1	1	1	1	1	0	0	0
übertragen	1	1	1	0	1	1	0	1	0	0	0	0	0	1	1	1
INVERT	0								1							
tatsächliche Änderungen	2								5							
übertragene Änderungen	2								3							

Tabelle 3.2 Beispiele der Datenübertragung mit möglichst wenigen Signaländerungen

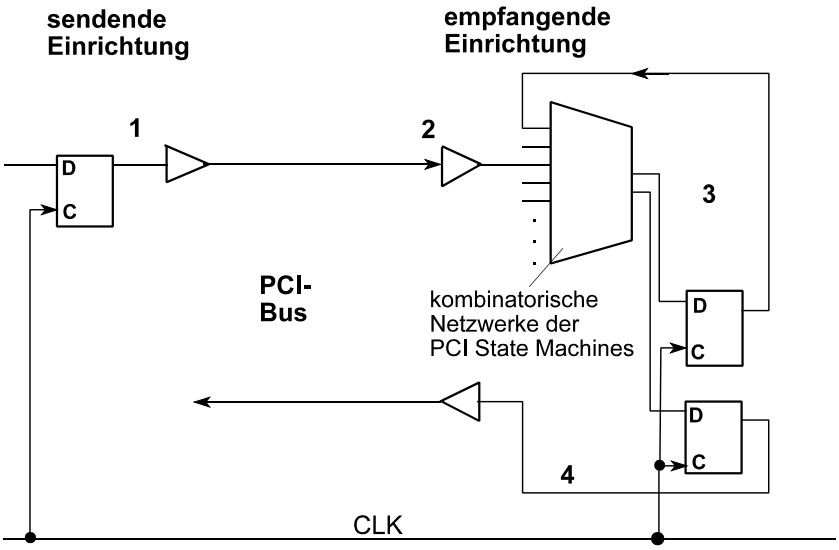


Abb. 3.15 Zum Zeitverhalten des PCI-Bus

1) mit der Low-High-Flanke des Taktes schaltet die sendende Einrichtung ihre Signale auf den Bus,
2) die Busbelegung breitet sich aus und kommt an der empfangenden Einrichtung an,
3) die empfangende Einrichtung muß darauf reagieren; über den betreffenden Zustandsübergang muß noch im selben Taktzyklus entschieden werden,
4) mit der nächsten Low-High-Flanke des Taktes schaltet die empfangende Einrichtung ihre Antwort auf den Bus.

Dieses Wirkprinzip steht offensichtlich einer weiteren Erhöhung der Taktfrequenz im Wege. Die Zeit, die zur Verfügung steht, um die kombinatorischen Schaltungen zu durchlaufen, ist ohnehin schon kurz genug (ca. 7 ns bei 33 MHz, ca. 3 ns bei 66 MHz; vgl. t_{SU} in Abb. 3.8).

Der Ausweg: wir schalten auf der empfangenden Seite ein weiteres Flipflop direkt in den Signalweg (Abb. 3.16). Die empfangende Einrichtung reagiert jetzt einen Taktzyklus später; die kombinatorischen Schaltungen haben aber nahezu einen ganzen Taktzyklus zur Verfügung (Pipelining; Register-Register-Busprotokoll). Diese Lösung wurde beispielsweise für den FSB der modernen Intel-Prozessoren und für das Bussystem PCI-X gewählt.

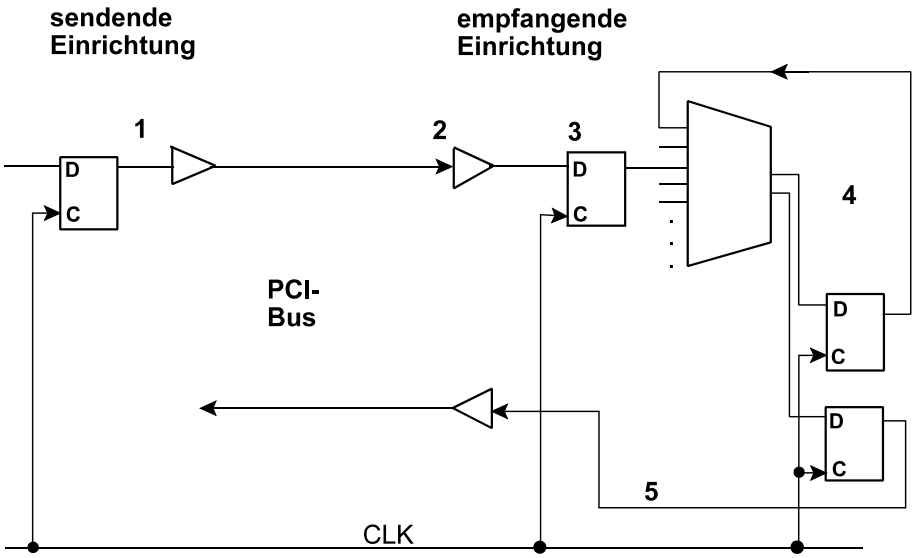


Abb. 3.16 Das Register-Register-Busprotokoll (PCI-X)

1) mit der Low-High-Flanke des Taktes schaltet die sendende Einrichtung ihre Signale auf den Bus,
2) die Busbelegung breitet sich aus und kommt an der empfangenden Einrichtung an,
3) mit der nächsten Low-High-Flanke des Taktes übernimmt die empfangende Einrichtung die Belegung in ihre Empfangsregister,
4) die empfangende Einrichtung ermittelt aus der Empfangsregisterbelegung und ihrem aktuellen Zustand den jeweiligen Folgezustand. Hierfür hat sie nahezu einen ganzen Taktzyklus Zeit.
5) mit der nächsten Low-High-Flanke des Taktes schaltet die empfangende Einrichtung ihre Antwort auf den Bus.

Auf diese Weise dauert jede Antwort zwar einen Takt länger, dafür steht aber nahezu eine ganze Taktperiode zur Auswertung zur Verfügung, so daß man es sich leisten kann, die Taktfrequenz zu erhöhen (Abb. 3.17).

3.3.9 Punkt-zu-Punkt-Interface oder Bus?

Die Auslegung als Punkt-zu-Punkt-Interface erlaubt es, die Anforderungen an die elektrischen Betriebskennwerte gleichsam zu entschärfen (mit den Vorteilen: weniger Siliziumfläche, geringerer Strombedarf, niedrigere Störintensität (Ground Bounce, EMI)). Das Problem: wir brauchen ziemlich aufwendige Verteilerschaltkreise in großen Gehäusen (wegen der vielen Anschlüsse).

Wodurch unterscheiden sich Busleitungen von „gewöhnlichen" Verbindungen zwischen Digitalschaltkreisen?

- an eine Busleitung sind mehrere Schaltkreise (Treiber oder Empfänger) angeschlossen. Das bedeutet mehrere „Stoßstellen" (genauer: Inhomogenitäten der Leitung) und unterschiedliche Signallaufzeiten, je nachdem welcher Treiber und welcher Empfänger gerade aktiv ist.
- die Busleitung kann von einem beliebigen Treiber angesteuert werden. Das heißt, sie wird nicht nur von einem Ende aus erregt.
- die vielen kapazitiven Belastungen (durch die angeschlossenen Einrichtungen, aber auch bereits durch leere (nicht belegte) Slot-Steckverbinder) führen zu einem vergleichsweise geringen Wellenwiderstand und zu einer geringeren Geschwindigkeit der Signalausbreitung.

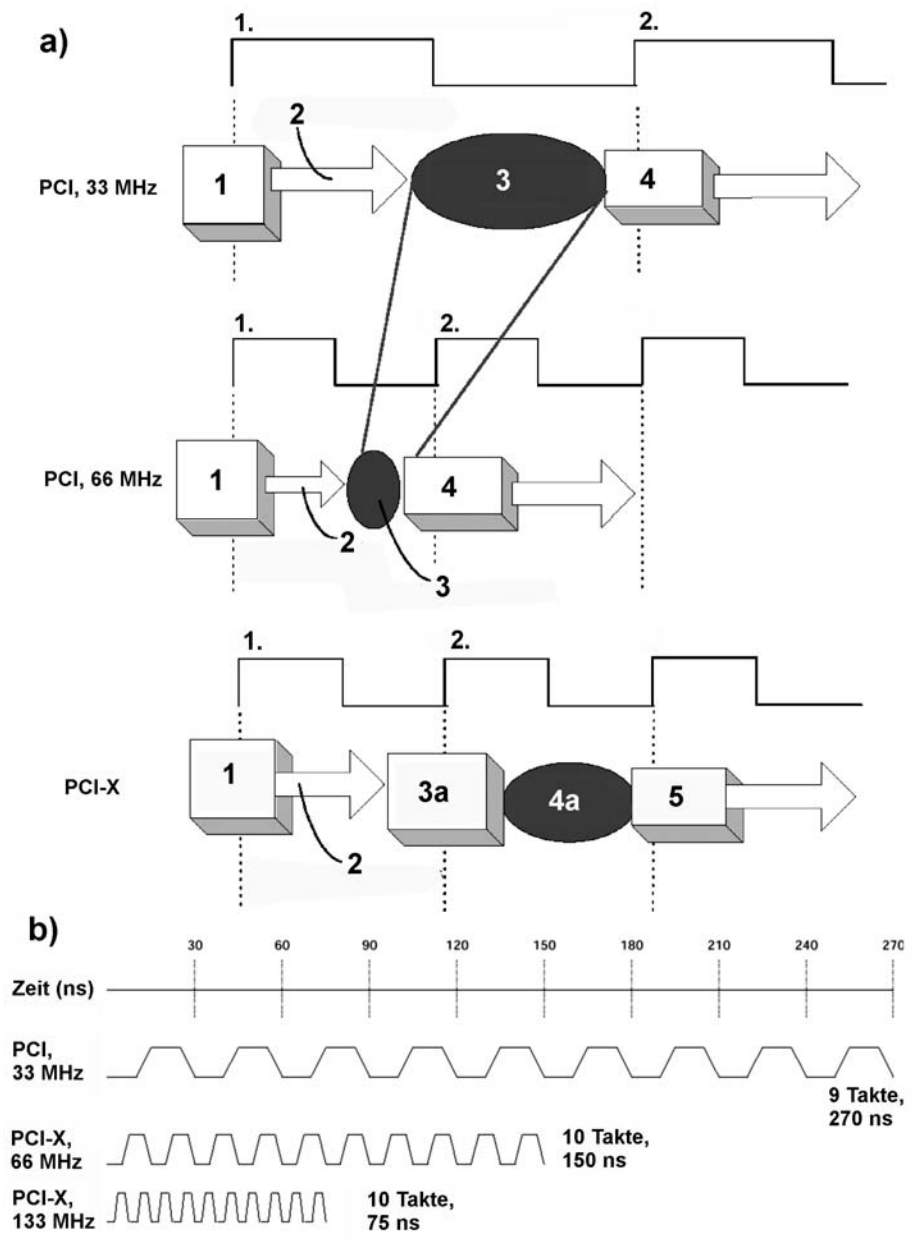


Abb. 3.17 Taktzyklen im Vergleich (Compaq). a) - Übersicht über die Zeitanteile im Taktzyklus; b) Zugriffsbeispiel. PCI kommt hierbei mit 9 Taktzyklen aus, PCI-X braucht zwar 10 Zyklen, ist aber, weil mit schnellerem Takt betrieben, insgesamt eher fertig. Die Zeitanteile im einzelnen (vgl. die Abbildungen 3.14 und 3.15): 1 - sendende Einrichtung schaltet Signal auf Bus; 2 - Signal läuft über die Busleitung; 3 - Durchlauf der Kombinatorik in der empfangenden Einrichtung; 4 - empfangende Einrichtung schaltet Antwort auf Bus. Was bei PCI-X anders läuft: 3a - Signal wird in Register übernommen; 4a - Durchlauf der Kombinatorik; 5 - empfangende Einrichtung schaltet Antwort auf Bus.