1. Bussysteme

Nachdem wir im Band I dieses Buches die Prozessoren als „Zentraleinheiten“ der Mikrorechner-Systeme beschrieben haben, wollen wir uns in diesem Kapitel nun mit den Verbindungswegen befassen, die es den Komponenten eines Mikrorechner-Systems ermöglichen, miteinander zu kommunizieren, d.h. Daten und Steuerinformationen untereinander auszutauschen. Mit Eigenschaften und Funktionen des grundlegenden Verbindungswegs, des Systembusses, haben wir uns im Abschnitt I.2.6[1] bereits ausführlich beschäftigt. Hier wollen wir zeigen, daß komplexe Rechnersysteme über eine umfassende Hierarchie von Bussen verfügen können, die sich in ihren technischen Realisierungen und ihrer Leistungsfähigkeit sehr stark unterscheiden.

1.1 Einführung

Im Bild 1.1-1 ist die allgemeine Verbindungsstruktur eines Rechensystems dargestellt. Alle Komponenten sind über Schnittstellen an einem Kommunikationsnetz angeschlossen. Durch das Netz werden ein physikalisches Übertragungsmedium sowie einen Satz von festen Regeln vorgegeben, nach denen die Kommunikation stattfinden muß, dem sog. Protokoll. Die Schnittstellen *(Interfaces)* dienen als Ankoppeleinheiten und übernehmen die mechanische, elektrische und zeitliche Anpassung an das Kommunikationsmedium. Häufig führen sie auch eine Codierung bzw. Decodierung der Daten durch.

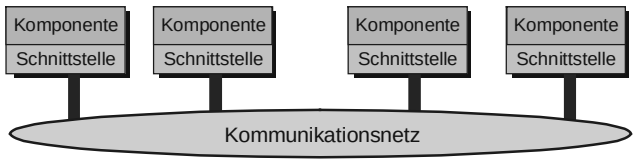

Bild 1.1-1: Allgemeine Kommunikationsstruktur eines Rechensystems

Das Kommunikationsnetz kann auf vielfältige Art und Weise realisiert werden. Die beiden Extreme bestehen auf der einen Seite aus dedizierten Verbindungswegen *(Point-to-Point)* zwischen allen Komponenten, auf der anderen Seite aus einem einzigen, gemeinsamen Verbindungsweg für alle Komponenten. Dieser gemeinsame Verbindungsweg besteht im einfachsten Fall aus einer „Sammelschiene“ von bis zu eini-

[1] Zur Erinnerung: ‚Abschnitt I.2.6‘ meint Abschnitt 2.6 in Band I.

gen Dutzend Leitungen zur Übertragung der Daten und Steuerinformationen und wird als (physikalischer) **Bus** bezeichnet. Es ist einsichtig, daß in einem sehr komplexen System dieser physikalische Bus zum „Flaschenhals“ *(Bottleneck)* wird. Durch Weiterentwicklung der grundlegenden Busstruktur wurde seit vielen Jahren versucht, diesen Engpaß zu vermeiden oder wenigstens zu erweitern. Diese verallgemeinerten Busstrukturen sind Gegenstand dieses Kapitels. Wir bezeichnen sie im folgenden als **logische Busse**[2]. Definierende Kennzeichen dieser Busse sind

- Existenz eines gemeinsamen physikalischen Verbindungsnetzes, an dem alle Komponenten angeschlossen sind,
- unterschiedliche mögliche Topologien,
- Selektion eines Kommunikationspartners über eindeutige Adressen[3],
- Vermeidung von Zugriffskonflikten bei mehreren Komponenten, die gleichzeitig aktiv auf den Bus zugreifen können.

Im Abschnitt 1.2 werden wir zunächst die allgemeinen Grundlagen zu Bussystemen darstellen. Abschnitt 1.3 beschreibt im Detail die „Hauptproblemkreise“ von Bussen: Übertragungsverfahren, Buszuteilung und Adressierung von Komponenten. Aktuelle Beispiele für Bussysteme in Arbeitsplatzrechnern, also in PCs oder Workstations, werden in den Abschnitten 1.5 bis 1.8 beschrieben. Im Abschnitt 1.9 werden wir dann ein Bussystem behandeln, das im Bereich der „eingebetteten“ Steuerungen *(Embedded Control)* große Bedeutung besitzt.

1.2 Grundlagen zu Bussystemen

In diesem Abschnitt werden wir die verschiedenen Busse definieren und klassifizieren, ihre Topologien betrachten und uns mit den Koppeleinheiten beschäftigen.

1.2.1 Definitionen und Klassifizierung

Im Bild 1.2-1 ist der Aufbau eines physikalischen Busses, d.h. eines Busses im engeren Sinne, dargestellt. Dieser Bus besteht (wie bereits oben gesagt) aus einer Gruppe von einer oder mehreren Leitungen, an die alle Komponenten des Systems angekoppelt sind.

Zu jeder Zeiteinheit kann höchstens eine Komponente Daten aussenden, also auch höchstens eine Datenübertragung stattfinden. Hingegen können eine einzige Komponente, mehrere Komponenten *(Multicast)* oder alle Komponenten *(Broadcast)* gleichzeitig die ausgesendeten Daten empfangen.

---

[2] Sie werden in der Literatur manchmal auch „virtuelle Busse“ genannt.
[3] Ausnahme: der CAN-Bus, s. Abschnitt 1.9

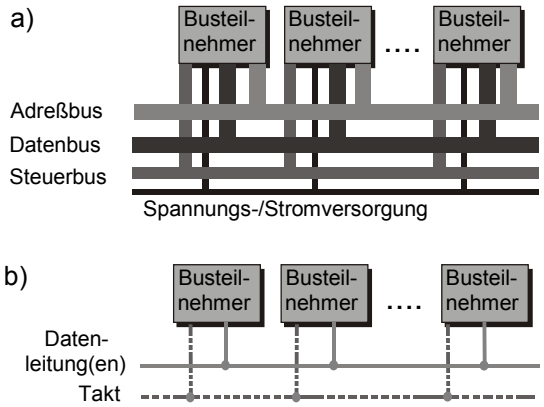

Bild 1.2-1: Aufbau eines physikalischen Busses; a) parallel, b) seriell

Im Bild 1.2-1a) ist ein paralleler Bus dargestellt. Über ihn werden die Daten „bit-parallel“, d.h. mehrere Bits simultan, übertragen. Der Parallelbus besteht aus den drei getrennten Teilbussen, die zur Übertragung von Adressen (Adreßbus), Daten (Datenbus) oder Steuerinformationen (Steuerbus) dienen. Dazu kommt noch eine mehr oder weniger große Anzahl von Leitungen zur Strom-/Spannungsversorgung. Typischerweise werden 4, 8, 16, 32 oder 64 Datenbits transferiert, also Halbbytes (*Nibbles,* Tetraden), Bytes, Wörter, Doppelwörter oder Quadwörter. Um Anschlüsse und Leitungen zu sparen, werden häufig Adressen und Daten (oder jeweils bestimmte Teilmengen davon) nacheinander über die gleichen Leitungen transferiert. In diesem Fall spricht man – wie bereits im Unterabschnitt I.2.6.5 gesagt – von einem Multiplexbus. Wesentliche Nachteile des Parallelbusses sind die begrenzte Länge der Leitungen sowie die relativ geringe Anzahl von zulässigen Anschlüssen. Die Sicherung gegenüber Übertragungsfehlern wird hauptsächlich durch Paritätsbits oder fehlererkennende/-korrigierende Codes (*Error Detecting/Correcting Code* – EDC/ECC) vorgenommen. Parallelbusse außerhalb von gedruckten Leiterplatten sind wegen der aufwendigen Kabel und Steckverbinder sehr teuer. Außerdem besteht die Gefahr des „Übersprechens“, d.h. der gegenseitigen Beeinflussung von Signalen, zwischen den Leitungen. Dennoch sind zum jetzigen Zeitpunkt die Parallelbusse immer noch die bevorzugten Bussysteme innerhalb von Rechnersystemen.

Bild 1.2-1b) zeigt einen seriellen Bus, bei dem die Kommunikation „bit-seriell“ über gemeinsame Leitungen für Adresse, Daten und Steuerinformationen geschieht. Im einfachsten Fall kommt ein serieller Bus mit zwei Leitungen aus, z.B. mit einer Datenleitung und einer Masseleitung. In diesem Fall, in dem ohne besondere Taktleitung gearbeitet wird, spricht man von einem asynchronen seriellen Bus. Hier muß die Synchronisation zwischen dem Sender eines Datums und dem Empfänger durch spezielle Steuerinformationen vorgenommen werden. Die übertragenen Informationseinheiten umfassen in der Regel nur jeweils ein einzelnes Zeichen. Beim synchronen seriellen Bus hingegen existiert entweder eine zusätzliche Taktleitung, oder der Takt wird in den Daten codiert übertragen und muß vom Empfänger daraus zurückgewon-

nen werden. Die Datenübertragung geschieht in formatierten Blöcken, sog. Paketen, welche die Daten in einen Rahmen aus Steuerinformationen, Sender- und Empfängeradressen und Prüfzeichen einbetten. Serielle Busse wurden zunächst hauptsächlich außerhalb eines Rechnergehäuses eingesetzt, u.a. zur Kommunikation zwischen Rechnern oder mit den Peripheriegeräten. Sie werden in naher Zukunft aber auch immer häufiger innerhalb von Rechnersystemen anzutreffen sein. Ihre wesentlichen Vorteile sind die erreichbaren, langen Übertragungsstrecken sowie die kostengünstigen Kabel und Steckverbinder. Die Datensicherung wird durch Paritätsbits, Prüfsummen oder zyklische Redundanzprüfungen (*Cyclic Redundancy Check* – CRC) unterstützt. Durch die Übertragung von Prüf- und Synchronisierinformation wird die erreichbare Nutzdatenrate z.T. erheblich herabgesetzt.

Komponenten, die an einem Bus angeschlossen sind und Daten senden oder empfangen können, nennen wir im weiteren **Busteilnehmer** oder **Knoten** *(Nodes)*. Wir unterscheiden aktive Knoten *(Master)*, die selbständige Buszugriffe durchführen können, und passive Knoten, deren Buszugriffe von einem Master gesteuert werden müssen. Ein **Multimaster-Bus** erlaubt den Anschluß von mehreren aktiven Knoten und erfordert den Einsatz von Komponenten und Mechanismen zur Regelung des konfliktfreien Buszugriffs *(Bus Arbiter)*. In diesen Bussen wird ein Master, der eine Datenübertragung (zum Senden oder Empfangen) auslöst, als ***Initiator*** bezeichnet, der ausgewählte Kommunikationspartner (Master oder Slave) heißt ***Target***.

Bussysteme aus einem einzelnen Bus der beschriebenen Form findet man gewöhnlich nur noch in einfachen Rechnern für Steuerungsaufgaben. Wie bereits erwähnt, stellt ihre beschränkte Busbandbreite in modernen Arbeitsplatzrechnern einen wesentlichen Engpaß für die Leistungsfähigkeit dar. Deshalb findet man in diesen Rechnern heutzutage ein hierarchisch strukturiertes Bussystem aus mehreren unterschiedlichen Bussen, wie es im Bild 1.2-2 dargestellt ist.

Die Busse werden je nach Einsatzart und Einsatzumgebung mit den folgenden Begriffen bezeichnet:

- **Prozessorbus** (synonym: Systembus, CPU-Bus, *Host Bus*, *Front-Side Bus – FSB*)
  Dies ist i.d.R. ein Parallelbus mit direkter Verbindung an die Prozessoranschlüsse. Die Datenübertragung wird durch den Prozessor selbst, einen Cache- oder DMA-Controller gesteuert. Dieser Bus ist zwar sehr schnell, erlaubt jedoch nur eine sehr geringe kapazitive Belastung und eine geringe räumliche Ausdehnung.
- **Speicherbus** *(Memory Bus)*
  Dieser Bus ist ein schneller Parallelbus zur Anbindung des Arbeitsspeichers an den Prozessor, der (häufig) mit derselben Übertragungsrate arbeitet wie der Prozessorbus. Er muß stets sehr kurz gehalten werden. In einfachen Systemen wird der Prozessorbus selbst als Speicherbus verwendet. Zur Reduzierung der Zugriffszeiten werden verstärkt Speicherbausteine mit speziellen Speicherbussen eingesetzt, deren Controller z.T. direkt im Prozessor integriert werden (vgl. den Rambus in Unterabschnitt 2.5.8).

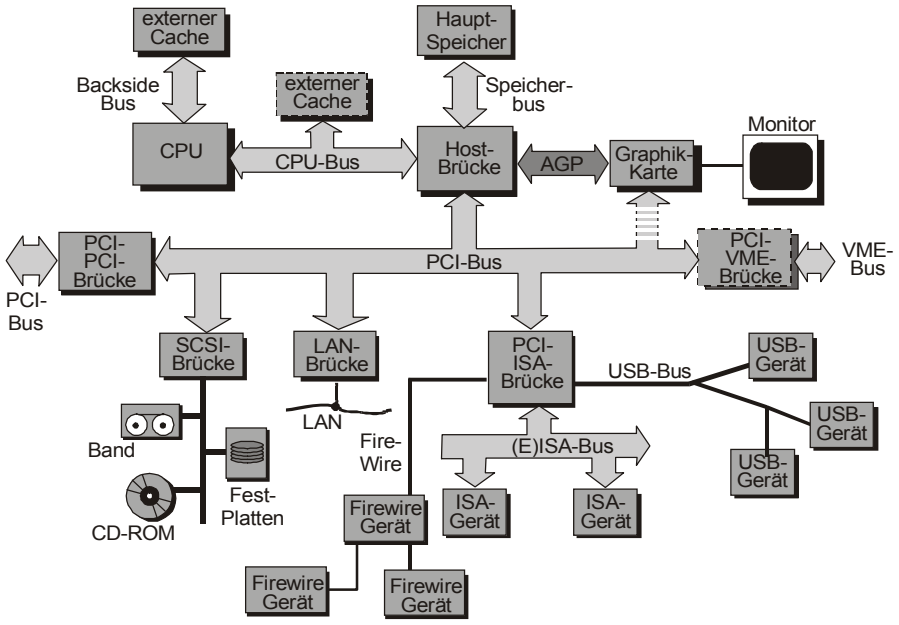

Bild 1.2-2: Typisches hierarchisches Bussystem

- **Peripheriebus** (Beispiel: PCI-Bus, synonym: lokaler Bus – *Local Bus*)
  Hierunter versteht man einen Bus zum Anschluß der internen und externen Peripheriekomponenten. Er stellt in gewisser Weise eine Erweiterung des Prozessorbusses zur Erhöhung der Anzahl der anschließbaren Komponenten und zur Vergrößerung der Entfernungen dar. Dazu muß er die sehr unterschiedlichen Bandbreitenanforderungen der angeschlossenen Geräte erfüllen. Die Kopplung zwischen Prozessor- und Peripheriebus geschieht über einen sog. Brückenbaustein.
- **Ein-/Ausgabebus** (synonym: Erweiterungsbus – *Expansion Bus*)
  Diese Busklasse dient zum Anschluß von externen Ein-/Ausgabegeräten oder Massenspeichern. Dazu gehören insbesondere ältere Standardbusse mit den entsprechenden Geräten, z.B. der (E)ISA-Bus *((Extended) Industry Standard Architecture)* oder der *Microchannel* (MCA). Ein-/Ausgabebusse werden über Brückenbausteine mit dem Peripheriebus verbunden.

In einer Bushierarchie steht auf höchster Stufe i.d.R. der CPU/Prozessorbus. Zur Erhöhung der kumulierten Busbandbreite können mehrere Busse derselben Klasse eingesetzt sein, z.B. mehrere Peripheriebusse (theoretisch z.B. bis zu 256 PCI-Busse, s. Abschnitt 1.5). Die Kopplung der verschiedenen Busse geschieht durch die bereits erwähnten Brückenbausteine *(Bridges)*, die im Unterabschnitt 1.2.3 kurz beschrieben werden.

1.2.2 Bustopologien

In diesem Abschnitt wollen wir kurz die gebräuchlichsten Topologien besprechen, die als logische Busse Verwendung finden. Sie sind im Bild 1.2-3 dargestellt.

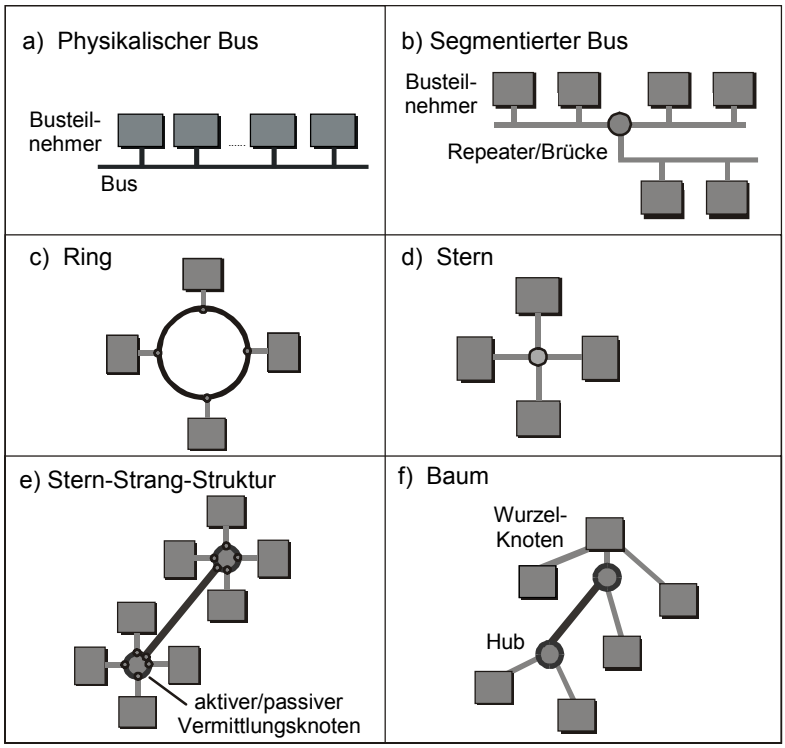

Bild 1.2-3: Die gebräuchlichsten Bustopologien

Bild 1.2-3a) zeigt den oben bereits beschriebenen **physikalischen Bus**, bei dem alle Teilnehmer an denselben Signalleitungen angeschlossen sind und der zu jedem Zeitpunkt nur eine Datenübertragung erlaubt. In Bild 1.2-3b) wird gezeigt, wie durch Koppeleinheiten mehrere physikalische Busse zu einem **segmentierten Bus** zusammengeschlossen werden. Durch *Repeater* werden Busse gleichen Typs verbunden (vgl. Unterabschnitt 1.2.3). Brücken *(Bridges)* koppeln Busse unterschiedlicher Klassen. Bild 1.2-3c) zeigt einen **Ring**, bei dem die Knoten durch unidirektionale Verbindungen *(Links)* nur mit ihren direkten Nachbarn verbunden sind. Jeder Anschluß dient dabei als *Repeater* und gibt die empfangenen Signale verstärkt und zeitlich regeneriert auf die Ausgangsverbindung aus. Je nach Länge der *Links* und ihrer Übertragungsraten können ein oder mehrere Daten gleichzeitig im Ring kursieren. Aus Kapazitäts- und Fehlertoleranzgründen werden oft zwei (gegenläufige) Ringe eingesetzt.

Im Bild 1.2-3d) ist ein **Stern** *(Star)* gezeigt, bei dem alle Komponenten über dedizierte Verbindungsleitungen an einem Vermittlungsknoten angeschlossen sind. Die Kommunikation findet nur über diesen Vermittlungsknoten statt. Dieser kann in passiver Form vorliegen, d.h. er muß durch die angeschlossenen Knoten gesteuert werden und kann keinerlei eigene Verarbeitung vornehmen. Oder er ist als aktiver Knoten ausgelegt, der die Vermittlung zwischen den Anschlußleitungen selbst vornimmt und dabei zusätzliche Funktionen ausführt – wie z.B. die Zwischenspeicherung der Daten. Je nach Realisierung kann der Vermittlungsknoten zu jeder Zeiteinheit höchstens ein Datum übertragen oder aber mehrere Datenübertragungen zwischen disjunkten Knotenmengen gleichzeitig durchführen.

Die Sterntopologie hat in Form der sog. ***Switches*** in den lokalen Netzen eine große Bedeutung gewonnen. In der Ausprägung als *Crossbar Switches* (Kreuzschienenschalter[1]) wird sie auch verstärkt in Rechensystemen eingesetzt. Die **Stern-Strang-Topologie** nach Bild 1.2-3e) verbindet die Vermittlungsknoten mehrerer Sterne durch zusätzliche Verbindungsleitungen (Stränge). Dabei kann sich die Übertragungskapazität der Stränge sehr stark von derjenigen der Verbindungen in den Sternen unterscheiden; sie muß jeweils nach dem Kommunikationsaufwand zwischen den Sternen festgelegt werden. Die Stern-Vermittlungsknoten können aktiv oder passiv sein.

In Bild 1.2-3f) wird die **Baum-Topologie** *(Tree)* skizziert. Sie koppelt die Komponenten in Form eines Wurzelbaums, wobei die Übertragungswege („Zweige") bidirektional arbeiten. Alternativ kann die Kommunikation nur zwischen dem Wurzelknoten und einem der (Blatt-)Endknoten oder aber auch zwischen beliebigen Knoten stattfinden. Die Verzweigungen werden durch sog. *Hubs* realisiert. Als „fetter Baum" *(Fat Tree)* besitzt der Baum Zweige, deren Übertragungskapazitäten mit größerer Nähe zum Wurzelknoten *(Root Node)* steigen und dadurch dem erhöhten Kommunikationsanforderungen zur Wurzel Rechnung tragen.

1.2.3 Koppeleinheiten

Im vorletzten Unterabschnitt 1.2.1 wurden bereits zwei der Einheiten erwähnt, die zur Kopplung von Bussen dienen: Die einfachste Form ist der ***Repeater,*** der zur Verbindung zweier identischer Busse dient. Dabei findet im wesentlichen nur eine elektrische Verstärkung, zeitliche Regenerierung und (Re-)Synchronisierung der Bussignale ohne Pufferung der Daten statt. Alle Komponenten benutzen die gleiche Adressierung und teilen sich denselben Adreßraum.

Aufwendiger aufgebaut als der Repeater ist ein ***Hub*** (Nabe, Mittelpunkt), wie er in Bild 1.2-4a) skizziert ist. Vergleichbar mit einem Repeater hat er die Kopplung identischer Busse mit gleicher Adressierung und Adreßraum zur Aufgabe, nun aber eines hierarchisch übergeordneten Busses *(Upstream)* mit mehreren untergeordneten Bussen *(Downstream)*. Zur Erfüllung seiner vielfältigen Funktionen besitzt er einen integrierten *Hub Controller*. Dieser überträgt Daten vom *Upstream Port* als Rundspruchdaten *(Broadcast)* zu den *Downstream Ports*.

[1] vgl. Unterabschnitt I.6.4.3

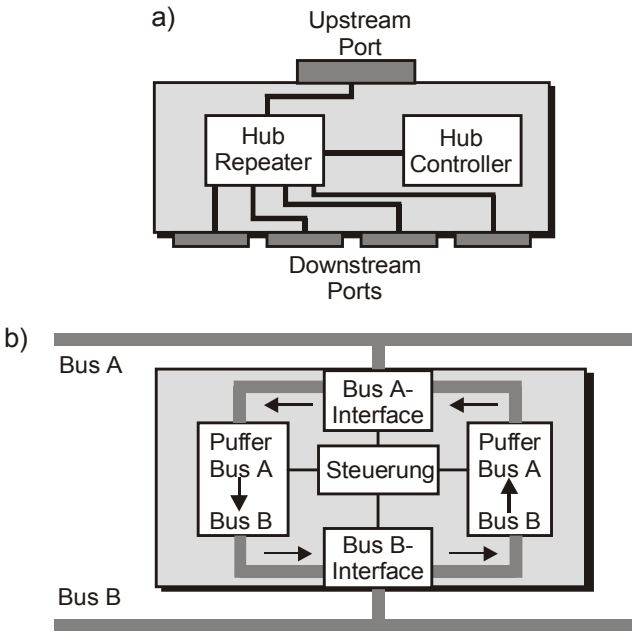

Bild 1.2-4: Aufbau eines Hubs (a) und einer Brücke (b)

Daten in umgekehrter Richtung werden je nach Netzverwaltung ebenfalls als Rundspruchdaten an alle anderen Ports (*Upstream* und *Downstream Ports*) geliefert oder aber durch Punkt-zu-Punkt-Kommunikation nur zum *Upstream Port* transferiert. Darüber hinaus muß der Hub aber auch selbständig den Aufbau und Abbau einer Verbindung sowie den Wiederaufbau einer unterbrochenen Verbindung durchführen. Er schaltet dazu die einzelnen Ports an und ab, setzt sie gezielt zurück und versorgt sie u.U. mit Strom und Spannung. Der Hub erkennt selbständig den Anschluß bzw. das Abtrennen von Geräten während des Betriebs *(Hot Plug & Play)* sowie das Auftreten von Fehlern und meldet dies an die zuständige Netzkomponente weiter.

Die komplexeste Form einer Koppeleinheit im Mikrorechner-Bereich[2] ist die **Brücke**, die in Bild 1.2-4b) dargestellt ist. Sie wird zur Kopplung zweier identischer oder unterschiedlicher Busse eingesetzt. Dazu besitzt sie die busspezifischen Schnittstellen beider Busse. Sie verfügt über Pufferspeicher zur zeitlichen und protokollspezifischen Anpassung der Datenübertragungen auf beiden Bussen. Dabei übernimmt sie eine Umsetzung und Filterung der angelegten Adressen, d.h. sie überträgt nur Daten, die für die am „Zielbus“ angeschlossenen Buskomponenten vorgesehen sind. Die verbundenen Busse können daher unterschiedliche Adreßräume unterstützen. In der Brücke sind häufig zusätzliche Komponenten, wie Cache-Controller, Speicher-Controller und weitere Schnittstellen integriert.

[2] Im Bereich der Rechnernetze gibt es noch weit komplexere Koppeleinheiten.

1.3 Konzepte für Bussysteme

In diesem Abschnitt wollen wir uns mit den wichtigsten Eigenschaften von Bussen beschäftigen und dabei insbesondere zeigen, wie der Datentransfer synchronisiert wird, wie die Busteilnehmer selektiert („adressiert“) werden und wie Zugriffskonflikte gelöst oder vermieden werden.

1.3.1 Abschätzung des Bandbreitenbedarfs

Zur Motivation der in den folgenden Unterabschnitten beschriebenen technischen Maßnahmen zur Beschleunigung des Datentransfers über Busse wollen wir zunächst für einige typische Anwendungen aus dem immer wichtiger werdenden Multimediabereich die Anforderungen an die Busbandbreite grob abschätzen. Unter Busbandbreite verstehen wir dabei den Durchsatz in byte/s oder bit/s[1], der mit einem speziellen Bus erreicht werden kann. Bereits in der Einleitung zu diesem Kapitel haben wir darauf hingewiesen, daß in modernen Arbeitsplatzrechnern das Bussystem ein „Flaschenhals“ sein kann – und nicht so sehr der Prozessor oder Graphik-Beschleuniger. Dies wird unmittelbar einsichtig, wenn man bedenkt, daß auch Prozessoren mit einem Arbeitstakt von vielen hundert MHz noch mit CPU- oder Peripheriebussen arbeiten müssen, die lediglich eine maximale Taktfrequenz von 33, 66, 100 oder 133 MHz erlauben.

Hier nun einige Beispiele für die Anforderungen an Busbandbreite:

- *High Quality Video:* Für die Übertragung von 30 Bildschirminhalten pro Sekunde mit jeweils 640×480 Bildpunkten und einer Auflösung von 24 bit pro Bildpunkt wird eine Bandbreite von 221 Mbit/s – also ca. 27,6 Mbyte/s – benötigt.
- *Reduced Quality Video:* Die Reduzierung auf 15 Rahmen mit 320×240 Punkten und 16-bit-Auflösung führt zu einem Bedarf von 18 Mbit/s, also 2,3 Mbyte/s.
- Videokonferenz: Hier wird für Übertragungen mit reduzierter Qualität in beide Richtungen wenigstens eine Bandbreite von 4,5 Mbyte/s verlangt.
- *High Quality Audio:* 44.100 Abtastwerte pro Sekunde mit jeweils 16-bit-Auflösung für zwei Stereokanäle ergibt einen Bandbreitenbedarf von 1,4 Mbit/s, d.h. ca. 176 kbyte/s.
- MPEG-2: Übertragungen von Videobildern, die nach diesem Verfahren komprimiert wurden, benötigen eine Bandbreite von ca. 72 Mbit/s, also 9 Mbyte/s.

Diese Beispiele zeigen jedoch nur die Bandbreite, die von den erwähnten Aufgaben selbst verlangt werden. Man darf dabei nicht vergessen, daß in modernen (Multimedia-)Anwendungen mehrere dieser Aufgaben simultan ausgeführt werden müssen. Dazu kommen dann u.U. noch Datenübertragungen von schnellen lokalen Netzanschlüssen (*Local Area Network* – LAN) sowie schnellen Peripheriespeichern, die sich

[1] Bei Übertragungsraten steht M... in diesem Buch für 10^6 – und nicht 2^{20} (vgl. Unterabschnitt I.1.1.4).

die begrenzte Bandbreite des Bussystems teilen müssen. Schon in naher Zukunft werden Hochleistungs-Graphiksubsysteme eine Übertragungsrate von bis zu 1,6 Gbyte/s verlangen.

1.3.2 Busankopplung

In diesem Unterabschnitt wollen wir uns in aller Kürze mit den schaltungstechnischen Grundlagen beschäftigen, die den Anschluß mehrerer Signalausgänge an einer gemeinsamen Busleitung ermöglichen. Im Bild 1.3-1 sind die am häufigsten eingesetzten Realisierungen von Treiberausgängen zur Busankopplung dargestellt. Bild 1.3-1a) zeigt[2] links einen Treiber mit *Open-Collector*-Ausgang, rechts mit *Open-Drain*-Ausgang. Im ersten Fall wird ein bipolarer, im zweiten Fall ein MOS-Transistor *(Metal on Silicon)* benutzt.

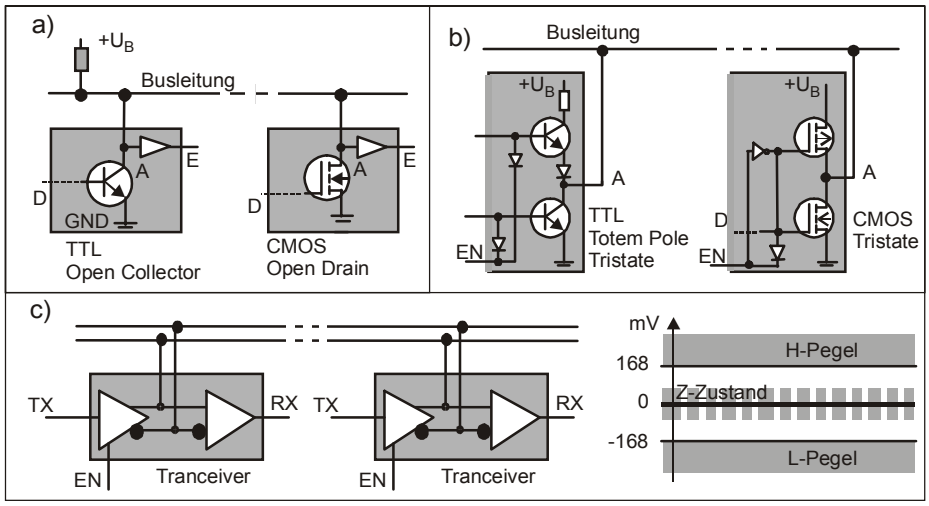

Bild 1.3-1: Möglichkeiten der Busankopplung

Beiden gemeinsam ist, daß durch einen einfachen leitenden Transistor die Busleitung auf Massepotential (*Ground* – GND) heruntergezogen wird, wenn der Steuereingang D des Transistors auf positivem Potential liegt. Liegt D auf niedrigem Potential, so sperrt der Transistor und der Ausgang A ist hochohmig von der Signalleitung getrennt. Durch den Zusammenschluß mehrerer Ausgänge wird eine logische Und-Verknüpfung, ein sog. *Wired-AND*, erzeugt, bei dem sich ein niedriger Signalpegel (L-Pegel) gegenüber jedem Ausgang mit positivem Potential (H-Pegel) durchsetzt. Der L-Pegel heißt deshalb **dominant**, der H-Pegel **rezessiv**. Elektrische Kurzschlüsse sind bei dieser Art von Treiberausgängen nicht möglich. Nachteilige sind jedoch die relativ großen Schaltzeiten. Im Bild 1.3-1a) ist außerdem angedeutet, wie bei einer bidi-

[2] als Wiederholung von Exkurs I.2.1.2.4

rektional betriebenen Leitung das Bussignal über eine Schaltung (dargestellt durch ein Dreieckssymbol) mit hochohmigem Eingang abgegriffen und als Eingangssignal E zur angeschlossenen Komponente geführt wird.

Bild 1.3-1b) zeigt sogenannte *Tristate*-Treiberausgänge[3]. Auch hier ist links wieder eine Realisierung mit bipolaren, rechts eine mit MOS-Transistoren gezeigt. Wie der Name andeutet, besitzen diese Ausgänge neben dem L-Pegel und H-Pegel noch einen dritten Zustand Z. In diesem Zustand ist der Ausgang A hochohmig gegen Masse und positive Betriebsspannung (+U_B), weil beide Transistoren des Ausgangs gesperrt sind. Der Ausgang wird in den Z-Zustand gesetzt, wenn der Steuereingang EN *(Enable)* auf Massepotential gezogen wird.

In seriellen Bussen werden Signale sehr häufig differentiell übertragen. Dies ist im Bild 1.3-1c) skizziert. Dabei wird das Sendesignal TX durch einen Verstärker mit komplementären Ausgängen auf ein Paar von Busleitungen gelegt. Im Bild 1.3-1c) sind rechts beispielhaft die Spannungsbereiche für die beiden möglichen Signalzustände ‚H-Pegel' und ‚L-Pegel' dargestellt. Zur Vermeidung von Kurzschlüssen muß bei dieser Form der Busankopplung durch die konfliktfreie Ansteuerung der Signale EN dafür gesorgt werden, daß zu jedem Zeitpunkt höchstens ein Treiberbaustein seine Ausgänge aktiviert. Als Empfänger wird ein sogenannter Komparator eingesetzt, der feststellt, auf welcher Leitung der höhere Signalpegel liegt und daraus das binäre Eingangssignal RX generiert. Für die Kombination von Sende- und Empfangsschaltung wird sehr häufig das Kunstwort *Transceiver* benutzt (aus *Transmitter* und *Receiver*). Zum Teil sind beide Schaltungseinheiten so ausgelegt, daß sie (ebenfalls) einen dritten Zwischenzustand Z auf den Leitungen erzeugen bzw. erkennen können. Dieser ist gegenüber den beiden Pegeln H und L rezessiv und dient zur Signalisierung bestimmter Situationen (vgl. Unterabschnitt 1.8.3).

Die differentielle Form der Signalübertragung bietet eine höhere Sicherheit gegenüber Störungen auf den Busleitungen, da sich viele Störungen auf beiden Leitungen durch eine Spannungsabweichung in derselben Richtung auswirken und so vom Komparator durch die Differenzbildung ‚herausgefiltert' werden können.

1.3.3 Synchronisations- und Übertragungsverfahren

Auf den verschiedenen parallelen Bussen in einer der erwähnten Bushierarchien werden z.T. dieselben Synchronisations- und Übertragungsverfahren angewandt wie beim Systembus. Aus Platzgründen wollen wir diese Verfahren hier nicht noch einmal beschreiben. Erwähnt werden soll nur, daß auch für die anderen Parallelbusse (Ein-/Ausgabebus, Peripheriebus, z.T. auch Speicherbus,...) zur Erhöhung der Übertragungsrate häufig eine Blockübertragung mit überlappender Adressierung eingesetzt wird *(Pipelined Burst Bus)*. Zur Reduzierung der Signalleitungen und der dazu benötigten Platinenfläche sind darüber hinaus wichtige, standardisierte Parallelbusse als Multiplexbusse ausgelegt, bei denen sich Adressen und Daten dieselben Leitungen teilen müssen (vgl. den PCI-Bus in Abschnitt 1.5).

[3] vgl. den Exkurs in Unterabschnitt I.2.6.2

Bei seriellen Bussen mit (relativ) niedriger Übertragungsgeschwindigkeit werden die Daten- und Taktsignale aus Kostengründen meist asymmetrisch *(single-ended)* übertragen, d.h. pro Signal wird nur eine Signalleitung benutzt[4]. Schnellere serielle Busse übertragen hingegen Daten- und Taktsignale in differentieller Form, wie es im Unterabschnitt 1.3.2 beschrieben wurde (vgl. auch die Abschnitte 1.7 – 1.9).

Bei der Bitübertragung auf synchronen, parallelen Bussen tritt (i.allg.) zwischen zwei ‚1'-Bits kein Signalwechsel auf, also kein Wechsel zum L-Pegel. Durch die Übermittlung des Bustaktes kann es dabei im Empfänger eines Datums nicht zu Interpretationsproblemen kommen. Bei der asynchronen seriellen Übertragung wird häufig ebenfalls ein Verfahren ohne Signalwechsel zwischen zwei ‚1'-Bits eingesetzt, das im Bild 1.3-2a) skizziert ist und **NRZI-Verfahren** *(Non Return to Zero Inverted)* genannt wird. Bei diesem Verfahren führt (nur) jedes ‚0'-Bit zu einem Signalwechsel.

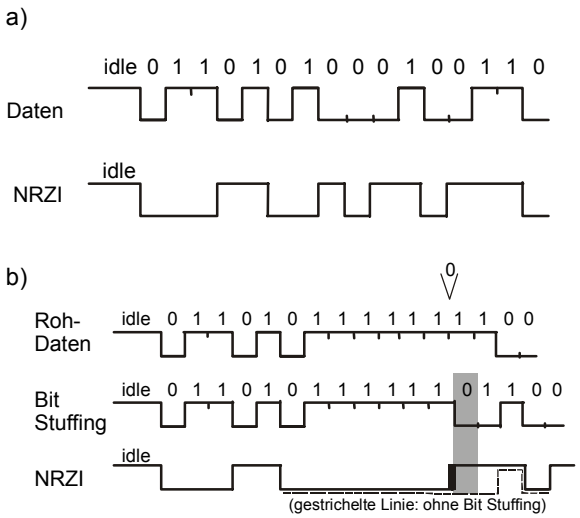

Bild 1.3-2: *Non-Return-to-Zero*-Übertragung (a) und *Bit Stuffing* (b)

Zur Vermeidung von zu langen Übertragungsketten von ‚1'-Bits ohne Signalwechsel muß der Sender eines Datums nach spätestens sechs ‚1'-Bits ein ‚0'-Bit (und damit einen Signalwechsel) einfügen. Zur Regenerierung der ursprünglichen Bitfolge entfernt der Empfänger seinerseits jedes ‚0'-Bit, das einer Kette von sechs ‚1'-Bits folgt. Dieses Verfahren wird als *Bit Stuffing* bezeichnet. Es ist im Bild 1.3-2b) gezeigt.

Wenn sich bei der synchronen seriellen Übertragung die Signale auf der Takt- und Datenleitung gleichzeitig ändern, kann das zu elektrischen Problemen führen (auf die wir hier nicht eingehen können). Im Bild 1.3-3 ist ein Übertragungsverfahren dargestellt, bei dem diese Probleme vermieden werden.

[4] nicht gerechnet die Signalmasse-Leitung(en)

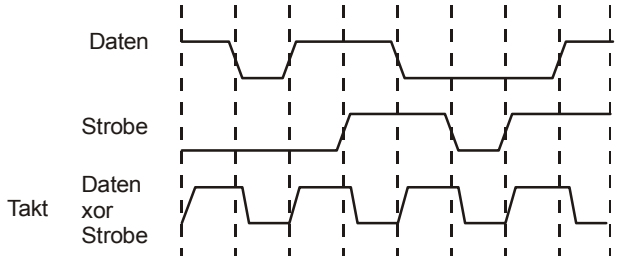

Bild 1.3-3: Übertragung mit Daten/Takt-Mischung

Hier wird der Takt durch ein *Strobe*-Signal ersetzt, das stets genau dann einen Signalwechsel aufweist, wenn sich das Datensignal nicht ändert. Der Empfänger kann durch einfache XOR-Verknüpfung (Antivalenz) den Takt aus den erhaltenen Daten- und *Strobe*-Signalen ableiten.

1.3.4 Adressierung der Buskomponenten

In diesem Unterabschnitt wollen wir nun die verschiedenen Möglichkeiten betrachten, mit denen ein Prozessor die anderen Busteilnehmer gezielt selektieren und in diesen spezielle Speicherzellen oder Register adressieren kann. Zunächst wollen wir uns mit den Varianten beschäftigen, die bei parallelen Bussen eingesetzt werden. Im Bild 1.3-4a) ist (noch einmal[5]) die Variante dargestellt, die hauptsächlich in einfachen Mikroprozessor-Systemen – z.B. Einplatinen-Computern – angewandt wird, in denen der Prozessor die einzige aktive Komponente mit selbständigem Buszugriff ist. Hier wird jede vom Prozessor ausgegebene Adresse durch einen zentralen Adreßdecoder ausgewertet, der aus den höherwertigen Adreßsignalen den angesprochenen Kommunikationspartner ermittelt und das ihm zugeordnete Selektionssignal aktiviert. Die niederwertigen Adreßsignale werden zur Adressierung einer Speicherzelle oder eines Registers direkt an die Komponente weitergeführt.

Im Bild 1.3-4b) kann auf einen zentralen Adreßdecoder dadurch verzichtet werden, daß in jede Komponente ein (dezentraler) Adreßdecoder integriert ist, der den für die Komponente relevanten Adreßraum erkennt. Größe und Lage des Adreßraums müssen der Komponente fest oder veränderbar einprogrammiert werden bzw. – z.B. über Schalter – einstellbar sein. (Ein Beispiel hierfür ist die Adressierung von Ein-/Ausgabeschnittstellen am ISA-Bus.)

In Bild 1.3-4c) werden die Datenleitungen selbst zur Adressierung der Komponenten benutzt, indem jeder Komponente eine Leitung eindeutig zugewiesen wird. Die Anzahl der anschließbaren Komponenten ist durch die Anzahl der Datenleitungen begrenzt. Außerdem setzt dieses Verfahren voraus, daß die Adressierungs-/Selektionsphase und die Datentransferphase zeitlich getrennt sind. (Diese Variante wird z.B. im SCSI-Bus eingesetzt – vgl. Abschnitt 1.6).

[5] vgl. Unterabschnitt I.2.6.6

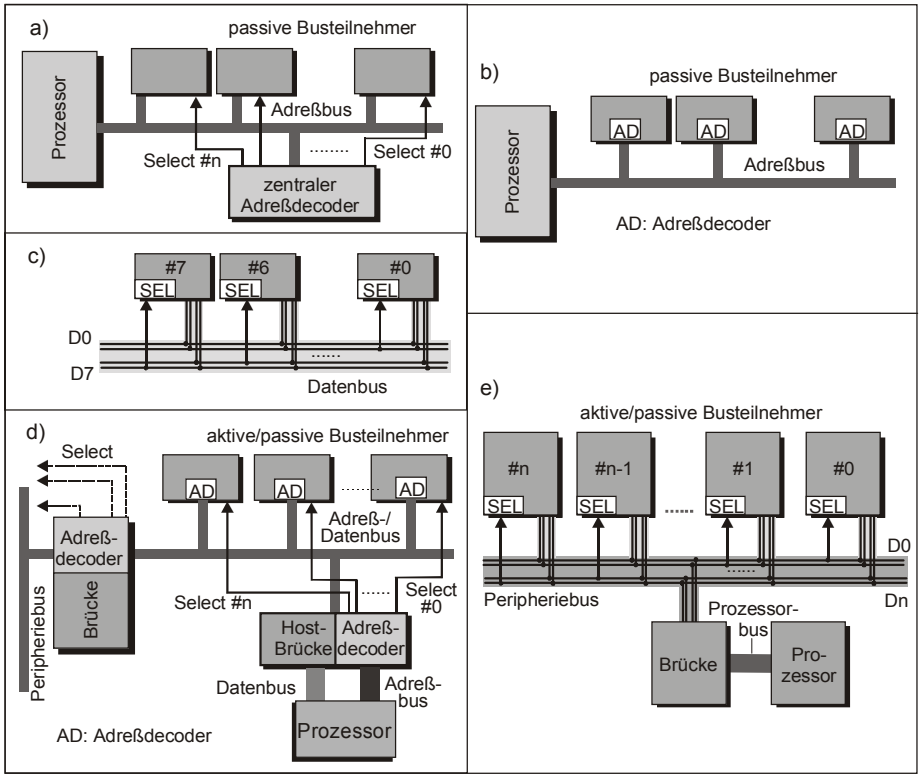

Bild 1.3-4: Mögliche Adressierungsverfahren bei parallelen Bussen

Eine Kombination der bisher beschriebenen Adressierungsvarianten ist in Bild 1.3-4d) und Bild 1.3-4e) skizziert, die sich an die Adressierungsmöglichkeiten des PCI-Busses orientieren (s. Abschnitt 1.5). Die Adressierung der Speicherzellen und Ein-/Ausgaberegister der Buskomponenten geschieht hierbei nach Variante b), d.h. jede Komponente verfügt über einen (dezentralen) Adreßdecoder. Der Registersatz, der zur Konfigurierung der Komponente dient, wird – je nach Implementierung – nach Variante a) oder c) adressiert: Nach a) übergibt der Prozessor die Registeradresse an die Host-Brücke, die den zentralen Adreßdecoder enthält und über das geeignete Selektionssignal die Komponente auswählt – sofern sie am angeschlossenen Peripheriebus liegt. Liegt sie hingegen an einem hierarchisch untergeordneten Peripheriebus, so wird die Registeradresse bis zur entsprechenden Brücke weitergereicht und erst vom Adreßdecoder dieser Brücke in ein Selektionssignal umgesetzt. Nach Variante e) wird jeder Komponente des Peripheriebusses eine Busleitung als Selektionssignal zugeordnet, das in der Adressierungsphase aktiviert werden kann. Durch diese Lösung wird die Anzahl der benötigten Anschlüsse der Brücke reduziert.

Im folgenden Bild 1.3-5 werden einige Adressierungsverfahren in seriellen Bussen dargestellt.

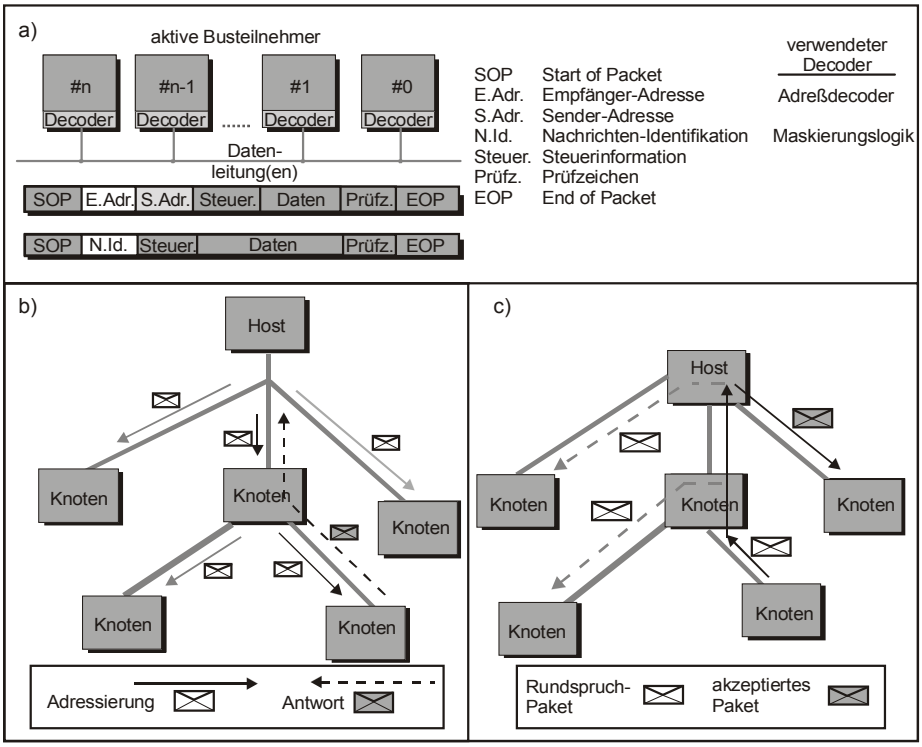

Bild 1.3-5: Adressierung in seriellen Bussen

Teilbild a) zeigt zunächst den typischen Aufbau eines Datenpakets, wie es über den seriellen Bus übertragen wird. Auf die Bedeutung der einzelnen Bitfelder im Paket wollen wir hier nicht eingehen (vgl. dazu die Abschnitte 1.7 – 1.9). Wichtig ist nur, daß im oberen Fall im Paket die Adresse des Empfängers übertragen wird. Durch diese Adresse kann eine einzelne Komponente, eine Gruppe von Komponenten *(multicast)* oder die Gesamtheit aller Komponenten *(Broadcast)* angesprochen werden. Im unteren Fall wird im Paket – anstelle einer Empfängeradresse – eine Nachrichtenidentifikation übertragen. Dieses Paket kann von allen Komponenten empfangen werden, für die diese Nachricht von Bedeutung ist. Die Entscheidung darüber wird durch eine Maskierungslogik im Empfänger getroffen.[6]

Bild 1.3-5b) zeigt die Adressierung von Komponenten in einem baumförmigen Netz, in dem die Kommunikation nur zwischen dem Wurzelknoten *(Host, Root Node*) und einer Komponente stattfinden kann (vgl. USB in Abschnitt 1.7). Der Wurzelknoten schickt jedes Paket als Rundspruch-Nachricht *(Broadcast)* an alle Komponenten. Diese werten die im Paket enthaltene Empfängeradresse aus. Nur der angesprochene Knoten quittiert den Empfang des Pakets durch ein Antwortpaket *(Acknowledge)*.

[6] Diese Art der Adressierung wird im CAN-Bus – *Controller Area Network* – zur Kopplung von Mikrocontrollern eingesetzt, vgl. Abschnitt 1.9.

Bild 1.3-5c) zeigt eine andere Variante der Adressierung in einem Baumnetz, bei der die Kommunikation zwischen beliebigen Knoten stattfinden kann. Der Sender eines Pakets verschickt dieses an seinen „Vaterknoten“ und alle seine „Söhne“. In jedem Zwischenknoten wird das Paket wiederum in alle Richtungen verteilt. Auf diese Weise „durchflutet“ ein Paket das gesamte Netz und erreicht so auch den vorgesehenen Empfänger. Nur von diesem wird es akzeptiert und durch ein Quittungspaket beantwortet, das wiederum das gesamte Netz durchlaufen muß. Eine ähnliche Form der Adressierung wird im FireWire benutzt (s. Abschnitt 1.8).

1.3.5 Buszuteilung

In diesem Unterabschnitt wollen wir uns nun mit Bussystemen beschäftigen, in denen mehrere Busmaster selbständig und unabhängig auf den Bus zugreifen können. Dies kann natürlich zu Konflikten führen, wenn mehr als ein Zugriff zur gleichen Zeit stattfindet. Wir werden zeigen, wie sich die Busmaster um den Bus bewerben können und wie ihnen der Zugriff gewährt wird.

Im Bild 1.3-6a) ist das Verfahren der unabhängigen Anforderung[7] *(Independent Request)* dargestellt. Im System gibt es einen zentralen Bus-Arbiter („Schiedsrichter“), der nach verschiedenen Verfahren[8] genau eine von mehreren zugriffswilligen Komponenten auswählt. Jede Komponente besitzt eine eigene Anforderungsleitung BRQi *(Bus Request)*, über die sie ihren Zugriffswunsch an den Arbiter melden kann. Über die individuelle Leitung *BGRi (Bus Grant)* wird ihr vom Arbiter die Genehmigung des Buszugriffs, die Buszuteilung, übermittelt. Von dieser Genehmigung darf eine Komponente jedoch erst dann Gebrauch machen, wenn sie anhand der gemeinsamen, bidirektionalen Leitung BUSY festgestellt hat, daß der vorhergehende „Businhaber“ den Bus freigegeben hat. Im Bild 1.3-6b) ist für einen etwas „komfortableren“ Arbiter, der den Zugriff zum Adreß- und Datenbus getrennt verwaltet, der Signalverlauf auf dem Bus dargestellt. In der Phase der Arbitrierung um den Adreßbus (..-AB) zeigen „interessierte“ Komponenten ihren Zugriffswunsch durch ihr Signal BRQ-AB an. Einen Takt später teilt der Arbiter einem Bewerber durch das Signal BGRi-AB der Komponente i den Zugriff zu, und die selektierte Komponente kann ihre Adresse auf den Adreßbus legen. Durch das Signal VA *(Valid Address)* zeigt die Komponente das Vorliegen einer gültigen Adresse an und fordert gleichzeitig den Datenbus an. Sobald der laufende Datenbus-Transfer abgeschlossen ist, teilt der Arbiter der Komponente i über BGRi-DB den Datenbuszugriff zu. Die zweite im Bild angegebene gültige Adresse kann von einer weiteren Komponente stammen, die über ihre individuellen Anforderungs- und Zuteilungssignale BRQ und BGR den Buszugriff erteilt bekommen hat.

Andererseits ist im Bild auch die Möglichkeit des *Bus Parkings* gestrichelt gezeichnet, bei welcher der aktuelle Busmaster den Zugriff solange behält, bis er von einem anderen angefordert wird. In diesem Fall kann die zweite Adresse auch ohne erneute Busanforderung vom letzten Busmaster geliefert werden.

---

[7] vgl. auch Unterabschnitt I.2.6.8
[8] auf die hier nicht näher eingegangen werden kann

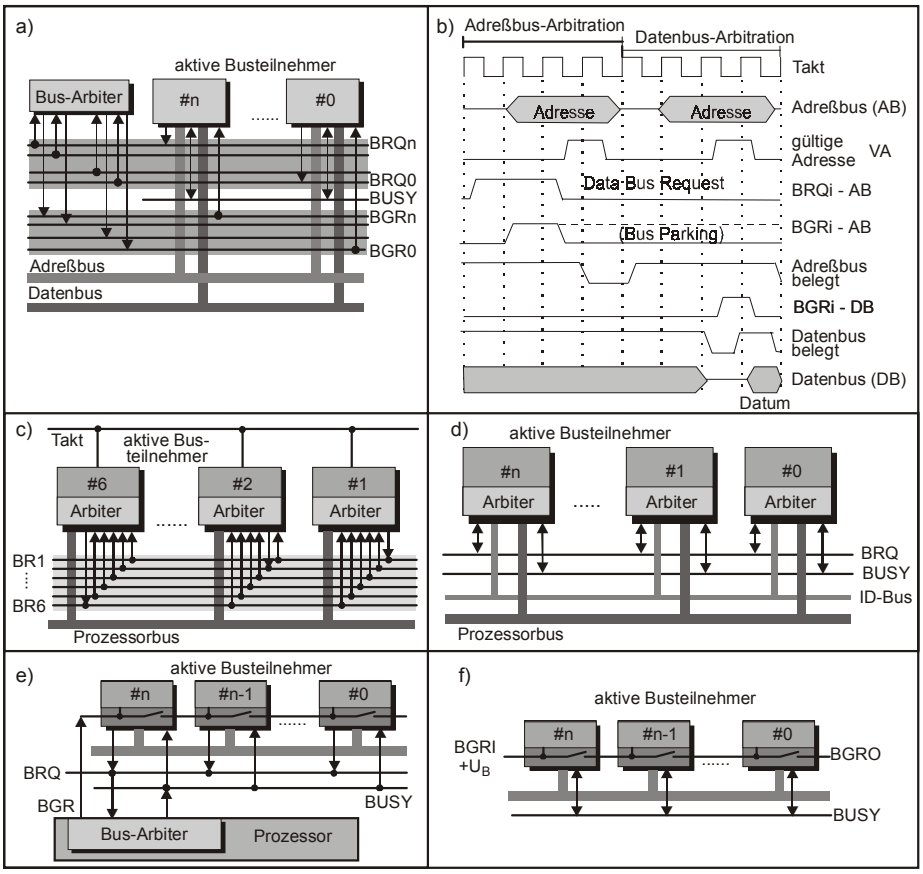

Bild 1.3-6: Arbitrierungsverfahren für Parallelbusse

Im Bild 1.3-6c) ist eine Variante mit dezentralem, in jedem Busteilnehmer implementierten Arbiter gezeichnet. Jeder Teilnehmer kann über eine spezielle Ausgangsleitung BRi *(Bus Request)*, die für alle anderen Teilnehmer eine Eingabeleitung ist, seinen Zugriffswunsch bei den dezentralen Arbitern anmelden. Auf der Basis einer gemeinsamen Strategie entscheiden diese über die Anforderung höchster Priorität. Alle „Verlierer“ bei dieser Entscheidung ziehen ggf. ihren eigenen Zugriffswunsch zurück. Der Vorteil dieses Verfahrens liegt darin, daß keinerlei externer Schaltungsaufwand getrieben werden muß. Jedoch ist der Gesamtaufwand durch die mehrfachen Arbiter sehr hoch. In der Regel kann durch dieses Verfahren nur eine relativ kleine Anzahl von Teilnehmern (z.B. bis zu 6) verwaltet werden.

Bild 1.3-6d) zeigt ein ähnliches Verfahren, bei dem (zur Erhöhung der Anzahl der anschließbaren Teilnehmer) alle Knoten ihre Anforderung über dieselbe Leitung BRQ melden. Gleichzeitig legen sie ihre eindeutige Kennung auf den ID-Bus *(Arbitration Identification Bus)*. Die Busankopplung geschieht dabei durch *Open-Collector*- bzw.

Open-Drain-Ausgänge, wie sie in Abschnitt 1.3.2 beschrieben wurden. Mit dem höchstwertigen Bit beginnend, prüft nun jeder Knoten bitweise, ob der Pegel auf der ID-Busleitung mit dem von ihm selbst ausgesandten Zustand übereinstimmt. Ist dies nicht der Fall, hat er selbst einen rezessiven Pegel ausgegeben und wenigstens ein anderer Teilnehmer einen dominanten. Als „unterlegener" Knoten zieht er seine Anforderung sofort zurück. Auf diese Weise setzt sich auf eindeutige Weise der Knoten mit den höchstwertigen dominanten Bits durch. Dieser Knoten muß wiederum solange warten, bis er anhand des BUSY-Signals die Beendigung des letzten Buszugriffs feststellt.[9]

Bild 1.3-6e) zeigt das sogenannte *Daisy-Chain*-Verfahren mit zentralem Arbiter, der z.B. in einem ausgezeichneten Busmaster integriert sein kann. Alle Knoten können über eine gemeinsame Leitung BRQ den Buszugriff beim Arbiter anfordern. Dieser bietet über die Leitung BGR der ersten Komponente den Zugriff an. Falls diese das Zugriffsrecht momentan nicht benötigt, reicht sie es an ihren „rechten" Nachbar weiter[10]. Auf diese Weise bekommt stets die erste Komponente in der Kette den Zugriff, deren sämtliche linken Nachbarn momentan keinen Zugriff wünschen – das heißt, die Priorität der Teilnehmer steigt mit ihrer Nähe zum Arbiter. Die Leitung BUSY hat hier dieselbe Funktion wie in den bisher beschriebenen Verfahren: Es zeigt an, wann ein zugriffsberechtigter Knoten sein Zugriffsrecht ausüben darf. Das Verfahren in Bild 1.3-6f) verzichtet auf den zentralen Arbiter, indem der linken Station das Zugriffsrecht – durch positives Potential am Eingang BGR – permanent zugeteilt wird. Wie eben beschrieben, wird es zum rechten Nachbarn weitergereicht, wenn momentan kein Zugriff benötigt wird.

Im Bild 1.3-7 sind gebräuchliche Arbitrierungsverfahren in seriellen Bussystemen skizziert.

Das Verfahren nach Bild 1.3-7a) wird beispielsweise im USB-Bus eingesetzt (s. Abschnitt 1.7). Der Host-Prozessor ermittelt im *Polling*-Verfahren mögliche Zugriffswünsche, indem er zyklisch, in regelmäßigen zeitlichen Abständen Abfragepakete an alle Knoten sendet. Die Reihenfolge der Abfrage ist zwar frei wählbar, liegt danach aber fest. Die selektierten Knoten senden entweder ein positives Antwortpaket, das bereits Daten enthalten kann, oder nur eine Quittung *(Acknowledge)*. Sie können aber auch mit einer ‚negativen' Quittung antworten und dadurch anzeigen, daß momentan kein Zugriffswunsch vorliegt, eine fehlerhafte Übertragung festgestellt wurde oder der Knoten nicht mehr betriebsbereit ist.

Bild 1.3-7b) zeigt ein Verfahren, bei dem ein beliebiger Knoten zu jeder Zeit einen Zugriffswunsch an den *Host* stellen darf. Dies geschieht z.B. dadurch, daß er eine Anforderung an seinen „Vater" schickt. Dieser reicht sie wiederum an seinen eigenen Vaterknoten weiter und blockiert gleichzeitig nachfolgende Anforderungen seiner anderen Söhne. Der Host entscheidet dann nach einer festgelegten Strategie nur noch über die gleichzeitig von seinen Söhnen übermittelten Anforderungen und weist durch eine spezielle „Information", die bis zu den anfordernden Knoten übertragen wird, alle nicht zugelassenen Anforderungen zurück. Die erforderlichen Informationen zur Busanforderung, Buszuteilung und Zurückweisung geschehen durch bestimmte Si-

[9] Dieses Verfahren wird z.B. im Multibus II der Firma Intel eingesetzt.
[10] Dies soll durch den Schalter in der Busschnittstelle symbolisiert werden.

gnalkombinationen auf den Busleitungen. Dazu unterstützen die Treiber und Komparatoren noch einen dritten rezessiven Zustand Z (s. Bild 1.3-7c), der während der Arbitrierung um die Punkt-zu-Punkt-Verbindung von einem Knoten und seinem Vater ausgegeben wird. Bei gleichzeitigem Zugriff setzt sich der Vater stets gegen seine Söhne durch.

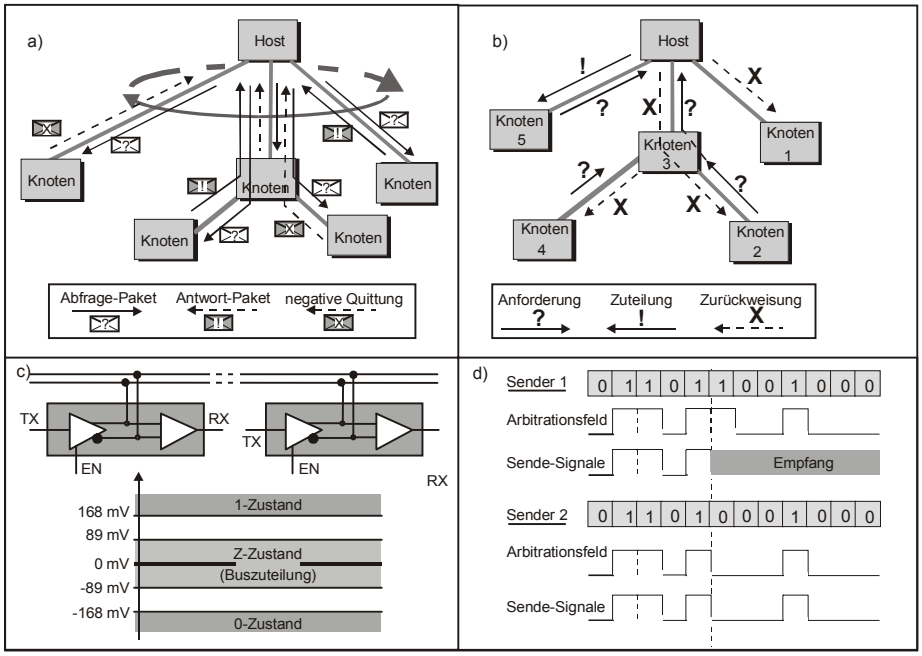

Bild 1.3-7: Arbitrierungsverfahren für serielle Busse

Bild 1.3-7d) zeigt die serielle Variante des Verfahrens nach Bild 1.3-6c). Es wird z.B. im CAN-Bus (vgl. Abschnitt 1.9) oder dem I^2C-Bus[11] eingesetzt und gehört zu der Klasse der *CSMA/CA*-Verfahren *(Carrier Sense, Multiple Access with Collision Avoiding)*, also den „Verfahren mit mehrfachem Buszugriff und Kanalabtastung mit Kollisionsvermeidung“. Diese Verfahren setzen voraus, daß sich mehrere potentielle Sender zunächst synchronisieren *(Multiple Access)*. Dann geben sie ihre Knoten- oder Nachrichten-Kennung Bit für Bit auf die Busleitung, wobei sie permanent den Zustand der Busleitung überwachen *(Carrier Sense)*. Dabei setzen sich die dominanten ‚0‘-Bits durch, d.h. es findet eine logische Und-Verknüpfung *(Wired AND)* statt. Ein unterlegener Knoten zieht sich sofort als Sender vom Bus zurück *(Collision Avoiding)* und „hört“ diesen nur noch ab. Er muß in diesem Fall später einen erneuten Übertragungsversuch starten.

[11] den wir hier aus Platzgründen nicht besprechen können

1.4 Zukünftige Entwicklung der Bussysteme

In den folgenden Abschnitten dieses Kapitels wollen wir nun verschiedene Bussysteme etwas näher betrachten. Dazu haben wir mit dem PCI- und SCSI-Bus zwei typische Parallelbusse ausgewählt, die in vielen Arbeitsplatzrechnern eingesetzt sind. Die beiden beschriebenen seriellen Busse – USB und FireWire – werden sicher in naher Zukunft eine immer größere Bedeutung erlangen. Im letzten Abschnitt betrachten wir dann mit dem CAN-Bus einen seriellen Bus, der speziell für die Kopplung von Mikrocontrollern entwickelt wurde. Zuvor wollen wir jedoch auf die mögliche weitere Entwicklung der Bussysteme in der Zukunft kurz eingehen.

Nach der Meinung vieler Experten wird der PCI-Bus bzw. PCI-X-Bus[1] (in Kombination mit dem AGP) in absehbarer Zukunft der „Bus der Busse“ bleiben. Im PC-/Workstation-Bereich besitzt er heute (noch) eine fast 100%ige Verbreitung. Er bietet – bezogen auf die Bandbreite – die geringsten Anschlußkosten, auch gegenüber seinen seriellen Konkurrenten. Um seine Nachfolge im PC-/Workstation-Bereich ‚tobt‘ jedoch bereits ein ‚Erbfolgekrieg‘, in dem sich maßgeblich die beiden konkurrierenden Prozessorhersteller AMD und Intel gegenüberstehen.

AMD hat bereits vor einiger Zeit mit der **HyperTransport-Technologie** seine Lösung präsentiert, die nur für den Einsatz auf einer ‚Mutterplatine‘ *(Motherboard, Mainboard)* – also in einem Gehäuse[2] – vorgesehen ist. Dabei handelt es sich um eine Punkt-zu-Punkt-Verbindung, die über zwei getrennte unidirektionale Pfade eine Vielzahl von Halbleiterbausteinen, also Prozessoren[3], Brücken, Speichereinheiten und Peripheriekomponenten, im Vollduplex-Betrieb miteinander koppelt. In Hypertransport-Systemen sind Taktfrequenzen bis zu 800 MHz möglich, wobei eine Zweiflanken-Übertragung *(Double Data Rate* – DDR, *Doubled Pumped)* eingesetzt wird. Die Übertragungsrate pro Richtung liegt damit bei 400 bis 1.600 MTransfers/s. Sie entspricht einer Datenrate von 100 bis 6.400 Mbyte/s für jede der beiden Richtungen einer Verbindung, also einer maximalen Gesamtrate von 12,8 Gbyte/s. Die Verbindungspfade sind als parallele Busse ausgelegt. Ihre Breite kann in Abhängigkeit von den angeschlossenen Komponenten 2, 4, 8, 16 oder 32 bit betragen, wobei die Verbindungen der beiden Übertragungsrichtungen unterschiedliche Breiten haben können. Die Übertragung der Informationen geschieht in Form von kurzen Paketen mit einer maximalen Länge von 64 Bytes. Dabei werden Anforderungs-, Anwort- und Rundspruchpakete unterschieden, die jeweils Befehle, Adressen und Daten enthalten können. Die Übertragung der Signale auf den Busleitungen geschieht differentiell mit niedrigen 1,2V-Spannungspegeln (*Low-Voltage Differential Signaling* – LVDS).

Die Konkurrenzlösung der Firma Intel zum HyperTransport wird als **3GIO** *(3rd Generation I/O)* bezeichnet. Wie die HyperTransport-Lösung ist auch 3GIO für den Einsatz in einer Vielzahl von Anwendungsumgebungen (Arbeitsplatzrechner, Kommunikationssysteme, Steuerungsanlagen,...) geplant, die sich durch sehr unterschiedliche Systemarchitekturen unterscheiden.

---

[1] PCI-X und AGP werden ebenfalls in Abschnitt 1.5 kurz beschrieben.
[2] *In the box.* Neben dem PC-Bereich ist der Bus auch für Netzwerkkomponenten, Telekommunikationsgeräte, Steuerungssysteme, Mobiltelefone, Geräte der Konsumer-Industrie u.a. vorgesehen.
[3] Der ‚standardisierte‘ HyperTransport ersetzt dabei den ‚individuellen‘ System-/Prozessorbus.

Im Bild 1.4-1 ist der mögliche Aufbau eines 3GIO-Systems im PC-/Workstation-Bereich skizziert. Hier soll der 3GIO eine Reihe der unterschiedlichen Busse (mit unterschiedlicher Leistungsfähigkeit) in einem hierarchischen Bussystem (vgl. Bild 1.2-2) ersetzen. Insbesondere soll er den ‚überalterten' PCI-Bus zur Kopplung der Brückenbausteine und seine Variante, den AGP *(Accelerated Graphics Port)*, zum Anschluß des Graphik-Subsystems ablösen. Dabei wird aber eingeplant, daß – wenigstens in einer Übergangsphase – der eingesetzte Chipsatz auch weiterhin den PCI-Bus (bzw. PCI-X-Bus) unterstützen wird, um den Einsatz älterer Komponenten zu ermöglichen.

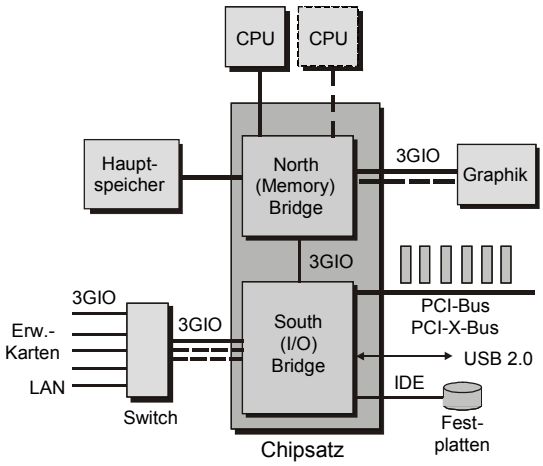

Bild 1.4-1: Aufbau eines 3GIO-Systems

Die 3GIO-Pfade stellen sehr schnelle, serielle Punkt-zu-Punkt-Verbindungen zwischen den Brückenbausteinen und den angeschlossenen Komponenten dar. Diese Verbindungen bestehen aus zwei Leitungspaaren, über welche die Signale differentiell übertragen werden. Dabei erfolgt die Übertragung unidirektional, im Vollduplex-Betrieb aber in beiden Richtungen. Als Verbindungsleitungen können Leiterbahnen auf der Platine, Kupferkabel oder Lichtwellenleiter eingesetzt werden. Die Übertragungsrate pro Leitungspaar soll anfänglich 2,5 Gbit/s betragen und in späteren Versionen bis zu 10 Gbit/s erreichen. Zur Erhöhung der Bandbreite einer Verbindung können die Übertragungspfade vervielfacht werden, also z.B. zwischen 2 und 32 Leitungspaare parallel betrieben werden[4]. Die Anzahl der Leitungspaare kann in beiden Übertragungsrichtungen unterschiedlich sein. Zum Anschluß einer Vielzahl von Peripheriekomponenten werden spezielle Schalteinheiten *(Switches)* eingesetzt, welche die externen 3GIO-Pfade untereinander oder mit den Brückenbausteinen verbindet. Diese Switches werden zunächst als eigenständige Einheiten realisiert, später sicher auch in den Chipsatz integriert werden.

[4] Da die 8 Bits eines Bytes in 10 Bits codiert übertragen werden, benötigt man für die Parallelübertragung eines Bytes 5 Leitungspaare.

Erste Implementierungen von 3GIO werden erst ab 2003 erwartet. Bis dahin wird unklar bleiben, welcher der beiden Kontrahenten, HyperTransport oder 3GIO, den ‚Krieg' gewinnen wird. Es ist jedoch nicht auszuschließen, daß beide nebeneinander bestehen können – wenn auch nicht in gleichwertiger Weise. So wird auch für möglich gehalten, daß die Rolle der HypertTransport-Technologie auf die Verbindung zwischen Prozessor und Host-Brücke bzw. zwischen den Brücken beschränkt wird, während alle anderen Rechnerkomponenten – auf der Hauptplatine und auf den Einsteckkarten *(Add-in Cards)* – über den 3GIO angeschlossen werden.

Allgemein ist bei den momentanen und zukünftigen Entwicklungen ein eindeutiger Trend zu den seriellen Bussen festzustellen. Zum Anschluß externer Komponenten und Geräte haben sich – im Bereich der Arbeitsplatzrechner – in den letzten Jahren der USB und (etwas verzögert) der FireWire durchgesetzt, wobei nach Expertenmeinung beide Busse nebeneinander – häufig sogar im selben Rechner – auftreten und dabei für jede Geschwindigkeitsanforderung die kostengünstigste Lösung bieten werden. FireWire wird insbesondere im Bereich Multimedia und Konsumelektronik *(Consumer Electronics)* eine überragende Rolle zugetraut. Für Festplattensysteme werden im Arbeitsplatz-Rechnerbereich auch dem *Fibre Channel* (s.u.) gute Prognosen gestellt. Neben seinen seriellen „Ablegern", dem Fibre Channel und FireWire, wird aber wohl die Bedeutung des parallelen SCSI-Busses durch erneute Leistungssteigerungen (auf 640 Mbyte/s bereits in naher Zukunft) erhalten bleiben.

Wir wollen hier, eher stichwortartig, noch einige weitere Busse beschreiben, die in Arbeitsplatzrechnern oder Servern eingesetzt werden.

- Der *S-Bus (System Expansion Bus)* der Firma SUN ist ein Ein-/Ausgabebus im Server- und Großrechnerbereich. Er dient hauptsächlich dem Anschluß von fehlertoleranten Peripheriekomponenten in SPARC-Rechnern. Ursprünglich wurde er über den M-Bus *(Module Interconnect Bus)* von SUN mit den Prozessoren gekoppelt; heute geschieht dies meistens über den PCI-Bus. Der S-Bus besteht in der nicht gemultiplexten Ausführung aus einem 32-bit-Datenbus und einem 28-bit-Adreßbus. In der Multiplexform werden über gleiche Leitungen 64 Datenbits mit den 28 Adreßbits übertragen. Der Bustakt beträgt 16 bis 25 MHz. Im *Burst Mode* erreicht der S-Bus eine Transferrate von 95 Mbyte/s über einen 32-bit-Datenbus und 190 Mbyte/s über einen 64-Bit-Datenbus.
- Obwohl sie hauptsächlich in RAID-Plattenstapeln *(Redundant Array of Inexpensive Disks)* eingesetzt wird, soll hier auch die *Serial Storage Architecture* (SSA) der Firma IBM erwähnt werden. Dieser Bus erlaubt bei einer Übertragungsrate von 160 bis 320 Mbit/s den Aufbau einer Vielzahl von Topologien, wie Punkt-zu-Punkt-Verbindungen, Ring, Strang-Stern und Stern. Insbesondere werden Sterne mit schnellen Schalteinheiten *(Switched Structures)* unterstützt. Als Doppelring ausgelegt, können über den SSA-Bus bis zu 640 Mbit/s (80 Mbyte/s) übertragen werden. SSA erlaubt den Anschluß von bis zu 27 Busteilnehmern. Dabei darf der Bus als Kupferkabel 40 m, als Glasfaserkabel 600 m lang sein.
- Der *Fibre Channel Arbitrated Loop* (FC-AL) wird häufig als serielle Ausprägung des SCSI-Busses angesehen, weil er u.a. denselben Satz an Befehlen (Kommandos) unterstützt und über Brücken mit dem SCSI-Bus gekoppelt werden kann.

Auch er wird bevorzugt zum Anschluß von RAID-Platten benutzt, aber auch von Bandgeräten und CD-ROMs. Als Topologien kommen Punkt-zu-Punkt-Verbindungen, Ring, Strang-Stern, Stern und *Switched Structures* in Frage. Fibre Channel bietet Übertragungsraten zwischen 100 und 800 Mbit/s, im Doppelring auch 1600 Mbit/s. Dabei muß berücksichtigt werden, daß eine 8-aus-10-Codierung durchgeführt wird, so daß die „Nutzrate" 20% geringer ist. Wie beim FireWire und USB wird auch die isochrone Übertragung[5] unterstützt. FC-AL erlaubt mit verdrillten Kupferkabeln *(Twisted Pair)* oder Koaxial-Kabel Buslängen bis 30 m, mit Glasfaserkabeln bis zu 10 km. An den Bus können bis zu 127 Geräte angeschlossen werden.

- Die *Infiniband*-Architektur ist eine Vernetzungsmöglichkeit für den Einsatz im Bereich der Hochleistungs-Server, die dort den völlig ‚überforderten' PCI-Bus ablösen soll. Sie ist als Kompromiß aus zwei konkurrierenden Bustechniken hervorgegangen: dem Future-I/O einer Firmengruppe um IBM und dem NGIO *(Next-Generation I/O)* von Intel. Wie der 3GIO-Ansatz ist auch das Infiniband eine ‚geschaltete' Verbindung *(Switched Architecture)*: Der *Infiniband Switch* ist Zentrum einer Anzahl von Punkt-zu-Punkt-Verbindungen mit einer Bandbreite von 2,5 Gbit/s. Zu jedem Zeitpunkt können mehrere unabhängige Verbindungen geschaltet werden. Dies erhöht einerseits die Zuverlässigkeit der Kopplung, da beim Ausfall einer Verbindung eine andere geschaltet werden kann. Andererseits besitzt die Architektur eine gute Skalierbarkeit; denn zur Leistungssteigerung können auf einfache Weise weitere Switches eingesetzt werden. Der *Infiniband Switch* koppelt mehrere Server miteinander. Er ermöglicht durch den Einsatz eines *Fibre-Channel Switches* aber auch den Anschluß von leistungsfähigen Speichersystemen *(Storage Systems)*. Weiterhin können besondere *Switches* zur Realisierung vielfacher Netzverbindungen *(Ethernet Switches)* eingesetzt werden.
- Das *Scalable Coherent Interface* (SCI, IEEE-Standard 1596-1992) dient hauptsächlich zum Aufbau von Hochleistungs-Parallelrechnern. Es kann aber auch in Ein-Rechnersystemen eingesetzt werden. Die Topologie ist in weiten Grenzen frei wählbar – von einem Ring bis zu Systemen mit Kreuzschienenschaltern *(Crossbar Switches)*. Auch die Anzahl der Knoten kann stark variieren. Zur Vereinfachung der elektrischen Probleme sind die Leitungen zwischen den Knoten als unidirektionale Punkt-zu-Punkt-Verbindungen realisiert. Es werden zwei Übertragungsraten spezifiziert: 1 Gbit/s können über Koaxial-Kabel mit einer Länge von einigen Dutzend Metern oder Glasfaserkabeln bis zu 10 km erreicht werden. 1 Gbyte/s sind über 18 differentielle Signalpaare über wenige Meter möglich. Das SCI-Protokoll unterstützt sowohl die Datenübertragung in Form von Nachrichten wie den direkten Zugriff auf gemeinsame Daten im Speicher *(Shared Memory)*. Die Busadressen sind 64 bit lang – 16 bit für die Knotenadresse und 48 bit für die Speicheradresse.

[5] „Isochron" bedeutet sinngemäß: zum gleichen Zeitpunkt in einem immer wiederkehrenden Zeitrahmen auftretend.