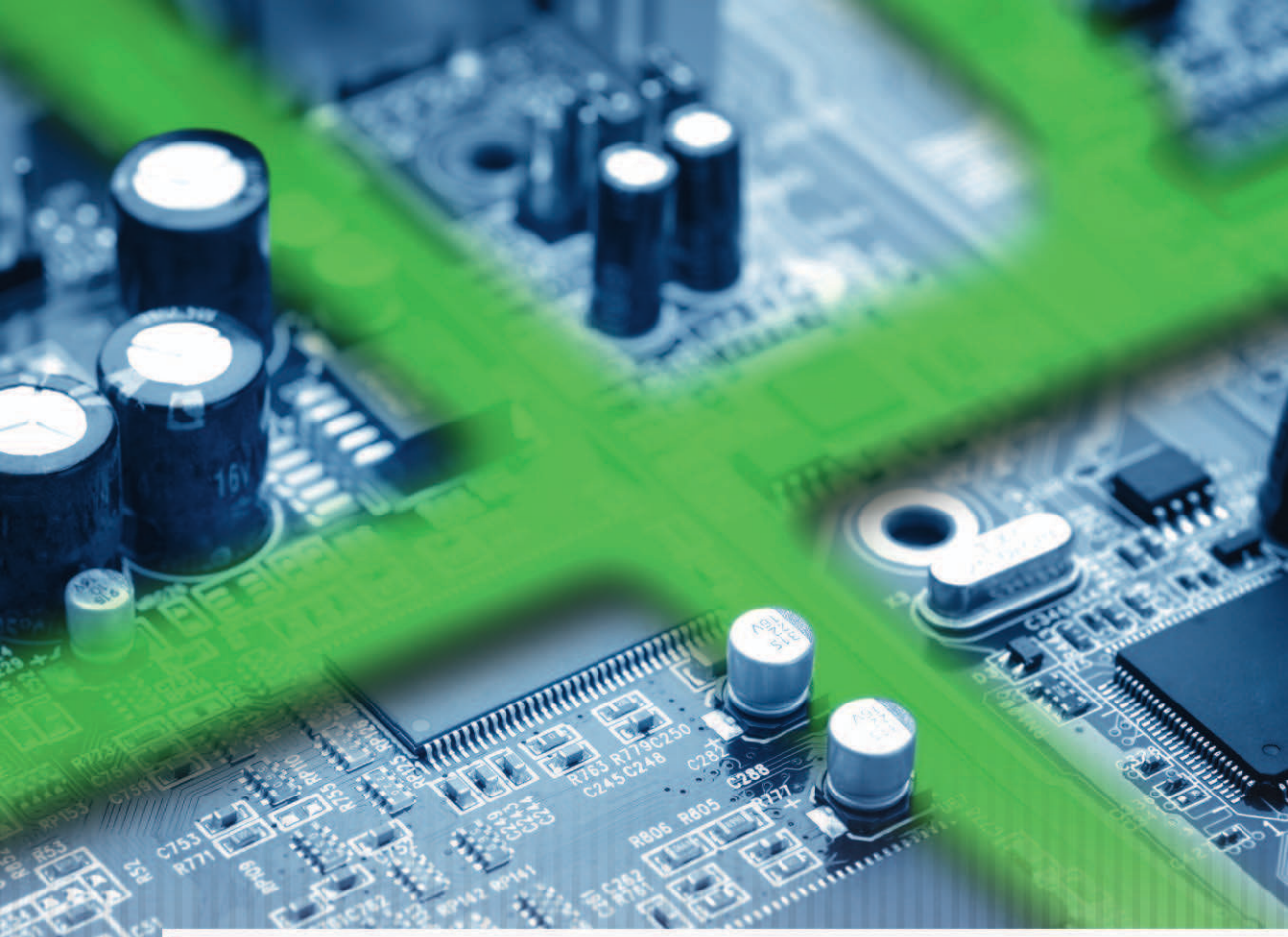

Rechnerarchitektur

Von der digitalen Logik zum Parallelrechner

6., aktualisierte Auflage

Andrew S. Tanenbaum
Todd Austin

Pearson

EXTRAS ONLINE

Inhaltsverzeichnis

lässt sich wie folgt auf einen Nenner bringen: Im Innersten stellt ein PC eine Ansammlung von CPU, Speicher und E/A-Controllerchips dar, die untereinander verbunden werden müssen. PCI Express stellt einen universellen Switch bereit, um Chips mithilfe von seriellen Verbindungen zusammenzuschalten. ▶Abbildung 3.51 zeigt eine typische Konfiguration.

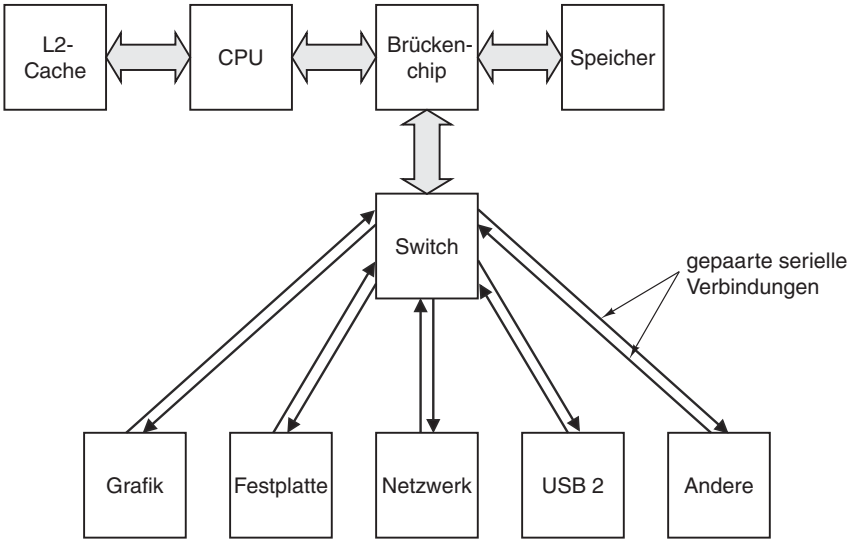

Abbildung 3.51: Ein typisches PCI-Express-System

Wie Abbildung 3.51 zeigt, sind die CPU, der Speicher und der Cache in der herkömmlichen Form an den Brückenchip angeschlossen. Das Neue hier ist ein Switch, der mit der Brücke verbunden (und möglicherweise Teil der Brücke selbst) ist. Von jedem E/A-Chip führt eine dedizierte Punkt-zu-Punkt-Verbindung zum Switch. Jede Verbindung umfasst ein Paar unidirektionaler Kanäle – einer zum Switch hin und einer von ihm weg. Ein Kanal besteht aus zwei Leitungen, einer Signalleitung und einer Masseleitung, um hohe Störsicherheit während der Hochgeschwindigkeitsübertragung zu gewährleisten. Diese Architektur ersetzt die alte durch ein wesentlich einheitlicheres Modell, das alle Geräte gleich behandelt.

Die PCI-Express-Architektur unterscheidet sich von der alten PCI-Bus-Architektur in drei wesentlichen Punkten. Zwei haben wir bereits kennengelernt: ein zentraler Switch gegenüber einem Mehrpunktbus und die Verwendung von schmalen Punkt-zu-Punkt-Verbindungen gegenüber einem breiten Parallelbus. Der dritte Punkt ist etwas subtiler. Konzeptionell löst beim PCI-Bus ein Busmaster einen Befehl an einen Slave aus, um ein Wort oder einen Block von Wörtern zu lesen. Beim PCI-Express-Modell sendet ein Gerät ein Datenpaket an ein anderes Gerät. Das Konzept des **Pakets**, das aus Header und Nutzdaten besteht, stammt aus der Netzwerkwelt. Der **Header** enthält Steuerinformationen, die die vielen Steuerleitungen des PCI-Busses überflüssig machen. Die **Nutzdaten** (Payload) enthalten die zu übertragenden Daten. Letztlich ist ein PC mit PCI Express ein Paketvermittlungsnetz (Packet Switching Network) en miniature.

Der Bruch mit der Vergangenheit dokumentiert sich neben diesen drei Hauptpunkten auch noch in mehreren kleineren Unterschieden. So sind viertens die Pakete mit einem Fehlerkorrekturcode versehen, was eine höhere Zuverlässigkeit als beim PCI-Bus bringt. Fünftens ist die Verbindung zwischen einem Chip und dem Switch mit 50 cm länger als bisher, um eine Systemaufteilung zu ermöglichen. Sechstens ist das System erweiterbar, weil ein Gerät durchaus auch ein anderer Switch sein kann und sich somit ein Baum von Switches bilden lässt. Siebentens sind Geräte Hot-Plugging-fähig, d.h., sie lassen sich im laufenden Betrieb des Systems ein- und ausbauen. Und da schließlich die Größe der seriellen Steckverbinder nur einen Bruchteil der alten PCI-Steckverbinder ausmacht, kann man Geräte und Computer wesentlich kleiner gestalten. Insgesamt gesehen ist PCI Express eine radikale Abkehr vom PCI-Bus.

PCI-Express ist ein Punkt-zu-Punkt-Netzwerk mit seriellen Leitungen. Die Geräte sind **Hot-Plugging**-fähig. Sie lassen sich bei laufendem System ein- und ausbauen.

Der PCI-Express-Protokollstapel

Entsprechend dem Modell eines paketvermittelten Netzwerks baut das PCI-Express-System auf einem geschichteten Protokollstapel auf. Ein **Protokoll** ist eine Menge von Regeln, die die Konversation zwischen zwei Teilnehmern bestimmen. Ein Protokollstapel umfasst eine Hierarchie von Protokollen, die sich mit unterschiedlichen Fragen auf unterschiedlichen Ebenen befassen. Nehmen Sie als Beispiel einen Geschäftsbrief. Er hält sich an bestimmte Konventionen hinsichtlich Platzierung und Inhalt von Briefkopf, Empfängeradresse, Datum, Anrede, eigentlichem Text, Unterschrift usw. Das kann man sich als Briefprotokoll vorstellen. Daneben gibt es eine Menge von Konventionen zum Briefumschlag – zum Beispiel Größe, Lage und Format des Absenders, Lage und Format der Anschrift, Anordnung der Briefmarke usw. Diese beiden Ebenen und ihre Protokolle sind unabhängig. Zum Beispiel ist es möglich, den Brief vollständig neu zu formatieren, aber denselben Umschlag zu verwenden und umgekehrt. Geschichtete Protokolle fördern ein modulares flexibles Design und haben sich seit Jahrzehnten in der Welt der Netzwerke weithin etabliert. Neu ist hier, dass man sie in die „Bus"-Hardware integriert.

▶Abbildung 3.52a zeigt den Protokollstapel des PCI-Express-Systems.

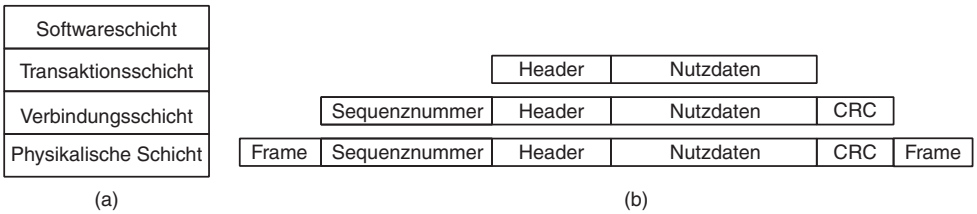

Abbildung 3.52: (a) Der PCI-Express-Protokollstapel (b) Format eines Pakets

Wir untersuchen nun die Schichten von unten nach oben. Die unterste Schicht ist die **physikalische Schicht** (Physical Layer). Hier findet die Übertragung der Bits von einem Sender zu einem Empfänger über eine Punkt-zu-Punkt-Verbindung statt. Jede Punkt-zu-Punkt-Verbindung besteht aus einem oder mehreren Paaren von Simplexverbindungen (d.h. unidirektionalen Verbindungen). Im einfachsten Fall gibt es in jeder Richtung ein Paar, jedoch sind auch 2, 4, 8, 16 oder 32 Paare zulässig. Eine derartige Verbindung wird als **Lane** (Spur, Strang) bezeichnet. Die Anzahl der Lanes muss in jeder Richtung gleich sein. Produkte der ersten Generation müssen auf jedem Weg eine Datenrate von mindestens 2,5 Gbit/s unterstützen. Es ist aber recht bald damit zu rechnen, dass man auf jedem Weg zu einer Geschwindigkeit von 10 Gbit/s übergeht.

Im Unterschied zu den ISA/EISA/PCI-Bussen verfügt PCI Express nicht über einen Mastertakt. Geräte können aufs Geratewohl mit einer Übertragung beginnen, sobald sie zu sendende Daten haben. Diese Freiheit macht das System schneller, führt aber auch zu Problemen. Nehmen wir an, dass ein 1-Bit mit +3 V und ein 0-Bit mit 0 V codiert ist. Wenn die ersten Bytes sämtlich 0 sind, wie erkennt dann der Empfänger, dass Daten übertragen werden? Immerhin sieht eine Folge von 0-Bits genau wie eine Verbindung im Leerlauf aus. Als Lösung hat man die sogenannte **8b/10b-Codierung** gewählt. Dabei werden 10 Bits verwendet, um 1 Byte der eigentlichen Daten in einem 10-Bit-Symbol zu codieren. Von den 1024 möglichen 10-Bit-Symbolen hat man solche mit genügend Taktübergängen ausgewählt, um Sender und Empfänger auf den Bitgrenzen selbst ohne Mastertakt synchronisieren zu können. Wegen der 8b/10b-Codierung kann eine Verbindung mit einer Bruttokapazität von 2,5 Gbit/s nur 2 Gbit/s Benutzerdaten (Nettodaten) transportieren.

Während die physikalische Schicht mit der Bitübertragung zu tun hat, beschäftigt sich die **Verbindungsschicht** (Link Layer) mit der Paketübertragung. Sie übernimmt den Header und die Nutzdaten von der Transaktionsschicht, fügt diesen Daten eine Sequenznummer hinzu, berechnet einen Fehlerkorrekturcode und hängt diesen an das Paket an. Der als **CRC** (**Cyclic Redundancy Code**) bezeichnete Fehlerkorrekturcode wird nach einem bestimmten Algorithmus aus den Header- und Nutzdaten berechnet. Erhält der Empfänger ein Paket, führt er die gleiche Berechnung auf den Header- und Nutzdaten durch. Stimmt das Ergebnis mit dem im Paket angefügten CRC-Wert überein, dann sendet der Empfänger ein kurzes **Bestätigungspaket** (Acknowledgment Packet) zurück, das den korrekten Empfang quittiert. Bei abweichenden Ergebnissen fordert der Empfänger eine Neuübertragung an. Dieses Verfahren gewährleistet eine deutlich bessere Datenintegrität gegenüber PCI-Bussystemen, die keinerlei Vorkehrungen für Verifizierung und Neuübertragung der über den Bus gesendeten Daten kennen.

Um zu verhindern, dass ein schneller Sender einen langsamen Empfänger in Paketen erstickt, die dieser nicht behandeln kann, wird eine **Flusssteuerung** verwendet. Der verwendete Mechanismus stützt sich darauf, dass der Empfänger dem Sender eine bestimmte Anzahl von Credits gibt, die grundsätzlich der Größe des Puffers entsprechen, die er für eintreffende Pakete verfügbar hat. Wenn die Credits aufgebraucht sind, muss der Sender das Senden stoppen, bis er neue Credits erhält. Dieses in allen Netzwerken übliche Schema verhindert, dass Pakete aufgrund nicht übereinstimmender Geschwindigkeiten von Sender und Empfänger verloren gehen.

Die **Transaktionsschicht** behandelt Busaktionen. Um ein Wort aus dem Speicher zu lesen, sind zwei Transaktionen erforderlich: Die CPU oder ein DMA-Kanal leitet eine Transaktion ein, um bestimmte Daten anzufordern, das Target initiiert eine zweite Transaktion, um die Daten bereitzustellen. Allerdings hat die Transaktionsschicht noch mehr zu tun, als reine Lese- und Schreibvorgänge abzuwickeln. Sie bereitet die von der Verbindungsschicht angebotene Rohpaketübertragung auf. Dazu kann sie zunächst jede Lane in bis zu acht **virtuelle Verbindungen** unterteilen, die jeweils eine andere Klasse von Datenverkehr behandeln. Die Transaktionsschicht markiert Pakete entsprechend ihrer Verkehrsklasse, wozu Attribute wie hohe Priorität, niedrige Priorität, „schnüffelt“ nicht, kann unabhängig von der Reihenfolge bereitgestellt werden usw. gehören. Anhand dieser Tags kann der Switch entscheiden, welches Paket als Nächstes an die Reihe kommt.

Jede Transaktion verwendet einen von vier Adressräumen:

1. Speicherraum (für normale Lese- und Schreiboperationen)
2. E/A-Raum (für die Adressierung von Geräteregistern)
3. Konfigurationsraum (für Systeminitialisierung etc.)
4. Nachrichtenraum (für Signalisierung, Interrupts etc.)

Die Speicher- und E/A-Räume sind mit aktuellen Systemen vergleichbar. Mit dem Konfigurationsraum lassen sich Funktionen wie Plug-and-Play implementieren. Der Nachrichtenraum übernimmt die Rolle vieler vorhandener Steuersignale. Etwas in der Art wie dieser Raum ist unbedingt notwendig, da in PCI Express keine Steuerleitungen des PCI-Busses existieren.

Die **Softwareschicht** stellt die Schnittstelle des PCI-Express-Systems zum Betriebssystem her. Sie kann den PCI-Bus emulieren, sodass vorhandene Betriebssysteme unverändert auf PCI-Express-Systemen laufen können. Natürlich nutzt ein derartiger Betrieb nicht die volle Leistung von PCI Express, aber die Abwärtskompatibilität ist ein notwendiges Übel, auf das man erst verzichten kann, wenn die Betriebssysteme vollständig auf PCI Express ausgerichtet sind. Die Erfahrung zeigt, dass das noch eine ganze Weile dauern kann.

Abbildung 3.52 b zeigt den Informationsfluss. Wird ein Befehl an die Softwareschicht erteilt, dann übergibt sie ihn an die Transaktionsschicht, die ihn als Header und Nutzdaten formuliert. Diese beiden Teile werden an die Verbindungsschicht übergeben, die eine Sequenznummer vorn und eine CRC-Prüfsumme hinten anfügt. Dieses vergrößerte Paket geht dann zur physikalischen Schicht, die Rahmeninformationen an jedes Paket anfügt, um das physische Paket zu bilden, das dann tatsächlich gesendet wird. Am empfangenden Ende findet der umgekehrte Prozess statt, wobei die voran- und nachgestellten Verbindungsinformationen entfernt werden und das Ergebnis an die Transaktionsschicht übergeben wird.

Das Konzept, dass jede Schicht zusätzliche Informationen an die Daten anfügt, wenn sie den Protokollstapel von oben nach unten durchlaufen, wird seit Jahrzehnten mit Erfolg in der Netzwerkwelt angewandt. Während aber der Code in der Netzwerkwelt nahezu immer durch Software des Betriebssystems realisiert ist, gehört er bei PCI Express zur Gerätehardware.

PCI Express ist ein komplexes Thema. Weitere Informationen finden Sie in [Mayhew und Krishnan, 2003] und [Solari und Congdon, 2005]. Außerdem schreitet die Entwicklung auch bei PCI Express voran. So wurde 2007 PCIe 2.0 veröffentlicht. Diese Version unterstützt 500 MB/s je Lane (die bis zu 32 Leitungen umfasst), was eine Bandbreite von insgesamt 16 GB/s ergibt. Der 2011 veröffentlichte Standard PCIe 3.0 hat die Codierung von 8b/10b auf 128b/130b geändert und kann 8 Milliarden Transaktionen/s bewältigen, das Doppelte von PCIe 2.0.

3.6.3 USB (Universal Serial Bus)

Der PCI-Bus und PCI Express eignen sich gut zum Anschließen von Hochgeschwindigkeitsperipherie an einen Computer, sind aber für langsame E/A-Geräte wie Tastaturen und Mäuse viel zu teuer. Im Verlauf der Geschichte wurde jedes E/A-Standardgerät auf besondere Weise an den Computer angeschlossen, wofür freie ISA- und PCI-Steckplätze genutzt wurden. Leider war dieses Verfahren von Anfang an mit Problemen behaftet.

Beispielsweise wird ein neues E/A-Gerät meistens mit einer eigenen ISA- oder PCI-Karte geliefert. Der Benutzer muss manchmal die Schalter und Steckbrücken (Jumper) auf der Karte selbst einstellen und gewährleisten, dass die Einstellungen nicht zu Konflikten mit anderen Karten führen. Dann muss der Benutzer das Gehäuse öffnen, die Karte vorsichtig einstecken, das Gehäuse schließen und den Rechner neu starten. Für viele Benutzer ist dieser Vorgang schwierig und birgt viele Fehlerquellen. Außerdem ist die Anzahl der ISA- und PCI-Steckplätze recht begrenzt (normalerweise auf zwei oder drei). Plug-and-Play-Karten ersparen dem Benutzer zwar die Einstellung der Jumper, zur Installation muss er aber trotzdem den Computer öffnen und die Karte einstecken. An der Anzahl der Bussteckplätze hat sich nichts geändert.

Um dieses Problem anzugehen, haben sich 1993 die Vertreter von sieben Unternehmen (Compaq, DEC, IBM, Intel, Microsoft, NEC und Northern Telecom) an einen Tisch gesetzt, um ein besseres Designkonzept für den Anschluss langsamer E/A-Geräte an einen Computer auf den Weg zu bringen. Seither sind Hunderte weiterer Unternehmen der Gruppe beigetreten. Der daraus hervorgegangene Standard heißt **USB** (**Universal Serial Bus**) und wird inzwischen durchgängig in Personalcomputern implementiert. Detaillierte Informationen finden Sie in [Anderson, 1997] und [Tan, 1997].

Mit dem USB-Projekt verfolgten die Gründungsmitglieder der USB-Arbeitsgruppe unter anderem folgende Ziele:

1. Benutzer sollen keine Schalter und Jumper auf Platinen oder Geräten einstellen müssen.
2. Benutzer sollen das Gehäuse nicht zur Installation neuer E/A-Geräte öffnen müssen.
3. Es soll nur eine Kabelart für den Anschluss aller Geräte verwendet werden.
4. Die Stromversorgung der E/A-Geräte soll über das Kabel erfolgen.
5. An einen einzigen Computer sollen sich bis zu 127 Geräte anschließen lassen.
6. Das System soll Echtzeitgeräte (z.B. Sound, Telefon) unterstützen.
7. Geräte sollen bei laufendem Computer installierbar sein.
8. Nach der Installation eines neuen Geräts soll kein Neustart des Computers erforderlich sein.
9. Der neue Bus und seine E/A-Geräte sollen preisgünstig gefertigt werden können.

USB ist ein Bus zur Anbindung langsamer E/A-Geräte, der eine baumartige Verbindungsstruktur realisiert, bei der jedes Gerät nur mit dem Root-Hub kommuniziert. **USB 3.0** bietet eine maximale Bandbreite von 5 Gbits/s.

Der USB erfüllt alle diese Ziele. Er wurde für langsame Geräte wie Tastaturen, Mäuse, Standbildkameras, Büroscanner, digitale Telefone usw. ausgelegt. Version 1.0 hatte eine Bandbreite von 1,5 Mbit/s, die für Tastaturen und Mäuse ausreicht. Version 1.1 läuft bei 12 Mbit/s, was für Drucker, Digitalkameras und viele andere Geräte genügt. Version 2.0 unterstützt Geräte mit bis zu 480 Mbit/s, sodass sich auch externe Festplatten, HD-Webcams und Netzwerkadapter problemlos betreiben lassen. Die zuletzt definierte USB-Version 3.0 erhöht die Geschwindigkeit auf bis zu 5 Gbit/s. Es bleibt abzuwarten, welche neuen und bandbreitenhungrigen Anwendungen von dieser Schnittstelle mit ultrahoher Bandbreite beflügelt werden.

Ein USB-System besteht aus einem **Root-Hub**, der in den Hauptbus (Abbildung 3.47) eingesteckt wird. Dieser Hub hat Buchsen für Kabel, über die E/A-Geräte oder Erweite-

rungshubs angeschlossen werden können, sodass weitere Buchsen bereitstehen. Die Topologie eines USB-Systems ist also ein Baum mit der Wurzel beim Root-Hub im Innern des Computers. Die Steckverbinder der Kabel unterscheiden sich zwischen Hub-Seite und Geräteseite, um zu verhindern, dass man versehentlich zwei Hubs zusammensteckt.

Das Kabel besteht aus vier Leitungen – zwei für Daten, einer für die Betriebsspannung (+5 V) und einer für Masse. Das Signalisierungssystem überträgt eine 0 als Spannungsübergang und eine 1 als Fehlen eines Spannungsübergangs, sodass lange Folgen von Nullen einen regelmäßigen Impulsstrom erzeugen.

Der Root-Hub erkennt, wenn ein neues E/A-Gerät eingesteckt wird, und unterbricht das Betriebssystem. Das Betriebssystem fragt das Gerät danach ab, um welches es sich handelt, und wie viel USB-Bandbreite es benötigt. Stellt das Betriebssystem fest, dass ausreichend Bandbreite für das Gerät vorhanden ist, weist es ihm eine eindeutige Adresse (1–127) zu. Dann lädt es diese Adresse und andere Informationen in Konfigurationsregister innerhalb des Geräts. Auf diese Weise können neue Geräte problemlos hinzugefügt werden, ohne dass der Benutzer etwas konfigurieren oder installieren muss. Nicht initialisierte Karten erhalten zunächst die Adresse 0, damit sie sich ansprechen lassen. Zur Vereinfachung der Verkabelung sind in vielen USB-Geräten Hubs zur Aufnahme zusätzlicher USB-Geräte eingebaut. Ein Monitor kann beispielsweise zwei Hub-Buchsen bekommen, um den linken und rechten Lautsprecher aufzunehmen.

Logisch kann man das USB-System als Menge von Datenströmen vom Root-Hub zu den E/A-Geräten betrachten. Jedes Gerät kann seine Bit-Pipe in maximal 16 Teil-Pipes für unterschiedliche Datenarten (z.B. Audio und Video) aufteilen. Innerhalb jeder Pipe oder jeder Teil-Pipe fließen Daten vom Root-Hub zum Gerät oder umgekehrt. Zwischen zwei E/A-Geräten gibt es keinen Datenverkehr.

Für die Synchronisation sendet der Root-Hub genau alle 1,00 ± 0,05 ms einen neuen Frame an alle Geräte. Ein Frame ist mit einer Bit-Pipe verbunden und besteht aus Paketen. Das erste Paket kommt vom Root-Hub des Geräts. Die folgenden Pakete im Frame können ebenfalls in diese Richtung oder vom Gerät zurück zum Root-Hub laufen. ▶Abbildung 3.53 zeigt eine Folge von vier Frames.

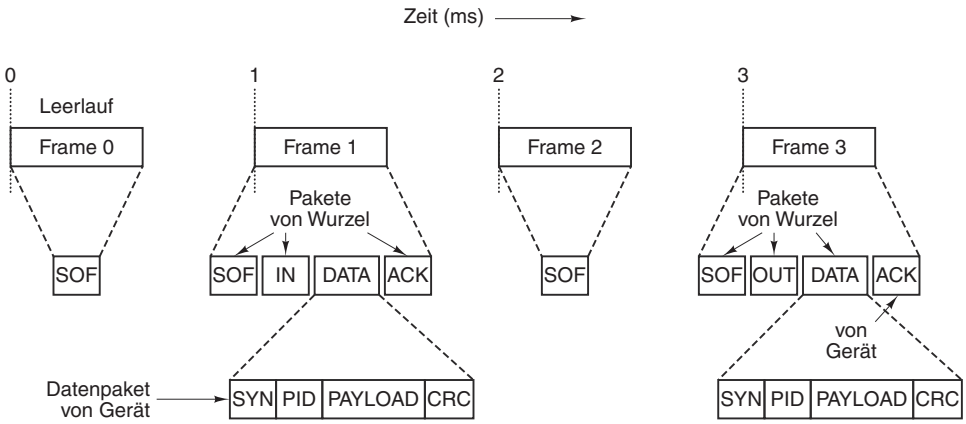

Abbildung 3.53: Der USB-Root-Hub sendet alle 1,00 ms Frames aus

In Abbildung 3.53 haben die Frames 0 und 2 nichts zu tun, sodass nur ein SOF-Paket (Start Of Frame) erforderlich ist. Dieses Paket wird immer als Rundruf an alle Geräte gesendet. Frame 1 ist eine Abfrage, z.B. eine Anforderung an einen Scanner, die im abgetasteten Bild gefundenen Bits zurückzugeben. Frame 3 besteht aus den Bereitstellungsdaten für ein bestimmtes Gerät, wie zum Beispiel einen Drucker.

USB unterstützt vier Arten von Frames: Control, Isochronous, Bulk und Interrupt. Control-Frames werden benutzt, um Geräte zu konfigurieren, Befehle an sie zu erteilen und ihren Status abzufragen. Isochronous-Frames sind für Echtzeitgeräte wie Mikrofone, Lautsprecher und Telefone vorgesehen, die Daten in genauen Zeitintervallen senden oder annehmen müssen,. Sie haben eine genau vorhersagbare Verzögerung, bieten im Fall von Fehlern aber keine Möglichkeit der erneuten Übertragung. Bulk-Frames werden für umfangreiche Übertragungen zu oder von Geräten ohne Echtzeitanforderungen, z.B. Drucker, benutzt. Schließlich sind Interrupt-Frames nötig, weil der USB keine Interrupts unterstützt. Anstatt die Tastatur bei jedem Tastenanschlag einen Interrupt auslösen zu lassen, kann das Betriebssystem die Tastatur alle 50 ms abfragen, um eventuell ausstehende Tastenanschläge zu sammeln.

Ein Frame besteht aus einem oder mehreren Paketen, die möglicherweise in beide Richtungen gesendet werden. Es gibt vier Paketarten: Token, Data, Handshake und Special. Token-Pakete fließen vom Root-Hub zu einem Gerät und dienen der Systemsteuerung. Die Pakete SOF, IN und OUT in Abbildung 3.53 sind Token-Pakete. Das Paket SOF (Start Of Frame) ist das erste in jedem Frame und kennzeichnet den Frame-Beginn. Wenn es nichts zu tun gibt, ist SOF das einzige Paket im Frame. Das Token-Paket IN ist eine Abfrage, die das Gerät auffordert, bestimmte Daten zurückzugeben. Die Felder im IN-Paket geben Auskunft darüber, welche Bit-Pipe abgefragt wird, sodass das Gerät weiß, welche Daten zurückzugeben sind (wenn es mehrere Datenströme hat). Das Token-Paket OUT kündigt an, dass Daten für das Gerät folgen. Ein vierter Token-Pakettyp SETUP (in der Abbildung nicht gezeigt) wird für die Konfiguration verwendet.

Neben dem Token-Paket gibt es noch drei weitere: DATA (für die Übertragung von 64 Byte Informationen in beide Richtungen), Handshake und Special. Das Format eines Datenpakets ist aus Abbildung 3.53 ersichtlich. Es besteht aus einem 8-Bit-Synchronisierungsfeld, einem 8-Bit-Pakettyp (PID), den Nutzdaten (Payload) und einem 16-Bit-**CRC** (**Cyclic Redundancy Code**) zur Fehlererkennung. Es sind drei Arten von Handshake-Paketen definiert: ACK (das vorherige Datenpaket wurde korrekt empfangen), NAK (ein CRC-Fehler wurde erkannt) und STALL (bitte warten, bin momentan beschäftigt).

Sehen Sie sich Abbildung 3.53 noch einmal an. Alle 1,00 ms muss ein Frame vom Root-Hub gesendet werden, auch wenn nichts zu tun ist. Die Frames 0 und 2 bestehen nur aus einem SOF-Paket, das diesen Leerlauf anzeigt. Frame 1 ist eine Abfrage und beginnt deshalb mit SOF- und IN-Paketen vom Computer zum E/A-Gerät, gefolgt von einem DATA-Paket vom Gerät zum Computer. Das ACK-Paket teilt dem Gerät mit, welche Daten korrekt empfangen wurden. Bei einem Fehler würde ein NAK an das Gerät zurückgeschickt und das Paket bei Bulk-Daten (nicht aber bei isochronen Daten) erneut übertragen werden. Der Aufbau von Frame 3 entspricht dem von Frame 1, außer dass die Daten vom Computer zum Gerät fließen.

Nachdem der USB-Standard im Jahre 1998 verabschiedet war, gingen die Designer von USB gleich daran, an einer neuen Hochgeschwindigkeitsversion von USB namens **USB 2.0** zu arbeiten. Dieser Standard ist dem älteren USB 1.1 ähnlich und dazu abwärtskompatibel, außer dass er den bisher vorhandenen zwei Geschwindigkeiten mit 480 Mbit/s noch eine dritte Geschwindigkeit hinzufügt. Des Weiteren gibt es einige kleinere Unterschiede,

wie zum Beispiel bei der Schnittstelle zwischen dem Root-Hub und dem Controller. Mit USB 1.1 waren zwei Schnittstellen verfügbar. Die erste namens **UHCI** (**Universal Host Controller Interface**) stammt von Intel und verlagert einen Großteil der Arbeitslast auf die Softwareentwickler (sprich: Microsoft). Die zweite Schnittstelle mit der Bezeichnung **OHCI** (**Open Host Controller Interface**) kommt aus dem Hause Microsoft und bürdet die meiste Arbeit den Hardwareentwicklern (sprich: Intel) auf. In USB 2.0 haben sich nun alle Partner auf eine einzige neue Schnittstelle namens **EHCI** (**Enhanced Host Controller Interface**) geeinigt.

USB ist nun mit einer Arbeitsgeschwindigkeit von 480 Mbit/s ein klarer Konkurrent zum seriellen IEEE-1394-Bus, der unter dem Namen FireWire bekannter ist und bei 400 Mbit/s läuft. Da praktisch jeder neue Intel-basierte PC mit USB 2.0 oder USB 3.0 (siehe unten) ausgeliefert wird, dürfte 1394 in absehbarer Zeit verschwinden. Das hängt nicht so sehr mit einer Überalterung, sondern eher mit Revierkämpfen zusammen. USB ist ein Produkt der Computerindustrie, während 1394 aus dem Bereich der Konsumgüterelektronik kommt. Beide Lager sind natürlich daran interessiert, dass ihre Kabel verwendet werden, wenn es darum geht, Kameras an Computer anzuschließen. Diesmal sieht es so aus, als ob das Lager der Computerindustrie das Rennen gemacht hat.

Acht Jahre nach Einführung von USB 2.0 wurde der Schnittstellenstandard **USB 3.0** angekündigt. USB 3.0 unterstützt eine kolossale Bandbreite von 5 Gbit/s über das Kabel, wobei aber die Modulation der Verbindung adaptiv ist und diese Topgeschwindigkeit wahrscheinlich nur mit professionellen Kabeln erreichbar sein dürfte. USB-3.0-Geräte sind mit den früheren USB-Geräten strukturell identisch und implementieren den USB-2.0-Standard vollständig. An einem USB-2.0-Anschluss arbeiten sie also ebenfalls korrekt.

3.7 Schnittstellen

Ein typisches kleines bis mittleres Computersystem besteht aus einem CPU-Chip, Speicherchips und einigen E/A-Controllern, die alle durch einen Bus miteinander verbunden sind. Manchmal sind alle diese Komponenten in einem Ein-Chip-System integriert, wie zum Beispiel beim OMAP4430 von Texas Instruments. Speicher, CPUs und Busse haben wir bereits mehr oder weniger ausführlich behandelt. Nun ist es an der Zeit, den letzten Teil des Puzzles – die E/A-Schnittstellen – zu betrachten. Über diese E/A-Ports kommuniziert ein Computer mit der Außenwelt.

3.7.1 E/A-Schnittstellen

E/A-Schnittstellen sind in Hülle und Fülle erhältlich und laufend kommen neue hinzu. Zu den Standardschnittstellen gehören UART, USART, CRT-Controller, Festplattencontroller und PIO. Ein **UART** (**Universal Asynchronous Receiver Transmitter**) ist eine E/A-Schnittstelle, die ein Byte vom Datenbus lesen und es bitweise über eine serielle Leitung an ein Terminal ausgeben oder Daten von einem Terminal entgegennehmen kann. UARTs unterstützen verschiedene Geschwindigkeiten von 50 bis 19.200 Bit/s, Zeichenbreiten von 5 bis 8 Bits, 1, 1,5 oder 2 Stoppbits und gerade, ungerade oder keine Parität. Alle diese Parameter lassen sich per Programm einstellen. Ein **USART** (**Universal Synchronous Asynchronous Receiver Transmitter**) beherrscht synchrone Übertragungen nach verschiedenen Protokollen sowie alle UART-Funktionen. Da Telefonmodems praktisch von der Bildfläche verschwunden sind, spielen UARTs nur noch eine untergeordnete Rolle. Wir untersuchen deshalb die parallele Schnittstelle als Beispiel für einen E/A-Chip.

PIO-Schnittstellen

▶Abbildung 3.54 zeigt eine typische **PIO**-Schnittstelle (**Parallel Input/Output**, nach dem Design des PIO-Schaltkreises 8255A von Intel). Sie besitzt mehrere E/A-Leitungen (24 im Beispiel gemäß Abbildung 3.54), die Geräte mit digitaler Logik ansteuern können, z.B. Tastaturen, Schalter, Lampen oder Drucker. Kurz gesagt kann das CPU-Programm auf jede Leitung eine 0 oder eine 1 schreiben oder den Eingangsstatus einer Leitung lesen, sodass sich dieser Schaltkreis sehr flexibel einsetzen lässt. Ein kleines, CPU-basiertes System mit einer PIO-Schnittstelle kann verschiedenste physische Geräte steuern, beispielsweise einen Roboter, einen Toaster oder ein Elektronenmikroskop. PIO-Schnittstellen sind auch typisch für eingebettete Systeme.

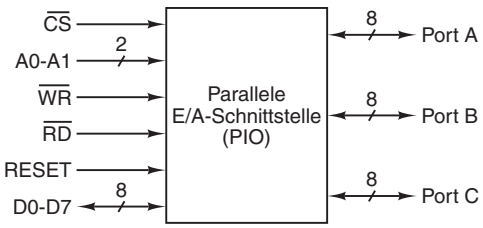

Abbildung 3.54: Ein PIO-Chip vom Typ 8255A

Über ein 3-Bit-Konfigurationsregister lässt sich festlegen, ob die drei unabhängigen 8-Bit-Ports als Eingänge (0) oder Ausgänge (1) für digitale Signale arbeiten sollen. Mit dem entsprechenden Wert im Konfigurationsregister ist jede Kombination von Eingängen und Ausgängen der drei Ports programmierbar. Jedem Port ist ein 8-Bit-Zwischenregister (Latch) zugeordnet. Um die Leitungen auf einem Ausgabeport zu setzen, schreibt die CPU eine 8-Bit-Zahl in das entsprechende Register. Dann erscheint die 8-Bit-Zahl auf den Ausgangsleitungen und bleibt dort, bis das Register erneut beschrieben wird. Um einen Port für die Eingabe zu verwenden, liest die CPU einfach das entsprechende Register.

Es ist auch möglich, komplexere PIO-Schnittstellen zu realisieren. So sieht eine gängige Betriebsart Handshaking mit externen Geräten vor. Um beispielsweise eine Ausgabe an ein nicht ständig zur Datenaufnahme bereites Gerät zu schicken, kann der PIO-Chip auf einen Ausgangsport Daten legen und warten, bis das Gerät mit einem Rückmeldeimpuls anzeigt, dass es die Daten akzeptiert hat und auf weitere Daten wartet. Die erforderliche Logik, um derartige Impulse zwischenzuspeichern und sie für die CPU bereitzustellen, umfasst ein Bereitschaftssignal und eine Warteschlange mit 8-Bit-Registern für jeden Ausgabeport.

Aus dem Funktionsdiagramm des PIO-Schaltkreises ist ersichtlich, dass er zusätzlich zu den 24 Pins für die drei Ports noch über acht Leitungen, die direkt mit dem Datenbus verbunden sind, eine Chipauswahlleitung, Lese- und Schreibleitungen, zwei Adressleitungen und eine Leitung zum Zurücksetzen des Chips verfügt. Die beiden Adressleitungen wählen eines der vier internen Register aus, die den Ports A, B, C und dem Statusregister entsprechen. Die Bits des Statusregisters legen die Ports für Eingabe und Ausgabe sowie andere Funktionen fest. Normalerweise sind die beiden Adressleitungen mit den niederwertigen Bits des Adressbus verbunden. Die Chipauswahlleitung ermöglicht es, die 24-Bit-PIO zu größeren PIO-Schnittstellen zu erweitern. Dazu werden mithilfe weiterer Adressleitungen die Chipauswahlleitungen der einzelnen PIO-Schaltkreise aktiviert.

3.7.2 Decodierung von Adressen

Bis jetzt haben wir absichtlich ungenaue Aussagen darüber gemacht, wie die Chipauswahl auf den bisher beschriebenen Speicher- und E/A-Chips aktiviert wird. Nun ist es an der Zeit, sich mit diesem Thema zu befassen. Betrachten wir einen einfachen eingebetteten 16-Bit-Computer, der aus einer CPU, einem 2K×8-Byte-EPROM für das Programm, einem 2K×8-Byte-RAM für die Daten und einer PIO-Schnittstelle besteht. Dieses kleine System lässt sich als Prototyp für die Steuerung eines billigen Spielzeugs oder eines einfachen Haushaltsgeräts einsetzen. Nach der Erprobung kann das EPROM durch ein ROM ersetzt werden.

Die PIO-Schnittstelle kann man auf zweierlei Weise auswählen: als echtes E/A-Gerät oder als Teil des Speichers. Soll der PIO-Schaltkreis als E/A-Gerät arbeiten, dann müssen wir ihn über eine besondere Busleitung auswählen, die anzeigt, dass ein E/A-Gerät und nicht der Speicher angesprochen wird. Der zweite Ansatz ist die sogenannte **Memory Mapped I/O**. Dabei werden den drei Ports und dem Steuerregister 4 Bytes des Speicheradressraums zugeordnet. Die Wahl einer dieser Varianten kann mehr oder weniger willkürlich geschehen. Wir entscheiden uns für Memory Mapped I/O, weil dieser Ansatz interessante Aspekte der E/A-Schnittstellen veranschaulicht.

Beim **Memory Mapped I/O** werden die Register des E/A-Gerätes in den normalen Adressraum des Prozessors abgebildet und können so geschrieben und gelesen werden.

Das EPROM und das RAM benötigen jeweils einen Adressraum von 2K. Der PIO-Schaltkreis belegt 4 Bytes. Da der Adressraum in unserem Beispiel 64K groß ist, müssen wir entscheiden, wo wir die drei Geräte einordnen wollen. ▶Abbildung 3.55 zeigt eine Möglichkeit. Das EPROM belegt die Adressen bis 2K, das RAM von 32K bis 34K und der PIO-Schaltkreis die höchsten 4 Bytes des Adressraums, 65532 bis 65535. Aus Sicht des Programmierers macht es keinen Unterschied, welche Adressen wir benutzen. Für die Schnittstellen ist das aber von Bedeutung. Hätten wir uns entschieden, den PIO-Schaltkreis über den E/A-Adressraum anzusprechen, würde er keine Speicheradressen belegen (dafür aber 4 E/A-Adressen).

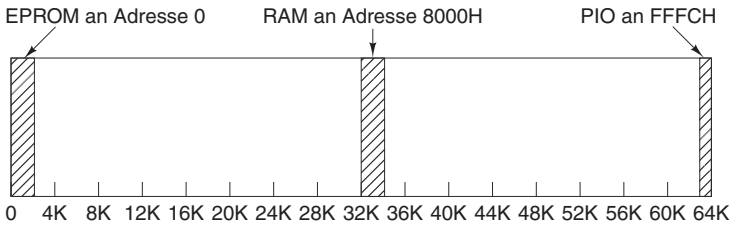

Abbildung 3.55: Speicherzuordnung für EPROM, RAM und PIO in einem 64-KB-Adressraum

Mit den Adresszuordnungen nach Abbildung 3.55 soll das EPROM durch eine 16-Bit-Speicheradresse im Format 0000 0xxx xxxx xxxx (binär) ausgewählt werden. Anders ausgedrückt: Jede Adresse, deren fünf höherwertige Bits Nullen sind, fällt in die unteren 2K des Speichers, also in den EPROM-Bereich. Somit lässt sich die EPROM-Chipauswahl mit einem 5-Bit-Komparator realisieren, dessen Vergleichseingänge fest auf 00000 gelegt sind.

Copyright