1 Einleitung

Analog zu *Moore's Law* [Moo65] als Postulat der stetig wachsenden Anzahl von Transistoren in einer integrierten Schaltung entstand der nach George Gilder [Gil00] als *Gilder's Law* bezeichnete Zusammenhang. *Gilder's Law* impliziert jedoch, dass die durch die Übertragungstechnik ermöglichten Datentransferraten über das gleiche Zeitintervall betrachtet in einem noch höheren Maße steigen als die Verarbeitungsleistung programmierbarer Mikroprozessoren. Abbilding 1.1 zeigt nach einer Darstellung von Bernard Aboba [Abo01] den qualitativen Vergleich der äquivalenten Verarbeitungsleistung eines *State-of-the-Art*-Mikroprozessors gegenüber der Datentransferrate der Netzwerktechnologie *Ethernet* über die Zeit. Offensichtlich besteht ein „Technologie-Gap" bei der Verarbeitung von Netzwerkanwendungen, bei denen der Datentransfer eine wesentliche Rolle spielt. Bereits aktuelle Netzwerktechnologien können nur durch spezialisierte Mikroprozessoren unterstützt werden [Lek03]. Auf Grundlage der rasanten Entwicklungen neuer Technologien kommt es zum Paradigmenwechsel in den Netzwerk- und Computersystemen [Mar02].

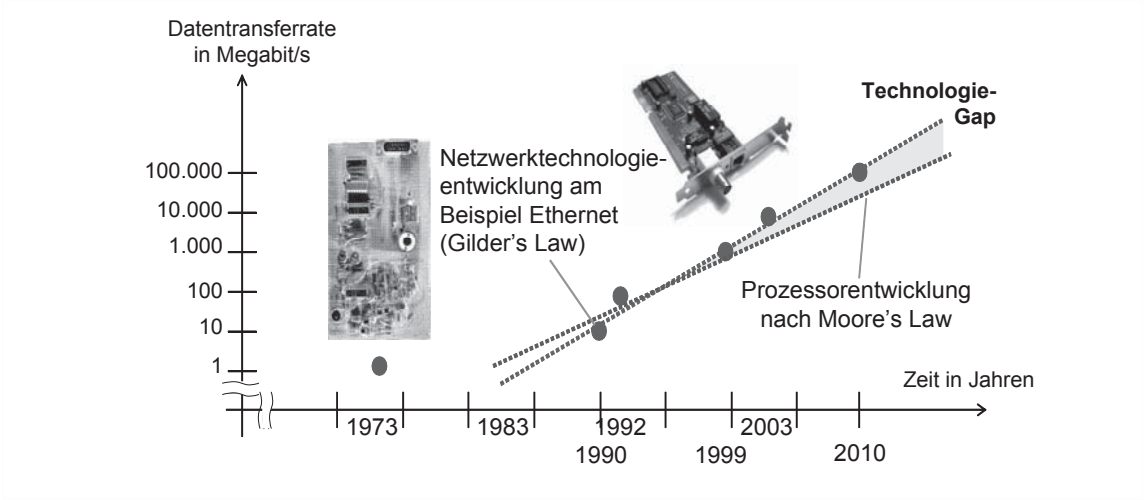


Abbildung 1.1: **Implikationen durch Gilder's Law [Eth11, Abo01, Bal05].**

Denkbare Architekturkonzepte zur Steigerung der Verarbeitungsleistung von Netzwerk- und I/O-Anwendungen betreffen nunmehr die zu den Computersystemen gehörigen Speichersysteme (inklusive der Organisation von Caches), die Prozessorpipeline und den Instruktionssatz der Mikroprozessoren sowie die Kopplung von Prozessoren und Netzwerk- bzw. I/O-Schnittstellen [PH09]. In der Vergangenheit konnte eine Leistungssteigerung durch den Einsatz von zusätzlichen, spezialisierten

integrierten Schaltungen (Chips) ermöglicht werden. Einerseits waren allerdings die Anforderungen an die Verarbeitungseinheiten im Vergleich zum aktuellen Stand der Technik moderat. Andererseits sind maßgebliche Änderungen mit solchen spezialisierten Chips nicht möglich, da eine fixe mikroelektronische Realisierung nicht wie eine Software anpassen lässt. Aufgrund der Vielzahl von Standards und häufig proprietären Erweiterungen ist neben Verarbeitungsleistung auch Flexibilität ein zentrales Kriterium für neuartige Architekturen.

Teilweise wird in Diskussionsforen angeregt, Speicher-, I/O- und Netzwerkschnittstellen derart zu verschmelzen, dass Cache- und Netzwerkstrukturen gänzlich transparent werden [Kat08]. Umsetzungen seitens der Halbleiterindustrie gibt es bereits in Form von leistungsstarken I/O- und Speicherschnittstellen [AT03, MSS09], welche gerade in Mehrprozessorsystemen hochratigen *Intra-System*-Datentransfer sicherstellen. Ein Beispiel ist die „Quickpath"-Schnittstelle der Firma Intel [MSS09]. Gleichwohl wird damit aber nicht die Unterstützung eines hochratigen *Inter-System*-Datentransfers im Sinne von aufkommenden Netzwerktechnologien geklärt. Der Großteil aktueller Schnittstellen ist nicht oder nur unzureichend programmierbar und nicht standardübergreifend einsetzbar. Angesichts der Forschungsarbeiten auf dem Gebiet der Netzwerkverarbeitung kann konstatiert werden, dass bereits eine Reihe von vielversprechenden Architekturkonzepten zusammengestellt worden ist [Cro03, Cro04, Cro05, Gil08]. Bei entsprechenden *Netzwerkprozessor*-Konzepten wird häufig ein Kompromiss zwischen Verarbeitungsleistung und Programmierbarkeit getroffen. Doch beziehen sich die meisten Ausführungen auf explizite Netzwerkgeräte und weniger auf Endgeräte, speziell *Server*.

Vor diesem Hintergrund erscheint es als ein geeigneter Kompromiss, spezialisierte, aber programmierbare Coprozessoren auf dem Mikroprozessorchip entsprechend Abbildung 1.2 zu integrieren. Diese Strategie geht einher mit den beachtlichen Möglichkeiten der Integration von unterschiedlichen Kompomenten auf einem Chip [Moo65]. Vorteile bestehen hier im Wesentlichen darin, dass redundante Schnittstellen vermieden und eine Leistungssteigerung ohne Verzicht auf Flexibilität erreicht werden kann. Gewöhnliche Datentransfers lassen sich direkt in der Speicherschnittstelle ausführen, bestimmte Ausnahmefälle mit Hilfe der zentralen Mikroprozessoren durch eine Software behandeln.

Ein anderer Anknüpfungspunkt ist die Frage nach alternativen Verarbeitungseinheiten, welche innerhalb der *Netzwerkprozessoren* eingesetzt werden können. Vielfach werden heute ohne technisch fundierte Begründungen RISC-basierte Architekturen [HPAD07, Appendix J] verwendet. Da sowohl bei *Intra-* als auch bei *Inter-System*-Datentransfer fast ausschließlich Paket-basierte Protokolle [Tan03, SW94] eingesetzt werden, kann eine offensichtliche Möglichkeit der Spezialisierung darin bestehen, auch die Mikroarchitektur der internen Verarbeitungseinheiten an die Datenpaketverarbeitung anzupassen.

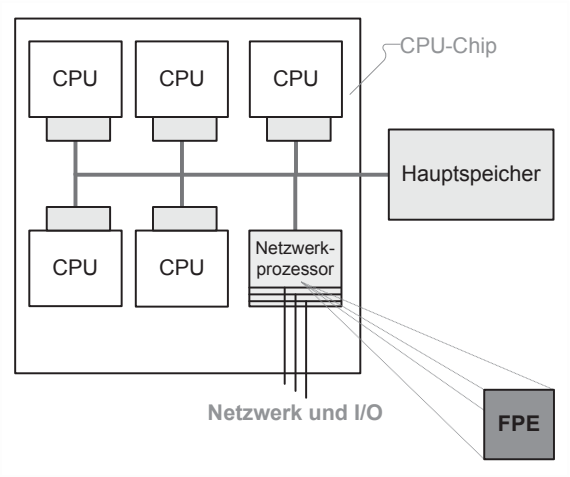


Abbildung 1.2: **Grundkonzept für integrierten Netzwerkprozessor.** *Der Block FPE (FSM-based Processing Engine) steht im Zentrum der Arbeit.*

1.1 Zielsetzung und Einordnung

Eine direkte Umsetzung der Datenpaketverarbeitung kann effizient mit Hilfe eines Zustandsautomaten (Englisch: Finite State Machine, kurz FSM) geschehen. Diese Arbeit befasst sich in diesem Zusammenhang mit einer neuartigen Prozessoreinheit zur hochratigen Datenpaketverarbeitung, der FPE (FSM-based Processing Engine). Deren Mikroarchitekturkonzept beruht auf der Kopplung einer programmierbaren FSM mit parallelen Datenpfaden und entstand aus Arbeiten von v. Lunteren [Lun06a, Lun06b] und einem Kooperationsprojekt mit dem Böblinger Entwicklungslabor der IBM Research & Development GmbH [IBM08] und später der Leibniz Universität Hannover. Der Ansatz ist insbesondere deshalb interessant, weil es sich um eine hybride Lösung mit potentiell neuen Anwendungen handelt.

Entgegen einer Vielzahl von FSM-basierten Vorschlägen aus der Literatur kann die FPE vereinfacht als frei programmierbare Verarbeitungseinheit verstanden werden [Sep10]. Die FSM ersetzt im Mikroarchitekturkonzept der FPE die konventionellerweise in einem Prozessor umgesetzte *Instruction-Fetch*-Stufe [HPAD07, Appendix A]. Aufgrund der, wie später gezeigt wird, generell kurzen Pipeline und sehr effizienten Umsetzung der FSM ergeben sich signifikante Vorteile bei der Abbildung von Sprüngen im Programmcode. Diese lassen sich insbesondere bei der Datenpaketverarbeitung nutzen. Damit ist eine Abgrenzung zum einen zu frei programmierbaren Prozessoren sowie zum anderen zu optimierten, dedizierten Einheiten möglich, wobei die FPE Vorteile der verschiedenen Ansätze miteinander zu verbinden scheint.

Jedoch gilt es, konkret zu werden und ein „Pro und Kontra“ auf quantitativ belegbare Aussagen zurückzuführen. In diesem Zusammenhang ergibt sich das erste Ziel mit der Ausarbeitung eines ausgewogenen Designs. Unter Berücksichtigung der Eckdaten eines potentiellen Chipdesigns und der Evaluation der Programmcode-

ausführung kann dann die Verarbeitungsleistung der Einheit abgeschätzt werden. Dazu sind geeignete *Benchmarks* heranzuziehen. Am Ende gilt es die Frage zu beantworten, inwieweit die FPE als Komponente in einem *Netzwerkprozessor* vorteilhaft einsetzbar ist und somit die Ergebnisse als Basis für weiterführende Arbeiten dienen können.

Der Schwerpunkt dieser Arbeit wird neben einer konsequenten Umsetzung der Grundidee und abschließenden Design-Evaluation auf den Entwurf und die Implementierung von Compiler-Werkzeugen. Dieses Vorgehen soll kurz motiviert werden: Die Programmierung und damit auch die Evaluation der Einheit stellt somit aufgrund der andersartigen Mikroarchitektur eine zunächst offene Fragestellung dar. Das Abbilden jeglicher Anwendungen auf die FPE erfordert die Konstruktion eines geeigneten Zustandsautomaten. Realistische Anwendungen lassen sich praktisch betrachtet nicht ohne eine geschlossene Programmierumgebung beherrschen. Da sich die Mikroarchitektur der FPE grundlegend von denen konventioneller Prozessoren unterscheidet, kann nicht auf verfügbare kommerzielle Werkzeuge zurückgegriffen werden. Daher soll zur effizienten Instruktionscodeerzeugung eine sogenannte *Toolchain* in Form von Compiler- und Simulator-Werkzeugen für die FPE vorgestellt werden. Diese erreicht eine starke Abstraktion der Mikroarchitektur, weist jedoch selbst eine nicht geringe Komplexität auf, die in der Arbeit angedeutet werden soll.

1.2 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich wie folgt. Kapitel 2 befasst sich einführend mit den Grundlagen der Datenpaketverarbeitung sowie exemplarisch mit einigen Standards und Trends der hochratigen Datenpaketverarbeitung. Schließlich werden Verarbeitungsmodelle und Anforderungen abgeleitet, welche sich in Bezug zu den steigenden Datenraten setzen lassen. Kapitel 2 fasst zudem Konzepte für *Netzwerkprozessoren* zusammen; hierbei wird insbesondere auf Charakteristika der verwendeten Einheiten zur Verarbeitung der Datenpakete eingegangen. Da es für programmierbare FSMs eine Vielzahl von Einsatzmöglichkeiten im Kontext der Datenpaketverarbeitung gibt, wird in Kapitel 3 neben einigen etablierten Architekturen bei der Datenpaketverarbeitung das Thema der programmierbaren FSMs aufgegriffen und ein Überblick zum Stand der Technik gegeben.

Kapitel 4 beschäftigt sich mit der Umsetzung der neuartigen, oben erwähnten Verarbeitungseinheit FPE, Kapitel 5 mit der zugehörigen Toolchain der FPE. Kapitel 6 befasst sich mit der in der Arbeit vorgenommenen Evaluation und stellt die wesentlichen Ergebnisse der Untersuchungen vor. Denkbare Perspektiven sowie Einsatzmöglichkeiten für die FPE werden in Kapitel 7 diskutiert. Kapitel 8 fasst die in der Arbeit erzielten Ergebnisse zusammen. Abbildung 1.3 skizziert die wesentlichen Schwerpunkte der einzelnen Kapitel im Kernteil der Arbeit.

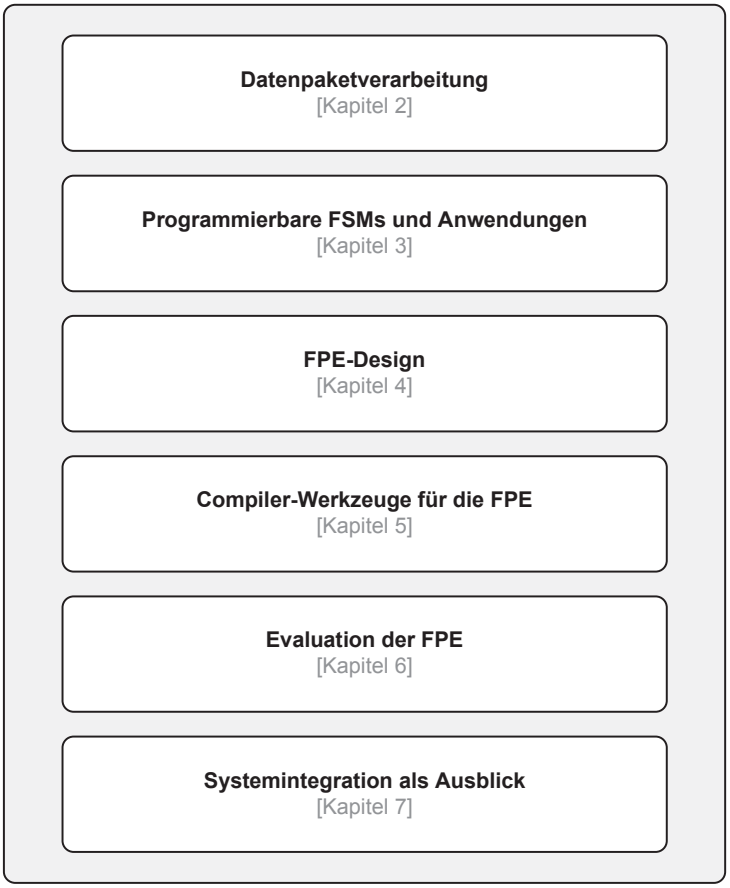


Abbildung 1.3: **Aufbau und Schwerpunkte der Arbeit.**

2 Datenpaketverarbeitung

Zunächst wird in diesem Kapitel eine kurze Einführung in Grundlagen und Anwendungen sowie derzeitige Trends bei der Datenpaketverarbeitung gegeben. Hierbei sollen im Wesentlichen die in der Literatur gebräuchlichen Terminologien bzw. korrespondierende deutsche Begriffe verwendet werden. Nach einer Einführung werden Anforderungen heutiger und zukünftiger Standards abgeleitet und existierende Konzepte für *Netzwerkprozessoren* diskutiert.

2.1 Grundlagen, Anwendungen und Trends

Neue Technologien im Bereich der Informationsverarbeitung, Mikroprozessoren, hochintegrierter Speicher und Übertragungstechnik bildeten und bilden immer wieder einen Grundstein für Mikroelektronikanwendungen mit neue Anforderungen und Perspektiven. Generell ist jetzt die digitale Kommunikation und Vernetzung mit ihren Möglichkeiten kaum mehr aus der modernen Gesellschaft wegzudenken. Das *Internet* beispielsweise ist ein Netzwerk aus abertausenden von untergeordneten Netzwerken (kurz Netzen) größtenteils kommerzieller Anbieter (Service Provider) [Nau05, CL07, S. 2f.], und das *World Wide Web* ein innerhalb des Internets angebotener Dienst. Dieser sowie viele andere Dienste und Netzwerkanwendungen werden mit Hilfen leistungsstarker Computersysteme realisiert.

Aufgrund der jeweiligen Funktion eines Systems im Netz kann zwischen expliziten Netzwerkgeräten und Endgeräten unterschieden werden. Netzwerkgeräte realisieren die Basisfunktionen eines Netzes, Endgeräte zudem Funktionen auf höherer Ebene. Solche Endgeräte, welche anderen Endgeräten entsprechende Dienste im Netz anbieten, sollen als *Server* bezeichnet werden. Hierbei ist in erster Linie eine hinreichend leistungsstarke Mikroprozessorplattform gemeint. Die konkrete Umsetzung von Datentransfers und Protokollen spielt mit Hinblick auf hohe Dienstgüteparameter wie Datentransferrate, Ausfallsicherheit usw. eine offensichtlich entscheidende Rolle.

> Ein Protokoll stellt eine exakte Vereinbarung bezüglich des Datentransfers zwischen Kommunikationsinstanzen dar [Tan03, S. 27], die durch ein Netz miteinander verbunden sind. Eine solche Vereinbarung besteht aus einem Satz von Regeln und Formaten (Syntax), die den Datentransfer der kommunizierenden Instanzen bestimmen (Semantik).

Für drahtgebundene Computernetze hat sich Ethernet [Eth98, Eth10] als auf – im weiteren Sinne – Datenpaketen basierende Technologie breit durchgesetzt. Ethernet umfasst sowohl die Spezifikation von Regel und Formaten sowie Kabeln und Steckern. Mittlerweile gibt es unterschiedlichste Ausprägungen von industriellen,

Home- und Entertainment-Netzwerken, auch solchen in Kraftfahrzeugen oder Flugzeugen [SGF+10], welche auf Ethernet basieren. Entsprechend kostengünstig ist die Realisierung heute. Je nach Anbieter wird auch Ethernet zunehmend im *Backbone* des Internets eingesetzt [HWM+09], wo sonst eine Vielzahl traditionell nicht auf Datenpaketen basierenden Protokolle und Technologien [Kar03] für den Datentransfer über größere Entfernungen eingesetzt werden. Der 10-Gigabit/s-Ethernet-Standard [Eth10] definiert etliche Übertragungstechnologien, darunter Spezifikationen verschiedener Glasfaserkabel- und zweier Kupferkabeltypen. Letztere haben Vorteile gegenüber Glasfaser bezüglich Lebensdauer und Kosteneffizienz. Des Weiteren können mit Kupferkabeln einfacher mehrere Applikationen parallel genutzt werden. Größere Entfernungen lassen sich dagegen einfacher unter Verwendung von Single-Mode-Glasfaserkabeln überbrücken. Aktuelle Zielsetzungen bei der Standardisierung von Ethernet sehen Datenraten von 40 und 100 Gigabit/s vor, wozu verschiedene Beiträge von D'Ambrosia vorliegen [D'A08].

Parallel zu rein Ethernet-basierten Szenarien entstehen sogenannte *Converged Infrastructures*: Ein Beispiel hier ist „Fibre Channel over Ethernet", wobei die durch das Fibre-Channel-Protokoll definierten Pakete innerhalb von Ethernet-Frames[1] übertragen werden. Geräte, welche Ethernet und die sogenannte Fibre-Channel-Technologie (FC) unterstützen, ermöglichen dann den Einsatz von FC-Speichergeräten innerhalb von LANs. Je nach Anwendung werden jedoch fast generell diverse Protokolle nebeneinander verwendet bzw. kombiniert (*Protokollstack*). Dies geschieht zwecks Verminderung der Designkomplexität in festen, als *Layer* bezeichneten Schichten L1, L2, L3, L4, L5, L6 und L7 mit aufsteigendem Abstraktionsgrad [Zim80,IEE07, S. 599f.], wobei diese – zumindest im Zusammenhang mit Netzwerkprotokollen – als *Physical Layer*, *Data Link Layer* und *Network* und *Transport Layer* bezeichnet werden. Entsprechende Datenpakete bestehen dann aus Steuerdaten und zu übertragenden Nutzdaten. Die Steuerdaten kodieren typischerweise Herkunft, Art und Ziel der Nutzdaten. Im Falle von geschachtelten Protokollen beinhalten die Daten eines Layers wiederum ein Datenpaket für den nächsten Layer. Daraus ergeben sich etliche Implikationen für die mikroelektronische Realisierung. In der Zukunft werden sich optische Medien [SDK10] sehr wahrscheinlich weiter durchsetzen, so dass generell Integration und Kopplung mit auf CMOS-Technologie basierenden Schaltungen interessante Fragestellungen und technische Herausforderungen aufwerfen. Auf der anderen Seite werden sich sicher auch neue Formen drahtloser Übertragungstechnik herausbilden [Wel09], welche ebenfalls leistungsfähige Prozessoren für die Datenpaketverarbeitung benötigen.

Maßgebend ist die Entwicklung der für die Internetdienste verwendeten Protokolle: Trotz einiger Problematiken sind die in den achtziger Jahren entwickelten TCP/IP-Protokolle [SW94] ein fester Bestandteil heutiger Hardware und Software in Computersystemen [PH09]. Ethernet gilt hier als Netzwerktechnologie. Die TCP/IP-Pakete werden wie bei dem „FC over Ethernet" ineinander geschachtelt und innerhalb

[1] In Zusammenhang von Ethernet soll durchweg von *Frames* und nicht von Paketen gesprochen werden, ohne auf die Unterscheidung hier einzugehen.

der Ethernet-Frames übertragen. Die Datenpaket- und Protokollverarbeitung auf Ebene von TCP/IP sollte dann entsprechend schritthalten können, was für Datenraten jenseits der derzeit in privaten Computernetzwerken verbreiteten Größe von knapp 1 Gigabit/s eine Herausforderung darstellt. In einer Arbeit von Hauger et al. [HWM+09] wird ein Überblick über die Fragestellungen und Lösungsmöglichkeiten für 100 Gigabit/s und 1000 Gigabit/s vorgestellt. Das Fazit der Autoren ist, dass mittels stark angepasster, schneller und paralleler Prozessorsysteme eine schritthaltende Verarbeitung durchaus denkbar ist. Jedoch müssen auch die verwendeten Protokolle vereinfacht und Pufferspeicheranforderungen deutlich reduziert werden.

Im Gegensatz zu TCP/IP als Netzwerktechnologien für den *Inter-System*-Datentransfer kommt sogenannten I/O- oder Interconnect-Technologien ebenfalls eine hohe Bedeutung in *Servern* zu. Auf diesen basiert häufig ein Großteil des *Intra-System*-Datentransfers. Exemplarisch kann hier der PCI Express-Standard (PCIe) [BAS03], das durch die Firma AMD initiierte Hyper Transport (HT) [AT03], InfiniBand (IB) [SW03] sowie proprietäre Lösungen wie der GX-Bus von IBM [CCD+09] angeführt werden. Solche Bus-Protokolle sind insbesondere für die hochratigen Datentransfers über kurze bis sehr kurze Distanzen (zwischen Speicher und Mikroprozessoren) gedacht. Sie ersetzen dabei allesamt klassische, nach einem *Request-Grant*-Prinzip arbeitende Bussysteme durch *Credit*-basierte *Punkt-zu-Punkt-Übertragungen* [HPAD07, S. 390]. Dadurch kann die zur Verfügung stehende Bandbreite optimal ausgenutzt werden. Beispielhafte Anwendungen sind externe Peripherie-Geräte wie Festplatten-Arrays, Netzwerkkarten oder Graphikkarten, welche Daten aus dem Hauptspeicher beziehen können.

Um ein repräsentatives Beispiel aus dem High-End-Bereich anzuführen, soll nun auf die I/O-Systemarchitektur des Systems z10 *Mainframe-Servers* von der Firma IBM verwiesen werden [SBDT+09, CCD+09, MWS09]. Abbildung 2.1 stellt eine Übersicht der I/O-Strukturen im System z10 dar. Mit Hilfe der Cache und I/O-Strukturen lassen sich bis zu 80 solcher Module zu einem Cluster koppeln. Im Vordergrund steht dabei neben Bereitstellung von Diensten mit hoher Verarbeitungsleistung die Ausfallsicherheit durch (Cluster-)Protokolle sowie durch redundante Hardwarestrukturen.

Für die verschiedenen eingesetzten Protokolle (Ethernet, FC-Protokoll, Cluster-Protokoll, InfiniBand usw.) sind spezifische Chips erforderlich, um die unterschiedlichen *Busse* wie GX, IB, PCI usw. zu koppeln. Dadurch entsteht zwar ein modulares und sehr leistungsfähiges System-Design. Änderungen an den Protokollen oder dem Aufbau von Datenpaketen schließen in der Regel jedoch Änderungen an den I/O-Chips mit ein. In anderen Worten: Eine stellenweise Konfiguration oder Programmierung wäre anstelle zeit- und kostenintensiver Re-Designs der Chips wünschenswert, auch im Sinne einer Markunabhängigkeit oder Multistandard-Verarbeitung.

2.1.1 Aufbau und Verarbeitung von Datenpaketen

Diese Arbeit befasst sich vorrangig mit Paket-basierter Datenübertragung, bei welcher der Datentransfer mittels gekapselter Dateneinheiten umgesetzt wird. Eine

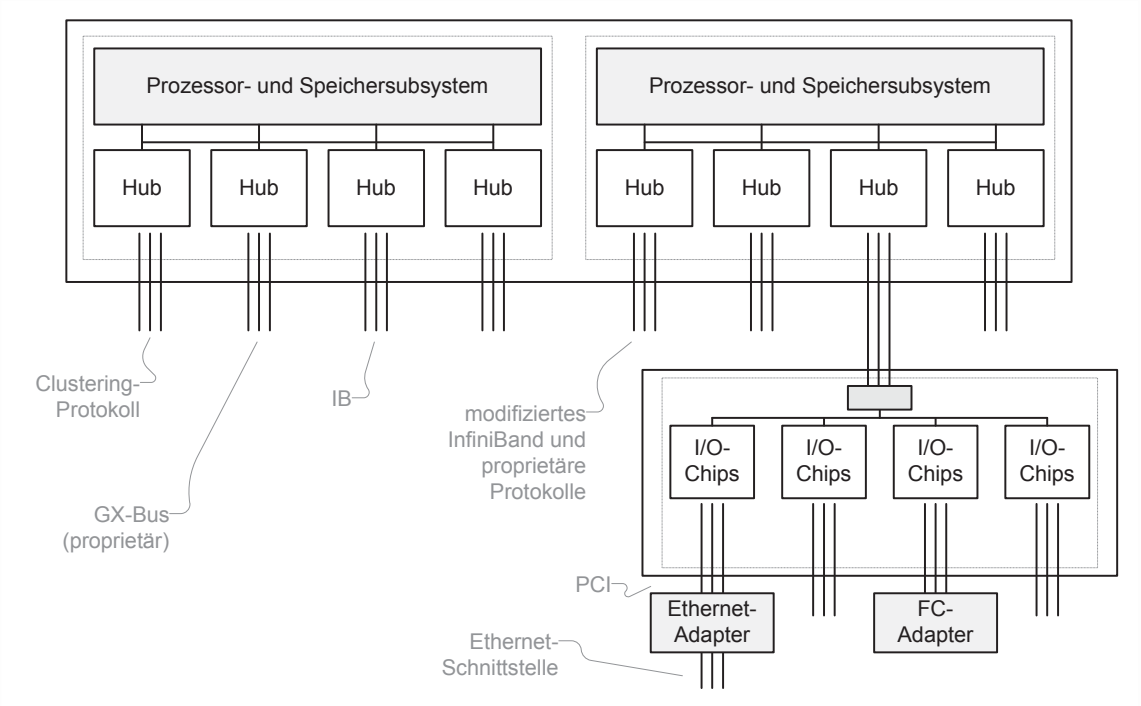


Abbildung 2.1: **I/O-Systemarchitektur im System z10.**

solche Dateneinheit wird als Datenpaket, oder kurz *Paket*, bezeichnet. Die Steuerinformation mit den Protokoll-relevanten Daten wird als *Header* bezeichnet, einzelne darin enthaltene Bitgruppen als *Header-Felder*. Diese Felder müssen bei der Verarbeitung auf der Sendeseite erzeugt und auf der Empfangsseite interpretiert werden. Um den Aufbau und die Verarbeitung von Datenpaketen zu veranschaulichen, sollen zwei einfache Beispiele verwendet werden.

TCP/IP over Ethernet

Ein Beispiel für den Aufbau eines TCP/IP-Datenpakets ist mit Abbildung 2.2(a) gegeben. IP (Internet Protocol) ist die Basis der TCP/IP-Protokolle [SW94, S. 33]. Die Lenkung von Paketen im Netz gemäß der Adressen (Paket-Routing) ist hier das zentrale Konzept bei IP. TCP (Transmission Control Protocol) ist ein zweites, auf IP aufsetzendes Protokoll, welches den eigentlich Datentransfer zwischen den Kommunikationsinstanzen steuert. TCP-Pakete werden dann wie andere Protokolle[2] innerhalb der IP-Pakete versendet. IP selbst ist nicht verbindungsorientiert und auch nicht mit Garantie zuverlässig. Pakete selbst können später in veränderter Reihenfolge ankommen oder sozusagen „verloren“ gehen.

Die wesentlichen Felder im IP-Paket-Header sind die 32-bit Quell- und Ziel-Adresse. Zusammen mit dem Feld „Daten-Offset“ und einer Längenangabe kann Anfangs-

[2] UDP, ICMP oder etwa IGMP

und Endposition der Daten bestimmt werden. Die Prüfsumme sichert bei IP den Paket-Header. Da IP-Pakete *fragmentiert* werden können, muss ebenfalls Information zur Behandlung der Fragmente auf Empfängerseite im IP-Header vorgesehen werden. Das Feld „TTL“ wird beim Vorgang des Paket-Routings fortlaufend dekrementiert, es entspricht der Lebensdauer eines IP-Paketes (kurz: *time-to-live*).

TCP umfasst im Wesentlichen folgende Aufgaben: Verbindungsauf- und -abbau, Aufteilen und Transfer von Daten über einen Byte-Stream, Flusskontrolle, Überprüfung und Bestätigung von Transfers. Unterschiedliche Verbindungen können über *Ports* unterschieden werden. Der TCP-Header beinhaltet neben Quell- und Zielport, einem „Daten-Offset“-Feld und Flags zur genauen Spezifikation des Pakettyps (als Bestätigungspaket, Paket zum Verbingungsauf- oder -abbau etc.) noch die wichtigen Felder „Sequenznummer“, „Bestätigungsnummer“ und „Empfangsfenster“. Mit Hilfe dieser wird eine Byte-weise Bestätigung von Daten sowie mit der Angabe des Empfangsfensters eine Flusskontrolle realisiert. Die Prüfsumme dient dem Erkennen von Fehlern beim Transfer (vgl. Anhang A.2). Typischerweise werden sämtliche Daten ebenfalls noch mit einer weiteren Prüfsumme auf Ebene der bis zu 1518 Byte großen Ethernet-Frames gesichert.

Wesentliche Verarbeitungsschritte für TCP/IP liegen demnach darin:

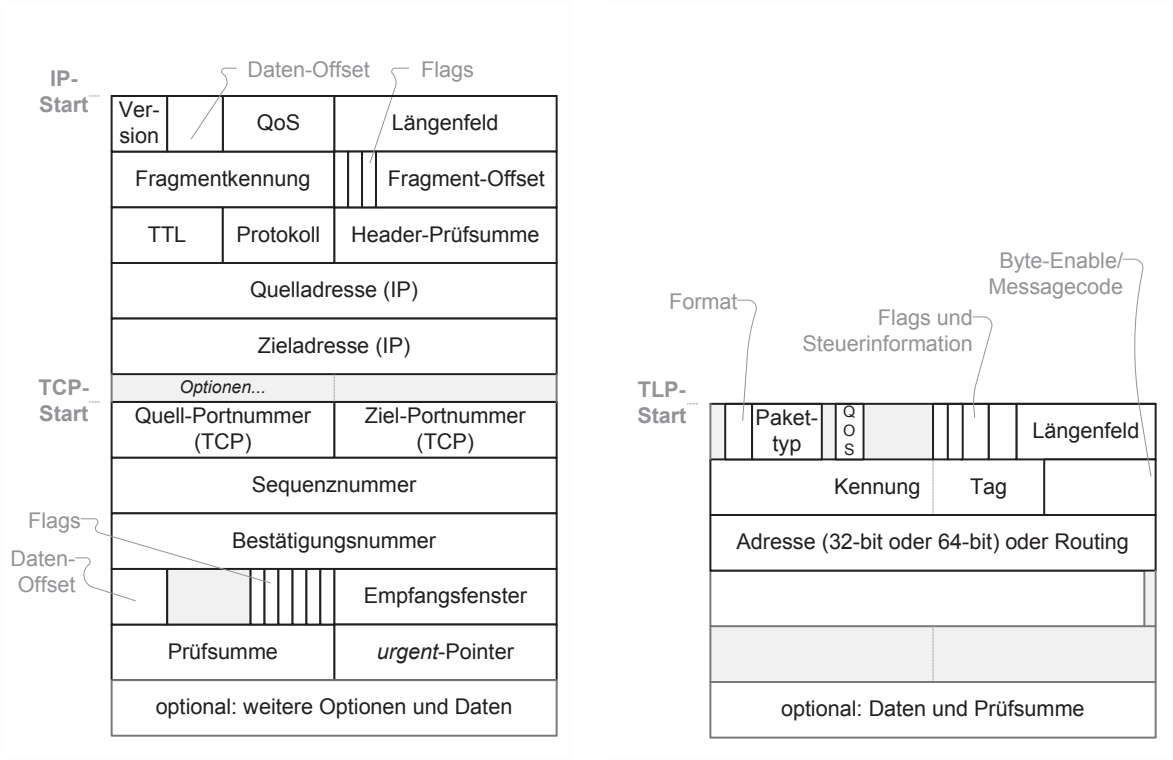
- Pakete auf IP-Ebene anhand der IP-Adressen anzunehmen[3] oder gegebenenfalls weiterzuleiten.
- Generell müssen eine mögliche Fragmentierung oder vertauschte Paketreihenfolge aufgehoben werden, wozu Pufferspeicher notwendig sind.
- Verbindungsbezogene Daten gilt es, zu pflegen und eine Pufferung der eingehenden Daten zu ermöglichen.
- Da zudem die Zahl unterschiedlicher Verbindungen, die sich über IP-Adressen plus Ports definieren lassen, sehr hoch[4] ist, stellt das Laden der zu einem eingehenden Paket gehörigen Verbindungsdaten typischerweise eine Herausforderung dar.

Weiterhin sind mit den TCP/IP-Protokollen verschiedene Algorithmen und Optionen definiert [RFC11], welche funktionale und zeitliche Vorgänge bei der Verarbeitung näher spezifizieren. Diese gilt es bei der Umsetzung des Protokolls zu berücksichtigen.

Generell ist ein eingehender *Paket-Header* zu analysieren und im Kontext des zugehörigen TCP/IP-Pakets zu bewerten.

[3] Die Zuordnung von Ethernet-Adressen auf IP-Adressen geschieht mittels eines eigenständigen Protokolls, ARP (Address Resolution Protocol).

[4] Diese Aussage gilt auch in der Praxis für leistungsstarke *Server*.)



(a) Aufbau eines TCP/IP-Pakets (b) Aufbau eines PCIe-Pakets

Abbildung 2.2: **Grundsätzlicher Aufbau zwei verschiedener Paket-Header.**

PCI Express

Ein weiteres Beispiel für den Aufbau eines Datenpakets ist mit Abbildung 2.2(b), jetzt stellvertretend für den Bereich der Interconnect-Technologien, gegeben. PCI Express (PCIe) unterscheidet eine ganze Reihe unterschiedlicher Pakettypen und Paketformate, wo unterschiedliche Computer- und Bussystemstrukturen möglichst auch durch den neuen Standard abgebildet werden sollen. Das PCIe definiert unterschiedliche Layer mit unterschiedlichen Funktionen [BAS03, S. 55ff.]. Hier gezeigt ist ein PCIe-Paketformat nach dem Standard PCIe Gen.3 für eine Speicheranforderung ein TLP (Transaction Layer Packet). Zur Unterscheidung der Pakettypen und Paketformate gibt es zwei Felder, welche gleich am Anfang eines Paket-Headers liegen: „Format“ und „Pakettyp“. Für eine Speicheranforderung werden weiterhin eine Adresse und eine Länge benötigt; dazu ist im Header mit einem hier als „Kennung“ markierten Feld die anfordernde Instanz festgehalten. Zusätzlich werden Pakete durch ein *Tag* markiert. Das Paket spezifizierende Steuerinformationen und eine optionale Prüfsumme werden ebenfalls mit dem Header gesendet.

Vorschriften zu Abarbeitungsreihenfolgen und zeitlichen Vorgaben sind in den PCIe-Spezifikation [PCI11] festgehalten.

> Generell ist ein eingehender Paket-Header im PCIe-Interface zu analysieren und im jeweiligen Systemkontext zu bewerten.

2.2 Ableiten von Anforderungen

Typischerweise kann die Datenpaketverarbeitung gut durch die Aufteilung in Paket-Header-Verarbeitung und Datentransferoperationen beschrieben werden. Diese Aufteilung ähnelt somit grob dem Modell von Datenfluss und Kontrollfluss in Anlehnung an [RPK+00, Wil01], wenngleich praktisch betrachtet die Paket-Header-Verarbeitung auch Berechnungen, etwa von Adressen oder Parametern, miteinschließt. Die Datenpaketverarbeitung kann zudem wie jede Verarbeitungsvorschrift mit Hilfe eines Flussdiagramms dargestellt werden. Exemplarisch soll Abbildung 2.3 betrachtet werden, welche in Anlehnung an eine PCIe-Speicheranforderung ein bewusst allgemein gehaltene Paketverarbeitungsvorschrift zeigt. Auffallend sind hier die verschiedenen Mehrfachverzweigungen. Analoge Beispiele für IP oder PCIe wäre die Auswertung der *Header-Felder* Flags und Protokoll bei IP oder Pakettyp oder Format bei PCIe. Das heißt, abhängig von Header-Daten und Steuerungsinformation ist ein jeweiliger Verarbeitungsteilschritt durchzuführen. Für eine Speicheranforderung müssen also entsprechende Start- und Endadressen für die Daten bestimmt werden, oder es muss entschieden werden, welche konkreten Anfragen an das Speichersubsystem gestellt werden.

Ganz konkrete Funktionen, welche jede Einheit zur Datenpaketverarbeitung beherrschen muss, bestehen folglich im Auslesen und Modifizieren von *Header-Feldern*. Dazu sind typischerweise Maskier- bzw. Bit-Selektion-Operationen notwendig. Weiterhin müssen Adressen berechnet werden, nicht nur für interne Daten- und Kon-