Humboldt University Berlin
Computer Science Department
Systems Architecture Group

Rudower Chaussee 25
D-12489 Berlin-Adlershof
Germany

Phone: +49 30 2093-3400
Fax: +40 30 2093-3112
http://sar.informatik.hu-berlin.de

Interrupt-Behandlungskonzepte für die HHI CineCard-Familie

HU Berlin Public Report
SAR-PR-2006-14

December 2006

Author(s):
Robert Sperling

Zusammenfassung

Die qualitativ hochwertige Projektion auf sehr große Bildflächen stellt enorme Anforderungen an die digitale Projektionstechnik. Die hohen Qualitätsansprüche von Großprojektionen in Bezug auf Auflösung und Helligkeit können von einzelnen Projektoren auch mittelfristig nicht befriedigt werden.

Das Heinrich-Hertz-Institut hat mit der CineCard eine modulare und sehr flexible Grundlage für die kostengünstige Multiprojektion entwickelt. Sie bildet eine ideale Plattform für Projektionen mit sehr hohen Auflösungen oder ungewöhnlichen Bildseitenformaten.

Ähnlich wie viele andere Gerätetypen aus dem Multimediasektor erfordern die PCI-Karten der CineCard-Familie (z.B. bei Transportstromfehlern) eine sehr komplexe Interrupt-Behandlung. Aufgetretene Interrupts müssen bedingt durch die hohen Datenraten der wiedergegebenen Datenströme innerhalb eines sehr engen Zeitfensters bearbeitet werden. Hohe Effizienz und große Flexibilität sind deshalb für die Interrupt-Behandlung der CineCards unverzichtbar.

In dieser Dipmomarbeit werden verschiedene Konzepte der Interrupt-Behandlung auf ihre Zweckmäßigkeit hin untersucht und miteinander verglichen. Die dafür notwendigen Vergleichskriterien werden im Rahmen der Anforderungsanalyse bestimmt und orientieren sich stark an der späteren Verwendung der CineCard und dem bestehenden Anwendungsumfeld.

Ziel der Arbeit ist es, ein hinreichend optimales Interrupt-Behandlungskonzept für alle PCI-Karten der HHI CineCard-Familie zu entwickeln und zu implementieren.

Christian Weißig/Robert Sperling

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Projektübersicht

Das Heinrich-Hertz-Institut hat sich im Rahmen des Projektes CineVision 2006 zum Ziel gesetzt, eine flexible Infrastruktur zur ultra-hochauflösenden Projektion digitaler Bildinhalte auf sehr große Bildflächen zu schaffen.

Dieses Ziel soll mit Hilfe der Multiprojektion, der Projektion eines großen Gesamtbildes mit mehreren Projektoren, erreicht werden. Die Qualitätsansprüche heutiger Kino- und Großprojektionen überschreiten das Auflösungsvermögen marktüblicher Video-Projektoren bei weitem. Im aktuellen Projekt der Abteilung Bildsignalverarbeitung wird mit 5 SXGA+ Projektoren eine Gesamtauflösung von 5000×1400 Bildpunkten erreicht.

In einer späteren Ausbaustufe soll mit 20 Projektoren eine Auflösung von 11776×2048 Bildpunkten erreicht werden. Diese Auflösungen sind auch auf längere Sicht nicht mit nur einem einzigen Projektor realisierbar. Die hohen Qualitätsansprüche und damit verbundenen hohen Auflösungen erfordern einen großen technischen Aufwand, der sich bei bestehenden Projektionslösungen in erster Linie im Preis der verwendeten Komponenten niederschlägt. Speziallösungen einiger Projektoren-Hersteller disqualifizieren sich zudem oft durch mangelnde Flexibilität hinsichtlich des projizierten Bildseitenverhältnisses.

Die Kernkomponenten der Multiprojektion bilden die HHI CineCards. Sie ermöglichen es beliebig viele Projektoren horrizontal wie vertikal zu kaskadieren und zu einem Gesamtbild zusammenzufügen. Mit Hilfe dieser sehr modularen Lösung lassen sich nicht nur sehr hohe Auflösungen erreichen, sondern auch beliebige Bildseitenformate projizieren.

Die Vorteile der Multiprojektion und die damit verbundenen technischen Herausforderungen bei der Projektion von MPEG2 komprimierten Videodatenströ-

Abbildung 1.1: Präsentation auf der IBC 2005 in Amsterdam

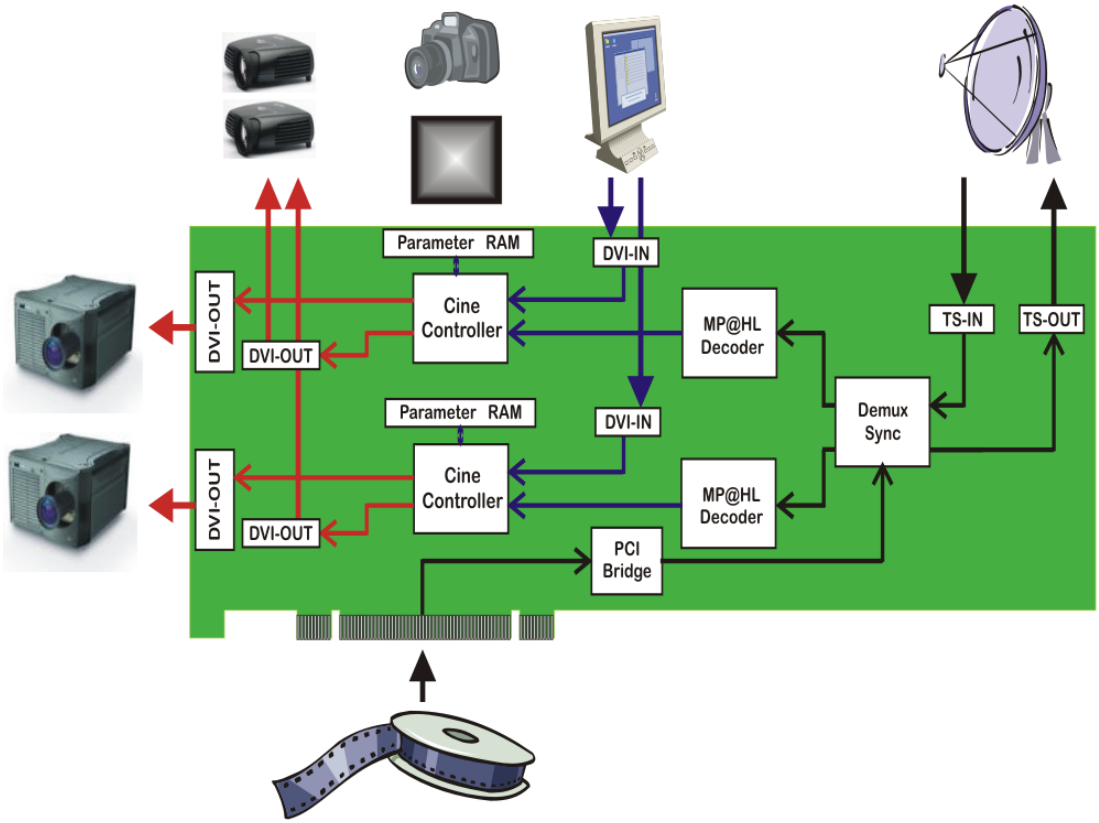
men werden ausführlich in der Studienarbeit zur HHI CineCard[1] behandelt. Zusammengefasst sind die wichtigsten im Rahmen des Projektes gelösten technischen Herausforderungen:

1. Projektion:
   (a) Nahtlose Überblendung der Stoßstellen der Projektorenbilder,
   (b) Geometrische Entzerrung des Bildes auf gekrümmten Leinwänden und
   (c) Abgleich der unterschiedlichen Farbtemperaturen und Farbräume der verschiedenen Projektoren.
2. Synchronisation:
   (a) Mindestens bildgenau synchronisierte Ausgabe von
   (b) verschiedenen (MPEG2 komprimierten) Bildquellen mit variabler Bitrate.

Die HHI CineCards

Die Familie der HHI CineCards (CineCard, HyperCard I/II) besteht aus mehreren, in ihrem Hardware-Aufbau sehr unterschiedlichen PCI-Karten. Die HHI HyperCard II ist in der Lage Videodatenströme von verschiedenen Quellen aufzuzeichnen und besitzt einen analogen Video-Ausgang, über den Videodaten mit einer Farbtiefe von 36 Bit wiedergegeben werden können.

Abbildung 1.2: Anwendungsmöglichkeiten der HHI CineCard



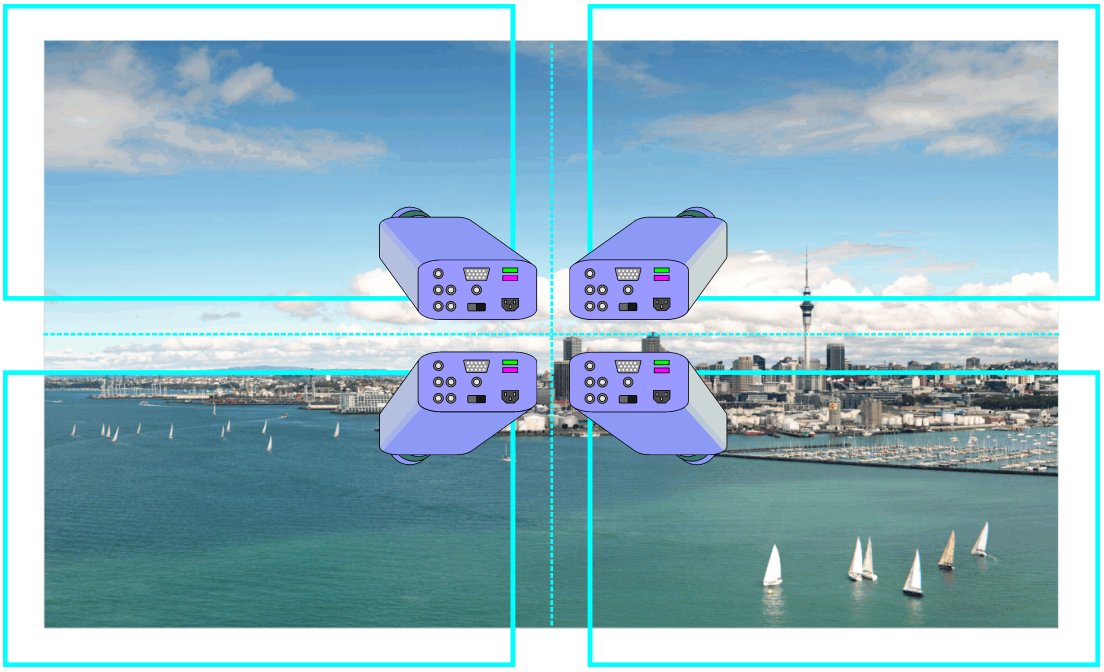
Für die kostengünstige Projektion mit einer sehr hohen Gesamtauflösung wurde die HHI CineCard entwickelt. Sie ist in der Lage, über DVI-Ausgänge zwei HDTV-Projektoren oder bis zu vier XGA-Projektoren anzusteuern. Die Überblendungen an den Überlappungsstellen der Teilbilder und die kolormetrische Anpassung der unterschiedlichen Projektoren werden von der CineCard-Hardware während der Ausgabe in Echtzeit berechnet werden. Die dafür notwendigen Blendparameter können mit Hilfe einer Digitalkamera im Rahmen einer einmaligen Kalibrierung automatisch bestimmt werden.

Die Projektionsmatrix

Jedes Mitglied der CineCard-Familie besitzt mindest einen Videoausgang. Bis zu vier CineCards lassen sich in einem Hostsystem gleichzeitig nutzen. Darüber hinaus lassen sich beliebig viele Hostsysteme zu einem großen Projektionsverbund zusammenschließen.

Die CineCard-Software verwendet zur Zuordnung der einzelnen CineCards zu ihrer Position auf der Projektionsfläche eine dreidimensionale Projektionsmatrix. Die X- und Y-Komponente einer Koordinate stehen für die Position auf der Projektionsfläche. Die Z-Komponente wird verwendet, wenn mehrere Projektoren z.B. bei 3D-Projektionen die gleiche Fläche auf der Projektionsleinwand bestrahlen.

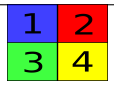
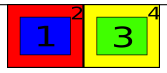
Abbildung 1.3: Beispiel für eine 2x2 Projektionsmatrix



Verfügt eine CineCard über mehr als nur einen (genutzten) Videoausgang bezieht sich die Koordinate in der Projektionsmatrix auf den mit dem ersten Ausgang der Karte verbundenen Projektor. Die Positionen der anderen von der Karte angesteuerten Projektoren wird durch einen so genannten Anordnungsindex bestimmt.

Alle Videodatenströme einer CineCard befinden sich in einem gemeinsamen Transportstrom. Die Anordnung der von der CineCard angesteuerten Projekto-

Tabelle 1.1: Beispiele für den Anordnungindex in der Projektionsmatrix

Anzahl der Projektoren	Anordnungsindex (binär)	Positionen der Projektoren relativ zueinander
1	00 (00 000)	
2	09 (01 001)	
	10 (01 010)	
	12 (01 100)	
4	25 (11 001)	
	27 (11 011)	
	29 (11 101)	

ren muss somit mit der Anordnung der im Transportstrom abgelegten Videobilder übereinstimmen. Um dies zu gewährleisten werden auch alle Datenströme mit einem Anordnungsindex versehen, der mit dem der wiedergebenen CineCard übereinstimmen muss.

1.2 Die Diplomarbeit

1.2.1 Ziel der Arbeit

Die PCI-Karten der CineCard-Familie verwenden zum Dekodieren von Videodatenströmen den MPEG2-Dekoder MB87j2020[3] (HiPEG+). Transportstromfehler, die zu einem Fehler im MPEG2-Dekoder führen, lösen so genannte Interrupt-Requests (auch Interrupt-Anforderung oder IRQ genannt) aus. Unbehandelte Interrupt-Requests dieser Art äußern sich sofort in einer gestörten Videowiedergabe. Neben dem MPEG2-Dekoder können auch andere Subsysteme der CineCard Interrupt-Requests auslösen.

In der Studienarbeit zur CineCard[1] wurden bereits die in direktem Zusammenhang mit der Wiedergabe und Aufnahme von Datenströmen stehenden und relativ einfach zu verarbeitenden Interrupt-Requests ausführlich besprochen.

Diese Diplomarbeit widmet sich den wesentlich aufwändiger zu handhabenden Interrupt-Requests des Datenstromdekoders zur Fehlerbehandlung. Auf Basis des in der Studienarbeit zur CineCard entwickelten Linux-Treibers soll ein Konzept zur Behandlung dieser Interrupt-Requests entwickelt und implementiert werden. Beim Entwurf des Konzeptes stehen besonders die folgenden Entwurfsziele im Vordergrund:

1. Möglichst große Flexibilität der Interrupt-Behandlung,
2. Nahtlose Integration in das bestehende Anwendungsumfeld,
3. Nach Möglichkeit Vermeidung von Änderungen an der Hardware und
4. Zukunftssicherheit durch hohe Wartbarkeit und gute Erweiterbarkeit.

1.2.2 Aufbau der Diplomarbeit

Das einleitende Kapitel stellt neben der Problematik Multiprojektion das Kernthema dieser Diplomarbeit vor: Interrupts. Ausgehend von der historischen Entwicklung werden hier einige wichtige Eigenschafen von Interrupts näher erläutert.

Das Kapitel 2, die *Problemanalyse*, beschäftigt sich mit den Anforderungen an das zu entwickelnde Konzept der Interrupt-Behandlung und mit der Ausgangsbasis für die zu erstellende Lösung. Der erste Teil befasst sich mit den festen Voraussetzungen, die ein Lösungsansatz erfüllen muss, um als mögliche Problemlösung in Betracht zu kommen. Im zweiten Teil wird näher auf die Ausgangsbasis in Hinblick auf die Hardware und das bestehende Anwendungsumfeld der Entwicklung eingegangen. Im dritten Teil werden basierend auf der Ausgangssituation geeignete Analysekriterien ermittelt, um gefundene Lösungsstrategien besser

bewerten zu können. Aus den gefundenen Analysekriterien wird anschließend eine Bewertungsmetrik erstellt, mit deren Hilfe die untersuchten Lösungsstrategien miteinander verglichen werden können.

Im Kapitel 3 werden *mögliche Problemlösungen* untersucht. Aus bestehenden Lösungen anderer Gerätetreiber lassen sich neue Herangehensweisen ableiten. Basierend auf den Gegebenheiten der verwendeten (CineCard-)Hardware und des Anwendungsumfeldes werden vier mögliche Lösungsansätze vorgestellt und auf ihre Zweckmäßigkeit hin untersucht. Mit Hilfe der zuvor entwickelten Bewertungsmetrik werden diese miteinander verglichen. Zusätzlich werden Faktoren disktutiert, die die Umsetzung begünstigen oder erschweren, sich aber nur mittelbar auf die Bewertung auswirken.

Kapitel 4 widmet sich der *Beschreibung der implementierten Lösung*. Dabei wird besonders auf die Einbettung des Interrupt-Behandlungssystems in das bestehende Anwendungsumfeld eingegangen.

1.2.3 Konventionen und wichtige Begriffe

Zur CineCard-Familie gehören die `HHI HyperCard II` und die `HHI CineCard` bzw. die `HHI CineCard Pro`. Die PCI-Karten der CineCard-Familie werden im Folgenden allgemein als *CineCards* bezeichnet. Bezieht sich eine Textpassage ausschließlich auf die `HHI CineCard/HHI CineCard Pro`, wird dies im Text gesondert hervorgehoben.

Um Verwechselungen zu vermeiden, wird das Computersystem, in dem die CineCards betrieben werden, in dieser Diplomarbeit *Host-System* genannt. Die Bezeichnung PC oder auch Computer wäre in diesem Falle nicht ausreichend bzw. nicht korrekt, weil es sich bei dem *Host-System* auch um ein Embedded-System, einen PowerPC oder eine UltraSparc-Workstation handeln könnte. Die allgemeine Bezeichnung Computersystem würde auch auf die im MPEG2-Dekoderkern integrierte Prozessoreinheit zutreffen und kann deshalb auch nicht verwendet werden. Um allgemein von einem Computersystem zu sprechen, dass Interrupt-Requests behandelt, wird die Bezeichnung *behandelndes System* verwendet.

Unter Linux (wie auch in dieser Diplomarbeit) werden durch Hardware-Komponenten ausgelöste Interrupts als *Interrupt-Request, Interrupt-Anforderung oder auch IRQ* bezeichnet, um der besonderen Bedeutung der Bearbeitung der Interrupts durch den Interrupt-Controller Rechnung zu tragen.

In der Fachliteratur werden die Begriffe *Interrupt-Service-Routine (ISR)* und *Interrupt-Handler* oft als Synonym verwendet, dies ist jedoch nicht ganz korrekt. Der *Interrupt-Handler* ist im engeren Sinne eine Funktion zur Detektion der Quelle eines *Interrupt-Requests*, das für die Behandlung des *Interrupt-Requests* zustän-

dige Gegenstück ist die *Interrupt-Service-Routine (ISR)*. Beide Teile zusammen bilden die *Interrupt-Routine*.

Bei der Ausführung von Programmcode werden zwei Typen von Funktionen unterschieden: Die sich im *Kernelspace* befindenden Programmteile werden im so genannten *Kernel-Kontext* ausgeführt. Analog dazu werden Applikationen der Benutzer im *Userspace* bzw. *User-Kontext* ausgeführt. Die gesonderte Rolle des Programmcodes in der *Interrupt-Routine* des Kernels bzw. des Gerätetreibers wird durch den Begriff *Interrupt-Kontext* hervorgehoben.

Als ein *Konzept zur Interrupt-Behandlung* wird die prinizipielle technische Vorgehensweise zur Reaktion auf einen aufgetretenen Interrupt-Request bezeichnet. Die konkrete *Strategie zur Behandlung eines Interrupts* kann sich dabei je nach Situation und Implementierung (zum Beispiel durch Parametrisierung der Interrupt-Routine) ändern.

Viele der in dieser Diplomarbeit verwendeten Fachbegriffe besitzen eine mehr oder weniger passende deutsche Übversetzung. Aus Gründen der besseren Lesbarkeit und Klarheit werden hier die in der Fachliteratur und im Kernel-Umfeld etablierten Bezeichnungen verwendet. Die wichtigsten dieser Begriffe werden in einem Glossar zusammengefasst und erklärt.

1.3 Interrupts und deren Behandlung

In der Informatik steht der Begriff *Interrupt* für eine kurzfristige Unterbrechung des aktuell laufenden Prozesses (bzw. Programms) durch eine zuvor definierte Befehlssequenz, die Interrupt-Routine. Nach Abschluß der Interrupt-Routine wird die Ausführung des Programms an der Unterbrechungsstelle fortgesetzt.

Ziel des Interrupt-Konzeptes ist es, schnell auf Ein-/Ausgabe-Anforderungen von Hardware-Geräten (zum Beispiel Tastatur, Festplatte, Soundkarte oder System-Timer) reagieren zu können. Interrupts zeichnen sich durch drei charakteristische Eigenschaften aus:

Zeitkritisch: Wenn ein anstehender Interrupt nicht innerhalb eines vorgegebenen Zeitfensters bearbeitet werden kann, drohen dadurch zum Beispiel Datenverluste oder Systemabstürze (hardwarespezifische Sicht).

Transparent: Laufende Anwendungen werden (außer bei Exceptions) nicht von Interrupt-Behandlungen beeinflusst (anwendungsspezifische Sicht).

Exklusiv: Andere Interrupt-Routinen werden nicht (unbeabsichtig) beeinflusst. (interruptspezifische Sicht).

Das Interrupt-Konzept wurde erstmalig im Zusammenhang mit dem UNIVAC (oder auch ERA für Engineering Research Associates genannt) erwähnt. Zunächst entwickelt, um Ausnahme-Fehler des laufenden Programms zu behandeln, wurden Interrupts schnell auch für die Kommunikation mit Hardware-Komponenten verwendet. Der UNIVAC 1103 wurde ursprünglich zur Stapelverarbeitung von Programmen entworfen und später mit Hilfe des Interrupt-Konzeptes für die Echtzeitverarbeitung von Daten aus einem Windkanal modifiziert. Ein ähnlicher Ansatz, jedoch zusätzlich mit maskierbaren Interrupts, wurde bei der PDP-8 etwa zehn Jahre danach verfolgt.

Die heute übliche Methode der Interrupt-Steuerung mit programmierbaren Timer-Interrupts und pro Prozess definierbaren Interrupt-Vektor-Tabellen wurde erstmals im IBM Stretch[4] umgesetzt. In modernen Computersystemen werden Interrupts nicht nur zur Kommunikation mit den Hardware-Komponenten des Computers verwendet, sondern auch für die Speicherverwaltung und den Speicherschutz eingesetzt. Selbst das in heutigen Betriebssystemen unverzichtbare preemptive Multitasking ist ohne Interrupts nicht realisierbar, da Tasks sonst nicht ohne ihr eigenes Zutun unterbrochen und umgeschaltet werden können. Bezüglich ihrer Wirkungsweise lassen sich Interrupts in zwei Grundtypen einteilen:

Synchrone Interrupts: Die auch Ausnahmen oder Exceptions genannten synchronen Interrupts werden von der CPU selbst ausgelöst und sind nur für den aktuell laufenden Prozess bestimmt (z.B. bei der Division durch Null).

Asynchrone Interrupts: Nicht an den aktuellen Prozess gebundene Interrupt-Requests werden durch einen Interrupt-Controller bearbeitet und meist von bezüglich der CPU externen Geräten ausgelöst.

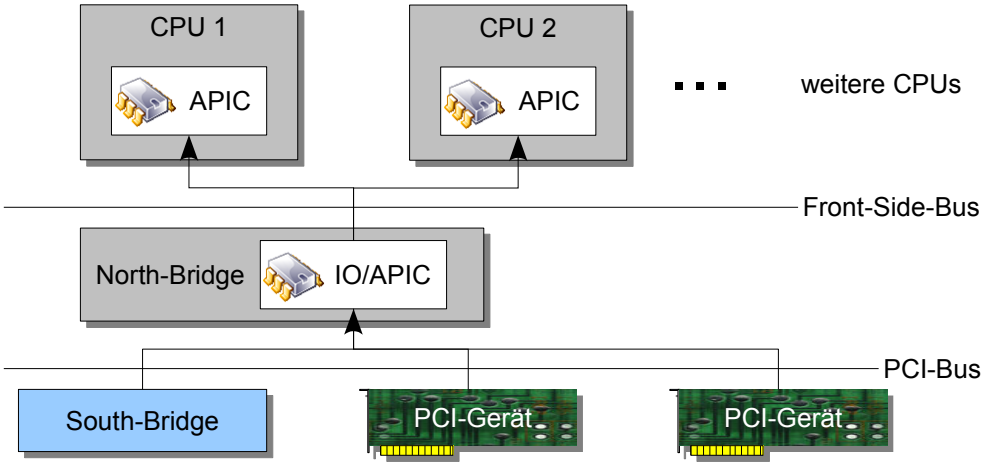
Die im Linux-Kernel zusätzlich verwendeten Software-Interrupts sind vollständig ohne Hardware-Unterstützung realisiert. Sie besitzen keine Gemeinsamkeit mit den klassischen Interrupts[6, Seite 606] und werden verwendet, um zeitverzögerte Vorgänge im Kernel effektiv zu implementieren. Beim Auftreten von Hardware-Interrupts anstehende Software-Interrupts werden regelmäßig am Ende des zentralen Interrupt-Handlers abgearbeitet.

1.3.1 Die Interrupt-Verarbeitungskette

Jedes Hardware-Gerät, das einen Interrupt auslösen kann, ist mit dem so genannten Interrupt-Controller (PIC), respektive dem IO-APIC[1], verbunden. Die auf x86-Computern verbreitete Kombination aus zwei 8256A/8259-PICs stellt 15 statische Interrupt-Leitungen zur Verfügung.

[1] (Advanced) Programmable Interrupt Controller

Abbildung 1.4: APIC/IO-APIC



Statt statischer Interrupt-Leitungen wie beim klassischen PIC, wird bei der so genannten APIC/IO-APIC-Architektur ein serieller Bus verwendet, der wesentlich mehr (virtuelle) Interrupt-Leitungen zur Verfügung stellen kann. Der IO-APIC modernerer x86-Computer stellt üblicherweise 24 Interrupt-Leitungen pro Prozessor bereit. Dieses Konzept wird auch bei anderen Prozessorarchitekturen verwendet. Beispielsweise stehen auf DEC-Alpha Wildfire-Boards 2048 virtuelle Interrupt-Leitungen zur Verfügung. Die Interrupt-Leitungen werden ständig vom (A)PIC überwacht. Stellt der Controller ein anliegendes Interrupt-Signal fest, erzeugt er einen dazu gehörigen Zahlenwert, den Interrupt-Vektor. Anschließend wird dem Prozessor mitgeteilt, dass ein neuer Interrupt-Vektor gelesen werden kann. Die CPU bestätigt den Empfang und der (A)PIC setzt die Überwachung der Eingangsleitungen fort. Die weitere Bearbeitung des Interrupt-Requests obliegt dem Interrupt-Handler des jeweiligen Betriebssystems.

Unter Linux besteht kein signifikanter Unterschied zwischen synchronen und asynchronen Interrupts. Geladene Treiber registrieren ihre privaten Interrupt-Service-Routinen beim zentralen Interrupt-Handler. Tritt ein synchroner oder asynchroner Interrupt auf, wird in den Kernel-Modus gewechselt, sofern sich der den Interrupt-Request bearbeitende Prozessor nicht schon im Kernel-Modus befindet, und der zentrale Interrupt-Handler ausgeführt. Dieser ruft nacheinander alle zum aktuellen Interrupt-Vektor passenden Service-Routinen der Treiber auf. Sind alle Interrupt-Service-Routinen erfolgreich abgearbeitet, wird wieder zum zuvor laufenden Prozess umgeschaltet.

Einen Interrupt-Request, der von keiner ausgeführten Service-Routine behandelt wurde, setzt der zentrale Interrupt-Handler zunächst nur zurück. Tritt ein unbehandelter Interrupt-Request mehrmals nacheinander auf, wird der Interrupt durch das Maskieren des Interrupt-Vektors deaktiviert[2].

1.3.2 Klasifizierung von Interrupts im Linux-Kernel

Die von den Gerätetreibern beim zentralen Interrupt-Handler registrierten Interrupt-Service-Routinen können in zwei Klassen unterteilt werden:

Kurze Interrupts: Geeignet für sehr kurze Interrupt-Service-Routinen. Alle Interrupts des aktuellen Prozessors sind während der Abarbeitung blockiert.

Lange Interrupts: Interrupts, die nicht von dem aktuellen Gerät stammen, sind zugelassen und dürfen bei höherer Priorität die aktuelle Behandlung unterbrechen. Dies ist der bervorzugte Fall in der Interrupt-Behandlung.

Bei der Abarbeitung der registrierten Interrupt-Service-Routinen unterscheidet der Linux-Kernel drei Prioritätsklassen[6, Seite 593]:

Critical: Kritische Teile der Interrupt-Service-Routine müssen sofort behandelt werden und dürfen nicht durch andere Interrupts unterbrochen werden.

Non-critical: Unkritische Interrupt-Sektionen müssen zeitnah behandelt werden, können jedoch von Interrupt-Requests mit einer höheren Priorität unterbrochen werden.

Non-critical deferrable: Aufschiebbare, unkritische Abschnitte besitzen die geringste Priorität und können zu einem späteren Zeitpunkt außerhalb des gerade laufenden Interrupt-Handlers abgearbeitet werden. Zur Implementierung unkritscher Abschnitte einer Interrupt-Routine, die zu einem späteren Zeitpunkt ausgeführt werden können, stellt der Linux-Kernel Tasklets zur Verfügung.

1.3.3 Behandlung von Interrupt-Requests

Die Behandlung eines Interrupt-Requests beginnt im zentralen Interrupt-Handler. Hier werden nacheinander alle für den aktuellen Interrupt-Vektor registrierten Interrupt-Routinen der Treiber aufgerufen. Die Interrupt-Routine des Treibers besteht aus zwei logischen Teilen, die nacheinander ausgeführt werden:

[2] Vorausgesetzt der Interrupt ist maskierbar, andernfalls wird eine Kernel-Panik ausgelöst.

1. Interrupt-Handler (Detektion, Selektion und Abfrage in der Reihenfolge der Gerätehirarchie)

2. Interrupt-Service-Routine (Behandlung des Interrupts)

In vielen Gerätetreibern werden die beiden Teile der Interrupt-Routine in einer Funktion zusammengefasst. Die erste Aufgabe des treiberinternen Interrupt-Handlers besteht darin, sicherzustellen, dass der Interrupt-Request auch wirklich von einem zum Treiber zugehörigen Gerät ausgelöst wurde. Dies ist insbesondere wichtig, wenn sich mehrere Geräte einen gemeinsamen Interrupt bzw. Interrupt-Vektor teilen. Wurde der Interrupt-Request von einem anderen Gerät ausgelöst, muss der treiberinterne Interrupt-Handler die Verarbeitung abbrechen und den Wert `IRQ_NONE` an den zentralen Interrupt-Handler des Kernels zurückliefern, damit der Interrupt-Request von einem anderen Gerätetreiber verarbeitet werden kann.

Bei einfach strukturierten Geräten mit einer wenig komplexen Interrupt-Verarbeitung kann oft auf das Auslesen der Interruptursache verzichtet und direkt mit der Interrupt-Behandlung begonnen werden. Befinden sich wie bei der CineCard mehrere getrennte Subsysteme auf dem Gerät, muss jedoch vor der Behandlung die Interrupt-Quelle innerhalb des Gerätes festgestellt werden. Nach der Ermittlung aller zur Behandlung des Interrupt-Requests notwendigen Parameter, kann die eigentliche Interrupt-Service-Routine abgearbeitet werden.

Kapitel 2

Problemanalyse

2.1 Anforderungen an den Funktionsumfang

Verglichen mit anderen vom Linux-Kernel unterstützten Gerätetypen erfordert die HHI CineCard-Familie eine sehr komplexe Behandlung von Interrupt-Requests. Die nötige hohe Effizienz bei gleichzeitig gewahrter Flexibilität erfordert es, einen geeigneten Kompromiss zwischen zwei konträren Zielen zu finden:

Komplexe Interruptbehandlung: Die zu realisierende Lösung muss mindestens die im folgenden Abschnitt beschriebene Basisfunktionalität erfüllen und darüber hinaus flexibel auf veränderte Gegebenheiten reagieren können. Die Interrupt-Behandlung ist dabei abhängig von dem verwendeten Hardware-Typ, der Hardware-Version und den Parametern des wiedergegebenen Datenstromes.

Schnellstmögliche Abarbeitung: Durch die bei der Wiedergabe von hoch aufgelöstem Videomaterial üblichen Datenraten ist eine zeitnahe Behandlung von Dekoderfehlern unabdingbar. Eine hohe Interrupt-Latenz führt schnell zu sichtbaren Fehlern bei der Wiedergabe. Eine zu langsame Behandlung des Interrupt-Requests dagegen könnte wieder zu Dekoderfehlern und damit zu neuen Interrupt-Requests führen, wenn in der Zwischenzeit das Transportstrom-FIFO[1] aufgrund eines anderen nicht behandelten Interrupt-Requests leer läuft. Eine zu lange Verweilzeit im Interrupt-Kontext wirkt sich zusätzlich negativ auf den Datendurchsatz des Systems aus und beeinträchtigt die Interaktivität des Host-Systems.

[1] Ein Zwischenspeicher für die Videodaten auf der PCI-Karte.

2.1.1 Basisfunktionalität

Die im Folgenden beschriebenen Anforderungen müssen von einer möglichen Problemlösung vollständig erfüllt werden. Kann die Basisfunktionalität nicht gewährleistet werden, ist eine weitere Evaluierung des Lösungsansatzes nicht sinnvoll. Die Kriterien der Basisfunktionalität werden somit als Voraussetzung für eine mögliche Problemlösung angesehen und gehen deshalb nicht in die spätere Bewertung ein.

Flexibilität

Eines der wichtigsten Merkmale der zu entwickelnden Interrupt-Behandlung der CineCard ist die Flexibilität. Viele Faktoren, zum Beispiel die Parameter des Videodatenstromes und die Position der CineCard in der Projektionsmatrix, beeinflussen die Interrupt-Behandlung und machen zum Teil sehr verschiedene Behandlungen derselben Interrupts nötig. Ein gutes Konzept zur Interrupt-Behandlung muss flexibel genug sein, um auf alle Interrupt-Situationen der CineCards reagieren zu können.

Die Behandlungs-Routinen von nicht datentransferbasierten Interrupts müssen zusätzlich vom Anwender manuell auslösbar sein, um das Verhalten des Systems auf die Interrupt-Behandlung testen zu können.

Sprachelemente

Aufgrund der notwendigen Flexibilität der umzusetzenden Interrupt-Behandlung kommt eine rein linear verlaufende, also nur aus schreibenden Registerzugriffen bestehende, Interrupt-Service-Routine nicht in Betracht. Die Notwendigkeit auf veränderte Statusregister und veränderliche Parameter reagieren zu können, setzt für die Interrupt-Behandlung mindestens die folgenden Elemente einer Programmiersprache vorraus:

Iteration: Das Zurücksetzen und das Re-Initialisieren einzelner Subsysteme der CineCard erfordert mehrfaches Auslesen einzelner Status-Register und wiederholtes Beschreiben von Konfigurationsregistern.

Verzweigung: Teile der Interrupt-Behandlungsroutine müssen in Abhängigkeit von Statusregistern abgearbeitet werden oder können gegebenenfalls übersprungen werden.

Variablen: Die Inhalte einiger Statusregister werden durch das Auslesen zurückgesetzt. Variablen ermöglichen das mehrfache Auswerten solcher Registerinhalte.

Persistente Variablen: Je nach Wiedergabesituation kann es erforderlich sein, den Status eines Registers beim letzten Auftreten eines Interrupts auszuwerten (z.B. zur Überwachung von Statusänderungen).

Registerschutz: Einige Komponenten der CineCard, wie zum Beispiel das zur Wiedergabe des Transportstromes verwendete FIFO, werden durch eine Folge von Schreibzugriffen auf das gleiche Konfigurationsregister initialisiert. Während der Initialisierung darf keine gleichzeitig laufende Service-Routine auf die zu initialisierende Komponente zugreifen.

Prioritäten: Interrupt-Behandlungen müssen stets nach Priorität geordnet ausgeführt werden. Die Reihenfolge mit der die Statusregister zum feststellen der Interrupt-Ursache ausgelesen werden müssen, kann jedoch von der späteten Ausführungsreihenfolge der Interrupt-Behandlungen abweichen.

2.2 Ausgangsbasis – Hardware

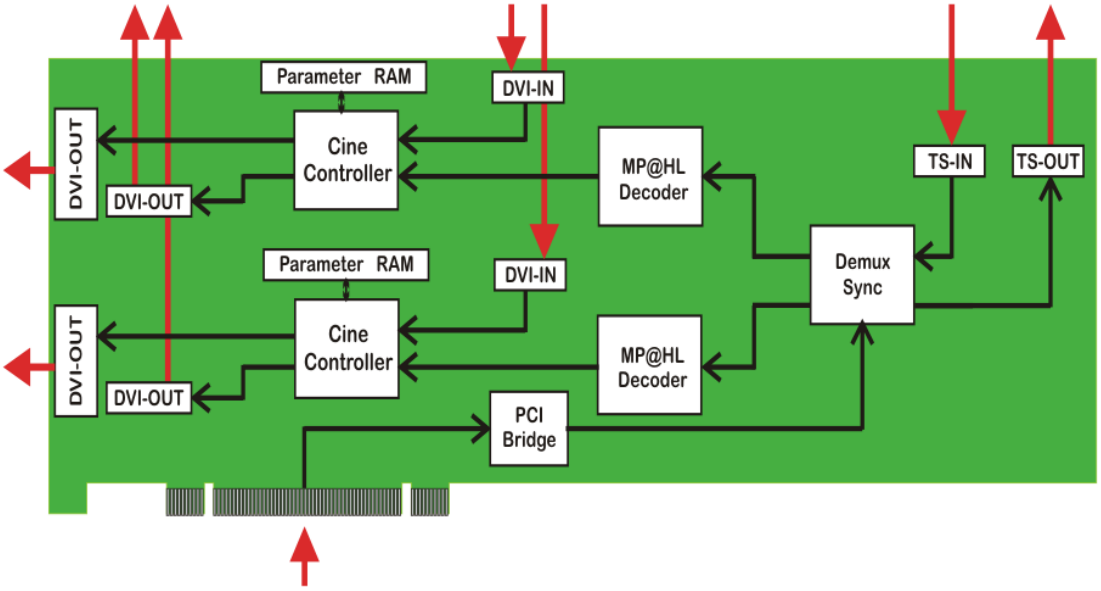
2.2.1 Die CineCard aus der Sicht des Treibers

Alle bisherigen PCI-Karten der HHI CineCard-Familie basieren auf der PCI-Brücke PLX 9054. Neben zwei getrennten DMA-Kanälen bietet diese PCI-Brücke auch die Möglichkeit, den Adressbereich des lokalen Busses der CineCard in den Hauptspeicher des Host-Systems einzublenden.

Die Kommunikation mit der PCI-Brücke findet ebenfalls über einen in den Hauptspeicher des Host-Systems eingeblendeten Speicherbereich statt. In diesem Konfigurationsbereich befinden sich zahlreiche Status- und Konfigurationsregister. Diese Register werden hauptsächlich zur Konfiguration des lokalen Busses und zum Auslösen von DMA-Transfers von und zur CineCard genutzt. Sie ermöglichen zusätzlich die Abfrage der Hardware-Revision der jeweiligen CineCard, mit der die verschiedenen Ausstattungsmerkmale der Karten unterschieden werden. Der aktuelle Zustand der Hardware lässt sich durch zahlreiche Statusregister auslesen und setzen. Der Adressbereich des lokalen Busses ist in mehrere Bereiche unterteilt, die die verschiedenen Subsysteme der CineCard repräsentieren.

Einige Steuerregister der CineCard sind durch den Aufbau der Hardware bedingt, nur beschreibbar und nicht auslesbar. Sollen während des Betriebes einzelne Bits in einem solchen Register modifiziert werden, muss der Treiber den gesetzten Zustand des Registers speichern. Ein Beispiel für ein solches nicht auslesbares Register ist das für die Interrupt-Verarbeitung sehr wichtige Interrupt-Maskenregister. Mit Hilfe dieses Registers kann der Treiber auswählen, welche Anforderungen von Subsystemen der CineCard zu einem Interrupt-Request führen.

Abbildung 2.1: Komponenten der CineCard und Datenfluss



2.2.2 Arten von Interrupts

Bezüglich ihrer Behandlung lassen sich die Interrupt-Requests der CineCard-Familie in drei Kategorien gliedern:

1. Interrupts der PCI-Brücke,
2. Interrupts des Transportstrom-FIFO und
3. Interrupts des MPEG2-Dekoders und anderer datenverarbeitender Subsysteme.

Die Behandlung von Interrupt-Requests der CineCard-Familie ist verglichen mit anderen vom Linux-Kernel unterstützten Gerätetypen sehr aufwändig. Der überwiegende Teil der von der CineCard ausgelösten Interrupt-Requests gehört zu den kritischen Interrupts. Bedingt durch die vielen zur Re-Initialisierung des MPEG2-Dekoders notwendigen Registerzugriffe, benötigt die Interrupt-Behandlung von Transportstromfehlern des HiPEG+ sehr lange Zeit. Der zeitkritische Aspekt stellt eine der größten Herausforderungen bei der umzusetzenden Interrupt-Behandlung dar, denn einige der Interrupt-Requests müssen innerhalb eines sehr engen Zeitfensters behandelt werden. Aus diesem Grund ist es besonders wichtig, die Quelle des Interrupt-Requests schnell zu bestimmen. Für das Bestimmen der Interrupt-

Tabelle 2.1: Speicherbereiche des lokalen Busses der CineCard

Offset-Adresse	**verbundenes Subsystem**
0x2100000	2. HiPEG+
0x3100000	1. HiPEG+
0x4000000	CineCard-Controller 2
0x4800000	CineCard-Controller 1
0x4D00000	De-Multiplexer
0x6000000	Transportstrom-FIFO

Quelle ist das PLX-IRQ-Register von besonderer Bedeutung. Das Register gibt Auskunft darüber, ob ein Interrupt von der PCI-Brücke selbst oder von einem Subsystem ausgelöst wurde, das mit dem lokalen Bus der CineCard verbunden ist.

Die DMA-Interrupts der PCI-Brücke werden für die Synchronisation des Datentransfers von und zur CineCard verwendet. Ein typisches Beispiel für eine solche Interrupt-Anforderung, ist das *DMA_DONE* Signal. Es wird nach dem erfolgreichen Abschluss eines DMA-Transfers versandt und meldet die Verfügbarkeit der DMA-Einheit der PLX-Brücke für weitere DMA-Transfers. Diese Art von Interrupt-Anforderungen ist relativ einfach zu behandeln.

Eng damit verbunden und ebenfalls einfach zu behandeln sind die Interrupt-Requests des zur Wiedergabe verwendeten Transportstrom-FIFOs, die zur Steuerung des Datenflusses von und zur CineCard verwendet werden. Nähert sich der Füllstand des FIFOs einem kritischen Niveau, werden über einen Interrupt-Request neue Daten angefordert. Fallen mehrere Ursachen für das Auslösen eines Interrupt-Requests zusammen, ist die Reihenfolge der Behandlung der Interrupts von entscheidener Bedeutung. Beispielsweise muss die Verfügbarkeit der DMA-Einheit der PLX-PCI-Brücke vor der Auswertung eines Interrupt-Requests des Transportstrom-FIFOs geprüft werden, um sofort wieder Daten verschicken zu können.

Die Behandlung der Interrupt-Anforderungen des Transportstrom-FIFOs und der PLX-PCI-Brücke wurden im Rahmen der Studienarbeit zur CineCard[1] umgesetzt. Die Interrupts des MPEG2-Dekoders und anderer datenverarbeitender Subsysteme auf der CineCard benötigen hingegen eine wesentlich komplexere Interrupt-Behandlung und bilden den Gegenstand dieser Diplomarbeit. Im Folgenden werden sie vereinfacht als Interrupt-Request bezeichnet.

2.3 Ausgangsbasis – Software

2.3.1 Bestehendes Anwendungsumfeld

Die Betriebssoftware der CineCard wurde in Hinblick auf eine maximale Flexibilität und Modularität entworfen. Sie besteht aus drei Kernkomponenten:

1. Gerätetreiber (CineCore, cinedriver, hyperdriver)
2. Management-Applikationen (Cineboxd, Skriptaufbereitung)
3. Benutzeroberflächen (rsend, rcontrol3, CineGUI, cinestatus)

Der Gerätetreiber kapselt die unterschiedlichen Hardware-Typen wie zum Beispiel die HyperCard II und die CineCard und stellt deren Funktionalität über eine einheitliche Kernelschnittstelle zur Verfügung. Die Kernelschnittstelle wurde mit Hilfe von so genannten I/O-Controls implementiert und ermöglicht den parallelen Zugriff mehrerer Konfigurationswerkzeuge.

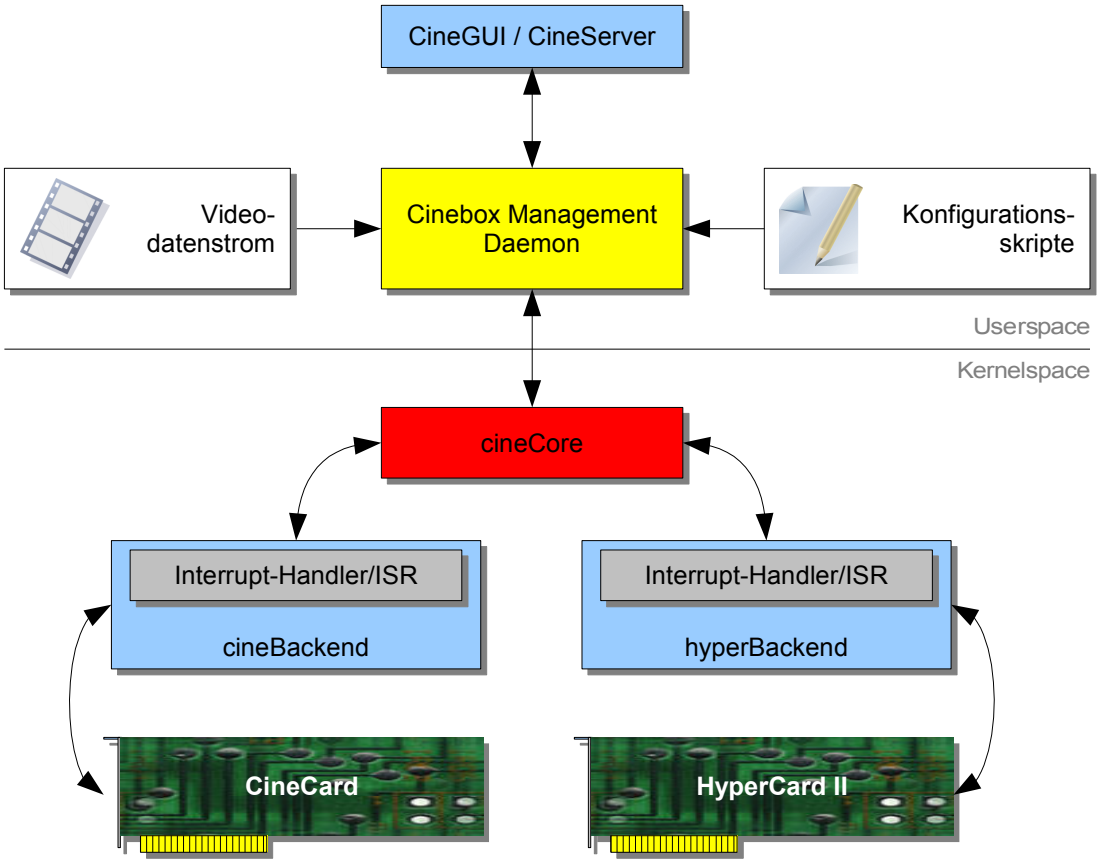
Die Aufnahme und Wiedergabe von Videodatenströmen findet über Lese- und Schreibzugriffe auf die vom Gerätetreiber bereitgestellten Gerätedateien statt. Dabei wird sichergestellt, dass jeweils nur ein Datenstom pro PCI-Karte abgespielt werden kann. Der aus dem CineCore und den Backends für die unterschiedlichen Gerätetypen bestehende Gerätetreiber wurde ausführlich in der Studienarbeit zur CineCard[1] besprochen, so dass hier nur auf die für die Interrupt-Behandlung notwendigen Aspekte des Gerätertreibers eingegangen wird.

Das Herzstück der CineCard-Software ist der CineBox-Management-Daemon[2], kurz Cineboxd. Er stellt eine zentrale Vermittlungsinstanz zwischen dem Gerätetreiber und den verschiedenen Benutzeroberflächen dar. Über den Cineboxd werden alle CineCards eines Host-Systems verwaltet und konfiguriert. Die verschiedenen Benutzeroberflächen kommunizieren mit dem Cineboxd über ein sehr einfaches, menschenlesbares Netzwerkprotokoll[2], so dass die Konfiguration auch über ein Telnet-Programm erfolgen kann.

Die Konfiguration der CineCards findet über zahlreiche Skripte statt. Für das Einlesen der Skripte und deren Interpretation wird der gleiche Parser wie für die Auswertung der über das Netzwerkprotokoll gesendeten Befehle verwendet. Damit lassen sich Konsistenzprobleme zwischen der Netzwerkschnittstelle und dem Skript-Interpreter vermeiden.

[2] Disc and execution monitor

Abbildung 2.2: Struktur der CineCard-Software



Die verwendete Befehlsstruktur wurde in Hinblick auf eine einfache Verwendbarkeit des Netzwerk-Protokolls in den auf Skriptsprachen basierenden Benutzeroberflächen entworfen. So konnte mit Hilfe der Skriptsprache Ruby in nur wenigen Zeilen Quellcode ein sehr flexibler Debugger für die verwendeten Konfigurationsskripte implementiert werden.

Wie in Abschnitt 1.1 beschrieben, kann jedem Videoausgang einer CineCard eine bestimmte Position in der Projektionsmatrix zugeordnet werden. In Abhängigkeit von der Position der CineCard in der Projektionsmatrix, werden durch den Cineboxd die zum Datenstrom zugehörigen Konfigurationsskripte ausgeführt. Nach erfolgreich abgeschlossener Konfiguration wird mit der Wiedergabe des Datenstromes begonnen.

Dem Benutzer stehen mehrere Benutzeroberflächen zur Verfügung, die sich stark in den Bedienkonzepten und den gebotenen Freiheitsgraden unterscheiden:

rsend Das rsend-Kommando ist eine einfache CLI-Anwendung[3]. Sie ist für die Verwendung in Shell-Skripten und zur verzögerten Ausführung von Befehlen gedacht und sendet je Aufruf eine Anweisung an einen Cineboxd.

rcontrol3 Die Bedienung von rcontrol3 orientiert sich stark an der Bourne-Again-Shell (kurz BASH) und bietet dem Benutzer den größten Freiheitsgrad. Mit rcontrol3 kann jeweils ein Cineboxd angesprochen werden.

cineGUI Für den produktiven Einsatz am Projektionsstandort ist die CineGUI vorgesehen. Mit ihr ist es möglich eine beliebige Anzahl von Cineboxen zu verwalten. Die Funktionalität der CineGUI ist auf die üblichen Möglichkeiten eines Wiedergabegerätes beschränkt, bietet jedoch zusätzlich umfangreiche Möglichkeiten zur Verwaltung des Projektorenverbundes.

cineStatus Die cineStatus-Applikation dient zur Überwachung der Wiedergabepuffer der Kerneltreiber einer oder mehrerer CineBoxen und ermöglicht eine detaillierte Leistungsmessung unter Wiedergabebedingungen.

2.4 Analyse-Kriterien

Um die Zweckmäßigkeit einer der im Folgenden vorgestellten Lösungen beurteilen zu können, müssen zunächst geeignete Kriterien zu deren Bewertung gefunden werden. Die Auswahl guter Analysekriterien ist keine triviale Aufgabe, weil sich die gewünschten Eigenschaften der Software zum Beispiel gegenseitig bedingen oder ausschließen können. Eine durch hohe Komplexität erschwerte Wartbarkeit eines Programmteiles schränkt beispielsweise dessen Erweiterbarkeit ein und wirkt sich negativ auf die Testbarkeit aus.

Die besondere Herausforderung der Auswahl der Analysekriterien besteht darin, möglichst viele Kriterien in geeigneter Form zusammen zu fassen und auf wenige, untersuchbare Aspekte zu reduzieren.

Sowohl die Interrupt-Behandlung als auch das dafür notwendige Software-Umfeld ist prinzipbedingt stark mit der verwendeten Hardware verwoben. Darum ist es wichtig, Aspekte der Software in Bezug auf die benutzte Hardware in die Bewertung eingehen zu lassen. Da es sich, wie schon zuvor beschrieben, bei der Interrupt-Verarbeitung um eine zeitkritische Aufgabe handelt, gehören ebenfalls das Laufzeitverhalten der Software beeinflussende Kriterien mit in die Bewertung. Darüber hinaus ist eine leichte Wart- und Erweiterbarkeit bei allen größeren Softwareprodukten von Bedeutung.

[3] Command line interface

Die für die Bewertung der Lösungsansätze sinnvollen Bewertungskriterien lassen sich so grob in die folgenden drei Kategorien untergliedern:

Hardware: Eigenschaften, die im direkten Zusammenhang mit der Hardware und deren Ansteuerung stehen,

Laufzeitverhalten: Faktoren, die die messbare Leistung oder den nutzbaren Funktionsumfang beeinflussen und

Implementierung: Aspekte, die die spätere Wartung und Erweiterung der Software begünstigen oder erschweren.

Um verschiedene Lösungsansätze bewerten zu können, müssen die gefundenen Bewertungskriterien in ihrer Bedeutung für die Problemlösung eingeordnet werden. Viele vorteilhafte Eigenschaften bei der Bedienung des Softwaresystems können zum Beispiel einen entscheidenden Nachteil im Laufzeitverhalten nicht ausgleichen. Dass eine komfortable Bedienung fehlende Funktionalität nicht ausgleichen kann, haben schon zahlreiche andere Softwareprodukte bewiesen. In den folgenden Abschnitten werden die wichtigsten Bewertungskriterien besprochen und ihre Wichtung erarbeitet.

2.4.1 Hardware

Die Entwicklung der CineCard-Familie ist noch lange nicht abgeschlossen. Es ist durchaus möglich, dass in einer späteren Version der CineCard zentrale Komponenten ausgetauscht werden. Die bisher nur aus Prototypen bestehende CineCard-Familie bildet eine ideale Plattform für viele verschiedene Einsatzgebiete. Die Weiterentwicklung der CineCard wird jedoch maßgeblich durch die Anforderungen des Marktes bestimmen. Es ist denkbar, dass in Zukunft der MPEG2-Dekoder durch eine andere Dekoder-Lösung ersetzt oder die CineCard mit einer PCI-Express-Schnittstelle ausgestattet wird.

Der Gerätetreiber der CineCard-Familie wurde von Anfang an im Hinblick auf eine hohe Hardware-Unabhängigkeit entwickelt. Die Interrupt-Behandlung sollte dem Rechnung tragen und sich ebenfalls durch eine geringe Hardware-Abhängigkeit auszeichnen. Dies betrifft ebenfalls die Verwendung der CineCard in anderen Rechnerarchitekturen, wie zum Beispiel PowerPC oder Sparc64, die durch die Art des Interrupt-Behandlungskonzeptes nicht eingeschränkt werden darf.

Die Entwicklung eines Hardware-Prototypen ist sehr aufwändig. Sollte sich im Rahmen dieser Diplomarbeit herausstellen, dass durch eine Änderung an der bestehenden Hardware, die Interrupt-Behandlung deutlich besser als mit der aktuellen Hardware implementiert werden kann, würde auch eine Hardwareänderung in Betracht kommen. Dabei sind jedoch folgende Punkte zu bedenken:

1. Strukturielle Veränderungen an einer vorhandenen Hardware sind generell sehr aufwändig. Eine Änderung des Platinen-Layouts würde zum Beispiel alle PCI-Kartentypen der CineCard-Familie betreffen und einen großen zeitlichen und technischen Aufwand nach sich ziehen.

2. Der Inbetriebnahme einer Hardware-Komponente gehen zahlreiche Entwurfsschritte und Simulationen voraus. Nach Erstellung eines Prototypen stehen umfangreiche Funktions- und Belastungstests an.

3. Die Hardwareänderung könnte zusätzlich eine Anpassung der Betriebssoftware bedingen.

Eine Umstrukturierung der bestehenden Hardware zugunsten eines optimalen Interrupt-Behandlungskonzeptes ist prinizipiell nicht ausgeschlossen, führt jedoch wegen des damit verbundenen zeitlichen Aufwandes zu einer deutlichen Abwertung des betrachteten Lösungsansatzes.

Da auf einer CineCard mehrere MPEG2-Dekoder verbaut sein können, muss auf eine mögliche Synchronisation der unterschiedlichen Interrupt-Behandlungen geachtet werden.

2.4.2 Laufzeitverhalten

Eine stabile Wiedergabe von Datenströmen ist für den praktischen Einsatz von entscheidener Bedeutung. Im Fehlerfall dürfen keine großen Verzögerungen bis zum Beheben der Störsituation auftreten. Wichtige Kenngrößen für die Wiedergabesicherheit sind die Interrupt-Latenz[4] und die Geschwindigkeit der Behandlung von aufgetretenen Interrupts.

Die Behandlung eines aufgetretenen Interrupts ist eine sehr kritische Phase. Dauert die Behandlung zu lange, können während der Initialisierung eines Subsystems neue Interrupt-Requests auftreten. Die Zeit der Behandlung der aktuellen Interrupt-Ursache führt zur Erhöhung der Interrupt-Latenz des folgenden Interrupt-Requests.

Im ungünstigsten Fall ist das den Interrupt behandelnde System mit der Bearbeitung der Interrupt-Service-Routine vollständig ausgelastet. Ist das Host-System für die Behandlung der Interrupt-Requests der MPEG2-Dekoder zuständig, können bei einer vollständigen Auslastung des Systems keine neuen Daten in das zur Wiedergabe verwendete Transportstrom-FIFO der CineCard geladen werden. Es droht ein Abriss des Datenstromes.

[4]Der Begriff Interrupt-Latenz beschreibt die Zeit, die das System von der Feststellung eines Interrupts bis zum Beginn der Behandlung benötigt.

Der Anteil der zusätzlich zur reinen Behandlung der Interrupt-Ursache notwendigen Verwaltung, der so genannte Behandlungs-Overhead, ist ein wichtiges Maß für die Einschätzung der Effizienz der Interrupt-Behandlung.

Eine zu hohe Interrupt-Latenz und eine zu langsame Interrupt-Behandlung wirken sich zusätzlich in mehrfacher Hinsicht negativ auf das Host-System aus. Befindet sich das Host-System im Interrupt-Kontext, steht es nicht für andere Aufgaben zur Verfügung. Eine lange Verweilzeit in einer Interrupt-Routine macht sich in einer schlechten Interaktivität des Host-Systems bemerkbar. Ist die Aufruffrequenz der Interrupt-Routine hoch, befindet sich das Host-System anteilig sehr lange im Interrupt-Kontext, was den Gesamtdurchsatz des Systems stark reduziert. Ein gutes Konzept zur Interrupt-Behandlung sollte sich deshalb durch eine geringe Interrupt-Latenz und einen geringen Behandlungs-Overhead auszeichnen.

Neben der Geschwindigkeit sind für die Beurteilung der Leistung eines Gesamtsystems auch dessen Funktionsumfang und deren Handhabung wichtig. Die in der Basisfunktionalität geforderten Kriterien der Flexibilität beziehen sich auf die prinzipiellen Möglichkeiten der Interrupt-Behandlung. Im Rahmen der Bewertung der Interrupt-Lösungen ist zusätzlich interessant, in welcher Form sich das Verhalten der Interrupt-Behandlung zur Laufzeit verändern und anpassen läßt, um zum Beispiel den Test neuer Hardwarekomponenten zu vereinfachen.

Für die effektive Behandlung der Interrupt-Requests der MPEG2-Dekoder und anderer Subsysteme der CineCards sind detaillierte Informationen über die Ursache und die Behandlung des Interrupts erforderlich. Zum Beispiel werden zur Re-Initialisierung des MPEG2-Dekoders viele unterschiedliche Parameter benötigt, die vor der Behandlung eines Interrupts der Interrupt-Service-Routine zur Verfügung gestellt werden müssen. Für die Bewertung des Lösungsansatzes entscheidend ist der Umfang und die Komplexität der Aufbereitung der Parameter zur Interrupt-Behandlung und wie gut die Parameteraufbereitung von den Nutzern der Software wartbar ist.

Komplexe Systeme und Systeme mit vielen Sonderfällen und Freiheitsgraden müssen vor dem produktiven Einsatz sehr sorgfältig getestet werden. Im Interrupt-Kontext in Verbindung mit Hardware-Komponenten ist die klassische Strategie der Fehlersuche mit Hilfe eines Debuggers nicht oder nur sehr eingeschränkt einsetzbar. Seiteneffekte der Gleichzeitigkeit sind über diese Methode nicht zu erfassen. Der Test einer neu umgesetzten Strategie[5] zur Interrupt-Behandlung oder deren veränderter Parameter hat jedoch einen entscheidenden Einfluss auf die Qualität des Endproduktes. Die Testbarkeit ist damit das wichtigste Bewertungskriterium in der Kategorie Laufzeitverhalten.

[5] Als Strategie zur Interrupt-Behandlung wird in diesem Zusammenhang eine konkrete Realisierung eines Interrupt-Behandlungskonzeptes bezeichnet. Die Wahl der Strategie zur Interrupt-Behandlung kann z.B. durch Parametrisierung der Interrupt-Service-Routine verändert werden.

2.4.3 Implementierung

Wie zuvor beschrieben, existiert neben dem Gerätetreiber bereits eine umfangreiche Sammlung von Userspace-Applikationen. Ein gutes Konzept für die Interrupt-Behandlung der CineCard-Familie sollte sich nahtlos in das bestehende Anwendungsumfeld integrieren lassen. Für die Entwicklung eines Bewertungsmaßstabes interessante Fragen sind dabei zum Beispiel:

- Wie integriert sich die Lösung in die bestehende Kernel-API?
- Welche Erweiterungen des Cineboxd sind für die Interrupt-Behandlung notwendig und mit welchem Aufwand sind sie verbunden?
- Wie werden die zur Laufzeit notwendigen Parameter der Interrupt-Behandlung ermittelt (Datenstromparameter)?
- Wie werden externe Informationen (wie zum Beispiel die Position der CineCard in der Projektionsmatrix) an die Interrupt-Routine übermittelt?
- Sind zur Übermittlung oder für die Aufbereitung der Parameter noch umfangreiche Programmteile zu entwickeln oder anzupassen?

Der Umfang der im Vorfeld notwendigen Änderungen und Erweitungen des Gerätetreibers und der Applikationen bildet ein wichtiges Bewertungskriterium. Für den Einsatz der Software ist jedoch die spätere Wartbarkeit von weitaus höherer Bedeutung. Die Inbetriebnahme neuer Subsysteme der CineCards oder andere Veränderungen an der Hardware der CineCard können sich auch auf die Strategie der Interrupt-Behandung auswirken, sollten aber möglichst keine Änderung des Gerätetreibers nach sich ziehen. Strukturielle Änderungen der Software und seiner Schnittstellen sollten durch einen möglichst flexiblen und modularen Aufbau vermieden werden. Eine auf Erweiterbarkeit ausgelegte Interrupt-Behandlung zeichnet sich vor allem durch eine gute Änderbarkeit und eine geringe Fehleranfälligkeit aus. Der Wartbarkeit wird somit in der Kategorie Implementierung die größte Bedeutung beigemessen.

Eng mit der Wartbarkeit einer Software verbunden ist deren Komplexität. Im Kernel-Kontext können sich Fehler fatal auswirken und sind schwer zu finden. Komplexe Verarbeitungsschritte sollten deshalb möglichst vermieden und wenn nötig nur im Userspace ausgeführt werden. Im Rahmen der Implementierung wird die Komplexität der Software innerhalb des Kernels gesondert betrachtet, weil Fehler im Kernel- und Interrupt-Kontext besonders schwere Folgen haben können. Aspekte der Gleichzeitigkeit, der zeitkritischen Abarbeitung und der schlechten Testbarkeit lassen die Fehleranfälligkeit komplexer Funktionen überproportional ansteigen. Weiterführende Informationen zu den Risiken hoher Komplexität

finden sich in dem Artikel *Fehlerursache Komplexität*[5] aus der Zeitschrift Design&Elektronik.

Eine optimale Strategie zur Interrupt-Behandlung besteht nur aus einfachen, gut zu testenden Funktionen und vermeidet Komplexität in allen kritischen Bereichen.

Bei der Implementierung ist weiterhin zu beachten, dass die Software später auch im produktiven Bereich bei Endkunden eingesetzt werden soll. Werden dabei preiswertere, festplattenloses Wiedergabesysteme eingesetzt, die das Betriebssystem auf einem Flash-Speichermedium enthalten, kann wegen der begrenzten lokalen Speicherkapazität nur eine eingeschränkte Anzahl von Programmpaketen installiert werden. Zusätzlich stellt eine umfangreiche Software-Ausstattung auf dem Kundensystem auch ein erhöhtes Sicherheitsrisiko dar. Deshalb sollten bei der Implementierung möglichst eine Abhängigkeiten von externer Software vermieden und auf eine kompakte Programmierung geachtet werden.

2.4.4 Zusammenfassung in einer Bewertungsmetrik

Die folgende Tabelle fasst die Bewertungskriterien zusammen und spiegelt die Wichtung der einzelnen Punkte wieder. Die gefundene Wichtung der einzelnen Bewertungskriterien stellt einen guten Kompromiss zwischen der Forderung nach flexibler Funktionalität und technischer Umsetzbarkeit dar. Wie zuvor schon erwähnt, befindet sich die CineCard noch in der Entwicklungsphase und wird zur Zeit am häufigsten im Forschungsumfeld eingesetzt. Für diesen Einsatzzweck günstige Eigenschaften, wie zum Beispiel eine gute Wartbarkeit der Interrupt-Behandlung, führen zu einer deutlichen Aufwertung einer betrachteten Lösung. Die Unabhängigkeit von einschränkenden Faktoren führt unmittelbar zu kürzeren Entwicklungszyklen und damit zu geringeren Kosten bei der Umsetzung neuer Technologien.

Zu einer deutlichen Abwertung führen Eigenschaften, die den späteren Gebrauch des Produktes einschränken, die Weiterentwicklung erschweren oder große potentielle Fehlerquellen mit sich bringen. Aspekte, die die Erst-Implementation des Interrupt-Konzeptes erschweren, führen dagegen nur zu einer leichten Abwertung.

Tabelle 2.2: Übersicht über die Bewertungsmetrik

Bewertungskriterium	**Auswirkung/Beschreibung**	**Wichtung**
Hardware		
Hardware-Änderungen	bedingt hohen zeitlichen und finanzieller Aufwand bei einem Re-Design der CineCards	20%
Hardware-Unabhängigkeit	vermindert späteren Aufwand bei der Portierung auf eine neue Hardware/Hardwareplattform	7%
IRQ-Synchronisation	potentiell schwierig umzusetzen, erfordert eventuell eine Anpassung der Hardware	3%
Laufzeitverhalten		
Testbarkeit	wichtig für Hardware-Entwickler beim Test neuer Behandlungs-Parameter- und Strategien	10%
Laufzeitanpassung	beschleunigt die Inbetriebnahme neuer Hardwarekomponenten	5%
Parameter-Aufbereitung	Komplexität der Umwandlung der Konfigurationsskripte zu einer funktionierenden Interrupt-Behandlung	5%
Interrupt-Latenz	hoher Einfluss auf die Interaktivität des Host-Systemes und die Fehlerstabilität bei der Videowiedergabe	5%
Behandlungs-Overhead	Einfluss auf Wiedergabe-Stabilität	3%
Implementierung		
Wartbarkeit	wichtig für Zukunftssicherheit und spätere Erweiterung der der Software	20%
Umfang der Änderungen	Integration in das bestehende Anwendungsumfeld kann sehr aufwändig sein	10%
Infrastruktur/Sicherheit	gestellte Anforderungen an das zur Wiedergabe verwendete Host-System	7%
(In-Kernel)-Komplexität	größte Fehlerquelle	5%

Kapitel 3

Mögliche Problemlösungen

Um geeignete Lösungen für die Interrupt-Behandlung der CineCard-Familie zu finden, lohnt es sich, bereits existierende Konzepte ähnlicher Anwendungsgebiete auf deren Tauglichkeit und Ausbaufähigkeit zu überprüfen. Sowohl im Umfeld von Linux als auch unter anderen Betriebssystemen sind zahlreiche, zum Teil sehr interessante Herangehensweisen an die Interrupt-Behandlung zu finden. Leider sind viele (der interessanten) Ansätze auf einen speziellen Hardwaretyp zugeschnitten und nicht ohne weiteres auf die HHI CineCard übertragbar. Oft eingesetzte proprietäre Firmware-Konzepte und ausschließlich in binärer Form vorliegende Treiber erschweren zusätzlich die Evaluation der Effektivität der verwendeten Techniken.

3.1 Analyse bestehender Konzepte

Um für HHI CineCard verwendbare Interrupt-Behandlungskonzepte finden zu können, ist es zweckmäßig, zunächst Gerätetreiber für ähnliche Hardware-Komponenten zu untersuchen. Der am ehesten mit den CineCards zu vergleichende Gerätetyp ist die Familie der Standard-TV DVB-Dekoderkarten. Die dafür im Linux-Kernel verwendeten Gerätetreiber basieren auf dem von der Berliner Firma Convergence entwickelten Treiber für die Fujitsu-Siemens DVB-PCI Karten. Anfangs wurden ausschließlich diese, auf einem Technotrend-Referenzdesign basierenden DVB-S Karten, unterstützt. Innerhalb der letzten sechs Jahre wurde eine umfangreiche Anzahl von weiteren Gerätetreibern für PCI- und USB-Geräte dieser Geräteklasse hinzugefügt.

Zur Interrupt-Behandlung der meisten PCI-Geräte dieses Typs wird eine statische Implementation mit überschaubarer oder geringer Komplexität verwendet, Interrupts von über den USB-Bus verbundenen DVB-Geräten werden durch ei-

ne proprietäre Firmware auf der externen Hardware selbst behandelt. Da bei der CineCard sehr komplexe Abläufe gesteuert werden müssen, kann nicht auf das vorhandene Interrupt-Behandlungskonzept der DVB-Karten zurückgegriffen werden. Kein anderer Gerätetyp, der über quellenoffene Gerätetreiber verfügt, ist in der Behandlung von Interrupt-Requests direkt mit der CineCard vergleichbar. Deshalb müssen für die HHI CineCard-Familie neue Konzepte entwickelt und evaluiert werden.

Ausgehend vom bestehenden Hard- und Softwareumfeld lassen sich mit Hilfe der Interrupt-Behandlungskonzepte anderer Gerätetreiber vier möglichen Herangehensweisen für die Behandlung von Interrupt-Anforderungen ableiten:

Hardware-gestützte Behandlung: In dem verwendeten MPEG2-Dekoder (HiPEG+) ist eine 32-Bit RISC-Prozessoreinheit integriert, die sich für die Interrupt-Behandlung verwenden lässt. Aus Sicht der Hardware ist dies die naheliegendste Lösung. Sie verspricht relativ geringe Interrupt-Latenzen und entlastet das mit der Datenstromwiedergabe beschäftigte Host-System.

Statische Implementierung: Der klassische Fall der Interrupt-Behandlung wird bei vielen Hardware-Treibern verwendet. Die höhere Geschwindigkeit des Host-Systems gegenüber einem eingebetteten System läßt sich im Interrupt-Kontext gut nutzen. Auch größere Datenmengen bei der Initialisierung können schnell von und zur CineCard transportiert werden.

Binäre, nachladbare Module: Eventuelle Engpässe hinsichtlich der Flexibilität einer statischen Implementierung sind durch ein ladbares, austauschbares Kernel-Modul für die Behandlung von Interrupts vermeidbar. Die speziell auf einen Hardwaretyp und die momentane Wiedergabesituation anpassbare Interrupt-Behandlung verspricht ein sehr günstiges Laufzeitverhalten, geringe Interrupt-Latenzen und hohe Flexibilität.

Skript-Lösung: Vom bestehenden Anwendungsumfeld und der Anforderung der höchstmöglichen Flexibilität aus gesehen, ist eine auf interpretierten Skripten basierende Lösung die zweckmäßigste. Weil ohnehin schon verhandene Konfigurationsskripte mit geringen Änderungen direkt zur Interrupt-Behandlung weiterverwendet werden könnten, würde diese Lösung sich sehr gut in die vorhandene Anwendungssoftware integrieren.

3.2 Hardware-gestützte Interrupt-Behandlung

In dem verwendeten MPEG2-Dekoder MB87J2020[3] (HiPEG+) ist eine 32-Bit RISC Prozessoreinheit vom Typ ARC Tangent A4 integriert. In einigen Settop-

Boxen wird diese Prozessoreinheit zur Anzeige von EPG-Daten, Untertiteln und zur Behandlung von Transportstromfehlern verwendet.

3.2.1 Beschreibung

Der ARC-Prozessor besitzt vollen Zugriff auf die zur Interrupt-Behandlung notwendigen Komponenten des MPEG2-Dekoders (zum Beispiel PID-Filter, Interrupt-Register und Status-Register). Der ARC-Core wird mit 54 MHz getaktet und verfügt somit über genügend Rechenleistung, um auch umfangreiche Initialisierungen durchzuführen.

Der HiPEG+ bietet ein Bus-Interface, über welches mit dem integrierten ARC-Prozessor oder MPEG2-Dekoder kommuniziert werden kann. Um den Zugriff vom Host-System auf den HiPEG+ zu ermöglichen, kann dieses Interface mit dem lokalen Bus der CineCard verbunden werden. Das Bus-Interface des HiPEG+ unterstützt folgende drei Betriebs-Modi[3]:

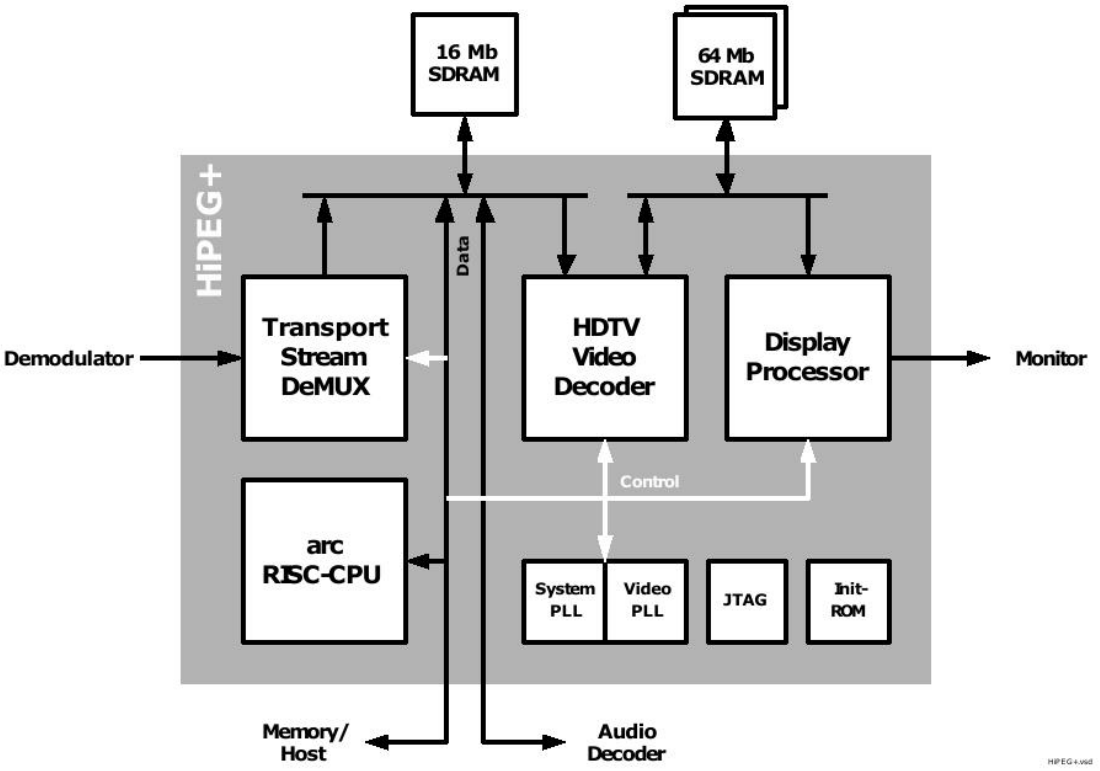
standalone ARC-Modus: Die im HiPEG+ integrierte ARC-CPU übernimmt die Konfiguration und Interrupt-Behandlung des MPEG2-Dekoders. Um vom Host-System auf die ARC-Prozessoreinheit zugreifen zu können, muss diese über ein Brücken-Interface mit dem lokalen Bus der CineCard verbunden werden. Ein Datenaustausch mit dem ARC-Prozessor ist in diesem Fall zum Beispiel durch einen vom ARC und dem lokalen Bus erreichbaren Speicherbereich möglich. Durch Implementierung eines geeigneten Protokolls kann eine Kommunikation zwischen dem ARC-Prozessor und Host-System aufgebaut werden. Über einen solchen Brückenbaustein könnte der ARC-Prozessor ebenfalls auf andere Komponenten und Subsysteme der CineCards zugreifen.

Host-Modus: Ein externer, über die so genannte μP-Schnittstelle angebundener Prozessor übernimmt die Steuerung und Konfiguration des MPEG2-Dekoders. Das Interface unterstützt eine Vielzahl von Prozessorarchitekturen wie z.B. SparcLite, PowerPC und MC68k. Im Host-Modus ist der integrierte ARC-Prozessor deaktiviert. Vom PCI-Bus des Host-Systems kann nur mit Hilfe des angebundenen Prozessors auf den MPEG2-Dekoder zugegriffen werden.

glueless PCI-Bridge Modus: Die Konfiguration des MPEG2-Dekoders erfolgt in diesem Modus ausschließlich über den PCI-Bus des Host-Systems. Sowohl der integrierte ARC-Prozessor, als auch die μP-Schnittstelle sind deaktiviert und stehen zur Interrupt-Behandlung nicht zur Verfügung.

In den aktuellen CineCards wird das Bus-Interface des HiPEG+ im glueless PCI-Bridge Modus betrieben. Um die integrierte ARC-Prozessoreinheit nutzen zu können, müsste der HiPEG+ jedoch im standalone ARC-Modus betrieben werden. Ein direkter Zugriff vom Host-System auf die Register des MPEG2-Dekoders ist im standalone ARC-Modus nicht möglich und alle Zugriffe auf den MPEG2-Dekoder müssen über den ARC-Prozessor durchgeführt werden. Zur Kommunikation mit der Außenwelt stehen dem ARC-Prozessor 16 programmierbare Mehrzweck-Ein- und Ausgabekanäle sowie eine I^2C-Schnittstelle zur Verfügung.

Abbildung 3.1: Architektur des HiPEG+[3]



Im standalone ARC-Modus muss, um auf Interrupt-Requests reagieren zu können, zuvor eine Betriebssoftware in Form einer binäre Firmware in den ARC-Core geladen werden. Diese wird mit Hilfe einer Cross-Compiler-Toolchain erzeugt und kann beispielsweise auf Basis der vorhandenen Konfigurationsskripte erstellt werden.

3.2.2 Analyse bezüglich der Basisfunktionalität

Die Anforderungen an die Basisfunktionalität lassen sich mit Hilfe der in den MPEG2-Dekoder integrierten ARC-Prozessoreinheit vollständig erfüllen. Durch die auf der GCC[1] basierenden Cross-Compiler-Toolchain lassen sich alle erforderlichen Sprachelemente umsetzen.

Die Strategie der Interrupt-Behandlung ist durch den Austausch der binären Firmware änderbar. Parameter der Interrupt-Verarbeitung lassen sich durch eine Kommunikation mit der ARC-Prozessoreinheit modifizieren. Zum Laden der Firmware und zur Kommunikation mit der ARC-Einheit ist eine Erweiterung der bestehenden Kernelschnittstelle notwendig.

Ein manuelles Ausführen der Interrupt-Behandlung in der ARC-Prozessoreinheit ist zum Beispiel durch das Implementieren eines RPC-Protokolls[2] zwischen dem Host-System und ARC-Prozessor möglich.

Bedingt durch den fehlenden direkten Zugriff auf den HiPEG+, ist ein Test der vom ARC-Prozessor ausgeführten Interrupt-Behandlung im Anwendungs-Kontext des Host-Systems unmöglich. Das Prinzip der Interrupt-Behandlung ließe sich höchstens durch eine Parallel-Implementation unter Zuhilfenahme der Konfigurations-Schnittstelle des ARC-Prozessors realisieren.

3.2.3 Evaluation

Hardware

Wie schon zuvor beschrieben, ist für die Nutzung des ARC-Prozessors im HiPEG+ eine Änderung der bestehenden Hardware der CineCard-Familie Voraussetzung. Eine Umstellung vom Host-Modus auf den für diesen Lösungsansatz notwendigen standalone-ARC-Modus betrifft zentrale Teile der Infrastruktur der CineCard.

Neben Elementen, die das Transferieren der Firmware und die Kommunikation zwischen dem ARC-Prozessor und dem Host-System erlauben, muss dem ARC-Prozessor zur Ausführung von Programm-Code ein lokaler Speicher zur Verfügung gestellt werden.

Weil der ARC-Prozessor beim Zurücksetzen des MPEG2-Dekoders ebenfalls zurückgesetzt wird, ist zusätzlich ein nicht flüchtiger Speicher zum Ablegen der Initialisierungsparameter und der Interrupt-Routine selbst notwendig. Eine Realisierung der Interrupt-Behandlung über die ARC-Prozessoreinheit bedingt damit ein vollständiges Re-Design aller PCI-Karten der CineCard-Familie. Dies führt

[1] GNU Compiler Collection
[2] Remote Procedure Call

wegen des hohen zu erwartenden Arbeitsaufwandes zu einer starken Abwertung dieser Problemlösung.

Die Ausführung der Interrupt-Behandlung auf der ARC-CPU bindet die Interrupt-Verarbeitung der CineCard vollständig an den verwendeten MPEG2-Dekoder. Das Konzept der Interrupt-Verarbeitung ist damit nicht mehr auf andere Videodekoder übertragbar. Der Austausch des HiPEG+ gegen eine andere Lösung zur Videodekodierung ist mit diesem Konzept ausgeschlossen.

Ein Vorteil der integrierten Lösung ist, dass die Interrupt-Behandlung nicht mit der des Host-System interferiert und somit sehr geringe Interrupt-Latenzzeiten erreicht werden können. Bei CineCards mit mehr als nur einem MPEG2-Dekoder, können die Interrupt-Requests verschiedener Dekoder vollständig parallel zueinander behandelt werden.

Dieser Vorteil ist aber gleichzeitig ein Nachteil, da eine Synchronisation der Interrupt-Behandlungen mehrerer MPEG2-Dekoder einer CineCard (bzw. mehrerer CineCards untereinander) in diesem Falle nur über zusätzliche Hardware auf der CineCard oder eine aufwendige Kommunikation mit dem auf dem Host-System laufenden Gerätetreiber realisierbar wäre. Auch der Zugriff auf die Ressourcen des Host-Systems oder der Datenaustausch mit anderen CineCards innerhalb eines Host-Systems ist nur mit Unterstützung des Gerätetreibers.

Eine solche Interrupt-Synchronisation ist zum Beispiel nötig, wenn nach einem nicht behebbaren Dekoderfehler auf allen Projektoren einer Projektionsmatrix Schwarzbilder ausgegeben werden sollen.

Laufzeitverhalten

Eine autark funktionierende Interrupt-Behandlung durch die ARC-Prozessoreinheit bietet neben der bereits erwähnten parallelen und vom Host-System unabhängigen Verarbeitung der Interrupt-Anforderungen zusätzlich noch weitere Vorteile. Durch die vom ARC-Prozessor nutzbare Hardware-Unterstützung zur Datenstromanalyse ist eine Anpassung der Behandlungs-Parameter zur Laufzeit in vielen Fällen unnötig. Die bei den auf dem Host-System laufenden Interrupt-Behandlungen sehr aufwändige Analyse des Datenstromes mit Hilfe der Software entfällt und die Parametrisierung der Interrupt-Behandlung kann, auf externe Parameter beschränkt, stark vereinfacht werden. Dadurch wird das Host-System zusätzlich stark entlastet.

Die Umsetzung der Interrupt-Verarbeitung in einem vom Host-System unabhängigen Embedded-System verringert jedoch die Testbarkeit einer neu umgesetzten Strategie zur Interrupt-Behandlung entscheidend. Die Fehlersuche und auch das Untersuchen der Reaktion der Interrupt-Behandlung auf veränderte Behand-

lungsparameter wird durch die nur eingeschränkt mögliche Kommunikation mit dem Embedded-System stark behindert. Eine Ausführung der Behandlungsroutine direkt im Ausführungskontext der Benutzerapplikation ist, wie bereits einleitend erwähnt, prinzipbedingt ausgeschlossen. Das Verhalten der Interrupt-Behandlung lässt sich im Userspace bestenfalls durch eine Parallel-Implementation unter Zuhilfenahme der Konfigurationsschnittstelle des Gerätetreibers nachempfinden. Darüber hinaus besitzen die Ergebnisse eines solchen Tests nur wenig Aussagekraft über das spätere Verhalten der Implementation auf dem Embedded-System.

Implementierung

Eine Implementierung der Betriebs-Software für die ARC-Prozessoreinheit ist wesentlich aufwendiger als eine vergleichbare Software-Implementierung auf dem Host-System. Durch die Beschränkungen des ARC-Prozessors kann kein vollwertiges Betriebssystem in der Firmware genutzt werden. Alle verwendeten Funktionen des eingesetzten Systems müssen erneut implementiert werden. Die Verwendung standardisierter Bibliotheken, zum Beispiel für nicht ganzzahlige Berechnungen und dynamische Speicherverwaltung, ist nur mit erheblichen Anpassungen möglich. Durch ein fehlendes Betriebssystem äußern sich Fehler in der Implementierung, wie Speicherzugriffsfehler und Überläufe des Stapelspeichers nur durch entsprechend auftretende Prozessor-Exceptions des ARC-Prozessors. Für solche Fälle müssen Debug-Möglichkeiten wie Ausgabe oder Speicherung der Registerinhalte des ARC-Prozessors aufwendig selbst implementiert werden. Oft lassen sich solche gravierenden Fehler nur durch umfangreiche Modultests finden.

Durch die mögliche Analyse des Datenstromes innerhalb des Embedded-Systems ist eine an den Datenstrom angepasste Interrupt-Behandlung ohne Kommunikation mit dem Host-System möglich. Damit verbunden ist jedoch auch eine hohe Komplexität der Interrupt-Verarbeitung auf dem Embedded-System und eine überdurchschnittliche Fehlerträchtigkeit der Software (Siehe auch[5]).

Der Einsatz eines Debuggers ist prinzipiell auch im Zusammenhang mit dem Embedded-System möglich, allerdings ist damit auch ein hoher technischer Aufwand verbunden. Gegenüber den Möglichkeiten eines Debuggers für klassische PC-Software sind die Möglichkeiten eines solchen Debuggers für Embedded-Systeme sehr limitiert. Bedingt durch die eingeschränkten Kommunikationsmöglichkeiten mit dem Embedded-System, ist eine Fehlersuche im Interrupt-Kontext des ARC-Prozessors nur mit aufwendig zu implementierender Test-Software möglich.

Änderungen an der Interrupt-Vearbeitung lassen sich nur unter Verwendung der ARC-Prozessoreinheit testen. Die Software muss mit Hilfe eines Cross-Comilers übersetzt und auf das Embedded-System übertragen werden. Die umfangreiche und komplexe Embedded-Programmierung und die eingeschränkte Testbarkeit

wirken sich auch auf die Wartbarkeit des gesamten Softwaresystems aus. Geht eine notwendige Änderung des Verhaltens der Interrupt-Behandlung über die Anpassung von Behandlungsparametern hinaus, muss eine neue Firmware übersetzt und installiert werden.

3.2.4 Zusammenfassung der Bewertung

Die Analyse der Interrupt-Behandlung über die in den HiPEG+ integrierte Prozessoreinheit offenbart große Schwächen in den wichtigsten für die Bewertungsmetrik relevanten Kategorien. Die im Vorfeld notwendige Modifikation der bestehenden Hardware und die spätere Bindung des Konzeptes an den ARC-Prozessor können nicht von den wenigen Vorteilen des Laufzeitverhaltens ausgeglichen werden. Die umfangreiche Anpassung des Gerätetreibers auf die veränderte Hardware und die zu erwartende schlechte Wartbarkeit des Gesamtsystems tragen ebenfalls zum schlechten Gesamtergebnis des hardwarebasierten Lösungsansatzes bei.

Im produktiven Betrieb mit jeweils nur einem MPEG2-Dekoder pro System ist die betrachtete Lösung gut einsetzbar. Der Einsatz in einigen Settop-Boxen und anderen Gerätetypen untermauert die Einsatzfähigkeit dieses Lösungsansatzes. Eine wichtige Voraussetzung für die Verwendung ist ein sich kaum veränderndes Anwendungsgebiet mit wenig veränderlichen Behandlungsparametern.

Für die Anwendung im Forschungsalltag des Heinrich-Hertz-Instituts ist die Interrupt-Verarbeitung mit Hilfe der in den HiPEG+ eingebetteten ARC-Prozessoreinheit jedoch denkbar schlecht geeignet. Die im Vorfeld der Implementation des Interrupt-Behandlungskonzeptes notwendigen Umstrukturierungen der Hardware und die hohe Hardware-Abhängigkeit würden einen nicht zu unterschätzenden zeitlichen und finanziellen Aufwand nach sich ziehen. Die Vorteile der relativ geringen Interrupt-Latenz und der einfachen Parameteraufbereitung können die mangelnde Flexibilität und Wartbarkeit nicht aufwiegen.

Tabelle 3.1: Bewertung der hardware-gestützten Interrupt-Behhandlung

Bewertungs kriterium	**Beschreibung**	**Note und Wichtung**
Kategorie Hardware		
Hardware-Änderungen	umfangreiches Re-Design aller CineCards notwendig	5 (20%)
Hardware-Unabhängigkeit	Lösung ist nicht auf andere Dekoder-Hardware übertragbar, Portierung nötig	5 (7%)
Interrupt-Synchronisation	sehr aufwändig, fehlerträchtig, nur mit Hardware-Unterstützung realisierbar	5 (3%)
Laufzeitverhalten		
Testbarkeit (zur Laufzeit)	durch Embedded-System sehr eingeschränkt kein Einzelschrittmodus möglich	5 (10%)
Laufzeit-anpassung	sehr aufwändig über Kommunikation mit ARC-Prozessor möglich	5 (5%)
Parameter-aufbereitung	durch Hardware oft unnötig, aber sehr aufwändig	3 (5%)
Interrupt-Latenz	unmittelbare Reaktion, vom Host-System unabhängig	1 (5%)
Behandlungs-Overhead	gering	1 (3%)
Implementierung		
Wartbarkeit	schwierig, keine normales Betriebssystem, Implementationstest sehr eingeschränkt	4 (20%)
Umfang der Änderungen	Firmware-Lademechanismus, Anpassung der Kernelschnittstelle, Firmware-Implementation	5 (10%)
Infrastruktur Sicherheit	Cross-Compiler Toolchain, Parameteraufbereitung	5 (7%)
Komplexität (im Kernel)	sehr hoch in der Firmware des Embedded-systems, Fehler schwer zu finden	5 (5%)
Gesamtnote	**der Lösungsansatz ist für den geplanten Einsatzzweck ungeeignet**	**4,4**

3.3 Statische Implementierung

Die überwiegende Anzahl der Gerätetreiber des Linux-Kernels verwendet eine statisch implementierte Interrupt-Routine. Weil die Interrupt-Quelle der meisten Geräte einfach festzustellen ist, oder gar nicht erst bestimmt werden muss, wird oft der treiberinterne Teil des Interrupt-Handlers mit der Interrupt-Service-Routine des Treibers zusammengelegt.

Bei der Wiedergabe von Datenströmen auf der CineCard wird ein Großteil der Interrupt-Requests von der DMA-Einheit der PCI-Brücke und dem Datenstrom-FIFO ausgelöst. Die beispielsweise durch Transportstromfehler verursachten Interrupt-Requests des MPEG2-Dekoders, können oft nur durch das Zurücksetzen des betroffenen HiPEG+ behandelt werden.

Die Hauptaufgabe der Interrupt-Service-Routine beschränkt sich in diesem Fall auf eine erneute Initialisierung des MPEG2-Dekoders. Unter diesen Voraussetzungen erscheint es naheliegend, die bestehende Interrupt-Service-Routine um eine einfache Reinitialisierungs-Routine zu erweitern.

3.3.1 Beschreibung

Die auf dem Host-System ausgeführte Interrupt-Behandlung der CineCard folgt dem auch bei anderen Gerätetypen eingesetzten Schema (siehe Abschnitt 1.3.3). Handelt es sich bei der Interrupt-Quelle um einen der MPEG2-Dekoder, lässt sich die Interrupt-Ursache meist nur durch einen Reset des Dekoders beheben.

Nach dem Zurücksetzen muss dieser wieder neu mit den aktuellen Wiedergabeparametern des abgespielten Datenstromes initialisiert werden. Je nach Wiedergabesituation und Art des wiedergegebenen Datenstromes muss der MPEG2-Dekoder unterschiedlich initialisiert werden.

Die Herausforderung bei der Umsetzung der Interrupt-Behandlung in einer statischen Interrupt-Service-Routine besteht in der Lösung der folgenden beiden Probleme:

1. Die (automatische) Bestimmung aller zur Initialisierung notwendigen Parameter oder deren Übermittlung an den Gerätetreiber.

2. Umsetzung der vielen unterschiedlichen Strategien zur Initialisierung des MPEG2-Dekoders in einer statischen Routine.

Die zur Re-Initialisierung des HiPEG+ notwendigen Parameter lassen sich beispielsweise über die im Folgenden besprochenen Herangehensweisen bestimmen:

1. Teile der Initialisierungs-Parameter für den MPEG2-Dekoder sind aus dem abgespielten Transportstrom ermittelbar. Dazu ist jedoch eine umfangreichere Analyse des Datenstromes notwendig, die ohne die dem ARC-Prozessor zur Verfügung stehende Hardware-Unterstützung sehr aufwändig ist und damit das Laufzeitverhalten des Systems negativ beeinflusst. Darüber hinaus lassen sich viele der Behandlungsparameter nicht direkt aus dem abgespielten Datenstrom heraus bestimmen, sondern sind von externen Faktoren abhängig (Position der CineCard in der Projektionsmatrix, Rolle der CineCard als Master oder Slave). Diese Herangehensweise ist für den betrachteten Lösungsansatz ungeeignet.

2. Eine andere Möglichkeit der Bestimmung der Reinitialisierungs-Parameter besteht darin, die Konfigurationsaufrufe bei der Initialisierung zu protokollieren. Die Reinitialisierung ähnelt sehr der Initialisierung vor dem Beginn der Wiedergabe. Durch eine intelligente Filterung und Anpassung der Befehlsfolge können die zur Reinitialisierung notwendigen Parameter bestimmt werden. Die Trennung der für die Re-Initialisierung notwendigen und unnötigen Registerzugriffe setzt jedoch eine relativ komplexe Programmlogik voraus. Ohne Filterung erreicht das Protokoll der Registerzugriffe schnell einen Umfang der den Einsatz für die Interrupt-Behandlung unmöglich macht. In einem vorangehenden Implementations-Versuch der Interrupt-Behandlung wurde der Protokoll-Ansatz verfolgt. Er hat sich jedoch als nicht effizient realisierbar herausgestellt und wurde verworfen.

3. Die zur Re-Initialisierung des HiPEG+ notwendigen Informationen über den wiederzugebenden Datenstrom und die externen Faktoren der Wiedergabe stehen dem Cineboxd in Form von Konfigurationsskripten zur Verfügung. Die für diesen Anwendungsfall zweckmäßigste Lösung ist, die für die Interrupt-Behandlung benötigten Parameter im Cineboxd auf Basis der Konfigurationsskripte zu bestimmen und über eine Erweiterung der zur Konfiguration verwendeten Kernelschnittstelle an den Gerätetreiber zu übermitteln.

Für die Bewertung der des Lösungansatzes der statischen Interrupt-Routine wird dritte Herangehensweise verwendet.

3.3.2 Analyse bezüglich der Basisfunktionalität

Die Anforderungen an die Basisfunktionalität lassen sich mit einer statischen Interrupt-Routine erfüllen. Alle geforderten Sprachaspekte sind durch die Implementations-Sprache C einfach realisierbar. Die Reaktion der Interrupt-Behandlung

lässt sich zum Beispiel über das Setzen von Flags und Variablen aus dem Userspace heraus beeinflussen. Die Wiedergabesituation verändernde äußere Faktoren, wie die Position der CineCard in der Projektionsmatrix, können auf diese Weise berücksichtigt werden. Die zur Verfügung stehenden Interrupt-Behandlungsmöglichkeiten der Interrupt-Routine lassen sich jedoch nur statisch zum Übersetzungszeitpunkt festlegen. Um auf sehr verschiedene Interrupt-Situationen reagieren zu können, muss in einer statischen Implementation jeder Spezialfall durch einen eigenen Programmteil berücksichtigt werden.

Das Aufrufen der Interrupt-Routine aus dem Userspace heraus kann durch eine Erweiterung der bisherigen Kernelschnittstelle ermöglicht werden. Durch den Aufruf mit speziellen Parametern lassen sich verschiedene Wiedergabesituationen simulieren und testen. Ein Test der Implementation im Userspace oder das Ausführen der Interrupt-Behandlung im Einzelschritt-Modus ist jedoch prinzipbedingt unmöglich und ließe sich allenfalls wie beim ARC-Beispiel durch eine Parallel-Implementation realisieren.

3.3.3 Evaluation

Hardware

Eine auf dem Host-System basierende Interrupt-Behandlung besitzt gegenüber der Hardware-Lösung zahlreiche Vorteile. Der wichtigste Vorteil ist, dass die Hardware der CineCards nicht verändert werden muss. Die Re-Initialisierung im Interrupt-Kontext bedient sich der gleichen Zugriffsmechanismen wie die Initialisierung vor der Wiedergabe von Datenströmen. Eine vollständige Hardware-Unabhängigkeit bezüglich der Infrastruktur der CineCard lässt sich durch eine statische Interrupt-Routine jedoch nicht erreichen. Die statische Implementation ist stark mit der Interna des verwendeten MPEG2-Dekoders verwoben und muss bei einem Wechsel des Videodekoders neu implementiert werden.

Mit Hilfe einer auf dem Host-System ausgeführten Interrupt-Routine ist eine Synchronisation der Interrupt-Behandlungen mehrerer MPEG2-Dekoder möglich. Die Interrupt-Routine kann dabei auf alle CineCards und die Ressourcen des Host-Systems zugreifen.

Laufzeitverhalten

Die, verglichen mit dem ARC-Prozessor, um ein Vielfaches höhere Verarbeitungsgeschwindigkeit des Host-Systems, lässt sich im Interrupt-Kontext gut nutzen. Selbst relativ umfangreiche Interrupt-Service-Routinen schränken den ebenfalls zeitkritischen Transport der Videodatenströme nur wenig ein.

Die statische Interrupt-Service-Routine ist die mit Abstand am häufigsten verwendete Lösung zur Interrupt-Behandlung, weil die meisten Geräte nur eine geringe Flexibilität hinsichtlich der Reaktion auf Interrupt-Requests benötigen. Die Mitglieder der CineCard-Familie unterscheiden sich untereinander stark in Bezug auf Hardware-Ausstattung und Aufbau und erfordern je nach Wiedergabesituation eine sehr viel komplexere Behandlung aufgetretener Interrupt-Requests als andere Gerätetypen. Eine typische Interrupt-Behandlung besteht aus mehr als 200 einzelnen Registerzugriffen[3]. Je nach Ausstattung der CineCard und Anzahl der verwendeten MPEG2-Dekoder sind viele Sonderfälle zu beachten. Die Berücksichtigung zahlreicher Sonderfälle und die Auswertung einer großen Anzahl von Re-Initialisierungsparameter hat ebenfalls einen großen Behandlungs-Overhead zur Folge, der sich negativ auf das Laufzeitverhalten auswirkt.

Aufgrund der hohen Komplexität der statischen Interrupt-Routine im Interrupt-Kontext, ist die Gefahr von Seiteneffekten bei einer parallelen Ausführung mehrerer Instanzen sehr hoch. Beispielsweise ist der gleichzeitige Zugriff auf Register, die durch das Auslesen zurückgesetzt werden, bei dieser Form der Interrupt-Behandlung nur sehr schwer zu synchronisieren. Eine vollständig reentrante Funktion unter Beachtung aller durch das Auslesen zurückgesetzter Register, ist bei der erwarteten Komplexität der statischen Interruptroutine nicht mit vertretbarem Testaufwand realisierbar.

Die zwangsweise sequentielle Behandlung unmittelbar nacheinander auftretender Interrupt-Requests erhöht jedoch die Interrupt-Latenz enorm. Da das Host-System neben den Interrupt-Anforderungen zur Fehlerbehandlung zusätzlich mit dem Transport der Video-Datenströme beschäftigt ist, wirkt sich eine zusätzliche Erhöhung der Interrupt-Latenz besonders kritisch auf die Fehlerstabilität und Interaktivität des Systems aus.

Im Forschungsalltag macht sich die eingeschränkte Flexibilität einer statischen Interrupt-Service-Routine stark bemerkbar. Jeder von der Implementierung nicht berücksichtigte Sonderfall bedingt eine Änderung des Gerätetreibers. Dieser muss anschließend übersetzt, erneut geladen und getestet werden. Größere Änderungen zur Laufzeit sind damit ausgeschlossen. Eine zu Testzwecken veränderte Reihenfolge der Behandlungsschritte ist beispielsweise nur durch eine Veränderung des Quelltextes erreichbar. Die Integration und Inbetriebnahme neuer Komponenten auf der CineCard wird durch die mangelnde Flexibilität stark behindert.

Die Übermittlung der für die Interrupt-Behandlung notwendigen Parameter lässt sich, wie zuvor beschrieben, über eine Erweiterung der bestehenden Kernel-Schnittstelle des Gerätetreibers realisieren. Deren Aufbereitung aus den vorhandenen Konfigurationsskripten ist jedoch sehr aufwändig, weil die Bestimmung der

[3] Testsequenz 0dvb

Parameter nur durch eine relativ aufwändige Analyse der Registerzugriffe der Konfigurationsskripte möglich ist. Die Analyse setzt ein Verständnis der Bedeutung der Registerzugriffe voraus und erhöht damit die Komplexität der zur Extraktion der Behandlungsparameter verwendeten Filter-Software.

Eine einfachere Möglichkeit besteht darin, die für die Interrupt-Verarbeitung benötigten Informationen parallel zu den Konfigurationsskripten in einer bereits aufgearbeiteten Form zu speichern. Diese Methode birgt jedoch das Risiko von Inkonsistenzen zwischen den zur Initialisierung verwendeten Konfigurationsskripten und den Parametern der Interrupt-Behandlung.

Implementierung

Die zuvor beschriebene hohe Komplexität im Interrupt-Kontext des Kernels wirkt sich negativ auf die Wartbarkeit der Interrupt-Behandlung aus und bedingt eine große Anzahl von Änderungen am bestehenden Software-Umfeld. Die große Anzahl der zur Interrupt-Behandlung notwendigen Parameter setzt eine umfangreiche Erweiterung der Kernelschnittstelle voraus.

Zur Übermittlung und Aufbereitung der zahlreichen Behandlungsparameter muss der Cineboxd ebenfalls stark erweitert werden. Der Umfang der Änderungen am bestehenden Anwendungsumfeld und der spätere Aufwand bei der Wartung des Software-Systems ist vergleichsweise hoch. Zusätzlich ist auf dem Wiedergabesystem für die Bestimmung der Behandlungsparameter auf Basis der Konfigurationsskripte weitere, noch zu implementierende Filter-Software nötig.

3.3.4 Zusammenfassung der Bewertung

Die auf dem Host-System ausgeführte Interrupt-Behandlung bietet gegenüber dem Ansatz über die ARC-Prozessoreinheit zahlreiche Vorteile: Die Hardware der CineCard-Familie muss nicht aufwändig angepasst werden. Der Zugriff auf die Ressourcen des Host-Systems ist problemlos möglich und der Einsatz einer Cross-Compiler Toolchain ist unnötig. Eine statische Interrupt-Routine besitzt jedoch zu viele Unzulänglichkeiten in Bezug auf die geforderte Flexibilität und die Wartbarkeit der Software und ist damit als Lösungstrategie für die Interrupt-Behandlung der HHI CineCard ungeeignet.

Tabelle 3.2: Bewertung der statischen Implementierung

Bewertungs kriterium	**Beschreibung**	**Note und Wichtung**
Kategorie Hardware		
Hardware-Änderungen	Re-Design der CineCards unnötig	1 (20%)
Hardware-Unabhängigkeit	hohe Abhängigkeit von der Hardware im Quelltext des Gerätetreibers	4 (7%)
Interrupt-Synchronisation	potentiell aufwändig zu realisieren, Hauptproblem: Gleichzeitigkeit	4 (3%)
Laufzeitverhalten		
Testbarkeit (zur Laufzeit)	sehr hohe Komplexität, kein Einzelschrittmodus möglich	5 (10%)
Laufzeit-anpassung	eingeschränkt über Parameter möglich, beschränkt durch Kernelschnittstelle	4 (5%)
Parameter-aufbereitung	Aufbereitung erfordert Verständnis der Bedeutung der Registerzugriffe	4 (5%)
Interrupt-Latenz	nur sequentielle Behandlung möglich, Host-System mit Wiedergabe belastet	3 (5%)
Behandlungs-Overhead	potentiell hoch durch viele mögliche Sonderfälle	3 (3%)
Implementierung		
Wartbarkeit	zu hohe Komplexität	5 (20%)
Umfang der Änderungen	sehr hoch, Interrupt-Service-Routine, Parameteraufbereitung & übermittlung	5 (10%)
Infrastruktur Sicherheit	notwendige Software zur Parameteraufbereitung	2 (7%)
Komplexität (im Kernel)	sehr hoch im Kernel und in der Parameteraufbereitung, nicht reduzierbar	5 (5%)
Gesamtnote	**der Lösungsansatz ist unzweckmäßig**	**3,7**

3.4 Binäre, nachladbare Module

Die hohe Komplexität der statischen Implementierung im Kernel und die damit verbundene schlechte Wartbarkeit und Fehleranfälligkeit legen eine Verschiebung der kritischen Teile in den Userspace nahe. Das automatische Generieren von Modulen zur Interrupt-Behandlung verspricht eine wesentlich erhöhte Flexibilität und eine stark reduzierte Komplexität im Kernel-Kontext.

3.4.1 Beschreibung

Durch das Wissen um die notwendige Konfiguration der CineCards in Bezug auf die Datenstromparameter und die Wiedergabesituation lässt sich im Applikationskontext des Userspace eine genau auf den wiedergegebenen Datenstrom und die momentane Wiedergabesituation angepasste Funktion zur Behandlung der möglichen Interrupt-Anforderungen erzeugen. Die Konfigurationsskripte bestehen aus mehreren logische Abschnitten, die sich auch ohne das Wissen um die semantische Bedeutung der Registerzugriffe anhand ihrer Position in den Konfigurationsskripten unterscheiden lassen.

Auf Basis der für die Interrupt-Verarbeitung relevanten Teile der Konfigurationsskripte lassen sich nachladbare Kernel-Module für die Interrupt-Behandlung erstellen. Die Konfigurationsskripte bestehen zum überwiegenden Teil aus einfachen Registerzugriffen, die sich sehr leicht in entsprechende Funktionen in einem C-Quelltext übersetzen lassen. Durch die direkte Transkodierung der einzelnen Gruppen von Initialisierungsbefehlen wird eine komplexe Suche nach Initialisierungsparameter in den Konfigurationsskripten (wie bei der statischen Interrupt-Routine) vermieden. Ein aus dem C-Quelltext erstelltes Kernel-Modul könnte im Rahmen der Initialisierung der CineCards vor dem Beginn der Wiedergabe in den CineCore geladen und aktiviert werden.

Das Einbinden der Interrupt-Behandlungsmodule entspricht weitgehend der Registrierung eines Gerätetreibers beim PCI-Subsystem des Linux-Kernels, jedoch findet die Registrierung beim CineCore und nicht direkt beim PCI-Subsystem des Kernels statt. Im Falle eines Interrupt-Requests führt der CineCore, geordnet nach Priotitäten, alle für den aktuellen Interrupt-Vektor registrierten Behandlungs-Routinen aus. Die vom Linux-Kernel eingesetzte späte Bindung von nachgeladenen Objekt-Modulen ermöglicht den Interrupt-Behandlungsmodulen die Nutzung der Zugriffsfunktionen auf die Hardware, die von dem jeweiligen Backend zur Verfügung gestellt wird.

Die Module zur Interrupt-Behandlung sind eng an die wiederzugebenden Datenströme gebunden. Zur Verteilung der Module auf die Wiedergabesysteme sind zwei grundsätzliche Möglichkeiten mit ihren Vor- und Nachteilen abzuwägen:

Generierung zur Laufzeit: Das Erzeugen der Interrupt-Behandlung auf Basis der Konfigurationsskripte geschieht direkt auf dem Wiedergabesystem. Es müssen sowohl die Quellen des laufenden Linux-Kernels, als auch die zur Übersetzung der Software notwendigen Programme installiert sein.

Binäre Distribution: Die fertig übersetzten Kernel-Module werden zusammen mit den Konfigurationsskripten und dem Datenstrom verteilt. Die Module müssen allerdings genau zur verwendeten Rechner-Architektur des Host-Systems und dessen Kernel-Version passen.

Werden binäre Module eingesetzt, besteht bei einer Kernel-Aktualisierung die Gefahr der Inkompatibilität der Kernel-Module zum laufenden Kernel. Dadurch ist eine erneute Übersetzung aller Module zur Interrupt-Behandlung notwendig. Die proprietären Treiber einiger Grafikkartenhersteller zeigen, dass dies zu einem stark erhöhten Aufwand bei der Administration der Wiedergabesysteme führt. Im Rahmen des Tests von Hardware-Komponenten der CineCards werden oft Änderungen an den Konfigurationsskripten vorgenommen. Für den geplanten Einsatzzweck sind in binärer Form gespeicherte Modulen wegen der hohen Gefahr von Inkonsistenzen zu den sich oft verändernden Konfigurationsskripten ungeeignet. Binäre Module müssten darüber hinaus für jede mögliche Kernel-Version der Wiedergabesysteme vorhanden und verwaltet werden. Dies erschwert den Einsatz eines inhomogenen Projektionsverbundes enorm, weil sich die Module für beispielsweise SMP- und Uniprozessormaschinen binär unterscheiden.

3.4.2 Analyse bezüglich der Basisfunktionalität

Die Erzeugung der Interrupt-Behandlung aus den zur Initialisierung der CineCards verwendeten Konfigurationsskripten bietet die Möglichkeit, auf alle geforderten Interrupt-Situationen reagieren zu können. Die durch den C-Compiler bereitgestellten Sprachaspekte erfüllen vollständig die Anforderungen an die Basisfunktionalität.

Die Interrupt-Routine ist wie im vorangegangenen Beispiel der statischen Implementation aus dem Kontext des Benutzers heraus aufrufbar. Durch selektives Aufrufen einzelner Interrupt-Behandlungsmodule sind zusätzlich verschiedene Interrupt-Situationen simulierbar.

3.4.3 Evaluation

Hardware

Der Lösungsansatz der Interrupt-Behandlung mit nachladbaren Kernel-Modulen profitiert von seiner Unabhängigkeit von der verwendeten Hardware der CineCard.

Zur Inbetriebnahme von neuen Hardware-Komponenten auf der CineCard sind oft nur kleinere Änderungen in den zur Initialisierung verwendeten Konfigurationsskripten notwendig. Die Änderungen an den Konfigurationsskripten wirken sich bei diesem Lösungsansatz automatisch auch auf die Interrupt-Behandlung aus. Das Konzept zur Interrupt-Behandlung ist damit weitestgehend unabhängig von der Hardware der eingesetzten CineCard.

Eine Synchronisation der Interrupt-Behandlungen verschiedener MPEG2-Dekoder ist mit dem betrachteten Lösungsansatz möglich, erfordert aber Informationen über die Interna des Gerätetreibers, die einer automatisch generierten Interrupt-Service-Routine nur sehr aufwändig zur Verfügung zu stellen sind. Der Informationsaustausch zwischen getrennt interagierenden Modulen zur Interrupt-Behandlung ist nur über wohl definierte Schnittstellen und Inter-Modul-Kommunikation möglich und bedingt damit einen erhöhten Testaufwand.

Laufzeitverhalten

Fehler in den automatisch generierten Kernel-Modulen können sich fatal auf die Systemstabilität auswirken. Die zur Aufbereitung und Übersetzung der Konfigurationsskripten verwendete Software, muss eine umfangreiche Anzahl von Fehler und potentiellen Fehlermöglichkeiten in den untersuchten Skripten erkennen und melden können.

Die dafür notwendigen Programmfunktionen besitzten deshalb eine relativ hohe Komplexität. Fehler bei der Aufbereitung der Initialisierungsparameter oder bei der Erstellung der Module, sind sehr schwer zu finden und äußern sich möglicherweise nur durch Seiteneffekte zur Laufzeit. Das Ergebnis einer automatischen Aufbereitung ist nur sehr schwer auf seine Richtigkeit zu überprüfen. Eine unveränderte Ausführung der Behandlungsmodule im Einzelschrittmodus ist durch die Art der Modulerzeugung prinzipbedingt ausgeschlossen.

Zur Laufzeit veränderliche Parameter müssen vor der Übersetzung der Interrupt-Behandlungsmodule an der richtigen Stelle im Quelltext eingefügt werden. Die Reihenfolge der Befehle zur Initialisierung in den Konfigurationsskripten kann sich jederzeit ändern. Die Position der zur Laufzeit veränderlichen Parameter muss deshalb in den Skripten explizit gekennzeichnet werden.

Ein großer Vorteil binärer Module für die Interrupt-Behandlung ist deren Effizienz. Durch die genau auf die Wiedergabesituationen angepasste Generierung der Kernel-Module wird praktisch kein Behandlungs-Overhead erzeugt. Die Abfrage der Statusregister zur Auswahl der Behandlungsstrategie kann ebenfalls genau auf die Anforderungen angepasst werden, damit lässt sich mit der betrachteten Lösung eine in Bezug auf die anderen Lösungsansätze sehr geringe Interrupt-Latenz erreichen.

Implementierung

Zur Umsetzung des Lösungsansatzes sind relativ viele Änderungen am bestehenden Anwendungumfeld vorzunehmen. Der Cineboxd muss für die Analyse der Konfigurationsskripte und die automatische Erzeugung der Interrupt-Behandlungsmodule erweitert werden. Die bestehende Kernelschnittstelle des CineCore unterstützt bisher nicht das Einbinden von Modulen zur Interrupt-Behandlung. Funktionen, die den Test der erzeugten Interrupt-Behandlungen ermöglichen und eine Überprüfung der korrekten Arbeitsweise der erzeugten Interrupt-Routine ermöglichen, sind noch nicht vorhanden.

Das Verlegen der Komplexität in den Userspace und die damit entfallende Filterung der Skripte nach Initialisierungsparametern erhöht die Wartbarkeit der Software und vermindert durch hohe Komplexität im Kernel entstehende Fehler.

Zusätzlich lässt sich eine Userspace-Applikation durch einfache Testbarkeit wesentlich besser erweitern und anpassen als vergleichbarer Kernelcode. Die im unveränderlichen Teil des Gerätetreibers vorhandenen Funktionen müssen nur bei großen strukturellen Änderungen modifiziert werden. Dies erhöht die Wartbarkeit der Interrupt-Routine gegenüber den zuvor betrachteten Lösungen enorm.

Der größte Schwachpunkt der Interrupt-Behandlung über nachladbare Kernel-Module liegt im Aufbereitungsprozess und dem dafür notwendigen Softwareumfeld. Zur Erzeugung von Kernel-Modulen müssen neben den Kernelquellen des momentan laufenden Linux-Kernels zusätzlich ein C-Compiler und alle zur Erzeugung von Kernel-Modulen notwendigen Dienstprogramme installiert sein. Dies schränkt die Einsatzmöglichkeiten der Software auf festplattenlosen Wiedergabesystemen stark ein.

Das Übersetzen und Laden von Kernel-Module erfordert zudem die Rechte des Systemverwalters und stellt eine nicht zu unterschätzende Sicherheitslücke für das Host-System dar. Ein durch Schadprogramme vorsätzlich eingeschleuster Programmcode besitzt in Form eines Kernel-Modules den vollen Zugriff auf das gesammte Wiedergabesystem. Unbeabsichtigte Fehler bei der Übersetzung der Kernel-Module können zu schweren Systemabstürzen führen.

Die Komplexität im Kernel-Kontext ist bei diesem Lösungsansatz durch die gute Anpassung auf die verwendete Wiedergabe-Hardware relativ gering. Die statische (nicht nachzuladenden) Funktionen des Kerneltreibers besitzen eine sehr geringe Komplexität, sind gut zu testen und müssen nur bei größeren Änderungen der Struktur der Interrupt-Behandlung angepasst werden.

Die automatische Generierung der nachladbaren Teile der Interrupt-Behandlung vermindert, gegenüber der statischen Interrupt-Routine, die Fehleranfälligkeit der im Kernel laufenden Software entscheidend. Bei der automatischen Generierung der Behandlungsmodule entstehende Fehler können jedoch zu schweren Systemabstürzen führen.

3.4.4 Zusammenfassung der Bewertung

Die Interrupt-Behandlung über nachladbare Module bietet gegenüber einer statischen Interrupt-Routine zwei große Vorteile: Die Behandlungsstrategie lässt sich wesentlich flexibler an die Erfordernisse anpassen und die Wartbarkeit erhöht sich durch die automatische Generierung entscheidend.

Die Handhabung der Module für die Interrupt-Behandlung ist ein nicht zu unterschätzender Schwachpunkt dieses Lösungsansatzes. Die für das Laden der Module zuständige Applikation muss mit den Rechten des Systemverwalters ausgeführt werden. Unbeabsichtig oder vorsätzlich geladener, schadhafter Code gefährdet die Systemstabilität des Host-Systems.

Die mit Hilfe der Konfigurationsskripte erzeugten Interrupt-Behandlungsmodule besitzen zwar eine wesentlich geringere Komplexität als die zuvor beschriebene statische Interrupt-Routine, deren Testbarkeit ist jedoch aufgrund der automatischen Generierung der Module sehr eingeschränkt.

Tabelle 3.3: Bewertung der binär nachladbaren Module

Bewertungs-kriterium	**Beschreibung**	**Note und Wichtung**
Kategorie Hardware		
Hardware-Änderungen	Re-Design der CineCards unnötig	1 (20%)
Hardware-Unabhängigkeit	durch Transkoder weitestgehend von der Hardware der CineCard	2 (7%)
Interrupt-Synchronisation	durch Inter-Module-Kommunikation relativ aufwändig	3 (3%)
Laufzeitverhalten		
Testbarkeit (zur Laufzeit)	sehr eingeschränkt, hohe Komplexität des Transkoders, kein Einzelschrittmodus	4 (10%)
Laufzeit-anpassung	Parametrisierung nur sehr aufwändig möglich,	5 (5%)
Parameter-aufbereitung	relativ komplexe Transkodierung der Befehle der Konfigurationsskripte	3 (5%)
Interrupt-Latenz	sehr gering, durch genau auf die Hardware angepasste Module	1 (5%)
Behandlungs-Overhead	sehr gering, Behandlung auf Wiedergabe-situation anpassbar	1 (3%)
Implementierung		
Wartbarkeit	gut durch Transkoder-Software im Userspace, Kernel-Funktionen nur selten anzupassen	2 (20%)
Umfang der Änderungen	relativ hoch, Lademechanismus für Module, Software zur Modul-Erstellung	3 (10%)
Infrastruktur Sicherheit	Kernelquellen und Compiler nötig, großen Sicherheitsrisiko	5 (7%)
Komplexität (im Kernel)	geringe Komplexität, automatisch generiert, Fehler bei Aufbereitung äußern sich fatal	2 (5%)
Gesamtnote	**geeignet, jedoch nicht optimal**	**2,5**

3.5 Skriptlösung

Alle zuvor betrachteten Lösungen basierten entweder auf der Hardware oder auf bereits etablierten Strategien zur Interrupt-Behandlung. Den besonderen Vorteilen, die die bereits vorhandenen Konfigurationsskripte mit sich bringen, wurde dabei wenig Aufmerksamkeit geschenkt.

Die enge Verwandtschaft der Interrupt-Behandlung mit der Initialisierungsprozedur vor der Datenstromwiedergabe legt nahe, auch große Teile der für die Konfiguration der CineCard verwendeten Software-Teile für die Interrupt-Behandlung zu verwenden. Viele schon vorhandene Funktionen, um zum Beispiel auf die Register der CineCard zuzugreifen, werden in gleicher Form auch im Interrupt-Kontext benötigt. Die Verwendung von Skripten zur Interrupt-Behandlung verspricht eine höchstmögliche Flexibilität bei guter Wartbarkeit der Software.

3.5.1 Beschreibung

Aus Sicht der zur Initialisierung der CineCards verwendeten Konfigurationsskripte ist eine ebenfalls auf Skripten basierende Interrupt-Behandlung sehr naheliegend. Die zur Initialisierung der CineCards verwendeten Konfigurationsskripte bestehen aus mehreren logischen Abschnitten, in denen die verschiedenen Subsysteme der CineCards konfiguriert werden. Einige Teile dieser Konfigurationsskripte lassen sich ohne Änderung auch für die Interrupt-Behandlung verwenden. Darüber hinaus benötigte Teile der Interrupt-Behandlung können sehr einfach und automatisch aus den Konfigurationsskripten generiert werden.

Um Interrupt-Skripte im Interrupt-Kontext ausführen zu können, müssen diese zuvor in den Adressbereich des Kernels kopiert werden. Eine Ausführung des Skript-Interpreters im Userspace würde sich wegen der häufigen Kontextwechsel zwischen dem Kernel- und Applikationskontext fatal auf die Laufzeit der Interrupt-Routine auswirken.

Zum Einlesen der Interrupt-Skripte kann der für die Konfigurationsskripte verwendete lexikalische Scanner des Cineboxd verwendet werden. Die Interrupt-Skripte werden über die bereits vorhandene Konfigurationsschnittstelle des Gerätetreibers in des Adressbereich des Kernels kopiert und dort gespeichert. Zum Laden und Entladen der Interrupt-Skripte sind nur minimale Änderungen an der Konfigurationsschnittstelle notwendig.

3.5.2 Analyse bezüglich der Basisfunktionalität

Die Interrupt-Behandlung durch Skripte zeichnet sich durch eine sehr hohe Flexibilität aus. Wie bei den zuvor betracheten Lösungsansätzen, lässt sich die Interrupt-Behandlung auch bei der Skriptlösung vom Anwender manuell auslösen.

Alle für die Behandlung von Interrupt-Anforderungen benötigten Sprachelemente lassen sich problemlos in einer Skriptsprache umsetzen. Deren Syntax und Struktur kann individuell auf die Anforderungen der CineCards angepasst werden.

3.5.3 Evaluation

Hardware

Der auf Interrupt-Skripten basierende Lösungsansatz ist vollständig unabhängig von der verwendeten Hardware. Weder der Typ des Host-Systems noch Änderungen an den Komponenten der CineCard wirken sich direkt auf die Skriptverarbeitung aus. Eine Änderung an der Hardware der CineCards ist damit ebenfalls unnötig.

Die Verwendung der bestehenden Funktionen des Gerätetreibers zur Konfiguration der CineCards schränkt die gleichzeitige Verarbeitung der Interrupt-Anforderungen mehrerer MPEG2-Dekoder nicht ein. Im Gegensatz zu den zuvor betrachteten Lösungangsätzen können die bereits im Gerätetreiber vorhandenen Funktionen zum Registerschutz und zur Umsetzung von Transaktionen wiederverwendet werden.

Laufzeitverhalten

Eine große Stärke der Skriptlösung für die Interrupt-Behandlung ist die gute Testbarkeit. Im Cineboxd ist der größte Teil der zur Fehlersuche benötigten Funktionen bereits enthalten. Durch die direkte Wiederverwendung der Konfigurationsschnittstelle ist sogar eine direkte Ausführung der zur Interrupt-Behandlung bestimmten Skripte im Cineboxd möglich.

Die große Ähnlichkeit zwischen den Konfigurations- und den Interrupt-Skripten macht eine aufwändige Aufbereitung der Parameter zur Interrupt-Behandlung unnötig. Zur Laufzeit veränderliche Parameter der Interrupt-Verarbeitung können in Form von Variablen zwischen den Konfigurationsskripten und des Interrupt-Skripten ausgetauscht werden.

Eine Interpretation der Skripte im Interrupt-Kontext kann sich fatal auf die Laufzeit der Interrupt-Routine auswirken. Ein Skript-Interpreter bedingt einen erhöhten Behandlungs-Overhead der Interrupt-Service-Routine und kann sich eben-

falls durch eine erhöhte Interrupt-Latenz bemerkbar machen. Durch die Interpretation der Skripte und den anschließenden Aufruf der zur Ansteuerung der Hardware verwendeten Funktionen, ist die Verarbeitungsgeschwindigkeit der Interrupt-Service-Routine geringer als bei einem direkten Zugriff auf die Register der CineCard. Der erzielbare prozentuale Geschwindigkeitsvorteil durch eine nicht interpretierte Lösung ist jedoch durch die maximale Geschwindigkeit, mit der auf die Hardware zugegriffen werden kann begrenzt.

Der Einfluss des Skript-Interpreters auf die Verarbeitungsgeschwindigkeit der Interrupt-Routine hängt zusätzlich entscheidend von der Implementierung des Interpreters und dessen Effizienz ab. Die im folgenden Kapitel 4 näher beschriebene Implementation besitzt ein sehr günstiges Laufzeitverhalten im Interrupt-Kontext und kann dadurch mit den anderen betrachteten Lösungsansätzen konkurrieren.

Implementation

Zur Umsetzung der skriptbasierten Interrupt-Behandlung sind nur wenige Änderungen an dem bestehenden Softwareumfeld vorzunehmen. Der größte Teil der Änderungen betrifft die zentrale Komponente des Gerätetreibers, den CineCore. Die Änderungen an der Kernelschnittstelle und am Cineboxd beschränken sich auf die Implementierung der Befehle zum Laden und Entladen der Interrupt-Skripte und die zur Erweiterung der Funktionalität notwendigen Kontrollstrukturen und Variablen.

Zur Verarbeitung der Skripte durch den im Cineboxd integrierten Parser ist keine zusätzliche Softwareausstattung auf dem Wiedergabesystem notwendig. Ein Zugriff auf die Hardware der CineCard und die Ressourcen des Gerätetreibers ist ausschließlich über die Funktionen der Konfigurationsschnittstelle des CineCore möglich. Schon vorhandene Mechanismen zum Speicherschutz und zur Syntaxüberprüfung können direkt wiederverwendet werden. Dies führt zu einer stark erhöhten Sicherheit und Fehlertolleranz des gesamten Softwaresystems. Werden beim Einlesen der Skripte Syntaxfehler oder zum Beispiel Bereichsüberschreitungen festgestellt, könnte das Laden der Interrupt-Skripte mit einer Fehlermeldung abgebrochen werden. Durch unbeabsichtigt (oder auch vorsätzlich) eingefügte fehlerhafte Anweisungen kann die Systemstabilität nicht wie bei der auf binären Modulen basierenden Interrupt-Behandlung gefährdet werden.

Durch den Einsatz einer Skriptlösung lässt sich die Wartbarkeit der Software ebenfalls stark erhöhen. Die im Kernel-Kontext laufenden Funktionen besitzen eine geringe Komplexität und lassen sich mit Hilfe der Konfigurationsschnittstelle sehr gut testen. Bei der auf Skripten basierenden Lösung bedingen nur sehr wenige Änderungen an der Hardware auch ein Änderung des Gerätetreibers oder der Anwendungen.

3.5.4 Zusammenfassung der Bewertung

Die hier betrachtete Lösung zur Interrupt-Behandlung über Interrupt-Skripte ist hinsichtlich ihrer hohen Flexibilität und guten Wartbarkeit am besten für den Einsatz unter den gegebenen Voraussetzungen geeignet. Die Skriptlösung besitzt wesentlich geringere Anforderungen an das notwendige Softwareumfeld auf dem Wiedergabesystem und zeichnet sich gegenüber den binär nachladbaren Modulen durch eine hohe Sicherheit aus.

Die zu erwartenden Probleme bei der Verarbeitungsgeschwindigkeit der Interrupt-Requests lassen sich durch eine effiziente Implementierung weitestgehend vermeiden.

Tabelle 3.4: Bewertung der Skriptlösung

Bewertungs-kriterium	**Beschreibung**	**Note und Wichtung**
Kategorie Hardware		
Hardware-Änderungen	Redesign der CineCards unnötig	1 (20%)
Hardware-Unabhängigkeit	vollständig unabhängig von der Hardware (Host und CineCards)	1 (7%)
Interrupt-Synchronisation	durch bereits vorhandene Funktionen relativ einfach	1 (3%)
Laufzeitverhalten		
Testbarkeit	sehr gut, Einzelschrittmodus möglich	1 (10%)
Laufzeit-anpassung	flexibel durch Variablen parametrisierbar	1 (5%)
Parameter-aufbereitung	sehr einfach und oft unnötig, direkte Übernahme aus Konfigurationsskripten	1 (5%)
Interrupt-Latenz	relativ geringe Latenzen erreichbar, Gleichzeitigkeit möglich	2 (5%)
Behandlungs-Overhead	durch Effizienz des Interpreters beeinflusst, vergleichsweise hoch	4 (3%)
Implementierung		
Wartbarkeit	hohe Flexibilität bei Erweiterungen, Kernel-Funktionen selten anzupassen	1 (20%)
Umfang der Änderungen	sehr gering, Erweiterung der Kernel-schnittstelle, und des Cineboxd	1 (10%)
Infrastruktur Sicherheit	keine Infrastruktur nötig, höhste, erreichbare Sicherheit	1 (7%)
Komplexität (im Kernel)	sehr niedrig, Funktionen sehr einfach zu testen	1 (5%)
Gesamtnote	**der Lösungsansatz ist sehr gut geeignet**	**1,1**

3.6 Die Lösungsansätze im Vergleich

Die Ausführung der Interrupt-Behandlung auf der im HiPEG+ integrierten Prozessoreinheit hat sich für den Einsatz von mehreren gekoppelten MPEG2-Dekodern wegen der eingeschränkten Kommunikationsmöglichkeiten als unzweckmäßig herausgestellt.

Mit binären Modulen, die zur Interrupt-Behandlung nachgeladen werden, könnten gegenüber den anderen vorgestellten Lösungsansätzen die kürzesten Interrupt-Latenzzeiten erreicht werden. Jedoch disqualifiziert sich diese Lösung durch die mangelhafte Handhabbarkeit der binären Module und die schlechte Testbarkeit.

Besonders interessant in diesem Vergleich ist das schlechte Abschneiden der statischen Implementierung. Für viele andere Gerätetypen ist dieses einfache Konzept der Interrupt-Behandlung vollkommen ausreichend. Bedingt durch eine hohe Anzahl von veränderlichen Parametern und zahlreichen Sonderfällen übersteigt die Komplexität einer statischen Interrupt-Routine jedoch schnell ein handhabbares Maß.

Die Gegenüberstellung der betrachteten Lösungsansätze zeigt deutlich, dass für die komplexe Behandlung der Interrupt-Requests der CineCards eine auf Interrupt-Skripten basierende Lösung am zweckmäßigsten ist.

Die besonderen Stärken, wie die hohe Flexibilität und der geringe Wartungsaufwand der Software, entsprechen genau dem Anforderungsprofil des Heinrich-Hertz-Instituts. Die im folgenden Kapitel vorgestellte Implementierung des Interrupt-Behandlungskonzeptes verwendet die Skriptlösung.

Tabelle 3.5: Vergleich der betrachteten Lösungsansätze

Kriterium	**hardware-gestützt**	**statisch**	**nachladbare Module**	**skript-basiert**
Hardware				
Hardware-Änderungen	*5*	1	1	1
Hardware-Unabhängigkeit	*5*	4	2	1
Interrupt-Synchronisation	*5*	4	3	1
Laufzeitverhalten				
Testbarkeit	5	*5*	4	1
Laufzeit-anpassung	5	4	5	*1*
Parameter-aufbereitung	3	4	3	*1*
Interrupt-Latenz	*1*	3	1	2
Behandlungs-Overhead	*1*	3	1	*4*
Implementierung				
Wartbarkeit	4	*5*	2	*1*
Umfang der Änderungen	5	5	3	*1*
Infrastruktur und Sicherheit	5	2	5	*1*
(In-Kernel)-Komplexität	*5*	*5*	2	*1*
Gesamtbewertung				
Hauptprobleme	Hardware-Änderung	Flexibilität, Wartbarkeit	Sicherheit, Handhabung	Geschwin-digkeit
Hauptvorteil	Interrupt-Latenz		Geschwin-digkeit	Flexibilität
Gesamtnote	**4,4**	**3,6**	**2,5**	**1,1**

Kapitel 4

Beschreibung der implementierten Lösung

Neben den schon zuvor erläuterten Vorteilen der skriptbasierten Interrupt-Behandlung, ermöglicht der gewählte Lösungsansatz, die zur Interrupt-Verarbeitung erweiterte Funktionalität auch innerhalb der zur Initialisierung verwendeten Konfigurationsskripte zu nutzen.

Mit Hilfe der für die Interrupt-Verarbeitung eingeführten Variablen, können Konfigurationsskripte anderen Konfigurationsskripten Parameter übergeben und von diesen Rückgabewerte erhalten.

Interrupt-Skripte können bei diesem Ansatz wie Konfigurationsskripte aus dem Userspace heraus ausgeführt und getestet werden. Eine Ausführung im Einzelschrittmodus und die Verwendung des Skript-Debuggers ist ebenfalls möglich. Durch die Verwendung der gleichen Funktionen für den Zugriff auf die Hardware, sind die Ergebnisse eines solchen Testlaufes besser auf die Interrupt-Behandlung übertragbar als bei den anderen betrachteten Lösungsansätzen.

Eines der Ziele beim Entwurf der zu implementierenden Lösung bestand darin, möglichst viel von der bereits bestehenden Software auch für die Interrupt-Behandlung zu verwenden. Durch den Einsatz nur *einer* Implementation für alle Arten von Zugriffen auf die CineCards können keine Konsistenzprobleme zwischen den Interrupt- und Konfigurationsskripten im Verhalten der Soft- und Hardware auftreten. Die Wiederverwendung der bestehenden Software und ein modularer Aufbau der Interrupt-Behandlung mindern zustätzlich die durch die Implementierung und Wartung des Softwaresystems entstehenden Kosten.

Um die Anforderungen innerhalb des bestehenden Software-Umfeldes zu erfüllen, ist die vorhandene Software um die folgenden, in den nächsten Abschnitten erläuterten, Punkte zu erweitern:

1. Lade- und Entlademechanismus für die Interrupt-Skripte,
2. Manuelles Starten von Interrupt-Skripten zu Testzwecken,
3. Umsetzung der nötigen Sprachelemente (Kontrollstrukturen und Variablen),
4. Möglichkeiten der Statusabfrage aus dem Userspace heraus (zum Beispiel der Variablen der Interrupt-Skripte),
5. Erweiterter Schutz vor Gleichzeitigkeit (nötig, falls eine Interrupt-Behandlung ein laufendes Konfigurationsskript unterbricht) und
6. Synchronisation der Interrupt-Skripte mehrerer Dekoder.

4.1 Applikationsschnittstelle

4.1.1 Lademechanismus für Interrupt-Skripte

Das bestehende Anwendungsumfeld muss zur Umsetzung des Interrupt-Behandlungskonzeptes über Skripte nur geringfügig modifiziert werden. Die zur Skriptverarbeitung (im Userspace) notwendigen Teile des Cineboxd, wie zum Beispiel die Funktion zur lexikalische Analyse, müssen nur um die neuen Befehle zum Skriptmanagement und die für die Interrupt-Verarbeitung notwendigen Sprachelemente erweitert werden.

Die vom Cineboxd eingelesenen Konfigurations- oder Interrupt-Skripte werden mit Hilfe des im Cineboxd integrierten, lexikalischen Scanners in einzelne Befehle zerlegt. Nach der Gültigkeitsüberprüfung der Argumente der übergebenen Befehle, werden diese über die bereits bestehende Kernel-Schnittstelle an den CineCore übermittelt.

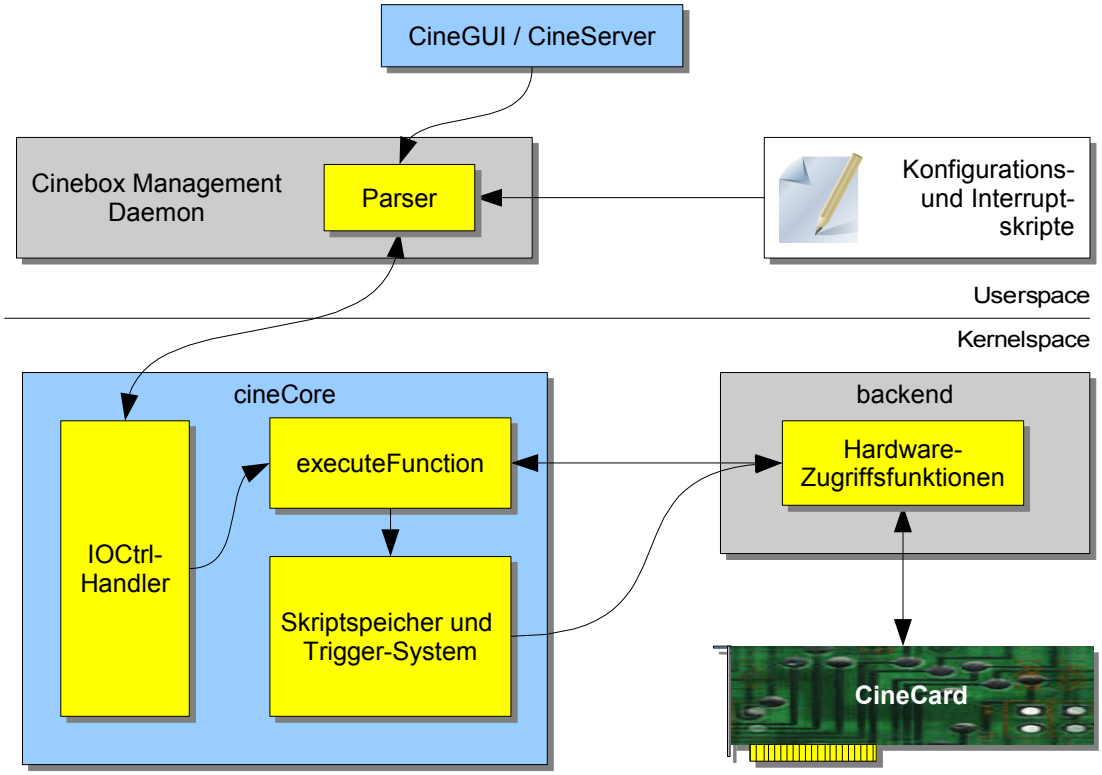
Durch die Verwendung nur einer Implementation für beide Skriptarten und die Befehle des Netzwerkprotokolls, besteht kein äußerer Unterschied zwischen den Befehlen der Interrupt- und Konfigurationsskripte.

Bezüglich des Ausführungszeitpunktes bestehen jedoch Unterschiede: Die Befehle eines Konfigurationsskriptes müssen sofort ausgeführt werden, während die Befehle eines Interrupt-Skriptes im Kernel gespeichert werden.

Zur Unterscheidung der für die Interrupt-Skripte bestimmten Befehle von den normalen Konfigurationsanweisungen werden die Befehle der Interrupt-Skripte in Blöcke gekapselt. Ein solcher Block wird durch die Anweisung `beginIntScript` eingeleitet und durch `endIntScript` beendet.

Ein Interrupt-Skript wird durch drei der Anweisung `beginIntScript` übergebene Parameter eindeutig gekennzeichnet. Der erste Parameter ist die Priorität

Abbildung 4.1: Befehlsverarbeitung



des Interrupt-Skriptes. Skripte mit hoher Priorität werden vor denen mit geringerer Priorität ausgeführt. Die beiden folgenden Parameter bestehen aus dem so genannten Trigger-Register und der Trigger-Maske. Die Rolle der Parameter wird im nächsten Abschnitt genauer besprochen.

Für die Ausführung der an den Kernel-Treiber übergebenen Skriptbefehle ist die Funktion `executeFunction` zuständig. Wird eine Befehlssequenz durch den Anfang eines Interrupt-Skript-Blockes eingeleitet, darf die Funktion `executeFunction` die folgenden Befehle nicht wie bisher sofort ausführen, sondern muss diese für die spätere Abarbeitung zum Interrupt-Zeitpunkt speichern. Die bisherige Funktion muss dafür entsprechend modifiziert werden.

Die in einem Interrupt-Skriptblock stehenden Befehle werden für die spätere Abarbeitung in einer verketteten Liste im CineCore abgelegt. Um eine möglichst effiziente Ausführung der Skripte im Interrupt-Kontext zu gewährleisten, werden

die Skriptbefehle in Form eines Zeigers auf die Hardware-Zugriffsfunktion des jeweiligen Backends und einer, die Argumente der Funktion enthaltenden, Datenstruktur gespeichert. Im Interrupt-Kontext kann die durch den Zeiger referenzierte Funktion direkt mit der gespeicherten Datenstruktur als Argument aufgerufen werden.

Durch den direkten Aufruf entfallen aufwändige Vergleiche zur Auswahl der betreffenden Funktion im Kernel. Dadurch wird die Verarbeitungsgeschwindigkeit der Skriptabarbeitung enorm gesteigert. Der Umgang mit komplexeren Sprachelementen der Skriptsprache wie Verzweigungen oder Variablen, wird im Abschnitt 4.2 erläutert.

Erfolgreich geladene Interrupt-Skripte stehen sofort für die Interrupt-Behandlung zur Verfügung. Die Ausführung eines geladenen Interrupt-Skriptes kann vom Anwender über das Kommando `execIntScript` manuell ausgelöst werden.

Werden Interrupt-Skripte nicht mehr benötigt, können sie mit Hilfe der Funktion `delIntSkript` wieder aus dem CineCore entfernt werden. Die Parameter der Funktionen `delIntSkript` und `execIntScript` müssen dabei exakt denen der Funktion `beginIntScript` entsprechen um das jeweilige Interrupt-Skript eindeutig bestimmen zu können. Sollen alle geladenen Interrupt-Skripte aus dem CineCore entfernt werden, kann dies mit dem Aufruf der Funktion `delIntScript` mit drei Nullen als Parameter erfolgen.

4.1.2 Flexibles Trigger-System

Die Entscheidung, ob beim Auftreten eines Interrupt-Requests des lokalen Busses ein Interrupt-Skript ausgeführt werden soll, wird mit Hilfe von so genannten Trigger-Registern getroffen. Diese werden vor der Ausführung eines Interrupt-Skriptes ausgelesen und mit der bei der Funktion `beginIntScript` übergebenen Trigger-Maske des Interrupt-Skriptes verglichen. Sind die in der Trigger-Maske gesetzen Bits im Trigger-Register ebenfalls gesetzt, wird das Skript ausgeführt.

Mehrere geladene Interrupt-Skripte können von einem Trigger-Register abhängen. Einige auch als Trigger-Register genutzte Register werden durch das Auslesen zurückgesetzt. Für diese Register muss sichergestellt werden, dass das Register nicht mehrfach zum Feststellen der Interrupt-Ursache ausgelesen wird.

Dieses Problem wird im CineCore durch die Verwendung von Trigger-Listen gelöst. Beim Laden eines Interrupt-Skriptes wird ein auszulesendes Register in der Trigger-Liste gespeichert. Ist das auszulesende Register schon von einem anderen Interrupt-Skript in die Trigger-Liste eingetragen worden, wird ein Referenzzähler erhöht. Trigger-Register können erst wieder aus der Liste entfernt werden, wenn der Referenzzähler Null ist, somit wird kein noch verwendeter Listeneintrag un-

beabsichtig entfernt. Vor der Ausführung der Interrupt-Skripte werden alle in die Trigger-Liste eingetragen Register einmalig ausgelesen und die Trigger-Masken mit den gespeicherten Werten verglichen.

4.2 Die Interrupt-Skriptsprache

Die in der Basisfunktionalität geforderten Sprachelemente werden bei den anderen zuvor vorgestellten Lösungsansätzen durch den verwendeten C-Compiler bereitgestellt. Bei der Skriptlösung müssen alle für die Interrupt-Verarbeitung benötigten Sprachelemente im Rahmen des Cineboxd und des Gerätetreibers neu implementiert werden. Die Verwendung einer Skriptsprache im Interrupt-Kontext erfordert eine möglichst einfache und effizient zu interpretierende Befehlsstruktur. Diese darf jedoch die Verwendbarkeit der Skriptsprache nicht einschränken.

Der Einsatz der bereits für die Konfiguration der CineCards verwendeten Skriptbefehle für die Interrupt-Behandlung ermöglicht eine sehr einfach Aufbereitung der Konfigurationsskripte zu Interrupt-Skripten. Um Interrupt-Skripte besser testen zu können, sollen sie ohne Modifikation wie Konfigurationsskripte vom Cineboxd ausführbar sein. Grundvoraussetzung dafür ist ein für beide Skriptarten konsistentes Verhalten der Software.

Um die geforderte Konsistenz zu wahren, wurden die skriptverarbeitenden Teile des Cineboxd und des CineCore für die zusätzliche Verarbeitung von Funktionen und Kontrollstrukturen für die Interrupt-Behandlung erweitert. Auf einige der wichtigsten Erweiterungen soll im Folgenden näher eingegangen werden.

Variablenkonzept

Eine wichtige Neuerung gegenüber den zuvor eingesetzten Konfigurationsskripten sind Variablen. Sie bilden das Bindeglied zwischen den verschiedenen Skriptenarten und sind eine wichtige Voraussetzung für die in den folgenden Abschnitten beschriebenen Kontrollstrukturen. Dynamisch zu erzeugende Variablen ermöglichen eine sehr flexible Kommunikation zwischen Interrupt-Skripten und eine einfache Parametrisierung der Interrupt-Skripte aus dem Userspace. Für die Arbeit mit den Interrupt-Skripten der CineCard werden drei Typen von Variablen benötigt:

Host-Pointer: Die CineCore-Variable zeigt auf eine Ressource innerhalb des Gerätetreibers.

Register-Pointer: Die Variable beinhaltet eine Referenz auf ein Konfigurations- oder Statusregister der PCI-Karte.

U32-Werte: In der Variablen kann ein beliebiger (vorzeichenloser) Integer-Wert gespeichert werden.

Variablen werden für jede CineCard getrennt im CineCore gespeichert und verwaltet. Damit wird sichergestellt, dass sich gleichzeitig ausgeführt Interrupt-Behandlungen verschiedener CineCards nicht unbeabsichtig beeinflussen. Gemeinsam genutzte Ressourcen zur Synchronisation der Interrupt-Behandlungen werden durch über *Host-Pointer* erreichbare Speicherbereiche des CineCore realisiert. Um möglicherweise durch gleichzeitigen Zugriff mehrerer Interrupt-Behandlungen entstehende Probleme wie Race-Conditions zu vermeiden, wird der Schreibzugriff auf *Host-Pointer* über die Atomic-Operations-API des Linux-Kernels realisiert. *Host-Pointer* können aus Sicherheitgründen nicht aus Skripten heraus erstellt oder gelöscht werden. Konfigurations- und Interrupt-Skripte können auf die im CineCore gespeicherten Variablen durch die folgenden Befehle zugreifen:

setvar: Erlaubt das Erzeugen bzw. Überschreiben einer CineCore-Variablen. Einmal erzeugt, bleibt die Variable erhalten, bis sie gelöscht wird oder das für die PCI-Karte zuständigen Backend entladen wird.

delvar: Entfernt eine Variable wieder aus dem CineCore. Dabei ist zu beachten, dass die Variable im Backend der angesprochenen CineCard entfernt wird, gleichnamige Variablen anderer Backends werden nicht angetastet.

getvar: Ermöglicht das Auslesen des Inhalts einer CineCore-Variablen aus dem Userspace heraus. Die Variableninhalte verschiedener CineCards eines Systems können sich unterscheiden.

eval: Mit der `eval`-Funktion können einfache Berechnungen oder Zuweisungen durchgeführt werden. Die Berechnung wird für jede CineCard getrennt durchgeführt.

Verzweigungen

Pro Aufruf wird über die Konfigurationsschnittstelle des Gerätetreibers jeweils ein Befehl übermittelt. Um bei Verzweigungen und Schleifen entscheiden zu können, ob ein Befehl ausgeführt werden darf oder nicht, ist es erforderlich, den Ausführungskontext des Befehls zu kennen.

Innerhalb eines im CineCore gespeicherten Interrupt-Skriptes ist dieser vollständig bekannt. Bei Befehlen aus einem Konfigurationsskript oder über die Benutzeroberfläche eingegebenen Befehlen, hat der Kerneltreiber keinen direkten

Abbildung 4.2: Beispiel einer Befehlsfolge für die Verzweigung

```
IF (TRUE)
   # auszuführende Befehle
   IF (TRUE)
          # auszuführende Befehle
   ELSE
          # zu ignorierende Befehle
          IF (TRUE)
               # wegen des umliegenden Blocks
               # auch zu ignorierende Befehle
          END
   END
   # diese Befehle sollten wieder ausgeführt werden
END
# auszuführende Befehle
IF (FALSE)
   # zu ignorierende Befehle
ELSE
   # auszuführende Befehle
END
```

Zugriff auf den Kontext der zu verarbeitenden Instruktion. Zum Ausführungszeitpunkt muss der Gerätetreiber jedoch wissen, ob die aktuell übergebene Anweisung auszuführen ist oder bedingt durch einen umgebenen Verzweigungsblock übersprungen werden sollte. Um dieses Problem zu lösen, sind beispielsweise die beiden folgenden Ansätze denkbar:

Blockspeicherung: Ein Block einer Verzweigung wird ähnlich wie die Interrupt-Skripte im Kernel gespeichert und nach Abschluss der Übertragung analysiert und ausgeführt. Der zum Ablegen der Befehle notwendige Speicherbedarf ist dabei nicht im Voraus bestimmbar. Durch diesen Ansatz mögliche Speicherüberläufe sind im Kernel-Kontext unbedingt zu meiden. Eine alternative Lösung mit einer statischen Größe des Befehls-Cache ist unzweckmässig, da das Festlegen einer maximalen Anzahl von Befehlen eine nicht hinnehmbare Einschränkung darstellt.

Stapelspeicher: Die Informationen, ob ein Programmblock ausgeführt werden soll oder nicht, wird je nach Verschachtelungstiefe in einem Stapelspeicher abgelegt. Die Größe des nötigen Speichers ist auch bei diesem Ansatz nicht im Voraus bestimmbar, wodurch die gleichen Probleme wie bei der Blockspeicherung entstehen.

Diese beiden Lösungsmöglichkeiten besitzen einen gemeinsamen Nachteil: Der benötigte Speicherplatz ist vor der Ausführung der Befehlsfolge nicht bekannt. Der erste Ansatz würde zudem die angestrebte Möglichkeit der Einzelschrittausführung verhindern. Nach Möglichkeit sollte jedoch eine Lösung mit konstantem Speicherverbrauch zum Einsatz kommen.

Abbildung 4.3: Verschachtelungstiefe und Ausführungsstatus bei der Verzweigung

```
Verschachtelungstiefe
|     Ignorieren?
|     |
v     v
0->1  0       IF (TRUE)
   1  0          # auszuführende Befehle
1->2  0          IF (TRUE)
   2  0                # auszuführende Befehle
2     0->1       ELSE
   2  1                # zu ignorierende Befehle
2->3  1                IF (TRUE)
   3  1                     # wegen des umliegenden Blocks
   3  1                     # auch zu ignorierende Befehle
3->2  1                END
2->1  1->0       END
   1  0          # diese Befehle sollten wieder ausgeführt werden
1->0  0       END
   0  0       # auszuführende Befehle
0->1  0->1    IF (FALSE)
   1  1          # zu ignorierende Befehle
1     1->0    ELSE
   1  0          # auszuführende Befehle
1->0  0       END
```

Die Umsetzung einer Lösung mit konstantem Speicherverbrauch wird durch die speziellen Anforderungen an einen Skript-Interpreter im Kernel-Kontext begünstigt. Für die Skriptverarbeitung im Kernel- und Interrupt-Kontext ist keine vollständige syntaktische Analyse notwendig. Es ist in diesem Fall ausreichend zu wissen, ob der aktuelle Skriptbefehl ausgeführt werden soll oder aufgrund des umliegenden Blockes übersprungen werden muss.

Bei der vorgestellten Implementation wird die Entscheidung, ob ein Befehl ausgeführt werden soll im Kernel-Kontext getroffen. Dies garantiert ein konsistentes Verhalten der Implementierung unabhängig davon, ob die jeweilige Anweisung aus einem im Kernel-Kontext laufenden Interrupt-Skript stammt oder vom Cineboxd aus dem Userspace heraus übermittelt wird.

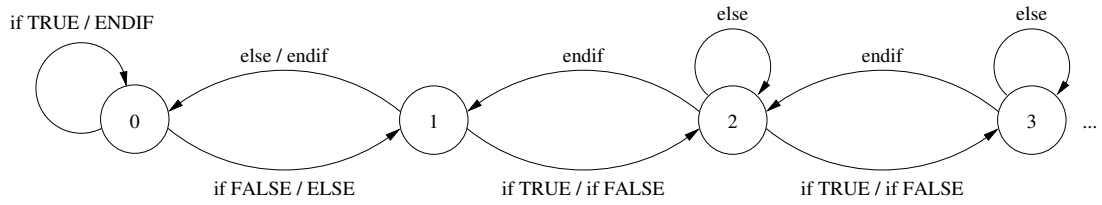
Zur korrekten Bestimmung des Ausführungsstatus müssen dem Interpreter nur zwei Informationen bekannt sein:

1. Müssen die Befehle des aktuellen oder eines übergeordneten Blockes ignoriert werden?

2. Wie hoch ist die Verschachtelungstiefe der zu überspringenden Blöcke?

Die zur Bestimmung des Ausführungsstatus notwendigen Informationen lassen sich in einer einzigen Variablen, dem Ausführungsstatus, zusammenfassen. Abbildung 4.4 verdeutlicht die Änderung des Ausführungsstatus beim Verarbeiten des Skript-Quelltextes.

Abbildung 4.4: Zustandsgraph des Ausführungsstatus



Die im Ausführungsstatus enthaltenen Informationen sind nicht ausreichend, um den aktuellen Zustand des Systemes zu jeder Zeit vollständig und richtig zu beschreiben. Im Rahmen der Interrupt-Behandlung ist es jedoch vollkommen ausreichend, wenn sich das System im Normalfall richtig verhält.

Fehler in der Befehlsstruktur der Verzweigung werden bei Skripten durch die im Userspace laufenden Teile des Lademechanismus abgefangen. Werden dennoch z.B. über die Netzwerkschnittstelle des Cineboxd fehlerhafte Anweisungsfolgen an den Gerätetreiber übermittelt, verhält sich der in Abbildung 4.4 gezeigte Algorithmus in vielen Fällen trotzdem richtig. Typische Vertreter solcher Fehler sind zum Beispiel redundante Abschlüsse eines Verzweigungsblockes (`END`). Im Interrupt-Kontext ist das Ignorieren solcher Fehler unproblematisch, darf jedoch in keinem Falle zu einem Absturz führen.

Bei der besprochenen Implementierung werden auch durch die Verzweigung nicht auszuführende Befehle über die Kernel-API an den Gerätetreiber übergeben. Dies führt zwar zu konstanten Durchlaufzeiten für die Konfigurationsskripte, weicht aber bei Verzweigungen mit sehr vielen Befehlen im nicht auszuführenden Block stark von dem zeitlichen Verhalten eines Interrupt-Skriptes ab, weil sehr

viel Rechenzeit durch die vielen Kontextwechsel zwischen Kernel- und Applikationsmodus verbraucht wird[1].

Durch die Verwaltung von Sprungmarken im Cineboxd lässt sich die Verarbeitungsgeschwindigkeit von Verzweigungsblöcken enorm steigern. Beim Aufruf des I/O-Controls für die `IF`-Anweisung wird dabei der aktuelle Ausführungsstatus zurückgeliefert. Ist dieser größer als Null, kann der folgende Block bis zum nächsten `ELSE` oder `END` vollständig übersprungen werden. Dieses Verfahren läßt sich jedoch nicht auf die sequentiell über die Netzwerkschnittstelle des Cineboxd eingelesenen Befehle anwenden, so dass diese Optimierung die zuvor beschriebene Implementation im Kernel nur ergänzen, aber nicht ersetzen kann.

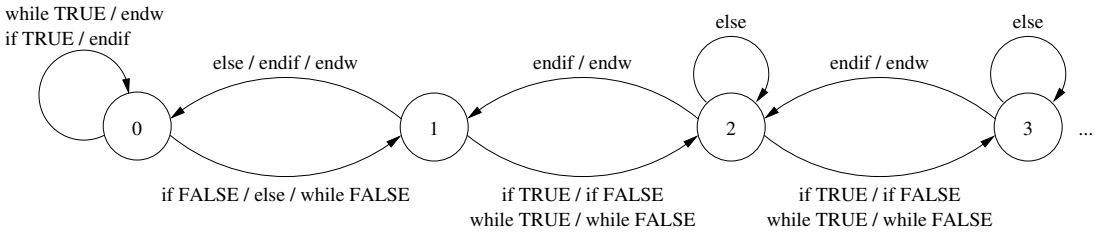
Schleifen

Ähnlich wie bei der Verzweigung ist eine für den Kernel-Kontext und Applikations-Kontext des Cineboxd konsistente Implementation der Schleifen sehr wünschenswert. Im Kernel-Kontext stellen Schleifen prinzipiell kein Problem dar. Durch die in den Kernel geladenen Interrupt-Skripte ist der Kontext der Schleife vollständig bekannt. Rücksprung- und Übersprungadressen können beim Laden eines Interrupt-Skriptes relativ einfach ermittelt und im Skript selbst abgespeichert werden.

Im Falle von Konfigurationsskripten und Befehlen, die über das Netzwerkprotokoll des Cineboxd übertragen werden, wird jedoch jeder Befehl einzeln an den Kernel übergeben. Nachfolgende Befehle sind zum Ausführungszeitpunkt nicht bekannt und vorherige sollten aus Gründen der Effizienz möglichst nicht gespeichert werden.

Dieses Problem läßt sich analog zur zuvor besprochenen Verzweigung durch eine Erweiterung des Ausführungsstatus auf While-Schleifen lösen:

Abbildung 4.5: Erweiterung des Zustandsgraphen um die While-Schleife



Die Entscheidung, ob ein von einer Schleifenanweisung eingeschlossener Befehlsblock ausgeführt oder übersprungen werden soll wird auch hier wieder im

[1] Bei aktuellen Linux-Kerneln verbrauch ein Kontextwechsel ca. 400000 Taktzyklen.

Kernel-Kontext getroffen. Der momentane Ausführungsstatus wird dem Cineboxd auch beim Aufruf des I/O-Controls der `WHILE`-Funktion zurückgegeben. Ist der Ausführungsstatus größer als Null, kann die gesamte Schleife von Cineboxd übersprungen werden. Die die Schleife abschließende `ENDW`-Anweisung löst einen Rücksprung auf die `WHILE`-Anweisung aus.

4.2.1 Implementationsaufwand

Die Implementation des auf Skripten basierenden *Konzeptes zur Interrupt-Behandlung* besteht aus den folgenden Teilbereichen:

1. Erweiterung der Kernel-API um die für die Interrupt-Verarbeitung benötigten Anweisungen.
2. Ausbau des Gerätetreibers zur Speicherung und Ausführung von Interrupt-Skripten.
3. Anpassung der Userspace-Applikationen zur Verarbeitung der neu eingeführten Befehle.
4. Schaffung einer Infrastruktur zur automatischen Erzeugung von Interrupt-Skripten.

Durch Wiederverwendung der bestehenden Konfigurations-API zur Übermittlung der Befehle der Interrupt-Skripte ist ein Test der Interrupt-Skripte aus dem im Userspace laufenden Cineboxd problemlos möglich. Das Verhalten der so ausgeführten Skripte entspricht (bis auf den Laufzeitaspekt) exakt dem der Interrupt-Service-Routine . Die Ausführung der Skriptbefehle im Einzelschrittmodus ist problemlos umsetzbar.

Die Speicherung der Skriptbefehle für die Interrupt-Verarbeitung läßt sich sehr einfach durch eine verkettete Liste der Funktionsaufrufe und deren Argumente realisieren. Die aufzurufenden Funktionen müssen dafür nicht modifiziert werden, womit die Implementation mit sehr wenig zusätzlichem Aufwand verbunden ist.

Die Anpassung des Cineboxd zur Verarbeitung der neu eingeführten Befehle gestaltet sich wegen der äußerst modularen Struktur der Software als sehr einfach. Die Befehle zum Laden- und Entladen der Interrupt-Skripte und die Befehle zur Manipulation der Variablen des CineCore lassen sich analog zu anderen schon vorhandenen Befehlen implementieren. Die Unterstüzung der Kontrollstrukturen erforderte jedoch einen leicht höheren Aufwand, da alle Befehle eines Konfigurations- oder Interrupt-Skriptes zunächst im Cineboxd zwischengespeichert und verlinkt werden müssen.

Besonders einfach gestaltet sich die Schaffung der Infrastruktur zur automatischen Erzeugung der Interrupt-Skripte. Da die Konfigurationsskripte ohnehin schon aus mehreren funktionell getrennten Teilen bestehen, reicht es aus, diese in einzelne Skriptdateien zu unterteilen. Ein Konfigurationsskript, das ein Interrupt-Skript in den Gerätetreiber läd, muss dabei nur die betreffenden Skriptdateien im `beginIntScript`/`endIntScript`-Block aufrufen.

Kapitel 5

Abschließende Betrachtung

Das Ziel dieser Diplomarbeit bestand in der Entwickung eines für den geplanten Einsatzzweck optimalen *Konzeptes zur Behandlung von Interupt-Request* der CineCard-Familie und dessen Implementierung. Das Ziel wurde in vollem Maße erreicht.

Eine der wichtigsten Aufgaben zur Bestimmung einer optimalen Lösung war das Finden geeigneter Bewertungskriterien und Maßstäbe für den Vergleich gefundener *Interrupt-Behandlungskonzepte*. Die verwendeten Kriterien ließen eine objektive und sinnvolle Gegenüberstellung der erarbeiteten Konzepte zu.

Bei der Erarbeitung der Bewertungmetrik stand eine Minimierung des Zeitaufwandes und der Kosten für das gesamte Projekt im Vordergrund. Der Vergleich möglicher Lösungansätze zeigte, dass nicht immer die naheliegensten Techniken auch eine optimale Problemlösung darstellen. Das gefundene *Interrupt-Behandlungskonzept* entspricht genau dem Anwendungs- und Anforderungsprofil des Heinrich-Hertz-Instituts.

Die anschließende Implementation des hier vorgestellten Interrupt-Behandlungskonzeptes ließ sich im Rahmen dieser Diplomarbeit abschließen. Der im Rahmen der Bewertungsmetrik abgeschätzte Arbeitsaufwand für die Anpassung der Software und die Implementierung entsprach dabei weitestgehend dem tatsächlichen Aufwand.

Die Verwendung einer Skriptsprache zur Interrupt-Behandlung ist in Gerätetreibern sehr ungewöhnlich, hat sich aber als sehr flexibel und leistungsfähig herausgestellt. Anfängliche Befürchtungen wegen des erwarteten hohen Behandlungs-Overheads einer Skriptlösung haben sich als haltlos herausgestellt.

Die implementierte Lösung hat sich als flexibel und voll einsatzfähig herausgestellt und befindet sich nun im produktiven Einsatz. Das Laufzeitverhalten und der erreichte Funktionsumfang übertrifft die anfänglichen Erwartungen bei Wei-

Abbildung 5.1: produktiver Einsatz auf der IFA 2006 in Berlin - Platzhalter

tem. In Verbindung mit dem in der Studienarbeit zu CineCard[1] entwickelten Gerätetreiber ließ sich der gleichzeitige Betrieb von bis zu vier CineCards selbst in einem Host-System der untersten Leistungsklasse realisieren.

Die nahtlose Integration des gefundenen *Konzeptes* in das bestehende Umfeld aus CineCard-Software und Konfigurationsskripten ist wie erwartet gelungen. Bestehende Skripte zu Konfiguration der CineCards lassen sich so in Teilen direkt für die Interrupt-Behandlung einsetzen. Dies trug erheblich zur Minderung des Migrationsaufwandes auf die neu eingesetzten Techniken bei.

Das durch die skriptbasierte Interrupt-Behandlung eingeführte Variablenkonzept stellt einen erheblichen Mehrwert für die verwendeten Konfigurationsskripte dar. Darüber hinaus können durch diese flexible Kommunikationsmöglichkeit die Konfiguration der PCI-Karte und die Interrupt-Behandlung optimal aufeinander angepasst werden.

Index

Glossar

Cinebox Als Cinebox wird ein mit den CineCards bestücktes Host=System bezeichnet, welches vom Cineboxd verwaltet wird.

Cineboxd Der Cinebox=Management=Daemon, das zentrale Verwaltungs- und Steuerprogramm einer Cinebox.

CineCore Zentrale Verwaltungsinstanz des Gerätetreibers der PCI-Karten der CineCard-Familie.

CineCore Zentraler Teil des CineCard-Gerätetreibers. Das CineCore verwaltet alle in einem Host-System betriebenen CineCards und stellt eine einheitliche Schnittstelle zum Userspace zur Verfügung.

Exception Ein auch Ausnahme-Fehler genannter, synchroner Interrupt. Exceptions werden von der CPU selbst ausgelöst und sind für die Behandlung von Fehlern des laufenden Prozessen bestimmt.

HiPEG+ HDTV MPEG2-Dekoder mit integrierter 32-Bit RISC-Prozessoreinheit. Der HiPEG+ bildet die Basis der aktuellen CineCard-Generation.

Host − System Als Host-System wird das Computersystem bezeichnet, das die HHI-CineCards enthält.

I/O − Control Methode zur Kommunikation mit dem Gerätetreiber über eine Gerätedatei. I/O-Controls sind flexibler als Lese- und Schreibzugriffe, erfordern aber die Definition einer gemeinsam genutzten API.

Interrupt − Behandlungskonzept Prinizipielle technische Vorgehensweise zur Reaktion auf einen aufgetretenen Interrupt=Request

Interrupt − Handler Stellt die Quelle des Interrupt-Requests fest und ruft die betreffende Interrupt Service Routine (ISR) auf.

Interrupt − Latenz (v. lat.: latens = verborgen) Das Zeitintervall vom Auftreten eines Interrupt-Requests bis zum Beginn der Behandlung in der ISR.

Interrupt – Request Ein Signal, meist von einem externen Gerät, mit dem der Prozessor aufgefordert wird, den aktuellen Programmablauf anzuhalten (zu unterbrechen) und eine zuvor definierte Interruptbehandlungsroutine aufzurufen.

Interrupt – Strategie Auf einem Konzept zur Interrupt=Behandlung basierende, konkrete Umsetzung in einer Implementation. Die Strategie kann sich dabei je nach Situation und Implementierung (zum Beispiel durch Parametrisierung der Interrupt=Routine) ändern.

ISR Interrupt Service Routine: In dieser Funktion wird der aufgetretene Interrupt durch Zugriff auf die Hardware behandelt.

SXGA+ Super Extended Graphics Array (+), die Abkürzung steht für ein Auflösungsvermögen von 1400x1050 Bildpunkten (1,5 Megapixel).

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Robert Sperling: Linux-Hardwaretreiber für die HHI CineCard-Familie;
http://sar.informatik.hu-berlin.de

[2] Gordon Kunkel/Robert Sperling: Cineboxd Socket Protokoll
Auf Anfrage über: Christian Weissig

[3] HiPEG+ Brief Product Info;
MikroM GmbH, http://www.mikrom.de

[4] Informationen über IBM's Stretch Computer von Eric Smith;
http://www.brouhaha.com/∼eric

[5] Montenegro, S.; Holzky, F.: Fehlerursache Komplexität;
Themenheft, Design und Elektronik, Weka Fachzeitschrifen 2004

[6] Wolfgang Mauerer: Linux Kernelarchitektur;
Carl Hanser Verlag München Wien, 2004, ISBN 3-446-22566-8

[7] Eva-Katharina Kunst, Jürgen Quade: Linux-Treiber entwickeln;
dpunkt.verlag GmbH, 2004, ISBN 3-89864-238-0

[8] Kern-Technik, Folge 1-20;
LINUX Magazin 2003-2005

[9] Linux Weekly News: API changes in the 2.6 kernel series;
http://www.lwn.net/Articles/2.6-kernel-api/

[10] Mailing-Liste der Kernelentwickler;
http://www.kernel.org

Reports published by Humboldt University Berlin, Computer Science Department, Systems Architecture Group.

1. SAR-PR-2005-01: Linux-Hardwaretreiber für die HHI CineCard-Familie. Robert Sperling. 37 Seiten.
2. SAR-PR-2005-02, NLE-PR-2005-59: State-of-the-Art in Self-Organizing Platforms and Corresponding Security Considerations. Jens-Peter Redlich, Wolf Müller. 10 pages.
3. SAR-PR-2005-03: Hacking the Netgear wgt634u. Jens-Peter Redlich, Anatolij Zubow, Wolf Müller, Mathias Jeschke, Jens Müller. 16 pages.
4. SAR-PR-2005-04: Sicherheit in selbstorganisierenden drahtlosen Netzen. Ein Überblick über typische Fragestellungen und Lösungsansätze. Torsten Dänicke. 48 Seiten.
5. SAR-PR-2005-05: Multi Channel Opportunistic Routing in Multi-Hop Wireless Networks using a Single Transceiver. Jens-Peter Redlich, Anatolij Zubow, Jens Müller. 13 pages.
6. SAR-PR-2005-06, NLE-PR-2005-81: Access Control for off-line Beamer – An Example for Secure PAN and FMC. Jens-Peter Redlich, Wolf Müller. 18 pages.
7. SAR-PR-2005-07: Software Distribution Platform for Ad-Hoc Wireless Mesh Networks. Jens-Peter Redlich, Bernhard Wiedemann. 10 pages.
8. SAR-PR-2005-08, NLE-PR-2005-106: Access Control for off-line Beamer Demo Description. Jens Peter Redlich, Wolf Müller, Henryk Plötz, Martin Stigge. 18 pages.
9. SAR-PR-2006-01: Development of a Software Distribution Platform for the Berlin Roof Net (Diplomarbeit / Masters Thesis). Bernhard Wiedemann. 73 pages.
10. SAR-PR-2006-02: Multi-Channel Link-level Measurements in 802.11 Mesh Networks. Mathias Kurth, Anatolij Zubow, Jens Peter Redlich. 15 pages.
11. SAR-PR-2006-03, NLE-PR-2006-22: Architecture Proposal for Anonymous Reputation Management for File Sharing (ARM4FS). Jens Peter Redlich, Wolf Müller, Henryk Plötz, Martin Stigge, Torsten Dänicke. 20 pages.
12. SAR-PR-2006-04: Self-Replication in J2me Midlets. Henryk Plötz, Martin Stigge, Wolf Müller, Jens-Peter Redlich. 13 pages.
13. SAR-PR-2006-05: Reversing CRC – Theory and Practice. Martin Stigge, Henryk Plötz, Wolf Müller, Jens-Peter Redlich. 24 pages.
14. SAR-PR-2006-06: Heat Waves, Urban Climate and Human Health. W. Endlicher, G. Jendritzky, J. Fischer, J.-P. Redlich. In: Kraas, F., Th. Krafft & Wang Wuyi (Eds.): Global Change, Urbanisation and Health. Beijing, Chinese Meteorological Press.
15. SAR-PR-2006-07: 无线传感器网络研究新进展(State of the Art in Wireless Sensor Networks). 李刚(Li Gang), 伊恩斯•彼得•瑞德里希 (Jens Peter Redlich).
16. SAR-PR-2006-08, NLE-PR-2006-58: Detailed Design: Anonymous Reputation Management for File Sharing (ARM4FS). Jens-Peter Redlich, Wolf Müller, Henryk Plötz, Martin Stigge, Christian Carstensen, Torsten Dänicke. 16 pages.
17. SAR-PR-2006-09, NLE-SR-2006-66: Mobile Social Networking Services Market Trends and Technologies. Anett Schülke, Miquel Martin, Jens-Peter Redlich, Wolf Müller. 37 pages.
18. SAR-PR-2006-10: Self-Organization in Community Mesh Networks: The Berlin RoofNet. Robert Sombrutzki, Anatolij Zubow, Mathias Kurth, Jens-Peter Redlich, 11 pages.
19. SAR-PR-2006-11: Multi-Channel Opportunistic Routing in Multi-Hop Wireless Networks. Anatolij Zubow, Mathias Kurth, Jens-Peter Redlich, 20 pages.
20. SAR-PR-2006-12, NLE-PR-2006-95: Demonstration: Anonymous Reputation Management for File Sharing (ARM4FS). Jens-Peter Redlich, Wolf Müller, Henryk Plötz, Christian Carstensen, Torsten Dänicke, 23 pages.
21. SAR-PR-2006-13: Interrupt-Behandlungskonzepte für die HHI CineCard-Familie, Robert Sperling, 72 Seiten