SAPIENZA
UNIVERSITÀ DI ROMA

Da definire

Facoltà di Ingegneria dell'informazione, informatica e statistica
Corso di Laurea in Informatica

Leonardo Ganzaroli
Matricola 1961846

Responsabile
Prof. Luigi Cinque

Corresponsabile
Dr. Diego Bellani

Anno Accademico 2024/2025

Da definire
Relazione di tirocinio - Sapienza Università di Roma

Questa relazione è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: ganzaroli.leonardo@gmail.com

Sommario

write your abstract here
Scaletta:

- Introduzione
  - Acceleratori hw, intel 8087, cenni storici
  - Esplosione degli stessi e cause
  - PCI, PCIe (cenni) causa della popolarità
  - Obiettivo, disaccopiamento hw e sw
- teoria PCIe
- Linux, driver
- QEMU, QOM, limiti
- Processo di implementazione
- Conclusioni (riassunto)

Indice

Capitolo 1

Introduzione

Questo capitolo fornisce una panoramica sugli acceleratori hardware e sulla loro diffusione nella sezione 1.1, passa poi all'obiettivo finale del tirocinio con le relative motivazioni nella sezione 1.2.

1.1 Acceleratori hardware

Gli acceleratori hardware sono dei componenti hardware progettati appositamente per svolgere determinate funzioni, essi permettono di svolgere quelle funzioni in modo più efficiente rispetto alla classica combinazione di software e CPU. I possibili campi d'applicazione sono molteplici ma alcuni mostrano chiaramente il vantaggio di delegare i compiti, per esempio:

- In crittografia sono spesso richiesti dei calcoli particolarmente intensivi, evitare di svolgerli nella CPU comporta una notevole riduzione del carico sulla stessa
- L'elaborazione grafica richiede molti calcoli che possono essere ottimizzati svolgendoli in parallelo, data la sua natura sequenziale la CPU non risulta la scelta migliore
- L'elaborazione dei segnali audio deve essere svolta in tempo reale e potrebbe richiedere delle operazioni aggiuntive, la CPU potrebbe non riuscire a gestire il tutto senza ritardi

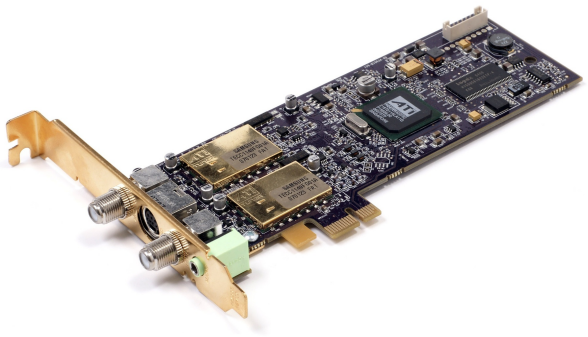

Figura 1.1. TV Wonder HD 650 di ATI Technologies

1.1.1 Evoluzione nel tempo

I primi acceleratori nacquero alla fine degli anni '70 / inizio anni '80 sotto forma di semplici coprocessori matematici atti a svolgere calcoli aritmetici complessi, con l'avanzare del tempo e conseguentemente della tecnologia sono stati sviluppati acceleratori sempre più vari e potenti. Come si vedrà in 1.1.2 gli acceleratori più diffusi in questo periodo sono quelli che si occupano di grafica e quelli relativi all'IA, per dare un'idea dell'evoluzione avvenuta seguono 3 esempi provenienti da anni diversi:

1. **Intel 8087** (1980)

   Si tratta del primo coprocessore matematico progettato per i processori 8086 e 8088, il suo scopo principale è quello di accelerare i calcoli in virgola mobile ma può anche calcolare funzioni come quelle trigonometriche ed i logaritmi [1].

2. **Sun Crypto Accelerator 1000** (2002)

   Lo scopo di questo acceleratore è quello di effettuare la computazione di vari algoritmi crittografici usati principalmente nei protocolli di sicurezza in ambito e-commerce. Essendo creato solamente per svolgere calcoli non presenta alcuna interfaccia esterna tranne quella PCI per il collegamento all'host [2].

3. **Pixel Visual Core** (2017)

   Presente negli smartphone Pixel 2 e Pixel 2 XL di Google, è un coprocessore che si occupa dell'elaborazione fotografica. Oltre al miglioramento delle foto permette di eseguire applicazioni di Machine Learning come il riconoscimento dei volti [3].

Con il tempo anche l'architettura generale degli elaboratori ha subito un'evoluzione, l'8087 aggiungeva 68 istruzioni al set del 8086 ed era collegato in parallelo ad esso, quando arrivava una sua istruzione il coprocessore prendeva il controllo e ad operazione avvenuta mandava un segnale al processore per farlo ripartire [1].

Risulta evidente che questo metodo comporta uno spreco della CPU, infatti ogni volta che deve essere eseguita anche una semplice operazione come $0.1 + 0.2$ il coprocessore (anche se per un tempo irrisorio) mantiene il controllo e non permette alla CPU di fare altro.

Nei sistemi odierni sarebbe impensabile usare questo metodo, considerando la mole di dati comunemente interessati in un'operazione si perderebbe una grande quantità di tempo. Per risolvere il problema si usa l'accesso diretto alla memoria, viene inserito un apposito chip detto *Controllore* contenente 4 registri:

- Il primo contiene l'indirizzo di partenza in memoria
- Il secondo il numero di byte interessati
- Il terzo l'indirizzo del dispositivo o dello spazio I/O interessato
- Il quarto indica se l'operazione è di lettura o scrittura

Così facendo si interpella la CPU solo all'inizio per settare i registri del controllore e alla fine inviando un apposito segnale, anche se in questo modo la CPU è libera di svolgere altri compiti non può comunque accedere al BUS perché lo scambio di dati ha priorità più alta. Ciò nonostante il guadagno ottenuto supera di gran lunga quest'ultimo problema [4].

1.1.2 Popolarità

Come visto nella sezione precedente l'uso degli acceleratori porta evidenti vantaggi, questo potrebbe portare a domandarsi quanto siano diffusi e se possano aver contribuito allo sviluppo di altre aree tecnologiche.

Alla prima parte si può rispondere con "parecchio", nel 2023 il mercato globale degli acceleratori era valutato a 2.87 miliardi di dollari e si stima che nel 2033 raggiungerà 177 miliardi [5].

Alla seconda parte si può rispondere nel medesimo modo, prendendo come esempio gli acceleratori grafici si scopre che non solo si sono evoluti per soddisfare richieste sempre maggiori ma vengono anche usati per scopi diversi da quelli originali, rispettivamente:

- La continua ricerca da parte del mondo videoludico di una grafica sempre più realistica è stata fin'ora una delle motivazioni principali per lo sviluppo degli acceleratori grafici [6].
- Le GPU presenti negli acceleratori sono ormai lo strumento standard usato nella fase di training delle reti neurali, un'altra applicazione non trascurabile è il mining delle criptovalute [7].

1.1.3 Interconnessione all'host

Per concludere questa panoramica sugli acceleratori resta solamente da vedere il metodo di connessione fisico con l'host. Nella definizione vista in 1.1 non si accenna a questo concetto perché non esiste un metodo universale ma invece ce ne sono svariati, ognuno con i propri pro e contro, ma cosa più importante ci sono metodi più diffusi di altri. A seguire una breve descrizione di alcuni metodi [8]:

- **Bump-in-the-wire**

  Inserimento tra due componenti esistenti tramite opportuni collegamenti.

- **PCIe**

  Approfondito nel capitolo 2.

- **Integrated on-chip**

  Inserimento in un *System on a chip*, a sua volta collegabile in diversi modi.

- **USB**

  Essendo il metodo più diffuso per collegare periferiche può essere usato anche in questo caso.

PCIe rimane il metodo più usato, questo è dovuto sia al fatto che si tratta della naturale evoluzione di PCI (già famoso) sia delle sue ottime caratteristiche riguardanti: retrocompatibilità, scalabilità, velocità di trasferimento, larghezza di banda e gestione degli errori [9].

1.2 Obiettivo del tirocinio

Per spiegare l'obiettivo di questo tirocinio va introdotto il principale problema riguardante gli acceleratori, il tempo. La progettazione ed il successivo sviluppo richiedono mesi, se non addirittura anni, nel caso di un sistema di tipo *Asic* ci vogliono mediamente dai 9 ai 18 mesi [10]. A tutto ciò va aggiunto il processo di sviluppo del lato software che può iniziare solamente quando è pronto almeno un prototipo del prodotto finale portando così ad ulteriori ritardi.

L'obiettivo finale è quello di capire se tramite opportuni strumenti di emulazione si possa creare una versione virtuale di un acceleratore hardware e fino a che punto possa essere fedele alla controparte fisica, in questo modo sarà possibile disaccoppiare lo sviluppo hardware da quello software riducendo i tempi di sviluppo complessivi. Verrà inoltre prodotto un prototipo di acceleratore con connessione PCIe, il relativo Driver per Linux ed una libreria utilizzabile dall'utente ed il tutto sarà emulato tramite QEMU.

I capitoli 2,3 e 4 forniscono le basi teoriche necessarie e danno una panoramica sugli strumenti usati, il capitolo 5 mostra i punti principali del processo di implementazione, ed il capitolo 6 tira le conclusioni sul tirocinio sostenuto.

Capitolo 2

PCIe

Questo capitolo fornisce una visione generale sullo standard PCIe, la sezione 2.1 è introduttiva mentre il resto del capitolo fornisce la teoria essenziale.

2.1 Concetti di base

PCI express è un'interfaccia di I/O ad alte prestazioni e ad uso generale, progettata per una vasta gamma di piattaforme informatiche e diretto successore di PCI e delle sue varianti PCI-X e CompactPCI. Mantiene alcune delle caratteristiche del suo predecessore ma cambia totalmente topologia [11].

È bene notare che questa interfaccia permette di connettere una vasta gamma di dispositivi tra cui: adattatori per la connessione al Wi-Fi e schede di espansione che aggiungono porte fisiche. La connessione avviene tramite gli slot fisici comunemente presenti sulle schede madri dei computer fissi, ma PCIe è usato anche in altri standard e interfacce che ne estendono il campo d'azione come:

- La scheda ExpressCard per i computer portatili [12].
- I connettori U.2 e M.2 per i dispositivi di archiviazione.

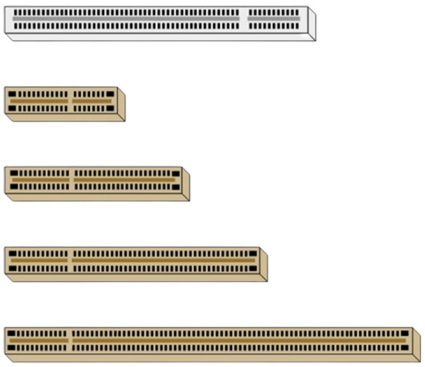

Figura 2.1. Rappresentazione degli slot fisici, dall'alto: PCI, PCIe (x1, x4, x8, x16)

Prima di passare alle specifiche è opportuno aprire una parentesi sull'evoluzione di questo standard, dall'anno di creazione fino ad oggi si sono succedute ben 7 generazioni (con un ottava programmata) e l'elemento migliorato maggiormente è il trasferimento di dati, passando da 2.5 GT/s a 128 GT/s. A partire dalla sesta generazione c'è stato anche un cambio della codifica di linea, si è passati da NRZ a PAM-4 + FEC [11].

2.2 Topologia

Come accennato nella sezione precedente la topologia di PCIe differisce da quella del suo predecessore, PCI ha una struttura basata su bus parallelo mentre PCIe è seriale e presenta delle somiglianze con le reti di computer. In particolare usa una forma di comunicazione a pacchetti ed è possibile dividere la sua architettura in 3 livelli logici che presentano delle similitudini con i livelli inferiori del modello OSI [11].

Livello PCIe	Descrizione	Livello OSI
Transazione	Gestione dei pacchetti e instradamento	Rete + Trasporto
Data Link	Controllo di integrità dei pacchetti	Collegamento
Fisico	Trasmissione fisica dei dati	Fisico

Tabella 2.1. Descrizione dei livelli e similitudini con livelli OSI

Questa somiglianza non si ferma ai soli livelli, risulta ancora più evidente una volta visti gli elementi di base presenti:

- **Lane**

  Una lane è formata da 2 coppie differenziali, ossia 4 tracce (2 per trasmettere e 2 per ricevere).

- **Link**

  Un semplice canale di comunicazione di tipo dual-simplex tra due componenti, deve contenere almeno una lane.

- **Root Complex**

  Mette in comunicazione CPU e memoria con una gerarchia di dispositivi I/O di cui è radice.

- **Endpoint**

  Tipo di funzione che può iniziare o completare transazioni PCIe, può farlo per conto proprio o per un dispositivo esterno. Si distinguono in:

  - **Legacy**
    Compatibile con le vecchie specifiche PCI e software legacy.
  - **Nativo PCIe**
  - **RCiEP**
    Integrato nel Root Complex, non ha un collegamento fisico e non compare nella gerarchia.

- **Switch**

  Insieme di Bridge PCI-to-PCI virtuali.

- **Bridge PCIe to PCI/PCI-X**

  Permette di collegare una gerarchia di dispositivi PCI/PCI-X a quella PCIe.

- **Ripetitore**

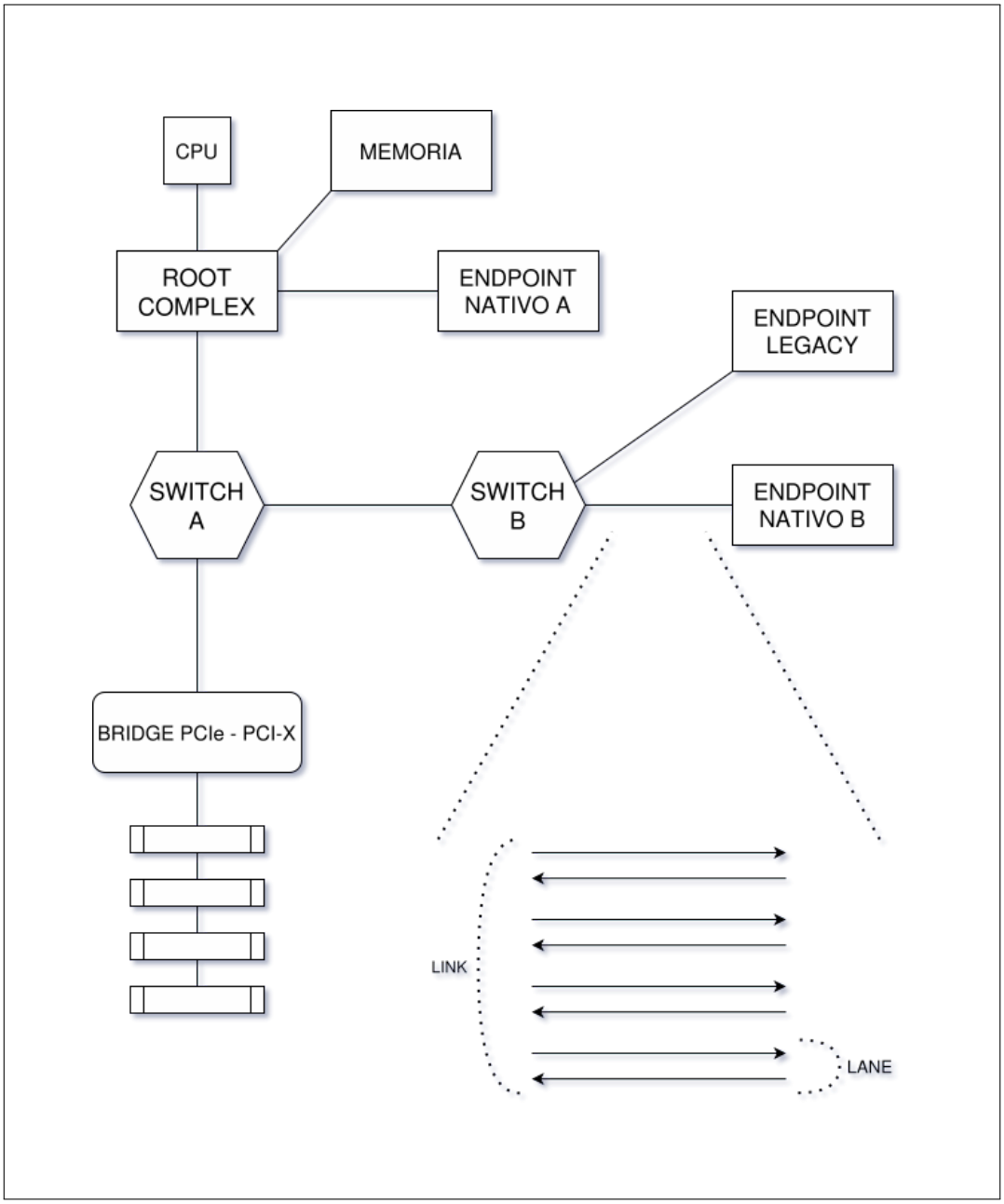

Figura 2.2. Esempio di topologia

In questo caso il link preso in esame viene definito x4 per la presenza di 4 lane, si può notare dalla figura 2.1 che in base al numero di lane cambia il numero di pin sullo slot. Secondo lo standard non sono supportati numeri di lane arbitrari ma solamente: 1, 2, 4, 8, 12, 16, 32 [11].

2.3 Link Training ed enumerazione

Il momento più importante è quello di inizializzazione dei Link (Link Training, visibile nella figura 2.3), si verifica quando tutti i dispositivi sono alimentati ed hanno ricevuto un clock di riferimento [13]. Si può dividere in 4 fasi:

1. **Detect**

   Ogni dispositivo attiva su ogni sua lane un circuito specifico che gli permette di capire se è collegato ad un altro dispositivo, in caso affermativo inizia ad inviare dati.

2. **Polling**

   Il Root Complex e gli Endpoint trasmettono un insieme di dati predefiniti, questa parte termina quando ogni dispositivo riesce ad interpretare i dati ricevuti e rispondere di conseguenza.

3. **Configuration**

   Viene stabilita la grandezza dei Link, si assegna un numero sia ai Link che alle Lane ed avviene l'associazione delle seconde alle prime.

4. **Link Equalization** (Opzionale)

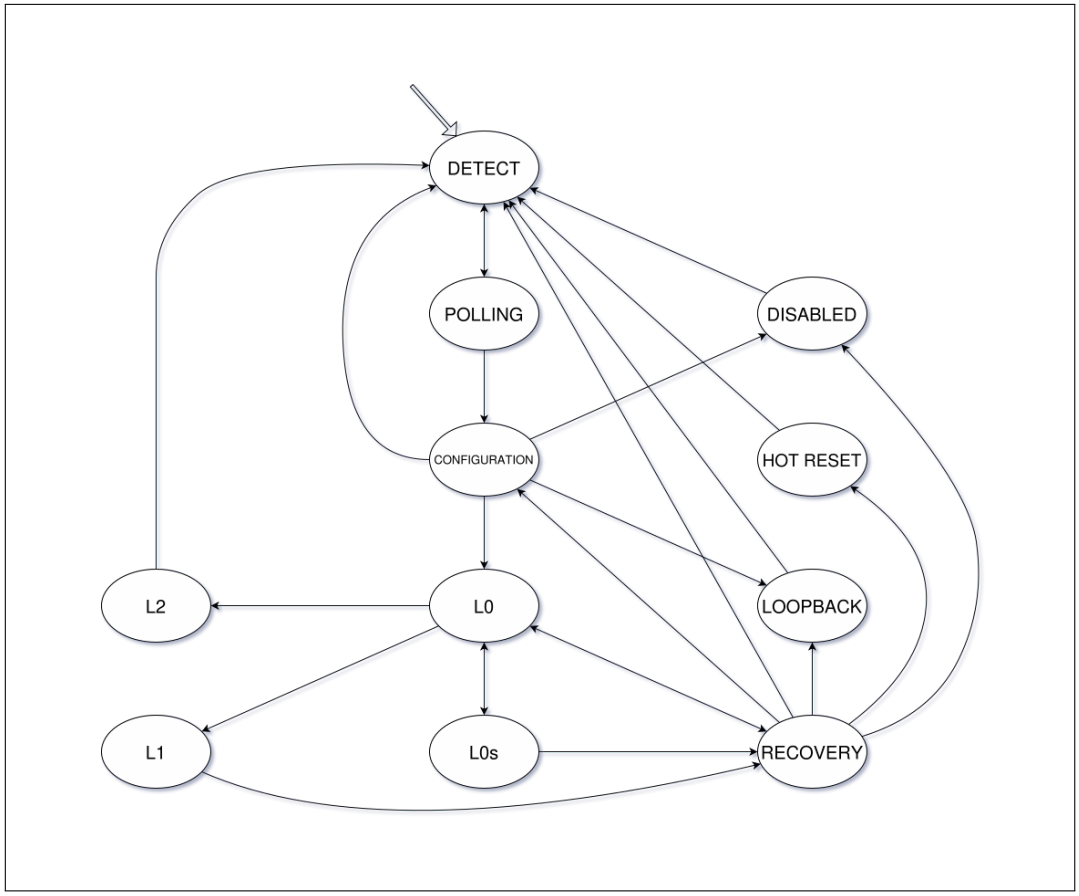

Figura 2.3. Link Training and Status State Machine

Nella figura 2.3 sono presenti anche i vari stati in cui si possono trovare i Link dopo la fine del processo visto prima, nel dettaglio:

- **L0** rappresenta il normale funzionamento
- **L0s**, **L1** e **L2** sono associati a diversi livelli di risparmio energetico
- **Loopback** è usato per la diagnostica
- **Disable** indica che il Link non è abilitato
- **Hot Reset** serve per eseguire un reset logico
- **Recovery** = Ripristino

Immediatamente dopo il Link Training avviene l'enumerazione dei dispositivi, ad ognuno viene assegnato un codice identificativo con forma:

DOMINIO	NUMERO BUS	NUMERO DISPOSITIVO	NUMERO FUNZIONE
2 byte	1 byte	5 bit	3 bit

Il numero di Bus in questo caso viene fornito da una divisione in Bus logici della topologia, così facendo è possibile semplificare l'instradamento dei pacchetti ed allo stesso tempo avere retrocompatibilità con PCI. Il numero di funzione è presente per permettere ad un dispositivo di esporre più funzioni logiche distinte. Il dominio ha una funzione solo in presenza di Root Complex multipli, permette di distinguere le varie gerarchie presenti nel sistema [11].

Questo processo avviene in modo semplice, vedendo il tutto come se fosse un grafo basta svolgere una visita in profondità ricorsiva partendo dal Root Complex e tenendo nota del livello in cui ci si trova.

Nella figura 2.2 una possibile enumerazione (ignorando le funzioni) può essere:

Elemento	Numero Bus	Numero Dispositivo
Root Complex	0	0
Endpoint A	0	1
Switch A	0	2
Switch B	1	0
Bridge	1	1
Endpoint B	2	0
Endpoint Legacy	2	1
Dispositivo PCI-X 1	3	0
Dispositivo PCI-X 2	3	1
...	...	...

2.4 Configurazione tramite software

L'enumerazione da sola non è però sufficiente per usare i dispositivi, ognuno di essi deve avere dei registri che contengano le informazioni utili per la configurazione, l'inizializzazione e l'associazione ai Driver giusti. Il loro insieme viene definito Spazio di Configurazione, nel caso PCIe questo spazio equivale a 4096 Byte e va ad estendere il vecchio spazio PCI. A differenza di PCI questo spazio esteso viene mappato direttamente nella memoria e permette di accedere ai singoli registri tramite opportune API fornite dal Sistema Operativo [11]. Ogni funzione di ogni dispositivo ne ha uno.

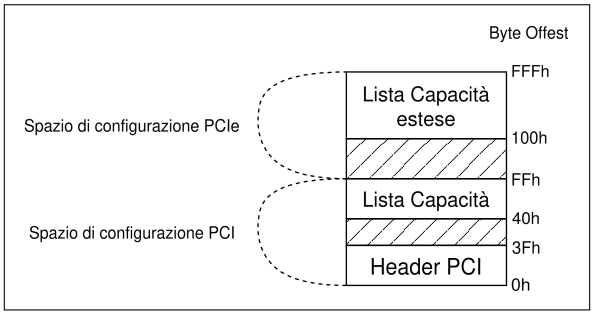

Figura 2.4. Layout dello spazio

Resta comunque disponibile il vecchio metodo di accesso usato da PCI che permette di accedere solo ai primi 256 Byte, ossia allo spazio non esteso. Come visto nella figura 2.4 le sezioni principali sono le 2 liste delle capacità e l'Header PCI[1], quest'ultimo è quello che contiene tutte le informazioni sul dispositivo. Sono presenti 2 tipi di Header ma quello di nostro interesse è il tipo 0 che viene usato per gli Endpoint.

31-24	23-16	15-8	7-0	Bit
ID Dispositivo		ID Venditore		
Stato		Comando		
Codice di Classe			ID Revisione	
BIST	Tipo di Header	Timer di latenza	Dimensione linea Cache	
BARs				
Puntatore a CIS per CardBus				
ID Sottosistema		ID Venditore Sottosistema		
Indirizzo base ROM di espansione				
			Puntatore alle capacità	
Latenza massima	Tempo minimo di accesso al bus	Pin di interrupt	Linea di interrupt	

Tabella 2.2. Header PCI di tipo 0, i campi in blu sono in comune con l'altro tipo

[1] Alcuni bit e registri vengono impostati Read-only e con valore 0 perché associati a funzionalità obsolete non più supportate in PCIe [11].

Breve descrizione dei registri:

- **ID Dispositivo**, **ID Venditore**, **ID Sottosistema**, **ID Venditore Sottosistema**, **ID Revisione**

  Identificano nel dettaglio il dispositivo.

- **Comando**

  Controlla varie funzionalità del dispositivo.

- **Stato**

  Contiene informazioni sullo stato del dispositivo e su eventuali errori.

- **Codice di Classe**

  Contiene Classe, Sottoclasse e Interfaccia del dispositivo, insieme identificano la funzione associata allo Spazio.

- **BIST**

  Usato per auto-diagnostica interna.

- **Tipo di Header**

  Indica il tipo di Header e se il dispositivo è multifunzione.

- **Timer di latenza**

  Obsoleto.

- **Dimensione linea Cache**

  Presente solo per retrocompatibilità.

- **Puntatore a CIS per CardBus**

  Obsoleto.

- **Indirizzo base ROM di espansione**

  Gestisce un'eventuale ROM di espansione locale.

- **Puntatore alle capacità**

  Punta alla Linked List delle capacità.

- **Latenza massima** e **Tempo minimo di accesso al bus**

  Obsoleti.

- **Pin e Linea di interrupt**

  Usati solo per interrupt Legacy.

- **BARs**

  Indicano al Sistema Operativo quante regioni di memoria e/o I/O (con relativa dimensione) devono essere allocate in memoria per consentire il corretto funzionamento del dispositivo. L'Header di tipo 0 può contenerne al massimo 6, quello di tipo 1 al massimo 2 [11].

2.4.1 *Capabilites*

Le Capacità (*Capabilities*) sono delle estensioni che permettono di aggiungere funzionalità ad un dispositivo PCI senza andare a modificare la struttura base dello Spazio di Configurazione. Sono organizzate in una lista che generalmente parte dall'offset visto nella figura 2.4, ogni nodo contiene sempre un ID ed un puntatore al nodo successivo e in alcuni casi anche dei registri aggiuntivi [14].

Quelle estese invece sono prerogativa dei dispositivi PCIe, hanno struttura simile a quelle normali ma offrono generalmente delle funzionalità più complesse [11]. Anch'esse partono in genere dall'offset visto nella figura 2.4.

In ogni caso i dispositivi PCIe devono sempre avere le Capacità *Power Management* ed *Express* [11].

Bit	0-7	8-15	16-19	20-31	...
Normali	ID	Puntatore al nodo successivo	...		
Estese	ID		Versione	Offset della Capacità successiva	...

Tabella 2.3. Struttura dei nodi

Tra tutte quelle disponibili ne spiccano alcune:

- **Express**

  Indica che il dispositivo è PCI express.

- **Power Management**

  Fornisce un'interfaccia con cui è possibile controllare il consumo elettrico del dispositivo [15].

- **MSI**

  Permette al dispositivo di generare un interrupt scrivendo un certo valore nell'indirizzo fornito dal Sistema Operativo [14].

- **MSI-X**

  Estensione di MSI.

Anche per quelle estese vale lo stesso concetto:

- **Advanced Error Reporting**

  Aggiunge controlli degli errori avanzati.

- **Device Serial Number**

  Assegna al dispositivo un numero di serie univoco.

- **Power Budgeting**

  Permette al dispositivo di dichiarare il proprio consumo elettrico e al sistema di allocare in modo dinamico la potenza erogata a *runtime*.

- **Resizable BAR**

  Consente la negoziazione tra dispositivo e Sistema Operativo riguardante la dimensione della memoria mappata.

[1] Entrambi i tipi devono avere i nodi allineati a 4 byte, per quelle normali si fa ciò settando i 2 bit meno significativi di ogni puntatore a 0 [11][14].

Capitolo 3

Linux

3.1 Gestione dei driver

Capitolo 4

QEMU

4.1 QOM

4.2 Limitazioni

Capitolo 5

Implementazione

Capitolo 6

Conclusioni

The grasping power of the mirror..

Note esplicative

Qui viene data una breve spiegazione di alcuni argomenti presenti in questo documento che il lettore potrebbe non conoscere.

- Con System on a chip si intende un circuito integrato che contiene alcuni/tutti gli elementi principali di un computer o sistema elettronico in un solo chip
- T/s è un'unita di misura informale che indica il numero di operazioni di trasferimento dati effettuate al secondo, i multipli principali sono MT/s (10^6) e GT/s (10^9)
- La codifica di linea è il processo di adattamento di un segnale al mezzo di comunicazione usato
- Non Return to Zero (NRZ) è una codifica di linea in cui i valori 0 e 1 vengono rappresentati da due livelli distinti di tensione senza ritorno allo stato neutro tra un bit e l'altro [16]
- La Pulse-amplitude modulation (PAM) è una modulazione in cui l'ampiezza della portante (impulsiva) varia in base in base all'ampiezza del segnale modulante (che può essere sia analogico che digitale), la PAM-4 usata da PCIe usa una modulante digitale ed il segnale modulato presenta solo 4 possibili livelli che vengono associati alle 4 possibili combinazioni ottenute da 2 bit [16]
- La Forward Error Correction (FEC) è una tecnica usata per rilevare e correggere errori nelle comunicazioni digitali
- Una coppia differenziale è composta da 2 tracce affiancate, esse portano un segnale con stessa ampiezza ma polarità opposta [17]. Nel caso digitale i 2 livelli logici si fanno corrispondere a: tensione X sulla linea A e 0 su B, tensione $-X$ sulla linea B e 0 su A
- Un Driver è un componente software che fornisce al sistema operativo un'interfaccia mediante la quale può usare un certo dispositivo hardware senza conoscerne i dettagli implementativi

Ringraziamenti

Bibliografia

[1] Intel, *8087 Math Coprocessor Datasheet*, October 1989.

[2] Oracle Corporation, *Sun Crypto Accelerator 1000 Installation and User's Guide, Version 2.0*, October 2005.

[3] O. Shacham and M. Reynders, "Pixel visual core: Image processing and machine learning on pixel 2." https://blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/, October 2017. Accessed: 2025-06-24.

[4] A. S. Tanenbaum and T. Austin, *Architettura dei calcolatori: Un approccio strutturale*. Pearson Education Italia, 5 ed., 2006.

[5] Fact.MR, "Hardware acceleration market outlook (2023 to 2033)," market research report, Fact.MR, July 2023.

[6] B. Zaid, "Analysis of the pc video games and gpu market." https://medium.com/@baraaz/analysis-of-the-pc-video-games-and-gpu-market-dbdaecc2b56f, January 2023. Accessed: 2025-06-25.

[7] W. mei Hwu and S. Patel, "Accelerator architectures — a ten-year retrospective," *IEEE Micro*, 2018.

[8] B. Peccerillo, M. Mannino, A. Mondelli, and S. Bartolini, "A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives," *Journal of Systems Architecture*, 2022.

[9] A. Verma and P. K. Dahiya, "Pcie bus: A state-of-the-art-review," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, July 2017.

[10] J. Jaeger, "Fpga-based rapid prototyping of asic, assp, and soc designs," *EDN (via plDesignLine / Design & Reuse)*, October 2009.

[11] PCI-SIG, *PCI Express ® Base Specification Revision 5.0, Version 1.0*, May 2019.

[12] PCMCIA, *ExpressCard ® Standard, Release 2.0*, 2009.

[13] J. B. Yoon and N. Malone, "Pcie link training overview," tech. rep., Texas Instruments, August 2022.

[14] PCI-SIG, *PCI Local Bus Specification Revision 3.0*, February 2004.

[15] PCI-SIG, *PCI Code and ID Assignment Specification Revision 1.11*, January 2019.

[16] Intel, "Nrz fundamentals: Nrz vs. pam-4 signaling for high-speed serial interfaces." https://www.intel.com/content/www/us/en/docs/programmable/683852/current/nrz-fundamentals.html, December 2019. Accessed: 2025-06-27.

[17] L. Harvie, "Understanding and implementing differential pair routing in high-speed pcbs." https://runtimerec.com/understanding-and-implementing-differential-pair-routing-in-high-speed-pcbs/, March 2025. Accessed: 2025-06-27.