2. MAGISTRALA PCI EXPRESS

Această lucrare de laborator prezintă varianta serială a magistralei PCI, numită PCI Express. După o prezentare generală a magistralei PCI Express, sunt prezentate detalii despre arhitectura sa, incluzând legătura PCI Express, topologia magistralei, nivelele arhitecturale, tranzacții și întreruperi. Nivelul fizic este prezentat în mai multe detalii și sunt descrise cele mai importante registre de configurație. Scopul aplicațiilor este de a accesa spațiul de configurație PCI și de a decodifica informațiile disponibile în registrele de configurație ale dispozitivelor PCI și PCI Express.

2.1. Prezentare generală a magistralei PCI Express

Magistrala PCI Express (PCIe) reprezintă a treia generație a magistralei PCI (*Peripheral Component Interconnect*), cu o performanță și fiabilitate mai ridicate comparativ cu generațiile anterioare PCI și PCI-X. Spre deosebire de aceste versiuni anterioare, care sunt magistrale paralele, PCIe este o magistrală serială. Din cauza naturii seriale a magistralei PCIe, aceasta are mai multe avantaje față de o magistrală paralelă: număr mai redus de pini ai circuitelor integrate, complexitate mai redusă a plăcilor de circuite imprimate și cost mai redus al acestora.

Specificațiile magistralei PCIe provin din specificațiile magistralei 3GIO (*Third Generation I/O*), care au fost elaborate de grupul de lucru Arapahoe (*Arapahoe Work Group*). Acest grup era format din reprezentanți ai firmelor Compaq Computer, Dell Computer, Hewlett-Packard, IBM, Intel și Microsoft. Specificațiile au fost transferate în anul 2002 la organizația PCI-SIG (*PCI Special Interest Group*), un grup de peste 900 de firme care a elaborat și actualizat standardele diferitelor versiuni ale magistralelor anterioare PCI și PCI-X (www.pcisig.com). Noua magistrală a fost redenumită PCI Express, nume care reflectă atât viteza ridicată a magistralei, cât și compatibilitatea software a acesteia cu generațiile anterioare PCI și PCI-X. Figura 2.1 ilustrează simbolul magistralei PCI Express.

Figura 2.1. Simbolul magistralei PCI Express.

Proiectanții magistralei PCIe au menținut principalele caracteristici avantajoase ale arhitecturii generațiilor anterioare de magistrale PCI. De exemplu, magistrala PCIe utilizează același model de comunicație ca și magistralele PCI și PCI-X. Sunt păstrate aceleași spații de adrese: de memorie, de I/E și de configurație. Magistrala PCIe permite utilizarea acelorași tipuri de tranzacții ca și magistralele anterioare: citire/scriere a spațiului de memorie, citire/scriere a spațiului de I/E, citire/scriere a spațiului de configurație. Prin aceasta, se păstrează compatibilitatea cu sistemele de operare și driverele software existente, care nu necesită modificări.

Ca și magistralele PCI anterioare, magistrala PCIe permite interconexiuni între circuite integrate și interconexiuni între plăci de circuite imprimate prin conectori și plăci de extensie. Plăcile de extensie au o structură similară cu plăcile utilizate de magistralele PCI și PCI-X. O placă de bază PCIe are dimensiuni similare cu plăcile de bază ATX existente, utilizate la calculatoarele personale.

Pe lângă păstrarea unor caracteristici avantajoase ale magistralelor PCI și PCI-X, magistrala PCIe introduce diferite îmbunătățiri pentru creșterea performanței și reducerea

costurilor. Spre deosebire de generațiile anterioare PCI și PCI-X, care sunt magistrale paralele partajate, magistrala PCIe utilizează o interconexiune serială punct la punct pentru comunicația între două dispozitive periferice. În primul rând, o interconexiune serială elimină dezavantajele unei magistrale paralele, în special dificultatea sincronizării între liniile multiple de date datorită nesimetriei de propagare a semnalelor. Cauza acestei nesimetrii poate fi lungimea diferită a căilor de date parcurse de diferitele semnale sau propagarea pe straturi diferite ale plăcii de circuit imprimat. Deși semnalele de date ale unei magistrale paralele sunt transmise simultan, ele pot ajunge la destinație la momente de timp diferite. Creșterea frecvenței semnalului de ceas al unei magistrale paralele este dificilă, deoarece durata ciclului de ceas poate deveni mai mică decât durata nesimetriei de propagare a semnalelor (care poate fi de câteva nanosecunde). La o magistrală serială nu apare problema nesimetriei de propagare a semnalelor, deoarece nu există un semnal de ceas extern, informațiile de sincronizare fiind înglobate în semnalul serial transmis. În al doilea rând, o interconexiune punct la punct implică o încărcare electrică redusă a legăturii, ceea ce permite creșterea frecvenței semnalului de ceas utilizat pentru transferurile de date.

Performanța magistralei PCIe este scalabilă, ceea ce se obține prin implementarea unui număr variabil de benzi de comunicație pentru o interconexiune, în funcție de cerințele de performanță pentru acea interconexiune.

Magistrala PCIe implementează o tehnologie bazată pe comutatoare pentru interconectarea unui număr mare de dispozitive periferice. Pentru interconexiunea serială se utilizează un protocol de comunicație bazat pe pachete. În locul unor semnale speciale pentru diferite funcții, cum sunt semnalarea întreruperilor, gestiunea erorilor sau gestiunea puterii consumate, atât datele, cât și comenzile sunt transmise în pachete. Prin aceasta se reduce numărul de pini ai circuitelor și scade costul acestor circuite.

Magistrala PCIe dispune de mai multe facilități avansate. Astfel, facilitatea de calitate a serviciilor (*Quality of Service* – QoS) permite asigurarea unor performanțe diferențiate pentru diferite aplicații. Facilitatea de conectare și deconectare a unor module în timpul funcționării permite realizarea unor sisteme care funcționează fără întrerupere. Facilitățile de gestiune avansată a puterii consumate permit implementarea unor aplicații mobile cu un consum redus de putere. Facilitatea de gestiune a erorilor permite utilizarea magistralei PCIe în sistemele robuste necesare serverelor performante.

2.2. Caracteristici ale magistralei PCI Express

Principalele caracteristici ale magistralei PCIe sunt următoarele:

- Unifică arhitectura de I/E pentru diferite tipuri de sisteme, cum sunt calculatoare de birou, calculatoare mobile, stații de lucru, servere, platforme de comunicație și sisteme înglobate.
- Permite interconectarea atât a circuitelor integrate de pe placa de bază, cât și a plăcilor de extensie prin intermediul unor conectori sau cabluri.
- Comunicația este bazată pe pachete, cu rată de transfer și eficiență ridicate.
- Interfața este serială, ceea ce permite reducerea numărului de pini și simplificarea conexiunilor.
- Performanța este scalabilă, obținută prin posibilitatea implementării unei anumite interconexiuni cu ajutorul mai multor benzi de comunicație.
- Modelul software este compatibil cu arhitectura PCI clasică, ceea ce permite configurarea circuitelor PCIe, încărcarea sistemelor de operare și utilizarea driverelor software existente, fără a fi necesare modificări.
- Permite o calitate diferențiată a serviciilor (QoS) prin posibilitatea de alocare a unor resurse dedicate pentru anumite fluxuri de date, de configurare a politicilor de arbitra-

re QoS pentru fiecare componentă și de a utiliza transferuri izocrone pentru aplicații în timp real.

- Pune la dispoziție o gestiune avansată a puterii consumate prin identificarea posibilităților de gestiune a puterii consumate de către fiecare periferic, trecerea unui periferic într-o stare cu un anumit consum de putere și recepționarea notificărilor asupra stării curente a perifericului.
- Asigură integritatea datelor la nivelul legăturii pentru toate tipurile de tranzacții.
- Permite raportarea și gestionarea avansată a erorilor pentru îmbunătățirea izolării defectelor și recuperarea erorilor.
- Permite conectarea și deconectarea perifericelor în timpul funcționării, fără a fi necesară utilizarea unor semnale suplimentare.

2.3. Arhitectura magistralei PCI Express

2.3.1. Legătura PCI Express

O legătură PCIe minimală constă din două canale de comunicație unidirecționale (simplex) între două dispozitive periferice PCIe, un canal pentru transmisie și unul pentru recepție (figura 2.2). Pe aceste canale sunt transmise pachete de date și de comenzi. Fiecare canal este implementat fizic printr-o pereche de fire pe care se transmit semnale diferențiale cu tensiuni reduse. O asemenea legătură PCIe minimală este numită bandă de comunicație (*lane*). Pentru scalarea ratei de transmisie, o legătură PCIe poate agrega benzi de comunicație multiple, notate cu xN, unde N este lățimea legăturii. Specificațiile magistralei PCIe indică posibilitatea utilizării unor legături cu lățimi de x1, x2, x4, x8, x12, x16 și x32.

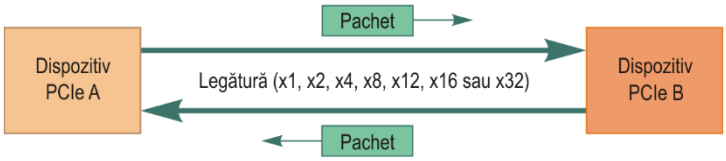

Figura 2.2. Legătură PCI Express.

În timpul inițializării hardware, pentru fiecare legătură PCIe se negociază lățimea legăturii și frecvența de funcționare. Lățimea legăturii și frecvența de funcționare sunt setate în mod automat de către dispozitivele de la capetele legăturii, fără implicarea sistemului de operare. După inițializare, fiecare legătură trebuie să opereze doar la frecvența de funcționare care a fost setată. Prima versiune a specificațiilor magistralei PCIe a definit o frecvență de funcționare de 2,5 GHz, ceea ce corespunde unei rate de transfer efective de 2,5 Gbiți/s pentru fiecare bandă de comunicație și direcție. În versiunile ulterioare, frecvența de funcționare a crescut la 5 GHz, iar apoi la 8 GHz.

2.3.2. Topologia magistralei PCI Express

Un sistem PCIe este compus din legături PCIe care interconectează un set de componente. Un exemplu de topologie este ilustrată în figura 2.3. Principalele componente ale acestei topologii sunt un complex rădăcină, multiple puncte terminale (dispozitive de I/E), un comutator și o punte PCIe-PCI, toate fiind interconectate prin legături PCIe. Toate dispozitivele și legăturile asociate cu un complex rădăcină, care sunt conectate la acesta direct sau indirect (prin comutatoare și punți) reprezintă o *ierarhie*.

Complexul rădăcină este dispozitivul care conectează unul sau mai multe procesoare și subsistemul de memorie la dispozitivele de I/E. Acest dispozitiv reprezintă rădăcina unei ierarhii de I/E. Complexul rădăcină poate avea mai multe porturi PCIe; în figura 2.3, complexul rădăcină conține trei porturi. Fiecare port definește un *domeniu ierarhic* separat. Un do-

meniu ierarhic poate fi compus dintr-un singur punct terminal sau o sub-ierarhie conținând unul sau mai multe comutatoare și puncte terminale.

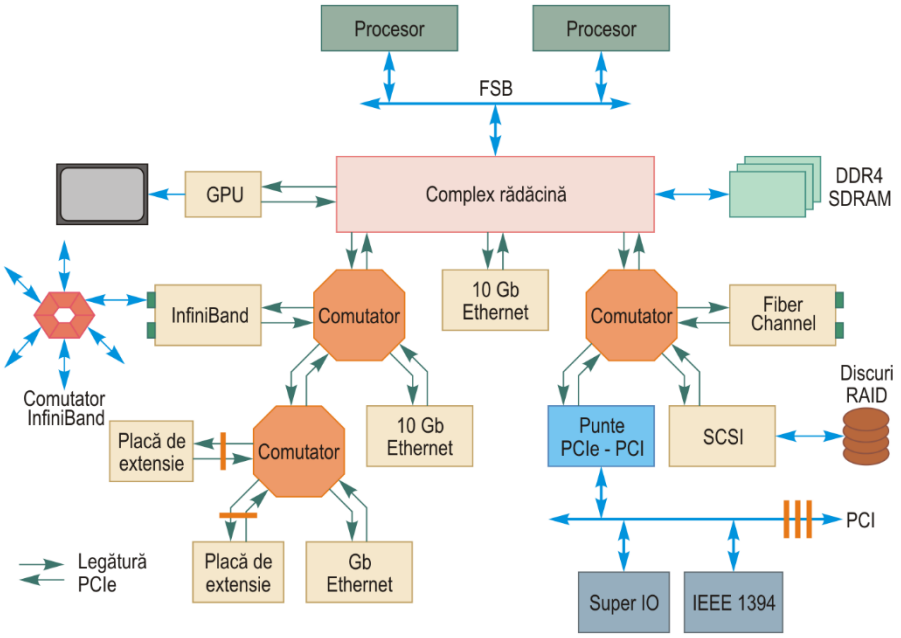

Figura 2.3. Exemplu de topologie PCI Express.

Un complex rădăcină implementează diferite resurse, cum sunt controlerul de întreruperi, controlerul pentru gestiunea puterii consumate, logica pentru detecția și raportarea erorilor. Complexul rădăcină conține o magistrală internă, reprezentând magistrala cu numărul 0 din întreaga ierarhie. Acest dispozitiv inițiază cereri de tranzacție din partea unui procesor, transmite pachete de la porturile sale și recepționează pachete la aceste porturi pe care le transmite la memorie. În mod opțional, un complex rădăcină cu mai multe porturi poate ruta pachete de la unul din porturi la un alt port.

Punctele terminale (*endpoints*) reprezintă dispozitive periferice care participă la tranzacțiile PCIe. Există două tipuri de puncte terminale. Un punct terminal *inițiator* inițiază o tranzacție în sistemul PCIe, iar un punct terminal *destinație* răspunde la tranzacțiile care îi sunt adresate. Într-o ierarhie PCIe, pe lângă puncte terminale PCIe, pot exista și puncte terminale compatibile cu generațiile anterioare ale magistralei PCI. Ca și în cazul magistralei PCI clasice, dispozitivele PCIe pot avea până la opt funcții logice, astfel încât un punct terminal poate fi compus din până la opt funcții numerotate de la 0 la 7. Fiecărui punct terminal i se atribuie un identificator de dispozitiv (ID) care se compune din numărul magistralei, numărul dispozitivului și numărul funcției.

Un *comutator* este definit ca un ansamblu logic de mai multe punți virtuale între diferite magistrale PCI, fiecare punte fiind asociată cu un port al comutatorului. Comutatorul din figura 2.4 conține patru punți virtuale. Aceste punți sunt conectate printr-o magistrală internă. Unul din porturile comutatorului este conectat la complexul rădăcină, iar celelalte porturi sunt conectate la puncte terminale sau la alte comutatoare. Pentru programele de configurare, un comutator apare ca două sau mai multe punți logice între magistrale PCI.

Un comutator transferă pachete de la oricare din porturile sale de intrare la unul din porturile de ieșire, într-un mod similar cu o punte PCI-PCI. Pachetele sunt transferate printr-un mecanism de rutare care se bazează fie pe o adresă, fie pe un identificator. Se utilizează un mecanism de arbitrare prin care se determină prioritatea cu care sunt transferate pachetele de la porturile de intrare la cele de ieșire.

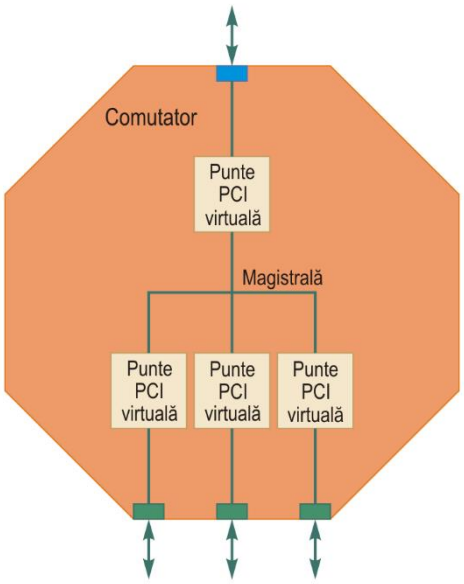

Figura 2.4. Structura internă a unui comutator.

2.3.3. Nivele ale arhitecturii PCI Express

Un sistem PCIe poate fi structurat pe un număr de cinci nivele logice, care sunt descrise pe scurt în continuare.

- *Nivelul de configurare/SO* gestionează configurarea dispozitivelor PCIe de către sistemul de operare pe baza specificațiilor *Plug-and-Play* pentru inițializarea, enumerarea și configurarea dispozitivelor de I/E.
- *Nivelul software* interacționează cu sistemul de operare prin intermediul acelorași drivere ca și magistrala PCI convențională.
- *Nivelul tranzacțiilor* gestionează transmiterea și recepția informațiilor utilizând un protocol bazat pe pachete.
- *Nivelul legăturii de date* asigură integritatea transferurilor de date prin detecția erorilor cu ajutorul unui cod ciclic redundant (CRC – *Cyclic Redundancy Check*).
- *Nivelul fizic* realizează transmiterea pachetelor pe legăturile seriale PCIe.

Specificațiile PCIe definesc arhitectura dispozitivelor PCIe sub forma a trei nivele logice, care sunt ultimele trei nivele dintre cele enumerate anterior. Fiecare din aceste nivele se poate diviza în două secțiuni, una care procesează informațiile care vor fi transmise și una care procesează informațiile recepționate (figura 2.5). Această organizare logică nu implică însă o anumită implementare a dispozitivelor PCIe.

Magistrala PCIe utilizează pachete pentru transferul informațiilor între perechile de dispozitive conectate printr-o legătură PCIe. Considerăm mai întâi transferul informațiilor de la dispozitivul A la dispozitivul B. Pachetele sunt formate în nivelul tranzacțiilor pe baza informațiilor obținute de la nucleul dispozitivului și de la aplicație. Un anumit pachet este memorat într-un buffer pentru a fi transmis către nivelele inferioare. Nivelul legăturii de date extinde pachetul cu informații suplimentare necesare pentru detecția erorilor de către un dispozitiv receptor. Acest pachet este apoi codificat de către nivelul fizic și este transmis prin semnale diferențiale pe legătura PCIe de către porțiunea analogică a acestui nivel.

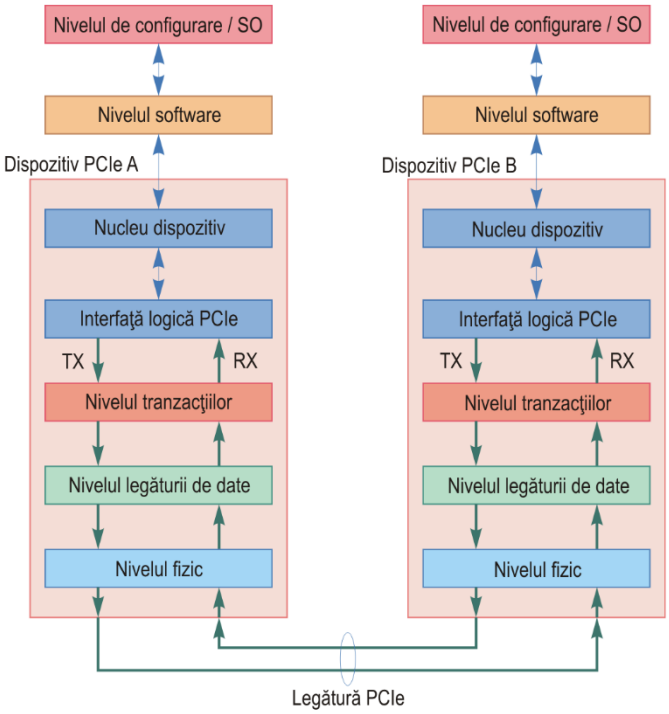

Figura 2.5. Nivele ale unui sistem PCIe și ale dispozitivelor PCIe.

Considerăm acum recepția informațiilor de către dispozitivul B. Pachetele sunt decodificate de către nivelul fizic și conținutul acestora este transferat la nivelele superioare. Nivelul legăturii de date verifică dacă există erori într-un pachet recepționat, iar în cazul în care nu există erori transmite pachetul la nivelul tranzacțiilor. Acest nivel memorează pachetul într-un buffer și convertește informațiile din pachet într-o reprezentare care poate fi procesată de către nucleul dispozitivului și aplicație.

Figura 2.6 ilustrează fluxul conceptual al informațiilor care sunt transferate prin cele trei nivele logice al dispozitivelor PCIe.

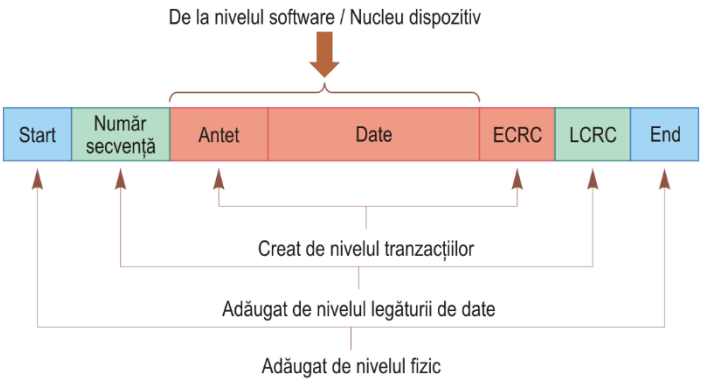

Figura 2.6. Fluxul pachetelor prin nivelele logice ale dispozitivelor PCIe.

Nivelul software sau nucleul dispozitivului transmite nivelului tranzacțiilor informațiile necesare pentru a crea secțiunea principală a pachetului. Aceste informații sunt antetul și câmpul de date al pachetului. În mod opțional, se calculează un cod CRC și se adaugă la pachet prin câmpul ECRC (*End-to-End* CRC). Acest câmp este utilizat de către dispozitivul destinație al pachetului pentru a detecta erorile CRC din antet și câmpul de date.

Pachetul creat de către nivelul tranzacțiilor este transmis nivelului legăturii de date, care concatenează la acest pachet un număr de secvență și un alt câmp CRC, LCRC (*Link* CRC). Câmpul LCRC este utilizat de către dispozitivul receptor de la celălalt capăt al legăturii pentru a detecta erorile CRC din pachetul creat de nivelul tranzacțiilor și din numărul de secvență. Pachetul rezultat este transmis nivelului fizic, care adaugă două caractere Start și

End de câte un octet care vor încadra pachetul. Pachetul este codificat apoi și este transmis prin semnale diferențiale pe o legătură PCIe utilizând numărul disponibil de benzi de comunicație.

2.3.4. Tranzacții PCI Express

O *tranzacție* este definită ca o serie de una sau mai multe transmisii de pachete necesare pentru a realiza un transfer de informații între un dispozitiv inițiator și unul destinație. Există patru categorii de tranzacții PCIe: de memorie, de I/E, de configurație și de mesaje. Primele trei categorii existau și la magistralele anterioare PCI și PCI-X. Exemple de asemenea tranzacții sunt citirea și scrierea memoriei, citirea și scrierea spațiului de I/E, citirea și scrierea registrelor de configurație. Tranzacțiile de mesaje sunt specifice magistralei PCIe. Tranzacțiile de mesaje, numite și mesaje, se utilizează pentru semnalarea întreruperilor, gestiunea puterii de alimentare sau semnalarea erorilor.

Pe de altă parte, în cazul arhitecturii PCIe există două tipuri de tranzacții. Primul tip este reprezentat de tranzacții la care dispozitivul destinație returnează un pachet de terminare la dispozitivul inițiator, ca răspuns la pachetul de cerere transmis de către dispozitivul inițiator. Aceste tranzacții sunt executate conform protocolului definit pentru tranzacțiile divizate care există la magistrala PCI-X. La tranzacțiile divizate, după inițierea unei tranzacții, dispozitivul destinație memorează informațiile necesare executării tranzacției și semnalează un răspuns întârziat. Dispozitivul inițiator eliberează magistrala, care va fi disponibilă pentru alte tranzacții. Dacă au fost solicitate date de la dispozitivul destinație, cum este cazul unei tranzacții de citire, dispozitivul destinație pregătește aceste date, obține controlul asupra magistralei și returnează datele cerute. Pachetul de terminare returnat de dispozitivul destinație confirmă faptul că pachetul de cerere a fost recepționat de către dispozitivul destinație.

Al doilea tip este reprezentat de tranzacții la care dispozitivul destinație nu returnează un pachet de terminare la dispozitivul inițiator. În acest fel, timpul necesar terminării tranzacției este mai redus, dar dispozitivul inițiator nu cunoaște dacă pachetul de cerere a fost recepționat cu succes de către dispozitivul destinație.

2.3.5. Întreruperi PCI Express

Dispozitivele conectate la magistrala PCIe pot semnala cererile de întrerupere utilizând unul din două metode disponibile. Dispozitivele compatibile PCIe trebuie să utilizeze mecanismul nativ al magistralei PCIe pentru semnalarea întreruperilor, cel al întreruperilor semnalate prin mesaje (MSI – *Message Signaled Interrupts*). Dispozitivele compatibile cu generațiile anterioare PCI și PCI-X pot utiliza semnalele dedicate ale magistralei PCI pentru cererile de întrerupere.

Mecanismul nativ al magistralei PCIe pentru semnalarea întreruperilor (MSI) a fost definit în versiunea 2.2 a specificațiilor magistralei PCI ca un mecanism opțional, devenind obligatoriu la magistrala PCI-X. Termenul de *întreruperi semnalate prin mesaje* poate crea confuzii în contextul magistralei PCIe din cauza existenței tranzacțiilor de mesaje PCIe. O întrerupere semnalată prin mesaje nu reprezintă un mesaj PCIe, ci o simplă tranzacție de scriere în memorie. O tranzacție de scriere în memorie asociată cu mecanismul MSI se deosebește de alte tranzacții de scriere în memorie doar prin adresele lor de destinație, adresele de memorie pentru semnalarea întreruperilor fiind rezervate de sistem.

Mecanismul compatibil cu magistrala PCI pentru semnalarea întreruperilor presupune utilizarea semnalelor pentru cererile de întrerupere definite pentru magistrala PCI. Aceste semnale sunt *INTA#*, *INTB#*, *INTC#* și *INTD#* (caracterul # indică un semnal activ în starea logică 0). Achitarea unei cereri de întrerupere este indicată printr-o anumită configurație pe liniile de comandă ale magistralei PCI. Un dispozitiv compatibil cu magistrala PCI va activa una din liniile *INTx#* pentru a semnala o cerere de întrerupere. Deoarece magistrala PCIe nu dispune de liniile de întrerupere *INTx#*, se utilizează mesaje speciale care au rolul unor linii virtuale *INTx#*. Aceste mesaje sunt destinate controlerului de întreruperi amplasat, de obicei, în complexul rădăcină.

Figura 2.7 ilustrează semnalarea întreruperilor generate de către trei tipuri de dispozitive. Dispozitivul PCIe utilizează mecanismul nativ MSI. Dispozitivul PCI-X utilizează mesaje INTx pentru semnalarea întreruperilor. Dispozitivul PCI utilizează semnalele *INTx#* pentru semnalarea întreruperilor către puntea PCIe-PCI, iar această punte comunică prin mesaje de activare INTA cu controlerul de întreruperi.

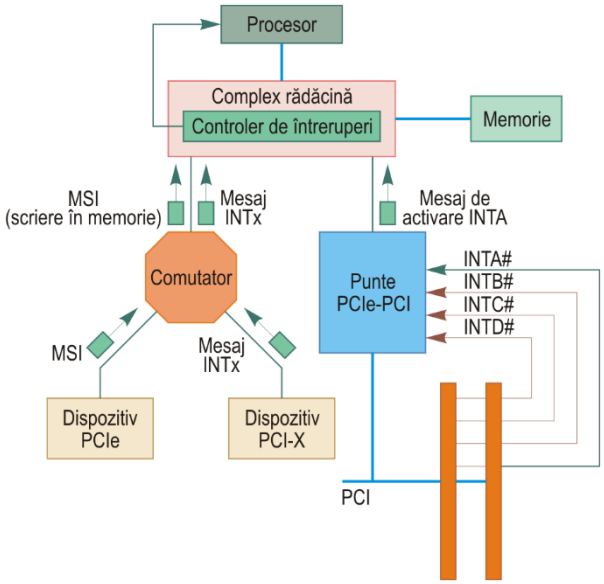

Figura 2.7. Metode de semnalare a întreruperilor într-un sistem PCIe.

2.4. Nivelul fizic

Nivelul fizic realizează transmisia pe o legătură PCIe a pachetelor recepționate de la nivelul legăturii de date. De asemenea, nivelul fizic recepționează pachete de pe o legătură PCIe și le transmite la nivelul legăturii de date.

Nivelul fizic este împărțit în două părți, nivelul fizic logic și nivelul fizic electric. Nivelul fizic logic conține circuite logice pentru procesarea pachetelor înaintea transmisiei acestora pe o legătură PCIe și pentru procesarea pachetelor recepționate de pe o legătură PCIe înaintea transmisiei lor la nivelul legăturii de date. Nivelul fizic electric reprezintă interfața analogică utilizată pentru conectarea la legătura PCIe. Acest nivel constă din transmițătoare și receptoare diferențiale pentru fiecare bandă de comunicație a unei legături PCIe.

2.4.1. Secțiunea de transmisie

Pachetele generate de nivelul tranzacțiilor sau de nivelul legăturii de date sunt recepționate de nivelul fizic și sunt memorate într-un buffer. Aceste pachete sunt încadrate apoi cu un caracter *Start* și un caracter *End* de câte un octet. Aceste caractere sunt utilizate de către un dispozitiv receptor pentru a detecta începutul și sfârșitul unui pachet.

Dacă un pachet va fi transmis pe o legătură PCIe care conține mai multe benzi de comunicație, octeții pachetului vor fi transmiși în mod întrețesut, ceea ce înseamnă că octeții succesivi din pachet vor fi transmiși pe benzi de comunicație succesive ale legăturii PCIe. Deși această întrețesere a datelor necesită o complexitate hardware semnificativă pentru asamblarea corectă a octeților recepționați într-un pachet, această metodă reduce în mod semnificativ întârzierea cu care va fi recepționat un anumit octet pe o legătură.

Fiecare octet al unui pachet este apoi cifrat utilizând un registru de deplasare cu reacție inversă liniară. Prin această operație, se elimină din șirul de date transmis configurațiile de biți care se repetă, cu scopul de a se reduce interferențele electromagnetice. Octeții rezultați sunt apoi codificați printr-o metodă care asigură limitarea lungimii șirurilor cu biți consecutivi de 1 și de 0. Scopul principal al acestei codificări este de a se crea în șirul de biți care va fi transmis tranziții suficiente de la valoarea 1 la 0 și de la 0 la 1, ceea ce va facilita generarea

unui semnal de ceas de recepție de către dispozitivul receptor cu ajutorul unui circuit cu calare de fază PLL (*Phase-Locked Loop*). Astfel, nu este necesară transmiterea unui semnal de ceas pentru sincronizare împreună cu datele.

La primele versiuni ale magistralei PCIe (până la versiunea 3.0), metoda de codificare utilizată este 8b/10b, prin care fiecare octet este codificat printr-un simbol de 10 biți. Prin această metodă, lățimea de bandă efectivă este redusă cu 20%. La versiunea 3.0 a magistralei PCIe se utilizează metoda de codificare 128b/130b, prin care lățimea de bandă efectivă se reduce cu doar aproximativ 1,5%.

Octeții codificați ai unui pachet sunt apoi convertiți într-un șir de biți cu ajutorul unui convertor paralel-serial și sunt transmiși pe benzile de comunicație ale legăturii PCIe.

2.4.2. Secțiunea de recepție

Secțiunea de recepție a nivelului fizic recepționează șirurile de biți seriale de pe fiecare bandă de comunicație a legăturii. Șirurile de biți sunt convertite în simboluri de 10 biți sau 130 de biți cu ajutorul unui convertor serial-paralel. Logica de recepție conține și un buffer care compensează variația între frecvența semnalului de ceas al transmițătorului și frecvența semnalului de ceas al receptorului.

Simbolurile de 10 biți sau 130 de biți sunt convertite în octeți cu ajutorul unui decodificator. Octeții sunt apoi decifrați, iar logica de asamblare recreează pachetul original transmis prin combinarea octeților recepționați pe benzile de comunicație ale legăturii PCIe.

2.4.3. Inițializarea și antrenarea legăturii

O funcție suplimentară a nivelului fizic este procesul de inițializare și antrenare a legăturii PCIe. Acest proces este automat și nu implică nivelul software. În timpul procesului de inițializare și antrenare a legăturii sunt determinați parametri de funcționare cum sunt dimensiunea legăturii, rata datelor legăturii, inversarea benzilor de comunicație, inversarea polarității și compensarea nesimetriei de propagare a semnalelor pe benzile de comunicație multiple ale unei legături.

Dimensiunea legăturii. Este posibilă conectarea a două dispozitive prin porturi cu un număr diferit de benzi de comunicație pe legătură. După inițializare, dimensiunea legăturii va fi setată la numărul minim al benzilor de comunicație al celor două porturi conectate. De exemplu, un anumit dispozitiv având un port PCIe cu dimensiunea x2 poate fi conectat cu un alt dispozitiv având un port PCIe cu dimensiunea x4. Pentru comunicația între cele două dispozitive, dimensiunea legăturii va fi setată la x2.

Rata datelor legăturii. Inițial, rata datelor unei legături este setată la valoarea minimă de 2,5 Gbiți/s. În timpul antrenării legăturii, fiecare dispozitiv prezintă rata maximă a datelor pe care îl permite. Legătura va fi inițializată cu frecvența comună maximă permisă de cele două dispozitive de la capetele legăturii.

Inversarea benzilor de comunicație. Dacă o legătură conține mai multe benzi de comunicație, acestea se numerotează. La conectarea fizică a două dispozitive, este posibil ca benzile de comunicație ale porturilor dispozitivelor să nu fie conectate corect. Într-un asemenea caz, antrenarea legăturii permite inversarea numerelor benzilor de comunicație, astfel încât aceste numere ale porturilor adiacente de la capetele legăturii să corespundă.

Inversarea polarității. Este posibil ca perechile de fire diferențiale D+ și D- ale două dispozitive să nu fie conectate corect. În acest caz, în urma antrenării legăturii, dispozitivul receptor va inversa polaritatea terminalelor circuitelor de recepție.

Compensarea nesimetriei de propagare. În cazul unei legături cu mai multe benzi de comunicație, din cauza variației lungimii liniilor fizice și a caracteristicilor diferite ale circuitelor transmițătoare/receptoare, șirurile de biți pe o bandă de comunicație pot fi recepționate decalat față de alte benzi de comunicație. Circuitele de recepție trebuie să compenseze acest decalaj prin inserarea unor întârzieri pe unele benzi de comunicație.

2.5. Versiuni ale specificațiilor PCI Express

Prima versiune 1.0a a specificațiilor magistralei PCIe a fost publicată de organizația PCI-SIG în anul 2003. Această versiune specifica o frecvență de funcționare de 2,5 GHz și o rată maximă a datelor de 250 MB/s pe o bandă de comunicație.

Versiunea 1.1 a fost publicată în anul 2005. Această versiune a introdus mai multe îmbunătățiri ale magistralei, dar nu a specificat rate mai ridicate ale datelor.

Versiunile 2.0 și 2.1 au fost publicate în anul 2007. Frecvența de funcționare a fost dublată la 5 MHz, ceea ce permite o rată maximă a datelor de 500 MB/s pe o bandă de comunicație. Astfel, o legătură PCIe x32 poate asigura o rată maximă teoretică a datelor de 16 GB/s. Specificațiile PCIe 2.x introduc îmbunătățiri ale protocolului de transfer punct la punct și ale arhitecturii nivelului software. Conectorii plăcilor de bază PCIe 2.x sunt compatibili cu plăcile de extensie PCIe 1.x.

Versiunea 3.0 a specificațiilor PCIe fost publicată de organizația PCI-SIG în anul 2010. Frecvența de funcționare a crescut la 8 GHz, rata maximă a datelor fiind de 8 GT/s (giga transferuri pe secundă) sau 985 MB/s pe o bandă de comunicație. Specificațiile au introdus îmbunătățiri legate de protocolul electric, integritatea datelor și recuperarea erorilor. De asemenea, a fost introdusă codificarea 128b/130b a datelor, care este mai eficientă decât codificarea 8b/10b utilizată la versiunile PCIe anterioare.

Versiunea 4.0, publicată în anul 2017, a dublat rata maximă a datelor asigurată de versiunea 3.0 la 16 GT/s sau 1,97 GB/s pe o bandă de comunicație. Puterea consumată a dispozitivelor în starea lor activă și inactivă a fost optimizată. De asemenea, această versiune a introdus conectorul OCuLink-2, care este versiunea a doua a conectorului OCuLink (*Optical-Copper Link*), cu până la patru benzi de comunicație (7,88 GB/s) pe fire de cupru; este posibil să fie dezvoltată o versiune pe fibră optică în viitor.

Versiunea 5.0 a fost publicată în mai 2019. Această versiune dublează din nou rata maximă a datelor față de versiunea precedentă la 32 GT/s pe o bandă de comunicație și păstrează compatibilitatea cu versiunile precedente.

Versiunea 6.0 a fost publicată în ianuarie 2022. La această versiune, rata maximă a datelor poate ajunge la 64 GT/s pe o bandă de comunicație, astfel încât o legătură PCIe x32 poate asigura o rată maximă teoretică a datelor de 256 GB/s.

2.6. Registre de configurație

Fiecare funcție (dispozitiv) PCIe implementează un set de registre de configurație care permit nivelului software descoperirea existenței funcției și configurarea acesteia pentru funcționarea normală. La comanda programelor de aplicație, complexul rădăcină al sistemului PCIe inițiază tranzacții de configurație pentru citirea și scrierea registrelor de configurație ale funcțiilor PCIe.

2.6.1. Spațiul de configurație al unei funcții PCIe

Registrele de configurație ale unei funcții PCIe sunt implementate în spațiul de configurație al arhitecturii PCIe. Spre deosebire de o funcție PCI sau PCI-X, care poate avea un spațiu de configurație de 256 B, o funcție PCIe are un spațiu de configurație extins la 4 KB. Rezultă că dimensiunea spațiului de configurație PCIe este de 256 MB. Aceasta se obține prin multiplicarea dimensiunii de 4 KB cu 8 funcții pe un dispozitiv, cu 32 de dispozitive pe o magistrală și cu 256 de magistrale pe un sistem PCIe.

Structura spațiului de configurație al unei funcții PCIe este ilustrată în figura 2.8.

Spațiul de configurație al unei funcții PCIe este împărțit în două secțiuni. Prima secțiune reprezintă spațiul de configurație compatibil PCI și ocupă primii 256 B (64 de cuvinte duble de 32 de biți) ai spațiului de 4 KB. Primele 16 cuvinte duble ale acestei secțiuni reprezintă antetul de configurație PCI, iar celelalte 48 de cuvinte duble sunt rezervate pentru implementarea registrelor de configurație specifice funcției.

A doua secțiune a spațiului de configurație al unei funcții PCIe reprezintă spațiul de configurație extins PCIe și ocupă 3840 B (960 de cuvinte duble). Acest spațiu este utilizat

pentru implementarea registrelor extinse PCIe, care sunt opționale. Exemple de asemenea registre sunt setul de registre pentru raportarea avansată a erorilor, setul de registre pentru canale virtuale sau setul de registre pentru numărul serial al dispozitivului.

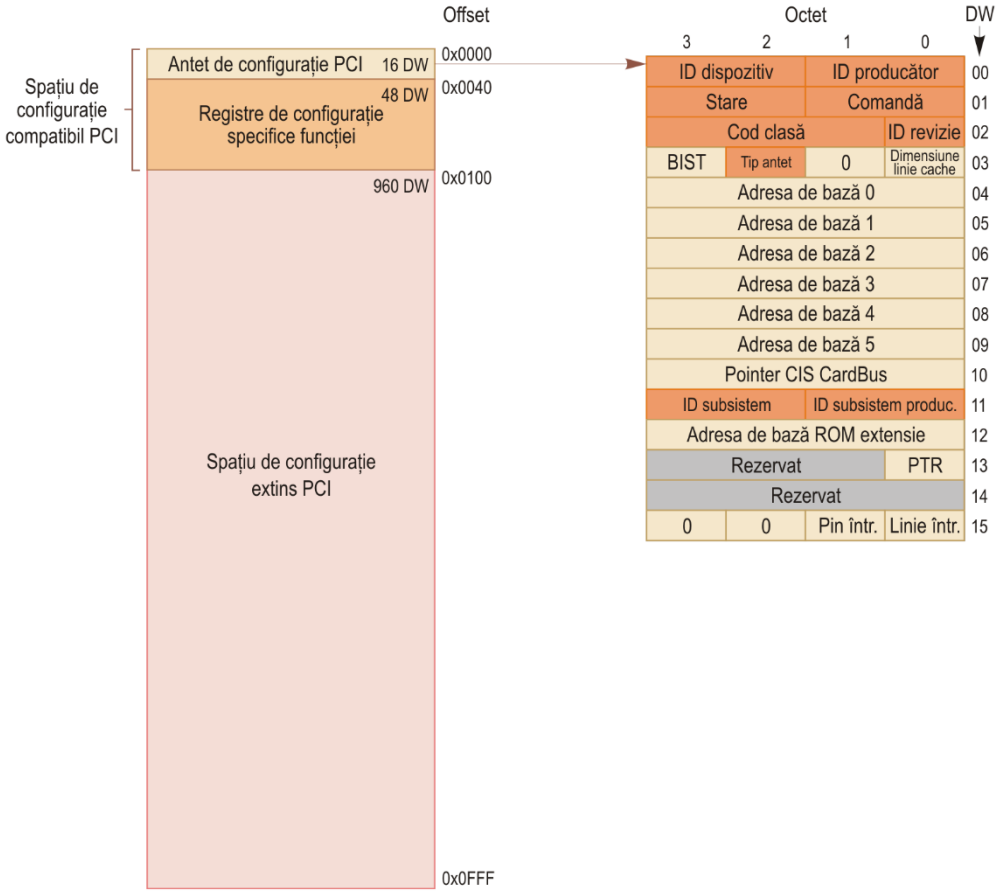

Figura 2.8. Structura spațiului de configurație al unei funcții PCIe.

Spațiul de configurație compatibil PCI poate fi accesat prin două metode, fie prin mecanismul de configurație compatibil PCI, fie prin mecanismul de configurație îmbunătățit PCIe. Aceste mecanisme de acces sunt prezentate în secțiunile următoare. Spațiul de configurație extins al unei funcții PCIe poate fi accesat numai prin mecanismul de configurație îmbunătățit PCIe.

2.6.2. Mecanismul de configurație compatibil PCI

Pentru calculatoarele compatibile PC-AT bazate pe procesoare x86, specificațiile magistralei PCI (versiunea 2.3) definesc o metodă care utilizează accese de I/E pentru a solicita ca puntea UCP-PCI să execute tranzacții de configurație pentru accesul la registrele de configurație ale funcțiilor PCI. Specificațiile nu definesc un mecanism de configurație pentru alte calculatoare, diferite de cele compatibile PC-AT.

Din cauza limitării spațiului de I/E al procesoarelor x86 la 64 KB, registrele de configurație nu se pot mapa direct în spațiul de I/E al procesorului. Accesul la aceste registre se poate realiza indirect prin intermediul a două porturi de I/E de 32 de biți implementate în puntea UCP-PCI (în cazul magistralei PCIe, puntea este situată în complexul rădăcină). Aceste porturi sunt următoarele:

- Portul adreselor de configurație, cu adresa de I/E 0x0CF8.
- Portul datelor de configurație, cu adresa de I/E 0x0CFC.

Accesul la unul din registrele de configurație compatibile PCI ale unei funcții PCIe se realizează în două etape:

1. Se înscrie în portul adreselor de configurație numărul magistralei PCI, numărul dispozitivului, numărul funcției și adresa registrului de configurație (numărul cuvântului dublu), în formatul ilustrat în figura 2.9, și se setează bitul *Enable* al acestui port (bitul 31) la 1.
2. Se execută o citire din sau o scriere în portul datelor de configurație a unui cuvânt dublu (32 de biți).

Ca răspuns la aceste operații, puntea UCP-PCI din complexul rădăcină compară numărul magistralei specificate cu numerele magistralelor conectate la acea punte și, în cazul în care magistrala specificată este conectată la punte, inițiază o tranzacție de citire sau de scriere în spațiul de configurație (după cum procesorul execută o operație de citire sau de scriere cu portul datelor de configurație).

Portul adreselor de configurație memorează informațiile înscrise în acest port numai atunci când procesorul execută o scriere de 32 de biți în port. Deci, memorarea informațiilor nu este posibilă prin mai multe operații de scriere de 8 sau 16 biți în port. O citire de 32 de biți din port returnează conținutul acestuia. Informațiile înscrise în portul adreselor de configurație trebuie să fie organizate în modul prezentat în figura 2.9.

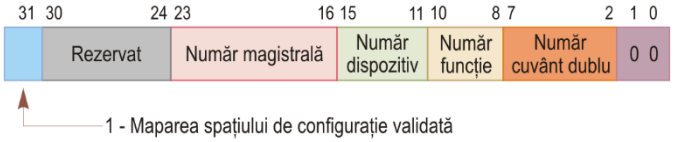

Figura 2.9. Structura informațiilor înscrise în portul adreselor de configurație.

Semnificația câmpurilor din portul adreselor de configurație este următoarea:

- Bitul 31 reprezintă bitul de validare pentru maparea spațiului de configurație. Acest bit trebuie setat la 1 pentru a valida translatarea unui acces ulterior al procesorului la portul datelor de configurație într-o tranzacție de acces în spațiul de configurație. Dacă acest bit este setat la 0 și procesorul inițiază un acces la portul datelor de configurație, operația va fi translatată într-o tranzacție de acces în spațiul de I/E și nu în spațiul de configurație.
- Biții 30..24 sunt rezervați și trebuie setați la 0.
- Biții 23..16 identifică numărul magistralei PCI (0..255).
- Biții 15..11 identifică numărul dispozitivului (0..31).
- Biții 10..8 identifică numărul funcției (0..7) din cadrul dispozitivului.
- Biții 7..2 identifică registrul de configurație al funcției prin numărul cuvântului dublu (0..63) din spațiul de configurație compatibil PCI al funcției.
- Biții 1..0 sunt setați la 0 și nu pot fi modificați.

2.6.3. Mecanismul de configurație îmbunătățit PCIe

Mecanismul de configurație îmbunătățit PCIe realizează maparea spațiului de configurație al arhitecturii PCIe în spațiul adreselor memoriei principale. Fiecărui controler PCIe dintr-un sistem i se alocă o zonă din memoria principală. La inițializarea sistemului, programul BIOS determină adresa de bază a zonei din memoria principală alocată unui controler PCIe și o comunică complexului rădăcină și sistemului de operare. Metoda de comunicare este specifică implementării și nu este definită în specificațiile magistralei PCIe.

Fiecărui registru de configurație al unei funcții PCIe i se asignează o anumită adresă de memorie din zona alocată controlerului PCIe al acelei funcții. Complexul rădăcină al sistemului PCIe monitorizează accesele la memorie, iar dacă detectează un acces în zona de 256 MB alocată controlerului PCIe, inițiază o tranzacție de configurație pentru accesul la spațiul de configurație PCIe.

Spațiul de configurație al fiecărei funcții PCIe începe de la o adresă aliniată la 4 KB din zona de memorie de 256 MB. Structura adresei care trebuie specificată pentru accesul la un anumit octet, cuvânt sau cuvânt dublu din spațiul de configurație al unei funcții PCIe este următoarea:

- Biții 63..28 reprezintă adresa de bază a zonei de memorie de 256 MB alocată ca spațiu de configurație al controlerului PCIe corespunzător funcției PCIe.
- Biții 27..20 selectează magistrala PCI (0..255).
- Biții 19..15 selectează dispozitivul (0..31).
- Biții 14..12 selectează funcția (0..7) din cadrul dispozitivului.
- Biții 11..2 selectează cuvântul dublu (0..1023) din spațiul de configurație al funcției PCIe.
- Biții 1..0 indică deplasamentul octetului (0..3) din cadrul cuvântului dublu selectat.

Pentru accesul la spațiul de configurație extins al unei funcții PCIe este necesar să se determine adresa de bază a zonei de memorie alocată ca spațiu de configurație pentru funcțiile PCIe ale unui controler PCIe. Există mai multe metode care se pot utiliza pentru determinarea acestei adrese de bază. De exemplu, una din metode se bazează pe căutarea în memorie a anumitor tabele ale sistemului și accesul la aceste tabele. Aceste metode nu sunt prezentate în această lucrare de laborator.

Observație

- Fișierul sursă PCIBaseAddressUEFI.cpp, disponibil pe pagina laboratorului, conține funcția `PciBaseAddressUEFI()`, care returnează adresa de bază a spațiului de configurație extins PCIe ca o valoare de 64 biți (`DWORD64`).

2.6.4. Registre obligatorii ale antetului de configurație

După cum s-a prezentat în secțiunea 2.6.1, primele 16 cuvinte duble ale spațiului de configurație compatibil PCI al unui dispozitiv reprezintă antetul de configurație PCI. Specificațiile PCI definesc trei formate ale antetului, numite antet de tip 0, 1, respectiv 2. Antetul de tip 2 este definit pentru punți CardBus, antetul de tip 1 este definit pentru punți PCI-PCI, iar antetul de tip 0 este definit pentru toate celelalte dispozitive. În această lucrare de laborator, ne referim doar la antetul de tip 0.

Structura antetului de configurație de tip 0 este ilustrată în figura 2.8. O parte a registrelor din acest antet trebuie implementate de toate dispozitivele PCI sau PCIe, inclusiv de către punți. Aceste registre obligatorii sunt indicate cu o culoare mai întunecată în figura 2.8. Registrele obligatorii ale antetului de configurație sunt descrise în continuare.

Observație

- Fișierul antet PCI.h, care este utilizat pentru aplicații, definește antetul de configurație de tip 0 într-o structură numită `PCI_CONFIG0`.

Registrul ID al producătorului

Acest registru de 16 biți identifică producătorul dispozitivului. Identificatorul producătorului este asignat de organizația PCI SIG. Valoarea 0xFFFF este rezervată și este returnată de puntea UCP-PCI atunci când se încearcă execuția unei citiri dintr-un registru de configurație al unei funcții PCI inexistente.

Registrul ID al dispozitivului

Acest registru de 16 biți conține un identificator asignat de producătorul dispozitivului, prin care se identifică tipul dispozitivului. Împreună cu registrul ID al producătorului și,

eventual, cu registrul ID al reviziei, registrul ID al dispozitivului se poate utiliza pentru a identifica un driver specific funcției pentru dispozitivul respectiv.

Registrul ID al reviziei

Acest registru de 8 biți conține un identificator asignat de producătorul dispozitivului și care identifică numărul reviziei dispozitivului.

Registrul codului clasei

Structura acestui registru este ilustrată în figura 2.10. Este un registru de 24 de biți divizat în trei câmpuri de câte 8 biți: codul clasei (octetul superior), codul sub-clasei (octetul mijlociu) și interfața de programare (octetul inferior). Acest registru identifică funcția de bază a dispozitivului (de exemplu, un controler de stocare masivă), o sub-clasă mai specifică a dispozitivului (cum este un controler de stocare masivă SATA) și, în anumite cazuri, o interfață de programare cu un set de registre specific.

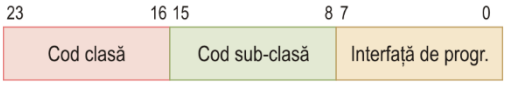

Figura 2.10. Structura registrului pentru codul clasei.

Pentru numeroase combinații ale codului clasei și codului sub-clasei, octetul interfeței de programare returnează valoarea zero, astfel încât nu are semnificație. Pentru alte combinații însă, octetul interfeței de programare are semnificație, deoarece identifică aranjamentul exact al setului de registre al funcției, care poate varia de la o implementare la alta. De exemplu, există diferite tipuri de controlere USB cu același cod al clasei și cod al sub-clasei, dar cu interfețe de programare diferite (UHCI, OHCI, EHCI și XHCI).

Observație

- Fișierul antet PCI.h conține codurile clasei, codurile sub-clasei și interfețele de programare definite în prezent, într-o structură numită `PCI_CLASS_TABLE`. Această structură conține și pointeri la doi descriptori (texte) care se pot utiliza pentru decodificarea informației conținute în registrul codului clasei: primul este un descriptor al clasei și al sub-clasei, iar al doilea este un descriptor al interfeței de programare.

Registrul de comandă

Acest registru de 6 biți permite un control de bază asupra posibilităților dispozitivului de a executa tranzacții PCI sau PCIe. Conține biți care permit validarea sau invalidarea decodificatorului pentru spațiul adreselor de I/E, validarea sau invalidarea decodificatorului pentru spațiul adreselor de memorie, validarea sau invalidarea posibilității funcției de a genera cereri de acces la memorie sau cereri de I/E, validarea sau invalidarea raportării erorilor detectate de către funcție și validarea sau invalidarea posibilității funcției de a genera mesaje de întrerupere INTx. Registrul de comandă nu este descris în detaliu în această lucrare de laborator.

Registrul de stare

Acest registru de stare de 16 biți păstrează starea evenimentelor legate de magistrala PCI sau PCIe. Conține biți care indică starea întreruperilor (dacă funcția are o cerere de întrerupere în curs), dacă s-a detectat o eroare de paritate, sau dacă o tranzacție a fost abandonată de către dispozitivul destinație sau inițiator. Registrul de stare nu este descris în detaliu în această lucrare de laborator.

O parte a biților acestui registru au atributul RO (*Read Only*), în timp ce alți biți au atributul R/W (*Read/Write*). O caracteristică particulară a biților care pot fi înscriși este faptul că aceștia pot fi resetați, dar nu pot fi setați. Un bit poate fi resetat prin înscrierea acestuia cu valoarea 1; acest atribut este notat cu RW1C (*Read/Write 1 to Clear*). Această metodă a fost aleasă pentru a simplifica programarea. După citirea stării și identificarea biților de eroare

care sunt setați, programatorul poate șterge acești biți prin scrierea valorii citite înapoi în registru.

Registrul tipului antetului

Biții 6..0 ai acestui registru de 8 biți definesc tipul antetului de configurație. În prezent sunt definite următoarele tipuri de antet:

0: Funcție diferită de o punte;
1: Punte PCI(X)-PCI(X);
2: Punte CardBus.

Bit 7 definește dispozitivul ca un dispozitiv cu o singură funcție (dacă bitul 7 este 0) sau un dispozitiv cu funcții multiple (dacă bitul 7 este 1). În timpul configurării, programatorul poate testa starea acestui bit pentru a determina dacă mai există alte funcții ale dispozitivului care trebuie configurate.

Registrele ID subsistem al producătorului și ID subsistem

Registrul de 16 biți ID subsistem al producătorului conține un identificator asignat de organizația PCI SIG. Registrul de 16 biți ID subsistem conține un identificator asignat de producătorul funcției. O valoare zero citită din aceste registre indică faptul că nu există un identificator subsistem al producătorului și un identificator subsistem asociat cu funcția respectivă.

O funcție poate fi amplasată pe o placă de extensie sau într-un dispozitiv înglobat. Funcțiile care sunt proiectate pe baza acelorași circuite nucleu PCI, PCI-X sau PCIe pot avea același identificator al producătorului și identificator al dispozitivului asignate de către producătorul circuitelor nucleu. În acest caz, sistemul de operare nu ar putea identifica driverul corect care trebuie utilizat pentru acea funcție. Registrele ID subsistem al producătorului și ID subsistem se utilizează pentru a identifica în mod unic placa de extensie sau subsistemul în care este amplasată funcția. Ca urmare, sistemul de operare poate realiza distincția dintre plăci sau subsisteme fabricate de producători diferiți dar care sunt proiectate pe baza acelorași circuite nucleu.

2.6.5. Registre opționale ale antetului de configurație

Cele mai importante registre opționale ale antetului de configurație sunt descrise în continuare.

Registrul BIST

Acest registru poate fi implementat atât de către o funcție inițiator, cât și de către o funcție destinație. Dacă o funcție dispune de o operație de autotest BIST (*Built-In Self-Test*), trebuie să implementeze acest registru, cu structura ilustrată în figura 2.11. Dacă bitul 7 al registrului BIST este 1, aceasta înseamnă că funcția dispune de o operație BIST. Dacă funcția nu dispune de o operație BIST, acest registru va returna valoarea zero la citire. Operația BIST a funcției este invocată prin setarea bitului 6 la 1. Funcția trebuie să termine operația BIST într-o limită de timp de două secunde, iar apoi trebuie să reseteze bitul 6. Rezultatul testului este indicat în biții 3..0 ai registrului. Un cod de terminare zero indică terminarea cu succes a testului. O valoare diferită de zero reprezintă un cod de eroare specific funcției.

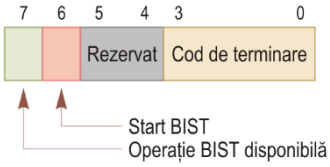

Figura 2.11. Structura registrului BIST.

Registre ale adreselor de bază

Majoritatea funcțiilor utilizează un spațiu al adreselor de memorie și/sau un spațiu al adreselor de I/E pentru a implementa un set de registre specifice funcției, registre utilizate pentru a controla funcția și pentru a identifica starea sa. La punerea sub tensiune, sistemul trebuie configurat astfel încât spațiile de memorie și de I/E ale fiecărei funcții să ocupe domenii de adrese mutual exclusive. De aceea, sistemul trebuie să detecteze care sunt spațiile de memorie și de I/E necesare unei anumite funcții. În plus, sistemul trebuie să programeze decodificatoarele de adrese ale funcției pentru a asigna domenii de adrese neconflictuale acestora.

Registrele adreselor de bază (BAR – *Base Address Register*) sunt amplasate în cuvintele duble 4..9 ale spațiului antetului și se utilizează pentru a implementa decodificatoarele programabile de memorie și/sau de I/E ale unei funcții. Fiecare registru este de 32 biți sau 64 biți (în cazul unui decodificator de memorie al cărui bloc de memorie asociat poate fi amplasat deasupra spațiului de 4 GB). Bitul 0 poate fi doar citit și indică dacă registrul este un decodificator de memorie sau un decodificator de I/E:

Bit 0 = 0: registrul este un decodificator al adreselor de memorie;
Bit 0 = 1: registrul este un decodificator al adreselor de I/E.

Decodificatoarele pot fi implementate în oricare din registrele adreselor de bază. În timpul configurării, programul de configurare trebuie să verifice toate cele șase registre ale adreselor de bază din antetul de configurație al unei funcții pentru a determina care din registre sunt implementate efectiv.

Structura unui registru al adresei de bază pentru memorie

Un registru al adresei de bază pentru memorie poate avea o dimensiune de 32 biți sau 64 biți. Figura 2.12 ilustrează structura unui registru de 64 biți al adresei de bază pentru memorie. Bitul 0 este zero și indică un decodificator al adreselor de memorie. Biții 2..1 definesc dimensiunea decodificatorului de memorie:

00: decodificator de memorie de 32 biți;
10: decodificator de memorie de 64 biți.

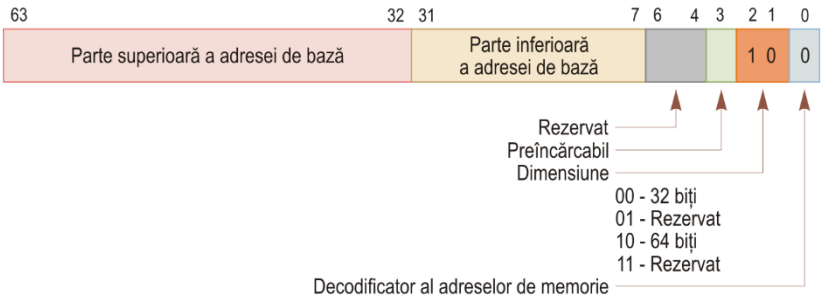

Figura 2.12. Structura unui registru de 64 biți al adresei de bază pentru memorie.

În cazul unui decodificator de memorie de 32 biți, registrul adresei de bază conține o adresă de început a memoriei din primii 4 GB ai spațiului adreselor de memorie. În cazul unui decodificator de memorie de 64 biți, registrul adresei de bază conține o adresă de început oriunde în spațiul adreselor de memorie de 2^{64} octeți. În acest caz, registrul adresei de bază ocupă două cuvinte duble consecutive în spațiul antetului de configurație. Primul cuvânt dublu conține cei 32 de biți inferiori ai adresei de început a memoriei, iar al doilea cuvânt dublu conține cei 32 de biți superiori ai adresei de început a memoriei.

Bitul 3 indică dacă blocul de memorie este preîncărcabil (bit 3 = 1) sau nu (bit 3 = 0). În cazul unui bloc de memorie preîncărcabil, este acceptabil ca o punte dintre un inițiator și o destinație din memorie să încarce în avans datele din memorie într-un buffer pentru a obține performanțe superioare.

Biții 31..7 ai unui decodificator de memorie de 32 biți și biții 63..7 ai unui decodificator de memorie de 64 biți conțin adresa de bază a memoriei.

Pentru fiecare registru al adresei de bază, programul de configurare trebuie să determine dacă registrul este implementat, care este dimensiunea registrului (32 biți sau 64 biți) și care este dimensiunea spațiului de memorie corespunzător registrului. Dimensiunea spațiului de memorie se poate determina utilizând următoarea procedură:

1. Se citește conținutul registrului adresei de bază într-o variabilă temporară.
2. Se înscrie valoarea constând din toți biții setați la unu în registrul adresei de bază.
3. Se citește conținutul registrului adresei de bază. Dacă valoarea citită este zero, aceasta indică faptul că registrul adresei de bază nu este implementat și procedura este terminată.
4. Dacă valoarea citită nu este zero, se parcurg biții valorii citite în sus începând cu bitul cel mai puțin semnificativ al câmpului adresei de bază (bit 7) până când se găsește primul bit setat la unu. Valoarea ponderată cu puterea lui 2 a primului bit setat la unu reprezintă dimensiunea spațiului de memorie asociat cu registrul adresei de bază.
5. Se reface conținutul registrului adresei de bază din variabila temporară.

Ca un exemplu, presupunem că se înscrie valoarea 0xFFFFFFFF într-un registru al adresei de bază, iar valoarea citită din registru este 0xFFF00000. Deoarece valoarea citită nu este zero, registrul este implementat. Pentru că bitul 0 este zero și biții 2..1 sunt 00, registrul este un decodificator de 32 biți ai adreselor de memorie. Bitul 20 este primul bit setat la unu din câmpul adresei de bază. Valoarea ponderată cu puterea lui 2 a acestui bit este 2^{20}, ceea ce înseamnă că dimensiunea spațiului de memorie corespunzător acestui registru este 1 MB.

Structura unui registru al adresei de bază de I/E

Un registru al adresei de bază de I/E are o dimensiune de 32 biți. Figura 2.13 ilustrează structura unui registru al adresei de bază de I/E. Bitul 0 este unu și indică un decodificator al adreselor de I/E. Bitul 1 este rezervat și returnează întotdeauna zero la citire. Biții 31..2 reprezintă câmpul adresei de bază.

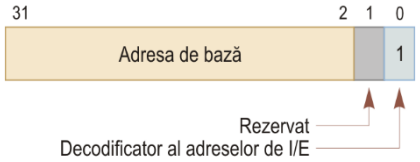

Figura 2.13. Structura unui registru al adresei de bază de I/E.

Cei 16 biți superiori ai unui registru al adresei de bază de I/E pot fi setați la zero de către producător atunci când o funcție este proiectată în mod specific pentru un calculator compatibil PC, deoarece procesoarele Intel x86 sunt limitate la un spațiu de I/E de 64 KB.

Dimensiunea spațiului de I/E corespunzător unui registru al adresei de bază de I/E se poate determina utilizând aceeași procedură ca și cea pentru determinarea dimensiunii spațiului de memorie. Singura deosebire este că bitul cel mai puțin semnificativ al câmpului adresei de bază este bitul 2 în locul bitului 7. Ca un al doilea exemplu, presupunem că se înscrie valoarea 0xFFFFFFFF într-un registru al adresei de bază, iar valoarea citită din registru este 0xFFFFFF01. Bitul 0 este unu, indicând faptul că registrul este un decodificator al adreselor de I/E. Parcurgând biții în sus începând cu bitul 2, bitul 8 este primul bit setat la unu în câmpul adresei de bază. Valoarea ponderată cu puterea lui 2 a acestui bit este 2^8, ceea ce înseamnă că dimensiunea spațiului de I/E corespunzător acestui registru este de 256 octeți.

2.7. Aplicații

2.7.1. Răspundeți la următoarele întrebări:

a. Care sunt îmbunătățirile introduse de magistrala PCIe comparativ cu magistralele precedente PCI și PCI-X?

b. Care sunt principalele componente ale topologiei magistralei PCIe?

c. Care sunt metodele care pot fi utilizate de dispozitivele PCIe pentru semnalarea întreruperilor?

d. Care sunt parametrii funcționali determinați în timpul inițializării și antrenării legăturii?

e. Care sunt registrele de configurație ce trebuie utilizate pentru identificarea unică a unei funcții PCIe?

2.7.2. Creați o aplicație Windows pentru identificarea fiecărui dispozitiv PCIe din calculator. Ca model pentru aplicația Windows, utilizați aplicația AppScroll, ale cărei fișiere sursă sunt disponibile pe pagina laboratorului în arhiva AppScroll.zip. Executați următoarele operații pentru a crea proiectul aplicației:

1. În mediul de programare *Visual Studio 2022*, creați un nou proiect gol de tip *Windows Desktop* cu utilitarul *Windows Desktop Wizard*. Bifați opțiunea *Place solution and project in the same directory* pentru a evita crearea unui alt director pentru soluția creată.
2. Verificați dacă platforma activă a soluției este setată la x64.
3. Modificați proprietatea *Character Set* a proiectului deschizând fereastra de dialog *Properties*. În această fereastră, expandați opțiunea *Configuration Properties*, expandați opțiunea *Advanced*, selectați linia *Character Set* în panoul din dreapta și selectați opțiunea *Not Set*.
4. Copiați în directorul proiectului fișierele din arhiva AppScroll.zip și adăugați la proiect aceste fișiere.
5. Copiați în directorul proiectului fișierele Hw.h și Hw64.lib din directorul unui proiect creat anterior.
6. Copiați în directorul proiectului fișierele antet și fișierul sursă din arhiva PCI.zip, disponibilă pe pagina laboratorului.
7. Adăugați la proiect fișierele PciBaseAddressUEFI.cpp, Hw.h, Pci-vendor-dev.h și PCI.h.
8. Specificați fișierul Hw64.lib ca o dependență suplimentară pentru linkeditor.
9. În fișierul sursă AppScroll.cpp, adăugați directive `#include` pentru a include fișierele antet PCI.h și Pci-vendor-dev.h. Declarați `PciBaseAddressUEFI()` ca o funcție fără parametri care returnează o valoare de tip `DWORD64`.

În fișierul sursă AppScroll.cpp, apelați mai întâi funcția `PciBaseAddressUEFI()` pentru a determina adresa de bază a spațiului de configurație extins PCIe și memorați adresa de bază returnată într-o variabilă globală. Dacă funcția returnează 0, adresa de bază nu se poate determina cu succes, iar în acest caz aplicația trebuie să revină cu un cod de eroare. În caz contrar, afișați adresa de bază sub forma a două cuvinte duble. În continuare, scrieți o funcție care returnează un pointer la antetul de configurație al unei funcții PCIe utilizând mecanismul de configurație îmbunătățit PCIe. Funcția are ca parametri de intrare numărul magistralei, numărul dispozitivului și numărul funcției PCIe, și returnează un pointer la o structură de tip `PCI_CONFIG0` conținând antetul de configurație al funcției PCIe. Mecanismul de configurație îmbunătățit este descris în secțiunea 2.6.3. În această funcție, utilizați variabila globală care conține adresa de bază a spațiului de configurație extins PCIe. Apoi, utilizați această funcție pentru căutarea dispozitivelor PCIe de pe fiecare magistrală între 0 și 63, pentru fiecare dispozitiv (0..31) și pentru fiecare funcție (0..7) a unui dispozitiv. Pentru fiecare dispozitiv PCIe existent, trebuie să se afișeze următoarele informații (pe linii separate):

- Numărul magistralei, numărul dispozitivului, numărul funcției;

- Codul clasei, codul sub-clasei, interfața de programare, identificatorul subsistemului producătorului, identificatorul subsistemului;
- Descriptorul clasei/sub-clasei, descriptorul interfeței de programare.

Utilizați structurile definite în fișierul antet PCI.h. Pentru afișarea descriptorului clasei/sub-clasei și a descriptorului interfeței de programare, căutați în tabloul `PciClassTable` utilizând codul clasei, codul sub-clasei și interfața de programare ca și chei de căutare.

Observații

- Dacă registrul ID al producătorului unei funcții PCIe returnează valoarea 0xFFFF la citire, înseamnă că funcția respectivă nu există. În acest caz, nu trebuie afișat nici un mesaj, deoarece numeroase funcții nu există.
- Registrele de configurație ale unei funcții PCIe nu trebuie accesate direct, ci prin intermediul driverului Marvin HW. De exemplu, presupunând că `pRegPci` este un pointer la antetul de configurație al unei funcții, registrul ID al producătorului poate fi citit în variabila `wVendorID` astfel:

```
wVendorID = _inmw((DWORD_PTR)&pRegPci->VendorID);
```

2.7.3. Extindeți aplicația 2.7.2 pentru a afișa informații suplimentare despre dispozitivele PCIe existente din calculator. Informațiile suplimentare care trebuie afișate sunt următoarele:

- Identificatorul producătorului, descriptorul producătorului;
- Identificatorul dispozitivului, descriptorii circuitului.

Utilizați fișierul antet PCI-vendor-dev.h adăugat la proiect. Pentru a afișa descriptorul producătorului, căutați în tabloul `PciVenTable` utilizând identificatorul producătorului ca și cheie de căutare și afișați membrul `CONST CHAR *VenFull` al structurii `PCI_VENTABLE`. Pentru a afișa descriptorii circuitului, căutați în tabloul `PciDevTable` utilizând identificatorul producătorului și identificatorul dispozitivului ca și chei de căutare și afișați membrii `CONST CHAR *Chip` și `CONST CHAR *ChipDesc` ai structurii `PCI_DEVTABLE`.

Observații

- Numărul de intrări din tabloul `PciVenTable` este definit ca `PCI_VENTABLE_LEN`.
- Numărul de intrări din tabloul `PciDevTable` este definit ca `PCI_DEVTABLE_LEN`.
- Atunci când identificatorul unui dispozitiv nu se găsește în tabloul `PciDevTable`, este posibil să se găsească anumite informații despre acel dispozitiv în fișierul pci.ids de pe pagina *The PCI ID Repository* (http://pci-ids.ucw.cz/), fișier care este disponibil și pe pagina laboratorului.

2.7.4. Extindeți aplicația 2.7.3 pentru a afișa aceleași informații despre dispozitivele PCIe ca și cele specificate în aplicația 2.7.2, dar, de data aceasta, utilizând mecanismul de configurație compatibil PCI pentru accesarea spațiului de configurație. Acest mecanism este descris în secțiunea 2.6.2. Mai întâi, scrieți o funcție care citește conținutul unui singur cuvânt dublu din antetul de configurație al unei funcții PCIe utilizând mecanismul de configurație compatibil PCI. Funcția are următorii parametri de intrare: numărul magistralei, numărul dispozitivului, numărul funcției PCIe și numărul cuvântului dublu. Funcția returnează conținutul cuvântului dublu specificat. Apoi, utilizați această funcție pentru citirea registrelor care au fost utilizate pentru aplicația 2.7.2, specificând numărul cuvântului dublu corespunzător la apelul funcției (numerele cuvintelor duble sunt indicate în figura 2.8). Extrageți informațiile relevante din aceste registre (de exemplu, codul clasei, codul sub-clasei, interfața de programare) și afișați aceleași informații ca și cele afișate pentru aplicația 2.7.2.

2.7.5. Creați o aplicație Windows pentru identificarea controlerului SMBus (*System Management Bus*) al calculatorului și afișarea conținutului registrelor adreselor de bază ale

acestui controler. După crearea proiectului, scrieți o funcție care caută un dispozitiv PCIe cu codul clasei și codul sub-clasei specificați ca parametri. Funcția returnează într-un cuvânt dublu numărul magistralei, numărul dispozitivului și numărul funcției dispozitivului PCIe; funcția returnează 0 dacă dispozitivul PCIe cu codul clasei și codul sub-clasei specificate nu poate fi găsit. Funcția caută dispozitivul PCIe specificat pe fiecare magistrală între 0 și 63, fiecare dispozitiv (0..31) și fiecare funcție PCIe (0..7) a unui dispozitiv. Funcția utilizează mecanismul de configurație îmbunătățit PCIe pentru accesul la spațiul de configurație. Apelați această funcție pentru identificarea controlerului SMBus, pentru care codul clasei este 0x0C și codul sub-clasei este 0x05. După identificarea controlerului, afișați numărul magistralei sale, numărul dispozitivului și numărul funcției. Apoi, pentru fiecare din cele șase registre ale adreselor de bază ale controlerului executați următoarele operații:

- Afișați tipul registrului (decodificator de memorie sau de I/E);
- Dacă registrul este un decodificator de memorie, afișați dimensiunea acestuia (32 biți sau 64 biți) și adresa de bază corespunzătoare a memoriei;
- Dacă registrul este un decodificator de I/E, afișați adresa de bază corespunzătoare de I/E.

Bibliografie

[1] Ajanovic, J., “PCI Express (PCIe) 3.0 Accelerator Features”, Intel Corporation, 2008, http://www.intel.com/content/dam/doc/white-paper/pci-express3-accelerator-white-paper.pdf.

[2] Bhatt, A. V., “Creating a PCI Express Interconnect”, Intel Corporation, 2002, http://www.advancedaudiorentals.com/phpkb/admin/attachments/pci_express_white_paper.pdf.

[3] Budruk, R., Anderson, D., Shanley, T., *PCI Express System Architecture*, MindShare Inc., Addison-Wesley Developer’s Press, 2008, https://www.mindshare.com/files/ebooks/PCI%20Express%20System%20Architecture.pdf.

[4] PCI-SIG, “PCI Code and ID Assignment Specification”, Revision 1.11, January 24, 2019.

[5] PCI-SIG, “PCI Express Base Specification Revision 3.0”, November 10, 2010.

[6] Shanley, T., Anderson, D., *PCI System Architecture*, Fourth Edition, MindShare Inc., Addison-Wesley Developer’s Press, 1999.

[7] PCI Vendor and Device Lists, http://www.pcidatabase.com.

[8] The PCI ID Repository, https://pci-ids.ucw.cz.