OBSAH

Úvod

Zvolil jsem si pro svou bakalářskou práci téma sběrnice PCI Express , protože je to na poli výpočetní techniky velice zajímavé a poměrně hodně skloňované téma poslední doby. Velikou motivací pro mne byl fakt, že jsem o tomto systému nic nevěděl a chtěl jsem tedy nabít znalosti o tomto druhu sběrnice.Specifickou částí mé práce je implementace a simulace bloků linkové vrstvy. Jak linková vrstva řadiče, tak i verifikační testbenche pro simulaci jednotlivých bloků, mají být napsány pro hradlové pole FPGA v jazyce VHDL.

Nutno podotknout, že v době vypracovávání neexistovalo moc zmínek o tomto systému v českém jazyce. Pokud nějaké texty existovaly v češtině, obsahovaly velice strohé informace. Takže jsem musel získat téměř veškeré informace z anglicky psané literatury. To je také důvod, proč jsem na několika místech používal jak termín psaný anglicky, tak i česky (pokud český ekvivalent existoval). Anglicky psané termíny jsou v textu označeny *kurzívou*. Ve své práci používám značnou část přejatých obrázků z uvedených zdrojů. Proto jsem i u mnou vytvořených obrázků ponechal anglicky psanou terminologii.

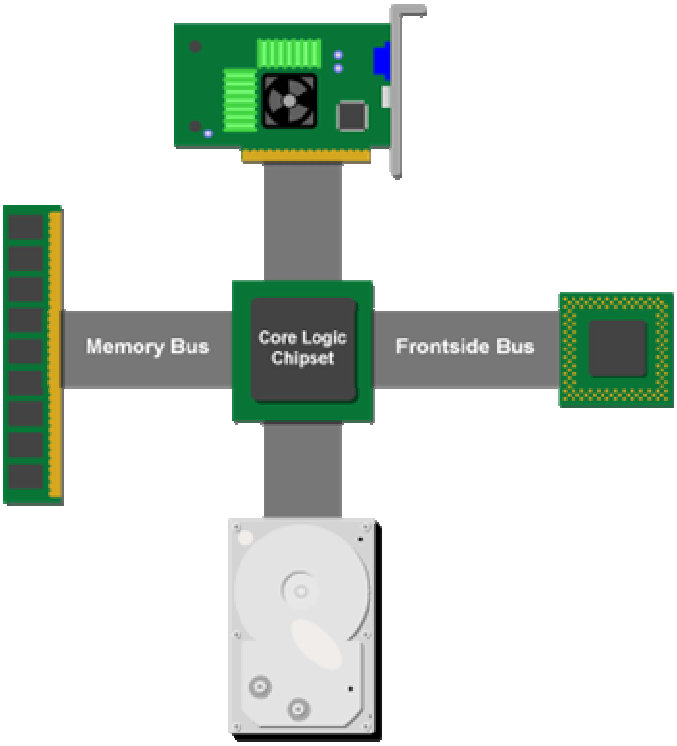
Protože většinu zdrojů tvořily pouze roztroušené články na internetu, nebo technické specifikace, myslím si, že jsem vytvořil první literaturu v českém jazyce, která podává ucelený pohled na tuto problematiku.

1 Základní architektura sběrnic v PC systému

V této kapitole jsou popsány důvody vzniku PCI Express a ve zkratce proveden úvod do problematiky sběrnicových systémů.

1.1 Architektura jednoho bloku

Průměrný systém založený na sběrnici PCI je znázorněn na Obr. 1-1.



Obr. 1-1 Rozvržení systému s PCI♣

Jádro logiky čipsetu funguje jako přepínač (*switch*) nebo přepojovač (*router*), který řídí vstupně/výstupní přenosy mezi jednotlivými zařízeními, které tvoří systém jako celek.

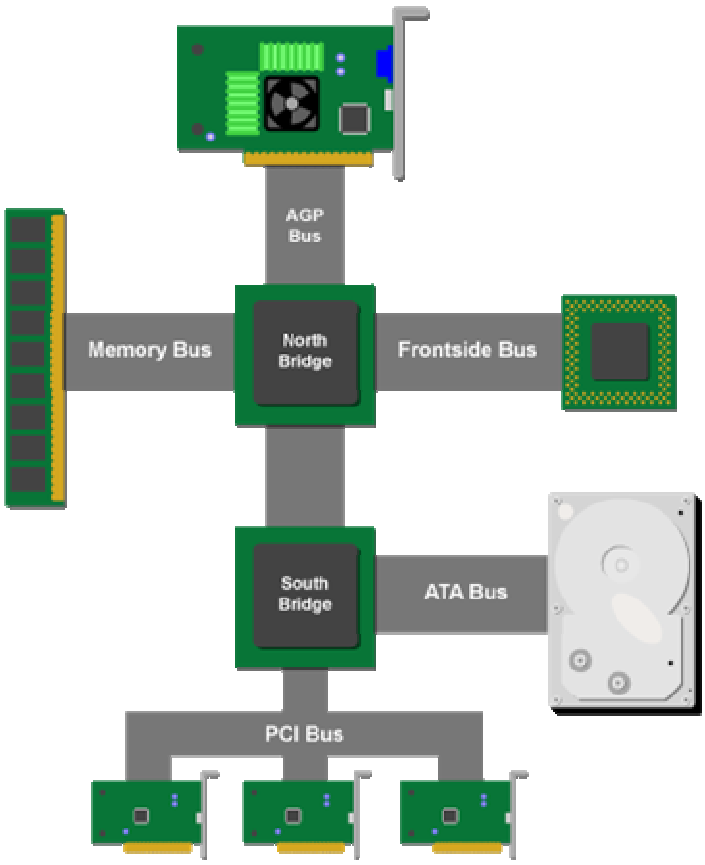
Ve skutečnosti je jádro logiky čipsetu rozděleno na dvě odlišné části. Na severní a jižní můstek (*Northbridge* a *Southbridge*), někdy též označovaný jako I/O můstek. Toto rozdělení je zde z několika důvodů. Nejdůležitějším z nich je skutečnost, že se zde vyskytují tři druhy zařízení, které potřebují velice úzce spolupracovat, a proto potřebují velmi rychlý vzájemný přístup.

♣ zdrojem obrázku je literatura [5]

Tyto zařízení jsou procesor (*CPU*), pracovní paměť (*main memory*, *RAM*), a zobrazovací adaptér (*video card*). V moderních systémech plní procesor video adaptéru (*GPU*) funkci druhého či třetího procesoru systému, a proto potřebuje sdílet privilegovaný přístup k pracovní paměti s procesorem (*CPU*). Následkem toho jsou tato tři zařízení seskupena dohromady na severním můstku.

1.2 Severní a jižní můstek

Severní můstek je svázán dohromady se sekundárním můstek. Jako sekundární můstek slouží jižní můstek, který přepojuje spojení mezi různými vstupně/výstupními zařízeními systému. Jsou to pevné disky (*hard drives*), USB porty, síťové porty (*Ethernet*), atd. Komunikace těchto zařízení je směrována skrz jižní můstek do severního můstku a potom na CPU nebo do paměti.



Obr. 1-2 Severní a jižní můstek ♣

♣ zdrojem obrázku je literatura [5]

Na obrázku Obr. 1-2 je znázorněna sběrnice PCI připojená k jižnímu můstku. Tato sběrnice je obvykle nejstarší, v moderním systému je nejpomalejší a nejvíce vyžaduje aktualizaci. Z obrázku je patrné, že v moderním PC je pestrá sbírka specializovaných sběrnicí s různými protokoly a o různých šířkách přenášených pásem.

1.3 Soubor sběrnic

Mix specializovaných sběrnicí navržených k tomu, aby připojil různé druhy hardwaru přímo k jižnímu můstku, byl vytvořen v důsledku vývoje počítačového průmyslu s limitujícími faktory stárnoucí sběrnice PCI. Protože sběrnice PCI nebyla uzpůsobena pro přenosy typů *Serial ATA* nebo *fire wire*, byl postupný trend připojit oba typy rozhraní (jak interní, tak externí I/O) přímo do jižního můstku. Takže dnešní jižní můstek je něco jako Švýcarský nůž, ve kterém jsou místo čepelí různé vstupně/výstupní přepínače, což samozřejmě celý systém značně zatěžuje.

V ideálním světě bude existovat pouze jeden primární typ sběrnice a jeden sběrnicový protokol, který bude dohromady spojovat všechna různá vstupně/výstupní zařízení včetně video adaptéru (*GPU*), procesoru (*CPU*) a pracovní paměti. Samozřejmě, tato idea ,,jedna sběrnice, která vládne všemu,, je ideál a nikdy nebude schopnost v rukou inženýrů zrealizovat ji v reálném světě. Nebude to možné zrealizovat ani s PCI Express, ani se sítí *Infiniband* (teoreticky by se to se systémem *Infiniband* zrealizovat mohlo, ale musel by se z PC odstranit veškerý dnes v PC používaný hardware a muselo by se vše od píky realizovat periferiemi vyhovujícími standardu *Infiniband*, což je vzhledem ke zpětné kompatibilitě zařízení a k přechodu na nový standard v praxi nerealizovatelné).

I když nebude utopického ideálu jedné sběrnice a jednoho sběrnicového protokolu pro každé zařízení nikdy dosaženo, musí existovat způsob, jak do dnešního chaosu přivést alespoň trochu řádu. Naštěstí něco takového přišlo v podobě nového standardu sběrnice *PCI Express*.

S Intelovským nedávným vypuštěním čipsetů série 900 a oznámení *ATI* a *NVIDIA* o nových *PCI Express* kartách bude brzy *PCI Express* sběrnicí pro několik příštích let.

1.4 Přiblížení na PCI

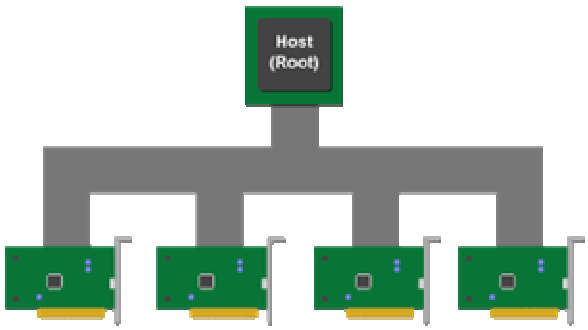
Dříve než se pustím do detailů v popisu *PCIe*, nastíním některá řešení na sběrnici *PCI*. Pomůže to lépe pochopit práci systému a funkční omezení stávající sběrnice *PCI*.

1.4.1 PCI standard

PCI sběrnice pracovala po dlouhou dobu na frekvenci 33 MHz, měla 32 bitovou šířku sběrnice a její teoretická vrcholová šířka byla 132 MB/s. Ovšem s postupujícím časem byly již tyto rychlosti pro zbývající části systému nedostatečné. Vývoj způsobil jak zvyšování pracovní frekvence, tak i zvětšení šířky sběrnice. Nejdříve byla sběrnice rozšířena na 64 bitů při stávající frekvenci 33 MHz, jejíž šířka pásma byla 264 MB/s. Ale i to se později zdálo nedostatečné. Proto nastoupil nový standard sběrnice PCI, který byl taktéž 64 bitový, ovšem na vyšší pracovní frekvenci. Sběrnice pracovala na 66 MHz a šířka přenášeného pásma byla již 512 MB/s.

1.4.1.1 Sdílená topologie

PCI používá sdílenou topologii sběrnice (*shared bus topology*), která umožňuje komunikaci mezi různými zařízeními na sběrnici. Různá *PCI* zařízení (např. síťová karta, zvuková karta, *RAID* řadič) jsou všechna připojená na stejnou sběrnici, kterou používají ke komunikaci s *CPU*. Na Obr. 1-3 je znázorněno jak sdílená sběrnice vypadá.



Obr. 1-3 Sdílená sběrnice PCI[♣]

Protože všechna zařízení napojená na sběrnici musí sdílet připojení mezi sebou, musí zde existovat nějaký způsob sběrnicového řízení (*bus arbitration*), které rozhodne, kdo a kdy dostává přístup ke sběrnici, zvláště v situacích, kdy vícenásobná zařízení (*multiple device*) potřebují použít sběrnici zároveň. Jedno zařízení má kontrolu nad sběrnicí, takže se stává hlavním zařízením (*bus master*), což znamená, že může používat *PCI* sběrnici k tomu, aby komunikovalo s CPU nebo s pamětí skrz čipsetový jižní můstek.

[♣] zdrojem obrázku je literatura [5]

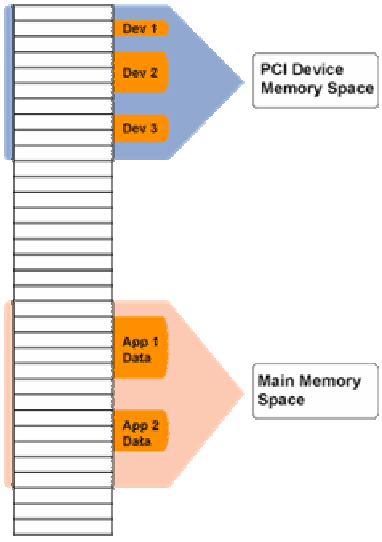
1.4.1.2 Komunikace můstků

Komunikace jižního můstku Obr. 1-2 ukazuje jak celek funguje v reálném světě, oproti ideální representaci Obr. 1-1 . Jižní můstek, severní můstek a CPU jsou spojeni dohromady aby mohli plnit funkci hostitele (*host*) nebo kořene (*root*). *Root* řídí a inicializuje *PCI* zařízení a standardně ovládá *PCI* sběrnici. Další způsob jak popsat funkci systému by mohl být takový, že účel *PCI* sběrnice je spojit vstupně/výstupní zařízení s kořenem tak, že kořen může do I/O zařízení zapisovat a číst z nich, a obecně mohou vstupně/výstupní zařízení skrz kořen komunikovat s pamětí nebo s okolním světem.

1.4.1.3 Výhody a nevýhody

Mezi hlavní výhody sdílené topologie sběrnice patří, že je to jednoduché, levné a snadno realizovatelné. Tyto výhody platí do té doby, než se na sdílené sběrnici budete snažit vytvořit něco speciálního. Jakmile se začne od sdílené sběrnice vyžadovat vyšší výkon a větší funkčnost, tak se začneme setkávat s jejími omezeními. Nyní se pokusím některá z těchto omezení popsat. Tato omezení vyústila ve vytvoření nového standardu, a tím je *PCI Express*.

1.4.1.3.1 Paměťový prostor



Z perspektivy procesoru jsou *PCI* zařízení přístupná skrze přímo mapovaný paměťový systém (*load-store mechanism*). Využívá se kus adresového prostoru paměti, jež je využit pouze PCI zařízeními. Pro procesor vypadá tento kus paměti stejně jako jiný kus paměti s jiným obsahem. Hlavním rozdílem je to, že v každé řadě adres je umístěno jedno *PCI* zařízení namísto bloku paměťových buněk obsahujících kód nebo data.

Obr. 1-4 Paměťový prostor ♣

♣ zdrojem obrázku je literatura [5]

Takže úplně stejným způsobem jak procesor přistupuje k adresnímu prostoru paměti pro čtení nebo zápis, tak přistupuje k zařízením *PCI* skrze čtení a zápis na jejich absolutní adresy.

1.4.1.3.2 Inicializace po restartu

Při startu počítače vybaveného sběrnicí *PCI* se musí inicializovat *PCI* podsystém přidělením pořádného kusu paměti pro adresování *PCI* zařízení tak, aby k nim mohl procesor přistupovat. Jakmile jsou jednotlivá zařízení inicializována a vědí, které části adresního prostoru jsou jejich, tak mohou začít zpracovávat jakékoliv příkazy či data, které se na jejich adresním prostoru mohou vyskytovat. Jakmile *PCI* zařízení ví, že jeho adresní prostor byl umístěn na sběrnici, tak může začít číst jakákoliv data následující za adresou.

Podle tohoto schématu je systém funkční pouze v případě, že se na sběrnici vyskytuje jen několik málo zařízení, která naslouchají adresy či data. Ale z přirozeného schématu sběrnice vyplývá, že jakékoliv zařízení, které na sběrnici naslouchá, produkuje na sběrnici jisté množství šumu. Takže čím více je na sběrnici připojených zařízení, tím více šumu se na sběrnici vyskytuje a tím se stává těžší dostat čistý signál skrz tuto sběrnici.

1.4.1.3.3 Sdílená sběrnice

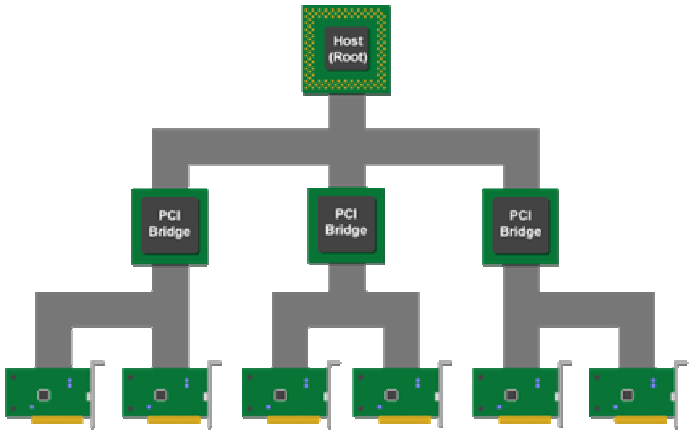
Šumový fenomén sdílené sběrnice spolu s problémy s časováním (šikmá náběžná hrana – *skew*) jsou důvod, proč je sběrnice PCI omezena maximálně na 5 rozšiřujících karet. Pokud jsou zařízení mimo sloty přímo napájené na základní desku, je šum nižší a může být počet zařízení na sběrnici o něco málo vyšší.

V praxi tento jev znamená, že pokud je potřeba k počítači připojit více než pět zařízení, tak se musí použít *PCI-to-PCI* můstkové čipy.

1.4.1.3.4 Stromová struktura

Hierarchická stromová struktura na Obr. 1-5 je jedním z rysů, která rozlišuje *PCI* rovnocennou síťovou komunikaci (*peer-to-peer*) na komunikaci dvoubodovou (point-to-point). Tato komunikace nové generace se spojuje jako *Hyper transport* nebo *Infiniband*. Kořen na vrcholu schématu funguje jako hlavní kontrolér, který je zodpovědný za inicializaci a konfiguraci všech *PCI* zařízeních při startování počítače. To způsobí, že každé *PCI* zařízení pracuje jako podřízené (*slave*), jež je ovládáno nadřazeným zařízením

(master). A protože nadřazené zařízení musí mít všechna podřazená zařízení namapována, tak nemohou existovat žádná nová připojení za provozu (*hot-plugging*).



Obr. 1-5 PCI-to-PCI most[♣]

1.4.1.3.5 Organizace sběrnicového provozu

Obecně vzato, existují dva typy kategorií, do kterých mohou být veškeré sběrnicové přenosy umístěny. První kategorií jsou adresové a datové přenosy. Adresa je umístění specifického zařízení (nebo oblast uvnitř specifického zařízení), kam jsou posílána data pro ni určená. Takže jakákoliv sběrnice, která podporuje vícenásobná zařízení, bude potřebovat znát způsob, jak zacházet s adresovými a s datovými přenosy, a jak je rozlišovat.

1.4.1.3.6 Přenosová architektura

Do druhé kategorie, která překrývá první kategorii, patří řídicí přenosy a přenosy typu zápis/čtení. Příkaz sestává z dat, která obsahují nějaký typ konfigurace nebo řídicí informace (specifický typ dat), který je poslán do nějakého specifického zařízení (jednotlivá adresa) na sběrnici. Příkazové přenosy obsahují obojí, jak adresové tak datové přenosy. Jako příklad příkazových přenosů může být inicializující instrukce pro zařízení,instrukce resetující zařízení, konfigurační příkaz který přepíná zařízení mezi jednotlivými pracovními režimy. Příkazové přenosy dovolují procesoru řídit, jak bude PCI zařízení ovládat data proudící z něj a do něj.

Přenosy typu čtení/zápis jsou nejdůležitějším typem přenosů, protože obsahují aktuální informaci, která je posílána do zařízení. Například řadič *PCI RAID* užívá přenosy typu čtení/zápis k tomu,aby přenášel aktuální soubory mezi pevnými disky, *PCI* zvuková

[♣] zdrojem obrázku je literatura [5]

karta využívá přenosy typu čtení/zápis k tomu, aby dostala zvuková data na výstupní port reproduktorů, atd. Tak jako příkazové přenosy, tak přenosy typu čtení/zápis obsahují adresy spojené s daty, a tak obsahují řízení pro oba tyto druhy přenosů.

Různé sběrnice a různé typy protokolů mají rozdílné způsoby, jak zacházet s přenosy těchto čtyř druhů. Například mnoho běžně používaných sběrnicí používá ve skutečnosti dvě separátní sběrnice. A to adresovou sběrnici a datovou sběrnici. Adresy jsou umístěné na adresové sběrnici a data jsou umístěna na datové sběrnici, takže data mohou téct rychleji mezi zařízeními, protože každý typ přenosu má přidělenou svojí vlastní sběrnici.

1.4.1.4 Multiplexovaná sběrnice

Alternativou k odděleným sběrnicím je *multiplex*. Adresa i data jsou na stejné sběrnici. To znamená, že na sběrnici se nejprve vystaví adresa, za níž následují data, která se mají na tuto adresu poslat. PCI používá tento způsob. Jednoduchou 32bitovou sběrnici, na které jsou adresy a data multiplexována.

Multiplexování má o něco méně účinnou šířku pásma než dvě oddělené sběrnice, protože adresové přenosy zaplní drahocennou šířku pásma, která by mohla být lépe využita sběrnicovými přenosy dat. Ale multiplexované sběrnice jsou mnohonásobně levnější než sdílené sběrnice (*shared bus*), protože je potřeba jen poloviční počet vodičů sběrnice a zařízení na sběrnici potřebuje jen poloviční počet vstupně/výstupních pinů.

1.4.1.5 PCI a MSI

Pozdější verze PCI specifikace přidávají funkce pro organizaci sběrnicových přenosů. Ty se nazývají „postranní sběrnice„ (*side-band bus*) pro přenos různých typů řídicích příkazů. Postranní sběrnice je menší typ sběrnice, který se skládá z několika linek věnovaných přenosu řídicích a konfiguračních informací. Samozřejmě, tato postranní sběrnice zvyšuje počet pinů, odběr energie, cenu, atd. A proto to není optimální řešení.

V nejnovějších verzích *PCI* specifikace jsou navrženy metody pro operace typu čtení/zápis tak, aby se ke koncovému *PCI* zařízení přenášel pouze jeden druh příkazů, kterým se říká řídicí přenosy. Tato metoda se nazývá *Message Signal Interrupt* (*MSI*), která pomocí speciální zprávy nastaví v paměťovém adresním prostoru PCI řídicí zprávu nazývanou přerušení.

A jak popíši v následujícím textu, *PCI Express* rozšiřuje specifikaci MSI ne jen o přerušení, ale o veškeré řídicí signály postranní sběrnice.

1.4.2 Historický přehled přenosových rychlostí

Tak jak byla nejdříve vytlačena sběrnice *ISA* sběrnicí typu *PCI*, tak bude též v krátkém časovém horizontu vytlačena sběrnice *PCI* novým standardem sběrnice *PCI Express*.

Zde uvádím tabulku přenosových rychlostí, šířek sběrnic a pracovních frekvencí u sběrnic *PCI* a jejích předchůdců v Tab. 1-1.

Tab. 1-1Přehled PCI a jejích předchůdců

Bus Type	Bus Width	Bus Speed	MB/sec
ISA	16 bits	8 MHz	16 MBps
EISA	32 bits	8 MHz	32 MBps
VL-bus	32 bits	25 MHz	100 MBps
VL-bus	32 bits	33 MHz	132 MBps
PCI	32 bits	33 MHz	132 MBps
PCI	64 bits	33 MHz	264 MBps
PCI	64 bits	66 MHz	512 MBps
PCI	64 bits	133 MHz	1 GBps

1.4.3 Přehled nedostatků PCI

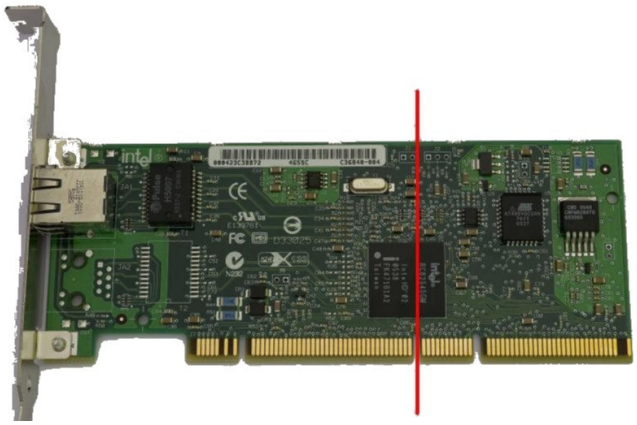
Sběrnice PCI,tak jak existuje a tak jak ji dnes známe má vážné nedostatky, které zabraňují rozšiřování šířky přenosového pásma a které nedovolí poskytnout nové rysy potřebné pro dnešní i pro budoucí generace vstupně/výstupních zařízení a pro vysokorychlostní velkokapacitní paměťová zařízení nové generace. Výslovně, její vysoce paralelní sdílená sběrnicová architektura ji ponechává v zadu a limituje její maximální rychlost a rozšiřitelnost. Její jednoduchý *load-store* komunikační model je méně robustní a schopný rozšíření než nový model založený na sériových paketových přenosech.

1.4.4 PCI-X

1.4.4.1 PCI-X: širší, rychlejší, přesto však zastaralý

PCI-X specifikace byl pokus o aktualizaci PCI tak bezbolestně, jak jen to bylo možné. Byla to snaha jak zachovat tento standard ještě o několik málo let déle. Ovšem toto byl ukázkový příklad, kdy nový druh specifikace na starém standardu nezmění nic k lepšímu. Ba naopak, ve skutečnosti se některé problémy pronásledující sběrnici PCI staly na sběrnici PCI-X ještě mnohem horšími.

Na Obr. 1-6 PCI-X síťová karta si nelze nepovšimnout podobnosti s klasickou kartou do *PCI*. Kdyby jsem ji odstřihl v místě, které jsem vyznačil červenou čarou, tak zbude jen obyčejná síťová karta do *PCI*.



Obr. 1-6 PCI-X síťová karta

1.4.4.2 Specifikace PCI-X

PCI-X specifikace v podstatě zdvojnásobila šířku sběrnice z 32 bitů na 64 bitů, tím zvýšila paralelní schopnosti přenosu dat PCI a rozšířila jeho adresní prostor. Specifikace také zvýšila základní taktovací kmitočet PCI ze 66 MHz na 133 MHz a vytvořil tak vysokoúrovňovou variantu, která poskytovala ještě větší šíři přenosového pásma, a to až na hranici 1 GB/s (ve 133 MHz verzi).

Poslední verze PCI-X specifikace (PCI-X 266) se nechala inspirovat technologií známou z klasických počítačových pamětí. Využívá tzv. *Dual technology*. To znamená, že jsou data přenášena jak na náběžnou hranu hodin, tak i na spádovou hranu hodin. Zatímco teoreticky tato technologie přiblížila přenosovou šířku pásma vrcholu, tak ve skutečnosti byly trvalé zisky v šířce přenosového pásma mnohem více skromné.

1.4.4.2.1 Jednotlivá vylepšení

Zatímco oba tyto kroky zvýšily přenosovou šířku pásma *PCI* a jeho užitečnost, tak ale také mnohem více zdražily realizaci těchto nových karet. Když běží sběrnice na vyšších frekvencích, tak se stává mnohem více citlivá na šum a na přeslechy. Výrobní standardy na vysokorychlostní sběrnice jsou neúměrně přísné, požaduje se použití pouze těch nejkvalitnějších materiálů a jakékoliv výrobní chyby se přímo přenášejí do šumu na zařízeních. To znamená, že výrobní cena vysokorychlostních sběrnic a karet do *PCI-X* je několikanásobně vyšší než u sběrnic a rozšiřujících karet do *PCI*. Pokud by tato technologie byla nasazená pouze v několika případech (např. ovládací panely raketoplánů), tak by výrobní náklady a požadavky na přesnost výroby neměly takový význam. Ale

pokud má být tento standard nasazen v masovém počítačovém průmyslu, tak je již před příchodem na trh odsouzen k rychlému zániku.

1.4.4.2.2 Zjištěné problémy

Vyšší taktovací kmitočet počítače není jediné, co zvyšuje problémy s šumem a výrobní náklady u *PCI-X*. Jako další faktor se projevu zvětšená šířka sběrnice. Protože je sběrnice širší a sestává z více vodičů, vznikají mnohem častěji a mnohem více přeslechy na vedení. Navíc všechna tato nová vedení mnohem více zatěžují sběrnici a injektují do ní mnohem více šumu z připojených zařízení. A nakonec je to problém, že nový standard obsahuje 32pinů navíc oproti sběrnici *PCI*, což zvyšuje výrobní náklady každého samostatného zařízení a cenu konektorů na základních deskách.

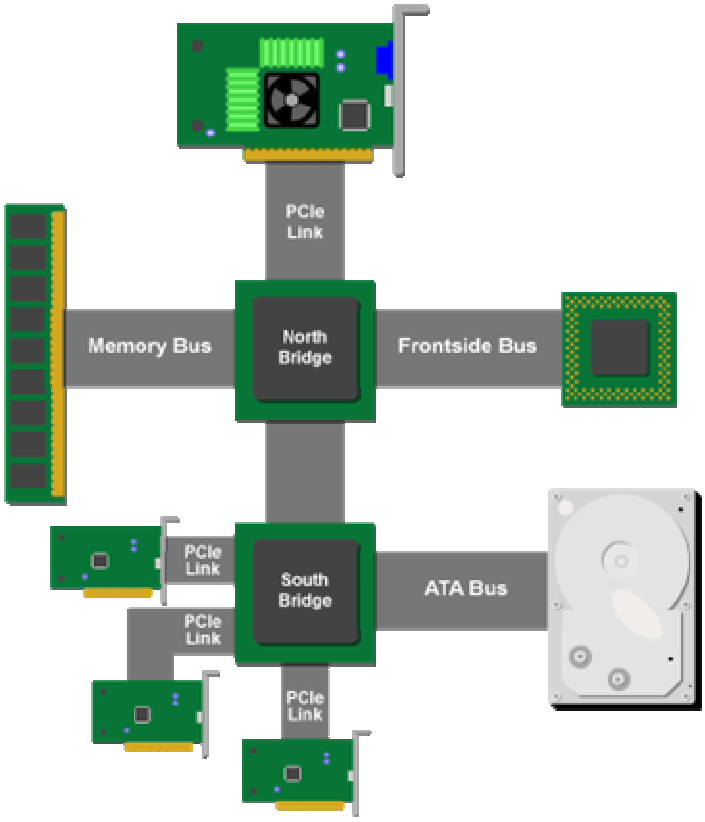
1.4.4.2.3 Neúspěch PCI-X

Když vezmu všechny tyto faktory dohromady a spojím je s vyšší taktovací frekvencí, tak to z *PCI-X* udělá oproti *PCI* velice drahou záležitost. A cena byla to, co drželo posledních několik let sběrnici *PCI* na základních deskách. A musím také podotknout, že všechny tyto problémy se stoupajícím paralelismem a mechanismy reakce jak na náběžnou, tak na spádovou hranu hodin, jsou značně omezující faktory pamětí *DDR*, a obzvláště pamětí a specifikace *DDR-II*.

A přes všechny tyto neduhy se musí stále pracovat s PCI sdílenou topologií sběrnice, která má v sobě skryté také mnohé nectnosti (např. omezený počet připojených zařízení).

2 PCI Express

Topologie čipu s *PCIe* je zobrazena na Obr. 2-1.



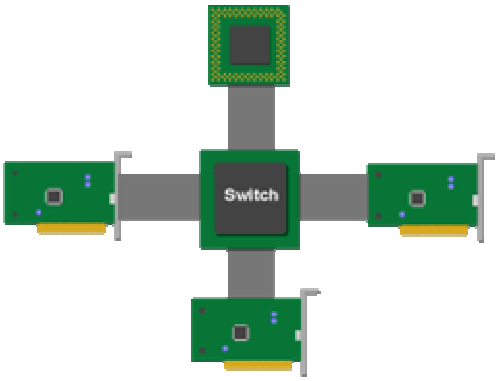
Obr. 2-1 Čipset PCIe ♣

PCI Express (PCIe) je nejnovější jméno pro technologie dříve známé jako 3GIO. Ačkoliv byla PCIe specifikace dokončená již v roce 2002, zařízení založená na této sběrnici se dostávají na trh až v dnešních dnech.

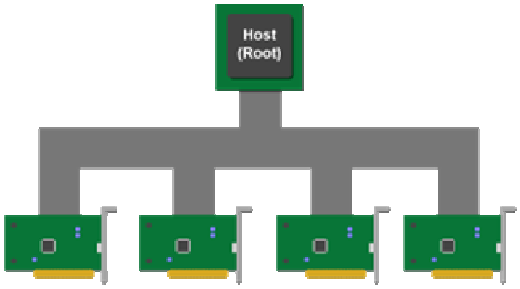
Asi nejvíce drastické a zřejmě nejlepší zlepšení *PCI Express* oproti *PCI* je jeho dvoubodová technologie (*point-to-point*).

Ve dvoubodové sběrnicové topologii nahrazuje sdílený přepínač sdílenou sběrnici, skrze který všechna zařízení komunikují. Nefunguje to jako v topologii sdílené sběrnice, kde se musí všechna zařízení domluvit o používání sběrnice. Každé zařízení má unikátní přístup k přepínači. Rozdíly jsou dobře vidět na obrázcích Obr. 2-2 a Obr. 2-3.

♣ zdrojem obrázku je literatura [5]



Obr. 2-2 Sdílený přepínač ♣



Obr. 2-3 Sdílená sběrnice

Jinými slovy leží každé zařízení na své vlastní sběrnici, která je pojmenována jako spojení(*link*).

2.1 Popis činnosti

Switch zde funguje stejně jako přepínač v síťových nebo telefonních komunikacích. Přepíná přenosy na sběrnici a stanovuje dvoubodové spoje mezi nějakými dvěma komunikačními zařízeními na systému.

Ve dvoubodovém schématu na Obr. 2-2 může procesor skrze přepínač komunikovat s jakýmkoliv z *PCIe* připojených zařízení pomocí jejich adresy. Procesor si otevře soukromé privilegované spojení. Samozřejmě spojení funguje stejně jako u moderního telefonního přístroje, nebo u internetové komunikace prohlížeče se serverem. Dvě komunikující strany si jen myslí, že spolu mluví navzájem přes privátní přímé nepřetržité spojení. Ve skutečnosti je komunikační proud rozdělen do několika diskrétních balíčků dat, které *switch* přepojuje.

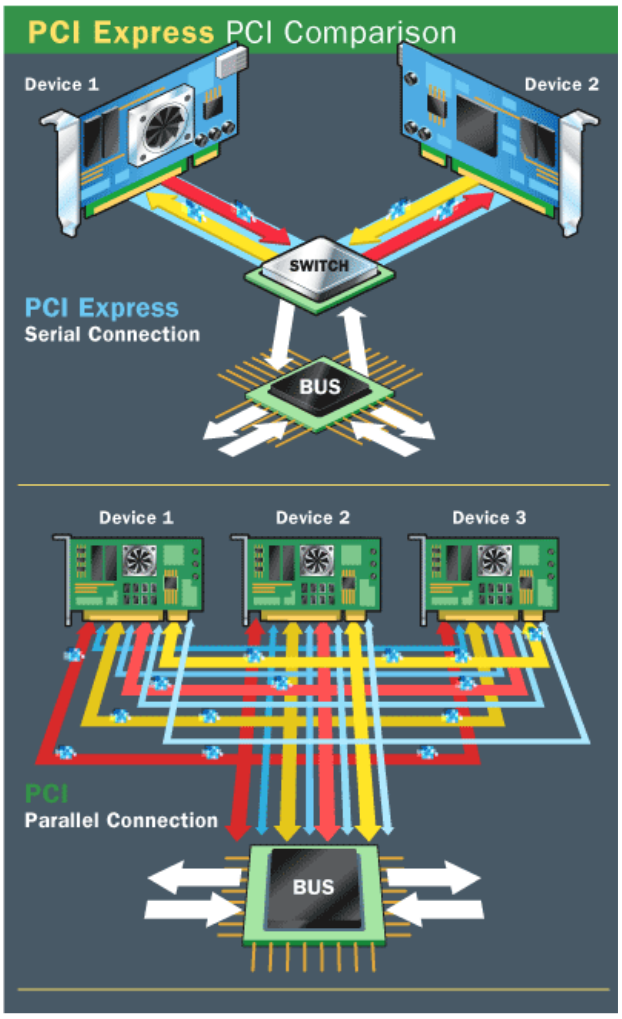
2.2 Dodržení kvality služeb (Quality of Service)

Celkový efekt přepínané topologie je v tom, že potřebuje jediné zařízení k řízení a směrování přenosu a to je integrováno v jednom jediném čipu, v přepínači (*switch*). Se sdílenou sběrnicí musí každé jednotlivé zařízení užívat arbitrážní schéma k tomu, aby rozhodlo, jak mezi sebou distribuovat sdílený zdroj – sběrnici. S přepínanou strukturou dělá ohledně sdílených prostředků všechna rozhodnutí *switch.*

♣ zdrojem obrázku je literatura [5]

Centralizované směrování provozu a řízení zdrojů je jednou z věcí, kterou PCIe umožňuje. Ale nabízí také služby využitelné převážně pro příští generace komunikace – požadovaná kvalita služeb (*QoS*). PCI Express *switch* může jednotlivým paketům přiřadit různou prioritu, takže například může dát vysokou prioritu paketům pracujícím v reálném čase, např. video či audio *stream*. Ty pak budou mít na přepínači přednost před pakety, které nejsou časově tolik kritické. To by mělo napomoci menšímu trhání obrazu ve hrách nebo menšímu časovému zpoždění softwaru pro digitální záznam zvuku.

2.3 Zpětná kompatibilita



Obr. 2-4 PCIe *vs.* PCI ♣

Nový typ sběrnice byl navržen tak, aby byl zpětně kompatibilní s *PCI*. Operační systém osobního počítače může startovat na systému s *PCI Express* bez modifikace.

Na Obr. 2-4 je vidět rozdíl v komunikačních tocích jednotlivých standardů.

Z obrázku je jasně vidět, proč systém u PCI při zvyšující se datové propustnosti tolik trápí šumy a přeslechy.

PCI a PCI Express byly navrženy tak, aby přenášela data podle normalizovaného systému OSI.

PCI realizuje první čtyři vrstvy OSI, které specifikují fyzické aspekty přenosu skrze vyšší nadřazené rozhraní které využívá PCI k vysílání a příjmu dat. Návrháři PCI Express

♣ zdrojem obrázku je literatura [4]

nechaly tuto část *load-store* paměťového modelu nezměněn. Takže koncové aplikace mohou i v PCIe stále číst nebo zapisovat do nějakého specifického adresního prostoru. Další vrstvy berou požadavky na čtení nebo zápis, rozdělují je do paketů, přidávají ochranné kódy jako např. CRC, umísťují je do rámců a posílají na určené adresy.

Takže si aplikace stále myslí že čte nebo zapisuje do paměťového místa PCI zařízení, ale ve skutečnosti je v pozadí úplně jiný typ komunikace s odlišnými protokoly po jiném rozhraní.

2.4 Řídicí a kontrolní signály

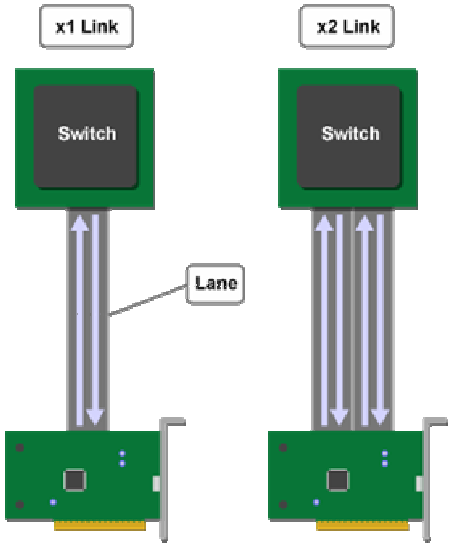
To vede zpět k tématu řídicích a kontrolních signálů. PCIe přebírá všechny postraní signály PCI a převádí je na MSI signály, které jsou typu čtení/zápis. Ty jsou poté zapouzdřené do jednotlivých paketů a přenášené jako jakékoliv jiné přenosy typu *load-store*. Z toho plyne že veškeré přenosy na rozhraní PCI Express ,ať už řídicí příkazy, příkazy typu čtení/zápis, adresy a data , jsou přenášena po jedné sériové sběrnici.

Pod PCIe jsou logicky oddělené dva druhy komunikačních přenosů, ale na fyzické úrovni jsou pouze na jedné sběrnici. První dva druhy přenosu, adresa a data, jsou kombinované ve formě paketu. Jádro paketu sestává z adresy v kombinaci s daty, takže struktura paketu spojí tyto dva typy.

Takže pakety se tedy v normě vyskytují ve dvou typech. Pakety řídicí a pakety typu čtení/zápis. Ovšem v literatuře se v poslední době označují jako řídicí pakety (*command packets*) a datové pakety(*data packets*).

2.5 Přenos v proudech(lanes)

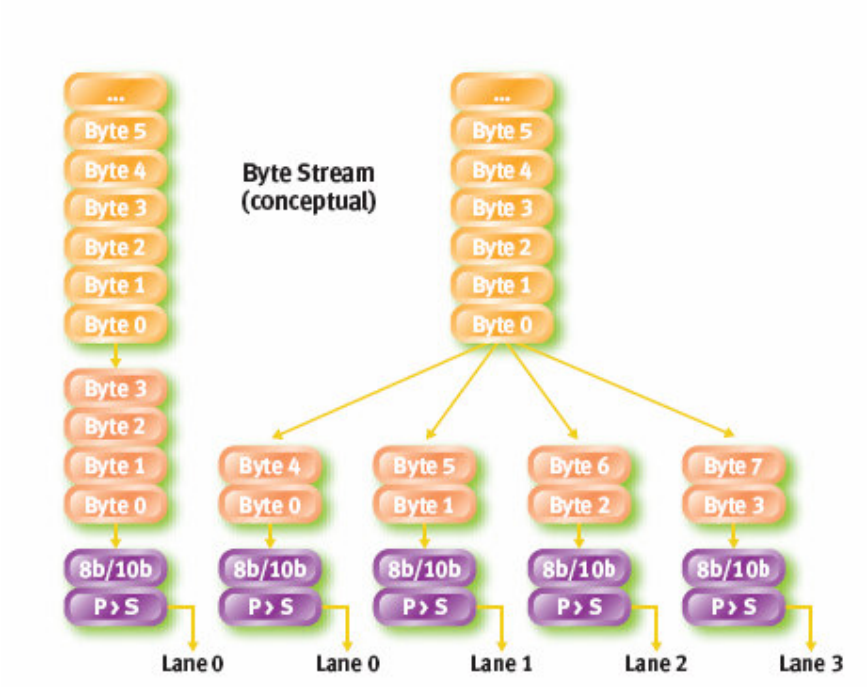
Když se návrháři systému PCIe zamýšlely co změnit ve stávajícím systému PCI pro příští generace, jedním z hlavních problémů byl počet pinů na sběrnici. Jak jsem naznačil v předcházejících kapitolách, stávající paralelismus u sběrnic PCI přinášel řadu problémů (šum, cena, rostoucí frekvence, šířka přenosového pásma, atd.). Vývojáři vyřešili tyto problémy tím, že navrhly PCIe jako sériovou sběrnici.



Obr. 2-5 Proud a linky ♣

Spojení mezi dvěma PCIe zařízeními a mezi přepínačem se nazývá linka (*link*). Každá linka se skládá z jednoho nebo z více proudů (*lanes*) a každá linka je schopná v jednom okamžiku přenést jeden byte oběma směry. Tato plně duplexní (*full-duplex*) komunikace je možná, protože každá linka je složena z jednoho páru signálu – vysílací a přijímací(*send and receive*).

Na Obr. 2-6 je znázorněno, jak se data dělí při použití jednoho nebo více proudů. Je v něm i naznačeno překódování 8 na 10 bitů kvůli symetrickému zatížení linky na fyzické vrstvě.

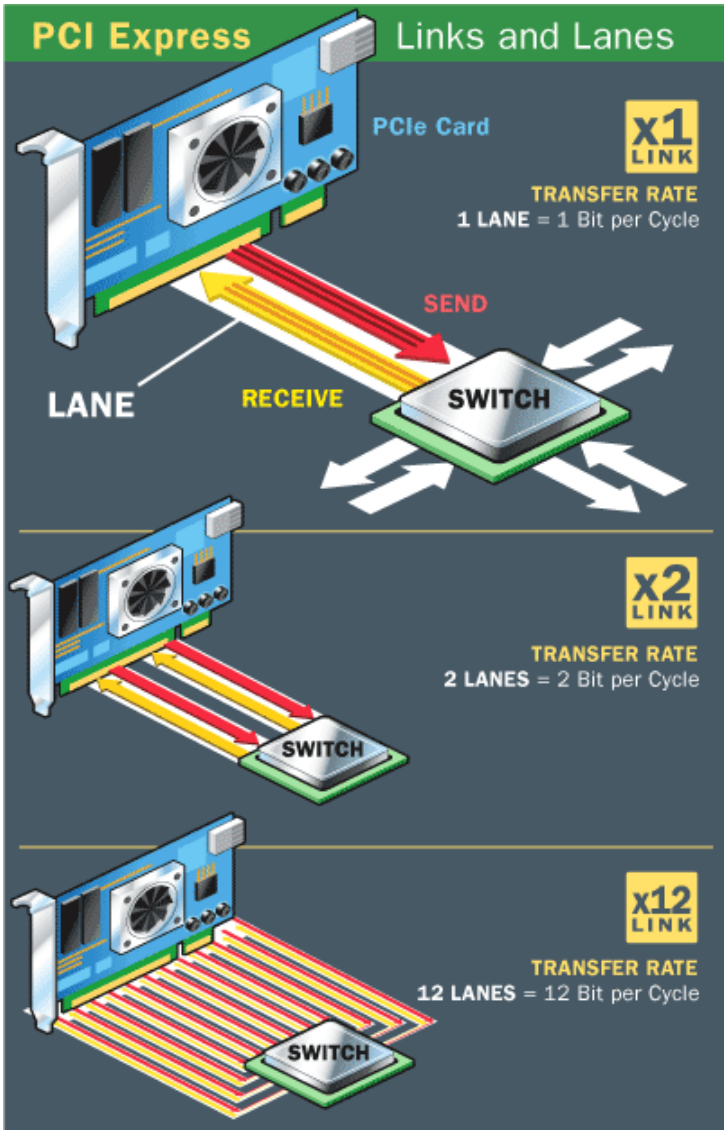


Obr. 2-6 Linka z jednoho a více proudů ♥

♣ zdrojem obrázku je literatura [5]
♥ zdrojem obrázku je literatura [6]

2.6 Přenos paketů

K tomu, aby PCIe přenášel jednotlivé pakety, které jsou složeny z několika bajtů, musí každá linka rozkouskovat pakety na jednotlivé bity a ty potom přenášet po lince značnou rychlostí.



Obr. 2-7 Vícenásobné linky ♣

Zařízení na přijímací straně musí zkompletovat všechny bajty a poté z nich opět vytvořit celý paket. Toto rozebrání a opětovné zkompletování paketů musí proběhnout velice rychle, aby byl přenos transparentní i pro ostatní vrstvy. To znamená že na každé straně spoje musí být dostatečný výkon pro tuto operaci. Protože je každá linka jen jeden bit široká, není pro přenos potřeba velký počet pinů na koncových zařízeních. Toto je způsob,jakým PCI express zvýšila šířku pásma. Došlo zde v podstatě k využití Moorova zákona,kdy se paralelní přenos nahradil značným výpočetním výkonem na obou stranách linky sériového přenosu.

Na Obr. 2-7 je dobře vidět, jak si lze představit komunikaci více než jedné linky.

♣ zdrojem obrázku je literatura [4]

Každý proud se může skládat z jedné nebo z několika linek. Jedním z nejhezčích rysů specifikace PCIe je fakt, že může propojovat jednotlivé linky dohromady a tvořit tak jedno spojení. Neboli jinak, pokud se ke spojení použije ne jedna ale hned dvě linky, můžou se pomocí proudu (*lane*) posílat dva bajty najednou, a tudíž se zdvojnásobila šířka přenosového pásma.

2.7 Spojení (lane)

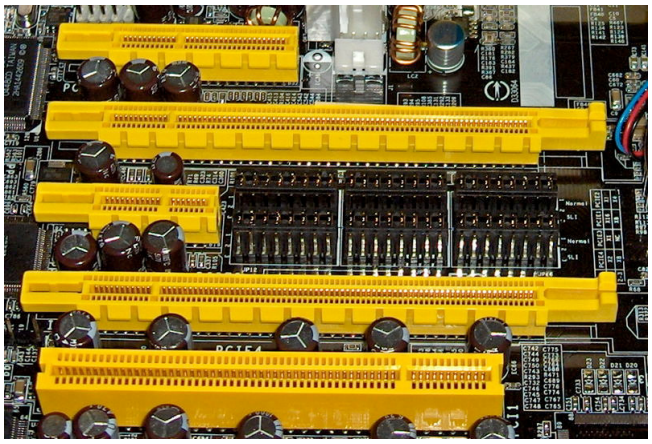
Spojení, které je složeno z jediné linky se nazývá x1 *link*, spojení které je složeno ze dvou linek se nazývá x2 *link*, spojení které je složeno ze čtyř linek se nazývá x4 *link*, atd. Norma PCI Express 1.1 podporuje spojení x1, x2, x4, x8, x12, x16 a x32.

Tab. 2-1 Maximální přenosové rychlosti PCI Express

Šířka spoje	Frekvence	Propustnost (Duplex, po bitech)	Propustnost (Duplex, po Bytech)	Očekávané použití
x1	2,5 Ghz	5 Gb/s	400 MB/s	Sloty, Giga *Ethernet*
x2	2,5 Ghz	10 Gb/s	800 MB/s	
x4	2,5 Ghz	20 Gb/s	1,6 GB/s	Sloty, 10G *Ethernet*, SCSI, SAS
x8	2,5 Ghz	40 Gb/s	3,2 GB/s	
x16	2,5 Ghz	80 Gb/s	6,4 GB/s	Grafické karty

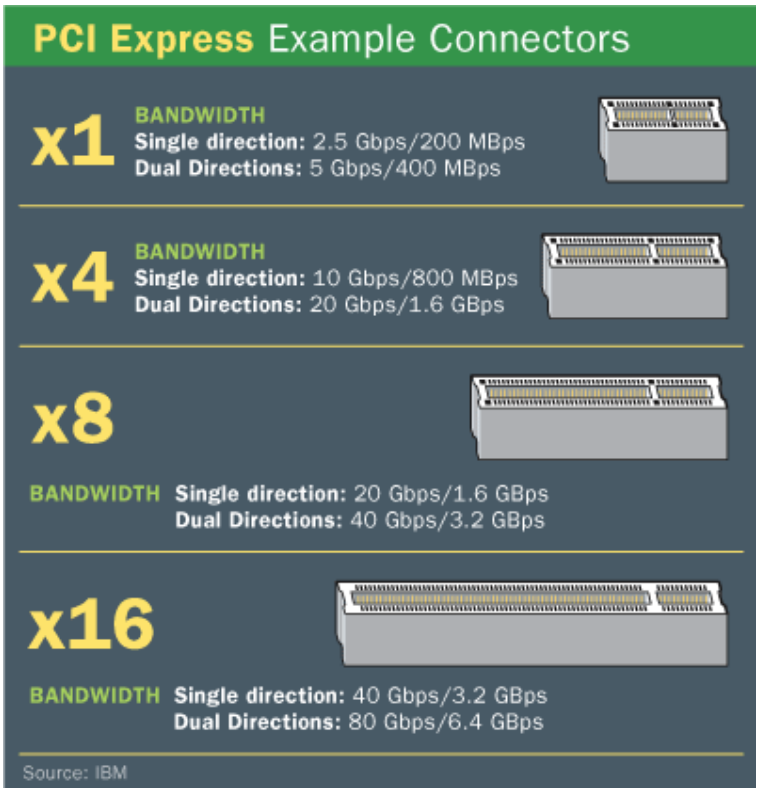
Zisky šířky pásma PCIe oproti PCI jsou značné. Každá linka je schopna přenášet současně v obou směrech 2,5 Gb/s. Pokud se použijí dvě linky zároveň, tak šířka přenosového pásma bude 5 Gb/s. Každou přidanou linkou ve spojení roste šířka přenosového pásma.

2.8 Sloty a rychlosti



Obr. 2-8 Sloty PCI Express

Na Obr. 2-8 je motherboard osazený čipsetem Nforce 4 ultra D na kterém jsou vyobrazeny jednotlivé patice PCI Express a to od shora v tomto pořadí x4, x16, x1, x16 a zcela vespod je klasický slot pro 32 bitovou kartu stávajícího systému s PCI. Ještě stojí za povšimnutí, že sloty x16 jsou na konci vybaveny zámkem a jsou primárně určeny pro připojení grafických karet, tak jak je to zvykem u slotu AGP.

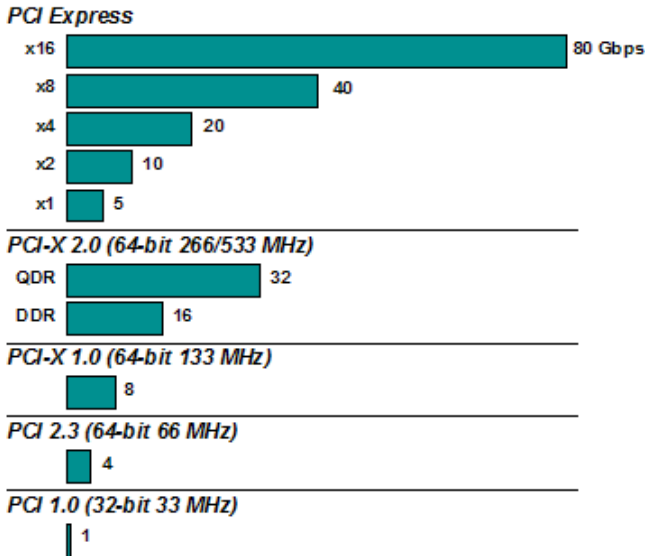


Na Obr. 2-9 , který uveřejnila firma IBM, jsou nakresleny jednotlivé sloty s jejich datovou propustností. Nutno podotknout, že uvedená čísla opravdu odpovídají realitě, a ne jako u sběrnic PCI-X, kde byla teoretická čísla datových propustní značně vyšší, než jaká byla poté ve skutečnosti naměřena s reálnými aplikacemi v provozu.

Obr. 2-9 PCIe Sloty ♣

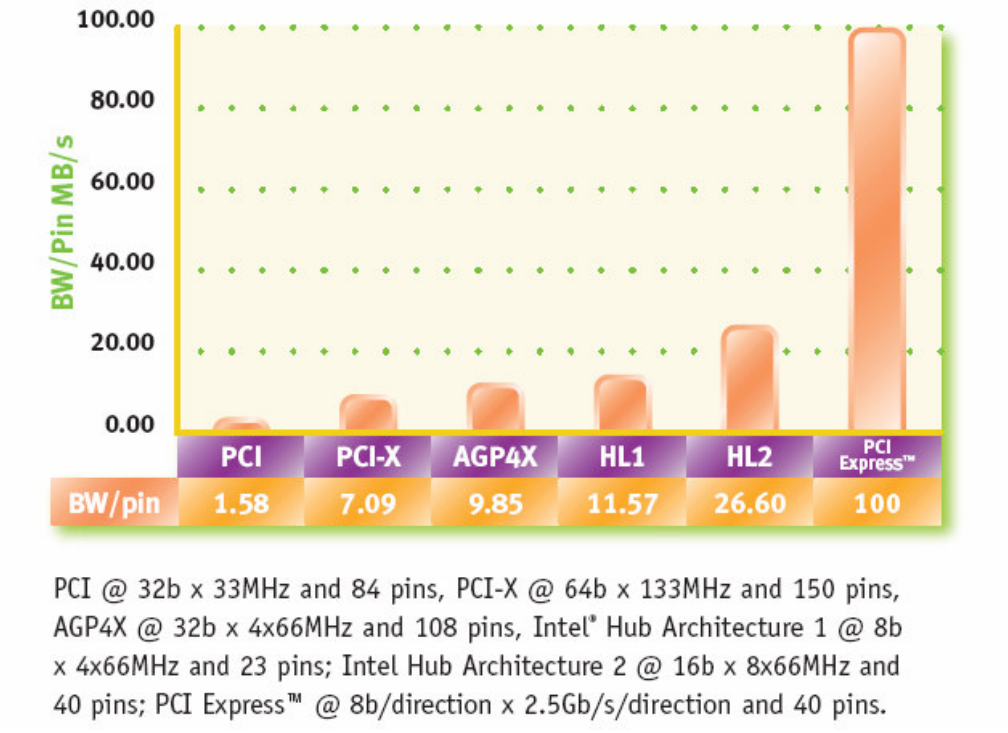
Na dalším obrázku Obr. 2-10 PCIe vs. PCI přenosová rychlost je dobře vidět nárůst přenosové šířky pásma PCIe oproti sběrnici PCI. Pro představu jsou zde uvedeny i rychlosti nepovedeného směru PCI-X.

♣ zdrojem obrázku je literatura [4]



Obr. 2-10 PCIe vs. PCI přenosová rychlost

A na závěr uvádím poslední graf, který porovnává jednotlivé přenosové rychlosti architektur mezi sebou. Na Obr. 2-11 je jasně vidět nárůst šířky přenosového pásma v poměru k počtu použitých pinů.

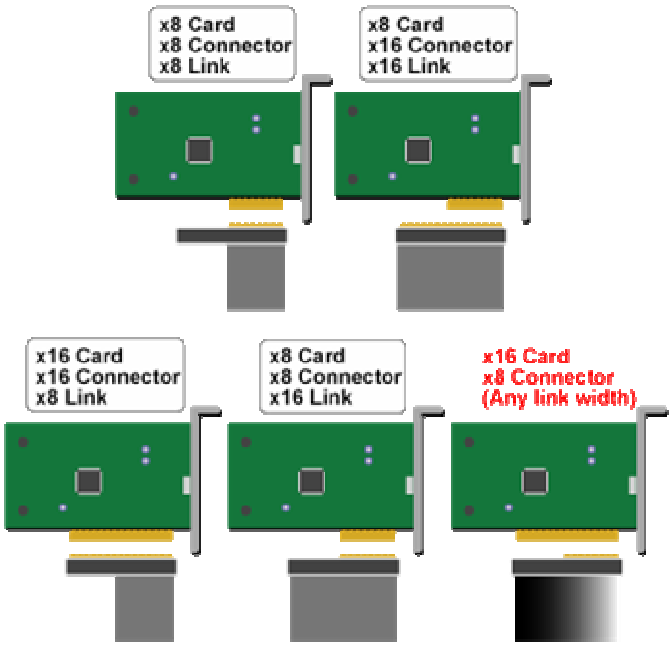


Obr. 2-11 Šířka přenosového pásma vzhledem k počtu použitých pinů ♣

♣ zdrojem obrázku je literatura [6]

2.9 Určování počtu linek při startu

Při startu PC musí každé zařízení připojené ke sběrnici PCIe nejdříve vyjednat maximální počet linek, ze kterého bude spojení uskutečněno. Tato maximální šířka spojení vychází z maximální počtu linek, které se buď na kartě nebo na slotu sběrnice nacházejí. Různé možnosti kombinací jsou znázorněny na Obr. 2-12.



Obr. 2-12 Vyjednávání počtu linek ♣

Šířka spoje samozřejmě závisí i na šíři přepojovacího rozhraní, ale já zde vycházím z předpokladu,že šířka přepojovacího rozhraní je stejná jako fyzická šířka spojení.

Každé zařízení certifikované pro sběrnici PCIe má již na kartě zabudovaný určitý počet linek. Plný počet linky x16 mají doposud jen karty *Nvidia* SLI. To znamená, že jsou to jediné karty na trhu které mají plnou šířku sběrnice, tak jak byla ratifikována PCI Express normou 1.1.

Jestliže nebude mít slot na sběrnici plnou šíři pro podporu 16 linek, tak nebudou všechny signály přicházející z karty správně interpretovány a zbylé, na které se nedostanou piny budou prostě ignorovány. Zařízení by tudíž nemělo pracovat správně, nebo spíše vůbec.

Toto je jediný případ kdy komunikace selže (nedostatek měděných pinů na slotu). Při startu systému se přepínač jednotlivých zařízení dotáže, v kolika linkách jsou schopny komunikovat. Takže pokud se vyskytne karta s menším počtem pinů v širší sběrnici (některé piny ve slotu nezapojeny, budou se muset ignorovat), tak přepínač i karta určí

♣ zdrojem obrázku je literatura [5]

nejužší hrdlo komunikace, a pro tuto šíři nastaví počet linek a tudíž i rychlost komunikace. To znamená, že systém je navržen tak, aby sloty v plné šíři mohly podporovat karty v plné šíři, ale i karty kratší. Na Obr. 2-12 je červeně vyznačen jediný případ zapojení, kdy nebude karta ve slotu fungovat. Bude to pro to, že se signály z karty x16 se budou ve slotu x8 prostě ztrácet. Šedivý pruh pod každým slotem vyjadřuje šířku toku, kterou je schopen přenášet přepínač. To znamená, že v systémech, kde z nějakého důvodu nebude možné realizovat spoje v maximální šíři stačí zabudovat sloty v plné šíři, a případ nefunkční karty nebude moci nastat (ovšem za cenu snížení rychlosti, ve které je karta schopna komunikovat, ale to je pořád dobrá výměna - snížení počtu linek namísto úplné nefunkčnosti karty).

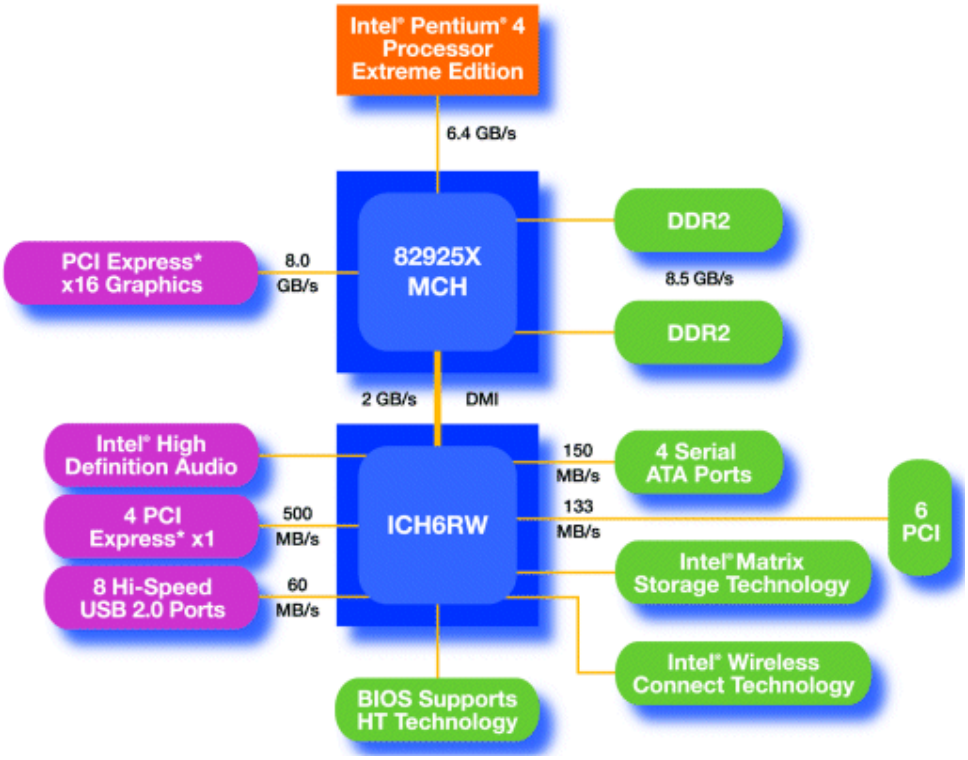
2.10 Přemostění PCI do PCIe

Most PCI to PCIe překládá pakety jdoucí po PCI Express sběrnici zpět do sběrnicových signálů, tak jak je známe u PCI. Tyto přemostění dovolují používat staré karty PCI zapojené do nového systému vybaveného PCIe. Přemostění může být provedeno jak na základní desce, tak i na kartě. NVIDIA využívá tohoto postupu u svých nových grafických karet, které tvoří první generaci karet pro sběrnici PCI Express. NVIDIA má *bridge* zabudovaný přímo v grafické kartě, takže grafická karta je vlastně karta do AGP, která lze zasunout do slotu PCIe. Společnost ATI podporuje přímo nativní PCIe rozhraní, a tudíž mostní čipy nepotřebuje.

2.11 PCI Express v praxi

Výborný příklad přemostění PCIe do PCI je na základních deskách od společnosti Intel. Jejich série s označením 900 jsou toho důkazem. Na Obr. 2-13 jsou vidět jednotlivá připojení PCIe rozhraní na severním i jižním můstku.

Z obrázku je dobře vidět, že sběrnice se PCIe pomalu stává standardem v osobních počítačích.



Obr. 2-13 Blokové schéma čipsetu 925x od společnosti Intel ♣

2.12 Softwarová kompatibilita PCIe a PCI

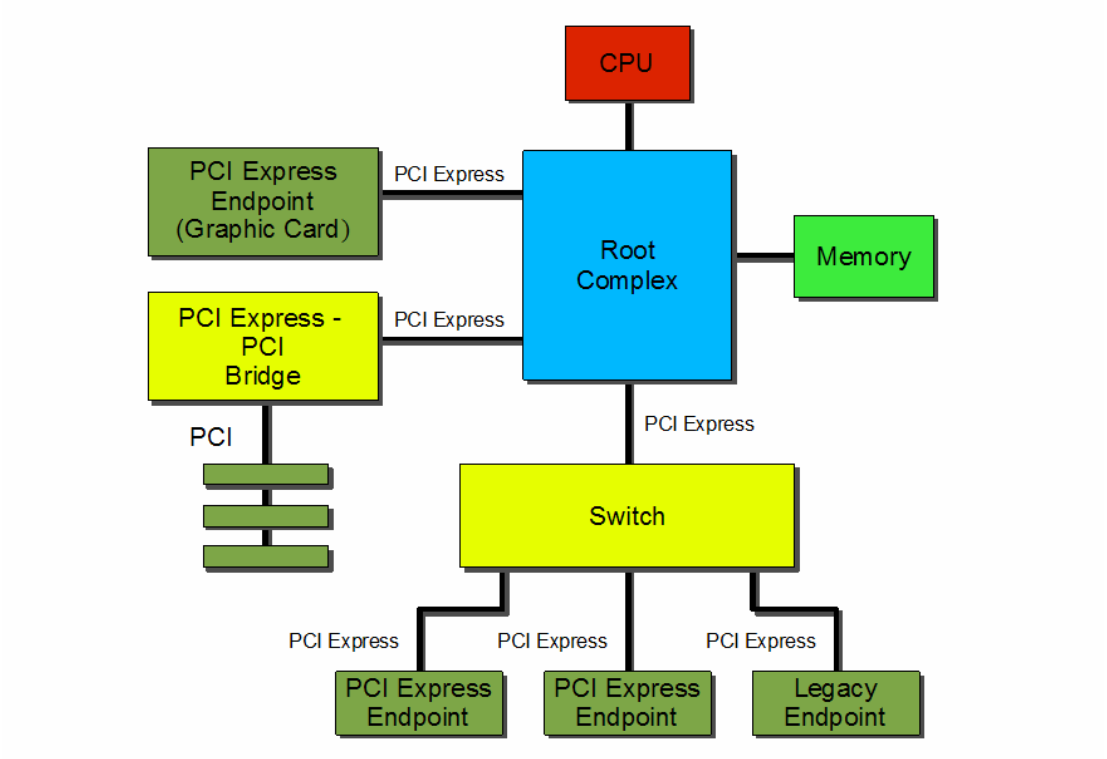
Na úrovni aplikační vrstvy je PCI Express kompatibilní se systémem PCI. Proto je možné veškerá zařízení připojená ke sběrnici PCIe konfigurovat pomocí softwaru původně psaného pro konfiguraci zařízení na PCI. PCI a PCIe se neliší pouze na fyzické úrovni, ale shodují se podle standardu až na softwarové úrovni. Tuto kompatibilitu zajišťuje topologie, která je u PCIe zavedena.

2.13 Topologie PCIe

Rozhraní PCI Express je spojení typu *point to point*. Je tedy určeno k přímému spojení dvou zařízení. Hierarchie u propojení více PCIe zařízení je podobná struktuře rozhraní USB, které využívá stromovou hierarchii a pro rozvětvení linky je využíván přepínač.

Příklad struktury systému založeném na sběrnici PCI Express je na Obr. 2-14. Výchozím bodem v tomto obrázku je zařízení označené jako *Root Complex*, které tvoří prostředníka mezi sběrnicí procesoru, pamětí a sběrnicí PCI Express.

♣ zdrojem obrázku je literatura [6]



Obr. 2-14 Topologie PCI Express ♣

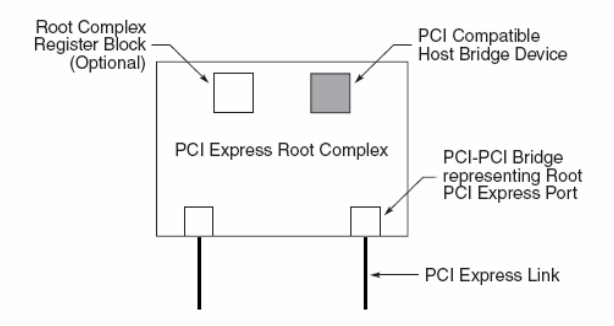
Endpoint , neboli koncová zařízení jsou ve většině případech připojena pomocí přepínače. To ovšem neplatí pro port určení k připojení grafické karty, který vyžaduje mnohem větší datové toky, a tudíž je připojen přímo k *Root Complex*. Přepínače slouží k rozšíření počtu linek a také se mohou dále řetězit (stejně jak to známe u přepínačů USB). Do zvláštní kategorie spadá zařízení označované jako *PCI Express – PCI Bridge*. Toto zařízení slouží jako rozhraní mezi sběrnicemi *PCI* a *PCI Express*, a dovoluje v nových systémech používat karty určené nebo vyrobené pouze pro sběrnice PCI.

2.13.1 Kategorie zařízení PCIe

Z pohledu PCIe je *Root Complex* primární zdroj operací typu *Request* a musí tedy podporovat operace *Configuration Request* jako *Requester*. Toto zařízení může generovat vstupně výstupní operace a operace *Lock* jako *Requester*. *Root Complex* ovšem nesmí podporovat operace typu *Lock* jako *completer*, protože by při jeho uzamčení byla blokována funkce celého systému. *Root Complex* je znázorněn na Obr. 2-15.

♣ zdrojem obrázku je literatura [1]

Root Complex



Obr. 2-15 Root Complex ♣

Toto zařízení je kořenem celého systému s PCI Express. Zajišťuje přechod mezi pamětí, procesorem a rozhraním PCI Express. Jak lze usoudit z Obr. 2-13 společnosti Intel, je běžně součástí čipsetu na základní desce osobního počítače.

PCI Express Endpoint

Toto zařízení je koncovým zařízením struktury PCI Express. Může to být jakákoliv vstupně výstupní karta a nebo zařízení, které je přímo implementované v čipsetu základní desky. Nejčastější případ karty tohoto druhu je grafická karta počítače.

PCI Express *Endpoint* musí obsahovat konfigurační prostor s hlavičkou (*Configuration Space* Leader) typu 0 (zde platí podobné značení jaké je u zařízení PCI). Toto zařízení také musí podporovat operace typu *Configuration Request* jako *Completer*, a tedy odpovídat na žádosti o přístup do konfiguračního prostoru. Ovšem nesmí generovat vstupně výstupní *Request* operace a ani podporovat operace typu *Lock* jako *Completer* a ani jako *Requester*. Zařízení, která spadají do této kategorie by měla používat 64 bitové adresování pro přístup do paměťového prostoru.

Legacy Endpoint

Do této kategorie spadají zařízení, která jsou podobná zařízením PCI Express *Endpoint*, ale liší se ve svých omezeních. Důvodem pro zavedení této kategorie je zpětná kompatibilita zařízení a možnost pro využití softwaru, který byl navržen pro starší zařízení PCI.

Od zařízení PCI Express *Endpoint* se *Legacy Endpoint* liší tím, že může podporovat vstupně výstupní operace jako *Requester* i jako *Completer*. Zařízení tohoto typu mohou podporovat operace typu *Lock* pouze jako *Completer*. Zařízení *Legacy Endpoint* mohou podporovat jak 64bitové, tak i 32bitové adresování paměťového prostoru.

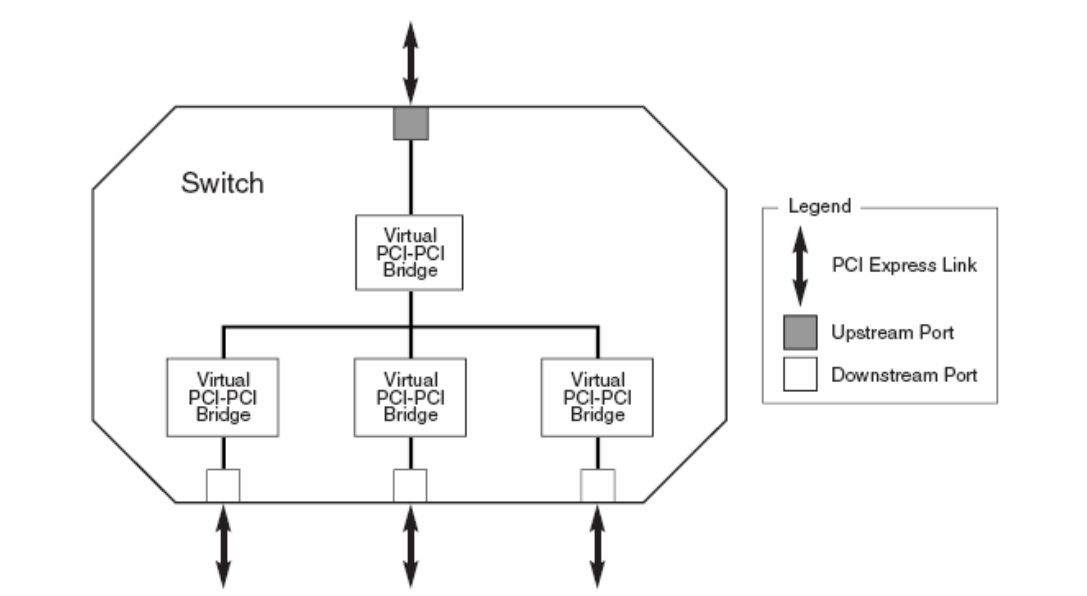
♣ zdrojem obrázku je literatura [1]

PCI Express – PCI Bridge

Zařízení, které spadají do této kategorie slouží k připojení sběrnice PCI do struktury systému založeném na PCI Express. Tato zařízení pracují jako mosty mezi rozhraním PCI a PCIe.

Switch

Zařízení tohoto typu slouží k rozšíření počtu portů PCI Express. Díky čemuž se zvedne i celkový počet koncových zařízení, které je možno k systému s PCIe připojit.

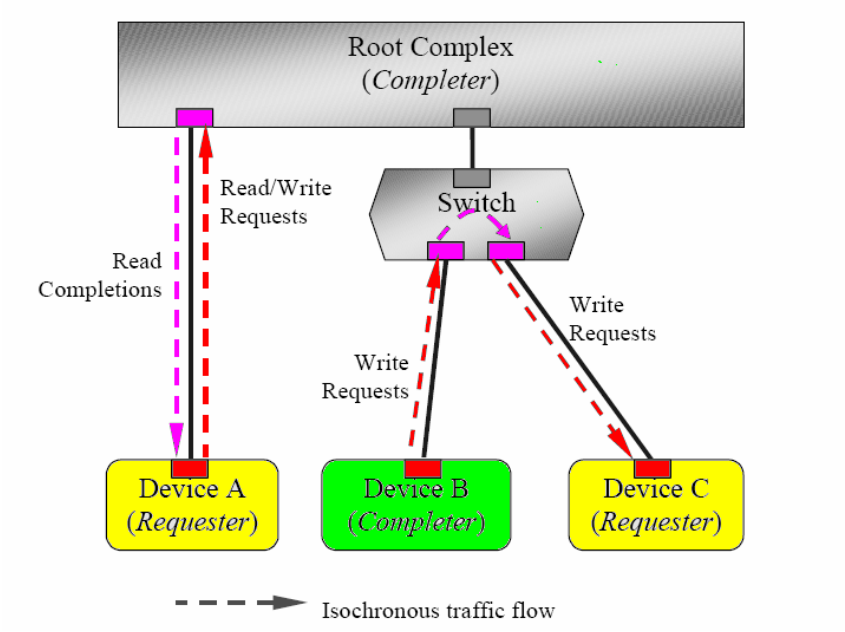


Obr. 2-16 Logické zapojení přepínače ♣

Z hlediska softwarové kompatibility se přepínač jeví jako několik virtuálních PCIe – PCI mostů. Zřízení spadající do této kategorie musí být schopna přeposílat transakce,které jsou podporovány rozhraním PCI express.

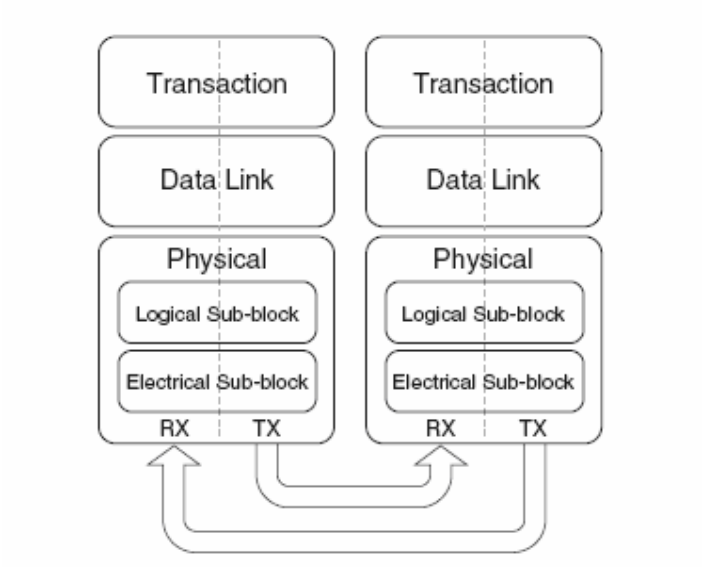
Na dalším obrázku Obr. 2-17 jsou ukázány dva typy komunikace podle specifikace PCI Express. *End - point to host* mezi zařízením A a *Root Complex (Endpoint – to – Root Complex*), a potom *Peer-to-Peer* mezi zařízeními B a C (*Endpoint - to - Endpoint*), u tohoto typu komunikaci se používají pouze transakce.

♣ zdrojem obrázku je literatura [1]



Obr. 2-17 Příklad komunikačního modelu ♣

2.14 Vrstvy PCI-Express



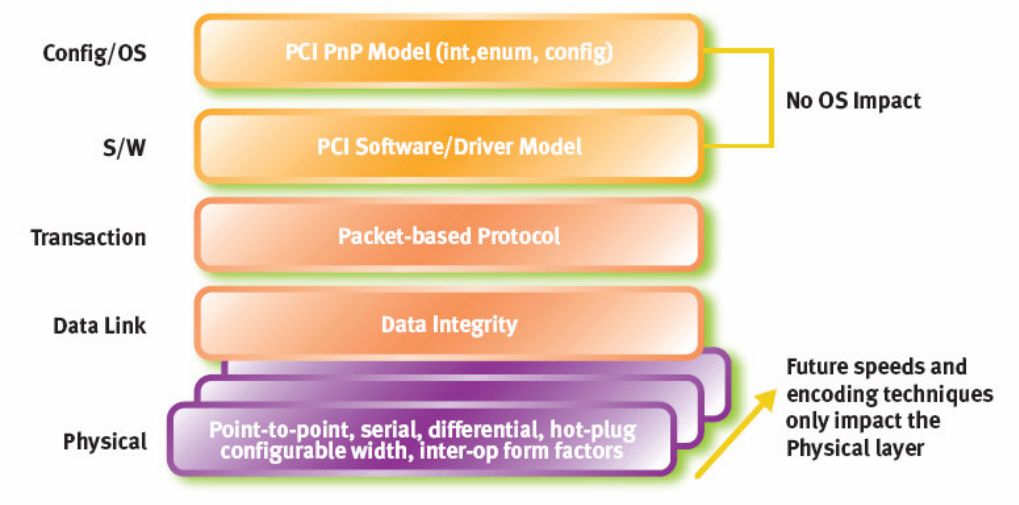
Obr. 2-18 Vrstvový model ♥

Rozhraní PCI-Express je popsáno modelem, který se skládá ze tří diskrétních vrstev. Z fyzické vrstvy (*Physical Layer*), z linkové vrstvy (*Data Link Layer*) a z transakční vrstvy (*Transaction Layer*). Každá z těchto částí se dále dělí na dvě části, jedna pro příjem a druhá pro vysílání dat. Vrstvový model je znázorněn na Obr. 2-18.

Vrstvový model je zaveden kvůli lepší přehlednosti a snazší implementaci Podobné modely se již osvědčily například v síťových komunikacích a zajišťují maximální

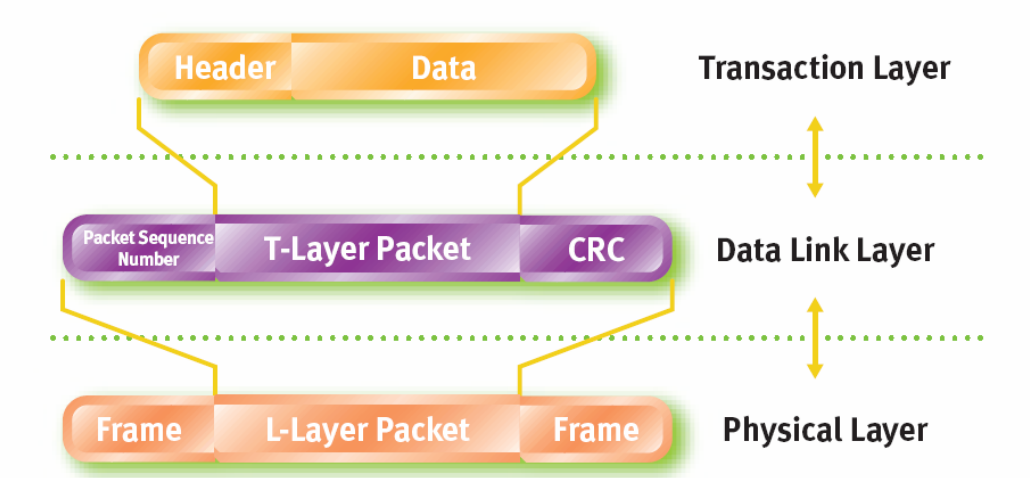
♣ zdrojem obrázku je literatura [1]
♥ zdrojem obrázku je literatura [1]

přenositelnost mezi jednotlivými platformami. Na Obr. 2-19 jsou vyznačeny jednotlivé vrstvy i s napojením na aplikační vrstvu se základním popisem funkcí, které zajišťují.



Obr. 2-19 Vrstvový model se základními funkcemi vrstev ♣

Na Obr. 2-20 je zobrazen datový paket a je na něm předvedeno, jaké druhy nadbytečných dat přidávají jednotlivé vrstvy k základním přenášeným datům.



Obr. 2-20 Vrstvový model s funkcemi prováděnými na paketu ♥

2.14.1 Fyzická vrstva

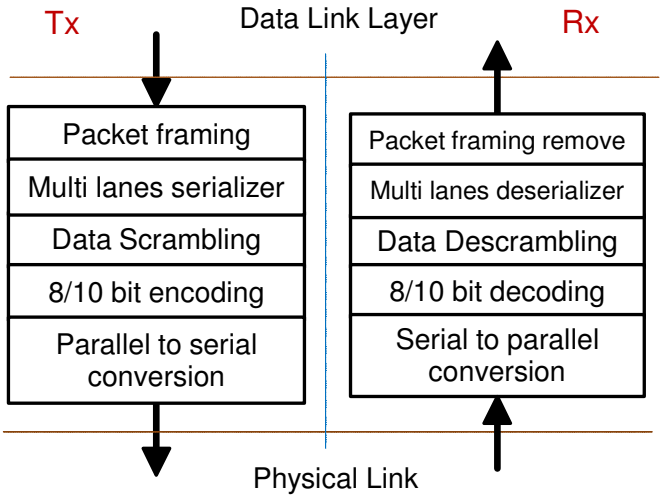
Tato vrstva je v modelu vrstvou nejspodnější. Data jsou v ní representována na nejnižší možné úrovni. Fyzická vrstva je také pro lepší popis rozdělena do dvou oddělených podbloků. Jsou to úrovně, na kterých jsou signály rozděleny podle logických

♣ zdrojem obrázku je literatura [6]
♥ zdrojem obrázku je literatura [6]

úrovní (*Logical Sub block*) a podle elektrických úrovní (*Electrical Sub block*).Elektrická část této vrstvy zodpovídá za elektrickou kompatibilitu signálů vyskytujících se na lince, kdež to logická část se stará o řízení přístupu k fyzické lince a o napojení fyzické linky na datovou strukturu. Úkolem této nejspodnější vrstvy je správa fyzické linky a zajištění přenosu dat a řídicích signálů do vyšších vrstev. Jednotlivé činnosti vrstvy by se daly popsat tímto seznamem:

- **Konverze ze sériového přenosu na přenos paralelní a opačně**
- **Kódování a dekódování z osmi na deset bitů (8/10)**
- **Funkce scramblingu a descramblingu**
- **Serializace a deserializace přenosu**
- **Aplikace framingu**
- **Konfigurace linky, stavový automat LTSSM – *Link Training and Status State Machine***
- **Vysílání a příjem paketů TLP –*Transaction Layer Packets* a DLLP-*Data Link Layer Packets* skrz fyzickou vrstvu**
- **Implementace budičů a přijímačů sběrnice**
- **Udržování hodinového signálu a bitová a bajtová synchronizace příjmu**

Zjednodušeně se dá práce s pakety na fyzické vrstvě popsat obrázkem Obr. 2-21.

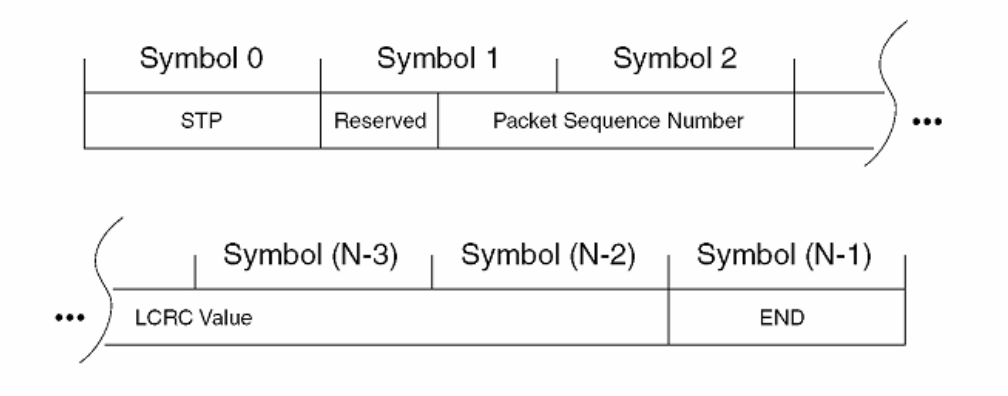


Obr. 2-21 Zpracování paketů fyzickou vrstvou ♣

♣ obrázek vytvořen v Smart draw 2007

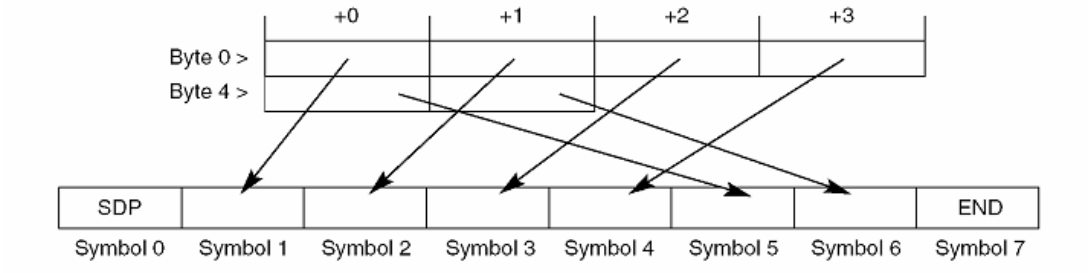
2.14.1.1 Způsoby framingu a rozdělení dat do proudů (*Lanes*)

V případě, že linka obsahuje více než jeden *Lane*, tak se pro řazení symbolů používají dva různé způsoby framingu. Do první kategorie spadají speciální posloupnosti používané fyzickou vrstvou. Do druhé kategorie patří pakety linkové (DLLP) a transakční vrstvy (TLP). Na speciální posloupnosti je *framing* aplikován tak, že posloupnost se vysílá sériově pro každou *Lane* zvlášť.



Obr. 2-22 Řazení a framing TLP symbolů na lince 1x ♣

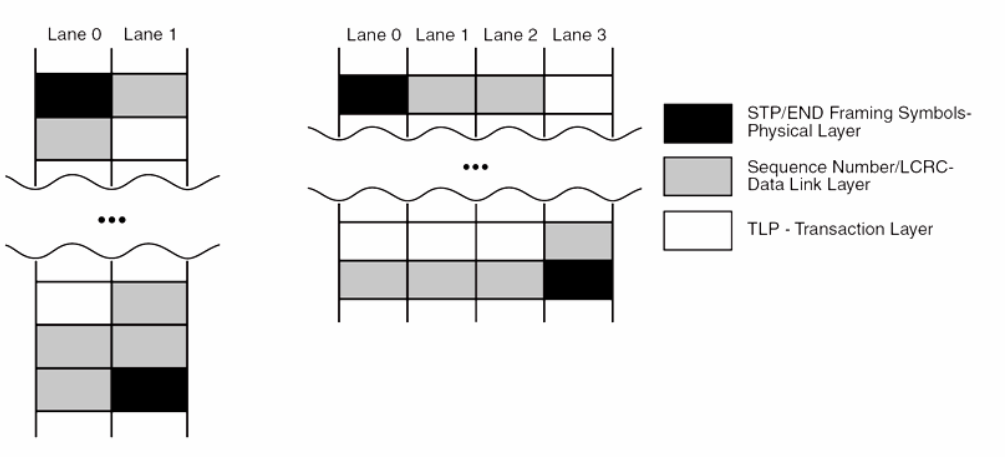
U paketů linkové a transakční vrstvy se rozdělují bajt po bajtu na jednotlivé *Lanes*. Znak, který označuje začátek paketu, je umístěn vždy na *Lane* 0. Na Obr. 2-22 a Obr. 2-23 je znázorněn způsob řazení DLLP a TLP paketů na lince 1x.



Obr. 2-23 Řazení a framing DLLP symbolů na lince 1x ♥

Na Obr. 2-24 je ukázáno řazení symbolů na lince 2x a 4x. Při rekonstrukci na přijímací straně je třeba aplikovat opačný proces k sestavení celého paketu.

♣ zdrojem obrázku je literatura [1]
♥ zdrojem obrázku je literatura [1]



Obr. 2-24 Řazení a framing symbolů na lince 2x, 4x ♣

2.14.1.2 Řídicí znaky

Před posíláním dat fyzickou vrstvou jsou data překódována kódem 8/10. To ovšem není tak úplně pravda, jelikož vstupní data z linkové vrstvy nejsou 8 bitová, ale devítibitová. Devátý bit se používá pro odlišení datového znaku a speciálního řídicího znaku. Logická 0 značí datový znak a logická jednička řídicí znak. V literatuře [1] je použito označení D jako datový symbol a K jako řídicí symbol. Kódovaný symbol má na prvním místě 0 nebo 1 (D nebo K) a za ním následují 3 bity reprezentující číslo v kódu za tečkou, a po těchto třech bitech je 5 bitů reprezentujících číslo kódu před tečkou. Např. kódový symbol K 28.2 označuje začátek DLLP paketu. Na nejvyšším místě bude mít 1, protože jde o řídicí znak. Poté bude následovat posloupnost "010 11100" (5C hexadecimálně) , takže celý devítibitový znak bude 15Ch. V tabulce Tab. 2-2 je uveden seznam jednotlivých řídicích znaků.

♣ zdrojem obrázku je literatura [1]

Tab. 2-2 **Tabulka řídicích znaků**

	Zkratka názvu	Název	Popis
K28.5	COM	Comma	Používá se na začátku všech speciálních posloupností.
K27.7	STP	Start TLP	Označuje začátek *TLP*.
K28.2	SDP	Start DLLP	Označuje začátek *DLLP*.
K29.7	END	End	Označuje konec *TLP* a *DLLP*.
K30.7	EDB	End Bad	Označuje konec chybného paketu (zrušeného).
K23.7	PAD	Pad	Používá se při nastavování šířky linky a pořadí *Lanes*.
K28.0	SKP	Skip	Slouží ke kompenzaci odlišných bitových frekvencí.
K28.1	FTS	Fast Training Sequence	Používá se jako součást speciální posloupnosti při opouštění *L0s* úsporného režimu.
K28.3	IDL	Idle	Používá se ve speciální posloupnosti při přechodu vysílacího budiče do stavu *Idle*.
K28.4			Rezervováno.
K28.6			Rezervováno.
K28.7			Rezervováno.

2.14.2 Linková vrstva

Linková vrstva je prostřední vrstvou v rozhraní sběrnice PCI Express. Její funkce je přenos vysílaných paketů z transakční vrstvy do vrstvy fyzické a přenos přijatých paketů z fyzické vrstvy do vrstvy transakční.Hlavní úkol této vrstvy je zabezpečit tyto přenosy. V případě potřeby opravit chyby v paketech nebo je znovu poslat, pokud nebyly správně doručeny.Dále má linková vrstva obstarávat správu *Flow Control* a *Power Management*. Linkové vrstvy pro vzájemnou komunikaci mezi sebou používají speciální druh paketů, které se nazývají DLLP (*Data Link Layer Packet*). Úkony, které musí provádět linková vrstva jsou následující:

- **Zajištění integrity všech paketů pomocí CRC kódu a kontrola CRC kódu proti chybám**

- **Funkce *Power* managementu**
- **Zprostředkování TLP paketů mezi transakční a fyzickou vrstvou**
- **Příjem a odesílání linkových paketů DLLP**
- **Zajištění správného doručení paketů transakční vrstvy a jejich případné opakované poslání v případě jejich nedoručení pomocí *Retry Buffer***

2.14.2.1 Linkové pakety

Pro zajištění funkce linkové vrstvy se používají linkové pakety, které jsou označovány jako DLLP (*Data Link Layer Packet*). Tyto pakety jsou posílány mezi jednotlivými linkovými vrstvami. Tyto pakety se posílají pouze mezi dvěma porty na jedné lince, takže u nich nedochází k přesměrování pomocí přepínačů jako u paketů transakční vrstvy. Pakety linkové vrstvy lze rozdělit do skupin podle funkce, kterou tyto pakety zajišťují.

Flow Control

Tyto pakety se používají k zajištění funkce *Flow Control* systému řízení přenosu dat. Pomocí *Flow Control* paketů se přenáší informace o počtu kreditů v přijímacích bufferech v transakční vrstvě.Neboli určují velikost volného místa těchto front.

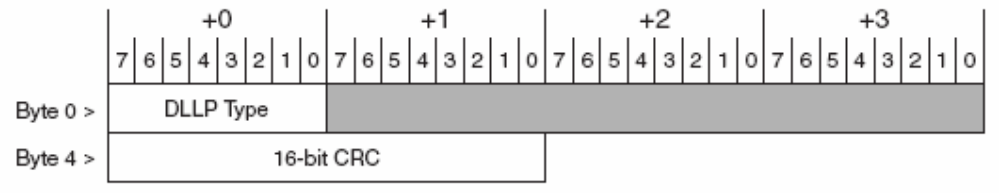
Potvrzování TLP

Pakety ACK – kladné potvrzování, se vysílají po určité době a potvrzují TLP se stejným sekvenčním číslem. Tímto potvrzením potvrdí s tímto TLP i všechny starší, neboli ty co měly nižší sekvenční číslo. Pokud je TLP chybně přijat, vyšle se záporné potvrzení-NAK, které po přijetí inicializuje znovu poslání TLP paketů.

Výrobcem definované

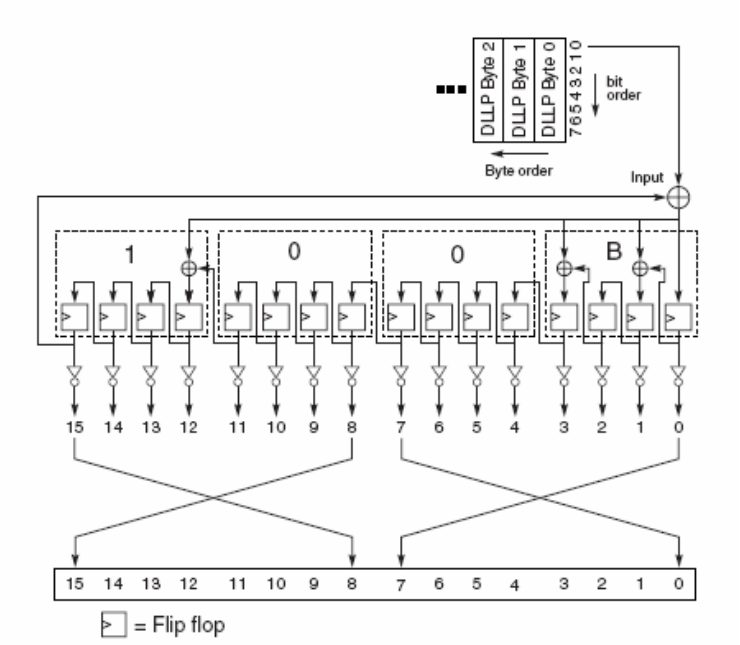
Běžně se nepoužívají a jejich implementace není povinná. Jsou definovány výrobcem příslušného zařízení.

Na obrázku Obr. 2-25 je vidět struktura DLLP paketu. Paket se skládá ze 6ti bajtů. První Byte označuje typ paketu. Poslední dva bajty obsahují 16ti bitové CRC, vypočítané z prvních 4 bajtů. Toto CRC zabezpečuje přenos paketu proti chybám.



Obr. 2-25 Struktura paketu linkové vrstvy ♣

Schéma výpočtu 16ti bitového CRC je na obrázku Obr. 2-26. Výpočet se provádí v posuvném registru.



Obr. 2-26 Výpočet 16ti bitového CRC pro DLLP ♥

Tab. 2-3 Mapování jednotlivých bitů do pole CRC

CRC Result Bit	**Corresponding Bit Position In the 16 bit CRC Field**	CRC Result Bit	**Corresponding Bit Position In the 16 bit CRC Field**
0	**7**	**8**	**15**
1	**6**	**9**	**14**
2	**5**	**10**	**13**
3	**4**	**11**	**12**
4	**3**	**12**	**11**
5	**2**	**13**	**10**
6	**1**	**14**	**9**
7	**0**	**15**	**8**

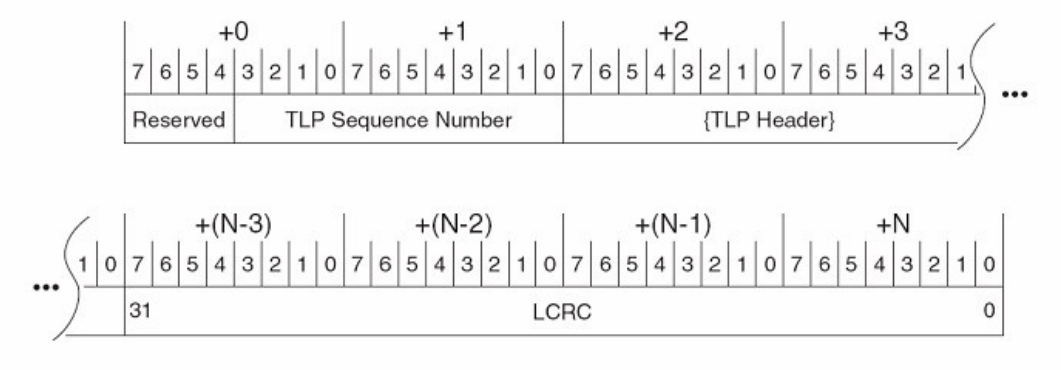
Vypočtené CRC se mapuje do paketu křížem. Mapovací tabulka je vyznačena vlevo v tabulce Tab. 2-3.

♣ zdrojem obrázku je literatura [1]
♥ zdrojem obrázku je literatura [1]

Integrita TLP

Za nejdůležitější úkol linkové vrstvy se dá považovat zajišťování integrity přenášených paketů transakční vrstvy. Pakety TLP obsahují data, jejichž přenos je smyslem rozhraní PCIe. Integrita TLP paketů se zajišťuje dvěma způsoby. Za prvé kontrolním součtem označovaném jako LCRC (*Link CRC*). Za druhé se jednotlivé pakety číslují.

Z pohledu linkové vrstvy jsou přijaté a odesílané pakety transakční vrstvy vnímány jako *black box*. Jejich sekvenční číslo je umístěno na začátek paketu. CRC je umisťováno na konec paketu. Postup je naznačen na obrázku Obr. 2-27.



Obr. 2-27 Očíslovaný paket TLP s CRC ♣

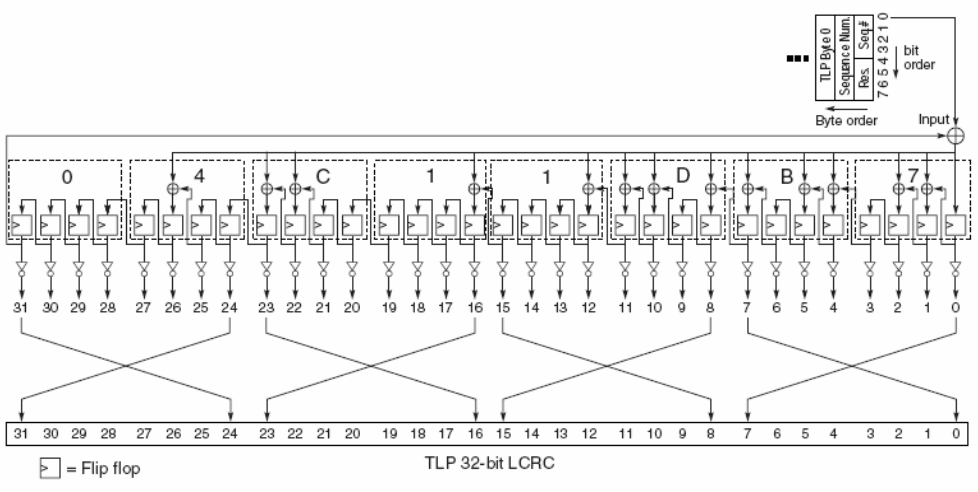
Sekvenční číslo

Sekvenční číslo se skládá z 12ti bitů. Při inicializaci je toto číslo nastaveno na 0. Při každém odeslání paketu se sekvenční číslo zvýší o 1. Na toto číslo je potom provedeno dělení modulo 4096 (takže pakety jsou se sekvenčními čísly od 0 do 4K). Na přijímací straně se potom snadno detekuje vynechaný či ztracený paket. Negativním potvrzením pomocí paketu DLLP NAK se potom vyvolá opětovné vysílání paketů.

LCRC

32bitové CRC zajišťuje integritu každého paketu TLP. Pro výpočet CRC se používají reservované bity, sekvenční číslo a TLP. Na přijímací straně se provede stejný výpočet a výsledky se mezi sebou srovnají.Výsledek výpočtu CRC je uložen v poli LCRC. Schéma posuvného registru na výpočet LCRC je na obrázku Obr. 2-28.

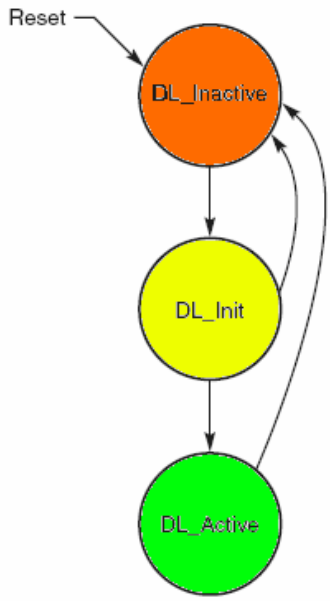
♣ zdrojem obrázku je literatura [1]



Obr. 2-28 Výpočet LCRC pro pakety TLP ♣

2.14.2.2 DLCMSM

Řízení systému linkové vrstvy zajišťuje stavový automat, který se nazývá DLCMSM (*Data Link Control and Management State Machine*). Základní funkce automatu je vyjádřena třemi stavy. Jsou to DL_*Inactive*, DL_*Init* a DL_*Active*. Tento automat slouží k inicializaci a provozu linkové vrstvy, zprostředkovává informaci o stavu linky vrstvě transakční a inicializuje FC (*Flow Control*) kreditní systém. Automat je na Obr. 2-29.



Obr. 2-29 DLCMSM Stavový automat ♥

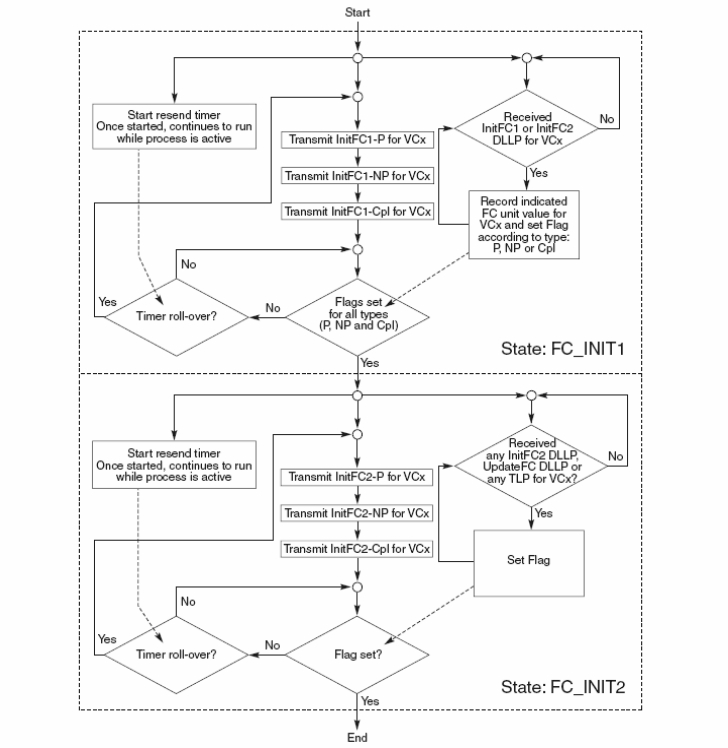
♣ zdrojem obrázku je literatura [1]
♥ zdrojem obrázku je literatura [1]

DL_Inactive

Do tohoto stavu se automat dostane, pokud je systém po resetu, je-li ukončen aktivní provoz linky a nebo pokud neproběhne inicializace systému zcela korektně. Stav DL*_Inactive* je výchozím stavem automatu. Při přechodu do tohoto stavu jsou všechny stavové hodnoty nastaveny na inicializační úroveň (respektive veškeré příznaky jsou vynulovány). Také veškerý obsah *Retry Buffer* je vynulován. Automat přejde do stavu DL*_Init* pouze v případě že není linka softwarově zakázána a že je signál *Physical Link Up* =1. nastaven fyzickou vrstvou.

DL_Init

V tomto stavu dochází k inicializaci *Flow Control* systému. Primárního virtuálního kanálu VC0. Pomocí DLLP paketů se zde vysílají a přijímají inicializační hodnoty FC_Init1 a FC_Init2. Do stavu DL*_Active* se přejde pouze v případě úspěšné inicializace *Flow Control*. Stavový digram *Flow Control* je na Obr. 2-30. Fyzická vrstva musí mít signál *Physical Link Up* =1. Pokud by se změnil stav fyzické vrstvy, nastavil by se signál *Physical Link Up*=0 a stavový automat by přešel zpět do výchozího stavu DL*_Inactive*.



Obr. 2-30 Inicializace Flow Control ♣

♣ zdrojem obrázku je literatura [1]

DL_Active

Tento stav vyjadřuje běžný provoz. TLP pakety jsou zpracovávány linkovou vrstvou a ta nastavuje pro transakční vrstvu signál DL_*Up*=1. Pokud by se změnil stav na fyzické vrstvě, tak fyzická vrstva nastaví signál *Physical Link Up*=0 a stavový automat přejde ze stavu DL_*Active* do stavu DL_*Inactive*.

2.14.2.3 Retry Buffer

Jedním z úkolů linkové vrstvy je také opětovné vysílání nedoručených TLP paketů. Tyto pakety je potřeba někde ukládat. Pro tento účel je součástí linkové vrstvy *Retry Bufferu*. Do něj jsou na vysílací straně ukládány všechny pakety doručené transakční vrstvou. Jestliže je nějaký paket nebo více paketů potvrzeno řídicím paketem linkové vrstvy ACK DLLP , tak jsou tyto pakety odstraněny z *Retry Bufferu* a je uvolněno místo těmito pakety zaplněné. Jestliže je *Retry Buffer* plný, tak nejsou další pakety akceptovány do té doby, než se místo v *Retry Bufferu* uvolní. Vysílání nepotvrzených paketů z *Retry Bufferu* nastane v případě, že přišel řídicí paket negativního potvrzení NAK DLLP, a nebo nebyl po nastavenou dobu paket potvrzen ACK. Velikost *Retry Bufferu* by měla být dostatečná, aby nemohl být zablokován v důsledku žádného místa v *Retry Bufferu*. Velikost *Retry Bufferu* není normou nijak specifikovaná, ale měla by se volit dostatečně velká, aby nedocházelo k příliš častému odesílání celého *Bufferu*.

2.14.3 Transakční vrstva

Tato vrstva je nejvyšší vrstvou v hierarchii PCI Express. Hlavní úkol transakční vrstvy je generování a zpracovávání paketů transakční vrstvy (TLP) . Další úkoly této vrstvy jsou *Flow Control* (ten svými funkcemi zapadá jak do transakční, tak do linkové vrstvy). Poté se transakční vrstva stará o *Virtual Channel* Management (neboli o správu virtuálních kanálů). Tato vrstva také zajišťuje správné řazení paketů. Nakonec může transakční vrstva zajišťovat integritu přenášených paketů transakční vrstvy pomocí 32bitového CRC kódu, který je označovaný jako ECRC. Tento kód ovšem není povinný, protože integrita TLP paketů pomocí CRC kódu (LCRC kód) je zajišťovány předchozí linkovou vrstvou.

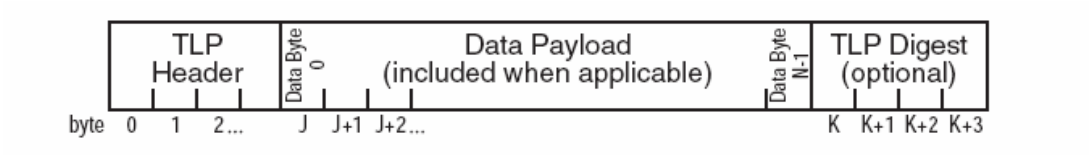
Úkoly transakční vrstvy jsou:

- **Kreditní systém řízení toku (*Flow Control*)**

- **Rozložení přijatého paketu transakční vrstvy a jeho zprostředkování do aplikační vrstvy a naopak složení odesílaného paketu na základě požadavku aplikační vrstvy**
- **Aplikace *Quality of Service* (QoS)**
- **Správa virtuálních kanálů**
- **Volitelné zajištění integrity paketů transakční vrstvy ECRC kódem**
- **Uchovávání a zprostředkování konfiguračních informací**

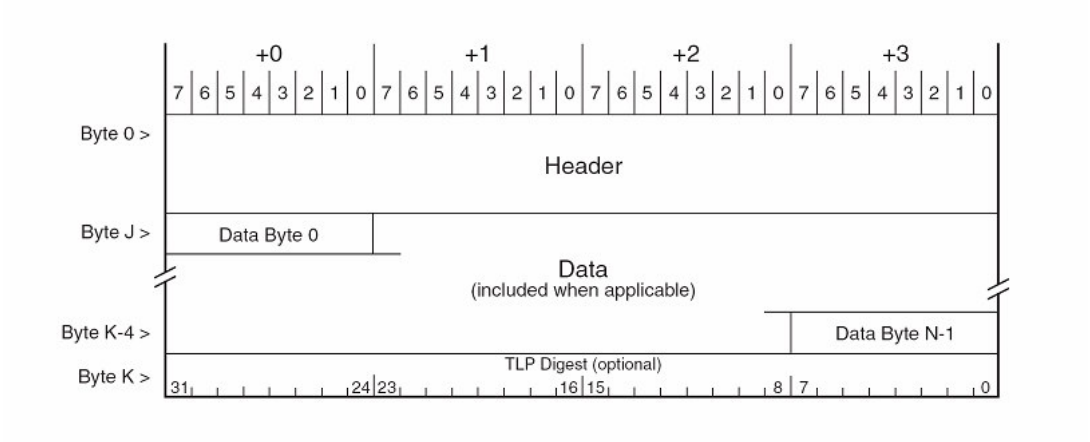
2.14.3.1 Pakety transakční vrstvy

Pro uskutečnění transakcí se používají pakety transakční vrstvy (TLP).



Obr. 2-31 Sériové zobrazení paketu transakční vrstvy ♣

Těchto paketů je několik typů, podle toho, pro jakou transakci jsou určeny. Struktura TLP paketu je na Obr. 2-32.



Obr. 2-32 TLP paket ♥

Na začátku každého paketu je hlavička, která se skládá ze 3 nebo ze 4DW. V případě, že daný typ paketu obsahuje data následuje datová oblast. Na konci může být nepovinné pole, označované jako TLP *Digest*, které může obsahovat 32 bitový kód ECRC pro zajištění integrity dat.

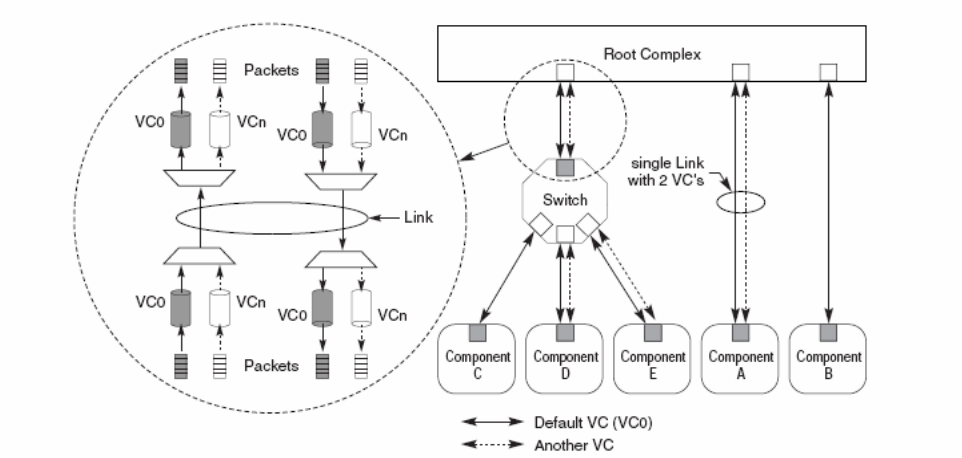
♣ zdrojem obrázku je literatura [2]
♥ zdrojem obrázku je literatura [1]

Pakety TLP mohou obsahovat požadavek na čtení nebo zápis do paměti, mohou číst nebo zapisovat do vstupně výstupního prostoru a nebo číst nebo zapisovat do konfiguračního prostoru. Podrobnější informace o jednotlivých typech paketů jsou ve specifikaci [1].

Za zmínku stojí ještě způsob odesílání těchto paketů. Kombinuje se v něm jak způsob *Big Indian*, tak i způsob *Little Indian*.

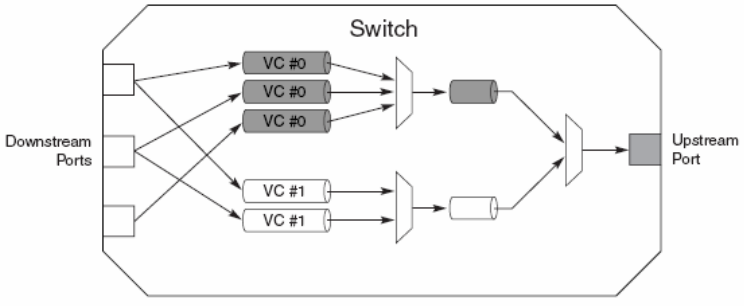
Adresa je odesílána v pořadí od MSB po LSB, aby bylo možné dopředné dekódování adresy, ale data jsou přenášena postupně od nejnižšího bajtu po nejvyšší. Ovšem jednotlivé bajty se však přenášejí v pořadí od MSB po LSB.

2.14.3.2 Virtuální kanály



Obr. 2-33 Spojování virtuálních kanálů ♣

Na sběrnici PCI Express slouží virtuální kanály k oddělení různých typů přenosů. Jednotlivé virtuální kanály mohou mýt nastavenou různou prioritu. Rozdělení virtuálních kanálů nijak nesouvisí s fyzickým rozdělením linky na jednotlivé Proudy (*Lanes*).

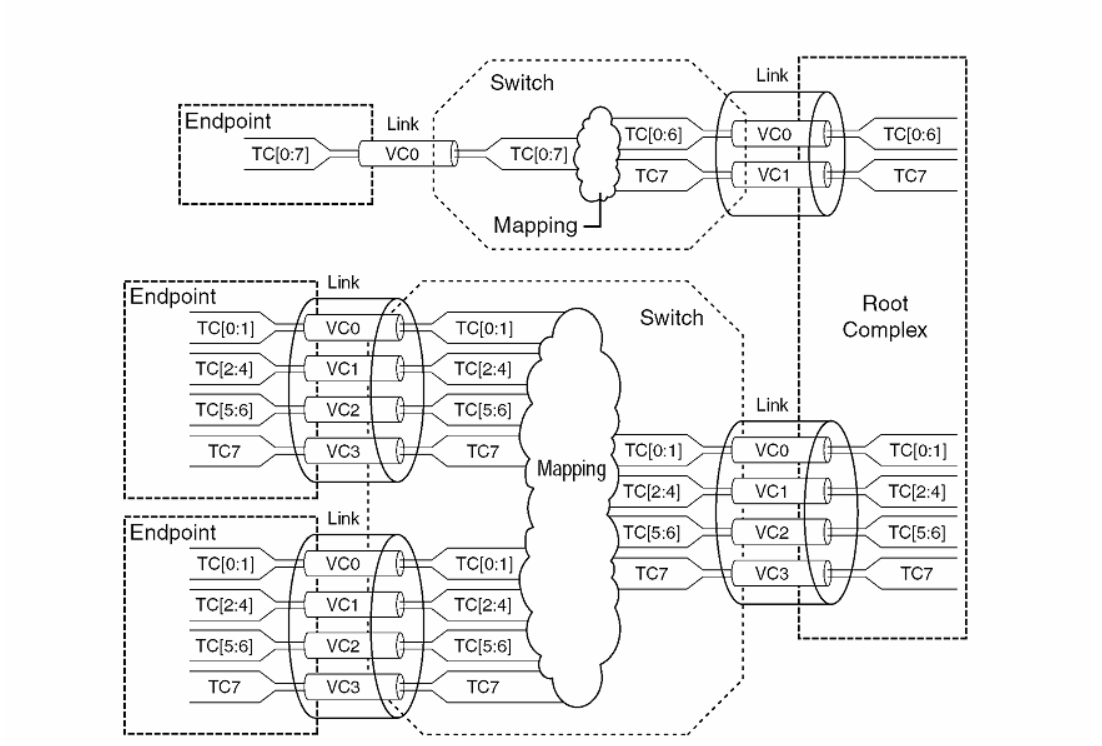


Obr. 2-34 Řazení virtuálních kanálu z front ♥

♣ zdrojem obrázku je literatura [1]
♥ zdrojem obrázku je literatura [1]

Kanály jsou rozděleny v úrovni transakční vrstvy. Pro přepínání jednotlivých virtuálních kanálů slouží časový *multiplex*. Každému kanálu je podle priority přiděleno určité časové kvantum, které může vyžívat. Při přenosu jsou jednotlivé virtuální kanály odlišeny pomocí tříd přenosu (*Traffic Classes –TCs*). Ty jsou uvedeny v každé hlavičce paketu transakční vrstvy. Vztahy mezi jednotlivými virtuálními kanály a třídami přenosu jsou určeny volitelným mapováním. Každý jednotlivý virtuální kanál používá vlastní řízení toku (*Flow Control Management*-FCM) a má tedy vyhrazen vlastní místo v přijímacích a odesílacích frontách (bufferech).

Na Obr. 2-35 jsou vidět struktury, jak jsou virtuální kanály mapovány na jednotlivé třídy přenosu.



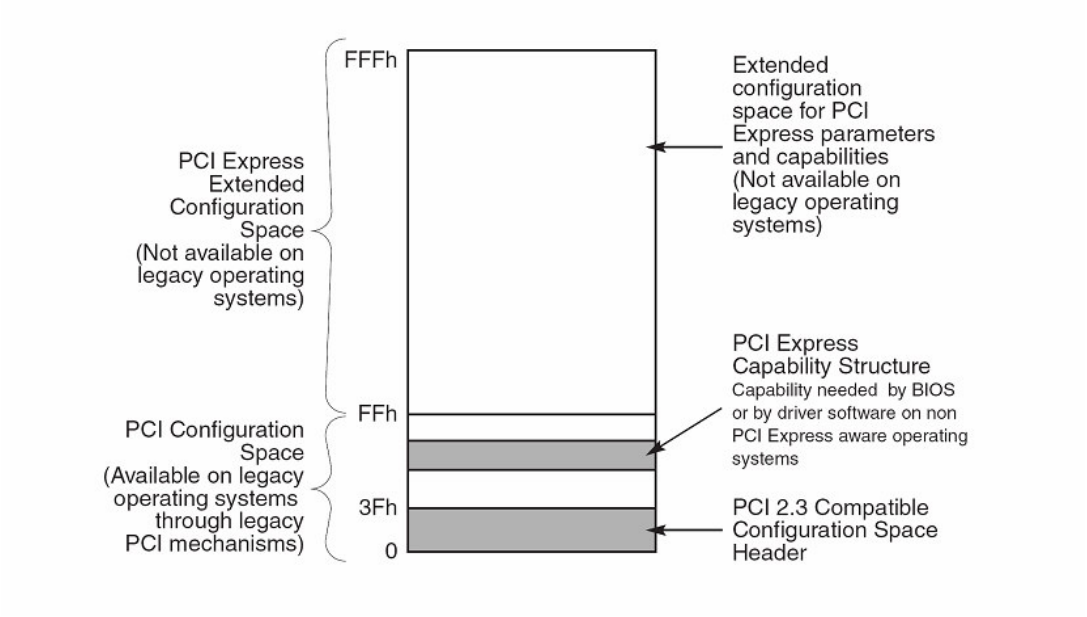
Obr. 2-35 Mapování VCs na TCs ♣

Koncová zařízení na PCI Express musí povinně implementovat pouze virtuální kanál VC0 a třídu přenosu TC0. Všechny ostatní virtuální kanály a třídy přenosu jsou při implementaci nepovinné. Každý přepínač musí přeposílat pakety s jakoukoliv třídou přenosu.

♣ zdrojem obrázku je literatura [2]

2.14.3.3 Konfigurační prostor

U PCIe je konfigurační prostor (*Configuration Space*) zpětně kompatibilní s konfiguračním prostorem PCI verze 2.3. Ovšem u konfiguračního prostoru PCI Express je zavedena celá řada rozšíření. Zvětšila se velikost z původních 256 byte u PCI na 4096 byte u PCI Express.Pro představu může posloužit obrázek Obr. 2-36.



Obr. 2-36 Rozšíření konfiguračního prostoru ♣

Zařízení PCIe ovšem nemusí rozšířenou část konfiguračního prostoru používat. Veškeré informace mohou být pouze ve spodních části 256 byte.

Další rozšíření je *Configuration Capabilities*. Pro všechna zařízení *Legacy Endpoint* jsou *Configuration Capabilities* nepovinná. Ovšem všechna zařízení *PCI Express Endpoint* musí podporovat tato rozšíření:

PCI Express Capability Structure – definice možností PCI Express zařízení, definice linky a řídicích registrů, jejichž pomocí se nastavují některé parametry

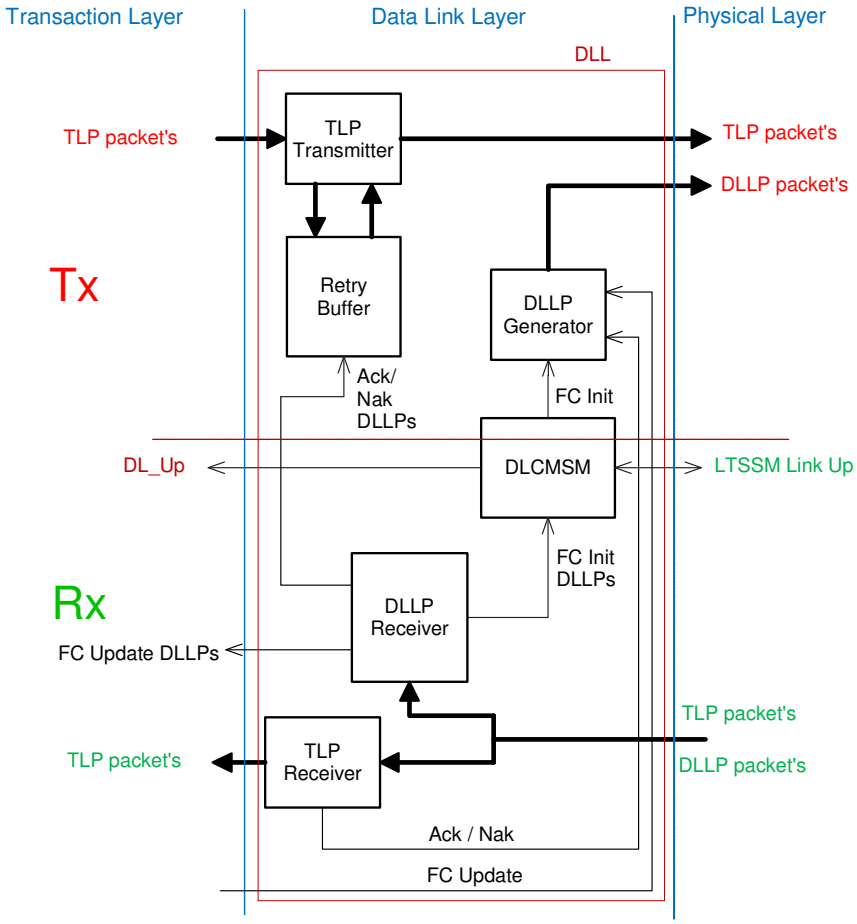
Message Signaled Interrupts Capability Structure – definice možností zasílaní přerušení pomocí zpráv. Tato funkce je povinná, pokud zařízení používá přerušení

PCI Power Management Capability Structure – definice možností *Power* managementu

♣ zdrojem obrázku je literatura [1]

3 Implementace linkové vrstvy

Linková vrstva je podle normy PCI Express rozhraní druhé úrovně, které se nachází mezi fyzickou vrstvou (*Physical Layer*) a transakční vrstvou (*Transaction Layer*). Z toho plyne, že linková vrstva je jedním z prostředníků umožňujících komunikaci a přenos mezi jednotlivými vrstvami.



Obr. 3-1 Blokové schéma Data Link Layer (DLL) ♣

Hlavní doménou Linkové vrstvy je zprostředkování vysílaných paketů z transakční vrstvy do vrstvy fyzické a opětovné příjímání těchto paketů z vrstvy fyzické a jejich doručení vrstvě transportní. Tyto pakety se označují zkratkou TLP (*Transaction Layer Packet*), což by v překladu mohli být pakety pro transakční vrstvu. Linková vrstva zabezpečuje přenos těchto paketů. V případě, že by některý z paketů nebyl doručen nebo byl-li poškozen, linková vrstva zajišťuje jeho opětovné vysílání. Jako další funkci, kterou

♣ obrázek vytvořen pomocí programu Smart Draw 2007 podle [2]

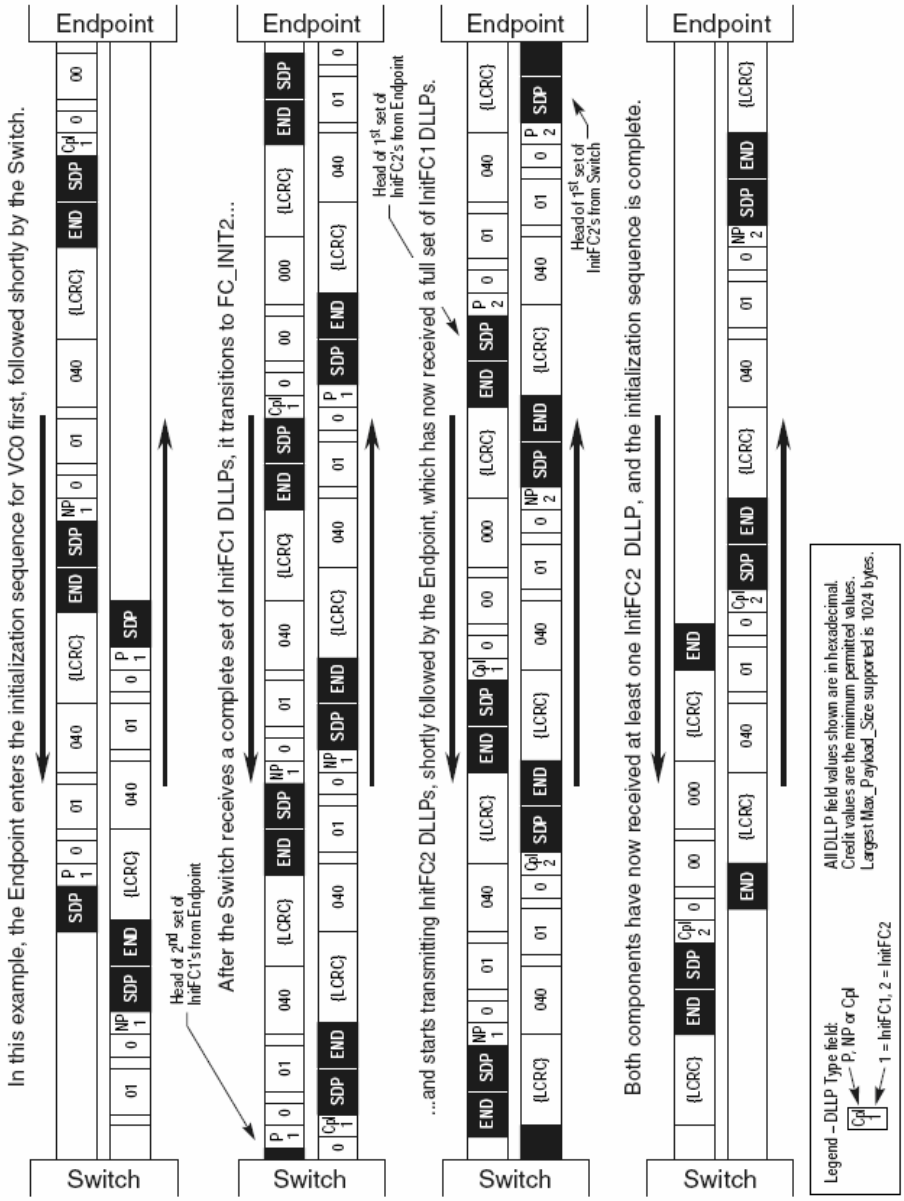
má linková vrstva podle specifikace zajišťovat, patří nastavení a transport *Flow Control* systému. *Flow Control* (FC) se využívá k řízení přenosu dat. Dále by linková vrstva měla zajišťovat *Power Management*. Pro komunikaci a řízení vzájemně spojených zařízení PCI Express se používají pakety linkové vrstvy. Tyto pakety se označují DLLP (*Data Link Layer Packet*).

Všechny tyto funkce jsou detailně popsány v kapitole 3 literatury [1]. Nutno říci, že ne všechny funkce musejí být provedeny přesně v dané vrstvě. Jednotlivé vrstvy se mohou částečně překrývat a tudíž může být některá z funkcí provedena v sousední vrstvě v důsledku snadnější implementace nebo pro účely ladění systému přenosu. Ve své práci jsem přidal do linkové vrstvy generování a detekci *framing* symbolů pro účely ladění systému. Naopak jsem do linkové vrstvy nezahrnul funkci *Power* Managementu, který jsem pro prvotní zprovoznění systému v minimalistické variantě 1x nepotřeboval a jež podle specifikace mezi činnosti linkové vrstvy patří. Blokové schéma zapojení Linkové vrstvy je na Obr. 3-1. Všechny jednotky linkové vrstvy jsou časovány frekvencí 62,5 MHz, která je odvozena od základního kmitočtu fyzické vrstvy (PHY) 250 MHz.

Řízení linkové vrstvy provádí stavový automat DLCMSM. Ten zajišťuje inicializaci linkové vrstvy a řízení *Flow Control* (FC) systému. O stavu linkové vrstvy DLCMSM informuje jak transakční vrstvu, tak vrstvu fyzickou. Z fyzické vrstvy jsou vstupní data přivedena na vstup DLLP *Receiveru* a TLP *Receiveru*. Vstupní data jsou ve formátu 7x9 bitů. Tato data jsou již v paralelní formě od fyzické vrstvy označené jako *Physical Layer* MAC a ne od PCI Express PHY, ve které jsou již data sériově zpracovávána. Pozor tedy na zaměnění těchto dvou pojmů.TLP *Receiver* zajišťuje kontrolu a detekci přijatých TLP paketů a celé TLP pakety přivádí v paralelní formě do transakční vrstvy. DLLP *Receiver* kontroluje, detekuje a dekóduje DLLP pakety získané z fyzické vrstvy (MAC) , dává vědět DLCMSM automatu o přijatých FC paketech a říká *Retry Bufferu*, jestli byly některé TLP pakety potvrzeny. Generátor DLLP paketů generuje DLLP pakety na základě požadavků od DLCMSM stavového automatu. Tyto pakety vysílá do fyzické vrstvy v paralelní formě 4x9 bitů. TLP pakety poslané transakční vrstvou jsou předány do jednotky TLP *Transmitter*, který všechny přijaté pakety ukládá do *Retry Bufferu* a odesílá je v paralelní podobě 4x9 bitů do fyzické vrstvy.

PCI EXPRESS ***Flow Control Management***

Smyslem *Flow Control* je ochrana proti přetečení přijímacích *bufferů*, jestliže požadavky na transakce přicházejí v kratším intervalu, než dokáže jednotka zpracovávat. *Flow Control* Management ovlivňuje svou funkcí jak linkovou, tak transakční vrstvu. Systém běží mezi dvěma zařízeními na téže lince (*point-to-pint*) a ne na koncových zařízeních transakce (*end-to-end*). To umožňuje zajistit přijímací *buffery* propojovacích zařízení - přepínače(*Switch*).



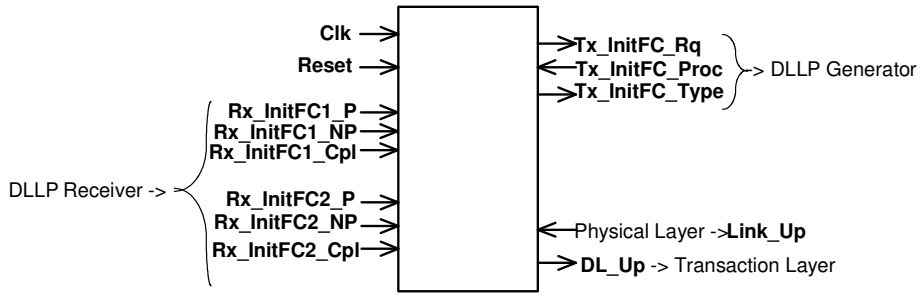
Obr. 3-2 Inicializace Flow Control ♣

♣ zdrojem obrázku je literatura [1]

Přijímací buffery se nalézají na úrovni transakční vrstvy. Ovšem pro přenášení veškerých informací týkajících se *Flow Control* se používá linková vrstva. Ta má pro tento typ dat vyčleněny DLLP pakety. Linková vrstva tyto pakety jak generuje, tak také přijímá. Jednotka, která vyjadřuje poměr zaplnění přijímacího *bufferu* se nazývá kredit. Tyto kredity jsou rozděleny do několika skupin podle obsažené informace. Kredity se dělí podle typu transakce. Na Obr. 3-2 je příklad inicializace *Flow Control* (Více informací je v kapitole 3.3 literatury [1])

3.1 DLCMSM

Tuto jednotku tvoří stavový automat. DLMSM (*Data Link Control and Management State Machine*) přijímá signál *Link_Up* od stavového automatu fyzické vrstvy a jejím stavu. Dále má na vstupu signály od jednotky DLLP *Receiver*. Ten informuje o jednotlivých přijatých paketech týkajících se inicializace *Flow Control*.Dále má výstupní signál DL_*Up*, kterým informuje transakční vrstvu, že je ve stavu DL_*Active* a že linková vrstva je připravena přijímat TLP pakety. Zbylé signály slouží ke komunikaci s jednotkou DLLP *Generator*, které říká v jakém stavu inicializace se nachází, a jaké FC pakety má tedy DLLP *Generator* odesílat.



Obr. 3-3 Zapojení entity DLCMSM ♣

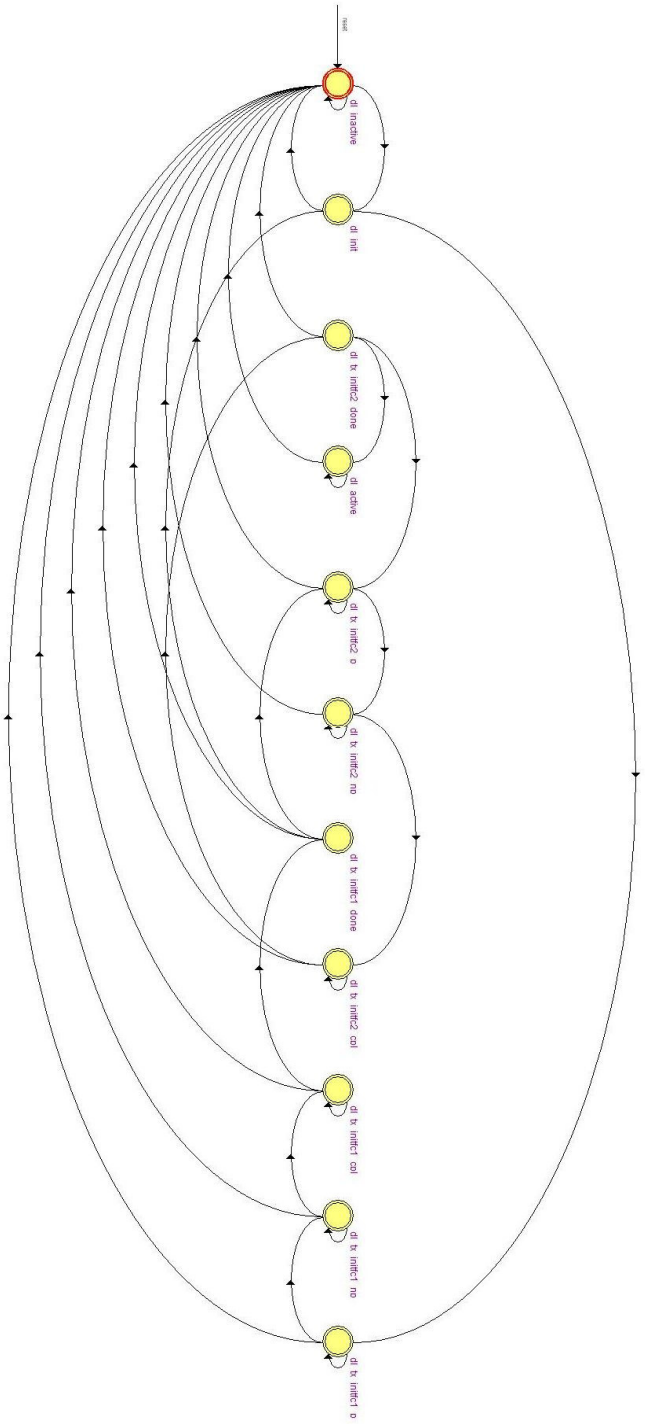
Zapojení DLCMSM je znázorněno na obrázku Obr. 3-3 .

Vnitřní schéma stavového automatu je na. Popis stavů je následující:

DL_*Inactive*: Stav po inicializaci, *Flow Control* type nastaven na 000. Požadavek na inicializaci FC nastaven na 0. Pokud fyzická vrstva hlásí *Link_Up* = '1', tak automat přejde do stavu DL_*Init*.

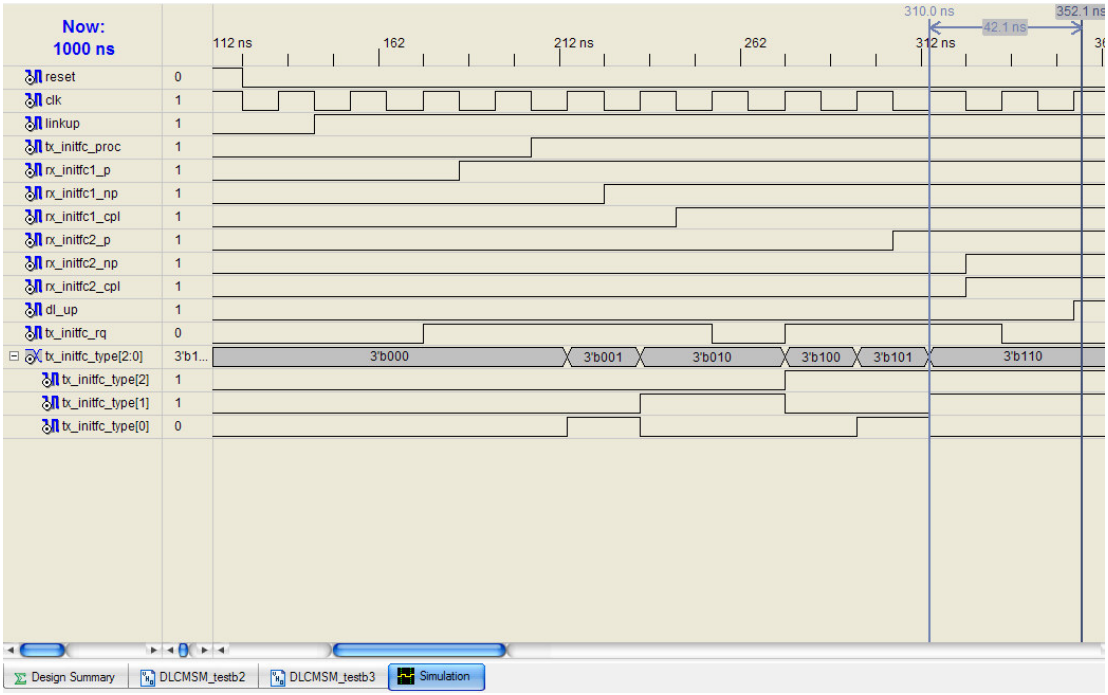
♣ obrázek vytvořen pomocí programu Smart Draw 2007podle [2]

- DL_*Init*: Nastavení požadavku FC na 1. Přechod do stavu DL_*Tx*_InitFC1_P.
- DL_Tx_InitFC1_P: Čeká na nastavení signálu Tx_InitFC_Proc = '1', když je signál nastaven na 1 tak nastaví FC type na 001, přechod do stavu DL_Tx_InitFC1_NP.
- DL_Tx_InitFC1_NP: Čeká na nastavení signálu Tx_InitFC_Proc = 1', když je signál nastaven na 1 tak nastaví FC type na 010, přechod do stavu DL_Tx_InitFC1_Cpl.
- DL_Tx_InitFC1_Cpl: Čeká na nastavení signálu Tx_InitFC_Proc = '1', když je signál nastaven na 1 tak nastaví požadavek FC na 0. Přechod do stavu DL_Tx_InitFC1_Done
- DL_Tx_InitFC1_Done: Pokud jsou nahozeny všechny flagy FC1 systému, tak nastaví požadavek na vysílaní FC2, inicializace FC2 a přechod do DL_Tx_InitFC2_P .
- Dále probíhá inicializace a nastavování FC2 stejným způsobem jako FC1 pouze s nastavováním typů FC2 místo FC1. Po úspěšné inicializaci dojde k přechodu do stavu DL_*Active* .



Obr. 3-4 DLCMSM stavový automat ♣

♣ obrázek byl vysyntetizován nástrojem Quartus II ze zdrojového VHDL entity



Obr. 3-5 Simulace DLCMSM ♣

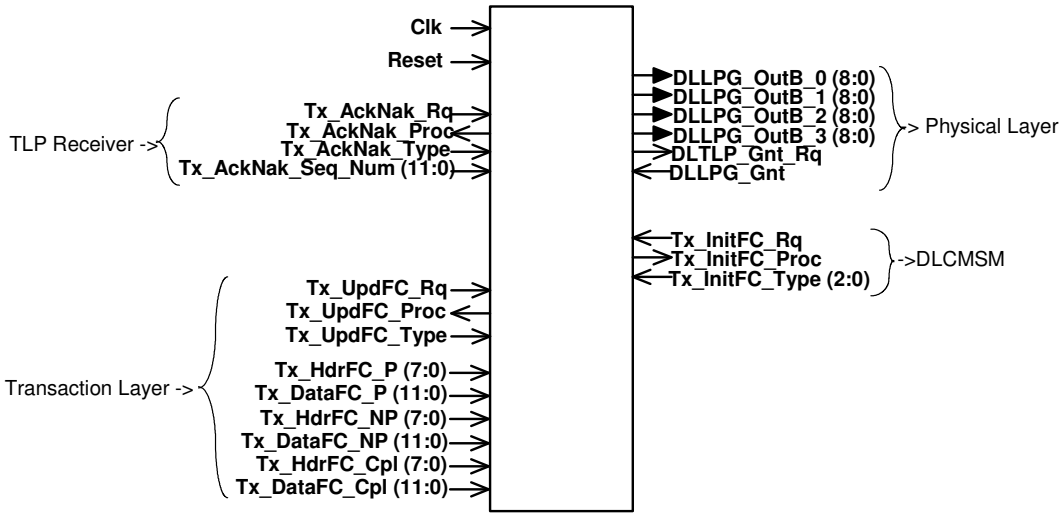
Na Obr. 3-5 je vidět průběh simulace stavového automatu podle napsaného testbenche. Jak samotná entita ve VHDL, tak i testbench jsou součástí přílohy 1. V printscreenu simulace je vidět, jak po obdržení všech paketů pro FC dojde k nahození signálu dl_*up*. V printscreenu je vidět pouze první část testbenche po resetu.

3.2 DLLP Generator

Jednotka DLLP *Generator* generuje DLLP pakety. Tyto pakety jsou vytvářeny na základě požadavků od jednotek DLCMSM, TLP *Receiver*, a od transakční vrstvy. Pakety jsou zakódovány, je vypočteno CRC a je aplikován *framing* pomocí symbolů SDP a END z tabulky Tab. 2-2 Tabulka řídicích znaků. *Framing* je v normě aplikován až ve fyzické vrstvě, ale pro účely ladění tvorby a vysílání paketů jsem jej již zahrnul jako součást linkové vrstvy. Pro výpočet CRC je použit paralelní algoritmus z literatury X. CRC je v tomto případě 16ti bitové a počítá se z 32 bitových dat.

Generované pakety jsou posílány do fyzické vrstvy (MAC) v paralelní formě 4x9 bitů. Blokové schéma jednotky je znázorněno na Obr. 3-6.

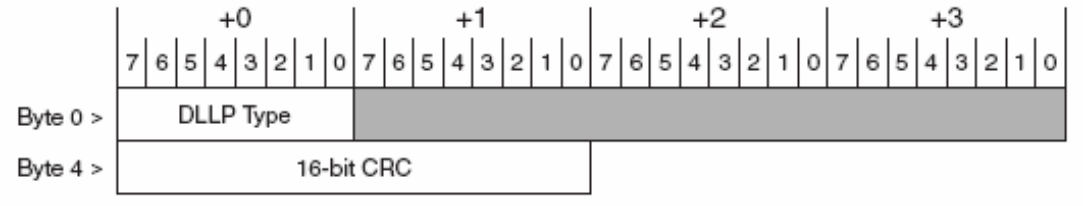
♣ obrázek vytvořen v simulačním prostředí Xilinx ISE 8.1



Obr. 3-6 Zapojení entity DLLP Generator ♣

DLLP pakety

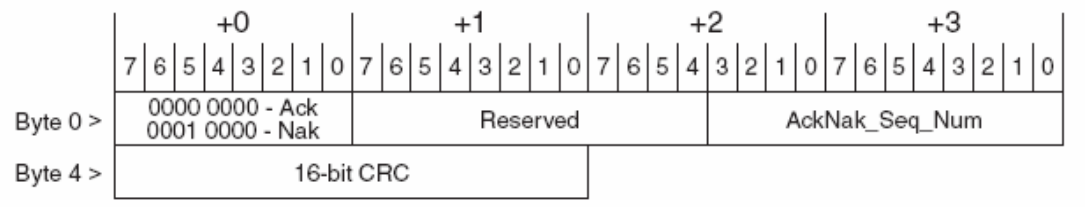
Všechny DLLP pakety musejí být na začátku vyplněny samými nulami. Všechny DLLP pakety obsahují jejich typ (kódové označení) a 16ti bitové CRC, tak jak je ukázáno na Obr. 3-7 .



Obr. 3-7 Paket DLLP ♠

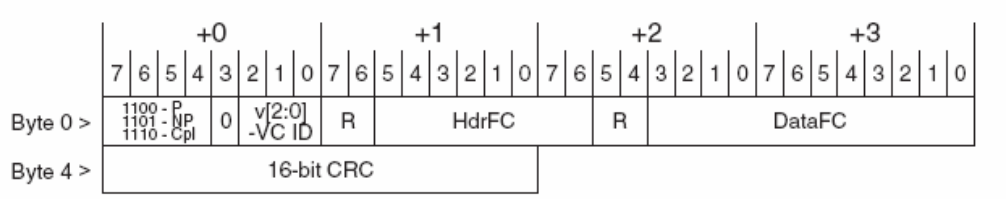
Typy jednotlivých kódů a k nim náležících paketů jsou v tabulce Tab. 3-1.

Na následující sadě obrázků Obr. 3-8 až Obr. 3-10 jsou uvedeny některé konkrétní typy DLLP paketů.

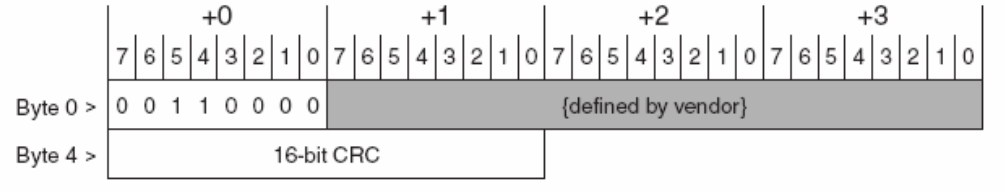


Obr. 3-8 Formát Ack a Nak DLLP paketu ♥

♣ obrázek vytvořen pomocí programu Smart Draw 2007podle [2]
♠ zdrojem obrázku je literatura [1]
♥ zdrojem obrázku je literatura [1]



Obr. 3-9 Formát DLLP paketu pro InitFC2 ♣

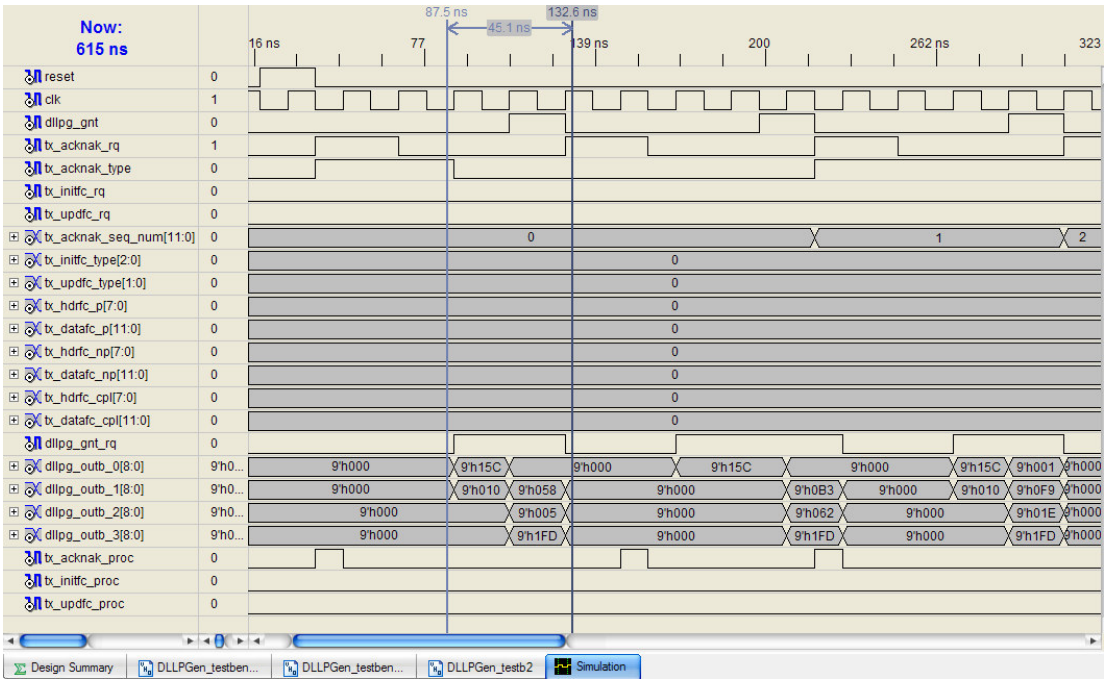


Obr. 3-10 Formát DLLP paketu definovaného výrobcem ♥

Tab. 3-1 Kódování DLLP

Encodings	DLLP Type
0000 0000	Ack
0001 0000	Nak
0010 0000	PM_Enter_L1
0010 0001	PM_Enter_L23
0010 0011	PM_Active_State_Request_L1
0010 0100	PM_Request_Ack
0011 0000	Vendor Specific – Not used in normal operation
0100 $0v_2v_1v_0$	InitFC1-P (v[2:0] specifies Virtual Channel)
0101 $0v_2v_1v_0$	InitFC1-NP
0110 $0v_2v_1v_0$	InitFC1-Cpl
1100 $0v_2v_1v_0$	InitFC2-P
1101 $0v_2v_1v_0$	InitFC2-NP
1110 $0v_2v_1v_0$	InitFC2-Cpl
1000 $0v_2v_1v_0$	UpdateFC-P
1001 $0v_2v_1v_0$	UpdateFC-NP
1010 $0v_2v_1v_0$	UpdateFC-Cpl
All other encodings	Reserved

♣ zdrojem obrázku je literatura [1]
♥ zdrojem obrázku je literatura [1]

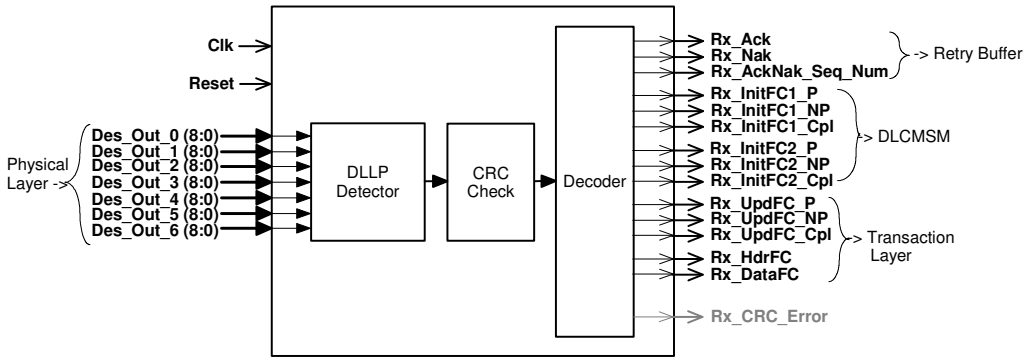


Obr. 3-11 Simulace DLLP Generator ♥

Na obrázku Obr. 3-11 je vidět jedna ze simulací DLLP Generátoru. Vyznačený paket je DLLP Nak se sekvenčním číslem 0. Na začátku paketu je vidět *framing* symbol 15Ch, což označuje SDP a na konci paketu je 1FDh, což označuje END. Na outb_1 je vidět 010h, což označuje DLLP paket typu Nak.

3.3 DLLP Receiver

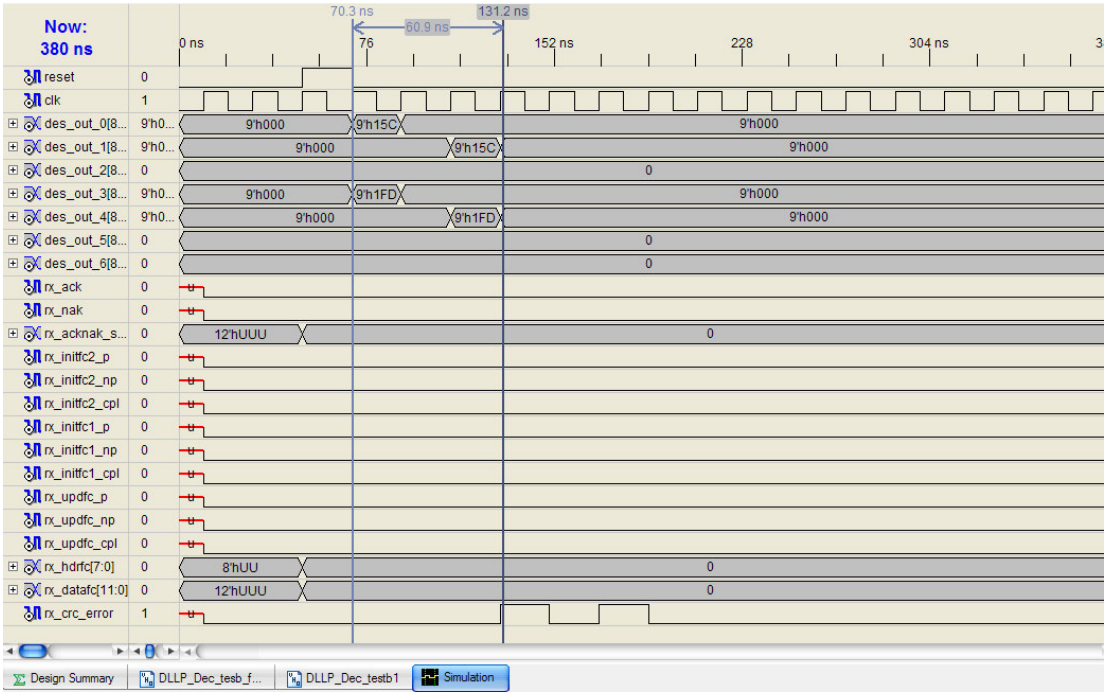
Tato jednotka provádí detekci a dekódování DLLP paketů. Vstup do jednotky je z fyzické vrstvy



Obr. 3-12 Zapojení entity DLLP Receiver ♣

♥ obrázek vytvořen v simulačním prostředí Xilinx ISE 8.1
♣ obrázek vytvořen pomocí programu Smart Draw 2007podle [2]

Informaci o případném přijetí paketu podává do *Retry Bufferu*, DLCMSM a do transakční vrstvy. Na vstupu je nejprve provedena detekce *framing* symbolů SDP a END. Potom se zkontroluje integrita paktu výpočtem nového CRC a tato hodnota se porovná s přijatým CRC v paketu. Pokud se tyto hodnoty shodují, provede se dekódování paketu. Nejdříve se zjistí jeho typ, poté se v paketu dekódují případné další informace. Podle těchto informací se nastaví příslušné výstupy jednotky. Při detekované chybě v CRC se nahodí výstupní chybový signál Rx_CRC_*Error*. Jednotka je zobrazena na Obr. 3-12 .



Obr. 3-13 Simulace DLLP Receiveru ♣

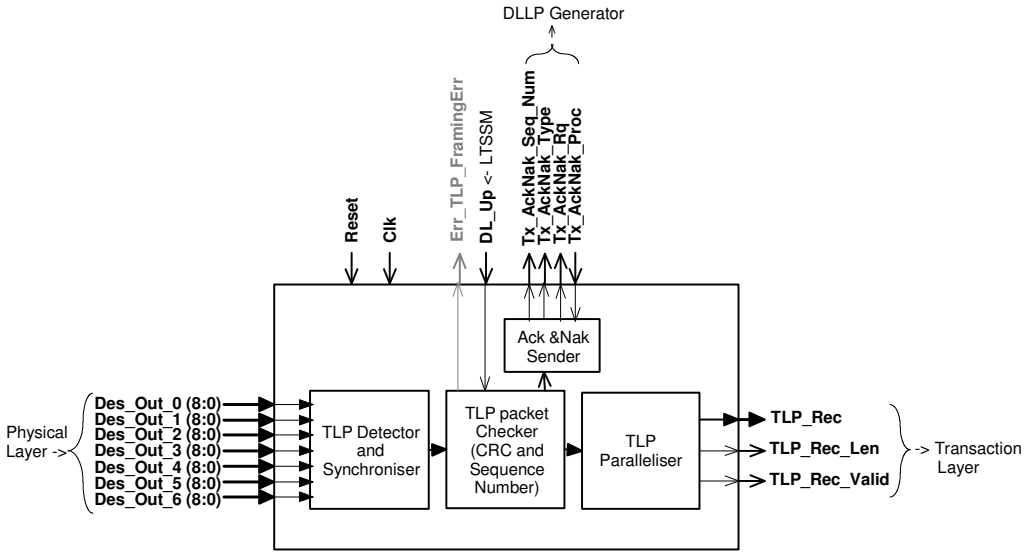
Na obrázku Obr. 3-13 je na simulaci testbenche vidět *framing* symbol DLLP paketu SDP a END. Všechna data jsou ponechána prázdná a je posláno vadné CRC. Vypočtené 16ti bitové CRC pro 32 bitová data se neshoduje se zaslaným CRC, které je součástí paketu, a tudíž je nahozen signál Rx_CRC_*Error*, protože byla detekována v poslaném paketu chyba.

3.4 TLP Receiver

TLP *Receiver* provádí detekci přijatých paketů. Schéma jednotky je na Obr. 3-14 .Tato jednotka je zodpovědná za zpracovávání TLP paketů transakční vrstvou. Data vstupují do TLP *Receiveru* z fyzické vrstvy v paralelní formě 7x9 bitů.TLP *Receiver*

♣ obrázek vytvořen v simulačním prostředí Xilinx ISE 8.1

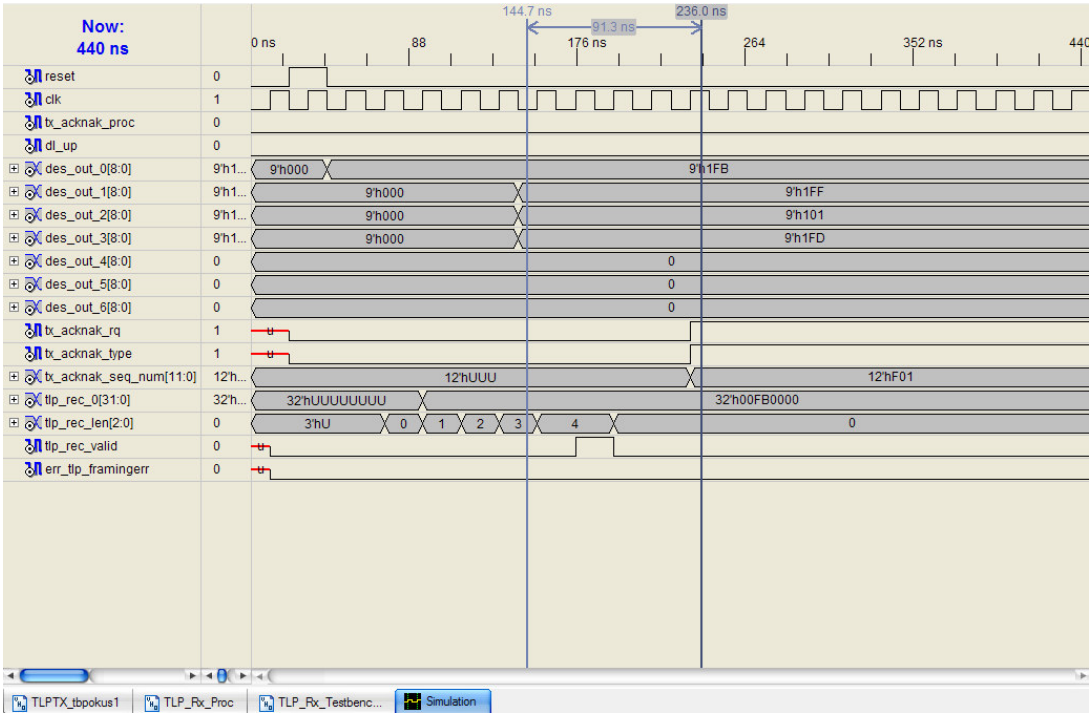
provádí jejich převod na 32bitová data Na počátku je provedena detekce TLP paketu podle *framing* symbolů STP a END. Poté se z dat odstraní a je provedena 32bitová synchronizace. Nyní jsou TLP pakety předávány synchronně se začátkem paketu transakční vrstvě. Poté se kontroluje integrita přijatých dat pomocí LCRC kódu a sekvenčního čísla paketu. Po kontrole těchto údajů jsou skrz DLLP *Generator* posílány potvrzovací pakety DLLP, a to buď Ack nebo Nak. Všechny pakety, které projdou kontrolou integrity, jsou zbaveny LCRC a sekvenčního čísla, paralelizovány na 32bitové rozhraní a předány transakční vrstvě. Chybový signál Err_TLP_FramingErr je nahozen při detekované chybě v přijatém TLP paketu.



Obr. 3-14 Zapojení entity TLP Receiver ♣

Na Obr. 3-15 je zobrazen testbench entity TLP Receiver. Na des_out_0 je přiveden symbol FB, což značí začátek TLP paketu (STP). Na des_out_3 je přiveden symbol FD, což značí konec paketu (END). CRC číslo přijatého paketu a CRC vypočtené se shodují, a tak je nahozen signál tlp_rec_valid. Ovšem přijaté číslo paketu je F01 a očekávané číslo je 001, takže je nahozen signál tx_acknak_rq pro odeslání DLLP paketu a signál tx_acknak_type je nastaven na 1, což značí negativní potvrzení přijetí TLP paketu před vypršením časovače a požadavek na odeslání DLLP paketu Nak.

♣ obrázek vytvořen pomocí programu Smart Draw 2007podle [2]

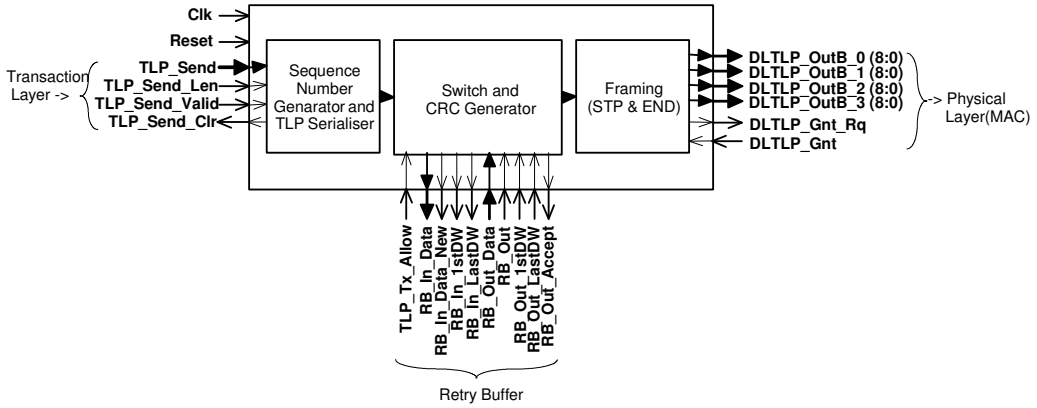


Obr. 3-15 Simulace TLP Receiveru ♣

3.5 TLP Transmitter

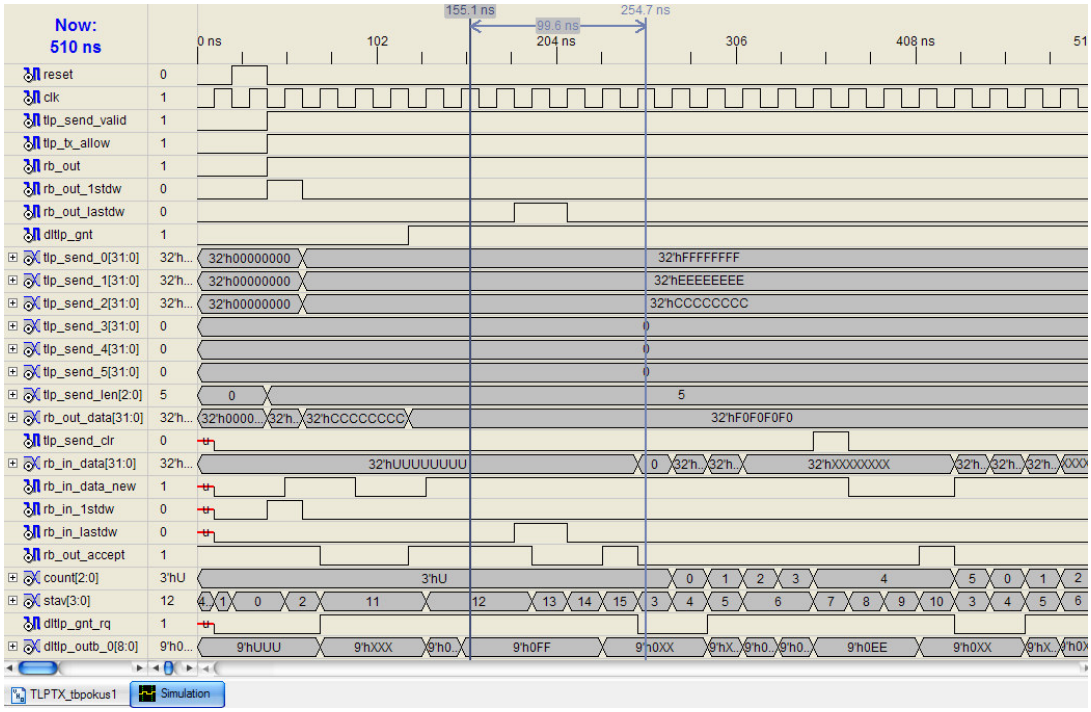
Tato jednotka se používá k zajištění funkcí při odesílání TLP paketů. Jsou to ty pakety, které generuje transakční vrstva a jsou určeny k odeslání fyzickou vrstvou. Pakety vstupují do TLP *Transmitteru* z transakční vrstvě v paralelní podobě. To znamená že přijde celý paket najednou. Tomuto paketu je přiděleno sekvenční číslo, a potom se provede částečná serializace paketu. Paket je rozdělen do 32 bitových bloků. Těmto blokům se vypočítá CRC kód, který se přidělá na konec tohoto paketu. Každý paket, který je odesílaný transakční vrstvou je zároveň po 32 bitech kopírován do *Retry Bufferu*. Je to proto, aby mohl být znovu odeslán v případě poškození či nedoručení TLP paketu. Detekční CRC kód se do *Retry Bufferu* neukládá. V případě potřeby odesílání paketů z Retry *Bufferu* jsou data čtena po 32 bitech a je jim počítáno CRC. Povolení zpracování TLP paketů z transakční vrstvy je řízeno signálem TLP_Tx_*Allow*. To, jestli se budou pakety posílat z transakční vrstvy nebo z *Retry Bufferu* určuje signál RB_*Out*. Těsně před odesláním paketu je aplikován *framing*, přidají se symboly STP a END. Potom jsou pakety předány fyzické vrstvě. Blokové schéma jednotky je na Obr. 3-16.

♣ obrázek vytvořen v simulačním prostředí Xilinx ISE 8.1



Obr. 3-16 Zapojení entity TLP Transmitter ♣

Na Obr. 3-17 je vidět simulace TLP *Transmitteru*. Je označen 1. a poslední dw a signál rb_out_accept značí, že se mají pakety posílat z transakční vrstvy a ne z Retry Bufferu. Na dltlp_outb_0 je vidět částečná serializace prvních 8mi bitů z transakční vrstvy tlp_send (FF a poté EE, 0na začátku označuje data, 1 by byl řídicí znak). Oproti zapojení entity je zde navíc signál stav, kterým jsem si simuloval logický analyzátor, aby jsem věděl v jaké části programu se obvod nachází.

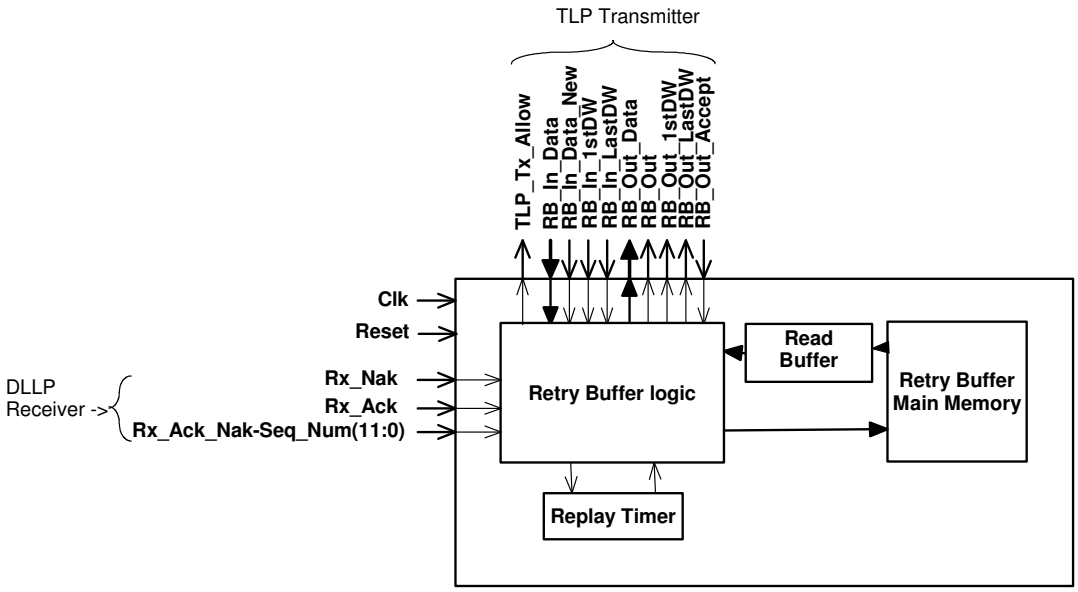


Obr. 3-17 Simulace TLP transmitteru

♣ obrázek vytvořen pomocí programu Smart Draw 2007podle [2]

3.6 Retry Buffer

Tato jednotka ukládá do své vnitřní paměti všechny TLP pakety, které se posílají z transakční vrstvy do vrstvy fyzické. V případě, že nebyly některé TLP pakety doručeny, nebo byly doručeny s chybou, tak tato jednotka zajišťuje jejich opětovné poslání. Po restartu linky je *Retry Buffer* prázdný. To znamená, že je nastaven signál TLP_Tx_*Allow*, který určuje, že jsou jednotkou TLP *Transmitter* zpracovávány TLP pakety z transakční vrstvy. Schéma zapojení jednotky a jejích signálů je na Obr. 3-18.



Obr. 3-18 Zapojení entity Retry Buffer ♣

Signály z DLLP *Receiver* indikují příjem potvrzovacích DLLP paketů. Paket Ack znamená kladné potvrzení paketu, paket Nak značí záporné potvrzení. Signál Rx_Ack_Nak-Seq_Num značí číslo potvrzovaného DLLP paketu. Toto číslo je 12ti bitové a umožňuje tudíž indexovat 4096 paketů. Všechny TLP pakety, které jsou potvrzeny kladným potvrzením jsou ihned z *Retry Bufferu* vymazány. Všechny TLP pakety, které jsou potvrzeny negativním potvrzením jsou z *Retry Bufferu* znovu odeslány do fyzické vrstvy. Také dosažení časového limitu odeslání *Reply Timeru* způsobí odeslání znovu všech paketů v *Retry Bufferu* obsažených. *Replay Timer* je nastaven na 178 hodinových cyklů. Hodnota 178 cyklů odpovídá doporučení specifikace. Tato hodnota může být ovšem v případě potřeby změněna snadnou úpravou konstanty RB_*Time_Out*. Velikost *Retry*

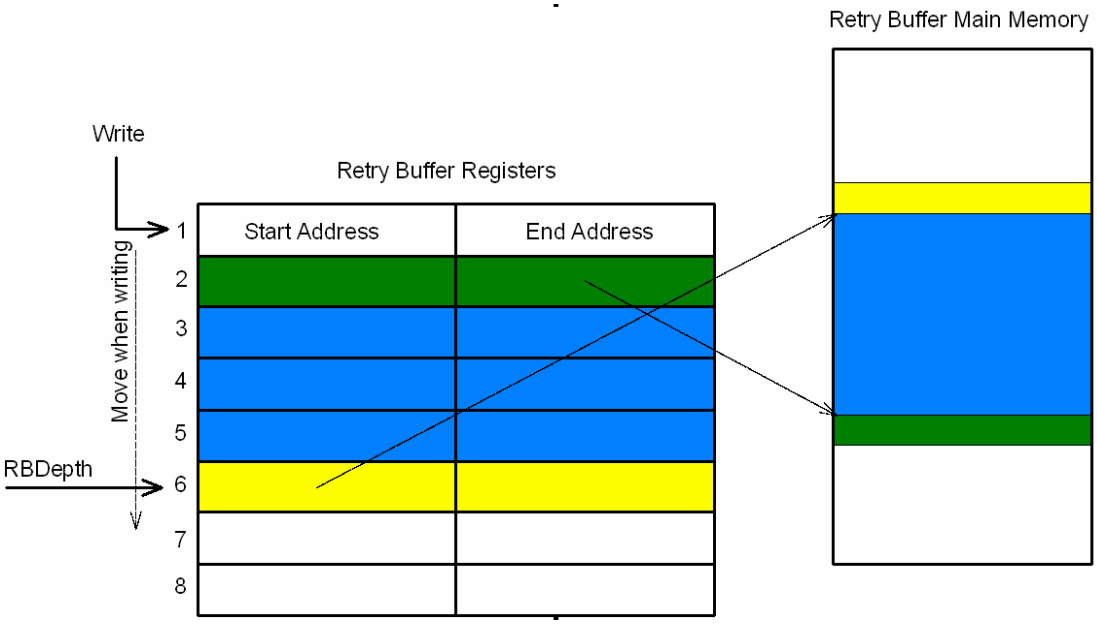
♣ obrázek vytvořen pomocí programu Smart Draw 2007podle [2]

Bufferu byla nastavena na 20 TLP paketů. Tato hodnota může být také snadno změněna pomocí konstanty RBDepth. Hodnota 20 TLP paketů byla zvolena podle doporučení specifikace. Byl brán zřetel na to, aby nedocházelo ke zbytečnému přemazávání paměti po jejím úplném zaplnění. Vyčítání TLP paketů z paměti je řešeno pomocí FIFO vyrovnávací paměti.

Informace o jednotlivých paketech jsou uchovávány v sadě registrů. V registru je vždy počáteční a koncová hodnota TLP paketu. Pokud není *Retry Buffer* prázdný, tak je v registru číslo 1 uložena adresa posledního přijatého paketu, to znamená toho nejnovějšího. RBDepth označuje počet paketů v *Retry Bufferu* a zároveň ukazuje na nejstarší nepotvrzený paket. Při každém zápisu TLP paketu do *Retry Bufferu* jsou všechny záznamy posunuty do registru s indexem o jedničku vyšším, a adresa nového paketu je uložena do registru číslo 1. Samozřejmě se zvýší i hodnota RBDepth.

Princip přístupu do paměti *Retry Bufferu* je znázorněn na obrázku Obr. 3-19.

Velikost obsazené paměti a její rozsah je tedy vymezen adresou *Start Address* v registru s nejstarším TLP paketem a adresou *End Address* v registru 1. V tom je obsažena adresa nejnovějšího uloženého paketu.

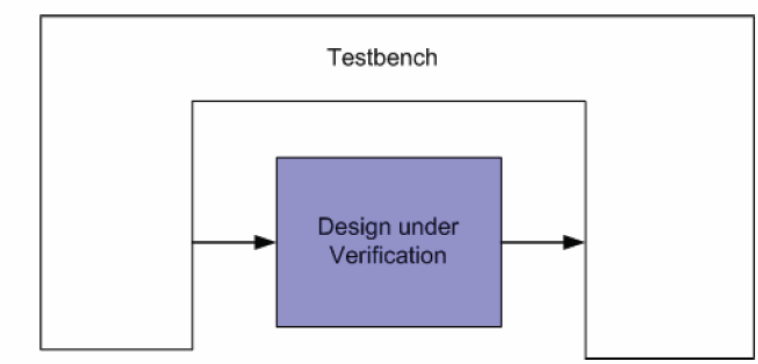


Obr. 3-19 Adresování paměti v Retry Buffer ♣

♣ obrázek vytvořen pomocí programu Smart Draw 2007podle [2]

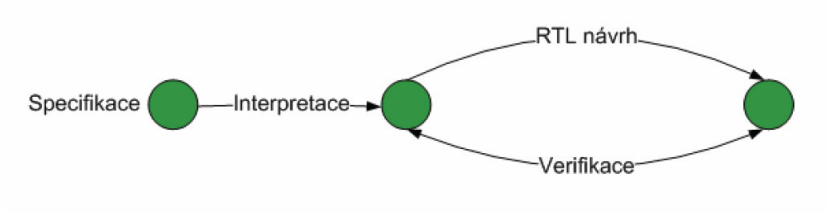
3.7 Simulace

Simulace jednotlivých entit jsem prováděl pomocí testbenchů napsaných v jazyce VHDL. Pro jejich spouštění a odladění jsem využíval převážně Xilinx ISE 8.1, který má IDE vyvinuté přímo pro testování a verifikaci. Model testbenche je znázorněn na Obr. 3-20.



Obr. 3-20 Model Testbeche ♥

Testbench vytváří virtuální svět testovanému obvodu. Vytváří vstupní stimuly a kontroluje výstupní odezvy. Verifikace je nalezení shody (a v horším případě míry shody) mezi dvěma representacemi stejné věci. Postup návrhu integrovaného obvodu ve VHDL i s verifikací je znázorněn na Obr. 3-21.

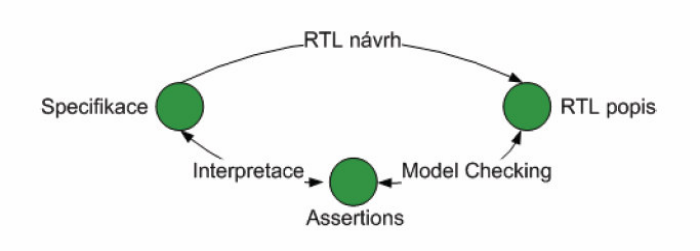


Obr. 3-21 Postup návrhu designu ♣

Dobrá simulace je vytvořena tak, že vyjme lidský faktor z průběhu simulace (např. optická kontrola waweformu). U testbenchů se to provádí tak, že se buď použijí testovací vektory a ty se poté automatizovaně kontrolují, a nebo se provede sada assertions , které representují testovací vektory ,,krok po kroku,, v jednotlivých iteracích a stavech obvodu. Ověřuje se, jestli obvod odpovídá zjednodušenému modelu obvodu. Assertions jsou logická tvrzení, která musí obvod splňovat. Jsou to například posloupnosti průběhů signálů

♥ zdrojem obrázku je literatura [12]
♣ zdrojem obrázku je literatura [12]

na rozhraní obvodu. Přesně tímto způsobem jsem postupně jednotlivé entity testoval. Model schématu tohoto typu testování je na .



Obr. 3-22 Model schématu kontroly modelu ♥

Přístupy k verifikaci

Existují tři typy funkčních verifikací, jsou to:

- ***Black box*** - tento test se připravuje bez znalostí o vnitřní struktuře a stavu testovaného obvodu
- ***White box –*** tento druh testu se připravuje se znalostmi o vnitřní struktuře obvodu, využívá se pro nastavení a otestování zajímavých stavů automatů a datových cest
- ***Grey box –*** tato verifikace je mixem *Black box* metody a *White box* metody

Ve své práci jsem používal druhý způsob testování jednotlivých entit, tedy *White box* verifikaci.

♥ zdrojem obrázku je literatura [12]

4 Závěr

Úkolem mé bakalářské práce bylo implementovat linkovou vrstvu standardu PCI Express. Motivací pro tvorbu této závěrečné práce bakalářské etapy studia pro mne byl hlavně fakt, že PCI Express se stále ještě řadí mezi novinky v oblasti IT.

Při tvorbě jednotlivých entit a jejich tesbenchů bylo nutné osvojit si trochu odlišný styl programování. Jazyk VHDL je značně odlišný od jiných programovacích jazyků. U programovacích jazyků, jako jsou například Java nebo C++ se provádí postupné (sekvenční) vykonávání jednotlivých příkazů. V jazyce VHDL se vykonávání příkazů rozděluje na dvě části. Na sekvenční část jako v jiných jazycích a na paralelní část. Uvnitř jednotlivých procesů běží příkazy sekvenčně, ale přiřazení jednotlivých signálů probíhá v jeden okamžik – tedy paralelně. Tento způsob programování „simuluje„ překlápění celého obvodu na jednotlivé náběžné hrany hodin. Vlastní specifickou část poté tvoři jazyk VHDL pro generování testbenchů, který nemá u procesů uveden citlivostní seznam, a tudíž se musí běh programu řídit jednotlivými *wait* stavy.

Tato práce byla velkým přínosem v rozšíření znalostí okolo tvorby programovatelných obvodů ve VHDL.

Myslím si také, že se mi povedl můj vlastní cíl napsat první českou literaturu udávající ucelený pohled na standard PCI Express a že se mi povedlo splnit zadání práce.

V této bakalářské práci je možná dobrá návaznost na práci diplomovou, ve které by mohly být realizované a odsimulované všechny vrstvy PCI Express a realizován kompletní řadič PCI Express.

5 Literatura

[1] *PCI Express Base Specification*, Revision 1.0a, PCI-SIG 2003

[2] *PCI Express Core – Reference Manual*, version 1.4.1, PLD Applications 2005

[3] DAVID PELERIN *VHDL Made easy* Prentica Hall PTR 1997

[4] TRACY V. WILSON. *How PCI Express Works.* URL < http://computer.howstuffworks.com/pci-express.htm>

[5] JON STOKES *PCI Express: An Overview* URL < http://arstechnica.com/articles/paedia/hardware/pcie.ars >

[6] AJAY V. BHAT. *Creating a PCI Express Interconnect.* Technology and Research Labs, Intel Corporation.

[7] DOC. ING. JAROMÍR KOLOUCH, CSC. *Programovatelné logické obvody a návrh jejich aplikací v jazycích ABEL a VHDL. Počítačové cvičení.* Vydavatelství Vysoké učení technické v Brně 2002.

[8] DOC. ING. JAROMÍR KOLOUCH, CSC. *Programovatelné logické obvody – počítačové cvičení.* Vydavatelství Vysoké učení technické v Brně 2001.

[9] ZDENĚK BARTOŇ, JAN DROBEK, JAROMÍR KOLOUCH, JAN KOVALSKÝ, JIŘÍ MITRYCH, VLADISLAV MUSIL, KAREL VLČEK. *Návrh digitálních integrovaných obvodů. Jazyk VHDL.* Cvičení. Vydavatelství Vysoké učení technické v Brně 2000.

[10] ING. PETR ORNST. *Modul pro snímání dat z inkrementálních senzorů pro PCI Express – Diplomová práce.* Katedra měření ČVUT FEL v Praze 2005.

[11] EASICS: *Generator of synthesizable CRC functions* [online]. [cit. 2005-11-10]. URL: <http://www.easics.com/webtools/crctool>

[12] M. BEČVÁŘ, M. DANĚK. *Přednášky z předmětu Praktika z návrhu číslicových obvodů X36PNO.* Katedra počítačů © 2006, 2007 M. Bečvář, M. Daněk, ČVUT-FEL.

SEZNAM OBRÁZKU

SEZNAM TABULEK