České vysoké učení technické v Praze
Fakulta elektrotechnická

Diplomová práce

Zpracování protokolu „Packet over Sonet“ na FPGA

Michal Trs

Vedoucí práce: Ing. Tomáš Marek

Studijní program: Elektrotechnika a informatika magisterský

Obor: Informatika a výpočetní technika

leden 2008

Poděkování

Rád bych poděkoval všem lidem, kteří mě podporovali při studiu na CVUT FEL. Zvláštní poděkování patří ing. Tomáši Markovi za přivedení k projektu Liberouter a vedení této práce.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Jinočanech dne 16.1. 2008 .....

Abstract

The aim of this thesis is analysis, design proposal and implementation of Packet over SONET (PoS) protocol in FPGA. PoS (RFC 2615) is used for data transfers in SONET/SDH telecommunication networks. In this thesis, VHDL language is used for hardware description and simulation.

Abstrakt

Práce se zabývá analýzou, návrhem řešení a implementací protokolu Packet over SONET (PoS) v obvodu FPGA. PoS (RFC 2615) je používán v telekomunikačních sítích SONET/SDH k přenosu dat. K popisu hardwarové implementace a simulaci je v této práci použit jazyk VHDL.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Práce je vyvíjena v rámci projektu Liberouter, který je jednou z aktivit společnosti Cesnet. Tato společnost je provozovatelem akademické sítě a zároveň řešitelem výzkumného záměru „Vysokorychlostní optické sítě“ financovaného z fondů Evropské unie. Cesnet se skládá z několika aktivit, kde nejrozsáhlejší je „programovatelný hardware“, která je známá pod názvem Liberouter. Původní cíl projektu bylo vytvořit hardwarovou platformu a na ní implementovat router. Po několika letech byl projekt úspěšně dotažen do konce. Během vývoje routeru bylo vyvinuto mnoho jednotek, které byly použity v dalších projektech. V současnosti nejúspěšnější projekt je FlowMon, což je pasivní sonda sledující datové toky na síti. Dále za zmínku stojí projekty IDS (systém vyhledávající vzory v paketech) a SCAMPI (sledování jednotlivých paketů na síti).

Doposud vyvinuté projekty jsou určeny pro sítě Ethernet (1GB/s, 10GB/s). Ethernet je definován standardem IEEE 802.3 a byl původně určen pro lokální sítě. Je nezávislý na fyzickém médiu. Může být použit koaxiální kabel, kroucená dvojlinka, nebo optická vlákna. S rozvojem síťových technologií se uplatňuje na páteřních linkách, díky nižším pořizovacím nákladům a srovnatelné přenosové rychlosti.

Zatím co Ethernet je synonymem pro přenos dat v počítačových sítích, v oblasti telekomunikací je dominantní technologií SONET/SDH pro přenos digitalizovaných telefonních hovorů. 95% světových telekomunikačních operátorů používá právě tuto technologii a proto je komerčně velice úspěšná. Přenosové rychlosti jsou srovnatelné s Ethernetem, avšak jako médium je použito převážně optické vlákno. S rozvojem Internetu a mobilní komunikace však přestal být zájem o využívání pevných linek pro přenos hlasu. Telefonní operátoři proto hledali způsob, jak použít stávající infrastrukturu k přenosu dat. Vznikly technologie ATM, FrameRelay a Packet over SONET (PoS). ATM se používá spíše pro připojení koncových zákazníků, FrameRelay je moderní technologie poskytující virtuální trvalé okruhy. PoS je pro svoji efektivitu používán na páteřních sítích.

Cílem této práce je navrhnout vstupní/výstupní (v/v) jednotky pro technologii Packet over SONET (PoS). Během specifikace byl kladen důraz na co nejvyšší kompatibilitu se stávajícími v/v jednotkami pro technologii Ethernet vyvinutých v rámci projektu Liberouter. Tím bude umožněno využití současných aplikací vyvinutých v rámci tohoto projektu v sítích SONET.

Diplomová práce je psána česky. Přejaté obrázky z anglické literatury však nejsou překládány a vždy je u nich uveden zdroj. V textu se také vyskytují anglické termíny, pro které neexistuje český ekvivalent.

2 Popis problému, specifikace cíle

Cílem této diplomové práce je analyzovat, navrhnout a realizovat obousměrný převod protokolu Packet over SONET na protokol FrameLink. V této kapitole je popsána technologie SONET/SDH a vysvětlena její architektura. Následuje rozbor technologie Packet over SONET (PoS) používané pro přenos paketů síťové vrstvy. Na závěr je uveden stručný popis implementační platformy a specifikace protokolu FrameLink.

2.1 SONET/SDH

SONET (Synchronous Optical NETwork) a SDH (Synchronous Digital Hierarchy) jsou technologie pro synchronní přenos digitalizovaných telefonních hovorů po optických vláknech. SONET je používán v Severní Americe, technologie SDH ve zbytku světa. Základní rychlost technologie SONET je 51,84 Mb/s a je označována jako OC-1.
SDH bylo specifikováno o několik let později normou ITU-T G.707 [14]. V té době byla běžná rychlost 155,52 Mb/s (SONET OC-3). Proto je základní rychlost SDH 155,52 Mb/s, která je označována STM-1 (Synchronous Transport Module).
Ostatní rozdíly mezi oběma technologiemi jsou pouze formální (jiné termíny v definicích apod.), což zaručuje kompatibilitu a možnost propojení obou technologií. SONET/SDH podporuje celá řada výrobců (Alcatel, Cisco, Fujitsu, Lucent, Marconi, Nortel a dalších) a neustále se vyvíjí. Přenosové rychlosti se stále zvyšují. V dnešní době je široce rozšířeno OC-48 (2,5 Gb/s) a začíná se používat OC-192 (10 Gb/s). Podporované rychlosti spolu s označením v dané technologii jsou shrnuty v tabulce 2.1.

Přenosová rychlost	SONET	SDH
52 Mb/s	OC-1	(STM-0)
156 Mb/s	OC-3	STM-1
467 Mb/s	OC-9	-
622 Mb/s	OC-12	STM-4
933 Mb/s	OC-18	-
1.2 Gb/s	OC-24	-
1.9 Gb/s	OC-36	-
2.5 Gb/s	OC-48	STM-16
5 Gb/s	OC-96	-
10 Gb/s	OC-192	STM-64
40 Gbps	OC-768	STM-256
160 Gbps	OC-3072	-

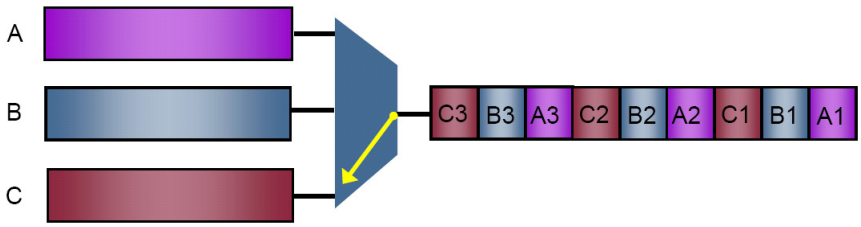
Tabulka 2.1: Podporované rychlosti SONET/SDH

V diplomové práci je převzata terminologie používaná u sítě SONET, která je rozšířenější. Většina obrázků je však převzata z [14], což může čtenáře mást. Proto jsou v textu (při prvním výskytu) uvedeny vždy oba názvy (SDH a SONET).

2.1.1 Architektura

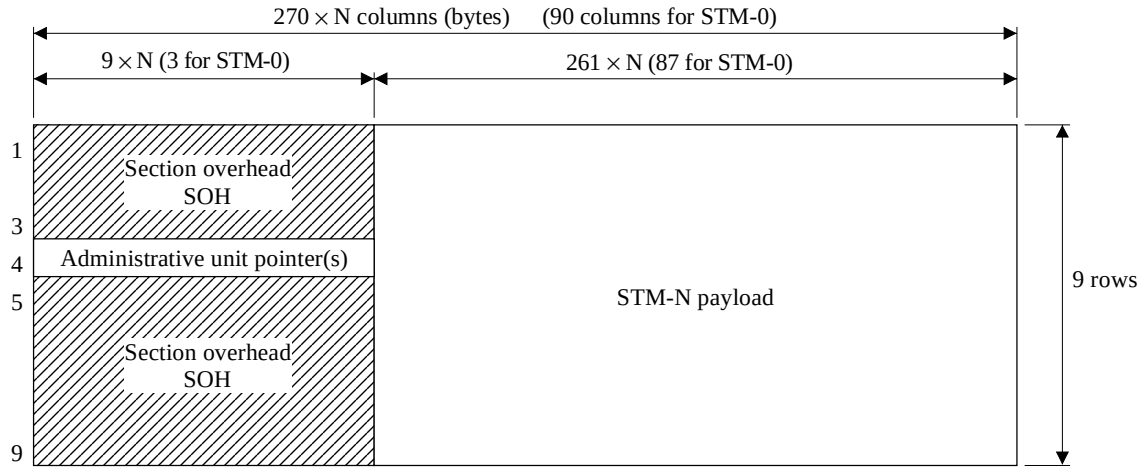
SONET/SDH představuje fyzickou vrstvu modelu OSI. Používá časový multiplex, kdy jsou skládány pomalejší datové toky do rychlejších. To je zajištěno Round-robin přepínáním N pomalejších toků v pravidelných intervalech do toku N-krát rychlejšího. Přepínáno je po 125μs, což je časový interval pro přenos jednoho rámce (nezáleží na použité rychlosti). Tato skuteč-

nost je symbolicky znázorněna na obrázku 2.1. Datový tok má konstantní charakter na rozdíl od Ethernetu, kde dochází ke krátkodobým přetížením. Základní jednotkou je byte, v [14] též označovaný jako oktet.



Obrázek 2.1: Round-robin přepínání toků (zdroj: Xilinx)

Rámec, který je přenášen každých 125μs, se skládá z hlavičky (SDH označení: SOH - Section Overhead, SONET označení: TOH - transport overhead) a dat (SDH označení: STM-N payload, SONET označení: SPE - synchronous payload envelope). Na obrázku 2.2 je znázorněn obecný rámec, kde za N dosadíme příslušné číslo určující rychlost STM (OC/3) a dostaneme velikost konkrétního rámce.

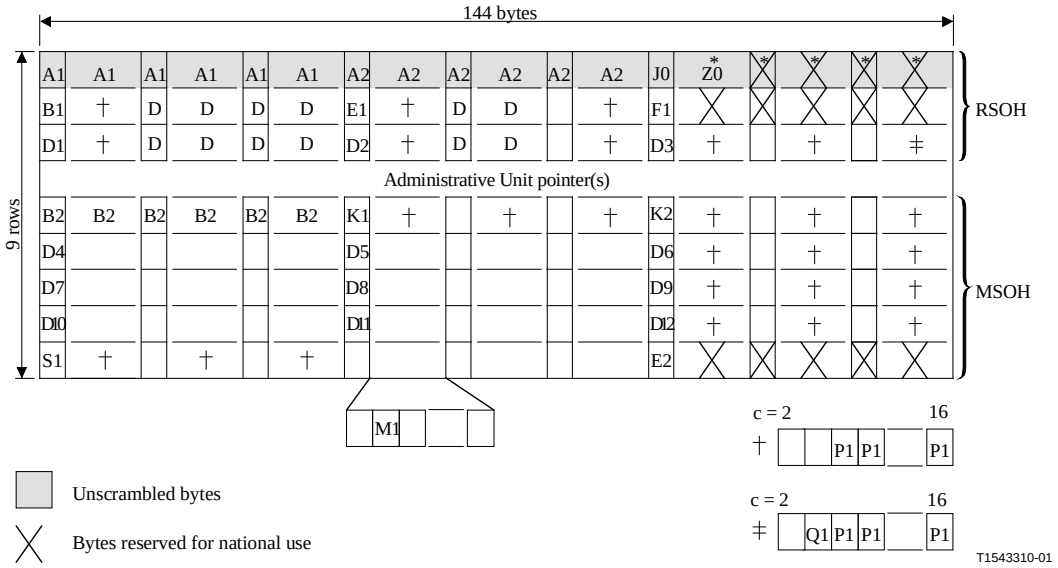


Obrázek 2.2: Struktura obecného SONET/SDH rámce (zdroj: ITU-T)

Rámec je zobrazen jako 2D struktura, protože nakreslit jej jako posloupnost bytů, nevešel by se na řádek. Tato 2D struktura je přenášena po řádcích zleva doprava od shora dolů, tím dochází k prokládání hlavičky a dat. To je podstatný rozdíl oproti Ethernetu, kde je nejprve poslána hlavička a poté data.

2.1.2 Hlavička (TOH)

Na obrázku 2.3 je již konkrétní TOH pro OC-48. Můžeme jej chápat jako 48x TOH základní rychlosti proložených byte po bytu, nebo jako jeden velký TOH s jiným významem bytů. Právě druhá možnost je vyžadována u PoS. Zde je rozdíl v terminologii mezi SONET a SDH. U SONETu se celá hlavička nazývá TOH. První 3 řádky se jmenují Section overhead (SOH), následuje jeden řádek ukazatel na SPE (data) a za ním 5 řádků Line overhead (LOH). U SDH se hlavička označuje Section overhead (SOH), první 3 řádky Regenerator Section Overhead (RSOH), pak následuje ukazatel na SPE a za ním je 5 řádků označovaných jako Multiplex



Obrázek 2.3: SONET OC-48 TOH / SDH STM-16 SOH (zdroj: ITU-T)

Section Overhead (MSOH). Např. v [23] jsou uvedeny veškeré rozdíly mezi SONET a SDH. V tabulce 2.2 je seznam a popis jednotlivých položek (1 nebo více bytů) hlavičky.

označení octetu	anglický název	význam	hodnota
A1, A2	Framing	začátek rámce	0xF6, 0x28
J0	Regenerator Section Trace	ověření spojení	CRC-7 / 0x01
Z0	Spare	nepoužit	
B1	BIP-8	bitová parita pro sekci	BIP-8
E1, E2	Orderwire	pořadí hlasových kanálů	
F1	User channel	rezervováno pro ISP	
D1-D3	DCCR	komunikační kanál 192 kb/s	
B2	BIP-24xN	bitová parita pro kanál	BIP-Nx24
K1, K2 (b1-b5)	APS channel	ochrana přepínání sekcí	
D4-D12	DCCM	komunikační kanál 576 kb/s	
S1 (b5-b8)	Synchronization status	synchronizační zpráva	
M0, M1	MS-REI	detekce vzdálené chyby	

Tabulka 2.2: Význam bytů v SONET/SDH hlavičce

2.1.3 Data (SPE)

V SPE mohou být přenášena téměř libovolná data. Pro zařízení na síti je důležité, aby mezi nimi dokázala jednoduše rozlišit. To je zajištěno bytem C2 v Path Overhead (POH), který je součástí SPE. V tabulce 2.3 jsou uvedeny hodnoty bytu C2 a jejich význam.
V následujícím výčtu krátce popíšu jednotlivé typy přenášených dat.

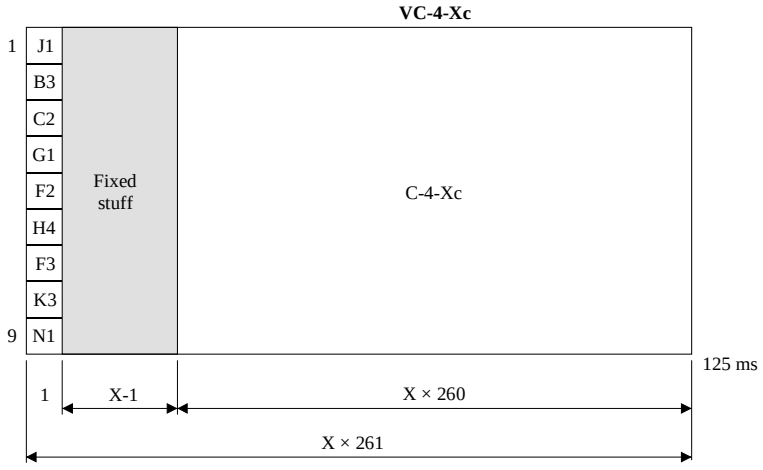
- SONET umožňuje přenos pomalých datových (hlasových) toků v takzvaných virtuálních kontejnerech (Virtual Tributaries). VT mohou být různě velké, podle potřené šířky

Hodnota bytu C2	Data v SPE
0x02	virtuální kontejner (VT)
0x04	DS-3
0x13	ATM
0x16	PoS (HDLC)
0xCF	PoS (PPP)

Tabulka 2.3: Typ přenášených dat

pásma. Každý VT má svůj POH ve kterém je opět byte C2. Existují 2 způsoby mapování virtuálních kontejnerů: fixní a plovoucí. Více informací najdete v [14] str. 118.

- DS-3 představuje 672x 64 kb/s ISDN linek, které dávají výsledný datový tok 44,736 Mb/s. Pozorný čtenář si jistě všiml této nerovnosti: 672x 64 kb/s < 44,736 Mb/s. Je způsobena rozcházející se frekvencí hodin v koncových zařízení. Tento trafik je umístěn do SPE-1 rámců, ale není umožněn přístup k pomalejším DS-2 až DS-0 tokům.
- ATM je založeno na buňkách fixní velikosti 53B, které nejsou kompatibilní s Ethernet rámci. ATM buňky mohou být efektivně vkládány do SONET rámce, ale při vyšších rychlostech je overhead ATM buněk značně velký (48B data a 5B overhead).
- Packet Over SONET poskytuje mechanismy pro přenos paketů přímo uvnitř SONET SPE prostřednictvím HDLC nebo PPP rámců. PPP/HDLC rámce jsou vkládány do spojených virtuálních kontejnerů označovaných VC-4-Xc. Kontejner obsahuje POH (path overhead) a dále již samotná data. Tento kontejner (obr. 2.4) přímo odpovídá velikosti SPE do kterého je vložen.



Obrázek 2.4: Virtuální kontejner VC-4-Xc (zdroj: ITU-T)

2.2 Packet Over SONET (PoS)

V současné době je PoS [15] specifikován v RFC 2615 [23] (PPP over SONET), který nahradil RFC 1619 [20]. Point-to-Point Protocol (PPP) představuje standardní metodu pro přenos datagramů různých typů po dvoubodových spojích.

RFC 2515 [23] definuje tyto požadavky:

- Velikost virtuálního kontejneru VC stejná jako SPE (High-order Containment)
- Zarovnání po bytech (Octet Aligment)
- Kódování dat (Payload scrambling)

Při vysílání je postup následující: IP paket → PPP rámec → výpočet kontrolního součtu → nahrazení bytů 7D a 7E → zakódování → vložení do SONET/SDH SPE (Synchronous payload envelope).

Na příjmu je postup obdobný: Nalezení začátku SONET/SDH rámce → Dekódování → nahrazení dvojice bytů 7D XX, kde XX představuje libovolnou hodnotu → ověření kontrolního součtu → PPP rámec → IP paket

Nyní se budu podrobně zabývat všemi kroky, které jsou nutné pro vložení libovolného paketu síťové vrstvy (IP, IPX, ...) do SONET/SDH rámce a jeho zpětnému vyjmutí.

2.2.1 Vložení IP (paketu síťové vrstvy) do PPP

PPP je stavový protokol definovaný v [21]. Skládá se z těchto komponent: metoda pro zapouzdření paketů síťové vrstvy, protokol LCP (Link control protocol) a NCP (Network Control Protocol).

2.2.1.1 Formát PPP rámce

PPP rámec se skládá z pole „protocol“, „information“ a případného zarovnání viz obrázek 2.5. Pro rozlišení typu dat je určeno 16b číslo protokolu. Přenášené informace mohou mít velikost 0-64 kB. Zarovnání je použito, pokud je vyšším protokolem vyžadováno přenést přesnou, předem domluvenou MRU (Maximum-Receive-Unit) velikost dat.

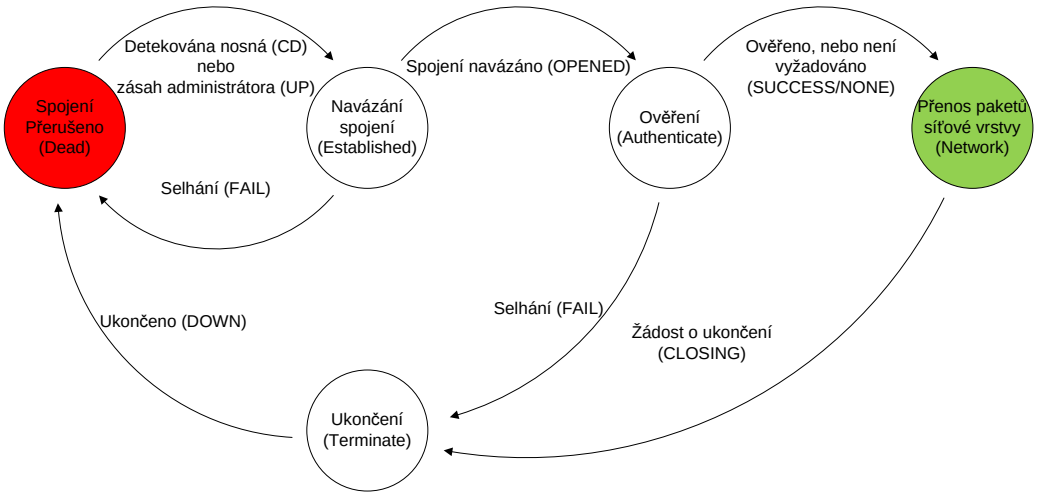
Obrázek 2.5: PPP rámec (zdroj Cisco)

2.2.1.2 Navázání spojení

Před tím, než je možné přenášet data (pakety síťové vrstvy), je nutné navázat spojení. Jednotlivé fáze (stavy) navazování jsou patrné na diagramu 2.6. Přechod do další fáze je vyvolán přijetím LCP (Link Control Protocol) paketu, nebo zásahem administrátora.

Nyní krátce popíšu jednotlivé fáze protokolu.

- Spojení přerušeno (Dead)
  Toto je počáteční a zároveň koncový stav automatu. Do stavu Establish se přejde buď po detekování nosné (CD - carrier detection) na úrovni vrstvy L1, nebo zásahem administrátora ze SW.
- Navázání spojení (Established)
  K navázání spojení je použit protokol LCP (více v RFC 1661 [21], strana 26). Po odeslání a následném přijetí Configure-Ack paketu automat přejde do další fáze.



Obrázek 2.6: Stavový diagram protokolu PPP

- Ověření (Authenticate)
  Ověření obou spojů se používá hlavně při bezdrátové komunikaci, nebo obecně tam kde nemůžeme jinak zaručit pravost druhé strany. To ovšem není případ páteřních spojů. Více informací najdete v RFC 1661 v kapitole 3.5.

- Přenos paketů síťové vrstvy (Network)
  V tomto stavu dochází ke konfiguraci protokolu síťové vrstvy a vlastnímu přenosu dat. Ke konfiguraci je použit protokol NCP.

- Ukončení (Terminate)
  K ukončení přenosu může dojít z mnoha důvodů (ztráta nosiče, chyba při ověření, nízká kvalita spoje, time-out, zásah administrátora). Opět je použit protokol LCP, konkrétně typ paketu Terminate-Ack.

2.2.2 Zapouzdření PPP do HDLC rámce

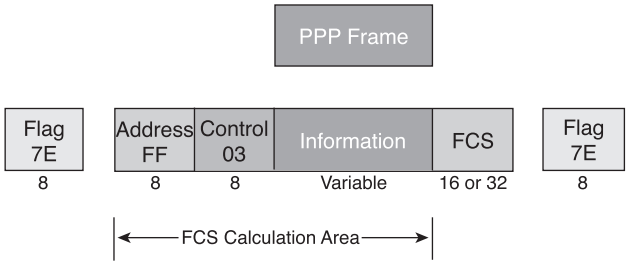
HDLC protokol je použit k oddělení jednotlivých PPP rámců od sebe [22]. Dále také zajišťuje ochranu proti chybám na linkové vrstvě, kterou PPP neposkytuje. V následujících třech podkapitolách popíšu formát HDLC rámců, výpočet kontrolního součtu a náhradu bytů 7E a 7D.

2.2.2.1 Formát rámce

Na obrázku 2.7 je formát HDLC rámce s použitými hodnotami při vložení PPP rámce. Flagy mají hodnotu 0x7E a udávají začátek a konec rámce. Vysílací zařízení generuje tyto značky jako časovou výplň, pokud nejsou připravena žádná data k vysílání. Pole „address“ je obvykle nastaveno na 0xFF, protože každý rámec je typu broadcast. Pole „control“ má hodnotu 0x03 a značí PPP rámec. Za vloženými daty (PPP rámcem) následuje 16b nebo 32b kontrolní součet (FCS). Pro rychlosti OC-12 a vyšší je defaultně použit 32b kontrolní součet.

2.2.2.2 Výpočet kontrolního součtu

Pro výpočet kontrolního součtu je použito klasické CRC-32. Podle specifikace je možné také použít CRC-16, ale to není v případě rychlých sítí doporučeno. Kontrolní součet se počítá před



Obrázek 2.7: HDLC rámce (zdroj Cisco)

náhradou bytů 0x7E a 0x7D z polí address, control a PPP rámce (information).

2.2.2.3 Náhrada bytů 7E a 7D

Jelikož je hodnota 0x7E použita k oddělení HDLC rámců, její výskyt uprostřed přenášených dat by způsobil předčasný konec rámce. Tato skutečnost je řešena náhradou bytů podle tabulky 2.4. Jednoduše řečeno je přidán prefix 0x7D a původní hodnota je xorovaná 0x20.

byte v datech	Nahrazen za
0x7E	0x7D 0x5E
0x7D	0x7D 0x5D

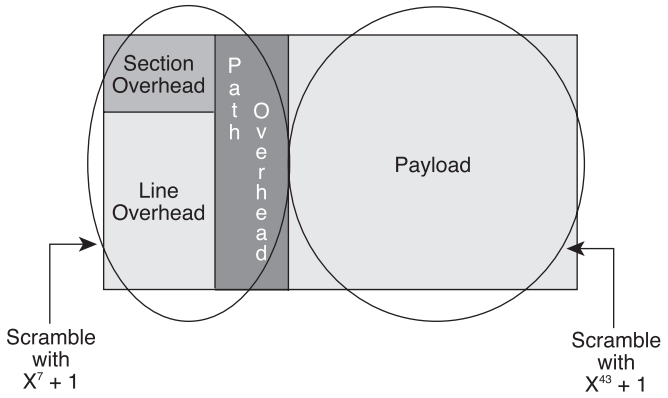
Tabulka 2.4: Náhrada bytů v HDLC

2.2.3 Kódování dat

SONET defaultně používá 7b polynom pro kódování hlasu, ten ovšem není dostatečný pro data. Data mohou obsahovat dlouhé sekvence 0 nebo 1 a proto by mohlo docházet ke ztrátám synchronizace. Z tohoto důvodu je přidáno další kódování, které je použito pouze pro data (payload).
SONET hlavička je zakódována polynomem $1 + x^6 + x^7$ s výjimkou TOH položek A1/A2 a J0 (viz. tabulka 2.2).
Pro data (payload) je použit delší polynom $x^{43} + 1$, který je samo-synchronizační. Nemusíme před začátkem dekódování znát výchozí stav, ale prvních 43 přenesených bytů bude neplatných. Byte C2 v POH udává zda je kódování pro payload zapnuto. Pokud ano má hodnotu 0x16 nebo 0xCF. Na následujícím obrázku 2.8 je situace dobře patrná.
Tento proud dat je již přímo vložen do SONET SPE.

2.3 Implementační platforma

Rodina karet Combo [2] je vyvíjena aktivitou Liberouter již několik let. Název Combo značí, že se jedná o 2 karty, které se přes konektory spojí v jeden celek. Mateřská karta se přes konektory spojuje s tzv. interfacovou kartou (IFC). IFC karta existuje v několika verzích, které se od sebe liší síťovým interfacem. Označení MTX značí konektory RJ-45, SFP a SFPRO jsou osazeny optickými konektory.
První generace karet Combo6 je postavená na FPGA Xilinx Virtex 2. Je nasazena v živých sítích společně s aplikací FlowMon na mnoha univerzitách. Bohužel nedosahuje plného gigabitu a proto je v současné době nahrazena kartou Combo6X, která je označována za generaci 1.5. Ta



Obrázek 2.8: Kódování SONET rámce (zdroj Cisco)

opravuje několik chyb v kartách Combo6 a díky čipům Xilinx Virtex 2 Pro posouvá výkonnostní hranici ke zpracování 4 gigabitových toků současně. Součástí této generace jsou i nové IFC karty, ke kterým patří i Combo-4SFPRO/OC48.
V této době je vyvíjena zcela nová generace karet pod označením Combo v2. Bude obsahovat FPGA čip Xilinx Virtex 5 a moderní PCI Express sběrnici. Spolu s ní budou vyvinuty IFC karty pro zpracování až čtyř 10 Gb/s toků, případně 40 Gb/s toku. Dále budou dostupné karty pro SONET OC-192 a případně OC-768.

2.3.1 Combo6x

Mateřská karta, označovaná Combo6X, je vybavena interfacem PCI pro zasunutí do PC. Skládá se z velkého FPGA (aplikačního) a malého FPGA v roli PCI bridge. Dále obsahuje konektor pro vložení dynamické paměti, několik statických pamětí a paměť CAM. Přes PCI sběrnici je nahráván firmware do aplikačního FPGA a FPGA na interfacové kartě. Pro více informací o této kartě odkazuji na web projektu Liberouter [2].

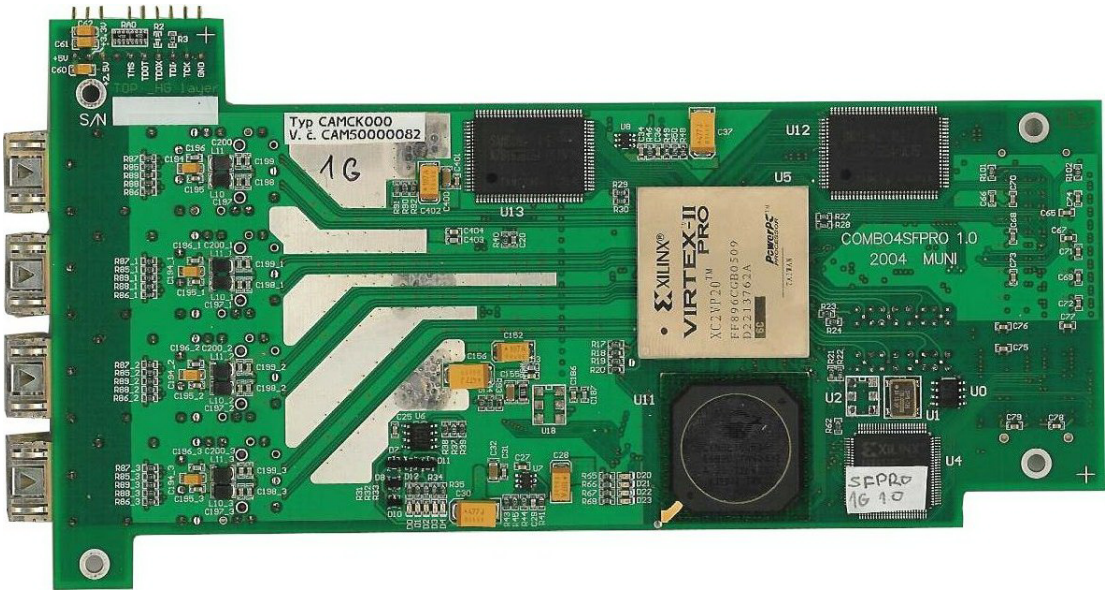
2.3.2 Combo-4SFPRO/OC48

Tato karta je na obrázku 2.9 a 2.10.

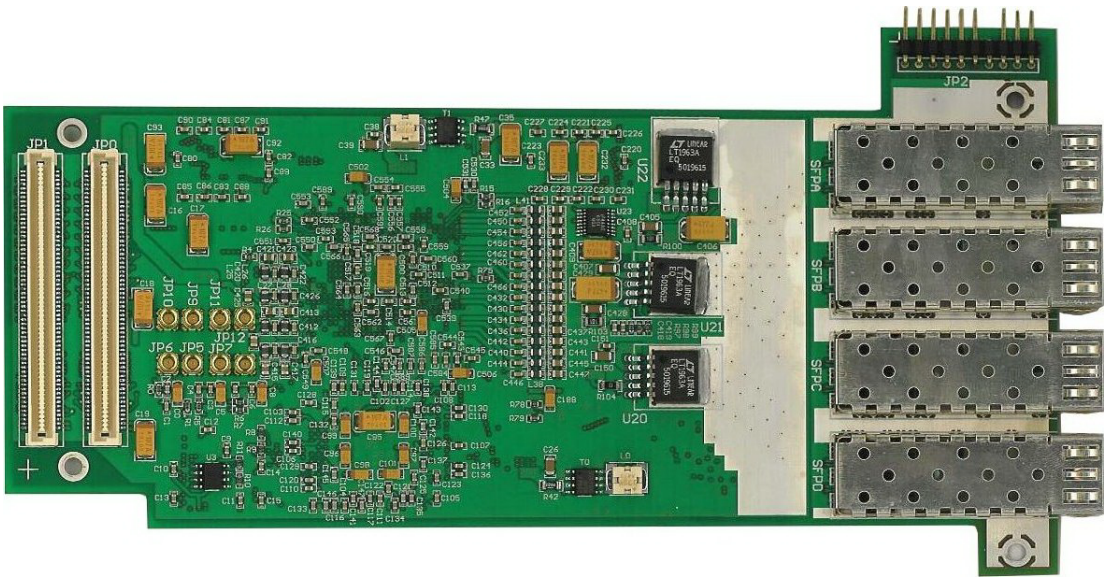
Karta obsahuje tyto komponenty:

- Virtex II - XC2VP20
- 2xSSRAM - 2MB 512K36
- 1xEEPROM - 93S66
- 4xXFP klece
- konektor pro připojení k mateřské kartě
- hodinový krystal 155,52 MHz, který je vhodný pro OC-48 (2.488 Gb / 16 = 155.52 MHz)

Použité FPGA je Xilinx Virtex 2 PRO-X (XC2VPX20) od firmy Xilinx. Je vyráběno 150nm technologií a jedná se o velice zdařilou architekturu (narozdíl od Virtex 4). Toto FPGA obsahuje 8 vylepšených rychlých sériových linek Rocket IO-X, které umožňují přenos až 10.3125 Gb/s. S rezervou to stačí pro SONET OC-48 (tj. 2,5 Gb/s).



Obrázek 2.9: COMBO-4SFPRO/OC48 karta (pohled shora)



Obrázek 2.10: COMBO-4SFPRO/OC48 karta (pohled zdola)

FPGA Xilinx XC2VPX20 se skládá z 22032 logických buněk a dále několika specializovaných bloků. Kromě již zmiňovaných Rocket IO-X rozhraní je to 88 BlockRAM pamětí (celkem 1584 kb), 88 18x18 rychlých násobiček, 8 DCM (Digital clock managment) a 2 IBM PowerPC 405 procesory.
Rocket IO-X přímo podporuje tyto komunikační protokoly: SONET OC-48, SONET OC-192, PCI Express, Infiniband, XAUI (10-Gigabit Ethernet), XAUI (10-Gigabit Fibre Channel, Aurora (Xilinx protokol).

2.4 FrameLink

FrameLink (FL) je protokol pro přenos dat ve formě paketů, který byl vyvinut a je používán v projektu Liberouter. FL je inspirován protokolem LocalLink (LL) [3] od firmy Xilinx, ale obsahuje několik změn.

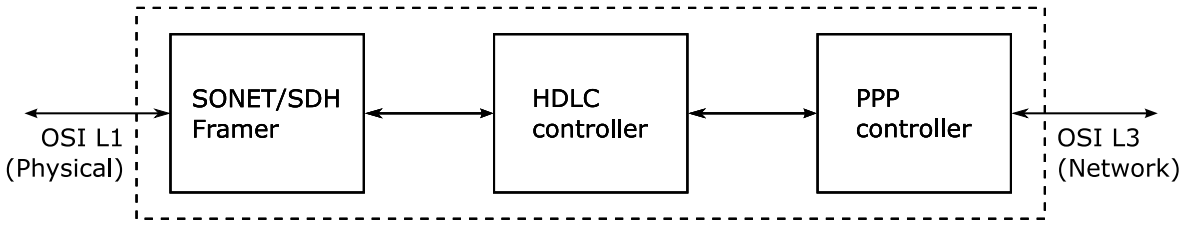
Mezi hlavní rozdíly proti LL patří:

- data jsou vždy zarovnaná, v jednom slově nemůžou být dvě části paketu
- SOP_N a EOP_N ohraničuji každou část paketu
- není explicitní informace ve které části paketu jsme, je to dáno pořadím.
- používá se uspořádání bytů little endian

Ze změn oproti LL je patrné, že FL je jeho zjednodušenou verzí. To přináší jednodušší implementaci komponent, ale zároveň snižuje propustnost. Mezi jednotlivými pakety vznikají mezery, což je nepříjemné hlavně u velkých datových šířek. Stejně jako LL podporuje FL flow control, tj. datový přenos může pozastavit jak vysílací, tak přijímající strana.
Pro více informací viz Příloha A.

3 Analýza dostupných řešení

V této kapitole je uveden přehled dostupných řešení na trhu, které se zabývají zpracováním protokolu Packet over SONET. U každého řešení je rozbor jejich výhod a omezení.
FrameLink a RocketIO jsou proprietální protokoly, a proto je jasné, že množina dostupných, bez úprav použitelných, řešení bude velice malá, pravděpodobně prázdná. Z RFC 2615 [23] (viz. kapitola 2.2) je dobře patrné jak postupovat při vkládání paketů síťové vrstvy do SONET/SDH rámce.



Obrázek 3.1: Ilustrační schéma jednotky pro zpracování Packet over SONET

Obrázek 3.1 znázorňuje jednotlivé bloky, na které se dá zpracování protokolu Packet over SONET rozebrat. Krátce je zde popíšu a v následujících podkapitolách budou již konkrétní řešení.

SONET/SDH Framer je koncové zařízení na SONET/SDH síti, které generuje/ověřuje TOH a vkládá/extrahuje data do/z SPE.

HDLC controller se stará o oddělení rámců vyšší vrstvy od sebe a detekuje chyby pomocí kontrolního součtu.

PPP controller přidává k paketům síťové vrstvy pole „protocol“, pomocí kterého je možné snadno rozlišit obsah PPP rámce. Dále se stará o navázání/ukončení spojení.

3.1 Dostupná řešení SONET/SDH Frameru

Univerzální SONET/SDH Framer, který by pracoval s rozhraním RocketIO se mi nepodařilo nalézt. Uvádím zde zajímavá (architekturou, vlastnostmi, použitím,...) řešení s jiným rozhraním a jednoduché komponenty od firmy Xilinx, ze kterých se dá Framer částečně poskládat.

3.1.1 Altera SONET/SDH Compiler

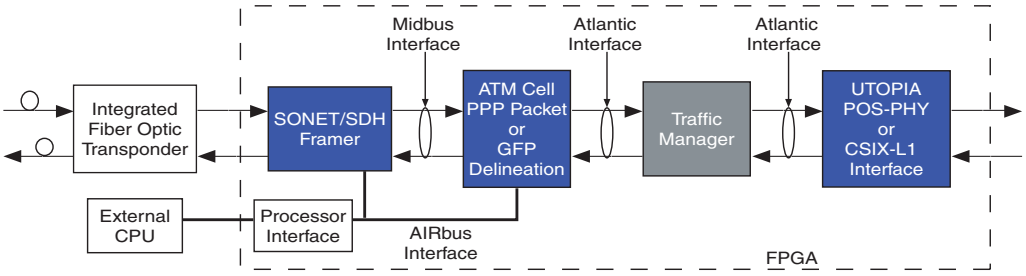
Tato kapitola popisuje SW nástroj SONET/SDH Compiler [5] od firmy Altera, který generuje SONET/SDH Framer. Na základě specifikace vlastností frameru, v nástroji IP Toolbench (Altera), je vygenerován zakódovaný netlist (trial verze), VHDL nebo Verilog soubor.
Tato implementace dokáže pracovat až na rychlosti OC-192 a podporuje plné zpracování TOH. Je určena pro platformu Altera Stratix (až OC-192) a APEX/APEX II (do OC-48). Ostatní platformy a výrobci nejsou z výkonnostních a licenčních důvodů podporovány.

Výčet vlastností:

- Podporované rychlosti OC: 1, 3, 12, 48 a 192
- Struktury SONET rámce: odpovídající rychlosti OC (concatenated), STS-1/AU-3 (channelized), STS-3c/AU-4 (channelized) viz. [14]

- Generuje a detekuje Byte A1/A2
- Generuje a sleduje Byte B1/B2/B3
- Ukončuje section (SOH), line (LOH) a path (POH) overhead
- Zahrnuje kódování/dekódování
- Generování a zpracování ukazatelů na data (Byte H1/H2)
- Plně duplexní
- Rozhraní Midbus (Altera) pro připojení do designu, datová šířka 8, 16, 32, 64 b (závisí na rychlosti OC a frekvenci designu)

Na obrázku 3.2 je doporučené zapojení této jednotky. Znázorňuje použití ATM, PPP, nebo GFP pro přenos dat po SONET/SDH síti. Aplikační rozhraní je UTOPIA POS-PHY nebo CSIX-L1.



Obrázek 3.2: Altera SONET/SDH Framer - doporučené zapojení (zdroj Altera)

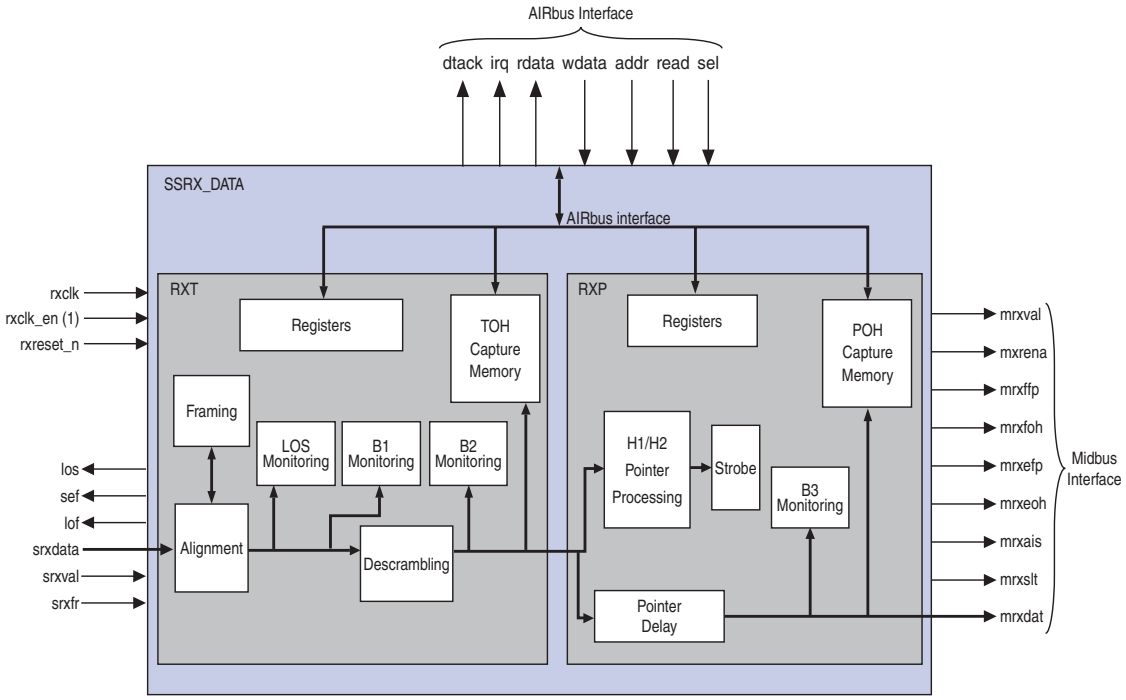
3.1.1.1 Popis funkce/architektury

SONET/SDH framer se skládá z 3 jednotek (vysílací část, přijímací část a řídicí procesor) pracujících v navzájem různých časových doménách. Na následujících obrázcích 3.3 a 3.4 je blokové schéma vysílací a přijímací části. Uvádím je zde proto, že je z nich velice dobře patrná funkcionalita při vysílání/příjmu dat po síti SONET.
Jak vysílací, tak přijímací část je dále rozdělena na 2 části zpracovávající zvlášť hlavičku TOH a POH.

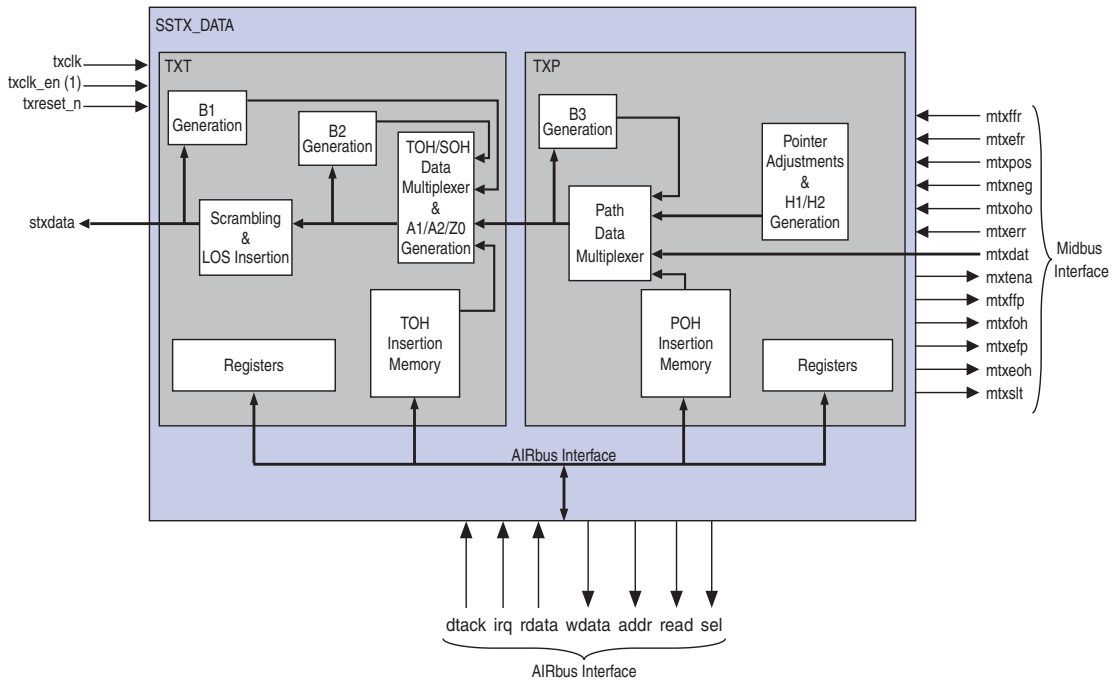
Přijímací část se skládá z jednotky RXT a RXP. RXT zajišťuje zpracování hlavičky TOH (SOH + LOH). Na vstupu jednotky je blok Aligment, který detekuje začátek rámce. Za ním je blok Descrambler (viz. kapitola 2.2.3) a bloky pro monitorování chyb B1, B2 a ztrátu nosné LOS. Dále obsahuje paměť TOH Capture Memory pro průběžné ukládání hlavičky.
RXP zpracovává POH a samotná data (SPE). Blok Pointer delay je FIFO paměť, kde je průběžně ukládáno SPE, dokud není zjištěn jeho začátek v bloku H1/H2 pointer processing. Dále obsahuje blok pro monitorování chyby B3 a paměť pro uložení POH.

Vysílací část se skládá z jednotky TXT a TXP. TXT generuje TOH, což znamená generování A1, A2, B1, B2, kódování (viz. kapitola 2.2.3) a přepínání mezi TOH a SPE. Dále obsahuje paměť, ze které je možné vkládat další položky do hlavičky.
Hlavní funkcí jednotky TXP je přepínání mezi jednotlivými SPE. Dále počítá položku B3, generuje ukazatele H1/H2 a obsahuje paměť, ve které jsou uloženy další POH položky.

AIRbus je sběrnice pro připojení externího procesoru. Midbus slouží pro připojení jednotky do designu.



Obrázek 3.3: Přijímací část Altera SONET/SDH Frameru (zdroj Altera)



Obrázek 3.4: Vysílací část Altera SONET/SDH Frameru (zdroj Altera)

3.1.1.2 Klady/zápory

Klady a zápory jsou shrnuty v tabulce 3.1

Klady	Zápory
Podpora všech SONET/SDH standardů	Určeno pouze pro FPGA f. Altera
Plné zpracování TOH informací	
Podporuje rychlost až OC-192	
Podrobný a přehledný manuál	

Tabulka 3.1: Klady a zápory SONET/SDH Framer firmy Altera

3.1.2 Xelic SONET/SDH Transport processor core (XCS48F)

XCS48F [24] poskytuje vkládání/sledování TOH, zarovnává příchozí SONET/SDH rámce, detekuje chyby a sleduje výkon.
Jednotka se skládá z nezávislé vysílací a přijímací části. Zahrnuje porty pro vkládání a zpracování TOH v externí jednotce. Podporovanou rychlostí je pouze OC-48. Rozhraní pro připojení do designu má datová šířku 32b při frekvenci 77,76MHz.
Ostatní vlastnosti jsou totožné s předchozí jednotkou.

3.1.2.1 Klady/zápory

Klady a zápory jsou shrnuty v tabulce 3.2

Klady	Zápory
Dodáváno ve formě VHDL kódů nebo EDIF	Pouze OC-48
Splňuje ITU-T G.707 a Telcordia GR-253-CORE	Fixní datové rozhraní 32b/77,76MHz
Plné zpracování TOH informací	Blackbox (velice stručný manuál)

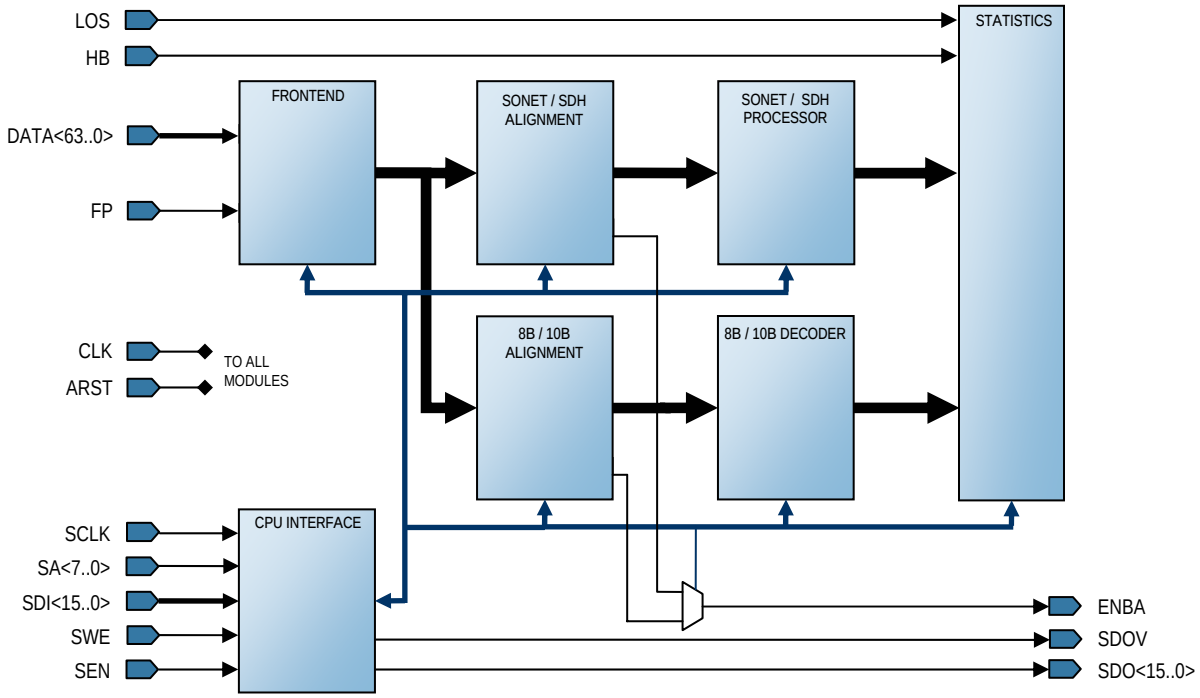
Tabulka 3.2: Klady a zápory Xelic XCS48F

3.1.3 Calyptech Performance monitor core

Pro zajímavost zde uvádím jednotky CORE-PM-2G5 [7] a CORE-PM-10GB [6] poskytující sledování výkonu pro technologie SONET/SDH, Fibre Channel a Gigabit Ethernet. Nejedná se sice o SONET/SDH Framer a nelze je tedy využít pro implementaci PoS, nicméně nynější projekty, aktivity Liberouter, jsou zaměřeny na síťovou bezpečnost, kde by jistě tyto jednotky našli uplatnění, a proto je zde uvádím.
Obě jednotky mají stejnou architekturu (obrázek 3.5), liší se pouze v podporovaných rychlostech.

Vlastnosti:

- Přenosová rychlost až 2.488 Gb/s (CORE-PM-10GB až 9,953 Gb/s)
- podporované standarty: OC-3, OC-12, OC48, OC192 (pouze CORE-PM-10GB), GbE, FC
- Detekce ztráty signálu (Loss Of Signal)



Obrázek 3.5: Sledovač výkonu CORE-PM f. Calyptech (zdroj Calyptech)

- Stav zarovnání rámce (Frame Alignment)
- Čítač chyb Bytů B1/B2
- Stav AIS, RDI, REI
- Ukládání M1 (pouze CORE-PM-2G5), K1, K2 a J0 Bytů
- dekódování 8B / 10B
- rozhraní pro CPU
- dodáváno ve formě VHDL kódů nebo EDIF

Nebudu zde uvádět klady a zápory, protože jednotky jsou určeny pro jinou aplikaci, než zpracování PoS.

3.1.4 Xilinx

Firma Xilinx neposkytuje kompletní řešení Frameru, ale nabízí dokumentované příklady, ze kterých se dá SONET/SDH Framer částečně poskládat.
Xilinx je v současnosti jedním z největších výrobců FPGA. Kromě vlastního „železa“ poskytuje vývojové nástroje ISE a EDK. Součásti těchto nástrojů je mimo jiné CORE generátor, který slouží k parametrizaci dodávaných „soft“ IP jader. Dále firma publikuje mnoho tzv. „Application Note“ pod označením xapp. Jedná se většinou o jednoduché jednotky na řešení daného problému. Jsou nabízeny se zdrojovými kódy a manuálem.

Pro Framer s fyzickým rozhraním RocketIO-X jsou použitelné tyto jednotky:

xapp649 Konverze datové šířky/frekvence
SONET Rate Conversion in Virtex-II Pro Devices

xapp651 Scrambler/descrambler
SONET and OTN Scramblers/Descramblers

xapp652 Detekce začátku rámce
Word Alignment and SONET/SDH Deframing

3.1.4.1 Konverze datové šířky/frekvence (xapp649)

Rychlé sériové rozhraní RocketIO-X uvnitř pracuje s datovou šířkou 20 bitů. Pokud je uvnitř použito kódování 8b/10b (Ethernet) venkovní rozhraní má obvyklou šířku 16 bitů. Pro aplikace kde není toto kódování použito (např. SONET/SDH) má venkovní rozhraní datovou šířku 20 bitů, což je značně nestandardní.
Komponenta Xilinx xapp649 [19] poskytuje řešení jak převést 20b na běžných 16b. Skládá se ze 2 částí. První částí je logika, která dělá vlastní konverzi datové šířky. Jedná se o 80b registr, do kterého je v 5 taktech uloženo 16b a na výstupní straně vyčteno 20b ve 4 taktech. Pro opačný směr je princip analogický. Druhou částí je časování. 16b data musí být přenášena s větší hodinovou frekvencí než 20b, aby byla zaručena funkčnost jednotky. Z datových šířek dostáváme poměr 5:4 pro hodinovou frekvenci. Frekvence odpovídající datové šířce je shrnuta v tabulce 3.3

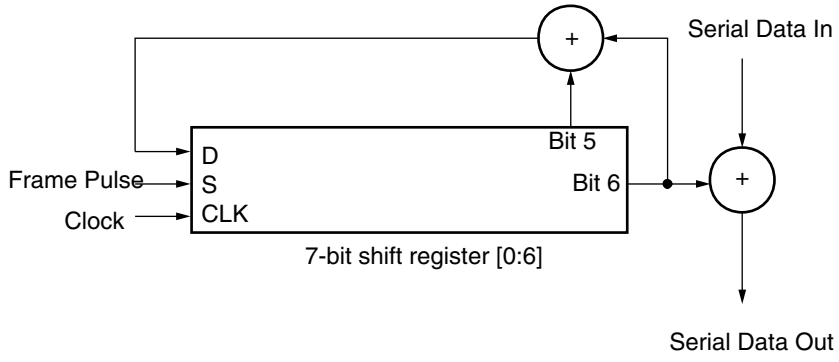
Datová šířka [b]	Hodinová frekvence [Mhz]
16	155,56
20	124,416

Tabulka 3.3: Hodinová frekvence x datová šířka

Je nutné zajistit, aby referenční hodiny pro RocketIO-X `REFCLK` byly přesně 0.8 násobek systémových hodin. Bohužel nejde použít DCM (Digital clock manager), které je uvnitř FPGA. Rozptyl periody nutný pro RocketIO-X musí být menší než 40 ps, což DCM nesplňuje. V dokumentu [19] jsou popsány 2 možnosti získání takto přesných hodin, ale obě dvě vyžadují externí součástky, které nejsou na kartě k dispozici.

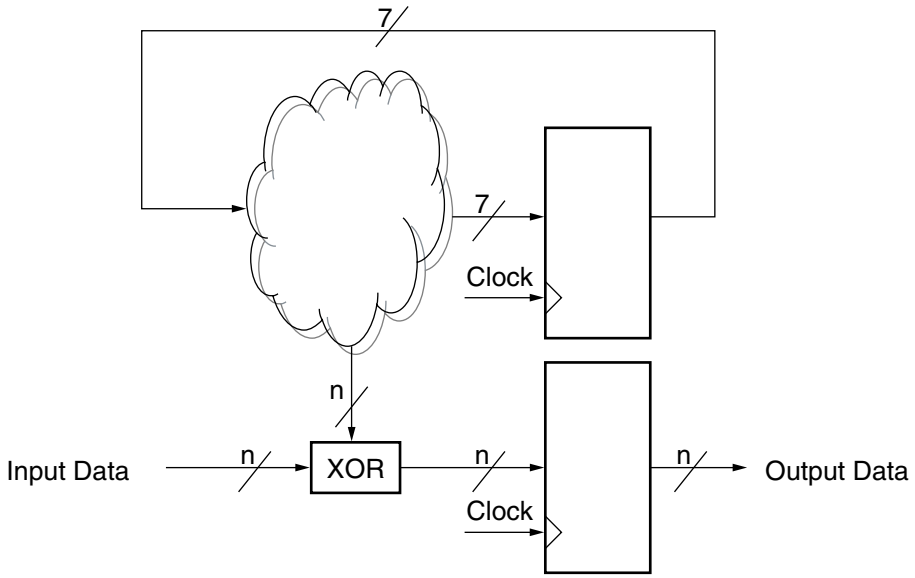
3.1.4.2 Scrambler/descrambler (xapp651)

SONET/SDH extrahuje hodiny z přijímaných sériových dat. Aby nedocházelo ke ztrátám synchronizace, musí být zaručeno dostatečné množství změn přijímaného signálu. To je zajištěno kódováním na vysílací a dekódováním na přijímající straně. Je použit ireducibilní polynom.



Obrázek 3.6: SONET/SDH scrambler/descrambler, 1 bit (zdroj Xilinx)

Pro SONET/SDH je použit 7b polynom $1 + x^6 + x^7$. Resetován je signálem INIT (na obr. pojmenovaný Frame pulse) do známého stavu „1111111“. Schematické znázornění situace je na obrázku 3.6, kde je vstup a výstup sériový. SONET OC-48 většinou zpracováváme paralelně po slovech (16b). Proto je nutné výpočet paralelizovat. Takovéto řešení je na obrázku 3.7. Toto řešení [17] pracuje dobře pro datové šířky 8, 16, 32 a 64 bitů. Pro datovou cestu 16b širokou, jednotka pracuje až do frekvence 360 MHz, zabírá pouze 24 Flip-flopů a 25 LUT.



Obrázek 3.7: SONET/SDH scrambler/descrambler, n bitů (zdroj Xilinx)

3.1.4.3 Detekce začátku rámce (xapp652)

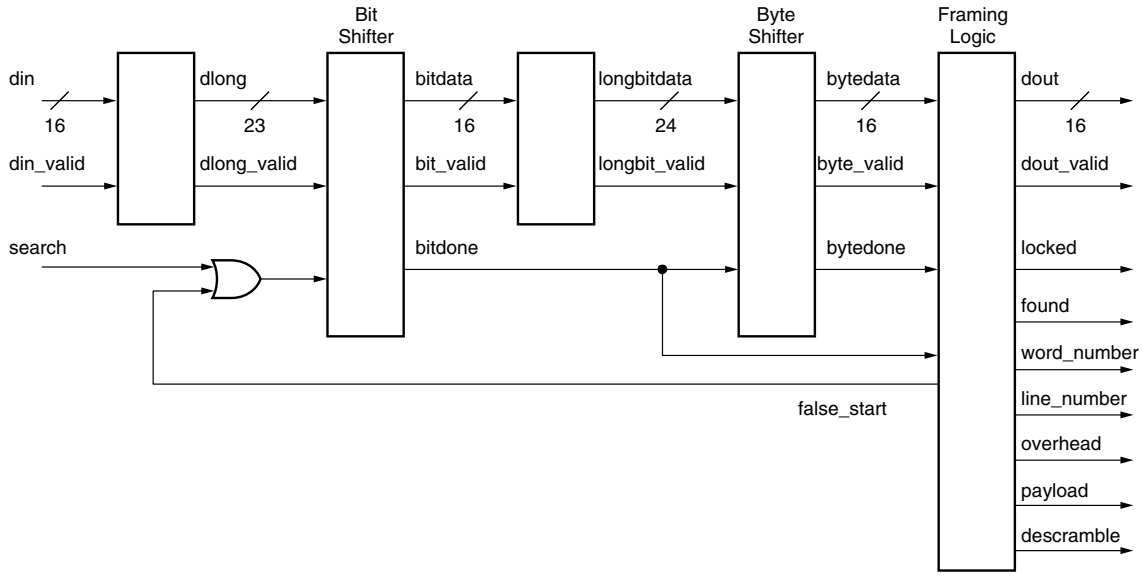
Po síti SONET/SDH jsou data přenášena sériově. Zpracování se však provádí paralelně, protože jsme limitováni frekvencí designu (max. 100ky MHz). Typicky SONET OC-48 je zpracováván po 16b při frekvenci 155 MHz. Aplikace potřebuje určit jakých 16b odpovídá původním 1b toku dat. Proto jsou v hlavičce SONET/SDH rámce byty A1 a A2. Jakmile je zjištěna tato posloupnost bytů, můžeme následně posunout slovo tak, aby odpovídalo hranicím bytu.

Vysvětlující příklad: Mějme datovou cestu 16b širokou. Nalezli jsme posloupnost bytů A1, A2 a pomocí ni určili začátek dat na bit [5] v aktuálním slově. To vyžaduje posunutí bitů [15:5] aktuálního slova na pozici [10:0] výstupního slova. Bity [15:11] výstupního slova jsou vzaty z předchozího slova z pozice [4:0].

Zarovnávací obvod [18] je poměrně snadný. Varianta pro 16b datovou cestu je zobrazena na obrázku 3.8. Pro následné snadnější zpracování jsou k dispozici 2 signály určující aktuální řádek (line_number [0 - 8]) a sloupec (word_number [0 - 2159]) rámce. Dále máme k dispozici signály, které určují, zda výstup odpovídá TOH (overhead) nebo SPE (payload). Poslední výstupní signál se jmenuje descramble a zapíná dekódování dat (popsáno v předchozí kapitole). Pro další podrobnosti ohledně funkce obvodu odkazuji na [18].

3.1.4.4 Core generátor RocketIO 8.2i

Core generátor pro RocketIO je dodáván bezplatně s ISE. Umožňuje vytvoření konfigurací pro SONET OC-48, SONET OC-192, PCI Express, Infiniband, XAUI (10-Gigabit Ethernet), XAUI (10-Gigabit Fibre Channel) a Aurora (Xilinx protocol).



Obrázek 3.8: Zarovnávací obvod, 16b data (zdroj Xilinx)

Tuto jednotku je při použití karty Combo 4SFPRO/OC-48 nutno zapojit vždy, protože SFPRO cartrige jsou připojeny pouze přes RocketIO-X k FPGA.
Obsáhlá problematika RocketIO rozhraní je popsána v [25].

3.2 Dostupná řešení PPP a HDLC kontroléru

3.2.1 MTI 6.195 Byte-wide HDLC controller

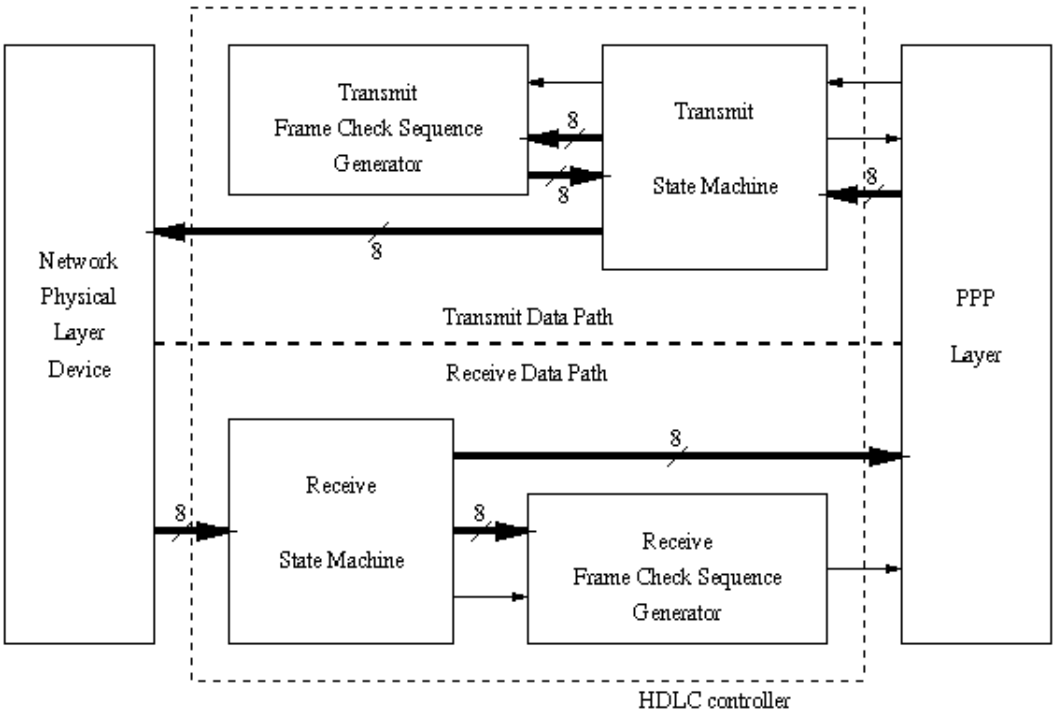
Jedná se o projekt z univerzity MIT [10], který implementuje HDLC (High-level Data Link Control) kontrolér velice přehledným způsobem. Na následujícím obrázku 3.9 je ilustrující blokové schéma.
Je rozdělen na vysílací a přijímací část, která se vždy skládá z řídicího automatu a jednotky pro výpočet kontrolního součtu. Jak již název naznačuje datová šířka je 8b, což znemožňuje snadné použití pro SONET OC-48. Snadnému rozšíření pro datovou šířku 16b brání řídicí automat, který je napsán „zvláštním“ způsobem. Stavy automatu jsou konstanty kódovány 1 z N, což znemožňuje syntéznímu nástroji optimalizaci. Zásadní překážkou v rozšíření však je pouze jeden automat, který generuje/přijímá L1 datový proud a zároveň převádí byty 0x7D a 0x7E. Při zvýšení datové šířky 2x se počet stavů zvýší přibližně 4x, protože musíme brát v úvahu různé kombinace bytů 0x7D a 0x7E.
Další možností je zapojit 2 tyto jednotky paralelně vedle sebe. Zařadit před a za ně multiplexor/demultiplexor a vyrovnávací FIFO paměti.

3.2.1.1 Klady/zápory

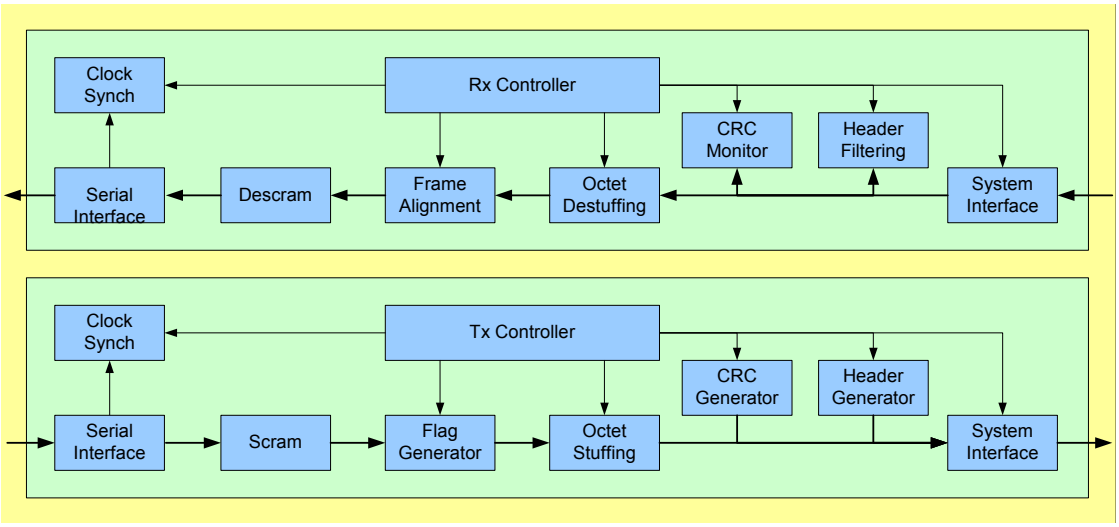
Klady a zápory jsou shrnuty v tabulce 3.4

3.2.2 Modelware: nAccess HDLC controller

Soft core použitelné pro FPGA a ASIC je opět rozděleno na nezávislou vysílací a přijímací část. Pro úplnost je na obrázku 3.10 uvedeno blokové schéma.



Obrázek 3.9: MTI 6.195 HDLC controller (zdroj MTI)



Obrázek 3.10: Modelware HDLC controller

Klady	Zápory
Otevřený kód, dobře dokumentovaný	kód obsahuje chyby
bezplatné použití	obtížná rozšiřitelnost na 16b datovou cestu

Tabulka 3.4: Klady a zápory HDLC kontroléru z MIT

Základní funkce jsou stejné jako u předchozího kontroléru z MIT. Přidané jsou možnosti konfigurace. Můžeme definovat obsah pole address, control a protocol které budeme generovat nebo naopak přijímat. Ostatní rámce budou při přijetí zahozeny. Je možné volit mezi kontrolním součtem CRC16 a CRC32. Dále jednotka poskytuje velké množství kontrolních statistik. Jsou to čítače přijatých rámců, přijatých krátkých rámců (příliš krátké pro výpočet CRC), přijatých rámců se špatným kontrolním součtem, odmítnutých rámců (jiné pole address, control, protocol), odeslaných rámců.
Dále nabízí kódování/dekódování dat polynomem $x^{43} + 1$, které vyžaduje [23] pro vložení do SONET/SDH SPE.

3.2.2.1 Klady/zápory

Klady a zápory jsou shrnuty v tabulce 3.5

Klady	Zápory
Dodáváno ve formě VHDL kódů nebo EDIF	nejasná licence
Robustní řešení	datová šířka není uvedena (nepochybně 8b)
volitelné (de)kódování polynomem $x^{43} + 1$	
poskytované statistiky	

Tabulka 3.5: Klady a zápory HDLC kontroléru f. Modelware

3.2.3 CoreEL PPP8 HDLC core (CC318f)

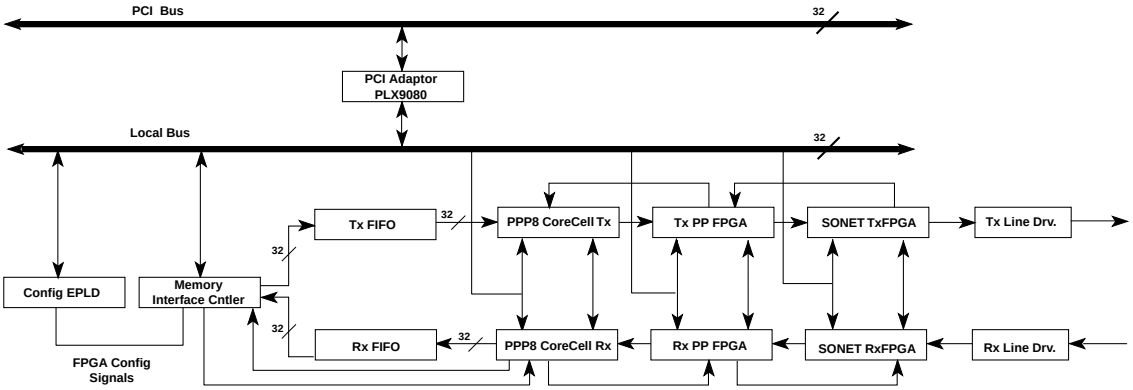
Mezi další komerční řešení ve formě IP core patří CC318f [11] od firmy CoreEL MicroSystems. Jedná se o 8b PPP/HDLC kontrolér, který splňuje RFC 1619 (nahrazeno RFC 2615 [23]).
Datová šířka je opět 8b a proto by bylo nutné zapojit více těchto jednotek paralelně. Blokové schéma vysílací a přijímací části nebudu uvádět, protože nepřináší nic nového vzhledem k předchozím dvěma kontrolérům. Prostudování [11] mě přivádí k závěru, že se rozhodně nejedná o PPP kontrolér ve smyslu udržování spojení na linkové vrstvě. Tento kontrolér pouze přidává pole PPP rámce, které je nastaveno na předem definovanou hodnotu. Zpracování LCP (Link Control Protocol) a NCP (Network Control Protocol) paketů by bylo nutné řešit v externí jednotce.
Doporučené zapojení této jednotky pro PoS je na obrázku 3.11. Uvádím jej zde pro představu, jak může implementace PoS vypadat.

3.2.3.1 Klady/zápory

Klady a zápory jsou shrnuty v tabulce 3.6

3.3 Komplexní HW řešení

Pro zajímavost a srovnání zde uvádím již hotová hardwarová řešení v podobě čipů.



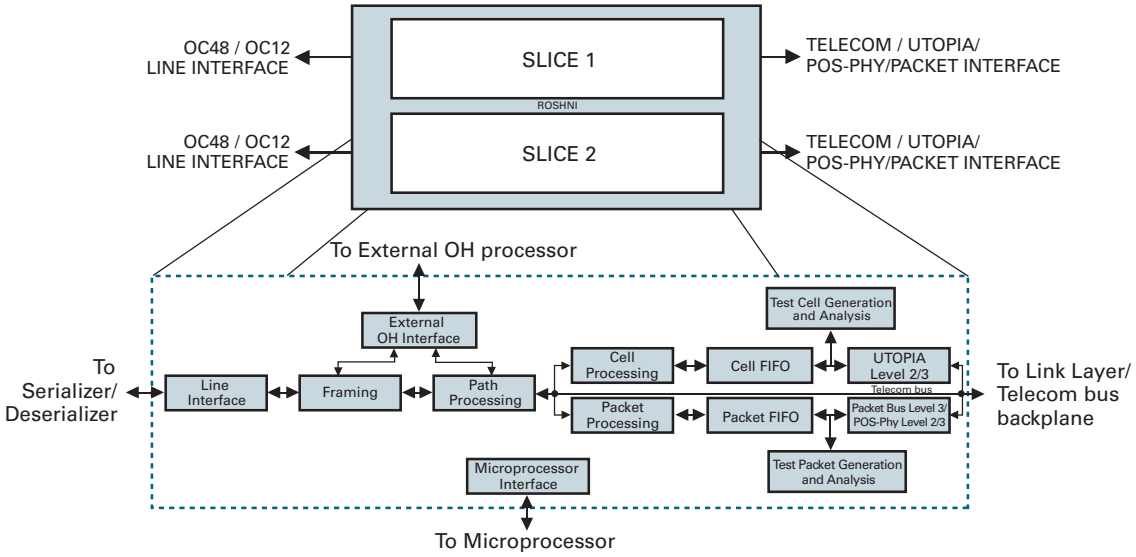
Obrázek 3.11: Zapojení CoreEL CC318f

Klady	Zápory
Robustní řešení	Datová šířka 8b
Pseudo PPP kontrolér	Neimplementuje LCP a NCP
Poskytované statistiky	
Zahazování poškozených paketů	

Tabulka 3.6: Klady a zápory HDLC kontroléru f. CoreEL

3.3.1 Conexant SONET/SDH OC-48 POS/ATM Mapper (Roshni PX-4805)

Jedná se o hardwarové řešení, které v čipu zahrnuje dvou-kanálový SONET/SDH OC-48 framer a pointer procesor s možností zpracovávat PoS a ATM. Výrobce [9] uvádí nízkou spotřebu, která je 1,25W na jeden OC-48 kanál. Čip je vyroben 0,14μm technologií a použité pouzdro je TBGA s 648 piny. Blokové schéma je na obrázku 3.12.



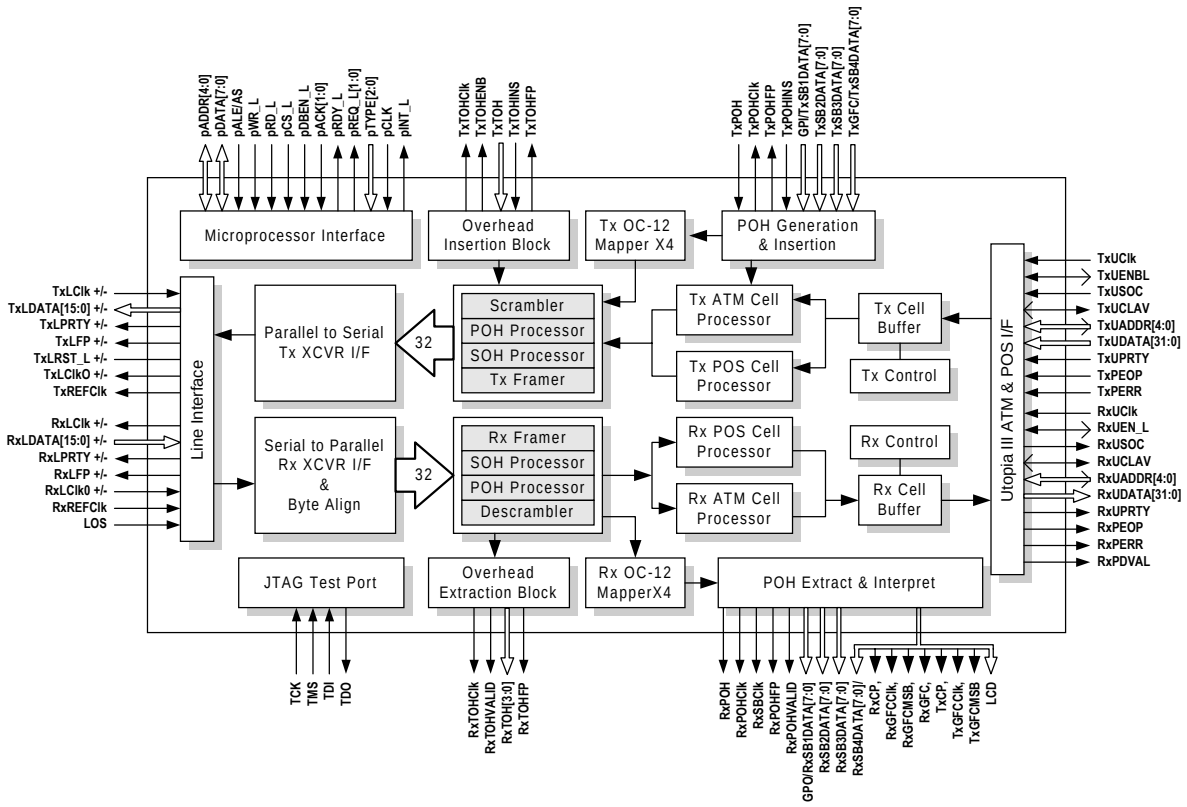
Obrázek 3.12: HW řešení PoS f. Conexant

Je možné zapnout tzv. transparentní mód, ve kterém jsou pakety pouze přeposílány. To je vý-

hodné pro RPR (Resilient Packet Rings) aplikace. Jako u samostatných HDLC kontrolérů je též možné filtrování polí address, control a protocol. Každý kanál může být nezávisle nakonfigurován pro STS-48c, STS-12c a 4x STS-12c. Dále pro každý SONET/SDH datový proud je možné použít mapování do ATM buněk nebo PoS. Dvě nezávislá rozhraní do systému podporují protokoly UTOPIA Level 2/3, POS-PHY Level 2/3, Packet Bus Level 3 nebo 8/32b „telecom bus“. Připojení k fyzickému rozhraní je realizováno několika způsoby. První se nazývá OIF-54, což je 4 bitový LVDS (Low Voltage Differential Signal). Další jsou 16b LVPECL (Low Voltage Positive Emitter Coupled Logic) v STS-48 módu a 8 bitový LVPECL v STS-12 módu.
Samozřejmostí je zpracování/generování pointerů na data, SOH (Section Overhead), LOH (Line Overhead) a POH (Path Overhead). Dále jednotka detekuje a generuje: OOF (Out Of Frame), LOF (Loss Of Frame), LOS (Loss Of Signal), chyby APS (Automatic Protection Switch), AIS (line/path alarm), RDI (line/path remote defect). Implementuje specifikace ATM Forum User Network a PPP over SONET/SDH (RFC 2615).

3.3.2 Exar SONET/SDH OC-48 ATM/UNI/POS Mapper (XRT95L51)

Velice komplexní HW řešení je od firmy Exar [13]. Pro představu jak je jednotka navržena, opět uvádím blokové schéma, které je na obrázku 3.13. Čip je zapouzdřen v PBGA (Plastic Ball Grid Array) a má 388 pinů.



Obrázek 3.13: HW řešení PoS f. Exar

Jedná se o ATM/PPP procesor s integrovanou jednotkou pro zapouzdření dat do SONET/SDH OC-48 rámců. SONET/SDH framer splňuje specifikace ANSI/ITU-T a má podobné možnosti generování/detekce hlavičky (SOH, LOH, POH)jako předchozí řešení. Samozřejmostí je (de)kódování dat jak 7b polynomem pro SONET/SDH tak rozšířeným 43b polynomem pro PoS. Dále umožňuje vkládání a zpracování alarm signálů. SONET/SDH vysílací a přijímací

část je možné nakonfigurovat pro OC-48c nebo 4x OC-12. Rozhraní do systému má datovou šířku 32b při frekvenci 77 MHz. Zařízení je konfigurovatelné přes registry, které jsou přístupné přes 8b paralelní rozhraní. Pro výčet všech vlastností doporučuji přečtení [13].

3.4 Zhodnocení dostupných řešení

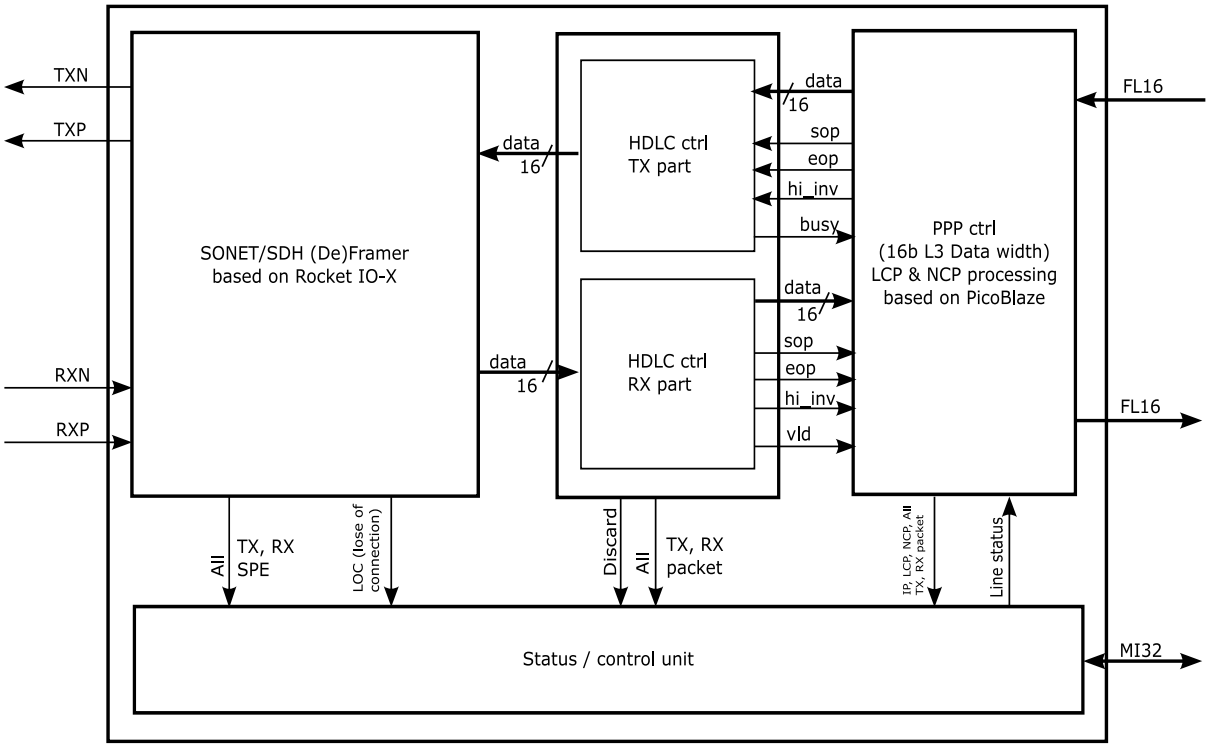
Jednotka, která převádí Packet over SONET (přenášený po rozhraní RocketIO) na FrameLink (LocalLink), není na trhu dostupná. Největší výhody představuje použití již hotového hardwarového řešení. Ušetříme zdroje v aplikačním FPGA a vývoj je možné soustředit k samotným aplikacím. Tento přístup však znamená redesign celé interfacové karty a to vzhledem k již existujícímu PCB není možné. Také by to znamenalo snížení univerzality této karty.
Proto navrhneme vlastní jednotku inspirovanou předešlou analýzou. Pro návrh komponenty SONET Framer bude dobré vycházet z Altera SONET/SDH Compiler (viz. kapitola 3.1.1). Je velice dobře dokumentovaný, univerzální a odpovídá požadavkům kladených ve specifikaci na tuto jednotku. HDLC kontroléry jsou dostupné s datovou šířkou pouze 8b, což je neslučitelné s požadavkem přenést OC-48 (2,5Gb/s). Budu nutné navrhnout paralelní zapojení již hotových 8b HDLC kontrolérů, nebo navázat na projekt z univerzity MIT (viz. kapitola 3.2.1) a rozšířit jej na datovou šířku 16b. PPP kontrolér řešící pouze enkapsulaci a dekapsulaci paketů síťové vrstvy z/do PPP rámců je součástí HDLC kontroléru CoreEL PPP8 (viz. kapitola 3.2.3). Takový PPP kontrolér, který zpracovává pakety linkové vrstvy, není dostupný. Návrhem vlastního řešení jednotky se budu zabývat v následující kapitole.

4 Návrh vlastního řešení

Vzhledem k tomu, že neexistuje komplexní řešení, bylo nutné vyvinout vlastní s ohledem na Combo 4SFPRO/OC-48 kartu. V této kapitole je uveden ideový návrh, detailnější návrh, ze kterého vychází implementace, je popsán v následující kapitole.

4.1 Blokový návrh

Navržené blokové schéma je dané jednotlivými kroky, které je nutné provádět během příjmu/vysílání SONET/SDH rámce.
S ohledem na čas vývoje jsem se snažil využít již hotové jednotky s odpovídající licencí. Těmto požadavků odpovídají pouze Xilinx aplikační příklady a generické jednotky vyvinuté v rámci projektu Liberouter.
Jednotka *pos_buf* přímo neimplementuje rozhraní RocketIO-X, ale poskytuje k němu 16b rozhraní. Tato jednotka je vložena do „obálky“ *pos_buf_rio*, která instancuje RocketIO-X rozhraní a komponentu pro změnu datové šířky. Důvodem k takové implementaci je požadavek na externí násobičku nebo druhý přesný krystal o frekvenci 124,416 MHz, který na kartě není k dispozici (viz. kapitola 3.1.4.1). Toto řešení přináší výhody v následném testování, kdy je od sebe odděleno poměrně složité rozhraní RocketIO a vlastní jednotka *pos_buf*.

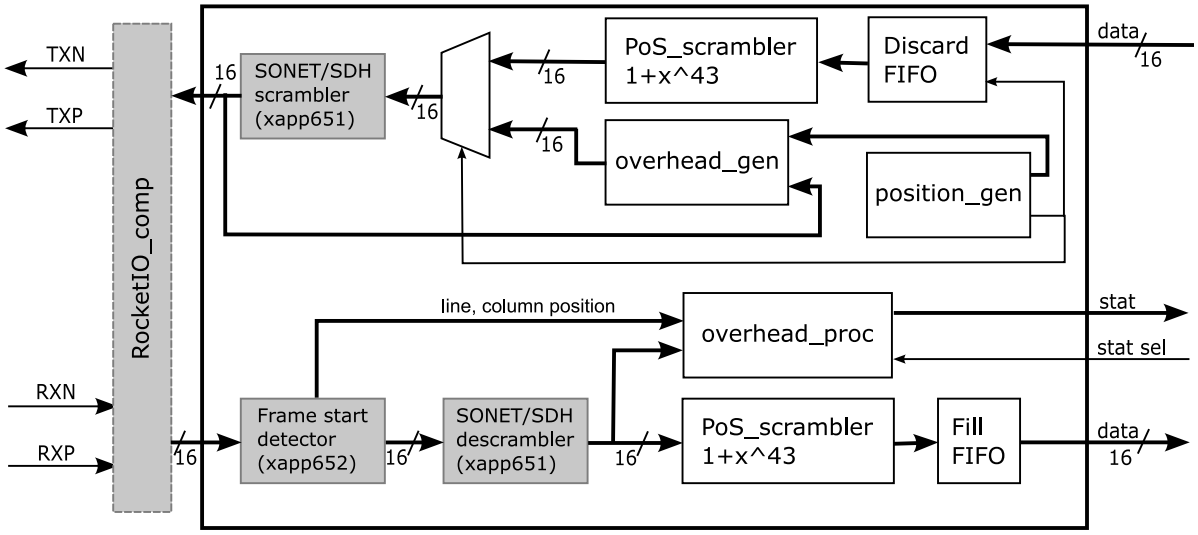


Obrázek 4.1: Navrhované blokové schéma jednotky pos_buf

Obrázek 4.1 představuje blokové schéma vlastního řešení jednotky pro zpracování PoS. PPP je stavový protokol a vyžaduje propojení vstupní a výstupní části. Proto je návrh jednotky řešen jako celek. Budu jej označovat *pos_buf*. V následujícím textu nastíním implementaci této jednotky.

4.2 SONET/SDH (de)framer

Pro realizaci fyzické vrstvy není možné použít žádné z popsaných existujících řešení. Řešení f. Altera je licenčně svázáno s FPGA obvody této firmy. Řešení f. Xelic není příliš dokumentováno a jeho koupě by byla riskantní. Z těchto důvodů je nezbytné tuto jednotku navrhnout. Při vlastním návrhu bylo využito Xilinx aplikačních příkladů a blokových schémat od firmy Altera. Blokový návrh (obrázek 4.2) je inspirován SONET/SDH Framerem od f. Altera. Bloky označené šedou barvou jsou již hotové (xapp), ostatní bloky je nutné implementovat. Níže jsou popsány části, ze kterých se jednotka skládá.



Obrázek 4.2: Blokové schéma SONET/SDH (de)frameru

4.2.1 Rocket IO komponenta (RocketIO_comp)

Tato komponenta není součástí SONET/SDH (de)frameru, ale obálky *pos_buf_rio*. Popisuji ji na tomto místě z důvodů přímé návaznosti na SONET/SDH (de)framer. Zahrnuje v sobě kódy z core generátoru pro zpřístupnění Rocket IO-X rozhraní. Pro přenosovou rychlost 2.488 Gb/s je doporučená datová šířka 16b. Combo 4SFPRO karta je osazena oscilátorem o frekvenci 155.52 MHz což odpovídá 2.488 Gb / 16. Použité nastavení GT10_OC48_2 je popsáno v [25] na str. 195. Tato konfigurace poskytuje 20b datovou šířku, protože je vnitřně uzpůsobena pro kódování 8/10. Z tohoto důvodu je hned za RocketIO rozhraním připojena xapp649 [19], která převádí 20b na 16b.

4.2.2 Generátor pozice (position_gen)

Jedná se o dva čítače, které generují pozice SONET/SDH rámce. Výstupem této jednotky je pozice v hlavičce rámce a signál, který přepíná multiplexer mezi hlavičkou a daty.

4.2.3 Generování (overhead_gen) a zpracování (overhead_proc) SONET/SDH hlaviček

Jednotka pro generování hlavičky obsahuje paměť, ve které jsou uloženy jednotlivé položky. Čítače z generátoru pozic slouží jako adresa do paměti. Některé údaje je nutné počítat přímo z dat (kontrolní součty, ukazatel na data, ...).

Naopak pro ověření informací v hlavičce slouží overhead_proc. Plná implementace je však nepovinná. Např. chyby v komunikačním kanále můžeme detekovat až při ověření kontrolního součtu po HDLC dekapsulaci. Pro naši aplikaci postačí implementovat pouze položky „Požadováno“ z tabulky 4.1 (převzato z ITU-T G.707 [14] str.85).

SOH byte	Vysílací část	Přijímací část
A1, A2	Požadováno	Požadováno
J0-Z0/C1	Nepovinné	Nepovinné
B1	Požadováno	Nepoužito
E1	Nepoužito	Nepoužito
F1	Nepoužito	Nepoužito
D1-D3	Nepoužito	Nepoužito
B2	Požadováno	Požadováno
K1, K2 (APS)	Nepovinné	Nepovinné
K2 (MS-AIS)	Požadováno	Požadováno
K2 (MS-RDI)	Požadováno	Požadováno
D4-D12	Nepoužito	Nepoužito
S1	Nepoužito, generováno 00001111	Nepoužito
M1	Požadováno	Nepovinné
E2	Nepoužito	Nepoužito

Tabulka 4.1: Redukované TOH položky

4.2.4 POS scrambler

Implementuje polynom pro kódování dat (payload) $1 + x^{43}$ popsaný v RFC 2615 [23].

4.2.5 Discard FIFO

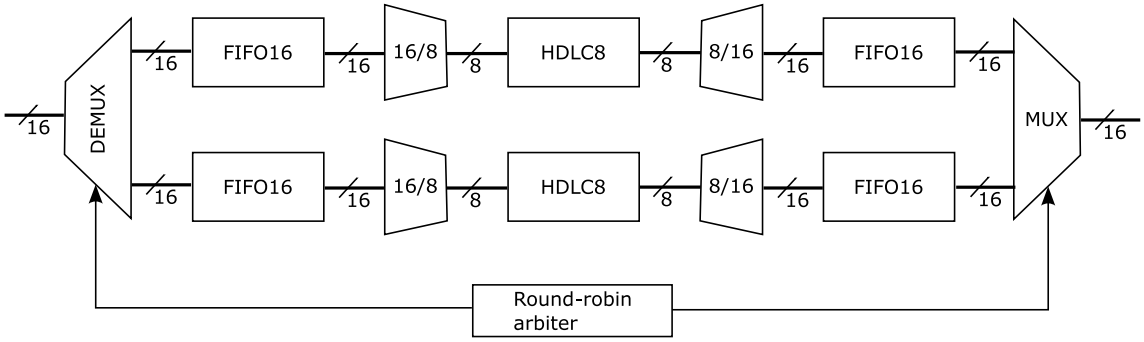
Zaobaluje FIFO paměť [2] z projektu Liberouter. Přidává možnost odstranění flagu 0x7E z dat. Flag buď není do FIFO paměti vůbec vložen, nebo naopak je vyčten a není poslán na výstup. Tato funkcionalita je zavedena k odstranění přeplňování FIFO paměti, ke které by jinak docházelo důsledkem vkládání SONET/SDH hlavičky.

4.2.6 Fill FIFO

Tato jednotka generuje nepřetržitý datový tok. Data jsou do FIFO paměti ukládána pouze, přichází-li SONET/SDH data (SPE). Tím by docházelo k vyprázdnění FIFO paměti a možnému přerušení rámce. Proto je výstupní datový tok doplněný o flagy 0x7E ve vhodných místech. Vhodným místem se rozumí situace, kdy je flag ve výstupních datech z FIFO paměti.

4.3 HDLC kontrolér

Na linkové vrstvě je situace obdobná. HDLC kontrolérů je dostupných několik, ale vždy jen pro datovou cestu širokou 8b. Na obrázku 4.3 je návrh jak zapojit dva (snadno rozšiřitelné pro N) 8b HDLC (PPP) kontroléry.
Obrázek popíšu z levé strany. Na vstupu je „demultiplexor“, který je pravidelně přepínán po průchodu celého rámce. Rámec je uložen ve FIFO paměti o datové šířce 16b. Z ní jsou data vyčítána 2x pomaleji (na stejné hodinové frekvenci) a převedeny na datovou šířku 8b, zpracovány

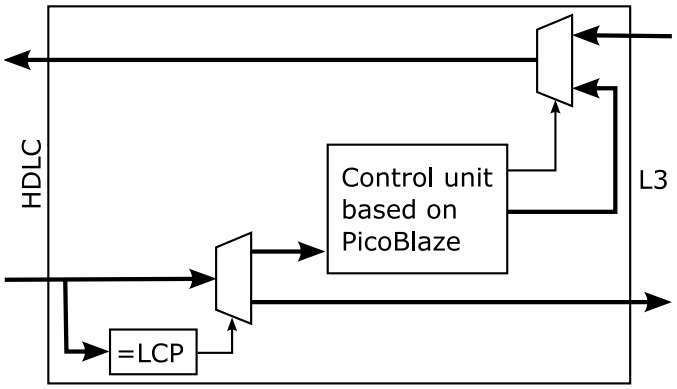


Obrázek 4.3: Paralelní zapojení 8b HDLC kontrolérů

HDLC (PPP) kontrolérem. Poté jsou převedeny zpět na 16b a uloženy do vyrovnávací FIFO paměti. Odtud jsou vyčítány přes „multiplexor“ plnou rychlostí. Multiplexor je opět přepínán po průchodu celého rámce.
Druhou možností je napsat vlastní HDLC kontrolér, který bude nativně 16b. Tento přístup je vhodnější, protože zabere přibližně jen 2x víc zdrojů na čipu než HDLC 8b. U prvního řešení musíme započítat 2x FIFO16, multiplexor, demultiplexor, čítač pro přepínání mezi toky a registry pro změnu datové šířky.
Z těchto důvodů budu realizovat druhý návrh, tj. nativně 16b HDLC kontrolér. Realizován bude pomocí konečných automatů, konkrétně dvou na vysílací a dvou na přijímací straně. První automat vytváří HDLC protokol, tj. vkládá pole address, control a kontrolní součet. Následuje automat, který se stará o tzv. „Byte stuffing“. Pokud nejsou žádná data vysílána, generuje výplň 0x7E. Ve vysílaných datech nahrazuje byty 0x7D a 0x7E za 0x7D a xor původního bytu s 0x20. Analogická situace je na přijímající straně.

4.4 PPP kontrolér

Poměrně dost času jsem věnoval hledáním PPP kontroléru, který implementuje LCP a NCP protokol a je napsaný v jazyce, který se dá syntetizovat. Mé hledání bylo však neúspěšné. Proto navrhnu vlastní (nativně 16b) kontrolér.



Obrázek 4.4: Návrh PPP kontroléru

Jedná-li se o rámec, ve kterém je vložen paket síťové vrstvy (pole „Protocol“ v PPP rámci má hodnotu menší než 0x4000) musí být co nejrychleji přenášen ze vstupu na výstup. V opačném

případě se jedná o řídicí paket linkové vrstvy, a proto je přesměrován do řídicí jednotky. Pro implementaci se nabízí konečný automat, nebo jednoduchý procesor. Druhá možnost je vhodnější. Umožňuje postupné přidávání dalších protokolů (PAP, CHAP, ...) bez složitého zásahu do HW. Také se tím zjednoduší implementace. Jako procesor bude použit softprocesor PicoBlaze [26] [8], který svým výkonem pro tuto aplikaci postačuje. Pro naši aplikaci postačí implementovat pouze základní řídicí protokol LCP. Vzhledem k navržené architektuře je možné následně přidat další protokoly dopsáním programu pro procesor. Na obrázku 4.4 je pro představu blokové schéma této jednotky.

4.5 Nastavení a informace o stavu jednotky

Přes tuto jednotku bude možné konfigurovat celou jednotku a naopak o ní zjišťovat informace. Bude jen zaobalovat kontrolní výstupy z ostatních jednotek. Připojení k SW bude realizováno přes sběrnici MI32, která je vyvinuta v rámci projektu Liberouter. Ve schématu je pod názvem Status/control unit.

5 Realizace

Při realizaci jsem použil vývojové prostředí ISE od firmy Xilinx. Zahrnuje v sobě editor, syntézní nástroj XST, simulátor, mapování na technologii, place & route (PaR) a generování bitstreamu. Zdrojové kódy jsou psány v jazyce VHDL [12], který umožňuje behaviorální i strukturální popis jednotky. Z hlediska doby realizace jsem se snažil využít dostupné komponenty.

Adresář projektu má tuto strukturu:

```
_ise           projekt + soubory generované ISE
comp
   common      obecné jednotky z projektu Liberouter (FIFO, ...)
   ctrl_stat   implementace control / status jednotky
   hdlc_ctrl   implementace HDLC kontroléru
   ppp         implementace PPP kontroléru
   sonet       implementace SONET/SDH Frameru
coregen        vygenerované soubory z Xilinx CORE generator
pkg            globální VHDL package
sim            simulace celé jednotky
pos_buf.vhd    top level
```

5.1 SONET/SDH Framer

SONET/SDH Framer je implementován podle blokového schématu 4.2 uvedeného v analýze. Implementace frameru je maximálně podřízena zadání, podporuje pouze rychlost OC-48, virtuální kontejner VC-4-16c a PPP rámce v SPE (byte C2 = 0xCF). Vysílací a přijímací část je na sobě nezávislá, jsou pouze zapouzdřeny ve společné „top level“ entitě `sonet_framer.vhd`. Jednotka je umístěna v adresáři `sonet` a skládá se z těchto souborů:

```
sim/                              simulace (testbench pro ModelSim)
discard_fifo.vhd                  Vyrovnávací FIFO pro HDLC rámce
fill_fifo.vhd                     FIFO generující nepřetržitý tok HDLC rámců
overhead_gen.vhd                  generátor SONET/SDH hlaviček
overhead_proc.vhd                 zpracování SONE/SDH hlaviček
pos_scrambler.vhd                 implementace polynomu 1 + x^43
position_gen.vhd                  generátor pozice
sonet_framer.vhd                  top_level entita a komponenta
xapp651_scrambler_sonet_16.vhd    SONET/SDH scrambler (Xilinx)
xapp652_oc48_aligner_16.vhd       detekce začátku SONET/SDH rámce (Xilinx)
xapp652_*.vhd                     pomocné jednotky xapp652
```

Xilinx xapp651 (SONET/SDH Scrambler/descrambler) je použit, tak jak jej poskytuje firma Xilinx. V xapp652 (detekce začátku rámce) bylo nutné v souboru `xapp652_oc48_aligner_16.vhd` zakomentovat/odkomentovat řádky č. 160 a 161 takto:

```
160 -- n <= "000001001111"; -- use to keep simulation quick
161    n <= "100001101111"; -- use for implementation
```

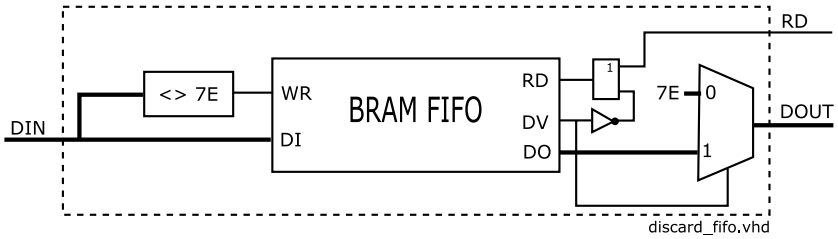
Bez této úpravy pracuje jednotka s menším SONET/SDH SPE, ale TOH zůstává stejné, což neodpovídá OC-48. Tato skutečnost není nikde popsána, a proto to zde uvádím.

5.1.1 FIFO

Zde popíšu „discard“ a „fill“ FIFO jednotky, které jsou navrženy výhradně pro práci s HDLC rámci. Obě zaobalují generickou FIFO paměť [2] realizovanou na projektu Liberouter, která umožňuje nastavit typ použité paměti (distribuovaná, bloková), šířku dat (1, 2, 4, 9, 18, 36), velikost položky a počet položek. V obou případech je použito FIFO s 18b datovou šířkou (využito pouze 16b)

5.1.1.1 „Discard“ FIFO

Jednotka slouží k dočasnému uložení nepřetržitého toku dat, který přichází z HDLC kontroléru a nemůže být přímo odesílán (přenos TOH). Aby nedocházelo k přeplnění FIFO paměti, a následnému zahazování dat jsou odstraněny přebytečné flagy 0x7E. K oddělení HDLC rámců stačí pouze jeden flag 0x7E, ostatní jsou použity jako výplň. Zjednodušené (vynechány vstupní/výstupní a pomocné registry) schéma jednotky je na obrázku 5.1.

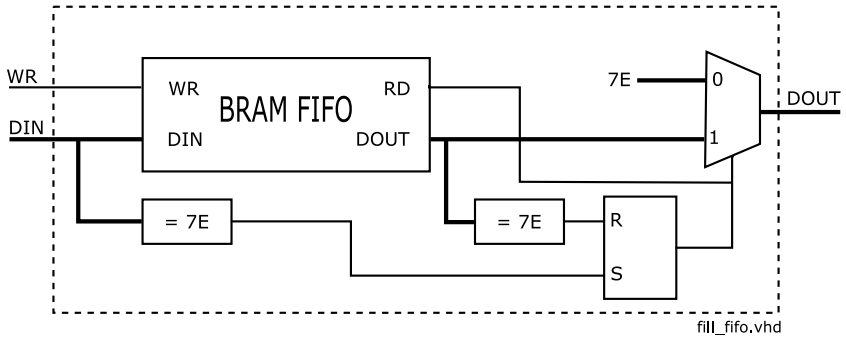


Obrázek 5.1: Discard FIFO

Počet položek FIFO paměti je zvolen takto: `k x TOH_width x OC-48`, kde `k` je konstanta, `TOH_width` je šířka hlavičky (=3) a `OC-48` je použitá rychlost (=48). Je patrné, že tato implementace spoléhá na mezery mezi přenášenými HDLC rámci. Pokud bude tok dat nepřetržitý, začne docházet ke ztrátám. To je vyřešeno vytažením signálu `FULL` z FIFO paměti a navázáním na FrameLink protokol, čímž zaručíme pozastavení vstupu.

5.1.1.2 „Fill“ FIFO

Ideové schéma jednotky je na obrázku 5.2. Nejsou v něm zakresleny vstupní / výstupní registry a signály ošetřující platnost dat.



Obrázek 5.2: Fill FIFO

Data `DIN` jsou do FIFO paměti ukládána, je-li signál `WR = '1'` (je přenášen SPE). Pokud je ve vstupních datech flag 0x7E je nastaven RS registr a to způsobí vyčítání dat. Nastavení má

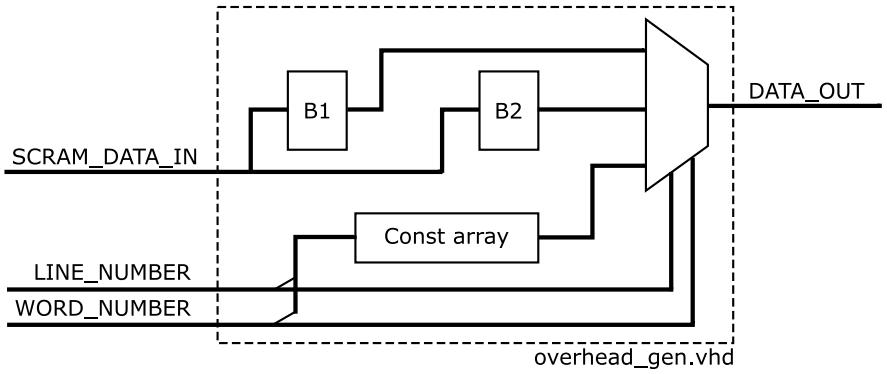
větší prioritu než reset. Jakmile začnou přicházet jiná data než flag, zastaví se vyčítání dat z paměti do doby, než flag opět přijde. Tím je zaručeno, že HDLC rámec nebude přerušen (např. TOH). Mezi tím je na výstupu `DOUT` neustále hodnota 0x7E. Flag oddělující dva HDLC rámce je vložen pouze jednou (není zakresleno ve schématu). Tím je minimalizována odezva mezi koncem uložením rámce a začátkem jeho vyčítání.
Počet položek FIFO paměti je nastaven na 33000. `16B x 33000 > 64kB`, což je maximální velikost HDLC/PPP rámce.

5.1.2 Generátor pozice (position_gen)

Jednotka realizuje dva binární čítače navržené pro generování pozice v SONET OC-48/VC-4-16c rámci. Čítač řádek `LINE_NUMBER` má rozsah 0:8, čítač slov `WORD_NUMBER` 0:2159. Poskytuje signály `OVERHEAD` (hlavička) a `PAYLOAD` (data), které udávají, zda aktuální pozice náleží hlavičce nebo datům. Do hlavičky je brán SOH, LOH i POH. To je možné díky VC-4-16c, kde je POH na pevném offsetu. Proto je u Xilinx xapp652, který je obecnější, signál `OVERHEAD` aktivní pouze při SOH a LOH. Signál `SCRAMBLE` je roven 0 pokud jsou přenášeny byte A1, A2 a J0. Slouží jako enable pro xapp651.

5.1.3 Generátor hlavičky (overhead_gen)

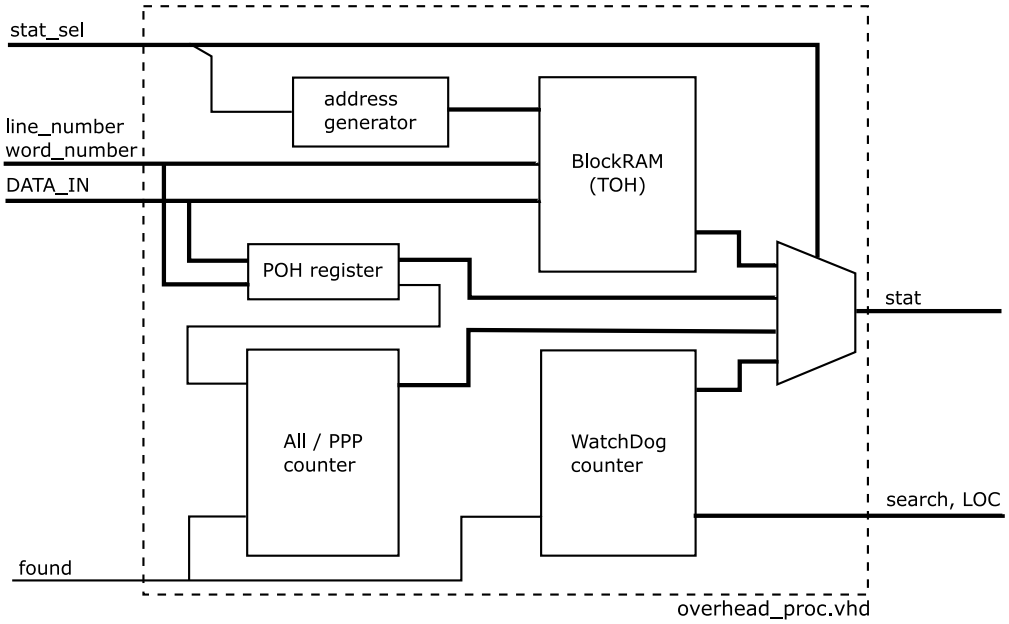
Jednotka generuje SOH, LOH i POH hlavičky. Jsou implementány pouze „Požadované“ byty shrnuté v tabulce 4.1.
V distribuované paměti jsou uloženy konstanty (A1, A2, ...). Podle specifikace jsou počítány kontrolní součty B1 a B2 z předchozího zakódovaného rámce. Ilustrační schéma je na obrázku 5.3.



Obrázek 5.3: Generátor hlavičky

5.1.4 Zpracování hlavičky (overhead_proc)

Jednotka zpracovávající SONET/SDH hlavičky (TOH a POH) má za úkol především shromažďovat statistiky o přijatých rámcích. Skládá se z několika bloků (obrázek 5.4).
„WatchDog counter“ při přetečení generuje (je nulován signálem `FOUND` = nalezen začátek rámce) signál `LOC` (lose of connection = ztráta nosné) a `search` (startuje hledání začátku SONET/SDH rámce). V „BlockRAM“ je průběžně ukládána (přepisována) hlavička TOH pro možnost vyčtení do SW. Pro POH hlavičku k tomu samému slouží „POH register“. „All / PPP counter“ představuje dva 32 bitové čítače pro uložení počtu všech přijatých SONET/SDH a SPE/PPP rámců.



Obrázek 5.4: Zpracování hlavičky

Hodnota signálu stat_sel	Význam	výstup `STAT`
SONET_CAPTURE_FROOZEN	zastaví ukládání TOH	-
SONET_CAPTURE_ENABLE	start ukládání TOH (default)	-
SONET_CAPTURE_READ	vyčtení TOH	648 x TOH(15:0)
SONET_STATUS	stav jednotky	(1) LOCKED [1], (0) LOC
SONET_RX_ALL	# přijatých SONET rámců	počet 32b
SONET_RX_PPP	# přijatých SPE/PPP rámců	počet 32b

Tabulka 5.1: Poskytované informace jednotkou „overhead_proc“

Pro přepínání mezi výstupy z jednotek slouží signál `STAT_SEL`. Možné hodnoty definované jako konstanty jsou uvedeny v tabulce 5.1.

5.1.5 PoS_scrambler

Jednotka realizuje (od)kódování polynomem $1 + x^{43}$. Verze pro sériová data (1b) je v RFC 2615 [23] na straně 3 a 4. Zapojení vysílací a přijímací části se od sebe liší vstupem do posuvného registru. Vysílací část má vstup za xor obvodem, přijímací před ním. Ke změně funkce (vysílání/příjem) jednotky slouží generic `scramble`. Realizace pro 16b data vypadá zjednodušeně takto:

```
xor_data <= shift_reg(15 downto 0) xor DATA_IN;
DATA_OUT <= xor_data;

-- 43b shift reg
process(CLK)
begin
   if CLK = '1' and CLK'event then
      if SCRAMBLE then
         shift_reg <= xor_data & shift_reg(42 downto 16);
      else
         shift_reg <= DATA_IN & shift_reg(42 downto 16);
      end if;
   end if;
end process;
```

5.2 HDLC kontrolér

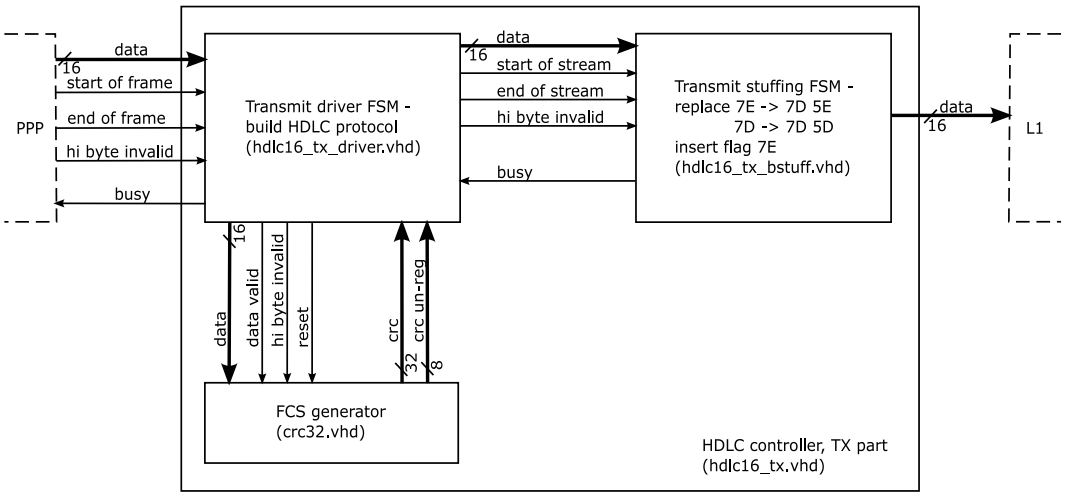
Návrh je inspirován projektem 6.195 Final Project z MIT [10]. Je rozdělen na vysílací a přijímací část, ale oproti [10] je navržen pro datovou šířku 16b. Dalším rozdílem je rozdělení komponenty s automatem na 2 komponenty. Realizace se skládá z těchto souborů:

```
sim/                    simulace (testbench pro ModelSim)
hdlc16_tx.vhd           top level vysílací části
hdlc16_tx_driver.vhd    automat generující HDLC protokol
hdlc16_tx_bstuff.vhd    automat zajišťující "Byte stuffing" na 16b datech
hdlc16_rx.vhd           top level přijímací části
hdlc16_rx_driver.vhd    automat přijímající HDLC protokol
hdlc16_rx_bstuff.vhd    automat zajišťující "Byte destuffing" na 16b datech
crc32.vhd               výpočet kontrolního součtu
```

V následujících podkapitolách popíšu zvlášť vysílací a přijímací část a komponentu pro výpočet kontrolního součtu, která je pro obě části společná.

5.2.1 Vysílací část (TX)

Blokové schéma vysílací části je na obrázku 5.5.
Komponenta „Transmit driver FSM“ (`hdlc16_tx_driver.vhd`) implementuje automat, který je na obrázku 5.6. Zajišťuje generování HDLC protokolu. Název stavu značí, že po přechodu byla určitá data odvysílána. Pro příklad ve stavu `st_DF` bylo odvysíláno defaultní pole address a control tj. 0xFF03.



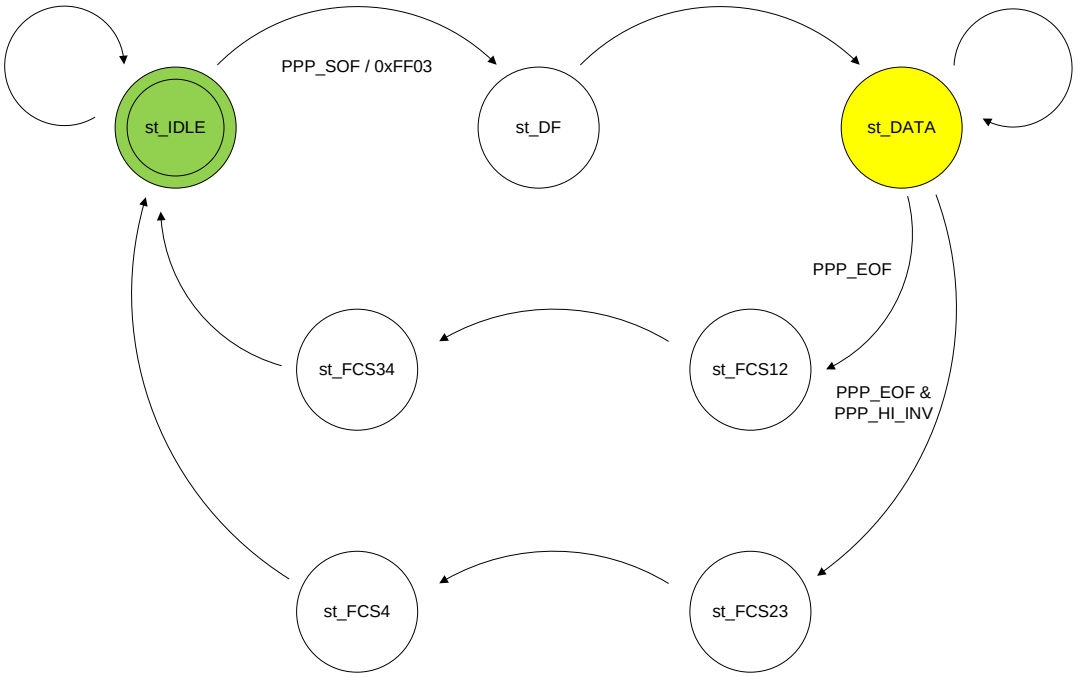
Obrázek 5.5: HDLC kontrolér - vysílací část

Automat zůstává ve stavu `st_IDLE` dokud není signál[2] `PPP_SOF = '1'`. Poté jsou odvysílány defaultní položky a začíná přenos samotných dat z vyšší vrstvy, což odpovídá stavu `st_DATA`. V tomto stavu automat setrvává, dokud je `PPP_EOF = '0'`. Poté je vložen kontrolní součet. Pokud byl počet bytů dat sudý, automat pokračuje stavy `st_FCS12` (spodních 16b kontrolního součtu) a `st_FCS34` (horních 16b kontrolního součtu). V opačném případě je spodních 8b kontrolního součtu odesláno již s posledním bytem dat. Poté je odesláno prostředních 16b (`st_FCS23`). Ve stavu `st_FCS4` je odesláno horních 8b kontrolního součtu a nastaven signál `HDLC_HI_INV = '1'`[3].
Výstup je připojen do komponenty „Transmit stuffing FSM“ (`hdlc16_tx_bstuff.vhd`). Jedná se opět o konečný automat, který zajišťuje transparentnost dat. Uveden je na obrázku 5.7, který je značně zjednodušený (stavy SHIFT a END jsou sloučeny do jednoho a nejsou zakresleny výstupy). Pokud nejsou přenášena žádná data, je vysílán tzv. „Flag“ 0x7E7E. V přenášených datech jsou nahrazovány byty 0x7D a 0x7E za 0x7D a xor původního bytu. Pro 8b verzi by měl automat 3 stavy (idle, direct, stuff). 16b verze má 10 stavů a 42 přechodů, které musí řešit „stuffing“ dolního, horního nebo obou bytů ve vstupním slově a následné posunutí následujících slov.
V původním návrhu nebyly zahrnuty registry mezi komponentami „Transmit driver FSM“ a „Transmit stuffing FSM“. Po syntéze byl odhad hodinové frekvence pouze 106,68 MHz. Po vložení registrů je odhad frekvence 156,17 MHz (pro OC-48 potřebujeme minimálně 155,52 MHz).

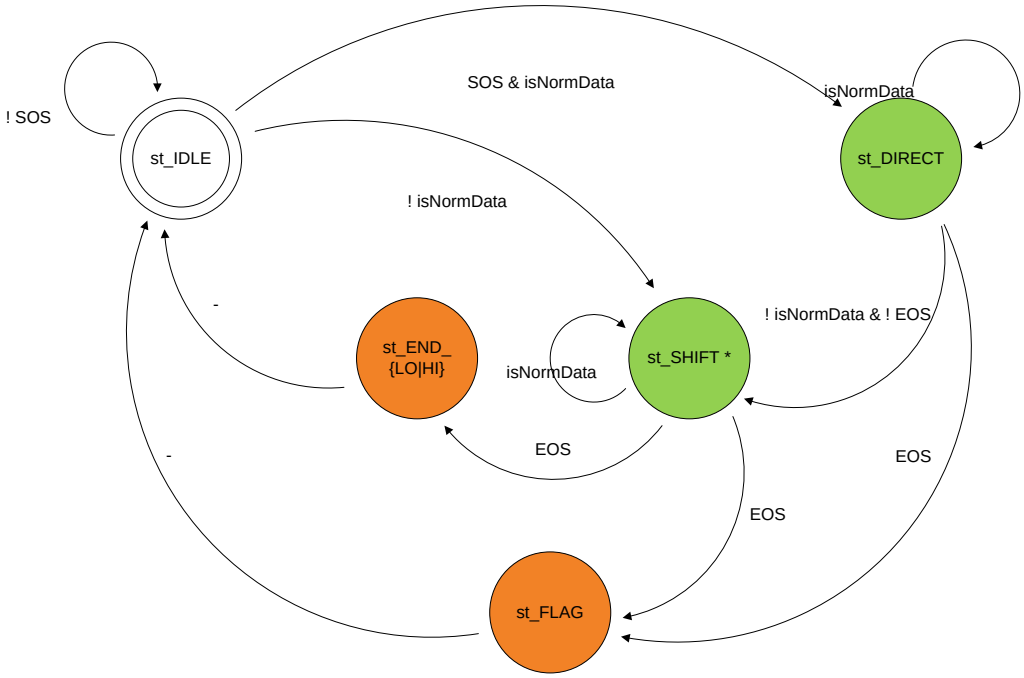
5.2.2 Přijímací část (RX)

Přijímací část je velice podobná té vysílací. Blokové schéma je na obrázku 5.8.
Komponenta „Recieve destuffing FSM“ (`hdlc16_tx_bstuff.vhd`) převádí proud dat na rámec ohraničený signály `SOS` a `EOS`. Je realizována konečným automatem, který je velice podobný automatu v komponentě „Transmit stuffing FSM“. Pokud je přijímán „Flag“ 0x7E7E zůstává automat ve stavu `st_IDLE`. Po přijetí jiného bytu přechází k „destuffingu“ dat a je generován signál „start of stream“ (`SOS`). Jsou nahrazovány byty 0x7D 0x5D bytem 0x7D a 0x7D 0x5E bytem 0x7E. Při dalším přijetí bytu 0x7E, je vygenerován signál „end of stream“ (`EOS`) a dochází k ukončení přenosu.

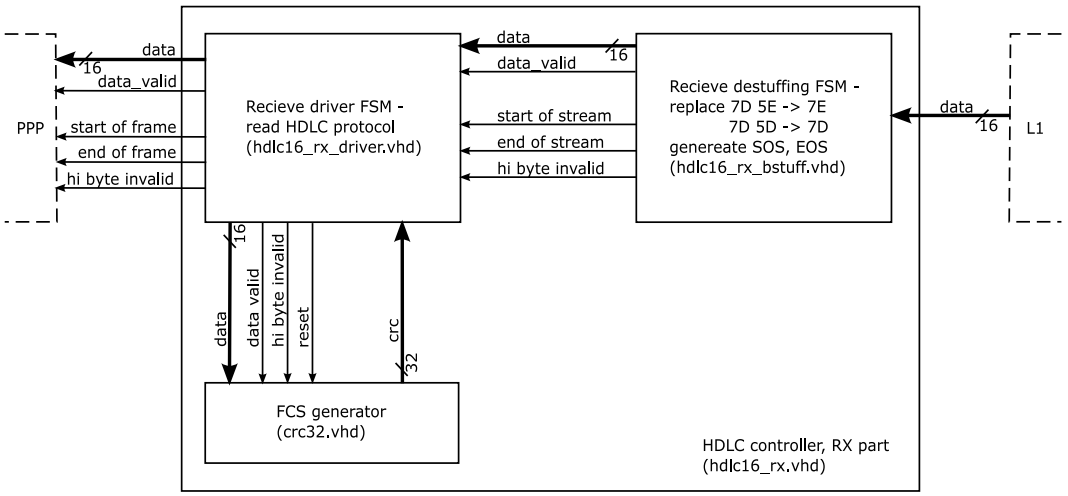
[2]Signály *PPP_SOF* a *PPP_EOF* generuje PPP kontrolér a ohraničují PPP rámec.
[3]HI_INV = hi invalid = platné byty 7:0



Obrázek 5.6: Automat generující HDLC protokol

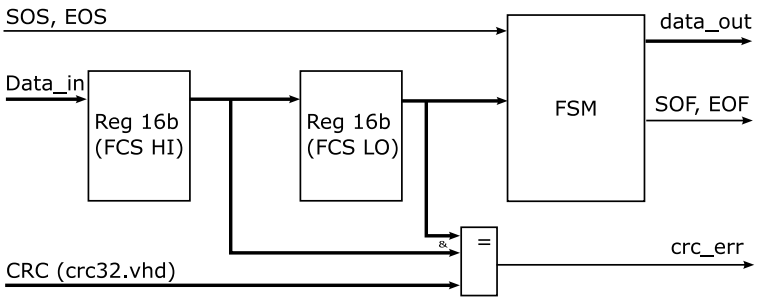


Obrázek 5.7: Automat zajišťující transparentnost dat



Obrázek 5.8: HDLC kontrolér - přijímací část

Následuje komponenta „Recieve driver FSM“ (`hdlc16_rx_driver.vhd`), která extrahuje data z HDLC rámce. Vstup je registrován ve dvou úrovních z důvodu ověření kontrolního součtu (obr. 5.9). Pokud přijde signál „end of stream“ (`EOS = '1'`) je v registrech uložen kontrolní součet. Ten je porovnán s vypočteným kontrolním součtem a přenos je ukončen. Podle výsledku porovnání je nastaven signál `ERR_CRC`.



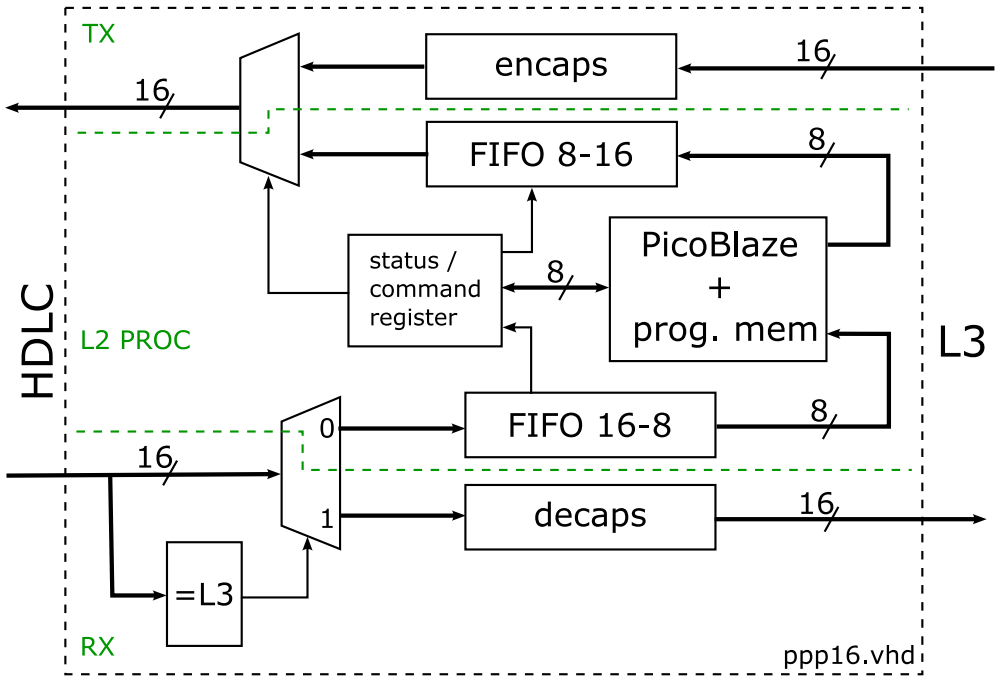
Obrázek 5.9: Komponenta pro dekapsulaci dat z HDLC rámce

5.2.3 Generátor kontrolního součtu

Do HDLC hlavičky je vkládán 32b CRC kontrolní součet, který je na přijímací straně znovu vypočítán a porovnán s údajem v hlavičce. Tímto je snadno detekován poškozený rámec.
Komponenta (`crc32.vhd`) zahrnuje registr pro uložení doposud vypočteného kontrolní součtu. Ten je počítán pomocí xor stromu, který poskytují funkce `nextCRC32_D16` a `nextCRC32_D8`. K vygenerování těchto funkcí byl použit nástroj na stránce `http://www.easics.com` [1]. Výstupem této komponenty je 32b cyklický kód (CRC32), který je opožděn za vstupem o 1 hodinový takt. Druhým výstupem je nejspodnějších 8b CRC32 právě vypočtených. To je vhodné pro vysílací jednotku, kdy je lichý počet vysílaných bytů a kontrolní součet musíme začít vkládat v tomto taktu.

5.3 PPP kontrolér

PPP kontrolér vychází z hrubého návrhu v kapitole 4.4, obrázek 4.4. Blokové schéma je na obrázku 5.10. Zelenou přerušovanou čarou je rozděleno na tři pomyslné části. Vysílací (TX), přijímající (RX) a část (L2 PROC) zpracovávající pakety linkové vrstvy (LCP, NCP, ...).



Obrázek 5.10: PPP kontrolér - blokové schéma

Pro aplikace zaměřené na analýzu provozu, síťovou bezpečnost a jistě mnoho dalších, kde je zpracování nebo vysílání vlastních LCP paketů nežádoucí by postačovala zjednodušená implementace. V té by byly pouze části TX a RX. Z důvodu zachování univerzality řešení je implementován PPP kontrolér podle schématu. Níže je výčet souborů a adresářů, ze kterých se skládá řešení.

`prog/`	program a překladač pro PicoBlaze
`sim/`	simulace (testbench pro ModelSim)
`decaps_ppp_l3.vhd`	dekapsulace paketů síťové vrstvy (OSI L3) z PPP rámce
`embedded_kcpsm3.vhd`	PicoBlaze s pamětí programu
`encaps_l3_ppp.vhd`	enkapsulace OSI L3 do PPP rámců
`fifo_16_8.vhd`	FIFO paměť pro PicoBlaze (16b vstup, 8b výstup)
`fifo_8_16.vhd`	FIFO paměť pro PicoBlaze (8b vstup, 16b výstup)
`kcpsm3.vhd`	PicoBlaze
`ppp16.vhd`	PPP top level

Nyní popíšu jednotlivé bloky schématu, přičemž se přidržím rozdělení na tři části (RX, TX, L2 PROC).

5.3.1 Vysílací část (TX)

Vysílací část je tvořena blokem „encaps“ (`encaps_l3_ppp.vhd`). Obsahuje v sobě vyrovnávací FIFO paměť (implementována na projektu Liberouter) a logiku pro přidání PPP pole „protocol“. Velikost FIFO paměti je možné nastavit genericem ITEMS. Pole „protocol“ se zadává

signálem `PPP_PROTO: std_logic_vector(15 downto 0)` a je tedy možné jej za běhu měnit. V této aplikaci je na pevno nastaveno na hodnotu $0x0021$, což odpovídá IP paketu. Vstupem komponenty je 16b FrameLink a výstupem jsou signály pro HDLC kontrolér.

5.3.2 Přijímací část (RX)

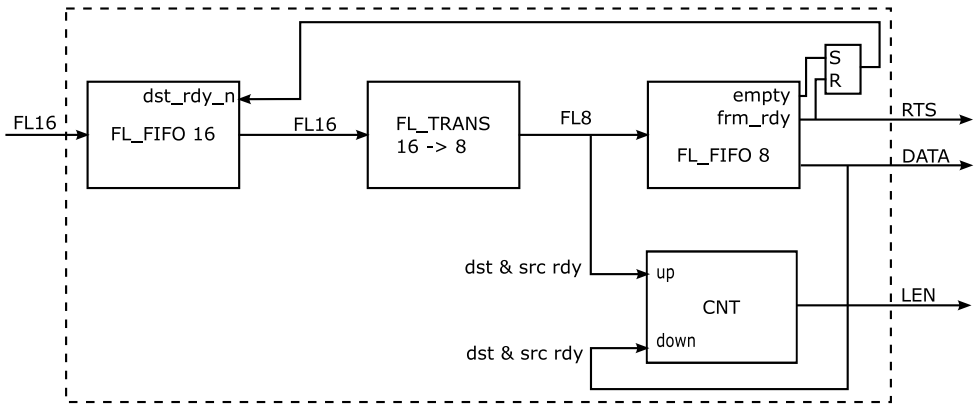
Přijímací část je tvořena blokem „decaps“ (`decaps_ppp_l3.vhd`). Opět v sobě zahrnuje vyrovnávací FIFO paměť a logiku pro odebrání PPP pole „protocol“. Generic `L3_KEEP_PPP` udává, zda má být pole odebráno. Jeho zachování může být v některých aplikacích užitečné. Vstupem jednotky jsou signály z HDLC kontroléru a výstupem je 16b FrameLink.

5.3.3 Zpracování paketů linkové vrstvy (L2 PROC)

Zpracování paketů linkové vrstvy se skládá z několika komponent. Na přijímací straně (od HDLC kontroléru) je multiplexor, který je přepínán komparátorem. Na začátku přenášeného rámce (aktivní signál `SOF`) jsou porovnány horní 4 bity a podle nich je nastaven typ přijímaného rámce, potažmo multiplexor. Typy PPP rámců [21] jsou definované typem `t_pkt_type` a jsou shrnuty v tabluce 5.2. V této implementaci jsou rámce označené `pkt_L3` posílány do jednotky „decaps“, ostatní pakety jsou zpracovány v procesoru.

Hodnota pole „protocol“	Význam	Hodnota `t_pkt_type`
0*** - 3***	Protokol síťové vrstvy (NLP)	pkt_L3
4*** - 7***	Protokol s nízkou zátěží (LVP)	pkt_LVP
8*** - b***	Protokol asociovaný s NLP (NCP)	pkt_NCP
c*** - f***	Řídicí protokol linkové vrstvy (LCP)	pkt_LCP

Tabulka 5.2: Typy a označení PPP rámců



Obrázek 5.11: Komponenta FIFO 16-8

Před procesorem je komponenta nazvaná „FIFO 16-8“ (obrázek 5.11). Skládá se z 16b FL FIFO paměti, převodníku FL16 na FL8, 8b FL FIFO, čítače a několika registrů. Pakety po síti SONET/SDH jsou přenášeny řádově rychleji, než je PicoBlaze dokáže zpracovávat, ale je jich malé množství. Pakety jsou ukládány do FIFO paměti o šířce 16b. Z té jsou vyčítány přes převodník 16b /8b. Do 8b FIFO paměti je načten vždy jen jeden celý paket, poté je nastaven signál „Ready to sent“ (`RTS`), který je zapojen do Status / command registru. Teprve až je celý paket vyčten procesorem, je načten do FIFO paměti další paket. Abychom nemuseli programově

zjišťovat, jestli jsou na vstupu ještě platná data, je zde čítač. Je vhodné jej nejprve načíst a podle něj nastavit počet opakování cyklu, ve kterém FIFO vyčítáme.
Následuje již samotný procesor PicoBlaze s pamětí programu a externím Status / command registrem. Ukázkový program se nachází v podadresáři `prog/`. Podrobněji bude popsán v následující podkapitole věnované programátorskému modelu.
Za procesorem je komponenta „FIFO 8-16“, která je tvořena převodníkem 8b na 16b a 16b FL FIFO paměť. Po nahrání celého paketu procesorem do FIFO paměti je nutné programově nastavit bit `sr_of_rts` ve Status / command registru. To zajistí přepnutí výstupního multiplexoru (pokud je právě přenášen paket síťové vrstvy, je multiplexor přepnut až po přenesení celého paketu) a odvysílání tohoto paketu.

5.3.3.1 Programátorský model

Procesor obsahuje 16 8b univerzálních registrů a 64B paměti. Každá instrukce se provádí 2 takty. Instrukční sada procesoru je popsána v uživatelské příručce [26] a dle mého názoru v praktičtěji napsaném manuálu [8]. S okolím komunikuje pomocí vstupních / výstupních portů a přerušení.
Přerušení je vyvoláno detekcí nosné (nastavuje SONET/SDH Framer). V programu se to projeví odskokem na adresu 0x3FF (konec paměti). Je potřeba, aby na tomto místě byla instrukce `JUMP`, která odskočí do přerušovací rutiny. Z obsluhy přerušení se vrátíme instrukcí `RETURNI`.
Veškerá řídicí komunikace mezi procesorem a okolím je zprostředkována přes „Status / command register“ (SCR), který je dostupný pro čtení a zápis na portu PID_SR. Kromě tohoto registru je k procesoru připojena vstupní a výstupní FIFO paměť, přes které jsou odesílány a přijímány pakety. Pokud je ve vstupní FIFO paměti (port PID_FIN) připraven paket je nastaven bit sr_if_rts v SCR. Informaci o délce paketu je možné získat z čítače na portu PID_FLC. Pokud jej čteme během zpracování paketu, udává počet bytů zbývajících k načtení. Do FIFO paměti je načten nový paket až po úplném vyčtení předchozího. Pokud zapíšeme paket do výstupní FIFO paměti (port PID_FOUT), je poté nutno nastavit bit sr_of_rts v SCR. Tento bit je po odeslání hardwarově nulován. Přehled portů je shrnut v tabulce 5.3 a význam bitů SCR v tabulce 5.4.

Význam	Označení	Hodnota	čtení (R) / zápis (W)
Status / command register	PID_SR	0x01	R/W
Délka paketu ve FIFO z HDLC	PID_FLC	0x02	R
Vstupní FIFO (z HDLC)	PID_FIN	0x04	R
Výstupní FIFO (do HDLC)	PID_FOUT	0x08	W

Tabulka 5.3: V/V porty procesoru PicoBlaze

Význam	Označení	Hodnota	čtení (R) / zápis (W)
Potvrzení funkčnosti fyzické vrstvy	sr_L1_wrk	0x01	W
Povolení přenosu paktů síťové vrstvy	sr_L2_conf	0x02	W
Paket ve výstupním FIFO připraven	sr_of_rts	0x04	W
Vstupní FIFO obsahuje paket	sr_if_rts	0x10	R

Tabulka 5.4: Položky Status / command registru

Nyní krátce popíšu ukázkový program pro procesor PicoBlaze. Program se stará o navázání spojení na linkové vrstvě (implementuje LCP protokol). Po jeho navázání umožní přenos paketů

síťové vrstvy nastavením bitu sr_L2_conf v SCR.
Program se skládá z hlavní smyčky a pomocných rutin pro příjem a odeslání LCP paketu. V hlavní smyčce je povoleno přerušení, v obslužných rutinách je zakázáno. Pokud dojde k přerušení, jsou vynulovány interní registry, je odeslán „LCP Configure-Request“ paket a nastaven příznak o jeho odeslání. V hlavní smyčce je dále realizován time-out čítač (možné realizovat též hardwarově), který je spuštěn odesláním „Request“ paketu a nulován přijetím „Ack“ paketu. Pokud dojde k přetečení time-out čítače, je znovu poslán odpovídající „Request“ paket. Nedílnou součástí hlavní smyčky je načítání SCR. Pokud je zjištěn paket ve vstupní FIFO paměti, je přijat, a podle jeho typu je provedena následná akce. Pokud se jedná o „LCP Configure-Ack“ paket, který odpovídá předchozímu „LCP Configure-Request“ paketu je spojení nakonfigurováno a nastaven bit sr_L2_conf v SCR. Zpravidla příchozí „Request“ paket vyvolá sestavení a odeslání příslušného „Ack“ paketu.
V úvahu připadala rozdílná implementace založená na tabulce přerušení. Při jakékoliv změně „Status / command register“ se vyvolá přerušení. V obsluze přerušení je nutné tento registr načíst a zareagovat na něj příslušnou akcí. Odpadá tím nutnost periodicky vyčítat „Status / command register“ což lze brát za výhodu. Přináší to ovšem problém s maskováním přerušení. Pokud jsme v obsluze přerušení, je vyvolání dalšího přerušení defaultně zakázáno. Tím bychom mohli nějakou událost ztratit. Proto jsem se rozhodl pro implementaci výše popsanou.

5.4 Shrnutí

V tabulce 5.5 jsou shrnuty odhady frekvence po syntéze a využité zdroje FPGA čipu pro jednotku *pos_buf*. Frekvence je uvažována pro FPGA Virtex 2 Pro s označením xc2vpx20-6ff896.

Blok	Frekvence [MHz]	# Slices
hdlc16_tx	162	213
hdlc16_rx	238	193
ppp16	172	3015
sonet_framer	168	1389
ctrl_stat	253	152

Tabulka 5.5: Odhad frekvence po syntéze a využité zdroje na čipu

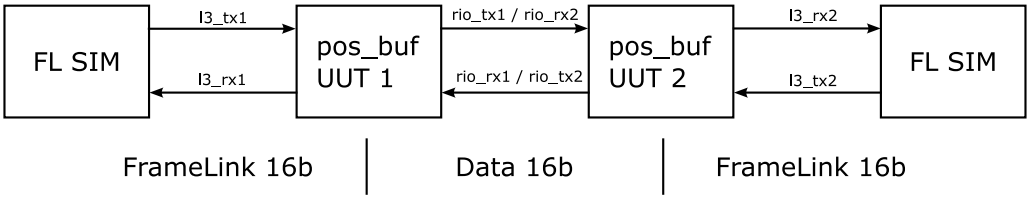
Pro syntézu byl použit nástroj XST od firmy Xilinx. Celá jednotka zabírá 4596 Slices což je 46% tohoto FPGA. Odhad maximální frekvence po syntéze je 126 MHz. Mapování na technologii (MAP) a Place and Route (PaR) úspěšně proběhnou. Počet využitých Slices se po MAP snížil na 3929 (40%) a frekvence se po PaR zvýšila na 136,5 MHz.
Vložíme-li jednotku do připravené „obály“ *pos_buf_rio* s insatncovaným RocketIO-X rozhraním a DCM (Digital Clock Manager), počet obsazených Slices se po MAP zvýší na 4057, ale frekvence po PaR zůstane přibližně stejná (135,6 MHz).
Je patrné, že v obou případech nejsou splněné časové constrainy stanovené na 6,4 ns (156 MHz). V navazující práci bude nutné podrobně analyzovat kritické cesty a navrhnout jejich odstranění.

6 Testování

Zaměřil jsem se pouze na funkční testy celé jednotky *pos_buf*, jejich jednotlivých bloků a vybraných komponent těchto bloků. Veškeré simulace jsou prováděny na úrovni RTL (Register transfer level). V projektu Liberouter je k simulaci používán program ModelSim [16] od firmy Mentor Graphics a není důvod jej také nepoužít. Testování v hardwaru bude možné až po vyřešení problému s časováním RocketIO rozhraní a zvýšení hodinové frekvence odstraněním kritických cest. Nyní popíšu zvlášť simulaci celé jednotky a bloků, z kterých se skládá.

6.1 Simulace celé jednotky

Pro testování celé jednotky *pos_buf* bylo navrženo zapojení na obrázku 6.1, které simuluje jednoduchou síťovou aplikaci. Poskytuje snadné otestování vysílací / přijímací části, ale neodhalí případné nesrovnalosti vzhledem ke specifikaci. K úplnému ověření správnosti jednotky je zapotřebí nahradit jednu UUT (Unit Under Test) jednotkou třetí strany, která je již otestována. Otestovaná jednotka třetí strany však není k dispozici.



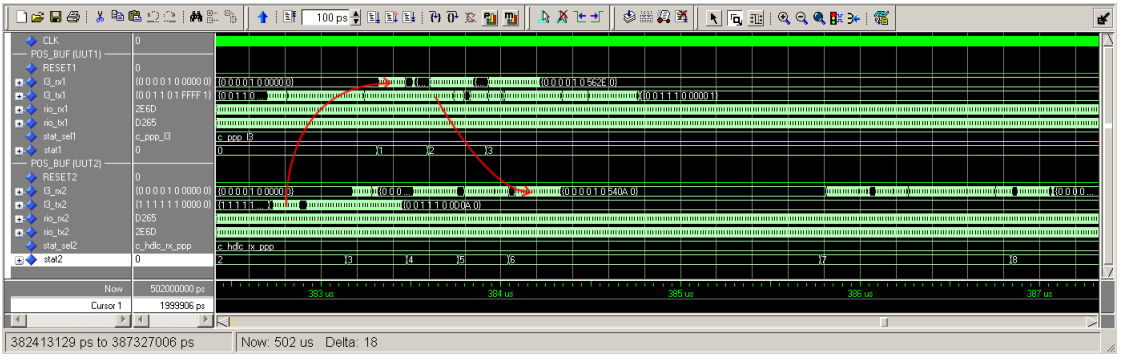
Obrázek 6.1: Zapojení top level testu

Ke generování protokolu FrameLink je použita komponenta `FL_SIM` vytvořená na projektu Liberouter. Pomocí procedury `fl_send32("paket.txt")` se odesílají data uložená v souboru *paket.txt*. Soubor může obsahovat více paketů[1], kde každý paket je ukončen znakem #. Komponenta umožňuje logování přijatých a odeslaných paketů. Pokud chceme pomocí komponenty logovat příchozí data, nesmí je tato komponenta generovat.
Pomocí programu Wireshark [4] bylo odchytnuto několik druhů paketů (ARP, ICMP, TCP, ...) ze živé sítě, které jsou uloženy v adresáři `sim/packets/`. Pro otestování plně duplexního provozu jsou tyto pakety odesílány z obou UUT ve stejném čase. Výsledné datové toky jsou generovány během simulace a jsou uloženy v adresáři `sim/logs/`. Soubor `flow1_tx.txt` odpovídá datům odeslaným jednotkou UUT1 do UUT2. Data přijatá jednotkou UUT2 jsou uložena v souboru `flow1_rx.txt`. Analogická situace je pro opačný směr, (tj. z UUT2 do UUT1) a soubory `flow2_*.txt`. Pro ověření bezchybného přenosu dat, postačuje po doběhnutí simulace (450μs simulačního času) porovnat soubory `flow1_{tx|rx}.txt` a `flow2_{tx|rx}.txt`. Test tedy není zcela automatický, protože soubory musíme porovnat ručně (např. programem *diff*). Průběh simulace se zakreslenými směry přenosu je na obrázku 6.2.

6.2 Simulace jednotlivých bloků

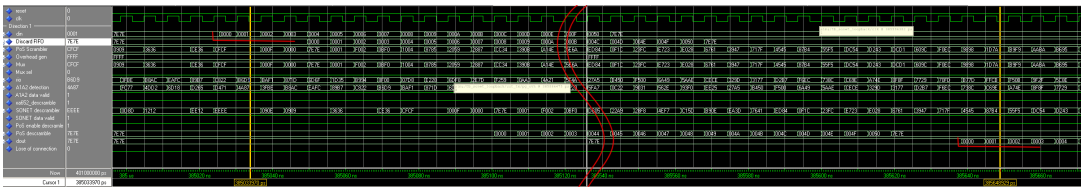
Bloky jednotky *pos_buf* jsem testoval tak, že jsem je vložil do testbenche a jednoduchými modely simuloval okolní komponenty. Testy nejsou automatické, spoléhají na manuální zkontrolování průběhů signálů vzhledem ke specifikaci.
Na následujícím obrázku 6.3 je patrná simulace komponenty SONET/SDH Framer. Komponenta začíná přenášet data (signál `LOC` je nastaven na nulu) až po synchronizaci s protější stra-

[1] Chápáno paketů síťový vrstvy, v terminologii FL protokolu označeno jako „Frame“



Obrázek 6.2: Průběh simulace jednotky pos_buf (ModelSim)

nou. K té dochází, pokud je třikrát (možné nastavit v souboru `xapp652_oc48_aligner_16.vhd`) přijat SONET/SDH rámce s byty A1A2 přesně po 125us. V tomto testu dochází k přenosu dat přibližně po 385us. Bloky přenášených data jsou generována v cyklu a jsou od sebe odděleny „flagem“ 0x7E podobně jako HDLC rámec, který mají představovat. Blok dat je dlouhý 180 slov a za ním následuje 50 slov „flagu“ 0x7E7E. Tento cyklus se opakuje 50krát. Tím je ověřeno správné pozastavení dat v době vkládání SONET hlavičky. Začátek vysílaných a přijímaných dat je označen červenou barvou. Ze simulace (případně obrázku) je patrné, že přijatá data odpovídají vyslaným. Do okna „wave“ jsou vytaženy veškeré datové cesty po průchodu jednotlivými bloky. Je z nich dobře patrné ukládání ve vyrovnávacích FIFO pamětech, kódování a dekódování apod.



Obrázek 6.3: Průběh simulace SONET/SDH Frameru (ModelSim)

U HDLC kontroléru je k dispozici několik testů. Lze testovat zvlášť vysílací a přijímací část a obě části společně zapojené za sebe. Pomocí vstupních dat jsou testovány různé kombinace (pozice) bytů 7E a 7D. Průběh testu vypadá podobně jako v předcházejícím případě a proto zde neuvádím screenshot. Doporučuji však si prohlédnout živou simulaci. Nachází se v adresáři `hdlc_ctrl/sim/` a spustíme ji příkazem `vsim -do hdlc16_loopback_tb.do`.
Testování PPP kontroléru se zaměřuje na otestování otevření spojení na linkové vrstvě (zpracování a generování LCP paketů) a přenos samotných paketů síťové vrstvy. Je testována enkapsulace a dekapsulace paketů síťové vrstvy do/z PPP rámce.
Testování komponent, ze kterých se výše popsané bloky skládají, nebudu popisovat, protože se jedná o poměrně jednoduché testbenche, které mají dobrou vypovídající schopnost.

6.3 Spouštění simulací

Simulace je možné spouštět přímo z prostředí ISE. Je nutné vybrat položku „Sources for: Behavioral Simulation“, která je nad seznamem souborů projektu. Nyní je možné v tomto okně zvolit simulaci požadované jednotky. Poté v okně „Processes“ spustit „Simulate Behavioral Model“.

Druhou možností, jak simulace spouštět je příkazová řádka. U každé komponenty je adresář `sim/`, ve kterém jsou testbenche a skripty spouštějící simulaci. Samotné spuštění provedeme příkazem `vsim -do soubor.do`. Skript zajistí kompilaci potřebných souborů do knihovny *work*, přidání signálů do okna „wave“ a běh samotné simulace.

7 Závěr

Výsledkem práce je jednotka *pos_buf* pro obousměrný převod protokolu Packet over SONET (PoS) na FrameLink (FL). Je popsaná v jazyce VHDL a určena pro obvod Xilinx Virtex 2 Pro. Jednotka poskytuje 16b rozhraní pro připojení k SONET/SDH síti prostřednictvím RocketIO-X. Rozhraní RocketIO-X není implementováno přímo v této jednotce, ale je součástí „obálky" *pos_buf_rio*, do které je jednotka *pos_buf* společně s komponentou měnící datovou šířku (RocketIO 20b / *pos_buf* 16b) vložena. Důvodem pro takovouto implementaci je oddělení platformě závislé a nezávislé části a tím je umožněno snazší a efektivnější testování samotného převodu PoS na FL.
Vzhledem k chybějícímu krystalu (nebo násobičce) na kartě SFPRO a následné nemožnosti převodu 20b výstupu RocketIO-X nebylo možné kompletní odladění jednotky v reálném provozu. Posyntezní odhad a výsledky po PaR ukazují, že pro nasazení v síti SONET OC-48 bude ještě pravděpodobně nutné důkladně analyzovat kritické cesty a posléze je odstranit, abychom dosáhli frekvence minimálně 156 MHz.
Jednotka *pos_buf* je navržena flexibilně a je ji možné použít s poměrně malými úpravami pro různé aplikace. Např. pro použití při monitorování sítě stačí vypustit z PPP kontroléru blok pro zpracování paketů linkové vrstvy (viz. kapitola 5.3).
Přínos práce vidím ve shrnutí požadavků a kroků nutných k implementaci protokolu Packet over SONET a analýze dostupných řešení využitelných v FPGA. Ačkoli nasazení v reálném provozu není z popsaných důvodů v tomto stádiu možné, můžeme využít jednotlivé bloky jednotky v dalších projektech. Velkým přínosem práce je implementace nativního 16b HDLC kontroléru. Oproti dostupným 8b řešením poskytuje dvojnásobnou propustnost a tím umožňuje zpracování vyšších rychlostí.

8 Literatura

[1] Generator of synthesizable CRC functions.
URL `http://www.easics.com/webtools/crctool` 5.2.3

[2] Liberouter project.
URL `http://www.liberouter.org` 2.3, 2.3.1, 4.2.5, 5.1.1, A.1

[3] LocalLink User Interface.
URL `http://www.xilinx.com/products/ipcenter/LocalLink_UserInterface.htm` 2.4

[4] Wireshark Interactively dump and analyze network traffic, Manual pages.
URL `http://www.wireshark.org` 6.1

[5] Altera: *SONET/SDH Compiler User Guide*. 10 2002.
URL `www.altera.com` 3.1.1

[6] Calyptech: *CORE-PM-10GB OC-192 / OC-48 / OC-12 / GbE / FC Multi-Rate Performance Monitor*. 2004.
URL `http://www.calyptech.com` 3.1.3

[7] Calyptech: *CORE-PM-2G5 OC-48 / OC-12 / OC-3 / GbE / FC Multi-Rate Performance Monitor*. 2004.
URL `http://www.calyptech.com` 3.1.3

[8] Ken Chapman: *KCPSM3 8-bit Microcontroller*. Xilinx, 2003. 4.4, 5.3.3.1

[9] Conexant: Dual SONET/SDH Framer and Mapper OC-48 POS/ATM (Roshni PX-4805). 3.3.1

[10] Chia (Janet) Wu Daniel Lee: 6.195 Final Project: Byte-wide High-level Data Link Control Protocol Controller.
URL `http://sunpal7.mit.edu/6.195/f97/omega.janetwu/project2/project2.html` 3.2.1, 5.2

[11] Amit Dhir: HDLC Controller Solutions with Spartan-II FPGAs. Technická zpráva, Xilinx, 2000. 3.2.3

[12] Jirí Douša: *VHDL (jazyk pro simulaci a syntézu císlicových obvodu)*. 5

[13] Exar: OC-48 ATM/UNI/POS/Mapper IC (XRT95L51). 3.3.2, 3.3.2

[14] ITU-T G.707/Y.1322: *Network node interface for the synchronous digital hierarchy (SDH)*. 2000. 2.1, 2.1, 2.1.1, 2.1.3, 3.1.1, 4.2.3

[15] Jan Janeček: Přenosové technologie IPv4 - POS, Slidy k přednášce předmětu X36MTI, CVUT FEL.
URL `http://dsn.felk.cvut.cz/education.cz/X36MTI/prednasky.html` 2.2

[16] Mentor Graphics: *ModelSim SE User's Manual, version 6.1b*. 2005. 6

[17] Nick Sawyer: SONET and OTN Scramblers/Descramblers. Technická zpráva, Xilinx, 2002. 3.1.4.2

[18] Nick Sawyer: Word Alignment and SONET/SDH Deframing. Technická zpráva, Xilinx, 2004. 3.1.4.3

[19] Nick Sawyer a Francesco Contu: SONET Rate Conversion in Virtex-II Pro Devices. Technická zpráva, Xilinx, 2002. 3.1.4.1, 3.1.4.1, 4.2.1

[20] W. Simpson: *RFC 1619: PPP over SONET/SDH*. 1994. 2.2

[21] W. Simpson: *RFC 1661: The Point-to-Point Protocol (PPP)*. 1994. 2.2.1, 2.2.1.2, 5.3.3

[22] W. Simpson: *RFC 1662: PPP in HDLC-like Framing*. 1994. 2.2.2

[23] W. Simpson: *RFC 2615: PPP over SONET/SDH*. 1999. 2.1.2, 2.2, 3, 3.2.2, 3.2.3, 4.2.4, 5.1.5

[24] Xelic: *XCS48F - SONET/SDH Transport Processor Core*.
URL `www.xelic.com` 3.1.2

[25] Xilinx: *RocketIO X Transceiver User Guide*. 2004. 3.1.4.4, 4.2.1

[26] Xilinx: *PicoBlaze 8-bit Embedded Microcontroller User Guide*. 2005. 4.4, 5.3.3.1

A Protokol FrameLink

A.1 Úvod

FrameLink (FL) je protokol pro přenos dat ve formě paketů v rámci designů projektu Liberouter. FL nahrazuje doposud používaný command protokol, protože ten je při vyšších datových šířkách značně neefektivní. FL je inspirován protokolem LocalLink od Xilinxu ale obsahuje závažné změny (viz dále). Celý text je převzat z privátní wiki stránky [2] projektu liberouter.

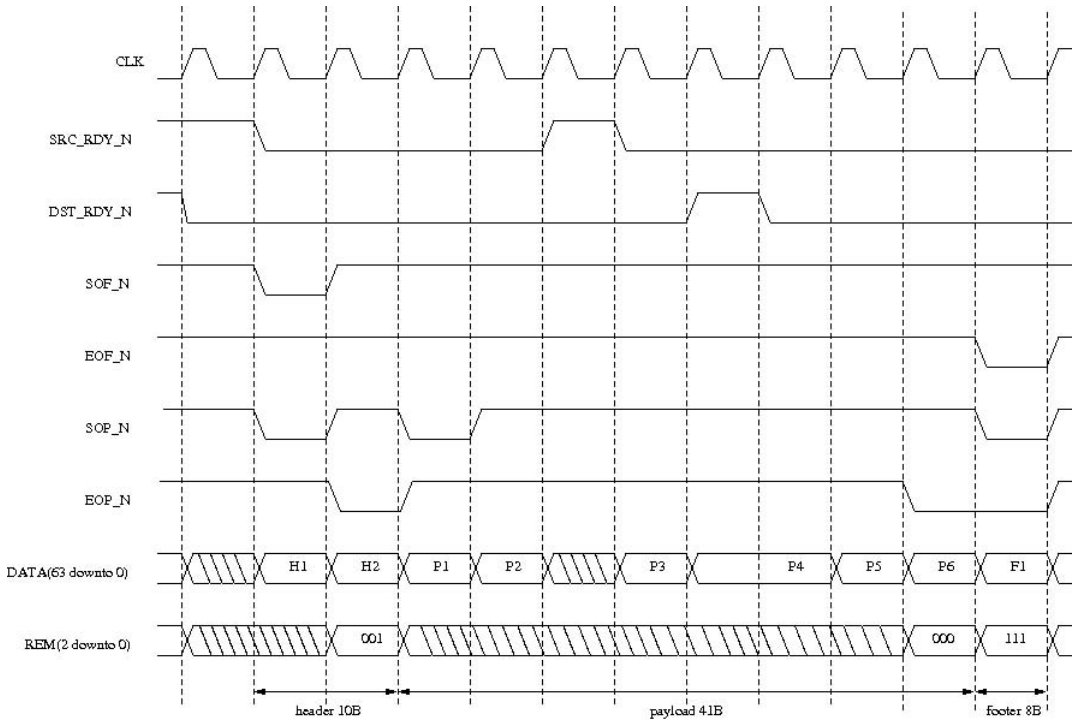
A.2 Rozhraní protokolu

- DATA - přenášená data o šířce N*8 bitu
- REM - signál určuje kolik bytů z datové cesty je platných. Signál je binárně zakódován, takže jeho šířka je log(N). Signál je platný pokud je aktivní signál EOP_N - tzn. je konec bloku a zajímá nás kolik bytů v tomto posledním slově je platných, přičemž se počítá zpráva a od nuly (little endian, stejně jako CMD protokol). Hodnota REM ukazuje na poslední platný byte. Např. pro DATA o šířce 32 bitů má REM šířku 2 bity. Pokud končí blok dat a REM je "00" pak platný je pouze první byte - tedy 7 downto 0, "01" pak platné jsou první 2 byty, "11" pak platné jsou všechny byty. Více viz specifikace LocalLinku, kde je REM řešen stejně (Pozor! LocalLinku používá big Indian).
- SOF_N - start of frame, aktivní v nule. Ohraničuje začátek celého paketu, ten se pak muže skládat z více částí, které jsou ohraničeny pomocí SOP_N a EOP_N. Zároveň s SOF_N je vždy nastaven SOP_N v první části paketu.
- EOF_N - end of frame, aktivní v nule. Ukončuje paket, muže se vyskytovat zároveň se SOF_N (pak má paket méně než N bytů). Zároveň s ním je vždy nastaven EOP_N v poslední částí paketu.
- SOP_N - začátek části paketu, aktivní v nule. Částí muže být v paketu libovolné množství ale typicky to budou dvě nebo tři.
- EOP_N - konec části paketu, aktivní v nule. Valid pro signál REM. Může se vyskytovat zároveň se SOP_N pokud je daná část menší než šířka dat.
- SRC_RDY_N - source ready, aktivní v nule. Zařízení, které vysílá data, tímto signalizuje připravenost odesílat data (má nějaké data na odesílání). Data jsou připravena na výstupu.
- DST_RDY_N - destination ready, aktivní v nule. Zařízení přijímající data tímto signalizuje schopnost přijmout data. Přenos probíhá, jen pokud jsou SRC_RDY_N i DST_RDY_N aktivní.

A.3 Struktura paketu

- celý paket je ohraničen signály SOF_N a EOF_N
- paket se pak muže skládat z libovolného počtu části ohraničených signaly SOP_N a EOP_N
- každá část je zarovnaná, tedy při SOP_N je vždy první byte na první pozici (je použito schéma little endian)

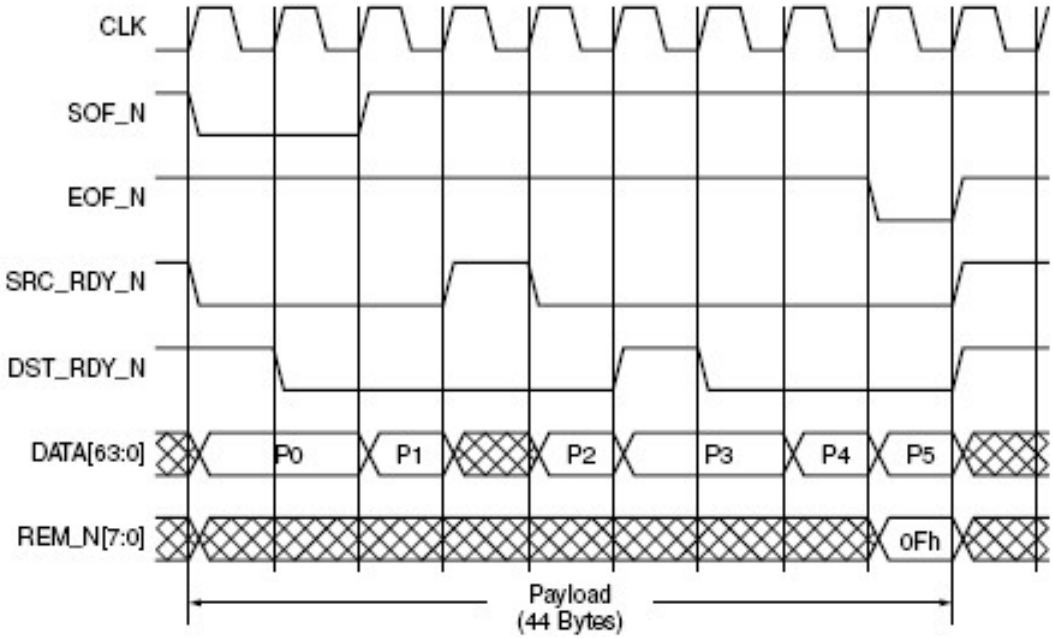
- v jednom slově dat tedy nemůžou být přenášeny dvě různé části jako v LocalLinku
- paket se typicky bude skládat až ze tří částí, kterými budou HEADER, PAYLOAD a FOOTER
- libovolná část muže být v konkrétním projektu vynechána, vznikne tak např. jenom HEADER a PAYLOAD
- bude také možné počet částí paketu zvýšit, to se ale nepředpokládá
- která část je která není explicitně uvedeno, je to dáno kontextem (pořadí časti v celém paketu)
- pořadí bytů je dáno dle uspořádání Little endian, tedy v 32 bitovém slově (31 downto 0) je první byte (7 downto 0) pak (15 downto 8) atd
- možný průběh komunikace (A.1) pokud se paket skládá ze tří časti



Obrázek A.1: Možný průběh komunikace

A.4 Přenos dat

Přenos je řízen signály SRC_RDY_N a DST_RDY_N stejně jako u LocalLinku. Ve zkratce: signály znamenají připravenost komponenty vysílat/přijímat data. Vlastní přenos pak probíhá, pouze pokud jsou oba signály aktivní. U komponenty, která data odesílá, se navíc předpokládá, že pokud bude mít nějaká data na odeslání, tak je sama aktivně připraví na výstup (bez nějakého signálu read) ale na další data přejde až po potvrzení připravenosti přijímat z druhé strany. Příklad komunikace A.2



Obrázek A.2: Příklad komunikace

A.5 Rozdíly proti LocalLinku

- data jsou vždy zarovnaná, v jednom slově nemůžou být dvě části paketu
- SOP_N a EOP_N ohraničuji každou část paketu
- chybí explicitní informace, ve které části paketu jsme, je to dáno pořadím. Nicméně komponenta bude od tohoto odstíněna použitím generického dekodéru viz dále
- používá se uspořádání bytů little endian (kvůli uložení v BlockRAM, přístupu do SW), LocalLink používá Big Endian

A.6 Rozhraní komponent s FrameLinkem

Pokud má komponenta jen jedno FL rozhraní (SW_RX/TX_BUFFER, IBUF, OBUF) jsou možnosti jmen následující:

- dle rozhraní protokolu viz výše, s tím rozdílem že místo REM se použije DREM (data remainder) - REM není možné použít, protože je to operátor VHDL
  - SOF_N
  - SOP_N
  - EOP_N
  - EOF_N
  - SRC_RDY_N
  - DST_RDY_N
  - DATA
  - DREM

- použije se předpona TX nebo RX podle toho zda se jedná o vstupní nebo výstupní rozhraní:
  - RX_SOF_N
  - RX_SOP_N
  - RX_EOP_N
  - RX_EOF_N
  - RX_SRC_RDY_N
  - RX_DST_RDY_N
  - RX_DATA
  - RX_REM

Pokud má komponente vstupní i výstupní rozhraní FL protokolu tak se použije předpona RX resp. TX viz výše.

B Seznam použitých zkratek

2D Two-Dimensional

AIS Alarm Indication Signal

ANSI American National Standards Institute

APS Automatic Protection Switch

ASIC Application-specific Integrated Circuit

ATM Asynchronous Transfer Mode

AU Administrative Unit

BIP Bit Interleaved Parity

CD Carrier Detection

CPU Central Processing Unit

CRC Cyclic Redundancy Check

DCCM Data Communication Channel Multiplex Section

DCCR Data Communication Channel Regenerator Section

DCM Digital Clock Manager

DS Digital Signal

EDIF Electronic Design Interchange Format

EEPROM Electrically Erasable Programmable Read-Only Memory

EOP End Of Packet

FCS Frame Check Sequence

FIFO First In First Out

FL Frame link

FPGA Field Programmable Gate Array

FSM Finite State Machine

GFP Generic Framing Procedure

HDLC High-level Data Link Control

CHAP Challenge-Handshake Authentication Protocol

IDS Intrusion Detection System

IEEE Institute of Electrical and Electronics Engineers

IFC Interface

IP Internet Protocol

IPX Internetwork Packet Exchange

ISDN Integrated Services Digital Network

ITU-T International Telecommunication Union - Telecommunication

LCP Link Control Protocol

LL Local link

LOF Loss Of Frame

LOH Line overhead

LOS Loss of Signal

LVDS Low Voltage Differential Signal

LVPECL Low Voltage Positive Emitter Coupled Logic

LVP Low Volume traffic Protocols

MIT Massachusetts Institute of Technology

MRU Maximum Receive Unit

MSOH Multiplex Section Overhead

MS Multiplex Section

NCP Network Control Protocol

NLP Network-Layer Protocol

OC Optical Carrier

OOF Out Of Frame

OSI Open Systems Interconnection

PAP Password Authentication Procedure

PaR Place and Route

PBGA Plastic Ball Grid Array

PCB Printed Circuit Board

PCI Peripheral Component Interconnect

POH Path Overhead

POS Packet Over SONET

POS-PHY Packet over SONET - Physical layer Protocol

PPP Point to Point Protocol

RDI Remote Defect Indication

REI Remote Error Indication

RFC Request For Comments

RJ Registered Jack

RPR Resilient Packet Rings

RSOH Regenerator Section Overhead

RTS Ready To Sent

RX Receive

SCAMPI Scaleable Monitoring Platform for the Internet

SDH Synchronous Digital Hierarchy

SFP Small Form-Factor Pluggable

SOH Section Overhead

SONET Synchronous Optical Network

SOP Start of Packet

SPE Synchronous Payload Envelope

SSRAM Synchronous Static Random Access Memory

STM Synchronous Transport Module

STS Synchronous Transport Signal

SW Software

TBGA Tape Ball Grid Array

TOH Transport Overhead

TX Transmit

UUT Unit Under Test

VHDL VHSIC (very high speed integrated circuits) hardware description language

VT Virtual Tributaries

VC Virtual Container

XAPP Xilinx Application Note

XFP 10 Gigabit Small Form Factor Pluggable

C Obsah přiloženého CD

Součástí této práce je přiložené CD, které obsahuje realizovanou jednotku *pos_buf* a tento dokument. Adresářová struktura CD je následující:

readme.txt - soubor readme

src - adresář se zdrojovými kódy jednotky

- **_ise** - projekt pro nástroj Xilinx ISE
- **comp** - komponenty jednotky
- **coregen** - vygenerované soubory pomoci Xilinx Core generator
- **pkg** - globální VHDL package
- **sim** - simulace celé jednotky
- **pos_buf.vhd** - top_level entita
- **pos_buf_rio.vhd** - top_level entita propojená s RocketIO

text - text diplomové práce

- **dipl.pdf** - výsledný tištitelný formát nezávislý na systému
- **src** - zdrojové soubory pro LaTeX