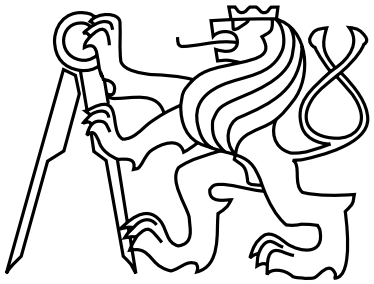
Na tomto místě bude oficiální zadání vaší práce

- Toto zadání je podepsané děkanem a vedoucím katedry,
- musíte si ho vyzvednout na studijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Diplomová práce

Rychlé ethernetové jádro

Bc. Jiří Dostál

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný, Navazující magisterský

Obor: Výpočetní technika

13. května 2011

Poděkování

Tímto bych chtěl poděkovat panu Ing. Pavlu Kubalíkovi, Ph.D., vedoucímu diplomové práce, za jeho čas a cenné rady. Dále pak děkuji všem, kteří mi pomohli k zdárnému dokončení práce.

Prohlášení

Prohlašuji, že jsem 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 Praze dne 13. 5. 2011 .....

Abstract

This thesis deals with the developement of the fast ethernet core (Medium Access Control layer), which operates at 10, 100 and 1000 Mbps. The core is written in the VHDL language and the main goal is low latency, space requirements and symplicity. Further usage of this score is in school projects instead of using commercial one (e. g. TEMAC). Because of previous reasons, there are implemented a subset of the IEEE 802.3 standard, which provides a functionality at 10, 100 and 1000 Mbps speeds with full duplex. Work also describes way of the implementation, testing and possibilities of usage.

Abstrakt

Tato diplomová práce se zabývá vývojem rychlého ethernetového jádra (vrstva Medium Access Control), pracujícím na rychlostech 10, 100 a 1000 Mbps. Jádro je psáno v jazyce VHDL a při implementaci je kladen důraz na jednoduchost, nízkou latenci a na malé prostorové nároky. S využitím jádra se počítá v dalších projektech, ve kterých by muselo být jinak použito komerční řešení TEMAC. Z předešlých důvodů je implementována pouze podmnožina standardu IEEE 802.3, která zajištuje funkcionalitu na rychlostech 10, 100 a 1000 Mbps v plně duplexním režimu. Práce nadále pojednává o způsobu implementace, testování a možnostech využití.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V posledním desetiletí můžeme sledovat mohutný rozvoj a diverzifikaci síťových technologií, propojování počítačových sítí či jednotlivých komunikačních bodů. Již klasickým příkladem je celosvětová počítačová síť Internet, která propojuje počítače z globálního hlediska, jejíž název se mj. stal synonymem pro uživatelsky nejvíce využívanou službu World Wide Web. Nastupuje také důležitost rychlých lokálních sítí (LAN), které slouží k připojené koncových uživatelů zejména v podnikové a akademické sféře. V této sféře se nejvíce rozšířila síť typu Ethernet [1] (IEEE 802.3), která vytlačila ostatní do té doby používané technologie. Její obliba v poslední době vzrostla natolik, že se používá jako základní přenosová technologie i na datových sítí vyššího geografického rozsahu (CAN, MAN). Ethernetovým rozhraním začínají být vybavovány i výrobky spotřební elektroniky (TV set-top boxy apod.).

Má diplomová práce je věnována právě návrhu ethernetového rozhraní, a to hardwarového jádra, které zajišťuje funkcionalitu podvrstvy řízení přístupu k médiu (MAC). Jádro bude (dle oficiálního zadání) pracovat na přenosových rychlostech 100 a 1000 Mbps; pro jeho implementaci bude použit jazyk VHDL.

V následujících kapitolách se budu nejprve zabývat existujícími podobnými řešeními a důvody, které vedly k návrhu této jednotky. Následně se budu věnovat analýze technologie CSMA/CD a linkové vrstvy sítě typu Ethernet. Zde se zaměřím hlavně na strukturu ethernetového rámce, základní datové jednotky této vrstvy, a na funkční strukturu klienta podvrstvy MAC. Dále bude popsáno vlastní HW řešení, které bude implementováno pro obvod FPGA pomocí jazyka VHDL. V části testování popíši jednotlivé testy používané při verifikaci jádra a shrnu problémy, které při samostatné realizaci vyvstaly.

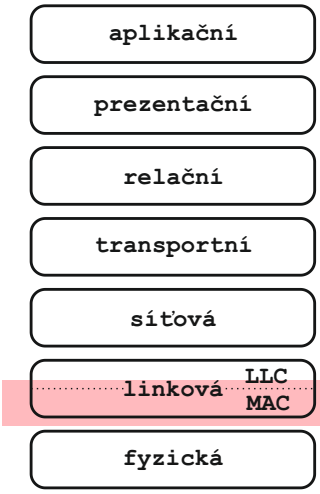
Jako vedlejší produkt tohoto digitálního návrhu vnikla také funkcionalita jádra na nejnižší přenosové rychlosti 10 Mbps a také jednoduchá jednotka správy rozhraní MII.

Kapitola 2

Popis problému, specifikace cíle

V této kapitole popíši hlavní existující nekomerční i komerční řešení ethernetových jader, jejich výhody a nevýhody. Dále pak specifikuji cíl práce a závěrem vyjmenuji možné budoucí využití řešeného jádra.

2.1 Ethernetové jádro



Obrázek 2.1: Vrstvy modelu OSI. Barevně je vymezeno MAC jádro.

Ethernetovým jádrem se v následujícím textu rozumí klient podvrstvy MAC, podle specifikace IEEE 802.3 [1]. Hlavním úkolem tohoto jádra je řízení přístupu k médiu, zapouzdřování dat a s tím spojených operací. Jádro je na jedné straně spojeno s fyzickou vrstvou Ethernetu, která má na starost fyzickou signalizaci na médiu, a na straně druhé s klientem vyšší vrstvy, kterému dodává přijímaná data a obdržená data odvysílá. Umístění MAC jádra v hierarchii OSI je uvedeno na obrázku 2.1.

MAC podvrstva bývá v současné době řešena velice často softwarově, jako součást driveru přídavné karty pro PC, či např. jako obslužný program v mikrokontrolérových aplikací. Toto

řešení sebou nese řadu nevýhod – asi nejznatelnější je zbytečné zabírání času procesoru při I/O operacích na fyzické vrstvě Ethernetu. Výhodou je samozřejmě nižší cena, která je dána absencí obvodového řešení přímo na čipu. Tato práce se věnuje naopak hardwarovému řešení, které s sebou přináší řadu výhod – a to především výkonostních.

Dalším problémem je jazyk pro popis harwaru (HDL) použitý k implementace. V době psaní této práce neexistovalo volně dostupné jádro napsané v jazyce VHDL, který se mj. výhradně používá při školních projektech a při výuce. Volně dostupná řešení jsou napsána v jazyku Verilog.

2.2 Existující řešení

Před zahájením prací na návrhu nového jádra jsem prozkoumal stávající podobná řešení – ať již volně dostupná či licencovaná. Z velkého množství projektů mě zaujal webový portál OpenCores.org a komerčně dostupné IP Core do Firmy Xilinx, jejichž obvody FPGA využíváme při školních pracech.

2.2.1 OpenCores.org

Obrázek 2.2: Webový komunitní portál OpenCores.org

OpenCores.org [10] je webový portál, který sdružuje největší celosvětovou komunitu zabývající se digitálním návrhem open-source hardwaru. Jedná se o obdobu podobných a daleko více početných sdružení pro vývoj svobodného open-source softwaru. Koncem roku 2010 dosáhl počet registrovných uživatelů OpenCores hodnoty 95 000 a počet uveřejněných projektů se pohyboval kolem čísla 800.

Projekty týkající se Ethernetu jsou k nalezení v sekci „Communication controller“, těch, které se týkají implementaci MAC klienta je překvapivě málo (čtyři, z toho pouze jeden pro rychlost 1000 Mbps).

2.2.1.1 Ethernet MAC 10/100 Mbps

„Ethernet MAC 10/100 Mbps“ – jedná se, jak už sám název napovídá, o Ethernetové jádro pro rychlosti 10/100 Mbps. Projekt je ve stavu production/stable. Velikost jádra je cca 28 000 ekvivalentních hradel, kód je psán v jazyce Verilog. K projektu je dostupná bohatá dokumentace, kód je kompletní včetně testbench. Jádro je na jedné straně připojeno na MII, s klientem vyšší vrstvy je připojeno pomocí sběrnice Wishbone. K dispozici je také obvod pro správu fyzické vrstvy.

Výhody:

- bohatá dokumentace
- CVS repository
- vývoj ukončen, stabilní stav
- modulární struktura, řídí se doporučeními z IEEE 802.3

Nevýhody:

- psáno v jazyku Verilog
- podporuje pouze rychlosti 10/100 Mbps

2.2.1.2 10_100_1000 Mbps tri-mode ethernet MAC

„10_100_1000 Mbps tri-mode ethernet MAC“ – jedná se o obdobný projekt, avšak rozšířený o rychlost 1000 Mbps. Je opět psán v jazyce Verilog. Autoři slibují velikost implementace pod 2000 logických buněk při zachování plné funkcionality. Projekt je šířen pod licencí LGPL. Ke stažení je opět celý kód a specifikace ve stádiu draft. Projekt obsahuje také verifikaci, jádro bylo testováno pro příjem mezních velikostí rámců na všech implementovaných rychlostech.

Výhody:

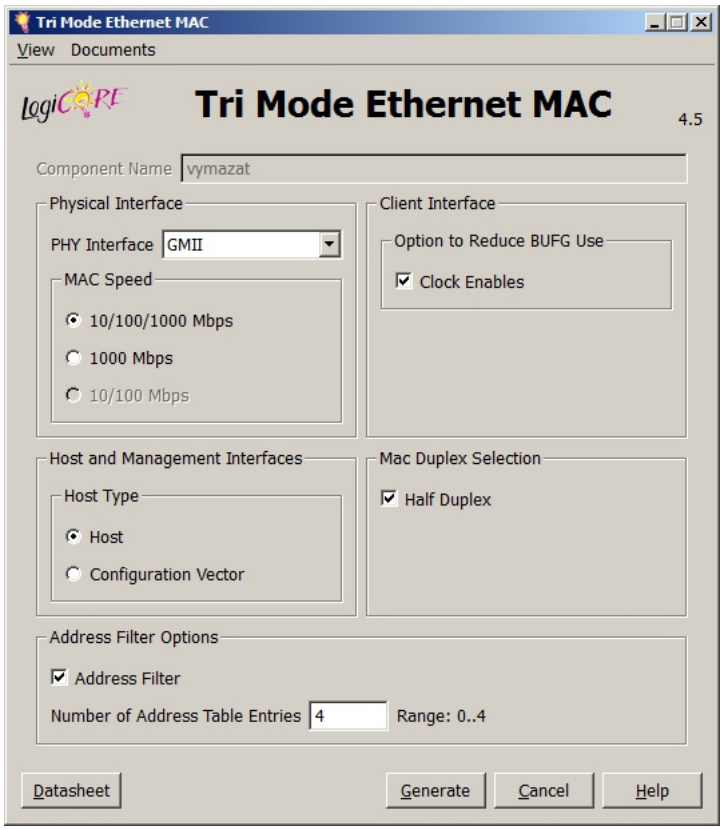
- rychlosti 10, 100 a 1000 Mbps
- včetně half-duplex režimu pro rychlosti 10 a 100 Mbps
- kompaktní velikost
- podpora jumbo rámců
- možnost jednoduché uživatelské konfigurace FIFO

Nevýhody:

- psáno v jazyku Verilog
- nedostatečná dokumentace

2.2.2 Xilinx

Firma Xilinx je přední výrobce programovatelných logických obvodů. Ve svých vývojových prostředích nabízí produkty pod značkou LogiCORE IP Cores – některá jsou dostupná pouze komerčně. Do této kategorie bohužel spadá zdařilé jádro „Tri-Mode Ethernet MAC“.



Obrázek 2.3: Konfigurační dialog jádra Tri-Mode Ethernet MAC

2.2.2.1 Tri-Mode Ethernet MAC

„Tri-Mode Ethernet MAC“ – jedná se o IP Core licencované pro syntézu. Jádro pracuje na všech rychlostech, je dispozici velice přehledný a bohatý konfigurační průvodce. Nelicencované jádro se dá plně odsimulovat, nechybí i ukázkový návrh. K jádru je dodávána i velice bohatá dokumentace, která popisuje jednotlivé funcionality. Pro toto jádru jsou přizpůsobeny i vyšší vrstvy a napojení na procesor MicroBlaze v EDK.

Výhody:

- odpovídá normě IEEE 802.3–2008
- rychlosti 10, 100 a 1000 Mbps
- poloviční i plný duplex
- konfigurovatelné rozhraní fyzické vrstvy (GMII/RGMII/SGMII)
- MDIO rozhraní a správa fyzické vrstvy
- rozšíření o zpracovávání .1Q VLAN

Nevýhody:

- dostupné po zakoupení licence

2.3 Specifikace cíle

Cílem mé práce bude navrhnout hardwarové MAC jádro. Po prostudování problematiky a existujících řešení jsem vzal v potaz všechny silné stránky jednotlivých jader a snažil jsem poté sestavit specifikaci, jak by mělo nové jádro vypadat. Část je již obsažena v oficiálním zadání diplomové práce.

Specifikace:

- HW implementace MAC podvrstvy
- provoz na rychlostech 10, 100 a 1000 Mbps
- podpora pouze plně duplexního přenosu
- rozhraní fyzické vrstvy GMII
- rozhraní klienta MAC kompatibilní s řešením Xilinx TEMAC
- jazyk VHDL

Kapitola 3

Analýza a návrh řešení

V této kapitole nejdříve popíšu strukturu rámce MAC jakožto data, se kterými MAC jádro operuje. V případě příjmu jsou pak tato data kontrolována, v případě vysílání pak naopak jádrem generována. Dále se budu věnovat rozhraní MII, které propojuje jádro s fyzickou vrstvou. V závěru navrhnu řešení na úrovni funkčních bloků, které budou v následující kapitole implementovány.

3.1 Struktura rámce MAC

Rámec MAC je protokolární datovou jednotkou (PDU) linkové vrstvy Ethernetu. Skládá se z jednotlivých datových polí, která na sebe navazují. Každé datové pole má velikost v celočíselných násobcích 8 bitů, které bývají v oblasti síťových technologií nazývány *oktety.*[1] Jednotlivé oktety jsou vysílány vždy od nejméně významného bitu. Rámec MAC má následující zobecněnou strukturu: označení začátku rámce, adresní pole, délku/typ dat převzatých z vyšší vrstvy, vlastní data a kontrolní datový součet. V následující části popíšu podrobněji jednotlivá datová pole rámce. Používat budu jejich původní anglické znění podle [1], poněvadž český překlad je přinejmenším velice krkolomný.

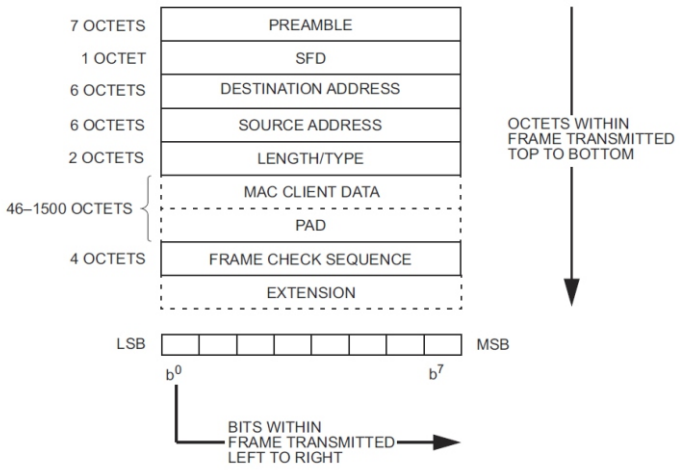
3.1.1 Preamble

Preambule je první datová struktura rámce, která slouží k označení začátku vysílaného rámce, ustavení vysílacího stavu na fyzickém médiu a synchronizaci přijímacích obvodů. Je dlouhá 7 oktetů a obsahuje následující sekvenci bitů: 10101010 10101010 10101010 10101010 10101010 10101010 10101010. Tato sekvence po odvysílání vytvoří díky kódování Manchester, které využívá fyzická vrstva, periodický průběh o poloviční frekvenci.

3.1.2 Start Frame Delimiter (SFD)

Pole SFD je druhé v pořadí. Navazuje na preambuli a označuje, že počínaje následujícím oktetem jsou přenášena vlastní data. Pole je tvořeno sekvencí bitů 10101011. Dohromady

[1] Z různých zdrojů jsem vypátral, že jednotka byte neměla původně specifikovanou velikost. Proto byl zaveden oktet, který je vždy interpretován jako 8 bitů. V následujícím textu budu používat termín oktet tam, kde se jedná o konkrétní tok dat po síti, termínu byte budu používat při popisu obecných dat.



Obrázek 3.1: Struktura rámce MAC a způsob jeho vysílání. Převzato z [1].

tedy tvoří s preambulí 64 po sobě jdoucích jedniček a nul, přičemž místo poslední nuly je odvysílán znak 1.

3.1.3 Destination Address

Toto pole obsahuje fyzickou adresu (adresy) cílového zařízení, pro kterou je rámec určen. Adresa je dlouhá 6 oktetů a má následující vlastnosti:

- První nejméně významný bit 1. vysílaného oktetu (nejvyšší byte) označuje, zda-li je adresa individuální (hodnota 0), nebo jedná-li se o adresu skupinovou (hodnota 1). Speciálním případem skupinové adresy je adresa všesměrového vysílání (broadcast), kterou tvoří samé jedničky.
- Druhý nejméně významný bit 1. vysílaného oktetu (nejvyšší byte) označuje, zda-li je adresa přidělena z globálního adresního rozsahu, nebo jedná-li se o lokálně spravovanou adresu. Pro adresu všesměrového vysílání je tento bit tedy logicky roven 1.
- Zbývajících 46 bitů tvoří vlastní adresu zařízení.

3.1.4 Source Address

Toto pole obsahuje fyzickou adresu (adresy) zdrojového zařízení, které rámec vyslalo. Pro zdrojovou adresu platí stejné zásady jako pro cílovou, viz výše, s malou odchylkou: první nejméně významný bit 1. vysílaného oktetu (nejvyšší byte) je vždy roven 0.

3.1.5 Lenght/Type

Toto pole je dlouhé 2 oktety a nabývá dvou různých významů, v závislosti na jeho numerické hodnotě:

- Je-li jeho hodnota menší než 1536, označuje toto pole délku následujících dat, převzatých z vyšší vrstvy.
- Je-li hodnota tohoto pole vyšší nebo rovna číslu 1536 (0x0600 v šestnáctkové soustavě), jedná se o označení typu protokolu vyšší vrstvy. Nejčastěji se můžeme setkat s hodnotou 0x0800, která označuje protokol IP.

Případné doplnění dat pomocí sekvence oktetů PAD je nezávislé na výše uvedené interpretaci a bude popsáno dále.

3.1.6 MAC Client Data

Toto pole obsahuje vlastní přenášená uživatelská data. Jediným omezením přenášených dat je jejich velikost:

- Přípustná maximální délka dat je 1500 bytů. V případě předání delších dat k odvysílání je toto interpretováno jako chyba a rámec je zahozen.
- Minimální délka dat je 46 bytů. Je-li délka dat menší, doplní se mechanizmem PAD popsaným dále na minimální hodnotu.

Hodnota maximální přípustné velikosti přenášených dat bývá nazývána jako MTU.

3.1.7 PAD

Z časových důvodu při přenášení dat po fyzickém médiu, musí mít rámec minimální délku, která je 576 bitů. Po odečtení velikostí preambule, SFD, adresních polí a délky/typu získáme velikost 46 bytů. Klient podvrstvy MAC tak musí zajistit doplnění datového pole o další oktety v případě, že vyšší vrstva poskytla data menší, než je hodnota 46 bytů.

3.1.8 Frame Check Sequence (FCS)

Toto pole slouží k zabezpečení přenosu dat, má délku 4 oktety a jeho hodnota je vypočtena pomocí cyklického redundantního součtu (CRC). Hodnota CRC je vypočtena ze vstupních dat, která zahrnují pole adresy, typu/délky a vlastních dat – tzn. ze všech předešlých polí s výjimkou preambule a SFD.

Norma [1] specifikuje následující generující polynom:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Podrobnosti algoritmu výpočtu jsou popsány v [1], algoritmus je také znám pod názvem CRC-32-IEEE 802.3.

3.2 Media Independent Interface

Rozhraní MII se nachází mezi fyzickou vrstvou a klientem podvrstvy MAC (popsáno v [1]). Díky tomuto rozhraní je řešení klienta MAC nezávislé na typu použitého média. Původně mělo toto rozhraní formu fyzického propoje pomocí propojovacího kabelu, nyní je realizováno většinou na desce plošného spoje či přímo uvnitř čipu. V tabulce 3.1 je toto rozhraní popsáno výpisem signálů včetně jejich směru a významu. Některé signály jsou pro každou přenosovou rychlost různé, popř. nejsou použity – tyto případy jsou buď specifikovány v tabulce nebo budou popsány v podkapitole 3.3.

Signál	Název signálu	Směr	Význam
RXD(3:0)	Receive Data	O	přijímaná data, při 1000 Mbps spodní 4 bity
RXD(7:4)	Receive Data	O	pouze při 1000 Mbps; přijímaná data, horní 4 bity
RX_DV	Receive Data Valid	O	indikuje platnost dat na sběrnici RXD
RX_ER	Receive Error	O	indikuje chybu během přijímání rámce
CRS	Carrier Sense	O	indikuje přítomnost nosné frekvence na médiu
COL	MII Collision Detect	O	indikuje přítomnost kolizního signálu na médiu
RX_CLK	Receive Clock	O	hodinový signál pro přijímaná data; 2,5 MHz při 10 Mbps, 25 MHz při 100 Mbps a 125 MHz při 1000 Mbps
TXD(3:0)	Transmit Data	I	vysílaná data, při 1000 Mbps spodní 4 bity
TXD(7:4)	Transmit Data	I	pouze při 1000 Mbps; vysílaná data, horní 4 bity
TX_EN	Transmit Enable	I	povoluje vyslání dat přítomných na sběrnici TXD
TX_ER	MII Transmit Error	I	vyvolá vyslání chybového slova na médium
TX_CLK	Transmit Clock	O	hodinový signál pro vysílaná data; 2,5 MHz při 10 Mbps, 25 MHz při 100 Mbps
GTX_CLK	Gigabit Transmit Clock	I	pouze při 1000 Mbps; hodinový signál pro vysílaná data 125 MHz

Tabulka 3.1: Rozhraní (G)MII z hlediska fyzické vrstvy

3.3 Rychlost přenosu

Ethernet byl původně navržen pro přenosovou rychlost 10 Mbps, postupem času však tato rychlost přestala dostačovat. Proto se jeho specifikace dočkala několika změn, které popisovaly

zvyšování přenosové rychlosti. Dnes již patří k standardu pro uživatelská zařízení podpora rychlostí 10, 100 a 1000 Mbps, v oblasti distribuce a agregace síťových linek je možné objevit i rychlost 10 Gbps. Nedávno byla schválena i norma, popisující přenos při rychlosti 40 a 100 Gbps.

Hodnota rychlosti se udává v násobcích jednotky bps (bitů za sekundu) a určuje, kolik datových bitů je přeneseno přes rozhraní (G)MII jedním směrem mezi klientem MAC a fyzickou vrstvou.

3.3.1 10 Mbps

Přenosová rychlost původního standardu Ethernet. Pro přenos přes rozhraní MII jsou používány 4 datové bity (nibble), frekvence hodinového signálu RX_CLK je 2,5 MHz, perioda 400 ns. Přenosovou rychlost můžeme snadno vypočítat: $4\,bit \times 2,5\,MHz = 10\,000\,000\,bps = 10\,Mbps$. Při této rychlosti je možné využívat režimy poloviního a plného duplexu. K řízení přístupu k médiu jsou používany signály CRS a COL.

3.3.2 100 Mbps

Z důvodu nedostatečné datové prostupnosti byla přenosová rychlost navýšena na 100 Mbps. Tento standard byl nazván „Fast Ethernet“ z důvodu odlišení od předchozí verze. Pro tuto přenosouvou rychlost platí vše, co pro rychlost 10 Mbps s následujícími odchylkami: frekvence hodinového signálu RX_CLK je 25 MHz, perioda 40 ns. hodnotu přenosové rychlosti získáme obdobně při následujícím výpočtu: $4\,bit \times 25\,MHz = 100\,000\,000\,bps = 100\,Mbps$.

3.3.3 1000 Mbps

Tento režim přenosu bývá častěji nazýván „Gigabit Ethernet“ a rychlost udávána jako 1 Gbps. Narozdíl od předešlého navýšení rychlosti se změna netýkala jen zvýšení hodinové frekvence (a na ní závislého časování), ale postihla specifikaci ve více směrech. Zvýšení hodinové frekvence prostým vynásobením 10 nebylo tehdy z technickcýh důvodů možné, proto bylo přistoupeno k následujícímu řešení:

- počet datových bitů byl navýšen ze 4 na 8
- hodinová frekvence RX_CLK byla navýšena na 125 MHz (perioda 8 ns)
- hodinový signál pro přenos TX_CLK, který u nižších rychlostí poskytuje fyzická vrstva, není používán
- byl zaveden signál GTX_CLK, který je dodáván klientem MAC do fyzické vrstvy a slouží pro časování signálů určených k vysílání.
- byla zrušena podpora režimu poloviního duplexu

Rozhraní MII se tak označuje jako GMII (Gigabit MII). Změny ohledně frekvence a zdroje hodinového signálu mají svůj důvod v obvodovém řešení, kdy Signál RX_CLK je rekonstruován z přijímaných dat. Signály GTX_CLK a RX_CLK nejsou pevně fázově svázány. Přenosovou rychlost můžeme vypočítat jako: $8\,bit \times 125\,MHz = 1\,000\,000\,000\,bps = 1\,000\,Mbps = 1\,Gbps$.

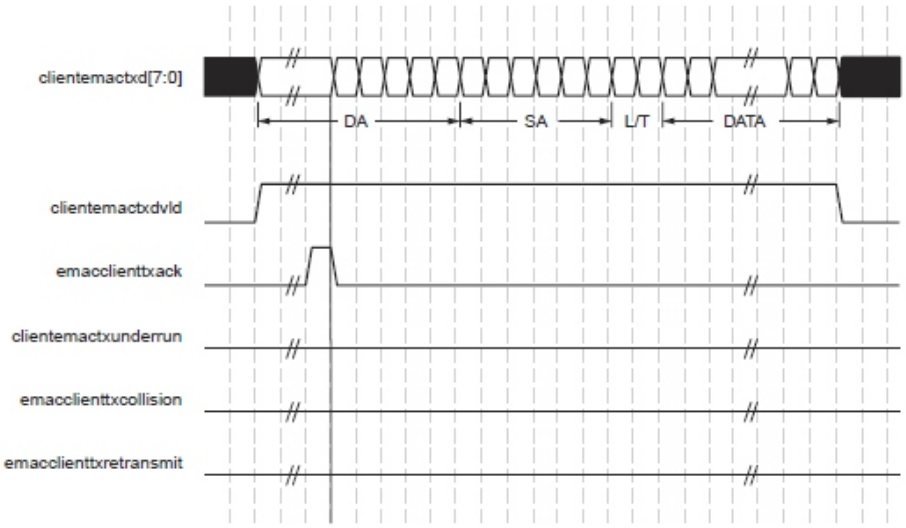
3.4 Rozhraní klienta vyšší vrstvy

Jako rozhraní mezi MAC jádrem a klientem vyšší vrstvy bylo použito řešení firmy Xilinx podle [4] z důvodu kompatibility jádra s ostatními bloky již existujících řešeních vyšších vrstev.

Tabulka 3.2 popisuje signály části rozhraní, které je určeno pro odvysílání dat MAC jádrem. Na obrázku 3.2 je vyobrazen časový průběh signálů. Pokud chceme data odvysílat, připravíme je na datové sběrnici a čekáme do té doby, než jádro vystaví handshake signál emacclienttxack (z důvodu času potřebného pro vygenerování polí začátku rámce), po jehož aktivaci můžeme zahájit samotný přenos.

Signál	Směr	Význam
clientemactxd(7:0)	I	data určená k odvysílání
clientemactxdvld	I	řídící signál oznamuje, že na portu clientemactxd jsou připravena data k odvysílání
emacclienttxack	O	handshake přenosu dat, oznamuje, že klient vyšší vrstvy může začít s vlastním přenosem
clientemactxunderrun	I	chybový signál od klienta vyšší vrstvy, MAC jádro přeruší vysílání dat a vyšle chybové slovo

Tabulka 3.2: Klientské rozhraní pro vysílání



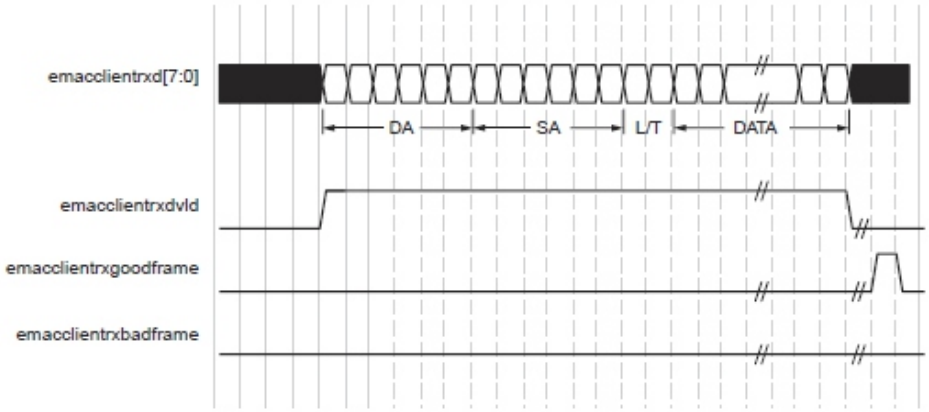
Obrázek 3.2: Časový průběh přenosu dat určených k vysílání na klientském rozhraní. Převzato z [4].

Tabulka 3.3 popisuje signály, které MAC jádro vystavuje při příjmu. Na obrázku 3.3 je vyobrazen časový průběh signálů. Data jsou po dobu přijmu předávány vyšším vrstvám za

současné aktivace signálu emacclientrxdvld. Po skončení příjmu dat je tento signál deaktivován a v závislosti na kontrole příjmu rámce je buď aktivována indikace jeho správnosti nebo v opačném případě jeho chyby.

Signál	Směr	Význam
emacclientrxd(7:0)	O	přijímaná data
emacclienttxdvld	O	řídící signál oznamuje, že na portu emacclientxd jsou platná data pro příjem
emacclientrxgoodframe	O	signál vyslán po skončení vlastního příjmu dat, indikuje správnost přijatého rámce
emacclientrxbadframe	O	signál vyslán po skončení vlastního příjmu dat, indikuje chybu přijatého rámce; klient vyšší vrstvy by tento rámec měl zahodit

Tabulka 3.3: Klientské rozhraní pro příjem



Obrázek 3.3: Časový průběh přenosu přijatých dat na klientském rozhraní. Převzato z [4].

Všechny řídící signály jsou aktivní v logické 1, data jsou zpracovávána s náběžnou hranou hodin. Perioda hodinového signálu je závislá na rychlosti přenosu a nabývá následujících hodnot:

- 800 ns pro přenosovou rychlost 10 Mbps
- 80 ns pro přenosovou rychlost 100 Mbps
- 80 ns pro přenosovou rychlost 1000 Mbps

3.5 Návrh řešení

V následujícím textu popíši návrh mého řešení na úrovni funkčních bloků a jejich propojení. Konkrétní implementace pak bude popsána v další kapitole.

Ethernetové MAC jádro bude řešeno dvěma základními bloky: blokem pro příjem a blokem pro vysílání. Tyto bloky budou na sobě pracovat nezávisle, kromě společných řídicích signálů. Spojením obou bloků vznikne samotné jádro s rozhraním, které bylo definováno v tabulkách 3.1, 3.2 a 3.3.

3.5.1 Blok Rx

Tento hlavní funkční blok slouží pro příjem MAC rámce z (G)MII rozhraní fyzické vrstvy. Po zpracování rámce budou odpouzdřená data poskytnuta ke zpracování vyšší vrstvě. Blok pro příjem bude mít za úkol následující:

- úpravu vstupních dat do oktetové formy
- přizpůsobení časování pro rozhraní vyšší vrstvy
- zachycení preambule a začátku rámce
- kontrolu CRC
- předání uživatelských dat vyšší vrstvě
- na základě kontroly rámce signalizovat stav vyšší vrstvě

3.5.1.1 Úprava vstupních dat

Tento funkční blok má zajistit předání dat z fyzické vrstvy k dalšímu zpracování. Jeho úkolem bude data převést do oktetové formy a těmto datům přizpůsobit stavové signály převzaté z rozhraní (G)MII. Jeho funkcionalita bude záviset na konkrétní přenosové rychlosti, pro kterou je MAC rámec zpracováván. Mimo dříve specifikovaných rozhraní bude blok řízen i signálem určujícím přenosovou rychlost, na jehož základě bude mít rozdílnou funkcionalitu.

Pro rychlost 1000 Mbps je funkcionalita tohoto bloku jednoduchá, blok jen musí předat sekvenci oktetů a řídicích signálů na společnou sběrnici pro zpracování dalšími jednotkami.

Pro rychlosti 10 a 100 Mbps musí tento blok zajistit tyto úkony: zachycení vstupních dat se správným fázovým posuvem, jejich pararelizaci (převod dvou po sobě jdoucích nibble na byte) a správné nastavení stavových signálů. Toto je zapotřebí z důvodu, že při těchto rychlostech jsou data předávána jen po 4 bitech. Blok také musí zajistit předání dat z jedné hodinové domény do druhé (data jsou přijímána z fyzické vrstvy na dvojnásobné frekvenci než jsou předávána vyšší vrstvě; toto při gigabitovém režimu odpadá).

3.5.1.2 Kontrola CRC

Tento blok bude zajišťovat kontrolu pole FCS. Na základě řídicích signálů bude průběžně počítat kontrolní součet a v případě jeho správnosti vystaví přislušný signál jednotce řízení příjmu. Ethernetový rámec je k průběžnému výpočtu uzpůsoben, kontrolní sekvence figuruje jako poslední datová posloupnost rámce.

3.5.1.3 Řízení příjmu

Tento blok bude mít na starost samotný příjem rámce. V prvním kroku detekuje preambuli a SFD, poté začne zajišťovat průběžný výpočet CRC a data předávat vyšší vrstvě (s vystavením příslušných stavových signálů). Na základě vstupu z nižší vrstvy a výsledku kontrolního součtu potvrdí vyšší vrstvě platnost rámce či jeho zahození. Během příjmu kontroluje jednotka také správnou délku přijímaných dat.

3.5.1.4 Správa Rx hodinového signálu

Vstupem této jednotky bude signál RX_CLK, jehož frekvence se mění dle přenosové rychlosti (popsáno v 3.3). Výstupem budou dva hodinové signály CLK a CLK2. Signál CLK bude pracovat na frekvenci klientského rozhraní vyšší vrstvy a bude sloužit jako základní hodinový signál pro celý blok příjmu. Hodinový signál CLK2 bude pracovat na frekvenci fyzické vrstvy a bude využit v bloku vstupních dat. Pro přenosovou rychlost 1000 Mbps budou tyto signály tedy shodné.

3.5.2 Blok Tx

Tento hlavní funkční blok slouží pro vysílání MAC rámce z klientského rozhraní vyšší vrstvy. Po obdržení uživatelských dat budou tato data zapouzdřena do MAC rámce a vygenerována příslušná datová pole. Následně dojde k jejich předání fyzické vrstvě přes rozhraní (G)MII. Blok pro vysílání bude mít za úkol následující úkony:

- úpravu vysílaných dat a přizpůsobení časování pro rozhraní (G)MII
- vygenerování preambule a SFD
- výpočet CRC a jeho vložení do FCS
- kontrolu délky uživatelských dat
- případné generování PAD sekvence

3.5.2.1 Úprava dat pro rozhraní (G)MII

Tento funkční blok má za úkol předat vygenerovaný rámec přes rozhraní (G)MII fyzické vrstvě k odvysílání. Blok bude řízen signálem specifikujícím přenosovou rychlost, na jehož základě bude mít různou funkcionalitu.

Pro rychlost 1000 Mbps je funkcionalita tohoto bloku jednoduchá, blok jen musí předat sekvenci oktetů a řídicích signálů ze společné sběrnice na rozhraní (G)MII.

Pro rychlosti 10 a 100 Mbps musí tento blok zajistit tyto úkony: převedení dat do formy za sebou jdoucích nibble, správné vystavení řídicích signálů pro fyzickou vrstvu a zajistit přechod dat z hodinové domény klientského rozhraní do hodinové domény fyzické vrstvy (data jsou vysílána do fyzické vrstvy na dvojnásobné frekvenci než jsou přijímána z vyšší vrstvy; toto při gigabitovém režimu odpadá).

3.5.2.2 Výpočet CRC

Tato jednotka bude mít za úkol průběžný výpočet CRC z uživatelských dat. Po dokončení příjmu uživatelských dat bude vypočtená hodnota CRC předána na společnou sběrnici ve formě pole FCS. Jednotka také vystaví příslušné stavové signály za účelem řízení tvorby rámce.

3.5.2.3 Řízení vysílání

Tento blok bude mít na starost samotné vysílání rámce. V prvním kroku vygeneruje preambuli a sekvenci SFD. Po provedení handshake začne se zahrnutím uživatelských dat do struktury rámce a vydá pokyn pro výpočet CRC. Po dokončení příjmu dat vloží na konec rámce pole FCS. Kontroluje také velikost vstupních uživatelských dat: v případě překročení maximální povolené délky dojde k zahození rámce, v případě menší velikosti dojde k vygenerování polí PAD.

3.5.2.4 Správa Tx hodinového signálu

Tato jednotka bude o něco složitější než jednotka pro příjem. Mimo vstupu pro signál TX_CLK bude obsahovat i výstup pro signál GTX_CLK. Jednotka bude opět generovat hodinové signály CLK a CLK2 pro hodinové domény klientského rozhraní a fyzické vrstvy.

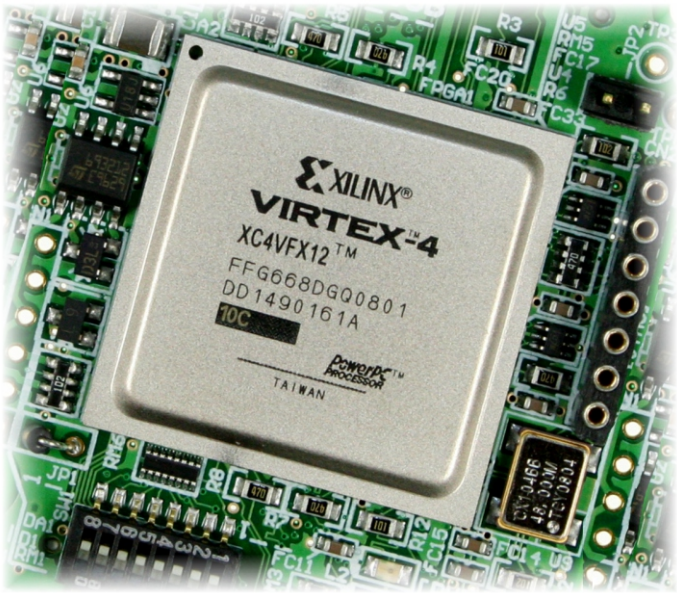
Kapitola 4

Realizace

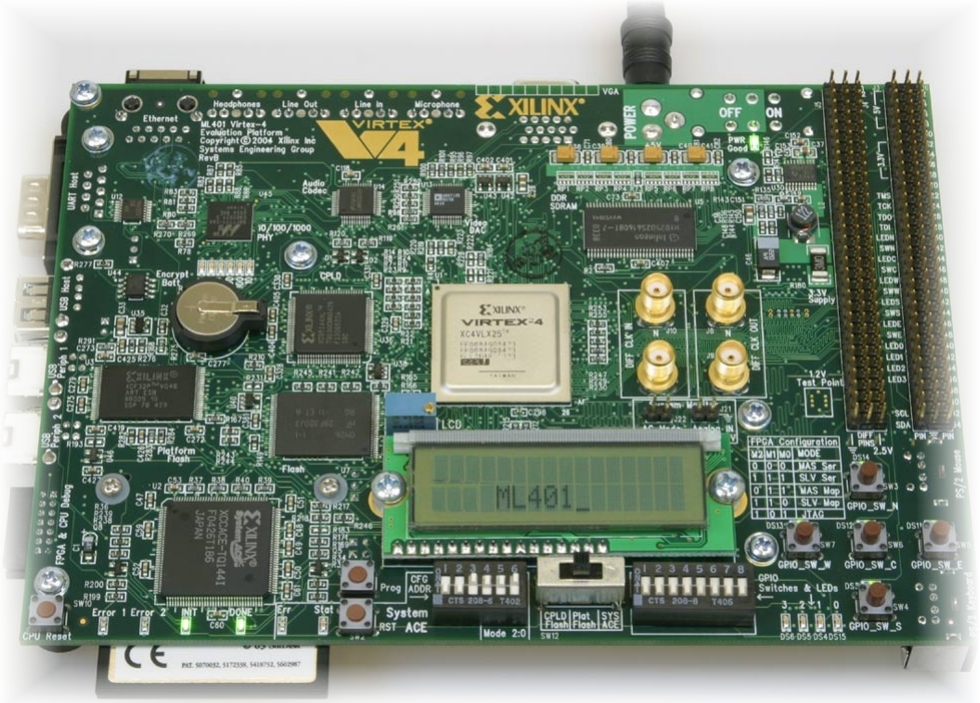
Pro implementaci hardwarového řešení byl použit jazyk VHDL (dle oficiálního zadání). Cílovou platformou byl obvod FPGA Virtex™-4 4.1 od firmy Xilinx, konkrétně XC4VLX25™. Jako vývojová platforma sloužila deska ML401 4.2 (dle oficiálního zadání), která mimo výše popsaného obvodu obsahuje integrovaný obvod ethernetové fyzické vrstvy Alaska 88E1111 od firmy Marvell. Deska dále obsahuje všechny další potřebné podpůrné obvody a jednotky včetně hodinových oscilátorů a rozhrani JTAG.

Pro syntézu byl využit program XST a vývojové prostředí Xilinx ISE. Logické obvody byly popsány syntetizovatelnými konstrukty jazyka VHDL-93.

V následujícím textu popíši strukturu zdrojového kódu a konkrétní řešení jednotlivých částí.



Obrázek 4.1: FPGA obvod Virtex-4.



Obrázek 4.2: Vývojová deska Xilinx ML401.

4.1 Struktura zdrojového kódu

Zdrojový kód je rozdělen na několik částí:

- Rx – přijímací část
- Tx – vysílací část
- Common – společné komponenty (generické registry apod.)
- miim – správa fyzické vrstvy
- emac_top – top architektura

4.2 emac_top

Top modul celého MAC jádra. Spojuje moduly rx_engine a tx_engine a definuje rozhraní celého jádra.

Oba moduly jsou odděleny a pracují nezávisle na sobě. Sdílejí pouze společné řídicí signály specifikující přenosovou rychlost. Rozhraní každého modulu je shodné s částmi rozhraní celého jádra – lze je tudíž použít pro konkrétní specifické aplikace samostatně (např. přijímač pro síťovou analýzu, generátor IP paketů apod.).

V následujícím výpisu je uvedeno jeho rozhraní:

```
ENTITY eth_mac IS
PORT(
 clientemactxd : IN std_logic_vector(7 downto 0);
 clientemactxdvld : IN std_logic;
 clientemactxifgdelay : IN std_logic_vector(7 downto 0);
 clientemactxunderrun : IN std_logic;
 rst : IN std_logic;
 txgmiimiiclk : IN std_logic;
 rxgmiimiiclk : IN std_logic;
 clk125 : IN std_logic;
 speed : IN std_logic;
 phyemacrxd : IN std_logic_vector(7 downto 0);
 phyemacrxdv : IN std_logic;
 phyemacrxer : IN std_logic;
 emacclientrxd : OUT std_logic_vector(7 downto 0);
 emacclientrxdvld : OUT std_logic;
 emacclientrxgoodframe : OUT std_logic;
 emacclientrxbadframe : OUT std_logic;
 emacclienttxack : OUT std_logic;
 speedis1000 : OUT std_logic;
 speedis10100 : OUT std_logic;
 gtx_clk : OUT std_logic;
 rx_client_clock : OUT std_logic;
 tx_client_clock : OUT std_logic;
 emacphytxd : OUT std_logic_vector(7 downto 0);
 emacphytxen : OUT std_logic;
 emacphytxer : OUT std_logic
 );
END eth_mac;
```

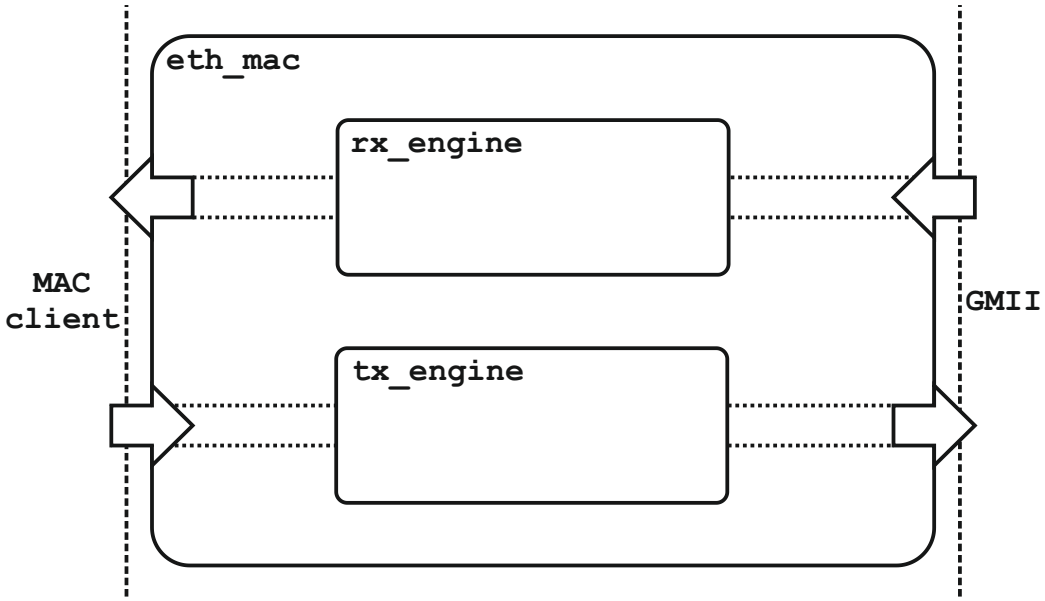
4.2.1 Rx_engine

Jedná se o top modul přijímací části. Sdružuje jednotku řízení hodinového signálu, datové cesty a řadič přijímače.

4.2.1.1 Rx_datapath

V tomto modulu se nalézají funkční jednotky pro příjem rámce. Nejdůležitějšími jsou: úprava vstupních dat, ověření CRC a výstupní registry. Modul sdílí řídicí rozhraní z blokem rx_ctrl.

Jednotka úpravy vstupních dat je zde hlavně z důvodu provozu na přenosových rychlostech 10 a 100 Mbps. Zajišťuje převod dat z hodinové domény fyzické vsrstvy a jejich pararelizaci – při těchto rychlostech se data přenáší po 4 bitech s periodou hodin 400 resp. 40 ns. Ve výstupnim registru jsou pak již přítomna vlastní osmibitová data; je také provedena synchronizace stavových signálů, zejména RX_DV (při přechodu hodinové domény je



Obrázek 4.3: Blokové schéma modulu eth_mac

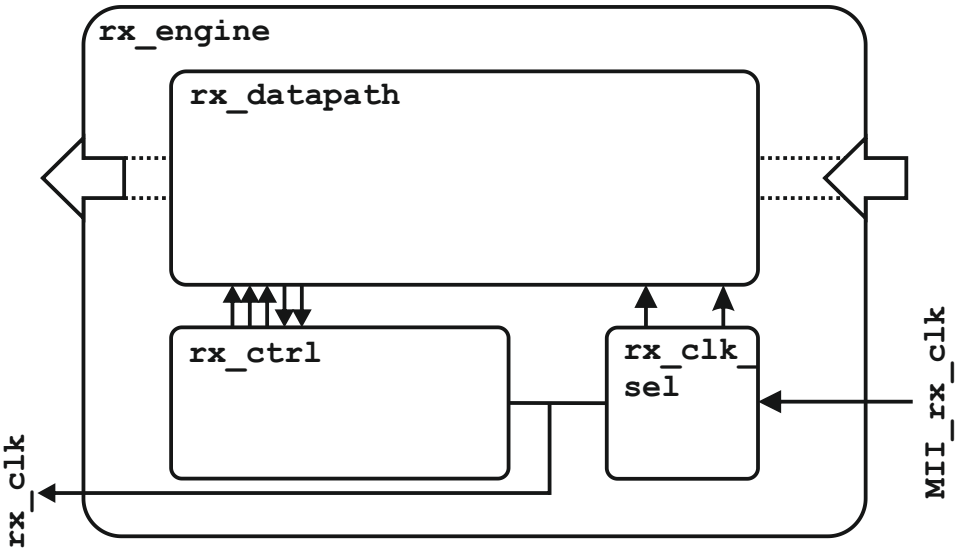
nutné provést synchronizaci dat na správnou náběžnou hranu; jinak by došlo k nechtěnému prohození nibblů). Pro pro režim přenosové rychlosti 1000 Mbps data a stavové signály procházejí pouze sadou registrů (tady a v celém návrhu je pro přepínání datových cest použito multiplexorů, u nichž log. 0 na řidicím vstupu znamená selekci signálů pro rychlost 1000 Mbps a log. 1 pro rychlosti 10 a 100 Mbps).

Pro modul kontroly CRC byla použita paralelní hardwarová implementace algoritmu výpočtu s lineárním zpětnovazebním registrem podle [5]. Při kontrole CRC jsem s výhodou využil techniku „magického čísla“, kdy se do vstupních dat zahrne i samotná hodnota CRC (MAC rámec toto umožnuje, hodnota CRC je obsažena v poli FCS). Jestliže hodnota CRC odpovídala přenášeným datům, zbytek po dělení v případě generujícího polynomu dle [1] bude roven hodnotě 0xC704DD7B.

4.2.1.2 Rx_ctrl

Na obrázku 4.5 je vyobrazen graf konečného automatu řadiče přijímacího modulu. Graf je vyobrazen pouze zjednodušeně bez vyznačených vstupů a výstupů kvůli přehlednosti. Přechody jsou pro zájemce podrobně popsány přímo ve zdrojovém kódu, podle mě v daleko přehlednější podobě.

Konečný automat o devíti stavech je typu Moore a je implementován pomocí kódu se třemi procesy (tuto technikou jsem použil i u ostatních konečných automatů v dalších částech mého návrhu). Bylo provedeno ošetření kódování neplatného stavu a nastavení výchozího stavu. Jako výstup jsem zvolil signál ctrl_vecotr, který je posléze kvůli přehlednosti rozložen na jednotlivé stavové signály. Mimo samotného automatu modul ještě obsahuje podpůrný čítač pro odpočet délky jednotlivých polí MAC rámce (SFD, minimální délka apod.).



Obrázek 4.4: Blokové schéma modulu rx_engine.

4.2.1.3 Rx_clk_sel

Modul, který slouží ke správě hodinových signálů pro různé přenosové rychlosti. Jeho vstupem je zdroj hodinového signálu RX_CLK z rozhraní (G)MII a výstupem jsou dva hodinové signály CLK a CLK2.

Pro provoz na rychlostech 10 a 100 Mbps je perioda hodinového signálu na vstupu 400 resp. 40 ns. Tento signál je kopírován na výstup CLK2, pro výstup CLK je frekvence vydělena dvěma tak, aby odpovídala periodě signálu 800 resp 80 ns klientského rozhraní. Signál CLK je také hlavním hodinovým signálem celého modulu Rx_engine, CLK2 je použit na rozhraní fyzické vrstvy pro registry paralelizace vstupních dat.

Pro provoz na rychlosti 1000 Mbps je vstupem signál o periodě 8 ns, který odpovídá i periodě klientského rozhraní. Ovšem díky zpoždění datových signálů bylo nutné (jen pro rychlost 1000 Mbps) tento signál fázově posunout asi o 2 ns (zjištěno experimentálně), jinak dochází k nesprávnému čtení dat. Samotný fázový posuv a časová korekce hodinového signálu jsou zajištěny pomocí DCM [12]. V tomto případě jsou hodinové signály CLK a CLK2 shodné.

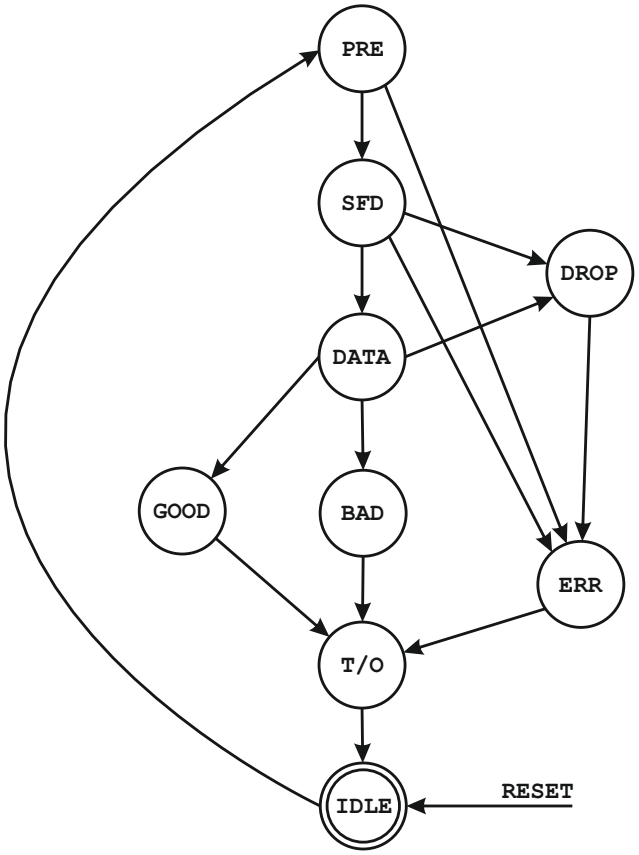
Přepínání hodinových signálů na výstupy CLK a CLK2 je realizováno pomocí dvojice multiplexorů hodinového signálu BUFGMUX [12].

4.2.2 Tx_engine

Top modul vysílací části. Sdružuje jednotku řízení hodinového signálu, datové cesty a řadič přijímače.

4.2.2.1 Tx_datapath

V tomto modulu se nalézají funkční jednotky pro generování a vysílání MAC rámce. Nejdůležitějšími jsou: generování polí MAC rámce (preambule, SFD a FCS) a výstupní registry.



Obrázek 4.5: Konečný automat řadiče přijímací části – modulu rx_ctrl.

Modul sdílí řídicí rozhraní s blokem tx_ctrl. Modul je vytvořen podobně jako u přijímače až na několik odlišností.

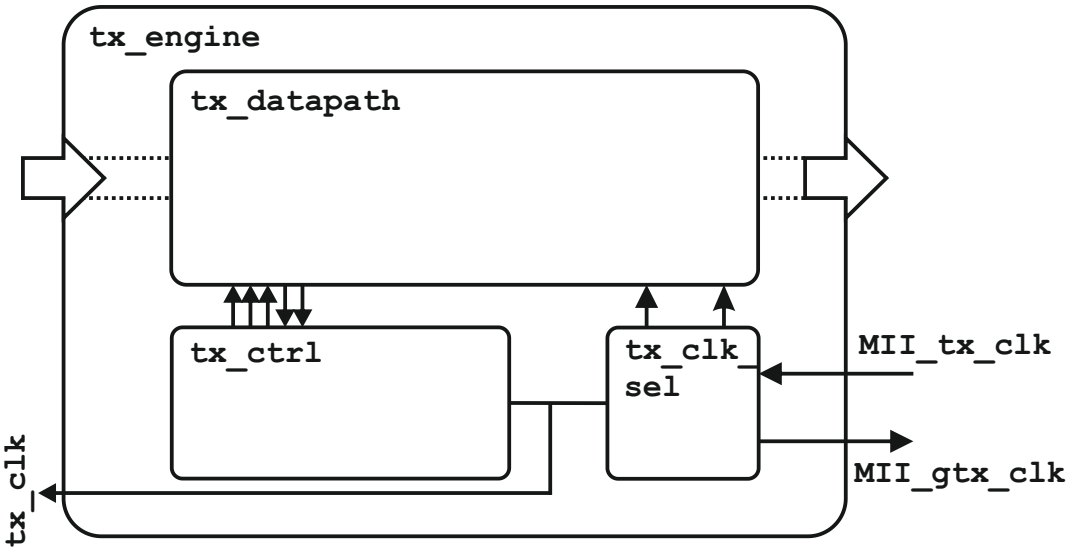
Jednotka výpočtu CRC nyní FCS neověřuje, ale generuje. Opět je použito řešení podle [5], kdy po vystavení řídicího signálu jednotka ve čtyřech hodinových taktech vyšle na svůj výstup jednotlivé oktety pole FCS.

Soubor výstupních registrů v případě rychlostí 10 a 100 Mbps přizpůsobuje data hodinové doméně fyzické vrstvy a zajišťuje deparalelizaci výstupních dat. Také synchronizuje řídicí signály pro fyzickou vrstvu.

Důležitou součástí datových cest je i multiplexor, který slouží k přidávání jednotlivých polí MAC rámce na výstupní sběrnici. V závislosti na řídicích signálech přepíná na výstup data 0x55 (preambule), 0xD5 (SFD), klientská data z vyšší vrstvy nebo výstup z CRC jednotky.

4.2.2.2 Tx_ctrl

Na obrázku 4.7 je vyobrazen graf konečného automatu řadiče vysílacího modulu. Konečný automat má osm stavů, byl proveden podobným způsobem jako u modulu rx_clk a platí pro něj výše uvedené zásady.



Obrázek 4.6: Blokové schéma modulu tx_engine.

Podpůrnými obvody jsou opět čítače pro jednotlivá pole MAC rámce. Navíc oproti přijímacímu modulu je vybaven čítačem pro generování časové prodlevy IFG (časová mezera mezi vysílanými rámci) – počet hodinových taktů je možné dynamicky měnit pomocí nastavení hodnoty signálu clientemactxifgdelay.

Při vysílání rámce je používán handshake signál emacclienttxack, který klientovi vyšší vrstvy potvrdí, že může začít s vystavováním dat na klientské rozhraní (handshake je použit z důvodu generování preambule a pole SFD).

4.2.2.3 Tx_clk_sel

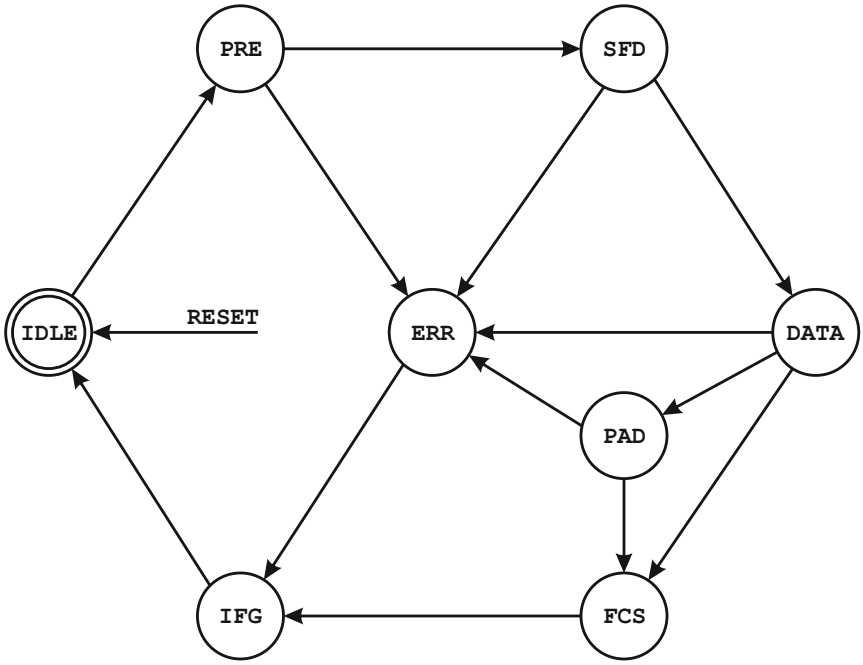
Pro rychlosti 10 a 100 Mbps je opět pro generování signálů CLK a CLK2 využito signálu fyzické vrstvy TX_CLK, který je jedním ze vstupů. Obdobně jako u přijímače je řešeno dělení frekvence hodina a jejich přepínání na výstupy.

Při rychlosti přenosu 1000 Mbps je situace odlišná. Hodinový signál o periodě 8 ns musí být zajištěn MAC vrstvou a přes signál gtx_clk dodán do fyzické vrstvy. Proto má jednotka o hodinový vstup navíc – vstup signálu o frekvenci 125 MHz.

Obě jednotky správy hodin dodávají pro klientské rozhraní hodinový signál o periodě 800, 80 resp. 8 ns v závislosti na rychlosti přenosu. Kvůli ošetření předstihu dat jde tento signál na výstup přes DDR registr, který zajistí fázový posuv o 180 stupňů (náběžná hrana hodin se tak dostane do středu datového intervalu).

4.3 MIIM

Pro dynamickou konfiguraci přenosových rychlostí byla vytvořena jednotka miim_sta, která má na starost komunikaci s konfiguračními registry fyzické vrstvy přes sběrnici MDIO. Fy-

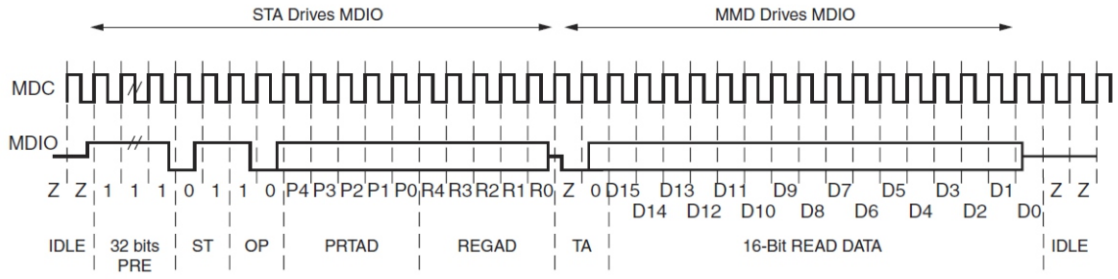


Obrázek 4.7: Konečný automat řadiče vysílací části – modulu tx_ctrl.

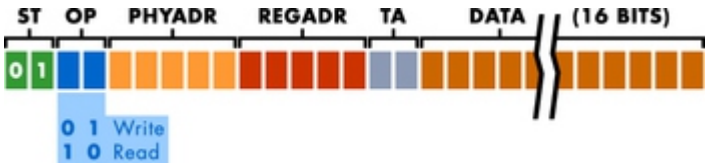
zická vrstva disponuje registrovým polem, ve kterém jsou uloženy stavové hodnoty jednotlivých součástí fyzické vrstvy. Jejich adresy a význam lze nalézt ve standardu [1].

Pro komunikaci se správou fyzické vrstvy je použita dvouvodičová sběrnice MDIO, struktura rámce je popsána na obrázku 4.9. Data jsou na sběrnici posílána pomocí třístavových budičů (datový vodič slouží pro obousměrnou komunikaci). Pro vyslání rámce je použit posuvný registr, do kterého se zpětně zapisují přijatá data. Adresa fyzické vrstvy je pro desku ML401 [9] nastavena na hodnotu 0, číslo registru s informacemi o právě nastavené rychlosti je 17 (0x11).

Jednotka miim_sta je oddělena od vlastního MAC jádra a nemusí být nutně použita, pokud budeme jádro provozovat pouze na jedné rychlosti.



Obrázek 4.8: Komunikace na sběrnici MDIO, časový průběh. Převzato z [4].



Obrázek 4.9: Formát rámce sběrnice MDIO. Převzato z [7].

4.4 Report

```
Device Utilization Summary:

  Number of BUFGs                   4 out of 32     12%
  Number of BUFGCTRLs               4 out of 32     12%
  Number of DCM_ADVs                2 out of 8      25%
  Number of External IOBs          49 out of 448    10%
     Number of LOCed IOBs          49 out of 49    100%

  Number of OLOGICs                 3 out of 448     1%
  Number of Slices                264 out of 10752   2%
     Number of SLICEMs              5 out of 5376    1%
```

Kapitola 5

Testování

Během samotného návrhu jsem prováděl řadu testování a simulací. Toto se dá rozdělit na dva základní problémy, které jsem řešil: funkční verifikace jednotlivých modulů a verifikace celého jádra – o testování jednotlivých modulů se rozepíšu v následující podkapitole 5.1.

Pro ověření funkčnosti celého jádra (resp. bloků rx_engine a tx_engine) bylo nutno vygenerovat nezapouzdřené MAC rámce: toto jsem dělal za pomoci volně dostupných aplikací Colasoft Packet Builder a Wireshark. První z výše uvedených nástrojů umožňuje pomocí GUI vygenerovat libovolné rámce, pakety či segmenty. Obsluha je uživatelsky velice příjemná, program zobrazuje ve stromové struktuře jednotlivé pole PDU a sám vypočítává hodnoty CRC, velikostí a dalších dynamicky se měnících parametrů (toto chování je samozřejmě možné potlačit.). Wireshark je velmi známý a rozšířený analyzátor síťového provozu založený na knihovně pcap.

Každý MAC rámec pro testování příjmu má určitou vlastnost, která se má ověřit:

- velikost dat rovna minimální velikosti
- velikost dat rovna minimální velikosti - 1
- velikost dat rovna maximální velikosti
- velikost dat rovna maximální velikosti + 1
- chybná hodnota pole FCS
- špatně formovaná preambule či SFD

U dat určených k testování vysílání je situace o něco jednodušší:

- velikost dat menší než minimální velikost
- velikost dat větší než minimální velikost
- velikost dat rovna maximální velikosti
- velikost dat větší než maximální velikost

Tato data jsou uložena v samostatných souborech a jsou určena ke generování stimulů.

```
SIZE = MINSIZE

Wireshark:
0x00  da 12 34 56 78 90 5a 12  ..4Vx.Z.
0x08  34 56 78 90 08 00 45 dd  4Vx...E.
0x10  00 2e 00 ab 03 00 40 11  ......@.
0x08  9f 1e c0 a8 01 c8 13 a8  ........
0x20  01 01 41 42 43 44 65 20  ..ABCDe
0x28  70 6f 72 6e 2e 62 61 74  porn.bat
0x30  2f 64 65 76 2f 6e 75 6c  /dev/nul
0x38  6c 2d 53 48              l-SH

Colasoft Packet Builder:
DA 12 34 56 78 90 5A 12 34 56 78 90 08
00 45 DD 00 2E 00 AB 03 00 40 11 9F 1E
C0 A8 01 C8 13 A8 01 01 41 42 43 44 65
20 70 6F 72 6E 2E 62 61 74 2F 64 65 76
2F 6E 75 6C 6C 2D 53 48

FCS: 0xA0F04CF6
```

5.1 Simulace

Pro simulaci jsem využil simulátor ModelSim. Jako první jsem přeložil a odsimuloval ukázku od firmy Xilinx, a to jádra TEMAC [4]. Tohoto jsem poté využil v dalším návrhu, při simulacích se daly odpozorovat jednotlivé časové průběhy a komunikaci po rozhraních. Rozcházení ukázkové aplikace bránila řada problémů (hlavně se simulačními knihovnami), nakonec jsem ale vše zdárně vyřešil.

Jako další a největší blok verifikací mě čekal při vlastním návrhu. Zde jsem vždy napsal pro každý modul samostatný testbench (lze nalézt v adresáři test). Pro danou jednotku byl vždy vygenerován model post-map a post-par.

```
 D | G B   # Hlavicka s popisem signalu
 A | O A   # data, good frame, bad frame
 T | O D   #
 A | D     #
---------  # Oddelovac
00 | 0 0   # zachycena data
00 | 0 0   #
00 | 0 0   #
55 | 0 0   # zacatek preambule
55 | 0 0   #
55 | 0 0   #
```

```
55 | 0 0   #
55 | 0 0   #
55 | 0 0   #
55 | 0 0   #
D5 | 0 0   # SFD
DA | 0 0   # cilova adresa
12 | 0 0   #
34 | 0 0   #
56 | 0 0   #
78 | 0 0   #
90 | 0 0   # zdrojova adresa
5A | 0 0   #
12 | 0 0   #
00 | 0 0   #
```

Pro testování celého jádra jsem s výhodou použil knihovnu text_io, která umožňuje čtení a zápis dat ze souborů. Bohužel její možnosti jsou velice omezené, proto jsem podle [11] využil podpůrné knihovny std_logic_textio (původně od firmy Synopsys, nyní je již zahrnuta do knihovny IEEE), která podporuje mj. načítání čísel v šestnáctkové soustavě. Jako poslední knihovnu pro práci s textem jsem využil txt_util [11], která umožňuje dělat podobné operace, na které jsme zvyklí u programovacích jazyků pro software. Pomocí výše uvedených knihoven jsem načítal data pro stimul a výsledná data ukládal, viz uvedený výpis.

5.2 ChipScope

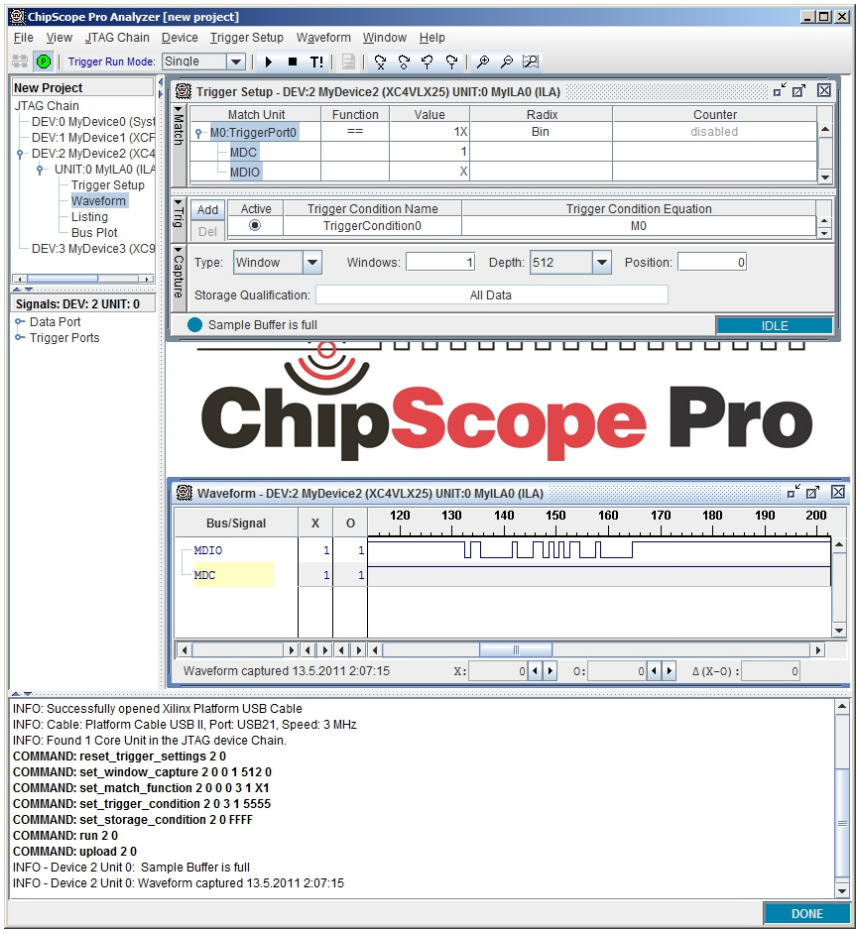
ChipScope^{TM}Pro je logický analyzátor, který se dá jako IP Core vložit do testovaného návrhu. Po napojení na testované signály tak získáváme mocný nástroj, kterým můžeme přímo na čipu kontrolovat téměř bezprostředně logické signály. Nástroj se vkládá jako několik jader – prvním jádrem je řídicí obvod, který zajišťuje komunikaci s uživatelskou aplikací. Další jádra jsou jednotlivé analyzátory, které můžeme vkládat tam, kde je potřeba. S hlavní jednotkou jsou propojeny pomocí zvláštní sběrnice.

ChipScope mi velice usnadnil kontrolu návrhu - hlavně jsem díky němu viděl, jak signály ve skutečnosti vypadají. Nezastupitelnou roli měl při odlaďování komunikace přes sběrnici MDIO, která sdílí jeden přenosový vodič a je buzena třístavovými budiči.

5.3 Reálný provoz

Nejzajímavější částí bylo určitě testování jádra a jeho součástí v reálném provozu. Za tímto účelem jsem vytvořil následující prostředí: PC stanice s generátorem rámců, přepínače Cisco Catalyst 2960 a 3750, laptop se síťovým analyzátorem a samotné jádro na desce ML401 s připojeným loopbackem.

Původně jsem měl k dispozici vývojovou desku s obvodem Spartan-3E, která obsahovala pouze fyzickou vrstvu s rychlostí do 100 Mbps [2]. Ta mi posloužila k vytvoření pomocného projektu, kdy jsem na ní realizoval „packet-sniffer“ a ethernetové rámce jsem přes FIFO



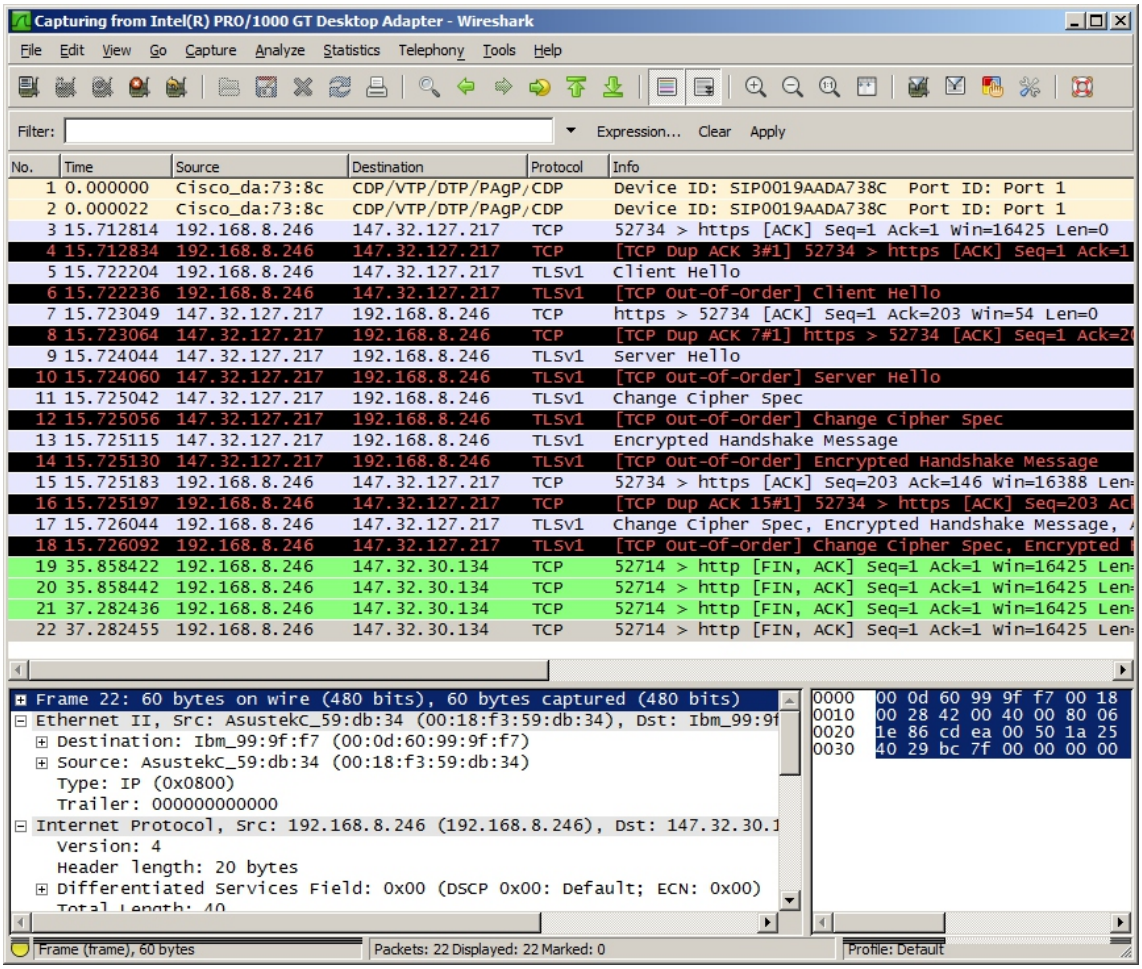
Obrázek 5.1: GUI programu ChipScope. V okně Waveform můžeme vidět zachycený průběh signálu na sběrnici MDIO.

posílal sériovou linkou do počítače (využíval jsem velice zdařilé terminálové aplikace RealTerm). Takto jsem přijímal přímo surová data z fyzické vrstvy a mohl jsem tak svůj návrh konfrontovat s reálnými daty na médiu.

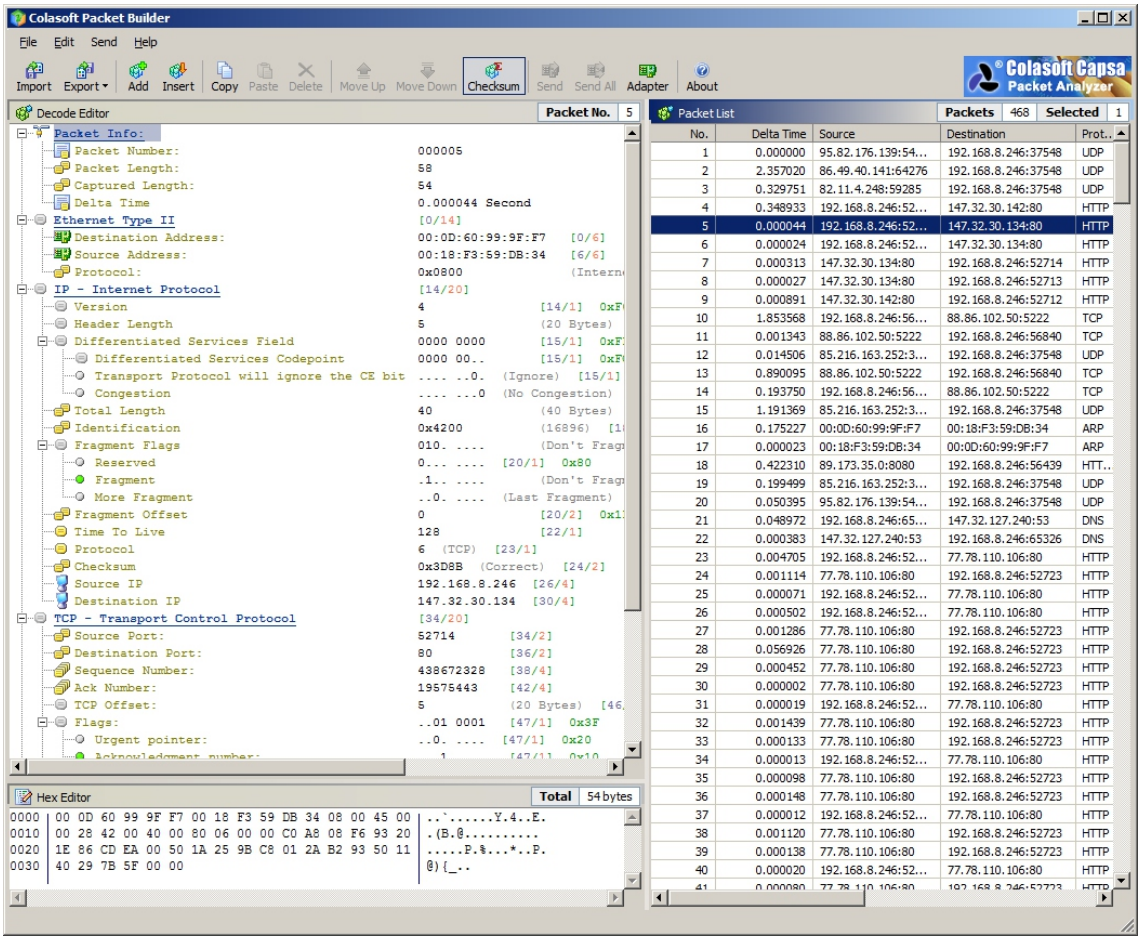
Jádro fungovalo dobře již při prvních testech, bylo potřeba jen odladit handshake na klientském rozhraní a experimentálně nastavit fázový posuv hodinového signálu při rychlosti 1000 Mbps. Rámce byly generovány pomocí Packet Builderu a zachytávány programem Wireshark – zde se ve výpisu objevovaly jak odeslané, tak přijaté rámce. Po srovnání jejich obsahu se dala kontrolovat funkčnost celého návrhu.

Pro otestování na živé počítačové síti jsem využil přepínače Catalyst 3750, jehož jeden port jsem nastavil do režimu SPAN – toto zajistilo funkcionalitu podobnou opakovači. Data jsem získával několika způsoby: přeposláním na sériovou linku, analyzátorem ChipScope a upraveným loopbackem.

Testování na reálném provozu potvrdilo funkcionalitu MAC jádra na rychlostech a provozních režimech odpovídající specifikaci.



Obrázek 5.2: GUI programu Wireshark. Na výstupu programu můžeme pozorovat zachycené pakety.



Obrázek 5.3: GUI programu Colasoft Packet Builder.

Kapitola 6

Závěr

Cílem mé diplomové práce bylo podle oficiálního zadání vytvořit v jazyku VHDL funkční MAC jádro pracující na rychlostech 100 a 1000 Mbps, provést verifikaci jednotlivých částí i celého jádra a demonstrovat funkci jádra na všech zadaných rychlostech.

Nejprve jsem prozkoumal podobná existující řešení a specifikoval jsem si, jaké bude mít mé jádro rozhraní. Poté následovalo dlouhé studium dokumentace [1]. Pro analýzu rámců jsem vytvořil jednoduchý packet-sniffer na platformě Spartan-3E. Po nasbírání potřebných informací jsem se dal do návrhu samotného MAC jádra. Během návrhu jsem každý modul verifikoval. Nakonec jsem dospěl k funkčnímu řešení. To jsem poté testoval jak na úrovni funkční simulace, tak i v reálném provozu. Pro demonstraci funkčnosti jsem udělal jednoduchý loopback na klientském rozhraní.

Všechny cíle zadání se mi podařilo splnit, navíc jsem navrhl funkční řešení pracující i na nejnižší rychlosti 10 Mbps a modul správy fyzické vrstvy, který komunikuje přes sběrnici MDIO.

„Rychlé ethernetové jádro“ najde uplatnění v dalších projektech, zabývajících se digitálním návrhem pro FPGA obvody, které budou vyžadovat komunikaci po síti Ethernet. Jádro lze využít jako celek nebo Tx či Rx modul pro konkrétní aplikaci samostatně.

Diplomová práce byla vysázena pomocí systému LaTeX.

Literatura

[1] IEEE 802.3. *Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.* New York : The Institute of Electrical and Electronics Engineers, Inc., 2005. [2696 s.] ISBN 0-7381-4741-9, SS95348.

[2] LAN83C185. *High Performance Single Chip Low Power 10/100 Mbps Ethernet Physical Layer Transceiver : Datasheet, Rev. 0.8.* Hauppauge (NY) : SMSC, 2008. 60 s. Dostupné z WWW: <http://www.smsc.com/media/Downloads_Public/Data_Sheets/83c185 .pdf>.

[3] 88E1111. *Alaska®Integrated 10/100/1000 Ultra Gigabit Ethernet Transceiver : Product Brief.* Santa Clara (CA) : Marvell Semiconductor, Inc., 2009. 52 s. Dostupné z WWW: <http://www.marvell.com/products/transceivers/alaska_gigabit_ether net_transceivers/Alaska_88E1111-002.pdf>. Doc. No. MV-S100649-00.

[4] Tri-Mode Ethernet MAC. *Xilinx®LogiCORE^{TM}IP Tri-Mode Ethernet MAC v4.4 : Product Specification.* San Jose (CA) : Xilinx, Inc., 2011. 24 s. Dostupné z WWW: <http://www.xilinx.com/support/documentation/ip_documentation/tri_mode_eth_ mac_ds297.pdf>. DS297.

[5] STAVINOV, Evgeni. A Practical Parallel CRC Generation Method. *Circuit Cellar : the magazine for computer applications* [online]. January 2010, Issue 234, [cit. 2011-05-07]. s. 38-45. Dostupný z WWW: <http://outputlogic.com/my-stuff/circuit-cellar-january-2010-crc.pdf>. ISSN 1528-0608.

[6] DP83861. *DP83861VQM-3 EN Gig PHYTER®10/100/1000 Ethernet Physical Layer.* Santa Clara (CA) : National Semiconductor, Oct 2009. 88 s. Dostupné z WWW: <https://www.national.com/ds/DP/DP83861.pdf>.

[7] *Total Phase* [online]. 2011 [cit. 2011-05-08]. MDIO Backround. Dostupné z WWW: <http://www.totalphase.com/support/kb/10042>.

[8] ADÁMEK, Jiří. *Kódování.* Vydání první. Praha : SNTL, 1989. 191 s.

[9] ML401. *ML401/ML402/ML403 Evaluation Platform : User Guide.* San Jose (CA) : Xilinx, Inc., May 24, 2006. 32 s. Dostupné z WWW: <http://www.xilinx.com/support/ documentation/boards_and_kits/ug080.pdf>. UG080 (v2.5).

[10] *OpenCores : the #1 community within open source hardware IP-cores* [online]. c2011 [cit. 2011-05-08]. Dostupné z WWW: <http://opencores.org>.

[11] DOLL, Stefan. *Stefan VHDL : VHDL Verification Course* [online]. 2011 [cit. 2011-05-08]. Dostupné z WWW: <http://www.stefanvhdl.com>.

[12] Virtex-4. *Virtex-4 FPGA : User Guide.* San Jose (CA) : Xilinx, Inc., December 1, 2008. 406 s. Dostupné z WWW: <http://www.xilinx.com/support/documentation/user_guides/ug070.pdf>. UG070 (v2.6).

[13] *Virtex-4 Family Overview : Product Specification.* San Jose (CA) : Xilinx, Inc., August 30, 2010. 9 s. Dostupné z WWW: <http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf>. DS112 (v3.1).

[14] *QuickHDL : User's and Reference Manual.* Wilsonville (OR) : Mentor Graphics Corporation, 1995. 258 s. Dostupné z WWW: <http://paws.kettering.edu/~mcdonald/class/ce422/qhdl>. Software Version 8.5_4.

[15] *Spartan-3E Starter Kit Board : User Guide.* San Jose (CA) : Xilinx, Inc., March 9, 2006. 164 s. Dostupné z WWW: <http://www.digilentinc.com/Data/Products/S3EBOARD/S3EStarter_ug230.pdf>. UG230 (v1.0).

Příloha A

Seznam použitých zkratek

CAN Campus Area Network

COL COLlision

CRS CaRrier Sense

CSMA/CD Carrier Sense Multiple Access with Collision Detection

DCM Digital Clock Manager

EDK Embedded Development Kit

FCS Frame Check Sequence

FPGA Field-Programmable Gate Array

GMII Gigabit Media Independent Interface

HDL Hardware Description Language

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

IP Core Intellectual Property Core

ISE Integrated Synthesis Environment

JTAG Joint Test Action Group

LAN Local Area Network

LGPL Lesser General Public License

MAC Media Access Control

MAN Metropolitan Area Network

MDIO Management Data Input/Output

MII Media Independent Interface

MTU Maximum Transmission Unit

PDU Protocol Data Unit

RGMII Reduced Gigabit Media Independent Interface

SFD Start Frame Delimiter

SGMII Serial Gigabit Media Independent Interface

VHDL Very-high-speed integrated circuits Hardware Description Language

VLAN Virtual LAN

Příloha B

Referenční příručka jádra Ethernet MAC

```
COMPONENT eth_mac
PORT(
 clientemactxd : IN std_logic_vector(7 downto 0);
 clientemactxdvld : IN std_logic;
 clientemactxifgdelay : IN std_logic_vector(7 downto 0);
 clientemactxunderrun : IN std_logic;
 rst : IN std_logic;
 txgmiimiiclk : IN std_logic;
 rxgmiimiiclk : IN std_logic;
 clk125 : IN std_logic;
 speed : IN std_logic;
 phyemacrxd : IN std_logic_vector(7 downto 0);
 phyemacrxdv : IN std_logic;
 phyemacrxer : IN std_logic;
 emacclientrxd : OUT std_logic_vector(7 downto 0);
 emacclientrxdvld : OUT std_logic;
 emacclientrxgoodframe : OUT std_logic;
 emacclientrxbadframe : OUT std_logic;
 emacclienttxack : OUT std_logic;
 speedis1000 : OUT std_logic;
 speedis10100 : OUT std_logic;
 gtx_clk : OUT std_logic;
 rx_client_clock : OUT std_logic;
 tx_client_clock : OUT std_logic;
 emacphytxd : OUT std_logic_vector(7 downto 0);
 emacphytxen : OUT std_logic;
 emacphytxer : OUT std_logic
 );
COMPONENT eth_mac;
```

Obecné vlastnosti:

- rychlost 10/100/1000 Mbps
- pouze režim full-duplex
- klopné obvody aktivní na náběžnou hranu
- asynchronní reset aktivní v log. 1
- vyvíjeno pro platformu Virtex-4
- testováno na desce ML401
- testováno s fyzickou vrstvou Marvell Alaska 88EE1111

Signál reset je aktivní v logické 1, všechny klopné obvody jsou synchronizovány na náběžnou hranu hodin.

Název signálu	**Směr**	**Význam**
clientemactxd(7:0)	IN	data určena k odeslání
clientemactxdvld	IN	platnost dat, signál musí být klientem vyšší vrstvy vystaven po celou dobu přenosu rámce
Clientemactxifgdelay(7:0)	IN	nastavení hodnoty inter-frame gap
clientemactxunderrun	IN	při vystavení během přenosu dojde k zahození právě vysílaného rámce
Emacclientrxd(7:0)	OUT	přijatá data
emacclientrxdvld	OUT	signál je aktivován po celou dobu příjmu
emacclientrxgoodframe	OUT	po skončení příjmu potvrzuje správnost přijatých dat
emacclientrxbadframe	OUT	po skončení příjmu signalizuje chybu v přijatých datech; dosud přijatá data je nutné zahodit
emacclienttxack	OUT	handshake pro vysílání dat, signál je vystaven po aktivaci clientemactxdvld
rx_client_clock	OUT	hodiny pro klienta vyšší vrstvy, perioda 800, 80 a 8 ns podle rychlosti
tx_client_clock	OUT	hodiny pro klienta vyšší vrstvy, perioda 800, 80 a 8 ns podle rychlosti

Tabulka B.1: Signály klientského rozhraní

Název signálu	**Směr**	**Význam**
emacphytxd(7:0)	OUT	data pro fyzickou vrstvu k odeslání, pro rychlosti 10 a 100 Mbps jsou využity spodní 4 bity
emacphytxen	OUT	platnost vysílaných dat
emacphytxer	OUT	signál vystaven v případě chyby
phyemacrxd(7:0)	IN	data přijatá z fyzické vrstvy, pro rychlosti 10 a 100 Mbps jsou využity spodní 4 bity
phyemacrxdv	IN	platnost přijímaných dat
phyemacrxer	IN	chyba během příjmu dat
txgmiimiiclk	IN	hodiny pro vysílání
rxgmiimiiclk	IN	hodiny pro příjem
gtx_clk	OUT	hodiny pro vysílání při rychlosti 1000 Mbps

Tabulka B.2: Signály rozhraní fyzické vrstvy

Název signálu	**Směr**	**Význam**
speedis1000	OUT	aktivní, pracuje-li jádro v režimu 1000 Mbps
speedis10100	OUT	aktivní, pracuje-li jádro v režimu 10 nebo 100 Mbps
clk125	IN	vstup pro hodinový signál o frekvenci 125 Mhz
speed	IN	0 pro režim 1000 Mbps, 1 pro ostatní rychlosti
rst	IN	reset, aktivní v log. 1

Tabulka B.3: Společné signály

Příloha C

Referenční příručka modulu MIIM

```
COMPONENT miim_sta
 PORT(
   clk : IN std_logic;
   rst : IN std_logic;
   mdio : INOUT std_logic;
   mdc : OUT std_logic;
   speed10 : OUT std_logic;
   speed100 : OUT std_logic;
   speed1000 : OUT std_logic
   );
END COMPONENT;
```

Jedná se o jednoduchý modul, který komunikuje pomocí sběrnice MDIO s fyzickou vrstvou a vyčítá konfigurační registry. V tomto případě zjišťuje, na jaké rychlosti komunikuje fyzická vrstva (registr 0x11).

Obecné vlastnosti:

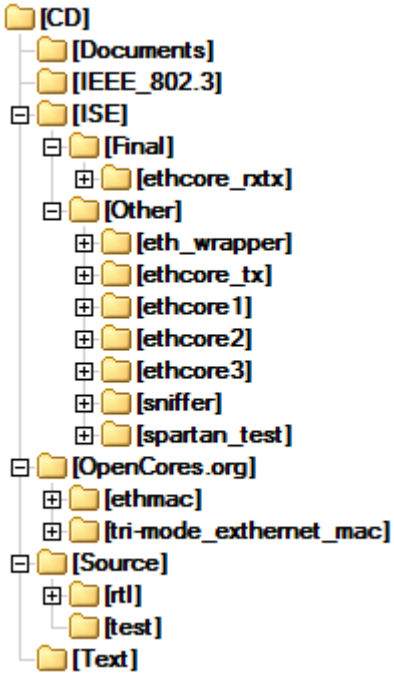
- komunikace po sběrnici MDIO
- implementováno vyčítání přenosové rychlosti
- klopné obvody aktivní na náběžnou hranu
- asynchronní reset aktivní v log. 1
- vyvíjeno pro platformu Virtex-4
- testováno na desce ML401
- testováno s fyzickou vrstvou Marvell Alaska 88EE1111

Tabulka C.1: Signály rozhraní modulu MIIM_STA

Název signálu	**Směr**	**Význam**
clk	IN	vstup pro hodinový signál o frekvenci 125 Mhz; frekvence může být i libovolně nižší
rst	IN	reset, aktivní v log. 1
mdio	INOUT	data, signál sběrnice MDIO, buzen třístavově
mdc	OUT	hodiny sběrnice MDIO
speed10	OUT	rychlost 10 Mbps
speed100	OUT	rychlost 100 Mbps
speed1000	OUT	rychlost 1000 Mbps

Příloha D

Obsah přiloženého CD



Obrázek D.1: Výpis obsahu přiloženého disku CD.

Documents	...	různé dokumenty, podklady
IEEE_802.3	...	standard IEEE 802.3
ISE	...	projekty programu Xilinx ISE
ISE/Final	...	finální projekt MAC jádra
ISE/Other	...	nižší verze a podpůrné projekty
OpenCores.org	...	projekty z portálu OpenCores.org
Source/rtl	...	zdrojové kódy pro syntézu
Source/test	...	zdrojové kódy pro testování
Text	...	text této diplomové práce v elektronické podobě