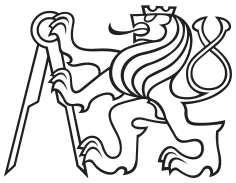
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra telekomunikační techniky

Ethernetové rozhraní na přípravku Nexys4 v jazyce VHDL

Bc. Pavel Gregar

Vedoucí práce: Ing. Pavel Lafata Ph.D.
2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Gregar** Jméno: **Pavel** Osobní číslo: **392920**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Sítě elektronických komunikací**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Ethernetové rozhraní na přípravku Nexys4 v jazyce VHDL

Název diplomové práce anglicky:

Ethernet port of Nexys4 kit in VHDL

Pokyny pro vypracování:

Seznamte se s přípravkem Nexys4 a možností jeho ovládání v jazyce VHDL. Zaměřte se zejména na standardní rozhraní Ethernet 100Base-T. Vytvořte vlastní VHDL projekt pro základní ovládání a funkce Ethernetového rozhraní, případně využijte dostupné funkce či IP cores. Navrhněte a vytvořte jednoduchou ukázku pro demonstraci využití rozhraní Ethernet na daném kitu.

Seznam doporučené literatury:

[1] Manuál kitu Nexys 4, dostupný z: https://reference.digilentinc.com/
[2] P. J. Ashenden: The Designer's Guide to VHDL / Edition 3, Elsevier Science, 2008
[3] D. Perry: VHDL: Programming By Example 4th Edition, McGraw-Hill Education, 2002

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Pavel Lafata Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.02.2017** Termín odevzdání diplomové práce: **26.05.2017**

Platnost zadání diplomové práce: **30.09.2018**

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Děkuji vedoucímu mé diplomové práce Ing. Pavlu Lafatovi za konzultace, připomínky a rady během vypracování této práce.

Dále děkuji rodičům a celé mé rodině za jejich podporu při studiu i v životě a mým přátelům a kolegům za jejich cenné rady a připomínky.

Prohlášení

Prohlašuji, že jsem zadanou diplomovou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé diplomové práce nebo její části se souhlasem katedry.
Datum:

Podpis: .....

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací obsluhy rozhraní Ethernet přípravku Digilent Nexys4 pomocí jazyka VHDL.

V práci jsou uvedeny možnosti ovládání tohoto rozhraní pomocí různých dostupných modulů a IP cores. Je v ní popsán vlastní vytvořený modul implementující funkcionalitu rozhraní a jeho porovnání s dostupnými variantami obsluhy tohoto rozhraní.

Poslední části práce je ukázka využití tohoto modulu pro ovládání rozhraní na daném přípravku Digilent Nexys4.

Klíčová slova: Ethernet, FPGA, VHDL, Digilent Nexys 4, MAC, MII, RMII, LAN8720A

Vedoucí práce: Ing. Pavel Lafata Ph.D.

Abstract

This diploma thesis deals with the design and implementation of Ethernet interface of Digilent Nexys4 kit using VHDL.

The paper presents the possibilities of controlling this interface using a variety of available modules and IP cores. The work describes the created module which implements the functionality of an interface and its comparison with the existing variants for managing this interface.

The last part of the thesis shows a sample demonstrating the use of this module to control the interface Digilent Nexys4 kit.

Keywords: Ethernet, FPGA, VHDL, Digilent Nexys 4, MAC, MII, RMII, LAN8720A

Title translation: Ethernet port of Nexys4 kit in VHDL

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V současné době je často žádoucí nebo dokonce nutné přenášet data mezi různými zařízeními, jako jsou například vestavěné (embedded) systémy, počítače a jiné digitální systémy, z důvodu jejich dalšího zpracování, archivace, sdílení, vizualizace a mnoha dalších rozličných důvodů závislých na požadavcích na daný systém.

Pro přenos těchto dat je možné využít technologie Ethernet, která je nyní velmi rozšířená v oblasti telekomunikačních sítí. Infrastruktura využívající technologií Ethernet je velmi rozšířená, nabízí několik možností rychlosti datového přenosu od jednotek Mbit/s po desítky Gbit/s, různé typy fyzických médií, možnost napájení zařízení technologií Power over Ethernet a podporuje přenos dat velkého množství protokolů využívaných pro přenos dat. Pomocí těchto protokolů je možné data přenášet s využitím sítě Internet a propojení zařízení, tak lze realizovat na globální úrovni.

Cílem této práce je implementovat rozhraní Ethernet na přípravku Digilent Nexys 4 pomocí vlastního modulu v jazyce VHDL, případě s využitím dostupných IP cores, a pomocí něho umožnit přenos dat mezi FPGA obvodem na přípravku a připojeným počítačem.

V teoretické části této práce je uveden popis technologie Ethernet a jejích vrstev, dále popis rozhraní použitých při vytváření implementace modulu uvedeného v praktické části práce. Také je zde uveden stručný popis použitého přípravku a vývojového prostředí, které bylo při implementaci rozhraní použito.

V praktické části je popsán vytvořený VHDL modul, implementující obsluhu ethernetového rozhraní na úrovni jednotlivých modulů a bloků s popisem jejich funkce a jejich vztahem k jednotlivým částem standardu Ethernet. Na konci práce je uveden demonstrační program využívající implementovaný VHDL modul pro obsluhu ethernetového rozhraní, zhodnocení funkce tohoto modulu a možnosti pro budoucí práci na jeho dalších rozšířeních.

Kapitola 2
Teoretická část

Tato kapitola je teoretickým úvodem do problematiky standardu Ethernet s hlavním zaměřením na vrstvu MAC (Media Access Control) a rozhraní MII (Media Independent Interface), specificky na jeho variantu RMII (Reduced MII). Dále jsou v této kapitole zmíněny informace o použitém přípravku Digilent Nexys 4 a na něm umístěném Ethernet transceiveru Microchip LAN8720A.

2.1 Ethernet

Termínem Ethernet se označuje řada technologií používaných převážně v lokálních sítích (LAN) souhrnně popisovaných standardem IEEE Std 802.3 (dále jen IEEE 802.3 nebo 802.3). Tento standard byl poprvé publikován v roce 1985 pracovní skupinou IEEE 802.3 standardizační organizací IEEE.

Tento standard navazoval na proprietární řešení poloduplexního algoritmu pro přístup k médiu (MAC - Media Access Control) nazývané CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Tento způsob přístupu k médiu byl vytvořen společností Xerox v laboratořích Palo Alto Research Center a datová rychlost rozhraní s využitím tohoto algoritmu byla 2,94 Mbit/s. Na toto řešení navázalo vytvoření specifikace, nyní známé pod názvem Ethernet II, s rychlostí 10 Mbit/s společnostmi Digital Equipment Corporation (DEC), Intel a Xerox v roce 1980.

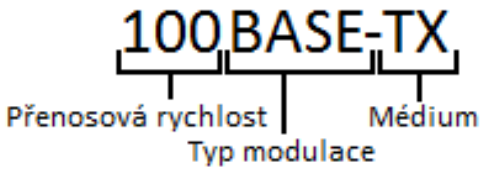
Toto řešení bylo uznáno standardizační organizací IEEE v roce 1983 a bylo publikováno pracovní skupinou IEEE 802.3 v roce 1985 jako standard IEEE Std 802.3-1985. Tento standard označovaný také 10BASE5 používal jako přenosové médium tlustý koaxiální kabel a přenos dat rychlostí 10 Mbit/s.

Na tento standard bylo v následujících letech navazováno dalšími pracovními skupinami IEEE 802.3 přidáváním nových druhů použitých médií (tenký koaxiální kabel, metalický kabel, optický kabel), režimů přenosu (plný duplex), rychlejších používaných datových rychlostí a dalších technologií (napájení prvků, řízení toku, nová rozhraní, ...). Nyní standard IEEE Std 802.3-2015 definuje fyzickou a spojovou vrstvu datové komunikace odpovídající referenčnímu modelu ISO/OSI pro přístup k médiu pro Ethernet realizovaný pomocí koaxiálního kabelu, optických vláken a různých druhů metalických kabelů.

Nyní se tento standard skládá z šesti sekcí a obsahuje popis fyzické a spojové vrstvy pro datový přenos od rychlostí 1 Mbit/s (bez uvažování technologií pro využití Ethernetu v přístupových sítích) po rychlost 100 Gbit/s a specifikace dalších přidaných možností použití Ethernetu (řízení toku, Energy-Efficient Ethernet, použití v přístupových sítích, ...).

Tento standard je v současné době využíván nejen v místních sítích LAN, ale i v sítích typu WAN a v následujících letech jsou naplánována další rozšíření tohoto standardu pracovními skupinami IEEE 802.3, které budou dále začleňovány do tohoto standardu. ([1], str. 21,22).

V současné době existuje velké množství variant Ethernetu v závislosti na tom jakou fyzickou vrstvu používají. Značení podle standardu IEEE 802.3 je následující ([1], str 61):



Obrázek 2.1: Značení Ethernetu

- **Přenosová rychlost** - udává rychlost v Mbit/s, v případě použití přípony G udává rychlost v Gbit/s,
- **Typ modulace** - udává jak jsou data modulována na médium, nyní většinou BASE - přenos v základním pásmu.
- **Médium** - udává použité fyzické médium a další parametry rozhraní, např. T - symetrický pár, F - optické vlákno, C - měděný kabel, ...

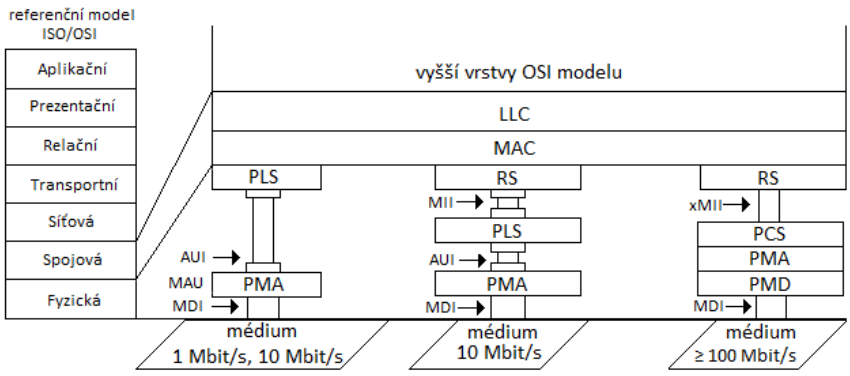
Podle tohoto značení, tedy verze Ethernetu uvedená na předchozím obrázku označuje Ethernet s přenosovou rychlostí 100 Mbit/s, který je přenášen v základním pásmu pomocí metalického symetrického kabelu (twisted pair).

2.2 Architektura sítě Ethernet

Architektura sítě Ethernet popsaná ve standardu IEEE 802.3 popisuje spojení na úrovni fyzické a spojové vrstvy referenčního modelu ISO/OSI. Spojová vrstva je zde navíc rozdělena na dvě části: podvrstvu LLC (Logical Link Control) a podvrstvu MAC (Medium Access Control).

- **LLC** je určena k realizaci přenosu bez spojení a se spojením přímo pomocí spojové vrstvy a k přepojování datových rámců mezi odlišnými realizacemi lokálních sítí (např. Ethernet a Token Ring). Přenos se spojením a bez spojení v současnosti většinou realizují vyšší vrstvy ISO/OSI modelu (transportní vrstva) a přepojování mezi různými realizacemi lokálních sítí, z důvodu rozšíření Ethernetu oproti ostatním realizacím lokálních sítí, již není tolik využíváno. Z těchto důvodu je tato podvrstva upozaděna a její největší význam má definování formátu Ethernetového rámce ve standardu 802.2, který tuto podvrstvu definuje.
- **MAC** je podvrstva která je zodpovědná za přenos dat z fyzické vrstvy a přenos dat na fyzickou vrstvu. Podvrstva slouží jako rozhraní mezi fyzickou vrstvou a podvrstvou LLC. V této podvrstvě je definována přístupová metoda k médiu CSMA/CD (Carrier Sense Multiple Access with Collision Detection).

Na obrázku 2.2 převzatého ze standardu pro Ethernet [1] je uvedeno základní dělení použitých vrstev a podvrstev architektury Ethernet v porovnání s referenčním modelem ISO/OSI a dělení použitých podvrstev sloužících k propojení podvrstvy MAC a fyzického média v závislosti na rychlosti ethernetového rozhraní.



Obrázek 2.2: Ethernet - porovnání s RM ISO/OSI

Propojení podvrstvy MAC a fyzické (PHY) vrstvy může být velmi rozdílné a závisí na používané datové rychlosti a používaných zařízeních. U nejstarších zařízení bylo používáno rozhraní AUI (Attachement Unit Interface) realizované 15-ti pinovým CANON konektorem a odpovídajícím AUI kabelem (levá část obrázku 2.2).

Ke konverzi logických signálů MAC vrstvy na fyzickou reprezentaci signálů AUI rozhraní je určena podvrstva PLS (Physical Layer Signaling). Následující podvrstva PMA (Physical Medium Attachment) provádí převedení signálu do podoby, která je výhodná pro konkrétní použitou fyzickou vrstvu. U těchto nejstarších zařízení byla realizována pomocí MAU (Medium Attachement Unit). MAU zde byla aktivní „T“ odbočka připojená ke koaxiálnímu kabelu a AUI sběrnicí k počítači. MDI (Medium Dependent Interface) je poslední podvrstvou, která závisí na použitém médiu a definuje jeho konkrétní fyzické parametry a způsob propojení k médiu. S tímto druhem ethernetového propojení pomocí AUI a MAU se v současnosti téměř nesetkáme, protože bylo nahrazeno novějšími způsoby realizace rozhraní mezi MAC a PHY a použitím jiného fyzického média.

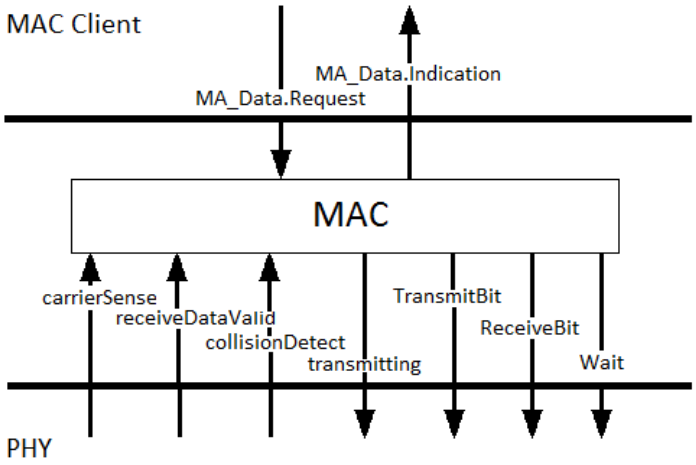
Standardizací 100 Mbit/s Ethernetu byl tento způsob propojení nahrazen rozhraním MII (Media Independent Interface). Původně sériové AUI rozhraní bylo nahrazeno paralelním, které je lépe uzpůsobeno pro další zpracování fyzickou vrstvou (kódování 4B/5B). Snížením rychlosti přenosu dat na desetinu je toto rozhraní navíc kompatibilní s 10 Mbit/s Ethernetem (prostřední část obrázku 2.2).

U rozhraní s rychlostí vyšší než 100 Mbit/s jsou použity větší datové šířky pro přenos dat, vyšší taktovací frekvence a při přenosu dat jsou využity obě hrany taktovacího signálu, což vede k možnosti přenosu v řádu Gbit/s. K těmto přenosům slouží například rozhraní GMII a XGMII.

Podrobné informace o jednotlivých detailech architektury pro různé druhy Ethernetu jsou uvedeny v příslušných částech standardu IEEE 802.3 a mnohonásobně přesahují rozsah této práce ([3], kapitola 7.2 a [1], str. 54-58).

2.3 MAC

V této kapitole je popsán abstraktní model podvrstvy MAC, která poskytuje svoje služby jejímu klientovi. MAC klientem může být jak podvrstva LLC, jak je to bylo definováno v prvních částech standardu Ethernet, tak i jiné entity, které využívají služby podvrstvy MAC na jiných vrstvách modelu ISO/OSI. Tento model je definován ve standardu IEEE 802.3 v části Clause 2, ze které je také převzat obrázek zobrazující model MAC 2.3.



Obrázek 2.3: Specifikace služby MAC

Služby MAC jsou zde popsány abstraktně a nemají žádnou konkrétní implementaci ani žádné definované rozhraní. Ty jsou definovány v následujících částech standardu u jednotlivých typů Ethernetu.

Zde definované služby MAC poskytují výměnu dat mezi klienty MAC vrstvy pomocí primitiv služby MA_Data.Request a MA_Data.Indication.

Primitiva služby MA_Data.Request slouží k předání dat (rámce) od klienta MAC fyzické vrstvě pro její odeslání na zadanou cílovou adresu. Dalšími parametry této primitivy jsou zdrojová adresa a kontrolní sekvence, která může být případně doplněna podvrstvou MAC.

Primitiva služby MA_Data.Indication slouží k příjmu dat (rámce) z fyzické vrstvy od MAC klienta na opačné straně spojení (peer client). Parametry této primitivy jsou stejné jako u předchozí primitivy s rozdílem, že kontrolní sekvenci není povinné předat MAC klientovi a jsou zde navíc předány informace o stavu přijatých dat (např. zda souhlasí kontrolní sekvence rámce).

Další informace o těchto primitivách služby, funkcích, procedurách a proměnných uvedených na obrázku 2.3 jsou uvedeny ve standardu IEEE 802.3 [1] v části SECTION 1 Clause 2,3 a 4 a jsou částečně popsány v dalších částech této kapitoly.

2.3.1 Formát rámce MAC

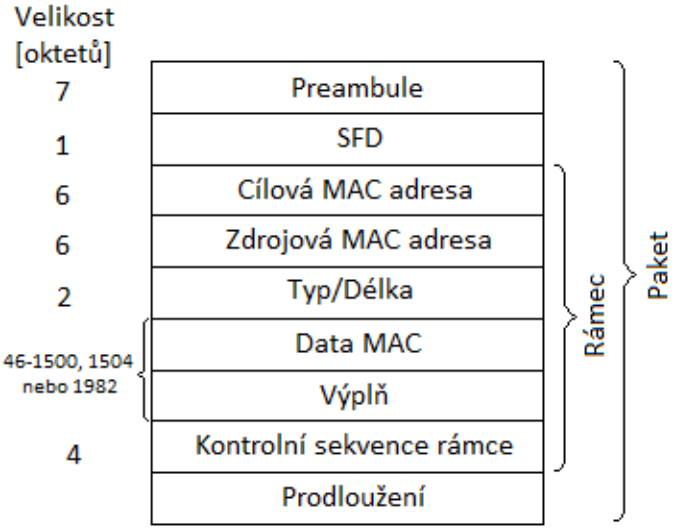
V této kapitole je popsán formát MAC rámce, popis jeho částí a způsob vytvoření Ethernet paketu, který je odeslán fyzické vrstvě (PHY) uvedený v [1] na stranách 108-113.

Během vývoje Ethernetu došlo k dalším úpravám formátu MAC rámce a z tohoto důvodu nyní existují tři typy MAC rámců:

- klasický rámec,
- Q-tagovaný rámec,
- envelope rámec.

Všechny typy MAC rámců mají podobnou strukturu, liší se pouze velikostí datové části a Q-tagovaný rámec má mezi zdrojovou MAC adresu a pole Typ/Délka vložen hlavičku 802.1Q o velikosti 4 oktetů.

Na následujícím obrázku je zobrazena struktura MAC rámce a z něj vytvořeného paketu pro PHY. Paket je zde zobrazen ve tvaru tabulky, kde řádky zobrazují jednotlivá pole paketu. Oktety paketu jsou odeslány v pořadí odshora dolů a odesílání jednotlivých oktetů začíná vždy nejméně významným bitem.

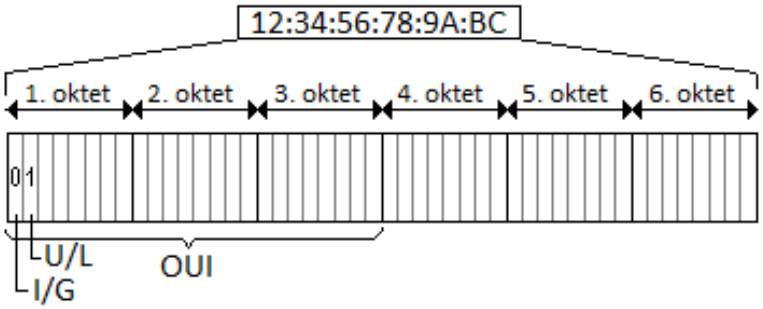


Obrázek 2.4: Formát MAC rámce a paketu

Preambule: První odesílanou částí paketu je preambule. Ta je tvořena sedmi oktety s binárním obsahem "10101010". Tento obsah byl důležitý u starších verzí Ethernetu jako 10BASE2, 10BASE5 a 10BASE-T. Pomocí této posloupnosti se synchronizovala smyčka fázového závěsu přijímače pro synchronizaci taktu mezi přijímačem a vysílačem. U novějších verzí Ethernetu již tato preambule ztrácí smysl, protože přijímající strana je v klidovém stavu kontinuálně synchronizována s vysílací. Z důvodu zpětné kompatibility je preambule u Ethernet paketu zachována.

SFD: Po preambuli následuje identifikátor začátku rámce (SFD - Start Frame Delimiter). Ten je tvořen bitovou posloupností "10101011". Tato posloupnost je pokračováním preambule následované dvěma bity nastavenými do logické ’1’. Tyto bity indikují začátek MAC rámce.

Cílová a zdrojová adresa : Dále následují adresová pole rámce obsahující adresu cílové a zdrojové stanice. Pole adresy cílové stanice specifikuje adresu stanice, pro kterou je MAC rámec určen a pole zdrojové adresy identifikuje stanici, která tento MAC rámec odeslala. Struktura adres je následující:



Obrázek 2.5: Struktura MAC adresy

Oba typy adres mají velikost 48 bitů a jsou složeny z osmi oktetů. Tyto adresy se většinou zapisují v hexadecimálním tvaru s oddělenými dvojicemi oktetů. Každý oktet adresy je odesílán od nejméně významného bitu (LSB). LSB prvního oktetu (první odesílaný bit rámce), na obrázku označený jako I/G, označuje u cílové adresy její typ. Pokud je tento bit nulový jedná se o individuální MAC adresu (unicast). V tomto případě adresa reprezentuje pouze jednu stanici, které je rámec s danou adresou určen. V případě že je bit I/G nastaven do logické ’1’, jedná se o skupinovou adresu, která se používá pro adresování více stanic (multicast). Speciálním případem je adresa složená ze samých logických ’1’, tedy v hexadecimálním tvaru adresa FF:FF:FF:FF:FF:FF, která adresuje všechny stanice (broadcast) a rámec s touto cílovou adresou je přijat všemi stanicemi v síti. U zdrojové adresy musí být tento bit nastaven do logické ’0’.

Druhý bit prvního oktetu, označený jako U/L, definuje u MAC adresy, zda se jedná o adresu lokální (L - Local) nebo globálně administrovanou (U - Universal). Lokální adresy mají tento bit nastaven do logické ’1’ a globální do logické ’0’. První tři oktety globálně administrované MAC adresy (označované jako OUI - Organizationally Unique Identifie) jsou unikátně přiřazovány jednotlivým výrobcům a organizacím za poplatek organizaci IEEE. Přiřazení zbývajících tří oktetů má na starost samotný výrobce a jediným požadavkem je zajištění unikátnosti přiřazovaným MAC adresám. Lokální adresa může být přidělena kýmkoliv, ale měla by být používána pouze v lokální síti.

V MAC rámci se v poli cílové adresy může objevit unicastová, multicastová nebo broadcastová adresa. V poli zdrojové adresy by vždy měla být adresa stanice odesílající daný rámec, tedy unicastová adresa.

Typ/Délka: Toto pole MAC adresy o velikosti dvou oktetů má v závislosti na jeho hodnotě jeden ze dvou významů:

- **Délka**: Pokud je hodnota v tomto poli menší nebo rovna hodnotě 1500 (5DC hexadecimálně) určuje toto pole velikost dat v následující datové části rámce.
- **Typ**: Pokud je hodnota v tomto poli větší než hodnota 1536 (600 hexadecimálně) určuje toto pole typ paketu vyšší vrstvy, který je v datové části MAC rámce. Hodnoty pro přiřazení hodnoty typ jednotlivým protokolům je spravováno organizací IEEE.

Data MAC: V tomto poli je umístěna sekvence oktetů obsahující datovou část MAC rámce. V datovém poli se může objevit sekvence libovolných oktetů s velikostí od 46 do 1500 oktetů (klasický rámec). Existují ještě Q-tagované rámce které mají maximální velikost 1504 oktetů a envelope rámce s maximální velikostí 1982 oktetů. Pokud je velikost přenášených dat menší než 46 oktetů je nutné z důvodu zajištění minimální velikosti MAC rámce, doplnit mezi data a následující kontrolní sekvenci (FCS) výplň, která zajistí prodloužení rámce na minimální velikost. Toto prodloužení je zde pro zajištění správného fungování CSMA/CD algoritmu.

Kontrolní sekvence rámce: Tato část rámce je tvořena čtyřmi oktety, jejichž obsah je generován pomocí cyklické redundantní kontroly (CRC). FCS složí k detekci chyb vzniklých při přenosu rámce. CRC je vypočítáno z chráněných částí MAC rámce, tedy cílové a zdrojové adresy, pole Typ/Délka, dat od klienta MAC a případné výplně. Obsah pole je vytvořen následujícím generujícím polynomem:

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

Výpočet hodnoty CRC je proveden následovně:

1. Prvních 32 bitů je komplementováno,
2. Hodnoty n bitů rámce jsou uvažovány jako koeficienty polynomu $M(x)$ stupně $n-1$, kde první bit rámce odpovídá koeficientu x^{n-1} a poslední bit koeficientu 1 (x^0) ,
3. Polynom $M(x)$ je vynásoben hodnotou x^{32} a vydělen polynomem $G(x)$,
4. Koeficienty zbytku polynomu $R(x)$ vzniklém po tomto dělení jsou uvažovány jako 32-bitová sekvence,
5. Tato sekvence je komplmentována a odeslána v poli FCS v pořadí odpovídajícím koeficientům $x^{31}, x^{30}, ..., x^0$.

Prodloužení: Tato část rámce je používána k prodloužení minimální velikosti rámce na fyzické vrstvě pro správnou funkci CSMA/CD algoritmu bez ovlivnění minimální velikosti datové části rámce. Používá se u poloduplexního přenosu dat rychlostí 1 Gbit/s. V této práci, která je zaměřena na přenos s nižší datovou rychlostí se tedy toto pole rámce nepoužívá.

2.3.2 Model funkce MAC

Model uvedený v kapitole 2.3 je v této části práce rozveden do popisu funkce podvrstvy MAC realizující CSMA/CD algoritmus pro přístup k médiu, popis tohoto algoritmu a jeho funkce při odesílání a příjmu MAC rámců.

Podvrstva MAC definuje služby nezávislé na médiu, které jsou poskytovány fyzickou vrstvou na definovaném fyzickém médiu a tyto služby poskytuje LLC podvrstvě nebo jinému klientovi MAC. Tyto služby může MAC poskytovat na fyzických médiích, na kterých je možné využít algoritmus přístupu k médiu CSMA/CD.

MAC vrstva provádí následující činnosti, které jsou obvykle spojeny s činností spojové vrstvy referenčního modelu ISO/OSI:

- **Zapouzdření dat** - vytváření rámců, přiřazení zdrojové a cílové adresy, detekci chyb při transportu,
- **Správa přístupu k médiu** - alokaci média, předcházení a řešení kolizí.

Podvrstva MAC může pracovat ve dvou různých režimech činnosti:

- plně duplexní režim(full-duplex)
- poloduplexním režim (half-duplex).

V poloduplexním režimu je médium sdíleno mezi komunikujícími stanicemi a k řízení přístupu je použit CSMA/CD algoritmus. Tento režim je dostupný na všech typech médií a je vyžadován na médiích, kde není možná současná obousměrná komunikace, například při použití 10BASE2 nebo 100BASE-T4.

Plně duplexní režim je možné využít za předpokladu, že médium podporuje současné vysílání a příjem, jako například u 100BASE-TX, v síti se nachází právě dvě stanice a tyto stanice jsou nakonfigurovány pro plně duplexní režim. V tomto případě je možné současné vysílání a příjem obou stanic zároveň a algoritmus CSMA/CD není použít ([1], str. 114).

2.3.3 Odesílání a příjem rámce

Tato podkapitola obsahuje popis odesílání a příjmu MAC rámců v plně duplexním i poloduplexním režimu a popis funkce algoritmu CSMA/CD tam, kde je při těchto činnostech využíván.

Jsou zde popisovány pouze části standardu IEEE 802.3, které mají souvislost s vypracováním této diplomové práce a nejsou zde uvedeny informace související s přenosy vyšších rychlostí (například prodloužení rámce - extension a řetězení rámců - frame bursting). Případný zájemce si tyto informace může vyhledat v příslušných pasážích uvedených ve standardu IEEE 802.3.

Odesílání rámce

Odesílání rámce je zahájeno příjmem požadavku od MAC klienta, který podvrstvě MAC poskytne data, která požaduje odeslat. Podvrstva MAC k těmto datům připojí preambuli, SFD, cílovou a zdrojovou adresu, pole Typ/Délka, zkontroluje velikost dat a pokud rámec nesplňuje minimální velikost, tak k datům přidá výplň. V případě že MAC klient neposkytne kontrolní sekvenci rámce, tak MAC tuto sekvenci vypočítá a připojí na konec rámce, v opačném případě připojí poskytnutou kontrolní sekvenci pro detekci chyb při přenosu.

Zjednodušené schéma níže popsaného algoritmu odesílání rámce při použití verzí Ethernetu s rychlostí 10 nebo 100 Mbit/s je na obrázku 2.6.

Takto vytvořený paket se poté podvrstva MAC pokusí odeslat. V poloduplexním módu MAC vrstva před odesláním monitoruje aktivitu signálu carrier sense, který je poskytovaný komponentou PLS. Tento signál je aktivní v době, kdy médium je využíváno. V tomto případě MAC vrstva vyčkává do doby, než dojde k uvolnění média.

Po uvolnění média je zahájeno odesílání po krátké pauze, která slouží pro správnou funkci CSMA/CD algoritmu a relaxaci fyzického média a přijímací elektroniky komunikujících stanic (u starších typů 10BASE2, 10BASE5 a 10BASE-T). U verzí Ethernetu s vyšší přenosovou rychlostí je tato mezera použita pro synchronizaci taktů přijímače a vysílače komunikujících stanic.

Podvrstva MAC poté vyšle fyzické vrstvě bitovou posloupnost pro odeslání na médium. Fyzická vrstva generuje odpovídající signál na médiu reprezentující vysílaný rámec. Během vysílání rámce monitoruje médium a MAC vrstvě poskytuje signál collisionDetect, informující o vzniku kolize na médiu v případě, že se o vysílání v této době pokouší i jiná stanice.

Pokud je odesílání rámce dokončeno a během něho nedojde k detekování kolize je klient MAC informován o úspěšném odeslání rámce a MAC vrstva čeká na další požadavek na vysílání. V plně duplexním módu nemůže dojít ke kolizi a veškeré signály indikující kolizi a obsazení média poskytované fyzickou vrstvou jsou v tomto režimu ignorovány. Rámec je vždy odeslán a CSMA/CD algoritmus není využíván.

V poloduplexním módu dojde ke kolizi v případě, že se více stanic pokusí vysílat ve stejnou dobu i přes jejich snahu o pozdržení vysílání. Ke kolizi může dojít v počáteční fázi odesílání rámce v době kratší, než má odesílaný

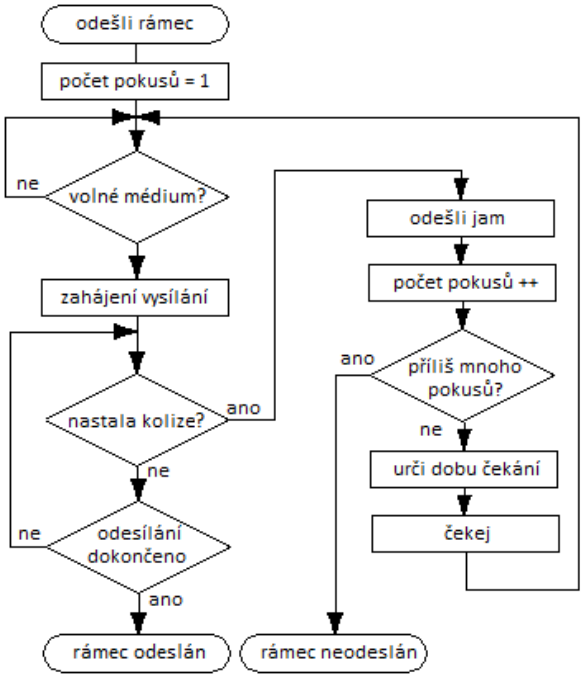
signál čas se rozšířit ke všem stanicím na médiu. Po uplynutí této doby by již ke kolizi nemělo dojít, protože správně pracující stanice by měli detekovat aktivitu na médiu a neměly by do ukončení vysílání začít vysílat.

Při vzniku kolize detekuje vysílající stanice interferenci na médiu a signálem collisionDetect informuje MAC o detekování kolize, kterou je zahájena obsluha řešení kolize. Na jejím počátku je ukončeno vysílání rámce a na médium je odeslána bitová sekvence označovaná jako jam.

Tato posloupnost zajišťuje, že se informace o vzniku kolize médiem rozšíří ke všem stanicím, které ji mohli způsobit. Stanice, které přijímají rámec při jehož odesílání došlo ke kolizi a sami nevysílají nemají možnost zjistit, že ke kolizi došlo. Jimi přijatý rámec poškozený kolizí bude zahozen, protože jeho velikost bude příliš malá z důvodu předčasného ukončení odesílání rámce při detekci kolize.

Po odeslání jamu je ukončeno vysílání a MAC čeká po náhodnou dobu. Tato doba je navíc závislá na tom kolikrát byla již při odesílání tohoto rámce detekována kolize. Postupné zvětšování intervalu náhodné doby čekání vede ke snižující se pravděpodobnosti, že dojde k další kolizi při opakovaném vysílání.

Toto opakované odesílání vede buď k úspěšnému odeslání rámce v době, kdy ostatní stanice zachycené v kolizi čekají nebo k vyčerpání počtu pokusů na odeslání rámce. V tomto případě je MAC klient informován, že se z důvodu vadného média nebo jeho přetížení nepodařilo rámec odeslat.



Obrázek 2.6: Odesílání MAC rámce - CSMA/CD algoritmus

Příjem rámce

Příchod rámce je v každé přijímající stanici detekován fyzickou vrstvou, která se s přicházející preambulí synchronizuje a poté indikuje právě přicházející rámec signálem receiveDataValid.

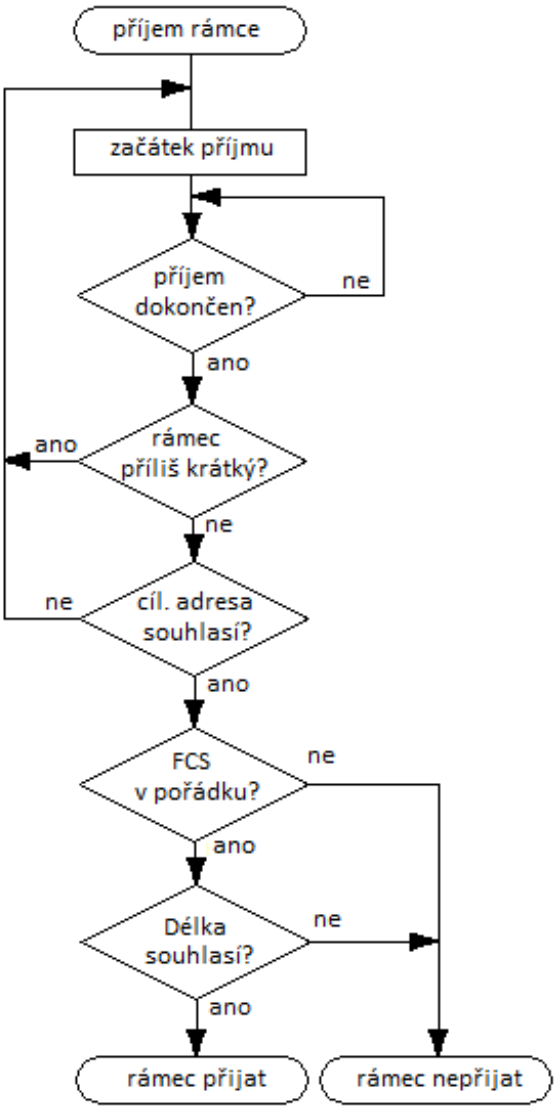
Přicházející signál je dekódován zpět na binární posloupnost, která je fyzickou vrstvou odesílána do podvrstvy MAC, která odstraní počáteční část rámce, tedy preambuli a SFD. Další přicházející bity jsou MAC ukládány po dobu kdy je signál receiveDataValid aktivní.

Po dokončení příjmu rámce, který je indikován zrušením nastaveni signálu receiveDataValid, je rámec podvrstvou MAC zkontrolován. Pokud je přijatý rámec menší, než je minimální možná velikost rámce (512 bitů - 64 oktetů) je tento rámec zahozen. Takové rámce (označované jako runt frames) vznikají při kolizi na médiu a následnému ukončení odesílání rámce, jak bylo uvedeno v předchozí části věnované odesílání MAC rámce při vzniku kolize.

Dále MAC u přijatého rámce kontroluje, zda je rámec určen pro přijímající stanici podle přijaté cílové adresy v rámci. Pokud adresa souhlasí je rámec předán k další kontrole. Podvrstva MAC poté porovná přijatou kontrolní sekvenci rámce (FCS) s hodnotou vypočítanou CRC z přijatých dat. Tím vrstva zkontroluje, zda přijatý rámec nebyl poškozen při přenosu. V případě že pole Typ/Délka přijatého rámce obsahuje informaci o délce rámce, MAC navíc zkontroluje, zda délka přijatého rámce souhlasí s délkou rámce udanou v tomto poli. Případně ověří, zda typ rámce (Ethertype) je určen pro připojeného klienta MAC ([1], str. 116, 122).

Pokud všechny uvedené kontroly rámce proběhnou s kladným výsledkem, je přijatý rámec předán klientovi MAC, v opačném případě je přijímaný rámec zahozen a klient MAC je informován o neúspěšném příjmu rámce.

Diagram zobrazující tuto kontrolu provedenou při přijmu rámce je uveden na obrázku 2.7.



Obrázek 2.7: Příjem MAC rámce

2.4 Rozhraní MII

Media Independent Interface (MII) poskytuje propojení mezi podvrstvou pro přístup k médiu (MAC) a entitami fyzické vrstvy (PHY) a mezi fyzickou vrstvou a entitami pro správu stanice (Station Management - STA). Rozhraní MII podporuje datový přenos rychlostí 10 Mbit/s a 100 Mbit/s s využitím přenosu dat po čtyřech bitech (označovaných jako nibble) pro vysílání a příjem dat.

Vztah MII rozhraní, referenčního modelu ISO/OSI a modelu IEEE 802.3 pro model sítě LAN s algoritmem CSMA/CD je uveden v kapitole 2.2 na obrázku 2.2 ve střední a pravé části.

Účelem tohoto rozhraní je poskytovat jednoduché, levné, lehce implementovatelné propojení mezi podvrstvou přístupu k médiu MAC a fyzickou vrstvou pro datové přenosy 10 Mbit/s a 100 Mbit/s.

Toto rozhraní má následující vlastnosti:

- Podpora datového přenosu s rychlostí 10 Mbit/s a 100 Mbit/s, podpora funkcí pro správu stanice,
- Změna dat je synchronní s referenčními hodinovými signály,
- Poskytuje nezávislé čtyřbitové datové cesty pro vysílaná a přijímaná data,
- Používá signály s TTL úrovní kompatibilní s CMOS technologií,
- Poskytuje jednoduché rozhraní pro správu stanice,
- Je schopné vybudit omezenou délku stíněného kabelu,
- Poskytuje plně duplexní režim činnosti.

Toto rozhraní je nezávislé na použitém fyzickém médiu a může být použito pro různé druhy nestíněných a stíněných symetrických metalických kabelů, optických kabelů a potenciálně i pro další média.

Rozhraní může být implementováno jako propojení integrovaných obvodů na desce plošných spojů, jako propojení dvou nebo více desek plošných spojů nebo jako rozhraní mezi zařízeními propojenými kabelem s odpovídajícím konektorem. ([2], str. 39, 45-46).

2.4.1 Signály rozhraní MII

Každý směr přenosu dat je na rozhraní MII uskutečňován pomocí 7 signálů: data (4 signály), hodinového taktu, oznámení chyby a signálu oddělovače (delimiter), dále rozhraní obsahuje 2 signály pro řešení kolizí a dva pro správu PHY.

Pro přenos z MAC do PHY se v tomto pořadí jedné o signály TXD(3:0), TX_CLK, TX_ER a TX_EN. Pro opačný směr pak RXD(3:0), RX_CLK, RX_ER a RX_DV. Další dva signály slouží k indikaci stavu média, signál CRS indikuje přítomnost nosné a COL informuje o přítomnosti kolize na médiu. Rozhraní pro správu stanice je poskytováno pomocí dvou signálů MDIO a MDC a poskytuje přístup k parametrům a službám pro řízení stanice pomocí PHY registrů.

Toto rozhraní tedy potřebuje celkem 18 signálů (16 nesdílených mezi více PHY), to může představovat problém u zařízení, kdy je potřeba využít více ethernetových rozhraní. Z tohoto důvodu bylo vytvořeno redukované MII rozhraní (RMII) jehož popis je uveden v kapitole 2.5. V této podkapitole byl použit popis signálů MII v [2], str. 51-58.

TX_CLK

Signál TX_CLK (transmit clock) je kontinuálně generovaný hodinový signál poskytující časovou referenci pro signály TXD(3:0), TX_EN, TX_ER určené pro odesílání dat do PHY. Zdrojem tohoto signálu je PHY.

Frekvence tohoto signálu by měla být čtvrtinová oproti přenosové rychlosti s povolenou odchylkou ±100 ppm. Tedy pro přenos dat rychlostí 100 Mbit/s musí být frekvence 25 MHz a pro přenos rychlostí 10 Mbit/s 2,5 MHz. Střída signálu by měla být mezi 35 a 65 %.

RX_CLK

Signál RX_CLK (receive clock) je kontinuálně generovaný hodinový signál poskytující časovou referenci pro signály RXD(3:0), RX_DV, RX_ER určené pro odesílání dat do PHY. Zdrojem tohoto signálu je PHY.

Střída signálu by měla být mezi 35 a 65%. V době kdy je signál RX_DV aktivní by přijímaná data měla být synchronní s tímto hodinovým signálem a jeho frekvence by měla odpovídat 25 % přenosové rychlosti (25 nebo 2,5 MHz).

TX_EN

Signál TX_EN (transmit enable) indikuje PHY vrstvě, že MAC vysílá data rozhraním MII. Tento signál by měl být nastaven synchronně s hodinovým signálem TX_CLK při prvním nibblu preambule a měl by zůstat nastaven do doby, kdy je odesílání celého rámce dokončeno. TX_EN by měl být zrušen před první hranou TX_CLK po posledním nibblu rámce.

TXD(3:0)

Signál TXD(3:0) (transmit data) je složen ze čtyř datových signálů, pomocí kterých jsou odesílány jednotlivé nibbly dat do PHY. Tento signál by měl být měněn synchronně s hodinovým signálem TX_CLK. V době kdy není nastaven signál TX_EN a TX_ER by PHY nemělo na tento signál reagovat s výjimkou definované signalizace pro PHY (např. nastavení PHY do EEE režimu).

TX_ER

Signál TX_ER (transmit coding error) složí k indikaci pro PHY, že má v právě odesílaném rámci odeslat symbol nebo více symbolů, které nejsou součástí platných dat. Nastavením tohoto signálu synchronně s hodinovým signálem TX_CLK na jeden nebo více hodinových taktů během odesílání rámce dojde k poškození odesílaného rámce, který nebude protější stranou vyhodnocen jako bezchybný. PHY nemusí zachovat relativní pozici v rámci vzhledem k nastavení tohoto signálu během odesílání. Pokud tento signál není pro funkci MAC potřebný měl by být kontinuálně nastaven v logické '0'.

RX_DV

RX_DV (receive data valid) indikuje, že PHY prezentuje dekódované nibbly přijímaného rámce na RXD(3:0) a tyto data jsou synchronní k hodinovému signálu RX_CLK. Tento signál by se měl měnit synchronně s RX_CLK a měl by být kontinuálně nastaven od prvního dekódovaného nibblu rámce do posledního nibblu rámce. RX_DV by měl být zrušen před první hranou hodinového signálu RX_CLK, poté co je na RXD vyslán poslední nibble rámce.

Aby byl přijatý rámec správně interpretován podvrstvou MAC musí aktivní signál RX_DV obklopovat celý rámec na RXD(3:0), začínat nejdéle v době SFD a končit před oddělovačem konce rámce (End-of-Frame Delimiter).

RXD(3:0)

Signál RXD(3:0) je složen ze čtyř datových signálů, které se mění synchronně s taktem RX_CLK. Každým taktem RX_CLK při kterém je RX_DV nastaven jsou na RXD(3:0) přenášeny čtyři bity dekódovaných dat z PHY do MAC. PHY by nemělo na RXD(3:0) reagovat v době, kdy není signál RX_DV aktivní.

RX_ER

Signál RX_ER je vybuzen PHY na dobu jednoho nebo více hodinových taktu RX_CLK k indikaci chyby (např. kódové chyby nebo jiné chyby, kterou dokáže PHY detekovat a která by jinak nebyla detekovatelná podvrstvou MAC) ke které došlo během příjmu rámce, který je právě odesílán pomocí

RXD(3:0) do podvrstvy MAC. V době kdy není nastaven signál RX_DV by RX_ER neměl mít vliv na MAC.

CRS

Signál CRS je nastaven PHY v době, kdy odesílací nebo přijímací médium není v klidovém stavu. PHY musí zajistit, že CRS je nastaven po celou dobu trvání případné kolize. Signál CRS nemusí být synchronní k RX_CLK nebo TX_CLK.

Chování signálu CRS není definováno pokud je rozhraní nastaveno do plně duplexního režimu.

COL

Signál COL je nastaven PHY v případě detekce kolize na médiu a měl by zůstat aktivní po dobu trvání kolize. Navíc je tento signál nastaven v 10 Mbit/s režimu v reakci na zprávu signal_quality_error z PMA. Signál COL nemusí být synchronní k RX_CLK nebo TX_CLK a jeho chování není definováno pokud je rozhraní v plně duplexním režimu, protože je používán pouze pro detekci kolize.

MDC

Signál MDC je generován z STA (část MAC pro management PHY) pro PHY jako časová reference k datovému signálu MDIO. MDC je aperiodický signál, který nemá definované žádné maximální doby trvání pro logickou '0' a logickou '1'. Minimální doba trvání obou logických úrovní je 160 ns a minimální perioda je 400 ns nezávisle na RX_CLK a TX_CLK.

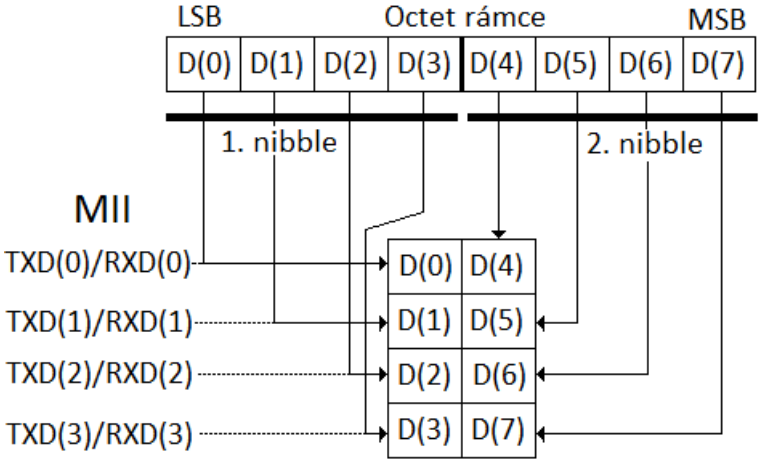
MDIO

Signál MDIO je obousměrný signál mezi PHY a STA. Je určen k přenosu řídících informací a statusu PHY a STA (viz 2.4.3) Řídící informace jsou předávány z STA a vzorkovány PHY synchronně k MDC. MDIO by mělo být buzeno pomocí třístavových budičů, z důvodu umožnění buzení signálu MDIO od PHY i od STA.

2.4.2 Přenos dat na MII

Paket odesílaný přes rozhraní MII mají následující formát:
<Inter-Frame><Preambule><SFD><Data><EFD>

Každý oktet rámce (odesílaný i přijímaný) je odesílán po jednotlivých nibblech s řazením bitů uvedeným na následujícím obrázku.



Obrázek 2.8: MII - přenos dat (řazení bitů)

Bity každého oktetu jsou odeslány nebo přijímány jako dva nibbly. Bity 0 až 3 odpovídají 1. nibblu, bity 4 až 7 odpovídají 2. nibblu.

Část **Inter-Frame** trvá časově nespecifikovanou dobu po kterou není na datové části rozhraní MII žádná aktivita způsobená odesílání nebo příjmem paketu. Tedy v době kdy je TX_EN nastaven v logické '0' pro odesílací část, případně kdy je RX_DV nastaven v logické '0' pro přijímací část.

Preambule obsahuje bitovou posloupnost:
10101010 10101010 10101010 10101010 10101010 10101010 10101010
Tato bitová posloupnost je uvedena v pořadí jak bude odeslána, tedy první bit oktetu (zleva) na TXD(0), druhý na TX(1), ..., poslední na TX(3).

SFD (Start Frame Delimiter) indikuje začátek rámce a následuje za preambulí. Jeho bitová hodnota je 10101011.

Data obsahují N oktetů dat odesílaných po 2xN nibblech. V případě kolize může být odeslán lichý počet nibblů.

Nastavení signálu TX_EN pro odesílání nebo RX_DV pro příjem dat představuje **EFD** (End of Frame Delimiter), ukončující přenos paketu na MII rozhraní.

2.4.3 MII Management Interface

MII Management Interface (rozhraní pro správu MII) je jednoduché rozhraní pro propojení entity STA (Station Management) a připojených PHY. Účelem tohoto rozhraní je ovládání PHY a získávání informací o jeho stavu. Rozhraní se také někdy označuje jako SMI (např. v datasheetu PHY obvodu použitého v této práci [7], str. 23).

Toto rozhraní je tvořeno dvěma signály (MDC a MDIO), které fyzicky přenášejí informace o správě PHY přes rozhraní MII. Standard IEEE 802.3 dále specifikuje formát rámců přenášejících tyto informace, protokol pro výměnu rámců a sady registrů, které lze číst a do kterých lze zapisovat pomocí těchto rámců.

MII používá dva základní registry (basic) a další nepovinné rozšiřující (extended). Základní registry jsou Control Register (adresa 0) a Status Register (adresa 1)., které by měly být implementovány ve všech PHY. Další registry jsou součástí rozšířené registrové sady (adresy 2 až 14), nebo jsou specifické pro výrobce PHY (adresy 15 až 31). Registry z rozšířené sady obsahují například řízení procesu Auto-Negotiation a informace o jeho průběhu nebo jedinečný identifikátor PHY. Podrobnější informace lze nalézt v [2] Clause 22.2.4.3 a 22.2.4.3.13.

Basic Control Register

Tento registr musí být umístěn na adrese 0. Registr slouží například k resetu PHY (bit 15), použití smyčky na MII (bit 14), zapnutí a restartu Auto-Negotation (bit 12 a 9), nastavení rychlosti a duplexu (pokud není zapnutý Auto-Negotiation) (bit 13 a 8). Podrobný popis registru je uveden v [2] Clause 22.2.4.1.

Basic Status Register

Tento registr musí mít adresu 1. Registr informuje MAC o tom, které režimy Ethernetu PHY podporuje (bity 15-9). Dále je zde uvedeno zda PHY podporuje rozšířený status (bit 8), jednosměrný přenos (bit 7), management rámce s potlačenou preambulí (bit 6), rozšířenou registrovou sadu (bit 0) a Auto-Negotiation (bit 3). Také jsou zde informace o dokončení Auto-Negotiation (bit 5), vzdálené chybě (bit 4), stavu spojení (bit 2) a deketovaný jabber (bit 1). Podrobný popis registru je uveden v [2] Clause 22.2.4.2.

Formát rámce

Formát rámce na rozhraní MII Management Interface je následující:
PRE-ST-OP-PHYAD-REGAD-TA-DATA-IDLE

Ve stavu IDLE (klidový stav) je MDIO nastavené ve stavu vysoké impedance a vlivem pull-up odporu PHY je MDIO v logické ’1’. Rámec začíná odesláním preambule sklájící se z 32 logických ’1’. Tato bitová posloupnost spolu s hodinovým signál na MDC generovaným STA slouží k synchronizaci PHY. Některé PHY tuto část rámce nepotřebují a rámec je možné začít rovnou sekvencí ST bez odeslání preambule.

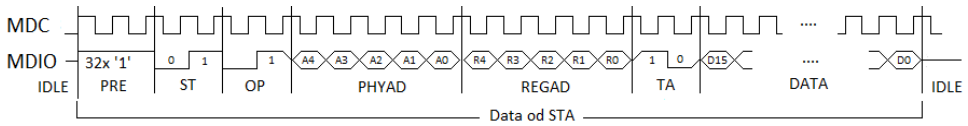
Sekvence ST (start of frame - začátek rámce) se skládá z přechodu do logické ’0’ a zpět do logické ’1’. Poté následuje OP (operation code - kód operace) <01> pro zápis nebo <10> pro čtení. PHYAD (PHY Address - PHY adresa) se skládá z pěti bitů a umožňuje adresování 32 různých PHY připojených k STA. Adresa je odesílána od MSB a STA musí předem znát adresy příslušných PHY v případě, že je jich připojeno více.

REGAD (register address - adresa registru) se také skládá z pěti bitů a složí k adresaci 32 unikátních registrů uvnitř každého připojeného PHY. Adresa registru je odesílána od MSB a musí používat adresy zmíněné výše (podle [2] Table 22-6).

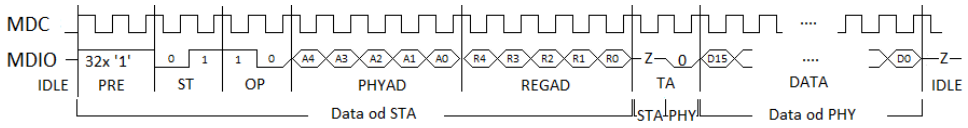
TA (turnaround) je dvou bitová mezera mezi adresou registru a datovou částí management rámce. Při čtení by STA i PHY měly v prvním bitovém intervalu zůstat ve stavu vysoké impedance. Při druhém bitu by PHY mělo nastavit MDIO do logické ’0’. Toto nastavení slouží k ověření, že sběrnice a PHY jsou schopné přenést data ze čteného registru. Během zápisu musí STA nastavit v prvním bitu TA logickou ’1’ a ve druhém logickou ’0’.

Poslední části management rámce je šestnáctibitové datové pole, které je na MDIO odesílané od MSB. Při čtení registru je tato část rámce generována PHY, při zápisu je generována STA.

Celý formát rámce pro zápis a čtení PHY registru je uveden na obrázcích 2.9 a 2.10.



Obrázek 2.9: MII - zápis registru na SMI



Obrázek 2.10: MII - čtení registru na SMI

2.5 Rozhraní RMII

Rozhraní RMII™ (Reduced Media Independent Interface™) je redukovanou variantou rozhraní MII a je použitelné pro propojení s fyzickou vrstvou (PHY) Ethernet podle IEEE 802.3. Bylo vytvořeno z důvodu redukce potřebných pinů pro propojení více PHY najednou u zařízení, která potřebují více Ethernetových portů.

RMII poskytuje méně pinovou alternativu k rozhraní MII specifikovanou ve standardu IEEE Std 802.3-2015 Clause 22 [2]. U rozhraní správy (MII Management Interface) se předpokládá, že je identické s MDC a MDIO uvedenými ve standardu IEEE 802.3. V této kapitole je čerpáno ze standardu [4] společnosti RMII Consortium.

Specifikace RMII má následující vlastnosti:

1. Podporuje datový přenos rychlostí 10 Mbit/s a 100 Mbit/s,
2. Používá jeden referenční hodinový signál z MAC do PHY (nebo externí pro MAC i PHY),
3. Poskytuje dvou bitový přenos vysílaných a přijímaných dat,
4. Používá TTL úrovní kompatibilní s CMOS technologií.

Rozdíly mezi RMII a MII jsou následující: dva hodinové signály TX_CLK a RX_CLK jsou nahrazeny jedním hodinovým signálem REF_CLK, Tento hodinový signál je vstupem do PHY, takže je možné jedním hodinovým signálem taktovat více PHY u víceportových zařízení. Frekvence tohoto hodinového signálu je dvojnásobná oproti frekvenci hodinových signálů MII rozhraní (tedy 50 MHz). To vede k úspoře 4 pinů rozhraní, protože použitá šířka datové sběrnice je snížena ze čtyř na dva bity. Signály RX_DV a CRS jsou sloučeny do signálu CRS_DV a signál COL je u RMII odstraněn.

Soupis všech signálů rozhraní RMII je uveden v následující tabulce a jejich popis je uveden v kapitole 2.5.1.

Signál	Směr (od MAC)	Šířka	Popis
REF_CLK	vstup/výstup	1	Hodinový signál pro PHY
TXD	výstup	2	Odesílaná data
TX_EN	výstup	1	Zapnutí odesílání TXD
RXD	vstup	2	Přijímaná data
CRS_DV	vstup	1	Data na RXD + detekce nosné
RX_ER	vstup	1	Chyba příjmu
MDC	výstup	1	MIIM - hodinový signál
MDIO	obousměrný	1	MIIM - data

Tabulka 2.1: Signály rozhraní RMII

2.5.1 Signály rozhraní RMII

REF_CLK

REF_CLK je kontinuálně generovaný hodinový signál poskytující časovou referenci pro signály TX_EN, TXD(1:0), CRS_DV a RXD(1:0). Tento signál je generován podvrstvou MAC nebo externím zdrojem a může být použit jako vstupní nebo výstupní, podle toho jestli je REF_CLK generován přímo podvrstvou MAC nebo zda je poskytován externím zdrojem (např. z PHY).

Každé PHY by mělo mít vstup na tento hodinový signál, ale je možné použít stejný hodinový signál pro více PHY. Frekvence REF_CLK by měla být 50 MHz ±50 ppm a jeho střída by měla být mezi 35 a 65 %.

Během odesílání z MAC využívá PHY tento signál jako hodinový takt pro data a nepotřebuje tedy žádné bufferovaní odesílaných dat. Při příjmu rámce může PHY obnovovat hodinový takt z přijatých dat, přijímač by měl počítat s odchylkou lokálního taktu z REF_CLK a obnoveného taktu z přijímaných dat a na vyrovnání použít dostatečně hluboké vyrovnávací paměti (elastického bufferu). Tento buffer by měl tolerovat ± 10 bitů dat (velikost bufferu 20 bitů) a data by neměla být na RXD odesílána dokud není buffer naplněn do poloviny.

TX_EN

Signál TX_EN indikuje, že MAC odesílá dibity na TXD(1:0) pro odeslání fyzickou vrstvou. Tento signál by měl být nastaven synchronně s REF_CLK současně s prvním dibitem preambule odesílaného rámce a měl by zůstat nastaven do odeslání posledního dibitu.

TXD(1:0)

Signál TXD(1:0) je složen ze dvou datových signálů pomocí kterých jsou odesílány jednotlivé dibity paketu do PHY. Tento signál by měl být měněn synchronně s hodinovým signálem REF_CLK. Pokud je signál TX_EN nastaven jsou data na TXD(1:0) přijímány PHY pro odeslání na médium.

V klidovém stavu (není nastaven TX_EN) by tento signál měl být nastaven na hodnotu "00", jiné hodnoty jsou určeny pro signalizaci pro PHY (například pro přepnutí do EEE módu).

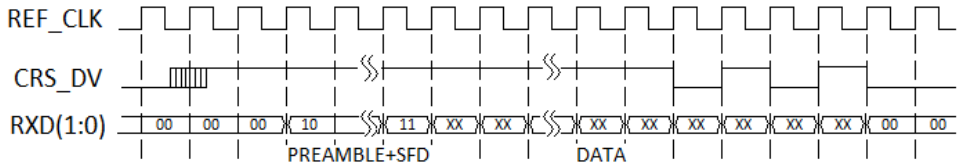
Data na TXD(1:0) by měla být platná tak, aby je při datové rychlosti 10 Mbit/s mohlo PHY vzorkovat každý desátý takt REF_CLK bez ohledu na to v které části dibitu tyto data vzorkuje.

CRS_DV

CRS_DV obsahuje sdružený signál RX_DV a CRS použitý v rozhraní MII. PHY nastavuje tento signál, když médium není v klidu. Definice klidového stavu pro 10BASE-T a 100BASE-TX je uvedena v IEEE 802.3.

CRS_DV je nastaven asynchronně při detekování nosné způsobené událostmi závislými na použité módu (10BASE-T nebo 100BASE-TX). Ztráta nosné způsobí zrušení nastavení signálu CRS_DV synchronně s REF_CLK se začátkem prvního nibblu bajtu po kterém je tato ztráta detekována.

Pokud má PHY po zrušení CRS_DV další bity ve vyrovnávací paměti nastavuje CRS_DV s frekvencí 25 MHz (nebo 2,5 MHz) v době, kdy vysílá na RXD(1:0) druhý dibit každého nibblu a ruší CRS_DV při prvním dibitu každého nibblu. Toto opětovné nastavování je prováděno, až do doby kdy PHY již nemá žádná data ve vyrovnávací paměti. Tento způsob indikování dalších bitů ve vyrovnávací paměti po ukončení příjmu rámce (konec CRS) je uveden na obrázku 2.11. Po skončení CRS jsou zde v bufferu čtyři dibity, které jsou odeslány přes RMII do MAC pomocí tohoto způsobu indikace na CRS_DV.



Obrázek 2.11: Příjem RMII - funkce CRSDV

V případě že dojde ke generování falešného nosného signálu (false carrier event) je CRS_DV nastaveno po celou dobu trvání této události spolu s hodnotou "10" na RXD.

RXD(1:0)

Signál RXD(1:0) je složen ze dvou datových signálů, pomocí kterých jsou přijímány jednotlivé dibity rámce z PHY. Tento signál je měněn synchronně s hodinovým signálem REF_CLK a s každým jeho taktem je na RXD(1:0) poslán jeden dibit přijímaného rámce.

Vzhledem k tomu, že frekvence REF_CLK je desetkrát větší než datová rychlost v režimu 10 Mbit/s je hodnota RXD(1:0) platná tak, aby signál RXD(1:0) bylo možné vzorkovat každý desátý takt bez ohledu na takt, při kterém se mění data.

MII obsahuje signál RX_ER, který zajišťuje propagováni chyby pokud PHY nedokáže dekódovat přijatý signál správně. Pro odstranění nutnosti používat tento signál by data na RXD(1:0) při zjištění chyby měly být nahrazeny hodnotou "01" do konce přijímaného rámce. Tato změna dat způsobí, že FCS přijatého rámce bude chybné a rámec bude zahozen. Pro odlišení této chyby od normální chyby CRC je nutné, aby PHY měl na adrese 15h (případně v adresovém prostoru 10h-1Fh) obsažen čítač chyb způsobující nastavení RX_ER.

RX_ER

Signál RX_ER je vybuzen PHY na dobu jednoho nebo více hodinových taktu RX_CLK k indikaci chyby (např. kódové chyby nebo jiné chyby, kterou dokáže PHY detekovat a která by jinak nebyla detekovatelná podvrstvou MAC), ke které došlo během příjmu rámce, který je právě odesílán pomocí RXD(1:0) do podvrstvy MAC. V době kdy není nastaven signál RX_DV by RX_ER neměl mít vliv na MAC.

Jak bylo zmíněno u popisu signálu RXD(1:0), tento signál není u RMII povinný z důvodu nahrazení dat při detekci chyby. Signál RX_ER signal je poskytován pro aplikace, které požadujíc jeho funkcionalitu.

MDC a MDIO

MDC a MDIO poskytují rozhraní pro správu PHY a standardní rozhraní pro přístup do registrů PHY. Funkcionalita signálů MDC a MDIO je stejná jako u rozhraní MII viz 2.4.1.

Detekce kolize

Signály CRS_DV a TX_EN společně indikují začátek odesílání a příjmu rámců a můžou být využity k indikaci kolize. MAC může obnovit signál COL logickým součinem signálu CRS obnoveného z CRS_DV a signálu TX_EN.

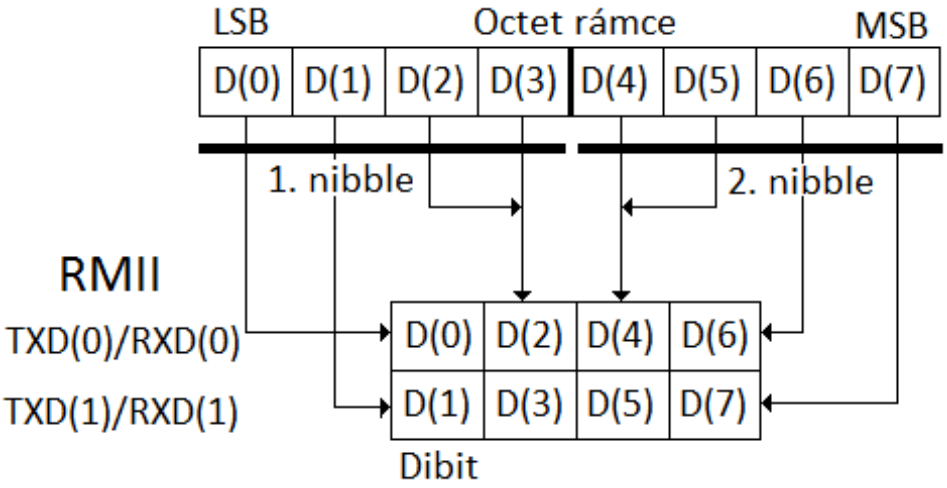
Nelze použít přímo signál CRS_DV, protože na konci rámce se tento signál může přepínat mezi logickou ’0’ a ’1’ v případě využití vyrovnávací paměti po ukončení CRS.

2.5.2 Přenos dat na RMII

Paket odesílaný přes rozhraní RMII mají následující formát (identický jako u rozhraní MII):

<Inter-Frame><Preambule><SFD><Data><EFD>

Rozdíl oproti rozhraní MII je pouze v řazení bitů při přenosu a šířce datového signálu. Každý oktet rámce (odesílaný i přijímaný) je odesílán po jednotlivých dibitech s řazením bitů uvedeným na následujícím obrázku.



Obrázek 2.12: RMII - přenos dat (řazení bitů)

Bity každého oktetu jsou odeslány a nebo přijímány jako čtyři dibity. Bity 0 a 1 odpovídají 1. dibitu, bity 2 a 3 odpovídají 2. dibitu, atd.

2.6 Použitý hardware a software

K realizaci ethernetového rozhraní je v této práci využit přípravek Digilent Nexys 4. Součástí přípravku Nexys 4 je obvod Microchip 10/100 Ethernet PHY (označení LAN8720A) propojený s ethernetovým konektorem. Popis přípravku, obvodu LAN8720A a vývojového prostředí využitého při realizovaní této práce je uveden v této kapitole.

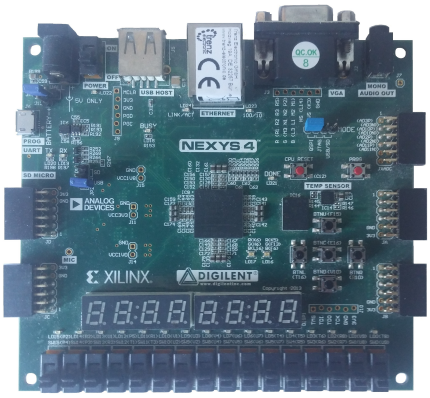
2.6.1 Digilent Nexys 4

Přípravek Digilent Nexys 4™ Artix-7 FPGA Board[1] (dále jen Nexys 4), vytvořený společností Digilent, Inc., je digitální vývojová platforma založená na programovatelném hradlovém poli (FPGA) od společnosti Xilinx Inc.

Základním prvkem přípravku je FPGA Artix-7™, ke kterému je připojeno velké množství různých periferií a komunikačních portů. Mezi periferie na přípravku patří přepínače, tlačítka, akcelerometr, senzor teploty, LED,sedmisegmentový displej, MEMS (mikroelektromechanický systém) mikrofon, reproduktor, slot na MicroSD kartu a DRAM paměť. Další periferie je možné připojit pomocí konektorů umístěných na přípravku.

Nexys 4 lze pomocí komunikačních konektorů, mezi než patří USB (Universal Serial Bus), VGA (Video Graphics Array), PS/2, audio jack a Ethernet, propojit s dalšími zařízeními a pomocí těchto konektorů s nimi komunikovat. Díky velkokapacitnímu hradlovému poli, rozličným portům a periferiím může přípravek Nexys 4 pojmout mnoho různých projektů a může sloužit k velkému množství odlišných aplikací.

Nexys 4 je kompatibilní s novým vývojovým prostředím Vivado® Design Suite a vývojovým prostředím ISE, jejichž součástí jsou další vývojové prostředky jako ChipScope a EDK. Společnost Xilinx nabízí volně stažitelnou verzi ISE, která byla využita v této práci pro programování hradlového pole přípravku a její popis uveden v kapitole 2.6.3 ([6], str. 1,2).



Obrázek 2.13: Přípravek Digilent Nexys 4

[1] Nexys 4™ - ochranná známka společnosti Digilent, Inc, ostatní ochranné známky a obchodní názvy zmíněné v této práci jsou majetkem příslušných vlastníků

2.6.2 Microchip LAN8720A

Přípravek Nexys 4 obsahuje obvod Microchip LAN8720A (dále označovaný jako PHY obvod). Tento obvod je 10BASE-T a 100BASE-TX transceiver fyzické vrstvy připojený k ethernetovému konektoru na přípravku.

Obsahuje plně duplexní 10BASE-T/100BASE-TX transceiver a podporuje přenosovou rychlost 10 Mbit/s (10BASE-T) a 100 Mbit/s (100BASE-TX). Obvod podporuje automatické nastavení přenosové rychlosti a duplexu (auto-negotiation) a možnost připojení pomocí kříženého nebo přímého kabelu díky Auto-MDIX. PHY obvod je v souladu se standardem IEEE 802.3 pro verze Ethernetu 10BASE-T a 100BASE-TX. ([7], str. 1,4)

PHY obvod podporuje komunikaci s Ethernet MAC vrstvou pomocí RMII rozhraní (viz kapitola 2.5). LED diody umístěné na přípravku (LD23 a LD24) propojené s PHY obvodem zajišťují zobrazení stavu spojení a použité přenosové rychlosti.

Po zapnutí napájení přípravku je PHY obvod nastaven do následující konfigurace dané fyzickým zapojením obvodu na přípravku.

- RMII mód,
- zapnuté auto-negotiation, podporované rychlosti 10 a 100 Mbit/s,
- PHY adresa = 00001 (pro MIIM)

Tuto konfiguraci lze měnit provedením hardwarového nebo softwarového resetu obvodu. Více informací o použití a vlastnostech obvodu lze nalézt například v datasheetu [7] nebo na webových stránkách výrobce.

Odvod obsahuje rozšířenou sadu PHY registrů podle IEEE 802.3 a registry specifikované výrobcem (popis je v [7] kapitola 4.0).

2.6.3 Vývojové prostředí

Pro vytvoření projektu, realizujícího obsluhu ethernetového rozhraní na přípravku Nexys 4, bylo využito vývojové prostředí ISE Design Suite 14.7 od společnosti Xilinx Inc.

Zdrojový kód realizující funkci modulu byl napsán v programovacím jazyce VHDL a projekt je primárně určen pro přípravek Nexys 4. Projekt nebyl vyzkoušen na jiných hradlových polích, ale po úpravě souboru ucf a modulu MMCM, který obsahuje IP core určené pro FPGA Artix-7, by bylo pravděpodobně možné tento projekt použít i na jiném hradlovém poli propojeném s podobným PHY obvodem s RMII rozhraním.

Zdrojové kódy, které jsou součástí práce obsahují celý projekt, který je možné v otevřít ve vývojovém prostředí ISE. Vlastní vytvořené zdrojové kódy a použité IP cores jsou umístěny ve složce /src.

Pro vyzkoušení vzorového projektu pro ovládání ethernetového rozhraní pomocí IP core TEMAC a soft-procesoru Microblaze, který je zmíněn v kapitole 3.1, bylo použito novější vývojové prostředí Vivado Design Suite.

K simulaci funkce obvodu byl během návrhu využíván simulační program ISim pro behaviorální simulaci a pro zjištění přesného časování výstupních signálů byla využita post-route simulace v tomtéž simulátoru.

Při návrhu byl dále využíván software ChipScope Pro. Ten umožňuje přidání IP core do navrhovaného projektu, které obsahuje integrovaný logický analyzátor, který je při syntéze obvodu umístěn do hradlového pole. Tento analyzátor poté umožňuje monitorování a zobrazení vnitřních signálů uvnitř projektu, čehož bylo využito pro testování přímo na přípravku bez nutnosti použití simulace.

Kapitola 3

Praktická část

V této kapitole je popsán návrh a implementace ovládání ethernetového rozhraní přípravku Digilent Nexys 4. Vlastní VHDL kód realizující obsluhu tohoto rozhraní je rozdělen do několika modulů a bloků pro větší přehlednost a možnost jednoduššího vylepšení nebo přidávání dalších funkcí rozhraní. Popis jednotlivých bloků je uveden na následujících stranách práce.

3.1 Možnosti realizace ovládání ethernetového rozhraní

V FPGA obvodu Artix-7 použitém na přípravku Nexys 4 je možné implementovat IP core Tri-Mode Ethernet Media Access Controller (TEMAC) [9], který slouží k implementaci MAC vrstvy architektury Ethernet. Pomocí této komponenty je možné implementovat funkcionalitu 10/100 Mbit/s, 1 Gbit/s nebo 10/100/1000 Mbit/s MAC vrstvy. TEMAC podporuje rozhraní MII, GMII, RGMII a SGMII pro komunikaci s připojeným PHY obvodem. Spojení IP core s vyšší vrstvou je realizováno pomocí AXI sběrnice.

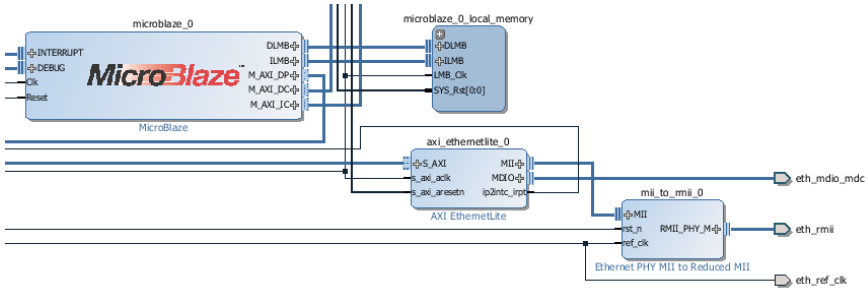
Přípravek Digilent Nexys 4 kvůli použitému PHY obvodu LAN8720A podporuje pouze rozhraní RMII, viz kapitola 2.6.2. Z tohoto důvodu je nutné k propojení TEMAC a PHY obvodu použít další IP core MII to RMII [10].

Vzorový projekt pro ovládání ethernetového rozhraní pomocí IP core a softprocesoru Microblaze, který realizuje echo server na zvolené IP adrese jsem realizoval v návrhovém prostředí Vivado.

Funkce tohoto vzorového projektu je následující: znaky naspané do terminálu propojeného s IP adresou přípravku, jsou odeslány do přípravku pomocí IP paketů. Pakety jsou přípravkem zpracovány a odeslány zpět na připojený terminál.

Popis a návod na realizaci toho echo serveru je možné nalézt v [8]. Ovládání rozhraní pomocí procesoru Microblaze je realizováno v jazyce C s využitím kompletního open-source TCP/IP stacku lwIP pro realizaci protokolů IP, ICMP, UDP, TCP, IGMP a ARP.

Část řešení tohoto vzorového projektu, zobrazující IP core ovládající ethernetové rozhraní, v návrhovém prostředí Vivado je na obrázku 3.1



Obrázek 3.1: Microblaze - echo server

Zadáním této práce je realizovat základní ovládání ethernetového rozhraní přípravku, a tak je tento způsob realizace příliš komplexní. Navíc použití této varianty s využitím IP core a soft-procesoru Microblaze by bylo vázané pouze na novější FPGA obvody, které tuto funkcionalitu podporují. Nebylo by ho možné použít ve starších, ale stále využívaných FPGA obvodech, jako je například Spartan-3. Použití IP core je navíc vázáno licencí od firmy Xilinx, která není běžně volně dostupná pro vývojové prostředí Xilinx ISE.

Z těchto důvodu bylo ovládání ethernetového rozhraní realizováno pomocí vlastního modulu komunikujícího s PHY obvodem přes rozhraní RMII. Tento modul je přenositelný pro různé obvody (s drobnými úpravami) a není zatížen použitím licencí pro IP cores.

Při návrhu programu pro ovládání ethernetového rozhraní bylo pro syntézu a implementaci využito vývojové prostředí zmíněné v kapitole 2.6.3.

Kód zajišťující funkcionalitu komunikačního rozhraní je napsán v jazyce VHDL a je určen pro přípravek Digilent Nexys 4, jehož popis je uveden v kapitole 2.6.1.

3.2 Projekt ethernet_top

Projekt ethernet_top je hlavním modulem této práce. Tento modul spojuje všechny ostatní moduly a bloky do jednoho celku a zajišťuje propojení FPGA obvodu s použitými periferiemi na přípravku Nexys 4 (PHY obvod, oscilátor, tlačítka a přepínače).

Uvnitř projektu ethernet_top jsou další moduly, které jsou definované v následujících zdrojových souborech:

- MMCM.sch,
- MIIM.sch,
- MAC.sch,

Každý z těchto souborů definuje skladbu jednoho z modulů, jehož význam a funkce je popsána v následujících podkapitolách. Celé blokové schéma modulu ethernet_top.sch je uvedeno na konci práce v příloze A.

Modul MMCM slouží k vytvoření potřebných hodinových a resetovacích signálu, jejich rozvodu uvnitř FPGA obvodu a vyvedení hodinového signálu REF _CLK do PHY obvodu.

Modul MAC a MIIM implementují podvrstvu MAC a RMII rozhraní s PHY obvodem na přípravku Nexys 4. Použité piny FPGA obvodu Artix-7 definované v souboru ethernet_top.ucf pro propojení s PHY obvodem jsou uvedeny v následující tabulce.

Port	Pin obvodu Artix-7
PHY_REF_CLK	D5
PHY_RESET	B3
PHY_TXD(1:0)	A8, A10
PHY_TXEN	B9
PHY_RXD(1:0)	C11, D10
PHY_CRS_DV	D9
PHY_RX_ER	C10
PHY_MDIO	A9
PHY_MDC	C9

Tabulka 3.1: Mapování vývodů obvodu Artix-7 na PHY obvod LAN8720A na přípravku Digilent Nexys 4

3.2.1 Modul MMCM

Modul MMCM generuje hlavní hodinový signál CLK_50MHZ pro ostatní moduly projektu a hodinový signál PHY_REF_CLK pro PHY obvod LAN8720A.

Modul dále vytváří resetovací signál RESET_50MHZ pro reset všech ostatních modulů. Tento reset je generován při stisku tlačítka připojeného na port RESET nebo při ztrátě uzamčení MMCM (signál LOCKED).

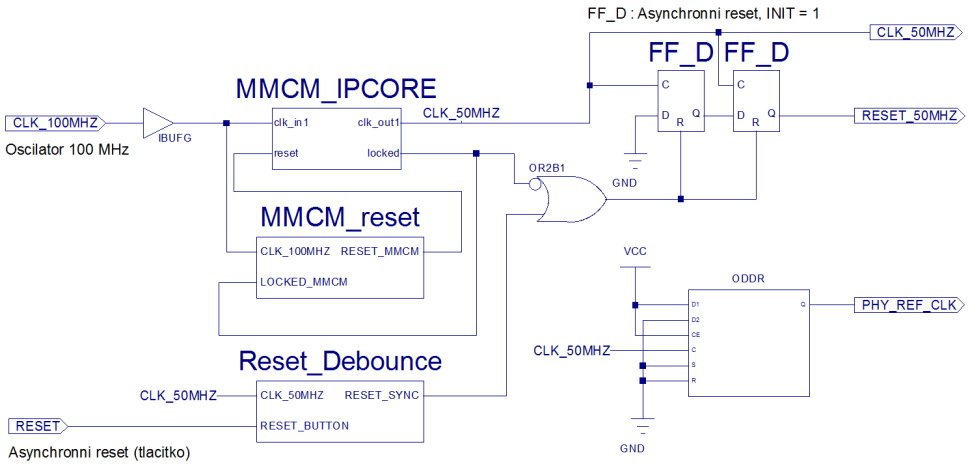
Pro správnou funkci modulu MMCM a celého projektu je potřeba použít hodinový signál s frekvencí 100 MHz přivedený na port CLK_100MHZ.

Vstupní a výstupní porty modulu jsou uvedeny v následujících tabulce:

Port	Směr	Šířka	Popis
CLK_100MHZ	vstupní	1	Hodinový signál, $f = 100$ MHz
RESET	vstupní	1	Asynchronní reset (tlačítko)
CLK_50MHZ	výstupní	1	Hlavní hodinový signál projektu
RESET_50MHZ	výstupní	1	Hlavní resetovací signál projektu
PHY_REF_CLK	výstupní	1	Hodinový signál pro PHY obvod

Tabulka 3.2: Modul MMCM - porty

Blokový diagram modulu MMCM je uveden na obrázku 3.2.



Obrázek 3.2: Modul MMCM - blokové schéma

Vstupní hodinový signál je vstupním bufferem IBUFG přiveden na vstup clk_in1 primitivy MMCM [11], která zde slouží pro syntézu hodinového signálu s frekvencí 50 MHz, filtrování fázového chvění (jitter) a odstranění skluzu hodinového signálu (clock skew) uvnitř FPGA. Na blokovém schématu je tato primitiva uvedena s názvem MMCM_IPCORE.

MMCM je nastaven na dělení frekvence vstupního signálu dvěma. Vnitřní zpětná vazba je realizována uvnitř MMCM_IPCORE a na blokovém schématu není uvedena. Výstupní signál LOCKED tohoto bloku je po zapnutí obvodu nastaven do logické '1', poté co MMCM začne korektně generovat svůj výstup, a není tedy nutné po zapnutí primitivu MMCM resetovat.

Pokud během činnosti dojde k výpadku hodinového signálu nebo k jeho fázové chybě je signál LOCKED nastaven do logické '0' a primitivu MMCM je nutné resetovat. ([11], str. 83).

Po zapnutí obvodu je signál RESET_50MHZ nastaven do logické '1' kvůli počáteční hodnotě výstupu klopných obvodů typu D (FF_D v blokovém schématu). Pokud je signál LOCKED nastaven do logické '1' a není tlačítkem na portu RESET vyvolán reset (viz kapitola 3.2.1) přejde po 2 hodinových taktech signálu CLK_50MHZ resetovací signál RESET_50MHZ do logické '0' a tím dojde ke spuštění všech ostatních modulů projektu.

Pokud je blokem MMCM_IPCORE nastaven port LOCKED do logické '0' jsou zároveň asynchronně resetovány oba klopné obvody FF_D. Tím je jejich výstup nastaven do logické '1', čímž je generován reset na portu RESET_50MHZ pro ostatní moduly v době, kdy není hlavní hodinový signál z MMCM správně generován.

Další informace o rozvodu a vytváření hodinových signálu uvnitř použitého FPGA a informace o MMCM lze nalézt v [11].

Hodinový signál pro PHY obvod je generován pomocí primitivy ODDR. Jedná se o registr v bloku OLOGIC, který implementuje výstupní DDR registr. Blok OLOGIC umístěný u vstupně-výstupního bloku (IOB), je synchronní blok určený k odesílání dat z FPGA.

Primitiva ODDR může být využita pro generování kopie hodinového signálu použitého uvnitř FPGA na jeho výstupní pin. Toho je dosaženo nastavením vstupu D1 do logické '1' a vstupu D2 do logické '0'. Xilinx doporučuje používat tento způsob pro přenos hodinového signálu z FPGA na výstupní piny ([12], str. 127-128)

Použitý ODDR v modulu MMCM má pevně nastavené piny S, R a D2 do logické '0' a vstupy D1 a CE do logické '1'. Vstup C je spojen s hodinový signál na generovaným pomocí MMCM. Tím je primitiva nastavena na kontinuální kopírování hodinového signálu z FPGA na výstupní pin, který je spojen s pinem REF_CLK PHY obvodu LAN8720A.

Blok MMCM_reset

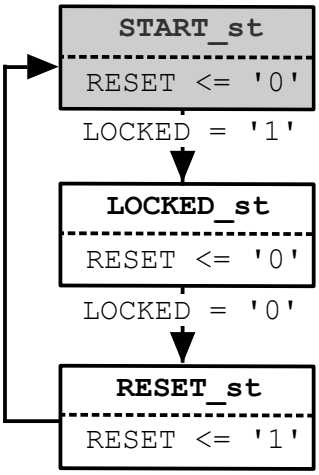
K resetu MMCM slouží blok MMCM_reset, který při ztrátě uzamčení MMCM (signál LOCKED) generuje resetovací pulz pro MMCM. Tento blok je realizován jednoduchým stavovým automatem s následujícím stavy a s diagramem přechodů.

Vstupní a výstupní porty bloku jsou uvedeny v následujících tabulce:

Port	Směr	Šířka	Popis
CLK_100MHZ	vstupní	1	Hodinový signál, $f = 100$ MHz
LOCKED_MMCM	vstupní	1	Indikace uzamčení MMCM
RESET_MMCM	výstupní	1	Reset MMCM

Tabulka 3.3: Blok MMCM_reset - porty

- START_st - počáteční stav ve kterém automat čeká na nastavení signálu LOCKED do logické '1' (automaticky po naprogramování FPGA),
- LOCKED_st - stav korektní funkce MMCM, kde je testováno trvající uzamčení MMCM,
- RESET_st - stav generující resetovací pulz pro blok MMCM



Obrázek 3.3: Blok MMCM_reset - diagram přechodů FSM

VHDL proces realizující stavový automat je uveden na obrázku 3.4, který se nachází na následující straně.

Po zapnutí obvodu čeká blok MMCM_reset ve stavu START_st na nastavení signálu LOCKED do logické '1' (řádek 10). V tomto případě přejde stavový automat bloku do stavu LOCKED_st, ve kterém je blok při normální funkci MMCM. V těchto stavech je výstup RESET_MMCM nastaven do logické '0' (řádek 1). Pokud během funkce obvodu dojde ke stavu, který zapříčiní nastavení signálu LOCKED do logické '0' přejde stavový automat do stavu RESET_st (řádek 16). V tomto stavu je generován impulz na portu RESET_MMCM o délce 10 ns (podle [11] minimální délka resetovacího impulzu je 5 ns), který resetuje MMCM (blok MMCM_IPCORE) a modul se vrací do stavu START_st, kde čeká na opětovné uzamčení MMCM.

```
1  RESET_MMCM <= '1' when state = RESET_st else '0';
2
3  process(CLK_100MHZ)
4
5  begin
6    if rising_edge(CLK_100MHZ) then
7      case state is
8        when START_st =>
9          -- otestovani jestli jiz doslo k uzamceni MMCM
10         if LOCKED_MMCM = '1' then
11           state <= LOCKED_st;
12         else
13           state <= START_st;
14         end if;
15       when LOCKED_st =>
16         if LOCKED_MMCM = '0' then
17           state <= RESET_st;
18         else
19           state <= LOCKED_st;
20         end if;
21
22       when RESET_st =>
23         -- staci jeden hodinovy cyklus - minimalni doba resetu
24         -- je 5 ns (datasheet 7-Series Clocking)
25         state <= START_st;
26     end case;
27   end if;
28 end process;
```

Obrázek 3.4: Blok MMCM_reset - FSM

Blok Reset_Debounce

K synchronizaci resetovacího signálu na vnitřní hodinový signál CLK_50MHZ a kvůli odstranění falešných stisků tlačítka způsobených zákmity slouží blok Reset_Debounce.

Signál z tlačítka, který je použit pro resetování celého projektu je připojen k mechanickému tlačítku na přípravku Nexys 4. Z tohoto důvodu je tento signál asynchronní a dochází u něj při stisku k zákmitům, které můžou vyvolat několikanásobný resetovací impulz.

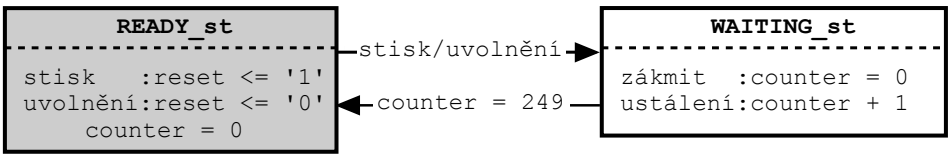
Vstupní a výstupní porty bloku jsou uvedeny v následujících tabulce:

Port	Směr	Šířka	Popis
CLK_50MHZ	vstupní	1	Hodinový signál, $f = 50$ MHz
RESET_BUTTON	vstupní	1	Připojené tlačítko
RESET_SYNC	výstupní	1	Synch. tlačítko bez zákmitů

Tabulka 3.4: Blok Reset_Debounce - porty

Tento blok je realizován jednoduchým stavovým automatem s následujícím stavy a s diagramem přechodů:

- READY_st - počáteční stav ve kterém automat čeká na stisk nebo uvolnění tlačítka,
- WAITING_st - stav čekání na ustálení zákmitů tlačítka.



Obrázek 3.5: Blok Reset_Debounce - diagram přechodů FSM

V počátečním stavu READY_st čeká stavový automat bloku na rozdílné hodnoty signálů, které jsou generovány po synchronizaci dvěma klopnými obvody typu D při stisku nebo uvolnění tlačítka. Tyto klopné obvody zároveň zajišťují synchronizaci asynchronního signálu z tlačítka do hodinové domény CLK_50MHZ. Tyto klopné obvody mají v ucf souboru definován parametr ASYNC_REG, jak je uvedeno na obrázku níže.

```
1 INST "MMCM_1/Reset_Debounce_1/button_delay_0" ASYNC_REG = "TRUE";
2 INST "MMCM_1/Reset_Debounce_1/button_delay_1" ASYNC_REG = "TRUE";
```

Obrázek 3.6: Deklarace parametru ASYNC_REG

Tento parametr informuje vývojové prostředí, že klopný obvod je určen pro zpracování asynchronního signálu na pinu D vzhledem k hodinovému signálu. Parametr také zajišťuje umístění těchto klopných obvodů co nejblíže u sebe pro minimalizaci zpoždění mezi výstupem prvního a vstupem druhého klopného obvodu, čímž je maximalizován čas pro odeznění metastabilního stavu.

Při stisku tlačítka-* ve stavu READY_st je výstup RESET_SYNC nastaven do logické '1' a stavový automat přejde do stavu WAITING_st. V tomto stavu je využit signál counter sloužící jako synchronní čítač. Pokud dojde k zákmitu tlačítka a signál z obou synchronizačních klopných obvodů není stejný, dojde k resetování čítače.

Pokud po dobu 250 hodinových taktů tj. 5 μs (pozn. tento čas byl určen pozorováním zákmitů tlačítka na osciloskopu, které trvaly max. 2 μs) nedojde k zákmitu a výstup klopných obvodů zůstane stabilní, přejde stavový automat zpět do stavu READY_st. V tomto stavu stavový automat čeká na uvolnění tlačítka, kdy dojde při přechodu do stavu WAITING_st k nastavení výstupu RESET_SYNC do logické '0'.

V tomto čekání ve stavu WAITING_st při uvolnění tlačítka je znovu zahájeno čekání po dobu 250 hodinových taktů na ustálení zákmitů tlačítka. Po této době se stavový automat vrací zpět do stavu READY_st a je připraven na další stisk tlačítka.

```
1   process(CLK_50MHZ)
2   begin
3     if rising_edge(CLK_50MHZ) then
4       case state is
5         when READY_st =>
6           counter <= 0;
7
8           if button_delay(0) /= button_delay(1) then
9             if button_delay = "01" then
10              reset <= '1';
11              state <= WAITING_st;
12            elsif button_delay = "10" then
13              reset <= '0';
14              state <= WAITING_st;
15            end if;
16          end if;
17
18        when WAITING_st =>
19          -- ignorovani tlacitka po dobu nez dobehne citac
20          if button_delay(0) /= button_delay(1) then
21            -- reset citace pri zakmitech na tlacitku
22            counter <= 0;
23          else
24            counter <= counter + 1;
25          end if;
26
27          if counter = 249 then
28            -- ustalene dostatene dlouho, navrat do READY_st
29            -- 5 us - maximalni zakmit na OSC byl asi 2 us
30            state <= READY_st;
31          end if;
32      end case;
33    end if;
34  end process;
```

Obrázek 3.7: Blok Reset_Debounce - FSM

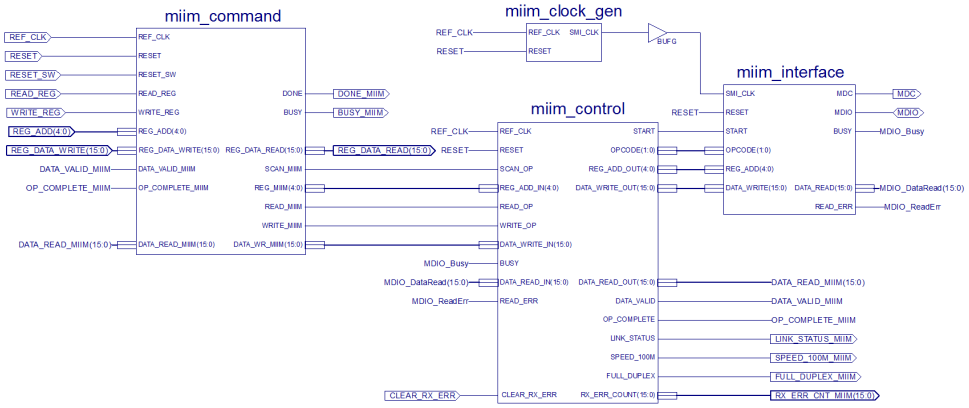
3.2.2 Modul MIIM

Modul MIIM (Media Independent Interface Management) slouží ke komunikaci s PHY obvodem LAN8720A pomocí MII Management rozhraní (podle IEEE 802.3), také označované jako SMI (Serial Management Interface). Tímto modulem je možné číst a zapisovat data z registrů PHY obvodu (viz kapitola 2.4.3). Další funkcí je provedení softwarového resetu a nastavení volitelné konfigurace PHY obvodu.

Modul MIIM se skládá z následujících bloků a jeho blokový diagram je uveden na obrázku 3.8:

- miim_command,
- miim_control,
- miim_clock_gen,
- miim_interface,

Blok miim_command složí ke zpracování požadavku na komunikaci z vyšší vrstvy. Blok miim_control zpracovává požadavek z bloku miim_command a řídí blok miim_interface, který zajišťuje komunikaci s PHY obvodem pomocí SMI rozhraní. Poslední blok miim_clock_gen zajišťuje vytvoření hodinového signálu pro komunikaci na SMI rozhraní. Detailní popis portů a funkce jednotlivých bloků je uveden v samostatných částech této kapitoly.



Obrázek 3.8: Modul MIIM - blokové schéma

Pro správnou funkci modulu MIIM je potřeba použít hodinový signál s frekvencí 50 MHz přivedený na port REF_CLK. Tento hodinový signál je v projektu generován modulem MMCM (viz kapitola 3.2.1)

Porty modulu jsou uvedeny v následujících tabulkách.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál modulu MIIM
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.5: Modul MIIM - hodinové a resetovací porty

Porty v tabulce 3.6 jsou použity pro komunikaci modulu MIIM s vyšší vrstvou, která tento modul využívá ke čtení a zápisu do libovolného PHY registru. Pokud je modul MIIM připraven (port BUSY_MIIM je nastaven do logické '0'), je příjem signálu na portu RESET_SW, READ_REG nebo WRITE_REG zahájena požadovaná operace. Dokončení této operace je indikováno nastavením portu DONE_MIIM do logické '1'. Při operaci čtení jsou data přečtená z registru držena na portu REG_DATA_READ až do zahájení další operace.

Port	Směr	Šířka	Popis
RESET_SW	vstupní	1	Zahájení SW resetu PHY
READ_REG	vstupní	1	Zahájení čtení z registru
WRITE_REG	vstupní	1	Zahájení zápisu do registru
REG_ADD	vstupní	5	Adresa reg. pro čtení/zápis
REG_DATA_WRITE	vstupní	16	Data z PHY registru
REG_DATA_READ	výstupní	16	Data do PHY registru
BUSY_MIIM	výstupní	1	Indikace činnosti modulu
DONE_MIIM	výstupní	1	Indikace dokončení operace
CLEAR_RX_ERR	vstupní	1	Smazání čítače chyb
RX_ERR_CNT_MIIM	výstupní	16	Čítač chyb čtení

Tabulka 3.6: Modul MIIM - komunikace s modulem

Porty v tabulce 3.7 složí pro komunikaci s PHY obvodem pomocí SMI rozhraní (porty MDC a MDIO) a pro indikaci stavu ethernetového spojení získané z PHY registrů obvodu LAN8720A pro vrstvu MAC (porty LINK_STATUS_MIIM, SPEED_100M_MIIM a FULL_DUPLEX_MIIM).

Port	Směr	Šířka	Popis
MDC	výstupní	1	Hodinový signál
MDIO	obousměrný	1	Data
LINK_STATUS_MIIM	výstupní	1	Indikace stavu spojení
SPEED_100M_MIIM	výstupní	1	Indikace rychlosti
FULL_DUPLEX_MIIM	výstupní	1	Indikace duplexu

Tabulka 3.7: Modul MIIM - komunikace s PHY odvodem, indikace stavu PHY

Blok miim_command

Blok miim_command slouží k příjmu požadované operace pro modul MIIM a řízení ostatních bloků modulu pro její provedení. Modul MIIM může provést následující operace:

- RESET_SW - provedení SW resetu PHY a nastavení jeho konfigurace
- READ_REG - přečtení dat z PHY registru s adresou REG_ADD
- WRITE_REG - zápis dat do PHY registru s adresou REG_ADD

Tento blok je realizován stavovým automatem, jehož první část je uvedena na obrázku 3.9. Na diagramu přechodů je zobrazena logika výběru operace, kterou bude blok MIIM provádět.

Blok používá hodinový signál s frekvencí 50 MHz na portu REF_CLK a je ho možné resetovat synchronním resetem na portu RESET.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.8: Blok miim_command - hodinové a resetovací porty

Porty uvedené v tabulce 3.9 slouží k příjmu typu operace, kterou má modul MIIM provést a jejích případných parametrů (adresa, zapisovaná data). Blok dále indikuje provádění a dokončení požadované operace a na svůj výstup předává data z přečteného registru.

Port	Směr	Šířka	Popis
RESET_SW	vstupní	1	Zahájení SW resetu PHY
READ_REG	vstupní	1	Zahájení čtení z registru
WRITE_REG	vstupní	1	Zahájení zápisu do registru
REG_ADD	vstupní	5	Adresa reg. pro čtení/zápis
REG_DATA_WRITE	vstupní	16	Zapisovaná data do registru
REG_DATA_READ	výstupní	16	Přečtená data z registru
BUSY_MIIM	výstupní	1	Indikace činnosti bloku
DONE_MIIM	výstupní	1	Indikace dokončení operace

Tabulka 3.9: Blok miim_command - komunikace s vyšší vrstvou

Porty uvedené v tabulce 3.10 slouží k indikaci bloku miim_control, kterou operaci má s využitím bloku miim_interface provést. Další porty slouží k informování bloku miim_command o činnosti bloku miim_control.

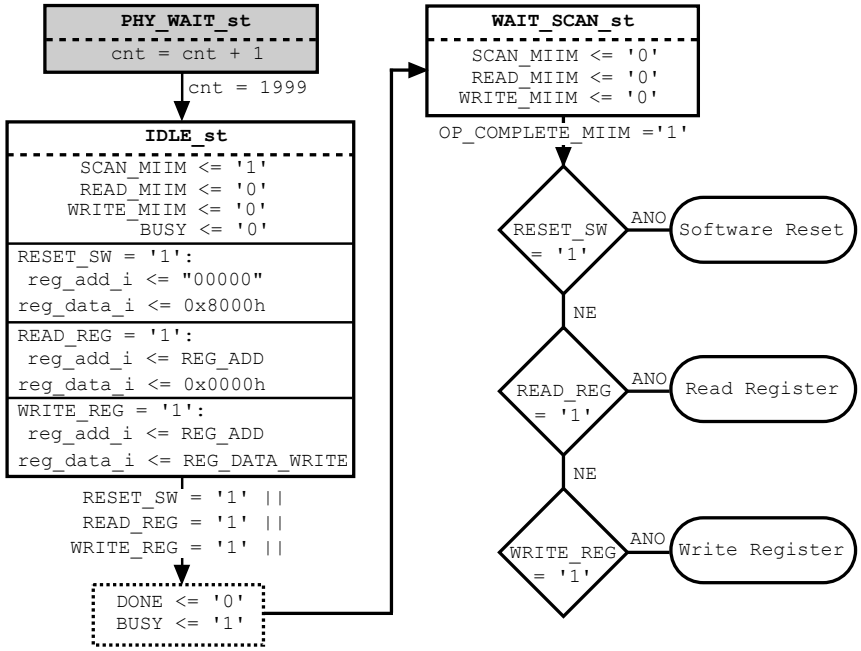
Port	Směr	Šířka	Popis
SCAN_MIIM	výstupní	1	Žádost o SW reset PHY
READ_MIIM	výstupní	1	Žádost o čtení z registru
WRITE_MIIM	výstupní	1	Žádost o zápis do registru
REG_MIIM	výstupní	5	Adresa reg. pro čtení/zápis
DATA_WRITE_MIIM	výstupní	16	Zapisovaná data
DATA_READ_MIIM	vstupní	16	Přečtená data
DATA_VALID_MIIM	vstupní	1	Indikace platnosti dat
OP_COMPLETE_MIIM	vstupní	1	Indikace dokončení operace

Tabulka 3.10: Blok miim_command - komunikace s blokem miim_control

Ve stavu PHY_WAIT_st, ve kterém se blok nachází pouze po zapnutí nebo pro resetu, čeká blok po dobu 2 μs. Toto čekání je zde z důvodu dokončení inicializace PHY obvodu. Bez tohoto stavu PHY obvod nereagoval při prvním čtení PHY registru.

Z tohoto stavu přechází stavový automat do stavu IDLE_st. V tomto stavu je nastaven port SCAN_MIIM do logické '1'. Tím je bloku miim_control indikováno, že má provádět kontinuální čtení PHY registrů s adresou 1 a 31 pro zjišťování aktuálního stavu ethernetového spojení. Vyšší vrstvě je na portu BUSY, který je nastaven do logické '0', indikováno, že modul MIIM je připraven pro přijetí další operace (čtení, zápis, SW reset). Operace je zahájena při nastavení portu RESET_SW, READ_REG nebo WRITE_REG do logické '1'. Podle požadované operace jsou nastaveny vnitřní signály reg_add_i a reg_data_i a vyšší vrstvě je indikováno na portu BUSY, že modul pracuje na požadované operaci. Stavový automat bloku miim_command přechází do stavu WAIT_SCAN_st.

V tomto stavu blok čeká na ukončení právě prováděné operace, což je indikováno blokem miim_control logickou '1' na portu OP_COMPLETE_MIIM. Při zaznamenání této hodnoty je podle požadované operace zahájeno čtení registru, zápis do registru nebo softwarový reset PHY obvodu.



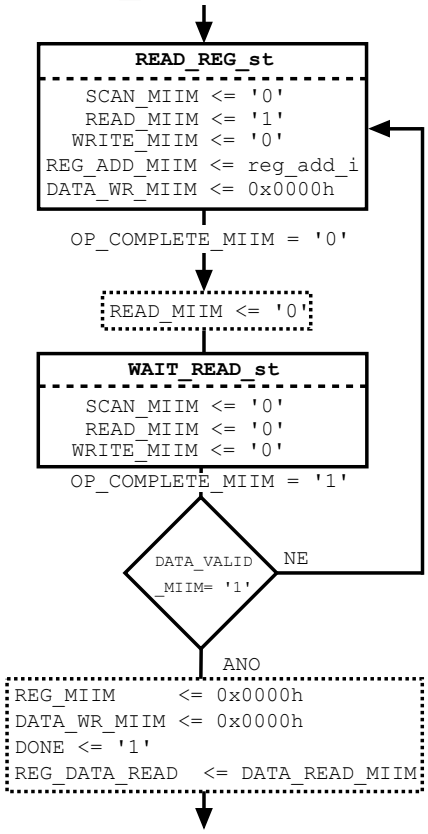
Obrázek 3.9: Blok miim_command - diagram přechodů FSM - výběr operace

Operace čtení je realizována stavy READ_REG_st a WAIT_READ_st (obrázek 3.10). V prvním stavu READ_REG_st je bloku miim_control indikován požadavek na čtení dat z PHY registru na portu READ_MIIM nastaveném do logické '1'. Zároveň je nastaven port REG_ADD_MIIM s požadovanou adresou registru uloženou přii přechodu ze stavu IDLE_st.

Stavový automat bloku poté čeká na potvrzení přijetí požadavku blokem miim_control, který je indikován nastavenou logickou '0' na portu OP_COMPLETE_MIIM. Ta způsobí přechod do stavu WAIT_READ_st. Při tomto přechodu je nastaven port READ_MIIM zpět do logické '0'.

Ve stavu WAIT_READ_st stavový automat bloku čeká na potvrzení dokončení čtení registru, které je indikováno blokem miim_control logickou '1' na portu OP_COMPLETE_MIIM. Po přijetí potvrzení dokončení čtení registru je zkontrolováno, zda bylo čtení registru provedeno bez chyby. To je indikováno logickou '1' na portu DATA_VALID_MIIM. Pokud PHY obvod správně zareagoval na požadavek o přečtení registru a logická '1' je na portu DATA_VALID_MIIM nastavena, jsou data z přečteného registru umístěna na port REG_DATA_READ a vyšší vrstvě je indikováno na portu DONE_MIIM dokončení čtení a blok se vrací do stavu IDLE_st.

Pokud došlo při čtení registru k chybě (viz popis bloku miim_interface) a na portu DATA_VALID_MIIM se nachází logická '0', přechází stavový automat bloku zpět do stavu READ_REG_st a znovu generuje žádost o přečtení registru bloku miim_control.

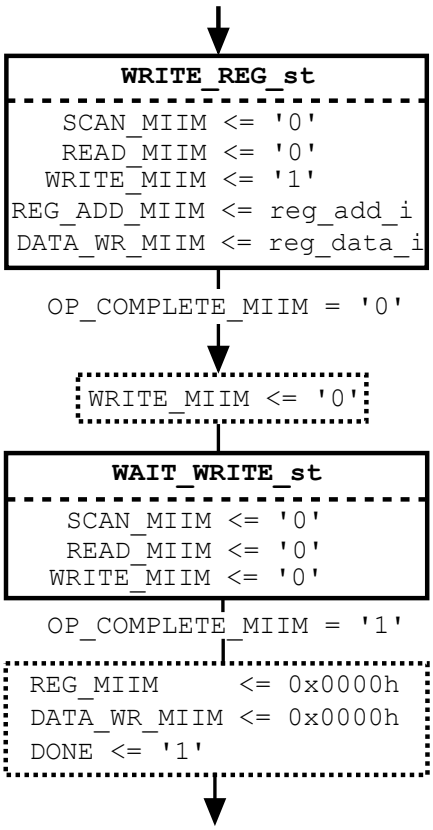


Obrázek 3.10: Blok miim_command - diagram přechodů FSM - čtení registru

Zápis dat do PHY registru je realizován pomocí stavů WRITE_REG_st a WAIT_WRITE_st (obrázek 3.11). Ve stavu WRITE_REG_st je bloku miim_control indikován požadavek na zápis dat do PHY registru na portu WRITE_MIIM nastaveném do logické '1'. Zároveň je nastavena požadovaná adresa registru na portu REG_ADD_MIIM a zapisovaná data na portu DATA_WR_MIIM.

Stavový automat bloku v tomto stavu čeká na potvrzení přijetí požadavku blokem miim_control. Ten je indikován na portu OP_COMPLETE_MIIM logickou '0'. Ta způsobí přechod do stavu WAIT_WRITE_st. Při tomto přechodu je nastaven port WRITE_MIIM zpět do logické '0'.

Zde stavový automat čeká na potvrzení dokončení zápisu do registru, který je indikován logickou '1' na portu OP_COMPLETE_MIIM. Po přijetí potvrzení dokončení zápisu jsou vynulovány hodnoty na portu adresy registru a zapisovaných dat. Dále je vyšší vrstvě je indikováno na portu DONE_MIIM dokončení zápisu a stavový automat bloku se vrací do stavu IDLE_st.



Obrázek 3.11: Blok miim_command - diagram přechodů FSM - zápis do registru

Poslední operace kterou blok miim_command může uskutečnit je softwarový reset PHY obvodu a nastavení jeho konfigurace. Pro provedení systémového resetu využívá blok stejnou logiku přechodů stavů pro čtení dat z PHY registru a zápis dat do PHY registru, jak je uvedeno na obrázku 3.10 a 3.11.

V první části softwarového resetu je do registru s adresou 0 (Basic Control Register) zapsána hodnota 0x8000h. Tím je nastaven MSB tohoto registru na hodnotu logická '1', čímž je zahájen reset PHY obvodu. Tento bit se po provedení resetu sám vynuluje. Během resetu nemají být nastavovány další bity v tomto registru. (viz [7], str. 42).

Po provedení tohoto zápisu přejde stavový automat do stavu pro čtení tohoto registru. Stavový automat bloku čeká na potvrzení dokončení čtení a potvrzení že při čtení nedošlo k chybě. Navíc je ve stavu WAIT_READ_st kontrolován MSB přečtených dat z registru 0. Čtení tohoto registru je opakováno dokud není přečtený bit '0', který značí dokončení softwarového resetu.

Po dokončení softwarového resetu PHY je zapsána konfigurace, uložená v konstantě PHY_CONFIGURATION, do stejného registru. Tuto konfiguraci je možné případně změnit v souboru miim_command.vhd, po naprogramování je neměnná. Standardní nastavení je hodnota 0x1000h, které zapne Auto-Negotiation, Loopback a izolování RMII od PHY jsou vypnuté, ostatní bity pro nastaveni rychlosti a duplexu jsou při zapnutém Auto-Negotiation ignorovány. (viz [7], str 41-43).

Posledním krokem konfigurace PHY po systémovém resetu je přečtení registru s adresou 31 (PHY Special Control/Status Register). Při čtení je zkontrolováno správné přečtení registru kontrolou logické '1' na portu DATA_VALID_MIIM. Pokud je v konfiguraci PHY_CONFIGURATION nastavený bit 12 (zapnuté Auto-Negotiation) je navíc zkontrolováno jeho dokončení. To je indikováno v 12. bitu přečteného registru. Pokud dojde k chybě nebo případné Auto-Negotiation ještě není dokončeno, tak blok zahájí další čtení tohoto registru a pokračuje se čtením až do doby, kdy se podaří přečíst registr a Auto-Negotiation je případně dokončeno.

Blok miim_control

Blok miim_control je určen k příjmu požadované operace (čtení registru, zápis do registru, zjištění stavu ethernetového spojení) z bloku miim_command a k následné obsluze bloku miim_interface, který se stará o obsluhu komunikace s PHY obvodem LAN8720A pomocí SMI rozhraní.

Blok používá hodinový signál s frekvencí 50 MHz na portu REF_CLK a je ho možné resetovat synchronním resetem na portu RESET.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.11: Blok miim_control - hodinové a resetovací porty

Porty, uvedené v tabulce 3.12, složí ke komunikaci s MAC vrstvou projektu, indikují stav ethernetového spojení získané z PHY registrů obvodu LAN8720A. Jedná se o informaci o tom zda je ethernetové spojení aktivní, jeho rychlost (100 Mbit/s nebo 10 Mbit/s) a o typ duplexu (plný nebo poloviční duplex). Blok také poskytuje vyšší vrstvě čítač chyb, ke kterým došlo čtení PHY registrů a reset tohoto čítače.

Port	Směr	Šířka	Popis
LINK_STATUS	výstupní	1	Indikace navázaného spojení
SPEED_100M	výstupní	1	Indikace rychlosti rozhraní
FULL_DUPLEX	výstupní	1	Plně duplexní přenos
CLEAR_RX_ERR	vstupní	1	Reset čítače chyb čtení
RX_ERR_COUNT	výstupní	16	Čítač chyb čtení

Tabulka 3.12: Blok miim_control - komunikace s vyšší vrstvou

Porty uvedené v tabulce 3.13 slouží k příjmu typu operace a jejích parametrů (adresa registru, zapisovaná data) z bloku miim_command. Dále jsou zde výstupní porty pro předání přečtených dat z registru, jejich platnosti a indikace dokončení požadované operace blokem miim_control.

Pokyn pro zahájení operace blok miim_control zpracovává a podle typu operace řídí blok miim_interface pomocí portů uvedených v tabulce 3.14. V této tabulce jsou také uvedeny vstupní porty, které bloku miim_control indikují stav bloku miim_interface v průběhu a po dokončení provádění obsluhy SMI rozhraní.

Port	**Směr**	**Šířka**	**Popis**
SCAN_OP	vstupní	1	Zahájení operace SCAN
READ_OP	vstupní	1	Zahájení operace READ
WRITE_OP	vstupní	1	Zahájení operace WRITE
REG_ADD_IN	vstupní	5	Adresa registru
DATA_WRITE_IN	vstupní	16	Data do PHY registru
DATA_READ_OUT	výstupní	16	Data z PHY registru
DATA_VALID	výstupní	1	Indikace platnosti dat
OP_COMPLETE	výstupní	1	Indikace dokončení operace

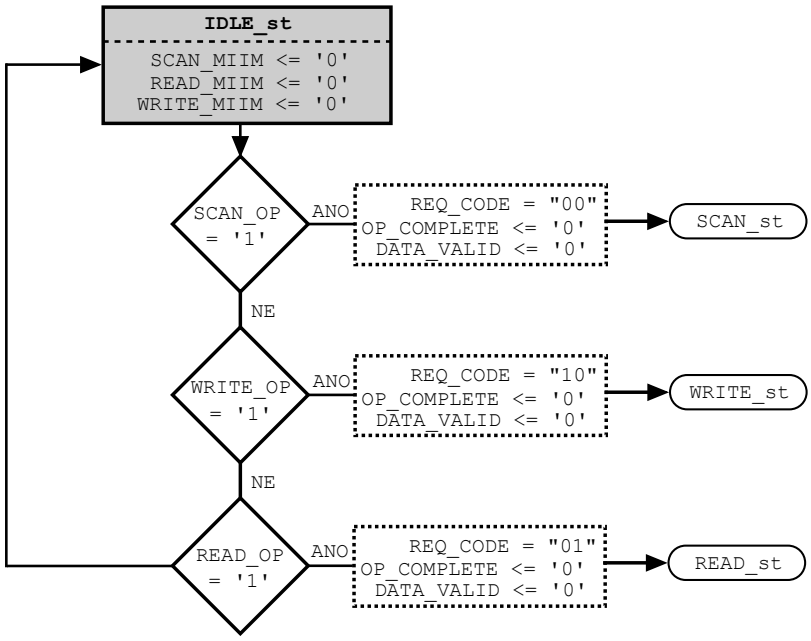
Tabulka 3.13: Blok miim_control - komunikace s blokem miim_command

Porty	**Směr**	**Šířka**	**Popis**
START	výstupní	1	Zahájeni operace
OPCODE	výstupní	2	Kód operace
REG_ADD_OUT	výstupní	5	Adresa registru
DATA_WRITE_OUT	výstupní	16	Zapisovaná data
DATA_READ_IN	vstupní	16	Přečtená data
READ_ERR	vstupní	1	Indikace chyby čtení
BUSY	vstupní	1	Indikace činnosti bloku

Tabulka 3.14: Blok miim_control - komunikace s blokem miim_interface

První část diagramu přechodů stavového automatu bloku miim_control je uvedena na obrázku 3.12. Po zapnutí nebo resetu se stavový automat bloku nachází ve stavu IDLE_st, ve kterém čeká na spuštění operace SCAN, READ nebo WRITE, jejíž zahájení je indikováno nastavením logické '1' na některém z portů SCAN_OP, READ_OP nebo WRITE_OP blokem miim_command.

Pokud dojde k nastavení některého z těchto portů je do vnitřního signálu REQ_CODE uložena hodnota přiřazená operaci, která se bude provádět, je zrušena platnost dat na portu DATA_READ_OUT nastavením portu DATA_VALID do logické '0' a bloku miim_command je indikováno na portu OP_COMPLETE, že blok pracuje na požadované operaci.



Obrázek 3.12: Bloku miim_control - diagram přechodů FSM - výběr operace

Blok poté přejde do stavu SCAN_st, READ_st nebo WRITE_st, podle toho jaká operace byla při přechodu ze stavu IDLE_st odeslána blokem miim_command. V těchto stavech je nastaven port OPCODE na hodnotu "10" pro čtení (SCAN nebo READ) nebo "01" pro zápis (WRITE). Ve stavech READ_st a WRITE_st je nastaven port REG_ADD_OUT podle hodnoty na portu REG_ADD_IN, která určuje adresu PHY registru pro čtení nebo zápis. Ve stavu WRITE_st jsou navíc na port DATA_WRITE_OUT nastavena zapisovaná data, které blok miim_command nastavil na portu DATA_WRITE_IN.

Ve stavu SCAN_st je adresa registru REG_ADD_OUT nastavena na hodnotu 0x01h při lichém čtení registru a na hodnotu 0x1Fh při sudém čtení registru. Pokud blok miim_command nezpracovává žádnou operaci z vyšší vrstvy je v klidovém stavu kontinuálně nastaven port SCAN_st a blok miim_control (s využitím bloku miim_interface) čtením těchto dvou registrů kontroluje stav ethernetového spojení.

Z výše zmíněných stavů přechází stavový automat bloku do stavu START_st, kde čeká na potvrzení od bloku miim_interface. Ten v klidovém stavu drží port BUSY v logické '0'. Tento stav způsobí přechod FSM do stavu ACK_st spolu s nastavením portu START do logické '1'. Tím je spuštěn blok miim_interface, který z nastaveného kódu operace (OPCODE), adresy registru (REG_ADD) a případných zapisovaných dat (DATA_WRITE) spustí požadovanou obsluhu rozhraní SMI.

Po spuštění obsluhy rozhraní blok miim_interface nastaví port BUSY do logické '1', čímž způsobí přechod FSM bloku miim_control do stavu BUSY_st. Při tomto přechodu je zrušeno nastavení portu START_st, aby nebyla operace po ukončení vykonána vícekrát. Ve stavu BUSY_st vyčkává stavový automat na indikaci ukončení činnosti bloku miim_interface, znovu indikované logickou '0' na portu BUSY. Při detekování tohoto stavu přechází FSM bloku do posledního stavu DONE_st.

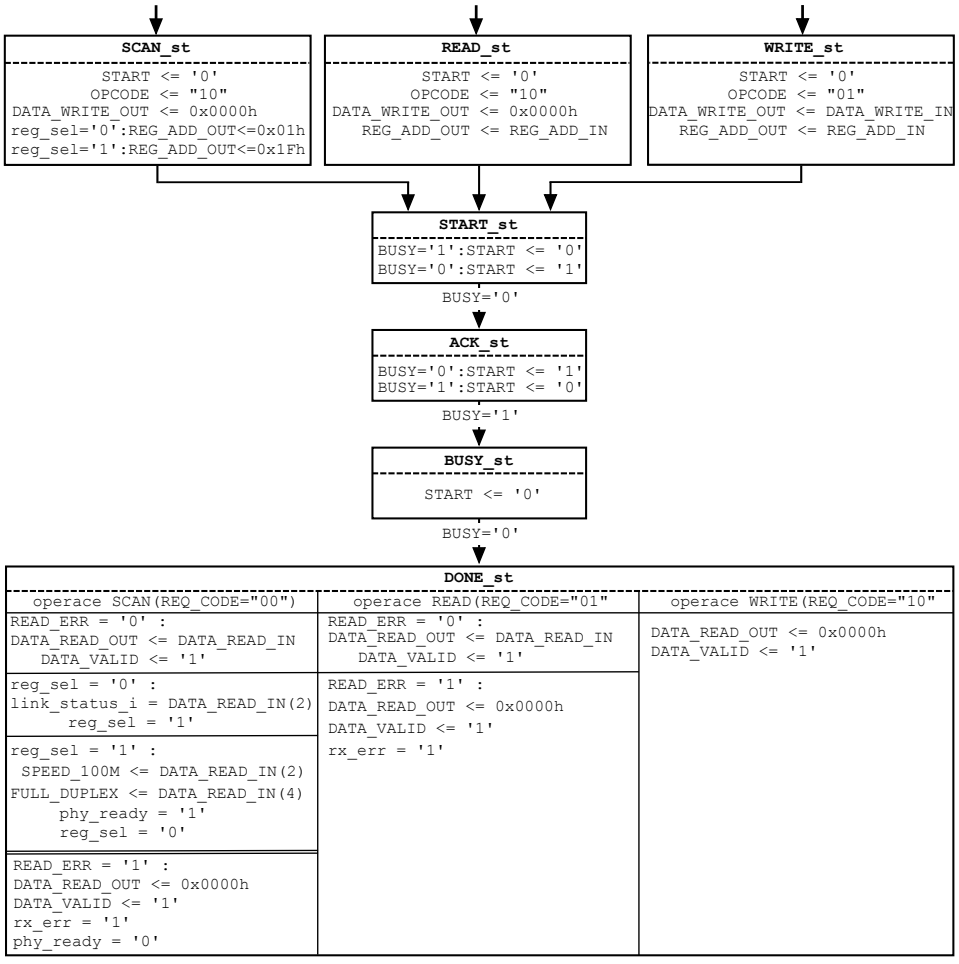
V tomto stavu činnost bloku závisí na prováděné operaci. V případě zápisu dat do PHY registru (WRITE) blok neposkytuje žádná data a pouze bloku miim_command návratem do stavu IDLE_st indikuje nastavením portu OP_COMPLETE do logické '1' dokončení zápisu dat. Pokud bylo prováděno čtení registru (READ), je zkontrolováno, zda nedošlo při čtení k chybě (port READ_ERR). V případě chyby je signálem rx_error inkrementován čítač chyb (obrázek 3.13), není indikována platnost dat (DATA_VALID) na výstupu DATA_READ_OUT. Pokud bylo čtení úspěšné jsou přečtená data z bloku miim_interface zkopírována na port DATA_READ_OUT a portem DATA_VALID je indikována jejich platnost.

```
1  RX_ERR_COUNT <= std_logic_vector(to_unsigned(rx_error_cnt,
       RX_ERR_COUNT'length));
2
3  process(REF_CLK)
4
5  begin
6    if rising_edge(REF_CLK) then
7      if RESET = '1' then
8        rx_error_cnt <= 0;
9      else
10       if CLEAR_RX_ERR = '1' then
11         rx_error_cnt <= 0;
12       else
13         if rx_error = '1' then
14           rx_error_cnt <= rx_error_cnt + 1;
15         end if;
16       end if;
17     end if;
18   end if;
19 end process;
```

Obrázek 3.13: Blok miim_control - čítač chyb čtení PHY registru

Při provádění operace SCAN je použita stejná logika testování chyby při čtení jako u operace READ. Navíc je zde použit vnitřní signál reg_sel, který slouží k přepínání hodnoty adresy čteného registru. Při čtení registru s adresou 0x01h je do vnitřního signálu link_status_i uložen třetí bit přečtených dat, ve kterém je indikováno aktivní ethernetové spojení. V následujícím čtení registru s adresou 0x1Fh je na porty SPEED_100M a FULL_DUPLEX nastavena hodnota třetího a pátého bitu, které indikují rychlost respektive typ duplexu ethernetového spojení. Port LINK_STATUS je maskován pomocí vnitřního signálu phy_ready, který je nastaven až po přečtení druhého registru, tak aby byla MAC vrstvě poskytována kompletní informace o stavu spojení. V případě že dojde při provádění operace SCAN k chybě při čtení, je signál phy_ready vynulován a MAC vrstvě je indikována ztráta ethernetového spojení. Stejně jako u operace READ je při této chybě nastaven signál rx_error inkrementující čítač chyb čtení.

Po provedení vybraných úkonů ve stavu DONE_st se FSM bloku vrací zpět do stavu ILDE_st.



Obrázek 3.14: Bloku miim_control - diagram přechodů FSM - provedení operace

Blok miim_clock_gen

Blok miim_clock_gen zajišťuje vytvoření hodinového signálu SMI_CLK pro blok miim_interface. Ze vstupního hodinového signálu REF_CLK s frekvencí 50 MHz je dělením pomocí čítače vytvořen signál s periodou 200 ns. Odvozený generovaný hodinový signál na portu MDC bloku miim_interface má dvojnásobnou periodu 400 ns.

Podle datasheetu použitého obvodu je minimální perioda signálu MDC 400 ns. Pro správné časování signálů na portu MDIO je potřeba vytvořit signál s poloviční periodou, aby byla zajištěna doba předstihu a přesahu signálu MDIO vůči MDC pro zápis do PHY. Zároveň je nutné dodržet vzorkování signálu z PHY v době, kdy jsou po vzestupné hraně MDC data platná.

Vstupní a výstupní porty bloku jsou uvedeny v následujících tabulce:

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset bloku
SMI_CLK	výstupní	1	Hodinový signál pro blok miim_interface

Tabulka 3.15: Blok miim_clock_gen - porty

Funkce bloku je realizována jednoduchým VHDL procesem, který je uveden na následujícím obrázku. Jedná se pouze o synchronní čítač, který každých 5 hodinových taktů REF_CLK neguje signál SMI_CLK_i přivedený na výstupní port SMI_CLK. Výsledkem je signál na portu SMI_CLK s periodou 200 ns.

```
1  SMI_CLK <= SMI_CLK_i;
2
3  process(REF_CLK, RESET)
4  begin
5    if rising_edge(REF_CLK) then
6      if RESET = '1' then
7        cnt <= 0;
8        SMI_CLK_i <= '0';
9      else
10       if cnt = 4 then
11         SMI_CLK_i <= not SMI_CLK_i;
12         cnt <= 0;
13       else
14         cnt <= cnt + 1;
15       end if;
16     end if;
17   end if;
18 end process;
```

Obrázek 3.15: Blok miim_clock_gen - generování SMI_CLK

Blok miim_interface

Blok miim_interface zajišťuje obsluhu SMI rozhraní, která složí ke komunikaci s PHY obvodem. Tato sběrnice je částí MII rozhraní a je určena ke čtení a zápisu dat PHY registrů určených ke konfiguraci a kontrole stavu PHY obvodu. (víz kapitola 2.4.3). Blok je ovládán signály z bloku miim_control.

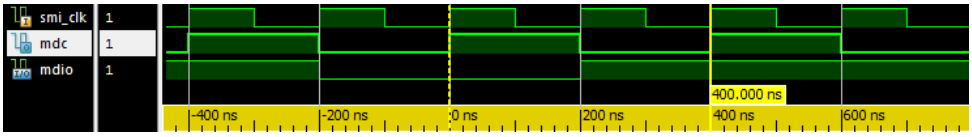
Blok používá hodinový signál s periodou 200 *ns* na portu SMI_CLK, který je generován blokem miim_clock_gen. Z tohoto signálu je vytvořen signál MDC, vyniklý dělením tohoto hodinového signálu dvěma. Sestupná hrana signálu MDC taktuje stavový automat bloku a určuje okamžik, ve kterém dochází ke změnám na portu MDIO při komunikaci směrem k PHY. Tím je zaručena potřebná doba předstihu a přesahu signálu MDIO vůči MDC, která je podle datasheetu 10 *n*s. Stejnou sestupnou hranou signálu MDC je vzorkováno MDIO ve směru od PHY. Minimální doba po kterou jsou data z PHY platná po vzestupné hraně MDC je podle datasheetu PHY obvodu 300 *ns*. Vzorkování je prováděno 200 *ns* pro vzestupné hraně MDC a tato doba vyhovuje požadovaným časovým parametrům obvodu.

Vzájemný vztah těchto signálů je na obrázku 3.16, kde je na simulaci zobrazena část zápisu dat do PHY obvodu.

Oproti ostatním dříve zmíněným blokům je reset bloku miim_interface asynchronní, protože při resetu již není generován signál SMI_CLK a blok by při použití synchronního resetu nebyl správně resetován.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Asynchronní reset bloku

Tabulka 3.16: Blok miim_interface - hodinové a resetovací porty



Obrázek 3.16: Blok miim_interface - simulace vztahu SMI_CLK, MDC a MDIO

Porty uvedené v tabulce 3.17 slouží ke komunikaci s blokem miim_control, tedy k přijmutí požadavku o zahájení obsluhy rozhraní (START), příjmu kódu operace (OPCODE), kterou má blok miim_interface provést a jejích parametrů (adresa - REG_ADD, zapisovaná data - DATA_WRITE). Blok dále indikuje provádění požadované operace na portu BUSY a na svůj výstup předává data ze čteného registru (DATA_READ) nebo informaci, že PHY na čtení nereaguje (READ_ERR).

Port	Směr	Šířka	Popis
START	vstupní	1	Zahájení čtení/zápisu
OPCODE	vstupní	2	Kód operace
REG_ADD	vstupní	5	Adresa reg. pro čtení/zápis
DATA_WRITE	vstupní	16	Zapisovaná data do registru
DATA_READ	výstupní	16	Přečtená data z registru
READ_ERR	výstupní	1	Indikace chyby čtení
BUSY	výstupní	1	Indikace probíhající operace

Tabulka 3.17: Blok miim_interface - komunikace s blokem miim_control

Ke komunikaci s PHY obvodem slouží porty rozhraní SMI uvedené v tabulce 3.18. Tyto porty jsou vyvedené na piny (Al, Ca) FPGA obvodu Artix-7 na přípravku Digilent Nexys 4.

Port	Směr	Šířka	Popis
MDC	vstupní	1	SMI - Hodinový signál
MDIO	obousměrný	1	SMI - Data

Tabulka 3.18: Blok miim_interface - komunikace s PHY obvodem

Činnost bloku je řízena stavovým automatem, jehož diagram přechodů je uveden na obrázku 3.18. Počátečním stavem bloku je stav IDLE_st. V tomto stavu je datová sběrnice v klidovém stavu vysoké impedance a blok indikuje na portu BUSY logickou '0', že je připraven na obsluhu SPI rozhraní. Ta je zahájena nastavením logické '1' na portu START, která způsobí přechod stavového automatu do stavu PREAMBLE_st. Během tohoto přechodu je nastavena logická '1' na portu MDIO a blok indikuje zahájení obsluhy rozhraní logickou '1' na portu BUSY. Dále je při tomto přechodu nastavena hodnota čítače bit_counter na hodnotu 31. Tento čítač je ve všech dalších stavech využit k časování jednotlivých částí obsluhy SMI rozhraní

Ve stavu PREAMBLE_st je odesláno 32 logických '1' na portu MDIO, které spolu s hodinovým signálem MDC zajišťují synchronizaci PHY. Po odeslání preambule je ve stavu OPCODE_st odeslána značka ST a kód operace OP (vnitřní signál start_opcode). Značka ST je tvořena bitovou posloupností <01> a kód operace je posloupnost <10> pro čtení nebo <01> pro zápis. Poté je ve stavu PHY_ADD_st odeslána fyzická adresa PHY obvodu. Tato adresa je napevno nastavena na hodnotu "00001", která je dána fyzickým zapojením PHY obvodu na PCB přípravku Digilent Nexys 4. Po odeslání adresy následuje ve stavu REG_ADD_st odeslání adresy PHY registru, ke kterému je v dané transakci přistupováno.

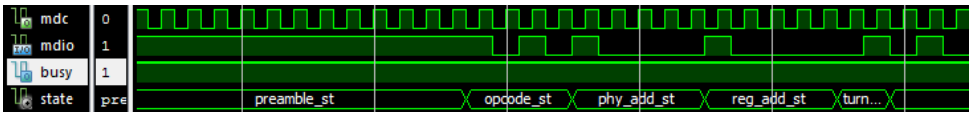
Stav do kterého stavový automat bloku přejde po odeslání adresy registru závisí na typu právě prováděné transakce. Pokud se jedná o zápis do PHY registru, přechází stavový automat do stavu TURNAROUND_WR_st. V případě že je požadováno čtení PHY registru přechází stavový automat do stavu TURNAROUND_RD_st.

Ve stavu TURNAROUND_WR_st je odeslána bitová posloupnost <10> a FSM poté přechází do stavu WRITE_st, kde jsou postupně od MSB odeslány jednotlivé bity zapisovaných dat, které jsou na portu DATA_WRITE na MDIO sběrnici. Po odeslání je bloku miim_control indikováno dokončení zápisu do registru nastavením portu BUSY do logické '0'.

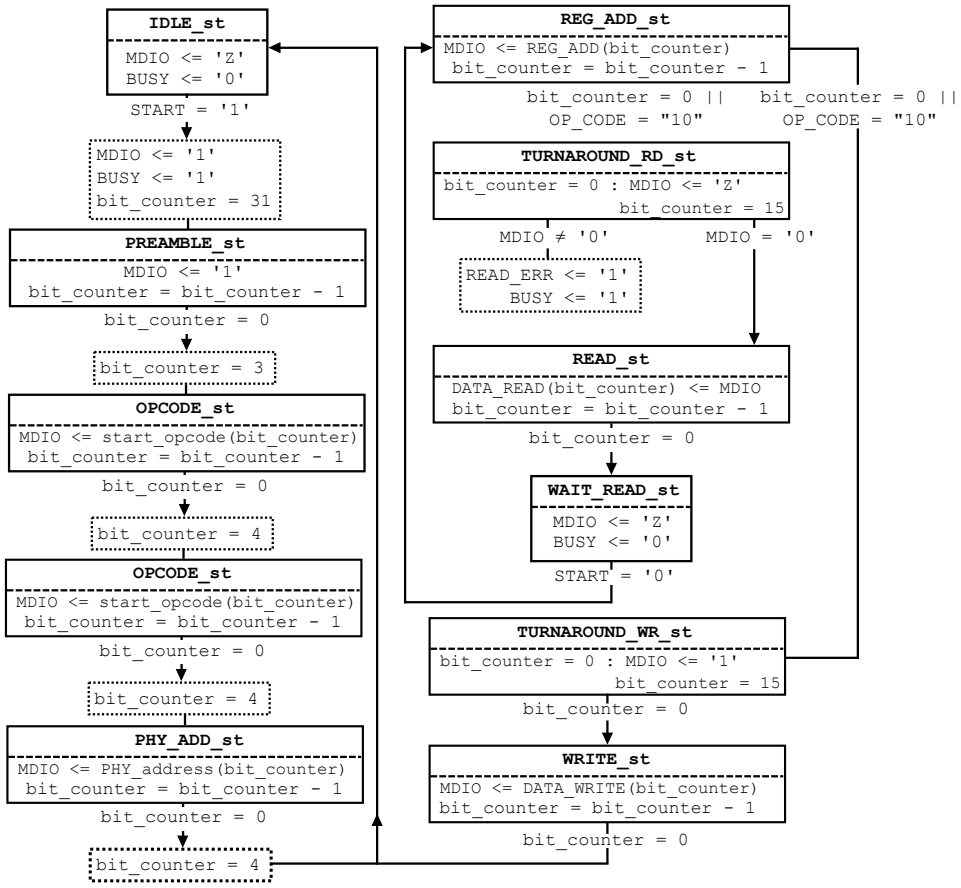
Ve stavu TURNAROUND_RD_st je v prvním taktu nastavena sběrnice MDIO do stavu vysoké impedance ('Z'). V dalším taktu by PHY obvod měl nastavit MDIO do logické '0'. Toto nastavení slouží k ověření, že sběrnice a PHY jsou schopné přenést data. V případě že PHY nenastaví sběrnici do logické '0', a kvůli pull-up odporu je sběrnice držena ve stavu logické '1', je detekována chyba čtení. Bloku miim_control je na portu READ_ERR indikována chyba a ukončení operace čtení nastavením portu BUSY do logické '0'. Pokud PHY správně odpoví přejde stavový automat bloku ze stavu TURNAROUND_RD_st do stavu READ_st.

V tomto stavu jsou postupně od MSB ukládány hodnoty jednotlivých přečtených bitů z MDIO na výstupní port DATA_READ. Po uložení všech 16-ti bitů dojde k přechodu do stavu WAIT_READ_st. V tomto stavu je indikováno dokončení operace na portu BUSY nastaveném do logické '0' a sběrnice MDIO je nastavena zpět do klidového stavu vysoké impedance. Stavový automat bloku zde čeká na potvrzení příjmu přečtených dat blokem miim_control nastavením portu START do logické '0'.

Na následujícím obrázku je ukázka z simulace činnosti bloku miim_interface při zápisu do registru při softwarovém resetu PHY. Je zde zobrazena část preambule a zápis pouze MSB zapisovaných dat (tedy bit pro reset PHY, nastavený do logické '1'). Na obrázku si lze povšimnout změny dat na MDIO na sestupnou hranu MDC, začátku rámce a kódu operace pro zápis (ST&OP), adresy PHY (PHY_ADD = 0x01), a adresy registru do, kterého je touto transakcí zapisováno. Dále je zde zobrazena sekvence <10> pro turnaround při zápisu a zmíněný MSB zapisovaných dat. Signál state obsahuje stav FSM bloku miim_interface a řídí odesílanou část rámce. Data na MDIO jsou oproti stavu FSM zpožděna o jeden takt.



Obrázek 3.17: Blok miim_interface - zápis do PHY registru na SMI rozhraní



Obrázek 3.18: Bloku miim_interface - diagram přechodů FSM

3.2.3 Modul MAC

Modul MAC realizuje funkci podvrstvy MAC a slouží k sestavení a odesílání MAC rámců z dat přijatých z vyšší vrstvy. Modul dále zpracovává přijaté rámce z PHY obvodu a přijatá data z těchto rámců poskytuje vyšší vrstvě. Blokové schéma modulu je uvedeno v příloze A. Tento modul se skládá z následujících bloků:

- mac_tx,
- mac_tx_crc,
- mac_tx_random,
- mac_reset_phy,
- mac_rx_crsdv,
- mac_rx,
- mac_rx_crc,

Blok mac_reset_phy generuje hardwarový reset PHY obvodu a nastavuje jeho piny potřebné pro nastavení jeho konfigurace. Blok mac_rx_crsdv odděluje signály CRS a RX_DV ze signálu RMII rozhraní CRS_DV (viz 2.5.1). Blok mac_tx slouží k odesílání MAC rámců, k čemuž využívá bloků mac_tx_crc (výpočet pole FCS odesílaného rámce)a blok mac_tx_random (backoff algoritmus pro řešení kolizí). Blok mac_rx slouží k příjmů MAC rámců spolu s blokem mac_rx_crc, který ověřuje FCS v přijatém rámci. Tento modul implementuje MAC vrstvu a část RMII propojení s PHY obvodem (MIIM část RMII rozhraní je implementována modulem MIIM). Detailní popis portů a činnosti těchto bloků je uveden v samostatných částech této kapitoly.

Pro správnou funkci modulu MAC je potřeba použít hodinový signál s frekvencí 50 MHz přivedený na port REF_CLK. Tento hodinový signál je v projektu generován modulem MMCM (viz kapitola 3.2.1)

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál modulu MAC
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.19: Modul MAC - hodinové a resetovací porty

Pro komunikaci modulu MAC s vyšší vrstvou při odesílání dat jsou určeny porty v tabulce 3.20. Z přijatých oktetů na portu TX_DATA je vytvořen rámec, který je odeslán pomocí RMII rozhraní do PHY. Jednotlivé oktety jsou po zpracování modulem MAC potvrzovány na portu TX_ACK. Úspěšné odeslání dat je vyšší vrstvě indikováno portem TX_COMPLETE. Pokud při odesílání rámce dojde ke kolizi je odesílání přerušeno a vyšší vrstva je o tomto stavu informována portem TX_RETRY. Rámec by měl být vyšší vrstvou znovu odeslán do modulu MAC, který se pokusí o opětovné odesílání. Pokud dojde k vyčerpání počtu pokusů ma opakované odeslání rámce je o tomto stavu informována vyšší vrstva na portu TX_ERROR.

Port	Směr	Šířka	Popis
TX_DATA	vstupní	8	Odesílaná data
TX_DATA_VALID	vstupní	1	Platnost dat na TX_DATA
TX_ACK	výstupní	1	Potvrzení příjmu TX_DATA
TX_COMPLETE	výstupní	1	Indikace dokončení odesílání
TX_ERROR	výstupní	1	Indikace chyby při odesílání
TX_RETRY	výstupní	1	Žádost o znovu odeslání rámce

Tabulka 3.20: Modul MAC - odesílání rámců

Pro komunikaci modulu MAC s vyšší vrstvou při příjmu dat z PHY jsou určeny porty v tabulce 3.21. Jednotlivé oktety přijímaného rámce jsou modulem odesílány portem RX_DATA spolu s indikací na portu RX_DATA_VALID potvrzujícím platnost dat na portu RXDATA. Pokud dojde při příjmu rámce k chybě (chybné FCS, chybná délka rámce) je o tom vyšší vrstva informována na portechRX_FCS_ER respektive RX_LENGTH_ERR.

Port	Směr	Šířka	Popis
RX_DATA	vstupní	8	Přijímaná data
RX_DATA_VALID	vstupní	1	Platnost dat na RX_DATA
RX_LENGTH_ERR	výstupní	1	Indikace chyby délky rámce
RX_FCS_ERR	výstupní	1	Indikace chybného FCS

Tabulka 3.21: Modul MAC - příjem rámců

Ke komunikaci s PHY obvodem LAN8720A jsou určeny porty uvedené v tabulce 3.22. Jedná se o porty rozhraní RMI a hardwarový reset PHY obvodu. Datová komunikace při příjmu dat probíhá směrem z PHY do MAC, porty jsou obousměrné z důvodu nastavení konfigurace obvodu při resetování obvodu (viz blok mac_reset_phy v kapitole 3.2.3).

Port	Směr	Šířka	Popis
RMII_TXD	výstupní	2	RMII - odesílaná data
RMII_TX_EN	výstupní	1	RMII - indikace odesílání
RMII_RXD	obousměnrný	2	RMII - přijímaná data
RMII_CRS_DV	obousměnrný	1	RMII - signál CRS a DV
RMII_RX_ER	obousměnrný	1	RMII - chyba příjmu
PHY_RESET	výstupní	1	Reset PHY obvodu
PHY_INT	obousměnrný	1	Přerušení z PHY obvodu

Tabulka 3.22: Modul MAC - komunikace s PHY

Blok mac_reset_phy

Blok mac_reset_phy slouží k nastavení pinů PHY obvodu, použitých pro konfiguraci, do stavu vysoké impedance během HW resetu obvodu. Při resetu je podle konfiguračních pull-up rezistorů připojených k PHY obvodu nastavena konfigurace PHY adresy pro MIIM a mód komunikace. Pokud blok není resetován portem RESET, propouští signály RMII rozhraní RXD(1:0) a CRS_DV do dalších bloků modulu MAC.

PHY obvod LAN8720A má některé piny určené na nastavení obvodu při ukončení hardwarového resetu (tzv. configuration straps, viz [7], kap 3.7). Tato konfigurace je uložena při resetu po zapnutí obvodu a při ukonření hardwarového resetu na pinu nRST. Piny určené pro tuto konfiguraci mají vnitřní rezistor pro zabránění nedefinovaného stavu, pokud nejsou připojeny. Nastavení obvodu pomocí těchto pinů by mělo být provedeno externím pull-up nebo pull-down rezistorem, pomocí kterého je nastaveno napětí na těchto pinech během resetu obvodu. Reset by měl být nasteven po minimální dobu 150 μs a po resetu by měla být hodnota na těchto konfiguračních pinech držena ještě minimálně 1 ns ([7], str. 59). Na PCB přípravku jsou na těchto pinech umístěny externí pull-up rezistory, které nastavují obvod do požadované konfigurace, proto není nutné toto nastavení měnit a piny FPGA spojené s těmito konfiguračními piny jsou během resetu nastavené do stavu vysoké impedance.

Vstupní a výstupní porty bloku jsou uvedeny v následující tabulce:

Port	Směr	Šířka	Popis
RESET	vstupní	1	Globální reset
RMII_RXD	obousměrný	2	RMII - přijímaná data
RMII_CRS_DV	obousměrný	1	RMII - signál CRS a DV
RMII_RXD_i	výstupní	2	RMII - RXD do MAC
RMII_CRS_DV_i	výstupní	1	RMII - CRS_DV do MAC
RMII_RX_ER	obousměrný	1	RMII - chyba příjmu
PHY_RESET	výstupní	1	Reset PHY obvodu

Tabulka 3.23: Blok mac_reset_phy - porty

Blok mac_rx_crsdv

Blok mac_rx_crsdv slouží k rozdělení signálu CRS_DV, který je součástí rozhraní RMII, na signály CRS (carrier sense - detekce nosné) a DV (data valid - platná data). PHY nastavuje signál CRS_DV do logické '1' v době, kdy médium není v klidovém stavu. CRS_DV je nastaven asynchronně a ztráta nosné způsobí zrušení signálu CRS_DV synchronně s REF_CLK. Tento blok je realizován stavovým automatem, jehož diagram přechodů je uveden na obrázku 3.19.

Vlivem rozdílného taktu REF_CLK a taktu generovaného v PHY během příjmu rámce může po dokončení příjmu rámce zůstat ve vyrovnávací paměti v PHY několik dibitů přijatého rámce (max. 10 dibitů). V tomto případě je po nastavení CRS_DV do logické '0' při ukončení nosné, periodicky nastavován signál CRS_DV do logické '1' v době, kdy PHY vysílá na RXD(1:0) druhý dibit každého nibblu a do logické '0' v době, kdy je odesílán první dibit každého nibblu. Toto odesílání dibitů z vyrovnávací paměti je prováděno, až do doby kdy je vyrovnávací paměť prázdná. Signál CRS_DV se tedy po ukončení nosné může periodicky měnit s frekvencí 25 MHz nebo 2,5 MHz (podle rychlosti rozhraní 100 Mbit/s nebo 10 Mbit/s).

Z tohoto důvodu je pro správné generování signálu pro detekci vzniku kolize rozdělit signál CRS_DV na signály CRS a DV. Signál CRS indikuje stav, kdy médium není v klidu. Signál DV indikuje, že PHY odesílá na RXD(1:0) dekódované dibity přijímaného rámce.

Pro správnou činnost bloku je, kvůli rozdílné frekvenci změn signálu CRS_DV během vyprazdňování vyrovnávací paměti, nutné použít hodinový signál REF_CLK s frekvencí 50 MHz. V případě použití rychlosti rozhraní 10 Mbit/s (indikovaná na portu SPEED_100M) dochází k přechodům mezi relevantními stavy FSM pouze každý desátý takt REF_CLK.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.24: Blok mac_rx_crsdv - hodinové a resetovací porty

Port	Směr	Šířka	Popis
SPEED_100M	vstupní	1	Indikace rychlosti rozhraní z MIIM
RMII_RXD_i	vstupní	2	RMII - přijatá data z PHY
RMII_RXD	výstupní	2	RMII - přijatá data do MAC
RMII_CRS_DV	výstupní	1	RMII - signál CRS_DV
RMII_CRS	výstupní	2	RMII - oddělené CRS
RMII_DV	výstupní	1	RMII - oddělené DV

Tabulka 3.25: Blok mac_rx_crsdv - porty pro oddělení CRS a DV

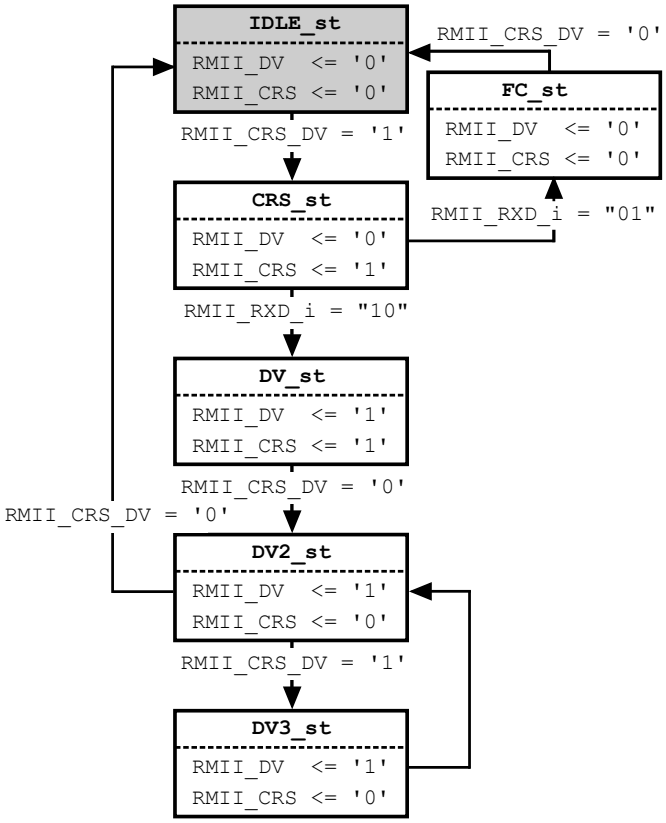
Počátečním stavem FSM bloku je stav IDLE_st, ve kterém blok čeká na detekci nosné indikované nastavením portu RMII rozhraní CRS_DV do logické '1'. Při detekci tohoto stavu přechází stavový automat do stavu CRS_st, kde blok čeká na dekódovaná data z PHY.

V případě že dojde k detekci platných dat, tedy preambule, nastaví PHY signál RXD(1:0) do stavu "10". Tento stav způsobí přechod do stavu DV_st, ve kterém je do dalších bloků MAC indikována platnost dat na RXD(1:0) signálem DV. V případě že dojde k detekci falešné nosné (false carrier, viz 2.5.1) nastaví PHY na RXD(1:0) stav "01", který způsobí přechod do stavu FC_st, ve kterém blok setrvá až do ukončení nastavení CRS_DV. FSM se poté vrací zpět do stavu IDLE_st.

V případě korektního příjmu rámce se po detekci prvního dibitu preambule dostane FSM bloku do stavu DV_st. V tomto stavu je FSM do doby, kdy je ukončen příjem rámce na médiu a je zrušen signál CRS_DV. Tím dojde k přechodu do stavu CRS_DV, kde blok čeká na další takt REF_CLK (případně deset taktů při rychlosti rozhraní 10 Mbit/s) a testuje zda nebyl CRS_DV znovu nastaven do logické '1'. Pokud zůstane CRS_DV v logické '0' (ve vyrovnávací paměti nejsou další dibity) je ukončen příjem rámce a stavový automat se vrací do stavu IDLE_st.

Pokud je ve stavu DV2_st detekováno opětovné nastavení CRS_DV, přechází stavový automat bloku do stavu DV3_st, ve kterém setrvá jeden nebo deset taktů REF_CLK. Poté se FSM vrací do stavu DV2_st, kde je znovu testována aktivita na CRS_DV. Toto testování trvá do doby, kdy již PHY nemá v bufferu další dibity a CRS_DV zůstane v logické '0' kontinuálně po 2 nebo 20 taktů (dle rychlosti rozhraní).

Kvůli čekání na testování opětovného nastavení signálu CRS_DV, v případě odesílání dibitů z vyrovnávací paměti, je v bloku mac_rx_crsdv signál RXD(1:0) (na portu RMII_RXD(1:0)_i) zpožděn, tak aby si signál DV generovaný tímto blokem a tento zpožděný signál RXD(1:0) (na portu RMII_RXD(1:0)) vzájemně časově odpovídaly.

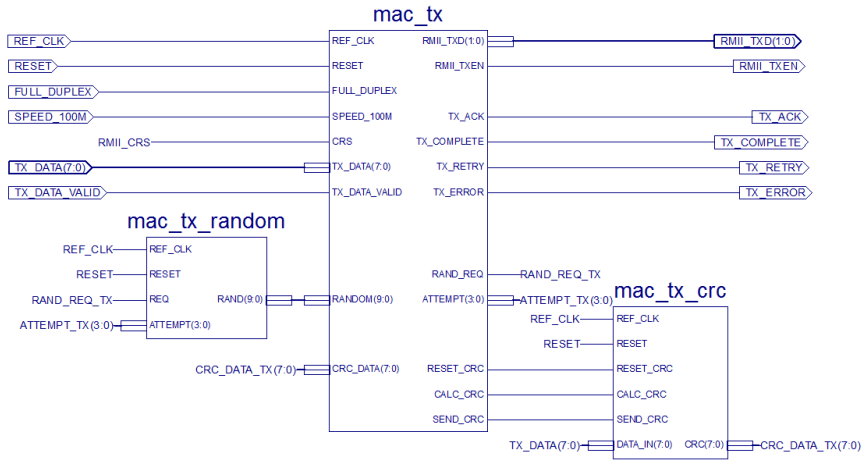


Obrázek 3.19: Modul mac_rx_crsdv - diagram přechodů FSM

Blok mac_tx

Tento blok realizuje vytvoření MAC rámce (respektive Ethernet paketu) a jeho odeslání RMII rozhraním do PHY. Blok přijímá data z vyšší vrstvy (jednotlivé datové oktety odesílaného rámce), připojuje k nim FCS vypočítaný pomocí CRC (výpočet je prováděn blokem mac_tx_crc), výplň pro dodržení minimální velikosti rámce, preambuli a SFD. Blok dále realizuje CSMA/CD algoritmus pro řešení přístupu k médiu v případě použití poloviční duplexu (pro získání pseudonáhodné doby čekání pro backoff CSMA/CD algoritmu slouží blok mac_tx_random).

Blokové schéma těchto bloků je uvedeno na obrázku 3.20.



Obrázek 3.20: Modul MAC - odesílací část

Pro správnou činnost bloku mac_tx je nutné použít hodinový signál REF_CLK s frekvencí 50 MHz kvůli ovládání RMII rozhraní. Reset bloku je synchronní s REF_CLK a je v projektu generován modulem MMCM.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.26: Blok mac_tx - hodinové a resetovací porty

Pro správnou obsluhu rozhraní RMII jsou použity signály z modulu MIIM. Signál SPEED_100M řídí rychlost přechodů FSM bloku pro obsluhu rozhraní při použité rychlosti 10 Mbit/s nebo 100 Mbit/s. Signál FULL_DUPLEX v logické '0' zapíná v bloku část stavů pro obsluhu CSMA/CD algoritmu.

Port	Směr	Šířka	Popis
SPEED_100M	vstupní	1	Indikace rychlosti rozhraní
FULL_DUPLEX	vstupní	1	Indikace použitého duplexu

Tabulka 3.27: Blok mac_tx - komunikace s modulem MIIM

Blok mac_tx používá porty uvedené v tabulce 3.28 pro komunikaci s vyšší vrstvou (klientem MAC). Vyšší vrstva odesílá bloku jednotlivé oktety rámce, které chce odeslat, portem TX_DATA se současným nastavením logické '1' na portu TX_DATA_VALID. Blok mac_tx každý oktet potvrzuje nastavením logické '1' na portu TX_ACK. Po potvrzení oktetu musí klient MAC na TX_DATA umístit další oktet a znovu čekat na jeho potvrzení. Pokud klient MAC již nemá další oktety rámce k odeslání nastaví port TX_DATA_VALID do logické '0'. Pokud odeslání vytvořeného paketu proběhne úspěšně, je o tom klient MAC informován portem TX_COMPLETE. Pokud během odesílání dojde ke kolizi na médiu (pouze při polovičním duplexu), je o tom klient MAC informován na portu TX_RETRY. Klient MAC poté musí tento rámec bloku mac_tx odeslat znovu od prvního oktetu (blok mac_tx odesílané oktety neukládá). Pokud se ani po patnácti pokusech vlivem kolizí nepodaří rámec odeslat, je nastavem port TX_ERROR do logické '1' a odeslání rámce pomocí CSMA/CD algoritmu je ukončeno.

Port	Směr	Šířka	Popis
TX_DATA	vstupní	8	Odesílaná data
TX_DATA_VALID	vstupní	1	Platnost dat na TX_DATA
TX_ACK	výstupní	1	Potvrzení příjmu TX_DATA
TX_COMPLETE	výstupní	1	Indikace dokončení odesílání
TX_ERROR	výstupní	1	Indikace chyby při odesílání
TX_RETRY	výstupní	1	Žádost o znovu odeslání rámce

Tabulka 3.28: Blok mac_tx - komunikace s vyšší vrstvou

Odesílání Ethernet paketu do PHY probíhá přes rozhraní RMII jehož porty jsou uvedeny v tabulce 3.29.

Port	Směr	Šířka	Popis
RMII_TXD	výstupní	2	RMII - odesílaná data
RMII_TX_EN	výstupní	1	RMII - indikace odesílání

Tabulka 3.29: Blok mac_tx - komunikace s PHY

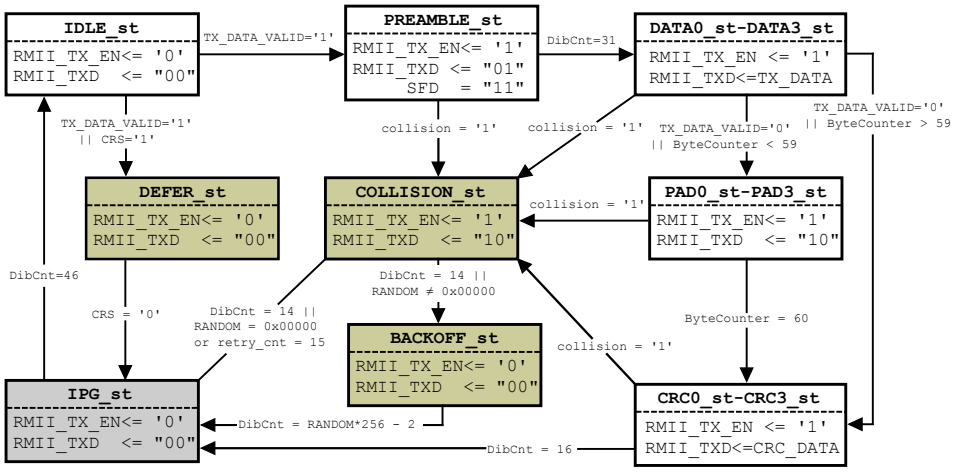
Další porty bloku mac_tx jsou určeny pro komunikaci s bloky mac_tx_crc a mac_tx_random a jsou uvedeny v částech práce věnované těmto blokům.

Funkce bloku je zajištěna stavovým automatem, jehož zjednodušený diagram přechodů je uveden na obrázku 3.21. Stav IDLE_st je počátečním stavem, ve kterém se FSM nachází po resetu bloku.

Na tomto diagramu není vyznačena komunikace s bloky mac_tx_crc a mac_tx_random a nejsou zde vyznačeny hodnoty vnitřních signálů v jednotlivých stavech.

Stavy, které jsou na tomto diagramu vybarveny žlutě (DEFER_st, COLLISION_st a BACKOFF_st) jsou používány pouze pokud je blok v poloduplexním režimu (FULL_DUPLEX = '0'). Při rychlosti rozhraní 10 Mbit/s (SPEED100M = '0') jsou přechody FSM, možné pouze každý desátý takt REF_CLK (vnitřní signál enable).

Stavy pro odesílání dat, FSM a výplně, jsou složené vždy ze 4 různých stavů. Pro větší přehlednost jsou na diagramu sloučeny vždy do jednoho stavu (např. stavy DATA0_st, DATA1_st, DATA2_st, DATA3_st jsou sloučeny jako DATA0_st-DATA3_st).



Obrázek 3.21: Blok mac_tx - diagram přechodů FSM

Ve stavu IDLE_st čeká FSM na příchod prvního oktetu dat, který chce vyšší vrstva odeslat, a na nastavením logické '1' na portu TX_DATA_VALID. Pokud je blok v plně duplexním režimu je možné okamžitě zahájit odesílání preambule, k čemuž slouží stav PREAMBLE_st. Pokud je aktivní poloviční duplex čeká FSM bloku na případné uvolnění média (stav DEFER_st). Pouvolnění média je vložena mezera mezi Ethernet pakety pro správnou funci CSMA/CD algoritmu (stav IPG_st).

Ve stavu PREAMBLE_st je odeslána preambule paketu a SFD, FSM poté přechází do stavu DATA0_st. V tomto stavu je odeslán první dibit oktetu z vyšší vrstvy umístěný na portu TX_DATA, následující dibity jsou odeslány ve stavech DATA1_st, DATA2_st a DATA3_st. Ve stavu DATA3_st je portem TX_ACK potvrzeno odeslání oktetu vz339 vrstvě a vyšší vrstva musí na port TX_DATA umístit následující oktet právě odesílaného rámce. Pokud již vyšší vrstva odeslala modulu MAC všechny oktety, nastaví port TX_DATA_VALID do logické '0' a ukončí tím ukončí odesílání datové části

rámce. Pokud již bylo odesláno 60 oktetů (minimální velikost datové části MAC rámce) přechází FSM do stavu CRC0_st, kde je zahájeno odesílání FCS rámce.

Pokud klient MAC odeslal méně než 60 oktetů je ve stavech PAD0_st, PAD1_st, PAD2_st a PAD3_st odeslána do PHY výplň nulovými oktety do doby než je odesláno celkově 60 oktetů. Poté FSM přechází do stavu CRC0_st. Ve stavech CRC0_st, CRC1_st, CRC2_st a CRC3_st jsou odesílány jednotlivé bajty FCS vypočítané a odesílané blokem mac_tx_crc do bloku mac_tx. Po odeslání celého FCS (4 bajty) přechází FSM do stavu IPG_st.

Ve stavu IPG_st je realizováno čekání po dobu 48 dibitů (na diagramu přechodů je uvedena hodnota 46, kvůli resetu čítače při přechodu mezi stavy CRC3_st a IPG_st). Tato doba odpovídá $0,96$ μs ve 100 Mbit/s a $9,6$ μs v 10 Mbit/s režimu. Po vložení této mezi-paketové mezery přechází FSM do stavu IDLE_st a blok je opět připraven na příchod dat pro odeslání z vyšší vrstvy.

Pokud je aktivní poloviční duplex může během odesílání rámce dojít ke kolizi na médiu. Ta je indikována vnitřním signálem collison, který je vytvořen jako logický součin signálu CRS a signálu transmitting (signál transmmiting je aktivní ve stavech, kdy je TX_EN = '1').

Pokud ke kolizi dojde přechází FSM do stavu COLLISION_st. V tomto stavu je odeslán jam a z bloku mac_tx_random je vyžádána náhodný počet timeslotů pro čekání ve stavu BACKOFF_st. Pokud je tato hodnota nulová, přechází FSM rovnou do stavu IPG_st a poté do stavu IDLE_st, kde musí vyšší vrstva znovu odeslat tento rámec (pokud již neproběhlo 15 pokusů pro odeslání), který se kvůli kolizi nepodařilo odeslat.

Pokud ještě neproběhlo 15 pokusů o odeslání a hodnota počtu timeslotů pro čekání algoritmu CSMA/CD není nulová, přechází FSM bloku do stavu BACKOFF_st. Zde FSM bloku čeká po pseudonáhodný počet timeslotů získaný z bloku mac_tx_random a poté se FSM vrací do stavu IPG_st a dále do stavu IDLE_st, jak bylo zmíněno výše.

Komunikace s bloky mac_tx_crc a max_tx_random během odesílání rámce je podrobněji popsána u popisu těchto bloků v následujících podkapitolách.

Blok mac_tx_crc

Tento blok počítá hodnotu 32-bitové kontrolní sekvence (FCS) odesílaného MAC rámce postupným výpočtem z oktetů přicházejících od klienta MAC.

Výpočet uvedený v kapitole 2.3.1 v části věnované kontrolní sekvenci rámce, není pro zpracování s postupným paralelním odesíláním částí rámce vhodný, protože potřebuje pro výpočet FCS kompletní odesílaný rámec. Další možností výpočtu FCS je výpočet pomocí LFSR (Linear Feedback Shift Register - posuvný registr se zpětnou vazbou). Tento výpočet je jednoduchý, ale je použitelný pouze pro sériová data, případně při použití rychlejšího hodinového taktu a serializaci dat na vstupní sběrnici.

Pro výpočet s paralelními daty je možné použít předem vypočítané tabulky zbytků po dělení polynomu pro všechny kombinace vstupních dat. Zbytek po dělení pro vstupní paralelní data se sečte s hodnotou zbytku po dělení z předchozího stavu. Velikost paměti S pro uložení předem vypočtených hodnot závisí na šířce vstupních dat N a délce kontrolní sekvence M. Tato velikost je dána vztahem: $S = N \cdot 2^M$. Při použití datové sběrnice N = 8 bitů a 32-bitové FCS pro Ethernet je potřebná velikost paměti 1 KiB (8192 bitů). Výhodou tohoto způsobu výpočtu je jeho rychlost pro větší šířky datové sběrnice, ale velikost potřebné paměti rychle stoupá.

Jinou možností výpočtu FCS je paralení výpočet CRC s využitím logické funkce XOR popsaný například v článku [13]. Hodnota jednotlivých bitů CRC, které jsou použity v FCS poli MAC rámce, je dána kombinační XOR logickou funkcí vstupního oktetu dat (*DataIn*) a hodnotou CRC v předchozím stavu (*CrcIn)*. Tato metoda využívá výpočtu matice H_1 pro one-hot vstupní data (pouze jeden nastavený bit - 0x01h,0x02h, 0x04h, ...) a nulový stav předchozího CRC a matice H_2 pro nulová data a one-hot stav předchozí hodnoty CRC.

Pro hodnotu 32 bitového CRC a 8 bitové šířky datové sběrnice dostaneme matici H_1 s 32 sloupci a 8 řádky a matici H_2 s 32 sloupci a 32 řádky. Jednotlivé hodnoty prvků v matici jsou vypočteny pomocí sériového výpočtu CRC pro výše zmíněná vstupní data a generující polynom pro CRC uvedený v kapitole 2.3.1. Každý nastavený bit j ve sloupci i matice H_1 se účastní výpočtu bitu *CRC(i)* jako *DataIn(j)*. Podobně každý nastavený bit j ve sloupci i matice H_2 se účastní výpočtu bitu *CRC(i)* jako *CrcIn(j)*. Všechny zúčastněné vstupy *CrcIn(j)* a *DataIn(j)*, které tvoří *CRC(i)* jsou vzájemně sečteny pomocí logického exkluzivního součtu (XOR). Podrobný popis algoritmu pro sestavení těchto rovnic je uveden v [13].

Tato metoda využívá lineárních vlastností CRC a lineární nezávislosti řádků výše zmíněných matic. Výsledkem této metody je pro požadovanou kontrolní sekvenci rámce 32 logických rovnic pro výpočet jednotlivých bitů CRC ze vstupního oktetu dat *DataIn* a předchozího stavu *CrcIn*. Postupným výpočtem pro vstupní oktety přicházející z vyšší vrstvy a ukládáním vnitřního stavu CRC je hodnota požadovaného FCS vypočtena ihned po příchodu posledního oktetu odesílaného rámce. Z tohoto důvodu nemusí modul MAC jednotlivé bajty rámce ukládat a blok nezanáší přidanou latenci potřebnou při výpočtu FCS s využim sériového LFSR.

Blok mac_tx_crc realizuje výpočet FCS pomocí této paralelní metody výpočtu CRC. Pro správnou činnost bloku mac_tx_crc pro jeho využití blokem mac_tx je nutné použít na portu REF_CLK hodinový signál s frekvencí 50 MHz. Reset bloku je synchronní s REF_CLK a je v projektu generován modulem MMCM.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.30: Blok mac_tx_crc -hodinové a resetovací signály

Porty modulu mac_tx_crc, propojující tento blok s blokem mac_tx, jsou následující:

Port	Směr	Šířka	Popis
RESET_CRC	vstupní	1	Reset vnitřního stavu výpočtu
CALC_CRC	vstupní	1	Zapnutí výpočtu FCS pomocí CRC
SEND_CRC	vstupní	1	Odesílání FCS/výpočet nového stavu
DATA_IN	vstupní	8	Vstupní oktet dat
CRC	výstupní	8	Vypočítaný bajt FCS pro mac_tx

Tabulka 3.31: Blok mac_tx_crc - komunikace s blokem mac_tx

Logická hodnota na portech RESET_CRC, CALC_CRC a SEND_CRC definuje vnitřní stav bloku, který podle jejich kombinací provádí reset výpočtu CRC, výpočet následujícího stavu vnitřní hodnoty CRC registru, posuv bajtů vypočítaného FCS na port CRC(7:0), případně jejich kombinace. Normální činnost bloku je následující: při začátku odesílání preambule Ethernet paketu je blok resetován, po odeslání SFD je zapnut výpočet CRC a s každým přicházejícím datovým oktetem rámce na portu DATA_IN(7:0) je počítána nová hodnota CRC. Datový oktet je před výpočtem navíc bitově otočen, protože použité FCS je počítáno od LSB. Pomocí rovnic pro výpočet jednotlivých bitů CRC (viz minulá strana popisu bloku) je z přicházejícího oktetu rámce a minulé hodnoty CRC vypočítána nová hodnota CRC.

Po přijetí posledního rámce je zastaven výpočet CRC a jednotlivé oktety vypočítané kontrolní sekvence rámce (po další bitovém otočení kvůli odesílání dat od LSB na RMII rozhraní) jsou postupně odeslány bloku mac_tx. V případě že je rychlost rozhraní 10 Mbit/s jsou vstupní data odesílána blokem mac_tx s de setinovou rychlostí (každých 800 ns) a port SEND_CRC je nastavován periodicky s každým odeslaným dibitem. Stejně jako u rychlosti 100 Mbit/s je další bajt vypočítaného FSC odeslán po čtyřech taktech, kdy je nasteven signál SEND_CRC po ukončení výpočtu FCS (bajty jsou tedy odeslány vždy po dokončení odeslání bajtu na RMII rozhraní).

Blok mac_tx_random

Blok mac_tx_random je částí modulu MAC určený pro odesílání dat přijatých od klienta MAC pomocí Ethernet paketů odesílaných rozhraním RMII do PHY. Tento blok slouží k získání pseudonáhodného, celého čísla v intervalu maximálně 0 až 1023, které je použito v bloku mac_tx pro určení počtu timeslotů pro čekaní (backoff) algoritmu CSMA/CD.

Pokud na médiu vznikne kolize, je každou stanicí která ji detekuje odeslán jam. Při n-tém odeslání rámce poté každá stanice zvolí celočíselný náhodný počet timeslotů r (pro přenosovou rychlost 10 nebo 100 Mbit/s je to doba pro odeslání 512 bitů, tedy $51,2\ \mu s$ respektive $5,12\ \mu s$), po kterou bude čekat, podle následujícího vztahu: $0 \leqslant r < 2^k$. Kde $k = min(n, 10)$.

Při první kolizi každá stanice čeká 0 nebo 1 timeslot. Při druhé 0 až 3 timesloty, při třetí 0 až 7 timeslotů, ..., při desáté a vyšší 0 až 1023 timeslotů.

Blok max_tx_random obsahuje LFSR s maximální délkou pro generování desetibitového čísla. Jednotlivé bity tohoto čísla jsou při nastavení portu REQ odeslány na port RANDOM. Počet odeslaných bitů je řízen portem ATTEMPT, který poskytuje počet provedených opakování odesílání rámce. Tento způsob zajišťuje generování celých čísel z daného intervalu, tak jak je uvedeno v CSMA/CD algoritmu, který byl zmíněn výše.

Pro správnou činnost bloku mac_tx_random je nutné použít hodinový signál REF_CLK s frekvencí 50 MHz. Reset bloku je synchronní s REF_CLK a je v projektu generován modulem MMCM.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.32: Blok mac_tx_random -hodinové a resetovací signály

Porty modulu mac_tx_random, propojující tento blok s blokem mac_tx, jsou následující:

Port	Směr	Šířka	Popis
REQ	vstupní	1	Žádost o psudonáhodné celé číslo
ATTEMPT	vstupní	4	Probíhající pokus odesílání rámce
RAND	výstupní	10	Vygenerované číslo pro CSMA/CD backoff

Tabulka 3.33: Blok mac_tx_random - komunikace s blokem mac_tx

Pokud tedy dojde ke kolizi v poloduplexním režimu, je blokem mac_tx ve stavu COLLISION vygenerován požadavek pro blok mac_tx_random na portu REQ se současným nastavením počtu provedených opakování odesílání rámce na portu ATTEMPT. Tím je do výstupního registru na portu RAND uložena současná hodnota z LFSR omezená na požadovaný počet bitů podle hodnoty na portu ATTEMPT.

Na obrázku 3.22 je uveden VHDL kód realizující výše popsanou činnost bloku. První proces (řádky 1 ař 10) je LFSR kontinuálně generující pseudonáhodnou posloupnost. Zpětná vazba LFSR (řádek 12) je realizována signálem Feedback.

V druhé části kódu (řádky 14 až 23) je výběr počtu bitů realizující vybrání velikosti intervalu podle počtu pokusů nastaveného na portu ATTEMPT. Poslední proces (řádky 25 ař 36) realizuje výstupní registr, na jehož výstup je uložena hodnota z LFSR při nastavení logické '1' na portu REQ.

```
1  process(REF_CLK)
2  begin
3    if rising_edge(REF_CLK) then
4      if RESET = '1' then
5          x <= (others => '0');
6      else
7        x <= x(8 downto 0) & Feedback;
8      end if;
9    end if;
10 end process;
11
12 Feedback <= not (x(6) xor x(9));  -- zpetna vazba LFSR
13
14 Rand_i(0) <= x(0);
15 Rand_i(1) <= x(1) when (ATTEMPT > "0001") else '0';
16 Rand_i(2) <= x(2) when (ATTEMPT > "0010") else '0';
17 Rand_i(3) <= x(3) when (ATTEMPT > "0011") else '0';
18 Rand_i(4) <= x(4) when (ATTEMPT > "0100") else '0';
19 Rand_i(5) <= x(5) when (ATTEMPT > "0101") else '0';
20 Rand_i(6) <= x(6) when (ATTEMPT > "0110") else '0';
21 Rand_i(7) <= x(7) when (ATTEMPT > "0111") else '0';
22 Rand_i(8) <= x(8) when (ATTEMPT > "1000") else '0';
23 Rand_i(9) <= x(9) when (ATTEMPT > "1001") else '0';
24
25 process(REF_CLK)
26 begin
27   if rising_edge(REF_CLK) then
28     if RESET = '1' then
29       RAND <= (others => '0');
30     else
31       if REQ = '1' then
32         RAND <= Rand_i;
33       end if;
34     end if;
35   end if;
36 end process;
```

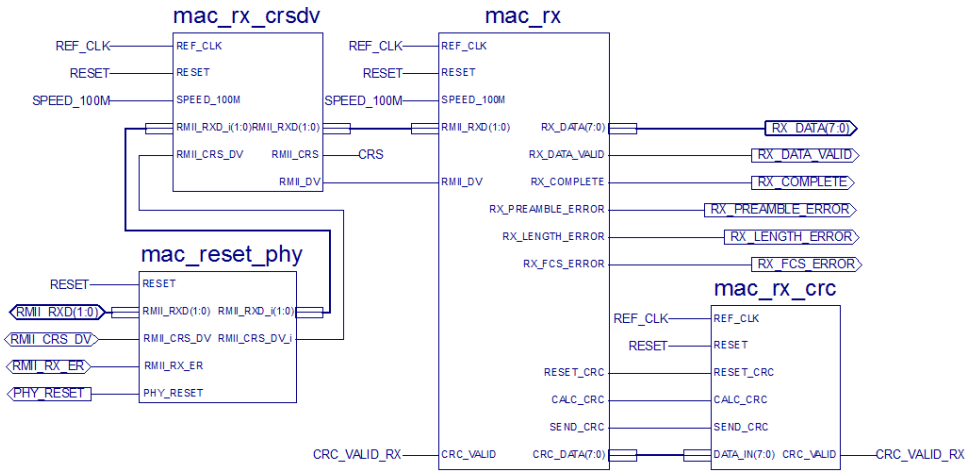
Obrázek 3.22: Blok mac_tx_random - VHDL

Blok mac_rx

Tento blok slouží k příjmu MAC rámce (respektive Ethernet paketu), jeho upravení a odeslání vyšší vrstvě. Signály rozhraní RMII jsou přivedeny do bloku mac_reset_phy, který je předává (pokud není aktivní reset) bloku mac_rx_crsdv. Tento blok odděluje signál RMII_CRS_DV na signály CRS (pro blok mac_tx) a DV (pro blok mac_rx) a zpožďuje signál RXD(1:0), tak aby si tyto signály vzájemně časově odpovídaly.

Blok mac_rx přijímá signál RXD(1:0) a DV, ze kterých obnovuje jednotlivé oktety přijímaného rámce a odesílá je vyšší vrstvě na portu RX_DATA. Blok dále odstraňuje preambuli paketu a pole FCS, které nejsou předány vyšší vrstvě. Blok provádí filtrování rámců určených pro jinou MAC adresu, než je broadcast a pevně nastavená MAC adresa (lze měnit pouze při syntéze). Blok poskytuje vyšší vrstvě informaci o těchto chybách rámce: chybné pole FCS, chybná preambule, chybná délka rámce (menší než minimální délka 64 oktetů, větší než maximální délka 1518 (nebo 1522 pro Q-taggovaný rámec) oktetů a délka nesouhlasící s hodnotou v poli délka).

Blokové schéma těchto bloků určených pro příjem Ethernet paketu a předání MAC rámce a jeho stavu vyšší vrstvě je uvedeno na obrázku 3.23.



Obrázek 3.23: Modul MAC - přijímající část

Pro správnou činnost bloku mac_rx je nutné použít hodinový signál REF_CLK s frekvencí 50 MHz pro vzorkování signálů RMII rozhraní. Reset bloku je synchronní s REF_CLK a je v projektu generován modulem MMCM.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.34: Blok mac_rx - hodinové a resetovací porty

Blok mac_rx používá porty uvedené v tabulce 3.35 pro komunikaci s vyšší vrstvou (klientem MAC). Vyšší vrstva přijímá jednolivé oktety přijímaného rámce posílané blokem mac_rx portem RX_DATA se současným nastavením logické '1' na portu RX_DATA_VALID. Po odeslání všech datových oktetů rámce (od cílové MAC adresy po poslední oktet datové části) je vyšší vrstvě indikováno dokončení příjmu nastevením logické '1' na portu RX_COMPLETE. Zároveň jsou vyšší vrstvě na příslušných portech indikovány případné chyby, ke kterým došlo během příjmu rámce (chybná preambule, FCS nebo délka rámce).

Port	Směr	Šířka	Popis
RX_DATA	výstupní	8	Přijatá data
RX_DATA_VALID	výstupní	1	Platnost dat RX_DATA
RX_COMPLETE	výstupní	1	Indikace ukončení příjmu
RX_PREAMBLE_ERROR	výstupní	1	Indikace chyby preambule
RX_FCS_ERROR	výstupní	1	Indikace chybného FCS
RX_LENGTH_ERROR	výstupní	1	Indikace chybné délky

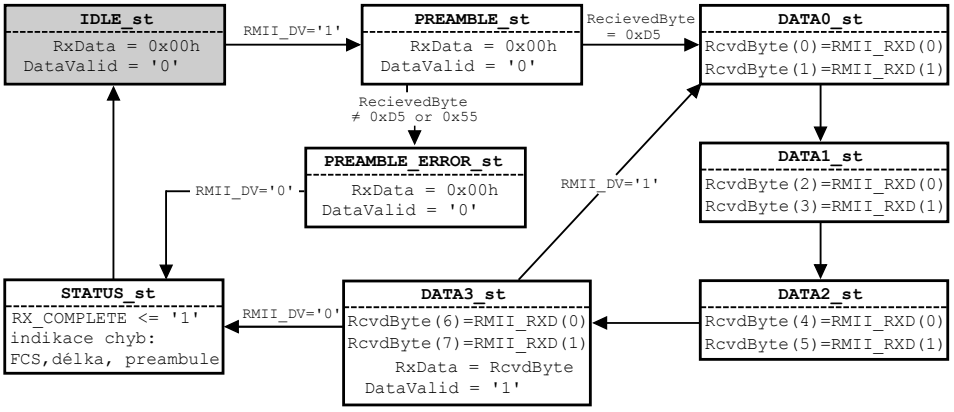
Tabulka 3.35: Blok mac_rx - komunikace s vyšší vrstvou

Příjem Ethernet paketu z PHY probíhá přes rozhraní RMII a po úpravě blokem mac_rx_crsdv jsou blokem mac_rx přijímány signály pomocí portů uvedených tabulce 3.36.

Port	Směr	Šířka	Popis
RMII_RXD	vstupní	2	RMII - přijímaná data
RMII_CRS	vstupní	1	RMII - indikace dat

Tabulka 3.36: Blok mac_rx - komunikace s PHY

Funkce bloku je zajištěna stavovým automatem s diagramem přechodů uvedeným na obrázku 3.24. Stav IDLE_st je počátečním stavem, ve kterém se FSM nachází po resetu bloku nebo pokud není přijímán paket z PHY. Na tomto diagramu není vyznačena komunikace s blokem mac_rx_crc a nejsou zde vyznačeny hodnoty vnitřních signálů v jednotlivých stavech. Při rychlosti rozhraní 10 Mbit/s (SPEED_100M = '0') jsou přechody FSM možné pouze každý desátý takt REF_CLK (k řízení přechodů je použit vnitřní signál enable).



Obrázek 3.24: Blok mac_rx - diagram přechodů FSM

Ve stavu IDLE_st čeká FSM bloku na nastavení portu RMII_DV do logické '1'. Ta indikuje začátek příchodu dekódované preambule paketu z PHY, jejíž první dibit je uložen do proměnné RcvdByte. Současně je zrušena indikace dokončení příjmu předchozího rámce a jeho případných chyb pro vyšší vrstvu, je resetován blok mac_rx_crc a FSM bloku přechází do stavu PREAMBLE_st.

V tomto stavu jsou postupně ukládány jednotlivé dibity přicházejících oktetů preambule do proměnné RcvdByte. Po přijetí celého oktetu je otestováno, zda je jedná o platný oktet, který se může vyskytnout v preambuli. Tedy oktet s bitovou hodnotou "01010101"nebo "11010101"(řazení jednotlivých dibitů přicházejících na RMII_RXD je uvedeno v kapitole 2.5.2). Pokud je z PHY přijat oktet s jinou hodnotou, je přicházející paket považován za neplatný a FSM přechází do stavu PREAMBLE_ERROR_st, kde čeká na ukončení příjmu tohoto paketu. Poté se FSM vrací do stavu IDLE_st, kde čeká na příjem dalšího paketu z PHY.

V případě detekce oktetu obsahujícího SFD (tedy oktet s hodnotou 0xD5), přechází FSM do stavu DATA0_st. V tomto stavu je uložen první dibit přijímaného oktetu do proměnné RcvdByte a bloku mac_rx_crc je indikováno na portu CALC_CRC zapnutí výpočtu CRC pro ověření FCS rámce.

V následujících stavech DATA1_st a DATA2_st jsou pouze uloženy další dibity přijímaného oktetu a FSM přechází do stavu DATA3_st.

Zde je uložen poslední dibit oktetu a celý přijatý oktet je uložen do vnitřního signálu RxData. Zároveň je signálem DataValid indikována platnost těchto dat. V tomto stavu je po jednotlivých oktetech ukládána cílová MAC adresa přijímaného rámce, která je po jejím přijetí porovnána s MAC adresou

uloženou v konstantě FPGA_MAC_ADDRESS a s adresou pro broadcast. Pokud adresa přijatá v MAC rámci nesouhlasí je nastaven vnitřní signál MAC_error do logické '1'. Tím je resetován klopný obvod odesílající data portem RX_DATA a vyšší vrstvě, tak rámce s jinou MAC adresou nejsou předány (signál na výstupním portu RX_DATA je signál RxData zpožděný o 6 oktetů kvůli této filtraci rámcu).

V tomto stavu je také uloženo a testováno pole Typ/Délka. Po jeho uložení je otestováno zda hodnota v tomto poli je menší nebo rovna hodnotě 1500 (v tomto poli se nachází délka datové části rámce), větší než 1536 (v tomto poli se nachází typ rámce) nebo je tato hodnota jiná (není definováno, rámec je označen jako chybný). Pokud pole obsahuje typ rámce (mimo hodnoty 0x8100h pro hlavičku IEEE 802.1Q) nebo je hodnota v poli Délka 1 až 46 není počítána předpokládaná velikost rámce testovaná ve stavu STATUS_st po přijetí celého rámce. V případě že pole obsahuje v poli Délka hodnotu v rozmezí 46 až 1500, je vypočítána předpokládaná délka celého rámce, která je uložena do vnitřního signálu FrameLength. Pokud rámec v poli Typ/Délka obsahuje hodnotu 0x8100h je proveden podobný test pole Typ/Délka umístěného po hlavičce 802.1Q (VLAN tagging viz. kapitola 2.3.1).

Přechod ze stavu DATA3_st je řízen signálem na portu RMII_DV. Pokud je detekována hodnota logická '1' přechází FSM zpět do stavu DATA0_st a blokem je přijímán další oktet rámce. Zároveň je přijatý oktet odeslán bloku mac_rx_crc na portu CRC_DATA spolu s nastavením logické '1' na portu SEND_CRC. V případě že je v tomto stavu na portu RMII_DV logická '0', přechází FSM bloku do stavu STATUS_st. Při tomto přechodu je zrušena indikace platnosti dat (signál DataValid) a je ukončen příjem oktetů rámce.

Ve stavu STATUS_st je nastavena logická '1' na portu RX_COMPLETE. Tato hodnota indikuje vyšší vrstvě dokončení příjmu rámce. Pokud došlo k chybě při příjmu preambule je nastaven port RX_PREAMBLE_ERROR. Pokud z bloku mac_rx_crc není na portu CRC_VALID detekována logická '1' a tedy přijaté FCS se liší od FCS vypočítaného z přijatých dat (viz. popis bloku mac_rx_crc), je nastaven port RX_FCS_ERROR. Porovnáním hodnoty přijatých oktetů s minimální velikostí rámce (64 oktetů), maximální velikostí rámce (1518 pro klasický a 1522 pro 802.1Q rámec) a případně hodnoty přijaté v poli Délka je ověřeno zda má přijatý rámec správnou velikost. Pokud ne, je vyšší vrstvě tato chyba indikována portem RX_LENGTH_ERROR.

Signály RxData a DataValid generované FSM jsou zpožděny o 6 oktetů (kvůli filtraci rámců s jinou MAC adresou). K tomu slouží posuvné registry RxData_D a DataValid_D. Ze zpožděného signálu jsou také odstraněny poslední čtyři oktety obsahující FCS rámce. Takto vzniklé signály jsou poté odesílány na porty RX_DATA a RX_DATA_VALID.

Blok mac_rx_crc

Tento blok pomocí CRC výpočtu ověřuje integritu přijímaného rámce kontrolou datové části rámce a pole FCS. Blok z přicházejících oktetů rámce, které odesílá blok mac_rx počítá hodnotu CRC.

Takto vypočítaná hodnota je v bloku porovnávána s hodnotou hexadecimální 0xC704DD7Bh, která při výpočtu CRC vždy vznikne v případě výpočtu CRC z korektně přijatého rámce s přidaným pole FCS. Tato hodnota souvisí s generujícím polynomem použitým při CRC výpočtu, bitovou inverzí vypočítaného CRC, které je připojeno jako FCS a pořadím odesílání bitů odpovídajícím jednotlivým mocninám zbytku po dělení.

Blok mac_rx_crc realizuje výpočet CRC pomocí paralelní metody zmíněné v části věnované bloku mac_tx_crc (kapitola 3.2.3). Funkce a činnost bloku je téměř identická s tímto blokem a je uvedena v kapitole popisující blok mac_tx_crc. Jediným rozdílem je, že tento blok neposkytuje jako výstup jednotlivé bajty vypočítaného CRC, ale pouze indikaci rovnosti vypočtené hodnoty CRC s hodnotou 0xC704DD7Bh.

Pro správnou činnost bloku mac_rx_crc pro jeho využití blokem mac_rx je nutné použít na portu REF_CLK hodinový signál s frekvencí 50 MHz. Reset bloku je synchronní s REF_CLK a je v projektu generován modulem MMCM.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.37: Blok mac_rx_crc -hodinové a resetovací signály

Porty modulu mac_rx_crc, propojující tento blok s blokem mac_rx, jsou následující:

Port	Směr	Šířka	Popis
RESET_CRC	vstupní	1	Reset vnitřního stavu výpočtu
CALC_CRC	vstupní	1	Zapnutí výpočtu FCS pomocí CRC
SEND_CRC	vstupní	1	Odesílání FCS/výpočet nového stavu
DATA_IN	vstupní	8	Vstupní oktet dat
CRC_VALID	výstupní	8	Indikace správného CRC

Tabulka 3.38: Blok mac_rx_crc - komunikace s blokem mac_rx

3.3 Demonstrace funkce modulu

Druhou praktickou částí této práce je návrh a vytvoření jednoduché ukázky pro demonstraci využití rozhraní Ethernet na přípravku Nexys 4. Pro samotou obsluhu rozhraní byly využity moduly uvedené v kapitole 3.2.

K blokům a modulům pro realizaci obsluhy rozhraní Ethernet popsaných v kapitole 3.2.3 byl přidán další blok pro generování odesílaných dat do modulu MAC, blok pro zpracování přijatých dat z modulu MAC, blok pro zobrazení těchto dat na sedmisegmentovém displeji a blok pro zobrazení stavů bloků na LED diodách přípravku.

3.3.1 Ověření odesílání rámců

Pro otestování funkce odesílání ethernetových paketů byla zvolena varianta odeslání předem vytvořeného IP paketu se zapouzdřeným UDP datagramem, do kterého jsou vloženy dva bajty dat, ve kterých je zakódována poloha nastavení přepínačů SW0 až SW15 na přípravku v době zahájení odesílání paketu. Odesílání rámce je zahájeno stisknutím pravého mikrospínače BTNR na přípravku a další odeslání je možné až po jeho opětovném uvolnění. Indikace stavů rozhraní, bloku mac_tx a bloku tx_test je zobrazována pomocí LED na přípravku (jednotlivé pozice jsou popsány v části kapitoly 3.3.2 popisující blok led_switch).

MAC rámec odesílaný blokem tx_test má následující obsah (není zde uvedena přidaná výplň na doplnění na minimální velikost rámce a pole FCS, které jsou doplněny modulem MAC):

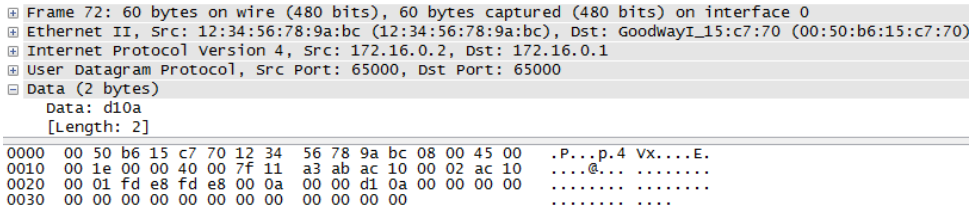
<table>
<tr><th></th><th colspan="8">Oktet 1</th><th colspan="8">Oktet 2</th><th colspan="8">Oktet 3</th></tr>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>0
3</td><td colspan="24">Cílová MAC adresa</td></tr>
<tr><td>6
9</td><td colspan="24">Zdrojová MAC adresa</td></tr>
<tr><td>12</td><td colspan="16">Typ / Délka</td><td colspan="4">Verze IP</td><td colspan="4">Délka záhlaví</td></tr>
<tr><td>15</td><td colspan="8">TOS/DCSP</td><td colspan="16">Celková délka</td></tr>
<tr><td>18</td><td colspan="16">Identifikace</td><td>-</td><td>DF</td><td>MF</td><td colspan="5"></td></tr>
<tr><td>21</td><td colspan="8">Offset fragmentace</td><td colspan="8">TTL</td><td colspan="8">Protokol</td></tr>
<tr><td>24</td><td colspan="16">Kontrolní součet IP hlavičky</td><td colspan="8"></td></tr>
<tr><td>27</td><td colspan="24">Zdrojová IP adresa</td></tr>
<tr><td>30</td><td colspan="24">Cílová IP adresa</td></tr>
<tr><td>33</td><td colspan="8"></td><td colspan="16">Zdrojový UDP port</td></tr>
<tr><td>36</td><td colspan="16">Cílový UDP port</td><td colspan="8">Délka</td></tr>
<tr><td>39</td><td colspan="8"></td><td colspan="16">Kontrolní součet UDP</td></tr>
<tr><td>42</td><td colspan="16">Data</td><td colspan="8"></td></tr>
</table>

Obrázek 3.25: Obsah rámce odesílaného blokem tx_test

Hodnoty v jednotlivých polích odesílaného rámce jsou následující (lze je před syntézou projektu změnit v konstantě bloku mac_tx):

- **Cílová MAC adresa**: MAC adresa zařízení komunikujícího s přípravkem (adresa 00:50:B6:15:C7:70),
- **Zdrojová MAC adresa**: MAC adresa přiřazená přípravku (lokální adresa 12:34:56:78:9A:BC),
- **Typ/Délka**: Typ Internet Protocol verze 4 (0x0800h),
- **Verze IP**: IPv4 (0x4h),
- **Délka záhlaví**: pět 32-bitových slov (0x5h),
- **TOS/DSCP**: 0 - Best Effort, bez ECT (0x0000h),
- **Celková délka**: 30 oktetů (0x1E),
- **Identifikace**: není fragmentováno, nedůležitá hodnota (0x00h),
- **-**: musí být nula (0),
- **DF**: Don't Fragment-zakázána fragmentace (1),
- **MF**: More Fragments: nenásledují další fragmenty (0),
- **Offset fragmentace**: první fragment, není fragmentováno (0x0000h),
- **TTL**: Time To Live, doba života paketu (0x7Fh - 127),
- **Protokol**: typ protokolu v datové části IP paketu - UDP (0x11h),
- **Kontrolní součet IP hlavičky**: předem vypočítaná hodnota kontrolního součtu hlavičky, při změně této hlavičky je nutné tuto hodnotu vypočítat a nahradit (0xA3AB),
- **Zdrojová IP adresa**: IP adresa FPGA (172.16.0.2 - 0xAC100002h),
- **Cílová IP adresa**: IP adresa cílového zařízení (172.16.0.1 - 0xAC10000h1),
- **Zdrojový UDP port**: hodnota zdrojového UDP port (65000 - 0xFDE8h),
- **Cílový UDP port**: hodnota cílového UDP portu (65000 - 0xFDE8h),
- **Délka**: celková velikost UDP datagramu (10 - 0x000Ah),
- **Kontrolní součet UDP**: je nastaven na hodnotu značící nepoužití kontrolního součtu z důvodu variabilních oktetů v datové části UDP datagramu a nepočítání kontrolního součtu v FPGA (0x0000),
- **Data**: datová část UDP (stav přepínačů na přípravku 0x0000h-0xFFFFh).

MAC rámec odeslaný z FPGA zachycený v cílovém počítači pomocí packet snifferu Wireshark je uveden na obrázku 3.26. Na tomto zachyceném paketu jsou vidět výše zmíněné hodnoty jednotlivých polí rámce a výplň přidanou modulem MAC.



Obrázek 3.26: Odeslaný rámec - Wireshark

Blok tx_test

Tento blok slouží k odesílání jednotlivých oktetů rámce MAC popsaného v kapitole 3.3.1. Tento rámec má v datové části UDP diagramu dva bajty dat, které obsahují polohu přepínačů na přípravku Nexys 4. UDP datagram je zapouzdřen uvnitř IP paketu směrovaného na pevně nastavenou IP a MAC adresu příjemce (tento MAC rámec je uložen v konstantě Udp_Packet a je možné ho měnit pouze před syntézou projektu) Tento MAC rámec je při stisku mikrospínače na přípravku (BTNR) odeslán modulu MAC a je jím odeslán v ethernetovém paketu RMII rozhraním do PHY na přípravku, který ho odešle na připojené fyzické médium.

Pro správnou činnost bloku tx_test je nutné použít hodinový signál REF_CLK s frekvencí 50 MHz kvůli komunikaci s blokem MAC. Reset bloku je synchronní s REF_CLK a je v projektu generován modulem MMCM.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.39: Blok tx_test - hodinové a resetovací porty

Činnost bloku je ovlivněna stavem ethernetového rozhraní, které bloku poskytuje modul MIIM. V případě že je spojení neaktivní je zabráněno odesílání rámců do modulu MAC. Podobně jako u bloků mac_tx a mac_rx je v případě použití rychlosti rozhraní 10 Mbit/s zpomalen přechod mezi stavy FSM bloku na desetinu vnitřním signálem enable.

Port	Směr	Šířka	Popis
LINK_STATUS	vstupní	1	Indikace stavu spojení
SPEED_100M	vstupní	1	Indikace rychlosti rozhraní

Tabulka 3.40: Blok tx_test - komunikace s modulem MIIM

S modulem MAC komunikuje blok pomocí portů uvedených v tabulce 3.41. Data jsou blokem posílána po oktetech portem TX_DATA a jejich platnost je indikována na portu TX_DATA_VALID. Modul MAC poskytuje bloku informace o stavu odesílání rámce na portech TX_ACK, TX_COMPLETE, TX_RETRY a TX_ERROR, popis těchto signálů je v kapitole 3.2.3.

Port	Směr	Šířka	Popis
TX_DATA	výstupní	8	Odesílané oktety do MAC
TX_DATA_VALID	výstupní	1	Platnost dat na TX_DATA
TX_ACK	vstupní	1	Potvrzení příjmu oktetu
TX_COMPLETE	vstupní	1	Indikace dokončení odesílání
TX_ERROR	vstupní	1	Indikace chyby při odesílání
TX_RETRY	vstupní	1	Žádost o znovu odeslání rámce

Tabulka 3.41: Blok tx_test - komunikace s modulem MAC

Pro ovládání a zobrazení činnosti bloku jsou použity porty uvedené v tabulce 3.42. Signál SW je připojen k přepínačům SW0 až SW15 na přípravku a určuje hodnotu v datové části UDP datagramu odeslaného při stisku tlačítka připojeného k portu SEND (signál z tlačítka je upraven blokem pro odstranění zákmitů popsaném v kapitole 3.2.1). Signál STATE_TX slouží k indikaci stavu FSM a je připojen pomocí bloku led_switch k LED diodám přípravku. Signál DISPLAY_DATA odesílá zakódované znaky do bloku display pro dekódování a zobrazení pomocí pravé části sedmisegmentového displeje přípravku (DISP2).

Port	Směr	Šířka	Popis
SW	vstupní	16	Stav přepínačů přípravku
SEND	vstupní	1	Tlačítko pro odeslání rámce
STATE_TX	výstupní	5	Stav FSM bloku tx_test
DISPLAY_DATA	výstupní	20	Data pro blok display

Tabulka 3.42: Blok tx_test - komunikace s periferiemi

Základním stavem FSM je stav IDLE_st, který je výchozím stavem při resetu a dále se v něm blok nachází po dokončení odesílání rámce. V tomto stavu je vyhodnocován stav portu SEND, kterým je indikována žádost o odeslání rámce s aktuální hodnotou stavu přepínačů přípravku. Pokud dojde k nastavení portu SEND do logické '1' (tedy stisku BTNR) a zároveň je ethernetové spojení aktivní (port LINK_STATUS = '1') je uložena hodnota na portu SW do vnitřního signálu udp_data a FSM bloku přechází do stavu SEND_FIRST_st. Pokud je spojení neaktivní zůstává FSM ve stavu IDLE_st a portem DISPLAY_DATA je odeslána hodnota pro zobrazení znaků ---- na sedmisegmentovém displeji.

Stav SEND_FIRST_st slouží k odeslání prvního oktetu rámce uloženého v konstantě Udp_Packet do modulu MAC pomocí portů TX_DATA. V tomto stavu je modulu MAC signálem TX_DATA_VALID v logické '1' indikována žádost o odeslání rámce. Po příchodu potvrzení na portu TX_ACK (po dokončení odesílání preambule Ethernet paketu) přechází FSM bloku do stavu SEND_st.

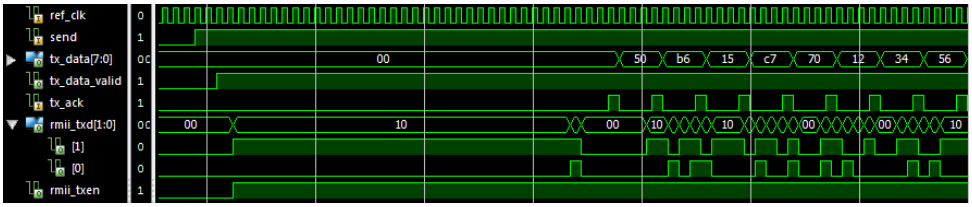
V tomto stavu jsou odeslány ostatní oktety rámce uloženého v konstantně Udp_Packet a hodnota stavu spínačů uložená při přechodu ze stavu IDLE_st (udp_data). Po odeslání celého rámce je nastaven port TX_DATA_VALID do logické '0' a FSM přechází do stavu WAIT_st.

Ve stavu WAIT_st čeká FSM bloku na dokončení odesílání rámce modulem MAC indikované logickou '1' na portu TX_COMPLETE. Pokud bylo odesílání v pořádku je nastaven vnitřní signál send_complete a blok čeká na návrat signálu na portu SEND do logické '0' (uvolnění BTNR). Zároveň je portem DISPLAY_DATA odeslána hodnota pro zobrazení odeslaného stavu spínačů bloku display pro zobrazení na sedmisegmentovém displeji.

Pokud se blok nachází ve stavu SEND_FIRST_st, SEND_st nebo WAIT_st je možné, že během odesílání rámce je modulem MAC ukončeno odesílání vlivem kolize na médiu (indikováno pomocí TX_RETRY = '1') nebo případné ukončení odesílání kvůli vzniku 15-ti kolizí při po sobě následujících pokusech o odeslání (indikováno pomocí TX_ERROR = '1'). V tomto případě blok tx_test ukončí odesílání oktetů modulu MAC a FSM bloku přechází do stavu WAIT_COLLISION_st respektive WAIT_ERROR_st.

Ve stavu WAIT_COLLISION_st je portem DISPLAY_DATA odeslána hodnota pro zobrazení znaků CoL na sedmisegmentovém displeji a je zde čekáno na opětovné uvolnění tlačítka připojeného k portu SEND, po kterém se FSM bloku vrací po uvolnění tlačítka BTNR do stavu SEND_FIRST_st a je opakován pokus o odeslání rámce. Stav WAIT_ERROR_st je podobný stavu WAIT_COLLISION_st pouze je na displej odeslána hodnota pro zobrazení zanků Err a FSM se vrací do stavu IDLE_st a není zahájen další pokus o odeslání rámce.

Na obrázku 3.20 je uvedena simulace začátku odesílání ethernetového paketu modulem MAC a jeho komunikace s blokem tx_test. Je zde vidět odesílání jednotlivých oktetů do modulu MAC, jejich potvrzování modulem MAC na portu TX_ACK a generované dibity odesílané na RMII rozhraní.

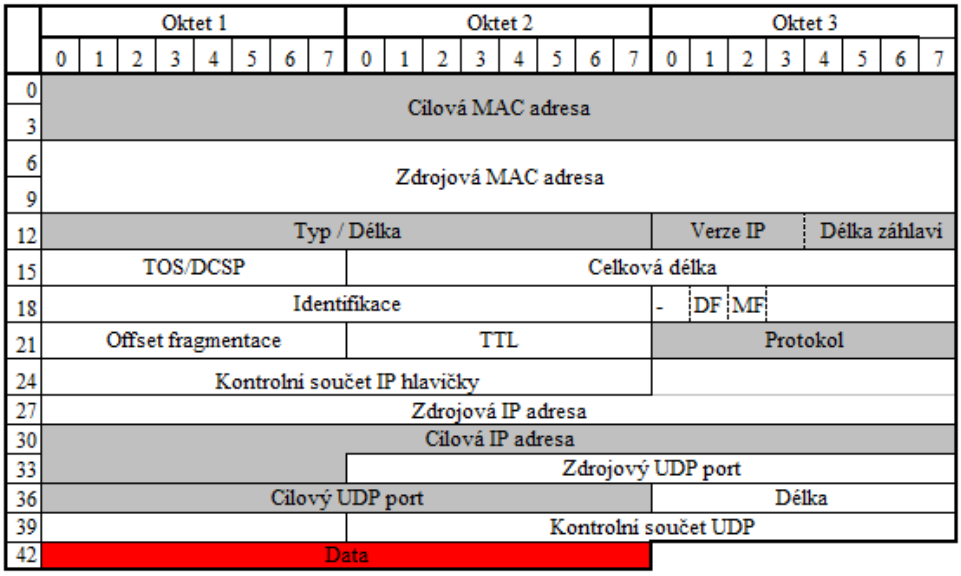


Obrázek 3.27: Simulace odesílání části rámce MAC blokem tx_test

3.3.2 Ověření příjmu rámců

Otestování funkce příjmu ethernetových paketů je realizováno pomocí zpracování přijímaných oktetů z modulu MAC, porovnáním relevantních částí příchozího paketu a případným zobrazením oktetů obsažených v datové části UDP datagramu v přijatém MAC rámci na sedmisegmentovém displeji přípravku. Stav bloku mac_rx a rx_test je zobrazován pomocí LED na přípravku (jednotlivé pozice jsou popsány v kapitole 3.3.2 popisující blok led_switch).

MAC rámec, na který reaguje blok mac_rx odesláním dat pro zobrazení sedmisegmentovým displeje, má následující obsah (není zde zobrazena výplň na minimální velikost rámce a pole FCS):



Obrázek 3.28: Obsah přijímaného MAC rámce

Na obrázku 3.28 jsou šedě označeny části rámce porovnávané při příchodu každého rámce. Tyto části musí být stejné s pevně nastavenou maskou uloženou v konstantách bloku mac_rx. Pokud se všechny tyto části s maskou shodují a rámec je přijat bez dalších chyb indikovaných modulem MAC, jsou Data (označena červeně) zobrazena na sedmisegmentovém displeji přípravku.

Hodnoty v testovaných polích přijímaného rámce musí být následující (lze je před syntézou projektu změnit v konstantách bloku mac_rx), aby příchod rámce způsobil odeslání datové části pro zobrazení na sedmisegmentovém displeji:

- **Cílová MAC adresa** : MAC adresa přípravku (12-34-56-78-9A-BC),
- **Typ/Délka**: Typ Internet Protocol verze 4 (0x0800h),
- **Verze IP**: IPv4 (0x4h),
- **Délka záhlaví**: pět 32-bitových slov (0x5h),
- **Protokol**: typ protokolu v datové části IP paketu - UDP (0x11h),
- **Cílová IP adresa**: IP adresa cílového zařízení (172.16.0.2 - 0xAC100002h),
- **Cílový UDP port**: hodnota cílového UDP portu (65000 - 0xFDE8h).

Blok rx_test

Tento blok slouží k příjmu oktetů rámce MAC, které jsou bloku poskytovány modulem MAC. Části každého příchozího rámce uvedené v kapitole 3.3.2 jsou otestovány a pokud se shodují s konstantami obsaženými v bloku test_rx (tyto konstanty je možné ho měnit pouze před syntézou projektu) je na sedmisegmentovém displeji zobrazena hexadecimální hodnota reprezentující data obsažená v příchozím rámci. Pokud se některé části liší je na displeji zobrazen znak E.

Pro správnou činnost bloku rx_test je nutné použít hodinový signál REF_CLK s frekvencí 50 MHz kvůli komunikaci s blokem MAC. Reset bloku je synchronní s REF_CLK a je v projektu generován modulem MMCM.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.43: Blok rx_test - hodinové a resetovací porty

Činnost bloku je ovlivněna použitou datovou rychlostí ethernetového rozhraní, poskytovanou bloku modulem MIIM pomocí portu SPEED_100M. V případě použití rychlosti rozhraní 10 Mbit/s je zpomalena frekvence přechodu mezi stavy FSM bloku na desetinu vnitřním signálem enable.

Port	Směr	Šířka	Popis
SPEED_100M	vstupní	1	Indikace rychlosti rozhraní

Tabulka 3.44: Blok rx_test - komunikace s modulem MIIM

S modulem MAC komunikuje blok pomocí portů uvedených v tabulce 3.45. Jednotlivé oktety přijímaného rámce jsou odesílané modulem MAC na port RX_DATA a jejich platnost je indikována na portu RX_DATA_VALID. Modul MAC dále bloku mac_rx poskytuje informaci o dokončení příjmu rámce na portu RX_COMPLETE a indikaci chybného pole FCS přijatého rámce na portu RX_FCS_ERROR.

Port	Směr	Šířka	Popis
RX_DATA	vstupní	8	Přijímané oktety z MAC
RX_DATA_VALID	vstupní	1	Platnost dat na RX_DATA
RX_COMPLETE	vstupní	1	Indikace dokončení příjmu
RX_FCS_ERROR	vstupní	1	Indikace chyby FCS

Tabulka 3.45: Blok rx_test - komunikace s modulem MAC

Pro zobrazení činnosti bloku jsou použity porty uvedené v tabulce 3.46. Signál PACKET_MATCH slouží k indikaci, které částí naposledy přijatého rámce se shodují s požadovanými hodnotami v konstantách bloku. Signál DISPLAY_DATA_RX odesílá zakódované znaky přijaté v datové části vyhovujících paketů do bloku display pro dekódování a zobrazení pomocí levé části sedmisegmentového displeje přípravku (DISP2) nebo informaci, že přijatý rámec má nevyhovující obsah.

Port	Směr	Šířka	Popis
PACKET_MATCH	výstupní	5	Indikace souhlasících polí
DISPLAY_DATA	výstupní	20	Data pro blok display

Tabulka 3.46: Blok rx_test - komunikace s periferiemi

Základním stavem FSM bloku je stav IDLE_st, který je výchozím stavem při resetu a dále se v něm blok nachází po dokončení příjmu rámce z modulu MAC. V tomto stavu je vyhodnocován stav portu RX_DATA_VALID, kterým je indikován příchod oktetů přijímaného rámce z modulu MAC. Pokud dojde k nastavení portu RX_DATA_VALID do logické '1' je uložena hodnota prvního příchozího oktetu rámce do vnitřního signálu RcvdDestMACAdd, který je určen pro uložení cílové MAC adresy příchozího rámce.

Přechod mezi stavy je řízen pomocí signálu cnt, který slouží jako čítač příchozích oktetů, tento signál je inkrementován každé 3 takty hodinového signálu, kdy je FSM aktivní (80 nebo 800 ns). K této inkrementaci je využit signál DibCnt, který každé 3 takty inkrementuje signál cnt. Tento způsob inkrementace je použit ve všech stavech FSM, kde je potřeba ukládat přicházející oktety rámce.

Ze stavu IDLE_st přechází FSM do stavu SAVE_MAC_st s příchodem druhého oktetu z MAC (tedy při inkrementaci signálu cnt). V tomto stavu je uložen zbytek MAC adresy do vnitřního signálu RcvdDestMACAdd a po něm přechází FSM do stavu SAVE_HEADER_st.

Ve stavu SAVE_HEADER_st jsou uloženy tři oktety ve kterých je uloženo pole Typ/Délka a pole obsahující verzi IP a délku IP záhlaví (pokud se jedná o rámec obsahující IPv4 paket). Dále následují stavy SAVE_PROTOCOL_st, SAVE_IP_st a SAVE_PORT_st, kde jsou uloženy oktety, kde se v UDP datagramu zapouzdřeného v IP paketu nachází typ protokolu uvnitř datové části IP paketu, cílová IP adresa a cílový UDP port.

Ze stavu SAVE_PORT přechází FSM do stavu CHECK_MATCH_st, kde jsou porovnány hodnoty částí rámce, které byly uloženy v předchozích stavech. Pokud se některá z testovaných částí liší od požadované hodnoty definované ve výše zmíněné masce, přejde FSM bloku do stavu NO_MATCH_st, kde zůstává do doby něž je dokončen příjem rámce. Zároveň jsou nastaveny jednotlivé bity portu PACKET_MATCH, podle testovaných částí rámce, které vyhovují masce uložené v konstantách bloku. Od LSB se jedná o shodu v části cílové MAC adresy, části IP hlavičky, zapouzdřeného protokolu UDP, cílové IP adresy a cílového UDP portu. Pokud se všechny uložené oktety zpracovávaného rámce rovnají příslušným hodnotám konstant a došlo k příchodu požadovaného

rámce, který obsahuje využívanou datovou část, přechází FSM bloku do stavu SAVE_DATA_st.

V něm jsou uloženy první dva bajty datové části UDP datagramu do vnitřního signálu UdpData a je zde vyčkáno na dokončení příjmu rámce. Pokud je rámec v pořádku a není v něm detekována chyba FCS jsou uložené bajty datové části odeslány v hexadecimálním tvaru bloku displej portem DISPLAY_DATA_RX, který je zobrazí na levé části sedmisegmentového displeje. Pokud je detekována chyba FCS je místo hodnot přijatých bajtů odeslána hodnota, která na displeji zobrazí hodnotu FCs. Po odeslání některé z těchto hodnot se FSM bloku vrací zpět do stavu IDLE_st, kde čeká na příchod dalšího MAC rámce.

Blok display

Blok displej slouží k převodu hodnot z bloků tx_test a rx_test pro zobrazení na sedmisegmentovém displeji přípravku. Displej je ovládán adresací jednotlivých pozic a nastavením aktivních segmentů vybrané pozice. Displej se skládá ze dvou fyzicky oddělených displejů označených na přípravku jako DISP1 a DISP2, které jsou ale zapojeny jako jeden displej s osmi znakovými pozicemi.

Levý displej (DISP2) zobrazuje poslední odeslaný stav přepínačů blokem tx_test, případně vnik kolize (CoL) nebo ukončení odesílání kvůli vzniku patnácti kolizí (Err). Pravý displej zobrazuje poslední přijatou dvou bajtovou hodnotu v korektně přijatém MAC rámci se správnými hodnotami jednotlivých části popsaných v kapitole 3.3.2, případně znak E při příjmu rámce, který se liší v některé z testovaných částí (MAC adresa, IP adresa, protokol, ...).

Pro správnou činnost bloku tx_test je nutné použít hodinový signál REF_CLK s frekvencí 50 MHz, ze kterého je odvozen vnitřní signál refresh pro obnovování jednotlivých pozic sedmisegmentového displeje.

Port	Směr	Šířka	Popis
REF_CLK	vstupní	1	Hodinový signál bloku
RESET	vstupní	1	Globální synchronní reset

Tabulka 3.47: Blok tx_test - hodinové a resetovací porty

Displej přípravku má vyvedeno osm společných anod LED tvořících každou pozici displeje a katody všech těchto pozic jsou společné. Na displeji tak lze najednou zobrazit pouze jeden znak, který je zobrazen na pozicích adresovaných pomocí jejich společné anody. Signál pro společné anody a katody je tedy nutné při současném zobrazení různých znaků na různých pozicích displeje řídit tak rychle , aby přepínání jednotlivých pozic nebylo lidským okem postřehnutelné. Každá pozice displeje bude svítit pouze 1/8 času, ale z důvodu setrvačnosti lidského oka bude při dostatečně rychlém přepínání pozic vypadat jako trvale rozsvícená. Pro zobrazení všech osmi znaků byla zvolena frekvence přepínání pozic 800 Hz. Obnovovací frekvence, dostatečná pro kvalitní vjem zobrazení znaků na celém displeji, tedy bude 100 Hz.

Tento signál přepínající aktivní pozici displeje (refresh) s frekvencí 800 Hz je ze vstupního hodinového signálu vytvořen pomocí čítače do hodnoty 62499, který generuje impulz každých 62500 taktů hodinového signálu REF_CLK (tedy požadovaných 1,25 ms při použití hodinového signálu REF_CLK s frekvencí 50 MHz).

Blok přijímá z bloků mac_tx a mac_rx hodnoty pro zobrazení na jednotlivých pozicích displeje pomocí portů DISPLAY_DATA_TX a DISPLAY_DATA_RX. Každý znak je zakódován pěti bitovou hodnotou, celkem je tedy každým blokem generován signál o datové šířce dvacet bitů. Tento signál je blokem displej rozdělen na čtvrtiny a každý dekódovaný znak (viz obrázek 3.29)je pomocí signálu SEGMENT zobrazen na příslušné aktivní pozici displeje řízené signálem DIGIT.

Port	Směr	Šířka	Popis
DISPLAY_DATA_TX	vstupní	20	Data z bloku mac_tx
DISPLAY_DATA_RX	vstupní	20	Data z bloku mac_rx

Tabulka 3.48: Blok display - komunikace s bloky tx_test a rx_test

Port	Směr	Šířka	Popis
SEGMENT	výstupní	8	Řízení aktivních segmentů
DIGIT	výstupní	8	Řízení aktivní pozice

Tabulka 3.49: Blok display - řízení displeje

```
1  -- dekodovani znaku pro sedmisegmentovy displej
2  with char select
3  segment <= "11000000" when "00000",   -- 0
4             "11111001" when "00001", -- 1
5             "10010010" when "00010", -- 2
6             "10110000" when "00011", -- 3
7             "10101001" when "00100", -- 4
8             "10100100" when "00101", -- 5
9             "10000100" when "00110", -- 6
10            "11110001" when "00111", -- 7
11            "10000000" when "01000", -- 8
12            "10100000" when "01001", -- 9
13            "10000001" when "01010", -- A
14            "10001100" when "01011", -- b
15            "11000110" when "01100", -- C
16            "10011000" when "01101", -- d
17            "10000110" when "01110", -- E
18            "10000111" when "01111", -- F
19            "11001110" when "10000", -- L
20            "10011111" when "10001", -- r
21            "11111111" when "10010", -- nic
22            "10011100" when "10011", -- o
23            "10111111" when "10100", -- pomlcka
24            "10100100" when "10101", -- S
25            "00000110" when others;  -- E.
```

Obrázek 3.29: Blok display - dekódování znaků

Blok led_switch

Blok led_sw je sběrnicový dvoukanálový multiplexor pro výběr jednoho zde dvou signálů na portu LED_1in nebo LED2_in pro zobrazení pomocí LED připojených k výstupnímu portu LED_out.

Port	Směr	Šířka	Popis
LED1_in	vstupní	16	Vstupní signál č.1
LED2_in	vstupní	16	Vstupní signál č.2
SWITCH	vstupní	1	Výběr signálu
LED_out	výstupní	16	Výstup multiplexoru

Tabulka 3.50: Blok led_switch - porty

Funkce je patrná z VHDL kódu zobrazeného na obrátku 3.30.

```
1 LED_out <= LED1_in when SWITCH = '0' else LED2_in;
```

Obrázek 3.30: Blok led_switch - funkce bloku

K portu LED1_in jsou připojeny následující signály (od MSB). Tento port je zvolen pokud je port SWITCH je v logické '0' (spínač BTNU není stisknut).

- LINK_STATUS - stav rozhraní Ethernet ('1' = aktivní)
- SPEED_100M - rychlost rozhraní Ethernet ('1' = 100 Mbit/s)
- FULL_DUPLEX -duplex rozhraní Ethernet ('1' = plný duplex)
- L - logická '0'
- L - logická '0'
- L - logická '0'
- STATE_TX(5:0) - stav FSM bloku mac_tx
- L - logická '0
- TX_ERROR - indikace nezdařeného odeslání vlivem opakovaných kolizí
- TX_RETRY - indikace kolize na m0diu modulem MAC
- TX_COMPLETE - indikace dokončení odesílání rámce

K portu LED2_in jsou připojeny následující signály (od MSB). Tento port je zvolen pokud je port SWITCH je v logické '1' (spínač BTNU je stisknut).

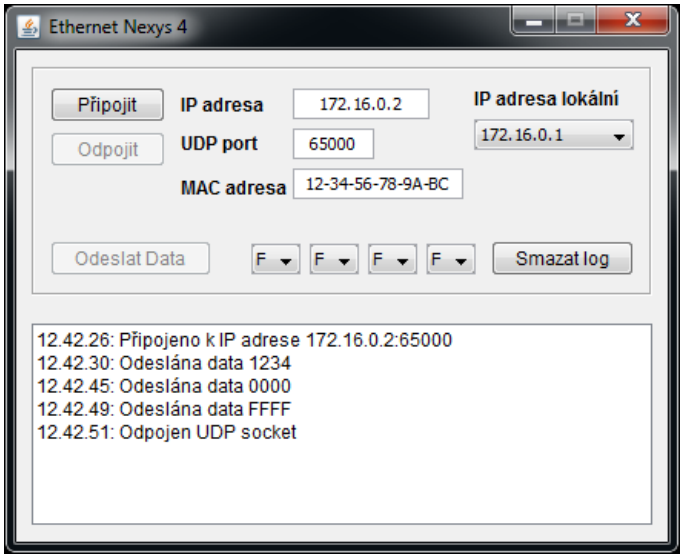
- L - logická '0'
- L - logická '0'
- L - logická '0'
- L - logická '0'
- L - logická '0'
- L - logická '0'
- PACKET_MATCH(4:0) - stav shody rámce přijatého blokem mac_rx
- L - logická '0
- RX_PREAMBLE_ERROR - indikace chyby preambule
- RX_LENGTH_ERROR - indikac chybné délky
- RX_FCS_ERROR - indikace špatného FCS
- RX_COMPLETE - indikace dokončení příjmu rámce

3.3.3 Program pro odesílání rámců

V rámci ověření příjmu rámců pomocí vytvořeného modulu byl v programovacím jazyce Java, s vyžitím rozhraní Swing, vytvořen pomocný program s názvem MAC_Tester k odesílání Ethernet rámců s obsahem uvedeným v kapitole 3.3.1. Z tohoto souboru s příponou jar byl pomocí programu Lauch4j vytvořen spustitelný soubor s příponou exe. Hlavní okno tohoto programu je uvedeno na obrázku 3.31.

Program byl vyzkoušen pouze na operačním systému Windows 7, ale měl by být bez problému spustitelný na jiných operačních systémech Windows podporujících JRE (Java Runtime Enviroment) verze 1.8 a vyšší. Pro správnou funkci programu je nutné při spuštění souboru potvrdit souhlas se zvýšením oprávnění tohoto programu z důvodu vytváření záznamu v ARP tabulce příkazem arp -s, pro jehož spuštění jsou nutná zvýšená oprávnění.

Dále je možné spustit přímo soubor s příponou jar, ten je ale nutné spouštět z příkazové řádky jako správce. Tato varianta je zde například pro spuštění programu v operačním systému typu UNIX.



Obrázek 3.31: Program MAC Tester - okno programu

Program při stisknutí tlačítka „Připojit“ vytvoří UDP soket na zvoleném portu a vytvoří záznam v ARP tabulce se zadanou MAC a IP adresou na zvoleném rozhraní s danou lokální IP adresou. Pokud tyto činnosti proběhnou úspěšně je v dolní části okna programu zobrazena zpráva o připojení k dané IP adrese a UDP portu.

Po připojení je možné stisknutím tlačítka „Odeslat Data“ odeslat na zadanou IP adresu UDP paket se zadaným cílovým UDP portem. Do datové části tohoto UDP datagramu jsou vložena data, jejíž hexadecimální reprezentace je nastavena vlevo od tlačítka „Odeslat Data“. Pokud se zdaří odeslání datagramu je v dolní části okna programu zobrazena zpráva o odeslání dat.

Tlačítko „Odpojit“, které je aktivní pouze pokud je program připojen k UDP soketu, složí k jeho odpojení, ke kterému dojde i při zavření programu. Tlačítko „Smazat log“ je určeno k vymazání zpráv zobrazených v dolní části okna programu.

Funkce programu

Po připojení přípravku k počítači pomocí ethernetového rozhraní je možné programem odesílat IP pakety, které připojený přípravek příjme a pokud se hodnoty použité MAC adresy, IP adresy a cílového portu odesílaných rámců rovnají nastavení masky zvolené v bloku mac_rx při syntéze projektu nahraného do FPGA přípravku je možné nastavením odesílaných dat v programu řídit levou část displeje přípravku a odesílaná data na něm zobrazovat.

3.4 Zhodnocení a budoucí práce

V rámci této diplomové práce jsem v programovacím jazyce VHDL vytvořil modul pro ovládání rozhraní Ethernet přípravku Nexys 4 a demonstrační projekt pro využití tohoto modulu s přípravkem Nexys 4 a program pro odesílání UDP datagramů z počítače do připojeného přípravku. Výhoda této varianty řešení pomocí vlastního VHDL modulu je, že realizovaný modul je přenositelný (s drobnými úpravami danými použitím primitiv dostupných pouze v některých FPGA) do jiných FPGA obvodů a nevyžaduje použití žádných dalších IP cores a soft-procesorů. Je možné ho tedy využít i pro ovládání ethernetových rozhraní staršími typy FPGA obvodů (např. Spartan-3) a bylo by ho možné využít i s jiným PHY transceiverem s RMII rozhraním.

V modulu je implementováno rozhraní RMII pro komunikaci s PHY obvodem LAN8720A. Vytvořený modul umožňuje číst data z PHY registrů pomocí MIIM rozhraní a data do těchto registrů zapisovat. V demonstračním projektu není funkcionalita pro čtení a zápis do libovolného registru využita (je použito pouze pro zápis a čtení pevně zvolených registrů pro realizaci SW resetu a čtení stavu PHY), ale byla vyzkoušena a je plně funkční.

Modulem MAC bylo implementováno ovládání RMII rozhraní pro odesílání a příjem Ethernet paketů, který byl využit v demonstračním projektu uvedeném v kapitole 3.3. Modul může fungovat v 100 Mbit/s nebo 10 Mbit/s režimu s použitím plného nebo poloviční duplexu. V zadání bylo určeno zaměřit se na 100 Mbit/s režim, ale rozhodl jsem se vytvořit variantu se všemi možnými typy spojení podporovanými použitým PHY transceiverem. Modul realizuje vrstvu MAC pro vytváření Ethernet paketu a jeho odeslání do PHY. Blok přijímá oktety od klienta MAC, připojuje k nim FCS, výplň pro dodržení minimální velikosti rámce, preambuli a SFD a takto vytvořený ethernetový paket odesílá RMII rozhraním do PHY. Modul také realizuje CSMA/CD algoritmus pro řešení přístupu k médiu v případě použití poloviční duplexu. Modul přijímá z PHY datový tok na RMII rozhraní, ze kterého obnovuje jednotlivé oktety přijímaného rámce a odesílá je klientovi MAC. U přijímaných paketů dále odstraňuje preambuli paketu, SFD, pole FCS a provádí filtrování rámců určených pro jinou MAC adresu, než je adresa pro broadcast a pevně nastavená MAC adresa. Modul také klientovi MAC poskytuje informaci o detekci chyb u přijímaných rámců (nesouhlasící pole Délka, neplatná velikost rámce, chybné pole FCS a chybná preambule rámce).

Využití modulu MAC je zatím pouze omezené. Pro odesílání a příjem MAC rámců je nutné předem znát MAC a IP adresu cílové stanice. Možným vylepšením modulu by tedy mohla být implementace ARP (Address Resolution Protocol) protokolu pro odesílání a odpovídání na dotazy pro propojení MAC a IP adresy cílové stanice. Bloky pro demonstraci využití rozhraní rx_test a tx_test jsou zjednodušené a neimplementují plně IPv4 a UDP protokol (např. zde není kontrola a generování kontrolních součtů jednotlivých částí paketu). Další práce by tedy mohla být zaměřena na implementaci plné podpory IPv4 protokolu a dalších do něj zapouzdřených protokolů jako UDP nebo ICMP (Internet Control Message Protocol).

Kapitola 4

Závěr

V této diplomové práci jsem navrhl a realizoval ovládání rozhraní Ethernet na přípravku Nexys 4, využívající rozhraní RMII pro odesílání a příjem MAC rámců pomocí vlastního modulu realizovaného v jazyce VHDL a vytvořil jsem demonstrační projekt pro ověření funkce tohoto modulu.

Vytvořený modul zajišťuje funkci podvrstvy MAC pro odesílání a příjem rámců RMII rozhraním a management PHY pomocí rozhraní MIIM. Modul dokáže přijímat rámce typu Ethernet II (rámce obsahující pole Typ) i rámce typu LLC (rámce obsahující pole Délka). Modul je možné využít pro příjem a odesílání dat pomocí rozhraní Ethernet, jak bylo vyzkoušeno v demonstračním projektu uvedeném v praktické části práce. Tento modul by bylo možné využít i s jinými hradlovými poli a podobnými PHY obvody a jeho použití nevyžaduje využití přídavných IP cores.

Dále jsem vytvořil testovací program pro odesílání UDP datagramů z PC do připojeného přípravku, kterým byla ověřena funkčnost přijímací části modulu. Odesílací část byla testována pomocí programu Wireshark, kterým byly odchytávány pakety odeslané realizovaným modulem z přípravku Nexys 4. Odesílání a příjem rámců vytvořeným modulem během testování nevykazovalo žádné odchylky od standardu Ethernet.

Zadání diplomové práce se mi tak podařilo zcela a úspěšně splnit. Realizoval jsem nejen 100 Mbit/s variantu rozhraní, ale i pomalejší 10 Mbit/s variantu. Modul také umožňuje funkci při poloviční i plném duplexu. Všechny zdrojové soubory pro programování hradlového pole a program MAC_Tester pro odesílání UDP datagramu jsou součástí elektronických příloh této práce.

Díky této práci jsem se blíže seznámil s rozhraním Ethernet a jeho standardem IEEE 802.3 pro varianty 10BASE-T a 100BASE-T, rozhraním MII a RMII a s funkcionalitou PHY transceiveru Microchip LAN8720A. Dokázal jsem naprogramovat základní ovládání ethernetového rozhraní, které by v budoucnu mohlo být dále rozšiřováno přidáním například navazujících protokolů vyšších vrstev komunikujících s podvrstvou MAC, jak jsem zmínil v předchozí kapitole.

Literatura

[1] IEEE 802.3-2015. *IEEE Standard for Ethernet: Section One*. Revision of IEEE Std 802.3-2012. New York, NY: IEEE Standards Association, 2015.

[2] IEEE 802.3-2015. *IEEE Standard for Ethernet: Section Two*. Revision of IEEE Std 802.3-2012. New York, NY: IEEE Standards Association, 2015.

[3] BOHÁČ, Leoš a Pavel BEZPALEC. *Datové sítě: Přednášky*. Praha: České vysoké učení technické v Praze, 2011. ISBN 978-80-01-04694-4.

[4] RMII^{TM} Specification. Rev. 1.2. RMII Consortium, 1998.

[5] ASHENDEN, Peter J. *Digital design: an embedded systems approach using VHDL*. Boston: Elsevier/Morgan Kaufmann Publishers, c2008. ISBN 01-236-9528-7.

[6] Nexys 4^{TM} FPGA Board Reference Manual. *Digilent: Digilent Documentation* [online]. Pullman, WA: Digilent, 2016 [cit. 2017-04-10]. Dostupné z: `https://reference.digilentinc.com/_media/nexys:nexys4:nexys4_rm.pdf`

[7] LAN8720A. Microchip Technology Inc. [online]. Chandler, AZ: Microchip, 2016 [cit. 2017-03-15]. Dostupné z: `http://ww1.microchip.com/downloads/en/DeviceDoc/00002165B.pdf`

[8] Nexys 4 - Getting Started with Microblaze Servers. *Digilent: Digilent Documentation* [online]. Pullman, WA: Digilent, 2014 [cit. 2017-01-10]. Dostupné z: `https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-4-getting-started-with-microblaze-servers/start`

[9] LogiCORE IP Tri-Mode Ethernet MAC v5.4: Product Guide. *Xilinx - All Programmable* [online]. San Jose, California: Xilinx, 2012 [cit. 2017-02-15]. Dostupné z: `https://www.xilinx.com/support/documentation/ip_documentation/tri_mode_eth_mac/v5_4/pg051-tri-mode-eth-mac.pdf`

[10] MII to RMII v2.0: LogiCORE IP Product Guide. *Xilinx - All Programmable* [online]. San Jose, California: Xilinx, 2015 [cit. 2017-02-15]. Dostupné z: www.xilinx.com/support/documentation/ip_documentation/mii_to_rmii/v2_0/pg146-mii-to-rmii.pdf

[11] 7 Series FPGAs Clocking Resources User Guide: UG472 (v1.13) March 1, 2017. *Xilinx - All Programmable* [online]. San Jose, California: Xilinx, 2017 [cit. 2017-03-03]. Dostupné z: https://www.xilinx.com/support/documentation/user_guides/ug472_7Series_Clocking.pdf

[12] 7 Series FPGAs SelectIO Resources User Guide: UG471 (v1.8) September 27, 2016. *Xilinx - All Programable* [online]. San Jose, California: Xilinx, 2016 [cit. 2017-03-03]. Dostupné z: https://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf

[13] TAVINOV, Evgeni. A Practical Parallel CRC Generation Method. *Circuit Cellar* [online]. 2010, January 2010(Issue 234), 38-45 [cit. 2017-05-16]. Dostupné z: http://outputlogic.com/my-stuff/parallel_crc_generator_whitepaper.pdf

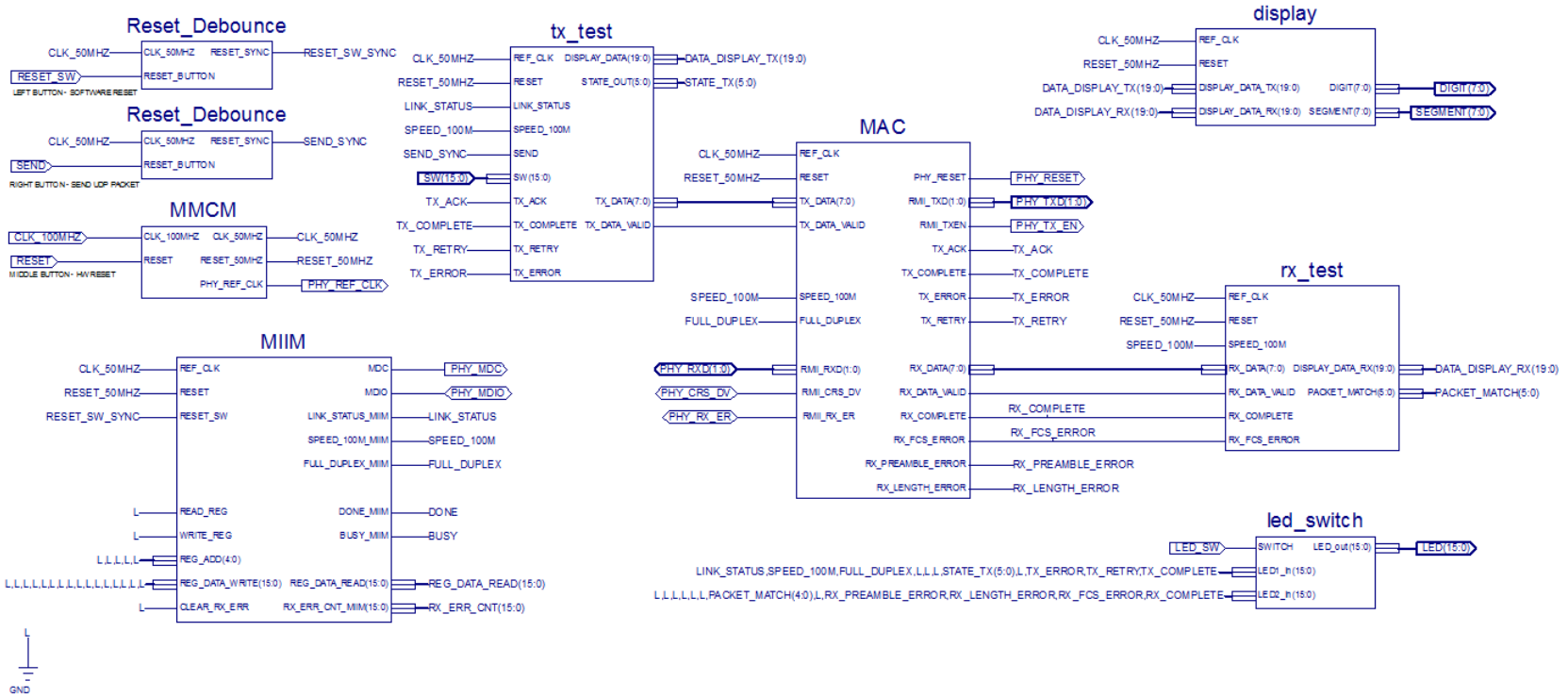
Seznam zkratek

AUI	Attachment Unit Interface
CRC	Cyclic Redundancy Check
CRS	Carrier Sense
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DDR	Double Data Rate
DRAM	Dynamic Random Access Memory
DSCP	Differentiated Services Code Point
ECT	Explicit Congestion Notification-Capable Transport
EDK	Embedded Development Kit
EEE	Energy-Efficient Ethernet
FCS	Frame Check Sequance
FPGA	Field Programable Gate Array
GMII	Gigabit Media Independent Interface
IEEE	Institute of Electrical and Electronics Engineers
IOB	Input/Output Block
IP	Intelectual Property (IP core), Internet Protocol
ISO	International Organization for Standardization
LAN	Local Area Network
LED	Light Emmiting Diode
LLC	Logical Link Control
LSB	Least Significant Bit
LFSR	Linear-Feedback Shift Register
MAC	Media Access Control
MAU	Medium Attachement Unit
MDI	Medium Dependent Interface
MII	Medium Independent Interface
MSB	Most Significant Bit
ODDR	Output DDR register
OSI	Open Systems Interconnection model
PHY	Physical Layer

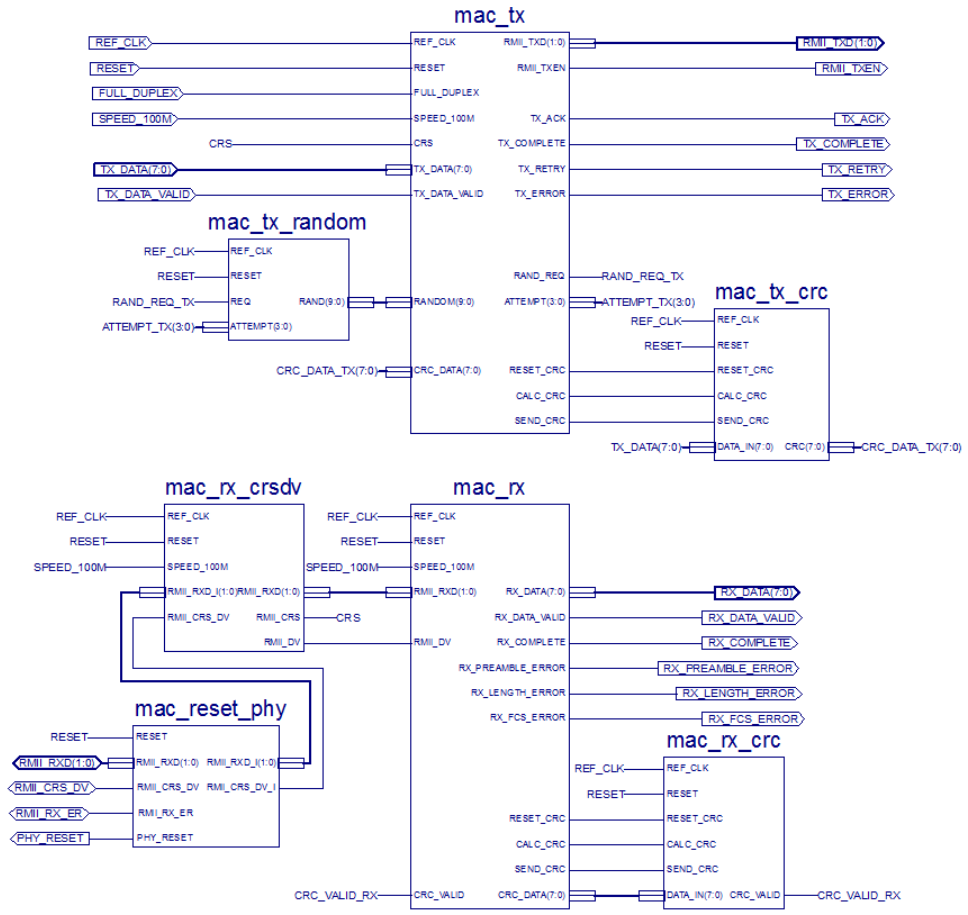
PLS	Physical Layer Signallng
PMA	Physical Medium Attachment
RGMII	Reduced gigabit media-independent interface
RM	Reference model ISO/OSI
RMII	Reduced Medium Independent Interface
SFD	Start of Frame Delimiter
SGMII	Serial gigabit media-independent interface
SMI	Seriál Management Interface
STA	Station Management
TOS	Type of Service
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
WAN	Wide Area Network
XGMII	10 Gigabit Media Independent Interface

Příloha A

Bloková schémata



Obrázek A.1: Blokové schéma projektu ethernet_top



Obrázek A.2: Blokové schéma modulu MAC