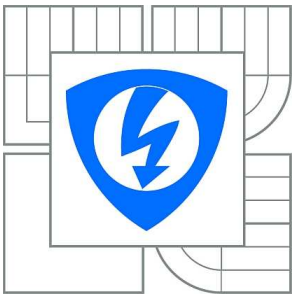
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF MICROELECTRONICS

IMPLEMENTACE RYCHLÝCH SÉRIOVÝCH SBĚRNIC V OBVODECH FPGA

IMPLEMENTATION OF FAST SERIAL BUS ON FPGA

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

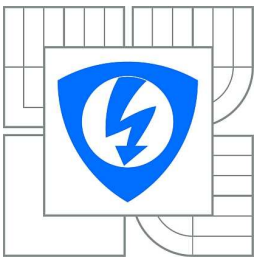
AUTOR PRÁCE — Bc. JAKUB DRBAL

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARIÁN PRISTACH

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav mikroelektroniky

Diplomová práce

magisterský navazující studijní obor
Mikroelektronika

Student:	Bc. Jakub Drbal	***ID:***	125405
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Implementace rychlých sériových sběrnic v obvodech FPGA

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi implementace rychlých sériových sběrnic v obvodech FPGA. Prostudujte signálové standardy používané v diferenčních linkách u komerčně používaných zařízení. Na základě získaných informací navrhněte obvod pro rychlý sériový přenos dat. Obvod bude popsán v jazyce VHDL a optimalizován pro implementaci do cílových obvodů FPGA Spartan-3 a Spartan-6 od firmy Xilinx. Navrhněte aplikaci demonstrující funkci obvodu, která bude provádět přenos dat mezi dvěma zařízeními.
V druhé části diplomové práce se zaměřte na návrh SATA kontroléru, který bude umožňovat přímé připojení pevného disku. Kontrolér popište v jazyce VHDL a implementujte do obvodu FPGA Virtex-5.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání:	10.2.2014	***Termín odevzdání:***	29.5.2014

Vedoucí práce: Ing. Marián Pristach
Konzultanti diplomové práce:

prof. Ing. Vladislav Musil, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce se zabývá implementací rychlé sériové sběrnice a SATA kontroléru do obvodu FPGA. Je rozdělena do dvou částí. V první části je navržen sériový vysílač pro komunikaci mezi obvody FPGA a v druhé je navržen kontrolér pro přímé připojení SATA pevného disku k obvodu FPGA. Sériový vysílač pro komunikaci mezi obvody FPGA je navržen podle SATA specifikace. Linková a fyzická vrstva je popsána v jazyce VHDL a implementována do programovatelné logiky.

ABSTRACT

This diploma thesis deals with implementation of fast serial bus and SATA controler in the FPGA chip. The work is divided into two parts. In the first part the circuit for communication between the FPGAs is designed and in the second part the circuit for direct connection of SATA hard disk to a gate array is created. The circuit for communication between the FPGA is designed according to SATA specification. Link layer and physical layers are implemented in VHDL with programmable logic resources.

KLÍČOVÁ SLOVA

Sériová sběrnice, diferenční signály, FPGA, VHDL, LVDS, SATA.

KEYWORDS

Serial bus, differential signals, FPGA, VHDL, LVDS, SATA.

BIBLIOGRAFICKÁ CITACE DÍLA

DRBAL, J. *Implementace rychlých sériových sběrnic v obvodech FPGA*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 50s. Vedoucí diplomové práce Ing. Marián Pristach.

PROHLÁŠENÍ AUTORA O PŮVODNOSTI DÍLA

Prohlašuji, že svoji diplomovou práci na téma Implementace rychlých sériových sběrnic v obvodech FPGA jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této semestrální práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Mariánu Pristachovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době je kladen požadavek na stále rychlejší přenos dat. Moderní rozhraní přecházejí z paralelních přenosů na sériové. Sériově se data přenáší téměř na všech vrstvách systému. Příkladem nových standardů pracujících se sériovým přenosem je SATA nebo PCI express. Sériový přenos bývá upřednostněn, protože umožňuje pracovat na stále se zvyšujících frekvencích. Maximální realizovatelná frekvence přenosu paralelních dat je okolo 2 GHz. U sériového přenosu je teoretická maximální frekvence někde okolo 30 GHz.

Cílem diplomové práce bylo prozkoumat možnosti implementace rychlých sériových linek v obvodech FPGA. Dále navrhnou obvod, který umožní jejich praktické využití a nakonec vytvořit kontrolér, který umožní přímé připojení SATA pevného disku k obvodu FPGA.

V teoretické části práce jsou popsány běžně používané diferenční signálové standardy a popsány možnosti implementace rychlých sériových linek do obvodů FPGA. Dále teoretická část obsahuje popis jednotlivých vrstev SATA kontroléru a příklady ATA příkazů.

V praktické části bylo navrženo rozhraní umožňující komunikaci mezi dvěma obvody FPGA. Rozhraní bylo uzpůsobeno pro levnější obvody, které neobsahují hardwarové prostředky pro rychlou sériovou komunikaci. Testování přenosu dat probíhalo mezi obvody Spartan-3E a Spartan-6. V druhé části byl navržen kontrolér pro přímé připojení pevného disku k obvodu FPGA. Protože nejnižší rychlost přenosu dat je 1,5 Gb/s, musel být využit gigabitový vysílač (GTP), který je obsažen pouze ve vyšších řadách obvodů FPGA. Pro implementaci byl zvolen obvod Virtex-5.

1 Teoretická část

1.1 Diferenční signály

Diferenční signály využívají dva vodiče mezi přijímačem a vysílačem. Obvykle označujeme jeden vodič jako pozitivní a druhý jako negativní. Oba signály mají stejnou hodnotu, ale opačnou polaritu. Logická úroveň signálů se vyhodnocuje podle rozdílu napětí. Pokud je rozdíl kladný má hodnotu log. 1 a pokud je záporný tak log. 0. Díky tomu se žádný signál nevrací přes zem obvodu a nedochází k rušení.

Diferenční signály mají jednu zřejmou nevýhodu a to, že potřebují dva vodiče na jejích přenos. Naproti tomu, ale mají mnoho výhod. Když se žádný signál nevrací skrz zem, je rušení na zemnicím vodiči relativné nepodstatné. Diferenční analogové signály jdoucí do digitálního obvodu není potřeba napěťově přizpůsobovat. Oddělení od napájení je díky tomuto mnohem jednodušší.

Diferenční obvody jsou velmi nápomocné v aplikacích, které využívají nízkonapěťové signály. Pokud je signál velmi malý, nebo je problém se šumem v obvodu, tak diferenční signály zvýší úroveň signálu na dvojnásobek. Často je toto využíváno v systémech pracujících s velmi malými signály.

Diferenční přijímače jsou navrženy tak, aby byli citlivé na rozdíl signálu a přitom necitlivé na jejich souhlasnou složku. Diky tomu jsou tyto signály využívány v systémech s vysokým šumem. Protože je u diferenčních signálů přesně definována rozhodovací úroveň je časování signálu mnohem přesnější. Více o diferenčních signálech v [1].

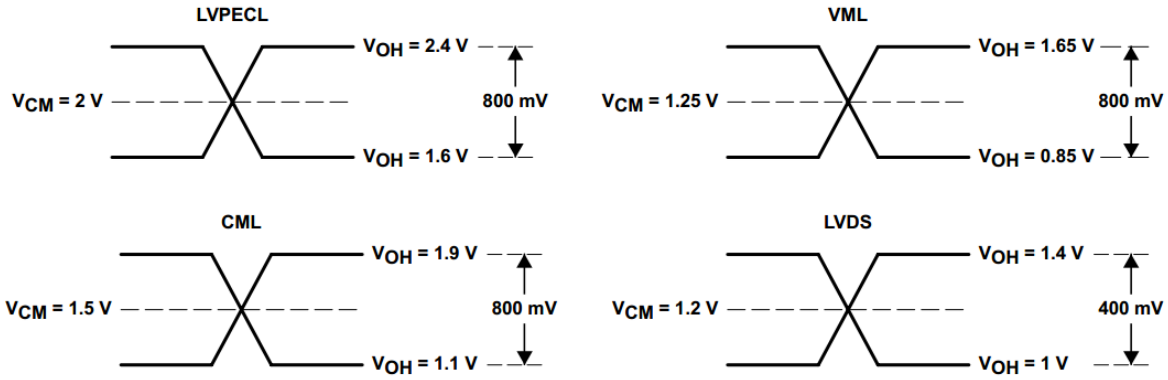
1.1.1 Diferenční standarty

V dnešních moderních telekomunikačních a datových systémech se využívají diferenční signály různých standardů. Mezi hlavní diferenční standarty se řadí low-voltage positive/pseudo emmiter-coupled-logic (LVPECL), current-mode logic (CML), voltage-mode logic (VML) a low-voltage diferential signaling (LVDS). V tab. 1 jsou napěťové parametry jednotlivých standardů.

Tab. 1 Napěťové parametry diferenčních standardů [2]

Parametr	LVPECL	CML	VML	LVDS
V_{OH}	2,4 V	1,9 V	1,65 V	1,4 V
V_{OL}	1,6 V	1,1 V	0,85 V	1 V
Výstupní napětí (diferenční)	800 mV	800 mV	800 mV	400 mV
Souhlasné napětí	2 V	1,5 V	1,25 V	1,2 V

Rozhodovací úroveň diferenčních standardů se nachází v oblasti souhlasného napětí. Grafické znázornění signálů jednotlivých standardů je na obr. 1.



Obr. 1 Grafické zobrazení napěťových úrovní jednotlivých standardů [2]

Pro zajištění všech předpokládaných výhod diferenčních signálů musí být dodrženo pár základních pravidel při návrhu plošného spoje. Více v [1].

1. Dráhy diferenčních signálů musí být stejně dlouhé
2. Dráhy diferenčních signály musí být vedeny těsně u sebe
3. Vzdálenost mezi drahami diferenčních signálů musí byt v celé délce konstantní

1.1.2 Diferenční standardy podporované obvody FPGA

Obvody FPGA od firmy Xilinx podporují mnoho diferenčních standardů. Přizpůsobení a nastavení vstupů/výstupů je popsáno v katalogových listech. Standardy podporované vybranými obvody jsou v tab. 2.

Tab. 2 Diferenční standardy podporované obvody firmy Xilinx

Obvod	Podporované diferenční standardy
Spartan 3 [3]	LDT (ULVDS), LVDS, LVPECL, RSDS, HSTL, SSTL
Spartan 3E [4]	LVDS, RSDS, MINI_LVDS, LVPECL, BLVDS, HSTL,SSTL
Spartan 3A [5]	LVDS, BLVDS, MINI_LVDS, LVPECL, RSDS, TMDS, PPDS, HSTL, SSTL
Spartan 6 [6]	LVDS, BLVDS, DISPLAY PORT, LVPECL, MINI_LVDS, RSDS, TMDS, PPDS, LPDDR, HSTL, SSTL
Virtex 4 [7]	LDT, LVDS, LVDSEXT, LVPECL, RSDS, ULVDS, BLVDS, HSTL, SSTL
Virtex 5 [8]	LDT, LVDS, LVDSEXT, LVPECL, RSDS, ULVDS, BLVDS, HSTL, SSTL, DISPLAY PORT, TMDS, PPDS, LPDDR

Nastavení diferenčních vstupů/výstupů se provádí přímo při konfiguraci obvodů. Je možné nastavit diferenční standart daného vstupu/výstupu a také jestli se má použít integrovaná terminace.

1.1.3 Možnosti implementace do obvodů FPGA

Obvody FPGA od firmy Xilinx mají integrovaný hardware pro podporu diferenčních standardů. Většina vstupů je možné nakonfigurovat jako diferenční, přičemž jeden vstup je pro pozitivní signál a vedlejší pro negativní signál.

Implementace závisí na modelu časování signálů. Tyto modely jsou tři možné. První je systémově synchronní, druhý zdrojově synchronní a třetí asynchronní.

Pro systémově synchronní model jsou data, která mají být vysílána a přijímána synchronizována pomocí hodinového signálu, který je společný pro přijímač i vysílač. Problém tohoto modelu nastává pokud má hodinový signál vstupující do více obvodů různá zpoždění. Může docházet ke špatnému časování a tím ztrátě dat.

U zdrojově synchronních signálů je společně s daty vysílán také hodinový signál. Pro tento model je důležité přesné synchronizování hodinového signálu s daty. Také zpoždění dat a hodinového signálu na straně přijímače by mělo být stejné.

Asynchronní signály využívají speciální kódování odesílaných dat. Data jsou kódována tak, aby bylo možné z těchto dat opět rekonstruovat hodinový signál. Hlavní výhodou tohoto modelu je, že není potřeba žádného dodatečného signálu k časování. Nevýhoda je potřeba speciálního hardwaru, který umožňuje zpětnou rekonstrukci hodin nebo přímo dat.

1.2 Příjem asynchronních dat

Pro příjem asynchronních dat je potřeba rekonstruovat hodinový signál. Běžně se uvádějí tři možnosti rekonstrukce hodin a nebo dat:

- metoda založené na PLL
- optická metoda
- metoda převzorkování.

Metoda založená na PLL využívá fázového závěsu, který pomocí fázového detektoru vzorkuje datový signál. Tato metoda je závislá na způsobu detekce. Nevýhoda této metody spočívá v potřebě použití nastavitelného oscilátoru.

Optická metoda je variace na metodu využívající PLL, ale použitá přímo na optický datový proud. Může pracovat na velmi vysokých rychlostech, ale pouze v přenosech s optickými vlákny.

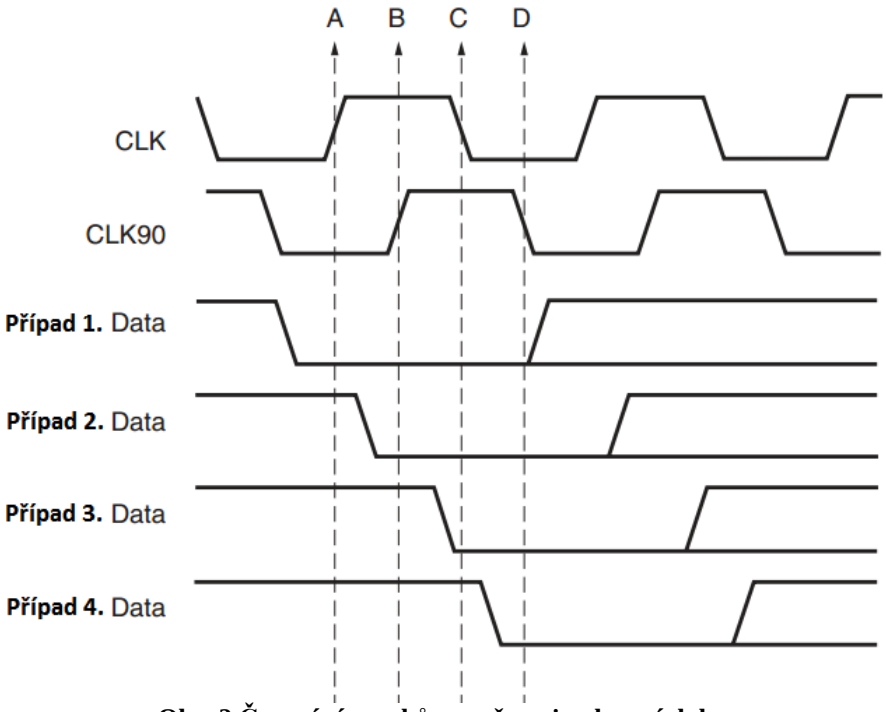
Metoda převzorkování využívá vyšší frekvence pro vzorkování příchozího datového signálu na několika místech. Tato metoda není schopná obnovit hodinový signál, ale obnoví data. Obvykle se používají 4 až 5 vzorků na periodu signálu pro vysokorychlostní datové přenosy a 3 vzorky u gigabitových přenosů. Převzorkování je jednoduchá, robustní metoda často využívána u programovatelných obvodů, které nemají žádný hardware pro metodu PLL a je potřeba plně digitální design.

Aby byla možná zpětná rekonstrukce dat, musí se data kódovat speciálním kódem. Hlavním důvodem je dostatek změn ve výstupním datovém proudu, ze kterých je možné zpětně zrekonstruovat hodinový signál. Dále jsou časté změny v signálu potřeba, aby se předešlo neurčitosti v dlouhých sekvencích jedniček nebo nul, kdy není jisté kolik dat mělo být přijato. Kódování se využívá také proto, aby stejnosměrná složka vysílaného signálu zůstala konstantní. Díky tomu není potřeba stejnosměrné oddělení vysílače od přijímače. Např. kód 8b/10b řeší obě výše zmíněné podmínky. Bohužel kód 8b/10b způsobuje redundanci dat o 20%. Toto řeší nově využívané kódy 64b/66b a 128b/130b kde je redundance dat mnohem menší, ale již nezachovávají konstantní stejnosměrnou složku. V důsledku toho je nutné při použití těchto kódů oddělovat stejnosměrnou složku signálu.

1.2.1 Metoda převzorkování

Zkoumaná metoda využívá 4 vzorky na jednu periodu dat. Je využito hodinového signálu běžícího na stejné frekvenci jako data. Ve skutečnosti jsou využity dva hodinové signály, které běží na stejné frekvenci, ale jsou vzájemně posunuty o 90 stupňů. Data jsou vzorkovány na náběžné i sestupné hrany hodinových signálů. Dostáváme tedy 4 vzorkovací signály, které jsou vzájemně posunuty o 90 stupňů.

Hlavním důvodem převzorkování je nalezení hran datového signálu a následného vyhodnocení, který ze čtyř vzorků bude použit pro rekonstrukci dat. Jednotlivé vzorky se nacházejí ve čtyřech časových doménách A, B, C a D.

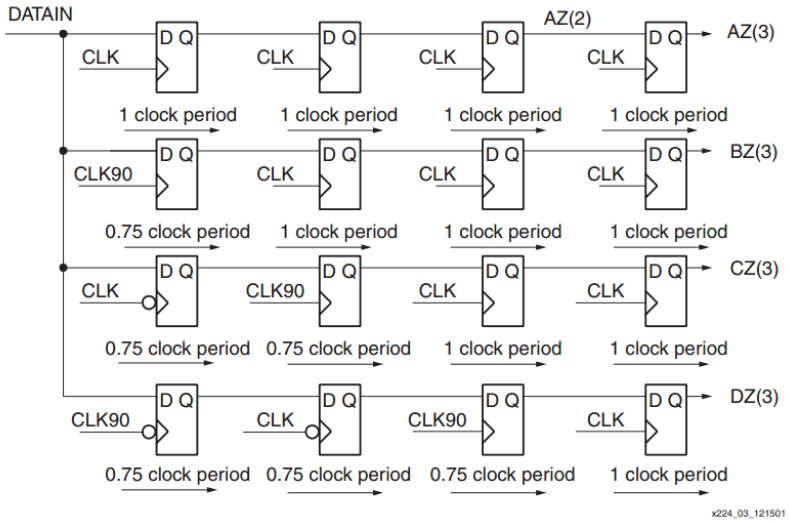


Obr. 2 Časování vzorků a možnosti nalezených hran

Na obr. 2 je zobrazeno časování jednotlivých vzorků datového signálu a jejich možných umístění v datech. Jsou možné čtyři případy:

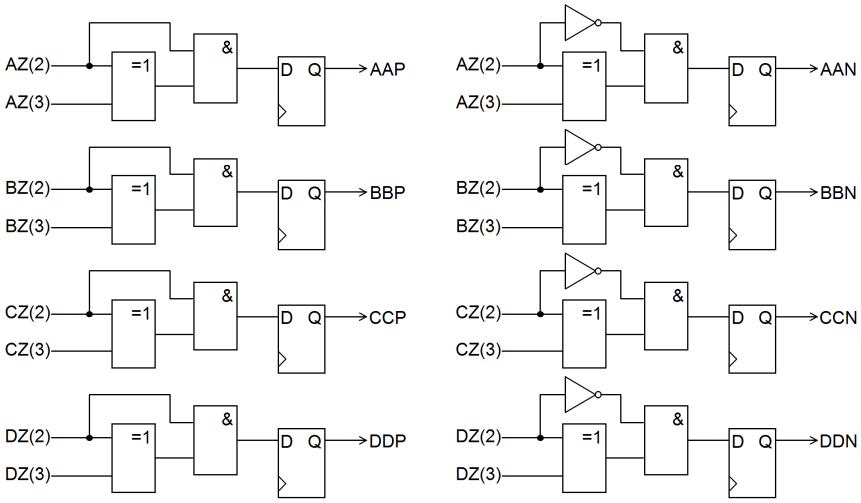
1. Všechny čtyři vzorky vzorkují ta samá data. Začátek dat je nalezen v doméně A, tedy pro rekonstrukci se použijí data z domény C.
2. Minulá data byla v doméně A a aktuální data jsou v ostatních časových doménách. Začátek dat je nalezen v doméně B, tedy pro rekonstrukci jsou použity data z domény D.
3. Aktuální data jsou v doménách A a B. Budoucí data jsou vzorkovány časovými doménami C a D. Začátek dat je nalezen v doméně C, tedy pro rekonstrukci dat jsou použity data z domény A.
4. Aktuální data jsou v časových doménách A, B a C. Budoucí data v doméně D. Začátek dat je tedy v doméně D, pro rekonstrukci jsou použity data z domény B.

Jak je možné si všimnou, doména ze které jsou načítána data, je posunutá o dvě domény od té, která první nalezla aktuální data. Toto je zvoleno z důvodu, aby se zamezilo chybě načítaných dat v důsledku šumu na datové lince.



Obr. 3 Vstupní obvod pro vzorkování a převod do jedné časové domény

Vstupní obvod (na Obr. 3) využívá čtyři klopné obvody reagující na různé vzorkovací hodinové signály. Následně jsou použity další klopné obvody sloužící k přenosu navzorkovaných signálů do jedné časové domény. Pro vyhodnocování hran přijímaného signálu se využívají vzorky z třetího a čtvrtého stupně vstupního obvodu. Tyto vzorky vstupují do vyhodnocovacího obvodu.



Obr. 4 Vyhodnocení hran datového signálu

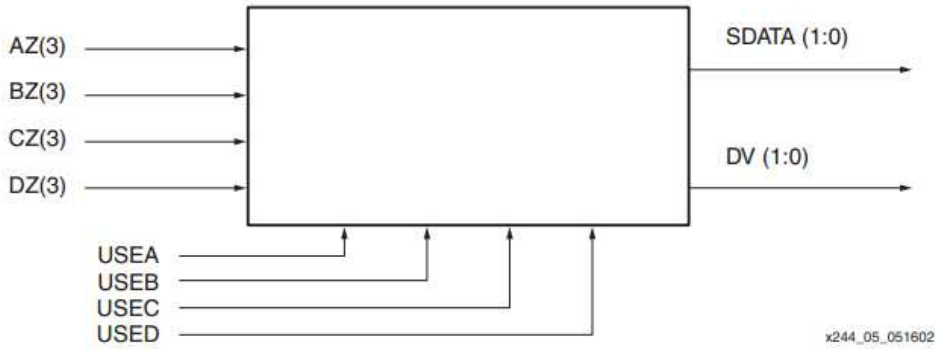
Ve vyhodnocovací části (obr. 4) je zjišťováno jestli v dané časové doméně došlo k nalezení hrany datového signálu a také jestli je tato hrana náběžná nebo sestupná. Na základě výstupu z tohoto obvodu je zvolena časová doména ze které budou přijímaná data.

1. Z domény A jsou data přijímána pokud signály AAP = BBP = 1 a zároveň CCP = DDP = 0, nebo AAN = BBN = 1 a zároveň CCN = DDN = 0.
2. Z domény B jsou data přijímána pokud signály AAP = BBP = CCP = 1 a zároveň CCP = 0, nebo AAN = BBN = CCN = 1 a zároveň CCN = 0.
3. Z domény C jsou data přijímána pokud signály AAP = BBP = CCP = DDP =1, nebo AAN = BBN = CCN = DDN =1.
4. Z domény D jsou data přijímána pokud AAP = 1 a zároveň BBP = CCP = DDP = 0, nebo AAN = 1 a zároveň BBN = CCN = DDN = 0.

Pokud není v aktuální taktu nalezena žádná hrana signálu je použita časová doména, která byla použita jako poslední. Dále je nutné počítat s tím, že díky malému rozdílu mezi frekvenci vysílače a přijímače se bude časová doména přijímaných dat po určité době měnit. Z tohoto důvodu je potřeba vyhodnocovat přechody mezi časovými doménami. Nastávají zde dva případy kdy je nutné upravit příjem dat a to při přechodu z časové domény A do D a z D do A.

Pokud se přechází z časové domény A do D a neupravil by se příjem znamenalo by to, že jeden bit dat by byl při přechodu přijat dvakrát. Z tohoto důvodu se při přechodu žádný bit nepřijímá a začátek příjmu se posouvá o jeden hodinový takt.

V dalším případě při přechodu z časové domény D do A by naopak došlo k vynechání jednoho datového bitu. Z tohoto důvodu je nutné při tomto přechodu přijmout dva datové bity z obou domén. V následujícím hodinovém taktu se již přijímají data z domény A.



Obr. 5 Obvod pro vyhodnocování přijmu dat

Aby se předešlo výše uvedenému problému je nutné upravit příjem dat tak, aby bylo možné přijímat 0 až 2 bity dat. Na obr. 5 je blokové schéma obvodu vyhodnocování přijímaných dat. Do obvodu vstupují datové signály ze všech časových domén a také signály určující ze které časové domény se budou přijímat data. V závislosti na vstupech se generují výstupní data SDATA které jsou 2 bitové a signál DV určující jestli se bude načítat jeden, dva nebo žádný bit ze signálu SDATA do posuvného registru sloužícího k paralelizaci dat.

Výstupní posuvný registr musí být proto také upraven, aby mohl přijímat jeden nebo dva bity vstupních dat. Tedy pokud je nutné přijmou dva bity dat, aby se posunula všechna data o dvě pozice výše. Pokud se přijímá pouze jeden bit, posouvají se data v registru o jednu pozici. Při přijmu se vstupní bit/bity načítají na nejnižší pozici v registru.

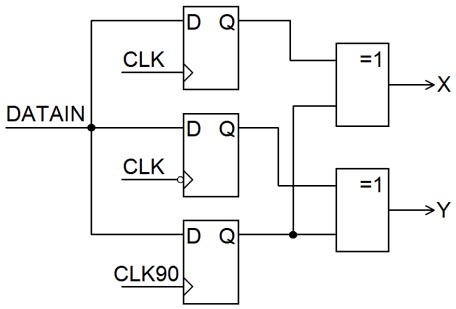
Počet dat načtených do posuvného registru je zaznamenán. Na základě počtu přijatých dat jsou paralelní data předána k dalšímu zpracování. Kvůli již zmíněné možnosti přijmu dvou datových bitů je nutné, aby posuvný registr byl o jeden bit větší než je šířka výstupních paralelních dat.

Paralelní data jsou předávána, když je načteno do registru množství dat odpovídající požadované šířce paralelních dat. Pokud dojde k načtení dvou datových bitů zrovna, když bude v registru chybět již pouze jeden bit dat, předávají se nejvyšší datové bity načtené do registru. Čítač musí na tuto možnost adekvátně reagovat, tak, že se nevynuluje, ale nastaví se na jedničku (jeden bit dat je již načten). Popsaná metoda je převzata z [9].

1.2.2 Metoda založena na fázovém posuvu

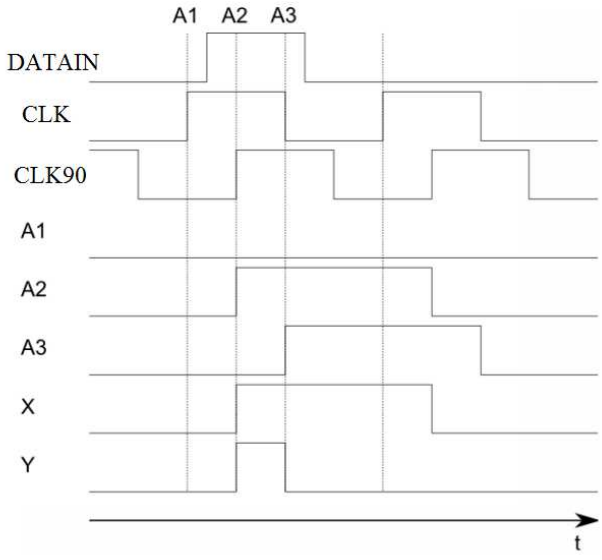
V [10] je uvedena možnost jak rekonstruovat hodinový signál z přijímaných sériových dat. Je zde využit obvod DCM, které je obsažen i v levných obvodech FPGA. Obvody DCM slouží především k syntéze hodinového signálu, ale umožňují také fázově posouvat výstupní hodinový signál. Fázový posuv je možné zvolit buď pevně nebo je ho možné posouvat během chodu obvodu. Posuv během chodu je umožněn pouze o jeden krok v daném rozsahu. Fázi je možné posouvat jak kladně tak záporně.

Pro vyhodnocení fáze se využívá bang-bang (obr. 6) fázový detektor, který je schopen určit pouze znaménko fáze (kladná nebo záporná). Pro vyhodnocení jsou použity tři vzorky z datového signálu (A1, A2, A3). Fáze mezi jednotlivými vzorky je 90 stupňů.



Obr. 6 Bang-bang fázový detektor

Časové průběhy signálů v bang-bang fázovém detektoru jsou na obr. 7. Pro vyhodnocení fáze slouží signály X a Y. Pokud je X=1 a Y=0 je fáze kladná a pokud je X=0 a Y=1 je fáze záporná. Ostatní kombinace X a Y jsou neplatné a nic nám o fázi neřeknou.



Obr. 7 Časové průběhy signálů v bang-bang fázovém detektoru

Stejně jako u metody převzorkování jsou použity čtyři vzorky ze vstupního datového signálu. Vzorkovací signál je ale pouze na poloviční rychlosti oproti rychlosti datového signálu. Protože data budou vzorkována pořád ve stejné časové oblasti, je použit DDR registr, který načítá data na náběžnou i sestupnou hranu hodinového signálu.

Tato metoda umožňuje přijímat data přibližně na dvojnásobné frekvenci, než je schopna metoda převzorkování. To je ale vykoupeno delším časem potřebným k sesouhlasení fáze hodinového signálu s datovým signálem.

Sesouhlasení fáze probíhá pomocí jednoduchého stavového automatu, který zjišťuje aktuální fázový posuv (kladný nebo záporný) a podle toho mění fázi hodinového signálu. Zpoždění je způsobeno tím, že nejprve je potřeba dosáhnout opačného znaménka fázového posuvu než byl po zahájení přijmu. Jakmile je dosaženo opačného znaménka fáze je možné říct, že data a hodiny jsou synchronizovány.

Fázi je také potřeba neustále dostavovat, protože bang-bang fázový detektor umožňuje pouze zjištění znaménka fáze. Není tedy možné určit kdy jsou hodiny ve fázi s daty. Dostavování probíhá tak, že se fáze hodinového signálu posouvá neustále v opačném směru než je aktuální zjištěná fáze dat.

1.2.3 Integrovaný GTP vysílač

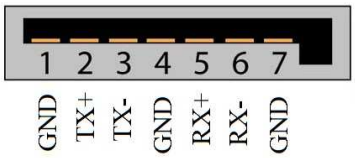
Obvody Virtex a vybrané obvody Spartan 6 od firmy Xilinx mají přímo integrovány hardwarové přijímače umožňující přijímat sériová data v rychlostech řádově gigabitů. Tyto přijímače mají integrované vše potřebné pro příjem dat, tedy PLL, deserializér a také enkodéry a dekodéry. Problém u těchto zařízení je, že jsou pouze v nejvyšších řadách daných obvodů a tedy jejich cena je činí běžně nedostupnými. Např. ve vybraných obvodech Spartan 6 se nacházejí GTP vysílače, které umožňují příjem sériových dat od rychlosti 600 Mb/s po rychlost 3 Gb/s. Přehled o maximálních množstvích a rychlostech GTP vysílačů ve vybraných obvodech je v tab. 3.

Tab. 3 Množství a rychlost GTP vysílačů obvodů firmy Xilinx

Obvod	Max. počet GTP vysílačů	Max. rychlost
Spartan 6	2	3,125 GT/s
Virtex 4	8	3,75 GT/s
Virtex 5	8	3,75 GT/s

1.3 Fyzická vrstva SATA

Fyzická vrstva SATA zajišťuje vysílání a příjem dat [12]. Jsou využívány dvě sériové diferenční linky. Jedna slouží pro odesílání dat (TX) a druhá pro příjem dat (RX). Vzhled a popis SATA konektoru je na obr. 8.



Obr. 8 SATA konektor

Rychlost vysílání je určena verzí SATA kontroléru a také verzí SATA zařízení. V dnešní době jsou dostupné tři verze a to SATA 1, SATA 2 a SATA 3. Rychlost těchto verzí je v tab. 4. Aby byla komunikace mezi zařízením a kontrolérem možná, musí být zvolená rychlost podporována oběma. Kvůli zajištění zpětné kompatibility jsou většinou podporovány, jak u zařízení tak kontroléru, i pomalejší verze než je jejich aktuální. Např. pevný disk, který je ve verzi SATA 3 umožňuje komunikaci také při rychlosti SATA 2

a SATA 1. Tímto je zaručeno že novější zařízení bude fungovat i ve starších systémech a naopak.

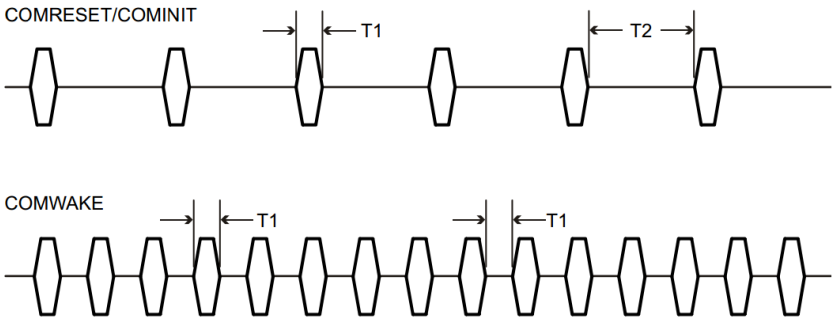
Tab. 4 Rychlost jednotlivých SATA verzí

Verze SATA	Rychlost
SATA 1	1,5 Gb/s
SATA 2	3 Gb/s
SATA 3	6 Gb/s

Dále fyzická vrstva zajišťuje inicializaci zařízení a nastavení rychlosti. Inicializace je zajištěna pomocí OOB signálů. Inicializace probíhá vždy při zapnutí zařízení nebo při přechodu z úsporného stavu. Po inicializaci následuje vyjednávání o rychlosti. Tímto vyjednáváním je zajištěno, že komunikace bude probíhat na nejvyšší možné rychlosti a také zpětná kompatibilita starších zařízení.

1.3.1 Inicializace OOB signály

Spojení se inicializuje pomocí OOB signálů [12]. Tyto signály jsou dva a nazývají se podle toho jestli je vysílá kontrolér nebo zařízení. Signály jsou definovány pouze dobou po kterou se vysílá a dobou po kterou se nevysílá. Časování se udává v UI. Je to počet odeslaných bitů pro rychlost SATA 1. Časování OOB signálů je na obr. 9 a v tab. 5.

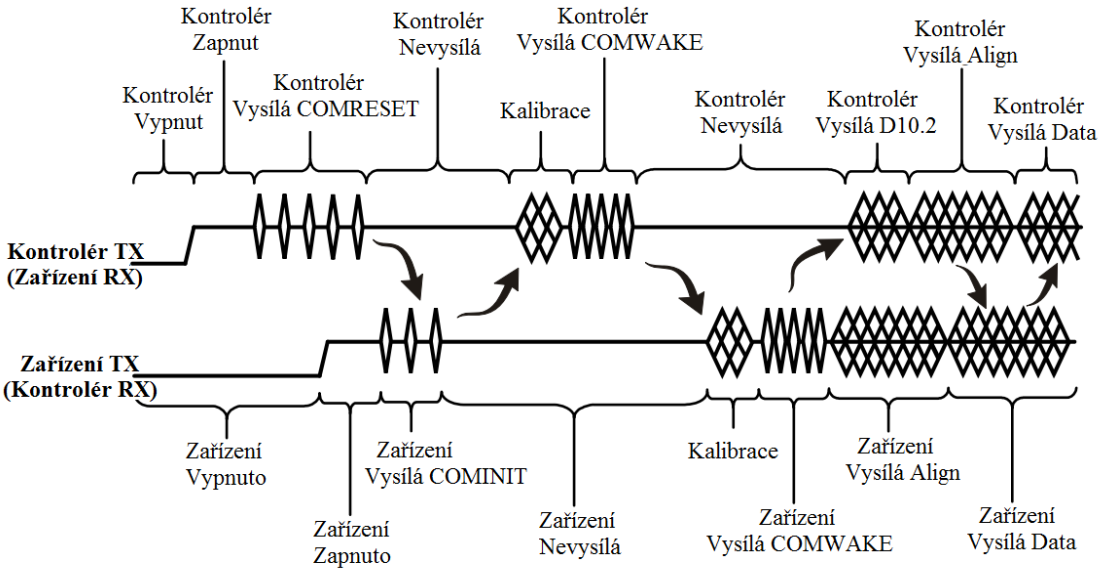


Obr. 9 Časování OOB signálů

Tab. 5 Časování OOB signálů

Čas	Hodnota
T1	160 UI (106.7 ns)
T2	480 UI (320 ns)

První signál se nazývá *COMRESET* (kontrolér) nebo *COMINIT* (zařízení). Tento signál slouží k resetování zařízení a také jím zařízení říká že je připraveno ke komunikaci. Druhý signál je *COMWAKE* (kontrolér i zařízení). Slouží k určení začátku synchronizace. První vždy odesílá signály kontrolér a zařízení na ně odpovídá.



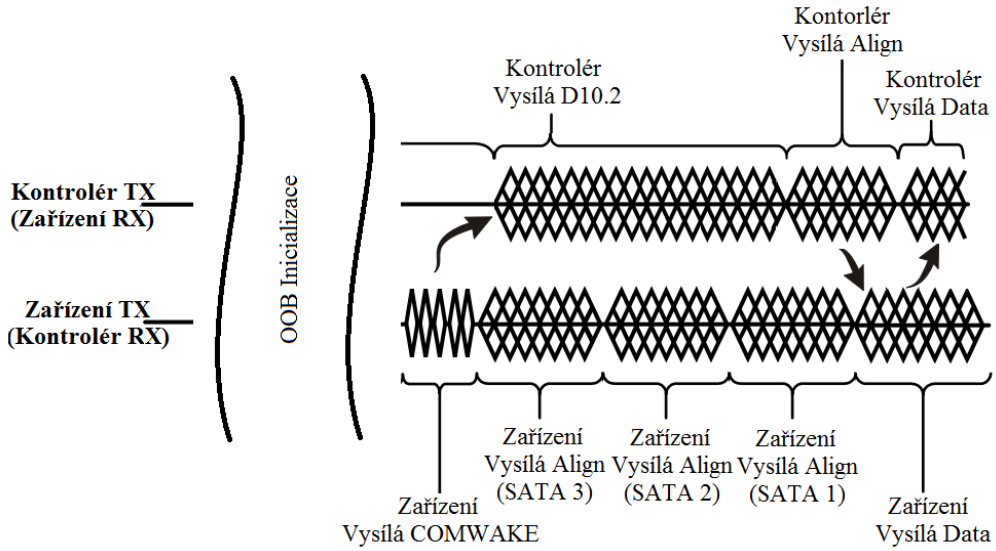
Obr. 10 Časování OOB inicializace komunikace

Inicializace komunikace začíná vždy tím, že kontrolér odešle signál *COMRESET*. Jakmile zařízení příjme *COMRESET* začne vysílat *COMINIT*. Kontrolér příjme COMINIT a začne vysílat *COMWAKE*. Zařízení po přijetí *COMWAKE* vyšle signál *COMWAKE* a začne vysílat primitivu *align* (viz. kódování 8/10). Kontrolér po přijetí *COMWAKE* vysílá data D10.2 a snaží se přijmout primitivu *align*. Pokud primitivu kontrolér přijme, začne ji také vysílat. Když zařízení zachytí primitivu *align*, začne vysílat primitivu *sync* a spojení je navázáno.

1.3.2 Vyjednávání o rychlosti

Vyjednávání o rychlosti probíhá tak, že nejprve se kontrolér snaží komunikovat na své nejvyšší možné rychlosti. Děje se to po přijetí signálu COMWAKE od zařízení, kdy se kontrolér snaží zachytit primitivu *align*. Doba po kterou se ji snaží zachytit je zvolená tak, aby zařízení mohlo až třikrát změnit rychlost. Zařízení vysílá primitivu *align* také nejvyšší možnou rychlostí. Pokud zařízení nepřijme primitivu *align* od kontroléru do určité doby, sníží svou rychlost. To se opakuje dokud se spojení nenaváže, nebo nedojde ke snížení rychlosti na nejnižší možnou.

Pokud se kontroléru nedaří přijmout od zařízení primitivu *align*, tak po určitém čase sníží svou rychlost a opakuje inicializační sekvenci. Toto také pokračuje do té doby než dojde ke spojení nebo se nesníží rychlost na nejnižší podporovanou. Příklad vyjednávání o rychlosti pro kontrolér SATA 1 a zařízení SATA3 je na obr. 11.



Obr. 11 Příklad vyjednávání o rychlosti (kontrolér SATA 1, zařízení SATA 3)

1.4 Linková vrstva SATA

Linková vrstva slouží k řízení přenosu dat a také zajišťuje kódování a výpočet CRC [12]. Pro řízení se využívají speciální znaky (primitivy). Mají délku 32 bitů a vždy začínají kontrolním znakem, který je definován v kódování 8/10. Například každý rámec obsahující data začíná primitivou *SOF* a končí primitivou *EOF*. Používané primitivy jsou v tab. 6.

Tab. 6 Popis používaných primitiv

Název primitivy	Hodnota	Popis
align	0x7B4A4A**BC**	Označuje správné odsazení dat
cont	0x9999AA**7C**	Pokračuje předešlá primitiva
dmat	0x3636B5**7C**	Předčasné zastavení DMA přenosu
EOF	0xD5D5B5**7C**	Konec vysílaného datového rámce
hold	0xD5D5AA**7C**	Požadavek o pozastavení právě odesílaných dat
holda	0x9595AA**7C**	Data byla pozastavena
r_err	0x5656B5**7C**	Chyba při příjmu datového rámce
r_ip	0x5555B5**7C**	Probíhá příjem dat
r_ok	0x3535B5**7C**	Příjem dat proběhl bez chyb
r_rdy	0x4A4A95**7C**	Připraven přijmout data
SOF	0x3737B5**7C**	Začátek vysílaného datového rámce
sync	0xB5B595**7C**	Synchronizace
wtrm	0x5858B5**7C**	Čeká na potvrzení bezchybného příjmu dat
x_rdy	0x5757B5**7C**	Připraven odeslat data

Primitiva *align* je využívaná k zajištění správného odsazení dat. To je potřeba, aby sériová data přijatá fyzickou vrstvou byla převedena na paralelní ve správném pořadí a data mohla být dále zpracována. Tato primitiva jako jediná začíná kontrolním znakem K28.5 (viz. kódování 8/10), který je jedinečný a může tedy být v datovém proudu nalezen. Podle specifikace SATA je potřeba tuto primitivu odesílat po každém odeslání 256 DW (32 bitových datech).

Primitiva *cont* označuje pokračování odesílání předešlé primitivy. Například se čtyřikrát odešle primitiva *sync*, která se odesílá v nečinnosti, a po ní primitiva *cont*, která

značící její pokračování. Je určena hlavně pro potlačení EMI a to tím, že za touto primitivou se odesílají náhodná data. Nedochází tedy k odesílání stále stejné sekvence dat a EMI je redukováno. Primitivy, které nemůžou předcházet primitivu *cont* jsou *align*, *dmat*, *SOF* a *EOF*. Předchozí primitiva se ruší přijetím jakékoliv jiné primitivy.

Pokud je připraven k odeslání rámec s daty začne se vysílat primitiva x_rdy. Jakmile je přijímač připraven data přijmou odesílá primitivu *r_rdy*. Začátek odesílaných dat je vždy uvozen primitivou *SOF*. Tato primitiva se odesílá pouze jednou a za ní vždy začíná přenos dat. Přenos dat končí kontrolním součtem a primitivou *EOF*. Poté je odesílaná primitiva *wtrm*. Touto primitivou žádá vysílač o vyhodnocení příjmu. Pokud byla data přijata v pořádku (nebyla nalezena chyba) odešle se primitiva *r_ok*. Pokud došlo k chybě při příjmu (např. chybný kód 8/10, chybné CRC) odešle se primitiva *r_err*. Příklad odesílání datového rámce je v tab. 7.

Tab. 7 Příklad odeslání datového rámce

Data odeslána Kontrolérem	Data odeslána Zařízením	Popis
sync	sync	Klidový stav
sync	sync	Klidový stav
x_rdy	sync	Kontrolér připraven odeslat data
x_rdy	sync	Zařízení dekóduje požadavek o odeslání dat
x_rdy	r_rdy	Zařízení připraveno přijmout data
x_rdy	r_rdy	Kontrolér dekóduje potvrzení o příjmu
SOF	r_rdy	Kontrolér odešle SOF (začátek rámce)
Data 0	r_rdy	Kontrolér odešle Data 0
Data 1	r_ip	Kontrolér odešle Data 1
----	----	----
Data n	r_ip	Kontrolér odešle Data n
CRC	r_ip	Kontrolér odešle CRC
EOF	r_ip	Kontrolér odešle EOF (konec rámce)
wtrm	r_ip	Zařízení dekóduje konec rámce
wtrm	r_ip	Zařízení vypočítalo správné CRC
wtrm	r_ok	Zařízení potvrzuje správnost přijatých dat
wtrm	r_ok	Kontrolér dekóduje dobrý konec
sync	r_ok	Kontrolér se vrací do klidového stavu
sync	sync	Zařízení se vrací do klidového stavu

1.4.1 Kontrolní součet CRC

Cyklický redundantní součet, neboli CRC je používán k detekci chyb během přenosu nebo ukládání dat. Je schopný poměrně spolehlivě odhalit náhodnou chybu vznikající při přenosu. Princip metody vychází z dělení polynomu polynomem. Tedy data jsou dělena polynomem G(x). Výhodou CRC je také poměrně snadná implementace a její rychlost.

CRC pro standart SATA má šířku 32 bitů. Použitý polynom má hodnotu (1) a inicializace (nastavení CRC před výpočtem) je 0x52325032.

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

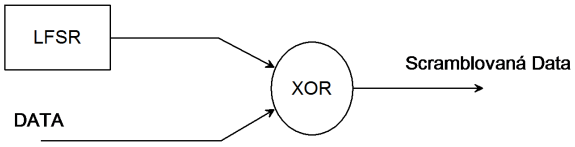
CRC detekuje:

- Všechny jednobitové chyby
- Všechny dvoubitové chyby, pokud má G(x) aspoň 2 členy
- Všechny liché počty chyb, pokud G(x) obsahuje x+1
- Všechny sluky chyb do délky CRC
- Shluky chyb delší než CRC odhalí s pravděpodobností $1/2^n$

Výpočet CRC se aplikuje na všechna data odesílána v rámci, kromě primitiv a dat za primitivou *cont*.

1.4.2 Scrambling dat

Ve standartu SATA jsou odesílaná data nejdřív scramblována. Scramblování znamená, že data jsou logickou operací XOR sečtena s pseudonáhodným kódem, který generuje LFSR čítač. Scramblování je prováděno ze účelem rozprostření EMI v celém vysílacím pásmu.

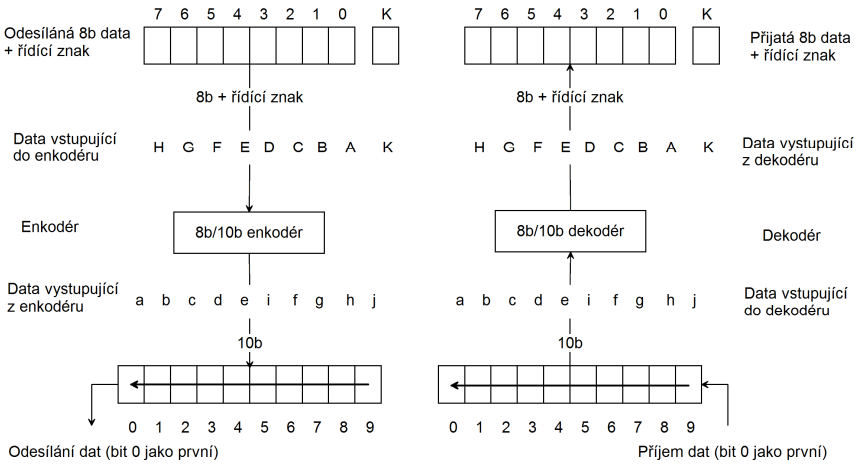


Obr. 12 Princip scramblování dat

Po přijmu takovýchto dat je potřeba je opět descramblovat. Tím se data obnoví do své původní podoby. Použitý polynom pro LFSR čítač je $G(x) = X^{16} + X^{15} + X^{13} + X^4 + 1$ a inicializace je 0xFFFF. Princip fungování scrambleru je na obr. 12.

1.4.3 8b/10b kódování

Kódování 8b/10b převádí 8 datových bitů na 10 bitů kódu. Toto kódování se využívá běžně v telekomunikačních linkách. Výstupní kód je vytvořen tak, aby zůstala zachována konstantní stejnosměrná složka vysílaného signálu a také tak, aby bylo možné rekonstruovat hodinový signál (dostatek změn ve vysílaném datovém proudu).



Obr. 13 Kódování a dekódování 8/10

Tento kód byl vytvořen firmou IBM v roce 1983. Využívá dva enkodéry. Jeden pro převod spodních 5 bitů (EDCBA) dat na horních 6 bitů kódu 8b/10b (abcdei) a druhý na převod horních 3 bitů (HGF) dat na 4 spodní bity kódu 8b/10b (fghj). Enkódovací tabulky jsou v tab. 8 a tab. 9.

Aby byla zachována konstantní stejnosměrná složka vysílaného signálu je potřeba zachovat ve výstupním datovém proudu stejný počet jedniček a nul. Nejlepší by tedy bylo využití 10 bitových kódů, které obsahují 5 jedniček a 5 nul. Bohužel takových kombinací 10 bitového kódů není dostatek pro zakódování všech kombinací 8 bitového kódu. Musejí být využity tedy i kombinace, které mají 4 jedničky a 6 nul nebo 6 jedniček a 4 nuly.

Tab. 8 Enkódovací tabulka pro kód 5b/6b

Vstupy		abcdei Výstupy		rd´	Vstupy		abcdei Výstupy		rd´
Dx	**EDCBA**	**rd+**	**rd-**		**Dx**	**EDCBA**	**rd+**	**rd-**	
D0	00000	011000	100111	-rd	D16	10000	100100	011011	-rd
D1	00001	100010	011101		D17	10001	100011		rd
D2	00010	010010	101101		D18	10010	010011		
D3	00011	110001		rd	D19	10011	110010		
D4	00100	001010	110101	-rd	D20	10100	001011		
D5	00101	101001		rd	D21	10101	101010		
D6	00110	011001			D22	10110	011010		
D7	00111	000111	111000	-rd	D23	10111	000101	111010	-rd
D8	01000	000110	111001		D24	11000	001100	110011	
D9	01001	100101		rd	D25	11001	100110		rd
D10	01010	010101			D26	11010	010110		
D11	01011	110100			D27	11011	001001	110110	-rd
D12	01100	001101			D28	11100	001110		rd
D13	01101	101100			D29	11101	010001	101110	-rd
D14	01110	011100			D30	11110	100001	011110	
D15	01111	101000	010111	-rd	D31	11111	010100	101011	

Tab. 9 Enkódovací tabulka pro kód 3b/4b

Vstupy		fghj Výstupy		rd´
Dx.y	**HGF**	**rd+**	**rd-**	
Dx.0	000	0100	1011	-rd
Dx.1	001	1001		rd
Dx.2	010	0101		
Dx.3	011	0011	1100	-rd
Dx.4	100	0010	1101	
Dx.5	101	1010		rd
Dx.6	110	0110		
Dx.P7	111	0001	1110	-rd
Dx.A7	111	1000	0111	
Poznámka: A7 nahradí P7 když [(rd>0) a (e=i=0)] nebo [(rd<0) a (e=i=1)]				

Kvůli těmto kombinacím je zaveden koncept běžící disparity. Disparita může být kladná nebo záporná. Aktuální disparita vstupuje nejprve do enkodéru 5b/6b (tab. 8). Na základě vstupních dat se vybere výstupní kód. Pokud má výstupní kód stejný počet

jedniček a nul zůstává aktuální disparita nezměněna a vstupuje do dalšího enkodéru 3b/4b (tab. 9). Pokud má výstupní kód rozdílný počet jedniček a nul vybere se výstupní kombinace podle aktuální disparity. Pokud je disparita kladná (větší počet jedniček jak nul) vybere se výstupní kód takový, který má větší počet nul a naopak. Takovýto kód vždy mění aktuální disparitu.

Výsledný kód se skládá z výstupů z enkodéru 5b/6b a 3b/4b. Disparita vstupuje nejprve do enkodéru 5b/6b a až po zpracování do enkodéru 3b/4b. Následně je uložena disparita z enkodéru 3b/4b pro zpracování následujících dat.

V kódu 8b/10b se nacházejí také speciální znaky (K znaky), které jsou určené k řízení komunikace. Kontrolní znaky použité ve standardu SATA jsou v tab. 10.

Tab. 10 Kontrolní znaky u SATA

Název	Byte	abcdei fghj Výstupy	
		Aktuální rd-	Aktuální rd+
K28.3	0x7C	001111 0011	110000 1100
K28.5	0xBC	001111 1010	110000 0101

1.5 Transportní vrstva SATA

Transportní vrstva se stará o vytváření rámců a také jejich rozložení na požadovaná data. SATA má několik typů rámců [12]. Např. pro odesílání hodnot uložených v registrech nebo pro odesílání dat, která byla přečtena nebo mají být zapsána.

1.5.1 Typy rámců

Rámce ve specifikaci SATA mají zkratku FIS. Mezi hlavní rámce patří rámec pro odesílání obsahu registrů a rámec pro odesílání dat. Rámec pro odesílání obsahu registrů slouží k nastavení tzv. stínových registrů, které obsahují příkazy a další informace potřebné pro jejich vykonání. Více o registrech v kapitole SATA registry.

Tab. 11 Typy rámců FIS

Typ FIS	Hodnota	Popis
Registry z Kontroléru do Zařízení	0x27	Přenos dat v registrech
Registry ze Zařízení do Kontroléru	0x34	Přenos dat v registrech
DMA aktivace	0x39	DMA aktivováno
DMA nastavení	0x41	Nastaveni DMA
Data	0x46	Pro odesílání dat
BIST aktivace	0x58	Pro testování
PIO nastavení	0x5F	Nastavení vstupu/výstupu
Nastavení bitů zařízení	0xA1	Přenos dat status registru

Pro přenos obsahu registrů z kontroléru do zařízení se používá *FIS Registry z Kontroléru do Zařízení*. Je to způsob jak předat zařízení ATA příkazy nebo nastavit *Control* registr. Struktura rámce je na obr. 14.

0	Features(7:0)	Command(7:0)	C	R	R	R	Port	Typ FIS (27h)
1	Device(7:0)	LBA(23:16)	LBA(15:8)					LBA(7:0)
2	Features(15:8)	LBA(47:40)	LBA(39:32)					LBA(31:24)
3	Control(7:0)	ICC(7:0)	Count(15:8)					Count(7:0)
4	Další(31:24)	Další(23:16)	Další(15:8)					Další(7:0)

Obr. 14 FIS pro Přenos registrů z Kontroléru do Zařízení

Popis datových polí v rámci:

- *Features*: obsahuje data z registru features, který je ve stínových registrech.
- *Command*: obsahuje data z registru command, který je ve stínových registrech.
- *C*: je nastaven na jedničku pokud se má aktualizovat příkazový registr v zařízení. Nebo je nastaven na nulu pokud se má aktualizovat kontrolní register v zařízení.
- R: rezervováno. Vždy nastaveno na nulu.
- Port: pokud je přítomen rozbočovač portu obsahuje číslo portu na kterém je přítomno zařízení.
- *Device*: obsahuje data z registru Device, který je ve stínových registrech.
- *LBA*: obsahuje data z registru LBA, které jsou ve stínových registrech.
- *Control*: obsahuje data z registru Control, který je ve stínových registrech.
- *ICC*: obsahuje hodnotu, která informuje zařízení o časových limitech. Pokud není podporováno je tato hodnota nulová.
- *Count*: obsahuje data z registru Count, který je ve stínových registrech.
- *Další*: obsahuje parametry pro určité příkazy.

Pro přenos obsahu registrů ze zařízení do kontroléru se používá *FIS Registry ze Zařízení do Kontroléru*. Je to způsob jak zařízení aktualizuje registry v kontroléru. Struktura rámce je na obr. 15.

0	Error(7:0)	Status(7:0)	R	I	R	R	Port	Typ FIS (34h)
1	Device(7:0)	LBA(23:16)	LBA(15:8)					LBA(7:0)
2	Rezervován(0)	LBA(47:40)	LBA(39:32)					LBA(31:24)
3	Rezervován(0)	Rezervován(0)	Count(15:8)					Count(7:0)
4	Rezervován(0)	Rezervován(0)	Rezervován(0)					Rezervován(0)

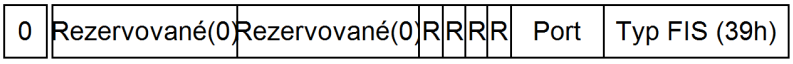
Obr. 15 FIS pro Přenos registrů ze Zařízení do Kontroléru

Popis datových polí v rámci:

- *Error*: obsahuje data z registru Error, který je ve stínových registrech.
- *Status*: obsahuje data z registru Status, který je ve stínových registrech.
- *I*: bit přerušení. Pokud je povoleno přerušení od zařízení obsahuje tento bit hodnotu přerušení v zařízení.
- *R*: rezervován. Vždy nastaveno na nulu.
- *Port*: pokud je přítomen rozbočovač portu obsahuje číslo portu na kterém je přítomno zařízení.

- *Device*: obsahuje data z registru Device, který je ve stínových registrech.
- *LBA*: obsahuje data z registru LBA, který je ve stínových registrech.
- *Count*: obsahuje data z registru Count, který je ve stínových registrech.
- *Rezervován*: vždy nastaveno na nulu.

Pro aktivaci DMA přenosu z kontroléru do zařízení vyšle zařízení FIS pro DMA aktivaci. Struktura rámce je na obr. 16.



Obr. 16 FIS pro DMA aktivace

Pro nastavení vstupu/výstupu se používá FIS pro PIO nastavení. Tento rámec odesílá zařízení, aby informovalo kontrolér např. o počtu dat která budou odeslána ze zařízení nebo mají být odeslána do zařízení. Struktura rámce je na obr. 17.

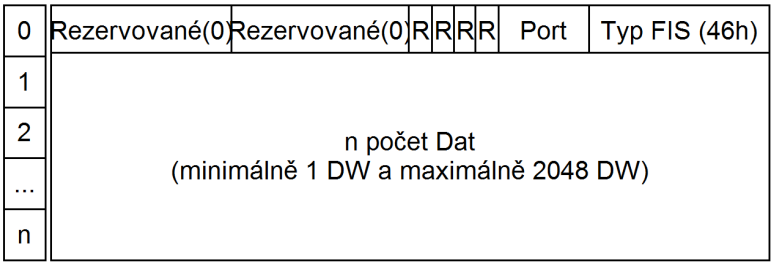
0	Error(7:0)	Status(7:0)	R	I	D	R	Port	Typ FIS (5Fh)
1	Device(7:0)	LBA(23:16)	LBA(15:8)					LBA(7:0)
2	Rezervován(0)	LBA(47:40)	LBA(39:32)					LBA(31:24)
3	E_Status(7:0)	Rezervován(0)	Count(15:8)					Count(7:0)
4	Rezervován(0)	Rezervován(0)	Počet odesílaných bytů(15:0)					

Obr. 17 FIS pro PIO nastavení

Popis datových polí v rámci:

- *Error*: obsahuje data z registru Error, který je ve stínových registrech.
- *Status*: obsahuje data z registru Status, který je ve stínových registrech.
- *I*: bit přerušení. Pokud je povoleno přerušení od zařízení obsahuje tento bit hodnotu přerušení v zařízení.
- *D*: určuje směr vysílání dat. Pokud je D nastaveno na log. 0 znamená to vysílání směrem od kontroléru do zařízení. Pokud je D nastaveno na log. 1 znamená to vysílání směrem od zařízení do kontroléru.
- *R*: rezervováno. Vždy nastaveno na nulu.
- *Port*: pokud je použit rozbočovač portu obsahuje číslo portu na kterém je přítomno zařízení.
- *Device*: obsahuje data z registru Device, který je ve stínových registrech.
- *LBA*: obsahuje data z registru LBA které jsou ve stínových registrech.
- *E_Status*: obsahuje konečnou hodnotu registru Status. Zapíše se až po dokončení přenosu.
- *Count*: obsahuje data z registru Count, který je ve stínových registrech.
- *Počet odesílaných bytů*: obsahuje počet bytů, které budou odeslána z a nebo do zařízení.
- *Rezervován*: vždy nastaveno na nulu.

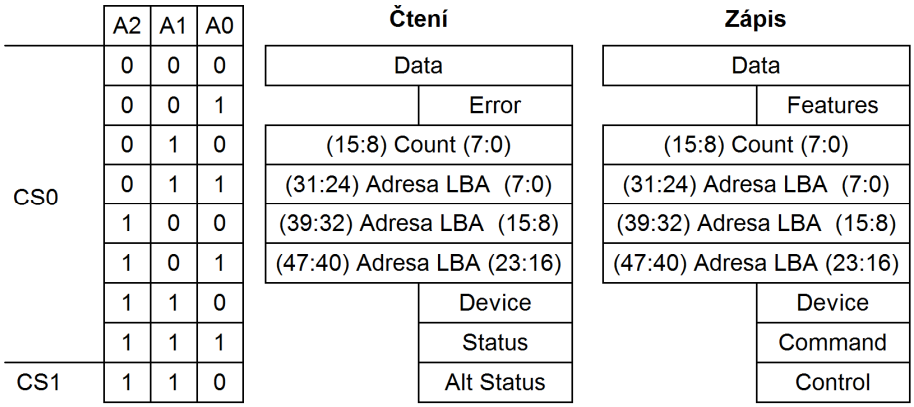
Pro odesílání dat se používá FIS pro data. Obsahuje pouze hlavičku a za ní již následují data. Tento rámec může přenášet minimálně 4 B dat a maximálně 8 kB dat. Struktura rámce je na obr. 18.



Obr. 18 FIS pro Data

1.6 SATA registry

Registry v SATA kontroléru se dělí na stínové a SATA. Stínové registry [13] vycházejí ze starších rozhranní a zůstávají kvůli kompatibilitě. Nazývají se stínové, protože zrcadlí obsah registrů v zařízení a kontroléru. Dělí se na registry pro zápis a pro čtení. Registry pro zápis obsahují data, která mají být odeslána do zařízení. Registry pro čtení obsahují data, která byla přijata ze zařízení. Názvy a adresy registrů jsou na obr. 19.



Obr. 19 Stínové registry

Popis stínových registrů pro čtení:

- *Data*: obsahuje přečtená data.
- *Error*: obsahuje chybový status po vykonání posledního příkazu (význam jednotlivých bitů je na obr. 20).
- *Count*: obsahuje počet sektorů, které byly přečteny nebo zapsány
- *Adresa LBA*: obsahuje adresu, ze které byla naposledy přečtená nebo na kterou byla zapsaná data.
- *Device*: určuje zařízení ze kterého byla čtena nebo do něhož byla zapsána data. Také určuje režim adresace a to buď pomocí staršího CHS (Cylindr Hlava Sektor) a nebo LBA (Adresa Logického Bloku). Význam jednotlivých bitů je na obr. 21.
- *Status*: obsahuje aktuální status zařízení. Význam jednotlivých bitů je na obr. 23.
- *Alt Status*: obsahuje totéž co status, ale při jeho čtení se neruší žádost o přerušení.

Popis stínových registrů pro zápis:

- *Data*: obsahuje data, která mají být zapsána
- *Features*: obsahuje data funkce pro určité ATA přikazy.
- *Count*: obsahuje počet sektorů které mají být přečteny nebo zapsány.
- *Adresa LBA*: obsahuje adresu prvního sektoru ze kterého mají být čtena, nebo do kterého mají být zapsána data.
- *Device*: určuje zařízení ze kterého mají být čtena, nebo do něhož mají být zapsána data. Také určuje režim adresace a to buď pomocí staršího CHS (Cylindr Hlava Sektor) a nebo LBA (Adresa Logického Bloku).
- *Command*: obsahuje ATA příkaz.
- *Control*: slouží pro nastavení a resetování zařízení. Význam jednotlivých bitů je na obr. 22.

Bit	7	6	5	4	3	2	1	0
Název	BBK	UNC	MC	IDNF	MCR	ABRT	TK0NF	AMNF

Obr. 20 Error registr

Popis error registru:

- *BBK*: indikuje chybné označení požadovaného sektoru.
- *UNC*: došlo k neopravitelné chybě dat.
- *MC*: rezervované pro vyjímatelná média.
- *IDNF*: požadovaný sektor nebyl nalezen.
- *MCR*: rezervované pro vyjímatelná média.
- *ABRT*: indikuje zastavení vykonávání příkazu z důvodu chyby zařízení nebo chybného příkazu.
- *TK0NF*: stopa 0 nebyla nelezena.
- *AMNF*: požadovaná adresa nebyla nelezena.

Bit	7	6	5	4	3	2	1	0
Název	1	LBA	1	DRV	HEAD			

Obr. 21 Device registr

Popis registru zažízení:

- *LBA*: indikuje adresaci pomocí *LBA* (pokud není nastaven, tak je adresace pomocí *CHS*).
- *DRV*: určuje zařízení (log. 1 master, log. 0 slave). Nepoužívá se u SATA.
- *HEAD*: číslo požadované hlavy (u CHS adresování).

Bit	7	6	5	4	3	2	1	0
Název	X	X	X	X	1	SRST	nIEN	0

Obr. 22 Control registr

Popis kontrol registru:

- *SRST*: reset zařízení
- *nIEN*: povolení přerušení od zařízení.

Bit	7	6	5	4	3	2	1	0
Název	BSY	DRDY	DWF	DSC	DRQ	CORR	IDX	ERR

Obr. 23 Status registr

Popis status registru:

- *BSY*: zařízení vykonává příkaz.
- *DRDY*: pokud je v log 1 je zařízení připraveno k vykonání příkazu.
- *DWF*: chyba při zápisu.
- *DSC*: pro optická média. Značí vyhledávání zápisu.
- *DRQ*: jsou požadována data pro zápis nebo jsou připravena přečtená data.
- *CORR*: indikuje opravu dat.
- *IDX*: index.
- *ERR*: indikuje že došlo k nějakému typu chyby.

1.7 ATA příkazy

Zápis a čtení ze zařízení se řídí pomocí ATA příkazů. Tyto příkazy jsou popsány v ATA specifikaci [14]. Hlavní dělení těchto příkazů je na PIO a DMA.

Typy ATA příkazů jsou:

- Ne-datové
- PIO čtení dat
- PIO zápis dat
- DMA protokol
- Paketový protokol (pro optická zařízení)
- DMA fronta protokol
- Diagnostika zařízení
- Reset zařízení

PIO příkazy jsou pro čtení a zápis, které vyžadují, aby data byla čtena nebo zapisována přímo z kontroléru. Pokud je např. požadováno, aby data byla zapsána z pevného disku do operační paměti je potřeba nejprve přijatá data přečíst z kontroléru a následně zapsat do paměti.

Naproti tomu DMA příkazy nevyžadují téměř žádnou asistenci a data přečtena z pevného disku jsou ihned zapsána do operační paměti. Nevýhodou těchto příkazů je, že potřebují DMA kontrolér, který bude zápis řídit. Příklady používaných ATA příkazů jsou v tab. 12.

Tab. 12 Příklady ATA příkazů

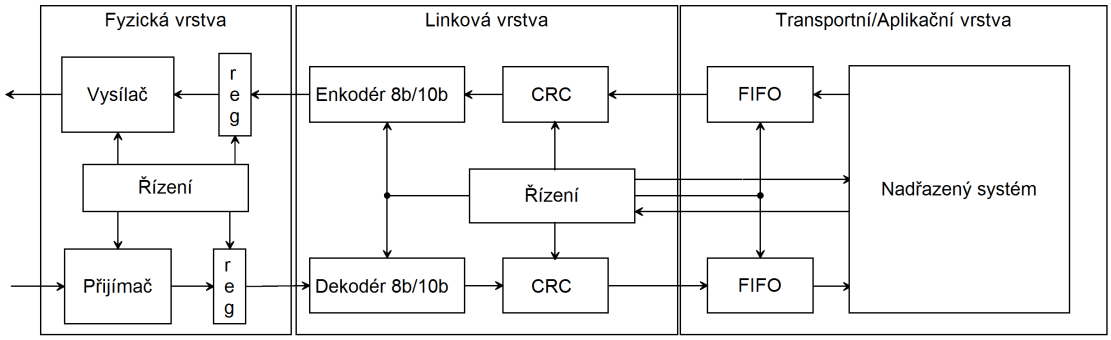
ATA příkaz	Hodnota	Typ příkazu	Popis
Identifikuj zařízení	0xEC	PIO	Zařízení odešle informace o sobě.
Čti sektory ext.	0x24	PIO	Čtení několika sektorů.
Zapiš sektory ext.	0x34	PIO	Zápis několika sektorů.
Čti DMA ext.	0x25	DMA	Čtení pomocí DMA
Zapiš DMA ext.	0x35	DMA	Zápis pomocí DMA

2 Praktická část

V praktické části byly navrženy dva obvody. První obvod slouží jako sériový vysílač pro rychlí sériový přenos dat mezi obvody FPGA. Druhý navržený obvod slouží pro přímé připojení pevného disku k obvodu FPGA. Popisy bloků, stavů a signálů odpovídají použitým názvům ve VHDL souborech.

2.1 Realizace sériového vysílače

Pro možnost otestování a také následného využití bylo potřeba navrhnout vysílač, který umožní implementaci výše popsaných metod sériového příjmu dat. Navrhovaný vysílač musí umožňovat navázání komunikace mezi dvěma zařízeními a odesílání a příjem dat. Jako vzor byla brána komunikace mezi zařízeními SATA kontrolér a SATA zařízení. Blokové schéma navrhovaného vysílače je na obr. 24.

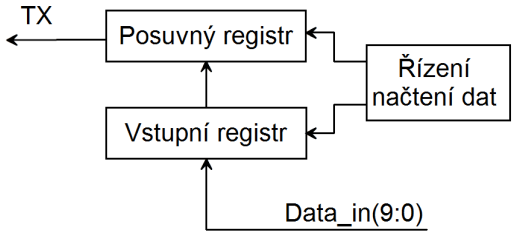


Obr. 24 Blokové schéma navrženého vysílače

Zařízení je navrženo tak, aby fyzická a linková vrstva pracovali na rozdílných hodinových kmitočtech. Fyzická vrstva pracuje na stejném kmitočtu jako jsou odesílána data. Linková vrstva je omezena pouze minimálním kmitočtem, který je maximálně desetkrát menší než je kmitočet na kterém pracuje fyzická vrstva.

2.1.1 Fyzická vrstva

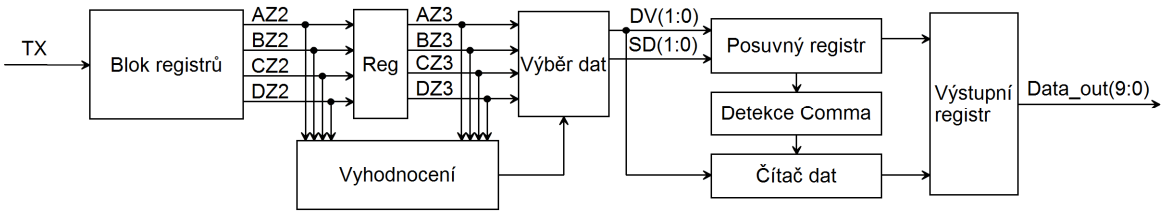
Fyzická vrstva se stará o odesílání a příjem dat. Zajišťuje také správné odsazení jednotlivých dat. Přijatá data jsou deserializována a předána linkové vrstvě k následnému zpracování. Protože, pro kódování přenosu je použit kód 8b/10b, jsou předávána 10 bitová paralelní data.



Obr. 25 Blokové chéma seriového vysílače

Vysílač je realizován, jako jednoduchý posuvný registr. Blokové schéma vysílače je na obr. 25. Nejprve jsou data paralelně nahrána do registru a poté postupným

vysouváním sériově odesílána. Je zde použit čítač, který slouží k časování načítání paralelních dat. Paralelní data jsou získávána z linkové vrstvy sériového vysílače.



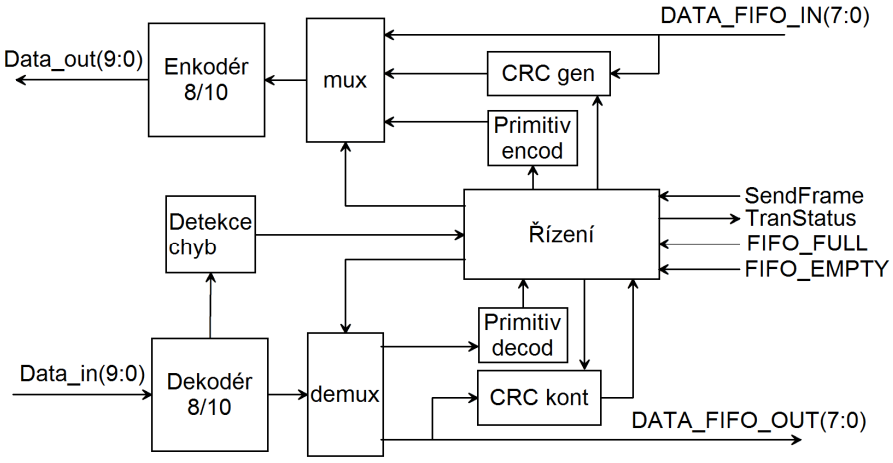
Obr. 26 Blokové schéma sériového přijímače

Přijímač využívá metodu převzorkování. Blokové schéma je na obr. 26. Jak již bylo popsáno v teoretické části (Metoda převzorkování), data jsou vzorkována čtyřikrát vyšší rychlostí než je rychlost odesílání dat. Na základě těchto vzorků se vyhodnocuje časování dat a data jsou zrekonstruována. Přijatá deseti bitová data jsou odesílána linkové vrstvě. Přijímač se také stará o správné odsazení dat. Odsazení je detekováno při přijmu speciálního znaku K28.5 (comma). Tento znak je jedinečný, tzn. že ve standardním případě je nemožné ho získat jakoukoli kombinací běžně odesílaných dat.

2.1.2 Linková vrstva

Linková vrstva slouží k řízení přenosu. Je navržena jako zjednodušená verze Linkové vrstvy SATA standardu [12]. Řízení přenosu dat je naprosto totožné se standardem, ale není zde využito řízení napájení, které standard také obsahuje.

Řízení je realizováno pomocí speciálních znaků (primitiv), která jsou předávána mezi zařízeními obsahující tento vysílač. Dále se linková vrstva stará o kódování přenosu a vypočet CRC. Data která mají být vysílána jsou uložena ve vstupní FIFO paměti. Přijatá data jsou po přijetí uložena ve výstupní FIFO paměti. Blokové schéma linkové vrstvy sériového vysílače je na obr. 27.

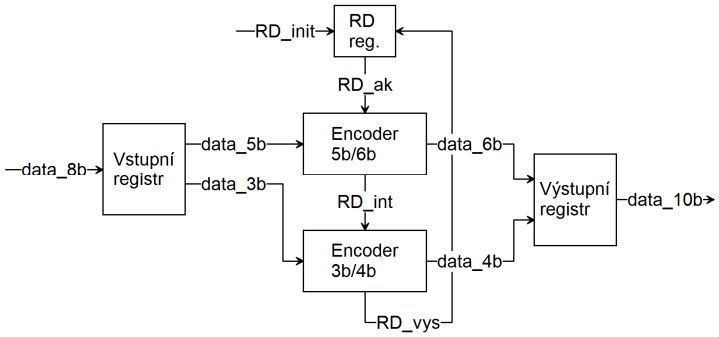


Obr. 27 Blokové schéma linkové vrstvy sériového vysílače

Encoder/decoder 8b/10b

Pro kódování přenášených a přijímaných dat je použit kód 8b/10b. Enkodér je realizován pomocí enkódovací tabulky. Kód, který bude přenášen je vybrán podle vstupní hodnoty dat a aktuální hodnoty disparity. Ve skutečnosti jsou v enkodéru obsaženy

dvě enkódovací tabulky. Jedna pro kódování 5b/6b a druhá pro kódování 3b/4b. Kódování probíhá podle metody popsané v teoretické části. Blokové schéma enkodéru 8b/10b je na obr. 28.



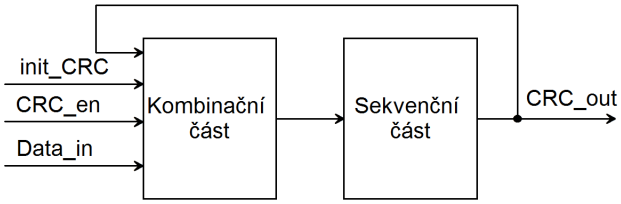
Obr. 28 Blokové schéma enkodéru 8b/10b

Enkodér si sám řídí načítání odesílaných dat. Jsou zde čtyři možnosti dat, které mohou být odeslána. První je odeslání primitivů, které slouží k řízení přenosu. Druhá jsou data určená k odeslání. Třetí je odeslání vypočítané hodnoty CRC a čtvrtá je odesílání tzv. výplňkových dat, která jsou odesílána pokud není potřeba odesílat žádnou z výše zmíněných možností.

Dekodér 8b/10b převádí přijatý deseti bitový kód zpět na jejich osmi bitovou hodnotu. deseti bitová data jsou načítána z fyzické vrstvy vysílače/přijímače. Po dekódování jsou tato dat předána k dalšímu zpracování. Dekodér také signalizuje pokud byl přijat řídicí znak. Dále v dekodéru probíhá kontrola přijatých dat na základě platnosti deseti bitového kódu a aktuální disparity. Pokud jsou přijata neplatná data, dekodér o tom informuje. Poté na základě počtu neplatných dat je vyhodnoceno jestli je připojení stabilní a nebo došlo k rozvázání spojení.

CRC

Pro výpočet kontrolního součtu CRC byl vytvořen CRC generátor. CRC generátor byl převzat z [11]. Tento generátor je navržen tak, že si na základě požadovaného polynomu sám vygeneruje kombinační logiku pro výpočet. Dále umožňuje nadefinovat šířku CRC kontrolního součtu a šířku dat ze které bude počítán. Blokové schéma generátoru CRC je na obr. 29.



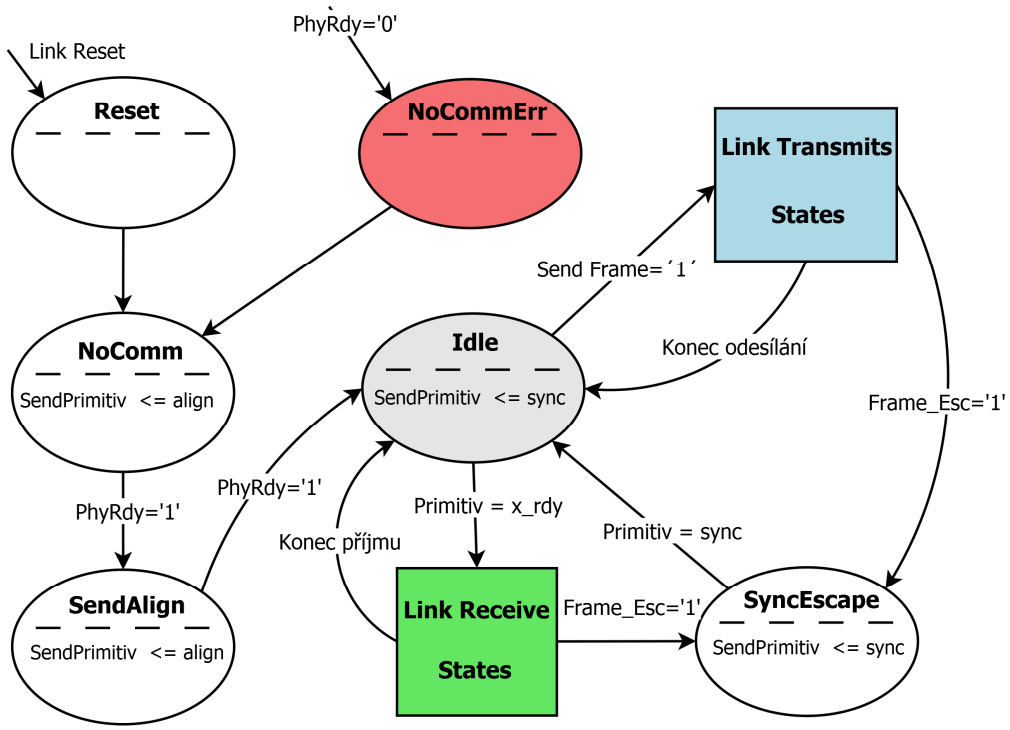
Obr. 29 Blokové schéma generátoru CRC

2.1.3 Řídící stavový automat

Příjem a odesílání dat řídí stavový automat umístěný v linkové vrstvě. Stavový automat je navržen podle SATA specifikace [12]. Komunikace na této vrstvě je zajištěna pomocí primitiv. Ty zajišťují správnou synchronizaci příjmu a odesílání dat a také řízení provozu.

Idle stavy

Při spuštění přijímače se nejprve stavový automat nachází v *Idle* stavech. Tyto stavy zajišťují synchronizaci mezi komunikujícími zařízeními a navázání spojení. Pokud je spojení navázáno je možné přejít do stavů *transmits*, které slouží k odesílání dat nebo do stavů *receive*, které zajišťují příjem dat. Stavový diagram *Idle* stavů je na obr. 30.



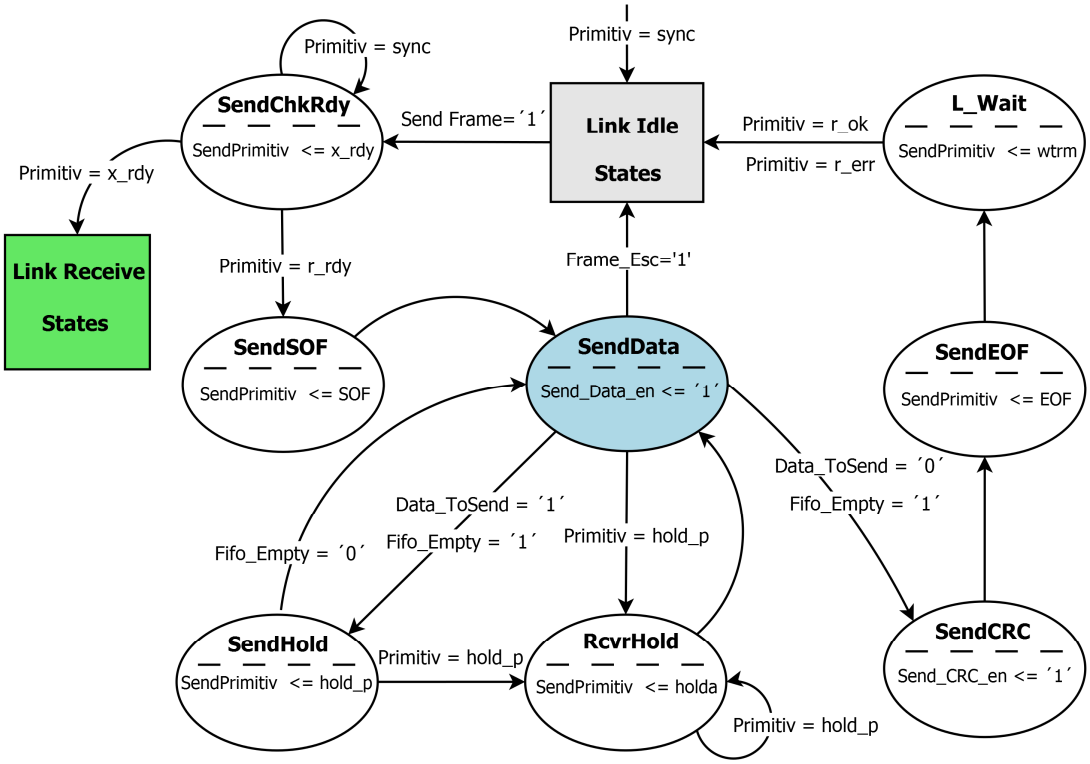
Obr. 30 Stavový diagram pro Idle stavy

Po resetu je stavový automat ve stavu *Reset*. Z tohoto stavu přechází, po odeznění resetovacího signálu, okamžitě do stavu *NoComm*. Pokud je fyzická vrstva připravená komunikovat (*PhyRdy*=‘1‘) přechází se do stavu *SendAlign*. V tomto stavu je odesílán řídící znak *align*, který určuje správné odsazení přijímaných dat. Po odeslání znaku align přechází stavový automat do stavu *Idle*, ze kterého je již možné přijímat a odesílat data. Ve stavu *Idle* se vysílá primitiva sync.

V *Idle* stavech se nachází ještě stav *NoCommErr*, do kterého se přechází pokud není fyzická vrstva připravená komunikovat (*PhyRdy*=‘0‘). Dále je zde stav *SyncEscape*, který slouží k resynchronizaci komunikace během odesílání nebo přijmu dat. Pokud se přechází do tohoto stavu znamená to většinou také ztrátu dat.

Transmits stavy

Pro odesílání dat slouží *Transmits* stavy stavového automatu. Do těchto stavů se přechází pokud je stavový automat ve stavu *Idle* a vyšší vrstva signalizuje, že chce odeslat data (*SendFrame=´1´*). Stavový diagram *Tramsmits* stavů je na obr. 31.



Obr. 31 Stavový diagram pro Transmits stavy

Prvním stavem do kterého se stavový automat dostává je *SendChkRdy*. V tomto stavu je odesílán řídící znak *x_rdy*, který oznamuje druhému zařízení, že jsou připravena data k odeslání. Pokud je druhé zařízení připraveno přijmou data odešle řídící znak *r_rdy*. Po přijmu tohoto znaku přechází stavový automat do stavu *SendSOF*, ve kterém je odeslán řídící znak *SOF*, oznamující začátek odesílání dat. Okamžitě potom se přechází do stavu *SendData*, ve kterém jsou již odesílána samotná data.

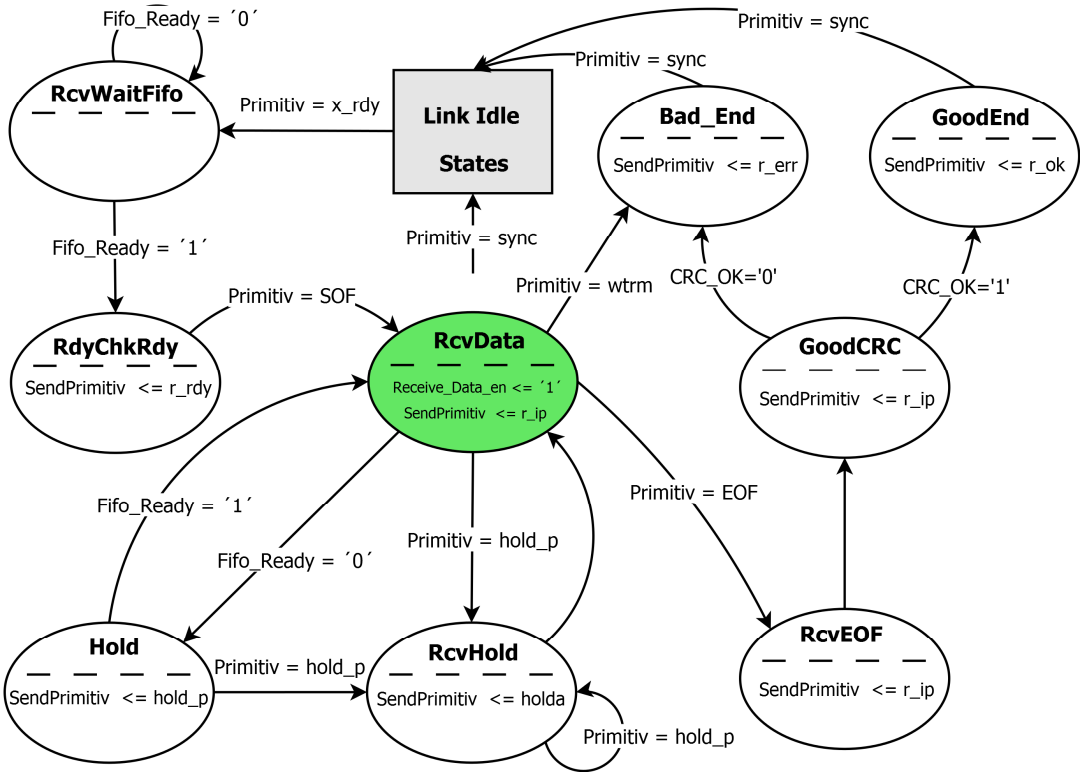
V stavu *SendData* může dojít k situací, kdy vyšší vrstva stále chce odesílat data, ale ta nejsou připravena. V takovém případě se přechází do stavu *SendHold*, ve kterém se odesílá řídící znak *hold_p*, který signalizuje pozastavení odesílání dat. Jakmile jsou data opět připravena přechází se opět do stavu *SendData*. Může zde dojít také k situaci, že zařízení přijímající data není schopné dočasně přijímat data (např. z důvodů plného bufferu). Takový stav je signalizován přijetím řídícího znaku *hold_p*. V tomto případě se přechází do stavu *RcvrHold* ve stavu se setrvává, dokud druhé zařízení odesílá řídící znak *hold_p*.

Po odeslání dat se přechází do stavu *SendCRC*. V tomto stavu je odeslán výsledek kontrolního součtu dat CRC. Po odeslání CRC se přechází do stavu *SendEOF*, ve kterém je odeslán řídící znak EOF, který signalizuje konec odesílání dat. Dále se přechází do stavu *L_Wait*, ve kterém se odesílá řídící znak wtrm a čeká se na potvrzení příjmu dat. Příjem dat je potvrzen buď řídícím znakem *r_ok* (příjem proběhl v pořádku), nebo r_err (během přijmu došlo k chybě, např. chybné CRC). Potom se přechází do stavu *Idle*.

Pokud ve stavech *Tramsmits* (kromě stavu *SendChkRdy*) dojde k přijetí řídícího znaku sync, přechází se do stavu *SyncEscape* a odesílání dat je zrušeno.

Receive stavy

Pokud je stavový automat ve stavu *Idle* a je přijat řídící znak *x_rdy*, který signalizuje, že jsou připraveny data k přijetí, přechází se do *Receive* stavů stavového automatu. Stavový diagram je na Obr. 32.



Obr. 32 Stavový diagram pro Receive stavy

První stav do kterého se stavový automat dostává je stav *RcvWaitFifo*, ve kterém setrvává doku není připraven buffer pro příjem dat. Pokud je připraven, přechází se do stavu *RdyChkRdy*, ve kterém je odesílán řídící znak r_rdy, který signalizuje připravenost k příjmu dat. Jakmile je přijat řídící znak SOF, přechází se do stavu *RcvData*, ve kterém je povoleno načítání dat do výstupního bufferu. Během příjmu dat je odesílán řídící znak *r_ip*, který signalizuje příjem dat.

Během příjmu je možnost, že ve výstupním bufferu není dostatek místa. V takovém případě se přechází do stavu *Hold*, ve kterém se začne odesílat řídící znak *hold_p* jako požadavek o přerušení odesílání dat. Jakmile je vstupní buffer opět připraven přechází se do stavu *RcvData*. Pokud je během příjmu dat přijat řídící znak *hold_p* znamená to pozastavení odesílání dat a přechází se do stavu *RcvHold*. V tomto stavu se setrvá dokud je přijímán řídící znak hold_p.

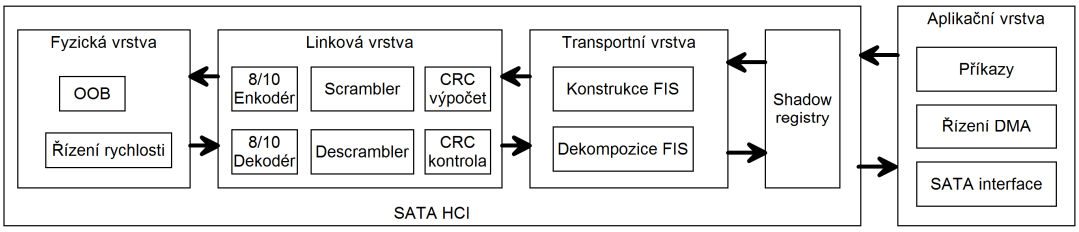
Konec příjmu dat nastává po přijetí řídícího znaku *EOF*. V takovém případě se přechází do stavu *RcvEOF*. Dále se přechází do stavu *GoodCRC*, ve kterém je zkontrolována správnost kontrolního součtu CRC. Pokud součet souhlasí, přechází se do stavu *GoodEnd*, ve kterém je odesílán řídící znak *r_ok* a po přijetí řídícího znaku *sync* se přechází do stavu *Idle*. Pokud kontrolní součet nesouhlasí, přechází se do stavu *Bad_End* a je odesílán řídící znak *r_err*. *Po* přijetí řídícího znaku *sync* se přechází do stavu *Idle*.

Pokud je během přijmu přijat řídící znak *sync* (kromě stavu *GoodEnd* a *Bad_End*) přechází se do stavu *SyncEscape* a příjem dat je zrušen.

2.2 Realizace SATA kontroléru

Kontrolér pro připojení SATA disku obsahuje (oproti rozhraní pro komunikaci mezi obvody FPGA) transportní a aplikační vrstvu, která vychází přímo ze specifikace SATA. Tyto vrstvy nebyly použity u rozhraní pro komunikaci mezi obvody FPGA. Jsou totiž uzpůsobeny pro řízení paměťových médií, zatímco navržené rozhraní je pro univerzální použití, kdy uspořádání dat a způsob komunikace závisí na konkrétní aplikaci.

Blokové schéma navrženého SATA kontroléru je na obr. 33 . Umožňuje odesílání a příjem příkazů nebo dat. Navržený kontrolér je určen k připojení na vnitřní sběrnici systému, který bude obsahovat řídící procesor.

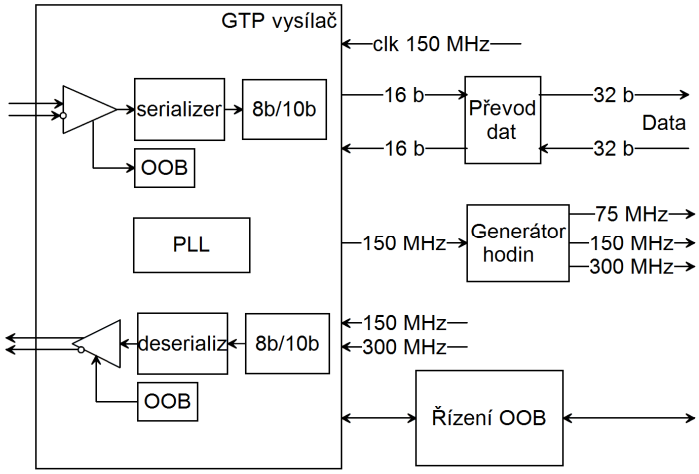


Obr. 33 Blokové schéma SATA kontroléru

2.2.1 Modul fyzické vrstvy

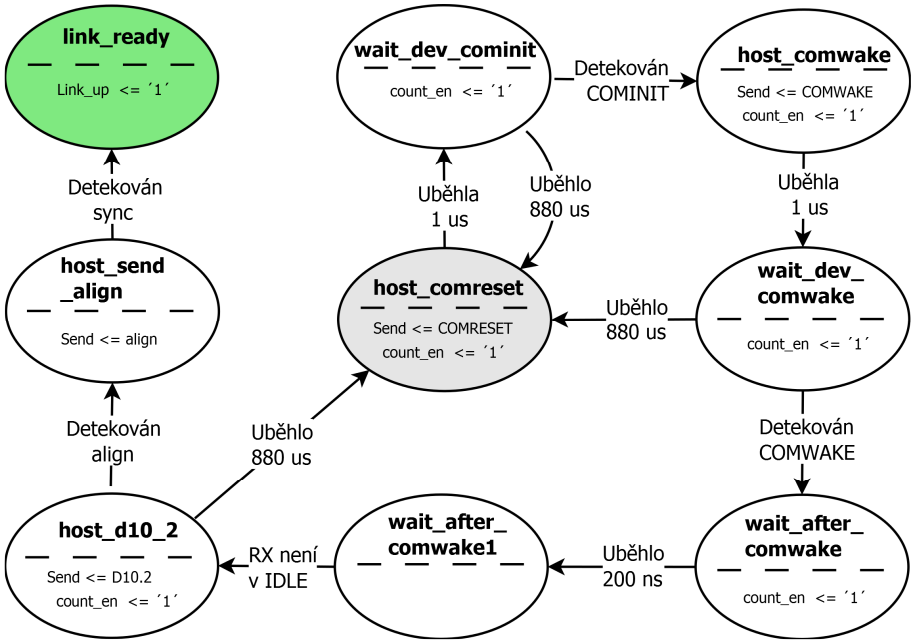
Fyzická vrstva využívá GTP vysílač, který je obsažen v obvodu FPGA Virtex 5. Tento vysílač umožňuje komunikaci při rychlosti SATA 2 (3 Gb/s). U vysílače je možné nadefinovat šířku paralelních dat a jestli má být použit enkodér a dekodér 8/10. Šířku dat je možné zvolit buď 8 bitů nebo 16 bitů. Dále GTP vysílač umožňuje inicializaci komunikace pomocí OOB signálů.

Protože se data na vyšších vrstvách přenášení po 32 bitech byl zvolen 16 bitový datový výstup z GTP vysílače a ten následně převeden na 32 bitů. To přináší také tu výhodu, že vyšší vrstvy budou pracovat na nižší frekvenci (75 MHz pro rychlost SATA 2). Pro svou činnost vyžaduje GTP vysílač hodinový signál na frekvenci 150 MHz nebo 300 MHz. Dále vyžaduje taktovací signál pro vnitřní 8 bitová data 300 MHz a 150 MHz pro 16 bitová výstupní data (SATA 2). Hodinové signály jsou generovány pomocí DCM obvodu. Blokové schéma fyzické vrstvy je na obr. 34.



Obr. 34 Blokové schéma fyzické vrstvy

Pro inicializaci komunikace je ve fyzické vrstvě obsažen blok řízení OOB. Tento blok řídí vysílání OOB signálů a na základě jejich příjmu zahajuje komunikaci. Stavový diagram řízení OOB inicializace je na obr. 35.



Obr. 35 Stavový diagram řízení OOB

OOB inicializace začíná vysíláním signálu *COMRESET*. Tento signál se vysílá po dobu 1 μs. Po této době se přechází do stavu *wait_dev_cominit*, ve kterém se čeká na odpověď od zařízení. Pokud je detekován signál *COMINIT*, přechází se do stavu *host_comwake*. Pokud není detekován signál *COMINIT* po uplynutí 880 μs, přechází se zpět do stavu host_comreset.

Ve stavu *host_comwake* se po dobu 1 μs odesílá signál *COMWAKE*. Poté se přechází do stavu *wait_dev_comwake*. V tomto stavu se po dobu 880 μs čeká na přijetí signálu *COMWAKE* od zařízení. Pokud do této doby není signál detekován přechází se do stavu *host_comreset*. Pokud je detekován signál *COMWAKE*, přechází se do stavu *wait_after_comwake* a po uplynutí 200 ns do stavu *wait_after_comwake1*.

Ve stavu *wait_after_comwake1* se čeká dokud není detekován příjem dat. Po detekci příjmu se přechází do stavu *host_d10_2*, kde se odesílají data *D10.2* (0101010101_b viz. kódování 8/10) a čeká se na přijetí primitivy *align*. Pokud není primitiva přijata do 880 μs přechází se zpět do stavu *host_comreset*. Pokud je přijata primitiva align přechází se do stavu *host_send_align*.

Ve stavu *host_send_align* se odesílá primitiva *align* a čeká se na přijetí primitivy *sync*. Po přijetí primitivy *sync* se přechází do stavu *link_ready* a spojení je navázáno. To je signalizováno signálem *Link_up*.

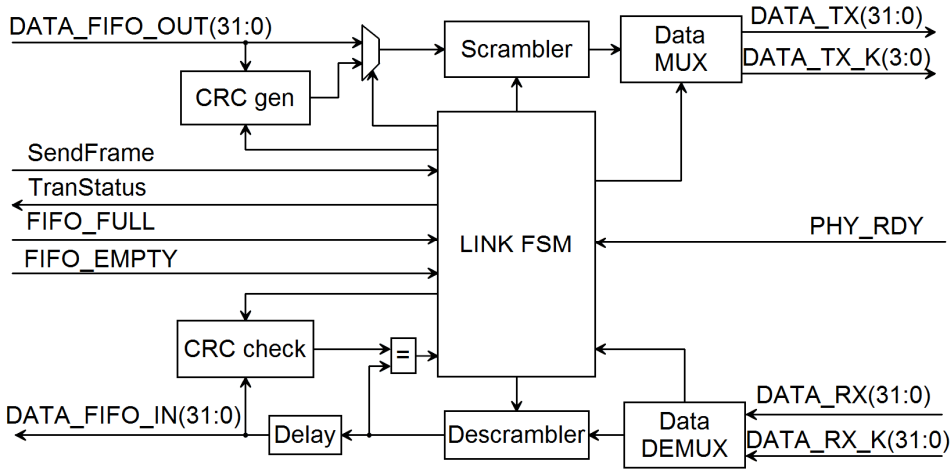
2.2.2 Modul linkové vrstvy

Linková vrstva SATA kontroléru je složena z datového multiplexeru a demultiplexeru, scrambleru a descrambleru, CRC generátoru a CRC kontroly a řídícího stavového automatu. Blokové schéma linkové vrstvy je na obr. 36. Narozdíl od linkové vrstvy v navrženém sériovém vysílači neobsahuje enkodér a dekodér 8/10 , který je přítomen v použitém GTP vysílači. Řídící stavový automat funguje stejně jako u navrženého sériového vysílače. Další rozdíl v linkové vrstvě SATA kontroléru oproti

sériovému vysílači je v šířce přenášených dat (32 bitů oproti 8 bitům) a použití scrambleru a descambleru.

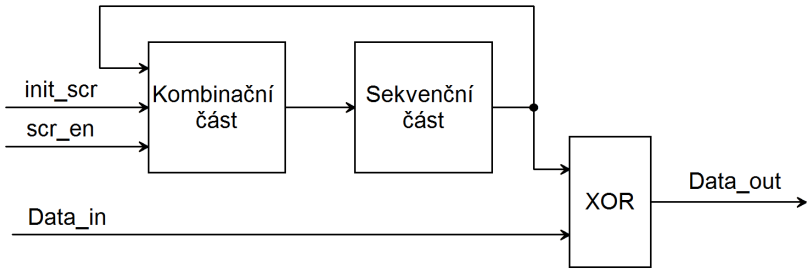
Pomocí datového multiplexeru se vybírá mezi odesíláním dat a nebo primitiv. To co se bude vysílat řídí stavový automat linkové vrstvy. Stavový automat také řídí jaká primitiva se vysílá. Ve specifikaci SATA je požadováno, aby se po každých 256 odeslaných 32 bitových datech odeslala primitiva align. Toto zajišťuje samotný datový multiplexer, který počítá odeslaná data a po odeslání daného počtu dat sám tuto primitivu odešle.

Datový demultiplexer se stará o roztřídění na primitivy a data. Pokud je přijata primitiva, je předána řídícímu stavovému automatu. Dále datový demultiplexer ošetřuje přijetí primitivy cont, která značí pokračování předešlé primitivy. To je potřeba proto, že po primitivě cont následují náhodná data, která by mola být považována, při příjmu datového rámce, za požadovaná data.



Obr. 36 Blokové schéma linkové vrstvy

Scambler a descampler pracují na principu popsaném v teoretické části. Využívají paralelní LFSR čítač, ze kterého vystupují 32 bitová data. Tato data jsou následně pomocí logické funkce XOR sečtena s odesílanými a nebo přijímanými daty. Blokové schéma scambleru je na obr. 37.

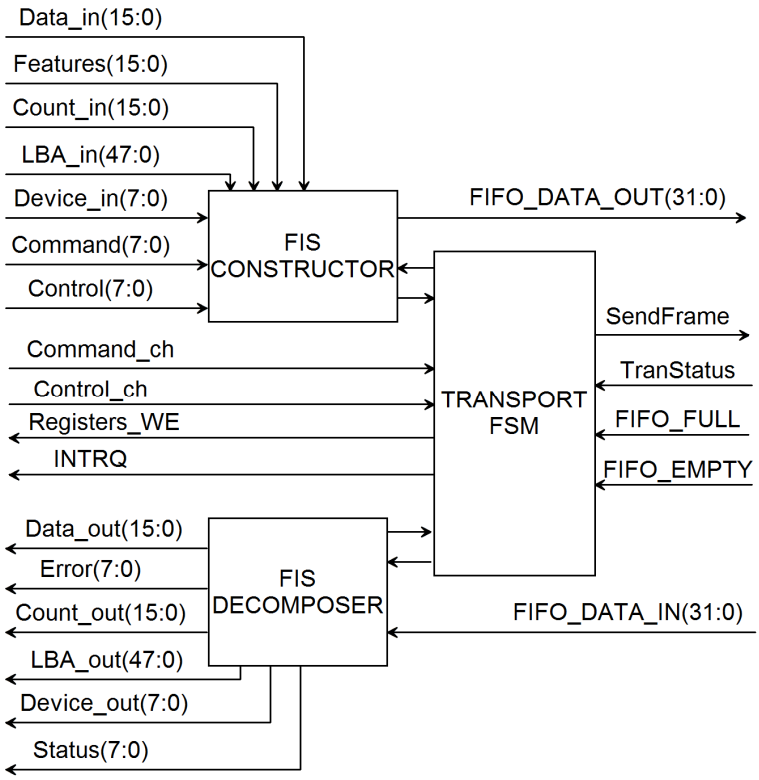


Obr. 37 Blokové schéma Scramleru/Descrambleru

Pro výpočet a kontrolu CRC je použit stejný obvod jako u sériového vysílače. Pouze vstupní data mají šířku 32 bitů. Data která byla přečtena nebo mají být odeslána jsou uložena ve výstupní a vstupní FIFO paměti.

2.2.3 Modul transportní vrstvy

Transportní vrstva se stará o vytváření odesílaných a také rozebrání přijatých datových rámců. Blokové schéma transportní vrstvy je na obr. 38. Rámce jsou vytvářeny na základě dat uložených v registrech pro zápis. O vytváření se stará obvod FIS CONSTRUCTOR. Rámec je vytvořen na základě uložení dat do registru Command a nebo registru Control. Tím se vytvoří rámec pro odeslání registrů. Poté na základě přijatého rámce se může vytvořit datový rámec. To záleží na ATA příkazu, který byl odeslán.

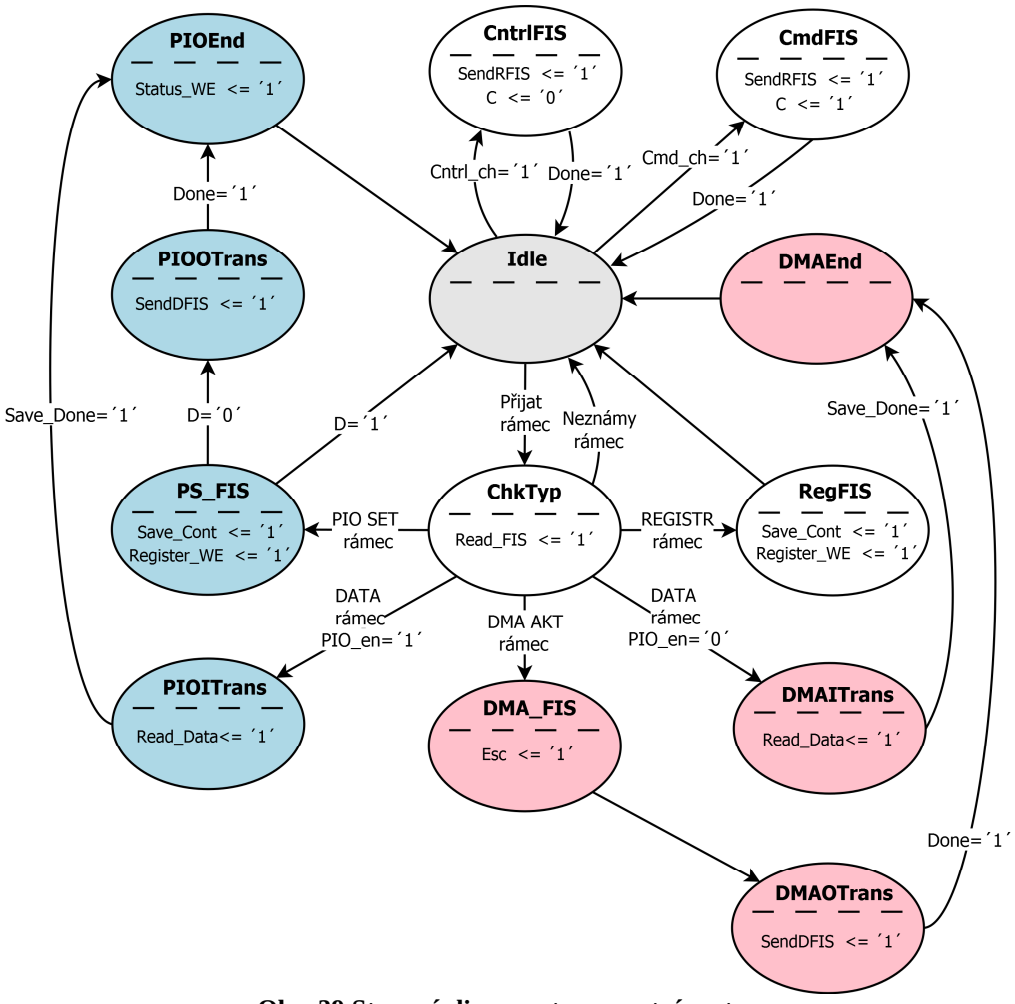


Obr. 38 Blokové schéma transportní vrstvy

O rozložení datového rámce na příslušná data se stará obvod FIS DECOMPOSER. Data jsou po úspěšném přijetí uložena do registrů pro čtení. O tom, že byl přijat datový rámec se transportní vrstva dozví od linkové vrstvy. Data rámce jsou přitom uložena ve výstupní FIFO paměti. Na základě hlavičky rámce je tento buď rozebrán na data jednotlivých registrů nebo se vyhodnotí jako neznámý a je zahozen.

Stavový automat transportní vrstvy

O řízení na této vrstvě se stará stavový automat transportní vrstvy. Ten řídí jaké rámce budou odeslány a také vyhodnocuje přijaté rámce. Umožňuje přenos dat dvěma způsoby a to PIO nebo DMA. Rozdíl mezi PIO a DMA je vysvětlen v teoretické části práce. Stavový diagram stavového automatu transportní vrstvy je na obr. 39.



Obr. 39 Stavový diagram transportní vrstvy

Stavový automat se na počátku nachází ve stavu *Idle*. Z tohoto stavu přechází, buď při změně registru *Command* nebo *Control*, a nebo při příjmu datového rámce. Při změně registru *Command* přechází do stavu *CmdFIS*. V tomto stavu vyžaduje vytvoření rámce pro přenos registrů, přitom je C nastaveno na log. jedničku (viz. Typy rámců). Při změně registru *Control* přechází do stavu *CntrlFIS*. V tomto stavu vyžaduje vytvoření rámce pro přenos registrů, přitom je C nastaveno na log. nulu (viz. Typy rámců). Po odeslání rámce se vrací zpět do stavu *Idle*.

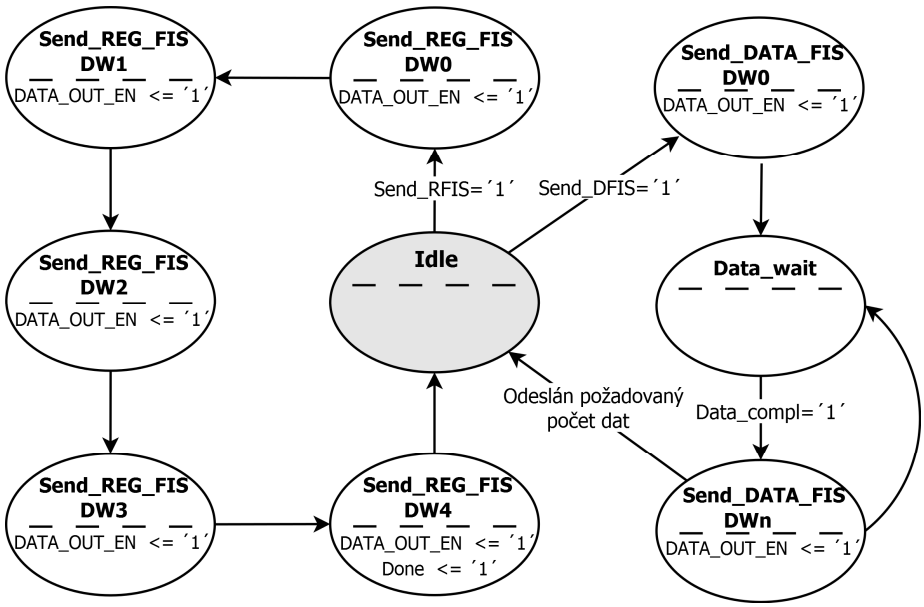
Při zjištění přijetí rámce se ze stavu *Idle* přechází do stavu *ChkTyp*. V tomto stavu je zjištěn typ přijatého rámce. Pokud se jedná o rámec pro přenos registrů přechází se do stavu *RegFIS*. V tomto stavu je vyžadováno rozebrání rámce na jednotlivá data a ta uložena do příslušných registrů. Po uložení dat se přechází do stavu *Idle*.

Při přijetí rámce pro nastavení PIO se přechází do stavu *PS_FIS*. V tomto stavu jsou načtena data z rámce a na základě D (viz. Typy rámců) se buď přechází do stavu *PIOOTrans*, když je D nulové, a nebo *Idle*, když je D nastaveno na jedničku. Ve stavu *PIOOTrans* je vytvořen datový rámec pro odesílání dat. Po načtení požadovaného počtu dat do vstupní FIFO paměti je tento rámec odeslán. Po odeslání se přechází do stavu *PIOEnd* kde je zapsán E_Status do Status registru a přechází se do stavu *Idle*.

Při přijetí rámce pro odesílání dat se přechází do příslušného stavu na základě toho, jestli byl předtím přijat rámec pro nastavení PIO nebo nebyl. Pokud byl, přechází se do stavu *PIOITrans* a data v rámci jsou ukládána do data registru. Po přečtení všech dat se přechází do stavu *PIOEnd* kde je zapsán E_Status do Status registru a přechází se do stavu *Idle*.

Pokud předtím nebyl přijat rámec pro nastavení PIO, přechází se do stavu *DMAITrans* kde by měla být data uložena na příslušnou pozici v paměti pomocí DMA kontroléru. Ale protože tento kontrolér není prozatím implementován, jsou data uložena stejně jako v případě *PIOITrans* stavu.

Posledním případem může být přijetí rámce pro aktivaci DMA. V takovém případě se přechází do stavu DMA_FIS a následně do stavu DMAOTrans. V tomto stavu je jako v případě *PIOOTrans* vytvořen rámec pro přenos dat a po uložení požadovaného počtu dat do vstupní FIFO paměti je odeslán.

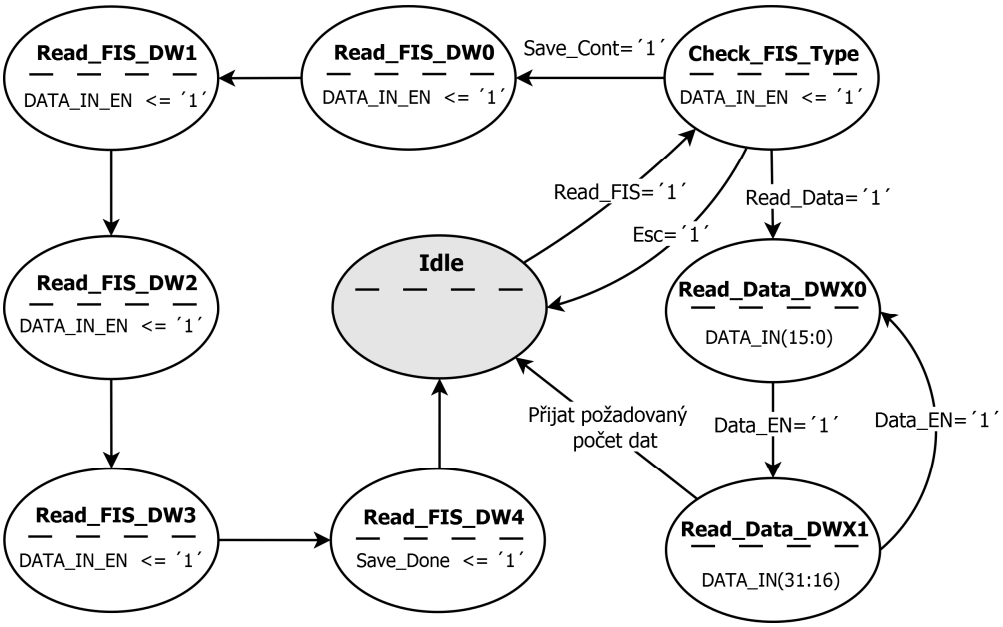


Obr. 40 Stavový diagram konstrukce FIS

Stavový automat pro konstrukci rámců

Stavový diagram stavového automatu pro konstrukci rámců je na obr. 40. Tento stavový automat se stará o vytváření rámců pro přenos registrů a rámců pro přenos dat. Pokud je vyžadováno vytvoření rámce pro přenos registrů, přechází se ze stavu *Idle* do stavu *Send_REG_FIS_DW0* kde je vytvořeno první 32 bitové slovo rámce. Toto slovo je následně uloženo do vstupní FIFO paměti. Protože tento rámec obsahuje takovýchto slov pět, tak se postupně vytvářejí ve stavech *Send_REG_FIS_DW1* až *Send_REG_FIS_DW4*. V posledním stavu je stavový automat transportní vrstvy informován o vytvoření požadovaného rámce a přechází se zpět do stavu *Idle*.

Další možnost je požadavek na vytvoření rámce pro přenos dat. V takovém případě se přechází do stavu *Send_REG_DATA_FIS_DW0* kde je vytvořena hlavička tohoto rámce a ta následně uložena do vstupní FIFO paměti. Poté se přechází do stavu *Data_wait*, ve kterém se čeká na uložení dat do datového registru. Pokud byla data uložena přechází se do stavu *Send_DATA_FIS_DWn*, kde se data načtou do vstupní FIFO paměti. Pokud je takto uložen požadovaný počet dat, je o tom informován stavový automat transportní vrstvy a přechází se do stavu *Idle*. Pokud nebyl načten požadovaný počet dat přechází se opět do stavu *Data_wait*.



Obr. 41 Stavový diagram dekompozice FIS

Stavový automat pro dekompozici rámců

Stavový diagram stavového automatu pro rozložení přijatých rámců je na obr. 41. Tento stavový automat se stará o rozložení přijatých rámců na data příslušných registrů. Pokud byl přijat rámec, vyžaduje stavový automat transportní vrstvy jeho identifikaci. To se děje nastavením signálu *Read_FIS* na logickou jedničku. V takovém případě se přechází do stavu *Check_FIS_Type*. V tomto stavu je přečten typ přijatého rámce a tento předán stavovému automatu transportní vrstvy. Pokud je poté vyžadováno uložení obsahu rámce do dočasných registrů, nastaví se signál *Save_Cont* na log. jedničku. V takovém případě se přechází do stavu Read_FIS_DW0 kde je uložen obsah z prvního 32 bitového slova rámce. Poté se postupně přechází až do stavu *Read_FIS_DW4* kde je uložen obsah z posledního pátého 32 bitového slova rámce. Následně se o úspěšném uložení dat informuje stavový automat transportní vrstvy a přechází se zpět do stavu *Idle*.

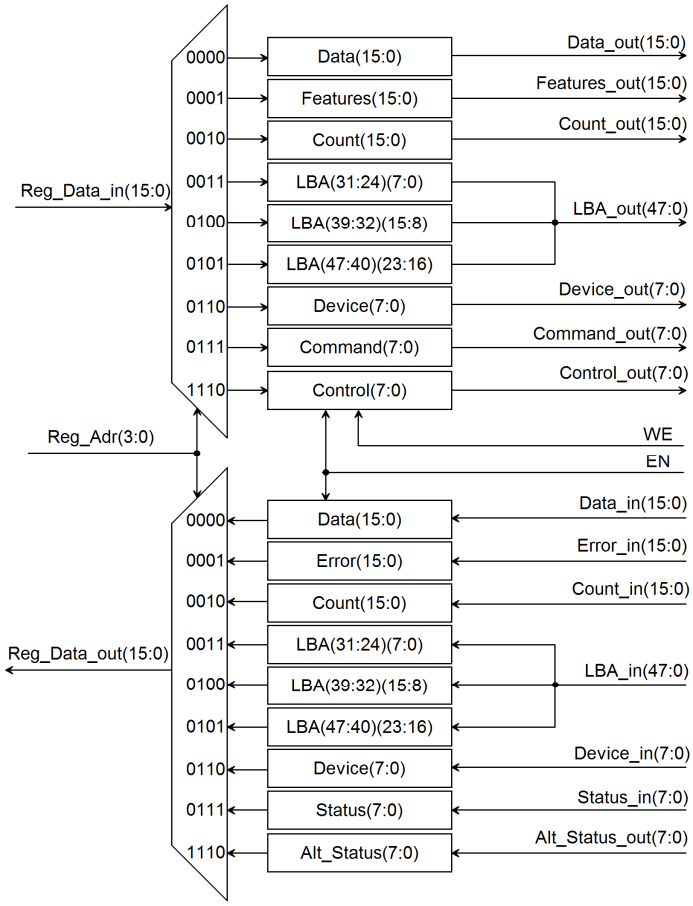
Další možností je požadavek na uložení dat z rámce pro jejich přenos. V takovém případě se přechází do stavu *Read_Data_DWX0*, kde je do datového registru uloženo spodních 16 bitů z přijatého 32 bitového slova. Po přečtení tohoto registru se přechází do stavu *Read_Data_DWX, kde* je do datového registru uloženo horních 16 bitu z přijatého slova. Pokud je požadováno načtení dalších dat přechází se opět do stavu *Read_Data_DWX0* a situace se opakuje. Pokud byl již načten požadovaný počet dat, přechází se do stavu *Idle*.

2.2.4 Modul registrů

Jak již bylo zmíněno v teoretické části, SATA obsahuje dva typy registrů. První typ je určen pro čtení a druhý pro zápis. Tyto registry jsou důležité pro samotnou funkci kontroléru, protože obsahují informace bez kterých by nebylo možné vykonávat příkazy. Tyto informace jsou např. od kterého sektoru mají být čteny data a kolik sektorů se má přečíst. Dále obsahují příkaz, který se má vykonat.

Výhodou je, že kontrolér nemusí znát jednotlivé příkazy. On pouze na základě dat uložených v registrech sestavuje a odesílá rámce. To, jak se bude daný příkaz vykonávat, určuje samo zařízení a to podle jeho odezvy. Pokud například odešle rámec pro nastavení POI, je v tomto rámci specifikován směr vysílání a také počet dat, které je třeba odeslat a nebo přijmout.

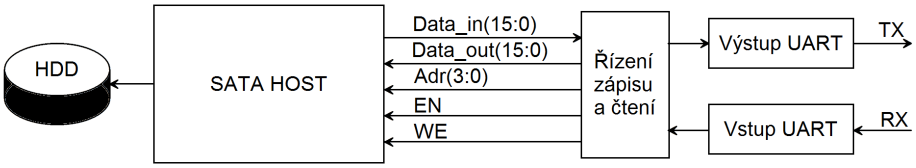
Registrů pro čtení i zápis je stejné množství a to 9. To do kterého registru se bude zapisovat a nebo číst specifikuje jeho adresa pomocí signálu Reg_Adr. Blok registrů obsahuje také signál pro povolení zápisu WE a signál pro povolení přístupu do registru EN. Blokové schéma SATA registrů je na obr. 42.



Obr. 42 Blokové schéma SATA registrů

2.2.5 Testovací aplikační vrstva

Pro otestování byla vytvořena jednoduchá aplikační vrstva. S touto vrstvou se komunikuje přes sériový port počítače. Umožňuje nastavit adresu registru, číst a nebo zapisovat do registru a přečíst a nebo zapsat jeden sektor. Blokové schéma aplikační vrstvy je na obr. 43 a přehled funkcí v tab. 13.



Obr. 43 Blokové schéma aplikační vrstvy

Pokud chceme číst a nebo zapsat do registru je potřeba nejprve nastavit jeho adresu. To se udělá tak, že se odešle šestnáctibitové slovo 0x0001. Po odeslání tohoto slova je možné odeslat adresu registru. Adresa je v rozsahu 0x0000 až 0x000E. Po nastavení adresy je již možné číst z požadovaného registru. Pro čtení se musí odeslat

0x0003. Dále je možné zapsat do registru. To se provede odeslání 0x0002 a dále odesláním hodnoty, která má být zapsána do registru.

Pokud chceme přečíst sektor musí se nejprve nastavit příslušné registry a nakonec command registr (např. pro čtení sektoru 0x0024). Po vykonání příkazu je možné data přečíst a to odesláním hodnoty 0x0005.

Pokud chceme zapsat sektor musí se nejprve nastavit příslušné registry a nakonec command registr (např. pro zápis sektoru 0x0034). Po vykonání příkazu je možné data zapsat a to odesláním hodnoty 0x0004 a odesláním 512 B dat.

Tab. 13 Přehled funkcí aplikační vrstvy

Funkce	Přijatá data	Popis
Nastavení adresy registru	Word 1: 0x0001 Word: Adr (0x0000 až 0x000E)	Nastavení adresy registru ze kterého se bude číst nebo do kterého se bude zapisovat
Zápis do registru	Word 1: 0x0002 Word 2: data registru	Přijatá data se zapíšou do registru
Čtení z registru	Word 1: 0x0003	Data z registru se přečtou a odešlou
Zápis jednoho sektoru dat	Word 1: 0x0004 Word 2 – 257: data sektoru	Přijme data pro zápis jednoho sektoru
Čtení jednoho sektoru dat	Word 1: 0x0005	Odešle data jednoho sektoru přečtených dat

2.3 Implementace navržených rozhraní do obvodů FPGA

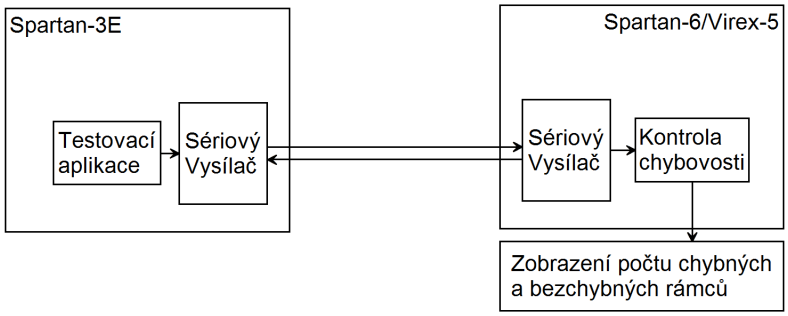
V tab. 14 je přehled spotřebovaných prostředků u různých obvodů FPGA po implementaci navržených rozhraní. Protože, obvody Spartan-6 a Virtex-5 obsahují 6 vstupé LUT tabulky oproti 4 vstupové LUT u obvodu Spartan-3E, je jejich spotřeba nižší. Implementaci je také možné uzpůsobit nastavením syntetizátoru. Tím je možné do jisté míry ovlivnit využití zdrojů.

Tab. 14 Implementace navržených rozhraní do různých obvodů FPGA

Obvod	Spartan-3E (XC3S500E)	Spartan-6 (XC6SLX45)	Virtex-5 (XC5VSX50T)
Sériový vysílač	Registrů: 494 (5%) LUT: 763 (8%) BRAM: 1 (5%) Řezů: 509 (10%)	Registrů: 491 (1%) LUT: 622 (2%) BRAM: 0 (0%) Řezů: 224 (3%)	Registrů: 484 (1%) LUT: 709 (2%) BRAM: 0 (0%) Řezů: 291 (3%)
SATA kontrolér	-	-	Registrů: 1032 (3%) LUT: 1412 (4%) BRAM: 2 (1%) Řezů: 725 (8%)

2.3.1 Testování navržených rozhraní

Pro otestování sériového vysílače byla navržena jednoduchá aplikace, která odesílá a přijímá data. Data jsou generována čítačem a vždy po načtení 1 KB dat se odešlou. V druhém obvodu jsou data přijata a pomocí kontrolního součtu je rozhodnuto o jejich bezchybnosti. Dále druhý obvod počítá kolik datových rámců bylo chybných a kolik bezchybných. Testování proběhlo mezi obvody Spartan-6 a Spartan-3E a mezi obvody Virtex-5 a Spartan-3E. Data vždy odesílal obvod Spartan-3E.



Obr. 44 Blokové schéma testovací aplikace

Blokové schéma testovací aplikace je na obr. 44. Počet chybných a bezchybných datových rámců je zobrazena na LED diodách vývojové desky.

Chybovost přenosu dat mezi obvody Spartan-6 a Spartan-3 byla poměrně velká. Přibližně jeden z pěti datových rámců obsahoval chybu. Toto mohlo být způsobeno špatným vedením signálů, protože při testování přenosu dat mezi obvody Virtex-5 a Spartan-3E, kdy byl použit jiný typ vodičů, nebyl zachycen chybný rámec ani po půl hodině příjmu.

3 Závěr

V první části práce je zkoumána možnost implementace rychlé sériové linky v obvodech FPGA. Sériové rozhraní bylo popsáno v jazyce VHDL. Je složeno z fyzické a linkové vrstvy. Fyzická vrstva odesílá a přijímá data, která jsou kódována tak, aby umožňovala jejich asynchronní přenos. Příjem dat je realizován metodou převzorkování. Linková vrstva, která slouží k řízení přenosu, je navržena podle specifikace SATA. Navržené rozhraní bylo implementováno do obvodu Spartan-3E, Spartan-6 a Virtex-5. Dále bylo otestováno navázání spojení mezi těmito obvody a odesílání dat. Maximální rychlost je momentálně 200 Mb/s.

V druhé části byl vytvořen kontrolér pro přímé připojení SATA pevného disku k obvodu FPGA Virtex-5. Kontrolér se skládá ze tří vrstev a to fyzické, linkové a transportní. Umožňuje čtení a zápis dat na pevný disk. Rychlost komunikace je 3 Gb/s (SATA 2). Kontrolér byl prakticky testován na vývojové desce ML506, která je osazena obvodem FPGA Viretx-5-XC5VSX50T. K testování navrženého kontroléru bylo použito PC. Data přečtená z pevného disku byla odesílána přes sériový port do PC. Data, která měla být zapsána, byla odesílána z PC do navrženého kontroléru.

Navržená rozhraní naleznou uplatnění v aplikacích, které budou potřebovat přenášet a ukládat velké objemy dat a ty následně zpracovávat. Příkladem takové aplikace je systém, který obsahuje několik měřících desek. Na každé desce je obvod FPGA, který sbírá změřená data a ta posílá k dalšímu zpracování hlavnímu obvody, ke kterému je připojen pevný disk. Data jsou hlavním obvodem zpracována a uložena pro budoucí vyhodnocení na pevný disk.

Cíl práce byl splněn. Oba navržené obvody jsou funkční. První obvod je vhodný i pro levnější obvody FPGA, protože nevyžaduje žádné speciální bloky, které by u levných obvodů chyběly. Druhý obvod je určen pouze pro vyšší řady obvodů FPGA, protože vyžaduje využití gigabitového vysílače (GTP).

4 Seznam použité literatury

[1] BROOKS, Douglas. *Differrential Signals: Rules to Live By*. Printed Circuit Design. 2001 4 [cit. 2013-12-6], 11, Dostupné z WWW: http://www.ieee.li/pdf/essay/differential_signals.pdf.

[2] TEXAS INSTUMENTS. *Interfacing Between LVPECL, VML, CML, and LVDS Levels* [online], 2002 [cit. 2013-12-6]. Dostupné z WWW: http://www.ti.com/lit/an/slla120/slla120.pdf.

[3] XILINX. *Spartan-3 FPGA Family: Data Sheet* [online]. V2.5, 2009-12-4 [cit. 2013-12-6]. Dostupné z WWW: http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf.

[4] XILINX. *Spartan-3E FPGA Family: Data Sheet* [online]. V4.1, 2013-06-19 [cit. 2013-12-6]. Dostupné z WWW: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf.

[5] XILINX. *Spartan-3A FPGA Family: Data Sheet* [online]. V2.0, 2010-12-4 [cit. 2013-12-6]. Dostupné z WWW: http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf.

[6] XILINX. *Spartan-6 FPGA Family: Data Shee* [online]. V2.0, 2011-10-25 [cit. 2013-12-6]. Dostupné z WWW: http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf.

[7] XILINX. *Virtex-4 FPGA Family: Data Shee* [online]. V3.1, 2010-09-30 [cit. 2013-12-6]. Dostupné z WWW: http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf.

[8] XILINX. *Virtex-5 FPGA Family: Data Shee* [online]. V2.5, 2009-01-6 [cit. 2013-12-6]. Dostupné z WWW: http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf.

[9] SAWYER, Nick. *Data recovery*. Aplication note: Virtex II and Spartan 3 series. 2005, [cit. 2014-2-6]. Dostupné z WWW: http://epp.fnal.gov/DocDB/0002/000227/001/xapp224.pdf.

[10] HALLER, István. *High-Speed Clock Recovery for Low-Cost FPGAs*. EDAA, 2010, [cit. 2014-2-6]. Dostupné z WWW: http://www.date-conference.com/proceedings/PAPERS/2010/DATE10/PDFFILES/IP2_04.PDF.

[11] CAMPOBELLO, Giuseppe. *Parallel CRC Realization*. IEEE Transaction on computer. 2003[cit. 2014-2-6]. Dostupné z WWW: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1234528.

[12] The Serial ATA International Organization. *Serial ATA Specification*. Rev. 3.0, 2009-06-02 [cit. 2014-2-6]. Dostupné z WWW: https://www.sata-io.org/purchase-spec.

[13] Seagate. *ATA Interface Reference Manual*. Rev. C, 1993-05-21 [cit. 2014-2-6]. Dostupné z WWW: http://www.cs.vsb.cz/grygarek/tuox/sources2/hdd/111-1c.pdf.

[14] Information Technology. *AT Attachment with Packet Interface - 6(ATA/ATAPI-6).* Rev. 3b, 2002-09-26 [cit. 2014-2-6]. Dostupné z WWW: http://www.t13.org/documents/UploadedDocuments/project/d1410r3b-ATA-ATAPI-6.pdf

5 Seznam zkratek

CML	Current Mode Logic (logika v proudovém režimu)
DCM	Digital Clock Manager (správce digitálních hodin)
DDR	Dual Data Rate (dvojnásobná rychlost dat)
DMA	Direct Media Access (přímý přistup k paměti)
DW	Double Word (32 bitové slovo)
FIFO	First In First Out (první dovnitř první ven)
FPGA	Field Programmable Gate Array (programovatelné hradlové pole)
GTP	Gigabit Transport Protocol (protokol pro gigabitové vysílače)
LUT	Look Up Table (náhledová tabulka)
LVPECL	Low Voltage Positiv Emmiter Coupled Logic (nízko napěťová logika s pozitivně vázanými emitory)
LVDS	Low Voltage Diferential Signals (nízko napěťové diferenční signály)
OOB	Out Of Band (vysílání mimo pásmo)
PIO	Programable Input Output (programovatelný vstup a výstup)
SATA	Serial Advanced Technology Attachment (pokročilá technologie sériového připojení)
VHDL	VHSIC Hardware Description Language (VHSIC jazyk pro popis hardwaru)
VML	Voltage Mode Logic (logika v napěťovém režimu)