VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

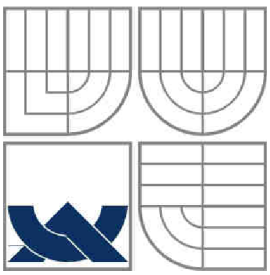
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

BEZPEČNÉ PROPOJENÍ POČÍTAČŮ

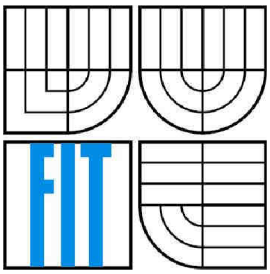
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE JAROSLAV BEDNÁŘ
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

BEZPEČNÉ PROPOJENÍ POČÍTAČŮ

SECURE PC CONNECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV BEDNÁŘ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JÁN KUBEK

BRNO 2007

Zadání bakalářské práce/4096/2006/xbedna24

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Bednář Jaroslav**
Obor: Informační technologie
Téma: **Bezpečné propojení počítačů**
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s výukovou platformou FITkit, rozhraním USB a RS-485 (plně duplexní průmyslová sběrnice založená na 20mA smyčce).
2. Navrhněte konfiguraci pro FPGA FITkitu umožňující propojení dvou kitů pomocí RS-485.
3. Navrhněte jednoduchý software umožňující jednoúčelovou komunikaci dvou PC, ke kterým je přes USB připojen FITkit.
4. Zabezpečte komunikaci po lince RS-485 proti rušení či chybám v přenosu.

Literatura:

- http://www.fit.vutbr.cz/kit

Při obhajobě semestrální části projektu je požadováno:

1. Seznamte se s výukovou platformou FITkit.
2. Seznamte se s rozhraním USB a RS-485.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kubek Ján, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Jaroslav Bednář**
Id studenta: 84467
Bytem: Prušánecká 4208/9, 628 00 Brno
Narozen: 17. 08. 1984, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Bezpečné propojení počítačů
Vedoucí/školitel VŠKP: Kubek Ján, Ing.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Práce se zabývá návrhem a implementací linky RS-485 pro propojení dvou výukových platforem FITkit. Práce přibližuje vlastnosti a možnosti využití této linky, její alternativy použití s FITkitem a popis aplikace, která je pro FITkit napsána.

Klíčová slova

Sériová komunikace, asynchronní komunikace, RS-485, EIA/TIA-485, FITkit, FPGA, MCU.

Abstract

The bachelor thesis deals with the concept and the implementation of RS-485 serial line and its connection to FITkits. The work describes the features and possibly usage of the line, the alternatives of interfacing it to FITkit and the description of the application which was developed for FITkit.

Keywords

Serial communication, asynchronous communication, RS-485, EIA/TIA-485, FITkit, FPGA, MCU.

Citace

Jaroslav Bednář: Bezpečné propojení počítačů, bakalářská práce, Brno, FIT VUT v Brně, 2007

Bezpečné propojení počítačů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Jána Kubeka.
Další informace mi poskytl Zdeněk Vašíček.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Děkuji za pomoc mému vedoucímu Jánu Kubekovi a Zdeňku Vašíčkovi, bez kterých by práce byla o mnoho těžší.

Obsah

Úvod

V době, kdy je všeobecně na elektronickou komunikaci brán stále větší zřetel, stoupají nároky a komunikační technologie se prudce vyvíjí, se najdou odvětví, kde starší technologie zůstávají díky své jednoduchosti, osvědčeným kvalitám a nízké ceně. Pro linku RS-485 hovoří takové parametry, jako potřeba komunikace na poměrně velké vzdálenosti bez nutnosti přenášet velké objemy dat, nízké náklady na pořízení a provoz, či velká odolnost proti rušení. Vlastnosti linky předurčily použití linky pro potřeby průmyslu, kde je nejen dodnes používaná, ale je stále kvalitní alternativou i pro nově vyvíjené systémy. Hovoří pro ni také široké spektrum dostupných zařízení, jako jsou převodníky na jiné druhy komunikace: na linku RS-232, která je stále přítomna na mnoha PC, na Ethernet, což umožňuje připojit zařízení k internetu a data přenášet či příslušné zařízení řídit pomocí internetu nebo na dnes používané rozhraní u osobních počítačů USB a další. Dále existují zařízení s přímo zabudovaným překladačem protokolu, např. Modbus. Pro zvětšení vzdálenosti či zvětšení počtu zařízení na lince je možno použít opakovače. Je tedy velmi pravděpodobné, že se s linkou RS-485 budeme setkávat i v budoucnu.

Cílem mé bakalářské práce je navrhnout komponentu a implementovat pro FPGA výukové platformy FITkit v jazyce VHDL, která podporuje rozhraní RS-485. Toto rozhraní nemá ve standardu žádný komunikační protokol, tedy návrh musí obsahovat i protokol, podle kterého bude komponenta pracovat. Je vhodné, aby protokol počítal i s budoucími možnostmi rozšíření a použitím pro komunikaci mezi FITkity. Dále je vhodné navrhnout součástku, která bude vhodně použitelná k propojení dvou FITkitů, tedy převodník TTL úrovně využívající FITkit na napětí na lince RS-485. Přímo hardwarové propojení v této práci nebude prováděno a zůstane v rovině návrhu. Avšak propojení by v nutnosti mohlo být realizováno za krátkou dobu. Při navrhování je také důležité připravit simulaci jak se komponenty *RS485_dvr* a *help_component* a všechny jejich vnitřní signály chovají. Celkový test bench bude dodán spolu s aplikací. Další částí mé práce je napsat program pro mikrokontoler na FITkitu. Je dostupný překladač pro jazyk C či asembler. Část pro mikrokontroler budu psát v jazyce C, výhodami je větší přehlednost a dostupné kódy pro FITkit, které jsou napsány právě v tomto programovacím jazyce. Mikrokontroler zajišťuje obsluhu FPGA a komunikaci s PC, je to řídící článek platformy FITkit.

První kapitola se zabývá teoretickou přípravou pro práci s linkou RS-485. Jsou připomenuty obecné vlastnosti asynchronní sériové komunikace. Přes rozhraní, které lince RS-485 předcházelo, proudovou smyčku, se dostáváme k charakteristikám právě linky RS-485. Je zde také zmíněna linka RS-422, která je lince RS-485 velmi podobná. Na závěr kapitoly je zmínka o vodičích, jejichž vlastnosti určují ve značné míře parametry celé komunikační linky.

Druhé kapitola se zabývá samotným návrhem a vypracováním práce. Začíná popsáním přístroje, na kterém aplikace poběží. Tím je výuková platforma FITkit. V navazující podkapitole je

popsán návrh a řešení, která jsem si pro aplikaci vybral. Nastiňuje alternativní řešení úkolu a jeho problémy. Po fázi návrhu následuje samotný popis implementace. V tomto oddílu se popisuje systém příjmu a odeslání dat, hlavní vnitřní signály a rozhraní komponent, které jsem navrhnul a implementoval.

V třetí kapitole jsou shrnuty výsledky celé práce.

1 Komunikace

V této kapitole jsou uvedeny výhody sériové komunikace a důvody pro její používání. Nejvíce se kapitola zaobírá sériovou linkou RS-485, která je pro tuto práci stěžejní.

1.1 Sériová komunikace

Sériová komunikace je komunikace, kdy jsou jednotlivé bity posílány po lince postupně. V mnoha případech je celkové řešení levnější na implementaci a i přes počáteční domněnku, že paralelní přenos, kdy bity jsou posílány vedle sebe, musí být vždy zákonitě rychlejší, je sériový přenos dat v mnoha případech rychlejší než při komunikaci paralelní. Sériová komunikace dovoluje použití vyšších frekvencí než paralelní. Mezi důvody dovolující použití vyšších frekvencí patří zpoždění hodinového signálu mezi jednotlivými zařízeními, ovlivňování paralelních vodičů mezi sebou a méně vodičů dovoluje použití lepší izolace vodičů od okolí.

1.1.1 Asynchronní sériová komunikace

Je dosti rozšířená z důvodů své jednoduchosti, ceně a možnosti komunikovat na poměrně velké vzdálenosti.

Asynchronní sériová komunikace je druh komunikace na fyzické vrstvě, kde není přenášen společný hodinový signál. Hodinový signál je generován v každém zařízení zvlášť. Je tedy velmi důležité, kvůli správnosti komunikace, aby se vysílač a přijímač za dobu mezi synchronizacemi nerozešly o více než půl doby bitového intervalu.

Komunikace může začít v jakémkoli okamžiku, proto je nutné sladit přijímač a vysílač a ustanovit začátek přenosu. Používá se k tomu tzv. start-bit. Je to opačná hodnota bitu než je hodnota klidová. Nejčastěji je klidovou hodnotou stanoven signál logická „1“, tudíž start-bit je nejčastěji logická „0“. Poté následuje předem daný počet datových bitů, za ně může být připojen paritní bit. Zde musí být také předem dáno o jakou paritu se jedná. Za paritním bitem, pokud je v komunikaci zařazen, následuje jeden či dva bity klidové tzv. stop-bity. Ten rozděluje od sebe přenášené bloky informací. Má stejnou hodnotu jako klidová hodnota, kvůli možnosti rozeznat začátek dalšího cyklu přenosu. Celek start-bit, data, paritní bit a stop-bit(y) se nazývá rámec.

1.2 Proudová smyčka

Proudová smyčka je jedno z nejstarších sériových rozhraní. Vznikalo zároveň s dálnopisnou technikou na začátku dvacátého století. Logické úrovně jsou vyjádřeny pomocí proudových signálů.

Z tohoto důvodu je smyčka velmi odolná proti rušení. Smyčka je buzena zdrojem proudu. Průchod proudu znamená log.„1“, pokud žádný proud neprochází je na lince log.„0“. Je také potřeba, aby přijímače měly nízký odpor, aby linkou mohl protékat dostatečný proud a napájecí napětí nemuselo být vysoké. Na lince mohou existovat dva druhy zařízení:

1. proud do smyčky dodávají
    - nazývají se aktivní
    - na lince může být pouze jedno
2. proud do smyčky nedodávají
    - nazývají se pasivní
    - na lince se jich může vyskytovat několik

Při větším počtu pasivních zařízení na lince, jsou zařízení zapojena do série. Je možnost mít linku poloduplexní či plně duplexní. Při plně duplexním přenosu můžou obě zařízení komunikovat oběmi směry ve stejný okamžik, poloduplexní přenos umožňuje komunikaci v jeden okamžik pouze jedním směrem. Plně duplexní linka může být pouze v případě jednoho aktivního a jednoho pasivního člena na 2 smyčkách (pro každý směr komunikace jedna). Nejčastěji se používá proud 20 mA, žádná norma však tuto velikost proudu neukládá, a proto se vyskytují i proudy jiné. Vlastnosti proudové smyčky jsou však horší než u linky RS-485, proto pro nová zařízení není vhodná.

1.3 RS-485 standard

Aktuální název RS-485 je TIA/EIA-485 standard, RS-485 bylo předchozí pojmenování linky a dodnes je zažité. Je to standard popisující elektrické vlastnosti fyzické vrstvy referenčního modelu OSI poloduplexní sériové linky (obr. 1.1). Specifikuje diferenciální přenos signálu, kdy je samotný signál přenášen na dvou vodičích zároveň a rozdíl napětí na těchto dvou vodičích určuje logickou hodnotu signálu.

1.3.1 Napětí na lince

Prahové diferenciální napětí pro přijímač je ±200 mV, které je ještě určeno jako signál. Napětí mezi -200mV a 200mV je zakázané. Avšak někteří výrobci tuto problematiku vyřešily tím, že vše pod 200mV je log.„0“ a nad 200mv je to normálně log.„1“. Zakázané pásmo pro tato zařízení neplatí, ale jednotliví výrobci se v tomto ohledu mohou lišit. Pro vysílač platí, že musí při plné zátěži mít na výstupu minimálně ±1,5V a nemělo by přesáhnout ±6V. Maximální zatížení linky je při 32 přijímačích, každý má vstupní impedanci 12kOhm a dvou odporech, nejčastěji každý po 120Ohm, které jsou umístěny na obou koncích linky. Maximální napěťový rozsah (rozdíl zemnících potenciálů + alternující signálové napětí) na vstupu přijímače může být od -7V do +12V. Ve většině případů je linka napájena ze zdroje napětí 5V a země. Na lince se v tomto případě neobjeví větší rozdílové

napětí jak ±5V a vůči zemi budou na obou vodičích napětí kladná. Normou není dáno, jakou logickou hodnotu mají vyjadřovat jaké stavy na vodičích. Většina obvodů pracuje se stavem, kdy ve stavu log.„1“ je vodič A kladnější než vodič B.

1.3.2 Odpory

Velikost ukončovacích odporů by měla být stejná jako je charakteristická impedance vodiče nebo až o deset procent větší. To platí pro linky, kde je využita poměrně velká rychlost nebo délka vedení. Pokud zpoždění při průchodu signálu po vedení je výrazně menší než doba přenosu jednoho bitu, je z důvodu menší zátěže vysílače lepší volit variantu bez zakončovacích odporů. Tato varianta zatěžuje méně vysílač. Délka linky je omezena odporem linky. Pokud zeslabení signálu nemá být větší jak třetina, nemá odpor kabelu přesáhnout ukončovací odpor. Stejnosměrný odpor měděného drátu o průměru 0,511mm je zhruba 84Ohm/km, délka vyplývající z těchto omezení je kolem 1200m.

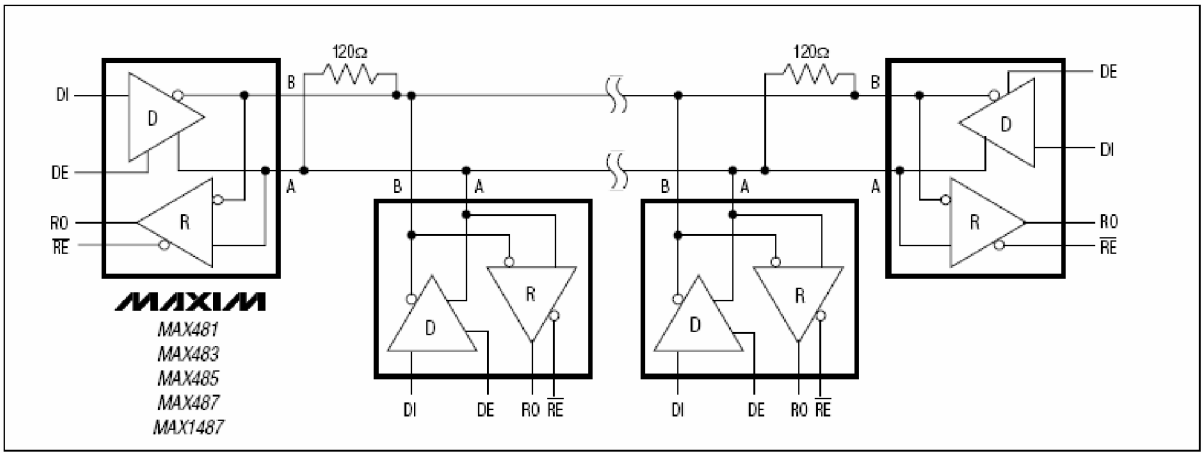
Z důvodů možnosti připojení více vysílačů na linku je nutné, aby vysílače byly třístavové a při nečinnosti přešly do stavu vysoké impedance. Přináší to komplikaci při provozu, kdy se vyskytují situace, že jsou odpojeny od linky všechny vysílače a na lince je nedefinovaný stav. V této situaci je linka náchylná na šumy a hrozí nebezpečí přeslechů. Z tohoto důvodu jsou do linky zapojeny i dva posilovací odpory. Ty definují na lince rozdíl aspoň 200mV v tu chvíli kdy všechny vysílače jsou odpojeny. Ty by měly mít velikost přibližně 720Ohm každý. Jeden je připojen přes napětí k vodiči A a druhý je připojen k zemi a vodiči B. Dále pak jsou definovány velikosti proudu považované za zkrat. Mezi vodiči A a B je to 250mA, proti zemi je to proud 150mA. Pro správnou funkčnost vysílače je nutnost existence zpětné návratové cesty. Jsou dvě možnosti, jak tuto cestu vytvořit. Jeden způsob je ke dvěma stávajícím vodičům přidat třetí a tím propojit země jednotlivých zařízení. Druhá možnost je uzemnit jednotlivá zařízení zvlášť. Při obou variantách se připojují k zařízení ještě odpory zmenšující proudy protékající těmito vodiči. Mají přibližně velikost 100Ohm, tato hodnota je pouze orientační. Při větším potenciálovém rozdílu zemí přijímače a vysílače je nutno použít obvody s galvanickým oddělením.

1.3.3 Rychlosti a vzdálenost přenosu

Rychlost přenosu dat na lince závisí jak na implementaci vysílače, tak i na vzdálenosti na kterou komunikuje. Při vzdálenosti zhruba do 12 metrů se neprojevují zkreslení ani útlum způsobované vedením. Rychlost není omezena vedením a může se pohybovat až do 35Mbit/s. Ve vzdálenosti 12 až 1200 metrů se snižuje přenosová rychlost. Ve 1200 metrech může být maximální přenosová rychlost kolem 100Kbit/s. V této vzdálenosti naráží linka na stejnosměrná omezení vedení a linku nelze prodlužovat ani při dalším snižování přenosové rychlosti.

Někteří výrobci nabízí zařízení podporující linku RS-485 i plně duplexní, kdy každým směrem komunikace vede dvojice vedení.

Standard je nyní spravován organizací TIA a je označen jako TIA-485-A, *Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems (ANSI/TIA/EIA-485-A-98) (R2003)*, značící že standard byl znovu revalidován bez technických změn v roce 2003.



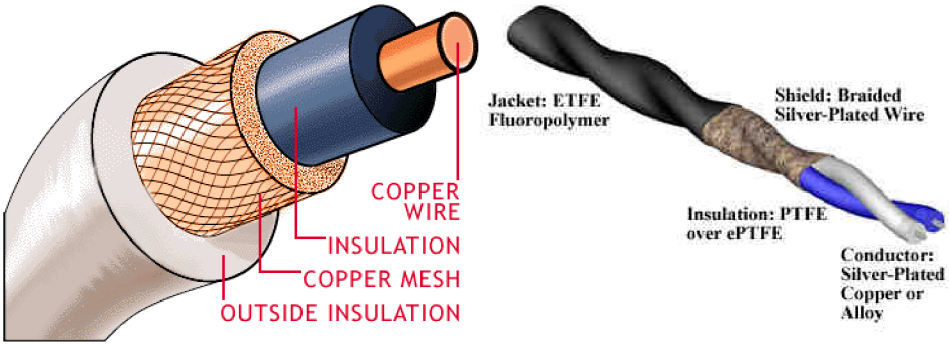
Obr. 1.1: Typické zapojení poloduplexní linky RS-485

1.4 RS-422-B standard

Aktuální jméno je TIA/EIA 422-B standard. Linka podobná lince RS-485. Elektrické vlastnosti jsou stejné jako u linky RS-485, stejné je i to, že není definován žádný protokol či koncovka připojení. Zaměřením se tato linka však liší. Zatímco linka RS-485 je sběrnicového typu, kde se může vyskytnout více vysílačů, linka RS-422 má pouze jeden vysílač a jeden nebo více přijímačů. Vysílač se tedy nemusí od linky odpojovat a na lince nemusí být napojeny posilovací odpory. Ukončovací odpor je pouze jeden na nejvzdálenějším místě od vysílače. Linka je v tomto ohledu jednodušší. Vlastnosti jako maximální vzdálenost vysílače od přijímače nebo maximální přenosové rychlosti se nemění.

1.5 Vodiče

Standard RS-485 žádné vodiče nedefinuje. Ale vzhledem k tomu, aby cena vodičů byla malá a rušení bylo co možná nejmenší, je nejlepším řešením kroucená dvoulinka. Ten má výhody před nekroucenými kabely v tom, že má po celé délce kabelu střední vzdálenost stejnou u obou vodičů, napětí přenesené kapacitní vazbou je u obou vodičů stejné a zmenšuje se vlastní indukčnost krouceného páru. Pro vysoké rychlosti nebo odolnost proti šumu lze použít i koaxiální kabel (obr. 1.2). Ten je z důvodů větší finanční náročnosti v pozadí za krouceným párem.



Obr. 1.2 Příklady vodičů. Nalevo koaxiální kabel, napravo kroucená dvoulinka

2 Návrh a implementace práce

Druhá kapitola seznamuje blíže s výukovým kitem a uvádí jeho některé technické detaily. Následuje stručný popis jazyka VHDL, který je použit pro naprogramování FPGA. Výsledky návrhu jsou uvedeny v podkapitole Návrh 2.3. Jsou zde uvedeny z mého pohledu nejlepší alternativy pro daný problém. K implementaci se vyjadřuje další podkapitola. Jsou zde zmíněny nejdůležitější části aplikace.

2.1 Výuková platforma FITkit

Výuková platforma pro studenty obsahující mikrokontroler, programovatelné hradlové pole FPGA-reprogramovatelný hardware a množství periferií (obr. 2.1). Hradlové pole je podobně programovatelné jako software a s jeho využitím lze testovat a vytvořit hardware bez nutnosti ho přímo vyrábět pro každou aplikaci zvlášť. Je žádoucí, aby si studenti mohli zkusit navrhnout hardware. Navrhování pro FPGA totiž nese sebou některá omezení, která si člověk při navrhování a implementování programu pro simulační nástroje nemusí vůbec uvědomit. Syntéza může na některé nedostatky upozornit. Vzhledem k tomu,že z jazyka VHDL je syntetizovatelná pouze podmnožina, nemusí se syntéza podařit, i když po syntaktické stránce je program v pořádku.

Na FITkitu je FPGA XC3S50-4PQ208C řady Spartan 3. Je to nejjednodušší FPGA z této vývojové řady. Obsahuje až 124 uživatelských vstupů/výstupů, 92 konfigurovatelných logických bloků (CLBs), 1728 logických buněk, 50000 logických hradel. Paměť RAM má 12 kbitů distribuovaných a 72 kbitů v jednom bloku. Dále jsou přítomny 2 jednotky pro správu hodin a 4 násobičky 18x18 bitů.

MCU (z angl. Micro-Controller Unit) je typu MSP430F168 firmy Texas Instruments. Vyznačuje se nízkým příkonem, maximálně 330 uA v aktivním režimu. Je postaven na 16bitové architektuře RISC, instrukční cyklus 125 ns. Paměť je dvou druhů: flash 48 KB + 256 B a paměť RAM 10 kB. Mikrokontroler obsahuje sériové rozhraní, 16 bitové časovače a 12 bitové A/D a D/A převodníky.

2.1.1 Libfitkit

Libfitkit je knihovna jazyka C a je určena pro zjednodušení práce s FITkitem. Sdružuje základní funkce pro práci s mikrokontrolerem. Knihovna zajišťuje komunikaci s terminálem na PC, konfigurace FPGA, komunikace s FPGA a FLASH a další podpůrné funkce. Je volně dostupná a šířitelná pod BSD licencí.

Obr. 2.1 Výuková platforma FITkit

2.2 Rozdělení práce

2.2.1 Program pro MCU

Program pro mikrokontroler je napsán v jazyce C. Používá už hotové funkce z knihovny libfitkit. Samotný program má na práci obsluhu přerušení, která přicházejí z FPGA (klávesnice a komponenta *RS485_dvr*). Z programu se odesílají znaky jak na terminál počítače, tak na displej FITkitu. Mezi obsluhou přerušení dává mikrokontroler blikáním červené diody D6 najevo, že program běží.

2.2.2 VHDL, program pro FPGA

Program je naprogramován v jazyce VHDL. Další alternativou by byl jazyk Verilog, jež je používán více v Americe a v asijských zemích. VHDL je programovací jazyk pro popis hardware. Původně byl vyvinut pro modelování a simulování elektronických systémů. Není závislý na architektuře. Jazyk je standardem IEEE. Existují dvě verze tohoto standardu a to starší z roku 1987 označována jako VHDL-87 a pak novější verze z roku 1993 označována jako VHDL-93. VHDL obsahuje prvky pro paralelismus, pro explicitní vyjádření času a konektivitu. Používá se jak pro popis simulace, tak pro návrh logických obvodů. Od novější verze lze jím popsat i obvody analogové. V architektuře se může objevit jak paralelní tak sekvenční prostředí. Jsou tři možnosti popisu architektury:

- Behaviorální - je použita vysoká míra abstrakce. Nejsou uvažovány šířky sběrnic atp. Syntéza může být neproveditelná, potřebuje nižší míru abstrakce.
- Dataflow - popisuje tok dat. Používá příkazy jako CASE-WHEN nebo WITH-SELECT-WHEN. U paralelních příkazů nezáleží na pořadí v kterém jsou napsány. Vhodné pro syntézu.
- Strukturální popis - spočívá ve vkládání komponent do útvaru zvaného netlist. Tímto způsobem je možné sestavit hierarchickou strukturu. Jsou zde omezení syntézy.

2.3 Návrh

2.3.1 RS-232 na FITkitu

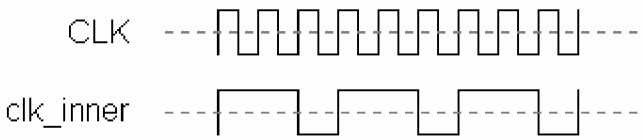
Sériová linka známá hlavně z PC je přítomna i na platformě FITkit. Linka RS-485 se často používá právě jako prodloužení sériového rozhraní RS-232. Problémem v tomto přístupu je, že RS-232 má vodiče i na řídící signály, ale linka RS-485 má vodiče jen na jeden logický signál. Aplikace pracující s převodníkem napětí mezi linkami musí s tímto problémem počítat a použít řídící signály na přepínání vysílání na linku RS-485. Pro propojení dvou FITkitů je možnost propojit je linkou RS-485 jako prodloužení RS-232, ale vyhnuli bychom se vlastnímu návrhu programu a implementaci konfigurace pro FPGA.

2.3.2 Rámec

V případě této komponenty rámec obsahuje celkem 11 bitů a skládá se ze start-bitu, osmi bitů datových, jednoho bitu sudé parity a poslední je jeden stop-bit.

2.3.3 Vnitřní hodinový signál

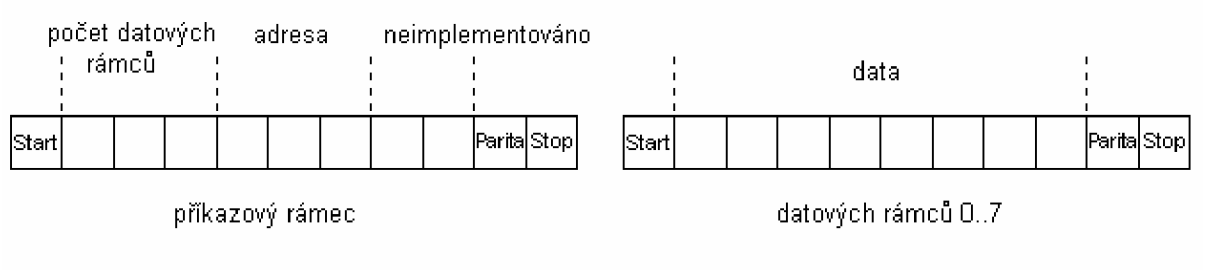
Vnitřní hodinový signál je dělen třikrát oproti hodinovému signálu ve FITkitu (obr. 2.2). Vysílání a porovnání počtu vzorků probíhá vždy s náběžnou hranou vnitřního hodinového signálu komponenty. Toto řešení dostalo přednost před možností, kdy je signál dělen šestkrát a vysílání probíhá jak při náběžné, tak sestupné hraně.



Obr. 2.2: Zobrazení vnitřního hodinového signálu v komponentě RS485_dvr

2.3.4 Protokol

Komunikace probíhá na základě jednoduchého protokolu (obr. 2.3). Na začátku je vyslán příkazový rámec s počtem rámců datových a adresou, které budou následovat. Na počet datových rámců, které budou následovat jsou vyhrazeny první tři bity. Následují tři bity adresy a pak jsou ještě dva bity neimplementovány. U jednoho bitu se naskýtá možnost, kdy bude poslán jen rámec příkazový a sám bude značit vyžádání dat od příjemce. Druhý bit je zcela nevyužitý.



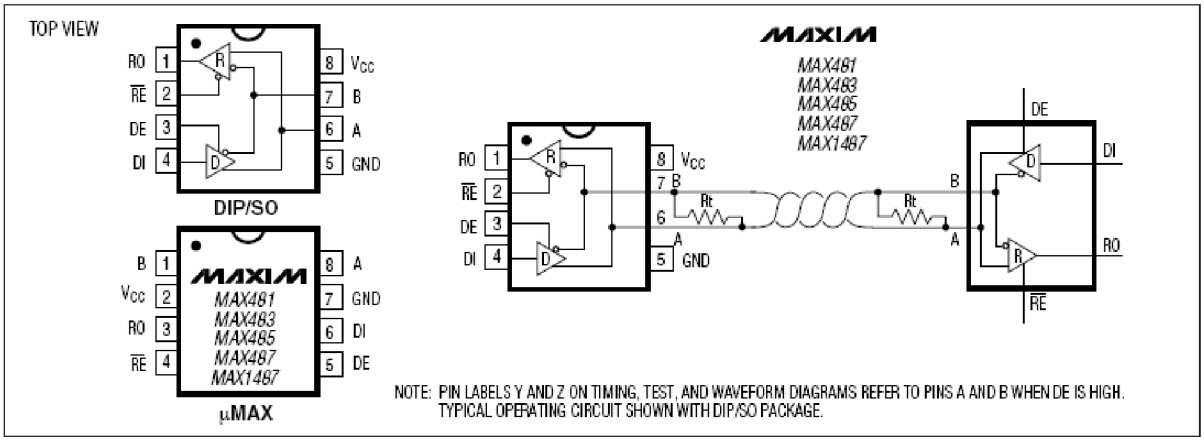
Obr. 2.3: Formát komunikačního protokolu

2.3.5 Příchozí signál

Pro jasné rozpoznání příchozího signálu je tento signál vzorkován při sestupné hraně hodinového signálu FITkitu. To znamená, že máme vždy tři vzorky signálu za jeden takt hodinového signálu komponenty. Převažující logická hodnota příchozího signálu je pak vysílána na vnitřní datový signál.

2.3.6 Převodník napětí

Z důvodů rozeznání napěťové diference signálu při příjmu a pro vysílání diferenciální signálu při propojení FITkitů je nutné použít vnější obvod zajišťující tyto činnosti. Tyto převodníky jsou dostupné v České republice v ceně pod sto korun od více výrobců. Příkladem budiž obvody MAX481 od firmy Maxim či SN75176B od firmy Texas Instruments. Realizace takového propojení by byla časově nenáročná a bylo by ho možno provést do několika dnů. Na obrázku 2.4 je vidět pouzdro a vývody obvodu MAX481 a dalších jemu podobných. Tyto přídavné obvody jsou většinou napájeny napětím 5V a uzemněny na zem. Existují i verze s napájením 3.3V. Mají povolovací vstupy na příjem i vysílání linku,kvůli odpojení od linky. Důvod je, jako i u komponenty *RS485_dvr*, že linka RS-485 je sběrnicového typu a může na ní být více než jeden vysílač. Abychom tuto komponentu připojili k FITkitu, bylo by nutné vyvést signály DATA a HELP_LINK z komponenty *RS485_dvr* na některé z pinů JP9. Ty pak propojit s převodníkem. Signál DATA bychom propojili s piny RO a DI a HELP_LINK s RE a DE na převodníku. Při reálném propojení je obvod nutné doplnit odpory, které jsou zmíněny dříve. Maximální teoretická přenosová rychlost pro převodník MAX481 na krátké vzdálenosti je 2,5Mbit/s.



Obr. 2.4: Piny pro zapojení převodníku MAX481

2.4 Implementace

2.4.1 Celková konfigurace FPGA

Aplikace navržená pro FITkit má obsaženy už hotové části: SPI dekodér, řadič přerušení, řadič displeje, řadič klávesnice a mnou navrhnuté a implementované komponenty *RS485_dvr* (viz. kapitola 2.4.2) a pomocná komponenta pro *RS485_dvr* komponenta *help_component*. Aplikace je postavena na principu získávání dat z klávesnice FITkitu. Ta vyvolá přerušení při zmáčknutí klávesy pro mikrokontroler. Mikrokontrolér převezme znak z klávesnice a pošle ho na první komponentu *RS485_dvr* a na terminál na počítači. Ta znak pošle na druhou komponentu *RS485_dvr*. Mezi nimi se nachází pomocná komponenta *help_component* (viz. kapitola 2.4.8). Druhá komponenta *RS485_dvr* znak přijme, vyvolá přerušení a ten je vzápětí pomocí mikrokontroleru poslán na displej FITkitu. V konečném výsledku by se tedy znak na displeji a na terminálu počítače neměl lišit.

2.4.2 Komponenta RS485_dvr

Je základní komponentou v práci. Obsluhuje příjem dat i jejich vysílání. Přijímač a vysílač nejsou aplikovány jako zvláštní komponenty a obě součásti jsou v *RS485_dvr* dohromady. To dovolilo komunikaci mezi vysílačem a přijímačem, takže pokud pracuje vysílač, přijímač nepracuje a naopak. Tato komponenta zpracovává data ze sériové linky pomocí protokolu, a převádí je na paralelní podobu. V paralelní podobě mohou data být poslána na dekodér SPI, tou je pak poslána k mikrokontroleru. Z mikrokontroleru mohou být data poslána jak do FPGA (displej,ide rozhraní) tak na terminál PC přes USB.

2.4.3 Příjem dat

Pro příjem dat v protokolu je implementován Moorův konečný automat (obr. 2.5). Ten má čtyři stavy. Stav SIdle značí nečinnost linky, ve stavu SCmdRcv se přijímá příkazový rámec. Stav SDataWait je stav vyčkávací na další rámec, který nemusí následovat hned za stop-bitem předešlého rámce. Ve stavu SDataRcv se provádí příjem dat. Automat začíná ve stavu SIdle a čeká na start-bit. Po něm přechází do stavu SCmdRcv kde se přijímá příkazový rámec. Pokud je ověřena správnost rámce, je zapsána hodnota prvních třech bitů do proměnné packcntr a stavový automat přechází do stavu SDataWait. V tomto stavu se očekávají datové rámce, protože ty nemusejí následovat hned v dalším hodinovém taktu. Při příchodu start-bitu přechází automat do stavu SDataRcv. Po přijetí datového rámce se kontroluje velikost proměnné packcntr a pokud značí, že datové rámce jsou přijaty všechny, přechází automat do stavu nečinnosti SIdle. Při packcntr větší než jedna je proměnná dekrementována o jedničku a stav přechází do SDataRcv, kde se očekávají další datové rámce.

2.4.3.1 Stav SIdle

Počáteční stav konečného automatu značící nečinnost linky. Při jakékoli detekci chyby na lince při příjmu, přechází automat do tohoto stavu. Ve stavu SIdle je povoleno vysílat, nastavuje se tu příslušný signál, tím je signál send_en. Automat také po celou dobu vysílání ve stavu SIdle setrvává. Ze stavu SIdle do stavu SCmdRcv automat přechází při příchodu signálu start-bit.

2.4.3.2 Stav SCmdRcv

Příchozí datové bity příkazového rámce se zapisují do posuvného registru shreg_master. Při příchodu stop-bitu je provedena kontrola správnosti rámce, což značí signál pckt_complete v log.„1“. Signál cmd_complete značící první rámec komunikace určuje, zda jsou bity náležející počtu datových rámců (první tři datové bity rámce) překopírovány do proměnné packcntr. Kopírování je provedeno pokud je cmd_complete v log.„0“.

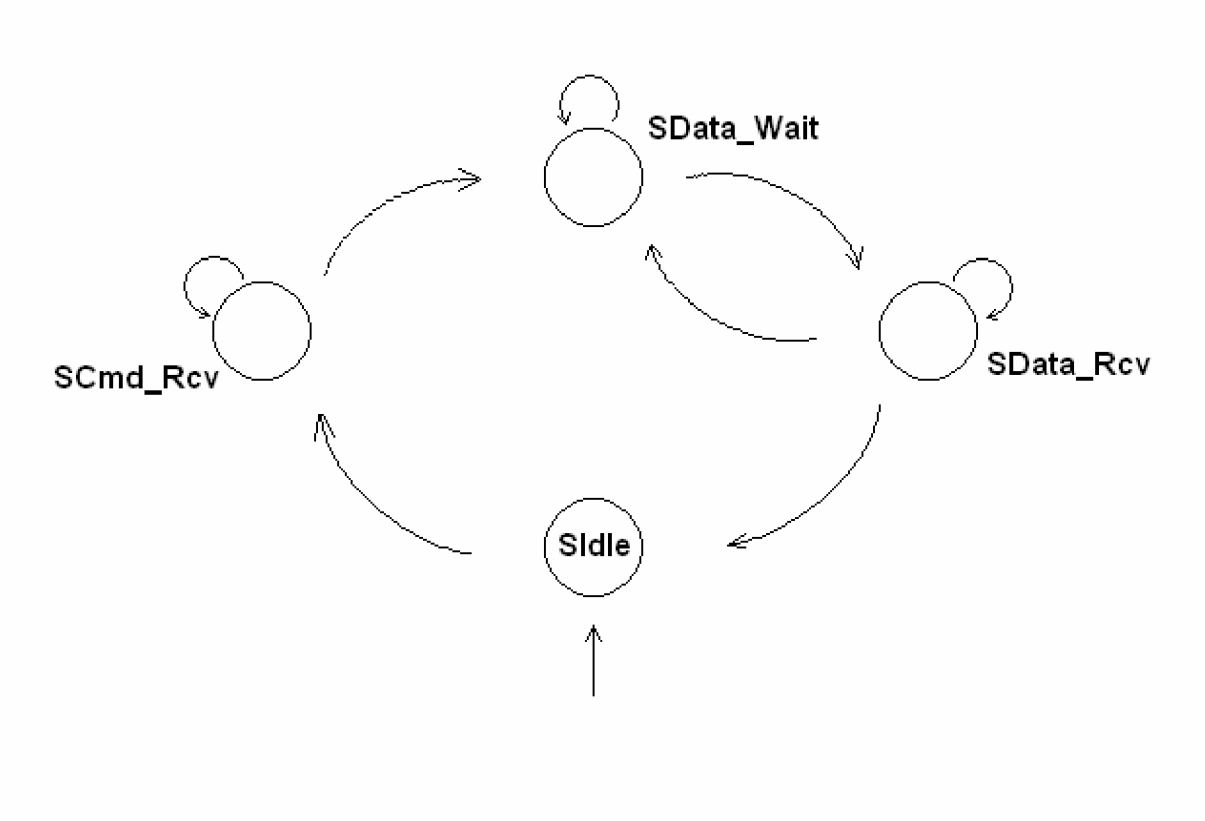
2.4.3.3 Stav SDataWait

Stav je podobný stavu SIdle. Je to také stav vyčkávací. Jen v tomto stavu není povoleno vysílání. Během vysílání se v tomto stavu automat vyskytuje několikrát.

2.4.3.4 Stav SDataRcv

Příjem datových rámců. Jejich počet určuje dříve zmíněná proměnná packcntr. Datové bity, které jsou přijmuty jsou zapisovány do stejného posuvného registru, jako datové bity příkazového rámce. Tím je registr shreg_master. Při přijmutí celého rámce a při úspěšné kontrole správnosti dat je registr překopírován do registru shreg_do. Po přijetí rámce se vyvolá přerušení, port pro přerušení je port DATA_VLD komponenty *RS485_dvr*. Registr shreg_do je vystaven na port DATA_OUT pokud je

na portu WRITE_EN log.„1“. Při přijmutí posledního rámce stavový automat přechází zpět do stavu SIdle.



Obr. 2.5: Diagram stavového automatu

2.4.4 Kontrola správnosti dat

Při každém přijetí rámce probíhá kontrola správnosti tohoto rámce. Rámec je platný pokud souhlasí paritní bit s výpočtem sudé parity dat přijatého rámce a zároveň musí mít stop-bit logickou hodnotu „1“. Pokud se tak nestane, stavový automat přechází do stavu nečinnosti (stav SIdle). A čeká na další rámec. V případě kladného posouzení správnosti rámce je příslušný signál pckt_complet nastaven na log.„1“.

2.4.5 Vysílání dat

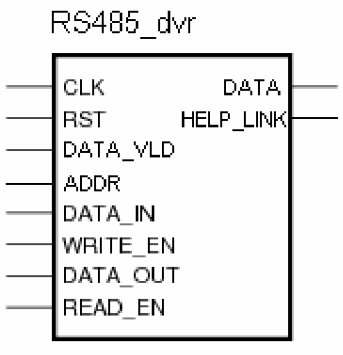
Vysílání dat je řízeno pouze procesem a není prováděno stavovým automatem. Pokud jsou data vystavena na vstup do komponenty, jsou nakopírována do registru shreg_di, do kterého se doplní start-bit, data, paritní bit a stop-bit. Zároveň by měla být vystavena i adresa, z té je vypočítána její sudá parita. Adresa, parita a stop_bit jsou nakopírovány do registru shreg_cmdo, v kterém je připraven příkazový rámec na odeslání. V této implementaci vysílač vždy předpokládá, pouze jeden datový rámec. Vysílání tedy začíná vždy, jestliže jsou data zapsána na vstup a signál send_en je v logické „1“. Pokud v čase žádosti není vysílání povoleno (probíhá příjem dat), je kontrola volnosti

linky prováděna každý takt vnitřních hodin. Po uvolnění linky vysílač přechází ze stavu vysoké impedance na vysílání a v zápětí je odeslán jeden rámec příkazový, který je následován hned jedním rámcem datovým. Po odvysílání dat přechází vysílač zpět do stavu vysoké impedance.

2.4.6 Důležité vnitřní signály komponenty RS485_dvr

- data_inner: vnitřní datový signál, značí hodnotu signálu,který převažoval tři minulé hodinové tiky. Z tohoto signálu se určují signály jako start_bit, stop_bit a další. Z důvodů vzorků je vnitřní hodinový signál o jeden tik vnitřních hodin zpožděný.
- clk_inner: vnitřní hodinový signál, dva tiky vnějších hodin je v log.„1“ a jeden tik je v log. „0“
- link_idle: signál pro indikaci nečinnosti linky
- link_activ: při tomto signálu je aktivní vysílaní. Vysílač není ve vysoké impedanci.
- pstate a nstate: signály stavového automatu. pstate je aktuální stav a pstate je stav příští
- data_cnt: počitadlo pro zjištění počtu vzorků v log.„1“ během jednoho intervalu vnitřních hodin
- bitcntr: proměnná počitadla pro zjištění počtu bitů v rámci, které byly přijaty či odeslány. Pokud je linka ve stavu nečinnosti, je proměnná rovna 0. V režimu příjmu počitadlo počítá od 0 do 10, kdy při start-bitu proměnná není inkrementována. Při odesílání se situace mírně liší. Start-bit se chová jako všechny ostatní bity a proměnná je i při jeho vysílání inkrementována. Při vysílání proměnná nabývá hodnot od 1 do 11.
- pakcntr: Proměnná značící počet datových rámců do konce přenosu. Nastavuje se při příjmu příkazového rámce.
- start_bit: Signál pro indikaci start-bitu, aktivní v log.„1“
- stop_bit: Signál pro indikaci stop-bitu, aktivní v log.„1“
- even_parity_chck: Potvrzení správnosti sudé parity přenesených dat a parity uložené v rámci.
- pckt_complete: Při přijmutí celého rámce tento signál značí, zda je celý rámec bez chyby (sudá parita a stop-bit).
- send_activ: Signál označující aktivitu odesílání.

2.4.7 Popis rozhraní komponenty RS485_dvr



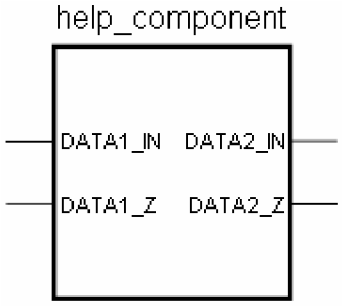
Obr. 2.6: Rozhraní komponenty RS485_dvr

- CLK: vstup hodinového signálu
- RST: vstup pro reset
- DATA_VLD: výstup pro připojení na řadič přerušení, vyvolá přerušení při příjmu rámce
- ADDR, DATA_IN, WRITE_EN, DATA_OUT, READ_EN jsou signály, které se připojí k řadiči SPI, avšak kromě ADDR jsou signály brány ze strany komponenty. Na dekodér se připojí v převráceném pořadí. Signály se připojí následovně: DATA_IN na DATA_OUT, WRITE_EN na READ_EN, DATA_OUT na DATA_IN a READ_EN na WRITE_EN.
- DATA: signál vstupu a výstupu sériové komunikace.
- HELP_LINK: signál pro signalizaci zda je vysílač ve vysoké impedanci. Logická „1“ znamená stav vysoké impedance.

2.4.8 Pomocná komponenta help_component

Komponenta je navrhnuta pro aplikaci, kdy na jednom čipu FPGA na FITkitu budou dvě komponenty RS-485 komunikovat mezi sebou. Z důvodu, že linka RS-485 je sběrnicového typu a všechny vysílače pokud nevysílají se odpojí, je na vodiči mezi komponenty nedefinovaný stav. V reálném propojení se tato situace vyřeší pomocí posilovacích odporů. V FPGA je tento stav dosažen komponentou *help_component*. Každá komponenta RS-485 má pomocný vývod HELP_LINK, který určuje, zda je vysílač ve vysoké impedanci či nikoliv. Tento vývod také potřebují převodníky napětí z TTL na napětí linky RS-485, které vysílače také odpojují. Pokud obě komponenty *RS485_dvr* vysílají signál, že jejich vysílače jsou ve vysoké impedanci, je na komunikační linku mezi těmito komponentami vyslán pomocí komponenty *help_component* signál označující nečinnost linky. V případě komponenty *RS485_dvr* je to log.„1“.

2.4.9 Popis rozhraní komponenty help_component



Obr. 2.7: Rozhraní komponenty help_komponent

DATA1_IN, DATA2_IN: Vstup výstupní porty pro sériovou komunikaci.

DATA1_Z, DATA2_Z: Vstupy pro pomocné signály HELP_LINK z *RS485_dvr*

2.5 Simulace

Simulace je důležitou součástí pro návrh hardwaru. Simulace nám vysvětluje jak se daná komponenta v reálném čipu chová. Simulace mnou navrhnutá simuluje předání celého jednoho packetu (jeden rámec příkazový a jeden datový). Je simulována ta část aplikace, kterou jsem navrhoval a implementoval . Jsou to tedy dvě komponenty *RS485_dvr* propojené sériovou komunikační linkou (1 bitový signál) přes pomocnou komponentu *help_component* , která určuje stav na lince při odpojení obou vysílačů od linky.

3 Závěr

Vysílač je pro naše účely mírně zjednodušený. Na provoz aktuální aplikace postačuje, když vysílač nemá generické parametry ohledně délky dat. Spokojili jsme se s pevnou šířkou 8 bitů, které budou posílány a také přijímány. Je to zjednodušení, které je v budoucnu vhodné odstranit a doplnit vysílač na možnost odesílat maximální délku dat danou protokolem, tedy až sedm datových rámců, každý po osmi bitech. Přijímač je navrhován a implementován na přijetí celého množství dat. Jen není implementován žádný buffer a po každém přijetí datového rámce je vyvoláno přerušení. S ohledem na možnou vzdálenost komunikace lze pro větší bezpečnost také do protokolu zahrnout kontrolu celých dat. Řešením by bylo vyhradit jeden datový rámec jen pro účely kontrolního součtu. Velmi používanou metodou je pro tyto účely je CRC, cyklický redundantní součet. Je to druh hashovací funkce, která se provádí před odesláním dat a připojuje se k datům. Provedení spočívá ve dvojkovém vydělení dat bez přenosu, používá se logická funkce XOR. Čím více bitů je na kontrolní součet vyhrazeno, tím je větší pravděpodobnost, že se chyba odhalí. U CRC je také důležitý generální polynom s kterým se funkce XOR uskutečňuje. Pro různá média jsou vhodné jiné polynomy. Ty lze najít v literatuře. Jsou zmíněny i v ODKAZ NA WIKI. V našem případě při vyhrazení jednoho datového rámce by se používalo CRC-8. Mezi nejčastěji používané CRC patří délky 8,16,32 a 64 bitů. Sudá či lichá parita je také vlastně CRC o délce jednoho bitu, jde však pouze o nejjednodušší možnost.

Tato práce rozšiřuje soubor aplikací, které jsou pro FITkit volně přístupné. Zprostředkovává komponentu pro FPGA a rozhraní, které je poměrně časté v komunikaci v průmyslu na větší vzdálenosti. Je vhodné se seznámit i s technologiemi, se kterými se student na škole v praktickém použití spíše nesetká. Avšak v budoucím zaměstnání je pravděpodobnost setkání s touto linkou poněkud vyšší. Při zakoupení převodníků lze s touto linkou pracovat a testovat ji, jak by pracovala v reálných podmínkách.

Literatura

[1] HLAVA, J.: Prostředky automatického řízení. Praha, ČVUT, 2000,s. 100-112. Dokument dostupný na URL http://www.fm.vslib.cz/~krtsub/fm/par/Skripta_PAR.pdf (květen 2007)

[2] FITkit. Dokument dostupný na URL www.fit.vutbr.cz/FITkit (květen 2007)

[3] EIA-485. Dokument dostupný na URL http://en.wikipedia.org/wiki/RS-485 (květen 2007)

[4] Serial communications. Dokument dostupný na URL http://en.wikipedia.org/wiki/Serial_communication (květen 2007)

[5] Cyclic redundancy check. Dokument dostupný na URL http://en.wikipedia.org/wiki/Cyclic_redundancy_check (květen 2007)

[6] Poucha, P.: Přenos dat po linkách RS485 a RS422. Dokument dostupný na URL http://www.hw.cz/ART705-Prenos-dat-po-linkach-RS485-a-RS422.html (květen 2007)

[7] Staněk, J.: RS 485 & 422. Dokument dostupný na URL http://www.hw.cz/Dokumentace/Teorie-a-praxe/ART821-RS-485-%26amp%3B-422.html (květen 2007)

[8] Kolář, M.: VHDL. Dokument dostupný na URL http://www.fm.vslib.cz/~kes/pages/nove_predmety/nhk/Prednasky/nhk_08.html 2003 (květen 2007)

[9] Ashenden, P. J.: The Student's Guide to VHDL. San Francisco, 1998, ISBN 1-55860-520-7

[10] RS-232. Dokument dostupný na URL http://en.wikipedia.org/wiki/RS-232 (květen 2007)

[11] Holeček, O.: CRC (kontrolní součet). Dokument dostupný na URL http://www.root.cz/clanky/crc-kontrolni-soucet/ (květen 2007)

[12] Kolouch, J.: Jazyk VHDL a jeho užití při syntéze číslicových systémů. Dokument dostupný na URL http://www.urel.feec.vutbr.cz/~kolouch/pld/2_cviceni/kapitola03_01.html (květen 2007)

[13] IEEE Standard 1076, VHDL

[14] datasheet Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers. Maxim, Dokument dostupný na URL http://datasheets.maxim-ic.com/en/ds/MAX1487-MAX491.pdf (květen 2007)

Seznam příloh

Příloha 1. Manuál k aplikaci.

Příloha 2. Zdrojové texty aplikace.

Příloha 3. CD s aplikací a návodem na spuštění.

Příloha č.1:

Manuál k simulaci posílání dat přes linku RS-485.

Simulace simuluje celé FPGA na FITkitu při přenesení dvou rámců mezi dvěmi komponentami komunikujícími přes linku RS-485.

Použité soubory v této aplikaci:

FPGA

units/spi_ics/spi_ctrl.vhd	-- SPI - spi control
units/spi_ics/spi2adc.vhd	-- SPI - spi decoder
ctrls/interrupt/interupt.vhd	-- radic preruseni
ctrls/keyboard_4x4/keyboard_4x4.vhd	-- radic klavesnice 4x4
ctrls/keyboard_4x4/keyboard_4x4_int.vhd	-- radic klavesnice 4x4 s prerusenim
ctrls/lcd/lcd_ctrl.vhd	-- radic LCD
chips/fpga_xc3s50.vhd	-- definice rozhrani FPGA
chips/fpga_xc3s50.ucf	-- definice prirazeni pinu k
	-- jednotlivym signalum
apps/RS_485/top/top_level.vhd	-- definice aplikace (vyskladane komponenty)
apps/RS_485/top/RS485.vhd	-- transciever pro linku RS-485
apps/RS_485/top/Help_component.vhd	-- pomocná komponenta definující
	-- log. úroveň linky RS-485 v klidu

Řadiče pro LCD, klávesnici a přerušení jsou v aplikaci zahrnuty pro budoucí použití pro FITkit.

Pro simulaci je použit test bench v souboru rs485_tb3.vhd v adresáři apps/RS_485/top/sim.

Zprovoznění simulace:
Nejprve si stáhněte či udělejte update SVN ze stránek www.fit.vutbr.cz/kit.

Nahrejte adresář RS_485 do adresáře apps.
Spusťte simulaci pro posílání znaků po lince RS-485:

```
cd  apps/RS_485/top
make sim
```

Následuje spuštění ModelSimu, jsou vidět signály SPI controleru, SPI dekodér a dvou komponent RS-485. Jedna vysílá data, která přijme od dekodéru SPI. Druhá komponenta RS-485 data přijímá. Po dokončení příjmu v druhé komponentě simulace končí.

Příloha č.2:

Soubor RS485.vhd (licence vynechána):

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity RS485_dvr is
    port (
      -- Synchronizace
      CLK      : in  std_logic;
      RST      : in std_logic;
      --rozhrani pro komunikaci
      DATA       : inout std_logic:='Z';       --komunikacni linka
      HELP_LINK: out std_logic:='1';
      DATA_VLD : out  std_logic;               --port pro preruseni

          --Datove rozhrani
      ADDR     : in  std_logic_vector (2 downto 0);

      DATA_OUT : out  std_logic_vector (7 downto 0);
      DATA_IN  : in   std_logic_vector (7 downto 0);

       WRITE_EN : in  std_logic;    --poslani dat do MCU pres
DATA_OUT
      READ_EN  : in  std_logic     --cteni z MCU pres DATA_IN
   );
end RS485_dvr;

architecture basic of RS485_dvr is
```

```
subtype tcntr is integer range 0 to 2;
signal  clk_cnt: tcntr :=0;

subtype dcntr is integer range 0 to 3;
--pocitadlo jednicek pro rozhodnuti vnitrniho signalu

signal  data_cnt: dcntr:=0;

subtype bcntr is integer range 0 to 11;
signal  bitcntr : bcntr :=0;

subtype pcntr is integer range 0 to 7;
signal  pakcntr : pcntr :=0;


signal  clk_inner : std_logic;       --vnitrni hodinovy signal RS485
signal  link_idle:  std_logic;                    --necinnost linky
signal  cmp_11bits: std_logic;              --aktivace pri rovnosti
                                       --poctu bitu ramce a bitcntr
signal  packet_all: std_logic;
signal  data_inner: std_logic;           --navzorkovana vnitrni data
signal  bitcntr_rst:std_logic;
signal  start_bit:  std_logic;
signal  stop_bit:   std_logic;
signal  even_parity_chck:std_logic;
                                  --potvrzeni spravnosti sude parity
signal  even_parity_addr:std_logic;             --suda parita adresy
signal  parity_help:std_logic;
signal  cmd_complete:std_logic;         --odeslani prikazoveho ramce
signal  pckt_complete:std_logic;
signal  pckt_complete_h:std_logic;

type    FSMstate is (SCmdRcv, SDataRcv, SIdle,SDataWait);--stavy
vnitrniho automatu
signal  pstate    : FSMstate;                    -- aktualni stav
signal  nstate    : FSMstate;                    -- dalsi stav
```

```
signal  shreg_do:   std_logic_vector(7 downto 0);  --registr pro
                                                -- vystup prijatych dat
signal  shreg_di:   std_logic_vector(10 downto 0); --registr pro
                                            -- data, ktera se budou vysilat
signal  shreg_cmdo: std_logic_vector(10 downto 0); --registr pro
                                --prikazovy ramec, ktery se bude vysilat
signal  shreg_master:std_logic_vector(7 downto 0); --univerzalni
                                                --registr pro prijem

signal  cmdreg_en_h:std_logic;
--signal  direg_en:   std_logic;
signal  send_en:    std_logic;                  --povoleni vysilani
signal  send_req:   std_logic;                   --zadost o prenos
signal  send_req_h: std_logic;                --posunuti o jednu dobu
signal  send_activ: std_logic;                    --aktivni posilani
signal  send_cmd:   std_logic;       --zda se poslal prikazovy ramec
signal  send_complete:std_logic;              --odesilani kompletni
signal  send_end:   std_logic;
signal  send_end_h: std_logic;
signal  send_complete_res:std_logic;
signal  packet_all_rst:std_logic;             --prijaty vsechny ramce
begin
   --comparators
cmp_11bits  <= '1' when (bitcntr = 10) else '0';
parity_help <= (shreg_master(7) xor shreg_master(6)xor
shreg_master(5)xor shreg_master(4)xor shreg_master(3)xor
shreg_master(2)xor shreg_master(1)xor shreg_master(0)) when
(RST='0') else '0';--suda parita
stop_bit    <= '1' when (bitcntr = 10)and(data_inner='1') else '0';
   --ports
DATA_VLD    <= packet_all_rst;
DATA_OUT<=shreg_do when(WRITE_EN='1')else (others=>'0');
                                                    --cteni z RS485
--generovani vnitrnich hodin a vzorkovani signalu
clk_inner_GEN:process(CLK,RST)
  begin
   if(RST='1')then
```

```
      clk_cnt<=0;
        data_cnt<=2;
        clk_inner<='0';
        data_inner<='1';
       elsif (CLK'event)and (CLK='1')then
         if(clk_cnt=0) then
           if(data_cnt>1)then
                              --rozhodnuti signalu podle poctu vzorku
               data_inner<='1';
           else
               data_inner<='0';
           end if;
         data_cnt<=0;
         end if;
            if(clk_cnt=2)then
              clk_cnt<=0;
              clk_inner<='0';
         else
              clk_cnt<=clk_cnt+1;
              clk_inner<='1';
            end if;
       end if;
    if (CLK='0')and(DATA='1')and(RST='0') then  --vzorkovani signalu
         data_cnt<=data_cnt+1;
       end if;
end process;

--vytvoreni sude parity adresy
process(ADDR,RST)
  begin
    even_parity_addr<=not(ADDR(2) xor ADDR(1) xor ADDR(0));
end process;

data_registers:process(CLK,RST)
  begin
    if(RST='1')then
        send_req<='0';
```

```
        send_req_h<='0';
       elsif (CLK='1')and(READ_EN='1')  then
      send_req<='1';
       elsif (READ_EN='0')and (send_req='1')  then
         send_req<='0';
            send_req_h<='1';
       elsif(send_req_h='1')and (send_activ='1')then
         send_req_h<='0';
       end if;
end process;

data_sending_reg:process(RST,send_req_h,send_en,clk_inner,send_compl
ete)
  begin
       if(RST='1')then
         send_activ<='0';
      send_end<='0';
       elsif (send_req_h='1')and(send_en='1')  then
                                        --cekani na povoleni posilani
        send_activ<='1';
                                        --aktivace posilani dat
       elsif(send_complete='1')and(send_activ='1')then
      send_activ<='0';
           send_end<='1';
       elsif(send_end='1')and(clk_inner='0')then
         send_end<='0';
       end if;
end process;

data_sending_complete_rst:process(RST,clk_inner)
  begin
    if(RST='1')then
      send_complete_res<='0';
       elsif(send_complete_res='1')then
          send_complete_res<='0';
       elsif(clk_inner='0')and(send_complete='1')then
       send_complete_res<='1';
```

```
    end if;
end process;

--posilani dat
data_sending:process(RST,cmdreg_en_h,send_req,send_end,send_complete
_res)
  begin
    if(RST='1')then
           send_cmd  <='0';
           send_complete<='0';
           send_end_h<='0';
           shreg_di  <=(others=>'0');
      shreg_cmdo<=(others=>'0');
      elsif (send_complete_res='1')then
        send_complete<='0';
      elsif (send_end_h='1')then
        send_cmd<='0';
          send_complete<='1';
          send_end_h<='0';
      elsif(send_req='1')then
      shreg_cmdo<= "0001" & ADDR & "00" & even_parity_addr & '1';
      shreg_di<= '0'& DATA_IN & (DATA_IN(7)xor DATA_IN(6)xor
DATA_IN(5)xor DATA_IN(4)xor DATA_IN(3)xor DATA_IN(2)xor
DATA_IN(1)xor DATA_IN(0)) & '1';--even_parity
      elsif(send_end='1')then
          DATA<='Z';
            HELP_LINK<='1';
      elsif(send_activ='1')then
       HELP_LINK<='0';
           if (send_cmd='0')  then
        DATA<=shreg_cmdo(10);
             shreg_cmdo<=shreg_cmdo(9 downto 0)& '0';
             if(bitcntr=11)then
               send_cmd<='1';
             end if;
           else
             DATA<=shreg_di(10);
```

```
            shreg_di<=shreg_di(9 downto 0)& '0';
              if(bitcntr=11)then
                   send_end_h<='1';
              end if;
             end if;
     end if;
 end process;

parity: process(RST,cmdreg_en_h,start_bit)
  begin
    if(RST='1')or(start_bit='1')then
        even_parity_chck<='0';
      elsif (bitcntr=9)then
        if (parity_help=data_inner)then
          even_parity_chck<='1';--kontrola parity
        else
          even_parity_chck<='0';
        end if;
      end if;
end process;

end_recieving: process(RST,clk_inner)--potvrzeni konce uspesneho
prijmu
  begin
    if(RST='1')then
        packet_all_rst<='0';
      elsif (packet_all='1' and pstate=SIdle) then
          packet_all_rst<='1';
      elsif(packet_all_rst='1')and(clk_inner='1')then
        packet_all_rst<='0';
      end if;
end process;

-- kontrola prijmu celeho ramce
complete: process(RST,cmp_11bits,start_bit,link_idle,packet_all_rst)
  begin
    if(RST='1')or(start_bit='1')or(packet_all_rst='1')then
```

```
      if(start_bit='0')then
          pckt_complete_h<='0';
            cmd_complete<='0';
          end if;
          if(RST='1')then
            shreg_do  <= (others=>'0');
          end if;
          packet_all<='0';
          pckt_complete <= '0';
       elsif (link_idle='1')then
         cmd_complete<='0';
       elsif (cmp_11bits='1') then
       if( even_parity_chck='1')and(stop_bit='1')then
             pckt_complete <= '1';
             if(cmd_complete='0')then
               pakcntr<=conv_integer(shreg_master(7 downto 5));
               cmd_complete<='1';
             else
               shreg_do <= shreg_master;
               if(pakcntr>1)then
                 pakcntr<=pakcntr-1;
               elsif(pakcntr=1)then
                 packet_all<='1';
               end if;
             end if;
          else
             pckt_complete <= '0';
         end if;
         pckt_complete_h<=not pckt_complete_h;
       end if;
end process;

-- pocitadlo bitu
bitecntr: process( bitcntr_rst, clk_inner,RST)
   begin
      if(RST='1')then
             cmdreg_en_h<='0';
```

```
            elsif (clk_inner'event) and (clk_inner='1') then
            if ( bitcntr_rst='1') then
            bitcntr <= 0;
               else
            if(bitcntr=11)then
                     bitcntr <= 1;
                  else
                      bitcntr <= bitcntr + 1;
                  end if;
                  cmdreg_en_h<=not cmdreg_en_h;
        --pomocny signal pro signalizaci zmena pocitadla bitu v ramci
         end if;
            end if;
   end process;

   -- rozeznani start bitu
startbit: process(clk_inner,RST)
  begin
    if(RST='1')then
        start_bit<='0';
      elsif(clk_inner'event) and (clk_inner='1')then
        if(start_bit='1')then
            start_bit<='0';
           elsif (data_inner='0')and( bitcntr_rst='1') then
         start_bit <= '1';
           end if;
      end if;
end process;

-- nahrani dat do posuvneho registru shreg_master ze navzorkovanych
dat data_inner
shift:process (RST, cmdreg_en_h)
   begin
       if (RST = '1') then
       shreg_master <= (others=>'0');
     elsif (clk_inner='1')then
       if(bitcntr<9)then
```

```
              shreg_master <= shreg_master(6 downto 0) & data_inner;
            end if;
        end if;
end process;


--FSMachine
FSM:process(clk_inner, RST)
  begin
    if (RST = '1') then
      pstate <= SIdle;
    elsif (clk_inner='1') and (clk_inner'event) then
      pstate <= nstate;
    end if;
end process;

FSMlogic:process (pstate,start_bit,pckt_complete_h,send_activ)
  begin
    bitcntr_rst <= '0';
    link_idle<='0';
    send_en<='0';
    nstate <= SIdle;

       case pstate is

      --link is idle
      when SIdle =>
        if (start_bit = '1') then
          nstate <= SCmdRcv;
        else
          if(send_activ='0')then
                 bitcntr_rst <= '1';
            send_en<='1';
            link_idle<='1';
                 end if;
             end if;
```

```
--command receive
 when SCmdRcv =>
   nstate <= SCmdRcv;
   if (cmp_11bits = '1')then
     if (pckt_complete='1') then
       nstate <= SDataWait;
            else
       nstate <= SIdle;
       link_idle<='1';
            end if;
     bitcntr_rst <= '1';
   end if;

--waiting for data
 when SDataWait =>
   if (start_bit = '1') then
     nstate <= SDataRcv;
   else
     nstate <= SDataWait;
     bitcntr_rst <= '1';
   end if;

--DATA receive
 when SDataRcv =>
   nstate <= SDataRcv;
   if (cmp_11bits = '1')then
     if (pckt_complete='1') then
       if(packet_all='1')then
         nstate<=SIdle;
         link_idle<='1';
       else
         nstate <= SDataWait;
            end if;
     else
       nstate <= SIdle;
       link_idle<='1';
            end if;
```

```
          bitcntr_rst <= '1';
        end if;
     end case;
   end process;

end basic;
```

Soubor Help_component(licence vynechána):

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity help_component is
    port (
         --RST      : in std_logic;
         --CLK      : in std_logic;
         DATA1_IN  : inout std_logic:='Z';
         DATA2_IN  : inout std_logic:='Z';
         DATA1_IN_Z : in std_logic;
         DATA2_IN_Z : in std_logic
   );
   end help_component;

  architecture base of help_component is
  begin

 DATA1_IN <= '1' when (DATA1_IN_Z ='1' and DATA2_IN_Z ='1')else
        DATA2_IN when (DATA1_IN_Z ='1' and DATA2_IN_Z ='0')else 'Z';
 DATA2_IN <= '1' when (DATA1_IN_Z ='1' and DATA2_IN_Z ='1')else
        DATA1_IN when(DATA2_IN_Z ='1' and DATA1_IN_Z ='0')else 'Z';

end base;
```