VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

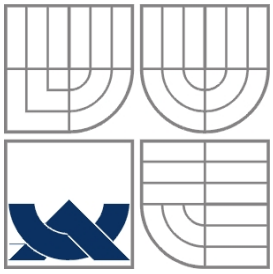
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

VYSÍLAČ SÉRIOVÝCH DAT VE FORMÁTU RS-232

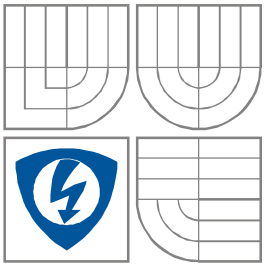
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MARTIN PAUL
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

VYSÍLAČ SÉRIOVÝCH DAT VE FORMÁTU RS-232

RS-232 SERIAL DATA TRANSMITTER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Martin Paul

VEDOUCÍ PRÁCE doc. Ing. Jaromír Kolouch, CSc.
SUPERVISOR

BRNO, 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Martin Paul
Bytem: Optiky 3, Přerov, 750 00
Narozen/a (datum a místo): 21. února 1983 v Přerově

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

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

   ☐ disertační práce
   ☐ diplomová práce
   ☒ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

   Název VŠKP: Vysílač sériových dat formátu RS-232
   Vedoucí/ školitel VŠKP: doc. Ing. Jaromír Kolouch, CSc.

   Ústav: Ústav radioelektroniky
   Datum obhajoby VŠKP: _____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

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.

* hodící se zaškrtněte

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 pořizovaní 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: 3. června 2008

……..	……
Nabyvatel	Autor

Abstrakt

Cílem této bakalářské práce je navržení vysílače sériových dat formátu RS-232 ve vývojovém prostředí Xilinx ISE WebPACK s možností nastavit detaily formátu vysílaných dat vstupními signály. Dále je nutné prozkoumat požadavky navrženého vysílače na spotřebu strukturních prvků v různých cílových obvodech PLD a na závěr provést ověření funkčnosti návrhu pomocí vývojového kitu.

Klíčová slova

sériové rozhraní, vysílač sériových dat formátu RS-232, VHDL, PLD, obvody FPGA

Abstract

The aim of my bachelor thesis is to create an RS-232 transmitter in the Xilinx ISE WebPACK development system with the possibility to set the details of transmitted data by input signals. It is necessary to explore demands of the designed transmitter in terms of consumption of the structural components in the PLD network. The function test of the project was done by the help of development kit.

Keywords

serial interface, RS-232 serial data transmitter, VHDL, PLD, FPGA

Citace

PAUL, M. *Vysílač sériových dat ve formátu RS-232*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 38 s. Vedoucí bakalářské práce doc. Ing. Jaromír Kolouch, CSc.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Vysílač sériových dat formátu RS-232 jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 3. června 2008 .....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Doc. Ing. Jaromíru Kolouchovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 3. června 2008 .....
podpis autora

Obsah

1 Úvod

Tématem této bakalářské práce je vysílač sériových dat formátu RS-232.[1] Úkolem je vytvořit projekt vysílače dat tohoto formátu ve vývojovém prostředí Xilinx ISE WebPACK s možností nastavit detaily formátu vysílaných dat vstupními signály. Dalším úkolem je prozkoumání požadavků navrženého vysílače na spotřebu strukturních prvků v různých cílových obvodech PLD, pro něž daný návrhový systém umožňuje implementaci. Na závěr bylo nutné ověřit funkčnost návrhu pomocí vývojového kitu.

Rozhraní RS-232 je plně duplexní, avšak v našem případě budeme data pouze vysílat, nikoli přijímat, jedná se tedy o simplexní spojení.

Práce je rozdělena do šesti kapitol, které jsou zaměřeny na teoretické i praktické provedení.

V první kapitole se zaměřuji na teoretické znalosti, konkrétně na možnosti sériového přenosu dat formátem RS-232, dále se věnuji elektrickým parametrům formátu přenosu a řídícím signálům.

V druhé kapitole se zabývám požadavky na vysílač dat formátu RS-232 a jeho návrhem.

Třetí kapitola se věnuje jazyku VHDL, který používám k naprogramování vysílače dat. Tato kapitola také obsahuje ukázky nejdůležitějších částí vysílače.

Ve čtvrté kapitole jsou zmíněny vývojové softwarové i hardwarové prostředky, které byly během realizace vysílače použity. U hardwaru jsou řešeny jednotlivé problémy spojené s realizací propojení. Mezi ně patří například popis výukové vývojové desky Starter kit a jejich jednotlivých částí, které budou nezbytné pro realizaci vysílače.

V další kapitole jsou řešeny požadavky našeho vysílače na spotřebu strukturních prvků. K tomu byly použity dvě často používané řady, a to *9500XL* a *XC2 (CoolRunner-II).*

Poslední kapitola nás seznamuje s postupem při ověření funkčnosti návrhu naprogramovaného vysílače dat formátu RS 232.

Pro realizaci vysílače byla použita vývojová deska Starter kit od firmy Digilent, která je na uvedené téma vhodná.

Cílem této bakalářské práce je navrhnout konfiguraci pro FPGA umožňující vysílání dat pomocí rozhraní RS-232 .

[1] Dále jen vysílač dat, nebo stručněji vysílač.

2 Sériový přenos dat

Jako sériový přenos označujeme takový přenos dat, kdy se signálové prvky téhož datového proudu předávají za sebou - sériově. Naopak při paralelním přenosu se určitý počet signálových prvků (např. bitů) přenáší současně, [3].

2.1 Synchronní a asynchronní sériový přenos

Při přenosu informace (sériovém i paralelním) musíme zabezpečit, aby přijímač správně vyhodnotil okamžiky platnosti jednotlivých značek generovaných vysílačem. Vysílač a přijímač proto musí být spolu nějakým způsobem časově synchronizovány. Právě podle druhu synchronizace rozlišujeme sériový přenos na synchronní a asynchronní.

Synchronní přenos - se děje pomocí izochronního signálu, tedy takového, kde odstup dvou libovolných charakteristických okamžiků (např. začátků a konců jednotlivých značek) je celistvým násobkem určitého (apriorně daného) jednotkového intervalu. Komunikační kanál je taktován společným hodinovým signálem (vedeném zvlášť nebo obsaženém v datovém signálu), který vymezuje intervaly platnosti jednotlivých značek.

Synchronní přenos se nejčastěji používá u bitově orientovaných protokolů, kde se informace seskupuje do rámců. V datových komunikacích se používá zejména pro přenos větších objemů dat.

Asynchronní (správněji arytmický[2]) přenos [3] - vysílač a přijímač nemají společný hodinový signál, který by vymezoval intervaly platnosti značek. Namísto toho mají obě strany své vlastní hodiny, dostatečně přesné, aby se po fázovém zasynchronizování mohly po několik značkových intervalů považovat za izochronní. Jelikož je třeba hodiny pravidelně synchronizovat, používá se tento způsob nejčastěji pro přenos krátkých bitových posloupností, znaků (5,6,7 nebo 8 bitů). Synchronizace probíhá před každým znakem, i když prvnímu významovému bitu vždy předchází tzv. start bit, jenž je reprezentován opačnou hodnotou signálu, než je klidová úroveň na lince a trvá stejnou dobu jako následující bitové intervaly. Arytmický přenos je ze své podstaty vhodný např. pro komunikaci s počítačem prostřednictvím terminálu, [3].

---

[2] Poznámka: arytmický přenos je kombinací přenosu synchronního a asynchronního - jednotlivé znaky mohou započít kdykoli, avšak po odvysílání start bitu do konce znaku má signál isochronní charakter.

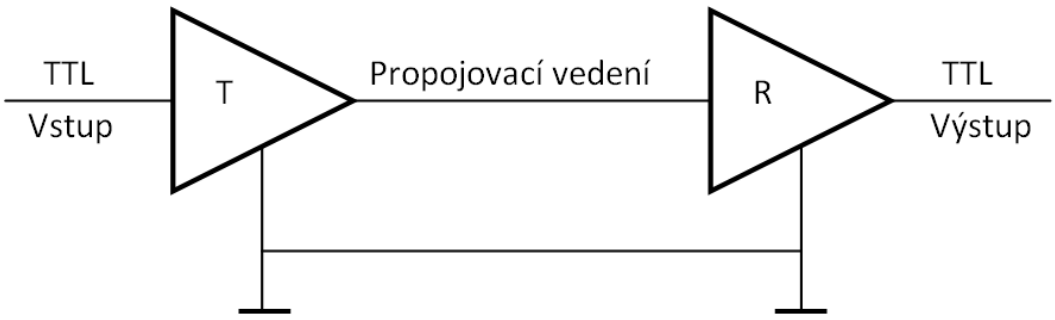
[3] Poznámka: "čistě" asynchronní přenos se děje prostřednictvím anizochronního signálu, kdy bitové intervaly obecně nemají stejnou délku; Prakticky to může být realizováno například 3-stavovým signálem.

2.1.1 Sériové komunikační rozhraní TIA/EIA 232 F

U číslicových regulátorů a číslicových zařízení vůbec je styk s okolím umožňující vytváření složitějších struktur řídicích systémů řešen pomocí sériových komunikačních rozhraní. V nejjednodušším případě se jedná o rozhraní dovolující pouze dvoubodovou komunikaci mezi jedním přijímačem a jedním vysílačem (point to point communication). V této souvislosti nelze opomenout sériové komunikační rozhraní TIA/EIA 232 F. První varianta tohoto rozhraní byla americkou organizací EIA (Electronic Industries Association) definována již v roce 1962 pod označením RS-232 (RS obvykle chápáno jako zkratka z Recommended Standard, ve skutečnosti však z Radio Section). Později byla opakovaně revidována a významného rozšíření pak doznala především její třetí revize RS-232 C pocházející z roku 1969. Po ní následovaly další varianty (EIA 232 D, TIA/EIA 232 E). Jeho v současné době nejnovější verze je jakožto společné doporučení EIA a TIA (Telecommunications Industry Association) oficiálně uváděna pod názvem TIA/EIA 232 F. Vzhledem k tomu, že rozdíly mezi RS-232 C a pozdějšími variantami rozhraní nejsou příliš rozsáhlé a vzhledem k velké zažitosti staršího označení, je oficiální pojmenování používáno spíše výjimečně. V literatuře se proto o tomto rozhraní stále hovoří nejčastěji jako o rozhraní RS-232 C anebo prostě jen RS-232. Sériové rozhranní, které je v podstatných rysech shodné s TIA/EIA 232 F, je definováno rovněž mezinárodními doporučeními ITU-T^6 pod označeními V.24 (definice datových a řídicích signálů) a V.28 (elektrické parametry), [10].

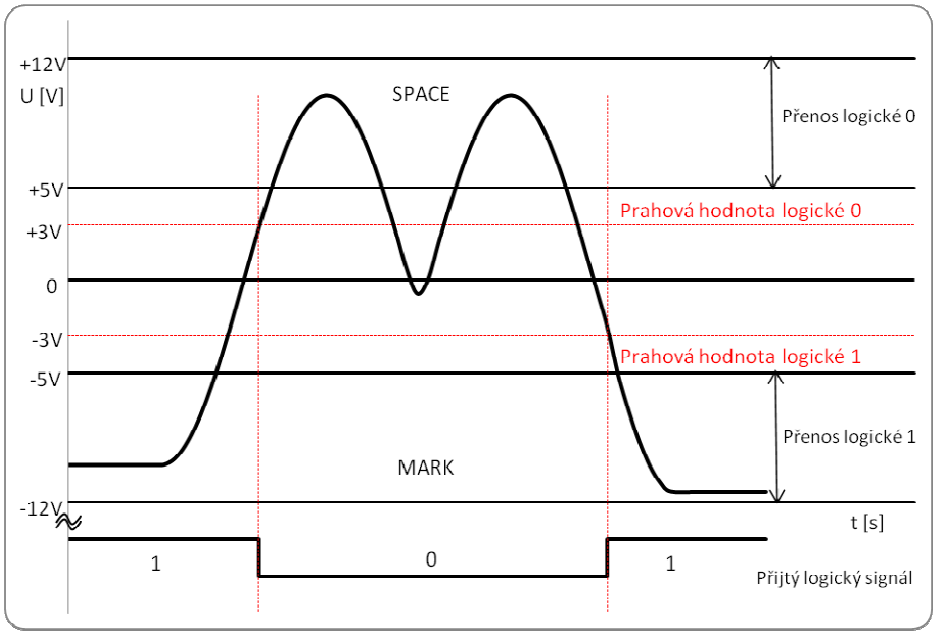
2.1.2 Elektrické parametry rozhraní TIA/EIA 232 F

Struktura vazebního obvodu tohoto rozhraní je schematicky znázorněna na obrázku 2.1. Zem vysílače a přijímače je společná a logické úrovně na signálovém vodiči jsou definovány vzhledem k této společné zemi. Jelikož šumová imunita signálů v úrovních TTL je velmi malá (v nejnepříznivějším případě jen 0,4 V) a budiče TTL nejsou určeny k buzení delších vedení s většími parazitními kapacitami, jsou pro přenos po tomto rozhraní stanoveny odlišné logické úrovně. Hodně log. 0 na výstupu vysílače odpovídá napětí +5 až +15V, zatímco hodnotě log. 1 odpovídají napětí -15 až -5V. Na vstupu přijímače je ovšem jako log. 0 interpretován signál kladné polarity s napětím již od +3 V a jako log. 1 signál záporné polarity s napětím do -3 V. Je tak zaručeno, že šumová imunita bude v nejhorším případě alespoň 2 V, [10].



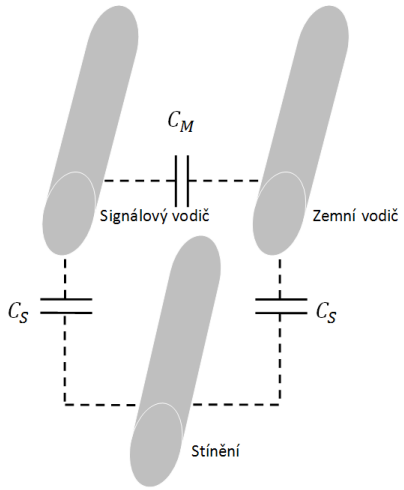
Obrázek 2.1: TIA/EIA 232 F

U standard RS-232 F je signál přenášen relativně ke společnému vodiči – uzemněnému obvodu (v jiných rozhraních, jako např. RS-422, se používají symetrické diferenční signály). Rozhraní neposkytuje galvanické oddělení zařízení. Napětí v rozsahu od -15V do -3V odpovídají logické jedničce (stav *MARK*) na vstupu dat přijímače (signál RxD). Napětí mezi +3V a +15V odpovídá logické nule (stav *SPACE*). V případě vstupů řídících signálů odpovídají napětí mezi +3V a +15V stavu ON a napětí od -15V do -3V stavu OFF. Rozsah mezi -3V a +3V je mrtvá zóna, která zajišťuje hysterezi přijímače: stav linky se považuje za změněný teprve poté, kdy překročí prahovou hodnotu. Úrovně signálů na výstupech vysílačů by měly být v rozpětí od -15V do -5V a od +5V do +15V, obrázek 2.2. Rozdíl potenciálů mezi zeměmi signálů (SG) připojených zařízení by měl být nižší než 2V. Vyšší rozdíl potenciálů může způsobit chybný příjem signálů. Rozhraní předpokládá, že připojená zařízení mají ochranné uzemnění, jsou-li napájena ze střídavé sítě a zároveň mají síťové filtry, [2].



Obrázek 2.2: Příjem signálů RS-232 F

Chování přijímače v zakázaném pásmu, tzn. při vstupních úrovních, které se pohybují od -3V do +3V není předepsáno. U jednotlivých typů převodníků mezi úrovněmi TIA/EIA 232 F a TTL se liší. Často bývají prahy logických úrovní stanoveny tak, aby přijímač mohl správně interpretovat i signál v úrovních TTL. Např. u velmi rozšířeného převodníku MAX 232 je jako log. 0 vyhodnoceno již vstupní napětí větší než 2,4 V a jako log. 1 napětí menší než 0,8 V. Vedle logických úrovní specifikuje doporučení TIA/EIA 232 F dále i vstupní odpor přijímače, který by se měl pohybovat mezi 3 a 7 kΩ.



Obrázek 2.3: Parazitní kapacity vedení

Maximální délka vedení byla původně podle doporučení RS-232 C stanovena na 15 m. S ohledem na to, že limitujícím faktorem však není ani tak délka vedení sama o sobě, ale jeho parazitní kapacity, nespecifikuje již současná varianta doporučení přímo maximální délku vedení, ale stanoví pouze, že zatěžovací kapacita vysílače smí být nejvýše 2500 pF. Tato zatěžovací kapacita je součtem vstupní kapacity přijímače a kapacity vedení. Vstupní kapacita přijímače bývá poměrně malá a obvykle nepřesahuje cca 20pF. Rozhodující je tedy kapacita vedení. Označíme-li celkovou kapacitu vedení v pF na metr délky (tedy celkovou měrnou kapacitu) jako C_C, můžeme pak nejvyšší přípustnou délku vedení v metrech přibližně počítat podle vztahu:

$$l_{max} = 2500/C_C \quad [\text{m}]. \tag{2.1}$$

Vzdálenost, na níž budeme moci komunikovat, bude tedy výrazně závislá na typu použitých kabelů, neboť měrná kapacita mezi vodiči závisí na průměru i vzdálenosti vodičů, stejně jako na permitivitě materiálu, z něhož je zhotovena jejich izolace.[4] Při stejném uspořádání vodičů i druhu izolačního materiálu pak bude navíc ještě záležet na tom, zda použitý kabel bude stíněný nebo nestíněný. Jak je naznačeno na obrázku 2.3, lze celkovou

[4] Kapacita se zvětšuje s průměrem vodičů a relativní permitivitou izolace, klesá s rostoucí vzdáleností vodičů.

kapacitu vedení počítat jako součet vzájemné kapacity mezi vodiči vedení (C_M) a rozptylové kapacity mezi vodiči a zemí, resp. stíněním (C_S).

$$C_C = C_M + C_S \quad [\text{F}]. \tag{2.2}$$

S kapacitou vedení také těsně souvisí dosažitelná rychlost přenosu. Propojení přijímače a vysílače na sériové lince obvykle nezahrnuje jen dva vodiče s případným stíněním, jak je tomu na obrázku 2.3, ale obsahuje ještě vodič pro přenos dat opačným směrem a většinou i řadu dalších vodičů pro řídicí signály. Proto bývá nejčastěji realizováno jako vícežilový kabel a uplatní se zde nejen kapacity mezi signálovým a zemním vodičem, ale také vzájemné vazební kapacity a indukčnosti mezi jednotlivými signálovými vodiči. Vlivem těchto kapacitních a indukčních vazeb dochází k tzv. přeslechům. Při změně logické hodnoty signálu v jednom vodiči se v sousedních vodičích objevuje rušivé napětí. Přitom platí, že velikost tohoto rušivého napětí vzrůstá se strmostí náběžných a sestupných hran signálů. Z tohoto důvodu je maximální rychlost změny napětí na signálových vodičích normou TIA/EIA 232 F omezena na 30 V/µs.

Jelikož maximální zkratové proudy, které jsou běžné budiče schopné poskytnout, nejsou nijak vysoké (např. 10 mA pro MAX 232), nelze v podstatě na maximální vzdálenost, tzn. při max. přípustné zatěžovací kapacitě 2500 pF, komunikovat rychlostmi většími než zhruba 20kbit/s. V souladu s tím specifikuje příslušné doporučení i ve své současné variantě TIA/EIA 232 F jako maximální přenosovou rychlost stále pouze 19,2 kbit/s. Na menší vzdálenosti lze ovšem komunikovat rychleji.

Přenosové rychlosti[5], které se prakticky používají, leží ve standardní řadě. Ta je uvedena v tabulce 2.1. Při komunikaci rychlostmi převyšujícími maximum stanovené normou se pak nejčastěji užívají rychlosti vyšší, např. 28800, 38400, 57600 a 115200 bit/s, [10].

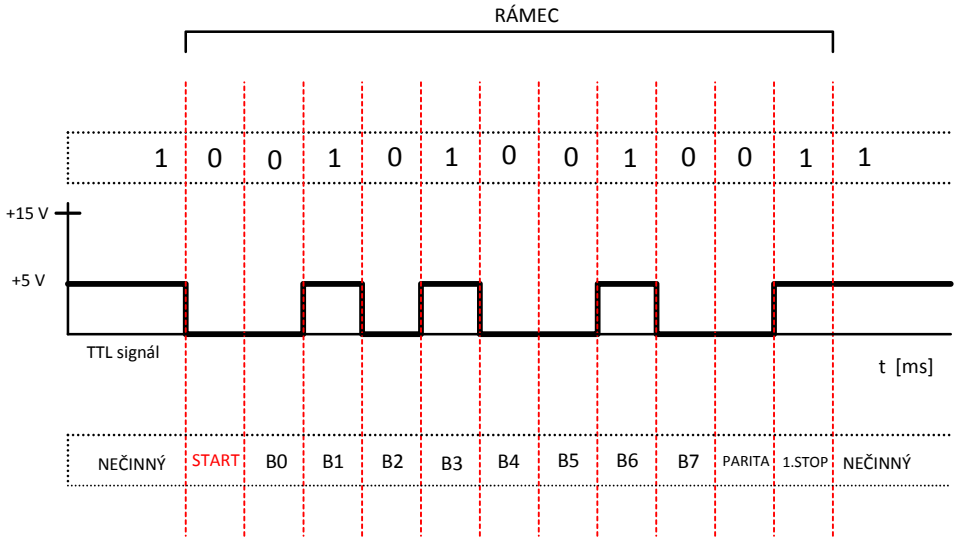
Modulační rychlost	Přenosová rychlost	Perioda	Modulační rychlost	Přenosová rychlost	Perioda
[Bd= Baud]	[bit/s]	[ms]	[Bd= Baud]	[bit/s]	[ms]
50	50	20,000000	1200	1200	0,833000
75	75	13,300000	2400	2400	0,417000
110	110	9,090000	4800	4800	0,208300
150	150	6,670000	9600	9600	0,104170
300	300	3,330000	19200	19200	0,052083
600	600	1,667000			

Tabulka 2.1: Jednotlivé modulační a přenosové rychlosti dvoustavového přenosu

[5] Přenosová rychlost se vyjadřuje počtem bitů přenesených za jednotku času. Jednotkou přenosové rychlosti je 1 bit/s. Je to rychlost přenosu dat, při níž za 1 sekundu přeneseme právě 1 bit. Při dvoustavovém přenosu (v našem případě) tato jednotka splývá s jednotkou tzv. modulační rychlosti 1Bd (Baud). Modulační rychlost (Baud rate) vyjadřuje počet změn signálu za sekundu. V textu bude upřednostňována přenosová rychlost Bit/s.

2.1.3 Formát přenosu dat po rozhraní TIA/EIA 232 F

Toto rozhraní používá variantu asynchronního přenosu, při níž jsou nezávislé hodiny přijímače a vysílače vždy znovu synchronizovány při vysílání každého datového slova pomocí zvláštních synchronizačních bitů tzv. start bitů. Princip tohoto způsobu přenosu je znázorněn na obrázku 2.4. V klidovém stavu, kdy není vysíláno nic, je na přenosové lince logická hodnota 1. Stav přenosové linky je přijímačem periodicky vzorkován s frekvencí, která je celočíselným násobkem přenosové rychlosti (obvykle je 16x nebo 64x vyšší). Pokud přijímač zjistí, že došlo ke změně stavu z log. 1 do log. 0, interpretuje to jako počátek start bitu, počká po dobu odpovídající polovině doby, která je při zvolené přenosové rychlosti vyhrazena na přenos jednoho bitu a stav linky otestuje znovu. Pokud zjistí, že se vrátil do log. 1, znamená to, že předchozí změna byla pouze náhodným šumem a nikoliv skutečným start bitem a přijímač proto začne znovu pravidelně vzorkovat stav linky jako předtím a čeká na nový přechod z log. 1 do log. 0. Jestliže se však signál po uplynutí bitové periody stále rovná log. 0, jedná se zřejmě o skutečný start bit a přijímač začne testovat přicházející signál vždy po uplynutí jedné bitové periody. Tedy tak, aby k testování stavu linky došlo vždy zhruba v polovině vysílání každého jednotlivého bitu. Tímto způsobem jsou postupně načteny hodnoty jednotlivých vysílaných datových bitů a na závěr je pak jednou či dvakrát testována hodnota stop bitu. Podle výsledku tohoto testu se určí, zda datové slovo bylo přeneseno, nebo došlo k tzv. chybě rámce (to znamená, zda stop bity byly skutečně nějakým způsobem porušeny nebo jsou přijímač i vysílač nakonfigurovány na jiný počet stop bitů), [10].



Obrázek 2.4: Přenos jednoho osmibitového slova s lichou paritou a jedním stop bitem.

Celé vysílané slovo je tak zahrnuto do rámce, který začíná jedním nulovým start bitem a zakončují jej volitelně buď jeden, jeden a půl anebo dva jedničkové stop bity. Varianta jeden a půl stop bitu přitom fakticky znamená, že se log. 1 na lince objeví po dobu odpovídající jeden a půl násobku času, který je při zvolené přenosové rychlosti vymezen na vyslání jednoho bitu. Hlavním důvodem, proč jsou stop bity vysílány, je poskytnout přijímacímu zařízení čas, aby se mohlo připravit k přijetí dalšího slova. Používání většího počtu stop bitů než jeden má proto smysl jen u zvláště pomalých zařízení, jako jsou např. elektromechanické dálnopisy. Samo vysílané slovo může volitelně obsahovat 5 až 8 datových bitů a k tomu jeden paritní bit. Paritní bit přitom může být nastaven jedním z následujících způsobů:

- sudá parita – bit je nastaven tak, aby celkový počet jedničkových bitů ve vysílaném slově včetně paritního bitu bylo sudé číslo
- lichá parita – bit je nastaven tak, aby celkový počet jedničkových bitů ve vysílaném slově včetně paritního bitu bylo liché číslo
- nulová parita (space parity) – paritní bit je vždy v log. 0
- jedničková parita (mark parity) – paritní bit je vždy v log. 1
- žádná parita – paritní bit se nepoužívá

Skutečný význam mají pouze varianty: sudá parita a lichá parita, které představují nejjednodušší formu zabezpečení přenosu dat. Dojde-li právě u jednoho bitu k chybě přenosu, tzn. log. 0 bude přijata jako log. 1 nebo obráceně, počet jedniček se změní podle nastavení druhu parity ze sudého na lichý nebo opačně a tato situace bude vyhodnocena jako chyba parity. Toto jednoduché zabezpečovací schéma však již není schopno odhalit dvě chyby v přenosu, neboť v tomto případě se parita nezmění. Podobně také nedokáže rozpoznat, zda došlo jen k jedné nebo většímu lichému počtu chyb a samozřejmě není také schopno chyby opravit. Má-li přenos proběhnout správně, je třeba, aby nejen přenosové rychlosti přijímače i vysílače byly nastaveny na stejnou hodnotu, ale také musí mít oba nastavený stejný počet datových bitů, stop bitů a stejnou paritu.

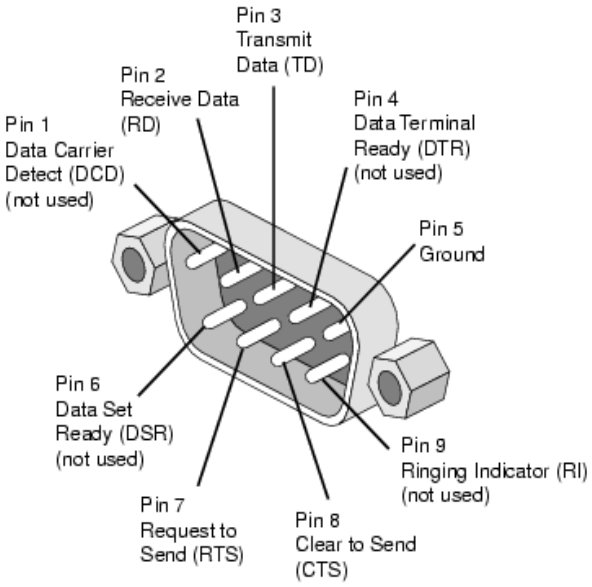
Při popsaném způsobu sériového přenosu dat je počátek vysílání datového slova skutečně asynchronní událostí, která může nastat kdykoliv bez vazby na jakýkoliv synchronizační signál. V rámci jednoho vysílaného rámce však již přenos probíhá synchronně. Celý postup je proto možné chápat jako zvláštní kombinaci synchronního a asynchronního přenosu, a proto bývá někdy v literatuře, která dodržuje přesné rozlišování, vyčleňován jako samostatná kategorie a označován jako tzv. arytmický přenos. Daleko častěji je však označován prostě jako asynchronní. K vysílání i příjmu datových slov po tomto rozhraní ve formátu podle obrázku 2.4 jsou určeny speciální obvody označované obvykle jako univerzální asynchronní přijímač a vysílač (ve zkratce UART z anglického Universal Asynchronous Receiver and Transmitter). Tyto obvody zajišťují oboustranný převod mezi sériovým a paralelním tvarem dat, generování a kontrolu ostatních prvků vysílaného rámce (start a stop bitů i parity) a obvykle i alespoň části řídicích signálů rozhraní, [10].

2.1.4 Nejdůležitější řídící signály rozhraní TIA/EIA 232 F

TIA/EIA 232 F specifikuje kromě vlastních datových signálů také řadu signálů pro řízení přenosu dat. Jejich názvy a funkce vycházejí především z toho, že ačkoliv se toto rozhraní běžně používá v řadě nejrůznějších aplikací, jeho původním hlavním určením, s ohledem na nějž byla norma vytvořena, je přenos dat mezi počítačem a modemem nebo obecněji mezi koncovým zařízením pro přenos dat (KZD anglicky DTE z data terminal equipment) a zařízením ukončujícím datový okruh (UZD anglicky DCE z data circuit-terminating equipment resp. data communication equipment).

Signál	Funkce
PG	Protected Ground (ochranné uzemnění): Je připojeno ke krytu zařízení a stínění kabelu.
SG	Signal (Circuit) Ground [zem signálu (obvodu)]: Přísluší k pracovním úrovním signálu.
TD	Transmit Data (přenos dat): Sériová data, výstup vysílače.
RD	Receive Data (příjem dat): Sériová data, vstup přijímače.
RTS	Reques To Send (požadavek na odeslání)
CTS	Clear To Send (povolení odeslat)
DSR	Data Set Ready (sada dat připravena)
DTR	Data Terminal Ready (datový terminál připraven)
DCD	Data Carrier Detected (detekována nosná frekvence dat)
RI	Ring Indicator (indikátor vyzvánění)

Tabulka 2.2: Význam signálů rozhraní RS-232



Obrázek 2.5: DB-9 Male (samec)

Jako spojovací prvek norma specifikuje 25ti kolíkový konektor DB-25 a od verze E i menší 26-ti kolíkový konektor. Prakticky se však nejčastěji používá 9ti kolíkový konektor, který obsahuje pouze nejdůležitější z řídicích signálů, viz obrázek 2.5. Počítač přitom bývá jakožto KZD osazen konektorem se špičkami (samec), zatímco ukončující zařízení (UZD) jsou obvykle osazeny konektorem s dutinkami (samice). Signály, které jsou dostupné pouze na 25ti kolíkovém konektoru, se používají jen velmi zřídka. Proto jsou v tabulce 2.3 uvedeny jenom signály vyvedené na 9ti kolíkovém konektoru. V tabulce 2.2 je popsán význam jednotlivých signálů, [10].

Signál			Kontakt konektoru		Směr
TIA/EIA 232 F	RS-232C	V.24	DB-25P	DB-9P	
PG	AA	101	1	5	-
SG	AB	102	7	5	-
TxD	BA	103	2	3	DTE->DCE (ven)
RxD	BB	104	3	2	DCE->DTE (dovnitř)
RTS	CA	105	4	7	DTE->DCE (ven)
CTS	CB	106	5	8	DCE->DTE (dovnitř)
DSR	CC	107	6	6	DCE->DTE (dovnitř)
DTR	CD	108/2	20	4	DTE->DCE (ven)
DCD	CF	109	8	1	DCE->DTE (dovnitř)
RI	CE	125	22	9	DCE->DTE (dovnitř)

Tabulka 2.3: Rozložení signálu rozhraní RS 232 na konektoru DB-9 a DB-25

3 Návrh sériového vysílače

Tato kapitola se zabývá vlastním návrhem sériového vysílače. Jsou zde zmíněny požadavky návrhu vysílače, způsob zadávání vstupních dat a způsob, jakým budou měněny parametry ovlivňující výstupní signál.

3.1 Požadavky na vysílač

K požadavkům na vysílač patří možnosti přenášet 5 – 8 informačních bitů, dále měnit přenosovou rychlost, a to podle rychlostí, které jsou uvedeny v tabulce 2.1. Je nutné umět počítat a přiřadit lichou i sudou paritu, popřípadě přiřadit paritu v hodnotě logické jedničky nebo nuly. Samozřejmě bude také možné paritu zcela vypnout. Vysílač nám musí dále umožnit nastavení počtu vyslaných stop bitů. Budeme moci přiřadit jeden nebo dva stop bity.

Jeden a půl stop bitu v našem vysílači neuvažuji, ale je možné jeden a půl stop bitu naprogramovat. Neprovedl jsem to prakticky, protože to považuji v dnešní době za málo využitelné. Jeden a půl stop bit je určen pro přenášení především 5 informačních bitů. Byl zaveden z důvodu, aby zařízení stihlo zpracovat přenesené informace, [11]. Teoreticky by se to dalo provést zdvojnásobením frekvence, která řídí stavový automat. Stavový automat by v každém stavu setrvával dva impulsy s výjimkou stavu, kdy bych na výstup přiřazoval dobu trvání jednoho a půl stop bitu. Dalo by se to vyřešit i jiným způsobem, a to tím, že by stavový automat reagoval i na sestupnou hranu signálu. Zde bychom nemohli vytvářet pouze impuls v požadovanou dobu, jak tomu bylo u předcházejícího řešení. Museli bychom generovat signál se třídou 1:1. Tím pádem by stavový automat reagoval na sestupnou hranu v polovině periody.

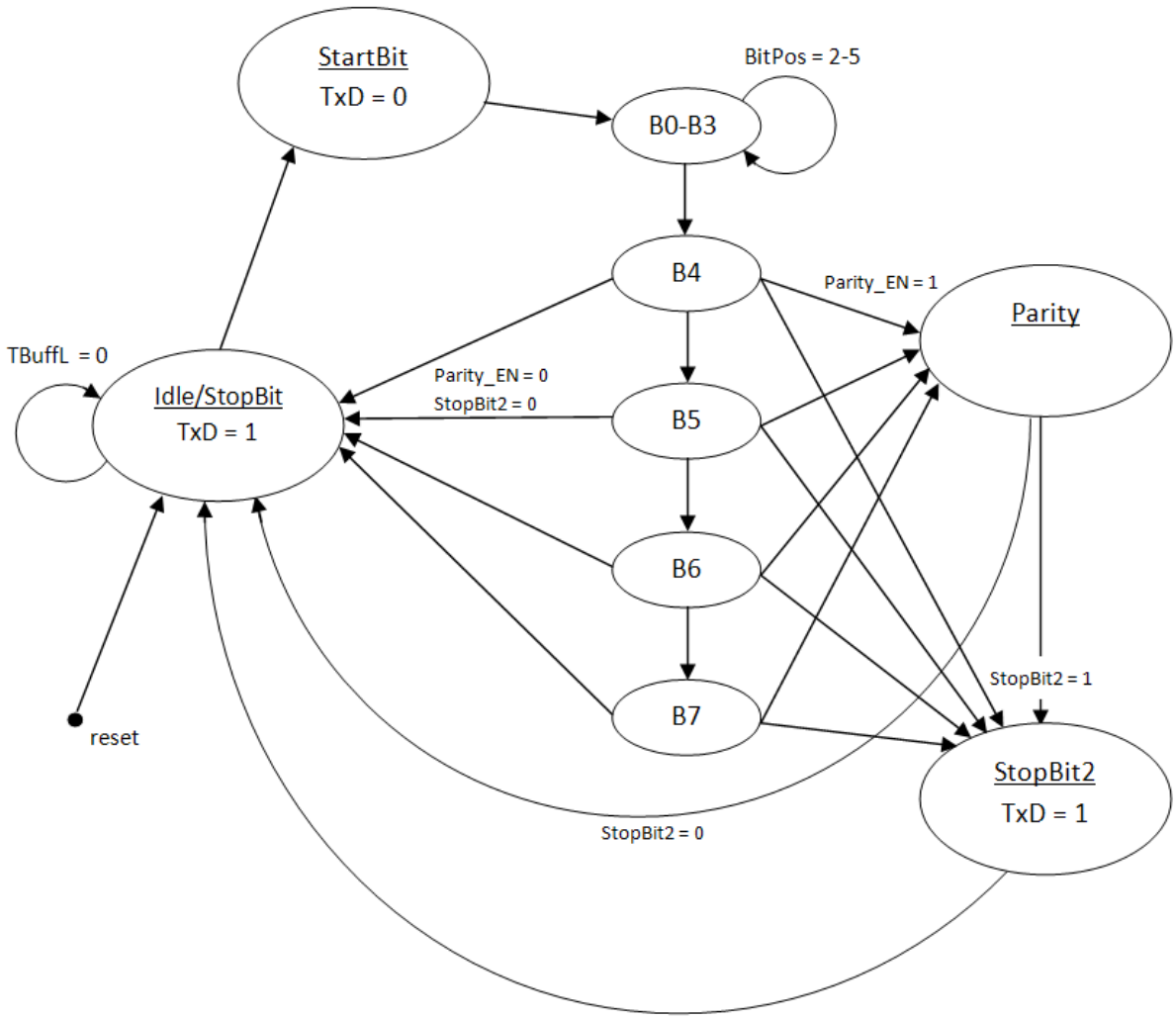
Všechny tyto parametry bude možné nastavovat pomocí čtyř mikrotlačítek a zobrazovat na čtyřmístném sedmisegmentovém LED displeji. Vstupní data budou nastavena pomocí osmi posuvných přepínačů.

3.2 Návrh vysílače a popis jeho vlastností

Navržený vysílač se bude skládat z několika částí. Jednou z částí je osmibitový datový registr, do kterého se ukládají vstupní data určená k odesílání. Další částí je Moorův automat, který bude řídit proces odesílání. Tento konečný automat FSM (Finite State Machine) je typ sekvenčního obvodu, který je navržen tak, aby vytvářel výstupní posloupnost odeslaných dat podle předem daných pravidel. U Moorova konečného automatu jsou výstupy závislé pouze na hodnotách vnitřního stavového registru, [5]. Další součástí je čítač, který nám řídí četnost vysílaných dat.

Na vstup vysílače jsou přiváděna paralelně vstupní data, která má vysílač odeslat a signál určující platnost těchto dat. Z vysílače vede signál, který oznamuje, že vysílač je právě zaneprázdněn vysíláním a signál, po kterém jsou data posílána do přijímače.

Jak již bylo řečeno, hraje Moorův automat a jeho správný návrh velmi důležitou roli při funkčnosti vysílače. Pro svůj vysílač jsem nadefinoval dvanáctistavový Moorův konečný automat, který je zobrazen na obrázku 3.1.



Obrázek 3.1: Stavový automat vysílače

3.2.1 Popis stavového automatu

Stav Idle/StopBit

Jedná se o počáteční a zároveň koncový stav. Zde se vysílač připravuje k vysílání tak, že data určená pro odesílání budou uložena do registru a na výstup bude přivedena logická 1. To vše za předpokladu, že vstupní data byla potvrzena k odeslání. Poté se automat přepne do následujícího stavu, který je označován jako *StartBit*.

Stav StartBit

V tomto stavu automat začíná vysílat, přivede na výstup sériové linky logickou 0 na místo logické 1.[6] Logická 0 nám symbolizuje *StartBit* . V tento okamžik se přiřadí logická jednička do signálu *SBusy*. Tento signál nám pak oznamuje, že vysílač je zaneprázdněn a že vysílá. Poté už následuje přepnutí do dalšího stavu *B0*.

Stavy B0-B3

Jedná se o čtyři stavy. První z těchto stavů je stav *B0 a* značí načtení bitu z registru, který je na pozici nula. Stav *B1* značí načtení bitu na pozici 1, další stav načtení bitu na pozici 2 a poslední stav načtení bitu na pozici 3. V datovém registru jsou bity seřazeny podle důležitosti, a to od nejméně významného bitu až po nejvíce významný bit. Tímto jsou data připravena k jednotlivému vyčítání, jak je uvedeno výše. Poté přechází automat do dalšího stavu *B5*.

Stav B5 (B6, B7 a B8)

Z registru je načten bit z pozice *B4* a současně se testuje, z kolika datových bitů je rámec vytvořen. Zde se testují čtyři možné hodnoty, 5, 6, 7 a 8 (počet přenášených bitů). Máme-li přenášet 5 bitů, musíme nadále zjišťovat, co bude bezprostředně následovat. Testujeme, jestli máme vysílat s paritou anebo bez parity. Jestliže máme paritu aktivovanou, následuje přechod do stavu *Parity*. Pokud ovšem nemáme aktivovanou paritu, musíme testovat, kolik se má přiřadit stop bitů (jeden stop bit, nebo dva stop bity). Z toho plyne, že pokud vysíláme jeden stop bit, následuje přechod do stavu *Idle/StopBit*. V případě, že se vysílají dva jednobitové stop bity, následuje stav *StopBit2*. Jestli je ovšem počet přenášených datových bitů větší než 5, provede se načtení z datového registru, konkrétně z pozice *B5*. Poté přejde automat do stavu *B6,* kde následuje podobný proces. Stejně tento proces probíhá i u stavu *B7* a stavu *B8*.

Stav Parity

V tomto stavu se provádí zabezpečení datových bitů pomocí kontrolního bitu nazvaného paritní bit. Tento paritní bit se mění na základě přenášených bitů tak, aby se počet odeslaných bitů doplnil do lichého nebo sudého počtu, popřípadě se může nastavit na pevno do hodnoty logické 1 nebo logické 0. Dále se musí testovat, kolik se má přiřadit stop bitů. Jestliže je přiřazen jeden stop bit, následuje přechod do stavu *Idle/StopBit*. V případě dvou jednobitových stop bitů se přechází do stavu *StopBit2* .

Stav StopBit2

Zde se přiřadí na sériový výstup logická 1, která reprezentuje stop bit. Stavový automat přechází do konečného, respektive do počátečního stavu *Idle/StopBit*.

---

[6] Jestliže se vysílač nacházel v *Idle/StopBit* módu, pak posloupnost "10" znamená, že se má přijímač zasynchronizovat.

4 Jazyk VHDL

Název VHDL představuje akronym – VHSIC Hardware Description Language. Samo označení VHSIC je další akronym představující název projektu, v rámci něhož byl jazyk VHDL zpracován a znamená Very High Speed Integrated Circuits. I když vysoká či nízká rychlost popisovaných obvodů není pro použití jazyka VHDL podstatná, uvedené označení se vžilo a obecně se používá. Jazyk VHDL byl původně vyvinut především pro modelování a pro simulaci rozsáhlých systémů, [9].

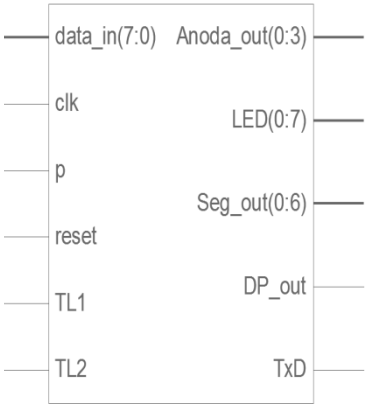
Pro popis modelu navrhovaného v jazyku VHDL budeme používat termín zdrojový text.

4.1 Důležité části vysílače popsané jazykem VHDL

Deklarace entity je základní částí jazyka VHDL a popisuje vstupy a výstupy konstrukce, v našem případě tedy popisuje vstupy a výstupy vysílače. Na obrázku 4.1 je zachycena entita naprogramovaného vysílače a na dalším obrázku 4.2 je vidět jeho RTL schéma.

```
30  entity BLOKRS232 is
31      Port (  clk,reset : in std_logic;
32              p         : in std_logic;
33
34              data_in   : in std_logic_vector(7 downto 0);
35              TL1,TL2   : in std_logic;
36              --sedmisegmentovka
37              Seg_out   : out std_logic_vector(0 to 6);
38              Anoda_out : out std_logic_vector(0 to 3);
39              DP_out    : out std_logic;
40              --LED diody
41              LED       : out std_logic_vector (0 to 7);
42              --seriovy vystup
43              TxD       : out std_logic
44          );
45  end BLOKRS232;
```

Obrázek 4.1: Rozhraní komponenty popsané ve VHDL



Obrázek 4.2: RTL schéma vysílače

Nezbytnou součástí vysílače je čítač s asynchronním resetem, který je řízen hodinovým signálem *clk* 50MHz. Funkce samotného čítače je velmi jednoduchá. Když je signál *reset* v úrovni logické 0, tak se s každou nástupnou hranou signálu *clk* inkrementuje proměnná *DivTx* o jedničku. Toto čítání probíhá až do splnění podmínky *DivTx = Divisor- 1*. Poté se čítač vynuluje a přiřadí do signálu *STopTx* logickou jedničku. V dalším cyklu se logická jednička v signálu přepíše na logickou nulu. Tímto způsobem se generuje puls, kterým se řídí stavy stavového automatu. Když je signál *reset* v úrovni logické 1, čítání neprobíhá a proměnné *DivTx* a signálu *STopTx* se přiřadí nulová hodnota. Ukázka zdrojového textu je na obrázku 4.3.

```
GEN: process (reset,clk,Divisor)
        variable DivTx: integer;
begin
   if reset = '1' then
      STopTx <= '0';
      DivTx := 0;
   elsif clk'event and clk='1' then
      STopTx <= '0';
      if DivTx = Divisor-1 then
         DivTx := 0;
         STopTx <= '1';
      else
         DivTx := DivTx +1;
      end if;
   end if;
end process GEN;
```

Obrázek 4.3: Zdrojový text čítače s asynchronním resetem

Na obrázku 4.4 je uvedeno, jak je přiřazován poměr do signálu *Divisor*, čímž se mění přenosová rychlost.

Příkaz »*case*« slouží, podobně jako předchozí příkaz »*if*«, pro vykonání sekvence příkazů v závislosti na vyhodnocení podmínek, avšak bez priorit. Zde je tento příkaz používán pro přiřazení požadovaného poměru do signálu *Divisor*, který je definován jako »*type Divisor is range 600 to 115200*«. Tento signál Divisor pak řídí předchozí čítač, a tím i přenosovou rychlost vysílače. Za dvěma pomlčkami je uvedena přenosová rychlost, obrázek 4.4.

```
Rychlost:process (reset,clk,SnBaud)
begin
      if reset ='1'then
         Divisor <= 0;
      elsif clk'event and clk ='1' then
         case SnBaud is
            when "0000" => Divisor <= 5208;  --   9600
            when "0001" => Divisor <= 83333; --    600
            when "0010" => Divisor <= 41667; --   1200
            when "0011" => Divisor <= 20833; --   2400
            when "0100" => Divisor <= 10417; --   4800
            when "0101" => Divisor <= 3472;  --  14400
            when "0110" => Divisor <= 2604;  --  19200
            when "0111" => Divisor <= 1736;  --  28800
            when "1000" => Divisor <= 1302;  --  38400
            when "1001" => Divisor <= 893;   --  56000
            when "1010" => Divisor <= 868;   --  57600
            when "1011" => Divisor <= 434;   -- 115200
            when others => Divisor <= 5208;  --   9600
         end case;
      end if;
end process Rychlost;
```

Obrázek 4.4: Použití příkazu »*case*«, k řízení přenosové rychlosti

Příkaz »*case*« používám i pro výpočet a přiřazení parity. U výpočtu jej používám proto, že na základě něj určíme, z kolika informačních bitů se má parita počítat. K samotnému výpočtu

parity již používám operátor *xor.* Funkce tohoto operátora je zřejmá z názvu. Tento operátor nám zajistí výpočet sudé parity. Pro výpočet liché parity je zapotřebí ještě tento výpočet negovat, a to za pomocí operátora *not*.

Při návrhu vysílače jsem se zabýval způsobem odesílání jednotlivých bitů. Nabízelo se několik možností, jak budou data zpracovávána a následně vysílána. Vytyčil jsem si tedy, že při návrhu budu využívat stavový automat, který bude řídit celý proces odesílání. Bylo nutné zvolit správný počet stavů, které tento automat bude nabývat. Postup při implementaci byl následující: Při nultém stavu (*Idle/StopBit*) automat čeká na data a potvrzení jejich platnosti. Po splnění této podmínky se data uloží do registru tak, že na nejnižší pozici se uloží bit nejméně významný a na nejvyšší pozici se uloží bit nejvíce významný. Tímto jsou data připravena k dalšímu použití. Poté automat přejde do stavu, kde začne s odesíláním dat. Prvním odeslaným bitem musí být vždy start bit, který bude upozorňovat přijímač na začátek přenosu, proto se tento stav nazývá *StartBit*. Po jeho odeslání přijdou na řadu data uložená v registru. Odeslání těchto dat bude mít na starosti několik dalších stavů automatu. Jsou to stavy *2-5(B0-B3), 6(B4), 7(B5), 8(B6), 9(B7)*. Po těchto osmi stavech byla tedy odeslána všechna data z registru. Současně se bude zjišťovat, který stav bude následovat. Bude-li vysílačem podporována parita, přejde automat do stavu *10* nazvaný *Parity*. Zde se přidá bit, který reprezentuje spočítanou paritu. Jestli ovšem vysílač paritní bit vysílat nebude, přejde rovnou do stavu *11(StopBit2)* a přeskočí stav *10*(*Parity).* Ovšem i stav *StopBit2* může vysílač přeskočit, pokud nebude nastaveno, že má vyslat dva jednobitové stop bity za sebou. Po těchto stavech bude následovat počáteční stav neboli nultý stav označený jako *Idle/StopBit.* V tomto klidovém stavu, kdy vysílač nevysílá žádná data, je na výstupu hodnota reprezentující logickou jedničku. Ukázka části automatu a jeho logiky pro určení následujícího stavu je na obrázku 4.5.

```
case BitPos is
   when 0 =>----------------stav 0(Idle/StopBit)
      TxD <= '1';
      Sbusy <= '0';
      ...
   when 1 =>----------------stav 1(StartBit)
      TxD <= '0';
      BitPos := 2;
      Sbusy <= '1';
   when 6 =>----------------stav 6(B4)
      TxD <= TReg(4);
      if Bits = 5 then
         if Parity_EN = 0 then
            if StopB = 1 then
               BitPos := 0;
            elsif StopB = 2 then
               BitPos := 11;
            end if;
         elsif Parity_EN = 1 then
            BitPos := 10;
         end if;
      else
         BitPos := 7;
      end if;
   when 7 =>----------------stav 7(B5)
      TxD <= TReg(5);
      if Bits = 6 then
         ...
      else
         BitPos := 8;
      end if;
   when 8 =>----------------stav 8(B6)
      TxD <= TReg(6);
      if Bits = 7 then
         ...
      else
         BitPos := 9;
      end if;
   when 9 =>----------------stav 9(B7)
      TxD <= TReg(7);
      if Bits = 8 then
         ...
      end if;
   when 10 =>---------------stav 10(Parity)
      TxD <= MakeParity; --priradi se parita
      if StopB = 1 then
         BitPos := 0;
      elsif StopB = 2 then
         BitPos := 11;
      end if;
   when 11 =>---------------stav 11(StopBit2)
      TxD <= '1' ;       --priradi se stop bit
      BitPos :=0;
   when others =>-----------stav 3-5(B0-B3)
         TxD <= TReg(BitPos - 2);
         BitPos := BitPos + 1;
   end case;
```

Obrázek 4.5: Zdrojový kód části stavového automatu

5 Vývojové prostředky

Mezi tyto prostředky patří software a hardware. Nejdříve se budeme zabývat softwarovými vývojovými prostředky, poté si popíšeme hardwarové vývojové prostředky.

5.1 Softwarové vývojové prostředky

Pro vývoj aplikací s FPGA existuje několik návrhových systémů. Pro vývoj je nutné použít minimálně dvou nástrojů. Prvním je nástroj pro syntézu, který převede textový popis návrhu ve VHDL jazyce do formátu vhodného pro fyzickou realizaci obvodu, například software od firmy Xilinx pod názvem ISE WebPACK. Druhý nástroj se jmenuje ModelSim XE a zajistí simulaci, která nám ověří činnost obvodu na počítači.

5.1.1 Syntéza

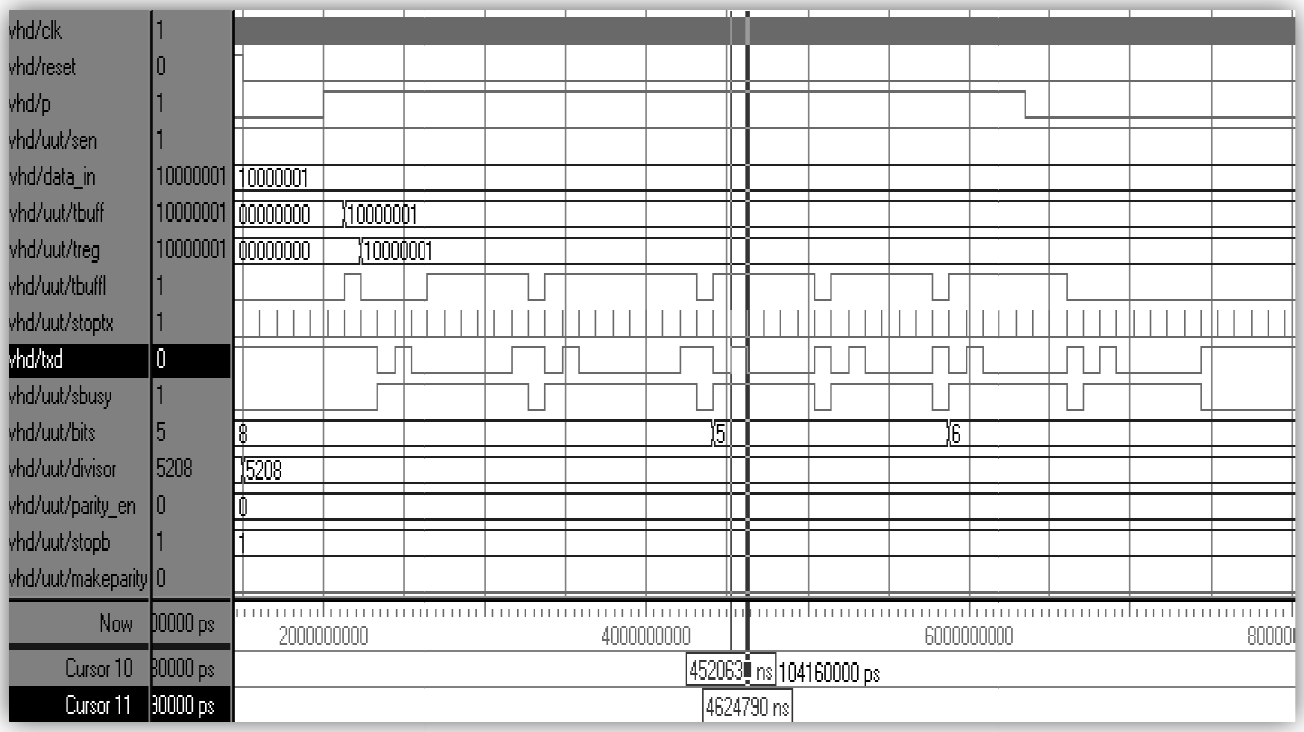
Po napsání zdrojového textu byla provedena syntéza v ISE WebPACK (verze 9.2i). Syntéza je vytvoření netlistu, tj. zapojení obvodových prvků (logické členy, klopné obvody, registry atd.). Jedná se o vytvoření schématu zapojení s obvodovými prvky, které jsou obsaženy v předpokládaných cílových obvodech a které vykonávají požadovanou funkci. U složitých obvodů CPLD a FPGA jsou kromě výše uvedených obvodových prvků do netlistu zařazeny prvky specifické pro cílové obvody (obvykle jsou společné pro určitou řadu obvodů, například *XC9500XL*, *Spartan-II* a podobně). Pokud nejsou tyto specifické prvky použity, je netlist přenositelný na jiné cílové obvody. Jsou-li použity, je přenositelnost omezena na takové obvody, které tyto prvky obsahují. Jakákoliv konstrukce je v principu realizovatelná bez těchto prvků (s výjimkou velmi speciálních prvků, jako jsou prvky pro úpravu hodinového signálu, pro řízení odběru z napájecího zdroje apod.). Jejich použití však může výsledek syntézy výrazně zlepšit a v mnohých případech je tento výsledek bez nich tak složitý, že je prakticky nepoužitelný (příklady: blokové násobičky a další prostředky pro podporu aritmetiky, paměti RAM atd.). Netlist se nejčastěji zapisuje ve formátu EDIF, [8].

5.1.2 Funkční simulace

Funkční simulace se typicky provádí pro ověření syntaxe zapsaného kódu a k ověření, zda model správně funguje, aniž se přihlíží k časovému rozměru. Tato simulace není závislá na architektuře cílového obvodu, pokud se nepoužívají primitivy jako vložené komponenty. Používá se k ní model představovaný zdrojovým textem, kterým konstruktér popsal svou představu o funkci vyvíjeného zařízení. Provádí se před dalším zpracováním tohoto textu

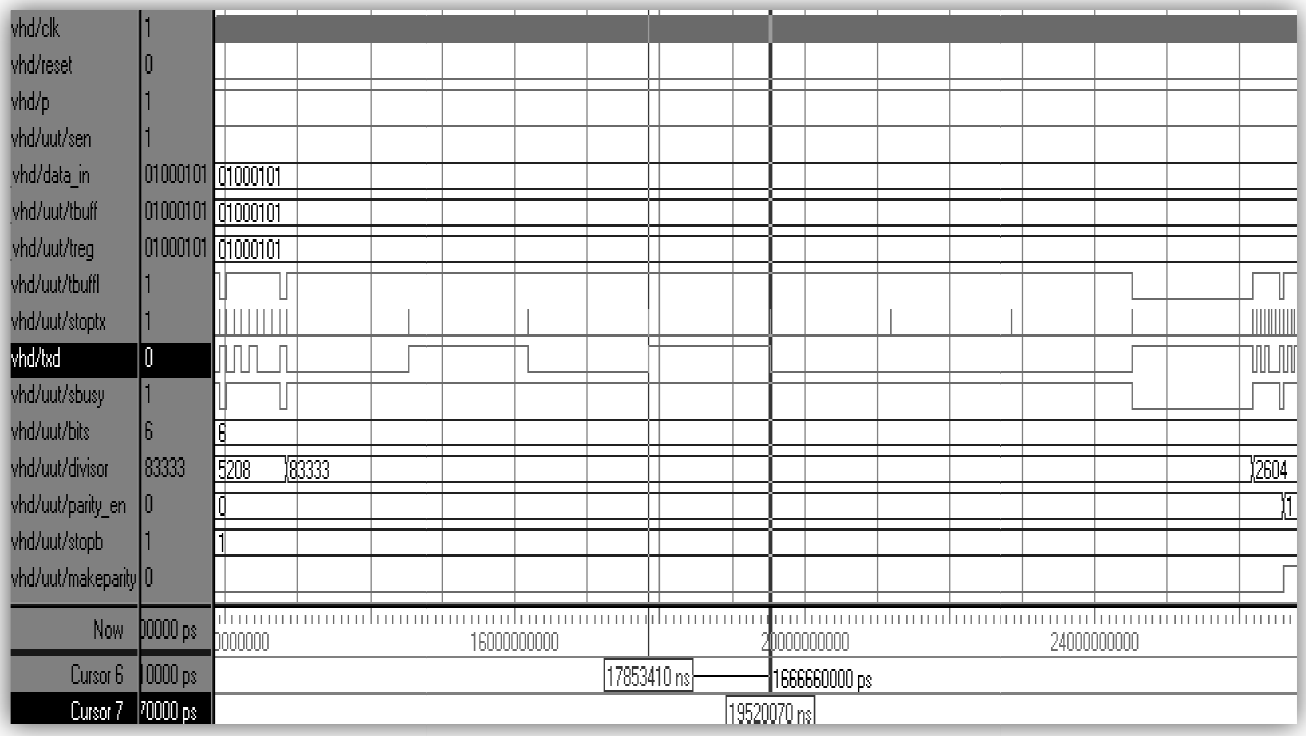
systémem – proto se jí někdy také říká simulace před fittingem nebo před propojením. Tento model neobsahuje údaje o konkrétních vlastnostech cílového obvodu, jako například údaje o zpoždění a podobně, [8]. Pro ověření funkce vysílače bylo tedy nutné vytvořit speciální program v jazyce VHDL – testbench.

Testbench je speciální entita v jazyce VHDL, která slouží pro generování vstupních testovacích signálů pro ověření správné funkce námi naprogramované entity. Tyto testovací signály pak pomocí vazby *component* přivádí na námi naprogramovanou entitu, [5]. Simulace byla provedena pomocí softwaru ModelSim XE III/Starter (verze 6.2g). Výsledky simulace jsou uvedeny na obrázcích 5.1, 5.2, 5.3 a také v Příloze 2.



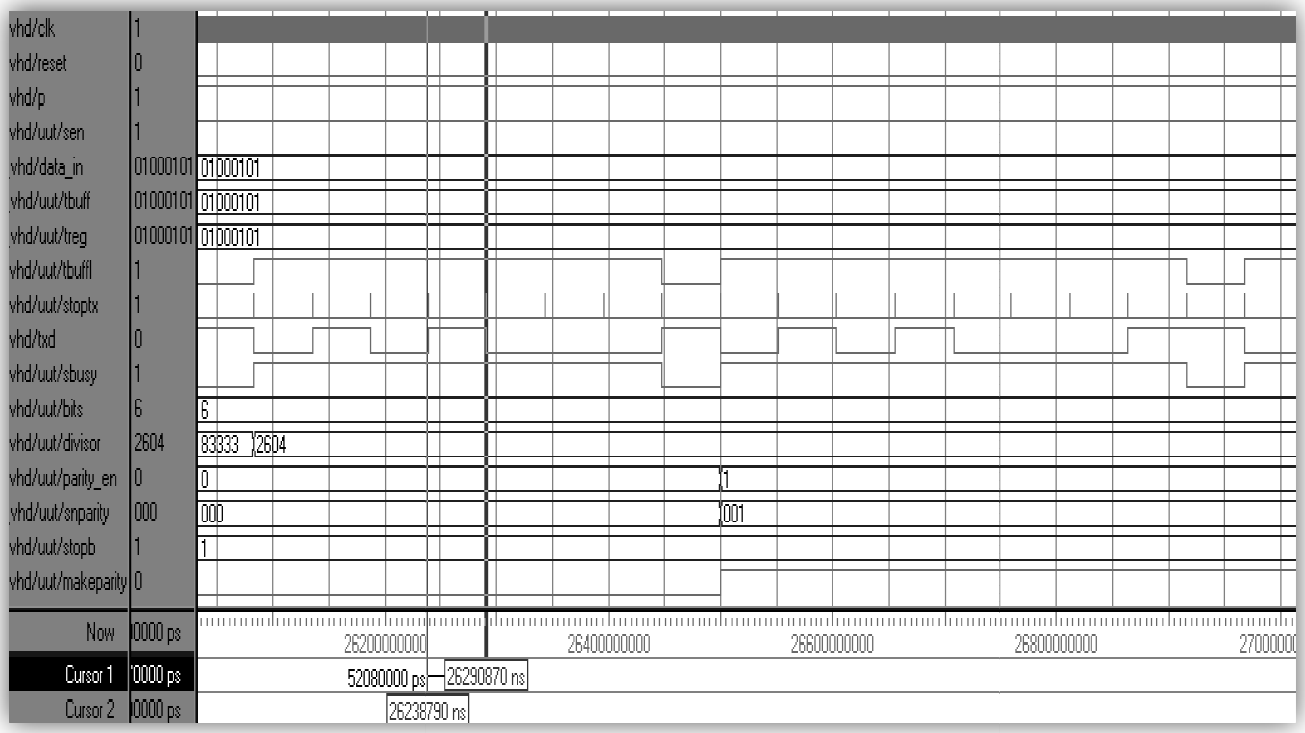
Obrázek 5.1: Simulace vysílače provedena v ModelSim XE

V našem případě jsme na vstup přivedli data *10000001* (*data_in*). Tyto data se tedy zapíšou do *TBuff* při potvrzení vysílání. Dále vysílač tyto data načte do registru, a to vždy v okamžiku, kdy se hodnota signálu *TBuffL* mění z logické 1 na logickou 0. Signál *STopTx* vychází z čítače, ten je řízen hodnotou *Divisor*. *Divisor* je hodnota, která řídí přenosovou rychlost. Pokud tedy chceme měnit přenosovou rychlost, musíme změnit konstantu *Divisor*. Na obrázku 5.1 měníme počet informačních bitů, a to z osmi na pět a dále pak na šest přenášených informačních bitů. Na výstupu vysílače, který je značen jako *TxD,* jsou pak patrné změny způsobené počtem přenášených informačních bitů. Tyto změny nastávají pouze po ukončení vysílaného rámce. Ukončení rámce nám signalizuje signál *SBusy*. Parita se při našem vysílání neuplatňuje proto, že hodnota *Parity_EN* je na nízké úrovni (logické 0), tudíž je zakázaná. Na výstupu *TxD* si můžeme ověřit, kolik používáme při vysílaní stop bitů, v našem případě je to jeden stop bit (*STopB = 1*), což je rovněž vidět na obrázku 5.1.



Obrázek 5.2: Simulace vysílače provedena v ModelSim XE

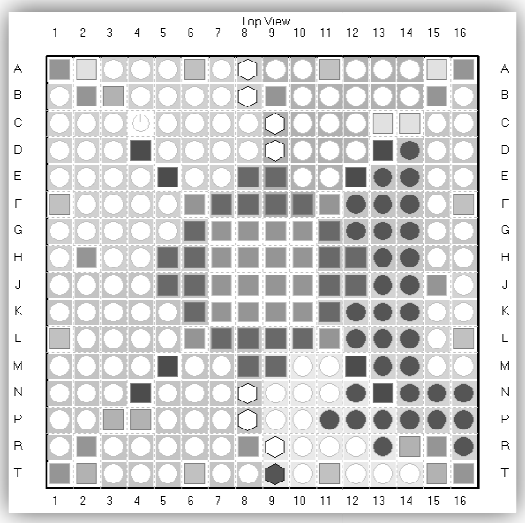
Na obrázku 5.2 je zachycena změna přenosové rychlosti. Mění se z 9600Bit/s na 600Bit/s a potom na 19200Bit/s, což lze ověřit pomocí změřené periody vysílaného signálu a srovnat s tabulkou 2.1. Na dalším obrázku 5.3 je zachyceno přidání liché (*001*) parity. Protože je vyslán sudý počet jedniček (*101000*), přidá se parita ve vysoké úrovni.



Obrázek 5.3: Simulace vysílače provedena v ModelSim XE

5.1.3 Přiřazení signálů

Před vlastní implementací syntetizovaného návrhu je třeba zadat umístění vývodů vysílače na cílové architektuře. Tuto informaci obsahuje textový soubor nazývaný Implementation Constraints File (UCF), který je třeba vytvořit. Ten je pak možné v následujícím dialogu asociovat k požadovanému modulu, [12]. Tabulka 5.1 ukazuje přiřazení signálů jednotlivým vývodům (pinům) obvodu Spartan-3. Dále je na obrázku 5.4 vidět rozmístění pinu na obvodu a také, které z těchto pinů používáme. K takto značeným vývodům v následující kapitole přiřadíme použité části vývojového kitu Spartan-3 Starter.



Obrázek 5.4: Tmavá kolečka označují využívané piny obvodu Spartan-3

	I/O Name	I/O Direction	Loc	Bank
	Anoda_out<0>	Output	d14	BANK2
	Anoda_out<1>	Output	g14	BANK2
	Anoda_out<2>	Output	f14	BANK2
	Anoda_out<3>	Output	e13	BANK2
	clk	Input	t9	BANK4
	data_in<0>	Input	f12	BANK2
	data_in<1>	Input	g12	BANK2
	data_in<2>	Input	h14	BANK2
	data_in<3>	Input	h13	BANK2
	data_in<4>	Input	j14	BANK3
	data_in<5>	Input	j13	BANK3
	data_in<6>	Input	k14	BANK3
	data_in<7>	Input	k13	BANK3
	DP_out	Output	p16	BANK3
	LED<0>	Output	k12	BANK3
	LED<1>	Output	p14	BANK3
	LED<2>	Output	l12	BANK3
	LED<3>	Output	n14	BANK3
	LED<4>	Output	p13	BANK4
	LED<5>	Output	n12	BANK4
	LED<6>	Output	p12	BANK4
	LED<7>	Output	p11	BANK4
	p	Input	m13	BANK3
	reset	Input	l14	BANK3
	Seg_out<0>	Output	e14	BANK2
	Seg_out<1>	Output	g13	BANK2
	Seg_out<2>	Output	n15	BANK3
	Seg_out<3>	Output	p15	BANK3
	Seg_out<4>	Output	r16	BANK3
	Seg_out<5>	Output	f13	BANK2
	Seg_out<6>	Output	n16	BANK3
	TL1	Input	m14	BANK3
	TL2	Input	l13	BANK3
	TxD	Output	r13	BANK4

Tabulka 5.1. Přiřazení vývodů jednotlivým signálům

5.1.4 Implementace

Implementace zahrnuje několik postupných kroků, které vyústí do vytvoření popisu propojení cílového obvodu, který je podkladem pro jeho „vypálení“ do PLD nebo pro vytvoření tzv. bitstreamu používaného pro konfiguraci FPGA. Terminologie je zde poněkud odlišná u cílových obvodů PLD a FPGA. Důležité jsou zejména kroky označené *Fitting* (u PLD) a *Mapping a Place-and-Route* (u FPGA). Zhruba řečeno, mapování spočívá v přiřazení obvodových prvků použitých ve výsledku syntézy konkrétním prvkům obsažených v cílovém obvodu. Poté následuje rozmístění (*placement*) těchto prvků, tedy jejich přiřazení odpovídajícím obvodovým strukturám, které jsou v nenaprogramovaném cílovém obvodu

vytvořeny. Zdařilá volba rozmístění má velký význam pro provedení následujícího kroku – propojení (*routing*). Propojením se vytvoří plán výsledné struktury včetně potřebného nakonfigurování programovatelných propojek. Požadovaný stav jednotlivých propojek je pak obsažen v souboru, který je výsledkem implementace (*bitstream*), [8].

5.2 Hardwarové vývojové prostředky

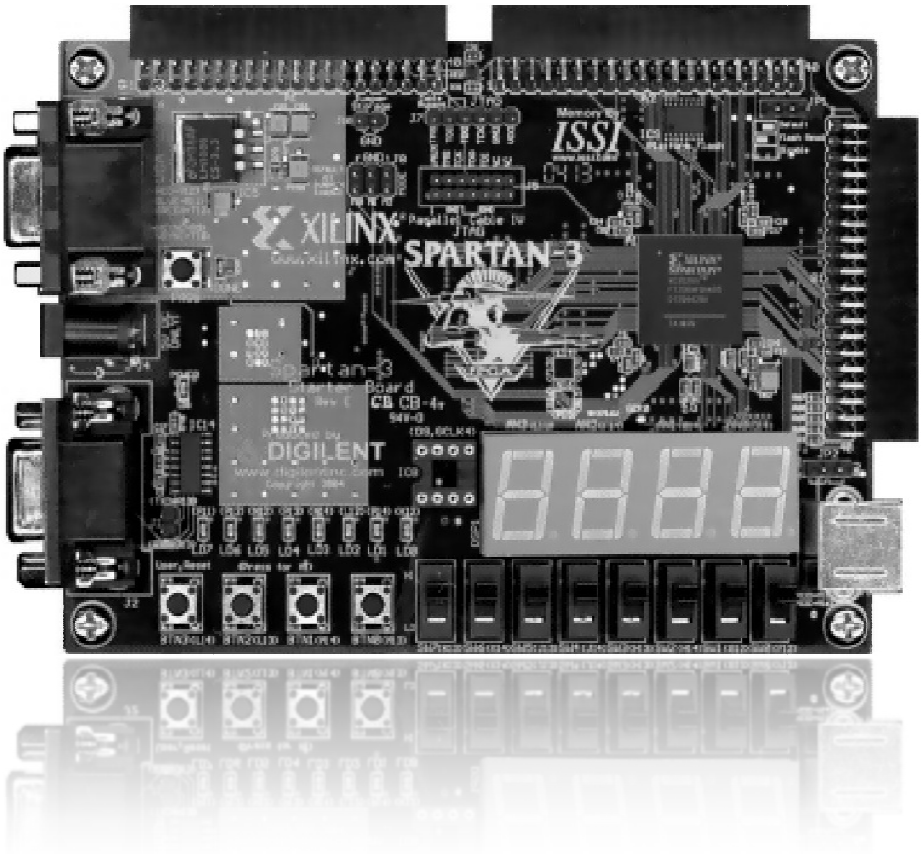
Funkčnost návrhu bude ověřena pomocí vývojového kitu, konkrétně pomocí Spartan-3 Starter Kit board. Tento kit plně vyhovuje našim požadavkům. Stručně si shrneme části kitu a ty, které budeme používat, podrobněji popíšeme.

5.2.1 Spartan-3 Starter Kit board

Spartan-3 Starter Kit je platforma sloužící pro návrh a testování FPGA designů. Obsahuje základní rozhraní potřebné pro interakci s běžnými periferiemi či přímý vstup uživatele.

Hlavní částí Starter Kitu je Xilinx Spartan-3 XC3S200 FPGA. Obsahuje 200 000 ekvivalentních hradel.

Obrázek 5.5: Spartan-3 Starter Kit



Součásti Starter Kitu:

- Xilinx XCF02S Platform Flash – slouží k trvalému uložení konfiguračního PROM souboru pro FPGA, obvod pracuje hned po zapojení - potřebnou konfiguraci má uloženou v této trvalé paměti
- 1MB SRAM – 2 moduly 256Kx16 SRAM s latencí 10ns, společná adresní sběrnice
- 3-bitový VGA konektor – umožňuje zobrazovat 8 barev
- 9-pinový DB-9 konektor – deska obsahuje převodník napěťových úrovní (MAX232)
- PS/2 konektor – možnost připojení myši nebo klávesnice
- čtyři sedmisegmentové displeje
- 8 LED diod
- 4 mikrotlačítka
- 8 přepínačů
- 50MHz krystal – zdroj hodinového signálu
- tlačítko, kterým lze vynutit rekonfiguraci FPGA, standardně se tak děje při připojení napětí
- 40-pinové rozšiřující porty – možnost připojení dalších periferních zařízení
- JTAG rozhraní, [4].

Nyní se zaměříme na jednotlivé části Spartan-3 Starter Kitu, které budeme v našem projektu vysílače využívat.

5.2.1.1 Tlačítka, přepínače, LED diody

Čtyři mikrotlačítka, která na desce nesou označení BTN3(Reset), BTN2, BTN1, BTN0, přivedou po stisknutí logickou jedničku na příslušný pin obvod FPGA, podle níže uvedené tabulky 5.2, [4].

Mikrotlačítko	BTN3 (Reset)	BTN2	BTN1	BTN0
FPGA Pin	L14	L13	M14	M13

Tabulka 5.2: Spojení mikrotlačítek se Spartanem-3 FPGA

Přepínače lze přepnout trvale do jedné logické polohy. Tabulka 5.3 ukazuje vzájemné propojení s jednotlivými piny obvodu FPGA.

Přepínač	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA Pin	K13	K14	J13	J14	H13	H14	G12	F12

Tabulka 5.3: Spojení přepínačů se Spartanem-3 FPGA

LED diody jsou připojeny přes odpory a svítí při logické 1. Opět jejich propojení, tabulka 5.4.

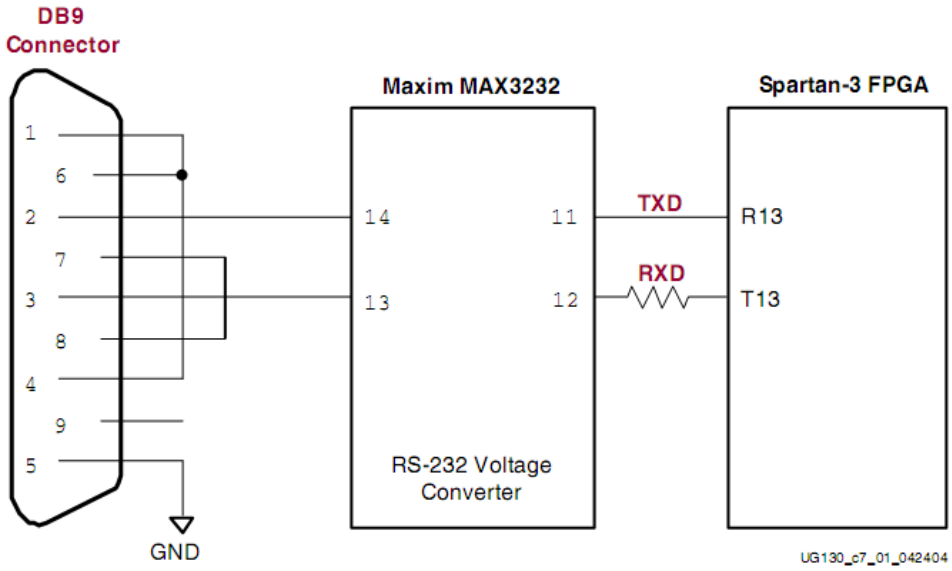
LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA Pin	P11	P12	N12	P13	N14	L12	P14	K12

Tabulka 5.4: Spojení LED se Spartanem-3 FPGA

5.2.1.2 Sériový port RS-232

Spartan-3 Starter Kit board obsahuje sériové rozhraní RS-232. Vysílaný a přijímaný signál je veden na samičí konektor DB 9, na desce značen J2. Obrázek 5.6 ukazuje propojení mezi FPGA a konektorem DB-9, včetně napěťového konvertoru Maxim MAX3232. Obvod FPGA podporuje na výstupu napěťové standardy LVTTL nebo LVCMOS, které pak přivedeme na jednotlivé vstupy obvodu MAX3232. Tento obvod poté vhodným způsobem přemění logickou hodnotu do vhodné (RS-232) napěťové úrovně.

Hardwarové řízení není podporováno. Signály DCD, DTR a DSR jsou na konektoru vzájemně propojeny, jak je ukázáno na obrázku 5.6. Stejně se vzájemně propojují i signály RTS a CTS, [4].



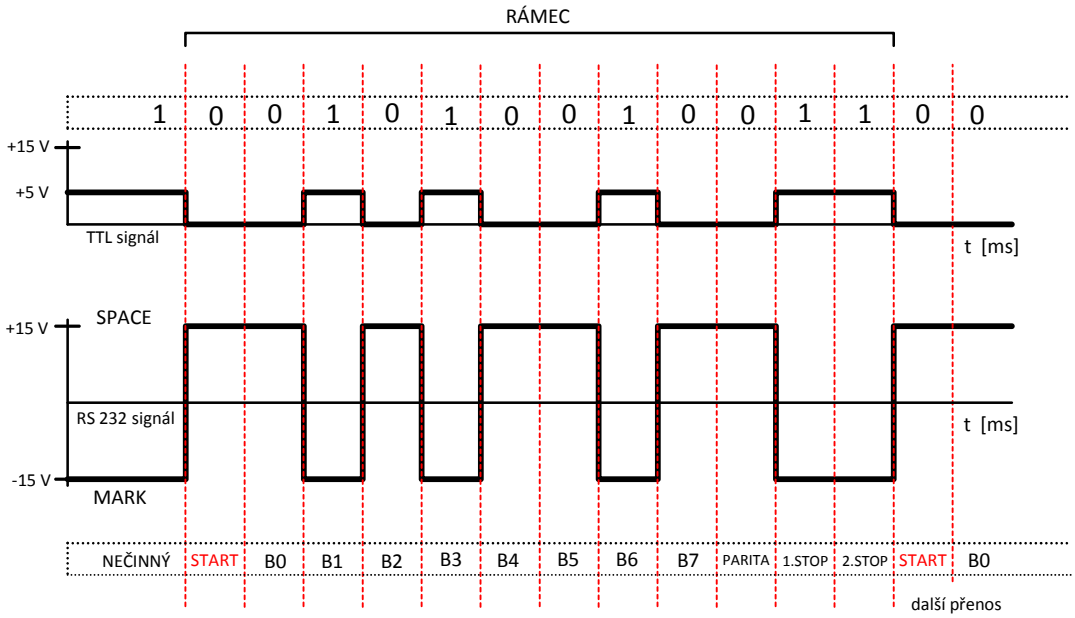
Obrázek 5.6: Sériový port RS-232

Tabulka 5.5 oznamuje, kterými piny je propojen obvod FPGA s převodníkem Maxim. Obvod Maxim obsahuje 2 převodníky TTL->RS232 a 2 převodníky RS232->TTL. Jednu dvojici (RXD, TXD) máme vyvedenou na konektor DB 9, jak ukazuje obrázek 5.6. Zbylou dvojici máme na desce vyvedenou na konektorové kolíky, [4].

Signal	FPGA Pin
RXD	T13
TXD	R13
RXD-A	N10
TXD-A	T14

Tabulka 5.5: Spojení MAX3232 se Spartanem-3 FPGA

Obrázek 5.7 ilustrativně zachycuje generovaný výstupní signál z obvodu FPGA. Tento signál je tedy přiváděn na vstup napěťového konvertoru MAXIM. Tímto konvertorem je dále převáděn na napěťový standard formátu RS-232.



Obrázek 5.7: Přenos jednoho osmibitového slova s lichou paritou, dvěma stop bity a začínajícím dalším přenosem.

5.2.1.3 Oscilátor

Součásti Spartan-3 Starter Kitu je sériový hodinový oscilátor 50 MHz Epson SG - 8002JF, [4]. Ten budeme v našem případě používat jako hodinový signál. Dále je možno desku dovybavit dalším oscilátorem, který lze umístit do připravené patice (IC8), v našem případě tomu tak nebude.

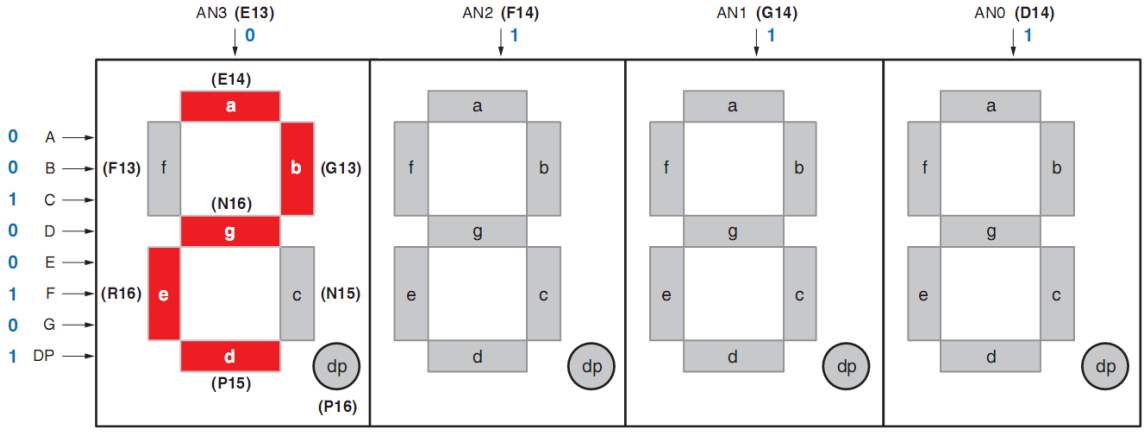
Oscilátor	FPGA Pin
50 MHz (IC4)	T9
Socket (IC8)	D9

Tabulka 5.6: Spojení se Spartanem-3 FPGA

5.2.1.4 Čtyřmístný sedmisegmentový LED displej

Protože vysílač nastavujeme do nejrůznějších vysílacích módů, a těchto módů je celá řada, musíme zajistit jejich zobrazení právě pomocí LED displeje.

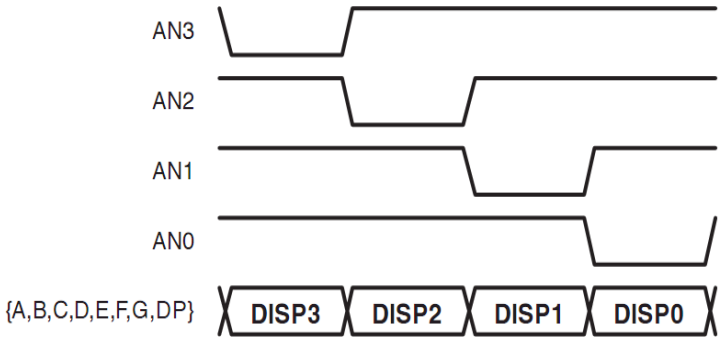
Každý číselný segment sdílí osm vstupů, které se přivádí na jednotlivé LED segmenty. Tyto číselné segmenty mají oddělený anodový řídící vstup. Zobrazení dat na displeji zachycuje obrázek 5.8 a obrázek 5.9. Pokud tedy chceme zobrazit na displej číslo dva, musíme na osm vstupů přivést požadované logické úrovně a současně také na anodový vstup (AN3) logickou 0.



Obrázek 1.8: Čtyřmístný sedmisegmentový LED displej

Data, která chceme zobrazit na displej, musíme časově multiplexovat pomocí řídících anodových vstupů. Na jednotlivé anodové vstupy přivádíme postupně impulsy o nízké úrovni (tzn. logické 0). Tyto signály, kterými řídíme zobrazení na sedmisegmentovém displeji, jsou zachyceny na obrázku 5.9.

Jednotlivý segment je rozdělen na osm částí, kterým je přiděleno písmeno A-G a poslední části je přiděleno označení DP. Každá část segmentu je spojena s patřičným pinem FPGA. Což zachycuje tabulka 5.7, [4].



Obrázek 5.9: Řídící anodové signály, které aktivují jednotlivé číselné segmenty

Segment	FPGA Pin
A	E14
B	G13
C	N15
D	P15
E	R16
F	F13
G	N16
DP	P16
Anode Control	FPGA Pin
AN3	E13
AN2	F14
AN1	G14
AN0	D14

Tabulka 5.7: Spojení čtyřmístného sedmisegmentového LED displeje se Spartanem-3 FPGA

6 Požadavky navrženého vysílače na potřebu strukturních prvků v PLD obvodech

Obvody typu CPLD patří do skupiny elektricky reprogramovatelných PLD obvodů. Většina CPLD obvodů je programovatelná přímo v cílovém systému, nesou tedy označení ISP. Tyto obvody jsou typické svou programovatelnou maticí hradel AND následovanou hradlem OR a makrobuňkou. Na výstupu hradla OR je formována požadovaná logická funkce ve tvaru součtu součinu, [6]. Jednotlivé bloky jsou analogické obvodům SPLD. Uvádí se, že z hlediska zpoždění jsou obvody CPLD jednodušší než obvody FPGA, rovněž algoritmy implementace jsou snadněji zvládnutelné, [7].

Spotřeba strukturních prvků byla provedena pro dvě v dnešní době často používané CPLD řady. Tedy pro řadu *9500XL* a řadu *XC2 (CoolRunner-II).*

Pro zjištění spotřeby strukturních prvků byl využit software ISE od firmy XILINX, který toto nabízí. Návrh vysílače byl implementován[7] (v případě CPLD fitoval) do obvodu CPLD řady *9500XL*. V případě úspěšné fitace se vytvoří zpráva s kompletní dokumentací realizovaného návrhu (*Fitter Report*). Ve zprávě byl doporučen obvod *XC9572XL-5-PC44* a jeho obsazenost strukturními prvky. Návrh vysílače v obvodu zabral 43 makrobuněk a použil 44 součinových termů (Product Terms). Další parametry, jako počet použitých registrů, pinů a funkčních bloků jsou uvedeny v tabulce 6.1, kde je zobrazeno i procentuální využití. To stejné bylo provedeno i s řadou XC2. Výsledky jsou uvedeny v tabulce 6.2.

Device Used	XC9572XL-5-PC44			
Macrocells Used	Product terms Used	Registers Used	Pins Used	Function Block Inputs Used
43/72 (60%)	44/360 (13%)	35/72 (49%)	13/34 (39%)	83/216 (39%)

Tabulka 6.1: XC9572XL-5-PC44

Device Used	XC2C64A-5-PC44			
Macrocells Used	Product terms Used	Registers Used	Pins Used	Function Block Inputs Used
49/64 (77%)	80/224 (36%)	35/64 (55%)	13/33 (40%)	89/160 (56%)

Tabulka 6.2: XC2C64A-5-PC44

[7] Zde se implementuje vysílač bez periférií, tedy nezbytných částí pro funkci vysílače.

6.1 Řada 9500XL

Pracovní napájecí napětí jádra: VCCINT = 3,3 V
Pracovní napájecí napětí vstupně-výstupních bloků: VCCIO = 2,5 až 3,3 V
Pracovní rozsah vstupního napětí: 0 až 5,5 V, úrovně kompatibilní s TTL, možnost přímého připojení k výstupům obvodů s napájením 5 V
Dovolený rozsah výstupního napětí 0 V až VCCIO
Libovolný postup zapínání a vypínání zdrojů
Odběr řádu desítek mA (9536XL: 20-50 mA v normálním módu, 10-30 mA v nízkopříkonovém módu; levá hodnota – statický odběr, pravá – při nejvyšším kmitočtu)
Nízkopříkonový mód nastavitelný individuálně pro jednotlivé makrobuňky
Rozsah počtu makrobuněk: 36 až 288
Klopné obvody D/T, Clock Enable, synchronní nebo asynchronní reset a preset
Programování ISP JTAG
Podpora Boundary Scan
Zaručován počet 10000 cyklů mazání/programování
Zaručena doba uchování obsahu konfigurace 20 let, [7]

6.2 Řada XC2 (CoolRunner-II)

Pracovní napájecí napětí jádra: VCCINT = 1,8 V
Pracovní napájecí napětí vstupně-výstupních bloků: VCCIO = 1,5 až 3,3 V
Počet bank vstupně-výstupních bloků: 1 až 4, každá může mít svou velikost napájecího napětí z uvedeného rozsahu, takže umožňuje přímé připojení obvodů s jmenovitým napájením 1,5; 1,8; 2,5; 3,0 a 3,3 V
Libovolný postup zapínání a vypínání zdrojů
Statický odběr menší než 100 uA
Rozsah počtu makrobuněk: 32 až 512
DataGATE – řízení vstupů pro snížení spotřeby
Vstupy mohou být nakonfigurovány do funkce Schmittova KO
Klopné obvody D/T/Latch, Clock Enable, asynchronní reset a preset
Možnost řízení registrů oběma hranami hodinového signálu (DualEDGE triggered registers)
Dělič hodinového kmitočtu číslem 2, 4, 6, 8, 10, 12, 14, 16, 33
CoolCLOCK – snížení spotřeby rozvodu hodinového signálu současným užitím dělení kmitočtu dvěma a volby DualEDGE
Možnost konfigurace výstupů s otevřeným kolektorem pro funkci montážního součinu a buzení LED
Volitelný rys bus-hold nebo slabý pullup na vybraných I/O vývodech
Volitelná možnost konfigurace nevyužitých I/O vývodů jako přídavných zemnicích vývodů
Možnost zasouvání při připojeném napájení – hot plugging
Programování ISP JTAG

Podpora Boundary Scan
Vývojový prostředek WebPACK volně dostupný z www Xilinx
Zaručován počet 1000 cyklů mazání/programování
Zaručena doba uchování obsahu konfigurace 20 let, [7]

7 Ověření funkčnosti návrhu

V této kapitole si ověříme především činnost naprogramovaného vysílače dat formátu RS-232.

7.1 Zadávání a zobrazení dat

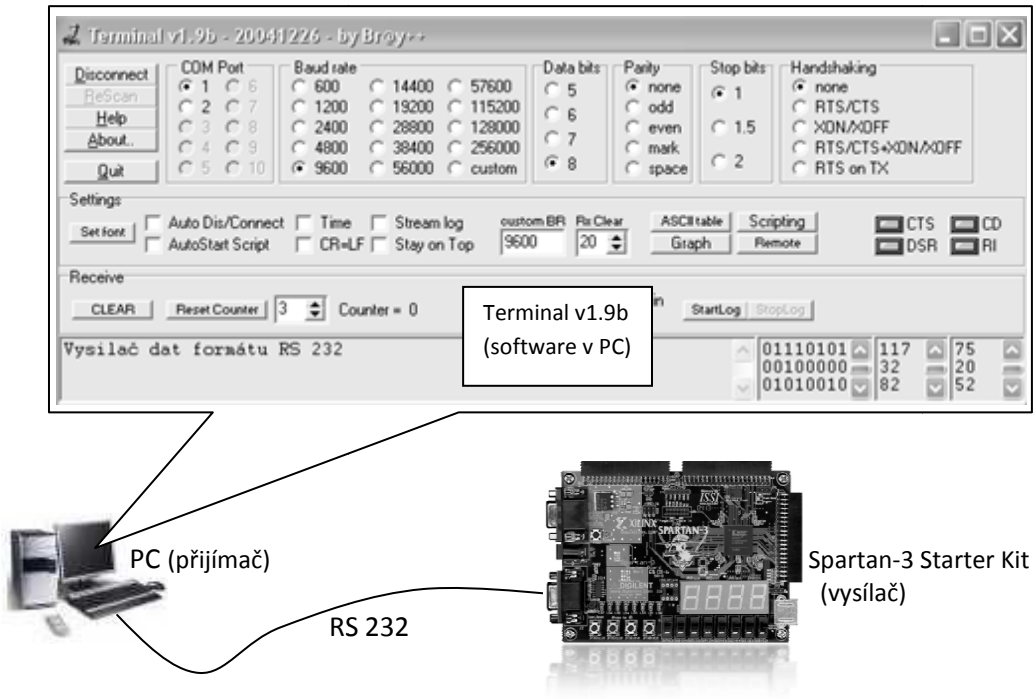
Prostředkem, kterým se budou zadávat data pro odeslání, jsou zvolena posuvná tlačítka, která svoji polohou značí logickou jedničku nebo logickou nulu. Těchto tlačítek je na kitu osm. Tím pádem můžeme posílat všechny ASCII znaky do přijímače, který tvoří počítač s patřičným softwarem. Pro příjem vysílaných dat přes rozhraní RS-232 existuje celá řada softwaru. Já osobně jsem používal dva, a to software, který je přímo součástí Windows. Najdeme ho v příslušenství a jmenuje se Hyperterminal (verze 5.1). Druhý software, který byl častěji použit, se jmenuje Terminal (verze v1.9b). U tohoto softwaru se dají mnohem lépe měnit parametry přijímaného signálu. Není tedy potřeba přerušovat spojení mezi kitem a Terminálem. Parametry *Baud rate*, *Data bits, Parity* a *Stop bits* můžeme okamžitě měnit při navázaném spojení. Přijímaný signál se zobrazuje přímo v okně, kde můžeme volit formát zobrazení. Můžeme zobrazovat přímo ASCII znaky a současně binární kód, který byl vysílán, popřípadě můžeme zobrazit dekadické číslo a další.

Na kitu jsou čtyři mikrotlačítka, která slouží pro ovládání vysílače. Mikrotlačítko pod označením *BTN3* slouží jako *reset*. Tímto tlačítkem uvedeme vysílač do počátečního stavu. V tomto stavu je vysílač nastaven na osm informačních bitů, parita je zakázána a je použit jeden stop bit, to vše je přenášeno rychlostí 9600Bit/s. Tlačítkem *BTN0* se povoluje vysílání. Pokud ho krátce zmáčkneme, vyšle se pouze jeden rámec, při delším podržení začne vysílat rámce jeden za druhým. Další tlačítko *BTN1* je určeno pro přepínání v hlavním MENU. Na první pozici se nastavuje počet informačních bitů, na druhé pozici se nastavuje přenosová rychlost, na třetí pozici se nastavuje parita, na čtvrtém místě nastavujeme počet stop bitu a nakonec v páté poloze povolujeme a zakazujeme vysílání. Všechny tyto polohy jsou signalizovány na sedmisegmentovém LED displeji za pomocí teček na jednotlivých segmentech. V každé poloze MENU se rozsvítí patřičná tečka na LED displeji, pouze v páté poloze MENU se rozsvítí všechny čtyři tečky. Tlačítko *BTN2* pak slouží k vybrání konkrétní možnosti. Nabízené možnosti jsou zobrazeny na sedmisegmentovém displeji formou čísel

nebo počátečních písmen. Těmto číslům a počátečním znakům odpovídá konkrétní nastavení, viz příloha1.

7.2 Ověření funkčnosti návrhu

Funkčnost návrhu byla ověřena tak, že jsem kit propojil s počítačem přes rozhraní RS-232. Na počítači jsem měl spuštěn zmiňovaný software, pomocí kterého byla zobrazena vyslaná data. Na naprogramovaném kitu jsem zadával data binárně pomocí osmi posuvných přepínačů. Nastavil jsem binární hodnotu *01000010*. Tato binární hodnota v ASCII tabulce symbolizuje velké písmeno A. Nastavil jsem stejné parametry na mém vysílači i na softwaru v počítači. Povolil jsem odesílání pravým mikrotlačítkem. Počítač tento znak přijal správně. Tento postup jsem všemožně kombinoval, tím jsem ověřil správnou funkci vysílače dat RS-232.



Obrázek 7.1: Propojeni kitu a počítače rozhraním RS-232

8 Závěr

Úkolem této bakalářské práce bylo navrhnout vysílač sériových dat ve formátu RS-232. Tato problematika zahrnuje celou řadu variant řešení. Cílem této práce bylo tedy navržení vysílače sériových dat ve formátu RS-232 a vytvoření projektu vysílače dat tohoto formátu ve vývojovém prostředí Xilinx ISE WebPACK s možností nastavit detaily formátu vysílaných dat vstupními signály. Pro realizaci byl použit vývojový kit Spartan-3 Starter, který je určena pro ověření funkčnosti návrhu.

Bakalářská práce zahrnuje problematiku sériového vysílače. Dále jsem se zaměřil na podrobný popis jednotlivých kroků návrhu řešení a způsobů implementace. Mnou navržený a implementovaný sériový vysílač umožňuje nastavení jednotlivých parametrů, konkrétně počtu přenášených informačních bitů, nastavení parity, nastavení stop bitu a přenosové rychlosti. Tyto parametry jsou měněny pomocí tlačítek umístěných na vývojovém kitu.

Mým úkolem bylo prozkoumat také požadavky navrženého vysílače na spotřebu strukturálních prvků v různých cílových obvodech PLD. Byly vybrány dvě řady obvodů PLD, řada *9500XL* a řada *XC2 (CoolRunner-II).* Z každé řady byl vybrán konkrétní obvod. V prvním případě obvod *XC9572XL-5-PC44*, kde se spotřebovalo 43 makrobuněk a 44 součinových termů. Tyto dva parametry patří k nejdůležitějším, které charakterizují spotřebu strukturních prvků. Podobně byl zvolen obvod i pro druhou řadu, tedy konkrétně obvod *XC2C64A-5-PC44.* Zde se spotřebovalo 49 makrobuněk a 80 součinových termů.

Funkčnost vysílače byla ověřena za pomoci vývojového kitu, konkrétně kitu Spartan-3 Starter. Do tohoto kitu jsem naimplementoval naprogramovaný vysílač a tento kit jsem propojil s počítačem přes rozhraní RS-232. Počítač byl vybaven softwarem Terminal, který nám umožňoval příjem dat a tím pádem jejich ověření.

Bakalářská práce mi umožnila nahlédnout do problematiky návrhu projektu až po vlastní implementaci do různých cílových obvodů. Při postupu jednotlivými fázemi jsem se setkával s nejrůznějšími komplikacemi, které bylo nutno řešit. Získal jsem tak cenné zkušenosti, které lze využít v podobných projektech.

Tuto bakalářskou práci je možné dále rozšířit o přijímač a realizovat tak obvod UART (Universal Asynchronous Receiver-Transmitter – univerzální asynchronní přijímač a vysílač). Tento obvod by pak podporoval duplexní komunikaci.

9 Literatura

[1] VLACH, Jaroslav; VLACHOVÁ, Viktorie. *Počítačová rozhraní - přenos dat a řídicí systémy*. Praha, 2000, 175 s.

[2] GOOK , Michael. *Hardwarová rozhraní. Průvodce programátora*. Brno, 2006, 463 s.

[3] GRYGAREK, Petr. *Sériový přenos dat* [online], [cit. 2008-04-23]. Dostupný z WWW: <http://www.cs.vsb.cz/grygarek/PS/sem/sercomm.html>.

[4] *Spartan-3 Starter Kit Board User Guide* [online]. UG130 (v1.1), 2005 [cit. 2008-04-23]. Dostupný z WWW: <http://www.xilinx.com/bvdocs/userguides/ug130.pdf>.

[5] BARTOŇ, Zdeněk. *Návrh digitálních integrovaných obvodů*. Brno, 2000, 88 s.

[6] PINKER, Jiří; POUPA, Martin. *Číslicové systémy a jazyk VHDL*. Praha, 2006, 349 s.

[7] KOLOUCH, Jaromír. *Subsystémy používané v obvodech CPLD a FPGA a techniky jejich implementace* [online], [cit. 2008-03-21]. Scripta electronica. Učební text pro doktorské studium. Dostupný z WWW: <http://www.urel.feec.vutbr.cz/~kolouch/pld/data/DrStudTxt.pdf>.

[8] KOLOUCH, Jaromír. *Programovatelné logické obvody*. Brno, 2006, 76 s.

[9] KOLOUCH, Jaromír. *Programovatelné logické obvody a návrh jejich aplikací v jazyku VHDL*. Brno, 2006, 85 s.

[10] HLAVA, Jaroslav. *Prostředky automatického řízení II. Analogové a číslicové regulátory, elektrické pohony, průmyslové komunikační systémy*. Praha 2000.

[11] DEMBOWSKI, Klaus; MESSNER, Hans-Peter. *Velká kniha hardware*. Brno, 2005, 1224 s.

[12] HAZDRA, Pavel. *Syntéza integrovaných elektrických systém.* [online], [cit. 2008-02-01]. Dostupný z WWW: http://www.micro.feld.cvut.cz/home/hazdra/x/cviceni/dokumenty/LC1.pdf

10 Seznamy zkratek, symbolů.

ASCII American Standard Code for Information Interchange
americký standardní kód pro výměnu informací.

CPLD Complex PLD
komplexní programovatelný logický obvod

EIA Electronic Industries Alliance
sdružení v odvětví elektroniky

EN Enable
povolit činnost

FPGA Field Programmable Gate Array
programovatelné hradlové pole

HDL Hardware Description Language
jazyk pro popis hardware

ISP In System Programmable
programovatelný v systému

JTAG Joint Test Action Group
Skupina výrobců integrovaných obvodů a jejich prodejců

MARK Marking state
značí logickou 1

PLD Programmable Logic Device
programovatelný logický obvod

reset
nulování

RTL Register Transfer Language
jazyk RTL

SPACE Space state
značí logickou 1

SPLD Standard PLD
standardní programovatelný obvod logický obvod

TIA/EIA Telecommunications Industry Association – Electronic Industries Association

VHDL VHSIC Hardware Description Language
HDL jazyk, produkt projektu VHSIC

VHSIC Very High Speed Integrated Circuit
velmi rychlý integrovaný obvod – projekt, z něhož vznikl jazyk VHDL

11 Seznam příloh

Příloha 1. Návod na obsluhu vysílače dat formátu RS 232.

Příloha 2. Výstup simulace vysílače

Příloha 3. CD