VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Tomáš Ostřížek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

TESTOVACÍ PLATFORMA PRO BOARD-LEVEL TESTY
SW/HW TOOLSET FOR BOARD-LEVEL TESTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Tomáš Ostřížek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Marek Bohrn, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**

Ústav mikroelektroniky

Student: Bc. Tomáš Ostřížek ***ID:*** 177377

Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Testovací platforma pro board-level testy

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvoření sady softwarových a hardwarových nástrojů pro automatické testování elektronických modulů.
Hardwarová část bude obsahovat řídicí obvod typu Zynq System-on-Chip. Dále budou implementovány potřebná rozhraní dle norem ECSS (např. MIL 1553B, SpaceWire, RS422, SBDL, BLD, SHP, ELP, SLP, LVDS) a rozhraní pro připojení k PC.
Dále bude navržena vrstvová architektura řídícího softwaru pro automatické testy. Posledním cílem práce je vytvoření programového vybavení pro ověření funkčnosti HW rozhraní.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Marek Bohrn, Ph.D.

doc. Ing. Lukáš Fujcik, Ph.D.
předseda oborové rady

ABSTRAKT

Tato práce se věnuje návrhu testovací platformy pro monitorování a ovládání vybraných rozhraní při vývoji pro vesmírné aplikace. Požadavky na implementaci těchto rozhraní vycházejí z ECSS, IEEE a TIA standardů rozebraných v teoretické části práce. Testovací zařízení navržené v této práci je řízeno obvodem SoC Xilinx Zynq-7000 a komunikuje s počítačem prostředníctvím Ethernetu. Hardware zařízení, navržený na obvodové úrovni, zajišťuje splnění příslušných norem. Softwarovou část tvoří knihovna pro řídicí počítač v jazyce Python, jejíž funkce jsou základními stavebními bloky pro tvorbu testovacích procedur, a C program pro ARM procesor předávající data přes AXI rozhraní programovatelné logice s vytvořenými řadiči jednotlivých rozhraní.

KLÍČOVÁ SLOVA

board-level testování, vesmírné aplikace, Ethernet, SpaceWire, MIL-STD-1553B, system-on-chip

ABSTRACT

This thesis describes the design of a board-level testing platform for monitoring and driving a selected set of interfaces used in space applications. The requirements of these devices are based on the corresponding ECSS, IEEE, and TIA standards described in the theoretical part of this thesis. The designed testing device is controlled by the Xilinx Zynq-7000 system-on-chip and is connected to a control PC via an Ethernet connection. The hardware, designed on a schematic level is responsible for meeting the standards' requirements. The software part consists of a Python module for the control PC providing a set of functions to be used in the testing process and a C application for the embedded ARM processor that forwards the data through the AXI interface to the interface drivers in the programmable logic.

KEYWORDS

board-level testing, space applications, Ethernet, SpaceWire, MIL-STD-1553B, system-on-chip

OSTŘÍŽEK, Tomáš. *Testovací platforma pro board-level testy*. Brno, 2018, 66 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav mikroelektroniky. Vedoucí práce: Ing. Marek Bohrn, PhD.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	Application programming interface
AXI	Advanced extensible interface
BC	Bus controller
BLD	Bi-level digital
BM	Bus monitor
DHCP	Dynamic host configuration protocol
ECSS	European Cooperation for Space Standardisation
EEP	End of error packet
EHP	Extended high power
EMIO	Extended multiplexed input/output
EOP	End of packet
FCT	Flow control token
FPGA	Field-programmable gate array
GPIO	General purpose input/output
IEEE	Institute of Electrical and Electronics Engineers
LVDS	Low voltage differential signaling
MDI	Media dependent interface
MII	Media independent interface
MIO	Multiplexed input/output
TCP/IP	Transmission control protocol / Internet protocol
PLL	Phase locked loop
PS	Processing system
RGMII	Reduced gigabit media independent interface
RT	Remote terminal
SBDL	Standard balanced digital link
SHP	Standard high power
SLP	Standard low power
SoC	System on chip
UART	Universal asynchronous receiver transmitter
USB	Universal serial bus

ÚVOD

Předmětem této práce je návrh systému pro testování jednotek na úrovni komunikačních a ovládacích rozhraní. Výběr těchto rozhraní je součástí zadání práce a vychází ze zamýšleného použití testovacího zařízení při vývoji pro vesmírné aplikace. Úkolem zařízení navrženého v této práci je umožnit vývojovému týmu snadné ovládání a monitorování toku dat mezi jednotlivými součástmi testované jednotky za nutnosti minimálního vývoje dodatečného hardwaru a softwaru. Testovací zařízení zajišťuje převod jednoduše popsatelných testovacích programů přes softwarové vrstvy operačního systému, jejich přenos pomocí sjednoceného komunikačního protokolu a jejich provedení v koncovém bodě připojeném k testované jednotce.

Koncepčně lze testovací systém rozdělit na hardwarovou a softwarovou část. Software v systému slouží k usnadnění tvorby testovacích procedur použitím vhodných funkcí z vytvořené programové knihovny. Hardwarová část zařízení je tvořena především obvody zajišťujícími soulad fyzických vrstev rozhraní s normami.

Specifikace rozhraní používaných ve vesmírných aplikacích, které jsou v testovacím systému implementovány, je popsána zejména v ECSS normách, případně původních standardech, na které tyto normy navazují. ECSS je organizace zajišťující standardizaci v evropském vesmírném sektoru zejména pro potřeby Evropské vesmírné agentury. Její normy jsou závazné pro veškerý vývoj ve vesmírných programech ESA a obsahují požadavky na elektrické, mechanické nebo radiační parametry vyvíjených zařízení. Pro testovací systém popsaný v této práci spočívá význam těchto norem v popisu specifikace komunikačních rozhraní na úrovni obvodového zapojení, typu signalizace a vyšších komunikačních vrstev. Soulad s normami není pro testovací zařízení vyžadován, dodržení parametrů pouze zajišťuje kompatibilitu spojení s testovanou jednotkou.

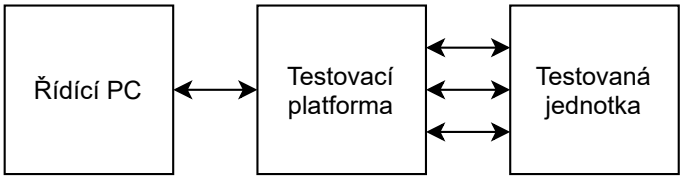
Použití testovacího systému je směřováno na budoucí projekty vyvíjené Ústavem mikroelektroniky. Tyto projekty se mohou lišit v konkrétní implementaci komunikačních rozhraní, tedy v jejich počtu i typu. Z toho důvodu není cílem této práce návrh fyzického hardwarového řešení, nýbrž návrh takového systému, který lze díky své modularitě pro konkrétní projekt snadno upravit nebo rozšířit bez nutnosti opakovaného vývoje základní funkčnosti.

Stejně jako u hardwarové stránky je cílem softwarového návrhu systému umožnit minimální časovou náročnost testovacího procesu při samotném vývoji zařízení. Tento úkol plní v systému softwarová knihovna navázaná na hardwarový návrh, která poskytuje testovacímu týmu jednoduché funkce, pomocí kterých je možné zařízení ovládat.

Pojmy používané v této práci pro jednotlivé součásti systému vychází z překladů obecně používané terminologie nebo z konceptu platformy. Řídicím počítačem je v této práci myšlen počítač, na kterém je spuštěna testovací procedura skládající se z funkcí uživatelské knihovny. Tento počítač pomocí protokolů rozhraní Ethernet komunikuje s testovacím zařízením. Toto testovací zařízení je předmětem hardwarové části návrhu této práce a jedná se o převodník sjednoceného komunikačního protokolu Ethernetu na specifické protokoly vyžadované koncovými rozhraními. Těmito rozhraními jsou myšlena rozhraní vybraná jako součást zadání této práce, která jsou přímo spojena s testovanou jednotkou. Testovaná jednotka je pak samotné zařízení, jehož funkčnost je testovacím systémem ověřována, tedy zařízení vyvíjené pro vesmírnou aplikaci. Testovací zařízení je postaveno na obvodu SoC Xilinx Zynq. Tento obvod je v této práci popisován jako řídicí obvod. Jeho součástí je ARM procesor, který zprostředkovává funkčnost spojenou s rozhraním Ethernet a programovatelná logika obsahující řadiče jednotlivých koncových rozhraní.

1 CELKOVÝ KONCEPT PLATFORMY

Testovací platforma zajišťuje propojení řídicího počítače s testovanou jednotkou. Spojení s PC je zajištěno pomocí rozhraní snadno ovládatelného z testovacího softwaru. Na straně testované jednotky je tato platforma připojena přes řadu rozhraní podle požadavků zadání. Úkolem této platformy je dostatečně rychlé obousměrné předávání dat mezi testovanou jednotkou a počítačem tak, aby bylo možné cílová rozhraní v reálném čase řídit i monitorovat. Základní koncept celého systému je na obrázku 1.1.



Obr. 1.1: Zakladní schéma koncepce testovacího systému

Platforma navržená v této práci je modulární a její přesná podoba může být upravena ve smyslu přidání nebo odebrání cílových rozhraní podle specifikace konkrétního projektu.

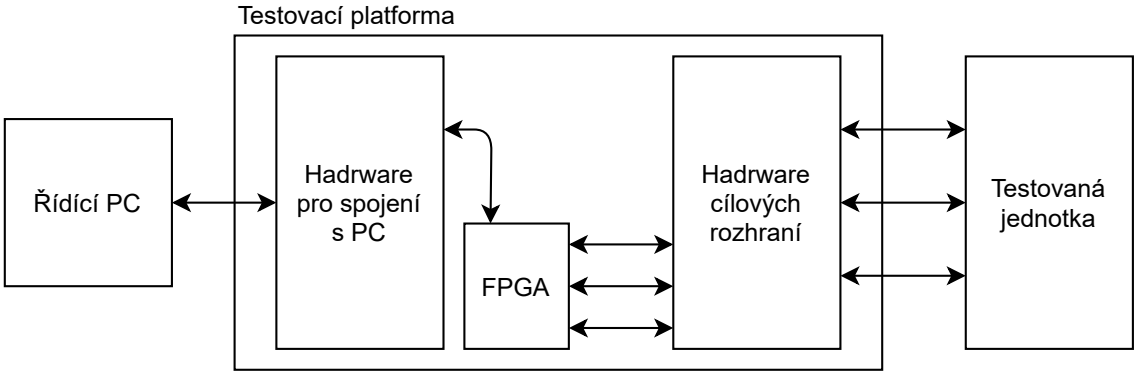
1.1 Požadavky na testovací platformu

Požadavky na testovací platformu vyplývají ze zadání práce. Na straně řídicího počítače je nutné zajistit dostatečně rychlou datovou komunikaci pro potřeby testovacího procesu a zároveň využít vhodnou formu spojení pro univerzální využití v testovacím softwaru.

Na straně cílových rozhraní je platforma navržena jako univerzální zařízení pro použití na libovolném projektu využívající tato rozhraní. Princip tohoto řešení spočívá v naddimenzování počtu všech potřebných rozhraní tak, aby bylo pomocí jednoho zařízení možné komunikovat se všemi potřebnými koncovými body testované jednotky.

Zadání práce definuje cílová rozhraní podle plánovaného použití testovací platformy na projektech pro vesmírné aplikace. Z toho vychází seznam těchto rozhraní. Zamýšlený princip propojení řídicího počítače s testovací platformou pomocí jednoho datového spojení má za důsledek velkou paralelizaci datových

spojení. Tento požadavek vylučuje použít jako základ platformy mikrokontrolér nebo jiné zařízení neschopné komunikovat se všemi koncovými rozhraními současně. Koncept zařízení je proto založen na programovatelném hradlovém poli, které je schopné multiplexovat a demultiplexovat komunikaci mezi všemi rozhraními a datovým spojením s počítačem. Blokové schéma testovacího systému postaveného na hradlovém poli je na obrázku 1.2.



Obr. 1.2: Podrobná koncepce testovacího systému

Práce se zabývá návrhem konceptu testovací platformy a dílčích obvodových řešení. Získaná obvodová řešení je pak možné při fyzické realizaci použít jako moduly a výsledné zařízení přizpůsobit aplikaci jak na úrovni modulů, tak i s případnými menšími úpravami uvnitř modulů samotných.

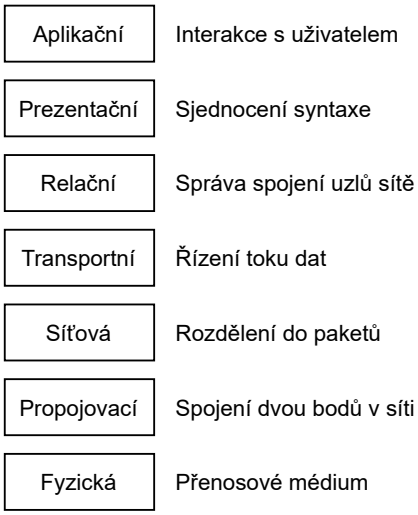
2 TEORETICKÝ ROZBOR POŽADAVKŮ

Tato kapitola rozebírá požadavky (zejména dle příslušných norem), které je nutné dodržet při návrhu testovacího systému. Vzhledem k tomu, že rodina obvodu pro řízení je určena v rámci zadání práce, jedná se zejména o požadavky na potřebná rozhraní. Upřesnění parametrů řídicího obvodu je v kapitole 3. Ty jsou rozděleny do dvou částí - spojení hradlového pole s řídícím PC přes Ethernet a spojení hradlového pole s koncovými rozhraními testované jednotky.

2.1 Ethernet

Většina požadavků spojená s přenosem dat mezi počítačem a testovací platformou pramení z požadované rychlosti spojení pro odbavení všech požadavků v reálném čase. Hradlové pole zajišťující předávání dat je připojeno k velkému množství rozhraní komunikujících na rychlostech až 200 Mbit/s. [6]

Dalším požadavkem je snadné posílání a čtení dat z testovacího programu běžícího v řídicím počítači. Pro tento účel bylo vybráno rozhraní Ethernet, jehož nejnižší vrstvy jsou snadno implementovatelné v hradlovém poli nebo pomocí jednoduchých standardních součástek a které je díky své rozšířenosti možno využít v libovolném programovém prostředí testovacího softwaru.



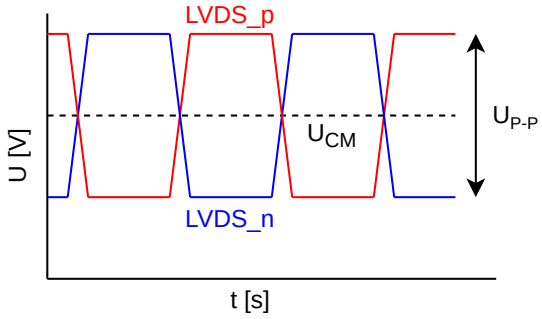
Obr. 2.1: Vrstvy protokolu Ethernet a jejich funkce

Ethernet je síťový protokol definovaný normou IEEE 802.3. Architektura tohoto protokolu se skládá ze sedmi vrstev zobrazených na obrázku 2.1. Tento protokol je využíván zejména v datové infrastruktuře pro vytváření místních sítí a pro připojení počítačů k síti internet. [1]

Nejvyšší čtyři vrstvy Ethernetu jsou v počítači spravovány běžícím softwarem a operačním systémem. Pro základní funkčnost předávání dat není nutná jejich implementace v hradlovém poli. V rámci Ethernetu lze využít řada přenosových protokolů, pro pevné spojení dvou bodů v síti není nutné řídit tok dat ani spravovat spojení s více uzly. K této zjednodušené komunikaci pomocí Ethernetu lze využít protokoly vyšších vrstev *TCP/IP sockets*, *TCP/IP raw* nebo *Telnet*.

2.2 LVDS

LVDS (*low voltage differential signaling* - nízkonapěťový diferenční přenos) je univerzální rozhraní pro komunikaci mezi jedním vysílačem a jedním přijímačem. Parametry jsou definovány v normách IEEE 1596.3 a ANSI/TIA/EIA-644. [2] LVDS signály mají pro napájení $5\,V$ typicky rozkmit $\pm 400\,mV$ a stejnosměrný offset $1,2\,V$. Modifikace povolují využití jednoho diferenčního páru pro obousměrný přenos nebo připojení více přijímačů k jedné přenosové lince. Pro různé konfigurace specifikuje norma doporučenou formu terminace vedení. Časový diagram LVDS komunikace je na obrázku 2.2, kde U_{CM} je stejnosměrný posun signálu a U_{P-P} je mezišpičkové napětí. [3]

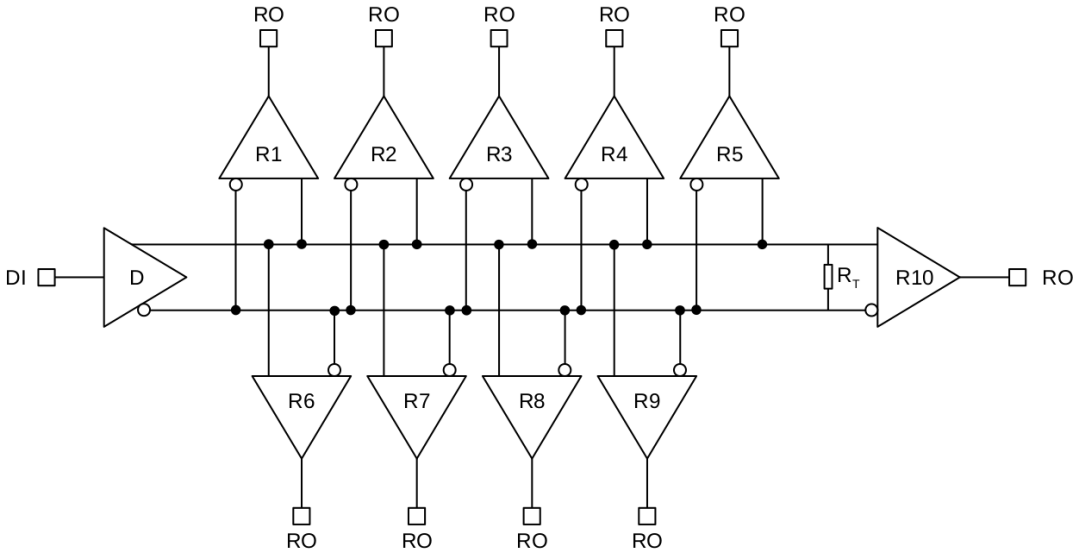


Obr. 2.2: Časový diagram LVDS komunikace

Diferenční komunikace mají smysl zejména u rychlých nebo velmi dlouhých přenosů, které jsou náchylné k rušení. V případě indukování nežádoucího zdroje signálu na přenosové médium jsou napěťově ovlivňovány oba signály diferenčního vedení, nikoli však jejich rozdíl.

2.3 RS-422

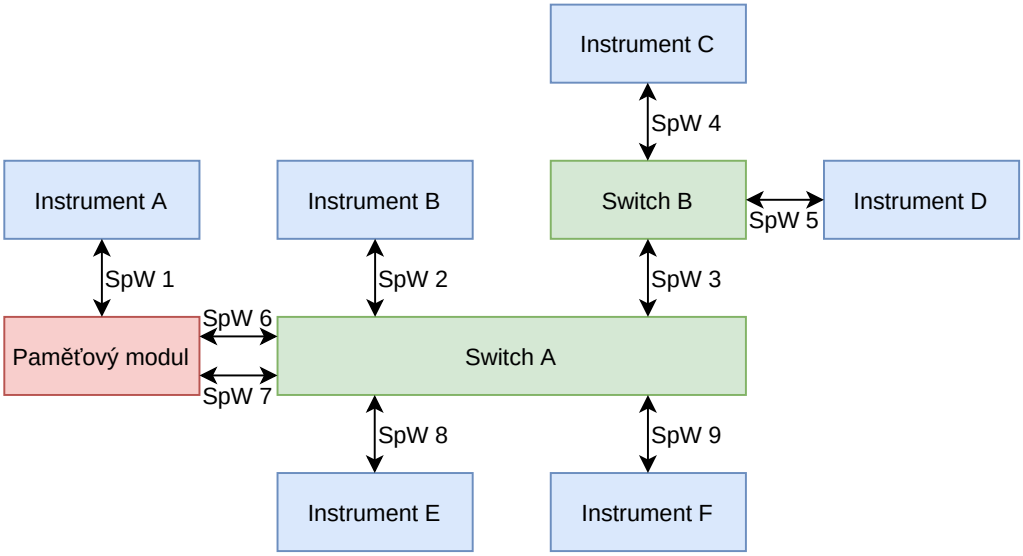
RS-422 je protokol pro diferenční dvouvodičový přenos mezi jedním vysílačem a maximálně deseti přijímači. Datová rychlost dosahuje maximálně $10\,Mbit/s$ a délka vedení až $1,2\,km$. Norma definuje stejnosměrný posun signálu (též napětí *common-mode*) mezi datovými vodiči jako $U_{CM} = (U_{IA} + U_{IB})/2$, kde U_{IA} a U_{IB} jsou napětí na datových vodičích vztažená k signálové zemi. Napětí v obou logických úrovních může dosahovat maximálně hodnoty $\pm 7\,V$. Vedení RS-422 je terminováno na nejvzdálenějším přijímači, přičemž vstupní impedance každého přijímače musí být větší nebo rovna $4\,k\Omega$. Principiální schéma RS-422 systému je na obrázku 2.3. [4][5]



Obr. 2.3: Zapojení a terminace RS-422, převzato z [5]

2.4 SpaceWire

SpaceWire je síťový protokol vyvinutý pro vesmírné aplikace. Zajišťuje přenosové rychlosti od 2 do 200 Mbit/s. Důvodem vývoje tohoto protokolu bylo sjednocení komunikačních propojení mezi jednotlivými instrumenty elektroniky vesmírných projektů. SpaceWire je používán agenturami ESA, NASA i dalšími subjekty v oblasti vesmírného vývoje. [6]



Obr. 2.4: Příklad SpaceWire architektrury, převzato z [6]

Komunikace přes SpaceWire probíhá vždy mezi dvěma uzly, pro větvení komunikační sítě jsou použity přepínače. U instrumentů produkujících velké množství dat je možné paralelizovat komunikaci pomocí paměťového modulu na více komunikačních linek. Příklad jednoduché SpaceWire architektury je na obrázku 2.4.

Protokol SpaceWire je definovaný na několika vrstvách - síťové (*network*), spojovací (*data link*), enkódovací (*encoding*) a fyzická (*physical*). V rámci implementace tyto vrstvy definují jednotlivé úkoly SpaceWire komunikace od paketizace až po specifikace přenosového média. [6]

2.4.1 Síťová vrstva

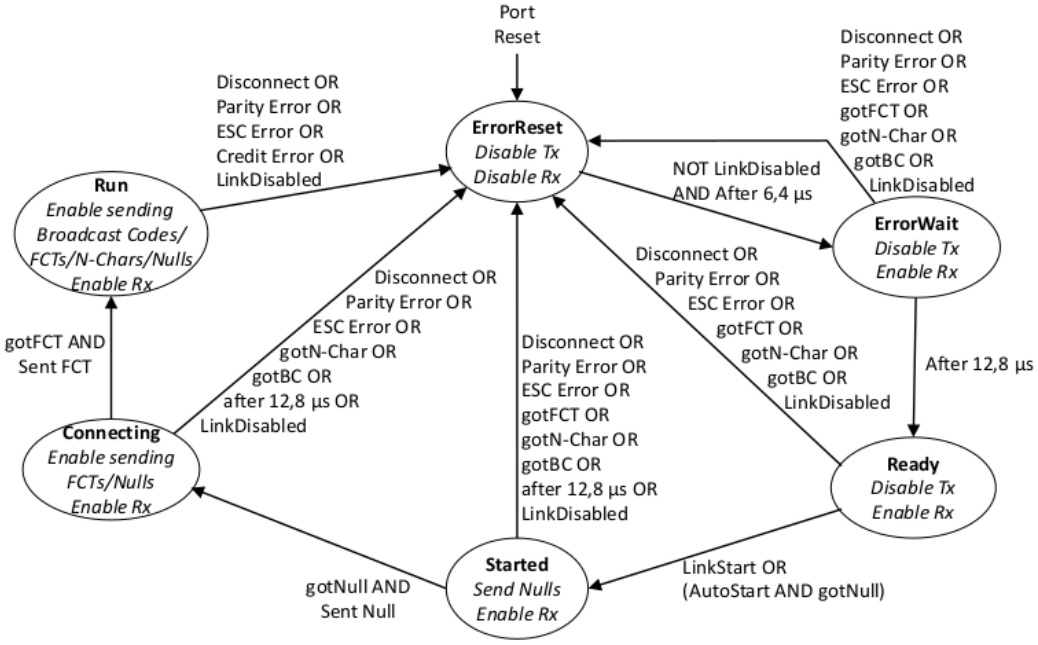
SpaceWire paket se skládá ze tří částí - cílové adresy, vlastních dat a ukončovací značky (EOP/EEP). Cílová adresa obsahuje koncový bod, pro který je paket určen, nebo posloupnost koncových bodů, které má paket projít. Tuto adresu je možné vynechat, pokud je SpaceWire spojení navázáno jen mezi dvěma uzly bez další

propojovací sítě. Vlastní data mohou mít zdola i shora neomezenou délku, musí být zarovnána na celé bajty. Ukončovací značka je posledním znakem ve SpaceWire paketu, jakákoliv následující data jsou považována za data dalšího paketu. Struktura paketu je na obrázku 2.5. [6]

Obr. 2.5: Struktura SpaceWire paketu, převzato z [6]

2.4.2 Spojovací vrstva

Spojovací vrstva SpaceWire protokolu se stará o navázání spojení a jeho udržování po dobu komunikace. Inicializace spojení je realizována na základě vysílání a přijímání řídicích znaků (*null, FCT, N-Char*. Inicializační proces se řídí stavovým diagramem na obrázku 2.6. [6]

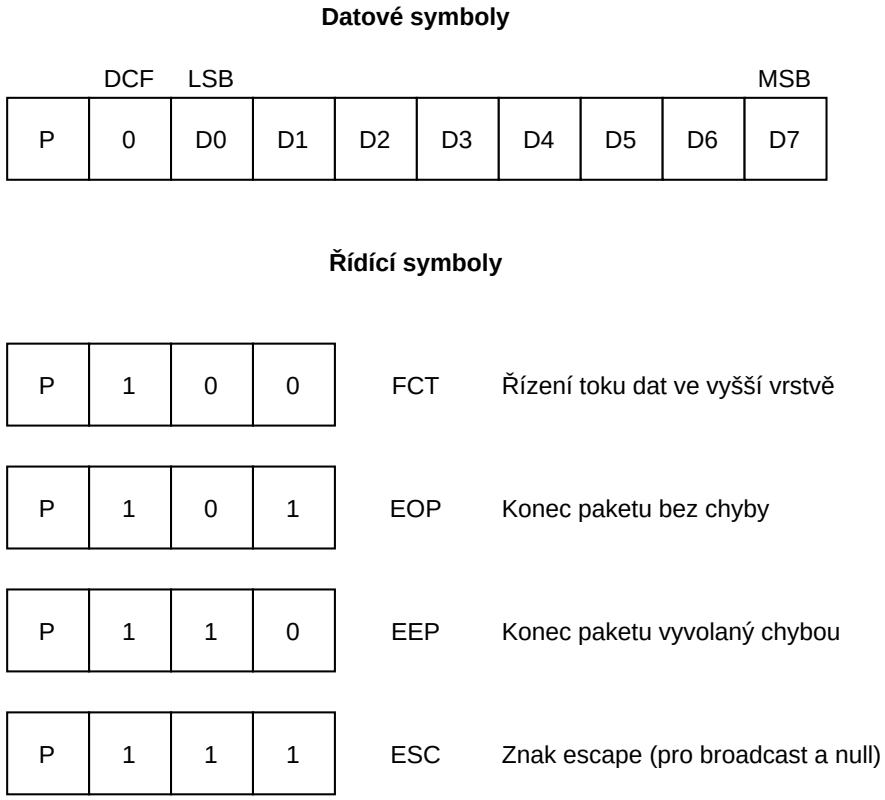


Obr. 2.6: Inicializace SpaceWire spojení, převzato z [6]

2.4.3 Enkódovací vrstva

Enkódovací vrstva protokolu SpaceWire se stará o zakódování jednotlivých slov SpaceWire paketu do přenášených symbolů respektive o jejich zpětné dekódování

na straně přijímače. Datový symbol je složen z 10 bitů - paritní bit, řídicí bit a 8 datových bitů. Řídicí bit je pro datové symboly v logické nule. Kromě datových symbolů je normou definováno několik řídicích symbolů, ty jsou indikovány řídicím bitem v logické jedničce. Struktura datového symbolu a seznam řídicích symbolů je na obrázku 2.7. [6]

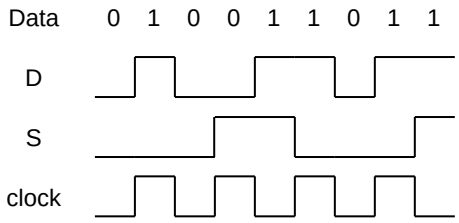


Obr. 2.7: Struktura datového a řídicího symbolu SpaceWire

Datový proud SpaceWire komunikace je enkódovaný pomocí *data-strobe* modulace. Tato modulace je popsána dvěma signály - *data* a *strobe*. Datový signál je ve stejné logické úrovni jako přenášená informace. Signál strobe mění polaritu pokaždé, když je logická úroveň dvou po sobě jdoucích datových bitů stejná. Díky tomu je možné pomocí funkce exkluzivního součtu signálů D a S rekonstruovat hodinový signál clock. Příklad *data-strobe* modulace je na obrázku 2.8. [6]

2.4.4 Fyzická vrstva

Z hlediska uspořádání jednotlivých uzlů je SpaceWire plně duplexním protokolem. Každé spojení je tvořeno dvěma vysílači a dvěma přijímači. Každý přenosový kanál je řešený diferenčním přenosem terminovaným na přijímači.



Obr. 2.8: Data-strobe enkódování SpaceWire signálů, převzato z [6]

Napěťové úrovně SpaceWire signálů jsou převzaty ze standardu TIA-644 pro LVDS. [6]

2.5 MIL-STD-1553B

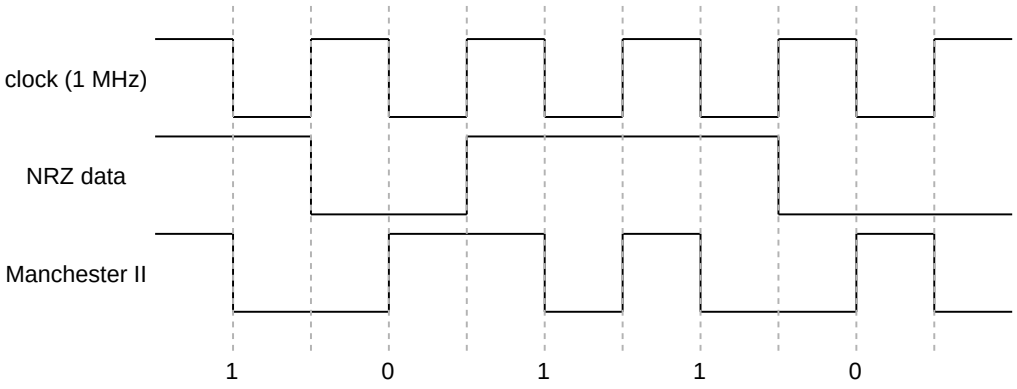
MIL-STD-1553B je sběrnicový komunikační protokol určený pro posílání krátkých zpráv do délky 512 bitů. Standard byl původně vyvinut Ministerstvem obrany Spojených států amerických pro použití ve vojenské sféře. Postupně byl protokol rozšířen do vesmírných aplikací a přesněji specifikován normami ECSS.

Sběrnice tohoto protokolu je poloviční duplexní, data mezi dvěma koncovými body putují po stejných vodičích oběma směry. Kvůli zamezení kolizím na sběrnici je protokol MIL-STD-1553B provozován na principu příkazů a odpovědí. Řízení komunikace je zajišťováno jedním z koncových bodů, nazývanému *bus controller* (BC). Ostatní koncové body jsou nazývány *remote terminal* (RT). Koncové body, které neumožňují odesílání dat a pouze sledují data tekoucí po sběrnici se nazývají BM (*bus monitor*). [7]

Komunikaci po sběrnici lze rozdělit na tři případy. Prvním typem komunikace je časová synchronizace připojených uzlů a posílání časových značek. Tuto komunikaci zajišťuje vždy koncový bod plnící roli BC. Pro posílání krátkých zpráv oběma směry slouží služby *set data* a *get data*. Tento proces je vždy inicializován zařízením BC, u služby *get data* se však jedná o přenos informace ve směru od RT k BC. Posílání delších zpráv do délky 1024 nebo 4096 bajtů podle nastavení sběrnice zajišťuje služba *data block transfer service*.

2.5.1 Spojovací vrstva

Komunikace pomocí protokolu MIL-STD-1553B je realizována sériově 16bitovými slovy předcházenými 3bitovou synchronizační značkou a následovanými jedním paritním bitem. Data na sběrnici jsou při komunikaci kódována pomocí dvoufázového *Manchester II* enkódování. Řídicí hodinový signál protokolu má frekvenci $1\,MHz$. Způsob enkódování slov je na obrázku 2.9. [7]



Obr. 2.9: Enkódování slov v protokolu MIL-STD-1553B, převzato z [7]

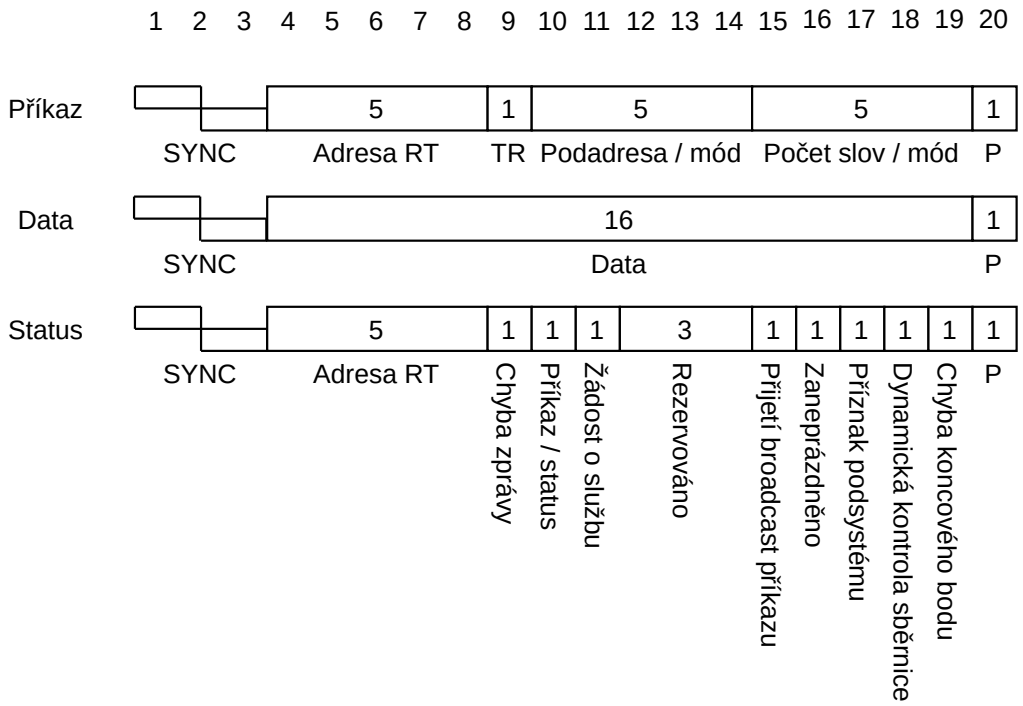
Protokol rozlišuje tři typy slov - příkazy, datová slova a statusová slova. Tyto typy slouží k oddělení datové komunikace po sběrnici a jejího řízení, o které se stará řídící koncový bod (BC). Struktura příkazů, datových a statusových slov je na obrázku 2.10. [7]

2.5.1.1 Definice součástí slov

Součástí MIL-STD-1553B slov jsou některé kontrolní bity nebo skupiny bitů, které určují, jak má koncový bod nakládat s přijatým slovem. Bit TR sděluje koncovému bodu, jestli bude v následující relaci přijímat nebo odesílat data. Bit P je paritní bit a je asociován se všemi druhy přenášených slov. Protokol využívá lichou paritu. Dynamická kontrola sběrnice umožňuje řídícímu koncovému bodu předat kontrolu nad sběrnicí. Desátý bit statusového slova je vždy v logické nule a je použit pro rozlišení příkazů a statusových slov. Žádost o službu je nepovinná součást protokolu, logická jednička na pozici tohoto bitu značí žádost koncového bodu o provedení předem definovaného úkolu v rámci podsystému tohoto koncového bodu. Přijetí broadcast příkazu je indikováno logickou jedničkou na 15. bitu statusového slova. Příznak zaneprázdnění je přenášen na 16. bitu statusového slova a značí neschopnost koncového bodu vysílat nebo přijímat data po sběrnici. [7]

2.5.1.1 Synchronizační značka

Na začátku jakéhokoli slova při přenosu je nejprve odesílána tříbitová synchronizační značka. Tato značka se skládá z jednoho a půl bitu logické nuly a jednoho a půl bitu logické jedničky. Rozpoznání značky na sběrnici je založeno na její nevaliditě z pohledu *Manchester II* enkódování. [7]



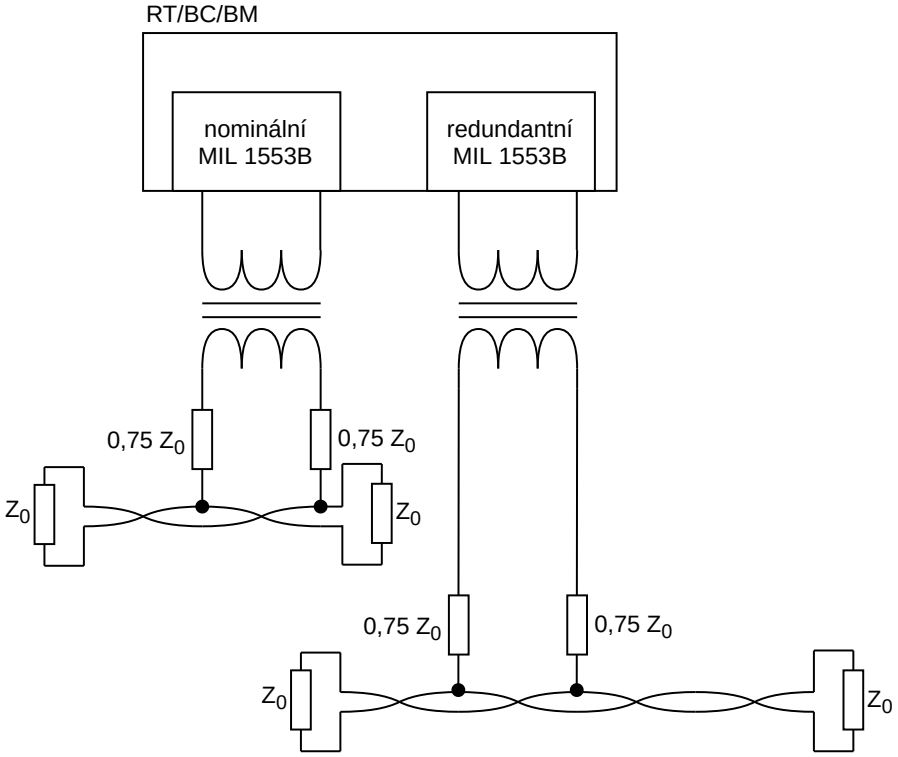
Obr. 2.10: Struktura slov v protokolu MIL-STD-1553B, převzato z [7]

2.5.2 Fyzická vrstva

Systém využívající sběrnici MIL-STD-1553B je sestaven z koncových bodů, z nichž alespoň jeden je schopen vykovávat funkce řídící komunikaci po sběrnici (BC). Koncové body, které tuto funkci nevykonávají jsou označeny RT. Každý koncový bod je připojený ke dvěma fyzickým sběrnicím - jedné nominální a jedné redundantní. Tím je zaručena možnost zachování komunikace při poruše nominální sběrnice. [8]

Ze signálového hlediska je sběrnice tvořena jedním diferenčním párem s impedancí $75 - 80\,k\Omega$ na frekvenci $1\,MHz$. Každý koncový bod je k této sběrnici

připojen přes vazební transformátor a dvojici sériových rezistorů s nominálním odporem $0,75Z_0$, kde Z_0 představuje impedanci sběrnice. Připojení jednoho koncového bodu k nominální sběrnici MIL-STD-1553B je na obrázku 2.11. [8]

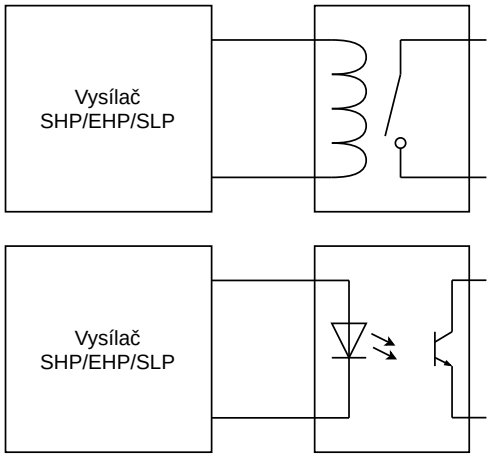


Obr. 2.11: Připojení koncového bodu ke sběrnici MIL-STD-1553B [7]

2.6 Spínání výkonových zátěží

Pro spínání výkonových zátěží je využíváno rozhraní SHP (*standard high power on/off command*), EHP (*extended high power on/off command*) a SLP (*standard low power on/off command*). Jejich specifikace je stanovena z ohledem na spínaní většího výkonu než u datových rozhraní. Buzení signálu musí být odolné přechodným jevům vznikajícím při spínaní záteží indukčního charakteru jako jsou elektromagnetická relé. Základní schéma systému spínajícího výkonovou zátěž je na obrázku 2.12. Označení SHP, EHP a SLP se liší v proudové zatížitelnosti budícího obvodu. [9]

SHP musí být schopné spínat zátěž $\geq 180\,mA$ na $22 - 28\,V$. Délka spínacího pulzu je $32 - 64\,ms$. EHP spíná na $22 - 28\,V$ proud až $360\,mA$ s délkou pulzu $512\,ms$. SLP je určeno pro spínání proudu až $100\,mA$ při napětí $2,5 - 25,1\,V$. Délka pulzu je $32 - 64\,ms$.



Obr. 2.12: Zapojení dvou spínačů rozhraní SHP/EHP/SLP

2.7 SBDL

SBDL (*standard balanced digital link*) je doplnění specifikace standardu RS-422. Tato specifikace je doplněna o hodnotu minimální tolerance zkratového proudu na výstupu vysílače $150\,mA$, výstupní napětí vysílače v intervalu $0 - 7\,V$, toleranci vysílače v intervalu od $-0,5$ do $7\,V$. Přijímač má maximální výstupní chybové napětí $-0,5 - 7\,V$ a toleranci $\pm 12\,V$. [9]

2.8 BLD

BLD (*bi-level digital*) je rozhraní používané pro diskrétní telemetrii, v signálové vrstvě používá standardní dvoustavové logické kódování, kde logická nula je přenášena napěťovou úrovní $0\,V \leq U \leq 0,5\,V$ a logická jednička napěťovou úrovní $2,4\,V \leq U \leq 5,5\,V$. Rozhraní je specifikováno pro každý signálový vodič zvlášť, v konkrétní aplikaci je možné sdružení více signálů k vytvoření paralelního spojení. Vysílač BLD signálů je vztažen k zemi, přijímač je diferenční. Výstupní impedance vysílače musí být $\leq 7,5\,k\Omega$. Vstupní impedance přijímače musí být $\geq 100\,k\Omega$ při příjmu dat a $\geq 20\,M\Omega$ v klidovém stavu. [9]

Důvodem specifikace rozhraní je především zajištění tolerance chyb v připojených obvodech. Výstupní napětí vysílače v chybovém stavu musí být v intervalu od -1 do $12\,V$ a každý připojený bod musí být schopen snést $\pm 16,5\,V$. Napětí injektované přijímačem ve stavu chyby musí být maximálně v rozsahu $\pm 16\,V$ a tolerance chybového napětí na vstupu minimálně $\pm 15,8\,V$. [9]

3 PRINCIP FUNKCE ŘÍDICÍHO OBVODU

Tato kapitola se zabývá funkcí řídicího obvodu testovacího zařízení. Požadavky jsou popsány v koncepci platformy v kapitole 1.1. Vzhledem k násobné paralelizaci komunikačních kanálů v řídicím obvodu z jednoho spojení s řídicím počítačem na jednotlivá koncová rozhraní je nevhodné použití mikrokontroléru. Pro zajištění obsluhy všech rozhraní v reálném čase je žádoucí využití hradlového pole pro vytvoření základních jader nejnižších vrstev těchto rozhraní.

3.1 Spojení s PC

Cílem vytvoření univerzální testovací platformy je usnadnění celého testovacího procesu a zkrácení času přípravy v procesu vývoje testovaného zařízení. Pro splnění tohoto požadavku je nutné zajistit snadný přístup testovacího algoritmu přímo ke koncovým bodům. Ke spojení s řídicím počítačem je vhodné využít takové rozhraní, ke kterému existuje snadný přístup z libovolného programovacího prostředí a jazyka.

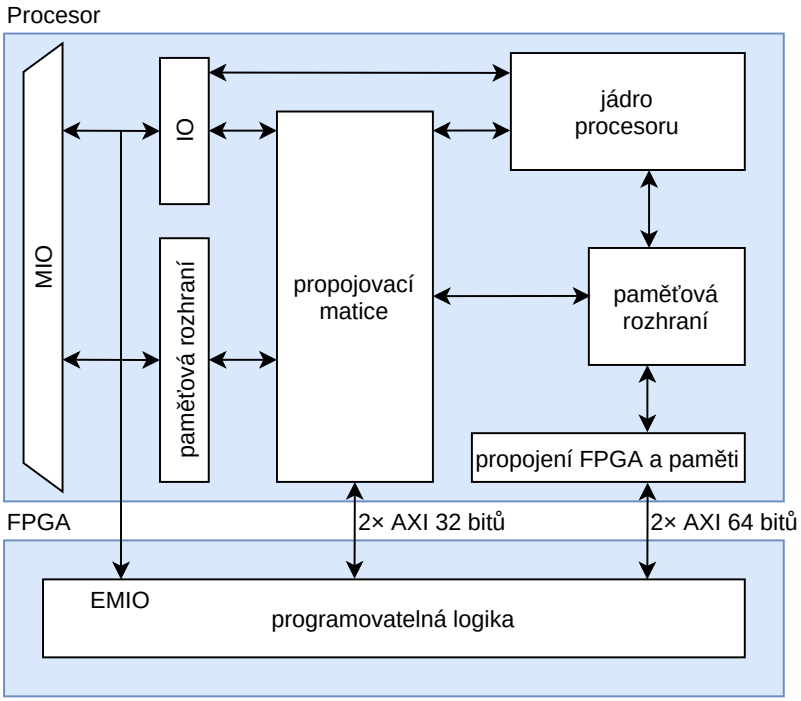
Nejrozšířenějšími standardizovanými rozhraními v osobních počítačích jsou Ethernet a USB. Z důvodu plně duplexní komunikace a větší míry flexibility Ethernetu bylo pro spojení s počítačem vybráno toto rozhraní. Komunikace přes Ethernet vytváří na řídicí obvod několik hardwarových i softawarových požadavků vyplývajících z teoretického popisu v kapitole 2.1.

Ethernet je vrstvový protokol s komplexní strukturou, která má za úkol jeho snadné použití v rozvětvených aplikacích, jako je komunikace přes internet. Pro spojení mezi dvěma koncovými body není nutné implementovat všechny vrstvy tohoto protokolu. V rámci návrhu testovacího zařízení je nutné zajistit pouze navázání spojení a oboustrannou výměnu dat mezi počítačem a testovacím zařízením. Míra implementace protokolu Ethernet je určena nejnižší vrstvou, kterou jde ovládat z testovacího softwaru v PC. Tato vrstva je z praktického hlediska určena operačním systémem.

V praxi je problematické z libovolného programového prostředí přistupovat přímo ke druhé (síťové) vrstvě Ethernetu, která by byla pro *point-to-point* spojení mezi dvěma koncovými uzly dostatečná. Implementace tohoto protokolu je tedy vyžadována minimálně po třetí (propojovací) vrstvu. Usnadnění přístupu z testovacího programu pomocí implementace třetí vrstvy Ethernetu však zvyšuje

nároky na řídicí obvod. Vícevrstvá implementace je nepraktická pro hradlové pole a vhodnější realizovat softwarově v procesoru.

Z důvodů zmíněných výše vyplývá jako nejvhodnější řešení použití obvodu SoC, který v jednom pouzdře obsahuje ARM procesor i programovatelné hradlové pole. Což je v souladu s požadavkem zadání práce na použití obvodu řady Zynq System-on-Chip. Zjednodušené blokové schéma obvodů řady Zynq-7000 SoC od výrobce Xilinx je na obrázku 3.1.



Obr. 3.1: Vnitřní blokové schéma obvodu SoC Zynq-7000 [10]

Obvod nabízí několik možností připojení k výstupním buňkám. Pro rozhraní řízená pouze procesorem je možné využít vytvořenou logiku jednoho z vybraných rozhraní (Ethernet, USB, UART, I2C, SPI a další). Tato rozhraní jsou připojena přes MIO (*multiplexed IO*) vývody, které lze nakonfigurovat jako připojené k procesoru i k programovatelné logice. Spojení mezi hradlovým polem a procesorem je zajištěno pomocí AXI sběrnice případně pomocí paměťových propojení. [12]

Obvody z rodiny Zynq-7000 lze připojit k Ethernetu za využití rozhraní GMII, RGMII nebo SGMII, která se liší v míře serializace dat a využívají pro jeden směr komunikace 12, 6 respektive 2 vodiče. Buzení kabelu na fyzické vrstvě je realizováno pomocí PHY čipu, jeho výběr a zapojení jsou popsány v kapitole 4.8. Volba konkrétního rozhraní pro komunikaci je dána datovou propustností vývodů integrovaného obvodu.

3.2 Připojení k rozhraním

Vzhledem k tomu, že spojení s jednotlivými koncovými rozhraními testovací jednotky má být paralelní a nezávislé pro každé jednotlivé rozhraní, je potřebné vytvořit základní jádra jednotlivých rozhraní v programovatelné logice, aby bylo zaručeno odbavení dat v reálném čase. Dodatečný hardware potřebný k zajištění souladu s normami popsané v kapitole 2 je k hradlovému poli připojen pomocí univerzálních vstupně-výstupních bloků. Žádné z rozhraní nevyžaduje z důvodu vysokého kmitočtu použití *SelectIO* bloků.

Pro výběr konkrétního integrovaného obvodu je nutné provést kalkulaci potřebného množství vstupů a výstupů připojených k hradlovému poli. Všechna rozhraní založená na LVDS lze budit diferenčně za použití dvou vývodů hradlového pole nebo nediferenčně za použití jednoho vývodu hradlového pole a externího budiče nebo přijímače diferenčního signálu. Pro rozhraní specifikovaná normami ECSS je z důvodu dodržení předepsaných parametrů nutné tyto externí integrované obvody použít a je počítáno s využitím pouze jednoho vývodu pro vysílání a jednoho vývodu pro přijímání. Detailní obvodové zapojení a výběr konkrétních integrovaných obvodů je řešeno v kapitole 4.

3.3 Vnitřní zapojení SoC

Spojením požadavků popsaných v kapitolách 1, 2 a 3 vzniká celkový koncept propojení testovacího systému. Řídicí počítač komunikuje pomocí rozhraní Ethernet s procesorem obsaženým v SoC Zynq-7000. Procesor zpracovává přijaté pakety a na základě jejich vyhodnocení je předává skrz propojovací matici samostatně běžícím jádrům koncových rozhraní. Tato jádra vytvořená v programovatelné logice jsou připojena na vývody integrovaného obvodu a za využití externího hardwaru (typicky posouvače úrovní a ochranné obvody) komunikují na cílových rozhraních s testovanou jednotkou.

3.3.1 Propojení procesoru a programovatelné logiky

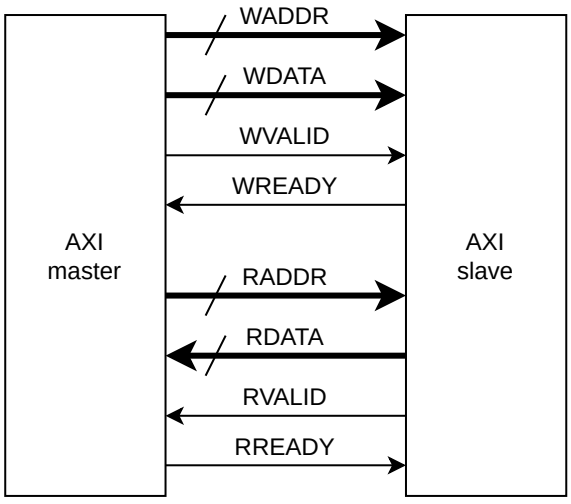
Předávání dat mezi procesorem a programovatelnou je v architektuře Zynq-7000 zajištěno pomocí dvou 32bitových AXI sběrnic a dvou 64bitových AXI sběrnic. AXI je rozhraní navržené pro propojení jednotlivých komponent v modulárních návrzích.

Při spojení dvou bodů má jeden vždy roli *master* a druhý roli *slave*. Architektura SoC obsahuje 32bitová AXI rozhraní s maximální hodinovou frekvencí 150 MHz a 64bitová rozhraní se stejnou frekvencí. Rozhraní o šířce 32 bitů jsou schopny maximální rychlosti 600 MB/s pro zápis a 600 MB/s pro čtení. 64bitová rozhraní mají dvojnásobnou rychlost. [12]

Základními signály AXI rozhraní jsou dvě datové sběrnice pro čtení a zápis. Čtením se rozumí přenos dat ve směru od modulu *slave* k modulu *master*. Adresování je zajištěno pomocí dvou adres, jednou pro čtení a jednou pro zápis. Tyto adresy jsou vždy řízeny zařízením *master*. [11]

Dvě zařízení propojená AXI rozhraním si předávají informaci o připravenosti na čtení a zápis pomocí diskrétních signálů. Zařízení *slave* signalizuje připravenost na zápis signálem *WREADY* a připravenost na čtení signálem *RREADY*. Podobně je přenášena informace o platné adrese signály *AWREADY* a *ARREADY*. [11]

Platná data jsou doprovázena v obou směrech přenosu signály *WVALID* a *RVALID*. Obdobně je tomu s adresami a jejich doprovodnými signály *AWVALID* a *ARVALID*. Součástí rozhraní jsou také signály pro vícenásobné čtení a zápis, pro potvrzení přijetí dat a další signály pro rozšířenou funkčnost AXI rozhraní. Na obrázku 3.2 je zobrazeno základní AXI spojení zařízení *master* a *slave*. [11]

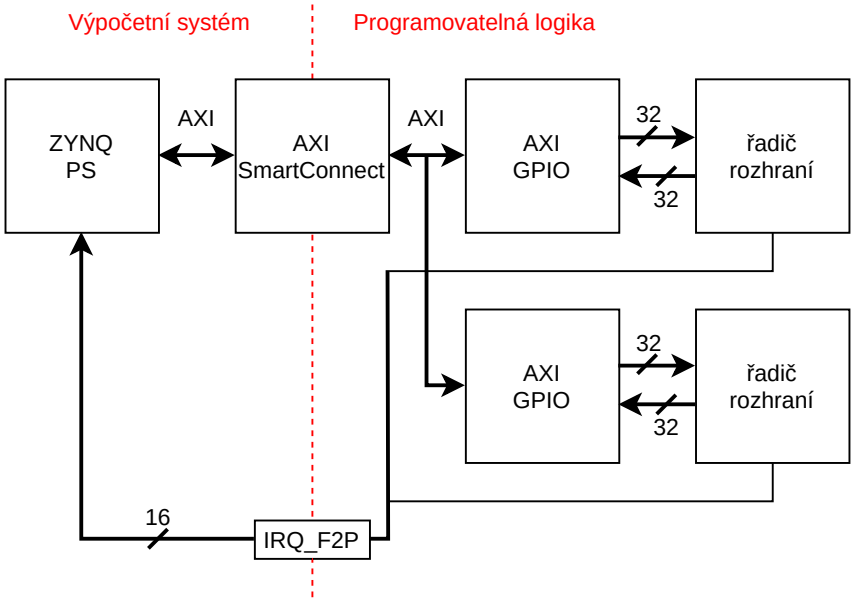


Obr. 3.2: Spojení pomocí AXI rozhraní

V obvodech rodiny Zynq-7000 jsou AXI rozhraní mezi výpočetní jednotkou a programovatelnou logikou spojována přes dva 32bitové porty M_AXI_GP a S_AXI_GP, z nichž v prvním je procesor zařízením *master* a v druhém zařízením *slave*. Dva 64bitové porty S_AXI_ACP a S_AXI_HP jsou konfigurovány jako *slave* pro procesor. Port S_AXI_HP je určen k přímému přístupu do DRAM paměti nebo do OCM (on-chip-memory) paměti. [12]

Pro testovací platformu je vhodné využít adresy AXI rozhraní jako identifikátor jednotlivých koncových modulů připojených v programovatelné logice. Pro každý z těchto modulů lze využít pouze jednu přidělenou adresu, jelikož procesy čtení i zápisu do těchto modulů probíhají v čase sekvenčně podle pořadí příchozích a odchozích dat. Takový přístup vede k vytvoření jednoho AXI spojení mezi procesorem a programovatelnou logikou za využití adresy jako označení pro každé rozhraní. Pro zamezení kolizí na straně programovatelné logiky řídí toto AXI spojení procesor, využít lze proto pouze port M_AXI_GP, na kterém je procesor zařízením *master*.

Pro převod komunikace mezi AXI rozhraním a řadičem koncového rozhraní jsou použity AXI GPIO bloky z knihovny distribuované jako součásti vývojového prostředí Vivado. Rozvětvení AXI rozhraní mezi jednotlivé řadiče je zajištěno blokem AXI SmartConnect. Samotný port řadiče se pak skládá z 32bitového vstupního vektoru a 32bitového výstupního vektoru. Struktura dat v těchto vektorech je popsána v kapitole 5.1. Vnitřní zapojení SoC je na obrázku 3.3.

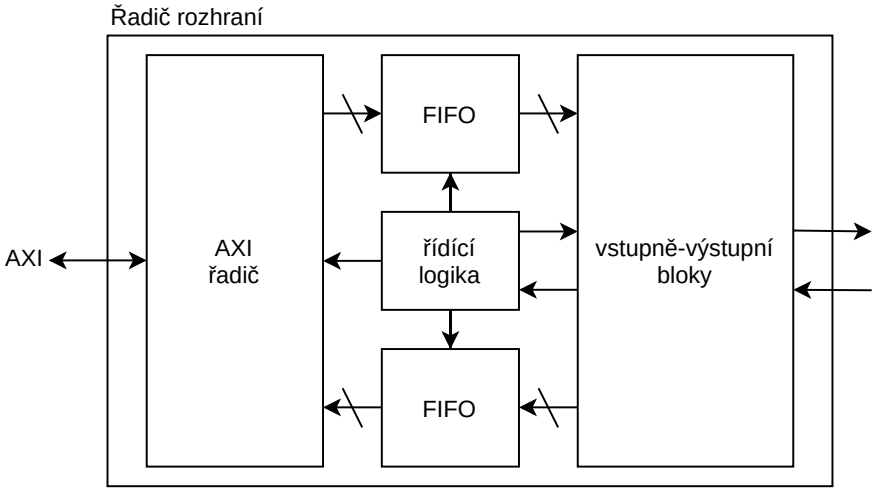


Obr. 3.3: Vnitřní zapojení SoC

Vyvolání čtení ze strany řadiče rozhraní, které je v roli AXI *slave* je řízeno pomocí portu IRQ_F2P. Ten obsahuje 16 signálů přerušení se stanovenou prioritou. Připojením řadičů tak lze určit priorita čtení z jednotlivých rozhraní podle jejich důležitosti v testovaném systému. Procesor v obsluze přerušení reaguje na tyto signály vyvoláním čtení z AXI adresy přidělené danému rozhraní.

3.3.2 Vyrovnávací paměť

V případě přijetí dat z více rozhraní ve stejném okamžiku je nutné data ukládat do vyrovnávací paměti do doby než je procesorem umožněno je odeslat do řídicího počítače. Ve vyrovnávací paměti tvořené FIFO pamětí se tvoří fronta neodbavených slov. Principiální schéma řízení toku dat uvnitř jádra rozhraní je na obrázku 3.4.



Obr. 3.4: Řízení toku dat uvnitř jádra rozhraní

3.4 Výběr cílového obvodu

V tabulce 3.1 je srovnání obvodů rodiny Zynq-7000. Nároky ze strany softwaru splňují všechny obvody, ale kvůli teoretické rozšířitelnosti funkcí obvodu je vhodné použití dvoujádrového obvodu. Dvou výpočetních jader lze využít k současnému zpracovávání více vláken programu v případě velkého datového toku z koncových rozhraní. Z hlediska programovatelné logiky je nejdůležitějším parametrem velikost blokové paměti kvůli vyrovnávacím FIFO pamětem v řadičích. Samotná logika řadičů nevyžaduje velké prostředky.

Tab. 3.1: Srovnání SoC rodiny Xilinx Zynq-7000 [10]

Označení	f_{max}	**Pouzdra**	**Log. buňky**	**Jádra**	**BRAM**
Z-7007S	766 MHz	CLG225 / CLG400	23 000	1	1,8 Mb
Z-7012S	766 MHz	CLG485	55 000	1	2,5 Mb
Z-7014S	766 MHz	CLG400 / CLG484	65 000	1	3,8 Mb
Z-7010	866 MHz	CLG225 / CLG400	28 000	2	2,1 Mb
Z-7015	866 MHz	CLG485	74 000	2	3,3 Mb
Z-7020	866 MHz	CLG400 / CLG484	85 000	2	4,9 Mb
Z-7030	1 GHz	SBG485 / FFG676	125 000	2	9,3 Mb
Z-7035	1 GHz	FBG676 / FFG900	275 000	2	17,6 Mb
Z-7045	1 GHz	FBG676 / FFG900	350 000	2	19,2 Mb
Z-7100	800 MHz	FFG900 / FFG1156	444 000	2	26,5 Mb

Pro účely této práce byl vybrán obvod ze středu spektra Z-7020. Důvodem výběru bylo osazení obvodu na vývojové desce StemLAB Red Pitaya 122-16, která byla použita k testování komunikace testovací platformy. Pro konkrétní projekt je při hardwarové realizaci možné tento obvod nahradit jiným.

4 HARDWAROVÉ ŘEŠENÍ

Tato kapitola se zabývá návrhem hardwarové části testovací platformy. Jejím obsahem je jednak digitální návrh programovatelné logiky uvnitř hradlového pole, který zajišťuje funkčnost nejnižších vrstev rozhraní a jednak obvodové zapojení mimo SoC, které zajišťuje spojení řídicího obvodu s koncovými body při dodržení kompatibility s danými normami shrnutými v kapitole 2. Kromě hardwarového návrhu souvisejícího s jednotlivými rozhraními obsahuje tato kapitola řešení napájení a konfigurace obvodu SoC, které je nutným předpokladem k jeho funkčnosti v rámci systému.

Ke každému rozhraní je v této kapitole navržen řadič implementovatelný v programovatelné logice obvodu SoC. Základní funkcí těchto řadičů je překlad protokolů mezi sjednocenou komunikací po sběrnici řízenou procesorem a specifickými protokoly používanými jednotlivými fyzickými vrstvami rozhraní. Tato základní funkce je v řadiči vykonávána dvěma bloky, dekódérem příkazů a blokem řídícím tok dat. Dekódér příkazů reaguje na hodnotu pole D/C specifikovaného jako součást protokolu v kapitole 5.1, který rozlišuje datová slova od příkazů. Tento blok je povinnou součástí každého řadiče, aby byla zaručena kompatibilita se softwarovou knihovnou popsanou v kapitole 5.2. Řízení toku dat je nutné implementovat pouze u těch rozhraní, po kterých probíhá datová komunikace. Řadiče bez tohoto bloku nejsou určené k předávání dat na stranu koncového rozhraní.

Zapojení obvodů souvisejících s cílovými rozhraními je v této práci řešeno na úrovni schematického návrhu. Tato kapitola konkrétně obsahuje k rozhraním navrženým bez budících integrovaných obvodů dílčí simulace zapojení mezi obvodem SoC a koncovým bodem připojení testované jednotky, jejichž úkolem je splnění požadavků kladenými normami. Pro rozhraní buzená integrovanými obvody je kompatibilita s normami zaručena parametry udávanými výrobcem.

4.1 SpaceWire

Prvním rozhraním rozebíraným po hardwarové stránce je SpaceWire. Z cílových rozhraní má nejsložitější implementaci v programovatelné logice z důvodu složitosti jeho řadiče související s vrstvovou architekturou. Mimo programovatelnou logiku pak hardwarová stránka rozhraní vychází z hojně rozšířeného LVDS standardu. Komunikace po tomto rozhraní je plně duplexní a není tudíž nutné zajišťovat řízení toku dat ve smyslu otáčení směru komunikace po sběrnici.

4.1.1 Digitální návrh pro FPGA

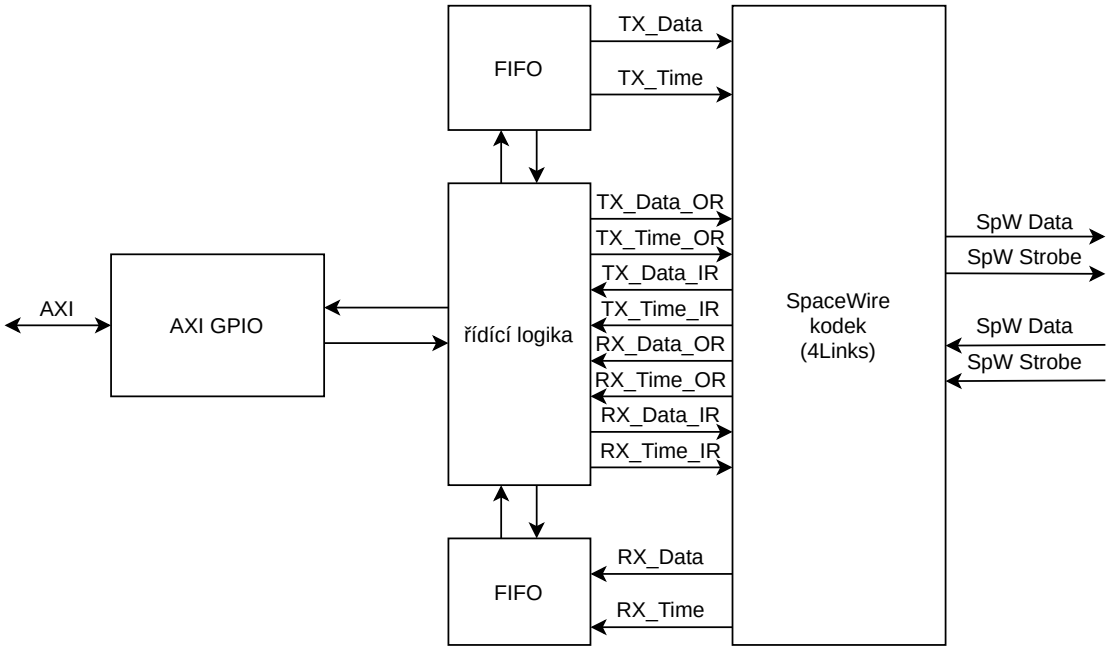
Jak bylo specifikováno v kapitole 2.4, SpaceWire je vrstvový protokol. K funkčnosti v systému testovací platformy je nutná implementace síťové, spojovací a fyzické vrstvy. Fyzická vrstva je řešena mimo programovatelnou logiku, v rámci FPGA je realizována vrstva síťová a spojovací.

K implementaci SpaceWire protokolu bylo použito *open-source* řešení vyvinuté společností 4Links. [23] Tento kodek se stará o dekódování *data-strobe* modulace, zajišťuje inicializaci rozhraní po resetu a signalizuje chybové příznaky na rozhraní.

Na straně rozhraní je jádro připojeno čtyřmi diferenčními páry - *data* a *strobe* pro odesílání a přijímání dat. Vstupy a výstupy z programovatelné logiky jsou řešeny jako DFC (*data flow channel*), tedy jeden paralelní datový signál a dva řídicí signály *Data_OR* a *Data_IR*. Tyto signály značí připravenost rozhraní přijímat data (*input ready*) a odesílat data (*output ready*). Obě strany mohou pozastavit výměnu informací logickou nulou na jednom ze signálů. [23]

Kromě datového spojení je z jádra vyvedeno rozhraní pro časové rámce protokolu SpaceWire (*time service*) a několik řídicích signálů pro signalizaci řídicích znaků a chyb, konkrétně *Error_inject* a *Error_select* pro injektování chyby do rozhraní, *Force_timeout_error* pro vnucení vypršení časovače, *Connected* jako příznak funkčního SpaceWire spojení a signály *Rx_ESC_ESC*, *RX_ESC_EOP*, *RX_ESC_EEP*, *RX_Parity_error* pro přijetí řídicích znaků. [23]

Spojení mezi SpaceWire kodekem třetí strany a komunikační sběrnicí řídicího obvodu je zajištěno pomocí řídicí logiky SpaceWire řadiče. Jeho úloha je předávat data kodeku přijatá z ARM procesoru, ukládat data z rozhraní do vyrovnávací paměti a vystavovat je na sběrnici, když o ně procesor požádá přepnutím signálu *RREADY* do logické jedničky. Řídicí logika zároveň obsahuje dekodér příkazů. Tyto



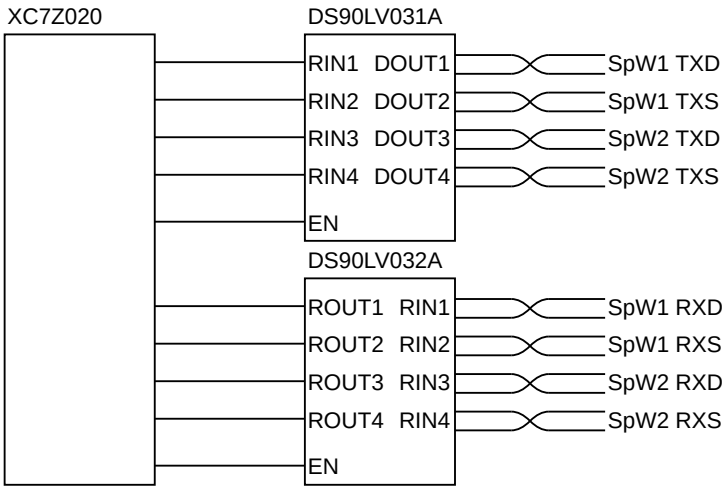
Obr. 4.1: Blokové schéma SpaceWire řadiče

příkazy jsou součástí komunikačního protokolu mezi řídicím počítačem a testovacím zařízením. Příkazy jsou specifické pro každé konkrétní rozhraní a jejich struktura je řešena v kapitole 5. Blokové schéma SpaceWire řadiče je na obrázku 4.1.

4.1.2 Schéma připojení k FPGA

Jak bylo rozebráno v kapitole 2.4, SpaceWire je na signálové úrovni fyzické vrstvy kompatibilní s LVDS normou TIA-664. Vývody řídicího obvodu Zynq-7000 je možné nastavit jako LVDS vstupy a výstupy, ale přímé připojení bez oddělovacích obvodů není vhodné z důvodu výkonového zatěžování SoC obvodu a ochrany řídicího obvodu před šířením elektrických chyb z testované jednotky.

Pro oddělení obvodu SoC a testované jednotky jsou použity LVDS budiče a přijímače. Z důvodů úspory vývodů FPGA je vhodné vybrat takové obvody, jejichž rozhraní na straně FPGA není diferenční. Při výběru je také nutné vzít potaz kompatibilitu napěťových úrovní vstupně-výstupních standardů FPGA a těchto integrovaných obvodů. Jako LVDS vysílač byl vybrán obvod DS90LV031A, jako přijímač obvod DS90LV032A. Tyto obvody obsahují v pouzdře 4 vysílače respektive přijímače LVDS. Každé SpaceWire spojení vyžaduje 2 LVDS páry pro *data* a *strobe*, pomocí jednoho páru integrovaných obvodů tak lze vytvořit 2 SpaceWire rozhraní. Schéma zapojení SpaceWire rozhraní je na obrázku 4.2. [14][15]



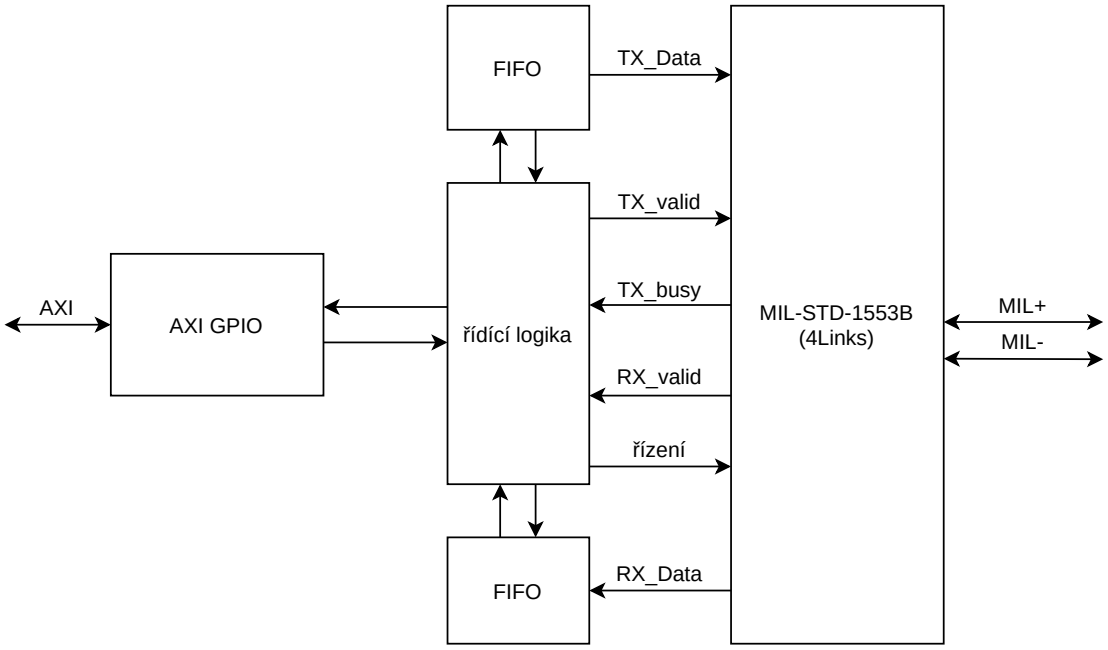
Obr. 4.2: Zapojení rozhraní SpaceWire

4.2 MIL-STD-1553B

V testovacím systému může testovací zařízení připojené ke sběrnici MIL-STD-1553B plnit jednu ze tří rolí. V módu BC (*bus controller*) je zařízení zodpovědné za řízení celé sběrnice, rozesílá všem uzlům časové synchronizační značky a inicializuje komunikaci služeb *get data*, *send data* a *data block transfer*. V módu RT (*remote terminal*) se zařízení řídí pokyny od BC, ale je schopné odpovídat na zprávy a na vyzvání odesílat bloky dat. V módu BM (*bus monitor*) zařízení pouze čte data ze sběrnice. Pro testovací systém je nejvhodnějším řešením rozhraní přepínatelné mezi těmito třemi módy.

4.2.1 Digitální návrh pro FPGA

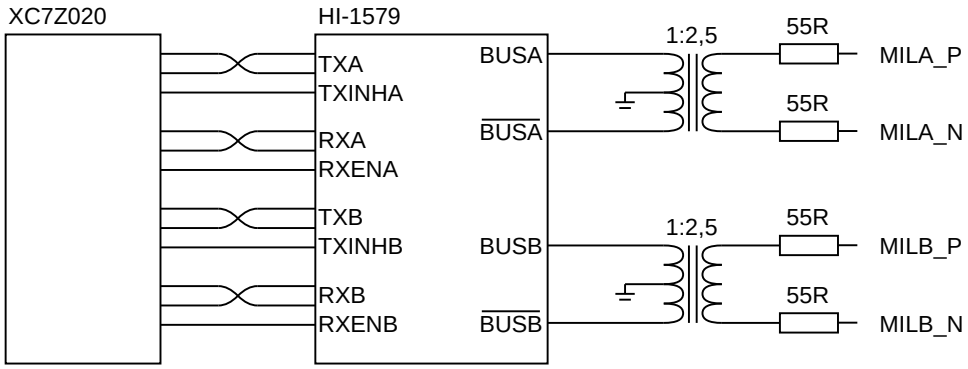
Kromě povinných bloků nutných k připojení řadiče MIL-STD-1553B musí řadič obsahovat logiku zabezpečující funkce řízení sběrnice a časové synchronizace. Tato funkčnost je využita pouze při provozování koncového bodu v režimu BC. Přepínání mezi jednotlivými módy je řešeno příkazy z řídicího počítače, jejich struktura je popsána v kapitole 5.2. Návrh řídicího jádra rozhraní MIL-STD-1553B není součástí této práce, počítá se s využitím některého z řešení dostupných na trhu. Blokové schéma řadiče je na obrázku 4.3.



Obr. 4.3: Řadič rozhraní MIL-STD-1553B

4.2.2 Schéma připojení k FPGA

Požadavky fyzické vrstvy rozhraní MIL 1553B byly rozebrány v kapitole 2.5. Sběrnice je tvořena jedním diferenčním párem odděleným od koncového uzlu vazebním transformátorem a ochrannými rezistory. Pro buzení rozhraní byl vyrbán integrovaný obvod Holt integrated circuits HI-1579. Jedná se o základní řadič rozhraní MIL-STD-1553B obsahující dva vstupně-výstupní kanály s povolovací logikou. Schéma zapojení rozhraní MIL-STD-1553B je na obrázku 4.4. [16]



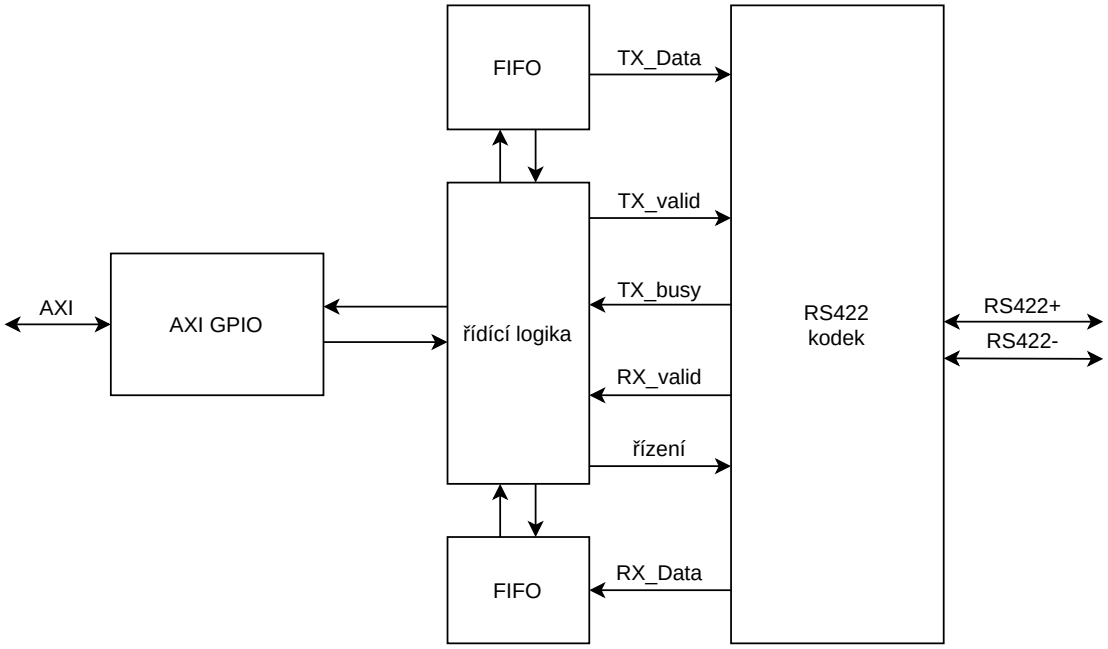
Obr. 4.4: Zapojení rozhraní MIL-STD-1553B

4.3 RS-422

Požadavky na implementaci rozhraní RS-422 byly rozebrány v kapitole 2.3. Vzhledem k řadiči se jedná o jednosměrnou komunikaci, připojení je tak realizováno dvěma oddělenými komunikačními kanály, které dohromady tvoří plně duplexní komunikaci.

4.3.1 Digitální návrh pro FPGA

Součástí RS-422 nejsou podle specifikace žádné stavové signály ani příznaky, které je možné v řídicí logice vyhodnocovat jako součást telemetrie. Řadič tohoto rozhraní je proto řešen pouze jako překladač protokolů. Tato funkce je stejně jako u ostatních řadičů řešena dekóderem příkazů a blokem řídícím tok dat. Z důvodu univerzálního použití je v řadiči implementována změna datové rychlosti rozhraní. Blokové schéma řadiče je na obrázku 4.5.



Obr. 4.5: Řadič rozhraní RS-422

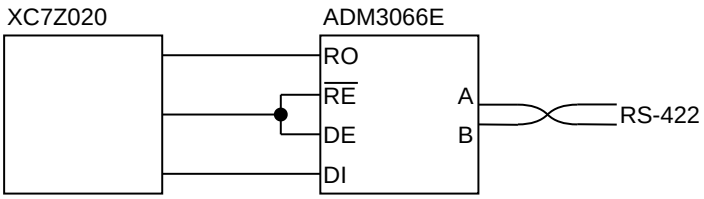
4.3.2 Schéma připojení k FPGA

Napěťové úrovně na fyzické vrstvě rozhraní RS-422 nejsou kompatibilní se vstupně-výstupními standardy programovatelné logiky obvodu Zynq-7000.

Pro vzájemné propojení je vyžadováno použití vhodného integrovaného obvodu. Rozhraní RS-422 je na přenosovém médiu shodné s rozhraním RS-485, integrované obvody jsou proto často kompatibilní s oběma těmito rozhraními. Rozdíl je v časovém multiplexu komunikace souvisejícím s obousměrným přenosem po jednom diferenčním páru u rozhraní RS-485. V případě použití takového obvodu by vysílací nebo přijímací polovina obvodu zůstala nevyužita.

Pro testovací systém byla vybrána varianta, ve které je použit integrovaný obvod plnící funkci vysílače i přijímače. Celý řadič rozhraní je možné provozovat pouze v jednom z těchto režimů. Výhodou tohoto přístupu je možnost softwarové modifikace systému vedoucí k vytvoření rozhraní RS-485 v případě potřeby. Jako budič rozhraní byl vybrán integrovaný obvod ADM3066E. Tento obvod zajišťuje poloviční duplexní komunikaci, umožňuje použít nižší napájecí napětí na straně připojení k SoC a dosahuje rychlosti až 50 Mbit/s. [17]

Kromě vstupních a výstupních datových vývodů obsahuje integrovaný obvod povolovací vývody čtení a vysílání. Vstup RE (*read enable*) je v tomto případě invertovaný za účelem jeho elektrického propojení se signálem DE (*data enable*) a možností obracení směru toku dat pouze jedním řídicím signálem. Zapojení rozhraní RS-422 je na obrázku 4.6.



Obr. 4.6: Schéma zapojení rozhraní RS-422

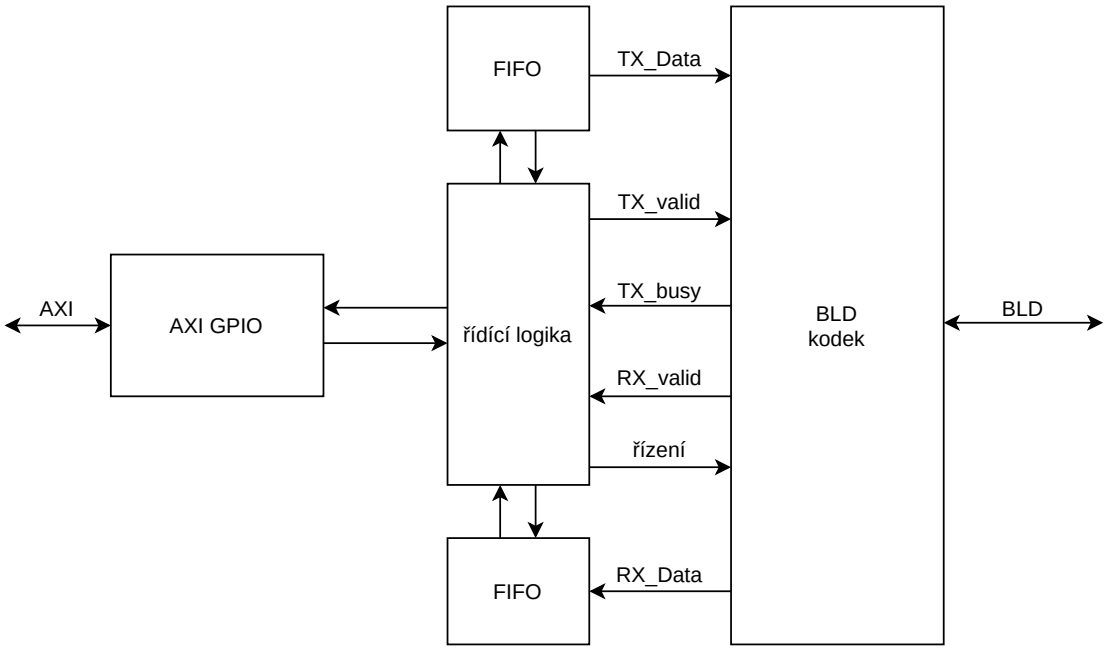
4.4 SBDL

Rozhraní SBDL je pouhé rozšíření RS-422 o specifikaci tolerance při poruše v obvodu. S tímto stavem není nutné u testovacího zařízení počítat. Celé zařízení je napájeno maximálním napětím $5\,V$, chybové napětí vysílače proto nemůže překročit povolenou mez $7\,V$. V opačném směru je řídicí obvod Zynq-7000 oddělen od koncového rozhraní přijímačem RS-422. Z toho důvodu je řadič i obvodové zapojení rozhraní SBDL shodné s rozhraním RS-422.

4.5 BLD

4.5.1 Digitální návrh pro FPGA

Z hlediska řadiče je u rozhraní BLD nutné zajistit pouze správné nastavení směru toku dat vstupů nebo výstupů. Ostatní bloky řadiče jsou implementovány pouze z důvodu kompatibility se sběrnicovým řešením přenosu dat. V případě nastavení rozhraní jako výstupu jsou data vystavována nastavenou rychlostí, v případě nastavení jako vstupu jsou data ukládána do vyrovnávací paměti a odeslána procesoru na základě povolovacího signálu RREADY. Blokové schéma řadiče je na obrázku 4.7.

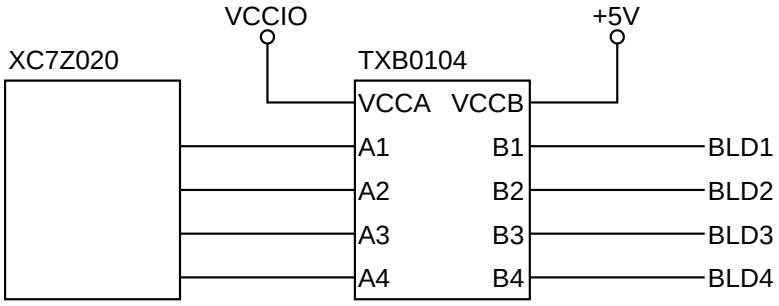


Obr. 4.7: Řadič rozhraní BLD

4.5.2 Schéma připojení k FPGA

Maximální napájecí napětí vstupně-výstupních bloků programovatelné logiky řídicího obvodu je $3,3\,V$. Pro spojení obvodu s pětivoltovou logikou rohraní BLD je zapotřebí převodníku napěťových úrovní. Jednosměrný převod na nižší napětí lze zajistit odporovým děličem, převod na vyšší napěťovou úroveň lze zajistit pomocí MOSFET tranzistoru a *pullup* rezistorů. Pro lepší oddělení SoC a koncového rozhraní byl na tuto úlohu vybrán integrovaný obvod Texas instuments TXB0104. Tento obvod zajišťuje obousměrný převod napěťových úrovní digitálního signálu mezi

stranou A s napětím v rozsahu $1,2 - 3,6\,V$ a stranou B s napětím v rozsahu $1,65 - 5,5\,V$. Při převodu z $2,5$ nebo $3,3\,V$ na $5\,V$ je schopen propustit datový signál až o rychlosti $100\,Mbit/s$. Zapojení převodníku a rozhraní BLD je na obrázku 4.8. [18]



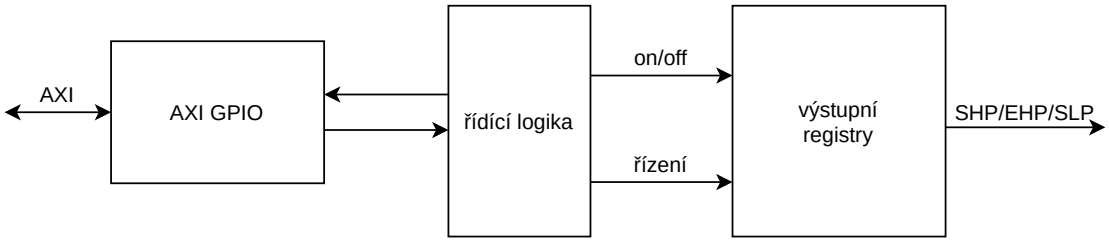
Obr. 4.8: Schéma zapojení rozhraní BLD

4.6 Spínání výkonových zátěží

Spínání zátěží pomocí rozhraní SHP, ELP a SLP nevyžaduje v řadiči složitou logiku. Spínání jednoho rozhraní na výstupu FPGA je řešeno dvoustavovou logikou, kde logická nula značí stav vypnuto a logická jednička stav zapnuto. Pro ušetření logických buněk je do jednoho individuálně adresovatelného řadiče sdruženo 8 výstupů, jejichž jednotlivé hodnoty jsou nastaveny pomocí příkazů. Rozhraní pro spínání nemají datovou část a není tedy nutná implementace řízení toku dat.

4.6.1 Digitální návrh pro FPGA

Řadič všech výkonových rozhraní je shodný. Kromě dekódéru příkazů obsahuje jen výstupní registr reagující na povolující signál z dekódéru. Stavové signály s telemetrií jsou v případě pouze výstupního rozhraní nepotřebné. Pro efektivnější využití AXI sběrnice je do jednoho řadiče sdruženo 8, 16, případně 32 výstupů řídítelných datovým slovem po jednotlivých bitech. Řadič pro spínání výkonových zátěží je na obrázku 4.9.

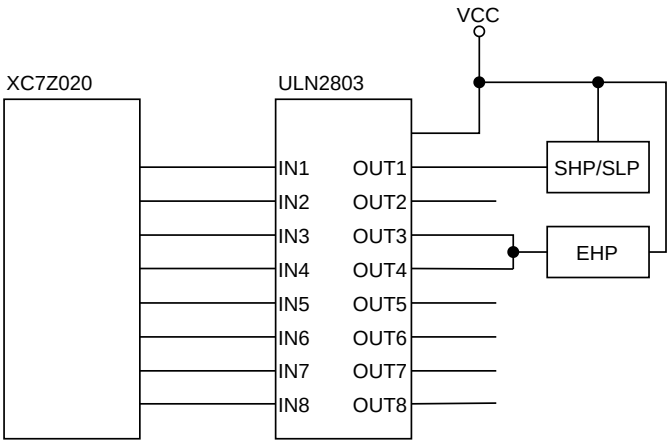


Obr. 4.9: Řadič pro spínání výkonových zátěží

4.6.2 Schéma připojení k FPGA

Požadavky na parametry výkonových rozhraní byly popsány v kapitole 2.6. Proudy ani napětí požadované normou není možné spínat přímo z vývodů FPGA, jejichž maximální napětí je $3,3\,V$ a proud $24\,mA$. [22]

Pro spínání výkonových zátěží byl vybrán integrovaný obvod Texas Instruments ULN2803A. Ten obsahuje osm Darlingtonových tranzistorových párů schopných spínat maximální proud celkem $500\,mA$ při proudu jedním výstupem až $300\,mA$. Vstupní napětí ovládacích vývodů je $3,3\,V$ nebo $5\,V$. Proudový odběr z vývodu FPGA při sepnutém stavu je maximálně $1,35\,mA$. [19]



Obr. 4.10: Schéma připojení výkonových zátěží

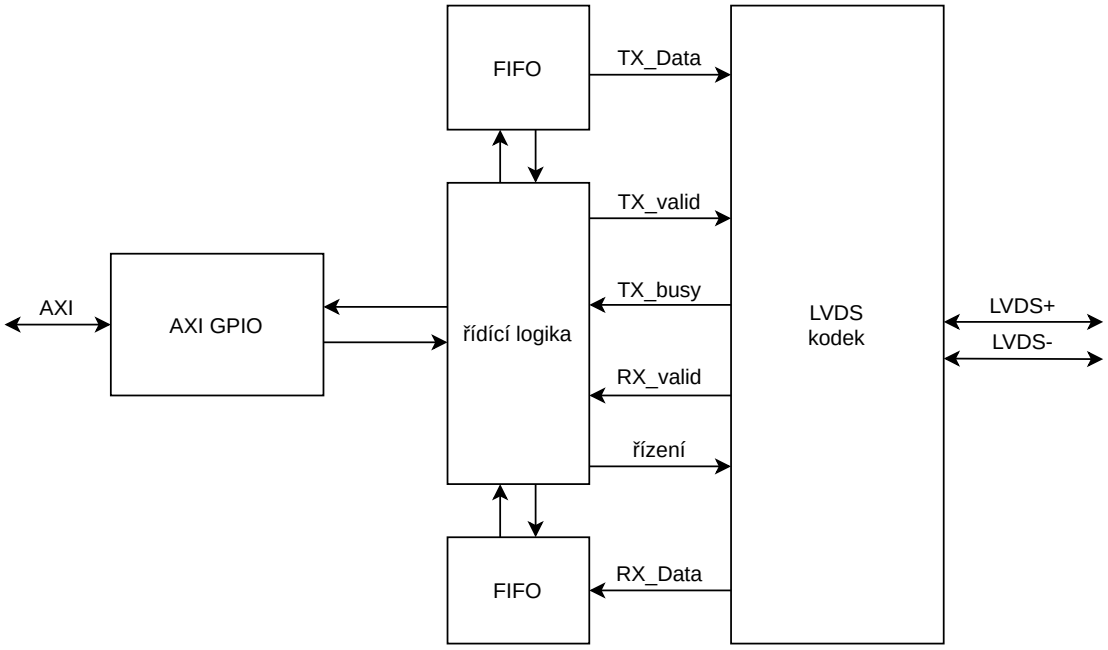
Integrovaný obvod je jedním vývodem připojen k napájecímu napětí spínané zátěže. Osm výstupních kanálů spínaných jedním obvodem musí mít společné napájecí napětí. Rozhraní SHP a SLP je možné spínat jedním kanálem, jehož proud nepřesahuje maximálně povolených $300\,mA$. Pro spínání rozhraní EHP je nutné spojit dva kanály na vstupu i na výstupu pro dosažení maximálního proudu až $500\,mA$. Schéma zapojení je na obrázku 4.10.

4.7 LVDS

V rámci univerzálního použití testovacího zařízení jsou jeho součástí i standardní rozhraní bez bližší specifikace zamýšleného použití. Jedním z nich je plně duplexní LVDS komunikace. Principy LVDS signalizace byly popsány v kapitole 2.2. Jelikož je rozhraní navrženo pro předem neurčený komunikační protokol, není součástí jeho implementace vyhodnocování navázání spojení nebo chybových stavů. Programovatelná logika má v tomto případě za úkol pouze obousměrnou komunikaci s AXI sběrnicí a serializaci nebo paralelizaci dat.

4.7.1 Digitální návrh pro FPGA

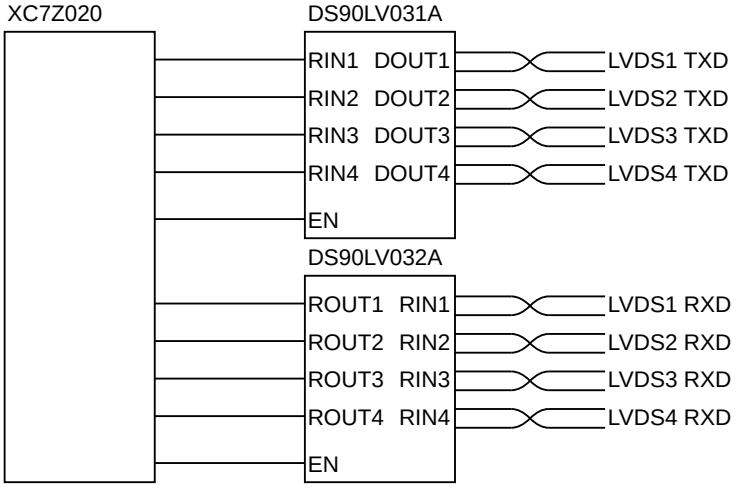
LVDS řadič v programovatelné logice obsahuje jako všechna rozhraní povinné bloky dekódéru příkazů a řízení toku dat. Z vyrovnávací paměti jsou data po bajtech předávána bloku zajišťující serializaci a následně vysílána po diferenčním páru na výstup FPGA. V opačném směru jsou data obdobně deserializována a uložena ve vyrovnávací paměti. Datová rychlost rozhraní lze měnit pomocí frekvence pulsů CE signálu a může tedy nabývat celých podílů frekvence systémového hodinového signálu. Přesnější řízení kmitočtu není v tomto případě žádoucí, protože vyžaduje použití DCM bloků hradlového pole a mezidoménové přechody.



Obr. 4.11: Řadič LVDS rozhraní

4.7.2 Schéma připojení k FPGA

Připojení LVDS rozhraní k FPGA je obdobné jako u rozhraní SpaceWire, které je jeho vrstvovou nadstavbou. Pro spojení na krátkou vzdálenost, u kterého jsou zaručeny druhou komunikační stranou bezpečné provozní podmínky (zejména napěťové úrovně), je možné připojení bez integrovaných obvodů pro buzení a přijímání LVDS komunikace. V takovém případě jsou jako ochrana proti nadměrnému zatěžování výstupních bloků použity sériově zapojené rezistory. V případě nutnosti ochrany řídicího obvodu je možnost využít stejný způsob buzení jako v případě SpaceWire. Pomocí jednoho páru integrovaných obvodů lze vytvořit 4 vstupní a 4 výstupní rozhraní. Schéma zapojení FPGA a LVDS rozhraní je na obrázku 4.12.



Obr. 4.12: Schéma zapojení rozhraní LVDS

4.8 Ethernet

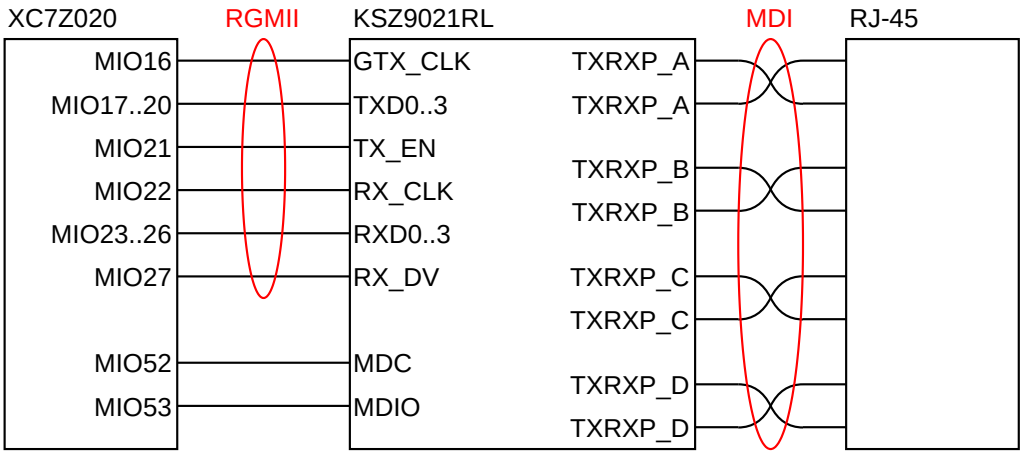
Funkce protokolu Ethernet v testovacím systému je přenášet informace mezi řídicím počítačem a procesorem v obvodu SoC. V místě tohoto spojení vzniká kritický bod související s rychlostí přenosu. Potřebná přenosová rychlost je určena rychlostmi jednotlivých rozhraní, časové střídě jejich komunikace a jejich počtem. Z důvodu použití rozhraní SpaceWire, jehož rychlost může podle normy dosahovat až 200 Mbit/s je vhodné provozovat Ethernet na rychlosti 1 Gbit/s.

Jelikož rozhraní Ethernetu není v navrhovaném testovacím systému koncovým rozhraním připojitelným k testované jednotce, je k obvodu SoC připojeno pomocí

MIO vstupů a výstupů a propojovací soustavou uvnitř čipu přivedeno k jádru ARM procesoru. Z toho vyplývá, že Ethernet nevyužívá žádnou programovatelnou logiku, pouze připravené struktury obvodu Zynq-7000.

4.8.1 Schéma připojení k SoC

Obvody řady Zynq-7000 jsou kompatibiltní s několika konkrétními rozhraními používanými k přenosu protokolu Ethernet. Tato rozhraní byla zmíněna v kapitole 3.1. O převod mezi těmito rozhraními a rozhraním MDI (*media dependent interface*), které zajišťuje vlastní přenos informace mezi dvěma zařízeními, se stará PHY čip. Tento integrovaný obvod lze zpravidla přímo spojit s konektorem RJ-45 obsahujícím oddělující magnetické obvody a jeho řídicí registry jsou přístupné přes sériové rozhraní *PHY management interface*.



Obr. 4.13: Připojení SoC k rozhraní Ethernetu pomocí PHY čipu

Pro testovací systém byl na základě předchozí zkušenosti vybrán integrovaný obvod KSZ9021RL, který využívá rozhraní RGMII verze 1.3, umožňuje automatické nastavení komunikační rychlosti mezi dvěma spojenými uzly sítě a je možné jej napájet napětím $2,5\,V$ nebo $3,3\,V$, což je nutná podmínka pro využití MIO vývodů obvodu Zynq-7000. Zapojení PHY čipu a SoC je na obrázku 4.13. [20]

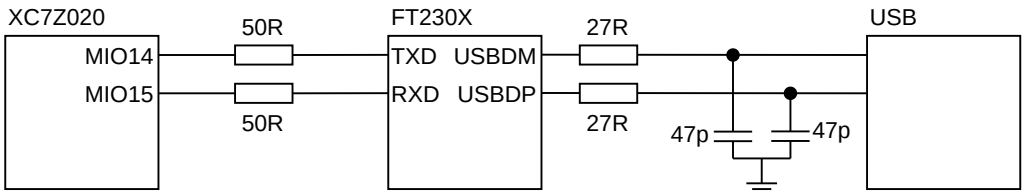
4.9 UART

Pro záložní komunikaci mezi řídicím počítačem je testovací zařízení připojitelné přes virtuální sériový port. Tento komunikační kanál je jednodušší alternativou Ethernetu pro případ ladění testovacího procesu. UART je jedno z rozhraní, pro které lze v integrovaném obvodu Zynq-7000 využít MIO vývody a řídit ho přímo z programu běžícího v procesoru. Připojení sériového portu dává procesoru možnost odesílat telemetrická data nebo jinak komunikovat s řídicím počítačem v případě selhání nebo nedostupnosti komunikace přes Ethernet.

Druhým potenciálním využitím sériového protokolu UART je komunikace s dalšími zařízeními v celém testovacím systému. UART lze využít díky své jednoduchosti. V tomto případě je vhodné, aby byla tomuto rozhraní přidělena adresa a bylo ho možné řídit pomocí hlavního komunikačního protokolu. Z toho důvodu je jeho řadič navržen pro programovatelnou logiku.

4.9.1 Komunikace s počítačem

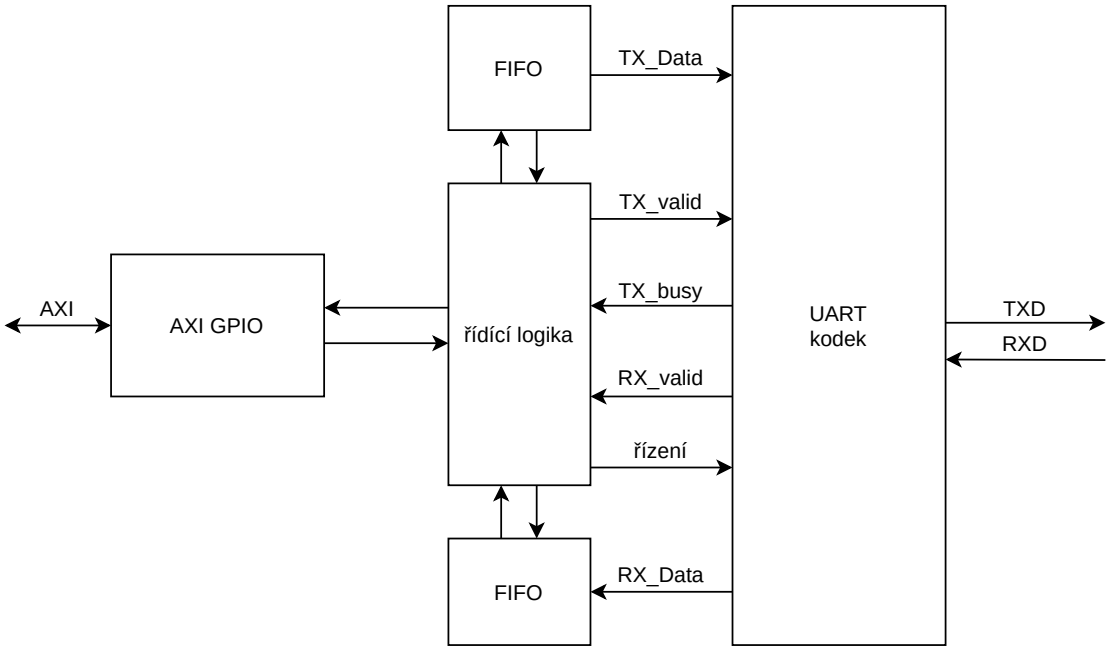
Sériový port lze k počítači připojit za použití USB sběrnice a převodníku USB na UART. Jednoduchým příkladem takového převodníku je obvod FT230X, který disponuje rozhraním USB 2.0 a plně duplexní sériovou komunikací na straně SoC. Sériová komunikace je provozována na napěťové úrovni $5\,V$ nebo $3,3\,V$. V rámci obvodu SoC je převodník připojen na MIO vývody 14 a 15. Schéma zapojení pro virtuální sériový port je na obrázku 4.14. [21]



Obr. 4.14: Schéma zapojení virtuálního sériového portu

4.9.2 Komunikace s jinými koncovými body

Pro zajištění kompatibility je nutné připojit řadič UART ke komunikační sběrnici uvnitř SoC. Stejně jako u ostatních rozhraní je proto v řadiči sériového portu dekódér příkazů a blok řídicí tok dat. Bitová frekvence a s ní související datový tok je nastavován specifikovaným příkazem. Schéma UART řadiče je na obrázku 4.15.



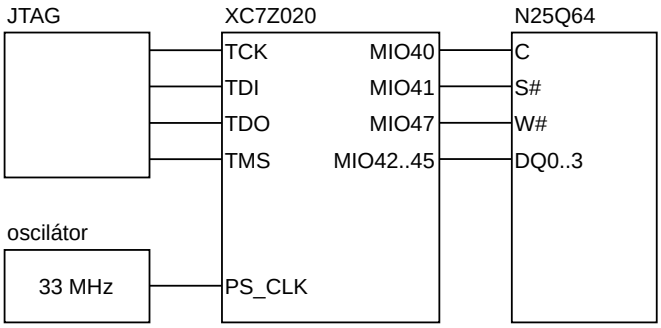
Obr. 4.15: Blokové schéma UART řadiče

4.10 Konfigurační obvody

Pro funkci řídicího obvodu SoC Zynq-7000 je vyžadováno připojení konfiguračních obvodů nutných k jeho inicializaci a nahrání digitálního návrhu programovatelné logiky. Technologické zapojení programovatelné logiky je z návrhového prostředí ukládáno do souboru *bitstreamu*, který obsahuje logické úrovně všech paměťových buněk řídicích propojovací matice, konfigurovatelné logické bloky a vstupně-výstupní bloky hradlového pole.

Paměťové buňky hradlového pole jsou volatilní a je nutné je inicializovat při každém připojení napájení nebo resetování obvodu. Bistream je možné nahrát pomocí rozhraní JTAG nebo ho uložit v externí nevolatilní paměti. Délka bitstreamu pro obvod xc7z020 je $32\,Mbit$. Doporučení výrobce udává minimální velikost konfigurační paměti $64\,Mbit$. Konfigurační rozhraní SoC využívá pro zápis a čtení QSPI. Vzhledem k podpoře vývojovými nástroji Xilinx byla vybrána flash paměť Micron N25Q64. [12]

Pro rekonfiguraci programovatelné logiky nebo zápisu konfigurace do konfigurační paměti využívá SoC rozhraní JTAG. Toto rozhraní je připojeno na dedikované vývody obvodu spolu s dalšími pomocnými signály. Signál *done* v logické jedničce značí úspěšnou konfiguraci programovatelné logiky a signál *program* slouží jako k vyvolání čtení z konfigurační paměti. Zapojení konfiguračního rozhraní programovatelné logiky obvodu Zynq-7000 je na obrázku 4.16. [12]



Obr. 4.16: Zapojení konfiguračního rozhraní SoC

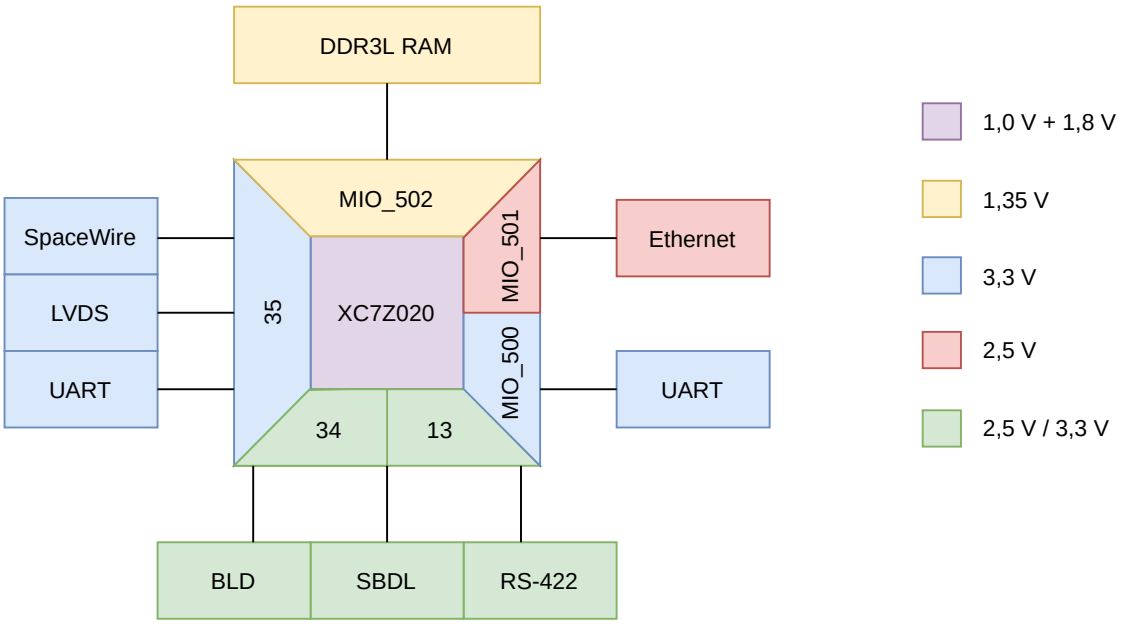
Software běžící v ARM procesoru řídicího obvodu vyžaduje připojení operační paměti. Podporovanými typy jsou LPDDR2 na napětí $1,2\,V$, DDR2 na napětí $1,8\,V$, DDR3 na napětí $1,5\,V$ a DDR3L na napětí $1,35\,V$. Bitová šířka paměťového rozhraní je 16 nebo 32 bitů. Na základě podpory výrobce byla vybrána paměť ISSI IS43TR16256BL-125 s celkovou velikostí $4\,Gbit$ organizovanou jako $16 \times 256\,Mbit$ a rozhraním DDR3L. [12]

4.11 Napájení

Obvody FPGA mají specifické požadavky na napájení. Vnitřní logika, doplňková logika a vstupně-výstupní bloky bývají zpravidla napájeny různými napájecími napětími. Při návrhu je nutné brát v potaz kompatibilitu s připojenými integrovanými obvody.

V testovacím systému je Zynq-7000 připojen k Ethernetu a rozhraní UART přes MIO vývody, které jsou organizovány ve dvou bancích s odděleným napájením. Pro Ethernet je povoleno napájení $1,8\,V$ nebo $2,5\,V$. Rozhraní je k integrovanému obvodu připojeno přes vývody MIO16 - MIO27. Tyto vývody náleží banku MIO_501. Obvod KSZ9021RL podporuje napájecí napětí na RGMII rozhraní $3,3\,V$ nebo $2,5\,V$, pro tento obvod a pro bank MIO_501 bylo proto zvoleno napájení $2,5\,V$. UART je připojen k banku MIO_500 na vývody MIO14 a MIO15. Integrovaný obvod FT230X je napájen ze $3,3\,V$, tímto napětím je proto napájen i bank MIO_500. Operační paměť DDR3L je připojena k banku MIO_502. Napájecí napětí tohoto banku proto musí odpovídat DDR3L standardu a jeho napájení musí být $1,35\,V$.

U ostatních banků je napájení odvozeno od připojených integrovaných obvodů. Přesný počet a typ připojených rozhraní není součástí této práce. Požadavky na napájení zbývajících IO banků programovatelné logiky proto nejsou specifikovány.



Obr. 4.17: Napájecí větve testovacího zařízení

Kromě vstupně-výstupních bloků vyžaduje SoC napájení vnitřní logiky procesoru, doplňkové logiky procesoru, PLL bloků procesoru, vnitřní programovatelné logiky, doplňkové programovatelné logiky a napájení BRAM pamětí v programovatelné logice. Hodnoty těchto napětí jsou podle specifikace výrobce $1,0\,V$ a $1,8\,V$. Napájení jednotlivých součástí testovacího zařízení je na obrázku 4.17. [22]

Výrobce obvodu SoC specifikuje, ve kterém pořadí mají být spínány jednotlivé napájecí větve, případně časové intervaly mezi náběhem jednotlivých větví na nominální úroveň. Postupné spínání napájecích zdrojů je možné řešit za využití dedikovaného sekvenceru připojenému k povolovacím vstupům daných integrovaných obvodů spínaných zdrojů. [22]

Vzhledem k absenci fyzické realizace testovací platformy není možné v této práci řešit kompletní návrh napájecího zdroje. Proudová spotřeba celého zařízení i jednotlivých napájecích větví je závislá na počtu a typu jednotlivých rozhraní. Napájení je možné zajistit za použití integrovaných obvodů spínaných zdrojů, pomocí prefabrikovaných modulů nebo pomocí laboratorních zdrojů.

5 OVLÁDACÍ SOFTWARE

Kromě hardwarového návrhu popsaného v kapitole 4 je součástí této práce popis softwaru vytvořeného ke zprostředkování komunikace s testovacím zařízením. Tento software slouží ke snadnému vytvoření procedur při procesu testování. Z hlediska cílové platformy je softwarová část rozdělena na knihovnu pro řídicí počítač, která zajišťuje vytvoření komunikačního portu protokolu Ethernet za využití dostupných prostředků operačního systému, a na program běžící na výpočetním jádře ARM procesoru integrovaném v obvodu SoC.

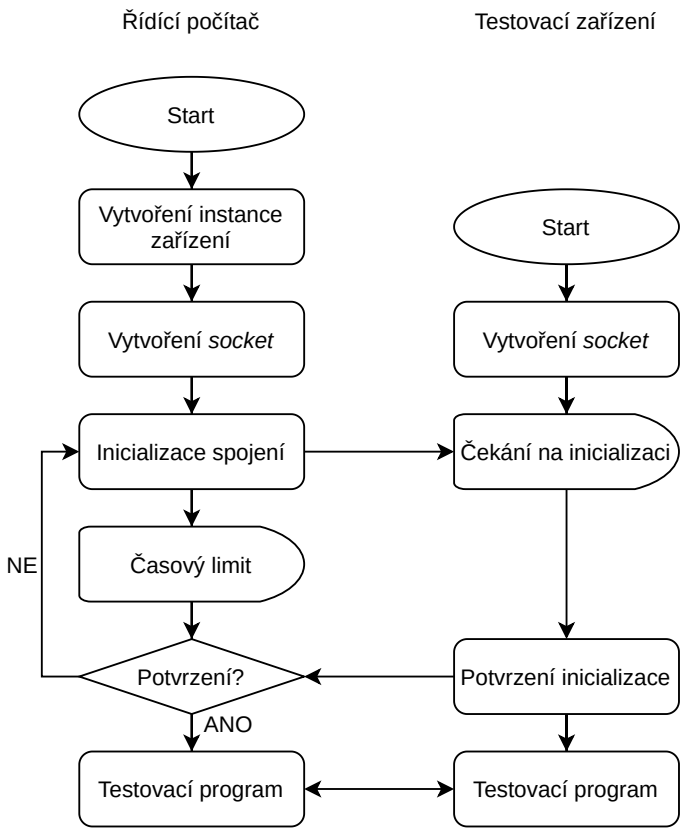
Programová část testovací platformy je navržena tak, aby uživateli zpřístupnila veškeré nízkoúrovňové funkce testovacího zařízení i jednotlivých koncových rozhraní a zároveň poskytla komplexnější funkce související s inicializací a řízením chodu rozhraní při procesu testování.

5.1 Komunikační protokol

Spojení testovacího zařízení a řídicího počítače při procesu testování je spojení mezi dvěma pevnými body bez další síťové infrastruktury. Pro zjednodušení navázání komunikace a využití maximální datové propustnosti fyzické vrstvy rozhraní Ethernetu je komunikace realizována na bázi jednoduchého TCP spojení, tzv. *sockets*. Pro realizaci tohoto spojení je nutné přidělit oběma spojeným zařízením IP adresu a přidělit komunikaci vyhrazený port. Pomocí vytvoření *socket* jsou pak programovému prostředí Python zpřístupněny základní funkce protokolu Ethernet pro odesílání a přijímání dat.

Přidělování IP adres ve složitějších síťových infrastrukturách je zajišťováno DHCP serverem, tento postup není při spojení dvou pevných bodů nutný. IP adresy jsou oběma zařízením přiděleny napevno. Tímto postupem je zamezeno případným adresním kolizím v síti a je zajištěno oddělení testovacího zařízení od místní sítě, ve které se nachází řídicí počítač. Nevýhodou totoho řešení je nutnost využití dvou síťových karet v řídicím počítači v případě nutnosti jeho připojení k vnější síťové infrastruktuře pro potřeby vzdáleného přístupu.

Proces vytvoření komunikačního kanálu je na obrázku 5.1. Prvním krokem každého testovacího programu je vytvoření instance objektu testovacího zařízení z řídicí knihovny. Při tomto vytvoření je pomocí *socket* vytvořen přístup programu k rozhraní Ethernet a je zahájena inicializace spojení vysláním inicializačního

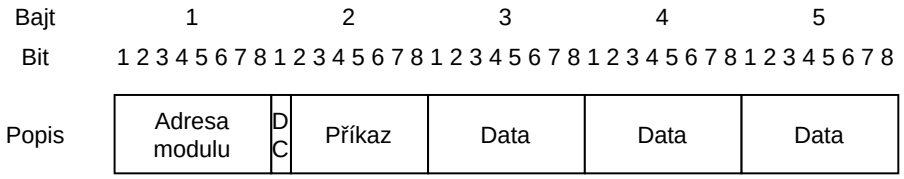


Obr. 5.1: Proces vytvoření komunikačního kanálu

příkazu. Řídicí počítač následně čeká na odpověď ze strany zařízení, případně opakuje odeslání inicializačních dat. Po přijetí potvrzení je spojení považováno za navázané a je možné využít metod instance objektu testovacího zařízení k samotnému testovacímu procesu. Proces spojení je ukončen zrušením instance objektu v testovacím programu nebo fyzickým rozpojením koncových bodů.

Datová struktura odesílané zprávy se skládá z cílového modulu, typu zprávy a případných dalších dat. Adresa modulu je v testovacím zařízení uložena jako osmibitová hodnota, což umožňuje teoreticky připojení až 254 rozhraní. Adresa 0 je vyhrazena pro samotné testovací zařízení a je použita například při inicializační sekvenci. Adresa 255 je rezervována pro vysílání shodného příkazu všem připojeným rozhraním. Datová slova a příkazy jsou posílána ve formátu popsaném na obrázku 5.2. Bit DC rozlišuje příkazy a datová slova, pokud je v logické nule, jedná se o data, pokud je v logické jedničce, jedná se o příkaz.

Tři osmibitová datová pole obsahují v případě datového slova samotná data, která se mají odeslat po cílovém rozhraní. V případě příkazového slova obsahuje toto pole nepovinně doplňující data příkazu.



Obr. 5.2: Komunikační slovo

5.1.1 Příkazy

Příkazy jsou v testovacím systému definovány jednotně pro všechny druhy rozhraní i samotné testovací zařízení. Z toho vyplývá, že některé příkazy nejsou pro konkrétní koncový bod vykonatelné, tedy ani platné. Kompatibilita příkazu a cílového rozhraní je zajištěna použitím tříd konkrétního rozhraní k veškeré datové komunikaci s testovacím zařízením. Instance třídy nepodporující daný příkaz neobsahuje metodu, která je schopná tento příkaz vyslat.

Samotné testovací zařízení je schopné odpovídat na některé typy příkazů. Příkladem je příkaz *ping* používaný při inicializaci bez informace o počtu a typu jednotlivých rozhraní. Každý modul systému odpovídá na příkaz odesláním příkazového slova s hodnotou v poli příkazu 0, která znamená úspěšné provedení příkazu. Mapa všech příkazů spolu s kompatibilními rozhraními je v tabulce 5.1. Příkaz je uložen jako sedmibitová hodnota a je tedy možné implementovat maximálně 128 příkazů.

Tab. 5.1: Seznam příkazů komunikačního protokolu

Hodnota	**Název příkazu**	**Rozhraní**	**Počet dat. bajtů**
0	OK	všechna	1
1	ping	všechna	0
2	enable	všechna kromě TZ	0
3	disable	všechna kromě TZ	0
4	baudrate	UART, LVDS, RS-422, BLD, SBDL, SpaceWire	3
5	direction	RS422, BLD, SBDL	1
6	mode	MIL-STD-1553B	1
7	parity	UART, LVDS, RS-422, SBDL, BLD	1

Příkaz *OK* je návratový příkaz, kterým jednotlivá rozhraní, případně samotné testovací zařízení odpovídají na libovolný příkaz. V datové části je odesílán typ rozhraní, aby bylo možné mu při inicializaci přidělit správnou třídu. Příkaz *ping* zajišťuje navázání spojení s testovacím zařízením při inicializaci nebo zjištění všech připojených rozhraní. Tento příkaz nevyvolává žádné změny, jeho přijetí je v cílovém bodě následováno odpovědí *OK*. Dvojice příkazů *enable* a *disable* slouží k povolení respektive zakázání cílového rozhraní. Zakázané rozhraní neposílá ani nepřijímá data, tedy nekomunikuje po vnitřní sběrnici SoC s výjimkou příkazů. Příkaz *baudrate* nastavuje komunikační rychlost cílového rozhraní. Komunikační rychlost obsažená ve třech datových bajtech tohoto příkazu je přenášena v jednotkách bitů za sekundu a může tedy nabývat hodnot $0 - 16,7\,Mbit/s$. U rozhraní SpaceWire je přenášena hodnota v Mbit/s, aby bylo možné dosáhnout jeho maximální rychlosti. Konkrétní povolené hodnoty se u jednotlivých rozhraní liší, o správnost hodnot se stará softwarová knihovna. Příkaz *direction* specifikuje u cílového rozhraní, zda se jedná o vstup nebo výstup. Vstup je reprezentován nulovou hodnotou v jednom datovém bajtu příkazu. Příkaz *mode* slouží k přepínání rozhraní MIL-STD-1553B mezi módy BC, RT a BM. V jednom datovém bajtu jsou tyto módy reprezentovány hodnotami 0 - BC, 1 - RT a 2 - BM. Příkaz *parity* slouží u podporovaných rozhraní k nastavení paritního bitu, 0 znamená komunikaci bez paritního bitu, 1 lichou paritu a 2 sudou paritu.

5.2 Knihovna pro PC

Programová knihovna řídicího počítače byla vytvořena v programovacím jazyce Python. Tento jazyk byl vybrán z důvodu jednoduché integrace systémových funkcí souvisejících s protokolem Ethernet, z důvodu snadné instalace na řídicím počítači bez nutnosti zakoupení licence a z důvodu jeho funkčnosti na libovolné hardwarové konfiguraci bez ohledu na operační systém. Knihovna je navržena takovým způsobem, aby byl její koncept přenositelný do jiných objektově orientovaných programovacích jazyků pouze úpravou nejnižší vrstvy zabezpečující spojení přes Ethernet pomocí modulu *socket*. Návrh vyšších vrstev je s programovacím jazykem svázán pouze syntaxí.

5.2.1 Architektura knihovny

Knihovna poskytuje uživateli dvě základní třídy. První z nich, třída *SutSystem*, je reprezentací testovacího zařízení. Její metody nesouvisí s žádným konkrétním koncovým rozhraním. Třída *SutInterface* je pak asociována s jedním konkrétním rozhraním jedinečně identifikovatelným pomocí adresy. Tato třída je mateřskou pro další, specifičtější třídy popisující konkrétní typ rozhraní. Určení typu rozhraní je nutná podmínka pro vytvoření instance této třídy. Důvodem existence zvláštní třídy pro každý typ rozhraní je použití specifických metod souvisejícím s funkčním principem rozhraní. Všechny metody, které jsou společné pro libovolný připojený koncový bod jsou sdruženy v obecné třídě *SutInterface*.

5.2.1.1 Popis třídy testovacího zařízení

Třída *SutSystem* je instanciována jednou pro každé testovací zařízení. Jejím úkolem je sdružení atributů souvisejících s celým zařízením a poskytnutí souhrnných metod pro řízení celého systému.

Atributy této třídy jsou v tabulce 5.2 a zahrnují IP adresy řídicího počítače i testovacího zařízení v datovém typu textového řetězce, komunikační port, na němž je vytvářeno spojení pomocí *socket* API, seznam připojených rozhraní, který je implementován jako pole, jehož položky jsou instance objektů třídy *SutInterface* a atribut *socketInst*, který slouží jako instance objektu *socket*.

Tab. 5.2: Seznam atributů třídy SutSystem

Název atributu	**Typ**
`localIP`	`String`
`remoteIP`	`String`
`port`	`Integer`
`liveInterfaces`	`List[SutInterface]`
`socketInst`	`socket`

Metody třídy *SutSystem* jsou v tabulce 5.3. Konstruktor třídy zajišťuje úvodní inicializaci zařízení. Volání této metody způsobí vytvoření komunikačního portu a zavolání příkazu *ping* adresovaného na adresu 0.

Metoda *sendData* využívá *socket* API k odeslání dat do testovacího zařízení. Tato metody není volána přímo z uživatelského programu, je volána ostatními metodami třídy nebo metodami třídy konkrétního rozhraní. Jejími parametry jsou adresa rozhraní (z nichž 255 je adresa vyhrazená pro posílání stejných dat všem

koncovým bodům), logická hodnota *command*, která rozlišuje datová a příkazová slova, a samotná data, která se mají odeslat. Jejich délka není nijak omezena.

Metoda *listConnected* odesílá příkaz *ping* všem zařízením na sběrnici. Slouží k rozpoznání připojených rozhraní. Tato metoda nemá žádné vstupní parametry. Její přijetí v koncovém bodě vyvolává odpověď příkazem *OK*, ve kterém je navrácen typ konkrétního rozhraní.

Tab. 5.3: Seznam metod třídy SutSystem

Název metody	**Parametry**
`__init__`	`localIP(String)`, `remoteIP(String)`
`sendData`	`address(String)`, `command(Bool)`, `data(Bytearray)`
`listConnected`	-

5.2.1.1 Popis třídy rozhraní

Třída *SutInterface* reprezentuje každý připojený koncový bod rozhraní. Každá instance této třídy je unikátně identifikovatelná adresou. Tato adresa je shodná s adresou používanou uvnitř řídicího obvodu k přenosu dat po AXI sběrnici. Adresa 0 je rezervována pro příkazy testovacímu zařízení jako celku a adresa 255 je rezervována pro přenos všem uzlům najednou. Teoretický maximální počet současně připojených rozhraní je tak 254. Atribut *linkOK* značí rozpoznání testované jednotky připojené přes koncový bod. Rozhraní, která rozpoznávání neumožňují nebo jsou pouze výstupní mají hodnotu atributu konstantně v logické jedničce. Další atributy třídy uvedené v tabulce 5.4 jsou rozděleně podle typu rozhraní, u kterých jsou implementovány.

Tab. 5.4: Seznam atributů třídy SutInterface

Název atributu	**Typ**	**Rozhraní**
`address`	`Integer`	všechna
`linkOK`	`Bool`	všechna
`mode`	`String`	MIL-STD-1553B
`baudrate`	`Integer`	UART, LVDS, RS422, BLD, SBDL, SpaceWire
`direction`	`Integer`	LVDS, RS422, BLD, SBDL
`mode`	`String`	MIL-STD-1553B

5.3 Programová část řídicího obvodu

Příjem a odpovědi na datová a příkazová slova zajišťuje v systému výpočetní jednotka ARM procesoru v řídicím obvodu. Ta se zároveň stará o vytvoření a udržování spojení s počítačem pomocí protokolu Ethernet. Sada nástrojů Vivado od výrobce SoC obsahuje vývojové prostředí Xilinx SDK umožňující použití jazyků C a C++ pro software běžící v integrovaném procesoru. Součástí vývojových nástrojů jsou i podpůrné balíky pro navázání procesoru, programovatelné logiky a MIO vstupů a výstupů.

Jednotlivé programové procesy mohou v ARM procesoru běžet samostatně nebo mohou být řízené operačním systémem. Součástí distribuce vývojových nástrojů jsou připravené verze systémů Linux a FreeRTOS. V případě testovacího zařízení není k využití operačního systému důvod. Program běžící v procesoru je jednovláknový a v případě nutnosti využití druhého vlákna je k dispozici druhé výpočetní jádro.

5.3.1 Ethernet

Pro implementaci Ethernetu ve verzi bez operačního systému nabízí Xilinx programovou knihovnu *lwIP*. Tato knihovna nabízí základní funkce Ethernetu související s protokoly TCP/IP, ARP nebo DHCP a zajišťuje mezivrstvu mezi uživatelskou aplikací a ovladačem Ethernetu integrovaným v SoC. Knihovna *lwIP* ve verzi bez operačního systému umožňuje pouze přístup k *raw* API založené na volání návratových funkcí při přijetí dat, úspěšném odeslání dat nebo chybě. [24]

Ethernet vyžaduje k platnému spojení správné nastavení IP adres a výchozích bran. Toto nastavení je možné provést přes implementovaný UART nebo je možné využívat výchozí hodnoty a nastavit kompatibilně síťovou kartu počítače. Výchozí hodnoty jsou 192.168.1.10 pro IP adresu a 7 pro poslouchající port.

Sériový port není v systému implementován jako hlavní komunikační rozhraní. Jeho funkce je omezena na ladění a nastavení připojení přes Ethernet. Proto testovací zařízení neumožňuje použití sériového portu k přístupu ke koncovým rozhraním nebo registrům souvisejícím s jejich nastavením. Jediná data, ke kterým má sériový port přístup jsou právě parametry nastavení softwarové knihovny *lwIP*. Vyjednávání o rychlosti spojení, rychlost a duplex spojení jsou parametry inicializované samotnou knihovnou, jejich změna je možná pouze kompilací softwarové aplikace se změněnými

parametry. V implementaci *raw* API knihovny *lwIP* pak zbývají pouze parametry související s protokolem TCP/IP, tedy IP adresa, maska podsítě, výchozí brána a port. Masku podsítě ani výchozí bránu není nutné při spojení dvou bodů měnit a jejich hodnota je trvale nastavena na 255.255.255.0 pro masku a 192.168.1.1 pro výchozí bránu. Případná změna těchto parametrů je možná přímo ve zdrojovém kódu. IP adresu lze změnit zasláním příkazu `IP!XXXXXXXX` skrz sériový port, kde `XXXXXXXX` je IP adresa odesílaná jako 4 bajty. Port lze změnit zasláním příkazu `PORT!XXXX`, kde `XXXX` jsou dva bajty reprezentující port. Port lze nastavit v rozsahu 0 - 65535. Sériový port je využíván také samotnou knihovnou *lwIP* pro účely zasílání stavových informací.

5.3.2 Předávání dat

Při příjmu dat z řídicího počítače jsou přijaté bajty rozděleny na adresu, příznak příkazu, typ příkazu a zbytek dat. Tento proces je pro většinu odeslané zprávy transparentní a data jsou pouze rozbalena a předána s přidruženou adresou na AXI sběrnici. Výjimkou jsou příkazy adresované přímo testovacímu zařízení s adresou 0, na které odpovídá přímo procesor bez využití sběrnice.

Oddělením adresy zůstávají z komunikačního slova 4 bajty, které jsou beze změny přenášeny 32bitovou AXI sběrnicí. Analogický proces se děje v opačném směru. Datům přijatým procesorem po AXI rozhraní je přidružena adresa podle vysílajícího modulu a výsledných 5 bajtů je odesláno skrze protokol Ethernetu do počítače.

6 VYHODNOCENÍ VÝSLEDKŮ

Tato kapitola popisuje vyhodnocení návrhu hardwarové a softwarové části testovací platformy. Hardwarová část je v této práci navržena zejména ve formě obvodových schémat zapojení fyzických vrstev koncových rozhraní testovacího systému a fyzických vrstev rozhraní potřebných pro komunikaci s řídicím počítačem. Softwarová část je navržena ve dvou oddělených částech, softwarové knihovně pro počítač, jejímž úkolem je poskytnutí potřebných funkcí uživateli při procesu testování a programu pro ARM procesor zpracovávající data v obvodu SoC.

6.1 Vyhodnocení hardwarové části

Návrh hardwaru testovacího zařízení se dotýká hlavně fyzických vrstev jednotlivých rozhraní. Tato rozhraní jsou teoreticky popsána v kapitole 2. Většina požadavků je stanovena normami ECSS, u rozhraní s obecnějším použitím jsou to pak normy ANSI/TIA/EIA a IEEE. Stanovené požadavky jsou normami upraveny primárně pro vyvíjené zařízení do vesmírných aplikací, testovací zařízení navržené v této práci je tak normami vázáno pouze v míře, která zaručí jejich funkčnost při procesu testování.

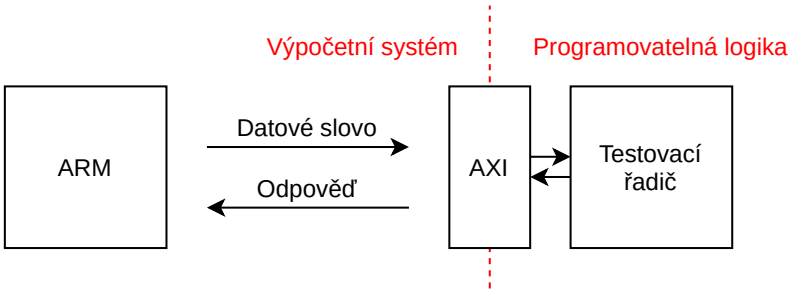
Dodržení souladu s normami bylo důležité hlavně u dvou rozhraní definovaných přímo pro vesmírné aplikace - SpaceWire a MIL-STD-1553B. Dodržení norem u rozhraní SpaceWire na signálové úrovni je zaručeno jejím odvozením od TIA normy pro LVDS. Pro jeho připojení proto byly zvoleny budící obvody, které tuto normu splňují. Rozhraní MIL-STD-1553B je definováno pouze původní normou Ministerstva obrany USA a doplňující normou ECSS. Dodržení norem na hardwarové vrstvě je u návrhu zajištěno dedikovaným integrovaným obvodem pro toto rozhraní.

Další rozhraní pro vesmírné aplikace SHP, EHP, SLP, BLD a SBDL jsou pouze upřesněním obecných konceptů o toleranci chyb a definované hodnoty napětí a proudů. Shoda hardwarového návrhu s normou je zajištěna použitím vhodných oddělovacích integrovaných obvodů a porovnáním jejich parametrů s hodnotami určenými normou.

Obecně používaná rozhraní RS-422, LVDS, UART a MDI/RGMII pro Ethernet jsou specifikována pouze normami IEEE a TIA. Pro tato rozhraní byly použity standardní ASSP obvody podle doporučených zapojení.

Návrh řadičů jednotlivých rozhraní zajišťující sjednocení komunikačního protokolu je navržen pro programovatelnou logiku řídicího obvodu SoC. Jeho složitost je závislá na složitosti komunikace po daném rozhraní. Pro vrstvová rozhraní SpaceWire a MIL-STD-1553B je součástí návrhu tohoto řadiče také IP jádro třetí strany.

Všechny řadiče v programovatelné logice mají společnou část, která umožňuje jejich připojení na AXI rozhraní ARM procesoru. Komunikace mezi ARM procesorem a moduly v programovatelné logice byla otestována pomocí testovacího řadiče neobsahujícího bloky žádného konkrétního rozhraní pouze za účelem ověření konceptu obousměrného přenosu informace mezi výpočetním jádrem a programovatelnou logikou. Jednoduché testovací schéma ověřující funkci AXI rozhraní je na obrázku 6.1. Z procesoru bylo vysláno datové slovo, které bylo řadičem přečteno a následně přečteno příslušnými kanály AXI rozhraní.



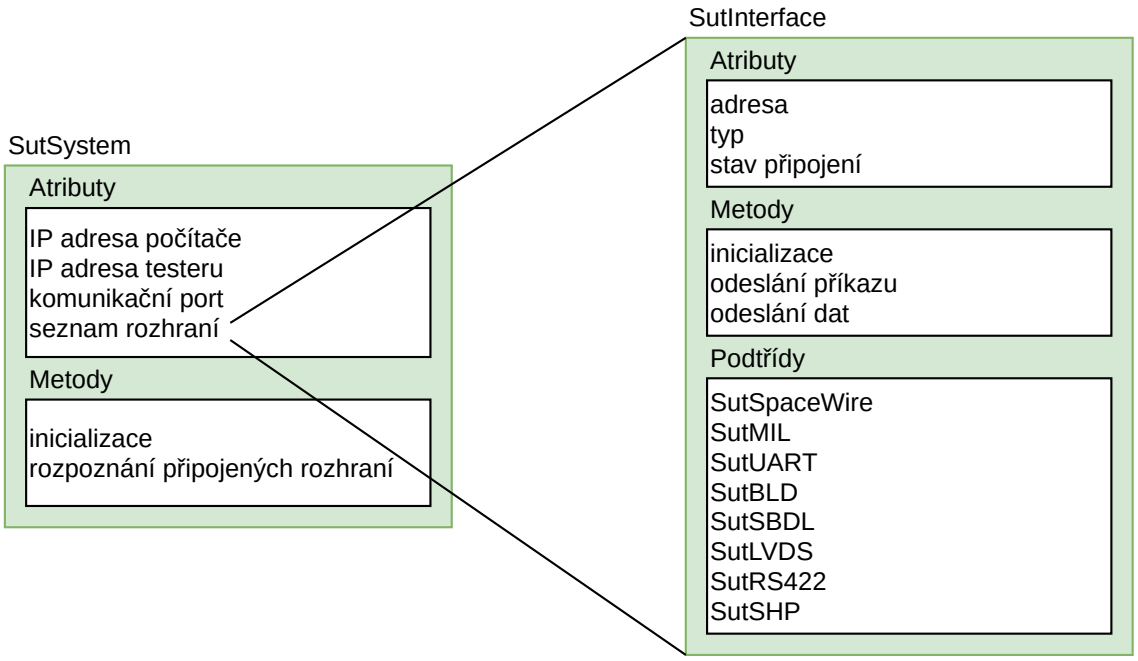
Obr. 6.1: Test komunikace po AXI rozhraní

6.2 Vyhodnocení softwarové části pro PC

Pro ovládání testovacího zařízení je v této práci navržena uživatelská knihovna v jazyce Python. Tato knihovna obsahuje třídy pro vytvoření instancí celého testovacího zařízení a následně i jednotlivých připojených rozhraní. Jejím cílem je snadné sestavení testovacích procedur využitím připravených metod.

Zdrojový kód vytvořené knihovny je přiložen k práci jako příloha **??**. Třída `SutSystem` reprezentující celé zařízení obsahuje metody pro inicializaci, třída `SutInterface` a její podtřídy pak reprezentují každé rozhraní, rozlišitelné adresou a typem. Vizualizace architektury knihovny je na obrázku 6.2.

Způsob využití knihovny k vytvoření testovací procedury je popsán ve zdrojovém kódu. Funkčnost byla otestována jednoduchými testovacími programy. Příklad



Obr. 6.2: Architektura uživatelské knihovny

programu, který provede inicializaci zařízení, provede změnu komunikační rychlosti rozhraní UART a následně odešle po tomto rozhraní data je v následujícím výpisu. Název modulu obsahující knihovnu je v tomto případě `interfaces.py`.

```
from interfaces import *

# Inicializace třídy, přidělení IP adres
tester = SutSystem('192.168.1.1', '192.168.1.10')

# Rozpoznání připojených rozhraní
tester.probeInterfaces()

# Přístup k objektu rozhraní UART
for ifc in tester.liveInterfaces:
    if ifc.__class__.__name__ == 'SutUART':
        # Změna komunikační rychlosti
        tester.sendData(ifc.address, COM_BAUDRATE, 9600)

        # Odeslání dat
        tester.sendData(ifc.address, False, 'ABC')
        break
```

7 ZÁVĚR

V rámci této diplomové práci byl řešen návrh testovací platformy směřující k použití ve vývoji aplikací pro vesmírný sektor. Koncepce tohoto návrhu je popsána v kapitole 1. Systém je tvořen řídicím počítačem a testovacím zařízením zajišťujícím překlad komunikace a návrhem hardwaru řadičů cílových rozhraní.

Požadavky vyplývající z konceptu jsou popsány v kapitole 2, která je především teoretickým shrnutím použitých norem. Pro rozhraní související s vesmírným vývojem byly zdrojovými dokumenty zejména ECSS normy definující vrstvová rozhraní SpaceWire, MIL-STD-1553B, rozhraní pro obecnou komunikaci jako SBDL a BLD nebo rozhraní pro spínání výkonových zátěží SHP, EHP a SLP. Doplňující infomace byly převzaty z původní normy pro MIL-STD-1553B vydané Ministersvem obrany USA a dalších dokumentů vydaných Evropskou vesmírnou agenturou. Součástí zadání práce byl také požadavek na implementaci obecných komunikačních rozhraní RS-422, LVDS a UART. Pro teoretický popis těchto rozhraní byly použity standardy TIA a IEEE. Kromě rozhraní cílových, určených ke komunikaci s testovanou jednotkou, je testovací zařízení připojeno rozhraním Ethernet k řídicímu počítači. Specifikace Ethernetu byla převzata ze standardu IEEE 802.3.

Kapitola 3 popisuje návrh zařízení související s řídicím obvodem. Tím je v navrženém systému obvod SoC Zynq-7000 vyráběný společností Xilinx. Tento obvod integruje výpočetní jednotku postavenou na dvou jádrech procesoru ARM Cortex-A9 a programovatelnou logiku vycházející ze sedmé generace hradlových polí Artix-7 a Kintex-7 firmy Xilinx. Funkce výpočetní jednotky obvodu je v systému spojena zejména s komunikací přes protokoly Ethernetu. Tato varianta řídicího obvodu byla zvolena právě s důrazem na snadnou implementaci a případnou úpravu komunikace mezi řídicím počítačem a testovacím zařízením. Programovatelná logika pak v systému zajišťuje paralelní funkci řadičů jednotlivých koncových rozhraní a převod jejich komunikačních protokolů na společný formát. V kapitole 3 je dále popsáno navržené řešení mechanizmu předávání dat mezi výpočetní jednotkou a programovatelnou logikou. To je řešeno využitím jednoho portu AXI rozhraní implementovaného v SoC, přičemž výpočetní jednotka plní v systému úlohu zařízení *master* a jednotlivé moduly řadičů rozhraní jsou připojeny v módu *slave*. Popis funkce řídicího obvodu je doplněn o výběr konkrétní součástky, který je však závislý na konkrétních požadavcích projektu, na kterém má být testovací systém využit.

Hardwarová část návrhu popsaná v kapitole 4 vychází zejména z požadavků norem rozebraných v teoretickém rozboru. Pro každé cílové rozhraní je v této kapitole navrženo obvodové zapojení mezi řídicím obvodem a bodem připojení k testované jednotce. Pro rozhraní SpaceWire je splnění požadavků zajištěno použitím integrovaného obvodu vysílače a přijímače, který podle parametrů udávaných výrobcem splňuje normy LVDS komunikace a proto na fyzické vrstvě splňuje také normu ECSS. Implementace rozhraní MIL-STD-1553B je řešena využitím dedikovaného integrovaného obvodu Holt Integrated Circuits, který spolu s impedančním přizpůsobením a galvanickým oddělením vyhovuje normě ECSS popisující toto rozhraní. Pro ostatní rozhraní byly k obvodovému návrhu využity standardní oddělovací obvody a posouvače úrovní, jejichž parametry byly porovnány s požadavky norem. Obvodové zapojení rozhraní Ethernet je omezeno požadavky řídicího obvodu na využití RGMII a je zajištěno integrovaným obvodem KSZ9021RL.

Druhou částí hardwarového návrhu je RTL popis řadičů v programovatelné logice. Část funkcí těchto řadičů je určena přímo koncepcí systému a souvisí s dekódováním a vykonáváním příkazů definovaných v komunikačním protokolu. Tyto příkazy slouží k nastavení parametrů datové komunikace a navázání spojení při inicializaci systému. Pro rozhraní SpaceWire a MIL-STD-1553B je součástí řadiče také IP jádro třetí strany, jehož návrh není předmětem této práce.

Softwarová část testovacího systému je popsána v kapitole 5. Její první složkou je programová knihovna pro řídicí počítač obsahující třídy, jejichž pomocí je při testovacím procesu celé zařízení ovládáno. Tato knihovna je navržena s důrazem na možné rozšíření typu rozhraní při použití v testovacím procesu projektu. Knihovna je vytvořena v programovacím jazyce Python a přidružuje objekty celému testovacímu zařízení i jeho koncovým rozhraním. Metody těchto objektů jsou pak výchozím stavebním blokem testovací procedury v procesu testování.

Program běžící v ARM procesoru je spojen se dvěma úlohami - komunikací po rozhraní Ethernet s řídicím počítačem a komunikací po rozhraní AXI s moduly řadičů rozhraní. Pro inicializaci spojení po Ethernetu byla použita softwarová knihovna *lwIP* dodávaná s distribucí vývojového prostředí Xilinx Vivado. Výhodou této knihovny je možnost jejího použití bez nutnosti běhu operačního systému v obvodu SoC. Knihovny zajišťující komunikaci po AXI rozhraní jsou součástí vývojového procesu nástrojů Vivado a jsou vázány přímo na cílový hardware.

Celý systém je spojen komunikačním protokolem, implementovaným ve všech složkách návrhu - v softwarové knihovně, v programu běžícím v řídicím obvodu i v řadičích jednotlivých rozhraní. Tento protokol je navržen tak, aby omezil řežii při přenosu dat, ale umožňoval využít všechny funkce implementované v systému.

V této práci byly navrženy následující součásti testovací platformy:

- Obvodové zapojení rozhraní SpaceWire, MIL-STD-1553B, LVDS, RS-422, UART, Ethernet, SBDL, BLD, SHP, EHP a SLP pro spojení řídícího obvodu a přípojného bodu daného rozhraní.

- Řadiče těchto rozhraní připojitelná na AXI sběrnici v obvodu SoC.

- Vnitřní zapojení obvodu SoC využívající jedno jádro ARM procesoru a programovatelnou logiku.

- Program pro ARM procesor předávající data mezi Ethernetem a AXI rozhraním.

- Softwarová knihovna v jazyce Python pro řídicí počítač.

Testovací platforma popsaná v této práci je navržena obecně pro použití na budoucích projektech. Přesná míra implementace jednotlivých bloků je závislá na požadavcích těchto konkrétních projektů a platforma byla navržena s cílem usnadnit její modifikovatelnost v počtu a typu implementovaných rozhraní. Využitím AXI sběrnic v obvodu SoC je zajištěno případné snadné připojení dalších modulů řadičů.

LITERATURA

[1] *IEEE Standard for Ethernet*. 802.3-2018. New York: IEEE Computer Society, 2018.

[2] *Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits*. Rev. A. Arlington: Telecommunications Industry Association, 2001.

[3] *An Overview of LVDS Technology* [online]. Dallas: Texas Instruments, 1998 [cit. 2019-12-13]. Dostupné z: http://www.ti.com/lit/an/snla165/snla165.pdf

[4] *Electrical Characteristics of Balanced Voltage Digital Interface Circuits*. Rev. B. Arlington: Telecommunications Industry Association, 1994.

[5] *RS-485/RS-422 Circuit Implementation Guide* [online]. Norwood: Analog Devices, 2008 [cit. 2019-12-06]. Dostupné z: https://www.analog.com/media/en/technical-documentation/application-notes/AN-960.pdf

[6] *SpaceWire: Links, nodes, routers and networks*. 1. Noordwijk: European Cooperation for Space Standardization, 2019.

[7] *Aircraft Internal Time Division Command/Response Multiplex Data Bus*. MIL-STD-1553B. Washington D.C.: Department of Defense, 1979.

[8] *Interface and communication protocol for MIL-STD-1553B data bus onboard spacecraft*. 1. Noordwijk: European Cooperation for Space Standardization, 2008.

[9] *ExoMars Rover Vehicle: General Design and Interface Requirements* [online]. Paris: Astrium, 2010 [cit. 2020-05-27]. Dostupné z: http://emits.sso.esa.int/emits-doc/ASTRIUMLIM/Exomars_Rover_Cameras/EXM-RM-RQM-ASU-0015_Iss3-1_RV-GDIR.pdf

[10] *Zynq-7000 SoC Data Sheet: Overview* [online]. San Jose: Xilinx, 2018 [cit. 2020-05-15]. Dostupné z: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[11] *AXI Reference Guide* [online]. San Jose: Xilinx, 2012 [cit. 2020-05-26]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/v13_4/ug761_axi_reference_guide.pdf

[12] *Zynq-7000 SoC Technical Reference Manual* [online]. San Jose: Xilinx, 2018 [cit. 2020-05-21]. Dostupné z: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf

[13] *Zynq-7000 SoC Packaging and Pinout* [online]. San Jose: Xilinx, 2018 [cit. 2020-05-21]. Dostupné z: https://www.xilinx.com/support/documentation/user_guides/ug865-Zynq-7000-Pkg-Pinout.pdf

[14] *DS90LV031A 3-V LVDS Quad CMOS Differential Line Driver* [online]. Dallas: Texas Instruments, 2016 [cit. 2020-05-18]. Dostupné z: http://www.ti.com/lit/ds/symlink/ds90lv031a.pdf?ts=1588760040678

[15] *DS90LV032A 3-V LVDS Quad CMOS Differential Line Receiver* [online]. Dallas: Texas Instruments, 2016 [cit. 2020-05-18]. Dostupné z: http://www.ti.com/lit/ds/symlink/ds90lv032a.pdf?ts=1590501390683

[16] *HI-1579, HI-1581: 3.3V Monolithic Dual Transceivers* [online]. Mission Viejo: Holt Integrated Circuits, 2017 [cit. 2020-05-29]. Dostupné z: http://www.holtic.com/documents/99-hi-1579_v-rev-rpdf.do

[17] *3.0 V to 5.5 V, ±12 kV IEC ESD Protected, 500 kbps / 50 Mbps RS-485 Transceivers* [online]. Norwood: Analog Devices, 2019 [cit. 2020-05-29]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/ADM3061E-3062E-3063E-3064E-3065E-3066E-3067E-3068E.pdf

[18] *TXB0104 4-Bit Bidirectional Voltage-level Translator With Automatic Direction Sensing and ±15-kV ESD Protection* [online]. Dallas: Texas Instruments, 2018 [cit. 2020-05-24]. Dostupné z: http://www.ti.com/lit/ds/symlink/txb0104.pdf?ts=1590741399033

[19] *ULN2803A Darlington Transistor Arrays* [online]. Dallas: Texas Instruments, 2017 [cit. 2020-05-29]. Dostupné z: http://www.ti.com/lit/ds/symlink/uln2803a.pdf?ts=1590738722493

[20] *KSZ9021RL/RN: Gigabit Ethernet Transceiverwith RGMII Support* [online]. San Jose: Micrel, 2014 [cit. 2020-05-29]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/ksz9021rl-rn_ds.pdf

[21] *FT230X USB to Basic UART IC* [online]. Glasgow: Future Technology Devices International, 2016 [cit. 2020-05-29]. Dostupné z: https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT230X.pdf

[22] *Zynq-7000 DC and Switching Characteristics* [online]. San Jose: Xilinx, 2018 [cit. 2020-05-21]. Dostupné z: https://www.xilinx.com/support/documentation/data_sheets/ds187-XC7Z010-XC7Z020-Data-Sheet.pdf

[23] *SpaceWire Codec* [online]. Milton Keynes: 4Links, 2019 [cit. 2020-05-26]. Dostupné z: https://4links.co.uk/downloads/ip/SpW_Codec_Getting_Started.pdf

[24] *PS and PL-Based Ethernet Performance with LightWeight IP Stack* [online]. San Jose: Xilinx, 2017 [cit. 2020-05-26]. Dostupné z: https://www.xilinx.com/support/documentation/application_notes/xapp1306-ps-pl-ethernet-performance-lwip.pdf