VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

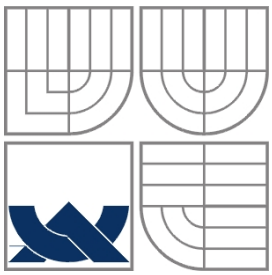
DIGITÁLNÍ OSCILOSKOP NA PLATFORMĚ FITKIT

DIPLOMOVÝ PROJEKT
MASTER'S THESIS

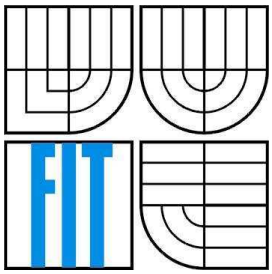
AUTOR PRÁCE
AUTHOR

Bc. ONDŘEJ VEŠKRNA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DIGITÁLNÍ OSCILOSKOP NA PLATFORMĚ FITKIT

DIGITAL OSCILLOSCOPE ON FITKIT PLATFORM

DIPLOMOVÝ PROJEKT
MASTER'S THESIS

AUTOR PRÁCE — Bc. Ondřej Veškrna
AUTHOR

VEDOUCÍ PRÁCE — Ing. Václav Šimek
SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2007/2008

Zadání diplomové práce

Řešitel: **Veškrna Ondřej, Bc.**
Obor: Počítačové systémy a sítě
Téma: **Digitální osciloskop na platformě FITkit**
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s výukovou platformou FITkit a jazykem VHDL pro návrh číslicového hardware.
2. Zpracujte krátkou studii ohledně problematiky převodu analogového signálu na číslicový.
3. Navrhněte koncepci osciloskopu na bázi FPGA a rychlých ADC převodníků v podobě rozšiřujícího modulu platformy FITkit.
4. V návrhovém prostředí Eagle nebo KiCAD vytvořte desku plošných spojů pro uvažovaný modul osciloskopu.
5. V jazyce VHDL vytvořte obvod pro řízení průběhu konverze analogového signálu a přenos získaných vzorků do PC.
6. Demonstrace funkčnosti vámi navrženého přípravku bude provedena formou zobrazení vzorků dat v jednoduché aplikaci.
7. Shrňte dosažené výsledky a diskutujte možnosti dalšího rozšíření.

Literatura:

- Pedroni, V. A. *Circuit Design with VHDL*, Massachusetts Institute of Technology, 2004, 376 s., ISBN 0-262-16224-5
- Noergaard, T. *Embedded Systems Architecture*, Elsevier, Burlington, 2005, 657 s., ISBN 0-7506-7782-9
- Ball, S. R. *Analog Interfacing to Embedded Microprocessor Systems*, Elsevier, New York, 2004, 335 s., ISBN 0-7596-7723-6
- Kester, W. *Data Conversion Handbook*, Elsevier, New York, 2005, 976 s., ISBN 0-7506-7841-0
- Baker, B. *A Baker's Dozen - Real Analog Solutions for Digital Designers*, Elsevier, New York, 2005, 362 s., ISBN 0-7506-7819-4

Při obhajobě semestrální části diplomového projektu je požadováno:

- Splnění bodů 1-3 zadání.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

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

Vedoucí: **Šimek Václav, Ing.**, UPSY FIT VUT
Datum zadání: 24. září 2007
Datum odevzdání: 19. května 2008

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

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Bc. Ondřej Veškrna**
Id studenta: 89473
Bytem: Benetice 4, 675 07 Čechtín
Narozen: 19. 02. 1984, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

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

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

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

Název VŠKP: Digitální osciloskop na platformě FITkit
Vedoucí/školitel VŠKP: Šimek Václav, Ing.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: .....

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

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

Článek 2
Udělení licenčního oprávnění

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

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

Článek 3
Závěrečná ustanovení

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

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Práce se zabývá návrhem zařízení, které umožní sledovat průběh měřeného signálu na obrazovce počítače na principu digitálního osciloskopu. Řídicím prvkem zařízení je programovatelné hradlové pole osazené na platformě FITkit. Konfigurace hradlového pole má za úkol řídit vzorkování vstupního signálu a získané vzorky odesílat prostřednictvím USB rozhraní do počítače. Aplikace implementovaná na PC se pokusí signál rekonstruovat a výsledek zobrazí na monitoru počítače.

Klíčová slova

Platforma FITkit, FPGA, programovatelné hradlové pole, analogově číslicový převod, A/D převodník, vestavěný systém, mikrokontrolér, MCU, SPI, USB rozhraní, C#, VHDL

Abstract

This thesis deals with the design of a device that enables to monitor the behavior of the measured signal on the computer screen, using the principle of the digital oscilloscope. The control element of the device is the field programmable gate array (FPGA) on FITkit platform. The FPGA configuration controls the input signal sampling and sends the received samples through the USB interface to the PC. The graphical application implemented in the computer tries to restore the signal and then displays it on the screen.

Keywords

FITkit platform, FPGA, programmable gate array, analog to digital conversion, A/D converters, microprocessor, embedded system, MCU, SPI, USB interface, C#, VHDL

Citace

Veškrna Ondřej: Digitální osciloskop na platformě FITkit. Brno, 2008, diplomový projekt, FIT VUT v Brně

Digitální osciloskop na platformě FITkit

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením Ing. Václava Šimka
Další informace mi poskytli Dr. Ing. Otto Fučík, Ing. Tomáš Martínek a Ing. Zdeněk Vašíček.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Veškrna Ondřej
27.12.2007

Poděkování

Především bych rád poděkoval vedoucímu své diplomové práce panu Ing. Václavu Šimkovi za odborné vedení a čas věnovaný konzultaci této práce. Také bych chtěl poděkovat panu Ing. Tomáši Martínkovi a Ing. Zdeňkovi Vašíčkovi za odbornou konzultaci problému syntézy číslicových obvodů.

Obsah

1 Úvod

Osciloskop patří bezesporu mezi velice důležité měřící přístroje a stává se tak nenahraditelným pomocníkem v mnoha oborech lidské činnosti. Je nezastupitelný při mnoha měřeních, kdy jiná měřidla neuspějí, anebo jimi zjištěné údaje nemají větší význam, než čistě orientační.

Osciloskop je nepochybně velmi užitečný přístroj. Přes všechny jeho výhody však není mezi studenty či zájemci o elektroniku příliš rozšířen. Hlavním důvodem tohoto faktu je zajisté jeho vysoká cena, která dokáže odradit i většinu horlivých zájemců. Proto se začaly vyrábět osciloskopické adaptéry (buď jako externí přístroje, nebo karty do rozšiřujících slotů), které umožňují sledovat průběhy signálů na osobním počítači. Přístroj potom pouze vzorkuje vstupní signál a naměřené hodnoty následně odesílá přes vhodné rozhraní do počítače. O jejich zpracování a správné vykreslení se již postará příslušný software. Takové zařízení je již samozřejmě poměrně levnější, avšak i tak je jeho cena dost vysoká.

Při realizaci podobného zařízení jsou kladeny vysoké nároky nejen na rychlost a kvalitu analogově digitálních převodníků, rychlost řídicích obvodů, ale zejména je zapotřebí velice rychlých pamětí na uložení naměřených vzorků.

K řízení rychlých převodníků, ukládání vzorků a následnému přesunu vzorků do PC bude použita výuková platforma FITkit, přičemž její výkonná část – FPGA čip je pro tuto činnost vhodná. Výhodou FITkitu je jeho dostupnost každému studentovi Fakulty informačních technologií Vysokého učení technického v Brně.

Práce je logicky členěna do několika kapitol. V teoretické části se zabývám problematikou převodu analogového signálu na číslicový, následuje kapitola s přehledem A/D převodníků. Teoretickou část završuje seznámení s platformou FITkit.

V páté kapitole je proveden rozbor realizace daného problému včetně patřičných blokových schémat. Poslední kapitola je věnována závěru.

2 Převod analogového signálu na číslicový

Číslicové zařízení může zpracovávat signál jen v číslicovém tvaru. Většina přirozených zdrojů a často používaných snímačů jej však dodává ve tvaru analogovém. Pro jeho číslicové zpracování je tedy nezbytné převést tento signál na digitální [2].

Analogově číslicový převod je elektronický proces, ve kterém je spojitá veličina (analogový signál) změněna na diskrétní (víceúrovňový) digitální signál, bez změn jeho základního obsahu [3].

Převod analogového signálu na číslicový tvar se provádí ve dvou krocích. Analogový signál je nejprve periodicky vzorkován, tj. je získán periodický sled úzkých impulsů, který je amplitudově modulován přiváděným analogovým signálem. Ve druhém kroku jsou amplitudy jednotlivých impulsů převáděny na číslicový tvar, což nazýváme kvantování [2].

2.1 Vzorkování

Vzorkování je proces, v němž je signál souvislého času nahrazován jeho částmi - vzorky. Vzorky jsou od sebe zpravidla rovnoměrně vzdáleny (rovnoměrné vzorkování) o vzorkovací periodu T_V. Vzorkovací kmitočet (rychlost vzorkování) je pak převrácenou hodnotou vzorkovací periody [2]:

$$f_v = \frac{1}{T_V}$$

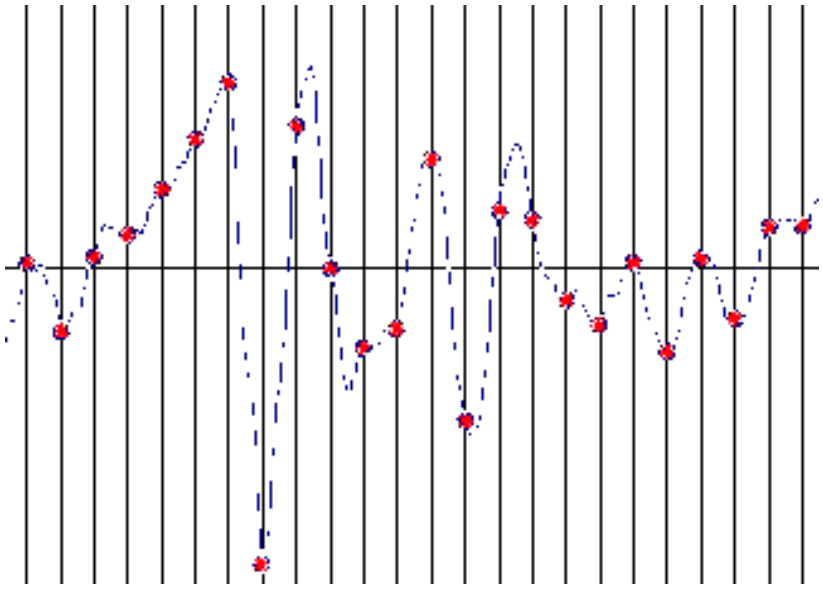
Rovnoměrné vzorkování lze chápat jako násobení signálu souvislého času periodickým vzorkovacím signálem. Pro zjednodušení výpočtů zavádíme ideální vzorkování, při němž je vzorkovacím signálem posloupnost Diracových impulsů. Reálné vzorkování je realizováno spínačem, který je po dobu odběru vzorku sepnut, jinak rozepnut. Obrázek 2-2 znázorňuje, jak může vypadat vzorkovaný signál. Znázorněná kolečka naznačují jednotlivé vzorky.

Důležitou vlastností vzorkovaného signálu je periodicita jeho spektra. Díky této periodicitě vzniká při vzorkování jisté nebezpečí nevratné ztráty informace v důsledku překrytí spekter dvou sousedních period. Aby k tomuto překrytí nemohlo dojít, je třeba, aby byl splněn vzorkovací (Shannonův-Nyquistův-Kotělnikův) teorém, který říká [4]:

„Přesná rekonstrukce spojitého, frekvenčně omezeného, signálu z jeho vzorků je možná tehdy, pokud byl vzorkován frekvencí alespoň dvakrát vyšší než je maximální frekvence rekonstruovaného signálu.“

$$f_v > 2 \cdot f_{\max},$$

kde f_{max} je maximální frekvenční složka spektra původního spojitého signálu. Tato podmínka se v praxi zajišťuje tzv. antialiasingovým filtrem typu dolní propust s mezním kmitočtem do poloviny maximálního možného kmitočtu zdrojového signálu. Tento filtr se zařadí mezi zdroj signálu a zařízení, které signál vzorkuje [2].



Obrázek 2-1 Vzorkování spojitého signálu

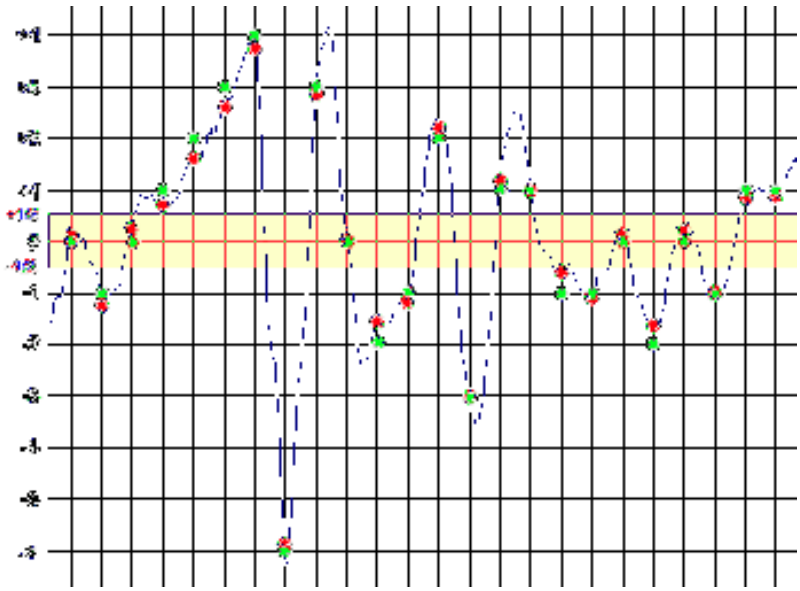
2.2 Kvantování

Kvantováním se mění signál se spojitou množinou hodnot na signál s diskrétní množinou hodnot. Každý vzorek je pak vyjádřen konečným počtem cifer (zaokrouhlování nebo usekávání) a toto číslo je vyjádřeno vhodným číselným kódem. Nejmenší možný skok kvantovaného signálu se nazývá *kvantovací krok* a rozdíl mezi vstupním a výstupním signálem kvantizačního obvodu *kvantizační šum r(t)* [2]. Kvantování vzorků signálu znázorňuje obrázek 2-3.

Při kvantování tedy dochází ke ztrátě informace. Stupeň narušení původního signálu můžeme popsat činitelem kvantizačního zkreslení *k*, který je definován jako poměr efektivní hodnoty R_{ef} kvantovacího šumu a efektivní hodnoty S_{ef} užitečného signálu. Podle literatury [5] lze činitel kvantovacího zkreslení vyjádřit vztahem:

$$k = \frac{R_{ef}}{S_{ef}} = \sqrt{\frac{2}{3}} \cdot \frac{1}{n},$$

kde *n* je počet hladin, které protne kvantovaný signál.



Obrázek 2-2 Kvantování vzorků signálu

3 Analogově číslicové převodníky

Analogově digitální převodník (zkratky A/D, v angličtině i ADC) je elektronická součástka určená pro převod spojitého (neboli analogového) signálu na signál diskrétní (neboli digitální). Důvodem tohoto převodu je umožnění zpracování původně analogového signálu na digitálních počítačích [6].

Číslicové výstupy převodníků používají různé kódování. Mezi nejběžněji používané kódy patří přímý dvojkový kód, inverzní kód, doplňkový kód, posunutý kód a kód BCD [7].

3.1 Statické a dynamické vlastnosti převodníků

Statické parametry převodníků jsou určovány pomocí převodní charakteristiky, zatím co dynamické vlastnosti se vyhodnocují z kmitočtového spektra převodníku.

3.2 Parametry A-D převodníků

Mezi základní statické parametry patří [7]

- rychlost převodu,
- rozlišení převodníku (resolution),
- přesnost (accuracy),
- chyba nastavení nuly (offset error),
- hystereze a další.

K hlavním dynamickým parametrům patří

- odstup signál-šum (signal to noise ratio - SNR),
- efektivní počet bitů (effective number of bits - ENOB),
- dynamický rozsah bez parazitních složek (spurious free dynamic range - SFDR),
- krátké přechodové špičky (glitches),
- šum - vrcholový, efektivní (noise - RMS, peak),
- doba přepnutí a ustálení a další.

3.2.1 Rychlost vzorkování

Rychlost vzorkování vstupního signálu patří mezi nejvýznamnější parametry A/D převodníků. Musí být dostatečně vysoká vzhledem k nejvyšší kmitočtové složce vstupního analogového napětí – jak již bylo uvedeno, je nutné přenést více jak dva body amplitudy nejvyšší kmitočtové složky sledovaného

signálu. Pokud nás naopak některá vyšší harmonická složka nezajímá nebo způsobuje chybu v následném zpracování dat (např. šumový signál), je možno ji odstranit vhodnou dolní propustí [8].

3.2.2 Rychlost převodu

Rychlost převodu je u A/D převodníků obvykle shodná s rychlostí vzorkování, resp. naopak, rychlost vzorkování vyplývá z nejkratší možné doby převodu. Doba převodu může být určena jako doba, která uplyne od okamžiku přivedení vstupního analogového napětí na vstup převodníku, až do doby, kdy je výstupu převodníku k dispozici platné výstupní datové slovo. Může být rovněž vyjádřena počtem úplných převodů za jednotku času nebo počtem bitů za jednotku času [8].

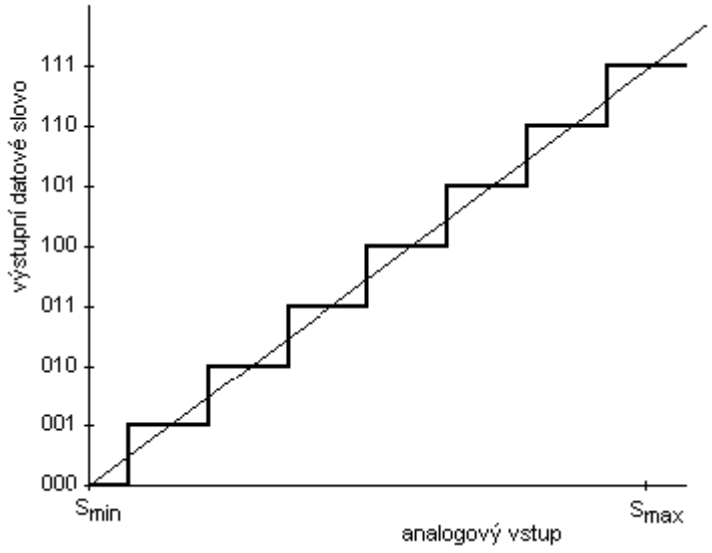
3.2.3 Rozlišení A/D převodníků

Rozlišovací schopnost převodníku AD je určena počtem úrovní, do kterého je rozdělen rozsah vstupního analogového signálu. Jelikož výstupní slovo převodníku vyjadřuje obvykle číslo v binárním kódu, je často rozlišovací schopnost vyjadřována počtem bitů výstupního slova. Sestává-li např. výstupní slovo z 10 bitů, pak vstupní rozsah je rozdělen na 1024 diskrétních úrovní. Rozlišovací schopnost je $1/2^N$, kde N je počet bitů výstupního slova. Je třeba si uvědomit, že čím je větší rozlišovací schopnost, tím je nižší rychlost převodu [8].

3.2.4 Kvantizační chyba převodníků

Vstupní analogový signál, který může nabývat libovolné úrovně v mezích vstupního rozsahu, je tedy kvantován do určitého počtu diskrétních úrovní (kvantizačních úrovní) [7].

Tímto procesem může vzniknout *kvantizační chyba*. Kvantizační chybu lze zmenšit pouze použitím více diskrétních úrovní, tj. použitím více bitů výstupního slova.



Obrázek 3-1 Převodní charakteristika A/D převodníku

Na obr. 3-1 je znázorněna ideální převodní charakteristika A/D převodníku. Schodovitý průběh převodní charakteristiky způsobuje odchylku od ideálního průběhu a projevuje se jako **kvantizační šum SNR** (Signal-to-Noise Ratio). Pro sinusový signál je teoretické SNR dáno vztahem [8]:

$$SRN = 6.02 \cdot n + 1.76[dB],$$

kde *n* je počet bitů datového slova (rozlišení).

Vlivem chyb převodníku je však skutečné SNR odlišné od ideálního, a proto pro porovnání kvality A/D převodníků zavádíme pojem **efektivní počet bitů ENOB** (Effective Number Of Bits)[8]:

$$ENOB = \frac{SNR - 1.76}{6.02} \leq n$$

V praxi se reálná převodní charakteristika liší od ideální vlivem napěťového posunu (označované též *chyba nuly či ofset*), změnou zisku (*chybou rozsahu*) nebo *nelinearitou převodníku*. Celková přesnost převodníku je pochopitelně také podstatně závislá na stabilitě zdroje referenčního napětí [8].

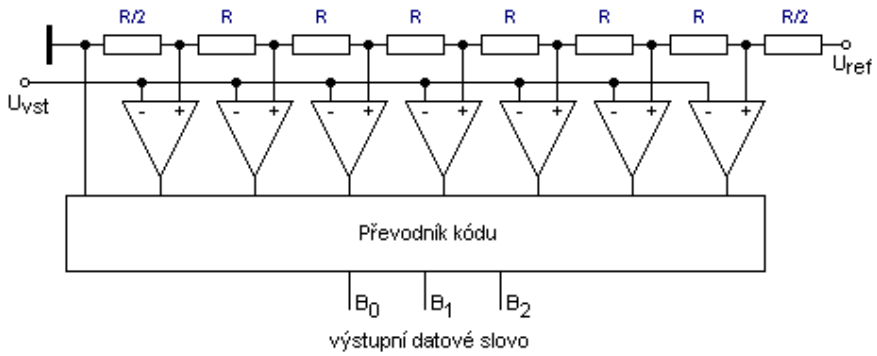
3.3 Typy A/D převodníků

A/D převodníky můžeme dělit podle různých kritérií. Podle způsobu činnosti dělíme převodníky na *synchronní* a *asynchronní*. U synchronních převodníků probíhá převod analogového napětí na výstupní datové slovo v určitém počtu kroků, které se uskutečňují synchronně s hodinovými (taktovacími) impulsy, u asynchronních převodníků může být převod rovněž uskutečněn v několika krocích, ovšem doba trvání těchto kroků závisí výhradně na časové odezvě dílčích obvodů převodníku a na jejich zpoždění [8].

Dále můžeme A/D převodníky dělit podle vstupního signálu na *přímé* a *nepřímé*. Přímé převodníky převádějí přímo vstupní analogové napětí na výstupní slovo, u nepřímých převodníků se vstupní analogové napětí nejprve převádí určitým obvodem na jinou analogovou veličinu (např. na dobu trvání impulsu) a dalším obvodem je teprve tato veličina převedena na výstupní datové slovo [8].

3.3.1 Paralelní A/D převodník

Paralelní A/D převodník je nejrychlejším a současně principiálně nejjednodušším typem přímého A/D převodníku. Princip převodníku je znázorněn na obrázku 3-2. Vstupní analogové napětí je přiváděno současně na vstupy soustavy napěťových komparátorů. Na těchto komparátorech se toto napětí porovnává s určitým referenčním napětím U_{refi} a výstup jednotlivých komparátorů se překlápí v případě, že $U_{vst} \geq U_{refi}$. Převodník kódu pak převede výstupy z napěťových komparátorů na výstupní datové slovo [8].

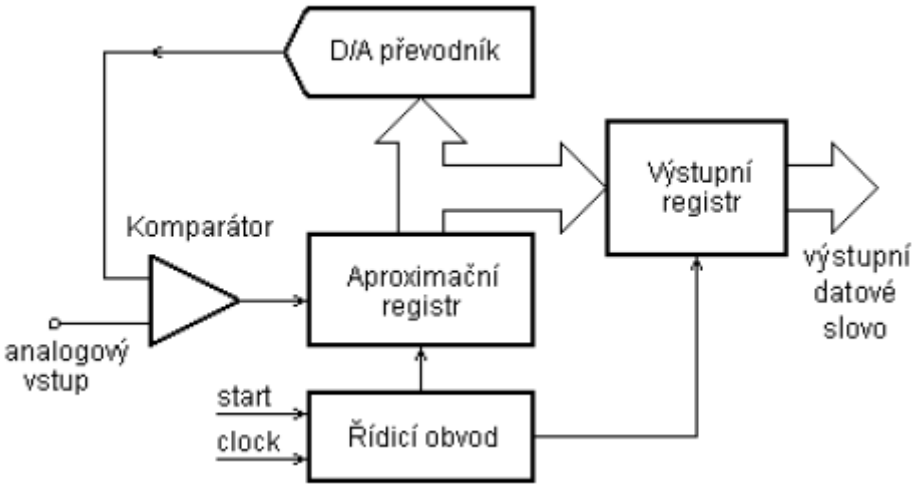


Obrázek 3-2 Tříbitový paralelní A/D převodník

Doba převodu paralelního převodníku je určena přenosovým zpožděním, resp. dobou ustálení napěťových komparátorů a přenosovým zpožděním v převodníku kódu. Převodníky tohoto typu jsou rychlé, ale nákladné, protože obsahují velký počet napěťových komparátorů [8].

3.3.2 Převodník s postupnou aproximací

A/D převodník s postupnou aproximací realizuje převod vstupního analogového napětí na výstupní datové slovo postupně po krocích, jejichž počet je roven počtu bitů výstupního datového slova. Blokové schéma A/D převodníku je na obrázku 3-3 [8].



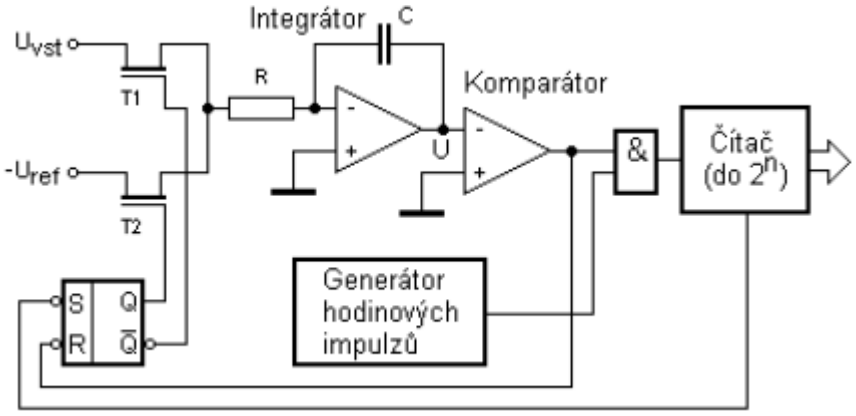
Obrázek 3-3 D/A převodník s postupnou aproximací

Tento převodník má integrovaný D/A převodník, napěťový komparátor, aproximační registr a výstupní registr. Převod se provádí postupně, od nejvyššího bitu směrem k nižším metodou půlení intervalu. Řídicí obvod převodníku nastaví hodnotu testovaného bitu (testované napěťové úrovně) na hodnotu 1, D/A převodníkem je generováno příslušné referenční napětí a napěťový komparátor porovná toto napětí se vstupním napětím. Je-li vstupní napětí větší než referenční, zůstane v příslušném bitu datového slova v aproximačním registru uchována jednička, v opačném případě se na toto místo dosadí nula. Převod pak pokračuje nastavením následujícího (nižšího) bitu datového

slova a stejný postup se opakuje. Nevýhodou převodníku je celková doba převodu, která je přímo úměrná počtu bitů výstupního datového slova [8].

3.3.3 A/D převodník s dvojitou integrací

A/D převodník s dvojitou integrací je příkladem nepřímého převodníku, u kterého je vstupní analogové napětí nejdříve převedeno na dobu trvání určitého elektrického signálu a velikost vstupního napětí je určována podle hodnoty slova v čítači, který je tímto napětím řízen. Schéma zapojení tohoto převodníku je na obrázku 3-4 [8].



Obrázek 3-4 Převodník s dvojitou integrací

Tento převodník je možno charakterizovat poměrně malou rychlostí převodu, značnou dosažitelnou přesností a obvodovou jednoduchostí bez větších nároků na přesnost většiny prvků [8].

3.3.4 A/D převodníky typu sigma-delta

V současné době se velice rozšířily A/D převodníky typu sigma-delta. Jádrem tohoto synchronního převodníku je opět integrátor a komparátor, který generuje sled pulzů, jejichž střední hodnota počtu za určitý interval odpovídá vstupnímu napětí. Střední hodnota se vytváří v číslicovém filtru

Nejdůležitější parametry všech A/D převodníků jsou uvedeny v tabulce 3-1 [8].

Typ	Rozlišení [bit]	Rychlost převodu [Hz]
Paralelní	6 ... 10	10^7 … 3.10^9
Aproximační	8 ... 16	3.10^4 … 3.10^6
Integrační	10... 27	10^{-1} … 10^3
Sigma-delta	16 … 24	10^1 … 10^5

Tabulka 3-1 Parametry A/D převodníků

4 Platforma FITkit

Výuková platforma FITkit byla vyvinuta na Fakultě informačních technologii Vysokého učení technického v Brně. Platforma FITkit umožňuje obsáhnout značnou část spektra znalostí a dovedností, které musí dnešní inženýr – informatik znát, aby byl schopen obstát na globálním trhu práce.

Obrázek 4-1 Platforma FITkit

Typickým příkladem využití informatiky v praxi jsou tzv. vestavěné systémy (anglicky Embedded Systems), které se v dnešní době dominantně uplatňují v běžném životě a jejichž význam ještě výrazně poroste. Jednoduše řečeno se jedná o veškerá zařízení, která v sobě mají nějakým způsobem vestavěn počítač (mobilní telefon, MP3 přehrávač, televizní přijímač atd.).

FITkit obsahuje výkonný mikrokontrolér s nízkým příkonem a řadu periferií. Důležitým aspektem je též využití pokročilého re-programovatelného hardwaru na bázi hradlových polí FPGA (anglicky Filed Programmable Gate Array), jenž lze, podobně jako software na počítači, neomezeně modifikovat pro různé účely dle potřeby – uživatel tedy nemusí vytvářet nový hardware pro každou aplikaci znovu [9].

Při návrhu aplikací se využívá skutečnosti, že vlastnosti hardwaru se v dnešní době převážně popisují vhodným programovacím jazykem (např. VHDL), díky čemuž se návrh softwaru a hardwaru provádí do značné míry obdobně.

Software po mikrokontrolér se tvoří v jazyce C a do spustitelné formy se překládá pomocí GNU překladače. Generování programovacích dat pro FPGA z popisu v jazyce VHDL probíhá zcela automaticky pomocí profesionálních návrhových systémů [9].

Uvnitř mikrokontroléru je umístěn kód, který umožňuje naprogramovat čip přes sériový kanál bez použití dalších externích prostředků a za pomoci volně dostupných programů, anebo lze použít externí programátor, který se napojuje na JTAG (Joint Test Action Group) rozhraní mikrokontroléru.

FPGA lze obdobně naprogramovat přes JTAG anebo za pomoci MCU skrze vysokorychlostní komunikaci protokolem SPI.

FITkit komunikuje s vnějším okolím přes USB rozhraní, ze kterého je přímo napájen. V případě velkého proudového odběru je nutné připojit externí napájení +5V do konektoru JP8. Externí napájení je chráněno pojistkou F0 o hodnotě 1A. Z napětí +5V se dále stabilizátory získává pro většinu komponent napětí +3.3V a pomocná napájení +2.5V a +1.2V pro napájení FPGA[10].

Iniciativa je koncipována jako open-source (pro software) a open-core (pro hardware), což znamená, že veškeré výsledky práce studentů s platformou FITkit jsou přístupné na internetu ve zdrojové formě pro kohokoli. Veškerý návrhový software ke generování programovacích dat pro FPGA, včetně rozsáhlé dokumentace, je k dispozici zdarma na internetových stránkách firmy Xilinx Inc. [11].

4.1 Hlavní části platformy FITkit

Výuková platforma FITkit obsahuje tyto prvky:

- FPGA Spartan 3 XC3S50-4PQ208C (Xilinx)
- MCU MSP430F168 (Texas Instruments)
- USB-UART převodník FT2232C
- Audio IN / OUT
- Konektory PS2
- Konektor VGA
- Konektor RS232
- DRAM 8x8mbit
- Klávesnice
- Řádkový LCD displej

FITkit disponuje dvěma programovatelnými jednotkami, a sice mikroprocesorem MCU řady MSP430 a FPGA řady Spartan3. Obě jednotky je možné programovat pomocí kabelu s rozhraním JTAG. Nicméně MCU lze programovat i přímo přes USB kabel. Pro nahrání konfigurace do FPGA čipu lze použít naprogramovaný MCU. Více o možnostech programování obou jednotek bude popsáno v kapitole 4.4.

4.1.1 FPGA

Na kitu je osazeno programovatelné hradlové pole XC3S50-4PQ208C řady Spartan 3 firmy Xilinx.

Mezi hlavní rysy tohoto čipu patří [9]:

- 192 konfigurovatelných logických bloků (CLBs) uspořádaných do matice o 16 řádcích a 12 sloupcích
- 1728 logických buněk
- 50000 logických hradel
- Paměť RAM
    - 12 Kbitů distribuovaných, 72 Kbitů v jednom bloku
- 2 jednotky pro správu vstupního hodinového signálu (DCMs - Digital Clock Managers)
- 4 násobičky 18x18 bitů

Pro komunikaci s okolím obsahuje čip až 124 uživatelských vstupů/výstupů (I/O), z nichž 45 pinů je sdruženo ve sběrnici X[0..45]. Tato sběrnice je vyvedena přímo na 50 pinový dvouřadý konektor JP10. Konektor se nachází v dolní části kitu. Rozteč jeho pinů je klasických 2.54 mm. Na zbývající piny konektoru je přivedeno i napájecí napětí +3.3V a +5V [4].

FPGA může také komunikovat s MCU pomocí SPI (Serial Peripheral Interface) anebo pomocí společné 8bitové sběrnice, označené jako P3M [10].

4.1.2 MCU

Dalším ústředním prvkem na kitu je mikrokontrolér MSP430F168 firmy Texas Instruments. Jedná se o nízko-příkonový mikrokontrolér s těmito parametry [9]:

- Nízký příkon
    - 330 µA v aktivním režimu při 1 Mhz a 2.2 V
    - 1.1 µA ve stand-by režimu
    - 0.2 µA v režimu vypnuto
- 16-bitová RISC architektura
- Instrukční cyklus 125 ns
- Paměť - Flash 48 KB + 256 B
- RAM 10 KB
- 16-bitové časovače
- Sériové rozhraní
- 12-bitové A/D a D/A převodníky

Vstupně výstupní porty jsou označeny jako P1-P6. Jim přísluší označení sběrnic P1M-P6M. Většina signálů z těchto sběrnic je vyvedena na konektor JP9, což je 40-ti pinový konektor s roztečí 2.54mm. Na tomto konektoru je také vyvedeno napájecí napětí +3.3V a +5V.

Port P5 se používá pro SPI komunikaci mezi FPGA a FLASH pamětí. Dále při programování FPGA a vysílání hodinových signálů SMCLK a ACLK.

Programování MCU lze provést dvěma způsoby. První přes JTAG rozhraní, připojené přes konektor JP11 nebo přes USB/UART, za použití rutiny označené jako Bootloader. Bootloader je krátký programový kód, který umožňuje přijímat data přes sériovou linku a zapisovat je do vnitřní FLASH paměti. Tento kód je výrobcem napevno uložen v obvodu MCU [10].

Na sběrnici P6M je vyvedeno 8 kanálů 12-ti bitových A/D převodníků a dva 12-ti bitové D/A převodníky. Parametry vestavěných A/D převodníků však nejsou pro tento projekt vhodné, proto je zapotřebí k FITkitu připojit externí modul s výkonnějšími převodníky.

4.2 Komunikace s FITkitem

Hlavním vnějším komunikačním prostředkem kitu je univerzální vysoko-rychlostní asynchronní přijímač a vysílač (UART) FT2232C od firmy FTDI Chip[12]. Tento čip převádí rozhraní USB ze strany PC na dva nezávislé konfigurovatelné UART/FIFO kanály.

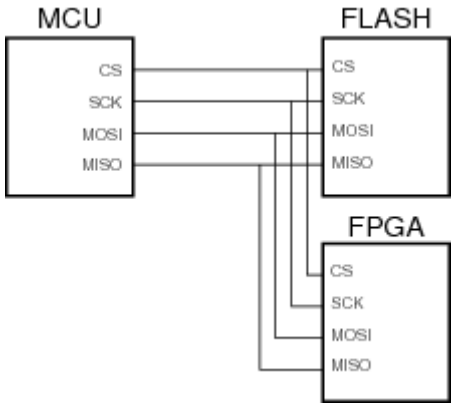
K PC se FITkit připojuje pomocí USB kabelu typu A-B. Po připojení a nainstalování potřebných ovladačů se oba komunikační kanály jeví jako nové sériové porty COM [10]. Jeden z těchto asynchronních kanálů je připojen k FPGA a druhý k mikrokontroléru. Díky této skutečnosti degraduje komunikace s prvky FITkitu na komunikaci po sériové lince. Oba komunikační kanály jsou schopny pracovat rychlostí až 921600 baudů/sec.

Pro komunikaci s mikrokontrolérem je vhodné využít terminálový program, který lze využít jak k ovládání spuštěných aplikací, tak k přenášení binárních dat prostřednictvím protokolu X-modem. Tímto způsobem lze nahrát i konfiguraci do FPGA čipu, o tomto způsobu však bude napsáno v kapitole 4.4. Komunikace probíhá rychlostí až 460800 baudů/sec.

4.3 Sběrnice SPI

Rozhraní SPI (Synchronous Peripheral interface) je sériové vysokorychlostní rozhraní typu master-slave. Rozhraní umožňuje obousměrnou komunikaci řízenou tzv. master (nadřazeným) zařízením s libovolným počtem slave zařízení, které jsou k této sběrnici připojeny [9].

SPI na fitkitu umožňuje procesoru, který je nakonfigurován v master režimu, komunikovat jednak s pamětí FLASH, uchovávající program pro MCU, a konfiguraci FPGA, ale také s periferiemi realizovanými uvnitř FPGA. Typicky se např. SPI používá ke komunikaci s řadiči periferií, pamětí RAM, apod. Slave zařízení FLASH a FPGA sdílejí veškeré signály (Obrázek 4-2) a rozlišení adresovaného zařízení se provádí na úrovní přenášených bitů. K zajištění bezchybné komunikace, musí mít FPGA i FLASH vzájemně disjunktní množinu operačních kódů [9].



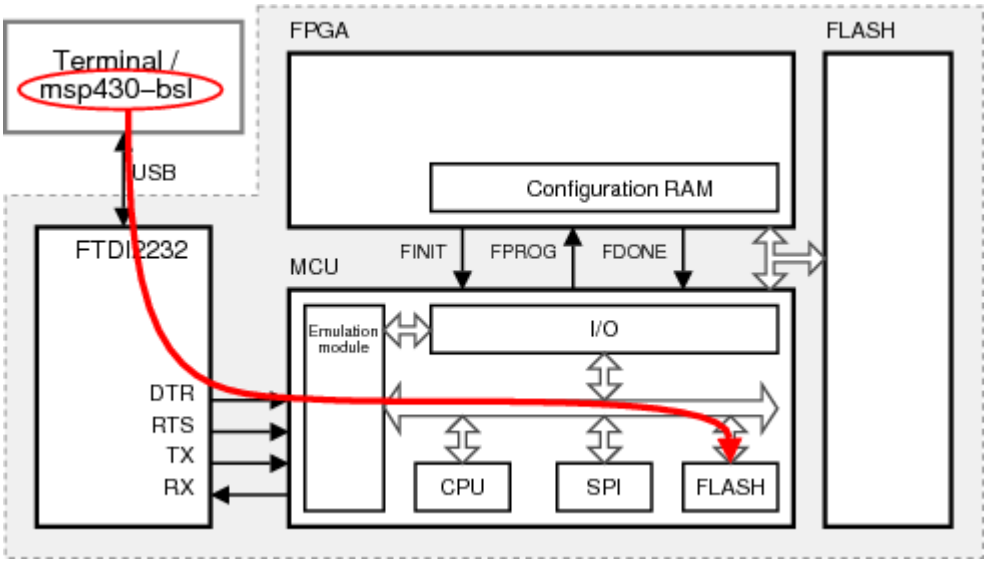
Obrázek 4-2 SPI na FITkitu

4.4 Způsob programování na FITkitu

4.4.1 Programování MCU

Naprogramování mikrokontroléru lze provést dvěma způsoby. Prvním způsobem je naprogramování MCU prostřednictvím rozhraní JTAG. K tomuto účelu lze využít speciální programátor MSP-FET430 firmy Texas Instruments [13]. Výhoda tohoto řešení spočívá v možnosti ladit kód za běhu procesoru. Vhodným vývojovým prostředím pro tento typ programování je například IAR, jehož omezenou distribuci je možné volně stáhnout na stránkách společnosti [13].

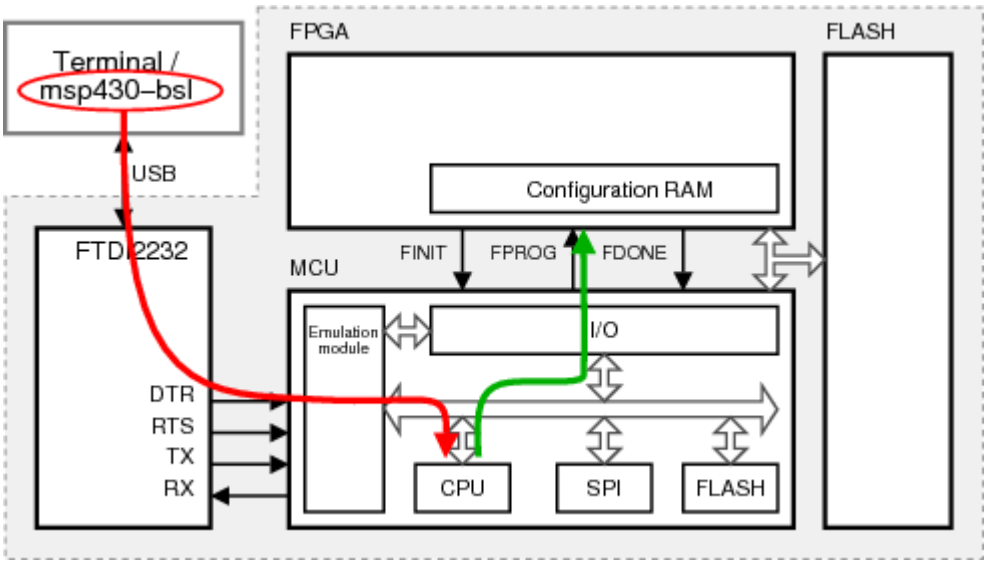
Druhý způsob využívá k naprogramování mikrokontroléru sériové rozhraní. Takovéto programování umožňuje tzv. „bootloader“, což je program umístěný výrobcem v mikroprocesoru. Bootloader je schopný nahrát program mikroprocesoru do jeho FLASH paměti. Nevýhodou této možnosti je, že není možné kód programu za běhu ladit nebo modifikovat, jako tomu bylo v případě programování přes rozhraní JTAG. K programování po sériové lince je možné využít konzolové aplikace msp430gdb-debug, která je v plné verzi volně dostupná v programovém balíku vývojového prostředí mspgcc[14]. FLASH paměť mikrokontroléru, kde je uložen programový kód, zůstává perzistentní i po odpojení napájecího napětí.



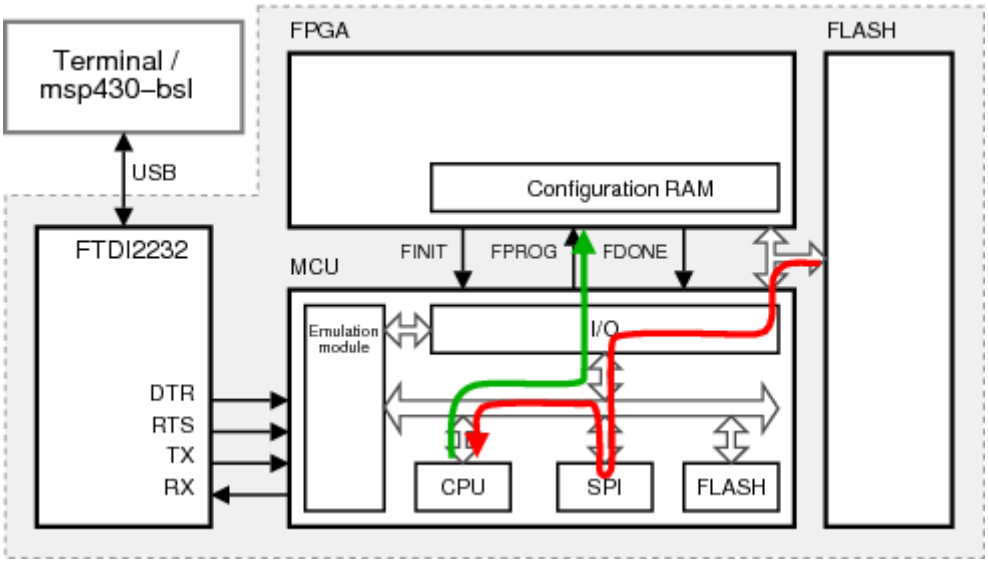
Obrázek 4-3 Programování MCU

4.4.2 Programování FPGA

Konfiguraci pro FPGA je narozdíl od MCU nutné po každém odpojení napájení přeprogramovat. Programování FPGA lze provádět buď podobně jako MCU pomocí rozhraní JTAG, nebo prostřednictvím rozhraní SPI. Druhý způsob programování zajišťuje mikrokontrolér. Ten může novou konfiguraci pro FPGA nahrát přes USB rozhraní přímo z počítače nebo ji nahraje z paměti FLASH umístěné na kitu. Pro přenos konfiguračního souboru z PC se používá terminál pro sériovou komunikaci pomocí protokolu XModem v režimu 1KCRC.



Obrázek 4-4 Nahrání konfigurace FPGA přímo do FPGA



Obrázek 4-5 Nahrání konfigurace FPGA z externí FLASH paměti

4.4.3 Překladový systém FITkitu

Pro usnadnění překladu, syntézy a programování byl v projektu FITkit vytvořen jednoduchý překladový systém, který využívá standardní GNU Makefile soubory. Protože byl kladen důraz na jednoduché použití, obsahují Makefile soubory (jenž jsou součástí projektu) pouze nezbytné informace potřebné k překladu projektu. Tzn. seznam všech zdrojových souborů a knihoven, které projekt pro překlad potřebuje. Soubor překladových pravidel (společný pro všechny projekty) je umístěn v adresáři *base*. Makefile u jednotlivých projektů se na tento soubor odkazují. Parametry, které se mohou lišit v závislosti na konfiguraci systému, jsou umístěny v souboru *base/settings.inc* [9].

Překladový systém pro překlad softwaru nabízí několik příkazů:

- **gmake** – pro kompilaci kódu. Výstupem je soubor (.hex), který je možné použít k naprogramování MCU. Dalším výstupním souborem je mapa paměti (přípona .map)
- **gmake lst** – vytvoření výpisu assembleru. (Soubor s příponou .lst)
- **gmake load** – slouží k naprogramování procesoru binárním souborem s příponou .hex. Pokud tento soubor dosud neexistuje, vytvoří se.
- **gmake clean** – odstraní všechny soubory vzniklé při kompilaci softwaru.
- **gmake cleanlib** - odstraní knihovnu funkcí libfitkit a veškeré soubory vzniklé během její kompilace. Tento příkaz je nutné použít vždy, když provedeme změnu v souborech knihovny.

Podobná struktura jako pro překlad softwaru je použita i pro syntézu VHDL kódu. Soubory Makefile pro syntézu VHDL kódů obsahují všechny potřebné zdrojové soubory většinou v podobě balíčků (*.inc* soubory). Tyto balíčky zapouzdřují cesty k jednotlivým VHDL souborům. Výhodou je

možnost dynamicky měnit seznam zdrojových souborů bez nutnosti zasahovat do již napsaných aplikací, které tyto soubory využívají.

Seznam příkazů týkajících se syntézy hardware [9]:

- **gmake** – vyvolá syntézu zdrojových kódů a vygeneruje binární soubor „output.bin“ obsahující konfiguraci, kterou je možné naprogramovat FPGA obvod.
- **gmake synth** – provede se pouze syntéza zdrojových kódů. Další fáze k vytvoření konfigurace se již neprovedou. Tato možnost je vhodná při ladění VHDL kódů.
- **gmake simmodel** – vygeneruje simulační model „post place&route“, který se používá pro časovou simulaci.
- **gmake rtl** – vygeneruje RTL schéma s příponou *.ngr*
- **gmake sim** – zkompiluje všechny zdrojové soubory, vytvoří simulační knihovnu *sim/work* a spustí skript *sim.fdo* v adresáři *sim.*
- **gmake cleam** – odstraní všechny soubory vzniklé při syntéze a generování konfigurace.

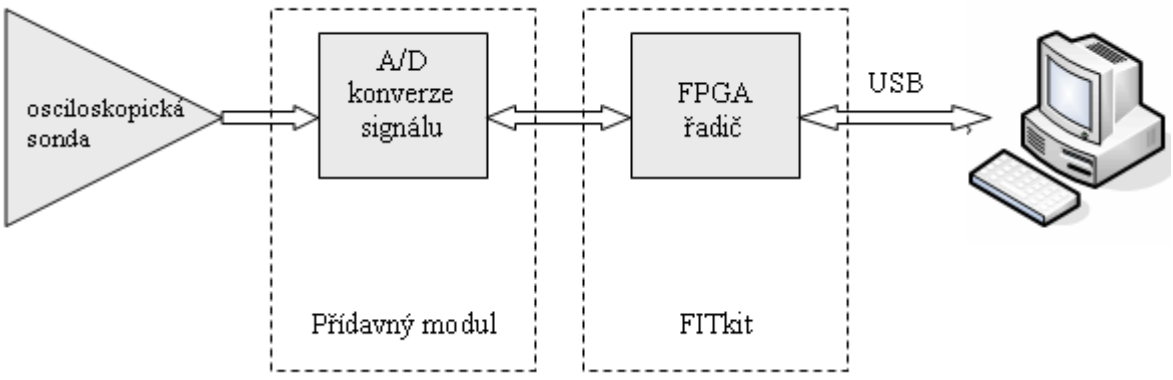
5 Koncepce řešení

5.1 Blokový popis

Celou práci je možno rozdělit do několika částí. Prvním problémem je vlastní analogově/digitální konverze vstupního napětí na číselnou reprezentaci vzorků zpracovatelných na FITkitu. K tomu je zapotřebí využít vhodných A/D převodníků. Převodníky integrované v mikrokontroléru na kitu jsou určeny především k vzorkování audio signálu a jejich parametry jsou tudíž nevhodné pro zpracování vysokofrekvenčního signálu s větším rozptylem hodnot napětí. Z toho důvodu je třeba navrhnout modul osazený rychlými A/D převodníky, který bude schopný komunikovat s čipem FPGA na FITkitu. Komunikace bude probíhat po sběrnici X čipu FPGA, která je vyvedena do spodního konektoru JP10 na FITkitu.

Další část projektu spočívá v navržení FPGA řadiče, který bude v pravidelných intervalech spouštět A/D převod. Získané hodnoty bude následně ukládat do RAM paměti na čipu. V okamžiku, kdy navzorkuje dostatečné množství dat, se vzorkování ukončí a hodnoty uložené v paměti RAM se přesunou přes USB rozhraní do PC. FPGA čip je pro toto použití velice vhodný zejména pro svoji schopnost velice rychlého čtení a ukládání dat do vestavěných pamětí RAM.

Poslední částí projektu je aplikace s grafickým rozhraním, běžící na PC. Tato aplikace umožní uživateli nastavit různé parametry vzorkování. V okamžiku stisknutí startovacího tlačítka začne FPGA řadič vzorkovat vstupní signál. Po navzorkování dostatečného množství dat a jejich přenesení do PC se na monitoru počítače v podobě grafu zobrazí průběh měřeného signálu. Blokové schéma celého návrhu je vyobrazeno na následujícím obrázku 5-1.



Obrázek 5-1 Blokový popis řešení

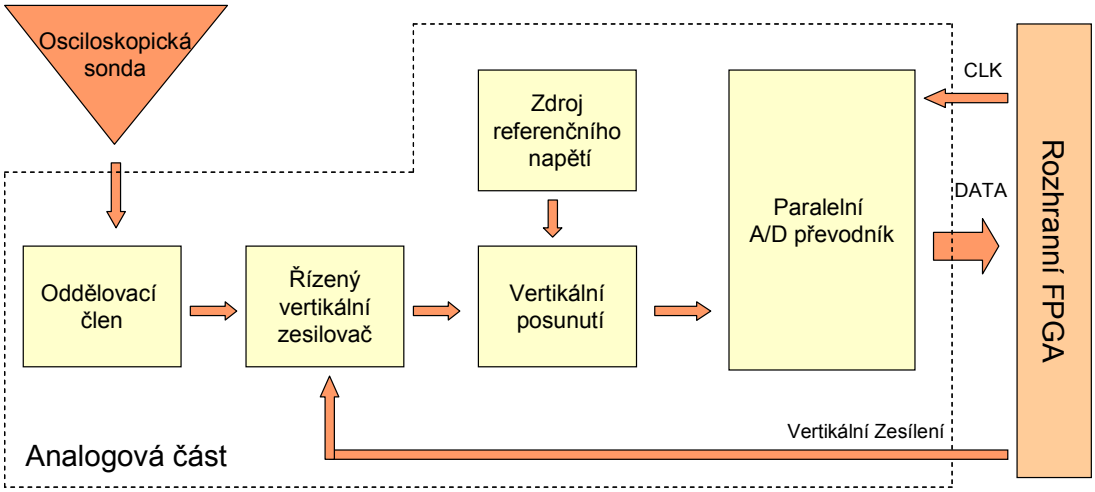
5.2 Obecný návrh analogové části

Nejdříve bylo nutné stanovit si parametry, které má osciloskop, resp. jeho analogová část splňovat.

Mezi základní požadavky jsem při návrhu stanovil:

- vysokou rychlost vzorkování (alespoň 20MHz),
- široký vstupní napěťový rozsah (± 20V),
- možnost měření i záporných hodnot napětí,
- možnost elektronicky měnit vertikální napěťové rozsahy,
- možnost napájení modulu přímo z FITkitu,
- odolnost proti rušení

Celou analogovou část jsem si poté rozčlenil do několika logických celků. Tyto celky jsem následně analyzoval a navrhnul konkrétní řešení. Blokové schéma analogové části je znázorněno na následujícím obrázku 5-2.



Obrázek 5-2 Blokové schéma analogové části

Prvním členem analogové části je pasivní osciloskopická sonda, pomocí které je měřený signál přiváděn na vstup osciloskopického modulu.

Vstupní signál je bezesporu nutné vhodným způsobem upravit. Prvním prvkem na vstupu modulu je oddělovací člen, který bude splňovat dvě funkce:

1. Omezí vstupní napětí na požadovaný napěťový rozsah.
2. Oddělí vstupní část modulu.

Za oddělovací částí následuje část vertikálního zesilovače, který je schopen řízeně upravovat vstupní napětí a měnit tak napěťové rozsahy.

Další částí modulu je blok vertikálního posunutí, který k upravenému vstupnímu signálu přičte napěťovou složku přivedenou z bloku referenčního napětí. Tato část je důležitá pro to, že A/D

převodník dokáže měřit pouze v omezeném kladném rozsahu hodnot napětí. Signály potřebné k přepínání rozsahů je potřeba vyvést na výstupní konektor, aby je bylo možné ovládat z řídicího FPGA řadiče.

Posledním blokem je samotný 8 bitový A/D převodník, jehož paralelní výstupy budou vyvedeny na výstupní konektor na modulu. Kromě paralelních výstupů převodníku je třeba na sběrnici vyvést ještě signál CLK, který bude sloužit k nastartování procesu konverze analogového vzorku na digitální.

Všechny datové, řídicí i napájecí vodiče budou vyvedeny na společný dvouřadý 50-ti pinový konektor tak, aby byl modul snadno připojitelný k platformě FITkit.

5.3 FPGA řadič - požadavky

Ústředním členem celého projektu je FPGA řadič implementovaný v programovatelném hradlovém poli osazeném na platformě FITkit.
Úloha FPGA řadiče je následující:

- Přečtení bloku dat z PC, kde bude obsažena informace o nastavení vzorkování. (Horizontální rozsah = frekvence vzorkování a vertikální rozsah)
- Řízení A/D převodu, (tzn. v pravidelných intervalech vybudit převodník vzestupnou hranou signálu CLK)
- Řízení vertikálního zesílení
- Dostatečně rychlé ukládání dostatečného počtu vzorků do paměti RAM
- Odesílání naměřených dat zpět do PC

Hlavním komunikačním prostředkem FITkitu s vnějším prostředím (např. s PC) je univerzální vysokorychlostní asynchronní přijímač-vysílač (UART) FT2232C firmy FTDI Chip [12], který je možné nakonfigurovat do různých módů komunikace:

- UART
- Asynchronní/synchronní paralelní port
- MPSSE (Multi Protocol Synchronous Serial Engine)
- MCU Host bus Emulation
- a další

5.4 Grafická aplikace na PC

Uživatelské prostředí osciloskopu bude řešeno formou aplikace s GUI implementované na PC.

Nejdůležitější částí aplikace bude zajisté graf, na kterém se bude zobrazovat průběh měřeného vstupního napětí v závislosti na čase. Díky skutečnosti, že o horizontální složku průběhu (čas) se postará výše uvedený FPGA řadič, není třeba zaznamenávat časovou složku vzorků. Aplikace pouze zobrazí na každém obrazovém bodu právě jeden naměřený vzorek. Pokud si vhodně zvolím výšku okna pro zobrazování vzorků (nejlépe 256 pixelů) nebude třeba ani nijak přepočítávat vertikální složku naměřených dat, ale bude možné přímo vykreslovat výstup z osmi-bitového převodníku na plochu vykreslovaného okna.

Dalšími nezbytnými prvky aplikace budou komponenty pro nastavení parametrů vzorkování. To je vzorkovací frekvence a vertikální zesílení signálu na vstupu.

Tlačítkem „SAMPLE“ vyvoláme proces vzorkování signálu. Nejprve bude potřeba odeslat zvolené parametry vzorkování a poté již očekávat naměřené vzorky.

Díky čipu FT2232C od firmy FTDI Chip[12] osazeném na FITkitu degraduje komunikace s kitem na komunikaci po sériovém kanálu. Téměř pro každý programovací jazyk jsou dostupné knihovny pro práci se sériovým portem. Součástí aplikace tedy musí být i formulář, který umožní nastavení komunikačního portu.

Mezi rozšiřující funkce programu zařadím:

- možnost pohybu po časové ose tak, aby si mohl uživatel prohlédnout každý detail navzorkovaného signálu
- možnost měření amplitudy, periody a frekvence vstupního signálu pomocí kurzorových úseček
- možnost vyexportování grafu do obrazového souboru

6 Realizace analogové části

Cílem této kapitoly je detailně rozebrat každou část navržené analogové části. Vrcholem kapitoly je kompletně realizovaná a osazená deska plošných spojů splňující všechny stanovené požadavky.

Při rozboru jednotlivých částí budu vycházet z blokového schéma analogové části v kapitole 5 (obr. 5-2). Vzhledem k faktu, že nejsem příliš zkušený v návrhu analogových, vysokofrekvenčních obvodů, snažil jsem se při realizaci vyjít z nějakého již hotového zařízení. Po jisté době jsem v praktické elektronice objevil článek [17], který mi návrh osciloskopického modulu velice usnadnil.

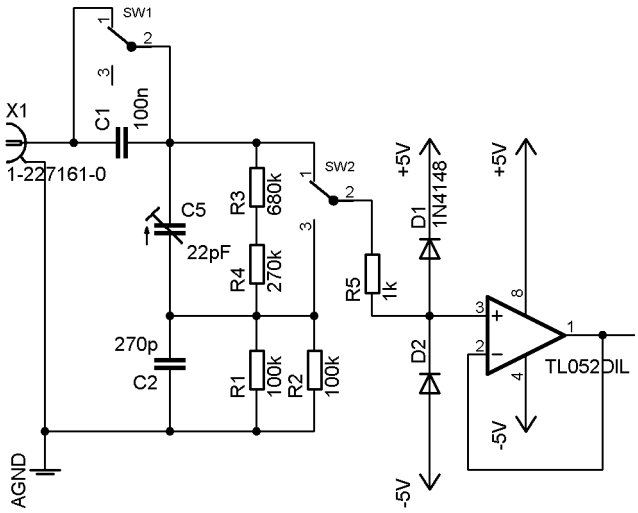
6.1 Oddělovací člen

Na vstupu analogové části se nachází oddělovací člen, jehož součástí je i BNC konektor pro připojení osciloskopické sondy. Jedná standardní 50Ω BNC konektor v provedení W, který je možné zapájet do desky plošných spojů (dále jen DPS). Plášť konektoru je připojen k analogové zemi (viz. kapitola 6-7), signálový vodič pokračuje dále přes kapacitu oddělující stejnosměrnou složku od střídavé. Tuto kapacitu je možné pomocí přepínače SW1 vyřadit.

Aby bylo možné měřit velké napěťové rozsahy, zařadil jsem za kondenzátor vstupní napěťový dělič, který je zapojen v poměru 1:10 a umožňuje měřit napětí v rozsahu až ± 20 V. Pro přesnější měření v rozsahu ± 2V je však možné dělič odpojit pomocí druhého přepínače SW2. Dělič by bylo možné odpojovat i pomocí relé. To jsem však kvůli vysoké spotřebě a požadavku napájení z kitu zamítnul. Protože odporový dělič obecně při vysokých kmitočtech „šumí“ bylo nutné jej kapacitně vykompenzovat. K tomuto účelu slouží dvojice kondenzátorů, z nichž jedna je nastavitelná.

Za děličem následuje poslední prvek oddělovacího členu, kterým je operační zesilovač TL052 splňující funkci impedančního převodníku. Tento operační zesilovač disponuje velkou rychlostí a je běžně dostupný v prodejnách s elektrotechnikou. Připojené diody plní funkci oříznutí hodnot napětí, které se vyskytují mimo měřitelný napěťový rozsah.

Nejvíce vypovídající je schéma zapojení oddělovacího členu, které je znázorněné na níže uvedeném obrázku 6-1.



Obrázek 6-1 Oddělovací člen

6.2 Řízený vertikální zesilovač

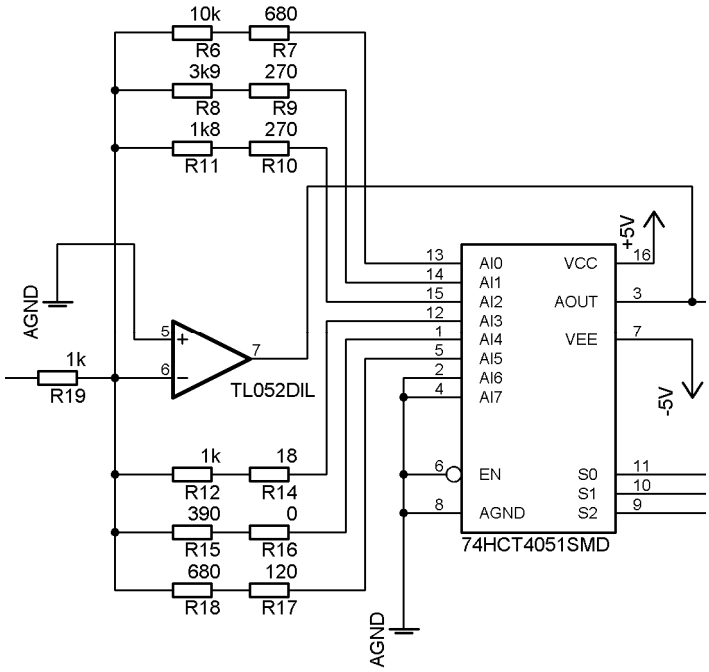
Za oddělovacím členem je zapojen blok s řízeným vertikálním zesilovačem. Tento prvek na základě vstupní adresy nastaví zesílení vstupního signálu na požadovanou hodnotu tak, aby jej mohl dále zpracovat A/D převodník.

Funkci vertikálního zesilovače tvoří operační zesilovač TL052 zapojený v invertujícím zapojení spolu s analogovým multiplexorem 74HCT4051.

Jedná se o osmikanálový analogový multiplexor/demultiplexor se třemi vstupy adresy, povolovacím vstupem EN aktivním v nízké napěťové úrovni, osmi nezávislými vstupy/výstupy (AI0 – AI7) a jedním společným vstupem/výstupem (AOUT). Vzhledem k potřebě zpracovávat i záporné napětí, musí být jak operační zesilovač, tak multiplexor napájen ± 5V. Povolovací vstup jsem zapojil „napevno“ k analogové zemi.

Aby mohl operační zesilovač zesilovat signál na různé úrovně, musí mít ve zpětné vazbě zapojeno více větví s rezistory. Podle adresy analogový multiplexor vybere právě jednu cestu a signál přivede zpět na invertující vstup operačního zesilovače. Zapojení obsahuje celkem 6 zpětnovazebních větví, z nichž 3 větve vstupní signál zesilují a 3 jej zeslabují. Pro rozlišení těchto úrovní využijeme všechny tři adresové vodiče multiplexoru. Hodnoty rezistorů jsou zvoleny tak, aby umožnily nastavit v aplikaci dostatečný rozptyl měřených hodnot. V každé větvi jsou potom zapojeny dva aby z nich bylo možné složit i hodnoty, které se běžně nevyrábí.

Na obrázku 6-2 je znázorněno schéma zapojení vertikálního zesilovače.



Obrázek 6-2 Řízený vertikální zesilovač

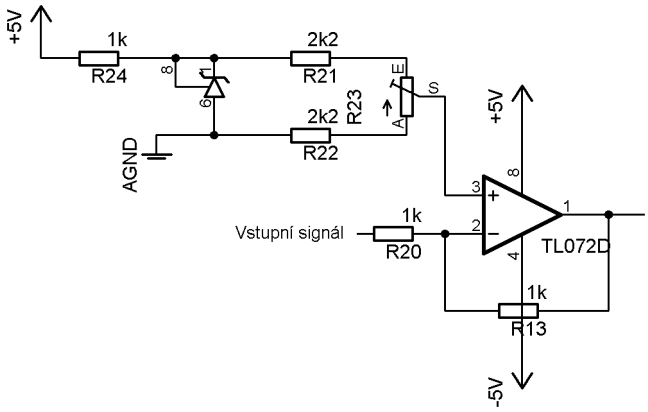
6.3 Blok vertikálního posunutí

Za blokem vertikálního zesilovače je připojen blok vertikálního posunutí. Protože použitý převodník zpracovává napětí v rozsahu 1.55 až 3.26 V [17], je nutné signál nakonec stejnosměrně posunout tak, aby při jeho nulové hodnotě (zkratovaný vstup) bylo na vstupu A/D převodníku napětí 2.4V, což je střed tohoto rozsahu.

K tomuto účelu slouží operační zesilovač TL072. Je opět zapojený v invertujícím zapojení, tím pádem nám otáčí fázi signálu zpět, tak jako byl na vstupu osciloskopického modulu. Stejnosměrný ofset (2.4V) je přiveden na neinvertující vstup operačního zesilovače ze zdroje referenčního napětí. Takto upravený signál je již možné přivést na vstup A/D převodníku.

6.4 Zdroj referenčního napětí

Jako zdroj referenčního napětí pro vertikální posunutí signálu slouží napěťová reference TL431 spolu s odporovým děličem a potenciometrem k přesnému doladění. Schéma zapojení zdroje referenčního napětí a bloku vertikálního posunutí je patrné na obrázku 6-3.



Obrázek 6-3 Vertikální posun signálu o referenční napětí

6.5 Analogově digitální převodník

K realizaci jsem zvolil osmi-bitový paralelní A/D převodník TDA8703 od firmy Philips Semiconductors [15].

Tento převodník je schopen navzorkovat až 40 milionů vzorků ze jednu sekundu. Přesto, že byl původně určen ke zpracování videosignálu, je tento převodník využíván v řadě amatérských zapojení zejména pro svoji dostupnost a nízkou cenu. Díky osmi paralelním třístavovým TTL výstupům může připojený FGPA řadič velice rychle číst naměřené vzorky. Další výhodou převodníku je fakt, že k němu není potřeba připojovat žádný obvod typu „sample-and-hold“, protože převodník na svých výstupech „podrží“ naměřenou hodnotu do doby, kdy je k dispozici další vzorek. Neméně důležitá je otázka spotřeby. Příkon použitého převodníku se pohybuje typicky kolem 290mW [15].

Převodník jsem zapojil podle katalogového zapojení v dokumentaci produktu [15]. Napájení převodníku je specifické díky vyžadovanému oddělenému napájení analogové části a digitální části. O problému s napájením bude blíže napsáno v následující části kapitoly. Měřený signál je přiveden na vstup převodníku označený jako *VI*. Ke spuštění měření signálu slouží vstup *CLK*. Každou náběžnou hranou na tomto vstupu je provedena konverze analogového signálu na digitální. Po dokončení převodu je hodnota k dispozici na osmi paralelních výstupech (D0 – D7). Převodník očekává na hodinovém vstupu *CLK* TTL signál. Protože však převodník chápe napětí na hodinovém vstupu větší než 2V jako „logickou 1“, můžu jej přímo připojit k výstupnímu konektoru a ovládat jej výstupem FPGA.

Problém ale nastává u výstupů převodníku. Jedná se o 5V TTL výstupy, které nemohou být připojeny přímo k FPGA. To totiž dokáže zpracovat pouze 3.3V logiku. Z tohoto důvodu je nutné mezi výstupy a konektor připojit ještě převodník napěťových úrovní.

Dobrou volbou se prokázal obvod SN74CB3T3245 od firmy Texas Instruments [16]. Jedná se o nízko-příkonový osmibitový převodník napěťových úrovní, který dokáže s minimálním zpožděním

(asi 0.25ns) převést 8 TTL výstupů z A/D převodníku na výstupy v 3.3 V logice. Tyto výstupy již je možné přímo připojit ke konektoru potažmo k FPGA obvodu.

6.6 Vstupně/výstupní konektor

Možnost připojení osciloskopického modulu zajišťuje dutinková lišta. Lišta čítá celkem 50 pinů ve dvouřadém provedení, díky čemuž lze jednoduše zasunout do konektoru JP10 na FITKitu.

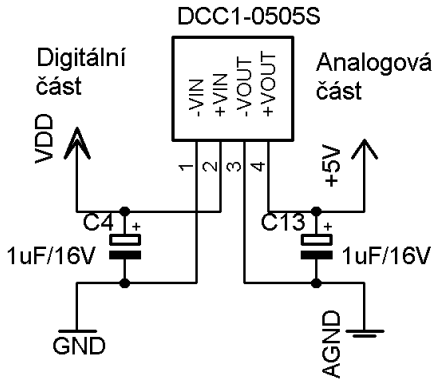
Výstupní 8-bitová datová sběrnice převodníku je připojena na piny 5 až 12. Pin č. 19 slouží k rozvodu hodin převodníku (Signál CLK). Prostřednictvím pinů 25, 26 a 27 je přivedena adresa pro výběr vertikálního zesílení. Vzhledem k faktu, že jak A/D převodník tak analogový multiplexor jsou vyrobeny technologií CMOS, je možné k jejich vstupům připojit i logiku 3.3V, přestože jejich napájecí napětí je 5V. CMOS obvody totiž „posoudí“ napětí 2V a vyšší, jako logickou úroveň „1 (High)“. Proto bylo možné připojit adresu multipexoru i hodinový signál CLK přímo ke konektoru bez nutnosti konverze 5V logiky. Výběr čísel pinů na konektoru jsem provedl až při navrhování DPS s ohledem na rozmístění všech součástek.

6.7 Napájení analogové části

Jedním z důležitých kritérií při návrhu osciloskopického modulu bylo napájení přímo z FITkitu, tedy bez nutnosti připojení externího napaječe.

Při této koncepci jsem se však potýkal se dvěma klíčovými problémy napájení. Jedním z nich je fakt, že použitý převodník vyžaduje oddělené napájení analogové a digitální části. Druhým problémem je potřeba záporného napětí pro napájení operačních zesilovačů. Záporné napájení zesilovače mi totiž dovoluje měřit záporná napětí vstupního signálu.

Pin konektoru č.1 je připojen na napětí 5V, nacházející se na kitu. Tento pin mi posloužil k napájení digitální části osciloskopického modulu. K oddělení napájecí větve analogových obvodů jsem použil obvod QDC1S-0505S firmy QLT POWER/AIMTEC [19]. Tento obvod plní funkci napěťového měniče 5V-5V a odděluje tak větev pro napájení analogových obvodů. Jeho zapojení je zobrazeno na obr. 6-5.

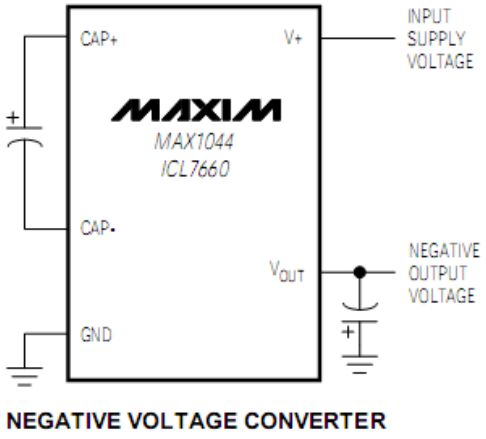


Obrázek 6-4 Zapojení napěťového měniče

Zdánlivě větším problém se zdála být zmiňovaná potřeba záporného napětí pro napájení operačních zesilovačů. Protože spotřeba proudu v záporné napájecí větvi je minimální, použil jsem pro „výrobu“ záporného napětí obvod ICL7660 firmy MAXIM [20]. ICL7660 je monolitický CMOS konvertor napětí, který pracuje na principu nábojové pumpy (podobně jako MAX232 pro konverzi TTL <=> RS232) . Po připojení kapacity a vstupního napájení +5V začne obvod generovat záporné napětí -5V. Katalogové zapojení tohoto obvodu je vyobrazeno na obrázku 6-6.

Napájení 3.3V pro obvod SN74CB3T3245, což je již zmiňovaný převodník napěťových úrovní, jsem vyvedl z pinu č. 2 na konektoru. „Digitální zem“ je vedena z pinů 3 a 4.

Kompletní schéma zapojení osciloskopického modulu je přiloženo v příloze B.



Obrázek 6-5 Katalogové zapojení ICL7660

6.8 Návrh a výroba desky plošných spojů

Po dokončení návrhu schématu zapojení osciloskopického modulu bylo nutné vybrat vhodná pouzdra všech použitých součástek a navrhnout desku plošných spojů.

Pro návrh DPS jsem se rozhodl použít nástroj EAGLE od firmy CadSoft Computer GmbH [21]. Jeho volně šířitelná verze EAGLE Light, která je k dispozici na stránkách výrobce, plně postačuje mým požadavkům. Mezi její největší omezení patří omezení velikosti navrhované desky na 100 x 80 mm. Tento rozměr se však pro moji desku nakonec ukázal jako dostačující.

Před vlastním kreslením schématu v prostředí EAGLE bylo nutné vytvořit si pouzdra a symboly všech použitých součástek. K tomuto účelu jsem si vytvořil novou knihovnu *FITKit.lbr*, ve které jsem sjednotil všechny použité součástky, jejichž seznam se nachází na přiloženém CD. Běžné součástky jako jsou rezistory a kondenzátory již byli vytvořeny v knihovnách dodávaných se softwarem. U těchto diskrétních součástek jsem volil pouzdra SMD o velikost 805. Kromě použitého A/D převodníku a operačního zesilovače TL052 jsem všechny součástky volil v SMD pouzdrech. Zmiňované dva obvody se mi podařilo koupit pouze v provedení THD.

Po dokončení schématu zapojení jsem započal návrh vlastní desky. Nejdříve bylo nutné vhodně uspořádat všechny součástky na desku tak, aby se jednak „gumové“ spoje EAGLU co nejméně křížily a také s ohledem na funkčnost zapojení. To znamená použití co nejkratších vodivých cest u vysokofrekvenčních signálů. Blokovací kondenzátory pro odstranění šumu napájecích větví bylo nutné umístit do těsné blízkosti napájecích vývodů všech integrovaných obvodů.

Pro „zaroutování“ všech spojů jsem využil dvě signálové vrstvy. Pro signály jsem použil šířku spoje o velikosti 0.4 mm, pro napájecí vodiče až 0.8 mm. Celý návrh signálových vrstev jsem završil zastíněním analogové i digitální části pomocí příslušných signálových zemí.

Následující seznam tvoří výčet všech použitých/vytvořených vrstev.

- *1 top, 17 Pads, 18 Vias* – Horní strana součástek (horní signálová vrstva)
- *16 bottom, 17 Pads, 18 Vias* – Spodní strana součástek (Spodní signálová vrsva)
- *29 tStop* - Maska součástek
- *30 bStop* - Maska spojů
- *20 Dimensions* – Obrys (velikost) DPS
- *44 Drils, 45 Holes* – Vrtání prokovené, neprokovené
- *101 tPrint* – Potisk horní strany součástek (TOP Layer)
- *102 bPrint* – Potisk strany spodní strany spojů (BOTTOM Layer)

Zakázku s doplňujícími parametry (konstrukční třída ...) jsme s Ing. Šimkem zadali firmě Gatema s r.o., která asi do deseti dnů doručila vyrobenou desku plošných spojů na fakultu.

Návrh DPS a fotografie vyrobené desky se nachází v příloze.

6.9 Osazení desky plošných spojů

Osazení desky jsem prováděl svépomocí v laboratořích L305 a L203 fakulty Informačních technologií. Zároveň bych chtěl poděkovat všem osobám, které mi umožnily trávit v laboratořích více času, než je zdrávo.

Jednotlivé součástky jsem pájel tamější páječkou od diskrétních SMD odporů a kondenzátorů přes SMD integrované obvody až po větší THD obvody a konektor.

Po zapájení všech součástek jsem celou desku umyl v lihové lázni, abych odstranil přebytečnou pájecí kapalinu. Fotografie osazeného osciloskopického modulu se nachází v příloze.

6.10 Oživení osciloskopického modulu

Před připojením osazeného modulu k FITkitu jsem proměřil všechny spoje. Ověřil jsem si tak, zda některé z nich nejsou přerušené či zkratované.

Jakmile se ukázalo, že jsou všechny spoje v pořádku, připojil jsem modul k napájecímu napětí +5V, abych ověřil spotřebu modulu a proměřil všechny napájecí větve obvodů. Spotřeba celého modulu se pohybovala kolem hodnoty 90mA. Díky této malé spotřebě modulu bylo možné napájet FITkit i modul pouze z USB portu počítače, bez nutnosti připojení externího adaptéru.

Pro ověření základní funkčnosti jsem si vyrobil ještě testovací modul osazený budičem sběrnice a osmi LED diodami, které umožňovali sledovat stav výstupu A/D převodníku. Na modulu se nachází i tlačítko s pull-up rezistorem, pomocí něhož lze generovat řídicí signál CLK převodníku. Kromě tlačítka jsem na testovací modul umístil i tři přepínače, která nastavují adresu pro výběr rozsahu vertikálního zesilovače.

Testovací modul mi umožnil zkontrolovat si výstup převodníku v každé situaci. Pomocí přepínačů jsem měnil rozsahy a tlačítkem si generoval hodinový signál *CLK*.

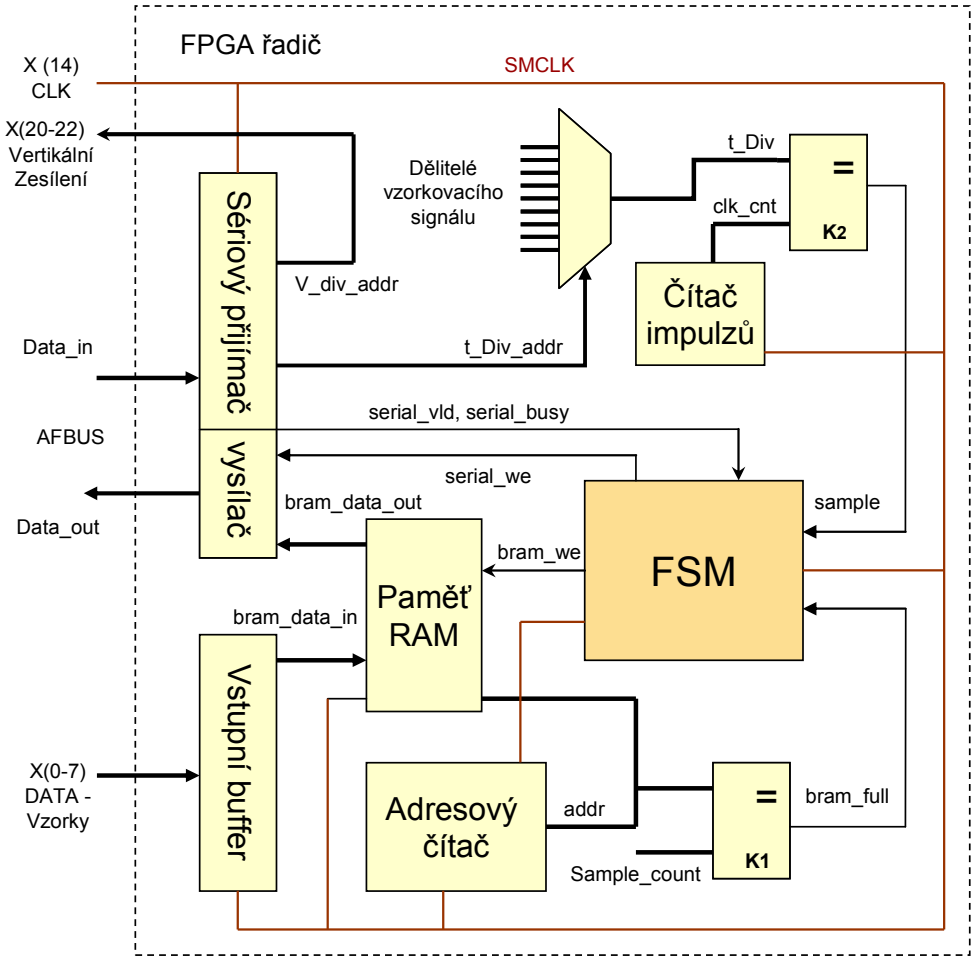
V okamžiku, kdy jsem si byl jistý funkčností modulu, připojil jsem jej k FITKitu. Schéma zapojení, návrh a fotografie testovacího modulu se nachází v příloze.

7 Realizace FPGA řadiče

Největším problémem realizace řadiče bylo navrhnout jeho architekturu. S ohledem na zadané požadavky jsem nejdříve sestavil blokové schéma z jednotlivých komponent. Tyto komponenty jsem v dalších fázích návrhu blíže analyzoval a po jedné implementoval.

7.1 Architektura FPGA řadiče

Na následujícím obrázku se nachází architektura celého řadiče.



Obrázek 7-1 Architektura FPGA řadiče

Rozhraní řadiče tvoří pětice signálů, z nichž trojice je napojena k osciloskopickému modulu a dvojice tvoří komunikační bránu mezi FPGA a grafickou aplikací běžící na PC.

Konkrétně se jedná o signály:

- *CLK (X14)* – Hodinový signál, který s každou náběžnou hranou spouští A/D konverzi převodníku

- *Vertikální zesílení (X(20-22))* – Adresa určující vertikální zesílení osazeného řízeného vertikálního zesilovače
- *Data – Vzorky (X(0-7))* – Digitální osmibitový výstup z převodníku
- *Data_in (AFBUS)* – Sériový vstup dat z PC
- *Data_out (AFBUS)* – Sériový výstup dat z řadiče

V následující části této sekce detailně popíšu zvlášť každou část řadiče.

7.1.1 Sériový přijímač/vysílač

FITkit obsahuje řadu periferií. Jejich ovládání není vždy jednoduché. Proto byla v rámci projektu FITkit vytvořena sada řadičů, které zjednodušují návrh aplikací nebo úloh na kitu. Mezi těmito řadiči byl implementován i řadič sériového rozhraní [18].

Tento řadič umožňuje komunikaci po asynchronní sériové lince pomocí protokolu rs232. Na FITkitu je možné tento řadič použít dvakrát. První možnost je na konektoru JP5, druhá je na USB portu A čipu FT2232C. Konektor JP5 a port A čipu FT2232C jsou přímo připojené k FPGA.

Řadič sériové linky umožňuje příjem i vysílaní slova podle nastavených parametrů rozhraní, které se nastavuje generickými parametry. Samotný řadič je složen ze dvou základních částí: řadič pro odesílaní dat (RS232_TXD) a řadič pro příjem dat (RS232_RXD) [18].

Řadič sériového rozhraní je možné nastavit pomocí pěti generických parametrů. Tyto parametry nastaví řadiče pro příjem a odesílaní na požadované vlastnosti.

Generické parametry:

- **SPEED** - nastavení rychlosti komunikace. Tabulka 7-1 ukazuje nastavení hodnoty SPEED, její odpovídající rychlosti a hodnotu, kterou se dělí hodinový signál SMCLK.
- **BITS** - Počet přenášených datových bitů. Možné hodnoty nastavení tohoto parametru jsou 5 - 8 datových bitů.
- **STOP** - Počet stop bitů. 1 nebo 2 *STOP*bity.
- **PARITY_EN** - Povolení paritního bitu. Pokud je PARITY_EN = 0 znamená to, že není nastavena žádná parita. Pokud je PARITY_EN = 1 znamená to povolení parity. O jakou paritu jde, určuje parametr PARITY.
- **PARITY** - Nastavení druhu parity. Pokud je PARITY = 0, bude počítaná a kontrolovaná sudá parita. Pokud je PARITY = 1, znamená to lichou paritu. Tento parametr se uplatní jenom v případě, pokud je nastavený PARITY_EN na hodnotu 1.

SPEED	Baud rate	Dělící poměr	Poznámka
"0000"	9600	768	7.3728MHz / 768 = 9600
"0001"	921600	8	7.3728MHz / 8 = 921600
"0010"	460800	16	7.3728MHz / 16 = 460800
"0011"	230400	32	7.3728MHz / 32 = 230400
"0100"	115200	64	7.3728MHz / 64 = 115200
"0101"	57600	128	7.3728MHz / 128 = 57600
"0110"	38400	192	7.3728MHz / 192 = 38400
"0111"	19200	384	7.3728MHz / 384 = 19200
"1000"	9600	768	7.3728MHz / 768 = 9600
"1001"	4800	1536	7.3728MHz / 1536 = 4800
"1010"	2400	3072	7.3728MHz / 3072 = 2400
"1011"	1200	6144	7.3728MHz / 6144 = 1200
others	9600	768	7.3728MHz / 768 = 9600

Tabulka 7-1 Hodnoty rychlosti přenosu podle nastavení parametru SPEED [18]

Při realizaci mého řadiče jsem nastavil přenosovou rychlost na maximální hodnotu 921600 Baudů. Protože šířka jednoho vzorku je 8 bitů, nastavil jsem stejnou hodnotu do parametru BITS, udávající počet přenášených bitů. Počet stopbitů jsem nastavil na hodnotu 1. Pro bezchybný přenos dat jsem nastavil sudou paritu.

Rozhraní sériového řadiče má relativně jednoduchou strukturu:

- *CLK, RESET* – Synchronní hodiny (SMCLK) a reset používané pro celé FPGA
- *DATA_IN* – Používá se pro vysílání dat na RS232_TXD
- *WRITE_EN* – Vysoká logická úroveň na tomto signálu vyvolá odeslání dat (DATA_IN) na RS232_TXD
- *DATA_OUT* – Používá se pro čtení dat přijatých na sériové lince.
- *DATA_VLD* – Vysoká logická úroveň na tomto signálu oznamuje příchozí data na sériové lince (DATA_OUT)
- *BUSY* – Vysoká logická úroveň na tomto signálu informuje, že právě probíhá komunikace po sériové lince
- *ERR* – Signál ERR informuje o případné chybě, která se vyskytla v průběhu komunikace. Do návrhu jej však neuvažuji, proto jsem jej označil klíčovým slovem OPEN
- *RXD, TXD, RTS, CTS* – Signálové vodiče pro sériovou komunikaci. V řadiči osciloskopu jsou připojeny na sběrnici AFBUS (0,1,2). Signál CTS ignoruji (OPEN)

Každý příchozí byte se v řadiči osciloskopu rozdělí do dvou signálů. Spodní 3 bity příchozích dat jsou prostřednictvím signálu *V_div_addr* vyvedeny signály skrze sběrnici X(20-22) na konektor JP10 na FITkitu. Tento signál určuje velikost zesílení vertikálního zesilovače.

Horní 4 bity vstupních dat jsou vyvedeny jako signál *t_div_addr* k multiplexoru, kde je na základě této adresy vybrán patřičný dělitel *t_Div*.

7.1.2 Vstupní buffer

Jedná se o klasický osmibitový registr synchronizovaný hodinovým rozvodem *SMCLK*. Registr je připojen přes konektor k výstupu A/D převodníku a slouží k uchování posledního naměřeného vzorku.

7.1.3 Paměť RAM

Jedním z hlavních požadavků na paměť je její rychlost. Jako naprosto dostačující paměť pro ukládání vzorků je paměť Block RAM, která je vestavěná přímo na čipu FPGA. Na použitém čipu XC3S50 jsou vestavěny 4 takovéto paměti. Velikost každé z nich je 18 kBitů.

Čtení z paměti Block RAM na čipu je asynchronní, což znamená, že po vystavení adresy jsou data na výstupu k dispozici okamžitě. Zápis dat je synchronizován vzestupnou hranou hodinového signálu *SMCLK*. Zápis dat se provede pouze v případě, že je nastaven signál *bram_we* ve vysoké logické úrovni.

Paměť RAM lze nastavit do několika konfigurací podle následující tabulky.

Konfigurace	Počet položek	Šířka datové položky	Počet bitů adresy
RAMB16_S1	16384	1 bit	14
RAMB16_S2	8192	2 bity	13
RAMB16_S4	4096	4 bity	12
RAMB16_S9	2048	8 bitů	11
RAMB16_S18	1024	16 bitů	10
RAMB16_S36	512	32 bitů	9

Tabulka 7-2 Konfigurace paměti Block RAM

Vzhledem k použitému osmibitovému převodníku jsem zvolil konfiguraci paměti RAMB16_S9. Tato konfigurace mi umožňuje uložit až 2048 vzorků, což je díky řízené časové základně dostačující počet.

Rozhraní komponenty RAMB16_S9 obsahuje tyto signály:

- *CLK* – synchronizační signál zápisu do paměti. Na tento signál je připojen rozvod hodinového signálu SMCLK.

- *DO* – Paralelní výstup dat z paměti. Prostřednictvím interního signálu *bram_data_out* je připojen přímo k sériovému vysílači.
- *DI* – Paralelní vstup dat do paměti. Prostřednictvím interního signálu *bram_data_in* je připojen k záchytnému registru (*Vstupní buffer*).
- *ADDR* – Adresa generovaná v bloku *čítač adresy*
- *EN* – Povolovací vstup. Nastavil jsem jej na vysokou logickou úroveň, aby byla paměť stále aktivní
- *WE* - Povolení zápisu do paměti. Prostřednictvím signálu *bram_we* jej řídí konečný stavový automat (blok *FSM*)

7.1.4 Adresový čítač

Čítač je řízen stavovým automatem FSM. prostřednictvím signálu *addr* je propojen s pamětí RAM a komparátorem *K1*.

Tento komparátor porovnává aktuální adresu s konstantou *Samples_count*, což je požadovaný počet vzorků. Tato hodnota je na začátku programového kódu nastavena na hodnotu 2000. Teoreticky je možné nastavit ji na hodnotu max. 2047, protože do paměti se vejde právě 2048 položek. V případě, že se hodnota adresy a hodnota *Samples_count* rovnají, je adresa vynulována a nastaví se signál *bram_full.*

7.1.5 Čítač impulzů

Podobně jako adresový čítač funguje čítač impulzů. Ten je důležitý pro správný chod časové základny osciloskopu. Jeho hodnota se zvyšuje o 1 s každou náběžnou hranou hodin.

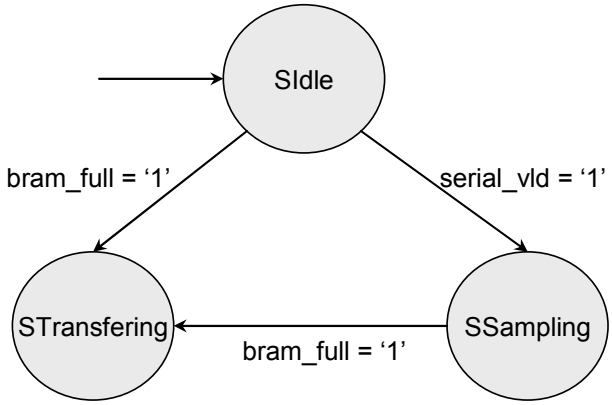
V komparátoru *K2* se hodnota čítače (*clk_cnt*) porovná s aktuálně nastaveným dělitelem vzorkovacího signálu (*t_Div*). Dělitel *t_Div* je vybrán multiplexorem na základě adresy *t_Div_addr*. Na vstupu je připraveno celkem 9 takovýchto dělitelů. Každý dělitel základního kmitočtu SMCLK je dopočítán tak, aby v oknu výsledné aplikace vycházeli „kulaté“ hodnoty časové základny na jeden dílek pomyslné obrazovky osciloskopu. (např. 1ms/dílek, 2ms/dílek ..)

Pokud se hodnota dělitele rovná hodnotě čítače impulzů (*clk_cnt*), nastaví se signál *sample* do log. „1“ a hodnota čítače se vynuluje. Tento mechanizmus se potom uplatní jako časová základna osciloskopu. Signál *sample* totiž zapříčiní uložení aktuálního vzorku do paměti. Více informací se nachází v následující části.

7.1.6 FSM

Blok FSM je konečný automat o třech stavech *SInit, SSampling* a *STransfering*. Automat je synchronizován hodinovým rozvodem SMCLK a řídí pomocí signálů většinu komponent řadiče.

Přechodový diagram automatu naznačuje obrázek 7-2.



Obrázek 7-2 Stavový automat FSM

7.1.6.1 SIdle

Stav *SIdle* je počátečním stavem automatu. V tomto stavu řadič neprovádí žádné akce. Čeká se pouze na příchozí byte, ve kterém se nachází informace o parametrech vzorkování. V případě, že přijde očekávaný byte, sériový přijímač nastaví signál *serial_vld* a automat přejde do stavu *SSampling*.

7.1.6.2 SSampling

V tomto stavu se vzorkují data a ukládají se do paměti. Data se vzorkují pořád stejnou rychlostí (SMCLK), avšak ne vždy se pořízený vzorek uloží do paměti. Vzorek se uloží jen v případě, že je „nahozen" signál *sample*. V tu chvíli se nastaví příznak *bram_we* a aktuální vzorek se uloží do paměti na místo, kam ukazuje adresa *addr*. Jakmile je vzorek uložen, inkrementuje se čítač adresy a vynuluje se čítač impulzů *clk_cnt,* který se jinak inkrementuje s každou náběžnou hranou hodin.

Ve chvíli, kdy je adresa *addr* rovna hodnotě *Samples_count,* je nastaven signál *bram_full* a automat přejde do stavu *STransfering*.

7.1.6.3 STransfering

V tomto stavu má řadič za úkol odeslat naplněnou RAM paměť se vzorky. S náběžnou hranou hodin nastaví řadič signál *serial_we* a zvýší adresu *addr,* čímž se odešle jeden vzorek z paměti. Tuto akci však vykoná jen v případě, že se právě neodesílají žádná data (*serial_busy = 0*).

Automat přejde do počátečního stavu *SIdle* v době, kdy je celá RAM odeslána (*bram_full = 1*) a čeká se na další akci uživatele.

7.2 Implementace FPGA řadiče

Jako implementační jazyk jsem zvolil jazyk VHDL. VHDL je programovací jazyk pro popis hardware (VHSIC HDL – Very High Speed Integrated Circuit Hardware Description Language). Díky tomu, že byl tento jazyk v roce 1987 standardizován organizací IEEE, je zaručena jeho kompatibilita a jednotnost. Jazyk VHDL se používá jak pro simulaci obvodů, tak i pro popis integrovaných obvodů, které se mají vyrábět. Je použitelný dokonce i pro popis analogových obvodů.

Pro implementaci řadiče jsem využil vývojového prostředí Project Navigator verze 9.2i firmy Xilinx, Inc. [11] . Toto vývojové prostředí je možné zdarma stáhnout na stránkách výrobce. Simulaci správné funkce řadiče jsem prováděl pomocí programu ModelSim XE 3 Starter stejné firmy, který je rovněž volně ke stažení na stránkách výrobce, nebo na stránkách projektu FITKit [9].

Každý VHDL soubor popisující hardware se skládá ze dvou částí. Tzv. entity a vlastní architektury.

- *Entita* definuje rozhraní obvodu, tzn. jednotlivé vstupní a výstupní signály.
- *Architektura* obsahuje VHDL kód, kterým je popsána funkce případně chování obvodu.

Jedna entita může mít i více architektur.

V projektu existuje vždy jedna nejvyšší (top-level) entita. V případě FITkitu její signály korespondují s jednotlivými piny FPGA obvodu. Z důvodu usnadnění práce byla v projektu FITkit vytvořena trojice základních entit, které jsou určeny k použití v projektech.

K mému projektu jsem si vybral top-level entitu *tlv_ide_ifc*, která je určena pro aplikace využívající sběrnici X jako port IDE případně jako univerzální rozhraní, na které je možné připojit různé periferie. K této entitě zbývalo dopsat architekturu znázorněnou v kapitole 7.1.

Nejdříve jsem si nadefinoval všechny potřebné signály a konstanty. Poté jsem do zdrojového kódu vložil komponentu *serial_transceiver* pro sériovou komunikaci a komponentu *RAM16_S9*. Jednotlivými procesy jsem naimplementoval jednotlivé bloky architektury, jako je multiplexor a komparátory K_1 a K_2. Stavový automat FSM jsem implementoval pomocí tří procesů:

- *pstatereg* - proces uchovávající aktuální stav
- *nstate_logic* – kde je definována logika následujícího stavu
- *output_logic* – udává, jaké akce se mají vykonávat v současném stavu.

Celou architekturu jsem naimplementoval do souboru *top.vhd*. K projektu bylo nutné připojit ještě řadu dalších souborů knihovny FITkit. Všechny zdrojové soubory nutné k procesu kompilace a syntézy jsou k dispozici na přiloženém CD.

8 Realizace aplikace na PC

Tato kapitola se zabývá realizací grafické aplikace umožňující komunikovat s FITkitem a zobrazovat na monitoru naměřené vzorky.

První část kapitoly se zabývá výběrem vhodného programovacího jazyka a implementačního prostředí, následuje rozbor komunikačních možností, dále rozdělení aplikace do jednotlivých tříd a nakonec popis jednotlivých tříd.

8.1 Výběr programovacího jazyka a prostředí

Po pečlivé úvaze a zhodnocení všech výhod a nevýhod jsem zvolil jazyk C# jako implementační jazyk pro moji aplikaci.

Jazyk C# je vysoce úrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework, který byl později standardizován komisemi ECMA a ISO. Jazyk C# je založen na jazycích Java a C++, ze kterého čerpá především syntaxi. C# lze využít k tvorbě databázových programů, webových aplikací a stránek, webových služeb, formulářových aplikací ve Windows, softwaru pro mobilní zařízení (PDA a mobilní telefony) atd. [22].

Vývojových prostředí pro jazyk C# je více (Turbo C# Explorer, SharpDevelop, MonoDevelop), vybral jsem si však oficiální vývojové prostředí firmy Microsoft – Visual Studio 2005. Visual Studio je podle mého názoru asi nejpropracovanějším prostředím pro formulářové aplikace, v jakém jsem měl možnost pracovat. Toto prostředí je možné pro studijní účely získat na serveru *msdn.e-academy.com* po předchozí registraci.

Pro tuto aplikaci se ukázal být programovací jazyk a prostředí téměř ideální, díky rychlému vytvoření layoutu aplikace a velice snadné komunikaci po sériovém portu. Jedinou velkou nevýhodou této volby je závislost na operačních systémech firmy Microsoft a nutnost instalace platformy .NET framework alespoň verze 2.0.

8.2 Komunikace PC aplikace s FITkitem

Jak již bylo psáno v kapitole 5.4, díky osazenému čipu FT2232C na FITkitu, degraduje komunikace s FPGA čipem na komunikaci po sériové lince.

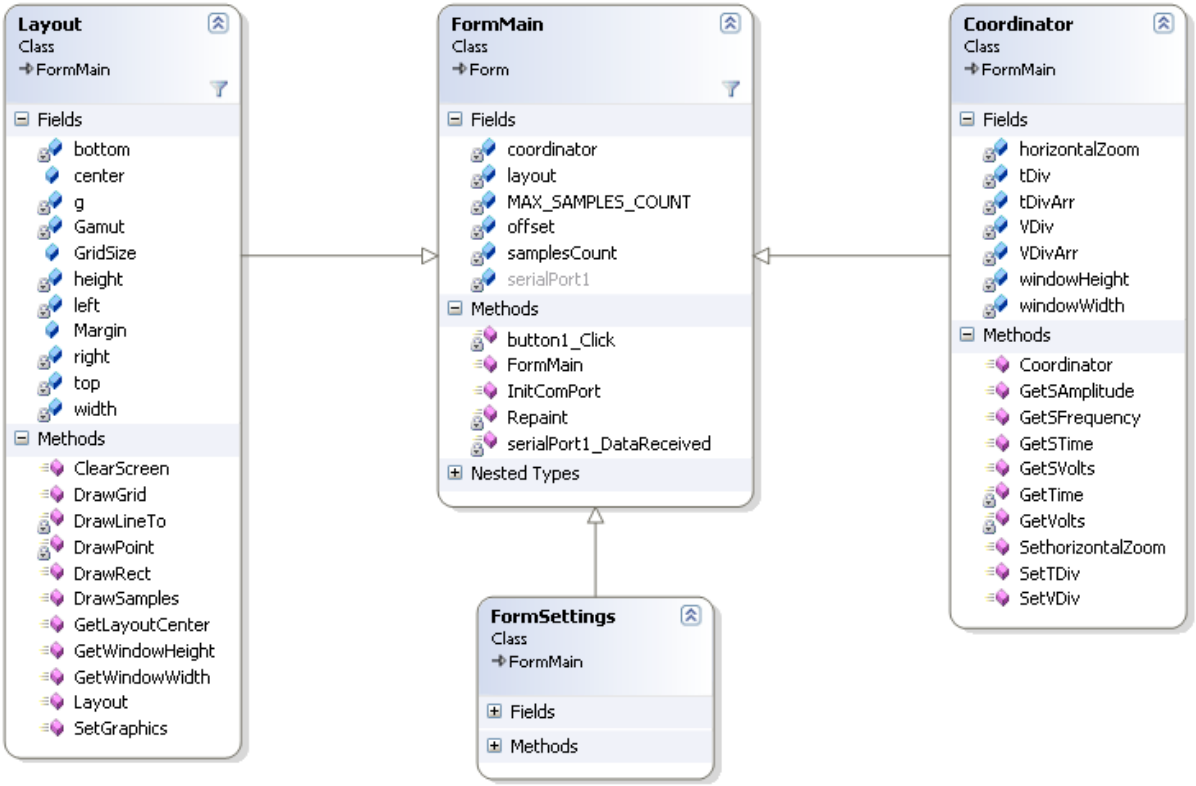
Díky použitému vývojovému prostředí byla problematika komunikace velice usnadněna. V prostředí MS Visual Studia je totiž integrovaná komponenta *serialPort* pro práci se sériovým portem.

Komponenta disponuje všemi vlastnostmi (Properties), které jsem při práci se sériovým portem potřeboval. Ve vlastnostech jsem si tedy mohl nastavit jméno komunikačního portu, přenosovou rychlost, paritu, počet stop Bitů a další.

V událostech komponenty (Events) jsem využil pouze jedinou. A sice událost *DataReceived*. Tato událost se zavolá vždy pokud přijdou na sériový port nějaká data. Potom již stačí metodou *Read* patřičná data z fronty vybrat.

8.3 Třídy aplikace

Aplikaci realizující požadované funkce jsem nazval *FITKit Scope*. Na následujícím obrázku 8-1 je znázorněn diagram tříd této aplikace. Mezi vlastnosti a metody tříd jsem v diagramu uvedl jen ty nejdůležitější.



Obrázek 8-1 Diagram tříd aplikace FITKit Scope

8.3.1 Třída FormMain

Třída FormMain je hlavní třída projektu, řídicí celý běh programu. Třída implementuje hlavní formulář aplikace, na který jsem přidal řadu komponent, umožňující ovládání aplikace.

Na formuláři se nachází dvojice komponent *CommboBox*. Jedna slouží pro výběr vertikálního zesílení a druhá k nastavení časové základny. Pro výběr časové základny je k dispozici celkem

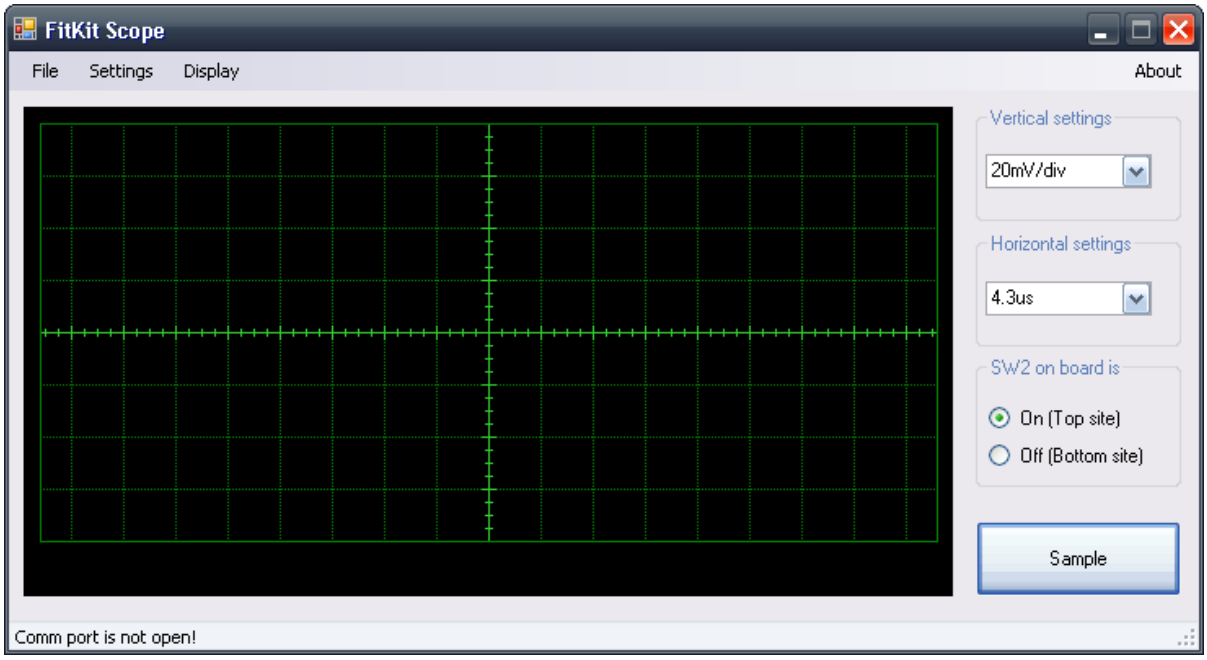
8 položek pohybujících se v rozmezí od 4µs do 20ms na jeden dílek vykreslený na pomyslné obrazovce osciloskopu. Každá položka komponenty potom zapříčiní odeslání adresy dělitele hodin FPGA řadiče *t_Div*, který je popsán v kapitole 7.1.

Nastavení vertikálního zesílení má však rozsahy dva. Jeden rozsah čítá 5 položek a je nastavitelný v rozmezí 20mV až 0.5V na dílek. Druhý rozsah pak nabízí možnosti 1V, 2V a 5V na jeden vykreslený dílek. O tom, který rozsah bude v komponentě zobrazen rozhoduje dvojice zaškrtávacích tlačítek (*radioButtons*). Tato tlačítka se vyptávají na stav přepínače SW2, který je osazený na desce osciloskopického modulu. Tento vypínač, jak bylo psáno v kapitole 6.2, odpojuje, resp. připojuje vstupní napěťový dělič a tím pádem ovlivňuje zesílení vstupního signálu. Protože přepínač není elektronicky řízen, aplikace vyžaduje, aby o stavu přepínače informoval uživatel prostřednictvím těchto tlačítek.

Pomyslná obrazovka osciloskopu se vykresluje do komponenty *pictureBox*. Pod touto komponentou se nachází trojice políček (*Label*), které zobrazují číselné údaje měření jako je amplituda, perioda a frekvence měřeného signálu.

V horizontálním menu aplikace se dále nacházejí funkce pro exportování naměřeného obrázku do souboru (*File/Export Image*), tlačítko pro vyvolání dialogu s nastavením připojení (*Settings/Connection settings*) a možnost různého vykreslování vzorků v menu *Display*.

Celý layout aplikace završuje tlačítko s názvem *Sample*, které slouží pro spuštění procesu vzorkování. Vzhled aplikace je patrný na následujícím obrázku 8-2.



Obrázek 8-2 Vzhled aplikace FitKit Scope

Následující seznam popisuje nejvýznamnější metody této třídy.

- **FormMain()** : Konstruktor třídy. Zde se inicializují všechny komponenty formuláře. Poté se vytvoří instance tříd *Layout* a *Coordinator*. Tyto třídy budou popsány níže.
- **pictureBox1_Paint()** : Metoda, která se automaticky zavolá v případě potřeby překreslit komponentu *pictureBox*. Odtud se volají metody třídy *Layout*, které se o správné vykreslení postarají.
- **InitComPort(string portName)** : Tato metoda inicializuje název sériového portu řetězcem předaným v parametru *portName* a pokusí se tento port otevřít.
- **serialPort1_DataReceived(...)** : Metoda, která se vyvolá v případě, že se na sériovém portu objeví příchozí data. V našem případě se vždy jedná o příchozí vzorky, které se v metodě uloží do připraveného pole *samples*.

8.3.2 Třída Layout

Třída Layout zapouzdřuje vlastnosti a metody pro vykreslování prvků na pomyslnou obrazovku osciloskopu (komponenta *pictureBox*). Třída prostřednictvím svých dat uchovává rozměry okna pro vykreslování, rozměry jednotlivých dílků, předdefinovaná pera pro kreslení na obrazovku a další.

Mezi její nejvýznamější metody patří:

- **Layout(int Width, int Height) :** Metoda je konstruktorem třídy. Z předaných rozměrů se spočítají některé důležité souřadnice jako je např. střed okna, okraje atd.
- **DrawGrid() :** Metoda do připraveného okna nakreslí pomocí čar a předdefinovaných per mříž. Tato mříž je rozdělena na 8 dílků ve vertikální poloze a počet dílků v horizontální poloze se dopočítá podle velikosti okna. Metoda nakonec nakreslí osy se stupnicí, které umožní odečítat hodnoty napětí a času.
- **DrawSamples(Byte[] Samples, int Count, int Offset) :** Metoda slouží pro vykreslení získaných vzorků v poli *Samples* na obrazovku. Parametr *Count* určuje počet dostupných vzorků a parametr *Offset* určí, od kterého vzorku se má začít vykreslovat. Aplikace totiž umožňuje se pomocí myši pohybovat po časové ose signálu. Metoda pro každý vzorek zavolá buď metodu *DrawPoint(int X, int Y)*, nebo metodu *DrawLineTo(int X, int Y),* podle toho, zda uživatel vybral zobrazování bodové, nebo pomocí úseček. Díky tomu, že o časovou základnu a vertikální zesílení se již postaral osciloskopický modul s FPGA řadičem, metoda *DrawSamples* už pouze vykresluje každý vzorek na jeden obrazový bod na časové ose. Tzn. 2000 naměřených vzorků znamená průběh o „délce“ 2000 pixelů. Protože je šířka plochy pro vykreslování vysoká 256 pixelů, není třeba přepočítávat ani vertikální složku vzorků.

8.3.3 Třída Coordinator

Tato třída zapouzdřuje data a metody pro práci s nastavením vertikální a horizontální složky signálu. Datové položky třídy uchovávají kromě jiného dvě pole koeficientů *VDivArr* a *tDivArr*. Tyto koeficienty slouží ke přepočítání polohy myši nacházející se nad plochou pro vykreslování vzorků na hodnoty napětí a času. Přepočet je samozřejmě ovlivněn nastavením časové základny a vertikálním zesílením osciloskopu.

Mezi nejvýznamnější metody třídy patří:

- **GetVolts (Point p1, Point p2):** Metoda nejdříve spočítá rozdíl vertikálních složek předaných bodů p1 a p2. Poté rozdíl vynásobí jedním z koeficientů z pole *VDivArr*. Vypočítanou hodnotu funkce vrátí v podobě čísla datového typu *double*. Pomocí metody lze určit okamžitou hodnotu napětí v závislosti polohy myši a nastaveném vertikálním rozsahu.
- **GetSVolts (Point p1, Point p2):** Obdobná funkce jako *GetVolts* s tím rozdílem, že hodnota napětí je vrácena v podobě řetězce včetně příslušné jednotky.
- **GetSAmplitude (Point p1, Point p2):** V podstatě stejná metoda jako předchozí. Avšak vrací absolutní hodnotu napětí. Tuto metodu používám k měření amplitudy signálu.
- **GetTime(Point p1, Point p2):** Vrací hodnotu času, který je vypočítán ze vzdálenosti předaných bodů *p1* a *p2*.
- **GetSTime(Point p1, Point p2) :** Hodnotu času vrátí jako řetězec.
- **GetSFreqency(Point p1, Point p2):** Vypočítá a v podobě řetězce vrátí měřenou frekvenci.

8.3.4 Třída FormSettings

Třída implementuje jednoduchý formulář s jednou komponentou *Combobox* a dvěma tlačítky. Jedná se o dialogový formulář, který umožňuje nastavit komunikační port. Pomocí *ComboBoxu* uživatel vybere požadovaný port Com. Tlačítkem *Connect* pak může volbu potvrdit a metodou *InitPort* třídy *FormMain* se aplikace pokusí zvolený port otevřít.

Druhé tlačítko *Cancel* volbu zruší a zavře dialogové okno.

8.4 Chování aplikace

Při spuštění aplikace *FitKit Scope* se v konstruktoru třídy *FormMain* zavolá metoda *InitCom*, která se pokusí otevřít komunikační port. Pokud se to metodě podaří, vypíše se tato informace do „statusBaru“ hlavního okna. Zároveň se při startu aplikace vytvoří instance tříd *Layout* a *Coordinator*.

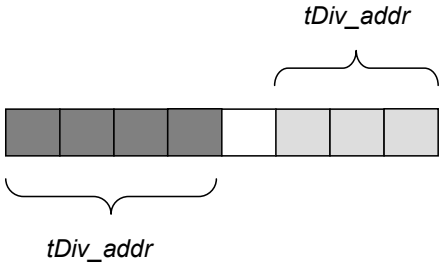
Prostřednictvím metod třídy *Layout* se na obrazovku vykreslí plocha pro vykreslování průběhu signálu se všemi dílky, osami a stupnicí.

Pokud uživatel začne měnit položky v komponentách *comboBox*, úměrně se mění i jednotlivé koeficienty *tDiv* a *VDiv* v instanci třídy *Coordinator.*

Najedeme-li kurzorem myší nad plochu pomyslné osciloskopické obrazovky, aplikace okamžitě vypočítává metodami *GetVolts a GetSVolts* napětí vzhledem k poloze myši a poloze časové osy. Toto napětí se průběžně překresluje do připravených políček (*Label*).

Stiskne-li uživatel levé tlačítko myši, uchová se její aktuální poloha a při dalším pohybu myši je na ploše vykreslován obdélník, který umožňuje provádět měření na ploše obrazovky. Metody třídy *Coordinator* totiž přepočítávají velikost zobrazeného obdélníku na hodnoty napětí, času a frekvence. Tyto údaje jsou rovněž zobrazovány do připravených políček pod „osciloskopickou obrazovkou“.

Událost vyvolaná při stisku tlačítka *Sample* musí odeslat jeden Byte po sériovém kanálu, který oznámí řadiči, že se uživatel rozhodl vzorkovat data. Tento Byte musí obsahovat jak adresu *tDivAddr,* tak adresu *VDivAddr*. Struktura tohoto Byte je na následujícím obrázku.



Obrázek 8-3 Struktura odeslaného Byte

Tyto adresy nejsou nic jiného než indexy zvolených políček v komponentách *ComboBox.* Zmiňovaný Byte se následně odešle metodou *Write* třídy *serialPort.*

Bezprostředně po odeslání dat, se vyvolá událost *DataReceived* oznamující příchozí vzorky. Tyto vzorky jsou ihned po přijetí vykresleny metodou *DrawSamples* třídy *Layout* a aplikace je znovu připravena na další akce uživatele.

8.5 Aplikace FitKit Logic Analyzer

Na přání Ing. Šimka jsem jako rozšíření diplomové práce naimplementoval ještě druhou aplikaci s názvem *FiTKit Logic Analyzer.*

Tato aplikace je vhodná k měření digitálních signálů a její funkce se dá přirovnat k funkci logického analyzátoru. Vykresluje tedy jakýsi idealizovaný průběh naměřeného digitálního signálu. Tzn. před vykreslením se každý vzorek analyzuje a určí se, zda hodnota napětí vzorku spadá do logické úrovně „0“ nebo „1“. Podle toho se poté vykreslí idealizovaný obdélníkový průběh. Pokud nastane situace, že vzorek nezapadne ani do jednoho intervalu pro log. „0“ nebo log. „1“, je vzorek vykreslen odlišnou barvou ve středu vertikální osy.

8.5.1 Formulář FormMain

Vzhled hlavního formuláře *FormMain* je velice podobný formuláři aplikace *FitKit Scope.* Nastavení časové základdany je identické jako u předchozí aplikace.

Hlavním rozdílem těchto aplikací je způsob vertikálního nastavení. V aplikaci *FitKit Logic Analyzer* je tato volba nazvaná jako „Switching Level“. Klasická volba napětí na jeden dílek zde chybí. Pomocí komponenty ComboBox *Switching Level* si uživatel vybírá „technologii“ měřeného signálu.

Na výběr jsou technologie *5V CMOS, 5V TTL, 3.3V LVTTL, 2.5V CMOS, 1.8V CMOS, 1.5V CMOS, 1.2V CMOS* a také položka *User defined.* Tyto položky jsou zase rozděleny do dvou kategorií podle toho, jak je nastaven přepínač SW2 na desce, potažmo podle nastavení komponent RadioButtons.

Poslední zmiňovaná položka *User Defined* je specifická možností nastavení vlastních rozhodovacích úrovní. Tyto rozhodovací úrovně si uživatel může nastavit v dialogovém oknu *Menu/Settings/Switching Level Settings*. Dialogové okno má název *FormTreshold* a obsahuje čtveřici textových polí a potvrzovací tlačítko. Do těchto polí stačí zadat minimální a maximální hodnotu napětí logické úrovně „0“ a minimální a maximální hodnotu napětí logické úrovně „1“.

Tlačítkem *Analyze* na hlavním formuláři pak uživatel spustí logickou analýzu vstupního signálu.

8.5.2 Třída Level

Třída Level zapouzdřuje data a metody, které se starají o rozhodnutí, zda naměřený vzorek spadá do intervalu log. „0“, nebo log. „1“.

Třída uchovává pole *tresholdsArr* datového typu *Treshold*. V tomto poli jsou uloženy prahové rozhodovací úrovně, podle kterých je každý vzorek „zaškatulkován“. Vlastnost *treshold* uchovává aktuálně nastavené prahové úrovně. Tj. jednu položku z pole *tresholdsArr*.

Mezi nejdůležitější metody třídy patří:

- **TresholdInit()**: Metoda je volána v konstruktoru třídy *Level*. Úlohou metody je naplnit pole *tresholdsArr* prahovými úrovněmi napětí podle jednotlivých technologií. Konkrétní úrovně jsem zvolil podle literatury [24].
- **SetUserTreshold(double min0, double max0, double min1, double max1)**: Nastaví uživatelsky definované prahové úrovně.
- **Is1(double Volts), Is0(double Volts)**: Metody vrátí hodnotu typu Boolean podle toho, zda je hodnota *Volts* vyhodnocena jako „log.1“ nebo „log.0“.

9 Testování

Tato kapitola se zabývá uvedením každé části projektu do chodu a nakonec zhodnotí naměřené výsledky v porovnání se sériově vyráběným osciloskopem.

9.1 Oživení všech částí projektu

Abychom mohli oživit hardwarovou část projektu, je třeba připojit osciloskopický modul ke konektoru JP10 na FITkitu, připojit FITkit pomocí USB kabelu k PC a nahrát vygenerovanou konfiguraci do čipu FPGA na FITkitu. K BNC konektoru na modulu je ještě nutné připojit osciloskopickou sondu. Důležité je, aby se jednalo o pasivní sondu 1:1. Některé sondy označené jako 1:10 totiž signál zeslabují. Nebo je možné použít sondu s přepínačem, kde je poměr 1:1 nastavitelný.

Konfigurace pro FPGA je přiložena na CD spolu se všemi zdrojovými VHDL kódy. Pro překlad těchto kódů a vygenerování konfiguračního souboru je možné buď využít vývojového prostředí Project Navigator, nebo projekt zkompilovat přiloženým makefilem, podobně jako ostatní aplikace projektu FITkit.

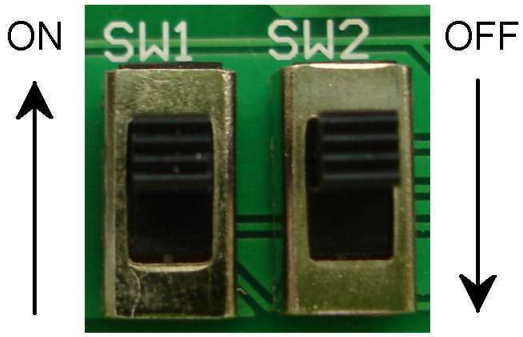
Zdrojové i spustitelné soubory aplikací *FitKit Scope a FitKit Logic Analyzer* jsou rovněž k dispozici na přiloženém CD. Více informací o všech souborech na CD je k dispozici v souboru *Readme.txt*, který se nachází v kořenovém adresáři CD.

V okamžiku, kdy máme vše připojeno a v FPGA se nachází správná konfigurace, můžeme spustit aplikaci *FitKit Scope*. Po spuštění se dole ve Status baru objeví informace, zda byl seriový port korektně otevřen, či nikoliv. Pokud ne, musíme v nastavení připojení vybrat správný port.

Po zdařilém připojení nejprve zkratujeme vstupní sondu a stiskneme tlačítko *Sample*. Na obrazovce se objeví červenou barvou průběh signálu. Pokud není průběh vykreslen přesně na hodnotě 0V, můžeme pomocí potenciometru P1 na osciloskopické desce převodník zkalibrovat.

Nyní již můžeme měřit libovolná napětí a frekvence v rozsahu, který nám aplikace dovoluje. Měli bychom si ale dát pozor na nastavení přepínače SW2 na desce modulu. Při měření napětí větších než ±2V musí být v tento přepínač poloze *Off (Bottom site)*. Poloha *Off* je taková poloha přepínače, kdy je jeho páčka přepínače dále od nápisu „SW2“ na desce. Lépe polohy přepínačů ilustruje obrázek 9-1.

Pokud bychom nechtěli zohlednit stejnosměrnou složku vstupního signálu, přepneme přepínač „SW1“ do polohy *Off,* čímž zařadíme do cesty signálu kondenzátor, který stejnosměrnou složku nepropustí. V případě, že budeme měřit rychle se měnící signál (obdélníkový signál), můžou se ve vykreslovaném signálu zobrazovat nežádoucí jevy typické při nabíjení či vybíjení kondenzátoru. Tento neduh lze odstranit pootočením regulovatelné kapacity C5 na desce, čímž se vykompenzuje vstupní napěťový dělič.



Obrázek 9-1 Polohy přepínačů SW1 a SW2

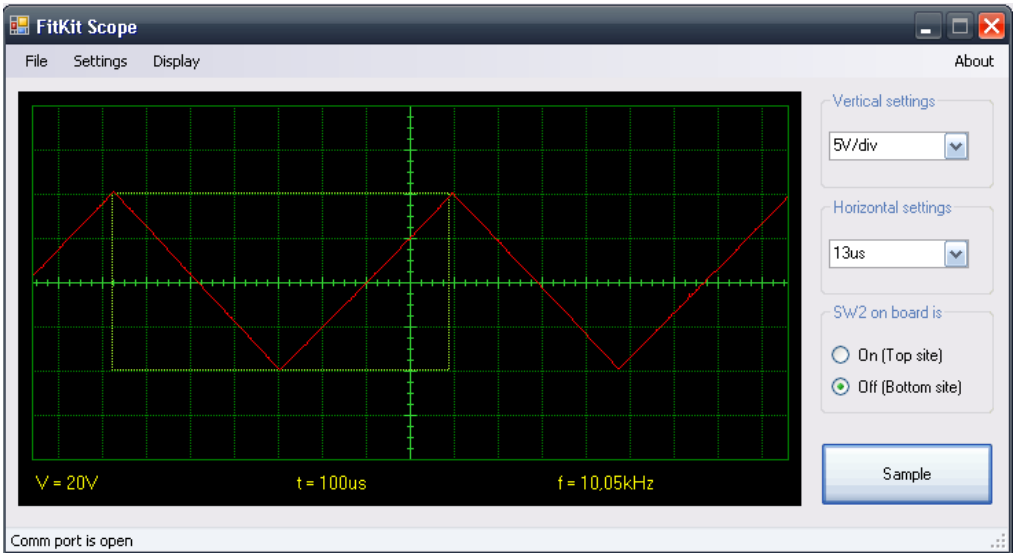
9.2 Měření a dosažené výsledky

Měření jsem prováděl ve fakultní laboratoři L305, kde jsem srovnával naměřené výsledky s výsledky naměřenými na tamějším osciloskopu. Vstupní data jsem generoval z generátoru signálu.

Po chvíli měření se ukázalo, že nesouhlasí časový údaj v jednom z rozsahů. Po přepočítání jednoho z dělitelů základního kmitočtu v programu FPGA byla chyba vyřešena. Drobné odchylky na časové ose lze sledovat u vysokých frekvencích ve stovkách kHz, při menších frekvencích je měření v časové ose naprosto přesné. Tato chyba je způsobena tím, že frekvence hodinového signálu není celé číslo, ale má dlouhý desetinný rozvoj. Uživatel však požaduje, aby hodnota času vyjádřená na jeden dílek byla pokud možno celé „kulaté“ číslo. Proto zde vznikne drobná zaokrouhlovací chyba.

Ve vertikální rovině je možné pozorovat výkyvy převodníku, které se pohybují kolem 5% z nastaveného rozsahu napětí. Při rozsahu 20mV na dílek tedy dosahuje odchylek ±1mV.

Následující obrázek ukazuje jeden z výstupů aplikace a zároveň dokumentuje funkčnost zařízení. Vstupní signál byl vygenerován generátorem pilového průběhu s amplitudou 10V a periodou 10kHz. Více výstupů aplikace je zobrazeno v příloze A.



Obrázek 9-2 Výstup aplikace při měření pilového průběhu signálu

10 Závěr

Cílem práce bylo seznámit se s problematikou analogově digitálního převodu, principem A/D převodníků, prostudovat jazyk VHDL pro návrh číslicového hardware a obeznámit se s výukovou platformou FITkit. S těmito znalostmi poté navrhnout a realizovat řešení, zabývající se využitím platformy FITkit a jeho programovatelných součástí, coby prostředníkem mezi modulem s analogově číslicovými převodníky a počítačem. Aplikace na PC pak měla umožnit zobrazovat průběhy měřeného napětí, podobně jako to dělá digitální osciloskop.

Studium literatury týkající se analogově digitálního převodu a A/D převodníků [1-8] se uplatnilo zejména při tvorbě úvodních kapitol práce a bylo důležité při návrhu konkrétního zapojení analogového modulu. K nastudování platformy FITkit mi pomohla především literatura dostupná na webových stránkách projektu FITkit [9]. S řešením hardwarové části projektu mi byla nápomocná literatura a produkty týkající se návrhu schématu zapojení [17], výběru součástek [15, 16, 17, 19, 20] a návrhu desky plošných spojů [21]. K návrhu a realizaci FPGA řadiče jsem využil literaturu a produkty vzniklé v projektu FITkit [9, 18, 23].

Výsledkem této práce je funkční osciloskopický modul, připojitelný k platformě FITkit, FPGA aplikace řídicí proces vzorkování signálu a dvě grafické aplikace, které umožňují zobrazení měřeného signálu na obrazovce počítače.

Co se týká možnosti rozšíření této práce, všichni jistě cítíme, že prostoru k rozšiřování je zde ještě dostatek. Největší výzva však pravděpodobně bude ve zvyšování výkonu a přesnosti osciloskopického modulu. Softwarové části projektu by pak bylo vhodné rozšířit například o možnost permanentního zobrazování vzorků v čase (Real-time), což je možné vidět u sériově vyráběných osciloskopů. Dále je možné řešit problémy s antialiasingem, či přidat různé možnosti spouštění osciloskopu (pre-trigger, post-trigger...).

Přínosem této práce je rozšíření projektu FITKit o další zajímavou aplikaci, která doufám vzbudí zájem i ze strany studentů. Největším přínosem však tato práce byla pro mě. Umožnila mi nahlédnout do problematiky návrhu hardwarového modulu a si pak jej vlastnoručně osadit. Dále mi rozšířila obzory v návrhu a fyzickému využití programovatelného hardwaru v podobě FPGA řadiče.

Jedním z cílů projektu byla také možnost využití osciloskopického modulu pro podporu výuky. Proto byl při návrhu kladen důraz na jednoduchost, přehlednost grafické aplikace a na výběr všech součástek s ohledem na jejich cenu a dostupnost.

Literatura

[1] Malina, V. *Poznáváme elektroniku VII, Osciloskop a jeho využití*. České Budějovice, 2002.

[2] Lavický, M. *Analogově číslicové převodníky*, dokument dostupný na URL: http://lavicky.webpark.cz/projekt/index.html (prosinec 2007)

[3] *Analog-to-digital conversion*, článek dostupný na URL: http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci213760,00.html (prosinec 2007)

[4] *Shannonův teorém*, článek dostupný na serveru: http://cs.wikipedia.org/wiki/Shannon%C5%AFv_teor%C3%A9m (květen 2008)

[5] Šebesta, V. *Systémy, procesy a signály 1*. Skriptum VUT, VUTIUM, Brno, 1997.

[6] *A/D převodník*, článek dostupný na URL: http://cs.wikipedia.org/wiki/A/D_p%C5%99evodn%C3%ADk (prosinec 2007)

[7] Háze, J., Vrba R., Fujcik L., Sajdl O. *Teorie vzájemného převodu analogového a číslicového signálu.* FEKT VUT, Brno, 2006

[8] Pospíšil, V. *A/D a D/A převodníky*, dokument dostupný na URL: http://praktika.fjfi.cvut.cz/data/VIP/V.Pospisil/scan/prevodniky%20A_D.pdf (prosinec 2007)

[9] Fučík, O., aj. *Projekt FITkit*, dokumentace dostupná na URL: http://merlin.fit.vutbr.cz/FITkit (květen 2008)

[10] Filip, T. *Testování výukové platformy FITkit*, diplomová práce, Brno, FIT VUT v Brně, 2007

[11] Xilinx, Inc., *WebPACK ISE 8.2i*, produkt dostupný na URL: http://www.xilinx.com/ise/logic_design_prod/webpack.htm (prosinec 2007)

[12] Future Technology Devices International Ltd., *Dual USB UART/FIFO IC* , 2007 dokument dostupný na URL: http://www.ftdichip.com/Products/FT2232C.htm (prosinec 2007)

[13] Texas Instruments, Inc., MSP430F168 *Mixed Signal Microcontroller*, dokument dostupný na URL: http://focus.ti.com/lit/ds/symlink/msp430f168.pdf (prosinec 2007)

[14] *The GCC toolchain for the Texas Instruments MSP430 MCUs*, produkt dostupný na URL: http://mspgcc.sourceforge.net/index.html (prosinec 2007)

[15] Philips Semiconductors, *TDA8703*, dokument dostupný na URL: http://www.nxp.com/acrobat_download/datasheets/TDA8703_3.pdf (prosinec 2007)

[16] Texas Instruments, Inc., *SN74CB3T3245* dokument k produktu dostupný na URL: http://focus.ti.com/lit/ds/symlink/sn74cb3t3245.pdf (prosinec 2007)

[17] Doležal, D. *Jednoduchý digitální osciloskop pro PC,* Praktická elektronika 10/2006

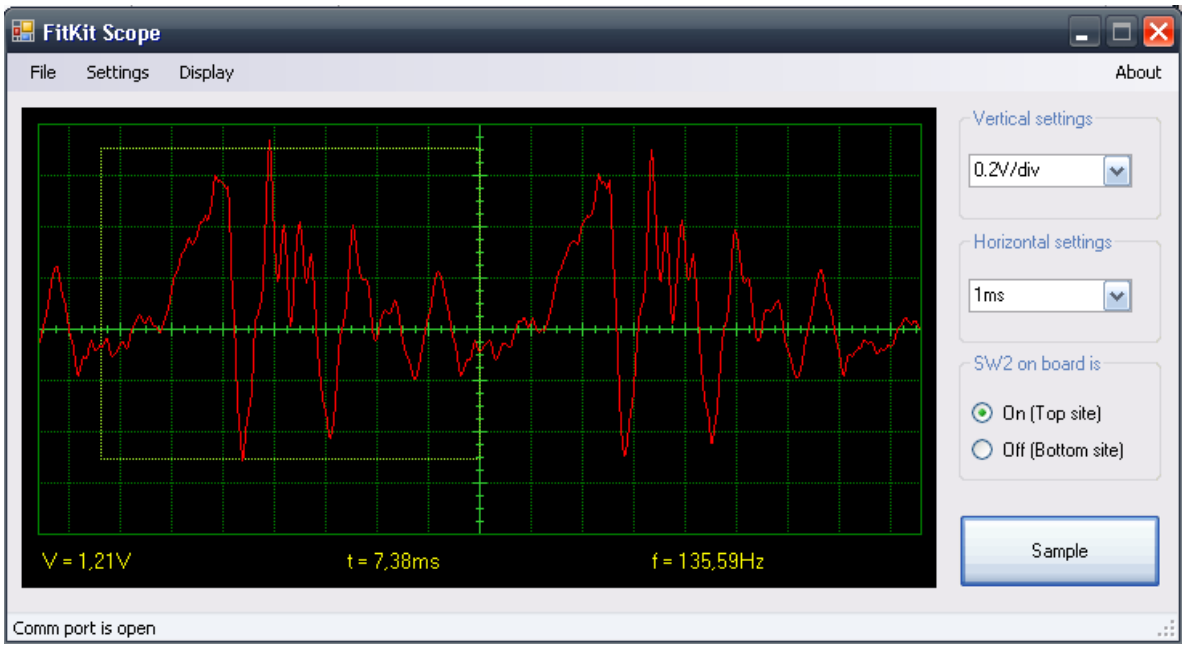
[18] Markovič, J. *Firmware pro ovládání sériového rozhraní*, 2007, produkt dostupný na URL: http://merlin.fit.vutbr.cz/FITkit/docs/firmware/20060414ser.html (prosinec 2007)

[19] QLT Power - Aimtec, *QDC1S-0505S*, dokument k produktu dostupný na URL: http://tme.cz/dok/a16/qdc2s-2405s.pdf (květen 2008)

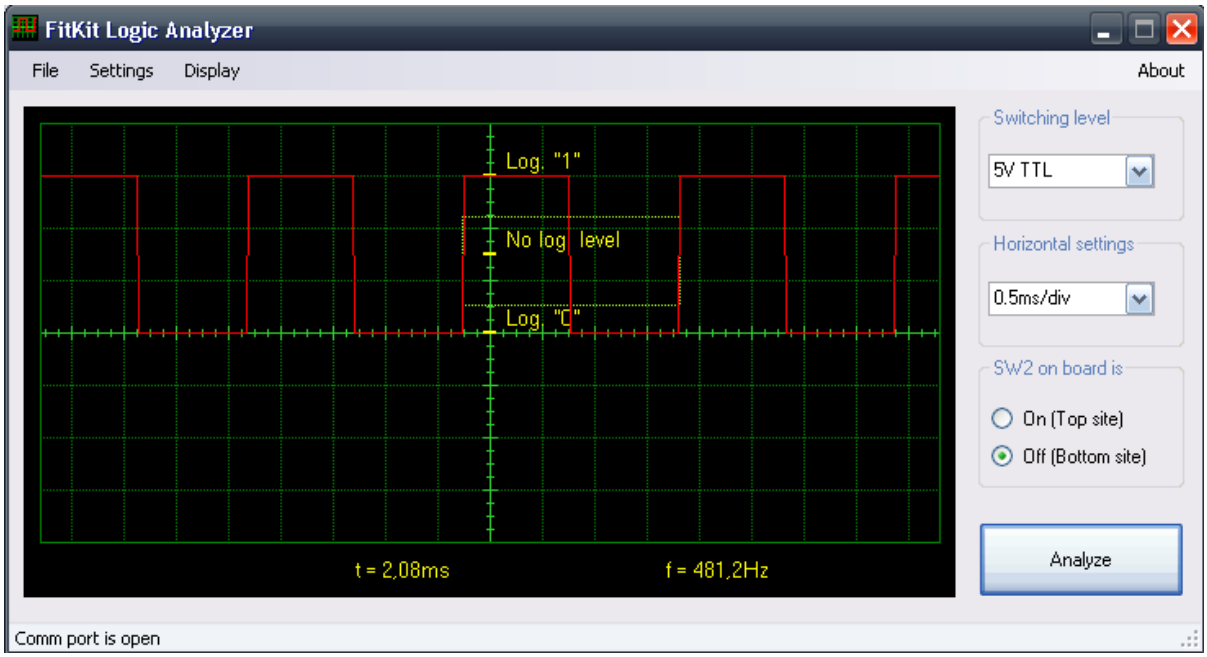
[20] Maxim, *ICL7660,* dokument k produktu dostupný na URL:
http://pdfserv.maxim-ic.com/en/ds/ICL7660-MAX1044.pdf (květen 2008)
[21] CadSoft, *Eagle Light,* produkt dostupný na URL:
http://www.elcad.cz/eagle (květen 2008)
[22] *C Sharp*, článek dostupný na URL:
http://cs.wikipedia.org/wiki/C_Sharp (květen 2008)
[23] Dhond, P. *Selecting the Right Level-Translation Solution*, 2004, dokument dostupný na URL:
http://focus.ti.com/lit/an/scea035a/scea035a.pdf (květen 2008)

Přílohy

Příloha A – Grafické výstupy aplikací

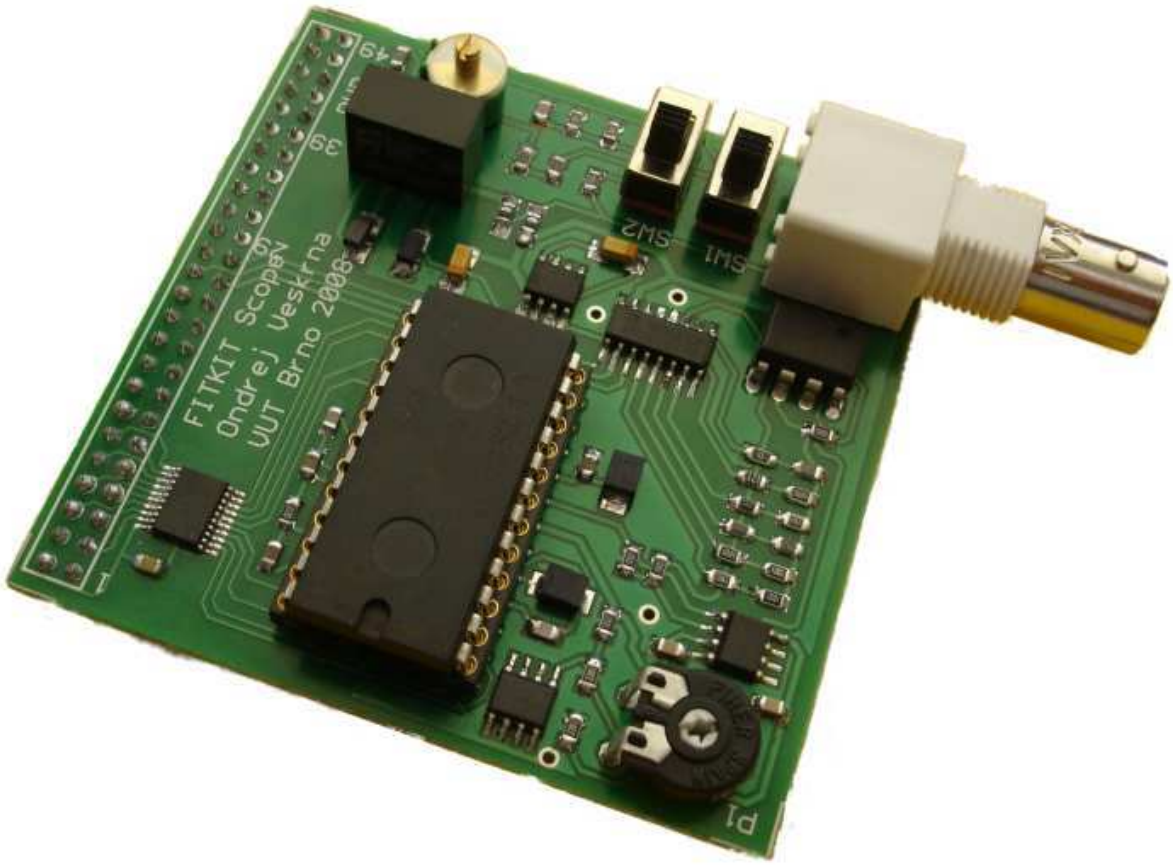


Měřený výstup zvukové karty aplikace FitKit Scope (Pokus o moje komorní „A“)



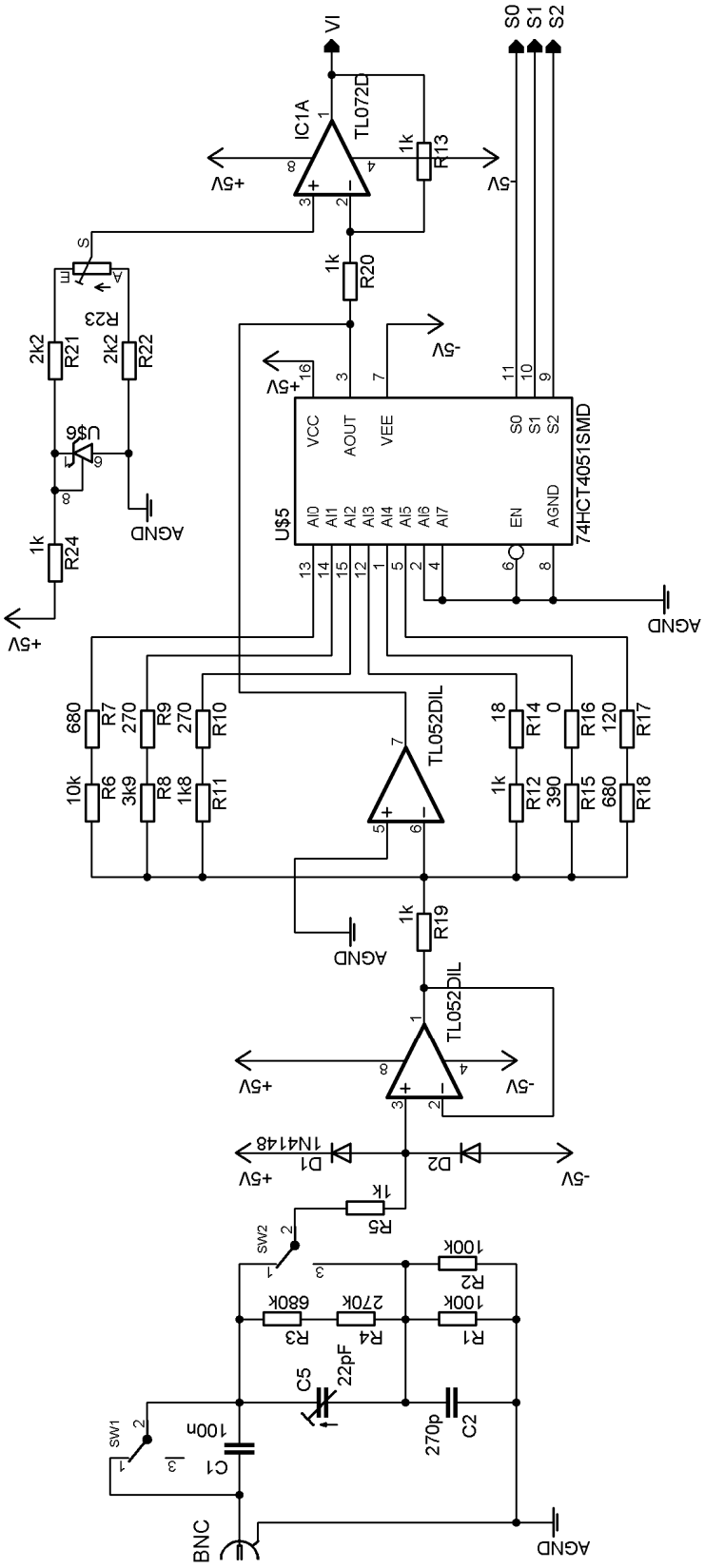
Grafický výstup logického analyzátoru FitKit Logic Analyzer při měření TTL signálu

Příloha B – Osazený osciloskopický modul

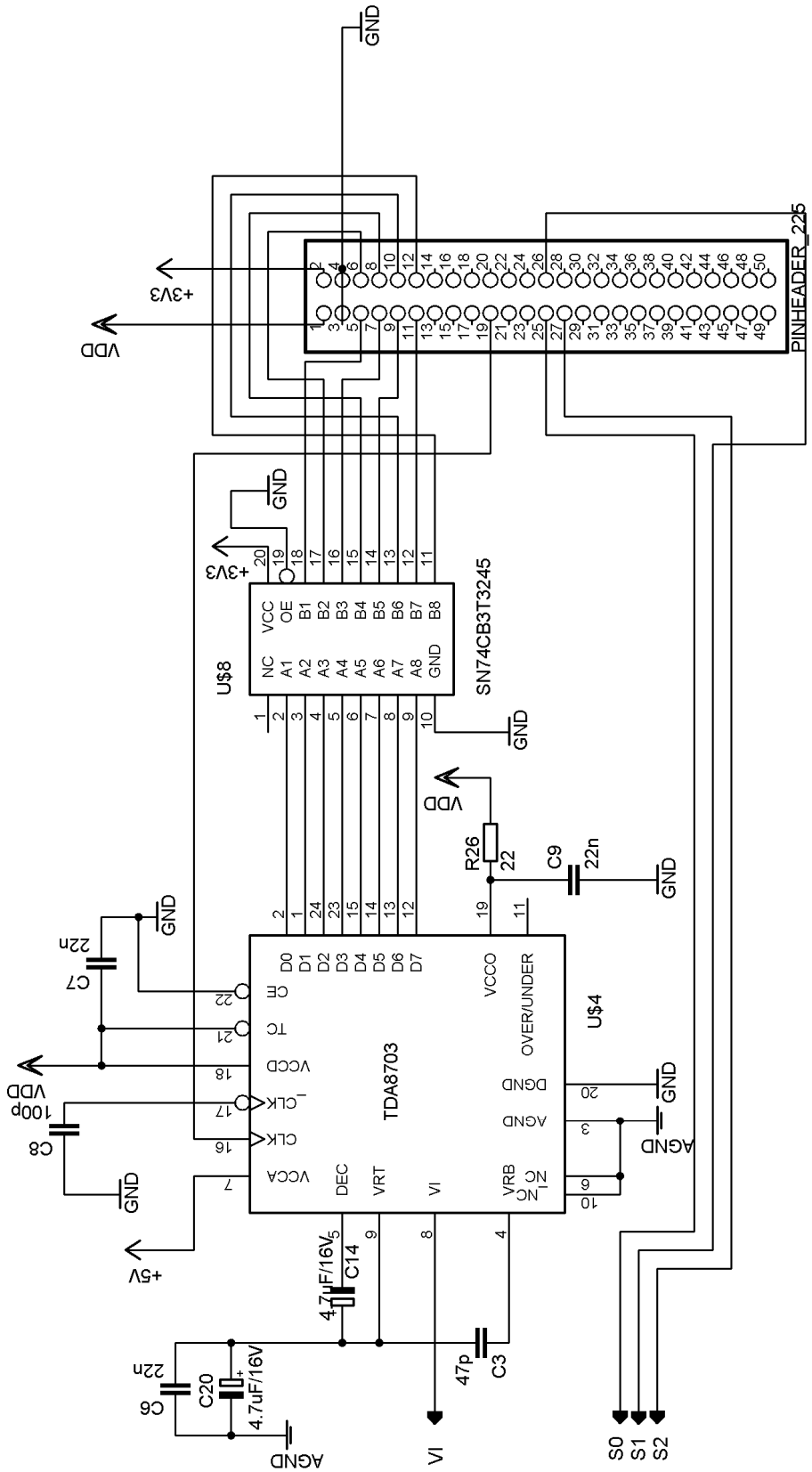


10-1 Osazený osciloskopický modul

Příloha C – Schémata zapojení



Analogová část schéma osciloskopického modulu



Digitální část schéma osciloskopického modulu

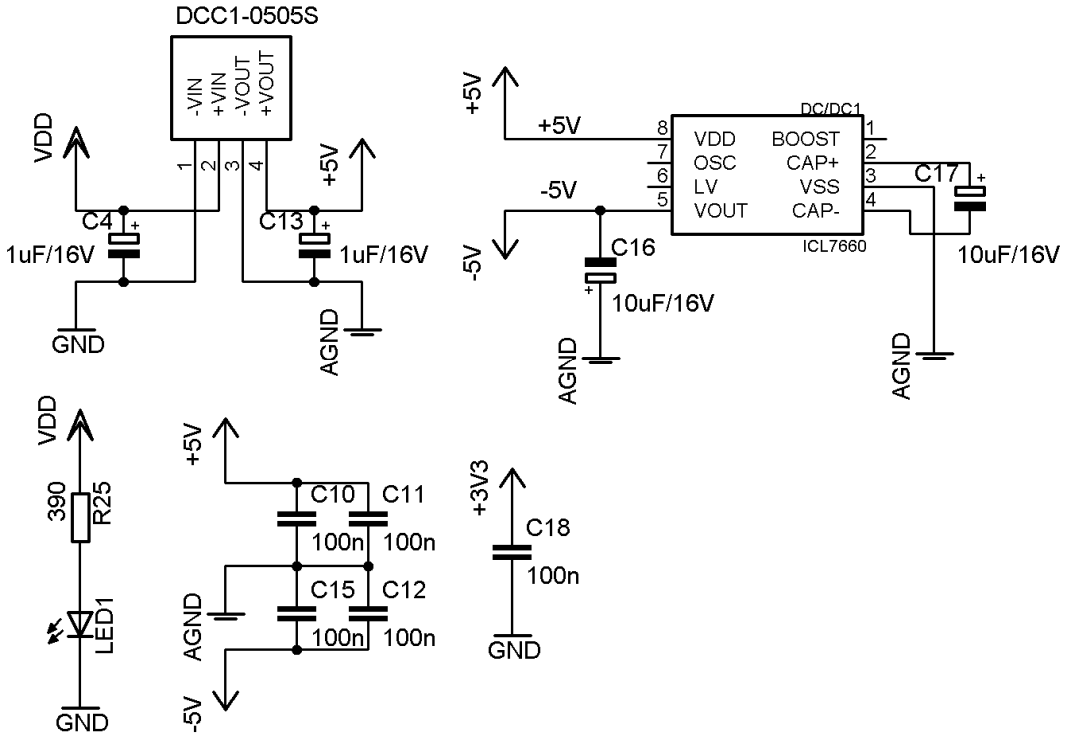


Schéma napájení osciloskopického modulu spolu s blokovacími kondenzátory

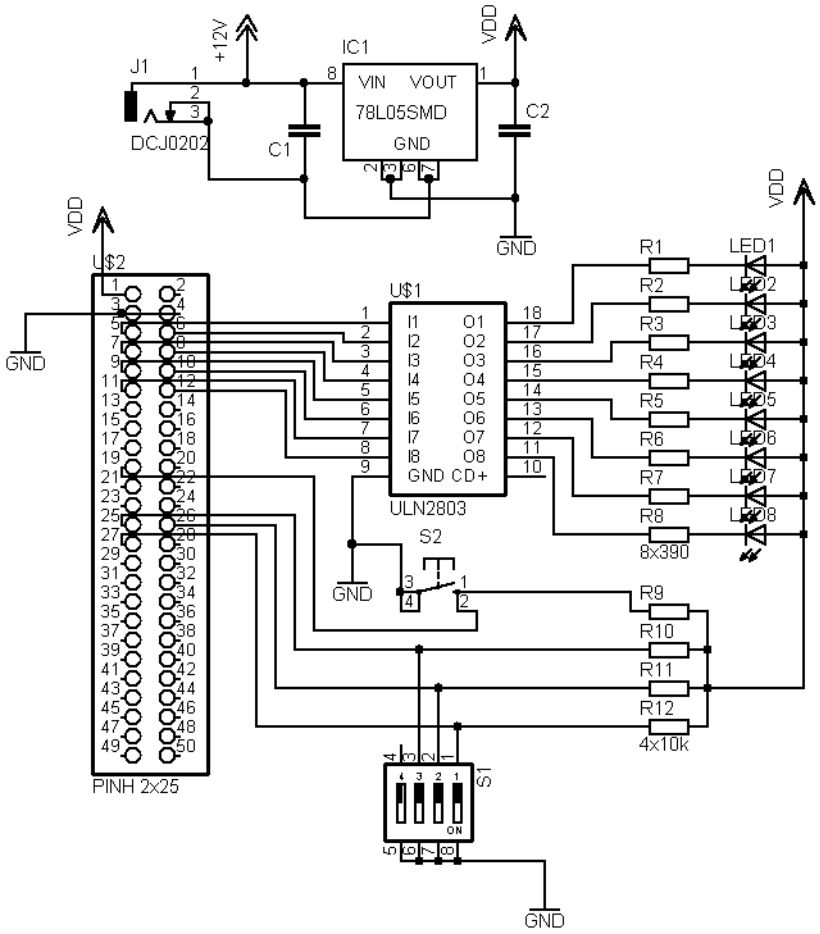
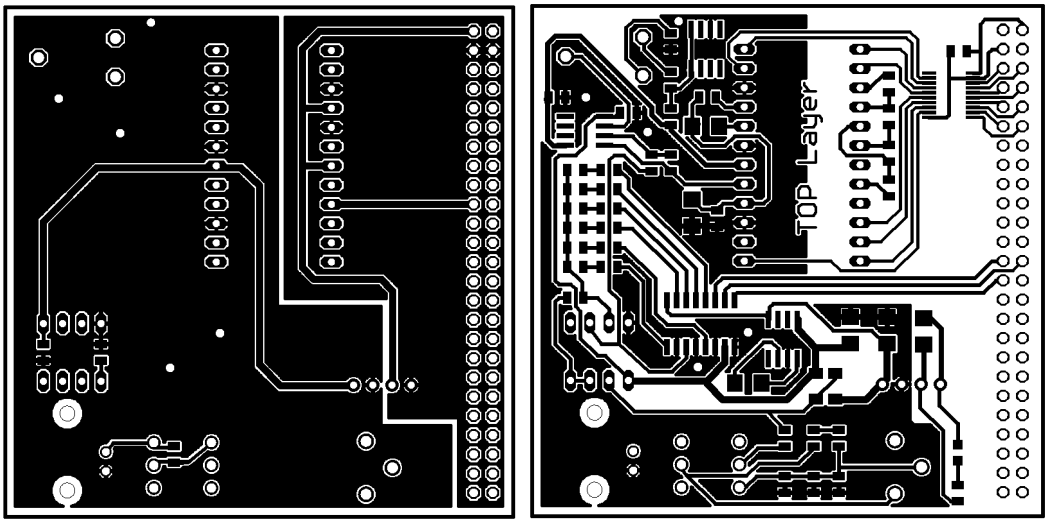


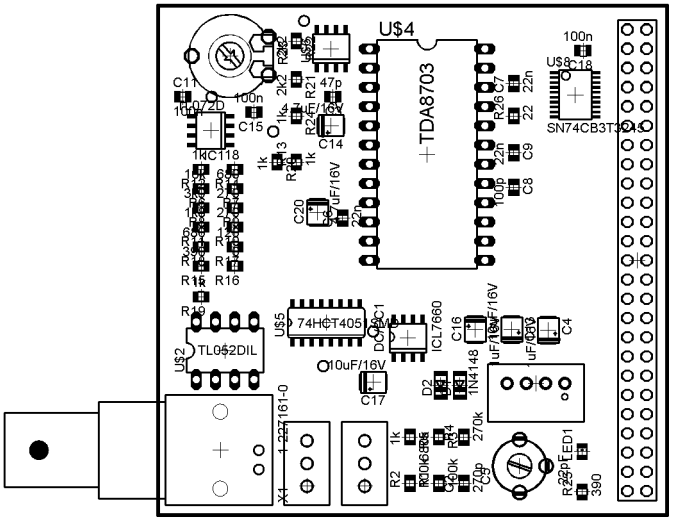
Schéma zapojení testovacího modulu

Příloha D – Návrh desky plošných spojů

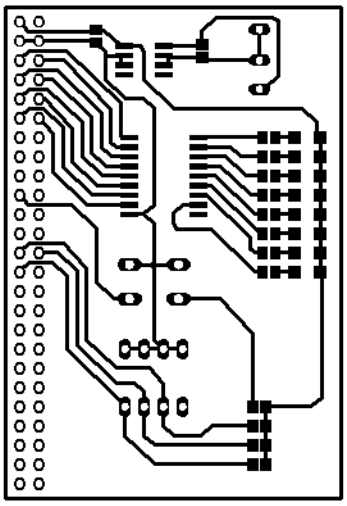
Horní a spodní strana spojů osciloskopického modulu

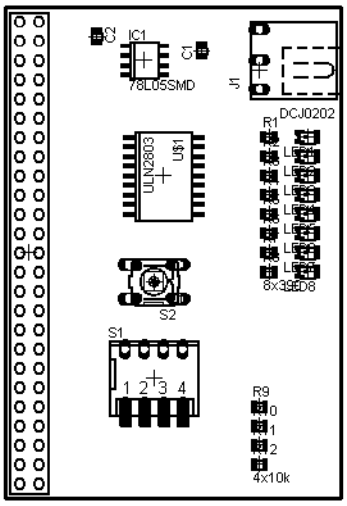


Rozmístění součástek osciloskopického modulu



Strana spojů a rozmístění součástek testovacího modulu





VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

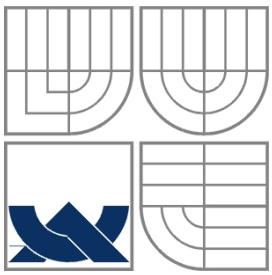
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DIGITÁLNÍ OSCILOSKOP NA PLATFORMĚ FITKIT

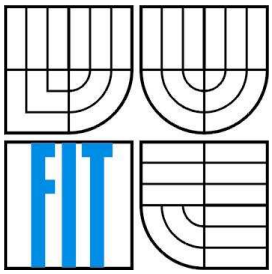
DIPLOMOVÝ PROJEKT
MASTER'S THESIS

AUTOR PRÁCE Bc. ONDŘEJ VEŠKRNA
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DIGITÁLNÍ OSCILOSKOP NA PLATFORMĚ FITKIT

DIGITAL OSCILLOSCOPE ON FITKIT PLATFORM

DIPLOMOVÝ PROJEKT
MASTER'S THESIS

AUTOR PRÁCE — Bc. Ondřej Veškrna
AUTHOR

VEDOUCÍ PRÁCE — Ing. Václav Šimek
SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2007/2008

Zadání diplomové práce

Řešitel: **Veškrna Ondřej, Bc.**
Obor: Počítačové systémy a sítě
Téma: **Digitální osciloskop na platformě FITkit**
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s výukovou platformou FITkit a jazykem VHDL pro návrh číslicového hardware.
2. Zpracujte krátkou studii ohledně problematiky převodu analogového signálu na číslicový.
3. Navrhněte koncepci osciloskopu na bázi FPGA a rychlých ADC převodníků v podobě rozšiřujícího modulu platformy FITkit.
4. V návrhovém prostředí Eagle nebo KiCAD vytvořte desku plošných spojů pro uvažovaný modul osciloskopu.
5. V jazyce VHDL vytvořte obvod pro řízení průběhu konverze analogového signálu a přenos získaných vzorků do PC.
6. Demonstrace funkčnosti vámi navrženého přípravku bude provedena formou zobrazení vzorků dat v jednoduché aplikaci.
7. Shrňte dosažené výsledky a diskutujte možnosti dalšího rozšíření.

Literatura:

- Pedroni, V. A. *Circuit Design with VHDL*, Massachusetts Institute of Technology, 2004, 376 s., ISBN 0-262-16224-5
- Noergaard, T. *Embedded Systems Architecture*, Elsevier, Burlington, 2005, 657 s., ISBN 0-7506-7782-9
- Ball, S. R. *Analog Interfacing to Embedded Microprocessor Systems*, Elsevier, New York, 2004, 335 s., ISBN 0-7596-7723-6
- Kester, W. *Data Conversion Handbook*, Elsevier, New York, 2005, 976 s., ISBN 0-7506-7841-0
- Baker, B. *A Baker's Dozen - Real Analog Solutions for Digital Designers*, Elsevier, New York, 2005, 362 s., ISBN 0-7506-7819-4

Při obhajobě semestrální části diplomového projektu je požadováno:

- Splnění bodů 1-3 zadání.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

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

Vedoucí: **Šimek Václav, Ing.**, UPSY FIT VUT
Datum zadání: 24. září 2007
Datum odevzdání: 19. května 2008

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

L. S.

Kotásek

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO
uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Bc. Ondřej Veškrna**
Id studenta: 89473
Bytem: Benetice 4, 675 07 Čechtín
Narozen: 19. 02. 1984, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

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

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

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

Název VŠKP: Digitální osciloskop na platformě FITkit
Vedoucí/školitel VŠKP: Šimek Václav, Ing.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: .....

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

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

Článek 2
Udělení licenčního oprávnění

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

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

Článek 3
Závěrečná ustanovení

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

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Práce se zabývá návrhem zařízení, které umožní sledovat průběh měřeného signálu na obrazovce počítače na principu digitálního osciloskopu. Řídicím prvkem zařízení je programovatelné hradlové pole osazené na platformě FITkit. Konfigurace hradlového pole má za úkol řídit vzorkování vstupního signálu a získané vzorky odesílat prostřednictvím USB rozhraní do počítače. Aplikace implementovaná na PC se pokusí signál rekonstruovat a výsledek zobrazí na monitoru počítače.

Klíčová slova

Platforma FITkit, FPGA, programovatelné hradlové pole, analogově číslicový převod, A/D převodník, vestavěný systém, mikrokontrolér, MCU, SPI, USB rozhraní, C#, VHDL

Abstract

This thesis deals with the design of a device that enables to monitor the behavior of the measured signal on the computer screen, using the principle of the digital oscilloscope. The control element of the device is the field programmable gate array (FPGA) on FITkit platform. The FPGA configuration controls the input signal sampling and sends the received samples through the USB interface to the PC. The graphical application implemented in the computer tries to restore the signal and then displays it on the screen.

Keywords

FITkit platform, FPGA, programmable gate array, analog to digital conversion, A/D converters, microprocessor, embedded system, MCU, SPI, USB interface, C#, VHDL

Citace

Veškrna Ondřej: Digitální osciloskop na platformě FITkit. Brno, 2008, diplomový projekt, FIT VUT v Brně

Digitální osciloskop na platformě FITkit

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením Ing. Václava Šimka
Další informace mi poskytli Dr. Ing. Otto Fučík, Ing. Tomáš Martínek a Ing. Zdeněk Vašíček.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Veškrna Ondřej
27.12.2007

Poděkování

Především bych rád poděkoval vedoucímu své diplomové práce panu Ing. Václavu Šimkovi za odborné vedení a čas věnovaný konzultaci této práce. Také bych chtěl poděkovat panu Ing. Tomáši Martínkovi a Ing. Zdeňkovi Vašíčkovi za odbornou konzultaci problému syntézy číslicových obvodů.

Obsah

1 Úvod

Osciloskop patří bezesporu mezi velice důležité měřící přístroje a stává se tak nenahraditelným pomocníkem v mnoha oborech lidské činnosti. Je nezastupitelný při mnoha měřeních, kdy jiná měřidla neuspějí, anebo jimi zjištěné údaje nemají větší význam, než čistě orientační.

Osciloskop je nepochybně velmi užitečný přístroj. Přes všechny jeho výhody však není mezi studenty či zájemci o elektroniku příliš rozšířen. Hlavním důvodem tohoto faktu je zajisté jeho vysoká cena, která dokáže odradit i většinu horlivých zájemců. Proto se začaly vyrábět osciloskopické adaptéry (buď jako externí přístroje, nebo karty do rozšiřujících slotů), které umožňují sledovat průběhy signálů na osobním počítači. Přístroj potom pouze vzorkuje vstupní signál a naměřené hodnoty následně odesílá přes vhodné rozhraní do počítače. O jejich zpracování a správné vykreslení se již postará příslušný software. Takové zařízení je již samozřejmě poměrně levnější, avšak i tak je jeho cena dost vysoká.

Při realizaci podobného zařízení jsou kladeny vysoké nároky nejen na rychlost a kvalitu analogově digitálních převodníků, rychlost řídicích obvodů, ale zejména je zapotřebí velice rychlých pamětí na uložení naměřených vzorků.

K řízení rychlých převodníků, ukládání vzorků a následnému přesunu vzorků do PC bude použita výuková platforma FITkit, přičemž její výkonná část – FPGA čip je pro tuto činnost vhodná. Výhodou FITkitu je jeho dostupnost každému studentovi Fakulty informačních technologií Vysokého učení technického v Brně.

Práce je logicky členěna do několika kapitol. V teoretické části se zabývám problematikou převodu analogového signálu na číslicový, následuje kapitola s přehledem A/D převodníků. Teoretickou část završuje seznámení s platformou FITkit.

V páté kapitole je proveden rozbor realizace daného problému včetně patřičných blokových schémat. Poslední kapitola je věnována závěru.

2 Převod analogového signálu na číslicový

Číslicové zařízení může zpracovávat signál jen v číslicovém tvaru. Většina přirozených zdrojů a často používaných snímačů jej však dodává ve tvaru analogovém. Pro jeho číslicové zpracování je tedy nezbytné převést tento signál na digitální [2].

Analogově číslicový převod je elektronický proces, ve kterém je spojitá veličina (analogový signál) změněna na diskrétní (víceúrovňový) digitální signál, bez změn jeho základního obsahu [3].

Převod analogového signálu na číslicový tvar se provádí ve dvou krocích. Analogový signál je nejprve periodicky vzorkován, tj. je získán periodický sled úzkých impulsů, který je amplitudově modulován přiváděným analogovým signálem. Ve druhém kroku jsou amplitudy jednotlivých impulsů převáděny na číslicový tvar, což nazýváme kvantování [2].

2.1 Vzorkování

Vzorkování je proces, v němž je signál souvislého času nahrazován jeho částmi - vzorky. Vzorky jsou od sebe zpravidla rovnoměrně vzdáleny (rovnoměrné vzorkování) o vzorkovací periodu T_V. Vzorkovací kmitočet (rychlost vzorkování) je pak převrácenou hodnotou vzorkovací periody [2]:

$$f_v = \frac{1}{T_V}$$

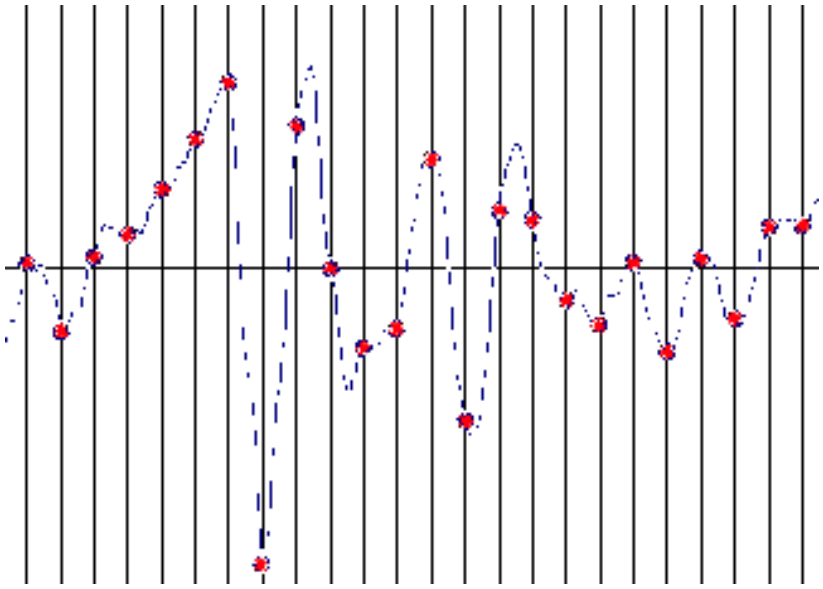
Rovnoměrné vzorkování lze chápat jako násobení signálu souvislého času periodickým vzorkovacím signálem. Pro zjednodušení výpočtů zavádíme ideální vzorkování, při němž je vzorkovacím signálem posloupnost Diracových impulsů. Reálné vzorkování je realizováno spínačem, který je po dobu odběru vzorku sepnut, jinak rozepnut. Obrázek 2-2 znázorňuje, jak může vypadat vzorkovaný signál. Znázorněná kolečka naznačují jednotlivé vzorky.

Důležitou vlastností vzorkovaného signálu je periodicita jeho spektra. Díky této periodicitě vzniká při vzorkování jisté nebezpečí nevratné ztráty informace v důsledku překrytí spekter dvou sousedních period. Aby k tomuto překrytí nemohlo dojít, je třeba, aby byl splněn vzorkovací (Shannonův-Nyquistův-Kotělnikův) teorém, který říká [4]:

„Přesná rekonstrukce spojitého, frekvenčně omezeného, signálu z jeho vzorků je možná tehdy, pokud byl vzorkován frekvencí alespoň dvakrát vyšší než je maximální frekvence rekonstruovaného signálu.“

$$f_v > 2 \cdot f_{\max},$$

kde f_{max} je maximální frekvenční složka spektra původního spojitého signálu. Tato podmínka se v praxi zajišťuje tzv. antialiasingovým filtrem typu dolní propust s mezním kmitočtem do poloviny maximálního možného kmitočtu zdrojového signálu. Tento filtr se zařadí mezi zdroj signálu a zařízení, které signál vzorkuje [2].



Obrázek 2-1 Vzorkování spojitého signálu

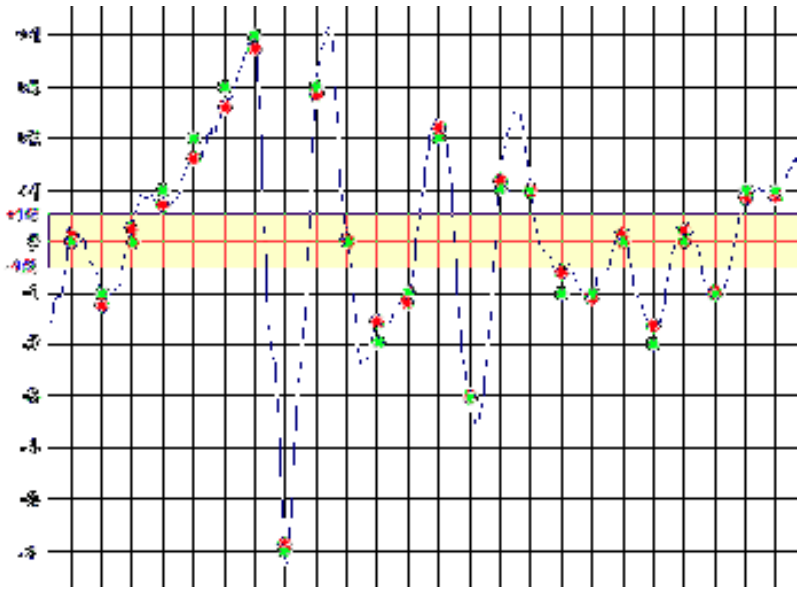
2.2 Kvantování

Kvantováním se mění signál se spojitou množinou hodnot na signál s diskrétní množinou hodnot. Každý vzorek je pak vyjádřen konečným počtem cifer (zaokrouhlování nebo usekávání) a toto číslo je vyjádřeno vhodným číselným kódem. Nejmenší možný skok kvantovaného signálu se nazývá *kvantovací krok* a rozdíl mezi vstupním a výstupním signálem kvantizačního obvodu *kvantizační šum r(t)* [2]. Kvantování vzorků signálu znázorňuje obrázek 2-3.

Při kvantování tedy dochází ke ztrátě informace. Stupeň narušení původního signálu můžeme popsat činitelem kvantizačního zkreslení k, který je definován jako poměr efektivní hodnoty R_{ef} kvantovacího šumu a efektivní hodnoty S_{ef} užitečného signálu. Podle literatury [5] lze činitel kvantovacího zkreslení vyjádřit vztahem:

$$k = \frac{R_{ef}}{S_{ef}} = \sqrt{\frac{2}{3}} \cdot \frac{1}{n},$$

kde n je počet hladin, které protne kvantovaný signál.



Obrázek 2-2 Kvantování vzorků signálu

3 Analogově číslicové převodníky

Analogově digitální převodník (zkratky A/D, v angličtině i ADC) je elektronická součástka určená pro převod spojitého (neboli analogového) signálu na signál diskrétní (neboli digitální). Důvodem tohoto převodu je umožnění zpracování původně analogového signálu na digitálních počítačích [6].

Číslicové výstupy převodníků používají různé kódování. Mezi nejběžněji používané kódy patří přímý dvojkový kód, inverzní kód, doplňkový kód, posunutý kód a kód BCD [7].

3.1 Statické a dynamické vlastnosti převodníků

Statické parametry převodníků jsou určovány pomocí převodní charakteristiky, zatím co dynamické vlastnosti se vyhodnocují z kmitočtového spektra převodníku.

3.2 Parametry A-D převodníků

Mezi základní statické parametry patří [7]

- rychlost převodu,
- rozlišení převodníku (resolution),
- přesnost (accuracy),
- chyba nastavení nuly (offset error),
- hystereze a další.

K hlavním dynamickým parametrům patří

- odstup signál-šum (signal to noise ratio - SNR),
- efektivní počet bitů (effective number of bits - ENOB),
- dynamický rozsah bez parazitních složek (spurious free dynamic range - SFDR),
- krátké přechodové špičky (glitches),
- šum - vrcholový, efektivní (noise - RMS, peak),
- doba přepnutí a ustálení a další.

3.2.1 Rychlost vzorkování

Rychlost vzorkování vstupního signálu patří mezi nejvýznamnější parametry A/D převodníků. Musí být dostatečně vysoká vzhledem k nejvyšší kmitočtové složce vstupního analogového napětí – jak již bylo uvedeno, je nutné přenést více jak dva body amplitudy nejvyšší kmitočtové složky sledovaného

signálu. Pokud nás naopak některá vyšší harmonická složka nezajímá nebo způsobuje chybu v následném zpracování dat (např. šumový signál), je možno ji odstranit vhodnou dolní propustí [8].

3.2.2 Rychlost převodu

Rychlost převodu je u A/D převodníků obvykle shodná s rychlostí vzorkování, resp. naopak, rychlost vzorkování vyplývá z nejkratší možné doby převodu. Doba převodu může být určena jako doba, která uplyne od okamžiku přivedení vstupního analogového napětí na vstup převodníku, až do doby, kdy je výstupu převodníku k dispozici platné výstupní datové slovo. Může být rovněž vyjádřena počtem úplných převodů za jednotku času nebo počtem bitů za jednotku času [8].

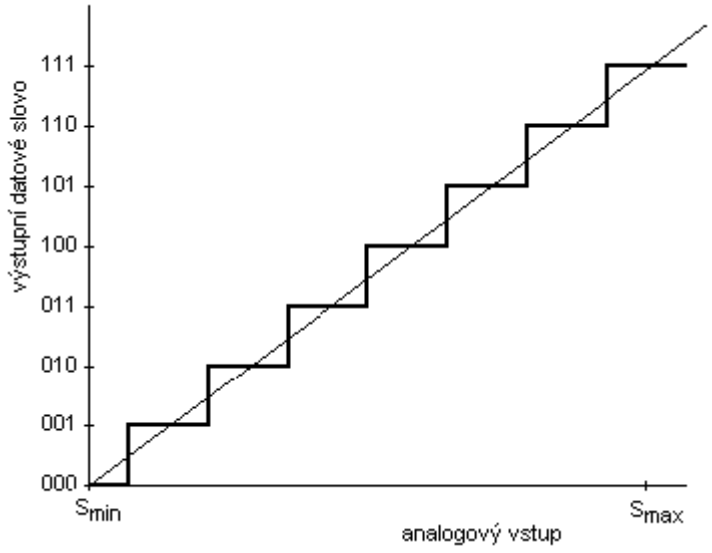
3.2.3 Rozlišení A/D převodníků

Rozlišovací schopnost převodníku AD je určena počtem úrovní, do kterého je rozdělen rozsah vstupního analogového signálu. Jelikož výstupní slovo převodníku vyjadřuje obvykle číslo v binárním kódu, je často rozlišovací schopnost vyjadřována počtem bitů výstupního slova. Sestává-li např. výstupní slovo z 10 bitů, pak vstupní rozsah je rozdělen na 1024 diskrétních úrovní. Rozlišovací schopnost je $1/2^N$, kde N je počet bitů výstupního slova. Je třeba si uvědomit, že čím je větší rozlišovací schopnost, tím je nižší rychlost převodu [8].

3.2.4 Kvantizační chyba převodníků

Vstupní analogový signál, který může nabývat libovolné úrovně v mezích vstupního rozsahu, je tedy kvantován do určitého počtu diskrétních úrovní (kvantizačních úrovní) [7].

Tímto procesem může vzniknout *kvantizační chyba*. Kvantizační chybu lze zmenšit pouze použitím více diskrétních úrovní, tj. použitím více bitů výstupního slova.



Obrázek 3-1 Převodní charakteristika A/D převodníku

Na obr. 3-1 je znázorněna ideální převodní charakteristika A/D převodníku. Schodovitý průběh převodní charakteristiky způsobuje odchylku od ideálního průběhu a projevuje se jako **kvantizační šum SNR** (Signal-to-Noise Ratio). Pro sinusový signál je teoretické SNR dáno vztahem [8]:

$$SRN = 6.02 \cdot n + 1.76[dB],$$

kde *n* je počet bitů datového slova (rozlišení).

Vlivem chyb převodníku je však skutečné SNR odlišné od ideálního, a proto pro porovnání kvality A/D převodníků zavádíme pojem **efektivní počet bitů ENOB** (Effective Number Of Bits)[8]:

$$ENOB = \frac{SNR - 1.76}{6.02} \leq n$$

V praxi se reálná převodní charakteristika liší od ideální vlivem napěťového posunu (označované též *chyba nuly či ofset*), změnou zisku (*chybou rozsahu*) nebo *nelinearitou převodníku*. Celková přesnost převodníku je pochopitelně také podstatně závislá na stabilitě zdroje referenčního napětí [8].

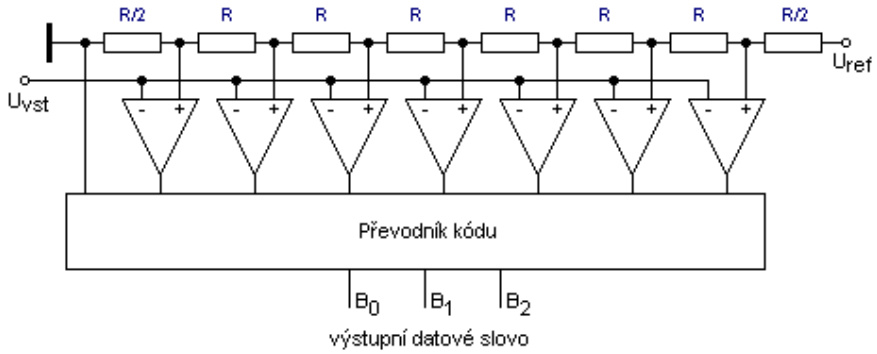
3.3 Typy A/D převodníků

A/D převodníky můžeme dělit podle různých kritérií. Podle způsobu činnosti dělíme převodníky na *synchronní* a *asynchronní*. U synchronních převodníků probíhá převod analogového napětí na výstupní datové slovo v určitém počtu kroků, které se uskutečňují synchronně s hodinovými (taktovacími) impulsy, u asynchronních převodníků může být převod rovněž uskutečněn v několika krocích, ovšem doba trvání těchto kroků závisí výhradně na časové odezvě dílčích obvodů převodníku a na jejich zpoždění [8].

Dále můžeme A/D převodníky dělit podle vstupního signálu na *přímé* a *nepřímé*. Přímé převodníky převádějí přímo vstupní analogové napětí na výstupní slovo, u nepřímých převodníků se vstupní analogové napětí nejprve převádí určitým obvodem na jinou analogovou veličinu (např. na dobu trvání impulsu) a dalším obvodem je teprve tato veličina převedena na výstupní datové slovo [8].

3.3.1 Paralelní A/D převodník

Paralelní A/D převodník je nejrychlejším a současně principiálně nejjednodušším typem přímého A/D převodníku. Princip převodníku je znázorněn na obrázku 3-2. Vstupní analogové napětí je přiváděno současně na vstupy soustavy napěťových komparátorů. Na těchto komparátorech se toto napětí porovnává s určitým referenčním napětím U_{refi} a výstup jednotlivých komparátorů se překlápí v případě, že $U_{vst} \geq U_{refi}$. Převodník kódu pak převede výstupy z napěťových komparátorů na výstupní datové slovo [8].

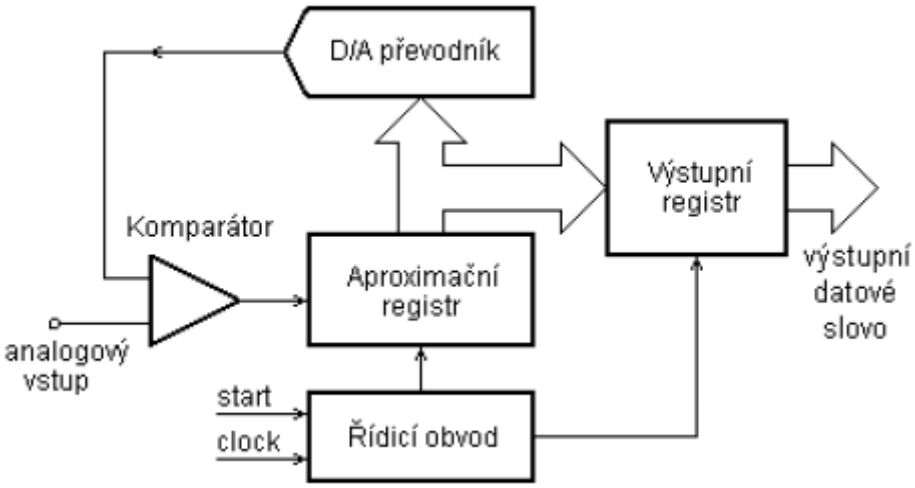


Obrázek 3-2 Tříbitový paralelní A/D převodník

Doba převodu paralelního převodníku je určena přenosovým zpožděním, resp. dobou ustálení napěťových komparátorů a přenosovým zpožděním v převodníku kódu. Převodníky tohoto typu jsou rychlé, ale nákladné, protože obsahují velký počet napěťových komparátorů [8].

3.3.2 Převodník s postupnou aproximací

A/D převodník s postupnou aproximací realizuje převod vstupního analogového napětí na výstupní datové slovo postupně po krocích, jejichž počet je roven počtu bitů výstupního datového slova. Blokové schéma A/D převodníku je na obrázku 3-3 [8].



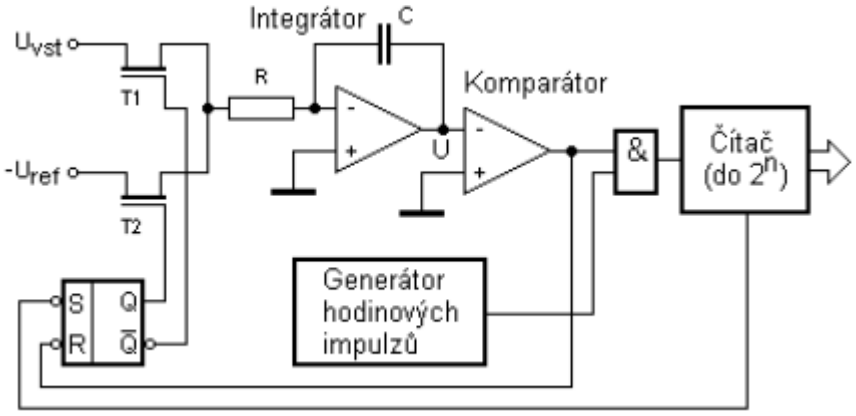
Obrázek 3-3 D/A převodník s postupnou aproximací

Tento převodník má integrovaný D/A převodník, napěťový komparátor, aproximační registr a výstupní registr. Převod se provádí postupně, od nejvyššího bitu směrem k nižším metodou půlení intervalu. Řídicí obvod převodníku nastaví hodnotu testovaného bitu (testované napěťové úrovně) na hodnotu 1, D/A převodníkem je generováno příslušné referenční napětí a napěťový komparátor porovná toto napětí se vstupním napětím. Je-li vstupní napětí větší než referenční, zůstane v příslušném bitu datového slova v aproximačním registru uchována jednička, v opačném případě se na toto místo dosadí nula. Převod pak pokračuje nastavením následujícího (nižšího) bitu datového

slova a stejný postup se opakuje. Nevýhodou převodníku je celková doba převodu, která je přímo úměrná počtu bitů výstupního datového slova [8].

3.3.3 A/D převodník s dvojitou integrací

A/D převodník s dvojitou integrací je příkladem nepřímého převodníku, u kterého je vstupní analogové napětí nejdříve převedeno na dobu trvání určitého elektrického signálu a velikost vstupního napětí je určována podle hodnoty slova v čítači, který je tímto napětím řízen. Schéma zapojení tohoto převodníku je na obrázku 3-4 [8].



Obrázek 3-4 Převodník s dvojitou integrací

Tento převodník je možno charakterizovat poměrně malou rychlostí převodu, značnou dosažitelnou přesností a obvodovou jednoduchostí bez větších nároků na přesnost většiny prvků [8].

3.3.4 A/D převodníky typu sigma-delta

V současné době se velice rozšířily A/D převodníky typu sigma-delta. Jádrem tohoto synchronního převodníku je opět integrátor a komparátor, který generuje sled pulzů, jejichž střední hodnota počtu za určitý interval odpovídá vstupnímu napětí. Střední hodnota se vytváří v číslicovém filtru

Nejdůležitější parametry všech A/D převodníků jsou uvedeny v tabulce 3-1 [8].

Typ	Rozlišení [bit]	Rychlost převodu [Hz]
Paralelní	6 ... 10	10^7 … 3.10^9
Aproximační	8 ... 16	3.10^4 … 3.10^6
Integrační	10... 27	10^{-1} … 10^3
Sigma-delta	16 … 24	10^1 … 10^5

Tabulka 3-1 Parametry A/D převodníků

4 Platforma FITkit

Výuková platforma FITkit byla vyvinuta na Fakultě informačních technologii Vysokého učení technického v Brně. Platforma FITkit umožňuje obsáhnout značnou část spektra znalostí a dovedností, které musí dnešní inženýr – informatik znát, aby byl schopen obstát na globálním trhu práce.

Obrázek 4-1 Platforma FITkit

Typickým příkladem využití informatiky v praxi jsou tzv. vestavěné systémy (anglicky Embedded Systems), které se v dnešní době dominantně uplatňují v běžném životě a jejichž význam ještě výrazně poroste. Jednoduše řečeno se jedná o veškerá zařízení, která v sobě mají nějakým způsobem vestavěn počítač (mobilní telefon, MP3 přehrávač, televizní přijímač atd.).

FITkit obsahuje výkonný mikrokontrolér s nízkým příkonem a řadu periferií. Důležitým aspektem je též využití pokročilého re-programovatelného hardwaru na bázi hradlových polí FPGA (anglicky Filed Programmable Gate Array), jenž lze, podobně jako software na počítači, neomezeně modifikovat pro různé účely dle potřeby – uživatel tedy nemusí vytvářet nový hardware pro každou aplikaci znovu [9].

Při návrhu aplikací se využívá skutečnosti, že vlastnosti hardwaru se v dnešní době převážně popisují vhodným programovacím jazykem (např. VHDL), díky čemuž se návrh softwaru a hardwaru provádí do značné míry obdobně.

Software po mikrokontrolér se tvoří v jazyce C a do spustitelné formy se překládá pomocí GNU překladače. Generování programovacích dat pro FPGA z popisu v jazyce VHDL probíhá zcela automaticky pomocí profesionálních návrhových systémů [9].

Uvnitř mikrokontroléru je umístěn kód, který umožňuje naprogramovat čip přes sériový kanál bez použití dalších externích prostředků a za pomoci volně dostupných programů, anebo lze použít externí programátor, který se napojuje na JTAG (Joint Test Action Group) rozhraní mikrokontroléru.

FPGA lze obdobně naprogramovat přes JTAG anebo za pomoci MCU skrze vysokorychlostní komunikaci protokolem SPI.

FITkit komunikuje s vnějším okolím přes USB rozhraní, ze kterého je přímo napájen. V případě velkého proudového odběru je nutné připojit externí napájení +5V do konektoru JP8. Externí napájení je chráněno pojistkou F0 o hodnotě 1A. Z napětí +5V se dále stabilizátory získává pro většinu komponent napětí +3.3V a pomocná napájení +2.5V a +1.2V pro napájení FPGA[10].

Iniciativa je koncipována jako open-source (pro software) a open-core (pro hardware), což znamená, že veškeré výsledky práce studentů s platformou FITkit jsou přístupné na internetu ve zdrojové formě pro kohokoli. Veškerý návrhový software ke generování programovacích dat pro FPGA, včetně rozsáhlé dokumentace, je k dispozici zdarma na internetových stránkách firmy Xilinx Inc. [11].

4.1 Hlavní části platformy FITkit

Výuková platforma FITkit obsahuje tyto prvky:

- FPGA Spartan 3 XC3S50-4PQ208C (Xilinx)
- MCU MSP430F168 (Texas Instruments)
- USB-UART převodník FT2232C
- Audio IN / OUT
- Konektory PS2
- Konektor VGA
- Konektor RS232
- DRAM 8x8mbit
- Klávesnice
- Řádkový LCD displej

FITkit disponuje dvěma programovatelnými jednotkami, a sice mikroprocesorem MCU řady MSP430 a FPGA řady Spartan3. Obě jednotky je možné programovat pomocí kabelu s rozhraním JTAG. Nicméně MCU lze programovat i přímo přes USB kabel. Pro nahrání konfigurace do FPGA čipu lze použít naprogramovaný MCU. Více o možnostech programování obou jednotek bude popsáno v kapitole 4.4.

4.1.1 FPGA

Na kitu je osazeno programovatelné hradlové pole XC3S50-4PQ208C řady Spartan 3 firmy Xilinx.

Mezi hlavní rysy tohoto čipu patří [9]:

- 192 konfigurovatelných logických bloků (CLBs) uspořádaných do matice o 16 řádcích a 12 sloupcích
- 1728 logických buněk
- 50000 logických hradel
- Paměť RAM
    - 12 Kbitů distribuovaných, 72 Kbitů v jednom bloku
- 2 jednotky pro správu vstupního hodinového signálu (DCMs - Digital Clock Managers)
- 4 násobičky 18x18 bitů

Pro komunikaci s okolím obsahuje čip až 124 uživatelských vstupů/výstupů (I/O), z nichž 45 pinů je sdruženo ve sběrnici X[0..45]. Tato sběrnice je vyvedena přímo na 50 pinový dvouřadý konektor JP10. Konektor se nachází v dolní části kitu. Rozteč jeho pinů je klasických 2.54 mm. Na zbývající piny konektoru je přivedeno i napájecí napětí +3.3V a +5V [4].

FPGA může také komunikovat s MCU pomocí SPI (Serial Peripheral Interface) anebo pomocí společné 8bitové sběrnice, označené jako P3M [10].

4.1.2 MCU

Dalším ústředním prvkem na kitu je mikrokontrolér MSP430F168 firmy Texas Instruments. Jedná se o nízko-příkonový mikrokontrolér s těmito parametry [9]:

- Nízký příkon
    - 330 µA v aktivním režimu při 1 Mhz a 2.2 V
    - 1.1 µA ve stand-by režimu
    - 0.2 µA v režimu vypnuto
- 16-bitová RISC architektura
- Instrukční cyklus 125 ns
- Paměť - Flash 48 KB + 256 B
- RAM 10 KB
- 16-bitové časovače
- Sériové rozhraní
- 12-bitové A/D a D/A převodníky

Vstupně výstupní porty jsou označeny jako P1-P6. Jim přísluší označení sběrnic P1M-P6M. Většina signálů z těchto sběrnic je vyvedena na konektor JP9, což je 40-ti pinový konektor s roztečí 2.54mm. Na tomto konektoru je také vyvedeno napájecí napětí +3.3V a +5V.

Port P5 se používá pro SPI komunikaci mezi FPGA a FLASH pamětí. Dále při programování FPGA a vysílání hodinových signálů SMCLK a ACLK.

Programování MCU lze provést dvěma způsoby. První přes JTAG rozhraní, připojené přes konektor JP11 nebo přes USB/UART, za použití rutiny označené jako Bootloader. Bootloader je krátký programový kód, který umožňuje přijímat data přes sériovou linku a zapisovat je do vnitřní FLASH paměti. Tento kód je výrobcem napevno uložen v obvodu MCU [10].

Na sběrnici P6M je vyvedeno 8 kanálů 12-ti bitových A/D převodníků a dva 12-ti bitové D/A převodníky. Parametry vestavěných A/D převodníků však nejsou pro tento projekt vhodné, proto je zapotřebí k FITkitu připojit externí modul s výkonnějšími převodníky.

4.2 Komunikace s FITkitem

Hlavním vnějším komunikačním prostředkem kitu je univerzální vysoko-rychlostní asynchronní přijímač a vysílač (UART) FT2232C od firmy FTDI Chip[12]. Tento čip převádí rozhraní USB ze strany PC na dva nezávislé konfigurovatelné UART/FIFO kanály.

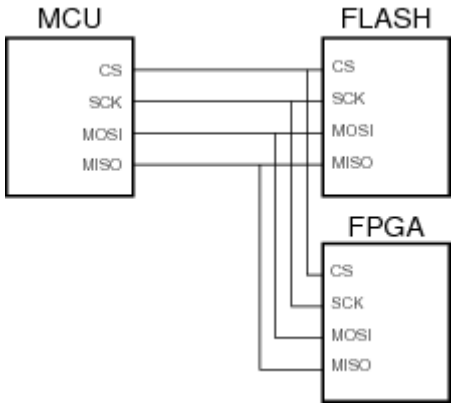
K PC se FITkit připojuje pomocí USB kabelu typu A-B. Po připojení a nainstalování potřebných ovladačů se oba komunikační kanály jeví jako nové sériové porty COM [10]. Jeden z těchto asynchronních kanálů je připojen k FPGA a druhý k mikrokontroléru. Díky této skutečnosti degraduje komunikace s prvky FITkitu na komunikaci po sériové lince. Oba komunikační kanály jsou schopny pracovat rychlostí až 921600 baudů/sec.

Pro komunikaci s mikrokontrolérem je vhodné využít terminálový program, který lze využít jak k ovládání spuštěných aplikací, tak k přenášení binárních dat prostřednictvím protokolu X-modem. Tímto způsobem lze nahrát i konfiguraci do FPGA čipu, o tomto způsobu však bude napsáno v kapitole 4.4. Komunikace probíhá rychlostí až 460800 baudů/sec.

4.3 Sběrnice SPI

Rozhraní SPI (Synchronous Peripheral interface) je sériové vysokorychlostní rozhraní typu master-slave. Rozhraní umožňuje obousměrnou komunikaci řízenou tzv. master (nadřazeným) zařízením s libovolným počtem slave zařízení, které jsou k této sběrnici připojeny [9].

SPI na fitkitu umožňuje procesoru, který je nakonfigurován v master režimu, komunikovat jednak s pamětí FLASH, uchovávající program pro MCU, a konfiguraci FPGA, ale také s periferiemi realizovanými uvnitř FPGA. Typicky se např. SPI používá ke komunikaci s řadiči periferií, pamětí RAM, apod. Slave zařízení FLASH a FPGA sdílejí veškeré signály (Obrázek 4-2) a rozlišení adresovaného zařízení se provádí na úrovní přenášených bitů. K zajištění bezchybné komunikace, musí mít FPGA i FLASH vzájemně disjunktní množinu operačních kódů [9].



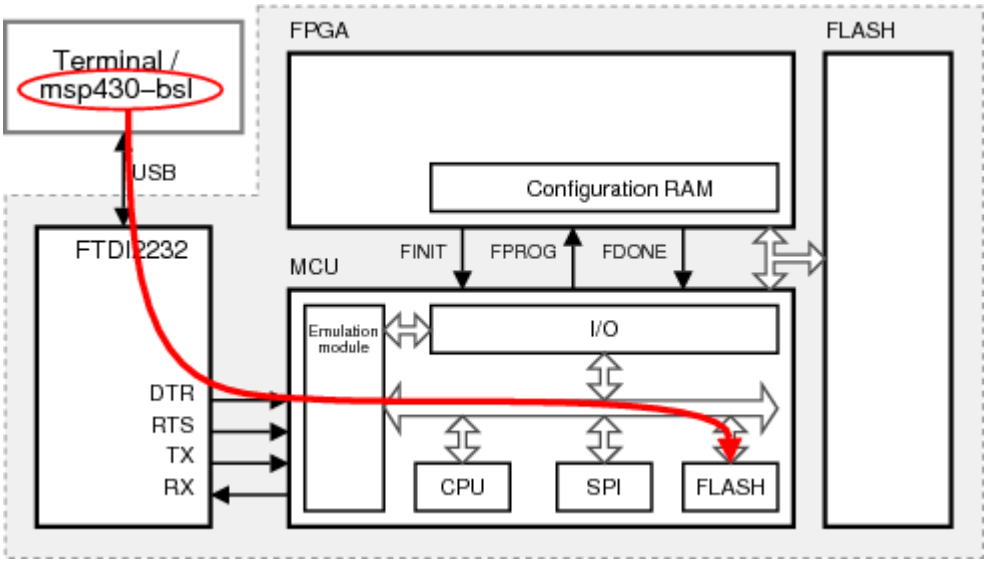
Obrázek 4-2 SPI na FITkitu

4.4 Způsob programování na FITkitu

4.4.1 Programování MCU

Naprogramování mikrokontroléru lze provést dvěma způsoby. Prvním způsobem je naprogramování MCU prostřednictvím rozhraní JTAG. K tomuto účelu lze využít speciální programátor MSP-FET430 firmy Texas Instruments [13]. Výhoda tohoto řešení spočívá v možnosti ladit kód za běhu procesoru. Vhodným vývojovým prostředím pro tento typ programování je například IAR, jehož omezenou distribuci je možné volně stáhnout na stránkách společnosti [13].

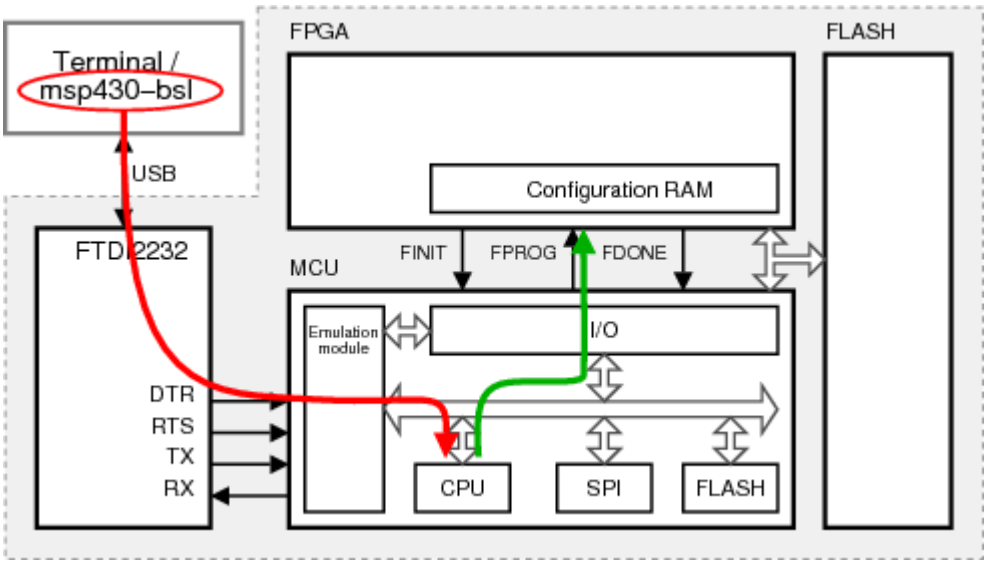
Druhý způsob využívá k naprogramování mikrokontroléru sériové rozhraní. Takovéto programování umožňuje tzv. „bootloader“, což je program umístěný výrobcem v mikroprocesoru. Bootloader je schopný nahrát program mikroprocesoru do jeho FLASH paměti. Nevýhodou této možnosti je, že není možné kód programu za běhu ladit nebo modifikovat, jako tomu bylo v případě programování přes rozhraní JTAG. K programování po sériové lince je možné využít konzolové aplikace msp430gdb-debug, která je v plné verzi volně dostupná v programovém balíku vývojového prostředí mspgcc[14]. FLASH paměť mikrokontroléru, kde je uložen programový kód, zůstává perzistentní i po odpojení napájecího napětí.



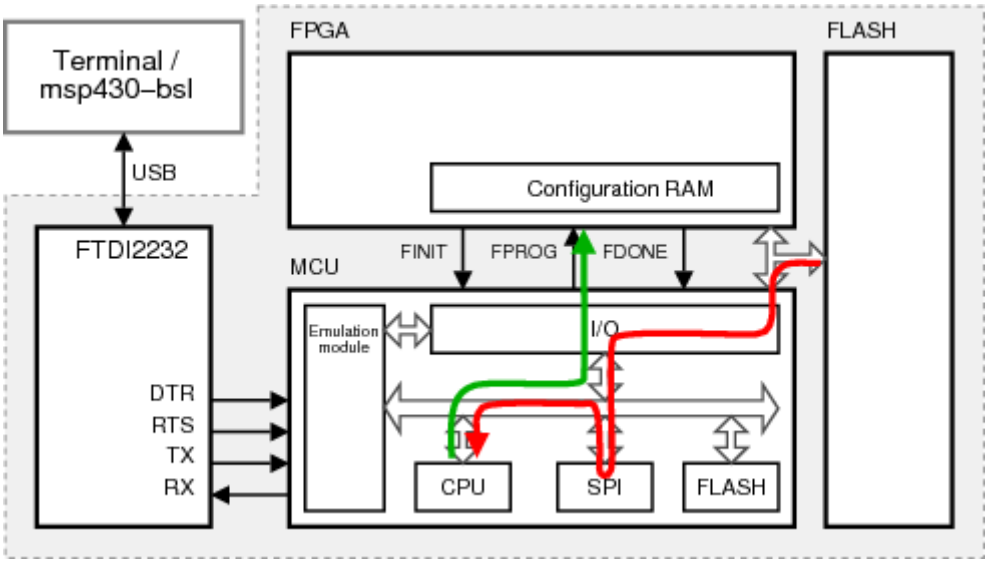
Obrázek 4-3 Programování MCU

4.4.2 Programování FPGA

Konfiguraci pro FPGA je narozdíl od MCU nutné po každém odpojení napájení přeprogramovat. Programování FPGA lze provádět buď podobně jako MCU pomocí rozhraní JTAG, nebo prostřednictvím rozhraní SPI. Druhý způsob programování zajišťuje mikrokontrolér. Ten může novou konfiguraci pro FPGA nahrát přes USB rozhraní přímo z počítače nebo ji nahraje z paměti FLASH umístěné na kitu. Pro přenos konfiguračního souboru z PC se používá terminál pro sériovou komunikaci pomocí protokolu XModem v režimu 1KCRC.



Obrázek 4-4 Nahrání konfigurace FPGA přímo do FPGA



Obrázek 4-5 Nahrání konfigurace FPGA z externí FLASH paměti

4.4.3 Překladový systém FITkitu

Pro usnadnění překladu, syntézy a programování byl v projektu FITkit vytvořen jednoduchý překladový systém, který využívá standardní GNU Makefile soubory. Protože byl kladen důraz na jednoduché použití, obsahují Makefile soubory (jenž jsou součástí projektu) pouze nezbytné informace potřebné k překladu projektu. Tzn. seznam všech zdrojových souborů a knihoven, které projekt pro překlad potřebuje. Soubor překladových pravidel (společný pro všechny projekty) je umístěn v adresáři *base*. Makefile u jednotlivých projektů se na tento soubor odkazují. Parametry, které se mohou lišit v závislosti na konfiguraci systému, jsou umístěny v souboru *base/settings.inc* [9].

Překladový systém pro překlad softwaru nabízí několik příkazů:

- **gmake** – pro kompilaci kódu. Výstupem je soubor (.hex), který je možné použít k naprogramování MCU. Dalším výstupním souborem je mapa paměti (přípona .map)
- **gmake lst** – vytvoření výpisu assembleru. (Soubor s příponou .lst)
- **gmake load** – slouží k naprogramování procesoru binárním souborem s příponou .hex. Pokud tento soubor dosud neexistuje, vytvoří se.
- **gmake clean** – odstraní všechny soubory vzniklé při kompilaci softwaru.
- **gmake cleanlib** - odstraní knihovnu funkcí libfitkit a veškeré soubory vzniklé během její kompilace. Tento příkaz je nutné použít vždy, když provedeme změnu v souborech knihovny.

Podobná struktura jako pro překlad softwaru je použita i pro syntézu VHDL kódu. Soubory Makefile pro syntézu VHDL kódů obsahují všechny potřebné zdrojové soubory většinou v podobě balíčků (*.inc* soubory). Tyto balíčky zapouzdřují cesty k jednotlivým VHDL souborům. Výhodou je

možnost dynamicky měnit seznam zdrojových souborů bez nutnosti zasahovat do již napsaných aplikací, které tyto soubory využívají.

Seznam příkazů týkajících se syntézy hardware [9]:

- **gmake** – vyvolá syntézu zdrojových kódů a vygeneruje binární soubor „output.bin“ obsahující konfiguraci, kterou je možné naprogramovat FPGA obvod.
- **gmake synth** – provede se pouze syntéza zdrojových kódů. Další fáze k vytvoření konfigurace se již neprovedou. Tato možnost je vhodná při ladění VHDL kódů.
- **gmake simmodel** – vygeneruje simulační model „post place&route“, který se používá pro časovou simulaci.
- **gmake rtl** – vygeneruje RTL schéma s příponou *.ngr*
- **gmake sim** – zkompiluje všechny zdrojové soubory, vytvoří simulační knihovnu *sim/work* a spustí skript *sim.fdo* v adresáři *sim.*
- **gmake cleam** – odstraní všechny soubory vzniklé při syntéze a generování konfigurace.

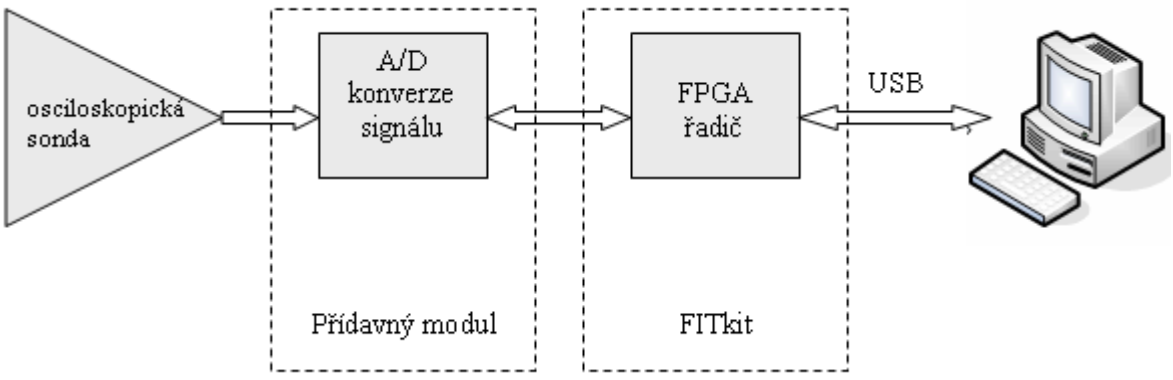
5 Koncepce řešení

5.1 Blokový popis

Celou práci je možno rozdělit do několika částí. Prvním problémem je vlastní analogově/digitální konverze vstupního napětí na číselnou reprezentaci vzorků zpracovatelných na FITkitu. K tomu je zapotřebí využít vhodných A/D převodníků. Převodníky integrované v mikrokontroléru na kitu jsou určeny především k vzorkování audio signálu a jejich parametry jsou tudíž nevhodné pro zpracování vysokofrekvenčního signálu s větším rozptylem hodnot napětí. Z toho důvodu je třeba navrhnout modul osazený rychlými A/D převodníky, který bude schopný komunikovat s čipem FPGA na FITkitu. Komunikace bude probíhat po sběrnici X čipu FPGA, která je vyvedena do spodního konektoru JP10 na FITkitu.

Další část projektu spočívá v navržení FPGA řadiče, který bude v pravidelných intervalech spouštět A/D převod. Získané hodnoty bude následně ukládat do RAM paměti na čipu. V okamžiku, kdy navzorkuje dostatečné množství dat, se vzorkování ukončí a hodnoty uložené v paměti RAM se přesunou přes USB rozhraní do PC. FPGA čip je pro toto použití velice vhodný zejména pro svoji schopnost velice rychlého čtení a ukládání dat do vestavěných pamětí RAM.

Poslední částí projektu je aplikace s grafickým rozhraním, běžící na PC. Tato aplikace umožní uživateli nastavit různé parametry vzorkování. V okamžiku stisknutí startovacího tlačítka začne FPGA řadič vzorkovat vstupní signál. Po navzorkování dostatečného množství dat a jejich přenesení do PC se na monitoru počítače v podobě grafu zobrazí průběh měřeného signálu. Blokové schéma celého návrhu je vyobrazeno na následujícím obrázku 5-1.



Obrázek 5-1 Blokový popis řešení

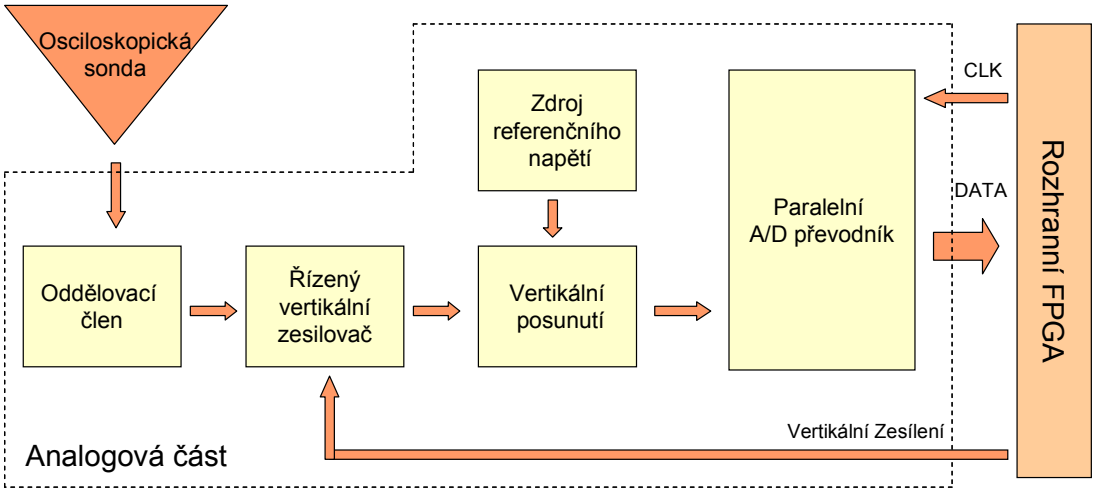
5.2 Obecný návrh analogové části

Nejdříve bylo nutné stanovit si parametry, které má osciloskop, resp. jeho analogová část splňovat.

Mezi základní požadavky jsem při návrhu stanovil:

- vysokou rychlost vzorkování (alespoň 20MHz),
- široký vstupní napěťový rozsah (± 20V),
- možnost měření i záporných hodnot napětí,
- možnost elektronicky měnit vertikální napěťové rozsahy,
- možnost napájení modulu přímo z FITkitu,
- odolnost proti rušení

Celou analogovou část jsem si poté rozčlenil do několika logických celků. Tyto celky jsem následně analyzoval a navrhnul konkrétní řešení. Blokové schéma analogové části je znázorněno na následujícím obrázku 5-2.



Obrázek 5-2 Blokové schéma analogové části

Prvním členem analogové části je pasivní osciloskopická sonda, pomocí které je měřený signál přiváděn na vstup osciloskopického modulu.

Vstupní signál je bezesporu nutné vhodným způsobem upravit. Prvním prvkem na vstupu modulu je oddělovací člen, který bude splňovat dvě funkce:

1. Omezí vstupní napětí na požadovaný napěťový rozsah.
2. Oddělí vstupní část modulu.

Za oddělovací částí následuje část vertikálního zesilovače, který je schopen řízeně upravovat vstupní napětí a měnit tak napěťové rozsahy.

Další částí modulu je blok vertikálního posunutí, který k upravenému vstupnímu signálu přičte napěťovou složku přivedenou z bloku referenčního napětí. Tato část je důležitá pro to, že A/D

převodník dokáže měřit pouze v omezeném kladném rozsahu hodnot napětí. Signály potřebné k přepínání rozsahů je potřeba vyvést na výstupní konektor, aby je bylo možné ovládat z řídicího FPGA řadiče.

Posledním blokem je samotný 8 bitový A/D převodník, jehož paralelní výstupy budou vyvedeny na výstupní konektor na modulu. Kromě paralelních výstupů převodníku je třeba na sběrnici vyvést ještě signál CLK, který bude sloužit k nastartování procesu konverze analogového vzorku na digitální.

Všechny datové, řídicí i napájecí vodiče budou vyvedeny na společný dvouřadý 50-ti pinový konektor tak, aby byl modul snadno připojitelný k platformě FITkit.

5.3 FPGA řadič - požadavky

Ústředním členem celého projektu je FPGA řadič implementovaný v programovatelném hradlovém poli osazeném na platformě FITkit.
Úloha FPGA řadiče je následující:

- Přečtení bloku dat z PC, kde bude obsažena informace o nastavení vzorkování. (Horizontální rozsah = frekvence vzorkování a vertikální rozsah)
- Řízení A/D převodu, (tzn. v pravidelných intervalech vybudit převodník vzestupnou hranou signálu CLK)
- Řízení vertikálního zesílení
- Dostatečně rychlé ukládání dostatečného počtu vzorků do paměti RAM
- Odesílání naměřených dat zpět do PC

Hlavním komunikačním prostředkem FITkitu s vnějším prostředím (např. s PC) je univerzální vysokorychlostní asynchronní přijímač-vysílač (UART) FT2232C firmy FTDI Chip [12], který je možné nakonfigurovat do různých módů komunikace:

- UART
- Asynchronní/synchronní paralelní port
- MPSSE (Multi Protocol Synchronous Serial Engine)
- MCU Host bus Emulation
- a další

5.4 Grafická aplikace na PC

Uživatelské prostředí osciloskopu bude řešeno formou aplikace s GUI implementované na PC.

Nejdůležitější částí aplikace bude zajisté graf, na kterém se bude zobrazovat průběh měřeného vstupního napětí v závislosti na čase. Díky skutečnosti, že o horizontální složku průběhu (čas) se postará výše uvedený FPGA řadič, není třeba zaznamenávat časovou složku vzorků. Aplikace pouze zobrazí na každém obrazovém bodu právě jeden naměřený vzorek. Pokud si vhodně zvolím výšku okna pro zobrazování vzorků (nejlépe 256 pixelů) nebude třeba ani nijak přepočítávat vertikální složku naměřených dat, ale bude možné přímo vykreslovat výstup z osmi-bitového převodníku na plochu vykreslovaného okna.

Dalšími nezbytnými prvky aplikace budou komponenty pro nastavení parametrů vzorkování. To je vzorkovací frekvence a vertikální zesílení signálu na vstupu.

Tlačítkem „SAMPLE“ vyvoláme proces vzorkování signálu. Nejprve bude potřeba odeslat zvolené parametry vzorkování a poté již očekávat naměřené vzorky.

Díky čipu FT2232C od firmy FTDI Chip[12] osazeném na FITkitu degraduje komunikace s kitem na komunikaci po sériovém kanálu. Téměř pro každý programovací jazyk jsou dostupné knihovny pro práci se sériovým portem. Součástí aplikace tedy musí být i formulář, který umožní nastavení komunikačního portu.

Mezi rozšiřující funkce programu zařadím:

- možnost pohybu po časové ose tak, aby si mohl uživatel prohlédnout každý detail navzorkovaného signálu
- možnost měření amplitudy, periody a frekvence vstupního signálu pomocí kurzorových úseček
- možnost vyexportování grafu do obrazového souboru

6 Realizace analogové části

Cílem této kapitoly je detailně rozebrat každou část navržené analogové části. Vrcholem kapitoly je kompletně realizovaná a osazená deska plošných spojů splňující všechny stanovené požadavky.

Při rozboru jednotlivých částí budu vycházet z blokového schéma analogové části v kapitole 5 (obr. 5-2). Vzhledem k faktu, že nejsem příliš zkušený v návrhu analogových, vysokofrekvenčních obvodů, snažil jsem se při realizaci vyjít z nějakého již hotového zařízení. Po jisté době jsem v praktické elektronice objevil článek [17], který mi návrh osciloskopického modulu velice usnadnil.

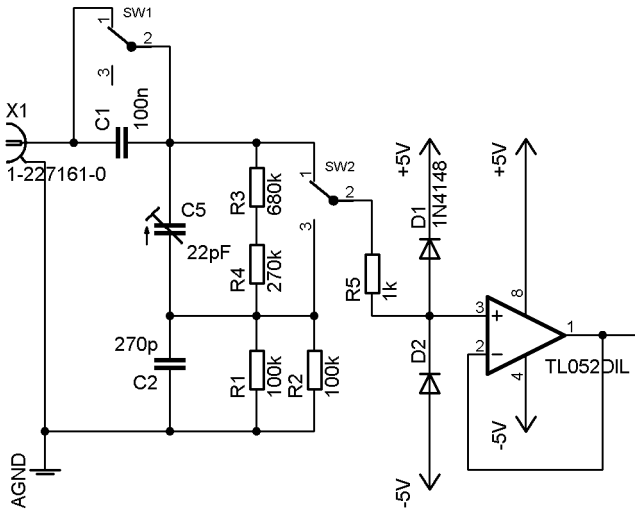
6.1 Oddělovací člen

Na vstupu analogové části se nachází oddělovací člen, jehož součástí je i BNC konektor pro připojení osciloskopické sondy. Jedná standardní 50Ω BNC konektor v provedení W, který je možné zapájet do desky plošných spojů (dále jen DPS). Plášť konektoru je připojen k analogové zemi (viz. kapitola 6-7), signálový vodič pokračuje dále přes kapacitu oddělující stejnosměrnou složku od střídavé. Tuto kapacitu je možné pomocí přepínače SW1 vyřadit.

Aby bylo možné měřit velké napěťové rozsahy, zařadil jsem za kondenzátor vstupní napěťový dělič, který je zapojen v poměru 1:10 a umožňuje měřit napětí v rozsahu až ± 20 V. Pro přesnější měření v rozsahu ± 2V je však možné dělič odpojit pomocí druhého přepínače SW2. Dělič by bylo možné odpojovat i pomocí relé. To jsem však kvůli vysoké spotřebě a požadavku napájení z kitu zamítnul. Protože odporový dělič obecně při vysokých kmitočtech „šumí“ bylo nutné jej kapacitně vykompenzovat. K tomuto účelu slouží dvojice kondenzátorů, z nichž jedna je nastavitelná.

Za děličem následuje poslední prvek oddělovacího členu, kterým je operační zesilovač TL052 splňující funkci impedančního převodníku. Tento operační zesilovač disponuje velkou rychlostí a je běžně dostupný v prodejnách s elektrotechnikou. Připojené diody plní funkci oříznutí hodnot napětí, které se vyskytují mimo měřitelný napěťový rozsah.

Nejvíce vypovídající je schéma zapojení oddělovacího členu, které je znázorněné na níže uvedeném obrázku 6-1.



Obrázek 6-1 Oddělovací člen

6.2 Řízený vertikální zesilovač

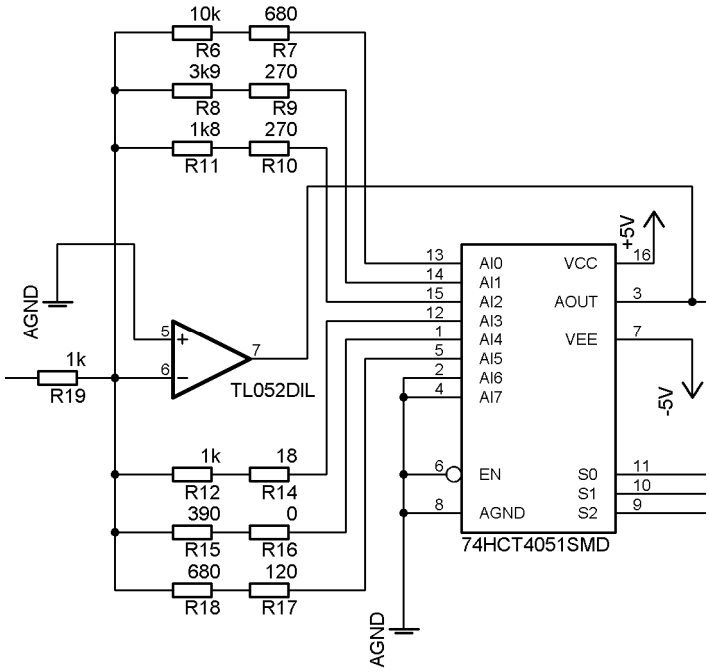
Za oddělovacím členem je zapojen blok s řízeným vertikálním zesilovačem. Tento prvek na základě vstupní adresy nastaví zesílení vstupního signálu na požadovanou hodnotu tak, aby jej mohl dále zpracovat A/D převodník.

Funkci vertikálního zesilovače tvoří operační zesilovač TL052 zapojený v invertujícím zapojení spolu s analogovým multiplexorem 74HCT4051.

Jedná se o osmikanálový analogový multiplexor/demultiplexor se třemi vstupy adresy, povolovacím vstupem EN aktivním v nízké napěťové úrovni, osmi nezávislými vstupy/výstupy (AI0 – AI7) a jedním společným vstupem/výstupem (AOUT). Vzhledem k potřebě zpracovávat i záporné napětí, musí být jak operační zesilovač, tak multiplexor napájen ± 5V. Povolovací vstup jsem zapojil „napevno“ k analogové zemi.

Aby mohl operační zesilovač zesilovat signál na různé úrovně, musí mít ve zpětné vazbě zapojeno více větví s rezistory. Podle adresy analogový multiplexor vybere právě jednu cestu a signál přivede zpět na invertující vstup operačního zesilovače. Zapojení obsahuje celkem 6 zpětnovazebních větví, z nichž 3 větve vstupní signál zesilují a 3 jej zeslabují. Pro rozlišení těchto úrovní využijeme všechny tři adresové vodiče multiplexoru. Hodnoty rezistorů jsou zvoleny tak, aby umožnily nastavit v aplikaci dostatečný rozptyl měřených hodnot. V každé větvi jsou potom zapojeny dva aby z nich bylo možné složit i hodnoty, které se běžně nevyrábí.

Na obrázku 6-2 je znázorněno schéma zapojení vertikálního zesilovače.



Obrázek 6-2 Řízený vertikální zesilovač

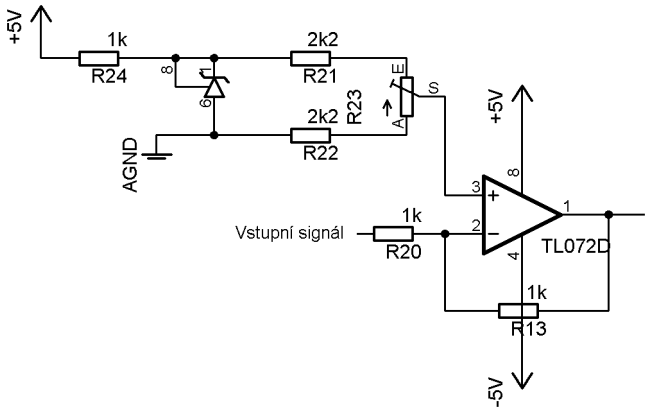
6.3 Blok vertikálního posunutí

Za blokem vertikálního zesilovače je připojen blok vertikálního posunutí. Protože použitý převodník zpracovává napětí v rozsahu 1.55 až 3.26 V [17], je nutné signál nakonec stejnosměrně posunout tak, aby při jeho nulové hodnotě (zkratovaný vstup) bylo na vstupu A/D převodníku napětí 2.4V, což je střed tohoto rozsahu.

K tomuto účelu slouží operační zesilovač TL072. Je opět zapojený v invertujícím zapojení, tím pádem nám otáčí fázi signálu zpět, tak jako byl na vstupu osciloskopického modulu. Stejnosměrný ofset (2.4V) je přiveden na neinvertující vstup operačního zesilovače ze zdroje referenčního napětí. Takto upravený signál je již možné přivést na vstup A/D převodníku.

6.4 Zdroj referenčního napětí

Jako zdroj referenčního napětí pro vertikální posunutí signálu slouží napěťová reference TL431 spolu s odporovým děličem a potenciometrem k přesnému doladění. Schéma zapojení zdroje referenčního napětí a bloku vertikálního posunutí je patrné na obrázku 6-3.



Obrázek 6-3 Vertikální posun signálu o referenční napětí

6.5 Analogově digitální převodník

K realizaci jsem zvolil osmi-bitový paralelní A/D převodník TDA8703 od firmy Philips Semiconductors [15].

Tento převodník je schopen navzorkovat až 40 milionů vzorků ze jednu sekundu. Přesto, že byl původně určen ke zpracování videosignálu, je tento převodník využíván v řadě amatérských zapojení zejména pro svoji dostupnost a nízkou cenu. Díky osmi paralelním třístavovým TTL výstupům může připojený FGPA řadič velice rychle číst naměřené vzorky. Další výhodou převodníku je fakt, že k němu není potřeba připojovat žádný obvod typu „sample-and-hold“, protože převodník na svých výstupech „podrží“ naměřenou hodnotu do doby, kdy je k dispozici další vzorek. Neméně důležitá je otázka spotřeby. Příkon použitého převodníku se pohybuje typicky kolem 290mW [15].

Převodník jsem zapojil podle katalogového zapojení v dokumentaci produktu [15]. Napájení převodníku je specifické díky vyžadovanému oddělenému napájení analogové části a digitální části. O problému s napájením bude blíže napsáno v následující části kapitoly. Měřený signál je přiveden na vstup převodníku označený jako *VI*. Ke spuštění měření signálu slouží vstup *CLK*. Každou náběžnou hranou na tomto vstupu je provedena konverze analogového signálu na digitální. Po dokončení převodu je hodnota k dispozici na osmi paralelních výstupech (D0 – D7). Převodník očekává na hodinovém vstupu *CLK* TTL signál. Protože však převodník chápe napětí na hodinovém vstupu větší než 2V jako „logickou 1“, můžu jej přímo připojit k výstupnímu konektoru a ovládat jej výstupem FPGA.

Problém ale nastává u výstupů převodníku. Jedná se o 5V TTL výstupy, které nemohou být připojeny přímo k FPGA. To totiž dokáže zpracovat pouze 3.3V logiku. Z tohoto důvodu je nutné mezi výstupy a konektor připojit ještě převodník napěťových úrovní.

Dobrou volbou se prokázal obvod SN74CB3T3245 od firmy Texas Instruments [16]. Jedná se o nízko-příkonový osmibitový převodník napěťových úrovní, který dokáže s minimálním zpožděním

(asi 0.25ns) převést 8 TTL výstupů z A/D převodníku na výstupy v 3.3 V logice. Tyto výstupy již je možné přímo připojit ke konektoru potažmo k FPGA obvodu.

6.6 Vstupně/výstupní konektor

Možnost připojení osciloskopického modulu zajišťuje dutinková lišta. Lišta čítá celkem 50 pinů ve dvouřadém provedení, díky čemuž lze jednoduše zasunout do konektoru JP10 na FITKitu.

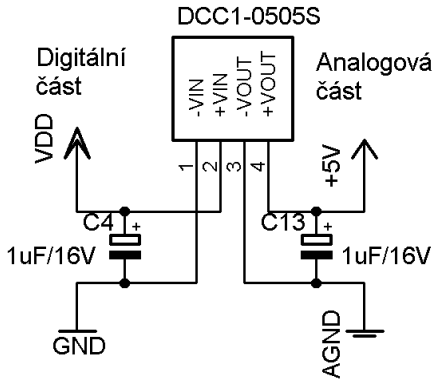
Výstupní 8-bitová datová sběrnice převodníku je připojena na piny 5 až 12. Pin č. 19 slouží k rozvodu hodin převodníku (Signál CLK). Prostřednictvím pinů 25, 26 a 27 je přivedena adresa pro výběr vertikálního zesílení. Vzhledem k faktu, že jak A/D převodník tak analogový multiplexor jsou vyrobeny technologií CMOS, je možné k jejich vstupům připojit i logiku 3.3V, přestože jejich napájecí napětí je 5V. CMOS obvody totiž „posoudí“ napětí 2V a vyšší, jako logickou úroveň „1 (High)“. Proto bylo možné připojit adresu multipexoru i hodinový signál CLK přímo ke konektoru bez nutnosti konverze 5V logiky. Výběr čísel pinů na konektoru jsem provedl až při navrhování DPS s ohledem na rozmístění všech součástek.

6.7 Napájení analogové části

Jedním z důležitých kritérií při návrhu osciloskopického modulu bylo napájení přímo z FITkitu, tedy bez nutnosti připojení externího napaječe.

Při této koncepci jsem se však potýkal se dvěma klíčovými problémy napájení. Jedním z nich je fakt, že použitý převodník vyžaduje oddělené napájení analogové a digitální části. Druhým problémem je potřeba záporného napětí pro napájení operačních zesilovačů. Záporné napájení zesilovače mi totiž dovoluje měřit záporná napětí vstupního signálu.

Pin konektoru č.1 je připojen na napětí 5V, nacházející se na kitu. Tento pin mi posloužil k napájení digitální části osciloskopického modulu. K oddělení napájecí větve analogových obvodů jsem použil obvod QDC1S-0505S firmy QLT POWER/AIMTEC [19]. Tento obvod plní funkci napěťového měniče 5V-5V a odděluje tak větev pro napájení analogových obvodů. Jeho zapojení je zobrazeno na obr. 6-5.

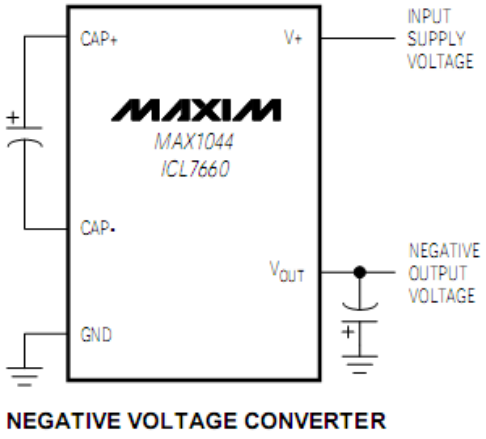


Obrázek 6-4 Zapojení napěťového měniče

Zdánlivě větším problém se zdála být zmiňovaná potřeba záporného napětí pro napájení operačních zesilovačů. Protože spotřeba proudu v záporné napájecí větvi je minimální, použil jsem pro „výrobu“ záporného napětí obvod ICL7660 firmy MAXIM [20]. ICL7660 je monolitický CMOS konvertor napětí, který pracuje na principu nábojové pumpy (podobně jako MAX232 pro konverzi TTL <=> RS232) . Po připojení kapacity a vstupního napájení +5V začne obvod generovat záporné napětí -5V. Katalogové zapojení tohoto obvodu je vyobrazeno na obrázku 6-6.

Napájení 3.3V pro obvod SN74CB3T3245, což je již zmiňovaný převodník napěťových úrovní, jsem vyvedl z pinu č. 2 na konektoru. „Digitální zem“ je vedena z pinů 3 a 4.

Kompletní schéma zapojení osciloskopického modulu je přiloženo v příloze B.



Obrázek 6-5 Katalogové zapojení ICL7660

6.8 Návrh a výroba desky plošných spojů

Po dokončení návrhu schématu zapojení osciloskopického modulu bylo nutné vybrat vhodná pouzdra všech použitých součástek a navrhnout desku plošných spojů.

Pro návrh DPS jsem se rozhodl použít nástroj EAGLE od firmy CadSoft Computer GmbH [21]. Jeho volně šířitelná verze EAGLE Light, která je k dispozici na stránkách výrobce, plně postačuje mým požadavkům. Mezi její největší omezení patří omezení velikosti navrhované desky na 100 x 80 mm. Tento rozměr se však pro moji desku nakonec ukázal jako dostačující.

Před vlastním kreslením schématu v prostředí EAGLE bylo nutné vytvořit si pouzdra a symboly všech použitých součástek. K tomuto účelu jsem si vytvořil novou knihovnu *FITKit.lbr*, ve které jsem sjednotil všechny použité součástky, jejichž seznam se nachází na přiloženém CD. Běžné součástky jako jsou rezistory a kondenzátory již byli vytvořeny v knihovnách dodávaných se softwarem. U těchto diskrétních součástek jsem volil pouzdra SMD o velikost 805. Kromě použitého A/D převodníku a operačního zesilovače TL052 jsem všechny součástky volil v SMD pouzdrech. Zmiňované dva obvody se mi podařilo koupit pouze v provedení THD.

Po dokončení schématu zapojení jsem započal návrh vlastní desky. Nejdříve bylo nutné vhodně uspořádat všechny součástky na desku tak, aby se jednak „gumové“ spoje EAGLU co nejméně křížily a také s ohledem na funkčnost zapojení. To znamená použití co nejkratších vodivých cest u vysokofrekvenčních signálů. Blokovací kondenzátory pro odstranění šumu napájecích větví bylo nutné umístit do těsné blízkosti napájecích vývodů všech integrovaných obvodů.

Pro „zaroutování“ všech spojů jsem využil dvě signálové vrstvy. Pro signály jsem použil šířku spoje o velikosti 0.4 mm, pro napájecí vodiče až 0.8 mm. Celý návrh signálových vrstev jsem završil zastíněním analogové i digitální části pomocí příslušných signálových zemí.

Následující seznam tvoří výčet všech použitých/vytvořených vrstev.

- *1 top, 17 Pads, 18 Vias* – Horní strana součástek (horní signálová vrstva)
- *16 bottom, 17 Pads, 18 Vias* – Spodní strana součástek (Spodní signálová vrsva)
- *29 tStop* - Maska součástek
- *30 bStop* - Maska spojů
- *20 Dimensions* – Obrys (velikost) DPS
- *44 Drils, 45 Holes* – Vrtání prokovené, neprokovené
- *101 tPrint* – Potisk horní strany součástek (TOP Layer)
- *102 bPrint* – Potisk strany spodní strany spojů (BOTTOM Layer)

Zakázku s doplňujícími parametry (konstrukční třída ...) jsme s Ing. Šimkem zadali firmě Gatema s r.o., která asi do deseti dnů doručila vyrobenou desku plošných spojů na fakultu.

Návrh DPS a fotografie vyrobené desky se nachází v příloze.

6.9 Osazení desky plošných spojů

Osazení desky jsem prováděl svépomocí v laboratořích L305 a L203 fakulty Informačních technologií. Zároveň bych chtěl poděkovat všem osobám, které mi umožnily trávit v laboratořích více času, než je zdrávo.

Jednotlivé součástky jsem pájel tamější páječkou od diskrétních SMD odporů a kondenzátorů přes SMD integrované obvody až po větší THD obvody a konektor.

Po zapájení všech součástek jsem celou desku umyl v lihové lázni, abych odstranil přebytečnou pájecí kapalinu. Fotografie osazeného osciloskopického modulu se nachází v příloze.

6.10 Oživení osciloskopického modulu

Před připojením osazeného modulu k FITkitu jsem proměřil všechny spoje. Ověřil jsem si tak, zda některé z nich nejsou přerušené či zkratované.

Jakmile se ukázalo, že jsou všechny spoje v pořádku, připojil jsem modul k napájecímu napětí +5V, abych ověřil spotřebu modulu a proměřil všechny napájecí větve obvodů. Spotřeba celého modulu se pohybovala kolem hodnoty 90mA. Díky této malé spotřebě modulu bylo možné napájet FITkit i modul pouze z USB portu počítače, bez nutnosti připojení externího adaptéru.

Pro ověření základní funkčnosti jsem si vyrobil ještě testovací modul osazený budičem sběrnice a osmi LED diodami, které umožňovali sledovat stav výstupu A/D převodníku. Na modulu se nachází i tlačítko s pull-up rezistorem, pomocí něhož lze generovat řídicí signál CLK převodníku. Kromě tlačítka jsem na testovací modul umístil i tři přepínače, která nastavují adresu pro výběr rozsahu vertikálního zesilovače.

Testovací modul mi umožnil zkontrolovat si výstup převodníku v každé situaci. Pomocí přepínačů jsem měnil rozsahy a tlačítkem si generoval hodinový signál *CLK*.

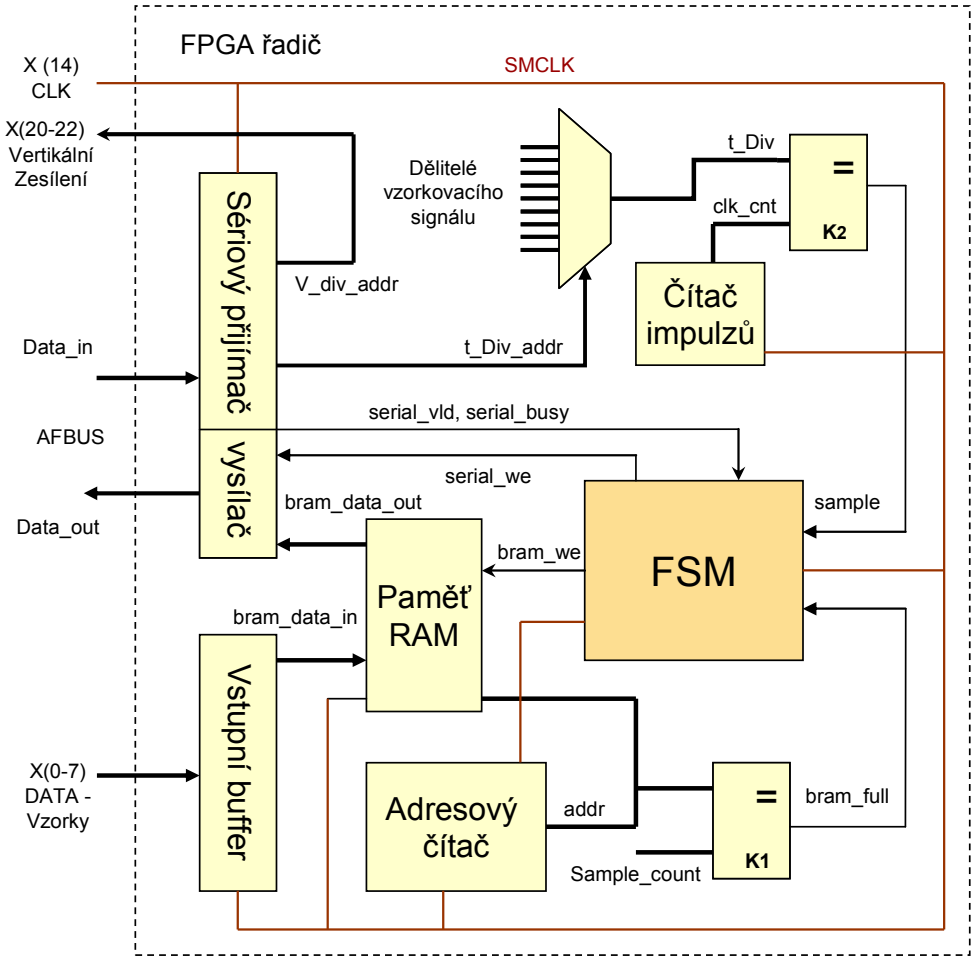
V okamžiku, kdy jsem si byl jistý funkčností modulu, připojil jsem jej k FITKitu. Schéma zapojení, návrh a fotografie testovacího modulu se nachází v příloze.

7 Realizace FPGA řadiče

Největším problémem realizace řadiče bylo navrhnout jeho architekturu. S ohledem na zadané požadavky jsem nejdříve sestavil blokové schéma z jednotlivých komponent. Tyto komponenty jsem v dalších fázích návrhu blíže analyzoval a po jedné implementoval.

7.1 Architektura FPGA řadiče

Na následujícím obrázku se nachází architektura celého řadiče.



Obrázek 7-1 Architektura FPGA řadiče

Rozhraní řadiče tvoří pětice signálů, z nichž trojice je napojena k osciloskopickému modulu a dvojice tvoří komunikační bránu mezi FPGA a grafickou aplikací běžící na PC.

Konkrétně se jedná o signály:

- *CLK (X14)* – Hodinový signál, který s každou náběžnou hranou spouští A/D konverzi převodníku

- *Vertikální zesílení (X(20-22))* – Adresa určující vertikální zesílení osazeného řízeného vertikálního zesilovače
- *Data – Vzorky (X(0-7))* – Digitální osmibitový výstup z převodníku
- *Data_in (AFBUS)* – Sériový vstup dat z PC
- *Data_out (AFBUS)* – Sériový výstup dat z řadiče

V následující části této sekce detailně popíšu zvlášť každou část řadiče.

7.1.1 Sériový přijímač/vysílač

FITkit obsahuje řadu periferií. Jejich ovládání není vždy jednoduché. Proto byla v rámci projektu FITkit vytvořena sada řadičů, které zjednodušují návrh aplikací nebo úloh na kitu. Mezi těmito řadiči byl implementován i řadič sériového rozhraní [18].

Tento řadič umožňuje komunikaci po asynchronní sériové lince pomocí protokolu rs232. Na FITkitu je možné tento řadič použít dvakrát. První možnost je na konektoru JP5, druhá je na USB portu A čipu FT2232C. Konektor JP5 a port A čipu FT2232C jsou přímo připojené k FPGA.

Řadič sériové linky umožňuje příjem i vysílaní slova podle nastavených parametrů rozhraní, které se nastavuje generickými parametry. Samotný řadič je složen ze dvou základních částí: řadič pro odesílaní dat (RS232_TXD) a řadič pro příjem dat (RS232_RXD) [18].

Řadič sériového rozhraní je možné nastavit pomocí pěti generických parametrů. Tyto parametry nastaví řadiče pro příjem a odesílaní na požadované vlastnosti.

Generické parametry:

- **SPEED** - nastavení rychlosti komunikace. Tabulka 7-1 ukazuje nastavení hodnoty SPEED, její odpovídající rychlosti a hodnotu, kterou se dělí hodinový signál SMCLK.
- **BITS** - Počet přenášených datových bitů. Možné hodnoty nastavení tohoto parametru jsou 5 - 8 datových bitů.
- **STOP** - Počet stop bitů. 1 nebo 2 *STOP*bity.
- **PARITY_EN** - Povolení paritního bitu. Pokud je PARITY_EN = 0 znamená to, že není nastavena žádná parita. Pokud je PARITY_EN = 1 znamená to povolení parity. O jakou paritu jde, určuje parametr PARITY.
- **PARITY** - Nastavení druhu parity. Pokud je PARITY = 0, bude počítaná a kontrolovaná sudá parita. Pokud je PARITY = 1, znamená to lichou paritu. Tento parametr se uplatní jenom v případě, pokud je nastavený PARITY_EN na hodnotu 1.

SPEED	Baud rate	Dělící poměr	Poznámka
"0000"	9600	768	7.3728MHz / 768 = 9600
"0001"	921600	8	7.3728MHz / 8 = 921600
"0010"	460800	16	7.3728MHz / 16 = 460800
"0011"	230400	32	7.3728MHz / 32 = 230400
"0100"	115200	64	7.3728MHz / 64 = 115200
"0101"	57600	128	7.3728MHz / 128 = 57600
"0110"	38400	192	7.3728MHz / 192 = 38400
"0111"	19200	384	7.3728MHz / 384 = 19200
"1000"	9600	768	7.3728MHz / 768 = 9600
"1001"	4800	1536	7.3728MHz / 1536 = 4800
"1010"	2400	3072	7.3728MHz / 3072 = 2400
"1011"	1200	6144	7.3728MHz / 6144 = 1200
others	9600	768	7.3728MHz / 768 = 9600

Tabulka 7-1 Hodnoty rychlosti přenosu podle nastavení parametru SPEED [18]

Při realizaci mého řadiče jsem nastavil přenosovou rychlost na maximální hodnotu 921600 Baudů. Protože šířka jednoho vzorku je 8 bitů, nastavil jsem stejnou hodnotu do parametru BITS, udávající počet přenášených bitů. Počet stopbitů jsem nastavil na hodnotu 1. Pro bezchybný přenos dat jsem nastavil sudou paritu.

Rozhraní sériového řadiče má relativně jednoduchou strukturu:

- *CLK, RESET* – Synchronní hodiny (SMCLK) a reset používané pro celé FPGA
- *DATA_IN* – Používá se pro vysílání dat na RS232_TXD
- *WRITE_EN* – Vysoká logická úroveň na tomto signálu vyvolá odeslání dat (DATA_IN) na RS232_TXD
- *DATA_OUT* – Používá se pro čtení dat přijatých na sériové lince.
- *DATA_VLD* – Vysoká logická úroveň na tomto signálu oznamuje příchozí data na sériové lince (DATA_OUT)
- *BUSY* – Vysoká logická úroveň na tomto signálu informuje, že právě probíhá komunikace po sériové lince
- *ERR* – Signál ERR informuje o případné chybě, která se vyskytla v průběhu komunikace. Do návrhu jej však neuvažuji, proto jsem jej označil klíčovým slovem OPEN
- *RXD, TXD, RTS, CTS* – Signálové vodiče pro sériovou komunikaci. V řadiči osciloskopu jsou připojeny na sběrnici AFBUS (0,1,2). Signál CTS ignoruji (OPEN)

Každý příchozí byte se v řadiči osciloskopu rozdělí do dvou signálů. Spodní 3 bity příchozích dat jsou prostřednictvím signálu *V_div_addr* vyvedeny signály skrze sběrnici X(20-22) na konektor JP10 na FITkitu. Tento signál určuje velikost zesílení vertikálního zesilovače.

Horní 4 bity vstupních dat jsou vyvedeny jako signál *t_div_addr* k multiplexoru, kde je na základě této adresy vybrán patřičný dělitel *t_Div*.

7.1.2 Vstupní buffer

Jedná se o klasický osmibitový registr synchronizovaný hodinovým rozvodem *SMCLK*. Registr je připojen přes konektor k výstupu A/D převodníku a slouží k uchování posledního naměřeného vzorku.

7.1.3 Paměť RAM

Jedním z hlavních požadavků na paměť je její rychlost. Jako naprosto dostačující paměť pro ukládání vzorků je paměť Block RAM, která je vestavěná přímo na čipu FPGA. Na použitém čipu XC3S50 jsou vestavěny 4 takovéto paměti. Velikost každé z nich je 18 kBitů.

Čtení z paměti Block RAM na čipu je asynchronní, což znamená, že po vystavení adresy jsou data na výstupu k dispozici okamžitě. Zápis dat je synchronizován vzestupnou hranou hodinového signálu *SMCLK*. Zápis dat se provede pouze v případě, že je nastaven signál *bram_we* ve vysoké logické úrovni.

Paměť RAM lze nastavit do několika konfigurací podle následující tabulky.

Konfigurace	Počet položek	Šířka datové položky	Počet bitů adresy
RAMB16_S1	16384	1 bit	14
RAMB16_S2	8192	2 bity	13
RAMB16_S4	4096	4 bity	12
RAMB16_S9	2048	8 bitů	11
RAMB16_S18	1024	16 bitů	10
RAMB16_S36	512	32 bitů	9

Tabulka 7-2 Konfigurace paměti Block RAM

Vzhledem k použitému osmibitovému převodníku jsem zvolil konfiguraci paměti RAMB16_S9. Tato konfigurace mi umožňuje uložit až 2048 vzorků, což je díky řízené časové základně dostačující počet.

Rozhraní komponenty RAMB16_S9 obsahuje tyto signály:

- *CLK* – synchronizační signál zápisu do paměti. Na tento signál je připojen rozvod hodinového signálu SMCLK.

- *DO* – Paralelní výstup dat z paměti. Prostřednictvím interního signálu *bram_data_out* je připojen přímo k sériovému vysílači.
- *DI* – Paralelní vstup dat do paměti. Prostřednictvím interního signálu *bram_data_in* je připojen k záchytnému registru (*Vstupní buffer*).
- *ADDR* – Adresa generovaná v bloku *čítač adresy*
- *EN* – Povolovací vstup. Nastavil jsem jej na vysokou logickou úroveň, aby byla paměť stále aktivní
- *WE* - Povolení zápisu do paměti. Prostřednictvím signálu *bram_we* jej řídí konečný stavový automat (blok *FSM*)

7.1.4 Adresový čítač

Čítač je řízen stavovým automatem FSM. prostřednictvím signálu *addr* je propojen s pamětí RAM a komparátorem *K1*.

Tento komparátor porovnává aktuální adresu s konstantou *Samples_count*, což je požadovaný počet vzorků. Tato hodnota je na začátku programového kódu nastavena na hodnotu 2000. Teoreticky je možné nastavit ji na hodnotu max. 2047, protože do paměti se vejde právě 2048 položek. V případě, že se hodnota adresy a hodnota *Samples_count* rovnají, je adresa vynulována a nastaví se signál *bram_full.*

7.1.5 Čítač impulzů

Podobně jako adresový čítač funguje čítač impulzů. Ten je důležitý pro správný chod časové základny osciloskopu. Jeho hodnota se zvyšuje o 1 s každou náběžnou hranou hodin.

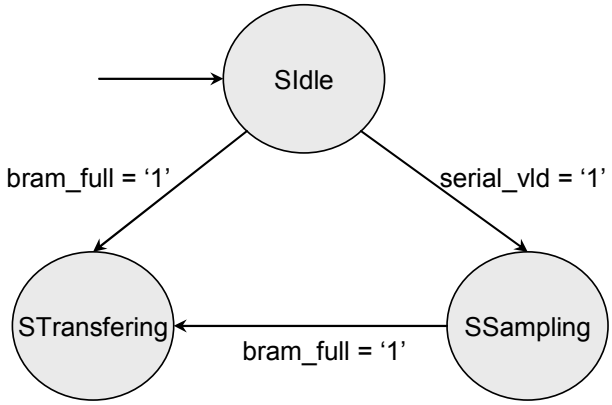
V komparátoru *K2* se hodnota čítače (*clk_cnt*) porovná s aktuálně nastaveným dělitelem vzorkovacího signálu (*t_Div*). Dělitel *t_Div* je vybrán multiplexorem na základě adresy *t_Div_addr*. Na vstupu je připraveno celkem 9 takovýchto dělitelů. Každý dělitel základního kmitočtu SMCLK je dopočítán tak, aby v oknu výsledné aplikace vycházeli „kulaté“ hodnoty časové základny na jeden dílek pomyslné obrazovky osciloskopu. (např. 1ms/dílek, 2ms/dílek ..)

Pokud se hodnota dělitele rovná hodnotě čítače impulzů (*clk_cnt*), nastaví se signál *sample* do log. „1“ a hodnota čítače se vynuluje. Tento mechanizmus se potom uplatní jako časová základna osciloskopu. Signál *sample* totiž zapříčiní uložení aktuálního vzorku do paměti. Více informací se nachází v následující části.

7.1.6 FSM

Blok FSM je konečný automat o třech stavech *SInit, SSampling* a *STransfering*. Automat je synchronizován hodinovým rozvodem SMCLK a řídí pomocí signálů většinu komponent řadiče.

Přechodový diagram automatu naznačuje obrázek 7-2.



Obrázek 7-2 Stavový automat FSM

7.1.6.1 SIdle

Stav *SIdle* je počátečním stavem automatu. V tomto stavu řadič neprovádí žádné akce. Čeká se pouze na příchozí byte, ve kterém se nachází informace o parametrech vzorkování. V případě, že přijde očekávaný byte, sériový přijímač nastaví signál *serial_vld* a automat přejde do stavu *SSampling*.

7.1.6.2 SSampling

V tomto stavu se vzorkují data a ukládají se do paměti. Data se vzorkují pořád stejnou rychlostí (SMCLK), avšak ne vždy se pořízený vzorek uloží do paměti. Vzorek se uloží jen v případě, že je „nahozen" signál *sample*. V tu chvíli se nastaví příznak *bram_we* a aktuální vzorek se uloží do paměti na místo, kam ukazuje adresa *addr*. Jakmile je vzorek uložen, inkrementuje se čítač adresy a vynuluje se čítač impulzů *clk_cnt,* který se jinak inkrementuje s každou náběžnou hranou hodin.

Ve chvíli, kdy je adresa *addr* rovna hodnotě *Samples_count,* je nastaven signál *bram_full* a automat přejde do stavu *STransfering*.

7.1.6.3 STransfering

V tomto stavu má řadič za úkol odeslat naplněnou RAM paměť se vzorky. S náběžnou hranou hodin nastaví řadič signál *serial_we* a zvýší adresu *addr,* čímž se odešle jeden vzorek z paměti. Tuto akci však vykoná jen v případě, že se právě neodesílají žádná data (*serial_busy = 0*).

Automat přejde do počátečního stavu *SIdle* v době, kdy je celá RAM odeslána (*bram_full = 1*) a čeká se na další akci uživatele.

7.2 Implementace FPGA řadiče

Jako implementační jazyk jsem zvolil jazyk VHDL. VHDL je programovací jazyk pro popis hardware (VHSIC HDL – Very High Speed Integrated Circuit Hardware Description Language). Díky tomu, že byl tento jazyk v roce 1987 standardizován organizací IEEE, je zaručena jeho kompatibilita a jednotnost. Jazyk VHDL se používá jak pro simulaci obvodů, tak i pro popis integrovaných obvodů, které se mají vyrábět. Je použitelný dokonce i pro popis analogových obvodů.

Pro implementaci řadiče jsem využil vývojového prostředí Project Navigator verze 9.2i firmy Xilinx, Inc. [11] . Toto vývojové prostředí je možné zdarma stáhnout na stránkách výrobce. Simulaci správné funkce řadiče jsem prováděl pomocí programu ModelSim XE 3 Starter stejné firmy, který je rovněž volně ke stažení na stránkách výrobce, nebo na stránkách projektu FITKit [9].

Každý VHDL soubor popisující hardware se skládá ze dvou částí. Tzv. entity a vlastní architektury.

- *Entita* definuje rozhraní obvodu, tzn. jednotlivé vstupní a výstupní signály.
- *Architektura* obsahuje VHDL kód, kterým je popsána funkce případně chování obvodu.

Jedna entita může mít i více architektur.

V projektu existuje vždy jedna nejvyšší (top-level) entita. V případě FITkitu její signály korespondují s jednotlivými piny FPGA obvodu. Z důvodu usnadnění práce byla v projektu FITkit vytvořena trojice základních entit, které jsou určeny k použití v projektech.

K mému projektu jsem si vybral top-level entitu *tlv_ide_ifc*, která je určena pro aplikace využívající sběrnici X jako port IDE případně jako univerzální rozhraní, na které je možné připojit různé periferie. K této entitě zbývalo dopsat architekturu znázorněnou v kapitole 7.1.

Nejdříve jsem si nadefinoval všechny potřebné signály a konstanty. Poté jsem do zdrojového kódu vložil komponentu *serial_transceiver* pro sériovou komunikaci a komponentu *RAM16_S9*. Jednotlivými procesy jsem naimplementoval jednotlivé bloky architektury, jako je multiplexor a komparátory K_1 a K_2. Stavový automat FSM jsem implementoval pomocí tří procesů:

- *pstatereg* - proces uchovávající aktuální stav
- *nstate_logic* – kde je definována logika následujícího stavu
- *output_logic* – udává, jaké akce se mají vykonávat v současném stavu.

Celou architekturu jsem naimplementoval do souboru *top.vhd*. K projektu bylo nutné připojit ještě řadu dalších souborů knihovny FITkit. Všechny zdrojové soubory nutné k procesu kompilace a syntézy jsou k dispozici na přiloženém CD.

8 Realizace aplikace na PC

Tato kapitola se zabývá realizací grafické aplikace umožňující komunikovat s FITkitem a zobrazovat na monitoru naměřené vzorky.

První část kapitoly se zabývá výběrem vhodného programovacího jazyka a implementačního prostředí, následuje rozbor komunikačních možností, dále rozdělení aplikace do jednotlivých tříd a nakonec popis jednotlivých tříd.

8.1 Výběr programovacího jazyka a prostředí

Po pečlivé úvaze a zhodnocení všech výhod a nevýhod jsem zvolil jazyk C# jako implementační jazyk pro moji aplikaci.

Jazyk C# je vysoce úrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework, který byl později standardizován komisemi ECMA a ISO. Jazyk C# je založen na jazycích Java a C++, ze kterého čerpá především syntaxi. C# lze využít k tvorbě databázových programů, webových aplikací a stránek, webových služeb, formulářových aplikací ve Windows, softwaru pro mobilní zařízení (PDA a mobilní telefony) atd. [22].

Vývojových prostředí pro jazyk C# je více (Turbo C# Explorer, SharpDevelop, MonoDevelop), vybral jsem si však oficiální vývojové prostředí firmy Microsoft – Visual Studio 2005. Visual Studio je podle mého názoru asi nejpropracovanějším prostředím pro formulářové aplikace, v jakém jsem měl možnost pracovat. Toto prostředí je možné pro studijní účely získat na serveru *msdn.e-academy.com* po předchozí registraci.

Pro tuto aplikaci se ukázal být programovací jazyk a prostředí téměř ideální, díky rychlému vytvoření layoutu aplikace a velice snadné komunikaci po sériovém portu. Jedinou velkou nevýhodou této volby je závislost na operačních systémech firmy Microsoft a nutnost instalace platformy .NET framework alespoň verze 2.0.

8.2 Komunikace PC aplikace s FITkitem

Jak již bylo psáno v kapitole 5.4, díky osazenému čipu FT2232C na FITkitu, degraduje komunikace s FPGA čipem na komunikaci po sériové lince.

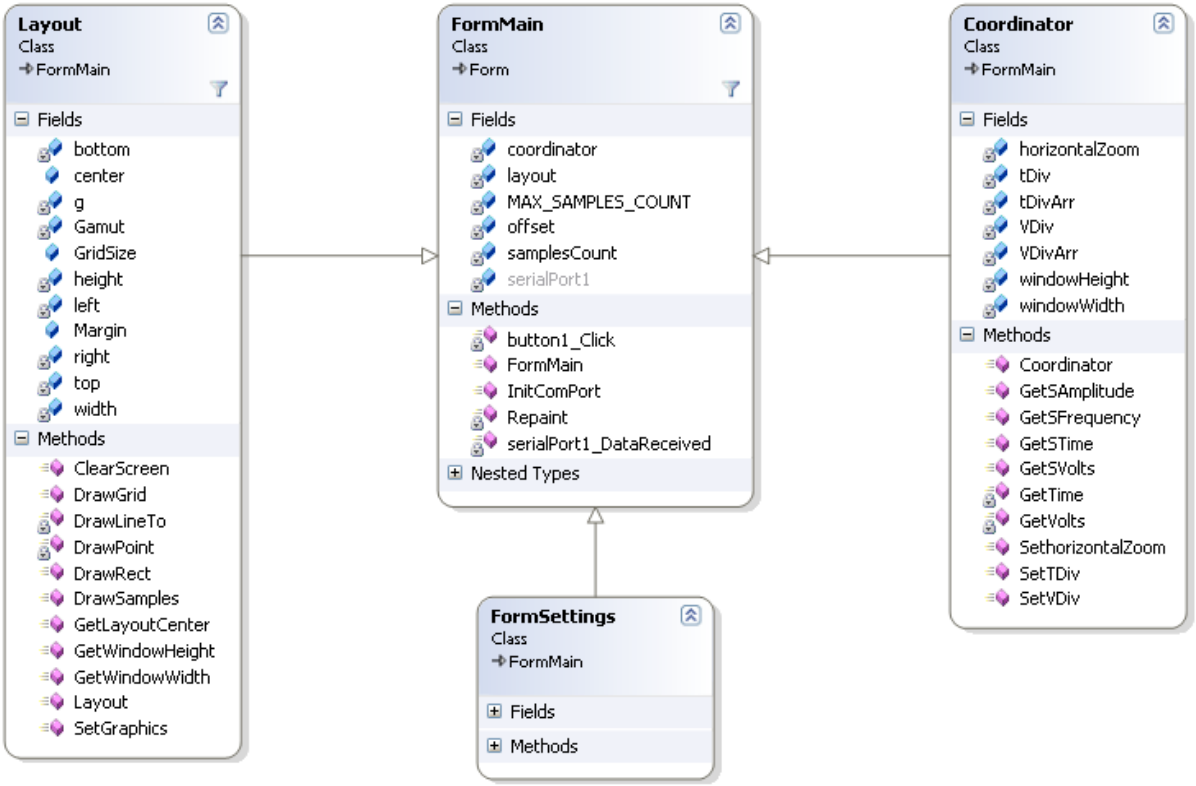
Díky použitému vývojovému prostředí byla problematika komunikace velice usnadněna. V prostředí MS Visual Studia je totiž integrovaná komponenta *serialPort* pro práci se sériovým portem.

Komponenta disponuje všemi vlastnostmi (Properties), které jsem při práci se sériovým portem potřeboval. Ve vlastnostech jsem si tedy mohl nastavit jméno komunikačního portu, přenosovou rychlost, paritu, počet stop Bitů a další.

V událostech komponenty (Events) jsem využil pouze jedinou. A sice událost *DataReceived*. Tato událost se zavolá vždy pokud přijdou na sériový port nějaká data. Potom již stačí metodou *Read* patřičná data z fronty vybrat.

8.3 Třídy aplikace

Aplikaci realizující požadované funkce jsem nazval *FITKit Scope*. Na následujícím obrázku 8-1 je znázorněn diagram tříd této aplikace. Mezi vlastnosti a metody tříd jsem v diagramu uvedl jen ty nejdůležitější.



Obrázek 8-1 Diagram tříd aplikace FITKit Scope

8.3.1 Třída FormMain

Třída FormMain je hlavní třída projektu, řídicí celý běh programu. Třída implementuje hlavní formulář aplikace, na který jsem přidal řadu komponent, umožňující ovládání aplikace.

Na formuláři se nachází dvojice komponent *CommboBox*. Jedna slouží pro výběr vertikálního zesílení a druhá k nastavení časové základny. Pro výběr časové základny je k dispozici celkem

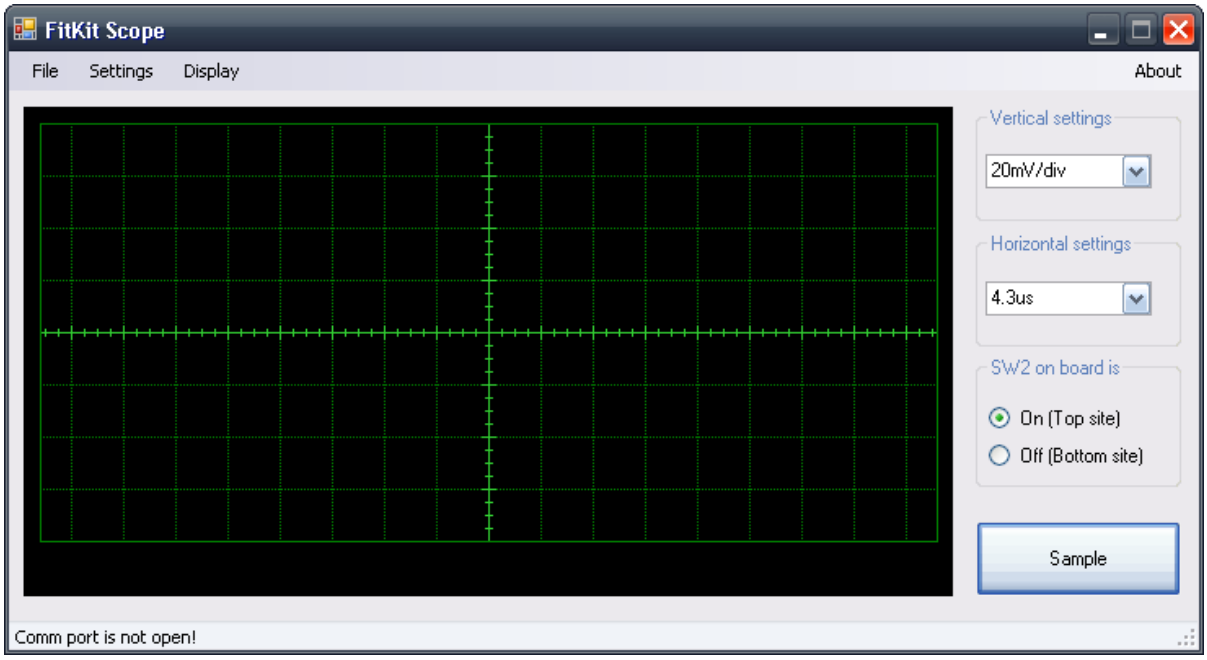
8 položek pohybujících se v rozmezí od 4µs do 20ms na jeden dílek vykreslený na pomyslné obrazovce osciloskopu. Každá položka komponenty potom zapříčiní odeslání adresy dělitele hodin FPGA řadiče *t_Div*, který je popsán v kapitole 7.1.

Nastavení vertikálního zesílení má však rozsahy dva. Jeden rozsah čítá 5 položek a je nastavitelný v rozmezí 20mV až 0.5V na dílek. Druhý rozsah pak nabízí možnosti 1V, 2V a 5V na jeden vykreslený dílek. O tom, který rozsah bude v komponentě zobrazen rozhoduje dvojice zaškrtávacích tlačítek (*radioButtons*). Tato tlačítka se vyptávají na stav přepínače SW2, který je osazený na desce osciloskopického modulu. Tento vypínač, jak bylo psáno v kapitole 6.2, odpojuje, resp. připojuje vstupní napěťový dělič a tím pádem ovlivňuje zesílení vstupního signálu. Protože přepínač není elektronicky řízen, aplikace vyžaduje, aby o stavu přepínače informoval uživatel prostřednictvím těchto tlačítek.

Pomyslná obrazovka osciloskopu se vykresluje do komponenty *pictureBox*. Pod touto komponentou se nachází trojice políček (*Label*), které zobrazují číselné údaje měření jako je amplituda, perioda a frekvence měřeného signálu.

V horizontálním menu aplikace se dále nacházejí funkce pro exportování naměřeného obrázku do souboru (*File/Export Image*), tlačítko pro vyvolání dialogu s nastavením připojení (*Settings/Connection settings*) a možnost různého vykreslování vzorků v menu *Display*.

Celý layout aplikace završuje tlačítko s názvem *Sample*, které slouží pro spuštění procesu vzorkování. Vzhled aplikace je patrný na následujícím obrázku 8-2.



Obrázek 8-2 Vzhled aplikace FitKit Scope

Následující seznam popisuje nejvýznamnější metody této třídy.

- **FormMain()** : Konstruktor třídy. Zde se inicializují všechny komponenty formuláře. Poté se vytvoří instance tříd *Layout* a *Coordinator*. Tyto třídy budou popsány níže.
- **pictureBox1_Paint()** : Metoda, která se automaticky zavolá v případě potřeby překreslit komponentu *pictureBox*. Odtud se volají metody třídy *Layout*, které se o správné vykreslení postarají.
- **InitComPort(string portName)** : Tato metoda inicializuje název sériového portu řetězcem předaným v parametru *portName* a pokusí se tento port otevřít.
- **serialPort1_DataReceived(...)** : Metoda, která se vyvolá v případě, že se na sériovém portu objeví příchozí data. V našem případě se vždy jedná o příchozí vzorky, které se v metodě uloží do připraveného pole *samples*.

8.3.2 Třída Layout

Třída Layout zapouzdřuje vlastnosti a metody pro vykreslování prvků na pomyslnou obrazovku osciloskopu (komponenta *pictureBox*). Třída prostřednictvím svých dat uchovává rozměry okna pro vykreslování, rozměry jednotlivých dílků, předdefinovaná pera pro kreslení na obrazovku a další.

Mezi její nejvýznamější metody patří:

- **Layout(int Width, int Height) :** Metoda je konstruktorem třídy. Z předaných rozměrů se spočítají některé důležité souřadnice jako je např. střed okna, okraje atd.
- **DrawGrid() :** Metoda do připraveného okna nakreslí pomocí čar a předdefinovaných per mříž. Tato mříž je rozdělena na 8 dílků ve vertikální poloze a počet dílků v horizontální poloze se dopočítá podle velikosti okna. Metoda nakonec nakreslí osy se stupnicí, které umožní odečítat hodnoty napětí a času.
- **DrawSamples(Byte[] Samples, int Count, int Offset) :** Metoda slouží pro vykreslení získaných vzorků v poli *Samples* na obrazovku. Parametr *Count* určuje počet dostupných vzorků a parametr *Offset* určí, od kterého vzorku se má začít vykreslovat. Aplikace totiž umožňuje se pomocí myši pohybovat po časové ose signálu. Metoda pro každý vzorek zavolá buď metodu *DrawPoint(int X, int Y)*, nebo metodu *DrawLineTo(int X, int Y),* podle toho, zda uživatel vybral zobrazování bodové, nebo pomocí úseček. Díky tomu, že o časovou základnu a vertikální zesílení se již postaral osciloskopický modul s FPGA řadičem, metoda *DrawSamples* už pouze vykresluje každý vzorek na jeden obrazový bod na časové ose. Tzn. 2000 naměřených vzorků znamená průběh o „délce“ 2000 pixelů. Protože je šířka plochy pro vykreslování vysoká 256 pixelů, není třeba přepočítávat ani vertikální složku vzorků.

8.3.3 Třída Coordinator

Tato třída zapouzdřuje data a metody pro práci s nastavením vertikální a horizontální složky signálu. Datové položky třídy uchovávají kromě jiného dvě pole koeficientů *VDivArr* a *tDivArr*. Tyto koeficienty slouží ke přepočítání polohy myši nacházející se nad plochou pro vykreslování vzorků na hodnoty napětí a času. Přepočet je samozřejmě ovlivněn nastavením časové základny a vertikálním zesílením osciloskopu.

Mezi nejvýznamnější metody třídy patří:

- **GetVolts (Point p1, Point p2):** Metoda nejdříve spočítá rozdíl vertikálních složek předaných bodů p1 a p2. Poté rozdíl vynásobí jedním z koeficientů z pole *VDivArr*. Vypočítanou hodnotu funkce vrátí v podobě čísla datového typu *double*. Pomocí metody lze určit okamžitou hodnotu napětí v závislosti polohy myši a nastaveném vertikálním rozsahu.
- **GetSVolts (Point p1, Point p2):** Obdobná funkce jako *GetVolts* s tím rozdílem, že hodnota napětí je vrácena v podobě řetězce včetně příslušné jednotky.
- **GetSAmplitude (Point p1, Point p2):** V podstatě stejná metoda jako předchozí. Avšak vrací absolutní hodnotu napětí. Tuto metodu používám k měření amplitudy signálu.
- **GetTime(Point p1, Point p2):** Vrací hodnotu času, který je vypočítán ze vzdálenosti předaných bodů *p1* a *p2*.
- **GetSTime(Point p1, Point p2) :** Hodnotu času vrátí jako řetězec.
- **GetSFreqency(Point p1, Point p2):** Vypočítá a v podobě řetězce vrátí měřenou frekvenci.

8.3.4 Třída FormSettings

Třída implementuje jednoduchý formulář s jednou komponentou *Combobox* a dvěma tlačítky. Jedná se o dialogový formulář, který umožňuje nastavit komunikační port. Pomocí *ComboBoxu* uživatel vybere požadovaný port Com. Tlačítkem *Connect* pak může volbu potvrdit a metodou *InitPort* třídy *FormMain* se aplikace pokusí zvolený port otevřít.

Druhé tlačítko *Cancel* volbu zruší a zavře dialogové okno.

8.4 Chování aplikace

Při spuštění aplikace *FitKit Scope* se v konstruktoru třídy *FormMain* zavolá metoda *InitCom*, která se pokusí otevřít komunikační port. Pokud se to metodě podaří, vypíše se tato informace do „statusBaru“ hlavního okna. Zároveň se při startu aplikace vytvoří instance tříd *Layout* a *Coordinator*.

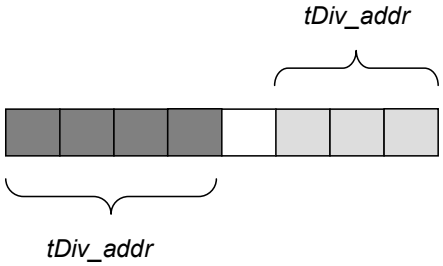
Prostřednictvím metod třídy *Layout* se na obrazovku vykreslí plocha pro vykreslování průběhu signálu se všemi dílky, osami a stupnicí.

Pokud uživatel začne měnit položky v komponentách *comboBox*, úměrně se mění i jednotlivé koeficienty *tDiv* a *VDiv* v instanci třídy *Coordinator.*

Najedeme-li kurzorem myší nad plochu pomyslné osciloskopické obrazovky, aplikace okamžitě vypočítává metodami *GetVolts a GetSVolts* napětí vzhledem k poloze myši a poloze časové osy. Toto napětí se průběžně překresluje do připravených políček (*Label*).

Stiskne-li uživatel levé tlačítko myši, uchová se její aktuální poloha a při dalším pohybu myši je na ploše vykreslován obdélník, který umožňuje provádět měření na ploše obrazovky. Metody třídy *Coordinator* totiž přepočítávají velikost zobrazeného obdélníku na hodnoty napětí, času a frekvence. Tyto údaje jsou rovněž zobrazovány do připravených políček pod „osciloskopickou obrazovkou".

Událost vyvolaná při stisku tlačítka *Sample* musí odeslat jeden Byte po sériovém kanálu, který oznámí řadiči, že se uživatel rozhodl vzorkovat data. Tento Byte musí obsahovat jak adresu *tDivAddr,* tak adresu *VDivAddr*. Struktura tohoto Byte je na následujícím obrázku.



Obrázek 8-3 Struktura odeslaného Byte

Tyto adresy nejsou nic jiného než indexy zvolených políček v komponentách *ComboBox.* Zmiňovaný Byte se následně odešle metodou *Write* třídy *serialPort.*

Bezprostředně po odeslání dat, se vyvolá událost *DataReceived* oznamující příchozí vzorky. Tyto vzorky jsou ihned po přijetí vykresleny metodou *DrawSamples* třídy *Layout* a aplikace je znovu připravena na další akce uživatele.

8.5 Aplikace FitKit Logic Analyzer

Na přání Ing. Šimka jsem jako rozšíření diplomové práce naimplementoval ještě druhou aplikaci s názvem *FiTKit Logic Analyzer.*

Tato aplikace je vhodná k měření digitálních signálů a její funkce se dá přirovnat k funkci logického analyzátoru. Vykresluje tedy jakýsi idealizovaný průběh naměřeného digitálního signálu. Tzn. před vykreslením se každý vzorek analyzuje a určí se, zda hodnota napětí vzorku spadá do logické úrovně „0" nebo „1". Podle toho se poté vykreslí idealizovaný obdélníkový průběh. Pokud nastane situace, že vzorek nezapadne ani do jednoho intervalu pro log. „0" nebo log. „1", je vzorek vykreslen odlišnou barvou ve středu vertikální osy.

8.5.1 Formulář FormMain

Vzhled hlavního formuláře *FormMain* je velice podobný formuláři aplikace *FitKit Scope.* Nastavení časové základdany je identické jako u předchozí aplikace.

Hlavním rozdílem těchto aplikací je způsob vertikálního nastavení. V aplikaci *FitKit Logic Analyzer* je tato volba nazvaná jako „Switching Level“. Klasická volba napětí na jeden dílek zde chybí. Pomocí komponenty ComboBox *Switching Level* si uživatel vybírá „technologii“ měřeného signálu.

Na výběr jsou technologie *5V CMOS, 5V TTL, 3.3V LVTTL, 2.5V CMOS, 1.8V CMOS, 1.5V CMOS, 1.2V CMOS* a také položka *User defined.* Tyto položky jsou zase rozděleny do dvou kategorií podle toho, jak je nastaven přepínač SW2 na desce, potažmo podle nastavení komponent RadioButtons.

Poslední zmiňovaná položka *User Defined* je specifická možností nastavení vlastních rozhodovacích úrovní. Tyto rozhodovací úrovně si uživatel může nastavit v dialogovém oknu *Menu/Settings/Switching Level Settings*. Dialogové okno má název *FormTreshold* a obsahuje čtveřici textových polí a potvrzovací tlačítko. Do těchto polí stačí zadat minimální a maximální hodnotu napětí logické úrovně „0“ a minimální a maximální hodnotu napětí logické úrovně „1“.

Tlačítkem *Analyze* na hlavním formuláři pak uživatel spustí logickou analýzu vstupního signálu.

8.5.2 Třída Level

Třída Level zapouzdřuje data a metody, které se starají o rozhodnutí, zda naměřený vzorek spadá do intervalu log. „0“, nebo log. „1“.

Třída uchovává pole *tresholdsArr* datového typu *Treshold*. V tomto poli jsou uloženy prahové rozhodovací úrovně, podle kterých je každý vzorek „zaškatulkován“. Vlastnost *treshold* uchovává aktuálně nastavené prahové úrovně. Tj. jednu položku z pole *tresholdsArr*.

Mezi nejdůležitější metody třídy patří:

- **TresholdInit()**: Metoda je volána v konstruktoru třídy *Level*. Úlohou metody je naplnit pole *tresholdsArr* prahovými úrovněmi napětí podle jednotlivých technologií. Konkrétní úrovně jsem zvolil podle literatury [24].
- **SetUserTreshold(double min0, double max0, double min1, double max1)**: Nastaví uživatelsky definované prahové úrovně.
- **Is1(double Volts), Is0(double Volts)**: Metody vrátí hodnotu typu Boolean podle toho, zda je hodnota *Volts* vyhodnocena jako „log.1“ nebo „log.0“.

9 Testování

Tato kapitola se zabývá uvedením každé části projektu do chodu a nakonec zhodnotí naměřené výsledky v porovnání se sériově vyráběným osciloskopem.

9.1 Oživení všech částí projektu

Abychom mohli oživit hardwarovou část projektu, je třeba připojit osciloskopický modul ke konektoru JP10 na FITkitu, připojit FITkit pomocí USB kabelu k PC a nahrát vygenerovanou konfiguraci do čipu FPGA na FITkitu. K BNC konektoru na modulu je ještě nutné připojit osciloskopickou sondu. Důležité je, aby se jednalo o pasivní sondu 1:1. Některé sondy označené jako 1:10 totiž signál zeslabují. Nebo je možné použít sondu s přepínačem, kde je poměr 1:1 nastavitelný.

Konfigurace pro FPGA je přiložena na CD spolu se všemi zdrojovými VHDL kódy. Pro překlad těchto kódů a vygenerování konfiguračního souboru je možné buď využít vývojového prostředí Project Navigator, nebo projekt zkompilovat přiloženým makefilem, podobně jako ostatní aplikace projektu FITkit.

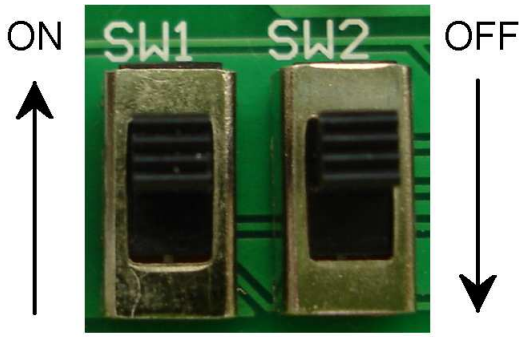
Zdrojové i spustitelné soubory aplikací *FitKit Scope a FitKit Logic Analyzer* jsou rovněž k dispozici na přiloženém CD. Více informací o všech souborech na CD je k dispozici v souboru *Readme.txt*, který se nachází v kořenovém adresáři CD.

V okamžiku, kdy máme vše připojeno a v FPGA se nachází správná konfigurace, můžeme spustit aplikaci *FitKit Scope*. Po spuštění se dole ve Status baru objeví informace, zda byl seriový port korektně otevřen, či nikoliv. Pokud ne, musíme v nastavení připojení vybrat správný port.

Po zdařilém připojení nejprve zkratujeme vstupní sondu a stiskneme tlačítko *Sample*. Na obrazovce se objeví červenou barvou průběh signálu. Pokud není průběh vykreslen přesně na hodnotě 0V, můžeme pomocí potenciometru P1 na osciloskopické desce převodník zkalibrovat.

Nyní již můžeme měřit libovolná napětí a frekvence v rozsahu, který nám aplikace dovoluje. Měli bychom si ale dát pozor na nastavení přepínače SW2 na desce modulu. Při měření napětí větších než ±2V musí být v tento přepínač poloze *Off (Bottom site)*. Poloha *Off* je taková poloha přepínače, kdy je jeho páčka přepínače dále od nápisu „SW2“ na desce. Lépe polohy přepínačů ilustruje obrázek 9-1.

Pokud bychom nechtěli zohlednit stejnosměrnou složku vstupního signálu, přepneme přepínač „SW1“ do polohy *Off,* čímž zařadíme do cesty signálu kondenzátor, který stejnosměrnou složku nepropustí. V případě, že budeme měřit rychle se měnící signál (obdélníkový signál), můžou se ve vykreslovaném signálu zobrazovat nežádoucí jevy typické při nabíjení či vybíjení kondenzátoru. Tento neduh lze odstranit pootočením regulovatelné kapacity C5 na desce, čímž se vykompenzuje vstupní napěťový dělič.



Obrázek 9-1 Polohy přepínačů SW1 a SW2

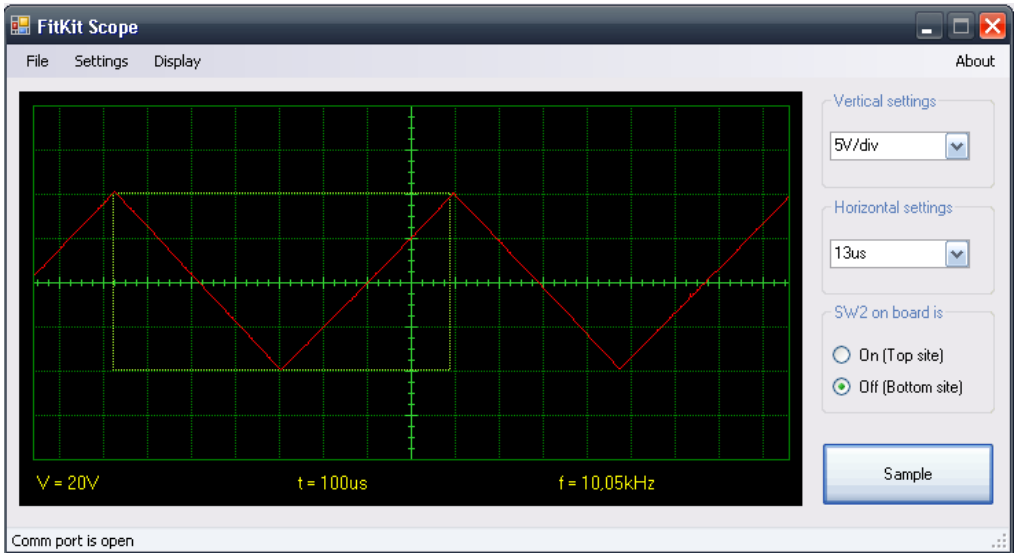
9.2 Měření a dosažené výsledky

Měření jsem prováděl ve fakultní laboratoři L305, kde jsem srovnával naměřené výsledky s výsledky naměřenými na tamějším osciloskopu. Vstupní data jsem generoval z generátoru signálu.

Po chvíli měření se ukázalo, že nesouhlasí časový údaj v jednom z rozsahů. Po přepočítání jednoho z dělitelů základního kmitočtu v programu FPGA byla chyba vyřešena. Drobné odchylky na časové ose lze sledovat u vysokých frekvencích ve stovkách kHz, při menších frekvencích je měření v časové ose naprosto přesné. Tato chyba je způsobena tím, že frekvence hodinového signálu není celé číslo, ale má dlouhý desetinný rozvoj. Uživatel však požaduje, aby hodnota času vyjádřená na jeden dílek byla pokud možno celé „kulaté“ číslo. Proto zde vznikne drobná zaokrouhlovací chyba.

Ve vertikální rovině je možné pozorovat výkyvy převodníku, které se pohybují kolem 5% z nastaveného rozsahu napětí. Při rozsahu 20mV na dílek tedy dosahuje odchylek ±1mV.

Následující obrázek ukazuje jeden z výstupů aplikace a zároveň dokumentuje funkčnost zařízení. Vstupní signál byl vygenerován generátorem pilového průběhu s amplitudou 10V a periodou 10kHz. Více výstupů aplikace je zobrazeno v příloze A.



Obrázek 9-2 Výstup aplikace při měření pilového průběhu signálu

10 Závěr

Cílem práce bylo seznámit se s problematikou analogově digitálního převodu, principem A/D převodníků, prostudovat jazyk VHDL pro návrh číslicového hardware a obeznámit se s výukovou platformou FITkit. S těmito znalostmi poté navrhnout a realizovat řešení, zabývající se využitím platformy FITkit a jeho programovatelných součástí, coby prostředníkem mezi modulem s analogově číslicovými převodníky a počítačem. Aplikace na PC pak měla umožnit zobrazovat průběhy měřeného napětí, podobně jako to dělá digitální osciloskop.

Studium literatury týkající se analogově digitálního převodu a A/D převodníků [1-8] se uplatnilo zejména při tvorbě úvodních kapitol práce a bylo důležité při návrhu konkrétního zapojení analogového modulu. K nastudování platformy FITkit mi pomohla především literatura dostupná na webových stránkách projektu FITkit [9]. S řešením hardwarové části projektu mi byla nápomocná literatura a produkty týkající se návrhu schématu zapojení [17], výběru součástek [15, 16, 17, 19, 20] a návrhu desky plošných spojů [21]. K návrhu a realizaci FPGA řadiče jsem využil literaturu a produkty vzniklé v projektu FITkit [9, 18, 23].

Výsledkem této práce je funkční osciloskopický modul, připojitelný k platformě FITkit, FPGA aplikace řídicí proces vzorkování signálu a dvě grafické aplikace, které umožňují zobrazení měřeného signálu na obrazovce počítače.

Co se týká možnosti rozšíření této práce, všichni jistě cítíme, že prostoru k rozšiřování je zde ještě dostatek. Největší výzva však pravděpodobně bude ve zvyšování výkonu a přesnosti osciloskopického modulu. Softwarové části projektu by pak bylo vhodné rozšířit například o možnost permanentního zobrazování vzorků v čase (Real-time), což je možné vidět u sériově vyráběných osciloskopů. Dále je možné řešit problémy s antialiasingem, či přidat různé možnosti spouštění osciloskopu (pre-trigger, post-trigger...).

Přínosem této práce je rozšíření projektu FITKit o další zajímavou aplikaci, která doufám vzbudí zájem i ze strany studentů. Největším přínosem však tato práce byla pro mě. Umožnila mi nahlédnout do problematiky návrhu hardwarového modulu a si pak jej vlastnoručně osadit. Dále mi rozšířila obzory v návrhu a fyzickému využití programovatelného hardwaru v podobě FPGA řadiče.

Jedním z cílů projektu byla také možnost využití osciloskopického modulu pro podporu výuky. Proto byl při návrhu kladen důraz na jednoduchost, přehlednost grafické aplikace a na výběr všech součástek s ohledem na jejich cenu a dostupnost.

Literatura

[1] Malina, V. *Poznáváme elektroniku VII, Osciloskop a jeho využití*. České Budějovice, 2002.

[2] Lavický, M. *Analogově číslicové převodníky*, dokument dostupný na URL: http://lavicky.webpark.cz/projekt/index.html (prosinec 2007)

[3] *Analog-to-digital conversion*, článek dostupný na URL: http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci213760,00.html (prosinec 2007)

[4] *Shannonův teorém*, článek dostupný na serveru: http://cs.wikipedia.org/wiki/Shannon%C5%AFv_teor%C3%A9m (květen 2008)

[5] Šebesta, V. *Systémy, procesy a signály 1*. Skriptum VUT, VUTIUM, Brno, 1997.

[6] *A/D převodník*, článek dostupný na URL: http://cs.wikipedia.org/wiki/A/D_p%C5%99evodn%C3%ADk (prosinec 2007)

[7] Háze, J., Vrba R., Fujcik L., Sajdl O. *Teorie vzájemného převodu analogového a číslicového signálu.* FEKT VUT, Brno, 2006

[8] Pospíšil, V. *A/D a D/A převodníky*, dokument dostupný na URL: http://praktika.fjfi.cvut.cz/data/VIP/V.Pospisil/scan/prevodniky%20A_D.pdf (prosinec 2007)

[9] Fučík, O., aj. *Projekt FITkit*, dokumentace dostupná na URL: http://merlin.fit.vutbr.cz/FITkit (květen 2008)

[10] Filip, T. *Testování výukové platformy FITkit*, diplomová práce, Brno, FIT VUT v Brně, 2007

[11] Xilinx, Inc., *WebPACK ISE 8.2i*, produkt dostupný na URL: http://www.xilinx.com/ise/logic_design_prod/webpack.htm (prosinec 2007)

[12] Future Technology Devices International Ltd., *Dual USB UART/FIFO IC* , 2007 dokument dostupný na URL: http://www.ftdichip.com/Products/FT2232C.htm (prosinec 2007)

[13] Texas Instruments, Inc., MSP430F168 *Mixed Signal Microcontroller*, dokument dostupný na URL: http://focus.ti.com/lit/ds/symlink/msp430f168.pdf (prosinec 2007)

[14] *The GCC toolchain for the Texas Instruments MSP430 MCUs*, produkt dostupný na URL: http://mspgcc.sourceforge.net/index.html (prosinec 2007)

[15] Philips Semiconductors, *TDA8703*, dokument dostupný na URL: http://www.nxp.com/acrobat_download/datasheets/TDA8703_3.pdf (prosinec 2007)

[16] Texas Instruments, Inc., *SN74CB3T3245* dokument k produktu dostupný na URL: http://focus.ti.com/lit/ds/symlink/sn74cb3t3245.pdf (prosinec 2007)

[17] Doležal, D. *Jednoduchý digitální osciloskop pro PC,* Praktická elektronika 10/2006

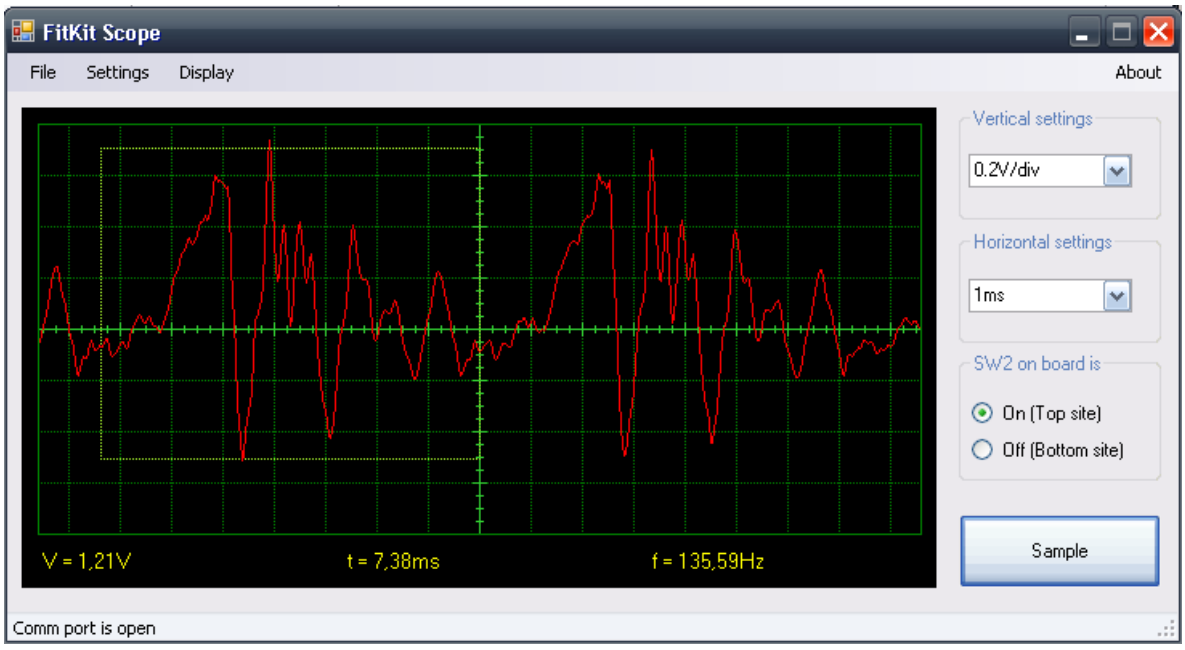
[18] Markovič, J. *Firmware pro ovládání sériového rozhraní*, 2007, produkt dostupný na URL: http://merlin.fit.vutbr.cz/FITkit/docs/firmware/20060414ser.html (prosinec 2007)

[19] QLT Power - Aimtec, *QDC1S-0505S*, dokument k produktu dostupný na URL: http://tme.cz/dok/a16/qdc2s-2405s.pdf (květen 2008)

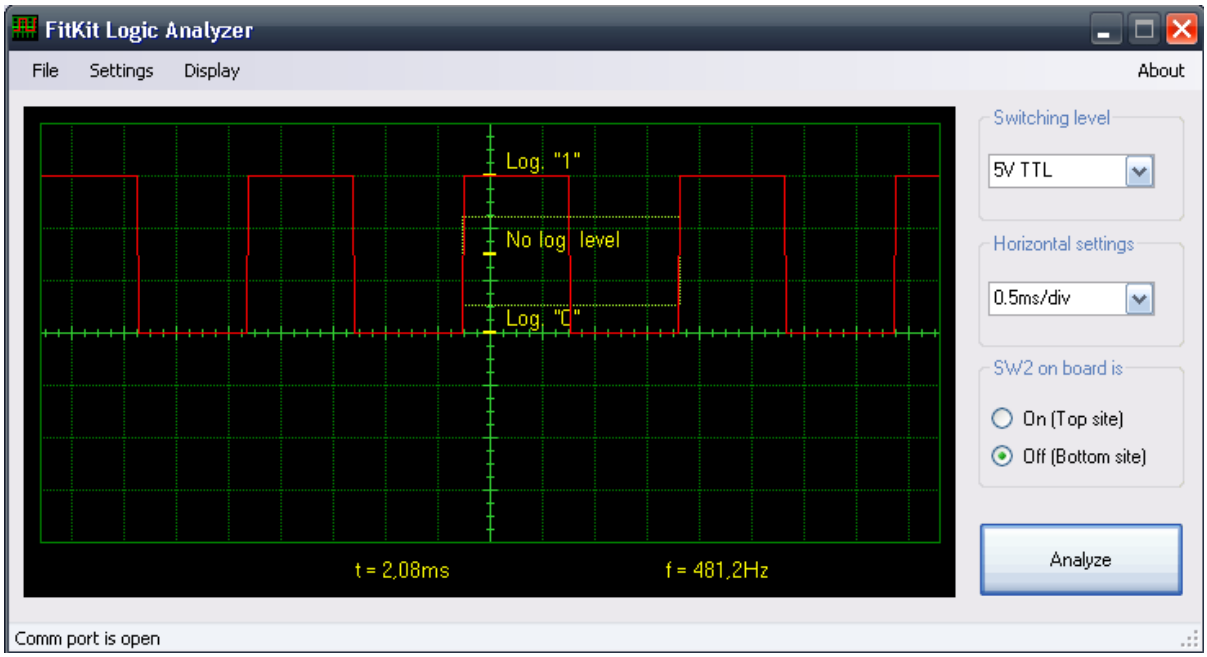
[20] Maxim, *ICL7660,* dokument k produktu dostupný na URL:
http://pdfserv.maxim-ic.com/en/ds/ICL7660-MAX1044.pdf (květen 2008)
[21] CadSoft, *Eagle Light,* produkt dostupný na URL:
http://www.elcad.cz/eagle (květen 2008)
[22] *C Sharp*, článek dostupný na URL:
http://cs.wikipedia.org/wiki/C_Sharp (květen 2008)
[23] Dhond, P. *Selecting the Right Level-Translation Solution*, 2004, dokument dostupný na URL:
http://focus.ti.com/lit/an/scea035a/scea035a.pdf (květen 2008)

Přílohy

Příloha A – Grafické výstupy aplikací

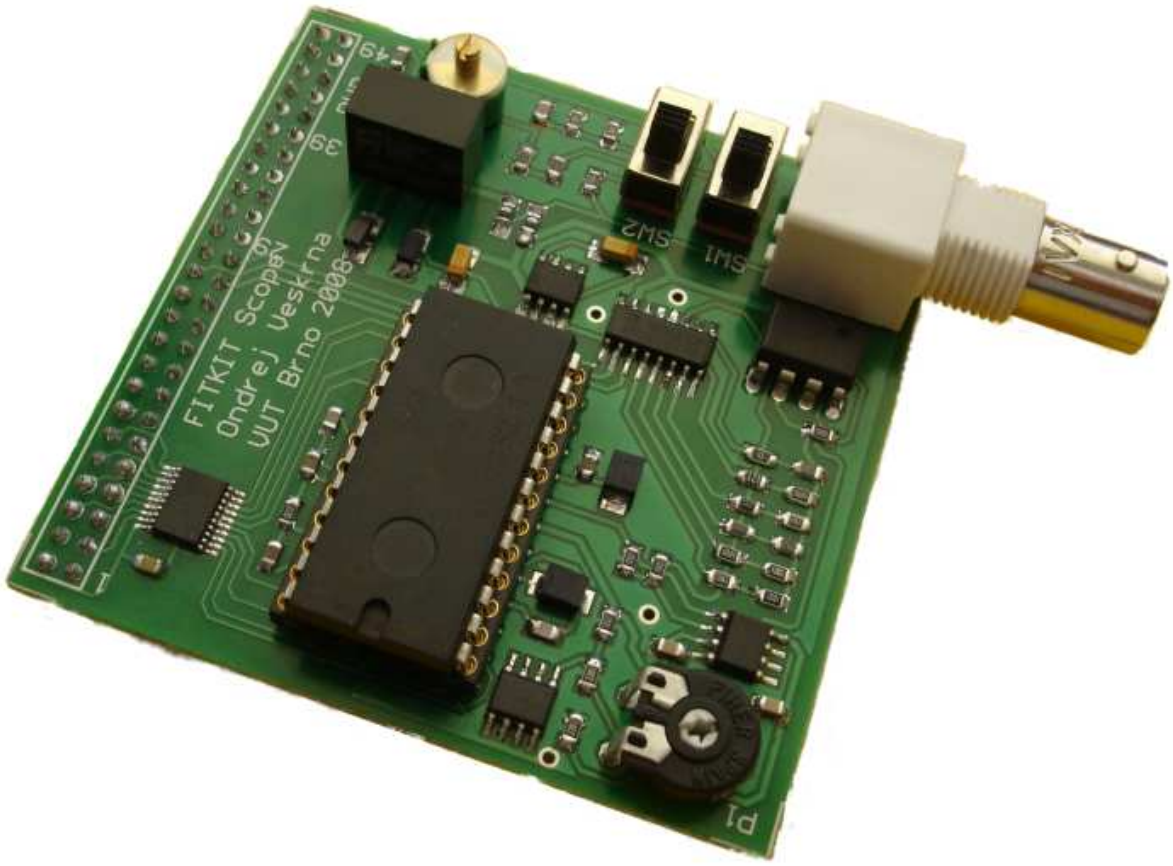


Měřený výstup zvukové karty aplikace FitKit Scope (Pokus o moje komorní „A“)



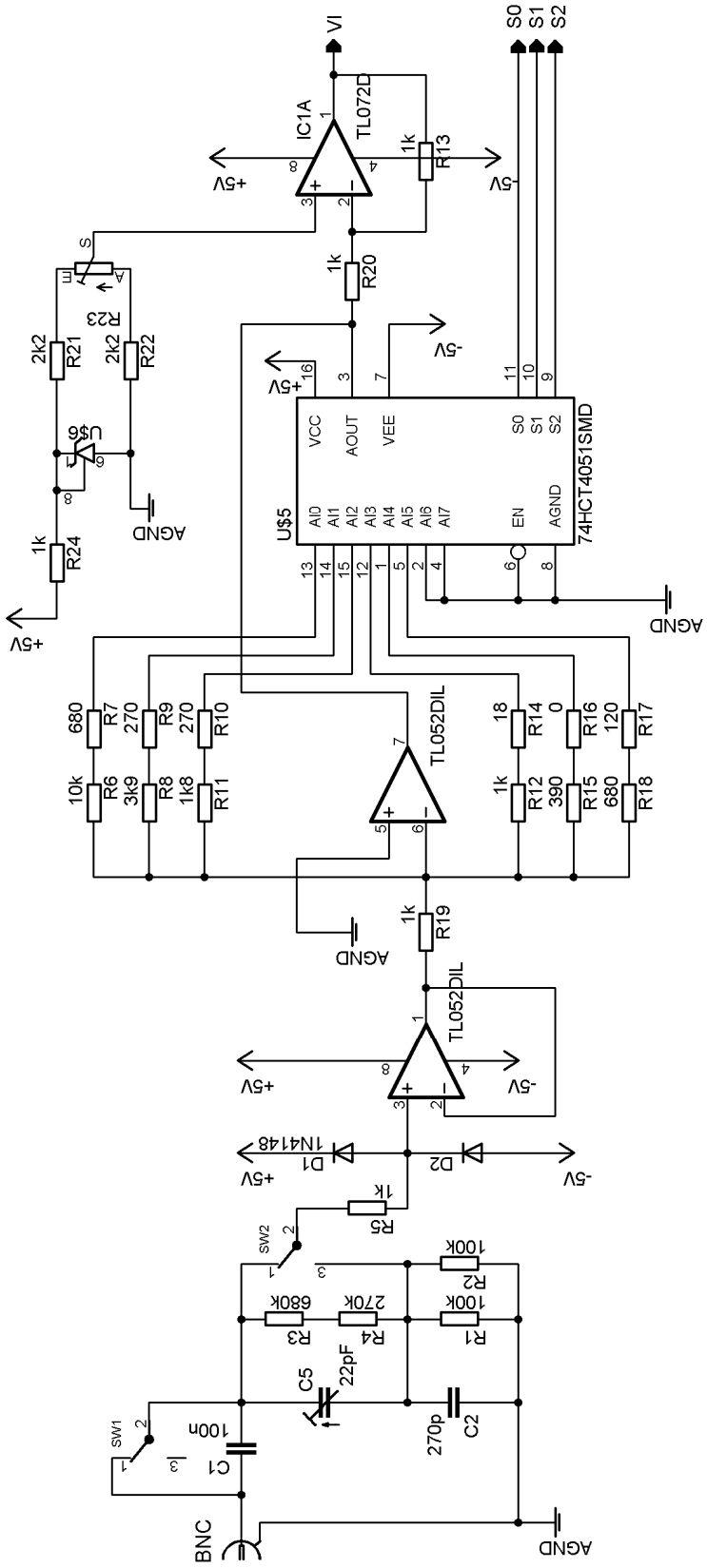
Grafický výstup logického analyzátoru FitKit Logic Analyzer při měření TTL signálu

Příloha B – Osazený osciloskopický modul

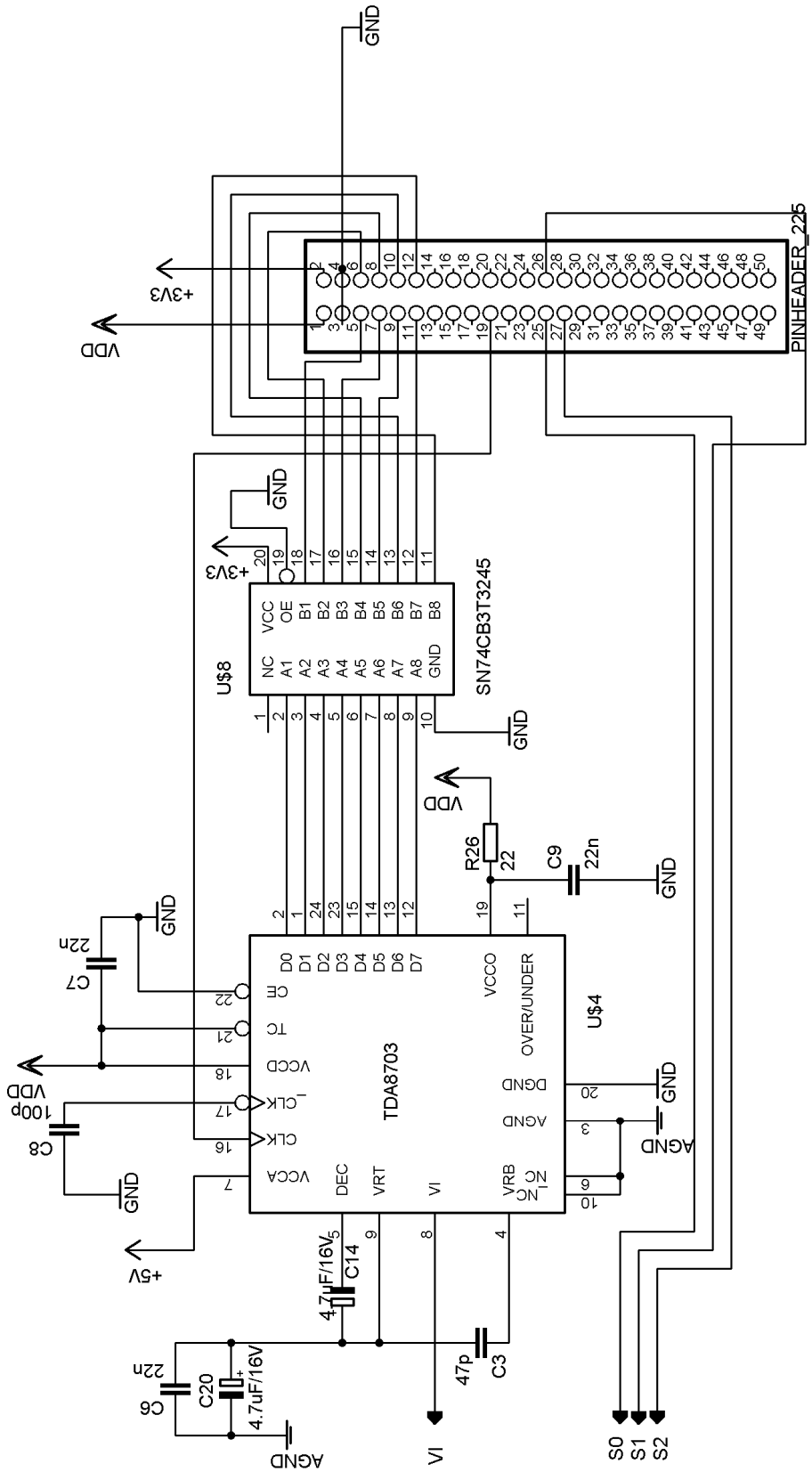


10-1 Osazený osciloskopický modul

Příloha C – Schémata zapojení



Analogová část schéma osciloskopického modulu



Digitální část schéma osciloskopického modulu

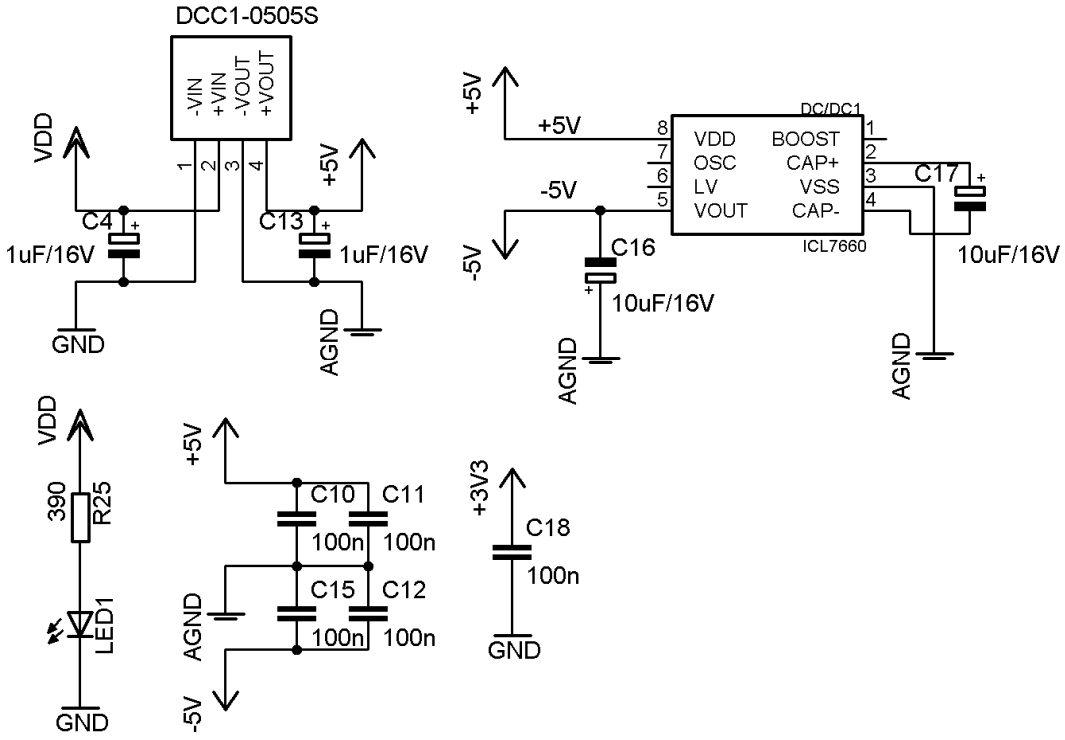


Schéma napájení osciloskopického modulu spolu s blokovacími kondenzátory

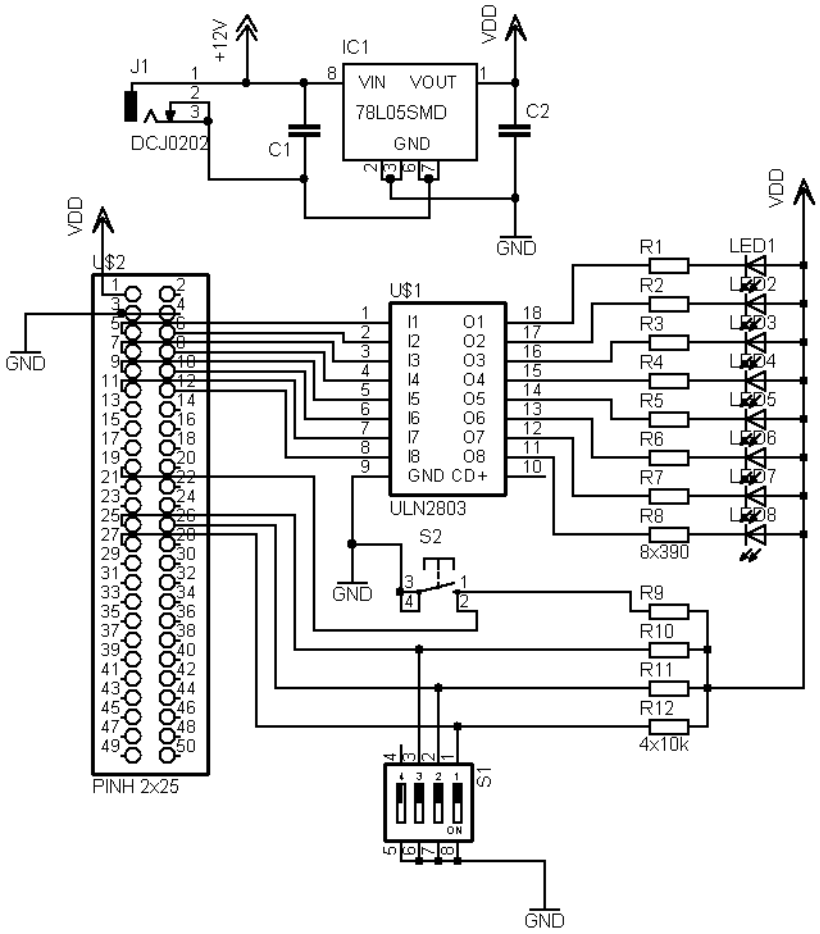
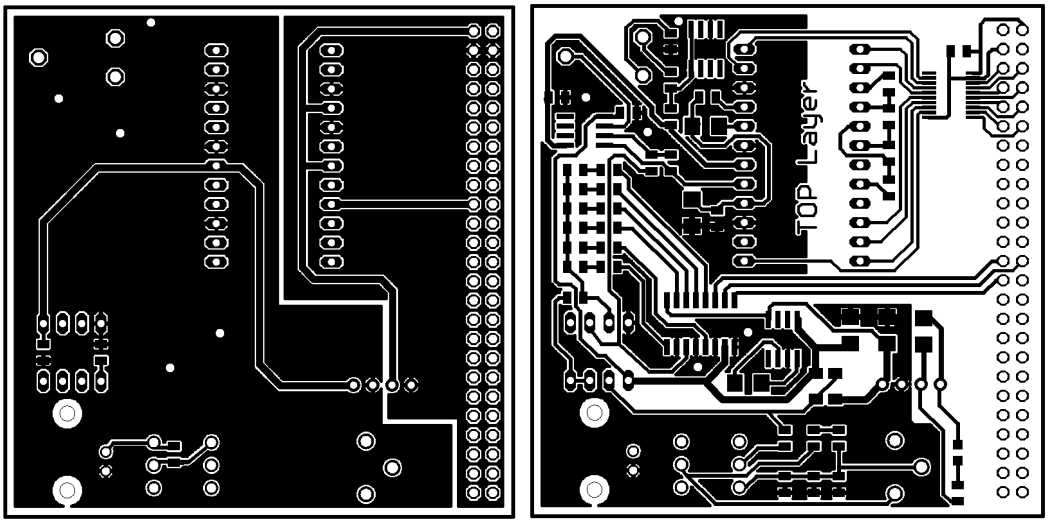


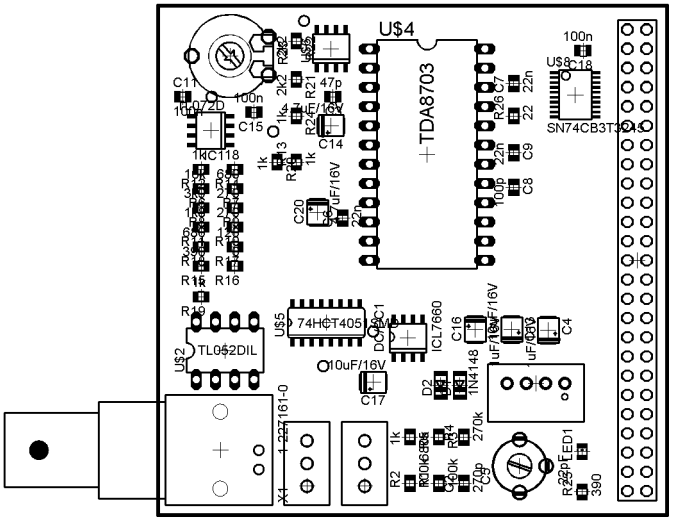
Schéma zapojení testovacího modulu

Příloha D – Návrh desky plošných spojů

Horní a spodní strana spojů osciloskopického modulu



Rozmístění součástek osciloskopického modulu



Strana spojů a rozmístění součástek testovacího modulu