VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

KNIHOVNA PRO VYUŽITÍ AUDIOKODEKU SGTL5000 NA VÝUKOVÉM KITU MINERVA

LIBRARY FOR UTILIZATION OF AUDIOCODEC SGTL5000 ON EDUCATIONAL KIT MINERVA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JIŘÍ HARTMANN**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/17358/2016/xhartm03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Hartmann Jiří**
Obor: Informační technologie
Téma: **Knihovna pro využití audiokodeku SGTL5000 na výukovém kitu Minerva**
Library for Utilization of Audiocodec SGTL5000 on Educational Kit Minerva
Kategorie: Vestavěné systémy

Pokyny:

1. Podrobně se zabývejte architekturou rekonfigurovatelných obvodů FPGA Spartan-6 a jazykem VHDL.
2. Seznamte se s rodinou mikrokontrolerů Kinetis a způsobem jejich programování. Pozornost věnujte zejména typu K60 a jeho periferním modulům I2C a I2S pro sériovou komunikaci.
3. Nastudujte obvodové zapojení výukové platformy Minerva, kde se zaměřte především na část související s obvody MCU, FPGA a zvukovým kodekem SGTL5000.
4. Navrhněte koncepci knihovny pro obsluhu komunikace mezi MCU a zvukovým kodekem SGTL5000.
5. Proveďte implementaci navržené knihovny a vše pečlivě zdokumentujte. Rozhraní knihovny musí umožňovat její snadnou použitelnost.
6. Připravte demonstrační aplikaci, která prokáže funkčnost vámi navrženého řešení.
7. Zhodnoťte dosažené výsledky a pokuste se navrhnout případná vylepšení či rozšíření.

Literatura:

- Dle pokunů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 4 zadání.

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

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

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-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: 1. listopadu 2016
Datum odevzdání: 17. května 2017

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

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Cílem této práce je vytvoření knihovny pro využití audiokodeku SGTL5000, který je přítomen na vývojovém kitu Minerva. Vývojový kit je osazen mikrokontrolérem Kinetis K60 a FPGA Xilinx Spartan-6. Knihovna obsahuje makra a řídící funkce, v jazyce C, pro spuštění a ovládání audiokodeku. Dále obsahuje modul pro komunikaci s audiokodekem přes FPGA pomocí protokolů I2C a I2S. Součástí je také vzorová aplikace, která přehrává a zaznamenává zvukové data na SD kartu.

Abstract

The target of this work is create create library for utilization of audiocodec SGTL5000 which is placed on educational kit Minerva. Educational kit is equipped by microcontroller Kinetis K60 and FPGA Xilinx Spartan-6. Library contain macros and control functions in C language for launch and control audiocodec. Further contain module for communication with audiocodec through FPGA with protocols I2C and I2S. This work also contain demonstration application which play and record sound data on SD card.

Klíčová slova

FITkit Minerva, SGTL5000, I2S, I2C, FPGA, Xilinx Spartan-6, Kinetis K60

Keywords

FITkit Minerva, SGTL5000, I2S, I2C, FPGA, Xilinx Spartan-6, Kinetis K60

Citace

HARTMANN, Jiří. *Knihovna pro využití audiokodeku SGTL5000 na výukovém kitu Minerva.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Šimek Václav.

Knihovna pro využití audiokodeku SGTL5000 na výukovém kitu Minerva

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing.Václava Šimka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Hartmann
18. května 2017

Poděkování

Chtěl bych poděkovat panu Ing.Václavu Šimkovi za cenné rady a jeho trpělivost.

Obsah

Kapitola 1

Úvod

Na zařízeních se, kterýma dnes člověk pracuje je zvukový vstup a výstup považován za samozřejmost. Pro zvuk se důležité aby byl dostatečně kvalitní tzn. bez šumu a dalších rušivých elementů. Digitální obvody při své práci vytvářejí určité rušení a proto je rozhraní pro zvuk tvořeno odděleným obvodem, které se nazývá audiokodek.

Cílem této práce je vytvořit knihovnu pro ovládání audiokodeku. Konkrétně se jedná kodek SGTL5000 od firmy NXP.

Kapitola 2

FPGA

Na kitu Minerva se nachází FPGA (Field Programmable Gate Array) obvod od firmy Xilinx. Konkrétně jde o typ XC6SLX9. Je to specializovaný číslicový obvod, který obsahuje programovatelné bloky spojené programovatelnou maticí spojů. Nastavení jednotlivých bloků a spojů je uloženo v konfiguračním souboru, který se nazývá Bitstream. Bitstream může být uložen ve volatilní paměti, která je přímo v FPGA čipu, nebo ve vnější nevolatilní paměti typu flash. Po připojení napájení proběhne načtení konfiguračního souboru z vnější paměti. Konfigurační soubor lze nahrát pomocí JTAG rozhraní nebo pomocí ICAP rozhraní[1].

Vývody FPGA lze jednotlivě nastavit jako vstupní, výstupní, nebo vstupně/výstupní. Ke každému vývodu lze rovněž nastavit parametry (např. maximální a minimální napětí) podle definovaných standardů. Možnosti vývodů lze nalézt v dokumentaci DS162[10].

Table 9: **Single-Ended I/O Standard DC Input and Output Levels**

I/O Standard	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V, Min	V, Max	V, Min	V, Max	V, Max	V, Min	mA	mA
LVTTL	–0.5	0.8	2.0	4.1	0.4	2.4	Note 2	Note 2
LVCMOS33	–0.5	0.8	2.0	4.1	0.4	V_{CCO} – 0.4	Note 2	Note 2
LVCMOS25	–0.5	0.7	1.7	4.1	0.4	V_{CCO} – 0.4	Note 2	Note 2
LVCMOS18	–0.5	0.38	0.8	4.1	0.45	V_{CCO} – 0.45	Note 2	Note 2
LVCMOS18 (-1L)	–0.5	0.33	0.71	4.1	0.45	V_{CCO} – 0.45	Note 2	Note 2
LVCMOS18_JEDEC	–0.5	35% V_{CCO}	65% V_{CCO}	4.1	0.45	V_{CCO} – 0.45	Note 2	Note 2
LVCMOS15	–0.5	0.38	0.8	4.1	25% V_{CCO}	75% V_{CCO}	Note 3	Note 3
LVCMOS15 (-1L)	–0.5	0.33	0.71	4.1	25% V_{CCO}	75% V_{CCO}	Note 3	Note 3
LVCMOS15_JEDEC	–0.5	35% V_{CCO}	65% V_{CCO}	4.1	25% V_{CCO}	75% V_{CCO}	Note 3	Note 3
LVCMOS12	–0.5	0.38	0.8	4.1	0.4	V_{CCO} – 0.4	Note 4	Note 4
LVCMOS12 (-1L)	–0.5	0.33	0.71	4.1	0.4	V_{CCO} – 0.4	Note 4	Note 4
LVCMOS12_JEDEC	–0.5	35% V_{CCO}	65% V_{CCO}	4.1	0.4	V_{CCO} – 0.4	Note 4	Note 4
PCI33_3	–0.5	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.5	10% V_{CCO}	90% V_{CCO}	1.5	–0.5
PCI66_3	–0.5	30% V_{CCO}	50% V_{CCO}	V_{CCO} + 0.5	10% V_{CCO}	90% V_{CCO}	1.5	–0.5
I2C	–0.5	25% V_{CCO}	70% V_{CCO}	4.1	20% V_{CCO}	–	3	–
SMBUS	–0.5	0.8	2.1	4.1	0.4	–	4	–
SDIO	–0.5	12.5% V_{CCO}	75% V_{CCO}	4.1	12.5% V_{CCO}	75% V_{CCO}	0.1	–0.1
MOBILE_DDR	–0.5	20% V_{CCO}	80% V_{CCO}	4.1	10% V_{CCO}	90% V_{CCO}	0.1	–0.1

Obrázek 2.1: Část tabulky ukazující možné nastavení portu

2.1 Block RAM

FPGA obsahuje RAM paměť uspořádanou po blocích 16 kb. Čip XC6SLX9 obsahuje 32 bloků RAM. Vstupně/výstupní port lze vygenerovat pomocí programu Xilinx CORE Generator. Možnosti výběru portu jsou tyto:

- Single Port RAM
- Simple Dual Port RAM
- True Dual Port RAM

Šířku a hloubku portu lze vybrat po jednotlivých bitech. Generátor poté sestaví výsledný port a zobrazí počet využitých bloků RAM. Zajímavé jsou dvouportové režimy, kde oba porty jsou na sobě nezávislé. To umožňuje do paměti zapisovat a číst různou rychlostí. V Simple Dual Port režimu je jeden port určen pouze pro zápis a druhý pouze pro čtení. V True Dual Port režimu jsou oba porty určeny pro zápis i pro čtení. To sebou přináší možnost vzniku kolize. Kolize nastane pouze, pokud oba porty přistupují na stejnou adresu ve stejný čas a jeden z nich má nastavenou operaci zápisu, dojde ke kolizi, a data na portu s operací čtení jsou nepředvídatelná. Podrobnější popis se nachází v příručce UG383[9].

2.2 Řadič RAM paměti

FPGA obsahuje řadič RAM paměti. Na kitu Minerva je k FPGA čipu připojena paměť DDR2 SDRAM o velikosti 512 Mb (64MB). Řadič generuje signály do čipu paměti a zpřístupňuje paměť na uživatelských portech. Port lze vygenerovat pomocí Xilinx CORE Generator.

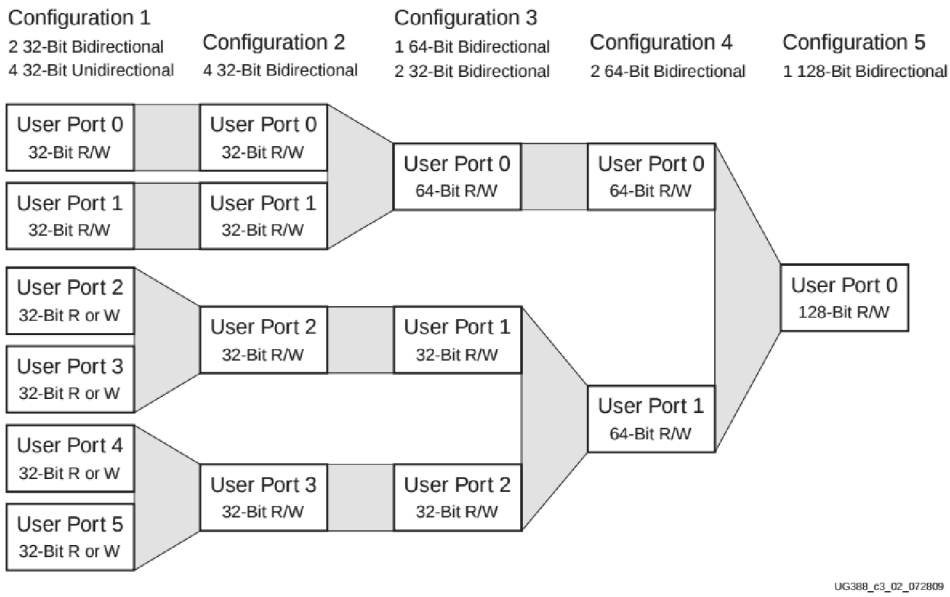


Figure 2-2: **Possible Port Configurations for the User Interface**

Obrázek 2.2: Část tabulky ukazující možné nastavení portu

2.3 Jazyk VHDL

VHDL je jazyk pro popis hardware. Je používán pro návrh obvodů, simulace, případná další využití v paralelních systémech. Zde je jazyk použit pro návrh obvodů, které jsou poté realizovány pomocí FPGA. Jazyk VHLD je obsáhlé téma, zaměřím se na konstrukci registru a konečného automatu.

V jazyku VHDL jsou proměnné tvořeny signály. Signály představují jednotlivé dráty obvodu. V jazyku VHDL se jednotlivé úlohy rozdělují do takzvaných procesů. Všechny procesy běží zároveň. V procesu se nachází přiřazovací příkazy, ty se provedou také zároveň. Pokud je v jednom procesu více přiřazení do jednoho signálu provede se vždy to poslední. Pro vytvoření paměťového místa (registru) vytvoříme dva signály, jeden pro vstup a druhý pro výstup. Potom vytvoříme proces, ve kterém při náběžné nebo sestupné hraně hodinového signálu přiřadí vstup na výstup. Pro řešení úkolu v FPGA se nejčastěji používá konečný automat. Pro popis konečného automatu je vhodný dvou procesový způsob, kdy je v jednom procesu synchronní část a v druhém procesu kombinační část. V synchronní části je třeba registr, pro udržení současného stavu, lze vytvořit způsobem uvedeným výše. Výstupní signál představuje současný stav a vstupní představuje následující stav. Je vhodné signály opatřit vlastním datovým typem, který bude obsahovat pouze názvy jednotlivých stavů. V kombinační části vytvoříme jednotlivé stavy pomocí signálu nesoucího současný stav. V jednotlivých stavech přiřazujeme do signálů hodnoty dle libosti. Na začátku kombinační části je třeba napsat přiřazení výstupů registru na vstupy registrů tím je dosaženo, že když vynecháme přiřazení v některém stavu bude registr ukládat svůj stav a tím držet hodnotu. Také je třeba nezapomenout, že hodnota na výstupu registru objeví až následujícím stavu. [8]

Kapitola 3

Mikrokontrolér K60

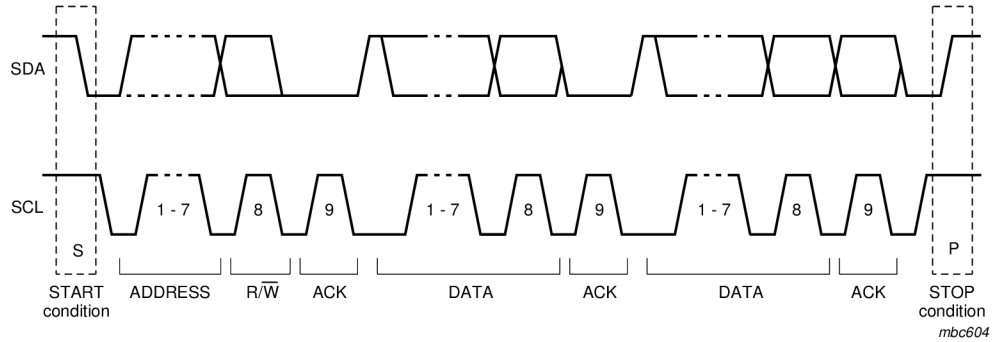
Mikrokontrolér je součástka se zabudovaným procesorem, pamětí a periferiemi. Na kitu je osazen model z K60 od společnosti Freescale. Konkrétně jde o model MK60DN512VMD10. Ten obsahuje 512 KB paměti flash pro program, 128 KB RAM paměti a periferie. Maximální frekvence je 100 MHz. Procesor je typu ARM Cortex-M4. [2] Programování probíhá pomocí rozhraní OSBDM, které je integrováno na kitu Minerva, a aplikace Kinetis Design Studio (KDS). KDS je postaveno na vývojovém prostředí Eclipse a obsahuje v sobě další potřebný software. KDS má také zabudovaný ladící nástroj takzvaný Debugger. Debugger umožňuje procházet kód a sledovat proměnné, což je velice užitečné, protože možnosti výpisu nějakého stavu jsou prakticky omezeny pouze na led diody. KDS obsahuje také režim Procesor Expert (PE). PE umožňuje do projektu přidat takzvané komponenty, které reprezentují jednotlivé části mikrokontroléru. Vlastnosti komponenty lze nastavit pomocí grafického rozhraní. Poté stačí spustit generování PE kódu. To však sebou přináší nevýhodu a to nemožnost zasahování do periférií, které jsou spravovány PE komponentou. Mikrokontrolér obsahuje také dva moduly s I2C sběrnicí a jeden modul s I2S sběrnicí. Tyto porty však nejsou využity.

3.1 I2C

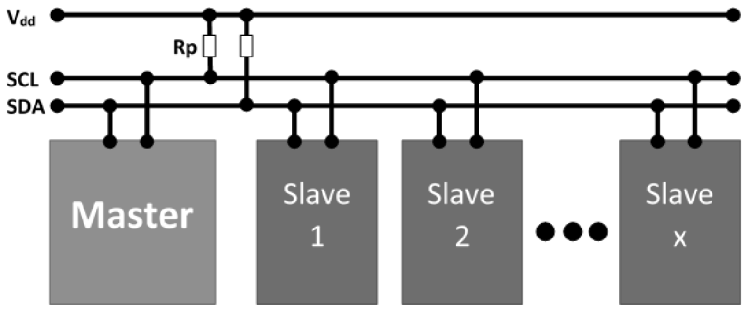
I2C je nízkorychlostní sériová sběrnice vyvinutá firmou Philips (nyní NXP). Skládá se z vodičů SDA (serial data) a SCL (serial clock). Zařízení na sběrnici jsou dvojího typu master a slave. Zařízení typu master zahajuje a ukončuje přenos a generuje hodinový signál na vodič SCL. Výstupy zařízení se nachází buď ve stavu L nebo ve stavu vysoké impedance (Z). Vodiče SDA a SCL jsou připojeny pull-up rezistory na úroveň H. Ve standardní verzi je frekvence stanovena na 100 kHz. Existují i rychlejší režimy přenosu například 400 kHz. Začátek a konec přenosu je indikován start a stop podmínkami. Adresa zařízení je standardně 7 bitů, avšak existuje i 10bitová varianta. Přenos probíhá od nejvíc významového bitu (MSB) po nejméně významový bit (LSB). Logická úroveň na SDA se smí měnit pouze je-li SCL v úrovni L.

3.2 I2S

I2S je sériová sběrnice pro přenos digitálního audio signálu představená firmou Philips (nyní NXP). I2S sběrnice skládá minimálně ze tří vodičů. Vodiče jsou oficiálně pojmenované SCK (continuous serial clock), WS (word select) a SD (serial data). Vyskytují se i jiná označení



Obrázek 3.1: Časový diagram přenosu po sběrnici I2C



Obrázek 3.2: Propojení zařízení přes sbernici I2C

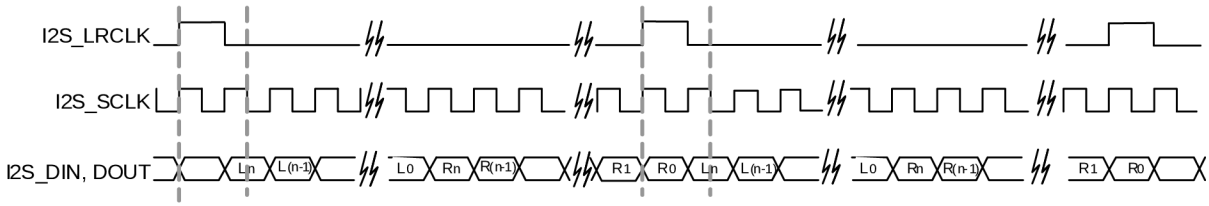
vodičů. Mohou se vyskytovat i další vodiče například pro tok dat i opačným směrem nebo další hodinový signál pro potřeby připojeného zařízení.

Typicky se I2S používá mezi dvěma zařízeními, například zdrojem audio dat a převodníkem digitálního signálu na analogový (DAC — Digital to Analog Converter), protože neobsahuje žádné adresování, nepoužívá pull-up rezistory jako I2C, a má definované vodiče pro vstup a výstup. Je třeba dávat pozor při zapojování.

Vodičem SCK probíhá přenos hodinového signálu. Vodičem SD probíhá přenos dat rychlostí určené vodičem SCK. Data na přijímací straně jsou vzorkována s náběžnou hranou hodinové ho signálu. Hladina vodiče WS určuje, zda se jedná o levý nebo pravý kanál a také rychlost přehrávání vzorků. Jeho perioda se rovná vzorkovací frekvenci přehrávání.

Některá přijímací zařízeni, například DAC, vyžadují pro svoji činnost rychlejší hodinový signál. Tento problém se řeší buď, přidáním dalšího vodiče nebo zrychlením SCK a tedy i dat na SD vodiči.

Audio data jsou přenášena ve formátu lineární pulzně kódové modulace (LPCM) se znaménkem. Znaménko je zakódováno pomocí dvojkového doplňku. Přenos probíhá následovně: Vodič WS je s vodičem SCK změní logickou úroveň a tím vytvoří hranu, s dalším hodinovým taktem dojde k odeslání nejvyššího významového bitu (MSB) a tak dále, až po nejnižší významový bit. Poté se čeká na další hranu vodiče WS. Existují i upravené formáty left-justified a right-justified. Jak již název napovídá, data jsou zarovnána k hraně vodiče WS doleva nebo doprava.



Obrázek 3.3: Diagram přenosu dat přes sbernici I2C

Kapitola 4

Cílová platforma

V této kapitole si představíme audiokodek a jeho možnosti využití na výukovém kitu Minerva. Dále pak protokoly I2C a I2S a neposlední řadě také návrh s ohledem na současný vývojový stav.

4.1 Audiokodek

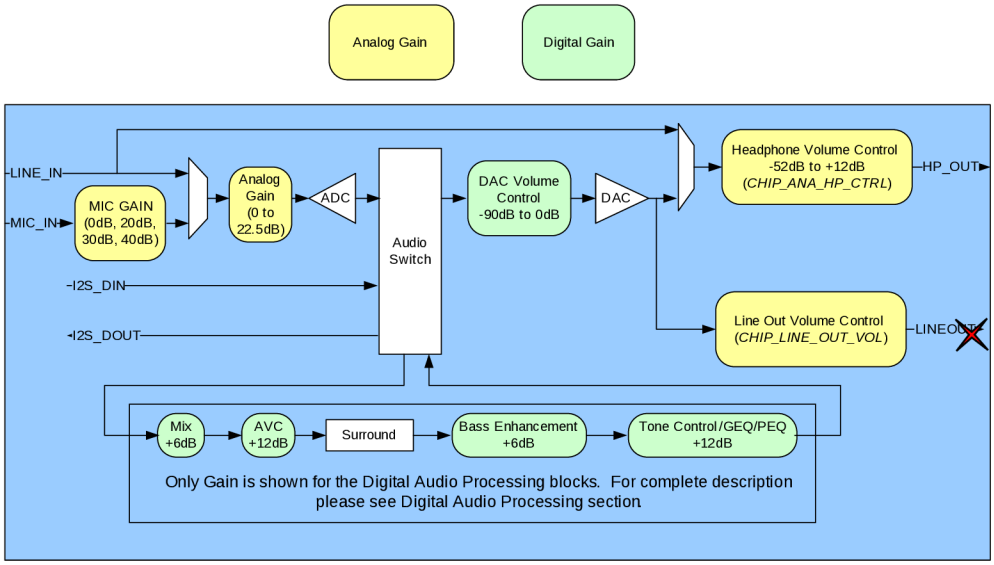
Na kitu je osazen audiokodek SGTL5000 od firmy NXP (bývalý Freescale Semiconductor). Kodek má vyveden stereo výstup na sluchátka do konektoru typu jack s maximální zátěží 16 Ω. Kodes disponuje také výstupem lineout. Ten však není na desce kitu nikam připojen. Druhý konektor je určen pro vstup a je připojen na propojky. Přes propojky si lze vybrat připojení, buď na stereo linkový vstup nebo na vstup mikrofonu. Vstup mikrofonu umožňuje větší zesílení. Kodek má oddělené napájecí větve pro digitální a analogové obvody. V rámci jednoduchosti jsou obě větve napájeny stejným zdrojem. V analogové větvi je vložena cívka jako filtr. Zdroj hodinového signálu poskytuje externě připojený krystal o frekvenci 24,576 MHz. Kodek již obsahuje soustavu děliček a násobiček pro svou činnost i pro odvození vzorkovací frekvence. Ta může nabývat hodnot 8 kHz, 11,025 kHz, 12 kHz, 16 kHz, 22,050 kHz, 24 kHz, 32 kHz, 44,1 kHz, 48 kHz a 96 kHz. Řízení probíhá zápisem patřičných hodnot do patřičných registrů. Ty mají velikost 16 bitů a jsou zpřístupněny rozhraním I2C. Zvukové data jsou přijímána a odesílána přes rozhraní I2S. Jako bonus obsahuje modul nazvaný Digital Audio Processing (DAP), který zahrnuje řadu funkcí jako například tři druhy ekvalizéru, zlepšení basů, automatickou kontrolu hlasitosti, a tak dále.

Velikosti vzorků, které kodek dokáže zpracovat jsou 16, 20, 24 a 32 bitů. Dokumentace udává maximální odstup signál/šum (SNR) 100 dB. Podle dokumentace se odstup signál/šum v různých režimech pohybuje okolo 97 dB. Data jsou ve formátu lineární pulzní kódové modulace(LPCM). [6]

4.2 Návrh a zapojení

Úkolem je vytvořit komunikační rozhraní mezi MCU a audiokodekem a následně knihovnu pro jeho obsluhu. Jak již bylo zmíněno k registrům audiokodeku lze přistupovat přes sbernici I2C. Z obvodového schématu je patrné, že audiokodek není připojen přímo k MCU, ale je připojen k FPGA a to jak I2C tak I2S.

První variantou je použít natažené vodiče mezi MCU a FPGA jako I2C a I2S přepnutím příslušných multiplexorů v MCU. Poté do FPGA navhnout konfiguraci, která umožňovala



Obrázek 4.1: Zjednodušený vnitřní blokový diagram

předávání signálů mezi vývody FPGA. Tato možnost však naráží na problémy. První je přemostění signálů sběrnice I2C přes vývody FPGA. Možné problémy může způsobit zpoždění přes registry v FPGA. Další problém je nevhodnost z hlediska dalšího vývoje platformy Minerva. Vodiče mezi FPGA a MCU jsou určeny pro sběrnici FlexBus, která má sloužit pro přenos dat mezi MCU a FPGA. Použití I2C a I2S by znemožnilo používat FlexBus zároveň.

Druhá varianta je použití rozhraní FlexBus pro přenos dat a v FPGA vytvořit komponenty pro rozhraní I2C a I2S. Tato varianta nebrání použití ostatních komponent komunikujících s MCU avproto je výhodná z hlediska dalšího vývoje platformy. Komunikace přes FlexBus již byla zprovozněna, takže by mělo stačit použít již hotový modul.[1]

4.2.1 FlexBus

FlexBus je sběrnice pro připojení externích zařízení k mikrokontroléru Kinetis jako jsou paměťové moduly, programovatelné logické obvody, atd. FlexBus rozhraní je s FPGA spojeno 41 vodiči, z nichž 32 je určeno pro přenos adresy a dat. Z předchozích prací [1] [3] jsem prozkoumal výše zmíněný modul v FPGA a zjistil jsem, že je použit pro programování flash paměti, která obsahuje konfigurační soubor pro FPGA. Parametry modulu jsou 16 bitů adresa a 16 bitů data. Hodinový signál je odvozen z FlexBus sběrnice. Hodinový signál pro FlexBus je určen generátorem, který zajišťuje hodinový signál pro mikrokontrolér a jeho periferie. V předchozích pracích je hodinový signál nastaven na 96MHz pro jádro mikrokontroléru a 48 MHz pro externí periferie a sběrnice, tedy i FlexBus. Je to z důvodu potřeby USB rozhraní.

Komponenta v FPGA zpracovává FlexBus signály a nabízí již oddělené signály pro adresu a čtená, zapisovaná data. Adresa byla předchozím autorem zvolena 16bitová a rozdělena na půl. Horní část adresy slouží pro adresaci jednotlivých komponent a spodní část pro adresaci uvnitř komponenty. Rozhodl jsem se tuto konvenci zachovat. Avšak datová část byla také 16bitová a tu jsem se rozhodl rozšířit na 32 bitů a to kvůli rychlejšímu přenosu.

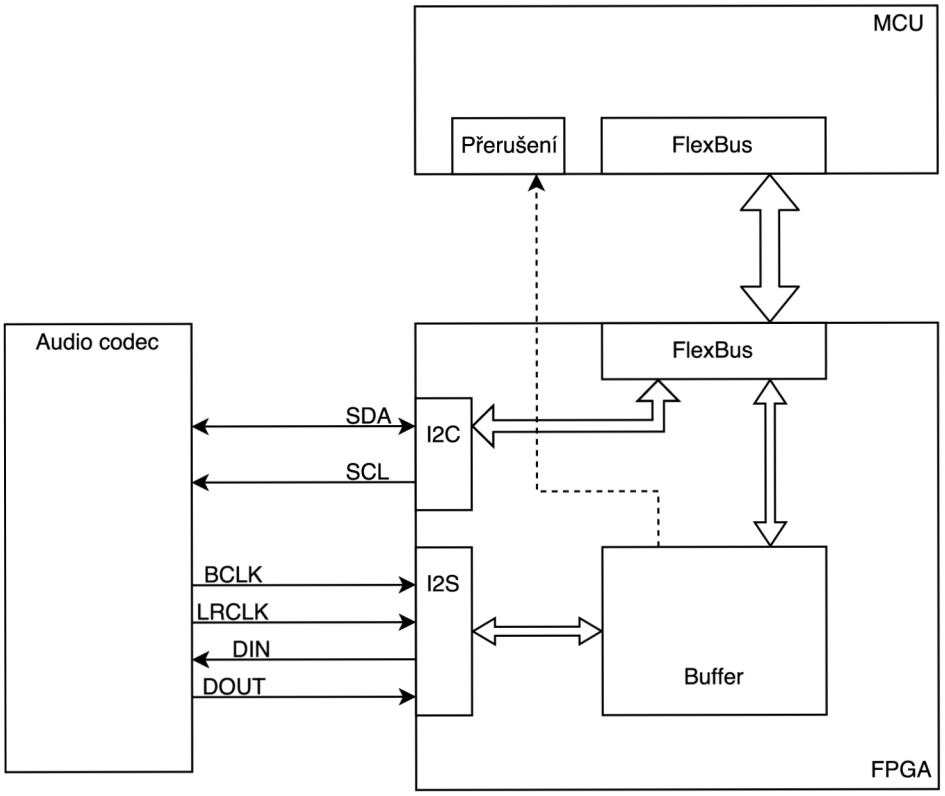
Přenos probíhá po transakcích. Transakci zahajuje vždy mikrokontrolér a ten je blokován do doby, než adresovaná komponenta potvrdí transakci.

4.3 Řešení

Rozhodl jsem se vytvořit dvě komponenty. Jedna bude číst a zapisovat registry v audiokodeku přes I2C, řídící část. Druhá se bude starat o přenos audio dat, datová část.

Řídící část bude pracovat následovně: V mikrokontroléru se aktivuje funkce pro čtení nebo zápis registru audiokodeku, ta vygeneruje transakci na FlexBus sběrnici. FlexBus komponenta předá data řidící komponentě, ta odešle nebo příjme data pomocí komponenty zpracovávající I2C a pak ukončí transakci. Vzhledem k množství přenášených dat by tato operace neměla výrazně ovlivnit chod mikrokontroléru.

S datovou částí je to složitější. Nejdřív je třeba vyřešit odkud bude brán hodinový signál, který bude zajišťovat správnou vzorkovací frekvenci a tím i správnou rychlost přehrávání. Nejjednodušší řešení poskytuje audiokodek sám. K audiokodeku je připojen krystal jako generátor referenčního hodinového signálu. Obvod uvnitř audiokodeku zajistí, že po správném nastavení vytváří požadovanou frekvenci. Tím je audiokodek na sběrnici I2S master a modul v FPGA bude přijímat a odesílat data. Data se nyní z audiokodeku dostávají do FPGA. Jelikož transakci pro přenos dat může zahájit pouze mikrokontrolér, je třeba mikrokontrolér upozornit a to nejlépe pomocí přerušení. Přerušení může být vedeno třeba externím drátem mezi FPGA a MCU vývody. Avšak rychlost přerušení by byla příliš velká a mikrokontrolér by nemusel stíhat, je nutné vytvořit vyrovnávací paměť takzvaný buffer na straně FPGA. Nabízí se opět několik řešení. První varianta co mě napadla bylo použití Block RAM paměti zabudované v FPGA. Dále se nabízí použít RAM řadič, ke kterému je připojena DDR2 paměť. Poslední možnost je připojit externí paměť k FPGA. Po konzultaci s vedoucím práce jsem se rozhodl použít Block RAM paměť.



Obrázek 4.2: Návrh komunikace mezi Mikrokontrolérem a audiokodekem

Kapitola 5

Knihovna

Kapitola pojednává o výsledné knihovně a popisu jejich části. Knihovna je rozdělena na dvě logické části. Řídící část zahrnuje obsluhu registrů audiokodeku v jazyce C a způsob provedení. Datová část se zabývá přenosem dat mezi audiokodekem a mikrokontrolérem.

5.1 Vývoj

Rozšíření rozhraní FlexBus sběrnice na 32 bitů vyžadovalo drobné úpravy, například nastavení příslušných vývodů. Nebylo však zřejmé, že dojde i ke změně adresového prostoru. Po změně na 32 bitů jsou poslední dva bity adresy nulové. To je způsobeno přístupem k FlexBus sběrnici pomocí mapovaní do adresového prostoru mikrokontroléru. Adresový prostor mikrokontroléru je rozdělen po jednotlivých bajtech, což je 8 bitů. Při zápisu 32bitového čísla dojde k zápisu do 4 bajtů paměti, tudíž další hodnota může být na adrese o 4 větší. Tenhle fakt narušil již vytvořené komponenty, u kterých bylo nutné změnit adresy.

Než jsem si tento fakt uvědomil, prováděl jsem i zápisy, které nejsou násobky čtyř. Mikrokontrolér se občas zasekl jinak to na sobě nedal znát. Odeslaná data byla samozřejmě špatná.

5.2 Řídící část

Přístup do registrů audiokodeku se provádí pomocí I2C sběrnice, která je připojená přímo na vývody FPGA. Na trase mezi audiokodekem a FPGA nejsou žádné další součástky, které I2C sběrnice potřebuje ke své funkci. V FPGA jsou proto zapnuty pull-up rezistory, které plně nahrazují vnější rezistory.

O generování I2C signálu se stará modul `i2c_master`, který je dostupný na internetu [4]. Modul v sobě obsahuje děličku hodinového signálu. Dělička je nastavitelná pomocí dvou konstant na začátku souboru. Vstupní hodnota je nastavená na 48 MHz, což je hodinový signál FlexBusu. Výstupní hodnota je nastavená na 400 kHz, což je hodinový signál I2C.

Nad I2C komponentou je vytvořena komponenta `flex_audio_control`, která připojena paralelně k FlexBus komponentě. `flex_audio_control` slouží k předávání transakcí do I2C modulu. Při zápisu do registru audiokodeku je horních 16 bitů použito jako adresa registru a spodních 16 bitů jsou zapisovaná data. Čtení registru audiokodeku probíhá pomocí dvou transakcí. První transakce zapíše do komponenty adresu registru, ze kterého se bude číst. Následuje čtecí transakce, která vrátí přečtená data.

V knihovně na straně jsou funkce `void audio_reg_write(uint16_t address, uint16_t data)` a `uin16_t audio_reg_read(uint16_t address)`. Ty zajišťují zápis a čtení registrů audiokodeku. Všechny ostatní funkce v konečném důsledku využívají tyto dvě.

Funkce `audio_reg_clear` nastaví bity na nulu podle zadané masky. Obdobně pracuje `audio_reg_set`. `audio_reg_value` zapíše hodnotu do registru na patřičné místo podle masky a posunu. Tyto manipulační funkce změní požadované hodnoty. Ostatní hodnoty zůstanou nezměněny.

5.3 Datová část

Datová část zajišťuje přenos dat mezi audiokodekem a mikrokontrolérem. Komponenta `flex_audio_data` obsahuje dva konečné automaty a podkomponentu `audio_buffer`. Komponenta `audio_buffer` obsahuje vygenerované porty pro přístup do Block RAM paměti. BRAM musí být nakonfigurována tak, aby se dala rozdělit na dvě stejné poloviny, jejíž poloviny určuje nejvýznamnější bit adresy (MSB). To přináší jednoduchost, v podobě přepínání mezi polovinami a omezení velikosti paměti na mocniny čísla 2.

5.3.1 I2S část

Audiokodek přijímá a odesílá data prostřednictvím I2S sběrnice. Audiokodek je nastaven do režimu master. Poskytuje tedy hodinový signál do FPGA. Vzorkování dat probíhá na staně audiokodeku s každou náběžnou hranou. Konečný automat pracuje s hodinový signálem z I2S. Je však nastaven na sestupnou hranu, aby změny na I2S vodičích byly prováděny v jiný časový interval, než změny na straně audiokodeku. Počet přenášených bitů je stanoven na 16. Hodinový signál `I2S_BCLK` je nastaven na hodnotu $64 \times Fs$ (Fs — vzorkovací frekvence). Odeslání jednoho vzorku, tj. 16 bitů, proběhne po změně signálu `I2S_LRCLK`. Po každém odeslání vzorku zbývá 15 hodinových taktů. To postačuje pro přístup do BRAM.

Pro přistup k BRAM je použit port B. Ten je nastaven na šířku 16 bitů a tudíž každá adresa odpovídá jednomu 16bitovému vzorku. Posun adresy je zajištěn pomocí zvyšujícího se čítače. Aby nedošlo k prohození vzorku pro levý a pravý kanál, například vlivem nejistého počátečního stavu, je spodní bit adresy řízen nepřímo signálem `I2S_LRCLK`. Šířka signálu čítače je tedy o 1 bit menší, než šířka adresového signálu portu B. Hodnota čítače se tedy mění až s každým druhým vzorkem.

Obousměrný přenos probíhá následovně: Nejdřív je přečteno paměťové místo jehož pozici udává čítač v kombinaci se signálem udávajícím kanál. Poté se čeká na změnu `I2S_LRCLK` po změně je odesílán vzorek načtený z paměti a zároveň přijímán vzorek z audiokodeku. Po výměně všech bitů je přijatý vzorek zapsán na stejné místo jako byl odesílaný vzorek čten. Pak dojde ke změně adresy na portu B a je čten nový vzorek. Poté se znovu čeká na změnu `I2S_LRCLK`.

Jak již bylo zmíněno nejvýznamnější bit adresy určuje přesnou polovinu paměti. Ten je hlídán a při jeho změně je vygenerováno přerušení do mikrokontroléru. Dále je použit k řízení přístupu na portu A a to mě přivádí k popisu druhého konečného automatu.

5.3.2 FlexBus část

Druhý konečný automat komunikuje s BRAM pomocí portu A. Port A je nastaven na šířku 32 bitů. Konečný automat obsluhuje transakce FlexBus komponenty. Adresování paměti BRAM je tvořeno čítačem, jehož signál je o jeden bit menší než signál adresy BRAM.

Nejvyšší významový bit signálu adresy portu A je řízen negací nejvyššího významového bitu adresy portu B. Tento mechanizmus znemožňuje přístup do poloviny paměti, se kterou pracuje I2S.

Poté co mikrokontrolér obdrží přerušení, musí stihnout provést změny, než dojde k překlopení do další poloviny paměti. Při přehrávání stačí data do paměti zapsat. Při nahrávání i přehrávání je nutné data nejprve přečíst a poté zapsat přehrávaná data. Posun adresy probíhá čítačem po každé transakci. V hlavičkovém souboru knihovny proto musí být uvedeno počet řádků paměti při 32bitech vydělený dvěma, aby byl zajištěn přenos vzorků bez ztrát.

Kapitola 6

Použití knihovny

Použití knihovny je následující. Nejprve je třeba provést počáteční nastavení FlexBus sběrnice pomocí funkce `flex_init` a poté zkontrolovat, zda FPGA obsahuje konfiguraci s FlexBus komponentou a to pomocí funkce `flexbus_recheck`. Ta je obzváště duležitá, protože nastavuje FlexBus do potvrzovacího režimu. Bez nich knihovna nebude fungovat. Pak je vhodné zkontrolovat ovládací část. K tomu slouží funkce `audio_check_connection`. Odpověď najdeme v proměnné `audio_connectionOK`. Kontrola je zajištěna přečtením registru, který obsahuje identifikátor audiokodeku. Poté jsou k dispozici všechny ovládací funkce a makra. Audiokodek je po připojení napájení ve vypnutém stavu, je třeba jej zapnout a nastavit k tomu slouží funkce `audio_init`. Funkce také nastavuje vzorkovaní frekvenci tu je možné měnit i po inicializaci.

Inicializační funkce obsahuje poměrně dlouhou sekvenci příkazů a dvě aktivní čekání. Každé po dobu zhruba 400ms. Je to z důvodu omezení prasknutí při zapínání sluchátkového zesilovače. Inicializační zapne vstup i výstup, odblokuje tlumení (mute) a nastaví výchozí hlasitost, avšak nezasahuje do nastavení trasy zvuku.

Trasa zvuku se nastavuje výběrem zdroje signálu na vstupech. Například na vstup chci pro propojit ADC s DAC tak využiji `AUDIO_DAC_INPUT_ADC`.

Dále je třeba správně nastavit přerušení od FPGA na obslužnou funkci `audioISR`. Ta obstará odesílání vzorků do vyrovnávací paměti v FPGA. Pro zahájení přehrávání, nahrávání případně obojího zároveň slouží funkce `audio_play`, `audio_record` a `audio_play_record`. Ty to funkce obsahují povinný parametr a to je adresa funkce, která předá data do připravené paměti.

Funkce audiokodeku, kterým stačí zapnutí, vypnutí, případně několik hodnot, jsou tvořeny makry. Typickým příkladem je funkce mute. Použití makra zmenšuje velikost kódu, který musí být nahrán do mikrokontroléru a také nezaplňuje zásobník zbytečným voláním funkce. Protipólem je obsluha ovládání hlasitosti. Hlasitost je ovládána příslušnými funkcemi a to rovnou v decibelech. Funkce zajistí příslušný převod hodnoty do registru.

Kapitola 7

Demonstrační aplikace

Pro potvrzení funkčnosti a otestování zda audiokodek funguje byla vytvořena demonstrační aplikace. Aplikace přehrává a nahrává soubory ve formátu WAV. Soubory jsou uloženy na SD kartě v souborovém systému FAT. K tomu využívá knihovny pro obsluhu SDHC karty.[5] Tlačítky SW5 a SW3 lze regulovat výstupní hlasitost. Led diody jsou využity pro kontrolu rychlosti obsluhy přerušení.

V aplikace pracuje se soubory ve formátu WAV. Ten však v sobě musí nést data ve formátu PCM 16 bitů little-endian a dva kanály. V konverzních programech je typicky pojmenován `pcm_s16le`.

Aplikace ve výchozím nastavení používá vyrovnávací paměť o velikosti 32 kB.

Kapitola 8

Testování a výsledky

Testování probíhalo za pomocí sluchátek, mikrofonu a obyčejné SD karty ADATA 8 GB SDHC Class 4.[11] Chyby byly odhalovány poslechem a pomocí led diod, které zobrazují rychlost přerušení a dobu trvání přerušení.

Přehrávání souboru bylo testováno se vzorkovací rychlostí 44,1 kHz, 48 kHz a 96 kHz. Velikost vyrovnávací paměti byla zvolena tak, aby pokryla jeden blok paměti BRAM což je 16 kb. Mikrokontrolér tedy zpracovával části o velikosti 8 kb. V přepočtu na bajty je to 512 bajtů, což je nejmenší adresovatelný blok na SD kartě. I přes vysokou rychlost přerušování mikrokontroléru nebyl zaznamenán žádný výpadek.

Zaznamenávaní do souboru bylo testováno obdobným způsobem. Byly však zaznamenány značné výpadky. Výsledný záznam prakticky nebyl použitelný. Možným řešením bylo zvětšit vyrovnávací paměť a tím i velikost zapisovaných bloků na SD kartu. Vyrovnávací paměť jsem zvětšil až na hodnotu 32 kB. To obsadí polovinu dostupné BRAM paměti. Velikost bloků zapisovaných na kartu je 16 kB.

Záznam již byl použitelný, však docházelo k výpadkům po zhruba 20sekundách záznamu. Po prozkoumání bylo zjištěno, že obslužnou rutinu brzdí zápisy na SD kartu. Dalším možným zlepšením je snížení režie souborového systému FAT. Použitá knihovna FatFs[7] umožňuje předalokovat souvislou část paměti, tím nemusí probíhat záznam do FAT tabulky s každým zápisem.

Výsledek se zlepšil. Minutu až minutu a půl je záznam bez záseku. Poté se objeví záseky. V programu Audacity jsem měřil čas mezi záseky. K záseku docházelo zhruba po 25 sekundách.

Následovalo další zvýšení vyrovnávací paměti a to na hodnotu 64 kB, což je maximální velikost paměti BRAM. Velikost bloků zapisovaných na kartu tím vzroste na 32 kB. Výsledek byl prakticky totožný s předchozím výsledkem. Zhruba minutu a půl byl záznam bez záseku. Poté nastaly záseky. Měření v Audacity ukázalo opět záseky po 25 sekundách. Ke zlepšení tedy vůbec nedošlo.

Po konzultaci s vedoucím práce jsem použil relativně novou kartu ADATA microSDHC 16GB UHS-I. Karta byla zasunuta do SD slotu pomocí pasivní redukce. S touto kartou jsem dosáhl výrazného zlepšení a to dva záseky na desetiminutovém záznamu. Poté jsem zkoušel snížit velikost vyrovnávací paměti zpět na 32 kB. Výsledek byl osm náhodných záseků na desetiminutovém záznamu.

Po uvážení jednotlivých výsledků je patrné, že SD karta nemá konstantní dobu zápisu. Doba zápisu karty záleží na jejím vnitřním stavu.

Kapitola 9

Možnosti dalšího vývoje

Vývojový kit Minerva disponuje řadou možností kde se dá zlepšovat. Již výše uvedených výsledků je patrné, že na plynulý přenos dat do SD karty se nedá spolehnout. Jedna z možností je zvolit jinou strukturu vyrovnávací paměti. Možným řešením je využít FIFO frontu. Do jedno konce by se vkládala data a z druhého konce by byla vybírána a zapisována na SD kartu. V FPGA je možné vytvořit tuto strukturu. Problém nastává jak často a jak moc frontu vybírat. Při prodlevách karty by bylo třeba parametry rychlosti výběru přizpůsobovat dynamicky. Mikrokontrolér obsahuje také DMA řadiče a ty lze použít pro přenos dat přes sběrnici FlexBus. To by však vyžadovalo zpřístupnit adresový prostor paměti na sběrnici FlexBus. Vyhodným řešením by bylo použití DDR2 paměti jako vyrovnávací buffer. Audiokodek by se k ní připojil přímo v FPGA. Adresový prostor paměti by byl zpřístupněn mikrokontroléru přes FlexBus rozhraní. Pro přenos dat by šlo využít DMA řadič.

Kapitola 10

Závěr

Cílem této bakalářské práce bylo vytvořit knihovnu pro obsluhu komunikace mezi mikrokontrolérem a zvukovým kodekem SGTL5000. Knihovna je určena pro výukovou platformu Minerva. Rozhraní knihovny by mělo umožňovat její snadnou použitelnost. Po důkladném nastudování obvodového schématu jsem provedl návrh koncepce komunikace mezi mikrokontrolérem a zvukovým kodekem SGTL5000 s ohledem na současný vývoj výukové platformy Minerva. Vzhledem ke specifickému propojení zvukového kodeku a mikrokontroléru přes obvod FPGA, bylo potřeba prostudovat již vytvořené práce, jejichž části by se dali využít. Vybraný návrh jsem poté implementoval s použitím již vytvořených modulů. Implementace části pro mikrokontrolér Kinetis K60 byla vyvinuta ve vývojovém prostředí Kinetis Design Studio verze 3.2.0. Implementace části pro FPGA byla vyvinuta ve vývojovém prostředí Xilinx ISE verze 14.7. Vytvořil jsem také demonstrační aplikaci, která umožňuje přehrávat a zaznamenávat zvuk uložený v souborech na SD kartě. Pomocí demonstrační aplikace jsem provedl testy. Výsledky testů potvrzují funkčnost přehrávání dat ze souboru na SD kartě, avšak dále z nich vyplývá problém s ukládáním zvukových dat na SD kartu. S použitím několika vylepšeními se podařilo dosáhnout o hodně lepších výsledků. I přesto, že byl problém podstatně minimalizován nedošlo k jeho úplnému odstranění. Problém způsobuje nestálá doba zápisu SD karty. Vytvořená knihovna je využitelná v dalších pracích, které budou potřebovat pracovat se zvukem. Knihovna usnadňuje nastavování jednotlivých funkcí zvukového kodeku a obsahuje také proceduru, která zajistí spuštění kodeku.

Literatura

[1] Buchta, P.: *Framework pro vývoj aplikací na platformě ARM*. Vysoké učení technické v Brně. Fakulta informačních technologií, 2015.

[2] Freescale: *Data Sheet: Technical Data, K60P144M100SF2V2*. Červen 2013, [Online; navštíveno 14.05.2017].
URL http://cache.freescale.com/files/32bit/doc/data_sheet/K60P144M100SF2V2.pdf

[3] Krůpa, T.: *Univerzální zavaděč pro mikrokontrolér Kinetis K60*. Vysoké učení technické v Brně. Fakulta informačních technologií, 2016.

[4] Larson, S.: *I2C Master (VHDL)*. Únor 2015, [Online; navštíveno 14.05.2017].
URL https://eewiki.net/pages/viewpage.action?pageId=10125324

[5] Nemček, O.: *Knihovna pro komunikaci mikrokontroleru Kinetis K60 s SD kartou*. Vysoké učení technické v Brně. Fakulta informačních technologií, 2015.

[6] NXP: *SGTL5000 Low Power Stereo Codec with Headphone Amp*. Listopad 2013, [Online; navštíveno 14.05.2017].
URL http://www.nxp.com/assets/documents/data/en/data-sheets/SGTL5000.pdf

[7] stránky, W.: *FatFs - Generic FAT File System Module*. Březen 2017, [Online; navštíveno 14.05.2017].
URL http://elm-chan.org/fsw/ff/00index_e.html

[8] Vašíček, Z.: *Popis HW pomocí VHDL*. Květen 2009, [Online; navštíveno 14.05.2017].
URL http://merlin.fit.vutbr.cz/FITkit/docs/navody/synth_templates.html

[9] Xilinx: *Spartan-6 FPGA Block RAM Resources User Guide*. Červenec 2011, [Online; navštíveno 14.05.2017].
URL https://www.xilinx.com/support/documentation/user_guides/ug383.pdf

[10] Xilinx: *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics*. Leden 2015, [Online; navštíveno 14.05.2017].
URL https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf

[11] Šurkala, M.: *Test paměťových karet SDHC a SDXC*. Květen 2016, [Online; navštíveno 14.05.2017].
URL http://www.digimanie.cz/test-pametovych-karet-sdhc-a-sdxc/5180-8

Přílohy

Příloha A

Obsah disku

Adresář	**Popis**
/dokumentace/dokumentace_k_cipum	Obsahuje dokumentace ze kterých jsem čerpal
/dokumentace/BP_Zdrojove_soubory	Obsahuje zdrojové soubory textu této práce.
/kompletni_projekty	Zde je celý projekt s demonstrační aplikaci.
/screeny_generovani_bram	Zde je postup generování portu pro přístup k BRAM.
/testovaci_soubory	Sobory použité při testování demonstrační aplikace.
/zdrojove_soubory	Obsahuje zdrojové soubory knihovny.

Tabulka A.1: Obsah disku