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/20393/2016/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 pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Bez požadavků.

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: 23. června 2017
Datum odevzdání: 31. července 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
31. července 2017

Poděkování

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

Obsah

Kapitola 1

Úvod

Doby gramofonových desek a magnetofonových pásek už jsou dávno pryč. Dnešní technický pokrok si vyžádal, aby se z analogových systémů přešlo na digitální. Snad každý člověk má doma alespoň jedno zařízení, které má digitální vstup a výstup pro zvuk. A jelikož je člověk náročný tvor vyžaduje zvuk ve vysoké kvalitě. Kvalitní zvuk by měl být bez šumu, bez zákmitů, vysoké tony by neměly být ořezané a nízké basové tony zkreslené. To ovšem oproti analogu není tak snadné dosáhnout jelikož digitální obvody vytvářejí mnoho rušivých vlivu, které mají na kvalitu zvuku negativní dopad. Proto se rozhraní pro zvuk řeší pomocí odděleného obvodu, aby byl co nejdále od rušivých elementů digitálních obvodů.

V této prací se setkáme s audiokodekem SGTL5000 od firmy NXP. Cílem této práce je vytvořit knihovnu pro obsluhu komunikace mezi MCU a zvukovým kodekem. Provést implementaci knihovny, vytvořit demonstrační aplikaci a otestovat dosažené výsledky vzhledem ke kvalitě zvuku.

Kapitola 2

Cílová platforma

Cílová platforma je výuková deska Minerva s neoficiálním názvem Fitkit3. Knihovna má umožňovat obsluhu audiokodeku programem v mikrokontroléru. Podle obvodového zapojení není audiokodek připojen k mikrokontroléru, ale k FPGA. Mezi mikrokontrolérem a FPGA je natažena 32bitová sběrnice FlexBus. Proto je výhodné využít již vytvořených obslužných knihoven a modulů, které jsou součástí dřívějších prací.

2.1 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. Obrázek 2.1 ukazuje část možností nastavení vývodů. Za povšimnutí stojí volba I2C, která má jinak definovánu dolní a horní úroveň proti klasickému LVCMOS33. Možnosti vývodů lze nalézt v dokumentaci DS162 [15].

FPGA obsahuje RAM paměť uspořádanou po blocích 16 kb. Čip XC6SLX9 obsahuje 32 bloků RAM. Vstupně/výstupní uživatelský 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 [14].

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. Počet a velikost uživatelských portů je omezen, jak ukazuje obrázek 2.2. Port lze vygenerovat pomocí Xilinx CORE Generator.

Jak již bylo zmíněno, FPGA se chová podle konfiguračního souboru. Aby bylo možné sestavit konfigurační soubor, je třeba vědět jaké zapojení má být v obvodu realizováno. Pro popis obvodu lze využít jazyk VHDL. VHDL je jazyk pro popis hardwaru. 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 registrů na vstupy registrů. Tím dosáhneme, že v registrech budou uloženy hodnoty, i když je nezadefinujeme v daném stavu. Také je třeba nezapomenout, že hodnota na výstupu registru se objeví až následujícím stavu [13].

2.2 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 [4]. Programování probíhá pomocí rozhraní OSBDM a aplikace Kinetis Design Studio (KDS). Rozhraní OSBDM je integrováno na kitu Minerva. 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. Čehož můžeme využít při debugování stavu, jelikož kontrola pomocí led diod je velice nepraktická. 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 má ale nevýhodu v tom, že nemůžeme zasahovat do periferií, které jsou spravovány PE komponentou. Mikrokontrolér obsahuje také dva moduly s I2C sběrnicí a jeden modul s I2S sběrnicí. Tyto porty však nejsou v této práci využity.

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.

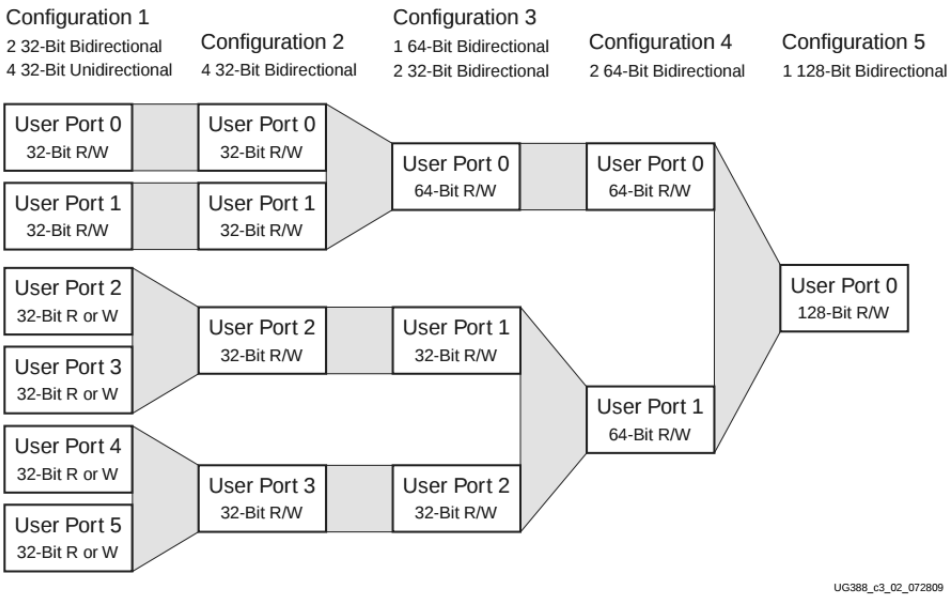


Figure 2-2: **Possible Port Configurations for the User Interface**

Obrázek 2.2: Část tabulky ukazující možné nastavení portu řadiče RAM paměti.

2.3 Audiokodek

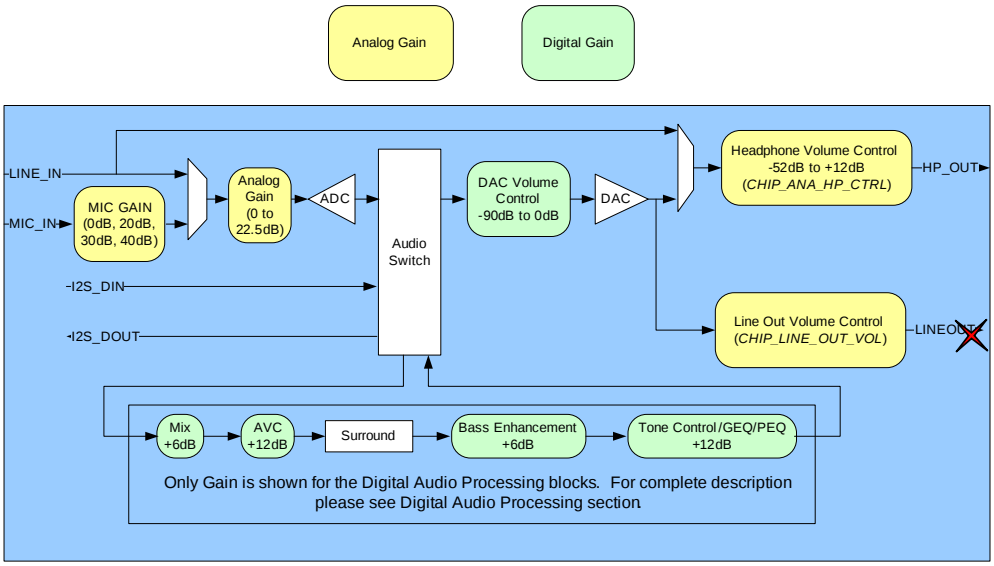
Na kitu je osazen audiokodek SGTL5000 od firmy NXP (bývalý Freescale Semiconductor). Jak již bylo zmíněno v úvodu, Audiokodek se využívá pro zpracování analogového zvuku na digitální a opačně. Aby nedocházelo k rušení analogového výstupu, má kodek oddělené napájecí větve pro digitální a analogové obvody. V rámci jednoduchosti jsou na Fitkitu obě větve napájeny stejným zdrojem a analogová větev je odrušena pouze cívkou. Kodek má vyveden stereo výstup na sluchátka do konektoru typu jack s maximální zátěží 16 Ω. Kodek 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 a zajišťuje předpětí pro mikrofón.

Digitální rozhraní je tvořeno sběrnicemi I2C a I2S. Přes I2C jsou zpřístupněny registry audiokodeku. Řízení probíhá zápisem patřičných hodnot do patřičných registrů. Registry mají velikost 16 bitů a jsou adresovány také 16ti bity. Seznam všech registrů a patřičných hodnot se nachází v dokumentaci [8].

I2S slouží pro přenos audio dat. Data jsou ve formátu lineární pulzní kódové modulace (LPCM). Velikosti vzorků, které kodek dokáže zpracovat jsou 16, 20, 24 a 32 bitů. Dokumentace udává maximální odstup signál/šum (SNR) 100 dB, v různých režimech pohybuje okolo 97 dB.

Zdroj hodinového signálu může poskytovat I2S nebo externě připojený krystal. Na Fitkitu je připojen 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.

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. Vnitřní propojení vstupů a výstupů ukazuje obrázek 2.3, podle něj je nutné nastavit zvukovou cestu. Veškeré informace o audiokodeku se nachází v dokumentaci [8].



Obrázek 2.3: Zjednodušený vnitřní blokový diagram audiokodeku.

2.4 Rozhraní

Audiokodek obsahuje pouze dvě digitální rozhraní a to I2C a I2S. I2C slouží pro zapisování a čtení vnitřních registrů, které nastavují a ovlivňují chování různých částí obvodu uvnitř audiokodeku. I2S slouží pouze pro přenos audio dat. Vzhledem k tomu, že audiokodek je připojen k FPGA, je nutné v něm rozhraní I2C a I2S implementovat.

2.4.1 I2C

I2C je nízkorychlostní sériová sběrnice vyvinutá firmou Philips (nyní NXP). Složí především k propojení integrovaných obvodů většinou v rámci jednoho zařízení. Skládá se z vodičů SDA (serial data) a SCL (serial clock). Zařízení na sběrnici jsou dvojího typu master a slave. Komunikace po vodičích probíhá obousměrně.

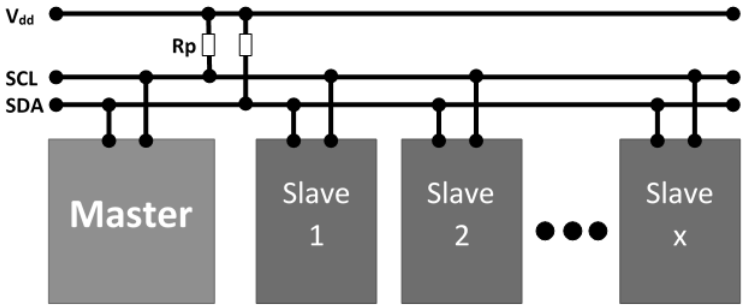
Z elektrického hlediska jsou výstupy zařízení na sběrnici implementovaná jako otevřený kolektor, což znamená, že se mohou nacházet buď ve stavu nízké úrovně L a ve stavu vysoké impedance Z. Oba vodiče musí být připojeny pomocí takzvaného pull-up rezistoru ke kladnému napájecímu napětí, tím je zajištěna vysoká úroveň H. Toto propojení znemožňuje vznik zkratu a tudíž nemůže dojít ke zničení zařízení, například chybou v implementaci protokolu. Zařízení na sběrnici jsou rozčleněna na řídící (master) a řízená (slave). Pouze řídící zařízení zahajuje a ukončuje přenos a generuje hodinový signál na vodiči SCL. Řízené zařízení je vybráno adresou. Komunikace může v jednom okamžiku probíhat pouze mezi jedním řídícím zařízením a jedním řízeným zařízením. Příklad zapojení se nachází na obrázku 2.4. Podle oficiální specifikace [10] je možné na sběrnici připojit více řídících zařízení, ale poté je nutné implementovat řešení kolizí.

Přenos probíhá kombinováním následujících celků:

- Stav klidu — Je zajištěn vysokou úrovní H na obou vodičích. Řídící zařízení negeneruje hodinový signál a neprobíhá žádný přenos.
- Start podmínka — Zahajuje přenos nebo jeho další část. Je vygenerována tak, že se změní úroveň SDA z H na L zatímco SCL je v úrovni H.
- Stop podmínka — Ukončuje přenos. Je vygenerována podobně jako start podmínka. Úroveň SDA se změní z L na H zatímco SCL je je v úrovni H.
- Přenos dat — Data jsou přenášena po osmi bitech od nejvíc významového bitu (MSB) po nejméně významový bit (LSB). Logická úroveň na SDA se smí měnit pouze je-li SCL v úrovni L. Při každém pulzu na SCL je přenesen jeden bit.
- Potvrzující bit ACK (acknowledge) — Tento devátý bit slouží hlavně k potvrzení příjmu dat — správnost dat zaručena není. Neodesílá jej řídící zařízení, ale vybrané řízené zařízení. Pokud je vše v pořádku, zařízení odešle ACK držením úrovně L. Pokud dojde k chybě, zařízení odešle NACK uvolněním sběrnice, pull-up rezistory zajistí úroveň H. NACK má v různých situacích více významů, například adresované zařízení nemusí existovat. Další možnosti jsou popsány v dokumentaci [10] v kapitole 3.1.6.

Přenos probíhá následovně. Nejprve je odeslána start podmínka a poté osm bitů dat, z nichž prvních sedm bitů je adresa řízeného zařízení. Osmý bit určuje zda bude probíhat zápis (0) nebo čtení (1). Existuje i varianta s deseti adresovými bity. Přenos a význam následujících dat specifikuje řízené zařízení. Ve standardní verzi je frekvence SCL stanovena

na 100 kHz. Dnes se běžně používají rychlejší režimy přenosu například 400 kHz, který využívá i audiokodek. Praktičtější informace o sběrnici lze nalézt [11].



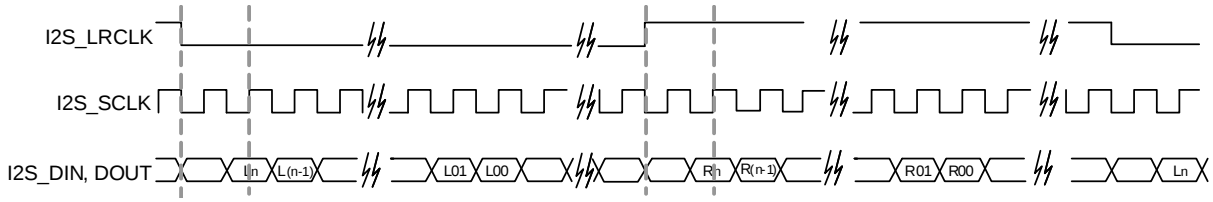
Obrázek 2.4: Propojení zařízení přes sběrnici I2C.

2.4.2 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 BCLK (bit clock), LRCLK (left-right clock) a SD (serial data). Vyskytují se i jiná označení 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 BCLK probíhá přenos hodinového signálu. Vodičem SD probíhá přenos dat rychlostí určené vodičem BCLK. Hladina signálu LRCLK určuje, zda se jedná o levý nebo pravý kanál. Jeho frekvence se rovná vzorkovací frekvenci přehrávání. Frekvence BCLK je typicky násobkem vzorkovací frekvence například $64 \cdot Fs$. Násobek musí být zvolen tak aby bylo možné přenést požadovaný počet bitů vzorku signálu jednoho kanálu za půlperiodu LRCLK. Přenos probíhá následovně: Signál LRCLK se signálem BCLK 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 signálu LRCLK. Ukázka je na obrázku 2.5. Existují i upravené formáty left-justified a right-justified. Jak již název napovídá, data jsou zarovnána k hraně LRCLK doleva nebo doprava.

Audio data jsou přenášena ve formátu lineární pulzně kódové modulace (LPCM) se znaménkem. Znaménko je zakódováno pomocí dvojkového doplňku.



Obrázek 2.5: Diagram přenosu dat přes sběrnici I2S.

Kapitola 3

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 sběrnici I2C. Z obvodového schématu je patrné, že audiokodek není připojen přímo k MCU, ale je připojen k FPGA.

První variantou je použít natažené vodiče mezi MCU a FPGA jako I2C a I2S přepnutím příslušných multiplexorů v MCU. Poté do FPGA navrhnout konfiguraci, která umožňovala 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 a proto je výhodná z hlediska dalšího vývoje platformy. Komunikace přes FlexBus již byla zprovozněna, takže by mělo stačit použít již hotový modul [1].

3.1 FlexBus

FlexBus je sběrnice pro připojení externích zařízení k mikrokontroléru Kinetis jako jsou paměťové moduly, programovatelné logické obvody a tak dále. FlexBus rozhraní je s FPGA spojeno 41 vodiči, z nichž 32 je určeno pro přenos adresy a dat. Z předchozích prací [1] [5] 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 96 MHz 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 16ti bitová 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 16ti bitová a tu jsem se rozhodl rozšířit na 32 bitů a to kvůli rychlejšímu přenosu.

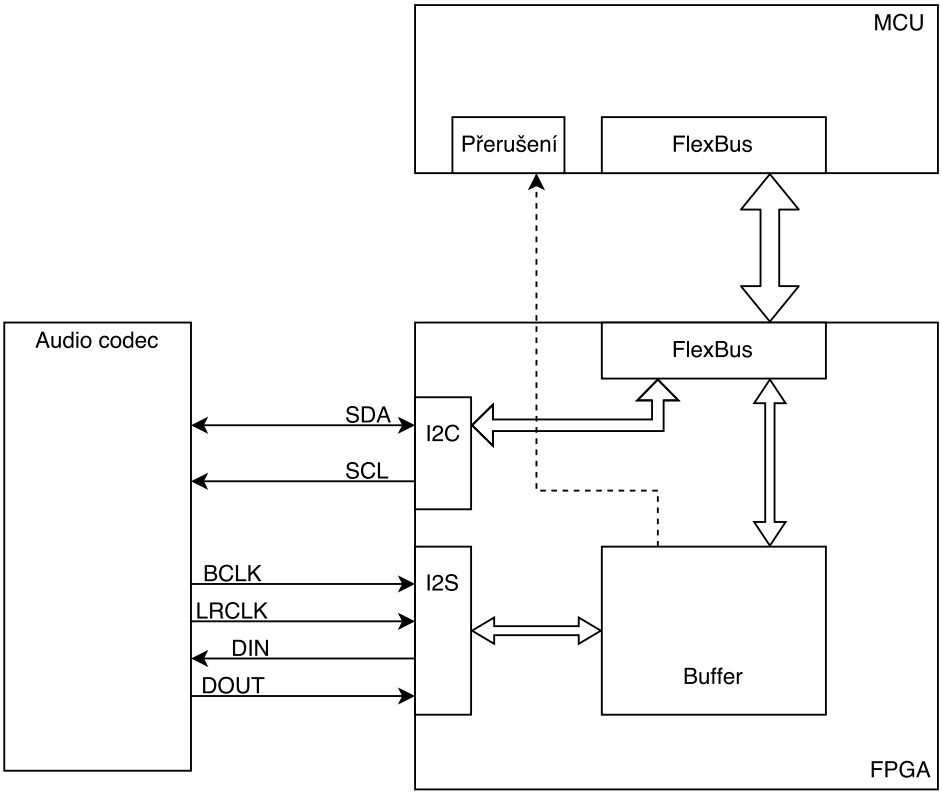
Přenos probíhá po transakcích. Transakci zahajuje vždy mikrokontrolér a ten je blokován do doby, než adresovaná komponenta potvrdí transakci.

3.2 Řešení

Rozhodl jsem se vytvořit dvě komponenty. První část bude číst a zapisovat registry v audiokodeku přes I2C, řídící část. Druhá část 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. Poté nastavit I2S rozhraní ať se chová jako řídící. 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 vodičem 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. Návrh řešení ukazuje obrázek 3.1.



Obrázek 3.1: Návrh komunikace mezi mikrokontrolérem a audiokodekem.

Kapitola 4

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.

4.1 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.

4.2 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 čipy a ani součástky, které I2C sběrnice potřebuje ke své funkci. V FPGA jsou proto zapnuty pull-up rezistory, které nahrazují vnější rezistory.

O generování I2C signálu se stará modul `i2c_master`, který je dostupný na internetu [6]. 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 doporučený v dokumentaci audiokodeku. Modul tedy generuje signál pro vodič SDA a SCL a stará se o správné načasování. Modul má signály pro sedmibitovou adresu I2C zařízení a dva osmibitové datové signály pro vstup a výstup dat. Ovládání probíhá pomocí signálů `ena`, `rw` a `busy`. Signál `rw` určuje zda bude probíhat zápis (0) nebo čtení (1). Signál `ena` slouží ke spouštění přenosu a signál `busy` indikuje činnost modulu. Přenos začíná nastavením signálu `ena` do logické 1. Jakmile signál `busy` přejde ze stavu 0 do 1 dojde k uložení adresy

a dat. V teto chvíli modul zahájil komunikaci a odesílá adresu zařízení a pak následuje přenos jednoho bajtu směrem, který určil signál `rw`. Jakmile přejde signál `busy` z 1 do 0, je přenos dokončen. Pokud je signál `ena` držen v 1 i ve chvíli, kdy signál `busy` přechází z 1 do 0, pro modul to znamená, že má přenášet další bajt. Data jsou načtena jakmile signál `busy` přechází znovu z 0 na 1. Pokud se změní adresa cílového zařízení nebo tok dat signálem `rw`, tak modul musí přerušit přenos a odeslat novou adresu a bit `rw`. K přerušení nedojde odesláním stop a start podmínky, ale takzvaným opakovaným startem (repeated start). Což je prakticky odeslání pouze start podmínky. Podrobnější vysvětlení opakovaného startu se nachází v kapitole 3.1.4 v dokumentaci [10]. Podrobné informace o modulu jsou na stránkách, kde je k dispozici [6].

Nad I2C modulem je vytvořena komponenta `flex_audio_control`, která připojena paralelně k FlexBus komponentě. `flex_audio_control` slouží k předávání transakcí do I2C modulu. Registry audiokodeku jsou adresovány 16ti bity a mají velikost 16 bitů. Zápis do registru se přes I2C provádí odesláním po sobě jdoucích čtyř bajtů, z nichž první dva bajty obsahují adresu registrů a další dva obsahují hodnotu co se má uložit. Čtení registru se přes I2C provádí odesláním po sobě jdoucích dvou bajtů, které obsahují adresu registru. Následuje přerušení přenosu (repeat start) a následné přečtení dvou bajtů, což je hodnota adresovaného registru. Diagram přenosu ukazuje tabulka 11 a 13 na straně 24 v dokumentaci [8].

Přenos přes FlexBus probíhá po transakcích o velikosti 32 bitů, což umožňuje při zápisu registru přenést adresu i data v jedné transakci. Zmíněných 32 je tedy odesíláno od I2C modulu po bajtech (osmi bitech) od MSB (nejvíce významový bit) po LSB (nejméně významový bit). Z toho vyplývá, že horních 16 bitů obsahuje adresu registru a dolních 16 bitu obsahuje zapisovaná data do registru. Transakce na rozhraní FlexBus je ukončena až po odeslání všech 32 bitů.

Čtení registru audiokodeku je složitější. Nejprve je nutné přenést adresu registru do audiokodeku a následně přenést hodnotu do mikrokontroléru. Tudíž po rozhraní FlexBus musí proběhnout dvě transakce. Transakce jsou od sebe rozlišeny adresou pro adresování uvnitř komponenty, popsanou výše 3.1. První transakce přenáší v dolních 16ti bitech adresu, která se uloží do pomocného registru `temp_reg` v komponentě `flex_audio_control`. Tímto je první transakce na rozhraní FlexBus ukončena. Druhá transakce je v režimu čtení. Po jejím příchodu je spuštěn I2C modul, který nejprve odešle adresu z pomocného registru `temp_reg` a poté příjme hodnotu registru audiokodeku. Hodnota je předána na rozhraní FlexBus v dolních 16ti bitech. Druhá transakce je tímto ukončena. Transakce jsou na sobě prakticky nezávislé.

V knihovně na straně mikrokontroléru jsou funkce `void audio_reg_write(uint16_t address, uint16_t data)`, které zapíše data do adresovaného registru a funkce `uint16_t audio_reg_read(uint16_t address)`, která vrátí obsah adresovaného registru. Funkce pracují s odpovídajícími datovými typy a generují transakce na rozhraní FlexBus ve výše uvedeném formátu a pořadí. 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. Poté již nasledují funkce a makra, které provádí obsluhu audiokodeku zapisováním hodnot pomocí výše zmíněných funkcí.

4.3 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, ale na druhou stranu omezení velikosti paměti na mocniny čísla 2.

4.3.1 I2S část

Audiokodek přijímá a odesílá data prostřednictvím I2S sběrnice. Po provedení inicializační funkce jsou podle vybrané vzorkovací frekvence nastaveny patřičné hodinové signály a audiokodek je nastaven jako řídící (master) na sběrnici I2S. Poskytuje tedy hodinový signál, na který reaguje FPGA. Audiokodek provádí vyhodnocování signálu na sběrnici I2S s každou náběžnou hranou `I2S_BCLK`. Konečný automat je řízen hodinovým signálem `I2S_BCLK`, reaguje však na sestupnou hranu. Tím je dosaženo, že změny na I2S vodičích provádí obě strany v jiný časová interval, tudíž by nemělo dojít chybě přenosu vlivem špatného časování.

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`. Frekvence signálu `I2S_LRCLK` je zároveň vzorkovací frekvencí audiosignálu. Formát je nastaven na klasický I2S formát, to znamená, že první datový bit následuje až po prvním hodinovém taktu, tak jako je na obrázku 2.5. Z toho můžeme vypočítat volné takty $(64 \div 2) - 1$, tedy po každém odeslání vzorku zbývá 15 hodinových taktů. To postačuje pro změnu adresy a přístup do BRAM, která má zpoždění jeden hodinový takt.

Pro přístup 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`. Logická 0 značí levý kanál a logická 1 značí pravý kanál. 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 audiosignálu.

Audiokodek obsahuje dva datové vodiče, jeden pro přenos audio dat do kodeku `I2S_DIN` a druhý pro přenos audio dat z kodeku `I2S_DOUT`. Rozdíl spočívá pouze ve směru toku, formát je shodný. Přenos tedy probíhá zároveň a to následovně. Nejdříve je přečteno paměťové místo jehož pozici udává čítač v kombinaci se signálem udávajícím kanál (`I2S_LRCLK`). Poté se čeká na změnu `I2S_LRCLK`. Po změně je odesílán vzorek načtený z paměti a zároveň přijímán vzorek z audiokodeku do pomocného registru `i2s_dout_data_reg`. Po výměně všech bitů je přijatý vzorek v pomocném registru zapsán do paměti, 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`.Z popsaného průběhu je patrné, že změna adresy na portu B proběhne dříve, než se změní signál `I2S_LRCLK`, proto je spodní bit adresy řízen nepřímo.

Funkčnost tohoto automatu lze ověřit pomocí audiokodeku. Nejdříve je potřeba ověřit, že je audiokodek správně nastaven a to pomocí puštění vstupu na výstup. Trasa musí vést přes digitální převodníky. Trasy vyobrazuje blokový diagram 2.3. Při použití mikrofonu a sluchátek musí být váš hlas slyšet. Pokud funguje, tak přepneme I2S výstup na mikrofón

a výstupní převodník na sluchátka přepneme na I2S vstup. Tímto nastavením se vzorky ukládají do paměti a následně přehrávají zpět. Svůj hlas můžete slyšet zpožděně, což vytvoří ozvěnu. Síla efektu záleží na zvolené vzorkovací frekvenci a velikosti paměti.

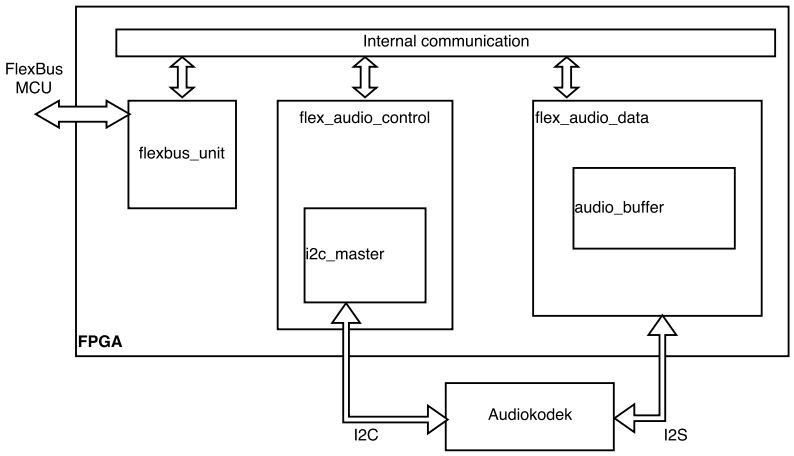
Jak již bylo zmíněno nejvýznamnější bit adresy (MSB) určuje přesnou polovinu paměti. Ten je hlídán a při jeho změně je vygenerováno přerušení do mikrokontroléru. Dále je použit k řízení přístupu na portu A, a to mě přivádí k popisu druhého konečného automatu.

4.3.2 FlexBus část

Druhý konečný automat slouží k přesunu dat mezi mikrokontrolérem a vyrovnávací pamětí. Komunikuje s BRAM pamětí pomocí portu A. Port A je nastaven na šířku 32 bitů, stejně jako velikost transakcí na rozhraní FlexBus. Konečný automat obsluhuje transakce FlexBus komponenty. Od ní je také využit hodinový signál pro běh tohoto automatu. Čtecí transakce přečte data z paměti a zapisovací transakce zapíše data do paměti. Adresa paměti je tvořena čítačem, jehož signál je o jeden bit menší než adresový signál portu A. Čítač se inkrementuje automaticky po každé transakci. Nejvyšší významový bit signálu adresy je řízen negací nejvyššího významového bitu adresy portu B. Tento mechanizmus znemožňuje přístup do opačné poloviny paměti, se kterou pracuje výše zmíněný konečný automat.

Transakce spouští mikrokontrolér poté, co obdrží přerušení od prvního automatu. Jelikož čítač adresy paměti je zvyšován každou transakcí, musí mikrokontrolér provést přesný počet transakcí, aby přečetl celou polovinu paměti a tím nedošlo ke ztrátě dat. V hlavičkovém souboru knihovny se nachází makro `HALF_BUF_SIZ`, které určuje počet 32 bitových transakcí. Výpočet lze provést z celkové velikosti paměti v bitech $(velikost\_pameti \div 2) \div 32$. Mikrokontrolér musí stihnout provést změny dříve, než dojde k překlopení do další poloviny paměti. Při přehrávání stačí data do paměti zapsat. Při nahrávání i přehrávání je nutné nahrávána data nejprve přečíst a poté zapsat přehrávaná data.

Oba automaty pracují prakticky nezávisle na sobě. Uspořádání komponent ukazuje obrázek 4.1.



Obrázek 4.1: Diagram komponent v FPGA.

4.4 Spuštění audiokodeku

Po zapnutí napájení jsou hodnoty v registrech audiokodeku ve výchozích hodnotách, tudíž je potřeba zapnout a nastavit. Ještě než můžeme přistupovat k audiokodeku je třeba mít zprovozněné FlexBus rozhraní a FPGA musí obsahovat patřičnou konfiguraci. Ovládání FlexBus rozhraní je převzato z předchozích prací [5] [1]. Je však upraveno pro přenos všech 32 bitů. Předchozí verze umožňovala přenášet pouze 16 bitů. K nastavení FlexBus rozhraní složí funkce `flex_init`. Funkce nastaví příslušné porty spustí hodinový signál a vybere adresový prostor. Po této funkci není rozhraní ještě plně funkční, nemá nastaveno blokovací režim. To znamená, že mikrokontrolér nečeká na potvrzení transakce vodičem `FXB_TA`, ale potvrdí transakci sám po 63 čekacích stavech, což je maximální hodnota. Je potřeba ověřit zda je v FPGA správná konfigurace a to pomocí funkce `flexbus_recheck`. Ta je ponechána beze změny. Funkce přečte první tři bajty z konfigurační flash paměti pro FPGA. Pokud proběhne v pořádku, nastaví FlexBus rozhraní do blokovacího režimu. Dále je potřeba ověřit spojení s audiokodekem, k tomu slouží funkce `audio_check_connection`. Odpověď se nachází v proměnné `audio_connectionOK`. Funkce přečte identifikátor čipu, podle dokumentace by měl být pro všechny audiokodeky stejný.

Nyní je audiokodek dostupný a můžeme jej zapnout. K tomu slouží inicializační funkce `audio_init`. Hned na začátku funkce se nachází zapnutí detekce průchodu nulou. Tento detektor je využit při změně hlasitosti. Se zapnutým detektorem nedochází ke změně okamžitě, ale až v okamžiku, kdy signál prochází nulou. Toto chování omezuje prasknutí či lupnutí ve zvuku. Poté následuje aktivace funkce mute, ta zajistí ztlumení výstupu. Dále vypnutí modulu VAG (Voltage Analog Ground). Vypnutí VAG neproběhne okamžitě, proto hned za tímto příkazem se nachází aktivní čekání. Po čekání proběhne vypnutí sluchátkového zesilovače. Tato sekvence je zde pro případ, že audiokodek je již nastaven a zapnut. Inicializační funkce se typický spouští na začátku programu, bez ohledu na to, zda audiokodek již je nastaven a zapnut, či nikoli. Ztlumení a vypnutí sluchátkového zesilovače zabraňuje praskání a lupání ve výstupu během změn v audiokodeku. Inicializační funkci je tedy možné spustit vícekrát. S výjimkou zapnutí detektoru průchodu nulou nemá sekvence při resetovaném audiokodeku na nic vliv.

Nyní již následuje start samotného audiokodeku. Hned po výše zmíněném úvodu se nachází nastavení VAG (Voltage Analog Ground). Jak již z názvu vyplývá, jedná se o napětí na analogové zemi. V obvodovém schématu je označená jako `AGND`. Analogová zem je oddělená od normální země (`GND`) a je vyvedena na analogové vstupy a výstupy. To znamená přímo na vnější krajní kontakty jack konektorů. Analogová zem je zde proto, aby bylo možné na výstupech vytvářet jak kladné tak záporné půlvlny analogového signálu. U zvukového signálu je stejnosměrná složka signálu nežádoucí, z toho plyne, že pro zajištění co největšího výkonu je nutné mít co největší jak kladnou tak zápornou půlvlnu. Proto je nejvhodnější nastavit analogovou zem na polovinu napájecího napětí. V dokumentaci k audiokodeku [8] je vzoreček pro výpočet $VDDA \div 2$. Avšak v tabulce číslo 32 jsou uvedeny hodnoty VAG pro některé kombinace napájecího napětí. Pro napětí na Fitkitu 3,3 V na `VDDA` i `VDDIO`, je uvedena hodnota VAG 1,55 V. Tato hodnota však není polovina napájecího napětí. Maximální napětí co lze na VAG zvolit je 1,575 V, což už není moc rozdíl oproti doporučeným 1,55 V. Zvolil jsem proto doporučené napětí 1,55 V. Napětí je použito také jako reference pro převodníky ADC a DAC. Jak jsem již zmínil, změna VAG neproběhne okamžitě. V normálním režimu trvá změna 200 ms, lze nastavit i pomalejší režim 400 ms. V knihovně jsem využil výchozí normální režim, lupnutí ve sluchátku není příliš výrazné. Aby bylo využito této prodlevy musí již být ostatní obvody zapnuté, než dojde k zapnutí VAG,

a naopak musí být vypínány až poté, co je VAG vypnut a uběhla zmíněná doba. Proto je vytvořena čekací funkce `audio_vag_delay`, která pokrývá výše zmíněnou dobu.

Následuje nastavení nábojové pumpy (charge pump). Nábojová pumpa je zde pro potřeby analogových obvodů. Pokud je napájecí napětí menší než 3,1 V je potřeba ji zapnout. Na Fitkitu je napájecí napětí 3,3 V. Z toho důvodu není nábojová pumpa využita, ve výchozím stavu je vypnuta. Dokumentace doporučuje přepnout zdroj nábojové pumpy na `VDDIO`, pokud obě napětí `VDDA` i `VDDIO` je větší než 3,1 V, což je provedeno, i když se pumpa nevyužívá.

Z napájecí části ještě zbývá napětí na `VDDD`. Pin `VDDD` na Fitkitu není připojen, napětí zajišťuje sám audiokodek a to hned několika způsoby. Automaticky po resetu je aktivován startovaní obvod (`STARTUP_POWERUP`) a jednoduchý napájecí zdroj (`LINREG_SIMPLE_POWERUP`). Tyto obvody zajistí napájení ihned po zapnutí nebo resetu audiokodeku dostatečné napětí na `VDDD`. K tomu je k dispozici hlavní lineární regulátor (`LINREG_D_POWERUP`), který lze nastavit v rozmezí 0,85 až 1,6 V. Povolené napětí na `VDDD` je v rozmezí 1,1 až 2 V. Na regulátoru jsem tedy zvolil nejvyšší napětí 1,6 V. Po zapnutí hlavního regulátoru jsou vypnuty oba výše zmíněné zdroje. Dokumentace k audiokodeku [8] doporučuje napájet `VDDD` externím zdrojem a všechny tři zmíněné zdroje vypnout. Je to z důvodu výskytu chyby ve startovacím obvodu, který po zapnutí nezajistí napětí na `VDDD`, což způsobí, že s audiokodekem není možné komunikovat přes I2C ani přes SPI. Chyba se na mnou vypůjčeném Fitkitu neprojevila. Podle dokumentu [9] je příčina pozvolném náběhu napájecího napětí.

Na řadu přichází nastavení hodinového signálu a vzorkovací frekvence. Pro nastavení vzorkovací frekvence je vytvořena funkce `audio_change_fs`. Funkce má jako parametr výčtový datový typ, ve kterém je zvolena vzorkovací frekvence. Funkci je možné volat za běhu audiokodeku a měnit parametry, třeba podle přehrávaného obsahu. Pomocí `SYS_FS` je vybrána základní frekvence a v kombinaci `RATE_MODE` je zvolena výsledná vzorkovací frekvence. Přehled frekvencí a všech možných kombinací je shrnut v tabulce 4.1.

RATE_MODE	**SYS_FS**			
1/1	96000	48000	44100	32000
1/2	48000	24000	22500	16000
1/4	24000	12000	11025	8000
1/6	16000	8000	*7350*	*5333,$\bar{3}$*

Tabulka 4.1: Přehled vzorkovacích frekvencí.

Nelze si nevšimnout, že v tabulce jsou některé frekvence vícekrát. Proto jsem vybral jen některé kombinace, které jsou označeny podtržením. Vybrané hodnoty jsou v dokumentaci označeny jako podporované. Kurzívou označené vzorkovací frekvence dokumentace audiokodeku nezmiňuje.

K nastavení vzorkovací frekvence ještě částečně patří nastavení fázového závěsu a to pomocí dvou hodnot. Hodnoty závisí na připojeném krystalu a zvolené základní frekvenci `SYS_FS`. Rozhoduje se, zda je to 44,1 kHz nebo ne. Hodnoty jsem tedy vypočítal pomocí vzorečků z dokumentace, které lze nalézt v tabulce číslo 34 v dokumentaci audiokodeku [8]. Výpočet a výsledné hodnoty jsou v komentáři ve zdrojovém kódu. Tímto opouštíme `audio_change_fs`.

Dále je potřeba zapnou děličku vstupního kmitočtu, protože na Fitkitu je vstupní kmitočet 24,576 MHz, což je větší než 17 MHz. Poté zesilovač napěťově řízeného oscilátoru (VCOAMP), který je součástí fázového závěsu. Následuje zapnutí samotného fázového zá-

věsu. Nyní by mělo stačit přepnout na fázový závěs, ale audiokodek nebude fungovat. Předtím je potřeba ještě nastavit I2S sběrnici do řídícího režimu (master). Pak už následuje jen nastavení I2S na 16 bitů, zapnutí převodníků, vypnutí funkce mute, nastavení hlasitosti. Na konec zapnutí sluchátkového zesilovače a VAG. Po čekání na náběh VAG je vypnuta mute funkce a nastavena hlasitost na sluchátkovém výstupu. Tímto inicializační funkce končí.

4.5 Použití knihovny

Knihovna je uložena ve složce `libaudiocodec`. Pro komunikaci s FPGA používá část knihovny z předchozích dvou prací [1] a [5], která je umístěná ve složce `fpga`. Funkce jsou pojmenovány malými písmeny, makra velkými písmeny a mají společnou předponu `audio_`.

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 obzvláště důležitá, 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 funkcí `audio_change_fs`.

Inicializační funkce, popsaná v sekci 4.4, obsahuje poměrně dlouhou sekvenci příkazů a dvě aktivní čekání. Každé po dobu zhruba 200ms. Je to z důvodu omezení prasknutí při zapínání sluchátkového zesilovače. Inicializační funkce zapne vstup i výstup, odblokuje tlumení (mute) a nastaví výchozí hlasitost, avšak nezasahuje do nastavení trasy zvuku. Trasu zvuku je potřeba nastavit a to 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 rutinu `audioISR`. Pro přenos přerušení jsem využil nevyužitý vodič `FLEXBUS_OE`. Je připojen na port B pin číslo 19. K registraci obslužné rutiny jsem využil režimu Processor Expert. Spouštění rutiny je potřeba nastavit na náběžnou hranu. Rutina obstará příjem a odesílání vzorků do vyrovnávací paměti v FPGA. Knihovna obsahuje funkce pro přehrávání `audio_play`, nahrávání `audio_record`, případně obojího zároveň `audio_play_record`. Tyto funkce obsahují povinný parametr a to je adresa uživatelem vytvořené funkce, která předá audio data do připravené paměti. Adresa připravené paměti bude předána do parametru uživatelem vytvořené funkce. Velikost paměti se nachází v makru `HALF_BUF_SIZ`.

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.

4.6 Demonstrační aplikace

Pro potvrzení funkčnosti a otestování zda audiokodek funguje byla vytvořena demonstrační aplikace. Jako úložiště audio dat využívá SD kartu. K přístupu na SD kartu je využita knihovna pro ovládání čtečky SD karet, která je součástí dřívější práce [7]. Ta také využívá knihovnu FatFS [2] pro přístup k souborovému systému FAT na SD kartě. Využil jsem však novější verzi R0.12b z listopadu 2016. SD kartě tedy musí být souborový systém FAT.

Formát pro ukládání dat jsem zvolil WAV. Je to nekomprimovaný formát. Na začátku obsahuje pevně danou hlavičku a dále následují pouze data. Na internetu jsem našel strukturu formátu WAV [12]. Většina údajů je nastavena fixně. Formát WAV umožňuje přenášet audio ve více vnitřních formátech. Aplikace však zpracovává pouze vzorky ve formátu LPCM 16 bitů little-endian a dva kanály. V konverzních programech je typicky pojmenován `pcm_s16le`. Soubor lze převést pomocí například pomocí programu avconv (ffmpeg) příkazem `avconv -i vstup.mp3 -c:a pcm_s16le vystup.wav`. Aplikace z WAV hlavičky načte rychlost vzorkovací frekvence a tu poté nastaví do audiokodeku, z toho důvodu jsou ve výčtovém datovém typu položkám přiřazeny čísla.

Po zapnutí se začne přehrávat soubor jehož název je napevno uložen v aplikaci. Tlačítky SW5 a SW3 lze regulovat výstupní hlasitost. Led diody jsou využity pro kontrolu rychlosti obsluhy přerušení. Aplikace ve výchozím nastavení používá vyrovnávací paměť o velikosti 32 kB.

4.7 Testování a výsledky

Testování probíhalo za pomocí sluchátek, mikrofonu a obyčejné SD karty ADATA 8 GB SDHC Class 4, která patří mezi průměr [16]. Chyby byly odhalovány poslechem a pomocí led diod, které zobrazují rychlost přerušení a dobu trvání přerušení. Poté i vizuální kontrolou v Audacity.

Přehrávání souboru jsem testoval 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 jsem testoval 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, což obsadí polovinu dostupné BRAM paměti. Velikost bloků zapisovaných na kartu je nyní 16 kB. Záznam již byl použitelný. Docházelo však k výpadkům po zhruba 20 sekundách záznamu. Po prozkoumání jsem zjistil, že obslužnou rutinu brzdí zápisy na SD kartu. Dalším možným zlepšením je snížení režie souborového systému FAT. Použitá knihovna FatFs [2] umožňuje předalokovat souvislou část paměti a 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 až 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.

Během testů jsem také narazil na zkreslení zvuku při zesílení sluchátkového zesilovače nad úroveň 0 dB. Při pátrání na internetu jsem objevil článek [3], který zmíněný problém popisuje. Jedná se o limitující velikost napájecího napětí, která omezuje maximální výstupní výkon zesilovače.

Kapitola 5

Závěr

Vytvořil jsem knihovnu pro obsluhu komunikace mezi mikrokontrolérem a zvukovým kodekem SGTL5000. Knihovna je určena pro výukovou platformu Minerva. Po důkladném nastudování obvodového schématu jsem provedl návrh koncepce komunikace mezi mikrokontrolérem a zvukovým kodekem SGTL5000 s ohledem na současný vývoj výukové platformy Minerva. Návrh zahrnuje Vzhledem ke specifickému propojení zvukového kodeku a mikrokontroléru přes obvod FPGA, bylo potřeba prostudovat již vytvořené práce, jejichž části by bylo možno 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. V FPGA jsem vytvořil moduly pro přenos řídících a zvukových dat mezi audiokodekem a mikrokontrolérem. V mikrokontroléru jsem vytvořil knihovnu pro obsluhu audiokodeku a jeho počáteční nastavení. Rozhraní knihovny umožňuje její snadnou použitelnost.

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 v podobě zvětšení vyrovnávací paměti a předalokací volného místa na SD kartě se podařilo dosáhnout o hodně lepších výsledků. Ještě lepších výsledků bylo dosaženo rychlejší a novější SD kartou. I přesto, že byl problém výrazně minimalizován nedošlo k jeho úplnému odstranění. Problém způsobuje nestálá doba zápisu SD karty.

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. Výhodným řešením by bylo použití DDR2 paměti jako vyrovnávací paměť. Ještě větší velikost vyrovnávací paměti by mohla problém odstranit. 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.

Vytvořená knihovna je využitelná v dalších pracích, které budou potřebovat pracovat se zvukem. Knihovna výrazně 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] Chan, E.: *FatFs - Generic FAT File System Module*. Březen 2017, [Online; navštíveno 14.05.2017].
URL http://elm-chan.org/fsw/ff/00index_e.html

[3] Chip: *Teensy Audio Board Headphone Level*. Listopad 2016, [Online; navštíveno 14.05.2017].
URL http://openaudio.blogspot.cz/2016/11/teensy-audio-board-headphone-level.html

[4] 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

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

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

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

[8] 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

[9] NXP: *SGTL5000 Silicon Errata*. Červenec 2013, [Online; navštíveno 14.05.2017].
URL http://www.nxp.com/docs/en/errata/SGTL5000ER.pdf

[10] NXP: *I2C-bus specification and user manual*. Duben 2014, [Online; navštíveno 14.05.2017].
URL http://www.nxp.com/documents/user_manual/UM10204.pdf

[11] Olejár, M.: *Stručný popis sběrnice I2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877*. Březen 2011, [Online; navštíveno 14.05.2017].
URL http://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256

[12] Schneider, J.: *C Wav Header Struct*. Listopad 2016, [Online; navštíveno 14.05.2017].
URL
https://gist.github.com/Jon-Schneider/8b7c53d27a7a13346a643dac9c19d34f

[13] 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

[14] 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

[15] 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

[16] 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