ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

BAKALÁŘSKÁ PRÁCE

2018 **Jiří Anděra**

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky

Generování tónů a využití audio výstupu přípravku Nexys4 v jazyce VHDL

květen 2018

Bakalant: Jiří Anděra

Vedoucí práce: Ing. Pavel Lafata, Ph.D.

Čestné prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

Datum: 25. 5. 2018

.....

podpis bakalanta

Poděkování

Děkuji vedoucímu mé práce Ing. Pavlu Lafatovi, Ph.D. za zapůjčení potřebných součástí k realizaci této práce, za odborné konzultace a připomínky, které byly velice cenné. Dále děkuji rodině za podporu ve studiu i v životě.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Anděra** Jméno: **Jiří** Osobní číslo: **439545**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Síťové a informační technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Generování tónů a využití audio výstupu přípravku Nexys4 v jazyce VHDL

Název bakalářské práce anglicky:

Using Audio Output of Nexys4 Kit to Generate Tones

Pokyny pro vypracování:

Seznamte se s přípravkem Digilent Nexys4 a jeho obsluhou pomocí jazyka VHDL. K přípravku připojte jednoduchou maticovou numerickou klávesnici a vytvořte potřebné VHDL kódy pro detekci stisknuté klávesy. Vytvořte pomocí jazyka VHDL program, který bude generovat základní tóny pro připojený reproduktor k audio výstupu přípravku. Jednotlivé tóny budou voleny pomocí kláves a zvolený tón bude zobrazen na segmentovém displeji přípravku.

Seznam doporučené literatury:

[1] Pinker, J.; Poupa, M.: Číslicové systémy a jazyk VHDL. 1. vydání, 2006. BEN-Technická literatura, Praha. ISBN: 80-7300-198-5.
[2] Digilent Xilinx - Manuály a materiály k vývojovým kitům Nexys3/4.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Lafata, Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **04.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

Ing. Pavel Lafata, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání Podpis studenta

Anotace:

Bakalářská práce se zabývá obsluhou přípravku Digilent Nexys4, vhodným popisem jeho vnitřních i vnějších částí. Zahrnut je popis řešení vnitřních částí jako je segmentový displej, generování tónů, obsluha mikrofonu, SD karty a vnější část, což je obsluha připojené maticové klávesnice. Pro popis je použit jazyk VHDL. Řešení daných částí je rozděleno do dílčích procesů, které dohromady tvoří funkční celek celé práce.

Klíčová slova: VHDL, Nexys4, Mikrofon, SPI rozhraní, Maticová klávesnice

Summary:

The bachelor thesis deals with Digilent product Nexys4 and its in/out interfaces. The solution is included like description of seven-segment diplay, tone generation, work with MEMS microphone, matrix keyboard and Secure Digital card. For description is used VHDL language. Solution is divided into separate processes which together create the base of this thesis.

Index Terms: VHDL, Nexys4, Microphone, SPI interface, Matrix Keyboard

Obsah

1 Úvod 10

1.1 Cíle práce 10

2 Teoretická část 11

2.1 Programovatelné logické obvody 11

2.1.1 FPGA 11

2.2 Nexys4 11

2.3 VHDL 12

2.3.1 Charakteristika 12

2.3.2 Nejdůležitější části 13

2.3.3 Příklad kódu 14

2.4 Použité komponenty a nástroje kitu Nexys4 při realizaci této práce 15

2.4.1 Mikrofon 15

2.4.2 Maticová klávesnice 15

2.4.3 Audio výstup 16

2.4.4 PDM 17

2.4.5 PWM 18

2.4.6 SD karta 18

2.4.7 Segmentový displej 22

3 Praktická část – vlastní realizace 24

3.1 Proces maticové klávesnice 24

3.2 Proces pro mikrofon 25

3.3 Proces pro SD kartu 26

3.3.1 Inicializace 26

3.3.2 Čtení 27
3.3.3 Zápis 27
3.4 Proces generování tónů 28
3.5 Proces segmentový displej 30
3.6 Pohled na projekt jako celek procesů 31
3.7 Testování a praktické výsledky 31
4 Závěr 34
5 Seznam použitých zdrojů 35
6 Seznam obrázků 36
7 Seznam použitých zkratek 37
8 Přílohy 38

1 Úvod

Práce se zabývá návrhem vhodného popisu logických obvodů pro generování základních tónů v jazyce VHDL (*VHSIC Hardware Description Language*). Generování tónů dnes můžeme nalézt v široké škále odvětví, ať od poplašných zařízení po hudební nástroje, či armádní a lékařské využití. Ke generování tónů je využit přípravek Digilent, kit Nexys4, obsahující FPGA Artix-7 od společnosti XILINX. Generované tóny jsou spouštěny pomocí jednoduché maticové klávesnice. U této mechanické maticové klávesnice je nutné vhodně ošetřit zápis a čtení. Na kitu je obsažen i audio výstup, který jsem využil pro přehrání daného tónu. Audio výstup má na vstupu zahrnut filtr, je tedy nutné použít vhodnou modulaci, v tomto případě PDM (*Pulse Density Modultion*) případně PWM (*Pulse Weight Modu*lation), které se běžně používají při zpracování zvuku. Jako další je v rámci této práce využit MEMS (*Micro Electro Mechanical System*) mikrofon, který dnes můžeme najít téměř ve všech moderních telefonech, diktafonech, či dalších inteligentních zařízeních. U mikrofonu je nutné vhodné načasování jednak hodin, se kterými se pracuje a jednak načasování čtení dat, která jsou ve formátu PDM. Jako poslední je zde implementována i paměťová SD karta, na kterou je na kitu připraven slot. Karta umožňuje záznam zvuku, zejména z mikrofonu, či již vygenerovaného tónu. U SD karty je nutné vhodně ošetřit a nastavit tzv. SPI rozhraní.

1.1 Cíle práce

Cílem práce je seznámení se s přípravkem Digilent Nexys4 a jeho obsluhou pomocí jazyka VHDL. Jako hlavní cíl je určeno připojení maticové klávesnice a její vhodné vyhodnocení, generování tónů, jeho zobrazení a jednoduchá indikace na segmentovém displeji. Jako dílčí cíle práce jsou navrženy zprovoznění mikrofonu a ukládání dat na SD kartu.

2 Teoretická část

2.1 Programovatelné logické obvody

Zkratka PLD (angl. *Programmable Logic Device*) se používá pro označení všech typů programovatelných logických obvodů. Mezi tři nejznámější architektury patří SPLD (angl. *Simple Programmable Logic Device*), CPLD (angl. *Complex Programmable Logic Device*) a FPGA (angl. *Field Programmable Gate Array*). Mezi největší výrobce patří firmy XILINX, Altera a Lattice Semiconductor. [6]

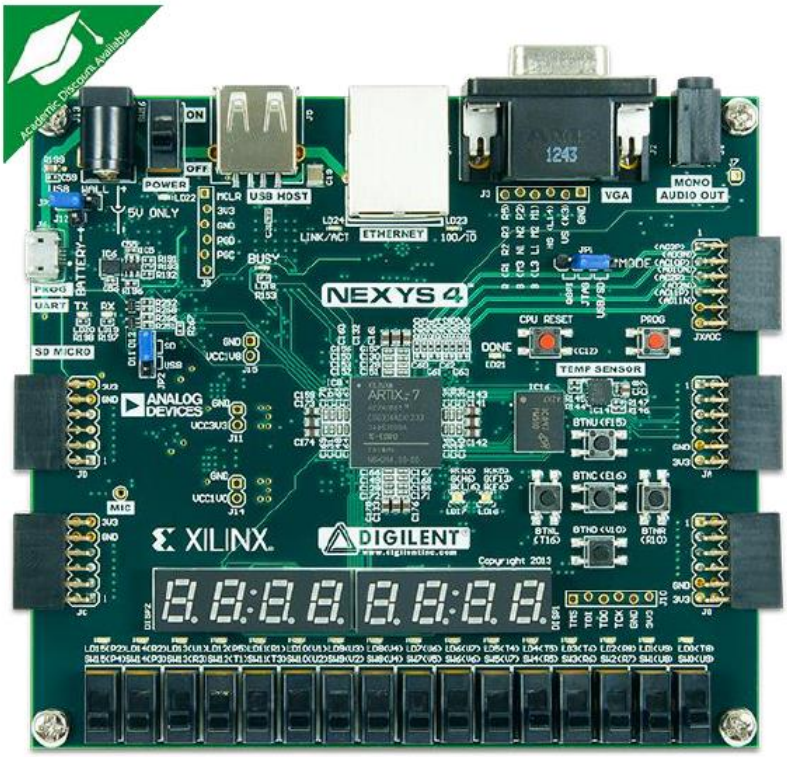
2.1.1 FPGA

Architektura FPGA patří do skupiny elektricky reprogramovatelných PLD obvodů. Tyto obvody jsou založeny na principu generátorů logických funkcí s paměťmi (LUT tabulky angl. *LookUp Tables*), klopných obvodech, vertikálními a horizontálními propojeními a na vstupně – výstupních programovatelných obvodech. Dnešní FPGA obvody obsahují i několik stovek tisíc LUT tabulek, což může vést i k několika desítkám miliónů ekvivalentních hradel. [6]

2.2 Nexys4

Digilent Nexys4 je platforma pro vývoj digitálních obvodů. Základ této platformy je hradlové pole Artix-7™ Field Programmable Gate Array (FPGA) od firmy Xilinx. Přípravek může hostovat řadu dalších zařízení od kombinačních obvodů po výkonné embedded procesory. Obsahuje několik periferních zařízení, jako akcelerometr, teplotní senzor, MEMS digitální mikrofon a audio výstup pro možnost zapojení reproduktoru. Dále je zde obsažena řada dalších portů, jako například VGA port, 10/100 Ethernet a USB port. Jako modulace se používají PDM (*Pulse Density Modulation* – pro mikrofon) a PWM (*Pulse Width Modulation* – pro audio výstup). Za zmínku určitě stojí i 16 přepínačů a ledek, dva čtyřmístné sedmi segmentové displeje, konektor pro SD kartu, či Pmod porty. K napájení kitu se používá 5 V a přípravek poskytuje 3,3 V napájení pro logické obvody. [1]

Nexys 4 je kompatibilní s novým vývojovým prostředním Vivado ® Design Suite i s ISE ®WebPACK™, který byl použit pro tuto práci. [1]



Obr. 2. 1: *Přípravek Nexys4 [1]*

2.3 VHDL

2.3.1 Charakteristika

Zkratka VHDL znamená *VHSIC Hardware Description Language*, kde *VHSIC* znamená *Very High Speed Intergrated Circuits*, čili jazyk pro popisování velmi rychlých integrovaných obvodů. Nejedná se tedy přímo o jazyk programovací. Vstupní údaje je nutné zapsat ve formě, kterou je model schopný převést na danou konstrukci. Zápis bývá nejčastěji textový nebo grafický (editory schémat s různými součástkami, stavové diagramy a podobně). Grafický zápis bývá obvykle nepřenositelný na jiný systém, proto se více používá zápis v textové formě. Jazyk VHDL byl původně vyvinut na Ministerstvu obrany USA s cílem vytvořit prostředek pro dokumentaci a popis chování integrovaných obvodů. První varianta jazyka VHDL byla zveřejněna v roce 1985 a na jejím vývoji se podílela například firma IBM. VHDL jazyk se také využívá k simulaci logických obvodů a psaní testovacích programů. Testovací program (*testbench*) slouží ke generování vstupních signálů a následné vyhodnocování. Překlad, tedy termín, který známe jako kompilace kódu není kompilace v jazyce VHDL, nýbrž syntéza. Syntézu neprovádí kompilátor, ale syntetizér, který syntetizuje obvod. [5] [9]

2.3.2 Nejdůležitější části

Každý syntetizovatelný VHDL kód sloužící pro realizaci digitálního obvodu v jazyce VHDL se skládá alespoň ze dvou jednotek: entity a architektury.

- **Entita** určuje především porty, které můžou být vstupní, výstupní, či vstupně-výstupní, to znamená, že se z nich dá číst, ale i zapisovat. Prostředí entity je pouze paralelní a jedna entita může mít více architektur. Pod pojmem entita si lze představit celkový výsledný digitální obvod uzavřený v pouzdře, mající vstupy a výstupy (porty), přičemž entitou (obvodem) může být klidně jedno jednoduché logické hradlo, ale například i složitý digitální obvod. [2, strana 25-26][3]
- **Architektura** specifikuje vstupně – výstupní záležitosti, to znamená jejich vzájemnou závislost. Architektura se dá dále rozdělit na tzv. Funkční a Strukturní, jediný rozdíl mezi nimi je, že Strukturní obsahuje oproti Funkční další vnořené entity a architektury. [2][3]
- **Komponenta**, pod tímto pojmem rozumíme využití jednodušší entity při realizaci složitější složené entity (entita v entitě). To si lze představit například jako digitální obvod složený z většího množství jednotlivých logických hradel, kde každé hradlo představuje samostatnou entitu (komponentu v celém výsledném obvodu). [2][3]

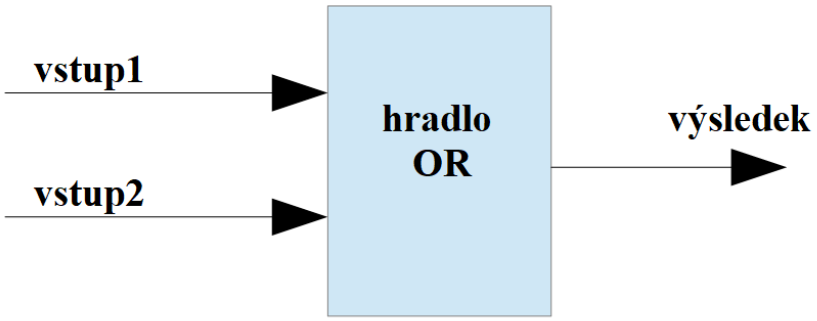
Mezi další důležitou část patří tři základní úrovně popisu: behaviorální, strukturní a data flow popis.

- **Behaviorální popis:** je nejblíže klasickému programování. Architektura je složena z jednoho nebo více procesů, které jsou popsány na úrovni algoritmu, to znamená podmínky, cykly, příkazy. Jedná se tedy o popis změn výstupních hodnot v závislosti na změnách vstupních signálů. [4][5]
- **Strukturní popis:** Popisuje, z čeho se dané zařízení či komponenta skládá, jakou má obvod strukturu. Zaměřuje se tedy na celky a jejich propojení pomocí signálu. [4][5]
- **Data flow:** Jak již název vypovídá, tento popis se nezaměřuje na celky jako je komponenta, nýbrž na to, jak se obvodem šíří data. Modeluje tedy datové závislosti. [4][5]

2.3.3 Příklad kódu

Jako příklad kódu jsem vytvořil jednoduchý logický součet, z čehož by mělo být jasnější, co přesně je entita a co architektura.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity soucet is
        port (
                vstup1: in std_logic;
                vstup2: in std_logic;
                výsledek: out std_logic);
end soucet;
architecture Behavioral of soucet is
-- zde může být deklarace pomocných proměnných a signálů
begin
výsledek <= vstup1 OR vstup2;
end Behavioral;
```



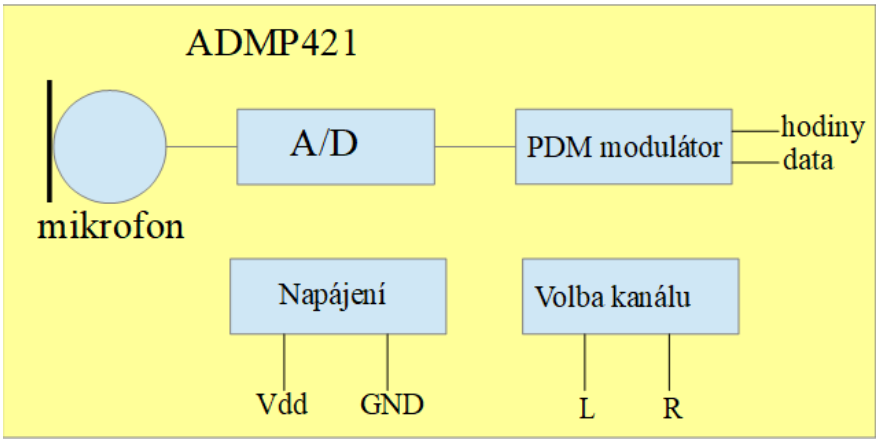
Obr. 2. 2: *obrázek pro příklad kódu v jazyce VHDL*

Ve VHDL kódu můžeme vidět entitu součet, která definuje pouze porty, což je vidět i na obrázku 2.2. Pokud by tedy byla napsána pouze entita, obvod by nedělal nic, protože jsme nedefinovali jeho chování architekturou. V architektuře se zpracují vstupní údaje a zapíší se na výstupní port "výsledek".

2.4 Použité komponenty a nástroje kitu Nexys4 při realizaci této práce

2.4.1 Mikrofon

Jako mikrofon je použit malý všesměrový model MEMS ADMP421 s rozměry 3 mm x 4 mm x 1 mm. Tento mikrofon má velký odstup signál šum (SNR = 61 dBA) a citlivost -26 dBFS, plochou frekvenční odezvu od 100 Hz do 15 kHz a malou proudovou spotřebu (<650 µA). Používá se Sigma-Delta modulátor čtvrtého řádu, výstup je tedy ve formátu digitálního PDM signálu. [1]



Obr. 2. 3: *Schéma mikrofonu MEMS ADMP421, a jeho vývody pro použití na FPGA. [vlastní zpracování dle: 7]*

Výstupy *DATA*, *L/R select* a *clk* je nutné ošetřit. Port *DATA* z mikrofonu je z našeho pohledu vstup, to znamená, že jej můžeme číst. Čtení je nutné vhodně nastavit, v závislosti na frekvenci a použité modulaci. Port *L/R select* je z našeho pohledu výstup, čili můžeme na něj zapisovat, v našem případě jej nevyužijeme, jelikož náš audio výstup nepodporuje stereo přehrávání. Jako poslední port je *clk*, tento port očekává námi vygenerovaný signál s určitou frekvencí, se kterou bude mikrofon pracovat, je to také výstupní port z našeho pohledu. S porty V_{dd} a *GND*, které představují napájení a zem se žádná akce neprovádí, jsou již připojeny na FPGA.

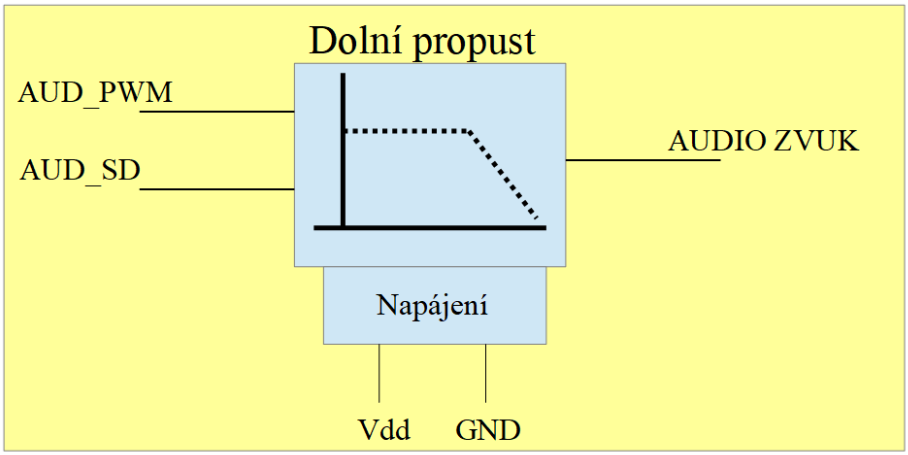
2.4.2 Maticová klávesnice

Jako maticová klávesnice je použit typ MR-MINI-4x4KEY. Tato maticová klávesnice má rozměry 7x6,5x0,5 cm a obsahuje 8 pinů, čili 4 řádky a 4 sloupce. Napájecí napětí +3,0V až +5,0V. U takovýchto mechanických kláves je nutné vhodné načasování, jelikož dochází k řadě zákmitů, ať už při stisku klávesy, nebo i bez něj.

Obr. 2. 4: *Maticová klávesnice [8]*

2.4.3 Audio výstup

Audio výstup je řízen pomocí Sallen-Key dolní propusti čtvrtého řádu. Vstup filtru je připojen na port A11, přičemž typicky se připojuje signál ve formátu PDM modulace, nicméně lze také použít signál ve formátu PWM modulace. Daný filtr tedy slouží k rekonstrukci analogového signálu z přijatého digitálního signálu. Tento filtr je obsažen na přípravku Nexys4, stačí tedy pouze vhodně připojit porty.



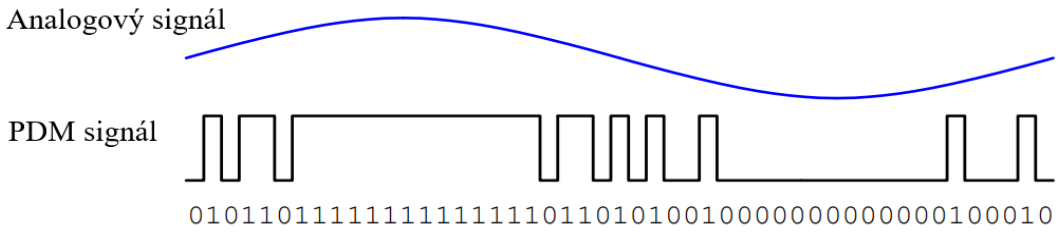
Obr. 2. 5: *Zjednodušené schéma dolní propusti [vlastní zpracování dle 1]*

U audio výstupu, čili dolní propusti pracujeme se dvěma porty, *AUD_PWM* a *AUD_SD*. Port *AUD_SD* je z našeho pohledu výstupní a musí být trvale držen na hodnotě logická jedna, pro správnou funkčnost obvodu a port *AUD_PWM* je z našeho pohledu také výstupní a je tedy možnost na něj zapisovat data, která by měla být namodulována buďto ve formátu PWM, či PDM. S porty *Vdd* a *GND* se žádné operace neprovádějí, jsou již připojeny na přípravku Nexys4. Výstupem z obvodu dolní

propusti je tedy audio zvuk. Vše je znázorněno na obrázku 2.5, jedná se o velice zjednodušený obrázek. Celý obvod dolní propusti je dán do přílohy.

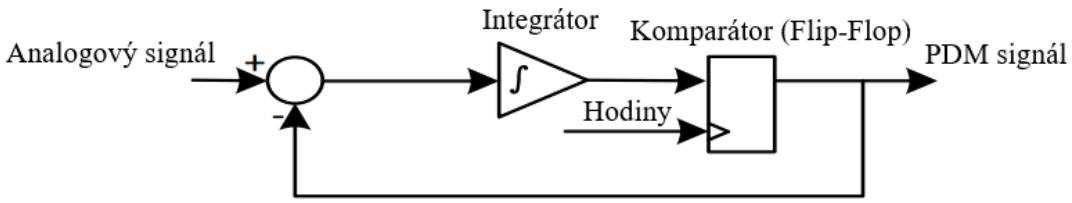
2.4.4 PDM

PDM (*Pulse Density Modulation*) modulace je využívána pro audio aplikace, například u mobilních telefonů a tabletů. Typická frekvence u PDM signálu kolísá mezi 1 MHz až 3 MHz, bitový tok pro 1 odpovídá napájecímu napětí (*Vdd*), naopak 0 odpovídá nulovému napětí (*GND*). PDM signál je produkován Delta-Sigma modulátorem. [1]



Obr. 2. 6: *Ukázka PDM signálu [1]*

Delta-Sigma modulátor:



Obr. 2. 7: *Delta – Sigma modulátor [1]*

Delta-Sigma modulátor obsažený na přípravku pracuje za předpokladu, že analogový vstup a digitální výstup mají stejný napěťový rozsah 0 až V_{dd}. Vstup Flip-Flop obvodu funguje jako komparátor, každý signál s amplitudou pod polovinu napájecího napětí je vyhodnocen jako logická 0 a každý signál s amplitudou nad polovinu napájecího napětí jako logická 1, viz tabulka 2.1. Vstup integrátoru je rozdíl vstupů analogového signálu a PDM signálu předešlého hodinového pulzu. Výstup integrátoru je vzorkován Flip-Flop obvodem s frekvencí, kterou si určíme vstupem ‘clk’. [1]

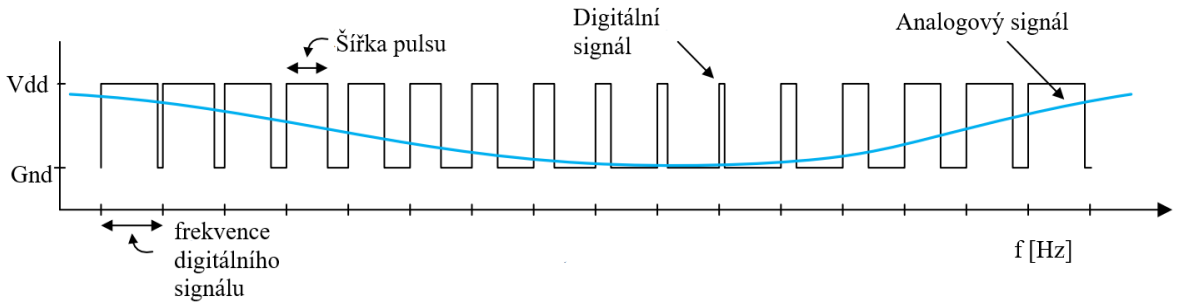
Suma	Výstup integrátoru	Výstup komparátoru
0,4 - 0 = 0,4	0 + 0,4 = 0,4	0
0,4 - 0 = 0,4	0,4 + 0,4 = 0,8	1
0,4 - 1 = -0,6	0,8 - 0,6 = 0,2	0
0,4 - 0 = 0,4	0,2 + 0,4 = 0,6	1
0,4 - 1 = -0,6	0,6 - 0,6 = 0	0
0,4 - 0 = 0,4	0 + 0,4 = 0,4	0
0,4 - 0 = 0,4	0,4 + 0,4 = 0,8	1
0,4 - 1 = -0,6	0,8 - 0,6 = 0,2	0

Tab. 2. 1: *Průběh vyhodnocení logických úrovní u Sigma-Delta modulátoru pro 0,4Vdd . [1]*

2.4.5 PWM

Pulsně šířková modulace (*PWM*) se obvykle používá při řízení otáček elektromotorů, stmívání světelných zdrojů a podobných aplikacích, ve kterých dochází k výkonovému řízení. Je to jeden ze způsobů, jak pomocí logických hodnot nasimulovat analogový signál. Pulsně šířková modulace má široké využití. Mezi nejčastější využití modulace lze zařadit řízení otáček motorů nebo použití při audio technice. Hlavními znaky této modulace je konstantní frekvence a měnící se střída signálu, tedy doba, po kterou je signál v hodnotě logická jednička a logická nula. Jednoduše řečeno, PWM modulace je řetězec pulsů konstantní frekvence, ale různé šířky (střídy). [12]

Pokud takovýto signál přivedeme na vstup dolní propusti, získáme na výstupu analogový signál o amplitudě, která je rovná právě nahromaděné energii v daném pulsu. Čím širší je tedy puls, tím větší amplituda se objeví na výstupu dolní propusti. Šířku pulsu udává takzvaný *duty cycle* (střída) a amplituda analogového signálu je omezena přivedeným napětím. Celý tento popis je zobrazen na obrázku 2.8. [1]



Obr. 2. 8: *Namodelovaný PWM signál a jemu odpovídající analogový signál po průchodu DP [1]*

2.4.6 SD karta

Secure Digital (SD) je formát standardu navržený a licencovaný organizací *SD card association* (SDA). Představen byl roku 1999 firmami SanDisk, Panasonic a Toshiba, jejichž společné úsilí jej umožnilo vyvinout. SD karta je přenositelné zařízení typu flash. Díky malým rozměrům, velkému uložišti a solidním přenosovým rychlostem patří v dnešní době mezi nejpopulárnější paměťové karty. [11]

2.4.6.1 Přehled

SD karty zahrnují čtyři rodiny ve třech rozdílných velikostech.

Tyto čtyři rodiny jsou:

1. SDSC (*Secure Digital Standard Capacity*)
2. SDHC (*Secure Digital High Capacity*)
3. SDXC (*Secure Digital eXtended Capacity*)
4. SDIO (*Secure Digital Input Output*)

Výše zmíněné tři velikosti ovšem nejsou dostupné pro všechny typy těchto karet, pouze první dvě (SDSC a SDHC) mají všechny tři velikosti (standartní, mini a micro). Karty SDXC nejsou dostupné v mini verzi a SDIO nejsou dostupné v micro verzi. [10] [11]

2.4.6.2 Adresování

SD karta je rozdělena na segmenty, které jsou velké 512 B. Každý segment má 16 sloupců a 32 řádků. Adresa prvního segmentu tedy začíná na 00000000(16), druhý segment 00000200(16) a tak dále. Každý další segment začíná s adresou +200 v šestnáctkové soustavě. K zobrazení dat na SD kartě byl použit program HxD, který zobrazí veškerá data v na daných bajtech segmentů.

2.4.6.3 Technické detaily

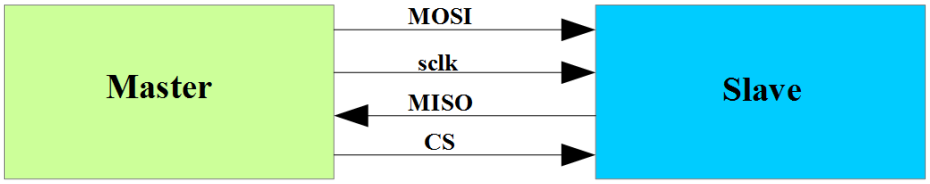
Mezi řekl bych nejpodstatnější technické detaily patří přenosové módy, které SD karty podporují. Těch existuje celá řada, nicméně mezi dva nejvíce používané patří SPI mód a SD mód

- **SPI mód**, který je primárně používán v *embedded* zařízeních, podporuje rozhraní pouze 3,3 V a nevyžaduje žádné povolení ani licenci, nevýhoda je nižší přenosová rychlost. Tento standard popisuje pouze fyzické rozhraní, datový protokol se může u různých zařízení lišit. [10] [11]
- **SD mód**, je rozdělen na jednobitový a čtyřbitový. V tomto módu je datový přenos zabezpečen pomocí 16-ti bitového CRC (*Cyclic Redundacy Check*) kódu. Což je další z rozdílů oproti SPI módu, kde se s CRC kódem pracuje jen při počáteční inicializaci. Rozdíl mezi jednobitovým a čtyřbitovým přenosovým režimem je v přenosové rychlosti, která je odvozena od počtu portů, které přenášejí data. U jednobitového je datový paket odeslán od MSB (*Most Significant Bit*) po LSB (*Less Significant Bit*), ale u čtyřbitového jsou data rozdělena do čtyř datových portů. [10] [11]

2.4.6.4 SPI rozhraní

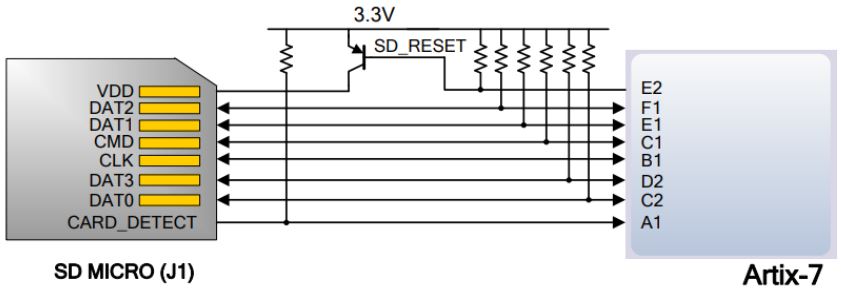
V SPI rozhraní jsou použity čtyři vodiče:

- ***sclk***: Tento vodič (port) je určen pro hodiny (frekvenci) se kterou bude rozhraní pracovat. Nutné je podotknout, že SPI rozhraní může pracovat s frekvencí až 25 MHz, nicméně v počáteční fázi, kdy probíhá inicializace, je nutné řídit rozhraní frekvencí mezi 100-400 kHz.
- ***CS***: Port CS (*Chip Select*) někdy také označován jako SS (*Slave Select*) je určen pro výběr zařízení, pokud pracujeme s více *slave* zařízeními.
- ***MOSI***: Port MOSI (Master Output Slave Input) slouží ke komunikaci směrem od přípravku Nexys4 do SD karty. Jsou skrze něj posílány příkazy pro *slave* obvod (SD karta) a také data, která požadujeme zapsat.
- ***MISO***: Port MISO (Master Input Slave Output) slouží ke komunikaci směrem od SD karty do přípravku Nexys4. Jsou skrze něj posílány odpovědi (tokeny) a také data, která *master* obvod (Nexys4) požaduje číst. [10] [11]



Obr. 2. 9: *Komunikace mezi přípravkem Nexys4 a SD kartou v SPI módu[11]*

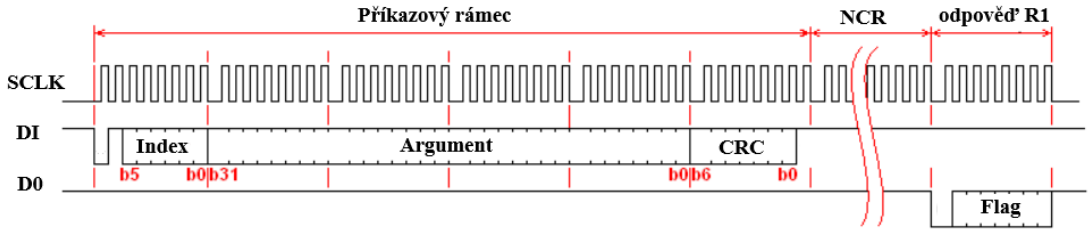
Na obrázku 2.10 můžeme vidět, jak je SD micro karta propojena s přípravkem Nexys4, nutné je ovšem podotknout, že toto zapojení platí všeobecně pro rozhraní, při použití SPI módu se nikterak neoperuje s porty DAT1 a DAT2. Základní princip tohoto módu je *Master-Slave* komunikace (Obr 2.9), *master* tedy řídí tok dat na sběrnici. Nexys4, čili *master* se ptá a karta odpovídá. Otázky jsou formou takzvaných *cmd* příkazů a odpovědi jsou specifické tokeny, pro režim SPI jsou definovány tokeny typu R1, R2 a R3. [10][11]



Obr. 2. 10: *Porty SD micro karty propojené s přípravkem Nexys4 (Artix-7)[1, strana 22]*

2.4.6.5 Příkazy a odpovědi

Každé operaci musí předcházet daný příkaz, aby karta věděla, co se po ní žádá. Tento příkaz začíná vždy sekvencí bitu 01, které následuje šestibitový index příkazu. Dalších 32 bitů určuje argument příkazu, zakončené osmi bity kontrolního součtu, viz obr 2.11. Pomocné signály DO a DI řídí celý proces komunikaci (určují kdy přesně se daný příkaz/odpověď pošle). Příkaz je vždy následován odpovědí, prodlevu mezi těmito dvěma událostmi určuje Ncr (*Command Response Time*). [9][10]



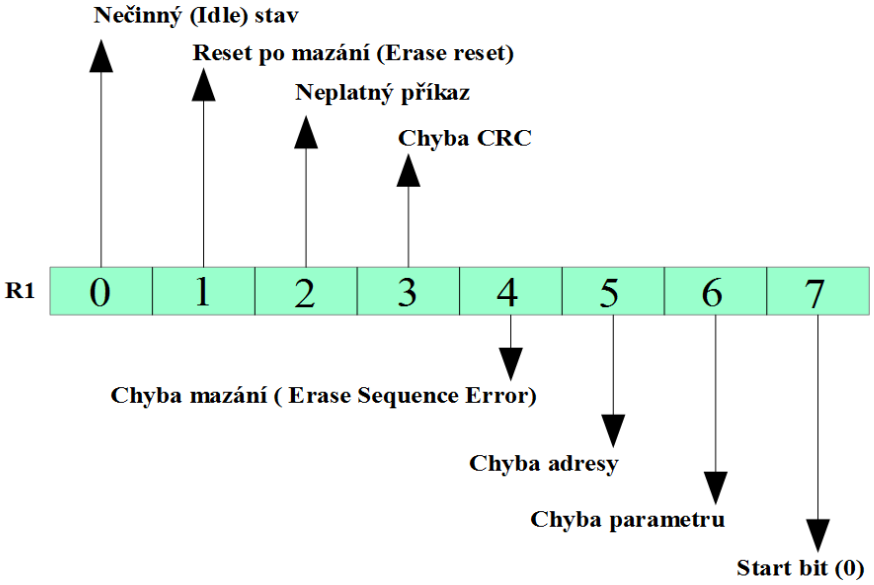
Obr. 2. 11: *Příklad časování poslání příkazu a přijaté odpovědi [11]*

- **Příkazy:** SD příkazy jsou posílány ve formě “CMDn“ a “ACMDn“, kde n je číslo v rozsahu 0-63 daného příkazu. Následující tabulka 2.2 popisuje příkazy, které jsem byl nucen použit při implementaci SD karty.

Příkaz	Argument	Typ odpovědi	Popis
CMD0	žádný	R1	Reset
CMD1	žádný	R1	Inicializace.
CMD41	žádný	R1	Jako CMD1, doporučeno pro SDC karty.
CMD55	žádný	R1	Nutno použít před CMD41.
ACMD41			CMD55 následován CMD41.
CMD17	Adresa 32 bitů	R1	Čti blok dat.
CMD24	Adresa 32 bitů	R1	Zapiš blok dat.

Tab. 2. 2: *Tabulka použitých příkazů a jejich popis [11]*

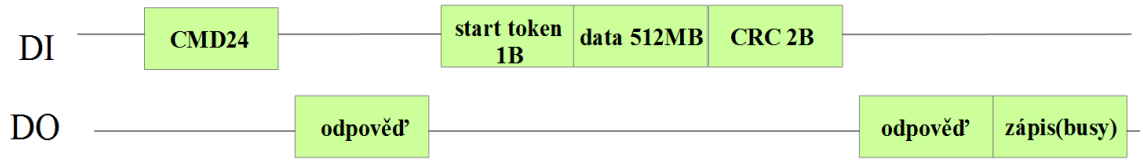
- **Odpovědi:** Jak již bylo řečeno odpovědí je několik druhů v závislosti na příkazu a použitém módu. V mé práci pracuji pouze s odpovědí typu R1, proto je zde popsán jen tento jeden případ.



Obr. 2. 12: *Odpověď typu R1 [vlastní zpracování dle: 11]*

Na obrázku 2. 12 lze vidět, co znamenají jednotlivé bity odpovědního rámce R1. Jestliže je jen jeden z těchto bitů nastaven na hodnota logická 1, nelze pokračovat v operaci, která byla vyžadována. Dle nastaveného bitu musíme odhalit chybu, opravit ji a následně se pokusit o znovu odeslání daného příkazu. [9][10]

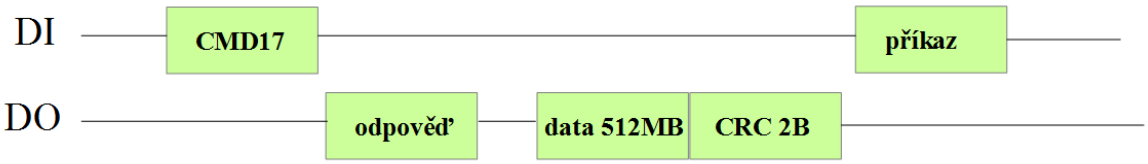
2.4.6.6 Zápis



Obr. 2. 13: *Průběh zápisu na SD kartu v SPI módu [vlastní zpracování dle: 11]*

Zápis probíhá po odeslání a přijetí kladné odpovědi. Možné je zapisovat jak po blocích, tak kontinuálně, tato práce se věnuje pouze zapisování po blocích, proto je použit příkaz CMD24. Blok dat je fixně nastaven na velikost 512 B, ale u některých karet SD je možné ji změnit. Tento blok začíná takzvaným *start tokenem*, který určuje začátek bloku a končí šestnáctibitovým CRC kódem, který se ovšem používá pouze při inicializaci. [10][11]

2.4.6.7 Čtení



Obr. 2. 14: *Průběh čtení z SD karty v SPI módu [vlastní zpracování dle: 11]*

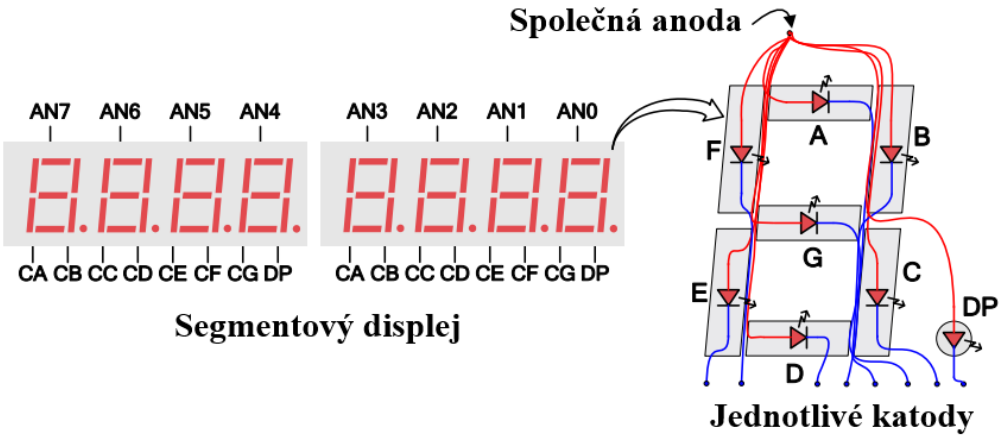
Čtení probíhá po odeslání příslušného příkazu (CMD17) a následné kladné odpovědi (žádné chyby, čili samé logické nuly). Číst data je možné po blocích či kontinuálně. Blok dat není možné číst odkudkoliv. Adresa, která je argument v příkazu CMD17 musí být přesně nastavena na začátek bloku. [10][11]

2.4.7 Segmentový displej

Zobrazení požadovaných hodnot probíhá pomocí segmentového displeje. Segmentový displej na přípravku Nexys4 se skládá z osmi 7-segmentových displejů, přičemž u každého displeje je možnost použít i desetinou tečku. To znamená, že jeden displej je tvořen z celkem osmi LED diod.

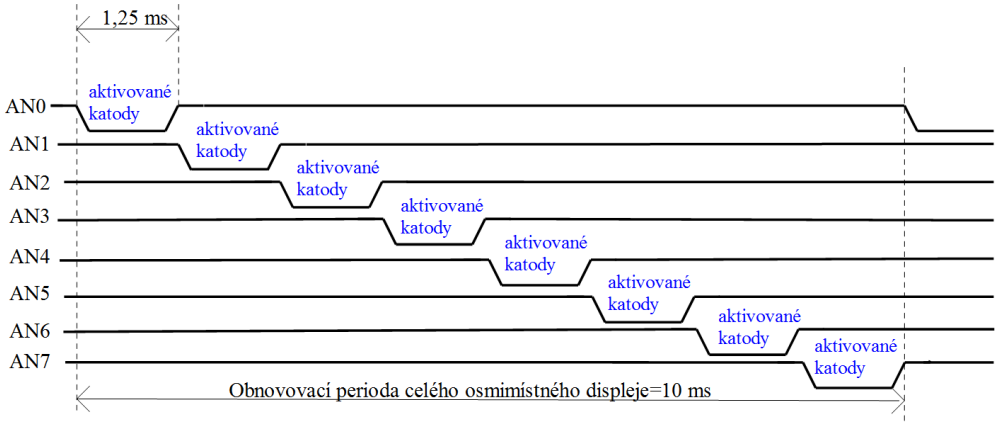
Anody těchto diod jsou u jednoho displeje spojeny do jedné, takzvané společné anody. Katody těchto diod zůstávají oddělené z pohledu jednoho displeje, nicméně z celkového pohledu všech osmi displejů, jsou spojeny vždy stejné katody jednotlivých displejů. Například segment s označením B má v rámci jednoho displeje spojenou anodu s ostatními anodami daných segmentů, nicméně katoda je spojena s dalšími displeji a jejich segmentem B, tak vzniká označení CA, CB atd. Nastává tedy otázka, proč se nerozsvítí při aktivaci daného segmentu B všechny segmenty na všech osmi displejích. Je to způsobeno tím, že daný rozsvícený segment musí být ve shodě se zrovna aktivovanou společnou anodou. Pro lepší představení je vše zobrazeno na obrázku 2.15. Osm společných anod AN0 až AN7 slouží tedy k vybrání pozice na displeji a katody CA až DP slouží k rozsvícení jednotlivých segmentů.

Nastavování pozic a rozsvícení daných segmentů by mělo probíhat řízením logickými jedničkami u anod a logickými nulami u katod. Nicméně obojí (pozice i segment) jsou řízeny nulami, jelikož pro řízení anod je použit tranzistor, který invertuje vstupní hodnotu. [1, strana 19-20]



Obr. 2. 15: *Zapojení diod v segmentovém displeji [1, strana 20]*

Výše popsaný způsob dovoluje zobrazit pouze jeden symbol, pro zobrazení více symbolů je nutné využít vhodného načasování. K tomu se využije nedokonalost lidského oka. Postupně se tedy musejí zobrazovat jednotlivé pozice na displejích, a to s takovou frekvencí, které není lidské oko schopno postřehnout. K přepínání se doporučuje obnovovací frekvence vyšší než 60 Hz (16 ms), pokud zvolíme tedy frekvenci 100 Hz (10 ms) na jednu část displeje, při použití osmimístného displeje zůstává na jednu pozici přibližně 10/8 = 1,25 ms, což odpovídá frekvenci 800 Hz. [1, strana 19-20]



Obr. 2. 16: *Průběh aktivace jednotlivých segmentů displeje [vlastní zpracování]*

3 Praktická část – vlastní realizace

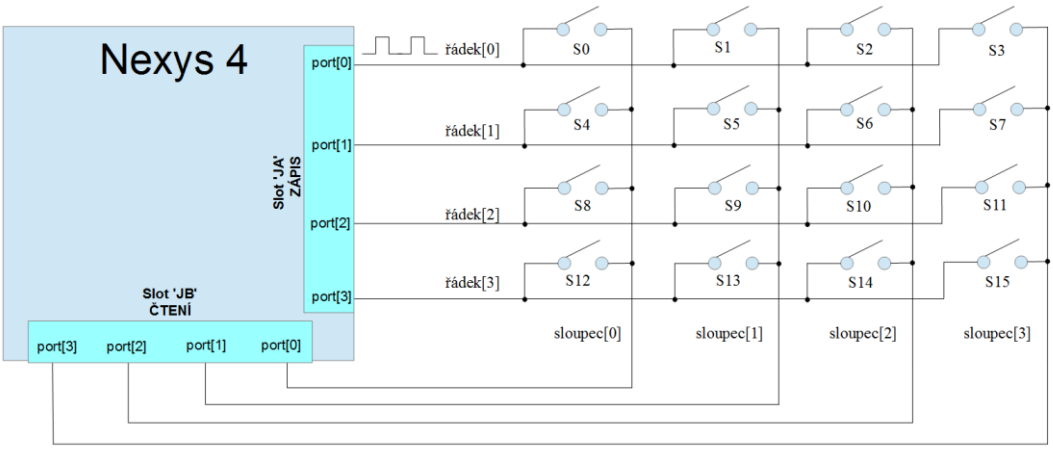
V následujících kapitolách se věnuji realizaci mé práce, popsány jsou všechny části, které byly použity při psaní kódu. Tyto části se kryjí s kapitolami v úvodní části práce. Kód není rozdělen do komponent, je vytvořen jako jeden celek s řadou procesů. Jednotlivé procesy se spustí vždy pokud dojde ke změně jeho parametrů, nicméně pro další postup je nutné nastavit vhodné spínače, popřípadě tlačítka. VHDL kód není přímo popisován. K popisu jsou použity názorné obrázky a diagramy pro lepší pochopení. Vlastní VHDL kód je pak celý uveden v příloze této práce.

3.1 Proces maticové klávesnice

Použité porty: řádek, výstupní čtyřbitový vektor, 4 porty

sloupec, vstupní čtyřbitový vektor, 4 porty

Maticová klávesnice je připojena do dvou takzvaných Pmod portů, které obsahuje přípravek Nexys4. Dle obrázku 3.1 můžeme vidět, jak jsou přesně připojeny dvě čtveřice, které vedou z klávesnice do daného přípravku. Jedna čtveřice tedy odděluje řádky a druhá sloupce.



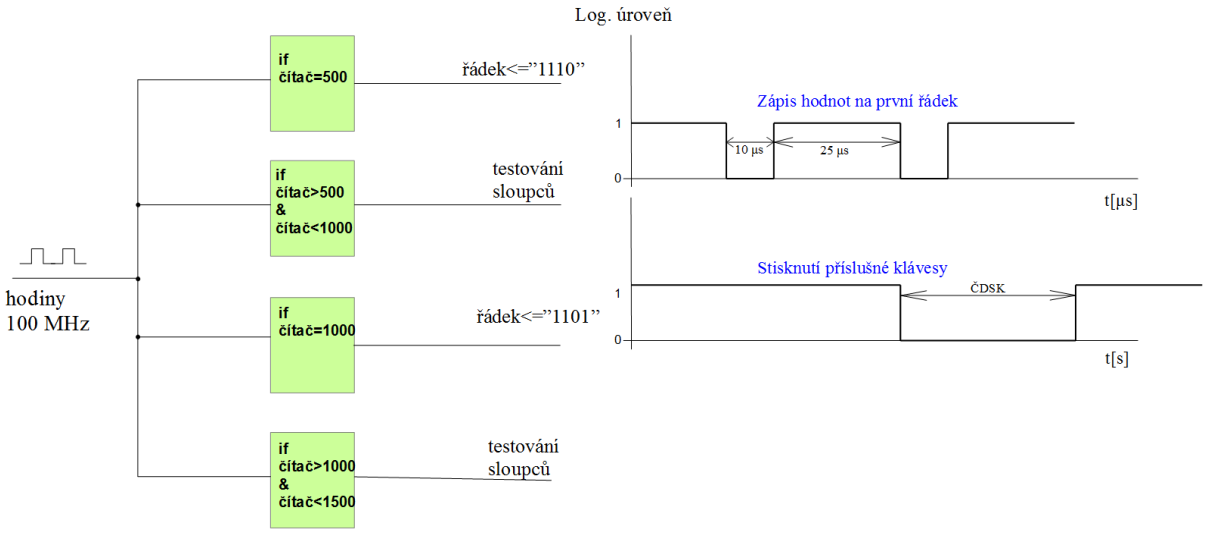
Obr. 3. 1: *Zapojení maticové klávesnice do přípravku Nexys4 [vlastní zpracování]*

Klávesnice funguje na principu zápis a čtení, jak je znázorněno na obrázku 3.2. Na němž můžeme vidět počáteční hodinový signál o frekvenci 100 MHz, který tvoří základ pro řízení procesu čtení a zápis. Použit je pomocný *čítač*, který počítá náběžné hrany našeho základního hodinového signál. Následný zápis a čtení probíhá tedy v určitém intervalu hodnoty *čítače*.

- **Zápis** hodnoty do určitého *n-tého* (*n* je 1-4) řádku nastane v *x-tém* okamžiku našeho *čítače*. Tato zapsaná hodnota zde zůstane do další změny, tedy do dalšího zápisu.
- **Čtení** nastává v okamžiku, kdy jsme zapsali onu hodnotu. Při čtení je nutné postupně ve vhodných okamžicích přečíst všechny čtyři sloupce a pomocí podmínek následně vyhodnotit, která klávesa byla stisknuta.

Důležité je jasně definovat, co a jak se má kdy udělat, nesmí nastat takzvané neurčité stavy. V takových stavech nevíme, co se přesně děje na přípravku. Co není znázorněno na obrázku je fakt, že prvních 500 náběžných a posledních 500 náběžných hran se nečte ani nezapisuje. Toto omezení bylo

nutné zavést po několika experimentálních pokusech, jelikož se klávesnice bez nich nechová tak, jak očekáváme.



Obr. 3. 2: *Průběh zápisu a čtení maticové klávesnice [vlastní zpracování]*

Dále můžeme na obrázku 3.2 vidět časový průběh, který znázorňuje dobu, po kterou se zapisuje a neurčitou dobu, po kterou je stisknuta klávesa – ČDSK (Čas Doby Stisknuté Klávesy). Jelikož z tohoto časového průběhu plyne fakt, že delší doba stisknuté klávesy se vyhodnotí pouze každých deset mikrosekund, bylo nutné zavést pomocnou proměnou, která v sobě “drží“ hodnotu logická jedna po dobu, dokud se klávesa nepustí. Takto je tedy ošetřeno správné čtení. Nebylo by to nutné, pokud bychom s výstupní hodnotou pracovali dále, třeba pouze na displeji, kde by tato frekvence bohatě stačila pro zobrazení vlivem nedokonalosti lidského oka, které dokáže vnímat blikání pouze do určité frekvence.

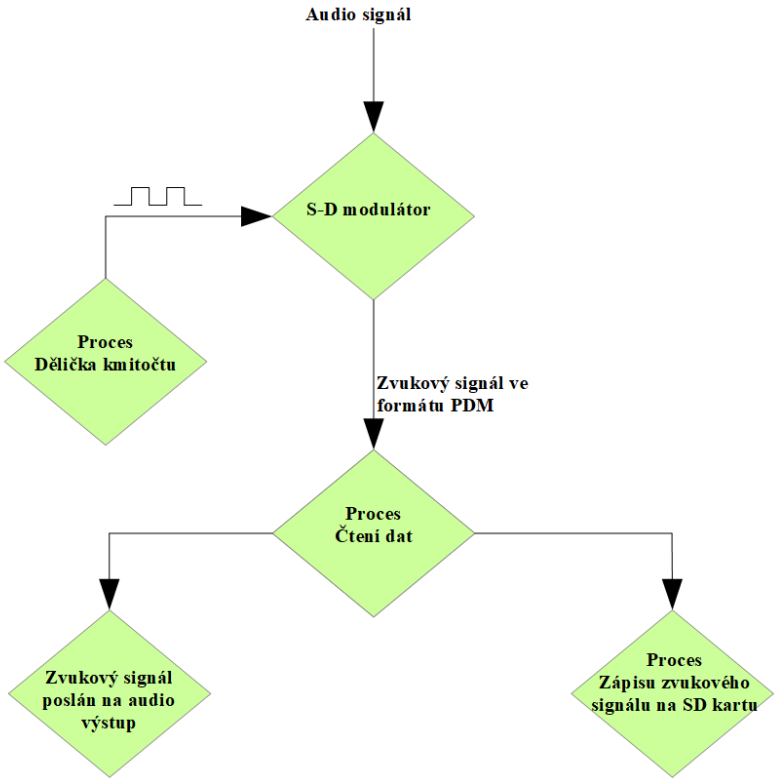
3.2 Proces pro mikrofon

Použité porty: hodiny pro mikrofon, výstupní port

nastavení pravého/levého kanálu pro stereo, výstupní port

data, vstupní port

Mikrofon je již implementován na přípravku Nexys4. Je tedy nutné vhodné vhodně nastavit zejména frekvence, se kterými se bude analogový signál zpracovávat ve Flip-Flop obvodu a frekvenci/časy ve kterých se budou číst data, která jsou výstupem Sigma-Delta modulátoru. Nastavení portu pro stereo je nastaveno trvale na hodnotu logická 0, jelikož audio výstup na přípravku podporuje pouze mono vysílání.



Obr. 3. 3: *Procesní diagram pro mikrofon [vlastní zpracování]*

Celý proces, který se týká mikrofonu, je zobrazen na obrázku 3.3. Základem je analogový signál (hlas, zvuk), který přijímá mikrofon, jehož součást je *Sigma-Delta modulátor*. Tento modulátor očekává určitý takt, se kterým bude pracovat jeho *Flip-Flop* obvod. Dle doporučení od výrobce přípravku je použit takt o frekvenci 2,4 MHz. Výstupem modulátoru jsou *data ve formátu PDM*, tato data se musí nějak zpracovat, k tomu slouží *blok procesu čtení dat*. Tento proces pracuje se dvěma frekvencemi. První frekvence (časování) je zápis na SD kartu, popřípadě na audio výstup, tato frekvence je 100 MHz, ale může být i nižší. Důležité je zachovat frekvenci pro čtení na 2,4 MHz. To znamená, že pokud budeme používat " zapisující" frekvenci 100 MHz a čtecí frekvenci 2,4 MHz je nutné použít děličku 1:42. Jakmile se tedy napočítá 42 náběžných hran frekvence 100 MHz, přečtou se data, která jsou momentálně výstupem *Sigma-Delta modulátoru*. Následujících 42 náběžných hran je tato hodnota zapisována na výstup.

3.3 Proces pro SD kartu

Přípravek Nexys4 obsahuje slot na SD kartu, není tedy nutné dodělávat další přídavné obvody. Pro SD kartu bylo nutné napsat vhodný stavový automat, který se skládá ze tří hlavních částí: inicializace, čtení a zápis. Celý proces těchto tří částí lze vidět na obrázku 3.4. Nutno podotknout, že celý proces je napsán pouze pro karty typu SDSC (standartní kapacita), pokud bychom chtěli připojit kartu typu SDHC nebo SDXC, bylo by nutné kód poupravit, a to zejména v části inicializační.

3.3.1 Inicializace

Proces inicializace začíná resetem (přivedením logické jedničky a následné logické nuly na výstupní port reset), což umožní kartu restartovat a zavést tak režim SPI. Po zavedení resetu se

doporučuje vyčkat minimálně 70 dalších pulzů, než se stavový automat posune dál. Ve stavu *reset* je tedy zaveden tento fakt a to tak, že se nastaví hodnota pro čekání na 160. Než se tedy celý proces inicializace posune dál, je nutné vyčkat 160 náběžných hran hodinového signálu. Toto čekání se odehrává ve stavu *Init*. Jako další ze stěžejnějších věcí, které obsahuje tento stav je nastavení adresy, na kterou se má zapisovat, nebo číst. Tato adresa je nastavena na začátek osmého sektoru SD karty a při každém resetu se nastaví opět na začátek osmého sektoru, tím je dosaženo funkce zápisu-reset-čtení. Toto řešení je zavedeno, protože SD karta není primárním cílem této práce.

Po vyčkání určité doby se vyšle sekvence příkazů CMD0, CMD55 a CMD41 a počká se na odpověď od SD karty. Odpověď přijde ve formě tokenu R1 o velikosti jeden bajt. Pokud jsou všechny bity nastaveny na hodnotu logická nula, může se přejít do *idle* stavu a vyčkat na další instrukce od uživatele. Pokud je první bit nastaven na hodnotu logická jedna, je nutné opakovat celý proces od příkazu CMD55. V případě, že ostatní bity nejsou nastaveny na hodnotu logická nula, dojde k chybové hlášce a karta se musí vyresetovat, jelikož většinou zamrzne. Nicméně nutno dodat, že toto není v práci ošetřeno, kontroluje se pouze první bit.

3.3.2 Čtení

Pokud chceme z SD karty číst data, je nutné nejdříve poslat příslušný příkaz(CMD17), aby karta věděla, co se po ní žádá. Následuje stav *čti_blok_čekej*, ve kterém se (jak název vypovídá) čeká. Čeká se na volný port *miso,* po kterém by se ještě mohla přenášet data z předchozích instrukcí. Dále se v tomto bloku nastaví velikost jednoho bloku dat a velikost jednoho bajtu.

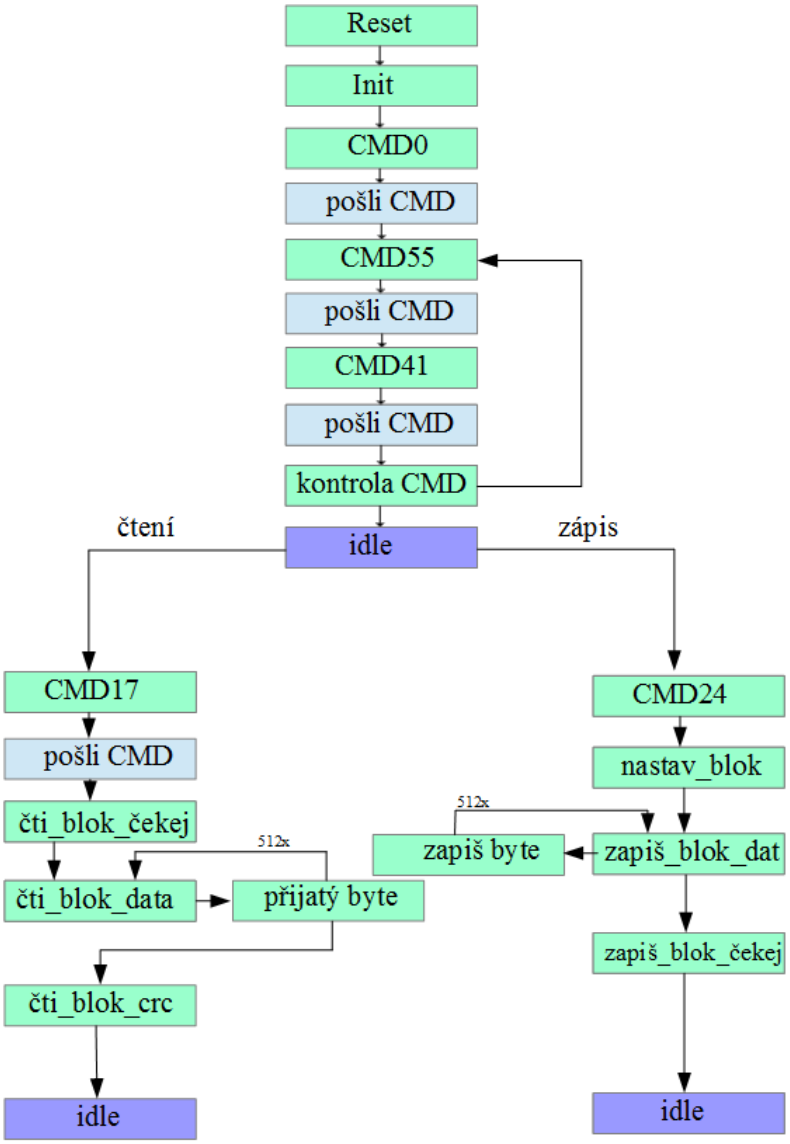
Po dokončení se přejde do stavu *čti_blok_data*. Tento stav je zacyklený po 512 taktů se stavem *přijatý_byte*. Dochází zde ke spolupráci těchto dvou stavů, kde stav *čti_blok_data* posouvá dané adresy sektoru, čili bajty a stav *přijatý_byte* čte z portu *miso* pomocí funkce *zřetezení*. Po přečtení osmi bitů (jednoho bajtu) se stav vrátí zpět do *čti_blok_data*, kde se tento celý bajt může číst najednou. Následně dojde opět k odečtení jedničky, tím se přesune celý proces k dalšímu z bajtů v daném sektoru. Po přečtení posledního bajtu, dojde k přesunu ze stavu přijatý_byte do stavu čti_blok_crc. Tento stav je takto pojmenován, jelikož by se mělo číst CRC ověření, nicméně to se u SPI módu nevyužívá, a tak dojde pouze k přičtení adresy o 200 (šestnáctková soustava) a nastavení hodnoty jednoho bajtu. Po dokončení se přejde do stavu *idle,* kde dle nastavení se pokračuje ve čtení.

3.3.3 Zápis

Celý tento proces začíná ve stavu *idle,* kde se musí nastavit spínač určený pro zápis. Při zápisu na SD kartu je nutné nejprve vyslat daný příkaz, aby karta věděla, co se po ní žádá. Po příkazu CMD24 se přechází do stavu nastav_blok, kde se nastaví především velikost bloku zapisovaných dat na 515, což je vysvětleno na obrázku 2.13.

Po dokončení se přejde do stavu *zapiš_blok_dat*, který je zacyklený po 515 taktů se stavem *zapiš_byte*. Ve stavu *zapiš_blok_dat* se tedy děje právě posouvání daných bajtů pomocí čítače, na které se zrovna zapisuje, ošetřen je pomocí podmínky první bajt (číslo 515) a poslední dva (1 a 2). Zbylých 512 bajtů se zapisuje při každém cyklu přes zavedení stavu *zapiš_byte*, kde by se měly číst jednotlivé bity a zápis celého bajtu by měl probíhat ve stavu *zapiš_blok_dat*, nicméně v mé práci je to lehce poupraveno/zjednodušeno. Nedochází k zápisu osmi bitů do jednoho bajtu, ale pouze jednoho bitu do jednoho bajtu, ostatní bity daného bajtu jsou vyplněny logickými nulami. Jakmile čítač dopočítá na hodnotu 0, dojde k zavedení dalšího stavu. Tento stav se jmenuje *zapiš_idle_čekej*. Dochází v něm

především k čekání, než se uvolní port *miso* a také následné změny adresy segmentu, do kterého se bude zapisovat při dalším požadavku na zápis.

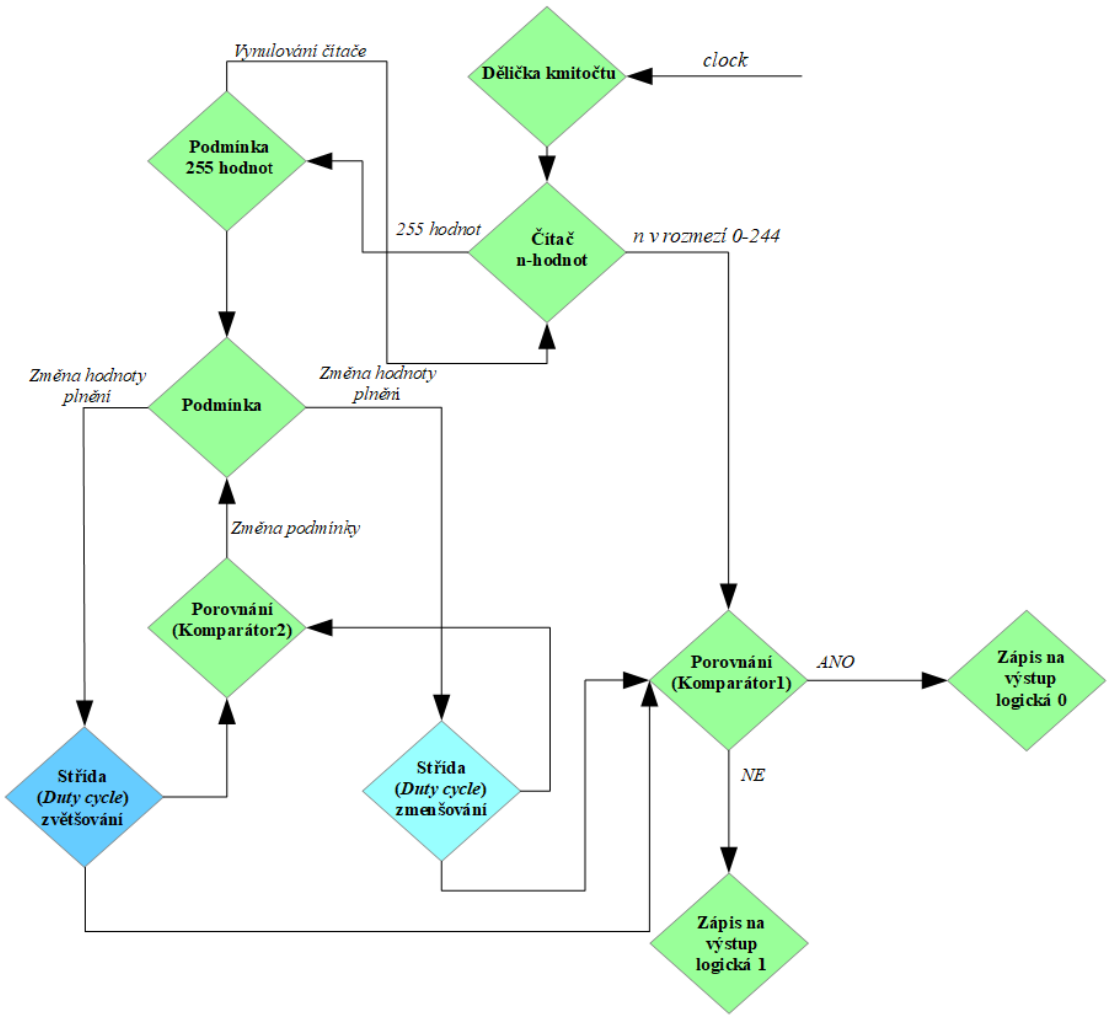


Obr. 3. 4: *Stavový diagram pro SD kartu pracující v SPI módu. Inicializace, čtení a zápis. [vlastní zpracování]*

3.4 Proces generování tónů

Ke generování tónů je využita pulsně šířková modulace. Teoreticky jde pouze o to, vydělit frekvenci, kterou nám poskytuje přípravek Nexys4 a přivést ji na vstup audio. Pro toto bychom potřebovali pouze čítač, který by počítal pulsy a při nějakém našem určeném momentu by se vynuloval, změnila by se tedy hodnota námi požadovaného signál a tím by se získal signál s určitou frekvencí, ale neměnnou střídou. Nicméně pokud chceme signál ve formátu PWM je zapotřebí ještě komparátor, který porovnává hodnotu čítače se zvolenou hodnotou vzorku a určuje tak dobu, po kterou je PWM signál v hodnotě logická jedna a logická nula (určujeme střídu). V této práci jsou generovány tóny s nekonstantní střídou a konstantní frekvencí. Frekvence je odvozena od komparační úrovně děličky kmitočtu a počtu bitů vzorků, které jsou použity.

$$f_{PWM} = \frac{frekvence\ hodin}{komparační\ úroveň\ děličky + 2^{počet\ bitů}}$$

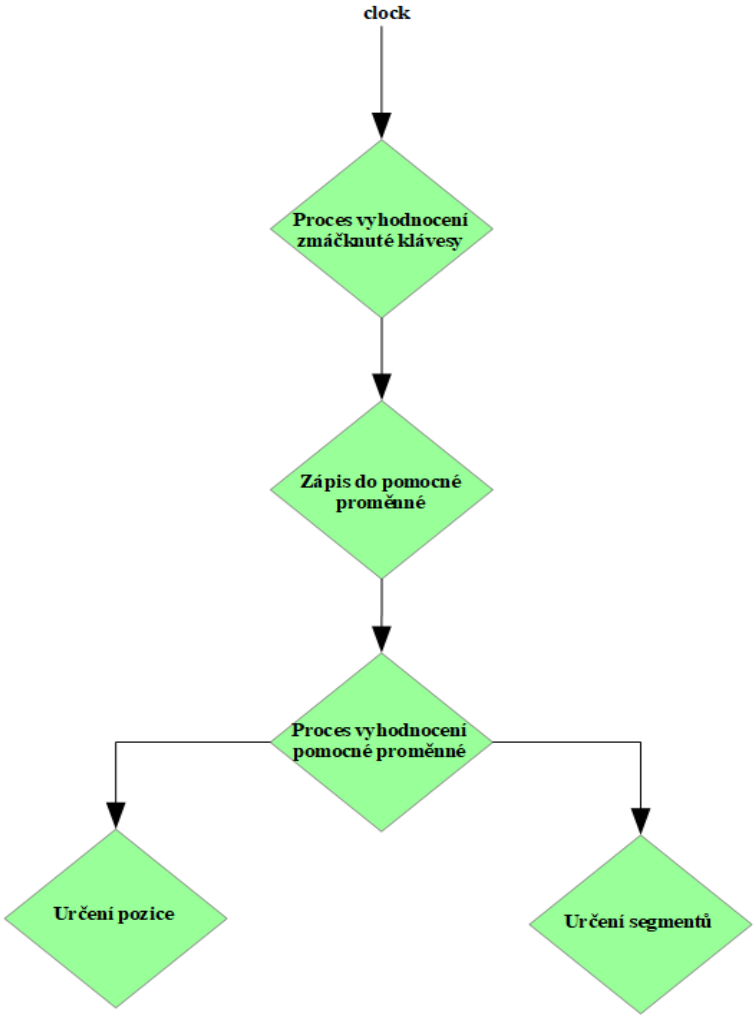


Obr. 3. 5: *Proces PWM modulace pro generování tónů [vlastní zpracování]*

Činnost generování tónů plyne z obrázku 3.5. Frekvence, kterou poskytuje přípravek Nexys4 je příliš velká, a proto je zapotřebí před procesem generování tónů frekvenci vydělit. K dělení frekvence slouží blok *Dělička kmitočtu*. Následuje blok *Čítač*, který každou svoji hodnotu porovnává v bloku *Komparátor1* s hodnotou v bloku *Střída*, jak si lze všimnout, tak tento blok je na obrázku obsažen dvakrát, pro lepší přehlednost jsou odděleny pouze barevně, a ne další podmínkou. V jednom čase se porovnává vždy jen jeden z těchto bloků, čehož je dosaženo vhodnými podmínkami v programu. Hodnota plnění se změní pouze při každém 255-tém cyklu. V tomto cyklu se také vyresetuje i *Čítač*. To znamená, že při každém vydělení se následně porovnává hodnota čítače s odlišnou hodnotou plnění, a tím se dosahuje proměnné hodnoty střídy. K dosažení efektu, který je na obrázku 3.5, je nutné nejen měnit hodnotu plnění (konstantně odčítat nebo přičítat), ale je také nutné vhodně navázat šířku pulzů, aby nedocházelo k razantním skokům, například z naplnění 90 % na naplnění 10 %. K tomu je nutné zavést další komparátor (*Komparátor2*), který ovládá podmínku pro změnu plnění. Tato podmínka je vhodně nastavena tak, aby na sebe plnění plynule navazovalo.

3.5 Proces segmentový displej

Segmentový displej má sloužit pro zobrazení právě zmáčknuté klávesy na maticové klávesnici, slouží tedy jako ověření, zda klávesnice reaguje správně. Zahrnuto je pouze zobrazení jednoho znaku na jednom místě daného displeje. Není to způsobeno velkou obtížností řešení, ale spíše nenalezením uplatnění pro zobrazení více číslic, či písmen najednou v této práci.



Obr. 3. 6: *Procesní diagram pro segmentový displej [vlastní zpracování]*

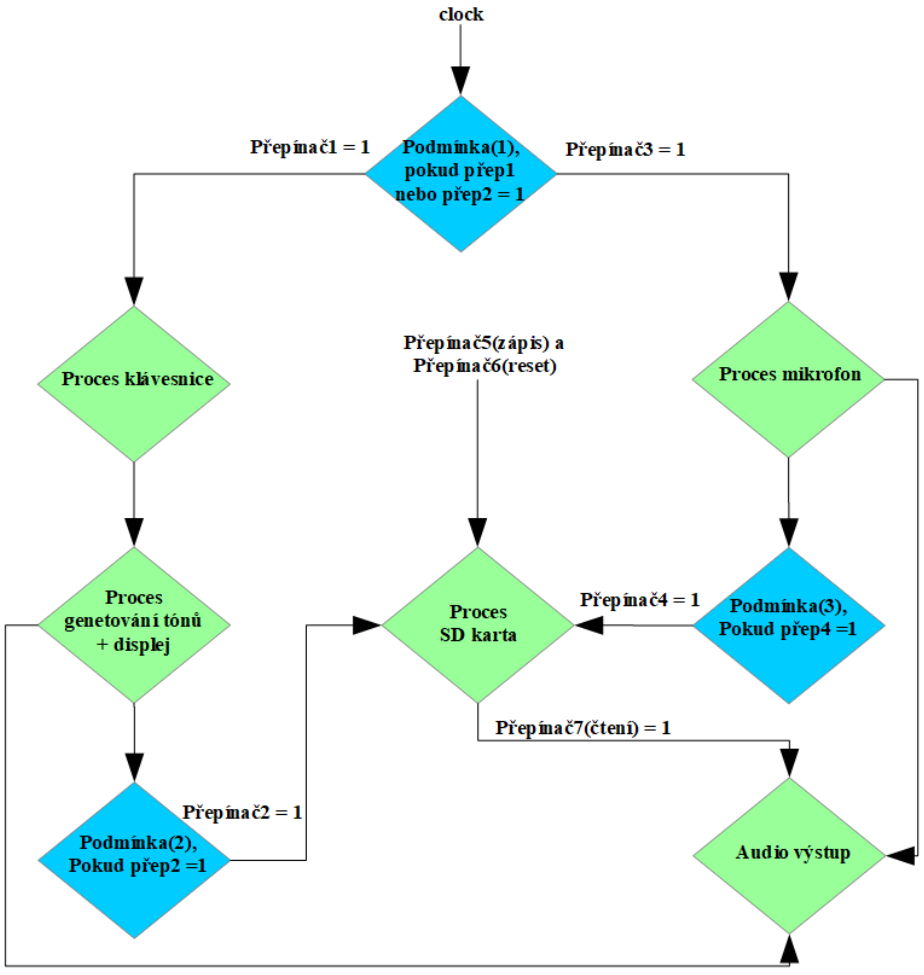
Na obrázku 3.6 můžeme vidět celý proces zápisu na segmentový displej. *Proces vyhodnocení zmáčknuté klávesy* funguje s frekvencí 33,3 kHz. Jakmile se zmáčkne určitá klávesa, dojde k zápisu do pomocné proměnné, tato proměnná uchovává hodnotu pro danou stisknutou klávesu po celou dobu stisku. Pomocná proměnná může být například čtyřbitový pomocný signálový vektor, kde každé kombinaci logických nul a jedniček přiřadíme jednu klávesu (*Proces vyhodnocení pomocné proměnné*). Těchto kombinací je šestnáct, stejně jako počet tlačítek na klávesnici. Po zapsání hodnot do pomocné proměnné následuje rozsvícení daného předem určeného místa a jemu odpovídající hodnotě z klávesnice.

3.6 Pohled na projekt jako celek procesů

Celý program ve VHDL funguje dle obrázku 3.7. Na začátku je uvedena podmínka, která určuje, zda budeme pracovat s klávesnicí a generováním tónů, nebo mikrofonem. Parametry podmínek se určují pomocí spínačů.

V případě, že podmínka projde do procesu klávesnice, dojde k automatickému generování tónů a zobrazení na segmentovém displeji, dle stisknuté klávesy. Nicméně další zápis je podmíněn druhou podmínkou, která určuje, zda chceme generované tóny rovnou přehrávat z reproduktoru, nebo nejdříve uložit a až poté přehrát.

Jestliže podmínka (1) vyhodnotí požadavek na práci s mikrofonem, dojde ihned k zpracování procesu, který se požadavku týká. Je ovšem nutné pomocí spínačů nastavit, zda chceme nahraný zvuk ukládat na SD kartu nebo rovnou přehrávat z audio výstupu.



Obr. 3. 7: *Přehled procesů jako fungující celek [vlastní zpracování]*

3.7 Testování a praktické výsledky

Při sestavování této práce se vycházelo především z teoretických předpokladů, jak by měly různé věci fungovat a pracovat. Pokud by něco nešlo dle předpokladů, byl v záloze vhodný logický analyzátor a osciloskop pro odhalení pravého dění věcí.

Testování probíhalo ve dvou fázích. První fáze bylo testování jednotlivých signálu a stavů daných portu pomocí takzvaného *testbench*, což je prostředí pro simulaci programu. Druhá fáze byla praktická zkouška, kde se testovala správná funkčnost programu.

Praktické výsledky a průběh řešení:

- Řešení pro maticovou klávesnici bylo uděláno víceméně experimentálně, byl znám princip čtení i zápisu, ale nedařilo se dlouhou dobu určit přesné časy, ve kterých se mají tyto dva děje odehrávat. Vše navíc ztěžovaly mechanické zákmity, které jsou bohužel u takovéto klávesnice běžné. Po několika pokusech s časováním se dospělo k funkčnímu řešení a klávesnice je plně funkční ve smyslu stisku jednoho tlačítka. Stisknutí více tlačítek najednou není nijak ošetřeno, nebylo to nutné pro správnou funkci generování tónů.
- Při implementaci SD karty se podařilo téměř vše, karta funguje, proběhne inicializace, zápis i čtení. Nicméně nastal problém, který se nepodařilo vyřešit, a tím je přímý zápis dat z mikrofonu na SD kartu. Data se zapisují, ale nejspíše bude problém se vzájemným sladěním časů, ve kterých probíhá čtení dat z mikrofonu a zápis na SD kartu. To má za následek pouze audio šum, při pokusu o čtení dat z SD karty a jejich přivedením na vstup audio. Myslím, že karta je plně funkční pro přesun souborů, které nevyžadují perfektní sladění v reálném čase, jako je čtení dat z mikrofonu ve formátu PDM. SD karta tedy funguje, ale je problém se šumem, který nastává po přehrání nahraných dat.
- Zbytek realizací probíhal bez větších problémů.

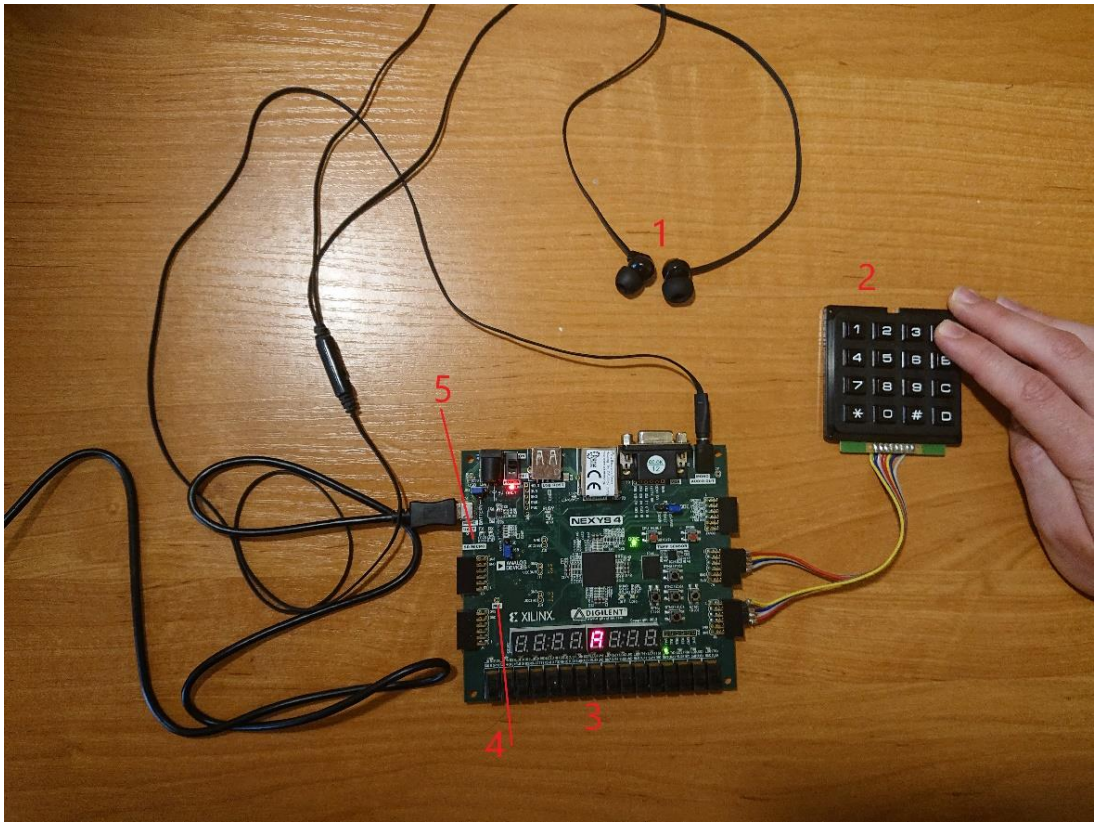
Na obrázku 3.7 můžeme vidět celý projekt jako celek, jsou zde vyznačené přepínače, které přímo odpovídají přepínačům na přípravku Nexys4, což ukazuje následující tabulka 3.1.

Číslo přepínače na obrázku 3.7	Přepínač na kitu
1	P4
2	P3
3	R3
4	T1
5	T3
6	U2
7	V2

Tab. 3. 1: *Tabulka přepínačů potřebných pro provoz programu na kitu Nexys4*

Obrázek 3.8 demonstruje funkční projekt k němuž jsou připojeny všechny části, kterou jsou popsány v této práci. Daná červená čísla na obrázku 3.8 ukazují, kde přesně se nacházejí dané části.

1. Reproduktor
2. Maticová klávesnice
3. Přepínače
4. Mikrofon
5. SD karta



Obr. 3. 8*: Zapojený přípravek Nexys4 s vyznačenými částmi [vlastní zpracování]*

Spouštění:

Po nahrání programu do přípravku Nexys4 jsou dvě možnosti, buďto aktivovat spínač1 a umožnit tím spuštění procesu maticové klávesnice, nebo aktivovat spínač2 a aktivovat tak proces pro mikrofon. Ať už je aktivována první či druhá možnost, obě tyto cesty vedou k automatickému přenosu audio signálu na audio výstup. Pokud budeme chtít nahrát tento audio signál na SD kartu, je nutné nejdříve aktivovat spínač6 (reset), který umožní inicializaci SD karty v režimu SPI. Karta se dostane do stavu, kde čeká, zda se bude zapisovat či číst. V této fázi se musí aktivovat jeden ze dvou spínačů (přepínač2 nebo přepínač4), aby bylo dovoleno zapisovat data z generování audio signálu pomocí maticové klávesnice, či data generována z mikrofonu. Po aktivaci jednoho z těchto přepínačů můžeme zapnout režim zápis na SD kartě pomocí přepínače5. Jelikož je zápis řešen poněkud zjednodušeně, jak je popsáno v kapitole 3.3.3, nedoporučuji zapisovat déle než 10 sekund. Po uplynutí této doby je nutné rozpojit přepínač5, a tím ukončit režim čtení. Karta se vrátí do stavu *idle* a je nutné opět kartu resetovat, ne z důvodu nějaké chyby, ale z důvodu nastavení adresy na počáteční, aby bylo možné data přečíst. Při čtení je vhodné vypnout přepínač, ať už pro proces maticové klávesnice, či pro proces mikrofonu, aby nedocházelo ke směšování dat na audio výstupu a přehrávala se opravdu jen data nahrána na SD kartě.

4 Závěr

Výsledkem bakalářské práce je návrh obvodu v jazyce VHDL určený pro přípravek Digilent Nexys4. Obvod je schopný vyhodnocování kláves z maticové klávesnice a následného generování tónů. Zahrnuto je i zobrazení na segmentovém displeji, mikrofon a SD karta.

Práce se v teoretické části zabývá důležitými částmi, které jsou nutné pro pochopení fungování přípravku Nexys4. Prvotní úsek této části je věnován stručnému seznámení s obvody PLD. O těchto obvodech by se dalo napsat spousta dalších stran, nicméně pro účel práce toto bohatě postačí. Dále jsou řečeny nejdůležitější části přípravku, na kterém je vše zpracováno – Digilent Nexys4. Důležitá část je také popis jazyka VHDL, kde se věnuji jeho historii a popisu nejdůležitějších částí. Uveden je i příklad kódu pro lepší pochopení syntaxe a pojmů entita, či komponenta. Dále práce popisuje důležité součásti, které jsou potřebné k vytvoření programu. Tento popis zahrnuje základní principy, jako například princip modulace PDM, která je použita u mikrofonu, či princip modulace PWM, která je použita při generování tónů. Nedílnou součástí je i popis SD karty a klíčové pojmy, které jsou nutné pro implementaci, jako například SPI rozhraní, či příkazy nutné pro zápis nebo čtení. Všechny součásti, které jsou popsány v teoretické části, jsou zároveň i zahrnuty při řešení.

Práce se v praktické části zabývá vlastním řešením jednotlivých částí, které jsou rozděleny do procesů. Tyto procesy tvoří dohromady jeden funkční celek. Jednotlivé procesy jsou rozděleny do kapitol, k procesům vždy vytvořen potřebný obrázek či diagram a následný popis řešení.

Na závěr praktické části se práce věnuje pohledu na projekt jako na celek procesů, jsou zobrazeny možnosti, které program obsahuje. Ošetřeny jsou pomocí vhodných podmínek, či spínačů, které jsou také vysvětleny.

Navržené řešení by šlo jistě dále poupravit a vylepšit. Například rozdělení procesů do komponent pro větší přehlednost a možnost následného používání v jiných programech. Za vylepšení by určitě stálo vyladění časování u SD karty, aby mohla být plně využita pro potenciál audio zvuku.

Tato práce by mohla směrovat až k činnosti podobné audio syntetizéru, či modulaci hlasu, což se dnes v praxi vyskytuje běžně. Nicméně pro použití do praxe by bylo nejspíše lepší rozdělit práci dle procesů, tedy například generování tónů pro poplašná zařízení či indikace chyby. Proces pro SD kartu by mohl sloužit například u dlouhodobého zapisování dat ze senzorů, které bude potřeba po určité době vyhodnotit. Maticová klávesnice by se dala využít například pro snímání hesla potřebného pro přístup do nějakého systému, či místnosti, nicméně se tento typ klávesnice moc nevyužije.

5 Seznam použitých zdrojů

[1] Nexys 4. *DIGILENT* [online]. Pullman, 2016 [cit. 2018-04-18]. Dostupné z: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4/nexys4_rm.pdf

[2] DOUŠA, Jiří. *Jazyk VHDL*. Praha: Vydavatelství ČVUT, 2003. ISBN 80-010-2670-1.

[3] *Základní seznámení s jazykem VHDL: výukový materiál pro předmět A2B99DIT*. Katedra telekomunikační techniky, FEL ČVUT v Praze, 2010.

[4] KOŘENEK, Jan. *Číslicové obvody a jazyk VHDL: Návrh počítačových systémů*. FIT VUT Brno, 2008.

[5] Úplné základy a nezbytná teorie. *Vhdl.cz* [online]. 2015 [cit. 2018-04-19]. Dostupné z: https://vhdl.cz/uplne-zaklady-a-nezbytna-teorie/

[6] PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL*. Praha: BEN - technická literatura, 2006. ISBN 80-730-0198-5.

[7] Firemní literatura firmy Analog Devices [online]. 11/2011. ADMP421. Dostupné z: http://www.analog.com/media/en/technical-documentation/obsolete-data-sheets/ADMP421.pdf

[8] Keypads. *Hobbycomponents* [online]. [cit. 2018-04-20]. Dostupné z: http://hobbycomponents.com/keypads/181-4-x-4-matrix-keypad

[9] Pavel LAFATA, Petr HAMPL a Michal PRAVDA. *Digitální technika*. V Praze: České vysoké učení technické, 2011. ISBN 978-800-1049-143.

[10] Secure_Digital. *Wikipedia[EN]* [online]. [cit. 2018-05-07]. Dostupné z: https://en.wikipedia.org/wiki/Secure_Digital

[11] *Elm-chan* [online]. [cit. 2018-05-07]. Dostupné z: http://elm-chan.org/docs/mmc/mmc_e.html

[12] KRÁL, Jiří. *Řešené příklady ve VHDL: hradlová pole FPGA pro začátečníky*. Praha: BEN - technická literatura, 2010. ISBN 978-80-7300-257-2.

6 Seznam obrázků

Obr. 2. 1: Přípravek Nexys4 [1] 12
Obr. 2. 2: obrázek pro příklad kódu v jazyce VHDL 14
Obr. 2. 3: Schéma mikrofonu MEMS ADMP421, a jeho vývody pro použití na FPGA. [vlastní zpracování dle: 7] 15
Obr. 2. 4: Maticová klávesnice [8] 16
Obr. 2. 5: Zjednodušené schéma dolní propusti [vlastní zpracování dle 1] 16
Obr. 2. 7: Ukázka PDM signálu [1] 17
Obr. 2. 8: Delta – Sigma modulátor [1] 17
Obr. 2. 9: Namodelovaný PWM signál a jemu odpovídající analogový signál po průchodu DP [1] 18
Obr. 2. 10: Komunikace mezi přípravkem Nexys4 a SD kartou v SPI módu[11] 20
Obr. 2. 11: Porty SD micro karty propojené s přípravkem Nexys4 (Artix-7)[1, strana 22] 20
Obr. 2. 12: Příklad časování poslání příkazu a přijaté odpovědi [11] 20
Obr. 2. 13: Odpověď typu R1 [vlastní zpracování dle: 11] 21
Obr. 2. 14: Průběh zápisu na SD kartu v SPI módu [vlastní zpracování dle: 11] 22
Obr. 2. 15: Průběh čtení z SD karty v SPI módu [vlastní zpracování dle: 11] 22
Obr. 2. 16: Zapojení diod v segmentovém displeji [1, strana 20] 23
Obr. 2. 17: Průběh aktivace jednotlivých segmentů displeje [vlastní zpracování] 23

Obr. 3. 1: Zapojení maticové klávesnice do přípravku Nexys4 [vlastní zpracování] 24
Obr. 3. 2: Průběh zápisu a čtení maticové klávesnice [vlastní zpracování] 25
Obr. 3. 3: Procesní diagram pro mikrofon [vlastní zpracování] 26
Obr. 3. 4: Stavový diagram pro SD kartu pracující v SPI módu. Inicializace, čtení a zápis. [vlastní zpracování] 28
Obr. 3. 5: Proces PWM modulace pro generování tónů [vlastní zpracování] 29
Obr. 3. 6: Procesní diagram pro segmentový displej [vlastní zpracování] 30
Obr. 3. 7: Přehled procesů jako fungující celek [vlastní zpracování] 31
Obr. 3. 8: Zapojený přípravek Nexys4 s vyznačenými částmi [vlastní zpracování] 33

7 Seznam použitých zkratek

SD	*Secure Digital*
MEMS	*Microelectromechanical Systems*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integration Circuits*
PWM	*Pulse Width Modulation*
PDM	*Pulse Density Modulation*
FPGA	*Field Programmable Gate Array*
PLD	*Programmable Logic Device*
LUT	*LookUp Tables*
Vdd	*Power supply pin*
GND	*Ground*
SDSC	*Secure Digital Standard Capacity*
SDHC	*Secure Digital High Capacity*
SDXC	*Secure Digital eXtended Capacity*
SDIO	*Secure Digital Input Output*
CRC	*Cyclic Redundacy Check*
MSB	*Most Significant Bit*
LSB	*Less Significant Bit*
CS	*Chip Select*
SS	*Slave Select*
MOSI	*Master Output Slave Input*
MISO	*Master Input Slave Output*
Ncr	*Command Response Time*
LED	*Light-Emitting Diode*
ČDSK	*Čas Doby Stisknuté Klávesy*
SPI	*Serial Peripheral Interface*

8 Přílohy

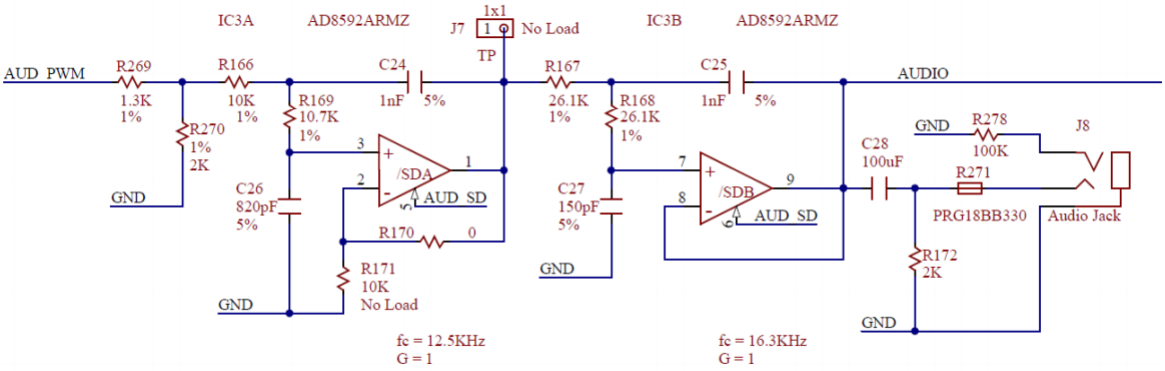
Seznam příloh:

Příloha 1: Zapojení dolní propusti čtvrtého řádu

Obsah přiloženého CD:

Andera_Jiri.pdf – bakalářská práce

bakalarka1.rar – archiv s projektem pro Xilinx ISE 14.7



Příloha 1 – Zapojení dolní propusti čtvrtého řádu