Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Strinni Bence

PROFESSZIONÁLIS TÖBBCSATORNÁS HANGRÖGZÍTÉS SD KÁRTYÁRA

TDK 2018

KONZULENS

Dr. Fehér Béla

BUDAPEST, 2018

Tartalomjegyzék

Rövidítések jegyzéke

ADC	analog-to-digital converter
BCLK	bit clock
CODEC	coder / decoder
DAC	digital-to-analog converter
DMA	direct memory access
FIFO	first in first out
FPGA	field-programmable gate array
GPIO	general-purpose input / output
GUI	graphical user interface
I2S	inter IC sound
IP	intellectual property
LPDDR	low power dual data rate
LRCLK	left-right clock
MCLK	master clock
MMCM	mixed-mode clock manager
MPSoC	multiprocessor system-on-(a)-chip
PCM	pulse-code modulation
PL	programmable logic
PLL	phase locked loop
PS	processing system
SDIO	Secure Digital Input Output
SoC	system-on-(a)-chip

Összefoglaló

A TDK munkám célja egy professzionális hangrögzítő eszköz tervezése. A legfontosabb specifikációk közé tartozik a kis méret, egyszerű kezelhetőség és hogy egyidejűleg tudjon 2 db sztereó csatornát rögzíteni 24 bit / 96 kHz bitmélységgel illetve mintavételi frekvenciával.

Az elsődleges felhasználási területe koncertek rögzítése lehet, dj-k illetve zenészek számára. Az egyik pár sztereó csatornán keresztül rögzíteni lehet a dj illetve zenész munkáját, a keverőpult kimenete felől, amíg a másik bemeneten pedig a közönség hangját lehet felvenni. Az így elkészült felvétellel visszaadható a koncertélmény. A csatornákat külön rögzíti az eszköz, így azok jelszintjét később egymáshoz képest be lehet állítani.

Természetesen lehetséges más módon rögzíteni koncerteket, például egy laptop és egy több csatornás hangkártya segítségével. Ez azonban egy kisebb zenekar vagy dj számára ahol nincs külön technikus sokszor bonyolult. Léteznek hasonló eszközök, azonban ezek általában nagyon drágák és nem annyira kompakt kivitelűek.

A TDK során az eszköz prototípusát szeretném kifejleszteni, egy Avnet Ultra96 kártyán, némi külső áramkörrel kiegészítve.

Az eszköz központi eleme egy Xilinx Zynq UltraScale+ SoC, amely többek között tartalmaz egy többmagos ARM mikrokontrollert és FPGA logikát is. Az analóg – digitális konverziót jó minőségű, az iparban is használt ADC-k oldják meg, amelyek I2S interfésszel rendelkeznek. A mikrokontroller és az ADC-k között egy az FPGA részen megvalósított modul teremti meg a kapcsolatot.

A hang rögzítése SD kártyára történik, amelynek kezeléséhez a Zynq-ben található SDIO modul nyújt segítséget. A kezelőfelület egy kijelzőből, néhány gombból és LED-ből áll.

Esetleges későbbi fejlesztés lehet, valamilyen jelfeldolgozás megvalósítása az FPGA-n belül, például hangszín szabályzás vagy effektek (például zengetés).

A prototípus kifejlesztése után célom egy saját áramkör megtervezése, amelyen minden szükséges alkatrész megtalálható a specifikációban leírtakhoz.

Abstract

The aim of my TDK work is to design a professional audio recorder device. The main specifications are the following: small size, to be easy to use and to be able to record 2 stereo channels in 24 bits / 96 kHz bit depth and sampling frequency in the same time.

The main application of the device is to record live concert for DJs and musicians. One of the stereo channels can be used for recording the main audio from the mixer of the DJ or the musician, while the other stereo channel can record the sound of the audience. Recordings made this way can give a good concert experience. The device records the channels separately, so the levels can be adjusted during post processing.

Of course there are other ways to record a concert, for example with the help of a laptop and a multi-channel soundcard, but this can be difficult for a band or DJ who don't have a technician with them at the concerts. There are similar devices, but they have high price and they are usually not compact.

In my TDK work I would like to design the prototype of the device, using an Avnet Ultra96 board and some external components.

The central unit of the device is a Xilinx Zynq UltraScale+ SoC, it contains a multicore ARM microcontroller and FPGA logics, among others. The analogue to digital conversion is provided by high quality ADCs with I2S interface. The connection between the ARM and the ADCs is realized by a module inside the FPGA.

The sound is recorded to an SD card, which is handled by the SDIO peripheral inside the Zynq. The user interface contains a display, some buttons and some LED-s.

There are possible improvements in the future thanks to the FPGA. For example audio processing, like equalization or reverb.

After finishing the prototype I would like to design my own circuit, which contains all the needed components for the specification.

1 Bevezetés

A TDK dolgozatom témája egy többcsatornás hangrögzítő eszköz tervezése, megvalósítása és tesztelése volt. Az eszköz specifikációi röviden: képes legyen 4 darab (tehát 2 sztereó) csatorna egy idejű rögzítésére, 24 bit bitmélységgel és 96 kHz mintavételi frekvenciával. A rögzített hangot SD kártyára mentse. Rendelkezzen egy egyszerű felhasználói felülettel, amin keresztül el lehet indítani illetve meg lehet állítani a felvételt, valamint információt szolgáltasson a felvétel állapotáról.

A feladat elég összetett, magában foglalt nyomtatottáramkör-tervezést, FPGA-s hardvertervezést és szoftverfejlesztést is. A fejlesztéshez felhasználtam egy Avnet Ultra96 fejlesztőkártyát. A kártyán egy Xilinx Zynq UltraScale+ MPSoC található, amely többek között tartalmaz egy többmagos ARM mikrokontrollert és FPGA logikát is. Arról, hogy miért esett erre a választás a 3.2 fejezetben lesz szó. A szoftver értelemszerűen az ARM processzoron fut, az FPGA logikában pedig gyári és saját tervezésű perifériák is találhatók. A kártyán rengeteg periféria mellett megtalálható egy SD kártya foglalat is, amelynek segítségével a rögzített hangot lehetséges SD kártyára menteni.

A fejlesztőkártya rendelkezik egy csatlakozósorral, amelyre a saját tervezésű nyomtatott áramkör kapcsolódik. Az áramkörön található egy grafikus kijelző, nyomógombok LED-ek valamint a két sztereó CODEC áramkör csatlakozója. A CODEC áramköre nem saját fejlesztésű, hanem az egyetemi LOGSYS rendszer egyik eleme [1]. Az áramkör analóg-digitális és digitális-analóg átalakításra is képes, 1-1 sztereó jack csatlakozót tartalmaz az analóg jelek számára.

A TDK során megtervezett eszköz egy prototípus. Ezért nem készült hozzá teljesen saját áramkör. Később szeretném megvalósítani az eszközt egy önálló áramkörön, fejlesztőkártya felhasználása nélkül.

A nyomtatottáramkör-tervezéshez az Autodesk EAGLE tervező program 6.3.0-s verzióját használtam fel. Az FPGA hardver tervezést a Xilinx Vivado 2018.2-vel végeztem, a szoftverfejlesztést pedig a Xilinx SDK 2018.2-vel.

Fejlesztés során egy Xilinx Platform Cable-t használtam az eszköz programozására, ez egy JTAG – USB kábel. Miután elkészült az FPGA-s hardver és a szoftver, ezekből egy boot fájl készítettem és az SD kártyára másoltam, így az eszköz már önállón is képes működni.

2 Irodalomkutatás

2.1 Zynq [1]

A Xilinx Zynq termékcsaládja egy (MP)SoC a (Multi-Processor) System-on-(a)-Chip rövidítése, amely magyarul így hangzik: (több processzoros) rendszer egy chipen. A kifejezés arra utal, hogy egyetlen chipen (több) processzor (akár eltérő processzorok, akár több magosak) és programozható logika is található. Innentől az MPSoC helyett az SoC rövidítést fogom használni, mivel a projektem szempontjából nem lényeges a több processzor létezése.

A Xilinx termékcsaládoknak létezik UltraScale+ változatuk, ezek az adott termékcsalád legerősebb tagjai. A felhasznált Ultra96 kártyán egy ilyen típusú Zynq található.

A Zynq SoC alapvetően két részre osztható: PS és PL. A PS jelentése: processing system, itt található a processzoros rendszer. Ez magában foglalja a 4 magos , 64 bites Cortex-A53 CPU-t, a két magos valós idejű Cortex-R5 processzort, egy GPU-t, egy videó feldolgozó egységet, egy rádiós egységet és a kiegészítő perifériákat. A kiegészítő perifériák között a hagyományos mikrokontroller perifériák találhatók meg: GPIO, SPI, I2C, SDIO, CAN, USB, stb.

A PL jelentése: programmable logic, ez jelenti az FPGA területet. Ez egy hagyományos FPGA-nak felel meg, megtalálhatók itt a szokásos elemek: rengetek logikai cella, flip-flop, block RAM, DSP Slice, stb.

A két terület között nagysebességű és nagy adatszélességű AXI buszok teremtenek kapcsolatot, valamint megszakítást is lehet küldeni mind a két irányba.

2.2 ADC, DAC

Az ADC (analog-to-digital converter) feladata, hogy az analóg jeleket digitálissá alakítsa. Az ADC bemenete egy analóg jel, például feszültség, a kimenete pedig egy bitsorozat. Az **I. mintavételi tétel** szerint, ha egy jel sávkorlátozott, azaz

$$X(f) = 0, ha\ |f| \geq B$$

akkor

$$f_s \geq 2B$$

esetén a folytonos időfüggvény hibátlanul visszaállítható (azaz a mintavételezéssel nem veszítünk információt). [2] Ez tehát nem azt jelenti, hogy a digitális bitsorozat megfelel az analóg jelnek, de abból elméleti szinten információvesztés nélkül vissza tudjuk állítani az eredeti analóg jelet. A valóságban persze már a mintavételezésnél is történik információvesztés. Például a jelek sosem tökéletesen sávkorlátozottak, ezen kívül kvantálási zaj is jelentkezik, stb.

DAC (digital-to-analog converter) feladata, hogy a digitális jelet visszaalakítsa analóggá. (Bitsorozatot például feszültséggé.) Sajnos ahhoz, hogy a digitális jelet hiba nélkül vissza tudjuk állítani szükség lenne egy ideális aluláteresztő szűrőre. Ez pedig egy nem kauzális rendszer, így a valóságban nem építhető ilyen. Egyéb hatások is fellépnek az átalakításkor, például kvantálási zaj. Természetesen attól még, hogy a digitális-analóg és az analóg-digitális átalakítás nem tökéletes lehet használni, de számolni kell a zajjal, torzítással.

Az ADC-k és DAC-k egyik legfontosabb tulajdonsága a bitmélység, tehát az, hogy egy digitális mintát hány biten tárol, és a mintavételi frekvencia, ami azt jelenti, hogy másodpercenkét hányszor vesz mintát az analóg jelből.

2.2.1 Szigma-delta átalakító [3]

A típust szokták delta-szigma átalakítónak is nevezni, valamint van, hogy a két görög betűvel utalnak rá: ΔΣ.

Audió jelek mintavételezésekor leggyakrabban szigma-delta átalakítót használnak. Ennek előnye a magas bitszám (akár 24-32 bit). Hátránya, hogy nem lehet nagyon magas mintavételi frekvenciát elérni vele (GHz), viszont audió jeleknél ez nem is szükséges.

Működése röviden: a bemenő jelet nagyfrekvencián mintavételezi, kevés bitszámmal (akár egyetlen biten). Ha a túlmintavételezett jelet decimáljuk, akkor azzal megnöveljük az effektív bitszámot. Ha egyetlen biten mintavételezünk, akkor az az egy bit biztosan lineáris lesz. A túlmintavételezés miatt nem lehet elérni nagy mintavételi frekvenciát.

2.3 Digitális audió [4]

Miután az analóg jeleket átalakítottuk digitálisra valamilyen formátumban tárolni kell őket. Az ADC-k kimenetéről pulzus kód modulációval érkeznek a jelek (anolul: pulse-code modulation, azaz PCM). Ez a kódolás azt jelenti, hogy minden kvantálási szinthez rendelünk egy digitális kódot. Majd mintavételezéskor az adott kvantálási szinthez tartozó kódot továbbítjuk. Ha a kvantálási közök egyenletesek (ahogy a legtöbb alkalmazásban azok), akkor

az így keletkező kódot LPCM-nek nevezzük (linear PCM, lineáris PCM). Gyakorlatban legtöbbször az LPCM-et egyszerűen PCM-nek hívják.

Ez a digitális kód lehet kettes komplemens ábrázolású vagy ofszet kód. A PCM előnye az egyszerű kezelhetőség, hátránya a nagy méret.

A PCM-et egyszerűen lehet tömöríteni: ha nem tároljuk el minden mintához az abszolút nagyságát, hanem csak azt, hogy mekkora az eltérése az előzőhöz képest, akkor ezzel a szükséges bitszám várhatóan csökken. Elképzelhetők olyan jelek, ahol ez nem jelent csökkenést, de általában az audió jelek két minta között csak keveset változnak. Ezt az eljárást DPCM-nek, azaz differenciális PCM-nek nevezik. Körülbelül 25%-os csökkenést jelnet az eredeti PCM-hez képest és teljesen veszteségmentes.

Léteznek veszteségmentes tömörítések is, az egyik legelterjedtebb az MP3.

2.3.1 WAV [5]

A WAV egy veszteségmentes audió formátum, LPCM kódolást használ. Gyakran használják veszteségmentes adatok hang adatok tárolására professzionális környezetben is. A fájlok egy fejléc és egy adatrészből állnak. A fejléc tartalmazza a mintavételi frekvenciát, a bitmélységet, a csatornák számát az adat blokk méretét, a teljes fájl méretét és néhány egyéb információt.

A mezők nagy része little endian formátumú, kivéve néhány szöveges mezőt (amelyben a ”RIFF”, ”WAVE” és az ”fmt ” sztringek találhatók).

Az adat blokk is little endian formátumú, egymás után tartalmazza a bal és a jobb csatorna mintáit felváltva, sztereó fájl estén. 24 bites bitmélységnél például egymást követő 3-3 bájtok jelentik a bal és a jobb csatorna mintáit (a bal csatornával kezdődik). Létezik többcsatornás WAV fájl is, ekkor a csatornák mintái szintén egymást követik. Például 4 csatornás fájl esetén, 24 bites mintáknál egymást követi 4 darab 3 bájtos szó. Így 12 bájt jelent egyetlen mintavételi periódust.

A fejlécben két mezőt utólag kell kitölteni: az adat blokk hosszát és teljes fájl méretét. Az összes többi előre kitölthető, mivel ezek előre ismertek.

2.4 I2S

Az I2S (vagy eredetileg I^2S) az Inter IC Sound rövidítése, a Philips által fejlesztett szabvány. Egy soros kommunikációs interfészt ír le, hangátvitel számára. Sok A/D és D/A konverter rendelkezik ezzel az interfésszel.

A busz 3 vezetékből áll: két órajel (SCK, WS) és egy adatvezetékből (SD). Ezeken kívül lehetséges még a mester órajel (MCK) átvitele is. Az órajelek forrása lehet az A/D vagy D/A konverter és a feldolgozó eszköz is. Az adatvezeték egy irányú ADC esetén forrása a CODEC chip, DAC esetén pedig a feldolgozó eszköz.

A SCK a soros adatokhoz tartozó órajel (ezért szokták BCLK-nak is hívni, a bit clock rövidítéséből). A WS (a Word Select rövidítése) megadja, hogy az adatvezetéken melyik csatorna jele található éppen (ezért szokták LRCLK-nak is hívni). A 0 jelenti a bal csatornát, az 1 pedig a jobbat. Ennek az órajelnek egy periódusa alatt a két csatornáról 1-1 mintát viszünk át, így ez az órajel megegyezik a mintavételi frekvenciával.

A dolgozat későbbi fejezeteiben a zárójelben megadott kifejezéseket fogom használni, tehát a BCLK-t és az LRCLK-t, a mester órajelet (MCK) pedig MCLK-nak fogom hívni.

Az adatvezetéken 16, 24 vagy 32 bites adatokat vihetünk át sorosan; a lehetséges keretméret 16 vagy 32 bit. Értelemszerűen 16 bites keretben csak 16 bites mintát vihetünk át. Ha 32 bites kereten viszünk át 16 vagy 24 bites mintát, akkor a keretből 16 vagy 8 bit nem definiált. Az adatok különböző módon helyezkedhetnek el a kereten belül: balra illesztett, jobbra illesztett. Az előbbinél a minta MSB-je rögtön a WS élváltását követően érkezik, az utóbbinál pedig az LSB után következik be a WS élváltás. (Mind a két esetben az MSB érkezik először.) A hagyományos I2S kommunikáció az balra illesztett átvitelhez hasonló, csak itt egy órajel késleltetéssel érkezik az adat a WS élváltás után.

A mester órajel általában a bit órajel 256-szorosa, de ez változhat egyes CODEC-eknél. Van amelyik több arányt is megenged.

3 Rendszerterv

3.1 Felhasznált eszközök bemutatása

3.1.1 Ultra96 kártya [7]

A fejlesztéshez a már említett Avnet Ultra96 kártyát választottam, mivel ez tartalmaz egy Zynq SoC-t és SD kártya foglalatot is, valamint egy bővítő csatlakozót, ahova a saját áramkört tudtam illeszteni. A Zynq pontos típusa: Zynq UltraScale+ MPSoC ZU3EG SBVA484.

Arról, hogy miért esett a választás egy Zynq SoC-re a 3.2 fejezetben hamarosan szó lesz. Ezeken a perifériákon kívül még rengeteg más eszközt is tartalmaz a fejlesztő kártya (WiFi, Bluetooth, Display port, stb.), ezeket azonban a projektben nem használtam.

Az Ultra96 kártyára egy 40 érintkezős csatlakozósoron keresztül illeszkedik az áramkör. A csatlakozón keresztül elérhető 16 db adatvezeték a PL-ből, 14 db adatvezeték a PS-ből és 1-1 tápfeszültség (1,8V és 5V).

3.1.2 LCD

Az LCD típusa: Electronic Assebly DOGS102W-6. A kijelző monokróm, felbontása 102 x 64 pixel. A vezérlést egy UC1701 kontroller végzi.

A kijelző kezelése SPI buszon keresztül történik. Működtetése meglehetősen egyszerű, nem igényel bonyolult külső áramkört (kontraszt beállításhoz, különleges tápfeszültségek, stb.). Részben ezért esett erre a típusra a választás. A másik indok pedig, hogy korábban már használtam ilyen kijelzőt, így ismertem a működését.

3.1.3 LOGSYS sztereó CODEC modul

A LOGSYS sztereó CODEC modulon [1] egy Wolfson WM8569 [9] chip található néhány kiegészítő áramkör mellett (tápfeszültség előállítása, átlapolásgátló szűrő). A chipben egy-egy 2 csatornás (sztereó), 24 bites szigma-delta ADC és DAC található. Az előbbi maximum 96 kHz, az utóbbi pedig 192 kHz-es mintavételi frekvenciára képes. Tehát teljesíti a specifikációban elvárt értékeket. Az adatokat I2S interfészen keresztül továbbítja, illetve fogadja.

A CODEC-eknek 3 órajel szükséges: MCLK (rendszer órajel), LRCLK (aktív csatorna kiválasztása az I2S kommunikáció során, ez megegyezik a mintavételi frekvenciával) és BCLK (bit órajel: az I2S kommunikáció során az átvitt biteket jelzi). A LOGSYS modulon a chip úgy van hardveresen konfigurálva, hogy az órajeleket kívülről kapja.

3.2 Rendszerterv

A beágyazott rendszerek központi elemét valamilyen feldolgozó egység szokta alkotni. Ez többek között lehet processzor vagy FPGA. A rendszer tervezésénél a kettő között kellett választani. A fájlrendszer kezelés és a felhasználói felület a processzoros rendszer kialakítását indokolja, mivel ezek megvalósítása FPGA-val meglehetősen bonyolult. Azonban van néhány feladat, amely kifejezetten FPGA-s megoldást kívánt meg. Ezért esett a választás egy Zynq SoC-re, amely egyaránt tartalmaz FPGA logikát és egy beágyazott keménymagos processzort is.

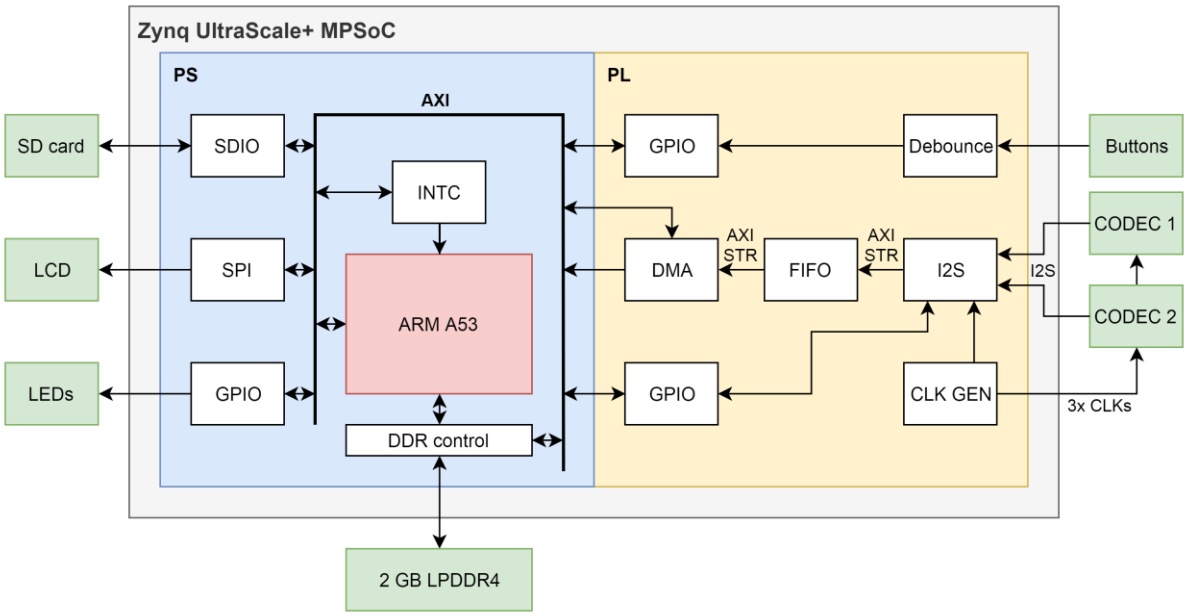
A processzor végzi az SD kártya kezelését (fájlrendszer kezelése és CODEC felől érkező hang mentése a kártyára), valamint a felhasználói felület vezérlését (gombok, LED-ek és kijelző kezelése). Az FPGA a CODEC-ek kezeléséért felelős: órajelek előállítása, I2S adatok feldolgozása és továbbítása a processzornak. Ezen kívül a nyomógombok pergésmentesítése is az FPGA-ban lett megvalósítva, így a szoftvernek ezzel már nem kell törődnie.

Látható tehát, hogy milyen előnyöket hordoz magában, ha egy rendszer központi elemének egy SoC-t használunk. A gombok, kapcsolók pergésmentesítése elég gyakori probléma, egy mikrokontrolleres rendszerben legtöbbször szoftveresen szokták megvalósítani. Ez azonban plusz program végrehajtási időt és plusz kódot is igényel. Ezzel szemben egy SoC-ben ezt a pergésmentesítést meg lehet valósítani hardveresen is, az FPGA-ban. Az FPGA terület későbbi fejlesztéseknél is felhasználható különböző jelfeldolgozási feladatokhoz. Valamint tetszőleges interfészt lehet illeszteni a mikrokontrollerhez, ha az nem rendelkezik a megfelelő perifériával. Például a Zynq-en belül nincs I2S periféria.

A felhasznált Zynq UltraScale+ MPSoC-ben található erőforrások bőven túlzóak a projekthez képest. Az FPGA területnek is csak egy kis része lett kihasználva, valamint a Zynq több processzormagjából is csak egyetlen működik. Ráadásul a Zynq-ben található rengetek periféria közül is csak néhány került felhasználásra. Lehetséges lett volna egy jóval kisebb méretű FPGA-n megvalósítani a projektet, egy beágyazott lágymagos processzorral, ekkor viszont az SD kártya kezelése lett volna bonyolultabb, mivel itt nem áll rendelkezésre a Zynq-

ben megtalálható SDIO periféria. Ez beilleszthető gyári IP-ként vagy saját tervezésű hardverként is. Előbbi esetben az IP-ért fizetni kell, utóbbi esetben viszont a fejlesztés hosszú ideig tarthat, mivel ez egy elég bonyolult periféria.

Vannak kisebb teljesítményű Zynq SoC-k is (nem UltraScale+ változat), ezek is megfelelőek lettek volna, azonban az Ultra96 kártyán ez a változat található meg.



1. ábra: Rendszerterv

Az 1. ábrán látható a rendszerterv. A Zynq SoC (szürke háttér) két részre van osztva: PS (kék háttér) illetve PL (sárga háttér). A PS tartalmazza a processzoros rendszert, és a szilíciumon előre kialakított perifériákat. A PS-en belül sokkal több periféria és további CPU magok is találhatóak, viszont ezek a rendszerben ki vannak kapcsolva, nem vesznek részt a hangrögzítésben.

A PL pedig az FPGA területet jelenti, ahova gyári és saját tervezésű modulok is kerültek. Zöld háttérrel a külső elemek láthatóak. A megszakítás vezérlőre (INTC) a DMA és a gombokhoz tartozó GPIO kapcsolódik, csak az átláthatóság kedvéért nem lettek ezek a vezetékek behúzva az ábrán. A perifériák a PL-en belül egyaránt 100 MHz-es órajelről működnek.

Az SD kártyát a PS-en belül található SDIO periféria vezérli. A specifikáció szerint 4 csatornát kell rögzíteni, ezeket egyenként 24 bites mélységgel és 96 kHz-es mintavételi frekvenciával. Ebből kiszámolható a szükséges adatátviteli sebesség:

$$\frac{4\ (ch) * 24\ (bit) * 96000\ (kHz)}{8 * 1024} = 1125\ kB/s$$

Ez még nem nagy írási sebesség, viszont mindenképpen jobb minőségű SD kártya szükséges hozzá, hogy a követelményt teljesíteni tudja. Erről a 6.1 fejezetben lesz még szó.

Az LCD kijelzőt szintén a PS-ben található SPI vezérlő kezeli. A LED-ek a PS-en belül található GPIO-ra kapcsolódnak, a gombok viszont a PL-en belülire. Ezt egyrészt az Ultra96 kártya bővítő csatlakozója határozta meg, mivel az ott található kivezetések egy része a PS-hez, egy másik része pedig a PL-hez kapcsolódnak. Másrészt így lehetőség volt a gombok és a GPIO közé egy pergésmentesítő modult tervezni. A pergésmentesítő saját tervezésű modul, Verilog nyelven implementálva.

A 2 GB-os LPDDR memória a fejlesztőkártyán található.

A CODEC-ek órajelének megválasztása a mintavételi frekvenciától és a bitmélységtől is függ. Az MCLK órajelet bizonyos kereteken belül szabadon megválaszthatjuk. A fenti paraméterekhez szükséges lehető legalacsonyabb órajelek a következőek:

- MCLK = 24,576 MHz = 256 * LRCLK
- BCLK = 6,144 MHz = 64 * LRCLK
- LRCLK = 96 kHz

A lehető legalacsonyabb órajelek indoka: az órajeleket az FPGA-n kívülre vezetjük, a csatlakozókon és a nyomtatott áramkörön keresztül a CODEC-ekhez, így érdemes minél kisebb frekvenciát használni. Nagyobb frekvencián a külső vezetékeket már nem lehetne egyszerű vezetéknek tekinteni, hanem távvezeték modellt kéne alkalmazni, így például a vezeték impedanciájával is számolni kéne. 24,5 MHz-en néhány cm-es vezetékeknél ez még szerencsére nem jelent problémát. (Még 15-ös számú felharmonikus esetében sem problémás.)

Az MCLK-t egy MMCM modul állítja elő egy PLL segítségével a 100 MHz-es PL rendszer órajelből. A PLL-re azért van szükség, mivel a 24,576 MHz nem egész számú osztója a 100 MHz-nek, így egyszerű számlálóval nem lehet előállítani azt. A BCLK és az LRCLK viszont már egész számú osztója az MCLK-nak, így azokat egy számláló állítja elő.

A CODEC felől érkező jeleket a saját tervezésű I2S modul dolgozza fel, alakítja át és küldi tovább egy FIFO-ba AXI-Stream buszon keresztül. A FIFO-ból a DMA másolja az adatokat DDR4 memóriába, ahonnan már a processzor eléri őket. Az I2S modul vezérlését egy GPIO periféria végzi. Illeszteni lehetett volna a periféria regisztereit is az AXI buszra,

viszont annyira egyszerű az egész vezérlése (csupán néhány vezeték), hogy kézenfekvőbb volt így megvalósítani.

A CODEC-eken keresztül meg lehet hallgatni a bemenő jeleket (csak valósidőben, tehát a korábban rögzített felvételeket az eszköz nem játssza le). Összesen két sztereó kimenete van az eszköznek (mind a két CODEC-en egy-egy). Mind a két kimeneten lehetőség van választani, hogy melyik bemenetet szeretnénk visszahallgatni, illetve a kettő keverékét is választhatjuk. Így az I2S modul 3 kimenettel is rendelkezik: a bejövő csatornák kivezetése illetve a két csatorna összegzése. Ezek nem szerepelnek az ábrán, a könnyebb áttekinthetőség érdekében. A CODEC-ek előtt 1-1 3 bemenetű multiplexerrel lehet választani a kimenetek közül, a multiplexert szintén az előbb ismertetett GPIO vezérli.

A megszakítás vezérlőbe a gombokhoz tartozó GPIO és a DMA vannak bekötve, így ezek képesek megszakítást küldeni a processzor számára. A GIPO periféria egy-egy gomb lenyomása (illetve felengedése) esetén küld megszakítást, a DMA vezérlő pedig akkor, amikor befejezte az aktuális átvitelt.

4 Nyomtatottáramkör-tervezés

4.1 Perifériák az áramkörön

A nyomtatott áramkörre került a felhasználói kezelőfelület (nyomógombok, LCD és LED-ek) és a CODEC-ek csatlakozói. Az áramkör és a fejlesztő kártya között a már említett 40 érintkezős csatlakozósor teremt kapcsolatot.

Az LCD kijelző a PS-hez csatlakozik 4 adatvezetékkel, tápfeszültséget kap, és kapcsolódik hozzá néhány kondenzátor. Az LCD adatvezetékei közül 3 a hagyományos SPI kommunikációval egyező: SCK (órajel), SDA (soros adat bemenet) és /CS (chip kiválasztás). A negyedik vezeték megadja, hogy az LCD-nek küldött egy bájtos szó vezérlést vagy adatot jelent. Tehát az LCD számára mind a 4 vezeték bemenet, szemben a hagyományos SPI kommunikációval, ahol az egyik kimenet lenne.

A modulon 7 darab gomb található: felvétel, menü és 5 db navigációs gomb: fel, le, jobbra, balra és kiválasztás. A gombok egyik érintkezője a földhöz csatlakozik, a másik érintkezők 1-1 PL adatvezetékhez. Külső felhúzó ellenállás nem szükséges, ez beállítható az FPGA-ban.

A 6 darab LED közül az egyik – a zöld – a tápfeszültséget jelzi. A többi (1 piros és 4 kék) egy-egy 470 Ohmos ellenálláson keresztül egy-egy PS adatvezeték és a föld közé vannak kapcsolva.

A CODEC modulok egyenként rendelkeznek 6 darab órajel bemenettel (1-1 MCLK, 1-1 BCLK és 1-1 LRCLK az ADC-nek és a DAC-nek) és 1-1 soros adat be- és kimenettel. Ez tehát összesen 12 órajelet jelent, amelyeket azonban 4-esével össze lehet fogni. (A két-két ADC és DAC járhat ugyanarról a rendszerórajelről, és az I2S órajelek is lehetnek azonosak.)

Így a 2 darab CODEC összesen 7 vezetékkel kapcsolódik a PL-hez: 3 órajel, 2 adat be- és 2 adat kimenet. CODEC-ek ezen kívül 5V-os analóg és 3,3V-os digitális tápfeszültséget igényelnek, ezek is a saját áramkörről érkeznek.

4.2 Tervezési hiba

A nyomtatott áramkör eredetileg 3,3V-os tápfeszültségre terveztem, az LCD, a LED-ek és a CODEC digitális része erről működtek volna. Így az adatvezetékeken is ezek a

jelszintek közlekedtek volna. Az FPGA-kban általában beállíthatók az IO blokkok (ki-és bemenetek) feszültség szintjei, így a perifériák 3,3V-os szintjei nem okoznának problémát. Az Ultra96 kártyát viszont 1,8V-os jelszintre tervezték, így az FPGA bankjai is 1,8V-ról működnek, ezért ennél magasabb feszültséget nem lehet beállítani az IO blokkoknak. (Ráadásul a PL-hez tartozó IO blokkok olyan bankban találhatók, amelyek egyáltalán nem konfigurálhatók 1,8V-os szint fölé.) Ez sajnos a nyák összeszerelése után, az első teszteléskor derült ki. Így a nyákon semmilyen periféria nem működött (a piros LED-en és a tápfeszültséget jelző zöld LED-en kívül).

Ha az eltérő jelszinteket előre figyelembe vettem volna, akkor jelszint konverterek alkalmazásával könnyen át lehetett volna hidalni a problémát. Viszont a kész áramkörön valamilyen egyszerű megoldást kellett alkalmazni, hogy ne kelljen sok módosítást elvégezni. A megoldás – a konzulensem, Fehér Béla ötlete alapján – a következő lett: ha a CODEC-ek és az LCD tápfeszültségét lejjebb visszük, akkor azok képesek lehetnek működni az 1,8V-os jelszintekről is.

Az LCD minimális tápfeszültsége 2,5V, a CODEC-é pedig 2,7V. Az LCD-nél a logikai magas jelszint minimum 0,8*VDD kell hogy legyen, a CODEC-nél ez az érték 0,7*DVDD. Ha tápfeszültséget 2,6V-ra csökkentjük, akkor ezzel éppen a működés határára (vagy kicsivel alá) kerülünk több szempontból is: egyrészt ez elméletileg már túl alacsony a CODEC-nek. Valamint az 1,8V kicsit kevesebb mint a tápfeszültség 70%-a, tehát elméletileg az LCD-nek ez már túl alacsony a logikai magas szinthez.

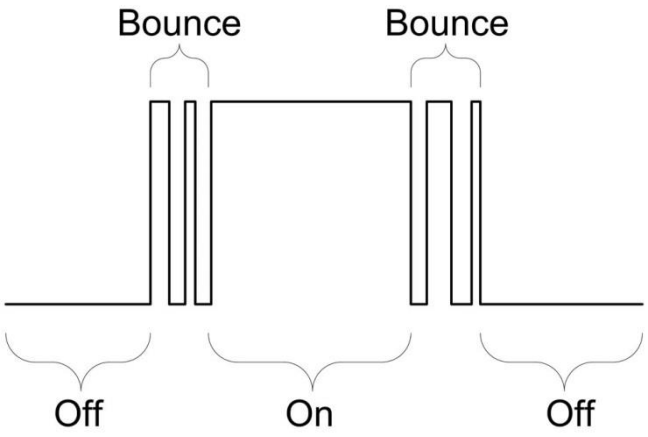
A tápfeszültség csökkentését egy Si-diódával értük el, amelyet a 3,3V-os feszültség stabilizátor után kötöttünk, a perifériák a dióda katódjáról kapják a tápot. A diódán 0,6-0,7V feszültség esik, így a perifériák 2,6-2,7V-ról működnek. Az áramkör szerencsére így már működött, tehát ezzel a trükkel sikerült megspórolni a jelszint konverterek használatát és az ezzel járó áramkör átalakítást. A CODEC-ek soros kimenete a digitális tápfeszültségről működnek, így a magas szint ott 2,6-2,7V, amely túl magas az FPGA számára, így a két adatvezetéket 1-1 ellenállás osztóval konvertáltam a megfelelő szintre. (1 kOhmos és 470 Ohmos ellenállás, ami alapján a 2,7V-ból: 2,7 / (1000 + 470) * 1000 = 1,84V lett, amely már megfelelő.) A többi adatvezetéket az FPGA hajtja meg, így azokkal további teendő nem volt a fentieken kívül.

A mellékletek között az utólag módosított áramkör kapcsolási rajzát mutatom be. A kapcsolási rajz a *Schematic.pdf* fájlban található. A függelékek között megtekinthetők az eredeti (módosítás nélküli) nyákrajzok: 1, 2 és 3.

5 FPGA hardverfejlesztés

5.1 Pergésmentesítő

Mechanikus kapcsolóknál és gomboknál sajnos szinte mindig jelentkezik a pergés jelensége (angolul bounce). Ez azt jelenti, hogy a gomb lenyomásakor illetve felengedésekor a kapcsoló nem egyetlen fel- illetve lefutó élt generál, hanem többet. Ez a gomb fizikai kialakításából adódik, amikor a gombot lenyomjuk az érintkezők érintkeznek (lefutó él generálódik) majd „visszapattannak”, de a nyomás hatására szinte azonnal újra érintkeznek. Ez egymás után többször (vagy több tízszer) megismétlődik. A jelenség néhány ms-ig tart. Ez látható a 2. ábrán.

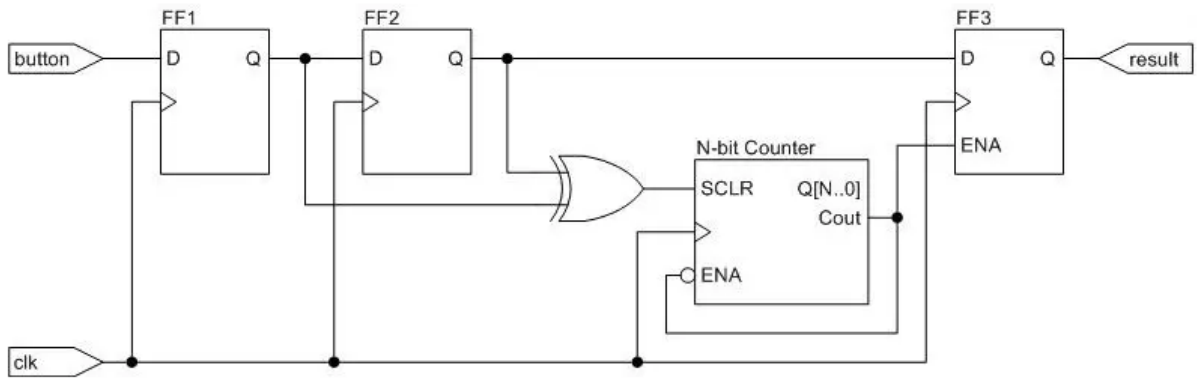


2. ábra: Kapcsoló vagy gomb pergése [1]

Vannak olyan gombok, amelyeknek két kimenetük van, a lenyomott és felengedett állapotot is 1-1 rövidzár jelzi a különböző kimeneteken. Ezeknél a pergésmentesítést egy SR flip-floppal egyszerűen el lehet végezni. Azonban egy olyan gombnál, amelynek csak egyetlen kimenete van (tehát a lenyomott állapotot rövidzár jelzi, a felengedetett pedig szakadás, esetleg fordítva) nincs más lehetőség, csak az, hogy megvárjuk a pergés végét.

5.1.1 Belső működés

A pergésmentesítést a 3. ábrán látható áramkör alapján implementáltam Verilog nyelven. Az áramkör bemenetei: button (bemenet gomb felől) és clk (órajel). Az áramkör kimenete a result, amely a pergésmentesített jel.



3. ábra: Pergésmentesítés [11]

Az áramkör működése: a bemenetről érkező jelet két D flip-floppal késleltetjük. A két flip-flop kimenetét kizáró vagy (XOR) kapcsolatba hozzuk, ezzel detektáljuk az élváltást. (A kapu kimenete le- és felfutó él esetén is 1 lesz, egyébként 0.)

Az áramkör tartalmaz egy N bites szinkron számlálót. A számlálót a carry jel negáltja engedélyezi, tehát a számláló túlcsordulás után nem indul újra. A számlálót a XOR kapu kimenete alaphelyzetbe állítja, így élváltáskor a számláló mindig újraindul.

A carry kimenet engedélyezi a kimeneti flip-flopot, tehát a kimeneti flip-flopba akkor töltjük be a bemenetet (pontosabban a bemenetet kétszeresen késleltetve), amikor a számláló túlcsordul. Így az utolsó élváltás után $2^N + 2$ órajellel kerül csak mintavételezésre a bemenet. Új élváltás esetén a kimeneti flip-flop megtartja az előző értékét, és csak jóval az utolsó élváltás után íródik bele az új érték. Ezzel megtörténik a pergésmentesítés.

A pergésmentesítési idő hossza a számláló bitszélességével szabályozható. 100 MHz-es órajel esetén 10 ms-os pergésmentesítési időhöz 20 bites számlálóra van szükség:

$$\frac{2^{20} + 2}{100\ MHz} = 10{,}49\ ms$$

Tehát ha 10 ms alatt lejátszódik a pergés jelensége, akkor az áramkör ezt képes elfedni, a GPIO számára már csak egyetlen fel- vagy lefutó él lesz látható. Természetesen az áramkör beiktatása késleltetést is eredményez. A gomb lenyomása után a GPIO erről csak 10,49 ms fog értesülni, ez azonban a projektben nem okoz problémát.

Implementáláskor a fenti áramkörön kicsit változtattam: a bemeneten 3 flip-flop található. A számlálót pedig nem a carry jel engedélyezi, hanem a számláló maximuma. (Pontosabban annak negáltja.)

5.1.2 Implementálás

A modul deklarálása a következő:

```
module debounce_bnt
#(
	parameter integer CNTR_WIDTH = 20
)
(
	input  wire clk,
	input  wire rst,
	input  wire bt_in,
	output wire bt_out
);
```

Paraméterben megadható a számláló bitszélessége (*CNTR_WIDTH*). A bemenetek: órajel (*clk*), alaphelyzetbe állító jel (*rst*) és a gomb bemenet (*bt_in*). A modul kimenete (*bt_out*) a pergésmentesített jel. Ez a modul egyetlen gomb pergésmentesítését képes elvégezni, ezért létrehoztam egy másik modult, amely ebből többet is példányosít.

A modul deklarációja a következő:

```
module debounce_bus
#(
	parameter integer BUS_WIDTH = 8,
	parameter integer CNTR_WIDTH = 20
)
(
	input  wire                  clk,
	input  wire                  rst,
	input  wire [BUS_WIDTH-1:0] bt_in,
	output wire [BUS_WIDTH-1:0] bt_out
);
```

Itt található egy további paraméter, amelyen megadhatjuk a busz szélességét (*BUS_WIDTH*), tehát azt, hogy hány gomb pergésmentesítését szeretnénk elvégezni. A be- és kimenetek azonosak az előző modullal, csak a *bt_in* és *bt_out* itt több bites lesz (a paraméterben megadott szélesség). Minden gombhoz tartozó számláló bitszélessége azonos, ezt a *CNTR_WIDTH* paraméterben lehet megadni.

A modult a tényleges kipróbálása előtt szimulációval ellenőriztem. A forráskódok a *HDL\debounce_btn.v* és a *HDL\debounce_bus.v* fájlban találhatóak.

5.2 I2S modul

5.2.1 Funkcionális működés

Az I2S modul a PL-ben található 100 MHz-es rendszerórajelről jár (*m_axis_clk*). Bemenetén megkapja a CODEC-ek felől érkező soros adatokat (*sdi0, sdi1*) és az I2S

órajeleket (*bclk, lrclk*). A modulnak van 3 soros audió kimenete is: 1-es csatorna (*sdo0*), 2-es csatorna (*sdo1*) és a kettő keveréke (*sdo_mix*).

A modul AXI-Streamen keresztül továbbítja a beérkező 24 bites mintákat. Az AXI-Stream adatvezetéke 32 bites (*m_axis_tdata*). Ezen kívül még 3 vezeték tartozik az AXI-Stream buszhoz: *m_axis_tvalid*, amely jelzi, ha van érvényes adat a buszon, *m_axis_tready*, amelyen keresztül a fogadó modul (pl. a FIFO) jelzi, hogy elvette az adatot. Valamint az *m_axis_tlast*, amelyen egy átviteli blokk utolsó adatát jelzi (például 1024 átvitt bájt után). Ennek számlálásához egy regiszter tartozik, amelynek szélessége – így a blokkban átvitt bájtok száma – a *TLAST_REG_WIDTH* paraméteren keresztül állítható be.

A modulba 1 mintavételi periódus alatt négy darab 24 bites minta érkezik (minden csatornáról egy), ez összesen 96 bit, tehát 3 AXI-Stream átvitel alatt lehet őket továbbítani. Az alábbi táblázatban látható, hogy a modul hogyan rendezi össze az I2S mintákat AXI-Stream szavakba. Az ilyen módon továbbított mintákat a DMA bemásolja a processzor memóriájába, ahonnan minden átrendezés nélkül írhatók a WAV fájlba.

AXI	**3. AXI**		**2. AXI**		**1. AXI**	
bitek	31 8	7 0	31 16	15 0	31 24	23 0
I2S	**CH2 jobb**	**CH2 bal**	**CH1 jobb**	**CH1 bal**		
bitek	23 0	23 0	23 0	23 0		

1. táblázat: Minták továbbítása AXI-Streamen

Az I2S minták beérkezés után egy átmeneti regiszterben lesznek tárolva, az AXI-Stream buszon ebből a regiszterből kerülnek elküldésre. A fogadó modulnak egy mintavételi periódus alatt tehát 3 mintát kell elvennie az I2S modultól. Az új I2S minta nem kerül tárolásra, ha az átmeneti regiszter nem üres (tehát a fogadó modul még nem vette el a mintákat), viszont ez így adatvesztéshez vezet. Minden bekövetkezett adatvesztés eggyel növeli a *data_loss_cntr_reg* értékét. Ez a regiszter 32 bites és a külvilág felől elérhető a következő vezetékeken: regiszter értéke (*data_loss_cntr*), törlés (*data_loss_clear*) és megszakítás jelzése, nem 0 érték esetén (*data_loss_irq*).

A modulnak van egy engedélyező jele is (*en*). A modul tiltása esetén a beérkező mintákat a továbbiakban nem menti és nem küldi el őket AXI-Streamen. Ha a modult tiltjuk, akkor az még befejezi az éppen folyamatban lévő AXI blokk átvitelt, tehát csak a tlast jellel együtt kerül tiltásra. Ezzel elkerülhető a DMA hiba, mivel az AXI DMA mindenképpen igényli a tlast jelet.

5.2.2 Belső működés

Az I2S órajelen éldetektálást végzünk: a bejövő órajelek egy 3 bites shift regiszterbe kerülnek, és ennek a regiszternek az utolsó két bitjét vizsgáljuk. A regiszterek jobbra shiftelnek, így az utolsó két bit 2'b01 értéke jelzi a lefutó élet, a 2'b10 pedig a felfutó élt jelenti.

Azért, hogy a soros adatok szinkronban maradjanak az órajelekkel, azok először egy-egy szintén 3 bites shift regiszterbe kerülnek. (A 3 bites shift regiszterek folyamatosan működnek, a rendszerórajel minden ütemére lépnek egyet.) Majd innen egy-egy 24 bit shift regiszterbe, amely balra shiftel (az I2S adatformátuma miatt). Ezeket a regisztereket a bit órajel (*bclk*) felfutó éle engedélyezi. Az aktív csatornát jelző *lrclk* jel lefutó élére a bal csatornákat, felfutó élre pedig a jobb csatornákat mentjük az átmenetei 96 bites regiszterbe, amennyiben a modul engedélyezve van és a regiszterben már nincs érvényes adat. (Az átmeneti regiszterbe az 1. táblázatban látható sorrendben kerülnek az adatok.)

Egy 2 bites számláló számolja az AXI átviteleket. A jobb csatornák megérkezése után az értéke 3 lesz állítva, majd minden AXI átvitellel eggyel csökken. Ez a számláló vezérel egy multiplexert, ami a 96 bites átmeneti regiszter megfelelő 32 bitjét az AXI adatkimenetre köti.

Egy másik számláló számolja az AXI átviteleket a tlast jel előállításához. Amikor a számláló eléri a maximumát, akkor küldd egy tlast jelet. A számláló bitszélessége paraméterben megadható.

A kimeneti I2S adatok (1. csatorna, 2. csatorna és mix) egy-egy 64 bites regiszterből kerülnek elküldésre. A regiszter alsó 24 bitjébe az *lrclk* órajel le- és felfutó élekor kerül betöltésre az adat (1. illetve 2. csatorna és a kettő összegének felső 24 bitje). Majd a *bclk* felfutó élekor eggyel balra léptetjük a regisztert. A regiszter legfelső bitje lesz a soros adat kimenet.

5.2.3 Implementálás

A modult Verilog nyelven készítettem el. Majd a hardvertervbe illesztés előtt szimulációval ellenőriztem a működését. A modul forráskódja melléketek között a *HDL\I2S.v* fájlban található.

5.3 Vivado block design

A block design megtalálható a mellékletek között a *Vivado Block Design.pdf* fájlban. A *HDL\const.xdc* fájlban a külső FPGA lábak és a belső vezetékek összekötésének leírása látható.

A teljes FPGA-s és a processzoros rendszert Vivado-ban raktam össze a block design funkció segítségével. Itt példányosítottam a saját Verilog modulokat és gyári IP-kat illesztettem be. Az alábbiakban leírom, hogy milyen modulok találhatók a rendszerben, zárójelben fogom jelezni a block designban megadott nevüket.

A rendszer központi eleme a Zynq UltraScale+ MPSoC (*zynq_ultra_ps_e_0*). A Zynq-en belül be van kapcsolva a GPIO, az SDIO 0 és az SPI 0 periféria, valamint ezek a megfelelő portokra ki vannak vezetve (40 érintkezős bővítő csatlakozó a fejlesztőkártyán). A 100 MHz-es PL órajelet a Zynq szolgáltatja, ezt is a beállításainál kell konfigurálni. A Zynq-nek két AXI portja van bekapcsolva a PL-lel történő kommunikációhoz: AXI HPM0 LPD és AXI LPD. Az előbbi egy master port, az utóbbi pedig slave.

A rendszer többi eleme gyári IP-ként vagy saját perifériaként lett hozzáadva a rendszerhez. A slave AXI LPD portra egy AXI Interconnect modulon keresztül (*axi_interconnect_0*) egy AXI DMA kapcsolódik (*axi_dma_0*). A DMA és a Zynq közötti kommunikációban a DMA viselkedik materként. A DMA AXI-Stream bemenettel rendelkezik, melyre egy AXI-Stream FIFO kapcsolódik (*fifo_generator_0*), a mélysége 1024 szó. A FIFO-ra kapcsolódik a saját tervezésű I2S modul (*I2S_0*). Az I2S modulban a tlast regiszter szélességéhez 8 bit van megadva, tehát a regiszter 256-ig tud elszámolni. Ez azt jelenti, hogy egy DMA átvitel egyszerre 1024 bájt (AXI-Stream adat szélessége 32 bit.)

Az I2S modul konfigurálását egy két csatornás AXI GPIO végzi (*i2s_ctrl*). A nyomógombok először a saját tervezésű pergésmentesítőre csatlakoznak (*debounce_bus_0*), majd innen egy AXI GPIO-ra (*btn_gpio*). A rendszerben található egy másik AXI Interconnect modul is (*ps8_0_axi_periph*), erre kapcsolódik a két GPIO modul és a DMA (*axi_dma_0*) egy slave AXI-Lite porttal. A DMA-t ezen a porton keresztül lehet felkonfigurálni. Az AXI Interconnect modul (*ps8_0_axi_periph*) a Zynq AXI HPM0 LPD portjára kapcsolódik, itt a Zynq a master eszköz.

A DMA és a gombokat kezelő GPIO képes megszakítást küldeni, a megszakítás vonalaikat egy Concat blokk fogja össze több bites vezetékké (*xlconcat_0*), majd így csatlakoznak a Zynq megszakítás bemenetére.

A CODEC órajelét egy Clocking Wizard (*clk_wiz_0*) állítja elő a rendszerórajelből. Majd ezt egy Binary Counter modul (*c_counter_binary_0*) leosztja. A bináris számláló kimenetéből egy-egy Slice blokk (*xlslice_0* és *xlslice_1*) választja ki a megfelelő vezetéket, így létrejön a BCLK és LRCLK. (Ezek a külvilághoz is és az I2S modulhoz is kapcsolódnak.)

A rendszerben található még két 1 bites, 3-1-es multiplexer (*mux_3_1_0* és *mux_3_1_1*), ezek kimenetére csatlakoznak a CODEC-ek soros bemenetei. A multiplexerek bemenetei az I2S modulra vannak csatlakoztatva. A vezérlést az I2S modul vezérléséért is felelős GPIO végzi (*i2s_ctrl*), egy-egy Slice blokkon keresztül (*xlslice_2* és *xlslice_5*).

6 Szoftverfejlesztés

A szoftvernek két lényeges feladata van: a felhasználói felület biztosítása és a minták mentése az SD kártyára. A szoftver forráskódja megtekinthető a mellékletek között a *C\audio recorder* mappában.

A projekt sikeres működéséhez szükséges volt megismerni az SD kártya írását, ehhez készítettem egy külön alkalmazást. Ennek a forráskódja a *C\SD speed test* mappában található.

6.1 SD sebesség teszt

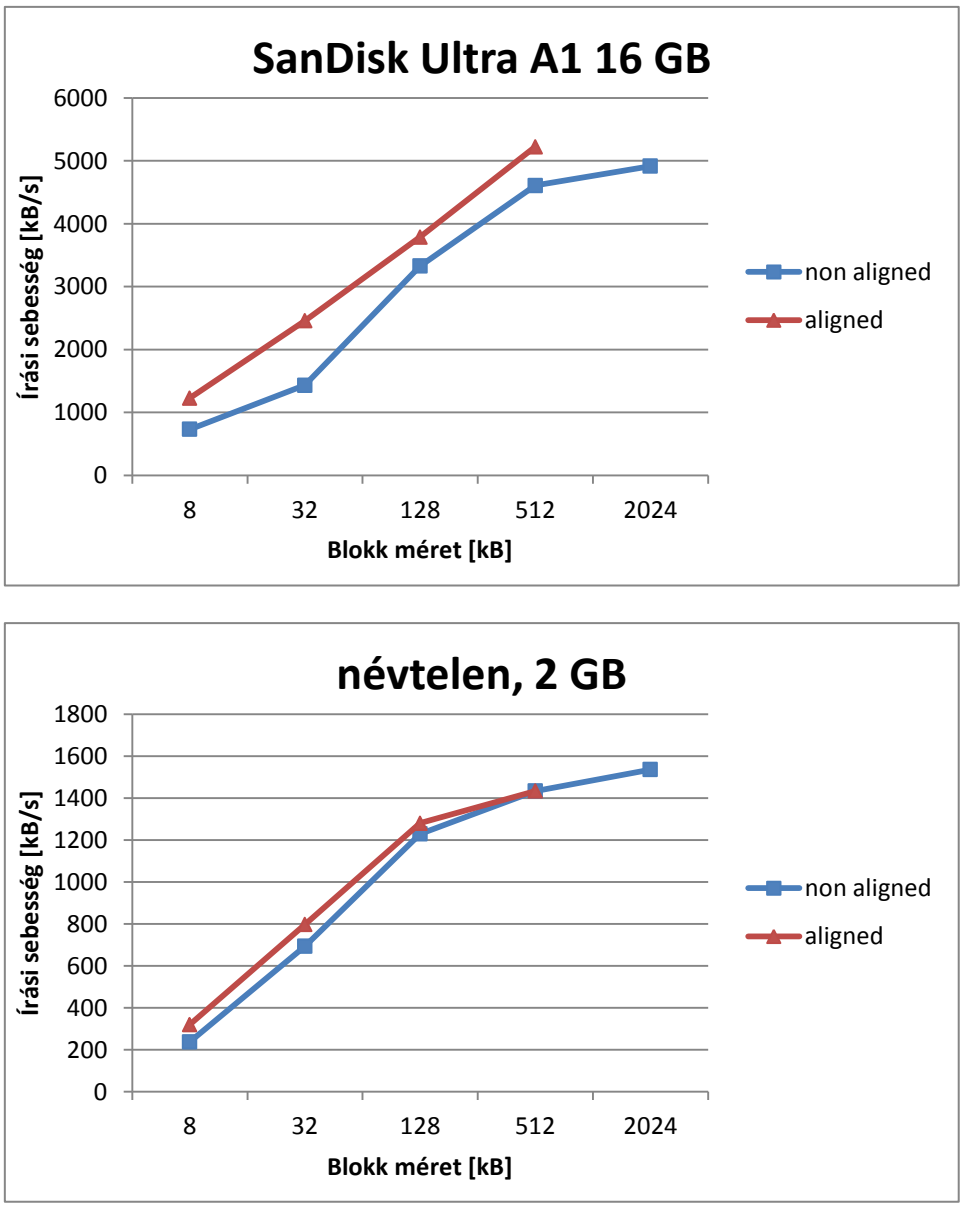
Az SD kártya írásánál nem mindegy, hogy mekkora blokkokat írunk egyszerre a kártyára. Minél nagyobb a blokkméret, annál nagyobb írási sebességet érhetünk el. Ennek vizsgálatára írtam egy teszt szoftvert. A szoftver a végleges projekthez hasonlóan működik, először létrehoz egy WAV fájlt, majd ahhoz ír hozzá újabb blokkokat. A blokkok írási sebességét méri, majd kiírja a mérési adatokat a kijelzőre.

A FAT fájlrendszerben van egy lemezfoglalási egység, amely formázásnál dől el, hogy mekkora lesz az adott adathordozón. Ez 2-nek egész számú hatványa, például 4-8-16-32 kB. Ahhoz, hogy gyors írást érjünk el érdemes legalább a lemezfoglalási egység méretével megegyező blokkokat írni a kártyára, de jobban járunk minél nagyobb mérettel.

A WAV fájl elején egy 44 bájtos fejléc található. Először tehát ez kerül írásra. Ennek írásakor lefoglalunk a egy lemezfoglalási egységgel megegyező blokkot a kártyáról, ez legyen mondjuk 4 kB. Amikor az első blokkot írnánk a hangfájlból (legyen ez is 4 kB), akkor az az előbb lefoglalt egységbe már nem fér el, mert annak az elején ott van 44 bájt. Tehát az írni kívánt blokkunk csak két lemezfoglalási egységbe fog elférni. Az összes következő blokknál fellép ez a probléma. Így jobban járunk, ha a WAV fejlécét úgy írjuk ki, hogy a 44 bájt után megfelelő számú 0-val töltjük ki a fejlécet. (Így a fájl elején lesz egy rövid csend.)

Az előző bekezdésben tárgyalt problémát is megvizsgáltam. Végeztem olyan írási teszteket, ahol rögtön a 44 bájtos fejléc után következett az első blokk. Valamint olyat is, ahol a fejlécet kiegészítettem annyi 0-val, hogy az megegyezzen az írási blokkmérettel. Ez utóbbi esetet aligned (illesztett) verziónak neveztem el.

A méréseket elvégeztem egy SanDisk Ultra A1 16 GB-os kártyával és egy névtelen 2 GB-os SD-vel is. A tesztekben az írási blokkméret és az illesztett / nem illesztett verzió függvényében vizsgáltam az írási sebességet. A teszt során írt teljes adatmennyiség a három kisebb blokkmérethez 64 MB volt, a két nagyobbhoz pedig 256 MB.



4. ábra: SD kártya írási sebesség tesztek

A 4. ábrán láthatók a teszt eredmények. A SanDisk kártyával jóval nagyobb sebességeket lehetett elérni. Valamint itt látható, hogy az illesztett írás is okoz sebesség növekedést. 32 kB-os írási blokkméret már bőven elegendő egy jó minőségű SD kártya esetén a specifikációban támasztott követelmények teljesítéséhez, mivel így kb. 2500 kB/s sebességet lehetett elérni. (A szükséges 1125 kB/s.)

A Zynq-ben található SDIO a kártyát 19 MHz-ről járatja, és 4 bites adatbuszt használ. Így az elméleti maximális írási sebesség 9,06 MB/s. Azonban ez csak elméleti érték, mivel az SDIO buszon keresztül nem csak adat közlekedik, hanem vezérlő és hibaellenőrző (CRC) jelek is.

6.2 Szoftver modell

A mintákat a DMA másolja be egy cirkuláris pufferbe, egyszerre 1024 bájtot ami megfelel az 5.3 fejezetben tárgyaltakkal. A puffer a rendszer memóriában található, mérete 64 MB. Ez elég nagy méret, de a teljes rendszermemória 2 GB, amihez képest elhanyagolható.

Lehetne a puffert kisebbre választani, de ennek az optimalizálásával nem foglalkoztam a projekt során. Azonban túl kicsire nem szabad állítani a puffert, mivel a mért kb. 2500 kB/s-os írási sebesség csak egy átlagos érték. Ekkor egy 32 kB-os blokk írása átlagosan 12,8 ms-ig tart. Előfordulhatnak olyan esetek is, amikor egy blokk írása ennél sokkal több időt vesz igénybe. Egy nagyméretű puffer ezeket a hosszabb időket képes elfedni.

Felvétel elindításakor elindul egy 1024 bájtos DMA átvitel. A DMA a Zynq és az I2S között található FIFO-ból automatikusan másolja az adatokat a memóriába. Amikor elkészül az 1024 bájttal, küld egy megszakítást a processzornak. A processzor a megszakítás rutinban elindít egy újabb 1024 bájtos átvitelt, amennyiben van még szabad hely a cirkuláris pufferben. Ha nincs hely, akkor egy változót növel, ezzel jelezve, hogy a felvételben keletkezett egy hiba. Ilyenkor is elindít egy DMA átvitelt, viszont ezt egy másik pufferbe másolja, ahonnan nem kerül kiolvasásra, tehát ezt a mintát egyszerűen eldobja. Erre azért van szükség, hogy az I2S modultól elvegyük az adatokat, ott ne keletkezzen hiba.

A DMA átvitel lezajlása után egy változó értékét 1024-el növeli a megszakítás rutin, ezzel jelzi, hogy újabb adat került a cirkuláris pufferbe. A program main függvényében található egy végtelen ciklus, ebben történik az SD kártyára írás. Amennyiben rendelkezésre áll elég adat (az előbb említett változó értéke nagyobb mint 32768 = 32 kB), akkor meghívunk egy függvényt, ami elvégzi a 32 kB-os blokk írását a kártyára. Ez a függvény csökkenti a változó értékét 32768-cal. Ha az írás valamiért sikertelen volt akkor egy másik változót növel, ezzel jelzi, hogy lesz hiba a mentett fájlban.

Minden SD blokk írása után a szoftver invertálja az áramkörön található piros LED-et. Így a LED villogása jelzi ha éppen felvétel zajlik. A LED nem egyenletesen villog, ennek az az oka, hogy a blokkok írási ideje nem egyezik meg, ahogy erről előbb szó volt.

Az idő mérését nem külön időzítő modul végzi, hanem a DMA megszakítás. A DMA megszakítások (körülbelül) egyenletesen érkeznek. 1024 bájt átvitele után érkezik egy megszakítás. Egy mintavételi periódus alatt 12 bájtot küld az I2S modul, tehát 1024 bájt megfelel 85,333 mintavételi periódusnak. Ha ezt elosztjuk a mintavételi frekvenciával (96000), akkor megkapjuk, hogy mennyi időnként érkezik egy DMA megszakítás: 0,888 ms. Ennek a reciprokjából megkapjuk, hogy 1125 db DMA megszakítás felel meg 1 másodpercnek. A DMA megszakítás tehát egy idő változót is mindig növel.

A gombok kezelése szintén megszakításon keresztül történik. Ha befut egy megszakítás a GPIO felől, akkor a megszakításrutin kiolvassa a GPIO regisztert és elmenti, egy-egy flagbe, hogy melyik gombot nyomták le.

A main függvény a mellékletek között a *C\audio recorder\main.c* fájlban található. A DMA függvényei a *C\audio recorder\dma.c* és *C\audio recorder\dma.h* fájlokban találhatók. A gombok kezelése a *C\audio recorder\buttons.c* és *C\audio recorder\buttons.h* fájlokban történik.

A megszakítások konfigurálása *C\audio recorder\intr.c* és *C\audio recorder\intr.h* fájlokban történik. A LED-ek kezelése pedig a *C\audio recorder\led.c* és *C\audio recorder\led.h* fájlokban.

Az I2S modul inicializálását, engedélyezését, tiltását és a regiszterének olvasását a *C\audio recorder\i2s.c* és *C\audio recorder\i2s.h* fájlokban található függvények végzik el.

6.2.1 LCD kezelés

Az LCD kezeléséhez készítettem egy alacsonyszintű drivert, amely a kommunikációt végzi a kijelzővel SPI buszon keresztül, valamint inicializálja azt. A szövegek kiírását egy GitHubról letöltött [12] függvénykönyvtár végzi. A függvénykönyvtárat némileg átalakítottam, hogy illeszkedjen a saját LCD driveremhez.

A saját driverek az *C\audio recorder\lcd_driver.c* és *C\audio recorder\lcd_driver.h* fájlban lett megvalósítva, a letöltött függvénykönyvtár fájljai: *C\audio recorder\uc1701.c C\audio recorder\uc1701.h* és *C\audio recorder\fonts.c*.

6.2.2 SD kártya kezelés

Az SD kártya csatlakoztatása, új fájl megnyitása, fájl írása, stb. az *C\audio recorder\sd_card.c* és az *C\audio recorder\sd_card.h* fájlokban található. Az új fájlt mindig

úgy nevezi el az eszköz, hogy a benne található szám eggyel nagyobb legyen az előző fájlnál. (Pl. előzőleg a „REC013.WAV” fájlt mentette a kártyára, akkor az új fájl „REC014.WAV” lesz.) A sorrendet áramtalaníts után is folytatja, ehhez az aktuális fájlnevet eltárolja egy szöveges fájlban az SD kártyán (LAST.TXT). Új fájl létrehozásakor először innen kiolvassa az előző fájl nevét.

A fájl létrehozás után a WAV fejlécet írja a fájlba, ez 44 bájt hosszú, így mögé ír 32724 darab 0 értékű bájtot. Így kerek 32 kB-ot ír a kártyába, tehát a mintákat tartalmazó blokkok illesztve kerülnek a kártyára, ahogy erről a 6.1 fejezetben szó volt. A fejlécben el kell tárolni a fájlméretet (és a mintákat tartalmazó blokk méretét is), ezek előre nyilvánvalóan nem ismertek. Így a felvétel befejezése után a program a fájl elejére ugrik és megfelelően kitölti ezeket a mezőket.

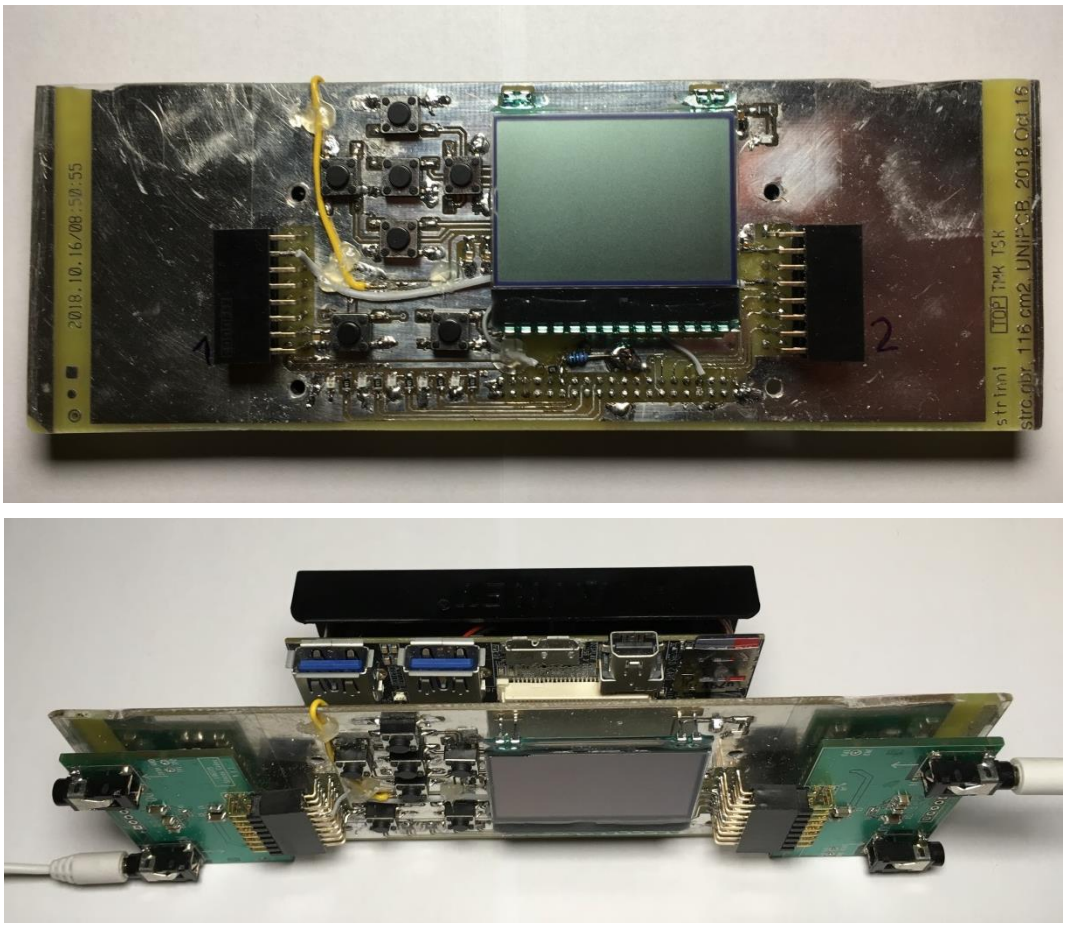
6.2.3 GUI

A grafikus felhasználó felület (angolul: graphical user interface, azaz GUI) a *C\audio recorder\gui.c* és az *C\audio recorder\gui.h* fájlokban található függvények vezérlik. A gui függvény akkor kerül meghívásra a main függvényből, ha valamelyik gombot lenyomták. Ekkor elvégzi a gomb lenyomásának megfelelő tennivalókat (felvétel indítása, megállítása, menü megnyitása, kijelző írása, stb.).

7 Felhasználói útmutató

Az eszközön két CODEC modul található, a bal oldali az egyes számú, a jobb oldali a kettes számú. Mind a két modulon egy bemenet és egy kimenet található, ezek egyaránt 3,5 mm-es sztereó jack csatlakozók. A be- és a kimenetek is vonalszintűek.

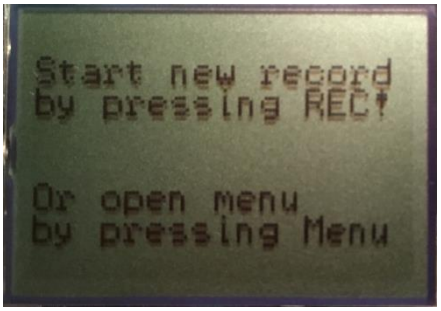
A panel közepén egy LCD kijelző, 7 gomb és 6 LED található. Ezek kezelőfelületként szolgálnak. Az SD kártyát az alsó nyomtatott áramkör jobb felső sarkában található foglalatba kell illeszteni. Az eszköz tápellátását (6-18V, 2A) az alsó áramkör bal alsó sarkánál lehet csatlakoztatni. Mellette található a be- és kikapcsoló gomb. Az 5. ábrán látható két fénykép az eszközről, a felsőn a kezelőpanel látható, az alsón pedig a fejlesztőkártya, az SD kártya foglalattal.



5. ábra: Fénykép az eszközről

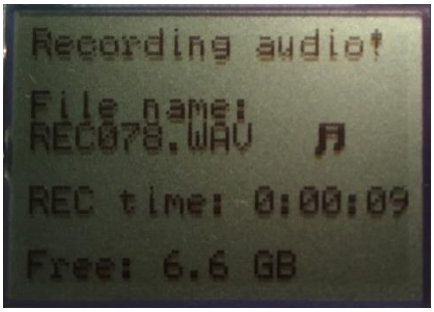
Az eszközön bekapcsolás után egy üdvözlő képernyő jelenik meg: „Hello!” felirat, és a szoftver fordításának időpontja. Ez rövid időn belül eltűnik és megjelenik a készenléti

képernyő (amely a 6. ábrán látható). Ez arról tájékoztat minket, hogy a REC gombbal (bal alsó) elindíthatjuk a felvételt, vagy a Menu gombbal (jobb alsó) megnyithatjuk a menüt.



6. ábra: Készenléti képernyő

A felvétel elindítása után az eszköz kiírja, hogy jelenleg felvétel zajlik („Recording audio!”), kiírja a fájl nevét (pl.: „REC078.WAV”), mellette egy hangjegyikon villogással jelzi a felvételt. Lejjebb a kijelző tájékoztat arról, hogy mióta tart a felvétel (pl.: „REC time: 0:00:09”), a legalsó sorban pedig az SD kártyán található szabad területet írja ki (pl.: „Free: 6.6 GB”). A képernyőkép a 7. ábrán látható.



7. ábra: Képernyő felvétel közben

A felvételt a REC gomb újbóli lenyomásával állíthatjuk meg, ekkor megjelenik a „Recording stopped” felirat. Amennyi a felvétel sikeres volt a „Successful REC!” felirat is megjelenik a kijelzőn. Ez azt jelenti, hogy minden beérkező mintát sikeresen mentett az eszköz az SD kártyára.

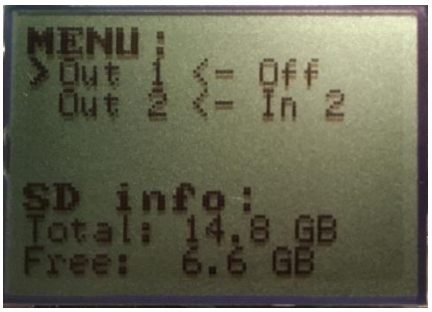
Amennyiben valamilyen hiba lépett fel felvétel közben, erről tájékoztat az eszköz (puffer túlcsordulás, SD kártya írási hiba, DMA hiba), valamint azt is kiírja, hogy ez a hiba hányszor lépett fel. Ekkor a felvétel nem lesz tökéletes. Legalul az eszköz kiírja a készült felvétel nevét és hosszát, valamint a „Wait” feliratot. A leállítást követő képernyőkép látható a 8. ábrán.

8. ábra: Információk felvétel megállítása után

Rövid időn belül az eszköz visszatér a készenléti állapotba, így új felvételt kezdhetünk meg.

A menü felvétel közben és a készenléti kijelzőről is megnyitható, a Menu gombbal. Ekkor a 9. ábrán látható képernyőhöz hasonló fogad minket. A menü tetején lehetőségünk van a kimeneteket beállítani. A fel-le gombbal választhatunk a két kimenet között, a jelenleg kiválasztott kimenetet egy „>” jelzi. A kiválasztott kimenetnél a jobbra-balra nyilakkal válthatunk, hogy melyik bemenetet küldje el rajta. Négy lehetőség között választhatunk: „In 1” (1-es számú CODEC bemenete), „In 2” (2-es számú CODEC bemenete), „Mix” (a két CODEC jelének keverése) valamint „Off”, ekkor az adott kimenten nem szól semmi.

A 9. ábrán látható beállítások szerint, az 1-es kimeneten nem lesz semmilyen hang továbbítva, a 2-esen pedig a 2-es bemenet hallgatható vissza.



9. ábra: Menü képernyő

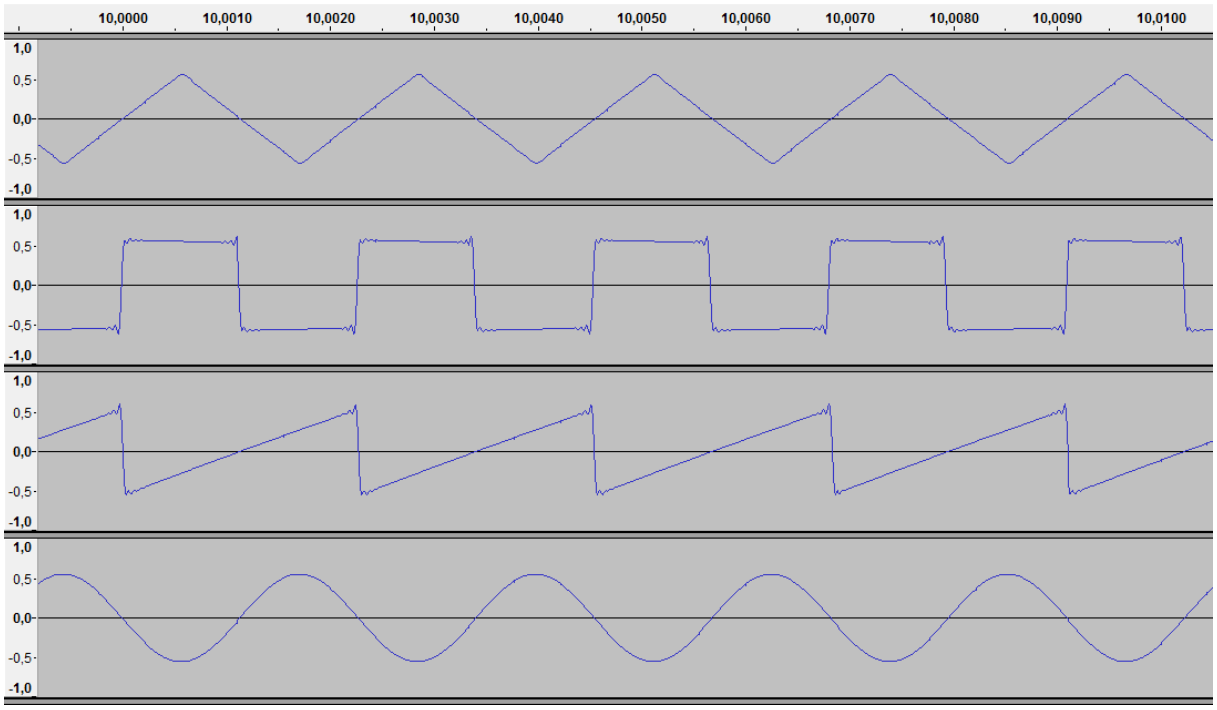
A menü képernyő alsó részén az SD kártyáról láthatunk információkat: a teljes kártya méretét és a szabad hely méretét. A fájlméreteket (itt és a felvétel közben is), formázottan jeleníti meg. Alapértelmezetten kB-ban, ha 1024 kB-nál nagyobb akkor MB-ban, ha pedig 1024 MB-nál is nagyobb akkor GB-ban.

Az eszköz működése közben nem szabad eltávolítani az SD kártyát.

8 Összegzés, továbbfejlesztési lehetőségek

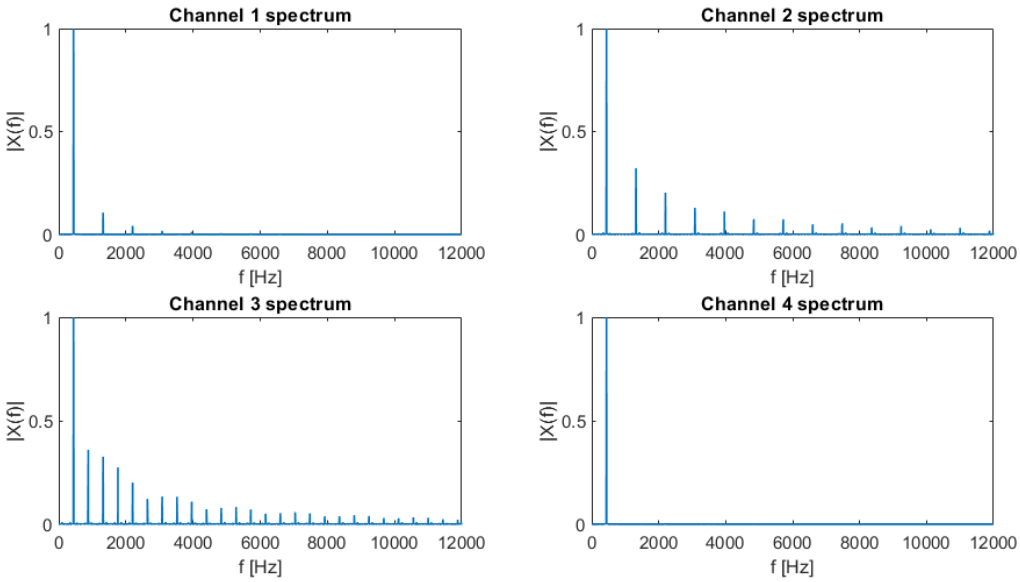
8.1 Tesztelés

A projekt fejlesztése közben több tesztet végrehajtottam. Az egyik teszt ezek közül az volt, hogy a 4 bemeneti csatornára 4 különböző teszt jelet kötöttem. Ezek a jelek rendre: háromszög, négyszög, fűrész és szinusz jel voltak. A frekvenciájuk egyaránt 440 Hz volt. Elindítottam így a felvételt, majd néhány másodperc után leállítottam és az SD kártyát áthelyeztem a laptopomba, hogy megvizsgáljam a rögzített fájlt.



10. ábra: Rögzített minta időtartományban

A rögzített minta látható időtartományban a 10. ábrán (Audacity programmal megnyitva). Látható a 4 csatornán a 4 különböző tesztjel. Ezek után a jelekből MATLAB segítségével spektrumot számoltam. A spektrumokat utólag átskáláztam, úgy, hogy minden spektrumban az alap harmonikus amplitúdója 1 legyen. Valamint csak a 0-12 kHz-s tartományon ábrázoltam őket (12 kHz fölött már közel 0 mindegyik spektrum értéke).



11. ábra: Rögzített minta amplitúdó spektruma

A 11. ábrán látható a MATLAB-bal ábrázolt amplitúdó spektrum. A spektrumok az elvárásoknak megfelelnek. Tehát a fenti teszt alapján jól működik a hangrögzítő.

Végeztem egy másik tesztet, ahol zenei anyagot rögzítettem hosszú ideig (1 óra 2 percig). A rögzítés leállítása után nem volt semmilyen hiba (puffer túlcsordulás, írási hiba, stb.), tehát a rögzítés sikeres volt minden bemeneti minta mentésre került. Az 1 órás felvétel közel 4 GB-ot foglal, ez a maximális fájlméret amely FAT32 fájlrendszeren tárolható. A rögzített anyagot visszahallgattam, teljesen megfelelő volt a minősége.

A kimeneti csatornák is megfelelően működtek. A menüből ki lehetett választani, hogy az egyes csatornákon melyik bemeneti csatorna szólaljon meg, vagy a kettő keveréke vagy ne szóljon semmi.

8.2 Fejlesztési lehetőségek

Az elkészült áramkör az alapvető követelményeket teljesíti, de még sok fejlesztési lehetőség maradt benne.

Ezek közül az első, egy teljesen saját áramkör tervezése, fejlesztő kártya és kiegészítő LOGSYS modulok nélkül. Így az áramkör önállóan lenne képes többcsatornás hangrögzítésre. Valamint az egyik vonalszintű bemenet lecserélése mikrofon szintre, ehhez valamilyen analóg előfok beépítése szükséges.

Lehetőség lenne 4-nél több csatornás hangrögzítésre. A tesztelt SD kártyával 4,5 MB/s-os írási sebességet biztonsággal el lehet érni, ahogy a 6.1 fejezetben láttuk. Ez alapján

akár 16 csatornát lehetne egyidejűleg rögzíteni 24 bites mélységgel és 96 kHz-es mintavételi frekvenciával. (A 3.2 fejezetből kiderült, hogy 1125 kB/s szükséges a fenti paraméterekkel 4 csatorna rögzítéséhez, tehát 16 csatorna rögzítése 4-szeres adatsebességet igényel, ami 4500 kB/s.)

A FAT32 fájlrendszer egyik tulajdonsága, hogy a maximális fájlméret 4 GB, ami a specifikáció szerinti paraméterek mellett azt jelenti, hogy az eszköz maximum 1 óra 2 perc 8 másodperc hosszú anyagot tud rögzíteni. Hosszabb anyag rögzítése megoldható lenne úgy, hogy ekkor az eszköz lezárja a fájlt, majd egy új fájlba kezdi meg a rögzítést. Ez némi szoftveres módosítást igényel.

Egy másik fejlesztési lehetőség, hogy a rögzítési paraméterek beállíthatók legyenek. A menüből ki lehetne választani, hogy hány csatornát szeretnénk rögzíteni, azokat milyen mintavételi frekvenciával és milyen bitmélységgel. Ehhez szükség lenne szoftveres és hardveres módosításra is. A szoftverben a menürendszeren és a WAV fájl fejlécének írásán kell módosítani. A hardverben az I2S FPGA modult kell úgy módosítani, hogy a FIFO-ba kerülő minták bitszélessége változtatható legyen (a bejövő 24 bites adatokból 16 biteseket képezzen). Valamint az előállított órajeleken is szükséges változtatni.

Ha az eszközből működés közben kivesszük az SD kártyát, majd visszahelyezzük, akkor nem képes további felvételek rögzítésére, mivel az SD kártyát a program futása elején egyszer inicializálja. Ezt a funkciót szintén be lehetne építeni, hogy az eszköz érzékelje az SD kártya csatlakozását és eltávolítását és ekkor elvégezze a szükséges műveleteket.

További fejlesztési lehetőség lehet, hogy a bejövő hangjeleken valamilyen előfeldolgozást végezzünk. Ezeket az FPGA-n belül lenne érdemes megvalósítani. A kijelzőn lehetne ábrázolni a jelek spektrumát, a spektrum számítása szintén az FPGA-n belül történhetne.

Ezen kívül további kényelmi funkciókat lehet még hozzáadni: például a felvételek visszahallgatása, vagy a fájlok listázásának és törlésének lehetősége. Az előbbi hardveres (FPGA-n belüli) és szoftveres módosítást is igényel, az utóbbihoz csak szoftveres módosítás szükséges.

Egy végleges áramkörön érdemes lenne a rögzített hanganyag minőségét is megvizsgálni.

8.3 Összegzés

Az elkészült áramkör teljesíti a kitűzött célokat. Képes a többcsatornás hangrögzítésre. Tehát a projektet sikeresnek tekintem. Ahogy írtam még bőven maradt továbbfejlesztési lehetőség a jövőre nézve.

Köszönetnyilvánítás

Ezúton szeretném megköszönni konzulensemnek, Dr. Fehér Bélának a TDK munkám elkészítéséhez nyújtott segítségét. A konzultációk során mindig hasznos tanácsokkal, észrevételekkel látott el, kérdéseimmel bármikor fordulhattam hozzá.

Irodalomjegyzék

[1] Xilinx: Zynq UltraScale+ Device - Technical Reference Manual
https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf
(2018. okt. 28.)

[2] BME MIT Tanszéki Munkaközösség: DIGITÁLIS JELFELDOLGOZÁS (MIT-VIMM4084-01)

[3] dr. Dabóczi Tamás - Beágyazott rendszerek

[4] Wikipedia: Pulse-code modulation
https://en.wikipedia.org/wiki/Pulse-code_modulation
(2018. okt. 28.)

[5] Wikipedia:WAV
https://en.wikipedia.org/wiki/WAV
(2018. okt. 28.)

[6] Wikipedia: *I2S*
https://en.wikipedia.org/wiki/I2S
(2018. okt. 28.)

[7] Avnet: Ultra96 Hardware User's Guide
http://www.zedboard.org/sites/default/files/documentations/Ultra96-HW-User-Guide-rev-1-0-V0_9_preliminary.pdf
(2018. okt. 28.)

[8] LOGSYS sztereó CODEC modul felhasználói útmutató
http://logsys.mit.bme.hu/sites/default/files/page/2009/09/LOGSYS_sztereo_CODEC_modul.pdf *(2018. okt. 28.)*

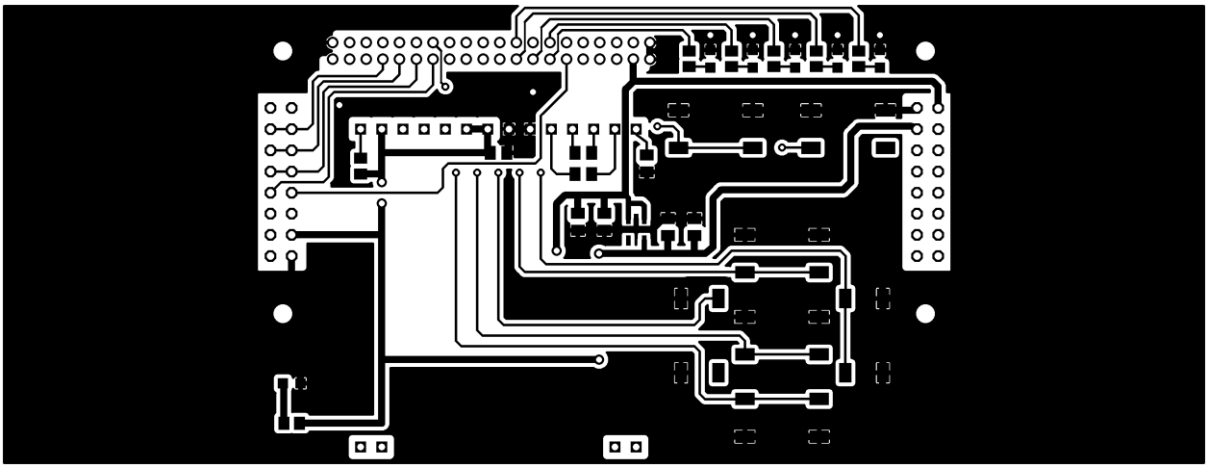
[9] Wolfson WM8569 adatlap
https://www.mouser.com/ds/2/76/WM8569_v4.0-1141826.pdf
(2018. okt. 28.)

[10] Kapcsoló pergése
https://i.stack.imgur.com/EGMeq.jpg
(2018. okt. 28.)

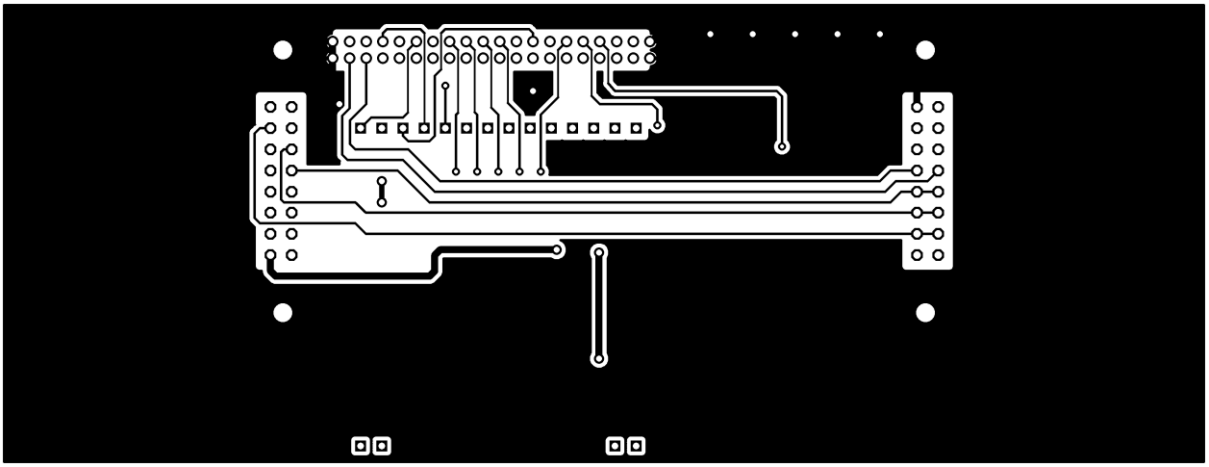
[11] Scott Larson: Debounce Logic Circuit
https://www.digikey.com/eewiki/pages/viewpage.action?pageId=4980758
(2018. okt. 28.)

[12] A C library (Linux + Arduino) to control the UC1701 128x64 monochrome LCD
https://github.com/bitbank2/uc1701
(2018. okt. 28.)

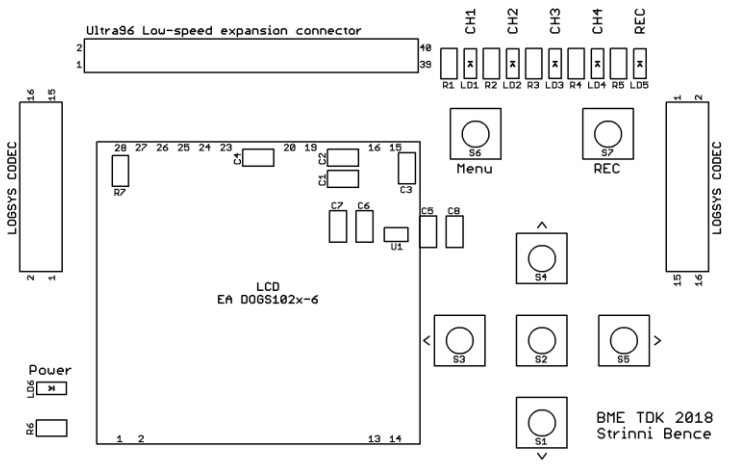
Függelékek



1. függelék: Nyákrajz (top)



2. függelék: Nyákrajz (bottom)



3. függelék: Feliratozás