Bogár István V. évf. Villamosmérnöki szak,
Faragó Ákos VI. évf. Villamosmérnöki szak,
Molnár Károly V. évf. Villamosmérnöki szak

Nyolccsatornás jelfeldolgozó rendszer fejlesztése

TDK dolgozat

Konzulens: Dr. Sujbert László
Méréstechnika és Információs Rendszerek Tanszék

Tartalomjegyzék

1.fejezet

Előszó

A digitális jelfeldolgozás létjogosultságát nem lehet megkérdőjelezni. A villamosmérnöki szakterületek széles skáláján igaz, hogy a klasszikus analóg módszereknél sokkal jobban teljesítenek a digitális megoldások. A hatékony digitális jelfeldolgozási algoritmusok mindig valamilyen központi egységet, processzort igényelnek, hiszen számításokat kell végezni a külvilág jeleit reprezentáló számértékeken. A központi egység lehet bármilyen általános célú processzor vagy logikai áramkörös (FPGA) megvalósítású. Leggyakrabban azonban egy jelfeldolgozási algoritmusok implementálására kifejlesztett specifikus jelfeldolgozó processzort (DSP) használunk.

A központi egység számára a digitális jeleket a rendszert körülvevő fizikai világ jeleiből állítjuk elő érzékelők és A/D átalakítók segítségével. Ahhoz pedig, hogy a digitális jel a fizikai világban is megjelenjen, D/A átalakítókra és esetleges további átalakítókra (pl. elektromechanikus) van szükség.

A jelfeldolgozó rendszerek közül sok csak egy bementtel és egy kimenettel rendelkezik (SISO rendszerek), de gyakran van szükség sok csatorna jeleinek egyidejű, real-time feldolgozására. (MIMO rendszerek). Néhány szakterület, ahol ilyen többcsatornás jelfeldolgozó rendszereket használatosak, a teljesség igénye nélkül:

- Audiotechnika
- Aktív zajcsökkentés
- Telekommunikációs megoldások
- Szonár rendszerek
- Radar rendszerek
- Visszaverődéses szeizmológia
- Computed Tomography (CT)

A dolgozat egy ilyen sokcsatornás jelfeldolgozó rendszer megtervezéséről és megvalósításáról szól. Olyan rendszer képzeltünk el, amelynek segítségével a fent említett különböző jelfeldolgozó alkalmazások implementálhatóak a gyakorlatban. Egy olyan hardver és szoftver környezetet kívántunk létrehozni, amely szilárd alapot biztosít az alkalmazásnak olyan módon, hogy az alkalmazásfejlesztő elméleti szakembernek ne kelljen foglalkoznia a hardver megvalósítás részleteivel.

Dolgozatunk a Méréstechnika és Információs Rendszerek Tanszék DSP laboratóriumában elvégzett munkánkat mutatja be. Elsőként a már létező, piaci forgalomban lévő hasonló eszközöket vizsgáltuk. Ezek után kerül sor az általunk tervezett rendszer specifikációjára. Részletesen ismertetjük a hardver és a szoftver tervezésének menetét, valamint bemutatjuk a készen felhasznált eszközöket. Rendszerünk tesztelésére egy példa alkalmazást implementáltunk. Ennek elméleti áttekintése és a megvalósítás menete is témája dolgozatunknak.

2. fejezet

Bevezető

2.1. Általános megfontolások

A digitális jelfeldolgozás fontos építőköve az, hogy számítási műveleteket végzünk a fizikai világ jeleit reprezentáló számértékeken, azaz a digitális jeleken. A különböző jelfeldolgozási algoritmusok futtatásához tehát mindig valamilyen számítási műveleteket végző egység szükséges.

A jelfeldolgozási rendszerek azonban a környezet valós jeleit kívánják feldolgozni, és ezekbe is szeretnének esetlegesen beavatkozni, tehát szükség van a valós jeleket digitális jellé, valamint a digitális jeleket valós jellé alakító egységekre a műveletvégző egység mellett. Az előbbi átalakítás érzékelőkkel és AD átalakítókkal történik, az utóbbi DA átalakítók és beavatkozók segítségével.

A jelfeldolgozási algoritmusok implementálására több megoldás kínálkozik. A legkézenfekvőbb megoldás az, hogy az algoritmusokat személyi számítógépen, valamilyen jól ismert környezetben (pl. Matlab) futtatjuk. E „felhasználóbarát" megoldás mellett az szól, hogy a programozás kényelmes, megszokott környezetben zajlik, és nem igazán kell törődni a hardver megvalósítás részleteivel. A problémát az jelenti, hogy a környezet fizikai jeleihez való real-time hozzáférés nehézkes, inkább off-line feldolgozás esetén jön szóba ez a megoldás. További problémát jelent, hogy az általános célú processzor műveletvégző egységei csak hagyományos számítási műveleteket végeznek, így a tipikus digitális jelfeldolgozási algoritmusok során felmerülő számítási műveletek végrehajtása esetleg kissé körülményes lesz.

A „hardverközeli" megoldás az, ha a számítási műveleteket végző egységet FPGA áramkör valósítja meg. Ez úgy történik, hogy a gyakran végrehajtásra kerülő műveleteket logikai kapukkal felépített hálózat értékeli ki. Ezen hálózatok kifejezetten egy művelet végrehajtására alkalmasak, viszont különböző párhuzamosítások segítségével a hálózat fokozott bonyolódása mellett a végrehajtási sebesség szinte tetszőleges mértékig növelhető. Ez a megoldás nagyon jó sebességkritikus real-time alkalmazások esetén, de ellene szól rugalmatlansága, és a tervezés nehézkessége.

Az általunk is választott megoldás ötvözi a fenti lehetőségeket, magába foglalva mindkettő jó tulajdonságait. A DSP (Digital Signal Processor) egy olyan viszonylag egyszerű processzor, amely a digitális jelfeldolgozás tipikus műveleteinek elvégzésére van optimalizálva. Egy olyan célprocesszor, amely egyszerű és olcsó, de a jelfeldolgozási algoritmusokat egy komplex mai processzornál gyorsabban valósítja meg.

A DSP program fejlesztése személyi számítógépen futó fejlesztői környezetben történik. A programozási nyelv általában gépi nyelv (assembly), amely minden gyártó esetében más jellegű, és természetesen processzorspecifikus. Ma már általában a DSP C nyelven történő programozására is van lehetőség, de legalább a programok

számítási idő kritikus lényegi részeit gépi nyelven szokás megírni, hiszen ekkor lehet maximálisan kihasználni a hardver nyújtotta lehetőségeket.

A jelfeldolgozási algoritmusok gyors végrehajtása annak köszönhető, hogy a DSP hardver szinten támogatja a jelfeldolgozási algoritmusok tipikus műveleteinek elvégzését, tehát olyan specifikus egységei vannak, amelyek funkcióit egy általános célú processzorban csak bonyolultan valósíthatók meg. (Példák: cirkuláris buffer, Multiply and Accumulate utasítás, bit-reversed üzemmód.) A gyorsaság másik oka az, hogy a leggyakoribb műveletek párhuzamos elvégzésére is van lehetőség az FPGA áramkörök mintájára. A párhuzamosítást segíti a Harvard−architektúra is, amely szétválasztja a memóriát kód-, illetve adat területre.

A DSP alapú jelfeldolgozási rendszer tehát egy egészséges kompromisszumot jelent a két előzőleg ismertetett megoldás között, megtartva mindkettő jó tulajdonságait.

A digitális jelfeldolgozó rendszerek a környező fizikai világgal különböző számú bemeneti és kimeneti jelúton keresztül vannak kapcsolatban. A továbbiakban egy bemeneti-kimeneti jelútpárt nevezünk csatornának. Ez az audiotechnikában szokásos elnevezés, amely megtévesztő lehet, hiszen az egyes bementi és kimeneti jelutak nincsenek eredendően egymáshoz rendelve, ezek teljesen függetlenül is működhetnek.

Többcsatornás rendszerekre akkor van szükség, ha több bemeneti jelet párhuzamosan és valós időben kívánunk feldolgozni, illetve ha több kimenetet kezelünk. Minél komplexebb egy alkalmazás, annál valószínűbb, hogy több csatornára van szükség. Például az aktív zajcsökkentést végző rendszerek annál hatékonyabbak, minél több mikrofon jelét képesek egyidejűleg feldolgozni. Többcsatornás rendszerre van szükség különböző hibrid szabályozási rendszerek megvalósításához is.

Vannak olyan alkalmazások is, amelyek jellegükből adódóan többcsatornásak, például az audiotechnikában használatos keverők. Ezen eszközök bemenetei a különböző hangforrások (pl. hangszerek) hangjelei, amelyek egyidejűleg különböző csatornákon mikrofonok segítségével kerülnek a keverőbe. A Master kimeneten a bemeneti csatornákból összekevert kész hangzás áll elő. Egy másik általában megvalósított kimenet a Monitor, amelyen tipikusan fejhallgatóval lehet meghallgatni az egyes csatornák jeleit. A digitális megoldás azért kedvező, mert különböző effektek valósíthatók meg az egyes bemeneti jeleken, például szűrések, kiemelések, visszhang, stb. Ez a példa is érzékelteti a DSP alapú megoldás azon előnyét, hogy a rendszert nagymértékben flexibilissé teszi. Egy ilyen rendszer teljesen általános célú, csak az alkalmazói szoftver dönti el, hogy ugyanazt az eszközt milyen különböző alkalmazások implementálása használjuk. Ez azt is jelenti, hogy a megvalósított alkalmazás a jövőben mindig továbbfejleszthető lesz a hardver változtatása nélkül.

Célunk pontosan egy ilyen általános célú rendszer megalkotása volt. Ezen hardver eszköz és a működtető szoftver egységesen képez egy olyan üzembiztos környezetet, amelybe az alkalmazások széles köre ágyazható be.

2.2. Irodalomkutatás

A piacon természetesen sok cég kínál olyan eszközöket, amelyek alkalmasak a fenti többcsatornás alkalmazások implementálására. Ennek megfelelően azelőtt, hogy megkezdtük volna saját eszközünk tervezését, irodalomkutatást végeztünk. Elsődleges forrásunk az Internet volt, a különböző DSP alapú eszközöket gyártó cégek honlapjait vizsgáltuk. Olyan eszközöket kerestünk, amelyeken a fenti többcsatornás

jelfeldolgozó alkalmazások implementálhatók lennének. Az alábbiakban ismertetjük a látókörünkbe került termékeket, amelyek főbb tulajdonságait az I. és a II. táblázatban foglaltuk össze.

2.2.1. Analog Devices [1]

Az Analog Devices igen sok fejlesztői kártyát gyárt (ADI terminológiával Evaluation Board). Ezek a kártyák általában egy-egy ADI termék köré vannak építve, ennek megfelelően a DSP alapú kártyákon a processzoron kívül csak egy kétcsatornás codec-et találunk.(Codec: kóder-dekóder, egy integrált áramköri tokon belül megvalósított AD és DA átalakító.) Vannak továbbá olyan AD illetve DA kártyák, amelyeken vagy sok bemenettel, vagy sok kimenettel rendelkeznek. Olyan kártya nincs, amelyen elegendő számú kétirányú csatorna van. Emiatt nem lehet készen vett gyári kártyák egyszerű csatlakoztatásával megfelelő többcsatornás rendszert építeni.

2.2.2. Innovative DSP [2]

Ez az amerikai cég jelfeldolgozó megoldásokat kínál különböző ipari vásárlók számára. Egyedi megoldásokat és kész rendszereket is kínálnak, igen magas színvonalon. A terméklistán két többcsatornás jelfeldolgozó kártyát találunk. Mindkettő a nagy számítási kapacitással rendelkező Texas Instruments TMS320C6711 DSP-n alapszik.

A Toro nyolc bemenetű, nyolc kimenetű kártya, amely 16 bites szukcesszív approximációs AD, illetve DA átalakítókkal rendelkezik. A cég nagyágyúja a Delfín fantázianevű termék. Ezen viszonylag új kártya 32 bemenettel és 6 kimenettel rendelkezik, szigma-delta AD és DA átalakítói 24 bitesek. A maximális mintavételi frekvencia 192 kHz.

Mindkét kártya tökéletes lenne többcsatornás jelfeldolgozó algoritmusok implementálására. A problémát természetesen a kártyák ára jelenti. A disztribútornál csak a Toro kártya áll rendelkezésre, ennek ára néhány kiegészítővel 12000 USD.

2.2.3. Bittware [3]

Szintén amerikai cég, DSP-n alapuló beágyazott rendszerekkel foglalkoznak. Elsősorban az Analog Devices SHARC DSP családjának tagjaira épülnek termékeik. A mi szempontunkból egyedül az Audio PMC+ nevű kártya érdekes, ami egy nyolccsatornás 96 kHz mintavételi frekvenciával dolgozó eszköz. Mint a neve is mutatja, kifejezetten audio alkalmazásokhoz ajánlják. A kártyán két ADSP-21065L processzor található. Az analóg audio interfészen kívül rendelkezik digitális audio interfésszel is. Az eszköz különböző buszrendszerek segítségével csatlakoztatható más kártyákhoz, vagy PC-hez is. A kártyát kész alkalmazói szoftverrel együtt kínálják, amely lényegében az adatok kódolását és dekódolását végzi, tehát ezt a kártyát arra a célra szánják, hogy más, esetleg még nagyobb számítási kapacitással rendelkező DSP kártyák számára készítse elő az adatokat. Természetesen lehetőség van arra is, hogy a kártya önállóan implementáljon jelfeldolgozói algoritmusokat, de alapvetően nem erre a célra tervezték. Ezért valamint magas ára miatt nem tekinthető optimális választásnak.

2.2.4. Pentek Inc.[4]

Ez a cég is sokféle DSP alapú terméket kínál, de a kínált megoldások kissé eltérőek a fenti cégek megoldásaitól. Ugyanis ez a gyártó külön árusít DSP kártyákat és IO (input - output) kártyákat, amelyek tetszőlegesen összepárosíthatók VME busz segítségével. Ilyen VME busz csatlakozó a gyártó minden kártyáján megtalálható.

Egy megfelelő többcsatornás jelfeldolgozó rendszer megépíthető lenne egy 4265-os és egy 4284-es sorszámú kártyából. Így egy Texas Instruments TMS320C40 alapú nyolc bemenetű, nyolc kimenetű rendszert kapnánk. Az AD és a DA átalakítók felbontása 16 bit, a maximális mintavételi frekvencia 50 kHz. Ennek a rendszernek az ára szintén nagyon magas, ami nem áll arányban a kész rendszer teljesítményével, ami egyáltalán nem kiemelkedő.

2.2.5. Domain Technologies [5]

Ez a cég a konkurensekhez képest szerényebb termékválasztékot kínál. Viszonylag olcsó, négycsatornás DSP alapú kártyákat készítenek. A kínálatban szereplő eszközök vagy Motorola vagy Texas Instruments DSP alapúak. Az alacsony árnak köszönhetően elég kevés kiegészítő funkciót valósítottak meg a kártyákon. Valószínűleg ez az oka, hogy az eszköz csak USB porton keresztül tud kommunikálni más eszközökkel, illetve hoszt számítógéppel. Az eszközök bővítése, sajnos, nem igazán megoldott, így a különböző alkalmazói szoftverek telepítése nehézkesnek tűnik.

2.2.6. Causal Systems [6]

A Causal Systems ausztrál illetőségű cég, amely kizárólag aktív zajszűrést megvalósító hardver és szoftver termékeket kínál. A kínálat nem túl széles, valójában az EZ-ANC II nevű sokcsatornás rendszer és a hozzá tartozó szoftver eszközök jelentik a teljes termékskálát. Dolgozatunk szempontjából viszont éppen ez a rendszer érdekes. Az EZ-ANC tíz csatornával rendelkezik, amelyeket öt AD1847-es codec segítségével kezelnek. Az eszköz vezérlését két DSP végzi: egy ADSP-21062 SHARC lebegőpontos, valamint egy ADSP-2181 fixpontos processzor (co-processor üzemmódban). A rendszerbe FLASH EPROM külső memóriát is beépítettek. Az EZ-ANC II személyi számítógéppel szabványos RS-232 porton keresztül érintkezik. A rendszerhez tartozó PC-s alkalmazás szerves része a terméknek, mert ennek segítségével lehet beállítani a működés paramétereit. A tíz kimenet egyike képes jelgenerátorként is működni. Két bemeneti csatornán pedig FFT analizátor funkciót képes végrehajtani. Ezt a rendszert, mint neve is mutatja (ANC, Active Noise Control) alapvetően aktív zajcsökkentés megvalósítására tervezték. Tulajdonképpen mindig csak két algoritmus (xLMS és uLMS) megvalósítását végzi, a PC-s alkalmazás arra szolgál, hogy ezen algoritmusok futási paramétereit beállíthassuk. A sok csatornát más célra, az eszközt másféle algoritmus implementálására nem lehet használni. Emiatt, bár számítási kapacitása és csatornaszáma képessé tenné rá, nem alkalmas arra a feladatra, hogy általános jelfeldolgozó rendszerként használjuk.

2.2.7. Összefoglalás

A lista természetesen nem teljes, hiszen ezeken a gyártókon kívül még sok más cég is foglalkozik különböző DSP alapú hardver és szoftver rendszerek fejlesztésével és árusításával. A lényeges következtetések azonban levonhatók ebből az áttekintésből is. Általában igaz, hogy az elérhető árú kártyák nem rendelkeznek kellő számú csatornával. A kellően sok csatornával rendelkező rendszereknek viszont kivétel nélkül nagyon magas az áruk. Ez annak köszönhető, hogy a sokcsatornás rendszerekbe mindig sok más magas szintű szolgáltatást is beépítenek. Például: általános célú FPGA áramkörök, memóriabővítés (nem ritka a kiegészítő 32 Mbyte RAM), különböző buszillesztések, több processzor. Ezek természetesen a rendszer összköltségét nagyban növelik.

A vizsgálatból azt a következtetést vontuk le, hogy van létjogosultsága annak, hogy egy saját sokcsatornás jelfeldolgozó rendszert tervezzünk és valósítsunk meg. A tanszék egyik már meglévő DSP kártyáját felhasználva terveztünk egy olyan rendszert, amely nyolc csatornával rendelkezik, kielégítve ezzel a legtöbb jelfeldolgozó algoritmus igényeit. Ezen rendszer költsége a hasonló csatornaszámú rendszerek árának töredéke.

A következő fejezetben a rendszerben felhasznált EZ-KIT kártyát ismertetjük. Ez után az általunk tervezett nyolccsatornás rendszer részletes ismertetésére kerül sor. Két külön fejezet tárgyalja a hardver, illetve a szoftver tervezés menetét és az ehhez tartozó megfontolásokat.

Végül egy konkrét jelfeldolgozói alkalmazás implementálásával mutatjuk be a teljes rendszert működés közben.

Gyártó	**Eszköz neve**	**DSP típusa**	**DSP-k száma**	**Mintavételi frekvencia**	**Csatornák száma**
Innovative Dsp	Toro	TMS320C6711	1	250 kHz	8
Innovative Dsp	Delfín	TMS320C6711	1	192 kHz	32 be, 6 ki
Bittware	Audio PMC+	ADSP-21065L	2	96 kHz	8
Pentek	4265 és4284	TMS320C40	1	50 kHz	8
Domain Technologies	Audio4-USB	Motorola DSP56307	1	48 kHz	4
Causal Systems	EZ-ANC II	ADSP-21062, ADSP-2181	2	81Hz – 32 kHz	10

I. táblázat. Eszközök összehasonlítása 1.

Gyártó	Eszköz neve	AD/DA típusa	ADDA bitszáma	Kiegészítők
Innovative Dsp	Toro	Sz. Approx	16	32 Mbyte RAM, PCI busz
Innovative Dsp	Delfín	Szigma-Delta	24	32 Mbyte RAM, PCI busz
Bittware	Audio PMC+	Nem ismert	24	16 Mbyte RAM
Pentek	4265 és 4284	Nem ismert	16	VME busz
Domain Technologies	Audio4-USB	Szigma-Delta	16	-
Causal Systems	EZ-ANC II	Szigma-Delta	16	Jelgenerátor kimenet, 2 csatornás FFTanalizátor

II. táblázat. Eszközök összehasonlítása 2.

3. fejezet

Az ADSP-21061 EZ-KIT Lite fejlesztői kártya

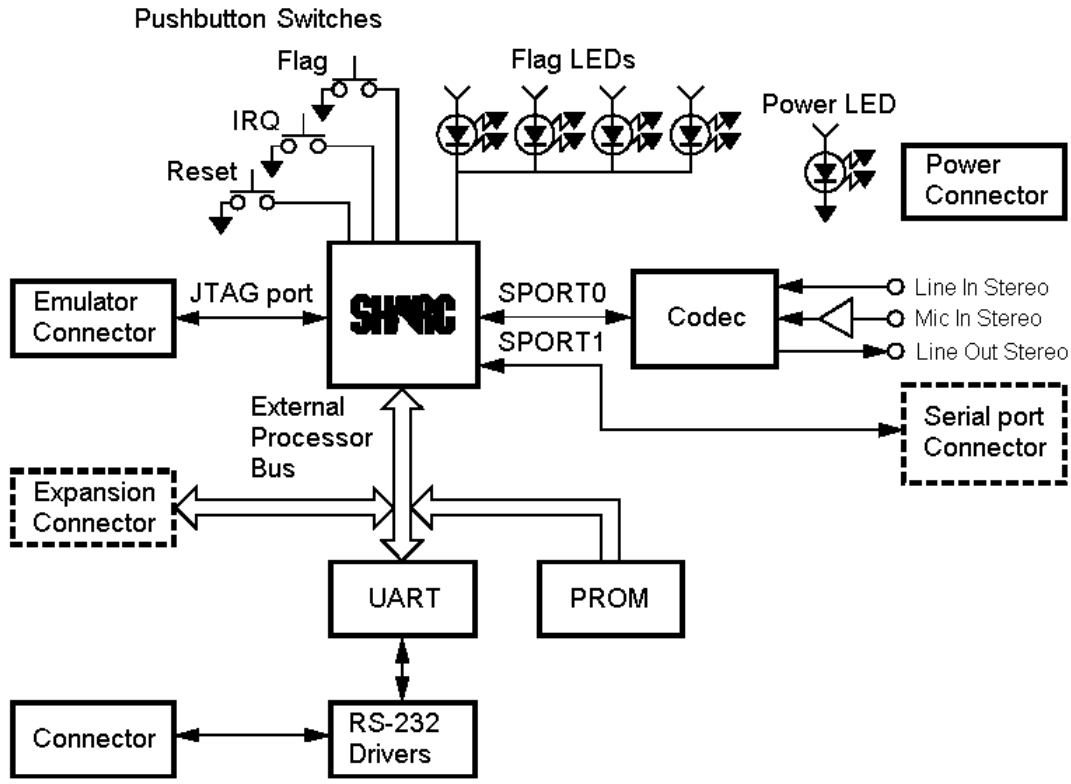
3.1. Bevezető

Az általunk tervezett többcsatornás jelfeldolgozó rendszer egyik fő építőeleme az ADSP-21061 EZ-KIT Lite fejlesztői kártya. Az Analog Devices Inc. (a továbbiakban: ADI) legtöbb termékéhez kínál ilyen vagy ehhez hasonló fejlesztői kártyát. Az ADSP-21061 EZ-KIT Lite az ADSP-21061 jelfeldolgozó processzor köré van építve. Ezek a kártyák mentesítik a vásárlót a tesztáramkör építése alól, ami például egy lebegőpontos DSP esetében négyrétegű nyomtatott áramkör tervezését jelenti. Ehelyett a vásárló egy olyan kártyát kap kézbe, amelynek segítségével az adott termék rögtön hozzáférhető és működés közben kipróbálható. Ez nagymértékben elősegíti az adott alkatrész gyors megismerését.

Az ADSP-21061 EZ-KIT Lite használata mellett szól az is, hogy kétféleképpen is csatlakoztatható személyi számítógéphez. Erre az egyik lehetőség a szabványos RS-232 port, amelyen keresztül a PC-n megírt programok letölthetők a DSP-re. A másik a JTAG emulátor csatlakozó, amely jelentősen megkönnyíti a DSP programjának fejlesztését, mert segítségével a fejlesztői környezet (VisualDSP) lehetőséget biztosít a hagyományos hibakeresési (debug) módszerek alkalmazására. Lehetőség nyílik a program futásának részletes követésére, töréspontok helyezhetők el, lehetséges az utasítások léptetett végrehajtása, a memóriaterületek valamint a processzor regisztereinek tartalma folyamatosan monitorozható.

3.2. A kártya felépítése [7]

Az ADSP-21061 EZ-KIT Lite fejlesztői kártya elsődlegesen arra szolgál, hogy az ADSP-21061 processzor minél több funkciója elérhetővé váljon, egyszerűen hozzáférhető legyen. A kártya felépítés az 3.1. ábrán látható.



3.1. ábra. Az EZ-KIT Lite kártya felépítése

Kártyán négy LED és három nyomógomb található. A LED-eket a DSP FLAG lábai hajtják meg. A három nyomógomb közül egy a RESET lábra, egy a FLAG1 lábra, egy pedig az IRQ1 lábra van illesztve. Ez utóbbi megnyomásakor megszakításkérés történik, amely lehetőség nagyon jól kihasználható a különböző alkalmazások implementálásakor.

A kártyán található portok egyike a JTAG emulátor port, amelyen keresztül hibakeresésre (debugging) nyílik lehetőség. A JTAG port használatáról bővebben a 3.4.2.-es pontban térünk ki.

A fejlesztői kártyán található szabványos RS-232 porton keresztül aszinkron soros kommunikációra van lehetőség. A JTAG porton kívül ez a másik lehetőség személyi számítógéphez való csatlakozásra. A processzor nem rendelkezik közvetlenül RS-232 szabvány szerinti lábakkal, ezért a port megvalósítása egy PAL 16V8 logikai áramkörrel, egy PC16550D típusú UART (Universal Asynchronous Receiver/Transmitter) IC-vel, valamint egy AD232AARN típusú RS-232 driverrel történik. A processzor a PAL áramörön keresztül vezérli a kommunikációt, amely a DSP külső memóriabuszára (extrenal bus) van illesztve.

A DSP külső memóriabuszára a logikai áramkörön kívül egy gyárilag beégetett EPROM-ot is illesztettek. A processzor ennek segítségével boot-ol hardver RESET után.

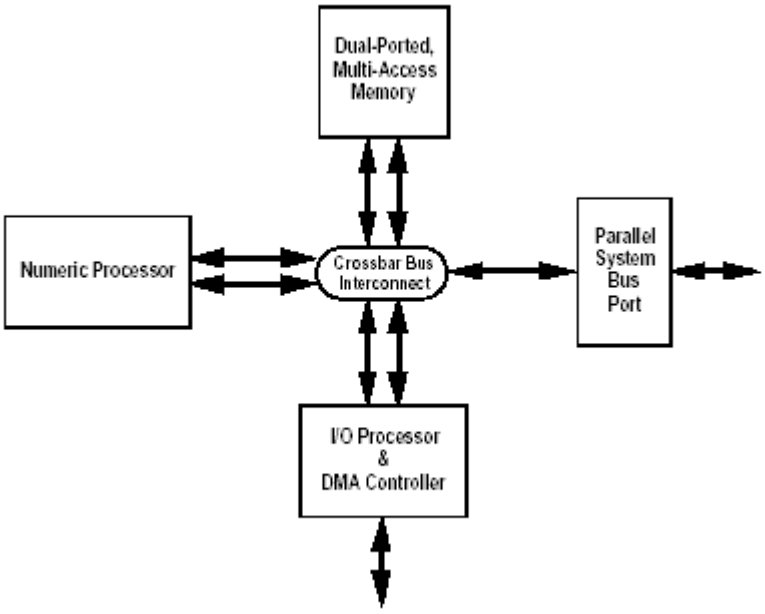
Egy AD1847-es sztereó codec-et is elhelyeztek a kártyán, amelyhez analóg oldalról két sztereó jack dugóval lehet csatlakozni (egy kimenet, egy bemenet). Ez két bementi és két kimeneti csatornát jelent. (Megjegyzés: a teljes nyolccsatornás rendszer csatornaszámába ezt a két csatornát nem számoljuk bele.) Az AD1847-es codec a DSP nulladik soros portjához (SPORT0) van illesztve.

A fejlesztői kártyán a DSP azon lábaihoz is vannak kivezetések, amelyekhez nincs alkatrész illesztve. Ezek a lábak akkor válnak elérhetővé, ha tüskesorokat forrasztunk a megfelelő helyekre. A nyolccsatornás rendszerben az egyes számú soros portot (SPORT1) használjuk a codec-kártyával történő kommunikációra. Ennek a portnak a lábai is a fenti módon vannak az EZ-KIT kártyára kivezetve, ilyen módon tüskesor felforrasztásával váltak elérhetővé. Ezen kívül még két tüskesort forrasztottunk a fejlesztői kártyára, melyek segítségével a DSP egyes FLAG lábait, illetve a külső memóriabusz illesztéséhez szükséges lábak kivezetését oldottuk meg. Az EZ-KIT kártya és a saját fejlesztésű codec-kártya e tüskesorokon keresztül csatlakozik egymáshoz. Ilyen módon a két kártya fizikailag egymás felett helyezkedik el, a függelék 19. oldalán látható módon. Az egyes számú soros port kivezetésére azért van szükség, mert ezen a porton keresztül kommunikál a DSP a codec-ekkel. A DSP FLAG2 lábát a codec-ek RESET lábához, a FLAG3-at a SELECT DEVICES lábakhoz vezettük. A külső memóriabusz lábaira a codec-kártyán található CPLD áramkör memóriaillesztése miatt van szükség.

3.3. ADSP -21061 [8]

A fejlesztői kártya, valamint egyben a nyolccsatornás rendszer legfontosabb eleme a DSP, a jelfeldolgozó processzor. Az ADSP-21061 a SHARC DSP-család egyik tagja, amely az ADI egyik leginkább elterjedt és széles körben használt termékcsaládja. A SHARC betűszó a Super Harvard ARhitecture Computer rövidítése. Ez a felépítés a DSP-k körében klasszikusnak számító Harvard-architektúra továbbfejlesztett változata. A Hardvard-architektúra a személyi számítógépek processzorainak Neumann-architektúrájával szemben megkülönböztet kód-, illetve adatmemóriát. A két memóriaterület külön buszon érhető el, így lehetőség van párhuzamos használatukra. Ez a lehetőség műveletek ciklikus elvégzésekor nagymértékben gyorsítja a program futását.

A szuper Harvard-architektúra (ld. 3.2. ábra) esetében már négy független busz van: két adatbusz, egy utasításbusz és egy IO busz. A SHARC processzorok egy órajelütemben egy utasítást hajtanak végre, ezt a tokon belüli megvalósítású utasítás-cache teszi lehetővé.



3.2. ábra. SHARC architektúra

Az ADSP-21061 a SHARC család első tagjának, az ADSP-21000-nek egy továbbfejlesztett változata. A processzor mag (processor core) megegyezik a két típusnál, de a 21061-es néhány fontos kiegészítővel látták el. Ezek a következők: SRAM, IO processzor, IO busz. Ezen egységek mindegyike tokon belül van megvalósítva.

A 21061-es nem a legfejlettebb modell a ma kapható DSP-k között, de sebessége és számítási pontossága kielégítő a legtöbb átlagos jelfeldolgozási alkalmazás szempontjából. Az iparban széles körben használják ezt a DSP-t, mert viszonylag alacsony ára mellett az összes olyan követelményt teljesíti, ami egy lebegőpontos jelfeldolgozó processzortól elvárható. Mindazonáltal várható, hogy a közeli jövőben ki fogja szorítani a piacról az ADSP-21065L processzor, amely a SHARC család legújabb és máris népszerű tagja. Ez a jövőbeli processzorváltás várhatóan nem okoz majd nagy problémát, hiszen a két DSP felfelé kódkompatibilis, azaz a 21061-esre írt programok hiba nélkül futnak a 21065L-en is.

A távolabbi jövőben várható, hogy a SHARC család tagjait fel fogják váltani az újabb DSP családok, például a ma már kapható, de még ritkaságszámba menő TigerSHARC család tagjai.

3.3.1. Teljesítményadatok

A CMOS megvalósítású ADSP-21061 processzor maximális órajelfrekvenciája 50 MHz. Mivel a DSP órajelütemenként egy utasítást hajt végre az utasítás cache segítségével, így az utasításvégrehajtási sebesség 50 MIPS (Million Instructions Per Second).

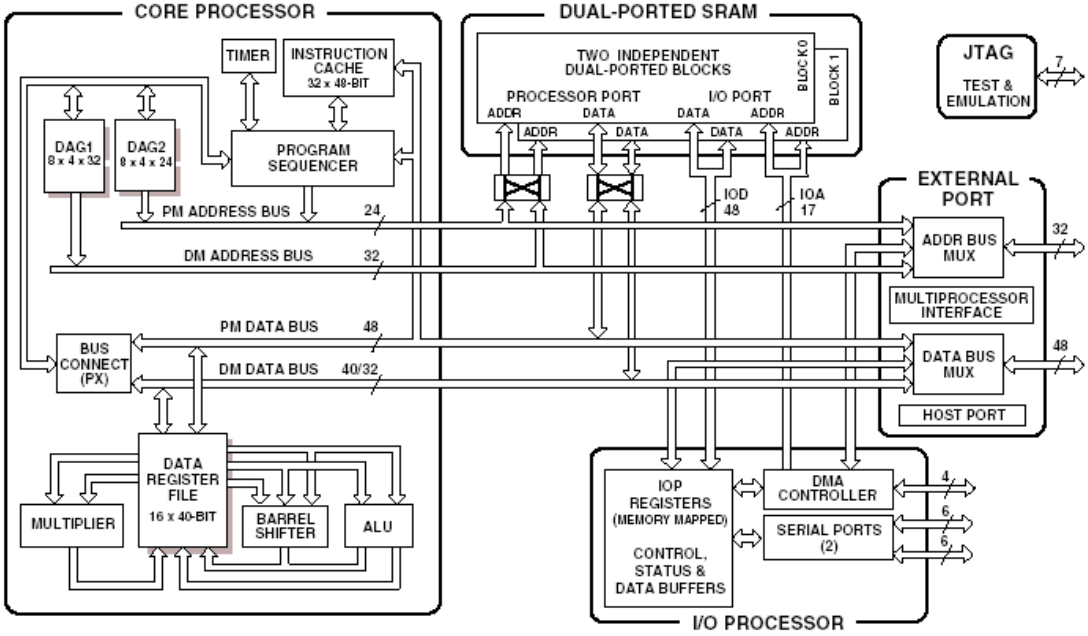
A processzor teljesítményét jellemző benchmark adatokat a III. táblázat ismerteti.

1024 pontos FFT számítás (Radix 4)	0,37 ms	18221 ciklus
FIR szűrés (együtthatónként)	20 ns	1 ciklus
IIR szűrés (per biquad)	80 ns	4 ciklus
Osztás művelet (y/x)	120 ns	6 ciklus
Inverz gyökvonás művelet($1/\sqrt{x}$)	180 ns	9 ciklus
DMA átvitel sebessége	300 Mbyte/sec	

III. táblázat. ADSP-21061 Benchmark adatok (f_{CLK} = 50 MHz)

3.3.2. Felépítés

Az ADSP-21061 felépítésen a 3.3. ábrán látható.



3.3. ábra. Az ASDP-21061 felépítése

A processzor fő részei a következők:

- **Műveletvégző egység**

A műveletvégző egység kétféle lebegőpontos számábrázolási formátum használatát támogatja, a 32 bites IEEE single-precision floating-point szabványos formátumot, illetve a 40 bites kiterjesztett (extended precision) formátumot. Lehetőség van 32 bites fixpontos formátum használatára is. A műveletvégző egység részei: ALU (Arithmetic and Logic Unit), szorzó egység, shifter egység. Ezek párhuzamosan vannak megvalósítva, tehát lehetőség van egyidejű használatukra a műveletvégzés gyorsítása érdekében.

Az egység számítási kapacitása általános esetben a fentebb említett 50 MIPS. Ez azonban egyes tipikus jelfeldolgozási számítások során a műveletek párhuzamos végrehajtás miatt megnövekszik. Az ilyen módon elérhető maximális számítási kapacitás 120 MIPS vagy 120 MFLOPS (Million Floatingpoint Operation Per Second). Ez utóbbi mérőszám is gyakran használatos, a processzor lebegőpontos számítási műveletekre vonatkoztatott számítási sebességet jellemzi. Mivel az ADSP-21061-es alapértelmezésben lebegőpontos számokon hajtja végre az utasításokat, esetében a két mérőszám megegyezik.

- **Adatregiszter blokk**

32 általános célú regiszter áll rendelkezésre a különböző adatmozgatási és számítási feladatokhoz. Ez a 32 regiszter két 16-os csoportra oszlik, az elsődleges (primary) és a másodlagos (alternate) regiszterblokkokra. A két blokk közül mindig csak az egyik aktív, a másik nem hozzáférhető. A blokkok közti váltás egy utasítással történik. Ezt a funkciót tipikusan a megszakítási rutinok használják ki, olyan módon, hogy az IT rutin csak a másodlagos regiszterblokkot használja, a főprogram pedig az elsődlegest. Ekkor elkerüljük azt, hogy a két rutinnak közös regisztereken kelljen osztoznia. Tekintve, hogy a megszakítási rutinok a főprogramot tetszőleges helyen szakíthatják meg, a közös regiszterhasználat előre nem tervezhető konfliktusokat okozna. Ez a probléma a másodlagos regiszterblokk hiányában csak nehézkesen és nagy overhead árán lenne kivédhető a regisztertartalom rendszeres memóriába mentésével.

- **Utasítás cache**

A Harvard-architektúra nyújtotta előnyök ezen egység segítségével használhatók ki maximálisan. Az utasítás cache-nek köszönhető, hogy utasítások ciklikus végrehajtása esetén három párhuzamos adatmozgatás és két számítási művelet hajtódik végre egy órajelcikluson belül. Ezek részletesebben: az adatmemória és a kódmemória buszain egy-egy adatot olvasunk be regiszterekbe. (A kódterületen is lehet adatok is tárolni.) A szorzó és az összeadó egyidejűleg hajt végre egy-egy műveletet regiszterekben tárolt adatokon. A következő végrehajtandó utasítás kódját pedig az utasítás cache szolgáltatja.

- **Adatcímző egységek**

Az adatcímző egységek egyike a kódterülethez, a másik az adatterülethez tartozik. Ezen egységek hardver szinten támogatják cirkuláris bufferek megvalósítását. Egy egyszerű memóriatömb azáltal lesz cirkuláris buffer, hogy a bufferen belüli címeket az adatcímző egység speciális módon számítja ki. Ez úgy történik, hogy a cirkuláris bufferhez egy mutató regisztert rendel hozzá, amely segítségével a címzés történik. A mutató regiszter a cirkuláris buffer báziscíméhez képesti eltolást tárolja, értéke maximálisan a buffer hossza lehet. Amennyiben a mutató regiszter olyan értékre módosulna, amely nagyobb, mint a buffer hossza, akkor a regiszter tartalma ezen értéknek a buffer hosszával osztott törtrésze lesz. Ilyen módon egy körkörös vagy cirkuláris buffer jön létre, amely nagyon kedvező néhány tipikus jelfeldolgozási algoritmus (FFT, FIR szűrés) megvalósításakor.

- **SRAM memória**

A tokon belül (vagy on-chip) megvalósított 1 Mbites SRAM memória miatt nem kell külön külső memóriát biztosítani a processzor működéséhez. Természetesen amennyiben nem lenne elegendő ennyi memória, lehetőség van további külső

memória illesztésére. Az SRAM memóriaterület a Harvard-architektúrának megfelelően két részre van osztva, adat-, illetve kódterületre. Ez a hagyományos elrendezés, de nincs semmiféle megkötés abból a szempontból, hogy milyen típusú adatokat tárolunk az egyes területeken. Az optimális működési sebesség elérésének érdekében mindig az adott alkalmazás szempontjából mérlegelni kell, hogy milyen elrendezéssel érhető el a lehető legtöbb párhuzamos memóriaművelet. Például FIR szűrés esetén a bemeneti adatokat és a szűrőegyütthatókat külön memóriaterületen érdemes tárolni.

- **Külső memória és periféria interfész**

Külső memória illesztésére a külső buszon (external bus) keresztül van lehetőség. Erre a külső buszra a belső buszok multiplexálva csatlakoznak, tehát egy külső eszköz mind adat, mind kód memóriaterületre illeszthető. Ilyen módon a DSP-hez külső memórián kívül tetszőleges periféria illeszthető. Ezen az interfészen keresztül csatlakoztatható például FPGA áramkörhöz, vagy többprocesszoros működés esetén (multiprocessing) másik DSP-hez.

- **Hoszt interfész**

A hoszt interfész nyújt lehetőséget szabványos mikroprocesszoros buszokhoz való illesztésre. Ezen az interfészen keresztül a DSP egyszerűen csatlakoztatható személyi számítógéphez.

- **DMA vezérlő**

A DMA (Direct Memory Access) vezérlő nagy tömegő blokkos adatmozgatások esetén használatos. Ekkor a processzor magjának igénybevétele nélkül a DMA vezérlő nagy sebességű adatmozgatást végez a belső SRAM memória és valamely más eszköz között. Ezen eszközök a következők lehetnek: külső memória, külső periféria, hoszt processzor, soros port. Ezen kívül DMA kapcsolat létesíthető a külső memória és más külső eszközök között is. A nyolccsatornás rendszerben is a DMA átvitellel történik az adatforgalom a soros port és a belső memória között.

- **Soros portok**

A DSP két szinkron soros porttal rendelkezik, amelynek segítségével különböző digitális perifériákhoz való csatlakozásra nyílik lehetőség. A soros portok sebessége széles tartományban mozoghat, maximálisan 40 Mbit/sec.

- **JTAG interfész**

Az IEEE JTAG 1149.1 szabványnak megfelelő port a DSP tesztelésére szolgál. Ezen port segítségével emulálásra van lehetőség, ami azt jelenti, hogy az integrált áramkörön belül vannak megvalósítva az emulátor funkciók. Tehát megfelelő eszköz vagy alkalmazás segítségével e porton keresztül a DSP működése nyomon követhető és tesztelhető. Ezt a portot használja a 3.4.2. alfejezetben ismertetett EZ-ICE Emulátor is.

3.3.3. Értékelés

Kijelenthető, hogy a processzor felépítése és különböző funkciói kielégítők abból a szempontból, hogy a megvalósítani kívánt jelfeldolgozási algoritmusok széles körének implementálását hardver szinten támogatja.

A processzor számítási kapacitása is elegendően nagy ahhoz, hogy a nyolccsatornás rendszer összes csatornáját kezelje, illetve, hogy alkalmazói programok fussanak rajta. A IV. táblázatban ismertetett számítási eredményeket itt csak bemutatjuk, levezetésükre és részletes magyarázatukra a 6. fejezetben kerül sor.

	f_{MV} = 64 kHz	f_{MV} = 32 kHz	f_{MV} = 16 kHz	f_{MV} = 8 kHz
Egyszerű működés	381 (47)	1162 (145)	2725 (340)	5850 (731)
Bővített működés	781 (97)	1562 (195)	3125 (390)	6250 (781)

IV. táblázat. Megszakítási rutin utasítások maximális száma (csatornánként)

Ezek az eredmények bizonyítják azt, hogy az ADSP 21061-es elegendően nagy számítási kapacitással rendelkezik ahhoz, hogy a nyolccsatornás rendszer teljeskörű vezérlését ellássa. Még a legmagasabb mintavételi frekvencia mellett is csatornánként 47 utasítás hajtható végre, amely az overhead miatt egy kb. 40 együtthatós FIR szűrő megvalósítására elegendő. A nyolccsatornás rendszeren megvalósítható alkalmazások többsége nem igényli azt, hogy codec-ek a maximális mintavételi frekvenciával üzemeljenek. Ezért az átlagos alkalmazások esetén elegendően hosszú jelfeldolgozási rutinok futtatására van lehetőség, ami biztosítja a problémamentes működést.

Összefoglalva: az ADSP-21061 processzor eleget tesz azon követelményeknek, amelyeket az általunk tervezett nyolccsatornás rendszer vezérlésének feladata támaszt vele szemben.

3.4. A VisualDSP fejlesztői környezet

A DSP programok fejlesztése az EZ-KIT kártyával nagymértékben együttműködni képes VisualDSP fejlesztői környezet segítségével történt. A VisualDSP az Analog Devices cég szoftver terméke. Ez egy személyi számítógépen futó integrált fejlesztői környezet, amelyet azt a célt szolgálja, hogy az ADI DSP-in futó programokat fejlesszék ki segítségével. Az alkalmazás Windows környezetben fut, grafikus környezetben.

A fejlesztői környezet két független alkalmazásból áll, a Projekt szerkesztőből és a Debugger-ből.

3.4.1. Projekt szerkesztő

A Projekt szerkesztőben projekt fájlokat lehet megnyitni és szerkeszteni. Egy ilyen projekt tulajdonképpen egymáshoz rendelt forráskód fájlokat jelent, amelyekből az alkalmazás egy DSP-re letölthető fájlt generál.

A kimeneti fájl típusa beállítható a Projekt tulajdonságok ablakban. A lehetőségek: debuggolható verzió, kész verzió, vagy EPROM-ba égethető bit-file. A kimeneti fájl kiterjesztése ettől a beállítástól függ, A kész verzió kiterjesztése .exe, a debuggolható verzióé .dxe. A forráskód fájlok lehetnek gépi nyelven megírtak (kiterjesztésük: .asm), C nyelven megírtak(.c), vagy a linkelést leíró úgynevezett Linker Description File-ok. (.ldf).

A Projekt szerkesztőn belül a következő funkciók vannak megvalósítva:

- **C fordító**

E funkció biztosítja, a DSP-k C nyelvű programozását, amely sok esetben kényelmesebb a nehézkesnek tűnő gépi kóddal szemben. Használata mégsem igazán elterjedt, mert a C fordító nem tudja maximálisan kihasználni a DSP hardver adta lehetőségeket. A DSP-ket pedig éppen azért használják a különböző időkritikus jelfeldolgozó algoritmusok megvalósítására, mert hardver szinten támogatják a jelfeldolgozási műveletek végrehajtását. Ezeket a speciális hardver lehetőségeket viszont csak egy assembly nyelven megírt program tudja igazán kihasználni. Sok esetben használatos az az egészséges kompromisszumot jelentő megoldás, hogy a DSP programok sokszor lefutásra kerülő részleteit gépi nyelven írja meg a fejlesztő, míg az egyszer lefutó inicializáló jellegű programrészleteket C-ben. Így a futási idő túlnyomó részében futó rutinok assembly nyelven vannak megírva, tehát a program futása nem lassul le lényegesen. [11]

- **Compiler**

A compiler végzi a gépi kódban megírt forrásfájlok valamint a C fordító által lefordított fájlok szintaktikai ellenőrzését, valamint előállítja a futtatható fájlokat.

- **Linker**

A linker felelős azért, hogy az egy projekten belüli különálló fájlokat egybefűzze, és előállítsa a kimenet .exe vagy .dxe fájlt. Minden projekthez tartozik egy Linker Description File (LDF) is, amely szintén a projekt részét képezi. Amenyiben a felhsználó nem ad meg ilyen LDF-et, akkor a Projekt szerkesztő defaultként egy gyári LDF-et használ.

A Linker Description File a linker számára tartalmaz lényeges információkat, tehát nem közvetlenül a DSP-n futó program része. Az LDF egy egyszerű, néhány utasításból álló nyelven megírt leírófájl, amely alapján a linker összefűzi a forrásfájlok kód és adatrészleteit. Ez a leírófájl tartalmazza azt a lényeges információt, hogy a különböző forrásfájlok adat-, és kódrészleteit a DSP-n fizikailag melyik memóriaterületre kívánja letölteni a felhasználó.

- **Loader és Splitter**

Ez a két funkció akkor használatos, ha a kész DSP programot EPROM-ba kívánjuk égetni. Ezen alkalmazások segítségével állíthatók elő a szükséges bit fájlok.

3.4.2. Debugger

A fejlesztői környezet másik nagy részegysége a Debugger alkalmazás. Ez az alkalmazás lehetőség biztosít a megírt DSP programok kipróbálására futás közben. Lehetőség van a program futásának részletes vizsgálatára, töréspontok helyezhetők el, lehet léptetve végrehajtani az utasításokat, a memóriaterületek valamint a rendszer regisztereinek tartalma folyamatosan monitorozható. A Debugger „bementi adata” a Projekt szerkesztő által előállított .dxe kiterjesztésű fájl.

A DSP program futtatásának helye beállítható, ez lehet emulátor vagy szimulátor.

- **Szimulátor**

Ebben az esetben a program fizikailag nem töltődik le a DSP-re, hanem az alkalmazás a személyi számítógépen szimulálja a DSP működését. Ez a szimuláció minden részletre kiterjed, tehát a DSP majdani várható működése vizsgálható. Ilyen

módon azonban nem lehet kipróbálni sem a fizikai DSP alapú rendszer többi egységét, illetve az egységek összehangolt működését sem. Ezért, ha lehetőség van rá, akkor az emulátor használata előnyt élvez.

- **Emulátor**

Az emulátor használatára akkor van lehetőség, ha rendelkezésre áll az ADI által gyártott EZ-ICE emulátor kártya a személyi számítógépben. Ehhez tartozik egy csatlakozó, amelyet az ADI fejlesztői kártyáin található JTAG Emulátor csatlakozóval lehet összekötni. Erre a processzor belső (vagy on-chip) emulálását lehetővé tevő szabványos JTAG portjának lábai vannak kivezetve. Az emulátor ezen az interfészen keresztül vezérli a teljes vizsgálatot. Ugyanitt történik a Projekt szerkesztő által előállított program letöltése a DSP memóriájába. A letöltés után a processzort a Debugger Halt állapotban tartja. Ekkor lehetőség van a program futtatására, az utasítások léptetett elvégzésére vagy töréspontok elhelyezésére. A DSP összes regiszterének tartalma lekérdezhető és igény szerint módosítható. A memóriaterületek teljes területének tartalma fájlba írható (dump) vagy feltölthető (fill). A fenti műveletek elvégzésének mindegyikét a JTAG szabvány szerint a DSP hardveresen támogatja, ezért elvégzésükhöz nincs szükség külön monitorprogram letöltésére. A Debugger alkalmazás segítségével a program futásának teljes körű vizsgálata lehetővé válik.

4. fejezet

A rendszer specifikációja

4.1. A rendszer áttekintése

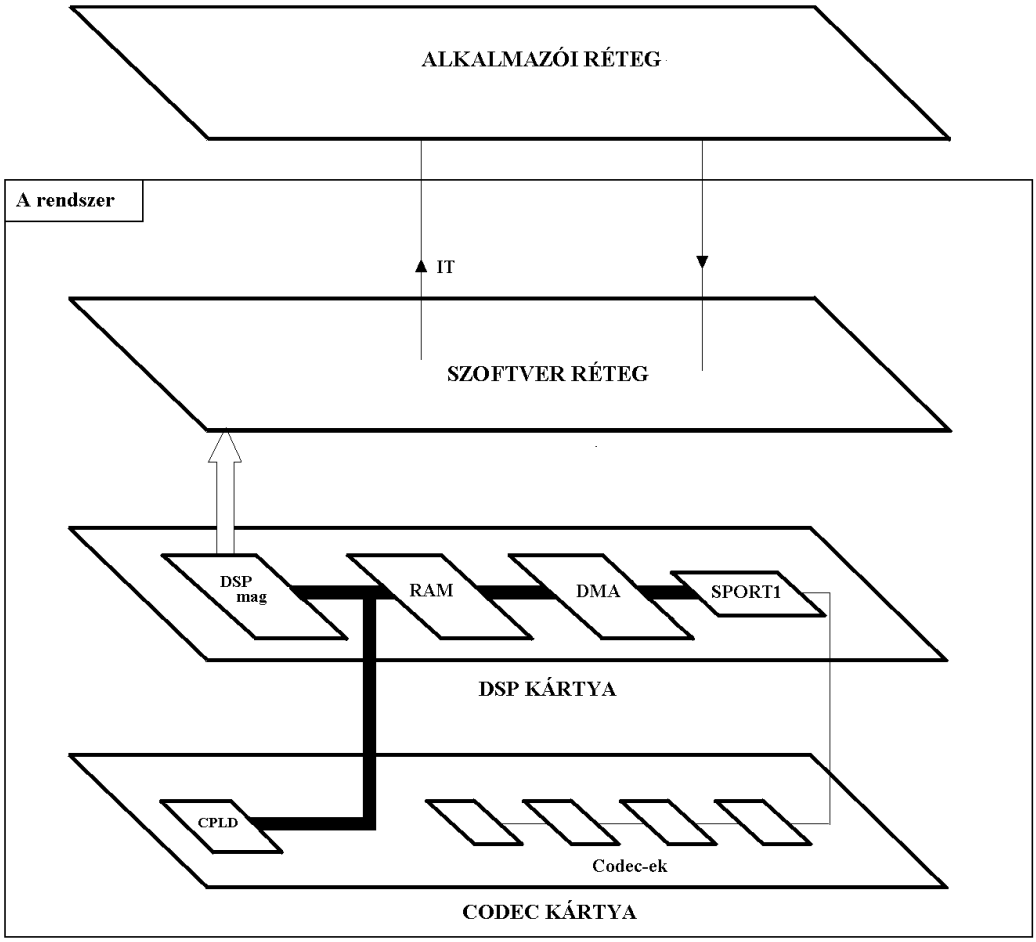
Az általunk tervezett nyolccsatornás rendszer három fő része az előző fejezetben ismertetett DSP kártya, egy codec kártya, valamint a működtető szoftver. (lásd 4.1. ábra)

A többcsatornás jelfeldolgozó rendszer tervezésének kezdeti fázisában mérlegeltük, hogy milyen előnyökkel, illetve hátrányokkal járna, ha a teljes rendszert egy közös nyomtatott áramkörön valósítanánk meg. Arra a következtetésre jutottunk, hogy a DSP nagy lábszáma és igen nagy komplexitású környezete miatt a nyomtatott áramkör tervezése jelentős mértékben nehezebb lenne, mint egy olyan nyomtatott áramköré, amelyen nincs DSP.

Ezért döntöttünk azon megoldás mellett, hogy a rendszer hardver részét két nyomtatott áramköri kártyából építjük meg. Ezek egyike egy készen vásárolt, gyári ADSP-21061 EZ-KIT Lite fejlesztői kártya, amelyet az előző fejezetben ismertettünk.

A rendszer másik hardver egysége, a codec kártya saját fejlesztés. Mivel a DSP az EZ-KIT kártyán található, a saját nyomtatott áramkör tervezése ilyen módon egyszerűbbé vált. Ezen kívül a két kártyából álló rendszernek további előnye, hogy kisebb a költsége mintha egy DSP-t is tartalmazó nyomtatott áramkört terveztünk volna.

Az EZ-KIT kártyán található gyári EPROM áramkör alapvetően arra szolgál, hogy a DSP boot programját tárolja. Ugyanakkor ez a memória tetszőlegesen felhasználható arra, hogy a boot programon kívül az elkészült DSP programokat is beégessük. Ekkor lehetővé válik az, hogy a rendszer személyi számítógéptől függetlenül működjön. Ilyen módon a rendszer mobilizálható, alkalmazása egyedülálló (stand – alone) rendszerként is lehetséges.



4.1. ábra A rendszer áttekintő ábrája

4.2. A codec kártya

A tervezés első fázisában alkalmas AD/DA átalakítókat kellett választanunk, amely a feldolgozni kívánt analóg jelek és a DSP közötti illesztést valósítja meg.

Az alkatrészválasztás során külön A/D és D/A IC-k helyett ún. codec-eket részesítettük előnyben. Az elnevezés a kóder-dekóder szókapcsolatból származik. Ezek olyan eszközök, amelyek egy tokba integrálva tartalmaznak A/D illetve DA átalakítókat. Az Analog Devices codec-jeit kifejezetten DSP-k mellé fejlesztették ki, ezért mind hardver szempontból, mind szoftver szempontból előnyösebb az alkalmazásuk, mint az egymástól független A/D és D/A átalakítóknak.

Előnyös tulajdonsága a codec-eknek, hogy egyszerűen kezelhető, kevés vonalat igénylő soros porttal rendelkeznek, amelynek segítségével egyszerűen illeszthetők DSP-khez. Vannak olyan codec-ek is, amelyek kaszkádosíthatók, tehát esetükben több codec együttes működésére is van lehetőség. A mi rendszerünkben éppen ilyen codec-ekre van szükség, mert a többcsatornás működést csak több codec egyidejű használatával érhetjük el.

A codec-ek használatának további előnye az, hogy több funkcionális egységet valósítanak meg egy IC tokon belül. Emiatt használatuk esetén a nyomtatott áramkör tervezése is némileg egyszerűsödik, valamint az áramkör disszipációja is csökken.

A codec-kártyát alapvetően aktív zajszűrési, szabályozási és méréstechnikai feladatok ellátására építettük. Ezért olyan codec-eket választottunk, amelyek tulajdonságai ilyen jellegű alkalmazások implementálása esetén kedvezők.

Emiatt a rendszer nagy csatornaszáma fontosabb jellemző volt, mint a codec-ek sávszélessége. Konkrétan az előzetes specifikáció egy minimum hatcsatornás (6 kimenetű, 6 bemenetű), de inkább egy nyolccsatornás 10-15 kHz-es sávszélességű eszközt határozott meg. A kívánatos csatornaszám azért ennyi, mert négy csatorna a bevezetőben felsorolt legtöbb alkalmazás esetén kevés lenne. Nyolcnál több csatorna esetén pedig egyre inkább előtérbe kerülnek a nem elegendően nagy számítási kapacitásból adódó problémák. Ezért a nyolc csatorna optimálisnak mondható egy egyetlen DSP által vezérelt rendszerben.

További igény volt az is, hogy a kártya DC jeleket is képes legyen feldolgozni. A most rendelkezésre álló laborkártyák ugyanis csak AC-csatolt működést tesznek lehetővé.

A sávszélességen kívül a codec-választás másik kérdésköre az, hogy az A/D és D/A átalakítók milyen elven működnek. Szukcesszív approximációs, illetve szigma-delta elven működő codec-ek jönnek szóba. Az audiotechnikában általában a túlmintavételezéssel működő szigma-delta átalakítókat használják, amelyek használata azért kedvező, mert nagy a linearitásuk, és nem igényelnek külön átlapolásgátló szűrőt. A hátrányuk az, hogy késleltetésük (group delay) nagyobb, mint a szukcesszív approximációs átalakítóknak. Ez utóbbi éppen kisebb késleltetése miatt kedvezőbb lenne a szabályozási alkalmazások szempontjából, de összességében a szigma-delta elven működő codec mellett szóló érvek voltak az erősebbek.

Ezért a megtervezett codec-kártyán az Analog Devices AD73322 codec-jét használtuk. Ez az eszköz az összes eddig felsorolt szempontból kedvező tulajdonságokkal rendelkezik. A codec részletes bemutatására az ötödik fejezetben kerül sor.

Az elkészített codec kártya egy saját fejlesztésű eszköz, amelynek tervezése során felhasználtuk az Analog Devices AD73322EB fejlesztői kártya kapcsolási rajzának néhány elemét. [9]

A kártyán négy darab AD73322 codec IC van, ezekhez 8 sztereó csatlakozóval lehet kapcsolódni. Így valósul meg a nyolc csatorna, azaz a 8 bement, 8 kimenet.

Felmerült az az igény is, hogy a rendszernek legyen valamiféle kezelői felülete is. Ennek érdekében a codec-kártyán elhelyeztünk egy ispLsi1016 típusú CPLD áramkört is. Ehhez kapcsolódik egy nyolc kapcsolóból álló DIP kapcsolósor, illetve nyolc LED. A kapcsolósor jól használható az alkalmazói program különböző paramétereinek futás közbeni változtatására, a LED-ekkel pedig a program futás közbeni állapotának kijelzése oldható meg.

A CPLD a DSP külső memóriabuszára van illesztve, tehát egy dedikált memóriacím van az eszközhöz rendelve. Ilyen módon egy egyszerű memóriaolvasási művelettel leolvasható a DIP kapcsolósor állapota, illetve ugyanerre a memóriacímre történő írással beállítható a LED-ek állapota.

4.3. Szoftver réteg

Tekintve hogy a rendszer vezérlő eleme, a jelfeldolgozó processzor egy programozható eszköz, a hardver egységeken kívül a szoftver eszközöket is specifikálni kellett.

Alapvetően arra van szükség, hogy a DSP-n futó gépi nyelven megírt program a két hardver egységet megfelelően működtesse, valamint hogy vezérelje az egységek között zajló kommunikációt. Ez a program inicializálja a hardver kártyákat, illetve vezérli az egységek kommunikációját működés közben.

A nyolccsatornás rendszer szoftver eszközei képezik a rendszer szoftver rétegét. Ezt az elnevezést az indokolja, hogy nyolccsatornás rendszerhez nem csak egy működtető programot készítettünk, hanem egy olyan szoftver keretet biztosítunk, amely túlmutat egy konkrét DSP program szerepén.

A szoftver réteg további funkciója az, hogy a rendszeren implementálandó jelfeldolgozói alkalmazásokat beágyazza a rendszerbe. Ez konkrétan azt jelenti, hogy a működtető program az alkalmazói programok számára egy keretprogramként jelenik meg. Tehát a rendszerhez tartozó szoftver réteg önmagában még nem valósít meg jelfeldolgozó algoritmusokat, de egy szilárd alapot biztosít ilyen alkalmazások implementálásához. Ilyen módon a szoftver réteg a hardver eszközök primitív operációs rendszerének tekinthető.

A megvalósítás ezen módja azért kedvező, mert így a jelfeldolgozó alkalmazást fejlesztő mérnöknek nem kell a hardver megvalósítás részleteit ismernie, a vezérlés részleteivel nem kell foglalkoznia, mert ezen funkciókat a rendszerhez tartozó keretprogram elvégzi helyette. Az alkalmazás rendszerbe ágyazása pedig olyan módon történik, hogy a keretprogram bonyolítja le a DSP – codec kommunikációt és az alkalmazói program számára előkészíti az adatokat. Az alkalmazói programnak csak dedikált memóriaterületekről kell beolvasnia az A/D felől érkező adatokat. A kiírás ugyanígy történik, tehát a DA-nak szánt adatokat dedikált memóriaterületre kell beírni.

A rendszer szofver rétegének ilyen megvalósítása biztosítja azt, hogy a nyolccsatornás rendszer valóban széles körben használható legyen.

A hardver illetve a szoftver specifikációt az alábbiakban foglaljuk össze. A közölt paraméterek többségét a fenti megfontolások alapján határoztuk meg, illetve a paraméterek egy része a tervezés során felhasznált alkatrészek tulajdonságaiból adódott.

4.4. Hardver specifikáció

Jelfeldolgozó processzor
Analog Devices ADSP-21061 lebegőpontos DSP
50 MIPS (max. 120 MFLOPS)

Analóg bemenetek
8 analóg bemenet, 4 db AD73322-es codec segítségével kialakítva
max. 0,7 V p-p [a codec bemeneti erősítőjének 0 dB-es beállításakor]
16 bites felbontás
szigma-delta A/D átalakítókkal mintavételezve
programozható bemeneti erősítés 0/38 dB
magas bemeneti impedancia
választható AC, DC csatolás
jackaljzat csatlakozás

Analóg kimenetek
8 analóg kimenet, 4 db AD73322-es codec segítségével kialakítva
1,4V p-p
16 bites felbontás
szigma-delta DA átalakítókkal mintavételezve
programozható bemeneti erősítés +6/-15 dB
jack aljzat csatlakozás

Mintavételi frekvenciák
64 kHz, 32 kHz, 16 kHz, 8 kHz [16,384MHz-es kristály esetén]
62,464 kHz, 31,232 kHz, 15,616 kHz, 7,808 kHz [16 MHz-es kristály esetén]
Egyéb mintavételi frekvenciák kristálycsere esetén elérhetőek.
Ennél kisebb mintavételi frekvenciák szoftver úton érhetők el

Csoport késleltetés
25 μsec A/D átalakításkor
50 μsec DA átalakításkor

Hoszt kommunikáció
Az EZ-KIT Lite rendelkezik RS-232-es soros porttal, és JTAG interfésszel

Teljesítmény igény
Dupla tápfeszültség, ±8 V - ±12 V, 800 mA (Az EZ-KIT Lite kártya fogyasztásával együtt)
AC, DC betáplálás

Méret
225mm x 165mm x 50mm

4.5. Szoftver specifikáció

Csatornák száma

8 analóg bemenet
8 analóg kimenet
12 bit (8+4bit) általános digitális kimenet (memóriába ágyazottan elérhető)
11 bit (8+3bit) általános digitális bemenet (memóriába ágyazottan elérhető)
CPLD átprogramozása esetén komplexebb I/O funkciók is elérhetőek

A megvalósítható FIR szűrők együtthatóinak száma (csatornánként)

	f_{MV} = 64 kHz	f_{MV} = 32 kHz	f_{MV} = 16 kHz	f_{MV} = 8 kHz
Egyszerű üzemmód	320 (40)	1104 (138)	2664 (333)	5800 (725)
Kibővített üzemmód	720 (90)	1504 (188)	3064 (383)	6192 (774)

Megjegyés:
Az együtthatók számának számításakor adatmozgatás miatti overhead-et figyelembe vettük.

Analóg bemenetek
szoftveresen állítható bemeneti erősítés (0, 6, 12, 18, 20, 26, 32, 38 dB)

Analóg kimenetek
szoftveresen állítható kimeneti erősítés (6, 3, 0, -3, -6, -9, -12, -15 dB)

Mintavételi frekvenciák
A hardveresen beállítható mintavételi frekvenciák csökkenthetők decimálással

Kezelői felület
8 általánosan használható DIP kapcsoló
8 általánosan használható LED
2 power LED
2 DIP kapcsoló a codec lánc hosszának beállítására
1 DSP RESET nyomógomb [EZ-KIT kártyán]
1 DSP megszakításkérő nyomógomb [EZ-KIT kártyán]
1 DSP FLAG bemenetre illesztett nyomógomb [EZ-KIT kártyán]
4 DSP FLAG lábra illesztett LED [EZ-KIT kártyán]

Alkalmazás fejlesztés
VisualDSP környezetben, assembly vagy C nyelven

Adatok mentése
A DSP memóriájának kimentése(dump) JTAG porton

Standalone működés
A rendszerben található EPROM segítségével lehetséges

5. fejezet

Hardver tervezés

5.1. Alkatrészek kiválasztása

5.1.1. A codec-ek

Elsősorban az Analog Devices cég termékei között kerestük az alkalmas eszközt, mert az alkatrészeik könnyen beszerezhetők és az eszközünket a fent említett cég DSP-jét tartalmazó kártyához szeretnénk illeszteni. Az elérhető codec-ek (AD1835, AD1836, AD1837, AD1838, AD1839, AD73311, AD73322) közül az AD73322-es eszköz bizonyult a legjobb választásnak. Az AD183x családot, amelyeknek a bemeneti, kimeneti sávszélességük 20 kHz felett van, amiatt zártuk ki, mert nem lehet őket megfelelő mennyiségben kaszkádosítani. Az AD73311-ből azonban már 4 tok is kaszkádosítható, de mivel tokonként csak egy AD/DA-t tartalmaznak, szintén nem lettek volna megfelelők. Az optimális választásnak tehát az AD73322-es bizonyult, amiből szintén csak négy tok kaszkádosítható, de az előbbi eszközzel ellentétben ez már két AD/DA-t tartalmaz, amelyek segítségével már kielégíthető az előzetes specifikáció. A következőkben feltüntetett adatok az Analog Devices honnlapjáról származnak: www.analog.com.

Az AD73322 főbb paraméterei:

- 2db 16 bites szigma-delta A/D, szimmetrikus bemenet
- 2db 16 bites szigma-delta DA, szimmetrikus kimenet
- 8, 16, 32, 64 kHz-es mintavételi frekvencia
- Beépített 1.2 V, 2.4 V-os referencia

Az AD73322 főbb alkalmazásai:

- Általános analóg ki/bemenet
- Beszédfeldolgozás
- Aktív zajcsökkentés

5.1.2. Az analóg bufferek (műveleti erősítők)

Mivel a beépített codec-ek csak szimmetrikus jeleket képesek venni, illetve adni, ezért szükség van analóg jelformáló áramkörökre mind a kimeneteken, mind a bemeneteken. A bemeneten megjelenő aszimmetrikus jelet át kell alakítani referencia központú szimmetrikus jellé, míg a codec által kiadott szintén referencia központú szimmetrikus jelet aszimmetrikussá. A kimenetekkel szemben támasztott további

követelmény, hogy terhelhetőségük nagyobb legyen a laborban található fejlesztői kártyákénál, tehát egy fejhallgatót is meg tudjanak hajtani, emiatt különböző műveleti erősítőket kell alkalmazni a bemeneten és a kimeneten. A kimeneti buffer választása szempontjából tehát a legfontosabb paraméter a maximális kimenő áram és a kettős tápfeszültség melletti működés. A műveleti erősítőkkel szemben támasztott további kritériumok voltak – többek között – az alacsony fogyasztás, a kis bemeneti ofszet feszültség és a nagy bemeneti impedancia. Az előbbi kritériumokat alapvetően FET alapú eszközökkel lehetett kielégíteni. A buffereket szintén az Analog Devices cég kínálatából választottuk a kártyán lévő alkatrészek beszerzésének egyszerűsítse végett.

Kimeneti buffernek az AD822 került beépítésre. Ez az eszköz, amely 2 műveleti erősítőt tartalmaz egy tokban, maximálisan kielégíti az igényeinket, mert már ±5 V esetén is képes kiadni a megfelelő áramot, és egy tok elegendő egy teljes sztereó kimenet bufferéléséhez.

Az AD822 főbb paraméterei:

- A nyílthurkú erősítés (A_0) sávszélessége 1.8 MHz
- Nyugalmi tápáram 800 μA
- Bemeneti impedancia 10^{13} Ohm
- Bemeneti ofszet feszültség 800 μV

Az AD822 főbb alkalmazása:

- Aktív szűrő

Bemeneti buffernek pedig az AD8042 választottuk, legfőképpen azért, mert a tervezéshez kiindulási alapul szolgáló AD73322 alapú fejlesztői kártyán is ezt ajánlották a bemeneti jelformálásra. Az eszköz maximálisan kielégíti a fentiekben megfogalmazott kritériumokat.

Az AD8042 főbb paraméterei:

- A nyílthurkú erősítés (A_0) sávszélessége 160 MHz
- Nyugalmi tápáram 5 mA
- Bemeneti impedancia 300 kOhm
- Bemeneti ofszet feszültség 3 mV

Az AD8042 főbb alkalmazásai:

- A/D meghajtó
- Videó kapcsoló

5.1.3. CPLD

Az általános célú digitális ki/bemenet illesztéséhez szükségünk volt egy programozható logikai eszközre. A fejlesztés során elkövethető hibák minimalizálása,

a hely csökkentése és a periféria flexibilitása érdekében döntöttünk a programozható logika mellett, annak ellenére, hogy diszkrét logikai áramkörök segítségével is meg lehetett volna oldani a feladatot, ha lemondtunk volna az előbb megemlített előnyökről. A választásunk a Lattice által gyártott ispLsi1016-os CPLD-re esett. Ennek legfőbb oka az volt, hogy a fejlesztés során föl tudtuk használni az eszközzel korábban szerzett tapasztalatokat. Mivel igen sok általános célú programozható ki-bemenetet tartalmaz, így már egy tok beépítésével el tudtuk látni az illesztést a DSP-hez, mindemellett maradt elegendő bit a ki- és bemenetek létrehozására a kártyán. Az eszköz beépítése mellett – számunkra ideális paraméterein kívül – még alacsony ára és könnyű beszerezhetősége is döntőnek bizonyult. Mivel az eszköz lehetőségeit egyelőre nem használtuk ki teljesen, így később lehetőség nyílik komplexebb digitális periféria megvalósítására is. Mivel sokszor újraprogramozható, ezért a különböző alkalmazásokban eltérő funkciókat is meg lehet valósítani vele. A feltüntetett adatok a www.latticesemi.com honlapról származnak.

Az ispLsi1016 főbb paraméterei:

- 32 programozható ki-bemenet
- 4 dedikált bemenet
- 96 regiszter (D flip-flop)
- 2000 programozható kapu
- Nagy sebesség
- Újraprogramozhatóság

Az ispLsi1016 főbb alkalmazásai:

- Komplexebb perifériák kialakítása
- Processzoros rendszerekben címdekódolás, memória kezelés

5.1.4. Egyéb alkatrészek

A kártyán található további digitális IC-k (inverterek, multiplexer) kiválasztásánál – azon kívül, hogy optimálisak legyenek az adott feladat megoldására – csak egy feltételnek kellett megfelelniük: keveset fogyasszanak. Ezért döntöttünk a 74HC sorozatú logikai elemek alkalmazása mellett, mert CMOS eszközök lévén statikus fogyasztásuk minimális, a bemenő impedanciájuk pedig nagy, így nem terheljük feleslegesen a többi digitális eszközt, és a nagy sebesség is biztosítható (16 MHz-es soros vonal).

A bipoláris kondenzátorok esetén tantál kondenzátorokat alkalmaztunk, mivel ezek alkalmasak legjobban tápszűrésre. A kisebb kapacitású kondenzátorok esetén pedig jó minőségű chip kondenzátorokat építettünk be.

Az ellenállások esetében csak a kis tűrés volt mérvadó, mert ezekkel erősítéseket állítunk be.

5.1.5. A választott alkatrészek tokozása

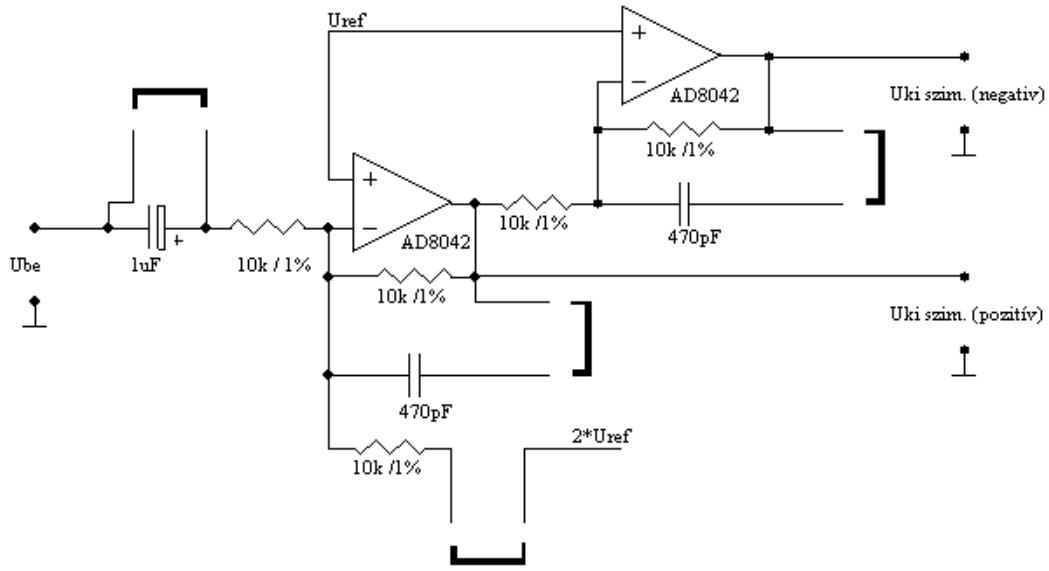
Ahol lehetséges volt, SMD kivitelű elemeket építettünk be, az egyszerű szerelhetőség, helymegtakarítás és a szükséges nagy, egységes földfelületek biztosíthatóságának érdekében.

5.2. Az egyes egységek funkciói

Ez a fejezet részletesen foglalkozik a negyedik fejezetben megismert blokkvázlat elemeivel.

5.2.1. Analóg bemenet

Az analóg bemenet teljes kapcsolási rajza a függelék 2.-5. oldalán található meg, egyszerűsített rajza pedig az alábbi ábrán látható.



5.1. ábra. Bemeneti egység egyszerűsített rajza

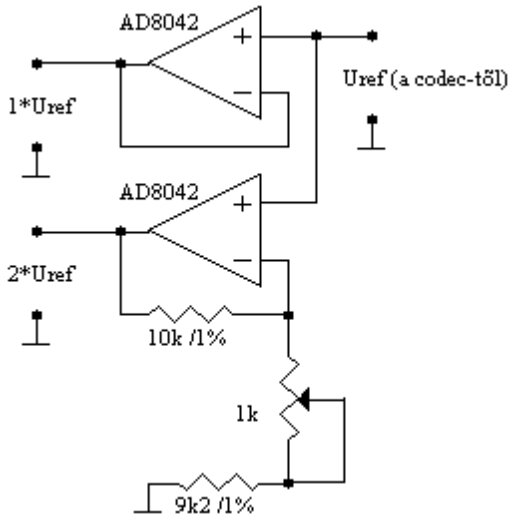
Ez az egység tulajdonképpen egy sztereó (2db aszimmetrikus) bemenetet illeszt a codec szimmetrikus bemeneteihez a codec-től kapott referencia jel segítségével. Az egység képes fogadni AC-, és DC- csatolt jeleket is, ezen üzemmódok közt jumperekkel lehet váltani. Mindemellett bemenetenként tartalmaz aluláteresztő szűrőt (anti-aliasing filter), ami szintén ki- és bekapcsolható. Bemeneti szűrőnek elég elsőfokú RC szűrőt alkalmazni, mert a szigma-delta átalakítás magas mintavételi frekvenciája miatt felesleges lenne nagyobb meredekségű szűrőt alkalmazni. Az egység alapvetően három fő komponensből épül fel, amelyek egy-egy műveleti erősítő köré épülnek:

- Bemeneti jelformáló jobb csatorna
- Bemeneti jelformáló bal csatorna
- Referencia jelerősítő, referencia jelkétszerező

Látható, hogy két bemeneti jelformáló komponens teljesen megegyezik. Az átalakítást tulajdonképpen két invertáló kapcsolásban működő műveleti erősítő végzi. Mivel az átalakított szimmetrikus jel központja a referencia feszültséggel kell megegyezzen, így a műveleti erősítők nem-invertáló bemenetét is a referencia

feszültségre kötöttük. A bemenő aszimmetrikus jel egy csatoló kondenzátoron keresztül kerül az első invertáló erősítő bemenetére, amelynek a kimenetén megjelenik a szimmetrikus jel egyik fele, majd az előbbi jelet rákapcsolva még egy invertáló erősítőre, létrejön a szimmetrikus jel másik fele. A DC csatolást a csatoló kondenzátorok rövidre zárásával és egy a műveleti erősítők munkapontjából adódó referencia feszültségnek megfelelő szintkülönbséget kompenzáló áram becsatolásával lehet létrehozni.

A referencia jel erősítő, amelynek kapcsolási rajzát az 5.2. ábra mutatja, feladata a codec-től kapott referencia jelet bufferelni, és a referencia jel kétszeresét is előállítani.



5.2. ábra. Referenciajel erősítők

A kétszeres erősítést finoman be lehet állítani az erre a célra beépített többfordulatú trimmer-potenciométer segítségével. Mivel a potenciométerrel sorba kötöttünk egy ellenállást, így az erősítés beállítás nélkül is kettő környékén van, és a potenciométer tekerésével kis mértékben lehet változtatni.

A bekapcsolható szűrő segít betartani a mintavételi törvényt. Mivel az esetek nagy részében nem dolgozunk szélessávú jelekkel, és általában fontos a kis késleltetés, ezért csak ritkán van szükség a szűrőre.

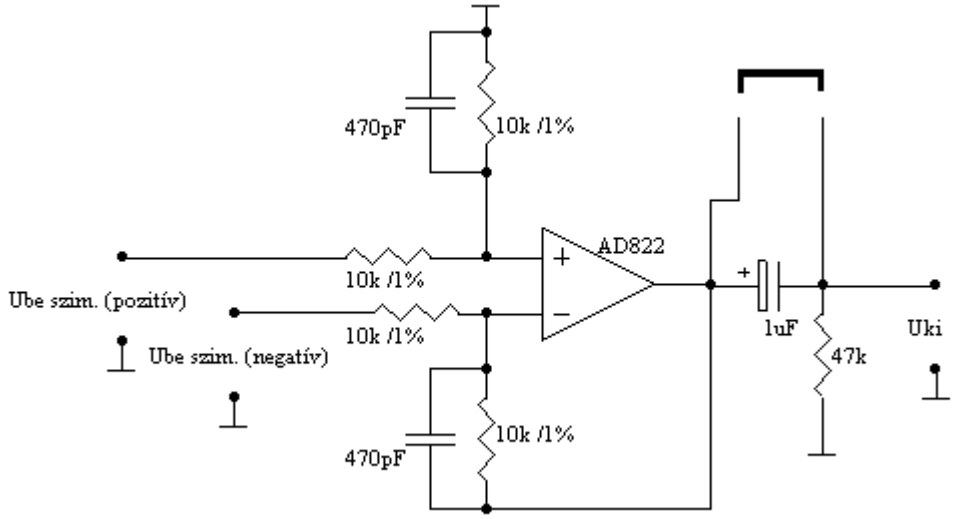
A bemeneti egység tartalmaz továbbá hidegítő kondenzátorokat, amelyek feladata a táp vezetékéken fellépő zavarok csökkentése, elnyomása. A jó minőségű tantál kondenzátorok és a velük sorba kapcsolt kis kapacitású chip kondenzátorok növelik a hidegítés hatásfokát. A zavarok további csökkentése érdekében az egység eleve egy, az analóg rész számára külön megszűrt tápponthoz csatlakozik, amit nem használnak a kártyán lévő digitális eszközök.

A bemenő jelet egy sztereó jackaljzaton keresztül fogadja az egység, mert a laborban található legtöbb mérőkábel jackdugóban végződik. A BNC-t azért vetettük el, mert az alacsony frekvencia tartomány nem teszi szükségessé.

5.2.2. Analóg kimenet

Az analóg kimeneti egység teljes kapcsolási rajza a függelék 6.-10. oldalán található.

Az alábbi ábrán a kimenet egyszerűsített rajza látható.



5.3. ábra. Kimeneti egység egyszerűsített rajza

Sztereó kimenet lévén két azonos komponensből áll. Ez az egység alakítja át a codec szimmetrikus kimeneteit egyszerűen kezelhető aszimmetrikus kimenetté, mindemellett még teljesítményerősítést is végez a nagyobb árammal való terhelhetőség érdekében. Az analóg bemeneti egységhez hasonlóan a kimenetek is lehetnek AC, vagy DC csatoltak. A DC csatolást itt is a csatoló kondenzátor rövidre zárásával lehet elérni. A DC csatolás tette szükségessé a negatív tápfeszültség alkalmazását, hogy az erősítők kimenetei le tudjanak menni negatív értékekre is. A műveleti erősítők tulajdonképpen kivonó erősítőként működnek egyszeres erősítéssel, tehát a kimeneti feszültség megegyezik a szimmetrikus kimenet pozitív és negatív vezetékein lévő feszültségek különbségével.

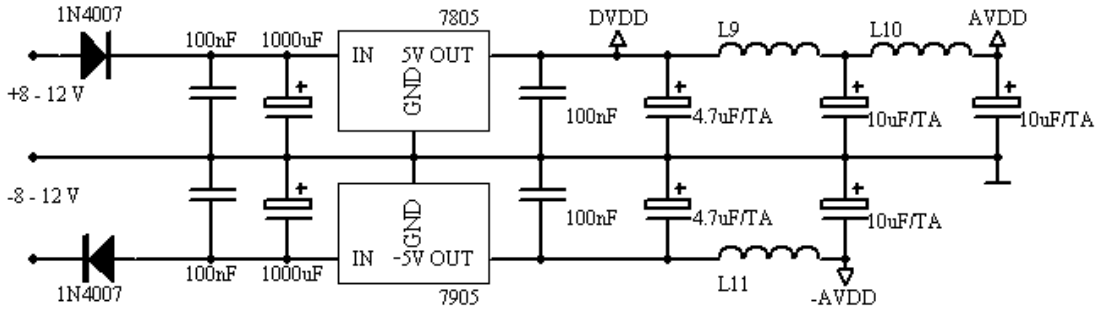
A műveleti erősítő különbségképzés mellett elsőfokú aluláteresztő szűrőként is működik, hogy eltüntesse a DA átalakítás miatt létrejövő mintavételi frekvenciával eltolt spektrumismétlődéseket, de mivel a szigma-delta DA is magas frekvencián működik, itt is elég volt elsőfokú szűrőket alkalmazni..

Az egység műveleti erősítői szintén az analóg részek számára külön megszűrt táppontokhoz csatlakoznak, tartalmaznak továbbá lokális táp hidegítő kondenzátorokat.

Az egyszerű kezelhetőség érdekében a kimenet is jack aljzatban végződik.

5.2.3. Tápegység

A tápegység a teljes kapcsolási rajza a függelék 13. oldalán megtalálható, egyszerűsített kapcsolási rajzát pedig az alábbi ábrán láthatjuk.



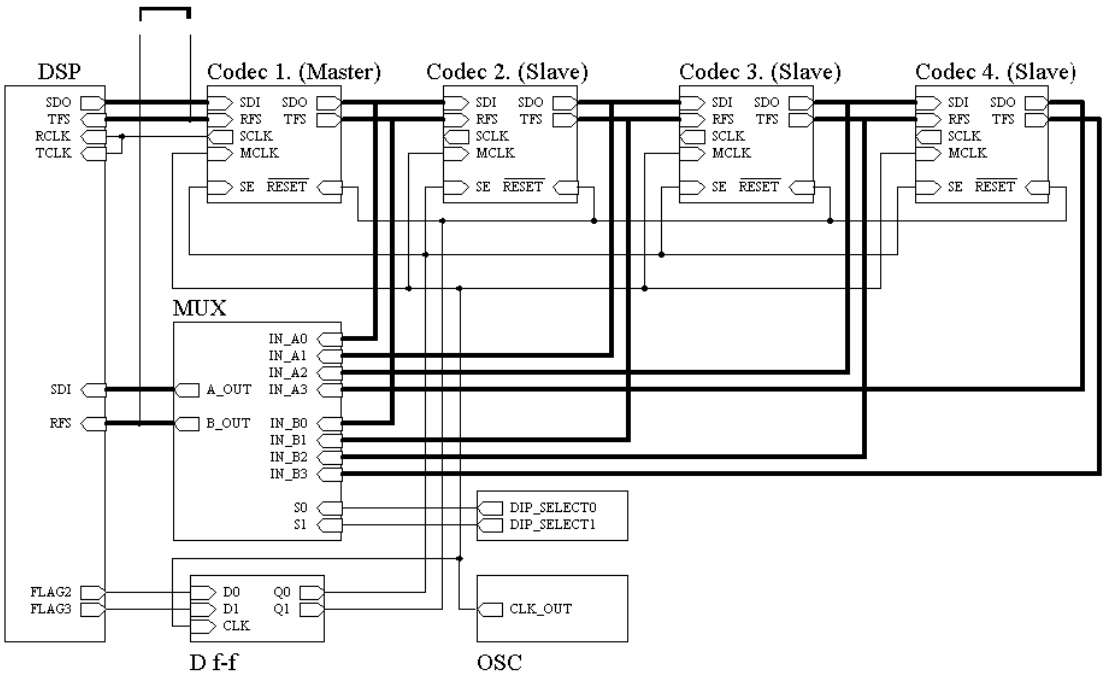
5.4. ábra. Tápegység

Az egység alkalmazása jelentősen megnöveli a kártya függetlenségét a külső tápforrásokkal szemben, ezáltal működőképes lesz AC, vagy DC betáplálás esetén is, ha a feszültség minimális értéke meghaladja az egyenirányítás után a 8 V-ot. A tápegység a rákapcsolt ±8 V - ±12 V-ból előállítja a kártyán lévő egységek számára a stabil ±5 V-ot, és ellátja a DSP kártyát is. A betáplálás minimális értékét (8 V) az adott ágban működő stabilizátor maradék feszültségének (drop feszültség) és a kimeneti stabilizált feszültségének összege határozza meg (8V = 5V + 3V). A betáplálás maximális értéke akár a 25 V-ot is elérhetné, mert emellett is működőképes lenne a kártyánk, de jobbnak láttuk alacsonyabbra előírni, mert nagy bemenő feszültségek esetén feleslegesen nagy lenne a stabilizátorokon hőveszteség formájában eldisszipált energia. Mivel a kártyát tápegységről, vagy hálózati transzformátorról fogjuk a későbbiekben táplálni, teljesen általános célú stabilizátorokat választottunk. Alapvetően két elvárásnak kellett megfelelniük, a maximális kimenő áramuk érje el a 0,8-1 A-t, és a kimenő feszültségük pedig legyen +5, illetve –5V. A tápegység a stabilizátorokon kívül tartalmaz még mindkét ágban egy-egy a fordított polaritás ellen védő, egyenirányító diódát. Továbbá mind a stabilizátorok bemenetén, mind a kimenetén beépítésre kerültek hidegítő, szűrő kondenzátorok, amelyek a gerjedést hivatottak megakadályozni. A bemeneten lévő nagykapacitású (4700 μF) kondenzátorok simítják a betáplált feszültséget. A tápegység helyes működését menet közben 2 LED folyamatos világítása jelzi. A digitális egységek táppontjai rögtön a stabilizátor után kapcsolódnak, míg az analóg részek egy, illetve két LC szűrőtagon keresztül kapcsolódnak a stabilizátorhoz. A szűrőtagok feladata a bejövő, illetve a digitális egységek által keltett nagyfrekvenciás (tüskeszerű) zavarok elnyomása, amely zavarok a feldolgozandó és a kimeneti jelben is megjelennének.

A tápegység csavaros sorkapcsok (nyák csokik) segítségével kapcsolódik a külvilághoz. Ezek a csatlakozók megfelelően stabil, kis impedanciás, de még bontható kötést valósítanak meg.

5.2.4. Codec-ek

A codec-ek és környezetük teljes kapcsolási rajza a függelék 10. oldalán található, az egység blokkvázlata pedig az 5.5. ábrán látható.



5.5. ábra. Codec-ek és környezetük

A négy kaszkádosított AD73322-es codec-et tekinthetjük a kártya legfontosabb részének. A kaszkádosítás félig hardveres, félig szoftveres úton történik. Az eszközök soros ki-,bemenetei egymáshoz tulajdonképpen láncszerűen kapcsolódnak. Az első master codec kapcsolatban van a DSP-vel és a második slave codec-el. A második codec pedig kapcsolódik az előbbi master-hez és egy másik slave-hez. A harmadik codec, amely szintén slave, már csak a második, illetve a negyedik slave-hez kapcsolódik, míg a negyedik szintén kapcsolódik a DSP-hez. Az ábrán vastag vonallal a codec-ek láncát jelöltük. Az adatok csak egy irányba mozoghatnak, a DSP-től az első codec-felé, majd azon keresztül a másodikhoz, mígnem keresztülhaladva a negyedik codec-en is, megint el nem érik a DSP-t. A kártya tartalmaz még egy jumpert, amelynek segítségével összekapcsolható a DSP felé menő soros adatok frame sync jele és a DSP felől jövő adatok frame sync jele. Így a kártyán lévő codec-ekkel kétféle kommunikációra van lehetőség.

A láncot az ábra szerint meg is lehet rövidíteni egy multiplexer segítségével. A rövidítés nagy segítségünkre volt az élesztésnél, és a kártya normál működése közben, nem teljes kihasználtság esetén fogyasztáscsökkentéshez is vezethet, mert a láncból kihagyott codec-ek alacsony fogyasztású üzemmódban maradnak. A láncszerűen összekapcsolt soros adat és a frame sync jeleken kívül egyéb jeleket is igényelnek a codec-ek. Az egyik legfontosabb jel a kommunikáció sebességét meghatározó MCLK jel, amit a master codec állít elő leosztással a kártyán lévő oszcillátor által létrehozott SCLK jelből. Több oszcillátor alkalmazását a codec-ek négyféle mintavételi frekvenciája tette szükségessé, 16.384 MHz-től eltérő kisebb frekvenciák alkalmazása esetén 64, 32, 16, 8 kHz-től eltérő mintavételi frekvenciák is előállíthatók. Egyéb közös vezérlőjelek még a RESET és a SELECT DEVICES. A RESET jel a codec-eket alaphelyzetbe állító jel, ami után újra be lehet állítani az összes codec-et. A SELECT DEVICES jel pedig egyszerű chipselect-ként működik, amely segítségével további kaszkádosítás is elképzelhető.

A multiplexert tehát a láncból csatol ki. Megkapja a SELECT DEVICES jelet is, mert ha a codec-ek nem működnek, akkor a kimenetei legyenek nagyimpedanciás

állapotban, mert ekkor lehetőség van másra használni az SPORT1-et. A kiválasztott csatornát DIP-kapcsolók segítségével állíthatja be a felhasználó.

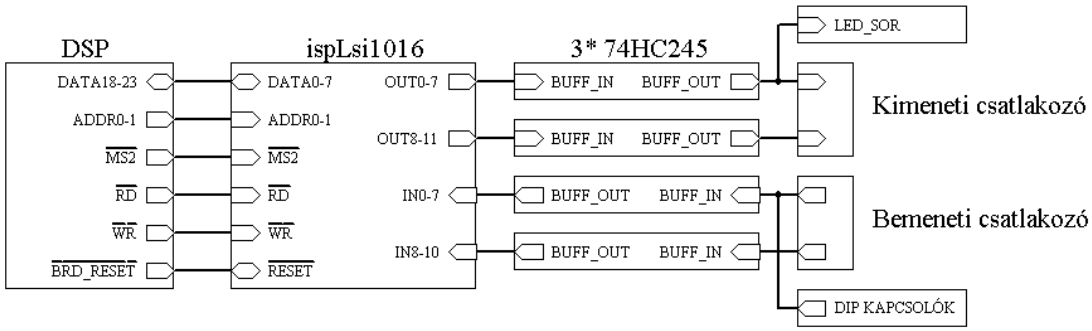
Az SCLK jel forrásaként négy oszcillátor közül is választhatunk. Ezek közül kettő tényleges beültethető oszcillátor, kettő pedig kristály által hangolható oszcillátor. A kétfajta oszcillátor-megvalósítás szintén a nagyobb flexibilitás érdekében jött létre. Az oszcillátorok között jumperrel lehet váltani, és a kiválasztott órajelet a helyes működés érdekében meg is bufferelték.

A RESET és az SELECT DEVICES jeleket, amelyeket a DSP FLAG2 és FLAG3-as jeléből hoztunk létre, D flip-flop-ok segítségével szinkronizáltuk az MCLK jelhez a helyes működés érdekében. Jumperek segítségével állítható a RESET jel aktív szintje, és létrehozható állandó chip-select is.

A codec-ek tápjának megszűrése rendkívül fontos volt. Mivel minden egyes codec külön-külön csatlakozik mind a digitális táphoz, mind az analóg táphoz, minden pontot meg kellett szűrni. Az codec-ek az analóg táphoz tekercsen keresztül kapcsolódnak, és ez a pont még kondenzátorokkal is meg van hidegítve. A digitális táp csatlakozását pedig elég volt csak kondenzátorokkal hidegíteni. Továbbá minden codec-hez csatlakozik egy 100 nF-os kondenzátor a REFCAP lábra. Ennek a kondenzátornak a feladata, hogy a codec által létrehozott referencia jel minél zajmentesebb legyen.

5.2.5. Digitális ki-bemenet

A digitális I/O-t megvalósító CPLD és környezetének kapcsolási rajza a függelék 12. oldalán található, míg a blokkvázlatát az alábbi ábrán láthatjuk.



5.6. ábra. CPLD és környezete

A CPLD a DSP párhuzamos port-jához kapcsolódik, és a DSP programozása szempontjából memóriába ágyazottan jelenik meg. A külvilág szempontjából pedig 12 bites állandó CMOS kimenetként, illetve 11 bites állandó CMOS bemenetként funkcionál. A CPLD az adatbusz 8 alsó bitjének vezetékén (DATA16-23) kívül, 2 címvezetékhez (ADDR0, ADDR1), az írást és olvasást vezérlő (WR, RD) vezetékekhez és a címdekódolásban nagy szerepet betöltő MS2 vezetékhez kapcsolódik. A fent említett jeleken kívül még a RESET-et is a DSP kártyától kapja. Mivel az MS2 jel kijelöli a DSP egy külső címtartományát, így a címdekódolásba bekapuzva elegendő volt csak két címbitet felhasználni, de emiatt az adott címtartományon más eszköz nem lehet.

Az egyelőre megvalósított funkciók megfelelő kezeléséhez elegendő lett volna egy címbitet is felhasználni, de – mint korábban említettük – a CPLD komplexitása bonyolultabb funkciók ellátására is alkalmassá teszi, amelyek elérésére már kevés

lehet az 1 bit. A kapcsolási rajzon jól látható, hogy a CPLD nem kap órajelet. Mivel az adott periféria funkciókat órajel nélkül is el tudja látni, csak felesleges zajt termelt volna. A CPLD és a külvilág közé 74HC245-ös 8 bites buszmeghajtókat építettünk a kimenetek nagyobb terhelhetősége és a CPLD védelme érdekében. Az alsó 8 kimeneti bit állapotát LED-ek jelzik, míg az alsó 8 bemeneti bit állapotát be lehet állítani DIP-kapcsoló segítségével is. A ki-bemeneteket az alapfunkciókon kívül fel lehet használni egyéb digitális eszközök vezérléséhez, ezt a célt szolgálja a két csatlakozó.

Az ABEL-VHDL nyelven írt programot fordítás után egy programozó segítségével töltöttük a CPLD-be. A program által megvalósított kapcsolási rajz a B függelék 14. oldalán található. A kimenetre kiírt értékeket D flip-flopokban tárolja, amelyek clk lábán fellépő beíró jel akkor aktív, ha az adott címre ír a DSP. Ezzel ellentétben olvasáskor nem a CPLD latch-el, hanem a DSP. Olvasás alatt a CPLD adott kimeneti buffere aktív, és a bemenetet egyszerűen kimásolja a DSP adat buszára.

5.3 A nyomtatott áramkör tervezése

A fejezetben a nyomtatott áramkör tervezésének menetéről, a tervezés során felmerült problémákról és megoldásaikról olvashatunk. Az elkészített panel két oldalának rajza a függelék 16. és 17. oldalán található.

5.3.1. Analóg és digitális kevert áramkör problémái

A nyomtatott áramkör megtervezésénél figyelni kellett arra, hogy az áramkör tartalmaz analóg egységeket, és az analóg egységek szempontjából erős zajforrásként funkcionáló digitális egységeket is. Megfelelő szeparációval és lokális hidegítésekkel azonban ki lehetett küszöbölni a problémát. A lokális hidegítésről az előző fejezetben már volt szó, a tervezésnél csak arra kellett ügyelni, hogy a hidegítő kondenzátorok a lehető legközelebb kerüljenek a hidegíteni kívánt IC-hez. A megfelelő szeparációt pedig az egyes egységek nyomtatott áramkörön való gondos elrendezésével, és csillagpontos földeléssel lehetett elérni.

5.3.2. Az alkatrészek elrendezése

A kártya elrendezésének vázlatát a függelék 15. oldalán találhatjuk. Az elrendezés többek között az alábbi kényszerek hatására jött létre:

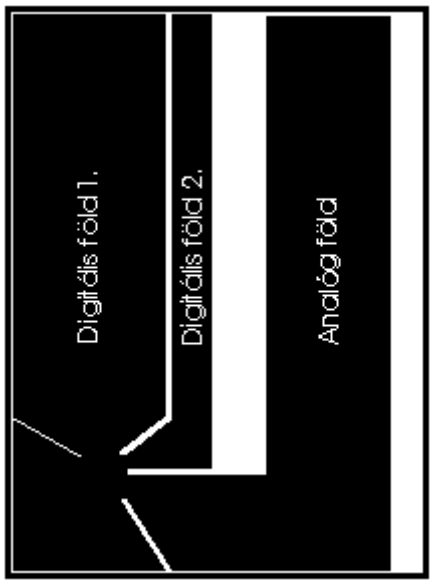
- Az analóg ki-, bemenetek egy oldalra kerüljenek
- Az analóg és digitális egységek elkülönüljenek
- A DSP kártya felhasznált csatlakozóinak elhelyezkedése kötött

Kiindulási pontként a DSP kártya méretei, és a kártyán lévő, – számunkra fontos – csatlakozók elhelyezkedése szolgált. Mivel a DSP kártya merev tüskesorok segítségével csatlakozik az általunk létrehozott kártyához, az eredmény egy emeletes elrendezés lett. A DSP kártya felhasznált csatlakozói U alakot alkotnak (ami az ábrán látható is), aminek következtében létrejött egy, az analóg egységek által nem használható, rosszul hozzáférhető terület. Emiatt került a digitális I/O-t megvalósító CPLD, és a codec-ek környezetét alkotó alkatrészek az előbb említett területre. A fenti kényszerek hatására kerültek az analóg ki-, bemenetek egymás mellé. Továbbá

az analóg egységek mellett foglalnak helyet a velük kapcsolatban lévő codec-ek is. A blokkvázlaton jól észrevehető, hogy a codec-eket összekötő képzeletbeli egyenes választja el a digitális egységeket az analóg egységektől. Az általunk tervezett kártya egyik hosszméretét a DSP kártya meghatározta meg, míg a másik méretet az egy sorba elhelyezett analóg egységek méreteinek összege határozza meg. A fennmaradó két sarokban került elhelyezésre a digitális I/O kezelőfelülete, csatlakozói és a tápegység.

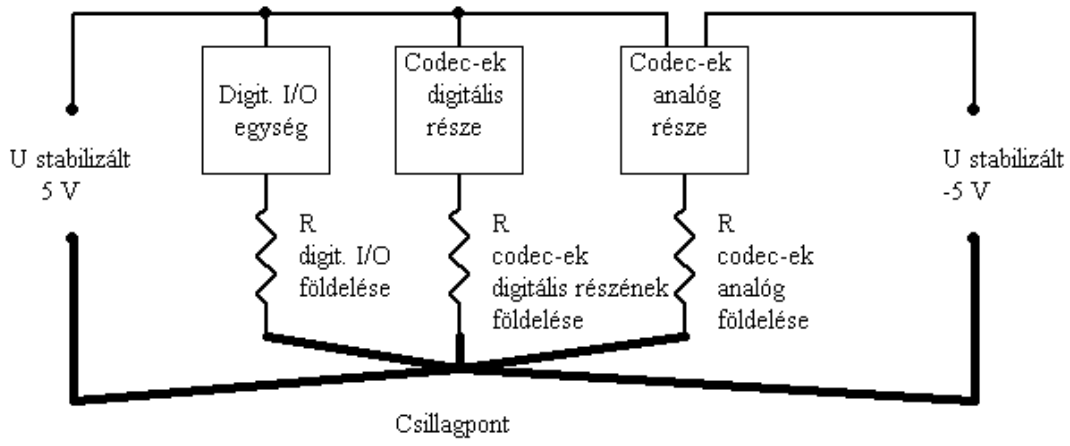
5.3.3. A földelés

A megvalósított földelést az 5.7 ábra mutatja.



5.7. ábra. A nyomtatott áramkörön kialakított földelés

Látható, hogy a csillagpont a két stabilizátor alatt, a referencia pontjaik csatlakozásánál található. A csillagpontos földelés lényege, hogy az egységeken keresztül folyó áramok külön úton jutnak el a közös csillagpontba. Mivel az adott pont (felület) ellenállása nagyon alacsony, ezért az egyes egységek földpotenciálja nem fog eltolódni. A csillagpontos földelés sematikus rajzát az 5.8. ábrán láthatjuk.



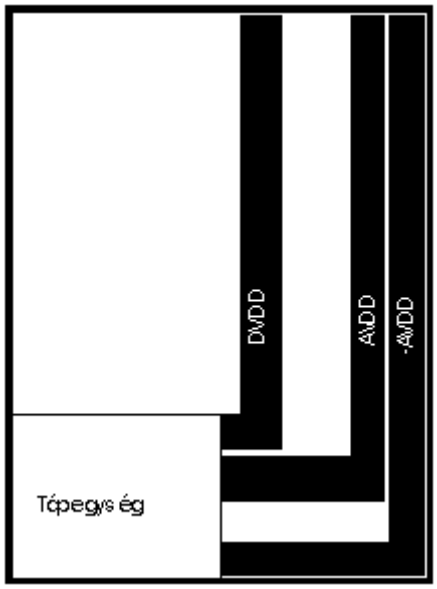
5.8. ábra. Csillagpontos földelés

A négy földsziget funkciója a következő:

- Analóg egység földje és a codec-ek analóg földje
- A codec-ek digitális földje
- A digitális ki-, bemenet földje
- Betáplálás, tápegység földje

A földvezetékek (földszigetek) alacsony impedanciáját a nagy felületek okozzák, ami azzal az előnnyel jár, hogy az adott egység árama sem fog jelentős földpotenciál növekedést okozni. Az analóg és digitális egységek további szeparációja pedig a tápegységben, LC tagok segítségével, a már említett módon történt.

A nagyobb felületű vezetékeket nemcsak a föld esetében alkalmaztunk, hanem a tápvezetékeknél is, törekedvén itt is a lehető legkisebb ellenállású kontaktusokra. A tápvezetékek elrendezését az 5.9. ábra mutatja.



5.9. ábra. Tápvezetékek elrendezése

5.3.4. A tervezés menete

A nyomtatott áramkör megtervezése a Protel 99-es program segítségével történt. Első lépésként el kellett készíteni az egyes egységek kapcsolási rajzát. Ezek után az egységek csatlakozási pontjait össze kellett rendelni a megfelelő egységekével. Az így elkészített dokumentumból már létre lehetett hozni a nyomtatott áramkör tervezéshez elengedhetetlen NET listát, amely tartalmazza a felhasznált alkatrészeket, tokozásukat és az egyes alkatrészek lábainak kapcsolatát.

Miután az összes NET listában szereplő alkatrész a kapcsolási rajzoknak megfelelő egységekbe lett rendezve, megkezdődhetett az egyes egységek optimális elrendezésének létrehozása. Ilyenkor kellett figyelembe venni az egységeket alkotó alkatrészek közötti kényszereket. Először az analóg ki-, bemenetek optimális elrendezését kerestük meg, arra figyelve, hogy a lehető legtöbb összeköttetést lehessen megvalósítani az alkatrészoldali rétegen, ezáltal minél egységesebb legyen a földsziget a másik oldalon. Az elrendezésnél törekedtünk arra is, hogy a jackaljzat és

az analóg tápvezetékek a téglalapszerű elrendezés egyik rövidebb oldalán csoportosuljanak, addig a vele szemközti oldalon lehessen csoportosítani a codec felé menő vezetékeket. Miután minden analóg egység – a már említett módon – el lett rendezve, egymás mellé kerültek elhelyezésre a panelen. Ezek után már egyszerű volt a codec-ek elhelyezése, mert mindegyiket az őt kiszolgáló analóg egységhez került közel. A következő fázis az elhelyezett elemek tápvezetékeinek létrehozása volt. Az analóg egységek +5 V-os és –5 V-os vezetékei egymás mellett futnak a jackaljzatok alatt az analóg földsziget mellett, míg a codec-ek analóg és digitális tápvezetékei a codec-ek alatt az analóg földsziget másik oldalán kerültek elhelyezésre. Az egyik megmaradt sarokban lett elrendezve a tápegység. Az alkatrészek elhelyezésekor figyelni kellett arra, hogy a nagyméretű pufferkondenzátorok elegendően távol kerüljenek a hűtőbordával ellátott stabilizátoroktól, ugyanis a magas hőmérséklet az elektrolit kondenzátorok idő előtti öregedését okozhatja. A hűtőbordák alá hőmérséklettűrő teflon távtartók kerültek, hogy a nyomtatott áramkör se legyen feleslegesen hőterhelés alatt, és így a hűtőborda is nagyobb felületen érintkezik a levegővel. A fennmaradó üres sarokban pedig a digitális I/O csatlakozóit, DIP-kapcsolóját, LED-jeit helyeztük el jól hozzáférhető módon.

A megtervezett nyomtatott áramkör a tervekből generált gyártó fájlok segítségével készült el.

Az alkatrészek beszerzése, és a nyomtatott áramkör legyártatása után megkezdődhetett az alkatrészek bemérése, majd a teljes rendszer bemérése, amelyet a 7. fejezetben ismertetünk részletesen.

6. fejezet

Szoftver tervezés

6.1. Bevezető

Az előző fejezetekben ismertetett hardver részegységek a szoftver réteggel együtt alkotják a nyolccsatornás jelfeldolgozó rendszert. Mivel a rendszer vezérlését a központi egység, a jelfeldolgozó processzor végzi, így a rendszer működtetését végző szoftver a konkrét esetekben a DSP-n futó programot jelenti.

A negyedik fejezetben részletesen kifejtettük, hogy a szoftver réteg szerepe túlmutat a hardver egységek vezérlésénél. A szoftver réteg nem egy konkrét jelfeldolgozó algoritmust megvalósító program, hanem egy általánosan használható szoftver környezet, amely alapja kíván lenni a konkrét jelfeldolgozó alkalmazásoknak. Ezzel biztosítjuk azt, hogy a nyolccsatornás rendszer valóban univerzálisan használható legyen, hiszen így a jelfeldolgozó alkalmazások fejlesztője az eszközt sokféle különböző alkalmazás implementálására használhatja anélkül, hogy meg kelljen ismernie az alkalmazást befogadó rendszer konkrét megvalósításának részleteit.

Dolgozatunk hetedik fejezetében bemutatjuk egy ilyen konkrét jelfeldolgozó alkalmazás implementálását. Ezzel amellett, hogy bizonyítjuk a rendszer helyes működését, egy példát is mutatunk arra, hogy milyen módon történik egy konkrét jelfeldolgozó alkalmazás megvalósítása a nyolccsatornás rendszeren.

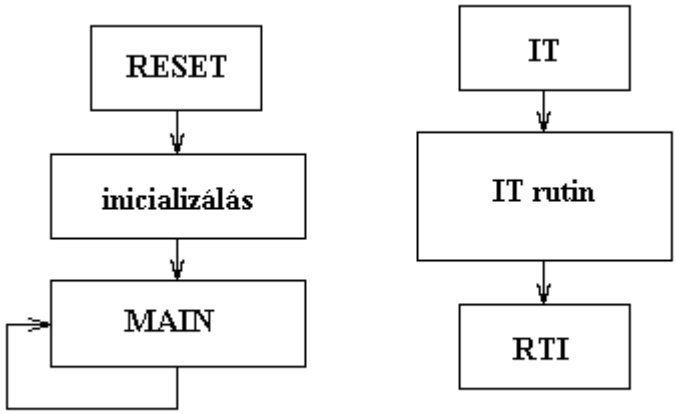
A DSP programok fejlesztése az előzőekben ismertetett VisualDSP fejlesztői környezet segítségével történt. A DSP programokat gépi nyelven írtuk a már ismertetett okok miatt, valamint annak érdekében is, hogy a lehető legnagyobb mértékben átlátható és kézben tartható legyen a DSP, illetve a többi hardver egység működése.

6.2. DSP programok struktúrája

6.2.1. Általános felépítés

Mielőtt ismertetnénk a nyolccsatornás rendszer konkrét működését és fejlesztésének menetét, az alábbiakban azon programtervezési irányelveket ismertetjük, amelyek a DSP programok esetében szokásosnak mondhatók, és amelyeket mi is szem előtt tartottunk a programfejlesztés során.

Az on-line működésű jelfeldolgozó programok szokásos struktúráját a 6.1. ábra szemlélteti.



6.1. ábra. A DSP programok általános struktúrája

RESET után az inicializáló rész fut le, melynek során a DSP különböző működési regiszterei kapják meg kezdeti értékeiket, előkészítve ezzel a majdani működést. Az inicializálás ideje alatt a megszakítások tiltva vannak. Az inicializálás végén engedélyezzük a megszakításokat, majd a program futása a főprogramba kerül. Ez tulajdonképpen nem más, mint egy önmagára visszaugró végtelen ciklus, amely nem végez semmilyen műveletet.

Ilyenkor a program valójában megszakításkérésekre várakozik. Ha bejön egy megszakításkérés, akkor az kiszolgálásra kerül, azaz lefut a hozzá tartozó rutin. Az A/D átalakítók akkor kérnek megszakítást, ha a beolvasott adat már rendelkezésre áll, tehát megkezdődhet az adatok feldolgozása. A jelfeldolgozó algoritmusokat ebben a megszakítási rutinban érdemes megvalósítani. Ennek magyarázata az, hogy a jelfeldolgozási feladatok általában felírhatók olyan módon, hogy egy adott pillanathoz tartozó bemeneti adatokból és a korábbi adatokból kiszámítjuk az adott pillanathoz tartozó kimeneti adatokat. Emiatt lesz kedvező az, ha a jelfeldolgozó algoritmust mindig akkor futtatjuk le, amikor új bemeneti adatok érkeztek. Ilyen programstruktúra adatvesztésmentes működésének feltétele az, hogy a jelfeldolgozó rutin futása befejeződjön azelőtt, hogy újabb megszakítás kérés érkezne az A/D felől. Ez a megszakítás kérés akkor érkezik meg, amikor az A/D felől megjön az újabb adatszó. Tehát a mintavételi idő áll rendelkezésre a megszakítási rutin teljes lefutásához. Ha ez az időtartam elegendő a rutin lefutásához, akkor a jelfeldolgozási algoritmus on-line működésű lesz.

Mivel az adatbeolvasás az A/D-ról és az adatkiírás a D/A-ra általában összehangoltan, egyidejűleg történik, a jelfeldolgozási rutin által kiszámított kimeneti adatok egy ütemnyi késleltetéssel jelennek meg a rendszer kimenetén.

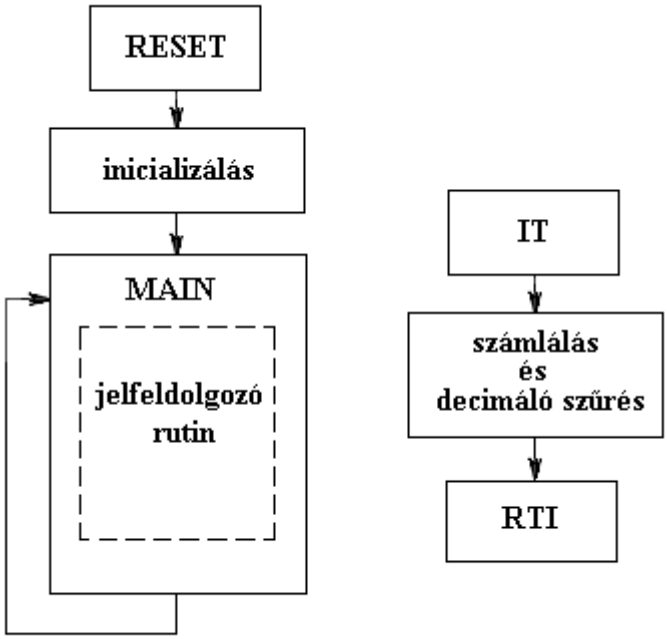
6.2.2. Decimálás

Sok jelfeldolgozási algoritmus számára nem megfelelő az, ha az audio alkalmazásoknál szokásosnak mondható több tíz kHz-es mintavételi frekvenciával mintavételezett adatokon futnak. Sok esetben néhány száz hertzes mintavételi frekvencia használatára van igény.

Ezért, amennyiben a rendszer hardver egységeinek minimálisan használható mintavételi frekvenciája túl magas, decimálást szokás alkalmazni. Az N-szeres decimálás azt jelenti, hogy az A/D felől érkező adatok mintavételi frekvenciáját N-ed

részére csökkentjük, így a feldolgozandó adatok száma is arányosan lecsökken. Ez konkrétan úgy történik, hogy a jelfeldolgozó rutin csak minden N-edik bejövő adat beérkezése után fut le. Ez a megoldás már magában hordozza azt, hogy a D/A számára is csak minden N-edik ütemben számítunk ki új adatot. A fennmaradó N-1 ütemben a D/A kimenetén az előző kiadott adat van, tehát egy nulladrendű tartó valósul meg.

Ha a jelfeldolgozási algoritmus megvalósításánál decimálást kívánunk alkalmazni, akkor érdemes kissé átrendezni a DSP program struktúráját. Ilyenkor a jelfeldolgozó algoritmust kedvezőbb a főprogramban elhelyezni, és csak akkor futtatni, mikor egy FLAG változó azt jelzi, hogy megérkezett a következő feldolgozandó adat. Az A/D megszakítást kiszolgáló rutinnak ekkor számláló funkciója van, amely vizsgálja, hogy a beérkező adatokat, és ha az aktuálisan beérkezett adat éppen az N-edik, akkor azt eltárolja, és beállítja az előbb említett FLAG változót. Ilyen módon nem lesz behatárolva a jelfeldolgozó rutin az eredeti mintavételi időközönként érkező megszakítások által. Ezt a működési struktúrát szemlélteti a 6.2.-es ábra.



6.2. ábra. DSP programstruktúra decimálás esetén

Alapvetően nincsen decimáló szűrő megvalósítva a rendszerben. Mivel a decimáló szűrőknek elég szigorú követelményeknek kell megfelelniük, ezért általában viszonylag nagy fokszámúak. Használatuk jelentősen csökkenti az alkalmazói program rendelkezésére álló futási idő. Ha szükség van használatára, akkor az ehhez tartozó digitális szűrést a megszakítást kiszolgáló rutinba kell elhelyezni, mivel ez az algoritmus még a magasabb mintavételi frekvenciával mintavételezett adatokon fut.

Általában nem szerencsés, ha a decimáló szűrést a keretprogram végzi el felhasználó elől elrejtve. Ugyanis elképzelhető olyan alkalmazás, ahol erre a szűrésre nincs is szükség, mert például a bemeneti jelek már eleve sávkorlátozottak; ezért kedvezőbb, ha a decimáló szűrést az alkalmazói program végzi el, amennyiben az adott esetben szükség van rá.

6.3. Soros kommunikáció

A nyolccsatornás rendszer két legfontosabb hardver egysége a vezérlő DSP, illetve az egy egységnek tekinthető codec-lánc. A szoftver réteg egyik fő feladata e két részegység kommunikációjának megszervezése és levezénylése. Az alábbiakban részletesen ismertetjük az egységek között definiált szinkron soros kommunikációs protokollt.

6.3.1. Az AD73322 soros portja

Az alkatrészválasztásnál sok más előnye mellett éppen azért esett a választásunk az AD73322-es codec-re, mert ez lehetőséget biztosított több codec kaszkádosítására. Ilyen típusú codec-ek használatakor nyolc egycsatornás eszköz összehangolt működésére van lehetőség. Fontos megjegyezni, hogy egy AD73322-es IC-n belül két egycsatornás codec van megvalósítva, tehát tulajdonképpen már egy darab önálló AD73322-es IC is kaszkád üzemmódban működik. Négy AD73322-es együttműködése esetén 8 codec 8 csatornát valósít meg párhuzamosan. A codec-ek kaszkádosítása [10] alapján történt.

Minden codec rendelkezik a következő lábakkal:

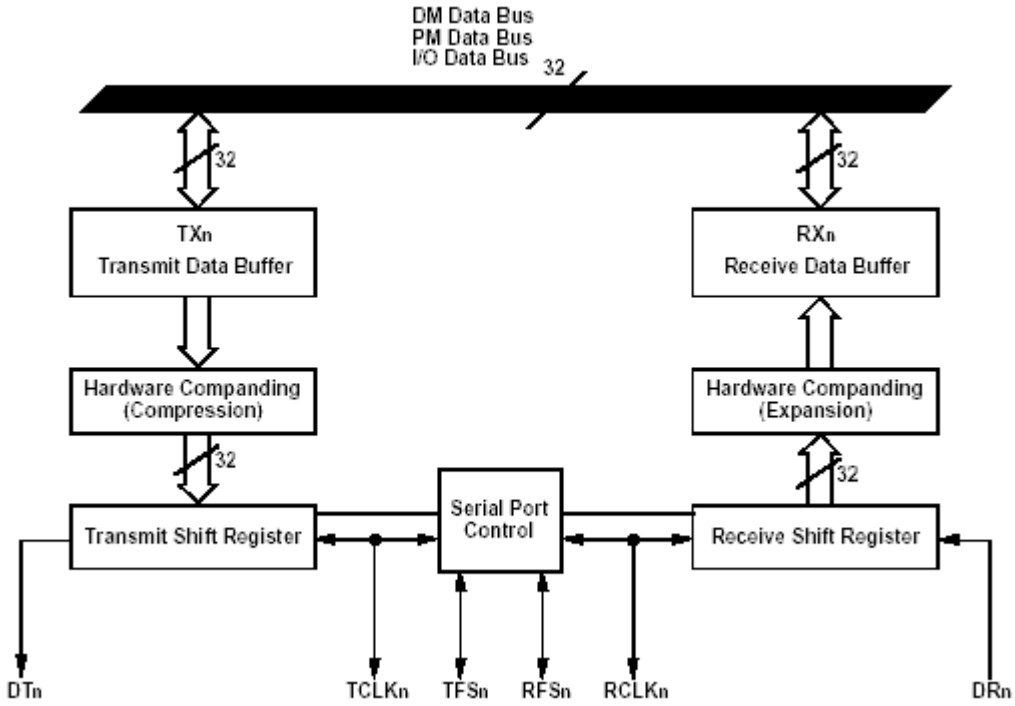
SDI	Serial Data In
SDO	Serial Data Out
SDIFS	Framing Signal Input for SDI serial Transfers
SDOFS	Framing Signal Output for SDO serial Transfers
SCLK	Serial Clock Output

V. táblázat. A codec soros portjának lábai

Az SDI és SDO lábakon történik a kétirányú soros adatkommunikáció. Az SDIFS és az SDOFS jelek a szinkron soros kommunikáció esetén használatos bemeneti- illetve kimeneti frame sync jelek. Az SDIFS láb a codec szempontjából bemenet, az SDI vonalon érkező 16 bites adatszó első bitjével egyidejűleg magas logikai szintűnek, egyébként alacsony logikai szintűnek kell lennie. Ehhez hasonló az SDOFS, ami a codec szempontjából kimenet, az SDO vonalon kiadott 16 bites adatszó első bitjével egyidejűleg egyes értékű, egyébként nulla. Az SCLK jel a soros kommunikáció szinkronizáló órajele, amelyet a codec biztosít.

6.3.2. A DSP soros portja

A DSP-k szinkron soros portja általában nagyon flexibilisen programozható, így a DSP-nek kell alkalmazkodnia a kevésbé rugalmas codec-ek által előírt kommunikációs protokollhoz. Az ADSP-21061 soros portjainak felépítése a 6.3. ábrán látható.



6.3.ábra. Az ADSP-21061 soros portja

Az ábrán látható lábak jelentése a következő:

DT	Data Transmit
TCLK	Transmit Clock
TFS	Transmit Frame Sync
RFS	Receive Frame Sync
RCLK	Receive Clock
DR	Data Receive

VI. táblázat. A DSP soros portjának lábai

A DSP soros portjának tehát ezeket a lábait kell olyan módon felprogramozni, hogy megvalósítsa az AD73322-es codec-ek által elvárt interfészt.

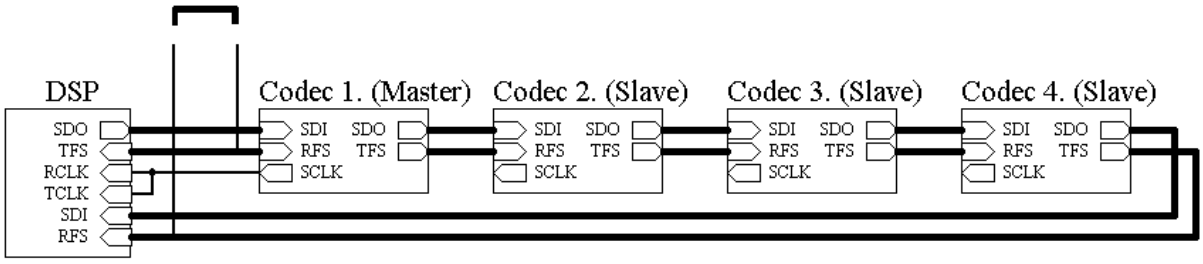
6.3.3. A soros kommunikáció

Az AD73322 codec-ek darabonként nyolc control regiszterrel rendelkeznek, amelyek segítségével működési paramétereik beállíthatók. A codec-ek RESET után ún. programozási üzemmódba kerülnek, ekkor kell a DSP-nek leküldenie az egyes codec-ekre a control regiszterek beállításait. Az összes regiszter felprogramozása után a codec-ek ún. adat módba kerülnek. Ebben az üzemmódban a control regisztereik értékét már nem lehet változtatni, tehát új értékek beállításához hardver RESET szükséges. Itt jegyezzük meg, hogy az AD73322 codec-eknek van egy ún. Mixed Program and Data üzemmódjuk is, amely során a regiszterek változtatása megoldható az adatkommunikáció közben is.

Rendszerünkben nem használjuk ezt az üzemmódot. Ennek használata olyan alkalmazás implementálásának esetén lenne mégis indokolt, ha a program futása közben mintavételi frekvenciát kellene váltani. A mi rendszerünkben ez úgy oldható meg, hogy a DSP futás közben RESET-eli a codec-láncot, majd leküldi a control registerek számára az új adatokat. (A codec-lánc közös RESET jele a DSP egyik FLAG lábához van kötve.)

Az AD73322 codec-ek előírják, hogy kaszkád működés esetén minden codec első kettő (A és B) control regiszterének ugyanazon értéket kell tartalmaznia. Ebben a két regiszterben találhatók ugyanis a kaszkád működés részleteit beállító bitek, például az, hogy hány codec van a codec-láncban. Ezekben a regiszterekben vannak a mintavételi frekvenciát beállító bitek is, ezért minden codec azonos mintavételi frekvencián működik. Az SCLK jelet a codec-ek a Master Clock (MCLK) órajelből osztják le, amelyet minden codec megkap. Ezek a leosztást beállító bitek is ebben a két regiszterben vannak, ezért minden codec azonos órajelhez szinkronizálva fogja adni, illetve várni az adatszavakat.

A codec-ek kaszkád működése az 6.4. ábrának megfelelő módon történik. A codec-ek a vezérlő DSP-vel együtt egy láncolatot alkotnak.



6.4. ábra. Codec-ek kaszkád működése

A lánc első tagja, a DSP számára az SCLK jelet adó codec a master, a többi slave.

A DSP külön TCLK és RCLK jelekkel rendelkezik, de esetünkben ezek össze vannak kötve, tehát mindkét láb bemenetnek van programozva, és a master codec SCLK jelét kapják. A többi codec számára azért nem kell biztosítani külsőleg ezt a soros órajelet, mert azt minden codec saját maga számára biztosítja. Ez azért igaz, mert a control regiszterekben ugyanaz az SCLK leosztás van beprogramozva minden codec-nek.

Az ábrán látható jumper azért van elhelyezve a TFS és az RFS jelek között, hogy választási lehetőség legyen a kommunikáció két irányának időbeli szétválasztására a [10]-ben ismertetett módon. Az általunk megvalósított megoldás esetében a jumper be van helyezve, tehát a frame sync jel közös. Ez azt jelenti, hogy a DSP és a codec-ek közti kommunikáció két iránya egyidejűleg zajlik le.

Az alábbiakban ismertetjük az adatkommunikáció konkrét lefolyását. A codec-ek az azonos control regiszter beállítások miatt szinkronizálva vesznek mintát az analóg bemenetekből. Ezt átalakítják egy 16 bites adatszóvá, és az SDOFS jellel szinkronizálva kiadják az SDO kimeneten. Ezzel egyidejűleg a lánc struktúra miatt minden codec megkapja az SDIFS jellel szinkronizálva a láncban felette elhelyezkedő codec által előállított és kiadott adatszót. A DSP DR lábán ekkor a lánc utolsó tagja által kiküldött adatszó érkezik. Ezzel egyidejűleg a DSP megkezdi a codec-eknek szánt adatok kiküldését is. Amiatt, hogy a DSP soros portjának RFS és TFS lábai

össze vannak kötve, az első adatszó kiküldése egyidejűleg zajlik az első adatszó beolvasásával.

Azt, hogy hány codec alkotja a láncot, minden codec „tudja”, ezért az első ütem után minden codec még ennyiszer (pontosabban a lánc hossza mínusz egyszer) továbbítja az SDO kimenetén azt az adatcsomagot, amelyet az előbb kapott az SDI bementén, tehát a lánc előző tagjától érkezett adatszót. Ilyen módon a DSP-be a codec-láncon lépésenként áthaladva, sorban beérkeznek a codec-ek adatai. Ezzel egyidejűleg, ugyanezen frame sync jelhez szinkronizálva folytatódik a codec-eknek szánt adatcsomagok kiküldése is. A mintavétel utáni nyolcadik frame sync után a DSP-be beérkezik a nyolcadik adatszó is, illetve a láncon keresztül utazó kimeneti adatfolyam célhoz ér, tehát minden codec-hez éppen a neki szánt adatszó érkezik meg.

Megjegyzés: a codec-ek számára az adatszavakat fordított sorrendben kell kiküldeni, tehát a master-nak szánt szót kell utoljára küldeni.

Az itt ismertetett folyamat minden mintavétel után lejátszódik. A 6.2. alfejezetben ismertetett működési struktúra olyan módon valósul meg, hogy az utolsó adatszó beérkezése után megszakításkérés érkezik a főprogram végtelen ciklusában futó DSP-nek, amely megszakítás kiszolgáló rutinjában vannak a jelfeldolgozó algoritmust végző utasításokat elhelyezve.

6.4. Szoftver eszközök feladatainak összefoglalása

Már az eddigiekből is kiderült, hogy az általunk fejlesztett többcsatornás rendszer szoftver rétegének több funkciót valósít meg párhuzamosan. Ezek a következők:

- Hardver inicializálása (boot)
  - DSP
  - Codec-lánc
- Adatkommunikáció biztosítása
  - DSP – codec
  - DSP – CPLD
- Alkalmazás beágyazása
  - hardver elrejtése

Egy konkrét program lefutásakor a fenti funkciók valósulnak meg. RESET után a inicializálni kell a rendszer intelligens egységeit, tehát a DSP-t, a codec-et valamint a CPLD-t. A DSP saját működési regisztereinek inicializálásánál fontos, hogy az egyes számú soros portot (SPORT1) olyan módon programozzuk fel, hogy a fentiekben ismertetett soros kommunikációs protokollt megvalósítsa.

Ennek lezajlása után a codec-lánc RESET-elése, majd a codec-ek inicializálása történik meg. E folyamat végén a codec-ek adat módba kerülnek és a program futása átkerül a fentiekben említett főprogramba.

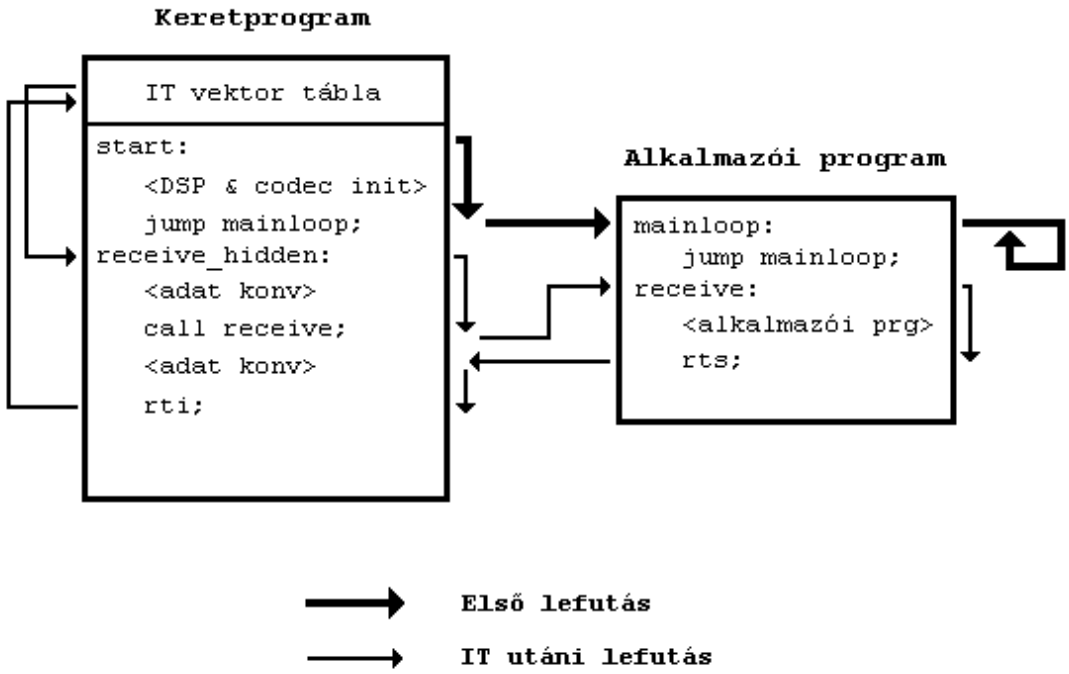
Az utolsó felsorolt funkció, az alkalmazások beágyazása, nem a program futása közben valósul meg, hanem inkább a DSP programok írásakor. A rendszer szoftver rétege ugyanis a jelfeldolgozó alkalmazást programozó mérnök szempontjából egy keretprogramként jelenik meg. Ezt a keretprogramot az alkalmazói szoftver forráskódjába be kell illeszteni (include). Ekkor az alkalmazói programnak nem kell a

rendszer megvalósításának hardver részleteivel törődnie, a keret a hardver vezérlését elvégzi helyette. A codec-ek felé irányú kétirányú adatkommunikáció részleteit elrejti, az alkalmazói programnak csak memóriaterületekről kell beolvasnia, illetve kiírnia az AD-ről érkező, illetve a DA-ra kiírandó adatokat. A DIP kapcsolók beolvasása, illetve a LED-ek vezérlése szintén ilyen módon történik.

6.5. Az elkészített szoftver eszközök

6.5.1. A program struktúrája

A fentiekben ismertetett feladatokat az általunk fejlesztett DSP program valósítja meg. A fejlesztésnél a 6.2.1.-es alfejezetben ismertetett decimálás nélküli struktúrát vettük alapul, ezt bővítettük ki. Az 6.5. ábra szemlélteti, hogy a DSP memóriájának kódterületére letöltődő konkrét program hogyan épül fel.



6.5.ábra. A DSP program struktúrája

- **Inicializálás**

A program futása RESET után a start címkére kerül, és megtörténik a DSP regisztereinek inicializálása. Ezalatt a DSP a codec-eket RESET állapotban tartja az egyik FLAG láb segítségével.

A DSP a saját regisztereinek inicializálása után leküldi a codec-ek számára a control regiszterek tartalmát. Az összes regiszter felprogramozása után a codec-ek adat üzemmódba (Data Mode) kerülnek. Az inicializálás végén engedélyezzük a megszakítás kérések érvényre jutását, majd a program futása a főciklusra tevődik át.

Az inicializálás során egy memóriacímkét rendelünk a külső buszhoz illesztett CPLD-hez, amely ezek után egyszerű memóriaműveletekkel elérhető.

- **Főprogram**

A főprogram a szokásos struktúrának megfelelően egy üres, önmagára ugró ciklus. A program futása egészen addig ebben a ciklusban marad, ameddig nem érkezik valamilyen megszakításkérés. Ekkor ez kiszolgálásra kerül, és lefut az adott megszakításhoz tartozó kiszolgáló rutin.

- **Soros kommunikáció megvalósítása**

A codec-ek felől érkező adatszavakat a DSP a soros porton keresztül fogadja. Az ADSP-21061 esetén mindkét soros porthoz két-két megszakítás kérési lehetőség van hozzárendelve. A „transmit" megszakítás kérés akkor lép fel, amikor a soros port TX regiszteréből kikerült az adat, tehát a regiszter újra feltölthető. A „receive" megszakítás kérés pedig akkor lép fel, amikor a soros port RX regiszterébe új adat érkezett. A jelfeldolgozói rutint ennek megfelelően a receive interrupt kiszolgálói rutinjában érdemes megvalósítani.

A 6.3. alfejezetben részletesen ismertetettük a DSP – codec soros kommunikációt. Ennek megfelelően az általunk tervezett nyolccsatornás rendszerben a mintavételezés után mind a nyolc adatszó egymás után megérkezik a soros portra. Kedvezőtlen lenne, ha minden adatszó után fellépne a megszakításkérés, mert akkor mindössze a soros óra (SCLK) periódusidejének megfelelő időtartam állna a rutin rendelkezésére. Ezért használjuk a processzor által támogatt DMA (Direct Memory Access) átvitelt.

- **DMA átvitel**

A DMA átvitel során a DMA vezérlő a soros portra beérkező adatokat továbbítja a belső memóriába. Ugyanígy a codec-nek szánt adatokat a memóriából a soros port felé továbbítja. Ez a megoldás azért rendkívül kedvező, mert a processzortól nem igényel külön processzoridőt az adatkomunikáció állandó menedzselése. A DMA átvitel inicializálása is a DSP saját vezérlőregisztereinek inicializálása során történik. Ekkor két memóriabuffert jelölünk ki, a „tx_buffer"-t és az „rx_buffer"-t. Mindkét memóriabuffer nyolc darab 16 bites adatszó tárolására képes. A DMA vezérlő az rx_buffer-be továbbítja a soros port RX regiszeterébe az egy mintavételi ütem után egymás után érkező nyolc adatszót. Ezzel egyidejűleg továbbítja a soros port TX regiszterébe a tx_buffer tartalmát adatszavanként. Mindez automatikusan történik, a processzor futásával párhuzamosan.

Ennél a működésnél a „receive" interrupt csak akkor lép fel, amikor a teljes rx_buffer megtelt, azaz nyolc adatszó beérkezése után. A „transmit" megszakítás kérés akkor lép fel, amikor a tx_buffer mind a nyolc adatát továbbította a DMA a soros portnak.

A jelfeldolgozói algoritmust ekkor is a „receive" megszakítást kiszolgáló rutinban érdemes elhelyezni. A megszakítás érvényre jutásakor az rx_bufferben egyszerre rendelkezésre áll a nyolc bemenet legfrissebb adata. A kiszolgáló rutinnak kell kiszámítania ezen bemeneti adatokból a nyolc kimeneti adatot. A kiszámított adatokat a tx_bufferbe kell eltárolni. Ezen adatokat a következő mintavétel után, az új adatok beérkezésével egyidőben fogja a DMA vezérlő soros portra kiadni.

A „receive" interrupt kiszolgáló rutinját két részre bontottunk, a 6.5. ábrán látható módon. A megszakító rutin érvényre jutása után a keretprogram részét képező „receive_hidden" programrészlet fut le, amely az rx_bufferbe érkezett adatokat numerikus konverzió után eltárolja az egyes csatornákhoz hozzárendelt

munkabufferekbe. Ezek a munkabufferek a belső memória egy-egy teljesen átlagos memóriarekeszeit jelentik. Miután ez lezajlott, „call” utasítás segítségével meghívjuk a „receive” címke alatt tárolt rutint. Ebben a rutinban kell megvalósítani a jelfeldolgozó algoritmust. Ez a rutin csak a munkabuffereket használja, és a kiszámított értékeket is a nyolc kimeneti munkabufferbe tárolja el. A rutin lefutása után a vezérlés visszakerül a „receive_hidden” programrészre, amely a kimeneti munkabufferek tartalmával elvégzi a fentivel ellentétes irányú munerikus konverziót, és a codec-eknek küldendő adatokat beírja a tx_bufferbe.

A „receive_hidden” programrészlet a keretprogram, a „receive” az alkalmazói program részét képezi. Ilyen módon megvalósul az alkalmazói programok beágyazása a rendszerbe. Az alkalmazói programnak (a keretprogram inculde-olása után) ugyanis elegendő a munkabufferekkel dolgozó „receive” rutint megvalósítania.

A szétválasztást még az is támogatja, hogy a receive_hidden – receive váltáskor átváltunk a másodlagos regiszterblokk használatára, hogy ne legyenek konfliktusok a regiszterek közös használatából.

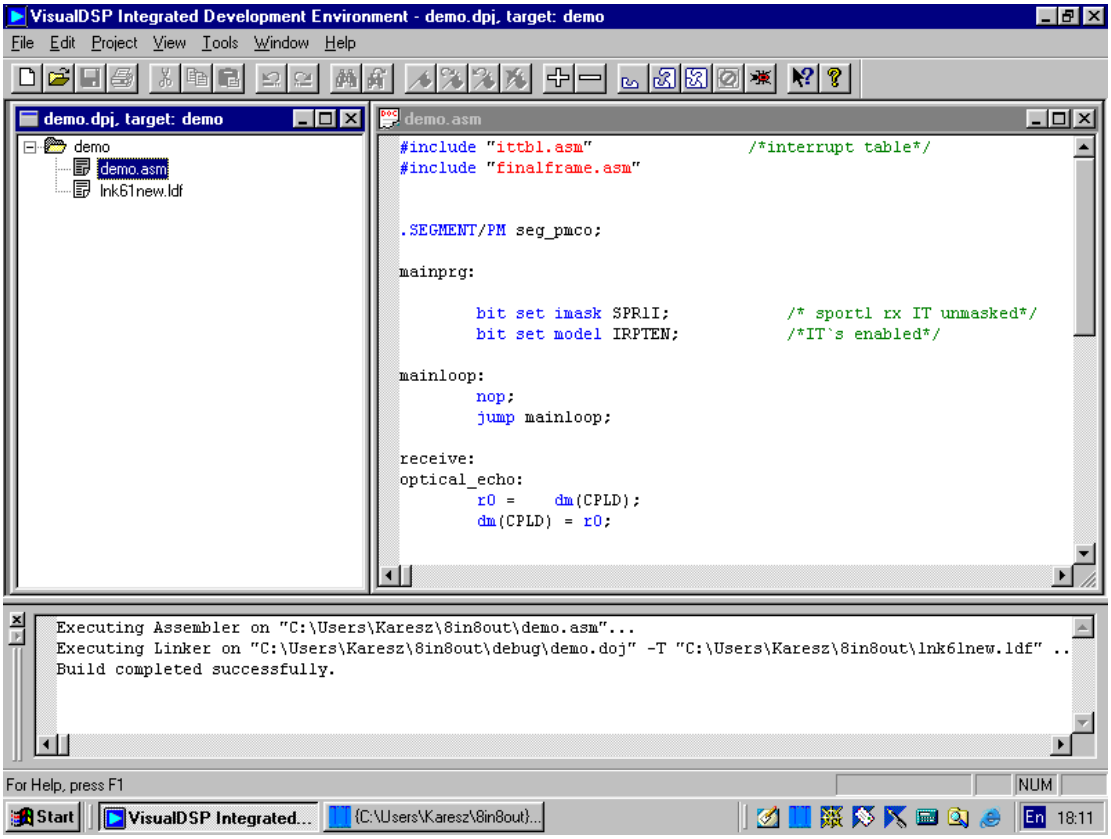
6.5.2. Fájlstruktúra

Az elkészített szoftver eszközök forráskódjai a következő három fájlban találhatók meg:

- finalframe.asm
- ldfnew.ldf
- ittbl.asm

Az „ittbl.asm” a DSP interrupt vektor tábláját tartalmazza. Az „ldfnew.dsp” a VisualDSP linker programjának szóló leírófájl. A nyolccsatornás codec-kártya használatához ezt a leírófájlt kell használni a default leírófájl helyett. A legnagyobb forrásfájl a „finalframe.asm” amely a DSP-n futó keretprogramot tartalmazza.

A felsorolt fájlokat úgy fejlesztettük ki, hogy feltételeztük a VisualDSP fejlesztői környezet használatát. Ilyen módon az alkalmazói szoftverek rendszerbe építése is úgy történik, hogy az alkalmazói program egy közös VisualDSP projektet alkot a fenti három forrásfájllal. Ez konkrétan olyan módon történik, hogy az alkalmazói program az LDF-fel együtt alkotja a projektet, a másik két forrásfájlt pedig include-olja. Ennek megfelelően a VisualDSP környezetben egy konkrét alkalmazás során a fájlok struktúráját az 6.6. ábra szemlélteti.



6.6. ábra. Fájlstruktúra VisualDSP környezetben

A fenti fájlstruktúrájú projektből a VisualDSP előállítja a DSP-re letölthető konkrét programot. A nyolccsatornás rendszer szofver keretét alkotó három forrásfájl közül a „frame.asm” és az „ittbl.asm” akkor is használható, ha az alkalmazói program nem használja a VisualDSP-t. Ilyen esetben az LDF használata nem lehetséges, kiváltása kis kényelmetlenségek árán megoldható.

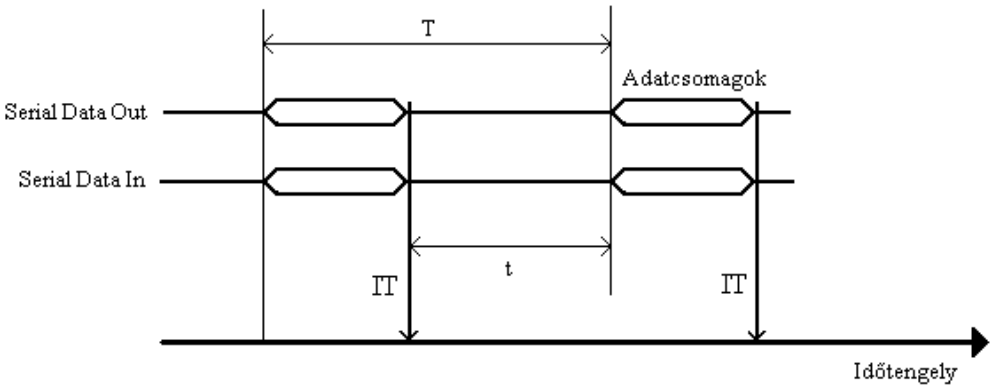
6.6. Időzítési számítások

A rendszer képességeinek vizsgálatakor érdemes megvizsgálni, hogy milyen hosszú lehet maximálisan a jelfeldolgozó rutin, ha biztosítani kívánjuk az adatvesztésmentes on-line működést. Ez akkor teljesül, ha a „receive” megszakításhoz tartozó kiszolgálórutin hamarabb lefut, mint hogy a következő ugyanilyen megszakításkérés beérkezne.

6.6.1. Egyszerű működés

A 6.5.1. alfejezetben részletesen kifejtett okokból a DMA vezérlő akkor szakítja meg a processzor futását, amikor mind a nyolc bejövő adatot a memóriába írta. Ezen interrupt kérés tehát a codec-ek mintavételi frekvenciájának megfelelő gyakorisággal fog jelentkezni. Mivel az adatküldést és az adatfogadást a DMA vezérlő egyidejűleg végzi, a processzornak a kimeneti adatokat elő kell állítania azon időpillanatig bezárólag, amikor a következő adatcsomag fogadása megkezdődik. A processzornak tehát e két időpillanat közötti időtartam áll rendelkezésre arra, hogy a megszakítást kiszolgáló rutinban feldolgozza a nyolc bementi adatot, és előállítsa a kimenetek

számára a nyolc kimeneti adatot. Az 6.7. ábrán tüntettük fel az időzítési viszonyokat. A számítások során először azt az esetet vizsgáltuk, mikor a codec-ek a maximális 64 kHz-es mintavételi frekvenciával üzemelnek.



6.7. ábra. Az egyszerű működés időzítési viszonyai

Jelölések: T a mintavételi idő, t a lefutás számára rendelkezésre álló idő.
Az adatcsomag nyolc darab 16 bites adatszóból áll, amelynek beolvasása az SCLK jellel szinkronizálva történik. A rendszerben az SCLK maximális frekvencián, 16 MHz-en üzemel.
Számítások:

$T = 1/64\ \text{kHz} = 15.625\ \mu\text{sec}$
$t_{adatcsomag} = 8 \cdot 16 \cdot (1 / SCLK) = 8\ \mu\text{sec}$
$15.625\ \mu\text{sec} - 8\ \mu\text{sec} = 7.625\ \mu\text{sec}$

Ekkor az IT rutinnak 7.625 μsec idő áll rendelkezésére ahhoz, hogy lefusson. A maximális 50 MHz-es órajel mellett egy DSP utasítás ideje 20 nsec. (Egy utasítás ciklusonként.) Ez azt jelenti, hogy a megszakításrutin legfeljebb 381 utasításból állhat. Csatornánként 47 utasításba a különböző előkészítő és lezáró műveletek miatt körülbelül egy 40 együtthatós FIR szűrés fér bele. Az előbbi számítás eredményeit összefoglalva mutatatja a 4. táblázat.

	f_{MV} = 64 kHz	f_{MV} = 32 kHz	f_{MV} = 16 kHz	f_{MV} = 8 kHz
Egyszerű működés	381 (47)	1162 (145)	2725 (340)	5850 (731)
Kibővített működés	781 (97)	1562 (195)	3125 (390)	6250 (781)

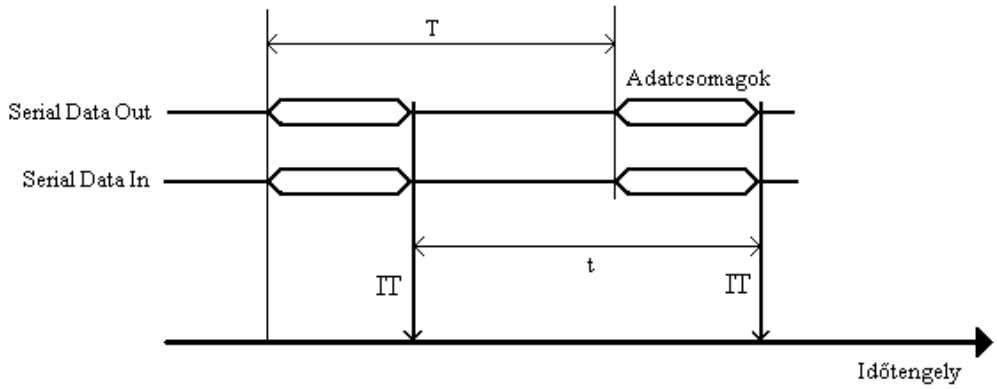
IV. táblázat. Megszakítási rutin utasítások maximális száma (csatornánként)

A csatornánkénti utasítások száma természetesen általában több a kiszámított értékeknél, hiszen ritka az az eset, amikor minden csatorna megvalósít valamilyen jelfeldolfgozó algoritmust, de a számítások során természetesen azt vizsgáltuk, mekkora a rendszer maximális terhelhetősége.

6.6.2. Kibővített működés

Az eddig ismertetett értékek a legtöbb alkalmazás számára elegendők, mert általában nincs szükség a legmagasabb mintavételi frekvencia használatára. Ha mégis ilyen igény merül fel, akkor az alábbiakban ismertetett megoldás sok esetben célravezető lehet.

A megszakítás kiszolgáló rutinjának futási ideje megnövelhető, ha megengedünk még egy mintavételi ütemnyi késleltetést. Ugyanis az előbbi esetben a futást azért kellett befejezni a következő ütem kommunikációjának megkezdődése előtt, hogy a kimeneti adatok már rendelkezésre álljanak az adatok kiküldésének megkezdésekor. A következő megszakítás kérés viszont csak ezen új adatcsomag teljes beérkezését követően fog fellépni, így a következő adatok feldolgozását csak ezt követően kell megkezdeni. Ezért a jelfeldolgozó rutin egészen az új megszakításkérés fellépéséig bezárólag futhat. Az ilyen módon nyert többlet futási idő ára az, hogy ennél a működésnél a kimenet két minta késésben lesz a bementhez képest, az előző működés egy mintányi a késleltetésével szemben. Ez az egy mintás többletkésleltetés egy nagyságrendbe esik a szigma – delta codec késleltetésével. Az 6.8.ábrán követhetők ennek a működésnek az időzítési viszonyai.



6.8.ábra. A kibővített működés időzítési viszonyai

Jelölések: T a mintavételi idő, t a lefutás számára rendelkezésre álló idő.

A 4. táblázatban az ehhez a működési módhoz tartozó maximális utasításszámokat is feltüntettük. Látszik, hogy jelentős nyereséget csak a nagy mintavételi frekvenciás működés esetén jelent a kibővített működési mód.

7. fejezet

A rendszer bemérése

Az élesztés művelete párhuzamosan történt az alkatrészek beültetésével. Mivel első verzióról van szó, a lehetséges tervezési hibákból következő alkatrész károsodások elkerülése érdekében nem egyszerre ültettük be az összes alkatrészt, hanem több lépésben.

Mivel a rendszer hardver és szoftver részegységeinek működése nem függetleníthető, ezért a rendszer bemérése során a két részegységet lépésről-lépésre, folyamatosan bővítettük.

7.1. A nyomtatott áramkör

Először is az elkészült nyomtatott áramkört tüzetesebben át kellett vizsgálni. Alapvetően a gyártás során keletkezett hibákat kerestünk. Megvizsgáltuk, hogy az egyes földszigeteken belül elhelyezkedő, de nem a földhöz tartozó alkatrészlábak valóban el vannak-e szigetelve a földszigettől. Ellenőriztük továbbá, hogy mindegyik IC táp-pontja helyes polaritással és alacsony (közel 0 Ohm) impedanciával csatlakozik-e a tápegységhez. Szintén ellenőrizni kellett, hogy a DSP kártya általunk felhasznált csatlakozóinak pozíciói azonosak-e (±0.1 mm) az általunk tervezett kártyáéval. Miután az összes vizsgálatnak megfelelt a nyomtatott áramkör, megkezdődhetett a beültetés.

7.2. A tápegység

A beültetés első lépéseként a tápegységet forrasztottuk fel. Azért került előre a tápegység, mert ha bármi hiba fellép benne, az összes már beültetett, jól működő egységet tönkretehette volna. Miután az egység alkatrészei a helyükre kerültek, következhetett az első élesztési lépés. A tápegység bemenetére ±12 V-ot adva bekapcsoltuk, és mértük az egység áramfelvételét és a stabilizátorok kimeneti feszültségét is. Az áramfelvétel mértéke fontos információval szolgált számunkra. Ha bármi rendelleneset tapasztaltunk volna, azonnali kikapcsolás után tüzetesebben át kellett volna vizsgálni az egységet. Rendellenességek lehettek volna többek között:

- Nagy áramfelvétel, amely meghaladja a stabilizátor és a tápfeszültség jelző LED által felvett áramot. ($I_{felv} <= I_{led} + I_{stab}$ = 6-15 mA)
- Aszimmetrikus áramfelvétel, tehát, ha a két ág áramfelvétele jelentősen eltért volna egymástól.
- A felvett áramnak a bekapcsolás után stabilizálódnia kell a normális érték környezetében, ellenkező esetben, ha például növekszik, fordított

polaritással bekötött elektrolit kondenzátorra kellene gyanakodnunk, ami típushiba a dupla tápfeszültségű rendszerek beültetése esetén.

Az áramfelvétel mellett a stabilizátorok kimenő feszültségének jellemzőire is vannak kikötések. A stabilizátorok kimenő feszültségét feszültségmérő és oszcilloszkóp segítségével vizsgáltuk. Először is a kimeneti feszültségeknek az előírt +5 V és –5 V környezetében kell lenniük, az eltérés csak pár 10 mV lehet. Másodszor az előírt tápegység bemeneti feszültségtartomány mindkét végértéke esetén biztosítaniuk kell a stabilizátoroknak az előírt kimeneti feszültséget. Tehát a kimeneti feszültség egy kis változástól eltekintve állandó legyen, ami bizonyítja a stabilizátorok kimenő feszültségeinek a betáplálástól való függetlenségét. Mindemellett a kimenő feszültségeknek függetlennek kell lenniük a terheléstől is. Ennek ellenőrzését próbaterhelésekkel, és a későbbiekben a működő kártya tápfeszültségeinek üzem közbeni mérésével végeztük. Oszcilloszkóppal a stabilizátorok által kiadott feszültségnek a "tisztaságáról", zajmentességéről kaphatunk információt.

7.3. Az analóg ki-, és bemeneti egységek

Miután a tápegység helyes működéséről meggyőződtünk, beültettük az első analóg ki-, bemeneti egységet. A beültetés után ismét élesztési fázis következett. Bekapcsoláskor figyeltük az áramfelvételt, és ezt minden új egység kipróbálásánál a későbbiekben is meg tettük. Mivel a vele kapcsolatban lévő codec még nem volt beépítve ebben a fázisban, a codec által adott referencia feszültséget két nyitó irányban bekötött diódával és egy előtét ellenállással helyettesítettük. Az így létrehozott kapcsolás előállítja a megfelelő feszültséget, aminek a segítségével már működőképes lesz az analóg ki-, bemeneti egység. A bemenetre kapcsolt függvénygenerátor segítségével előállítottuk a gerjesztést, és oszcilloszkóppal vizsgáltuk az áramkör fontosabb pontjait:

- Megtörtént a referencia jel bufferelése, és kétszerezése.
- Az átalakított szimmetrikus jelek, valóban a referencia feszültséggel vannak eltolva.
- A szimmetrikus jelek ellenfázisúak.
- Az áramkör megfelelő bemenet esetén nem torzít.
- Az átalakított jel nem tartalmaz az elvárhatónál magasabb zajt
- A –3 dB-es töréspontok 32 kHz környékén vannak.

Mivel az analóg kimenetet szimmetrikus jellel lehet vezérelni, az analóg bemenet szimmetrikus jellé alakított pontjait kötöttük össze az analóg kimeneti egység bemeneteivel. Az így létrejött áramkört a bemenetre korábban rákapcsolt függvénygenerátor segítségével lehetett gerjeszteni, és a kimeneti egység kimenetén mérni az eredményt. Miután AC-csatolt esetben ellenőriztük az egységet, következhetett a DC-csatolás ellenőrzése is. Ezt a funkciót ofszettel rendelkező szinuszjel segítségével vizsgáltuk. Mivel az ofszetet helyes polaritással vitte át a rendszer, ez a funkció is működőképes volt. Ezek után a többi analóg ki-, bemeneti egységet is beültettük, és a fentiekben leírtak szerint élesztettük egyenként.

7.4. A codec-ek

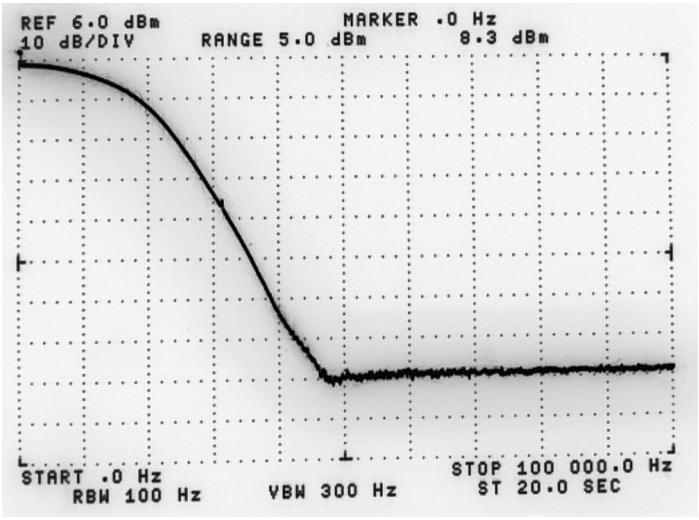
A következő fontos lépés a master codec felforrasztása, és élesztése volt. A műveletet a kártyára feltervezett 16.384 MHz-es oszcillátor (ami helyett beszerzési problémák miatt 16 MHz-es eszközt alkalmaztunk) beüzemelésével kezdtük, majd pedig a DSP SPORT1 soros portjának ellenőrzésével folytattuk. Még egyszer ellenőriztük a két kártyán lévő jelek helyes kapcsolatát is. A két kártya összekapcsolása után figyeltük az együttes áramfelvételt, ami nem lehetett nagyobb, mint a kártyák önálló áramfelvételének összege, ettől eltérő esetben hibára kellett volna gyanakodnunk. Következő lépésként a codec megkapta a DSP-től a RESET és az SE jeleket, aminek hatására az SCLK jelből létrehozta a nyolccal leosztott MCLK jelet. Miután oszcilloszkóppal ellenőriztük a helyes frekvenciát, jelalakot, és azt is, hogy a DSP is megkapja, az élesztést tovább folytattuk. Ellenőriztük, hogy a codec megadja-e a megfelelő kommunikációhoz szükséges jeleket (TFS, RFS). Mindezek után következett a konfigurációs szavak letöltése a codec-be. Mivel a letöltés után a codec aktiválta a referencia kimenetét és kiadta a kívánt értéket, meggyőződhettünk a kommunikáció és a soros port helyes működéséről. A tesztelés alatt nagy segítségünkre volt a codec két speciális funkciója, amit a konfigurációs szavak egy-egy bitjével lehetett bekapcsolni. A funkciók név szerint az analóg, illetve a digitális echo. Ezek lehetővé teszik, hogy a codec-en belül a bemeneti láncot megfelelő helyeken összekapcsoljuk a kimeneti jelúttal, és így a DSP nélkül is elvégezhető volt több mérés. Miután a DSP DMA segítségével sikeresen kommunikált az első codec-kel, tovább léphettünk a többi codec élesztése felé.

A codec-eket továbbra is egyesével élesztettük a lehetséges problémák elkerülése miatt, a már korábban megszerzett tapasztalatok alapján. Az élesztés során csak az MCLK osztási arányát kellett egyre állítani, hogy a megnövekedett sebesség következtében az adatcsomagok elférjenek. A fogyasztás is növekedett értelemszerűen, de hirtelen növekedést nem tapasztaltunk. Miután minden codec felélesztésre került, ellenőriztük, hogy a csatornák függetlenek-e egymástól.

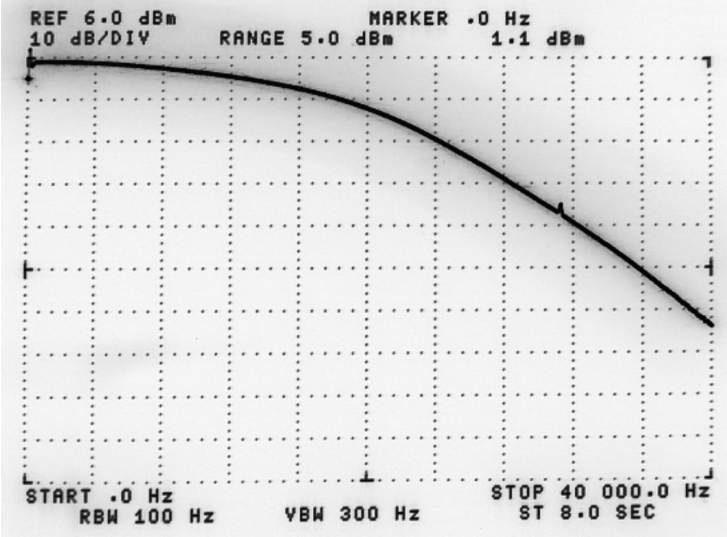
7.5. Mérési eredmények

A negyedik codec elhelyezése után Hewlett–Packard 3585B típusú spektrumanalizátorral vizsgáltuk meg az egyes csatornák átvitelét. A DSP-n futó alkalmazás a mérések során egyszerű echo volt, azaz a DSP-be beérkezett digitális adatszavakat módosítás nélkül írtuk vissza a bemenetnek megfelelő sorszámú kimenetre.

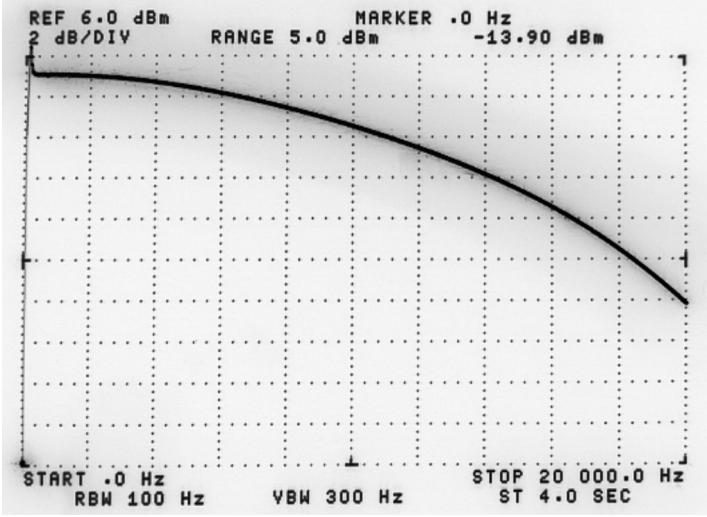
A mérést az összes csatornán elvégeztük, a kapott eredmény mind a nyolc esetben megegyezett. A 7.1., 7.2. és a 7.3. ábrákon a hetes sorszámú csatorna átvitelének mérési eredményei láthatók 64 kHz-es mintavételi frekvencia mellett.



7.1.ábra. Átviteli karakterisztika abszolútértéke 0 – 100 kHz között



7.2. ábra. Átviteli karakterisztika abszolútértéke 0 – 40 kHz között

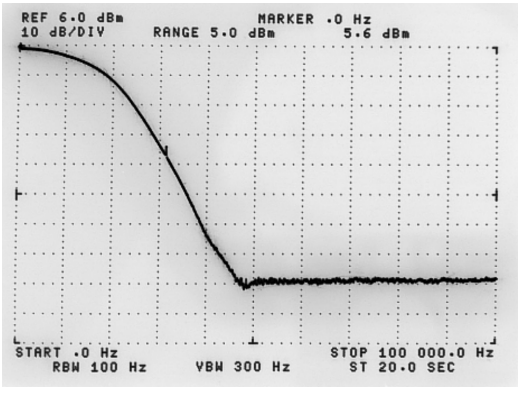


7.3. ábra. Átviteli karakterisztika abszolútértéke 0 – 20 kHz között

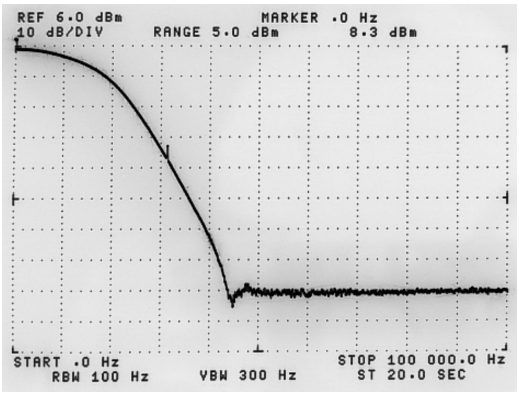
Ezek a karakterisztikák megfelelnek az AD73322-es codec adatlapján közölt specifikációval azzal a különbséggel, hogy a kimenti jelformáló egység tartalmaz egy elsőfokú aluláteresztő szűrőt, amelynek karakterisztikája is érvényesül. Ennek a szűrőnek törésponti frekvenciája 32 kHz.

A frekvenciameneten a mintavételi frekvencia felénél látható kiugrás a mérési összeállítás sajátossága, nem része a valós karakterisztikának.

A mérések során azt tapasztaltuk, hogy az egy codec-hez tartozó két csatorna karakterisztikái között minden codec esetében eltérés mutatkozott 43 kHz környezetében. Ez az eltérés a 7.4.-es ábrán figyelhető meg. Ennek oka jelenleg nem ismert, ha a jövőben ez a – jelentéktelennek tűnő – eltérés gondot okoz, további mérésekkel kideríthető az oka.

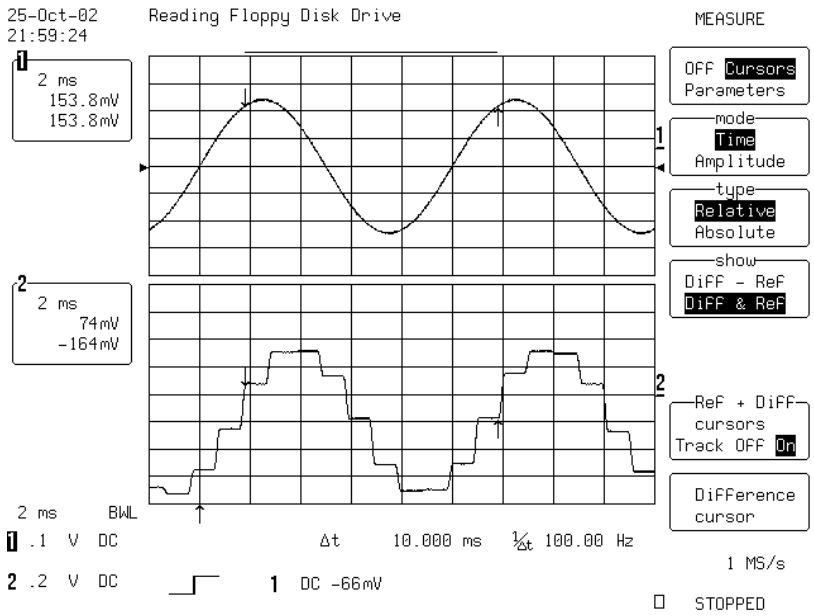


7.4.a. ábra. Bal csatorna



7.4.b. ábra Jobb csatorna

A 7.5.-ös ábrán látható a 6.2.2. pontban bemutatott decimáló működés. A felső görbe a bemenet, amely egy 100 Hz-es szinuszjel. Az alsó görbe a kimenő jel, amely a bejövő jel decimálás után. Látható, hogy a decimálás nyolcszoros, tehát minden nyolcadik ütemnél van új adat kiszámítva, a többi ütemben nulladrendű tartó valósul meg. A mérést, illetve a görbe eltárolását LeCroy LT342 digitális oszcilloszkóp segítségével végeztük.



7.5. ábra. Decimálás

7.6. A digitális I/O

Az utolsó fázisként a CPLD köré épülő digitális I/O-t élesztettük föl. Az élesztést a codec-ekhez hasonlóan a DSP perifériáinak ellenőrzésével kezdtük. Legmagasabb wait state értéket beállítva a lehető legjobban lelassítottuk a párhuzamos periféria általunk használt részét, a fejlesztés megkönnyítése érdekében. Egyszerre csak egy funkciót próbáltunk ki. Először a kimenetet és a hozzá tartozó VHDL kódot dolgoztuk ki, majd mértük le. Ellenőriztük, hogy minden egyes írási kísérlet a DSP részéről sikeresen végbemegy-e. Az írást egy sűrűn frissített, de lassan számláló állapotának kiiratásával végeztük. Digitális oszcilloszkóppal rámérve az egyes kimenetekre, figyeltük az impulzus hosszát, valamint azt, hogy ne legyen benne hazárdra, pillanatnyi rossz beírásra utaló ellentétes tüske. A digitális egységek élesztésénél figyeltük az analóg részek jelalakjait, táppontjait is, hogy nem csatolódnak-e be túlzottan a digitális egységek okozta zavarok. Miután a kimenet működőképességéről meggyőződtünk, a digitális bemenetet is felélesztettük. A digitális bemeneteket minden egyes ciklusban kimásolva a digitális kimenetekre, és a bementeket TTL jelgenerátorral gerjesztve ellenőrizhettük azt, hogy a DSP minden egyes ciklusban biztonságosan olvas a bemenetekről.

7.7. Hosszú idejű stabilitás

A kártyánkat az egyes egységek tesztelése után egy stabilitási tesztnek is alávetettük, amely során figyeltük a rendszer hardver elemeinek és az azt kezelő szoftvernek a stabil működését. A DSP teszt programját úgy írtuk meg, hogy minden egyes mintavételkor az adott csatorna bejövő értékét adja ki a kimenetre, és a digitális bemeneten beolvasott értékeket írja ki a digitális kimenetre. A rendszer bemeneteit generátorokkal gerjesztve, és a kimeneteit oszcilloszkópokkal figyelve üzemelt a kártya huzamosabb ideig, mialatt még a tápáram felvételét is mértük.

8. fejezet

Alkalmazói program implementálása

Mint azt már korábban említettük, a kártya megtervezésének és megépítésének alapgondolata az volt, hogy a kereskedelmi forgalomban kapható DSP eszközök, bár néhány esetben igen nagy számítási kapacitást kínálnak, azok kihasználására nem nyílik lehetőség, mivel csak korlátozott számú, többnyire egy sztereó be-, illetve kimenettel rendelkeznek (igaz, hogy szép számmal találhatunk többcsatornás kártyát is, ám ezek ára igen magas). Ez azért is zavaró, mivel a tipikus jelfeldolgozó alkalmazások ─ mint például mérésadatgyűjtés, audio berendezések, akusztikai rendszerek ─ jellegükből adódóan sok bemeneti jelet fogadnak és kimeneti jelet állítanak elő.

A kártya létjogosultságának igazolására, illetve képességeinek bizonyítására egy komplex eljárás megvalósítását mutatjuk be lépésről lépésre. A demonstrációra az XLMS algoritmust választottuk, amelyet igen elterjedten használnak különféle audio és szabályozástechnikai problémák megoldásakor, illetve a jelfeldolgozás legkülönbözőbb területein: echo cancellation telekommunikációs eszközökben, zajcsökkentő eljárások [12]. A nyolccsatornás rendszerben egy aktív zajszűrő alkalmazás épült meg, amely sztereó kivitel esetén négy bemenetet és két kimenetet igényel. Így bár nem használtuk ki a rendszer kínálta összes bemenetet és kimenetet ─ melynek oka, hogy a laborban megfelelő hardver híján eddig nem volt lehetőség többcsatornás rendszeren való fejlesztésre ─, de segítségével egyszerűen és szemléletesen bemutatható egy algoritmus jelfeldolgozó processzorokon való megvalósításának számos fontos problémája is. Az XLMS esetleges nagy számítási igénye (amely részben adaptív jellegéből, részben a benne megvalósított nagy fokszámú FIR szűrőkből adódik) felveti annak a lehetőségét, hogy különböző DSP eszközök teljesítőképességének mérésére, összehasonlítására alkalmazzuk.

8.1 Az LMS algoritmuscsalád

Az adaptív szabályzók számtalan típusát kifejlesztették az évek során, de máig klasszikusnak számítanak az LMS, illetve a belőlük továbbfejlesztett algoritmusok. Az eljárás célja az, hogy a hibajel teljesítményét a legkisebbre csökkentse (Least Mean Square). Az LMS alapú megoldások elterjedtségének az az oka, hogy néhány igen jó jellemzőjük kiemeli a másfajta megvalósítások közül:

- Robusztusság: a paraméterek kismértékű változására stabilak maradnak
- Egyszerűség: számításigényük a többi megbízhatóságában, pontosságában az LMS algoritmushoz mérhető eredményt produkáló eljárásokhoz képest alacsony, így egyszerűbb, ezért olcsóbb processzorral is implementálhatók
- Maradó hiba: megfelelő paraméterválasztással alacsony értéken tartható

- Konvergenciasebesség: viszonylag nagy, gondot jelenthet, hogy a sebesség növelése a maradó hiba növekedése és a stabilitás csökkenésének irányába hat

Az emellett matematikailag is igen jól leírt, könnyen kezelhető eljárások további előnyös tulajdonsága, hogy az iteratív modellillesztés elméleti levezetésében előírt, de a gyakorlatban nehezen kezelhető statisztikai paramétereket pillanatnyi értékekkel helyettesítik, ezáltal a bonyolult számítások, illetve az információhiány megkerülhető [13].

Ebben a dolgozatban a két leggyakoribb, a megépült rendszerben is használt típus ismertetésére szorítkozunk.

8.1.1 Az LMS algoritmus

Az LMS algoritmus (8.1. ábra) a beavatkozó jelet a referenciajel adott számú késleltetett mintájának súlyozásával állítja elő. Így központi eleme egy N együtthatós transzverzális FIR szűrő, amelynek együtthatóit úgy adaptáljuk, hogy a négyzetes hiba minimális legyen. Az *u(n)* beavatkozó jel ezért a következő formában áll elő:

$$u(n) = \sum_{i=0}^{N-1} w_{n-i} x(n),$$

$$u(n) = w * x(n),$$

ahol *w* a szűrő impulzusválaszát tartalmazó vektor. Az algoritmus másik bemenőjele a rendszerben jelenlévő zavarjel, amelyet az eljárással csökkenteni szándékozunk, oly módon, hogy a két jelet különbségképzőbe vezetve megvizsgáljuk, hogy az algoritmus az adott ütemben mekkora hibával közelítette a *d(n)* jelet az *u(n)* beavatkozó jellel, majd az ábrán LMS címkével jelölt blokk az *x(n)* referenciajel és az *e(n)* hibajel segítségével számolt mértékű korrekciót végez az adaptív szűrő együtthatóin, és a következő ütemben ezzel az új készlettel számolja a beavatkozó jelet. Az algoritmus helyes működése esetén a hiba csökken, a rendszer egyre jobban követi a *d(n)* jelet. A hibajel a *d(n)* zavar és az *u(n)* beavatkozó jel összegeként írható fel:

$$e(n) = d(n) - u(n),$$

$$e(n) = d(n) - w * x(n),$$

$$e(n) = d(n) - \sum_{i=0}^{N-1} w_{n-i} x(n)$$

Mivel a hibajel teljesítményét szeretnénk a lehető legkisebbre csökkenteni, ezért a minimalizálandó költségfüggvény a hibajel négyzetének várható értéke:

$$J = E\left\{e^2(n)\right\}$$

Mivel e kifejezés paramétereinek értéke a legtöbb esetben nem ismert, a minimumot meg kell keresni, amelyre adaptív algoritmust alkalmaznak. A négyzetes hibát minimalizáló *w* értékét egy N+1 dimenziós parabolafelületen keressük (ennek

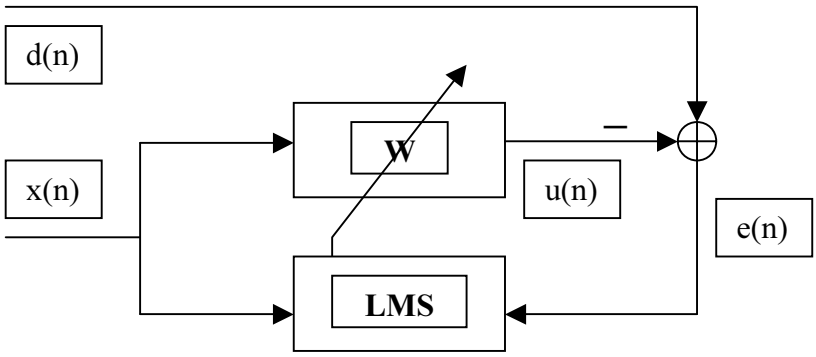
oka, hogy a hibajel négyzetének kifejezésébe *e(n)* fenti definícióját megfelelően behelyettesítve *w* együtthatóban N négyzetes kifejezéshez jutunk), így a minimumot egy egyszerű, a legmeredekebb úton leereszkedő algoritmus megtalálja:

$$w(n+1) = w(n) + \alpha \frac{dJ}{dw(n)},$$

ahol α a lépés mérete az N+1 dimenziós parabolafelületen. A fenti egyenletbe *J*, illetve *e(n)* definícióját behelyettesítve a rekurzív algoritmus

$$w(n+1) = w(n) + \mu \cdot e(n) \cdot x(n)$$

alakúra változik, ahol $\mu = 2\alpha$ a konvergencia-együttható. Ez az algoritmus az LMS algoritmus.



8.1. ábra. Az LMS algoritmus blokkvázlata

Az ezzel az algoritmussal megvalósított rendszer stabilitása függ μ nagyságától, annak 0 értékénél természetesen nincs konvergencia. Növekvő μ-t választva a konvergencia sebessége eleinte nő, azonban egy adott optimális értéket meghaladva a rendszer sebessége fokozatosan csökken, majd egy maximális értéknél nagyobb μ-t alkalmazva a rendszer instabillá válik.

Léteznek olyan alkalmazások, amelyekben az LMS algoritmus nem használható, ugyanis a rendszer hiányossága, hogy a jelek különböző úton terjednek, így feldolgozásuk, például összegzésük esetén a fázisuk az adott alkalmazásban a megengedhetőnél jobban eltér, amely hibás eredményre, a rendszer instabilitásához vezethet.

8.1.2 Az XLMS algoritmus

Az LMS algoritmus hátránya, hogy nem alkalmazható olyan esetekben, amikor nem közvetlenül használható (esetleg nem hozzáférhető) a kimenete, például, ha egy mikrofont alkalmazunk két akusztikus jel különbségének képzésére (ellentétes fázisú jelek összegzésére), akkor a különbségképzőbe nem a hangszórót meghajtó jel kerül, hanem annak az elektromos-akusztikus átalakítás, illetve az akusztikus átvitel során módosult változata. Ezt a problémát oldja meg az XLMS algoritmus (8.2. ábra), amely az LMS algoritmushoz hasonlóan az *x(n)* referenciajelből a *W* adaptív szűrő

segítségével számítja az *u(n)* beavatkozó jelet. Az ily módon előállított beavatkozó jelet ezután D/A átalakítóval villamos jellé alakítják, majd aluláteresztő szűrővel simítják. A kapott jelet felerősítik, majd a megfelelő beavatkozó szerven keresztül a szabályozni kívánt *A* rendszert hajtják meg vele. A hibaérzékelő által mért jelet is erősíteni kell, majd D/A átalakítóban az átlapolást kerülendő aluláteresztő szűrővel szűrik. Az így kapott jelet mintavételezik és kvantálják. A rendszer tehát számos késleltetéssel és egyéni átvitellel rendelkező elemből épül fel, amely összességében lineárisnak tekinthető, ha az elemek teljesítenek bizonyos feltételeket: az erősítők torzítása és zaja legyen kicsi, mind az erősítők mind az analóg-digitális és a digitális-analóg átalakítók esetében kerülni kell a túlvezérlést, mivel az jelentős nemlinearitást vinne a rendszerbe. A minél magasabb jel-zaj viszony érdekében az átalakítók felbontása legyen elegendően nagy, és a bemenő jelek minél jobban közelítsék a maximális kivezérlést, ellenkező esetben a felső bitek a felhasználás szempontjából értéktelenek lesznek. Amennyiben a rendszer bemenetén nem szigma-delta A/D átalakítót alkalmazunk, a vezérlő jel spektrumát a Nyquist-frekvencia alatti tartományra kell korlátoznunk. Most is a *d(n)* jel hordozza a környezetből származó zavarjelet, ezt a referenciajellel összegezve áll elő az *e(n)* hibajel, ennek négyzetét minimalizálja az eljárás úgy, hogy az LMS algoritmus a hibajel mellett most az *A* rendszert modellező $\hat{A}$ szűrővel szűrt *r(n)* jelet használja (neve is innen adódik: filtered-x LMS). Ekkor az algoritmus rekurzív formulája az alábbi alakúra változik:

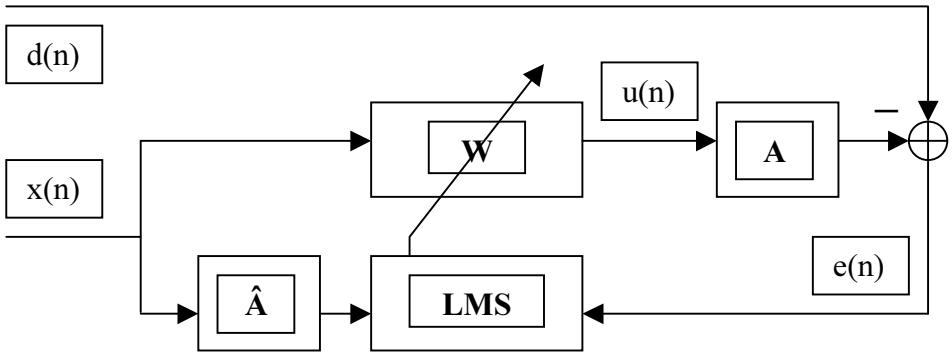
$$w(n+1) = w(n) + \mu \cdot e(n) \cdot r(n),$$

ahol *r(n)* a referenciajelből az

$$r(n) = \sum_{i=0}^{J-1} a_{n-i} x(n)$$

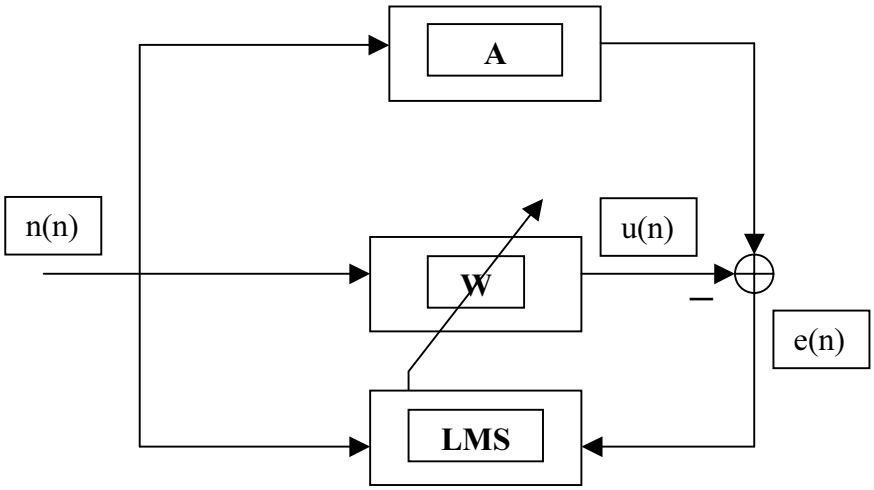
egyenlettel számítható.

Ezen *a* szűrőegyütthatók az XLMS végrehajtása előtt alkalmazott, az *A* rendszeren végzett identifikáció eredményeként állnak elő.



8.2. ábra. Az XLMS algoritmus blokkvázlata

Az identifikáció során olyan gerjesztőjelet kell alkalmazni, amely a rendszert minden olyan frekvenciapontban gerjeszti, ahol azt későbbiekben az XLMS algoritmussal használni kívánjuk, legelterjedtebb esetben ez lehet kellően egyenletes spektrumú zaj vagy multiszinuszos jel.



8.3. ábra. Rendszeridentifikáció LMS algoritmussal

Az identifikáció történhet például LMS algoritmus segítségével (8.3. ábra). Ekkor az LMS a *W* adaptív szűrővel modellezi az *A* rendszert, oly módon, hogy ütemenként úgy frissíti a szűrő paramétereit, hogy azok egyre pontosabban közelítik rendszer impulzusválaszának együtthatóit.

Az XLMS algoritmus előnye az LMS algoritmussal szemben, hogy a referenciajelet hozzákéslelteti, illetve fázisban is illeszti az akusztikai/mechanikus rendszer átvitelével módosult hibajelhez, így az algoritmus stabilitása biztosítható. Ezt használja fel a későbbiekben ismertetett aktív zajszűrő algoritmus is.

8.1.3 A többcsatornás XLMS algoritmus

A 8.4 ábrán egy többcsatornás XLMS algoritmus (Multiply Error LMS) hatásvázlata látható [14]. A rendszer tervezését nehezíti, hogy egyszerű paraméterkészlet helyett az átvitelt most mátrixok adják meg. Gyakorlati megvalósításban ez csatornánkénti szűrők alkalmazását jelenti. Amennyiben a rendszer K referenciajelet használ, és M beavatkozó jelet állít elő, a beavatkozó jelet az m-edik csatornán az

$$u_m(n) = \sum_{k=1}^{K} \sum_{i=0}^{N-1} w_{mki} x(n-i)$$

egyenlet adja meg. Azaz minden egyes kimenethez K, így a teljes rendszerhez K*M adaptív szűrő tartozik. Az átviteli rendszer modellezését minden csatornára külön-külön el kell végezni, azaz L hibajel esetén a referenciajelet kimenetenként L (összesen tehát M*L) számú modellszűrő szűri, az LMS k-adik bemenetén

$$r_{lmk}(n) = \sum_{j=0}^{N-1} c_{lmj} x_k(n-j)$$

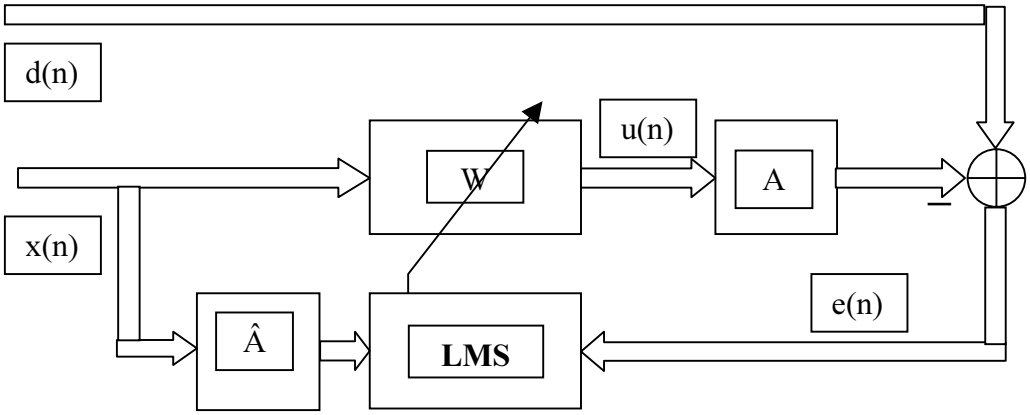
jelet kapja. Ezzel a rekurzív algoritmus alakja:

$$w_{mki}(n+1) = w_{mki}(n) + \alpha \sum_{l=1}^{L} e_l(n) \cdot r_{lmk}(n-i),$$

ahol α a konvergencia mértéke, *e(n)* hibajel pedig

$$e_l(n) = d_l + \sum_{m=1}^{M} \sum_{i=0}^{N-1} \sum_{k=1}^{K} r_{lmk}(n-i) w_{mki}$$

alakú.



8.4. ábra. Az XLMS algoritmus blokkvázlata

8.2 Számítási igények, lehetőségek

Digitális rendszerek alkalmazhatóságának, jóságának objektív mérőszáma a segítségükkel végrehajtható műveletek száma. Emellett elterjedt gyakorlat tipikus alkalmazások benchmark-ként való használata: ekkor az alkalmazás pontossága, amennyiben az algoritmus iteratív műveletet is tartalmaz, az iteratív ciklusok száma jelenti a rendszerrel szemben támasztott erőforrásigényt, a teljesítményt pedig meghatározza, hogy az eszköz ezt milyen mértékben tudja kielégíteni. Ily módon különböző típusú eszközök teljesítőképessége is összemérhetővé válik. A 6. fejezetben már szóltunk SHARC jelfeldolgozó processzor megszakításonkénti utasításainak számáról, ezért itt az ilyen műveletvégzési sebesség mellett megvalósított alkalmazásokkal mutatjuk be a rendszer lehetőségeit.

Az VII.táblázat tartalmazza az XLMS algoritmus különböző mintavételi frekvencia mellett elérhető legnagyobb komplexitását, amelyet meghatároz a felhasznált csatornaszám, és a rendelkezésre álló és a műveletek által igényelt utasítások száma. Az XLMS algoritmus J együtthatós modellszűrő, N együtthatós adaptív szűrőt és K csatorna esetén U=J*K²*1+K*N*3 utasítást igényel. Ebből visszaszámolható, hogy 8 kHz-es mintavételi frekvencia mellett a kártyán legfeljebb egy 100 együtthatós modellszűrőt és egy 30 együtthatós adaptív szűrőt tartalmazó 8 csatornás XLMS (ekkor egyetlen referenciajel van az összes csatornához, amelyet a másik soros porton lévő codec-en fogad a rendszer) valósítható meg.

	XLMS	
	8 kHz	64 kHz
1 csatorna esetén		
utasítások száma/csat.	5850	381
Modellszűrő fokszáma	1462	95
Adaptív szűrő fokszáma	1462	95
8 csatorna esetén		
utasítások száma/csat.	731	47
Modellszűrő fokszáma	66	4
Adaptív szűrő fokszáma	66	4

VII.táblázat. A megvalósítható XLMS algoritmusok legnagyobb komplexitása

A fenti adatokból látható, hogy 8 kHz-es mintavételi frekvencia mellett egy csatornán közel 1500 együtthatós szűrőket tartalmazó alkalmazás valósítható meg, amely elegendően magas érték, hogy segítségével precíz eredmény legyen elérhető. Változatlan mintavételezési frekvencia mellett nyolc csatorna használata esetén bár a végrehajtható utasítások száma a csatornák között eloszlik, még használható komplexitás áll elő. 64 kHz-es mintavételi frekvencián egy csatornás XLMS megvalósítása lehetséges, a modellszűrő fokszámát 150, az adaptív szűrő fokszámát 50 körüli értékre választva a felmerülő igényeknek sok esetben megfelelő algoritmus áll elő. A nyolc csatornás megvalósítás esetén viszont az adatbeolvasás mellett olyan nagy mértékben csökken a műveletvégzésre fordítható idő, hogy processzor 50 MIPS-es számítási kapacitása sem elegendő.

Mivel a codec-ek fő alkalmazási területe audio rendszerekben való alkalmazásuk, ezért magasabb mintavételi frekvenciás működés mellett használhatóak, fizikailag nem állítható be rajtuk a szabályozási folyamatokban, így a zajcsökkentés esetén is elegendő 1-2 kHz-es mintavételezés, decimáló algoritmus. Ezen segít, ha a magasabb frekvenciás működés mellett futtatott műveletvégző algoritmust decimálással kombináljuk. Ez, bár kis mértékű overhead-et jelent és át kell szervezni a program furását, jelentősen megnövelheti a végrehajtható utasítások számát, oly módon, hogy a decimálás mértékének megfelelő számú megszakítási rutint összefog, annak majdnem teljes ideje alatt a műveletvégző algoritmus futtatható.

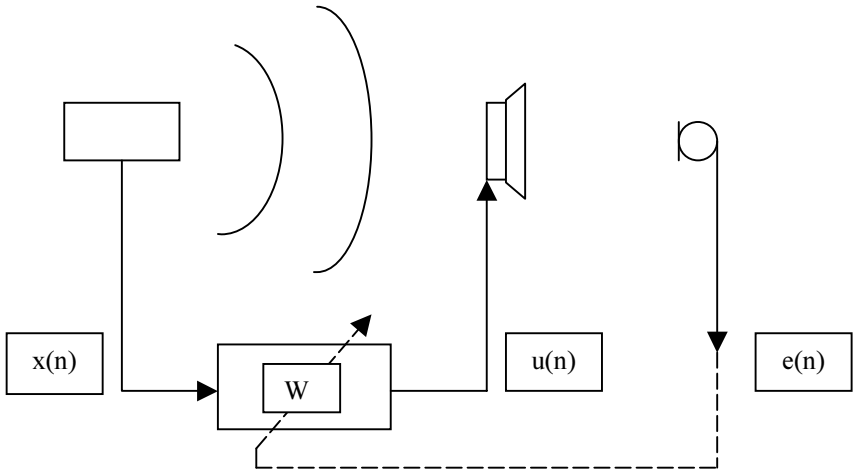
8.3 Az aktív zajszűrés

A környezetünkben működő gépekkel egyidős probléma a használatuk által keltett zajterhelés. Ezért a kezdetektől igény volt azok csökkentésére, ami a legutóbbi időkig néhány egyszerű és leginkább csak magasabb frekvenciákon hatásos megoldásban – mint például a berendezés megfelelően nagy fizikai távolságban való telepítése, hangszigetelő elemek, falak alkalmazása – merült ki. Ezen – a szakirodalom által passzív zajcsökkentésnek nevezett – megoldások közös hátránya a kis hatásfok, a nehézkes és drága telepítés, illetve az esetlegesen nagy méret.

Az 1950-es években indult, de csak az elmúlt 20 évben előtérbe került kutatási terület az ún. aktív zajcsökkentés. Több megoldás is ismert, amelyek a hanghullámok (közel lineárisnak tekinthető) terjedésekor bekövetkező interferencia jelenségek kiaknázásán alapulnak. A működés alapja, hogy ha egy pontban mérni tudjuk a környező hangkeltők (legyen az egy berendezés vagy hangsugárzó) által keltett jel

nagyságát egy ún. monitor mikrofonnal, akkor alkalmas eszközzel ellentétes fázisú, de azonos amplitúdójú jelet generálva elérhető, hogy a vizsgált pontban a kioltani kívánt zajforrásnak tekintett elsődleges forrás és a kioltó jelet szolgáltató másodlagos forrás jele éppen ellentétes fázisban találkozzanak, így eredő amplitúdójuk, ezáltal a hangérzet is jelentősen csökken. Ugyanezen megfontolások alapján előfordulhat, hogy egy másik pontban a két forrás jele éppen fázisban van, és kioltás helyett erősítik egymást, azaz a nemkívánatos zaj tovább nő. Ebből is látszik, hogy igen fontos a rendszer fizikai elrendezése. A legjobb megoldás olyan kialakításban adódik, ahol a monitor mikrofon és a másodlagos forrásként szolgáló hangsugárzó megfelelően közel helyezkedik el egymáshoz. A fizikai méretek meghatározzák az alkalmazhatóság körét is, mivel a mérőmikrofon és a beavatkozó egymáshoz közel helyezkednek el, illetve a megfelelő kioltáshoz szükséges fázishelyes összegzés miatt az eljárás csak alacsonyabb frekvenciás esetben alkalmazható, ezáltal megfelelő kiegészítő eljárásként alkalmazható a passzív módszerek mellett.

Egy tipikus alkalmazásban a rendszer felépítése a következő (8.5. ábra) [13]: Általános igény egy autó (repülő) utasterében a motor zajának legkisebb mértékre való csökkentése. Ekkor az elsődleges (azaz a zaj-) forrás az autó (repülő) motorja, a qiuet zone (magyarul csendes zóna, a szakirodalom nevezi így azt a térrészt, ahol a zajcsökkentés hatásos) középpontja a gépkocsivezető (pilóta), illetve valamely másik utas, ezért a hibajelet (ami nem más, mint az eredő zaj) mérő mikrofonokat az ülések környékére telepítik. Ebben a kialakításban a mikrofon és a motor közé kerülnek beépítésre a másodlagos forrásként szolgáló, a kioltó jelet előállító hangszórók. Természetesen a hangszóróból induló jelet a belső tér megváltoztatja: a jel csillapodik, emellett reflexiók lépnek fel a mérő mikrofonig (vagyis az utas füléig). Más szóval, ha a mikrofonnal megmérve ismert a motorból érkező jel nagysága a csendesíteni kívánt pontban, akkor nem annak ellentettjét kell a hangszóróval előállítani, hanem egy olyan jelet, amely a hangszóróból indulva terjedése során úgy változik meg, hogy a mikrofonnál a motor zajának ellentettjeként jelenik meg. Ezt a változást a jelúton a belső tér átvitele határozza meg, vagyis a zajcsökkentő eljárásnak feltétele a rendszer átviteli függvényének ismerete. Az átviteli függvényt megmérő (az akusztikai rendszer alkalmas modelljét szolgáltató) eljárást nevezi a szakirodalom identifikációnak. A zajcsökkentést minden esetben meg kell előznie az identifikációnak, amely segítségével előállnak egy olyan transzverzális szűrő együtthatói (az identifikáció során adaptív algoritmus számítja ki), amellyel a rendszer stabilitása érdekében a referencia jelet szűrni kell. A mérőmikrofon összegzi a két forrásból származó jelet: hibajelet állít elő. A rendszer működéséhez szükséges másik jel a referenciajel, amelynek alkalmas forrása például az autó (repülő) motorjának tengelyére szerelt fordulatszám-jeladó (ez az elrendezés előrecsatolt struktúrájú rendszernek tekinthető, így ennek okát ott bővebben tárgyaljuk). E két jelet pl. egy DSP kártyán futó adaptív algoritmussal feldolgozva a másodlagos forrásként alkalmazott hangszórók vezérelhetők oly módon, hogy a quiet zone-ban a motorból érkező zajt eliminálják.



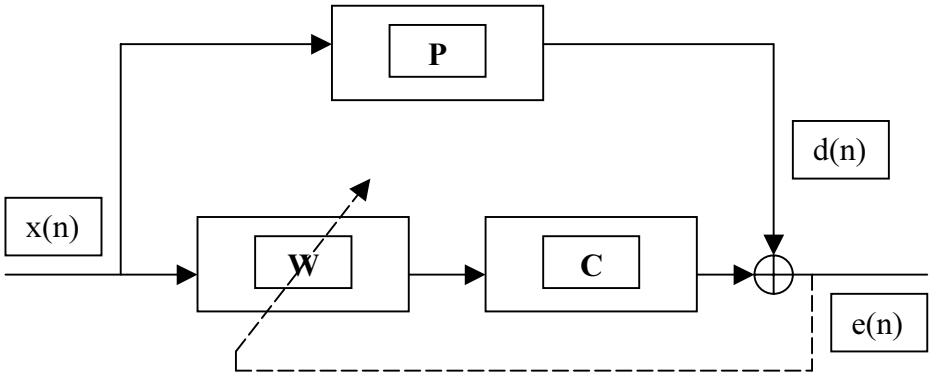
8.5. ábra. Zajszűrő alkalmazás

A tervezés szempontjából nem mindegy, hogy milyen célból kerül alkalmazásra a zajcsökkentő eljárás, mivel a rendszer többféle struktúrában építhető meg, ezen struktúrák különböző működésükből adódóan különböző típusú zavarjelek esetén lesznek hatásosak. Alapvetően kétféle jelet különböztetünk meg: periodikus jeleket, például ha a zajforrás egy forgó gép, illetve sztochasztikus (szélessávú) jeleket, ennek tekinthető egy helyiségbe beszűrődő forgalom zaja. Természetesen a két különböző típusú jel mindig együtt fordul elő, ám valamelyik domináns, így az annak kioltására tervezett rendszerrel már jelentős elnyomást lehet elérni.

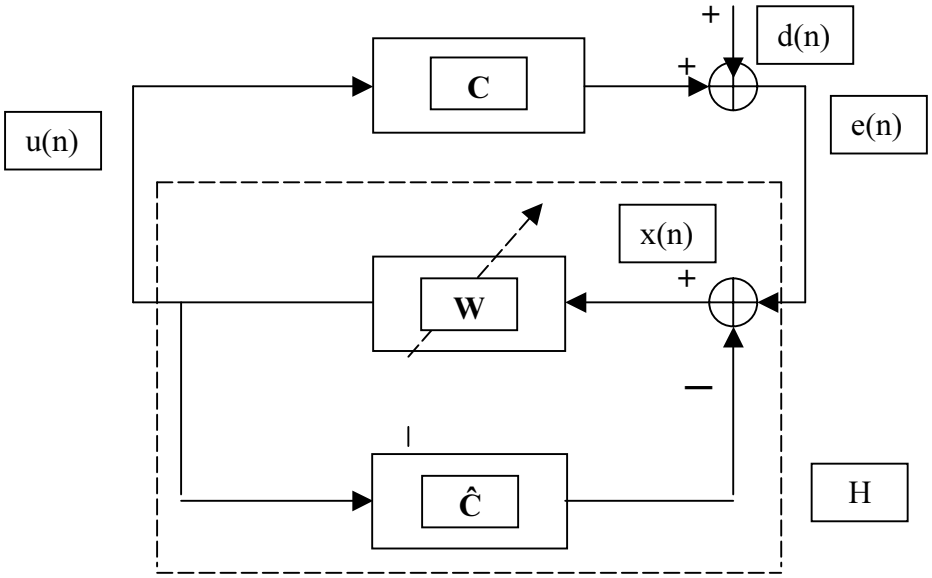
A periodikus jelek elnyomása az aktív zajcsökkentési feladatok közül az egyszerűbb. Ennek oka, hogy a periodikus jeleket determinisztikus jellegük miatt azokat véges számú pontban elég előírni, illetve azok korábbi mintáik alapján hibátlanul meghatározhatóak. Ezen tulajdonságok egy egyszerűbb és gyorsabb rendszer kialakítását teszik lehetővé. Sztochasztikus jelek, azok véletlen jellege miatt a teljes felhasznált frekvenciatartományban szükség van a kioltó jel előállítására.

Az aktív zajcsökkentő algoritmusokat kétféleképpen realizálhatjuk [14]: visszacsatolt (8.6. ábra) vagy előrecsatolt (8.7. ábra) struktúrában.

A visszacsatolt rendszerben a hibajelből áll elő a beavatkozó jel, amely a zavarjellel szuperponálva adja a hibajelet. A rendszernek tehát a hibajelből kell jósolnia a beavatkozó jelet, amely csak olyan jelek esetén alkalmazható hatásosan, amelyek mintái erősen korrelálnak.



8.6. ábra. Előrecsatolt struktúra



8.7. ábra. Visszacsatolt struktúra

Megemlítendő az előrecsatolt rendszer bemenőjelei közül az ún. referenciajel, amely szűrésével a hibajel előállítható. A referenciajelre a legfontosabb megkötés, hogy az elnyomandó zajjal korreláljon. Periodikus jeleknél ez teljesül, ha a zavarjel és a referenciajel frekvenciája megegyezik, azaz pl. forgó gép esetén a referenciajelet egy tachometer szolgáltatja, így a visszacsatolt rendszernél gyorsabb megoldáshoz jutottunk. Sztochasztikus jeleknél egy mikrofon a referenciajel-forrás, amely nem ideális átvitele miatt újabb nemlinearitások, zajok, és akusztikai visszacsatolás kerül a rendszerbe. Ez utóbbi jelentős stabilitási problémákat okozhat.

Ezen okok miatt a előrecsatolt struktúrát mind a periodikus, mind a sztochasztikus zajok csökkentésére használják, míg a visszacsatolt struktúra jobbára csak a periodikus jel csökkentésére alkalmas [14].

A fentiekből következik, hogy periodikus esetben elvileg mindig létrehozható tökéletes elnyomás, szélessávú jelek esetén erre csak akkor van mód, ha az elektronikus út (jelterjedés, jelfeldolgozás, jelelőállítás stb.) késleltetése kisebb, mint az akusztikus késleltetés, azaz a rendszer képes követni a kioltandó jel változásait.

8.4 Az implementálás

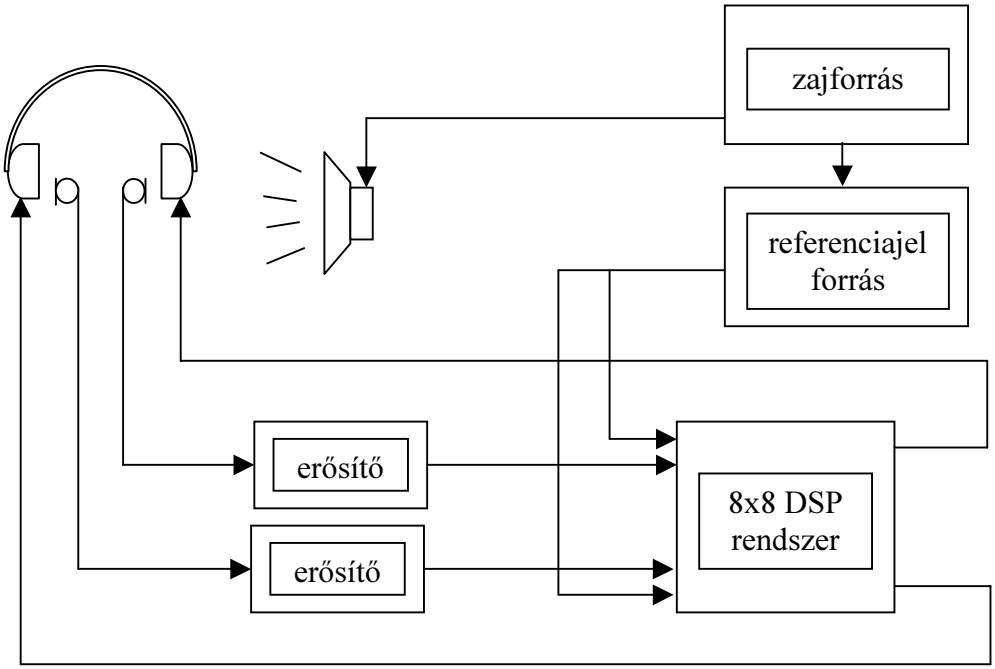
Periodikus zajok elnyomására tehát leggyakrabban adaptív előrecsatolt rendszereket alkalmaznak. Az adaptációt legelterjedtebben az LMS algoritmus család valamely típusával végzik. A megépített nyolccsatornás rendszeren XLMS algoritmus segítségével valósult meg.

8.4.1 A kísérleti összeállítás

Az 8.8. ábrán látható rendszerben egy csatorna jelei a következő úton terjednek: a fejhallgatóban a hangszórók mellett egy-egy mikrofon található, ezek érzékelik a külső hangszóróból származó kioltandó zaj és a fejhallgató hangszórója segítségével előállított beavatkozó jel eredőjét, ezt a jelet mikrofon–előerősítőn és erősítőn való

kondicionálás után mint hibajelet kapja meg a rendszer az egyik bemenetén, a másik bemeneten a referenciajel kerül beolvasásra. Ez a zajcsökkentő alkalmazásokban mikrofonnal történik, de most erre a célra – az egyszerűbb megvalósíthatóság érdekében – közvetlenül a zajt keltő rendszert meghajtó zavarjelet alkalmazunk. A processzor számításainak eredményeképpen előállt beavatkozó jel – a rendszer egyetlen kimenőjele – a fejhallgató hangszóróját hajtja meg. A jelutak mindkét csatornára (jobb, illetve bal csatorna) természetesen azonosak. A fejhallgató mellett a környezeti zajokat szimuláló hangszóró lett elhelyezve. Az összeállítás zajelnyomás esetén, illetve identifikáció elvégzésekor azonos. A különbség csak annyi, hogy identifikáció esetén a referenciajel forrása zajgenerátor, zajcsökkentés működése alatt az elnyomandó jellel korrelált jel, illetve az identifikáció során a zajforrás ki van iktatva a rendszerből.

A felhasznált műszerek listája a függelék 21. oldalán található.



8.8. ábra. A zajcsökkentő rendszer összeállítása

8.4.2 A megvalósított algoritmusok

Az XLMS algoritmust felhasználó zajelnyomáshoz elengedhetetlen az akusztikai rendszer identifikációja. Ezért a zajelnyomó alkalmazás megvalósítása az LMS és az XLMS algoritmus implementálását jelenti. A két algoritmust praktikus-kényelmi okok miatt egy program tartalmazza, amelyek között az I/O kártyán található DIP kapcsolók segítségével lehet váltani.

- **Az identifikáló algoritmus**

Mint azt korábban említettük, az identifikáció alatt az akusztikai rendszer impulzusválaszának megmérése történik és annak eredménye kerül a későbbiekben felhasználásra az akusztikai rendszer modelljében, ezért kívánatos minél pontosabb

értékek elérése. Az identifikációt LMS algoritmus végzi, amely egy transzverzális FIR szűrőt megvalósító és annak adaptálását végző rutint tartalmaz, melynek főbb lépései:

1. Aktuális érték beolvasása, cirkuláris memóriába való tárolása, kimenetre való kiírása
2. N együtthatós (adaptív) FIR szűrő futtatása, együtthatók mentése cirkuláris memóriába
3. Zavarjel mintájának beolvasása
4. Hibajel számítása
5. Adaptáció: a korrekció mértékének kiszámítása, az adaptív szűrő együtthatóinak ezzel az értékkel való frissítése

Az adaptív szűrő fokszámát meghatározza a közeg átvitelének bonyolultsága. Ebben az esetben a fejhallgató és a mikrofon közötti táv igen kicsi, így az átvitel egyszerű, a rendszer 8 kHz-es mintavételi frekvenciája mellett 200 pontra korlátozva az impulzusválaszt is elegendő információ áll elő.

Ezen lépéssor jelenti az identifikáció egy ciklusát. A ciklusok számát, azaz a futási időt a szükséges pontosság határozza meg. A nagyobb pontosság kisebb konvergencia együttható alkalmazásával lehetséges, de ez az identifikációs idő növekedéséhez vezet, ezalatt végig biztosítani kell az akusztikai rendszer változatlanságát.

Az identifikáció végét tehát a megfelelően kis hiba elérése jelzi. Ennek a „megfelelően kis érték”-nek a nagysága a felhasználó belátására van bízva (bizonyos ésszerű megszorításokkal), az identifikáció hibajelét kivezetve (melyre a kártya nagyszámú kimenete lehetőséget nyújt), annak csökkenése figyelemmel követhető.

- **Az XLMS algoritmus**

A tényleges zajcsökkentést ez az algoritmus végzi, működése közben felhasználja az identifikáció alatt használt adaptív szűrő utolsó értékeit modell szűrő együtthatókészleteként. Az algoritmus lépései:

1. Aktuális érték beolvasása, cirkuláris memóriába való tárolása, kimenetre való kiírása
2. N együtthatós (adaptív) FIR szűrő futtatása, együtthatók mentése cirkuláris memóriába
3. J együtthatós (modell) FIR szűrő futtatása, együtthatók mentése cirkuláris memóriába
4. Hibajel mintájának beolvasása
5. Adaptáció: a korrekció mértékének kiszámítása, az adaptív szűrő együtthatóinak ezzel az értékkel való frissítése

Az adaptív szűrő itt is a rendszer pontosságáért felel, míg a modellszűrő a referenciajel fázisát igazítja a hibajelhez. Ezért nem szükséges, hogy L nagy legyen – 8 kHz-es mintavételi frekvenciát alapul véve – az alkalmazott periodikus zavarjel esetén 100 körüli értéknél is elfogadható mértékű maradó hiba érhető el.

- **Kényelmi funkciók**

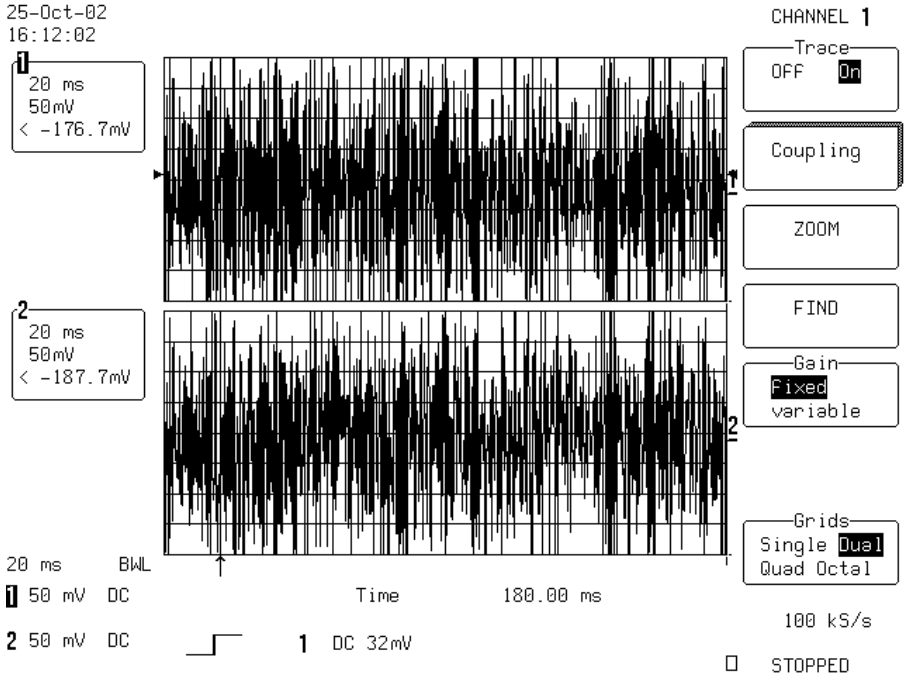
A megvalósított rendszerrel cél volt a kártya alkalmazhatóságának bizonyítása, így néhány olyan funkcióval is rendelkezik, amely a későbbi felhasználást megkönnyíti:

- az identifikáció, illetve az XLMS algoritmus kapcsolókkal indítható, leállítható
- a konvergencia együttható nagyságát állítani lehet
- az adaptív szűrő állapotváltozóit törölni lehet

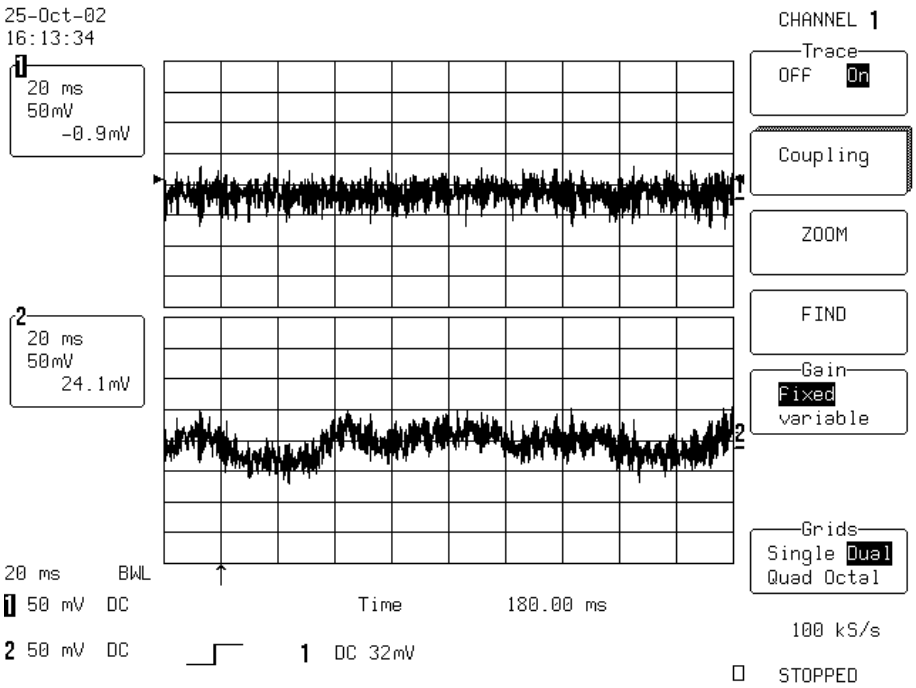
Ezen funkciók az I/O kártyán található DIP kapcsolósorral vezérelhetők, így lehetőség nyílik arra, hogy a rendszer felprogramozás után PC nélkül autonóm módon működjön.

8.4.3 Eredmények

A 8.8. ábrán látható összeállítás identifikációja egy 8 kHz mintavételi frekvencián 200 együtthatós FIR szűrővel történt, a konvergencia együttható értékét 0.00005-re választottuk. Bemenőjelként 1,5 kHz sávszélességű 200 mV-os effektív értékű zajt használtunk. Az identifikáció addig tartott, míg a kezdetben 200 mV nagyságú hibajel (8.9. ábra) 25 mV körüli értékre állt be (8.10. ábra). Ez a folyamat viszonylag gyorsan lezajlott, és a hibaérték a további felhasználásra már elegendően kicsinek bizonyult.

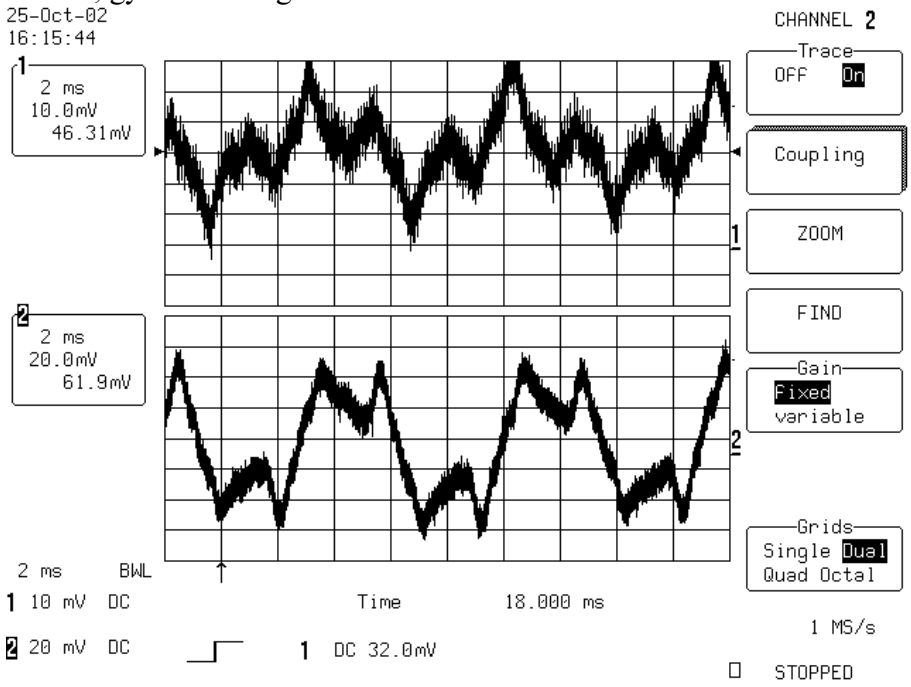


8.9. ábra. Az identifikáció kezdeti hibajele

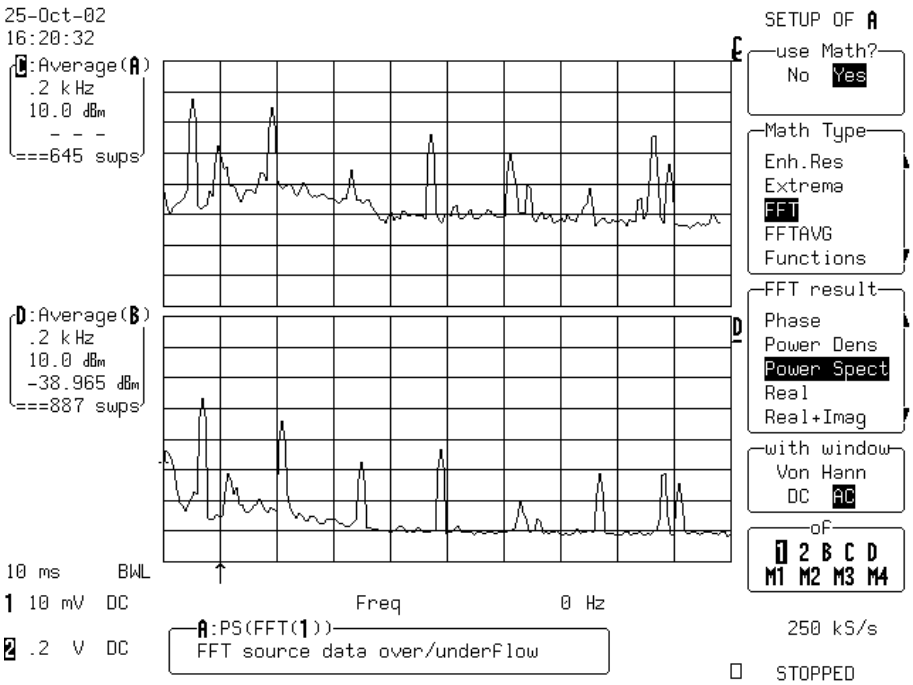


8.10. ábra. Az identifikáció hibajele lecsökkent

Az XLMS algoritmus adaptív szűrője 100, modellszűrője 200 együtthatóval rendelkezett. A konvergencia együttható 0.0005 volt, a mintavételi frekvencia nem változott. Referencia- ill. zavarjelnek 140 Hz-es négyszögjelet választottunk, így magasabb harmonikusok elnyomása is elemezhetővé vált. Ilyen paraméterek mellett stabil működés, gyors konvergencia lett elérhető.

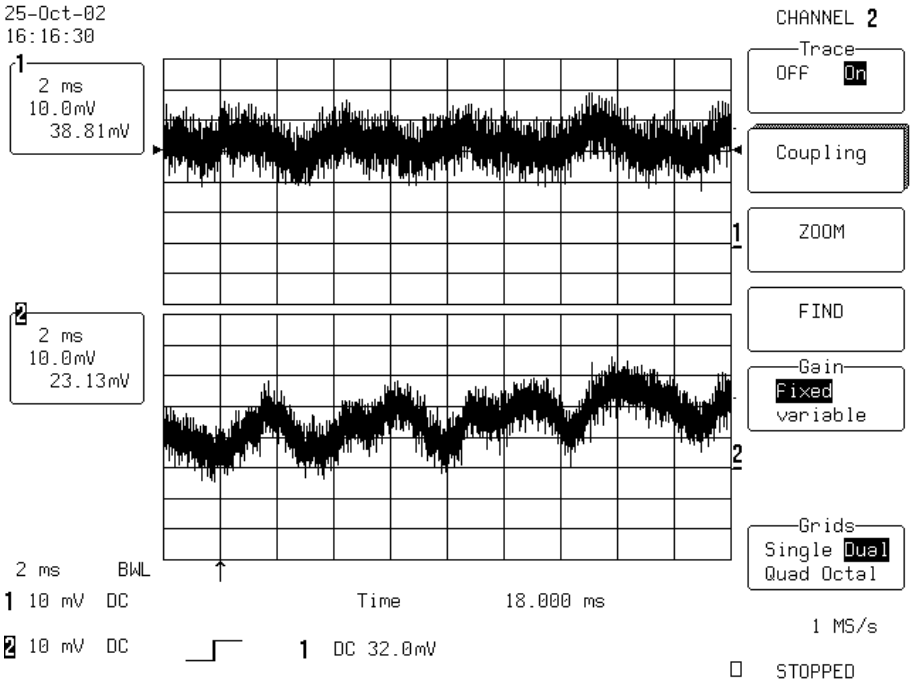


8.11. ábra. A rendszer hibajele XLMS működése nélkül

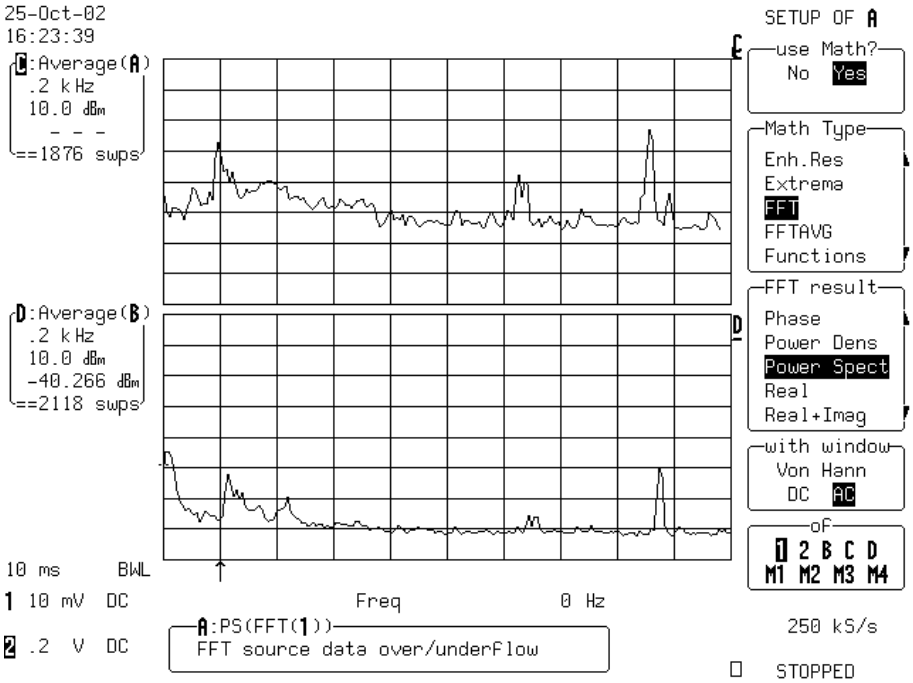


8.12. ábra. A hibajel teljesítményspektruma XLMS működése nélkül

Az alábbi két ábrán a hibajelek láthatóak az XLMS algoritmust alkalmazva. Az elérhető elnyomás az alapharmonikusra 30 dB volt a bal csatornán (felső görbe), és 40 dB a jobb csatornán (alsó görbe). Az ábrákon az is látszik, hogy magasabb harmonikusokra is hatásos volt az algoritmus (bár magasabb harmonikusokra beállás a lassabb, mint az alapharmonikus esetén), illetve, hogy a két csatorna átvite és zaja különbözött.



8.13. ábra. A rendszer hibajele XLMS működése mellett



8.14. ábra. A hibajel teljesítményspektruma XLMS működése mellett

8.5 Következtetések

A kezdeti eredmények biztatóak: a rendszerrel – a jelenlévő nemlineáris hatásokat figyelembe véve – megfelelő mértékű elnyomás volt elérhető. Amennyiben egy tökéletes zajelnyomó rendszer elkészítése lett volna a cél, akkor a bemutatott értékek nem lennének elegendően jók, hiszen egy ilyen alkalmazással szemben magasabb elvárásokat támasztunk Ez a rendszerben lévő nemlinearitások alaposabb vizsgálatát, a működésben fellelhető optimalizálási lehetőségek megvalósítását, az elnyomást és a sebességet növelő megoldások alkalmazását, az esetlegesen jelenlévő rejtett hibák javítását teszi szükségessé. A későbbi igényesebb megvalósítás esetére a rendszer rendelkezik a jobb eredményekhez szükséges nagyobb számítási kapacitással. A dolgozat célja ezzel szemben csupán a megépített rendszer alkalmazhatóságának, illetve működőképességének bizonyítása, az eredeti rendszert kiegészítő többletszolgáltatások bemutatása volt egy teljes értékű alkalmazás fejlesztési menetén keresztül. Ennek az elvárásnak maradéktalanul eleget is tett, mivel az elkészített komplett alkalmazás stabilan, a funkcióját megfelelően ellátva futott. Egy algoritmus jelfeldolgozó processzorokon való megvalósításának néhány problémáját bemutattuk, illetve azok megoldását megvilágítottuk. A gyengébb eredmények nem a rendszer hiányosságaiból adódtak, azok alaposabb megvalósítás esetén valószínűleg javíthatók.

9. fejezet

Összefoglalás, kitekintés

Dolgozatunkban egy nyolccsatornás jelfeldolgozó rendszer megtervezését és megvalósítását mutattuk be. A Méréstechnika és Információs Rendszerek Tanszéken folytatott munkánk során az előzetesen magunk elé tűzött célokat elértük.

Elkészült a saját tervezésű codec kártya, amelynek illesztése az EZ-KIT kártyához megtörtént és sikeres volt. A codec kártya néhány kisebb javítás után a vártnak megfelelően működött. A rendszert működtető szoftver réteg szintén elkészült, és valóban képes a rendszer működtetésére. A specifikációban előirányzott funkciókat a rendszer maradéktalanul teljesíti. Az előzetes terveknek megfelelően a rendszeren implementáltunk egy aktív zajszűrő algoritmust, amely tipikus példája azon alkalmazásoknak, amelyek érdekében a rendszer megszületett. Ez az implementált alkalmazás azon felül, hogy bizonyítja mind a hardver, mind a szoftver részegységek helyes működését, bemutatja a teljes rendszer együttes működését egy valódi jelfeldolgozási feladat ellátása közben. Az alkalmazás gyakorlati implementálása egyben a rendszer első átfogó tesztje is volt, amely a rendszer egyes részegységeinek bemérése után a teljes rendszer funkcionalitását is vizsgálta. A megvalósított alkalmazásnak ezen felül benchmark funkciója is van, azaz a megvalósítás különböző mérőszámai tájékoztatást nyújtanak a rendszer teljesítőképességéről.

A rendszer megvalósult és használható, de tervezzük még további fejlesztését. A közeli jövőben szándékozunk megvalósítani egy függvénykönyvtárat, amely arra szolgál, hogy a rendszer még inkább támogassa, még kényelmesebbé tegye a különböző alkalmazások beágyazhatóságát. Olyan függvényeket kívánunk megvalósítani, amelyek a majdan implementálásra kerülő alkalmazásokban várhatóan gyakran felmerülnek.

Az általunk tervezett nyolccsatornás rendszer segítségével megvalósíthatóvá, a gyakorlatban kipróbálhatóvá váltak a több csatorna egyidejű használatát igénylő jelfeldolgozási algoritmusok. Reményeink szerint a jövőben ez a rendszer is hasznos eszköze lesz a jelfeldolgozás területén végzett oktatási és kutatási munkának.

Irodalomjegyzék

[1] Analog Devices Inc., http://www.analog.com

[2] Innovative DSP, http://www.innovative-dsp.com

[3] Bittware, http://www.bittware.com

[4] Pentek Inc., http://www.pentek.com

[5] Domain Technologies, http://www.domaintec.com

[6] Causal Systems, http://www.causal.com

[7] *ADSP-2106x SHARC EZ-KIT Lite™ Reference Manual* , Analog Devices Inc., 1997

[8] *ADSP-2106x SHARC™ User's Manual*, Second Edition, Analog Devices Inc., 1997

[9] Evaluation Board for Low Cost, Low Power, CMOS General Purpose Dual Analog Front End, *EVAL-AD73322EB Datasheet*, Analog Devices Inc., 1998

[10] Low Cost, Low Power CMOS General-Purpose Dual Analog Front End, *AD73322 Datasheet*, Analog Devices Inc., 2000

[11] S.D. Snyder, „*Microprocessors for active control: bigger is not always enough*", Proceedings of Active 99 – The 1999 International Symposium on Active Control of Sound and Vibration, Fort Lauderdale, Florida, USA, 1999., pp. 45-62.

[12] B. Widrow, S.D. Stearns, „*Adaptive Signal Processing*" Prentice-Hall, Inc., Engelwood Cliffs, 1985

[13] Kollár I., Németh J., Sujbert L. „*Digitális jelfeldolgozás (Hallgatói segédlet)*" Budapesti Műszaki és Gazdaságtudományi Egyetem, Méréstechnika és Információsrendszerek Tanszék, 2001

[14] S. J. Elliott, P. A. Nelson „*Active noise control*" IEEE Signal Processing Magazine, Vol. 10, 1993

Függelék