Diplomaterv

Nyilatkozat

Alulírott *Bogár István*, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

_____

Bogár István

Tartalomjegyzék

Kivonat

A dolgozat nyolc bemeneti és nyolc kimeneti csatornával rendelkező jelfeldolgozó rendszer tervezésével és megvalósításával foglalkozik. A rendszer magját az Analog Devices által gyártott ADSP-21065L lebegőpontos jelfeldolgozó processzor alkotja, amely a cég SHARC típuscsaládjába tartozik. A rendszer alacsony ára mellett sokrétű szolgáltatást kínál. A kártya alkalmas felhasználói alkalmazás (online, offline jelfeldolgozó program) futtatására, valamint rendelkezik adatgyűjtő üzemmóddal is.

A rendszer rendelkezik személyi számítógépes kezelői felülettel, amelynek segítségével a rendszeren futó felhasználói alkalmazás vagy az adatgyűjtő üzemmód bizonyos határok között konfigurálható.

A dolgozatban bemutatom a rendszer hardveres és szoftveres komponenseinek tervezését és kivitelezését.

Az elkészült jelfeldolgozó rendszer hardveres adottságai mellett nagy előnye, hogy nagymértékben támogatja az alkalmazások gyors és biztonságos implementálását. Ennek megfelelően ismertetésre kerül a monitorra és alkalmazásra bontott szoftver struktúra, amely a fenti előnyöket biztosítja a felhasználó számára.

A dolgozat első részében általánosságban bemutatom a jelfeldolgozó rendszerek alkalmazási területeit valamint felépítésüket. Ezek után a jelenlegi fejlesztés hátterét és a rendszerrel szemben felmerült igényeket, valamint a specifikációt ismertetem.

Bemutatom a hardver és szoftver tervezés menetét a specifikációtól kiindulva a rendszerterveken át a megvalósult kártyáig, illetve monitor programig. Részletesen foglalkozom a nyomtatott áramkör tervezése során figyelembevett szempontok bemutatásával. Ismertetem az alapvető jelfeldolgozási struktúrákat, amelyek megvalósíthatók a rendszerben.

Ezután a megvalósított sokcsatornás adatgyűjtő bemutatása következik. Ez az alkalmazás alapvetően egy lehetséges felhasználása a rendszernek, de mégis kitüntetett szerepe van. A tervezés során mindvégig szem előtt tartottam az alkalmazás támasztotta erőforrásigényeket, és ennek megfelelően alakítottam ki a kártyát.

A dolgozat utolsó része tulajdonképpen használati utasításként szolgál egy általános alkalmazás sikeres implementálásához. Itt kerülnek bemutatásra azok a szoftveres elemek, amelyek segítségével az alkalmazás számára elérhetővé válik a hardver egységek kezelése.

Abstract

This master's thesis deals with the design and realization of a digital signal processing system which has eight inputs and outputs. The core of the system is the ADSP-21065L floating-point digital signal processor produced by Analog Devices. The mentioned device is a member of the SHARC family. Besides its low price the system provides many services. The device is suitable for running applications (on-line, off-line signal processing programmes), and it disposes of data acquisition mode.

The system has a user interface on personal computer, by which the user application or the data acquisition mode run on the system can be easily configured.

In this thesis I present the design and realization of the hardware and softvare components of the system.

The advantage of the complete signal processing system is its hardware capability and it supports considerably quick and safe implementation of the applications. According to this the structure of the software devided into monitor and application is presented.

In the first part of the thesis the application fields and the structure of the signal processing systems is introduced. After this the background of the present development and the requirements to be met are shown. The specification of the system is given as well.

I introduce the process of the design of hardware and software from the specification via system plans to the realized printed circuit board and monitor program. I deal with the points of view taken into account while planning the printed circuit board. I present the basic signal processing structures which can be implemented on the system.

After this the presentation of the implemented multi channel data acquisition system follows. This application is a possible use of the system, but it has a special role. During the design I always cared about the resources required by this application, and the hardware was worked out according to these requirements.

The last part of the thesis serves as a user manual for the succesfull implementation of a general application. At this part I present the software units, which provide the handling and accessing the hardware units.

1. fejezet

Bevezető

A digitális jelfeldolgozási algoritmusok alkalmazása napjainkban már rendkivül elterjedt dolognak számít. A villamosmérnöki szakterületek majdnem minden részén igaz, hogy a digitális rendszerek alkalmazása sokkal előnyösebb, mint az analóg rendszereké. A hatékony digitális jelfeldolgozási algoritmusok mindig valamilyen központi egységet igényelnek, mert számításokat kell végezni a külvilág jeleit reprezentáló számértékeken. A központi egység lehet általános célú processzor vagy logikai áramkörös (FPGA) megvalósitású. Azonban ilyen feladatra leggyakrabban jelfeldolgozó processzort (DSP) használunk. Ezeket a processzorokat jelfeldolgozási algoritmusok hatékony implementálására fejlesztették ki.

A központi feldolgozó egység számára a digitális jeleket a környező fizikai világ jeleiből állitjuk elő különböző érzékelők és AD átalakítók segítségével. Ahhoz pedig, hogy a központi egység által előállított digitális jelek a fizikai világban is megjelenjenek, DA átalakítóra és esetlegesen további átalakitókra (pl. elektromechanikus) van szükség.

A jelfeldolgozó rendszerek közül sok csak egy bemenettel és egy kimenettel rendelkezik (SISO rendszerek), de gyakran van szükség sok csatorna jeleinek egyidejű, valós idejű feldolgozására (MIMO rendszerk). Néhány terület, ahol többcsatornás jelfeldolgozó rendzsereket használnak:

- Audiotechnika
- Aktív zajcsökkentés
- Telekommunikáció
- Orvostechnika
- Diagnosztika

- Identifikáció

A dolgozat egy ilyen sokcsatornás jelfeldolgozó rendszer megtervezéséről és megvalósításáról szól. A rendszer segítségével az összes fent említett jelfeldolgozó alkalmazás implementálható a gyakorlatban. A cél az, hogy olyan hardver és szoftver környezet jöjjön létre, amely szilárd alapot nyújt az alkalmazás számára olyan módon, hogy az alkalmazásfejlesztő elméleti szakembernek ne kelljen foglalkoznia a hardver megvalósítás részleteivel.

2. fejezet

A fejlesztés háttere

2.1. Általános megfontolások

A digitális jelfeldolgozás alapvető ismérve az, hogy számítási műveleteket végzünk a fizikai világ jeleit reprezentáló számértékeken. A különböző jelfeldolgozási algoritmusok futtatásához tehát mindig valamilyen számítási művelet végrehajtására alkalmas központi feldolgozó egység szükséges.

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

A jelfeldolgozási algoritmusok megvalósítására több megoldás közül választhatunk. A legegyszerűbb megoldás, hogy az algoritmusokat PC-n futó valamilyen ismertebb programnyelvben vagy környezetben (pl. Matlab, C, C++) fejlesztjük ki, és futtatjuk. Ez rendkívül felhasználóbarát megoldás, hiszen az implemetálás egy jól megszokott könnyen debug-olható kötrnyezetben történik, és a hardver megvalósítás részleteivel sem kell törődni. Rendkívül nagy hátránya ennek a megoldásnak az, hogy a környezet fizikai jeleihez való real time hozzáférés nehézkes vagy egyáltalán nem oldható meg. Ezért inkább offline feldolgozás esetén jöhet szóba ez a megoldás. További problémát jelent, hogy az általános célú processzorok műveletvégző egységei csak hagyományos számítási műveleteket végrehajtására képesek, így a tipikus jelfeldolgozási algoritmusok során felmerülő számítási műveletek megvalósítása esetleg körülményes lesz.

A másik véglet a "hardverközeli" megoldás. Ebben az esetben a számítási műveleteket végző egységet egy FPGA valósítja meg. Ez úgy történik, hogy a gyakran végrehajtásra kerülő műveleteket logikai hálózatok értékelik ki. Ezen

hálózatok kifejezetten egy specifikus művelet végrehajtására képesek, viszont különböző párhuzamosítások alkalmazásával a hálózat bonyolódása mellett a végrehajtási sebesség növelhető. Ez a megoldás rendkívül jó sebességkritikus valósidejű alkalamzások esetén, ahol maga a feldolgozás nem túl bonyolult. Hátránya viszont a rugalmatlanság, és a tervezés nehézsége.

Az általam választott megoldás ötvözi a fenti lehetőségeket, magába foglalva mindkettő előnyös tulajdonságait. A DSP (Digital Signal Processor) egy olyan közepes bonyolultságú processzor, amelynek belső felépítését a jelfeldolgozás tipikus műveleteinek elvégzésére optimalizálták. Egy olyan céleszköz, amely nem túl bonyolult és olcsó, de a jelfeldolgozási algoritmusokat egy komplex mai processzornál gyorsabban valósítja meg.

A DSP programjának fejlesztése PC-n futó fejlesztői környezetben történik. A programozás nyelve általában processzor specifikus assembly utasításokból áll. Manapság azonban a bonyolultabb problémák megoldásánál már inkább C nyelven írt programokat alkalmaznak, ahol a kritikus lényegi részeket assembler betétek alkotják.

A jelfeldolgozási algoritmusok gyors végrehajtása annak köszönhető, hogy a DSP hardver szinten támogatja az algoritmusok által használt műveletek elvégzését. Ezek olyan specifikus egységek, amelyek funkcióit egy általános célú processzorban csak bonyolultan lehet megvalósítani. Ezek a funkciók többek közt a konvolúció alapját képző Multiply and Accumlate, a cikuláris buffer megvalósítása és a bit-reserved üzemmód. A gyorsaság másik oka az, hogy a leggyakoribb műveletek párhuzamos elvégzésére is van lehetőség az FPGA megvalósítás mintájára. A párhuzamosítást segíti elő a Harvard-architektúra is, amely szétválasztja a kód-, illetve az adatmemóriát.

A DSP alapú jelfeldolgozó rendszer tehát egy optimális középutat jelent a két előzőleg ismertetett megoldás között, megtartva mindkettő jó tulajdonságait.

A digitális jelfeldolgozó rendszerek a környező fizikai világgal különböző bemeneti és kimeneti jelúton keresztül vannak kapcsolatban. A továbbiakban egy bemeneti-kimeneti jelútpárt csatornának nevezünk.

Többcsatornás rendszerekre akkor van szükség, ha több bemeneti jelet párhuzamosan és valós időben kívánunk feldolgozni, illetve ha több kimenetet kell kezelnünk. Minél bonyolultabb egy alkalmazás, annál valószínűbb, hogy több csatornára van szükség. Tipikusan az aktív zajcsökkentő rendszerek annál hatékonyabbak, minél több mikrofon jelét képesek feldolgozni, és minél több ponton képesek beavatkozni. Többcsatornás rendszerre van szükség különböző hibrid szabályozási rendszerek esetén is.

Léteznek olyan alkalmazások is, amelyek jellegükből adódóan többcsatornásak, például az audio keverők. A digitális megoldás azért kedvező ebben az esetben, mert különböző effektek alkalmazhatók a bemeneti jeleken. 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 alkalmazások implementálására használjuk. Ez azt is jelenti, hogy mód van a már megvalósított alkalmazások továbbfejlesztésére is a hardver megváltoztatása nélkül.

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

2.2. A fejlesztés alapja

A fejlesztést TDK dolgozat keretében kezdtük el [10]. Ennek során létrehoztunk egy analóg és digitális perifférákkal rendelkező kiegészítő kártyát a laborban már rendelkezésre álló ADSP-21061-es processzort tartalmazó EZ-KIT Lite fejlesztői kártyához.

2.3. Irodalomkutatás

A piacon természetesen sok cég kínál olyan eszközöket, amelyek alkalmasak a fenti többcsatornás alkalmazások implementálására. A fent említett TDK dolgozat keretében széleskörű piackutatást végeztünk, ahol felmértük a lehetőségeket. Elsődleges forrásunk az internet volt, ahol DSP alapú eszközöket gyártó cégek honlapjait tanulmányoztuk. A kutatás eredménye az alábbiakban látható:

Analog Devices

Az Analog Devices igen sok fejlesztői kártyát gyárt (ADI terminológiával Evaluation Board) [1]. Ezek a kártyák általában egy adott ADI termék köré vannak építve, ennek megfelelően a DSP alapú kártyákon kívül csak egy codec-et találunk (Codec: kóder-dekóder, egy integrált áramkörben megvalósított AD és DA átalakító). Vannak továbbá olyan AD illetve DA kártyák, amelyek vagy sok bemenettel, vagy sok kimenettel rendelkeznek. Olyan kártya azonban nincs, amelyen elegendő számú kétirányú csatorna van.

Innovative DSP

Ez az amerikai cég jelfeldolgozó megoldásokat kínál ipari szereplők számára [5]. A terméklistán két többcsatornás jelfeldolgozó kártyát találtunk, mind-

kettő a nagy számítási kapacitással rendelkező Texas Instruments DSP-jén alapszik.

A Toro nyolc bemenetű, nyolc kimenetű kártya 16 bites szukcesszív approximációs AD, illetve DA átalakítókkal rendelkezik. A delfin fantéázia nevű termék ami egy viszonylag új kártya, 32 bemenettel és 6 kimenettel rendelkezik, szigma-delta AD és DA átalakítói 24 bitesek, a maximális minavételi frekvenciájuk pedig 192 kHz.

Mindkét kártya tökéletes lett volna számunkra, de rendkívül magas áruk miatt mégis kizártuk őket.

Bittware

Szintén amerikai cég, amely DSP-n alapuló beágyazott rendszerekkel foglalkozik [2]. Elsősorban az Analog Devices SHARC DSP családjának tagjaira fejlesztik termékeiket. A mi szempontunkból az Audio PMC+ nevű kártya lett volna érdekes ,ami egy nyolccsatornás 96 kHz mintavételi frekvenciával dolgozó eszköz. A kártyát kész alkalmazói szoftverrel együtt kínálják, mely lényegében az adatok kódolását és dekódolását végzi, tehát ezt a kártyát arra a célra szánják, hogy más, esetleg még nagyobb számítási kapacitással rendelkező egységek számára készítse elő az adatokat. Magas ára miatt nem volt optimális.

Pentek Inc.

Ez a cég is sokféle DSP alapú terméket kínál, de kissé eltérően a fenti cégek megoldásaitól [6]. Ugyanis ez a gyártó külön árusít DSp kártyákat és IO (input-output) kártyákat, melyek tetszőlegesen összekapcsolhatóak VME busz segítségével. A megfelelő többcsatornás rendszer megépíthető lenne egy 4265-ös és egy 4284-es típusszámú kártyából. Így egy Texas Instruments által gyártott DSP-t tartalmazó nyolc bemenetű, nyolc kimenetű rendszert kapnánk. Az AD és DA átalakítók felbontása 16 bit, a maximális mintavételi frekvencia 50 kHz. A rendszer ára nagyon magas, ami nem áll arányban a teljesítményével.

Domain Technologies

Ez a cég a többiekhez képest szerényebb termékválasztékot kínál. Viszonylag olcsó, négycsatornás DSP alapú kártyákat készítenek [4]. Az alacsony árnak köszönhetően kevés kiegészítő funkciót valósítottak meg. Az eszköz csak USB porton keresztül tud kommunikálni, és bővitése sem megoldott, így a különböző alkalmazói programok telepítése nehézkesnek tűnik.

Casual Systems

Az ausztrál illetőségű cég kizárólag aktív zajcsökkentést megvalósító hardver és szoftver egységeket kínál [3]. Az EZ-ANC II nevű sokcsatornás kártya és a hozzá tartozó szoftver eszközök jelentik a teljes skálát. Nagy hátránya, hogy az egyes csatornákat nem lehet teljesen szabadon definiálni, és az eszközt a zajcsökkentő algoritmusokon kivűl egyéb algoritmusok implementálására nem lehet használni.

Összefoglalás

A lista természetesen nem teljes, hiszen még számos további gyártó foglalkozik ilyen termékek előállításával. A lényeges következtetések azonban levonhatók már ezekből az információkból is. Látható, hogy az elérhető kártyák nem rendelkeznek elegendő csatornával. A kellően sok csatornával rendelkező rendszerek viszont nagyon drágának bizonyultak.

A viszgálatból az derült ki, hogy volt létjogosultsága egy saját rendszer fejlesztésének. A TDK dolgozat keretében a laborban található DSP kártya felhasználásával létrehoztuk a nyolccsatornás rendszert, aminek a költsége töredéke volt a piacon kaphatóknak.

2.4. Sokcsatornás adatgyűjtő

Bizonyos esetekben szükség van a vizsgált rendszer jeleinek szimultán figyelésére illetve rögzitésére. Az így létrejött adatok utólagos elemzése által rendkívül sok információt megtudhatunk az adott rendszerről. Az adatok lehetséges felhasználásai:

- Identifikáció
- Telemetria
- Dokumentálás

Identifikáció esetén a felvett adatok utólagos feldolgozásával meghatározuk a rendszer struktúráját, illetve a fontosabb paramétereit. Ilyenkor általában a rendszerre valamilyen speciális gerjesztő jelet kapcsolunk (pl. zaj, színusz, több színusz, speciális hullámforma), és figyeljük az általa létrehozott válaszjeleket. Az identifikációhoz rögzítenünk kell mind a rendszer gerjesztő-, mind a válaszjeleit.

Telemetria esetén az általunk létrehozott rendszer működését vizsgáljuk. Ekkor a rendszer állapotát és működésének helyességét reprezentáló jelek

kerülnek rögzítésre. Vizsgálhatjuk a rendszer hosszútávú stabilitását, extrém körülmények közti viselkedését.

Bizonyos esetekben pedig a jeleket dokumentálás céljából kell rögziteni.

A laborban ugyan rendelkezésünkre áll egy Fostex gyártmányú D-108 tipusú audio recorder, de használata bizonyos esetekben nem a legoptimálisabb. Fix 44.1 kHz-es mintavételi frekvenciája bizonyos alacsony frekvenciájú jelek esetében túl magas, és a DC csatolás hiánya is okozhat problémát, tehát konkrét igény mutatkozott egy az előbbi hiányosságokat pótló sokcsatornás adatgyűjtőre.

A korábbi sokcsatornás rendzser létrehozása során megszerzett tapasztalatok és az adatgyűjtő hardver igényei megfelelő kiinduló pontként szolgáltak a dolgozatban ismertetésre kerülő sokcsatornás rendzser kifejlesztéséhez. A cél az, hogy a létrejövő sokcsatornás rendzserben megfelelően implementálni lehessen az adatgyűjtőt, tehát a korábbi rendszer képességeit bővíteni kellett. Szükség lesz tehát egy nagy kapacitású adattároló egységre, amibe a mintavételezett adatok kerülnek, valamint fel kell ruházni a rendszert kommunikációs lehetőségekkel is, hogy a regisztrátumokat továbbítani tudja archiválás és feldolgozás céljából.

3. fejezet

Specifikáció és rendszerterv

3.1. Specifikáció

3.1.1. A rendszer áttekintése

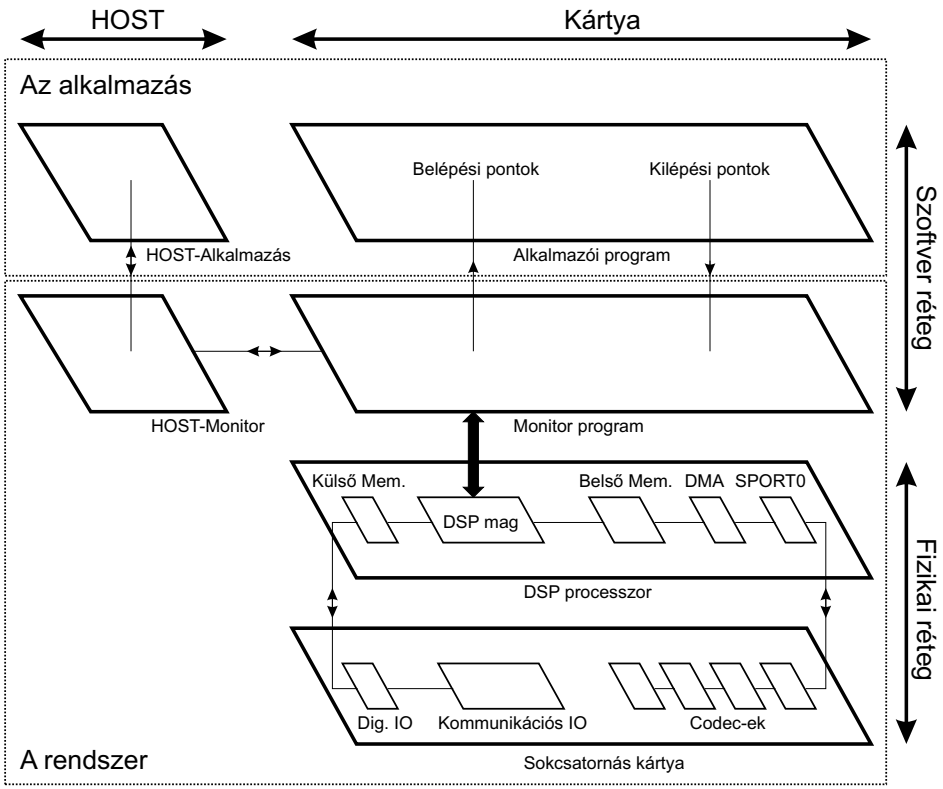
Az általam tervezett sokcsatornás rendszer három fő része maga a kártya, amely tartalmazza a jelfeldolgozó processzort és a perifériákat, a processzort működtető szoftver, és a HOST-ton futó szoftver, mely kapcsolatot tart a processzoron futó programmal. A rendzser áttekintése az alábbi 3.1 ábrán látható.

A sokcsatornás rendszer egy nyomtatott áramköri lapon került megvalósításra, így kezelése mechanikailag sokkal egyszerűbb, valamint nem kell foglalkozni az egyes egységek közti csatlakozókkal, amelyek potenciális hibaforrások lehetnek.

A kártyán található FLASH memória alapvetően arra szolgál, hogy a DSP-n futó szoftver részeit tárolja. Mivel a memória egyes részei tetszőlegesen felhasználhatók az adott alkalmazáshoz tartozó program és adatok tárolására, így lehetőség nyílik arra, hogy a rendszer a HOST számítógéptől függetlenül működjön. Ilyen módon a rendszer mobilizálható, alkalmazása egyedülálló (stand-alone) rendszerként is lehetséges.

3.1.2. A sokcsatornás kártya

A tervezés első fázisában alkalmas AD/DA átalakítót kellett választani, amely a feldolgozni kívánt jelek és a DSP közötti illesztést valósítja meg. A feladat igen egyszerű volt, mert a jelen esetben is a korábbi TDK dolgozatban alkalmazott code-eket használtam. Alkalmazásukkal kapcsolatosan szerzett tapasztalatok nagy segítségemre voltak a tervezésnél.



3.1. ábra. A rendszer áttekintése

A codec elnevezés a kóder-dekóder szókapcsolatból származik. Ezek olyan eszközök, amelyek egy tokba integrálva tartalmaznak AD 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 a független AD és DA átalakítóknak.

Előnyös tulajdonsága a codec-eknek a kevés vonalat igénylő, egyszerűen kezelhető soros illesztő port-juk, amivel a DSP-hez kapcsolódnak. Léteznek olyan codec-ek is, amelyek kaszkádosíthatók, tehát ebben az esetben több codec együttes működésére is van lehetőség. Ebben a rendszerben éppen ilyen codec-ekre van szükség, mert a többcsatornás működést csak több codec egyidejű használatával érhetjük el.

A codec-ek további előnye, hogy több funkcionális egységet valósítanak meg egy tokon belül, így alkalmazásuk esetén az általuk felhasznált nyomtatott áramkör tervezése is egyszerűsödik, továbbá az áramkör disszipációja is csökken.

A codec kártya alapvetően méréstechnikai, aktív zajszűrési és szabályozási

feladatok ellátása céljából jött létre. Ezért olyan codec-et kell alkalmazni, amelynek a tulajdonságai ilyen jellegű alkalmazások implementálása esetén kedvezők.

Emiatt a rendszer csatornaszáma fontosabb jelemző volt, mint a codecek 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 alkalmazásnál 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 vezérel rendszerben.

További igény volt az is, hogy a kártya DC jeleket is képes legyen feldolgozni. Ugyanis az alkalmazások egy része igényli azt.

A codec választása során az egyik igen fontos kérdés volt az átalakító működési elve. Alapvetően szukcesszív approximációs, illetve szigma-delta elven működő codec-ek jönnek szóba. Az audiotechnikában általában a túlmintavételezéssel működő szigma-delta átalakítókat használják, mert nagy a linearitásuk, és nem igényelnek átlapolásgátló szűrőt. A hátrányuk az, hogy késleltetésük (group delay) nagyobb, mint a szukcesszív approximációs átalakítóknak. Ez utóbbi éppen kisebb késleltetése miatt kedvezőbb lenne szabályozástechnikai célokra, de összességében a szigma-delta elven működő codec mellett szóló érvek az erősebbek.

A másik kérdés a kártya felépítését nagyban befolyásoló eszköz, az alkalmazott nagykapacitású tároló eszköz volt. Erre az eszközre mindenképpen szükség van a rendszerben, mert a létrejött adatokat el kell tárolni. Járható út lett volna a tárolandó adatok nagysebességű kapcsolaton való azonnali továbbítása. Ez a lehetőség azonban kizárta volna az offline feldolgozás és a stand-alone működés megvalósíthatóságát. Kezdetben sok lehetőség merült fel. Szóba jött többek között winchester, flash memória, SDRAM memória.

A winchester alkalmazása rendkívül bonyolultá tette volna a rendszert, mert mindenképpen ki kellett volna alakítani valamiféle Windows kompatibilis fájlrendszert. A flash memória pedig drágának és lassúnak bizonyult, ezért az SDRAM alkalmazása mellett döntöttem. Egyszerű kezelése, rendkívül alacsony ára, nagy sebessége igen előnyös volt a rendszer szempontjából.

A későbbiekben specifikált kapacitás mindenképpen több memória használatát tette szükségessé, emiatt SDRAM modult épitettem be a rendszerbe. A modul integrálva tartalmaz 8-16 darab memóriát, és párhuzamositásukat is megoldották a modul bonyolult nyomtatott áramköre segítségével, így a rendelkezésre álló csatlakozón keresztül egyszerűen elérhető minden tok.

Azonban a piacon kapható memória modulok olyan IC-ket tartalmaznak, amelyeket a DSP SDRAM vezérlő perifériája már nem tud kezelni. Ezért

a rendszerben meg kellett valósítani egy SDRAM kezelő egységet. Ez egy FPGA használatát tette szükségessé. Alkalmazása viszont további előnyökkel jár.

Az FPGA segítségével egyszerű volt megvalósítani az általános célú digitális IO-t. Ez egy egyszerű felület, mely felhasználható a rendszerhez későbbiekben csatlakozó perifériák (pl. relék, opto bemenetek) kezelésére. Ezenkívül a hozzá kapcsolódó LED-ek és DIP kapcsolók primitív kezelői felületként is használhatók.

Továbbá az FPGA segítségével a későbbiekben rendkívül egyszerűen meg lehet valósítani gyors kommunikációs egységek (Ethenet, USB) illetve más processzorok integrálását a rendszerbe.

3.1.3. A szoftver réteg

Mivel a rendszer vezérlő eleme, a jelfeldolgozó processzor egy programozható eszköz, a hardver egységeken kívül a szofver eszközöket is definiálni kellett.

Elsődlegesen arra van szükség, hogy a DSP-n futó assambly nyelven megírt program a hardver egységeket megfelelően működtesse, valamint hogy az egységek közötti (a perifériák és a DSP, HOST számítógép és a kártya) kommunikációt vezérelje. Ez a program inicializálja a kártyán található perifériákat, és előállitja a kommunikációs csatornákat.

A sokcsatornás rendszer szoftver eszközei képezik a rendszer szoftver rétegét. Ezt az elnevezést az indokolja, hogy a sokcsatornás rendszerhez nem csak egy működtető program jön létre a fejlesztés során, hanem egy olyan szoftver keret, amely túlmutat egy konkrét DSP program szerepén.

A szofver réteg egy nagyon fontos funkciója az, hogy a jelfeldolgozó alkalmazásokat beágyazza a rendszerbe. Ez konkrétan azt jelenti, hogy a monitor program az alkalmazói programok számára egy keretként jelenik meg. Tehát a rendszerhez tartozó szoftver önmagában még nem valósít meg jelfeldolgozó algoritmusokat, de szilárd alapot biztosít ilyen alkalamzások implementálásához. Ily módon a monitor program 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 monitor elvégzi helyette. Az alkalmazás rendszerbe ágyazása pedig oly módon történik, hogy a monitor bonyolítja le a DSP - codec kommunikációt, és az alkalmazói program számára előkészíti az adatokat. Az alkalmazói programnak csak a dedikált memória területeket kell írnia és olvasnia, valamint ezek után meg kell hívnia az adott perifériához tartozó kezelő függvényt.

Szintén a monitor réteg bonyolítja le a kártya - HOST számítógép közötti kommunikációt is. A kommunikáció segítségével mind a monitor, mind az alkalmazás működése befolyásolható. A HOST-on futó monitor program segítségével tesztelhető a kártyán található perifériák működése, illetve módosítható a FLASH memória tartalma. Ez utóbbi alatt új alkalmazás és a hozzátartozó adatok letöltését értjük.

Tehát új alkalmazás fejlesztése esetén a DSP programjában található megfelelő belépési pontokban implementálni kell jelfeldolgozó szoftver algoritmusát. Ugyanezt a módszert követve a HOST-on futó program megfelelő pontjai is kitöltendők, létrehozva az adott alkalmazáshoz tartozó kezelői felületet.

A rendszer szoftver rétegének ilyen módon való megvalósitása biztosítja azt, hogy a rendszer széles körben felhasználható legyen.

3.1.4. Sokcsatornás adatgyűjtő

Az adatgyűjtő tulajdonképpen a sokcsatornás rendszer egy lehetséges alkalmazása. Az átlagos alkalmazástól eltérően kitüntetett szerepe van, mert már a tervezés kezdetétől figyelembe kellett venni az általa támasztott igényeket.

Az alkalmazás sikeres implementálása bizonyítja mind a rendszer működőképességét, mind a használhatóságát.

A megvalósított adatgyűjtő maximum 8 bemeneti jelet képes szimultán rögzíteni maximum 4,5 percig 62.5 kHz mintavételi frekvencián. A regisztráció maximális ideje kevesebb csatorna vagy alacsonyabb mintavételi frekvencia esetén lineárisan nő. Természetesen az egyes regisztrátumokat mint önálló egységeket kell kezelni, így minden egyes hozzájuk kapcsolódó információt is rögzíteni kell.

A hibás felvételek elkerülésére szolgál a visszajátszás funkció, amely segítségével a regisztrátumot a rögzítésre kijelölt bemenetekhez tartózó kimeneteken meg lehet jeleníteni. A kimenetekre hangszórót vagy oszcilloszkópot kapcsolva ellenőrizhető a regisztrátum. Ha az utolsó regisztrátum hibásnak bizonyul, akkor törölhető.

Mindig csak a legutolsó regisztrátumot lehet törölni, ami egyszerűsíti a programot, mert nem vezet töredezett memóriához. A töredezett memória akkor jön létre, ha több regisztrátum közül nem a legutolsót töröljük, tehát képződik egy nem használt terület, aminek a kezdete és a vége is kötött. Szemétgyűjtő algoritmus, illetve egy allokációs tábla megvalósításával elkerülhető ez a jelenség, de mindkét megoldás növelné a feladat komplexitását.

A regisztrátumok HOST számítógépre való feltöltése után további feldolgozásra és archiválásra nyílik lehetőség. Az, hogy az adatmozgatás jelen esetben kétirányú, tehát az elöbbi eset ellentettje is megtehető, további le-

hetőségeket nyújt. Így tulajdonképpen a rendszert egy 8 csatornás hangkártyaként lehet felhasználni. A repetitív lejátszási mód pedig alkalamassá teszi a rendszert, hogy egy felhasználó által definiált sokcsatornás gerjesztőként funkcionáljon.

3.1.5. Hardver specifikáció

Jelfeldolgozó processzor

Analog Devices ADSP-21065L lebegőpontos DSP
60 MIPS (max. 180 MFLOPS)

Analóg bemenetek

8 analóg bemenet, 4db AD73322-e codec segítségével kialakítva
max. 0.7 V p-p [0dB bemeneti erősités esetén]
16 bites felbontás
szigma-delta AD átalakítókkal mintavételezve
programozható bemeneti erősítés 0/38dB
magas bemeneti impedancia
választható AC, DC csatolás
RCA csatlakozás

Analóg kimenetek

8 analóg bemenet, 4db AD73322-e codec segítségével kialakítva
max. 0.7 V p-p [0dB kimeneti erősités esetén]
16 bites felbontás
szigma-delta DA átalakítókkal mintavételezve
programozható kimeneti erősítés +6/-15dB
RCA csatlakozás

Mintavételi frekvenciák

64, 32, 16, 8 kHz [16.384 MHz-es oszcillátor esetén]
62.464, 31.232, 15.616, 7.808 kHz [16 MHz-es oszcillátor esetén]
Egyéb mintavételi frekvenciák oszcillátor csatlakoztatás esetén elérhetők
Ezeknél kisebb mintavételi frekvenciák szoftveres úton alkalmazhatók

Csoport késleltetés

25 μ usec AD átalakításkor
50 μ sec DA átalakításkor

Memória

256 Mbyte SDRAM

HOST kommunikáció

RS-232, RS-485 univerzális soros porton keresztül
Későbbiekben USB, Ethernet csatlakoztathatósága

Digitális IO

16 bit kimenet és 16 bit bemenet (3.3 V CMOS kompatibilis)

Teljesítmény igény

Dupla tápfeszültség, ± 6 V (± 10%), 700 mA

Méret

265 mm x 118 mm x 50 mm

3.1.6. Szoftver specifikáció

Csatornák száma

8 analóg bemenet
8 analóg kimenet
16 bit általános digitális bemenet (memóriába ágyazottan elérhető)
16 bit általános digitális kimenet (memóriába ágyazottan elérhető)
FPGA szabad lábainak felhasználása esetén komplexebb IO funkciók is elérhetők

Megvalósítható FIR szűrők együtthatóinak száma (csatornánként)

f_{MV}	64 kHz	32 kHz	16 kHz	8 kHz
N_{TAP}	1030 (128)	2062 (257)	4125 (515)	8250 (1030)

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

Szoftveresen állítható mintavételi frekvencia (az oszcillátor osztásával)
Oszcillátorok csatlakoztatása a rendszerbe

Memóri kezelés

8 darab 16 bites szó mozgatása egyszerre (írás, olvasás)
Automatikus memória frissítés

Kommunikáció

A későbbiekben definiált soros protokol által

Kezelői felület

8 általánosan használható DIP kapcsoló
8 általánosan használható LED
A HOST-on megvalósított felület, mely a soros porton keresztül vezérli a DSP-t

Alkalmazás fejlesztés

DSP program fejlesztése VisualDSP környezetben, assembly vagy C nyelven
HOST program fejlesztése Visual Basic, vagy Visual C segítségével

Adatok mentése

A HOST-on futó kezelői felület és a DSP monitor programja közötti adatmozgató üzenetekkel

Stand-alone működés

A rendszerben található FLASH memória felhasználásával lehetséges

3.1.7. Sokcsatornás adatgyűjtő specifikáció

Felvételkor kiválasztható bemenetek

Eldönthető, hogy az adott csatorna bemeneti jele felvételre kerül-e vagy sem

Mintavételi frekvencia

A codec-ek által nyújtott 4 féle mintavételi frekvencia
8 fokozatú felező decimáló egység segítségével további 8 kHz-nél alacsonyabb mintavételi frekvenciák egészen 30 Hz-ig

Bemeneti kompenzáló szűrő

A bemeneti csatornák átvitelének kompenzálásaára szolgál (audio felvételek esetén)
Opcionálisan ki-bekapcsolható

A felvételekkel kapcsolatos műveletek

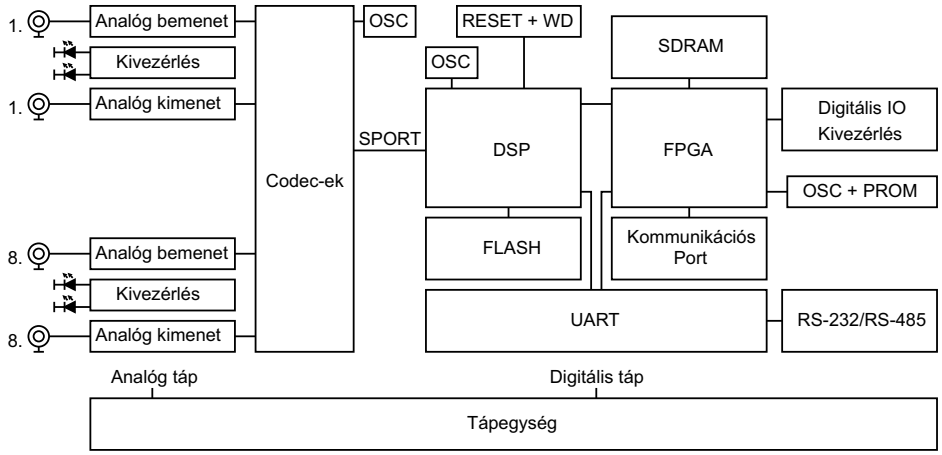
A felvételekhez tartozó beállítások kezelése
Bármelyik felvétel viszajátszható (egyszeri, ismétlődő)
A memóriában található legfrissebb felvétel törölhető
A felvételek feltöltése a HOST-ra Matlab vektor formátumban
Matlab vektor letöltése az adatgyűjtőre

3.2. Rendszerterv

A fejezet első részében specifikált rendszert alkalmasan megválasztott határvonalak segítségével négy nagyobb egységre lehet bontani. Ezeknek az egységeknek külön rendszertervet kellett készíteni.

3.2.1. Hardver rendszerterv

A hardver egységek egymáshoz való kapcsolódását a 3.2. ábra szemlélteti.



3.2. ábra. Hardver rendszerterv blokkvázlata

Megfigyelhető, hogy a blokkvázlat alapvetően öt nagyobb egységre tagolódik:

- A DSP és környezete.
- A codec-ek és a hozzájuk kapcsolódó analóg illesztő modulok
- Az FPGA és az általa lekezelt periféria egységek
- A soros vezérlő és az RS-232 illetve az RS-485 szintillesztő modul
- Tápegység

A kártya magját alkotó jelfeldolgozó processzor környezetében található egy oszcillátor, amely az órajelet szolgáltatja. A reset és a watchdog timer áramkör pedig a processzoron futó program biztonságos és stabil működését segíti elő. A reset funkció a bekapcsolás, illetve egy bizonyos határt meghaladó tápfeszültség ingadozás esetén a processzor regisztereit és perifériáit alaphelyzetbe állítja, ami tulajdonképpen a program megfelelő feltételek melletti újraindulását eredményezi. A watchdog funkció ezzel szemben a programot futás közben hivatott felügyelni. Bizonyos meghatározott időközönként a futó programnak impulzus segitségével az értékét törölni kell (watchdog timer reset), mert ezek elmaradása esetén reset jelet generál. A program legsérülékenyebb, lehetőleg periódikusan futó részeibe (interruptok)

kell elhelyezni a watchdog timert törlő rutint. A processzort érő külső hatások (elektromágneses impulzusok, rosszul vagy egyáltalán nem kezelt események) a futó program fagyását eredményezhetik, ami a periodikus törlés kimaradásával és a DSP és közvetve az egész kártya újraindulásával jár. Rosszul megírt programot a watchdog timer nem javítja meg, de bizonyos hibák esetén a rendszer távolról (a kommunikációs periférián és a monitoron programon keresztül) átkonfigurálható marad, azonban egy helyesen megírt program zajos környezetben való működésének stabilitását növeli.

A processzorhoz szervesen kapcsolódó FLASH memória tárolja a monitor és az alkalmazói programot, valamint az alkalmazói programhoz tartozó felhasználó álltal definiált további adatokat (adatbázisok, szűrőegyüthatók, futás során biztonságos formában eltárolt adatok). A reset bekövetkezése után ebből a memóriából tölti be a DSP a monitor és az alkalmazói programot. A monitor futása közben pedig a HOST-tól kapott parancs alapján módosíthatja a memória tartalmát.

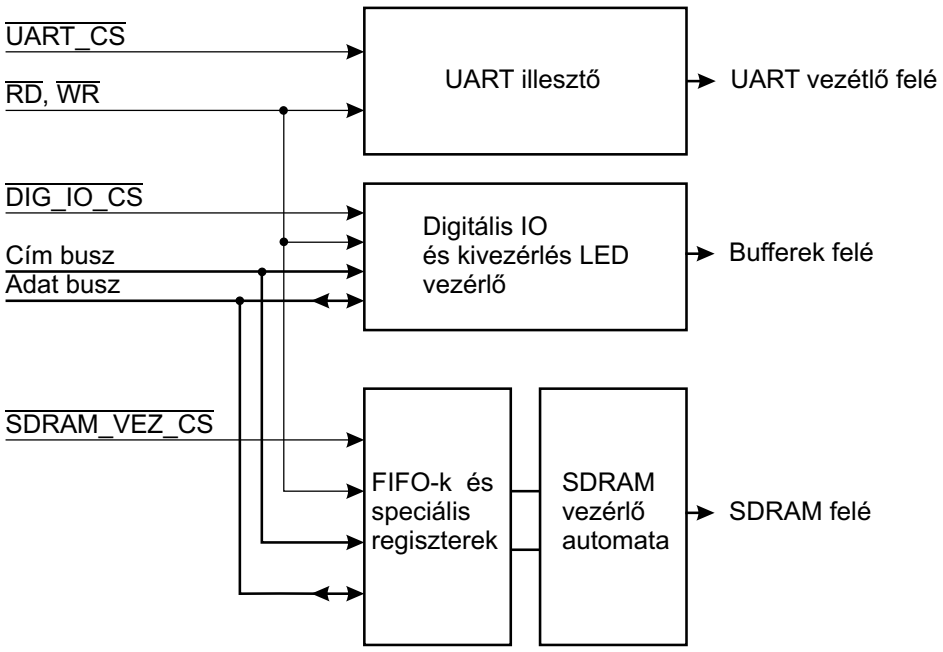
A DSP és az analóg bemeneti és kimeneti jelek kapcsolatáról az analóg illesztőmodulok és az azokhoz tartozó codec-ek gondoskodnak. Az analóg illesztőmodulok feladata a bemeneti és kimeneti AC illetve DC csatolás megvalósítása, a bemeneti jel nagy impedanciával történő fogadása és a kimeneti jel kis impedanciás szolgáltatása, továbbá a codec-ek szimmetrikus ki-bemeneteinek illesztése a kártya aszimmetrikus ki-bemeneteihez. A codec-ek kommunikációs interfészei láncszerű kapcsolatot alkotva a DSP soros portjával kommunikálnak. Ez a topológia nagymértékben leegyszerűsiti a tervezést, mivel a soros kommunikáció a codec-ek felbontásától függetlenül csak kevés számú vezetéket igényel. Ez jelentősen kevesebb, mint a párhuzamos elérés esetén (16 bites felbontású átalakítók alkalmazása esetén a két elérés által igényelt vezetékek számának aránya megközelítheti a négyet is), továbbá a DSP-nek is csak egy soros portját veszi igénybe. A codec-ek láncba kapcsolásán azt értjük, hogy a DSP soros portjának kimenete az első codec bemenetéhez kapcsolódik, a bemenete pedig az utolsó codec kimenetéhez. A láncban pedig a codec-ek adják tovább az adatokat egymásnak mindaddig, amíg el nem ér a címzetthez. A lánc első helyén álló codec a master, a többi egység a DSP-vel együtt pedig slave. A master kezdeményezi az adatcsere folyamatának megindulását, és ezzel egyetemben a mintavételezés időzítését is elvégzi. Az adatcsere folyamat (a DSP megkapja az előző mintavételi időponthoz tartozó bemeneti mintákat, és átadja a következő mintavételi időpont kimeneti mintáit) befejezésekor megtörténik a bemeneti csatornák mintavételezése, és a kimeneti csatornák értékeinek frissítése. Minden bemeneti csatornához tartozik továbbá két darab kivezérlésjelző LED is. A sárga LED már a bemenet -24 bB-t meghaladó szintje esetén világít, míg a piros LED csak a vágás közvetlen közelében kapcsolódik be. Mindkét LED

vezérlése tulajdonképpen egy szoftveresen megvalósított, újra triggerelhető monostabil multivibrátor. Ha a jel akárcsak egy minta idejéig is a megfelelő tartományban tartózkodik, a neki megfelelő multivibrátor kimenete aktiválódik. Erre a funkcióra a könnyebb észlelhetőség miatt van szükség.

A következő nagyobb egység az FPGA és környezete. Ez az eszköz plusz költséget jelent, de beépítését elsősorban az indokolja, illetve teszi szükségessé, hogy a DSP-ben található SDRAM vezérlő már nem képes megfelelően kezelni a felhasználni kívánt SDRAM modult. Beépítése azonban további lehetőségeket is kínál:

- Egyszerűsíti a nyomtatott áramkört.
- További perifériák megvalósításának lehetősége.
- Utólagos módosíthatóság és újradefiniálhatóság.
- A tervezés során elkövetett hibalehetőségek számának csökkentése.

Az FPGA-ban definiálandó hardver egységek rendszerterve a 3.3. ábrán található.



3.3. ábra. Az FPGA-n definiált hardver blokkvázlata

Az ábrán jól látható, hogy az FPGA közvetítő szerepet játszik a DSP és az SDRAM között, továbbá elvégzi a dinamikus memória frissítését is. A

közvetítő szerep miatt ugyan nő a memória kezelésének ideje (a DSP parancsot ad a memóriavezérlőnek írásra vagy olvasásra, majd a vezérlő elvégzi a műveletet, és értesíti róla a processzort), de mivel azt háttértárként, és nem operatív tárként használjuk, így ez nem fog problémát okozni. Az FPGA emellett diszkrét logikai elemek segítségével megvalósítja a digitális IO-t és a kivezérlésjelző LED-ek vezérlését is. Az elemeket egy buszon keresztül éri el, ami további egyszerűsödést okoz a nyomtatott áramkör tervezésekor, és a felhasználható lábakból is kevesebbet igényel. Az előbbieken kívül a kommunikációban is fontos szerepet játszik. Az UART vezérlő számára megfelelő módon átütemezi a DSP buszvezérlő jeleit (CS, RD, WR), valamint a későbbiekben beépítésre kerülő nagysebességű kommunikációs egységek (Ethernet, USB) illesztésében is szerephez jut.

A kártya első lépésként csak soros porton lesz képes kommunikálni, amit egy UART vezérlő valósít meg. A DSP párhuzamos buszára illesztett egységet a processzor interupt segítségével kezeli. A vezérlő ki-bemeneti jeleit választható módon egy RS-232 vagy egy RS-485 modul alakítja át a megfelelő formára. A szintillesztők két külön kártyán kerülnek megvalósításra, és a kiválasztott kommunikációnak megfelelő kártyát csatlakoztatjuk majd a DSP-t tartalmazó kártyákhoz, amit a rendszer automatikusan felismer. Az RS-485-ös kapcsolat egy izolált illesztő áramkörrel és egy, az izolált oldalt tápláló DC-DC konverter segítségével realizálódik. Ezáltal a kártya védett lesz a RS-485-ös vonalról érkező káros hatásokkal szemben is.

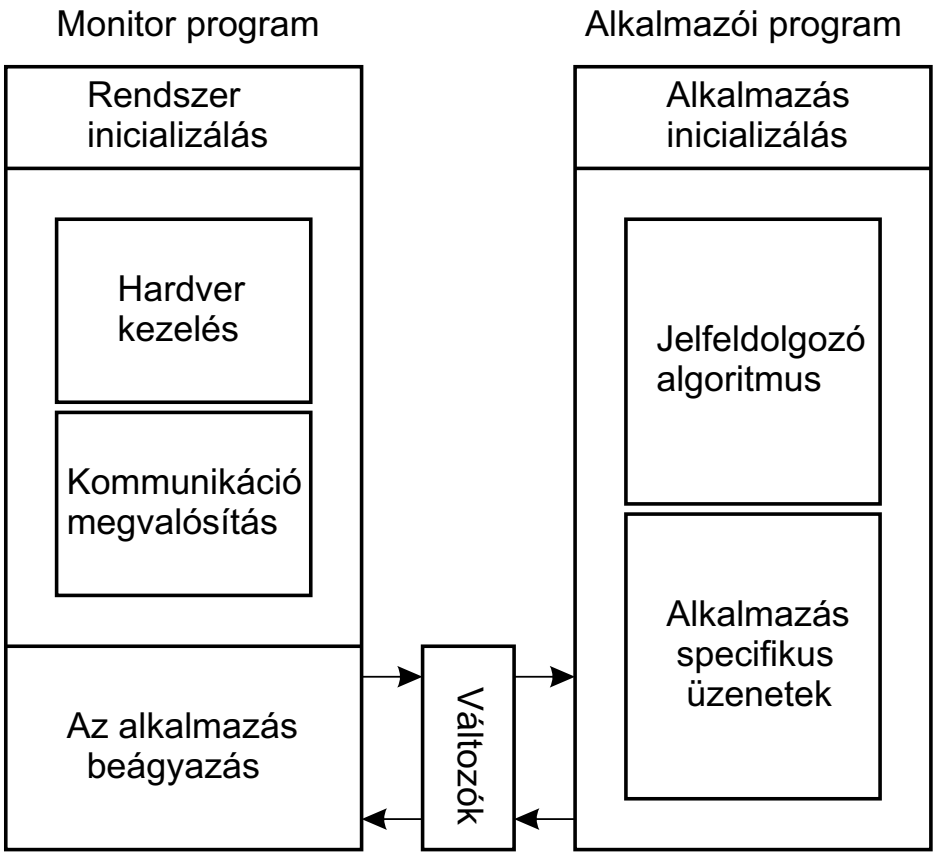
A tápegység a kártyán található eszközök táplálásán kívül védelmi funkciót is betölt. Meggátolja a felhasználó figyelmetlenségéből, vagy a rendszert tápláló egység meghibásodásából eredő károkat (rossz polaritású bekötés, nem megfelelő minőségű táplálás, a megengedettnél nagyobb feszültség). A tápegység megfelelő nagyságú, szűrt feszültségeket állít elő. Fontos követelmény a tápegység számára, hogy analóg és digitális egységeket is táplálnia kell, ezért törekedni kell a későbbiekben ezek megfelkelő elszeparálásra, továbbá a DC csatolás miatt az analóg egységek számára negatív feszültséget is alő kell állítani.

3.2.2. Szoftver rendszerterv

A jelfeldolgozó processzoron futó program rendszerterve az alábbi 3.4. ábrán látható.

Mint azt már korábban is említettem, a DSP-n futó program alapvetően két nagyobb egységre tagolódik: egy rezidens részre, amit monitornak nevezünk, és egy alkalmazás specifikus részre.

Látható, hogy a két egység közötti kommunikáció változókon keresztül valósul meg. A változók reprezentálhatnak mintákat, amelyek a jelfeldolgozó



3.4. ábra. Szoftver rendszerterv blokkvázlata

alkalmazás bemenetét vagy kimenetét képezik, valamint flag-eket, amelyek segítségével a két program bizonyos események bekövetkeztét jelzi egymásnak.

Az egységek felépítése hasonló. Mindkét program inicializálási szakasszal kezdődik, csak míg a monitor a hardver egységeket és a kommunikációt állítja be, addig az alkalmazás a benne implementált jelfeldolgozó algoritmus kezdeti feltételeit adja meg.

Mindkét egység tartalmaz egy részegységet, ami a kommunikáció lebonyolítását végzi. Magát a kommunikációt teljes mértékben a monitor valósítja meg és vezérli. Az alkalmazás csak a benne implementált alkalmazás specifikus módon használja a monitor által létrehozott csatornát.

A kommunikációhoz hasonlóan a hardver egységek kezelése is a monitor program feladata. Az alkalmazás csak változókon és flag biteken keresztül fér hozzájuk.

3.2.3. Kommunikáció rendszerterv

A kártya és a HOST számítógép, illetve a kátya és más egységek közötti kapcsolat egy aszinkron soros port és a rajta megvalósításra került kommunikációs protokoll segítségével jön létre.

A protokol főbb feladatai és tulajdonságai a következők:

- **Elemi egysége az üzenet.** A kommunikáció az egységek egymásnak küldött üzenetei által valósul meg. Az üzenet lehet címzett vagy broadcast (mindenkinek szóló). Az üzenetnek tartalmaznia kell a küldő és a vevő azonosítóját és küldésének célját.

- **Parancs és adat közvetítés.** Az üzenetek alapvetően két nagyobb csoportra oszthatók. Az egyik csoportba az adatok közvetítésére alkalmas nagyobb méretű, a másik csoportba pedig parancsot vagy jelzést közvetítő kisebb méretű üzenetek tartoznak.

- **Az egységek közötti szinkronizáció.** Az üzenetek küldése és fogadása által megvalósul az egységeken futó programok működése közötti szinkronizáció.

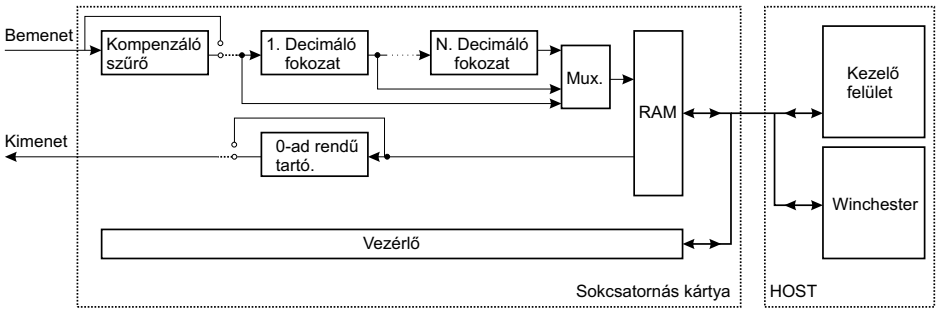
- **Pont-multipont kapcsolat (busz kezelés).** A protokol alkalmas pont-pont kapcsolat (kártya és a HOST számítógép) és pont-multipont kapcsolat (több egység és a HOST) megvalósítására.

- **Adat védelem.** Mindegyik üzenet tartalmaz olyan részeket, melyek segítségével vizsgálható az adott üzenet érvényessége valamint az épsége.

3.2.4. Adatgyűjtő rendszerterv

Megvalósítás előtt magának az alkalmazásnak is rendszertervet kell készíteni. Az adatgyüjtő blokkvázlatát a 3.5. ábra mutatja.

Látható, hogy a bemeneti jel módosítás nélkül, vagy a bemeneti sávszélességet kompenzáló szűrőn keresztül jut el a decimáló fokozatokig. A decimáló fokozatok mindegyike felező, ami egyrészt programozástechnikai, másrészt ütemezésbeli egyszerűsödéshez vezet. Azonos típusú decimáló fokozatokból elvileg végtelen számú darabot el lehet úgy helyezni, hogy ezekből minden mintavételi időpontban csak egy fusson. Az alkalmazott decimáló fokozatok számának a jelfeldolgozó processzor memóriájának mérete, és a legkisebb kívánt mintavételi frekvencia szab határt. A kiválasztott mintavételi frekvenciának megfelelően a multiplexer választja ki a RAM-ba eltárolni kívánt jelet. Visszajátszáskor a megfelelő időzítéssel a RAM-ból kiolvasott minták egy nulladrendű tartón keresztül jutnak el a kimenetig. Mind a felvétel,



3.5. ábra. Az adatgyűjtő rendszerterve

mind a lejátszás folyamatát a vezérlő irányítja a HOST-ton futó kezelőfelület parancsai szerint.

4. fejezet

Hardver tervezés

4.1. Alkatrész választás

A specifikáció és a rendszerterv elkészítését követő lépés a felhasználni kívánt alkatrészek kiválasztása a felsorolt szempontok szerint:

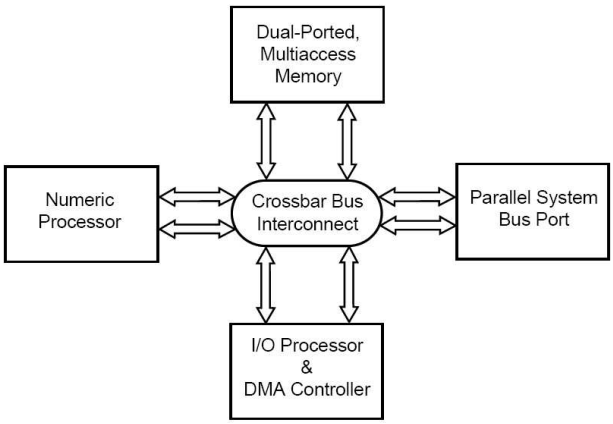
- **Technikai paraméterek.**
- **Ár.**
- **Beszerezhetőség.**
- **Alkalmazási tapasztalatok.**
- **Az eszköz dokumentáltságának mértéke.**

A fenti szempontokat figyelembe véve az alkalmazott eszközök jelentős részét az Analog Devices cég kínálatából választottam ki, könnyű beszerezhetőségük, jó minőségük és a velük kapcsolatos pozitív tapasztalatok miatt.

4.1.1. Az ADSP-21065L jelfeldolgozó processzor

Az ADSP-21065L az Analog Devices leginkább elterjedt és széles körben használt SHARC termékcsaládjába tartozik [7]. A SHARC betűszó a Super Harvard ARhitecture Computer rövidítése. Ez a felépítés a DSP-k körében klasszikusnak számító architektúra továbbfejlesztett változata. A harward architektúra megkülönbözteti a kód-, illetve az adat memóriát. A processzorban a két memóriaterület külön buszon érhető el, így lehetőség van párhuzamos használatukra. Ez a lehetőség a jelfeldolgozó algoritmusokban található műveletek ciklikus elvégzésekor nagymértékben gyorsítja a program futását.

A Super Harward-architektúrában (4.1 ábra) már négy független busz található: két adatbusz, egy utasítás busz és egy IO busz. Ezek a processzszorok egy órajel ütemben egy utasítást hajtanak végre, ezt pedig a lapkán megvalósított utasítás-cache teszi lehetővé.



4.1. ábra. Sharc architektúra

Az ADSP-21065L a SHARC család első tagjának, az ADSP-12000-nek a továbbfejlesztett változata. A processzor mag (processor core) megegyezik e két típusnál, de a 21065-öst néhány fontosabb perifériával látták el. Ezek a következők: belső SRAM, SDRAM vezérlő (ami számunkra sajnos nem megfelelő), IO processzor, IO busz. Mindegyik egység a lapkán került megvalósításra.

Az ADSP-21065L 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 nagy jelfeldolgozó processzortól elvárható.

A SHARC családban található már a 21065L-nél lényegesen nagyobb teljesítményű eszköz is, így az adott feladat igényei szerint ki lehet választani a megfelelő típust.

Az ADSP-21065L teljesitményét jellemző benchmark adatok a következők:

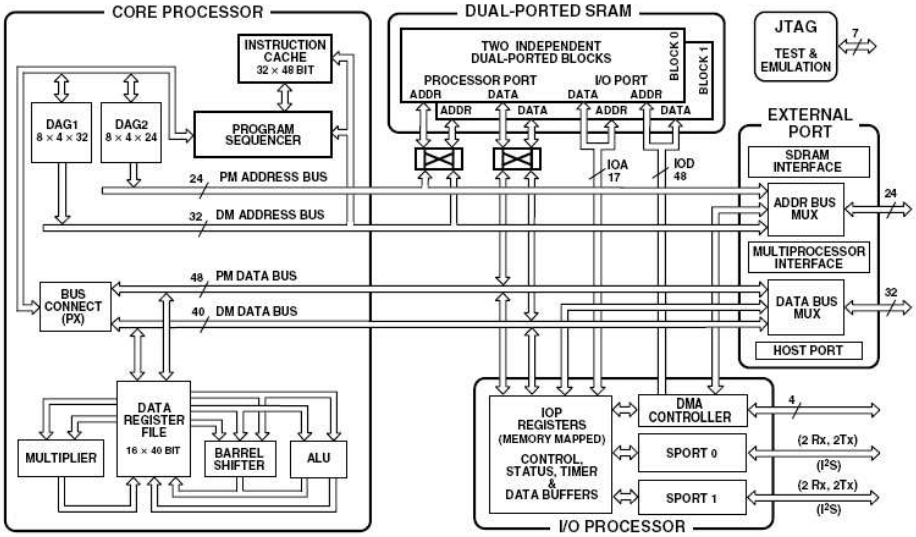
A processzor felépítésének vázlata a 4.2 ábrán látható.

A processzor főbb részei a következők:

- **Művelet végző egység.** Az egység kétféle lebegőpontos számábrázolási formátum használatát támogatja, a 32 bites IEEE single-precision

1024 pontos FFT számítás (Radix 4)	0,274 ms (18221 ciklus)
FIR szűrés (együtthatóként)	15 ns (1 ciklus)
IIR szűrés (per biquad)	60 ns (4 ciklus)
osztás művelet (y/x)	90 ns (6 ciklus)
Inverz gyökvonás ($1/\sqrt{x}$)	135 ns (9 ciklus)
DMA átviteli sebesség	264 Mbyte/sec

4.1. táblázat. ADSP-21065L Benchmark adatok ($f_{clk} = 66MHz$)



4.2. ábra. Az ADSP-21065L felépítése

floating-point szabványos formátumot, és a 40 bites kiterjesztett (extended precision) formátumot. Továbbá lehetőség van 32 bites fixpontos formátum használatára. 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 66 MIPS, de a jelfeldolgozó algoritmusokban használt párhuzamosítás esetén 198 MFLOPS (Million Floatingpoint Operation Per Second) is lehet.

- **Adatregiszterek.** 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 másodlagos (alternate)

regiszterblokkokra. A blokkok közötti váltás egy utasítással történik, és mindig csak az egyik lehet aktív. Ezt a funkciót tipikusan megszakítási rutinok használják ki, mert nem kell a megszakított program környezetét menteni és visszatölteni.

- **Utasítás cashe.** A harvard-erchitektúra nyújtotta előnyök ezen egység segítségével használhatók ki maximálisan. Az utasítás cashe-nek köszönhető, hogy utasítások ciklikus végrehajtása esetén három párhuzamos adatmozgatás és két számítási művelet hajtódik végre egy órajel cikluson belül.

- **Adatcímző egységek.** Az adatcímző egységek egyike a kódterülethez, másika az adatterülethez tartozik. Ezen egységek hardver szinten támogatják a 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 megtörténik. 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 jelfeldolgozási algoritmus (FFT, FIR szűrés) megvalósításakor.

- **SRAM memória.** A chipen megvalósított 544 kbites SRAM memória miatt nem kell külső memóriát biztosítani a processzor működéséhez. Természetesen amennyiben nem lenne elegendő ennyi memória, lehetőség van további külső eszköz illesztésére. A memóriaterület a Harvard-architektúrának megfelelően két részre van osztva (kód, adat terület).

- **Kűlső memória és periféria interfész.** A külső memória illesztésére a külső buszon keresztül van lehetőség. Erre a külső buszra a belső buszok multiplexálva csatlakoznak.

- **HOST interfész.** Ez az interfész nyújt lehetőséget szabványos mikroprocesszoros buszokhoz való illesztésre.

- **DMA vezérlő.** A DMA (Direct Memory Access) vezérlő nagy tömegű blokkos adatmozgatások esetén használatos. Ekkor a processzor magjának igénybevétele nélkül a DMA vezérlő nagy sebességű a belső SRAM memória és valamely más eszköz között. Ezen eszközök lehetnek: külső memória, HOST processzor, soros port.

- **Soros portok.** A DSP négy szinkron soros porttal rendelkezik, amelyek segítségével különböző digitális perifériákhoz való csatlakozásra nyílik lehetőség.

- **J-TAG interfész.** Az IEEE J-TAG 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 használatával ezen porton keresztül a DSP működése nyomon követhető.

Az ADSP-21065 elegendően nagy számítási kapacitással, és komplexitással rendelkezik a sokcsatornás rendszerben felmerülő feladatok elvégzéséhez.

Az ADSP-21065l főbb paraméterei:

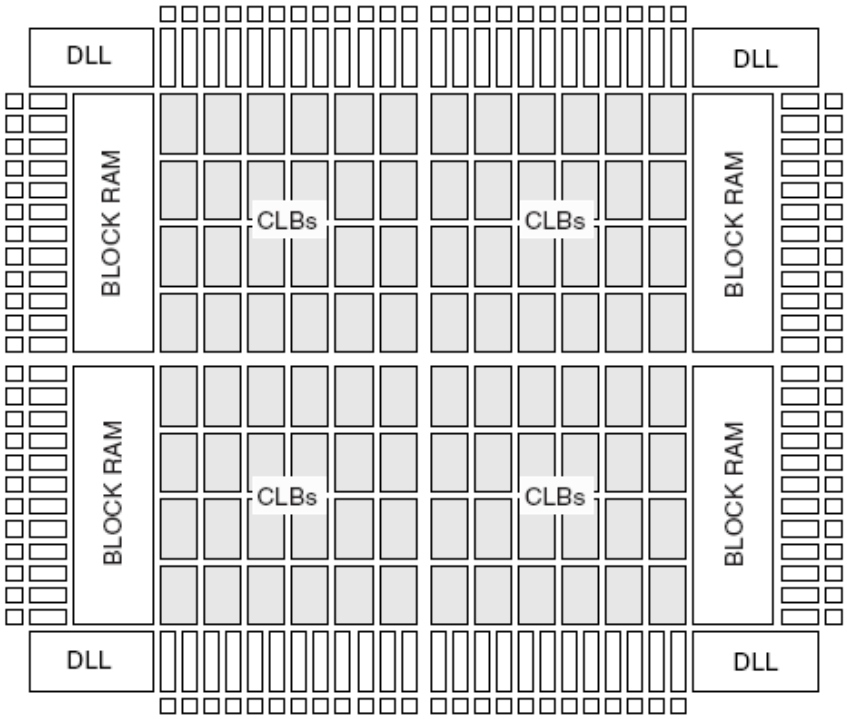
- Fix és lebegőpontos számábrázolás
- 66 MIPS, 198 MFLOPS
- 544 kb on-chip ram
- 3.3V-os tápfeszültség
- Soros, párhuzamos port
- DMA

Az ADSP-21065l főbb alkalmazásai:

- Kommunikáció
- Digitális audio
- Automatizálás
- Méréstechnika

4.1.2. A XC2S150 FPGA

Ez az FPGA (Field Programmable Gate Array) a Xilix cég széles körben elterjedt Spartan II-es családjának tagja [13]. Az eszköz segítségével bonyolultabb szinkron és aszinkron hálózatok valósíthatók meg kis helyen, gyorsan és olcsón. A benne található kész elemek (Blokk RAM, DLL, IO portok) felhasználásával a fejlesztési idő drasztikusan csökkenthető.



4.3. ábra. A Startan II-es család felépítése

Az eszköz felépítése az alábbi 4.3 ábrán látható.

A létrehozni kívánt hálózat komplexitásától függően több vagy kevesebb CLB-ből (Configurable Logic Block) és IO-blokkból áll. A hálózatot a CLB-kben definiált részhálózatok és a felhasznált CLB-k egymással való kapcsolata hozza létre. A CLB-k a köztük hálószerűen elhelyezkedő terület segítségével kapcsolódnak egymáshoz.

Az elkészített hálózat a lapka szélén található IO-blokkok segítségével kapcsolódik a külvilághoz. Minden egyes IO-blokk egy bit széles kapcsolatot jelent, tehát a tok egy lábához csatlakozik. Az IO-blokkban definiálni lehet az adott láb típusát (kimenet, bemenet, tri-state), meghajtó képességét (kisáramú, nagyáramú). Az IO-blokkok csoportokra vannak osztva, és mindegyik csoportnak meg lehet adni a feszültség szintjeit a hozzájuk kapcsolódó V_{ref} lábbal.

Az FPGA-ban található Block RAM, ami tulajdonképpen egy paraméterezhető (választható adatszélesség) szinkron SRAM memóriaterület. Használata a memória területet igénylő alkalmazások esetén jelentős sebességnö-

vekedést és komplexitás csökkenést jelent.

A DLL az FPGA-ban a szinkronizáció elősegítéséhez használható fel.

Mivel a Spartan II családban 4 különféle komplexitású eszköz található (50000, 10000, 150000, 200000 kapuszám), így az elkészíteni kívánt feladat becsült nagyságának függvényében kell kiválasztani a megfelelő tipust. Természetesen a kapu szám növekedésével egyenes arányban nő a realizálható feladat bonyolultsága.

Az FPGA-ban dfiniált hálózat adatait az eszköz egy külső soros elérésű tárolóból olvassa be minden bekapcsolás után.

A XC2S200 főbb paraméterei:

- 432 logikai cella
- 150000 kapu
- 140 darab IO láb

4.1.3. AD/DA átalakító (codec)

Az AD73322L szintén az Analog Devices kínálatából került kiválasztásra [8]. Már a TDK dolgozat keretében is ezt a típust használtuk. A korábbi fejlesztés során szerzett tapasztalatok, valamint a fejlesztési idő csökkentése miatt a jelenlegi fejlesztésben is ez került alkalmazásra.

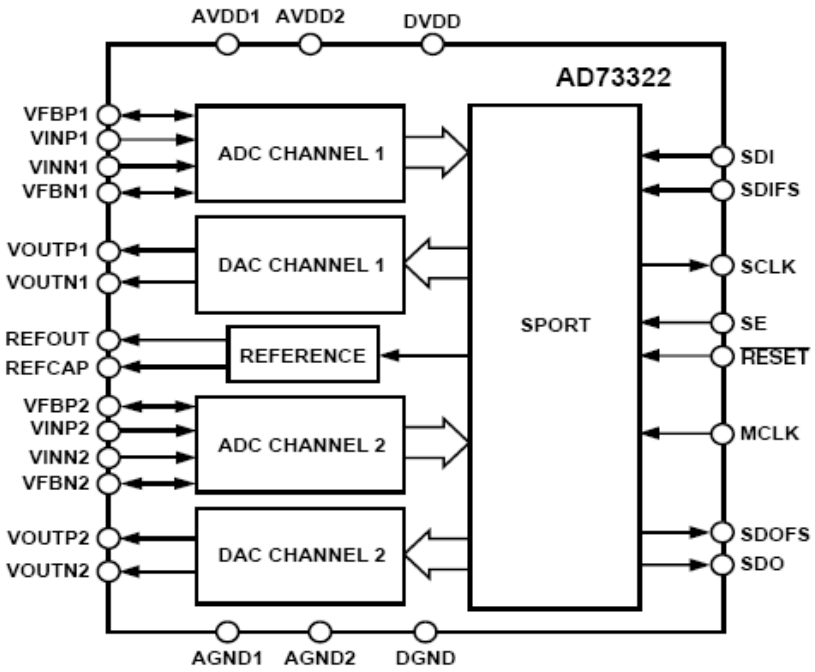
A codec felépítését a 4.4 ábrán találjuk.

Látható, hogy két analóg bemeneti és két analóg kimeneti csatornával rendelkezik. A csatornák aszimmetrikus és szimmetrikus jelek fogadására és adására képesek. A szimmetrikus jelek alkalmazása sok esetben ajánlott, mert növeli az adott áramkör zajjal szembeni védettségét. A codec környezetében található egységek számára a Referencia egység állítja elő a megfelelő stabil feszültséget.

A csatornák végén található AD, DA átalakítók digitális jeleit az SPORT interface alakítja át soros jellé. A soros interfész az SDI bemeneten kapja a soros adatokat, melyeknek a kezdetét az SDIFS (Seria Data Input Frame Syncron) vezetéken megjelenő impulzus jelzi. Az interfész adat kimenete pedig az SDO láb, amelyhez szintén tartozik egy SDOFS jel.

Magát az interfészt az SE (Select) jellel lehet kijelölni. A Reset jel az interfésszel együtt az egyész egységet alaphelyzetbe állítja.

Kifejezetten előnyös tulajdonsága a codec-nek, hogy a benne található analóg és digitális egységek tápcsatlakozási pontjai külön ki vannak vezetve. Ezáltal a fenti egységek könyebben elszeparálhatók egymástól.



4.4. ábra. Az AD73322-es codec vázlatos felépítése

Az AD73322L főbb paraméterei:

- 2db 16 bites szigma-delta AD, szimmetrikus bemenet
- 2db 16 bites szigma-delta DA, szimmetrikus kimenet
- 8, 16, 32, 64kHz-es mintavételi frekvencia
- Beépített referencia feszültség

Az AD73322L főbb alkalmazásai:

- Általános analóg ki-bemeneti periféria
- Beszédfeldolgozás
- Méréstechnika
- Szabályzások

4.1.4. SDRAM

A rendszer nagykapacitású tárolójaként SDRAM memória funkcionál [11]. A kutatás során kiderült, hogy az előirányzott 256 Mbyte kapacitást csak több tok alkalmazásával lehet megoldani, ezért SDRAM modul (DIMM) alkalmazása mellett döntöttem. Mivel ez személyi számítógép alkatrész, ezért könnyen beszerezhető, és rendkívül olcsó (olcsóbb mintha ezt a kapacitást diszkrét tokokból állítanánk össze). Alkalmazása a nyáktervezést is jelentősen egyszerűsíti, mert a szabvány csatlakózóban végződő modul kezeléséhez elegendő 3-4 rétegű nyomtatott áramkör. A konkrét választás a SAMSUNG által gyártott M366S3253DTU típusú PC-133-as modulra esett. Ez a típus a jelenleg elérhető leglassabb modulok közé tartozik, de még ezt a sebességet sem fogja kihasználni a rendszer.

Az SDRAM modul vázlatos felépítése a 4.5 ábra szemlélteti.

Látható, hogy a modul nyolc darab 8 bites adatszélességű diszkrét memóriából áll, melyek két chip select segítségével két négyes csoportba vannak rendezve, ami egyszerűvé teszi a modul adat szélességének csökkentését.

A modul főbb paraméterei:

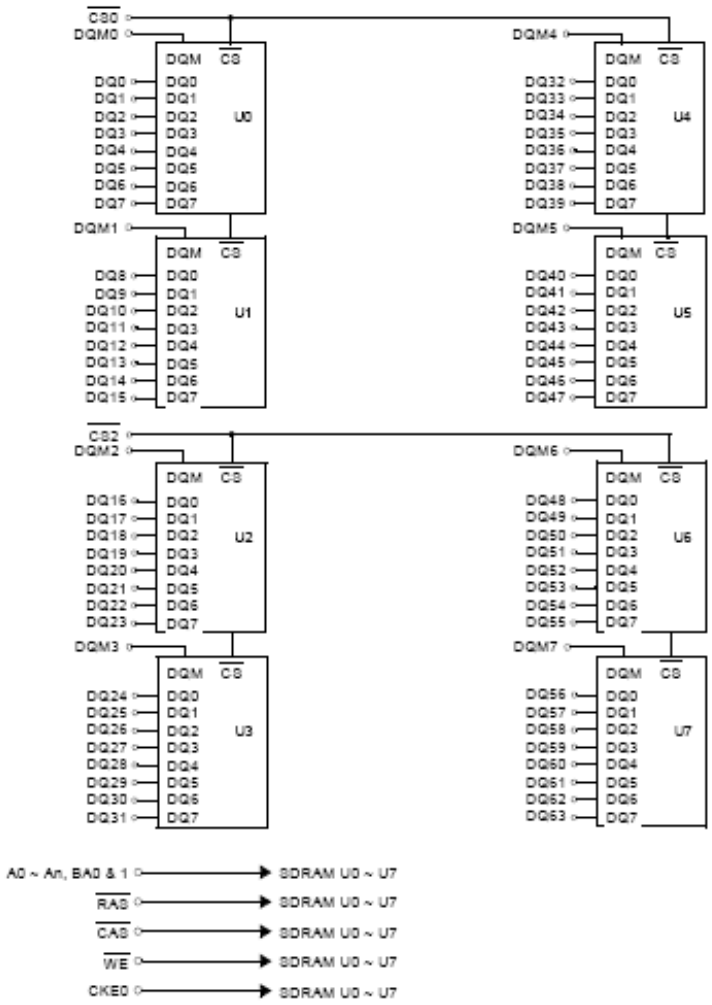
- 32Mx64 szervezésű DIMM
- 4 bank
- 8k refresh
- 3.3V
- Szinkron működés

4.1.5. FLASH

A rendszerben a nemfelejtő memória szerepét az ATMEL által gyártott AT49LV002 típusú FLASH tölti be [9]. Alacsony ára és korábbi alkalmazása során szerzett tapasztalatok miatt esett a választás erre az eszközre.

Az AT49LV002 főbb paraméterei:

- 2 Mbit, 8 bites szervezés
- 3.3V-os tápfeszültség (írás/olvasás esetén)
- Blokkos kezelés



4.5. ábra. Az SDRAM modul vázlatos felépítése

- Irásvédhető Boot block
- 70 ns-os elérési idő

4.1.6. Műveleti erősítők

Mivel a beépítésre kerülő codec-ek csak szimmetrikus, illetve referenciafeszültséggel eltolt aszimmetrikus jeleket képesek adni illetve feldolgozni, ezért szükség van a ki-bemeneteken analóg jelformáló áramkörökre. A bemeneten megjelenő aszimmetrikus jelet el kell tolni a referenciafeszültség értékével,

hogy a codec fel tudja dolgozni, illetve a codec kimenetén megjelenő referenci központú szimmetrikus jelet át kell alakítani aszimmetrikus jellé. A kimeneti áramkörrel szemben támasztott további követelmény, hogy akár egy füllhalgatót is meg tudjon hajtani, ezért a kimeneti erősítő választásánál figyelembe kell venni a terhelhetőségét, és a kettős tápfeszültség melleti működést is. A műveleti erősítőkkel szemben támasztott további kritériumok voltak - többek között - az alacsony fogyasztás, kis bemeneti offszet feszültség és a nagy bemeneti impedancia. Az előbbi kritériumokat alapvetően FET alapú eszközökkel lehet kielégíteni. Ezeket az elemeket szintén az Analog Devices cég kínálatából választottam ki. A kimeneti erősítőként az AD8532 került beépítésre. Mivel egy tokban két műveleti erősítőt tartalmaz, így egy sztereó kimeneti csatornát képes lekezelni.

Az AD8532 főbb paraméterei:

- Kimeneti terhelhetőség: 250 mA
- 2.7-6 V tápfeszültség
- 3 MHz-es sávszélesség

Az AD8532 főbb alkalmazásai:

- Audio alkalmazások
- ASIC kimeneti, bemeneti erősítő
- Fejhalgató meghajtó

Bemeneti buffernek pedig az AD8042-t választottam, legfőképpen azért, mert ezt az eszközt kifejezetten AD átalakítók elé ajánlják.

Az AD8042 főbb paraméterei:

- 160 MHz-es sávszélesség
- 3, 5, 10 V tápfeszültség
- Nagy bemeneti impedancia

Az AD8042 főbb alkalmazásai:

- AD meghajtóként
- Video kapcsolókban

4.1.7. UART vezérlő

Az aszinkron soros port megvalósítására a Texas által gyártott TL16C550-es IC-t választottam [12]. Ez az eszköz általánosan elterjedt, gyakorta használták régebbi PC-kben is. Alkalmazása azért szükséges, mert a DSP nem rendelkezik ilyen típusú perifériával.

Az TL16C550 főbb paraméterei:

- 1 csatorna
- 16 bájtos ki-bemeneti FIFO
- 3, 5V tápfeszültség
- 1Mbps maximális sebesség

4.1.8. Feszültség stabilizátorok

A kártyán található egységek táplálására lineáris stabilizátorokat használtam. A három különböző feszültség három fajta stabilizátor alkalmazását teszi szükségessé. Az áramkör nagy részét tápláló 3.3V-os feszültséget a Linear Technology által gyártott LT1086 fix kimeneti feszültségű stabilizátor állítja elő. Az FPGA 2.5V-os magfeszültségét egy LM317MDT változtatható kimenetű stabilizátor szolgáltatja, míg az analóg áramkörök -1.7V-os feszültségét egy 3.3V-ra kapcsolt referenciájú 79M05 állítja elő.

Az LT1086 főbb paraméterei:

- 3.3V-os kimeneti feszültség
- 1.5A maximális kimeneti áram
- Low dropout

Az LM317MDT főbb paraméterei:

- Változtatható kimeneti feszültség
- 0.5A maximális kimeneti áram

Az 79M05 főbb paraméterei:

- -5V kimeneti feszültség
- 0.5A maximális kimeneti áram

4.1.9. Csatlakozók

Az analóg ki-bemenetek RCA csatlakozókon keresztül kapcsolódnak a külvilághoz. Az RCA csatlakozók általánosan elterjedtek az audio alkalmazásokban. A mechanikailag robosztus, bontható mono csatlakozás megfelelően viselkedik nagyobb igénybevétel esetén is. A csatlakozók nyomtatott áramkörbe szerelhető verzióját választottam, mert így megspórolható a kábelezés.

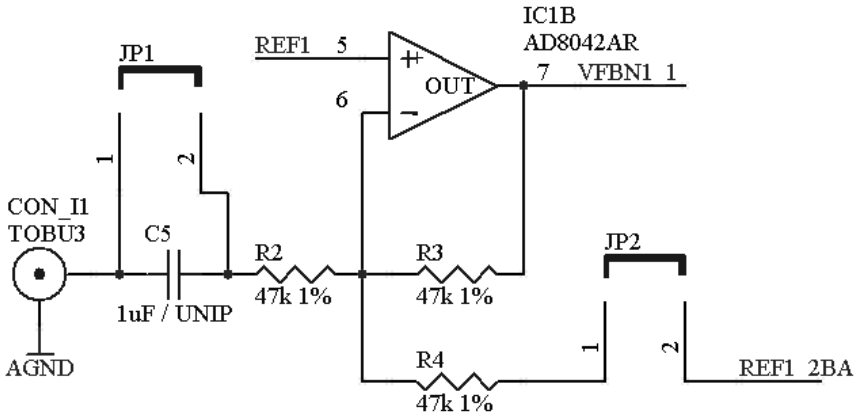
A soros port szabványos 9 pólusú szintén áramkörbe ültethető hüvelyben csatlakozóban végződik.

A kártya tápcsatlakozója pedig egy három pólusú beültethető sorkapocs, ami megfelelően szilárd, de bontható kötést hoz létre.

4.2. Az egyes áramköri egységek

4.2.1. Analóg bemenet

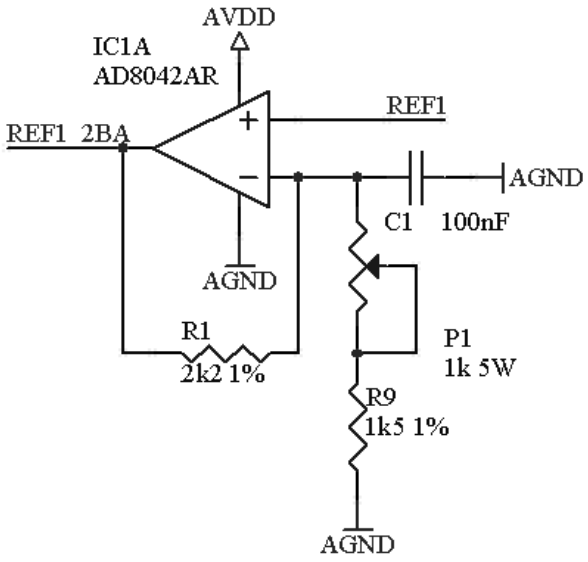
Az analóg bemenetek kapcsolási rajzai a függelék 5.-8. oldalán találhatók.



4.6. ábra. Analóg bemeneti csatorna

Az egységek mindegyike egy sztereo bemenetet illeszt a hozzá tartozó codec bemeneteihez. A jeleket képesek AC illetve DC csatolt módon is fogadni. A csatolási módok között jumperek segítségével lehet váltani. Az analóg bemeneti egységek alapvetően négy nagyobb komponensből épülnek fel:

- Bemeneti jelformáló jobb csatorna



4.7. ábra. Referencia kétszerező

- Bemeneti jelformáló bal csatorna
- Referenciakétszerező a jobb csatornához
- Referenciakétszerező a bal csatornához

Látható, hogy a két jelformáló komponens, és a két referencia kétszerező komponens teljesen megegyezik. Az illesztést tulajdonképpen invertáló üzemmódban bekötött műveleti erősítő végzi, melynek a referencia pontja (nem invertáló bemenet) a codec referencia kimenetére van kötve, ezáltal a bemeneti jel referenciafeszültséggel eltolva kerül a codec bemenetére. Az AC csatolást $1\mu F$-es unipoláris kondenzátor valósítja meg. DC csatolás esetén az imént említett kondenzátort zárjuk rövidre egy jumperrel. Ha csak enynyit tennénk, akkor a bemenet csak a referencia feszültség környezetében (+/- 0.68V) lenne képes a jel fogadására a codec bemenetének kivezérlése nélkül. A referenciakétszerező komponens ezt a problémát küszöböli ki, tehát az erősítőre kerülő jelet eltolja a referenciafeszültség értékével, hogy a jel az erősítőt a megfelelő munkapontban vezérelje. A DC csatolás bekapcsolásához a kondenzátor söntölése mellett az offset jelet is be kell kapcsolni az erősítőhöz.

A referencia kétszerező tulajdonképpen egy egyszerű nem invertáló erősítő, melynek kétszeres erősítését finoman szabályozni lehet. Az erősítés finomállítását

egy potméterrel és a vele sorba kötött ellenállással lehet megoldani. Az erősítés átfogása:

$$A_{min} = 1 + (1.5kOhm/2.2kOhm) = 1.68 \tag{4.1}$$

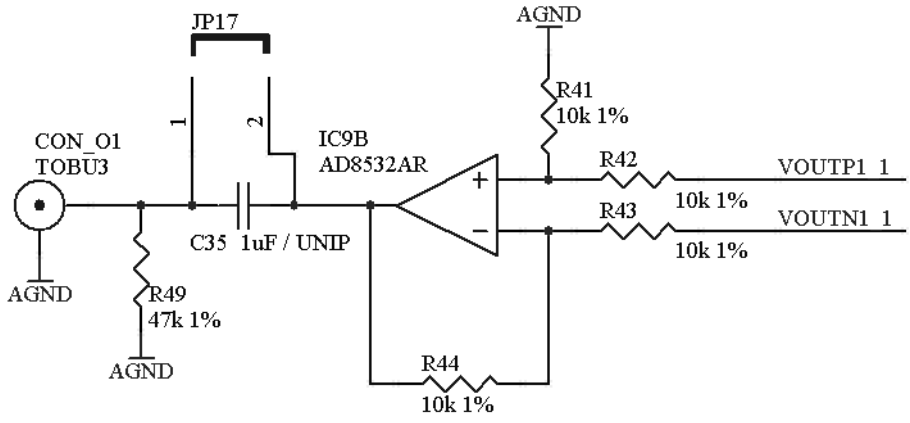
$$A_{max} = 1 + (2.5kOhm/2.2kOhm) = 2.13 \tag{4.2}$$

Az analóg bemeneti jelet az egység egy RCA csatlakozón keresztül fogadja. Ez a csatlakozó típus rendkívül elterjedt, és mono volta miatt könnyű kezelhetőséget eredményez.

A komponensek megfelelő működéséhez hozzájárulnak a műveleti erősítők tápja és földje között párhuzamosan elhelyezett 100nF-os és $10\mu F$-os kondenzátorok. A kondenzátorok az IC-k tápján lévő zajokat szűrik ki lokálisan, melyek egyébként az erősítők kimeneti jelének kismértékű tápfeszültség függése miatt megjelennének a feldolgozandó jelben.

4.2.2. Analóg kimenet

Az analóg kimenetek kapcsolási rajzai a függelék 9.-12. oldalán találhatók.



4.8. ábra. Analóg kimeneti csatorna

Az egységek két teljesen azonos komponensből épülnek fel, sztereo kimenetről lévén szó. A kivonó üzemmódban bekötött műveleti erősítő alakítja át a codec szimmetrikus kimenetét aszimmetrikussá, mindemellett teljesítmény erősítést végez a nagyobb árammal való terhelhetőség érdekében. Az analóg bemeneti egységekhez hasonlóan a kimenetek itt is lehetnek AC illetve

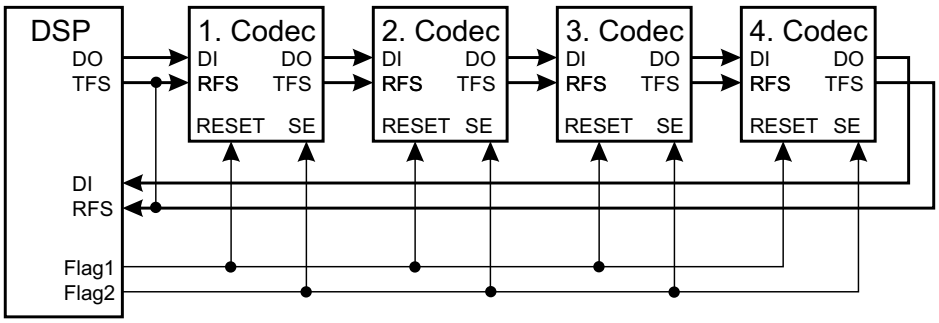
DC csatoltak. A DC csatolást jelen esetben is a csatoló kondenzátor rövidre zárásával lehet elérni. A DC csatolás tette szükségessé a negatív tápfeszültség alkalmazását, hogy az erősítők kimenete le tudjon menni negatív értékekre is. A kivonó erősítők erősítése egy, tehát a kimeneti feszültsége megegyezik a codec-ek szimmetrikus kimeneti vezetékeinek potenciálkülönbségével.

Ezekben az egységekben is találhatók tápszűrő kondenzátorok, csak ebben az esetben a pozitív tápfeszültséget és a negatív tápfeszültséget is külön kell szűrni.

Az egyszerű kezelhetőség érdekében itt is RCA csatlakozókat alkalmaztam.

4.2.3. Codec-ek

A codec-ek és környezetük teljes kapcsolási rajza a függelék 4. oldalán látható.



4.9. ábra. A codec-ek láncolatának blokkvázlata

A négy kaszkádosított AD73322-es codec hozza létre a kapcsolatot a jelfeldolgozó processzor és az analóg egységek között. 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 az elsőn kívül a harmadik codec-el is kapcsolatban van. A harmadik codec, amely szintén slave, már csak a második, illetve a negyedik slave-hez kapcsolódik. A masteren kívül a negyedik codec csatlakozik még a DSP-hez. Az ábrán a codec-ek láncát a vastag vonal mutatja. Az adatok egy irányban mozognak a láncban. A DSP-től indulva az elsőn, majd a másodikon, harmadikon végül a negyediken keresztül visszajut a processzorba.

A láncszerűen összekapcsolt soros adat és frame sync jeleken kívül egyéb jeleket is igényelnek a codec-ek. Az egyik legfontosabb jel a kommuniká-

ció sebességét meghatározó SCLK, amit a master codec állít elő osztással a kártyán lévő MCLK jelből. A kártyán lévő oszcillátor mellett a CON12-es csatlakozó által más oszcillátorok is csatlakoztathatók a kártyához. 16.384MHz esetén a lehetséges mintavételi frekvenciák 64, 32, 16, 8 kHz. A 16.384MHz-es oszcillátor nehéz beszerezhetősége miatt jelenleg 16MHz-et alkalmaztam. Egyéb közös vezérlőjelek még a RESET és a SELECT DEVICES. A RESET jel alaphelyzetbe állítja a codec-eket, ami után megkezdődhet a beállítási folyamat. A SELECT DEVICES jel tulajdonképpen chip select-ként működik, segítségével további kaszkádosítás is elképzelhető. A két jelet a DSP állítja elő a Flag1 illetve a Flag2 lábain, és egy D flip-flop-okon keresztül jutnak el a codec-ekig. A flip-flop szinkronizálja a jeleket a codec-eket vezérlő MCLK jelhez.

A codec-ek tápjának szűrése rendkívül fontos volt. Mivel minden egyes codec külön csatlakozik mind az analóg, mind a digitális táphoz, ezért minden pontot megfelelően meg kellett szűrni. A codec-ek az analóg táphoz tekercseken keresztül kapcsolódnak, és ez a pont kondenzátorokkal tovább van szűrve. A codec-ek digitális tápját elegendő volt csak kondenzátorokkal szűrni. Továbbá a codec-ek által előállított referencia jelek szűrése is 100nF-os kondenzátorokat igényelt, amelyeket a codec-ek REFCAP lábaira kellett csatlakoztatni.

4.2.4. DSP és környezete

A DSP és környezete kapcsolási rajza a függelék 1. oldalán látható.

Észrevehető, hogy a kapcsolási rajz hét nagyobb egységre tagolódik:

- Órajel generátor
- Reset és watchdog áramkör
- J-TAG interface
- FLASH memória
- Csatlakozók
- Fel-lehúzó ellenállások
- Processzor

Az órajel generátor szerepét egy oszcillátor látja el, melynek tápja a szokott módon kondenzátorokkal van meghidegítve.

A reset és watchdog funkciókat a Maxim által gyártott MAX6303 típusú IC látja el. Az aktiv L szintű reset kimenete a DSP reset bemenetére csatlakozik, míg a watchdog törlő jelet a DSP Flag0 lábától kapja. A reset szintjét az R122-es és R123-as ellenállásokból álló osztó határozza meg, jelen alkalmazás esetén ez az érték 3.05V a következő képlet szerint:

$$Urst = 1.22 * (R122 + R123)/R123. \tag{4.3}$$

Az osztó alsó tagjának lábait egy kétpólusú csatlakozóra is kivezettem, és ennek rövidrezárása reset gombként funkcionál. Jumperrel folyamatosan rövidre lehet zárni, és ez a kártya élesztésekor majd fontos szerepet fog betölteni. A watchdog timer működését a C140-es és a C142-es kondenzátor befolyásolja. A C142-es kondenzátor a watchdog timer periódus idejét, míg a C140-es kondenzátor a reset impulzus hosszát határozza meg. Látható, hogy a C142-es kondenzátort söntölni lehet egy jumper segítségével. Ez akkor hasznos, ha fejlesztés közben J-TAG-et használunk, és éppen leállítjuk a program futását. Ebben az esetben a törlés elmarad, és a reset adott időn belül bekövetkezne, ami a J-TAG és a leállított program közti szinkron elvesztéséhez vezetne.

A J-TAG a processzorhoz egy 14 pólusú csatlakozón keresztül kapcsolódik. A csatlakozó 3-as lábának hiánya a J-TAG helyes elhelyezését segíti elő.

A FLASH memória 8 bites adatbuszon, és 17 bites címbuszon keresztül kapcsolódik a processzorhoz. A reset láb a bekapcsoláskor alaphelyzetbe állítja a memória vezérlőjét, illetve 12V rákapcsolásával, a memóriában található lezárt boot block-ot újra írhatóvá teszi. A memória CS jelét a processzor BMS és MS0 nand kapcsolata adja. A BMS jel a reset utáni boot folyamat alatt aktív, míg az MS0 a DSP egy bizonyos címtartományát (0x2000-0xFFFFFF) biztosítja a memóriának.

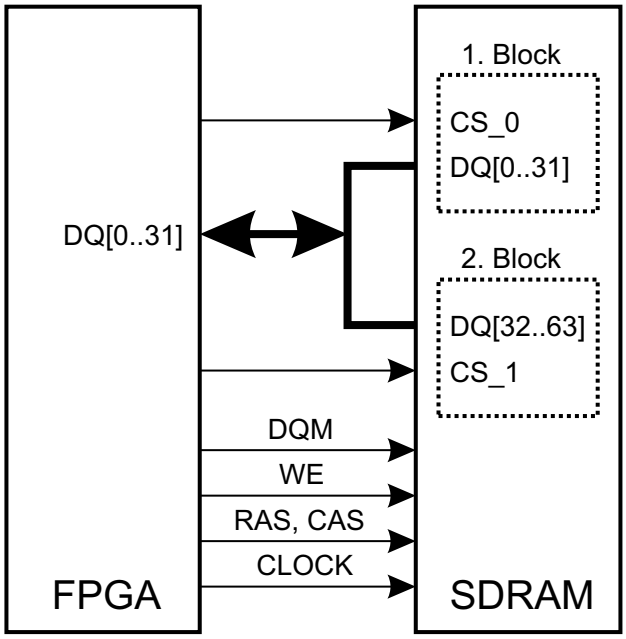
A JP27-es jumper az alkalmazói program indításában játszik majd szerepet. Ha a felhasználó rövidre zárta azt, akkor a reset után az alkalmazás automatikusan elindul. A CON11-es csatlakozó által pedig a kártyán fel nem használt soros port (SPORT1), flag (Flag8, Flag9) és PWM lábakat lehet elérni a későbbi felhasználhatóság céljából.

A felhúzó ellenállások az open-collectoros kimenetek működését segítik elő, illetve a bemenetek megfelelő kezdeti állapotát adják meg. Az R120 és R121 lehúzó ellenállások a codec-eket vezérlő jelek kezdeti értékeit állítják elő.

A processzor tápjának megfelelő szűréséről a tok négy sarkán elhelyezett 100nF-os és $10\mu F$-os kondenzátorok gondoskodnak.

4.2.5. Az FPGA és az SDRAM

Az FPGA és az SDRAM kapcsolási rajzai a függelék 2.-3. oldalán látható.



4.10. ábra. Az FPGA és az SDRAM kapcsolatának vázlata

Az FPGA 50MHz-es órajelét az OSC2-es oszcillátor szolgáltatja, programját pedig a hozzá soros interfészen keresztül kapcsolódó konfigurációs memória tárolja.

Az FPGA programjának fejlesztése és a konfigurációs memória tartalmának feltöltése a J-TAG port-on keresztül történik (CON7). A két eszköz láncba van kötve, ezt az elhelyezést a letöltő program támogatja. Az FPGA a 3.3V-on kívül, ami a kimeneti egységeit (IO block) táplálja, még 2.5V-ot is igényel. Ez a feszültség a magot táplálja.

Az FPGA és az SDRAM tápfeszültségeinek szűrését jelen esetben is a tok és a foglalat mentén egyenletesen elosztott kondenzátorok végzik.

Az SDRAM 64 bites adatbusza 32 bitesre lett szűkitve úgy, hogy a két chip select segítségével a két 32 bites csoportot külön lehet elérni, a chip select-eket külön vezérelve a csoportok közösíthetők. Az SDRAM órajelét a memóriavezérlő szolgáltatja.

4.2.6. Soros kommunikációs periféria

A soros kommunikációs perifériát a Texas Instruments által gyártott 16C550-es vezérlő valósítja meg, kapcsolási rajza a függelék 13. oldalán látható. Az egység 8 bites adat buszon és 3 bites címbuszon keresztül csatlakozik a processzorhoz. A vezérlő jeleket (CS,RD,WR) az FPGA-tól kapja, amelyek tulajdonképpen a DSP megfelelő módon átütemezett jelei. A CS-et felhúzó ellenállás az FPGA definiálatlan állapota esetén jut szerephez, tíltva a vezérlő busz hozzáférését. A vezérlő órajelét az 1.8432MHz-es kristály oszcillátor állítja elő.

A vezérlő INT kimenetét invertálni kellett, hogy a DSP aktív L szintű IRQ0 bemenete megfelelő, szintérzékeny módon fogadhassa.

A vezérlő soros ki-bemenetei és a modem vezérlő jelei a CON3 és CON4 csatlakozóra illeszthető RS-232 vagy RS-485 illesztőpanelen keresztül kapcsolódik a CON2-es soros port csatlakozóhoz.

4.2.7. RS-232/RS-485 illesztő

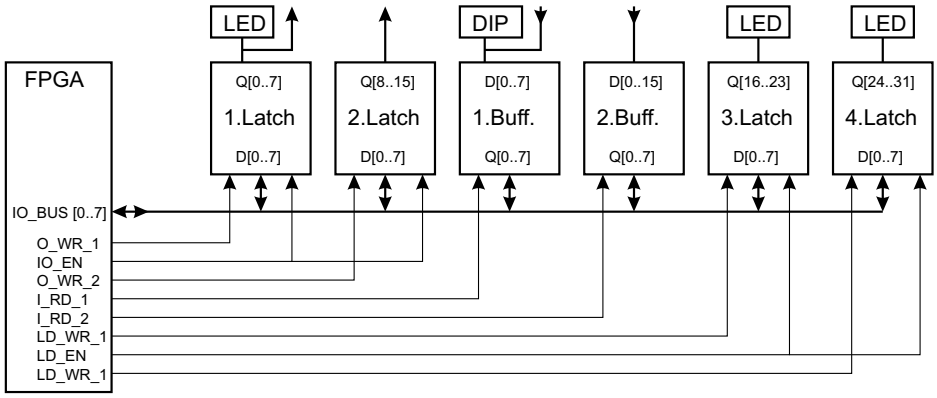
A két illesztő áramkör kapcsolási rajza a függelék 16.-17. oldalán található.

Az RS-232 illesztést egy MAX232-es áramkör valósítja meg. Mivel az IC 5V-os tápfeszültséget igényel, ezért a tápját a +POW feszültségre kötöttem ami 5 és 6V között lehet. Az IC és a 3.3V-os tápfeszültségű soros vezérlő közötti szintillesztést ellenállások valósítják meg. A DSP Flag4-es lábát ezen a panelen felhúztam az R3-as ellenállás segítségével, ami ezt a panelt azonosítja.

Az RS-485 illesztést pedig az IL485-ös IC valósítja meg. Mivel az IC galvanikusan leválaszt, ezért a leválasztott oldal táplásáról is gondoskodni kellett, ami egy DC/DC segítségével történik. A panel azonosítása céljából a Flag4-es lábat ebben az esetben földre húztam. Ez a láb az azonosítás mellett a meghajtó adás engedélyezését is vezérli. A vevő engedélyezésének módját a JP3-as jumper segítségével lehet kiválasztani. Ha az adás engedélyezésének negáltját alkalmazzuk, és ha az illesztő éppen nem ad, akkor vételi üzemmódban van. Ha állandóan földre kötjük az előbbi lábat, akkor a vevő állandóan be van kapcsolva. Ez az üzemmód alkalmas a busz állapotának figyelésére, és a kiküldött adatok ellenőrzésére. A leválasztott oldalon (busz oldal) a két buszvezetéket az R2 és az R3 ellenállás feszíti elő. A JP1 jumper bekapcsolásával lehetőség nyílik a busz 120 ohm-al való lezárására. További lehetőséget nyújt a JP2 jumper, amelynek segítségével a buszvezeték árnyékolását 100 ohm-on keresztül le lehet földelni, ami a busz zajok elleni védettségét növeli.

4.2.8. Digitális IO

Az egység kapcsolási rajza a függelék 14. oldalán található.



4.11. ábra. A digitális IO blokkvázlata

Az egység 16 bites bufferelt ki-bemenetet és a bemenetek melletti kivezérlés jelző LED-ek meghajtását valósítja meg. Bufferelt kimenetekként 74LVC573-as 8 bites latch-ek funkcionálnak, míg a bemeneteket 74LVC245-ös kétirányú buszmeghajtók fogadják. A latch-ek és a buszmeghajtók egy 8 bites buszra vannak felfűzve, hogy ez a periféria az FPGA-tól minél kevesebb lábat igényeljen. A periféria használatánál figyelembe kell venni, hogy a kibemenetek állapotai két külön időpontban 8 bitenként frissülnek. A kimenetek engedélyezését az LD_EN és az IO_EN jelek végzik, melyeknek állapotát a periféria speciális regiszterében lehet beállítani. A periféria az LD_WR_1, LD_WR_2, O_WR_1, O_WR_2, I_RD_1, I_RD_2 jelek segítségével kapcsolja a buszra a kiválasztott meghajtót, vagy latch-et. A kimenet alsó nyolc bitjének állapotát LED-ek is jelzik, míg a bemenet alsó nyolc bitjét DIP kapcsolóval is be lehet állítani. Ez lehetővé teszi egy primitiv kezelői felületet gyors megvalósítását. A kimeneteket és a bemeneteket, valamint a DSP Flag7-es és az IRQ2-es lábát a CON5-ös csatlakozóra vezettem ki. Mind a latch-ek, mind a buszmeghajtók vezérlő lábai fel vannak húzva tápfeszültségre, megakadályozva a hibás működést az FPGA definiálatlan tartalma esetén, amikor az IO lábai szabadon lebegnek.

4.2.9. Tápegység

A tápegység kapcsolási rajza a függelék 15. oldalán található.

Az egység bemenete a CON8-as csatlakozó, ahová $\pm 6V$ DC-t kell kötni a helyes működés érdekében. A bemenetet polaritásvédő diódák (D4, D5)

fogadják. Ezek a diódák hivatottak kivédeni a kártya fordított polaritású bekötését, ami a leggyakoribb felhasználói hiba. A kártyára az előírtnál magasabb bemeneti feszültség kapcsolását a D6 és D7 szupresszor diódák akadályozzák meg. A stabilizátorok elbírnának nagyobb bemeneti feszültséget, de akkor a disszipáció jelentősen nőne, ami a kártyán található analóg egységek működése szempontjából káros hőtermelődéssel járna. A bemeneti feszültség ingadozásait a C122, C123, C124 puffer kondenzátorok csillapítják, illetve szűrik. A bemeneti feszültségek meglétét a D2, és a D3 LED-ek jelzik.

A kártyán található egységek számára a megfelelő tápfeszültségeket négy darab lineáris stabilizátor állítja elő. Mindegyik stabilizátor kimenete és bemenete 100nF-os kondenzátorokkal van hidegítve, elkerülendő a káros gerjedést. A 3.3 V-ot két darab LT1086 állítja elő. Elvileg egy darab is elbírná a jelenlegi terhelést, de a kisebb disszipáció, és a későbbi fejlesztések igényelte nagyobb teljesítményigény céljából ezekből kettő került beépítésre a kártyára.

A digitális és az analóg egységek soros induktivitás és párhuzamos kapacitás segítségével szeparálódnak el egymástól.

Az FPGA magfeszültségét egy változtatható kimeneti feszültségű stabilizátor állítja elő a következő képlet szerint:

$$Uki = 1.25 * (R117 + R118)/R118. \tag{4.4}$$

Mivel a negatív tápfeszültség nem lehet nagyobb, mint 2.7 V, és ilyen feszültséghez nem találtam megfelelő terhelhetőségű és tokozású eszközt, így a 79M05 alkalmazása mellett döntöttem. A stabilizátor referencia pontját 3.3V-ra kötve a kimenetén -1.7 V-ot állít elő.

Minden stabilizátor kimenet $10\mu F$ tantál kondenzátorral lett megszűrve a kisebb zaj érdekében.

A digitális tápot és az FPGA magfeszültségét előállító tápegységek után mesterséges szakadások kerültek beépítésre. Ezek alkalmazása élesztéskor jut fontos szerephez, mert így a tápot bármikor ellenőrizni lehet anélkül, hogy rákapcsolnánk a kártya többi részére.

4.3. A nyomtatott áramkör

A kártya tervezése Protel DXP fejlesztő környezetben történt. Az elkészitett panel rétegeinek rajzai a függelék 18.-21. oldalán, a beültetett panel fotója pedig a függelék 22. oldalán található.

4.3.1. Analóg és digitális kevert áramkör problémái

A tervezésnél 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 lehet küszöbölni a problémát. A lokális hidegítésről (az aktív eszközök tápjának szűréséről) a fentiekben már volt szó, a tervezésnél csak arra kellett ügyelni, hogy a szűrő kondenzátorok a lehető legközelebb kerüljenek a hidegíteni kivánt IC-hez. A megfelelő szeparációt pedig az egyes egységek nyomtatott áramkörön való gondos elrendezésével, és csillagpontos földeléssel, illetve táplálással lehetett elérni.

4.3.2. Az alkatrészek elrendezése

A kártya elrendezése többek között az alábbi kényszerek hatására jött létre:

- Az analóg és digitális egységek különüljenek el
- Az analóg ki-bemenetek egy oldalra kerüljenek
- A kártya mérete minél kisebb legyen

Első lépésként az egyik analóg ki-bemeneti egység elrendezésére került sor. Az RCA csatlakozók egy HIFI torony hátlapjának megfelelő távolságban lettek elhelyezve, biztosítva azt, hogy bármely csatlakozót használni lehessen. Ezek után mindegyik analóg egységet az előbbi mintájára kellett elrendezni. A 16 RCA csatlakozó jelentős távolságot tett ki, így ez határozza meg a kártya hosszméretét. Ezek után az egységekhez tartozó codec-ek kerültek felhelyezésre, a hozzájuk tartozó kondenzátorokkal és induktivitásokkal együtt.

Második lépésként a DSP és a hozzá tartozó egységek (reset, oszcillátor, J-TAG csatlakozó, FLASH memória szűrőkondenzátorok) lettek elhelyezve. Mivel a FLASH memória jelentős felületet foglal el, és sok vezetéken kepcsolódik a processzorhoz, így a legoptimálisabb hely számára a DSP alatti felület.

Ezek után az elrendezett analóg bemeneti perifériákhoz illesztettem a DSP-t és környezetét, úgy, hogy a codec-ek és a DSP közti soros port huzalozása minél egyszerűbb legyen.

Következő lépésként az FPGA helyét kellett megtalálni, amihez a DSP melletti terület mutatkozott megfelelőnek. Az FPGA lábait a tervezés során többször is át kellett rendezni a minél egyszerűbb huzalozás érdekében.

Az SDRAM elhelyezése tulajdonképpen adta magát, mert jelentős hosszmérete miatt máshová nem is lehetett volna elhelyezni, mint a processzor és az FPGA mellé.

A digitális IO megfelelő elrendezés mellett elfért az FPGA alatti szabad területen. A busz egyszerű huzalozását az tette lehetővé, hogy a latch-ek és buszmeghajtók adatlábai a tervezés folyamán át lettek rendezve.

A tápegység és a tápcsatlakozó a digitális IO alatt került elrendezésre. A két 3.3V-os stabilizátor IC fölé SMD hűtőbordák kerültek.

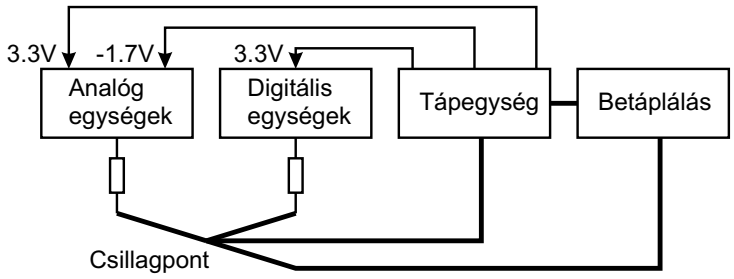
A soros port és a vezérlő a kártya jobb alsó sarkában kapott helyet. A soros port csatlakozója így az analóg bemenetekkel szemközti oldalra került, ezáltal az ott fellépő zavaró hatású jelek megfelelően el vannak különítve.

4.3.3. Huzalozás és földelés

Az alapkoncepció szerint felületszerelt alkatrészek kerültek fel a kártyára, ezáltal jelentősen csökkent az általuk elfoglalt felület nagysága. Továbbá amiatt, hogy a lábak helyén nem voltak furatok, és nem kellett kerülgetni azokat alkalmazásuk a huzalozást is egyszerűsítette. Mivel a tervezés során az volt a cél, hogy a kártya minél kisebb legyen, így az alkatrészeket viszonylag sűrűn helyeztem el, ami kettőnél több réteg használatát tette szükségessé.

A huzalozást négy rétegű panelen kezdtem el, ami a későbbi esetekben elégnek bizonyult. A két belső réteget a táp és a föld vezetékei (fólia szigetei) foglalták el.

A földelés kialakításánál figyelni kellett arra, hogy topológiája csillagpontos legyen. A csillag középpontjában a stabilizátorok referenciapontjai állnak. A csillagpontos földelés lényege, hogy az egységeken keresztül folyó áramok külön úton jutnak el a közös csillagpontba. Mivel az adott pont (felület) ellenállása rendkívül alacsony, ezért az egyes egységek földpotenciálja nem fog eltolódni. A csillagpontos földelés sematikus rajzát a 4.12 ábrán láthatjuk.



4.12. ábra. A csillagpontos földelés blokkvázlata

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 L, C tagok segítségével, a már említett módon történt.

A másik belső rétegen pedig a tápvezetékek kerültek megvalósításra fóliaszigetek formájában. A két belső réteg alkalmazásának további előnye az, hogy a nagy felületű táp és föld fóliaszigetek között létrejövő parazita kapacitás is részt vesz a zaj szűrésében.

5. fejezet

A hardver élesztése és programozása

5.1. Beültetés és élesztés

Az élesztés művelete párhuzamosan történt az alkatrészek beültetésével. Mivel első verzióról van szó, a lehetséges tervezési hibákból adódó károsodások megelőzése érdekében az alkatrészek nem egyszerre kerültek beültetésre, hanem több lépésben.

Mivel a rendszer hardver és szoftver egységeinek működése nem függetleníthető, ezért a rendszer élesztése során a két egységet párhuzamosan, lépésről lépésre kellett bővíteni.

5.1.1. A nyomtatott áramkör

Először az elkészült nyomtatott áramkör tüzetes átvizsgálására került sor. Alapvetően a gyártás során keletkezett hibák keresése és elhárítása volt a cél, de ezek mellett néhány apróbb konstrukciós hibára is fény derült. Meg kellett vizsgálni, hogy az egyes fóliaszigeteken belül elhelyezkedő, de nem ahhoz tartozó átvezetések, alkatrészlábak valóban el vannak-e szigetelve. Ellenőrizni kellett továbbá, hogy az alkatrészek táp és földpontjai helyes polaritással és alacsony (közel nulla ohm) impedanciával csatlakoznak-e a tápegység megfelelő pontjaihoz. Szintén ellenőrizni kellett az egyes tokozások, alkatrész lábaknak kialakított furatok megfelelő méreteit és pozícióit. Miután az összes felszínre került hiba elhárítása megtörtént és a vizsgálatoknak megfelelt a nyomtatott áramkör, megkezdődhetett a beültetés.

5.1.2. A tápegység

A beültetés első lépéseként egy külső cég által beültetésre kerültek a kézzel nehezen kezelhető stabilizátorok és hűtőbordáik, a DSP, az FPGA és a FLASH memória.

Következő lépésként már kézi úton a tápegység maradéka is beültetésre került. Rövidebb ellenőrzés után megtörtént a tápegység feszültség alá helyezése (a bemenetre +/-6 V-ot adva, 100 mA-es áramkorlát mellett). A mesterséges szakadások ekkor jutottak szerephez, tehát a beültetett drága és nehezen cserélhető alkatrészeket védték a tápegység esetleges hibás működése esetén. A működés során a következő paramétereket kellett figyelni:

- **Áramfelvétel.** A bekapcsolás után mért áramfelvétel nagysága körülbelül 8 mA volt. Túl magas, illetve túl alacsony áramértékből rövidzárra, illetve szakadásra lehetett következtetni. A fokozatosan növekvő érték esetén pedig meg kell vizsgálni, hogy a kondenzátorok helyes polaritással vannak-e bekötve.

- **Kimeneti feszültségek.** Mind üresjárásban, mind terhelt kimenet esetén ellenőrizni kellett a stabilizátorok kimeneti feszültségeit. Pár százaléknál nagyobb eltérés nem engedhető meg. Ellenőrizni kellett továbbá a bemeneti feszültségek változtatása esetén a kimeneti feszültségek stabilitását is. Abból, hogyha a kimeneti feszültség az előírt pár százaléknál jobban eltér, arra következtethetünk, hogy a stabilizátorok nem a megfelelő drop feszültséget figyelembevéve kerültek alkalmazásra. A statikus paraméterek mellett ellenőrizni kellett még a stabilizátorok kimeneti feszültségének AC komponenseit is. Bármilyen periodikus jel arra utal, hogy az adott stabilizátor gerjed, tehát a kimenete vagy a bemenete nincs megfelelő kapacitásokkal ellátva.

- **Hőtermelés.** Terhelt kimenetek és huzamosabb működés esetén ellenőrizni kellett, hogy az eszközök hőmérséklete az elfogadható tartományban van-e, tehát nem több mint 60-65 Celsius fok.

5.1.3. A DSP és az FPGA élesztése

A tápegységben található mesterséges szakadások söntölése után, 300 mA-es áramkorlát mellett megtörtént a kártya bekapcsolása. Ekkor a már korábban beépített DSP és az FPGA is feszültség alá került. Mivel a tápáram (230 mA) megfelelt a várt értéknek, megkezdődhetett a processzor, majd az FPGA tesztelése. Az egyszerű Flag8-as lábat vezérlő, végtelen ciklust tartalmazó program J-TAG-en keresztül töltődött le a processzorba. Az FPGA

működőképességének ellenőrzése szintén egy egyszerű, a 9. IO lábat az órajel 16-odával vezérlő program letöltése és mérése által történt. Miután az eszközök a letöltött programoknak megfelelően működtek, megkezdődhetett a kártya többi egységeinek beszerelése, és az egységeket kezelő programok megírása.

5.1.4. Analóg ki-bemeneti egységek

Következő lépésként az első analóg ki-bemeneti egységének beültetésére és bemérésére került sor. Bekapcsoláskor figyelni kellett az áramfelvételt, illetve az áramfelvétel növekedését az előzző állapothoz képest. Ezt minden új egység beültetése után a későbbiekben is figyelni kell, mert az áramfelvétel növekedése nem lehet nagyobb, mint az újonnan beültetett egység fogyasztása. Mivel az egységgel kapcsolatban lévő codec ekkor még nem volt beültetve, a codec által szolgáltatott referenciajelet két, nyitó irányban bekötött, ellenállással előfeszített diódával kellett létrehozni, az egységek helyes működése érdekében. A bemeneti egység kimenetét összekötve a kimeneti egység bemenetével mérhetővé vált az áramkör. A bemenetet függvénygenerátorral gerjesztve, meg kellett vizsgálni a következő kritériumokat:

- Az erősítők munkapontjai a helyes tartományban vannak.
- Az átalakított jelek referencia feszültséggel vannak eltolva.
- Az áramkör megfelelő bemenet esetén nem torzít.
- Az áramkör jelein található zaj mértéke elfogadhatóan kicsi.
- Az alsó töréspontí frekvenciák (-3 dB-es pontok) az előirányzott helyeken vannak.
- DC csatolás melletti helyes működés.

5.1.5. A codec-ek

A következő fontos lépés a master codec felforrasztása és élesztése volt. A műveletet a codec-et vezérlő 16 MHz-es oszcillátor beüzemelésével és a processzor soros portjának ellenőrzésével és a kezelő rutin megírásával kellett kezdeni. Miután a codec megkapta a processzortól a RESET és az SE jeleket a kommunikáció megkezdődött, ami a soros port vezetékein lévő jelek mérésével ellenőrizhető volt:

- **SCLK.** Ezt a jelet a master codec állítja elő, frekvenciája reset után nyolcada a MCLK jelnek. Jelalakja rendkívül fontos, mert jelentősebb torzulása esetén a processzor soros interfésze rossz időpontokban mintavételezi a bemenetét, illetve frissíti a kimenetét.

- **TFS, RFS.** Az egyes szavak kezdetét jelző vezetékeken jelen esetben ugyanannak a jelnek kell lennie. Későbbiekben több codec esetén az RFS, TFS vezetékeken található impulzusok számából és helyzetéből következtethetünk a helyes működésre.

- **Adatvezetékek.** A DSP által kiadott értéket oszcilloszkóppal ellenőrizni kell. Itt sem engedhető meg a jel nagyobb mértékű torzulása.

A vezetékek ellenőrzése után a tesztelés a codec-ek inicializálásával folytatódik, amit a konfigurációs szavak letöltésével lehet elérni. Mivel a refrencia bekapcsolódik az inicializáció során, ezért a REF kimenetre rámérve meggyőződhetünk a helyes működésről. Természetesen a ki-bemenetek tesztelésénél használt diódás referenciafeszültséget létrehozó egység eltávolításáról gondoskodni kell. Tesztelés alatt a codec-ek analóg és digitális echo funkcióinak használata ajánlott, mert ezáltal az egész csatorna mérhetővé válik. Ezeket a funkciókat a konfigurációs szavak egy-egy bitjével lehet bekapcsolni.

Miután a processzor DMA segítségével megfelelően kommunikált az első codec-kel, tovább lehet lépni a többi codec élesztése felé.

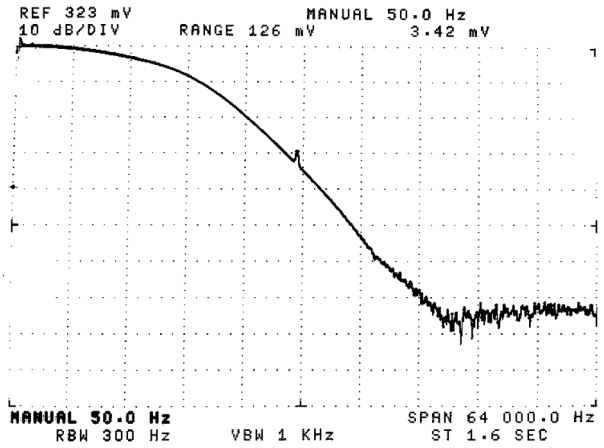
A többi codec felszerelése és élesztése továbbra is egyesével történt, a felmerülő problémák könnyebb elhárítása érdekében. A folyamat során a master codec-ben lévő MCLK osztó osztási arányát egyre kellett állítani, hogy a négy codec által létrehozott nyolc szó elférjen a láncban.

Az utolsó codec beüzemelése után a csatornák működésének ellenőrzése a laborban rendelkezésre álló Hewlet-Packard 3585B típusú spektrumanalizátor segítségével történt. A kártyán ebben az esetben egy egyszerű echo alkalmazás futott, ami a DSP-be beérkezett digitális adatokat módosítás nélkül írta vissza a bemenetnek megfelelő sorszámú kimenetre.

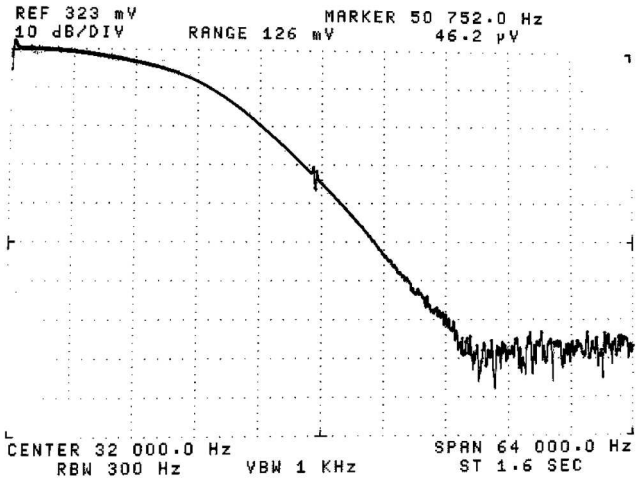
A mérést az összes csatornán elvégeztem, a kapott eredmény mind a nyolc esetben megegyezett. Az 5.1 és az 5.2 ábrákon az egyes és a kettes sorszámú csatorna átvitelének mérési eredményei láthatóak 62.5 kHz-es mintavételi frekvencia mellett.

Ezek a karakterisztikák megfelelnek az AD73322-es codec adatlapján közölt specifikációval. A mérési eredmények vizsgálatakor figyelembe kellett venni azt, hogy a kapott karakterisztikák a bemeneti és kimeneti szűrők karakterisztikáinak az eredői.

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



5.1. ábra. 1. Analóg csatorna átvitele



5.2. ábra. 2. Analóg csatorna átvitele

5.1.6. Soros port (UART) élesztése

A DSP párhuzamos buszára kapcsolódó eszközök közül ez került elsőként felélesztésre. Beültetés után meg kellett győződni az egység órajel-generátorának helyes működéséről. Mivel az oszcillátor (melynek a frekvenciáját egy külső kvarckristály határozza meg) rendkívül érzékeny a terhelésekre, ezért az oszcilloszkóp mérőfejét csak az oszcillátorban található inverter kimenetére lehetett csatlakoztatni. A frekvencia ellenőrzése és az processzor buszvezérlő

jeleinek átütemezését végző modul FPGA-ba való implementálása után megkezdődhetett a periféria tesztelése.

A vezérlőben található scratchpad-nek nevezett regiszter (használata semmiféle kihatással nincs a vezérlő működésére) írásával majd visszaolvasásával ellenőrizni lehetett a processzor és a periféria stabil, hibamentes kapcsolatát.

Az RS-232-es illesztő mudul felhelyezésével a kártya soros porti kapcsolata teljessé vált.

5.1.7. Digitális IO élesztése

A sorban következő, szintén az FPGA-hoz tartozó periféria a digitális IO volt. A periféria FPGA-n való implementálása, és a külső alkatrészek beültetése után meg kellett viszgálni a processzor és a periféria kapcsolatát. Ebben az esetben is egy, az FPGA-ban található regiszter (amit a későbbiekben eltávolításra került) írása és visszaolvasása segítségével győződhettünk meg a két egység és az FPGA-n definiált buszkezelő hardver helyes működéséről.

A periféria kimeneti részének helyes működéséről egy számláló kiiratásával, és az egyes bitjeinek oszcilloszkóppal való mérésével lehetett meggyőződni. A periféria hibás működésére utaló jelek:

- Az egyes kimenetek egymáshoz képest elcsúsznak, tehát a négyszög impulzusok felfutó, illetve lefutó élei nem azonos időpontra esnek.
- Impulzusok kimaradnak.
- Az impulzusok hosszának aránya nem kettő hatvány.

5.1.8. Az SDRAM élesztése

Utolsó perifériaként az SDRAM élesztésére került sor. Az általa felhasznált vezetékek és tápvonalak ellenőrzése után a foglalat a helyére került.

A vezérlő tesztelése az előbbi módon egy benne található regiszter írásával és visszaolvasásával történt. Ezek után az üres foglalat érintkezőin ellenőrizni kellett a vezérlő jeleket, illetve az egyes jelek clock-hoz képesti helyzetét. Pár ns-nál nagyobb késés hibás működést eredményezne.

A memória beépítése után a frissítési ciklusok gyakoriságának csökkentésével be kellett állítani egy elfogadható fogyasztást.

Ezek után az egész rendelkezésre álló memóriaterület ellenőrzése következett a már korábban is alkalmazott írás és visszaolvasás segítségével. Ügyelni kellett azonban arra, hogy a memóriában az egyes 32 bites rekeszeket különböző értékekkel töltsük fel.

5.2. Az FPGA-n definiált hardver

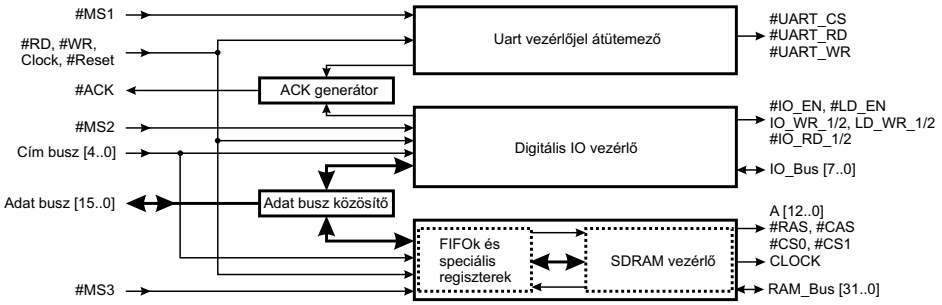
A hardver szintézisnek igen sok módja van, lehetőség van grafikus eszközök, illetve leírónyelvek használatára.

Grafikus rendszer használata esetén a megvalósítandó hardvert logikai elemeket tartalmazó kapcsolási rajzként definiáljuk, míg leíró nyelvek alkalmazásakor logikai függvények, feltételek, ciklusok segítségével "írjuk" le a hardver kívánt működését.

A fejlesztés során a Xilinx által létrehozott ingyenesen letölthető ISE 6.1 rendszert használtam, amely mind a grafikus, mind a szöveges bevitelt, illetve ezek keverékét is támogatta. A létrehozott forráskódot letöltés előtt szimuláltam is a ModelSim XE II 5.7 szoftver segítségével. Így lehetőség nyílt a rejtett hibák kiküszöbölésére, és a kód működésének extrém körülmények között való kipróbálására.

A lefordított kódot J-TAG kábel segítségével töltöttem föl az FPGA-ra. Ezek után oszcilloszkóppal lehetőség nyílt a hardver egységek valós körülmények közötti tesztelésére.

A létrehozott hardver vázlata az 5.3 ábrán található.



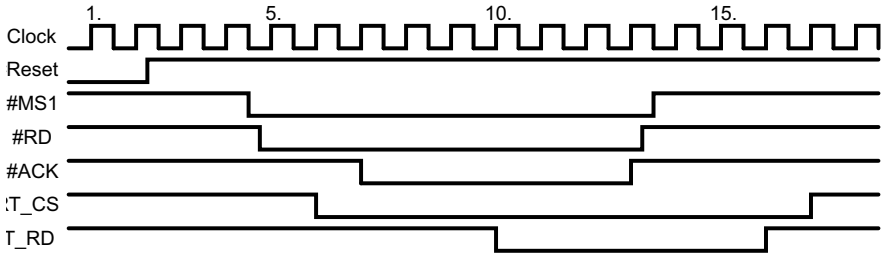
5.3. ábra. Az FPGA-n definiált hardver vázlata

Az ábrán jól látható, hogy a hardver öt nagyobb egységre osztható, amelyeknek a működése a következő alfejezetekben kerülnek ismertetésre.

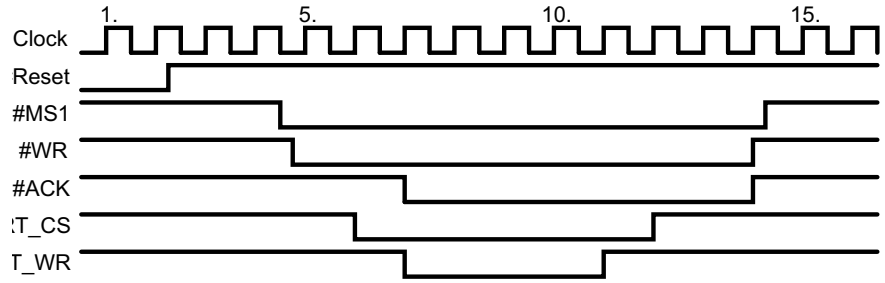
5.2.1. Az UART kezelő

Az egység feladata a DSP buszvezérlő jeleinek megfelelő átütemezése olyan módon, hogy az az UART vezérlő számára feldolgozható legyen. Az alábbi ábrákon az írás és olvasás ciklusa látható:

Az ábrákon látható, hogy az egység az ACK jel segítségével megnyújtja a DSP busz ciklusát. Erre azért van szükség, mert a nyomtatott áramkörön a DSP és az UART vezérlő igen nagy távolságban helyezkedik el egymástól,



5.4. ábra. Az átütemezett jelek olvasás esetén

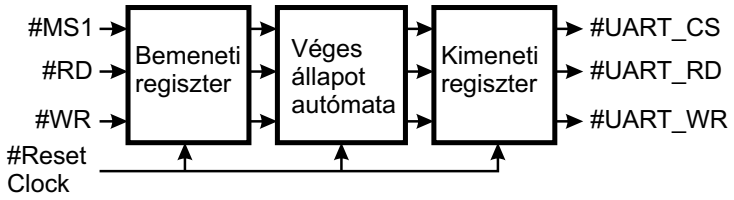


5.5. ábra. Az átütemezett jelek írás esetén

és rövidebb busz ciklus esetén elképzelhető, hogy a vezérlő hibás adatot ad vagy vesz.

Az egység másik feladata, hogy a DSP buszvezérlő jeleit megfelelő módon eltolja és rövidítse.

Az 5.6 ábra az egység felépítését szemlélteti.



5.6. ábra. Az UART kezelő egység felépítése

Látható, hogy az egység szinkron működésű, és induláskor a reset lábra adott L szintű impulzus állítja alaphelyzetbe a ki-bemeneti regisztereket, és az állapot automata állapotváltozóját.

Mivel a DSP és az FPGA nem azonos órajelről működik, ezért az egységet működtető órajel felfutó élének és a DSP kimenetinek változásának idő-

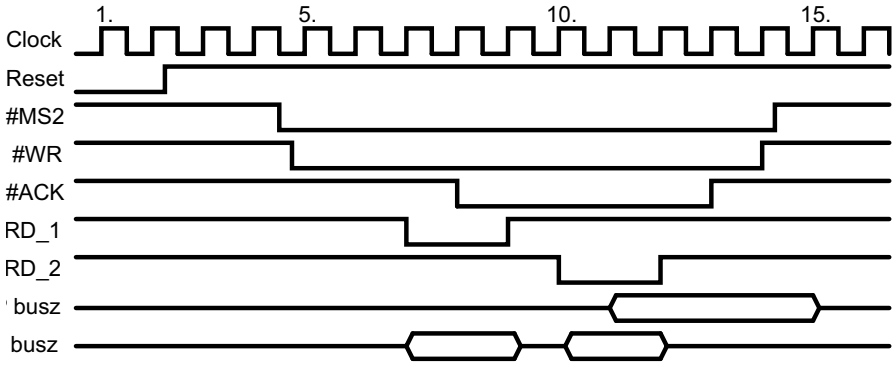
különbsége folyamatosan változik. Ha ez az időkülönbség az állapotautomata kombinációs hálózatának jelterjedési ideje alá esne, akkor az automata fals állapotba kerülne. Ezt elkerülendő az egység bemeneteit egy szintén a clock jelre működő regiszterrel mintavételezzük, így a bemenet változása és az órajel felfutó éle mindig azonos távolságban lesz egymástól.

Magát az ütemezést egy egyszerű állapotautomata végzi, amely írás vagy olvasás alatt kikerül a stabil állapotából, és aktuális állapotának függvényében állítja a kimeneti regiszter értékeit.

5.2.2. Az IO vezérlő

Ez az egység felelős a digitális IO és a kivezérlésjelző LED-ek kezeléséért. Mint már korábban is említettem, egyszerűsítési okokból a kimeneti latcheket és a bemeneti buffereket az egység egy nyolc bites busz (IO_BUS) segítségével éri el. Tehát az egység feladata tulajdonképpen a DSP párhuzamos buszát illeszteni az IO_BUS-hoz.

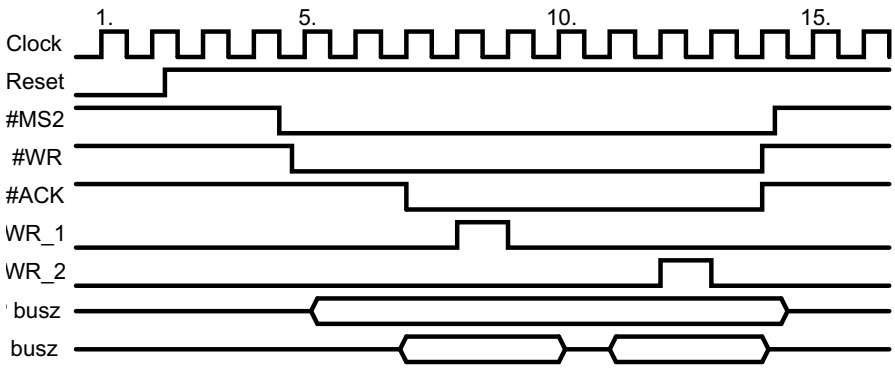
Mivel a DSP adatbusza 16 bites , az IO busz pedig csak 8 bites, ezért minden írási és olvasási művelet két fázisban zajlik le. Ez azt jelenti, hogy ha a DSP ír, akkor az a 16 bites érték két 8 bites értékként íródik ki a kijelölt latch-ekbe. Olvasáskor pedig a két buffer kiolvasása egymás után történik, és mint 16 bites érték kerül továbbításra a DSP felé. Az olvasási ciklust az 5.7 ábra, az írási ciklust pedig az 5.8 ábra szemlélteti.



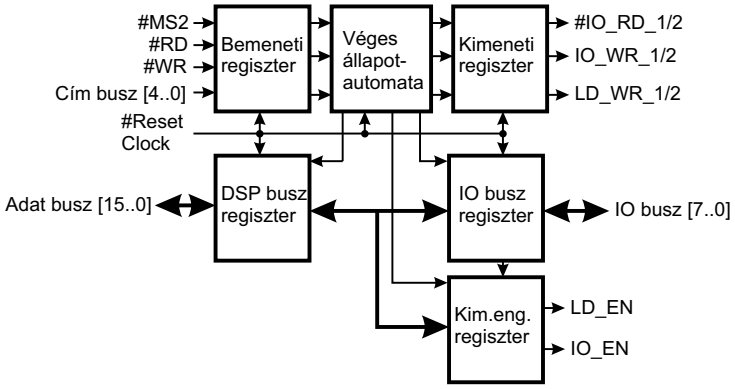
5.7. ábra. Olvasási ciklus

Látható, hogy az egység is rendelkezik ACK kimenettel, melynek segítségével megnyújtja a DSP busz ciklusát. Erre jelen esetben azért van szükség, mert a két fázisból álló írás vagy olvasás művelet nem hajtható végre rövidebb idő alatt.

Az periféria belső felépítése az 5.9 ábrán látható.



5.8. ábra. Írási ciklus



5.9. ábra. Az IO vezérlő egység felépítése

Ez az egység is szinkron működésű. Az UART kezelőhöz hasonlóan ennek az perifériának is egy mintavételezett bemenettel rendelkező véges állapotautomata vezérli a működését.

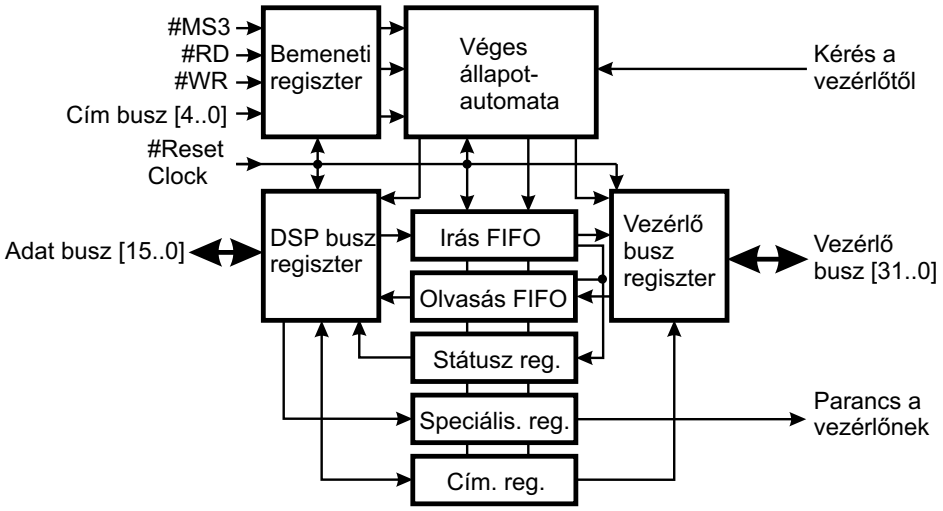
A két busz illesztését két kétirányú regiszter valósítja meg, melyeknek az irányát az állapotautomata vezérli. A kimeneti regiszter jelei választják ki az aktuális művelet alapján a használni kívánt latch-et vagy buffert.

A DSP felől írható kimenet engedélyező regiszter kimeneti bitjei a kimeneteket és LED-eket meghajtó latchek #OE (Output Enable) lábaira vannak kötve. Reset hatására a regiszter értékei H szintre állnak be, tiltva a kimeneteket. A latch-ek törlése után a DSP-n futó programban található inicializáló szakasz állítja L szintre a regiszter bitjeit, ami a korábban tiltott kimeneteket engedélyezi.

5.2.3. Az SDRAM vezérlő

Az egység feladata a hozzá kapcsolódó memória és a DSP közötti közvetítő szerep, valamint a memória frissítése. Mint már a korábbi blokkvázlaton is látszott, ez az egység két nagyobb részre bontható: FIFO-k és speciális regiszterek, vezérlő automata.

A FIFO-kat és speciális regisztereket tartalmazó részegység vázlata az alábbi 5.10 ábrán látható.



5.10. ábra. FIFO-k és regiszterek

Ez a részegység a vezérlő és a DSP közötti kapcsolatot valósítja meg. A folyamat vezérléséről jelen esetben is egy állapotautomata gondoskodik.

A részegység használata írás esetén a következőképpen történik:

- **Írási cím beállítása a címregiszterben.** Ez a regiszter reset után törlődik, és minden írási ciklus után automatikusan nő az értéke, ezért kezelése átlagos használat esetén nem szükséges.
- **Írás FIFO feltöltése.** A kiírandó 8 darab 16 bites adatot betöltése az írás FIFO-ba
- **Írási parancs.** A speciális regiszter irás bitjének H szintre állítása után megkezdődik az írási folyamat.
- **Státusz figyelés.** Az írási folyamat végét a státusz regiszter értéke mutatja.

A részegység használata olvasás esetén pedig a következőképpen történik:

- **Olvasási cím beállítása a címregiszterben.** A regiszter az írási regiszterhez hasonlóan működik.
- **Olvasási parancs.** A speciális regiszter olvasás bitjének H szintre állítása után megkezdődik az olvasási folyamat.
- **Státusz figyelés.** Az olvasási folyamat végét a státusz regiszter értéke mutatja.
- **Olvasás FIFO kiürítése.** A beolvasott 8 darab 16 bites érték az olvasás FIFO-ból egyenként, sorban kinyerhető.

A státusz regiszter a folyamatok állapotát jelző biteken kívül a FIFO-k állapotát is tartalmazza. Ez a regiszter - mint az az ábrán is látszik - csak olvasható, ezzel ellentétben pedig a parancsokat tartalmazó speciális regiszter csak írható.

Az aktuális írási és olvasási pozíciókat a címregiszter tárolja. Mint azt már korábban is említettem, mindkét cím a hozzájuk tartozó művelet esetén inkrementálódik, túlcsordulás esetén pedik nullázódik. Ezáltal megtakarítható a DSP oldaláról két írási ciklus, melynek során az aktuális cím beállítása történik.

Az olvasási és írási FIFO-k a memóriakezelés gyorsítására szolgálnak, így egyszerre nyolc adat kezelése történik meg egy helyett. Ezáltal a címzést és a lap lezárását egyszer kell elvégezni, nem mindegyik adat esetében, ami jelentős idő megtakarítást jelent.

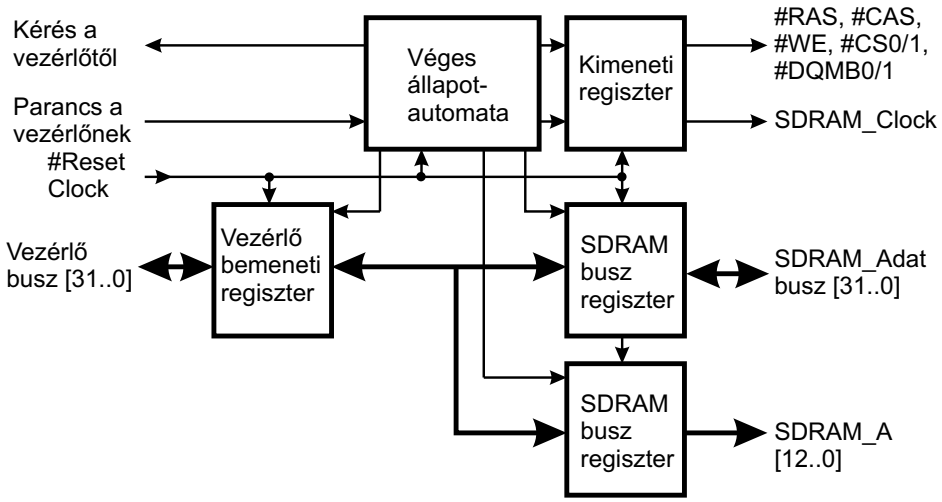
A memóriavezérlő vázlata az 5.11 ábrán látható.

Ez a részegység végzi a memória tényleges kezelését az előbbi részegységtől kapott paraméterek alapján, tehát vezérli az írási és olvasási műveleteket, illetve frissíti a memóriát.

Az adatvesztés elkerülése végett a memóriának frissítési parancsot kell adni időközönként, ami nem lehet ritkábban, mint $7.8\mu sec$. Ezeket a parancsokat mindenképpen be kell fésülni az írási és olvasási folyamatok végrehajtása közé.

A vezérlő #RAS, #CAS, #WE jelek segítségével ad parancsot a #CS0, #CS1 jelek által kijelölt memóriának. Az SDRAM_A jelű busz a címzéskor az SDRAM_Adat busz pedig adatmozgatás esetén jut szerephez.

Az állapotautomata reset után inicializálja a memóriát, majd pedig beáll egy végtelen ciklusba, melynek során minden 64-ik ciklus esetén kiadja a memória frissítés parancsot. Irás vagy olvasási parancs esetén először a címnek megfelelő lapot és sort állítja be, majd az oszlop címzése közben kiadja



5.11. ábra. SDRAM vezérlő

az aktuális folyamatot reprezentáló parancsot. Ezek után az adatok írása vagy olvasása történik meg. E folyamat a megnyitott lap lezárásával és egy memóriafrissítés parancsal végződik.

5.2.4. ACK generátor

Az egység feladata, hogy a Digitális IO és az UART kezelő ACK kéréseit közvetítse a DSP felé. A DSP ACK bemenetét az ACK generátor open-collector-os kimenet segítségével hajtja meg.

Az egység megkapja bemenetként az egyes hozzá csatlakozó egységektől az általuk kívánt ACK jeleket, illetve a hozzátartozó ACK engedélyező jelet is. Ha az ACK jel L szintű, és a hozzátartozó ACK engedélyező jel H szintű, akkor a korábban high-Z ACK kimeneti láb L szintűvé változik. A feltételek megszűnése után pedig visszavált high-Z állapotba.

5.2.5. Adat busz közösítő

Mivel az FPGA-n belül két egységnek is kétirányú hozzáférése van a DSP adatbuszához, így erre az egységre szükség van.

Bemenetként megkapja a két egységtől a hozzájuk tartozó adatbuszok kívánt irányát, és ezek alapján mozgatja az adatokat az egységek és a DSP között.

6. fejezet

Szoftver tervezés

6.1. A DSP-n futó szoftver

Az előzző fejezetben ismertetett hardver egységek és az FPGA-n definiált további perifériák a DSP-n implementált szoftver réteggel és a PC-n futó alkalmazás specifikus kezelői felülettel együtt alkotják a sokcsatornás jelfeldolgozó rendszert. Mivel az egész rendszer vezérlését a jelfeldolgozó processzor végzi (aminek a működésébe a HOST-on futó program parancsok vagy lekérdezések formájában bele tud avatkozni), így a rendszer működtetését végző szoftver az alkalmazások nagy százalékában a DSP-n futó programot jelenti.

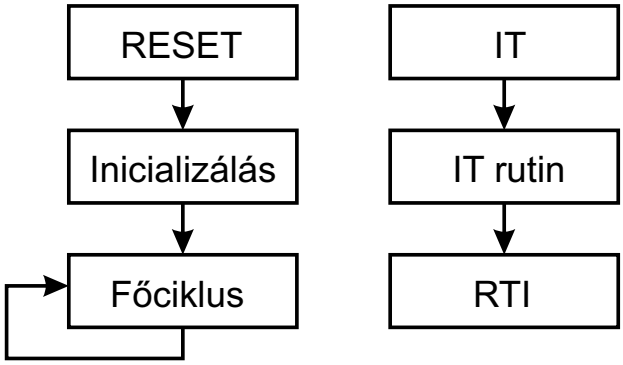
Az harmadik fejezetben látható volt, hogy a szoftver réteg szerepe túlmutat a hardver egységek kezelésénél. A szoftver réteg nem egy konkrét jelfeldolgozó algoritmust megvalósító program, hanem egy általánosan használható szoftver környezet, mely alapját képezi a megvalósítandó jelfeldolgozó alkalmazásoknak. Ez biztosítja azt, hogy a rendszer valóban 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.

A dolgozat hetedik fejezetében egy konkrét alkalmazás bemutatására kerül majd sor. Ez amellett, hogy bizonyítja a rendszer helyes működését, egy példát is mutat arra, hogy milyen módon történik egy konkrét jelfeldolgozó alkalmazás implementálása a sokcsatornás jelfeldolgozó rendszerben.

6.1.1. DSP program általános struktúrája

A sokcsatornás rendszer DSP-n futó programjának konkrét ismertetése előtt, az alábbiakban azon programtervezési irányelveket ismertetem, amelyek a DSP programok esetében szokásosnak mondhatók, és amelyeket a programfejlesztés során szem előtt tartottam.

Az online működésű jelfeldolgozó programok szokásos struktúrája a 6.1 ábrán látható.



6.1. ábra. A DSP programok szokásos struktúrája

Reset után az inicializáló rész fut le, melynek során a DSP és a hozzá kapcsolódó perifériák különböző működést vezérlő regiszterei kapják meg a kezdeti értékeiket, biztosítva a későbbi helyes működést. Az inicializálási folyamat alatt a megszakítások tiltva vannak. Az inicializálás legvégső fázisa engedélyezi a megszakításokat, majd átadja a vezérlést a főciklusnak. Ez tulajdonképpen nem más, mint egy semleges nop (no operation) utasítást tartalmazó végtelen ciklus, mely nem végez semmiféle műveletet.

Ilyenkor a program tulajdonképpen megszakításkérésre várakozik. Ha bejön egy megszakításkérés, akkor az kiszolgálásra kerül, azaz lefut a hozzá tartozó rutin. A megszakításkérés csak akkor kerül kiszolgálásra, ha a processzor éppen nem szolgál ki más megszakításkérést, mert ebben az esetben várakoznia kell az előző kiszolgálásának befejezéséhez.

Az AD átalakítók akkor kérnek megszakítást, ha a beolvasott adat(ok) már rendelkezésre áll(nak), tehát megkezdődhet az adat(ok) feldolgozása. A jelfeldolgozó algoritmusokat tehát 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 az új bemeneti adatok megérkeztek.

Az előbbi programstruktúra adatvesztésmentes működésének feltétele az, hogy a jelfeldolgozó rutin futása befejeződjön azelőtt, hogy újabb megszakítás kérés érkezzen az AD átalakító felől. Tehát két mintavételi időpont közötti idő áll rendelkezésre a megszakítási rutin (jelfeldolgozó algoritmus) egy adott bemeneti mintával való teljes lefutásához. Ha az időtartam elegendő a rutin

lefutásához, akkor a jelfeldolgozási algoritmus on-line működésű lesz.

Mivel az AD olvasása és a DA kimenetének frissítése azonos időpontban történik, így a jelfeldolgozó rutin által kiszámított kimeneti adatok egy ütemnyi késleltetéssel jelennek meg a rendszer kimenetén.

6.1.2. Decimálás

Bizonyos jelfeldolgozási algoritmusok számára az audio alkalamzásoknál szokásos több tíz kHz-es mintavételi frekvencia már túl magas. Sok esetben néhány száz vagy csak néhány 10 hertzes mintavételi frekvencia használatára van igény.

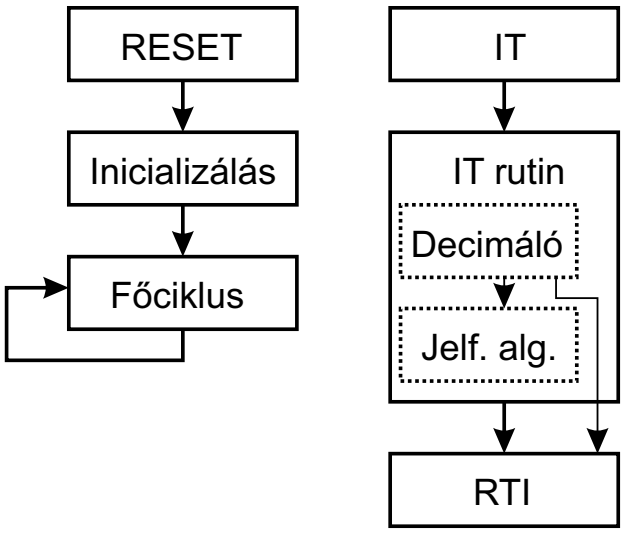
Ezért, ha a rendszerben található AD és DA átalakítók nem támogatják a szükséges alacsony mintavételi frekvenciát ,decimálást szokás alkalmazni. Az N-szeres decimálás azt jelenti, hogy az átalakítók felől érkező adatok mintavételi frekvenciáját N-ed részére csökkentjük, így a feldolgozandó adatok száma arányosan lecsökken. Ez tulajdonképpen úgy történik, hogy a jelfeldolgozó rutin csak minden n-edik bejövő minta esetén fut le. Ebből pedig az következik, hogy a DA számára is csak minden N-edik ütemben számítunk ki új adatot. A fennmaradó N-1 ütemben az átalakító kimenetén az előzző adat van, tehát nulladrendű tartó valósul meg. Elképzelhető olyan eset is, ahol nulladrendű tartó helyett interpoláló szűrőt alkalmazunk.

Ha decimálást alkalmazunk, akkor azt kétféle programstruktúrában tehetjük:

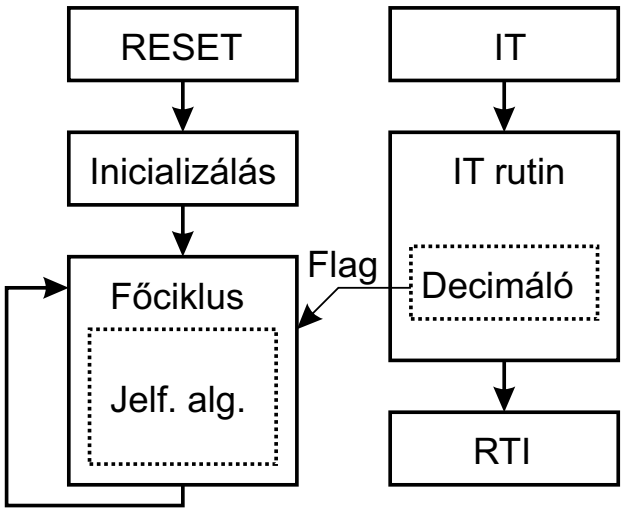
- Az IT rutinban helyezzük el a decimáló, és a jelfeldolgozó algoritmust is. (6.2 ábra)
- Az IT rutinban csak a decimálást végezzük, és maga a jelfeldolgozó algoritmus a főciklusban található. (6.3 ábra)

A 6.2 ábrán látható struktúra tulajdonképpen az általános struktúra decimálálással kibővített változata. Ebben az esetben a decimáló rutin minden egyes minatvétel után lefut, de csak minden N-edik mintavétel után generál kimenetet. N-1 ütemben a decimáló rutin lefutása után befejeződik az IT kiszolgálása, de az N-edik ütemben maga a jelfeldolgozó algoritmus is lefut a decimáló által előállított mintát feldolgozva.

Abban az esetben, ha a decimáló rutin és a jelfeldolgozó algoritmus nem tud egyszerre lefutni két mintavételi időpont között, alkalmazzuk a 6.3 ábrán látható struktúrát. Ekkor a jelfeldolgozó algoritmus a főciklusban van elhelyezve, és csak akkor fut le, ha a decimáló rutin egy Flag változón keresztül jelzi egy új minta keletkezését. Az AD megszakítását kiszolgáló rutin minden egyes beérkezett adat után lefuttatja a decimáló rutint, és egy számláló



6.2. ábra. DSP programstruktúra IT-ben elhelyekedő decimáló és jelfeldolgozó algoritmus esetén



6.3. ábra. DSP programstruktúra főciklusban elhelyezkedő jelfeldolgozó algoritmus esetén

funkcióval nyilvántartja az ütemet. Abban az esetben, ha az aktuális ütem éppen az N-edik, a decimáló rutin kimenetét eltárolja egy változóba, majd beállítja a korábban említett Flag változót. Ebből következik, hogy a jelfeldolgozó algoritmus nincsen behatárolva az eredeti mintavételi időközönként

érkező megszakítások által.

Nem került sor decimáló szűrő megvalósítására a rendszerben. Mivel ezeknek a szűrőknek rendkívül szigorú követelményeknek kell megfelelniük, ezért viszonylag magas fokszámúak. Használatuk jelentősen csökkenti a felhasználható futási időt, azonkívül a decimálás foka és a szűrő paraméterei az adott alkalmazástól függnek, ezért ezt a feladatot a felhasználónak kell implementálnia a már bemutatott struktúrák egyikének alkalmazásával. Továbbá elképzelhető olyan eset is, ahol a feldolgozni kívánt jel erősen sávkorlátozott, és emiatt a szűrő alkalmazására nincs szükség.

6.1.3. Az offline jelfeldolgozó program struktúrája

A fentiekben bemutatásra került struktúrák mindegyike az online működést tette lehetővé, tehát minden egyes mintavételi időpontban a jelfeldolgozó algoritmusok lefutnak, és feladattól függően kimenetet is generálhatnak.

Az offline működés lényege az, hogy a rendszer valamilyen felhasználó által meghatározott feltétel bekövetkezésére vár. Ha ez bekövetkezett, akkor a bejövő mintákat időrendi sorrendben eltárolja. A felvételt a memória megtelése vagy egy szintén a felhasználó által meghatározott feltétel bekövetkezése állítja le. Ezek után következik csak a jelfeldolgozó algoritmus futtatása, amely akár az egész regisztrátumon képes műveleteket végrehajtani.

Abban az esetben használjuk ezt a működést, ha a szükséges mintavételi frekvencia mellett a jelfeldolgozó algoritmus nem tud lefutni két mintavételi időpont között, és nincs szükség azonnali kimeneti minta(-ák) generálására.

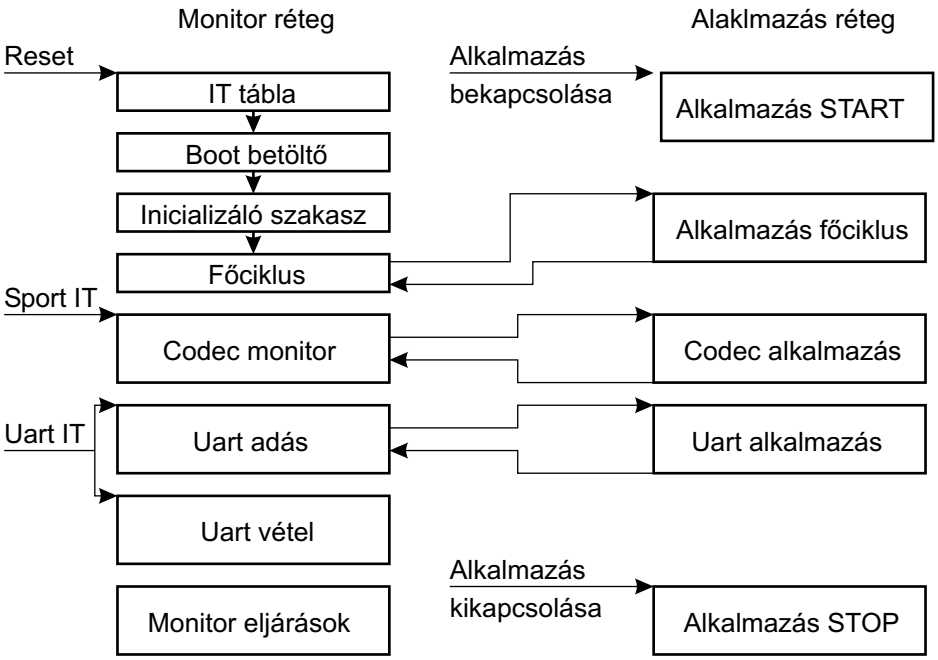
Használata csak akkor eredményes, ha a memória kapacitása elegendő ahhoz, hogy a feldolgozni kívánt regisztrátum elférjen benne.

A korábban ismertetett struktúrák bármelyikét át lehet alakítani erre a működési formára, úgy, hogy a jelfeldolgozó algoritmus helyére a felvételt indító, leállító és a mintákat eltároló rutin kerül. A jelfeldolgozó algoritmus pedig a főciklusból elérhető egy Flag bit vizsgálata után, ami új regisztrátum létrejöttét jelzi.

6.1.4. A megvalósított program struktúrája

A fentiekben leírt szoftvertervezési megfontolásokat figyelembe véve a DSP-n futó programot úgy alakítottam ki, hogy az összes ismertett strukúra megvalósitható legyen a felhasználásával. A program struktúráját a 6.4 ábrán szemléltetem.

Az ábrán látható, hogy a program alapvetően két nagyobb egységre: Monitorra és Alkalmazásra tagalódik. A feladat megoldása során a monitor ké-



6.4. ábra. A kialakitott program struktúrája

pezi a szoftver réteg rezidens, míg az alkalmazói réteg a szoftver réteg feladat specifikus részét. A monitor réteg részei:

IT tábla

Az interrupt vectorokat és a kiszolgáló rutinokra való hivatkozásokat tartalmazza. A monitor rétegben fel nem használt interrupt vectorok az alkalmazói réteg meghatározott helyeire mutatnak a későbbi felhasználhatóság céljából. Reset hatására a processzor a reset vectortól kezdi meg az egész program futtatását, majd innen rögtön átkerül a vezérlés a boot betöltő rutinra.

Boot betöltő

A reset után rögtön bekövetkező bootstrap-nek nevezett folyamat automatikusan beolvassa a hardveresen kijelölt forrásból (jelen esetben a FLASH memória) a processzor programjának első 256 darab utasítását, csak ezek után kerül majd át a vezérlés a reset vectorra. Az interrupt tábla ennek a 256 utasításnak az első felén, míg a boot betöltő a másik felén foglal helyet. A boot betöltő feladata a processzor monitor programjának fennmaradó utasításait és adatait betölteni. Ezeken felül beolvassa még az alkalmazói pro-

gramot és a hozzátartozó adatokat is. Lefutása után továbbadja a vezérlést az inicializáló szakasznak.

Inicializáló szakasz

A szakasz feladata beállítani a processzorban található és a hozzá kapcsolódó perifériákat. Első lépésként előállítja a codec-eket és az FPGA-t alaphelyzetbe állító jeleket (aktív L szintű impulzus) a Flag1 és Flag10-es kimenetek segítségével. Majd beállitja az egyes külső memóriablokkokhoz tartozó wait state generátorokat, amelyek biztosítják azt, hogy a processzor a megfelelő módon érje el az adott blokkokban található perifériákat. Ezek után beállitja az UART vezérlő megfelelő regisztereit. Többek között az órajel osztót, ami az átvitel sebességét befolyásolja, az interrupt regisztert, ami meghatározza, hogy milyen események hatására kell az INT lábon jelet generálni. Továbbá törli (nullázza) a digitális IO kimeneti regisztereit, valamit a kivezérlés jelző LED-ek állapotait tartalamzó regisztereket. Az elöbbi regiszterek kimeneteinek engedélyezése után átadja a vezérlést a főciklusnak.

Főciklus

Az első futáskor ellenőrzi a processzor a Flag3 bemenetére adott jel állapotát. Ha ez a jel L szintő, akkor automatikusan elindítja az alkalmazást. Ellenkező esetben belép a végtelen ciklusba.

Az alkalmazás elindítása esetén a monitor főciklusának magjában elhelyezett nop utasítás kicserélődik az alkalmazás főciklusára ugró utasításra, így a főciklus is átkerül az alkalmazói rétegbe, ahol akár jelfeldolgozó algoritmust is implementálhatunk benne.

Az előbbi folyamat ellentettje játszódik le az alkalmazás kikapcsolása esetén, tehát a monitor főciklusában található utasítás újra nop-ra cserélődik, és az alkalmazás főciklusába kerül egy ugró utasítás, ezáltal visszakerül a vezérlés a monitor főciklusához.

Codec interrupt

Minden egyes mintavételi időpontban lefut. Kiolvassa az új mintákat a soros port bemeneti bufferéből, és eltárolja azokat a megfelelő regiszterekben úgy, hogy az alkalmazói program által is elérhetőek legyenek. Végrehajtja a kivezérlés vizsgálatot, és az eredménynek megfelelően állítja a LED-eket. Majd beolvassa a digitális bemenetek állapotát, és átadja a vezérlést az alkalmazói programnak. Az alkalmazás lefutása után visszakapja a vezérlést, frissíti a digitális kimeneteket, és feltölti a soros port kimeneti bufferét az új mintákkal. Mindezek után visszaadja a vezérlést a főciklusnak.

UART adás interrupt

Ez a megszakítás akkor aktiválódik, ha az UART vezérlő kimeneti buffere üressé válik. Megvizsgálja, hogyy van e még kiküldendő adat, és ha talál ilyet akkor továbbítja azt az UART vezérlőnek, ellekező esetben pedig tiltja az adást.

UART vétel interrupt

Minden egyes UART által vett adat után lefut. Megvizsgálja az adatot a későbbiekben ismertetésre kerülő soros protokol szempontjából. Ha összeállt egy üzenet, akkor megvizsgálja az érvényességét, majd végrehajtja az üzenet típusának megfelelő műveletet.

Monitor eljárások

Ez a rész tulajdonképpen olyan eljárásokat tartalmaz, amiket mind a monitor, mind az alkalmazás meghívhat. Ezek a fejlesztés során létrehozott és tesztelt eljárások a kártyán található hardver egységek kezelését, és a processzoron futó program fontosabb műveletek végrehajtását végzik. A fontosabb eljárások és feladatuk a következők:

- **Codec-ek inicializálás.** Az alkalmazás igényeinek megfelelő mintavételi frekvencia, ki-bemeneti erősítés beállítása a codec-eken. A rutin először engedélyezi a soros port-ot és a vele kapcsolatban lévő DMA-t, majd a RESET és az SE jeleket H szintre emelve megkezdődik az inicializáló folyamat, aminek a lefutása után létrejön az interrupt-os vezérlési struktúra.

- **Codec-ek leállítás.** A codec-ek, a DMA és a soros port megfelelő módon való kikapcsolása.

- **FLASH memóriakezelés.** A FLASH írására, törlésére, lezárására szolgáló szekvenciák megfelelő időzítéssel való lefuttatása.

- **SDRAM memóriakezelés.** Az adott belső regiszterek tartalmának kimásolása az írás FIFO-ba vagy az olvasás FIFO tartalmának bemásolása. Az írás, olvasás folyamatának megindítása, a címregiszterek módosítása és az adott folyamat végének detektálása.

- **UART kezelés.** Az aszinkron soros vezérlő kezelére, inicializálása és a megfelelő adatcsere biztosítása.

- **Digitális IO kezelés.** A kimenetek frissítése és a bemenetek aktuális állapotának jelzése változókon keresztül.
- **Alkalmazás kontextusának betöltése, mentése.** A monitor és az alkalmazás által közösen használt cirkuláris bufferek beállításainak mentése és betöltése attól függően, hogy éppen monitorból alkalmazásba vagy az ellenkező irányba adódik a vezérlés.

Az alkalmazás réteg részei pedig a következők:

Alkalmazás start.

Az alkalmazás bekapcsolásakor egyszer lefut. Ide kerülnek az alkalmazás kezdeti feltételeit (cirkuláris buffereket, változó értékeket, SDRAM címeket, codec-eket ki-bemeneteket) beállító utasítások.

Alkalmazás stop.

Az alkalmazás kikapcsolásakor fut le. Itt kell gondoskodni a rendszer megfelelő feltételek melletti leállításáról, a létrejött adatok mentéséről.

Alkalmazás főciklus.

A különböző ismertetett jelfeldolgozási struktúráknak megfelelő részek kerülnek ide.

Codec alkalmazás.

Minden egyes mintavételi időpontban megkapja a vezérlést. A jelfeldolgozó algoritmusok nagy része itt implementálódik (kivéve a decimálásos struktúra egyik fajtájánál).

UART alkalmazás.

Az alkalmazásnak szóló üzenet vétele esetén kapja meg a vezérlést. Feldolgozza a frissen bejött üzenetet az összetevői alapján (adatmező, küldő, üzenet kód).

Az alkalmazás implementálása tulajdonképpen a fenti részek megfelelő kitöltését jelenti.

Bináris érték	Hex-ascii karakter
0000	0
1001	9
1010	A
1111	F

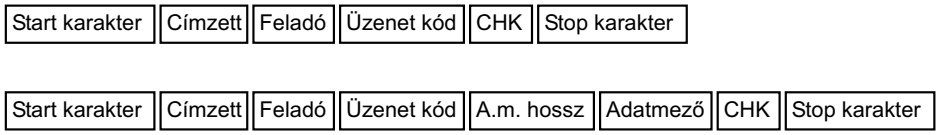
6.1.5. A megvalósított soros kommunikációs protokol

A kommunikációs protokolt a specifikációban leírt követelmények alapján kellett megvalósítani. A megvalósított protokol master-slave kapcsolatban lévő egységek közötti adatcserére szolgál, tehát a kommunikációt mindig a master kezdeményezi. Az adatcsere üzenetek formájában nyílvánul meg, amelyek közül alapvetően két fajtát különböztetünk meg:

- **Rövid üzenet.** Parancs vagy jelzés.
- **Hosszú üzenet.** Adatmozgató, illetve paraméterekkel rendelkező parancsok.

A protokol kialakítása két fontos szempont figyelembevételével történt. Az első az, hogy a kommunikáció aszinkron volta miatt bármely időpillanatban megindulhat az adatfolyam, ezért gondoskodni kell az egységek szinkronizálásáról. A második pedig, hogy feltételezhető kismértékű adatvesztés (főleg RS-485 alkalmazása esetén), ami az üzenet sérüléséhez vezethet, ezért támogatni kell az üzenet épségének ellenőrizhetőségét.

Az üzenetek felépítése az alábbi a 6.5. ábrán látható.



6.5. ábra. Üzenetek felépítése

A szinkronizálást az üzenet elején elhelyezkedő start karakter jelzi (*) és az üzenet végén elhelyezkedő stop karakter (;) segítségével lehet elérni.

A két karakter között helyezkedik el az üzenet érdemi része, melyben az adatok hex-ascii formátumban reprezentálódnak. Ez azt jelenti, hogy minden 4 bit-nyi adatnak megfelel egy ascii karakter, a bináris hexadecimális átalakításnak megfelelően. A megfeleltetést a 6.1.5 táblázat mutatja:

Az üzenet tartalmazhat 8 bites integert (2 hex-ascii karakter), 16 bites integert (4 hex-ascii karakter), és 32 bites integert valamint 32 bites lebegőpontos adatot (mindkettőnek 8 hex-ascii karakter felel meg).

Az üzenet felépítéséből látszik, hogy a start karakter után a címzett és a küldő kódja található. Mindkét kód integer típusú. Ezek szolgálnak az üzenet forrásának és céljának azonosítására. A protokol szerint a 00h kódú egység mindig a HOST számítógép. A kommunikáció többi szereplője a 01h és az FEh közötti címeken osztozhat. Az FFh cím pedig broadcast üzenetek küldése esetén kerül felhasználásra, melynek során az üzenetet minden szereplő a sajátjának tekinti és dolgozza fel, de nem válaszol rá.

A következő egység az üzenetkód. Ez az üzenet azonosítására szolgáló kód. Az üzenetkód egyben meghatározza azt is, hogy rövid vagy hosszú üzenetről van szó.

Az üzenetet pedig a stop karakter előtt az ellenőrző összeg (checksumm) található. Azt a 16 bites értéket az üzenet start karakter és az ellenőrző összeg közötti részéből képezzük a hex-ascii karakterek összeadásával. Az üzenet épségének ellenőrzését vétel után az üzenetben található és a számolt ellenőrző összegek összehasonlításával végzi a vevőegység.

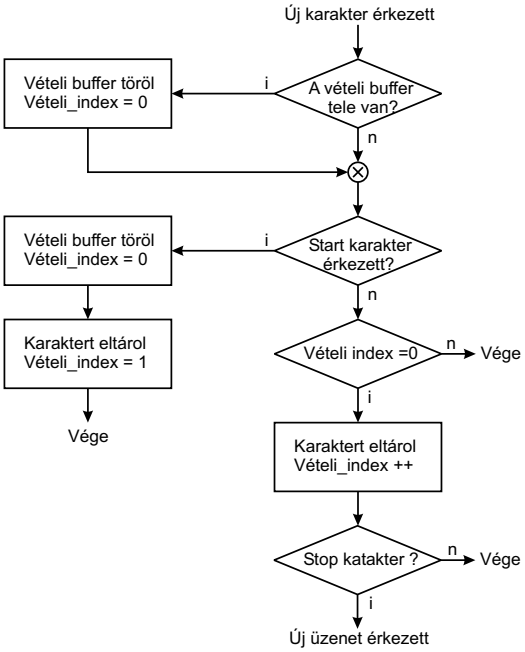
A hosszú üzenetben található adatmező két részből áll: az adatmező hosszát jelző 8 bites értékből és az utána következő tényleges adatokból. Az adatmező hossz tulajdonképpen az adatok hex-ascii konvertálása után kapott karakterlánc hossza.

A vételi folyamat minden egyes start karakter vétele után újra indul, amelynek menetét a 6.6 ábra szemlélteti.

A frissen érkezett üzenet feldolgozása pedig a stop karakter vétele után kezdődik. Csak a karakterek páros számának esetén folytatódik a folyamat. A vevő megvizsgálja a címzett kódját, és összehasonlítja a saját címével, csak egyezés esetén foglalkozik tovább az üzenettel, egyébként eldobja azt. Ezután az üzenetből kiszámítja az ellenőrző összeget és összehasonlítja az üzenetben elküldött értékkel. Egyezés esetén, ha hosszú üzenet érkezett, összehasonlítja az adatmező hosszát az üzenetben kapott értékkel. Ezek után mindkét üzenetfajta esetén az üzenetkódnak megfelelően jár el.

Egy érvényes rövid üzenet lehet például a következő: ***0100110123;.** Látható, hogy a címzett az 01h címet viselő egység, a feladó címe pedig 00h. Az üzenet kódja 11h, és az ellenőrző összeg pedig 0123h.

Alapvető szabály, hogy a bejövő üzenetre válaszolni kell. Ez kérés esetén a kért adatokat taratalmazó üzenet, parancs esetén pedig az ACK üzenet kódot tartalmazó rövid jelzőüzenet küldésével jár. A master egységben értelmezve van egy időzítő (Timeout Timer), amely minden egyes üzenet elküldése után elindul. Ha az értéke meghalad egy bizonyos határt (általában 1 sec), akkor az üzenet újbóli elküldésére kerül sor. Így biztosítható az üzenet sérülése



6.6. ábra. A vételi folymat

esetén a párbeszéd fenmaradása. Természetesen csak korlátozott számú üzenetismétlésnek van értelme.

7. fejezet

Alkalmazások

7.1. Adatgyűjtő üzemmód

Az adatgyűjtő megvalósításakor mind a DSP-n, mind a HOST-ton futó monitor programot ki kellett egészíteni alkalmazás specifikus részekkel.

7.1.1. Kompenzáló szűrő tervezése

Az adatgyűjtő rendszertervének blokkvázlatán látni lehet, hogy a bemenetek sávszélessége az opcionálisan bekapcsolható kompenzáló szűrők segítségével szélesíthető. Erre főleg audiofelvételek esetén van szükség.

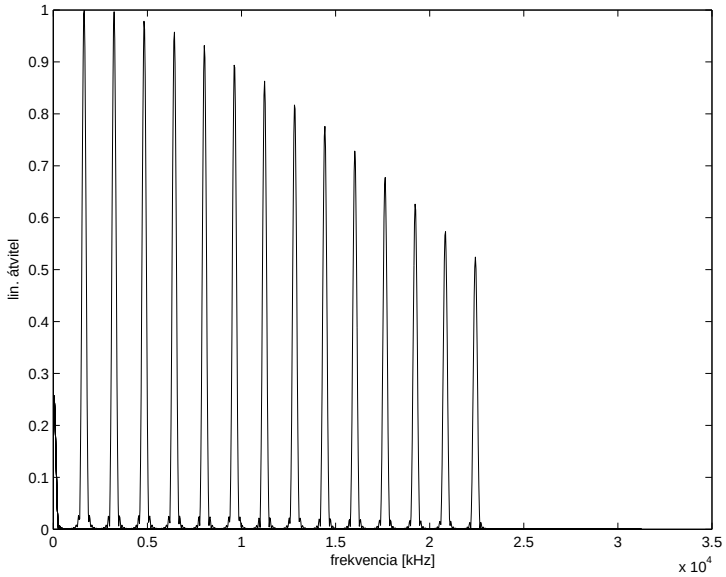
A cél egy olyan, lehetőleg kevés együtthatóból álló FIR szűrő létrehozása, amely jelen esetben a 62,4 kHz-es mintavételi frekvencia mellett legalább 20 kHz-ig biztosítja a 0 dB közeli átvitelt.

Első lépésként az egyik codec bemeneti csatornájának az átvitelét kellett lemérni. Ez úgy történt, hogy felvettem a rendszerrel 14 darab regisztrátumot, amelyeknek mindegyike egy adott frekvenciájú színusz jel volt. Ezek közül a legalacsonyabb 1.65 kHz-es, a legmagasabb pedig 22.5 kHz-es volt.

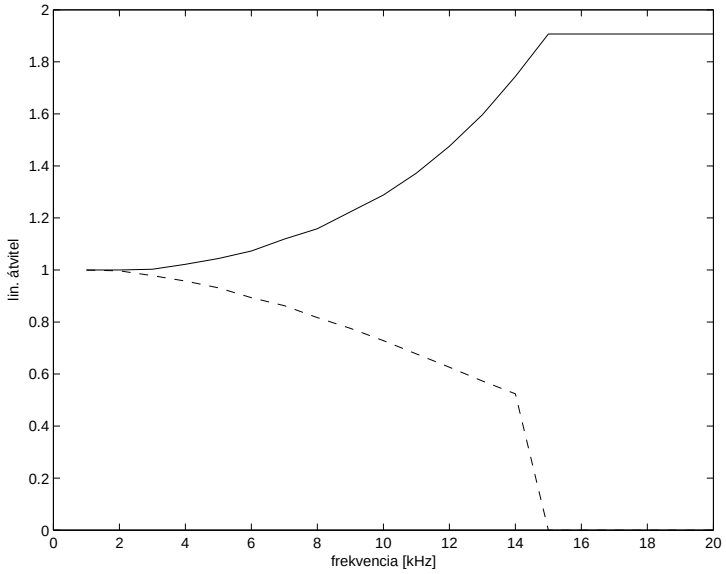
A kapott regisztrátumok feldolgozása és a szűrő tervezése Matlab segítségével történt.

Mindegyik regisztrátum spektruma egy az adott szinuszjel frekvenciájára jellemző helyen elhelyezkedő tüske volt, melynek csúcsértéke a codec adott frekvencián való átvitelét reprezentálta. A spektrumot FFT transzformáció és hanning ablak segítségével határoztam meg. A kapott eredmény a 7.1 ábrán látható.

Ennek megfelelően meg lehet határozni a kompenzáló szűrő specifikációját, amely a szükséges tartományban kiemel. A 7.2 ábrán a kapott átvitelt a szaggatott vonal, míg a szűrő specifikációt a folytonos vonal mutatja.



7.1. ábra. Az AD átalakító átvitelét reprezentáló 14 különböző frekvenciájú színuszjel együttes spektruma



7.2. ábra. Az AD átalakító interpolált átvitele (szaggatott vonal), a kompenzáló szűrő specifikációja (folytonos vonal)

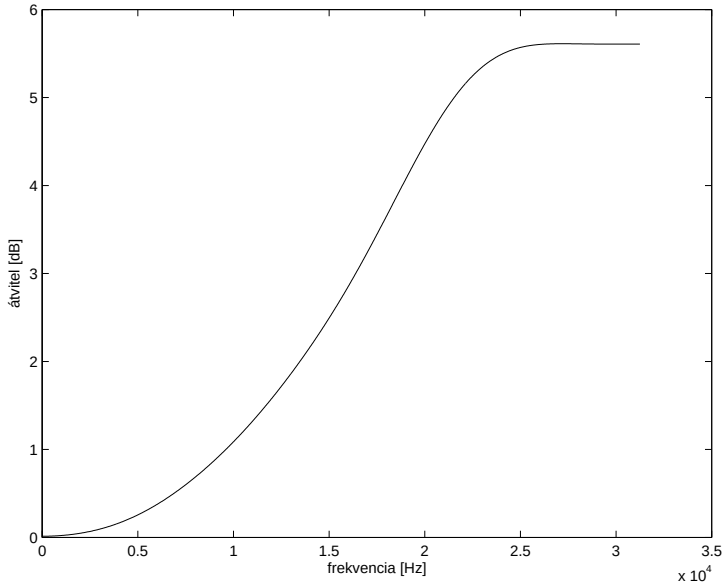
Látható, hogy a kompenzáló szűrő átvitele a nem definiált szakaszban egyenletes, ez biztosítja azt, hogy a szűrőtervező algoritmus minél kevesebb együtthatós szűrővel ki tudja elégíteni a specifikációt.

A decimáló szűrő tervezése frekvencia mintavételezéses módszerrel történt, a következő lépések szerint:

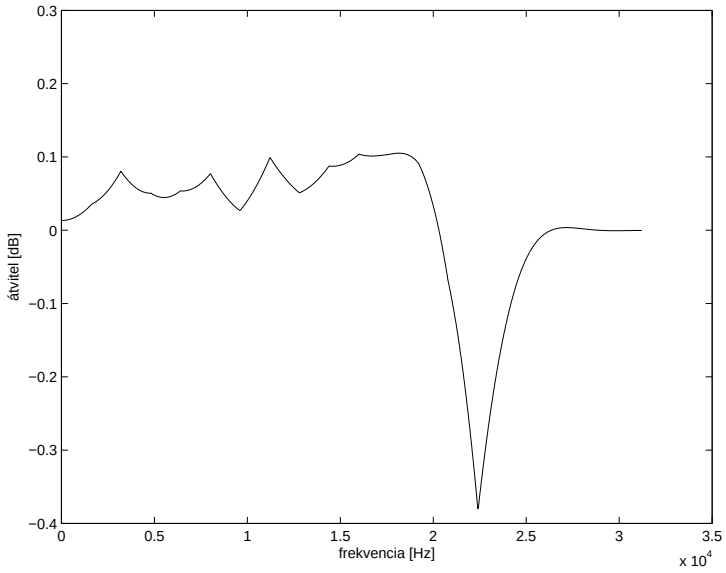
- **1. lépés** Nagy pontszámú (2048) specifikáció létrehozása a kiindulásként szolgáló kevés pontot (20) tartalmazó specifikációból lineáris interpoláció segítségével
- **2. lépés** A kapott specifikáció 2048 pontos IFT (Inverz Fourier) transzformálása.
- **3. lépés** A kapott eredmény lényeges (nullától bizonyos mértékben különböző) részének kiemelése hanning ablak segítségével. Az így kapott eredmény lesz a kompenzáló FIR szűrő együtthatókészlete.
- **4. lépés** A kapott együtthatókészlet nagy pontszámú (2048) FFT transzformálásával ellenőrizhető a szűrő tényleges átvitele. A maximálisan elfogadható eltérést jelen esetben 0.1 dB-ben határoztuk meg. Ha a szűrő nem teljesíti ezt a kritériumot, akkor a 3. lépésben alkalmazott hanning ablak szélességét növelni kell.

Jelen esetben a specifikációt 21 együtthatóval tudta teljesíteni a szűrő (7.3 ábra). A megtervezett szűrő átvitelének és a 14 pontban mért bemeneti átvitelnek az eredője 7.4 ábrán látható.

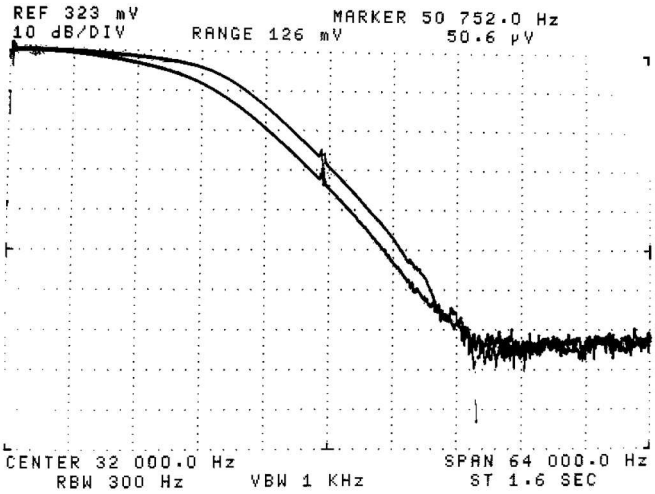
A 22.5 kHz közelében látható tüske magyarázata az, hogy a szűrőtervező algoritmus ilyen együttható szám mellett a specifikációban látható töréspontot nem tudta megfelelően közelíteni, de erre nincs is szükség, mert ez a pont már kívül esik a szükséges tartományon.



7.3. ábra. A megtervezett kompenzáló szűrő átvitele



7.4. ábra. A megtervezett szűrő és az átalakító eredő átvitele



7.5. ábra. Az egyes sorszámú csatorna átvitelének mérési eredményei (kompenzálatlan és kompenzált esetben)

A 7.5 ábrán az egyes sorszámú csatorna átvitelének mérési eredményei láthatók. Észrevehető, hogy a csatorna áteresztő tartományának átvitele kompenzált esetben sem vízszintes. Ez azért van, mert a szűrő csak a bemeneti átalakító átvitelét kompenzálja, és az eredő átvitelt a kimeneti átalakító lerontja.

7.1.2. Decimáló szűrő tervezése

Elvárás volt az adatgyűjtő alkalmazással szemben az, hogy széles mintavételi frekvenciatartomány mellett legyen képes a bemeneti jel rögzítésére. Ez a lehetőség a rendelkezésre álló véges kapacitású memória jobb kihasználtságát eredményezi, abban az esetben ha a rögzíteni kívánt jel alacsony frekvenciás komponensekből áll (5-10 Hz-es jeleket értelmetlen 64 vagy akár 8 kHz-el mintavételezni).

Mivel azonban a codec-ek legkisebb mintavételi frekvenciája is 8 kHz, így ezt szoftveres úton kell csökkenteni. Ezt az eljárást nevezzük decimálásnak, melynek során csak minden N-edik mintát tartjuk meg és a többit pedig eldobjuk. Ezáltal a létrejött mintavételi frekvencia az eredeti N-ed része. A decimálás során azonban a bemenőjelet először meg kell szűrni, hogy érvényes legyen a kimeneti jelre is a mintavételi törvény, tehát ne tartalmazzon $f_s/2N$-nél nagyobb komponenseket.

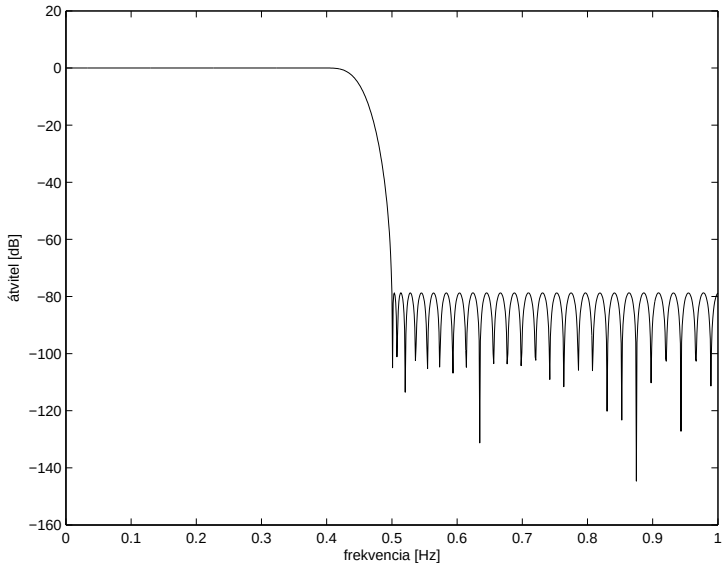
A rendszerben nyolcfokozatú decimálást alkalmaztam, és így a legkisebb

mintavételi frekvencia 30.518 Hz.

A decimálás magját a decimáló szűrő (ami esetünkben egy felező szűrő) alkotja, amely igen szigorú specifikációnak tesz eleget, ezért az együtthatók száma is igen magas. A szűrő specifikációja a következő:

- **Áteresztő tartomány:** DC-től $0.2 * f_s$-ig
- **Áteresztő tartomány maximális ingadozása:** 0.1%
- **Átmeneti tartomány:** $0.2 * f_s$-től $0.25 * f_s$-ig
- **Záró tartomány:** $0.25 * f_s$-től $0.5 * f_s$-ig
- **A záró tartomány átvitele:** -80 dB

A szűrőt remez algoritmus segítségével terveztem meg. Az algoritmus bemeneteként meg kell adni az áteresztő és záró tartományokat, és a hozzájuk tartozó átviteleket. Ügyelni kell arra, hogy az egyes tartományok között legyenek átmeneti tartományok is. Az algoritmus segítségével megtervezett szűrők sajátossága, hogy az áteresztő és záró tartományokban az átvitel ingadozik. A szűrő tervezése több lépésben történt, az együtthatószámot addig növelve, amíg a létrejött szűrő eleget nem tesz a specifikációnak. A 93 együtthatóból álló felező szűrő átvitele a 7.6 ábrán látható.



7.6. ábra. A felező szűrő átvitele

7.1.3. Az adatgyűjtő üzemmód működése

Az alkalmazás feladata a kiválasztott csatornák és mintavételi frekvencia alapján a beérkező jelek rögzítése. Az aktuális felvétel körülményeit a programban található *configurations* változó tartalmazza:

- **3.-0. bit** A szükséges decimáló fokozatok száma
- **7.-4. bit** A codec-eken baállítandó mintavételi frekvencia
- **8. bit** Kompenzáló szűrő engedélyezése
- **19.-12. bit** A bemenetek engedélyezése

A codec alkalmazás részben implementált jelfeldolgozó program a fenti bitek alapján alakítja ki a bejövő jel útját, tehát kapcsolja be vagy ki a kompenzáló szűrőt, illetve a szükséges fokozatokat.

Programozástechnikai okokból a jelfeldolgozó program a kiválasztott műveleteket minden bemeneti csatorna jelén elvégezi, és csak a rögzítés fázisában vizsgálja meg, hogy melyik csatorna(-ák) van(-nak) kiválasztva.

Jelen esetben a rendszerben 72 FIR szűrő került realizálásra. Ezek közül 8 a bemeneti csatornák átvitelét kompenzálja, 64 pedig a decimálásban vesz részt.

A programozástechnikai egyszerűsítés alapvetően az, hogy a szűrőket kiszámító rutint egy ciklus magjában helyeztem el, és a rutin által használt buffereket és változókat a rutin meghívása előtt minden egyes lépésben megfelelően módosítom.

A kompenzáló szűrők kimenetét tehát egy ciklus számítja ki, míg a decimáló szűrőket két egymásba ágyazott ciklus kezeli. A decimálás során a külső ciklus az aktuális decimáló fokozatot választja ki, a belső ciklus pedig az aktuális fokozatban elhelyezett 8 szűrőt vezérli. Decimálás esetén az adott szűrők nem mindig futnak le, és ezáltal nem mindig produkálnak kimenetet sem. Arról, hogy az adott fokozatok közül egy mintavételi időpontban mindig csak egy működjön, az ütemező gondoskodik.

Az adatgyűjtő több egymás után elkészített (max. 100) felvétel kezelésére alkalmas. A felvételekre jellemző adatokat táblázatban tárolja el. Az egyes felvételekhez tartozó adatok a következők:

- A felvétel beállítása
- A felvétel kezdeti címe az SDRAM-ban
- A felvett minták száma (csatornánként)

A felvétel beállítása, ami tulajdonképpen a *configurations* változó értéke a felvétel kezdetekor, nagyon fontos szerepet játszik a dokumentálás, valamint visszajátszás esetén, hiszen a felvételt megadott mintavételi frekvenciával és a megadott kimeneti csatornákon kell megjeleníteni.

Az alkalmazás fent említett része a *codec alkalmazás* belépési ponthoz tartozó területén került implementálásra. Ahhoz, hogy ez megfelelően működhessen, az SDRAM vezérlőt, a kompenzáló szűrő és a decimáló fokozatok által használt cirkuláris buffereket és változókat inicializálni kellett, az alkalmazás bekapcsolásakor egyszer lefutó alkalmazás start belépési ponthoz tartozó területen elhelyezett utasítások által.

A HOST-tól kapott vezérlő parancsok feldolgozása az UART alkalmazás belépési ponthoz kapcsolódó rutinok által történt. Az alkalmazásnak szóló üzenetek a következők lehetnek:

- **Paraméterek beállítása.** A felvétel körülményeinek (mintavételi frekvencia, engedélyezett bemenetek...) beállítására szolgál.
- **Felvétel start.** Az üzenet engedélyezi a feldolgozott jelek SDRAM-ba való eltárolását.
- **Felvétel leállítása.** A tárolás tiltása, és a felvétel beállításának, paramétereinek mentése a táblázatba.
- **Felvételeket nyilvántartó tábla lekérdezése.** Az üzenetben található indexnek megfelelő és az azt követő további kilenc érték lekérdezése a táblázatból.
- **Utolsó felvétel törlése.** A felvétel paramétereinek táblázatból való törlése és az SDRAM vezérlőben található írási cím visszaállítása által a legutolsó felvétel törlődik.
- **Kiválasztott felvétel lejátszása.** Az üzenetben található indexnek megfelelő felvétel lejátszása, a táblázatban rögzített paraméterek szerint.
- **SDRAM tartalmának lekérdezése.** Az üzenet segítségével a HOST lekérdezheti az SDRAM tartalmát. A felvétel kezdeti címe és mintái számának ismeretében az adott felvételek feltölthetők.
- **Új felvétel létrehozása.** Ez az első lépése annak a folyamatnak, amelynek során a HOST felvételt tölt le a kártyára. Az üzenet tartalmazza a letöltendő felvétel beállításait, amelyeket a kezdőcímmel együtt a program eltárol a táblázatba.

- **Felvétel letöltése a HOST-ról.** Ezzel az üzenettel történik a letöltendő felvétel átvitele. Ez az üzenet tartalmazza tulajdonképpen a felvétel mintáit.
- **Felvétel lezárása.** Ez az üzenet zárja a felvétel letöltését. Ekkor kerül eltárolásra a felvétel mintáinak száma.

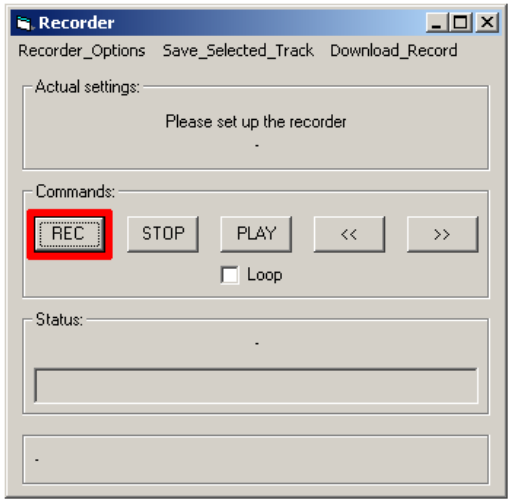
Ezeket a üzeneteket mindig a HOST adja ki, és az alkalmazás ennek megfelelően válaszol.

7.1.4. Az adatgyűjtő kezelői felülete

Az adatgyűjtő kezelői felülete a HOST számítógépen futó Visual Basic-ben implementált monitor program kiegészítése.

A vezérlő panel

Az adatgyűjtő kezelőfelületét három panel alkotja. Az adatgyűjtő vezérlő panelje a 7.7 ábrán látható.



7.7. ábra. A vezérlő panel

A panelen elérhető az összes felvételt és lejátszást vezérlő parancs. Az *Actual settings* keretben mindig az adatgyűjtő aktuális beállítása, míg a *Status* keretben a folyamatban lévő művelet és állapota látható.

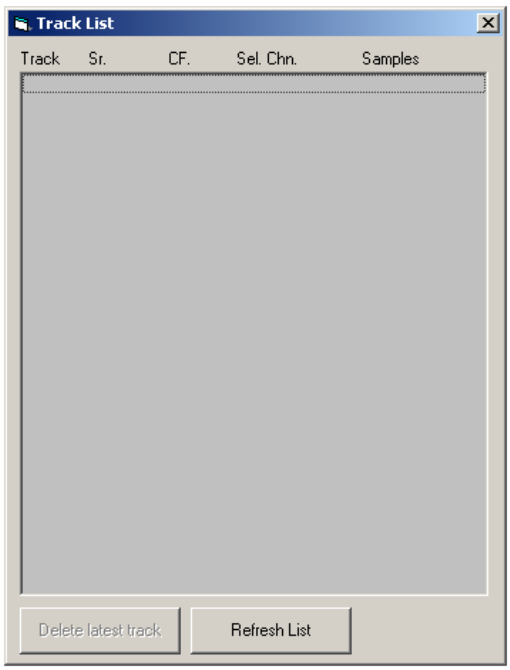
A felvétel vagy lejátszás elindítását, leállítását, valamint az aktuális felvétel (track) kiválasztását a *Command* keretben található gombok segítségével végezheti a felhasználó. A *Loop* opció bekapcsolásával a lejátszás repetitívvé válik.

A *Save_selected_Track* menüpont segítségével a kiválasztott felvételt feltölthetjük a HOST számítógépre archiválás vagy feldolgozás céljából. A felvétel Matlab által feldolgozható ascii fájlba kerül.

A *Download_Record* menüpont segítségével pedig az általunk készített ascii fájl letöltésére van lehetőség. Letöltés előtt azonban meg kell adni a rendszernek a felvétel beállításait.

A felvételek adatait megjelenítő panel

Az adatgyűjtő alkalmazás által használt, a felvételek adatait nyilvántartó táblázatra a kezelő felületnek is szüksége van. A DSP-ben és a HOST-ton található táblázatnak minden esetben azonosnak kell lennie egymással, ezért a program induláskor, minden felvétel, illetve felvétel törlés után egyezteti azokat. A táblázatban található adatokat a 7.8 ábrán látható panelen jeleníti meg.

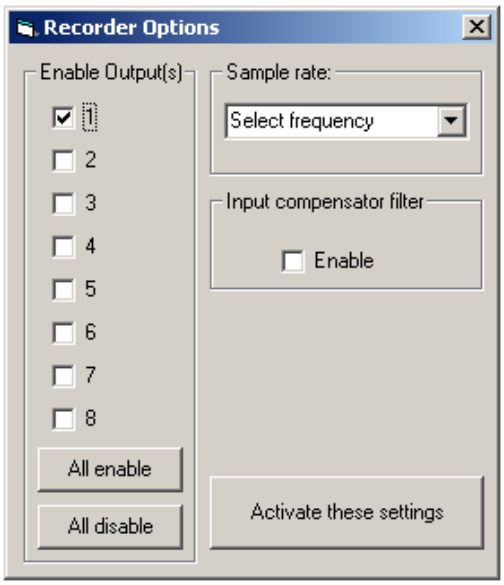


7.8. ábra. A felvételek adatait megjelenítő panel

A listában minden egyes felvétel mellé kijelzésre kerül a sorszáma, mintavételi frekvenciája, a kompenzáló szűrő állapota, a kiválasztott csatornák és a minták száma. A panelen található *Refresh_List* gomb segítségével a DSP-n található táblázat feltölthető. Az utolsó felvétel törlése pedig a *Delete_Latest_Track* gomb megnyomásával érhető el.

A felvétel beállítását tartalmazó panel

Az DSP-n futó adatgyűjtő alkalmazás *configurations* változójának tartalmát, és ezáltal a felvétel tulajdonságait a 7.9 ábrán látható panel segítségével módosíthatja a felhasználó.



7.9. ábra. A felvételek beállításait tartalmazó panel

Ezen a panelen választhatja ki a felhasználó azokat a csatornákat, melyeknek jele a későbbiekben rögzítésre kerül, valamint a mintavételi frekvenciát, amely a legördülő menüben található értékek egyike lehet. Továbbá itt lehet ki- vagy bekapcsolni a kompenzáló szűrőt. A paraméterek beállítását követően az *Activate these settings* feliratú gombbal lehet érvényesíteni azokat az adatgyűjtőben.

7.2. Általános alkalmazás futtatása

Ez a fejezet egy általános alkalmazásnak a sokcsatornás rendszerben való implementálásáról szól, tehát felfogható használati utasításként is.

Mint azt már korábban is említettem, a DSP-n futó programok fejlesztése VisualDSP segítségével történik a következő lépések szerint:

1. lépés - A fejlesztés környezetének beállítása.

Az alkalmazás számára létre kell hozni egy új könyvtárat, és benne egy VisualDSP projekt fájlt. Majd ide kell másolni a következő fájlokat:

- **It_8IO_DSP.asm** Interrupt táblázat.
- **Boot_8IO_DSP.asm** Boot betöltő rutin.
- **Init_8IO_DSP.asm** Inicializáló szakasz.
- **Mon_8IO_DSP.asm** Monitor program.
- **App_8IO_DSP.asm** Üres alkalmazást tartalmazó program.
- **def21065l.h** Deklarációk.
- **lnk_21065l_8IO_DSP.ldf** Linker fájl.

Ezek után a projekt fájlba be kell olvasni az App_8IO_DSP.asm, def21065l.h, lnk_21065l_8IO_DSP.ldf fájlokat.

2. lépés - Az alkalmazás implementálása

Az alkalmazás megfelelő részeinek elhelyezése a rendelkezésre álló belépési pontoknak megfelelő területeken:

- **Load_Application_Context** Az alkalmazás bekapcsolásakor kapja meg a vezérlést (Alkalmazás Start).
- **Application_Main** Az alkalmazás bekapcsolt állapota esetén használt főciklus.
- **Codec_Input_Application** Minden egyes mintavételi időpontban megkapja a vezérlést (Codec Alkalmazás).

- **Serial_Input_Application** Minden egyes monitor által értelmezhetetlen üzenetkódot tartalmazó bejövő üzenet esetén kapja meg a vezérlést (UART Alkalmzás)
- **Clear_Application_Context** Az alkalmazás kikapcsolásakor kapja meg a vezérlést (Alkalmazás Stop).

Az alkalmazás az *Incoming_Channel_1..8* fixpontos változókban kapja meg az egyes bemeneti csatornák aktuális értékeit, és az *Outgoing_Channel_1..8* fixpontos változók írásával módosíthatja a kimeneti csatornák értékeit. A változókban található értékek típusa miatt a jelfeldolgozó algoritmus futása előtt fixpontos-lebegőpontos átalakításra, utána pedig lebegőpontos-fixpontos átalakításra van szükség.

Az *Iputs_State* változó a digitális bemenetek állapotát tartalmazza, amely minden mintavételi időpontban frissül. A digitális kimenet állapotát pedig az *Outputs_State* változó írásával lehet módosítani, amely szintén minden mintavételi időpontban kiírásra kerül.

A bejövő üzenetet a monitor *Receive_Buffer_Byte* tömb átadásával teszi elérhetővé az alkalmazás számára. Ebben a tömbben az üzenet felépítésének megfelelő sorrendben helyezkednek el az adatok, azzal a különbséggel, hogy már nincs benne a tömbben a start és a stop karakter, illetve az üzenetben található hex-ascii karakterek kettesével bájtos alakra vannak konvertálva.

3. lépés - Az alkalmazás futtatása

Az elkészített alkalmazást a monitor programmal együtt le kell fordítani. Ezek után J-TAG használatával a megírt alkalmazás üzemszerűen kipróbálható.

4. lépés - Az alkalmazás letöltése

A kipróbált kész alkalmazás adat és utasítás mezőit VisualDSP *dump* parancsa segítségével két fájlba mentjük, melyek ezek után a HOST-on futó monitorprogrammal letölthetők a FLASH memóriába. Ezek után a rendszer önjáróvá válik.

8. fejezet

Összefoglalás, kitekintés

8.1. Az eredmények értékelése

Jelen dolgozat egy sokcsatornás jelfeldolgozó rendszer fejlesztésével és működésével, valamint a rendszerben implementált sokcsatornás adatgyűjtővel foglalkozik. A rendszer tervezése során mind hardveres, mind szoftveres területen meg kellett oldani a felmerült problémákat, valamint létre kellett hozni egy kommunikációs protokolt, amely a rendszer és a HOST számítógép közötti adatcserére szolgál.

A dolgozat elején bemutattam a fejlesztés hátterét, és azt a saját fejlesztésű adatgyűjtő iránti igényt, ami a fejlesztés egyik legfőbb kiváltó okának bizonyult.

A kövezető fejezetben a rendszert alkotó harver és szoftver komponensekkel, valamint az adatgyűjtővel szemben támasztott igényeket specifikáltam. Majd bemutattam az egyes komponensek rendszerterveit, amelyek a későbbi tervezés alapjául szolgáltak.

A negyedik fejezetben részletesen ismertettem a felhasznált főbb komponensek (DSP, FPGA, Codec-ek) tulajdonságait és működését, valamint a hozzájuk kapcsolódó környezetet, amelyek az áramkör főbb egységeit alkotják.

A létrejött kártya élesztésével és bemérésével az ötödik fejezet foglalkozik, és ugyancsak itt került bemutatásra az FPGA-ban található periféria egységek felépítése és működése.

A szoftver tervezése során meg kellett oldani a hardver megfelelő kezelését valamint az alkalmazások gyors és egyszerű beágyazását. A hatodik fejezet ennek a folyamatnak az eredményét mutatja be.

A sokcsatornás rendszeren megvalósítottam a felmerült igényeket kielégítő adatgyűjtő alkalmazást. A sikeres implemetáció bizonyította a rendszer mű-

ködőképességét, valamint használhatóságát is.

A dologozat utolsó fejezete pedig egyfajta használati utasításként szolgál, segítséget nyújtva az alkalmazások sokcsatornás rendszeren való implementálásához.

8.2. Továbbfejlesztési lehetőségek

A rendszer megvalósult és használható, de továbbfejlesztésére van mód. A jövőben szándékomban áll megvalósítani egy függvény könyvtárat, amely arra szolgál, hogy a rendszer még inkább támogassa, még kényelmesebbé tegye a különböző alkalmazások beágyazhatóságát.

Szintén továbbfejlesztési lehetőség a gyorsabb kommunikáció megvalósítása. Erre a célra rendelkezésre áll egy Ethernet modul, illetve egy USB chip, valamint az alkalmazásuk során felgyülemlett tapasztalatok.

Irodalomjegyzék

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

[2] Bittware. http://www.bittware.com.

[3] Casual Systems. http://www.casual.com.

[4] Domain Technologies. http://www.domaintec.com.

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

[6] Pentek Inc. http://www.pentek.com.

[7] Analog Devices, Inc. *ADSP-2106x SHARC User's Manual*, 1997.

[8] Analog Devices, Inc. *AD73322 Data Sheet*, 2001.

[9] Atmel Co. *AT49LV004(N)(T) Data Sheet*, 2003.

[10] Bogár I., Faragó Á., Molnár K. Nyolccsatornás jelfeldolgozó rendszer fejlesztése, TDK dolgozat, BME Méréstechnika és Információs Rendszerek Tanszék. 2002.

[11] Samsung Electronics Co. *M366S3253DTU Data Sheet*, 2002.

[12] Texas Instruments Incorporated. *TL16C550(C)(CI) Data Sheet*, 1994.

[13] Xilinx. *Spartan-II 2.5V FPGA Family: Complete Data Sheet*, 2003.

Függelék