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

Makai Adél Veronika

Zengető megvalósítása jelfeldolgozó processzorral

Konzulens: dr. Sujbert László

2005. május

Nyilatkozat

Alulírott Makai Adél, 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, melyet szó szerint, vagy azonos értelemben de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, 2005. május

.....

Makai Adél

Tartalomjegyzék

Kivonat

A dolgozat valós idejű, változtatható paraméterű zengető megvalósításával foglalkozik. Először röviden tárgyalja a zengető elméleti hátterének megértéshez szükséges teremakusztikai alapfogalmakat. Ezután áttekinti a zengetéshez használt módszereket, amelyek két nagy csoportra bonthatók, a valós impulzusválasszal végzett konvolúcióra és az impulzusválasz szintézisre. A konvolúciós módszer számításigénye nagy, még a mai nagy teljesítményű jelfeldolgozó processzorok kapacitását is meghaladja. Ennek a módszernek a megvalósításához konkrét terem impulzusválaszára van szükség, amelynek mérési módszereit a dolgozat egy fejezetben külön összefoglalja, valamint ismerteti a munka során végzett impulzusválasz mérések során használt mérési összeállítást és eredményeket.

Az elkészített zengető egy konkrét, Moorer által javasolt algoritmust valósít meg, amely FIR-szűrőből, fésűszűrőkből és mindentáteresztő szűrőből felépülő hálózattal szintetizálja az impulzusválaszt.

A zengetést megvalósító rendszer két egységből áll, az egyik a személyi számítógépen futtatható Java 2 Standard Edition 1.5.0 alatt írt felhasználói felület, a másik rész a nyolc-csatornás jelfeldolgozó kártya ADSP-21065L típusú lebegőpontos processzorán futó zengető assembly nyelven megírt algoritmus. A zengetés tulajdonságai kétféle felületen keresztül állíthatók be. Az egyiken közvetlenül a zengetést megvalósító szűrőhálózat erősítés és késleltetés paraméterei változtathatók, míg a másikon érzeti jellemzők értéke választható meg, mint az utózengési idő, közvetlen/korai/késői hang aránya. A felhasználó által beállított zengetési paraméterek üzenetek formájában, soros porton keresztül jutnak el a PC-ről a processzorba, amely az üzenetek értelmezése és feldolgozása után az új értékeket beépíti a zengető algoritmusba.

Az elkészült rendszer jelenleg 16 kHz-es mintavételi frekvencián működik, ez alacsonyabb az audio rendszerekben megszokottnál. Ennek oka, hogy az algoritmus memóriaigénye nagyobb mintavételi frekvencia esetén meghaladja a processzorban rendelkezésre álló belső memória méretét, ezért ezt a problémát másik hardver eszköz választásával lehet megoldani. A felületen beállítható érzeti jellemzők és szűrőparaméterek kapcsolatának meghatározása további vizsgálatokat igényel.

Abstract

This master's thesis deals with the realization of a real-time reverberation system with variable parameters. First of all, it discusses the basic elements of room acoustics required to understand the function of reverberation. Then it summarises the two main methods of reverberation, the convolution and the synthesis of impulse response. Although nowadays there exist high capacity signal processors, convolution is still impossible to achieve in a real-time application. This method needs the impulse response of a real room that can be measured with different techniques described in this work.

The reverberator developed uses the synthesis technique based on the algorithm proposed by Moorer that consists of a FIR filter, comb filters and an allpass filter.

The reverb system can be divided into two main part, first is the user interface on a personal computer, the second one is the algorithm running on the ADSP-21065L digital signal processor of an eight channel signal processing system. There are two different user interfaces. Through the first one, the gain and delay parameters of the filters can be modified, while through the second interface other parameters can be changed, such as reverberation time, dry/early/wet ratio. These parameters are converted into messages and sent from the PC to the signal processor where the new values are built into the algorithm.

The system now works on 16 kHz sampling frequency that cannot be raised as on higher frequencies the algorithm needs more memory that are available on the signal processor. This problem can be solved by choosing another processor. In addition, the conversion of sensation properties to filter parameters poses some challenging questions.

1 Bevezetés

Ha egy teremben körülnézünk, látjuk a minket körülvevő tér sajátosságait, a terem nagyságát és formáját, a falak minőségét, a teremben található tárgyak elhelyezkedését. Szemünket becsukva, pusztán hallásunkra hagyatkozva is érzékelhetjük a termek akusztikai tulajdonságait. Hangfelvételek készítését gyakran stúdióban, mesterséges körülmények között végzik, ezért hiányzik belőlük a tér hatása, ennek reprodukálását valósítja meg a zengető.

Léteznek valós időben működő, jó minőségű, ám sok jelfeldolgozó processzort tartalmazó és drága zengető berendezések. Olcsóbb megoldásként használhatók egyszerű asztali PC-n futtatható szoftverek, amelyek széles választékban kaphatók, de az általuk megvalósított algoritmusok nagy számításigényük miatt nem valós időben működnek.

Célom egy olyan zengető elkészítése volt, amely valós időben fut és paraméterei változtathatóak, ezáltal többféle térhatás létrehozására alkalmas. A zengető algoritmus mindössze egyetlen ADSP-21065L típusú lebegőpontos jelfeldolgozó processzort használ. A paraméterek változtatásához egy PC-n futó Java-s felhasználói felületet készítettem, mivel a Java egyszerű eszközöket biztosít felhasználói felületek fejlesztéséhez és esztétikus kialakításához.

Diplomatervem második fejezetében a zengetés megértéséhez szükséges teremakusztikai alapfogalmakat tekintem át, majd a harmadik fejezetben bemutatom a zengető rendszertervét. A negyedik fejezet többféle zengetési módszer működésének leírását tartalmazza, amelyek között szerepel az általam megvalósított algoritmus. Itt foglalom össze a termek mérésére használt módszereket is. Az ötödik és hatodik fejezetek tartalmazzák a zengető hardver részének bemutatását és a hardverrel együttműködő szoftver leírását. Az általam végzett terem impulzusválasz mérések eredményeit a hetedik fejezetben tárgyalom. Az összefoglalás keretében értékelem az elkészült rendszert és javaslatokat teszek a további fejlesztésre.

2 A zengetés

Mindennapi életünket akusztikus szemmel vizsgálva megállapíthatjuk, hogy különböző terekben mozgunk, amelyeknek mind sajátos, jellegzetes hangzása van. Gondoljunk csak az előadótermekre, irodákra, koncerttermekre, színházakra, templomokra vagy akár a fürdőszobánkra. Az előbb említettek közül néhány kifejezetten prózai, vagy zenés előadásokhoz épült, ezek akusztikáját ennek megfelelően tervezték. Manapság azonban nemcsak ilyen helyeken hallgatunk műsorokat és zenét, hanem például autóban, fülhallgatóval a buszon, stb. Ezekben az esetekben nagyon lényeges, hogy a felvétel tartalmazza az eredeti tér zengő hangzását, hiszen a lehallgató tér ezt nem tudja biztosítani. A felvételek egy része az előbb már említett helyszíneken készül, másik része „mesterséges" körülmények között, stúdióban. Léteznek stúdiók, amelyek egy-egy műfajra specializálódtak (pl. kis- és nagyzenei, felolvasó, hangjáték stúdiók), vagyis ennek megfelelő hangtérrel rendelkeznek. Bár nehéz olyan helyiséget tervezni, ami többféle műfaj felvételére illetve előadására alkalmas, már léteznek olyan termek, amelyek mozgatható falakkal, hangvető ernyőkkel, zengőkamrákkal, függönyökkel ezt lehetővé teszik. Ezeket a módszereket alkalmazzák az újonnan épült Nemzeti Hangversenyteremnél is. Vannak olyan felvételek is, amelyek gyakorlatilag süketszobában készülnek. Ezek jellegzetessége a készítés körülményeiből adódóan, hogy zengésmentesek, szárazak, ezért utófeldolgozást igényelnek. Előnyük, hogy mivel szinte teljesen térhangzástól mentesek, ízlés szerint alakíthatók.

Itt érdemes még megemlíteni a számítógépes zenét, amely szintén hanghatások, vagy más néven effektek sokaságát, így a zengetést is használja.

Élő koncerteken is gyakran használnak effekt berendezéseket. Ez valóban nélkülözhetetlen szabadtéren, ahol minimális a visszaverődések keletkezésének lehetősége, stadionokban, amelyek méretéből és formájából adódóan önmaguk nem biztosítanának jó hangzást. De találkozhatunk az effekt berendezésekkel kisebb, könnyűzenei élményt nyújtó helyeken is (pubok, egyetemi klubok), ahol a cél a műfajnak megfelelő hangzás előállítása. Ezek az alkalmazások abban különböznek a stúdiófelvételek utómunkálataitól, hogy valós időben zajlanak.

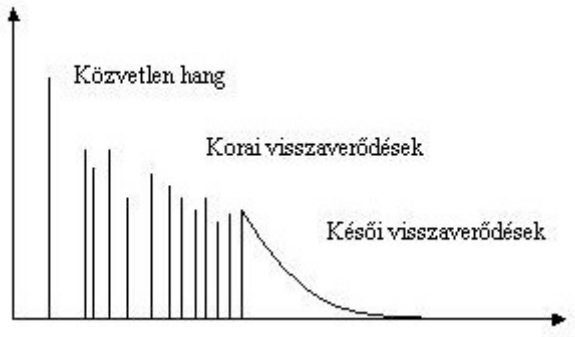
A zengetés valósághűség szempontjából két csoportra bontható, alkalmas konkrét terem hangzásának reprodukálására és egyszerűen csak kellemes, teremfüggetlen hangzás elérésére egy már elkészített száraz felvételen vagy akár élő koncerten.

A zengetés megvalósítására hardveres és szoftveres megoldásokat is alkalmaznak. Az előbbiek inkább valós idejű esetekben, az utóbbiak pedig utómunkálatoknál hasznosak. Hardveres megoldások közé tartoznak az analóg keverőpultok zengetői, de kaphatók külön erre a célra készült eszközök is. A legismertebb effekt rendszerek gyártói közé tartoznak a Lexicon, a t.c.electronics és a Yamaha, azonban az általuk készített eszközök ára igen borsos. Szoftveres megoldásnak tekinthető a digitális keverőpultok zengetője, és a felvételek készítésére és szerkesztésére alkalmas programok, például a Cubase, a Sound Forge és a Cool Edit.

A fejezet további részében összefoglalom a teremakusztikához kapcsolódó alapfogalmakat és zengetés alapvető módszereit.

2.1 Teremakusztikai fogalmak

Ha feltételezzük, hogy a terem egy lineáris, invariáns, kauzális rendszer, akkor a terem két pontja között mért impulzusválasz minden információt magában foglal a teremről. A pontok kiválasztása befolyásolja az impulzusválasz tulajdonságait, a választástól függően eltérések mutatkozhatnak. Az 1. ábra egy tipikus impulzusválaszt mutat.



1. ábra Jellegzetes impulzusválasz

Az impulzusválasz három részre bontható [1.]. Az első a közvetlen hang, ennek késleltetése a 0 időpillanathoz képest (amikor a hangforrás adni kezd) a hangforrás és a hallgató távolságáról ad információt. A második szakasz a korai visszaverődések szakasza. Ez alapvető fontosságú a terem méretének és alakjának érzékelésében. Ez az impulzusválasz helyfüggő része. A harmadik rész a diffúz visszaverődések szakasza, itt a visszaverődések már jóval sűrűbben követik egymást, mint a előző szakaszban, lecsengésük exponenciális jellegű. Mint ahogy az elnevezés is mutatja, ez a diffúz rész, vagyis ez nem nagyon függ a mérési pontok kiválasztásától.

Ha a terem alakja túl szabályos (pl. párhuzamos falak, kupolák), akkor az impulzusválaszban kialakulhat periodikusság, ez kellemetlen hangzással, csörgéssel járhat.

Az impulzusválaszból különböző jellemzőket számolhatunk, amelyek az esetleges összehasonlításokat egyszerűbbé, szemléletesebbé teszik.

Az ***utózengési idő*** megadja, hogy a teremben felhalmozódott energia mennyi idő alatt csökken egymilliomod részére, vagyis 60 dB-lel. Ez meghatározható egy valódi, megmért impulzusválaszból, vagy kiszámítható a Sabine-képlet, vagy a Norris–Eyering-egyenlet segítségével. A Sabine-képlet diffúz hangteret feltételez, vagyis közel egyenletes energiaeloszlást.

$$T_{60} = \frac{(24\ln 10)V}{c\sum_i \alpha_i A_i} \qquad (2.1)$$

A 2.1 képletben T_{60} jelenti az utózengési időt, V a terem térfogata, c a hangsebesség, A_i a terem felületeinek nagysága, α_i az A_i felületekhez tartozó elnyelési tényező. Ez utóbbi frekvenciafüggő mennyiség, így segítségével meghatározható az utózengési idő a frekvencia függvényében. A 2.2 Norris–Eyering-képlet szintén diffúz hangteret feltételez, de figyelembe veszi, hogy a lecsengés nem teljesen folyamatos, hanem a felületeken, szakaszosan történik meg.

$$T_{60} = 13{,}82\frac{4V}{cS\left[-\ln\left(1-\alpha'\right)\right]} \qquad (2.2)$$

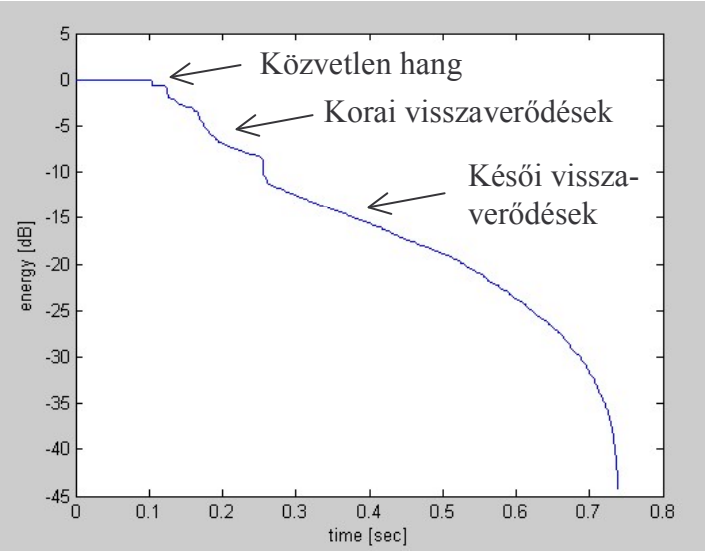
Az előző képlethez hasonlóan T_{60} az utózengési idő, V a terem térfogata, c a hangsebesség, S a terem teljes felülete, α' pedig a teremre számolt átlagos elnyelési tényező, amely szintén frekvenciafüggően adható meg.

Az utózengési idő tehát frekvenciafüggő jellemző. A teremben terjedő hanghullámok a felületekhez érve részben visszaverődnek (geometriai és szórt

visszaverődések formájában), részben elnyelődnek. A visszaverődés és elnyelődés aránya a felület tulajdonságaitól (pl. vastagság, sűrűség) függ. Közepes és nagyfrekvenciás elnyelésre szöveteket és egyéb pórusos anyagokat szoktak használni, kisfrekvenciás elnyelésre például kemény lapokat, feszített bőrt vagy Helmholtz-rezonátort.

Az energia eloszlásához kapcsolódóan használatos a ***korai lecsengési idő*** is (EDT – Early Decay Time), amely a lecsengés korai szakaszához (-10 dB) szükséges időt jelenti.

Az ***energialecsengési görbe*** (EDC – Energy Decay Curve) az impulzusválaszból számítható, az energia változását mutatja az idő függvényében. A 2. ábra egy ilyen görbét mutat, amelyen viszonylag jól elkülöníthető a közvetlen hanghoz tartozó energiacsökkenés, a korai és késői visszaverődési szakaszok. Az energialecsengési görbéből könnyen leolvasható az utózengési idő, amely a –60 dB-es ponthoz tartozik.



2. ábra Energialecsengési görbe

Az ***energialecsengési sík*** (EDC – Energy Decay Relief), vagy vízesés ábra szintén az energiaváltozást mutatja, de három dimenzióban, vagyis nemcsak az idő, hanem a frekvencia függvényében is.

Újabb jellemzők a ***clarity*** és a ***definition***, ezek egy adott τ időpont előtti és utáni energiamennyiség arányát adják meg. Kiszámításuk módját a 2.3 és 2.4 képletek adják meg.

$$C(\tau) = 10\log_{10} \frac{\int_0^{\tau} p^2(t)dt}{\int_{\tau}^{\infty} p^2(t)dt} [dB] \qquad (2.3)$$

$$D(\tau) = \frac{\int_0^{\tau} p^2(t)dt}{\int_0^{\infty} p^2(t)dt} [\%] \qquad (2.4)$$

Beszédcélú termeknél τ-t 50 ms-ra szokták választani, zenei célú termek esetén 80 ms-ra. Érhetőség szempontjából annál jobb, minél több energia érkezik τ idő előtt, a később érkező energia zavaró.

A zengés statisztikai jellemzésére szokták használni a ***módussűrűséget*** (modal density) és a ***visszaverődések sűrűségét*** (echo density).

Téglaltest alakú (shoebox) helységek sajátfrekvenciái kiszámíthatók a 2.5 képlettel, ahol $f_{x,y,z}$ a sajátfrekvencia, n_x, n_y és n_z a módusok csomópontjainak száma, *Lx*, *Ly* és L_z a helyiség oldalainak hossza.

$$f_{x,y,z} = \frac{c}{2}\sqrt{\left(\frac{n_x}{L_x}\right)^2 + \left(\frac{n_y}{L_y}\right)^2 + \left(\frac{n_z}{L_z}\right)^2} \qquad (2.5)$$

Kimutatható, hogy ebben az esetben a módusok száma a frekvencia négyzetével arányosan nő. Szabálytalan alakú termek szintén jellemezhetők sajátfrekvenciáikkal, de a helyiségek bonyolultsága miatt általában nem adható meg hozzájuk zárt formájú megoldás. A módusok számának négyzetes aránya a frekvenciához azonban általában itt is teljesül. Termek akusztikájának vizsgálatakor és tervezésekor nemcsak a módusok száma, hanem azok eloszlása is fontos, mert az egy sávban sűrűsödő módusok kellemetlen hangzást eredményeznek. Téglatest alakú helyiségekben léteznek ajánlások az oldalhosszak arányára, amelyek betartásával elérhető a módusok kedvező eloszlása.

A visszaverődések sűrűsége zárt térben időben egyre nő, ez a tulajdonképpeni zengés. A 4.1.1 pontban leírt tükörforrások módszeréből levezethető, hogy a visszaverődések száma (és a tükörforrások száma) az eltelt idő négyzetével arányosan nő.

2.2 Zengetési módszerek

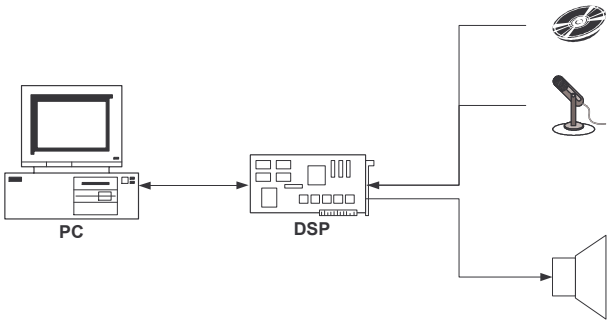
A zengetés megvalósításának korai időszakában mechanikai eszközöket használtak a térérzet létrehozásához. Ennek egyik példája a rugós zengető, amely néhány, keretre lágyan felfüggesztett rugót jelentett. A beérkező elektromos jelből átalakított mechanikai jel került a rugók egyik végére gerjesztésként, majd haladt rugók másik végéig, ahol a rugók rögzítésének illesztetlensége miatt, vagyis a nem hullámimpedanciával történő lezárás miatt visszaverődések keletkeztek, amelyek a többi rugót is gerjesztették. A beérkező jellel ellentétes oldalon egy elektromechanikai átalakító a mechanikai jelből elektromos jelet generált. Egy másik mechanikai megoldás a lemezes zengető volt, amely egy acélból készült zengőlemezen elhelyezett két elektromechanikai átalakítóból állt, a lemez két oldalával szemben egy-egy csillapító anyaggal bevont lemez távolsága szabályozta a zengést.

Digitális zengetésre alapvetően kétféle módszert alkalmaznak. Az első módszer esetében felveszik a terem impulzusválaszát, majd ezt konvolválják a száraz felvétellel, ez teljesen valósághű, konkrét teremhez tartozó eredményt szolgáltat. Bár ma már léteznek gyors konvolúciós módszerek, ez a feladat még így is meglehetősen erőforrásigényes. Ha a terem impulzusválasza 3 másodperces, akkor ez 44,1 kHz-es mintavételi frekvencia esetén 132300 pontos FIR szűrőt jelent. Ez óriási műveletigény, ugyanakkor nagy termek esetén ennél jóval hosszabb impulzusválasz is előfordulhat, akár 10 másodperc is, amely még nagyobb fokszámú szűrőt eredményez. A szűrők fokszáma csökkenthető az emberi hallás működését, korlátait figyelembe véve.

A digitális zengetés másik lehetősége az impulzusválasz mesterséges előállítása, szintézise. Mivel az impulzusválasz előző pontban említett részeinek, vagyis a korai és késői visszaverődések tulajdonságai eltérőek, az egyes részek szintézisére más-más eljárást szoktak alkalmazni. Ezek a módszerek nem konkrét, létező terem hangzását adják vissza, mint a konvolúciós módszer, de kisebb számításigényük mellett kellemes hangzást biztosítanak. Ezen módszerekből néhányat részletesebben bemutatok a 4.1 fejezetben.

3 A változtatható paraméterű zengető rendszerterve

Diplomamunkám keretében egy változtatható paraméterű zengetőt készítettem. A rendszer a korábban már nagyvonalakban áttekintett zengetést, vagy más néven reverb-et valósítja meg az impulzusválasz szintézisének módszerével. A zengető paramétereinek változtatása lehetővé teszi különböző térhatások létrehozását.
A zengető blokkvázlatát a 3. ábra mutatja be.



3. ábra A zengető blokkvázlata

A rendszer két egységből áll, az egyik a PC-n megvalósított felhasználói felület, a másik a DSP-n (Digital Signal Processor) futó zengető algoritmus.

A PC-n futó alkalmazás adja a felhasználói felületet, amelyen keresztül a zengetés paraméterei változtathatóak. Erre azért van szükség, mert a jelfeldolgozást végző egységen nem áll rendelkezésre semmiféle kezelőszerv, amellyel megoldható lenne a paraméterek változtatása. A PC bevonásának előnye, hogy bizonyos számítások így itt végezhetőek el. Kétfajta felület áll rendelkezésre. Az egyiken keresztül az egyik, későbbiekben bemutatásra kerülő impulzusválaszt szintetizáló szűrőhálózat paraméterei közvetlenül változtathatók meg. A másik felületen ugyanennek a szűrőhálózatnak a tulajdonságai változtathatók meg olyan paraméterek változtatása mellett, mint az utózengési idő, vagy a száraz és zengő hang aránya, amelyeket olyan felhasználó is könnyen tud értelmezni, aki nem ismeri a zengetést megvalósító algoritmust.

A zengető algoritmust digitális jelfeldolgozó processzor valósítja meg. A piacon jelfeldolgozó processzorok nagy választéka kapható, különböző tulajdonságokkal, így mindenki kiválaszthatja a feladathoz legmegfelelőbbet. A feladat megvalósításához a laborban rendelkezésre álló eszközök közül kellett választanom, vagyis az ADSP-2181,

ADSP-21061, ADSP-21065L EZ-Kit Lite fejlesztőkártyák és egy ADSP-21065L jelfeldolgozó processzort tartalmazó 8 csatornás jelfeldolgozó kártya (Bogár István munkája) közül.

Az alábbi táblázat a DSP-k kiválasztás szempontjából fontosabb tulajdonságait foglalja össze [2.].

Típus	Számábrázolás	Órajel frekvencia	On-cip RAM	Műveletvégzés
ADSP-2181	fixpontos	20MHz	80kByte	40 MIPS*
ADSP-21061	lebegőpontos	50MHz	1 Mbit	150 MFLOPS**
ADSP-21065L	lebegőpontos	66MHz	544kbit	198 MFLOPS

* Million Instructions Per Second

**Million Floating-Point Operations Per Second

A táblázat alapján a két lebegőpontos processzor alkalmazása előnyösebb, hiszen minden tekintetben jobb tulajdonságokkal rendelkeznek, mint az ADSP-2181. Az ADSP-21061 és az ADSP-21065L fejlesztőkártyákhoz soros porton keresztül lehet kapcsolódni. Mivel jelen esetben a DSP programozása és a paraméterek küldése is a PC felől történik, ezt a kétfajta kommunikációt kellene egy porton keresztül megoldani. A 8 csatornás jelfeldolgozó kártyán a soros port mellett rendelkezésre áll egy JTAG interface is, amelyen keresztül megoldható a processzor programozása, így a soros porton keresztül történhet a paraméterek kiküldése. Ennek a választásnak további előnye, hogy a 8 csatornás jelfeldolgozó kártyához készült egy általános szoftver, amely kezeli a hardver egységeket és a kommunikációs csatornákat, így többek között elvégzi a soros porton érkező üzenetek értelmezését is.

A választás egyik hátránya a rendelkezésre álló memória mérete, amely jelentősen korlátozza a megvalósítható zengetés maximális utózengési idejét. A másik nagy hátrány a mintavételi frekvenciák lehetséges értékei, azaz 8, 16, 32, 64 kHz. Az általam készített alkalmazás 16 kHz-en működik. Külső oszcillátor csatlakoztatásával elérhetők az előbbiektől eltérő mintavételi frekvenciák is, de amennyiben ez nagyobb, mint 16 kHz, akkor megnő a memória igény.

4 Elméleti háttér

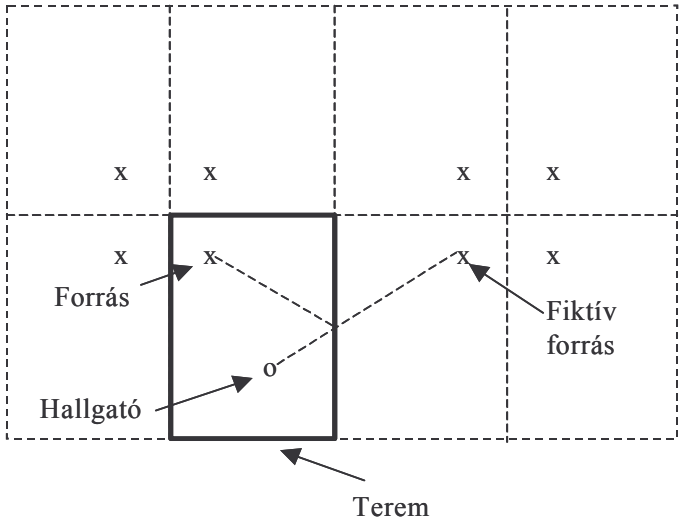
4.1 Zengető algoritmusok

Ebben a fejezetben áttekintem a 2.2 pontban említett, zengetésre alkalmazott impulzusválasz szintézis módszereket.

4.1.1 Korai visszaverődések

A korai visszaverődések szimulációjára alkalmas megoldás lehet, ha egy valódi impulzusválasz korai szakaszát használjuk fel a felvétellel történő konvolúcióhoz [3.]. A visszaverődések ilyenkor még nem olyan sűrűek, vagyis egy viszonylag kisebb fokszámú FIR szűrő elegendő a megvalósításhoz. Ha elég erőforrás áll rendelkezésre, akkor ezzel a módszerrel egy terem korai visszaverődéseinek pontos szimulációja valósítható meg.

Másik lehetőség a tükörforrások módszerének alkalmazása [4.]. A módszer lényegét két dimenzióban a 4. ábra mutatja.



4. ábra Tükörforrások módszere

A sok szakaszból álló visszaverődési útvonalakat egy egyenessé alakítjuk, és ennek megfelelő távolságban egy fiktív forrást helyezünk el a térben, mely a valódi szoba területén kívül helyezkedik el. A modellt úgy kezeljük, mintha mindegyik forrás

egyszerre adna. A forrásokhoz tartozó késleltetést a forrás és a hallgató távolsága adja meg. A forrásokról tudni kell, hogy hányadrendű visszaverődéshez tartoznak, hiszen minden falról történő visszaverődés valamekkora energia elnyelődésével jár. A levegő szintén elnyelőként viselkedik, nagyobb frekvencián az elnyelési tényezője nagyobb. A távolság függvényében tehát számítható a hang szintjének csökkenése. A forrásokhoz tartozó késleltetés és hangszint értékeket FIR szűrővé konvertálva megkapjuk a korai visszaverődéseket szimuláló szűrőt.

Egyszerű alaprajzú termekre, mint például a fenti ábrán látható téglalap alapú szobára, a számolás viszonylag egyszerű, bonyolultabb formájú termeknél a számolás is bonyolultabb és a források száma is nagyobb ütemben nő. Figyelembe véve a fül érzékenységéből adódó időbeli elfedéseket, a kapott FIR szűrő fokszáma csökkenthető [5.].

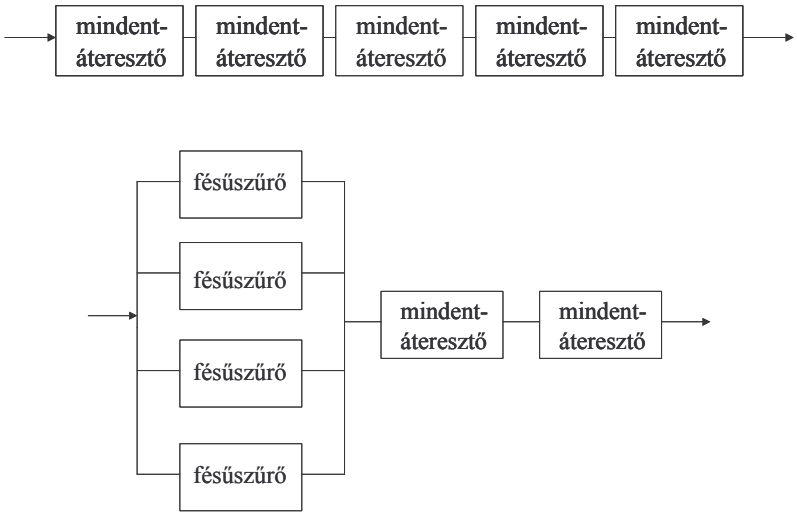
A tükörforrások módszerének hátránya, hogy nem veszi figyelembe a diffuzitást (a felületekről történő szórt visszaverődéseket, amely mértéke nagyobb frekvenciák felé egyre nő), valamint a diffrakciót (elhajlást). Mindezek ellenére elfogadható számításigény mellett elfogadható eredményt ad.

4.1.2 Késői visszaverődések

Késői visszaverődések szimulációjának egyik módszere a szűrőhálózatok használata. Ebben a fejezetben néhány szűrőhálózatot részletesebben bemutatok.

4.1.2.1 Schroeder-zengető

Az első diszkrét idejű jelfeldolgozáson alapuló mesterséges zengető Schroeder nevéhez fűződik [5.]. Az általa javasolt két elrendezést a késői visszaverődések generálásához a 5. ábra mutatja.



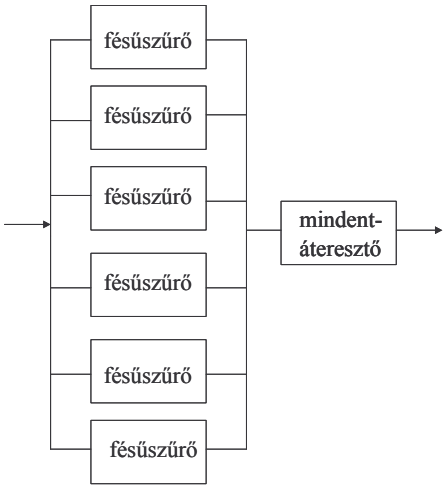
5. ábra Schroeder-zengetők felépítése

Az első, csak mindentáteresztő szűrőket tartalmazó struktúra előnye, hogy a szűrők frekvenciamenete egyenletes. A sorosan kapcsolódó egységek átviteli függvénye az egységek átviteli függvényének szorzata, ezért a struktúra egészének frekvenciamenete is egyenletes. Ez azonban csak hosszú távon teljesül, rövid, tranziens jellegű gerjesztéseknél elszíneződés tapasztalható, ami fémes hangzást eredményez.

A második struktúrában fésűszűrők szerepelnek. Ellentétben a digitális jelfeldolgozás terminológiájával, itt a visszacsatolt késleltetővonalat nevezik fésűszűrőnek. Mivel az általam olvasott cikkek nagy többsége ezt az elnevezést használja, ezért dolgozatomban továbbiakban fésűszűrő alatt a visszacsatolt késleltetővonalat értem. Ezeknek a szűrőknek a frekvenciaválasza nem egyenletes, de párhuzamos kapcsolásukkal, megfelelő paraméter-beállítások mellett elérhető a frekvenciatartománybeli csúcsok kellő sűrűsége. Ez a megoldás közelebb áll a helyiségek valódi átviteli függvényéhez, mint az egyenletes frekvenciamenet.

4.1.2.2 Moorer-zengető

Moorer a ’70-es végén egy az előzőhöz nagyon hasonló struktúrát javasolt, néhány új ötlettel kiegészítve [5.]. A 6.ábrán látható hálózat hat darab párhuzamos fésűszűrőt tartalmaz, visszacsatoló águkban egy-egy alulátereszt szűrővel, amelyek a levegő elnyelését szimbolizálják, ezeket követi egy mindentáteresztő struktúra.



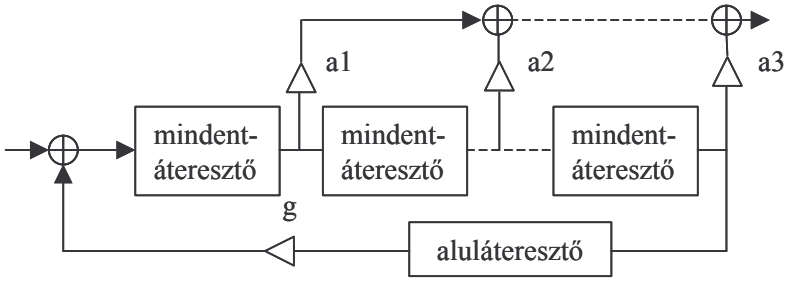
6. ábra Moorer-zengető felépítése

Schroederrel ellentétben, aki a közvetlen hangot vezette a diffúz visszaverődéseket szolgáltató hálózat bemenetére, Moorer a korai visszaverődések eredményét használta ugyanerre a célra, ezzel növelve a visszaverődések számát. A fésűszűrők megnövelt száma pedig a hosszabb utózengési idejű rendszerek jobb minőségű szimulációját teszi lehetővé.

Ezt a hálózatot részletesen is bemutatom a 6.1 fejezetben.

4.1.2.3 Gardner-zengető

Gardner a diffúz visszaverődések szimulációjához újdonságként bevezette az egymásba ágyazott mindentáteresztő szűrőket. Cikkében [4.] háromféle teremmérethez (kicsi, közepes, nagy) adott meg szűrőstruktúrákat ajánlott erősítési és késleltetés értékekkel. A struktúrák általános felépítése a 7.ábrán látható.



7. ábra Gardner-zengetők általános blokkvázlata

Az egymásba ágyazott mindentáteresztő szűrők használatának előnye, hogy mivel a belső szűrő által előállított visszaverődések ugyanebbe a szűrőbe vannak visszacsatolva, megvalósul a visszaverődések időbeli sűrűsödése, éppúgy, mint a valódi termek esetében.

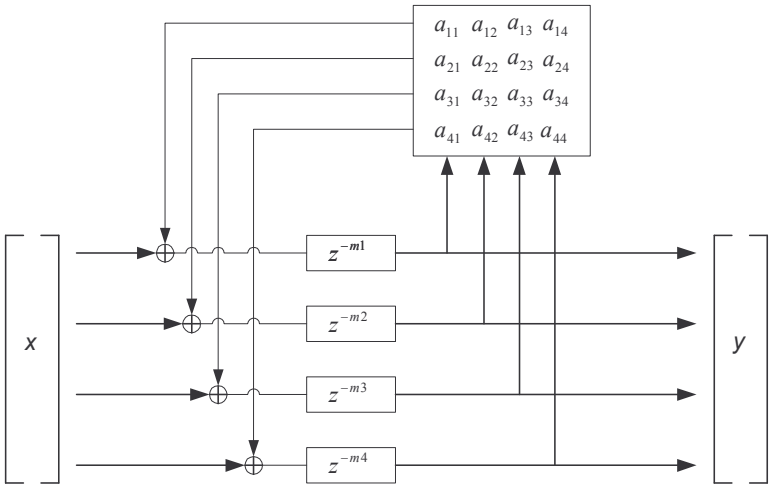
4.1.2.4 Dattoro-zengető

1997-ben Dattoro több más audio effekt mellett egy olyan zengető topológiáját is publikálta, amely a lemezes zengető szimulációjára alkalmas, és sztereo kimenetet állít elő [6.].

A késői visszaverődéseket előállító egység két nagy részre bontható fel. Az első a bemeneti késleltetést és aluláteresztő szűrőt követő négy mindentáteresztő szűrő blokkot tartalmazza, amelyek a bejövő jel dekorrelációjáért felelősek. A másik rész a tartály (tank), ez két szimmetrikus vonal keresztbe csatolását tartalmazza. A kimeneteket a tartály különböző pontjain történő kicsatolások súlyozott összege adja.

4.1.2.5 Feedback Delay Network (FDN)

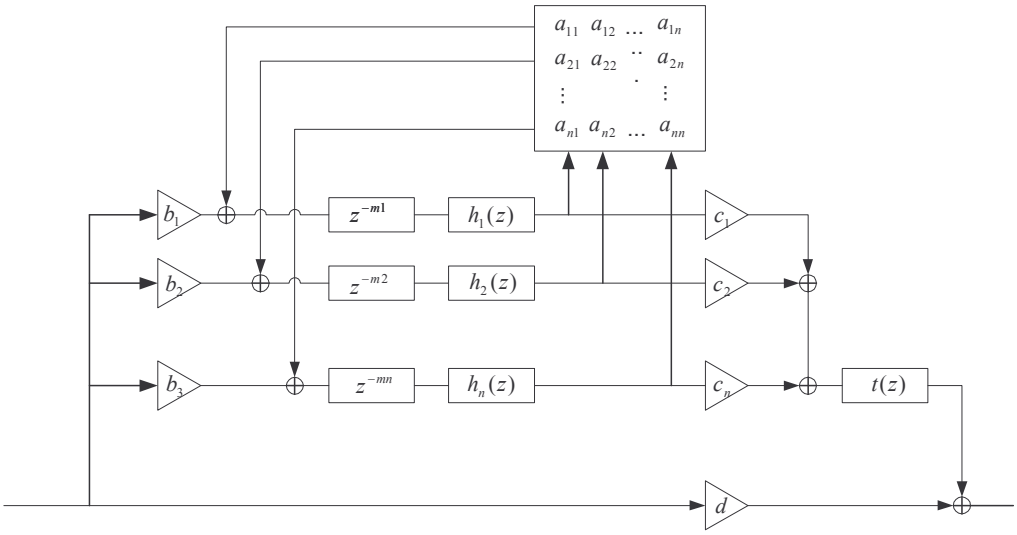
1982-ben Stautner és Puckette kifejlesztették a 8.ábrán látható négy csatornás zengetőt [1.].



8. ábra Stautner és Puckette zengetője

A hálózatban található visszacsatoló mátrix lehetővé teszi minden kimenet visszacsatolását minden bemenetre, ezzel mintegy általánosítva a Schroeder és Moorer által használt párhuzamos fésűszűrő struktúrát, amely egy diagonális visszacsatoló mátrixszal írható le. A rendszer akkor stabil, ha a visszacsatoló mátrix megadható egy egységmátrix és egy diagonális mátrix szorzataként, amelynek átlójában szereplő elemek nagysága egynél kisebb. Ezzel a módszerrel nagymértékben növelhető a visszaverődések sűrűsége, ami a párhuzamos fésűszűrők egyik kritikus pontja volt. A késleltetővonallal sorosan kapcsolt aluláteresztő szűrővel modellezhető a levegő elnyelése.

A 90-es évek elején Jot egy újabb zengető struktúrával állt elő [7.], amellyel tetszőleges utózengési idő és frekvenciaválasz előállítható tetszőleges idő és frekvencia sűrűség mellett. Ahogy a 9.ábrán látható, a hálózat két részre bontható. Az első rész egy aluláteresztő szűrőkkel kiegészített referenciaszűrő, amelyet a második rész kompenzáló szűrője követ (*t(z)*).



9. ábra Feedback Delay Network

A referenciaszűrőt leíró egyenletek:

$$Y(z) = c^T S(z) + dX(z) \qquad (4.1)$$

$$S(z) = D(z)[AS(z) + bX(z)] \qquad (4.2)$$

Az *A* mátrix többféleképpen megválasztható. Diagonális esetben Schroeder és Moorer modelljét kapjuk vissza, mert ilyenkor minden szűrő bemenetére a saját

kimenete csatolódik vissza. Mindentáteresztő szűrők soros kombinációja megfeleltethető egy háromszögmátrixnak. Amennyiben A-t egységmátrixnak választjuk, akkor a rendszer pólusai az egységkörön helyezkednek el. A késleltetővonalak mögé beszúrt k_i erősítési tényezővel elérhető, hogy a pólusok abszolút értéke továbbra is egyenlő legyen, de 1-nél kisebb, amennyiben k_i-t a 4.3 képletnek megfelelően választjuk meg, ahol α a pólusok új amplitúdójának nagysága, m_i pedig a késleltetővonalak hossza. Ezzel elérhető az exponenciálisan lecsengő impulzusválasz.

$$k_i = \alpha^{m_i} \qquad (4.3)$$

Ahhoz, hogy az utózengési idő a valósághoz hűen frekvenciafüggő legyen, egy szűrőt kell elhelyezni a késleltető vonalak mögé. Ezután a pólusok már nem egy körön helyezkednek el, hanem egy nagyjából folytonosnak tekinthető görbén. Az elnyelést szintetizáló aluláteresztő szűrők hatására az egész hálózat átviteli függvénye már nem egyenletes, hanem hangszínező hatású, ami ellensúlyozható a rendszer második egységével, a kompenzáló szűrővel.

4.2 Impulzusválasz mérése

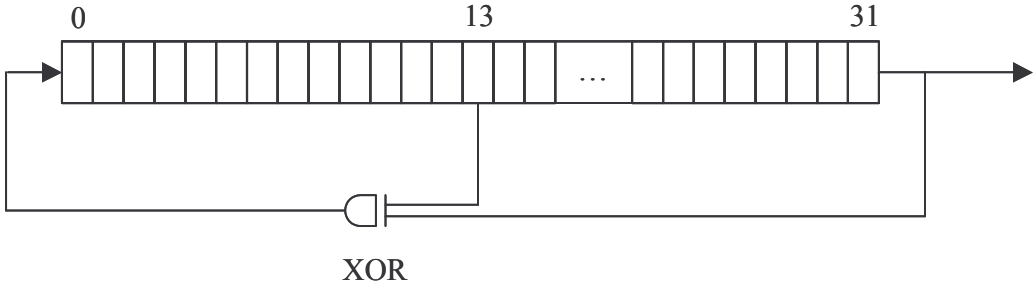
4.2.1 Mérési módszerek

Impulzusválasz mérése két pont között történik, egyik pontban helyezkedik el az adó, másikban a vevő. A gyakorlatban többféle gerjesztő jelet alkalmaznak [3.].

Egyik csoportba tartoznak az impulzus jellegű gerjesztések, mint a taps, a zacskópukkasztás és a pisztolylövés. Ezek a módszerek nem igénylenek bonyolult felszerelést, mindössze egy rögzítőt, amely a felveszi a gerjesztésre adott választ. A felvett anyag jelen esetben közvetlenül az impulzusválasz, ezért további számításokra sincsen szükség. Ilyen jellegű gerjesztések esetén a mérések jel/zaj viszonya elég rossz, mert a gerjesztőjelek energiája rövid időintervallumra koncentrálódik. További hátrány, hogy a mérés nem megismételhető, kétszer nem adja ugyanazt az eredményt.

A gerjesztések másik csoportja időben folytonos jeleket alkalmaz. Ezek egy része nem periodikus, mint a fehérzaj és a rózsazaj, másik részük periodikusan alkalmazható, mint az exponenciálisan vagy lineárisan (TDS – Time Delay Spectrometry) sweepelő szinusz, az álvéletlen zaj és az MLS (maximal length

sequence). Ez utóbbi egyesek és nullák véletlen bináris sorozata, amelynek előállítására a 10.ábrán látható visszacsatolt shift regisztert szokták használni. *N* hosszúságú regiszter esetén a sorozat periódushossza: 2^{N-1}-1, a periódusideje pedig $(2^{N-1}-1)/f_s$, ahol f_s a mintavételi frekvencia.



10. ábra MLS jelet előállító visszacsatolt shift regiszter

A periodikus jelek előnye, hogy a periódusok eredményét átlagolva javítható a mérések jel/zaj viszonya. Az ilyen típusú jeleknél fontos, hogy a mérőjel periódusideje legyen hosszabb, mint az impulzusválasz várható hossza, így elkerülhető az átlapolódás. Mivel ezek a jelek determinisztikusak, ezért a velük végzett mérések megismételhetők, vagyis ugyanolyan körülmények között (azonos mérési elrendezés, mérőeszközök, háttérzaj, hőmérséklet, páratartalom, stb.) azonos eredményt szolgáltatnak.

Mindegyik mérési módszerre jellemző, hogy lineáris, invariáns rendszer esetén ad pontos eredményt Ezért szükséges a háttérzaj (pl. légkondicionáló) minimálisra csökkentése, és a mozgások kerülése a mérések elvégzése alatt.

A mérések során szükség van mérőmikrofonra, valamint folytonos jelek esetén hangsugárzóra. Ilyenkor a terem átvitele mellett az eszközök átvitele is belekerül a mérésbe, az átviteli függvények ismeretében azonban utólag kompenzálni lehet a mért eredményeket.

Léteznek az impulzusválasz és egyéb jellemzők mérésére alkalmas mérőrendszerek is[8.], mint a MLSSA, vagy az AURORA. Az MLSSA hardver rendszer, amely MLS gerjesztőjelet használ, majd a gerjesztés és a válasz dekonvolúciójából állítja elő az impulzusválaszt. Az AURORA ezzel szemben szoftver rendszer, vagyis szoftveres úton állítja elő a gerjesztést és rögzíti a választ, mindezt egy hagyományos PC hangkártyáját felhasználva.

4.2.2 Mérés

4.2.2.1 Gerjesztés

Az általam elvégzett mérések során álvéletlen gerjesztést használtam, melyet egy Matlab-ban írt program segítségével generáltam. A méréshez használt 8 csatornás rögzítő csatornáit kimenetként és bemenetként is lehet egyidőben használni. Ezért az egyik csatornát kimenetként alkalmazva Matlab alatt generált álvéletlen zajt játszott le, miközben egy másik csatorna rögzítette a mérőmikrofon erősített jelét.

Álvéletlen zaj generálására több módszer is ismert. Az egyik az előző pontban már említett MLS-t előállító visszacsatolt shiftregiszter. Másik megoldás a lineáris kongruencia módszer, amely egész számokból álló sorozatot generál a 4.4 algoritmussal:

$$X_{n+1} = (A \cdot X_n + B) \bmod N \,,\; X_0 = adott \,,\; n = 0,1,2... \qquad (4.4)$$

A, B és N megválasztása alapvetően befolyásolja a keletkezett számsorozat tulajdonságait, például eloszlását. X_0 megválasztása tetszőleges, az álvéletlen sorozat kezdőértékét adja.

Termek méréséhez használt álvéletlen jel periódusidejének hosszabbnak kell lennie a terem impulzusválaszánál, amely kis méretű, átlagos elnyelő felületekkel borított termek esetén kisebb, mint 1 másodperc, nagyobb termek esetén 2-5, akár 10 másodperc is lehet. A általam végzett mérések esetén a termek előadók voltak, ahol fontos a beszédérthetőség, ezért legfeljebb 0.5-1 másodperc hosszúságú impulzusválaszt vártunk. Ezért olyan álvéletlen zajra volt szükség, amelynek periódusideje 2 másodperc körüli.

Egy lehetséges megoldás 16 bites álvéletlen számokból álló zaj generálása. Az ehhez tartozó algoritmust a 4.5 egyenlet írja le.

$$seed = (181 \cdot seed + 359) \bmod 2\text{^}16 \qquad (4.5)$$

Ennek a módszernek előnye, hogy egyenletes eloszlású sorozatot generál, egy periódus alatt az összes lehetséges számot előállítja, mindegyiket pontosan egyszer. Hátránya, hogy a periódusideje 44100 Hz-es mintavételi frekvencia mellett kb. 1.5 másodperc, ez nagyobb termek esetén nem biztos, hogy elegendő. Azonban a megmért

termekhez azonban megfelelőnek bizonyult, ezért a mérések során ezt a módszert használtam.

Jóval nagyobb termekhez használható a 32 bites álvéletlen számokat előállító 4.6 algoritmus.

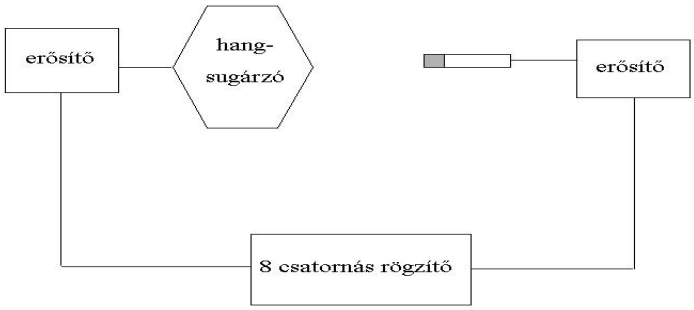
$$seed = (69069 \cdot seed + 1) \bmod 2\text{^}32 \qquad (4.6)$$

Ezzel a módszerrel generált sorozat spektruma egyenletes, periódusideje kb. 97391 másodperc, vagyis kb. 27,05 óra. Ez azonban túl hosszú, a pár perces mérés szempontjából tulajdonképpen teljesen véletlen fehérzajnak tekinthető. Csonkolással rövidíthető a periódusidő, de csonkolás után ellenőrizendő a spektrum változása. Minél nagyobb a csonkolás mértéke, vagyis minél hamarabb reseteljük a *seed* értéket, annál kevésbé használható a keletkező jel fehérzajként.

4.2.2.2 Mérési elrendezés

A méréseket a Budapesti Műszaki és Gazdaságtudományi Egyetem Informatika épületének IB027 és IB028-as termében végeztem.

A méréshez a 11.ábrán látható elrendezést használtam, a méréshez használt eszközöket a függelék 10.1 pontja tartalmazza. A 8 csatornás rögzítőn elhelyezett előző fejezetben leírt álvéletlen jelet tartalmazó .wav file szolgált gerjesztőjelként, amely erősítőn keresztül került a hangsugárzóra. A kiválasztott mérési pontokban elhelyezett mérőmikrofon által érzékelt jel erősítőn keresztül jutott a 8 csatornás rögzítőre és került felvételre. A mérési eredményeket Matlab-ban dolgoztam fel. A mérési pontokat, a méréshez használt eszközök felsorolását és a mérés eredményeket a 7. fejezet tartalmazza.



11. ábra Mérési elrendezés

4.2.2.3 A mérési eredmények feldolgozása

A mérések során rögzített anyagot Matlab segítségével dolgoztam fel.

A feldolgozás módját pár mondatban összefoglalom. A felvett gerjesztést és választ darabokra bontottam, méghozzá az álvéletlen gerjesztés periódushosszával megegyező hosszúságú darabokra. Ezeket egyenként Fourier transzformáltam, képeztem a gerjesztés és a válasz transzformáltjának hányadosát, majd ezeket a hányadosokat átlagoltam. Végül az átlagos átvitelt inverz Fourier transzformáltam, így állt elő az impulzusválasz. Az így kapott eredményeket és értékelésüket a 7.1.2 fejezet tartalmazza.

5 A változtatható paraméterű zengető hardver leírása

A fejezet 5.1 pontja tartalmazza a zengetőhöz használt 8-csatornás jelfeldolgozó kártya rövid leírását, az 5.2 pont pedig a kártyán található jelfeldolgozó processzor tulajdonságainak, működésének részletes bemutatását.

5.1 A nyolc-csatornás jelfeldolgozó kártya

A 3. fejezetben leírtaknak megfelelően a zengető algoritmust a nyolc-csatornás jelfeldolgozó kártyán valósítottam meg [9.].

A kártya hardver egységei:

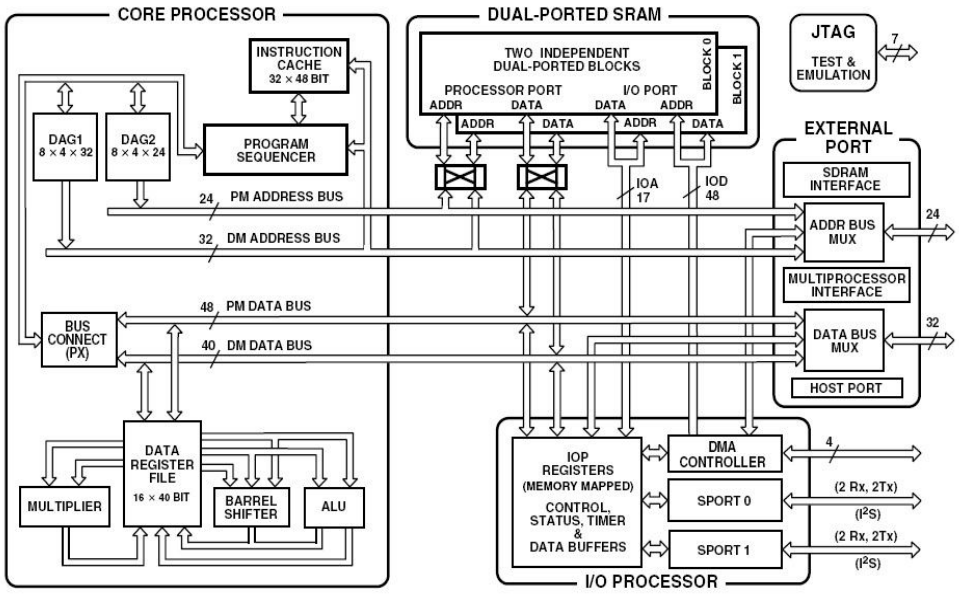
- Jelfeldolgozó processzor: ADSP21065L, részletesebb leírása az 5.2 pontban
- 8 db analóg bemenet: hozzájuk tartozik 4 AD73322 típusú codec, amelyek szigma-delta A/D átalakítást végeznek, 16 bites felbontás mellett
- 8 darab analóg kimenet, szintén 4 darab AD73322 codec-kel, szigma-delta D/A átalakítással és 16 bites felbontással
- Memória: 256 Mbyte SDRAM
- FPGA: XC2S150, a jelfeldolgozásban közvetlenül nem vesz részt. Ez az egység oldja meg az SDRAM kezelését, digitális I/O megvalósítást, a kivezérlésjelző LED-ek vezérlését, kapcsolatban áll az UART vezérlővel is.
- Tápegység

A kártyán található nagyméretű külső memória lehetővé teszi a jelfeldolgozó funkción kívül a kártya adatgyűjtőként történő használatát is. Az általam megvalósítani kívánt változtatható paraméterű zengető számára szintén hasznos lehetne a külső memória a DSP on-chip memóriájának korlátozott mérete miatt, a hozzáférés módja azonban kizárja ennek az egységnek a felhasználását. A memóriából egy művelet elvégzésekor 8 darab egymást követő 16 bites szó olvasható ki, vagy írható be. Egy mintavételi ciklus alatt, a mintavételi frekvenciától függően mindössze néhány írás és olvasás végezhető el, ami nem elegendő a zengető funkció megvalósításához.

5.2 Az ADSP 21065L jelfeldolgozó processzor

5.2.1 Általános leírás

A kártyán található digitális jelfeldolgozó processzor az Analog Devices SHARC termékcsaládjába tartozik [10.]. A SHARC rövidítés kifejtve Super Harvard Architecture Computer, amely továbbfejlesztett Harvard architektúrát jelent. Az egyszerű Harvard architektúrához hasonlóan az adatmemória és a programmemória itt is elkülönül, de emellett a memóriát a program- és adatbuszokon kívül két, az I/O processzorhoz csatlakozó busz is eléri. A 32 bites processzor fő felhasználási területei a kommunikációs, audio, automatizálási és ipari alkalmazások. Felépítését a 12. ábra mutatja.



12. ábra Az ADSP-21065L processzor

A processzormag 3 műveletvégző egységet tartalmaz, egy ALU-t (Arithmetical and Logical Unit), egy szorzót, és egy shiftert. Ezek az egységek 32 bites fixpontos, 32 bites lebegőpontos és kiterjesztett pontosságú 40 bites adatok kezelésére alkalmasak. Itt található a 32 általános célú, 40 bit szélességű regiszter, amelyek két, 16 regiszterből álló blokkot alkotnak. Ez a konstrukció a gyors kontextusváltást segíti elő, amely ezzel a felépítéssel a regiszterek tartalmának mentése nélkül is megoldható, mivel egyidőben

csak az egyik blokk lehet aktív. A memória címek előállítását két Data Address Generator (DAG) és egy Program Sequencer végzi. Az adatok címének előállításáért felelős DAG-ok egyike 32 bites címeket állít elő az adat memóriához, a másik 24 bites címeket a program memóriához. Ezáltal egy ciklusban két operandus érhető el. A Program Sequencer-t egy utasítás cache egészíti ki, amely lehetővé teszi egy utasítás és két adat elérését, ezzel meggyorsítva az olyan ciklikus műveleteket, mint a digitális szűrők szorzás-összeadásai (MAC – Multiply and Accumulute), vagy az FFT pillangó algoritmusa.

A processzor 544 kbit konfigurálható on-chip SRAM-mal rendelkezik, az ehhez kapcsolódó tudnivalókat külön pontban (5.2.2) foglalom össze, mert az alkalmazás szempontjából az egyik kritikus tényező a memória volt.

A külső port lehetővé teszi külső memóriaegység és perifériák illesztését. A Host Processor Interface a standard mikroprocesszor buszokhoz való könnyű kapcsolódási valósítja meg.

A processzor két soros porttal rendelkezik, ezzel olcsó és általános csatlakozási lehetőséget biztosítva sokféle eszközhöz.

A DMA (Direct Memory Access) vezérlő nagy mennyiségű adat mozgatását teszi lehetővé a processzor belső memóriája és a külső memória, perifériák, vagy host processzor között a processzor magjának megkerülésével és ezáltal nagy sebességgel.

A gyártó a processzor fejlesztéséhez szoftveres és hardveres támogatást is nyújt. A VisualDSP környezet fejlesztésre és debuggolásra egyaránt alkalmas. Kiegészítve az IEEE 1149.1 JTAG interface tesztelési és emulálási lehetőségével hatékony fejlesztést tesz lehetővé.

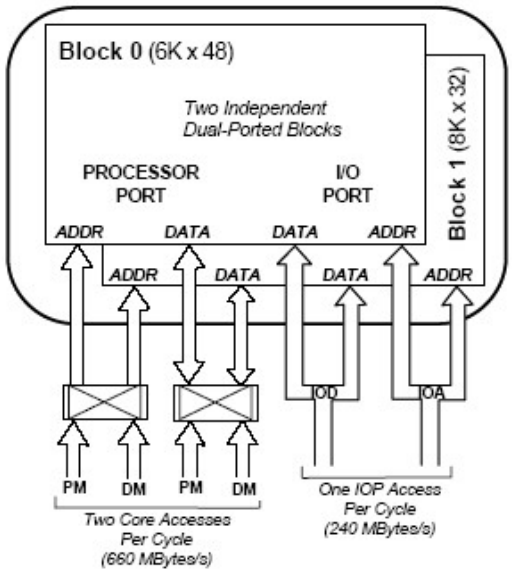
5.2.2 Memória

A memória felépítését és kezelését azért tárgyalom részletesebben, mert az általam DSP-re írt alkalmazásnak ez volt megvalósíthatósági szempontból a legkritikusabb része. A zengető nyolc különböző hosszúságú késleltetőt tartalmazó szűrőből épül fel, ez nagy mennyiségű adat tárolását igényli. Az adatok mennyiségét a 16 kHz-re választott mintavételi frekvencia csökkenti, további könnyebbséget jelent a

DSP-n rendelkezésre álló Short Word üzemmód, amely lehetővé teszi 16 bites értékek kezelését, ezáltal növelve az eltárolható adatmennyiséget.

A memória különböző formátumú adatokat képes kezelni, 48 bites utasításokat, 40 bites kiterjesztett pontosságú lebegőpontos, 32 bites lebegőpontos, valamint 16 bites short word értékeket.

A chip-en 544 kbit konfigurálható SRAM áll rendelkezésre, amely a 13.ábrán látható két partícióból áll.



13. ábra Az ADSP-21065L memóriájának felépítése

A Block0 288 kbites, fizikailag darab 2k * 16 bit szélességű oszlopból áll, míg a Block1 256 kbites, nyolc darab az előzővel megegyező felépítésű oszlop alkotja. Mindkét blokkhoz egy PM busz, egy DM busz és egy I/O busz kapcsolódik. Ennek megfelelően a processzor magja egy ciklus alatt két adathoz fér hozzá, míg az I/O processzor egyhez. Vannak tipikus DSP alkalmazások, mint például a digitális szűrés és az FFT, amelyeknek egy utasítás végrehajtásához két adatra van szükségük a memóriából. Az adatokhoz való hozzáférés akkor maximális hatékonyságú, ha ez egy ciklus alatt teljesíthető. Ennek feltétele, hogy a két adat külön blokkban helyezkedjen el, az egyiket a DM, a másikat a PM buszon keresztül lehessen elérni, valamint a DM busz által elért adat az utasítástól különböző memóriablokkban helyezkedjen el.

A memória mindkét blokkja minden korábban felsorolt adattípus tárolására alkalmas.

A 16 bites értékek egy oszlop szélességben elférnek. Címük előállítható a 32 bites memóriacím balra shiftelésével. MSW (Most Significant Word) elérése esetén a cím utolsó bitje 1, LSW(Least Significant Word) esetén 0. A memória partícionálásakor is beállítható az adott memóriaterület short word kezelése, így könnyen kialakítható cirkuláris buffer. A 32 és 48 bites szavak használata egy blokkon belül nehézkesebb, hiszen a 48 bites adatok 3 oszlopot igényelnek, a 32 bitesek ezzel szemben csak kettőt.

A következő szabályokat betartva elkerülhető az átlapolódás:

- Az utasítások tárolását a blokk legalacsonyabb címén kell kezdeni
- Az adatok tárolását páros számú oszlopon kell kezdeni
- Minden adat magasabb címen helyezkedjen el, mint az utasítások.

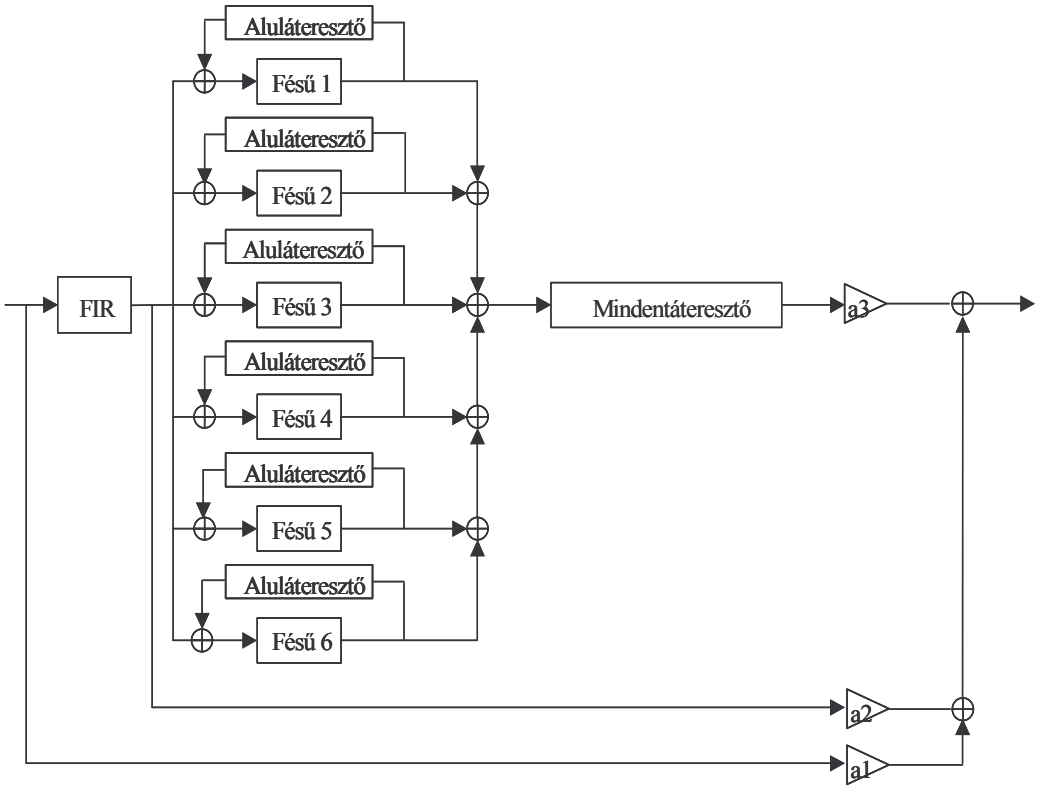
6 A változtatható paraméterű zengető szoftver leírása

A korábban felsorolt struktúrák közül Moorer zengetőjét valósítottam meg DSP-n. Ez a hálózat, egyszerűsége mellett, [11.] alapján Matlab-ban végzett korábbi kísérleteim szerint viszonylag jó hangzás elérésére alkalmas. Az algoritmus működését, a hálózatot felépítő egységeket és azok tulajdonságait a 6.1 pontban foglalom össze. A változtatható paraméterű zengető két részre bontható, PC-n és DSP-n futóra, melyek között soros porton zajlik a kommunikáció. A Java-ban készült, PC-n futó alkalmazás részletes leírását a 6.2 fejezet tartalmazza, feladatai röviden összefoglalva a felhasználói felület létrehozása és kezelése, az üzenetek generálása a felületen beállított paramétereknek megfelelően, valamint az üzenetek elküldése soros porton keresztül. A 6.3 pontban foglalom össze a DSP feladatait, a jelfeldolgozást, vagyis magát a zengető algoritmust és a DSP oldali kommunikációt.

6.1 A zengető algoritmus részletei

Moorer zengetőjének diffúz visszaverődéseket előállító része már szerepelt a 4.1.2.2 pontban. A 14.ábrán a 8 szűrőből álló teljes zengető struktúra látható.

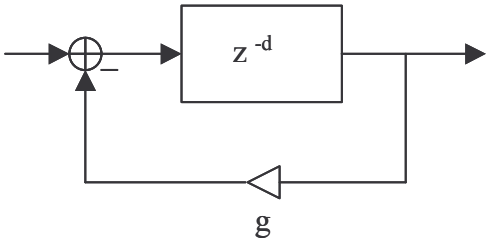
A korai visszaverődéseket FIR szűrő valósítja meg. Moorer 7 és 18 együtthatós szűrőket ajánlott, ezek a kis számok azzal magyarázhatók, hogy a cikk a 60-as évekből származik, amikor még nem álltak rendelkezésre olyan nagy teljesítményű és memóriájú jelfeldolgozásra alkalmas eszközök, mint manapság. Az általam készített változtatható paraméterű zengető elinduláskor a 18 együtthatós szűrőt alkalmazza a cikkben szereplő ajánlott paraméterekkel, amelyeket a függelék 1. táblázata tartalmazza.



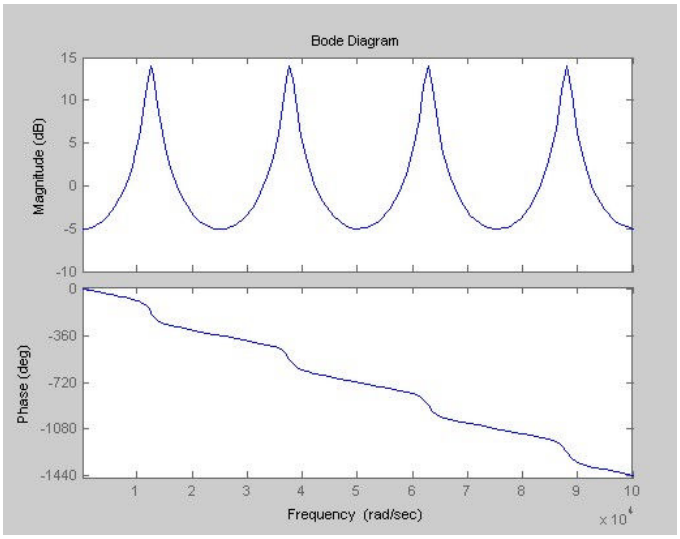
14. ábra Moorer teljes zengetőjének blokkvázlata

A késői visszaverődéseket a 4.1.2.2 pontban már bemutatott hat fésűszűrőből és egy mindentáteresztő szűrőből álló struktúra valósítja meg. A fésűszűrő általános jellemzőit, felépítését, impulzusválaszát, átviteli függvényét, pólus-zérus képét a 15. ábra foglalja össze, átviteli függvényét a 6.1 képlet írja le.

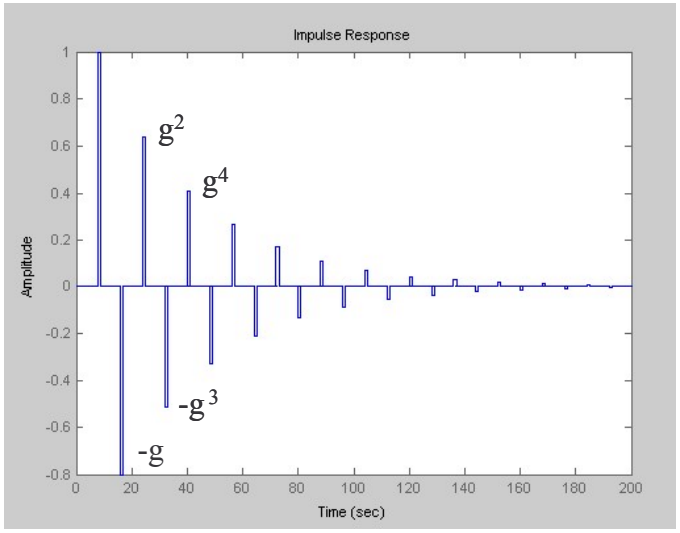
$$H(z) = \frac{z^{-d}}{1 + g \cdot z^{-d}} \qquad [6.1]$$



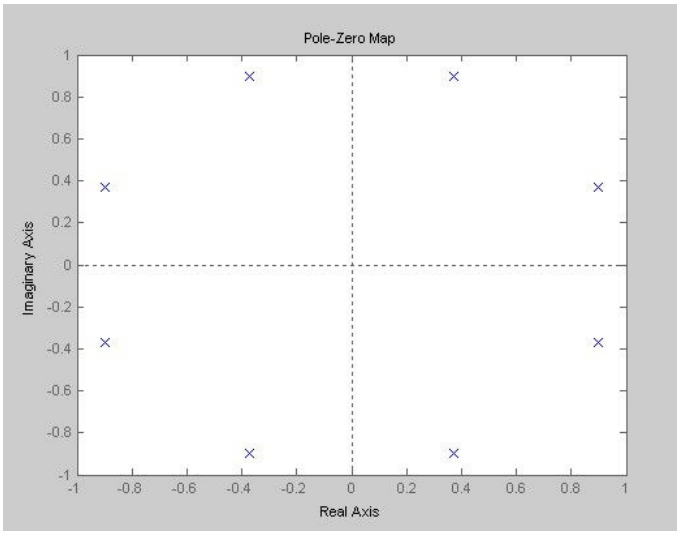
(a)



(b)



(c)



(d)

15. ábra Fésűszűrő struktúra (a), átviteli függvény (b), impulzusválasz (c), pólus-zérus kép (d). Az ábrákhoz tartozó szűrőparaméterek: g=0,8, d=8, fs=32kHz.

A fésűszűrőhöz tartozó utózengési idő meghatározható a hozzá tartozó Energy Decay Curve-ből, amelynek pontjai diszkrét idejű esetre a 6.2 képlettel határozhatók meg, ahol h_j az impulzusválasz j-ik pontjának amplitúdóját jelenti, ami fésűszűrő esetén g^j. Az összeg tagjai tehát mértani sort alkotnak.

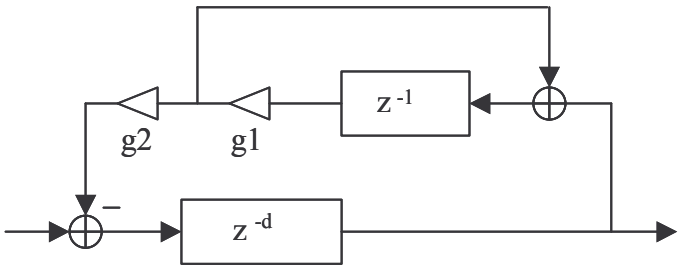
$$EDC(i) = \sum_{j=i}^{\infty} h_j^2 \qquad [6.2]$$

Ebből levezethető az EDC -60 dB-es pontjához tartozó, vagyis az utózengési időt megadó 6.3 képlet, amelyben g_i a szűrő erősítése, d_i a késleltetése, T pedig a mintavételi idő. Az erősítés állandón tartása mellett, a késleltetést növelve, vagy a késleltetést állandón tartva az erősítés növelve az utózengési idő nő.

$$T_{60} = \frac{-60(d_i T)}{20\log_{10}(g_i)} \qquad [6.3]$$

Az erősítés és késleltetés értékeket jól megválasztva beállítható a fésűszűrőhöz tartozó kívánt utózengési idő. A „jó megválasztás” azonban nem egyszerű. Rövid késleltetési idő mellett a visszaverődések száma gyorsan nő, de a frekvenciamenet csúcsai egymástól nagy távolságban lesznek. Ez azt jelenti, hogy a csúcsokhoz tartozó frekvenciák hosszú ideig zengnek, a többi frekvencián viszont gyors lecsengés tapasztalható. A késleltetés növelésével növelhető a frekvenciacsúcsok sűrűsége, de csökken a visszaverődések sűrűsége.

Moorer modelljében a 16.ábrán látható módon, a fésűszűrők visszacsatoló ága egy aluláteresztő szűrőt tartalmaz, amely a levegő elnyelését szimulálja.



16. ábra Fésűszűrő struktúra visszacsatoló ágában aluláteresztő szűrővel

Az alap fésű struktúra, melynek visszacsatoló ágában egyetlen erősítés van, akkor stabil, ha az erősítés abszolút értéke 1-nél kisebb. Ilyenkor a rendszer pólusai az egységkörön belül helyezkednek el. A visszacsatoló ágban elhelyezett T(z) átvitellel

rendelkező szűrő a 6.4 egyenlet szerint módosítja az átviteli függvényt. Ebben az esetben a stabilitás kritériuma $|g_2 \cdot T_{\max}| < 1$.

$$H(z) = \frac{z^{-d}}{1 - g_2 \cdot T(z) \cdot z^{-d}} \qquad [6.4]$$

$$T(z) = \frac{1}{1 - g_1 \cdot z^{-1}} \qquad [6.5]$$

Aluláteresztő szűrő esetén az átvitelt a 6.5 egyenlet adja meg, amelynek maximuma $1/(1-g_1)$. A stabilitási kritérium tehát ebben az esetben $g_2/(1-g_1) < 1$, tovább alakítva $g_1 + g_2 < 1$. Moorer bevezetett egy minden fésűszűrőre vonatkozó g erősítési értéket, amely a cikkben megadott ajánlott fésűszűrő késleltetések és aluláteresztő erősítések használata esetén 6.6 egyenlet szerint meghatározza fésűszűrő-mindentáteresztő rendszer utózengési idejét, 6.7 szerint kapcsolatot teremt g_1 és g_2 értékek között, valamint teljesül rá, hogy $0 < g < 1$.

$$g = 1 - \frac{0{,}366}{T_{60}} \qquad [6.6]$$

$$g_2 = g \cdot (1 - g_1) \qquad [6.7]$$

Egy fésűszűrő önmagában még nem lenne elegendő jó minőségű zengetéshez. Sem az általa létrehozott módussűrűség, sem a visszaverődések sűrűsége nem elég nagy. Több fésűszűrőt felhasználva a helyzet jelentősen javítható. Soros kapcsolásuk nem előnyös, mert átviteli függvényük ebben az esetben összeszorzódik. Párhuzamos kapcsolásukkal azonban növelhető a frekvenciatartomány csúcsainak száma és sűrűsége. A szűrők késleltetéseit relatív prímnek választva az egyes szűrőkhöz tartozó módusok mind eltérőek, így jobban lefedik a frekvenciatartományt, mint véletlenszerűen megválasztott késleltetés értékek esetén. A párhuzamos kapcsolás időtartományban is előnyös, mert az egyes szűrők kimenetei, vagyis a keletkező visszaverődések így csak a szűrő késleltetések legkisebb közös többszöröseinek értéke után esnek egybe.

A párhuzamos fésűszűrő struktúra módussűrűségét adja meg a 6.8 képlet, ahol τ_i az i-edik szűrőhöz tartozó késleltetés másodpercben, τ az átlagos késleltetés, N pedig a fésűszűrők száma. A módussűrűség tehát állandó, szemben a valódi helyiségekkel, ahol a frekvencia négyzetével arányosan nő. A visszaverődések sűrűsége a 6.9 képlettel

számolható, értéke szintén állandó, míg normál helyiségekben az idővel négyzetesen arányos. Adott D_e és D_m értékekhez 6.10-ből és 6.11-ből kiszámítható a szükséges szűrők száma, és késleltető vonaluk hossza.

$$D_m = \sum_{i=0}^{N-1} \tau_i = N \cdot \tau \qquad [6.8]$$

$$D_e = \sum_{i=0}^{N-1} \frac{1}{\tau_i} = N \cdot \frac{1}{\tau} \qquad [6.9]$$

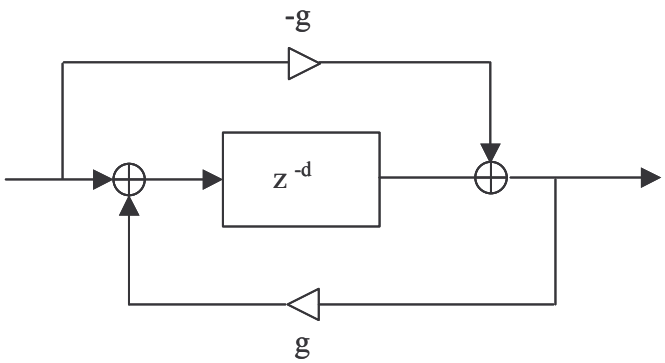
$$N = \sqrt{D_m D_e} \qquad [6.10]$$

$$\tau = \sqrt{\frac{D_m}{D_e}} \qquad [6.11]$$

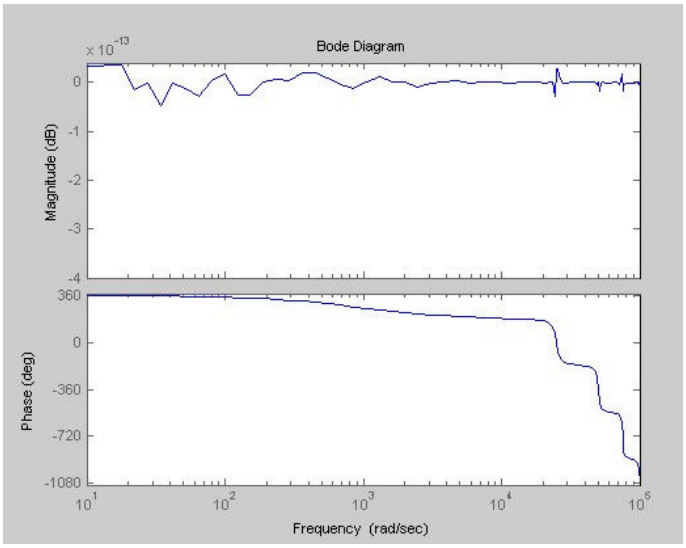
Schroeder D_e = 1000, D_m = 0.15 értékeket már elegendőnek talált, amelyhez 12 fésűszűrőre lenne szükség, átlagosan 12 ms-os késleltetésekkel [7.]. A visszaverődések sűrűségére ajánlott érték a későbbi tapasztalatok alapján túl alacsony, 10000 visszaverődés viszont már nagyjából elegendő. Így azonban jelentősen nő a szükséges fésűszűrők száma.

Moorer hálózatában a diffúz visszaverődéseket létrehozó egység egy mindentáteresztő szűrőt is tartalmaz. Ennek átvitelét a 6.12 egyenlet írja le, struktúráját, impulzusválaszát, átviteli függvényét és pólus-zérus képét a 17. ábra mutatja.

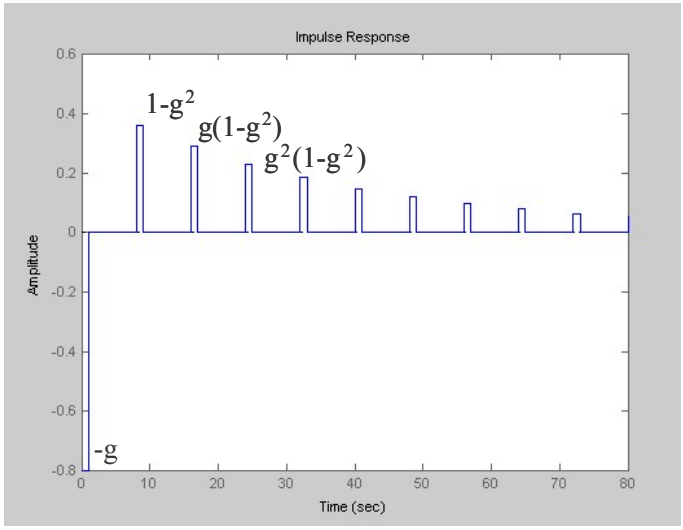
$$H(z) = \frac{-g + z^{-d}}{1 - g \cdot z^{-d}} \qquad [6.12]$$



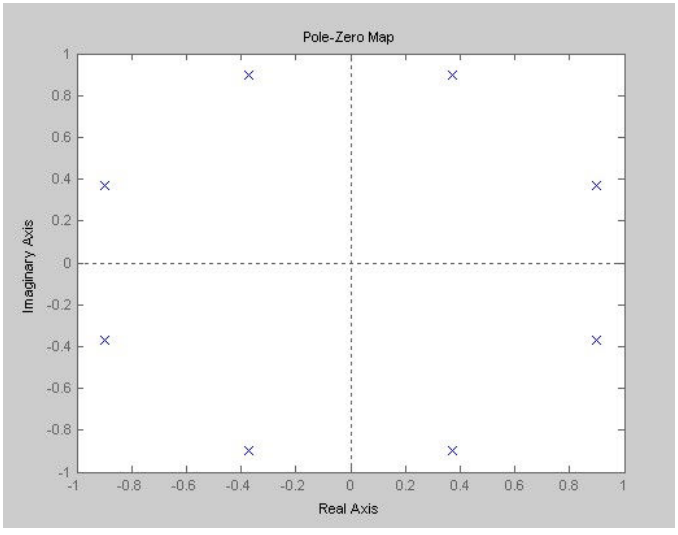
(a)



(b)



(c)



(d)

17. ábra Mindentáteresztő struktúra (a), átviteli függvény (b), impulzusválasz (c), pólus-zérus kép (d). Az ábrákhoz tartozó szűrőparaméterek: g=0,8, d=8, fs=32kHz.

A mindentáteresztő struktúra nagy előnye frekvenciatartománybeli egységnyi átvitele, amely lehetővé teszi a színezésmentes átvitelt, ami azonban tranziens gerjesztések esetén torzulhat.

A mindentáteresztő struktúra impulzusválaszának tagjai az első kivételével mértani sorozatot alkotnak, összegük megadható, így a szűrőhöz tartozó utózengési idő a 6.13 képletben látható zárt formában megadható.

$$T_{60} = \frac{-60 d_i T}{20 \cdot \log_{10}((1-g^2)g)} \qquad [6.13]$$

A fent bemutatott szűrőket kombinálva kihasználható azok előnyös tulajdonsága és kompenzálhatók hátrányaik, így született meg a hat fésűszűrőből és egy mindentáteresztőből felépülő rendszer. A szűrőkhöz Moorer által javasolt paramétereket a függelék 2. táblázata tartalmazza.

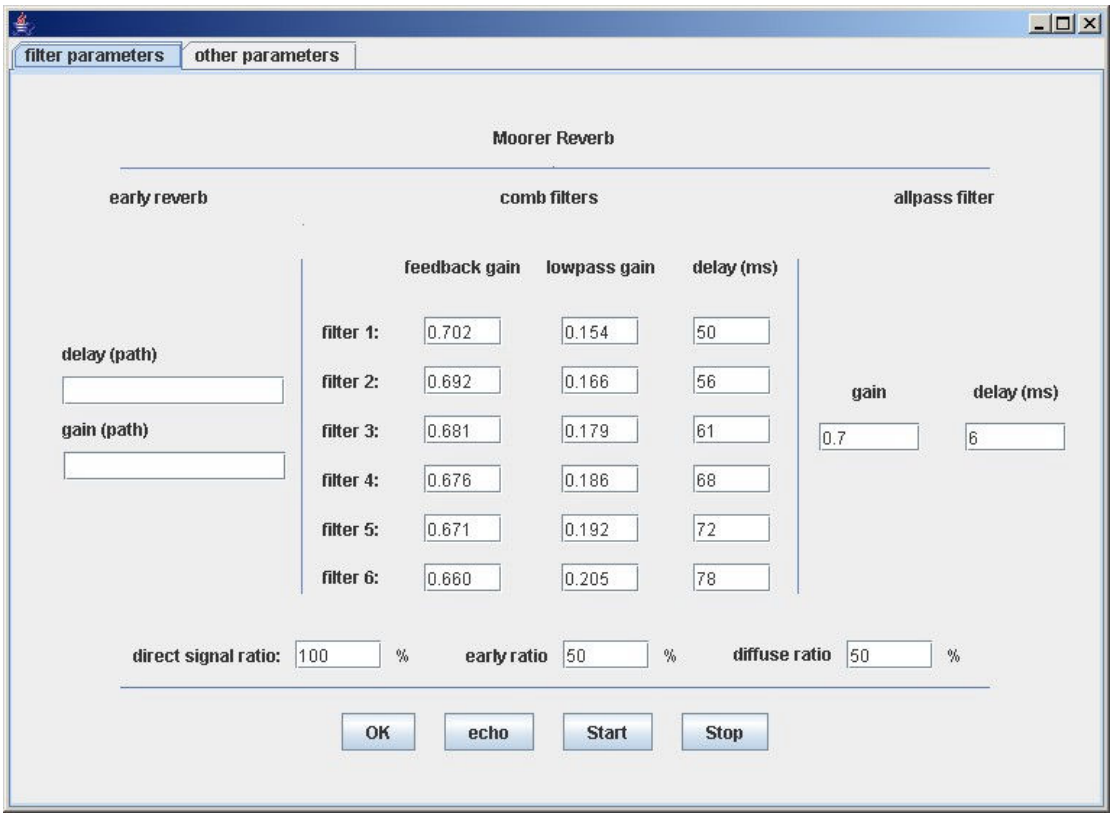
Moorer hálózata egy bemenetből egy kimenetet állít elő, az eredmény mono jel. Két, különböző paraméterekkel rendelkező hálózattal megoldható sztereo jelek előállítása is, ez azonban újabb kérdést vet fel, mégpedig a két hálózat paramétereinek kapcsolatát. Emellett két hálózat kétszer annyi adat tárolását és kétszer annyi művelet elvégzését igényli. Az általam készített zengető a rendelkezésre álló memória mérete miatt egy hálózatot valósít meg, vagyis mono kimenetet hoz létre.

6.2 A PC-n futó alkalmazás

A PC-n futó alkalmazás szolgáltatja a felhasználói felületet, ezen keresztül indítható el és állítható le az algoritmus futása, valamint itt adhatók meg a zengetés paraméterei. A zengető algoritmust megvalósító szűrőstruktúra paramétereinek változtatására kétféle felület áll rendelkezésre. Az elsőn keresztül közvetlenül a szűrők paraméterei változtathatók. A másikon olyan jellemzők állíthatók be, mint az utózengési idő, a korai visszaverődések típusa és a zengő hang arány, amelyeket a program szűrőparaméterekké konvertál. Ahhoz, hogy a kívánt értékek eljussanak a DSP-hez, a szűrőparamétereket üzenetek formájában a soros porton keresztül kell elküldeni, amelyeknek feldolgozása és jelfeldolgozó algoritmusba építése már a DSP feladata.

A PC-n futó alkalmazást Java 2 Standard Edition 1.5.0 alatt Netbeans 4.0 környezetben fejlesztettem [12.].

6.2.1 Szűrőparaméterek közvetlen változtatása



18. ábra Szűrőparaméterek közvetlen változtatására alkalmas felület

A 18. ábrán látható, szűrőparaméterek közvetlen változtatására szolgáló felületen egy echo gomb is található, amely echo üzenetet küld a DSP-nek. Ezzel ellenőrizhető a kommunikációs csatorna megléte és működése.

6.2.1.1 Korai visszaverődések

Moorer zengetőjében a korai visszaverődések megvalósítása FIR szűrővel történik, ezért ennek megváltoztatásához az új késleltetési értékeket és a hozzájuk tartozó erősítési értékeket kell megadni. Az eredeti cikk 18 együtthatós FIR szűrőt javasol, a jelenlegi alkalmazás ennél nagyobb, de max. 30 fokszámú szűrők megvalósítására is alkalmas. A maximális értéket korlátozó egyik tényező a memória mérete, hiszen az egész szűrőhálózat késleltetéseit összeadva az így kiadódó

adatmennyiségnek bele kell férnie a memóriába. Másik korlátozó tényező az DSP által egy utasításciklus alatt végrehajtható műveletek száma.

A korai visszaverődések paramétereit két text file–ban kell megadni, egyik a késleltetési értékeket tartalmazza, másik az erősítéseket. A file-ok soronként egy adatot tartalmazhatnak, az adat után szóköz és egyéb karakter nélkül rögtön a sorvége jelnek kell következnie. Az erősítési értékek 0 és 1 közötti pozitív számok lehetnek. A késleltetés értékeket ms-ban kell megadni, egész és tört számot is lehet. A maximális késleltetés 93,75 ms.

A PC-n futó alkalmazás a következő hibákat figyeli, előfordulásuk esetén hibajelzést küld:

- A file útvonala nem található
- A file-ból beolvasott szám nem megfelelő formátumú
- A két file-ban nem egyenlő számú adat található
- A FIR szűrő tap száma maximum 30, ennél több késleltetés és erősítés értéket nem lehet megadni
- Maximális késleltetés érték túllépése

6.2.1.2 Késői visszaverődések

A diffúz visszaverődéseket előállító hálózat két nagyobb egységre bontható, ez egyik hat fésűszűrőt tartalmaz, a másik a hálózat utolsó egységét képező mindentáteresztő szűrő. A fésűszűrőkre egyenként megadható a visszacsatoló ág erősítése, a visszacsatoló ágban elhelyezett aluláteresztő szűrő erősítése, illetve a belső késleltető vonalhoz tartozó idő ms-ban. Ezekre az adatokra ugyanazok a formátumbeli szabályok vonatkoznak, mint a korai visszaverődés erősítés és késleltetés értékeire. A mindentáteresztő szűrő esetében egy erősítés és egy késleltetés érték határozható meg, amelyek formátuma megegyezik az előzőekkel.

A fésűszűrőkhöz és a mindentáteresztő szűrőhöz tartozó maximális késleltetési értékeket a függelék 3. táblázata tartalmazza.

Az alkalmazás elindításakor a felhasználói felületen beállított paraméter értékek megegyeznek a Moorer által javasolt értékekkel, amelyeket a függelék 1. táblázata és 2. táblázata tartalmaznak.

Minden paraméter megváltoztatása után ENTER-t kell ütni, a program csak ilyenkor érzékeli a változást. Az összes megváltoztatni kívánt paraméter beállítása után az OK gomb lenyomásának hatására történik meg az üzenetek elküldése.

6.2.2 Szűrőparaméterek változtatása érzeti jellemzőkön keresztül

Az előző felület leginkább a szűrőparaméterek változtatása által okozott hangzásbeli különbségek tesztelésére alkalmas. A hétköznapi felhasználók szempontjából érdemesebb inkább érzeti jellemzőket megadni, ahogy ezt sok más PC-n futó audio lejátszó, szerkesztő program is teszi, így azok számára is könnyen használható, akik nem ismerik a zengető funkciót megvalósító algoritmust és annak részletes működését. Az ismertebb szoftverek közül néhányat megvizsgáltam változtatható paraméterek és azok értéktartománya szempontjából. Mivel nem minden jellemzőnek van magyar megfelelője, így az egyértelműség kedvéért mindegyik paraméter eredeti, angol megfelelőjét hagyom meg.

Cubase VST32:

- Room size (0–100 %)
- Reverberation time (0,20–12,5 s)
- Pre-delay (0–100 ms)
- Damp (-15–0 dB)
- Reverb mix (0–100 %)
- High cut, low cut

Sound Forge 4.0

- Pre-delay (0–200 ms)
- Decay time (0,5–5 s)
- Dry, early, reverb out
- Early reflections style
- Bass and high frequencies attenuation

CFX

- Reverberation time (0.20-5.00 s)
- Early reflections (sparse, dense, none)
- Dry mix (0–100%)
- Wet mix (0–100%)
- Frequency cutoff – high pass, low passs (4000-22050 Hz)

Cakewalk

- Decay (0.20-5.00 s)
- Sparse, dense, no echo
- Dry mix (0-1)
- Wet mix (0-1)
- LP and HP filters (16-16000 Hz)

CoolEdit

- Total Reverb length (20-6000 ms)
- Attack time (0-200 ms)
- Perception (smooth-echoey)
- High pass absorption time (10-4000 ms)
- Original signal (1-100%)
- Reverb (1-100%)

Samplitude

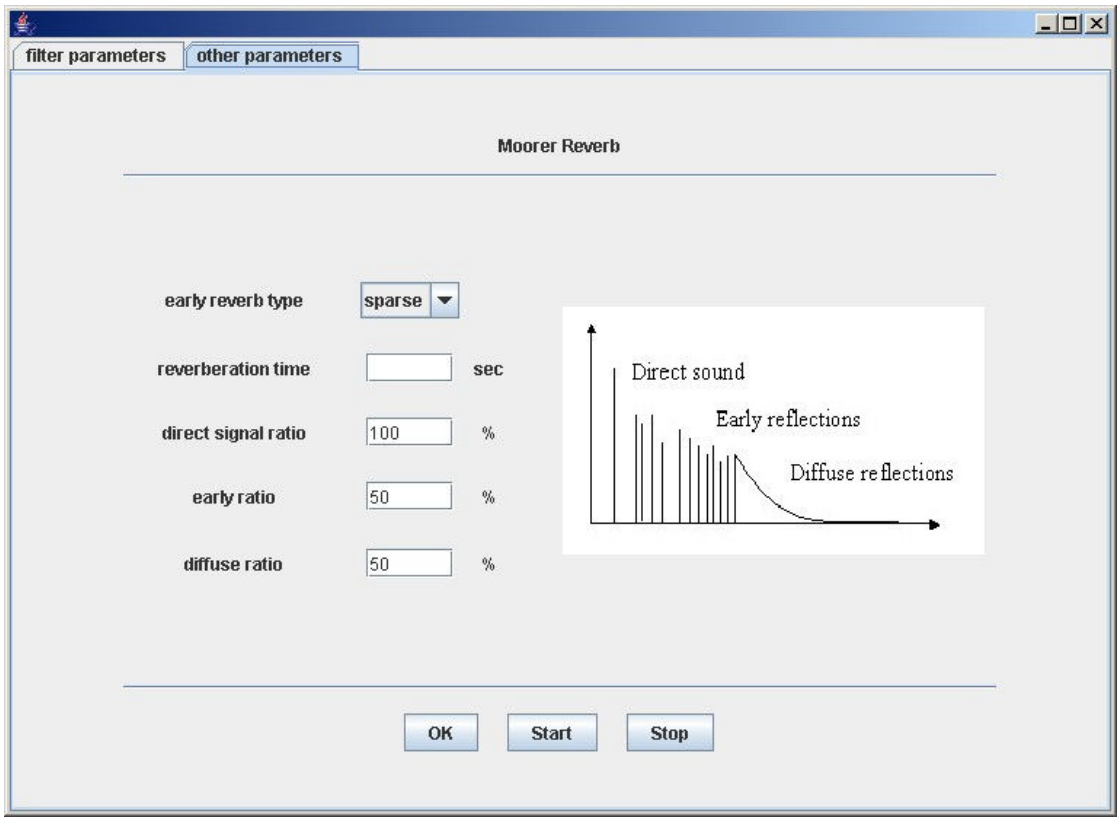
- Reverberation time (BPM (0-9999999), Millisec. (1-2000)
- Room size (10 (small)-50 (large))
- Reverb színezet (0 (dark)-1 (bright))
- Wet/Dry balance (0-100%)

A fenti programok felhasználói felületei igen változatosak. Vannak csak szöveges (Sound Forge), és képekkel kiegészített (Cubase) felületek is, ez utóbbi nagyban elősegíti az egyes paraméterek megértését, de könnyen befolyásolhat, mint például Cubase-ben a szobamérethez tartozó, a paraméter változtatásával párhuzamosan

növekvő-csökkenő térfogatú kocka látványa. Mindegyik szoftver rendelkezik standard beállításokkal (pl. dark hall, large empty hall, medium auditorium, shower)

Az elvégzett felmérés után az általam kiválasztott és a 19.ábrán látható felületen elhelyezett érzeti jellemzőket a következő felsorolás tartalmazza:

- Reverberation time (utózengési idő)
- Early reverbe type – sparse, dense (korai visszaverődések típusa - ritka, sűrű)
- Direct sound ratio (közvetlen hang aránya)
- Early ratio (korai visszaverődések aránya)
- Diffuse ratio (diffúz visszaverődések aránya)



19. ábra Érzeti jellemzők beállítását megvalósító felület

A fent felsorolt paraméterekből kell előállítani a szűrőparamétereket. Ezeket az átalakításokat írom le a következőkben, ezúttal nem a struktúra felépítését, hanem a paraméterek egymásból számíthatósági logikáját követve.

Közvetlen hang, korai és késői visszaverődések aránya:

A beolvasott százalék értéket 100-zal való osztás után 32 bites lebegőpontos számmá alakítva előáll az üzenetekhez tartozó adatmező.

Korai erősítés és késleltetés üzenetek:

A felületen háromféle korai visszaverődés típus (ritka, közepes, sűrű) választható ki, mindegyikhez tartozik egy-egy erősítés és késleltetés értékeket tartalmazó file, amelyet az első felülethez hasonlóan kell hosszú üzenetté konvertálni. A ritka típus jelenti a Moorer által javasolt 18 együtthatós FIR szűrőt. A sűrű típus 30 együtthatós. Erősítés és késleltetés értékeiket a függelék 4. táblázata tartalmazza. Ezeket az értékeket Moorer 18 együtthatós szűrőjének kibővítésével képeztem, betartva, hogy a visszaverődések időben sűrűsödnek és a maximális érték 80-90 ms körüli. Mind a két FIR szűrő impulzusválasza kb. 80 ms hosszú, mert a korai visszaverődések nagyjából ebbe az intervallumba esnek.

Mindentáteresztő szűrő üzenetek:

Moorer cikkében azt írja, hogy teszteléseik során a mindentáteresztő szűrő 6 ms késleltetése mellett tapasztalták a legjobb eredményt, rövidebb késleltetés mellett puffogó hangzás keletkezik, nagyobb késleltetés esetén kellemetlen periodikusság hallható. Ezért a mindentáteresztő szűrő paramétereinek a Moorer által ajánlott értékeket állítottam be fixen.

Fésűszűrő üzenetek:

Minden fésűszűrőhöz három paramétert kell meghatározni, a késleltetés nagyságát (d), a visszacsatoló ágban található erősítést (g_2) és a visszacsatoló ág aluláteresztő szűrőjének erősítését (g_1). Ezek kiszámításáról a 6.1 pontban már esett szó, de itt még egyszer megadom a kapcsolódó egyenleteket.

$$g_2 = g(1 - g_1) \qquad [6.14]$$

$$d = \frac{T_{60}^c \cdot 20\log_{10} g_2}{-60T} \qquad [6.15]$$

Ha a 6.14 egyenletből ismert g és g_1 értéke, akkor meghatározható g_2. A késleltetés kiszámításához további szükséges adatok T mintavételi idő és T_{60}^c, a fésűszűrőhöz tartozó utózengési idő. A minden szűrőre vonatkozó g paraméter

kiszámítható 6.17 képlettel, ahol T_{60} az egész hálózatra vonatkozó, felületen beállítható utózengési idő. .

$$g = 1 - \frac{0{,}366}{T_{60}} \qquad [6.17]$$

Az aluláteresztő szűrő visszacsatoló ágának g_1 értékét Moorer grafikonok alapján határozta meg, saját szűrői beállításai mellett, 25 és 50 kHz mintavételi frekvenciákhoz a függelék 2. táblázatában látható értékeket javasolta.

A fentiek alapján az első fésűszűrőhöz 16 kHz-en az állandó 0.16 értéket javaslom. Mivel a megvalósított zengető fésűszűrő késleltetéseinek maximális hossza egy nagyságrendbe esik a Moorer által javasolt késleltetésekkel, ezért ez az erősítés érték a maximális késleltetések esetén pontosnak tekinthető. Rövidebb késleltetések használata esetén ez a modell nem pontos, ezért a páratartalomtól és a fésűszűrő késleltető vonal hosszától való függőség megállapításához további vizsgálatokra lenne szükség. Az aluláteresztő szűrő erősítésének megadásával rendelkezésre áll g_1, g számítható, tehát $g2$ is kiszámolható. A fésűszűrő késleltetésének megadásához még egy adat hiányzik, a fésűszűrőhöz tartozó utózengési idő nagysága. Amit biztosan tudunk az utózengési időről, hogy a korai visszaverődéseknél, ritka, közepes és sűrű esetben is kb. 20 dB-lel csökken az energia, tehát a hálózat maradék részére, vagyis a fésűszűrőre és a mindentáteresztőre 40 dB energiacsökkenés jut. A két szűrőre külön-külön számítható az utózengési idő, és 6.18 és 6.19 képletekkel a fésűszűrőhöz (T_{40}^{c}) és a mindentáteresztőhöz (T_{40}^{ap}) tartozó T_{40} is.

$$T_{40}^{c} = \frac{-40(d_c T)}{20\log_{10}(g_c)} \qquad [6.18]$$

$$T_{40}^{ap} = \frac{-40 d_{ap} T}{20 \cdot \log_{10}((1 - g_{ap}{}^{2}) g_{ap})} \qquad [6.19]$$

A két szűrő sorosan kapcsolt, frekvenciatartományban átviteli függvényük szorzódik. Időtartományban azonban a szűrők közös impulzusválasza az impulzusválaszok konvolúciójából származik, amely már nem adható meg olyan egyszerűen, mértani sorral, mint az egyes szűrők esetében külön-külön.

Matlab-ban végzett kísérleteim során azt tapasztaltam, hogy a sorosan kapcsolt fésűszűrő és a korábban megadott fix paraméterekkel rendelkező mindentáteresztő

szűrő közös T_{40}-e kb. megegyezik a fésűszűrő fenti képlettel számítható T_{40}-évl. Ezért a felhasználói felületen beírt utózengési időből levonva a korai visszaverődésekhez tartozó 80 ms időt, kiszámítható a fésűszűrőhöz tartozó T_{40}^{c}.

A fent leírt módszerrel tehát kiszámítható az egy fésűszűrőhöz tartozó T_{40}^{c}, amiből g_2 ismeretében megadható a késleltetővonal hossza (d). A többi szűrőhöz tarozó késleltetések nagysága az előzőhöz képest 1-1,5 arányban oszlik el, szintén Moorer ajánlása alapján. Ezekhez tartozó g_2 értékek a 6.19 képletből, g_1 pedig 6.14 egyenletből számíthatók.

6.2.3 A felülethez tartozó függvények

- *public static void main(String args[])*: meghívja a felület konstruktorát (public Felulet())
- *public Felulet()*: Ez a konstruktor, amely meghívja initComponents függvényt és a soros port kezelő konstruktorát (public SorosPortKezelo())
- *private void initComponents()*: inicializálja a felület elemeit
- szövegdobozokhoz és gombokhoz tartozó ActionListener-ek
- konverziók:

  private int timeToPrimeSample (double szam): ms-ban megadott késleltetési értékeket prím mintaszámmá konvertál

  private int timeToSample (double szam): ms-ban megadott késleltetési értékeket mintaszámmá konvertál

  private int gainTo32Int (double szam): erősítés értékeket 32 bites integerré konvertálja

  private String intTo32Hex (int szam): Integer-t 32 bites hexadecimális stringgé konvertál
- adatmezők előállítása:

  public String earlyDelayConversion(int szam) korai visszaverődés értékek 32 bites integerré konvertálását végzi

  public String earlyGainConversion(double szam): korai visszaverődések erősítésének konvertálását végzi

  public String gainConversion(double szam, String uzenetkod): üzenetkódból és erősítés értékből állítja elő a megfelelő üzenetet

public String delayConversion(double szam, String uzenetkod): üzetekódból és ms-ban adott késleltetés értékből üzenetet generál

public String ratioConversion(int szazalek, String uzenetkod): üzenetkódból és százalékban adott arányértékből állít elő üzenetet

- üzenet generálás:

public String rovidUzenet(String code): rövid típusú üzeneteket állít elő, az üzenetkód ismeretében

private String hosszuUzenet(String code, String length, String hexStr): hosszú típusú üzenetet állít el, bemenő paraméterei az üzenetkód, az adatmező hossz és az adatmező

private String checkSzamolo(String checkszamolostr): a paraméterként kapott String-hez kiszámítja a CHK mező értékét

- Hibajelzések:

private void gainHibaJelzes(): ha valamelyik erősítés nem 0 és 1 között van

private void delayHibaJelzesNegativ(): ha valamelyik késletetés érték negatív

private void delayHibaJelzesMax(String filtername, double max): ha valamelyik szűrő késleltetése meghaladja a hozzá tartozó maximumot.

private void earlyDelayHibaJelzes(): a korai visszaverődéseknél megadott legnagyobb késleltetés meghaladja a lehetséges maximumot.

private void earlyHibaJelzesMax(): a korai visszaverődések száma túllépi a maximális értéket.

private void earlyHibaJelzesFile(): a file nem található

private void earlyHibaJelzesHossz(): a korai visszaverődések erősítés és késleltetés száma különbözik.

private void szazalekHibaJelzes(): közvetlen hang, korai vagy késői visszaverődések aránya nem a 0-100 % tartományba esik.

private void earlyDelayHibaJelzes(): korai késleltetés maximuma meghaladja az enegdélyezett maximumot.

6.2.4 A PC és a DSP közötti kommunikáció

A PC és a DSP közötti kommunikáció soros porton keresztül valósul meg. A kommunikáció master – slave jellegű, minden esetben a master kezdeményezi az üzenetküldést.

A kommunikációhoz szükséges a PC oldalán az üzenetek generálása, majd elküldésük az OK gomb lenyomásának hatására. Amennyiben az üzenet eljutott a DSP-hez, ellenőrzésen esik át. Ha minden kritériumnak megfelel, akkor feldolgozásra kerül.

Az DSP által értelmezhető üzenetek formája definiált, két csoportba sorolhatók [9.].

A 20.ábrán látható rövid üzenet parancsot vagy jelzést, míg a hosszú üzenet adatokat, vagy paraméterrel rendelkező parancsot tartalmaz.

20. ábra A DSP által fogadott üzenet típusok

A start karakter minden üzenet esetén ’*’. A Címzett értelemszerűen az üzenet címzettje, a Feladó pedig az üzenet feladója. A HOST számítógép címe ’00’, a kártyáé ’01’. Az üzenetkód tartalmazhat 8 bites integert, 16 bites integert, 32 bites integert vagy lebegőpontos számot. Az adatmező hossza mező 8 bites, az adatmező karaktereinek számát tartalmazza. Az adatmező típusa szintén 8 bites integer, 16 bites integer, 32 bites integer vagy lebegőpontos szám lehet. Az üzenet tartalmaz egy CHK mezőt, amely az üzenet start karakter és CHK mező közötti részéből számol ellenőrző összeget, méghozzá olyan módon, hogy a karakterek ASCII kódjainak összegét képzi, majd az összeg hexadecimális formájának alsó 4 karakterét hagyja meg. Ha az ellenőrző összeg 4 karakternél rövidebb, akkor az elejét nullákkal kell kiegészíteni. Az üzenet végére egy stop karakter kerül, amely ’;’.

Példák:

- Rövid üzenet: *0100110123; (echo üzenet a PC felől a DSP részére)

- Hosszú üzenet: *0100A1080000065B0338; (első fésűszűrő erősítés értéke a PC felől a DSP felé)

6.2.5 A PC oldali kommunikáció

6.2.5.1 Általános működés

Mindkét felület esetén az OK gomb megnyomásának hatására először egy Application_Stop üzenet generálódik. A következő üzenet egy Application_Start, amely elindítja a DSP-n futó applikációt. Az Application_Stop azért szükséges, mert nem szerencsés Application_Start-ot küldeni, miközben a DSP-n már fut az alkalmazás. Az újraindítás hatására a szűrőkhöz tartozó cirkuláris bufferek nullázódnak. Az Application_Start üzenet elindítaná a zengető algoritmus futását is, még a paramétereket tartalmazó üzenetek lekezelése előtt vagy közben, ami hangzásbeli tranzienseket okozhat. Ezért szükség van egy Algorithm_No_Run üzenetre, amelynek hatására az applikációs részben ilyenkor csak az üzenetek feldolgozása történik, a zengetés szünetel. Ezután kerülnek elküldésre a megváltozott paraméterekből generált üzenetek. Csak azok a paraméterek kerülnek elküldésre, amelyek értéke megváltozott. Ha egyik paraméter értéke sem változott meg, akkor a DSP oldali alkalmazás megállás nélkül fut tovább, mert nem kap egyetlen üzenetet sem. Ezek után következhet az Algorithm_Run, amely elindítja az újraparaméterezett zengetőt.

A korai visszaverődés késleltetés és erősítés értékei egy-egy hosszú üzenetben mennek át. A korai késleltetéseket megvalósító FIR szűrő maximális együttható száma 30. Az erősítés értékek mint 32 bites lebegőpontos számok, a késleltetés értékek pedig mint 32 bites fix pontos számok kerülnek elküldésre. Az üzenetek csak akkor generálódnak, ha a késleltetés és erősítés értékeket tartalmazó file-okból beolvasott adatok száma megegyezik, formátumuk, értékük megfelelő, egyébként hibaüzenet jön.

A fésűszűrők és a mindentáteresztő szűrő felületen állítható paraméterei közvetlenül kerülnek kiküldésre egy-egy hosszú típusú üzenetben. Minden érték külön üzenetbe kerül, így megoldható, hogy valóban csak a megváltozott paraméterek kerüljenek átküldésre.

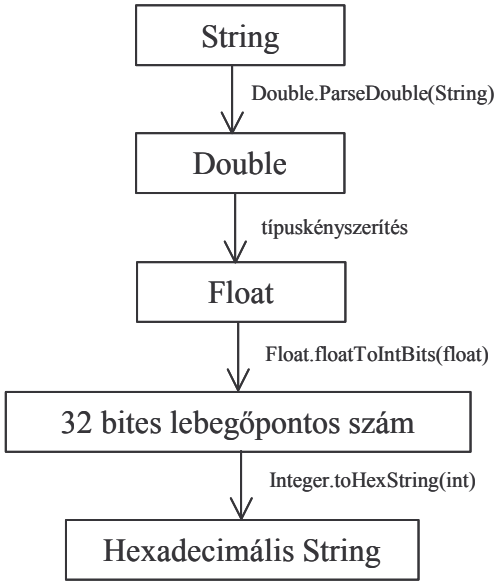
A DSP minden helyes üzenet átvétele után ACK üzenetet küld vissza. A PC ezeket jelenleg nem kezeli le, vagyis nem figyeli, hogy hány ACK üzenetet kapott, és melyik üzenetekre. Ezt a jelenleg hiányzó funkciót megvalósítva ki lehetne zárni a hibalehetőségeket, az üzenetek sérülése esetén megoldható lenne azok újraküldése. Mivel az üzenetek elvesztése ennél az alkalmazásnál súlyos következménnyel nem jár, hiszen a szűrőparaméterek regiszterei mindig tartalmaznak valamilyen értelmes adatot, ezért a funkció hiánya nagyobb gondot nem okoz.

Az üzenetekhez tartozó üzenetkódokat 5. táblázata tartalmazza.

6.2.5.2 Üzenetek generálása

A felhasználói felületről beolvasott értékek stringként állnak rendelkezésre, ezeket hexadecimális stringgé kell alakítani ahhoz, hogy üzenetként elküldésre kerülhessenek.

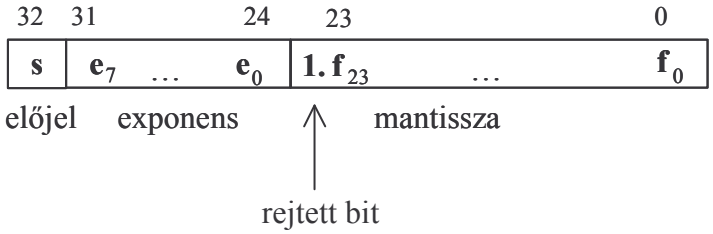
Az erősítés értékek átalakítását a 21. ábra mutatja.



21. ábra Erősítés értékek konvertálása

A felületről beolvasott stringet először a Double.parseDouble(String) függvény double értékre konvertálja, amely típuskényszerítéssel floattá alakítható. Ez azért szükséges, mert így a Float.floatToIntBits(float) függvénnyel a float értékből olyan, a 22.ábrán látható, IEEE 754 szabványnak megfelelő lebegőpontos szám hozható létre,

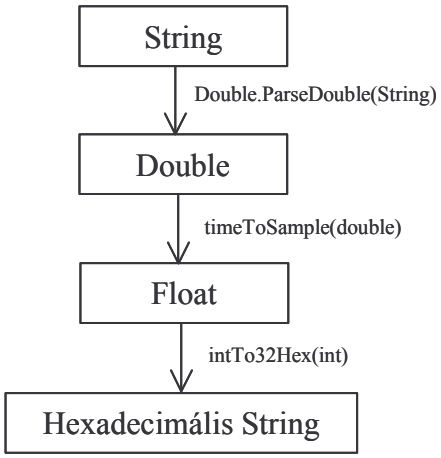
ahol Bit 31 az előjelbit, Bit 20-23 az exponens, Bit 22-0 a mantissza. A DSP ugyanezzel a formátummal dolgozik. double-ra is létezik függvény, amely a fenti szabványnak megfelelő, de 64 bites értéket állít elő (Double.doubleToLongBits(double).



22. ábra Lebegőpontos számábrázolás

Utolsó lépésként az Integer.toHexString(int) függvény meghívásával előállítható a hexadecimális string, amely a küldendő üzenet adatmezejét képezi.

A késleltetés értékek konvertálási lépései a 23.ábrán láthatók.



23. ábra Késleltetés értékek konvertálása

A felületről beolvasott késleltetést ms-ban tartalmazó stringből az erősítéshez hasonlóan a Double.parseDouble(str) függvény segítségével egy double keletkezik. Ezt az általam írt timeToSample(double) függvény ms-ból mintaszámmá konvertálja a mintavételi frekvenciának megfelelően. Az így kapott nem prím számértékeket a 6.1 pontban leírtak szerint érdemes prímszámmá alakítani. Ehhez rendelkezésre áll egy prímszámokat 1-től 5000-ig tartalmazó file, amelyből a függvény kikeresi a

mintaszámhoz legközelebb eső, nála kisebb prímszámot. Ez utóbbit az intTo32Hex(int) függvény hexadecimális értékké konvertálja, amely az üzenet adatmezejébe kerül.

6.2.6 A javax.comm package

A Java Platform eredetileg nem tartalmazza a kommunikációhoz szükséges classokat, interface-eket és exception-öket, ezeket egy külön installálandó kiegészítő package-ben gyűjtötték össze a javax.comm-ban, amely letölthető a http://java.sun.com/products/javacomm/downloads/index.html címről. Ez a csomag az RS-232 soros porton és az IEEE 1284 párhuzamos porton folyó kommunikációt támogatja.

A package telepítésének lépéseit azért írom le, mert az Interneten szereplő lépések végrehajtása nem elegendő.

1. javacomm20-win32.zip kicsomagolása, amelynek eredményeként keletkezik a commapi könyvtár, ez tartalmaz telepítési útmutatót, példákat, dokumentációt és a <jdk> bizonyos könytáraiba másolandó file-okat
2. win32com.dll másolása <jdk>\bin könytárba
3. comm.jar másolása <jdk>\lib könyvtárba
4. javax.comm.properties másolása <jdk>\lib könyvtárba
5. javax.comm.properties <jdk>\jre\lib
6. comm.jar másolása <jdk>\jre\lib\ext (Netbeans használata esetén szükséges)
7. comm.jar hozzáadása a CLASSPATH-hoz

6.2.7 A sorosport-kezelés működése és függvényei

A soros port aktiválásához szükséges lépéseket a 24. ábra mutatja be, az előforduló lehetséges kivételekkel. Ha valamelyik kivétel bekövetkezik és a port aktiválása nem sikerül, akkor lehetetlenné válik az üzenetek elküldése, ezért ilyenkor már maga a felület sem jelenik meg.

A soros port beállítása a kártya igényeinek megfelelően:

baud rate: 38400 baud,

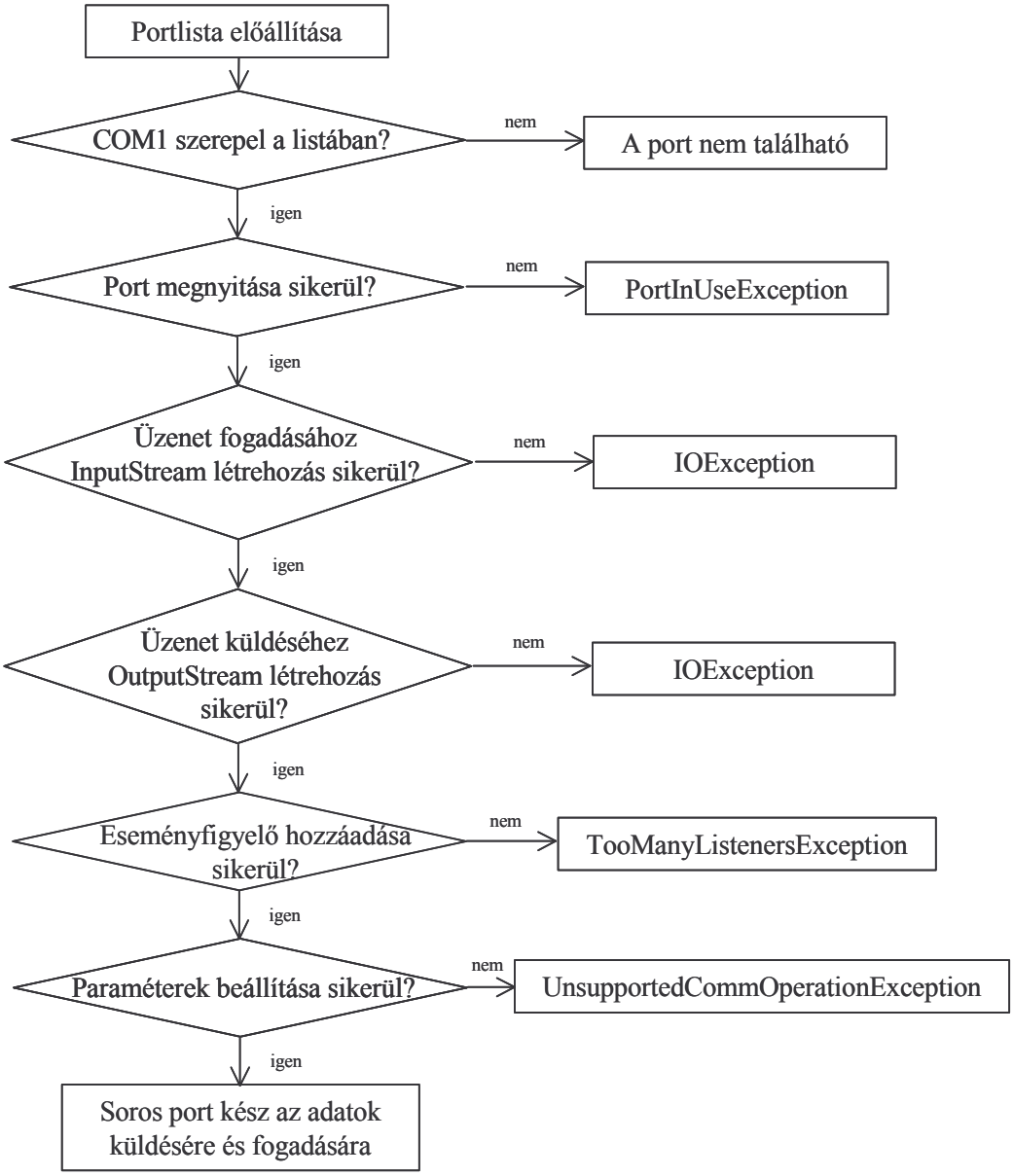
adatbitek száma: 8,

stop bitek száma: 1,

paritás: nincs.

A soros port kezeléséhez kapcsolódó függvények:

- public SorosPortKezelo(): folyamatábra



24. ábra Soros port megnyitásának folyamata

- public void uzenetKuld(String s): a paraméterként kapott stringet elküldi a soros porton
- public void portZar(): soros port lezárása
- public void serialEvent(SerialPortEvent event): a porton keletkező eseményeket kezeli, jelen esetben az OUTPUT_BUFFER_EMPTY és a DATA_AVAILABLE típusúakat
- public void uzenetErkezik(): bejövő üzenet fogadása, kiírása a felületre

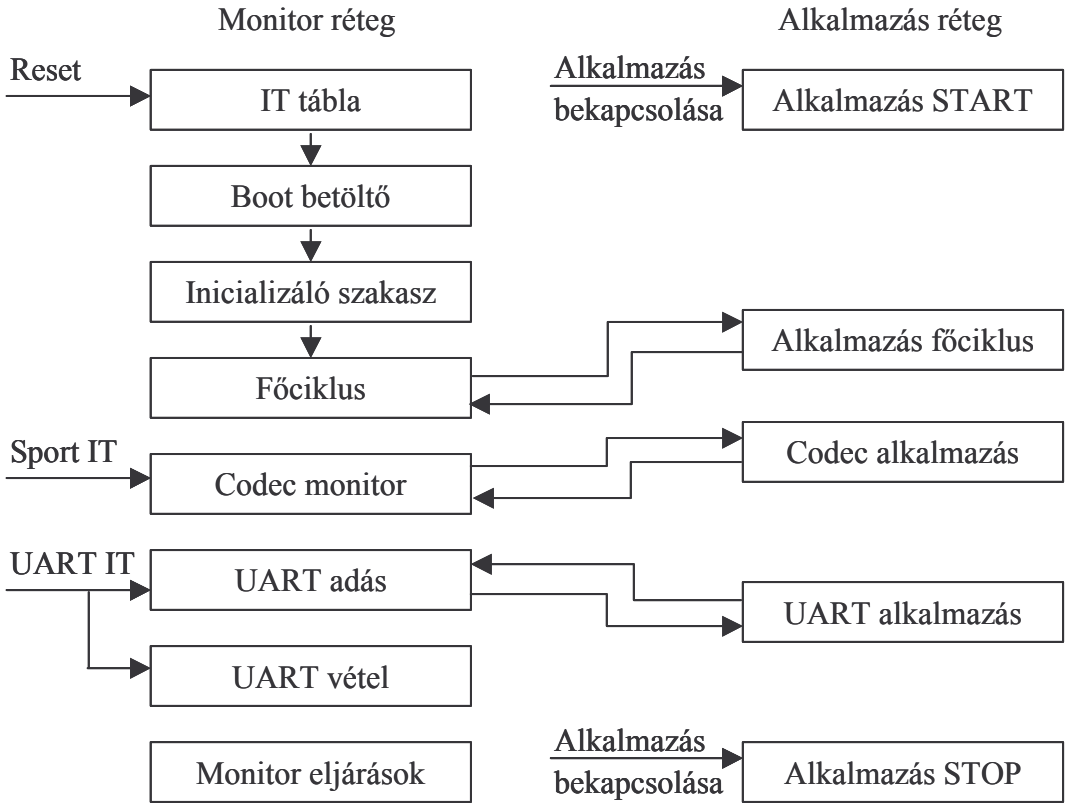
6.3 A DSP-n futó alkalmazás

A programfejlesztést VisualDSP++ környezetben assembly nyelven végeztem. A program feltöltése DSP-re és tesztelése JTAG-en keresztül történt.

A program általános struktúrájának részletes leírása [9.]-ben megtalálható. Az én feladatom az alkalmazásspecifikus rész, jelen esetben a szűrőhálózatok megvalósítása volt, valamint a PC által küldött üzenetek már elvégzett előzetes feldolgozása után azok értelmezése és végrehajtása.

6.3.1 A DSP-n futó program általános felépítése

A DSP-n futó program a 25.ábrán látható két nagyobb egységre tagolódik, a Monitorra és az Alkalmazásra.



25. ábra A DSP programjának általános felépítése

A Monitor RESET után megkezdi az inicializálást, ebben a fázisban a megszakítások tiltottak. Először betöltődik az IT tábla, ezután lefut a boot betöltő, majd megtörténik a processzor és a hozzá kapcsolódó perifériák inicializálása (codec, FPGA, UART, IO, LED-ek). Ezek után engedélyezésre kerülnek a megszakítások, majd elindul a főciklus. Ez tulajdonképpen egy ciklikus NOP utasítás, ilyenkor a processzor megszakításra vár. A Monitor további részei a codec monitor, amely a codec felől érkező interruptot figyeli, az UART monitor, amely az UART adás és vétel interruptot nézi, továbbá a monitor eljárások. Ez utóbbiak a Monitor és az Alkalmazás által is meghívható eljárásokat jelentenek, ide tartoznak a codec inicializálás, a codec leállítás, a FLASH és SDRAM memóriakezelés, az UART és IO kezelés és az Alkalmazás kontextus betöltés vagy mentés.

Az Alkalmazás részei a start (Load_Application_Context), stop (Clear_Application_Context), főciklus (Application_Main), a codec alkalmazás (Codec_Input_Application) és UART alkalmazás (Serial_Input_Application).

Az Alkalmazás start az alkalmazás bekapcsolásakor, az Alkalmazás stop kikapcsoláskor, egyszer fut le. A codec alkalmazás mintavételi időnként fut le, ez a rész tartalmazza a jelfeldolgozó algoritmust. A soros porti alkalmazás értelmezi az alkalmazáshoz kapcsolódó üzeneteket.

6.3.2 A DSP memóriájának szegmentálása

A keretprogram eredeti memória szegmentálását módosítani kellett, részben az adatok 16 biten történő tárolása miatt, részben a zengető algoritmus adatai által igényelt nagyméretű összefüggő memóriaterületek miatt.

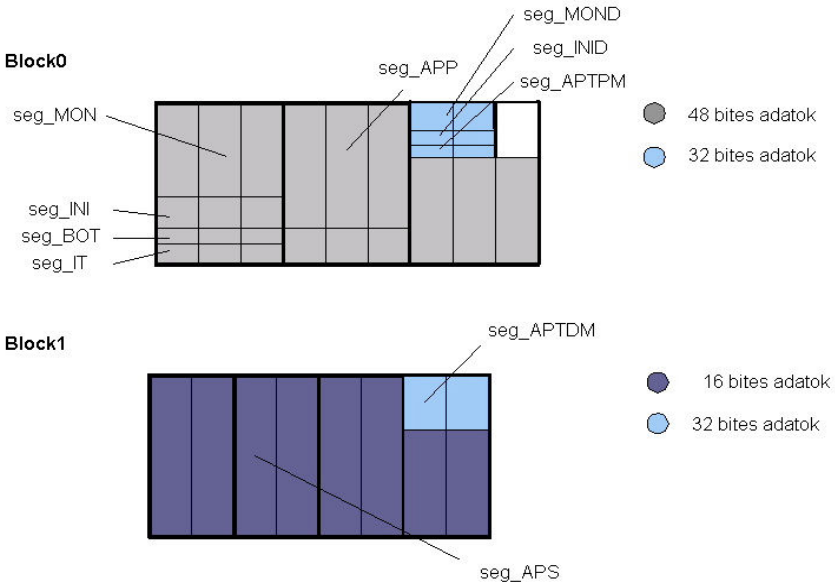
A memória két blokkjának felosztását az alábbi táblázat tartalmazza és a 26. ábra mutatja be.

szegmens	Memória típusa	Start address	End address	Memória szélesség
BLOCK0				
Seg_IT	PM*	0x00008000	0x0000807F	48
seg_BOT	PM	0x00008080	0x000080FF	48
seg_INI	PM	0x00008100	0x0000827F	48
seg_MON	PM	0x00008280	0x000088FF	48

seg_APP	PM	0x00008900	0x0000945F	48
seg_APTPM	PM	0x00009C95	0x00009CE6	32
seg_INID	DM**	0x00009CE7	0x00009D57	32
seg_MOND	DM	0x00009D58	0x00009FFF	32
BLOCK1				
seg_APS	DM	0x00018000	0x0001BF1F	16
seg_APTDM	DM	0x0000DF90	0x0000DFFF	32

* Program Memory

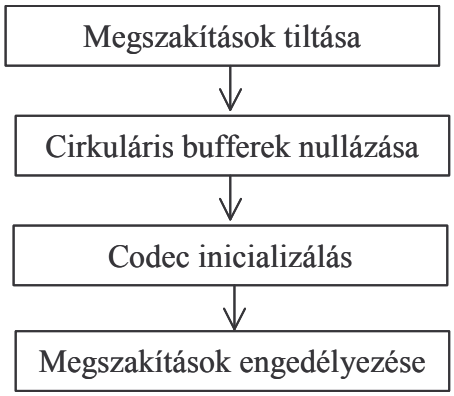
** Data Memory



26. ábra A DSP memória felosztása

6.3.3 Inicializálás

Az inicializáló szakasz lépéseit a 27. ábra tartalmazza.



27. ábra Inicializáló szakasz lépései

A megszakítások letiltása után megtörténik a szűrőkhöz használt cirkuláris bufferek nullázása. Az egyes bufferek hosszát a 3. táblázat tartalmazza, összegük nagyságrendileg 15000 .Egy ilyen nagy memóriaterület nullázása olyan sok műveletet jelent, ami nem hajtható végre egy mintavételi ciklus alatt, ezért van szükség ebben a szakaszban a megszakítások tiltására. Ugyanebből az okból kifolyólag a codec-ek beállítása és elindítása csak a bufferek resetelése után következik, mert ennek során engedélyezésre kerülnek az interruptok. A kártya 8, 16, 32 és 64 kHz-es mintavételi frekvenciával tud dolgozni, külső oszcillátor csatlakoztatásával tovább növelhető a lehetőségek száma. 8 kHz túl kevés egy audio alkalmazáshoz, 32 és 64 kHz-en viszont olyan nagy mennyiségű adat tárolására lenne szükség, ami már nem fér be a memóriába, ezért a szóba jöhető mintavételi frekvencia 16 kHz. Az inicializáló szakasz végén a megszakítások engedélyezésre kerülnek.

6.3.4 Az codec alkalmazás

A jelfeldolgozó processzoron megvalósított zengető valós idejű alkalmazás, ezért a codec alkalmazásba bele kell férnie az összes egy mintavételi ciklus alatt végrehajtandó utasításnak, jelen esetben mind a nyolc szűrő megvalósításának, valamint az adatok konvertálásának. Ez utóbbi tulajdonképpen azt jelenti, hogy a bejövő adatokat lebegőpontos értékekké kell alakítani. A codec-ről érkező adatok 16 bitesek, amelyek a 32 bites regiszterek alsó felén helyezkednek el. Ezeket egy 16 bites logikai balra shifteléssel, majd egy 16 bites aritmetikai jobbra shifteléssel és egy *float* utasítással 32 bites lebegőpontos értékké lehet alakítani. Ha azonban minden bejövő érték és minden

közbülső szűrőérték 32 bites formában kerülne a memóriába, akkor nem lenne elegendő a rendelkezésre álló belső memória. A kártyán van ugyan külső memória, ehhez azonban lassú a hozzáférés, és nem teljesen véletlenszerű, ezért ez a megoldás sem jöhet szóba. Ezért mind a bejövő mintákat, mind a szűrőhöz tartozó adatokat 16 bites formában kell tárolni, ezáltal feleakkora területet igénylenek. A DSP Short Word móddal támogatja a 16 bites adatok kezelését. A szűrők megvalósításához szükséges számítások előtt az 16 bites adatokat 32 bites formára kell átalakítani.

6.4 FIR-szűrő

A korai visszaverődéseket megvalósító FIR szűrő egyenlete a 6.20.

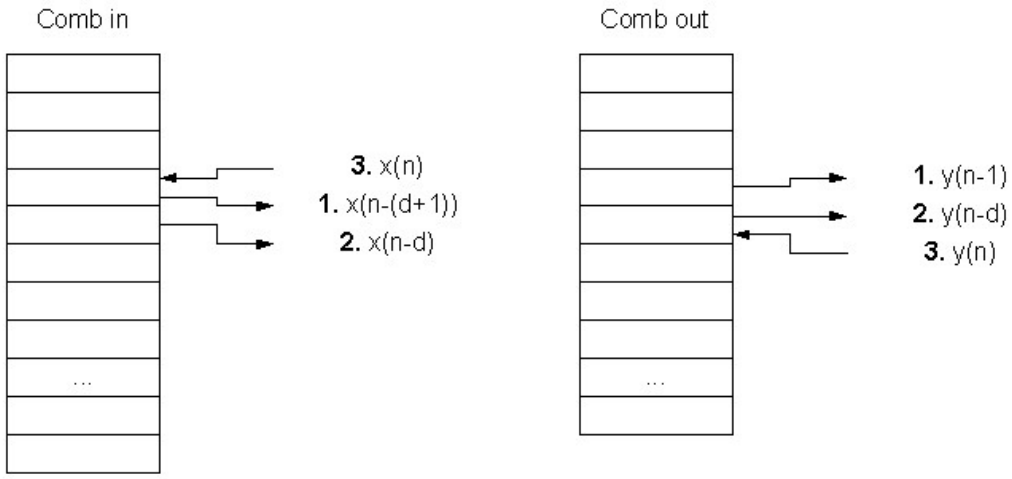
$$y(n+1) = g_0 \cdot x(n) + g_1 \cdot x(n-d_1) + g_2 \cdot x(n-d_2) + ... + g_m \cdot x(n-d_m) \qquad [6.20]$$

A FIR szűrő kimenetének kiszámításához szükséges korábbi bemenő adatokat egy cirkuláris buffer tartalmazza. Bár egyetlen kimenő adat kiszámításához maximum 30 bemeneti érték szükséges, az egymást követő kimenetek más bemenő adatokat igényelnek, ezért kb. 80 ms-nak megfelelő hosszúságú cirkuláris buffer szükséges. Az együtthatók és a korábbi bemenő adatok közötti lépésközök szintén egy-egy cirkuláris bufferben kapnak helyet.

6.5 Fésűszűrők

Egy fésűszűrőhöz két cirkuláris buffer tartozik, az egyik a bemenő, a másik a kimenő adatokat tárolja. A fésűszűrő kimenetének kiszámításához a 6.21 egyenletből láthatóan bemenő és kimenő adatokra is szükség van, az ezekhez tartozó adatmozgató lépéseket a 28. ábra mutatja be.

$$y(n+1) = x(n-d) - g_2 \cdot y(n-d) + g_1 \cdot y(n-1) - g_1 \cdot x(n-(d+1)) \qquad [6.21]$$

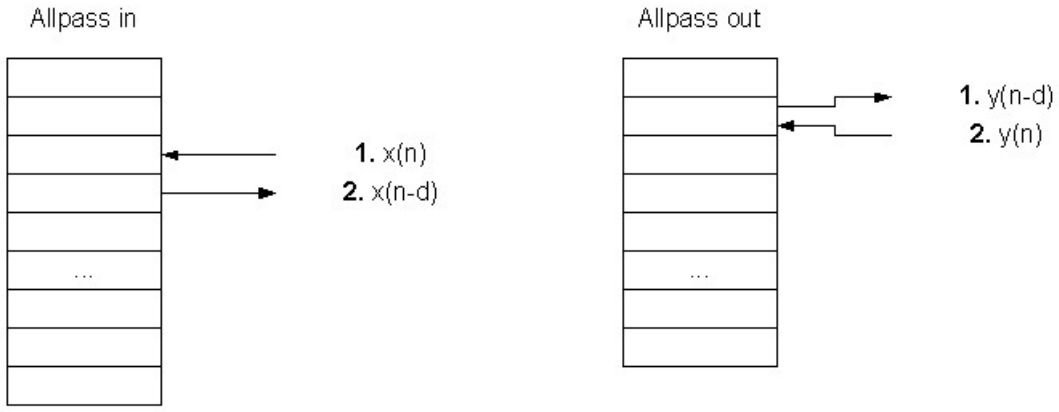


28. ábra Fésűszűrő adatokat tároló cirkuláris bufferek

6.6 Mindentáteresztő szűrő

A mindentáteresztő szűrő szintén két cirkuláris buffert igényel, mert mint az a 6.22 egyenletből látható, korábbi bemenő és korábbi kimenő adatokra is szüksége van. A cirkuláris bufferek és a hozzájuk tartozó adatmozgatások 29.ábrán láthatók.

$$y(n+1) = (-g) \cdot x(n) + x(n-d) + g \cdot y(n-d) \qquad [6.22]$$



29. ábra Mindentáteresztő szűrő cirkuláris bufferei

6.6.1 DSP oldali kommunikáció

A PC-s felületen az OK gomb megnyomásának hatására, amennyiben valamelyik paraméter értéke változott, megtörténik az üzenetek generálása és elküldése a DSP-re. Ilyenkor az ott futó zengető algoritmust le kell állítani. Az üzenetek tartalmának értelmezése után a megváltozott paraméter értékeket el kell tárolni. Végül az újraparaméterezett alkalmazás elindítása előtt nullázni kell az összes cirkuláris buffer tartalmát, nehogy az előzőekből maradt adatok zavart, torzulást okozzanak.

A beérkezett üzeneteket a Monitor réteg kapja meg. Amennyiben az nem tudja kezelni az üzenetet, de annak címzettje a kártya, a hossza és a CHK is megfelelő, akkor az üzenet kezelését átadja az alkalmazásnak. Azokat az üzeneteket, amelyeknek rossz a CHK értéke, nem megfelelő hosszúságúk, vagy amelyek nem a kártyának szólnak, a DSP eldobja. Ezért fontos a felhasználói felületen látható kezdeti paramétereket beállítani a DSP-n is, így biztosan lesz értelmes tartalom a késleltetés és erősítés regiszterekben, akkor is, ha üzenetvesztés miatt ez nem egyezik meg a felhasználó által kívánt, a felületen beállított értékkel.

Az üzenetek típusai és szerkezetük leírása a 6.2.4 pontban található.

Korai erősítés és késleltetés üzenetek:

Az üzenetben érkező adatmezőhossz megadja az adatmezőben érkezett karakterek számát, az adatmező pedig tartalmazza az egyenként 32 bites, vagyis 8 karakterből álló erősítés és késleltetés értékeket. Mivel a memóriafoglalás nem dinamikus, nem lehet minden paraméterváltáskor újra felosztani a memóriát a szűrők késleltetésének hossza szerint. Ezért mind a FIR-szűrőhöz, mind a többi szűrőhöz tartozik egy maximális késleltetés, amelynek nagyságát megadja az inicializáláskor hozzá kapcsolt memóriaterület mérete. Korai visszaverődések esetén nemcsak a legnagyobb késleltetés, hanem a FIR-szűrő tap száma is maximalizált.

Fésűszűrő és mindentáteresztő szűrő üzenetek:

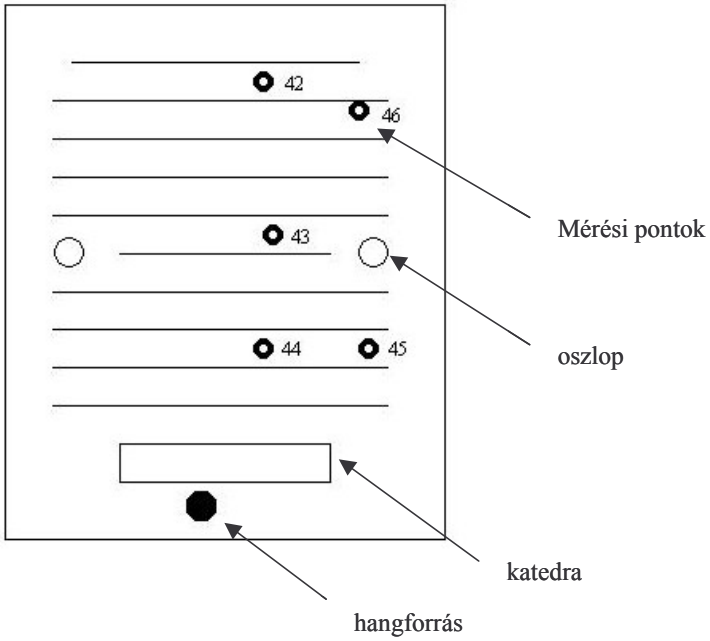
A beérkezett üzenetek értelmezése után az adatmezőben található erősítés és késleltetés adatokat a megfelelő memóriaterületekre kell kiírni.

7 Eredmények

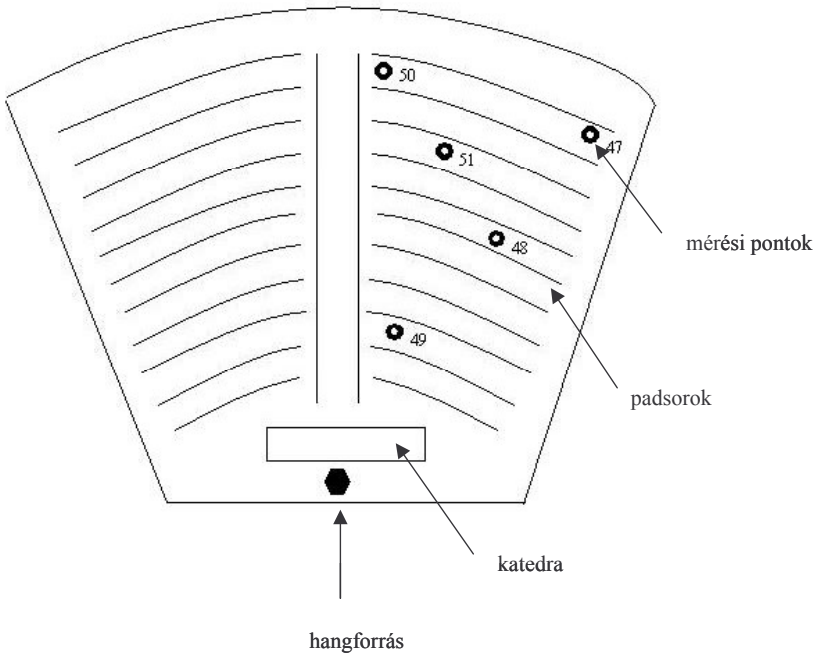
7.1.1 Mérési pontok

Az IB027 és IB028-as termek vázlatos rajzát a 30. és 31.ábra mutatja be. A pontok számozása a 8 csatornás rögzítőn hozzájuk tartozó felvétel számából származik.

A megmért termek előadóként funkcionálnak, ezért ehhez logikusan alkalmazkodva a gerjesztés helyének minden esetben a katedrához közeli pontot választottam. Mivel a termek hossztengelyükre szimmetrikusnak tekinthetők, ezért a mérési pontokat az egyik teremfélben választottam nagyjából egyenletesen elosztva. A termek üresek voltak, vagyis nem tartalmazták a hallgatók igen jó elnyelőnek mondható felületét. Az így mért impulzusválaszok hosszabbak, mintha hallgatókkal megtöltött termet mértem volna.



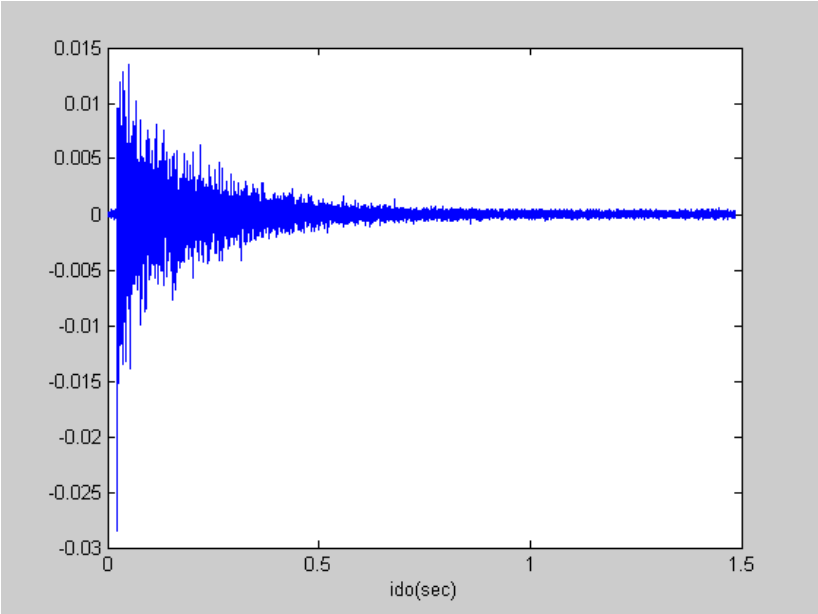
30. ábra Mérési pontok az IB027-ben



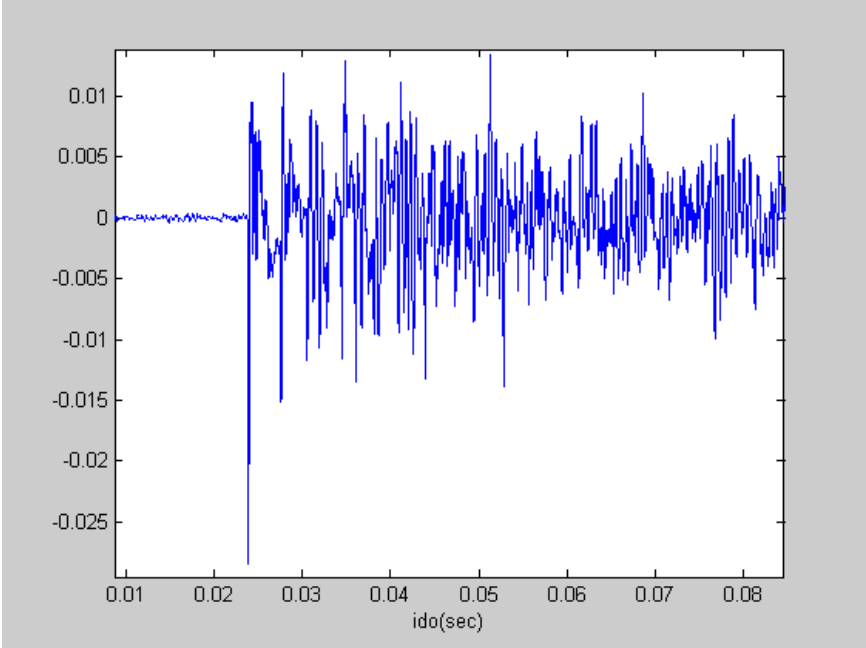
31. ábra Mérési pontok az IB028-ban

7.1.2 Mérési eredményekből számított impulzusválaszok

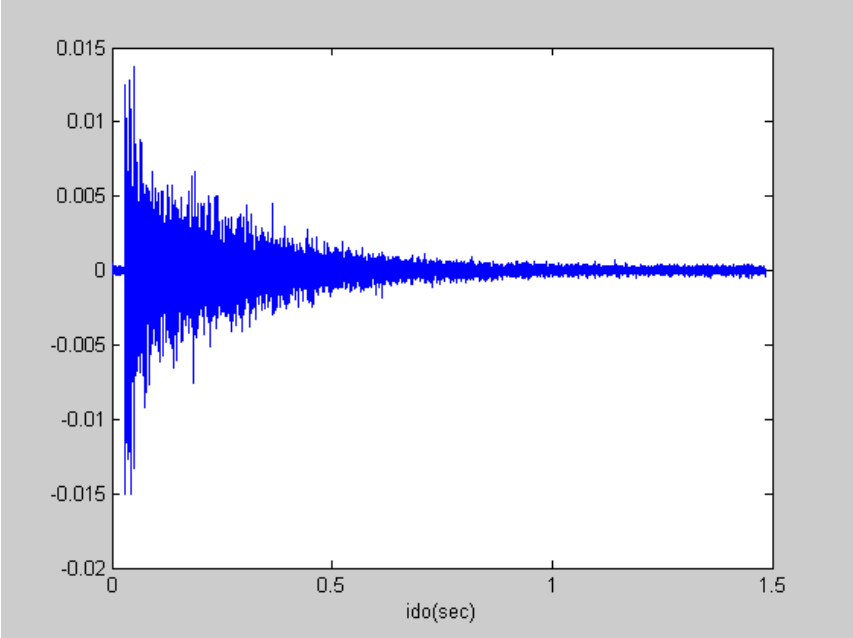
A mérési eredmények 4.2.2.3 pontban leírt feldolgozási módszerével kapott, a termek egy-egy mérési pontjához tartozó impulzusválaszokat a 32-35. ábrák mutatják.



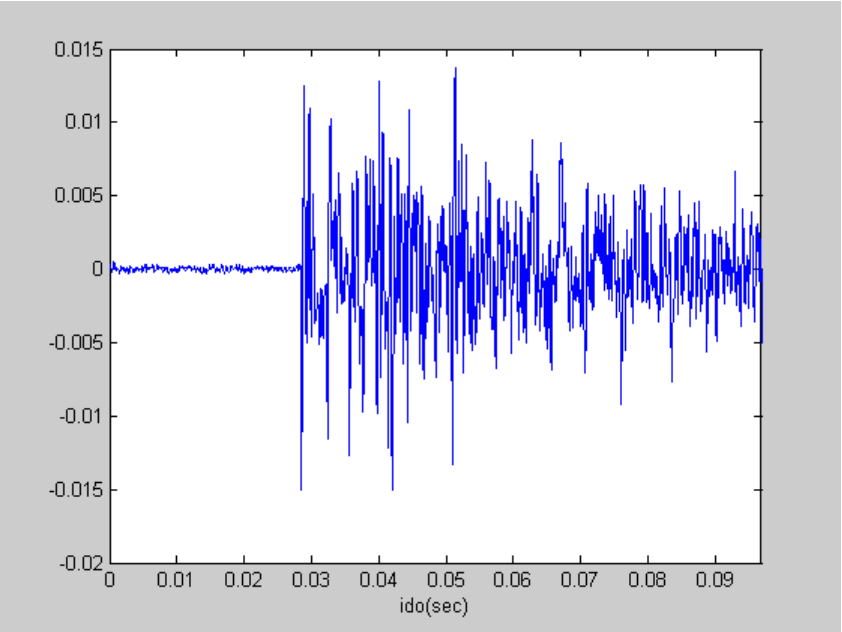
32. ábra Az IB027 előadó 43-as jelzésű pontjában mért impulzusválasz



33. ábra Az IB027 előadó 43-as jelzésű pontjában mért impulzusválasz eleje kinagyítva



34. ábra Az IB028 előadó 48-as jelzésű pontjában mért impulzusválasz



35. ábra Az IB028 előadó 48-as jelzésű pontjában mért impulzusválasz eleje kinagyítva

Mindkét teremben 0,5–1 másodperc közötti impulzusválaszok mérhetők, ami normálisnak tekinthető ekkora méretű termeknél, akkor is, ha azok előadóteremként funkcionálnak. A jó beszédérthetőség nemcsak az impulzusválasz hosszától és az utózengési időtől függ, hanem az energia időbeli eloszlásától is. Jóval rövidebb impulzusválaszt kapnánk, ha hallgatók által megtöltött termet mértünk volna.

A kinagyított képeken jól elkülöníthetők az első, korai visszaverődések. Az impulzusválasz vége azonban elég zajos, ez a mérési eljárás további finomításával valószínűleg javítható.

8 Összefoglalás

8.1 Értékelés

Dolgozatom keretében végeztem irodalomkutatást, áttekintettem a teremakusztikai alapfogalmakat és termek impulzusválaszának mérési módszereit, majd pedig magam is végeztem méréseket az egyetem két előadótermében.

Bemutattam néhány zengetési módszert, amelyek közül Moorer algoritmusát valósítottam meg egy nyolc-csatornás jelfeldolgozó kártya ADSP-21065L típusú processzorán. A megvalósított algoritmus kétféle felhasználói felületen keresztül paraméterezhető. Az elkészült rendszer hátránya, hogy a DSP korlátozott memória méret és a zengető algoritmus fokozott memóriaigénye miatt a zengető 16 kHz-es mintavételi frekvencián működik, ebből kifolyólag 8 kHz-ig visz át, ami audio alkalmazásoknál manapság már nem kielégítő. Szintén a memória kis méretéből következik, hogy a zengetést megvalósító szűrők késleltetésének hossza erősen korlátozott, ezért az így létrehozható utózengési idő is korlátozott.

Az érzeti paramétereket beállító felület esetében az utózengési időből számított szűrőparaméterek egy része jelenleg a Moorer által megadott értékeket kapja meg, ami nem pontos, mert ezek az értékek a javasolt ideális szűrő értékekhez tartoznak. Ezeknek a paramétereknek a meghatározása további megfontolásokat igényel.

8.2 Továbbfejlesztési lehetőségek

Az előző pontban már említett probléma, vagyis a mintavételi frekvencia alacsony értéke valószínűleg csak egy másik jelfeldolgozó processzor választással oldható meg, amely jóval nagyobb belső memóriával rendelkezik.

További feladat lehetne az elvégzett mérések eredményeinek szűrőparaméterekké konvertálása, amely a korai visszaverődések szakaszára viszonylag egyszerűbb, a késői szakaszra nézve azonban részletesebb vizsgálatot igényel. A kiszámított paramétereket beépítve a felhasználói felületbe standard beállítási lehetőségekkel bővülhetne a zengető.

9 Irodalomjegyzék

[1.] W. G. Gardner (1998). Reverberation Algorithms. Applications of Digital Signal Processing to Audio and Acoustics, M.Kahrs és K.Brandenburg, Kluwer Academic, Chapter 3, pp. 85-131.

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

[3.] S. Browne (2001). Hybrid Reverberation Algorithm Using Truncated Impulse Response Convolution and Recursive Filtering.Research project.

[4.] W. G. Gardner (1992). A Realtime Multichannel Room Simulator. J. Acousti. Soc. Am., vol. 92 (A), 2395

[5.] J. A. Moorer (1979). About This Reverberation Business. Computer Music Journal, vol. 3, no. 2, pp. 13-28.

[6.] J. Dattoro (1997). Effect Design. Part 1: Reverberator and Other Filters. J. Audio Engineering Society, vol. 45, no. 9, pp. 660-684.

[7.] J. M. Jot, A. Chaigne (1991). Digital Delay Networks for Designing Artifical Reverberators. Proc. Audio Eng. Soc Convention Preprint 3030.

[8.] P. Fausti, A. Farina (2000). Acoustic Measurements in Opera Houses: Comparison Between Different Techniques and Equipment. J. Sound and Vibration v. 232, pp.213-229.

[9.] Bogár István (2004). "Nyolccsatornás adatgyűjtő tervezése", diplomaterv, BME-MIT.

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

[11.] F. A. Beltran, J. R. Beltran, N. Holzem, A. Gogu (1999). Matlab Implementation of Reverberation Algorithms. Proc. Digital Audio Effects Workshop '99, Dafx'99 (Trondheim, Norvegia), pp. 91-96

[12.] Sun Microsystems, Inc. http://sun.com.

10 Függelék

10.1 Impulzusválasz mérésre használt eszközök

West Sound DS600 erősítő

dodekaéder hangsugárzó

B&K 4189 mikrofon (1/2’’)

B&K 2695 mikrofon előerősítő

ProPort Ariel Model 656, HAMEG Triple Power Supply (mikrofon erősítő)

Fostex 8 track Digital Recorder D-108

10.2 Táblázatok

Tap	Késleltetés (sec)	Erősítés
0	0	1
1	0,0043	0,841
2	0,0215	0,504
3	0,0225	0,491
4	0,0268	0,379
5	0.0270	0,380
6	0,0298	0,346
7	0,0458	0,289
8	0,0485	0,272
9	0,0572	0,192
10	0,0587	0,193
11	0,0595	0,217
12	0.0612	0,181
13	0,0707	0,180
14	0,0708	0,181
15	0,0726	0,176
16	0,0741	0,142
17	0,0753	0,167
18	0,0797	0,134

1. táblázat Korai visszaverődés Moorer által javasolt paraméterei

Szűrő	Késleltetés (ms)	g_1 (25 kHz)	g_1(50kHz)
Fésű1	50	0,24	0,46
Fésű2	56	0,26	0,48
Fésű3	60	0,28	0,50
Fésű4	68	0,29	0,52
Fésű5	72	0,30	053
Fésű6	78	0,32	0,56

2. táblázat Fésűszűrők Moorer által javasolt paraméterei

Szűrő	Max. késleltetés	Maximális mintaszám(16kHz)
FIR	93,75	1500
Fésű1	51,56	825
Fésű2	57,81	925
Fésű3	62,50	1000
Fésű4	70,31	1125
Fésű5	73,43	1175
Fésű6	76,68	1275
Mindentáteresztő	7,81	125

3. táblázat Szűrők maximális késleltetési értékei

Tap	Késleltetés (sec)	Erősítés
0	0	1
1	0.0043	0.841
2	0.0121	0.621
3	0.0133	0.583
4	0.0215	0.504
5	0.0229	0.491
6	0.0286	0.379
7	0.0270	0.380
8	0.0298	0.346
9	0.0378	0.298
10	0.0413	0.317
11	0.0426	0.273
12	0.0458	0.289
13	0.0485	0.272
14	0.0500	0.251
15	0.0572	0.192
16	0.0587	0.193
17	0.0595	0.217
18	0.0612	0.181
19	0.0633	0.201
20	0.0697	0.165
21	0.0707	0.180
22	0.0708	0.181
23	0.0726	0.176
24	0.0741	0.142
25	0.0753	0.167
26	0.0797	0.134
27	0.0811	0.131
28	0.0814	0.119
29	0.0823	0.111

4. táblázat 30 együtthatós FIR-szűrő adatai

early_delay	0xAA
early_gain	0xAB
comb1_delay	0xA1
comb1_gain1	0xA2
comb1_gain2	0xA4
comb2_delay	0xA5
comb2_gain1	0xA6
comb2_gain2	0xA8
comb3_delay	0xC1
comb3_gain1	0xC2
comb3_gain2	0xC4
comb4_delay	0xC5
comb4_gain1	0xC6
comb4_gain2	0xC8
comb5_delay	0xE1
comb5_gain1	0xE2
comb5_gain2	0xE4
comb6_delay	0xE5
comb6_gain1	0xE6
comb6_gain2	0xE8
allpass_delay	0xEA
allpass_gain	0xEB
Application_Start	0xB5
Application_Stop	0xB6
Start_codecs_Request	0xB0
code_direct_ratio	0xCA
code_early_ratio	0xCB
code_diffuse_ratio	0xCC

5. táblázat Üzenetkódok