Diplomaterv

Nyilatkozat

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

Molnár Károly

Tartalomjegyzék

Kivonat

A dolgozat elosztott jelfeldolgozó rendszerekkel foglalkozik. Ezek olyan speciális beágyazott rendszerek, amelyek a környezet fizikai jelein valós-idejű, online jelfeldolgozást végeznek. A dolgozatban bemutatom az elosztott struktúrájú beágyazott rendszerek előnyeit a hagyományos centralizált felépítésű rendszerekkel szemben, külön kiemelve a jelfeldolgozó rendszerekkel kapcsolatos előnyöket.

Az elosztott struktúrájú rendszerek egyik legfontosabb kérdése a csomópontok közötti kommunikáció megvalósítása. Ennek megfelelően ismertetem a kommunikációs protokollokkal szemben megfogalmazott azon követelményeket, amelyek a hard-real time jelfeldolgozó működésből adódnak. A megfogalmazott követelmények szemszögéből megvizsgálom a beágyazott rendszerekben elterjedten használt protokollokat. Kiemelten foglalkozom az Ethernet hálózati technológiával, megvizsgálom, hogy milyen feltételekkel használható elosztott beágyazott rendszerekben. Bemutatásra kerül egy új Ethernet alapú protokoll, a MAGIC, amely stúdiótechnikai elosztott jelfeldolgozó rendszerek számára készült.

Ez után a megvalósított egyszerű jelfeldolgozó rendszer bemutatása következik, amelynek megépítéséhez két DSP alapú fejlesztői kártyát használtam fel. A két DSP szinkron soros porton kommunikál egymással. A rendszeren egy jelfeldolgozó alkalmazást, egy adaptív Fourier-analizátort is megvalósítottam, annak érdekében, hogy bizonyítsam a rendszer helyes működését.

A dolgozat utolsó része egy elosztott jelfeldolgozó rendszerekre jellemző problémával, a mintavétel szinkronizálással foglalkozik. A probléma lényege, hogy az enyhén eltérő mintavételi frekvenciával dolgozó csomópontok összehangolt működése a jelfeldolgozási algoritmusok online működése miatt nem lehetséges. A probléma részletes vizsgálata után bemutatom ennek egy a digitális jelek interpolálásán és újramintavételezésén alapuló megoldását. A kidolgozott megoldásnak három különböző változatát implementáltam a rendszeren, a működő rendszeren méréseket végeztem. A mérések bizonyítják a kidolgozott megoldások elméleti helyességét.

Abstract

This master's thesis is about a special kind of distributed embedded systems, the distributed signal processing systems. These systems perform online digital signal processing on the sampled signals of the physical world. In the 2nd chapter of the thesis the advantages of distributed computing in the embedded field are considered, pointing out the special possibilities of distributed signal processing.

In the field of distributed systems, the most important issue is the communication among the different nodes. In the 3rd chapter the most important requirements of communication due to the hard real-time operation of the system are collected. This is followed by a survey of the widely spread protocol standards in the field of embedded systems.

The whole 4th chapter is about the Ethernet technology. The properties of the Ethernet based communication is considered from the viewpoint of the above stated requirements. The goal of this chapter is to investigate the possibilities of using Ethernet in distributed signal processing systems. A new protocol standard, the MAGIC is presented afterwards, as this is a typical signal processing-oriented, Ethernet-based protocol.

In the 5th chapter the built test-system is presented. This system is a simple distributed signal processing system, comprising two DSP boards. The DSPs communicate via a synchronous serial line. A signal processing algorithm, the adaptive Fourier-analyser is implemented on the test-system in order to prove the proper functionality of the built system.

The last chapter of the thesis is about a typical distributed signal processing problem. The synchronization of the sampling clocks at the different nodes of the distributed system is a common problem for the distributed signal processing systems, as with asynchronous sampling the consistent operation of the nodes are impossible. To resolve this problem, a generally utilizable solution is proposed, based on interpolation and resampling. Three different realizations of this solution is implemented on the test system. The correctness of the solution is proved by measurements.

1. fejezet

Bevezető

A beágyazott rendszerek területén megfigyelhető tendencia az elosztott architektúrájú rendszerek térhódítása. A közeljövőben egyre inkább előtérbe kerülnek az olyan beágyazott rendszerek, amelyek a hagyományos centralizált rendszerekkel szemben a számítógéphálózatok mintájára működő, autonóm és egyenrangú egységekből épülnek fel.

A diplomaterv az elosztott beágyazott rendszerek egy speciális csoportjával, az elosztott jelfeldoldozó rendszerekkel foglalkozik. Ezek olyan hard real-time rendszerek, amelyekben több együttműködő egység végez valós idejű digitális jelfeldolgozást a környező fizikai világ jelein.

Az elosztott jelfeldolgozó rendszerek a szenzorhálózatokból fejlődtek ki. A szenzorok környezetében elhelyezett jelfeldolgozó processzorok (DSP-k) a folyamatos technológiai fejlődésnek köszönhetően mára ugyanolyan képességekkel is rendelkeznek, mint az általános célú beágyazott processzorok. Ez a képesség lehetővé teszi azt, hogy fizikailag a szenzorok mellett elhelyezett egységek a teljes rendszer vezérélését is ellássák egy központi vezérlő számítógép nélkül.

Elosztott rendszerekben mindig központi jelentősége van annak, hogy a kommunikáció milyen módon valósul meg az egyes csomópontok között. A kommunikációs protokoll kiválasztása egy nehéz tervezési kérdés, mert a beágyazott rendszerek sokféleségének következtében számos különböző szabványos protokoll létezik. Ígéretes választás a helyi számítógép hálózatok (LAN) körében használt Ethernet technológia alkalmazása. Az Ethernet használata mellett az szól, hogy rendkívül széles körben elterjedt, kiforrott technológia, amely kellően univerzális ahhoz, hogy hard-real time jelfeldolgozó rendszerekben is alkalmazható legyen.

A jövőben egyre inkább erősödni fog az az igény, hogy mind a beágyazott, mind az asztali számítógép rendszerek összekapcsolhatók legyenek egymással, azaz lehetővé váljon a különböző funkciójú rendszerek közötti interakció.

Ennek lehet egy eszköze az Ethernet és az erre épülő TCP/IP technológia.

A dolgozatban ennek megfelelően külön foglalkoztam az Ethernet hálózati technológia vizsgálatával, elsősorban abból a szemszögből, hogy milyen feltételekkel használható elosztott jelfeldolgozó rendszerekben.

Annak érdekében, hogy gyakorlati vizsgálatokat, méréseket lehessen végezni az elosztott jelfeldolgozás területén, egy egyszerű elosztott jelfeldolgozó rendszert meg is építettem. E rendszer alkalmas arra, hogy az elosztott működésből adódó sajátosságok, problémák megvizsgálhatók, illetve különböző elosztott működésű alkalmazások implementálhatók legyenek. A rendszer ezen kívül az Ethernet alapú kommunikáció sajátosságainak szimulációjára is használható.

A dolgozatban részletesen foglalkozom egy tipikusan elosztott jelfeldolgozó rendszerek körében felmerülő problémával, a mintavétel szinkronizációval. E problémának ismertetem egy általánosan használható megoldását, illetve bemutatom ennek konkrét megvalósításait is a megépített rendszeren.

A dolgozat a bevezetője után, a második fejezetben, az elosztott jelfeldolgozás sajátosságait vizsgálom meg. A harmadik fejezetben összefoglalom azon követelményeket, amelyek a jelfeldolgozó rendszer sajátosságaiból adódnak a kommunikációs protokollal szemben, valamint megvizsgálom a beágyazott rendszerek körében gyakorlatban is használatos protokollokat ezen szempontok szerint.

A negyedik fejezet az Ethernet technológiával foglalkozik. Itt főként a harmadik fejezet szempontjai alapján járom körül, hogy milyen feltételekkel alkalmazható az Ethernet technológia elosztott jelfeldolgozó rendszerekben. A fejezet egy létező Ethernet alapú protokoll, a MAGIC ismertetésével zárul.

Az ötödik fejezetben a megvalósított próbarendszer kerül ismertetésre. A rendszeren egy jelfeldolgozó alkalmazást, egy adaptív Fourier-analizátort implementáltam, amely a megépített rendszer működőképességét igazolja.

A hatodik fejezetben részletesen megvizsgálom a mintavétel szinkronizáció problémáját, és ismertetem ennek egy lehetséges megoldását. A megoldást implementáltam a próbarendszeren, működőképességét mérésekkel igazoltam. A mérések során kihasználtam az implementált AFA azon tulajdonságát, hogy periodikus jelek pontos frekvenciamérését teszi lehetővé. A fejezet a mérési eredményeket is tartalmazza.

A dolgozatot az összefoglalás zárja, amely bemutatja az elért eredményeket és ismerteti a további fejlesztési lehetőségeket.

2. fejezet

Elosztott valós idejű jelfeldolgozás

2.1. Elosztott beágyazott rendszerek

A technológiai fejődésnek köszönhetően a mikroprocesszorok ára folyamatosan csökken. Ez a folyamat a beágyazott rendszerek területén is komoly változásokat eredményez, hiszen ma már reális alternatívát jelent a gyors, 32 bites processzorok alkalmazása minden olyan esetben, amely korábban csak 8 bites mikrokontrollerekkel volt gazdaságos.

Ez a folyamat tette lehetővé, hogy a beágyazott rendszerek körében is megjelenjenek az elosztott struktúrájú, kooperatív rendszerek.

2.1.1. Elosztott struktúra

A hagyományos centralizált beágyazott rendszerek alapvetően Master – Slave működésűek. Mindig van egy központi műveletvégző egység, amely a magasabb szintű számítási és vezérlési feladatokat végzi el, míg az érzékelők és beavatkozók közelében elhelyezett kontrollerek csak egyszerű adatgyűjtő funkciót töltenek be.

E rendszerekből nőttek ki az elosztott beágyazott rendszerek, olyan módon, hogy a processzorok árának csökkenésével a kontrollerek helyét egyre jobb képességű beágyazott processzorok foglalták el. A ma rendelkezésre álló 32 bites, néhány tíz MIPS számítási kapacitással rendelkező processzorok nemcsak gyorsabbak a korábbi kontrollereknél, hanem alkalmasak olyan magas szintű funkciók megvalósítására is, amelyek eddig csak a központi egységben voltak elképzelhetőek.

A mai beágyazott processzorok körében már gyakori az operációs rendszer megléte, a hálózati kommunikáció, a programok fejlesztése pedig a PC-s

világban megszokott fejlesztőeszközök segítségével történik. Általában véve igaz, hogy az asztali számítógépek és a beágyazott processzorok körében használatos technológiák sok tekintetben konvergenciát mutatnak.

Ennek a fejlődésnek eredménye, hogy kialakultak az olyan elosztott beágyazott rendszerek, amelyek a számítógép-hálózatokhoz hasonlóan több, egymással kooperáló, autonóm és egyenrangú processzorokat (csomópontok, node) tartalmaznak. E rendszerekkel már több mint tíz éve foglalkoznak, de igazán széles körben csak az utóbbi években terjedtek el a folyamatos technikai fejlődéssel párhuzamosan [Stankovic, 1988].

2.1.2. Elosztott rendszerek csoportosítása

Az elosztott rendszerek több szempontból csoportosíthatók. Egy rendszer *homogén*, ha minden csomópont egyenértékűnek tekinthető. *Heterogén* rendszerek esetén a csomópontok eltérnek egymástól funkcionálisan és/vagy nem azonos teljesítményűek. Egy rendszer *dinamikus*, ha futás közben képes strukturális változásokra, tehát képes kezelni csomópontok kiesését és reintegrálódását. A *statikus* rendszer struktúrája még a tervezés fázisában kialakul, és futás közben nem módosul.

2.1.3. Az elosztott struktúra előnyei, hátrányai

Az elosztott struktúra számos előnnyel rendelkezik a centralizált megoldással szemben, amelyek az alábbiakban foglalhatók össze:

- **Komponálhatóság.** Bonyolult, sokfunciós rendszerek építhetők autonóm, külön-külön tesztelt, egyszerű komponensekből. A komponálható rendszer funkcionális struktúrája megyegyezik a fizikai struktúrával oly módon, hogy minden funkciót egy önálló csomópont valósít meg (encapsulation). Ez a tulajdonság mind tervezési, mind hibatűrési szempontból előnyös. [Kopetz, 1997, 123–24. oldal]

- **Kiforrott módszerek.** Az elosztott számítástechnika (distributed computing) az informatika egy igen jól körüljárt területe, ezért sok esetben lehetőség van kidolgozott és kipróbált technológiák átvételére, illetve adaptációjára a beágyazott rendszerek sajátosságainak figyelembevételével.

- **Kölséghatékonyság.** Sok esetben a rendszer bizonyos erőforrásai megoszthatók, kiküszöbölve ezzel a felesleges erőforrás többszörözéseket, vagy különösen nagy számításigényű feladatok elvégzése megosztható több processzor között.

- **Hibatűrés.** Megvalósíthatók többszörözésen alapuló önellenőrző struktúrák (watchdog processzor, master-checker, szavazórendszerek).

- **Skálázhatóság.** A beágyazott rendszereknek képesnek kell lenniük környezetük hosszú távú változásaihoz való alkalmazkodásra. Az elosztott struktúra komplexitása új csomópontok rendszerbe integrálásával folyamatosan növelhető az egyes csomópontok komplexitásának növekedése nélkül.

- **Alkalmazkodási képesség.** Egy elosztott rendszer képes adaptálódni a környezet rövidtávú változásaihoz, a rendszer struktúrája képes lehet futás közbeni átalakulásra.

Az előnyök mellett sajnos számolni kell az elosztott struktúrából adódó hátrányokkal is:

- **Kommunikáció.** A csomópontok közti kommunikáció megtervezése nehéz és igen kritikus feladat, hiszen ennek meghibásodása a teljes rendszer működésképtelenségét is eredményezheti.

- **Időbeni szinkronizáció.** Az egységek időbeli szinkoronizációja nélkül a kooperatív működés elképzelhetetlen.

- **Adatbiztonság.** Biztosítani kell az egyes csomópontok adatainak és kommunikációjának integritását.

- **Dinamikus tagság.** Dinamikus rendszerek esetén nagy probléma a csomópontok kiesésének vagy reintegrálásának kezelése.

A dinamikus elosztott rendszerek sok előnnyel bírnak, de tervezésük és megvalósításuk jóval komplexebb feladat mint a statikus rendszereké. A problémát a beágyazott rendszerek hard real-time követelményei teszik igazán nehézzé, hiszen a határidők betartásának követelménye nehezen biztosítható időben kiszámíthatatlanul változó struktúrával. Ez a probléma a beágyazott rendszerek tématerületének egyik aktuális nagy kihívása. Ezért a továbbiakban csak statikus rendszerekről lesz szó, mert ezek egyszerűbbek és egyelőre jobban elterjedtek, mint a dinamikusak.

Elosztott rendszerek tervezése és működtetése általában véve bonyolultabb probléma, mint a korábban említett centralizált rendszereké. Itt is igaz a mérnöki tudományok területén általában helytálló megállapítás, mely szerint az előnyök csak gondos tervezés esetén használhatók ki, míg a hátrányok minden esetben jelentkeznek.

A centralizált rendszerek előnye az lehet az elosztott struktúrával szemben, hogy egyszerűbb esetekben a rendszer kiépítése kisebb költségű, hiszen egy elosztott rendszerben több processzor van, ki kell építeni a kommunikációt a csomópontok között, illetve bonyolultabb probléma a teljes tervezési folyamat.

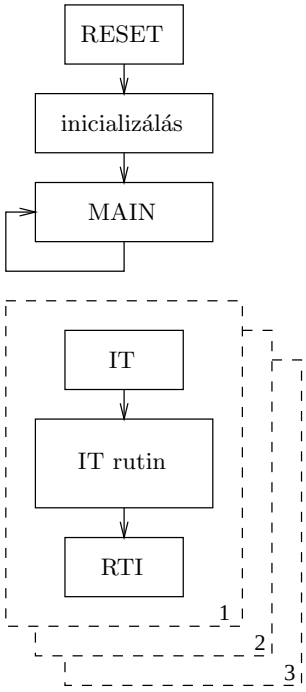
A beágyazott rendszerek azonban minden esetben hosszú távra épülnek (több év, évtized), így számolni kell azzal, hogy újabb és újabb igények merülnek fel, újabb funkciókat kell megvalósítani (skálázhatóság). A rendszer kiegészítésének, módosításának költsége viszont a komplexitás növekedésével centralizált esetekben exponálisan nő, míg elosztott esetben csak lineárisan [Kopetz, 1997]. Ilyen módon van egy bizonyos komplexitási szint, amely felett már megéri elosztott rendszert használni. A technológiai fejlődésnek köszönhetően ez egyre kisebb bonyolultság esetén lesz igaz, tehát a felsorolt nehézségek ellenére a jövőben várhatóan az elosztott struktúra aránya a beágyazott rendszerek körében egyre nagyobb lesz, a centralizált struktúrájú rendszerek pedig csak nagyon egyszerű esetekben kapnak majd szerepet.

2.2. Elosztott jelfeldolgozó rendszerek

Beágyazott rendszerekben a jelfeldolgozó processzorok hagyományosan *slave* üzemmódban, egy központi nagyobb kapacitású *master* számítógép által vezérelve kapnak szerepet. A DSP-k hagyományosan fizikailag az érzékelők és beavatkozók közelében helyezkednek el, és alacsony szintű adatgyűjtő, adatfeldolgozó funkciókat látnak el.

Az utóbbi években megfigyelhető, hogy a beágyazott rendszerekben egyre nagyobb számú érzékelőt és beavatkozót használnak, amely folyamat eredményeképpen létrejöttek az ún. érzékelőhálózatok. Emellett a 2.1. szakaszban vázolt technológiai fejlődés a jelfeldolgozó processzorok körében is megfigyelhető. Az új DSP-k speciális utasításkészletük és jelfeldolgozási feladatokra optimalizált felépítésük mellett ma már sokszor ugyanazokkal a fejlett tulajdonságokkal rendelkeznek, mint a gyors 32 bites mikrokontrollerek [Gustafson, 2002]. Ez az jelenti, hogy a DSP a jelfeldolgozás mellett képes általános vezérlési feladatok ellátására is. Megjegyzés: A DSP-k és a kontrollerek konvergenciája a kontrollerek oldalán is megfigyelhető, már léteznek olyan nagy kapacitású beágyazott processzorok, amelyek speciális jelfeldolgozó utasításkészlettel vannak kiegészítve [Francis, 2001].

A felsorolt fejlődési folyamatok eredményeképpen határozott igény merül fel olyan rendszerekre, ahol az érzékelők és beavatkozók környezetében elhelyezett DSP-k külön vezérlő processzor nélkül működtetik a teljes rendszert. E rendszereket nevezzük elosztott jelfeldolgozó rendszereknek.



2.1. ábra. DSP-k szokásos programstruktúrája

Az elosztott jelfeldolgozó rendszerek az elosztott beágyazott rendszerek egy speciális részhalmazát képezik. Ezek tipikusan heterogén rendszerek, mert általában eltérő számítási kapacitású és funkcionálisan is különböző feladatot ellátó csomópontok működnek együtt. Legtöbbször DSP-k, vagy jelfeldolgozás utasításkészlettel bővített általános célú processzorok vannak a csomópontokban, de gyakran fordulnak elő FPGA vagy ASIC megvalósítású jelfeldolgozó áramkörök, processzormagok.

2.2.1. Online működés

Az elosztott jelfeldolgozó rendszerek legfontosabb sajátossága az *online* működés, amely azt jelenti, hogy az egyes csomópontok folyamatosan, mintáról-mintára, a valós időben számítják kimeneteik értékét a bemenetek alapján. A bemenetek általában különböző szenzorok analóg jeleinek AD átalakítók által előállított mintáit jelentik, a kimenetek pedig olyan digitális jelek, amelyek DA átalakítókon és beavatkozókon keresztül a környezetet befolyásolják. Az online működés DSP-kben szokásos megvalósítását a 2.1 ábra szemlélteti.

A bekapcsolás után egy általános inicializálás történik, majd a program futása a főprogramba kerül, amely egy üres, végtelen ciklus. A jelfeldolgozó algoritmus az AD átalakító interrupt rutinjában van megvalósítva, tehát ez

a rutin az AD mintavétele után, a mintavételi frekvenciának megfelelően periodikusan lefut. Ez a rutin mindig az aktuálisan beérkezett bemenetekből állítja elő a kimeneteket. Ilyen módon a DSP-k működése periodikus, hiszen ugyanaz a számítási rutin fut le minden mintavétel után. Ez a struktúra azért kedvező, mert a jelfeldolgozási feladatok (szűrés, szabályozások) általában felírhatók olyan alakban, hogy a kimenet aktuális értéke a bemenet aktuális és korábbi értékeitől függ. Ez tulajdonképpen nem más, mint a jól ismert állapotváltozós leírás.

E működési mód különbözteti meg a jelfeldolgozó rendszereket a többi beágyazott rendszertől amellett, hogy a vizsgált elosztott rendszerek esetén a programstruktúra valamelyest bonyolódik, hiszen a csomópontoknak járulékos kommunikációs és vezérlési feladatokat is ellátnak.

Az online működés a csomópontokon futó program struktúráját alapvetően meghatározza. Az egyes csomópontok legfontosabb task-ja (a jelfeldolgozó algoritmus) periodikusan minden mintavétel után lefut. Mivel ez a task a legfontosabb, ezért a többi járulékos task-ot is ennek futásához érdemes szinkronizálni. Ilyen módon periodikus ütemezés valósul meg, amely a mintavevő áramkör valós idejű órája (real-time clock, RTC) vezérel.

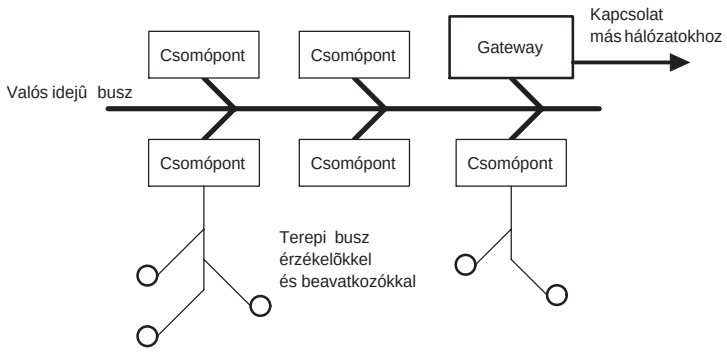
3. fejezet

Kommunikáció elosztott jelfeldolgozó rendszerekben

3.1. Kommunikáció beágyazott rendszerekben

3.1.1. Architektúra

Elosztott rendszerekben mindig központi kérdés az egyes egységek közötti kommunikáció. A közös kommunikációs csatorna az elosztott rendszer kritikus eleme, hiszen ennek meghibásodása a teljes rendszer működésképtelenségét eredményezi, valamint a kommunikációs protokoll alapvetően meghatározza az elosztott rendszer tulajdonságait és képességeit. A beágyazott rendszerek egy általános architektúráját szemlélteti a 3.1. ábra. Az ábra forrása: [Kopetz, 1997].



3.1. ábra. Beágyazott rendszerek architektúrája

Általános beágyazott rendszerek esetén megkülönböztetjük a terepi buszt (field bus) a valós-idejű busztól (real-time bus). Ez utóbbi kapcsolja össze a csomópontokat, míg a terepi busz funkcionális értelemben a megfelelő csomópontok részének tekinthető.

Az ábrán feltüntetett beágyazott rendszer rendelkezik egy olyan gateway csomóponttal, amely megvalósítja az összeköttetést a rendszer és más kommunikációs rendszerek, hálózatok (backbone network) között. Ezzel a tulajdonsággal nem minden beágyazott rendszer rendelkezik, de általában véve kijelenthető, hogy egyre nagyobb igény van a különböző hálózatok összekapcsolhatóságára. A jövőben ez a lehetőség egyre inkább előtérbe fog kerülni, a fejlődés egy mindenhol jelenlévő, minden alrendszert egyesítő hálózat felé tart. Ezt a folyamatot bizonyítja a valós-idejű kommunikációs hálózatok és az Internet hálózati technológiáinak konvergenciája is.

A beágyazott rendszer csomópontjait összekötő valós-idejű busz az a kommunikációs csatorna, amely az elosztott rendszer meghatározó eleme, az alábbiakban ennek vizsgálatával foglalkozom.

A valós idejű kommunikációs csatorna fizikailag egy közös erőforrást jelent (pl. csavart érpár), tehát a kommunikáció *multicast* jellegű, azaz minden csomópont minden üzenetet megkap. Ezek után a csomag tartalma alapján kell a megfelelő csomópontnak felismernie, ha a csomag neki szólt.

3.1.2. Követelmények

Hard real-time követelmények

A beágyazott rendszereknek eleget kell tenniük a hard real-time (HRT) követelményeknek, ezért a kommunikációs protokollnak is elsődlegesen ezt kell támogatnia. Ez konkrétan azt jelenti, hogy az üzeneteknek lehetőleg kis késleltetéssel kell megérkezniük a címzetthez, lehetőleg minimális jitterrel. A csomagvesztéseket szintén minimalizálni kell, de mivel ennek lehetőségét nem lehet kizárni, ezért fel kell készülni a csomagvesztés tényének felismerésére és a katasztrófamentes működés biztosítására (pl. Positive Acknowledge or Retransmission, PAR protokollok használata). HRT rendszerekben még csúcsterhelés esetén sem szabad lekésni a határidőket, tehát a protokoll késleltetését lehetőleg függetleníteni kell a terhelés mértékétől, de a minimális követelmény, hogy a csúcsterhelés esetén megnövekvő kommunikációs késleltetések se okozhassák a határidők lekésését.

Komponálhatóság és flexibilitás

Komponálható rendszer minden egysége (komponense) külön validálható kell legyen. Ez a kommunikációs csatornára is igaz, ilyen szempontból ez is a rendszer egyik komponense. A kommunikációs rendszernek továbbá támogatnia kell új csomópontok rendszerbe integrálását anélkül, hogy a teljes rendszer működési tulajdonságai érezhetően romlanának. A HRT működés nem függhet a csomópontok számától.

Hibatűrés

Követelmény a kommunikáció időviszonyainak jósolhatósága, illetve biztosítani kell a hibák detektálhatóságát és fel kell készülni a kezelésükre. Továbbá az is fontos, hogy egy meghibásodott csomópont hibája ne tudjon továbbterjedni a kommunikációs csatornán át a teljes rendszerre, ezért a hibás csomópontokat vagy fürtöket (cluster) le kell választani és kizárni a kommunikációból.

3.2. Elosztott jelfeldolgozás szempontjai

3.2.1. Architektúra

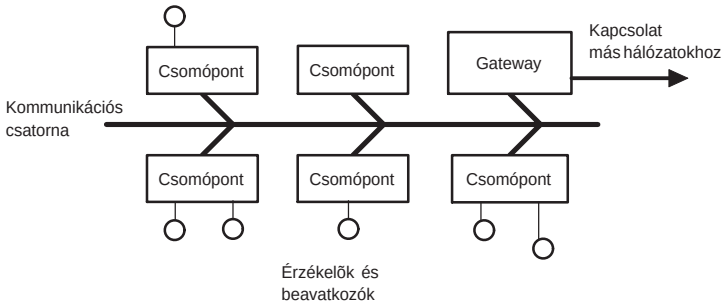
Az elosztott jelfeldolgozó rendszerek architektúrája némileg eltér a 3.1. ábrán bemutatottól. Az elosztott jelfeldolgozó rendszerek a szenzorhálózatokkal mutatnak rokonságot, így a csomópontokat összekötő kommunikációs hálózat egyesíti a terepi buszt és a valós-idejű kommunikációs hálózatot, mindkettő szerepét betölti. A jelfeldolgozó rendszerek általános struktúrája a 3.2. ábrán látható. Ez az architektúra némileg egyszerűbb, mint a 3.1. ábrán bemutatott, de a kommunikáció megtervezése bonyolultabb feladat, mint az előző esetben.

3.2.2. Követelmények

A 3.1.2. szakaszban felsorolt követelmények mindegyike fennáll elosztott jelfeldolgozó rendszerek esetén is, illetve a jelfeldolgozás speciális tulajdonságainak megfelelően még további megfontolásokat kell tenni.

Periodikus adatcsomagok

A jelfeldolgozást végző egységek közötti kommunikáció jellege eltér a beágyazott rendszerek körében megszokottól. A csomópontok itt tipikusan nem vezérlőjeleket vagy státuszinformációkat küldenek egymásnak, hanem valamely



3.2. ábra. Elosztott jelfeldolgozó rendszerek architektúrája

valós-idejű jel mintáiból álló jelfolyamot. A csomópontok közötti összeköttetés szerepe egy virtuális vezeték, amelyen folyamatosan áramlik a digitális jel. Ez a csatornára nézve azt jelenti, hogy minden csomópont a mintavételi frekvenciájának megfelelő ütemben küld csomagokat a többi csomópont felé. Az egyes csomópontok mintavételi frekvenciái a legtöbb esetben megegyeznek, de elképzelhetők különböző mintavételi frekvenciájú csomópontok is. A különböző csomópontok mintavételezésének szinkronizálásával foglalkozik a dolgozat 6. fejezete.

Quality of Service, QoS

Az ilyen jelfolyam jellegű kommunikáció hasonló a valós-idejű hang- és képtovábbítás TCP/IP hálózaton keresztül történő továbbításához, amelynek minőségi követelményeit a Quality of Service (QoS) követelmények rögzítik. Elosztott multimédia hálózatok QoS követelményeivel kapcsolatban rendkívül sok tanulmány és kutatási eredmény látott napvilágot az utóbbi években [Aurrecoechea et al., 1998].

Ezek a módszerek elosztott jelfeldolozó rendszerek esetén sajnos csak korlátozottan használhatók, hiszen az itt szokásos bufferelés nem alkalmazható például visszacsatolást tartalmazó szabályozási körök esetén a túl nagy késleltetés miatt. További gyakorlati probléma, hogy az alacsony költségű beágyazott jelfeldolgozó processzorok egyelőre nem rendelkeznek olyan magas szintű lehetőségekkel és erőforrásokkal, mint a szokásos multimédia munkaállomások. E problémával részletesebben foglalkozik [Gonzalez et al., 1996].

Sporadikus csomagok

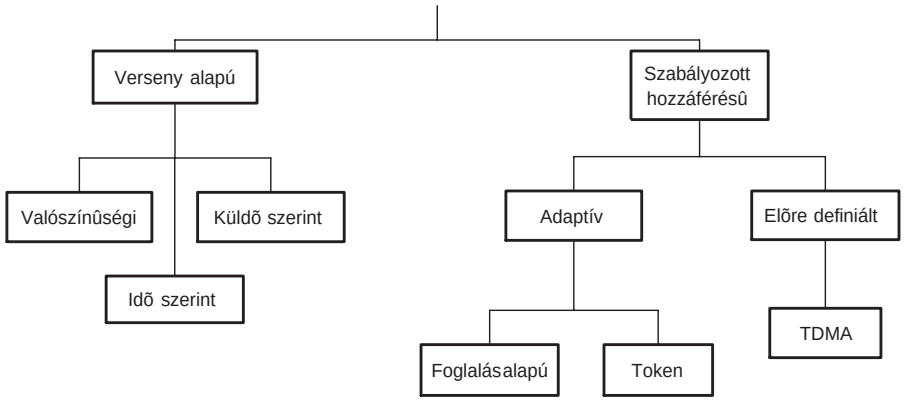
Az elosztott jelfeldolgozó rendszerekben a jelfolyam mellett természetesen vezérlési és státuszinformációt tartalmazó üzenetek küldésére is szükség van. Az

ilyen üzeneteket is a mintavételhez szinkronizálva, periodikusan kell küldeni (esetleg az adatcsomagokkal együtt), tehát a sporadikus csomagküldést el kell kerülni. Ugyanis a kommunikációs csatornán alapvető konfliktust jelent, ha periodikus és sporadikus csomagok egyszerre vannak jelen, hiszen ezek ütközése elkerülhetetlen. A véletlenszerűen bekövetkező ütközések a periodikus adatfolyamot nem jósolható módon feltartják, a szinkronizált, ütemezett működést felborítják [Kopetz, 1997, 158–159. o.]. A sporadikus csomagküldés további problémája, hogy nehezen illeszthető bele a csomópontok online, azaz mintavételhez szinkronizált működésébe az ilyen csomagok beérkezésének lekezelése.

3.3. Protokollok összehasonlítása

Az elosztott beágyazott rendszerek sokféleségének köszönhetően a körükben használt kommunikációs protokollok is meglehetősen sokfélék. Beágyazott rendszerekben az adott alkalmazástól függően más és más protokollt használnak. Ennek megfeleleően sok szabványos protokoll létezik, de ezek között nem igazán megoldott az átjárhatóság, egyelőre messze állunk a mindent körülvevő, egységes kommunikációs hálózat megvalósulásától.

A protokollok [Kurose et al., 1984]-alapján két fő csoportra oszthatók, a szabályozott hozzáférésű, illetve a tartalomalapú protokollokra. A felosztást szemlélteti a 3.3 ábra. Ez a felosztás némileg eltér [Kopetz, 1997, 165.o.]-ben ismertetett eseményvezérelt (ET) - idővezérelt (TT) felosztástól.



3.3. ábra. Protokollok felosztása

3.3.1. Szabályozott hozzáférésű protokollok

Az első nagy csoportot a szabályozott hozzáférésű, *(Controlled Access)* protokollok alkotják, amelyek használatával a közös csatornán ütközés sohasem fordul elő. Ezt olyan módon érik el, hogy minden időpillanatban szigorúan csak egy csomópont adhat a közös csatonán. Ezek a protokollok két további csoportba sorolhatók az alapján, hogy a csatornához való hozzáférési jog a csomópontok igényeinek függvényében változik vagy attól független, azaz előre definiált.

TDMA, TTP

Az előre definiált hozzáféréses protokollok csoportjába a széles körben elterjedt időosztásos működésen alapuló protokollok (Time Division Multiple-Access, TDMA) tartoznak. Ez a protokollcsalád tipikusan idővezérelt (Time Triggered, TT), mivel az egyes csomópontok adási joga a valós idő által vezérelve történik. A valós idő még tervezési időben felosztásra kerül annyi időszeletre, ahány csomópont van a rendszerben. Minden csomóponthoz egy időszelet van hozzárendelve, amelynek időtartama alatt csak az adott csomópont adhat. Ilyen módon egy csomópont periodikusan kap lehetőséget adásra. A helyes működés feltétele az, hogy minden csomópontban rendelkezésre álljon a globális idő, tehát a csomópontok óráit valamilyen módon szinkronizálni kell [Kopetz, 1997, 45-70. o.], [Kopetz and Ochsenreiter, 1987].

A TDMA protokoll nagy problémája, hogy a csatornakihasználtság potenciálisan nagyon rossz is lehet. Ugyanis, ha egy csomópont egy konkrét időszeletben nem kíván adni, akkor a csatorna kihasználatlan marad az időszelet teljes tartama alatt. Ez a probléma hatványozottan jelentkezik, ha a csomópontok kommunikációs igénye nagyon eltérő, vagy ha a kommunikáció sporadikus vagy *burst* jellegű.

A TDMA protokolcsalád egy tipikus példája a Time Triggered Protocol (TTP) [Kopetz, 1997, 171–192.o]. Két fajtája létezik, a TTP/C, amely hibatűrő HRT rendszerek számára készült, míg a TTP/A egyszerűbb terepi buszok protokolljaként használható.

A TTP/C protokollt implementáló rendszerben minden csomópont két részből áll, a *host* számítógépből és a kommunikációs vezérlőből. A kettő közötti interfész a CNI (Communication Network Interface). A kommunikációs vezérlő memóriája tartalmaz egy üzenet leíró listát (Message Description List, MEDL) amely tartalmazza kommunikáció "menetrendjét", azaz hogy mikor melyik csomópont jogosult adásra.

Ez az architektúra támogatja a komponálhatóságot, mert a kommunikációs vezérlő teljesen autonóm (független a *host*-tól), a működését a MEDL és

a globális óra vezérli. A MEDL-hez az egyes csomópontok nem férnek hozzá, illetve külön vezérlő jelek sincsenek a buszra vezetve, így a csomópontok esetleges hibája nem terjed tovább a kommunikációs csatornán keresztül.

A rendszerben a hibadetektálás *fail-silence* módon történik, azaz a hibás csomópont úgy ismerhető fel, hogy a MEDL által előírt időszeletben nem küld csomagot.

Adaptív időosztásos protokollok

A TDMA protokoll módosításával alakultak ki az adaptív időosztásos protokollok. A módosítások célja a csatornakihasználtság javítása oly módon, hogy az adni nem kívánó csomópontok ne tartsák fel az adásra kész csomópontokat.

A foglalási alapon működő rendszerekben adási és a foglalási időszeletek váltják egymást. A foglalási időszakban minden éppen adni kívánó csomópont jelzi adási szándékát. Ha több ilyen van, akkor valamilyen prioritási alapon eldől, hogy melyik csomópont lesz adásra jogosult a következő adási időszeletben. A megoldás hatékonysága nagyban függ a megvalósított döntési algoritmustól, mert például fix prioritású csomópontok esetén a magasabb prioritásúak "kiéheztethetik" az alacsonyabb prioritásúakat.

A gyakorlatban sokkal elterjedtebb a *token-passing* protokoll, amelynek lényege az, hogy a csomópontok valamilyen meghatározott sorrend szerint "körbeadnak" egy virtuális tokent, és mindig az a csomópont jogosult adásra, amelyiknél a token éppen van. Ha a token olyan csomóponthoz kerül, amelyik éppen nem szándékozik adni, akkor az rögtön továbbadja a tokent a sorban utána következőnek.

A csatornakihasználtság ezzel a módszerrel javul, de így sem lesz optimális, mert az adni nem kívánó csomópontok továbbra is feltartják az adni kívánókat a token továbbításnak időtartama alatt. A TDMA megoldással szemben egyértelmű hátrányt jelent, hogy a token "elveszhet". Ez úgy lehetséges, hogy egy meghibásodott csomópont nem adja tovább a tokent, amely a teljes rendszert leblokkolja.

3.3.2. Verseny-alapú protokollok

A verseny-alapú (*Contention-based*) protokollok legfontosabb tulajdonsága az, hogy a közös csatornán előfordulhat *ütközés*, amely akkor történik, amikor kettő vagy több csomópont egyidejűleg kíván adni. A működés lényege az, hogy amikor a csatorna szabad, akkor a csomópontok bármikor megkísérelhetnek küldeni egy csomagot, annak reményében, hogy másik csomópont nem kívánja használni a csatornát. A korai protokolloknál nem ellenőrizték

a csatorna foglaltságát az adás megkezdése előtt, így gyakrabban történt ütközés.

A ma használt *carrier-sense* tulajdonságú protokolloknál mindig megtörténik ez az ellenőrzés. Ütközés esetén valamilyen algoritmus alapján eldől, hogy melyik csomópont jogosult adásra. A döntő algoritmus működhet valószínűségi alapon, a csomag küldőjétől függően, vagy valamely időadat alapján.

Időadat alapján döntő protokollok

Ez utóbbi protokollok esetén a működés lényege az, hogy használatlan csatorna esetén az adásra jogosult csomópont kiválasztása valamilyen időadat alapján történik. Ez lehet az üzenetküldés időpontja, az üzenet hossza, az üzenet *laxity* (az üzenet küldésének leghamarabbi befejezése és határideje közötti időtartam), vagy az üzenet határideje.

Ezeknél a protokolloknál az adásra jogosult csomópont kiválasztása lényegében ugyanolyan algoritmusok alapján történik, mint amilyeneket egyes valós idejű operációs rendszerekben (RTOS) használnak task-ok ütemezésére. Ilyen algoritmusokra példa az EDF (Earliest Deadline First), vagy az LL (Least Laxity).

Időadat alapján döntő protokollcsalád a VTCSMA (Virtual Time Carrier Sense Multiple Access) [Zhao and Ramamritham, 1987]. A csomópontok mindegyike rendelkezik egy virtuális timer áramkörrel, amely a valós időnél nagyobb sebességgel jár. A kommunikációs csatorna felszabadulása esetén minden adni kívánó csomópont elindítja a saját virtuális timer-jét, és akkor küld üzenetet, amikor ennek értéke elér valamely értéket, például a csomag küldésének határidejét. Ezzel az EDF algoritmus valósul meg, hiszen mindig az a csomópont fogja leghamarabb lefoglalni a csatornát, amelyiknek leghamarabb esedékes a határideje.

CAN

A Bosch cég által kifejlesztett és az autóiparban széles körben használt szabványos protokoll, a CAN (Control Area Network) egy tipikus példája a küldő alapján döntő verseny-alapú protokolloknak [CAN, 1991]. A működés lényege az, hogy az egyes üzenetekhez külön prioritási szintek vannak rendelve, és ütközés esetén az alacsonyabb prioritású csomag küldőjének be kell fejeznie az adást. Ezt a működési elvet CSMA/CA-nak (Carrier Sense Multiple Access with Collision Avoidance) is nevezik.

A CAN protokollt kifejezetten gyors, robosztus kommunikáció megvalósítására fejlesztették ki. Tipikusan olyan beágyazott rendszerek esetén kedvező a használata, amikor sok egyenértékű csomópont viszonylag rövid üzenetek-

kel kommunikál. A CAN előnyös tulajdonsága továbbá, hogy rendelkezik bizonyos hibatűrő mechanizmusokkal (*Fault-Confinement*). Ez lényegében azt jelenti, hogy minden csomópont rendelkezik egy hibaszámlálóval, amelynek bizonyos értéke felett a csomópont Bus-Off állapotba kerül, azaz sem üzenetek adására, sem fogadására nem jogosult. A csomópont leválasztása nem végleges, létezik egy előírt procedúra, amely a Bus-Off állapotból reaktiválja a csomópontot bizonyos feltételek teljesülése esetén.

CSMA/CD, LON

A verseny-alapú protokollok valószínűségi alapon döntő alcsoportjába tartoznak a CSMA/CD (Carrier Sense Multiple Access with Collision Detect) protokollok. Ezek klasszikus példája az Ethernet protokoll, amellyel a 4. fejezetben részletesen foglalkozom.

A CSMA/CD protokollok egy másik példája az Echelon Corporation által fejlesztett LonWorks szabványos (ANSI/EIA 709.1) protokoll [LON, 1999]. A LonWorks speciálisan vezérlő rendszerek (*control systems*) számára készült. A protokoll a [Tannenbaum, 1996] szerint definiált ISO/OSI hét rétegű modell mindegyik rétegét megvalósítja, az alkalmazási réteget is beleértve.

Jelen dolgozat szempontjából leginkább a második, az adatkapcsolati réteg megvalósítása érdekes. Ebben a rétegben vannak definiálva az adatcsomagok küldésének szabályai, illetve az ütközések esetén követendő stratégia. A LonWorks egy egyedi algoritmust használ, az ún. *prediktív p-perzisztens CSMA* protokollt. A p-perzisztens CSMA protokoll részletesebben a 4. fejezetben kerül ismertetésre. Az Echelon Corp. álllítása szerint ez az algoritmus jobb tulajdonságokkal rendelkezik csúcsterhelés esetén, mint az 1-perzisztens CSMA, amelyet az Ethernet valósít meg. A LonWorks által megvalósított algoritmus azért *prediktív*, mert a csatorna terhelésétől függően változtatja a véletlenszerű hozzáférési késleltetések szórását.

A Lonworks rendszer előnyei a skálázhatóság és a komponálhatóság. Ez utóbbi azt jelenti, hogy rendelkezésre állnak olyan validált eszközök, amelyek megvalósítják a LonWorks specifikációt (*off-the-shelf* alkatrészek). A komponálhatóságot támogatja az alkalmazási rétegben definiált "Hálózati Változók" (*Network Variables*) használata is, amelyek olyan univerzális változók, amelyek a rendszer több csomópontja számára is hozzáférhetők. Ezek használata nagy mértékben megkönnyíti a rendszer egyes csomópontjai közötti kommunikáció megszervezését, mert a kommunikáció alacsonyabb rétegei teljesen el vannak rejtve a tervező elől, így annak konkrét megvalósításával nem kell törődnie.

A LonWorks protokollt vezérlő rendszerek számára fejlesztették ki, ezért használata kis méretű csomagok, és sporadikus forgalom esetén optimális.

3.3.3. Egyéb protokollok

Az eddig ismertetetteken kívül érdemes megemlíteni néhány olyan elterjedten használt protokollt, amelyek nem fértek bele a 3.3. ábrán látható kategóriák egyikébe sem.

Master-Slave

A 3.3 ábrán az elosztott rendszerekben használt protokollok vannak csoportosítva, tehát nem szerepelnek a centralizált hálózatokban használt protokollok. A gyakorlatban azért nem ennyire éles a határvonal az elosztott és a centralizált rendszerek között, így főleg egyszerűbb elosztott rendszerekben elterjedten használnak master-slave jellegű kommuniációt.

Ennek lényege az, hogy bármiféle kommunikációt csak egy csomópont, a master kezdeményezhet, a többi csak akkor küldhet adatcsomagot, ha erre a master utasítást adott. Ilyen rendszerek nyilvánvaló gyenge pontja az, hogy a master meghibásodása esetén a teljes rendszer működésképtelenné válik. Előnye az egyszerűségében rejlik, kis rendszerek jól áttekinthető módon működtethetők ezzel a módszerrel.

Az autóiparban használt master-slave protokoll a Local Interconnect Network (LIN) [LIN, 2002]. E protokoll célja a minél egyszerűbb (egy vezetékes kommunikáció) kivitel és az alacsony költség.

FlexRAY

Szintén autóipari felhasználásra tervezik a FlexRAY protokollt, amely jelenleg még fejlesztés alatt áll [FlexRAY, 2002]. Ez a protokoll a TDMA és a verseny-alapú megközelítés egy érdekes keverékét valósítja meg. Ez úgy történik, hogy a működés során periodikusan ismétlődik egy statikus és egy dinamikus időszegmens. A fix hosszúságú statikus szegmens ideje alatt a fentebb ismertetett TDMA működés történik, azaz minden csomópont kap egy azonos hosszúságú időszeletet, amikor csak ő adhat. A változó hosszúságú dinamikus szegmens ideje alatt pedig igény szerint bármelyik csomópont adhat. Az ütközések elkerülése prioritásos alapon történik.

3.4. Összefoglalás

A fenti felsorolás természetesen nem lehet teljes, hiszen az utóbbi évtizedekben a fentieken kívül még számtalan protokollt fejlesztettek ki beágyazott rendszerek számára. A fejezet céja az elterjedten használt protokollok megismerése, illetve a különböző megoldások áttekintése volt.

Az elosztott jelfeldolgozás szempontjából kedvezőek a TDMA protokollok a periodikus működés miatt. A TDMA rendszerek kifejezetten támogatják a HRT követelmények biztosítását, ugyanakkor a rendszerek nem flexibilisek, a helyes működés megszervezése körülményes. További probléma az óraszinkronizáció kérdése, amely a helyes működés elengedhetetlen feltétele. Jelfeldolgozó rendszerek esetén nem feltétlenül jár katasztrofális következményekkel egyes csomagok kiesése, ezért körükben a TDMA protokoll sokszor feleslegesen szigorú, így nem minden esetben érdemes a megvalósítás nehézségeit vállalni.

A vezérlő rendszerek számára kifejlesztett protokollok (CAN, LON) alapvetően nem kedveznek a jelfeldolgozó rendszerek megvalósításának, ezeket elsősorban sporadikus adatforgalomra tervezték. Ennek ellenére vannak kedvező tulajdonságaik is (komponálhatóság, hibatűrés).

Valójában nincs optimális választás, minden esetben kompromisszumot kell kötni. Az adott alkalmazás jellege, komplexitása, a biztonsági követelmények szintje mind befolyásolják a döntést. Mindig figyelembe kell venni a tervező lehetőségeit is. Ez nemcsak az anyagi lehetőségeket jelenti, hanem a már meglévő tapasztalatot, valamint az alkatrészek beszerezhetőségének problémáját.

A választásunk ezen tényezők miatt esett az Ethernet-re, mert univerzális, nagyon széles körben elterjedt hálózati protokoll, amelynek nagyon sok gyakorlati előnye van. A következő fejezetben részletesen foglalkozom az Ethernet protokollal, és a használatának lehetőségeivel az elosztott jelfeldolgozó rendszerek körében.

4. fejezet

Ethernet

4.1. Az Ethernet protokoll

A Xerox cég egyik kutatási projektjének eredményként született meg az Ethernet protokoll és hálózati technológia, az 1970-es években. E kutatás során olyan számítógéphálózatok számára dolgoztak ki kommunikációs protokollt, ahol egy közös kommunikációs médium van, azaz egy koaxiális kábelre csatlakozik minden számítógép. A protokollt alapvetően sporadikus jellegű, esetenként nagy mennyiségű adat továbbítására tervezték.

A gyors kezdeti sikerek után más cégek is bekapcsolódtak a fejlesztésbe (Digital Equipment Corp., Intel), és kidolgozták az Ethernet Version 1.0 specifikációt. Az 1985-ben elfogadott hivatalos szabvány (ANSI/IEEE Std. 802.3 - 1985) ezen a specifikáción alapul, csak minimális eltérés van közöttük. Az azóta eltelt idő alatt a szabványt igen sokszor egészítették ki a hálózati technológiák folyamatos fejlődésével párhuzamosan.

Az Ethernet a helyi hálózatok (Local Area Network, LAN) körében szinte egyeduralkodó, ma világszerte a helyi hálózatra kapcsolt PC-k és munkaállomások 85%-a használja ezt a protokollt [Cisco, 2002]. Beágyazott rendszerek körében egyelőre ennél jóval kisebb arányban használnak Ethernetet, de a jövőben mindenképpen várható az arány növekedése, a beágyazott technológiák és az asztali számítógépek már említett konvergenciájának következtében.

Megjegyzés: a 4.1. alfejezet [Tannenbaum, 1996] és [Cisco, 2002] alapján készült.

4.1.1. Rétegszerkezet

Az Ethernet protokoll a hét rétegű ISO/OSI modell alsó két rétegét, a fizikai, illetve az adatkapcsolati réteget valósítja meg, az erre épülő felsőbb rétegeket nem. Ez a határvonal megfelel a hardver és szoftver szétválasztásnak is, az Ethernet rétegei általában hardver, míg a felsőbb rétegek szoftver megvalósításúak.

Az Ethernet fizikai rétege definiálja a kommunikációs sebességet, az alkalmazott kódolást, tehát általában a kommunikációs csatorna tulajdonságait. Az adatkapcsolati réteg két részre oszlik, a Media Access Control (MAC) és a Logical Link Control (LLC) alrétegekre. Az utóbbi a MAC és a felsőbb rétegek közötti interfész szerepét tölti be.

A MAC alrétegnek két fő feladata van:

- A küldeni kívánt csomagok összeállítása, illetve a beérkezett csomagok felbontása és a hibák detektálása.
- A kommunikációs csatornához való hozzáférés (*media-access*) vezérlése az 1-perzisztens CSMA/CD algoritmus alapján.

4.1.2. Az 1-perzisztens CSMA/CD algoritmus

A CSMA/CD protokollokat azzal a céllal fejlesztették ki, hogy több csomópont tudja ugyanazt a közös kommunikációs csatornát használni anélkül, hogy bármiféle központi eszköz vezérelné a kommunikációt, vagy hogy szükség legyen a csomópontok bármilyen szinkronizálására vagy együttműködésére. Olyan protokoll kifejlesztése volt a cél, amely lehetővé teszi minden csomópont autonóm működését, olyan módon, hogy minden csomópont ugyanazt az algoritmust futtatja, és önállóan dönti el, hogy mikor adhat.

Az 1-perzisztens CSMA/CD működésének lényege az alábiakban foglalható össze. Az adni kívánó csomópont folyamatosan figyeli a csatornát, és ha az szabad, akkor haladéktalanul elkezdi a csomag adását. A csomag adása alatt előfordulhat ütközés, hiszen a jelterjedés ideje véges, így előfordulhat, hogy adni kezd egy másik olyan csomópont is, ahová még nem érkezett meg az első csomópont csomagja. Ha ütközés történik, azt mindkét csomópontnak érzékelnie kell, és az adást be kell fejezniük. Ilyenkor az ütközést elszenvedő csomagok küldése nem volt sikeres, így azokat mindkét csomópontnak újra kell küldenie.

Az 1-perzisztens CSMA/CD algoritmus valójában speciális esete a p-perzisztens CSMA/CD algoritmusnak. Ez utóbbi lényege az, hogy ha egy adni kívánó csomópont érzékeli, hogy a csatorna szabad, akkor p valószínűséggel kezdi meg az adást. (Tehát az Ethernet esetén $p = 1$.) A 3.3.2.

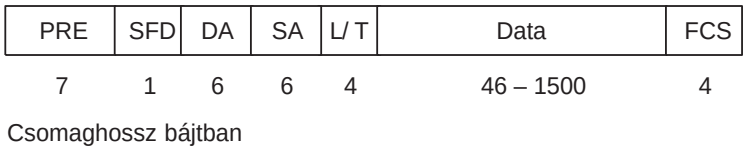
szakaszban ismertetett LonWorks protokoll által használt prediktív p-perzisztens CSMA/CD algoritmus esetén például $p \neq 1$. Nagy forgalom esetén a p-perzisztens CSMA/CD átlagosan kevesebb ütközést produkál, mint az 1-perzisztens, kis forgalom esetén viszont a csomagküldés átlagos késleltetési ideje nagyobb lesz p-perzisztens esetben, mint 1-perzisztens esetben.

4.1.3. Ütközések

Az a követelmény, hogy az ütközést mindkét csomagnak érzékelni kell, közvetlenül befolyásolja a minimális használható csomaghosszt és a hálózat maximális fizikai kiterjedését. Egyrészt a csomag minimális küldési ideje nagyobb kell legyen, mint a csomag terjedési idejének kétszerese a hálózat két legtávolabbi pontja között. Ha ennél kisebb csomagok küldése is engedélyezett, akkor előfordulhat az az eset, hogy az adást hamarabb kezdő csomópont hamarabb fejezi be az adást, mint hogy a később indított másik csomag megérkezik hozzá. Ebben az esetben a csomópont nem érzékelné az ütközést. A gyakorlatban a három paraméter közül az átviteli sebesség adott volt (10 Mbps az alap Ethernet esetén), így egy reális felső korlátot definiáltak a hálózat fizikai kitejedésére (2500 méter). A harmadik paraméter, a minimális csomaghossz ezek után 512 bit hosszúságúra adódott, melynek elküldési ideje 51.2 μsec. Ezt az időtartamot *slot idő*-nek is nevezik.

A CSMA/CD algoritmus nem írja elő a csomópontok ütközés utáni viselkedését. Az Ethernet a "Bináris exponenciális visszatartás" (Binary Exponential Back-off) algoritmust használja. E szerint az ütközés után mindkét csomópont egy előre meghatározott ideig várakozik (9.6 μsec), majd sorsol egy véletlen számot 0 és $2^i - 1$ között, ahol i az aktuális csomag által addig elszenvedett ütközések száma. A csomópontok az előállított véletlen számot beszorozzák a slot idővel, és a kapott időtartamig várakoznak. Ha ez letelt, újra megkísérlik elküldeni a csomagot. Ha $i > 10$, akkor már mindig csak a 0..1023 tartományban történik a sorsolás. Ha i értéke eléri 16-ot, azaz 16 próbálkozás után sem sikerült elküldeni a csomagot, akkor a MAC többet nem próbálkozik. Az adott csomag elküldésének sikertelenségét a MAC *excessive collision* hibaüzenettel jelzi a kommunikációs protokoll felsőbb rétegeinek.

A fenti tulajdonságok miatt az Ethernet egy nem-determinisztikus protokoll, azaz az egyes csomagok továbbítási idejére nem lehet felső korlátot biztosítani. Elvileg az is előfordulhat, hogy a hálózat csúcsterhelés esetén összeomlik, tehát az egyes csomópontok versengése felemésztheti a teljes adatátvitelre szánt időt.



4.1. ábra. Ethernet keretformátum

4.1.4. Keretformátum

Az Ethernet csomagok (más néven keretek) felépítését szemlélteti a 4.1. ábra. A keret egyes mezőinek jelentése:

- **Preamble (PRE)** – Egyesek és nullák váltakozó sosrozatából álló előtag. Szerepe az, hogy a vevő csomópontok szinkronizálódni tudjanak az adóhoz.
- **Start-of-Frame Deliminiter (SFD)** – A keret kezdetét jelző bájt.
- **Destination Address (DA)** – E mező a célcsomópont címét tartalmazza. A vevő csomópontok azonnal el tudják dönteni, hogy az üzenettel mi a teendő (eldobás/feldolgozás). A cím lehet egy csomópont egyéni címe, vagy csoportos cím *broadcast* üzenet esetén.
- **Source Address (SA)** – A küldő csomópont címe. Ez mindig egy egyéni csomópont címe.
- **Length/Type (L/T)** – Ennek a mezőnek más jelentése van az Ethernet, illetve az IEEE 802.3 szabvány szerint. Az Ethernet szabvány szerint ez a mező az üzenet típusát jelenti, ez általában valamely felső rétegbeli protokoll azonosítója (ARP, IP, stb). Az IEEE 802.3 szerint ez a mező a csomag hosszát jelenti. A két keretformátum nincs konfliktusban, mert ha a mező értéke 46 és 1500 (600H) között van, akkor ez egy IEEE 802.3 csomag, ha a mező értéke 600H-nál nagyobb, akkor Ethernet csomag.
- **Data** – Az adatmező jelenti a hasznos terhet. Ennek hossza a 4.1.3-ben leírtak alapján minimálisan 46 bájt. Amennyiben a küldendő adat ennél rövidebb, akkor a MAC kiegészíti a mező hosszát 46 bájt hosszúra (*padding*).
- **Frame Check Sequence (FCS)** – A keretek ezzel a 32 bites CRC ellenőrző összeggel fejeződnek be, amelyek az átvitel bithibáinak felderítését szolgálják.

4.1.5. Ethernet hálózatok felépítése, aktív eszközök

Az IEEE 802.3 szabvány jelenleg három különböző adatátviteli sebességű Ethernet hálózatot ír le. Ezek:

- 10 Mbps – 10BaseT Ethernet
- 100 Mbps – Fast Ethernet
- 1 Gbps – Gigabit Ethernet

A 10 Mbps sebességű Ethernetnek több fajtája is van. Az eredeti 1984-es szabványban a 10Base5 Ethernet szerepel, amely esetén a kommunikációs csatorna egy koaxiális kábel volt. Ezzel nagyjából egyidős a 10Base2, amely egy vékony 50Ω ellenállású koaxiális kábelt használ. Mind a 10Base5, mind a 10Base2 alapú hálózatok elég nehezen menedzselhetők, ezért használatuk mára háttérbe szorult.

A 10 Mbps sebességű Ethernet hálózatok között ma a 10BaseT szabvány szerinti, csavart érpárt alkalmazó Ethernet a leginkább elterjedt. Ennek lényeges tulajdonsága, hogy fizikailag mindig csak két eszköz között létesít pont-pont kapcsolatot, tehát nem üzenetszórásos elven működik. A csomópontok mindegyike fizikailag egy aktív eszközzel, egy ismétlővel (*repeater, hub*) van összekötve. Az ütközéseket is az ismétlő érzékeli, és annak megfelelően hajtja meg a különböző csomópontokhoz tartozó vezetékeket.

A 10 Mbps sebességű Ethernet hálózatokban a kommunikációs közeg nem csak csavart érpár lehet. Három különböző célú optikai kábeles szabványt is kidolgoztak, ezek a 10BaseF-P, a 10BaseF-B valamint a 10BaseF-L.

A 100 Mbps sebességű hálózatok körében a 100BaseTX csavart érpár alapú, valamint a 100BaseF optikai kábel alapú szabvány terjedt el. A csavart érpár alapú 100BaseTX lényegében a 10BaseT-hez hasonlóan működik, csak tízszer olyan gyors. Ebből az a lényeges különbség adódik, hogy az ütközések detektálhatósága miatt a hálózat fizikai kiterjedésének felső korlátját kevesebb, mint a tizedére kellett csökkenteni (körülbelül 200 méter).

Ez a probléma Gigabit Ethernet esetén is jelentkezik. A fizikai méret felső korlátját nem lett volna ésszerű tovább csökkenteni, ezért a szabvány inkább a csomagok minimális hosszát növelte meg 520 bájt hosszúságúra. Amennyiben egy csomag rövidebb, mint ez minimális hossz, akkor a 4.1 ábrán bemutatott keret végére bekerül egy hosszabító mező: *Gigabit Extension Field*.

Az Ethernet hálózatokban elterjedten alkalmaznak aktív eszközöket. A fentebb ismertetett okokból a 10BaseT szabványtól kezdve ezen eszközök a hálózat szerves részét képezik és a működéshez elengedhetetlenek. Az aktív eszközök az alábbi három csoportra oszlanak:

- **Ismétlő** – Az ismétlő egység sok csatlakozási ponttal rendelkezik, amelyekre egy-egy csomópont vagy egy teljes kábelszegmens (pl. 10Base2 esetén) csatlakozik. Az ismétlő minden beérkezett csomagot továbbít a többi csomópont felé. A csomagokat nem vizsgálja, mindössze helyreállítja a fizikai jelalakot. Az ismétlővel összekapcsolt csomópontok nem érzékelik az ismétlő jelenlétét, működési szempontból ugyanúgy egy közös közeg van, amit a többi csomóponttal meg kell osztani. Máshogy fogalmazva az ismétlővel összekapcsolt csomópontok egy *ütközési zónában* vannak.

- **Híd** – A híd (Ethernet Bridge) egy komplexebb eszköz, amely általában szoftverben van implementálva. Alapvetően több kábelszegmens összekapcsolására szolgál. A híd minden beérkezett üzenetet azonosít, és a keretben található célcím alapján csak abba a kábelszegmensbe továbbítja a csomagot, ahol a címzett csomópont van. Azt, hogy melyik csomópont melyik kábelszegmensben van, egy táblázatban tárolja, amelyet a beérkezett csomagok Source Address mezői alapján épít fel automatikusan. A híd lényeges tulajdonsága, hogy az ütközéseket nem terjeszti tovább, tehát a különböző kábelszegmensek különböző ütközési zónákat jelentenek. Ilyen módon túlterhelt hálózati szegmensek hidak beépítésével tehermentesíthetők.

- **Kapcsoló** – A kapcsoló (Ethernet Switch) egy sok csatlakozási ponttal rendelkező, teljesen hardver implementálású, nagysebességű híd. A kapcsolók általában alkalmasak különböző sebességű kábelszegmensek összekapcsolására. A kapcsolók portjain általában várakozási sorok vannak kialakítva, amelyek egy-egy üzenetekkel elárasztott csomópont tehermentesítésére szolgálnak. Ezek hasznossága vitathatatlan, de sajnos a csomagok késleltetési idejét bizonytalanná teszik. A kapcsolók további fontos tulajdonsága, hogy segítségükkel virtuális hálózatok hozhatók létre (VLAN), azaz az egyes csomópontok fizikai elhelyezkedésétől függetlenül létrehozhatók összetartozó csoportok.

Az optikai és a csavart érpárt alkalmazó hálózatok fizikailag külön csatornán bonyolítják az adást és a vételt. Egyes aktív eszközök segítségével ez kihasználható, azaz a fél-duplex működés helyett lehetőség nyílik full-duplex kommunikációra is, azaz egyidejűleg folyhat az adás és a vétel a csomópont és az aktív eszköz között. Ehhez persze az is szükséges, hogy a csomóponton megvalósított MAC réteg is támogassa a full-duplex működést.

4.2. Jelfeldolgozó rendszerek Ethernet alapú kommunikációja

Az előző ismertetésből is kiderült, hogy az Ethernet egy nagyon kiforrott, univerzálisan használható hálózati technológia. Ugyanakkor az is kiderült, hogy az Ethernet csak a kommunikáció alsó két rétegét valósítja meg, ami valójában csak arra biztosít egy módszert, hogy az elosztott rendszer különböző csomópontjai üzeneteket (csomagokat) küldjenek egymásnak. Az Ethernet használata ezért igen nagy mozgásteret hagy a tervezőnek, hiszen egyrészt a rendszer architektúrájára sincs megkötés (sebesség, struktúra, aktív eszközök, fél- vagy full-duplex működés), másrészt az Ethernet rétegeire épülő kommunikáció is tetszőlegesen alakítható ki. A valódi kérdés ezért nem az, hogy az Ethernet használható-e elosztott jelfeldolgozó rendszerekben, hanem az, hogy az Ethernet alapú kommunikációt milyen módon kell kialakítani ahhoz, hogy az elosztott jelfeldolgozó rendszerek speciális követelményeinek megfeleljenek.

Az alábbiakban megvizsgálom, hogy a különböző megvalósítási lehetőségek milyen következményekkel járnak, illetve milyen problémákat vetnek fel elosztott jelfeldolgozó rendszerek esetén. A vizsgálat fő szempontjai a 3. fejezetben leírtak szerint:

- Kis csomagkésleltetés
- Kis késleltetési jitter
- Elveszett csomagok detektálása, a hiba javíása
- Periodikus csomagküldés

4.2.1. Ütközésmentes kommunikáció

Alapvetően meghatározó, hogy a csomópontok kommunikációja során előfordulnak-e ütközések. Ma már reális lehetőség olyan Ethernet alapú hálózat alkalmazása, amelyen nincsenek közös kábelszegmensek, hanem a csomópontok egy Ethernet kapcsolóval vannak összekötve. Ilyenkor ütközés nem fordul elő, hiszen a kapcsolóba érkezik minden csomag, amely ezeket, akár párhuzamosan is, továbbítja a címzett csomópont vagy csomópontok felé. Ilyen hálózatok esetén a csomópont és a kapcsoló között külön vezetéken történik az adás és a vétel (full-duplex kapcsolat), így itt sem történik ütközés. A helyes működés feltétele az, hogy a csomópontok támogassák a full-duplex működést.

Az elosztott jelfeldolgozás szempontjából az ütközésmentes működés legnagyobb előnye az, hogy a csomagküldési idő determinisztikus. A 4.1.5. szakaszban utaltam arra, hogy a kapcsolók várakozási sorokat tartalmaznak az egyes túlterhelt csomópontok tehermentesítésének érdekében. A várakozási sorok hossza, és ezek miatt a csomagküldés időtartama a hálózat aktuális terheltségétől függ. A csomagküldési idő azért lesz mégis determinisztikus, mert a jelfeldolgozó rendszerekben nincs sporadikus adatforgalom, az egyes csomópontok a működés során konstans adatforgalmat generálnak. Ilyen módon a kapcsolókban nem alakulnak ki várakozási sorok, vagy azok konstans hosszúságúak.

Ez a gondolatmenet természetesen csak akkor állja meg a helyét, ha a hálózaton valóban nincsen sporadikus forgalom. Ez teljes mértékben az Ethernet rétegeire épülő magasabb szintű protokollok működésétől függ. A későbbiekben látni fogjuk, hogy például TCP szállítási protokoll esetén a sporadikus forgalom nem kerülhető el, amely a várakozási sorokon keresztül csomagküldés késleltetési idejére hat, azaz a csomagküldés jittere megnőhet.

Csomagok elvesztésével nem kell számolni, mert a 16 ütközés elszenvedése miatt sikertelen csomagok elvesztését (*excessive collision*) az ütközésmentes működés kizárja, így csak a fizikai hibákból adódó csomagvesztésekkel kell számolni. A mai jó minőségű alkatrészek használatával ez gyakorlatilag nem fordul elő.

Az ütközésmentes hálózat legnagyobb hátránya a magas költsége. Szükség van egy Ethernet kapcsolóra, általában véve is magasabb technológiai színvonalú alkatrészeket kell használni egy ilyen hálózatban. Ez igaz a kábelezésre és a csomópontokra is, hiszen ez utóbbiaknak támogatniuk kell a full-duplex kommunikációt. Ha ezt nem támogatják, akkor a csomópont és a kapcsoló között továbbra is előfordulhatnak ütközések.

4.2.2. Ütközéses kommunikáció

A szenzorhálózatok körében egyelőre még a hagyományos közös kábelszegmenses, 10 Mbps sebességű, fél-duplex kommunikáció jellemző, bár a jövőben ezek visszaszorulására lehet számítani. Ez ilyen hálózatok legfontosabb tulajdonsága az, hogy a közös csatornán ütközések vannak.

Az ütközések közvetlen hatása az, hogy a periodikusan elküldött csomagok jitterrel fognak megérkezni a címzetthez. A fentebb említett *excessive collision* szintén előfordulhat, tehát terhelés esetén egyes csomagok már az adó csomópontnál elvesznek. Elveszett csomagnak minősülnek azok a csomagok is, amelyek az esetlegesen nagy csatornakésleltetés miatt túl későn érkeznek meg a vevő csomópontba. A csomagvesztés harmadik lehetséges

helye maga a csatorna lenne, de hasonlóan az ütközésmentes esethez, a fizikai hibákból adódó csomagvesztés itt is elhanyagolható.

Az ütközések gyakorisága egyenes arányban áll a csomagok hosszával. Elosztott jelfeldolgozó rendszerekben az egyes csomópontok tipikusan kevés (néhány bájt) hasznos adatot küldenek a mintavételi frekvencia ütemében. Ezek az adatok az alkalmazott protokolloknak megfelelően speciális kódolással kerülnek az Ethernet csomagokba, így valójában ennél nagyobb adatmennyiséget kell átvinni, de még így is a legtöbb esetben belefér egy Ethernet keretbe a küldenő adat.

Felmerülhet az a megoldás, hogy csak minden N-edik mintavétel után legyen csomagküldés, de ekkor a csomagok N minta értékeit tartalmazzák. Ezzel a megoldással a csomópontok ritkábban küldenek nagyobb csomagokat, így az ütközések gyakorisága nem változik. Ugyanakkor mind az adó oldalon, mind a vevő oldalon nagyobb bufferekre van szükség, és egy csomag esetleges kiesése is nagyobb gondot jelent (hiszen több minta veszett el egyszerre). Mindezek alapján kedvezőbbnek tűnik, ha minden mintát egy darab minimális hosszúságú (64 bájt) Ethernet keret foglal magába, amely kereteket a csomópontok a mintavételi frekvenciának megfelelő ütemben küldik el.

4.2.3. Hard real-time jelfeldolgozás ütközéses hálózatokban

Amennyiben az elosztott jelfeldolgozó rendszerek csomópontjai ütközéses hálózatban vannak összekapcsolva, a csomagvesztés nélküli kommunikáció nem garantálható.

Általános esetben minden csomópont egy darab, minimális hosszúságú Ethernet csomagot küld a mintavételi frekvencia ütemében. A jelfeldolgozást végző csomópontok helyes működésének az a feltétele, hogy minden csomópont a mintavételi frekvencia ütemében megkapjon annyi csomagot, ahány bemenet feldolgozását végzi. Ez a feltétel még akkor sem teljesíthető biztosan, ha

$$T_s > N * T_{slot} \tag{4.1}$$

ahol N az adni kívánó csomópontok száma. Ugyanis, ha 16 csomópontnál több kíván adni, akkor a legrosszabb esetben előfordulhat, hogy az egyik csomag 16 ütközést szenved el, ami az adott csomag elvesztését jelenti. Ennél reálisabb gond, hogy a 4.1.3 szakaszban ismertetett bináris exponenciális visszatartás algoritmus miatt elképzelhető, hogy egy néhány ütközést elszenvedő csomag túl nagy várakozási értéket sorsol ki magának, és csak nagyon későn érkezik meg a vevőhöz.

Alacsony kihasználtságú csatorna esetén a csomagvesztések száma csökken, de teljesen nem küszöbölhető ki. A terheléssel arányosan a csomagok késleltetésének jittere is nő, melynek hatását bufferelléssel szokás csökkenteni. A bufferelés a jittert fix késleltetéssé alakítja, amely viszont gondot jelent szabályozási rendszerekben, mivel ezek visszacsatolást tartalmaznak. Az Ethernet önmagában nem alkalmas ezen hibák kiküszöbölésére, így mindig a magasabb szinteken megvalósított protokolloknak kell az alkalmazástól függően valamilyen hibatűrő algoritmust megvalósítaniuk.

4.2.4. Szabványos protokollok

Egy Ethernetre épülő hálózatban a követelmények betarhatók, ha a felsőbb kommunikációs rétegeket a tervező speciálisan az adott feladathoz tervezi meg. Mivel ez elég komplex és időigényes feladat, ezért a gyakorlatban inkább jól kipróbált, széles körben elterjedt szabványos protokollokat használnak.

Az Ethernetre épülő kommunikációs rétegeket a számítógép hálózatok többségében a TCP/IP (Transmission Control Protocol/Internet Protocol) család valósítja meg. Ennek hálózati rétege az IP, amely az Ethernet keretekbe ágyazott IP csomagokkal dolgozik. Ez tartalmazza a csomópontok ún. IP címét, amely nem ugyanaz, mint a csomópontok Ethernet címe. Ennek az a fontos következménye van, hogy minden csomagküldés előtt meg kell állapítani a címzett IP címéhez tartozó Ethernet címet. Ezt az ARP (Address Resolution Protocol) végzi, oly módon, hogy minden csomagküldés előtt az adni kívánó csomópont egy *broadcast* üzenetet küld (ARP kérés), amelyre az egyik olyan csomópont küld választ, amelyik ismeri az adott Ethernet címet. Ilyen módon az ARP kérés a csomagküldések előtt egy többlet késleltetést jelent. A helyzetet bonyolítja, hogy a csomópontok a legutóbbi néhány Ethernet címet egy cache-ben tárolják, tehát a késleltetés nem mindig jelentkezik. Jelfeldolgozó rendszerekben az ilyen jellegű sporadikus kommunikációt lehetőleg el kell kerülni, így statikus (lásd. 2.1.2.) rendszerek esetén nem cache-ben, hanem egy fix táblázatban kell tárolni az egyes csomópontok IP címeihez tartozó Ethernet címeket.

A TCP/IP család szállítási rétegében TCP és UDP (User Datagram Protocol) került definiálásra.

TCP

A TCP alapú kommunikáció alapvető tulajdonsága, hogy összeköttetés alapú kommunikációt valósít meg, amely nagyobb adatmennyiség átküldésére van optimalizálva. A kommunikáció úgy történik, hogy a két csomópont először felépíti a kapcsolatot, az adatáramlás során nyugtákkal kell igazolni a hiba-

mentességet, valamint a kommunikáció végén a kapcsolatot le kell bontani. Ezen kívül forgalomszabályozási és torlódás vezérlésről is gondoskodia kell az adónak. A hibamentes működést pozitív nyugtázás biztosítja, azaz a vevő a beérkezett csomagokról nyugtát küld az adónak, amely a nyugta meg nem érkezése esetén újraküldi a csomagokat.

Az alkalmazott forgalomszabályozás lényege az, hogy a vevő a nyugtákban mindig közli az adóval hogy még mennyi szabad buffer kapacitással rendelkezik. Az adó ennek megfelelően mindig csak ennyi további csomagot küld, tehát a vevő oldalon mindig van szabad buffer kapacitás. Ez a megvalósítás azt igényli, hogy a csomópontok rendelkezzenek elegendően nagy bufferekkel mind a vevő, mind az adó oldalon, hiszen az adónak is tárolnia kell azon csomagokat, amelyekről még nem érkezett minta. A TCP réteg komplexitása igen nagy, a megvalósítása jelentős erőforrásokat köt le a csomópontokon.

UDP

Az UDP alapú kommunikáció ezzel szemben összeköttetésmentes, azaz nem kell a kapcsolatokat felépíteni és lebontani, a csomagok nincsenek nyugtázva. UDP csomagok esetén lehetőség van *broadcast* üzenet küldésére. Általában UDP-t használnak mozgókép és hang továbbításra az Interneten.

Elosztott jelfeldolgozó rendszerek követelményeihez nyilvánvalóan az UDP áll közelebb. TCP esetén szinte lehetetlen a csomagok beérkezésének idejét megjósolni, illetve a szükséges erőforrásigény is nehezen biztosítható jelfeldolgozó processzorok esetén. UDP használatakor a működés kiszámíthatóbb, és alkalmas a mintánkénti csomagküldés megvalósítására.

4.3. MAGIC

A Gibson Guitar Corporation 1999-ben kezdett foglalkozni egy olyan protokoll kidolgozásával, amely elsődlegesen hangszerek, illetve stúdiótechnikai berendezések összeköttetését és együttműködését teszi lehetővé Ethernet hálózat felett. A protokoll a Media-accelerated Global Information Carrier (MAGIC) nevet kapta [MAGIC, 2003].

A MAGIC egy olyan Ethernet alapú protokoll, amelyet kifejezetten elosztott elosztott jelfeldolgozó rendszerek számára fejlesztettek ki, ezért érdemes részletesen megvizsgálni.

A MAGIC célja olyan stúdióhálózatok megvalósítása, ahol az összes hangszer és stúdiótechnikai eszköz egy közös Ethernet hálózaton működik együtt. A jeláramlás az egyes egységek között tisztán digitális, tehát a hangszerek analóg jeleit már a hangszeren belül, vagy annak közvetlen közelében AD

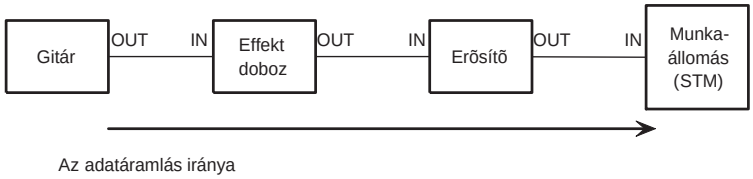
átalakító digitalizálja. Például egy elektromos gitártól az erősítőig a jel digitálisan, Ethernet csomagokba ágyazva jut el.

E megoldásnak számos előnye van, hiszen például jobb hangminőség érhető el, ha a fizikai jel csak egy AD és egy DA átalakításon megy keresztül a teljes feldolgozás alatt. Másik előny az Ethernet szabvány széleskörű elterjedtségéből adódik, ennek használatával minden különböző gyártó el tudja látni a hangszereit és berendezéseit MAGIC kommunikációra alkalmas hálózati kártyával. Ilyen módon lehetővé válik a jelenleg rengeteg különböző szabvány szerint kommunikáló eszközök univerzális összekapcsolhatósága. Ezt az Ethernet használata főleg azzal segíti elő, hogy széles körben állnak rendelkezésre Ethernet illesztő eszközök, ezek használata nagyon elterjedt. Az Ethernet továbbá egy kiforrott, széles körben jól ismert technológia.

4.3.1. Topológia

A MAGIC hálózatban minden csomópontnak vannak ki- és bemenetei. Tipikusan a hangszereknek csak egy kimenetük van, az erősítőknek egy be- és egy kimenetük, míg például egy keverőpultnak sok bemenete és egy-két kimenete van.

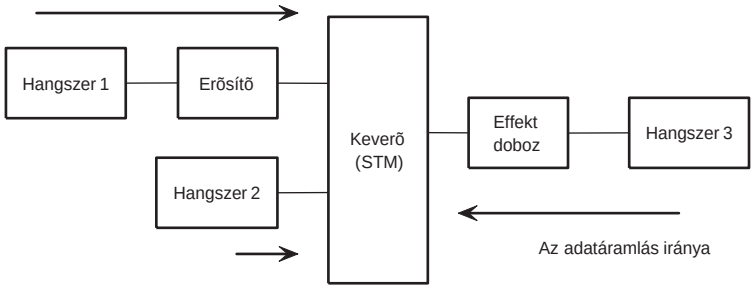
MAGIC hálózatok úgy alakíthatóak ki, hogy egy csomópont bemenetei más csomópontok kimeneteivel kapcsolhatók össze. Tehát a hálózatok lánc, vagy csillag topológiájúak. A hálózatban hurok nem alakulhat ki. Két lehetséges MAGIC hálózatot szemléltet a 4.2 és a 4.3 ábra.



4.2. ábra. Lánc topológiájú MAGIC hálózat

A fentiekből kiderül, hogy a MAGIC hálózatban nem egy közös kábelszegmensen van az összes egység, hanem pont-pont kapcsolat van a csomópontok között. A csomópontokat az Ethernet 100BaseT szabvány szerinti Category 5 csavart érpár köti össze, amelyen full-duplex kommunikáció történik.

A működés alapvetően a stúdiók igényeit tükrözi, hiszen tipikusan a hálózat végpontjaitól (hangszerek) érkező adatfolyam áramlik egy központi egység felé (keverő), amely adatfolyam közben módosulhat a közbenső csomópontok által (effekt processzor).



4.3. ábra. Csillag topológiájú MAGIC hálózat

4.3.2. Rétegszerkezet

A protokoll a szabványos Ethernet hálózatra épül rá, ezért csak az adatkapcsolati réteg feletti hálózati réteget definiálja. Az erre épülő alkalmazási rétegben egyelőre nem állnak rendelkezésre kiforott szoftver eszközök.

A hálózati rétegben az Ethernet csomagokba ágyazott MAGIC keretformátum van deifinálva. A MAGIC keret fix hosszúságú, az Ethernet kerettel együtt 222 bájt. Ez a keret két részből áll, adat és vezérlő részből. Az adat rész harminckét darab 32 bites hangmintát tartalmazhat maximálisan. A vezérlő rész különböző előre definiált vezérlési paramétereket, parancsokat, állapotjellemzőket tartalmazhat. A kereten belül az adat és a vezérlési rész is fix hosszúságú.

A MAGIC szabvány kompatibilis a stúdiótechnikában elterjedten használt MIDI szabvánnyal. A MIDI alapvetően hangszerek vezérlésére szolgál, a parancsok csak vezérlési információkat tartalmaznak, hangjelek digitális mintáit nem. A kompatibilitás azáltal biztosított, hogy a szabványos MAGIC csomagok vezérlési mezőjében lehetőség van MIDI parancsok küldésére is. Ilyen módon a MIDI és a MAGIC nem zárják ki egymást, a két szabvány együtt is használható.

4.3.3. Időzítések

A MAGIC hálózatot alapvetően adatfolyam (media-stream) továbbítására tervezték. Ez a gyakorlatban fix mintavételi frekvenciával mintavételezett minták továbbítását jelenti, azaz a hálózat egy kimenetétől a hozzá csatlakoztatott bemenetéig a mintavételi frekvencia ütemében kell egy MAGIC csomagot eljuttatni.

A csomópontok szinkronizált működését egy kitüntetett csomópont, a System Timing Master (STM) vezérli. Az, hogy melyik csomópont az STM, a hálózat topológiájából egyértelműen következik a következő szabály szerint:

1. Egy kimenettel rendelkező csomópont nem lehet STM.

2. Ha van csak bemenetekkel rendelkező csomópont, akkor mindig az az STM.

3. Ha minden csomópont rendelkezik ki- és bemenetekkel is, akkor az az STM, amelynek a kimenetei nincsenek más csomópont bemenetéhez csatlakoztatva

Az STM feladata az, hogy biztosítsa a hálózat számára az órajel frekvenciát, azaz az STM minden *bemenete* felé elküld egy MAGIC keretet a mintavételi frekvencia ütemében. E csomagok tehát az adatfolyam irányával szemben mennek, és az adatmezőben nem tartalmaznak tényleges mintákat.

A MAGIC hálózat többi csomópontja az STM felől érkező adatokkal szinkronban kell küldje a kimenetein az adatcsomagokat az STM felé (full-duplex működés). Továbbá minden csomópontnak továbbítania kell a *bemenetei* felé a mintavételi frekvencia ütemében az STM-től kapott csomagokat.

Ilyen módon a teljes hálózat az STM által diktált ütemben fog kommunikálni. A csomópontok között más csomagok nem mennek, hiszen a vezérlő információkat is a periodikusan küldött adatcsomagokkal együtt kell küldeni. Ez a működés megfelel a 3.2.2-ben megfogalmazott követelménynek, a hálózaton nincs sporadikus adatforgalom.

A helyes működés feltétele továbbá az is, hogy a csomópontok minimális késleltetéssel és kis jitterrel küldjék a csomagokat mindkét irányba. E feltételekre a MAGIC szigorú követelményeket ír elő.

5. fejezet

A megvalósított jelfeldolgozó rendszer

Az elosztott jelfeldolgozó rendszerek körében gyakorlati vizsgálatokat, méréseket is szerettem volna végezni, ezért építettem egy egyszerűbb mérőrendszert. E rendszer egyszerűbb elosztott jelfeldolgozási alkalmazások implementálására használható, a működés jellegzetességeinek, hibáinak megfigyelésére van lehetőség. A rendszer arra is alkalmas, hogy különböző hálózati protokollok (elsősorban az Ethernet) tipikus hibáit mesterségesen megvalósítsa, szimulálja.

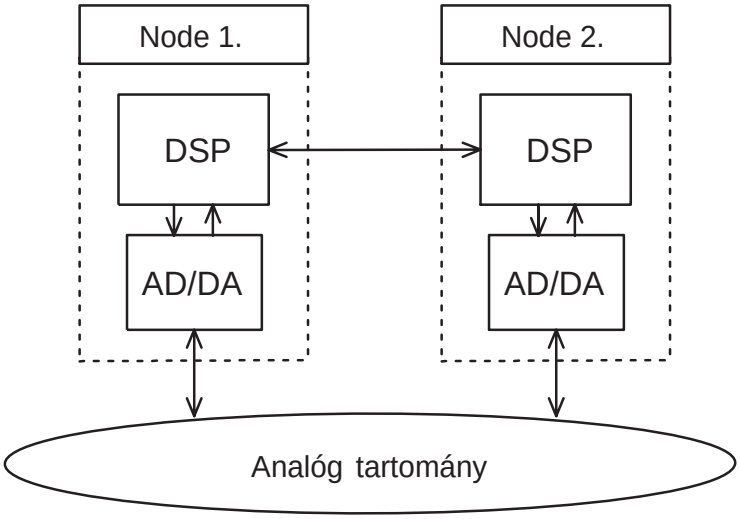
5.1. A rendszer felépítése

A megvalósított rendszer blokkvázlata az 5.1. ábrán látható. Ez a felépítés a 3.2.1. fejezetben bemutatott architektúra legegyszerűbb esete, a rendszer két egyenrangú csomópontból áll, amelyek kommunikálni képesek egymással. A csomópontokat DSP-k vezérlik, amelyek AD/DA átalakítókon keresztül állnak kapcsolatban a környezetükkel.

A rendszer megépítéséhez az Analog Devices Inc. két ADSP-21061 EZ-KIT Lite fejlesztői kártyáját használtam fel, amelyek a rendszer egy-egy csomópontját valósítják meg.

5.2. Az ADSP-21061 EZ-KIT Lite fejlesztői kártya

Az Analog Devices Inc. a legtöbb termékéhez kínál ilyen vagy ehhez hasonló fejlesztői kártyát. Az ADSP-21061 EZ-KIT Lite az ADSP-21061 jelfeldol-



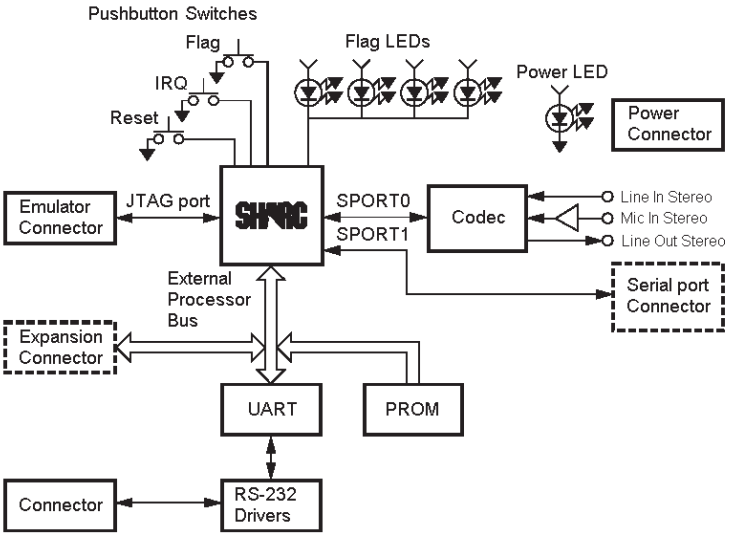
5.1. ábra. A megépített rendszer blokkvázlata

gozó processzor köré van építve. Ezek a kártyák mentesítik a vásárlót a tesztáramkör építése alól, ami például egy lebegőpontos DSP esetében négyrétegű nyomtatott áramkör tervezését jelenti.

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

5.2.1. A kártya felépítése

A kártya felépítése az 5.2 ábrán látható. Az ábra forrása: [EZ-KIT, 1997]. Az ADSP-21061 EZ-KIT Lite fejlesztői kártya elsődlegesen arra szolgál, hogy az ADSP-21061 processzor minél több funkciója elérhetővé váljon, egyszerűen hozzáférhető legyen. A kártyán négy LED és három nyomógomb található. A LED-eket a DSP flag lábai hajtják meg. A három nyomógomb közül egy a Reset lábra, egy a FLAG1 lábra, egy pedig az IRQ1 lábra van illesztve.



5.2. ábra. Az ADSP-21061 EZ-KIT Lite blokkvázlata

Ez utóbbi megnyomásakor megszakításkérés történik, amely lehetőség jól kihasználható a különböző alkalmazások implementálásakor.

A kártyán található portok egyike a JTAG emulátor port, amelyen keresztül hibakeresésre nyílik lehetőség. A JTAG port használatáról bővebben az 5.2.2 szakaszban lesz szó. A fejlesztői kártyán található RS-232 szabványos porton keresztül aszinkron soros kommunikációra van lehetőség. A JTAG porton kívül ez a másik lehetőség személyi számítógéphez való csatlakozásra. A processzor nem rendelkezik közvetlenül RS-232 szabvány szerinti lábakkal, ezért a port megvalósítása egy programozható logikai áramkörrel (PAL) történik. A processzor ezen PAL áramkörön keresztül vezérli a kommunikációt, amely a DSP külső memóriabuszára (external bus) van illesztve. A DSP külső memóriabuszára a logikai áramkörön kívül egy gyárilag beégetett EPROM-ot is illesztettek, a processzor erről boot-ol hardver reset után.

Egy AD1847-es sztereó codec-et is elhelyeztek a kártyán, amelyhez analóg oldalról két sztereó jack dugóval lehet csatlakozni (egy kimenet, egy bemenet). Az AD1847-es codec a DSP nulladik szinkron soros portjához van illesztve (SPORT0). A megvalósított rendszerben nincsenek valódi szenzorok vagy beavatkozók, de a két jack dugó a rendszer szempontjából két analóg bemenetet és két analóg kimenetet jelent, azaz egy kártyával egy két szenzorral és két beavatkozóval rendelkező csomópontot lehet szimulálni.

5.2.2. ADSP-21061

A fejlesztői kártya legfontosabb egysége az ADSP-21061 jelfeldolgozó processzor [SHARC, 1997]. Ez a SHARC DSP-család egyik tagja, amely az ADI egyik leginkább elterjedt és széles körben használt termékcsaládja. A SHARC betűszó a Super Harvard ARhitecture Computer rövidítése. Ez a felépítés a DSP-k körében klasszikusnak számító Harvard-architektúra továbbfejlesztett változata. A Hardvard-architektúra a személyi számítógépek processzorainak Neumann-architektúrájával szemben megkülönböztet kód-, illetve adatmemóriát. A két memóriaterület külön buszon érhető el, így lehetőség van párhuzamos használatukra. Ez a lehetőség műveletek ciklikus elvégzésekor nagymértékben gyorsítja a program futását.

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

Az ADSP-21061 a SHARC család első tagjának, az ADSP-21000-nek egy továbbfejlesztett változata. A processzor mag (processor core) megegyezik a két típusnál, de a 21061-est néhány fontos kiegészítővel látták el. Ezek a következők: SRAM, IO processzor, IO busz. Ezen egységek mindegyike tokon belül van megvalósítva. A 21061-es nem a legfejlettebb modell a ma kapható DSP-k között, de sebessége és számítási pontossága kielégítő a legtöbb átlagos jelfeldolgozási alkalmazás szempontjából. Ipari feladatokra is széles körben használják ezt a DSP-t, mert viszonylag alacsony ára mellett az összes olyan követelményt teljesíti, amely egy lebegőpontos jelfeldolgozó processzortól elvárható.

Teljesítményadatok

A CMOS megvalósítású ADSP-21061 processzor maximális órajelfrekvenciája 50 MHz. Mivel a DSP órajelütemenként egy utasítást hajt végre, így az utasításvégrehajtási sebesség 50 MIPS (Million Instructions Per Second). A processzor teljesítményét jellemző benchmark adatokat az 5.1 táblázat ismerteti.

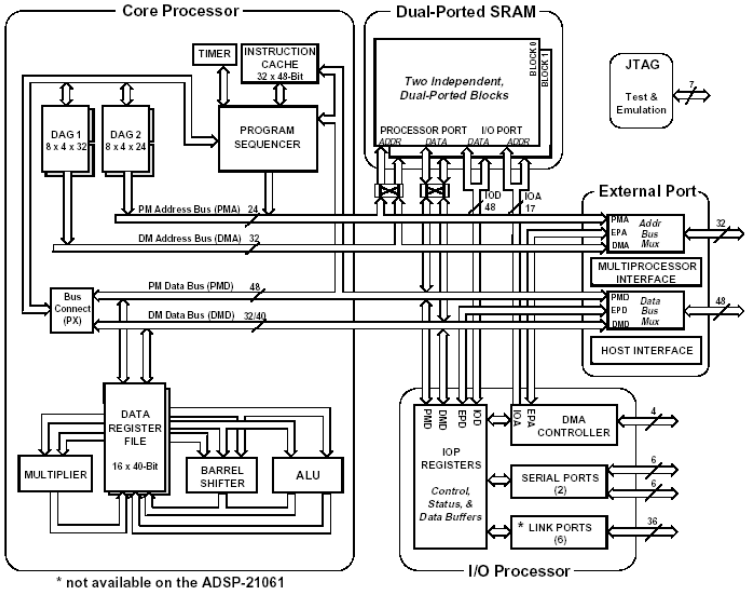
Felépítés

Az ADSP-21061 felépítése az 5.3 ábrán látható. (Forrás: [SHARC, 1997]) A processzor fő részei a következők:

- **Műveletvégző egység** – A műveletvégző egység kétféle lebegőpontos számábrázolási formátum használatát támogatja, a 32 bites IEEE

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

5.1. táblázat. Az ADSP-21061 benchmark adatai ($f_{\mathrm{CLK}} = 50\mathrm{MHz}$)



5.3. ábra. Az ADSP-21061 felépítése

single-precision floating-point szabványos formátumot, illetve a 40 bites kiterjesztett (extended precision) formátumot. Lehetőség van 32 bites fixpontos formátum használatára is. A műveletvégző egység részei: ALU (Arithmetic and Logic Unit), szorzó egység, shifter egység. Ezek párhuzamosan vannak megvalósítva, tehát lehetőség van egyidejű használatukra a műveletvégzés gyorsítása érdekében. Az egység számítási kapacitása általános esetben a fentebb említett 50 MIPS. Ez azonban egyes tipikus jelfeldolgozási számítások során a műveletek párhuzamos végrehajtása miatt megnövekszik. Az ilyen módon elérhető maximális számítási kapacitás 120 MIPS vagy 120 MFLOPS (Million Floating-point Operation Per Second). Ez utóbbi mérőszám is gyakran használatos, a processzor lebegőpontos számítási műveletekre vonatkoztatott számítási sebességet jellemzi. Mivel az ADSP-21061-es alapértelmezésben lebegőpontos számokon hajtja végre az utasításokat, esetében a két mérőszám megegyezik.

- **Adatregiszter blokk** – 32 általános célú regiszter áll rendelkezésre a különböző adatmozgatási és számítási feladatokhoz. Ez a 32 regiszter két 16-os csoportra oszlik, az elsődleges (primary) és a másodlagos (alternate) regiszterblokkokra. A két blokk közül mindig csak az egyik aktív, a másik nem hozzáférhető.

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

- **Adatcímző egységek** – Az adatcímző egységek egyike a kódterülethez, a másik az adatterülethez tartozik. Ezen egységek hardver szinten támogatják cirkuláris bufferek megvalósítását. Egy egyszerű memóriatömb azáltal lesz cirkuláris buffer, hogy a bufferen belüli címeket az adatcímző egység speciális módon számítja ki. Ez úgy történik, hogy a cirkuláris bufferhez egy mutató regisztert rendel hozzá, amely segítségével a címzés történik. A mutató regiszter a cirkuláris buffer báziscíméhez képesti eltolást tárolja, értéke maximálisan a buffer hossza lehet. Amennyiben a mutató regiszter olyan értékre módosulna, amely

nagyobb, mint a buffer hossza, akkor a regiszter tartalma ezen értéknek a buffer hosszával osztott törtrésze lesz. Ilyen módon egy körkörös vagy cirkuláris buffer jön létre, amely nagyon kedvező néhány tipikus jelfeldolgozási algoritmus (FFT, FIR szűrés) megvalósításakor.

- **SRAM memória** – A tokon belül (on-chip) megvalósított 1 Mbites SRAM memória miatt nem kell külön külső memóriát biztosítani a processzor működéséhez. Az SRAM memóriaterület a Harvard-architektúrának megfelelően két részre van osztva, adat-, illetve kódterületre. Ez a hagyományos elrendezés, de nincs megkötés abból a szempontból, hogy milyen típusú adatokat tárolunk az egyes területeken. Az optimális működési sebesség elérésének érdekében mindig az adott alkalmazás szempontjából mérlegelni kell, hogy milyen elrendezéssel érhető el a lehető legtöbb párhuzamos memóriaművelet. Például FIR szűrés esetén a bemeneti adatokat és a szűrőegyütthatókat külön memóriaterületen érdemes tárolni.

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

- **Host interfész** – A host interfész nyújt lehetőséget szabványos mikroprocesszoros buszokhoz való illesztésére. Ezen az interfészen keresztül a DSP egyszerűen csatlakoztatható személyi számítógéphez.

- **DMA vezérlő** – A DMA (Direct Memory Access) vezérlő nagy tömegű blokkos adatmozgatások esetén használatos. Ekkor a processzor magjának igénybevétele nélkül a DMA vezérlő nagy sebességű adatmozgatást végez a belső SRAM memória és valamely más eszköz között. Ezek a következők lehetnek: külső memória, külső periféria, host processzor, soros port. DMA kapcsolat létesíthető a külső memória és más külső eszközök között is.

- **Soros portok** – A DSP két szinkron soros porttal rendelkezik, amelynek segítségével különböző digitális perifériákhoz való csatlakozásra nyílik lehetőség. Ezen soros port adatátviteli sebessége széles tartományban mozoghat, órajelének frekvenciája maximálisan a DSP órajel frekvenciájával megegyező lehet. Az EZ-KIT kártyán a nullás sorszámú portra az AD1847-es codec van illesztve, az egyes sorszámú port

(SPORT1) nincs használva, de a lábai ki vannak vezetve a kártyára egy külső tüskesorra. A SPORT1-et használtam fel a két kártya összekötésére, a két DSP ezen keresztül kommunikál.

- **JTAG interfész** – Az IEEE JTAG 1149.1 szabványnak megfelelő port a DSP tesztelésére szolgál. Ezen port segítségével emulálásra van lehetőség, ami azt jelenti, hogy az integrált áramkörön belül vannak megvalósítva az emulátor funkciók. Tehát megfelelő eszköz vagy alkalmazás segítségével e porton keresztül a DSP működése nyomon követhető és tesztelhető. Ezt a portot használja az 5.2.3 szakaszban ismertetett EZ-ICE Emulátor is.

5.2.3. A VisualDSP fejlesztői környezet

A DSP programok fejlesztése az EZ-KIT kártyával nagymértékben együttműködni képes VisualDSP fejlesztői környezet segítségével történt. A fejlesztői környezet két független alkalmazásból áll, a Projekt szerkesztőből és a Debugger-ből.

Projekt szerkesztő

A Projekt szerkesztőben a DSP program forrásfájljainak szerkesztése történik. Az egymáshoz rendelt forráskód fájlok alkotnak egy projektet, amelyet a Projekt szerkesztő fordít le, és egy DSP-re letölthető programkód fájlt generál. A kimeneti fájl típusa beállítható a Projekt tulajdonságok ablakban. A lehetőségek: debuggolható verzió, kész verzió, vagy EPROM-ba égethető bit-fájl. A forráskód fájlok készülhetnek gépi nyelven (assembly), vagy C nyelven. A Linker számára az ún. Linker Description File tartalmaz utasításokat. A Projekt szerkesztőn belül a következő funkciók vannak megvalósítva:

- **C fordító** – E funkció biztosítja a DSP-k C nyelvű programozhatóságát, amely sok esetben kényelmesebb a sokszor nehézkesnek tűnő gépi kóddal szemben. A DSP-k vezérlő programjait a diplomamunka során ennek ellenére gépi nyelven írtam, mert a DSP speciális jelfeldolgozási feladatokhoz optimalizált hardver eszközeit így jobban ki lehet használni. A gépi nyelvű választást indokolja az is, hogy a DSP programok tipikusan egy rövid ciklusmagot futtatnak a futási idő túlnyomó részében, így a futás ideje gyakorlatilag attól függ, hogy ez a mag pontosan milyen hosszú. Egészséges kompromisszumot jelent, ha ezeket a sokszor futó programrészeket gépi nyelven, a kód többi részét pedig C-ben írja meg a tervező. [Snyder, 1999]

- **Compiler** – A compiler végzi a gépi kódban megírt forrásfájlok valamint a C fordító által lefordított fájlok szintaktikai és szemantikai ellenőrzését.

- **Linker** – A linker felelős azért, hogy az egy projekten belüli különálló fájlokat egybefűzze, és előállítsa a kimeneti fájlt. Minden projekthez tartozik egy Linker Description File (LDF) is, amely szintén a projekt részét képezi. A linker ez alapján helyezi el a forrásfájlok kód és adatrészleteit a DSP különböző memóriaterületeire.

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

Debugger

A fejlesztői környezet másik nagy részegysége a Debugger alkalmazás. Ez az alkalmazás lehetőség biztosít a megírt DSP programok kipróbálására futás közben. Lehetőség van a program futásának részletes vizsgálatára, töréspontok helyezhetők el, lehet léptetve végrehajtani az utasításokat, a memóriaterületek valamint a rendszer regisztereinek tartalma folyamatosan monitorozható. A Debugger "bementi adata" a Projekt szerkesztő által előállított, a DSP-re letölthető fájl. A DSP program futtatásának helye beállítható, ez lehet emulátor vagy szimulátor.

- **Szimulátor** – Ennek használatakor a program fizikailag nem töltődik le a DSP-re, hanem az alkalmazás a személyi számítógépen szimulálja a DSP működését. Ez a szimuláció minden részletre kiterjed, tehát a DSP majdani várható működése vizsgálható. Ilyen módon azonban nem lehet kipróbálni sem a fizikai DSP alapú rendszer többi egységét, illetve az egységek összehangolt működését sem. Ezért, ha lehetőség van rá, akkor az emulátort érdemes használni.

- **Emulátor** – Az emulátor használatához szükség van az ADI által gyártott EZ-ICE emulátor kártyára, amelyet a személyi számítógépre telepíteni kell. Ehhez tartozik egy csatlakozó, amelyet az ADI fejlesztői kártyáin található JTAG Emulátor csatlakozóval lehet összekötni. Erre a processzor belső (on-chip) emulálását lehetővé tevő szabványos JTAG portjának lábai vannak kivezetve. Az emulátor ezen az interfészen keresztül vezérli a teljes vizsgálatot. Ugyanitt történik a Projekt szerkesztő által előállított program letöltése a DSP memóriájába. Lehetőség van a program futtatására, az utasítások léptetett elvégzésére

vagy töréspontok elhelyezésére. A DSP összes regiszterének tartalma lekérdezhető és igény szerint módosítható. A memóriaterületek teljes területének tartalma fájlba írható (dump) vagy feltölthető (fill). A fenti műveletek elvégzésének mindegyikét a JTAG szabvány szerint a DSP hardveresen támogatja, ezért elvégzésükhöz nincs szükség külön monitorprogram letöltésére. A Debugger alkalmazás segítségével a program futásának teljes körű vizsgálata lehetővé válik.

5.3. Soros kommunikáció

Az EZ-KIT fejlesztői kártyán a DSP azon lábai is ki vannak vezetve tüskesorokra, amelyekhez nincs alkatrész illesztve. A két kártya közötti összeköttetést az egyes sorszámú szinkron soros port (SPORT1) segítségével valósítottam meg. Az SPORT az alábbi lábakkal rendelkezik:

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

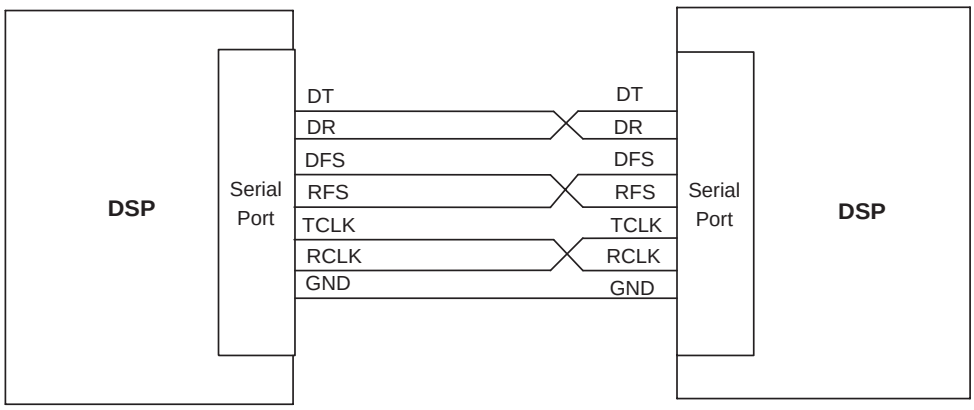
5.2. táblázat. Az SPORT lábai

A soros port full-duplex kommunikációt tesz lehetővé, az adás függetlenül történik a vételtől. A működés lényege az, hogy az adat lábon (DR vagy DT) a hozzátartozó CLK láb szerinti ütemben kell adni az adatszót. Az RFS és TFS lábak jelzik az adatszó kezdetét. Az ADSP-21061-es szinkron soros portja nagyon flexibilisen programozható, gyakorlatilag a működés minden paramétere beállítható.

A két kártya összeköttetése az 5.4 ábra szerint történt.

A DSP a SPORT1-hez két regiszteren, a TX (transmit) és a RX (receive) regisztereken keresztül fér hozzá. A küldeni kívánt adatokat a TX regiszterbe kell írni, a fogadott adatok az RX regiszterbe kerülnek. Mindkét regiszterhez tartozik egy megszakítási rutin is, a transmit akkor lép fel, amikor a port kiolvassa a TX regiszterből a küldendő adatot, a receive pedig akkor, amikor új beérkezett adat áll rendelkezésre az RX bufferben.

Lehetőség van arra is, hogy a soros port adatforgalma a háttérben, a DMA vezérlő által történjen, de ezt a megoldást a SPORT1 esetében nem



5.4. ábra. A két kártya összeköttése

használtam. Az SPORT0 esetén igen, tehát a codec felől jövő adatok automatikusan a memóriába kerülnek, és a memória dedikált területeiről kerülnek kiolvasásra a küldendő adatok. Itt minden mintavétel után két minta érkezik be, és két mintát is kell kiadni, azaz érdemes DMA-t használni, mert nem kell a főprogramban törődni a bufferek helyes feltöltésével, kiolvasásával.

Az SPORT1-et viszont olyan módon programoztam fel, hogy az Ethernethez hasonlóan bármikor lehessen csomagot küldeni, azaz a csomagok küldése nincs szinkronizálva semmilyen más időzítővel. (A SPORT0 esetén például az AD-tól érkező csomagok az AD1847-es mintavételi ütemében jönnek, és a DSP is ezekkel szinkronban küldi a mintákat a DA-nak.)

A csatornán tehát a kommunikáció úgy valósul meg, hogy amikor az egyik DSP új adatot ír a TX regiszterébe, akkor a SPORT1 a következő TCLK órajellel szinkronban elkezdi ennek bitjeit kiküldeni. A TX és az RX regiszterek 32 bitesek, ezért az adatcsomagok is ilyen méretűek.

A vételi oldalon egyszerűen az történik, hogy a bejövő adat, miután bekerült az RX regiszterbe, meghívja a Receive interruptot. Mivel a működés full-duplex, azaz a vétel független az adástól, ezért ütközés nem fordulhat elő, legfeljebb a DSP-n belül kerülnek konfliktusba az egyidejűleg érvényre jutó SPORT0 és SPORT1 Receive interruptok. A Transmit interruptok nincsenek engedélyezve, ezeket nem használtam.

A két DSP teljesen azonos módon van felprogramozva, az egyetlen aszimmetria az, hogy mind az RCLK, mind a TCLK órajelet az egyik DSP adja, mert ez a megoldás stabilabb működést eredményezett, mint amikor például mindkét DSP a saját TCLK órajelét adta.

Az RCLK és TCLK órajeleket a SPORT1 a fő órajelből osztja le. Az EZ-KIT kártyákon a DSP órajele 40 MHz, amelyet 8-cal osztottam le, így

f_{TCLK} = 5MHz, azaz a maximális sávszélesség 5 Mbps. (Ez akkor érhető el, ha a soros port órajele maximális értékű, azaz megegyezik a DSP órajelével, és a TX regiszterbe 32 utasításonként új adat kerül.)

5.4. Adaptív Fourier-analizátor megvalósítása

A megvalósított rendszert azzal a céllal építettem meg, hogy implementálni lehessen valódi jelfeldolgozó alkalmazásokat egy elosztott rendszerben. Az implementáció előnye az, hogy az elosztott megvalósításból adódó sajátosságok, nehézségek a gyakorlatban vizsgálhatók, a rendszeren külöböző mérések végezhetők. A rendszer arra is alkalmassá tehető, hogy szimulálja egy Ethernet hálózaton kommunikáló elosztott rendszer működési sajátosságait.

A megvalósított rendszerben az elsődleges feladat egy valós-idejű, mintavételezett jel mintáinak átvitele a két DSP között, a közös csatornán. A működés lényege az, hogy az egyik DSP a saját AD átalakítója által mintavételezett adatokat feldolgozás nélkül elküldi a másik DSP-nek. A beérkezett csomagokat a második DSP a saját DA átalakítóján kiadja. Mivel a két csomópont mintavételi frekvenciái megyegyeznek (azonos értékre vannak beállítva), ez a procedúra elvi szinten egy egyszerű *echo* megvalósítás, azaz a második DSP DA átalakítójának kimenetén az első DSP AD átalakítója által mintavételezett analóg jelnek kellene megjelennie.

A gyakorlatban ez nem valósul meg, a kiadott analóg jel fázisa az eredeti jelhez képest lassan, de folyamatosan változik, illetve bizonyos időközönként ugrás van benne. Ennek oka az, hogy a két csomópont mintavételi frekvenciája csak elvileg egyezik meg, valójában ezek minimális mértékben eltérnek egymástól. Az eltérés mértékét a kristályoszcillátorok pontossága határozza meg.

Elosztott jelfeldolgozó rendszerekben alapvetően fontos, hogy a csomópontok mintavételezése szinkronizálva történjen. Ezzel a problémával és a lehetséges megoldásokkal részletesen foglalkozom a következő (6.) fejezetben.

A mintavétel szinkronizációjának problémája meghatározta a rendszeren implementálásra kerülő alkalmazást is. Azért esett a választásom az adaptív Fourier-analizátor (AFA) megvalósítására, mert ez egy olyan gyakorlatban is használt jelfeldolgozó alkalmazás, amely valós periodikus jelek igen pontos frekvenciamérését teszi lehetővé.

5.4.1. Az adaptív Fourier-analizátor

Az adaptív Fourier-analizátor alapját a [Péceli, 1986] által ismertetett rezonátoros struktúra képezi. Ez a struktúra periodikus jelek modellezésére használható, mivel a struktúra a megfigyelt jel pontos analízisét adja a jel Fourier komponensei alapján.

A [Nagy, 1992] által ismertetett adaptív Fourier-analizátor egy olyan módosított rezonátoros struktúra, amely a rezonátorok frekvenciáját adaptívan változtatja, azaz az első (legalacsonyabb frekvenciájú) rezonátor frekvenciáját a megfigyelt periodikus jel alapharmonikusára hangolja. A többi rezonátor ennek megfelelően az első rezonátor frekvenciájának többszöröseire kerül, azaz az egyes rezonátorok frekvenciája megegyezik a periodikus jel felharmonikusainak frekvenciájával. Az alapharmonikust tekintve az AFA működése egy digitális megvalósítású PLL működéséhez hasonló.

Az AFA-t a gyakorlatban is jól lehet használni, például *order–analízis* megvalósítására. Az order–analízis során egy periodikus jel harmonikusaihoz tartozó Fourier komponenseket vizsgálják. Ez például forgó gépek vizsgálatakor hasznos, ahol az alapharmonikus a forgó gép fordulatszáma.

Az AFA-t használják továbbá aktív zajcsökkentési alkalmazásokban, periodikus zaj elnyomása esetén.

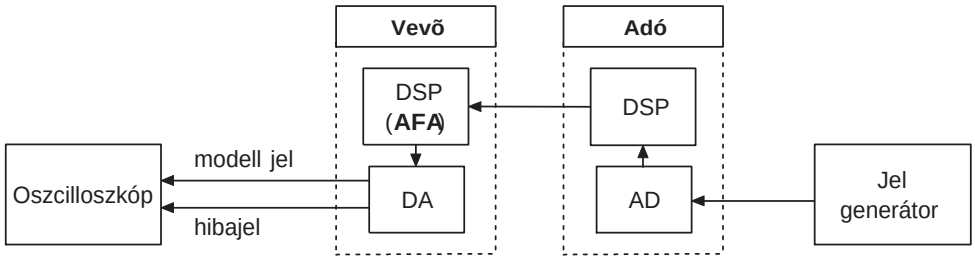
5.4.2. Megvalósítás

Az AFA implementálásához felhasználtam egy már működő zajcsökkentési alkalmazás programkódját [Kovács, 2002].

A megvalósított AFA esetén a rezonátorok száma állandó (N). Ha az alapharmonikus (f_a) frekvenciája annyira megnövekszik, hogy az N-edik felharmonikus frekvenciája (Nf_a) nagyobb lesz mint a mintavételi frekvencia fele, akkor az átlapolódás elkerülésének érdekében a rezonátort el kell távolítani a rendszerből. Ugyanígy, ha az alapharmonikus frekvenciája olyan mértékben lecsökken, hogy

$$(N+1)f_a < f_s/2 \tag{5.1}$$

akkor egy új rezonátort, az $N+1$-ediket be kellene léptetni a rendszerbe. A megvalósított AFA állandó számú rezonátorral rendelkezik, ezért a PLL analógiát követve egy befogási tartományt írtunk elő, amely tartományon kívülre a legalacsonyabb rezonátor frekvenciáját nem lehet állítani. Ez a tartomány az alacsonyabb frekvenciák irányába szélesebb mint amit az elvi korlát írna elő, mert a gyakorlatban nem jelent nagy problémát, ha nem léptetünk be rezonátorokat a mintavételi frekvencia felének környékén, csak



5.5. ábra. Az AFA-t implementáló rendszer blokkvázlata

a jelmodell nem lesz tökéletes. A megengedett felső korlát viszont nem lehet nagyobb mint az elvi korlát, mert a legnagyobb frekvenciájú rezonátor átlapolódása a rendszer instabilitását is okozhatja.

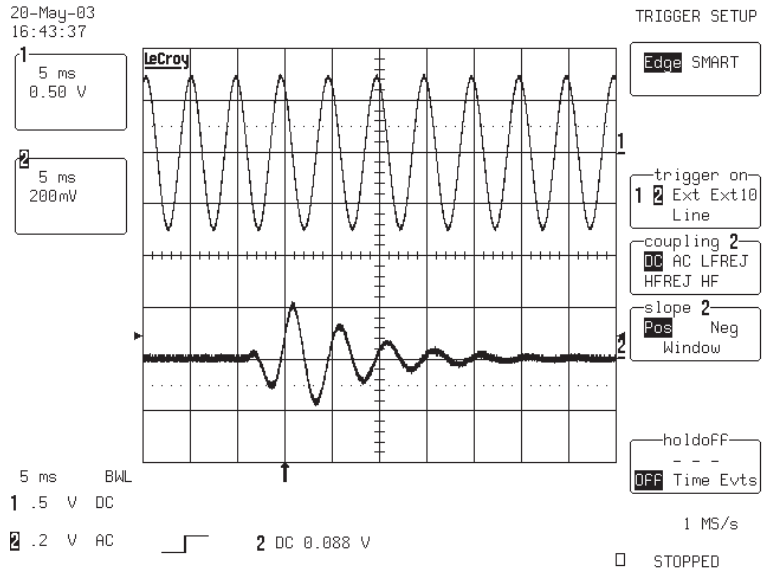
A megvalósított AFA 38 rezonátort használ, az alapfrekvenciája 200 Hz. (A mintavételi fekvencia 16 kHz.) A befogási tartomány: 100..300 Hz.

A vizsgálatok során az AFA bemenete (a megfigyelt jel) vagy a saját codec AD-je által mintavett jel, vagy a másik DSP-től érkező adatfolyam mintái voltak. Az AFA egyik kimenete a megfigyelt jel modellje, a másik a hibajel, azaz a modelljel és a megfigyelt jel különbsége. A hibajel a megvalósítás szempontjából hasznos információkat hordoz, mert az implementáció során ennek nagyságából derül ki, hogy a megfigyelt jelet milyen hibával sikerült modelleznie az AFA-nak. Az AFA akkor működik jól, ha a hibajel közelítőleg nulla.

A mérési elrendezést szemlélteti az 5.5 ábra.

Az AFA implementálása sikeres volt, a vártnak megfelelően működött. Az 5.6 ábra szemlélteti a hibajel megnövekedését és eltűnését a bemenő jel ugrászerű frekvenciaváltozásakor. Az ábrán a kimenetek változása látható a bemenőjel frekvenciájának ugrásszerű változásának hatására. (A bemenőjel 200 Hz-ről 210 Hz-re változott.) Az oszcilloszkóp csatornáin az AFA két kimenete látható, az 1. a modell jel, a 2. a hibajel.

Az AFA implementálásának elsődleges célja az volt, hogy pontos frekvenciaméréseket lehessen végezni valós jeleken. Erre a 6. fejezetben ismertetett probléma és a megoldások vizsgálatához van szükség. A mérési eredményeket a 6.2.5. alfejezetben foglaltam össze. Az AFA sikeres megvalósítása ezen kívül azt is demonstrálja, hogy a megépített elosztott jelfeldolgozó rendszer működőképes, gyakorlatban használt alkalmazások implementálására alkalmas.



5.6. ábra. Az AFA beállása. Felül a modelljel, alul a hibajel látható.

6. fejezet

A mintavétel szinkronizáció

Az 5. fejezetben bemutatott mérőrendszerben alapvető problémát jelent az, hogy a két csomópont mintavevő órája nincs szinkronizálva. Az elvileg azonos, de valójában kissé eltérő frekvenciájú mintavétel eredményeképpen a két csomópont működése nincs szinkronban, amely az egymás közötti kommunikációban problémát jelent.

Ugyanez a probléma nagyobb elosztott jelfeldolgozó rendszerekben is megjelenik. Egy elosztott jelfeldolgozó rendszerben, egy "okos" szenzorhálózatnak is tekinthető, tipikusan minden csomóponthoz más szenzorok tartoznak, azaz minden csomópont autonóm módon mintavételezi a környezet jeleit. Ha a mintavételezés nincs szinkronizálva, akkor a csomópontok bemeneti adatai nem konzisztensek, azaz ugyanazon sorszámú mintavételezéshez más fizikai időpont tartozik. A csomópontok közötti kooperáció elképzelhetetlen az időben szinkronizáltság és adatkonzisztencia nékül, amelynek jelfeldolgozó rendszerekben szükséges feltétele az egyes egységek mintavételezésének szinkronizációja.

6.1. A probléma vizsgálata

Elosztott jelfeldolgozó rendszerek alapvető tulajdonsága az online működés. A 2.2.1. fejezetben ismertetett okokból a jelfeldolgozó rendszer egyes csomópontjain futó jelfeldolgozó algoritmus a mintavételezéshez tartozó interrupt (IT) rutinban van megvalósítva. A csomópontokon futó feladatok közül ez a legfontosabb, ezért általában minden más task is ehhez szinkronizáltan fut. A jelfeldolgozást végző csomópontban lényegében egy olyan periodikus ütmezés valósul meg, amelyet a mintavevő óra időzít.

E tulajdonság miatt van különösen fontos szerepe a mintavételi frekvenciának. Az elosztott jelfeldolgozást végző csomópontok a futtatott jelfel-

dolgozási algoritmus bemeneti adatai beérkezésének ütemével szinkronban működnek, és a kimeneti adatokat is ebben az ütemben állítják elő. A probléma akkor jelentkezik, ha egy elosztott rendszerben egy csomópont olyan algoritmust futtat, amelynek bemenetei között vannak saját codec által mintavételezett, és egy másik csomópont által számított adatok is. A helyes működéshez szükséges, hogy ezek az adatok szinkronban álljanak rendelkezésre a feldolgozást végző csomópontnál. Emiatt kell szinkronizálni a két csomópont mintavételezését, hiszen az adó a saját mintavételének ütemében küldi a számított értékeket.

A fenti gondolatmenet is érzékelteti, hogy a mintavétel szinkronizációjának problémája izolálható olyan módon, hogy egy adó és egy vevő csomópont közötti kommunikációt vizsgálunk. Ilyen módon a megépített rendszer alkalmas a probléma vizsgálatára, illetve a különböző megoldások megvalósítására. A mérőrendszeren kidolgozott megoldások és mérési eredmények ismertetése után megvizsgálom, hogy a kidolgozott módszerek milyen feltételekkel használhatók nagyobb rendszerekben.

Drift

A fentiek alapján tehát a probléma akkor vizsgálható a legegyszerűbben, ha két csomópont között történő egyirányú adatforgalmat vizsgáljuk, jelfeldolgozási algoritmus megvalósítása nélkül. A működés során az adó egy analóg jelet mintavételez, amely mintákat a mintavételezés ütemében feldolgozás nélkül továbbít a vevőnek. A vevő minden beérkezett adatot feldolgozás nélkül kiad a saját mintavételi frekvenciájával működő DA-n. Az adótól érkező mintákat a soros porthoz tartozó IT rutin egy bufferbe helyezi (*mailbox*). A codec IT rutinja ebből a bufferből olvassa ki a DA-ra kiadandó értékeket. (Ezért jogos a *mailbox* elnevezés, hiszen a két rutin ezen a bufferen keresztül kommunikál.)

A fenti működésből az következik, hogy a két IT kérés a két mintavételező óra ütemében fog jelentkezni. Amennyiben a két mintavételi frekvencia pontosan megegyezne, akkor jól működne a jelátvitel egy fix késleltetéstől eltekintve.

A valóságban a két frekvencia kissé eltér, mert a mintavételi frekvenciát a codec kristályoszcillátora állítja be, amely véges pontosságú. Emiatt a két IT rutin meghívása közötti idő, azaz a minták által a mailboxban eltöltött idő mintáról-mintára változik, majd amikor a két IT kérés időben közel kerül egymáshoz, legalább egyszer felcserélődik a megszakítások addigi sorrendje, azaz kétszer egymás után fut le ugyanaz a rutin. Attól függően, hogy melyik csomópont mintavevő órája gyorsabb, két eset lehetséges. Az első esetben egy olyan adatot ír felül a beérkezett csomag interruptja, amelyet még nem

olvasott ki a codec IT. Ha viszont a codec IT jut hamarabb érvényre, akkor a mailbox tartalma még nem frissült a legutóbbi olvasás óta, azaz kétszer kerül a DA-ra ugyanaz az adat. Ennek eredményeképpen a kiadott szinusz bizonyos időközökkel tranzienseket tartalmaz az elrontott minták közelében.

Jitter

A fentiekben ismertetett tranziens jelenség a gyakorlatban nagy problémát okoz, mert a mintavétel jittere és a drift keveredésével a jelet erősen torzítja.

A jitter itt úgy nyilvánul meg, hogy az interrupt kérések időpontjának bizonyos szórása van, azaz mindkét IT rutin kérés beérkezési időpontjának szórása van egy bizonyos tartományban. Ezért amikor a drift következtében annyira közel kerül a két IT, hogy a két tartomány összeér, akkor véletlenszerűen dől el, hogy melyik IT kérés érkezik be hamarabb. Ameddig a két tartomány összeér, addig bizonytalan a működés, az interruptok beérkezési sorrendje változik. Ezért a tranziens jelenség sokkal nagyobb mértékű, mint ha csak egy minta átvitele lenne hibás. Ezt szemlélteti a 6.1 és a 6.2 ábra.

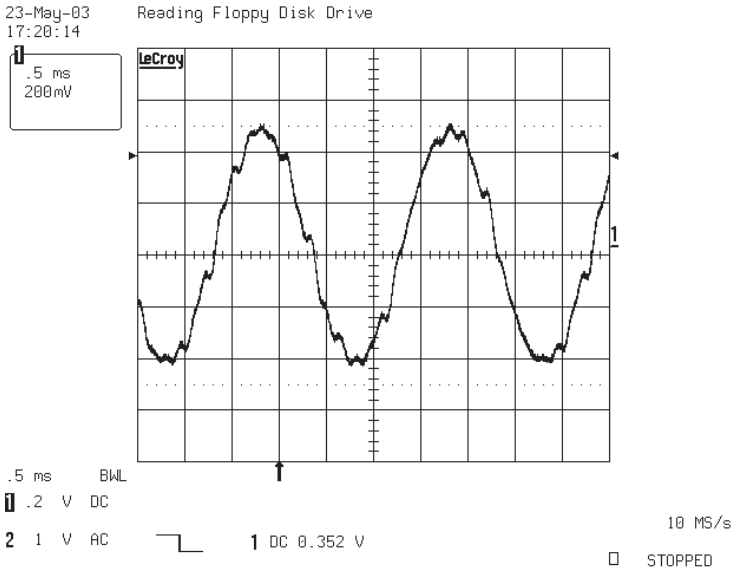
A jitternek több forrása van:

- **Mintavétel**– A mintavevő áramkör bizonyos jitterrel működik, amely hatás additív fehér zajjal modellezhető [Souders et al., 1990]. Ez a hatás a megépített rendszerben kicsi, a másik három tényező mellett elhanyagolható.

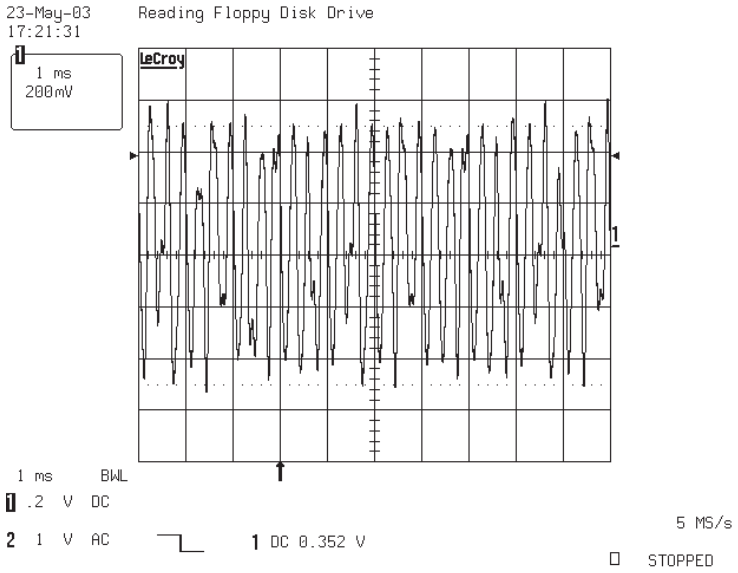
- **Küldés**– Az adó oldalon a codec interrupt kérés beérkezése és érvényre jutása közötti idő véletlenszerű, az ebből adódó jitter egy órajel periódus (T_{CLK}).

- **Kommunikáció**– A megépített rendszerben a kommunikáció szinkron soros porton történik, azaz a csomagküldés csak a soros port órajelével megegyező ütemben történhet. Mivel a port órajele a DSP órajelének nyolcada, ezért az ebből adódó jitter 8 T_{CLK}.

- **Fogadás**– A vevő oldalon is 1 T_{CLK} jittert okoz a soros port IT érvényre jutásának szórása.



6.1. ábra. Tranziens jelenség 500 Hz-es szinuszjelen



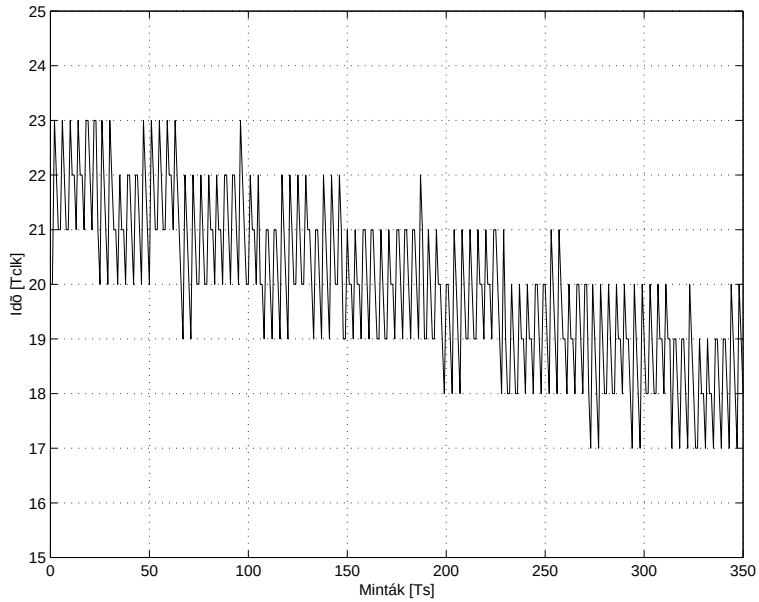
6.2. ábra. Tranziens jelenség 3 kHz-es szinuszjelen

A drift és a jitter együttes hatása

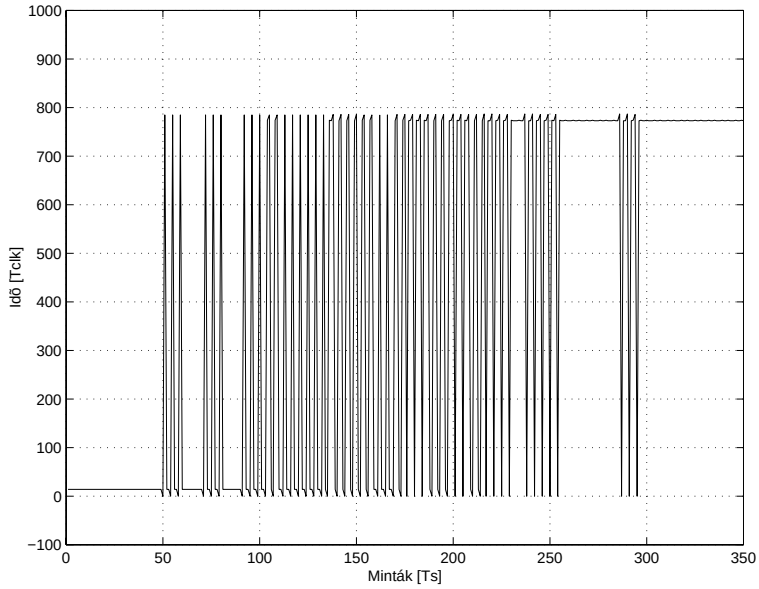
A 6.3. és a 6.4. ábrákon a drift és a jitter kombinált hatását lehet megfigyelni. Az ábrákon látható a görbék úgy készültek, hogy a vevő DSP futás közben a saját codec-hez tartozó IT rutinban minden mintánként eltárolta a legutóbbi soros vonali IT (csomag beérkezése) óta eltelt időt.

Az ábrán az egyes mintákhoz tartozó eltárolt időtartamok vannak megjelenítve. Az időmérés a DSP-ben található belső Timer áramkörrel történt, ennek felbontása megegyezik a DSP egy órajel periódusával ($T_{CLK} = 25$ nsec). Ennek megfelelően a függőleges tengely mértékegysége idő, a vízszintesé mintaszám.

A drift hatására ezen időtartamok mintáról mintára változnak, tehát az időadatokból egy egyenletes meredekségű csökkenő egyenes rajzolódik ki, ahogy a két IT egyre közelebb kerül egymáshoz. Amikor a két IT kérés időpontja nullára csökken, akkor a fentiekben bemutatott tranziens jelenség után újra a mintavételi periódusidőnek megfelelő különbséget már a timer a két IT között. Ilyen módon egy kis frekvenciájú fűrészjel áll össze az időadatokból, amelynek két részletét mutatja a 6.3. és a 6.4. ábra.



6.3. ábra. Megszakítások közötti időtartamok normál működés során



6.4. ábra. Megszakítások közötti időtartamok a tranziens szakaszban

Az ábrákból az alábbi következtetéseket lehet levonni:

Drift

A 6.3. ábrán jól megfigyelhető, a fent ismertetett jelenség, azaz hogy a mért értékek a drift hatására folyamatosan csökkennek, ahogy egyre közelebb kerülnek egymáshoz a megszakítások. A görbe meredekségből következtetni lehet a drift mértékére. Számításaim szerint ez 10 ppm körül van (10 μsec csúszás másodpercenként). Ez nagyjából megfelel a várakozásoknak, mert a codecben használt kristályok pontossága 50 ppm körül van. A tranziens jelenség bekövetkezésének periódusideje ebből 6 sec körüli értéknek adódik.

Tranziens szakasz

A fentiekben ismertetett jelenség jól látszik, az átmeneti időszak hossza átlagosan 250 minta, azaz a tranziens átlagosan 15 msec ideig tart.

Jitter

A jitter hatása a mért időadatok szórásában jelenik meg. Mértékére a tranziens szakasz hosszából lehet következtetni. A számítás a kinematikából jól ismert egyenes vonalú egyenletes mozgás analógia szerint történt. Az analóg

mennyiségek: út (s) – időváltozás(τ), idő (t)– mintaszám (N), sebesség (v)– drift (d).

$$s = v \cdot t \tag{6.1}$$

$$\tau = d \cdot N \tag{6.2}$$

A drift mértékegysége tehát időváltozás/mintaszám, de a mintaszám is kifejezhető szekundumban, így a drift mértékegysége sec/sec, azaz mértékegység nélküli.

A megszakítások beérkezési időpontjának jittere úgy képzelhető el, mintha a pontos időpont körül egy, a jitter hosszának megfelelő intervallum lenne (I). A két megszakítás pontos időpontja a driftnek megfelelően egyenletes "sebességgel" közeledik. Amikor a két intervallum összeér, akkor a 6.4 ábrán látható tranziens jelenség történik, és ez addig tart, ameddig a két intervallum el nem halad egymás mellett. A megtett "út" tehát $\tau = 2I$. A jitter értéke 6.2. felhasználásával:

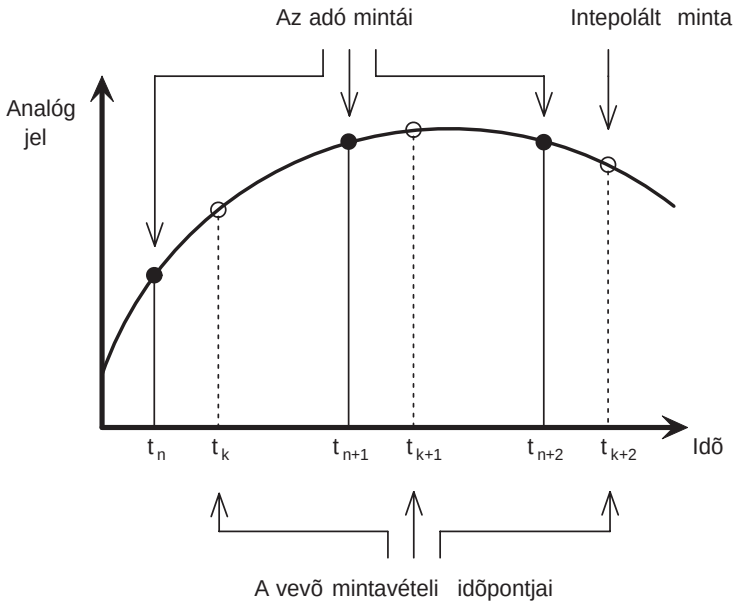
$$I = \frac{d \cdot N}{2} \tag{6.3}$$

$d = 10$ ppm $= 6.25 \cdot 10^{-10}$ sec/minta, illetve $N = 250$ minta értékeket behelyettesítve $I = 78$ nsec. A teljes kommunikáció jittere a két megszakítás jitterének összege, azaz $2I = 156$ nsec körül van. Mivel a DSP órajel periódusideje $T_{CLK} = 25$ nsec, ezért a jitter átlagosan 6.25 T_{CLK}. Ez az érték nagyjából megfelel az előzetes számításoknak.

Megjegyzés: A fenti számítások nemcsak a fenti ábrából származnak, a bemutatott ábrán kívül további méréseket is végeztem. Ennek ellenére az adatok csak tájékoztató jellegűek, az egyes adatok pontosabb mérésére pontos műszerekkel és a DSP program különböző átalakításával, az egyes egységeket külön-külön vizsgálva van lehetőség.

6.2. Mintavétel szinkronizáció interpolációval

Az előző szakaszban bemutatott jelenségek úgy küszöbölhetők ki, hogy valamilyen módon biztosítani kell azt, hogy a vevő által futtatott jelfeldolgozó alkalmazás szempontjából bemenetinek számító adatok ugyanazzal a frekvencával érkezzenek meg a vevőhöz. Ezen adatok forrása lehet a vevő saját AD átalakítója is, és érkezhetnek az adótól is. A megvalósított rendszerben alkalmazott AD és DA átalakítók mintavételi frekvenciája nem szinkronizálható direkt módon, tehát a vevő AD átalakítója felől érkező adatok frekvenciáját



6.5. ábra. Interpoláció

nem lehet hangolni. Ezért azt kell elérni, hogy az adó adatai is pontosan ezzel a frekvenciával érkezzenek.

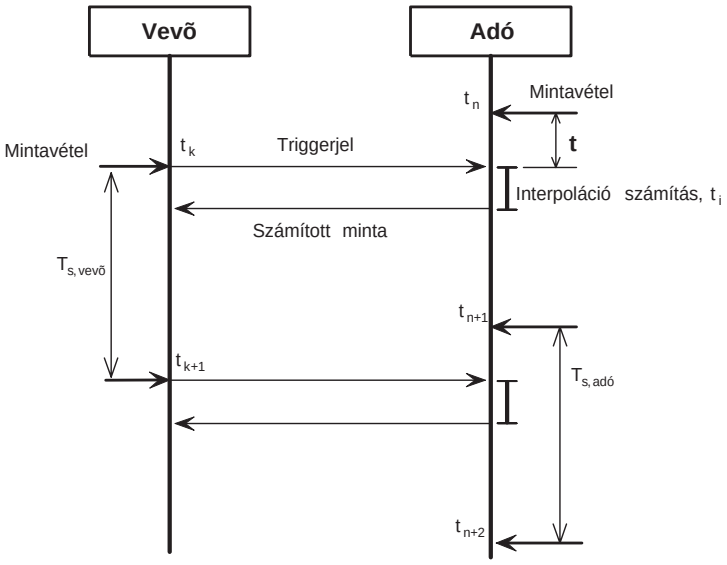
Ennek egy lehetséges megoldását az alábbiakban ismertetem. Az adó által küldeni kívánt jel a küldés előtt az adó mintavételi frekvenciája (f_S) szerint mintavételezve áll rendelkezésre. E mintákat interpolálni kell, és az interpolált jelet újramintavételezni a vevő mintavételi frekvenciája (f_D) szerint.

A fentieket a 6.5. ábra szemlélteti. Az interpoláció megvalósítására két lehetőség van.

Interpoláció a vevőnél

Ebben az esetben az adó a küldendő jel mintáin nem végez semmilyen feldolgozást, az egyes mintákat a saját mintavételi fekvenciájával elküldi a vevőnek. A vevő regisztrálja a minták beérkezési idejét, és a beérkezett mintákból kiszámítja a saját mintavételi időponhoz tartozó értékeket.

Ha ezt a megoldást kiterjesztjük több csomópontra, akkor ez a megvalósítás egyenrangú csomópontokból álló hálozatok számára optimális. A vevőnél történő interpoláció semmilyen követelményt nem ír elő az egyes adók számára, azaz mindig a vevő feladata az, hogy a számára szükséges bemeneti adatokat saját maga részére szinkronizálja. Ezzel a választással a csomópontok működése teljesen autonóm, és a rendszer minden csomópontja



6.6. ábra. A csomópontok közötti kommunikáció

egyenrangú.

Interpoláció az adónál

A megoldást a 6.6 szemlélteti. Itt a vevő a saját mintavételi időpontjaiban jelzést küld az adónak, amely regisztrálja ezeket az időpontokat. A jelzést követően rendelkezésre áll a t értéke, így saját mintái alapján kiszámítja a jelzési pillanathoz tartozó értéket. A számított értékek küldése a vevő jelzéseihez szinkronizálva történik.

Az adónál történő interpoláció inkább közelebb áll a hagyományos Master-Slave megközelítéshez, hiszen itt az adónak (slave) alkalmazkodnia kell a vevő (master) működéséhez. Amennyiben több csomópont is van, akkor mindig csak egy master lehet, amelynek órajeléhez minden más csomópontnak alkalmazkodnia kell. A 4.3 fejezetben bemutatott MAGIC alapú rendszerek tipikusan ilyenek.

6.2.1. Megvalósítás

A két megoldás két csomópontból álló hálózat esetén egyenértékű. A választás az adónál történő szinkronizálásra esett, mert ez egyenletesebb számításigényt jelentett, tekintve, hogy a vevő csomóponton valósítja meg az AFA-t.

A tervezett működés megvalósítása az alábbi módon történt. A vevő a

saját codec-jéhez tartozó megszakítási rutin legelején küld egy triggerjelet az adónak. Az adó a saját codec-jéhez tartozó rutinban nem végez jelfeldolgozást, csak eltárolja az AD által mintavett utolsó mintát egy cirkuláris bufferben, amely mindig a legutolsó mintákat tartalmazza. Az adón a soros porthoz tartozó IT akkor hívódik meg, amikor beérkezik a triggerjel a vevőtől. Ennek rutinjában történik meg az interpoláció számítása. A számításhoz szükség van a t időtartamra, amelynek mérése az adó DSP timer áramkörével történik. A timer az adó codec-jéhez tartozó megszakítási rutinban indul el, vagyis akkor, amikor adó codec-je mintát vesz. A timer-t a másik, azaz a soros porthoz tartozó megszakítási rutinja állítja meg, azaz a timer közvetlenül a 6.6 ábrán látható t-t méri.

Ez a működés gyakorlatilag azt jelenti, hogy az adó továbbra is a klasszikus online működést végzi, csak nem a saját codec-jének mintavevő órája ütemezi a működést, hanem a soros porthoz tartozó megszakítás, amely viszont a vevő codec-jének mintavételi ütemében fog bekövetkezni. A szinkron működést tehát az biztosítja, hogy áttételesen ugyanaz az óra ütemezi mindkét csomópontot.

A mérőrendszer hiányossága az, hogy az EZ-KIT kártyán az SPORT0 és az SPORT1 már hozzá van rendelve az egyes külső eszközökhöz, amelyek eldöntik az egyes egységekhez tartozó megszakítások prioritását. Ez sajnos éppen fordított, mint amelyre az adónál szükség lenne, hiszen a fenti működés szempontjából az lenne előnyös, ha a kommunikációs porthoz tartozó (SPORT1) IT lenne a magasabb prioritású, mint a codec-hez tartozó (SPORT0). Az EZ-KIT kártyán a codec-hez tartozó IT a magasabb prioritású, amelynek eredményeképpen előfordulhat, hogy vevő által küldött trigger jel csak késéssel jut érvényre az adónál, mert éppen a codec rutinjának kiszolgálása zajlik. Ez az idő mérésében hibát okoz. Ez a hiba csak a mérőrendszerben jelentkezik, és itt sem túl jelentős, mivel a codec IT meglehetősen rövid.

Ennek a problémának egy lehetséges megoldás az, hogy a codec kezelése teljesen a háttérben, DMA vezérlő által működik, és a vevő triggerjelére futó rutin csak lekérdezi a legutóbbi adatokat.

További problémát jelentett az, hogy a triggerjel beérkezése után az adónál rendelkezésre álló új t érték melyik két mintához tartozik. Ez főként a fent ismertetett tranziens szakaszban jelent gondot, azaz, amikor a két megszakítás közel esik egymáshoz, és előfordul, hogy az egyik típusú megszakítás kétszer fut le egymás után. (Ilyenkor a mért t értéke nulla vagy nagyobb, mint a mintavételi periódusidő.) A megvalósított rendszerben egy egyszerű rendező algoritmus biztosítja, hogy mindig a megfelelő minták között történjen az interpoláció.

6.2.2. Lineáris interpoláció

Az interpoláció számítására sok lehetőség kínálkozik, melyek közül legegyszerűbb a lineáris interpoláció. A 6.5. ábra jelöléseit alkalmazva a lineáris interpoláció a 6.4 egyenlet alapján számítja ki az interpolált adatokat, ahol x_n jelenti az adó által a t_n időpontban mintavételezett értéket, y_k pedig a t_k időponthoz tartozó interpolált értéket.

$$y_k = x_n + \frac{t(x_{n+1} - x_n)}{T_s} \tag{6.4}$$

Az egyenlet is mutatja azt, hogy egy mintányi késleltetést mindenképpen meg kell engedni. Ez az jelenti, hogy egy adott triggerjelhez tartozó interpolált eredmény a következő triggerjel beérkezésekor lesz számítható. A lineáris interpoláció során két mintányi késleltetéssel valósítottam meg a működést, mivel a tranziens szakaszban előfordul, hogy a triggerjelek beérkezésekor még nem áll rendelkezésre az előző trigger t-jéhez tartozó adat sem. Két ütem késleltetés viszont minden esetben elég.

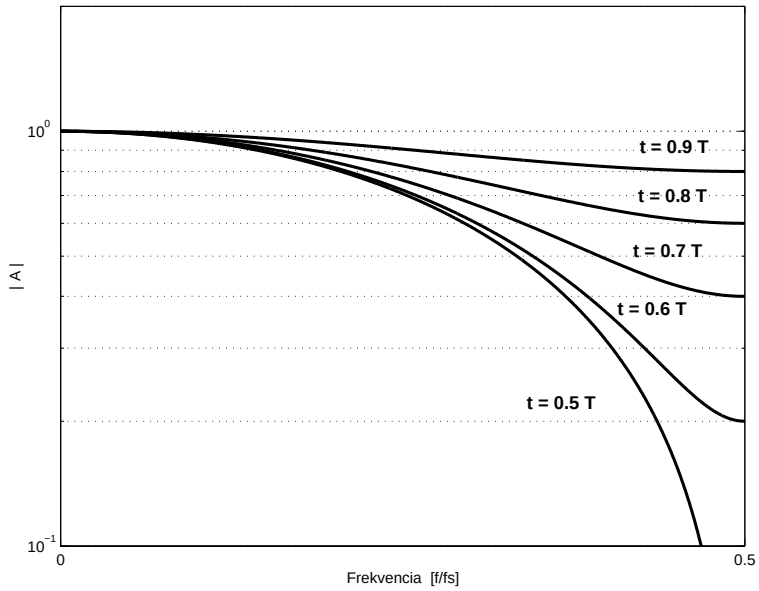
A lineáris interpoláció megvalósítása alapvetően sikeres volt, de magasabb frekvenciákon (a mintavételi frekvencia feléhez közeledve) az interpolált jel amplitúdója a drift ütemének megfelelő frekvenciával ingadozik. Ennek a jelenségnek az az oka, hogy a lineáris interpoláció átviteli karakterisztikája függ attól, hogy T intervallumon belül melyik pontot interpoláljuk. Az átviteli karakterisztika tehát az éppen aktuális t-től függ.

Ez a jelenség a lineáris interpoláció tulajdonsága. Ez úgy látható be legkönnyebben, ha a lineáris interpolációt egy 2 együtthatós FIR szűrésnek tekintjük, és megvizsgáljuk a szűrőkhöz tartozó átviteli karakterisztikákat. A két együttható (a_1 és a_2) értéke t-től függ, a következő összefüggés alapján:

$$a_1 = \frac{T_s - t}{T_s}, \quad a_2 = \frac{t}{T_s} \tag{6.5}$$

Például az intervallum középső pontjainak ($t = T_s/2$) számításakor az együtthatók [0.5 0.5], az intervallum negyedénél ($t = T_s/4$) az együtthatók [0.75 0.25]. A 6.7 ábrán néhány t értékhez tartozó átviteli függvények abszolút értékei láthatók, a (0..F_s/2) intervallumban. Jól látszik, hogy a mintavételi frekvencia feléhez közeledve egyre nagyobb mértékű a karakterisztikák eltérése, amely magyarázatot ad a jel amplitúdójának egyre nagyobb mértékű ingadozására.

Ezt a problémát interpoláló szűrés alkalmazásával lehet kiküszöbölni.



6.7. ábra. A lineáris interpoláció átviteli karakterisztikái

6.2.3. Interpoláló szűrés

DSP-k körében természetes választás a lineáris interpolálás helyett interpoláló szűrés alkalmazása, amely digitális jelek mintavételi frekvencia növelésének szokásos módja. A mintavételi frekvencia N-szeresre növelése általában úgy történik, hogy a bementi jel mintái közé $N-1$ darab nulla értékű minta kerül, majd az így kapott jelet egy olyan aluláteresztő FIR szűrővel szűrik, amely az új (magasabb) mintavételi frekvencia $2N$-ed részéig enged át.

Az interpoláció ezen megvalósítása sokkal számításigényesebb, mint a lineáris interpoláció, továbbá a FIR szűrésből a szűrő fokszámával arányos fix késleltetés is megjelenik. Lineárfázisú FIR szűrő esetén a késleltetés $(M-1)/2$ ahol a M szűrő fokszáma.

Az interpoláló szűrés implementációja során a számítás az elvi működéshez képest nagy mértékben egyszerűsíthető. Az ilyenkor szokásos első egyszerűsítés az, hogy a beszúrt nulla értékű mintákkal történő MAC (Multiple and ACcumulate) műveleteket valójában nem végzi el a DSP. Ez úgy valósítható meg, hogy a valódi (nullák nélküli) mintákat tartalmazó vektorral az N-edik szűrőegyüttthatókat tartalmazó vektort kell konvolválni. Ilyen módon nem kell külön nullákat beszúrni, e megoldás eredménye egyenértékű azzal, mintha a nullákat tartalmazó adatokat a szűrőegyüttthatókkal konvolválnánk.

A másik egyszerűsítés az, hogy az interpoláció számításakor már rendelkezésre áll a t-érték, azaz már lehet tudni, hogy az interpolált jelből melyik mintára lesz szükség. Ezért igen jelentős egyszerűsítés az, ha az összes N darab helyett, csak ennek az egy mintának a számítása történik meg.

Az így számított érték pontosabb lesz, mint a lineáris interpolációval számított, valamint az átviteli karakterisztikája sem ingadozik.

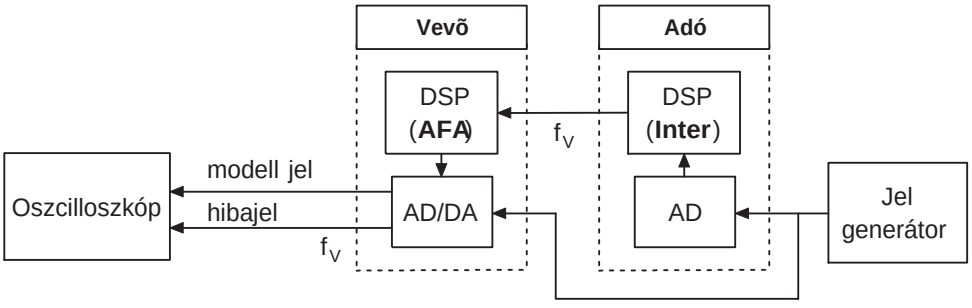
6.2.4. Interpoláló szűrés lineáris interpolálással

Az interpoláló szűrés annál jobb eredményt ad, minél nagyobb N értéke. Az N növeléséenk az szab határt, hogy nagyobb N-ekhez nagyobb igényű, azaz nagyobb fokszámú FIR szűrő szükséges. Ez egyrészt a számításigényt is növeli, másrészt egyre nagyobb késleltetéssel szolgáltatja a számított mintákat.

Ezért egy jó kompromisszumos megoldás, ha egy nem túl nagy fokszámú FIR szűrővel számított minták között lineáris interpolációval számítjuk a t-nek megfelelő értéket. Ekkor a megvalósítás során a számításigény körülbelül a kétszerese lesz a fent ismertetett egyszerű szűréshez képest, mert két minta értékét kell kiszámolni, majd közöttük a 6.4 egyenlet szerinti műveletet elvégezni. (Ez utóbbinak számításigénye elhnyagolható a szűrés számításigénye mellett.)

6.2.5. Mérési eredmények

A fentiekben ismertetett megoldásokat az 5. fejezetben ismertetett mérőrendszeren implementáltam, a 6.2.1. szakaszban ismertetett módon, azaz az interpoláció az adónál történt.



6.8. ábra. A mérési elrendezés

A mérési elrendezést személélteti a 6.8. ábra. A mérések során az adó egyik AD átalakítójára egy szinuszjelet kapcsoltam, amelynek mintáin az adó

elvégezte az interpoláció számítását a vevő által küldött triggerjelnek megfelelő időpontokban, majd a számított mintákat a triggerjelhez szinkronizálva elküdte a vevőnek. A vevő a saját codec-hez tartozó IT rutinban egyrészt küldi az adónak a triggerjelet, másrészt ebben a rutinban van megvalósítva a az 5.4.2. szakaszban ismertetett AFA. Az AFA bemenete az adótól érkező jel. A vevő csomópont analóg kimenetén általában a beérkezett jelet jeleníti meg, illetve frekvenciamérések esetén az AFA által modellezett jelet és a hibajelet.

Megfigyelések digitális oszcilloszkóppal

A megvalósítás mindhárom esetben (lineáris interpoláció, interpoláló szűrés, iterpoláló szűrés lineáris interpolálással kiegészítve) sikeresnek mondható.

Lineáris interpoláció esetén a driftből származó frekvenciakülönbség eltűnt, viszont a fázis enyhén ingadozik. A 6.1. szakaszban bemutatott tranziens hatás teljesen eltűnt a megvalósított rendező algoritmusnak köszönhetően. A lineáris interpoláció 6.2.2. szakaszban bemutatott tulajdonsága, mely szerint az átviteli karakterisztika ingadozik, magasabb frekvenciákon erősen jelentkezik.

Az interpoláló szűrés esetén a lineáris interpolációhoz hasonlóan eltűnt a frekvencikülönbség, ugyanakkor a lineáris interpolációra jellemző átviteli függvény ingadozás megszűnt, azaz ez a módszer a teljes frekvenciatartományban használható. A lineáris interpolációnál tapasztalt fázisingadozás is csökkent, de még látható. A lineáris interpolációval kiegészített szűrés jele a teljes frevenciatartományban stabil, a fázisingadozás megszűnt.

Itt említem meg, hogy interpolációs szűrés esetén továbbra is tapasztalható az interpolálatlan esetben fellépő tranziens jelenség, bár annál lényegesen kisebb mértékben. Ennek oka az, hogy a lineáris esetben alkalmazott rendezési algoritmus az interpoláló szűrés esetén nem működik tökéletesen.

Frekvenciamérés AFA-val

Az oszcilloszkópon tapasztalható jelenségek pontos mérésére az AFA-val lehetséges, mert segítségével pontosan meg lehet mérni az interpolált jel frekvenciáját.

Az adatok felvétele 200 Hz-es szinuszjel frekvenciájának mérésével történt. A szinuszjelet Brüel and Kjaer BK-1051 nagypontosságú szinuszgenerátor állította elő. A 6.1. táblázatban összefoglaltam a mért frekvenciák relatív hibáit a saját codec-ek által mintavételezett jel frekvenciájához viszonyítva. A táblázatban szereplő értékek számítása MATLAB segítségével történt, minden esetben több ezer mérési adatot átlagolva.

Interpoláció nélkül	$1.1164 \cdot 10^{-5}$
Lineáris interpláció	$0.2580 \cdot 10^{-5}$
Interpoláló szűrés	$1.0820 \cdot 10^{-5}$
Interpoláló szűrés lineáris interpolációval kiegészítve	$1.2036 \cdot 10^{-5}$

6.1. táblázat. A mért frekvenciák relatív hibájának abszolút értéke

Az interpolálás nélküli esetben a várt eredményt kaptuk, azaz a kristály pontosságának megfelelő eltérés mutatkozott. A lineáris interpolációval számított jel ennél egy nagyságrenddel kisebb hibával közelíti a helyes frekvenciaértéket.

A meglepő eredmény az, hogy az oszcilloszkópos mérések során stabil frekvenciájú interpoláló szűrés körülbelül ugyanakkora hibával közelíti a helyes frekvenciaértéket, mint amekkora a kristályok eltéréséből adódik.

Ennek oka a mérőrendszer fentebb ismertetett azon sajátossága, hogy a saját-codec interrupt magasabb prioritású, mint a kommunikációhoz tartozó soros port IT. Ebből ugyanis az adódik, hogy az alacsonyabb prioritású IT rutinban megvalósított interpoláció számítás során tiltani kell a codec megszakítását, mert ez a megszakítás felülírná azon adatokat, amelyeken a számítások éppen történnek. A késve érvényre jutó codec IT viszont azt jelenti, hogy a t mérésében hiba történik, hiszen a timer áramkör késve indul.

E hiba olyan módon küszöbölhető ki, hogy meg kell akadályozni, hogy a két IT ugyanazon memóriaterülethez férjen hozzá, azaz szükség van az IT rutinok elválasztására, például egy *mailbox* segítségével, vagy a javasolt DMA átvitel alkalmazásával.

A fenti mérési eredmények nem jelentik azt, hogy az interpoláló szűrés nem alkalmas a mintavételi szinkronizációra, csak az adott mérőrendszer esetén ezt körülbelül ugyanakkora hibával biztosítja, mint amekkora a hiba konkrét kristályok eltéréséből is adódik.

7. fejezet

Összefoglalás

7.1. Az eredmények értékelése

Jelen dolgozat a beágyazott rendszerek egy speciális típusával, az elosztott jelfeldolgozó rendszerekkel foglalkozik. A dolgozat elején részletesen bemutattam az elosztott jelfeldolgozó rendszerekre jellemző működési sajátosságokat, és ismertetem azokat a tulajdonságokat, amelyek megkülönböztetik e rendszereket a többi beágyazott rendszertől.

Elosztott rendszerekben mindig központi kérdés az egységek közötti kommunikáció megvalósítása, ezért rögzítettem azokat a követelményeket, amelyeket egy hard real-time jelfeldolgozást végző rendszer kommunikációs protokolljának teljesítenie kell.

A harmadik fejezetben a beágyazott rendszerek körében használatos szabványos kommunikációs protokollokat vizsgáltam meg, elsősorban abból a szemszögből, hogy az egyes protokollok milyen feltételekkel tudják a fenti követelményeket teljesíteni. A különböző protokollok vizsgálata után kiemelten, egy külön fejezetben foglalkoztam az Ethernet hálózati technológiával, amely elsősorban elterjedtsége és univerzalitása miatt került előtérbe.

Egy egyszerű elosztott jelfeldolgozó rendszert meg is építettem, amelynek részletes bemutatása az ötödik fejezetben történt meg. A megépített rendszer elsődleges célja az, hogy lehetővé tegye az elosztott jelfeldolgozás sajátosságainak és speciális problémáinak gyakorlati vizsgálatát, mérések elvégzését.

A mérőrendszeren egy jelfeldolgozó alkalmazást is implementáltam annak érdekében, hogy a rendszer működőképességét bizonyítsam. Ez az alkalmazás az adaptív Fourier-analizátor, amelyre azért esett a választás, mert valós-idejű periodikus jelek pontos frekvenciamérését teszi lehetővé. Az implementáció sikeres volt, az alkalmazás a vártnak megfelelően működik.

A dolgozat utolsó fejezete a mintavétel szinkronizáció problémájával fog-

lalkozik. A probléma részletes elemzése után egy olyan univerzálisan használható megoldást mutattam be, amely a különböző órák által mintavételezett jeleket interpolációval és újramintavételezéssel szinkronizálja.

Az elvi bemutatást követően a módszer három különböző változatát implementáltam a mérőrendszeren. A rendszeren végzett megfigyelések és mérések bizonyították, hogy az elvi megoldás helyes, a gyakorlatban alkalmazható. A mérési eredmények ennek ellenére nem tökéletesek, de a hibák a mérőrendszer sajátosságaira, korlátaira vezethetők vissza. A mérések során a korábban implementált adaptív Fourier-analizátort frekvenciamérésre használtam fel.

7.2. Továbbfejlesztési lehetőségek

A dolgozatban bemutatott rendszer és a kidolgozott módszerek alapját képezhetik az elosztott jelfeldolgozó rendszerek körében folytatott további vizsgálatoknak. A mintavétel szinkronizáció egy jó példája annak, hogy milyen újszerű problémák merülnek fel az elosztott megvalósítású jelfeldolgozás során, amelyeket egy megvalósított rendszerben kezelni kell.

Az eredeti célkitűzéseknek megfelelően a rendszer alkalmas arra is, hogy szimulálja speciálisan az Ethernetre jellemző működési sajátosságokat (csomagkiesés, késleltetési jitter) a soros port megfelelő programozásával. Ezen vizsgálatok elvégzése mindenképpen érdekes lehetőség, mert a rendszeren labor körülmények között lehet vizsgálni az egyes elosztott algoritmusok viselkedését a kommunikáció hibáinak függvényében.

Ezen szimulációs vizsgálatok eredményei alapját képezhetik egy olyan elosztott jelfeldolgozó rendszer megépítésének, amelynek csomópontjai valóban Ethernet hálózaton kommunikálnak.

Irodalomjegyzék

[Aurrecoechea et al., 1998] Aurrecoechea, C., Campbell, A. T., and Hauw, L. (1998). A survey of QoS architectures. *Multimedia Systems*, 6(3):138–151.

[CAN, 1991] CAN (1991). Specification, version 2.0. Bosch Corporation. http://www.can.bosch.com/.

[Cisco, 2002] Cisco (2002). *Internetworking Technology Handbook*. Cisco Systems Inc. http://www.cisco.com/univercd.

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

[FlexRAY, 2002] FlexRAY (2002). Requiremenets specification. FlexRAY Consortium. http://www.flexray.com/.

[Francis, 2001] Francis, H. (2001). ARM DSP-enhanced extensions. White Paper, ARM Ltd. http://www.arm.com/.

[Gonzalez et al., 1996] Gonzalez, O., Sen, S., Ramamritham, K., and Stankovic, J. A. (1996). Incorporation of multimedia capabilities in distributed real-time applications. *ACM SIGOPS Operating Systems Review*, 30(4).

[Gustafson, 2002] Gustafson, N. (2002). Succesful DSP design depends on methodology and tools. *RTC Magazine*, (5).

[Kopetz, 1997] Kopetz, H. (1997). *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers.

[Kopetz and Ochsenreiter, 1987] Kopetz, H. and Ochsenreiter, W. (1987). Clock synchronization in distributed real-time systems. *IEEE Trans. on Computers*, 36(8).

[Kovács, 2002] Kovács, K. (2002). Nagyméretű transzformátorok zajának aktív csillapítása. Diplomaterv. Budapesti Műszaki és Gazdaságtudományi Egyetem, Villamosmérnöki és Informatikai Kar.

[Kurose et al., 1984] Kurose, J. F., Schwarz, M., and Yemini, Y. (1984). Multiple-access protocols and time-constrained communication. *ACM Computing Surveys*, 16(1):43–70. Association for Computing Machinery.

[LIN, 2002] LIN (2002). Specification package, revision 1.3. LIN Consortium. http://www.lin-subbus.org/.

[LON, 1999] LON (1999). Introduction to the LonWorks system. Echelon Corporation. http://www.echelon.com/products/lonworks/.

[MAGIC, 2003] MAGIC (2003). Media-accelerated global information carrier. Engineering Specification, Revision 3.0c. Gibson Guitar Corp. http://www.gibsonmagic.com.

[Nagy, 1992] Nagy, F. (1992). Measurement of signal parameters using nonlinear observers. *IEEE Trans. on Instrumentation and Measurement*, IM-41:152–155.

[Péceli, 1986] Péceli, G. (1986). A common structure for recursive discrete transforms. *IEEE Trans. on Circuits and Systems*, CAS-33:1035–36.

[SHARC, 1997] SHARC (1997). *ADSP-2106x SHARC User's Manual*. Analog Devices, Inc., second edition edition.

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

[Souders et al., 1990] Souders, M., Flach, D., Hagwood, C., and Yang, G. (1990). The effects of timing jitter in sampling systems. *IEEE Trans. on Instrumentation and Measurement*, 39(1):80–85.

[Stankovic, 1988] Stankovic, J. A. (1988). Real-time computing systems: The next generation. In J. A. Stankovic, K. R., editor, *Hard Real-Time Systems*, chapter 2, pages 14–37. IEEE Computer Society Press.

[Tannenbaum, 1996] Tannenbaum, A. S. (1996). *Computer Networks*. Prentice Hall.

[Zhao and Ramamritham, 1987] Zhao, W. and Ramamritham, K. (1987). Virtula time csma protocols for hard real-time communication. *IEEE Trans. on Software Engineering*, 13(8).