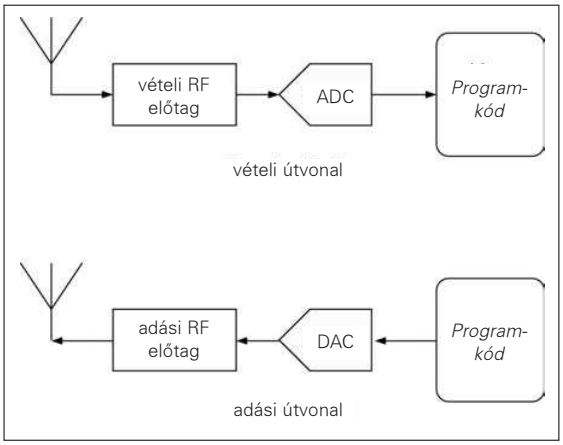
GNU Radio: A rádiófrekvenciás világ felfedezésének eszközei

A program alapú rádiók célja az, hogy a programkódot és az antennát a lehető legközelebbi kapcsolatba hozzák egymással. A GNU Radio mindenkinek biztosítja azokat az eszközöket, amelyek segítségével csatlakozhat a gyors processzoroknak köszönhető távközlési forradalomhoz.

A programrádió lényegében az antenna és a programkód viszonyának lehető szorosabbá tételét jelenti. Gyakorlatilag minden, a rádiókészülékkel kapcsolatos problémát programozási problémává alakít. Ennek a rádiónak a legfontosabb ismertetőjele, hogy egy program határozza meg az elküldött hullámformákat, és ugyancsak program végzi a fogadott hullámok visszaalakítását. Ez a működésmód szöges ellentétben áll a megszokott rádiókéval, amelyekben a jelfeldolgozást vagy digitális lapkákkal kiegészített vagy tisztán analóg áramkörök végzik. A GNU Radio egy szabadon hozzáférhető programozási eszközkészlet az említett programrádiók készítéséhez.
A programrádió valóságos forradalmat jelent a rádiók tervezésében, az így létrejövő készülékek üzem közben is változhatnak, korábban elképzelhetetlen lehetőségeket kínálva használójuknak. A programrádiók természetesen képesek mindarra, amire hagyományos társaik, de minket most sokkal inkább a programok használatából fakadó rugalmasság érdekel. A ma még igencsak korlátozott képességű, mindig csak egy adott feladatra használható készülékek helyett a közeli jövőben többfunkciós távközlési eszközöket fogunk használni. Képzeljünk el egy olyan készüléket, amely mobiltelefonként lehetővé teszi GPRS alapú kapcsolatok létesítését, de a 802.11 Wi-Fi és a 802.16 WiMax szabvánnyal valamint a műholdas összeköttetésekkel is megbirkózik – hogy a jövőbeli új megoldásokról most ne is beszéljünk.
A legérdekesebb mindebben, hogy központok nélküli távközlési rendszereket lehet majd építeni. Ha megnézzük a meglévő rendszereket, azt látjuk, hogy túlnyomó részük infrastruktúra alapú. A jelenlegi rádió- és TV-adások egyirányú csatornákon jutnak el hozzánk, szigorúan szabályozzák őket, és az általuk szolgáltatott tartalmat számos szervezet ellenőrzi. A mobiltelefonok ugyan kényelmessé teszik életünket, ám az általuk támogatott szolgáltatások köre a szolgáltató a vállalat érdekeit szolgálja, és nem a mienket.
A központosított rendszerek megnehezítik az újdonságok bevezetését. Ha ezt nehezen tudjuk elképzelni, akkor gondoljunk csak az internetre, és máris helyben vagyunk.
A mobiltelefonnak nem kellene tehetetlenül csüngenie a szolgáltató sokszor korlátozottan elérhető, korlátozott szolgáltatásokat nyújtó hálózatán, hanem igazán okos eszközzé válhatna. A hálózatot maguk a felhasználók tulajdonában lévő készülékek alkotnák. A készülékek egymás között hálót hoznának létre, egyeztetnék a háttérszolgáltatásokat; és minden újszerű megoldásra, szolgáltatásra és alkalmazásra nyitottak lennének.



1. ábra Általános programrádió blokkdiagramja

Blokkdiagram

Az *1. ábrán* egy általános programrádió blokkdiagramja látható. Ha meg akarjuk érteni a programrész működését, először a szerkezet jellemzőit kell gyorsan áttekintenünk. Ha megvizsgáljuk az 1. ábra vételi útvonalát, akkor egy antennát, egy rejtélyes RF előtagot, egy analóg-digitális átalakítót (ADC) és egy programkódot láthatunk. Az analóg-digitális átalakító köti össze a folytonos analóg jelek és a programból is kezelhető, diszkrét digitális minták világát.
Az ADC-k két fő jellemzője a mintavétel gyakorisága és a dinamikatartomány. A mintavételi gyakoriság azt mutatja meg, hogy az ADC másodpercenként hányszor méri meg

1. kódrészlet Hello Világ! (tárcsahang megszólaltatása)

```
#!/usr/bin/env python

from GnuRadio import *

def build_graph ():
    sampling_freq = 32000
    ampl = 8192

    fg = gr_FlowGraph ()

    src0 = GrSigSourceS (
        sampling_freq, GR_SIN_WAVE, 350, ampl)

    src1 = GrSigSourceS (
        sampling_freq, GR_SIN_WAVE, 440, ampl)

    sink = GrAudioSinkS ()

    fg.connect (src0, sink)
    fg.connect (src1, sink)

    return fg

if __name__ == '__main__':
    fg = build_graph ()
    fg.start ()         # szál(ak) létrehozása
➥fork hívásokkal és visszatérés
    raw_input ('A kilépéshez nyomd le az Enter
➥gombot: ')
    fg.stop ()
```

az analóg jelet. A dinamikatartomány a legkisebb és a legnagyobb megkülönböztethető jel közötti tartományt jelenti, amely az ADC digitális kimenetének és az átalakító kialakításának függvénye. Egy nyolcbites átalakító például legfeljebb 256 jelszintet képes ábrázolni, míg egy 16 bites átalakító már 65536-ot. Általában elmondhatjuk, hogy az átalakító fizikai kialakításától és árától függ, hogy mekkora kompromisszumot kell kötnünk a nagyobb mintavételi gyakoriság vagy a szélesebb dinamikatartomány tekintetében.

Mielőtt elmerülnénk a programozási kérdésekben, nem árt némi elméleti alapozást. 1927-ben a svéd fizikus és villamosmérnök ***Harry Nyquist*** fogalmazta meg azt a szabályt, amely szerint analóg-digitális átalakításnál az ADC mintavételi frekvenciájának legalább kétszer akkorának kell lennie, mint a mintavételezett jel sávszélességének. Ellenkező esetben ugyanis a jel nem állítható hibátlanul helyre, azaz jeltorzulás történik. Hasonló jelenségnek lehetünk tanúi, amikor a régi westernfilmekben úgy látjuk, mintha a lovaskocsik kerekei hátrafelé forognának. Ilyenkor a kamera mintavételi gyakorisága nem volt elég nagy ahhoz, hogy egyértelműen rögzíteni tudja a küllők állását.

Tegyük fel, hogy a jelek felülről korlátosak, vagyis a minket érdeklő sávszélesség 0 és `fmax` közötti. Ilyenkor a Nyquist-szabály szerint a mintavételi frekvencia értéke legalább 2 * `fmax` kell legyen. Igen ám, de ha az ADC-nk 20 MHz-en fut, akkor hogyan fogjuk hallgatni a 92,1 MHz-en sugárzott rádióadást? Pontosan ezért van szükség az RF előtagra.

A vevőoldali RF előtag a bemenetére adott frekvenciákat a kimeneten alacsonyabb tartományba tolva adja ki. Az RF például használható arra is, hogy a 90-100 MHz tartományba eső jeleket a 0-10 MHz tartományba tegyük át.

A legtöbb esetben elég az is, ha az RF előtagot egyfajta fekete dobozként kezeljük, amelynek vezérlő bemenetén csupán a megváltoztatandó frekvenciatartomány közepét kell megadnunk. Tényleges példaként említhetném azt a kábelmodemes vevőegységet, amelyet mi is sikerrel alkalmaztunk, és amely az 50-800 MHz tartomány egy 6 MHz-es szeletét teszi át az 5,75 MHz-es közepű tartományba. A kimeneti tartomány középső frekvenciáját középfrekvenciának (intermediate frequency, röviden IF) nevezzük.

Ha a végletekig egyszerűségre törekszünk, az RF előtagot akár el is hagyhatjuk. Van olyan, akinek a GNU Radio próbálgatása közben sikerült AM és rövidhullámú adásokat hallgatnia úgy, hogy egy száz méteres kábelt közvetlenül egy 20 millió minta/másodperc mintavételű ADC-hez csatlakoztatott.

Lássuk a programokat!

A GNU Radio tartalmaz egyrészt elemi jelfeldolgozó függvényeket, másrészt ezek egységes alkalmazássá gyúrásához kiegészítőket. A programozó a rádiót úgy állítja össze, mintha egy gráfot rajzolna meg. A gráf pontjai az elemi jelfeldolgozó függvények, az élek pedig a köztük folyó adatáramlást jelképezik. A jelfeldolgozó függvények C++ nyelven készültek. A függvények lényegében végtelen, a bemenetük felől a kimenetük felé haladó folyamokat alakítanak át. A függvények fontos jellemzője bemeneteik és kimeneteik száma, illetve az ezeken keresztülhaladó adatok típusa. Leginkább kisméretű egész (short), lebegőpontos (float) és komplex mennyiségeket használunk.

Bizonyos jelfeldolgozók csak kimenettel vagy csak bemenettel rendelkeznek, ezek adatforrásként és -nyelőként viselkednek. A források közt találunk olyat, amely fájlból vagy ADC-ről olvas, a nyelők pedig képesek fájlba, digitális-analóg átalakítóra (DAC) vagy grafikus kijelzőre írni.

A GNU Radio körülbelül száz elemi jelfeldolgozó függvényt tartalmaz, de újak készítése sem ördöngösség.

A gráfok alapvetően C++ nyelven állíthatók össze és futtathatók, de működő egység összeállítása Python alatt a legegyszerűbb. Az *1. kódrészlet* a „Hello Világ” GNU Radio-féle változata. Két szinuszhullámot állít elő, majd a hangkártyára küldi őket, az egyiket a bal, a másikat a jobb csatornára.

Első lépésünk egy áramlási gráf (flow graph) létrehozása, ebbe kerülnek majd az elemi jelfeldolgozó függvények és a közöttük fennálló kapcsolatok. A két szinuszhullám a `GrSigSourceS` hívások révén jön létre. Az S utótag jelzi, hogy a forrás `short` kimeneteket ad. Az egyik szinuszhullám 350, a másik 440 Hz-es frekvenciájú. Ha együtt szólaltatjuk meg őket, mintha az Amerikában megszokott tárcsahangot hallanánk.

A `GrAudioSinkS` olyan nyelő (sink), amely bemenetét (általában egy vagy két `short` típusokat tartalmazó folyam) a hangkártyára írja ki. A három elemi jelfeldolgozó függvényt az áramlási gráf `connect` függvényével kapcsoljuk össze.

Amikor készen áll a gráf, elindítjuk. A `start` hívás hatására egy vagy több szál jön létre (`fork` hívásokkal), ezekben futnak a gráf által megadott számítások; a vezérlés pedig azonnal visszakerül a hívóhoz. Ilyenkor egyszerűen csak várjunk, amíg a felhasználó le nem nyomja valamelyik billentyűt.

Teljes értékű FM vevő

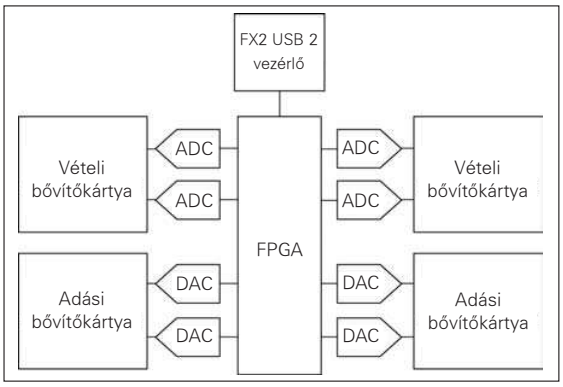
A 2. *kódrészlet* egy leegyszerűsített, de működő FM vevő összeállítását szemlélteti. Magába foglalja az RF vezérlését és a szükséges jelfeldolgozókat is. Ebben az esetben egy kábelmodemes vevőegységből épített RF előtag használatáról és egy 20 millió minta/másodperces analóg-digitális átalakító használatával számoltunk. Ahogy a Hello Világ!-os példánál, itt is, létrehozzuk a gráfot, összekötjük az elemi jelfeldolgozókat, majd kiadjuk az indulási parancsot. Forrásunk ebben az esetben a nagysebességű ADC, a `GrHighSpeedADC`. Ezt a `GrFreqXlatingFIRfilterSCF` követi, ami egy véges impulzusválaszú (FIR) szűrő. Ez választja ki a kívánt FM állomást, majd alakítja alapsávúra (0Hz, DC) a jelet. A 20 millió minta/másodperces átalakító és a kábelmodemes vevőegység használatával nagyjából a teljes tartomány egy 6 MHz-es szeletét vesszük. Ebbe a szeletbe tíz vagy akár több FM állomás is eshet, az általunk kívántat a `GrFreqXlatingFIRfilterSCF` segítségével választhatjuk ki. Most azt az állomást választjuk, amely az RF előtag IF értékének közepére esik (5,75MHz). A szűrő kimenete komplex minták folyama, 160 ezer minta/másodperc gyakorisággal. A komplex alapsávi jelet a `GrQuadratureDemodCF`-fel etetjük meg, ez végzi a tényleges FM demodulációt. A `GrQuadratureDemodCF` a szomszédos komplex minták szögeinek kivonásával végzi munkáját, lényegében differenciálva a frekvenciát. A `GrQuadratureDemodCF` kimenete tartalmazza a bal plusz jobb mono hangjelet, a 19 kHz-es elválasztójelet, a 38 KHz-es közepű bal mínusz jobb sztereojelet, illetve az e fölötti alhordozókat. Végső lépésként a jelet keresztülküldjük egy aluláteresztő szűrőn, majd megtizedeljük, miközben csak a bal plusz jobb hangsávot őrizzük meg. A végfok a hangkártya, 32000 minta/másodperc mintavételi gyakorisággal. A jelfeldolgozással kapcsolatosan a GNU Radio Wikiben lehet további tudnivalókat találni.

Grafikus felületek

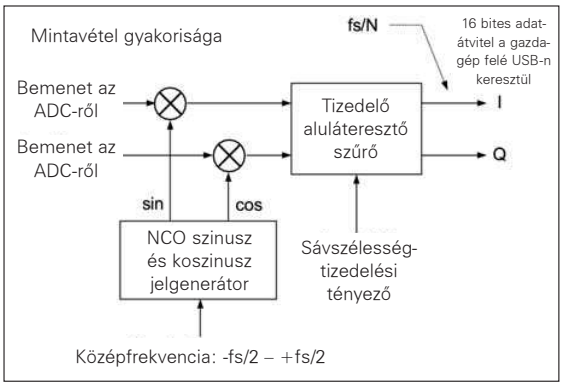
A GNU Radio alkalmazások grafikus felületei Python alatt készülnek, a kezelőfelületek tetszőleges, Python alól elérhető eszközkészlettel összeállíthatók. A géptípusok közötti hordozhatóság elősegítése miatt mi a wxPython használatát javasoljuk. A valós idejű C++ folyamatgráf és a Python-világ közötti kapcsolattartáshoz a GNU Radio megfelelő folyamatközti adattovábbításra alkalmas elemi eljárásokat biztosít.

Gépkövetelmények

A GNU Radio csaknem hardverfüggetlen. Napjaink sok gigahertzes, egyciklusú lebegőpontos egységgel felszerelt szuperskalár processzorai révén még asztali számítógépekkel is elég komoly digitális jelfeldolgozást lehet végezni. Egy 2 GHz órajelű Pentium vagy Athlon processzor 2 milliárd lebegőpontos FIR mintát tud kiértékelni másodpercenként. Majdnem kizárólag programokra alapozva olyan távközlési rendszereket építhetünk, amilyenekre néhány évvel ezelőtt még csak nem is gondolhattunk.



2. ábra Általános programrádió-periféria, USRP



3. ábra Digitális lefelé transzponáló elem

A számítási igény természetesen attól függ, hogy pontosan mit akarunk csinálni, de egy 1-2 GHz-es gép 256 MB memóriával elég kell legyen. Természetesen az analóg világ és a számítógép kapcsolatát is meg kell valahogy teremtenünk. Ha nem akarunk túl sokat költeni, egy jobb minőségű, 96 kHz-es, 24 bites hangkártya megteszi, de végső esetben gépünk beépített hangkártyáját is használhatjuk. Bármelyik megoldást is választjuk, viszonylag keskeny sávban fogunk tudni jelfeldolgozást végezni, és valamilyen keskenysávú RF előtagot is be kell szereznünk.
Egy másik megoldás készen kapható, nagysebességű, PCI foglalatú analóg-digitális átalakító kártya beszerzése. Ezek akár 20 millió mintát is képesek előállítani másodpercenként, ám áruk egy teljes számítógépével vetekszik. Az ilyen nagysebességű kártyákhoz kábelmodemes vevőegységet érdemes RF előtagként illeszteni.

A Universal Software Radio Peripheral

A legjobb megoldás a Universal Software Radio Peripheral (általános programrádió-periféria, USRP). A 2. *ábrán* az USRP blokkdiagramja szerepel. Az USRP Matt Ettus agyszüleménye, lényegében egy rendkívül rugalmas, USB csatolós készülék, amely képes a rádiófrekvenciás világhoz kapcsolni számítógépünket. Az USRP egy kisméretű alaplapot tartalmaz, amely legfeljebb négy darab 12 bites 64 millió minta/másodperces ADC-t, négy 14 bites, 128 millió minta/másodperces DAC-t, egy egymillió kapus, programozha-

2. kódrészlet Egyszerű FM vevő

```
#!/usr/bin/env python

# egyszerű FM vevő

from GnuRadio import *

#
# gr_FlowGraph visszaadása
#
def build_graph (IF_freq):
    input_rate = 20e6

    CFIR_decimate = 125
    RFIR_decimate = 5
    fm_demod_gain = 2200

    quad_rate = input_rate / CFIR_decimate
    audio_rate = quad_rate / RFIR_decimate

    volume = 1.0

    src = GrHighSpeedADCSourceS (input_rate)

    # FIR szűrőminták számítása a csatornaválasz-
    # táshoz
    channel_coeffs = \
      gr_firdes.low_pass (
        1.0,          # erősítés
        input_rate,   # mintavételi gyakoriság
        250e3,        # aluláteresztés vágási
                      # frekvenciája
        8*100e3,      # átvitt sáv szélessége
        gr_firdes.WIN_HAMMING)

    # bemenet: short; kimenet: complex
    chan_filter =
      GrFreqXlatingFIRfilterSCF (CFIR_decimate,
            channel_coeffs, IF_freq)

    # bemenet: complex; kimenet: float
    fm_demod =
      GrQuadratureDemodCF (volume *
                           fm_demod_gain)

    # FIR szűrőminták számítása a hangszűrőhöz
    width_of_transition_band = audio_rate / 32
    audio_coeffs =
      gr_firdes.low_pass (
        1.0,          # erősítés
        quad_rate,    # mintavételi gyakoriság
        audio_rate/2 - width_of_transition_band,
        width_of_transition_band,
        gr_firdes.WIN_HAMMING)

    # bemenet: float; kimenet: short
    audio_filter =
      GrFIRfilterFSF (RFIR_decimate,
                      audio_coeffs)

    final_sink = GrAudioSinkS ()

    fg = gr_FlowGraph ()

    fg.connect (src, chan_filter)
    fg.connect (chan_filter, fm_demod)
    fg.connect (fm_demod, audio_filter)
    fg.connect (audio_filter, final_sink)

    return fg

if __name__ == '__main__':

    # csatlakozás az RF előtaghoz
    rf_front_end = microtune_eval_board ()
    if not rf_front_end.board_present_p ():
        raise IOError, 'RF előtag nem található'

    # az erősítés és a rádióállomás frekvenciájá-
    # nak beállítása
    rf_front_end.set_AGC (300)
    rf_front_end.set_RF_freq (100.1e6)

    IF_freq = rf_front_end.get_output_freq ()
    fg = build_graph (IF_freq)
    fg.start ()     # szál(ak) létrehozása
                    # fork hívásokkal és
                    # visszatérés
    raw_input ('A kilépéshez nyomd le az Enter
              gombot: ')
    fg.stop ()
```

tó logikai tömböt (FPGA) és egy programozható USB 2.0 vezérlőt tartalmaz. Egy teljes kiépítésű USRP alaplap négy bővítőkártya használatát teszi lehetővé: kettő vevő- és kettő adókártyáét. Az RF előtagok a bővítőkártyákon találhatók. A különféle frekvenciasávok kezelésére számos bővítőkártya létezik. Amatőr rádiósok számára a kis energiájú bővítőkártyák a legmegfelelőbbek, ezek a 440 MHz-es és az 1,24 GHz-es sávban adnak-vesznek. Létezik csak vételre használható, kábelmodemes vevőegységre épített bővítőkártya is, amely az 50 MHz – 800 MHz tartományt kezeli. A bővítőkártyákat úgy tervezték, hogy egyénileg, kézzel is könnyen összeállíthatók, ezzel is segítik az érdeklődőket a kísérletezésben.

Az USRP rugalmassága egyrészt a két programozható elemből, másrészt a gazdagépen futó könyvtárral folytatott párbeszédből fakad. Ha ízelítőt akarunk az USRP képességeiből, elég megvizsgálnunk indításának folyamatát. Maga az USRP nem rendelkezik ROM-ba írt belső programmal, csu-

pán néhány bájtnyi gyártó- (VID) és termékazonosítóval (PID), valamint változatszámmal. Amikor az USRP-t először csatlakoztatjuk az USB-kapura, a gazdagépen futó könyvtár egy beállítások nélküli készüléket lát – a beállítások hiányát a VID, a PID és a változatszám kiolvasásával észleli.
A könyvtár első lépésként a 8051 kódot tölti le, ez határozza meg az USB-vezérlő viselkedését. Amikor ez a kód elkezd futni, az USRP leválik az USB-sínről, majd újracsatlakozik hozzá. Az újracsatlakozás után a gazdagép már egy másik eszközt lát, a VID, a PID és a változatszám ugyanis megváltozik. A most már meglévő és futó belső program megadja az USB végpontokat, felületeket és parancskezelőket. Az USB-vezérlőnek kiadható parancsok egyike az FPGA feltöltése. A könyvtárkód, miután érzékelte az USRP új eszközként való újracsatlakozását, a beállítási folyamat következő fázisára lép, és áttölti az FPGA beállítására szolgáló bitfolyamot.
Az FPGA-k általános célú lapkák, működésüket a beléjük írt beállító bitfolyam határozza meg. A bitfolyamot leginkább objektumkódként, a kívánt működés magas szintű leírásának fordítási kimeneteként képzelhetjük el. Esetünkben a kívánt működést a Verilog hardverleíró nyelven fogalmaztuk meg. Ez a kód is forrásként, a GNU Radio többi részéhez hasonlóan GNU GPL alatt érhető el.

Mi történik az FPGA-ban?

Az FPGA egy kisméretű, erősen párhuzamos számítógépként fogható fel, aminek működését az adott feladat alapján mi tervezhetjük meg. Programozása igényel némi tapasztalatot, ugyanis ha elrontjuk, akkor véglegesen használhatatlanná tehetjük a lapkát. Éppen ezért tettünk közzé egy általános beállítást, amely alkalmazások széles köréhez használható.
Egy jó USB-gazdavezérlővel az USRP folyamatos 32 MB/másodperces átvitelre képes az USB-sínen keresztül. Az USB-sín váltakozó kétirányú átvitelre használható, a 32 MB/másodperc kapacitást szükség szerint oszthatjuk el a vételi és az adási irány között. A normál beállítás lehetővé teszi, hogy a vételi irányban kiválasszuk a digitalizált spektrum minket érdeklő részét vagy részeit, majd ezeket szükség szerint alapsávú jellé alakítsuk vagy tizedeljük. Mindez egyenértékű azzal, ami az RF előtagnál történik, leszámítva, hogy ebben az esetben digitalizált mintákkal dolgozunk. Azt a kódrészletet, amely ezt a feladatot látja el, digitális lefelé transzponáló elemnek nevezzük. (*3. ábra*) Mindezt digitális tartományban végezzük el, aminek az előnye, hogy azonnal meg tudjuk változtatni a középfrekvenciát, ami elsősorban frekvenciaugrásos szórt spektrumú rendszereknél hasznos. Adási irányban ennek pontosan az ellenkezője történik. Az FPGA több digitális lefelé és felfelé transzponáló elemet tartalmaz, ezek szükség szerint azonos és eltérő ADC-khez is csatlakoztathatók. A mögöttük álló elméletet itt nem áll módunkban kifejteni, a GNU Radio Wikiben azonban minden szükséges tudnivaló megtalálható.

GNU Radio alkalmazások

A már említett példákon túl a GNU Radio részeként teljes értékű HDTV adóhoz és vevőhöz, spektrumelemzőhöz, oszcilloszkóphoz, párhuzamos többcsatornás vevőhöz és persze modulátorok és demodulátorok folyton bővülő gyűjteményéhez is hozzájutunk.

A jelenleg tervezés vagy megvalósítás alatt álló rendszerek közül néhány példa:

- A TiVoval egyenértékű rádiós megoldás, több csatorna egyidejű rögzítésének lehetőségével
- Passzív radarrendszer, amely a TV-adásokat használja jelforrásként.
  Akik antennán keresztül nézik a tévéadásokat, csak gondoljanak arra, hogy ha repülőgép halad el felettük, ugrálni kezd a kép.
- Rádiócsillagászat
- TETRA adó-vevő
- Digitális világrádió
- Program alapú GPS
- Elosztott érzékelőhálózatok
- Spektrumhasználat elosztott mérése
- Amatőr rádió adó-vevők
- Alkalmi hálós topológiájú hálózatok
- RFID érzékelő/olvasó
- Több bemenetű több kimenetű (MIMO) jelfeldolgozás

Politikai kérdések

A szabadon elérhető, rádiók építésére alkalmas programok nem mindenkinek tetszenek. Az Amerikai Egyesült Államokban például az Amerikai Mozgókép Szövetség ellenállásába futottunk bele, ők a Broadcast Flag (másolásvédelmi kiegészítő jel a digitális adásokban) alkalmazásával próbálják korlátozni a földi sugárzású digitális tévéadásokhoz épített vevőkészülékek körét.
Az Amerikai Szövetségi Kommunikációs Bizottság egy előzetes állásfoglalást adott ki az kognitív képességekkel rendelkező rádiós megoldásokra és a program alapú rádiókra vonatkozóan.
Az állásfoglalásban számos problémát vetnek fel, felmerül többek közt a nagysebességű digitális-analóg átalakítók értékesítésének korlátozása, az engedély nélküli programok távol tartása a program alapú rádiózáshoz szükséges vasaktól digitális aláírások vagy hasonló eszközök segítségével és új korlátozások bevezetése az amatőrök számára gyártott rádiókra vonatkozóan.

Összegzés

A program alapú rádiózás érdekes terület, a GNU Radio minden eszközt rendelkezésünkre bocsát, hogy megkezdhessük felfedezését.
Aki komoly tudást akar szerezni a program alapú rádiózásban, annak több szakterületet is meg kell ismernie. Mi csak annyit ígérhetünk, hogy mindent megteszünk az első lépések megkönnyítése érdekében.

Linux Journal 2004. június, 122. szám

Eric Blossom a GNU Radio Project alapítója. Mielőtt program alapú rádiózással kezdett volna foglalkozni, évekig a biztonságos telefonszolgáltatások területén dolgozott. Amikor éppen nem program alapú rádiót bütyköl, valószínűleg jógázással vagy ju-jitsuval tölti az idejét. Az eb@comsec.com címen érhető el.