



ÓBUDAI EGYETEM

Alba Regia Műszaki Kar
Mérnöki Intézet

SZAKDOLGOZAT

DIGITÁLIS MŰSOR LOKÁLIS HÁLÓZATON TÖRTÉNŐ STREAMELÉSE FPGA HASZNÁLATÁVAL

OE-AMK
2014

Megyeri Zoltán
T000393/FI12904/K-S

Tartalomjegyzék

1. Bevezetés	3
1.1. A DVB-T Magyarországon.....	5
1.2. A DVB-T adás szerkezete.....	6
1.2.1. PES – Packetized Elementary Stream.....	7
1.2.2. TS – Az MPEG-Transport Stream.....	8
1.2.3. PSI táblák.....	9
1.2.4. SI táblák.....	10
1.2.5 A Transport Stream IP környezetben.....	12
1.3 Az FPGA általános ismertetése.....	14
1.3.1 Az FPGA felépítése.....	15
1.3.2 A VHDL nyelv.....	20
1.4 Felhasznált eszközök bemutatása.....	21
1.4.1 CW-4875 OFDM Demodulator Quad.....	21
1.4.2 CW-4900 Ethernet Controller.....	22
2. Az alkalmazott összeállítás	24
2.1 IPTV hálózat konfigurálása	25
2.1.1 SW-4800 - CW-Net Principal 1.03.....	25
2.1.2 SW-4875 – OFDM Demodulator Quad Controller	27
2.1.3 SW-4901 – Gigabit Ethernet Controller 1.01.....	27
2.1.4 SW-4841 – ASI to IP and IP to ASI Converter	29
2.1.5 Mérés a SW-4811B – TS Analyzer használatával.....	31
2.1.6 Megjelenítés és mérés a VLC médialejátszó használatával	32
2.2 A csatorna szűrés megvalósítása	33
2.2.1 A csatorna szűrés folyamatának szimulációja	36
2.2.2 A csatorna szűrés gyakorlati megvalósítása	39
2.2.3 A csatorna szűrés mérése az SW-4811B és a VLC használatával.....	41
3. Összefoglalás	43
4. Summary.....	45
Mellékletek	47
Irodalomjegyzék.....	54

1. Bevezetés

Az IPTV (*Internet Protokoll Televízió*) mozaikszó hallatán a legtöbb embernek, a telekommunikációs szolgáltatók által kínált úgynevezett "csomagok" egyike jut először eszébe, amely az ADSL hálózaton keresztül háztartásokba juttatott televízió csatornák vétele mellett, különböző extra lehetőségeket nyújt az előfizetőknek. Ilyen plusz lehetőség például a megállítható adás, melynek megtekintését bármikor folytathatjuk onnan, ahol abbahagytuk azt. Másik ilyen szolgáltatás egy bizonyos műsor rögzítését, valamint annak egy másik időpontban történő megtekintését teszi lehetővé. Mindezek mellett akár kedvenc filmünket is kikölcsönözhetjük az IPTV videotékából.

Ha úgy döntünk, hogy egy ilyen ajánlattal szeretnénk élni, e kényelmi funkciókat élvezve csupán annyi változást észlelhetünk otthonunkban, hogy televíziónk mellett egy - a manapság sokszor hallott - set-top-box nevű készüléknek is helyet kell szorítanunk.

Ha viszont úgy döntünk, hogy inkább a magyarországi digitális átállást követően ma már igen jó lefedettséget biztosító földfelszíni sugárzás által kínált médiaszolgáltatásokat szeretnénk igénybe venni, viszont televíziónk még nem rendelkezik a direkt vételhez szükséges műszaki paraméterekkel, ebben az esetben is helyet kell hagynunk polcunkon a set-top-box számára, hogy élvezhessük az ingyenes vagy fizetős televízió- és rádióadásokat.

A médiatartalmat közvetítő szolgáltatók az előfizetői elégedettség megtartása érdekében olyan digitális eszközök használatára kényszerülnek, mely működési sebességükből adódóan nem korlátozzák a műsorok megjelenésének minőségét. Ezért a valós idejű tartalmak vételezése és megjelenítése miatt, mindkét fent említett vételi lehetőség megvalósításához igen gyors működésű áramkörrel kialakított eszközre van szükségünk.

Erre a célra tökéletes megoldás, a manapság egyre szélesebb körben népszerű FPGA hardverelem, melynek nagy számítási teljesítménye mellett, párhuzamos működésének köszönhetően egyre különbözőbb területeken kerül felhasználásra. Elsősorban e tulajdonságainak köszönhetően, használata a digitális televíziózásban is kézenfekvő, éppen ezért a set-top-box-ok nyomtatott áramköri kártyáin is ez az eszköz található meg.

Szakdolgozatom célja, hogy a magyarországi földfelszíni műsorszórás ingyenesen vételezhető műsorait elérhetővé tegyem egy helyi számítógép hálózaton, valamint az hogy megvalósítsam azt a lehetőséget, hogy egy dedikált vezérlő számítógép segítségével meghatározható legyen, hogy éppen mely csatorna legyen elérhető a hálózatra felcsatlakozott felhasználók számára.

Mindezt a CableWorld Kft. által intézményünk rendelkezésére bocsátott eszközei használatával valósítom meg, melyek mindegyike tartalmaz FPGA eszközt, lehetőséget adva a fent említett két, műsor-vételi lehetőség ötvöztetésének léterhozására, hogy lokális hálózaton, set-top-box nélkül legyen az ingyenes adások egyike elérhető. Ezen eszközök használatával egyidejűleg az adás vétele, valamint annak helyi hálózatba történő táplálása is kivitelezhető.

Szakdolgozatom első fejezetében, rövid bevezető után bemutatom, hogyan épül fel egy digitálisan sugárzott adás szerkezete - mely elengedhetetlen a csatorna kiválasztás szempontjából – és, hogy hogyan néz ki egy ilyen digitális adás IP hálózati környezetben.

Az alkalmazott eszközök bemutatása és a rendszer összeállításának ismertetőjét követően, a felhasznált készülékeket konfigurálom fel úgy, hogy az összes ingyenesen elérhető műsor a hálózatról egy időben vételezhető legyen. Az összeállítás megvalósítása után mérést végzek a továbbított adatfolyam összetételével és adatsebességével kapcsolatban.

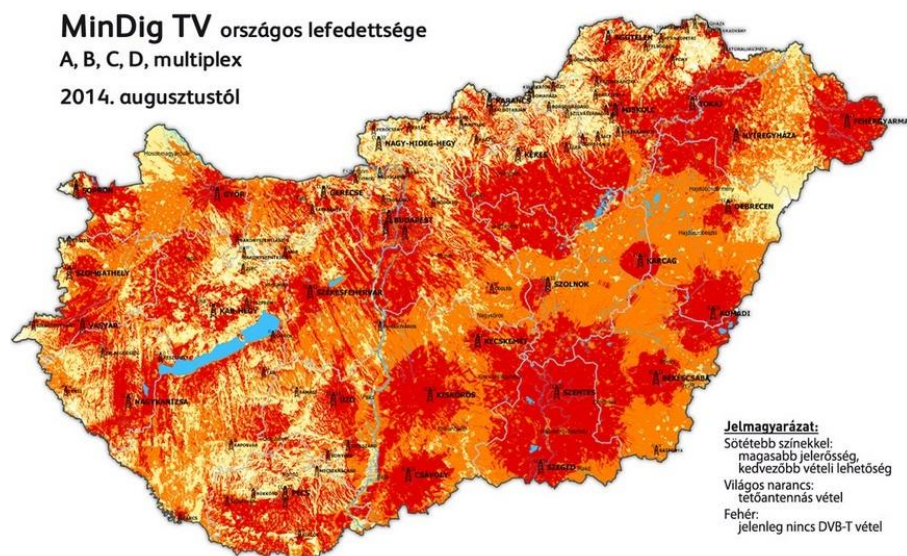
Ezt követően az felhasznált készülékek egyikén található FPGA áramkör megfelelő felprogramozásával egy szűrést valósítok meg, mellyel egy időben kizárólag egy csatorna elérhetőségét teszem lehetővé a hálózaton. Ennek megvalósítását, az FPGA programozását lehetővé tévő nyelvek közül, az alacsonyszintű VHDL programnyelv használatával végzem el. A feladat elvégzése után szintén egy mérés elvégzésével vizsgálom meg, hogy valóban egy csatorna érhető-e el a hálózaton, valamint azt, hogy az milyen adatsebességgel bír.

1.1. A DVB-T Magyarországon

A digitális műsorszórás elterjedésének első mérföldköve a DVB (*Digital Video Broadcasting*) projekt megalakulása volt 1993-ban, amely különböző szabványok kidolgozásával koordinálta az európai országok analóg rendszerről történő átállását. E szabványcsalád tartalmazza a digitális fölfelületi műsorszórásra vonatkozó DVB-T (*Digital Video Broadcasting – Terrestrial*) rendszer műszaki paramétereit is.

Magyarországon az átállást megelőző munkálatok két jelentősebb ütemben zajlottak le. Az első kísérleti adás sugárzását 1999-ben, az Antenna Hungária kezdte meg Budapesten, majd később a Kab-hegyi adó is becsatlakozott, elősegítve a vételi lehetőségek vizsgálatát.

A második szakasz 2008-ban kezdődött, mikor a Nemzeti Hírközlési Hatóság pályázatot írt ki a belföldi digitális televízió és –rádió-hálózat üzemeltetésére, melyeket az Antenna Hungária nyert el. Ekkor indult el a MinDigTV elnevezésű szolgáltatás, mely az ingyenesen fogható csatornák vételét biztosítja, valamint a MinDigTV Extra, amely előfizetés ellenében teszi lehetővé a további tartalmak elérését. E két szolgáltatás a hálózat folyamatos fejlesztésének, így az országos lefedettség növekedésének köszönhetően egyre több előfizetőt tudhat magáénak. Az analóg hálózat teljes üzemén kívül helyezése két ütemben zajlott és 2013. október 31-re valósult meg teljesen, miközben a DVB-T lefedettség elérte a 99%-ot (1. ábra). [1]



1. A DVB-T lefedettsége Magyarországon

1.2. A DVB-T adás szerkezete

A stúdiókban használt rögzítő eszközök közvetlen kimenetén keletkező, nagy adatsebességű videó jelek (SDTV esetén 270 Mbit/s) és audio jelek (CD minőségű, digitális stereo esetén 1,5 Mb/s) közvetlen továbbítása gazdasági szempontból nem lenne kifizetődő, ezért az adatsebességük csökkentését tömörítési eljárások alkalmazásával érik el, melyek valamely MPEG (*Moving Pictures Experts Group*) szabvány alapján valósulnak meg. Hazánkban a videojelek esetén jelenleg az MPEG-4 - 2003-ban megjelent - H.264 formátumú kódolását használják, amellyel - HD tartalom esetén a minőség észrevehető romlása nélkül - fele akkora adatsebesség érhető el a kimeneten, lehetővé téve ezzel azt, hogy ne csak SD minőségű adások legyenek elérhetőek az adott szolgáltatásban.

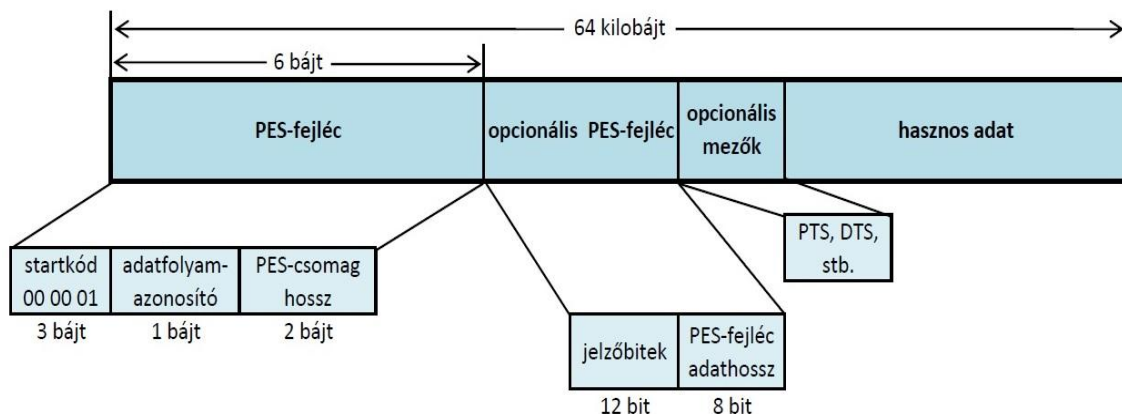
A digitális műsorszórásban a tömörített képek és hangjelek továbbítása mellett egyéb adatok továbbítására is sor kerül, melyek különböző kényelmi szolgáltatásokat nyújtanak a felhasználók számára - ilyen például a teletext -, ezek "egyszerű" adatként alkotnak egy adatfolyamot.

E jeleket külön-külön elemi adatfolyamoknak (*ES – Elementary Stream*) nevezzük, melyek tömörítés után közvetlenül adattartalmuk alapján változó méretű Csomagolt Elemi Adatfolyamokra (*PES - Packetized Elementary Stream*) kerülnek felosztásra. A létrejött PES-ek azonban továbbításra még nem alkalmasak, így ezek ismételt felosztásra kerülnek, most már egységes méretű TS csomagokra (*Transport Stream Packet*).

Az általunk továbbítani kívánt csatornák video és audio és egyéb TS csomagjai, az MPEG csoport egyik szabványában meghatározott úgynevezett 'MPEG – adatfolyamot', azaz Transport Stream-et alkotnak. A Transport Stream (TS) szerkezetét tekintve igen flexibilis, így előfordulhat olyan eset is, hogy az adatfolyam tartalma az adó és a vevő közötti csatornán változásokon megy keresztül. Ahhoz, hogy a vevő oldal pontos képet kapjon, hogy a TS éppen mely műsor csomagjait tartalmazza, úgynevezett táblákat is továbbítanunk kell majd, melyek nyilvántartják az adás összetevőit. [2]

1.2.1. PES – Packetized Elementary Stream

Az Elemi Adatfolyamok különböző méretű csomagokra bontása után a létrejött PES-csomagokat (2. ábra) adattartalmuk alapján különböztetjük meg, melyek lehetnek: video-PES, hang-PES és adat-PES.



2. PES csomag szerkezete

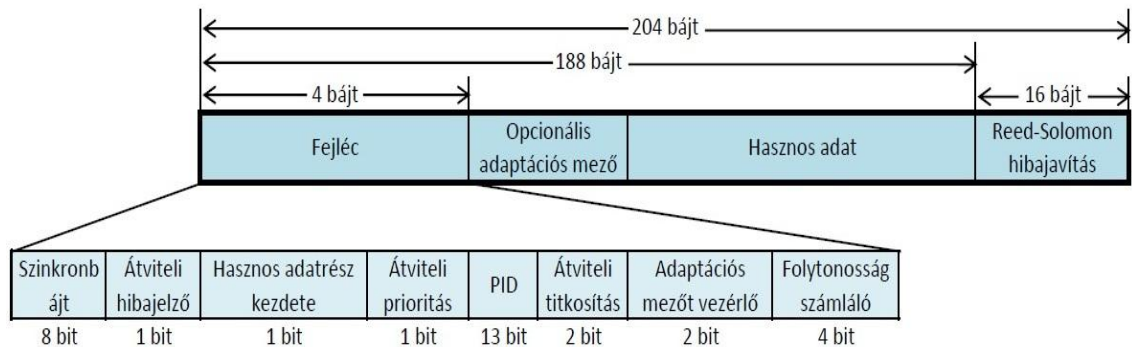
Egy csomag mérete maximum 64 kbyte lehet, de video-PES esetében előfordulhat olyan is, amikor ez az érték túllépésre kerül, ezt azonban a fejlécben jelezni kell, melynek mérete minimum 6 bájt.

Három részre oszthatjuk fel ezt az adatmezőt, melyek fix mérettel rendelkeznek. Az első 3 bájton a startkód található meg, a 00 00 01 bitsorozat négyszeres ismétlődése tölti a fejléc e részét. Ezt követi az adatfolyam-azonosító rész, ami 8 biten jelzi, hogy az elemi adatfolyam éppen video, hang vagy egyéb adatokat tartalmaz. A PES-fejléc utolsó két bájtján, az adott csomag hossza kerül jelzésre. A már említett csomagméret túllépése esetén, mindkét bájt a nulla értéket veszi fel.

A hasznos adatrész rovására, a csomag 'opcionális fejlécet' is tartalmaz, melyben 12 bites szakasz jelöli, hogy az opcionális mezők közül melyek találhatók meg ebben a szakaszban. Ezt követi egy 1 bájtos rész, amely a fejléc teljes adathosszát adja meg. Az opcionális mezők az elemi adatfolyamra és a PES-csomagra vonatkozó adatokat, valamint a megjelenítéshez szükséges segédadatokat tartalmazhatja. Ilyen segédadat a PTS (*Presentation Time Stamp*), amely a megjelenítés során a kép és a hang közötti szinkront, más néven ajakszinkront biztosító időzítési értékeket adja meg, és a DTS (*Decoding Time Stamp*), amely a tömörítés után továbbított-, megváltozott sorrendű képek helyes megjelenítését biztosítja. [2]

1.2.2. TS – Az MPEG-Transport Stream

A különböző PES-csomagokat a későbbi multiplexálhatóság és átvitel érdekében kisebb-, egységes méretű TS-csomagokra osztják fel (3. ábra).



3. TS csomag szerkezete

- A TS-csomagok mérete állandó, 188 bájt hosszú. Az első 4 bájtot a csomag fejléce alkotja, melynek első bájtja a 47_{hex} értékű szinkronbájt. Előfordulhat azonban, hogy a csomag hasznos adatrészében is megjelenik ez az érték, ezért a szinkronizálás a szinkronbájtok állandó távolságának (188 bájt) figyelésével kerülnek azonosításra.
- Ezt követi az átviteli hiba-jelző bit, melyet a vevő oldali demodulátor állít be akkor, ha a Reed-Solomon kódolással (mellyel maximum 8 hiba javítható) védett adat már nem helyreállítható.
- A hasznos adatrész kezdete 1 biten kerül jelzésre, melynek értéke akkor 1, ha új PES-csomag (a PES-fejléccel együtt)-, vagy egy új tábla továbbítása kezdődik.
- Az adott csomag prioritását egy szintén egy 1 bites rész jelzi, amely az ugyanolyan azonosítóval (PID-del) rendelkező csomagok között tesz különbséget.
- A TS-csomag hasznos adatrészben található tartalom azonosítására szolgál a PID (*Packet Identifier*) érték, ami egy 13 bites, hexadecimálisan megadott szám.
- Ha az átviteli titkosításjelző 2 bit közül valamelyik nem nulla, akkor a csomag titkosítva van, melynek feloldásához egy erre szolgáló tábla szükséges.
- A TS-fejléc kiterjesztésének jelenlétét az adaptációsmező-vezérlő 2 bit jelzi, ha ezek értéke nulla, akkor adaptációs mező nem foglal helyet a hasznos adatrészből.
- A folytonosság számláló a TS-csomagok fejlécében egy 4 biten történő belső számlálást végez 0-tól 15-ig, majd újraindul. Ha a számlálás nem folytonos, az adatfolyamban hiba keletkezett.

Az opcionális adaptációs mező hosszát a fejléct követő első 8 biten adhatjuk meg. Itt helyezkedik el a PCR (*Program Clock Referece*), ami egy 42 bites számláló, adó oldalon beültetett értékét tartalmazza. A számláló értékét az MPEG-kódoló 27 MHz-es órajelet szolgáltató oszcillátora lépteti, ez az úgynevezett STC (*System Time Clock*), rendszer-órajel. Túlsordulás esetén a számláló lenullázódik, ami kb. 26,5 óránként következik be, majd újraindul.

A vevő oldalon is szükség van az STC előállítására, melyhez ugyanúgy egy 27 MHz-es 42 bites számlálót léptető oszcillátorra van szükség. Működés során az adó és a vevő STC és PCR értékeinek azonosnak kell lenniük. Dekódolás alatt az MPEG-dekódoló kiolvassa a TS-be beültetett PCR értéket és összehasonlítja a saját számlálójának aktuális értékével. Abban az esetben, ha a két érték kis mértékben eltér egymástól, a probléma egy PLL-hurok alkalmazásával még korrigálható. Az adatfolyamnak az MPEG-szabvány alapján 40 ms-onként tartalmaznia kell PCR értékeket, melyekre nézve maximum ± 500 ns időelcsúszás, más néven "jitter" megengedett. [2]

1.2.3. PSI táblák

A Transport Stream már korábban említett rugalmas szerkezete miatt és ez által az, hogy több program adatai kerülnek továbbításra, szükségessé válik a programspecifikus információk (PSI – *Program Specific Information*) adatfolyamba való beültetése, melyek a következők:

- PAT (Program Association Table) – Program-hozzárendelési tábla
- PMT (Program Map Table) – Programleképezési tábla
- CAT (Conditional Access Table) – Feltételes hozzáférési tábla

A PAT táblák állandó 0000_h PID értékkel, valami szintén 00_h táblaazonosítóval rendelkeznek. Egy Transport Stream, egyetlen ilyen tartalmaz, melynek jelenlétét 500ms-onként biztosítani kell, ellenkező esetben a vevő oldal a beérkező adatokat nem képes azonosítani. A tábla Hasznos adat része, a különböző műsorok PMT tábláinak PID azonosítóit tartalmazza.

A PMT táblák egyedi PID azonosítókkal bírnak, melyek a DVB-szabvány szerint 0020_h és 1FFE_h közé kell, hogy essenek, táblaazonosítóik viszont 02_h értékűek.

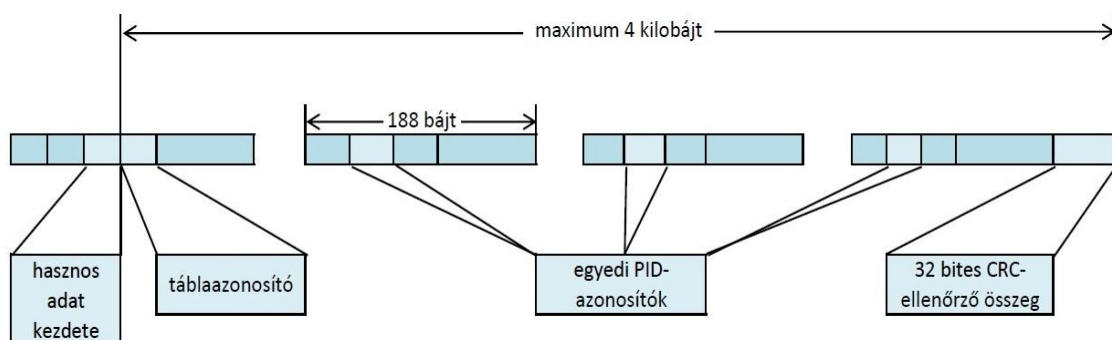
Ismétlődésük az adatfolyamban szintén 500ms-onként szükséges. A Hasznos adat részben egyetlen program elemi adatfolyamainak PID értékeit tartalmazza, melyek lehetnek video, hang teletext és PCR. Előfordulhat olyan eset, amikor egy adás több nyelven kerül közvetítésre, ekkor az ehhez tartozó PMT tábla az összes ehhez tartozó hang-ES azonosítóját tartalmazza. A PCR PID esetében, ha az a video-ES-be ültetve kerül a transport stream-be, annak értékével fog megegyezni.

Ha egy vagy több program elérhetőségét korlátozni szeretnénk, például, hogy csak fizetős szolgáltatás keretében váljon elérhetővé (Magyarországon ilyen a MinDigTV Extra szolgáltatás), akkor az adatfolyamot valamilyen eljárással titkosítanunk kell. Ahhoz, hogy a vételi oldalon a megfelelő jogosultságok mellett a titkosítás feloldhatóvá váljon, az ehhez szükséges adatokat a CAT táblában kell tárolnunk, melynek PID értéke 0001_h táblaazonosítója pedig 01_h. Ebben található az ECM és az EMM, melynek azonosítóit tartalmazza. Az ECM (*Entitlement Control Message*) üzenet, a titkosítás során használt kódokat tartalmazza és az EMM (*Entitlement Management Message*) üzenet, a vevő oldalra vonatkozó jogosultságokra vonatkozik. E két üzenet és a felhasználó által birtokolt kód együttese oldja fel a titkosítást és teszi elérhetővé a programot. [2]

1.2.4. SI táblák

Az MPEG-szabvány lehetővé teszi, hogy a TS-ben "privát táblákat" helyezünk el, az ezekre vonatkozó betartandó szerkezeti szabályok a következők (4 ábra):

- a táblákat egyedi PID azonosítóval kell ellátni
- a táblákat táblaazonosítóval kell ellátni, mely a hasznos adatrész első bájta
- a táblák tartalmát a hasznos adatrészben kell továbbítani
- a táblát legfeljebb 4 kilobájtos szakaszokra kell osztani
- a szakaszokat 32 bites CRC-ellenőrző összeggel kell zárni



4. Táblák szerkezeti felépítése

A "privát táblák" létrehozásának lehetőségét kihasználva a DVB-projekt további táblákat definiált a vevő oldali készülékek működésének megkönnyítése érdekében, melyek szolgáltatási információkat (*SI – Service Information*) tartalmaznak.

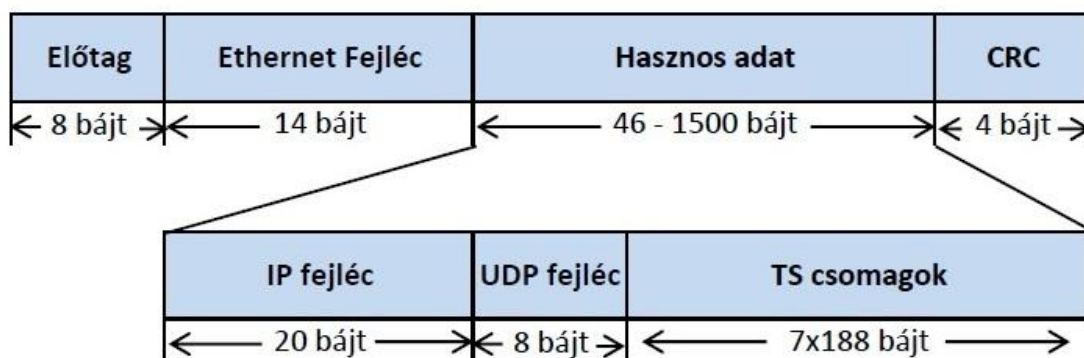
- Hálózati információs tábla (*NIT – Network Information Table*)
Tartalmazza az átviteli út fizikai paramétereit: a vételi frekvenciát, a hibavédelmet, a moduláció és az átvitel a típusát (DVB-S, DVB-C, DVB-T). A vevőkészülék ezen adatok eltárolásával, felgyorsítja a csatornaváltás folyamatát.
- Szolgáltatásleíró tábla (*SDT – Service Description Table*)
Az adatfolyam által tartalmazott programokkal kapcsolatos információkat és azok neveit és tartalmazza szöveges formában.
- Programcsokor-hozzárendelési tábla (*BAT – Bouquet Association Table*)
Ez a tábla az SDT-val megegyező PID értékkel rendelkezik, táblaazonosítójuk azonban más. Míg az SDT egy-, a BAT több fizikai csatorna programszerkezetét írja le együttesen.
- Eseményinformációs tábla (*EIT – Event Information Table*)
Az "elektronikus programkalauz" (*EPG –Electronic Program Guide*) szolgáltatás biztosítása esetén, ennek adatai az EIT-ban kerülnek továbbításra, amely a programok kezdési- és befejezési időpontját tartalmazza egy bizonyos időtartammal előre vetítve.
- Futási állapot tábla (*RST – Running Status Table*)
Ha egy műsort automatikusan, időzítéssel megoldva szeretnénk rögzíteni, de az adott program a tervezett kezdéshez képest csúszik, ebben a táblában érkezik az a vezérlőjel, amely hatására a felvétel elindul.
- Idő és dátum tábla (*TDT – Time and Date Table*)
Ebben a táblában a pontos greenwichi idő és dátum kerül továbbításra.
- Időeltolódási tábla (*TOT – Time Offset Table*)
Tartózkodási helytől függően, az aktuális időzónához tartozó időeltolást tartalmazza, melyet a TDT táblával együttesen kezel a vevőkészülék.
- Kitöltő tábla (*ST – Stuffing Table*)
Az adatfolyam tábláinak tartalmát vonhatjuk vissza, írhatjuk felül a segítségével.

1.2.5 A Transport Stream IP környezetben

A TS Packetek helyi hálózaton történő továbbítása Ethernet adatcsomagokban történik, e csomagok küldése esetén több alapfeltételnek kell teljesülnie:

- méretük nem lehet kisebb 64 bájtánál
- a csomagok között 12 bájtos szünetnek kell lenni

A csomag 14 bájtos fejlécében a csomagot fogadó és az azt küldő eszközök hálózati kártyáját azonosító MAC (*Media Access Control*) címe helyezkedik el. Ezt követi a hasznos adatot tartalmazó rész, melynek méretét tekintve – a minimum megengedett 64 bájtól levonva a 14 bájtos Ethernet fejléct és a csomag végén található 4 bájtos ellenőrző összeget, a CRC-t (*Cyclic Redundary Check*) – 46 bájtának kell lennie. Egy Ethernet csomag adattartalma maximum 1500 bájt lehet.



5. A Transport Stream Ethernet keretben

Az UDP/IP csomag fejléce az Ethernet csomag hasznos adattartalom részében helyezkedik el. Először a 20 bájtos IP (*Internet Protocol*) fejléc, majd ezt követően a 8 bájtos UDP (*User Datagram Protocol*) fejléc.

Az IP fejléc tartalmazza a küldő- és a fogadó eszközök IP címét, valamint, hogy az átvitel mely protokoll használatával történik (esetünkben az UDP protokoll használatos).

A Transport Stream küldésére, az audio- és a vizuális adatok valós időben történő megjelenítése miatt, az UDP protokoll használata a legelőnyösebb. Az UDP egy úgynevezett összeköttetés nélküli protokoll, amely a forrás- és a célalkalmazások között csupán adatokat továbbít. E fejléce tartalmazza a forrás- és a cél port számot, amelyre felcsatlakozva tudják fogadni az alkalmazások, az adott IP címre küldött adatcsomagok közül a számukra hasznosat.

A maximálisan 1500 bájtos UDP/IP csomag méretéből levonva e két fejléc hosszúságát, már csak 1472 bájtot van lehetőségünk továbbítani. Ezt az értéket elosztva a TS csomagok fix méretével (188 bájt), megkapjuk, hogy 7 teljes Transport Stream csomagot tudunk egyszerre küldeni (5. ábra).

Az adatcsomagok hálózaton történő továbbítása három különböző küldési módban történhet.

- Unicast átvitel:

Unicast küldési módról beszélünk, ha az üzenetküldés a kizárólag egy, adatot küldő és egy, adatot fogadó gép között történik.

- Broadcast átvitel:

Más néven adatszórásos átvitel, amely lehetővé teszi, hogy a küldendő adatcsomag egy időben, minden egyes hoszthoz eljusson. Hátránya azonban, hogy ez a folyamat úgy hajtódik végre, hogy az adatot küldő alkalmazás hálózatra felcsatlakozott minden egyes hoszt számára külön elküldi a csomagot, amely a sávszélesség-takarékosság szempontjából igen kedvezőtlen.

- Multicast átvitel:

Más néven többes küldés, mely abban különbözik az előbb említett adatszórásos átviteltől, hogy egy forrás felől ugyan több-, azonban nem minden cím irányába történik az adatküldés. Ez úgy valósul meg, hogy az adatokat vételezni kívánó hosztok egy úgynevezett multicast csoporthoz csatlakoznak. Minden ilyen csoportnak önálló IP címe van, amely a 224.0.0.0 – 239.255.255.255 tartományba esik. A hosztok az IGMP (*Internet Group Management Protocol*) protokollt használva, IGMP üzenetekkel csatlakozhatnak a csoporthoz - és vételezhetik az így továbbított adatfolyamot -, vagy hagyhatják el azt. [3] [4]

1.3 Az FPGA általános ismertetése



Az FPGA (*Field Programmable Gate Array*), olyan programozható kaputömb, melynek működése a felhasználó által szabadon meghatározható.

Beágyazott rendszer tervezésekor több különböző áramkörü alkotó elem működését kell összehangolni akár hardveres, akár szoftveres szempontból, hogy a tervezett eszköz megfelelően

kielégítse az elvárt igényeket. Ezen alkotóelemek együttes működésének meghatározása azonban egyetlen FPGA áramkör segítségével is történhet. Már a fejlesztés során élvezhető az FPGA újrakonfigurálhatóságának előnye, amely a meglévő paraméterek módosítását vagy akár a felépítés kibővítését teszi lehetővé. A különböző feladattal ellátott alkotóelemek működése egymástól független, mivel az eszköz működését tekintve párhuzamos, ami megköveteli a magas számítási teljesítményt is.

Az FPGA programozását tekintve három különböző lehetőség áll a felhasználók számára. Az egyik ilyen lehetőség a grafikus alapon történő tervezés, amely az áramkörü elemek sematikus rajzjeleinek használatával és azok helyes összekötésével történik. E megoldás hátránya, hogy vezetékezkor fenn áll a téves bekötés esélye.

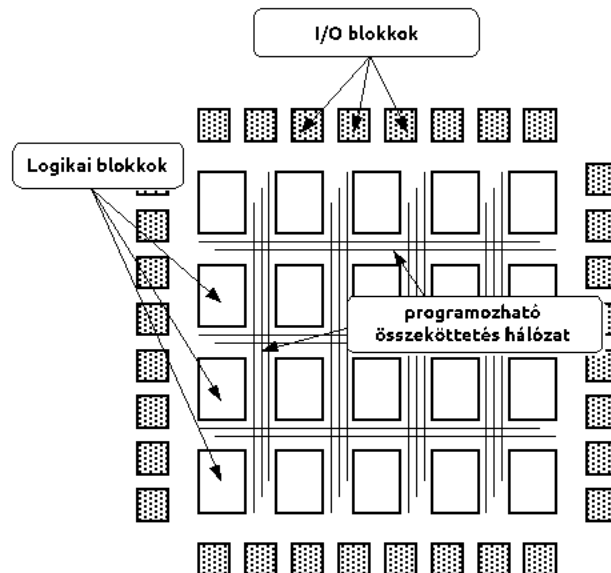
A másik két eset a VHDL és a Verilog alacsony szintű nyelvek használata, melyek hardver leíró nyelvek és megadhatjuk velük egyúttal a működést és a vezetékekezést is.

Az FPGA alkalmazásának területei például: jelfeldolgozás, orvosi alkalmazások, tudományos számítások, vezetékes és vezeték nélküli kommunikáció, műsorszórás. [5]

1.3.1 Az FPGA felépítése

A Xilinx által gyártott FPGA-k három fő részből állnak (6. ábra):

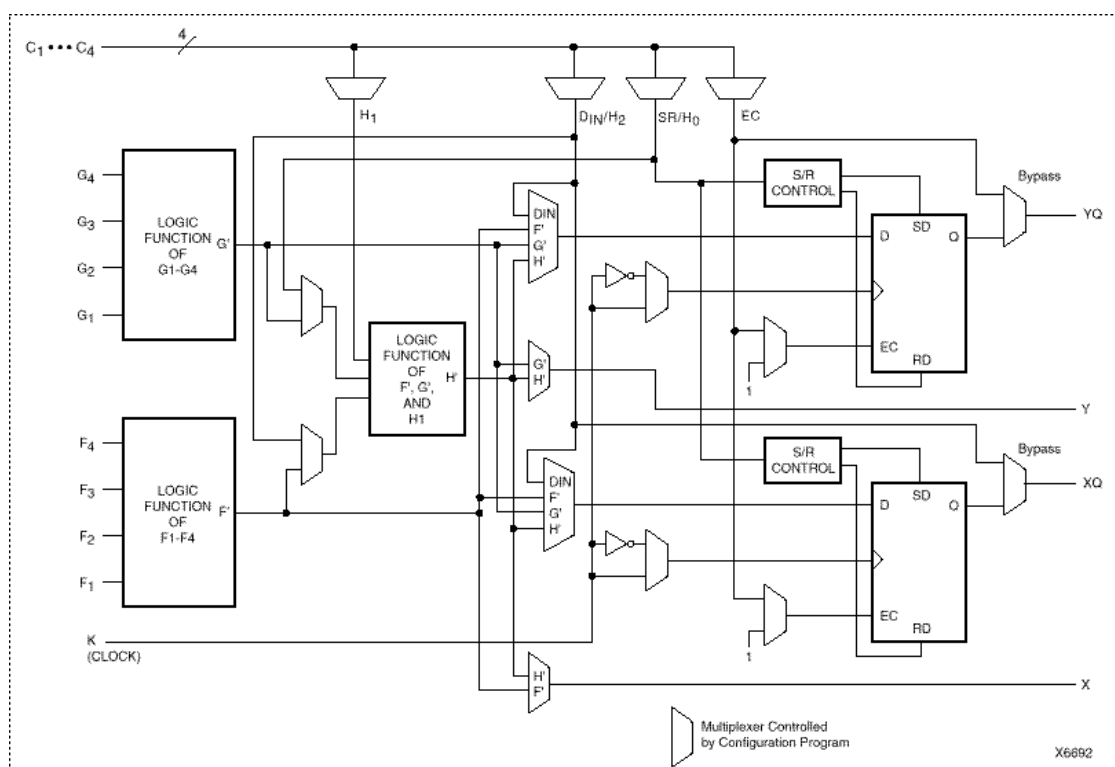
- Konfigurálható logikai blokkok (CLBs): a felhasználók számára szükséges logikai kapcsolatok megvalósítását végzik
- Ki/bemeneti blokkok (IOBs): a belső vonalak és a tok kivezetései között hozzák létre a kapcsolatot
- Programozható kötések: segítségével tudjuk behuzalozni a CLB-k közötti összeköttetéseket, valamint a CLB-k és az IOB-k közötti kapcsolatokat. [6]



6. Az FPGA belső felépítése

A konfigurálható logikai blokkok (CLB)

A konfigurálható logikai blokkok, a felhasználó által meghatározott működést valósítják meg. Felépítésük a 7. ábrán látható. Belsejükben F és G elnevezéssel, két darab négy bemenetű funkció generátor található meg, mivel a legtöbb kombinációs logikai elem négy, vagy annál kevesebb bemenetet használ. A CLB tartalmaz még mindemellett egy harmadik (H) funkció generátort is, aminek három bemeneti vonala van. F és/vagy G blokkok kimenetei közül, egyik se, az egyik, vagy mindkettő használható bemenetként, a H1 bemenet pedig a CLB-n kívülről érkezik. A 7. ábrán trapéz alakkal ábrázolt multiplexerek programozhatók, ezzel is növelve a nagyfokú flexibilitást.



7. A CLB felépítése

Minden CLB tartalmaz két tároló elemet, melyeket felhasználhatunk a funkció generátor kimeneteinek tárolására. Ezek a tároló elemek a funkció generátortól függetlenül is használhatóak, ebben az esetben a tároló elemek flip-flop-ként, vagy latch-ként működhetnek. A DIN direkt bemeneti jel is használható ezekhez a tároló egységekhez, a H1 viszont csak a H funkciós generátoron keresztül. A funkciós generátorok a tároló egységektől függetlenül is képesek meghajtani kimeneteket, mindez által rugalmasabbá, és egyszerűbben vezetékezzetűvé válik a rendszer. A tizenhárom CLB bemenet és négy CLB kimenet hozzáférést biztosít tehát a funkció generátorokhoz és a tárolókhoz. Ezek a ki-, és bementek csatlakoznak a programozható összeköttetésekre. [6]

Programozható belső összeköttetések

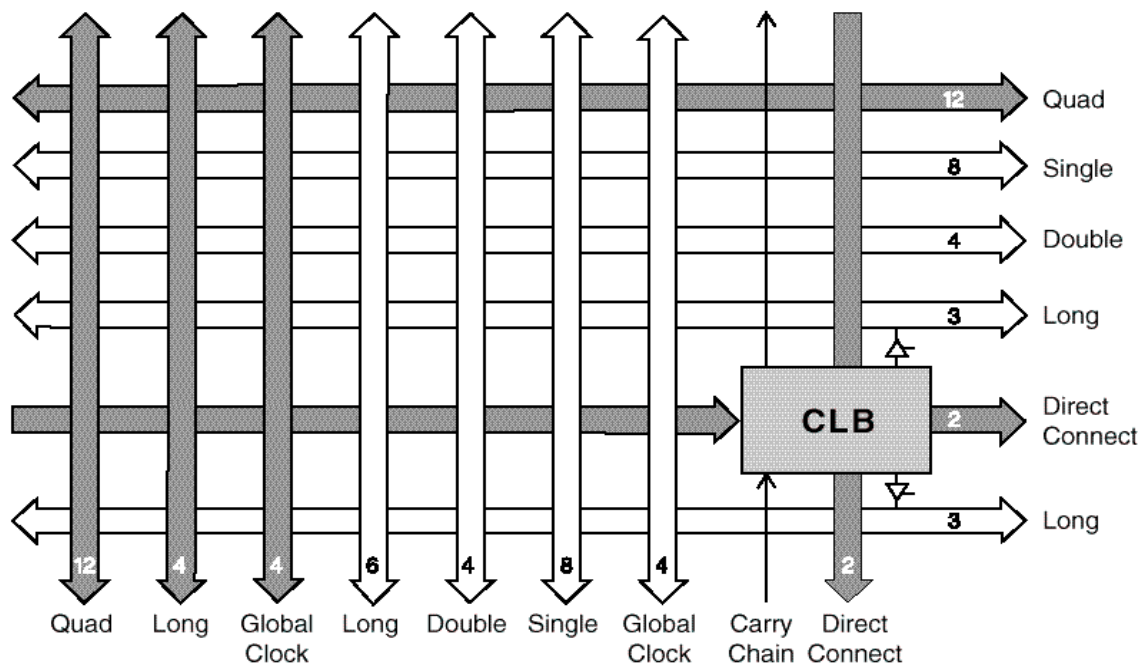
A belső összeköttetések programozható kapcsolási pontokból és kapcsolási mátrixokkal vannak megvalósítva, hogy a kívánt tervnek megfelelően köthessük össze az egységeket. Egy strukturált, hierarchikus mátrix áll rendelkezésünkre, amivel hatékonyan valósíthatjuk meg ezeket az összeköttetéseket.

Többféle összeköttetés típus létezik:

- A CLB vezetékek a sorokat és oszlopokat kötik össze a CLB tömbben.
- Az IOB vezetékek egy gyűrűt alkotnak a CLB blokkok tömbje körül, valamint azok összeköttetését valósítják meg a ki-, bemeneti blokkokkal.
- Léteznek még úgynevezett globális vezetékek is, amik arra hivatottak, hogy az órajelet biztosítsanak az eszköz belsejében (számolni kell minimális késleltetéssel).

A vezetékhalózat hossza és mérete alapján a következő típusokat különböztetünk meg:

- egyszeres (single) hosszúságú vonalak
- kétszeres (double) hosszúságú vonalak
- négyszeres (quad) hosszúságú vonalak
- nyolcszoros (octal) hosszúságú vonalak
- valamint hosszú vonalak

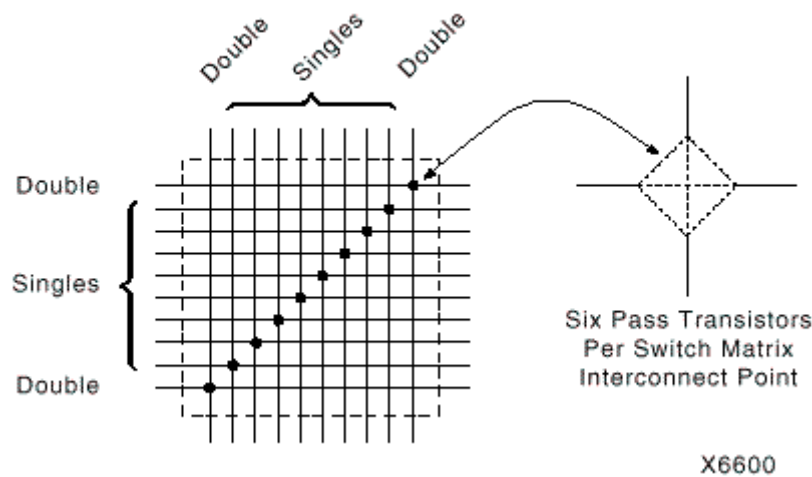


x5994

8. CLB összekötő útvonalak felépítése

A nyolcadik ábrán látható, hogy a CLB bemenetek és kimenetek egyaránt a blokk mind a négy oldalán találhatóak, segítve ezzel a rugalmas működést. Általánosságban elmondható, hogy a struktúra szimmetrikus és szabályos felépítésű. [7]

Azokat a csomópontokat, ahol vízszintes és függőleges, egyszeres és kétszeres vezetékek találkoznak, programozható kapcsoló mátrixnak (*PSM - Programmable Switch Matrix*) hívjuk. Ez látható a 9. ábrán. Minden egyes kapcsoló mátrix bizonyos irányba vezető tranzisztorokból állnak, amik a felprogramozástól függően összekötik a megfelelő vezetékeket vagy nem.

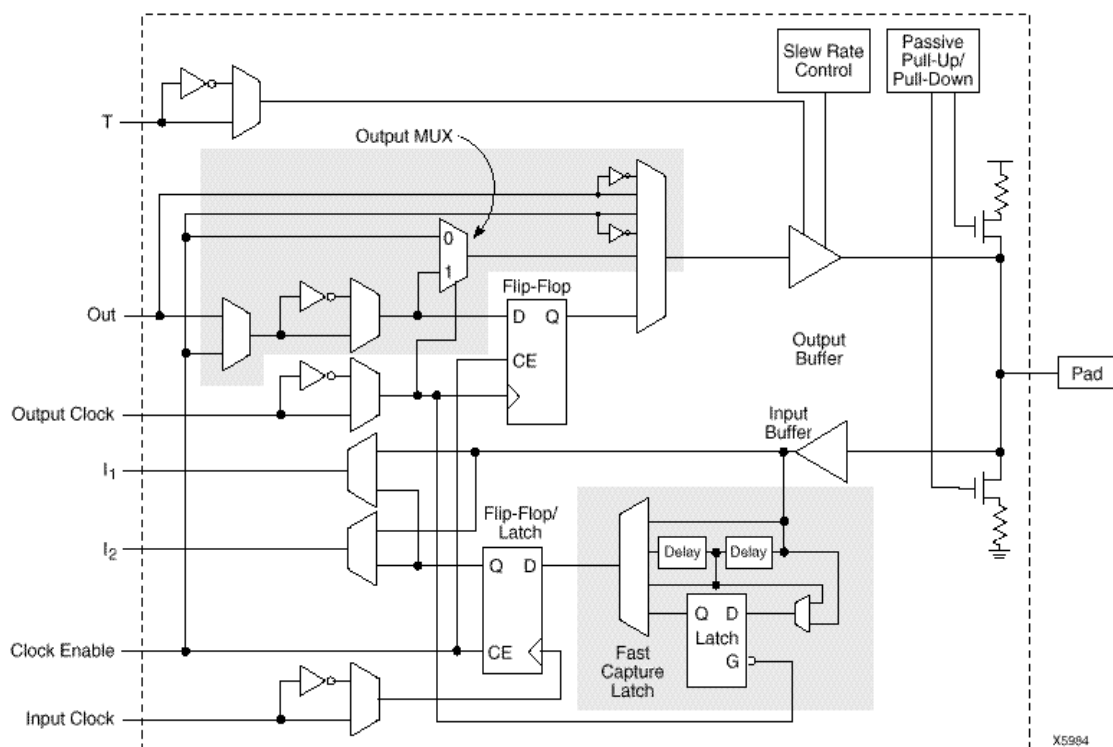


9. Programozható kapcsoló mátrix

Az egyszeres vonalak nyújtják a legnagyobb a flexibilitást, emellett gyors összeköttetést valósítanak meg a szomszédos blokkok között. Nyolc vízszintes és nyolc függőleges egyszeres vonal található minden CLB között. Ezek a vonalak kapcsoló mátrixokon keresztül kapcsolódnak egymáshoz, azonban minden egyes kapcsoló mátrixon való áthaladáskor késleltetéseket szenvednek el. Éppen ezért hosszú összeköttetésekre nem felelnek meg, helybeni összeköttetésekre és elágazásokra használjuk csak őket. Távolabbi összeköttetések megvalósítására a kétszeres vonalak használata optimálisabb. [7]

Bemeneti/kimeneti blokkok

A ki- és bemeneti blokkok biztosítják a felhasználó számára, hogy az FPGA belseje és a külső hardver között létrejöjjön a kapcsolat, mely az eszköz lábán keresztül lehet bemenet, kimenet vagy akár kétirányú is. Felépítését a 10. ábra szemlélteti.



10. Az I/O blokk belső felépítése

Két útvonalon érkeznek a bemeneti jelek az FPGA belsejébe, a képen I1 és I2-vel jelölt helyeken. A bemenetek egy programozható regiszterbe vannak előbb belevezetve, amik lehetnek él vezérlésű flip-flopok vagy szintvezérlésű latchek. A bemeneteket globálisan TTL vagy CMOS feszültségszintekre lehet beállítani. Valamint egy enyhe 300mV körüli hiszterézise van.

A kimeneti jelek opcionálisan invertálhatóak az IOB-ben és közvetlenül a kimenetre rakhatóak, vagy él vezérelt flip-flopokban tárolhatóak.

A maximálisan kivehető áramerősség általában 4mA, de az újabb FPGA-k akár már 12 mA-t is képesek kiadni. De ha két különálló lábat összekötünk, megmondhatjuk az FPGA-nak, hogy használja azokat együtt és így a kivehető áramerősség megduplázható. Alapértelmezésben a kimenet TTL szintű totem pole kimenetként viselkedik. Megadhatjuk, hogy CMOS feszültségszinteket használjon illetve képes még Open-drain és Tri-State-ként is viselkedni. [7]

1.3.2 A VHDL nyelv

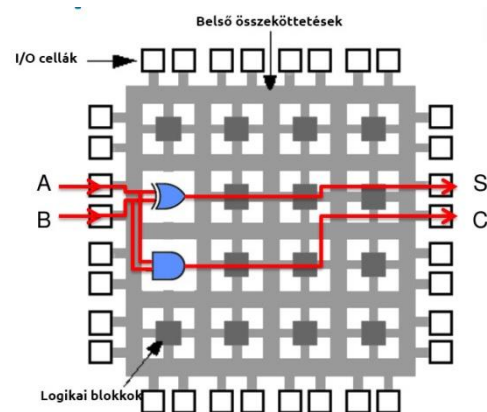
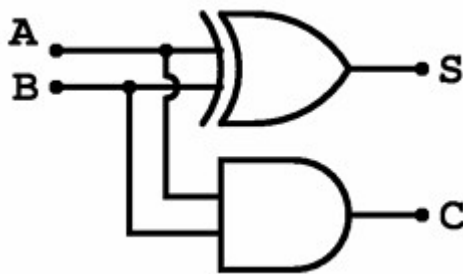
A VHDL ((VHSIC Hardware Description Language) (VHSIC - Very High Speed Integrated Circuit)) nevéből is adódóan, egy nagyon gyors sebességű integrált áramkör, hardvertervezési és -modellezési nyelve.

A logikai blokkok leírása két részre tagolódik:

- egységekre (*entity*)
- architektúrára (*architecture*)

Az egység meghatározása külső és belső szempontból történik, az egyik a neve valamint, hogy hova kapcsolódik (külső) és a működése (belső). Az egység, ha több különböző szintű összetevőből áll, külön leírhatjuk azok összekapcsolódását, valamint működését. [8]

Egy bites összeadó megvalósítása VHDL nyelven:



```
library ieee;
use ieee.std_logic_1164.all;

ENTITY Single_Adder IS

    PORT (
        A: IN std_logic
        B: IN std_logic ;
        S: OUT std_logic
        C: OUT std_logic
    );

END Single_Adder;

ARCHITECTURE Architecture_Single OF Single_Adder IS
BEGIN

    S <= A xor B;
    C <= A and B;

END Architecture_Single;
```

1.4 Felhasznált eszközök bemutatása

A DVB-T adás vételére, valamint annak IP hálózatban történő továbbítására a CableWorld Kft. által kifejlesztett készülékek kerültek alkalmazásra. A készülékek, a gyártó által egyedileg kifejlesztett CW-Net rendszerben működnek, amely:

- speciális CW-Net utasításokkal biztosítja a vezérlő számítógép és a készülékek közötti kommunikációt
- lehetőséget biztosít a Transport Stream továbbítására
- lehetővé teszi a TS analízisét

1.4.1 CW-4875 OFDM Demodulator Quad

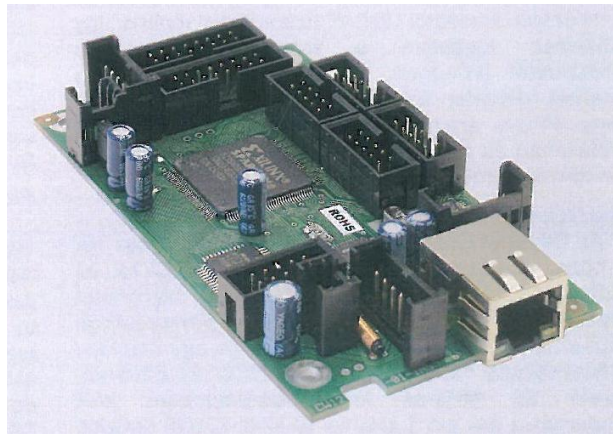
Az CW-4875 típusú készülék (11. ábra) négy OFDM demodulátort tartalmaz, mellyel egyidejűleg négy különböző frekvencia vételét teszi lehetővé. Az eszköz bemenetei felfűzhetők, így egyszerűbbé válik a bejövő jel szétosztása. A demodulátor paramétereinek beállításait egy erre a célra írt SW-4875 nevű szoftverrel lehet megadni.

A készülék része még egy CW-4901 típusú Gigabit Ethernet Controller is, mellyel a vételezett ASI (*Asynchronous Serial Interface*) jelet, számos beállítási lehetőség mellett IP hálózatba táplálhatjuk, valamint lehetőségünk van a hálózatról érkező, 4 különböző adatfolyam fogadására is. E funkciók számunkra megfelelő beállítását az SW-4901-es szoftveren keresztül lehet konfigurálni.[9]



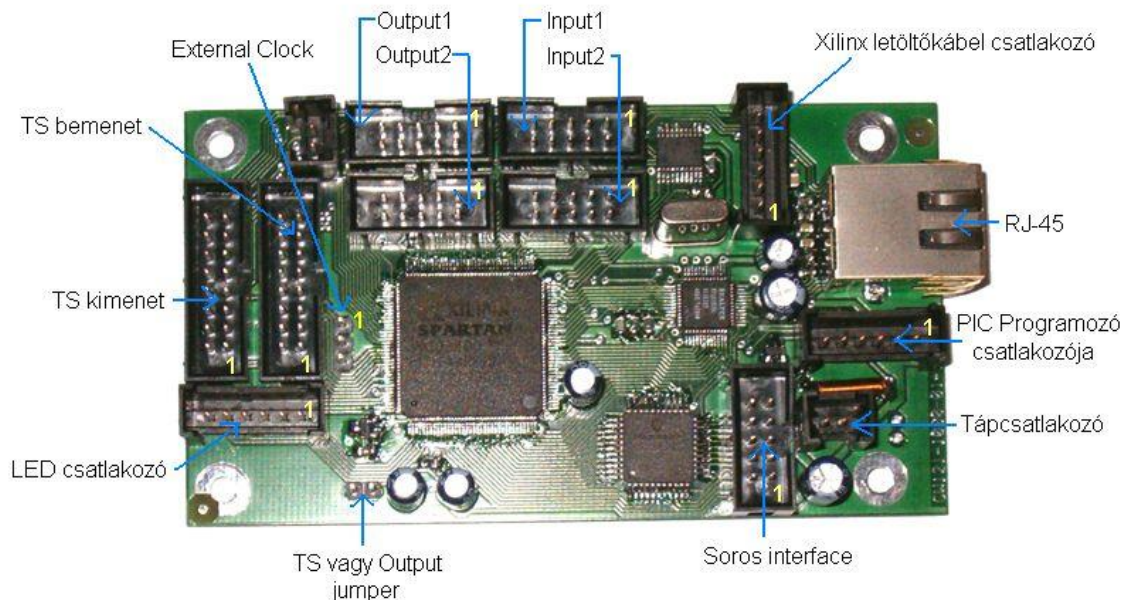
11. OFDM Demodulator Quad

1.4.2 CW-4900 Ethernet Controller



12. CW-4900 Ethernet Controller

A CW-4900 Ethernet Controller a CableWorld Kft. első olyan készüléke volt, mely képes a Transport Stream IP környezetbe helyezésére. Az eszköz interfésze 100Base-T környezetben működik. A TS csomagok küldésével kapcsolatos lehetőségek, a CW-4901-es készülékhez hasonlóan sokrétűek, annyi eltéréssel, hogy a CW-4900 csak unicast vételre képes. (12. ábra)



13. CW-4900 panel

A panelen található két-, egy TS küldésére- és egy TS fogadására használt csatlakozó, valamint két- CW-Net utasítással írható párhuzamos, 8 bites kimenet, és

bemenet. A TS kimenet és az Output1 csatlakozók funkciója egy jumper segítségével módosítható, ha az nyitott van mindkettő TS kimenet, ha zárt rendes Output.

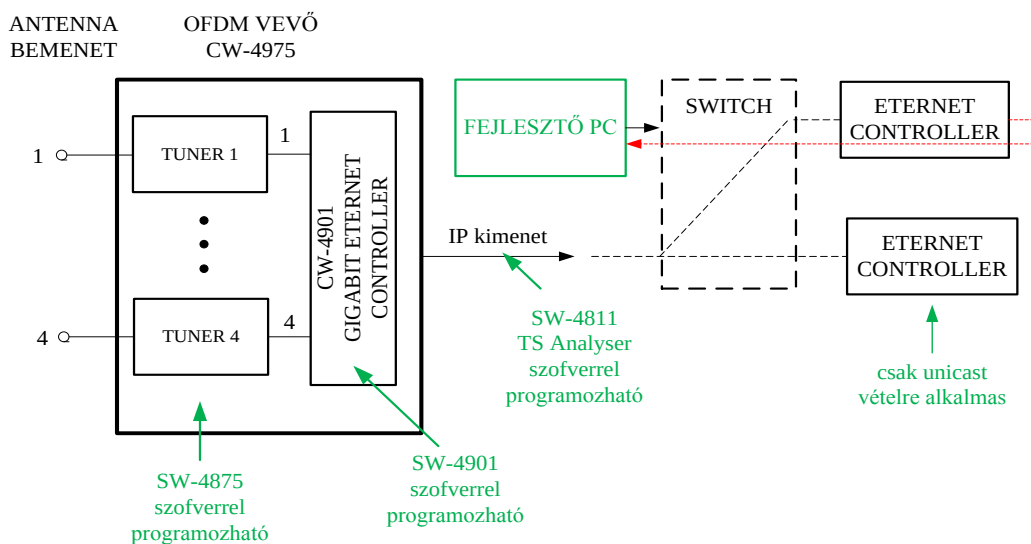
Az Ethernet Controller beállításai egy soros interfészen keresztül is írhatók vagy olvashatók ki, amely egy PIC18F452-es mikrokontroller lábaihoz csatlakozik, melynek EEPROM-jában az alkalmazott beállítások eltárolhatók, így azok ki- és bekapcsolás után is megmaradnak. A Transport Stream ASI kimenetre való küldéséhez szükséges órajelet egy NCO (*Numeric Controlled Oscillator*) szolgáltatja, melynek jele egy csatlakozóra is ki van vezetve. A készülék RJ-45 csatlakozóján keresztül, kereszt kábel használatával csatlakozik a hálózatra (13. ábra).

A panelen található 'Xilinx SPARTAN 3 XC3S400' FPGA feladatai:

- a bejövő ASI jel IP környezetbe ültetése
- a hálózatról érkező TS csomagok ASI kimenetre küldése
- ARP üzenetek kezelése
- CW-Net utasítások feldolgozása

2. Az alkalmazott összeállítás

A DVB-T antenna OFDM Demodulator tunerének bemenetére kötését követően, az SW-4875 szoftvert használva, a vételi jellemzők konfigurálására van lehetőség. A megfelelő beállítások alkalmazását követően a Gigabit Ethernet Controller-t vezérlő SW-4901 szoftverben az unicast küldési módot választottam, a CW-4900 Ethernet Controller IP címére küldtem a több programot tartalmazó MPTS-t (*Multi Program Transport Stream*). A kimeneti adatfolyamot egy arra kifejlesztett analízátor szoftverrel teszteltem, valamint a VLC média lejátszó segítségével jelenítettem meg. Az így tapasztalt eredmények alapján volt lehetőségem tesztelni a konfiguráció beállításait.



14. Az összeállítás blokkvázlata

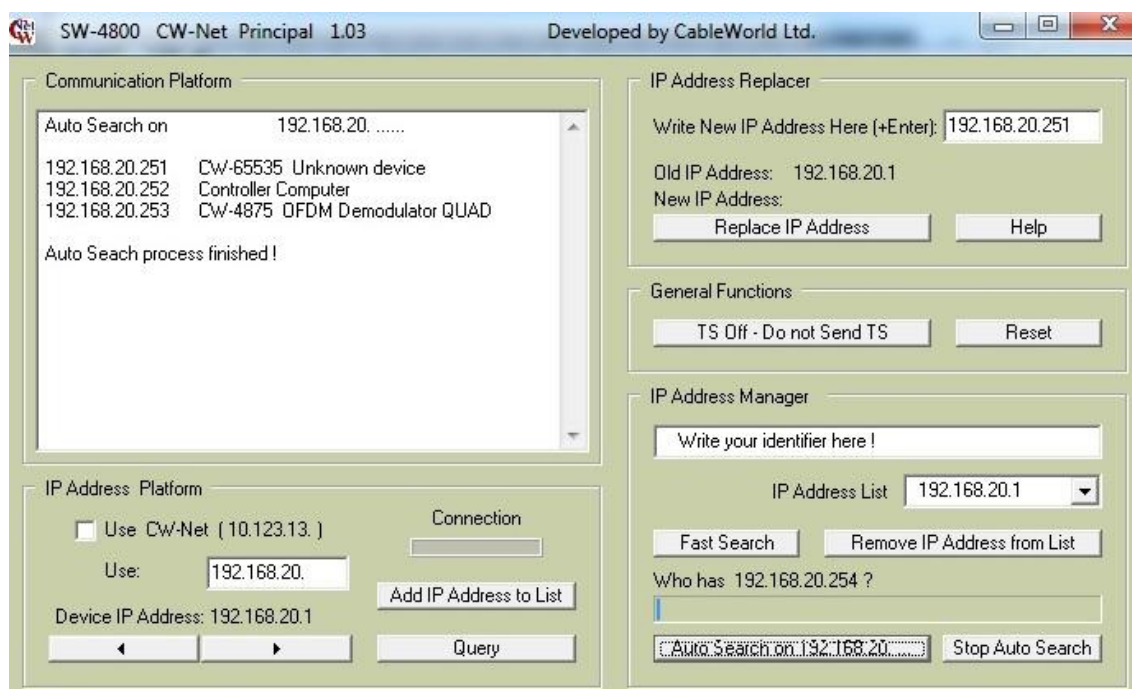
A CW-4900 panelre küldött IP adatfolyamot az FPGA a kimeneti TS csatlakozóra küldi, melyet - egy szalagkábel segítségével - a bemeneti TS csatlakozóval összekötve értem el, hogy a Transport Stream újra a hálózatba kerüljön továbbításra. Az FPGA-t VHDL nyelven felprogramozva valósítottam meg, hogy a több műsort tartalmazó adatfolyamból, csak az általunk nézni kívánt csatorna audio- és videó PID-jeivel rendelkező SPTS (*Single Program Transport Stream*) kerüljön továbbításra a felhasználók irányába. A szűrt, kimeneti TS-t már multicast küldési módban továbbítottam az intézményünk hálózatra (14. ábra).

Az összeállításról készített fénykép a négyes számú mellékletben látható.

2.1 IPTV hálózat konfigurálása

A Cable World Kft. készülékei kezelőszervekkel nem rendelkeznek, beállításuk számítógépen keresztül végezhető. A gyártó a készülékeit egy speciális CW-Net IP címmel látja el, melyek üzembe helyezéskor a 10.123.13.101 címen szólíthatók meg. Amennyiben saját hálózatunk nem ebben a tartományban van, a vezérlő számítógép hálózati azonosítóját, a készülék IP címének módosítása idejére 10.123.13.***-ra kell változtatni.

2.1.1 SW-4800 - CW-Net Principal 1.03

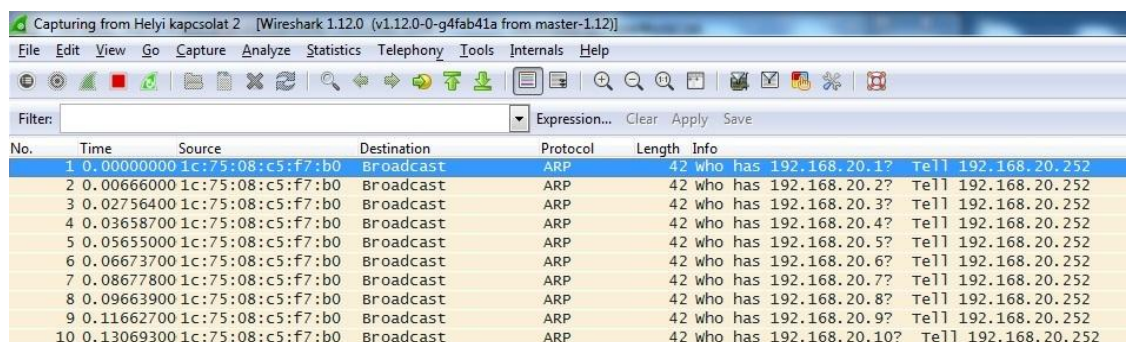


15. Az SW-4800 szoftver kezelőfelülete

A vezérlő számítógép IP címét, a készülék gyártáskor meghatározott IP tartományába állítva, a *Device IP Address* alatt található nyilakkal állítható be az adott cím mellyel kommunikálni szeretnénk. A 10.123.13.101-es IP cím megadását követően a *Query* gomb megnyomásával, sikeres kapcsolatfelvétel esetén a *Communication Platform*-on megjelennek a megszólított készülék adatai. Ezt követően az *IP Address Replacer* egységben írhatjuk be azt a címet, mellyel a jövőben használni szeretnénk az eszközt. Az Enter billentyű leütésével, majd a *Replace IP Address* gombra kattintva megtörténik a változtatás, melyet a saját számítógépünk helyes tartománybeli

visszaállításával, *Device IP Adress* átállításával, végül a *Query* lenyomásával a *Communication Platformon* ellenőrizhetünk (15. ábra).

Lehetőségük van a programmal kerestetni, hogy mely készülékek csatlakoznak egyidejűleg a hálózatra. Ehhez a 'Use:' ablakba kell begépelnünk a saját hálózatunk azonosítójának első három bájtyát, majd az 'Auto Search on 192.168.20. ...' gombra kattintanunk. Ekkor a szoftver Broadcast címmel az általunk meghatározott tartományban, ARP (*Address Resolution Protocol*) üzeneteket küldve, az IP negyedik bájtyának összes értékének lekérdezésével (1-255) deríti fel, hogy mely címmel rendelkező készülékek válaszolnak saját MAC címükkel az általa küldött üzenetekre. Ezt a folyamat szemléltethetjük a hálózati forgalom figyelésére alkalmas 'Wireshark' nevű program használatakor (16. ábra).



No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.1? Tell 192.168.20.252
2	0.00666000	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.2? Tell 192.168.20.252
3	0.02756400	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.3? Tell 192.168.20.252
4	0.03658700	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.4? Tell 192.168.20.252
5	0.05655000	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.5? Tell 192.168.20.252
6	0.06673700	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.6? Tell 192.168.20.252
7	0.08677800	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.7? Tell 192.168.20.252
8	0.09663900	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.8? Tell 192.168.20.252
9	0.11662700	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.9? Tell 192.168.20.252
10	0.13069300	1c:75:08:c5:f7:b0	Broadcast	ARP	42	who has 192.168.20.10? Tell 192.168.20.252

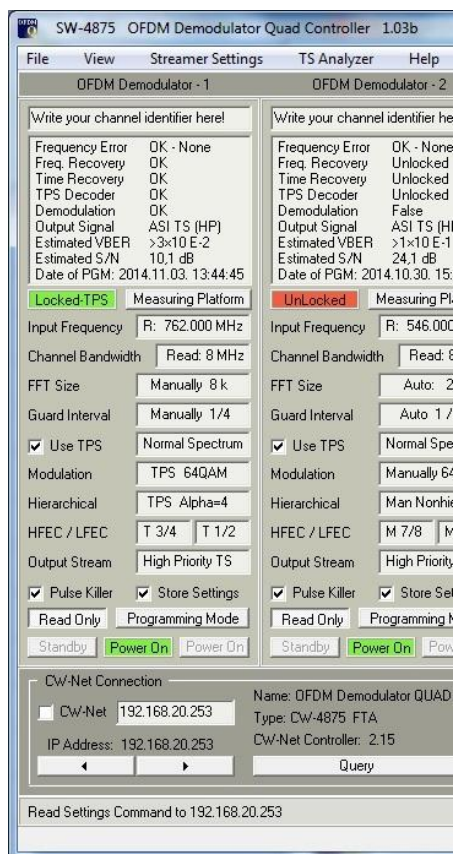
16. Hálózatra csatlakozott készülékek felderítése

A fentiekben ismertetett lépéseket elvégezve, az általam alkalmazott készülékek a következő IP címekkel kerültek alkalmazásra az összeállítás konfigurálása és a működtetése során:

- CW-4900 Ethernet Controller: 192.168.20.251
- Vezérlő számítógép: 192.168.20.252
- CW-4875 OFDM Demodulator Quad: 192.168.20.253

A 16. ábrán látható az "Auto Search..." lekérdezés eredménye, amely kilistázta a készülékek megnevezését, a már megváltoztatott IP címükkel együtt. [10]

2.1.2 SW-4875 – OFDM Demodulator Quad Controller



17. SW-4875 kezelőfelülete

Az SW-4875 szoftver segítségével konfigurálhatjuk a 4 demodulátort, a DVB-T adó paramétereire (17. ábra).

Első lépésben az általunk használt eszköz és a szoftver közötti kommunikáció létrejöttéről kell meggyőződnünk. Ha a CW-4875 IP címét megadva, majd a *Query* gombra kattintva, megjelenik a készülék neve és típuszáma, a kapcsolat létrejött.

Lehetőségünk van a vételi adatok egyszeri-, vagy ismétlődő automatikus kiolvasására, vagy a jellemzők manuális konfigurálására is.

Magyarországon az Antenna Hungária által megadott műszaki paraméterek a következők [1]:

Vivők száma: IFFT 8k

Moduláció: 64QAM

Kódarány: $\frac{3}{4}$

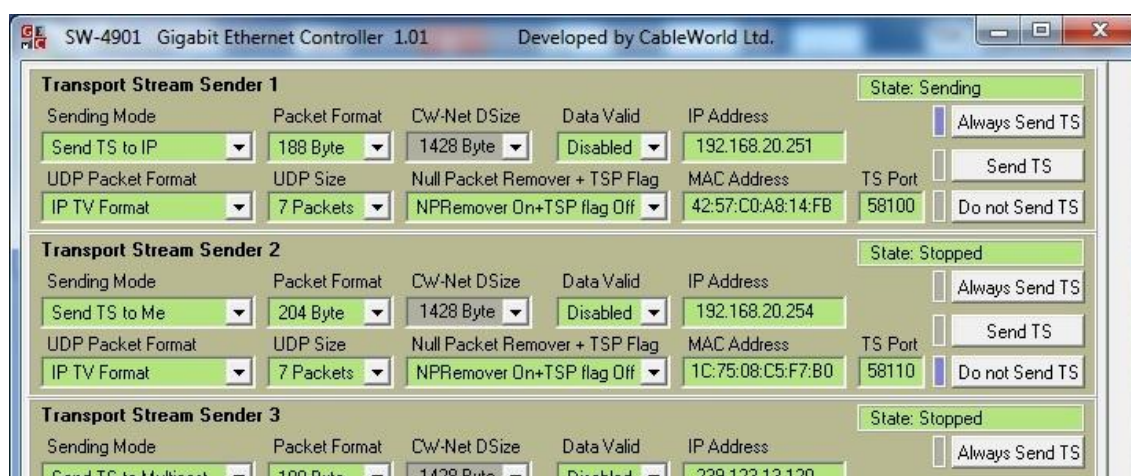
Védelmi intervallum: GI $\frac{1}{4}$

A konfigurálás során a vételi jellemzők manuális beállítására nem volt szükségem, az automatikus kiolvasatással történő beállítást alkalmaztam, amellyel a vétel minősége megfelelő volt.

2.1.3 SW-4901 – Gigabit Ethernet Controller 1.01

Az OFDM Demodulator Quad részét képező CW-4901 Gigabit Ethernet Controller kezelőfelülete több lapon kínál fel beállítási lehetőségeket.

Az első ilyen fül a *General Device Controller* mely kinézetében és funkcionalitásában alig tér el a CW-Net Principal szoftvertől. Ezt követi a *Network Setup*, ahol a hálózati paramétereket kérdezhetjük le, valamint szabályozhatjuk a kommunikációra-, valamint a TS küldésére használt port tartományt. Az *IP Receiver* opcióval négy különböző-, az IP hálózaton érkező Transport Stream vételét állíthatjuk be, ez azonban esetünkben most nem használatos.



18. TS Sender platform

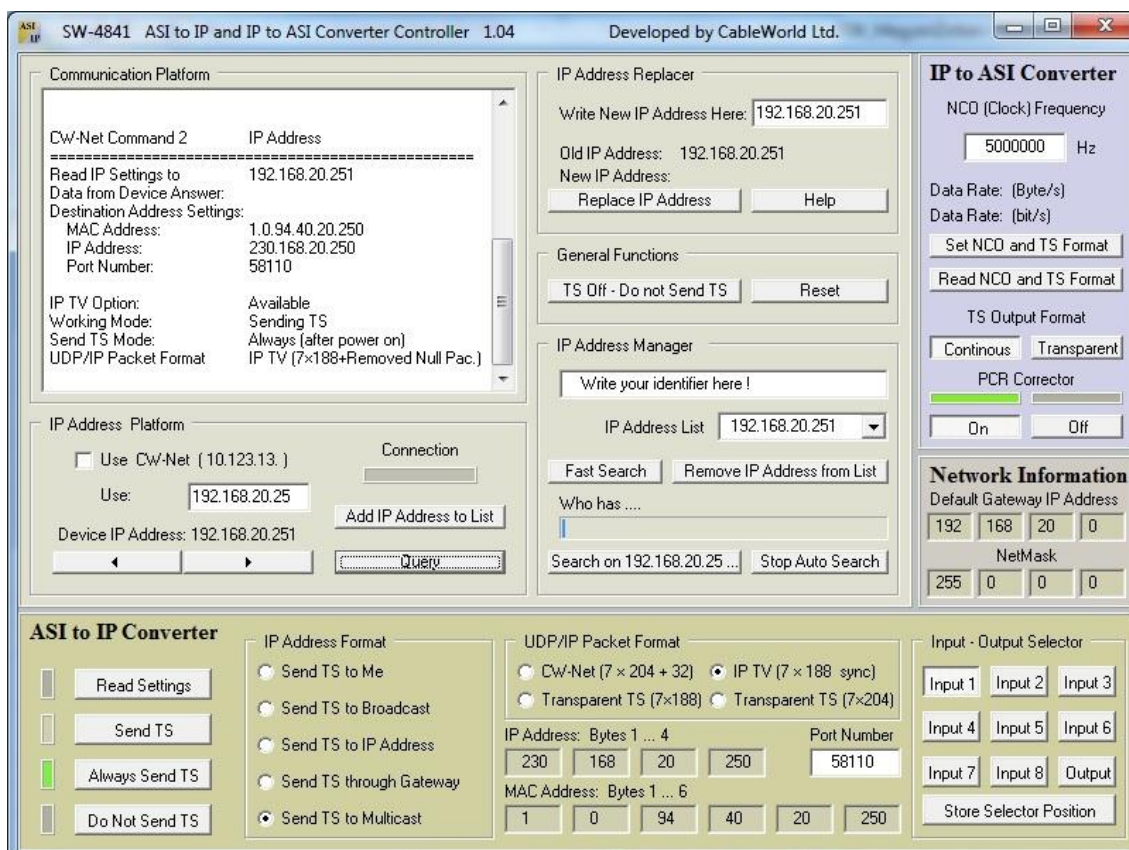
A *TS Sender Platform* segítségével van lehetőségünk a Demodulátor felől a CW-4900 felé küldeni az adatfolyamot. A küldési mód kiválasztását a *Sending Mode* menüvel tehetjük meg. Több küldési mód közül választhatunk, a kimeneti adatfolyamot küldhetjük: a vezérlő számítógép felé (*Send TS to Me*), üzenetszórásos küldéssel (*Send TS to Broadcast*), egy adott IP címre (*Send TS to IP*), multicast címre (*Send TS to Multicast*). Az előbb felsorolt esetekben, a szoftver magától állítja (multicast küldés esetén számolja) a 'MAC Address' mező értékét. A *Send TS to Man Address* küldési módot választva azonban lehetőségünk van az IP cím és a MAC cím általuk történő beállítására.

Mivel az eszköz, melyen a TS szűrését végezem el csak unicast vételre képes, ezért a *Sending Mode* legördülő menüből a *Send TS to IP* módot választottam, majd az *IP Address* mezőbe az Ethernet Controller 192.168.20.251-es címét adtam meg, valamint az 58100 –as portot választottam ki a Transport Streamet továbbítására (18. ábra).

Ezt követően az *Always Send TS* gombra kattintottam, mellyel megkezdtem az adatok küldését, mindemellett a készülék eltárolja a megadott beállításokat és ki-, bekapcsolás után is ezeket alkalmazza.

2.1.4 SW-4841 – ASI to IP and IP to ASI Converter

A CW-4900 Ethernet Controller. TS kimeneti és bemeneti csatlakozóik összekötése miatt szükségünk van egyszerre a kimeneti ASI jel paramétereinek, valamint az IP környezetbe ültetett adatfolyam kimeneti beállításainak konfigurálására. Erre a célra az SW-4841 szoftver használata a legmegfelelőbb (19. ábra).



19. Az SW-4841 –ASI ti IP and IP to ASI Converter Controller

A készülék IP címének megadását követően, majd a *Query* gomb lenyomása után meggyőződtem arról, hogy a program kommunikál a csatlakoztatott készülékkel. Az *IP to ASI Converter* felület alatt határoztam meg a panelen található NCO kimeneti frekvenciáját, valamint a TS küldési módját állítottam be. A *Continuous* kimeneti formátumot választottam, mivel így csak a hasznos adatok kerülnek kiküldésre, ellenben a *Transparent* formátumot választva, ahol a hasznos adatot nem tartalmazó Null Packet-ek is továbbbitódnak, megnövelve ezzel a TS adatsebességét. Ezt követően a *Set NCO and TS Format* gombot lenyomva eltároltam az alkalmazott beállítást.

A bemeneti TS hálózatra küldésének jellemzőit a Gigabit Ethernet Controller-hez hasonló lehetőségekkel tudjuk beállítani, itt viszont már, hogy a műsor helyi hálózaton

is elérhető lehessen a multicast küldési módot-, multicast címnek a 230.168.20.250-es IP címet választottam. A TS küldésére az 58110-es portot állítottam be. E szoftver a gigabites verziótól eltérően nem számolja ki automatikusan a multicast küldéshez szükséges speciális MAC címet, melyet a 20. ábrán látható összefüggés alapján határoztam meg hexadecimális formában és ezután decimális alakban adtam meg a szoftverben. Végül az *Always Send TS* gombra kattintva megkezdtem a több programot tartalmazó MPTS hálózatba történő streamelését. [3]

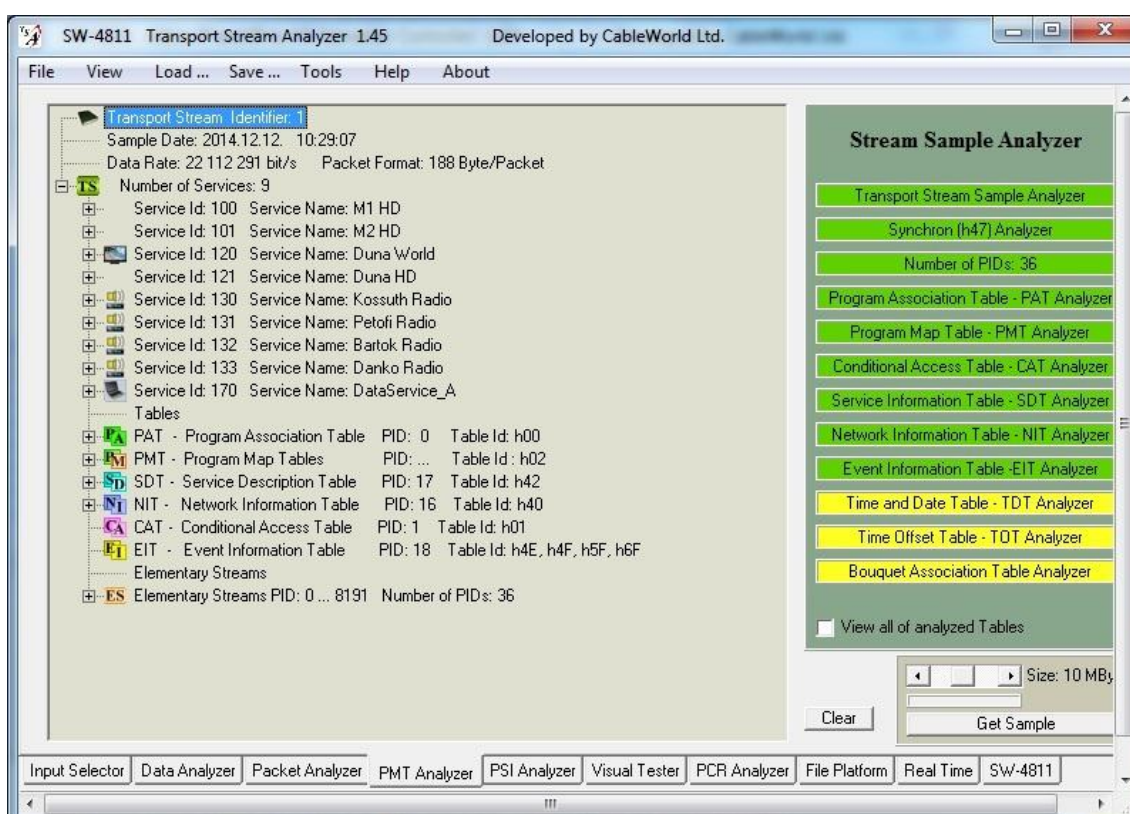
01:00:5E:(IP2 and h7F):IP3:IP4

20. A multicast MAC cím számítási módszere

2.1.5 Mérés a SW-4811B – TS Analyzer használatával

Az SW-4811B program alkalmazásával lehetőségünk van Transport Stream mintavételezésére, valamint a vett mintát különböző szempontok alapján elemezni, melyre a program számos lehetőséget kínál.

Az *Input Selector* felületen van lehetőségünk kiválasztani, mely - a hálózatban megtalálható - eszköz által küldött TS-t szeretnénk elemezni, vagy esetleg egy multicast streamből szeretnénk-e mintát venni. Ugyanitt állíthatjuk a várt UDP csomag formátumát, vagy annak meghatározását a rábízhatjuk a programra is.



21. A szűretlen multicast stream analízálása

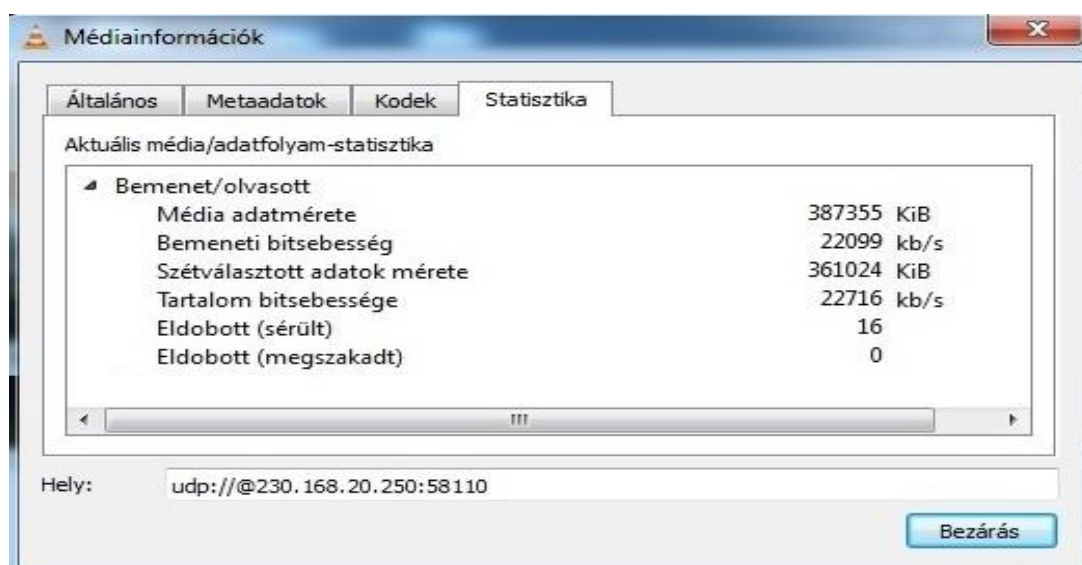
A program *View* menüjében a *Multicast Network Analyzer* opciót választottam, hogy a CW-4900-ról küldött multicast streamből tudjak mintát venni. A program *Input Selector* oldalán ennek megfelelően adtam meg a *Multicast IP (230.168.20.250)* és a *Transport Stream Port (58110)* mező értékeit, majd a *Get Sample* gombra kattintva, a felette található görgőn beállított, 10 Mbyte-os mintát vettem az adatfolyamból.

A TS összetételének szemügyre vételéhez a 21. ábrán látható *PMT Analyzer* felületén látható, hogy a teljes adatfolyam sebessége körülbelül 22 Mbit/s, a csomagok pedig 188 bájt méretűek. A stream 8 különböző televízió/rádió csatorna adatait tartalmazza,

melyek azonosítója és neve is kijelzésre kerül, ezek alatt találhatóak a különböző táblák PID értékei és táblaazonosítói, majd ezt követően az, hogy hány különböző Elemi Adatfolyamot tartalmaz a mintavételezett TS, valamint ezek PID értékei.

2.1.6 Megjelenítés és mérés a VLC médialejátszó használatával

A hálózatra küldött Transport Streamet a beépített MPEG4 dekóderrel rendelkező VLC médialejátszóval jelenítettem meg, melyet a Média menü legördítésével, majd a Hálózati műsor megnyitása menüpont kiválasztásával állítottam be, miután begépeltem az általam beállított multicast címet és portszámot a következő képpen: **udp://@230.168.20.250:58110**.



22. A szűretlen adatfolyam bitsebessége

A *Lejátszás* menü *Program* menüpontja fölé állítva az egér mutatóját jelenik meg a televízió-, valamint a rádió csatornák neve melyet tetszőlegesen kiválasztva megjeleníthetünk vagy hallgathatunk. A lejátszó lehetőséget ad az aktuálisan nézett televízió csatorna képűségjának megjelenítésére is, melyet a műsor megnyitása után a Play gomb feletti ikonra kattintva érhetünk el, megadva a nézni kívánt oldal számát.

A lejátszó *Eszközök* menüpontja alatt található *Médiainformációk* menüpontra kattintottam, mellyel megfigyelhettem, hogy a bemeneti bitsebesség igen kis mértékben tér el az előzőleg mintavételezett kijelzett adatsebességtől (22. ábra), a megjelenítés minősége nem volt kifogásolható.

2.2 A csatorna szűrés megvalósítása

Azt, hogy az általunk választott csatorna kerüljön csak streamelésre a hálózatba, a TS csomagok PID azonosítójának VHDL nyelven történő szűrésével végeztem el, kibővítve a CW-4900 Ethernet vezérlőn található FPGA forrásprogramját, melyet a ByteStudio Bt. fejlesztőmérnöke, dr. Zigó Tamás fejlesztett. A szűrést végző összetevő a VHDL kódban a *SynchronSeparatorZ* nevű azonosítóval rendelkezik és eredeti feladata karakterek írása volt, a Null Packet-ek hasznos adat részébe. Az egység három bemeneti- (*pRESET*, *pTSCLK*, *pTSDATAIN*), valamint egy kimeneti porttal (*pTSDATAOUT*) van ellátva (23. ábra).

A hálózatról érkező adatfolyam ASI kimenetre történő küldését az 1. számú mellékleten is látható *TSOutUnit* összetevő végzi, az ASI bemenetről érkező jelet pedig a *TSUnit*. Mindkettő részét képezi a *SynchronSeparatorZ*, megvalósítva ezzel azt, hogy az Ethernet vezérlő, ha a jelenlegi összeállítástól (ASI ki és bemenetek összekötése) eltérően van alkalmazva, ugyanúgy elvégezze a szűrési folyamatot.



23. A SynchronSeparatorZ kimeneti- és bemeneti portjai

A *TSOutUnit* összetevő az általunk beállított NCO frekvenciájával megegyező adatsebességgel folyamatosan Null Packet-eket generál, melyek hasznosadat részébe a 'CableWorld' karaktereket illeszti be. A CW-4901 Gigabit Ethernet Controller felől általunk küldött adatfolyamot egy belső-, FIFO (*First In First Out*) szervezésű memóriába pufferelem, majd a generált Null Packet-ek közé beillesztve helyezi a kimenetre. Az összetevő forráskódja úgy van kialakítva, hogy ha, az SW-4875 szoftverben az *IP to ASI Converter* felület alatt a *Transparent* opciót választjuk a TS küldési módjaként, akkor az adatfolyam megkerüli a kialakított szűrést végző összetevőt, így ha csak ASI kimenetről szeretnénk vételezni a szűrt streamet, a *Continous* beállítást kell alkalmaznunk. Esetünkben e beállítás helyes megválasztása a

ki és bemenetek összekötéséből adódóan nem jelent problémát, mivel a *TSUnit* egységben a szűrés ismételt végrehajtásra kerül.

E *TSUnit* összetevő *pTCLK* és *pTSDATA* bemeneti portjaira érkező adatfolyamot elsőként a *SynchonSeparatorZ* PID szűrő összetevő kapja meg, elvégezve a szűrést továbbítja a szintén a *TSUnit* részét képező összetevő felé, amely a TS csomagok szinkronizálását végzi a 47_h szinkronbájtok-, valamint azok távolságának figyelésének segítségével. A szinkronizáló egység a rendszerezett adatokat továbbítja, majd a *TSUnit* a csomagokat szintén egy FIFO szervezésű RAM-ba gyűjti. Ha azok száma eléri a 7-et, a PIC mikrokontrollertől kapott célcímet tartalmazó kerettel látja és küldi a hálózatba.

A szűrést végző *SynchonSeparatorZ* összetevő funkcionalitását tekintve hasonló működésű, mint a szinkronizáló egység, mivel a PID értékek vizsgálatához - a TS csomagban történő elhelyezkedésük miatt - ismernünk kell a csomag kezdetét.

A működés meghatározása a VHDL kódban (2. számú melléklet) egy sorrendi, algoritmikus működést eredményező *PROCESS* utasításon belül történik, amely az *pRESET* valamint a *pTCLK* portokon beérkező jelek felfutó éleire hajtja végre a benne meghatározott, éppen aktuális utasításokat. (Az *sTSDATAOUT* vektor közvetlen kapcsolatban áll az egység kimeneti *pTSDATAOUT* portjával, mely a kód a *PROCESS*-en kívül, az *egyidejű* működést meghatározó részében került bejelentésre.)

Mivel a TS packetek hasznos adatrészében is előfordulhat a szinkronbájttal azonos 47_h értékű bájtt, ezért szükség van egy *sCounter*-nek elnevezett 8 bites vektor használatára, amely számlálóként funkcionál és a TS csomag méretét számolja 0-tól (00_h) 187-ig (BB_h), vagy, ha a hálózatba küldés módjának nem az IPTV-, hanem a CW-Net opció van kiválasztva, 0-tól (00_h) 203-ig (CB_h). Szükség van még, egy 3 bites vektor használatára, amely a szinkronizálás sikerességét ellenőrzi. Ez a programban az *sLockCounter* elnevezéssel látható, amely 1-től 6-ig számol felfelé, amennyiben a szinkronizálás sikeres volt, majd tartja ezt az értékét. A szinkron elvesztésének esetén ezt az értéket csökkenteni kezdi, majd, ha az *sLockCounter* értéke eléri az 1-et, a program vár a következő 47_h értékű bájtra és újra kezdi a szinkronizálás próbálását.

Azt, hogy a program éppen a 47_h értékű bájtra várakozzon, vagy ellenőrizze, hogy az a bájtt szinkronbájtt-e, egy *Case* utasítás feltételében megadott, *vState* változó értékének meghatározásával történik (*Waiting*, *Trying*, *Locked*).

Ha a program 47_h bájtra várakozik (*Waiting*), az *sTSDATA* vektor értékét figyeli, miközben a beérkező adatokat a kimenetre továbbítja. Ha a vektor értéke egyezik a várt értékkel, az *sCounter* számláló értékét '0'-ba állítja, és a *vState* változónak *Trying* értéket ad.

Ekkor kezdődik el a vizsgálat, hogy a megfelelő-e a távolság a várt értékű bájtok között. A beérkező adatokat folyamatosan a kimenetre továbbítva elkezdődik a számláló értékének növelése, valamint az *sTSDATA* figyelése, mikor veszi fel a vektor ismét a szinkronbájt értékét. Amennyiben az *sTSDATA* és az *sCounter* vektorok értéke megfelelő volt, úgy az *sLockCounter* értéke '001' lesz, a számláló nullázásra kerül, az *s188or204* jel értéke a számláló állapotától függően az '1' vagy a '0' értéket veszi fel, valamint a *vState* változó a *Locked* értéket kapja meg. Amennyiben a számláló eléri a 203-as értéket anélkül, hogy az *sTSDATA* vektor 47_h értékű lett volna, újra megkezdődik az e bájtra történő várakozás a '*vState := Waiting*' értékadással.

A *vState* változó *Locked* értéke a programban, a Case utasítás '*others*' esete alatti működési algoritmus megkezdését eredményezi, amely az előzőleg említett *Trying* eset alatt nulla értékre állított számláló inkrementálását végzi. Ha a számláló értéke elérte a BB_h, vagy a CB_h értéket és az *s188or204* jel ezen értékek megfelelőjével egyezik, egy teljes csomag már tovább lett küldve a *TSUnit* összetevő felé. Ekkor az *sCounter* értéke ismét a 00_h értéket-, az *sTSDATAOUT* vektor pedig a 47_h értéket veszi fel. A számlálás ekkor újrakezdődik, és a TS bájtjainak értékét - a számláló értékének növekedésével egy időben - *sTSDATAOUT* vektor veszi fel, majd azok megjelennek a *pTSDATAOUT* kimeneti porton. Itt inkrementálódik vagy dekrementálódik az *sLockCounter* vektor is attól függően, hogy sikerült-e a TS csomagokat szinkronizálni, vagy sem.

A bemeneti *pTSDATAIN* porton érkező jel - a Transport Stream órajel, a *pTSCCLK* felfutó élére történő - mintavételezésével állapítható meg a PID érték, melyet akkor olvasunk ki, ha az *sTSDATA* vektor a szinkronbájt értékét veszi fel.

E mintavételezés az alábbi vektorokkal a következő adatfolyamból történik a programban, még a már fent említett Case utasítás előtt:

pTSDATAIN -> *sPIDL* -> *sPIDH* -> *sTSDATA*

A PID vizsgálat és szűrés a Case utasítás *vState* változójának *Locked* értékénél hajtódik végre.

2.2.1 A csatorna szűrés folyamatának szimulációja

A szűrés tesztelését az idődiagramok jobb átláthatósága érdekében, első lépésben csak 1 PID értékkel szemléltetem.

A Transport Stream csomag fejlécében elhelyezkedő 13 bites PID értékek összehasonlítását a programkódban úgy valósítottam meg, hogy az *sPIDH* vektor alsó öt bitjét (*4 downto 0*) és a nyolcbites *sPIDL* vektor összefűzésével vizsgáltam egy 'if' utasításban. Ha az összefűzött vektorok értékei a feltételben megadott bitsorozattal egyeztek vagy sem, az *sFILTER* jel került beállítására. Az értéke '0', ha az adott csomag tovább engedhető, ekkor a nézni a kívánt program PID értéke kerül a kimenetre a szinkronbájtot követően. Ellenkező esetben az *sFILTER* értéke '1', tehát a beérkezett TS csomagot szűrni kell. Ebben az esetben a nem kívánt csomagazonosítók helyett a Null Packet PID értéke (1FFF_h) kerül továbbításra a számláló megfelelő állapotánál a *TSUnit* egység felé. (24. ábra)

```
if (sPIDH(4 downto 0) & sPIDL = "00000001100100") then
    sFILTER <= '0';
else
    sFILTER <= '1';
end if;
```

24. PID érték vizsgálata

Az *sCounter* 00_h értékével egy időben a szinkronbájt már megjelenik a szűrés végző egység kimenetén, ha a csomagot szűrni kell, ahhoz hogy a TS fejléc második és a harmadik bájtján elhelyezkedő módosított PID értékek a megfelelő időzítéssel kerüljenek a kimenetre, egy órajel ütemmel előbb kell megadnunk a VHDL kódban, hogy milyen esemény történjen a következő *pTSCLK* jel felfutásakor. A számláló 00_h és 01_h állapotánál kell felülírnunk a TS csomag azonosítójának bitjeit, hogy azok a második és a harmadik bájtként kerüljenek a kimenetre. A PID érték alsó bájtjának felülírásával egy időben az *sFILTER* jel értékét '0'-ra állítottam, így az *sCounter* egyéb értékeinél a beérkező csomag bájtjai eredeti értékükkel kerülnek továbbításra.

Mivel a fejléc második bájtjának csak az alsó 5 bitje tartozik a PID értékhez, a felső 3 bitet (Átviteli hibajelző, Hasznos adatrészt kezdete, Átviteli prioritás) módosítás nélkül küldtem tovább a kimenetre, melyet az *sTSDATA* vektor és a Null Packet azonosítójának felső 5 bitje (1F_h) közötti érték, 'vagy' kapcsolattal történő maszkolásával valósítottam meg (25. ábra).

```

else
    if sFILTER = '1' then
        case sCounter is
            when X"00" => sTSDATAOUT <= (sTSDATA or X"1F");
            when X"01" => sTSDATAOUT <= X"FF"; sFILTER <= '0';
            when others => sTSDATAOUT <= sTSDATA;
        end case;
    else
        sTSDATAOUT <= sTSDATA;
    end if;

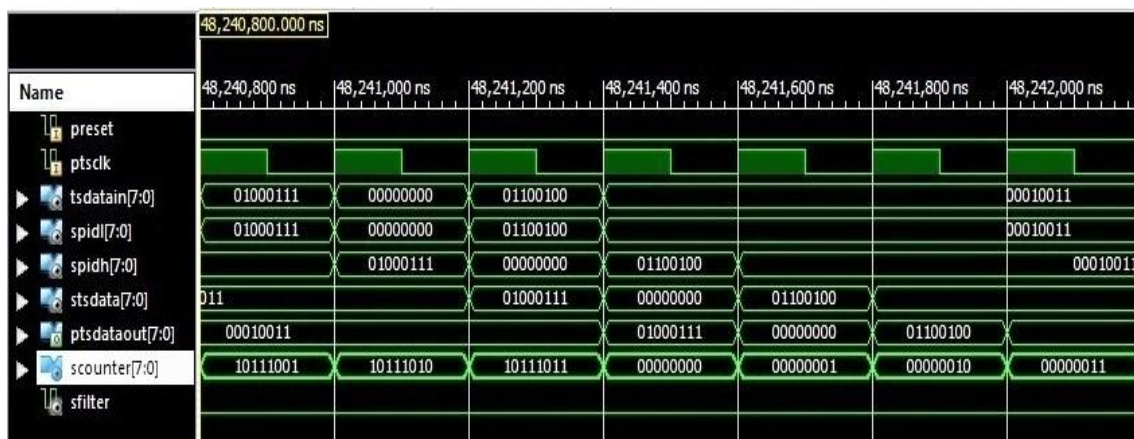
```

25. A felülírt és az eredeti PID érték kiküldése

A szűrési algoritmus szimulálására a Xilinx fejlesztőkörnyezeten belül van lehetőség, ehhez azonban szükséges egy tesztfájl létrehozása, mellyel az alkalmazott modul be- és kimeneti portjait megfelelően összekötve a tesztfájlban létrehozott jelvezetékekkel és vektorokkal, lehetőség van szabályozni a vizsgálandó komponens bemenetére küldött adatokat, valamint megfigyelni a kimenet alakulását a VHDL kódban megadott feltételek alapján (3. számú melléklet).

A tesztfájlban PID szűrő összetevő három bemeneti portjára történő jel- és adatgenerálás miatt, három különböző *PROCESS*-t hoztam létre. A *TSCLK_process* -ben, 100ns-os késleltetésekkel a jelszint váltások között- 200ns-os periódusidejű Transport Stream órajelet generáltam, a *RESETPROCESS* -ben a Reset jel magas szintjének tartását 150ns-ra, az alacsony szintjének tartását a folyamatok változásának megfigyelése miatt 3000us-ra állítottam. Valódi TS csomagok bájtjainak bemenetre történő generálását a *TSDATAINPROCESS* -ben valósítottam meg, ahol az órajel értéke miatt 200ns-os késleltetéssel először a szinkronbájtot, majd a következő két bájt jelen esetben a 0064_h küldtem a bemenetre. Hogy teljes méretű csomaggal történjen a szimuláció, az első három bájt fix értékét követően, a többi 185 bájt, állandó 13_h értékkel, 'for' ciklussal generáltam a bemenetre.

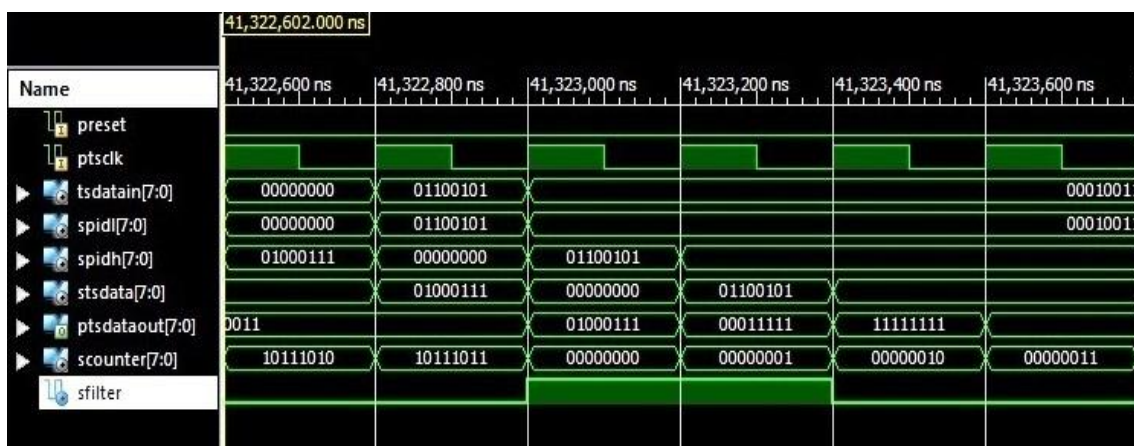
A 26. ábrán látható, hogyan veszik fel az értékeket a jelvezetékek a számláló különböző állapotaira. Látható, hogy a bemeneten elsőként a 47_h értékű szinkronbájt jelenik meg, mellyel egy időben ezt az értéket az *SPIDL* jel is felveszi, a számláló értékének növekedésével, tehát minden TS órajelre az *SPIDH* és végül az *sTSDATA* jelek kapják meg ezt az értéket. A modulon belül minden beérkező adat ezeken a vektorokon, ugyan ebben a lépésben halad.



26. Szimuláció a megengedett PID érték esetén

A PID értékek vizsgálata az *STSDATA* jel 47_h értékénél történik meg. Látható, hogy a tesztfájlban egy 0064_h PID értékű csomag került generálásra. Ezt a számláló 187 (10111011_b) értékénél figyelhetjük meg az *SPIDH* és az *SPIDL* vektorok vonalában. Mivel a szűrési feltétel erre a PID értékre volt állítva, az *SFILTER* jel 0 értéken maradt, így a *PTSDATAOUT* kimeneti porton a szinkronbajtot követően, az eredeti azonosító értékei kerültek a kimenetre.

A bemenetre generált, továbbítani nem kívánt PID érték esetén (jelen esetben: 0065_h) a 27. ábrán látható, hogy a vizsgálatot követően az *SFILTER* jelvezeték értéke - a számláló 0 értékénél – magas szintet vesz fel, a *PTSDATAOUT* kimeneten pedig ezt követően a Null Packet azonosítója, a 1FFF_h kerül a kimenetre.



27. Szimuláció a szűrendő PID érték esetén

2.2.2 A csatorna szűrés gyakorlati megvalósítása

A szűrés folyamatának megismerése után ahhoz, hogy egy csatorna váljon csak elérhetővé az adott hálózaton, ismernünk kell, hogy milyen PID értékkel rendelkező Elemi Adatfolyamok alkotják. A magyarországi DVB-T adás paramétereit - melyek az Antenna Hungaria Zrt. oldalán találhatóak meg - a szolgáltatási paramétereket és kódolási technikákat tartalmazó táblázatban tekinthetünk meg. E táblázat tartalmazza az egyes adásokhoz tartozó ES-ek PID-jeit. Az 'M1 HD' adását szűrtem ki az 'A'-Közszolgálati multiplexből. E csatornához tartozó Elemi Adatfolyam PID értékei a következők:

Elemi Adatfolyam	PID (decimális)	PID (bináris)
Videó	1001	0001111101001
1. audió	1002	0001111101010
2. audió	1004	0001111101100
3. audió	1008	0001111110000
Teletext	1003	0001111101011
DVB-felirat	1005	0001111101101

A szűrt csatorna megjelenítéséhez azonban a dekóder számára nem elég kizárólag a fent felsorolt értékek szűrése, szükséges még PAT és a PMT táblák továbbítása is, tehát ez azt jelenti, hogy a hat ES PID mellett szükség van még a 0000_h és az 'M1 HD'-hoz tartozó, 03E8_h PID értékek szűrésére is.

A csomagazonosító értékek vizsgálatához a VHDL kódban, nyolc darab, 13 bites vektort definiáltam *sFPID1*, *sFPID2*, *sFPID3*, *sFPID4*, *sFPID5*, *sFPID6*, *sFPID7*, *sFPID8* neveken.

E vektorok értékadása a Reset jel hatására történik meg, így az üzembe helyezéstől kezdve megtartják értéküket. A PID vizsgálat során az összefűzött, beérkező PID értékekkel hasonlítottam össze a létrehozott vektorokat, melyhez az *sFILTER* jel értékét állító feltételek számát is a vektorok számával egyenlővé tettem (2. számú melléklet).

A felülírandó TS csomagok bájtjainak kimenetre történő küldése eltér a szimulációban szemléltetett módszertől. A számláló első két értékénél, a PID érték átírásában nem történt változás. Az azt követő bájtban, az eredeti adatokat továbbítottam, hiszen azok még a fejléchez tartoznak (Átviteli prioritás (2 bit), Adaptációs mezőt vezérlő (2 bit), Folytonosság számláló (2 bit)). Ezt követően a "MEGYERI OE-AMK" karakterek ASCII kódját írom a TS csomag hasznos adat részébe, majd a Null Packet-ek adattartalmának eredeti értékével töltöm ki a maradék bájtokat a csomagban (28. ábra). Ebben az esetben, hogy a teljes csomag ki legyen töltve az FF_h értékkel az *sFILTER* jel, következő vizsgálathoz történő '0' értékbe állítása, a számláló 187 vagy 203 értékeinél történik meg.

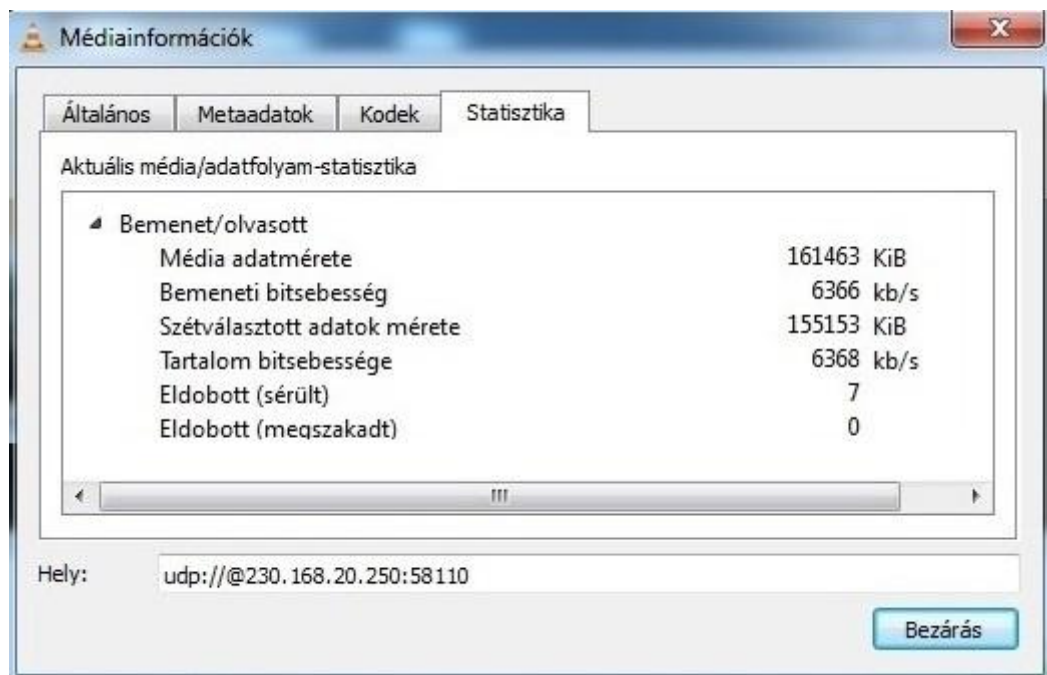
```

else
  if sFILTER = '1' then
    case sCounter is
      when X"00" => sTSDATAOUT <= sTSDATA or X"1F"; -- NP PID
      when X"01" => sTSDATAOUT <= X"FF"; -- NP PID
      when X"02" => sTSDATAOUT <= sTSDATA;
      when X"03" => sTSDATAOUT <= X"20"; -- szóköz
      when X"04" => sTSDATAOUT <= X"4D"; --M
      when X"05" => sTSDATAOUT <= X"45"; --E
      when X"06" => sTSDATAOUT <= X"47"; --G
      when X"07" => sTSDATAOUT <= X"59"; --Y
      when X"08" => sTSDATAOUT <= X"45"; --E
      when X"09" => sTSDATAOUT <= X"52"; --R
      when X"0A" => sTSDATAOUT <= X"49"; --I
      when X"0B" => sTSDATAOUT <= X"20"; -- szóköz
      when X"0C" => sTSDATAOUT <= X"4F"; --O
      when X"0D" => sTSDATAOUT <= X"45"; --E
      when X"0E" => sTSDATAOUT <= X"2D"; -- kötőjel
      when X"0F" => sTSDATAOUT <= X"41"; --A
      when X"10" => sTSDATAOUT <= X"4D"; --M
      when X"11" => sTSDATAOUT <= X"4B"; --K
      when others => sTSDATAOUT <= X"FF";
    end case;
  else
    sTSDATAOUT <= sTSDATA;
  end if;
end if;

```

28. PID szűrés és karakterbeszúrás

A VLC médialejátszó megnyitása után, az összeállítás konfigurálásakor alkalmazott módon a hálózati műsorának megnyitását követően, csak az általam meghatározott M1 HD csatorna került megjelenítésre, melynek adatsebességét a *Médiainformációk* menüpont alatt ellenőriztem (31. ábra), valamint a *Lejátszás* menü, *Program* menüpontja is szürke színű lett, ami azt jelenti, hogy nem található több csatorna a hálózaton.



31. A Szűrt TS adatsebességének mérése a VLC lejátszóban

3. Összefoglalás

Szakdolgozatom célja, helyi hálózatra történő digitális adás streamelése, azaz továbbítása volt, valamint az, hogy egy erre a célra dedikált vezérlő számítógéppel megvalósítható legyen a nézni kívánt csatorna kiválasztása.

A csatorna kiválasztás megvalósításához elengedhetetlen volt, hogy megismerjem a DVB-T adás-, azon belül is a Transport Stream csomag szerkezetét, hogyan, milyen formában kerülnek továbbításra a különböző műsorok adatai a stúdióktól egészen a felhasználókig, valamint e különböző adatok, milyen segédinformációkkal és hogyan kerülnek azonosításra a vételi oldalon.

A hálózaton történő továbbításhoz szükséges volt megismernem, milyen formában, szerkezeti felépítésben és mely protokollokat használva kerülnek kiküldésre a műsorok adatai a kliensek irányába, valamint hogy ennek megvalósítására, mely küldési mód alkalmazásával a leoptimálisabb.

A hálózati műsorszórás megvalósítását a CableWorld Kft. eszközeinek alkalmazásával végeztem. Mivel ezen eszközök egy egységes rendszert alkotnak, e rendszer tulajdonságait feltérképezve állítottam össze az ITPV rendszer alkalmazott konfigurációját. Beállításuk kizárólag számítógépes felületen végezhető, ezért szükséges volt megismernem az azok kezelésére alkalmas szoftverek működését a hálózati műsorszórás megvalósításához.

A megfelelő beállításokat elvégezve megkezdtem az ingyenes csatornák streamelését a felhasználók irányába. Ezt követően egy analízátor szoftver és egy médialejátszó segítségével méréseket végeztem. Az igen nagy adatsebességű, több műsort tartalmazó adatfolyam továbbítása zavartalan volt, minősége nem volt kifogásolható.

Az éppen nem nézett csatornák folyamatos hálózatba táplálása miatt ez a fajta műsorszórás ebből a szempontból gazdaságtalan, a hálózatot feleslegesen terheli, ezért, a streamelés adatsebességének csökkenése érdekében az adások szűrését valósítottam meg a CW-4900 Ethernet vezérlő kártyán található FPGA eszköz forrásprogramjának kiegészítésével, melyhez meg kellett ismerkednem az FPGA működésével valamint a programozásához használt egyik nyelvvvel, a VHDL-el.

A hardverelem belső felépítésének és működésének megismerését követően, egy szűrést végző összetevővel bővítettem ki annak forrásprogramját, majd ezzel a kibővített változattal programoztam fel a készüléket.

Ezt követően, az előző esetben használt analízátor-, és médialejátszó program segítségével méréseket végeztem az immáron a szűrt adatfolyammal kapcsolatban. Megvizsgáltam annak összetevőit és adatsebességét, a mérési eredményt kiértékelve látható volt, hogy a stream összetevőit csak az általunk kiválasztott csatorna audio- és videó összetevői alkották, tehát a csatornaszűrés sikerült, ezáltal a hálózat is csak e csatorna továbbításához szükséges adatsebességgel volt terhelve.

A médiatartalmak valós idejű megjelenítése elengedhetetlen a felhasználói elégedettség megtartása szempontjából. Az FPGA által nyújtott lehetőségek tökéletesen megfelelnek erre a célra. A gyors és párhuzamos működést biztosító eszköz e tulajdonságait kihasználva érhetjük el a már fent említett real-time adatküldést, mindemellett alacsony fogyasztása nem terheli plusz költséggel a felhasználót. Mivel az eszköz operációs rendszerrel nem rendelkezik, ezért a lefagyás veszélye sem áll fenn működés közben, valamint ezzel elkerülhető a rendszer illegális hozzáférési lehetősége is.

A jelenlegi megvalósítással a csatornaváltás egy vezérlő számítógéppel lehetséges, amely segítségével a VHDL kódot - az adott csatorna paramétereivel - módosítva érhető el a nézendő program kiválasztása. Ehhez szükséges azonban a forrásprogram megléte, a telepített fejlesztő környezet és a programozáshoz szükséges platform kábel. Hogy a nézni kívánt műsor meghatározása ne igényelje ezeket a feltételeket, fejlesztési lehetőségként felhasználó barát megoldás lenne egy erre alkalmas kezelőfelület létrehozása, mellyel a csatorna kiválasztás lehetőségét szoftveresen, számítógépes felületről lehetne megvalósítani.

4. Summary

My thesis objective is to stream a digital transmission on a local network so that the selection of required channels can be done with a dedicated controlling unit, a computer. To select a channel first I had to learn about the DVB-T broadcast and the Transport Stream package methods in it. With that I could understand how the studios stream their media to the subscribers and how the endusers identify the channels from the added information to the broadcast.

To reproduce a stream on the network I had to understand the formats, protocols the user agents use to communicate. There are more methods to broadcast so I had to find the most optimal solution for that. I used the equipment of CableWorld Kft. to maintain a broadcast. These devices make a unified system so knowing about its configuration I could build a working IPTV configuration. Only a computer can change the parameters in this setup so I had to handle the softwares which can control that.

After the right configuration was set I started to broadcast media free to the endusers. After that I made measurements with the help of an analyser and media player softwares. The stream which contained several channels at the same time had high transmission speed, the quality of the broadcast was unexceptionable.

This method isn't the best solution because the network is busy because of the traffic it handles. All the channels that are not used are loading the network unnecessarily. To reduce the speed of the data stream I applied a filter using an FPGA of the CW-4900 Ethernet adapter. To do so I had to learn the basics of FPGA and its programming language the VHDL.

After this update I made measurements again and the difference could be seen with the analyser program. I examined the components and the speed of the stream. The dataflow only contained the selected channels media content so that the filter was applied successfully. The network also was loaded with only this content.

The real time visualization of different media contents are indispensable to keep the endusers satisfaction high. The solution provided by the FPGA matches this purpose. The fast parallel operation can provide his real time broadcast and because of

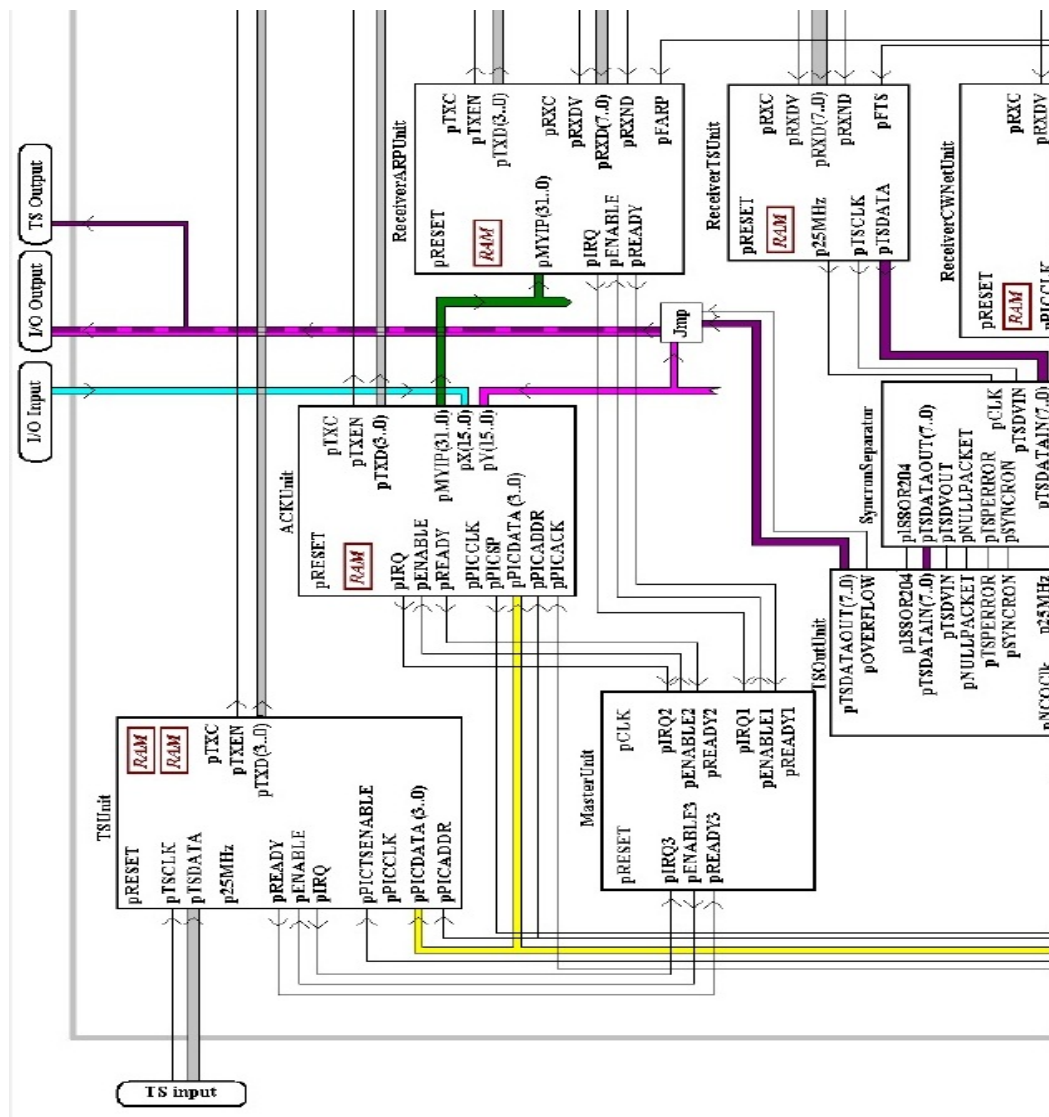
the low consumption of the device this application doesn't raise the expenses of the service to the users.

Because there is no operating system on this device the operation is freeze free. This solution also gives a protection against gaining illegal access to the media contents.

The selection of channels is performed by the controlling computer which provides the VHDL code with the modifications of channel configuration. In order to achieve this operation the source code, the installed development environment and the platform cable are necessary. Selecting a channel is also require these elements so creating an interface to the endusers which control these parts remotely can be a development to upgrade the system. With that the users could select from the channels via a software.

Mellékletek

1. számú melléklet – Részlet az FPGA blokkvázlatából



2. számú melléklet – A PID szűrést végző összetevő forrásprogramja

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity cSynchronSeparatorZ is
  Generic ( Position          : std_logic_vector(7 downto 0) := X"B0"
    );
  Port ( pRESET              : in std_logic;
        pTSCLK              : in std_logic;
        pTSDATAIN           : in std_logic_vector(7 downto 0);
        pTSDATAOUT          : out std_logic_vector(7 downto 0)
    );
end cSynchronSeparatorZ;

architecture Behavioral of cSynchronSeparatorZ is

  type tState is (Waiting, Trying, Locked);

  signal sTSDATA          : std_logic_vector(7 downto 0);
  signal sPIDL            : std_logic_vector(7 downto 0);
  signal sPIDH            : std_logic_vector(7 downto 0);
  signal sTSDATAOUT: std_logic_vector(7 downto 0);
  signal s188OR204        : std_logic;
  signal sLockCounter     : std_logic_vector(2 downto 0);
  signal sCounter         : std_logic_vector(7 downto 0);
  signal sFILTER: std_logic;
  signal sFPID1  : std_logic_vector(12 downto 0);
  signal sFPID2  : std_logic_vector(12 downto 0);
  signal sFPID3  : std_logic_vector(12 downto 0);
  signal sFPID4  : std_logic_vector(12 downto 0);
  signal sFPID5  : std_logic_vector(12 downto 0);
  signal sFPID6  : std_logic_vector(12 downto 0);
  signal sFPID7  : std_logic_vector(12 downto 0);
  signal sFPID8  : std_logic_vector(12 downto 0);

  begin

    pTSDATAOUT <= sTSDATAOUT;

    Separate : PROCESS(pRESET,pTSCLK)
      variable vState : tState;
      BEGIN

        if pRESET = '1' then

          vState := Waiting;
          s188OR204 <= '1';
          sFPID1 <= "0001111101001"; -- PID: 1001

```

```

sFPID2 <= "0001111101010"; -- PID: 1002
sFPID3 <= "0001111101100"; -- PID: 1004
sFPID4 <= "0001111110000"; -- PID: 1008
sFPID5 <= "0001111101011"; -- PID: 1003
sFPID6 <= "0001111101101"; -- PID: 1005
sFPID7 <= "0000000000000"; -- PID: 0000 PAT
sFPID8 <= "0001111101000"; -- PID: 0018 PMT
sFILTER <= '0';

elsif (pTSCLK' event and pTSCLK = '1') then
  sPIDL <= pTSDATAIN;
  sPIDH <= sPIDL;
  sTSDATA <= sPIDH;

case vState is

when Waiting =>
  sTSDATAOUT <= sTSDATA;
  if sTSDATA = X"47" then vState := Trying;
    sCounter <= X"00";
  end if;

when Trying =>
  sTSDATAOUT <= sTSDATA;
  sCounter <= sCounter+1;
  if (sCounter = X"BB" and sTSDATA = X"47") or      -- 188 format
     (sCounter = X"CB" and sTSDATA = X"47") then -- 204 format

    vState := Locked;
    sLockCounter <= "001";
    if sCounter = X"BB" then s188OR204 <= '1';
      else s188OR204 <= '0';
    end if;
    sCounter <= X"00";
  elsif sCounter = X"CB" then
    vState := Waiting;
  end if;

when others =>
  sCounter <= sCounter+1;
  if (sCounter = X"BB" and s188OR204 = '1') or      -- 188 format
     (sCounter = X"CB" and s188OR204 = '0') then -- 204 format
    sCounter <= X"00";
    sTSDATAOUT <= X"47";
    sFILTER <= '0';
    if sTSDATA = X"47" then -- Synchron OK
      if sLockCounter /= "110" then sLockCounter <= sLockCounter+1; end if;

      if ( sPIDH (4 downto 0) & sPIDL = sFPID1 ) or
         ( sPIDH (4 downto 0) & sPIDL = sFPID2 ) or
         ( sPIDH (4 downto 0) & sPIDL = sFPID3 ) or

```

```

( sPIDH (4 downto 0) & sPIDL = sFPID4 ) or
( sPIDH (4 downto 0) & sPIDL = sFPID5 ) or
( sPIDH (4 downto 0) & sPIDL = sFPID6 ) or
( sPIDH (4 downto 0) & sPIDL = sFPID7 ) or
( sPIDH (4 downto 0) & sPIDL = sFPID8 ) then
    sFILTER <= '0';
else
    sFILTER <= '1';
end if;
else -- Synchron Error
sLockCounter <= sLockCounter-1;
if sLockCounter = "001" then vState := Waiting;
end if;
end if;
else
    if sFILTER = '1' then
        case sCounter is
            when X"00" => sTSDATAOUT <= sTSDATA or X"1F";
            when X"01" => sTSDATAOUT <= X"FF"; -- NP PID
            when X"02" => sTSDATAOUT <= sTSDATA;
            when X"03" => sTSDATAOUT <= X"20"; -- szóköz
            when X"04" => sTSDATAOUT <= X"4D"; --M
            when X"05" => sTSDATAOUT <= X"45"; --E
            when X"06" => sTSDATAOUT <= X"47"; --G
            when X"07" => sTSDATAOUT <= X"59"; --Y
            when X"08" => sTSDATAOUT <= X"45"; --E
            when X"09" => sTSDATAOUT <= X"52"; --R
            when X"0A" => sTSDATAOUT <= X"49"; --I
            when X"0B" => sTSDATAOUT <= X"20"; -- szóköz
            when X"0C" => sTSDATAOUT <= X"4F"; --O
            when X"0D" => sTSDATAOUT <= X"45"; --E
            when X"0E" => sTSDATAOUT <= X"2D"; -- kötőjel
            when X"0F" => sTSDATAOUT <= X"41"; --A
            when X"10" => sTSDATAOUT <= X"4D"; --M
            when X"11" => sTSDATAOUT <= X"4B"; --K
            when others => sTSDATAOUT <= X"FF";
        end case;
    else
        sTSDATAOUT <= sTSDATA;
    end if;
end if;
end case;
end if;
end Behavioral;
END PROCESS;

```

3. számú melléklet – A szűrés tesztelésére létrehozott tesztfájl

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY SyncSepZ IS
END SyncSepZ;

ARCHITECTURE behavior OF SyncSepZ IS

    COMPONENT cSynchronSeparatorZ
    PORT(
        pRESET : IN  std_logic;
        pTSCLK  : IN  std_logic;
        pTSDATAIN : IN  std_logic_vector(7 downto 0);
        pTSDATAOUT : OUT std_logic_vector(7 downto 0)
    );
    END COMPONENT;

    --Inputs
    signal RESET : std_logic;
    signal TSCLK : std_logic;
    signal TSDATAIN : std_logic_vector(7 downto 0);
    --Outputs
    signal TSDATAOUT : std_logic_vector(7 downto 0);

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: cSynchronSeparatorZ PORT MAP (
        pRESET => RESET,
        pTSCLK => TSCLK,
        pTSDATAIN => TSDATAIN,
        pTSDATAOUT => TSDATAOUT
    );

    TSCLK_process :process
    begin
        TSCLK <= '1';
        wait for 100 ns;
        TSCLK <= '0';
        wait for 100 ns;
    end process;

```

```

RESETPROCESS: process
begin
  RESET <= '1';
  wait for 150 ns;
  RESET <= '0';
  wait for 3000 us;
end process;

TSDATAINPROCESS: process
begin
  TSDATAIN <= X"47";
  wait for 200 ns;
  TSDATAIN <= X"00";
  wait for 200 ns;
  TSDATAIN <= X"64";
  wait for 200 ns;
  for i in 0 to 184 loop
    TSDATAIN <= X"13";
    wait for 200 ns;
  end loop;
end process;

END;

```

4. számú melléklet – Fénykép az alkalmazott összeállításról

Irodalomjegyzék

- [1] <http://ahrt.hu/hu/tartalmak/digitalis-musorszoras.phpMyAdmin=dOBya%2C0qm0s97Vz-9CQk-Tizm68>
- [2] WALTER Fischer: A digitális műsorszórás alapjai. Bp.: Typotex Kiadó, 2005
ISBN 963 216 857 7
- [3] http://cableworld.hu/downloads/cwh_35.pdf
- [4] ANDREW S. Tanenbaum: Számítógép-hálózatok. Bp.: Panem Kiadó, 2013
ISBN 978 963 545 529 4
- [5] http://www.uni-obuda.hu/users/sandor.tamas/Begyazott_rendszerek/beagy_fpga_20140312.pdf
- [6] http://mzsola.iit.uni-miskolc.hu/cae/docs/fpga_rp.hu.html
- [7] <http://mzsola.iit.uni-miskolc.hu/cae/docs/pld1.en.html>
- [8] Dr. Hosszú Gábor, Dr. Keresztes Péter: VHDL-alapú rendszertervezés. Bicske: Szak Kiadó, 2012
ISBN 978 963 986 324 8
- [9] http://www.cableworld.hu/downloads/data_sheets/4875p-m.pdf
- [10] http://www.cableworld.hu/downloads/user_manuals/principal_help_2m.pdf