



Budapest Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Szélessávú Hírközlés és Villamosságtan Tanszék

TDK dolgozat

**Műholdfedélzeti kísérlet mérés-adatgyűjtő
egységének fejlesztése**

készítették:

Gugyin Adrián

(O0D7X7)

Szabó Dávid

(KDKLUY)

konzulens:

Dr. Csurgai-Horváth László

Budapest
2011.

Tartalomjegyzék:

1	Bevezetés.....	3
1.1	A dolgozat témájának rövid ismertetése	3
1.2	A dolgozat felépítése.....	4
2	A projekt szervezeti- és elméleti háttérének ismertetése	5
2.1	Az ESEO küldetés.....	5
2.2	Az ESEO LMP plazmadiagnosztikai kísérlet	6
2.2.1	Fizikai alapok	6
2.2.2	A szonda kivitelezése	8
2.3	Az űrbéli működés által támasztott speciális követelmények	9
2.4	Az FPGAs és mikrokontrolleres technológiákról általában.....	10
2.5	A mérésadatgyűjtő rendszer alapvető felépítése és működése.....	12
3	A digitális kártya fejlesztési lépcsőinek részletes bemutatása	15
3.1	A főbb feladatok ismertetése.....	15
3.2	Az adatgyűjtő rendszer konkrét megvalósítása	16
3.2.1	A digitális kártya végleges kapcsolási rajzának ismertetése.....	16
3.2.2	Az adatgyűjtő modul tápáramkörének elkészítése és bemérése.....	18
3.3	Az áramkör felélesztési lépései	20
3.4	Az FPGA áramkör tesztelése egy egyszerűbb logikai tervvel	27
3.5	FPGA áramkörbe ágyazott logikai terv részletes ismertetése.....	29
3.5.1	A modul elemei	29
3.5.2	A mikrokontroller core.....	30
3.5.3	Az APB busz, az APB perifériák és az APB alrendszer	31
3.5.4	A címdekóder elkészítése a memóriák és a külső címtartományba ágyazott perifériák kezeléséhez	34
3.6	Memóriaillesztési kihívások.....	36
3.7	A mikrokontroller felélesztése: az első működő program	38
3.8	A címdekódoló tesztelése assembly kóddal és logikai analízátorral.....	42
4	A digitális kártya funkcionális tesztelése	46
4.1	A külső perifériák tesztelése	46
4.1.1	A 16 bites D/A konverter tesztelése	47
4.1.2	A 8 bites D/A konverter tesztelése	47
4.1.3	A 8 bites A/D konverter tesztelése	48
4.1.4	Az UART és az FTDI chip tesztelése	49
4.2	A teljes kártya összetett működésének tesztelése	50
5	Összegzés	55
6	Köszönetnyilvánítás	56
7	Irodalomjegyzék.....	57
	MELLÉKLETEK	58

1 Bevezetés

1.1 A dolgozat témájának rövid ismertetése

Ennek a dolgozatnak a témája elsősorban annak a munkának a bemutatása, mely során társammal kifejlesztettünk egy univerzális, FPGA vezérelt adatgyűjtő egységet a tanszéken működő ESEO (European Student Earth Orbiter) kutatócsoport számára.

A tanszéki ESEO csoport diákokból áll, akik egy nemzetközi műhold-tervezési projekt bizonyos tervezési feladatait valósítják meg. A diákfejlesztéseket a tanszék úrkutató csoportjának számos tapasztalt oktatója és kutatója támogatja.

A csoport egyik jelenlegi kutatási témája egy műholdfedélzeti, plazmadiagnosztikai mérőműszer, a Langmuir Probe (LMP) kifejlesztése. Ez a műszer alapvetően egy analóg és egy digitális egységből tevődik össze.

Jelen TDK dolgozat szerzőinek az volt a feladata, hogy megtervezzék a digitális mérés-adatgyűjtő áramkör FPGA-ba illesztett központi vezérlőegységét, amely a fedélzeti kísérlet analóg jeleinek mérését, tárolását és továbbítását irányítja. Fontos feladat továbbá a szonda előfeszítő feszültségének az előállítása is. Mivel a műszer analóg részének megtervezése nem a mi feladatunk volt, ezért ebben a dolgozatban a mérés fizikai háttéréről csak olyan mélységben fogunk írni, amely számunkra a hardver eddigi fejlesztéséhez szükséges volt.

Mivel az általunk tervezett eszköznek ürkörülmények között majd működnie, ezért a tervezés során számos speciális követelményt vettünk figyelembe.

Amikor bekapcsolódtunk a csoport munkájába, a digitális vezérlő-modul blokkvázlat szinten már rendelkezésre állt, amely a munkánk kiindulási specifikációjaként szolgált.

A vezérlőkártya tervezésében a legnagyobb kihívás az FPGA-ba ágyazott logikai terv elkészítése, bemérése és a működtető firmware megírása, amely így a tervezés során a fő feladatunk volt.

Az FPGA-hoz kapcsolódó egysége kapcsolási rajzának véglegesítését és a nyomtatott áramkör megtervezését konzulensünk és Kocsis Gábor, az úrkutató csoport tagja végezte.

A nyomtatott áramkör elkészülte után részt vettünk az LMP kísérlet adatgyűjtő áramkörének felélesztésében, lépésenként bemértük és kipróbáltuk az alkatrészeket. Ebbe a folyamatba beletartozott mind az alkatrészek hardveres tesztelése, mind az alacsony szintű tesztelő szoftverek megírása. A digitális adatgyűjtő áramkör felépítését és működését részletesen egy későbbi fejezetben ismertetjük. Az áramkörnek 2 D/A átalakítót kell vezérelnie a szonda előfeszítő feszültségének beállításához, majd az ennek hatására kialakuló áramértéket kell mérnie egy A/D átalakítóval az LMP műszer mérőerősítőjének kimenetén. A rendszer lelkét egy Actel gyártmányú FPGA képezi, amelyben egy 8051-es mikrokontrollernek az ún. "soft-core"-jét helyeztük el. Ez vezérli az adatgyűjtési folyamatot és a mért adatokat CAN buszon keresztül továbbítja a műhold központi számítógépének, az On Board Data Handling egységnek (ODBH).

A projekt célja nem kizárólag az LMP kísérlet céljainak megfelelő rendszer kifejlesztése volt. Az elkészült berendezés alapját képezi egy univerzális, az alkalmazott FPGA és beágyazott technológia révén könnyen módosítható, üreszközökön jól használható mérés-adatgyűjtő rendszernek. Ez a rendszer a tanszéken fejlesztés alatt álló további üreszközön is jól alkalmazható, gyorsan testre szabható berendezés lesz.

1.2 A dolgozat felépítése

A dolgozat második fejezetében ismertjük a projekt szervezeti- és elméleti hátterét. Először az ESEO programról beszélünk, melynek keretében a munkánkat végeztük, majd áttérünk a műholdfedélzeten elvégzendő LMP plazmadiagnosztikai kísérlet rövid fizikai- és elektronikai hátterének leírására. Ezután bemutatjuk a mérésadatgyűjtő-rendszer alapvető felépítését, és tisztázunk néhány alapvető fogalmat az FPGA-s és a mikrokontrolleres technológiákról. A fejezet végén azt fogjuk bemutatni, hogy miért jelent nehezebb feladatot egy majdan űrbéli működésre szánt áramkör megtervezése, mint egy hagyományos földi példányé.

A harmadik fejezetben a digitális kártya fejlesztői lépcsőit fogjuk bemutatni. Először ismertetjük az általunk kapott részletes feladat-specifikációt. Ezután az adatgyűjtő kártya kapcsolási rajzát és a kártya teszteléséhez készített tápáramkört mutatjuk be. A fejezet további részében áttekintjük azt a folyamatot, amely során felélesztettük a kártyát. Ezután bemutatjuk, hogyan próbáltuk ki, hogy az FPGA működőképes-e egy egyszerűbb logikai tervvel. Ezután következik a legfontosabb alfejezet, mely az FPGA-ba beültetett és kipróbált véglegesen kész logikai tervet ismerteti. Azért ezt tartjuk a legfontosabb alfejezetnek, mert ebben van leírva, hogy ténylegesen hogyan építettük fel az FPGA-ba ágyazott logikai rendszerünket. Ezután egy rövid epizód következik, amely a memória-illesztési nehézségeinket írja le. Ezt követően bemutatjuk a mikrokontroller első kipróbálását egy négyszögjel-generátor programmal, majd egy nagyobb lélegzetű alfejezet következik, melyben a címdekódoló logikai analizátorral és fejlesztői környezettel való teszteléséről szól.

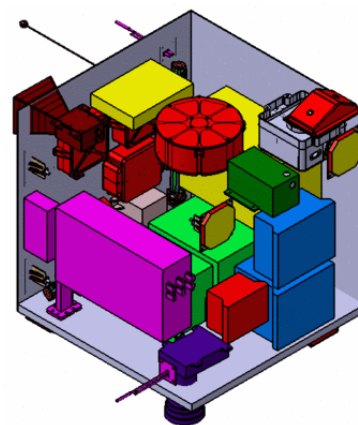
Az utolsó előtti fejezetben bemutatjuk az elkészült hardvert működés közben, a hozzá készült alacsony szintű szoftverrel együtt. Ez a tesztprogram a beállított mintavételi frekvenciával mintát vesz egy analóg jelből az A/D konverter segítségével, majd a D/A konverterre és az RS232 portra folyamatosan továbbítja a mért értékeket. Ez a tesztciklus az adatgyűjtő minden lényeges pontját ellenőrzi valós futási körülmények között. A rendszer helyes működését - a megfelelő műszerek segítségével - konkrét mérési eredményekkel igazoltuk.

Az utolsó fejezetben végül megköszönjük azoknak a munkáját és támogatását, akik segítségünkre voltak a projekt során.

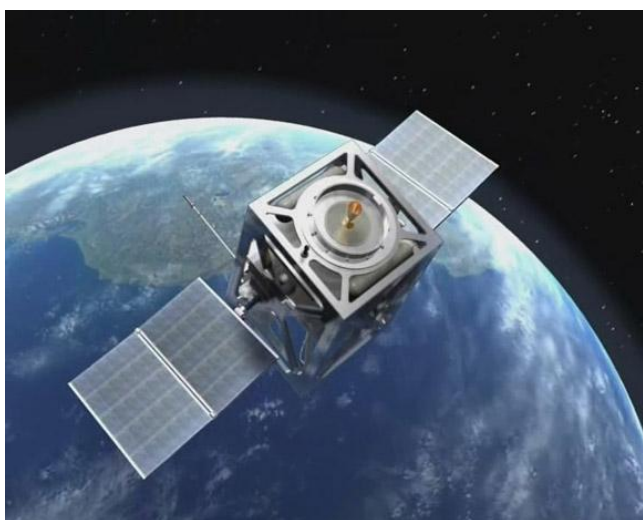
2 A projekt szervezeti- és elméleti háttérének ismertetése

2.1 Az ESEO küldetés

Az Európai Űrügynökség (ESA) által meghirdetett European Student Earth Orbiter nevű programban egy egyetemi hallgatók által megtervezendő és megépítendő műhold fejlesztése történik (1. ábra)[1]. Az ESEO egy alacsony földköri pályán (LEO – Low Earth Orbit) keringő kisműholdas misszió. Fejlesztése, integrációja és tesztelése európai egyetemi hallgatók által történik az ESA oktatási osztályának projektjeként. Az ESEO műhold Föld körüli pályáján fényképeket készít, sugárzási szinteket mér és technológiákat tesztel jövőbeli oktatási célú műholdas missziók számára. Az ESEO a harmadik missziója az Európai Űrügynökség oktatási célú műholdas programjának, mely az SSETI (felbocsátva: 2005-ben) és a YES2 (2007) programokban gyűjtött tapasztalatokat is felhasználja. A projekt jelenleg a B2 tervezési fázisban van, és Európa 13 egyetemének több mint száz diákja dolgozik rajta. Úgy tervezik, hogy az ESEO 2012-ben fog az alacsony Föld körüli pályára állni. A misszió nem titkolt célkitűzése ellátni a hallgatókat értékes és kihívást jelentő gyakorlati tapasztalatokkal az űrprojektek terén, hogy a jövőben így járuljon hozzá a jól kvalifikált űrmérnöki munkaerő-utánpótláshoz.



1. ábra: Az ESEO műhold



2. ábra Az ESEO műhold fantáziaterve [2]

Fizikai Kutatóintézet (KFKI) és az MTA Geodéziai és Geofizikai Kutatóintézet (GGKI) szakemberei.

Az űrbéli rendszer elsődleges kivitelezője a Carlo Gavazzi Space (CGS) nevű cég. Ez a vállalat biztosít - az ESA oktatási irodájával koordinálva - profi rendszerszintű-, és különleges technikai támogatást a projekt kivitelezése során az európai egyetemi fejlesztői csoportoknak. A következő részben vázlatosan ismertetjük a mi csoportunk által gondozott kísérletnek az elméleti háttérét és a kísérletet megvalósító műszer alapvető felépítését.

A Budapesti Műszaki Egyetemen az ESEO laborban három alrendszer konstrukciója történik [2]. Ezek egyik eleme az energiaellátó rendszer, Electrical Power System (EPS), illetve két műholdfedélzeti kísérlet: a Langmuir-szonda, Langmuir Probe (LMP) plazmadiagnosztikai kísérlet, továbbá a kozmikus sugárzást és annak élettani hatásait vizsgáló 3 dimenziós űrdozimetriai teleszkóp (röviden TriTel) vezérlő és adatgyűjtő egységeinek fejlesztései. Mindhárom alrendszernek megvan a maga fejlesztői csoportja a közösségen belül. A fejlesztésben támogatást nyújtanak a BME Űrkutató csoportjának tagjai, továbbá a Központi

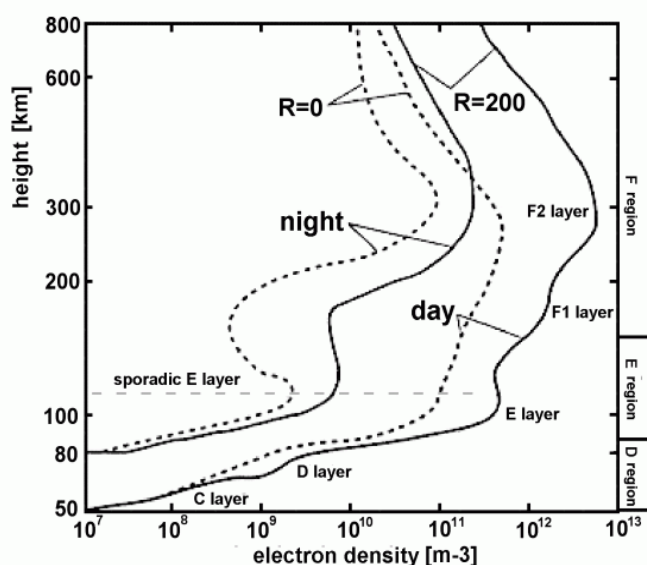
2.2 Az ESEO LMP plazmadiagnosztikai kísérlet

2.2.1 Fizikai alapok

Az atmoszféra felső régiójában, különösen az ionoszféra 60 és 1000 km-ig terjedő rétegében és a plazmaszférában (1000 km-től 3-4 Földugárig), a molekulák egy bizonyos része ionizált, plazmaállapotban van, de a tér egészében kvázi-semleges. Az ionizáció fő forrása a Naptól származó elektromágneses (UV és afeletti frekvenciatartomány) és részecskesugárzás [3], valamint a kozmikus sugárzás ionizáló komponense. A Föld mágneses tere befolyásolja a töltött részecskék mozgását, melynek következtében a plazmában szabálytalanságok képződnek a mágneses erővonalak mentén. Ezek a szabálytalanságok befolyásolják az elektromos eszközök működését és a töltött részecskék sűrűsége fontos szerepet játszik a hőterjedésben, ezáltal hatással van mind a földi, mind az űrbéli időjárásra [5].

Az ionoszféra az atmoszférikus gázok O , O_2 , N_2 ionizációjával jön létre [6]. Az ionizáció mértékének egyik kvantitatív jellemzője a szabad elektronok térfogati sűrűsége. A Földtől távolodva az elektronsűrűség növekedése jellemző az ionoszférára, amelyben ez alapján különböző rétegeket különböztetünk meg (3. ábra). Jelentős eltérés mutatkozik a nappali és éjszakai eloszlásban, ami illusztrálja az szoláris sugárzás hatásának döntő szerepét. Az ábrán az R paraméter egy a naptevékenységre jellemző mutató, a havi szoláris index mediánja. Az elektronsűrűség tehát a napi aktivitással is erősen összefügg –

más szavakkal az elektronsűrűségből következtethetünk a naptevékenységre, ha a többi tényező értékét jól becsüljük meg. Fontos, hogy a kérdéses közeg plazmaparamétereit befolyásoló folyamatok nagy része időben determinisztikusan, periodikusan változik, így finom modelleket lehet felállítani a kísérletben kapott mennyiségek értelmezésére.



3. ábra: Az ionoszféra-elektronsűrűség vertikális profilja[6]

Az ionizált rétegeken reflektálódnak a rádiófrekvenciás elektromágneses hullámok, ami nagy távolságú földfelszíni rádióösszeköttetéseket tesz lehetővé. A rövidhullámú jelek terjedése szempontjából elsősorban a legfelső, F-réteg a fontos [3].

Az alcímben szereplő kísérlet célja az ESEO mintegy 520 km-es keringési magasságának megfelelő, az ionoszféra F2 rétegében elhelyezkedő, legsűrűbb plazma vizsgálata, paramétereinek mérése. Ennek kivitelezése lehetséges az Irving Langmuir Nobel-díjas kémikusról elnevezett műszerrel. A Langmuir szonda egy ionizált közegbe merülő fémes elektróda, amelyet a plazma elektronsűrűségének, elektronsűrűségének és potenciáljának meghatározásához használnak. A megfelelő kísérleti payload kódneve tehát a műszer elnevezéséből származik (LMP, Langmuir probe).

A plazmák a bennük fellépő kollektív tulajdonságok miatt egy bizonyos méret felett nem tárgyalhatók a kinetikus gázelmélet ideális modelljével [3]. E karakterisztikus méret a Debye-hossz, vagy -sugár: $\lambda_D = \sqrt{\varepsilon_0 kT / ne^2}$ (1)

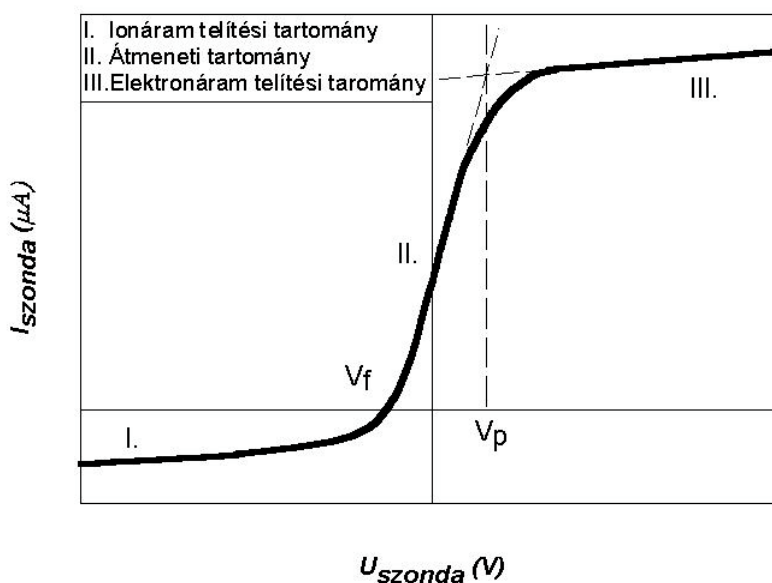
A képletben n az elektronsűrűség, T az elektronhőmérséklet, ε_0 a vákuum permittivitása, k a Boltzmann-állandó és e az elektron töltése. Ez a közelítés elhanyagolja az ionok hatását. Természetesen a világűrben a Debye-hossz széles tartományok között mozoghat tényezői függvényében. Az F ionoszférában ez az átlagos $10^{12}/\text{m}^3$ -es elektronsűrűségre és 10^3K -es hőmérsékletre 1 mm-es nagyságrendet jelent.

Egy kiterjedt plazma és anyagfelület határán az elektronok és ionok energiakülönbsége egy ionokban sűrűbb (emiatt pozitív töltéstöbblettel rendelkező) Debye-, vagy elektrosztatikus rétegnek, illetve –köpenynek (sheath) nevezett egyensúlyi töltéselrendezést hoz létre: az elektronok hőmérséklete legalább az ion- és elektrontömeg hányadosának négyzetgyökével arányos faktorral nagyobb, mint az ionoké, így a határfelület közelében a plazmából elektronok repülnek ki a felülete, ott relatív negatív töltéstöbbletet előidézve. Egy űrbéli objektum felületén tehát indukált töltés van, amely taszítja a távolabbi plazma elektronjait. A test felszíne az ún. lebegőpotenciál értékére töltődik fel, az eredő töltésáramlás ekkor zérus. A másik alapvetően fontos potenciál érték a plazmapotenciál, amelyen a töltések a termikus átlagsebességgel haladnak; ebben a térrészben (lényegében a plazma belsejében) nincs elektrosztatikus réteg.

A Langmuir szonda elektródája körül hasonlóan az azt hordozó műholdtesthez egy ilyen elektrosztatikus köpeny alakul ki, azonban ennek potenciálját előfeszítéssel változtatva a rajta kialakuló töltéseloszlást, illetve az egyensúlyi áramerősséget tudjuk befolyásolni. Ezt a viselkedést egy feszültség-áram karakterisztikával jellemezhetjük, amelyet Langmuir karakterisztikának is neveznek.

A Langmuir szonda elméleti karakterisztikája látható a 4. ábrán:

Szonda U-I karakterisztika



4. ábra – Az LMP szonda karakterisztikája [7]

A karakterisztikának három jellegzetes szakasza van [3].

I.: A detektor potenciálja negatív a plazmapotenciálhoz (V_p) képest, így a negatív töltésű részecskéket taszítja, a pozitív ionokat vonzza, a kialakuló pozitív töltésréteg elektrosztatikus tere a szonda potenciálját leárnyékolja. A kialakuló áramban ionok dominálnak, ez az ún. ionszaturációs régió, a hozzá tartozó negatívnak választott áramerősség az ionszaturációs áram. Növelve a detektor feszültségét csökken az elektronok taszítása, nő az elektronáram. A lebegőpotenciálon (V_f) éppen kiegyenlítődik a kétféle áram, 0 áramerősséget eredményezve. Mivel ez az érték lesz a műholdtest potenciálja is, ezt fogjuk referenciának tekinteni.

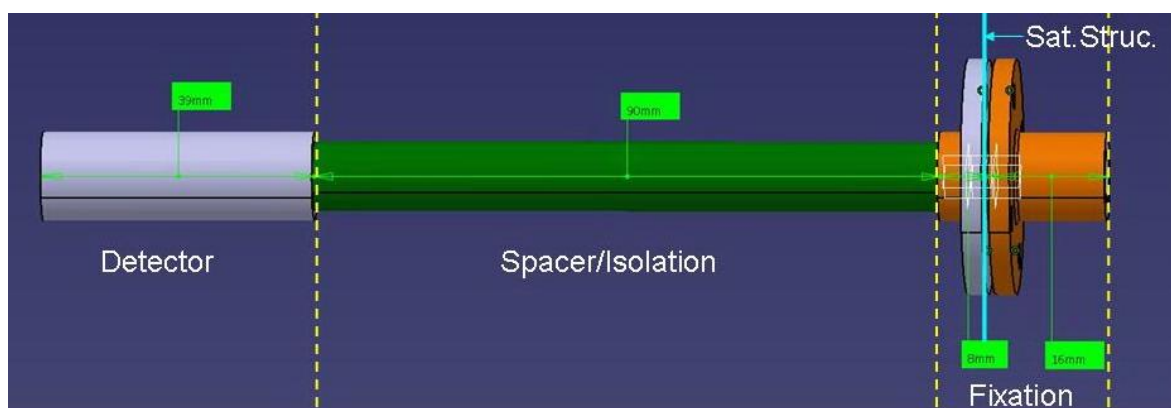
II.: A feszültség további növelésével az elektronáram nemlineáris függvény szerint nő, mivel az elektronok mozgékonyasága sokkal nagyobb az ionokénál. Ez az átmeneti régió. A plazmapotenciált elérve a szonda körüli töltött burok eltűnik, ekkor nincs elektrosztatikus vonzás, illetve taszítás, viszont az elektronok mozgása dominál azok nagyobb sebessége miatt.

III.: A detektor potenciálja ekkor magasabb, mint a plazmapotenciál, így a pozitív ionok nem tudják elérni a szonda felületét, az elektronáram pedig telítődik, ez az elektron-szaturációs régió.

A szonda működésének alapjául szolgáló elmélet a töltött részecskék Maxwell-i sebességeloszlását feltételezi, ezért a módszer csak olyan zavarmentes plazmákban alkalmazható, melyben ez a követelmény teljesül [3]. Minden zavaró hatást eliminálni kell. A karakterisztika függ az ionizált részecskék sűrűségétől, az elektronhőmérséklettől és a lebegőpotenciáltól (V_f). A szondaköpeny vastagsága Debye-hosszal egy nagyságrendű, ez azt jelenti, hogy az elektromos mező hatása csak egy bizonyos távolságig terjed ki és a plazma ezen túl zavarmentesnek tekinthető.

2.2.2 A szonda kivitelezése

Az LMP kísérlet hasznos terhe egy távtartó rúdra szerelt és a mérésszabályozó- és adatfeldolgozó egységhez elektromosan huzalozott hengeres szonda. A detektor anyaga titán, titán-nitrid öntettel. A vezérlőegység (LCB – LMP Control Box) és a szonda közötti kapcsolat egy triaxiális kábellel van megvalósítva. A vezérlőegységnek elő kell feszítenie a szondát, mérni az áramot, összegyűjteni, tárolni és továbbítani az adatokat.



5. ábra – az Langmuir szonda detektorának mechanikai vázlata [3]

A detektor egység (LDU – LMP Detector Unit) mechanikai vázlatán (5. ábra) kivehetők az említett egységek, a hengeres alakú detektor, a távtartó rúd (Spacer/Isolation) és a rögzítés az ESEO alvázához (Fixation).

A detektort függetleníteni kell a mőholdtest Debye-rétegetől. A becsőlt Debye-hossz a kőísérleti környezetben 5 és 40 mm között változik, ezért a távtartó hosszára 90 mm lett meghatározva. A detektorfej hengeres alakú, a mőhold méretéhez képest elhanyagolhatónak, viszont a minél nagyobb áramerősség elérése érdekében elegendően nagyoknak kell lennie. Ezen megfontolások alapján detektor méreteire 39 mm-es hossz és 19 mm-es átmérő lett meghatározva [3].

2.3 Az űrbéli működés által támasztott speciális követelmények

Mivel a tervezendő áramkörünk végleges (repölő) változatának ki kell majd állniuk - az űrbéli küldetésből fakadó és a mőholdra erős hatást kifejtő - speciális körülményeket, ezért az ilyen környezetben működő eszközöknek különleges minőségi és megbízhatósági feltételeknek kell megfelelniük. Ebben a fejezetben ezeket fogjuk röviden ismertetni.

- Ellenállás a mechanikai rázkódással és a sugárzásokkal szemben:

A felbocsátás során fellépő Pogo-oszcilláció, a közegellenállás miatt fellépő turbulencia és a gyorsulás rendkívüli mechanikai, a világűr viszonyai pedig sugárzási terhelést jelentenek egy áramköri panel számára. Emiatt az űrben elsősorban furatszerelt, kerámiatokozású IC-eket célszerű használni, amelyeket speciális forrasztanyaggal rögzítenek. A mechanikai sérülés esélye még így is fennáll. A különleges alkatrészek ára lényegesen meghaladhatja a hagyományos kivitelűekét. A probléma egyik megoldása az lehet, ha a digitális IC-k legnagyobb részét kiváltjuk egyetlen FPGA áramkörrel. A mai korszerű FPGA-kba könyvtári elemként be lehet fordítani a mikroprocesszorokat, például a 8051-es mikrokontrollert is. A kevesebb alkatrész mellett arra az előnyre is szert tehetünk, hogy elegendő az FPGA megbízhatóságát biztosítanunk és szükség esetén tesztelnünk, ami további megtakarítást jelent. Az FPGA-kból több megbízhatósági szintű változat is kapható. A legolcsóbb (kommersz) FPGA áramköröknél magasabb követelményeknek tesznek eleget az ipari kivitelű FPGA-k. Ezeknél jelentősen drágábbak a sugárzásnak is ellenálló FPGA-k. A sugárzásálló FPGA-k közül alapvetően 2 csoport közül választhatunk: a Rad-Tolerant és a Rad-Hard. A kettő közül az előbbinek kisebb a tűrőképessége, míg az utóbbi képes ellenállni az összes típus közül a legnagyobb sugárzásnak. Az ESEO mőholdon lesznek sugárzásálló FPGA-k, például ilyen - a szintén a mi csoportunkban fejlesztett - energia-ellátó rendszert vezérlő FPGA. Mivel az LMP kísérlet nem kritikus rendszer a mőholdon, az általunk megépítendő áramkörbe csupán az ipari szabványoknak megfelelő példány kerül.

Vannak azonban olyan áramkörök, amelyeket csak az áramkör FPGA-n kívüli részén helyezhettünk el. Ekkor viszont, ha az áramkört sugárzásállóvá szeretnénk tenni, elengedhetetlen, hogy minden alkatrészből ún. Rad-Hard változatot válasszunk, amelyek a hagyományos alkatrészek sugárzás-álló technológiával készült változatai.

- Bipoláris technológiák alkalmazása:

Az űrbéli magasabb sugárzási körülményeket a bipoláris áramkörök felépítésüknel fogva alapvetően jobban tűrik, ezért célszerű ilyeneket alkalmazni, amikor erre lehetőség van. A bipoláris technológia következtében azonban általában magasabb áramfelvételre kell számítani. Tipikus példa erre a bipoláris memóriák alkalmazása a rendszer-kritikus programok tárolása esetén.

- Bipoláris memóriák alkalmazása:

A memóriák nagyon fontos részei az adatgyűjtő áramköröknek, így a mi digitális adatgyűjtőnk működésében is kritikus szerepük van. Alapvetően alkalmazható az az elv, hogy minden alkatrészből sugárzásálló változatot kell beszerezni. Az ESEO mőholdon az LMP

műszer nem lesz folyamatosan áram alatt, ezért a processzor által futtatandó programkódot kikapcsoláskor "nem felejtő" memóriákba kell helyezni. Logikus megoldás lenne, hogy a programkódot könnyen újraindító EEPROM-ban vagy FLASH memóriában tároljuk. A rendszer szempontjából kritikus programok (operációs rendszer) tárolására azonban ezek nem használhatók. Ez okból a műholdakon a gyakorlatban "nem felejtő" memóriaként bipoláris PROM-okat alkalmaznak. Az SRAM-oknak létezik rad-hard változatuk. A mi esetünkben az első kísérleti áramkörünk még csak a földi működésre lett tervezve, ezért megengedhető volt, hogy EEPROM-ot és normál SRAM-okat helyezzünk el, a repülő példányban azonban mindenképpen bipoláris PROM-ot és speciális SRAM-ot kell majd használni.

- Energia-takarékosság:

Az műholdon az áramkörök számára rendelkezésre álló energia korlátozott. Általában jellemző, hogy a nagy energiát felvevő eszközöket és alkatrészeket csak addig kapcsoljuk be, amíg feltétlen szükség van rájuk. A mi rendszerünkben egy bipoláris memória, PROM fog elhelyezkedni. A PROM-nak lényegesen nagyobb az áramfelvétele, mint az SRAM-oknak, ezért a gazdaságosság kedvéért azt a megoldást választottuk, hogy műszer bekapcsolása után közvetlenül a programkód átmásolja magát az egyik - erre a célra fenntartott - SRAM-ba, átadja annak a vezérlést, majd lekapcsolja a PROM tápellátását. Ezáltal jelentős fedélzeti energia takarítható meg.

Az űrbéli speciális körülmények elleni védekezés módszereinek ismertetése után az elvégzett mérnöki munkát és annak folyamatát mutatjuk majd be, most előbb azonban tisztázunk néhány alapvető fogalmat a mi általunk alkalmazott technológiákról, amelyek segítségével később megvalósítottuk a tudományos méréshez szükséges digitális vezérlést.

2.4 Az FPGAs és mikrokontrolleres technológiákról általában

A programozható logikai áramkörök világa véleményünk szerint az egyik legérdekesebb és leggyorsabban fejlődő technológiai ágazat, amellyel egy egyetemi villamosmérnök hallgató kapcsolatba kerülhet a tanulmányai során. Ugyanakkor ezeket a rendszereket az ipar számos területén alkalmazzák, hiszen a minket körülvevő modern világban sok terméknek a vezérlését oldják meg ilyen eszközökkel. Ilyen áramkörök vannak ma már a mobiltelefonokban, a háztartási készülékekben, az autókban, műszerekben, és végül - de nem utolsósorban - az űreszközökben is. Egy beágyazott logikai áramkört sokféle módon és sokféle különböző szinten lehet megvalósítani. Általában, ha logikai tervezésről beszélünk mindenkinek a logikai kapukból és a *flipflop*okból megépített áramkörök jutnak először az eszébe. Tulajdonképpen ezt az irányvonalat képviseli a programozható áramkörök egyik nagy családja: az FPGA-k.

Az FPGA rövidítés jelentése: Field Programmable Gate Array, azaz programozható logikát tartalmazó tömb. Az FPGA-k logikai kapukat, flip-flopokat, memóriákat és esetleg egyéb kiegészítő áramköröket is tartalmazó eszközök, melyekben az elemek összehuzalozásáról és felhasználásáról a felhasználó - majdnem - szabadon dönthet úgy, hogyha feltölti a programját az FPGA-ba.

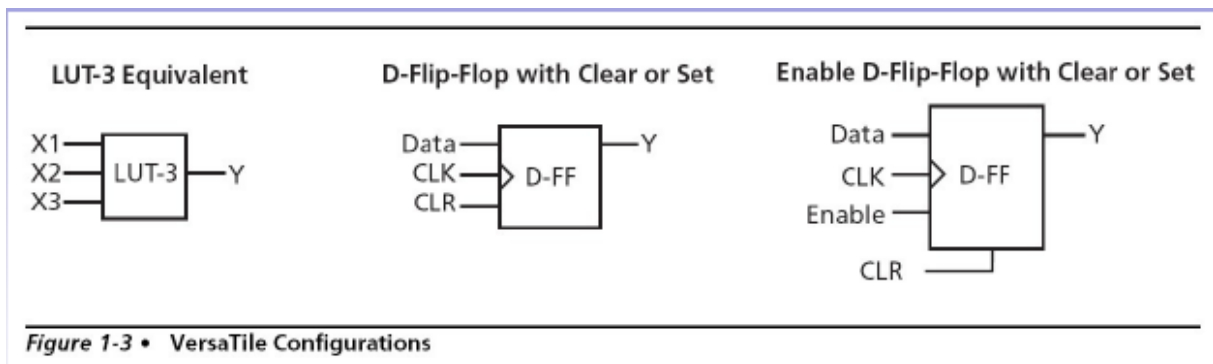
Granularitás [8]:

1. fine-grain: Finom felosztású (szemcsésségű) rendszerek
2. coarse-grain rendszer: Nagyobb modulokból felépített rendszerek

Általában ezeket ma már blokkonként keverve használják, vagyis bizonyos célfunkciókat specializált coarse-grain blokkokkal (memória, DSP, PLL, DCM stb.) valósítanak meg, míg az általánosabb működést fine-grain cellák használatával.

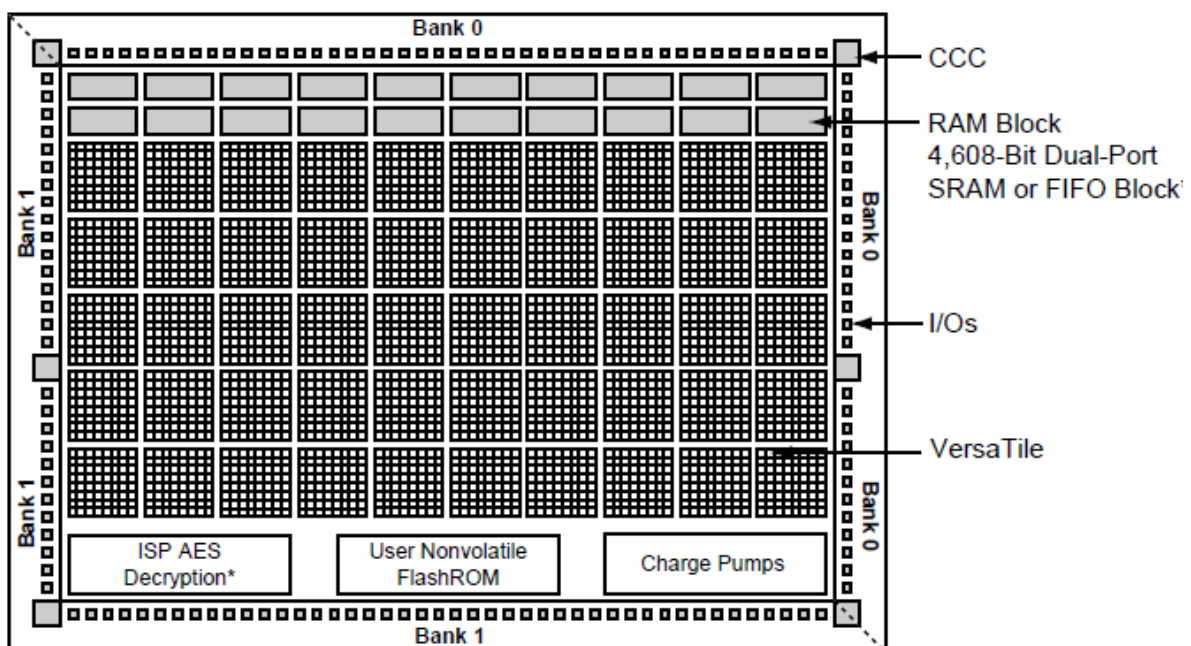
Egy FPGA-ban a legkisebb logikai egységek a logikai cellák [8]. Mi a munkánk során az Actel cég ProASIC3 családjába tartozó A3P400 típusú FPGA-jával dolgoztunk. Az FPGA

piacon elterjedtebb pl. a Xilinx cég Spartan3 típusa, melynek a logikai blokkjai úgymond "hagyományosabb" felépítésűek. Ennek a típusnak a logikai blokkjai az úgynevezett slice-ok, melyek fő alkotóelemei a LUT-ok és a flip-flopok. A LUT-ok (Look Up Table) tulajdonképpen néhány bemenetű kombinációs hálózatok megvalósítására alkalmasak, míg a flip-flopokat tipikusan regiszterként szokták felhasználni. Ehhez nagyban hasonlít az általunk használt Actel gyártmányú FPGA-k felépítése. Ezen áramkörök alapegységei az úgynevezett - *VersaTile*-ok. Ezek olyan logikai blokkok, melyekből - a bennük lévő (összeköttetések) kapcsolók és visszacsatolások beállításával rugalmasabban állíthatunk össze - nekünk szükséges - kombinációs és szinkron sorrendi hálózatokat. Tipikusan 1 versatile-ből 1 LUT-ot vagy 1 D flip-flopot lehet összeállítani. Ilyen megvalósítások láthatók a:



6. ábra Logikai alapcellák képzése "versatile"-okból (Actel) [8] [9]

Általában az FPGA-k az elemi logikai cellák mellett tartalmaznak még memóriákat, egyéb logikai céláramköröket, illetve tartalmaznak nem-logikai áramköröket is, ahogyan azt Az egyéb logikai céláramkörök közé sorolhatók a DSP blokkok, Hard Processzor IP -k, Hard periféria IP-k. A nem-logikai erőforrások közé sorolhatók a huzalozás, az órajel-szétosztó hálózat és az órajel-kezelő egységek, illetve az analóg blokkok.



Az FPGA-kban a felhasználó megadhatja a kívánt működés leírását egy hardverleíró nyelv segítségével (VHDL, Verilog, stb.), vagy megadhatja egy kapcsolási rajz szerkesztővel is grafikus formában. Nagyon fontos megjegyezni, hogy az FPGA-kban lehetőség van egyidejűleg több hardveres műveletet is végezni, mivel az FPGA-k nem csak szekvenciális utasítás-végrehajtásra képesek. (Verilogban az egymással párhuzamosan lefolyó értékadásokat "<=" jellel kell megadni, míg a szekvenciális értékadásokat "=" jellel).

A címben szereplő másik fontos eszközcsalád a mikrokontrollerek családja. Ezek olyan programozható logikai áramkörök, amelyeket szekvenciális programvégrehajtás megvalósításra fejlesztettek ki. A mikrokontrollerek tehát olyan processzorok, melyeknek a program-memóriáját a felhasználó szabadon meghatározhatja a saját céljainak a megvalósításához (pl. flash alapú feltöltés).

Ezeknek az eszközöknek fontos tulajdonsága, hogy általában a processzor mellett az áramkör tokjában megfelelően illesztett céláramkörök (pl. perifériák) helyezkednek el. A perifériáknak az a feladatuk, hogy segítsenek a processzornak a külvilággal való kommunikációban. A mikrokontrollerek egy jelentős része *Harvard* architektúrával rendelkezik, amely a - *Neumann* architektúrával ellentétben - azt takarja, hogy a processzor külön memóriából olvassa ki a programkódot és az adatokat. A mikrokontrollerek szekvenciálisan végrehajtják a programmemóriájukban elhelyezett kódot.

Az eddigiekben ismertetett mindkét áramkörtípust gyakran használják úgynevezett beágyazott rendszerekben. Ezek általában olyan mérnöki alkalmazások, ahol az áramkörnek speciális környezetben kell mérési, adatfeldolgozási és vezérlési feladatokat ellátni. Az áramkörökhöz szenzorok, beavatkozók, vagy más processzorok kapcsolódhatnak, s ezáltal komplex funkciókat lehet megvalósítani velük. A mikrokontrollerekből vannak 8/16/32 bites típusok is. A kevésbé komplex felépítésű 8/16 bites IC-k jól használhatók egyszerűbb mérési feladatokhoz, míg bonyolultabb jelfeldolgozási feladatokhoz már érdemesebb egy 32 bitest, vagy egy FPGA-t választani.

Ebben a szakaszban két alapvető áramkörtípust ismertetünk. Ezt a rövid áttekintést azért tartottuk fontosnak, hogy a dolgozatunk témájában kevésbé járatosak is képet kapjanak az általunk használt technológiák alapvető működéséről.

2.5 A mérésadatgyűjtő rendszer alapvető felépítése és működése

Az LMP szonda mérés-adatgyűjtő egységének architektúrális diagramja a 8. ábrán látható. Ebben a fejezetben ennek az alrendszernek a működését fogjuk ismertetni. A teljes mérésvezérlő elektronika alapvetően 2 részre osztható, melyek külön panelon kerültek megvalósításra:

- analóg elektronikát megvalósító panel (szaggatott vonalon kívül)
- digitális adatgyűjtést végző panel (szaggatott vonalon belül)

A mi munkánk a projektben főleg a digitális adatgyűjtő panel tervezéséhez és megépítéséhez kapcsolódik, ezért az analóg elektronikát alkotó elemeket csak nagyvonalakban mutatjuk be. A digitális adatgyűjtő kártya megvalósításának részleteit a 3. fejezet ismerteti.

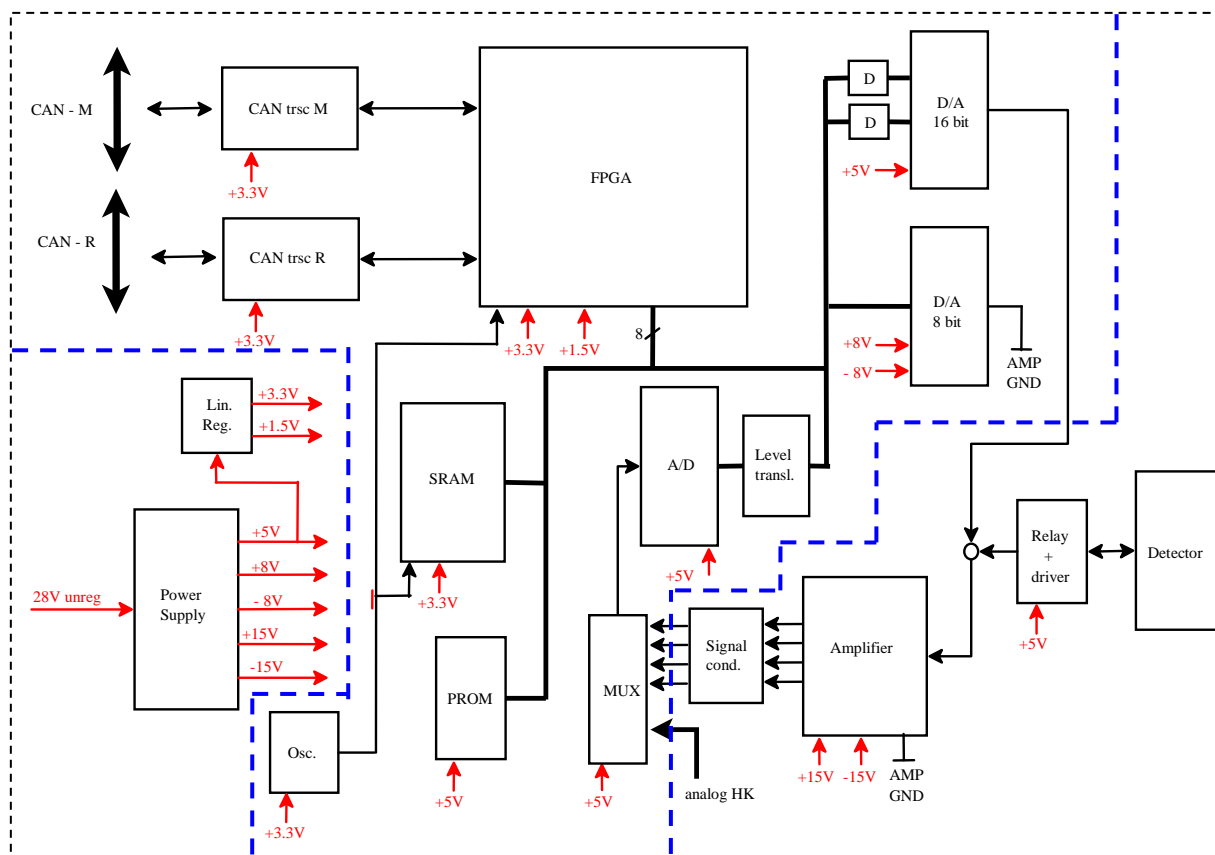
A műholdon lévő fedélzeti számítógép az autóelektronikában is elterjedt CAN buszra csatlakozik, tehát az adatgyűjtő egységnek ezzel kompatibilisnek kell lennie. A CAN kommunikációt az FPGA-ban implementált CAN core irányítja. Két melegen tartalékolt CAN adó-vevő képezi a CAN sínhez tartozó hardver interfészt. Itt jegyezzük meg, hogy a most elkészült áramköri modellben csak egy CAN interface került megvalósításra, a tartalékolt rendszer majd a repülő példány kialakítása során kerül be az áramköri tervbe. A műhold

fejlesztésének jelenlegi fázisában sajnos az OBDH-t tervező lengyel csoport még nem adta meg a CAN kommunikáció protokollját. Ezért az FPGA-ban egy UART core-t is elhelyeztünk, amellyel a tesztelés jelenlegi, első fázisában egy egyszerű, saját magunk által definiált protokoll szerint kommunikálhatunk az OBDH-t helyettesítő PC-vel.

A mérés-adatgyűjtő az FPGA-n kívül még A/D és D/A átalakítókat, különböző kiegészítő áramköröket, perifériákat, illetve memória áramköröket tartalmaz. A három memória áramkör közül az első egy egyszer programozható bipoláris PROM memória. Ez a panel bekapcsolásakor lép működésbe és a rendszer indítási folyamat során a tartalma átmásolódik az egyik statikus ramba, melyet mi ún. operatív tárnak neveztünk el. (A kezdeti blokkdiagramon ez az SRAM még nem szerepelt, ezért nincs rajta még a 8. ábrán.) A másik SRAM áramkör a mérési adatok átmeneti tárolására szolgál, amelyeket már részben feldolgoz és tömörít a központi adatgyűjtő egység. A mérést vezérlő logika, a memóriacímzéshez használt dekóder és néhány egyéb logikai elem - az eredeti tervek szerint - az FPGA-n belül került megvalósításra, ezáltal a készüléknek kevesebb alkatrészt kell tartalmaznia, és így nagyobb megbízhatóságú lesz.

A fedélzeti elektronikán a legfontosabb egységekből két példány (Main/Redundant - fő- és redundáns) lesz a jobb hibatűrés érdekében. A Langmuir detektor (LDE) egy különálló eszköz a műhold alvázához rögzítve, míg az összes többi egység a Langmuir vezérlődoboz (LCB) része. A kettő közötti elektromos összeköttetést egy triaxiális kábel valósítja meg, ahogyan azt a 2.2.2 szakaszban ismertettük. Amikor a műszer nem aktív vagy belső tesztek hajtódnak végre, a detektor és az elektronika különválasztásáról egy bistabil relé gondoskodik. Adatgyűjtés során a 4 csatornás nagy pontosságú analóg erősítő egység méri a detektoron folyó áramot. Egy jelkondicionáló áramkör szükséges az erősítő kimenetén, hogy elegendő feszültséget biztosítson az adatfeldolgozó elektronika számára. A négy analóg csatorna a HK (housekeeping) adatvonalakkal együtt multiplexálva van, jeleiket A/D konverzió után az FPGA-ban implementált 8051 processzornak dolgozza fel. Az A/D konverterre 0-2.5 volt közötti jelértékek érkeznek, melyek a jelkondicionálás révén változnak át a mérőerősítő kimeneti max. +- 7V-os feszültségeiből ezeké az értékekké. A processzor által kiértékelt és tömörített adatok az SRAM memóriamodulban tárolódnak, amely FIFO pufferként szolgál a CAN busznak a megfelelő protokoll szerinti kommunikációhoz. A memória 16 másodperc körüli tudományos adatmennyiség tárolására elegendő arra az esetre, ha a CAN busz ideiglenesen nem elérhető az LMP egység számára.

Az adatfeldolgozás mellett a 8051-es mag felelős az előfeszítő generátor irányításáért, amely egy 8 bites D/A konverter segítségével állítja be a feszültséget az erősítő földjén -7 és +7 V között. Ezen túlmenően az erősítő ofszetjét minden mérési ciklus után (azaz másodpercenként) újra kell számolni - a hőmérséklet változása alapján - és kompenzálni kell azt a 16 bites D/A átalakító segítségével (8. ábrán felső D/A). Mivel a processzor adatbusza csupán 8 bites, ezért a 16 bites D/A elé 2 db. 8 bites D latchet kellett elhelyezni (8. ábrán a D jelű blokkok). Ezekbe külön kell beírni a D/A bemenetén mintavételezésre szánt 16 bites érték felső- és alsó bájtyát.



8. ábra – LMP architektúrális diagram

A mérés során várhatóan másodpercenként 128 db 8 bites mintát vesz az A/D átalakító az LMP detektora felől. Ez egy földköri pályamenti körül-fordulásonként (ill. 95 percenként) 1.5 Mbyte-nyi adatmennyiséget jelent, amelyet továbbítani kell a fedélzeti számítógép, illetve a kommunikációs egység felé. Ezen feladatoknak az elvégzéséhez számításaink szerint elegendő a 8051-es architektúrájú processzormag alkalmazása (16 bites címzés 8 bit széles adatbusszal). A feladat memóriaigénye is átlagosnak (kicsinek) számít. A 8051 ezen túl kompatibilis az ESA PPL-jében szereplő alkatrésszel, de ennek itt az FPGA-s implementáció miatt nem lesz szerepe. A mikrokontroller mellett még helyet kapott az FPGA-ban a memóriaillesztéshez szükséges címdekódoló és további funkciókat ellátó egységek, pl. CAN vezérlő, időzítők illetve egy watchdog egység, amely szükség esetén újraindítja a rendszert, ha az bizonyos idő elteltével sem válaszolna vagy hibásan működne.

A műszer a szabályozatlan energiabuszról kap tápellátást [3]. Egy kapcsolóüzemű típusú átalakító (flyback converter, transzformátort tartalmazó Buck-Boost konverter) +/-15V, +/-8V, illetve +5V kimeneti feszültséget szolgáltat az elektronika számára. A kimenetek a bemenetről galvanikusan el vannak választva. További +1.5V-os és +3.3V-os feszültségek szükségesek az FPGA táplálásához. Ezeket egy lineáris feszültségszabályozó segítségével a konverter +5V-os kimenetéről állítjuk elő.

3 A digitális kártya fejlesztési lépcsőinek részletes bemutatása

Ebben a részben bemutatjuk az LMP digitális egység tervének és kivitelezésének részleteit és számot adunk a fejlesztési munka részfeladatairól, illetve a bemérések, tesztek során szerzett tapasztalatokról és eredményekről.

3.1 A főbb feladatok ismertetése

- Az ESEO LMP kísérlet mérés adatgyűjtőjének funkcionális és strukturális terve a munkánk kezdetekor rendelkezésre állt, ez alatt értendő a digitális kártya kapcsolási és beültetési terve, az alkatrészek megválasztása és a már korábban specifikált működési követelmények. Ezen specifikációk, **tervek, adatlapok áttanulmányozása** lényeges követelményt jelentett a későbbiekben, ugyanis a mi feladatunk volt a **programozható logikába ágyazott mikrokontrolleres vezérlő egység és a szükséges belső perifériák, logikák és azok összeköttetéseinek megtervezése**. Az egyik feladatunk tehát logikai tervezés volt, amelyre többféle tervbeviteli mód állt a rendelkezésünkre attól függően, hogy éppen milyen funkcióra volt szükség a konfigurációnkban.
- A kártya hétféle egyenáramú tápfeszültségének előállítására egy teszt-tápegységet készítettünk. A **tápáramkör tervezése, megépítése, bemérése** is a mi feladatunk volt. A tápáramkorról a szükséges áramellátás az adatgyűjtő kártyára tűskesoros csatlakozókon, szalagkábelrel van biztosítva.
- Miután a szükséges alkatrészek beültetésre kerültek és a tápellátás biztosítva volt, ellenőriznünk kellett az alapvető funkcionalitást, a működés szempontjából kritikus jelek meglétét. Ezt a lépésről lépésre történő folyamat volt a **rendszer** elemeinek **élesztése**, illetve **bemérése**.
- A további fejlesztések szempontjából elengedhetetlen volt a **hardver és szoftver környezet**, amelyben a fejlesztést végezzük, illetve a beméréshez használt **műszerek** (logikai analizátor, oszcilloszkóp) megfelelő **konfigurációja**, felügyelete és esetleges frissítése/újratelepítése/javítása, mivel ahogyan egy ilyen összetett fejlesztési munkánál nagy valószínűséggel előfordulhattak és elő is fordultak meghibásodások.
- Fontos még megemlíteni, hogy bizonyos, főleg saját készítésű **logikai modulok szimulációja** is szükséges volt a működésük ellenőrzésére.
- Az **inkrementális tervezés** során mindig egy-egy újabb **modult** próbáltunk az eddigi, már működőképes tervbe beilleszteni, annak működését (esetleg többféle módon is – hardveres/szoftveres) **tesztelni**. Ehhez az általunk használt 8051 utasításkészletű **assemblyben készítettünk szoftvereket**, illetve ún. on-chip debuggerrel fértünk hozzá a processzor regisztereihez, illetve a memóriához.
- A fejlesztés kezdeti szakaszai sok nem várt nehézséget tártak elénk és sokszor kényszerültünk **nem dokumentált paraméterek beazonosítására**. Gyakran volt szükség **kreatív problémamegoldásra**, új (esetleg máshol már létező, de általunk még nem ismert) **módszerek ki-, vagy megtalálására**.

A következőkben arról lesz szó, hogyan sikerült ezen feladatokat (sokszor hosszú és küzdelmes kitérők, trial-and-error sorozatok árán) megoldanunk.

3.2 Az adatgyűjtő rendszer konkrét megvalósítása

3.2.1 A digitális kártya végleges kapcsolási rajzának ismertetése

A bevezetésben található blokkvázlat egyes elemeinek megfeleltethetőek az LMP digitális paneljére beszerelt, illetve a későbbiekben beültetésre kerülő alkatrészek. Ezeknek az alkatrészeknek a listája látható az 1. táblázatban.

Alkatrész típusa	Alkatrész funkciója	Tokozás
Actel ProASIC3 A3P400	FPGA	PQFP208
AD7224KR-18	8 bit D/A	SOIC18
AD7822BR	8 bit A/D	SOIC20
AD7846JNZ	16 bit D/A	PDIP28
AT28C256-15PU	parallel EEPROM	PDIP28
BS62LV256SCG70	SRAM	SOP28
FTDI UM232R	USB/soros interface	
TXS0108E	szintillesztő	TSSOP
SN74HC573ADW	latch	SOIC20
TLE2022CD	műveleti erősítő	SOIC8
CD74HC4067M	analóg multiplexer	SOIC24
CSX750F 16.000MHz	oszcillátor	SDM
LM336BZ-2.5	2.5 V ref.	TO92
LM336BZ-5	5 V ref.	TO92
LM810M3-4.63	reset áramkör	SOT-23-3

1. táblázat – alkatrészlista

Jelenleg a panel az AT28C256 32 kByte-os EEPROM helyett egy AT28C64 8 kByte-os EEPROM-ot tartalmaz, ennek kapacitás is elegendő volt a tesztprogramok számára.

A panel végleges kapcsolási rajza a mellékletben látható (M1.1/3; 2/3; 3/3).

Az első oldalon (M1.1/3 ábra) található a memóriainterfész a 16 bites címbuszhoz tartozó két TXS0108 szintillesztővel (az FPGA 3.3V-os feszültségét konvertálja 5V-ra), a memória modulok (BOOT, OPERATIVE, STORAGE), továbbá a 16 bites D/A átalakító egység látható.

Az EEPROM (BOOT ROM), a 0x0000-0x7FFF címterületen helyezkedik el. A kerek címtartomány miatt elegendő az A15-ös címvezeték bekötése a memória /CE (Chip Enable) lábára. Az általunk használt 8 kByte-os IC 13 bittel címezhető, tehát az A14-es és A13-as címvonalak nem használtak, illetve a 0x2000-0x3FFF, 0x4000-0x5FFF, 0x6000-0x7FFF területeken is az alsó 8 kByte-os terület tartalmát lehet visszaolvasni. Az EEPROM írását engedélyező /WE (Write Enable) bemenetre a /DBG_WR jel van bekötve. Ezáltal az EEPROM-ot programozhatjuk az FPGA-ba ágyazott mikrovezérlő core debug interfészén keresztül, így nem szükséges külön EEPROM programozó. Az /RD (Read Enable) vezérlő bemenetre a /PSEN jel kerül, amely az utasításlehíváskor aktív.

A kapcsolási rajzon két SRAM modul látható: OPERATIVE, illetve STORAGE. Az első a 0x8000-es kezdőcímen érhető el. Az LMP szoftvere ebbe a modulba töltődik át a bekapcsolást követő folyamat során. Ennek az az oka, hogy a végleges modellben használt bipoláris PROM lényegesen nagyobb teljesítményt vesz fel az SRAM-okhoz képest, ezért a rendszerindítás során ebbe az operatív tárba másoljuk a kódot, majd a boot memória

tápfeszültségét lekapcsoljuk. Ezt az FPF2103 jelű IC segítségével végezzük. Ez az egység a PWR_ON jellel vezérelhető. A másik, 0x0000-0x7FFF területen található STORAGE RAM modul adatárként szolgál az LMP szoftver futása alatt, amelyben ideiglenesen tároljuk a mérések adatait.

A 16 bites AD7846JNZ típusú D/A átalakító segítségével a V_OFFSET feszültséget állítjuk elő (ezzel történik a mérőerősítő offset kompenzálása). Ez az egység szimmetrikus táplálású, +15V és -15V-os tápellátást igényel, valamint +5V-os referenciát, melyet egy feszültség-referencia áramkör biztosít. A felső és alsó byte az adatbuszról egy-egy 74HC573D latch-en keresztül írhatók be, amelyeket az LD_DAC16_H és LD_DAC_16_L jelekkel vezérlünk.

A kapcsolási rajz második oldalán (M1.2/3 ábra) található az LMP panel tápcsatlakozója. Mind a hét tápfeszültséghez (-15V, -8V,+8V, +15V, +5V, +3.3V, +1.5V) tartozik egy LED, amellyel ellenőrizhető, hogy megkapja-e a panel a szükséges tápfeszültségeket. Ezen a rajzon látható az analóg csatlakozó is, amelyre az LMP analóg egységének feszültségei kerülnek, illetve egy 4067D 16 csatornás analóg multiplexer, amelyről a 8 bites AD7822 A/D átalakító kapja a bemenetét jelkondicionáló erősítőn keresztül. Az A/D +5V-os tápfeszültséggel és +2.5V-os referenciával működik, utóbbit egy referencia-diódás áramkör állítja elő.

Itt látható még a 8 bites D/A egység kapcsolási rajza is, amelynek kimenetén jelkondicionáló erősítő állítja elő a V_BIAS feszültséget, amellyel a mérési ciklusok alatt történik az előfeszítés. A D/A -5V-os negatív tápfeszültségét egy lineáris feszültségszabályozó IC állítja elő a -8V-os tápfeszültségből.

A kapcsolási rajz második oldalán (M1.3/3 ábra) látható az LMP digitális paneljének központi egysége, az FPGA, amelyben a 8051-es CPU egység és a belső logikák, perifériák vannak implementálva. Az FPGA mellett a következő részek is kaptak még helyet: 16 MHz-es kristály oszcillátor (X1); státusz LED-ek (D2-D5); kézi kapcsolósor (S1-1-S1-4); LM809 RESET IC; TJA1040 CAN adó-vevő áramkör; továbbá szabadon felhasználható kivezetések az FPGA-ból tesztelési célokra (PAD1-PAD8). Ezek fontos szerepet játszottak a beméréskor, lásd később.

További áramkörök: UM232L FTDI UART/USB interfész, két JTAG interfész (az egyiket az FPGA programozásakor használjuk (JTAG1), a másik a felkonfigurált FPGA-ban a Core8051s processzor debug interfészéhez kapcsolódik, ezen keresztül tudjuk tesztelni a processzort számítógépről ugyanazzal a FlashPro3 eszközzel, amellyel az FPGA-t programoztuk. A debugger szoftver a PC-n egyúttal ezen az interfészen keresztül a programmemória írására is képes. Ezen az oldalon van még az adatbusz szintillesztője (TXS108E).

Az FPGA +1.5V-os és +3.3V-os, a JTAG, az oszcillátor és az FTDI modul +3.3V-os, a CAN adó-vevő +5V-os tápfeszültséget igényel.

Az FPGA lábkiosztása a tápfeszültségek és földpontok szempontjából kötött, a NYÁK-on pedig a kapcsolási rajzon jelölt portokra történnek a kivezetések.

Az FPGA jelei:

RESET – a tápfeszültség megjelenésekor újraindítja a rendszert.

Az oszcillátor kimenete egy 16 MHz-es órajel, ezt az FPGA megfelelő konfigurációjával módosítani tudjuk.

CPU_AD[0..15] – címkimenetek,

CPU_D[0..7] – adatbusz,

PWR_ON – az EEPROM ki-/bekapcsolását vezérlő jel,

/RD – adatmemória/periféria olvasás engedélyezés,

/PSEN – programmemória olvasás engedélyezés,

/WR – írás engedélyezés,

/GATED_WR – ezt rendszerindításkor használjuk, amikor az operatív RAM-ba töltjük át a programkódot.

/DBG_WR – ezt a jelet csak debug során használjuk a programmemória (EEPROM) írására.

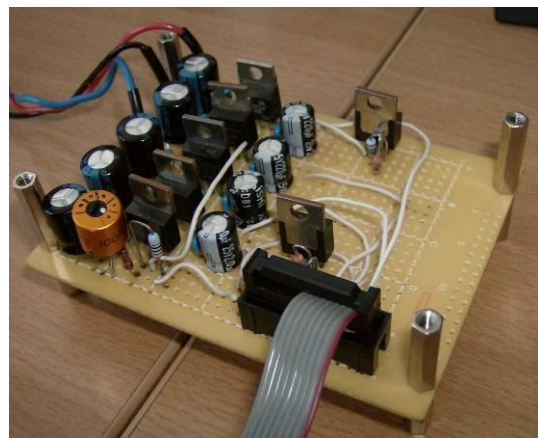
A további jelek egy része a perifériákat (A/D, 8, ill. 16 bites D/A, analóg multiplexer) vezérlik és funkciótól függően az FPGA-n belüli logika (pl. címdekódoló) vezérli őket. Ezeket a későbbiekben részletezzük.

3.2.2 Az adatgyűjtő modul tápáramkörének elkészítése és bemérése

Az adatgyűjtő modul felélesztéséhez mindenképpen szükséges volt elkészíteni egy olyan tápáramkört, amely képes a kártya működéséhez szükséges összes feszültségszint előállítására (9. ábra).

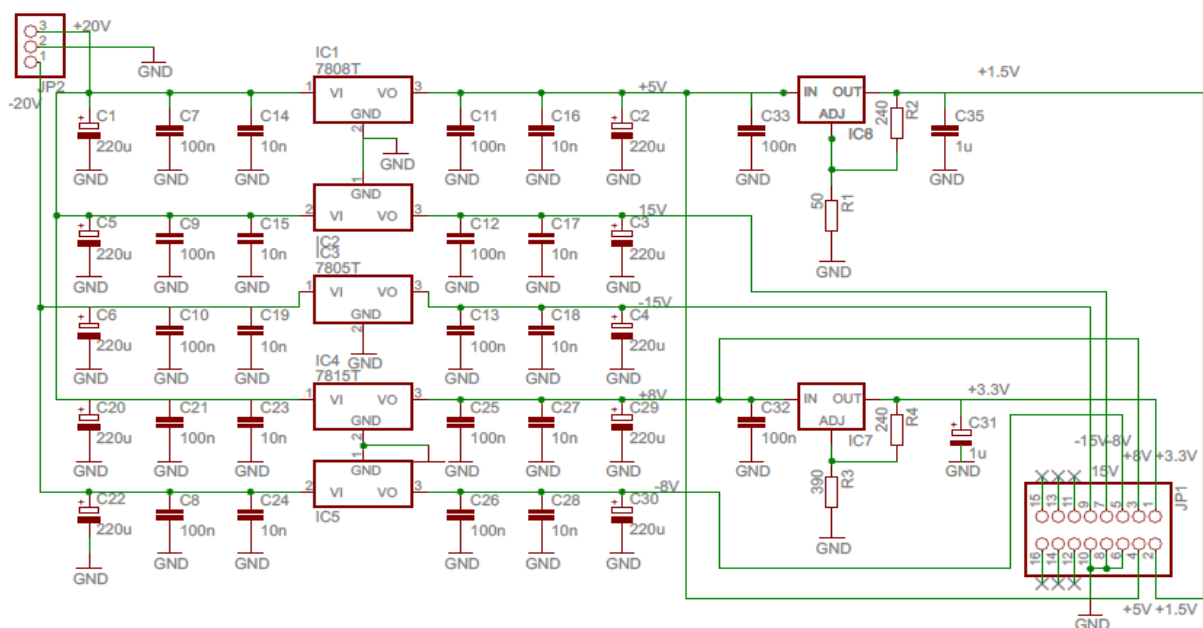
A következő felsorolásban ismertetjük, hogy a digitális kártyán a különböző alkatrészek működtetéséhez milyen feszültségek előállítására volt szükség:

- 16 bites D/A átalakító: +5V, +15V, -15V
- 8 bites D/A átalakító: +15V
- FPGA működéséhez: 3.3V, 1.5V
- 8 bites D/A kimenetén lévő műveleti erősítő: +8V, -8V



9. ábra : A tápáramkör

A digitális kártya működtetéséhez tehát az alábbi feszültségeket biztosítottuk: +15V, -15V, +8V, -8V, +5V, 3.3V, 1.5V. A tápáramkör kapcsolási rajzát a 10. ábra A tápáramkör kapcsolási rajzmutatja be:



10. ábra A tápáramkör kapcsolási rajza

Mivel a digitális áramkört minél előbb szeretnénk volna feléleszteni, egy viszonylag egyszerű, de könnyen megépíthető megoldás mellett döntöttünk. Az áramkört egy próba NYÁK-on, furatszerelt alkatrészekből építettük meg. A kívánt feszültségek előállítására stabilizátor IC-eket használtunk, áramkör táplálására pedig laboratóriumi tápegységet (FOK-GYEM TR-9178). A műholdon a 28V-os buszfeszültségből ugyanezeket a feszültségeket kell majd előállítani, de természetesen ettől eltérő megoldást kell majd alkalmazni.

A 78-as típusjelzésű áramkörök a pozitív feszültségeket állítják elő, míg a 79-es típusjelzésűek a negatív feszültségeket. A +3.3V-ot és az +1.5V-ot előállító modulokban LM317-es IC-t használtunk, melyeknek kimenő feszültségét ellenállásokkal állítottuk be.

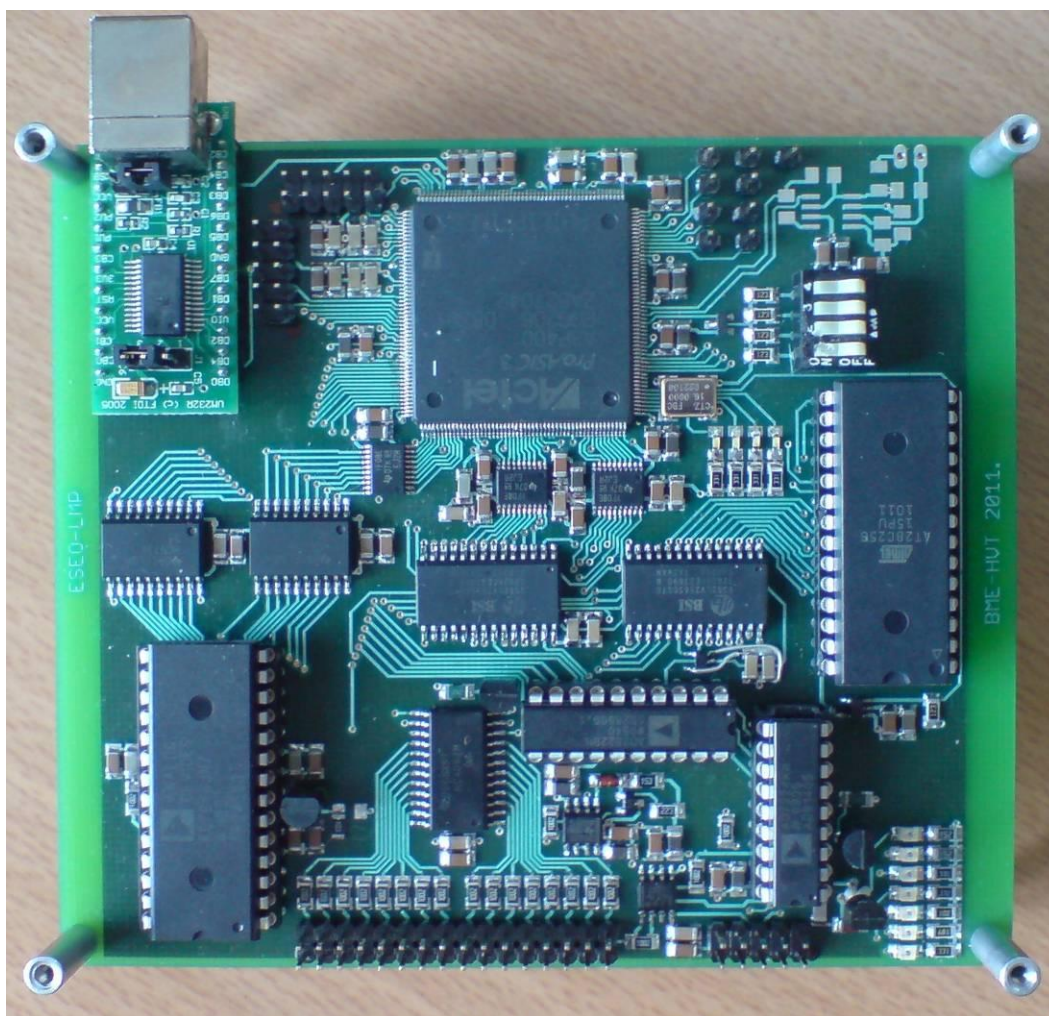
3.3 Az áramkör felélesztési lépései

Az LMP digitális paneljének fejlesztését a következő eljárás szerint végeztük, amely során mindig csak egy kisebb részegységet helyeztünk üzembe:

1. – alkatrész(ek) beszerelése a panelbe
2. – logikai tervezés (FPGA újrakonfigurálás)
3. – szoftver fejlesztés, debug
4. – bemérés, működés tesztelése, verifikációja

Az első három lépés közül sokszor csak az egyiket volt szükséges elvégezni.

Új alkatrész csak akkor lett beültetve, ha a régiekkel működést/kritikus paramétereket befolyásoló hibáktól mentesen működött a rendszer és/vagy szükséges volt az új alkatrész a következő teszthez. A készre szerelt panelt a **11. ábra** mutatja:



11. ábra : Az adatgyűjtő panel felülnézeti képe

A leggyakrabban a logikai tervet is módosítani kellett a tesztek elvégzéséhez. A bevezetőben már szó volt beágyazott rendszerekről, de csak érintőlegesen esett szó arról, hogy hogyan történik egy ilyen rendszer fejlesztése.

A legtöbb modern beágyazott rendszer alkalmazás-specifikus integrált áramköröket (ASIC), illetve a már tárgyalt FPGA-kat tartalmaz, amelyekhez többféle hardver leíró nyelvet (HDL-t) fejlesztettek ki, ilyen például a VHDL és a Verilog. Az előbbit eredetileg az Egyesült Államok kormányának irányításával alkották meg a nagysebességű integrált áramkörök dokumentációjára, a neve is innen ered (VHSIC hardware description language) az 1980-as években. A VHDL az Ada programnyelv szintaxisára épül és alkalmas az általa leírt logikai rendszerek szimulációjára. A hardver leíró nyelvek még nagyobb előnye, hogy felhasználható az automatizált elektronikai tervezésben – az ún. logikai szintézis módszerével egy jól kezelhető HDL leírásból előállítható a fizikai megvalósítás definíciója [11].

A hardver leíró nyelvek sokban hasonlítanak a programozási nyelvekhez – egyfajta végrehajtható specifikációk – azonban pontosabb ezeket a modellező nyelvek közé besorolni; a programnyelvek általában nem képesek a modell szimulációja során elengedhetetlen időskála kifejezésére. Ennek ellenére a legtöbb hardver funkció leírható velük, ezért jó néhány elterjedt programnyelvnek létezik hardver/rendszer leíró változata (SystemC, JHDL).

A programnyelvek és HDL-ek analógiájára az utóbbiak eszköztáiraiban sok mindent átvettek a szoftveres terminológiából. A fordító pl. megfeleltethető a szintézist végző programnak; ugyanúgy léteznek kódolási konvenciók, automatizált ellenőrzési módszerek, feltételes fordítási direktívák, elágazások, ciklusok, stb., a hardverfejlesztés során azonban sokkal körültekintőbben kell szerveznünk a blokkjainkat, ugyanis itt a nyelv parancsai párhuzamosan „hajódnak végre” és nem a programoknál megszokott vezérlési szál(ak) mentén.

A tervezéshez elsődlegesen a Verilog nyelvet használtuk, amely egy fokkal felhasználóbarátabb a VHDL-nél, szintaxisa egyszerűbb és rugalmasabb, sok közös vonása van a C programozási nyelvvel.

Az FPGA-kat gyakran használják programozható vezérlő egységként beágyazott processzorral, illetve mikrovezérlővel (MCU – microcontroller unit) felszerelve, amelyhez kész IP core-ok állnak rendelkezésre a gyártóktól, amelyek az FPGA tervbe egyszerűen beilleszthetők és a kezelői felülettel konfigurálhatók. A beágyazott vezérlők architektúrája, utasításkészlete gyakran megegyezik a műszaki gyakorlatban már elterjedt, nagy sorozatszámú gyártott modellekével, pl. az Intel 8051-es Harvard architektúrás mikrovezérlőjével, amilyennel mi is dolgozunk.

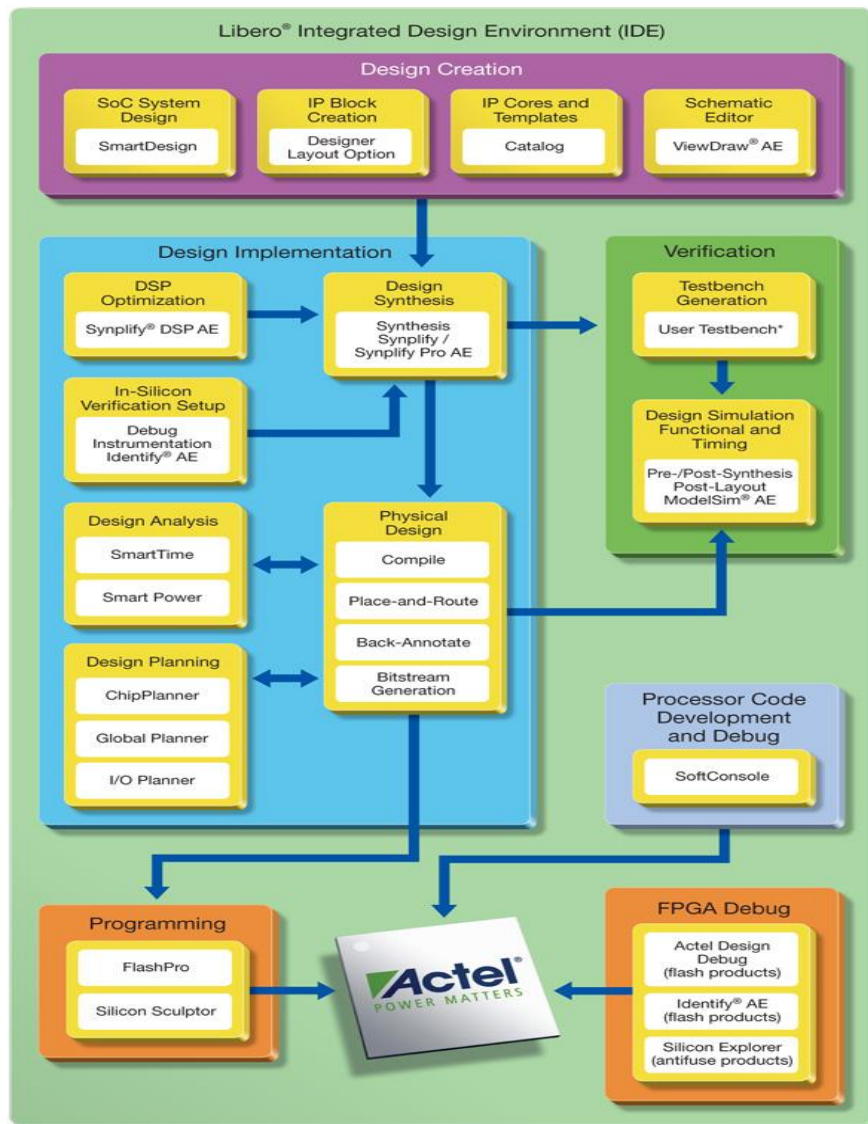
Az FPGA fejlesztőkörnyezetének ismertetése

A fejlesztést egy Actel ProASIC A3P400 típusú eszközön végeztük, amely egy kb. 400000 kapu komplexitású FPGA. A kivétel 208 lábú PQFP tokozású, maximum 194 I/O port felhasználó által definiálható [9].

A szoftver, amit a logikai tervezésre használtunk, az Actel Libero® integrált fejlesztőkörnyezete, melynek a 9.0-s verziójával dolgoztunk. Ez egy eszközlánc jellegű fejlesztőkörnyezet, amely több, a tervezés különböző fázisait segítő CAD programból áll, összefogva egy közös projektkezelő program által.

A Libero hatékonyan támogatja a gyártó által rendelkezésre bocsátott IP-k beépítését, a funkcionális blokkokkal történő tervezést, a tervezési szabályok betartásának ellenőrzését (DRC – design rules check) és automatikusan elvégez beállításokat, amelyek szükségesek a szoftvercsomag elemeinek használatához, tehát elég felhasználóbarát.

Alternatívaként a Mentor Graphics professzionális szoftvereivel is dolgozhattunk volna, de kevesebb munkával és kevesebb lehetséges kompatibilitási problémával járt az Actel-es környezet használata (másképpen az Actel Libero ingyenesen elérhető bárki számára). Az FPGA tervezési folyamata Libero-ban nyomon követhető a következő ábrán:

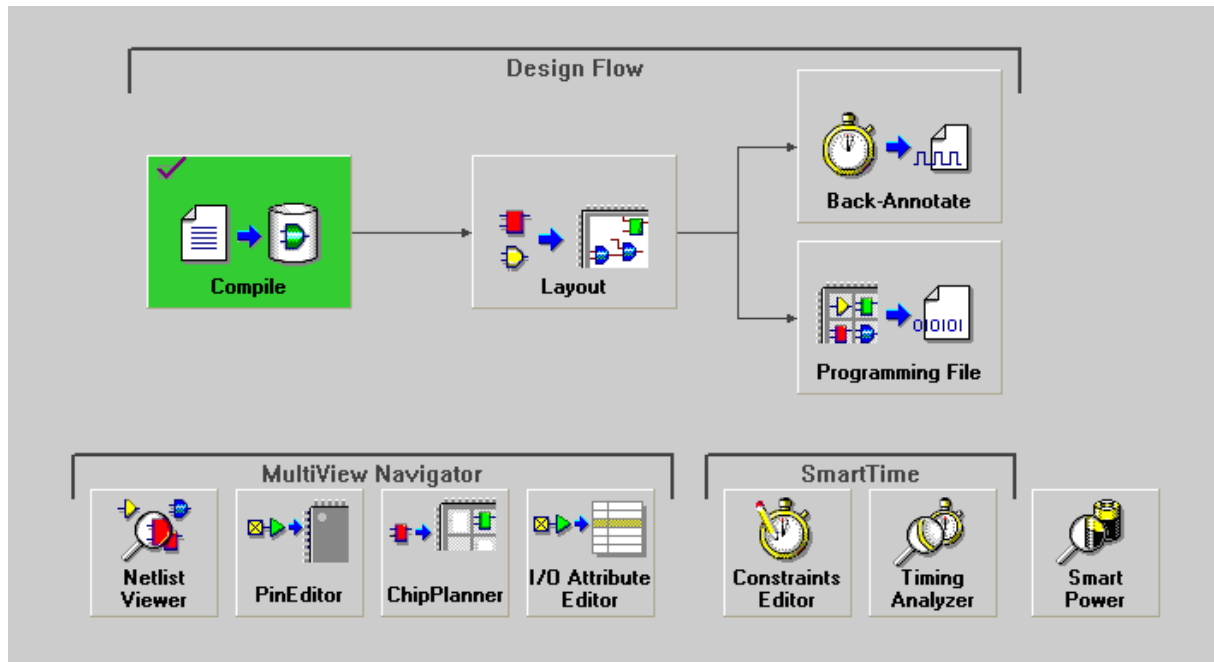


12. ábra - FPGA tervezési folyamat [12]

Sok esetben nem hajtottunk végre minden lépést, ami az ábrán szerepel, mert nem volt szükséges. A lényegesebb tervezési lépések eseteink többségében a következők voltak:

- 1. Logikai terv bevitele** (design entry) – Ez történhet HDL specifikációval, kapcsolási rajzzal, makrókkal, IP core-ok beillesztésével (Catalog), általunk létrehozott blokkok /modulok összekötésével (az áttekinthetőség szempontjából is célszerű volt a tervünket modularizálni, hierarchikusan lebontani) – ehhez a Libero SmartDesign grafikus felületét használtuk, amellyel egyúttal szintetizálható kódot is generáltattunk.
- 2. Logikai szintézis** – Ez a lépés készíti elő a platformfüggő fizikai tervezést. Az idáig elkészült (elsősorban viselkedést specifikáló) HDL kódból a szintetizátor szoftver egy RTL (Register Transfer Level) reprezentációt készít. Ez egy szinkron digitális rendszert modellező absztrakciós szint, amely a fejlesztés alatt álló hardver viselkedését regiszterek közötti jelfolyamokkal, illetve a rajtuk végzett logikai műveletekkel jellemzi. Sok esetben a szintézis után módosulhat a rendszer időbeli viselkedése a szinkron működés következtében.
- 3. Fizikai tervezés** – A hardvertervezésnek eme fázisában a logikai terv implementációja történik a cél FPGA-n. Az Actel szoftvercsomagjában a Designer programmal végezhető el ez a feladat (az ábrán a Designer grafikus felületének egy része látható, amelyen követhetők a tervezési folyamat lépései). Elsőként a *compile*-t, azaz a fordítást hajtjuk végre, amely a szintetizált kódot kapja bemenetként és egy EDIF netlist fájlt eredményez. Ezt követi a *layout*

tervezés, a geometriai elhelyezés és összehuzalozás az FPGA-n. A layout tervezés előtt kell beállítanunk, hogy a logikai terv fő szintjén lévő ki- és bemeneti portok az FPGA melyik lábára legyenek kivezetve, ehhez az I/O Attribute Editor használható, amely a MultiView Navigator program része – ebben a programban finomíthatjuk a fizikai elrendezést is. A *Back-Annotate* rész a NYÁK tervező programokhoz hasonlóan a place&route során tett módosításokat vezeti vissza a kapcsolási tervre. A fizikai design utolsó lépése a programozófájl generálása, amellyel az FPGA-t ténylegesen felkonfiguráljuk. Ez konkrétan a JTAG soros interfészen keresztül bináris állomány letöltését jelenti az eszközünkbe.



13. ábra. Az Actel Designer tervezési lépései

5. **Programozás** – vagy más néven konfigurálás egy FlashPro3 eszközzel történik, amely USB-n csatlakozik a PC-hez, és JTAG interfészen keresztül fér hozzá az FPGA-hoz. A letöltés vezérlése az ugyancsak FlashPro nevű programmal történik.

Előzetes verifikáció

A fenti lépéseket kiegészíti a rendszer, illetve az azt alkotó modulok verifikációja, amely a beprogramozás előtt szimulációval valósítható meg. Sok esetben túl bonyolult szimulálni a teljes rendszert, így annak működését magán a hardveren végzett tesztekkel ellenőrizzük. Viszont egyszerűbb, különösen az általunk megírt specifikációk esetén volt értelme a szimulációnak.

Általánosan egy áramkör szimulációjakor a bemenetekre adott gerjesztést mi adjuk meg. Logikai áramköröknél gyakran tesztvektornak nevezik a bemeneti jelsorozatot.

Hasonlóan a logikai áramkörök, vagy általánosan a hardver modelljének leírásához, a szimulációs környezetet is specifikálni kell, rendszerint ugyanazon a hardver leíró nyelven, amelyben a szimulálandó modul specifikálva lett. A mi esetünkben ehhez egy másik, ún. testbench Verilog fájl kell elkészíteni. Az Actel makrók és egyes IP core-ok esetén a HDL kód generálásakor egy testbench is generálódik, amelyet szükség esetén módosíthatunk.

Létezik a gerjesztés bevitelére egy alternatív módszer is, amelyben egy erre alkalmas program (a Libero korábbi verzióiban a WaveFormer Lite volt integrálva erre a célra, sajnos azóta fizetős lett) grafikusán megrajzolhatjuk a hullámformákat, amiket a bemenetre szeretnénk adni, ezután a program automatikusan generál egy Verilog vagy VHDL stimulus fájlt.

A Libero fejlesztőkörnyezetben a szimuláció a terv három különböző fázisában lehetséges:

A logikai szintézis előtti (pre-synthesis) szimulációval azt tudjuk ellenőrizni, hogy funkcionálisan helyesen működik-e a HDL specifikációja a fejlesztés alatt álló modulnak.

A szintézis utáni (post-synthesis) szimuláció már közelebbi eredményt ad a valósághoz, mivel ez az RTL modellből fordított hardver leírást szimulálja. A stimulus lehet ugyanaz, mint a szintézis előtti esetben, ugyanis a testbench modul nem lehet szintetizálni (olyan HDL nyelvi elemeket tartalmaz, amik kizárólag a szimulátor számára értelmezhetőek¹), másrészt a testbench modulnak nincsenek bemenetei, csak a tesztelés alatti eszköz (DUT – device under test) gerjesztéseinek előállítására szolgál. A post-synthesis szimulációban derül ki ténylegesen, ha a tervben szinkronitással/időzítéssel kapcsolatos hibák vannak, vagy valami nem úgy szintetizálódott, ahogy a tervező elképzelte.

A harmadik lehetőség a layout tervezés utáni állapotban történő post-layout szimuláció. Ez adja a legpontosabb becslést a tervezett hardver működéséről, mielőtt beprogramozzuk az FPGA eszközt és ténylegesen kipróbáljuk, mivel a tényleges elrendezés alapján szimulálja a megfelelő jelek késleltetéseit, ezáltal előzetesen felderíthetők az esetleges hazárdok a rendszerben.

A szimulációhoz a Mentor Graphics ModelSim-jének 6.5-ös Actel verzióját használtuk.

Bemérés, tesztelés, fizikai verifikáció

Az FPGA konfigurálása után elvileg már fizikailag tesztelhető, műszerekkel bemérhető a rendszerünk. Ehhez a laboratóriumban rendelkezésre álló műszereket használtuk:

- UNI-T UT71B Multiméter (4 ½ digitális kijelző, True RMS, stb.)
- Agilent 54622D Mixed-signal oszcilloszkóp (100 MHz, 200 Msps)

Az oszcilloszkóp képes 16 csatornás digitális jelek mérésére is, de nekünk sajnos több jelet kellett egyidejűleg mérnünk, ezért logikai analizátort használtunk.

- HP 1654B logikai analizátor (max. 100 MHz, 64 csatorna, 1Kbit/cs. memória)

Az FPGA programozására használt eszközt egyaránt használjuk konfigurálásra és a processzor tesztelése során hardverszondaként.

- Actel FlashPro3 device programmer

A számítógép, amelyen a fejlesztést végeztük:

- Intel Core2 Duo CPU, 2.8GHz, 2GB RAM

A tápegységünk számára a bemeneti feszültséget az előző pontban említett laboratóriumi tápegységgel állítottuk elő:

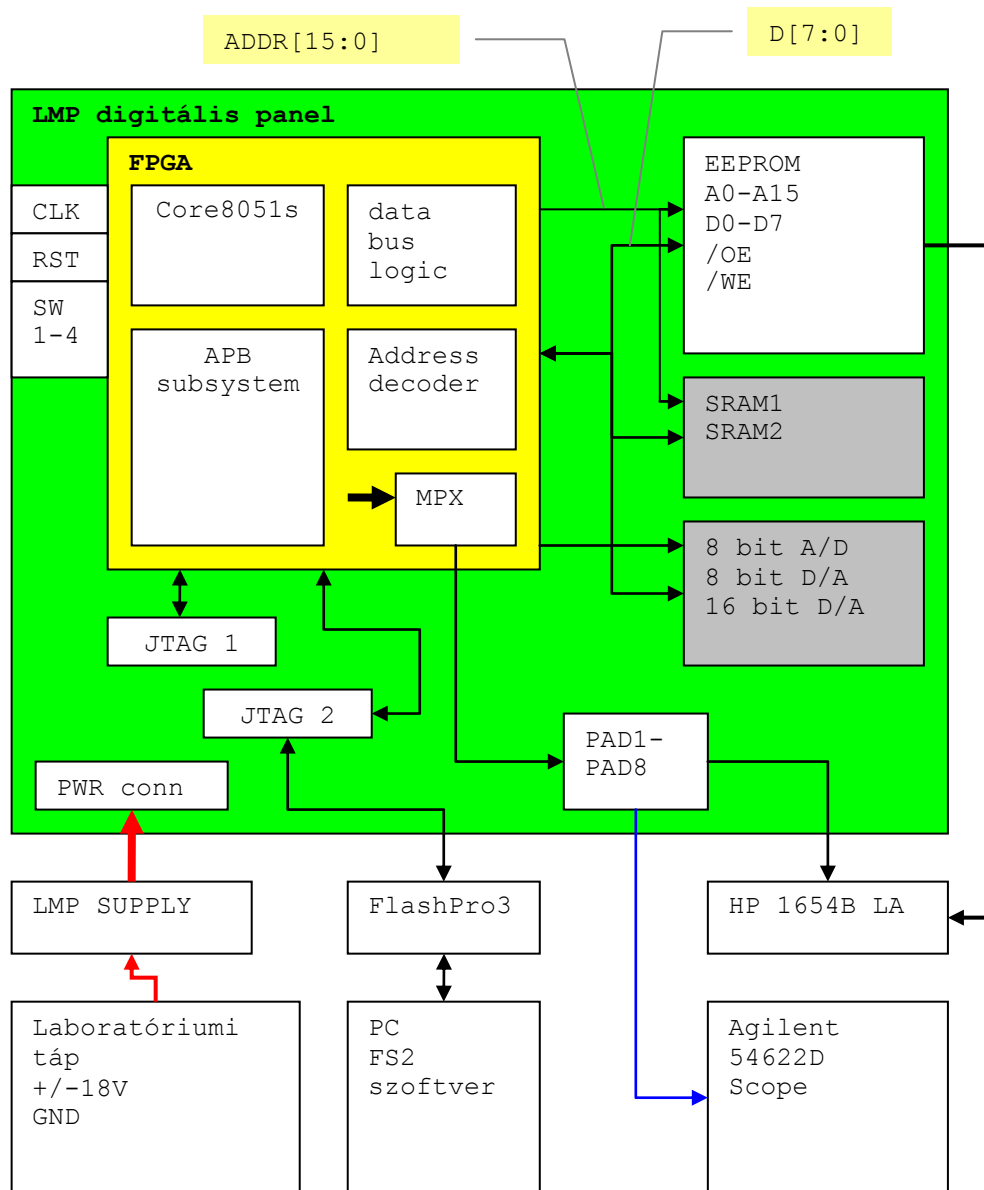
- FOK-GYEM TR-9178 DC POWER SUPPLY

A logikai analizátor egyes mérőfejeit a panel PADx jelű mérőpontokra kötöttük, amelyekre az FPGA-ból általunk speciálisan tesztelési célból vannak jelek kivezetve. Ezekből csak 8 van, tehát ide olyan jeleket vezettünk ki, amelyek nehezen hozzáférhetőek a panelen vagy belső jelek, ilyenek a mikrokontroller core és belső perifériáinak vezérlőjelei, egyes chipkiválasztó jelek, az órajel, a reset jel, stb. Később, amikor 8-nál több jelet kellett bemérni (a címdekóder tesztelésekor), egy FPGA-n belüli multiplexert használtunk. Az FPGA I/O jelei 3.3V-os amplitúdójú bináris jelek.

Az adat- és címbusz egyes bitjeihez, illetve a programmemória írás- és olvasás engedélyező jelekhez az EEPROM IC lábain fértünk hozzá, itt TTL (5V-os) szintű jelek mérhetők.

¹ Ez nem jelenti azt, hogy nem lehetne őket fizikailag megvalósítani. Léteznek olyan hardver leíró nyelvek, amelyekben csak szintetizálható elemek vannak.

A logikai analízátor 4 db 16 csatornás POD-on keresztül méri a jeleket, szerencsére POD-onként állítható a logikai komparálási feszültségszint, ezért nem okoz gondot a különböző szintű logikai jelek mérése. A mérési elrendezés blokk-sémája a következő ábrán látható:



14. ábra: mérési elrendezés blokk-sémája

Szoftverfejlesztés és hibakeresés (debug)

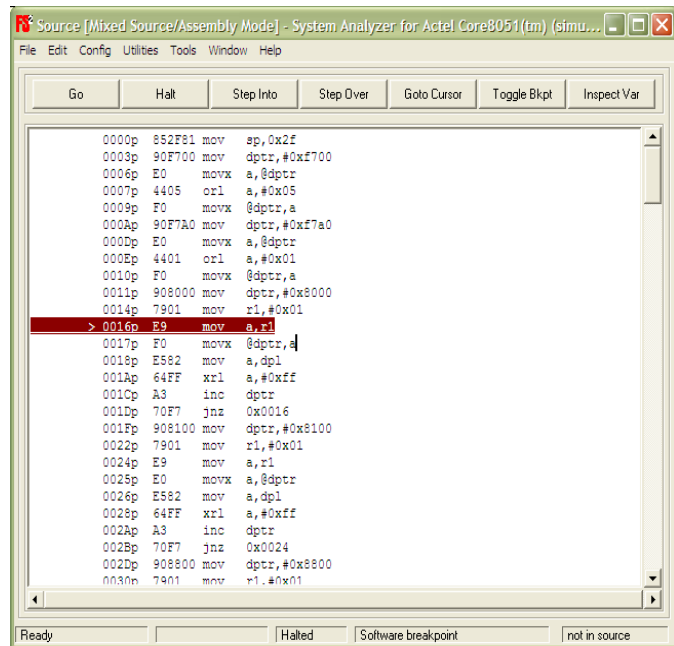
Az előzőekben a hardver fejlesztéséről és verifikációjáról esett csak szó, azonban ezt a tényleges működés során a rajta futó program irányítja. Szükséges volt tehát tesztprogramokat írunk a 8051-es mikrovezérlős platformra. A végleges szoftver C nyelven fog elkészülni, azonban az általunk végzett tesztek során elég volt egy 8051-es assembler használata.

A Core8051s rendelkezik egy ún. OCI (On-Chip Instrumentation) blokkal, amelyet a logikai tervezés során példányosításkor tudunk bekapcsolni. Ez a blokk teszi lehetővé a processzor debuggolását, azaz a bekapcsolt állapotban történő hibakeresést - gyakorlatilag a mikrovezérlő feletti irányítást adja a kezünkbe. A gépi kód letöltését az EEPROM-ba külön programozó berendezés nélkül szerettük volna megvalósítani, ezért szükség volt a Core8051s debug interfészére. Ehhez az Actel 8051-es mikrokontroller core-okhoz a First Silicon solutions cég által kifejlesztett FS2 ISA-ACTEL51 ún. „In-Target System Analyzer”, vagy

röviden debugger programot használtunk, amely a FlashPro programozót használja, hogy JTAG jelekkel hozzáférjen az MCU OCI blokkjához – és a mikrokontroller core-on keresztül a program- és adatmemóriához. A debugger (a továbbiakban a rövidség kedvéért FS2) rendelkezik grafikus és parancssoros felülettel is, illetve lehetőség van a 8051 szimulációjára is, amellyel hardver nélkül is kipróbálhatók a programok. A programmemóriában lévő gépi kódot az FS2 az ábrán látható módon automatikusan visszaalakítja assembly utasításokká egyszerűbbé téve a hibakeresést. Kódunkat lépésenként hajthatjuk végre, breakpoint-okat helyezhetünk el a programban, stb., tehát hasonlóan rugalmasan lehet programozni a beágyazott vezérlőt, mint egy normális CPU-t. Arra is lehetőség van, hogy közvetlenül a grafikus felületről assembly utasításokat átírjunk, azaz a memóriát módosítsuk. A programmemóriába tölthetünk Intel HEX, bináris, vagy OMF51 formátumú kódot [12]b .

Hosszabb kódok fordítására az ASEM-51 macro assembler-t használtuk, amely egy DOS alatt is működő egyszerű parancssoros assembly fordító [13].

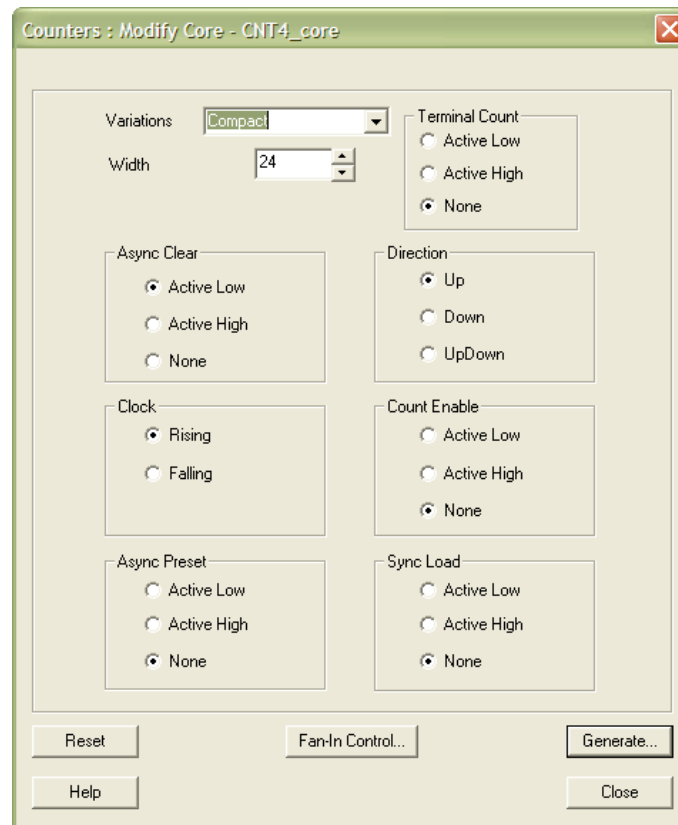
Az FS2 parancssoros felülete (CLI – Command-line Interface) a debugger szolgáltatásait alacsonyabb szinten teszi elérhetővé, emellett támogatja a TCL (Tool Command Language) nyelvű scriptek írását, amivel a mikrovezérlő tesztelése jól automatizálható.



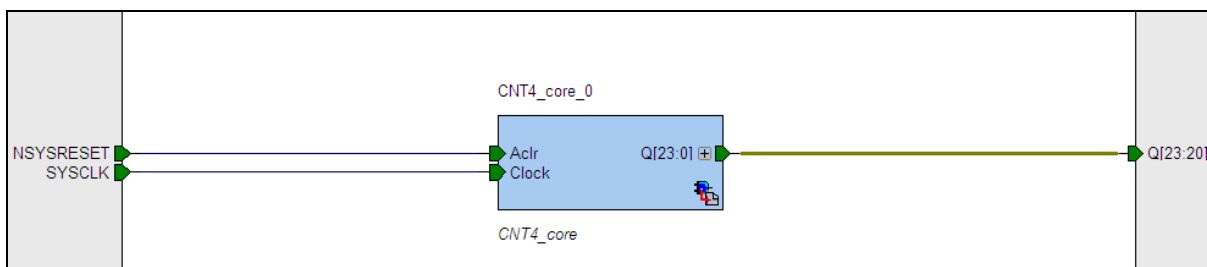
15. ábra: FS2 debugger GUI

3.4 Az FPGA áramkör tesztelése egy egyszerűbb logikai tervvel

Az FPGA és néhány egyéb minimálisan szükséges kiegészítő alkatrész (reset IC, órajelet előállító oszcillátor, szűrőkondenzátorok, ellenállások) beültetését követően egy egyszerű tesztkonfigurációval ellenőriztük, hogy a FlashPro3 programozóval észleljük-e az FPGA-t. A konfigurációban csak egy 24 bites számláló van a következő paraméterekkel:



16. ábra: számláló beállításai

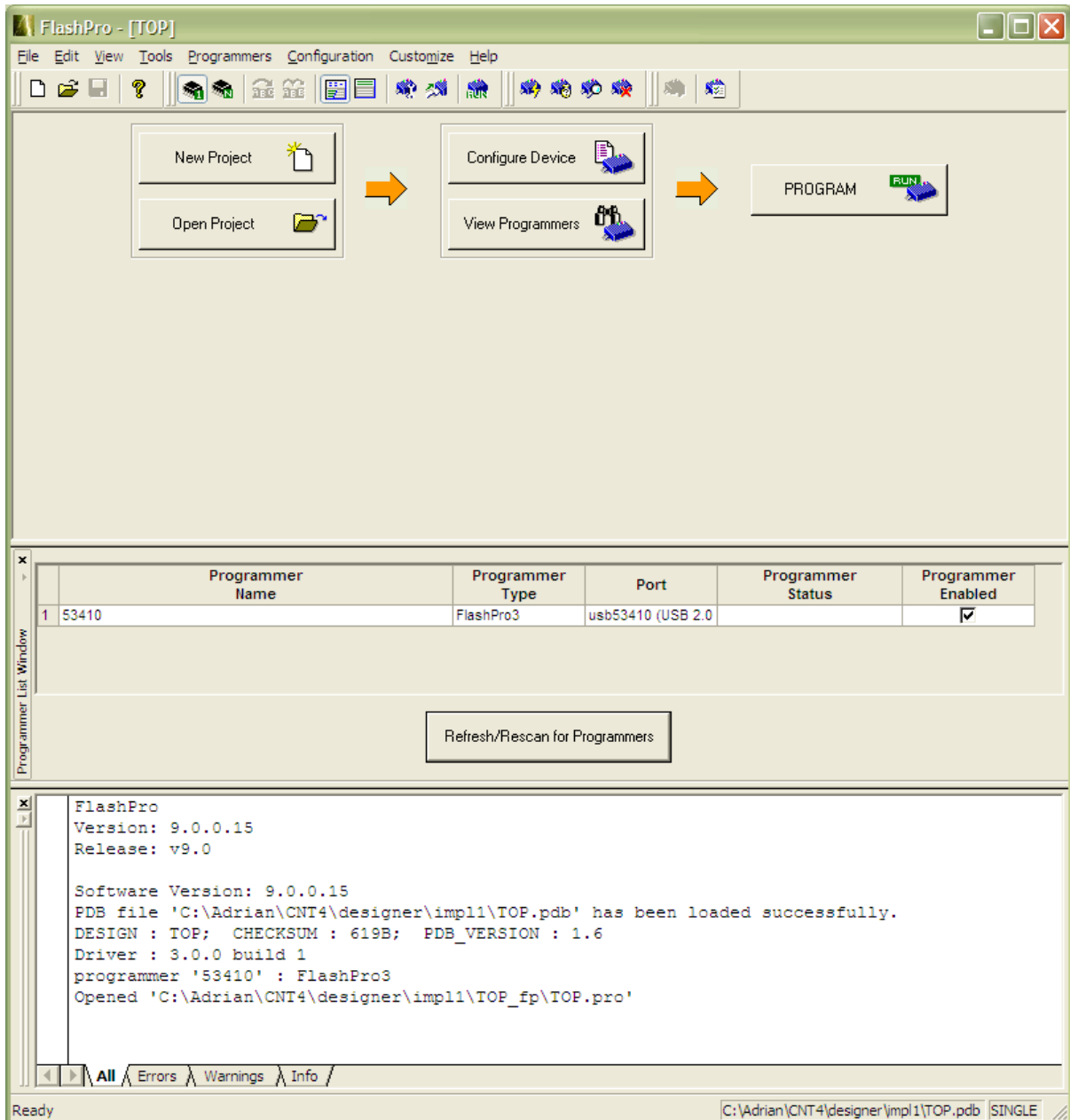


17. ábra: számláló modul példányosítása és kivezetése

A számláló a projekt TOP moduljának egyetlen komponense, ahogy az alsó ábrán látható. A 24 bitből csak a felső 4-et vezettük ki az FPGA megfelelő lábaira, amelyek a panelen 4 LED-et vezérelnek. A rendszer 16 MHz-es órajelet (SYSCLK) egy CSX750F típusú oszcillátor állítja elő. A számláló aszinkron törlése az NSYSRESET jel hatására történik. Ezt a panelen egy LM809 típusú reset IC küldi az FPGA-nak bekapcsolás után, amikor már stabil a tápfeszültség.

A tesztkonfiguráció tervezése a szokásos FPGA tervezési lépésekből áll: a számláló blokk példányosítása (ehhez egy Actel makrót használtunk, a paramétereket az ábrán látható ablakban lehet beállítani), hdl generálás, szintézis, kivezetések beállítása, place&route, programozófájl generálása.

Az FPGA programozása a FlashPro szoftverével történik, az alábbi ábrán a FlashPro GUI-ján látható, hogy a szoftver érzékelt, hogy a programozó csatlakoztatva van a számítógép USB portjához. A Programmer Status oszlopban a programozási folyamat állapota nyomon követhető. Indítás után a programozó törli az FPGA korábbi konfigurációját, majd letölti az általunk fejlesztett rendszert a JTAG interfészen keresztül.



18. ábra: A FlashPro programozó kezelőfelülete

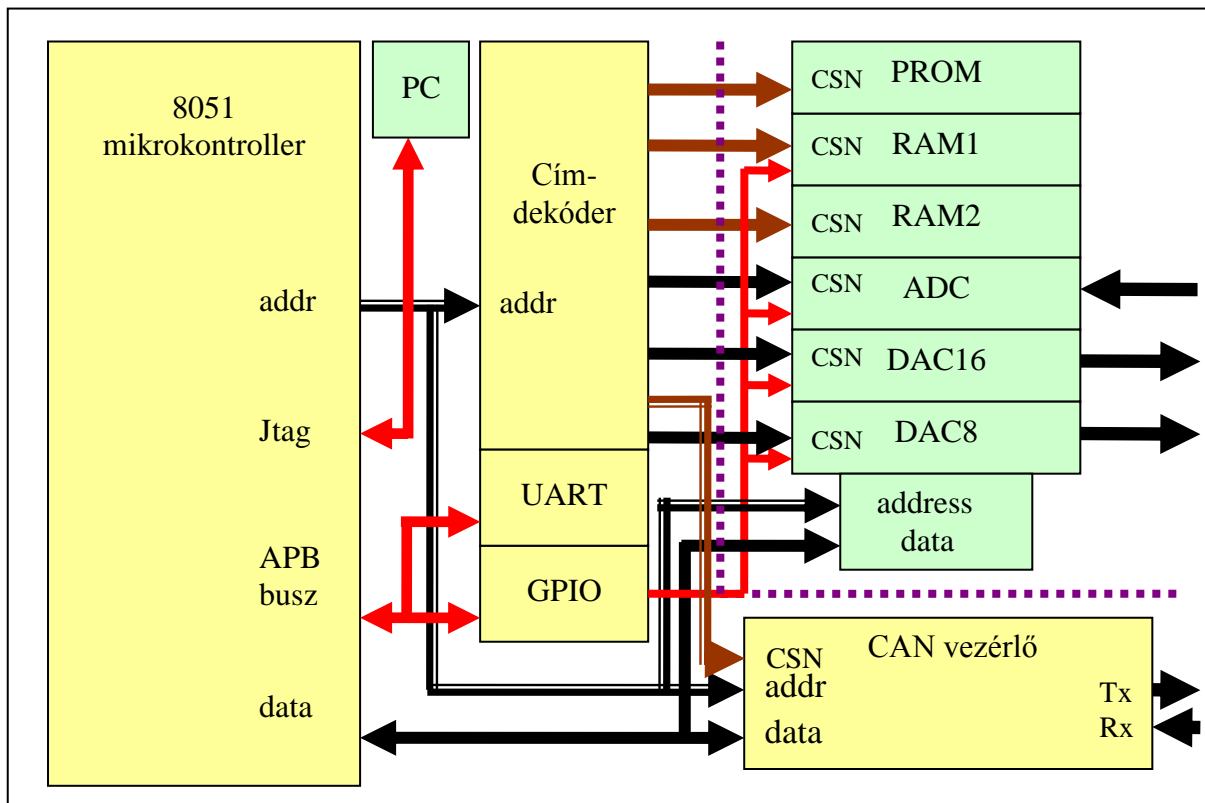
A 24 bites számláló $2^{24}/16 \times 10^6 = 1.048576$, azaz kb. 1 másodperc periódusidejű négyszögjelet állít elő a legnagyobb helyiértékű biten, tehát az MSB-re kötött LED villogásából ránézésre is egyszerűen megállapítható, hogy helyesen működik-e a konfiguráció. A számláló működését egy Agilent 54622D típusú oszcilloszkóppal is

ellenőriztük, ami sikeresnek bizonyult, tehát sor kerülhetett az FPGA következő, sokkal bonyolultabb konfigurációjára.

3.5 FPGA áramkörbe ágyazott logikai terv részletes

ismertetése

3.5.1 A modul elemei



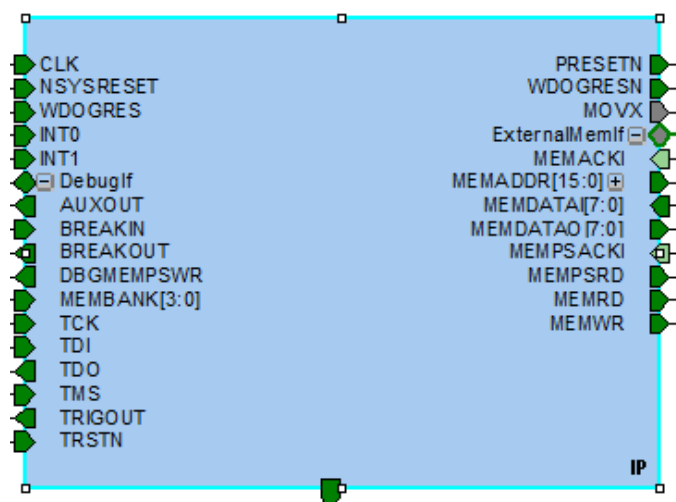
19. ábra: az FPGA modulban lévő elemek (sárga) és a hozzájuk kapcsolódó külső perifériák (zöld)

Az FPGA modul belső felépítését a 19. ábra mutatja. A tervet úgy alakítottuk ki, hogy a lehető legrugalmasabb működést tudjuk elérni a lehető legkevesebb erőforrás felhasználásával. A terv összeállításához az Actel cég Libero nevű ingyenes fejlesztői környezetét használtuk. A fejlesztői rendszeren keresztül díjmentesen elérhetőek - könyvtári elemként - az APB busszal rendelkező 8051 processzornak és az APB buszhoz csatlakoztatható perifériáknak az úgynevezett IP CORE-jai. Az IP a **szellemi tulajdon** (*intellectual property*) kifejezés rövidítése, míg a CORE magyarul magot jelent. Az IP magok valójában mások által már megtervezett FPGA modulok tervei, melyeket egy hardverfejlesztő építőelemként tud felhasználni a saját munkájának elkészítéséhez. Sajnos CAN busz vezérlő tervei a fejlesztői környezetben nem találhatóak, ezért ezeket az OpenCores.org weboldalról töltöttük le verilog formában. Az IP CORE-okat a tervbe való behelyezés előtt mindig példányosítani kell. Ilyenkor a tervezőprogram megkérdezi, hogy milyen opciókkal szeretnénk használni az eszközünket.

3.5.2 A mikrokontroller core

A logikai tervben először az FPGA modul központi elemét képező mikrokontrollert kellett elhelyezni. Az Actel 8051s nevű IP Core-ja egy olyan mikrokontroller, amelynek utasításkészlete egyezik ugyan az Intel 80C51 típusú mikrokontrollerével, a valós működésében és a perifériáinak kialakításában azonban jelentősen eltér.

Az eredeti Intel 80C31 processzor egy gépi ciklust 12 órajel alatt hajt végre, és egy utasítás végrehajtásához átlagosan 2 gépi ciklusra van szüksége. Az Actel processzora viszont a legtöbb utasítást 1 gépi ciklus alatt hajtja végre, és egy-egy gépi ciklushoz is csak egyetlen órajel-periódusra van szüksége. Mindezen tulajdonságokból következik, hogy a Core8051s átlagosan 8-szor nagyobb teljesítményre képes, mint hagyományos elődje. Másik fontos különbség köztük, hogy a Core8051-nek teljesen külön vannak az adat és címvezetékei, míg az eredeti processzor 8 bites adatjele a címjelek egyik 8 bitjével időben multiplexálva jelenik meg ugyanazon a 8 bites buszon. A harmadik fontos eltérés a két eszköz között az, hogy az Actel változatából kivettek számos, - az eredeti processzorban még benne lévő - perifériát, és helyettük csak egy APB busz áll rendelkezésre. Ez persze nem jelent problémát, mivel a processzor APB buszához tetszőlegesen hozzacsatlakoztathatjuk az általunk kívánt perifériákat, melyek szintén letölthetők az Actel honlapjáról. A processzor az APB buszt és eszközöket az APB vezérlő regiszterein keresztül látja, melyek benne vannak a processzor memória-címtartományában. A Core8051s mikrokontroller - az általunk használt ACTEL PROASIC-3 FPGA-ban - a dokumentáció szerint 24-27 Mhz körüli órajel-frekvencián képes működni, de a végső maximális sebességet csak akkor fogjuk tudni majd meghatározni, hogyha a tervezés végére érünk, és a fejlesztő környezettel kiszámoltatjuk a rendszerben lévő legnagyobb késleltetés nagyságát.



20. ábra: A Core8051s processzor lábkiosztása

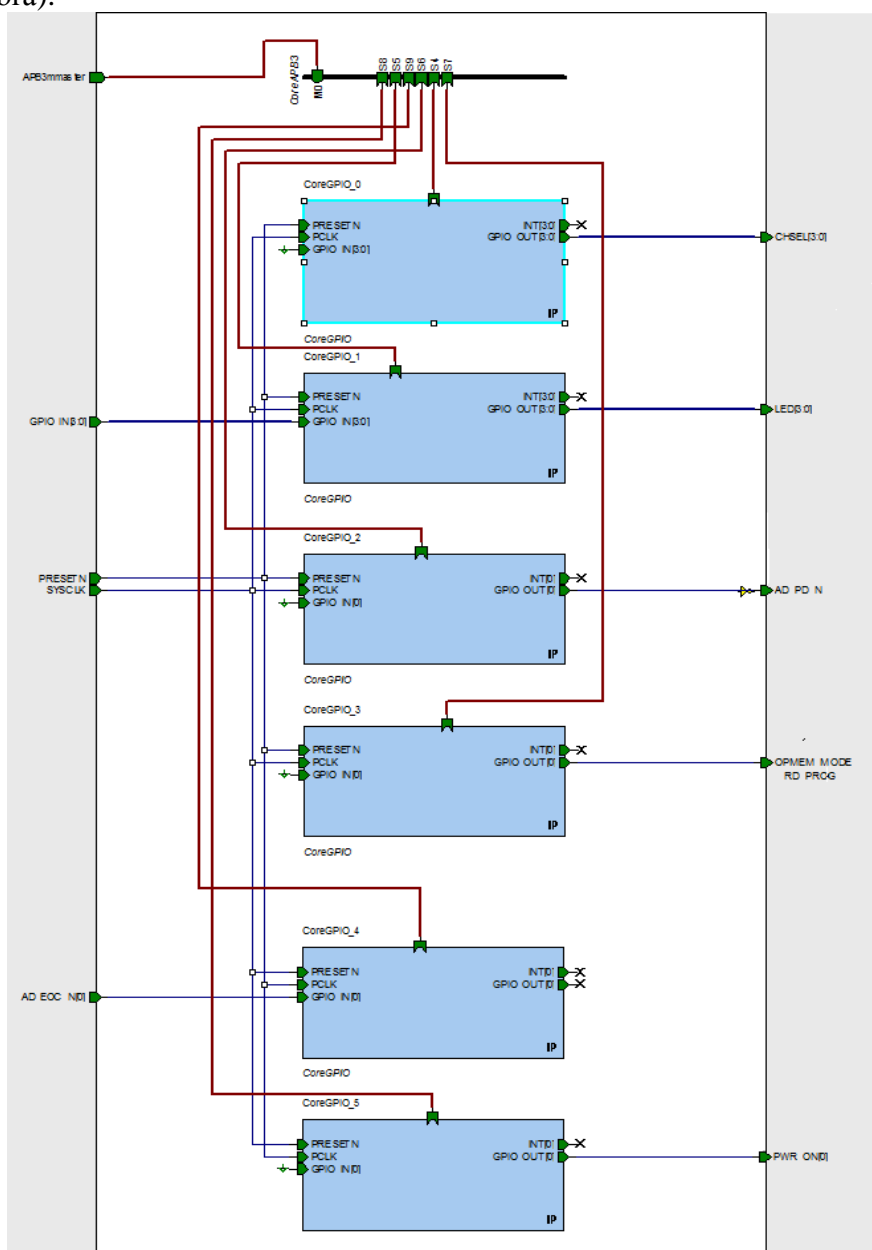
A 8051 példányosításánál meg kellett adnunk, hogy milyen paraméterekkel szeretnénk használni a processzort, ilyen opcionális paraméterek voltak például: JTAG interfész bekapcsolása, APB busz adatszéléesség, memória-visszajelzés ciklusok, belső memória bekapcsolása stb.

A processzornak külön van egy 16 bites címbusza és egy 8 bites adatbusza. Kétfajta memóriát képes kezelni: adatmemóriát és kódmemóriát. Mindkét memóriafajtának az adatbusza és a címbusza is közös: MEMDATAI, MEMDATAO, MEMADDR, viszont a mikrokontroller mindig különböző RD és WR lábait aktiválja attól függően, hogy a

kódmemória címtartományában, vagy az adatmemória címtartományában lévő eszközhöz szeretnénk hozzáférni. A processzor a kódmemóriából történő olvasáskor a MEMPSRD, az adatmemóriából történő olvasáskor az MEMRD lábakat mozgatja, míg az adatmemória írásakor a MEMWR lábát. Ha debug módban használjuk az eszközt, lehetőség van arra is, hogy a JTAG interfészen keresztül feltöltsük a kódunkat a kódmemóriába. Ilyenkor a processzor a DBGMEMPSWR lábbal vezérli a kódmemória írását. E műveletre azért volt lehetőségünk, mert az áramkör földi teszt példányában még EEPROM-ot használtunk - a könnyebb tesztelhetőség végett. PROM esetében értelmetlen lett volna a kódmemória írásával próbálkozni.

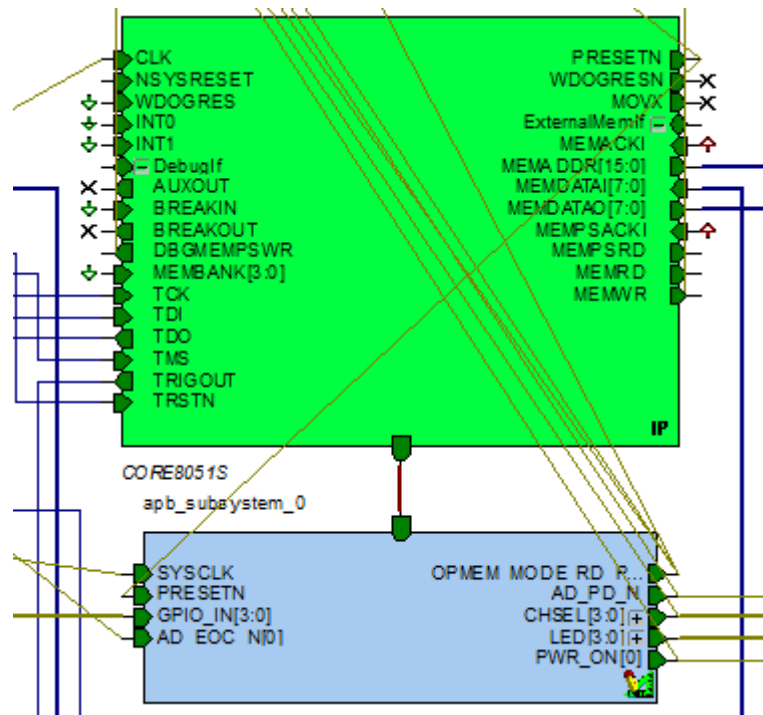
3.5.3 Az APB busz, az APB perifériák és az APB alrendszer

Ahogy az előző fejezetben már említettük, a processzor rendelkezik egy APB (Advanced Peripheral Bus) busszal, amin keresztül különböző perifériák csatlakoztathatók hozzá (21. ábra):



21. ábra: Az APB alrendszer

Az APB busz felépítése alapvetően olyan, hogy van egy buszvezérlő, melyre 1 master és több slave csatlakozik, a masternek van joga mindegyik slave-vel kommunikációt folytatni. Amikor példányosítjuk a processzort (20. ábra) és a buszvezérlőt, meg kell adnunk, hogy milyen adatszélességekben akarunk kommunikálni a buszon. Busz-szélességből beállíthatunk 8/16/32 bites értékeket is. Mi 8 bites busz-szélességet állítottunk be, hogy a lehető legkevesebb erőforrást használjuk fel az FPGA-ból. Hogy az APB eszközöket összefogva tudjuk kezelni, létrehoztunk egy modult, melyet Apb_subsystem-nek neveztünk el. Ehhez a modulhoz kapcsolódott a processzor masterként (22. ábra), a buszvezérlő és a perifériák pedig a modulon belül kerültek elhelyezésre².



22. ábra: A processzor és az APB alrendszer összekapcsolása

A következőkben az APB alrendszer felépítését ismertjük. Mielőtt az áramkör felélesztését végeztük volna, korábban készítettünk egy olyan hardvertervet, amelyben minden - számunkra szükséges - APB periféria benne volt, ezt azonban nem mertük teljes egészében elkezdni az éles áramkörtől tesztelni, inkább újra elkezdtünk felépíteni egy tervet, és fokozatosan próbáltuk ki az éppen beépített modulokat. Ezt a módszert nem csak az APB alrendszerénél használtuk, hanem a többinél is.

Az APB alrendszer fő komponense a buszvezérlő, amely a [10][16] felső részén helyezkedik el. Ehhez kapcsolódik a processzor masterként kívülről, illetve ehhez csatlakoznak a slave áramkörök. A mastert és a slaveket egyetlen kattintással hozzá lehetett huzalozni a buszvezérlőhöz, mert ezt a fejlesztőrendszer automatikusan elvégzi helyettünk - még az összekötő vezetékek nevét sem kellett részletesen megismernünk. Az általunk eddig összerakott rendszerben még a slave perifériákat csak 4 db. 8 bemenetű és 8 kimenetűre állított GPIO³ egység képviselte. Ezeket vezérlőjelek kapcsolgatására szeretnénk volna később használni, ezért az összes ilyen jelet bekötöttük rájuk. A fontosabb jeleket külön

² Az ábrán keresztül-futó vonalak valójában buszok, melyeket a fejlesztőkörnyezet sajnos ilyen módon jelöl. Bár a fejlesztői környezet grafikus megjelenítése sokszor nehézkes, ennek ellenére a funkcionális működésben nem tapasztaltunk semmi rendellenességet.

³ General Purpose Input And Output: Általános célú (digitális) ki- és bemenetek (kimenet megőrzi az értékét átírás után)

GPIO-ra kötöttük, hogy a majdani szoftver-fejlesztőknek könnyebb dolga legyen. A bekötött vezérlőjeleket most röviden felsoroljuk a címükkel együtt:

- chsel[3:0]: Ez a 4 bit az analóg multiplexer különböző bemenetei közötti választásért felelős.
- leds[3:0]: A NYÁK-on található 4 LED vezérelhető velük.
- AD_PD_N: Ha 0-át adunk erre a kimenetre, az AD7822 A/D konvertert standby-ba, hogyha pedig "1"-et, akkor újra visszakapcsoljuk.
- OPMEM_MODE_RDPROG: Ez a memória konfigurációk közötti átkapcsolásra való, melyről részletesen fogunk írni a címdekódról szóló fejezetben.
- PWR_ON: Ezzel a bittel lehet lekapcsolni az EEPROM (végleges verzióban a PROM) tápfeszültségét.

Az APB buszra kötött slave-eket a processzorból úgy lehet elérni, hogy minden slave-hez tartozik egy APB kezdőcím, amelytől kezdve a periféria vezérlőregiszterei egymás után, egy relatív címen helyezkednek el az APB címtartományban. Az APB címtartomány a processzor külső memória-címtartományának F000-tól kezdődő felső 4KBytejába vannak beágyazva. Ha például a 4. APB slave regisztereit akarjuk elérni, akkor F400 lesz a periféria báziscíme és ehhez kell hozzáadni az adott regiszter relatív címét. Minden APB perifériának más regiszterei vannak és másként működnek. Ha csak 8 bites a címbusz, akkor egy regisztert úgy érünk el, hogy egyszerűen az adott periféria adott regiszterének címére ki kell írni egy bájtot. Hogyha viszont a rendszert pl. 32 bites módban akarjuk használni, akkor 3 bájtot először be kell töltenünk 3 megfelelő adatregiszterbe, majd a negyediken kiírni az adott címmel az adatbuszra, és ekkor megvalósul a buszon a 32 bites kommunikáció. Mi most csak 8 bites APB szélességű kommunikációt valósítottunk meg, ezért nem kellett a kiegészítő regisztereket használnunk. Eddigi munkánk során a GPIO és az UART APB-perifériát használtunk. Most csak a GPIO modulok működését fogjuk elmagyarázni, mert az UART kezelése hasonlóan történik.

A CoreGPIO eszközök működését és regisztereinek címkiosztását a modul dokumentációjában találtuk meg [17]. A mi rendszerünkben 8 bites busz-szélességű APB buszt és modulokat használtunk. A modul vezérlőregisztereinek címét úgy kapjuk meg, hogy a modul báziscíméhez hozzáadjuk az adott regiszter belső címét. Ilyenkor a mikrokontroller programjának csak kiírni vagy beolvasni kell az adott bájtcímre/bájtcímről egy adatot.

A GPIO perifériáknak 3 legfontosabb regisztertípusa van, - az APB címtartományba ágyazva (A GPIO báziscíme mindenhol xy00h):

- I/O lábak konfiguráló regiszterei: minden bithez 1 bájtot (xy00,xy04,...)
- Input adatregiszter (xy90h)
- Output adatregiszter (xyA0h)

Amikor pl. a GPIO egyik kimenetét egybe kell állítanunk, akkor az adott bithez tartozó vezérlő regiszter címére egy - specifikációban megadott, - engedélyező értéket kell írunk, majd utána kiírhatjuk az új adatot az adatkimeneti regiszterbe. A beolvasás ugyanígy történik. Mi ezt a folyamatot az FS2 Console és Debugger software segítségével teszteltük végig.

3.5.4 A címdekóder elkészítése a memóriák és a külső címtartományba ágyazott perifériák kezeléséhez

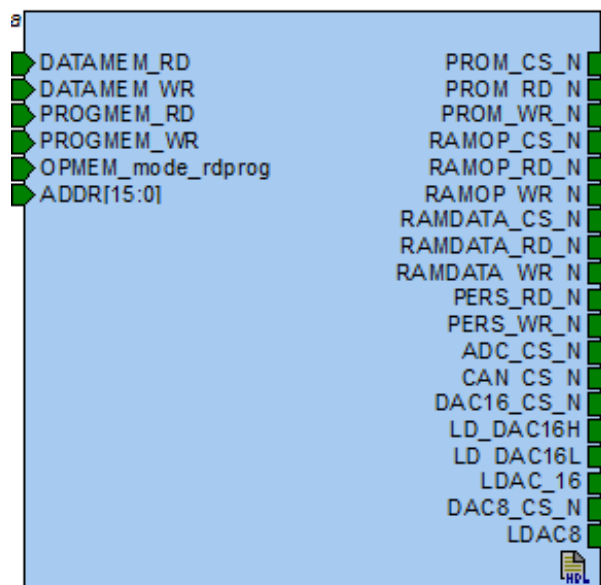
Az eddigiekben olyan perifériákról volt szó, melyek az APB buszon kapcsolódtak a processzorhoz. A valóságban tulajdonképpen ezek is a külső memória-tartományba vannak ágyazva az F000 cím fölött. A gyakorlatban azért volt könnyebb APB-s perifériákat használni, mert nem kellett külön címdekódert készíteni a beágyazásukhoz és megvalósítható lett volna akár 16 vagy 32 bites perifériák kezelése is. Az sem elhanyagolható szempont, hogyha szűkében lettünk volna a memória-területnek, akkor még inkább érdemes lett volna használni az F000 fölötti címeket, hogy ne használjuk fel feleslegesen a többi külső címet. Ebben az alfejezetben azokról a perifériákról is szó lesz, amelyeket a terv szerint - hagyományos módon - a processzor külső memória-tartományába ágyasztunk.

Ahhoz hogy a külső memóriatartományban elhelyezkedő memóriák és perifériák működni tudjanak, elengedhetetlen volt egy címdekóder logikai modul elkészítése. Ennek leírása fogja kitölteni e fejezet legnagyobb részét.

A címdekóder a 19. ábra közepén, felül helyezkedik el. Részletes rajzát a 23. ábra mutatja. A címdekóder alapvetően egy kombinációs hálózat. Az egység elsődleges feladata a memóriák és a memóriatartományba ágyazott perifériák negált CS (chip select) jelének lehúzása akkor, amikor az adott eszközt címzi a processzor. A dekóderbe általában a címvezetékek felső bitjeit kötjük be, melyek meghatározzák, hogy melyik eszköz kerüljön kiválasztásra, míg az alsó bitjeik azt adják meg, hogy eszközön belül melyik rekeszhez szeretnénk hozzáférni. Alapesetben ilyen követelményeink lennének csupán egy ilyen áramkörrel szemben, a mi esetünkben azonban egy speciális memória-vezérlést is meg kellett valósítani. Erről a problémáról és a megoldásáról ebben a fejezetben részletesen fogok írni, most azonban először a címdekóder alapvető működésének ismertetem.

Olyan logikát kellett készíteni, amely minden memória és periféria CSN (CS negált) jelét kezeli. Az egység külső sematikus rajzát a 23. ábra mutatja. A dekóder kódját Verilog hardverleíró nyelven készítettük el, ezt az 1. számú melléklet tartalmazza. A fejlesztői környezet lehetőséget biztosított ahhoz, hogy keverjük a különböző hardverleíró nyelveken (VHDL, Verilog) készített modulokat.

A rendszerben - mint már említettük - 3 memóriaegység, 2 D/A konverter, 1 A/D konverter és egy CAN busz vezérlő foglalt helyet. Az eszközök címkiosztását az alábbi táblázat tartalmazza:



23. ábra : Az FPGA modul címdekódere

konfiguráció 1. (OPMEM_mode_rdprog==0)			Dekóder kimenete:	Mely bemenetek kapcsolódnak hozzá?	
memória:	üzemmód:	címek (hex)	~CS	~RD	~WR
PROM	program	0000-7FFF	PROM_CS_N	~PROGMEM_RD	~PROGMEM_WR
RAM1(op.)	adat	8000-FFFF	RAMOP_CS_N	~DATAMEM_RD	~DATAMEM_WR
RAM2(data)	adat	0000-7FFF	RAMDATA_CS_N	~DATAMEM_RD	~DATAMEM_WR
perifériák	kikapcsolva	8000-FFFF	kikapcsolva	kikapcsolva	kikapcsolva

konfiguráció 2. (OPMEM_mode_rdprog==1)			Dekóder kimenete:	Mely bemenetek kapcsolódnak hozzá?	
memória	üzemmód:	címek (hex)	~CS	~RD	~WR
PROM	program	0000-7FFF	PROM_CS_N	~PROGMEM_RD	~PROGMEM_WR
RAM1(op.)	program	8000-FFFF	RAMOP_CS_N	~PROGMEM_RD	~PROGMEM_WR
RAM2(data)	adat	0000-7FFF	RAMDATA_CS_N	~DATAMEM_RD	~DATAMEM_WR
perifériák	adat	8000-FFFF	-	~DATAMEM_RD	~DATAMEM_WR

Perifériák:			Dekóder kimenet:
eszköz:	üzemmód	címek (hex)	~CS
DAC 16 bites	adat	8400-84FF	DAC16CS_N
	adat	8200-82FF	LD_DAC16H
	adat	8300-83FF	LD_DAC16L
	adat	8500-85FF	LDAC_16
DAC 8 bites	adat	8600-86FF	DAC8_CS_N
	adat	8700-87FF	LDAC_8
ADC 8 bites	adat	8000-80FF	ADC_CS_N
		8100-81FF	AD_CONVST_N
CAN vezérlő	adat	8800-88FF	CAN_CS_N

24. ábra: Az eszközök által használt címtartományok és a RD/WR jelek

A 3 memória közül a harmadiknak (RAM2) mindig adat üzemmódban kell lennie, vagyis akkor kell "válaszolni" a címbuszon megjelenő címre, hogyha a processzor földre húzza a MEMRD vagy MEMWR lábát. A táblázatban a RAM2 memóriához tartozó sorban azt találjuk, hogy a memória RD és WR jelére a dekódernek a ~DATAMEM_RD és ~DATAMEM_WR bemenete van kötve. Ezek ekvivalensek a MEMRD és MEMWR jelekkel, mivel a dekóder 2 bemenetére ezeket kötöttük be. A RAM2 memória a 0000-7FFF területet foglalja el az adat-címtartományból.

Az első memória (EEPROM/PROM) a 0000-7FFF címtartományban helyezkedik el és mindig programkód-lehívó üzemmódban van, vagyis csak utasításlehívás és a programkód feltöltése műveletek végezhető rajta. A processzor utasításlehíváskor a MEMPSRD lábát húzza le "0"-ba - ez a láb össze van kötve a dekóder PROGMEM_RD bemenetével - míg a programkód feltöltésekor a DBGMEMPSWR láb "0"-ba húzása vezérli az átvitelt - ez pedig a dekóder PROGMEM_WR lábára van kötve.

A 2. memória és a perifériák hozzáféréseinek vezérlése azonban már nem ilyen egyszerű. A helyzetet az bonyolította - az utóbbi esetekhez képest -, hogy a rendszert úgy kellett megtervezni, hogy indulás után az EEPROM/PROM-ban futó kód másolja át magát a RAM1 memóriába, majd adja át a vezérlést és ezt követően kapcsolódjon ki a kisebb fogyasztás érdekében. Ennek a műveletnek a megvalósításához a RAM1 memóriát kezdetben az 1. konfigurációnak megfelelően adat-memóriaként kezeljük, majd a vezérlés átadása előtt átkapcsoljuk program-memória üzemmódba, ahol a 2. konfiguráció szerint működik. A memória adott üzemmódja azt jelenti, hogy a memória RD és WR jeleit - az adott időben - a

processzor program-memóriához tartozó, avagy az adatmemóriához tartozó RD és WR jelekkel hajtjuk-e meg.

A táblázatból kiolvasható, hogy a második (RAM1) memória kezdetben a 8000-FFFF adat-címet foglalja el, átkapcsolás után azonban átkerül a programkód-címtartományba, ugyanezzel a címmel. Mivel a perifériák szintén a 8000-FFFF adat-címtartományban helyezkednek el, ezeknek a RD és WR jeleit a működés első fázisában (1. konfiguráció) le kell tiltani. Miután megtörtént a RAM1 átkapcsolása program-memóriába, az adatmemória 7FFF-FFFF tartománya felszabadul a perifériák számára, ekkor engedélyezhető, hogy megkapják a ~DATAMEM_RD, ~DATAMEM_WR jeleket.

Azt, hogy éppen melyik konfiguráció érvényes a rendszerben, a címdekóder OPMEM_MODE_rdrprog bemenetével lehet irányítani. Ezt a bemenetet az APB buszra csatlakoztatott GPIO modulok egyikéhez kapcsoltuk, hogy szoftverből beállítható legyen és hosszú időn keresztül meg is tartsa az értékét. A jel beállításának a módjáról később részletesen írunk.

Ha az OPMEM_MODE_rdrprog jel "0" értékű, akkor az 1. konfiguráció érvényes, míg hogyha a "1" értékű, akkor a második. A címdekóder mindig a táblázatban szereplő címeket és RD/WR jeleket rendeli az adott eszközhöz az éppen aktuális konfigurációnak megfelelően.

A táblázat első két részében a perifériák gyűjtőnéven szerepelnek, de a harmadik részben részletesen meg vannak adva az egyes perifériákhoz szükséges vezérlőjelek memóriacímei. Ezeket később fogjuk részletezni. A 16 bites D/A konverter regiszterei és vezérlőjelei a 8400-85FF memóriacímen helyezkednek el, míg a 8 bites D/A konverter regiszterei és vezérlőjelei a 8600-87FF memóriacímen. Az A/D konverter memóriatartománya a 8000-80FF címmel jelzett terület. A táblázatban ez alatt helyezkednek el a CAN busz regiszterei a 8800-88FF címen. A táblázatban látható, hogy egy-egy perifériához több címtartomány is tartozik.

3.6 Memóriaillesztési kihívások

Az előzőekben leírt módszerek és megoldások működnek, de a hozzájuk vezető utunk egyáltalán nem volt olyan egyszerű, mint először gondoltuk. Ebben a fejezetet a kihívásokról és nehézségekről lesz szó, amelyekkel elsősorban a memóriaillesztés során szembesültünk.

Miután az FPGA-t kipróbáltuk, még nem tudtuk, hogyan oldjuk meg az EEPROM in-circuit programozását. Az akkori elképzeléseink szerint a Core8051s (JTAG alapú) debug felületén keresztül a CPU alkalmas arra, hogy a JTAG-en sorosan érkező adatokat beírja a párhuzamos EEPROM-ba – ezt akkor a debugger szoftver (FS2 ISA-Actel51) nélkül még nem tudtuk volna megoldani.

Először egy külső (Needham's Electronics EMP-20⁴) programozóval kísérletük meg írni az EEPROM-ot. Ehhez az eszközhöz csak DOS-alapú szoftver állt rendelkezésre és sajnos a Windows virtuális gépe nem volt képes közvetlenül kezelni a párhuzamos portot, amelyen keresztül az EMP-20 csatlakozik a PC-hez. Alternatívaként próbálkoztunk a DOSBox emulátorral (annak kiegészítésével, amely elvileg hozzá tudott volna férni a nyomtató porthoz), de végül csak a hagyományos DOS alól működött rendesen a szoftver.

⁴ A Needham's Electronics cég univerzális memória programozója, amely a PC nyomtató interfészét használja adatátvitelre és saját 220V-os tápegységgel rendelkezik. A 48 lábú eszközfoglatat egy 100% rugalmasságot biztosító, Xilinx FPGA áramkörrel kialakított elektronikával van meghajtva. Ez azt jelenti, hogy TTL és nem TTL szintű jelekre tetszőleges szélességű és elhelyezkedésű cím, adat és vezérlő busz programozható a PC felől. DIL tokos eszközök programozásához semmilyen lábátjátzó adaptert nem igényel. [15]

Az EMP-20 nagyon sokféle eszköz programozására alkalmas, a programozandó eszközhöz mindig a megfelelő cserélhető panelt (family modul) kell a berendezéshez csatlakoztatni. A mi memóriánk egy AT28C64 párhuzamos EEPROM, szerencsére a programozható eszközök listájában szerepelt ezzel kompatibilis opció. Hosszas kísérletezés után sikerült (byte-onként 10 ms-os ciklusidővel) letölteni a memóriába az előzetes tesztelő kódot; az akkori elképzeléseink szerint az EEPROM-ot visszahelyezve az LMP panelbe bekapcsolás után a szoftver automatikusan futott volna rajta – végül nem így történt és a külső programozó használatának időigényessége, valamint egyéb kockázatok (pl. EEPROM IC lábainak törése a sűrű ki- és beszerelések miatt) elkerülése végett visszatértünk az eredeti ötlethez, a „hogyan?” kérdés viszont továbbra is fennállt.

A megoldást az előzőekben már tárgyalt FS2 debugger program kínálta, azonban ennek használata során további problémák merültek fel. Az első futtatáskor nagy örömeinkre vissza tudtuk olvasni a kódot, amelyet az EMP-20 eszközzel beírtunk.

A kód egy C-ből fordított program volt, amelynek az egyik LED-et kellett volna ki-be kapcsolgatnia – viszont tesztelési célra nagyon hosszúnak bizonyult, ezért rövidebb tesztutakat próbáltunk írni assembly-ben.

Az EEPROM-ba írás viszont csak byte-onként sikerült. Ennek, mint később kiderült, az volt az oka, hogy az EEPROM-nak túl gyors volt a debugger, illetve a CPU sebessége. A tesztutak byte-onkénti bevitele viszont így is rossz ötletnek tűnt, tehát azzal próbálkoztunk, hogy a mikrovezérlő írási parancsait késleltetjük egy alkalmas hardver logikával, amelyet az FPGA-ba illesztünk.

A Core8051s memória hozzáférési ciklusai a dokumentáció [12]a alapján a MEMACKI, illetve MEMPSACKI bemenetekkel szabályozhatóak, illetve fix számú wait ciklussal növelhetőek. Az AT28C64 EEPROM dokumentációja szerint maximum 10 ms egy írási ciklus hossza (ez viszont a 16 MHz-es órajel frekvencián nagyságrendekkel nagyobb, mint a CPU 62.5 ns-os ciklusideje, még akkor is, ha a lehető legtöbb fix számú wait ciklust iktatunk be az írások közé). Az EEPROM rendelkezik egy 64 byte-os lapregiszterrel, amellyel, ún. page write módban a memória egyszerre képes 64 byte rögzítésére – feltéve, hogy az előírt időzítési követelmények be vannak tartva.

Az extrém lassítás helyett egy intelligens késleltetést próbáltunk megvalósítani a programmemória írásához, amely ezt a page write funkciót is kihasználja.

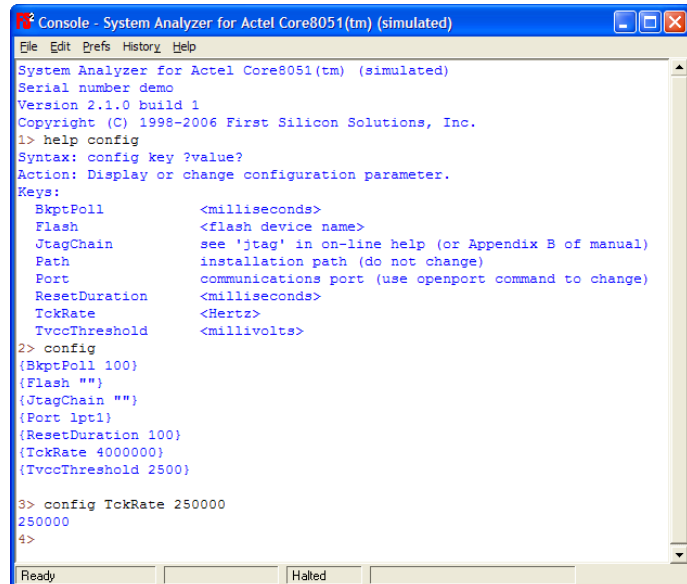
A működés ellenőrzéséhez arra is szükség volt, hogy írás engedélyezés jelet, illetve a cím/adat busz jeleit vizsgáljuk, ezért kezdtük el használni a HP 1654B logikai analizátort ezen jelek időbeli mérésére. A mérési elrendezést a 14. ábra mutatja.

A késleltető modul Verilogban fejlesztettük – a modul feladata volt a CPU core debug blokkjának programmemória írást vezérlő DBGMEMPSWR jelét késleltetni (a MEMPSACKI jel vezérlésével), illetve az EEPROM specifikációjában szereplő időzítési értékeket betartatni. Ehhez a modul többféle sorrendi elemet, számlálókat használt (az EEPROM page write-jához, az egyes beírt byte-ok között eltelt idő méréséhez, az impulzusok hosszabbításához, stb.) A működést előzetes szimulációkkal vizsgáltuk a Libero fejlesztőkörnyezetben. A tervezés és a szimuláció is meglehetősen bonyolult volt: a modul három héten keresztül fejlesztettük annak reményében, hogy a hardveres késleltetés megoldása után nem kell többet foglalkoznunk a sebességi kérdéssel. A beméréskor viszont újabb falba ütköztünk: A MEMPSACKI bemenettel ugyan le tudtuk tiltani a következő írásengedélyező jelet, amíg az EEPROM befejezi az írást, de eközben a JTAG interfészen érkező további adatok nem jutottak el a memóriáig. Tehát az ACK jel nem állítja le a debuggert, amíg nincs kész az írás – a késleltetést a debug felület felől kellett megoldani.

A mérések azt mutatták, hogy a debugger 2 ms körüli időközönként küld a CPU felé olvasási (MEMPSRD) impulzusokat és a debuggerből kezdeményezett programmemória írásánál is ilyen időközökkel érkeznek az írást engedélyező jelek. Ez épp kevés az EEPROM írásához, de a page write mód használatához túl sok. A 2 ms-os ciklusidő egyik hardver dokumentációval sem volt összhangban, mint végül kiderült, ez az időköz a JTAG sebességéből adódott.

A megoldás kulcsa a JTAG TCK (Test Clock) órajelének lassítása volt. Az FS2 szoftver korábbi használata során figyelmen kívül hagytuk, hogy a JTAG soros interfész adatátviteli sebessége beállítható a programból; a dokumentációt [12]b átolvasva sikerült megtalálni, hogy az FS2 console parancssoros (CLI) felhasználói felületén a `config TckRate ?value?` parancs éppen erre szolgál (25. ábra). A hardveres késleltetést, illetve page write vezérlést így tehát elvetettük. A logikai analízátorral és a debuggerrel kimértük, melyik az a legnagyobb TCK frekvencia, amellyel még folytonosan írhatók be az EEPROM-ba, ez alapján a 250000 Hz-es érték bizonyult megfelelőnek. Ezáltal már sikerült egész programokat egyszerre beírni a memóriába. A továbbiakban tehát már rugalmasabban tudunk a mikrovezérlős platformra szoftvereket fejleszteni.

A processzor órajelét a hardverkonfigurációban egy egyszerű számlálóval leosztottuk 4 MHz-re, hogy az EEPROM olvasási ciklusába (150 ns, [14]) beleférjen a CPU ciklusideje, ezáltal folyamatosan futtatható legyen rajta a program.



```

Console - System Analyzer for Actel Core8051(tm) (simulated)
File Edit Prefs History Help
System Analyzer for Actel Core8051(tm) (simulated)
Serial number demo
Version 2.1.0 build 1
Copyright (C) 1998-2006 First Silicon Solutions, Inc.
1> help config
Syntax: config key ?value?
Action: Display or change configuration parameter.
Keys:
  BkptPoll          <milliseconds>
  Flash             <flash device name>
  JtagChain         see 'jtag' in on-line help (or Appendix B of manual)
  Path             installation path (do not change)
  Port             communications port (use openport command to change)
  ResetDuration    <milliseconds>
  TckRate          <Hertz>
  TvcThreshold     <millivolts>
2> config
{BkptPoll 100}
{Flash ""}
{JtagChain ""}
{Port lpt1}
{ResetDuration 100}
{TckRate 4000000}
{TvcThreshold 2500}
3> config TckRate 250000
250000
4>
Ready | Halted

```

25. ábra A Jtag interfész sebességének beállítása

3.7 A mikrokontroller felélesztése: az első működő program

Miután sikerült beállítani a JTAG TCK frekvenciáját az EEPROM írási sebességéhez, most már nekiláthattunk, hogy valamilyen egyszerű - és viszonylag kevés utasításból álló - programot írjunk, majd rátöltsük azt az EEPROM-ra a processzor debug interfészén keresztül, majd ellenőrizzük, hogy a processzor az általunk megkívánt működést produkálja-e. Úgy döntöttünk, a tesztprogramunk egy egyszerű négyszögjel-generátor lesz, amely egyik - LED-ek számára fenntartott - kimenetet billegteteti meghatározott időközönként 0 és 1 között.

A tesztprogramot assembly nyelven készítettük el. Az FPGA lábát a processzor APB buszára illesztett egyik GPIO modul kimenetére kötöttük. A láb billegtetéséhez tehát az APB buszra csatlakozó egyik GPIO perifériának regisztereit kellett vezérelnünk a 8051 processzorhoz tartozó assembly nyelven.

A számunkra szükséges FPGA-kimenetet arra a GPIO egységre kötöttük előzőleg, amelyet az APB buszvezérlő 4. slave csatlakozójára kötöttünk, ezért a számunkra fontos GPIO regiszterei az $0xF000+0x0400=0xF400$ címtől kezdődően helyezkedtek el.

A következőkben ismertetjük a GPIO egységek vezérlő regisztereinek a funkcióját és címkiosztását [17]. (Csak a 8 bites busz-szélességű esetet vizsgáljuk.)

PADDR[7:0]	Type	Reset Value (hex)	Brief Description
0x00-0x7C (0x00, 0x04, 0x08, ..., 0x7C)	R/W	0x00	8-bit configuration registers for all 32 bits; 1 register per bit.
0x80	W	0x00	Interrupt clear register 1 (bits 7:0)
0x84	W	0x00	Interrupt clear register 2 (bits 15:8)
0x88	W	0x00	Interrupt clear register 3 (bits 23:16)
0x8C	W	0x00	Interrupt clear register 4 (bits 31:24)
0x90	R	0x00	Input register 1 (bits 7:0)
0x94	R	0x00	Input register 2 (bits 15:8)
0x98	R	0x00	Input register 3 (bits 23:16)
0x9C	R	0x00	Input register 4 (bits 31:24)
0xA0	R/W	0x00	Output register 1 (bits 7:0)
0xA4	R/W	0x00	Output register 2 (bits 15:8)
0xA8	R/W	0x00	Output register 3 (bits 23:16)
0xAC	R/W	0x00	Output register 4 (bits 31:24)

26. ábra : GPIO regisztereinek címkiosztása 8 bites busz-szélességű esetben

Fent látható az említett címkiosztás (26. ábra : GPIO regisztereinek címkiosztása 8 bites busz-szélességű esetben). A GPIO-nak max. 32 db. digitális bemenete és kimenete van, de mi most csak 8-at használtunk. Ha adatot szeretnénk a kimenetekre írni, akkor a táblázatban szereplő 4 adatregiszternek kell megfelelő értéket adni, hogyha pedig a bemeneteket szeretnénk olvasni, akkor a 4 bemeneti regisztert kell olvasni. (8 bites esetben csak 1-1 regiszterről van szó). A kimeneti és bemeneti regiszterek bitjeit azonban használat előtt mindig inicializálni kell, erre szolgál a táblázat felső részén, vagyis a címtartomány alsó 32 címén elhelyezkedő 32 bájt. Minden egyes bithez egy 1 bájtos konfigurációs regiszter tartozik, melyben beállítható, hogy az adott helyi-értéken megjelenjenek a rákötött bejövő bitek, illetve a kimeneten láthatóvá váljanak az adott kimeneti regiszter adott bitjére kiírt értékek. A konfigurációs regiszterek tartalmát mutatja a 27. ábra. Az összes használt bit konfigurációs regiszterében bekapcsoltuk a bitek kimeneti megjelenését, melyhez a megfelelő GPIO APB címekre '00000101b'-t kellett írni. Ezek után a kimeneti regiszterek tartalmát változtattuk meghatározott időközönként. Az ezt megvalósító assembly kód látható a következő helyen: 28. ábra.

Bits	Name	Function
7:5	INTTYPE	Sets the interrupt type for this particular bit: 000 – Level High 001 – Level Low 010 – Edge Positive 011 – Edge Negative 100 – Edge Both 101 to 111 – Invalid
4	Reserved	Unused
3	INTENABLE	Interrupt enable for this particular bit 1 – Enable interrupt generation 0 – Disable interrupt generation
2	OUTBUFF	Sets the output enable for this particular bit, whether via the GPIO_OE signal or implemented internally (see parameter "OE_TYPE" on page 11). 1 – Enables output 0 – Disables output
1	INREG	Input register enable 1 – Enables input register for this particular bit 0 – Disables input register for this particular bit
0	OUTREG	Output register enable 1 – Enables output functionality for this particular bit 0 – Disables output functionality for this particular bit

27. ábra: 1 bithez tartozó konfigurációs regiszter tartalma

```

;négyszögjel generátor
mov sp,2Fh
;Stack pointer beállítása 2F-re, hogy ne írja felül a nagyobb címek felé
terjeszkedő stack a bitváltozúinkat,
;melyek 20h és 2FFh között lehetnek majd.

;A 4. APB sloton lévő GPIO 0. bitjéhez tartozó konfigurációs regiszter
beállítása:
mov dptr,#0F400h ;GPIO_APBS4
;Ha hexa címet írok be és nem nulla az eleje, elé kell írni egy nullát!
movx a,@dptr
orl a,#00000101b
movx @dptr,a

;Innentől változtatgatom a GPIO output 4. bitjének értékét 1 és 0 között
mov dptr,#0F4A0h

hurok1:
movx a,@dptr
orl a,#00000001b;GPIO_APBS4 bit_0->1
movx @dptr,a

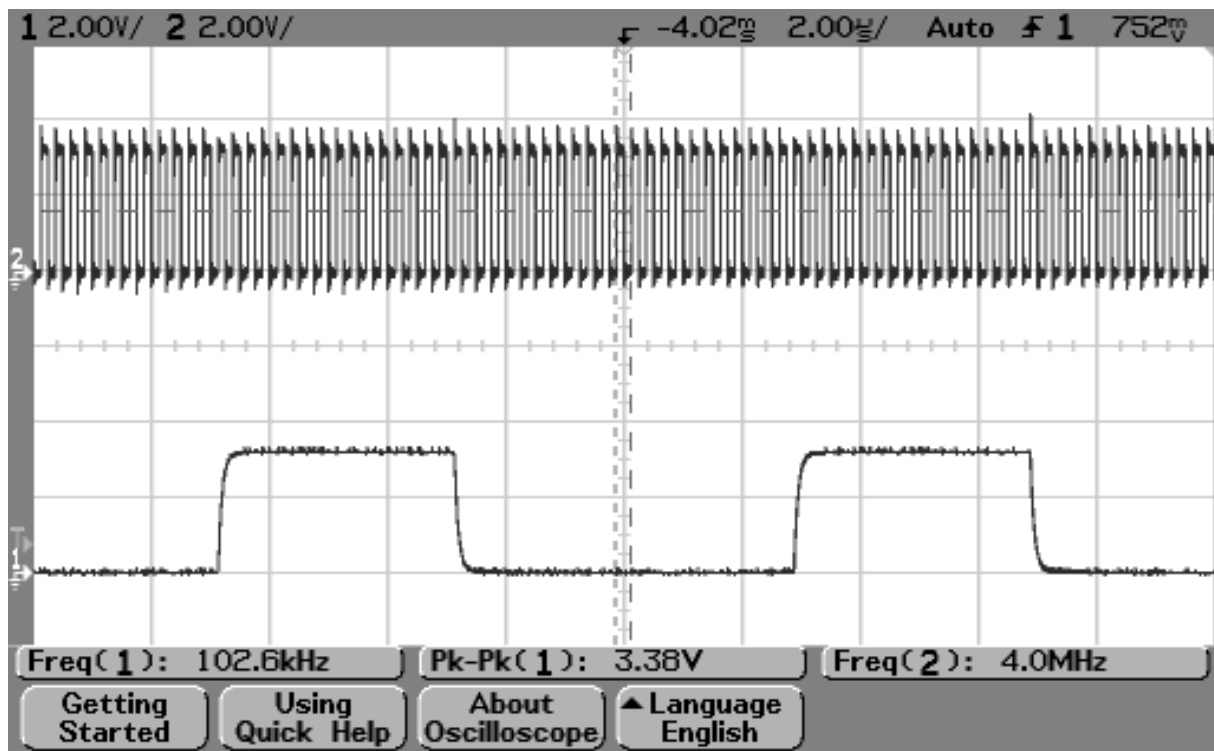
movx a,@dptr
anl a,#11111110b;GPIO_APBS4 bit_0->0
movx @dptr,a

ljmp hurok1
END

```

28. ábra: A négyszögjelet generáló assembly kód

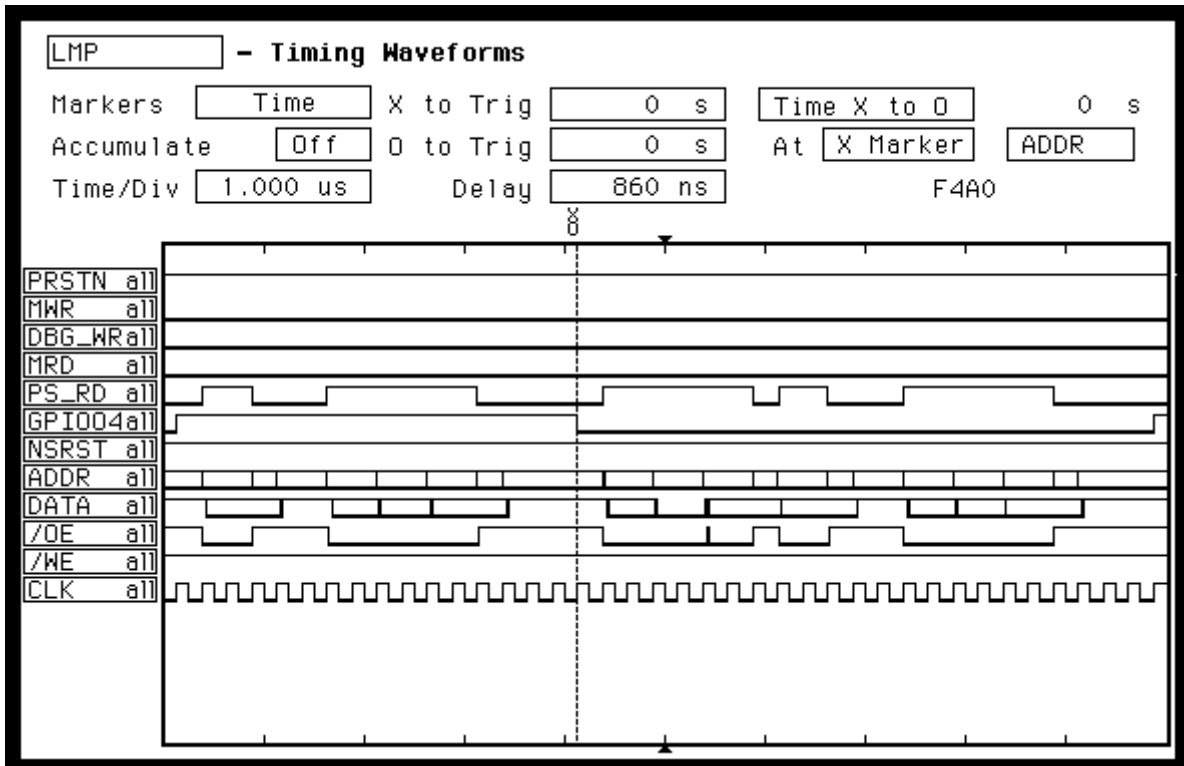
Végül, hogy megnézzük, hogy a kód valóban a kívánt működést valósítja meg, az adott kimenetre oszcilloszkópot csatlakoztattunk és megvizsgáltuk az FPGA-ból kijövő jelalakot.



29. ábra: A generált négyszögjel képe oszcilloszkópon

Az 1-es csatornán a program által ki-be kapcsolt jel, a 2-esen az órajel látható. Az adott konfigurációval tehát kb. 100 kHz frekvenciájú négyszögjelet tudunk szoftveresen generálni.

A jelet logikai analizátorral is megvizsgáltuk. Az analizátorba bekötöttük az EEPROM vezérlőjeleit is, ezért azok változását is megfigyelhetjük az alábbi ábrán:



30. ábra: A négyszögjel (GPIO04) és az EEPROM vezérlőjelek a logikai analízátoron

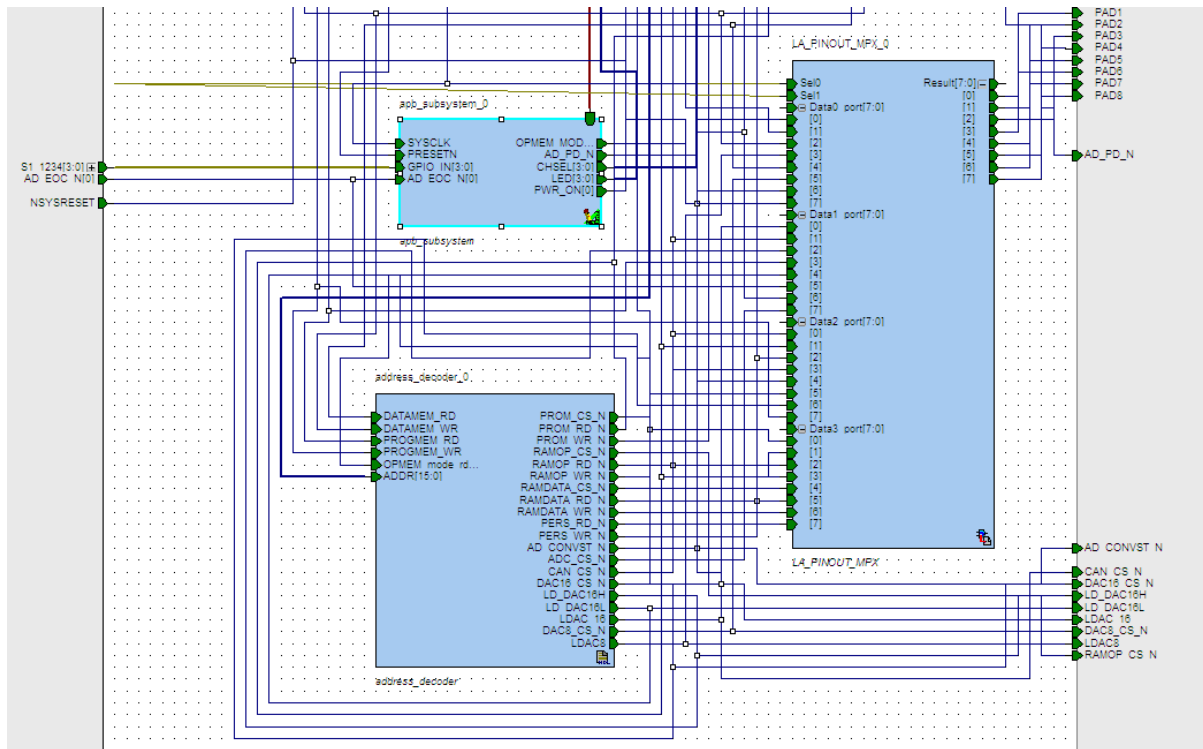
Ha alaposan megfigyeljük az ábrát, kivehető, hogy a PS_RD jel - az utasításle hívó jele a processzornak - mindig akkor aktiválódik, amikor egy bájtot hív le a processzor a programmemóriából. A képen az is látható, hogy a PS_RD szinkronizált az EEPROM OE_N lábával, ami szintén nem meglepő. Az analízátoron lehetőség volt az ADDR és DATA buszok értékének adott időpillanatbeli leolvasására, ezt azonban most nem részletezzük.

A következőkben a processzor Verilog nyelven megírt címdekóderének az ellenőrzéséről lesz szó.

3.8 A címdekódoló tesztelése assembly kóddal és logikai analízátorral

Eddig csak egyetlen EEPROM memóriát illesztettünk a processzorhoz. Ahhoz viszont, hogy minden memória és címtartományba ágyazott periféria kezelését biztonságosan meg tudjuk valósítani, szükséges volt, hogy megfelelően leteszteljük a korábban elkészített címdekóderet. A címdekóder teszteléséhez logikai analízátort használtunk, melynek során minden egyes periféria címezését ellenőriztük. Az lett volna az ideális, ha a processzor minden jelét és a címdekóder minden kimenetét egyszerre nyomon tudjuk követni a műszeren, ehhez azonban nem állt rendelkezésre elég bemenet.

Ennek a problémának az áthidalására a logikai tervünkben egy multiplexert helyeztünk el, melynek 4 darab 8 bites busz értéke közül mindig a kiválasztott sorszámú busz jelét adja a 8 bites kimenetén. A multiplexert a sel1 és sel0 bemenő bitekkel lehet vezérelni, ezeket a panelunk 2 kapcsolójára kötöttük. Így a multiplexerral a maradék 8 jel helyett 32 bitet tudunk megjeleníteni.



31. ábra: A címdekóder és a multiplexer összekapcsolása

A processzor jelei a multiplexer data0 bemeneti buszára csatlakoznak, a processzor az ábrára már sajnos nem fért rá, de az APB subsystem nevű modul felett helyezkedik el a tervben, amerre a vörös buszvezeték vezet. A multiplexer kimeneteit közvetlenül a NYÁK-on lévő tesztpontokra vezettük ki, melyekre tűskesor került felforrasztásra. Ezekre csatlakoztattuk a logikai analizátor mérővezetékét.

A címdekóderbe külön bemennek a processzornak az adat-olvasó (DATAMEM_RD), adat-író (DATAMEM_WR), programmemória-olvasó (PROGMEM_RD) és programmemória-író (PROGMEM_WR) kimenetei, melyek egy negálás után rögtön megjelennek a címdekóder kimenetén is.

A címdekóder kimenetén külön van választva a program SRAM-nak és az adatot tároló SRAM-nak a vezérlőjelei, ezeket azonban nem tartottuk szükségesnek tesztelni, mivel később - a memóriák kipróbálásánál - kiderül az esetleges hibás működés.

A perifériák a 2. konfigurációs üzemmódba való kapcsolás esetén válnak láthatóvá a külső "adat" címtartományban.

Ezért ahhoz, hogy a perifériákat el tudjuk érni, a - konfigurációs táblázat szerinti (20. ábra) -OPMEM_mode_rdprogot át kell állítani "1"-be. A perifériák PERS_RD_N és PERS_WR_N jele csak akkor tükrözi a DATAMEM_RD és DATAMEM_WR negáltjának értékét, hogyha az előző átállítást elvégezzük.

```

;;;OPMEM_mode_rdprogot először át kell állítani "1"-be
mov dptr,#0F700h ;Kiválasztom az Opmem_rdprog GPIO modul
; első bitjének a regiszterét.

movx a,@dptr
orl a,#00000101b ; Config regiszter beállítása, kimeneti regiszterek
movx @dptr,a

mov dptr,#0F7A0h ; GPIO 7.slot kimeneti reg. címének betöltése
movx a,@dptr
orl a,#00000001b ;eeprom_mode_rdprog bit->1

movx @dptr,a

```

Ezután következett annak az ellenőrzése, hogy az adott lábak aktiválódnak-e. Alapvetően 4 típusú kimenete volt a címdekódernek, Ezek közül mindegyikből csak 1-nek a tesztelését fogjuk bemutatni assembly kóddal együtt.

A címdekóder kimenetek első fajtái azok, amelyeknél már a megfelelő címnek a címbuszon való megjelenésének pillanatában működésbe kell lépniük. Ezek olyan perifériák elérését teszik lehetővé, melyeknek van RD és WR bemenete, ahova a PERS_RD és PERS_WR bemeneteket tudjuk kötni, így nem kell őket a CS.._N jelekbe belekapuzni. A kimenetek tesztelését az alábbi assembly kód írja le:

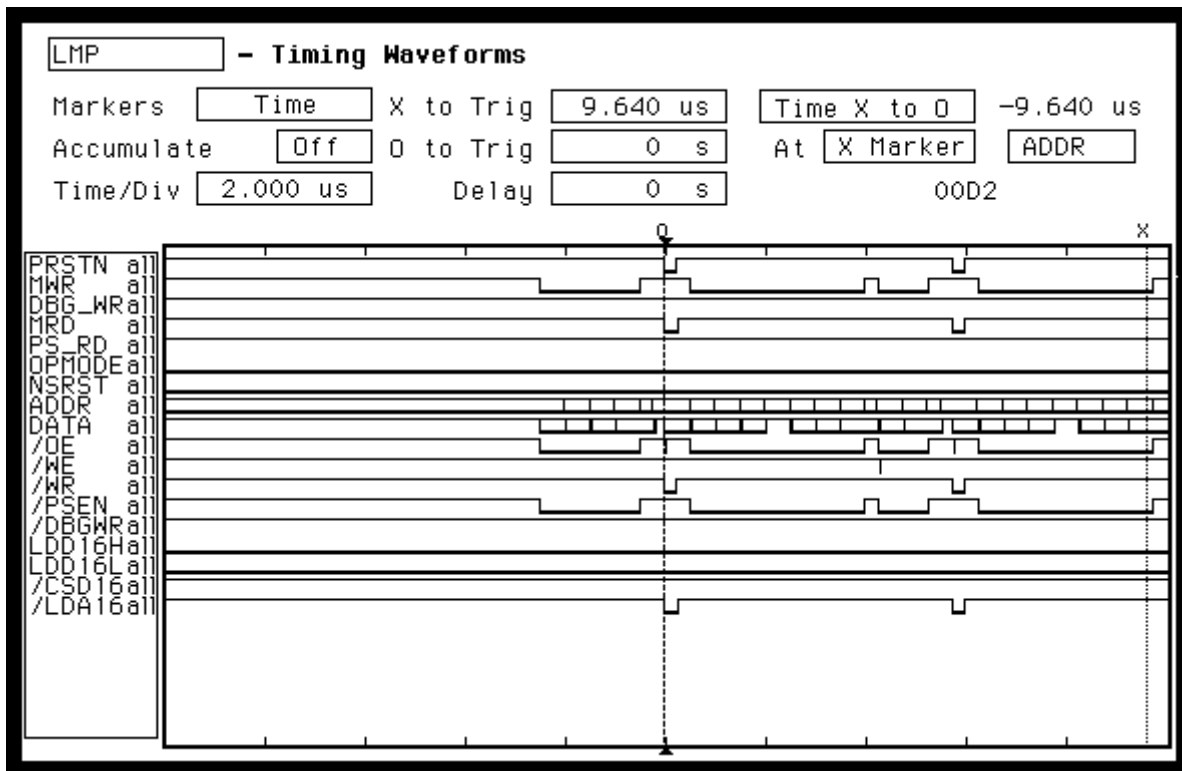
```

;;;ADC_CS_N tesztelése
mov dptr,#08000h
mov r1, #01

hurok01:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh ;akkor ad nullát, ha megegyeznek
inc dptr
jnz hurok01

```

Ebben a kódrészletben a 80xxH címen elhelyezkedő ADC_CD_N kimenetet teszteltük. Mivel a címdekóder kívánt működése alapján az eszköz csak a felső 8 bitet figyeli, ezért nekünk az egész 8000-80FF tartományra kell írásokat végeznünk, és igazolnunk a jel aktiválását logikai analízátorral.



32. ábra A címdekóder által előállított kiválasztó- és vezérlő jelek a 16 bites D/A konverterhez

Természetesen a logikai analízátor használata előtt a multiplexert a kapcsolók segítségével adott állapotba kell állítani, hogy a műszeren a kívánt jeleket lássuk.

Ugyanezzel a módszerrel tesztelhetők a többi, ebbe a csoportba tartozó lábak is, csupán a címüket kell a 80xxH helyére behelyettesíteni. Ezen lábak címzeit a rendszer FPGA-ba ágyazott verilog nyelven írt címdekóderében (melléklet:M4.) határoztuk meg, ezért innen kell programozáskor kiolvasnunk őket. Az alábbi verilog kódrészlet a címdekóderből való:

```
assign ADC_CS_N  =!(ADDR[15:8]==8'h80);
assign CAN_CS_N  =!(ADDR[15:8]==8'h88);
assign DAC8_CS_N =!(ADDR[15:8]==8'h86);
```

A második csoportba olyan perifériákhoz vezető kimenetek tartoznak, amelyeket csak akkor szabad aktiválni, hogyha az eszköz címe mellett megjelenik a PERS_RD_N jel "0" értéke is. Az ilyen kimeneteket az alábbi kóddal teszteltük:

```
;;;AD_CONVST_N tesztelése
mov  dptr,#08100h
mov  r1, #01h

hurok02:
mov  a,r1
movx a,@dptr
mov  a,dpl
xrl  a,#0FFh      ;akkor ad nullát, ha megegyeznek.
inc  dptr
jnz  hurok02
```

A perifériák nevének és címének megadása (részlet M4.-ből):

```
assign AD_CONVST_N =!(ADDR[15:8]==8'h81 && !PERS_RD_N);
```

Ugyanakkor van egy harmadik periféria-csoport is, ezek ugyanakkor lépnek működésbe, mint az előző esetben, csak a PERS_WR jellel vannak összekapuzva. Ezek a következő címeken helyezkedtek el (részlet M4.-ből):

```
assign DAC16_CS_N=(ADDR[15:8]==8'h84 && !PERS_WR_N);
assign LD_DAC16H=(ADDR[15:8]==8'h82 && !PERS_WR_N);
assign LD_DAC16L=(ADDR[15:8]==8'h83 && !PERS_WR_N);
assign LDAC16_N=(ADDR[15:8]==8'h85 && !PERS_WR_N);
assign LDAC8_N=(ADDR[15:8]==8'h87 && !PERS_WR_N);
```

Tesztelésük az alábbi assembly kóddal történt:

```
;;;DAC16_CS_N tesztelése
mov dptr,#08400h
mov r1, #01h

hurok04:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh ;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok04
```

Miután az összes kódrészlet hatását megvizsgáltuk a logikai analízátoron, megbizonyosodtunk arról, hogy az általunk írt címdekóder a specifikációnak megfelelően működik.

4 A digitális kártya funkcionális tesztelése

A címdekóder elkészítésével és bemérésével immár befejeztük az áramkör FPGA-ba ágyazott részének fejlesztését. E fejezet már a hardver-fejlesztési munkánk utolsó fázisát dokumentálja. Ez a fázis természetesen az volt, hogy először teszteltük az FPGA-n kívüli perifériák, majd az egész kártya együttes működését. Ehhez szükségképpen meg kellett írunk a processzorra az alacsony szintű program-részeket, amelyek vezérlik ezeket a külső modulokat.

4.1 A külső perifériák tesztelése

Minden perifériához saját tesztprogramot kellett készítenünk assembly nyelven. Ahhoz hogy az egyes perifériákat vezérlő tesztszoftverek működését a lehető legjobban el tudjuk magyarázni, elengedhetetlen, hogy bemutassunk néhány alacsony szintű programrészletet.

Minden olyan assembly program elején, amely a perifériákat használja, először el kell végezni a rendszeren néhány alapbeállítást, amelyekről már a korábbiakban volt szó.

Ahhoz, hogy a perifériákat a külső címtartományba ágyazva el tudjuk érni, át kell állítani a címdekóder OPMEM_MODE_RDPROG vezérlését a 2. konfigurációba. Ehhez egy GPIO lábat kell átállítanunk. Először az adott GPIO modul (F700) adott lábához (0.) tartozó regisztert kell úgy beállítani, hogy a láb kimenetként működjön és engedélyezve legyen, majd az adott GPIO modul kimeneti regiszterének a lábhoz tartozó bitjét kell átállítanunk.

```
mov sp,2Fh ;stack pointer beállítása
mov dptr,#0F700h ;OPMEM_MODE_RDPROG bit regisztere a GPIO-ban
```

```

movx a,@dptr
orl a,#00000101b
movx @dptr,a

mov dptr,#0F7A0h           ;bit 1: perifériák engedélyezése
mov a,#01h
movx @dptr,a

```

4.1.1 A 16 bites D/A konverter tesztelése

A 16 bites D/A konvertert úgy teszteltük, hogy egy fűrészfog alakú jelet generáltunk, majd oszcilloszkóppal mértük a konverter kimenetén az analóg jelet. Mivel a processzor adatbusza csak 8 bites, ezért a 16 bites értéket 2 lépésben kell a D/A bemeneteire kapcsolt két 8 bites latchbe beírni, melyek a 8200h és 8300h címen helyezkednek el. Ezután aktiválva a D/A periféria LDAC lábát, az átírja a 16 bites értéket a D/A belső latch-ébe. Végül utolsó lépésként a DAC16 8400h címére történő írással elvégzi az eszköz a D/A konverziót és az analóg kimeneten megjelenik a kívánt jel. A fűrészjelet egy ciklusban inkrementált változó értékével állítottuk elő.

```

;;;DAC16 teszt
loop1:

mov a,#0ffh

loop2:

mov dptr,#08200h           ;LD_DAC16H
movx @dptr,a

mov dptr,#08300h           ;LD_DAC16L
mov r1,a                   ;backup register a
mov a,#00h
movx @dptr,a
mov a,r1                   ;restore register a

mov dptr,#08500h           ;LDAC
movx @dptr,a               ;dummy

mov dptr,#08400h           ;CS_DAC16
movx @dptr,a               ;dummy

dec a
jnz loop2

ljmp loop1

```

4.1.2 A 8 bites D/A konverter tesztelése

A 8 bites DAC tesztelése ugyanúgy történt, mint a 16 bites D/A tesztelése, azzal a különbséggel, hogy itt a D/A-ba közvetlen írással be tudtuk tölteni az értékeket, nem kellett latcheket használni.

```

;;;DAC 8 teszt

loop1:

```

```

mov a,#0ffh

loop2:

mov dptr,#08600h          ;CS_DAC8, load input data
movx @dptr,a

mov dptr,#08700h          ;LDAC8, output analog data
movx @dptr,a              ;dummy

dec a
jnz loop2

ljmp loop1

```

4.1.3 A 8 bites A/D konverter tesztelése

Az A/D konverter tesztelését úgy végeztük, hogy az A/D konverter egy kiválasztott csatornájára kapcsolt jelből mintákat vettünk, a konvertált digitális értéket a D/A konverterrel visszaállítottuk, majd oszcilloszkóppal összehasonlítottuk az eredeti és a generált jelalakot.

Az A/D konverter esetében vezérelni kell a bemeneti multiplexer csatorna-kiválasztását is. E célra a 4 bites CHSEL GPIO-ra kötött jel szolgál. Az eddigiekhez hasonlóan először a CHSEL jel minden bitjéhez tartozó GPIO lábvezérlő regisztert kimenetként konfiguráltuk, majd a CHSEL 4 bites buszának címére (F4A0h) való kiírással beállítottuk, hogy az analóg bemenetek közül az analóg multiplexer a 0. analóg csatornát kapcsolja az ADC bemenetére. Az A/D power-down bitje (alacsony áramfelvételi módba történő kapcsolás) szintén egy GPIO lábba van kötve. Ezt az eddigiekhez hasonló módon 0-ba állítottuk, amivel engedélyezhető az A/D működése.

A folyamatos vezérlés minden ciklusában a 8100h (A/D AD_CONVST) címre történő olvasással indítható az A/D konverzió, majd az A/D EOC kimenetének figyelésével kell várakozni a konverzió végére. Ekkor beolvasható a konvertált érték a 8000h címről, majd továbbítható a D/A-ra. A teszt során a 16 bites DAC-t használtuk, a nagyobb helyiértékű byte vezérlésével. A teszt során az A/D konverter megfelelő működését tapasztaltuk.

```

;;;ADC teszt

mov dptr,#0F400h          ;CHSEL bit 1 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a
mov dptr,#0F404h          ;CHSEL bit 2 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a
mov dptr,#0F408h          ;CHSEL bit 3 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a
mov dptr,#0F40Ch          ;CHSEL bit 4 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a

mov dptr,#0F4A0h          ;CHSEL: csatorna kiválasztás
mov a,#00h

```



```

movx @dptr,a

mov dptr,#0F600h           ;AD_PD power down bit control register
movx a,@dptr
orl a,#00000101b
movx @dptr,a

mov dptr,#0F6A0h           ;0=power on, 1=power off
mov a,#00h
movx @dptr,a

loop1:
mov dptr,#08100h           ;AD_CONVST
movx @dptr,a

mov a,#255                 ;wait EOC
loop2:
dec a
jnz loop2

mov dptr,#08000h           ;AD_CS / konvertált érték beolvasása
movx a,@dptr
                               ;visszairás a DAC16-ra
mov dptr,#08200h           ;LD_DAC16H
movx @dptr,a

mov dptr,#08300h           ;LD_DAC16L
mov a,#00h
movx @dptr,a

mov dptr,#08500h           ;LDAC
movx @dptr,a               ;dummy

mov dptr,#08400h           ;CS_DAC16
movx @dptr,a               ;dummy

ljmp loop1

```

4.1.4 Az UART és az FTDI chip tesztelése

Az FTDI chip teszteléséhez először konfigurálni kell az UART vezérlő regisztereit, melyek az F200h címen helyezkednek el. Elsőnek a baud-rate beállítását végző, 8 bites baudval nevű értéket kell kiírni az F208H címre. Ezt az értéket a forráskód kommentjében megadott képlet szerint számoltuk ki ($CLK=4Mhz$). A beállítás után már írható az UART Tx (adás) regisztere, amely az F200 címen helyezkedik el. Az UART tesztelése után az adatgyűjtő már képes arra, hogy mérési adatsorokat küldjön át soros vonalon egy PC-re.

```

mov dptr,#0F208h           ;UART config: 8N1, 115200 baud
mov a,#19h                 ;baudval=CLK/(16*brate)-1=25.04216.~0x19
movx @dptr,a
mov dptr,#0F20Ch
mov a,#01h
movx @dptr,a

loop1:

mov a,#0ffh

```

```

loop2:

mov  dptr,#0f200h           ;UART Tx reg.
movx @dptr,a

mov  r3,a
mov  r1,dpl
mov  r2,dph
mov  dptr,#00000h
loop3:
inc  dptr
mov  a,dph
cpl  a
jnz  loop3
mov  dpl,r1
mov  dph,r2
mov  a,r3

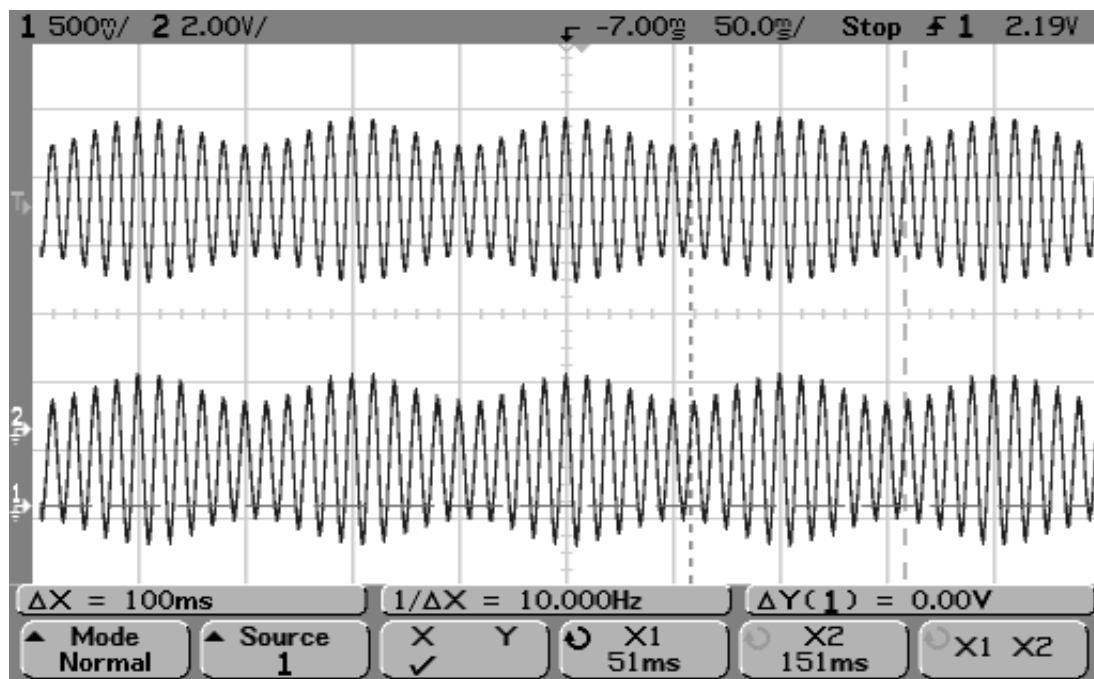
dec  a
jnz  loop2

ljmp loop1

```

4.2 A teljes kártya összetett működésének tesztelése

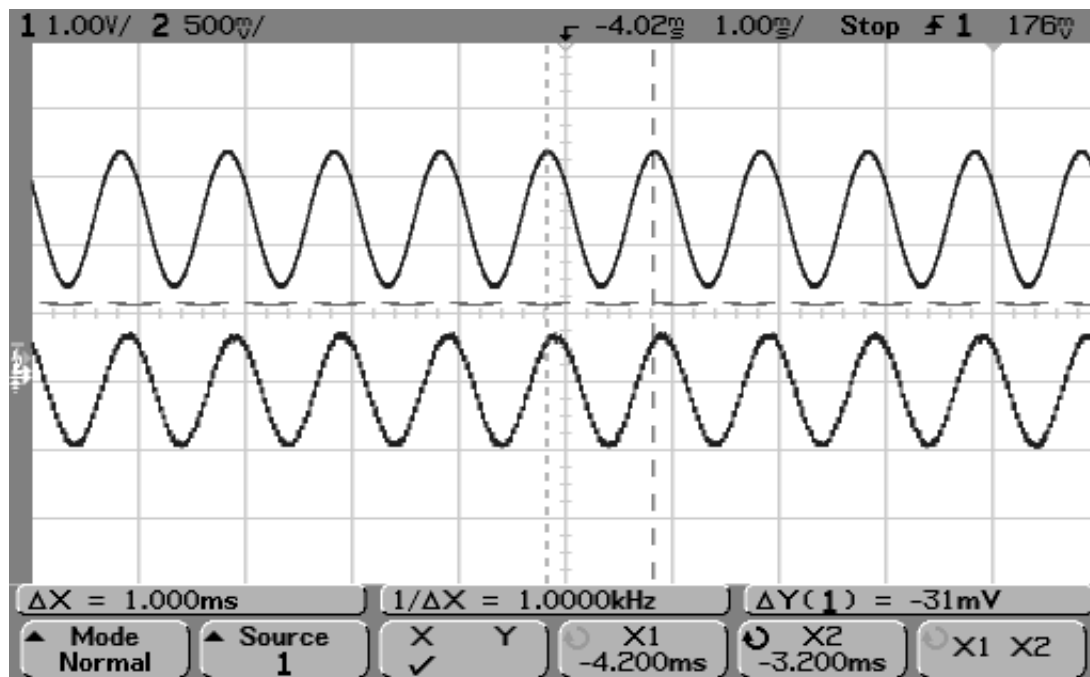
A végső funkcionális teszt volt az utolsó lépése a fejlesztési folyamatnak. A kártya analóg bemenetére 100Hz-es 10Hz-el 20% -ban modulált szinuszos AM jelet adtunk. A jelet mintavételeztük a kártyával, majd a 16 bites D/A segítségével visszaállítottuk. A mintákat ilyen sebességnél még folyamatosan tudtuk továbbítani a PC felé. A két jelet az oszcilloszkóp 2 csatornájára kapcsoltuk (33. ábra). Azt tapasztaltuk, hogy a rendszer megfelelően működik ebben a konfigurációban is.



33. ábra A 100Hz-es, 10 Hz-el modulált jel eredeti (felső) és visszaállított (alsó) változata

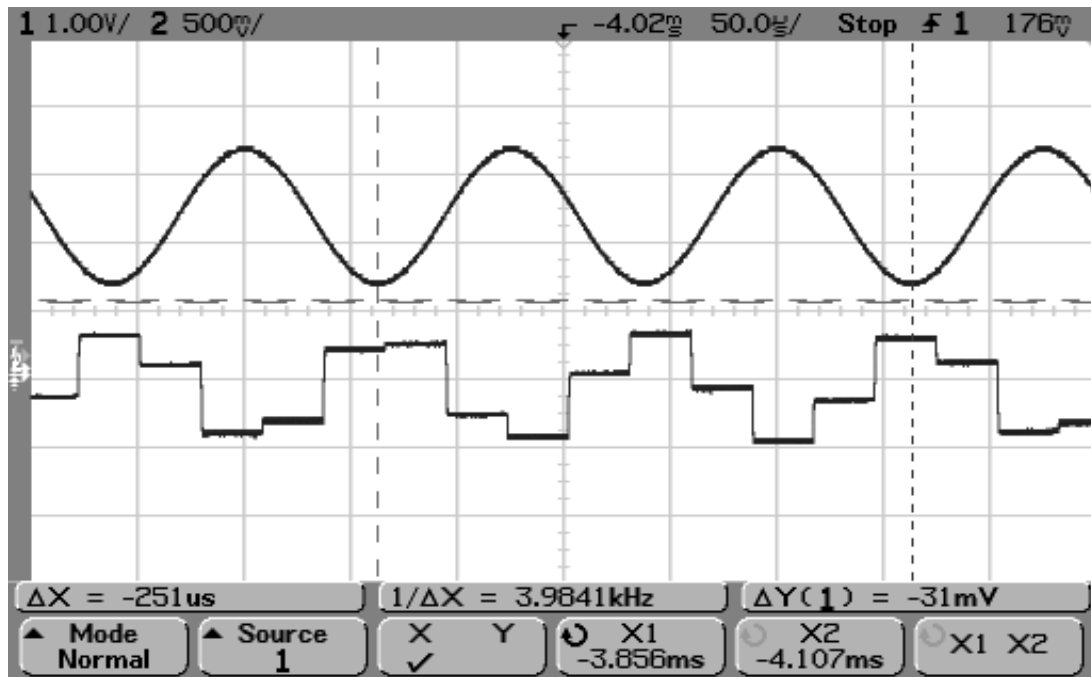
Ezután a kivettük a kódból azon sorokat, amelyek a mintákat minden ciklusban kiküldték a soros portra, és így is megfigyeltük a be és kimenetet. A rendszer egy 1kHz-es

szinuszos bemenő jelnél láthatóan jól működik (34. ábra). A bemenő jel frekvenciáját növelve természetesen egyre kevesebb mintát tud venni periódusonként, ami a kimenő jel alakján is megfigyelhető.

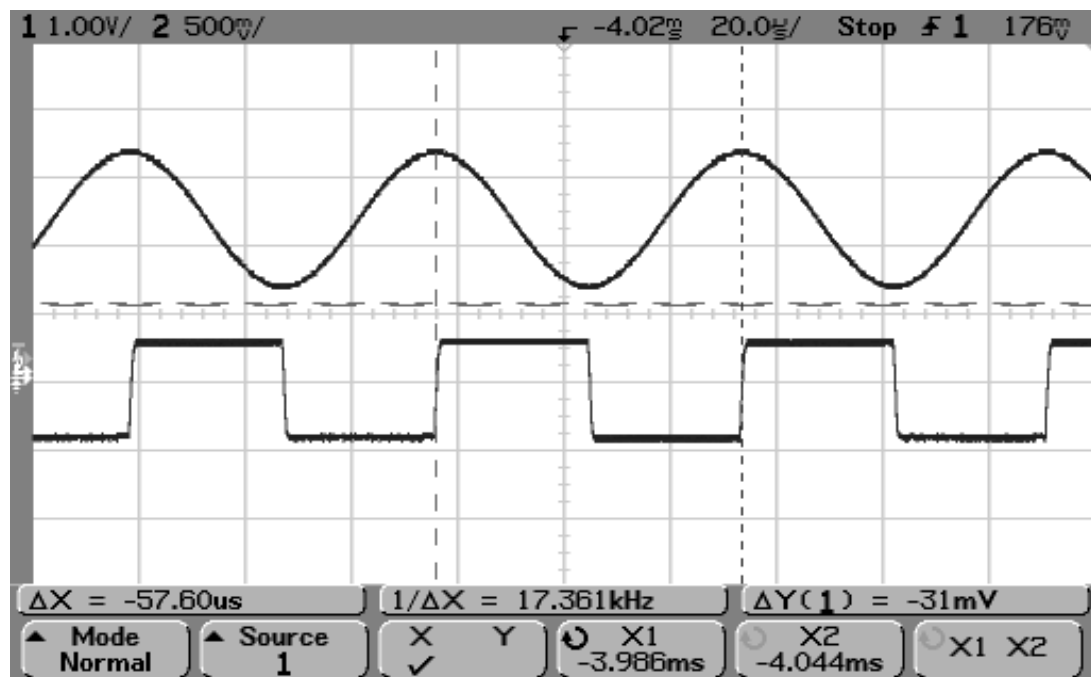


34. ábra 1kHz-es jel és visszaállítottja

Egy egyszerű szinusz bemenőjel segítségével közelítőleg megmértük a rendszer maximális működési frekvenciáját is. Ezt úgy tettük, hogy először beállítottunk egy olyan frekvenciát, ahol a kimenő jel már erőteljesen torzult (8kHz 35. ábra) és a bemenő jel 2 periódusa alatt megszámoltuk a kimenő jel hány mintából áll. Így megkaptuk, hogy átlagosan egy periódus 8.5/2 szakaszból áll. Ezt elosztottuk 2-vel, így ez a szám megmutatja, mennyivel emelhető még meg a frekvencia. Ezután addig emeltük tovább finom lépésekben a frekvenciát, amíg periódusonként 2 vízszintes szakaszt nem kaptunk a kimeneten (36. ábra). A Nyquist tétel értelmében ez a legnagyobb frekvencia, amelyet a rendszer még képes mintavételezni (17380 Hz). Ezt beszoroztuk 2-vel és a mintavételi frekvenciára kb. 2×17380 Hz adódott. Fontos még megemlíteni, hogy a koherens mintavételhez a bemenő jelnek egész számú periódusát kell mintavételezni, ezért amikor elkezdjük emelni a jel frekvenciáját, folyamatosan mozgó jelalakot kaptunk, mivel a jelnek mindig különböző fázisából vettünk mintát.



35. ábra A 8kHz-es jel és a már szakaszos kimenet



36. ábra: A maximális frekvenciájú szinusz és a visszaállított jel

```

;;;Teljes teszt

mov sp,2Fh

mov dptr,#0F700h           ;OPMEM_MODE_RDPROG regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a

mov dptr,#0F7A0h           ;bit 1: perifériák engedélyezése
mov a,#01h
movx @dptr,a

```

```
mov dptr,#0F400h           ;CHSEL bit 1 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a
mov dptr,#0F404h         ;CHSEL bit 2 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a
mov dptr,#0F408h         ;CHSEL bit 3 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a
mov dptr,#0F40Ch         ;CHSEL bit 4 regisztere
movx a,@dptr
orl a,#00000101b
movx @dptr,a

mov dptr,#0F4A0h         ;CHSEL: csatorna kiválasztás
mov a,#00h
movx @dptr,a

mov dptr,#0F800h         ;AD_EOC reg. olv. engedélyezés
movx a,@dptr
orl a,#00000010b
movx @dptr,a

mov dptr,#0F600h         ;AD_PD power down bit control register
movx a,@dptr
orl a,#00000101b
movx @dptr,a

mov dptr,#0F6A0h         ;0=power on, 1=power off
mov a,#00h
movx @dptr,a

mov dptr,#0F208h         ;UART config: 8N1, 115200 baud
mov a,#001h              ;baudval=CLK/(16*brate)-1=25.04216. 13 //~0x19
movx @dptr,a
mov dptr,#0F20Ch
mov a,#01h
movx @dptr,a

loop1:
mov dptr,#08100h         ;AD_CONVST
movx @dptr,a

mov a,#50                ;wait EOC
loop2:
dec a
jnz loop2

mov dptr,#0F8A0h         ;wait EOC
loop3:
movx a,@dptr
anl a,#01h
jnz loop3

mov dptr,#08000h         ;AD_CS / konvertált érték beolvasása
movx a,@dptr
```

```
; begyűjtött adat kiírása a soros portra
mov dptr,#0F200h          ;UART Tx reg.
movx @dptr,a

                                ;visszairás a DAC16-ra
mov dptr,#08300h          ;LD_DAC16H
movx @dptr,a

                                ;LD_DAC16L
mov dptr,#08200h
mov a,#00h
movx @dptr,a

                                ;LDAC
mov dptr,#08500h          ;dummy
movx @dptr,a

                                ;CS_DAC16
mov dptr,#08400h
movx @dptr,a

                                ;dummy

ljmp loop1
```

5 Összegzés

TDK dolgozatunkban az ESEO műhold egyik fedélzeti kísérletéhez, a Langmuir-szondához tartozó mérés-adatgyűjtő rendszer fejlesztését mutattuk be. A munkánk az adatgyűjtő FPGA-ba ágyazott rendszereinek a tervezése, megvalósítása, majd a teljes hardware megépítése és bemérése volt. A készüléknek az FPGA köré épített külső hardware elemei egy korábbi tervezési fázis eredményei, ebben és a nyomtatott áramkör tervezésében csak érintőlegesen vettünk részt.

Az FPGA-ban elhelyezett áramkör igen komplex, a kártya egészének alapvető működését befolyásoló egység. Ennek főbb elemei a 8051-es processzormag, a rácsatlakozó APB busz, GPIO modulok, UART és CAN kommunikációt biztosító egységek, továbbá címdekódoló logika és kisebb kiegészítő logikák. A rendszerélesztéshez szükséges processzor debug interface és az in-circuit program-memória írását lehetővé tevő áramkörök is a munkánk részét képezték.

A bemérés és tesztelés során a teljes rendszer működőképességét mérésekkel és szoftver tesztmodulok írásával bizonyítottuk. Az áramkör készen áll az LMP kísérlet adatának fogadására, és a végleges működtető szoftver feltöltésére.

A rendszer nem csak ebben a kísérletben használható, hanem alapja egy univerzális, könnyen konfigurálható és skálázható műhold fedélzeti mérés-adatgyűjtő rendszernek.

6 Köszönetnyilvánítás

Ezúton szeretnénk köszönetet mondani azoknak, akik segítettek a szakmai munkánkat az önálló tevékenységünk végzése folyamán. Szeretnénk köszönetet mondani a konzulensünknek, Dr. Csurgai-Horváth Lászlónak a folyamatos konzultációért, Kocsis Gábornak és Szimler Andrásnak a szakmai segítségükért és tanácsaikért, valamint a BME Úrkutató Csoport és az ESEO fejlesztői közösség minden olyan tagjának, aki segített nekünk.

7 Irodalomjegyzék

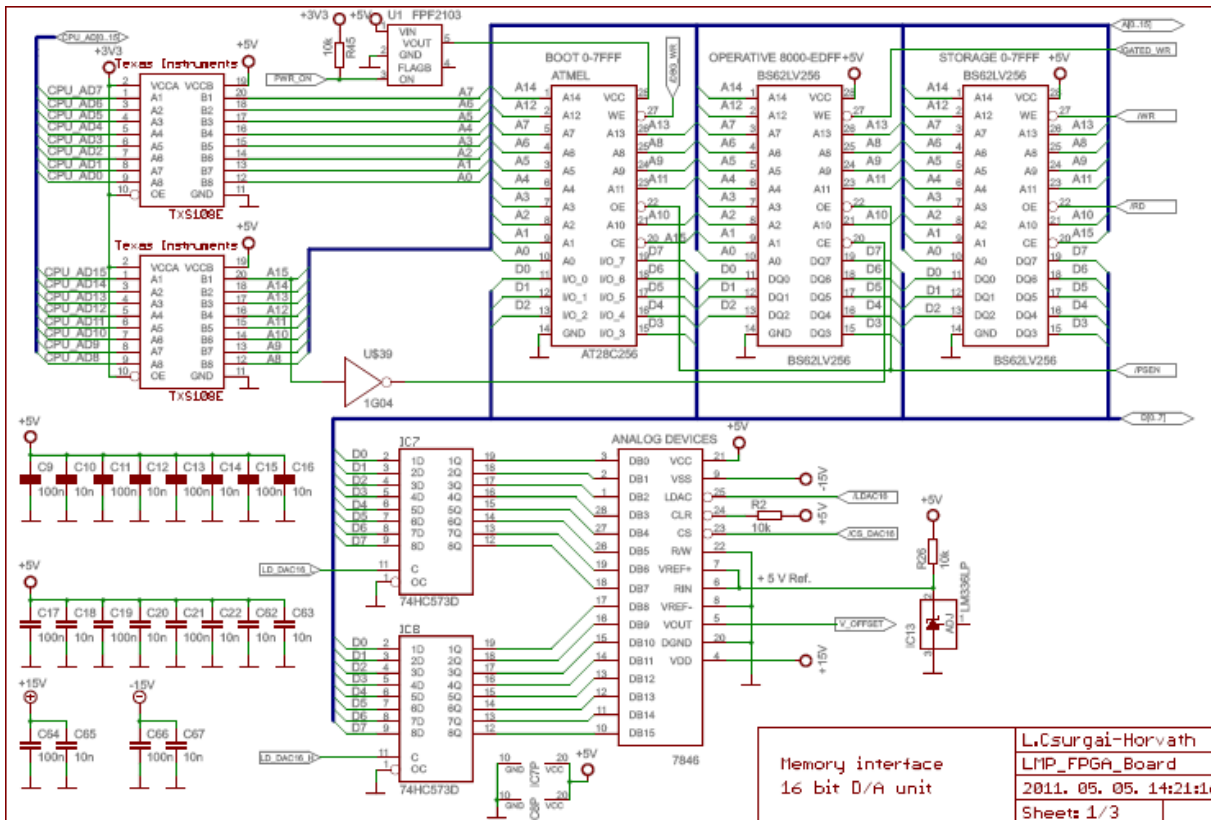
- [1] ESA oktatási részlegének honlapja / ESEO küldetés tájékoztató oldala
http://www.esa.int/esaMI/Education/SEM4DLPR4CF_0.html
- [2] Sík András: „Magyar műszerek Európa egyetemi műholdján”
<http://www.origo.hu/tudomany/20100311-eseo-magyar-muszerek-europa-egyetemi-muholdjan.html>
- [3] Gubicza Ágnes és Király Richárd: Az Ionoszféra Vizsgálata Langmuir Szondával
TDK dolgozat 2010.
- [4] ESA űrmérnök/rendszermérnök standard (ESA dokumentációk írásának szabályai):
ESA – Space Engineering: System engineering general requirements (ECSS-E-ST - 10)
- [5] CGS System Engineering and University Coordination Team / LMP Team (BME) - LMP
Subsystem Design Definition File (ESEO-TN-LMP-002)
- [6] Ludmány András: Bevezetés az űrfizikába, jegyzet, Debreceni Egyetem, 2008
<http://fenyi.sci.klte.hu/oktatas/F2701jegyzet/Urfizika.pdf>
- [7] Kenéz Lajos, Karácsony János: Plazmadiagnosztikai kutatások,
Elektron Ciklotron Rezonancia Ionforráson
http://epa.oszk.hu/00000/00028/00009/pdf/musze_EPA00028_2001_15_017-028.pdf
- [8] Rózsa Sándor: FPGA-k űrbéli alkalmazástechnikája:
http://www.mht.bme.hu/~csurgai/urtech/2010_osz/Rozsa_FPGAinSpace_2010_09_22.pdf
- [9] Actel Proasic 3 FPGA család dokumentációja: http://www.actel.com/documents/PA3_DS.pdf
- [10] Az Űrtechnológia c. tantárgy anyagai [<http://hvt.bme.hu/~csurgai/urtech>]
- [11] Wikipedia – VHDL
<http://en.wikipedia.org/wiki/VHDL>
- [12] Actel Core 8051s dokumentáció:
<http://www.actel.com/products/ip/search/detail.aspx?id=648>
 - a. handbook:
http://www.actel.com/ipdocs/Core8051s_HB.pdf
 - b. ISA Actel-51-debugger:
http://www.actel.com/download/reg/default.aspx?f=ISA_Actel51
- [13] ASEM-51 macro assembler dokumentáció:
<http://plit.de/asem-51>
- [14] Atmel AT28C256 EEPROM dokumentáció:
http://www.atmel.com/dyn/resources/prod_documents/doc0270.pdf
- [15] Needhams EMP-20 programozó leírása:
<http://www.chipcad.hu/arpgm.htm>
- [16] Actel CoreAPB3 dokumentáció
http://www.actel.com/ipdocs/CoreAPB3_HB.pdf
- [17] Actel GoreGPIO dokumentáció:
http://www.actel.com/ipdocs/CoreGPIO_HB.pdf

MELLÉKLETEK

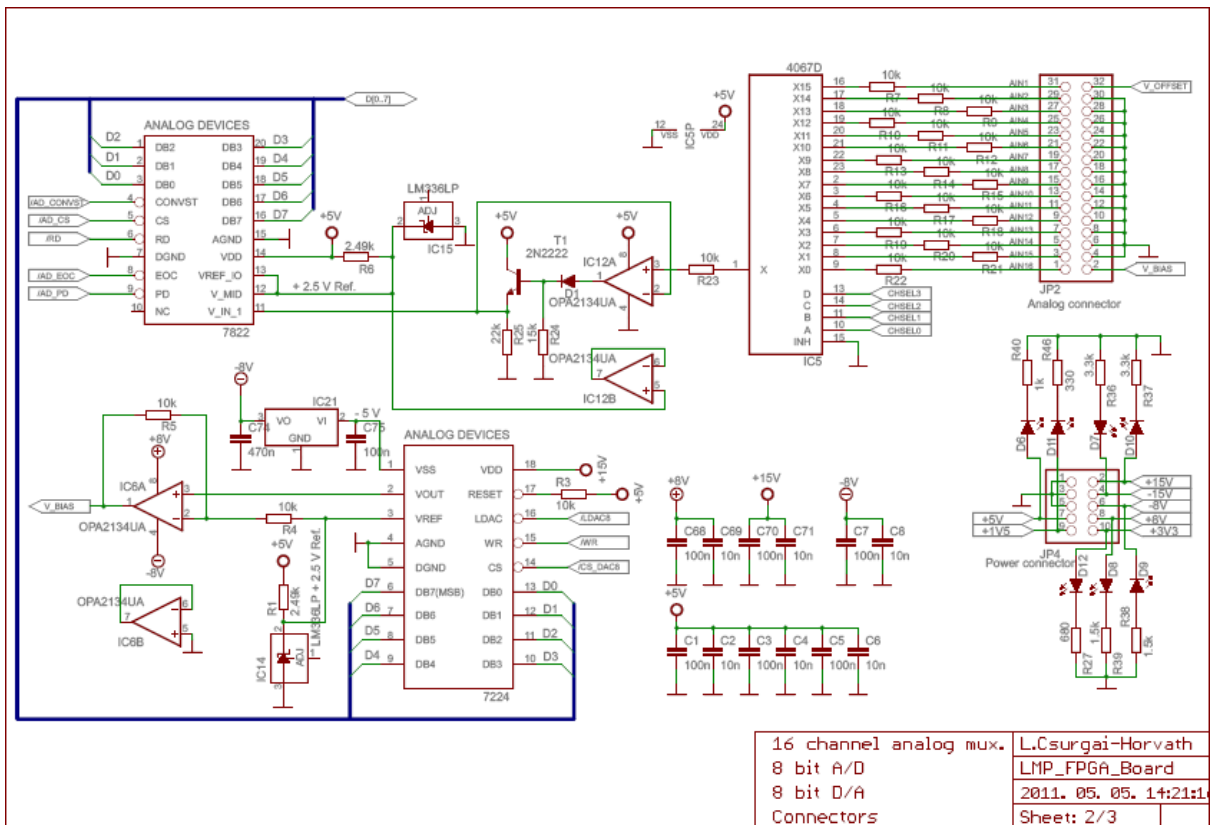
Lásd: következő oldalakon.

M1. Az LMP digitális kártya kapcsolási rajza

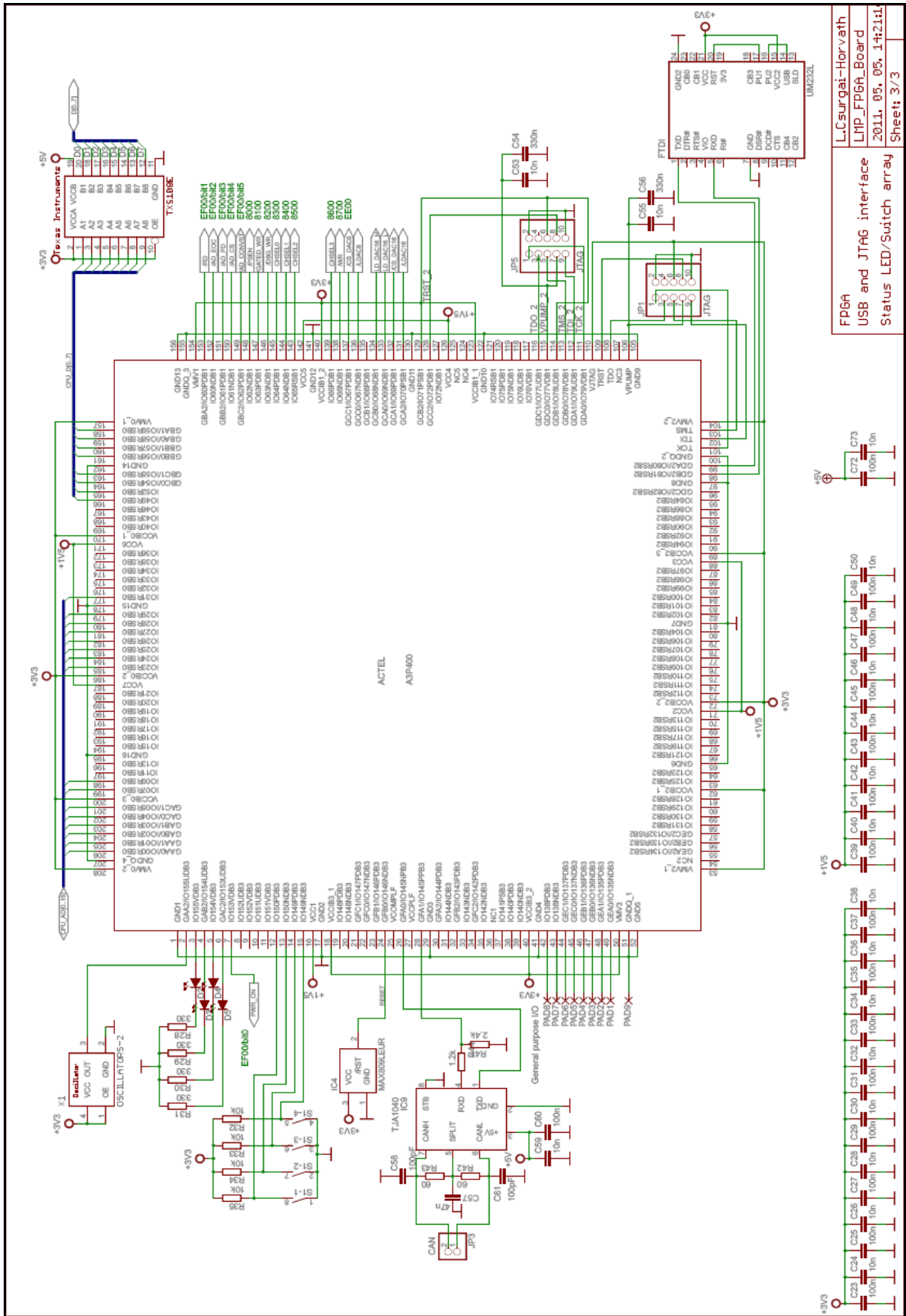
- M1.1 – memory interface + 16 bit D/A unit



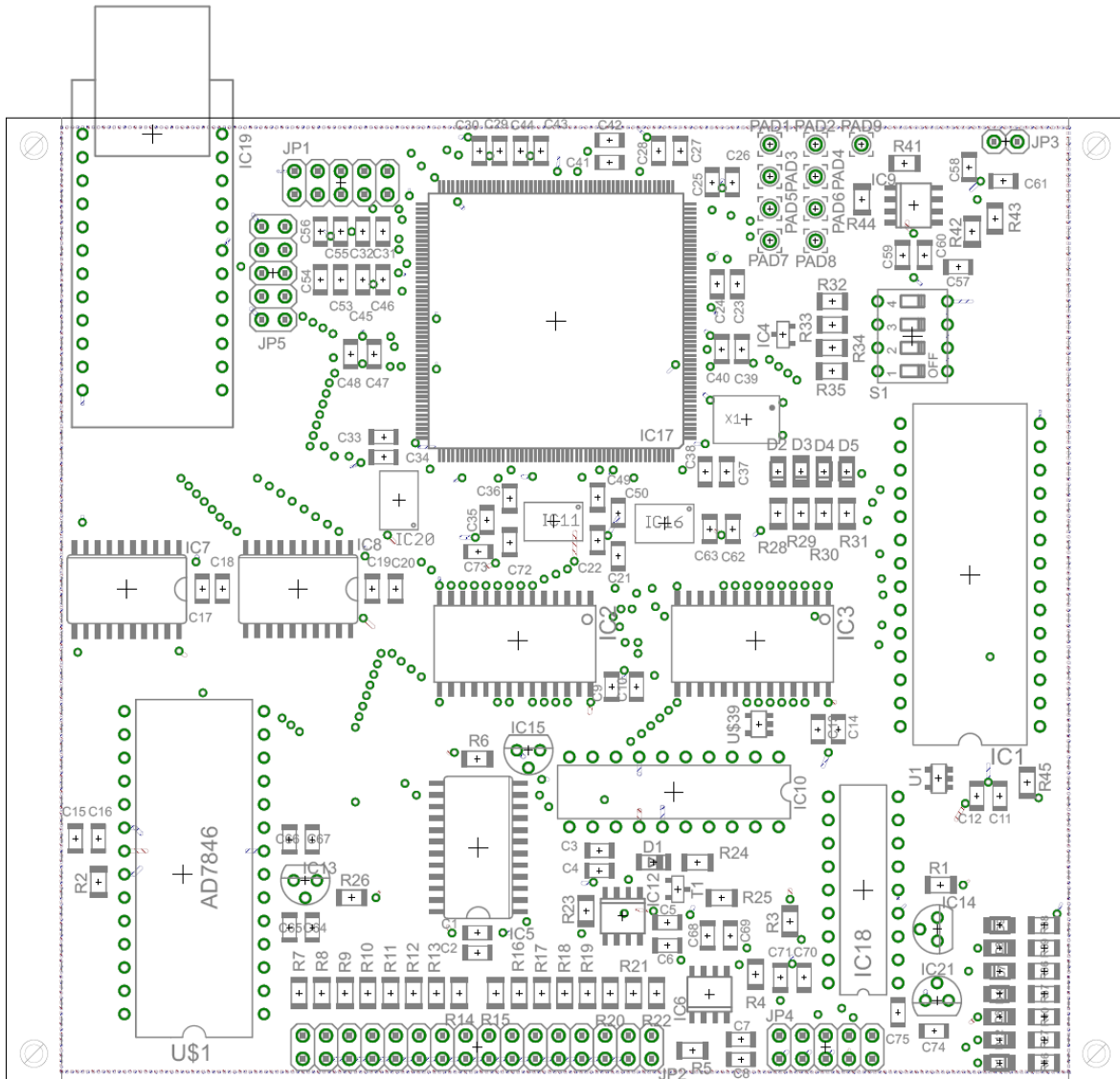
M1.2 – 16 channel analog mux. + 8 bit A/D + 8 bit D/A + connectors



M1.3 – FPGA + ISB and JTAG interface + Status LED/Switch array



M2. Az LMP digitális kártya beültetési rajza (layout)



M3. Alkatrészlista a beültetési rajzhoz

ID	alkatrész típusa	pozíció a nyákon	ID	alkatrész típusa	pozíció a nyákon
C1	100n	(51.45, 16.196)	C44	10n	(56.15, 101.45)
C2	10n	(51.45, 13.996)	C45	100n	(38.9, 87.4)
C3	100n	(64.75, 25.154)	C46	10n	(41.1, 87.4)
C4	10n	(64.746, 22.946)	C47	100n	(40.15, 79.28)
C5	100n	(72.154, 17.304)	C48	10n	(37.6, 79.28)
C6	10n	(72.154, 14.802)	C49	100n	(64.55, 63.65)
C7	100n	(80.2, 4.55)	C50	10n	(66.8, 61.95)
C8	10n	(80.2, 2.35)	C53	10n	(36.5, 87.4)
C9	100n	(66.096, 42.998)	C54	330n	(34.3, 87.4)
C10	10n	(68.808, 42.998)	C55	10n	(36.5, 92.65)
C11	100n	(108.446, 31.098)	C56	330n	(34.3, 92.65)
C12	10n	(105.944, 31.102)	C57	47n	(103.95, 88.8)
C13	100n	(88.598, 38.348)	C58	100pF	(105.1, 99.65)
C14	10n	(90.85, 38.354)	C59	10n	(97.85, 90.05)
C15	100n	(7.548, 26.446)	C60	100n	(100.25, 90.05)
C16	10n	(10.046, 26.45)	C61	100pF	(108.85, 98.15)
C17	100n	(21.398, 53.7)	C62	10n	(79.3, 60.2)
C18	10n	(23.598, 53.7)	C63	10n	(76.8, 60.2)
C19	100n	(39.948, 53.7)	C64	100n	(33.4, 16.75)
C20	10n	(42.5, 53.704)	C65	10n	(30.9, 16.75)
C21	100n	(66.8, 57.254)	C66	100n	(30.9, 26.35)
C22	10n	(64.552, 58.954)	C67	10n	(33.4, 26.35)
C23	100n	(79.8, 86.9)	C68	100n	(76.55, 15.85)
C24	10n	(77.6, 86.9)	C69	10n	(79.05, 15.85)
C25	100n	(77.05, 98)	C70	100n	(87.1, 11.3)
C26	10n	(79.25, 98)	C71	10n	(84.5, 11.3)
C27	100n	(73.6, 101.45)	C72	100n	(54.95, 58.75)
C28	10n	(71.2, 101.45)	C73	10n	(51.5, 57.75)
C29	100n	(53.85, 101.45)	C74	470n	(101.3, 5.45)
C30	10n	(51.65, 101.45)	C75	100n	(97.35, 7.5)
C31	100n	(41.1, 92.65)	D1		(70.646, 23.856)
C32	10n	(38.9, 92.65)	D2		(84.245, 66.455)
C33	100n	(41.2, 70.25)	D3		(86.75, 66.5)
C34	10n	(41.2, 68.05)	D4		(89.25, 66.45)
C35	100n	(52.45, 61.2)	D5		(91.75, 66.455)
C36	10n	(54.95, 63.55)	D6		(108.45, 7)
C37	100n	(78.65, 66.45)	D7		(108.45, 12)
C38	10n	(76.3, 66.45)	D8		(108.45, 14.5)
C39	100n	(80.3, 79.8)	D9		(108.45, 17)
C40	10n	(78.1, 79.8)	D10		(108.45, 9.5)
C41	100n	(65.85, 100.2)	D11		(108.45, 2)
C42	10n	(65.85, 102.6)	D12		(108.45, 4.5)
C43	100n	(58.35, 101.45)	IC1	ATMEL_AT28C256	(105.22, 55.2)

ID	alkatrész típusa	pozíció a nyákon	ID	alkatrész típusa	pozíció a nyákon
IC2	BS62LV256SOP	(55.85, 48.05)	R10	10k	(39.466, 9.612)
IC3	BS62LV256SOP	(81.8, 48.05)	R11	10k	(41.958, 9.612)
IC4	MAX809LEUR	(84.8, 81.4)	R12	10k	(44.45, 9.612)
IC5	4067D	(51.51, 25.488)	R13	10k	(46.942, 9.612)
IC6	OPA2134UA	(76.766, 9.464)	R14	10k	(49.434, 9.612)
IC7	74HC573D	(13.146, 53.652)	R15	10k	(53.426, 9.612)
IC8	74HC573D	(31.85, 53.656)	R16	10k	(55.968, 9.612)
IC9	TJA1040	(99.05, 95.5)	R17	10k	(58.46, 9.612)
IC10	AD7822BR(R20)	(72.888, 31.54)	R18	10k	(60.952, 9.612)
IC11	TXS0108E	(59.73, 61)	R19	10k	(63.444, 9.612)
IC12	OPA2134UA	(67.288, 18.004)	R20	10k	(65.936, 9.612)
IC13	LM336LP	(32.634, 21.902)	R21	10k	(68.478, 9.612)
IC14	LM336LP	(101.644, 16.562)	R22	10k	(70.97, 9.612)
IC15	LM336LP	(56.992, 36.086)	R23	10k	(63.292, 18.556)
IC16	TXS0108E	(71.85, 60.8)	R24	15k	(75.452, 23.858)
IC17	ACTEL_A3P400_PQFP208	(60, 82.9)	R25	22k	(78.1, 19.998)
IC18	AD72247AP	(93.55, 20.85)	R26	10k	(37.686, 20.008)
IC19	FTDI_UM232R	(15.93, 103.29)	R27	680	(113.3, 4.5)
IC20	TXS0108E	(42.85, 63.33)	R28	330	(84.25, 61.9)
IC21		(101.6, 8.8)	R29	330	(86.75, 61.9)
JP1	JTAG	(36.52, 97.97)	R30	330	(89.25, 61.9)
JP2	Analog	(51.4, 3.7)	R31	330	(91.75, 61.905)
JP3		(109.1, 102.45)	R32	10k	(90.18, 85.05)
JP4	Power	(89.45, 3.7)	R33	10k	(90.17, 82.48)
JP5	JTAG	(29.13, 88.12)	R34	10k	(90.16, 79.96)
PAD1	WIREPAD2,54/1,0	(83.35, 102.15)	R35	10k	(90.14, 77.44)
PAD2	WIREPAD2,54/1,0	(88.35, 102.15)	R36	3.3k	(113.3, 12)
PAD3	WIREPAD2,54/1,0	(83.35, 98.65)	R37	3.3k	(113.3, 9.5)
PAD4	WIREPAD2,54/1,0	(88.35, 98.65)	R38	1.5k	(113.3, 17)
PAD5	WIREPAD2,54/1,0	(83.35, 95.15)	R39	1.5k	(113.3, 14.5)
PAD6	WIREPAD2,54/1,0	(88.35, 95.15)	R40	1k	(113.3, 7)
PAD7	WIREPAD2,54/1,0	(83.35, 91.65)	R41	2.4k	(98.05, 100.1)
PAD8	WIREPAD2,54/1,0	(88.35, 91.65)	R42	60	(105.45, 92.65)
PAD9	WIREPAD2,54/1,0	(93.35, 102.15)	R43	60	(107.95, 94.05)
R1	2.49k	(101.978, 21.398)	R44	1.2k	(93.4, 96.15)
R2	10k	(10.048, 21.704)	R45	10k	(111.45, 32.55)
R3	10k	(85.548, 17.404)	R46	330	(113.3, 2)
R4	10k	(81.802, 11.648)	S1	DIP04S	81.24), R90
R5	10k	(74.91, 3.348)	T1	2N2222	(73.296, 20.95)
R6	2.49k	(51.396, 35.148)	U\$1	AD7846PDIP28	(19.222, 22.56)
R7	10k	(31.94, 9.612)	U\$39	1G04	(82.15, 38.93)
R8	10k	(34.432, 9.612)	U1	FPF2103	(101.8, 33)
R9	10k	(36.974, 9.612)	X1	OSCILLATORS-2	(80.8, 72.15)

M4.: A címdekóder verilog forráskódja:

```
//address decoder.v

module address_decoder(
input [15:0] ADDR,          //Memory Address
input DATAMEM_RD,         //data_rd           MEMRD
input DATAMEM_WR,         //data_wr           MEMWR
input PROGMEM_RD,         //Program memory read  MEMPSRD
input PROGMEM_WR,         //Program memory write DBGMEMPSWR
input OPMEM_mode_rdprog, // GPIO-ról jön, ez vezérli, hogy a RAMOP
adatmemóriaként látszon,
                //Program memory ==1
                // vagy adatmemóriaként ==0
                //Programmémória módban csak olvasni lehet.
                //Cim legyen ugyanaz adatmemória és programmemóriaként is

//output PROM_CS_N, //EEPROM
output PROM_RD_N,     //MEMPSRD_N EEPROM /OE
output PROM_WR_N,     //DBGMEMPSWR_N EEPROM /WE

//output RAMOP_CS_N, //Operativ mem
//output RAMOP_RD_N,
output RAMOP_WR_N,    //GATED_WR_N

//output RAMDATA_CS_N, //Adattarolo mem
output RAMDATA_RD_N, //MEMRD_N SRAM /OE
output RAMDATA_WR_N, //MEMWR_N SRAM /WE

//output PERS_RD_N,     //Periferiak olvasása
//output PERS_WR_N,     //Perifériák írása

output AD_CONVST_N,
output ADC_CS_N,
output CAN_CS_N,

output DAC16_CS_N,
output LD_DAC16H,
output LD_DAC16L,
output LDAC16_N,

output DAC8_CS_N,
output LDAC8_N
);
wire PERS_RD_N;
wire PERS_WR_N;

assign PROM_RD_N=~PROGMEM_RD;          //MEMPSRD
assign PROM_WR_N=~PROGMEM_WR;          //DBGMEMPSWR

assign RAMDATA_RD_N=~DATAMEM_RD;       //MEMRD
assign RAMDATA_WR_N=~DATAMEM_WR;       //MEMWR

// Amikor az operativ memoriat adat memoriakent hasznalom,
// (OPMEM_mode_rdprog=0)
// akkor a periferiak RD WR jat letiltom, mert a 8000 cim fole vannak
// bemappelve.
// Mikor az operativ memoriaba atmasolodott a program, es mar a processzort
// atvaltottam, hogy programmemorianak hasznalja az op. memoriat
// (OPMEM_mode_progrd=1)
```



```
// akkor mar engedelyezem a perifériák RD/WR jeleit

// assign RAMOP_RD_N!=(DATAMEM_RD && !OPMEM_mode_rdprog);
// Ezt nem tudjuk használni (külső adarmemóriaként olvasni), mert a 8000es
// című SRAMba csak a /PSEN van bekötve

assign RAMOP_WR_N!=(DATAMEM_WR && !OPMEM_mode_rdprog); //GATED_WR
// Op. ram WRje es RDje akkor lesz az adatrd es adatwr ertekevel
// megegyezo,
// hogyha az operativ memoriat eppen NEM programlehivo modban akarjuk
// hasznalni

assign PERS_RD_N!=(DATAMEM_RD && OPMEM_mode_rdprog); //Periferiak olvasasa
assign PERS_WR_N!=(DATAMEM_WR && OPMEM_mode_rdprog); //Periferiak irasa
// Periferiak WR és RD-je akkor a sima DATAMEMWR és DATAMEMRD értéke,
// amikor az operativ memoriat programlehivo modban akarjuk hasznalni

//assign PROM_CS_N= (!(ADDR[15]));
//assign RAMDATA_CS_N= (!(ADDR[15]));
//assign RAMOP_CS_N= !ADDR[15];

// 8 bites ADC jelei
assign ADC_CS_N =!(ADDR[15:8]==8'h80);
assign AD_CONVST_N=!(ADDR[15:8]==8'h81 && !PERS_RD_N);

// CAN i/f kiválasztó jel
assign CAN_CS_N=!(ADDR[15:8]==8'h88);

//16 bites DAC jelei:
assign DAC16_CS_N=!(ADDR[15:8]==8'h84 && !PERS_WR_N);
assign LD_DAC16H=(ADDR[15:8]==8'h82 && !PERS_WR_N);
assign LD_DAC16L=(ADDR[15:8]==8'h83 && !PERS_WR_N);
assign LDAC16_N=!(ADDR[15:8]==8'h85 && !PERS_WR_N);

//8 bites DAC jelei
assign DAC8_CS_N=!(ADDR[15:8]==8'h86);
assign LDAC8_N=!(ADDR[15:8]==8'h87 && !PERS_WR_N);

endmodule
```

M5. :címdékóder tesztelő teljes assembly program forráskódja: (melléklet/lmp04.a51)

```

;;;kód a címdékóder kipróbálásához

mov sp,2Fh
;Stack pointer beállítása 2F-re, hogy ne írja felül a nagyobb címek felé
;terjeszkedő stack a bitváltozúinkat,
;melyek 20h és 2Fh között lehetnek majd.

;A sima címtartományba ágyazott perifériák tesztelése:

;OPMEM_mode_rdprogot először is át kell állítani "1"-be
mov dptr,#0F700h ;Kiválasztom az Opmem_rdprog GPIO moduljának
; az első bitjének a regiszterét.
movx a,@dptr
orl a,#00000101b; Config regiszter beállítása a kimeneti regiszterek
movx @dptr,a

mov dptr,#0F7A0h; GPIO 7.slotjának kimeneti reg. címének betöltése
movx a,@dptr
orl a,#00000001b ;eeprom_mode_rdprog bit->1

movx @dptr,a

;;;ADC_CS_N tesztelése
mov dptr,#08000h
mov r1, #01

hurok01:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok01

;;;AD_CONVST_N tesztelése
mov dptr,#08100h
mov r1, #01h

hurok02:
mov a,r1
movx a,@dptr
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok02

;;;CAN_CS_N tesztelése
mov dptr,#08800h
mov r1, #01h

hurok03:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok03

```

```
;;;DAC16_CS_N tesztelése
mov dptr,#08400h
mov r1, #01h

hurok04:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok04

;;;DAC16H tesztelése
mov dptr,#08200h
mov r1, #02h

hurok06:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok06

;;;DAC16L tesztelése
mov dptr,#08300h
mov r1, #02h

hurok07:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok07

;;;LDAC16N
mov dptr,#08500h
mov r1, #01h

hurok08:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok08

;;;DAC8_CS_N
mov dptr,#08600h
mov r1, #01h

hurok09:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok09

;;;LDAC_8N
```

```
mov dptr,#08700h
mov r1, #01h

hurok10:
mov a,r1
movx @dptr,a
mov a,dpl
xrl a,#0FFh;akkor ad nullát, ha megegyeznek.
inc dptr
jnz hurok10

;;;PWR_ON kimenet tesztelése:
;;;APB buszon
;PWR_ON GPIO bit értékéhez tartozó engedélyező bitet át kell áll. "1"-be
mov dptr,#0F900h ;Kiválasztom az Opmem_rdprog GPIO moduljának
; az első bitjének a regiszterét.
movx a,@dptr
orl a,#00000101b; Config regiszter beállítása a kimeneti regiszterek
movx @dptr,a

mov dptr,#0F9A0h
mov a,#0
movx @dptr,a
END
```

