

Óbudai Egyetem

Kandó Kálmán Villamosmérnöki Kar

Műszer-Automatika Intézet



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

ANDOR FPGA-ARM FEJLESZTŐPANEL

Szerzők:

Balogh Péter

Villamosmérnök BSc. szak, IV. évf.

Pete Ádám

Villamosmérnök BSc. szak, III. évf.

Konzulens:

Sándor Tamás

adjunktus

Prológus	3
1. A hardver	4
1.1. Az alkatrészek kiválasztása	4
1.1.1. FPGA kiválasztása	4
1.1.2. Mikrokontroller kiválasztása	4
1.1.3. 5V-os tápegység.....	4
1.1.4. 1,2V és 3,3V-os tápegységek.....	5
1.1.5. USB-s kommunikáció	6
1.1.6. SD kártya interfész	7
1.1.7. Digitális és analóg képkiment.....	7
1.1.8. CAN	7
1.1.9. RS-485	8
1.1.10. Ethernet	8
1.1.11. Elektrosztatikus kisülés (ESD) elleni védelem	9
1.1.12. Az AndOr blokkvázlata	10
1.2. Nyomtatott huzalozású lemez	11
1.2.1. A NYHL elrendezése (Layout)	11
1.2.2. A vezetékek elhelyezése	11
1.2.3. Kapcsolóüzemű tápegységek	13
1.3. Mikrosztríp és élcsatolt mikrosztríp	16
1.3.1. Mikrosztríp impedancia számítások	16
1.3.2. Élcsatolt mikrosztríp impedanciáinak meghatározása.....	19
1.4. A kapcsolóüzemű tápegység méretezése.....	22
1.4.1. Teljesítményfokozat.....	23
1.4.2. PWM komparátor	26
1.4.3. Hibajelerősítő és a kompenzáló tag.....	27
1.4.4. A kompenzáció meghatározása.....	28
1.5. Gyártás és tesztelés után	34
2. A Szoftver.....	35
2.1. Szoftver bevezető.....	35
2.2. Bootloader	35

2.3.	Mass Storage.....	36
2.4.	A bootloader felépítése.....	37
2.5.	USB réteg.....	37
2.5.1.	USB kommunikáció lépései.....	37
2.5.2.	USB réteg a kontrolleren.....	40
2.5.3.	Az USB leírók.....	40
2.5.4.	Eszköz leíró (Device Descriptor).....	40
2.5.5.	Konfiguráció leíró (Configuration Descriptor):.....	42
2.5.6.	Interfész leíró.....	43
2.5.7.	Végpont leíró.....	44
2.6.	Bulk Only Transfer állapotgép.....	45
2.6.1.	CBW.....	47
2.6.2.	CSW.....	48
2.7.	2. 5. A SCSI réteg.....	49
2.7.1.	Implementált SCSI parancsok.....	49
2.8.	FAT.....	50
2.8.1.	Pszedó FAT.....	51
2.8.2.	A boot szektor.....	51
2.8.3.	Root entry.....	52
2.8.4.	Bootloader összefoglaló.....	54
2.9.	Példa project bemutatása az AndOr fejlesztőpanelen.....	55
2.9.1.	A feladat leírása.....	55
2.9.2.	Megoldás a panelen.....	55
3.	Melléklet.....	59
3.1.1.	MatLab script impedanciaszámításhoz.....	59
3.1.2.	MatLab script a kompenzáció számításához.....	62
3.2.	Központi egységek lábkiosztása.....	67
3.2.1.	A processzor lábkiosztása [22].....	67
3.2.2.	Az FPGA lábkiosztása [23].....	68
3.3.	Az első verzió elkészült NYHL-e.....	70
4.	Felhasznált irodalom.....	72

Prológus

Az utóbbi években rengeteg fejlesztőeszközt készítettek mind FPGA-ra, mind mikrovezérlőre azon belül is ARM családú kontrollerekre, ezek ára a tömeggyártásnak és a technológiák egyre csökkenő költsége miatt bárki számára megvásárolhatóak lettek. A nagy gyártók is rájöttek arra, hogy a termékeiket nem feltétlenül csak az ipari szereplőknek kell szánniuk, hanem az átlag embernek, aki hobbi szinten foglalkozik az elektronikával. Azt gondolják és talán nem alaptalanul, hogy ha valakivel megismertetik a termékportfóliójukat, akkor ha mérnökként kell választaniuk a termékek között, akkor a már megismert eszközöket, családokat fogják preferálni. Közös ismertetőjegyük, az alacsony, párezer forintos ár, akár ingyenes szállítási lehetőséggel közvetlenül a cégtől. USB-s csatlakozású programozó hardver, szintén saját gyártású eszközökkel, ehhez társul még a saját függvény könyvtár amivel felgyorsul és leegyszerűsödik a fejlesztés (pl.: TivaWare). Akár teljes fejlesztői környezetet is mellékelhetnek, mint például az NXP cég LPCXpresso-ja. Közös ismertetőjegyük még a kevés periféria, a költséghatékonyság miatt, természetesen a vezérlő kivezetései szabványos 2,54mm-es csatlakozósoron elérhetőek.

A legismertebb példák erre a kezdeményezésre talán a Texas Instruments Stellaris és Tiva-C fantázianevű fejlesztőpaneljai, valamint az STMicroelectronics Nucleo paneljai. A legolcsóbb megoldás talán a Cypress Semiconductor CY8C4XXX panel családja amik 1000 Ft környékén kaphatóak. FPGA és azon belül a Xilinx portfóliójára építő fejlesztőpanelek már nem érhetőek el ilyen alacsony árszegmensben, elsősorban az FPGA integrált áramkörök magas ára miatt, részben azért, mert más ezeknek a fejlesztőpaneleknek a koncepciója. A panelek árát már kisebb arányban lehet csökkenteni azáltal, hogy a perifériákat le hagyjuk a panelről, természetesen erre is vannak példák, mint pl.: a ZTEX USB-FPGA paneljai, vagy egy Kickstarter kezdeményezésként indult Mojo fantázianevű eszköz. A legtöbb FPGA fejlesztőpanel pedig sok perifériát tartalmaz, köztük sok nagysebességű megoldást úgymint HDMI, Ethernet... Ebből is adódik az FPGA fejlesztőpanelek magasabb ára, az ARM alapúakhoz képest, mivel ezeknek a nagy sebességű jelvezetékeknek a NYHL-en való realizálása körülményesebb, mint a normál perifériáké.

Innen jött az ötlet, hogy kombinálhatnánk az ARM panelek olcsóságát, és széles körben elterjedtségét, és az FPGA panelek lehetőségeit egyetlen panelen. A perifériákat a két központi egység megosztottan tudja használni. Az FPGA működhet egyfajta multiplexerként, a perifériák és az ARM között.

1. A hardver

1.1. Az alkatrészek kiválasztása

1.1.1. FPGA kiválasztása

Az FPGA gyártók közül a Xilinx termékeit részesítettük előnyben, mivel a piacvezető gyártó eszközeinek az elterjedtsége jobb és nem utolsósorban szakirányunkon Xilinx eszközökön folyik az oktatás. A termékcsaládok közül a Spartan-6-os széria volt a legköltséghatékonyabb megoldás, ezen belül is az LX9-es modell TQFP-144-es tokozással. A BGA tokozáshoz finomabb rajzolatú és kis minimális furatátmérőt használó lemez szükséges, ami drasztikusan megnöveli az előállítási költséget, és tervezési időt.

1.1.2. Mikrokontroller kiválasztása

Az FPGA mellett szerettünk volna még egy olyan ARM 32 bites családjába tartozó mikrokontrollert is alkalmazni, ami kellően gyors, de kis lábszámú tokozásban is elérhető. A választás a Cortex M3 magot tartalmazó STMicroelectronics gyártmányú STM32F103C8T6 típust választottuk. A választásnál szerepet játszó szempontok az ár-érték arány, és rendelkeznie kellett minimum az egyetemen tanult perifériákkal. Ez a processzor nem túl drága, de mégis sokféle perifériával rendelkezik. A ma már alap dolognak számító SPI, I2C mellett megtalálható rajta CAN busz illesztési lehetőség, illetve natív USB támogatás is. A CPU maximálisan 72Mhz-es órajelen képes működni, ami általánosnak mondható az ARM-ok világában.

1.1.3. 5V-os tápegység

Első lépésként meghatároztam a maximum terhelőáramot az adatlapokban talált értékek alapján. Az üzemi áramfelvételek a legrosszabb esetre következők: TJA1040 70mA, SN75176 250mA, LM358 60mA, DDC 50mA. Ezek alapján egy minimálisan 500mA átfolyó árammal is működőképes tápegységet kerestem. Az egyszerűsége és kis mérete miatt lineáris üzemű tápegység választottam ezen a tápágon. Az alkatrész kiválasztásakor a következő szempontokat

vettem figyelembe:

- dropout értéke
- maximális kimeneti áram
- szükséges kimeneti kondenzátor értéke és fajtája

Ezeket a szempontokat és az árat, beszerezhetőséget figyelembe véve a választás a 78M05-ös IC-re esett. Ennél az alkatrésznél is kompromisszumot kellett kötnöm, a dropout értéke 2V, tehát 5V-os kimeneti feszültséghez minimum 7V-nyi bemeneti feszültség szükséges. A kimeneti kondenzátor igénye csupán egy 100nF-os kerámiakondenzátor. A modernebb kis dropout-tal rendelkező IC-k legtöbbször nagyobb kapacitású és egy megadott értéknél magasabb ekvivalens soros ellenállással (ESR) rendelkező kondenzátort igényelnek. Ekkor alumínium elektrolit kondenzátort kell alkalmazni, vagy egy kis ellenállású ellenállást kell sorba kötni a kimeneti kondenzátorral. [1]

1.1.4. 1,2V és 3,3V-os tápegységek

A széles tápfeszültség tartomány és az akár 1-2A-es terhelőáram miatt kapcsolóüzemű tápegységet választottam a lényegesen jobb hatásfok miatt. Ez a megoldás viszont drágább, tervezése tovább tart, a panelen elfoglalt területe nagyobb, és a nagyfrekvenciás zaj szempontjából rosszabb, mint a lineáris tápegység.

STMicroelectronics gyártmányú A5973D-t választottam, mivel ehhez a típushoz tudtunk könnyedén hozzájutni. A fontosabb tulajdonságai a következők:

- 4-36V-os bemeneti feszültségtartomány
- 2A-es maximális kimeneti áram
- 1,235V-os belső referencia, ami megfelelő az FPGA magfeszültségéhez
- belső kapcsolótranszisztor
- külső szinkronizálás lehetősége, így mindkét IC-t összekötve a tápegységek ellentétes fázisban fognak kapcsolni, ezáltal alacsonyabb lesz a kisugárzott teljesítmény, mint szinkronizálás nélkül
- túl áram, túlfeszültség, hőmérséklet megfutás elleni védelem

A szükséges flyback dióda típusa PMEG3030EP gyors Schottky-dióda, 30V-os maximális nyitó irányú feszültséget és 3A maximális nyitó irányú áramot visel el, és 3A áramnál 360mV

feszültség esik rajta. Kisméretű SOD128-as tokozást használok, az anódérintkezője azonos oldalon és 2 furattal az alatta lévő rétegen egy-egy nagykiterjedésű földkitöltésre csatlakozik a hő ellenállás csökkentéséért. A kimeneti induktivitás egy alacsony tekercs ellenállású, árnyékolt Mitsumi gyártmányú C8-K5L típus. A tápegység használatához szükséges kompenzáció méretezéséről az 1.4. fejezetben írok.

1.1.5. USB-s kommunikáció

PC és az FPGA közti kommunikáció egyik legegyszerűbb formája az USB, mivel minden gép rendelkezik ezzel a csatlakozással, és implementálása a panelra a könnyen elérhető USB UART interfészeknek köszönhetően viszonylag egyszerű. A piacon elérhető interfészek közül az FTDI cég által gyártott FT232R típusúra esett a választás, a szóba jöhető egyéb megoldások [2]:

- egyéb USB-UART interfész: MCP2200, XR21V1410, CP21xx, CY7C64225
- már a panelen található natív USB-t támogató STM32-es mikrokontroller használata
- FPGA + külső transceiver vagy FPGA + bridge IC (transceiver + Serial Interface Engine)

A legutolsó megoldást bonyolultsága miatt vettem el, a panelon már meglévő mikrokontrollert pedig nem kívántuk erre a feladatra is felhasználni. Az USB-UART interfészek közül azért az FTDI ezen típusára esett a választás, mert könnyen beszerezhető, rengetegen használják és jó a szoftveres támogatottsága. Ellene szól az ára és az, hogy hamisítják, ezért csak megbízható forrásból szabad beszerezni. 5 vezetékes bekötés van alkalmazva az FPGA felé, az RX, TX és hardveres Flow-Control vonalak vannak használva. Az interfész csak az USB 5V-járól van üzemeltetve, az összes I/O lába ki van vezetve szabványos 0,1 inch rászterű csatlakozókra, az FPGA bemenetei pedig tápfeszültség nélkül nagyimpedanciás állapotban vannak, ezek miatt az FT232R teljesen függetlenül használható a panel többi részétől. [3]

A mikrokontrollerben lévő USB periféria is igényel külső alkatrészeket. Mindkét vonalon (USBDP, USBDN) van 1-1 soros lezáró ellenállás, ez csökkenti a fel- és lefutási időket és a túllövést, csökkentve ezzel a kisugárzott zavart. A tagok és a föld között kis értékű kerámiakondenzátorok is helyet kaptak zavarvédetség növelése céljából.

1.1.6. SD kártya interfész

Az SD kártya elérése lehetséges SPI vagy SD módban, minden elérhető csatlakozás bekötésre került az FPGA-ba. Az összes vezetékre $47k\Omega$ felhúzó ellenállás került, hogy azok ne lebeghessenek, emiatt csak a külső mechanikus CD (Card Detect, kártyaérzékelés) kivezetés használható a kártya érzékelésére, ami szintén bekötésre került. [4]

1.1.7. Digitális és analóg képkiment

A panelen helyet kapott egy DVI-I csatlakozó is, mind az analóg mind a digitális része be van kötve az FPGA-ba. A digitális adat csak Single-Link kommunikációt támogat, tehát színinformációnként 1 differenciális pár, valamint ezek mellé még egy pár órajel tartozik, tehát összesen 4 pár vezeték van használva. A kimenet bufferelés nélkül jut a kijelzőbe, emiatt a kábel hossza korlátozott. Az analóg képinformáció 4 bites színmélységgel rendelkezik, tehát összesen $2^{(3*4)}=4096$ színárnyalatot képes megjeleníteni. Ezeken kívül a Display Data Channel (a továbbiakban DDC) ami egy szabványos I2C busz, is bekötésre került a mikrokontrollerbe. Melyen keresztül elérhető a panelra kötött kijelzőben lévő Extended display identification data (a továbbiakban EDID), mely információt tartalmaz többek között a maximális és natív felbontásról, támogatott időzítésekről stb. Mivel a kijelzőben lévő ROM 5V-os ezért a kontrollerről kivezetett és 3,3V-ra felhúzott I2C buszra csak szintillesztőt közbeiktatva lehet csatlakoztatni. A mikrokontroller adott lábai 5V bemeneti feszültséggel is kompatibilisek a busz pedig nyitott kollektoros, ezért nem lett volna szükség a szintillesztőre, csupán 5V-ra kötni a felhúzó ellenállásokat, de a buszt 3,3V-osként kívántuk máshol is felhasználni, ezért kivezetésre került egy tűskesoron. Az analóg színjeleket egy-egy R-2R létra típusú digitális analóg átalakító hozza létre, a kijelzőben lévő 75Ω -os lezárás segítségével jön létre a 0..0,7V-os feszültség érték amely a színinformációt hordozza.

1.1.8. CAN

CAN (Controller Area Network) kontrollerként a mikrokontrollerbe épített periféria van használva, a jelszintek illesztéséről egy TJA1040 típusú transceiver gondoskodik. Számunkra ez volt könnyen elérhető, de a legtöbb CAN transceiver láb kompatibilis egymással, így használható

a panelen pl. MCP2551, 2561 is. 120Ω -os véglezárás kapcsolható a differenciális párra egy DIP kapcsolóval, így több panel könnyen egy buszt alkothat. Az ESD védelemről egy PESD1CAN típusú szupresszor dióda gondoskodik, SOT-23-as tokozásának köszönhetően kis helyet foglal a NYHL-en és a differenciális pár mindkét tagját védi.

1.1.9. **RS-485**

A kontrollerben található egyik UART (Universal Asynchronous Receiver Transciever, Univerzális aszinkron adóvevő) perifériára van rákötve egy SN75176 szabványos RS-485 csatoló. A választás azért esett erre a típusa, mert kellően olcsó, széles körben használt és könnyen beszerezhető típus. 120Ω -os véglezárás szintén kapcsolható a vonalra. Az adó és vevő áramkörök külön engedélyezhetőek, de mindkettőt egyszerre engedélyezve a buszra kiküldött jelszint visszaolvasható, figyelve az ütközésekre. ESD védelemről szintén PESD1CAN dióda gondoskodik.

1.1.10. **Ethernet**

Az Ethernet alkalmazásának legköltséghatékonyabb megoldása egy PHY (physical layer) integrált áramkör alkalmazása. Ez összeköti a fizikai médiumot, az OSI modellben feljebb lévő MAC-el (media access control), ami az FPGA-ban vagy a mikrovezérlőben implementálható. A kiválasztott IC a Micrel gyártmányú KSZ8081RNB, mely RMII-en (Reduced Media Independent Interface) keresztül kommunikál a MAC-el. A szükséges 50MHz-es órajelet az FPGA biztosítja, így nincs szükség külső órajelre.

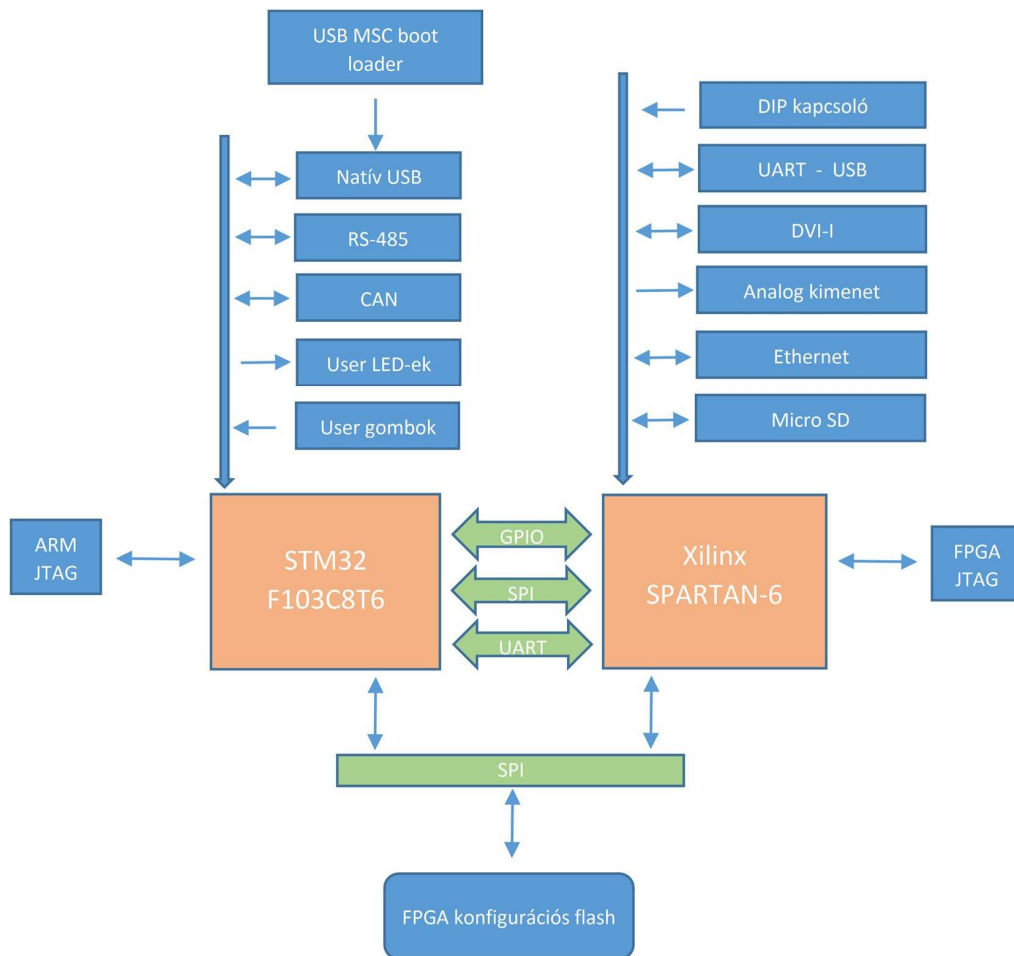
1.1.11. Elektrosztatikus kisülés (ESD) elleni védelem

A panelről kivezetett jelvezetékek megfelelő ESD (elektrosztatikus kisülés) védelemmel láttam el. A kivezetésekre párhuzamosan tranziens feszültségvédő szupresszor (TVS) diódákat helyeztem. Ezek többségében különlegesen kialakított Zener-diódák, alacsony parazita kapacitással, valamint nagyobb tranziensáramokat viselnek el károsodás nélkül. A cél olcsó, kisméretű, alacsony parazita kapacitású dióda választása volt. A kis méret az elhelyezést könnyíti és a kis méret velejárója az alacsony parazita induktivitás és kapacitás. Az alacsony parazita kapacitás azért kívánatos hogy a nagysebességű adatátvitelt minél kisebb mértékben befolyásolja. A kis parazita induktivitás az esetlegesen bekövetkező ESD hatásának csökkentése miatt fontos. A cél az, hogy a nagy felfutási meredekségű, nagyfeszültségű tüskék a védőáramkörökön keresztül hagyják el a panelt, kikerülve az erre érzékeny digitális áramköröket. A nagyfrekvenciás komponenseket tartalmazó kisülés a legkisebb impedancia felé "tart", természetesen azt szeretnénk, hogy ez az védőáramkör legyen, emiatt fontos, hogy a diódák a csatlakozókhoz minél közelebb kerüljenek elhelyezésre. Utóbbi azért is lényeges, hogy az panelen található közeli, nem védett vezetékszakaszokra ne csatolódjon kapacitív úton a kisülés.

A kiválasztott típusok: PRTR5V0U2X, PESD3V3S2UQ, CG0402MLA-5.5, PESD1CAN

A legelsőt használok az USB bemeneteken, mivel védi a tápfeszültség vonalat is. A második alkatrész két csatornás unipoláris, a harmadik 1 csatornás bipoláris szupresszor diódával együtt a többi kivezetés közt van elosztva. PESD1CAN kettős diódák vannak elhelyezve a CAN és RS485 vonalakra.

1.1.12. Az AndOr blokkvázlata



1.2. Nyomtatott huzalozású lemez

A panel komplexitása miatt 4 rétegű nyomtatott huzalozású lemezt (a továbbiakban NYHL) alkalmaztam. A rétegfelépítés (layer stackup) a következőképpen alakul felülről lefelé haladva:

- Jelréteg (főleg nagy sebességű jelek)
- Földréteg
- Tápréteg
- Jelréteg (főleg alacsonyabb sebességű jelek)

A panel méreteit 10x10cm-ben határoztam meg a gyártó árai és a szükséges terület alapján.

1.2.1. A NYHL elrendezése (Layout)

A NYHL tervezés egyik legtöbb legösszetettebb pontja az alkatrészek és a vezetősávok megfelelő elhelyezése. A panelünkön lévő nagyobb odafigyelést igénylő részegységek a következők: kapcsolóüzemű tápegységek, DVI, USB, Ethernet differenciális párok, FPGA konfigurálásához és FPGA és a mikrokontroller közötti kommunikációhoz használt SPI interfész. Az alkatrészek és vezetékek elhelyezésénél az alábbi ökölszabályokra kell figyelni:

Az áramkörön folyó áram a legkisebb impedancia felé folyik vissza a forrásba. Egyenáramnál ez a legkisebb ellenállást, tehát a legrövidebb utat jelenti, míg váltakozó áramnál a legkisebb induktivitást. Ez azt jelenti a gyakorlatban, hogy az áram az oda- és visszavezető ága között a legkisebb távolságra törekszik. Az elektromágneses kisugárzás tehát akkor a legkisebb, ha az áram a legkisebb körben folyik. Emiatt az egyik belső rétegre egy teljes földkitöltést alkalmazok.

[5][6][7]

1.2.2. A vezetékek elhelyezése

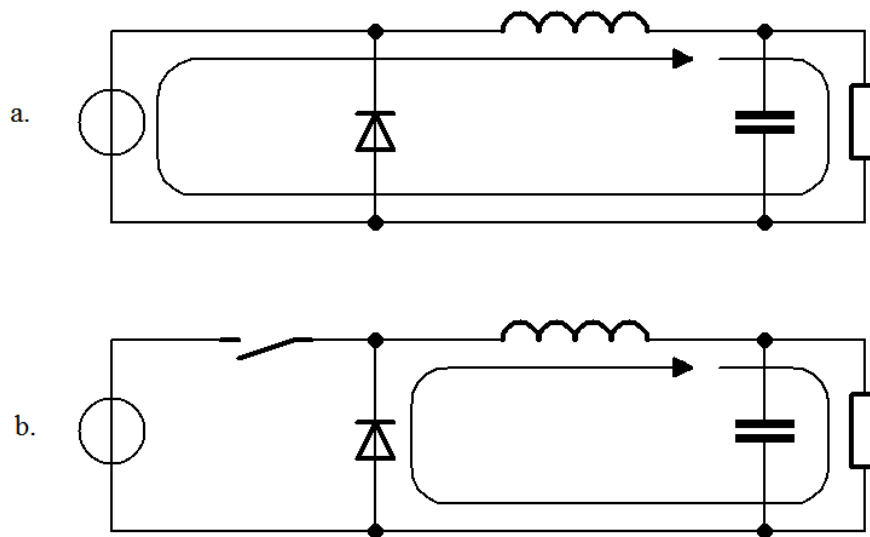
- A vezetékek irányváltások lehetőleg minél kisebb szögben történjenek, magasabb frekvenciás jeleknél görbe vonalak használata célszerű. Ez több dolog miatt is fontos, egyrészt a túl nagy szögben hajlított vezeték külső éle antennaként viselkedik és a szomszédos vezetékekre csatolódhat a rajta vezetett nagyfrekvenciás jel. Másrészt a belső

élről nehezebb eltávolítani a maratószert és ez alámaráshoz vezethet. Valamit a hirtelen átmérőváltozás impedancia ugrást eredményez, ami reflexiót és ezzel együtt a jelintegritás csökkenést okoz.

- A nagysebességű vezetékek egymáshoz képest a lehető legnagyobb távolságra helyezése
- A nagysebességű vezetékek alatt lévő föld és tápkitöltés folytonos legyen
- A differenciális párok hosszkülönbsége a lehető legkisebb legyen
- A differenciális párok mindkét tagja közti távolság lehetőleg változatlan maradjon a a vezeték hossza mentén. Mindkét tag azonos úton legyen vezetve, hogy a külső hatások mindkét vezetékre ugyanúgy hassanak.
- Alapelv, hogy az analóg és digitális részeket a lehető legtávolabb helyezzük egymástól, a panelen az egymásra hatás csökkentése miatt.

1.2.3. Kapcsolóüzemű tápegységek

A 1-0-1. ábrán látható módon folynak áramok a feszültségcsökkentő konverterben, a két áramkör közül a bemeneti körben a tranzisztor nyitott állapotánál (a.) folyó áram nagyobb frekvenciás komponenseket tartalmaz, di/dt értéke magasabb, mint a kimeneti körnél (b.). A bemeneti kör tervezésére kell több figyelmet fordítani, a képen lévő feszültséggenerátort a bemeneti kondenzátorok alkotják.

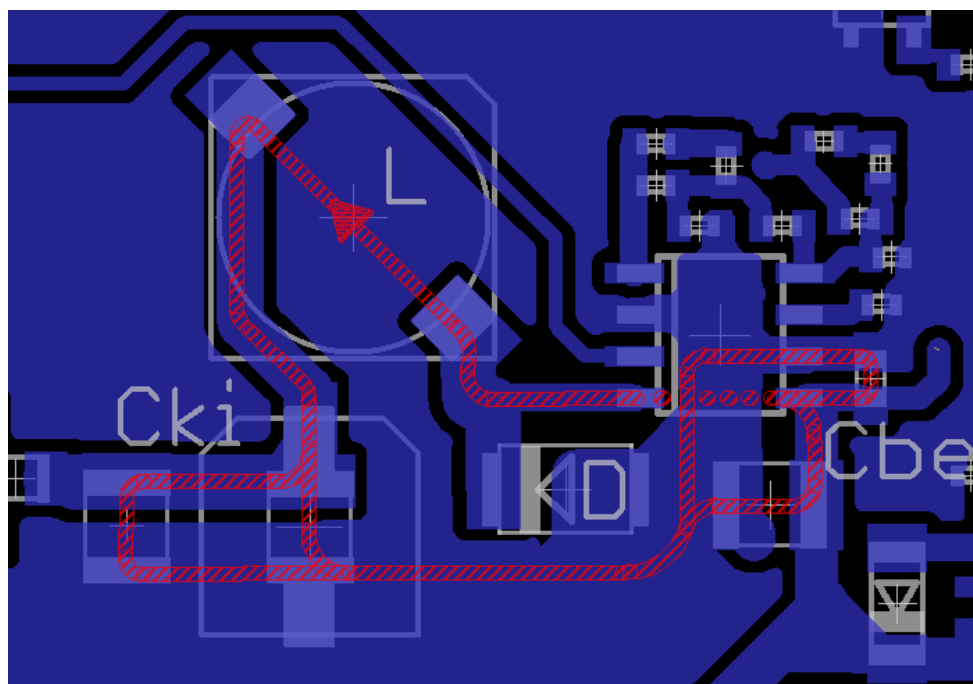


1-1-1. ábra Áramkörök a buck konverterben

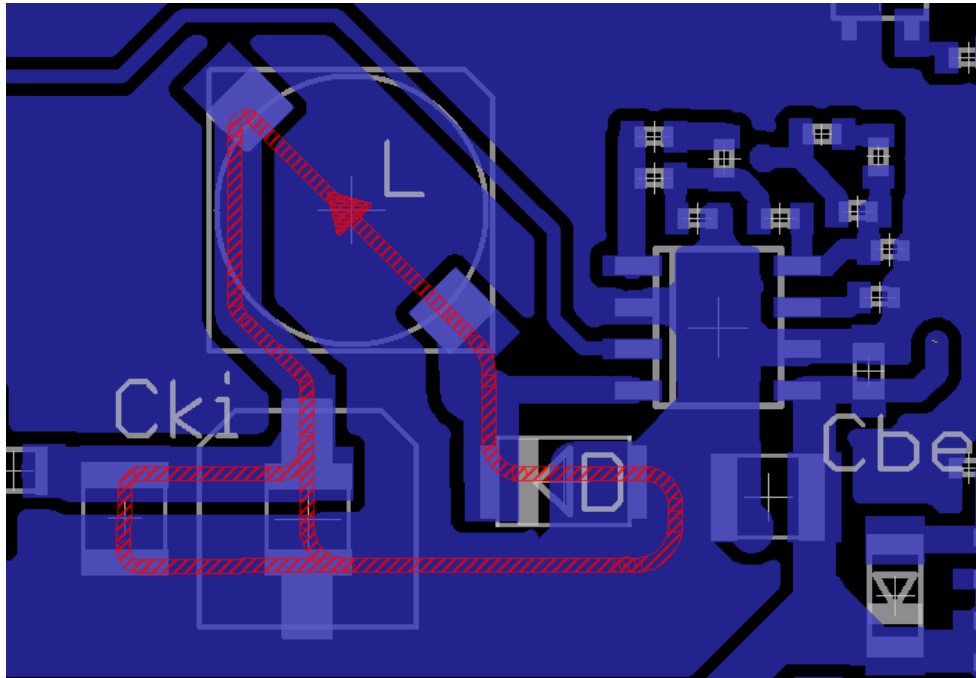
A bemenő hurkot a legcélravezetőbben úgy csökkenthetjük, hogy a bemenő és kimenő kondenzátor valamint a dióda azonos potenciálú pontját (ez a tápegység földpontja, referenciapontja) a lehető legrövidebb úton kötjük össze. Az integrált áramkör táp bemenete és a föld közé egy $1\mu\text{F}$ -os felületszerelt kerámiakondenzátort helyeztem a csatlakozás közvetlen közelébe, azonos oldalra. A bemenő áram nagyfrekvenciájú áramtüskeként jelentkezik, így ezt az odahelyezett kondenzátor tudja biztosítani.

A másik nagy di/dt értékkel rendelkező rész az a kapcsolópont, ami a kapcsolóelem, tekercs és a dióda közös pontja. A pont kétféleképpen növeli az elektromágneses interferenciát, egyrészt a

gyorsan változó feszültségtűskék kapacitív úton más vezetékszegmensekre csatolódnak, másrészt a váltakozó áram miatt maga a pont is sugároz. A kapcsolópont fizikai kiterjedésének csökkentésével lehet elérni az interferencia csökkenését.



1-1-2. ábra Bemenő kör



1-1-3. ábra Kimenő kör

Mindkét képen piros vonal jelzi az áram (ideális) folyási útját.

Szűrőkondenzátorok

- A szűrendő tápbemenethez minél közelebb kell elhelyezni a kondenzátorokat.
- A föld és a szűrendő tápfeszültség ágakhoz is minél rövidebb vezetősávon kell kapcsolódnia.
- Minden áramkörü elem legyen megfelelően szűrve.
- Ha a kondenzátort a túloldalra kell ültetni, akkor a hozzá tartozó átvezetések a kondenzátorhoz minél rövidebb vezetősávon csatlakozzanak és a lehető legtöbb átvezetést kell alkalmazni a soros induktivitás csökkentése miatt.
- Többféle kapacitású kondenzátor párhuzamos kapcsolása nagyobb frekvenciasávot fed le ahol a szűrés a leghatékonyabb.
- Minél kisebb ESL értékű kondenzátort kell választani magasabb frekvenciákra, ezt felületszerelt kerámiakondenzátorokkal lehet elérni

1.3. Mikrosztrip és élcsatolt mikrosztrip

A mikrosztrip (microstrip) más néven mikroszalagvonal olyan tápvonal ahol vékony vezetősáv van elhelyezve dielektrikumra ami alatt vékony, összefüggő rétegű fémzés található. Egyszerűen méretezhető és előállítható NYHL-en, nagyfrekvenciás jelek átvitelére szolgál. Másik NYHL-en egyszerűen létrehozható struktúra a stripline, ahol a vezetősávot mindkét oldalról dielektrikum és annak külső felületén nagy kiterjedésű fémzés határol. Utóbbi 1 réteggel többet igényel a mikrosztripnél, viszont támogatja a TEM-módusú (Transzverzális Elektromágneses) hullámterjedést ami közel veszteségmentes.[11] Míg a mikrosztrip, csak a kvázi-TEM módust, ami azt jelenti, hogy az elektromágneses hullámoknak van a longitudinális összetevője is. Azokon a frekvenciákon amik a panelon lévő interfészekon előfordulhatnak, elégséges a mikrosztrip felépítés használata.

1.3.1. Mikrosztrip impedancia számítások

A tervezési segédletben[8] és a hibajegyzékében[9] leírt számításoknál egy pontosabb ánde bonyolultabb megoldást választottam. E. Hammerstad és Ø. Jensen [10] által először alkalmazott számítási módszert használtam a hullámimpedancia, effektív dielektromos állandó és effektív vezetősáv szélesség számításához. A számításokat MatLab program segítségével oldottam meg.

Ehhez a alábbi bemenő paraméterekre van szükség:

t = a jelvezeték vastagsága, W = a jelvezeték szélessége h = a dielektrikum vastagága

ϵ_r = a dielektromos állandó, Z_0 = a szabad tér hullámellenállása, értéke megközelítőleg 377Ω [11]

A jelvezeték modellezhető egy 0 vastagságú de szélességében megnövelt jelvezetékkel, hogy a számítások leegyszerűsödjenek, az effektív jelvezeték szélesség kiszámítása:

A korrekció értéke homogén közegre:

$$\Delta W_1 = \frac{t}{h \cdot \pi} \cdot \ln \left(1 + \frac{4e}{\frac{t}{h} \cdot \coth^2 \sqrt{6,517W}} \right)$$

A korrekció értéke inhomogén közegre, mint amilyen a mikrosztrip:

$$\Delta W_r = \frac{1}{2} \Delta W_1 \cdot (1 + \operatorname{sech} \sqrt{\varepsilon_r - 1})$$

Az effektív jelvezeték szélesség inhomogén közegre: $W_r = W + \Delta W_r$

A vezeték elektromágneses terének terjedése részben a dielektrikum anyagában részben a levegőben történik, ezt a számításoknál figyelembe kell venni.

Effektív dielektromos állandó kiszámítása:

$$\varepsilon_{\text{reff}}(W, h, \varepsilon_r) = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2} \cdot \left(1 + 10 \frac{h}{W}\right)^{-ab}$$

ahol:

$$a(u) = 1 + \frac{1}{49} \cdot \ln \left(\frac{u^4 + (u/52)^2}{u^4 + 0,432} \right) + \frac{1}{18,7} \cdot \ln \left(1 + \left(\frac{u}{18,1} \right)^3 \right)$$

$$b(\varepsilon_r) = 0,564 \cdot \left(\frac{\varepsilon_r - 0,5}{\varepsilon_r + 3} \right)^{0,053}$$

$$u = \frac{W}{h}$$

A hullámimpedancia kiszámításánál az effektív jelvezeték szélességgel és az effektív dielektromos állandóval számolunk.

$$Z_{L1}(W, h) = \frac{Z_0}{2\pi} \cdot \ln \left(f_u \frac{h}{W} + \sqrt{1 + \left(\frac{2h}{W} \right)^2} \right)$$

$$Z_L(W, h, \varepsilon_r) = \frac{Z_{L1}(W, h)}{\sqrt{\varepsilon_r}} = \frac{Z_0}{2\pi \cdot \sqrt{\varepsilon_r}} \cdot \ln \left(f_u \frac{h}{W} + \sqrt{1 + \left(\frac{2h}{W} \right)^2} \right)$$

ahol:

$$f_u = 6 + (2\pi - 6) \cdot \exp\left(-\left(30,666 \cdot \frac{h}{W}\right)\right)^{0,7528}$$

Az effektív értékek behelyettesítése:

$$Z_L(W, h, t, \varepsilon_r) = \frac{Z_{L1}(Wr, h)}{\sqrt{\varepsilon_{\text{reff}}(Wr, h, \varepsilon_r)}}$$

Ezekkel a számításokkal a vezeték fizikai méreteiből számítható ki a hullámellenállás, számunkra viszont a hullámellenállás, dielektrikum magassága és relatív permittivitása, valamint a fémzés vastagsága adott és keressük a jelvezeték szélességét. Ehhez át kellene rendezni mindhárom számítást, de egyszerűbbnek láttam ha a Matlab program segítségével és próbálgatással számoltatom ki a kívánt szélességet. A bemenő paraméterek mértékegysége nem fontos, csak arra kell ügyelni, hogy mindhárom ugyanolyan dimenziójú legyen.

1.3.2. Élcsatolt mikrosztrip impedanciáinak meghatározása

A differenciális jeltovábbítás azon alapszik, hogy két abszolút értékében egyenlő de polaritásában ellenkező jelet bocsájt ki egy vezetékpárra. Ez amiatt előnyös, mert a közös módusú zajokat, tehát amik a pár mindkét tagjára ugyanolyan előjellel és amplitúdóval hatnak, majdnem teljesen figyelmen kívül hagyja. Ezt páratlan módusú jelterjedésnek is nevezik (odd mode propagation). A következő számítások az IPC-2141A tervezési segédletből[] és a hozzá tartozó hibajegyzékből[] származnak. Ezek alapján kiszámítható a páratlan módusú impedancia és ennek a duplája a differenciális impedancia.

ahol Z_{odd} a páratlan módusú impedancia és Z_{diff} a differenciális impedancia, valamint $Z_{diff}=2*Z_{odd}$
Valamint a fejezetben megismert értékeket használom.

$$Z_{odd} = \frac{Z_L \sqrt{\frac{\epsilon_{reff}}{\epsilon_{reff,0}}}}{1 - \frac{Z_L}{Z_0} \sqrt{\epsilon_{reff}} \cdot Q_{10}}$$

ahol:

$$c_0 = [b_0(\epsilon_r) - 0,207] \cdot e^{-0,414u}$$

$$b_0(\epsilon_r) = \frac{0,747\epsilon_r}{0,15 + \epsilon_r}$$

$$d_0 = 0,593 + 0,694 \cdot e^{-0,562u}$$

$$a_0(u, \epsilon_r) = 0,7287[\epsilon_{reff} - 0,5(\epsilon_r + 1)] \cdot (1 - e^{-0,179u})$$

$$\varepsilon_{reff,o} = \left[0,5(\varepsilon_r + 1) + a_0(u, \varepsilon_r) - \varepsilon_{reff} \right] \cdot e^{-c_0 g^{d_0}} + \varepsilon_{reff}$$

$$Q_{10} = \frac{1}{Q_2} (Q_2 Q_4 - Q_5 \cdot e^{\left[\ln(u) Q_6 u^{-Q_9} \right]})$$

ahol:

$$Q_1=0,8695\cdot u^{0,194}$$

$$Q_2=1+0,7519g+0,189g^{2,31}$$

$$Q_3=0,1975+\left[16,6+\left(\frac{8,4}{g}\right)^6\right]^{-0,387}+\frac{1}{241}\cdot\ln\left[\frac{g^{10}}{1+\left(\frac{g}{3,4}\right)^{10}}\right]$$

$$Q_4=\frac{2Q_1}{Q_2\left[e^{-q}u^{Q_3}+(2-e^{-g})\cdot u^{-Q_3}\right]}$$

$$Q_5=1,794+1,14\ln\left(1+\frac{0,638}{g+0,517g^{2,43}}\right)$$

$$Q_6=0,2305+\frac{1}{281,3}\ln\left[\frac{g^{10}}{1+\left(\frac{g}{5,8}\right)^{10}}\right]+\frac{1}{5,1}\ln\big(1+0,598g^{1,154}\big)$$

$$Q_7=\frac{10+190g^2}{1+82,3g^3}$$

$$Q_8 = e^{\left[-6,5 - 0,95 \ln(g) - \left(\frac{g}{0,15} \right)^5 \right]}$$

$$Q_9 = \ln(Q_7) \left(Q_8 + \frac{1}{16,5} \right)$$

$$g = \frac{s}{h}$$

A választott gyártó a következő paramétereket adta meg a lemezre vonatkozóan:

A dielektrikum vastagsága az 1. és 2. valamint a 3. és 4. réteg között: $h=0,2\text{mm}$

A dielektromos állandó: $\varepsilon_r=4,58$

A fémezés vastagsága: $t=35\mu\text{m}$

Az aszimmetrikus (single ended) vezetékhez (pl.: SPI, órajelek) szükséges 50Ω impedanciájú jelvezeték szélessége, a fenti értékeket figyelembe véve az alábbi érték jött ki:

$$W=0,365\text{mm}$$

A DVI valamint az Ethernet párokat 100Ω -os differenciális impedanciára kell tervezni, tehát a páratlan módusú impedanciája 50Ω , ekkor:

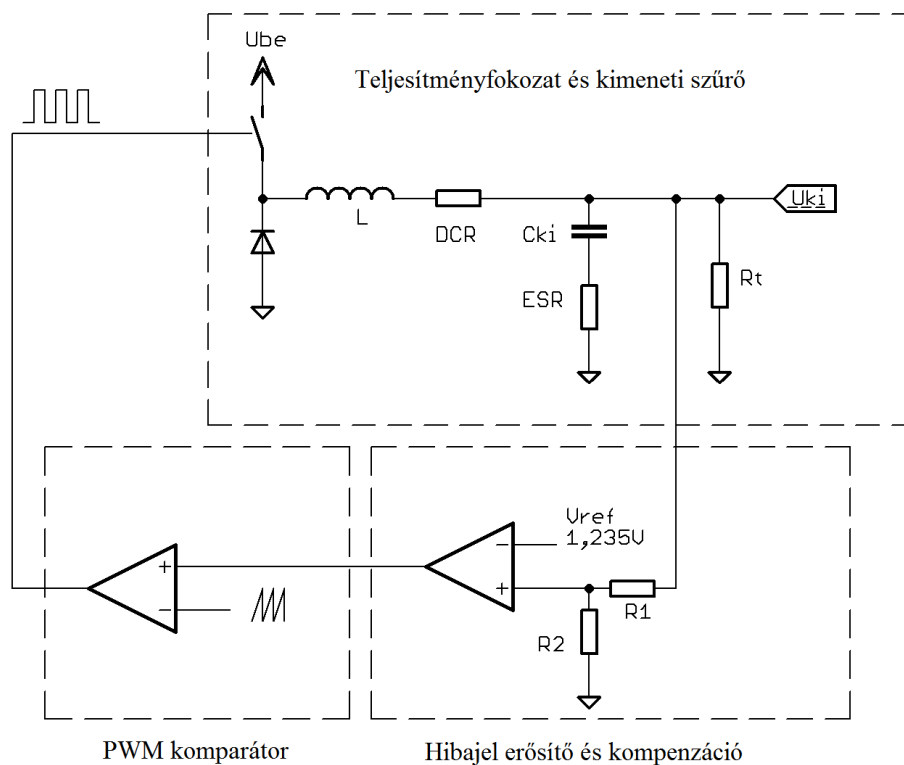
$$W=0,381\text{mm} \quad s=0,4318\text{mm}$$

Az USB pár 90Ω -os differenciális impedanciát igényel 45Ω -os páratlan módusú impedancia mellett:

$$W=0,376\text{mm} \quad s=0,254\text{mm}$$

1.4. A kapcsolóüzemű tápegység méretezése

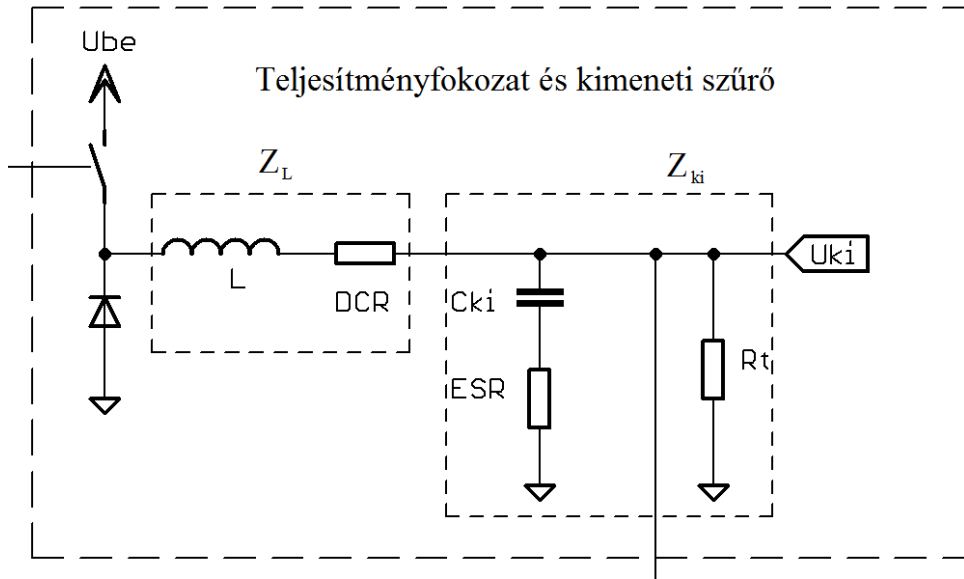
Első lépésként ki kell választanunk a kimeneti szűrőt alkotó kondenzátor(ok) és induktivitás paramétereit. Tápegység egyik fontos követelménye a minél alacsonyabb kimeneti feszültség hullámlás (Output Voltage Ripple). Ezt a legegyszerűbben a kimeneti szűrőkondenzátor(ok) helyes megválasztásával lehet elérni. Cél a minél alacsonyabb ekvivalens soros ellenállású és induktivitású kondenzátorok választása. Az ekvivalens soros induktivitást (ESL) elhanyagoltam, így egyszerűsödtek a számítások. Az ESL értéke főleg a kondenzátor tokozásától függ, a méret csökkenésével arányosan csökken. Az ekvivalens soros ellenállás (ESR) értéke a kondenzátor vezetékvezetésének és kivezetésének köszönhető. Ez utóbbi nem kívánatos tulajdonság okozza azt a problémát amiről majd a kompenzációt taglaló fejezetben fogok írni. Ez az egyik legfontosabb jellemző ami számunkra szükséges a kompenzáció módjának helyes meghatározásához és a szükséges elemek értékeinek kiszámításához. A kimeneti szűrő LC tagja 180° -os fáziskésést ad a szabályozási hurokhoz, ezért a stabil működéshez elengedhetetlen kompenzáció alkalmazása, hogy ezt a fáziskésést csökkentsük. Az áramkörön belül nem található kompenzáció, ezért ezt nekünk kell méretezni, így viszont nagyobb mozgásterünk van az alkalmazható kimeneti szűrőkondenzátor kiválasztásánál. A fejezethez felhasznált irodalom: [12] [13] [14][15]



1-1-4. ábra A szabályozási kör blokkdiagramja

1.4.1. Teljesítményfokozat

Több kimeneti kondenzátort alkalmazása mellett döntöttem, mivel az eredő ESR így még kisebb, valamint ötvözni tudom a különféle típusú kondenzátorok előnyeit. Egy polimer elektrolit a nagy fajlagos kapacitása miatt, vele párhuzamosan pedig egy X7R karakterisztikájú kerámiakondenzátor alacsony ESR és jobb nagyfrekvenciás viselkedése miatt alkotja a kondenzátor blokkot. A polimer elektrolit típusa: Nippon NP-CAP-PX-H70 , kapacitása 220 μ F, ESR értéke 35m Ω a kerámiakondenzátor kapacitása 22 μ F. A szűrőtekercs típusa Mitsumi C8-K5L, induktivitása 22 μ H, DCR ellenállása 53m Ω .



1-1-5. ábra A teljesítményfokozat és kimeneti szűrő

A teljesítményfokozat átmeneti függvénye:

$$G_P(s) = \frac{U_{ki}}{U_{be}} = \frac{Z_{ki}}{Z_L + Z_{ki}}$$

ahol Z_L az induktivitás, Z_{ki} pedig a teljesítményfokozat kimenő impedanciája

$$Z_{ki} = \frac{R_t \left(R_{ESR} + \frac{1}{sC_{ki}} \right)}{R_t + \left(R_{ESR} + \frac{1}{sC_{ki}} \right)} = \frac{sC_{ki} \cdot R_t \cdot R_{ESR} + 1}{sC_{ki} \cdot (R_t + R_{ESR}) + 1}$$

$$Z_L = sL + R_{DCR}$$

ahol R_{ESR} a kimeneti kondenzátor ekvivalens soros ellenállása, R_{DCR} a tekercselés egyenfeszültségű ellenállása, a kapcsolóelem csatorna ellenállása és a nyomtatott huzalozás ellenállása, ez utóbbit nem vettem figyelembe a számolás során

Visszahelyettesítve az előbbi képletbe:

$$G_p(s) = \frac{U_{ki}}{U_{be}} = \frac{\frac{sC_{ki} \cdot R_t \cdot R_{ESR} + 1}{sC_{ki}(R_t + R_{ESR}) + 1}}{sl + R_{DCR} + \frac{sC_{ki} \cdot R_t \cdot R_{ESR} + 1}{sC_{ki}(R_t + R_{ESR}) + 1}}$$

egyszerűsítés után:

$$G_p(s) = \frac{R_t \cdot (sC_{ki} \cdot R_{ESR} + 1)}{s^2 LC_{ki} \cdot (R_t + R_{ESR}) + s \{ L + C_{ki} \cdot [R_{DCR} (R_t + R_{ESR}) + R_t \cdot R_{ESR}] \} + (R_t + R_{DCR})} =$$

$$\frac{(C_{ki} + R_{ESR})s + 1}{\left(\frac{R_t + R_{ESR}}{R_t} \right) \cdot L \cdot C_{ki} s^2 + \left(\frac{L}{R_t} + R_{ESR} \cdot C_{ki} + R_{DCR} \cdot C_{ki} + \frac{R_{DCR} \cdot R_{ESR} \cdot C_{ki}}{R_t} \right) s + \frac{R_{DCR}}{R_t} + 1}$$

az átmeneti függvény zérusa:

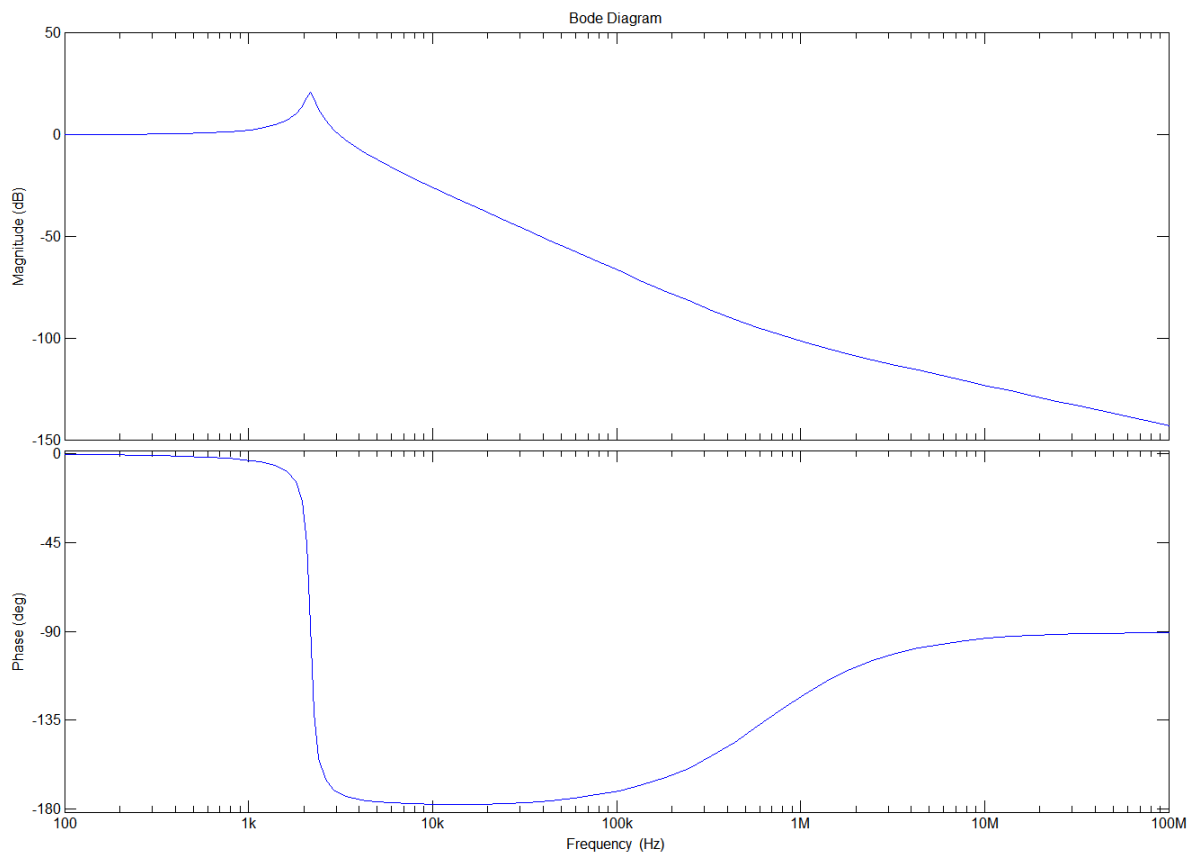
$$f_{z0} = \frac{1}{2\pi \cdot C_{ki} \cdot R_{ESR}}$$

a LC szűrő komplex póluspárja:

$$f_{p0} = \frac{1}{2\pi \sqrt{L \cdot C_{ki} \left[\frac{1 + (R_{ESR} / R_t)}{1 + (R_{DCR} / R_t)} \right]}}} \approx \frac{1}{2\pi \sqrt{LC_{ki}}}$$

A következő ábrán látható egy ideális ($R_{ESR}=R_{DCR}=0$) LC szűrő Bode-diagramja. Az amplitúdó karakterisztikán jól megfigyelhető, hogy a póluspárnál nagyobb frekvencián -40dB/dekád, a

zérus vizsont kikompenzálja az egyik pólust. A fáziskarakterisztika a póluspár miatt meredeken -180° felé tart, a zérus miatt vizsont -90°-ra emelkedik.



1-0-6. ábra LC szűrő Bode-diagramja

1.4.2. PWM komparátor

A PWM (Pulse Width Modulator, impulzus szélesség modulátor) komparátor erősítési tényezője a következőképpen alakul:

$$G_M = \frac{V_{CC}}{V_{OSCMAX} - V_{OSCMIN}}$$

Ahol V_{CC} a bemenő tápfeszültség, a V_{OSCMAX} - V_{OSCMIN} pedig a modulátor fűrészelének csúcstól csúcsig mérhető feszültsége

Voltage Feed-Forward alkalmazásával a G_M erősítési tényező változatlan, ez úgy kivitelezhető, hogy a PWM-hez szükséges fűrészel meredeksége a tápfeszültséggel arányosan változik:

$$V_{OSC\text{MAX}} - V_{OSC\text{MIN}} = K \cdot V_{CC}$$

Ahol a K arányossági tényező adatlapban megadott értéke 0,076, a PWM erősítési tényezője tehát:

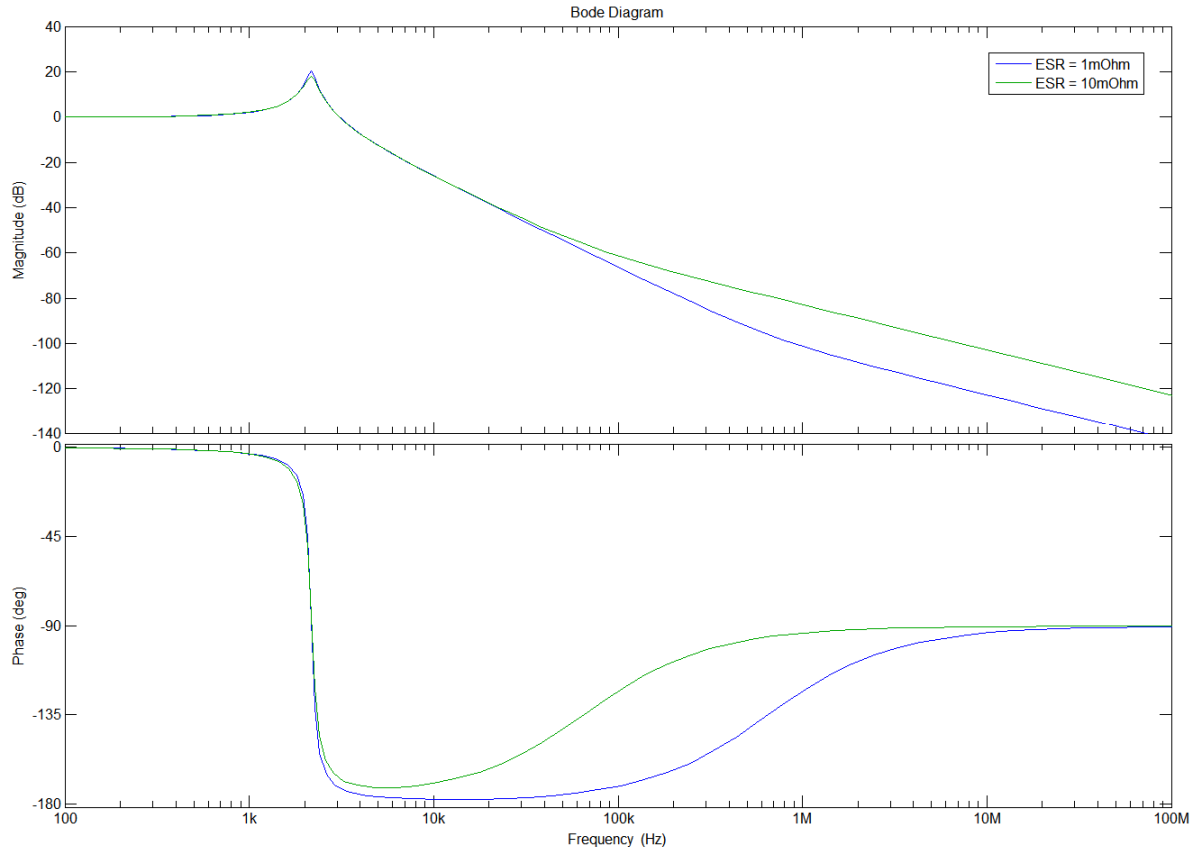
$$G_M(s) = \frac{1}{K} = \text{állandó}$$

Erősítése tehát minden frekvencián állandó és fázistolása nincs.

A Voltage Feed-Forward alkalmazásával a kapcsolóelem kitöltési tényezője a bemenő feszültség változásának hatására reagál, az egész szabályozási kört kikerülve. Ezáltal sokkal jobb feszültségstabilizálási szám (line regulation) érhető el.

1.4.3. Hibajelerősítő és a kompenzáló tag

A teljesítményfokozat átviteli függvényének zérusa, amit a kondenzátor kapacitása és az soros ellenállása határoz meg, kikompenzálja a póluspár egyik tagját így növelve a fázist -90° -ra, ez segít a kívánt fázistartalék beállításánál.



1-1-6. ábra ESR változásának hatása a frekvenciamenetre

A problémát az okozza, hogy ha az ESR értéke túl kicsi, akkor a zérus frekvenciája a kívánt vágási frekvenciához képest lényegesen nagyobb lesz, így a kompenzáló hatása nem érvényesül. A következő meghatározandó érték a vágási körfrekvencia (zero-crossover frequency), ennek a növekedése arányosan javítja a tranziensekre adott választ, a maximális értékét a Nyquist-kritérium szabja meg. Általánosan $1/10$ és $1/5 f_{OSC}$ közötti értéknek választják.

1.4.4. A kompenzáció meghatározása

Első lépésként meghatározzuk a vágási körfrekvenciát, ezt 25kHz-nek választottam. A kondenzátorok együttes ESR értékét $1m\Omega$ -nak számoltam. A jobb közelítés miatt a tekercs DCR ellenállásához hozzáadtam a kapcsoló MOSFET csatorna-ellenállását, ami $250m\Omega$, a rézvezetékezés ellenállását figyelmen kívül hagytam, mivel ezekhez képest elhanyagolható.

$$f_0 = 25kHz$$

$$C_{ki} = 220\mu F + 22\mu F = 242\mu F$$

$$L_{ki} = 22\mu H$$

$$R_{DCR} = 250m\Omega + 53m\Omega = 303m\Omega$$

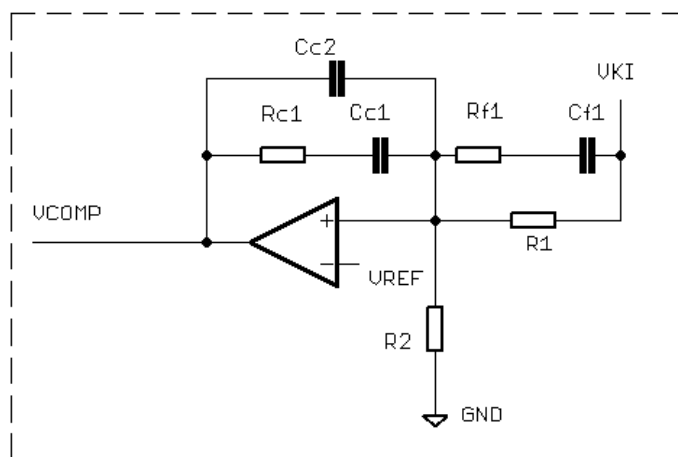
A pólus, zérus és átviteli függvény számítások a x.x. alapján.

$$f_{p0} = 2.181kHz$$

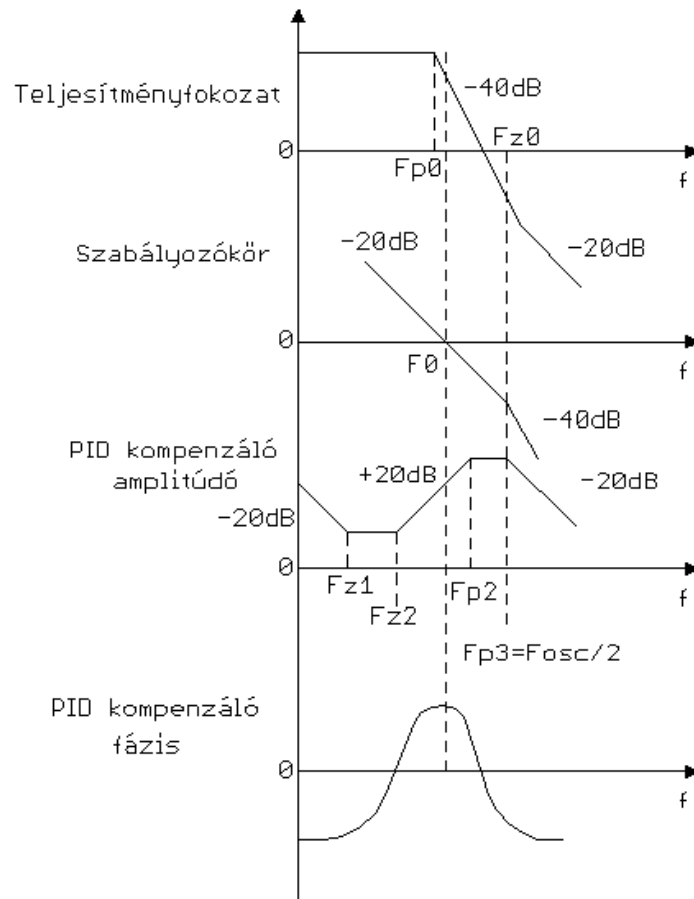
$$f_{z0} = 657.665kHz$$

Megfigyelhető, hogy a kimeneti szűrő zérusfrekvenciája több mint egy nagyságrenddel nagyobb, mint a választott vágási körfrekvencia. A frekvenciák egymáshoz képesti elhelyezkedése alapján Type III B (PID) típusú kompenzációt választottam. Ez a megoldás használatos kerámiakondenzátorok kikompenzálására.

$$f_{p0} < f_0 < f_{OSC/2} < f_{z0}$$



1-1-7. ábra PID kompenzáció felépítése



1-1-8. ábra A kompenzáció tipikus Bode-diagramjai

A PID kompenzáció 2 alaptípusa a zérusok és a pólusok egymáshoz képesti elhelyezkedésében eltér egymástól. A B típusú kompenzálótag pólusait és zérusait a következő frekvenciákra érdemes felvenni:

- zérus a teljesítményfokozat póluspárjának közelében
- pólus a Nyquist frekvencián, a nagyobb frekvenciás jelek csillapítására
- egy pólus és egy zérus a vágási frekvencia közelében, fázissiettető (lead) tagot alkotnak
- ezen kívül egy pólus található 0 Hz-en.

$$f_{z1} \approx f_{p0} \Rightarrow f_{z1} = f_{p0} / 2 = 1,091 \text{ kHz}$$

$$f_{p3} = f_{osc} / 2 = 125 \text{ kHz}$$

A kívánt fázisemelkedés $= \Theta = 70^\circ$, ekkor:

$$f_{P2} = f_0 \cdot \sqrt{\frac{1 + \sin \Theta}{1 - \sin \Theta}} = 141,782 \text{ kHz}$$

$$f_{Z2} = f_0 \cdot \sqrt{\frac{1 - \sin \Theta}{1 + \sin \Theta}} = 4,408 \text{ kHz}$$

$$f_{P1} = 0$$

Az alkatrészek meghatározása:

Legelső meghatározandó elemünk a kimeneti visszacsatoló osztó felső tagja, ez a gyári ajánlás szerint $1\text{k}\Omega$ és $10\text{k}\Omega$ közti értéket vesz fel, válasszunk $5.6\text{k}\Omega$ -os ellenállást

$$R_1 = 5.6 \text{ k}\Omega$$

A műveleti meredekség erősítő nem invertáló bemenete és kimeneti között lévő tag ellenállása a következő feltételnek kell, hogy megfeleljen:

$$R_{C1} \gg \frac{2}{gm} \quad \text{ahol } gm \text{ az erősítő meredeksége, } gm = 2300 \mu\text{S az adatlap alapján}$$

$$\frac{2}{2300 \mu\text{S}} = 870 \Omega \quad R_{C1}\text{-et } 8,2\text{k}\Omega\text{-nak választottam}$$

A vele sorosan és párhuzamosan lévő kondenzátorok:

$$C_{C1} = \frac{1}{2\pi \cdot R_{C1} \cdot f_{Z1}} = 17,8 \text{ nF} \approx 18 \text{ nF}$$

$$C_{C2} = \frac{1}{2\pi \cdot R_{C1} \cdot f_{P3}} = 155,3 \text{ pF} \approx 150 \text{ pF}$$

A kimenetre kötött feszültségosztó felső tagjával párhuzamos soros RC tag elemei:

$$R_{F1} = \frac{1}{2\pi \cdot C_{F1} \cdot f_{P2}} = 149,7 \Omega \approx 150 \Omega$$

$$C_{F1} = \frac{2\pi \cdot f_0 \cdot L \cdot C_{ki}}{R_{C1}} \cdot \frac{V_{OSC}}{U_{be}} = 7,751 \text{ nF} \approx 7,5 \text{ nF} \quad \text{ahol } \frac{V_{OSC}}{U_{be}} = \frac{U_{be} \cdot K}{U_{be}} = K$$

Az osztó alsó tagjának meghatározása, ha a kívánt kimeneti feszültség $U_{ki}=3,3\text{V}$

$$R_2 = \frac{U_{ref}}{U_{ki} - U_{ref}} \cdot R_1 = 3,349 \text{ k}\Omega \approx 3,3 \text{ k}\Omega$$

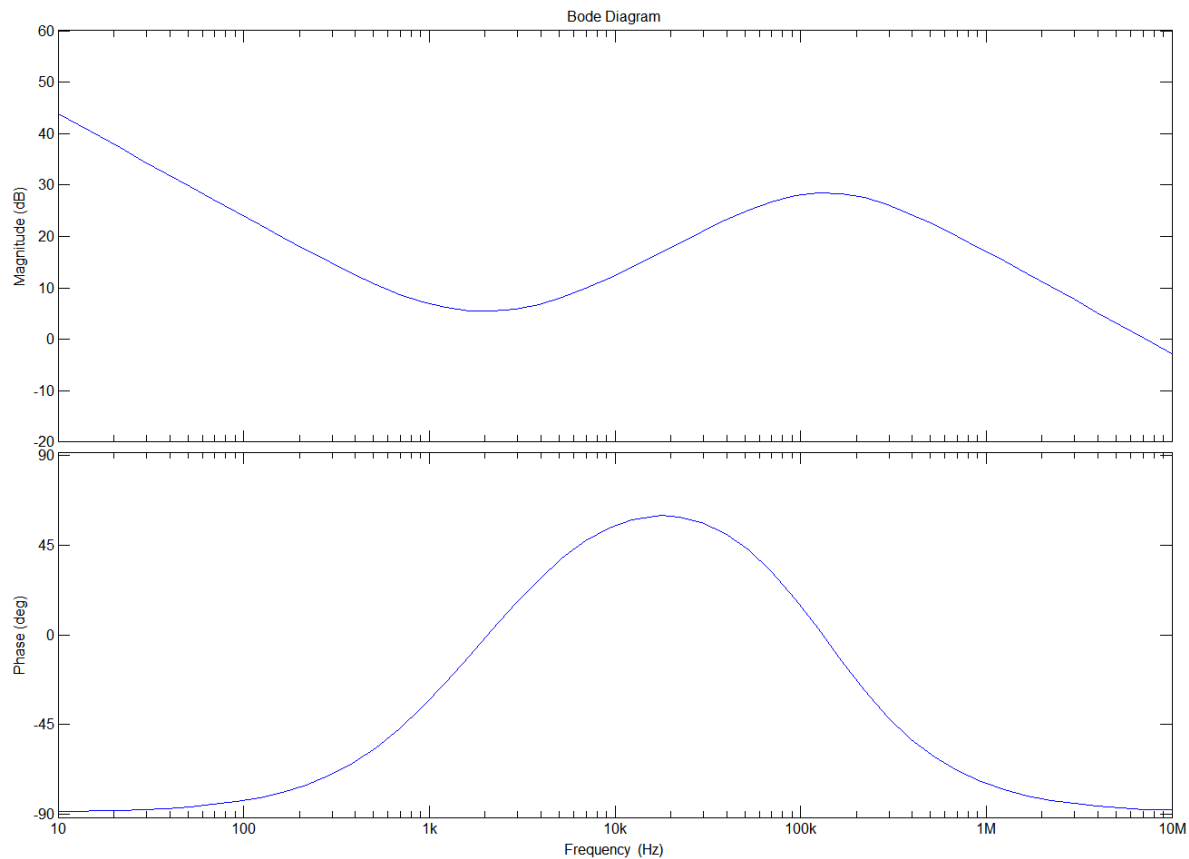
ahol a belső referenciafeszültség, $U_{ref}=1,235\text{V}$

A kimeneti feszültség a valós ellenállásértékekkel:

$$U_{kiv} = \frac{R_1 + R_2}{R_2} \cdot U_{ref} = 3,33V$$

A kompenzáció átviteli függvénye az alábbiak szerint fejezhető ki:

$$G_C(s) = \frac{1}{sR_{F1}(C_{C1} + C_{C2})} \cdot \frac{\left(1 + \frac{s}{2\pi \cdot F_{Z1}}\right) \cdot \left(1 + \frac{s}{2\pi \cdot F_{Z2}}\right)}{\left(1 + \frac{s}{2\pi \cdot F_{P2}}\right) \cdot \left(1 + \frac{s}{2\pi \cdot F_{P3}}\right)}$$

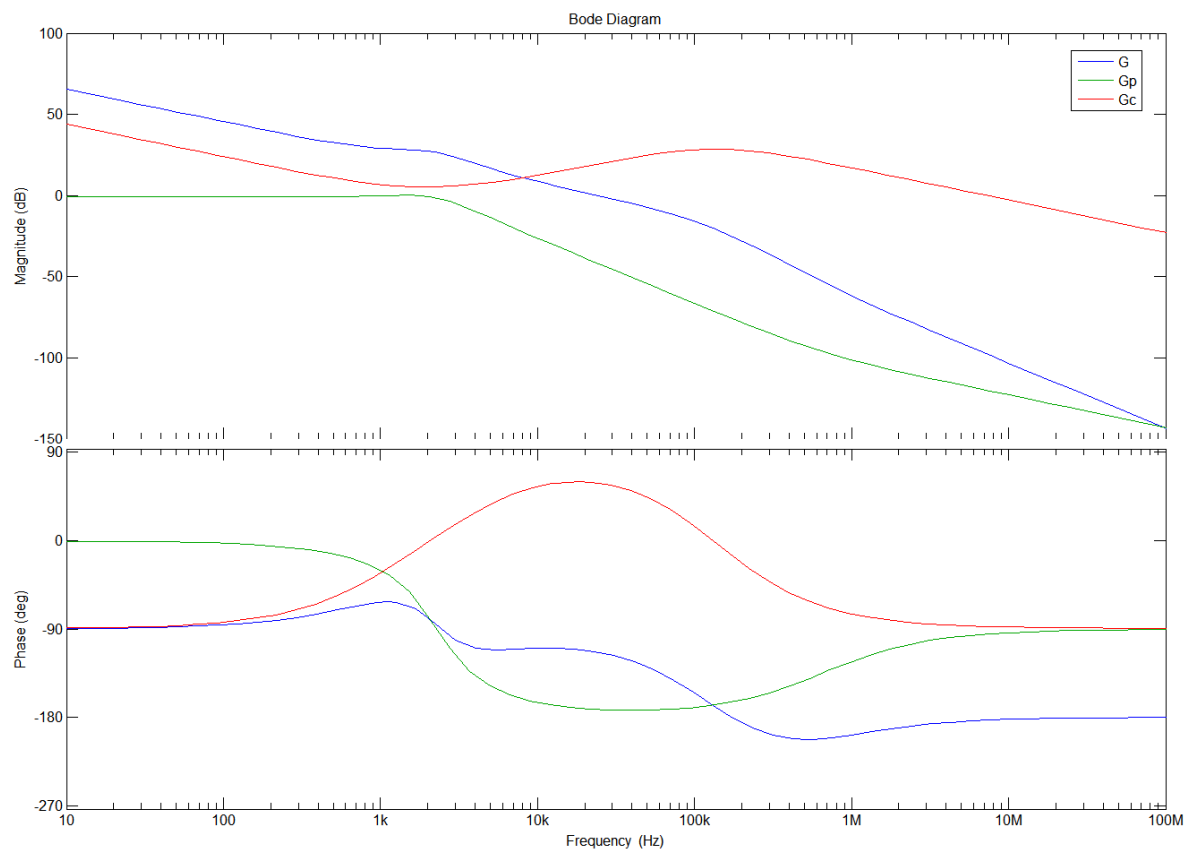


1-1-9. ábra A kompenzáció Bode-diagramja a valós értékek alapján

A felnyitott kör eredő átviteli függvénye:

$$G(s) = G_M(s) \cdot G_P(s) \cdot G_C(s)$$

Ahol: G_M a modulátor, G_P a teljesítményfokozat, G_C az erősítő és a kompenzáló hálózat átviteli függvénye



1-1-10. ábra Szabályozó kör Bode - diagramja

A vágási körfrekvencia és fázistartalék:

$$f_0 = 24,274 \text{ kHz}$$

$$pm = 65,92^\circ$$

A kompenzáció zérusai és pólusai a kiválasztott alkatrészek alapján számolva:

$$f_{z1} = \frac{1}{2\pi \cdot R_{C1} \cdot C_{C1}} = 1,078 \text{ kHz}$$

$$f_{z2} = \frac{1}{2\pi \cdot (R_l + R_{F1}) \cdot C_{F1}} = 4,408 \text{ kHz}$$

$$f_{p2} = \frac{1}{2\pi \cdot R_{F1} \cdot C_{F1}} = 141,471 \text{ kHz}$$

$$f_{p3} = \frac{1}{2\pi \cdot R_{C1} \left(\frac{C_{C1} \cdot C_{C2}}{C_{C1} + C_{C2}} \right)} = 130,473 \text{ kHz}$$

$$f_{p1} = 0$$

1.5. Gyártás és tesztelés után

Pár panel legyártása és tesztelése után kiderült hiányosságok:

- TMDS meghajtók az FPGA rossz BANK-jére kerültek, így ez a funkció nem használható.
- RJ-45 csatlakozó egyik LED-je fordítva került bekötésre.
- RESET gombok és USB csatlakozók footprintjei nem teljesen megfelelő bár beültethetőek és használhatóak.
- FT232RL visszajelző LED-jei a főtápra vannak kötve, így ha az FTDI IC az USB-ről van meg táplálva, akkor nem működnek.
- Az FPGA HSWAPEN bemenete konfigurálásig bekapcsolva tartja minden kivezetésen a felhúzó ellenállásokat ami miatt a rákötött LED-ek halványan világítottak.

Új funkciók, javítások a következő verzióban:

- TMDS bemenet használata, a szükséges felhúzó ellenállások beépítése és kapcsolhatóvá tétele, a DVI-I csatlakozó +5V és a Hot Plug Detect vonalainak kapcsolható tétele, hogy mind bemenetként, mind kimenetként használható legyen a panel.
- Új tápegység áramkörök használata, az új típus AP3211/XRP7659, előnye, hogy olcsó, kisméretű (SOT23-6), 1,4MHz-es kapcsolófrekvenciája miatt kerámiakondenzátorokkal stabil, külső kompenzálást nem igényel. Hátránya a csökkent bemenő feszültségtartomány (4,5-18V), kisebb maximális kimenőáram (1,5A)
- Több tesztpont, U alakú csatlakozás használata a földpotenciálra, így oszcilloszkópok földvezetéke könnyen csatlakoztatható.
- 128Mb-es SPI Flash memória használata a 4Mb-es helyett, így PetaLinuxot is lehet futtatni a panelon.
- APX809-as SOT23 tokozású Power Monitor alkalmazása az 1,2V-os tápegység indítására, ami a 3,3V-os ágot figyeli, hogy a flash memória indulása megfelelő legyen [] Erre a mikrokontrollert is lehetett volna használni, ami egy rövid ideig lehúzva tartja a Program_B bemenetét az FPGA-nak, késleltetve ezzel a konfigurálást, de ezt minden szoftverbe implementálni kell.
- HSWAPEN felhúzása +3,3V-ra, emiatt szükséges az FPGA CSO_B (SPI Chip Select) lába és a +3,3V közé egy 2,4k Ω -os felhúzó ellenállás illesztése.

2. A Szoftver

2.1. Szoftver bevezető

Egy fejlesztőpanel esetében, mint az AndOr, törekedni kell arra, hogy a felhasználónak minél kevesebb fejfájást okozzon annak használata. Gyakori probléma, hogy egy-egy olcsóbb, vagy éppen nem is olyan olcsó tanulópanelhez a hallgatók/felhasználók nem kapnak programozót. Ez plusz időt, illetve utánajárást jelent számukra, az anyagi ráfordításról nem is beszélve. Sokszor csak egy ötlet, vagy teória gyors kipróbálására lenne szükség, esetleg egy példa program kipróbálására, amihez nem feltétlen kellene IDE-ket, és hasonló szoftvereket telepíteni, konfigurálni stb.

Ezekre a problémákra, illetve a kényelem előtérbe helyezésére született meg az ötlet, hogy az AndOr rendelkezzen egy úgynevezett Mass Storage Bootloaderrel.

2.2. Bootloader

A bootloader feladata az, hogy képessé tegye a kontrollerünket arra, hogy már “üzemi állapotban” működve felül tudja írni a saját program memóriáját. Ez egy rendkívül hasznos funkció! Gondoljunk csak bele, ha egy már beüzemelt eszközön szeretnénk szoftvert frissíteni, hibákat orvosolni. Így nem szükséges szétszerelni a készüléket és programozóval csatlakozni rá. A már kialakított környezetben, normál használat közben, megoldható a szoftverfrissítés egy általunk választott kommunikációs csatornán keresztül.

A bootloader működési elve igen egyszerű, a processzor boot szekvenciája után az applikációs szoftver előtt ez a program fut elsőként. Általában a bootloader figyel valamilyen eseményt, amely ha bekövetkezik, a bootloader tényleges funkciója működésbe lép, ellenkező esetben egész egyszerűen a bootloader nem csinál semmit, és az applikációs szoftvernek adja át a végrehajtást. Ez a trigger esemény gyakorlatilag bármi lehet, akár egy jumper áthelyezése a hardveren, vagy egy bizonyos szekvencia fogadása az általunk használt kommunikációs csatornán. Célszerű úgy elkészíteni az applikációs szoftverünket, hogy onnan is át tudjunk ugrani bootloader módba.

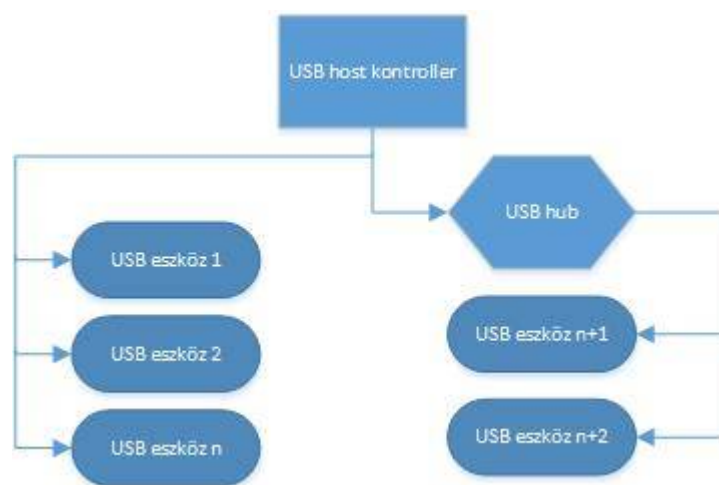
Biztonságkritikus rendszereknél figyelnünk kell arra, hogy a könnyűszerrel való ugrálás bootloader mód és applikációs szoftver mód között veszélyeket rejt magában. Arra az esetre, ha bármi rosszul sülné el a szoftverfrissítés során, célszerű lementeni az aktuális binárist, mielőtt

módosítjuk a memória tartalmát. Ez történhet akár egy kontrolleren kívüli memóriába, akár - ha a mikro vezérlő elég erőforrással rendelkezik - a belső flash egy részét is dedikálhatjuk ennek a funkciónak. Ezután töröljük a flash tartalmát. Előfordulhat, hogy nem szükséges az egész flash memória törlése, például csak egy képet, vagy egy konstans értéket szeretnénk felülírni. Ilyenkor csak azt a bizonyos szektort/szektorokat töröljük le a memóriában, amelyek érintettek. Ha a teljes szoftvert felül szeretnénk írni, célszerű a teljes memóriát törölni az új bináris feltöltése előtt, így megelőzhetjük, hogy az adat korrumpálódjon (memory corruption) a memóriában. Majd harmadik lépésben, az általunk használt kommunikációs csatornán fogadjuk az új bináris állományt, és beírjuk a memóriába. Ez a bináris adat érkezik pl. USB-n keresztül, vagy akár ethernetről, SPI-ről, beolvashatjuk SD kártyáról, gyakorlatilag bármilyen csatornán érkezik, amely rendelkezésünkre áll a processzoron.

Az új bináris letárolása után a bootloader az új bináris kezdetére ugrik. Innentől kezdve már az új applikációs szoftver fut a kontrollerünkön.

2.3. Mass Storage

Az USB szabványnak, illetve a már egyre növekvő tendenciának köszönhetően, miszerint manapság már az ARM kontrollerek nagy része tartalmaz natív USB támogatást, egy igen gyors, és rugalmas eszköz kerül a felhasználók/fejlesztők kezébe. Az USB-re építve könnyűszerrel köthetünk össze különféle eszközöket. Az USB topológiát az 2-0-1. ábrán láthatjuk.

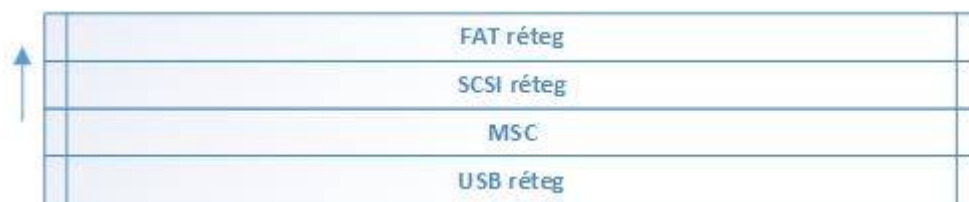


2-2-1. ábra USB topológia

Alapvetően van egy HOST a buszon, amely kézben tartja a kommunikációt a különböző node-k között. Egy hostra 127 eszköz csatlakozhat maximálisan, de úgynevezett USB hub eszközökkel ez megsokszorozható. A kontrollerünk USB eszközként való használatára, a teljesség igénye nélkül néhány példa: USB HID, MSC, CDC. Az említett üzemmódoknak mind dedikált funkciójuk van. HID, vagyis Human Interface Device, ami PC egerekhez, joystickokhoz, billentyűzetekhez használható. CDC, vagyis Communication Device Class, ami hubokhoz, modemekhez, Wifi stickekhez alkalmazandó. Illetve az utolsó példa, amelyet mi is felhasználtunk az AndOr esetében, a Mass Storage, amely egyszerű fájlátvitelt tesz lehetővé az USB host és device között.

2.4. A bootloader felépítése

Az USB MSC (Mass Storage Device Class) több protokoll egymással való együttműködéséből jött létre (2-0-2. ábra). Mint ahogy a neve is mutatja, az egész az USB kommunikációra épül, továbbá szükség van a SCSI (Small Computer Interface) használatára, és szükséges egy fájlrendszer használata is. Ebből a három rétegből épül fel esetünkben a bootloader, kiegészülve egy plusz réteggel, amely a flash írását, olvasását végzi.



2-2-2. ábra Bootloader rétegei

2.5. USB réteg

2.5.1. USB kommunikáció lépései

Minden kommunikáció a buszon három lépésből áll:

1. **Token csomag küldése:**

A Token csomagot a host generálja. Szerepel benne, hogy írás vagy olvasás fog-e következni, milyen címen található a céleszköz a buszon, továbbá annak az eszköznek melyik végpontja a cél. Felfogható egyfajta „megszólításként” a host részéről.

Token csomag típusai lehetnek:

- In - a host értesíti az eszközt, hogy olvasni szeretne
- Out - a host értesíti az eszközt, hogy írni szeretne
- Setup - kontroll átvitel indítását indikálja

Token csomag felépítése	
Mező neve	Mező leírása
Szinkron	Minden csomagnak tartalmaznia kell egy szinkron mezőt. Amely 8 bit a low és full speed USB esetében, illetve 32 bit a high speednél. A küldő és fogadó eszközök órajelét szinkronizálja.
PID(Packet ID)	4 bites identifikáció, ami megmondja, milyen típusú csomagot küldünk. Ez lehet: Token, Data, Handshake, Special.
ADDR (Address)	7 bites cím, ami megjelöli, melyik eszközzel szeretnénk kommunikálni. 7 bitből adódik, hogy 127 eszköz lehet maximum a buszon.
ENDP (Endpoint)	A „megcélzott” végpont azonosítója.
CRC5 (Cyclic Redundancy Checks)	5 bites ellenőrző összeg.
EOP (End of Packet)	A csomag végét jelzi.

2. Adatcsomag küldése:

Itt található a lényegi adat, amelyet közölni szeretnénk. Maximum adatcsomag méretek:

- Low-speed eszközök: 8 bájt
- Full-speed eszközök : 1023 bájt
- High-speed eszközök: 1024 bájt

Adat csomag felépítése				
Sync	PID	Data	CRC16	EOP

3. Státusz csomag küldése:

Az eszköz itt lejelenti, hogy a token, vagy az adat helyesen érkezett-e meg. Továbbá, ha valami oknál fogva az EP nem elérhető, avagy STALL állapotban van, az is itt kerül közlésre.

Lehetséges Handshake csomag fajták (PID értékek alapján):

- ACK – A csomag sikeresen kézbesítve.
- NAK- Értesítés, hogy az eszköz átmenetileg nem képes küldeni, vagy fogadni.
- STALL – Hiba történt a kommunikációban.

Státusz csomag felépítése		
Sync	PID	EOP

A második lépés opcionális. Nem feltétlen történik adatküldés a buszon. Ilyen esetben csak az első és harmadik csomag kerül küldésre.

Az STM32-höz elérhető a gyártó által biztosított USB stack. Az legalsó USB réteget már implementálva kapjuk. Mivel a szoftver csomag tesztelt, nyugodtan használhatjuk.

A hardver minimális inicializálása után beregisztrálunk kettő darab EP-t (End Point), A „végpontok” az adat forrása és céljai a kommunikációban. Szükségünk van egy Bulk Endpoint IN-re, amelyen keresztül a SCSI státuszokat küldi a device a host számára, illetve adatot, továbbá egy Bulk Endpoint OUT, amelyen keresztül a host ír a device memóriájába, és SCSI parancsokat küld neki.

Illetve van egy nulladik EP is, amely fogadja a többi eszköz felénk küldött kontroll és státusz csomagjait, és a buszra való enumerációban játszik szerepet. A PC operációs rendszere innen tudhatja meg, hogy egyáltalán az eszközünk létezik. A nulladik végpont gondoskodik arról is, ha hiba lép fel az MSC kommunikációban. Ilyenkor alaphelyzetbe állítja a BOT (részletesen később) állapotát, és ezt jelzi a host felé is. Minden végpontnak van saját címe, amely alapján azonosítani lehet őket a kommunikációban.

2.5.2. USB réteg a kontrolleren

Az általunk használt végpontok Bulk Transfer típusúak. Az isochronous, illetve az interrupt típusú végpontokkal szemben az a hátrányuk, hogy itt nem biztosított egy lefoglalt sávszélesség a buszon. Például az isochronous típusnál ez fixen biztosítva van. Továbbá a késleltetés időtartama (latency) sem evidens érték ez esetben. Csak a full és a high speed USB módokat támogatja. 8, 16, 32, illetve 64 bájtos, úgynevezett „bulk packet”-ek léteznek a kommunikációban. Esetünkben a maximális, 64 bájtos csomag hosszt választottuk. Megjegyzendő, hogy a high-speed szabvány 512 bájt hosszú csomagokat is nagyszerűen kezel már. Előnyei a 16 bites CRC általi hibaellenőrzés; az adatcsomagunk mindenképpen célba ér, ugyanis kérhető újraküldés a buszon, hibadetektálás esetén.

A minimális funkcionalitáshoz szükségünk van egy úgynevezett USB leíró megírására. Ami gyakorlatilag egy bájt sorozat, amely tartalmazza az összes szignifikáns jellemzőjét az eszközünknek. Végpontok száma, címe, eszközünk típusa, és így tovább.

2.5.3. Az USB leírók

Az USB leíró felépítése:

- Eszköz leíró (Device descriptor)
- Konfiguráció leíró (Configuration descriptor)
- Interfész leíró (Interface descriptor)
- Végpont leíró (Endpoint descriptor)
- Sztring leíró (String descriptor)

2.5.4. Eszköz leíró (Device Descriptor)

Ez a leíró tartalmazza az összes alapinformációt az eszközünkről. Például milyen USB szabványt használ, milyen eszközosztályba sorolható.

Eszköz leíró		
Mező neve	Mező hossza (bájt)	Leírás
bLength	1	A descriptor hossza. 18 bájt jelen esetben.
bDescriptorType	1	A leíró típusa, az eszköz leíró 0x01 értékű.
bcdUSB	2	A legújabb támogatott USB szabvány az eszközünk által. Az AndOr esetében ez 0x2000, mivel 2.0 szabványt használ.
bDeviceClass	1	Eszközosztályt azonosítja, ha ez a mező 0x00 értékű, akkor minden interfész ezt külön definiálja. Ezt a megoldást használjuk mi is.
bDeviceSubClass	1	Az eszközosztálynak lehet alosztálya is, ez szintén 0x00 értékű, mivel az interfész résznél határozzuk majd meg.
bDeviceProtocol	1	Beállítható milyen protokollt használ az eszközünk. Szintén 0x00.
bMaxPacketSize	1	A használt adatsomag hossza. 64 bájt jelen esetben.
idVendor	2	Az eszköz fejlesztőjének az azonosító száma.
idProduct	2	Az eszköz azonosítója.
bcdDevice	2	Az eszköz sorozat száma
iManufacturer	1	Annak a sztringnek a sorszáma, amely tartalmazza a gyártó nevét. ASCII karakterekkel.
iProduct	1	Annak a sztringnek a sorszáma, amely tartalmazza a termék nevét. ASCII karakterekkel.
iSerialNumber	1	A sorozatszám is egy stringben található, ez a mező mondja meg, hanyadik indexen található sztring az.
bNumConfiguration	1	Egy eszköznek több lehetséges konfigurációját adhatjuk meg, itt azt állíthatjuk be, melyek aktívak ezek közül.

Az idVendor és idProduct mezők alapján detektálják az operációs rendszerek, milyen drivert

használjanak az eszközünkkel való kommunikációra.

2.5.5. Konfiguráció leíró (Configuration Descriptor):

Egy USB eszköznek lehet több konfiguráció leírója, a host tudja utasítani az eszközt, hogy ezek között váltson. Ez a leíró meghatározza az USB táp ellátását, interfészeinek számát, illetve maximális áramot, amit felvehet az eszköz. A legtöbb eszköz leginkább egy darab konfiguráció leírót alkalmaz, esetünkben is így van ez.

Konfigurációs leíró		
Mező neve	Mező hossza(bájt)	Leírás
bLength	1	Az aktuális leíró hossza, bájtban megadva. 0x09 ennél a leírónál.
bDescriptorType	1	Ez esetben 0x02 az értéke, amely jelzi, hogy ez egy konfiguráció leíró.
wTotalLength	2	Amikor lekérdezi a host ezt a descriptor, a device leküldi az egész leíró hierarchiát. Ez a mező az egész hierarchia hosszát adja meg bájtban.
bNumInterfaces	1	Az eszköz interfészeinek a száma.
bConfigurationValue	1	Mivel több konfigurációs leírója lehet egy eszköznek, ez a mező azonosítja az egyes konfigurációs leírókat.
iConfiguration	1	Mint már az előző leírónál is láthattuk, megadhatunk sztringeket is, amelyek azonosítják az adott leírót. Ez esetben a string sorszámát itt adhatjuk meg.
bmAttributes	1	Az eszköz energiaellátását választhatjuk ki. 0x40 – self powered, tehát elsősorban nem az

		USB vezeték tápellátását használjuk.
bMaxPower	1	Itt a buszból nyerhető maximális áramot állíthatjuk be. 2mA-es felbontással tehetjük meg.

2.5.6. Interfész leíró

Az interfész leíró, mint ahogyan a neve is mutatja, egy kapcsolódási felületet ír le az eszközünkhöz. A kapcsolódó végpontokat tartja számon, illetve megadhatjuk, milyen protokoll, továbbá milyen device class van használatban az adott interfészen.

Interfész leíró		
Mező neve	Mező hossza (bájt)	Leírás
bLength	1	0x09, a leíró hossza bájtban kifejezve
bDescriptorType	1	0x04, interface descriptor
bInterfaceNumber	1	Interfészből is lehet több egy eszköznek, ez a mező azonosítja az adott interfész leírót.
bAlternateSetting	1	A host küldhet egy SetInterface parancsot a device-nak, ami ezt a mezőt módosítja, tulajdonképpen „on the fly” interfész leíró változtatást tesz lehetővé.
bNumEndpoints	1	Az adott interfészhez kapcsolódó végpontok száma. 0x02 jelen esetben, használunk egy „be” és egy „ki” végpontot.
bInterfaceClass	1	Az USB szövetség által meghatározott osztályok, illetve alsztályok állíthatók be. Ezek írják le pl. a HID, Mass Storage üzemmódokat. MSC esetében 0x08 az interfész osztály, 0x06 az alosztály, amely a SCSI transparent módot jelöli. 0x50 pedig a bulk only transportot jelöli meg az
bInterfaceSubClass	1	
nInterfaceProtocol	1	

		interface protocol mezőben.
iInterface	1	Az interfész string leírójának az indexe.

2.5.7. Végpont leíró

Az interfészre csatlakozó végpontokat is jellemeznünk kell, kivéve a nulladik végpontot, a fentebb leírt okokból. Minden végpontnak adnunk kell egy címet, azonosítót, meghatároznunk a típusát, illetve a maximális csomag méretet, amelyet használunk rajta keresztül.

Végpont leíró		
Mező neve	Mező hossza (bájt)	Leírás
bLength	1	Leíró hossza, 0x07
bDescriptorType	1	Descriptor típusa, végpont leírónál 0x05 ez az érték.
bEndpointAddress	1	A legfelsőbb biten a végpont irányát (be, ki) adjuk meg. 0 – bemenet, 1 – kimenet. 0-3. biteken maga a cím helyezkedik el. 4-6 bitek foglalt értékek, nullára szokás állítani.
bmAttributes	1	0-1 bitek: átvitel típusa 00 – Control 01 – Isochronous 10 – Bulk 11- Interrupt 2-3 bitek: szinkronizáció típusa 00- nincs szinkronizáció 01- aszinkron működés 10- adaptív 11- szinkron működés a busszal

		4-5 bitek: végpont típusa 00- Data végpont 01- Feedback végpont 10- Explicit Feedback Data Endpoint 11- Foglalt érték Isochronous végpontnál 2-7. bit foglaltak, nullába kell állítani.
wMaxPacketSize	2	A maximális csomag méret, amelyet képes küldeni/fogadni a végpont. Mi a maximális 64 bájt hosszt használjuk.
bInterval	1	Bulk végpontnál ezt a mezőt nem használjuk, egyébként az átviteleknel a polling idejét fejezi ki.

A fent említett leírókon kívül még használunk sztring leírókat. Láthattuk az egyes leírók tartalmazhatnak sztring indexeket, amelyek emberi szem számára is könnyen emészthető információval szolgálnak az egyes leírókról. Használatuk nem kötelező, ilyen esetben az egyes index mezőket nullára kell állítani.

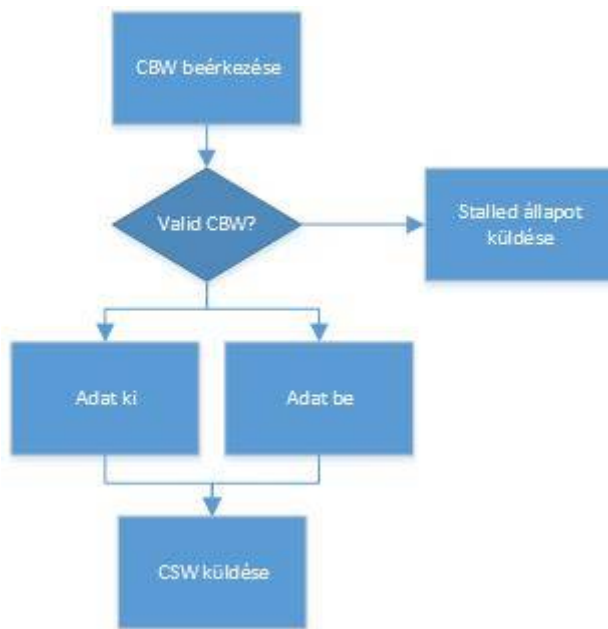
Ha a fent említett leírókat megfelelően beállítjuk, majd a host kérésére leküldjük számára, abban az esetben a számítógép már felismeri az eszközt, mint Mass Storage eszköz, enumeráció után.

2.6. Bulk Only Transfer állapotgép

Ezután implementálnunk kell egy úgynevezett BOT(Bulk Only Transfer) állapotgépet (2-0-3. ábra), amelynek négy állapota van:

1. Parancs (CBW)feldolgozása
2. Adat ki
3. Adat be
4. Státusz (CSW) küldése

5. Stalled állapot



2-2-3. ábra BOT állapotgép

Adat be, adat ki normális esetben nem lehetnek együtt egy körben. Mindig ellenőriznünk kell a megkapott CBW blokkokat. Könnyen eldönthető, hogy érvényes-e az adott blokk, ha ellenőrizzük annak hosszát, dCBWSignature mezőjét. Célszerű még ellenőriznünk, hogy ha érvényes is az adott CBW, van-e feldolgozható tartalma számunkra. Ezek kiderülnek, hogy általunk használt LUN került-e megcímezésre (bCBWLUN mező) és a CBWCB mező hossza érvényes hosszúságú adatot tartalmaz-e.

Ha a fenti ellenőrzéseken megbukik a kapott CBW blokk abban az esetben az úgynevezett stalled állapotot küldjük az érintett végpontra. Ez jelzi a hostnak, hogy hiba volt a kommunikációban, és meg is szakítja azt.

A host két módon kommunikál eszközünkkel, CBW (Command Block Wrapper), illetve CSW(Command Status Wrapper) csomagokon keresztül.

2.6.1. CBW

Bit	7	6	5	4	3	2	1	0
Bájt								
0-3	dCBWSignature							
4-7	dCBWTag							
8-11	dCBWDataTransferLength							
12	bmCBWFlasgs							
13	Foglalt				bCBWLUN			
14	Foglalt			bCBWLength				
15-30	CBWCB							

dCBWSignature: A CBW blokk egyedi azonosítóval rendelkezik, ami fixen 0x43425355.

dCBWTag: A host meghatároz egy azonosító taget, amelynek meg kell majd egyezni a CSW-ben visszaküldött értékkel.

dCBWDataTransferLength: Ha adatközlés következik, ebben a mezőben van meghatározva annak hossza. Ha nem történik adatátvitel a CBW után, akkor ez a mező nulla értékű.

bmCBWFlasgs:

7. bit – Az adat mozgásiránya, ha értéke 0 host -> device; ha 1, akkor device -> host.

6-0. bit – Nem használt

bCBWLUN: Mivel több logikai egységként is felcsatolhatunk Mass Storage meghajtókat, ezzel a mezővel tudjuk identifikálni őket. Esetünkben kettő LUNt használunk, így logikailag szét tudjuk választani az FPGA konfigurációs memóriáját, illetve a mikrokontroller belső memóriáját, és külön meghajtóként felcsatolni a PC-re.

bCBWLength : CBW blokk hossza

CBWCB: Maga a parancs, amelyet az eszköznek szánt a host

2.6.2. CSW

Bit	7	6	5	4	3	2	1	0
Bájt								
0-3	dCSWSignature							
4-7	dCSWTag							
8-11	dCSWDataResidue							
12	bmCSWFlasgs							

dCSWSignature: A CSW blokk azonosítója, ez minden CSW blokknál megegyezik. Értéke 0x53425355

dCSWTag: A CBW megegyező nevű mezőjével azonos. Gyakorlatilag így lehet a CBW és CSW blokkokat párba állítani, továbbá hibadetektálásra is használhatjuk.

dCSWDataResidue: Ebben a mezőben adja meg az eszköz, hogy a várt bájt mennyiség helyett mennyit kapott. Itt a fennmaradó bájtok számát kell megadni.

bmCSWFlasgs:

- 0x00 -Parancs feldolgozva
- 0x01- Parancs feldolgozásában hiba történt
- 0x02- Fázis hiba (Phase error)
- 0x03-0xFF – Nem használt értékek

A kommunikáció a host által kezdeményezett. A host leküld egy CBW blokkot a kiválasztott eszköz számára a buszon. Majd az eszköz feldolgozza a parancs mezőt, miután validálta azt, tehát a dCBWSignature mező megfelelt, illetve a CBWCB mezőt tudta értelmezni, akkor létrejön az adatátvitel. Ha nem volt adatátviteli kérelem, abban az esetben egyből; ha volt, akkor az adatblokk után az eszköz visszaküldi a CSW blokkot, amiből a host értelmezni tudja, hogy sikeres volt-e az átvitel, avagy a parancsot tudta-e értelmezni az eszközünk, esetleg újraküldésre van-e szükség.

2.7. 2. 5. A SCSI réteg

A fentebb említett CBW blokk szabvány SCSI parancsokat használnak a CBWCB mezőbe ágyazva, gyakorlatilag az adat helyén. Ezeket a parancsokat, illetve státusz értékeket használja minden USB Mass Storage eszköz. Mivel esetünkben nem egy teljes funkcionalitással bíró MSC eszközt kell készítenünk, elég néhány SCSI parancs/státusz implementálása.

2.7.1. Implementált SCSI parancsok

- SCSI_INQUIRY (0x12)

A host lekérdezi a SCSI réteggel kapcsolatos információkat. Ebben az esetben a device egy minimum 36 bájt hosszúságú bájt sorozattal válaszol. Jelen helyzetben, mivel csökkentett funkcionalitású SCSI réteget használunk, elegendő ez a 36 bájt hosszúság.

A válaszban szerepelnie kell:

Bájt	Mező leírása	Beállított érték
0	Periféria típusa, ami esetünkben közvetlen hozzáférésű eszköz	0x00
1	Eltávolítható eszköz	0x80
2	Az eszköz milyen szabványt támogat SCSI szempontból. A miénk SPC3, ez a legtöbb operációs rendszer által támogatott.	0x05
3	Megmondja a host számára, milyen formátumban fog választ adni. Ez szintén SPC3.	0x02
4	Mivel a szabvány csak a minimális hosszát határozza meg a blokknak, ezért itt meg kell adni, hogy a fentebb említett kötelező adatok után, milyen további hosszúságú adatot fogunk küldeni.	0x1F
5-7	Nem relevánsak esetünkben.	0x00
8-15	Azonosító stringet tárolhatunk itt a VENDOR-ra vonatkozóan.	„VENDOR”
16-31	Termék identifikáló string leíró rész	„AndOr”

32-35	Termékszám leíró rész.	„V1.0”
35-36	Termék sorozatszám	„0x00”

- **SCSI_READ_10 (0x28)**

A host olvasási igénnyel fordul az eszközünk felé. A kapott információkból kiderül számunkra, melyik logikai bloktól szeretné olvasni a host, ez egy 4 bájtos érték. (LBA) Illetve kettő bájton közli milyen hosszban szeretnénk adatot olvasni, blokkokban kifejezve.

- **SCSI_WRITE_10 (0x2A)**

A host írni szeretne az eszközünk memóriájába. Mint ahogyan SCSI_READ_10 parancsnál, itt is megmondja, milyen hosszban szeretne írni, illetve honnantól. A parancs után az eszköz felkészül a fogadásra.

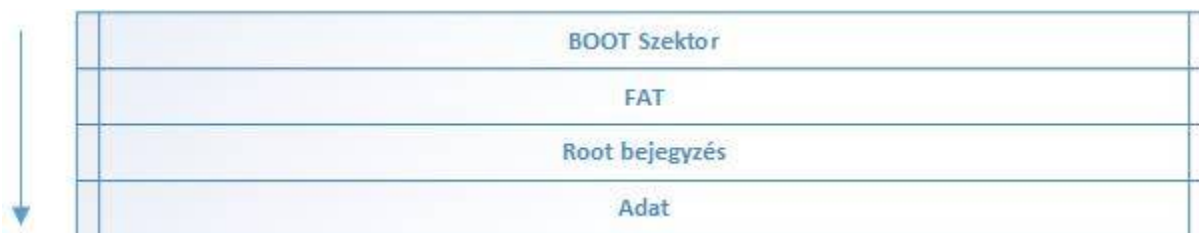
- **SCSI_READ_CAPACITY_10 (0x25)**

A host lekérdezi, hogy milyen nagy kapacitással rendelkezik az eszközünk. Minden LUN-nak külön kell elküldenünk, illetve külön kerül lekérdezésre. Eszközünk egy 8 bájtos értékkel felel.

2.8. FAT

Ahhoz, hogy a memóriánk tartalma a PC számára is könnyen kezelhető legyen, illetve ahhoz, hogy fájlokat tudjunk mozgatni a memóriába, szükségünk van egy fájlrendszerre. Célszerű valamilyen elterjedt fájlrendszert választani, mivel a belső memóriája a kontrollernek 64kbájt, illetve az FPGA konfigurációs memóriája is 4Mbit ezért a FAT12 (2-0-4. ábra) fájlrendszerre esett a választás, de persze minimális módosítással ez megváltoztatható, akár FAT32-re is, ha szükség lenne rá. Esetünkben elégséges a 12-es FAT.

Mivel a fájlrendszer feladata jelen esetben igen korlátozott, csak minimális funkcionalításra van szükségünk, illetve azért, hogy tárhelyet takarítsunk meg (ne feledjük, a bootloader a felhasználó applikációs szoftverétől veszi el a helyet) elegendő egy úgynevezett pszeudó FAT12 implementációt alkalmaznunk.



2-2-4. ábra FAT felépítése

2.8.1. Pszeudó FAT

Egyszerűen a PC-vel „elhitetjük”, hogy ő egy teljes értékű fájlrendszerrel rendelkező eszközt lát, és kommunikál vele. Amikor lekérdezi a FAT rendszer megfelelő részeit, mindig egy konstans értéket küldünk le neki, minden újraindítás után ugyan az a fájl szerepel a gyökér könyvtárban, nem marad konzisztens a rendszer, mivel nincs is erre szükségünk. Amit látni szeretnénk a meghajtó felcsatolásakor, az egy darab file, amely reprezentálja a binárisunkat, és felül tudjuk írni azt. Ahhoz, hogy ezt megtegyük, készítenünk kell két bájtsorozatot. Az egyik a boot szektort valósítja meg, a másik pedig az úgynevezett root bejegyzést írja le.

2.8.2. A boot szektor

A boot szektor tartalmazza a lemez paramétereit, amelyen a fájlrendszerünk helyezkedik el. A boot szektor alapvetően, mint ahogyan a neve is mutatja egy egész szektornyi területet foglal el. Ám helyspórolás miatt, illetve mivel nem teljes értékű fájlrendszert készítünk, nekünk elég 32 bájts információt, amit konstansul eltárolunk. Fontos, hogy a boot szektor utolsó két bájta 0x55, 0xAA legyen, ez a szektor azonosítója. A nem említett indexen lévő bájtok helyére nullákat töltünk fel. Továbbá megjegyezendő, hogy a két bájtos értékek little endian formátumban vannak tárolva

Bájt index	Leírás
0-2	Jump utasítás. Abban az esetben használjuk, ha erről az eszközről szeretnénk bebootolni egy PC-t. Most csak feltöltjük egy valid értékkel.
3-10	Annak az operációs rendszernek az OEM azonosítója, amelyen meg lett formázva a lemez.
11-12	Bájtok száma egy szektorban. Normál esetben ez 512, jelen esetben is ennyi.
13	Szektorok száma egy klaszterben. Lehetséges értékek: 1, 2, 4, 8, 16, 32, 64, 128. 64 szektort használunk fel.
14-15	Foglalt szektorok. FAT12-nél ez az érték 1.
16	FAT másolatok száma. Tipikusan ez kettő, de nekünk elegendő egy.
17-18	Root könyvtárban található bejegyzések maximuma. Nálunk egy darab fájl van itt, mivel csak a firmware binárisát tartalmazza.
19-20	A fájlrendszerben található összes szektor száma.
21	Média típusa. Itt jelezzük, hogy fix méretű lemezt használunk.
22-23	Mivel tárolhatunk több FAT kópiát, itt meg kell adnunk, egy kópiában mennyi szektor van.
24-25	Szektorok száma egy cilinderen.
26-30	Rejtett szektorok száma

2.8.3. Root entry

Létre kell hoznunk egy másik konstans bájt sorozatot, ami leírja a kötetünket, illetve a benne található fájlt. A kötetnek kell egy nevet adnunk, ez a név lesz látható az operációs rendszerben, mint az eszközünk neve. A benne található fajlnak is kell egy nevet adnunk, tipikusan a firmware binárisunkat jelöljük itt, illetve a rendelkezésre álló flash memóriát adjuk meg neki, mint méret.

Így a felhasználó egyszerűen tájékozódhat róla, mennyi hellyel is gazdálkodhat.

A fájlok, mappák, kötetek leírása mind egy szisztéma alapján történik, amelyet az alábbi táblázat foglal össze.

Bájt index	Leírás
0-10	8+3 szabály alapján, 8 karakterben megadhatjuk a kötet vagy fájl nevét. 3 karakterben a fájl kiterjesztését. Ha kötet leírást készítünk, abban az esetben a kiterjesztés szóköz karakterrel feltöltendő. ASCII karaktereket használva.
11	Atribútumok. 0. Csak olvasható a fájl 1. Rejtett fájl 2. Rendszerfájl, a leíró egy rendszerfájlhoz tartozik 3. Kötet név, a leíró egy kötetet jelöl 4. Mappa, a leíró egy mappához tartozik 5. „Achieve flag”, ha a fájl változott, akkor jelez 6. -7. Használaton kívüli
12	Windows NT számára fentartott bájt, nem használjuk
13	Létrehozás idejének milliszekundum paramétere. 0- 199 között lehet az érték.
14-15	Létrehozás ideje 0-4. bit másodperc 5-10. bit perc 11-15. bit óra
16-17	Létrehozás dátuma 0-6. bit év (0-127 tartományban, 1980-tól számolva) 7-10. bit hónap 11-15. bit nap
18-19	Utolsó fájllelés dátuma
20-21	FAT32 számára fentartott bájtok
22-23	Utolsó írás ideje
24-25	Utolsó írás dátuma
26-27	A leíróhoz tartozó fájl első darabját tartalmazó clusterre mutató cím.

A fenn részletezett FAT részletek kifejtése után már összerakhatjuk a pszeudó FAT rendszerünket. Miután így összeállt a három, USB, SCSI, FAT rétegeinkből kapott kombináció, a kontroller készen áll az új firmware fogadására.

2.8.4. Bootloader összefoglaló

A fent említett elemek összeillesztéséből összeáll a teljes bootloader. Ahhoz, hogy ez a szoftver működőképes legyen, mind a processzor, mind az FPGA számára csak egyetlen módosítást kell végre hajtánunk. Mégpedig annyit, hogy amikor beérkezik a SCSI parancs, és törölni szeretnénk a binárisunkat, továbbá felülírni, akkor a LUN számától függően vagy IAP (In Application Programming) módszerrel a Cortex flash memóriáját írjuk felül, vagy SPI interfészen keresztül az FPGA konfigurációs memóriáját változtatjuk meg.

Az egyszerűség kedvéért amikor a konfigurációs memóriát írjuk felül első lépésben az egész tartalmát letöröljük, majd az új tartalmat feltöltjük. Mivel ugyan arra az SPI buszra csatlakozik az ARM és a SPARTAN is, ezért fontos, hogy ameddig a konfigurációs memóriát módosítjuk az FPGA-t tartsuk reset állapotban. Ha végeztünk a feltöltéssel, akkor elengedjük a reset lábat, és, ha minden rendben történt, az FPGA betölti a konfigurációját a külső memóriájából.

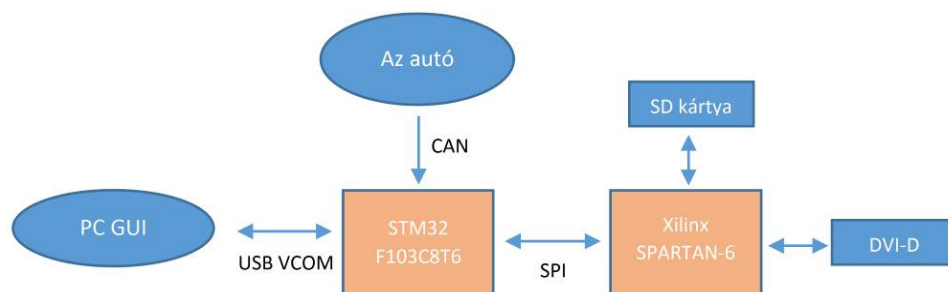
2.9. Példa project bemutatása az AndOr fejlesztőpanelen

2.9.1. A feladat leírása

Készítsünk egy olyan CAN busz sniffert, amely egy autóba helyezve, és annak CAN buszára csatlakoztatva olvassa az autó menet közbeni tulajdonságainak adatait. Képesek legyünk egy PC-n keresztül beállítani mely tulajdonságokat szeretnénk figyelni, a többit hagyjuk figyelmen kívül. A sniffelés induljon a gyújtás ráadásától és tartson a gyújtás elvételéig. A kívánt adatokat mentjük el, továbbá legyünk képesek a megadott adatokból grafikont rajzolni, különböző statisztikákat megjeleníteni PC nélkül.

2.9.2. Megoldás a panelen

A modern autókban, mint tudjuk az általánosan használt kommunikáció a CAN buszon történik. Egyes gyártók a felhasználók számára közzé adnak bizonyos információkat az itt folytatott kommunikációról, úgymint sebesség, vagy történt-e gyújtás ráadás, fékezés, és így tovább. Megjegyezendő, hogy rengeteg üzenet így is rejtve marad a felhasználó elől, legalábbis nincs róla információja, milyen ID-val, milyen üzeneteket látunk.



2-0-9. ábra A felhasznált AndOr perifériák

A sniffelés megkezdése előtt a natív USB porton csatlakoztassuk a számítógépünkhöz az AndOr-t. Ehhez szükséges az úgynevezett VCOM osztály implementálása, amelynek részletezésétől eltekintünk jelen írásban. Tulajdonképpen egy egyszerű soros kapcsolatot alakítunk ki a PC és az STM32 között. A számítógépen szükségünk van egy felhasználói felületre, amelyen keresztül például check boxokkal kiválaszthatjuk egy listából, mely üzeneteket kívánunk figyelni, továbbá ezekből milyen statisztikákat szeretnénk készíteni. Célszerű, ha a felhasználó egyszerűen tud bevinni új választható üzenet típusokat, és ezekre is tudja alkalmazni a meglévő statisztikai funkciókat. Olyan esetről is lehet szó, amikor nem egy statisztikát szeretnénk készíteni, hanem egyszerűen idő függvényében megjeleníteni az adatokat (pl. sebesség), és egy-egy kiemelt értéket-értékeket figyelni/keresni az adatok között, például hirtelen lassulás, motorhiba. Javasolandó különböző profilok kiválaszthatósága, különböző autó típusokhoz. Egy igazán dinamikus bővíthető GUI alakítható ki, megfelelő szemlélettel.

Miután beállítottuk a kívánt paramétereket célszerű elmenteni azokat. Gondoljunk arra az esetre, ha nem az autóban ülve egy notebookkal állítjuk be a paramétereket. Erre kiválóan használható az SD kártya. Az SD kártya közvetlenül az FPGA-ra kötve található meg, még hozzá 4 bites párhuzamos SD módban. A felhasználó itt először kerül testközelbe a multiprocesszoros rendszerrel. Ugyanis a PC-ről érkező adatokat a kontroller SPI interfészen keresztül küldi az FPGA-nak, amely kiírja azokat az SD kártyára. A két processzor zökkenőmentes együttműködése miatt implementáljunk egy saját protokollt, amely használatával a két központi egység tud kommunikálni az SPI interfészen keresztül.

- Gondolhatunk arra az esetre is, mikor az említett notebookkal ül a felhasználó az autóban és egyből használja is az eszközt. Ebben az esetben nem feltétlenül szükséges, a GUI-ból is indíthatjuk a sniffelést. -

Az FPGA-ba célszerű implementálni egy Microblaze soft processzort, amely tartalmazza az SD interfészt, az SPI interfészt, illetve tudja kezelni a FAT file rendszert. Illetve ide illeszthetjük még a videó kimenetünket is, mint periféria.

Miután elkészült ez az összeállítás, az első újraindítás után – kivéve, ha nem GUI indította a sniffelést – a Microblaze leellenőrzi, hogy rendelkezésre áll-e konfigurációs állomány, ha igen, akkor leküldi a benne lévő paramétereket a kontrollernek, amely miután felparaméterezte a CAN perifériát jelez az FPGA-nak. Majd a soft processzor készít egy új állományt az SD kártyán, amely után jelez az STM32-nek, hogy indulhat az adatgyűjtés. Ezen a ponton

megbizonyosodtunk a központi egységek készen állnak a loggolás indítására.

Innentől pedig figyeljük a CAN buszt az autóban, várunk a gyújtás ráadása üzenetre.

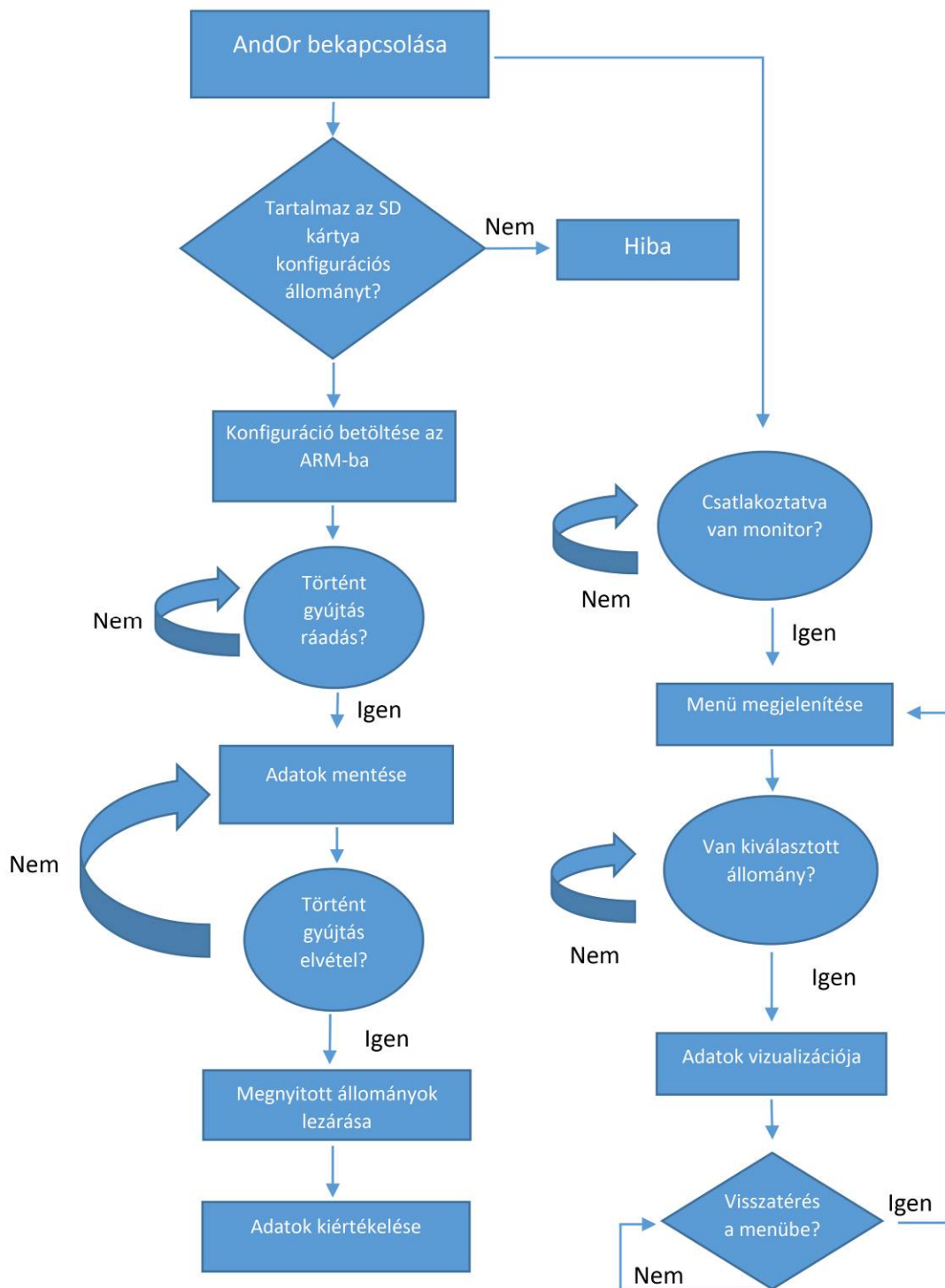
A működés közben csak az előre beállított ID-jű üzeneteket, illetve azok payload részének is egy részét vizsgáljuk. Mivel egy CAN üzenet nem az összes payload bájtja tartalmazza a sebesség adatokat például. Ezekről bővebb információt az autógyártó által kiadott dokumentumokban találunk. Ha az ID megfelelt az elvártnak - GUI-ban beállított -, akkor a payloadból kinyerjük a releváns információt, ezt egy keretbe illesztjük a fenn említett ARM – FPGA protokoll segítségével, majd elküldjük a Microblaze-nek. Az FPGA-ban lévő soft processzor pedig kiírja azokat a megnyitott állományba az SD kártyára.

Ha a gyújtás elvéve jelzés érkezik leáll az adatgyűjtés. A Microblaze lezárja az éppen megnyitott állományt. Megvizsgálja milyen matematikai műveleteket, milyen adatokkal kell elvégeznie, ezeket meg is teszi, és külön állományokba menti.

Ezzel a módszerrel az SD kártyán lévő hely függvényében tudunk új állományokat létrehozni.

Abban az esetben, ha a soft processzor érzékeli, hogy a videó kimenetre csatlakoztattunk egy monitort, a begyűjtött adatok vizualizációja elkezdődik. Egy egyszerű, de igényes menüt kialakíthatunk, amelyen a panelen lévő gombokkal tudunk navigálni a már elkészített állományok között amelyeket kiválasztva a monitoron grafikon formájában viszontláthatjuk adatainkat.

FAT file rendszer használata azért is előnyös, mert a kártyát kivéve az AndOr-ból számítógépben is megtekinthetjük állományainkat. Érdekes olyan file struktúrába menteni az adatokat, amelyek megnyithatóak elterjedt programokkal is, például .CSV kiterjesztés ideális erre az esetre amely megnyitható akár MATLAB-ban is, vagy Excel-ben is akár, további analízist végezni rajta.



2-0-10. ábra A feladat megoldásának folyamatábrája

3. Melléklet

3.1.1. MatLab script impedanciaszámításhoz

```
%Konstansok a panel felépítése és adatlapja alapján
Er = 4.58; t=1.4; h =7.87;
Zf0 = 376.73031; %szabad tér hullámellenállása
e = 2.718281828459; %euler szám
%{
str = 'A dielektrikum relatív permittivitása: '; disp(str);
str = 'Er = '; Er=input(str);
str = 'A dielektrikum magassága: '; disp(str);
str = 'h = '; h=input(str);
str = 'A fémzés vastagsága: '; disp(str);
str = 't = '; t=input(str);
}%
str = 'A jelvezeték szélessége: '; disp(str);
str = 'W = '; W=input(str);
str = 'A jelvezetékek távolsága: '; disp(str);
str = 's = '; s=input(str);

u=W/h;
g=s/h;
%-----
%effektív jelvezeték szélesség:
%homogén közeg (stripline):
%W1 =W+ t/(h*pi)*log(1+4*e/(t/h)*tanh((sqrt(6.517*W))^0.5)^2);
dW1=(t/(h*pi))*log(1+(4*e)/((t/h)*coth(sqrt(6.517*W))^2));
%inhomogén közegre (microstrip):
dWr = 0.5*dW1*(1+sech(sqrt(Er-1)));
W1=W+dW1; Wr=W+dWr;
%effektív dielektromos állandó:
```

$$a = 1 + (1/49) * \log(((u^4) + (u/52)^2) / ((u^4) + 0.432)) + (1/18.7) * \log(1 + (u/18.1)^3);$$

$$b = 0.564 * (((Er - 0.9) / (Er + 3))^{0.053});$$

$$E_{eff} = (Er + 1) / 2 + ((Er - 1) / 2) * ((1 + 10 * (h / W_r))^{-(a * b)});$$

%hullámimpedancia

$$f_u = 6 + (2 * \pi - 6) * \exp(-(30.666 * h / W_r)^{0.7528});$$

$$Z_{L1} = Z_{f0} / (2 * \pi) * \log(f_u * h / W_1 + \sqrt{1 + (2 * h / W_1)^2});$$

$$Z_L = Z_{L1} / \sqrt{E_{eff}};$$

%-----

%differenciális impedanciaszámítások IPC-2141A alapján

$$a_0 = 0.7287 * (E_{eff} - 0.5 * (Er + 1)) * (1 - e^{(-0.179 * u)});$$

$$b_0 = (0.747 * Er) / (0.15 + Er);$$

$$d_0 = 0.593 + 0.694 * e^{(-0.562 * u)};$$

$$c_0 = b_0 - (b_0 - 0.207) * e^{(-0.414 * u)};$$

$$E_{eff0} = (0.5 * (Er + 1) + a_0 - E_{eff}) * (e^{-c_0 * g^{d_0}}) + E_{eff};$$

$$Q_1 = 0.8695 * u^{0.194};$$

$$Q_2 = 1 + 0.7519 * g + 0.189 * g^{2.31};$$

$$Q_3 = 0.1975 + ((16.6 + (8.4 / g)^6)^{-0.387} + 1 / 241 * \log((g^{10}) / (1 + (g / 3.4)^{10}));$$

$$Q_4 = 2 * Q_1 / Q_2 * ((e^{-g}) * (u^{Q_3}) + (2 - e^{-g}) * u^{-Q_3});$$

$$Q_5 = 1.7194 + 1.14 * \log(1 + 0.638 / (g + 0.517 * g^{2.43}));$$

$$Q_6 = 0.2305 + 1 / 281.3 * \log((g^{10}) / (1 + (g / 5.8)^{10})) + (1 / 5.1) * \log(1 + 0.598 * g^{1.154});$$

$$Q_7 = (10 + 190 * g^2) / (1 + 82.3 * g^3);$$

$$Q_8 = e^{(-6.5 - 0.95 * \log(g) - (g / 0.15)^5)};$$

$$Q_9 = \log(Q_7) * (Q_8 + 1 / 16.5);$$

$$Q_{10} = (1 / Q_2) * (Q_2 * Q_4 - Q_5 * e^{(\log(u) * Q_6 * u^{(-Q_9)})});$$

$$Z_{odd} = Z_L * ((\sqrt{E_{eff} / E_{eff0}}) / (1 - ((Z_L / Z_{f0}) * Q_{10} * \sqrt{E_{eff}})));$$

$$Z_{diff} = 2 * Z_{odd};$$

%kiíratások

```
str=sprintf('ZL = %0.2f Ohm',ZL);  
disp(str)  
str=sprintf('Zodd = %0.2f Ohm',Zodd);  
disp(str)  
str=sprintf('Zdiff = %0.2f Ohm',Zdiff);  
disp(str)
```

3.1.2. MatLab script a kompenzáció számításához

```
%Type 3 B / PID kompenzáció számítás
%formázások, konstansok
format long g
opts = bodeoptions;
opts.Title.FontSize = 12; opts.YLabel.FontSize = 12;
opts.XLabel.FontSize = 12; opts.TickLabel.FontSize = 12;
opts.FreqUnits = 'Hz';
s=tf('s'); K = 0.076; Uref = 1.235; Fs=250000;
Gm = 1/K; R1=5600; nl = sprintf('\n');
%-----
%{
prompt = 'gm [uS] = ';
gm = input(prompt)*10^-6;
%Kimeneti szuro:
disp('Kimeneti LC szuro:')
prompt = 'L [uH] = ';
L = input(prompt)*10^-6;
prompt = 'Cki [uF] = ';
Cki = input(prompt)*10^-6;
prompt = 'RESR [mOhm] = ';
ESR = input(prompt)*10^-3;
prompt = 'RDCR [mOhm] = ';
RDCR = input(prompt)*10^-3;
%Kívánt kimeneti értékek:
disp('Kimeneti értékek:')
prompt = 'Uki [V] = ';
Uki = input(prompt);
prompt = 'Iki [A] = ';
Iki = input(prompt);
```

```

Rt = Uki/Iki;
%}
%adatok megadva:
gm=2300*10^-6; L=22*10^-6; Cki=242*10^-6; ESR=0.001;
RDCR=0.303; Uki=3.3; Rt=3.3;
%-----
%Számítások:
%-----
%A teljesítményfokozat átviteli függvénye:
disp(nl)
disp('A teljesítményfokozat átviteli függvényé: ')
disp('Gp(s) =')
Gp=tf([Cki*ESR 1], [(Rt+ESR)/Rt]*L*Cki
(L/Rt+ESR*Cki+RDCR*Cki+(RDCR*ESR*Cki)/Rt) 1+RDCR/Rt])
Fp0 = 1/(2*pi*sqrt(L*Cki));
Fz0 = 1/(2*pi*ESR*Cki);
disp('A teljesítményfokozat pólusa:')
str = sprintf('Fp0= %0.3f kHz',Fp0/1000);
disp(str)
disp('A teljesítményfokozat zérusa:')
str = sprintf('Fz0= %0.3f kHz',Fz0/1000);
disp(str)

prompt = 'Választott vágási körfrekvencia: [kHz] = ';
Fo = input(prompt)*10^3;
prompt = 'A kívánt fázistartalék [°] = ';
fmargin = input(prompt)/180*pi;

%A pólusok és zérusok kiszámítása:
Fz2 = Fo*sqrt((1-sin(fmargin))/(1+sin(fmargin)));
Fp2 = Fo*sqrt((1+sin(fmargin))/(1-sin(fmargin)));

```

```

Fz1 = Fp0*0.5;
Fp3 = Fs/2;

str = sprintf('Fz1= %0.3f kHz',Fz1/1000);
disp(str)
str = sprintf('Fz2= %0.3f kHz',Fz2/1000);
disp(str)
str = sprintf('Fp2= %0.3f kHz',Fp2/1000);
disp(str)
str = sprintf('Fp3= %0.3f kHz',Fp3/1000);
disp(str)
disp(nl)
%-----
%A kompenzáció alkatrészeértékeinek meghatározása:
Rc1 = (2/gm);
prompt = sprintf('Rc1 minimális értéke = %0.3f kOhm, a választott érték [kOhm] = ',Rc1/1000);
Rc1 = input(prompt)*1000;
Cc1 = 1/(2*pi*Fz1*Rc1);
prompt = sprintf('Cc1 számított értéke = %0.3f nF, a választott érték [nF]= ',Cc1*10^9);
Cc1 = input(prompt)*10^-9;
Cc2 = 1/(2*pi*Fp3*Rc1);
prompt = sprintf('Cc2 számított értéke = %0.3f pF, a választott érték [pF]= ',Cc2*10^12);
Cc2 = input(prompt)*10^-12;
Cf1 = ((2*pi*Fo*L*Cki)/Rc1)*K*10^9;
%Cf1=1/(2*pi*R1*Fz2)*10^9;
prompt = sprintf('Cf1 számított értéke = %0.3f nF, a választott érték [nF]= ',Cf1);
Cf1 = input(prompt)*10^-9;
Rf1 = 1/(2*pi*Cf1*Fp2);
prompt = sprintf('Rf1 számított értéke = %0.3f Ohm, a választott érték [Ohm] = ',Rf1);
Rf1 = input(prompt);
R2 = Uref/(Uki-Uref)*R1/1000;

```

```

prompt = sprintf('R2 számított értéke = %0.3f kOhm, a választott érték [kOhm]= ',R2);
R2v = input(prompt)*1000;
disp('A valós komponensek alapján számított frekvenciák: ')
%A valós komponensek alapján számított frekvenciák:
Fz1v=1/(2*pi*Rc1*Cc1);
Fz2v=1/(2*pi*(R1+Rf1)*Cf1);
Fp2v=1/(2*pi*Rf1*Cf1);
Fp3v=1/(2*pi*Rc1*(Cc1*Cc2)/(Cc1+Cc2));

str = sprintf('Fz1= %0.3f kHz',Fz1v/1000);
disp(str)
str = sprintf('Fz2= %0.3f kHz',Fz2v/1000);
disp(str)
str = sprintf('Fp2= %0.3f kHz',Fp2v/1000);
disp(str)
str = sprintf('Fp3= %0.3f kHz',Fp3v/1000);
disp(str)

%a kompenzáció átviteli függvénye:
sprintf('A kompenzáció valós átviteli függvénye:');
Gc=(1/(s*R1*(Cc1+Cc2)))*(((1+s/(2*pi*Fz1v))*(1+s/(2*pi*Fz2v)))/((1+s/(2*pi*Fp2v))*(1+s/(2
*pi*Fp3v))))

disp('A felnyitott kör eredő átviteli függvénye a valós értékek alapján: ')
disp('G(s) = Gp(s)*Gc(s)*Gm(s)')
%A felnyitott kör átviteli függvénye:
G = Gp*Gc*Gm;
%Átviteli függvények kirajzolása bode grafikonra
bode(G,Gp,Gc,opts)
%fázis tartományban és vágási frekvencia Hz-ben:
[tmp,pm,tmp,Wpm] = margin(G);

```

```
Fc = Wpm/(2*pi);  
str = sprintf('pm= %0.2f°',pm);  
disp(str)  
str = sprintf('Fc= %0.0f Hz',Fc);  
disp(str)  
str = sprintf('A kimeneti feszültség a választott értékek alapján:\n');  
disp(str)  
str = sprintf('Vki = %0.3fV',Uref/R2v*(R1+R2v));  
disp(str)
```

3.2. Központi egységek lábkiosztása

3.2.1. A processzor lábkiosztása [22]

PIN	Elsődleges funkciók	Map-elhető funkciók	AndOr funkció	PIN define nevek	PORT define nevek
PA0	ADC123_IN0/TIM2_CH1_ETR/TIM5_CH1/TIM8_ETR/WKUP/USART2_CTS		FPGA - ARM	AO_INTERCON_USART_CTS / AO_INTERCON_SPI_CS	AO_INTERCON_PORT
PA1	ADC123_IN1/TIM5_CH2/USART2_RTS/TIM2_CH2			AO_INTERCON_USART_RTS	
PA2	ADC123_IN2/TIM5_CH3/USART2_TX/TIM2_CH3			AO_INTERCON_USART_TX	
PA3	ADC123_IN3/TIM5_CH4/USART2_RX/TIM2_CH4			AO_INTERCON_USART_RX	
PA4	DAC_OUT1/ADC12_IN4/SPI1_NSS/USART2_CK		FPGA Config Flash	AO_CONF_FLASH_CS	AO_CONF_FLASH_PORT
PA5	DAC_OUT2/ADC12_IN5/SPI1_SCK			AO_CONF_FLASH_SCK / AO_INTERCON_SPI_SCK	
PA6	TIM8_BKIN/ADC12_IN6/SPI1_MISO/TIM3_CH1	TIM1_BKIN		AO_CONF_FLASH_MISO / AO_INTERCON_SPI_MISO	
PA7	TIM8_CH1N/ADC12_IN7/SPI1_MOSI/TIM3_CH2	TIM1_CH1N		AO_CONF_FLASH_MOSI / AO_INTERCON_SPI_MOSI	
PA9	USART1_TX/TIM1_CH2		RS485 RX-TX	AO_RS485_TX	AO_RS485_PORT
PA10	USART1_RX/TIM1_CH3			AO_RS485_RX	
PA11	USB_DN/USART1_CTS/CAN_RX/TIM1_CH4		Nativ USB	AO_USB_DN	AO_USB_PORT
PA12	USB_DP/USART1_RTS/CAN_TX/TIM1_ETR			AO_USB_DP	
PC13	Tamper_RTC		RS485 Data Enable	AO_RS485_DE	AO_RS485_PORT
PC14	OSC32_IN		RS485 Receive Enable	AO_RS485_RE	
PB8	TIM4_CH3/SDIO_D4	I2C1_SCL/CAN_RX	CAN RX-TX	AO_CAN_RX	AO_CAN_PORT
PB9	TIM4_CH4/SDIO_D5	I2C1_SDA/CAN_TX		AO_CAN_TX	
PB10	I2C2_SCL/USART3_TX	TIM2_CH3	Csatlakozó sor kivezetések	AO_HEADER_I2C_SCL / AO_HEADER_USART_TX	AO_HEADER_PORT
PB11	I2C2_SDA/USART3_RX	TIM2_CH4		AO_HEADER_I2C_SDA / AO_HEADER_USART_RX	
PB12	SPI2_NSS/I2S2_WS/I2C2_SMBA/USART3_CK/TIM1_BKIN			AO_HEADER_SPI_CS / AO_HEADER_I2C_SMBA / AO_HEADER_USART_CK	
PB13	SPI2_SCK/I2C2_CK/TIM1_CH1N/USART3_CTS			AO_HEADER_SPI_SCK / AO_HEADER_USART_CTS / AO_HEADER_I2C_CK	
PB14	SPI2_MISO/TIM1_CH2N/USART3_RTS			AO_HEADER_SPI_MISO / AO_HEADER_USART_CTS	
PB15	SPI2_MOSI/I2S2_SD/TIM1_CH3N			AO_HEADER_SPI_MOSI / AO_HEADER_I2C_SD	
PB4	SPI3_MISO	PB4 / TIM3_CH1/SPI1_MISO	User ledek	AO_LED_0	AO_LED_PORT
PB5	I2C1_SMBA/SPI3_MOSI/I2S3_SD	TIM3_CH2 / SPI1_MOSI		AO_LED_1	
PB6	I2C1_SCL/TIM4_CH1	USART1_TX		AO_LED_2	
PB7	I2C1_SDA/FSMC_NADV/TIM4_CH2	USART1_RX		AO_LED_3	

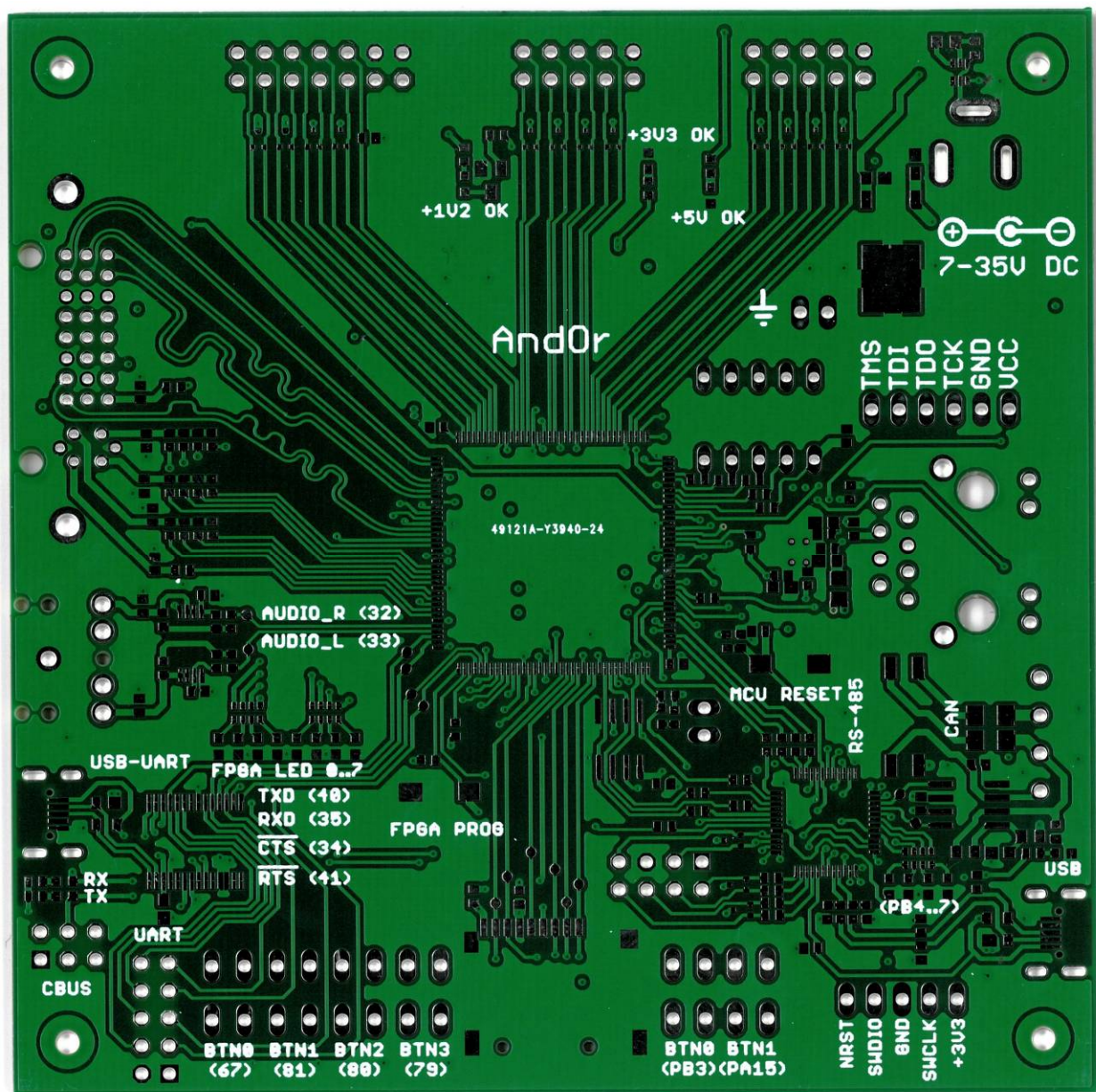
PIN	Elsődleges funkciók	Mappolható funkciók	AndOr funkció	PIN define nevek	PORT define nevek
PB3	SPI3_SCK / I2S3_CK	PB3/TRACES WO/TIM2_C H2/SPI1_SCK	USER gombok	AO_BTN_0	AO_BTN_PORT0 / AO_BTN_PORT1
PA15	SPI3_NSS/I2S3_WS	TIM2_CH1_E TR/PA15 / SPI1_NSS		AO_BTN_1	

3.2.2. Az FPGA lábkiosztása [23]

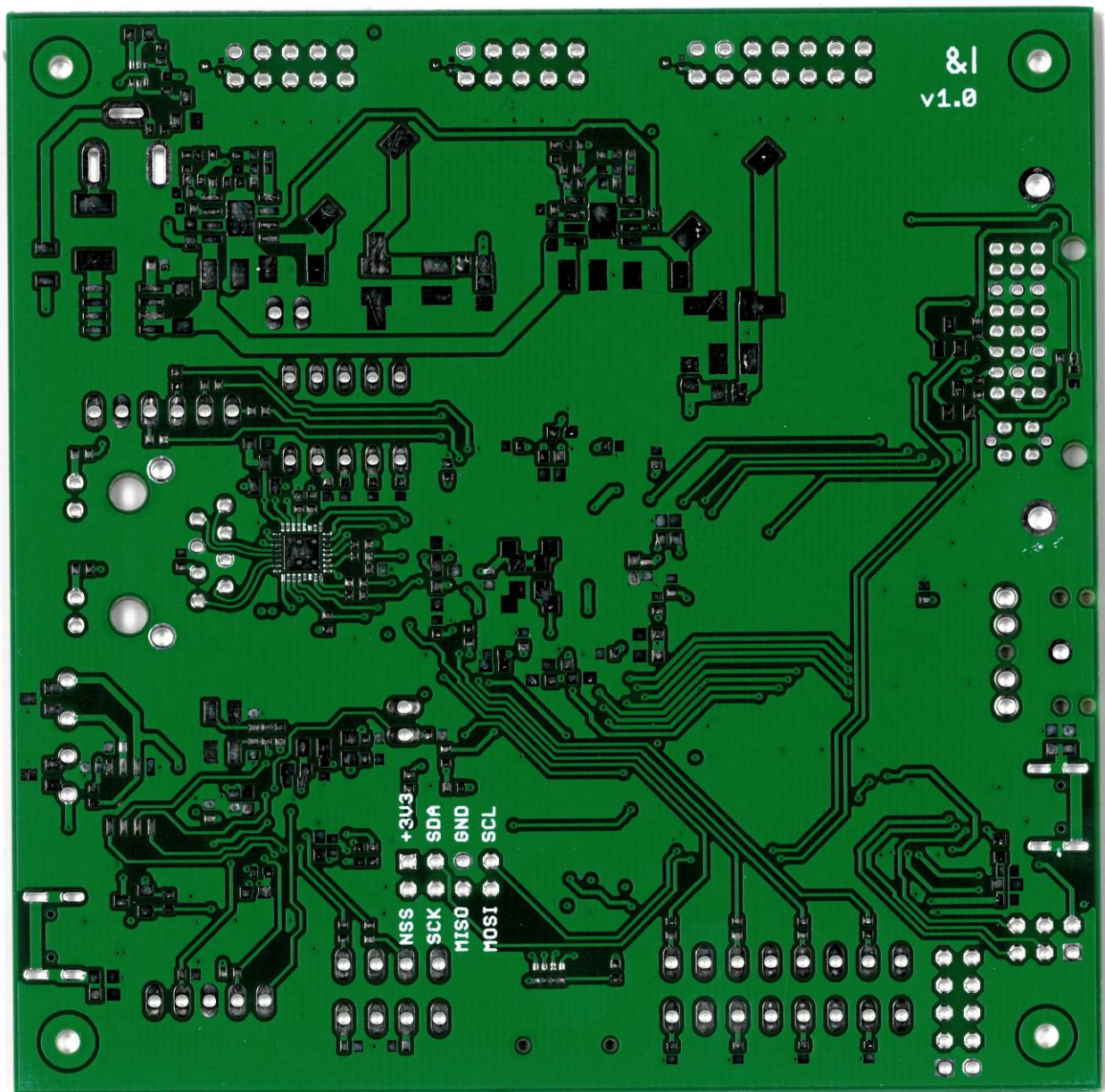
PIN	Pin név	UCF file	AndOr Funkció
100	IO_L1P_1	AO_DIP_SW<0>	DIP kapcsoló
101	IO_L1N_VREF_1	AO_DIP_SW<1>	
102	IO_L32P_1	AO_DIP_SW<2>	
104	IO_L32N_1	AO_DIP_SW<3>	
105	IO_L32P_1	AO_DIP_SW<4>	
78	IO_L47N_1	AO_INTERCON_GPIO<0>	FPGA - ARM
75	IO_L47P_AWAKE_1	AO_INTERCON_GPIO<1>	
74	IO_L74N_DOUT_BUSY_1	AO_INTERCON_GPIO<2>	
72	CMPCS_B_2	AO_INTERCON_GPIO<3>	
67	IO_L2P_CMPCLK_2	AO_BTN<0>	
81	IO_L46P_1	AO_BTN<1>	User gombok
80	IO_L46N_1	AO_BTN<2>	
79	IO_L47P_1	AO_BTN<3>	
99	IO_L33N_1	AO_DVI_HOT_PLUG_DETECT	
1	IO_L83N_VREF_3	AO_DVI_TMDS0_NEG	DVI
2	IO_L83P_3	AO_DVI_TMDS0_POS	
5	IO_L52N_3	AO_DVI_TMDS1_NEG	
6	IO_L52P_3	AO_DVI_TMDS1_POS	
7	IO_L51N_3	AO_DVI_TMDS2_NEG	
8	IO_L51P_3	AO_DVI_TMDS2_POS	
14	IO_L44N_GCLK20_3	AO_DVI_TMDS_CLK_NEG	
15	IO_L44P_GCLK21_3	AO_DVI_TMDS_CLK_POS	
16	IO_L43N_GCLK22_IRDY2_3	AO_DVI_VSYNC	
30	IO_L36P_3	AO_DVI_HSYNC	
9	IO_L50N_3	AO_DVI_VGA_RED<3>	
10	IO_L50P_3	AO_DVI_VGA_RED<2>	
11	IO_L49N_3	AO_DVI_VGA_RED<1>	
12	IO_L49P_3	AO_DVI_VGA_RED<0>	
17	IO_L43P_GCLK23_3	AO_DVI_VGA_GREEN<3>	
21	IO_L42N_GCLK24_3	AO_DVI_VGA_GREEN<2>	
22	IO_L42P_GCLK25_TRDY2_3	AO_DVI_VGA_GREEN<1>	
23	IO_L41N_GCLK26_3	AO_DVI_VGA_GREEN<0>	
24	IO_L41P_GCLK27_3	AO_DVI_VGA_BLUE<3>	
26	IO_L37N_3	AO_DVI_VGA_BLUE<2>	
27	IO_L37P_3	AO_DVI_VGA_BLUE<1>	
29	IO_L36N_3	AO_DVI_VGA_BLUE<0>	
32	IO_L2N_3	AO_AUDIO_RIGHT	Audio /Analog kimenet
33	IO_L2P_3	AO_AUDIO_LEFT	
34	IO_L1N_VREF_3	AO_HEADER_UART_CTS	UART header
35	IO_L1P_3	AO_HEADER_UART_RX	
40	IO_L64N_D9_2	AO_HEADER_UART_TX	
41	IO_L64P_D8_2	AO_HEADER_UART_RTS	
43	IO_L62N_D6_2	AO_LED<0>	User ledek
44	IO_L62P_D5_2	AO_LED<1>	
45	IO_L49N_D4_2	AO_LED<2>	
46	IO_L49P_D3_2	AO_LED<3>	
47	IO_L48N_RDWR_B_VREF_2	AO_LED<4>	
48	IO_L48P_D7_2	AO_LED<5>	
50	IO_L31N_GCLK30_D15_2	AO_LED<6>	

PIN	Pin név	UCF file	AndOr Funkció
51	IO L31P_GCLK31_D14_2	AO_LED<7>	Used leddek
66	IO_L2N_CMPMOSI_2	AO_SD_CARD_DETECT	SD kártya
61	IO_L12N_D2_MISO3_2	AO_SD_DATA<3>	
62	IO_L12P_D1_MISO2_2	AO_SD_DATA<2>	
56	IO_L30P_GCLK1_D13_2	AO_SD_DATA<1>	
57	IO_L14N_D12_2	AO_SD_DATA<0>	
59	IO_L13N_D10_2	AO_SD_CMD	
58	IO_L14P_D11_2	AO_SD_CLK	
95	IO_L40P_GCLK11_1	AO_ETH_TX<0>	Ethernet
97	IO_L34N_1	AO_ETH_TX<1>	
87	IO_L42N_GCLK6_TRDY1_1	AO_ETH_RX<0>	
85	IO_L43P_GCLK5_1	AO_ETH_RX<1>	
98	IO_L34P_1	AO_ETH_PHY_RESET	
94	IO_L40N_GCLK10_1	AO_ETH_TX_ENABLE	
93	IO_L41P_GCLK9_IRDY1_1	AO_ETH_INT_OUT	
92	IO_L41N_GCLK8_1	AO_ETH_RX_ERROR	Ethernet
88	IO_L42P_GCLK7_1	AO_ETH_CRSDV	
84	IO_L43N_GCLK4_1	AO_ETH_CLK_IN	
83	IO_L45P_1	AO_ETH_MDC	
82	IO_L45N_1	AO_ETH_MDIO	FPGA - ARM SPI
64	IO_L3N_MOSI_CSI_B_MISO_2	AO_INTERCON_SPI_MISO	
65	IO_L3P_D0_DIN_MISO_MISO1_2	AO_INTERCON_SPI_MOSI	
70	IO_L1P_CCLK_2	AO_INTERCON_SPI_SCK	
78	IO_L47N_1	AO_INTERCON_SPI_CS	FPGA header
143	IO_L1N_VREF_0	AO_HEADER_USER_PIN_0	
142	IO_L2P_0	AO_HEADER_USER_PIN_1	
141	IO_L2N_0	AO_HEADER_USER_PIN_2	
140	IO_L3P_0	AO_HEADER_USER_PIN_3	
139	IO_L3N_0	AO_HEADER_USER_PIN_4	
138	IO_L4P_0	AO_HEADER_USER_PIN_5	FPGA header
137	IO_L4N_0	AO_HEADER_USER_PIN_6	
134	IO_L34P_GCLK19_0	AO_HEADER_USER_PIN_7	
133	IO_L34N_GCLK18_0	AO_HEADER_USER_PIN_8	
132	IO_L35P_GCLK17_0	AO_HEADER_USER_PIN_9	
131	IO_L35N_GCLK16_0	AO_HEADER_USER_PIN_10	
127	IO_L36P_GCLK15_0	AO_HEADER_USER_PIN_11	
126	IO_L36N_GCLK14_0	AO_HEADER_USER_PIN_12	
124	IO_L37P_GCLK13_0	AO_HEADER_USER_PIN_13	
123	IO_L37N_GCLK12_0	AO_HEADER_USER_PIN_14	
121	IO_L62P_0	AO_HEADER_USER_PIN_15	
120	IO_L62N_VREF_0	AO_HEADER_USER_PIN_16	
119	IO_L63P_SCP7_0	AO_HEADER_USER_PIN_17	
118	IO_L63N_SCP6_0	AO_HEADER_USER_PIN_18	
117	IO_L64P_SCP5_0	AO_HEADER_USER_PIN_19	
116	IO_L64N_SCP4_0	AO_HEADER_USER_PIN_20	
115	IO_L65P_SCP3_0	AO_HEADER_USER_PIN_21	
114	IO_L65N_SCP2_0	AO_HEADER_USER_PIN_22	
112	IO_L66P_SCP1_0	AO_HEADER_USER_PIN_23	
111	IO_L66_SCP0_0	AO_HEADER_USER_PIN_24	

3.3. Az első verzió elkészült NYHL-e



3-1. ábra Felső oldal



3-2. ábra Alsó oldal

4. Felhasznált irodalom

- [1] Falin, Jeff ESR, Stability, and the LDO Regulator. Texas Instruments May 2002, SLVA115
- [2] Valsaraju, Hridya és Vijayakumar, Gopalakrishnan, INTEGRATING HIGH SPEED USB FUNCTIONALITY FOR FPGA- AND ASIC-BASED DESIGNS, Cypress Semiconductor Corp. December 2011
- [3] Xilinx Inc. Spartan-6 FPGA Power Management, May 16, 2014, UG394 (v1.2) p. 32
- [4] NXP Semiconductors SD(HC)-memory card and MMC interface conditioning. 2013, AN10911 p. 13
- [5] Zhang, Henry J. PCB Layout Considerations for Non-Isolated Switching Power Supplies. Linear Technology, June 2012, AN136
- [6] Texas Instruments, Layout Tips for EMI Reduction in DC / DC Converters, April 2012- Revised April 2013, AN-2155
- [7] Xilinx Inc. Spartan-6 FPGA PCB Design and Pin Planning Guide, October 17, 2012 UG393 (v1.3)
- [8] IPC Controlled Impedance Task Group, IPC-2141A, Design Guide for High-Speed Controlled Impedance Circuit Boards, March 2004
- [9] IPC Controlled Impedance Task Group, IPC-2141A Errata List
- [10] E. Hammerstad és Ø. Jensen “Accurate models for microstrip computer-aided design,” Microwave Symposium Digest, IEEE MTT-S International 1980, p. 407–409
- [11] Dr. Zombory Laszlo, Elektromágneses terek, Műszaki Kiadó, 2008
- [12] STMicroelectronic, A5973D Up to 2 A step-down switching regulator for automotive applications, August 2014
- [13] Rahimi, A. M., Parto, P., Asadi, P. Compensator Design Procedure for Buck Converter with Voltage-Mode Error-Amplifier, International Rectifier, AN-1162
- [14] Qiao, M., Parto, P., Amiram R. Stabilize the Buck Converter with Transconductance Amplifier, International Rectifier, AN-1043
- [15] Skyworks Solutions, Inc. Design Type III Compensation Network For Voltage Mode Step-down Converters, September 21, 2012
- [16] Xilinx Inc. DC and Switching Characteristics (v3.1), June 27, 2014, DS162
- [17] Xilinx Inc. Spartan-6 FPGA Configuration (v2.6), June 20, 2014, UG380

- [18] Jocelyn C. Connecting Spansion[®] SPI Serial Flash to Configure Xilinx[®] FPGAs, March 31, 2011
- [19] Seagate Technology, SCSI Commands Reference Manual
<http://www.seagate.com/staticfiles/support/disc/manuals/scsi/100293068a.pdf>
- [20] USB Implementers Forum, Universal Serial Bus Mass Storage Class, Bulk-Only Transport
http://www.usb.org/developers/docs/devclass_docs/usbmassbulk_10.pdf
- [21] Prof. Chidanandan, A. An overview of FAT12
<http://www.eit.lth.se/fileadmin/eit/courses/eitn50/Projekt1/FAT12Description.pdf>
- [22] STM32F103xC, STM32F103xD, STM32F103xE Datasheet,-production data, CD00191185,
<http://www.st.com/web/en/resource/technical/document/datasheet/CD00191185.pdf>
- [23] Xilinx Inc. Spartan-6 FPGA Packaging and Pinouts (v2.3), May 12, 2014, UG385