Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Mini naperőmű FPGA alapú valós idejű Hardware-In-the-Loop szimulátorának tervezése

TDK dolgozat

Debreceni Tibor

Konzulens:

Kökényesi Tamás
Kokenyesi.Tamas@aut.bme.com
Automatizálási és Alkalmazott Informatikai Tanszék

2013.10.25. Budapest

Köszönetnyilvánítás

Ezúton is köszönetet mondok konzulensemnek, Kökényesi Tamásnak, a tervezési és a kivitelezési folyamat során fellépő nehézségeken való túljutásban nyújtott segítségéért.

Külön köszönetet szeretnék mondani Dr. Varjasi Istvánnak, aki építő jellegű kritikáival, szakmai tanácsaival és tapasztalataival nagyban hozzájárult a munkám elkészüléséhez.

A munka kapcsolódik az "Elektromos autó hajtás inverter és energia ellátó rendszer optimalizálási technológia létrehozása" (KMR_12-1-2012-0188), a Siemens Zrt. és a Budapesti Műszaki és Gazdaságtudományi Egyetem közös kutatás-fejlesztési projektjéhez, mely a Magyar Kormány támogatásával, a Nemzeti Fejlesztési Ügynökség kezelésében, a Kutatási és Technológiai Innovációs Alap finanszírozásával valósul meg.

Kivonat

A modern teljesítmény-átalakítók vezérlésének fejlesztése manapság rendkívül költséges és időigényes feladat. A piac ezzel ellentétben gyors válaszokat vár az igényekre, természetesen a minőség romlása nélkül.

Egy ilyen átalakító két fő részre osztható: egy teljesítményfokozatra (főkörre) és egy digitális vezérlő egységre, amit általában valamilyen DSP segítségével valósítanak meg. A korszerű teljesítmény-konverziós feladatokhoz rendkívül bonyolult, többrétű szabályozás és vezérlés szükséges, melyek terepi – azaz magán a nagy teljesítményű főkörön történő– tesztelése nem csak a rendszerre, hanem a tesztelőkre is veszélyes lehet.

A főáramkör kisteljesítményű deszkamodellje felépíthető laboratóriumi körülmények között is, de a végső rendszertől eltérő paraméterekkel. Általában nem lehetséges ugyanazokat az időállandókat beállítani, mint az éles rendszerben és a relatív veszteségek is nagyobbak.

A Hardware-In-the-Loop (HIL) szimulátor jelen esetben az irányított teljesítmény-átalakítónak, annak megtáplálásának és terhelésének FPGA-n implementált modellje. A HIL szimuláció különösen célszerű eszköz a teljesítményelektronikában, hiszen paraméterezhető, monitorozható, logikai jelszintű, így gyors és nem veszélyes egy új vezérlő tesztelése. Az extrém hibaállapotok is könnyen reprodukálhatók, amik a valós rendszerben igen ritkák lennének, esetleg előidézésük sem lenne lehetséges a tesztelés során.

A számítógépes szimulációval szemben további előnye a módszernek, hogy a HIL szimulátor testre szabásával elérhető, hogy a vezérlőkártya ki- és bemeneti jelszintjei azonosak legyenek a vezérlés és a főkör közötti csatlakozási felület (vezérlések, érzékelt jelek) jelszintjeivel, így az irányító elektronika hardver és szoftver tesztelése is megvalósul.

A HIL szimulátor kifejlesztésénél amellett, hogy jó ár/minőség arányra kell törekedni, fontos szempont, hogy a főkör modelljének FPGA-ra implementálható HDL kódját magas szintű programnyelven (Matlab/Simulink) lehessen definiálni, így a szimulátor fejlesztésénél ne kelljen túlságosan a kód optimalizálásán dolgozni.

TDK dolgozatom témája egy ilyen valós idejű HIL szimulátor kifejlesztése FPGA felhasználásával. A modellezendő rendszer egy energiatárolással egybeépített mini naperőmű, azaz napelem panelek által táplált akkumulátortöltő teljesítményfokozat és az akkumulátor.

A vezérlő egység szerepét egy DSP látja el, ami méri a külvilágból érkező analóg jeleket és kiadja a megfelelő vezérlést a konverternek. Az analóg kimenő jelek előállítását az FPGA-ban Σ/Δ DA átalakítók segítik. A szimuláció az áramkör állapotváltozóinak periodikus számításával történik. A megcélzott frissítési frekvencia 40 MHz, ami 25 ns-os időléptékű szimulációs felbontást eredményez.

Abstract

The development of the control unit for modern power converters is nowadays a very cost- and time-critical task. In contrast to this, quick answers are expected for the claims by the market, certainly without quality loss.

A general power converter consists of two main parts: a power level (main circuit) and a digital controller unit, which is usually realized by using some kind of DSP. Remarkably complex and multiple controls are required for the state-of-the-art power conversion tasks, and the field test (test on the real main circuit) of these control units can be dangerous not just for the system itself, but for the testers.

A low-power model of the main circuit can be built under laboratory conditions, but it will have parameters differing from the ones of the original system. Generally it is not possible to set the same time constants as in the real system, and the relative losses are also higher.

The Hardware-In-the-Loop (HIL) simulator is now the model of the power converter with its supply and its load implemented on an FPGA. The HIL simulation is a very useful and practical tool in the power electronics, the simulator can be parameterized, monitored, it works on logic level, and thus the test of a newly designed control unit can be done quickly and without any danger. The extreme failure cases are also reproducible, which would happen very rarely in the real system.

The further benefit of this test method in contrast to the computer-based simulation method is that it can be attainable to have the same signal levels of the IOs of the control unit board as the signal levels of the interface (control and measured signals) between the control and the main circuit, thus the hardware- and the software test of the control electronics can be fulfilled.

In the development of the HIL simulator we have to try to get a proper ratio of the cost/quality, and for that the FPGA-implementable model of the main circuit can be defined on high level program language (Matlab/Simulink), so that less work remains with the code optimization during the development.

The subject of my TDK paper is the development of such a real-time HIL simulator using an FPGA. The system to be modeled is a mini solar power station with energy storage, which means that it is a solar panels-fed battery charger power stage with the battery itself.

The control unit is a DSP-based control board, which measures the analog signals come from the outside of that, and outputs the proper control to the converter. In the FPGA the analog signal producing is supported by Σ/Δ DA converters. The simulation is running with the periodic calculation of the state variables of the system. The aimed refreshing frequency is 40MHz, which means 25ns time step resolution for the simulation.

Tartalomjegyzék

1 Bevezető

1.1 Problémakörnyezet

A teljesítményelektronikai berendezések fejlesztése alapvetően a két fő alkotó egységük fejlesztéséből tevődik össze: az analóg teljesítményfokozaté, és a digitális vezérlőegységé. Két - a fejlesztési munkafázisok egymásra épüléséből adódó - problémát világítanék meg, melyek megoldása több szempontból is kiemelkedően fontos, sőt manapság már követelményként is előfordul a teljesítményelektronikát érintő ipar egyes részein.

Egy teljes rendszer kifejlesztésekor először a teljesítményfokozat specifikációja történik meg, melynek a tervezése szorosan összekapcsolódik a vezérlőegység tervezésével, kiemelten azon részek esetén, melyek az irányítandó mennyiségek méréséért és illesztésért felelősek, illetve melyek a vezérlő jelek fogadásáért és illesztéséért felelősek.

Általánosan mondhatjuk, hogy a digitális vezérlőegységre a tervezéstől a legyártott, kész modulig fordítandó idő és erőforrás nagyságrenddel kevesebb, mint a teljesítményfokozatnál. Ennek oka, hogy a vezérlő egység olyan elektronikai elemeket (DSP, DSC, MCU, meghajtó IC-k, csatlakozók stb.) tartalmaz, melyek ma már egyszerűen hozzáférhető és szerelhető, beültethető elemek. Ellenben a teljesítményfokozatokba tervezett transzformátorok, fojtótekercsek, EMC és harmonikus szűrők, mérőelemek, stb. sokszor egyedi megrendelés alapján lesznek legyártva, de ha nem, akkor is nehezebben hozzáférhetők, beültethetők. Alapvetően érezhető a két egység gyártási folyamata közötti különbség a nehézségeket tekintve, nem is beszélve még a bemérési és az élesztési munkákról.

Tehát az egymásra épülő munkafázisokból eredő egyik fő probléma a teljesítményelektronikai berendezések területén, hogy a fejlesztők számára jóval hamarabb rendelkezésre áll a vezérlő, mint maga az irányítandó egység. Ezért várni kellene addig, míg kész nem lesz a főköri modul, márpedig teendő bőven lenne a vezérlőegységgel addig is.

A másik fő probléma a következő: tegyük fel, hogy bárhogyan is, de eljutunk oda, hogy a kezünkben van az irányítandó- és az irányító egység is. Ekkor következhetnek az első érdemi tesztelések. Az esetek többségében igaz, hogy a teljesítményfokozatunk egy több kW-os rendszer energiatovábbításáért, elosztásáért vagy átalakításáért felelős.

A vezérlőnk lelkét jelentő központi egységben (pl.: DSP) elég egy rosszul implementált szabályozó, egy elírás a kódban, vagy egy rosszul beforrasztott meghajtó IC (és így tovább) ahhoz, hogy az első bekapcsoláskor a legjobb esetben is csak felrobbanjon valamelyik rész, tipikusan az IGBT hídból egy IGBT, vagy annak meghajtása, illetve a DC vagy AC link kondenzátorok valamelyike, vagy egésze. A rosszabb eset az, amikor tönkremegy a teljes főkör, és újra le kell gyártatni. Sőt a lehető legrosszabb eset, amikor az emberrel, a tesztelőkkel történik baleset, ami ekkora teljesítmények esetén könnyen előfordulhat.

A két problémát közösen kezelve érezhető, hogy szükség van egy a teljesítményfokozatot helyettesítő eszközre, ugyanis ekkor már a vezérlőegység elkészültekor lehetne rajta tesztelni a működését, és mindezt minden robbanást, anyagi- és életveszélyt elkerülve.

1.2 Szimuláció helye a teljesítményelektronikai fejlesztésben

Ma már kézenfekvő az, hogy megfelelő eszköz erre a szimuláció. Alapjában véve a szimulációk két fő fajtát különböztetjük meg, a szoftveres és a hardveres szimulációkat. Később látni fogjuk, hogy ez ennél azért bonyolultabb, mindenesetre mondhatjuk, hogy a legfőbb különbség a kettő között az, hogy a szoftveres egy számítógépen futó szimulációra épül, melynek tulajdonságait legfőképpen az adott számítógép memóriája, CPU-ja, annak számítási sebessége, számítási algoritmusai határozzák meg, és legtöbb esetben a számítógép interfészei a külvilág felé, a hardveres szimuláció esetében pedig nem.

A hardveres szimuláció alapja lényegében egy beágyazott rendszer, melyben a szimuláció tulajdonságait legfőképpen a rendszerbeli központi számító egység, annak számítási sebessége, a memória és a rendszer interfészei határozzák meg. Természetesen mindezeken felül a szimuláció minőségét maga az implementált feldolgozás, és kalkulációs algoritmusok optimalizáltsága határozzák meg (pl. lehet gyors az FPGA, ha a számítási algoritmusaink, vagy a memóriakezelésünk viszont lassú).

Legelőször is a szemléletmódunkat alakítsuk ki úgy, hogy szimulációs rendszerről, mint tesztkörnyezetről beszélünk. Ehhez azt kell alapul vennünk, hogy a tesztelendő rendszer melyik elemét szeretnénk elsősorban teszt alá venni. Nem feltétlenül igaz, hogy mindig a rendszer tesztelendő egységének szimulátorára van szükségünk ugye (pl. pont a HIL szimulátor, lásd később).

A most következő elnevezéseket és szempontrendszert önkényesen alakítottam ki annak érdekében, hogy legalábbis a teljesítményelektronikában, a gyakorlatban is használatos szimuláció típusokat összegyűjtve áttekinthetővé és világossá tegyem ezeket. Ekkor a következő elnevezésekkel élek:

- SWS: szoftver szimulátor
- HWS: hardver szimulátor
- HW: eredeti hardver

Kidolgoztam egy táblázatot, amely nagy segítséget nyújt mindezek összefoglalásában, ezt láthatjuk az 1.2.1 táblázatban.

Sz.	Term.	Tesztrendszer		Rövid leírás	Additív munkák a teszt megkezdéséhez	Értékelés								
						Szimuláció		Vezérlő leteszteltség		Teszt (felépítés) komplexitása	Tesztelés nehéz-sége	Költség	Bővíthetőség skálázhatóság átalakíthatóság	Anyagi kockázat és baleset veszély
		Teljesítmény-fokozat (TF)	Vezérlő egység (VE)			Érték- és idő felbontás	Közelítés valódi rendszerhez	HW	SW					
i	Nem használatosak	HW	HWS											
ii		HW	SWS											
iii		SWS	HWS											
iv		HWS	HWS											
v		HWS	SWS											
1	SW szimuláció	SWS	SWS	***Offline*** teszt	SW szimulációk kifejlesztése	H	L	L	H	H	L	L	H	L
2		SWS	HW	***Számítógép-alapú*** teszt: TF **SW** szimulátora az **eredeti** VE-gel	TF SW szimulációjának fejlesztése, VE HW-ének illesztése PC-hez	M	L	M	H	H	L	L	M	L
4	HW szimuláció	HWS	HW	***HIL szimuláció:*** **TF HW** szimulátora az **eredeti** VE-gel	TF HW szimulátorának fejlesztése	H	H	H	H	H	L	L	H	L
5		HWS	HW	***"Labor" szimuláció:*** TF **kisteljesítményű HW** szimulátora az **eredeti** VE-gel	TF HW szimulátorának fejlesztése	H	M	M	M	H	M	M	M	M
6	Nem szimuláció	HW	HW	**Terepi** teszt	-	Valóság	Valóság	ok	ok	L	H	H	L	H

Jelmagyarázat: H (High): magas; M (Medium): közepes; L (Low): alacsony | SWS: szoftver szimulátor, HWS: hardver szimulátor, HW: eredeti hardver

1.2.1 táblázat: Szimulációk a teljesítményelektronikai tesztelésekben

Mint látható, többféle lehetőség adódik a három fenti elnevezés kombinálásával, azonban nem mindegyiknek van úgymond értelme, ezeket szürkével jeleztem, nem használatosak ezen a területen. Meg kell említenem, hogy manapság az iparban az a szokásos, hogy egy már meglévő, folyamatosan gyártásban lévő teljesítményfokozatot használnak fel, és a fejlesztésben csak a vezérlőegységre fókuszálnak, azon belül is arra törekednek, hogy lehetőleg csak a vezérlő szoftvere módosuljon. Mindez azért van, mert ha a hardvert módosítani kellene, akkor új gyártási terveket kell leadni, újra kell programozni a gyártósor az adott - mellesleg már kipróbált és működő - termékre beállított elemeit, ami igen komoly költségekkel járna.

Tehát alapvetően a táblázatban a vezérlőegység tesztelését veszem fő szempontnak. Mivel a táblázatban minden módszert részletesen, több fejlesztési és üzleti szempont szerint sikerült összehasonlítanom, így ezeket nem részletezném itt.

Nézzük tehát: az első csoport az, amit e területen szoftveres szimulációnak nevezünk. A szoftveres módszerek közül a leggyakrabban használt az offline szimuláció, amikor is mind a főkört a táplálásával és terhelésével, mind a vezérlőegység működését számítógépen szimuláljuk, és „teszteljük” a rendszer működését. Ez a módszer nem biztosít valós idejű szimulációt, ennek ellenére igen gyakori és hatékony módszer a fejlesztés kezdeti, tervezési fázisában [4]. A másik szoftveres módszer lényege, hogy a teljesítményfokozat szoftveres szimulátora fut számítógépen, a vezérlőegység pedig eredeti formájában van jelen. Ugyanilyen elrendezésben nem használatos az, hogy az eredeti főkört használnánk fel az 1.1-ben elmondottak alapján. A teljesítményfokozat számítógépes szimulációs módszer legnagyobb hátránya, hogy bár GHz-es CPU-val segített a szimuláció, mégsem érhető el 50-250us-nál kisebb lépésköz, a számítógép I/O interfészei és a CPU egyes számítási metódusai lassítják [4].

A legtöbbet használt tesztelési metódusok a hardveres szimulációk, melynek két típusa van, a közös bennük, hogy a tesztfelépítésben a főkör hardver szimulátora, a vezérlőegység pedig eredeti formájában van jelen. Az egyik az ún. HIL szimuláció, a másik az ún. labor- vagy deszkamodell alapú szimuláció [3]. Az utóbbival kezdeném: arról van szó, hogy megépítjük a teljesítményfokozat deszkamodelljét, amely kisebb teljesítményű, mint eredeti megfelelője. Ennek azért van értelme, mert elkerüljük a tesztelés során előforduló valós veszélyeket, a legnagyobb hátránya, hogy a veszteségek és időállandók nem egyeznek meg a valódi főkörével,

így ha a vezérlőnk ezen jól is vizsgázott, nem biztos, hogy az eredeti fokozaton is megfelelően fog [3].

A másik, egyre gyakrabban használt tesztelési metódus, a HIL szimuláció lényege, hogy a vezérlőegység itt is eredeti formájában van jelen, de a teljesítményfokozatot a HIL szimulátorával helyettesítjük. Hardveres szimulátorról lévén szó, egy központi egység a lelke, melyen a rendszert leíró állapotegyenletek periodikus megoldása történik. A legfontosabb követelmények a lehető legkisebb időbeli lépésköz elérése, a lehető legpontosabb megoldások az állapotváltozókra, és az alacsony költség. Nagy előnye, hogy bár szimulátorról lévén szó, annak testre szabásával elérhető, hogy a vezérlőkártya ki- és bemeneti jelszintjei azonosak legyenek a vezérlés és a főkör közötti csatlakozási felület (vezérlések, érzékelt jelek) jelszintjeivel, így az irányító elektronika hardver és szoftver tesztelése is megvalósul [1] [2].

2 Elméleti áttekintés

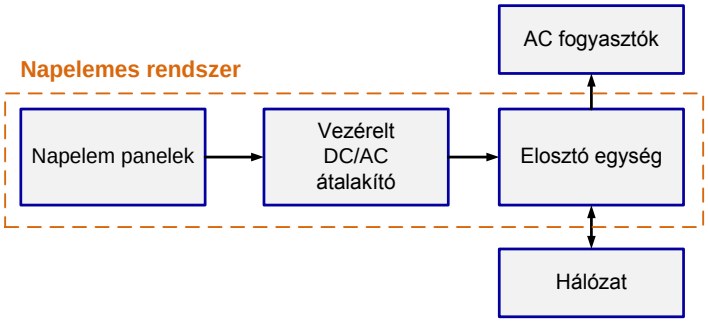
Ebben a fejezetben röviden bemutatom a munkám során felhasznált főbb elméleti ismereteket. A napelem panelek által biztosított energiaforrás szimulációjához azok működésének, karakterisztikáinak és a naperőművi rendszer lehetséges konfigurációinak ismerete szükséges. A szimuláció az áramkör állapotváltozóinak periodikus számításával történik, melyek analóg jellé alakításához Σ/Δ D/A átalakítókat használtam.

2.1 Naperőművi rendszerek

2.1.1 Általános leírás, típusok

A napelemes rendszerek több fajtáját is megkülönböztethetjük aszerint, hogy mik a felhasználási célok és az adott követelmények. Az egyik leglényegesebb különbség közöttük, hogy csatlakoznak-e egyéb energiaellátó rendszerekre, energiatároló egységekre. Ezenkívül megkülönböztethetők az alapján is, hogy DC vagy AC energiát szolgáltatnak [5].

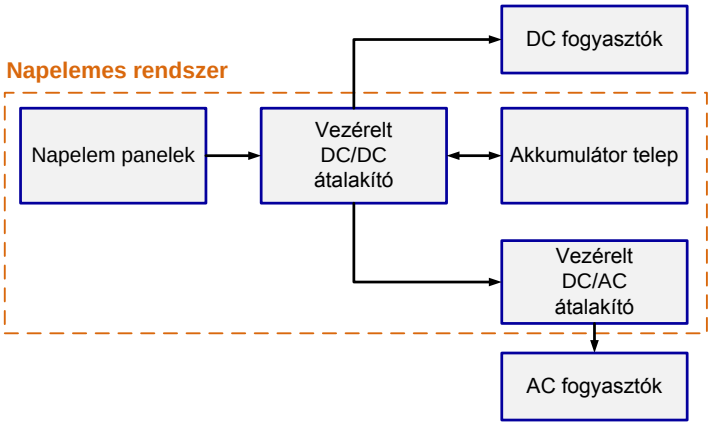
A hálózatra csatolt napelemes egységek az energiaellátó hálózattal párhuzamosan, azzal összeköttetésben működnek. Egyik leglényegesebb főáramköri elemük az inverter, amely a vezérlésének köszönhetően a napelem cellák által szolgáltatott DC energiát a hálózat jelszintjének és minőségi követelményeinek megfelelő AC energiává alakítja át. Amint a napelemes rendszer energiamentes állapotba kerül, az inverternek le kell választania a hálózatról. Az inverter fokozat után többnyire olyan elosztó egység található, amely egyrészt lehetővé teszi a napelemes rendszerre kötött AC fogyasztók közvetlen ellátását, másrészt a rendelkezésre álló többletenergia hálózatra való visszatáplálását, amennyiben a fogyasztók aktuális energiaigénye alacsonyabb a napelemes rendszer által szolgáltatott mennyiségnél. Amikor a napelemekből nyert energia nem elegendő a fogyasztók táplálására (tipikusan éjszaka), a hálózat is rendelkezésre áll a különböző energiaigények kielégítésére [6]. A 2.1.1 ábra egy hálózatra csatolt rendszer blokkvázlata.



2.1.1 ábra: Hálózatra csatolt napelemes rendszer

A fent említett másik fő típusú, az önállóan működő napelemes rendszereket a hálózattól való független működésre tervezik, tipikusan adott DC/AC fogyasztók energiaellátására. Legegyszerűbb esetben a napelem cellák közvetlenül táplálják a DC fogyasztót. Ekkor a rendszerre nem csatlakozik semmilyen energiatároló elem; következésképpen az ilyen rendszerek csak nappali alkalmazásokban működhetnek.

A folyamatos működés biztosítására alkalmazható akkumulátor is a napelemes rendszerben. A 2.1.2 ábra egy AC illetve DC fogyasztókat ellátó egységet ábrázol.



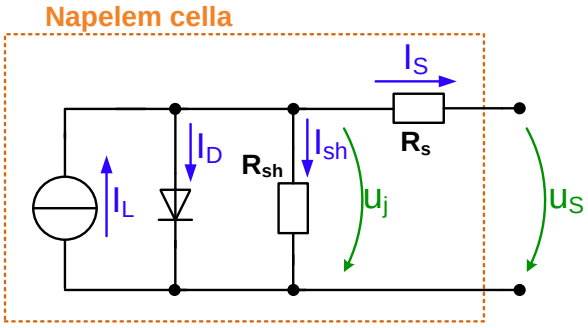
2.1.2 ábra: Önállóan működő, energiatárolással egybeépített napelemes rendszer

2.1.2 Napelem panelek összeállítása

A tervezőnek viszonylagos szabadsága van a panel kialakításakor, hiszen az egyes napelem cellákat lehet egymással sorba és párhuzamosan kötni. A keletkező panel paraméterei egyszerű skálázással számíthatók át az egyes cellák tulajdonságaiból a soros és párhuzamos összeköttetésekre vonatkozó szabályok figyelembevételével feltéve, hogy minden cella tökéletesen egyforma, illetve a hőmérséklet és a besugárzás is azonos az összes cella esetében. Amennyiben ez nem teljesül (tipikusan ha például árnyék vetül néhány cellára), a panel eredő karakterisztikája az egyes cellák görbéinek szuperpozíciójából továbbra is számítható. Bypass diódák az esetlegesen árnyékba kerülő cellák módosult karakterisztikájának negatív hatásaitól védik az áramkört. Nagyobb terhelési áramok eléréséhez párhuzamosan, magasabb feszültségértékekhez pedig sorosan kell kapcsolni a napelem paneleket.

2.1.3 Napelem cella karakterisztikák

A napelem cella DC helyettesítő képéből indulunk ki:



2.1.3 ábra: Napelem cella DC helyettesítő képe

A kimenő áramra vonatkozó egyenlet a következő:

$$I_S = I_L - I_D - I_{sh} \tag{2.1.1}$$

ahol az egyes tagok kifejtve:

$$I_D = I_0 \left(e^{\left(\frac{q \cdot (U_S + I_S \cdot R_S)}{n \cdot k \cdot T}\right)} - 1 \right) \text{ és } I_{sh} = \frac{U_S + I_S \cdot R_S}{R_{sh}} \tag{2.1.2}$$

Ezekkel tehát adódik egy közelítő karakterisztikus egyenlet [5]:

$$I_S = I_L - I_0 \left(e^{\left(\frac{q \cdot (U_S + I_S \cdot R_S)}{n \cdot k \cdot T}\right)} - 1 \right) - \frac{U_S + I_S \cdot R_S}{R_{sh}} \tag{2.1.3}$$

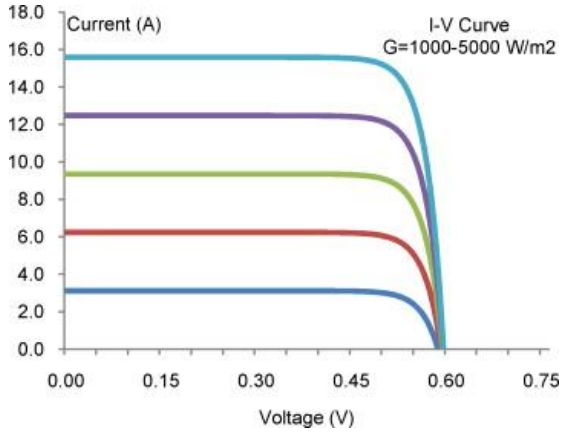
ahol:

- I_S: napelem cella kimeneti áram,
- I_L: besugárzással arányos, fotovoltaikus áram,
- I_0: telítési áram,
- T: napcella hőmérséklet,
- q: elemi töltés, elektron töltése: 1,6E-19 [°C],
- k: Boltzmann-állandó: 1,38E-23 [J/K],
- n: dióda jósági faktor,
- R_S: soros ellenállás,
- R_{sh}: párhuzamos ellenállás.

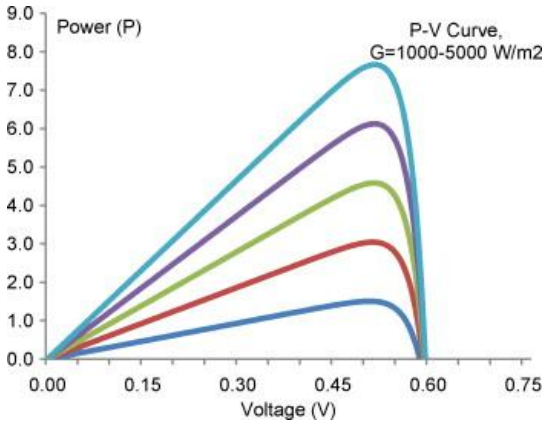
Feltételezve, hogy a napelem cella jó minőségű, tehát az $\mathbf{R_{sh}}$ elég nagy és így az $\mathbf{I_{sh}}$ elhanyagolható, és az $\mathbf{I_0}$ és az $\mathbf{R_S}$ elég kicsi, adódik az $\mathbf{U_{oc}}$ üresjárási feszültség és az $\mathbf{I_{sc}}$ rövidzárási áram [5].

$$U_{oc} \approx \frac{nkT}{q} \cdot \ln\left(\frac{I_L}{I_0} + 1\right)$$
$$I_{sc} \approx I_L \tag{2.1.4}$$

A 2.1.4 ábrán látható egy napcella karakterisztikája:



(a)



(b)

2.1.4 ábra: Napcella áramának (a) és teljesítményének (b) feszültségfüggése különböző megvilágítások mellett

2.2 Szigma-delta digitális-analóg átalakító

2.2.1 Problémafelvetés

A digitális szimuláció eredményeképpen előálló feszültség- és áramértékeket analóg jellé kell alakítani. Az egyik, és legegyszerűbb lehetőség, hogy az FPGA lábain az egyes mennyiségek numerikus értékét párhuzamosan jelenítjük meg és külső D/A átalakítót használunk Ez viszont - a felbontástól függően - sok I/O lábat igényel az FPGA-ban, ezért korlátozottan használható [7].

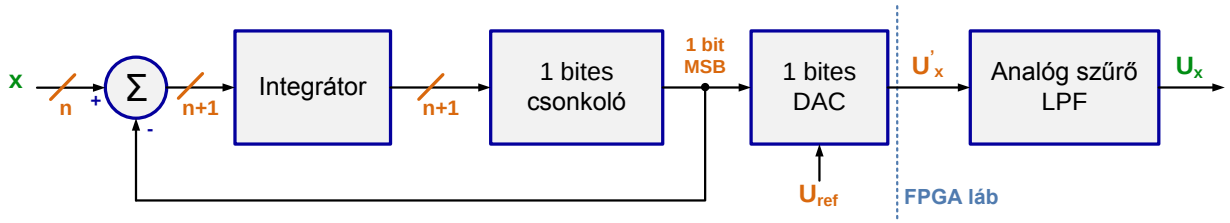
Hasonló esetekben gyakran alkalmazott megoldás az impulzusszélesség-moduláció (PWM) használata. Ebben az esetben egy jel csak egy FPGA-lábat igényel, azonban a kiadható jel sávszélessége erősen lecsökken, mivel a PWM által használt alapfrekvenciát ki kell szűrni a kimenetből. Így például 40 MHz-es órajellel 4000-es felbontást használva a PWM alapperiódusának frekvenciája máris 10 kHz-re csökken, amit ki kell szűrni. A használt szűrő minőségétől függően ettől csak még kevesebb lehet a kiadandó hasznos jel sávszélessége. Ez pedig jelentősen lecsökkenti a szimuláció pontosságát [7].

2.2.2 A Σ/Δ modulátor

A két megoldás előnyeit ötvözi a Σ/Δ modulátor használata. A Σ/Δ átalakító feladata, hogy a kvantálási zajt kitolja a nagyfrekvenciás tartományba, míg kisfrekvencián a hasznos jel érintetlen marad. A hiba-visszacsatolásos modulátor a nevét onnan kapta, hogy az újrakvantáló hibáját vezetjük a visszacsatoló ágba. A Σ/Δ moduláción alapuló D/A átalakítók magas jel-zaj viszonyt (SNR) biztosítanak többféle zajformálási technika felhasználásának segítségével. A digitális részében a valódi DAC-re már a zajformálással együtt kerül a bemeneti jel, így a DAC kvantálási zaja a nagyfrekvenciás tartományba van kitolva. Az analóg részében passzív aluláteresztő szűrők vágják le a kitolt kvantálási zajt, melynek eredményeképpen a szűrő kimenetén szinte nem vesztünk az eredeti, hasznos jel sávszélességéből [8].

Mivel nincsen visszacsatolás az analóg kimenetről a belső spektrum-átalakító szűrőbe, így felépítését tekintve ez egy előrecsatolt megoldás. Igaz D/A átalakítóról beszélünk, ez zavaró és félrevezető lehet, mivel az újrakvantáló hibájának visszacsatolása nem halad át az analóg-digitális részek határán.

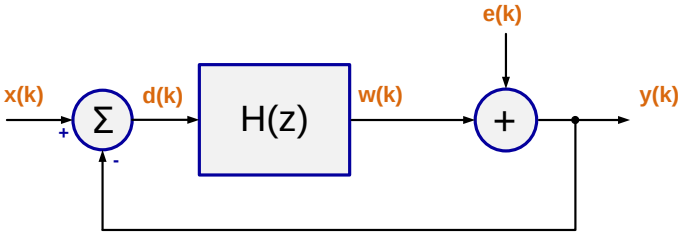
A szigma-delta DAC egy digitális-digitális és egy nagysebességű néhány (a megvalósítottban 1) bites digitális-analóg átalakítóból épül fel. A 2.2.1 ábrán látható a blokkvázlata.



2.2.1 ábra: Σ/Δ modulátor blokkvázlata

Lényegében arról van szó, hogy az **n** biten reprezentált megjelenítendő feszültségértéket egy integrátorba vezetjük. Ennek túlcsordulása esetén a kiadandó bit 1 lesz, különben 0. Túlcsorduláskor az integrátor értékét csökkenteni kell a bemeneti jel k biten lehetséges maximális értékével. Minél nagyobb a bemenő jel értéke, annál gyakrabban lesz túlcsordulás az integrátorban és annál nagyobb lesz a kimeneti feszültség átlagértéke is.

Jelfeldolgozási szempontból vizsgálva a fent leírtakat a következő lineáris modellt rajzolhatjuk fel:



2.2.2 ábra: Σ/Δ modulátor lineáris modellje

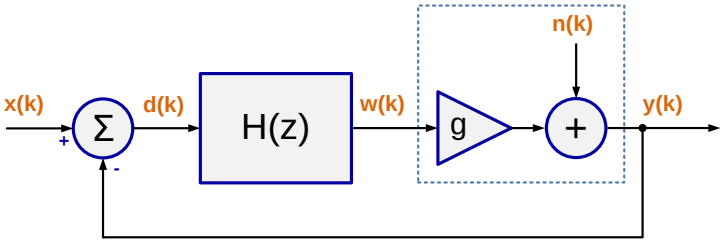
A modulátor kimenetének y(k) és a bemenetének w(k) különbsége adja az e(k) kvantálási hibát. A 2.2.2 ábra alapján:

$$y(k) = w(k) + e(k) = H(z) \cdot [x(k) - y(k)] + e(k) \tag{2.2.1}$$

$$y(k) \cdot [1 + H(z)] = w(k) + e(k) = H(z) \cdot x(k) + e(k) \tag{2.2.2}$$

$$y(k) = \frac{H(z)}{1 + H(z)} \cdot x(k) + \frac{1}{1 + H(z)} \cdot e(k) \tag{2.2.3}$$

A 2.2.3 egyenletből látható, hogy az y(k) kimenő jel az x(k) bemenő jel szűrt változatának és az e(k) kvantálási hiba szűrt változatának összegéből áll. Ha feltételezzük, hogy a kvantálási hiba nem függ a bemenő jeltől, akkor mondhatjuk, hogy a kvantálás egy lineáris g erősítésből és egy additív n(k) zajforrás tagból áll.



2.2.3 ábra: Σ/Δ modulátor lineáris modellje az n(k) hibával

Tehát megtehetjük, hogy a 2.2.3 egyenletben szereplő e(k) hibát lecseréljük n(k) kvantálási hibajelre, és a *g* erősítési tényezőt bevisszük a szűrőbe, ekkor az y(k) kimenet így írható:

$$y(k) = \frac{H(z)}{1 + H(z)} \cdot x(k) + \frac{1}{1 + H(z)} \cdot n(k) \tag{2.2.4}$$

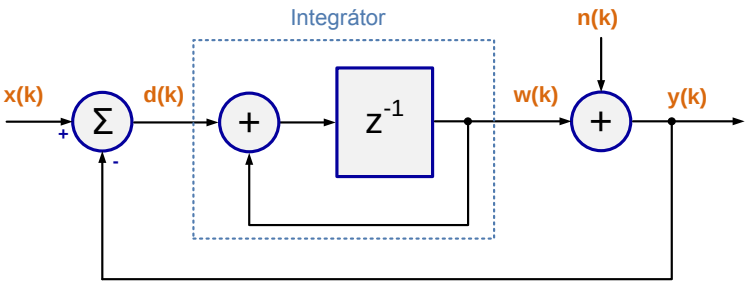
Ebből n(k)=0 esetet véve adódik az átviteli függvény a hasznos jelre, melynek a szokásos jelölése előrecsatolt rendszer specifikusan **S**ignal **T**ransfer **F**unction – **F**eed-**F**orward [8]:

$$STF_{FF} = \frac{y(k)}{x(k)} = \frac{H(z)}{1 + H(z)} \tag{2.2.5}$$

Ha pedig az x(k)=0 esetet vesszük, akkor adódik az átviteli függvény a zajra melynek a szokásos jelölése előrecsatolt rendszer specifikusan **N**oise **T**ransfer **F**unction – **F**eed-**F**orward [8]:

$$NTF_{FF} = \frac{y(k)}{n(k)} = \frac{1}{1 + H(z)} \tag{2.2.6}$$

Jobban konkretizálva a lineáris modellt a 2.2.1 ábra alapján, ha a *H* szűrőt a legegyszerűbb integrátorral helyettesítjük, melynek az előzőek alapján bevitt erősítése g=1, akkor egy elsőrendű modulátor lineáris modelljét kapjuk:



2.2.4 ábra: Elsőrendű Σ/Δ modulátor lineáris modellje

Az integrátor a jel előző ütembeli értékét adja hozzá a bemenethez. Lényegében a csonkolás nem más, mint az x(k) hasznos jelhez az n(k) kvantálási zaj hozzáadása. Az integrátor átviteli függvénye:

$$H(z) = \frac{z^{-1}}{1 - z^{-1}} = \frac{1}{z - 1} \tag{2.2.7}$$

Ezután 2.2.5 és 2.2.6 alapján egyből adódik:

$$STF_{FF} = \frac{y(k)}{x(k)} = \frac{H(z)}{1 + H(z)} = \frac{\frac{1}{z-1}}{1 + \frac{1}{z-1}} = \frac{1}{z + 1 - 1} = \frac{1}{z} \tag{2.2.8}$$

2.2.8 egyenletből látható, hogy a hasznos jel csak egy ütemnyi késleltetést szenved, nem torzul.

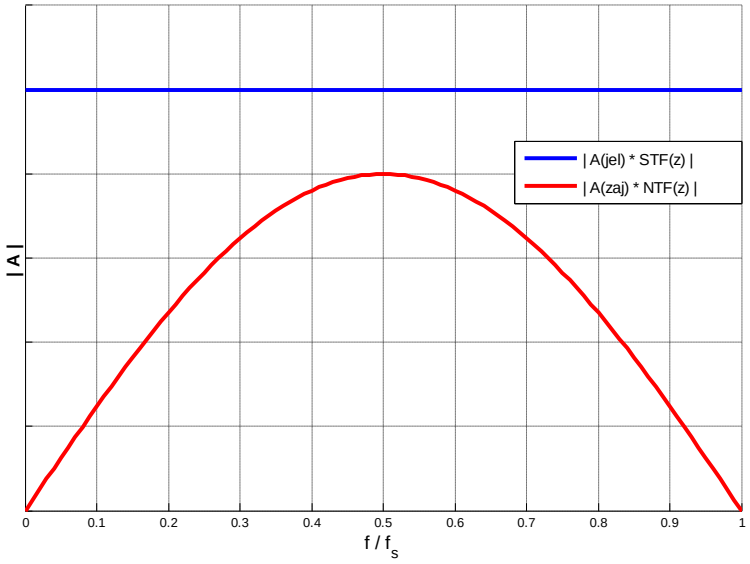
$$NTF_{FF} = \frac{y(k)}{n(k)} = \frac{1}{1 + \frac{1}{z-1}} = \frac{z-1}{z-1+1} = \frac{z-1}{z} = 1 - \frac{1}{z} \tag{2.2.9}$$

2.2.9 egyenletből látható, hogy a zajból mindig levonódik az előző ütembeli értéke. Ennek eredményeképpen annak mindig a konstans vagy a mintavételi frekvencia (f_s) többszöröseinél lévő komponenseit a modulátor elnyomja. Az STF_{FF} és az NTF_{FF} átviteli függvények abszolút értékét ábrázolhatjuk a frekvencia függvényében. A $z = e^{j\omega T}$ és a $T_s = \frac{1}{f_s}$ helyettesítésekkel élve:

$$|STF_{FF}(z)| = |z^{-1}| = \left|e^{-j\omega T_s}\right| = 1 \tag{2.2.10}$$

$$|\mathrm{NTF_{FF}}(z)| = |1 - z^{-1}| = \left|1 - e^{-j\omega T_s}\right| = \left|2j \cdot \frac{e^{\left(\frac{j\omega T_s}{2}\right)} - e^{-\left(\frac{j\omega T_s}{2}\right)}}{2j} \cdot e^{-\left(\frac{j\omega T_s}{2}\right)}\right| = \tag{2.2.11}$$

$$= \left|2j \cdot \sin\left(\frac{\omega T_s}{2}\right) \cdot e^{-\left(\frac{j\omega T_s}{2}\right)}\right| = 2 \cdot \sin\left(\frac{\omega T_s}{2}\right) = 2 \cdot \sin\left(\frac{\pi \cdot f}{f_s}\right)$$



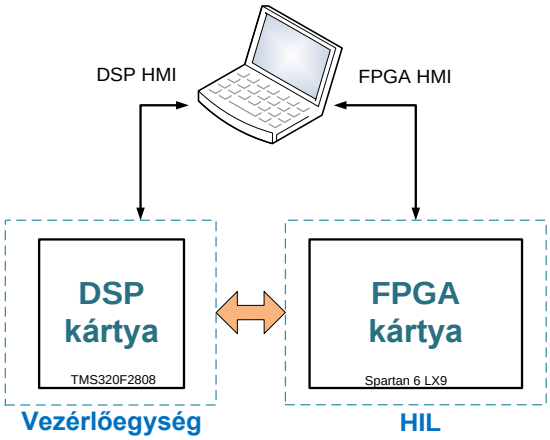
2.2.5 ábra: Elsőrendű Σ/Δ modulátor átvitele

A 2.2.5 ábrán tehát egy elsőrendű szigma-delta D/A átalakító adott átvitelét látjuk, kékkel jelölve a hasznos jel átvitelét, és pirossal jelölve a zaj átvitelét, jogosan feltételezve, hogy (|A(jel)|>>|A(zaj)|) a hasznos jel által felvehető értékek abszolút értéke nagyságrendileg nagyobb, mint a zajé. Ekkor a függőleges tengely (|A|) csak szemléltető, azt mutatja, hogy a modulátor adott átvitele a zajra kisebb, mint a hasznos jelre. Az ábrán jól látszódnak a szigma-delta D/A átalakító fent leírt tulajdonságai. Látható, hogy a kimeneti vonal megfelelő szűrésével a jel alacsony frekvenciájú összetevői kis zajterhelés mellett megtarthatóak. A TDK munkám keretében elkészített modulátort és a hasznos jeltartományt alapul véve a 2.2.5 ábrán az f/f_s tengelyen a 0 és a $\frac{8\ \mathrm{kHz}}{40\ \mathrm{MHz}} = 0.0002$ tartományban mozgunk (lásd még később 3.1.3 fejezetben).

3 A megvalósított HIL szimulátor

3.1 Tesztkörnyezet felépítése

A 3.1.1 ábrán a környezet felépítésének blokkvázlatát láthatjuk, mely a következőkből áll: egy FPGA alapú HIL szimulátor és egy DSP alapú vezérlőegység, melyek egy számítógéphez kapcsolódnak, mely felhasználói interfészeket (HMI) tartalmaz mindkét egység számára.



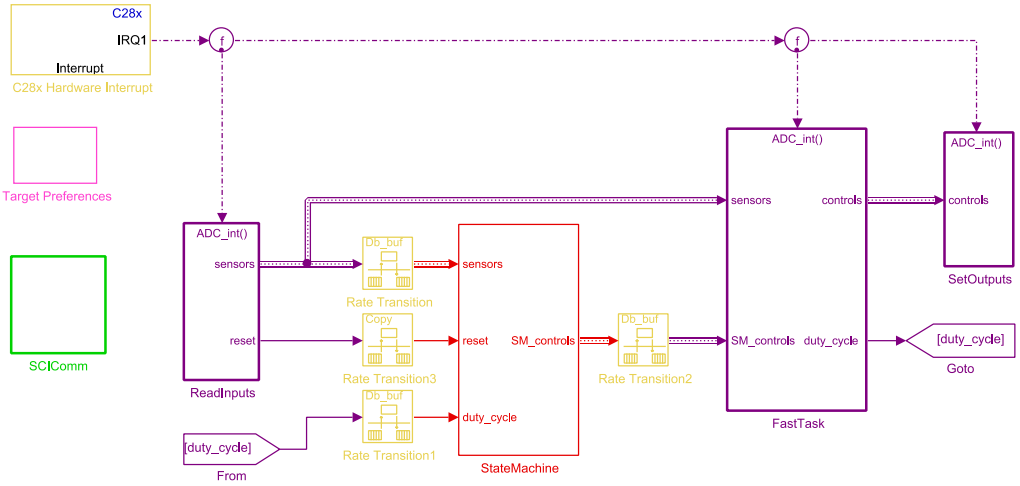
3.1.1 ábra: A tesztkörnyezet felépítése

Ezek által a felhasználó (fejlesztő) módosítani tudja az egyes paramétereket, és monitorozni képes egyes jeleket mindkét egységben. A DSP alapú vezérlő esetén megszokott feladat egy felhasználói interfészt készíteni, de az FPGA alapú HIL szimulátor esetén ez már nem ilyen hasonlóan bevett és megszokott alapfeladat (részletesen lásd később a 3.3.2 fejezetben).

3.1.1 A vezérlőkártya

A vezérlőegység a *Procon Hajtástechnika Kft.* egy DSP kártyája, melyen a központi egység egy 32 bites, fixpontos aritmetikájú, 100 MHz órajelű TMS320F2808–as DSP. Ez az egység tartalmaz egy állapotgépet, a töltőáramkörhöz a szabályozót, előtöltés vezérlést és a maximum teljesítmény pont megkereső algoritmust (MPPT). Mivel a projekt keretében az én feladatom a HIL szimulátor elkészítése volt, ezért a vezérlőegységről csak egy pár szót szólnék.

A vezérlőegység elkészítése Dr. Sütő Zoltán (egyetemi docens az Automatizálási és Alkalmazott Informatikai Tanszék Elektrotechnika csoportjában) kollégám érdeme. Mivel a projektben az egyik legfőbb cél az volt, hogy mindent, amit csak lehet, a Matlab programból generáljunk, ezért, mint a HIL szimulátor esetében, a vezérlőegységnél is ez volt előtérben. A 3.1.2 ábrán a vezérlőegység Simulink modelljének legfelső szintje látható, ennél mélyebben nem is mennék bele a részletekbe.



3.1.2 ábra: A vezérlőegység felső szintű Simulink modellje

3.1.2 Az FPGA fejlesztőkártya

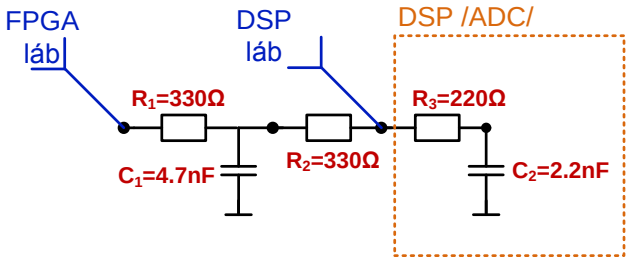
A HIL szimulátort fizikailag egy FPGA fejlesztőkártyára készítettem el, melyen egy Xilinx Spartan 6–os FPGA található. A Spartan-6 LX9 MicroBoard kártyán egy 27 MHz-es kristály oszcillátor kimenete egy hardveres PLL-en keresztül jut az FPGA-ra, mely maximálisan 100 MHz órajelet képest szolgáltatni az FPGA számára. Az elkészített HIL szimulátornál a cél 25ns –os időlépték volt, így én 40 MHz-es órajelet használtam.

Az FPGA közel 10.000 logikai cella kapacitása, a maximális használható 100 MHz-es órajel, a fejlesztőkártya kis mérete és rendkívül egyszerű kezelhetősége tökéletesen megfelelőnek bizonyultak ezen feladatra. Természetesen további léptékcsökkentés, nagyobb, komplexebb felépítés és bonyolultabb numerikus megoldási módszerek használatának igénye egy arra megfelelőbb fejlesztőkártya igényét is magával vonja.

3.1.3 Illesztés

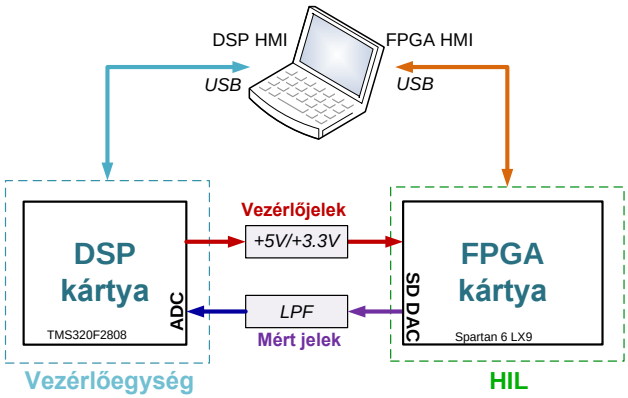
Az illesztés a rendszer egyes elemei között igen egyszerűnek mondható, ugyanis mind a DSP kártya, mind az FPGA kártya USB kábelen csatlakozik a számítógéphez, a két egység között pedig igen kevés alkatrészt igénylő átalakításra volt szükség.

A DSP kártyától érkező vezérlőjelek +5V–os digitális jelek, melyeket az FPGA kártya lábaival kompatibilis +3.3V–os jelszintre kellett osztani. A 2.2.2 fejezetben leírtak alapján a Σ/Δ D/A átalakítók kimenetére még egy analóg aluláteresztő szűrőfokozat kerül. Mivel itt rendkívül fontos, hogy a megfelelő értékekkel dolgozzunk, így kalkulálni kellett azon lábak bemeneti tulajdonságaival, melyeket az A/D átalakításra konfiguráltunk fel. Ez azt jelenti, hogy a belső ellenállást és kapacitást is számításba vettem, ezzel együtt a teljes szűrő fokozatot jól mutatja a következő ábra.



3.1.3 ábra: A Σ/Δ D/A-k kimenetének másodrendű aluláteresztő szűrése

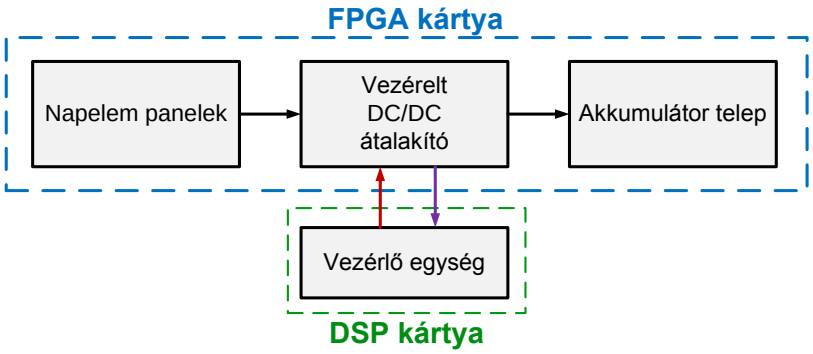
A szűrő törésponti frekvenciái 56.4 kHz és 239.3 kHz, így a center f_C=116.177 kHz. A Σ/Δ D/A 40MHz-es integrátorral dolgozik, így mivel a DSP ADC–vel 8 kHz-es frekvenciával veszünk mintát a jelből, így a nagyfrekvenciás tartományba került kvantálási zajt igen hatékonyan kiszűrjük.



3.1.4 ábra: Az illesztett tesztkörnyezet blokkvázlata

3.2 A modellezett rendszer

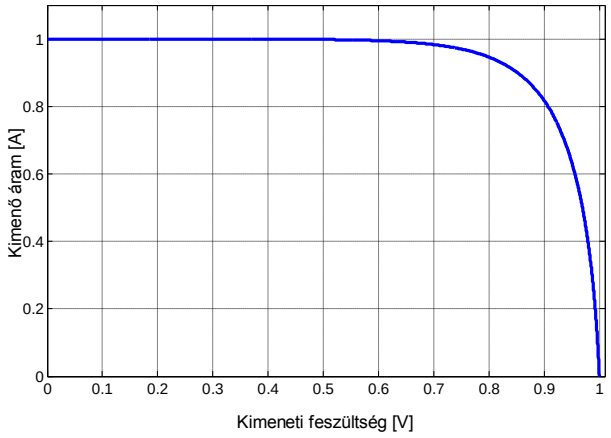
A modellezett rendszer a 3.2.1 ábrán látható azon rész, mely az FPGA kártyán található. Tehát egy olyan mini naperőmű, mely a felhasználó által konfigurálható napelem panelekből áll és egy akkumulátor töltése által energiatárolással rendelkezik.



3.2.1 ábra: A modellezett rendszer

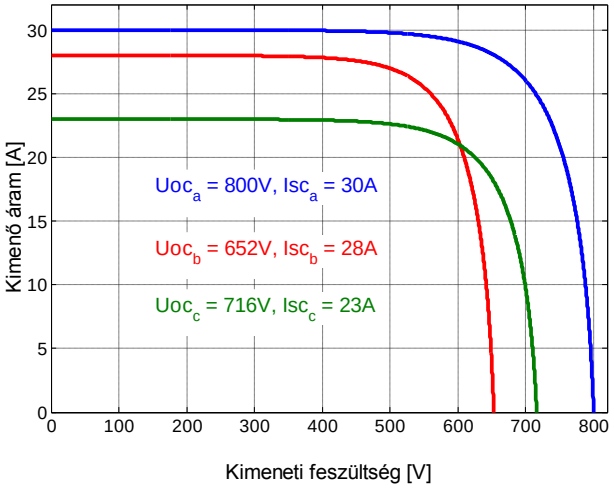
3.2.1 Napelem panelek

A 2.1.2 fejezettel összhangban az egyes napelem cellák párhuzamos és soros összekapcsolásával szinte tetszőleges üresjárási feszültséggel illetve rövidzárási árammal jellemezhető panelt össze lehet állítani. A modellalkotás a következőképpen történt: alapul vettem egy létező, MSX-60 típusú napelem panelt. Ennek a karakterisztikáját normáltam, ekkor egy olyan karakterisztikát kaptam, ahol mind az üresjárási feszültség, mind a rövidzárási áram 1-re normált. Ezt látjuk a következő ábrán.



3.2.2 ábra: A normált napelem panel karakterisztika

Ezt a karakterisztikát veszem alapul, diszkretizálom, azaz a feszültség érték alapján adott pontra osztom (pl.: 4096). A megvalósított szimulátorban 3 napelem panelt lehet szimulálni, természetesen többet is lehetne, de a módszer bemutatására a 3 is tökéletesen megfelel. E három napelem panel párhuzamosan van kapcsolva, és mindegyiknek külön-külön a szimuláció alatt változtatható az üresjárási feszültsége és a rövidzárási árama. Kis közelítéssel élve a hőmérséklet elsősorban az üresjárási feszültséget, a megvilágítás pedig a rövidzárási áramot befolyásolja. Ezáltal a felhasználó a szimuláció közben vizsgálni tudja ezen hatásokat a rendszerre. A következő ábrán a három panel karakterisztikája látható a 6 - futási időben megadható - paraméter példaértékei esetén.



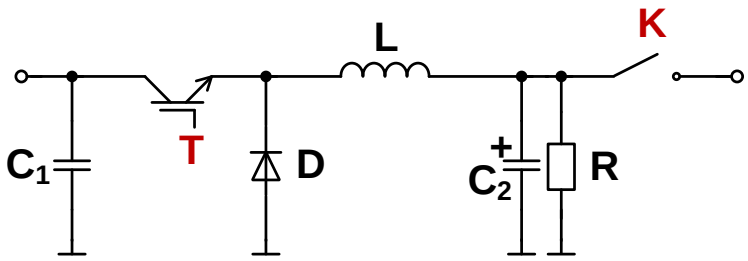
3.2.3 ábra: A 3 napelem panel karakterisztikája példa paraméterek esetén

A rendszer működéséből adódó feszültség és a felhasználó által megadott üresjárási feszültség paraméterek meghatározzák a pozíciót a feszültségtengelyen a normált karakterisztikában, amelyből adódik három áramérték, melyeket a paraméterként megadott rövidzárási áramokkal felskálázunk. Az adódó három értéket összegezve kapjuk eredményül a napelemes rendszer kimenő áram értékét (részletesen lásd később a 3.3.1.2 fejezetben).

3.2.2 Vezérelt DC/DC átalakító

A vezérelt DC/DC egy egyszerű feszültségcsökkentő (buck) konverter, melyben a vezérelt elemek egy IGBT, illetve egy DC kontaktor. Az átalakítóban az utóbbi elemen kívül minden mást ideálisnak vettünk. Az IGBT és a DC kontaktor vezérlése magas-aktív jelekkel történik, tehát vezérlőjeleik (rendre **T**, **K**) aktív állapotakor vannak bekapcsolva, egyébként pedig kikapcsolt állapotúak.

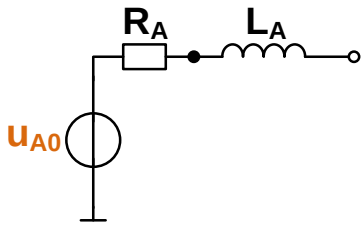
Az IGBT megfelelő vezérlésével szabályozható a $\mathbf{C_2}$ kimeneti kondenzátor töltőárama, és feszültsége. A DC kontaktorral segítjük azt, hogy a konverter kimenetére kapcsolt fokozat és a $\mathbf{C_2}$ kondenzátor csak az előtöltést követően kapcsolódjon össze.



3.2.5 ábra: Vezérelt DC/DC konverter kapcsolási rajza

3.2.3 Akkumulátor

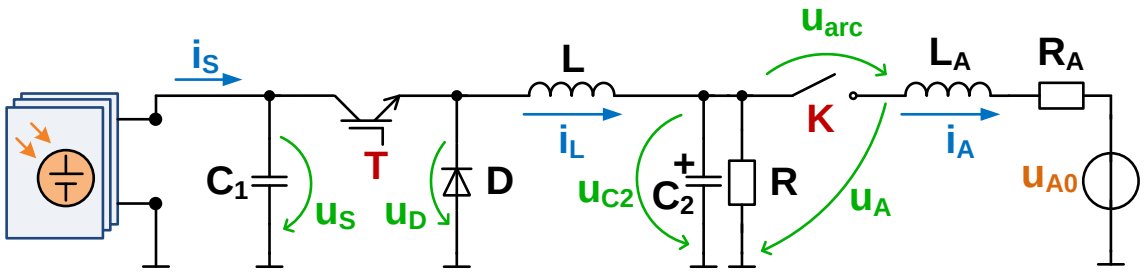
Az akkumulátor modellje alapvetően egy feszültségforrásból és az akkumulátor belső impedanciáját reprezentáló soros $\mathbf{R_A}$ ellenállásból és $\mathbf{L_A}$ induktivitásból áll.



3.2.6 ábra: Az akkumulátor modellje

3.2.4 A rendszer jelei, működése és leíró egyenletei

Ahhoz, hogy a dinamikus leíráson keresztül a megfelelő Simulink modellt kialakíthassam, először az állapotváltozós leírásra volt szükség. Ehhez a jellemző jelek elnevezéseit és a referenciairányokat a 3.2.7 ábrán találjuk.



3.2.7 ábra: A modellezett rendszer helyettesítő kapcsolása

Az $\mathbf{u_S}$ érték alapján a napelemes rendszer karakterisztikájából előáll az $\mathbf{i_S}$ áram. Az IGBT-t a **T** PWM jellel vezéreljük, a következő fokozat aszinkron buck konverter lévén a **D** dióda ellenütemben kapcsol az IGBT-vel. Ezáltal szabályozzuk az **L** tekercsen folyó $\mathbf{i_L}$ áramot, közvetetten $\mathbf{C_2}$ kondenzátor előtöltését. Az **R** kisütő ellenállás. $\mathbf{i_L} \geq 0$, mivel a **D** dióda ideális, így záróirányban áram rajta nem folyhat, sem $\mathbf{C_1}$ felé az IGBT-n keresztül.

A $\mathbf{C_2}$ kondenzátort előtöltjük közel akkora feszültségre, mint $\mathbf{U_{A0}}$ akkumulátor feszültség, majd csak azután kapcsoljuk rá az akkumulátorra. A kontaktor bekapcsolt állapota mellett $\mathbf{u_{C2}}$ feszültség jut az akkumulátorra, azonban kikapcsolásakor számolni kell a kontaktor $\mathbf{U_{arc}}$ ívfeszültségével. Ez tehát akkor lép fel, amikor a kontaktor **K** vezérlése inaktívvá válik, de az akkumulátorra jutó $\mathbf{i_A}$ áram még nagyobb, mint nulla.

Típus	Jelölés	Leírás
Paraméterek	U_{ocA}^{-1}, U_{ocB}^{-1} , U_{ocC}^{-1} , I_{scA}, I_{scB}, I_{scC}, U_{A0}	Parancs a HIL HMI-től (PC)
Vezérlőjelek	T, K	Vezérlőegységtől (DSP kártya)
Kimenetek	u_S, u_A, i_L	Vezérlőegységhez (DSP kártya)
Monitorozás	u_S, i_S, u_D, i_L, u_{C2}, u_A, i_A	Funkció a HIL HMI-ben (PC)

3.2.1 táblázat: A rendszer modelljének jelei

Ezt követően a fenti leírással összhangban a dinamikus működést leíró egyenletek a következők:

$$C_1 \cdot \frac{d\,u_S(t)}{dt} = i_S(t) - i_L(t) \tag{3.2.1}$$

$$L \cdot \frac{d\,i_L(t)}{dt} = u_D(t) - u_{C_2}(t), \text{ ahol } i_L(t) \geq 0 \tag{3.2.2}$$

$$C_2 \cdot \frac{d\,u_{C_2}(t)}{dt} = i_L(t) - i_A(t) - \frac{u_{C_2}(t)}{R} \tag{3.2.3}$$

$$L_A \cdot \frac{d\,i_A(t)}{dt} = u_A(t) - U_{A0}(t) - i_A(t) \cdot R_A \tag{3.2.4}$$

Mivel az FPGA-n ezen egyenletek periodikus megoldása szükséges diszkrét időben, így numerikus integrálási megoldási módszert alkalmaztam (lásd később részletesen). Ehhez az előbbi elsőrendű differenciálegyenleteket integrális formában írjuk fel:

$$u_S(t) = \frac{1}{C_1}\int_0^t \left(i_S(t) - i_L(t)\right) dt \tag{3.2.6}$$

$$i_L(t) = \frac{1}{L}\int_0^t \left(u_D(t) - u_{C_2}(t)\right) dt, \text{ ahol } i_L(t) \geq 0 \tag{3.2.7}$$

$$u_{C_2}(t) = \frac{1}{C_2}\int_0^t \left(i_L(t) - i_A(t) - \frac{u_{C_2}(t)}{R}\right) dt \tag{3.2.8}$$

$$i_A(t) = \frac{1}{L_A}\int_0^t \left(u_A(t) - U_{A0}(t) - i_A(t) \cdot R_A\right) dt \tag{3.2.9}$$

továbbá az ideális dióda és a nem ideális DC kontaktor miatt igazak:

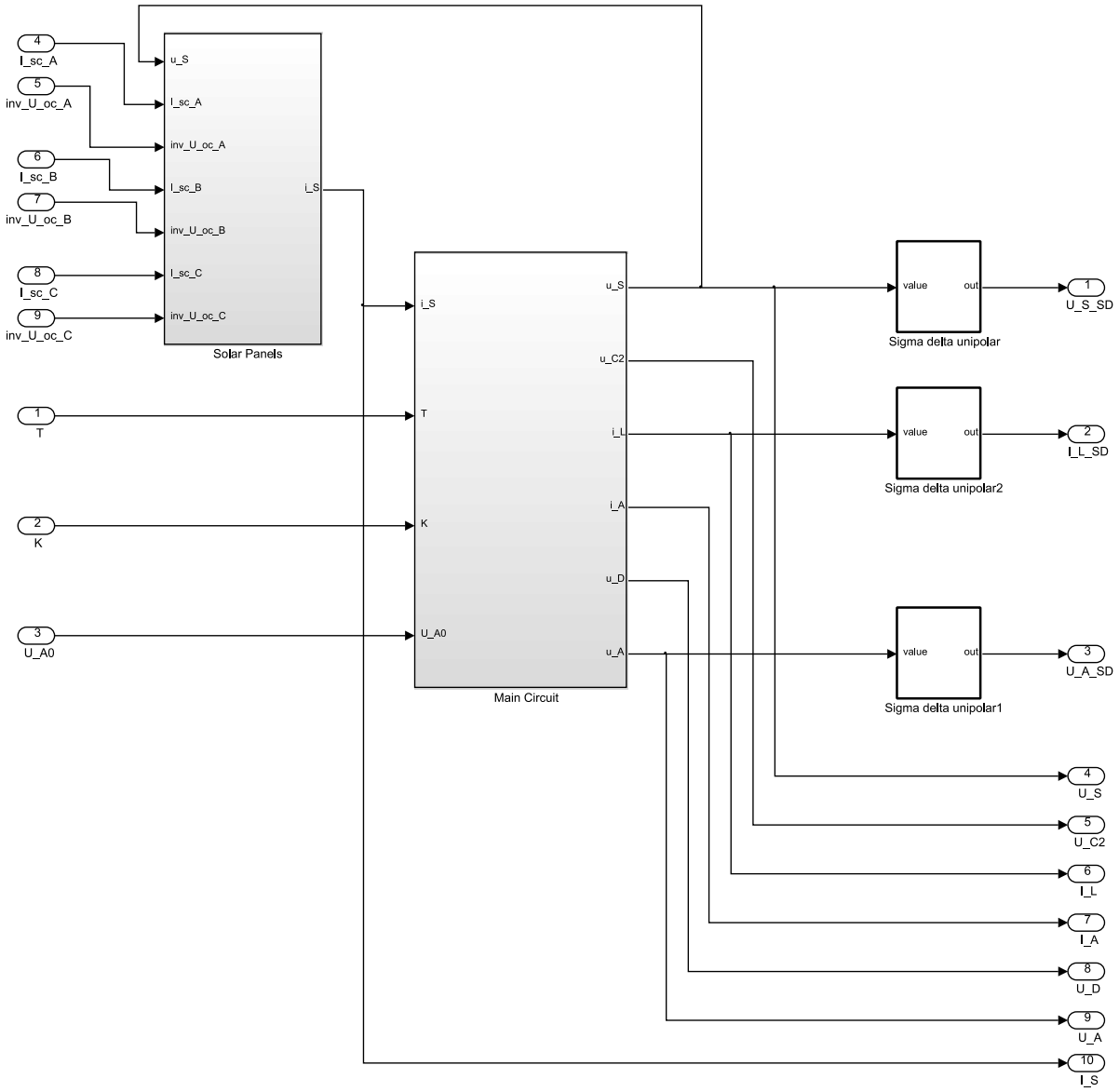
$$u_D(t) = \begin{cases} u_S(t), & T = 1 \\ 0, & T = 0 \text{ és } i_L(t) > 0 \\ u_{C_2}(t), & T = 0 \text{ és } i_L(t) = 0 \end{cases} \tag{3.2.10}$$

$$u_A(t) = \begin{cases} u_{C_2}(t), & K = 1 \\ u_{C_2}(t) - U_{arc}, & K = 0 \text{ és } i_A(t) > 0 \\ U_{A0}(t), & K = 0 \text{ és } i_A(t) = 0 \end{cases} \tag{3.2.11}$$

3.3 A Simulink modell

Ebben a fejezetben mutatom be a létrehozott Simulink modellt az elkülöníthető részeken egyesével haladva az eddig leírtak alapján, illetve ismertetem ezen részek HDL szintű megvalósításához szükséges részleteket.

3.3.1 A szimulátor hardver részei



3.3.1 ábra: A felső szintű Simulink modell

Jól láthatóan két nagyobb részre osztható a modell. Az egyik a napelemes rendszer modellje, amely a 3 napelem panel modelljét tartalmazza. Ennek 7 bemenete van: a 6 darab felhasználó által állítható paramétere és a napelemes rendszerre kapcsolt fokozat által meghatározott ($\mathbf{u_S}$) feszültsége. Áramkimenete ($\mathbf{i_S}$) kerül a következő fokozatra, mely az akkumulátortöltőt és az akkumulátor modelljét tartalmazza.

Ezen fokozatnak bemenetei az IGBT-nek T, a DC kontaktornak K digitális vezérlőjelei és az akkumulátor felhasználó által állítható ($\mathbf{U_{A0}}$) paramétere. A teljes modellnek 10 kimenete van, ebből 7 az FPGA HMI-je általi monitorozás céljából, 3 pedig Σ/Δ DA átalakítókon keresztül kerül a kimenetre.

3.3.1.1 Számábrázolás

A különböző mennyiségek ábrázolása fixpontosan történik, az FPGA-ban vannak beépített hardveres szorzók, melyekkel gyorsabb lesz a műveletvégzés, mint lebegőpontos számítással.

A mennyiségek két csoportját különböztessük most meg, az egyik az előző fejezetben tárgyalt integrálegyenletekben a szorzótényezők (pl.: $\frac{1}{C_1}$), a másik csoport pedig a többi változó (pl.: u_S). Nézzük először a második csoportot.

Legyen a mennyiség általánosan jelölve x-szel, ez lehet feszültség, áram, vagy ellenállás érték, vagy annak reciproka. Egy Excel tábla segítségével állítottam elő a szükséges fixpontos számábrázolást, mely egy Matlab eszköz, a Fixed-Point Designer segítségével valósul meg.

Az általános alakja: **fixdt(s, s+n+m, m)**, ahol ***s*** az előjelbit, ***n*** az egészrész bitek száma, és ***m*** a törtrész bitek száma. Az Excel táblában megadható az adott mennyiség minimum (x_{min}) és maximum (x_{max}) értéke, illetve felbontása (q), mind annak saját dimenziójában értelmezve. Ekkor a *fixdt*–s alakhoz a számítások a következők:

$$n = \begin{cases} \lceil \log_2 (\max\{|x_{min}|, |x_{max}|\}) \rceil, & \text{ha } |x_{min}| \geq 1 \text{ és } |x_{max}| \geq 1 \\ \lfloor \log_2 (\max\{|x_{min}|, |x_{max}|\}) \rfloor, & \text{egyébként} \end{cases} \quad (3.3.1)$$

$$m = \begin{cases} \lceil (-1) \cdot \log_2 q \rceil, & \text{ha } q < 1 \\ \lfloor (-1) \cdot \log_2 q \rfloor, & \text{egyébként} \end{cases} \tag{3.3.2}$$

Mivel az ábrázolás az x értékének teljes tartományára értendő, így természetesen a szélsőértékek abszolút értékei közül a nagyobbik dominál, ennek 2-es alapú logaritmusa mondja meg, hogy hány biten reprezentálható. Ezen számnak a felső egészrésze lesz a megfelelő ***n*** –re abban az esetben, ha a maximum és minimum értékek abszolút értékei legalább 1-re adódnak. Ellenkező esetben az alsó egészrész adja ***n*** értékét.

A törtrész bitek számának meghatározásához az adott mennyiség felbontását vesszük alapul (tipikusan 1-nél kisebb szám). A 2-es alapú logaritmusának -1–szerese adja meg, hogy hány biten reprezentálható a szám, ennek felső egészrésze adja a törtrész bitek számát abban az esetben, ha a felbontás 1-nél kisebb szám, alsó egészrésze egyébként.

Vegyünk egy példát: $\mathbf{u_S}$ értéke -2000V és 2000V között mozoghat, 0.1V –os lépésekben. Ekkor az adódó paraméterek: n=11, m=4, és természetesen s=1, mivel az értéke lehet negatív is, tehát előjeles. Ekkor a számábrázolási alak fixdt(1,16,4) –re adódik. Ha visszaszámoljuk ebből a tényleges értékeket, akkor természetesen nem pont a kijelölt tartományt és felbontást kapjuk (alsó és felső egészrészek képzése miatt). Az $\mathbf{u_S}$ minimuma -2048V, maximuma 2047.9375V, és az LSB-re, azaz a felbontásra 0.0625V adódik.

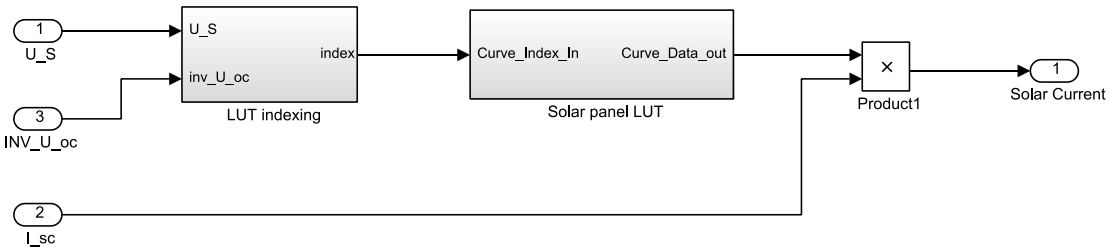
A mennyiségek másik csoportja azok a mennyiségek, melyek az integrálásokhoz tartozó szorzótényezők (pl.: $\frac{1}{R}$ vagy $\frac{dt}{C_1}$). Ahhoz, hogy integráláskor ne akkumuláljuk a kvantálási hibát, ki kell bővíteni az ábrázolási tartományt, melyhez meg kell határozni az integrátorhoz adandó bitek számát. Alapvetően ***s***+***n***+***m***=18, azaz az összes bitszám 18, mert az FPGA-ban 18 bites beépített hardveres szorzókat használunk.

A metódust lényege a következő: a szorzótényezők abszolút értéke kisebb, mint 1, ezért a szorzat LSB-je alacsonyabb helyiértékű biten fog végződni, mint a szorzandó értéké. Ezeket akár el is hagyhatnánk, viszont az integrátor kommulálja a hibát, ezt viszont el kell kerülni, ezért azokat kibővítjük annyi bittel, amennyit a szorzótényező indokol. Az integrátor kimenetén már el lehet hagyni a felesleges biteket, hiszen nem akkumulálódik a hiba, mert az integrátor pontosan tárolja az értéket.

3.3.1.2 Napelemes táplálás

A 3.2.1 fejezetben leírtak alapján a modell egy létező napelem panel karakterisztikájából indul ki. Az MSX-60 típusú panel karakterisztikájának előállítására egy gyártó által támogatott függvényt használok, melyet egy Matlab ***m***-fájlban implementáltam. Ezen áram-feszültség karakterisztikát 1-re normáltam, és 4096 pontra osztottam a feszültségértékeket, így kaptam egy 12 biten címezhető, 4096 elemű vektort. Tehát ezen vektor 0-tól 4095-ig 1 egészenként indexelhető, és az indexhez tartozó értékek a normált karakterisztikából az áramértékek.

Lássuk ez alapján egy napelem panel Simulink modelljét. Tehát szükség van a vektor valamilyen módú tárolására, és egy indexelő egységre. Amint előállt a tárolt karakterisztikából az áram értéke, csak egy szorzásra van szükség a felhasználó által futási időben megadható rövidzárási áram értékével.



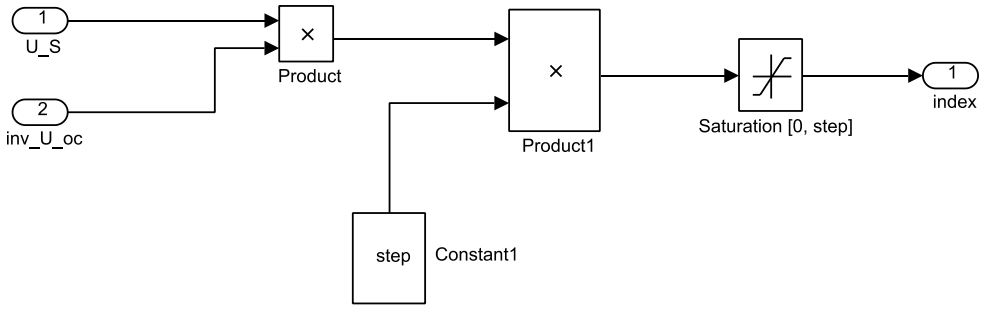
3.3.2 ábra: Egy napelem panel modellje (*Solar Panel*)

Mivel futási időben változtatni akarjuk az egyes napelem karakterisztikákat, így előre ezeket nem tárolhatjuk le. Ezért volt szükség a karakterisztika normálására, ugyanis ekkor tetszőlegesen skálázhatóvá válnak a feszültség- és áram értékek.

$$\text{index} = \frac{u_S}{U_{oc}} \cdot 4095 \tag{3.3.3}$$

A 3.3.3 egyenlet általánosan igaz egy napelem panelre. Az $\mathbf{U_{oc}}$ helyett annak reciprokát állítja be a fejlesztő futási időben, így elkerüljük az osztást kis feltételt szabva csupán. Ekkor a reciprokkal és 4095-tel szorozzuk az $\mathbf{u_S}$ értékét, így kapjuk az indexelést.

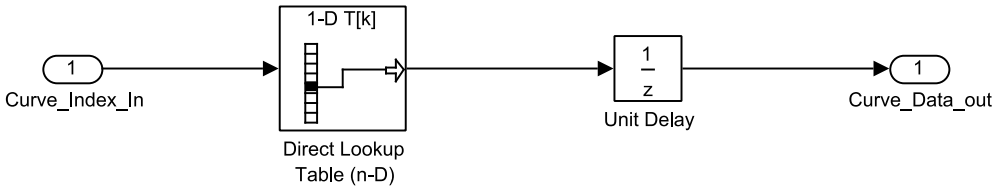
$$\text{index} = u_S \cdot {U_{oc}}^{-1} \cdot \text{step}, \text{ahol step} = 4095 \tag{3.3.4}$$



3.3.3 ábra: Az indexelés mechanizmusa (*LUT indexing*)

A 3 panel üresjárási feszültsége különböző, így az $\mathbf{u_S}$ értéke lehet nagyobb, mint valamely panel üresjárási feszültsége. Tehát ha $\mathbf{u_S}>\mathbf{U_{oc}}$ áll fenn, akkor **index**>4095 adódna, ami nem lehetséges. Szaturáljuk az értékét 0 és 4095 közé, így ha **index**>4095 lenne, akkor marad 4095, melyhez zérus kimenő áram fog tartozni, ahogy a valóságban is történik.

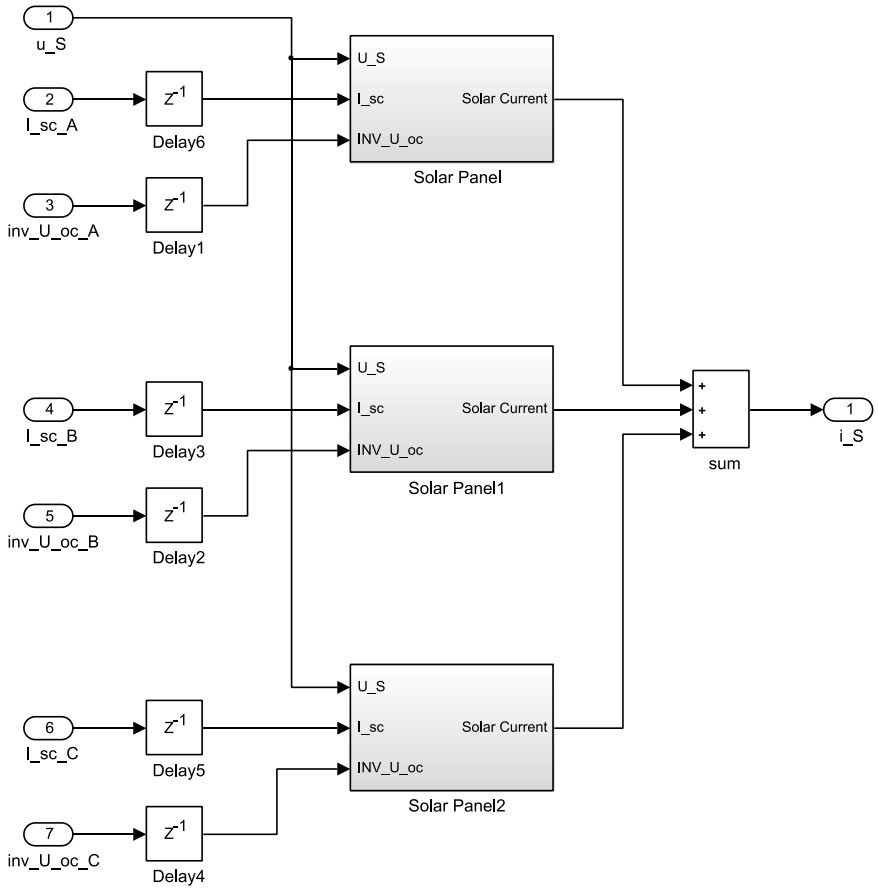
Az indexelés azon vektort kell címezze, mely a normált karakterisztikát tárolja. Mivel 3 különböző karakterisztikájú napelem panelünk van, így 3-szor 4096 érték letárolására lenne szükség regiszterekben. Szeretnénk, ha az FPGA-ban adott blokk RAM-okban tárolnánk inkább az értékeket, mely jóval kevesebb erőforrás felhasználást jelent. Meg kellett oldani, hogy blokk RAM-okba képződjenek le az értékek. A trükk a következő volt: egy mátrixokat tárolni képes direkt lookup table-t használtam, mely a normált karakterisztikát tartalmazó vektort (1D-s mátrix) tárolja. Ennek kimenetére egy 1 órajelnyi (időléptékű) késleltetést tettem.



3.3.4 ábra: A normált napelem karakterisztika Lookup-Table modellje (*Solar Panel LUT*)

Ekkor a kódgenerálást követően még mindig regiszterekben tárolva jelentek meg a vektor elemei, azonban a lényeg az, hogy a szintézis eszköz ezen értékek késleltetett használatából értelmezi, hogy az optimális elrendezés a blokk RAM-ok használata, ahol a kiolvasás pont ezen késleltetésnyi időt vesz igénybe. Minden egyéb - a Simulink modellben levő elrendezésből történő - generált HDL kód esetén elmaradt a blokk RAM-ok használatának felismerése a szintézis eszköz által.

Az előzőekben tehát egy napelem panel modelljének működését tekintettük át. A három panel együttesen 3 áramot állít elő, és mivel a panelek párhuzamosan vannak kapcsolva, így ezen értékek összegzésére van szükség, amiből megkapjuk az $\mathbf{i_S}$ kimenő áramot.

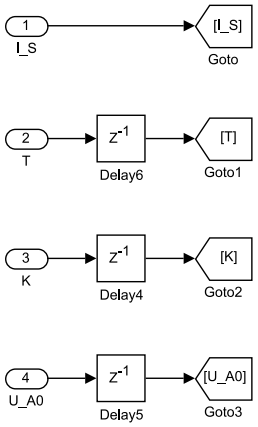


3.3.5 ábra: A 3 párhuzamosan kapcsolt napelem panel modellje (*Solar Panels*)

A napelemes rendszer bemenetei tehát a táplált rendszer által meghatározott $\mathbf{u_S}$ feszültség, az egyes panelek rövidzárási áramának értékei (**I_sc_A**, **I_sc_B** és **I_sc_C**) és üresjárási feszültségeinek reciprokai (**inv_U_oc_A**, **inv_U_oc_B** és **inv_U_oc_C**). Kimenete a panelek által létrehozott $\mathbf{i_S}$ áram. Mivel a 6 állítható paraméter a teljes modell bemenete, így szinkron mintavételüket mindnél a legegyszerűbb módon, egy 1 órajelnyi késleltetéssel tesszük.

3.3.1.3 Állapotváltozók periodikus számítása

Az állapotváltozók számításához az előző fejezetben tárgyalt egyenletek diszkretizációja szükséges. Erre számos módszer létezik, a legtöbb numerikus integráláson alapszik: előrelépő Euler, hátralépő Euler és trapéz módszer a legelterjedtebbek. A legösszetettebb módszer a negyedrendű Runge-Kutta, nagy hátránya viszont, hogy nagyobb a számításigénye, mint az egyszerűbbeké. A megvalósított szimulátorban előrelépő Euler módszert alkalmaztam szem előtt tartva első körben, hogy a megfelelő működés mellett a lehető legkisebb időlépésekkel dolgozzon a szimulátor.



3.3.6 ábra: A bemenetek illesztése, szinkronizálása

Először is nézzük a bemeneteket: az $\mathbf{i_S}$ áram a napelemes rendszerből érkező áram, a **T** és **K** rendre az IGBT és a kontaktor vezérlőegységtől érkező digitális vezérlőjele, az $\mathbf{U_{A0}}$ pedig a felhasználó által állítható akkumulátor feszültség. Mivel a **T**, **K**, és $\mathbf{U_{A0}}$ a teljes rendszeren kívülről érkező jelek, így a napelemes rendszernél leírthoz hasonló szinkronra van szükség. Az $\mathbf{i_S}$ a rendszer része, így ott nincs szükség szinkronizációra.

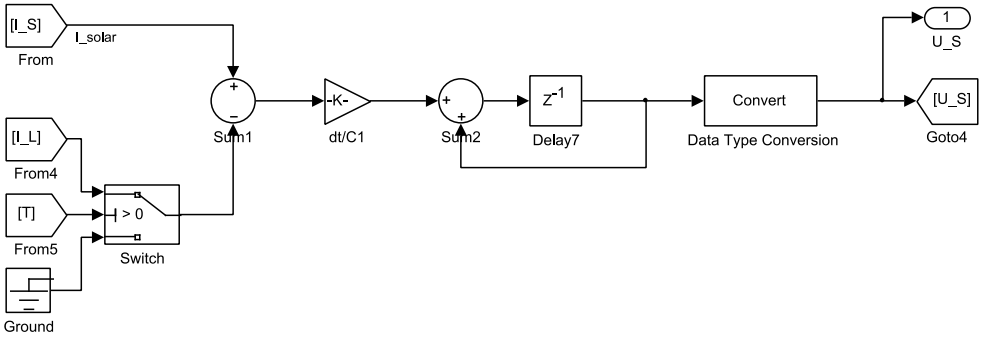
Véleményem szerint a leghatékonyabb módja az áttekintésnek, ha a megvalósítást az állapotváltozós egyenletekkel együtt mutatom be egyesével. Az áramköri paraméterek egy Matlab ***m***-fájlban lettek deklarálva, ezek közül a legfőbbek a következők:

$$\begin{array}{lll} C_1 = 680\ \mu F & R = 30.5\ k\Omega & fpga\_clk = 40\ MHz \\ C_2 = 2.2\ mF & R_A = 0.0453\ \Omega & dt = 25\ ns \\ L = 2.7\ mH & L_A = 0.02\ mH & \end{array} \qquad (3.3.5)$$

1. *Az* $\mathbf{u_S}$ *állapotváltozó számítása*

$$u_S(t) = \frac{1}{C_1}\int_0^t \left(i_S(t) - i_L(t)\right) dt \tag{3.3.6}$$

Az **iL** áram értéket abban az esetben vonjuk ki az $\mathbf{i_S}$ értékből, ha az IGBT vezet, azaz a jelenlegi modellben ha a T vezérlőjel aktív.



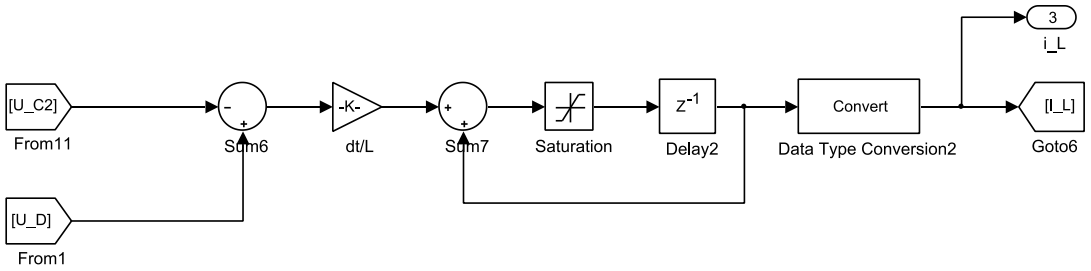
3.3.7 ábra: Az u_S változó számítása

A numerikus integráláskor a **dt** időléptékkel való szorzást összevonjuk az integrál előtti szorzóval (itt pl.: $\frac{dt}{C_1}$), majd ezen értéket az integrandussal szorozva integrálunk. Az előrelépő Euler módszert alkalmazva az integrátor egy pozitívan visszacsatolt késleltetés. Az integrál előtti szorzók és az integrál utáni értékek számábrázolása a 3.3.1.1 fejezetben leírtak alapján történik.

2. *Az* $\mathbf{i_L}$ *állapotváltozó számítása*

$$i_L(t) = \frac{1}{L}\int_0^t \left(u_D(t) - u_{C_2}(t)\right) dt, \text{ ahol } i_L(t) \geq 0 \tag{3.3.7}$$

A **L** fojtótekercs áramának értéke legalább zérus a 3.2.7 ábrán jelölt referencia irányban. Ha az IGBT vezet, akkor sem folyhat az $\mathbf{i_L}$ áram a jelölttel ellenkező irányban az IGBT-n, illetve a **D** dióda ideális, így záróirányban azon nem folyhat áram.



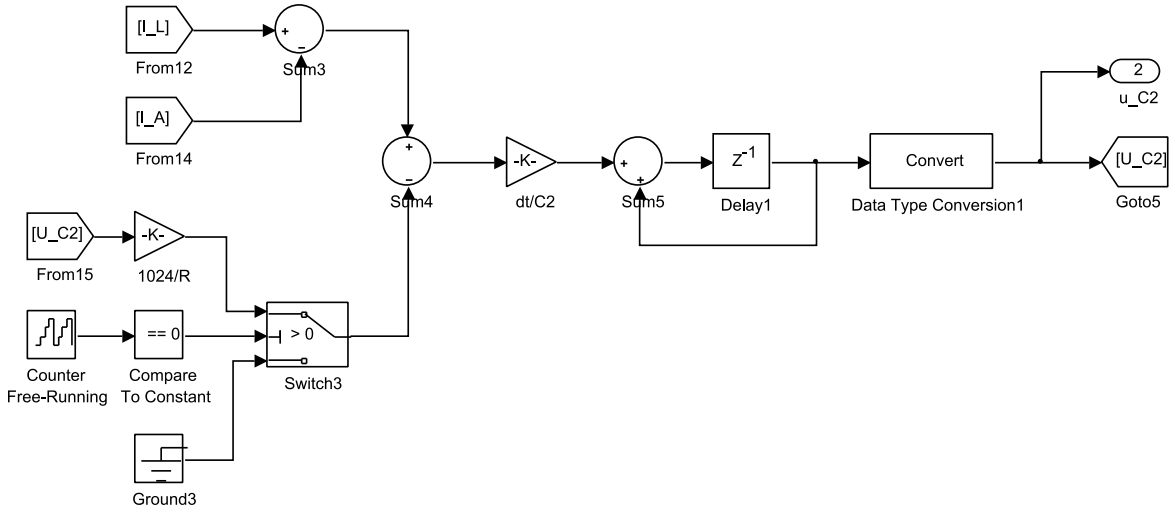
3.3.8 ábra: Az i_L változó számítása

Emiatt a megfelelő működés érdekében az integrátoron belülre került egy szaturáció, mely lényegében az áram alsó, zérus korlátját biztosítja.

3. Az $\mathbf{u_{C2}}$ *állapotváltozó számítása*

$$u_{C_2}(t) = \frac{1}{C_2}\int_0^t \left(i_L(t) - i_A(t) - \frac{u_{C_2}(t)}{R} \right) dt \qquad (3.3.8)$$

Itt arról van szó, hogy a $\mathbf{C_2}$ kondenzátort töltjük, melynek $\mathbf{u_{C2}}$ feszültsége egy állapotváltozó. A kondenzátorral párhuzamosan kapcsolt **R** ellenállás $R \cdot C_2$ (~1 perces) időállandóval süti ki a $\mathbf{C_2}$ kondenzátort.



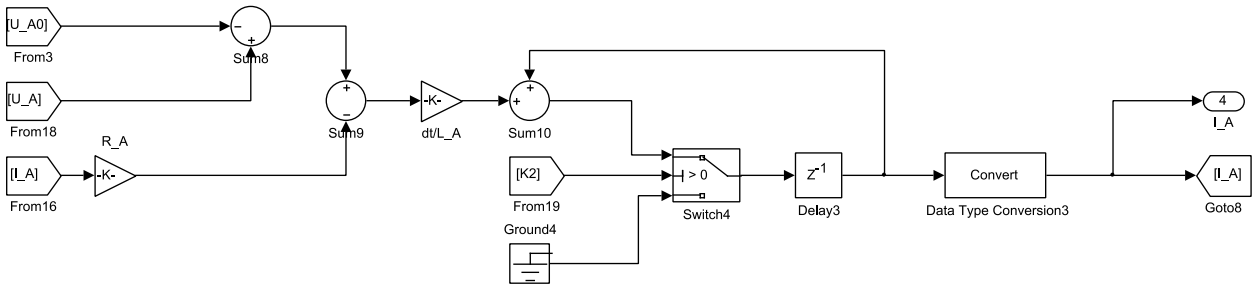
3.3.9 ábra: Az u_{C2} változó számítása

A kiegészítő számlálóra azért volt szükség, mert anélkül olyan kis lépésekben csökkenne $\mathbf{u_{C2}}$, hogy a választott számábrázolásban zérus lenne, emiatt egy határ alá képtelen lenne csökkenni. Emiatt 1024 lépésenként hajtjuk végre az $1024 \cdot \frac{u_{C2}}{R}$ érték levonását, így ábrázolható számot kapunk.

4. Az $\mathbf{i_A}$ állapotváltozó számítása

$$i_A(t) = \frac{1}{L_A} \int_0^t (u_A(t) - U_{A0}(t) - i_A(t) \cdot R_A)\, dt \tag{3.3.9}$$

Az $\mathbf{i_A}$ áram már az akkumulátorhoz tartozik, így a modellben az előállításához a DC kontaktor is szerepet játszik.



3.3.10 ábra: Az i_A változó számítása

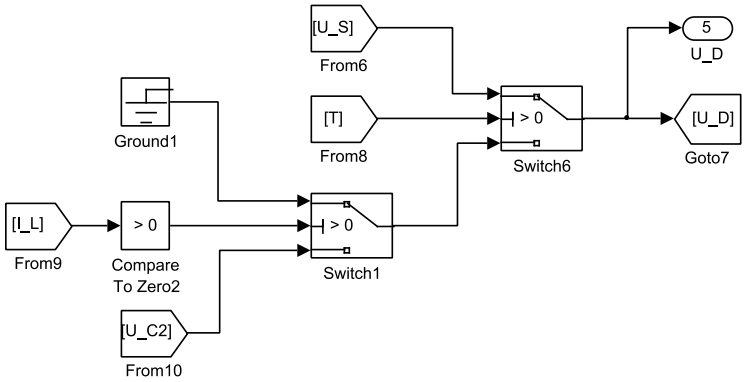
A 3.3.9 egyenlet alapján jól látszik, hogy mely rész kerül integrálásra, azonban egy $\mathbf{K_2}$ jeltől függően nem mindig, mely magyarázatra szorul. A $\mathbf{K_2}$ értéke abban az esetben 1, ha a K értéke 1, vagy ha az $\mathbf{i_A}$ áram értéke nagyobb, mint nulla.

Amikor a $\mathbf{K_2}$ értéke 0, akkor az integrandus zérus kell legyen. Azért nem lehet csupán a **K**-tól függő az integrálás, mivel ahogy az a 3.2.4 fejezetben is említettem, a kontaktor ívfeszültsége miatt folyhat $\mathbf{i_A}$ áram akkor is, ha már nincs bekapcsolva a kontaktor (**K**=0).

5. *Az* $\mathbf{u_D}$ *állapotváltozó számítása*

$$u_D(t) = \begin{cases} u_S(t), & T = 1 \\ 0, & T = 0 \text{ és } i_L(t) > 0 \\ u_{C_2}(t), & T = 0 \text{ és } i_L(t) = 0 \end{cases} \tag{3.3.10}$$

A **D** dióda ideális, a megvalósítása tökéletesen egyezik a 3.3.10 egyenletben foglaltakkal.

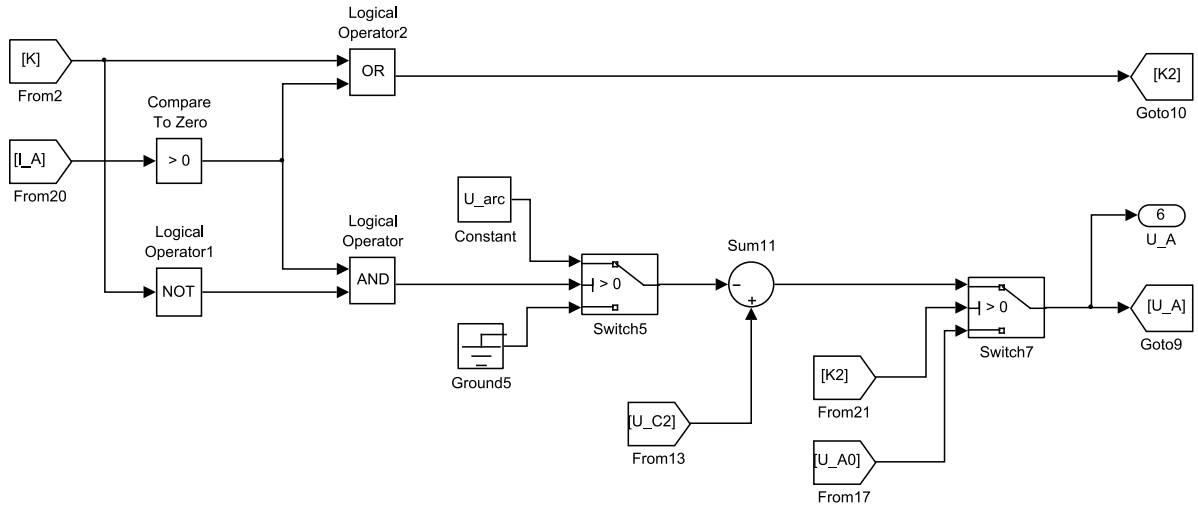


3.3.11 ábra: Az u_D változó számítása

6. *Az* $\mathbf{u_A}$ *állapotváltozó számítása*

$$u_A(t) = \begin{cases} u_{C_2}(t), & K = 1 \\ u_{C_2}(t) - U_{arc}, & K = 0 \text{ és } i_A(t) > 0 \\ U_{A0}(t), & K = 0 \text{ és } i_A(t) = 0 \end{cases} \tag{3.3.11}$$

Az $\mathbf{u_A}$ akkumulátor feszültség előállításának egyszerűbb áttekinthetősége érdekében előállítottam a már említett $\mathbf{K_2}$ segédváltozót.

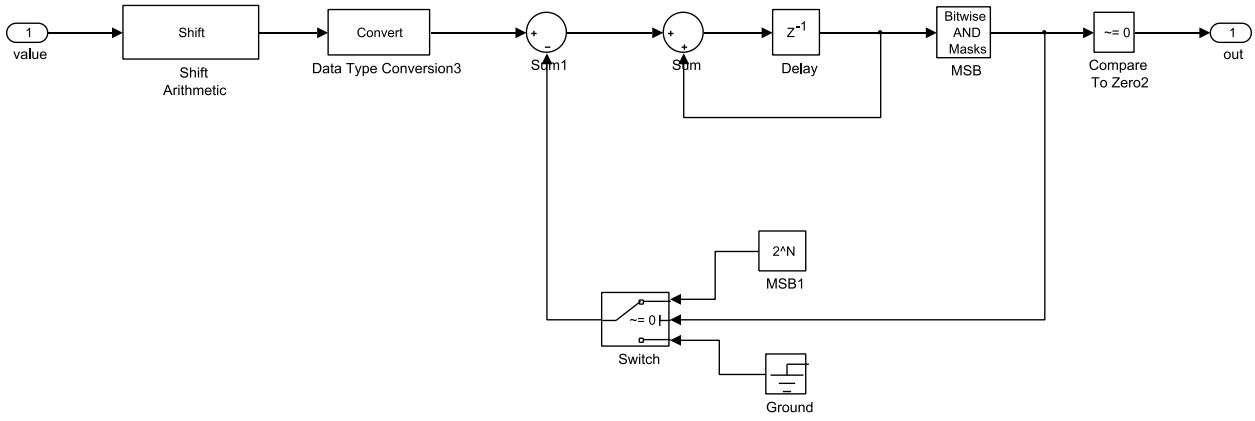


3.3.12 ábra: Az u_A változó számítása

Az $\mathbf{u_{C2}}$ értékéből akkor vonjuk le az ívfeszültséget, amikor a kontaktor vezérlése már inaktív (**K**=0), de az $\mathbf{i_A}$ áram értéke még nagyobb, mint nulla, egyébként marad $\mathbf{u_{C2}}$. Az így előállt érték csak akkor adja az $\mathbf{u_A}$ értékét, ha a $\mathbf{K_2}$=1, mivel ezen esetek a 3.3.11 egyenletben az első két sorral egyeznek, egyébként ha $\mathbf{K_2}$=0, akkor az u_A megegyezik $\mathbf{u_{A0}}$ értékével.

3.3.1.4 Szigma-delta modulátor

Alapvetően ebben a rendszerben csupán az $\mathbf{i_A}$ akkumulátor áram vehet fel negatív értéket, de mivel annak értékére nincsen szükség a vezérlőegységen belül, így unipoláris Σ/Δ DA átalakítókra volt csak szükség. A megvalósítása a 3.3.13 ábrán látható.



3.3.13 ábra: Az unipoláris Σ/Δ DAC Simulink modellje (*Sigma delta unipolar*)

Az átalakítandó ***value*** érték kerül a bemenetre, melynek a törtrész bitjeit nem törtrész bitekként visszük tovább, ezért egy átalakítás történik egész számmá. Az eredmény annyi biten ábrázolt, amennyi az eredeti szám összes bitszáma volt. Ezután ezt az értéket fogjuk integrálni egy egyszerű elsőrendű diszkrét integrátort alkalmazva, hasonlóan az eddigiekhez. A legfelső bitet vesszük alapul (bitmaszkolunk), ha annak értéke 1, akkor a lehető legnagyobb értéket vonjuk le az integrátor bemeneti értékéből, különben nullát. Az így előálló jelet komparáljuk nullához, ekkor az ***out*** kimenetünkön előálló jel a modulátor kimenete, mely már az FPGA lábra kerül ki.

A fejlesztő maga paraméterezheti fel a Σ/Δ DA átalakítókat, ehhez a használt órajel frekvencia, a bemenetre kerülő szám összes bitszáma és törtrész bitjeinek száma szükséges.

3.3.1.5 Felhasználói interfész (HMI)

Ahogy azt a 3.1 fejezetben is említettem, az FPGA-ra implementált HIL szimulátorhoz nem olyan általános feladat felhasználói interfészt készíteni, mint a DSP alapú vezérlőegységhez. Alapvetően nincsen olyan belső kommunikációs periféria, ami közvetlenül használható lenne a számítógéppel való kommunikációra, mint a DSP, DSC vagy MCU-k esetén. Egy ilyen kommunikációs hardver implementációja több erőforrást venne el, mint amennyi a hozzáadott értéke.

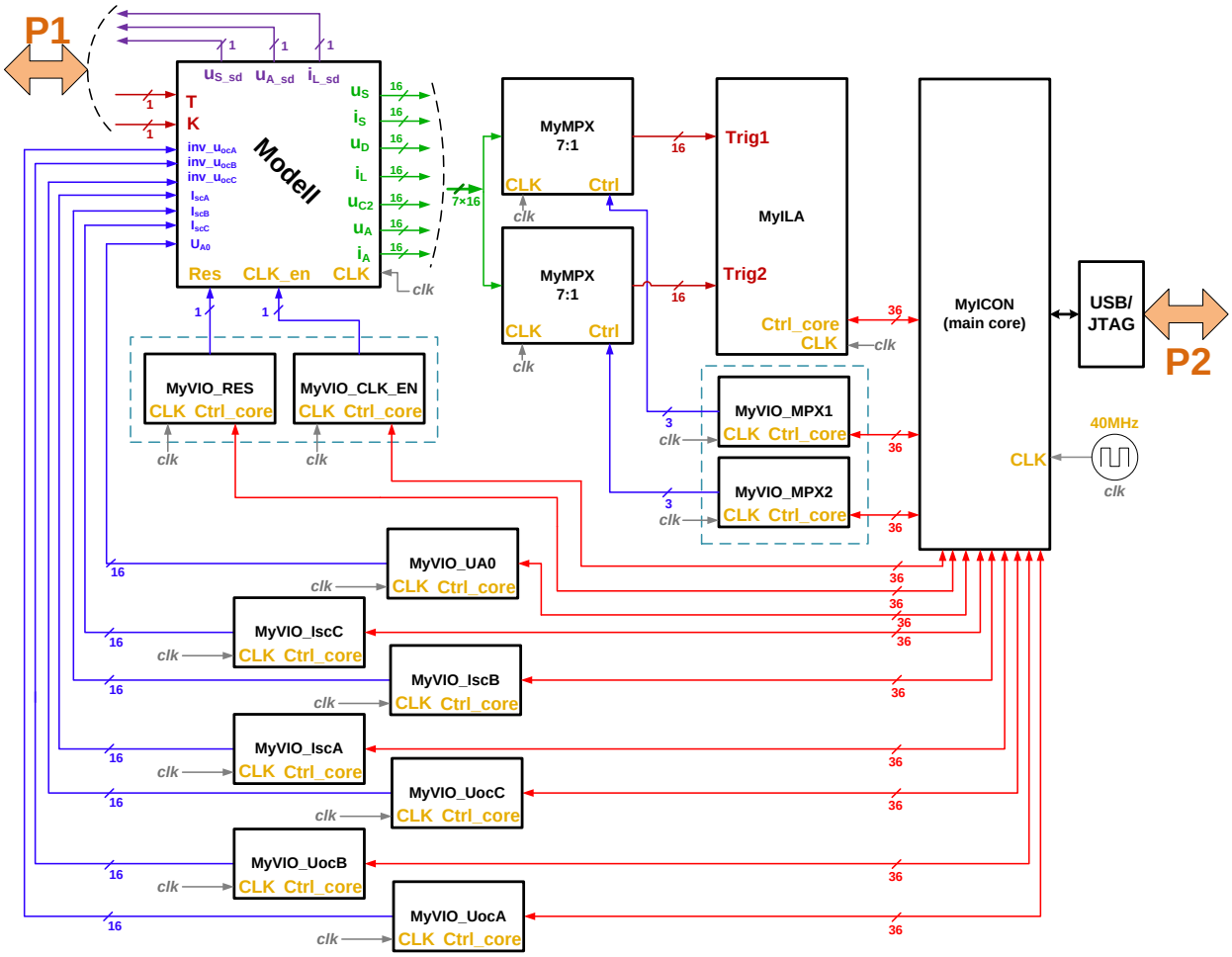
A Xilinx ChipScope Pro nevű alkalmazása az ISE Design Suite környezetbe ágyazható eszköz, amely már egy kész szolgáltatás felületét és lehetőségeit biztosítja számunkra, természetesen az FPGA-ra implementált saját hardverünkhöz magunknak kell definiálnunk az egyes jeleinkhez az egyes funkciókat még „HDL szinten”.

A metódust a következőképpen tudjuk elképzelni: az egyes HDL nyelven implementált modulok azon portjait, melyeket állítani vagy monitorozni szeretnénk, speciális feldolgozó modulokhoz kötjük. Ezen modulokat pedig egy fő modulhoz kötjük, mely az FPGA kártya USB/JTAG–jén keresztül kapcsolódik a ChipScope Pro felhasználói interfészén.

A speciális feldolgozó modulokat az egyszerűség kedvéért a beépített Core Generator programból IP (intellectual property) magokként generálhatjuk, a projekt keretén belül én is így tettem, sajnos ezeket még nem sikerült helyettesíteni a Matlab programból való kódgenerálással. Három típusát használtam a ChipScope Pro magoknak. Ezek voltak az ILA (Integrated Logic Analyzer), a VIO (Virtual Input/Output) és a minimálisan szükséges ICON (Integrated Controller) mag. Az utóbbi biztosítja az adatok útját a FPGA Boundary Scan portja és a ChipScope Pro magok között.

A szimulátorban a VIO magok segítségével szinkron módon tudjuk a 7 paramétert (lásd 3.2.1 táblázat) állítani a ChipScope Pro programból. Az ILA mag olyan, mintha egy több csatornás oszcilloszkóp lenne, melynek a mérőfejeit az FPGA-n belülre tetszőleges pontra tehetnénk.

A 3.3.14 ábrán látható **P1** és **P2** portok lényegében az FPGA kártya két interfésze a külvilág felé. A **P1** az interfész a DSP kártyához. A **MyICON** létesít kapcsolatot az összes ChipScope Pro mag és az USB/JTAG között, ami a **P2** porton keresztül kapcsolódik a számítógéphez, ami fizikailag egy USB interfész.



3.3.14 ábra: A ChipScope magok helye és szerepe

A **MyVIO_RES** és **MyVIO_CLK_EN** segítségével tudja a felhasználó állítani a Reset-et és engedélyezni az órajelet az egész modell számára.

A **MyVIO_UocA**, **MyVIO_UocB**, **MyVIO_UocC segítségével** az egyes napelem panelek üresjárási feszültségének reciprokát, a **MyVIO_IscA**, **MyVIO_IscB**, **MyVIO_IscC** segítségével az egyes panelek rövidzárási áramát tudja a felhasználó állítani. A **MyVIO_UA0** segítségével a felhasználó az akkumulátor feszültség paraméterét állíthatja be.

Az eddig nem említett magok mind a jelek monitorozásáért felelősek. A 7 darab 16 bites monitorozandó jelek közül mindig kettőt veszünk, egyszerre egy jelet vizsgálunk és hozzá kell egy referencia jel is. Ehhez nyújt segítséget a két 7:1 –es multiplexer, melyek kiválasztó értékeit szintén VIO magokon keresztül adhatjuk meg. Tehát a **MyVIO_MPX1** és **MyVIO_MPX2** magok és a két multiplexer segítségével a felhasználó kiválasztja a vizsgálandó jeleket, melyek így a **MyILA** magra kerülnek. Ez fel van készítve a jelek fogadására, a ChipScope Pro programból pedig a felhasználó állítani tudja az egyes trigger feltételeket.

3.3.3 Az FPGA kihasználtsága

Az imént ismertetett Simulink modellből a HDL kódgenerálást és a ChipScope magok IP magokként történő generálását követően az implementálás következett a *Xilinx ISE Design Suite* programban. A szintetizálást és implementálást követően az ISE Report-ja alapján a hardver az FPGA erőforrásainak csak kis részét használja ki. A legjelentősebb a felhasznált szorzókat tartalmazó DSP slice-ok 93%-os kihasználtsága, ez amiatt van, hogy a blokk RAM-ok címzéséhez szükséges indexek képzésekor, illetve az állapotváltozók számításakor nagyszámú szorzást hajtunk végre, melyhez ez a típusú Spartan 6-os FPGA DSP48A1s típusú slice-okat használ fel.

A flip-flopok-nak és latch-eknek 7%-os, a LUT slice-oknak 13%-os a kihasználtsága. Blokk RAM-ból 9 darab 16 bites, és 3 darab 8 bites generálódott, ami ezen típusú elemek esetén rendre csupán 28%-os és 4 %-os kihasználtságot jelent.

Összességében elmondható, hogy a szimulátornak nincs kifejezetten nagy hardver igénye. A kitűzött 40 MHz-es számítási periódusidő teljesíthető volt, és még egészen 70 MHz-ig növelhető lenne, tehát a sebesség nem korlátoz. A nagyszámú DSP slice használata csökkenthető lenne a használt belső szorzók multiplexálásával. Ehhez ütemezni kellene, hogy mely órajel ciklusban mely műveletek hajtódjanak végre, ehhez azonban a modell felépítésén alapjában véve módosítani kellene.

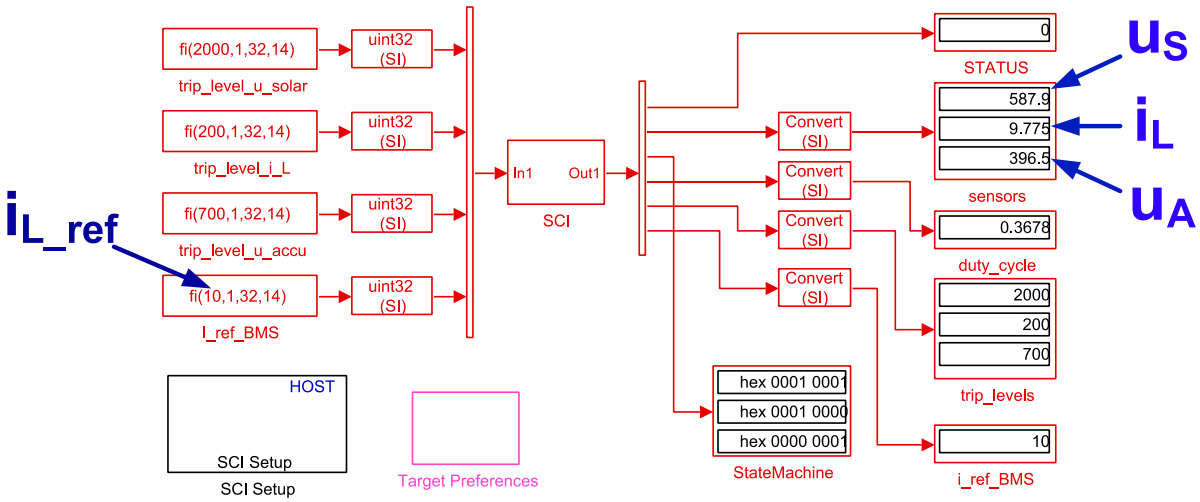
3.4 Mérési eredmények

A mérések természetesen az elkészített HMI-k segítségével történtek, tehát a vezérlőegységé soros kommunikációval ellátott Simulink alapú HMI, a HIL szimulátoré pedig a bemutatott ChipScope Pro alapú volt.

A HIL szimulátor paramétereinek beállítását a következő értékekkel tettem meg:

$$U_{ocA}{}^{-1} = \frac{1}{600}\,\frac{1}{V} \qquad I_{scA} = 20\ A \qquad U_{A0} = 400\ V$$
$$U_{ocB}{}^{-1} = \frac{1}{400}\,\frac{1}{V} \qquad I_{scB} = 20\ A \qquad \text{Reset} = 0 \qquad (3.4.1)$$
$$U_{ocC}{}^{-1} = \frac{1}{300}\,\frac{1}{V} \qquad I_{scC} = 10\ A \qquad \text{clk enable} = 1$$

Tehát a napelemes rendszer karakterisztikájában az eredő üresjárási feszültség a legnagyobb a 3 közül, ami 600V, az eredő rövidzárási áram pedig a 3 összege, ami 50A.

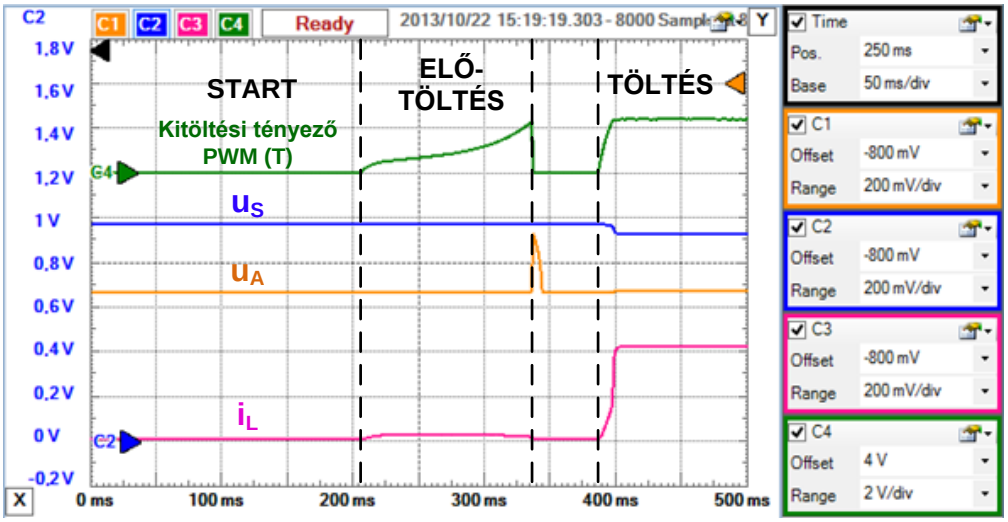


3.3.15 ábra: A vezérlőegység HMI felülete

A 3.3.15 ábrán a vezérlőegység HMI-je látható, melyben kiemeltem a lényegi részeket: a jobb oldalon azt a 3 értéket látjuk, melyeket a HIL szimulátorból a Σ/Δ DA átalakítókon, az aluláteresztő szűrőkön és a DSP AD konverterein keresztül kap meg a szabályozásért felelős rész

a DSP-ben. A másik fontos kiemelt rész az, ahol a referenciát tudjuk állítani az $\mathbf{i_L}$ –hez tartozó áramszabályozó számára.

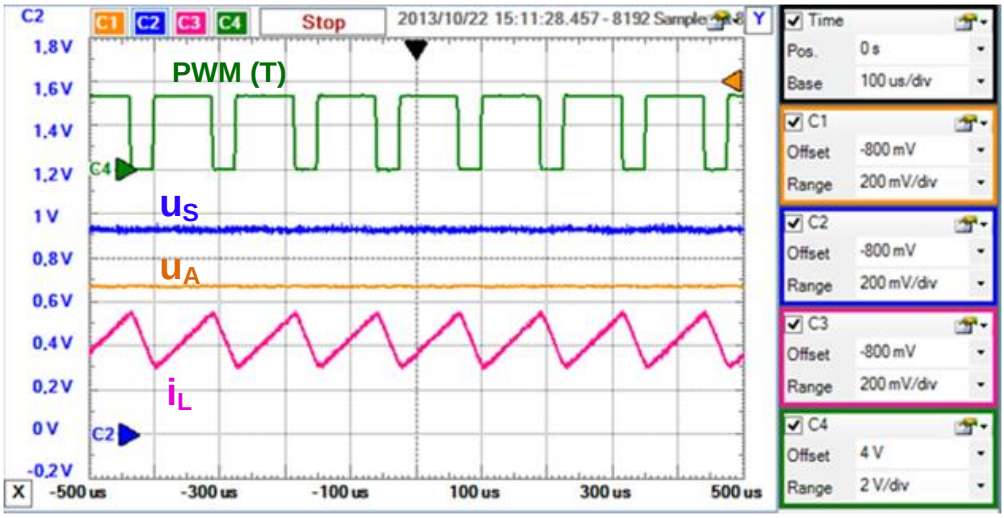
Mivel a DSP kártya és az FPGA kártya egy *Digilent* intelligens próbapanelen lett illesztve, így lehetőség van a próbapanel szolgáltatásait felhasználva a 3.3.16 ábrán lévő jelek ily módon való megjelenítésére. Ekkor figyelembe kell vennünk, hogy itt a teljes jeltartományt a 0 - +3.3V jelenti. Az egyes jelekhez tartozó osztásokat jobb oldalon találjuk.



3.3.16 ábra: A bekapcsolási folyamat

A 3.3.16 ábrán igen jól látszik a helyes működés. A $\mathbf{C_2}$ kondenzátor előtöltésekor kis $\mathbf{i_L}$ fojtóárammal dolgozunk, az $\mathbf{u_{C2}}$ feszültséget az áramszabályozó becsüli. Míg nem folyik az akkumulátorban $\mathbf{i_A}$ áram, addig az akkumulátor feszültség megegyezik az $\mathbf{U_{A0}}$ értékkel. Amikor a becsült érték az akkumulátor feszültségnél nagyobb lesz, akkor a vezérlőegység bekapcsolja a DC kontaktort. Ekkor összekapcsoljuk a $\mathbf{C_2}$ kondenzátort az akkumulátorral, ez látszik a 3.3.16 ábrán, amikor az előtöltést követően az $\mathbf{u_A}$ jelben történik egy feszültségugrás. A kontaktor a **K**=1 vezérlőjelre azonnal kapcsol, mivel az ívfeszültségén kívül nem vettük figyelembe a mechanikai késleltetését, ami most már nem jelentene gondot a modell kiegészítésével. Mindezek után megindul a töltési folyamat. Mivel az előtöltés csak kis árammal történik, így a napelemes táplálás $\mathbf{u_S}$ feszültség értéke láthatóan nem igen csökken le, azonban a nagyobb árammal történő akkumulátortöltés esetén jól látható a lecsökkenés. A napelemes rendszer eredő karakterisztikáján ugye a kisebb feszültségértékhez tartozik a nagyobb áram, melyre a konverternek a töltés véghezviteléhez van szüksége.

A következő ábrákon jól látszik a feszültségcsökkentő DC/DC konverter helyes működése. A 3.3.17 ábrán az áramszabályozó megfelelően nagy referenciaértéket kapott, így a fojtótekercs áramának jól ismert háromszög alakja jól követhető a **T** PWM vezérlőjel alapján.

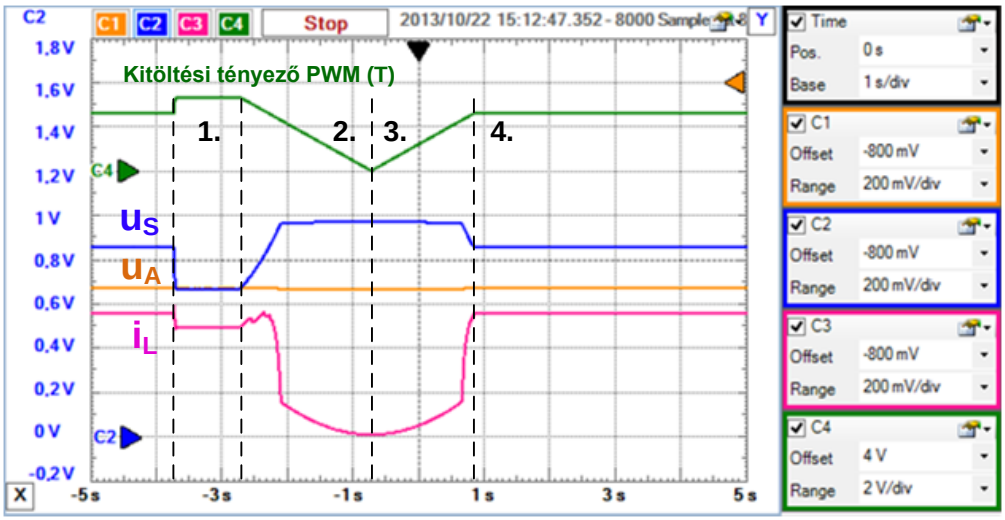


3.3.17 ábra: A buck konverter folytonos áramvezetésben

Abban az esetben, ha a szabályozó túl kicsi, de nullánál nagyobb áram referenciaértéket kapott, akkor a konverter szaggatott áramvezetésbe megy át. Ez látható a 3.3.18 ábrán, ekkor a fojtó áramának csak a középértéke nagyobb, vagy egyenlő, mint zérus, pillanatértékben nem mindig nagyobb, mint nulla, ezért hívják szaggatott vezetésnek.

3.3.18 ábra: A buck konverter szaggatott áramvezetésben

Ahogy a napelemes táplálású rendszereknél ez megszokott, jelen rendszerben is implementálva van maximum teljesítmény pont megkeresési (MPPT) algoritmus. Erre azért van szükség, mert ha nagyobb teljesítményt szeretnénk kivenni a napelemes táplálásból, mint amit az - az adott körülményekből adódóan - tud szolgáltatni, akkor a maximális teljesítmény ponton való működtetés a megoldás.



3.3.19 ábra: Az maximum teljesítmény pont keresés folyamata

A 3.3.19 ábrán látható a rendszerben egy MPPT algoritmus lefutása. A töltéshez egy nagyságrenddel nagyobb áram referenciát adunk a szabályozónak, mint amekkorához tartozó teljesítményt biztosítani tudna a napelemes táplálás. Ha szabályozó ezt nem tudja megvalósítani, akkor egy idő után az MPPT állapotba ugrik (**1.** szakasz). A **T** PWM jel kitöltési tényezőjét 100%-tól 0%-ig adott lépésközzel változtatja, és közben a mért teljesítményt eltárolja (**2.** szakasz). Hogy elkerüljük a kitöltési tényező értékének ugrásából adódó áramtüskét, ugyanilyen biztonságos meredekséggel olyan értékre állítja be a kitöltési tényezőt, melyhez a legnagyobb teljesítmény érték tartozott (**3.** szakasz). Amíg nem adunk új áramreferenciát, addig ebben a pontban tartja a kitöltési tényező értékét.

3.5 Továbbfejlesztési lehetőségek

3.5.1 Felhasználói interfész

Az egyik fő cél az volt, hogy lehetőleg a teljes HDL kódot a Matlab/Simulink programból lehessen generálni, ez sajnos a ChipScope Pro magok esetén nem teljesült, bár maga az FPGA-hoz tartozó HMI készítése is új feladat volt, és sikeresen teljesítve lett. Mivel generálhatóak ezen magok Simulinkből, csak megfelelő beállítás, és a blokk RAM létrehozásánál tapasztalt kiegészítéshez hasonló trükk kell hozzá, így a feladat a továbbiakban megoldásra kerül.

3.5.2 Napelemes rendszer

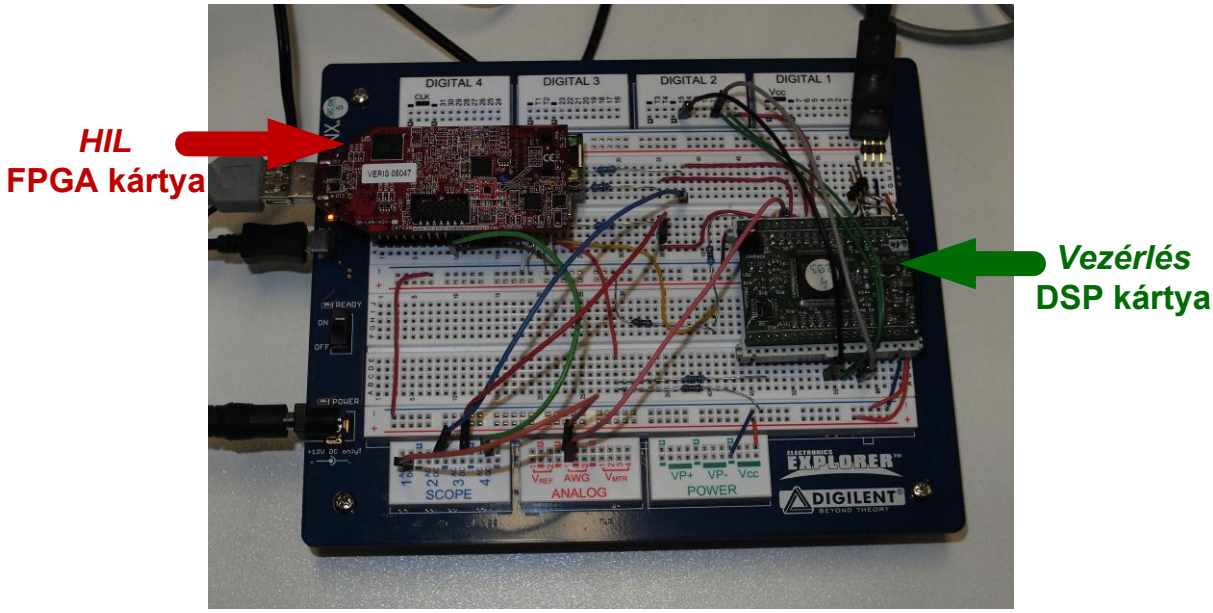
Mint láthattuk, még abban az esetben is elég komplikált az eltérő, illetve különböző hőmérsékletű vagy különbözőképpen megvilágított napelem panelek modellezése, amikor csak 3 darab panel párhuzamos kapcsolásáról beszélünk, és a jellemzők csak közvetett állítása lehetséges. Annak megoldása egy igen érdekes feladat és nagy kihívást jelent, hogy a felhasználó több konkrét típusú panelt tetszőlegesen összekapcsolhat, és mindegyikre külön állíthatja a konkrét jellemzőit. Mindezt futási időben, tehát azután, hogy a karakterisztikákat legeneráltuk, és a blokk RAM-okban eltároltuk.

3.5.3 Nemlineáris fojtótekercs

Az **L** tekercs esetén lineáris fojtótekercset feltételeztem, pedig tipikusan ez nem jellemző. Nemlineáris tekercs esetén az **L** induktivitás függ a rajta átfolyó áramtól: $L(i_L)$. Ha adott a nemlineáris függvény, akkor azt előre számítjuk és a napelem panel karakterisztikájához hasonlóan egy blokk RAM-ban tároljuk, így az állapotváltozó számításakor figyelembe vehetjük.

4 Értékelés

Az általam létrehozott HIL szimulátor működése megfelelő, sikerült elérni a kitűzött célokat. Az eszköz 3.5 fejezetben leírt továbbfejlesztéseit mindenféleképpen meg szeretném oldani a későbbi projektek elősegítésének érdekében is.



4.1 ábra: A megvalósított rendszer egy intelligens breadboard-on

A TDK dolgozatomban azon kívül, hogy bemutattam egy konkrét HIL szimulátor elkészítését, szerettem volna egy általános képet is adni ezen a téren, és egy fejlesztési módszert ismertetni. A HIL szimulátor egy olyan tesztelési metódust segít elő, mely a teljesítményelektronikai fejlesztések nagy százalékában igen komoly segítséget nyújthat a fejlesztők számára, így azt gondolom, hogy hasznos ezzel foglalkozni, és kutatást végezni e témában.

5 Irodalomjegyzék

[1] Tarek Ould Bachir, Jean-Pierre David, Christian Dufour, Jean B´elanger: Effective FPGA-based Electric Motor Modeling with Floating-Point Cores (978-1-4244-5226-2/10/$26.00 ©2010 IEEE)

[2] Handy Fortin Blanchette, Member, IEEE, Tarek Ould-Bachir, Member, IEEE, and Jean Pierre David, Member, IEEE: A State-Space Modeling Approach for the FPGA-Based Real-Time Simulation of High Switching Frequency Power Converters (IEEE Transactions on Industrial Electronics, Vol. 59, No. 12, December 2012)

[3] Tamás Kökényesi, István Varjasi Dr.: FPGA-Based Real-Time Simulation of Renewable Energy Source Power Converters (Journal of Energy and Power Engineering, ISSN 1934-8975, USA, Jan. 2010, Volume 4, No.1 (Serial No.26))

[4] Bin Lu, Member, IEEE, Xin Wu, Member, IEEE, Hernan Figueroa, Student Member, IEEE, and Antonello Monti, Senior Member, IEEE: A Low-Cost Real-Time Hardware-in-the-Loop Testing Approach of Power Electronics Controls (IEEE Transactions on Industrial Electronics, Vol. 54, No. 2, April 2007)

[5] Napelemek, Dr. Mizsei János, Timárné Horváth Veronika, BME EET hallgatói segédlet, 2003. október 15.

[6] Grid Conneceted Photovoltaic Power Systems: Survey of Inverter and Related Protection Equipments, Task V., Report IEA-PVPS T5-05: 2002, 2002.XII.

[7] Kökényesi Tamás: FPGA alapú valósidejű szimulátor megújuló energiaforrások átalakítóinak modellezéséhez (TDK dolgozat, 2010)

[8] E. Janssen, A. van Roermund, *Look-Ahead Based Sigma-Delta Modulation*, Analog Circuits and Signal Processing, DOI 10.1007/978-94-007-1387-1_2, © Springer Science+Business Media B.V. 2011