



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

# Korszerű akkumulátoros energitároló és töltő fokozat FPGA alapú Hardware-In-the-Loop szimulátorának tervezése

TDK dolgozat

Szabó Péter

Konzulens: Debreceni Tibor

Debreceni.Tibor@aut.bme.hu

Automatizálási és Alkalmazott Informatikai Taszék

Dr. Balázs Gergely György

gergely.balazs@siemens.com

Siemens Zrt.

Budapest, 2015.10.26.

## Köszönetnyilvánítás

Ez úton is szeretném megköszönni konzulensemnek, Debreceni Tibornak a tervezés és megvalósítás közben nyújtott segítségét.

Szeretnék köszönetet mondani Dr. Balázs Gergely Györgynek, hogy TDK munkám során állandó bejárást és betekintést engedett a Siemens Zrt. teljesítményelektronika laboratóriumába.

Valamint köszönettel tartozom Bihari Gábor kollégámnak, aki szakmai észrevételeivel és építő jellegű kritikáival hozzájárult a munkám elkészüléséhez.

## Kivonat

A teljesítményelektronikai berendezések tervezése és gyártása korunk egyik legdinamikusabban fejlődő ipari ágazata. A félvezető elemekből egyre növekvő kapcsolási frekvencián kinyerhető, egyre nagyobb teljesítmények, másrészt az egyre komplexebb szabályozási feladatok mind kutatási, mind fejlesztési irányokat és kihívásokat nyújtanak a villamosmérnökök számára. Emellett azonban tény, hogy az ipar gyors eredményeket vár, még a frissen megjelent technológiák, és vezérlési struktúrák esetében is, ami pozitív húzóerején kívül komoly kihívásokat is jelent a kutató- és fejlesztőmérnököknek egyaránt.

A fejlődés hatása és az egyre összetettebb igények legfőképpen a több kilo- és megawattos teljesítményű átalakítók esetében figyelhető meg. Ilyen teljesítmény szinteken az emberi és anyagi biztonság kulcsfontosságú a fejlesztés és tesztelés teljes folyamata során. Ennek érdekében az átalakítókat többrétű védelemmel és komplex szabályozási körökkel látják el a biztonságos, és hatékony működés elérése érdekében.

A vezérlő egységek tervezése rendszerint párhuzamosan zajlik a teljesítmény fokozat (főkör) tervezésével. A modern vezérlőket korszerű beágyazott rendszerekkel oldják meg, amik tipikusan DSP vagy mikrokontroller alapúak. Legfőképpen a nagy-teljesítményű átalakítók esetében igaz, hogy a főköri elemek egyedi tervezése szükséges, így a vezérlőegység - szinte áramköri bonyolultságától függetlenül - jóval hamarabb a fejlesztő, illetve tesztelő kezébe kerül.

Míndezek tekintetében, a vezérlő egység tesztelésére kínálkozó lehetőségek közül egyre népszerűbb a Hardware-In-the-Loop (HIL) szimulátor alkalmazása, melyben a főkör modellje egy dedikált központi egységre implementált. Ezzel a módszerrel valós időben ( $\sim 10 - 100$  nsec-os felbontású lépésközökkel) futtathatjuk a főkör szimulációját, ami periodikusan kiszámolja az állapotváltozók értékét, és interfészeinek megfelelő kialakításával elérhető, hogy azonos jelszintekkel kapcsolódjon a vezérlő egységhez, mint a valódi főkör esetén. A szabályozók hangolása valós időben végezhető le, a főkör valós idejű paraméterezésével, illetve belső változóinak monitorozása mellett. A HIL szimulátor használatával a vezérlő egység tesztelésekor lehetőség nyílik olyan extrém hibaállapotok előállítására is, amik a valódi főkör esetén nehezen, vagy egyáltalán nem valósíthatók meg, azonban a specifikáció

rendelkezik ilyen állapotok kezeléséről (pl.: fáziszárlatok, rövidzár védelmek).

TDK dolgozatomban egy ilyen FPGA alapú HIL szimulátor tervezését valósítottam meg, melyben a kezdetben megcélzott szimulációs lépésköz 100 nsec volt. A szimulátorban megvalósításra került egy korszerű lítium-ion akkumulátor cellákat tartalmazó akkumulátor pakk modell, és a hozzá kapcsolódó akkumulátor töltő főkörének modellje.

A szimulátort egy Xilinx Artix-7 FPGA-ra szintetizáltam, melyhez a HDL kódot MATLAB/Simulinkből generáltam a modell megtervezését és offline szimulációkkal történő verifikálását követően. A vezérlőegység szerepét egy ST Microelectronics által gyártott 32 bites mikrokontroller fejlesztőkártya látja el.

## Abstract

The design and manufacturing of power converters is one of the most dynamically developing industrial sectors of today. The increasing switching frequency and power of the semiconductors and the more complex structures of converter control demands give research and development directions and challenges for the engineers. Besides that, the fact is the industry expects rapid results, even from the newest technologies and control structures, which ensures a positive driving force and big challenges at the same time.

We can inspect the impact of development in case of power converters in the power range of kilo- and megawatts. In this range the human and material safety is the number one concern in the development and testing phase, as well. In order to maintain such safety and efficiency aspects, the converters have built-in more level and sort of protection, and advanced controller functions.

The development of these control units are usually parallel with the main development of the power stage (main circuit). Such modern control units are designed in form of advanced DSP- or micro-controller-based embedded systems. Especially for the high-power converters, it is true that the main circuit elements cannot be gained off-the-shelf, so custom design and manufacturing are needed. Therefore the control unit - almost independently from its circuit complexity - takes much less time to be in hand of the designer and the test team.

According to all the above mentioned aspects, applying of Hardware-In-the-Loop (HIL) simulator is the most favourite from the opportunities of testing the control unit, where the main circuit model is implemented in a dedicated central computational unit. With this method we can run real-time simulations ( $\sim 10$  -100 nsec step time), in which the state variables are periodically calculated and with a proper interface design, we can achieve connections with the control unit on the same signal levels, as with the real main circuit.

The subject of my TDK paper is the development of an FPGA based HIL simulator, in which the targeted step time is 100 nsec. The realized simulator can simulate a battery pack built-up by advanced lithium-ion battery cells and the related charger converter with its main circuit.

The simulator is realised on a Xilinx Artix-7 FPGA, where the HDL code was generated from MATLAB/Simulink after the design and verification of the models in offline simulations. The controller of the charger system is represented by an ST Microelectronics 32-bit developer kit.

# Tartalomjegyzék

<b>1.</b>	<b>Bevezető</b>	5
<b>2.</b>	<b>Elméleti áttekintés</b>	8
2.1.	Korszerű lítium-ion akkumulátor cella modellek	8
2.2.	Numerikus integrálási eljárások	15
<b>3.</b>	<b>A megvalósított HIL szimulátor</b>	21
3.1.	A tesztkörnyezet felépítése	21
3.2.	A vezérlőegység	21
3.3.	Az FPGA fejlesztőkártya[29]	22
3.4.	Szintillesztés	22
<b>4.</b>	<b>A modellezett rendszer</b>	24
4.1.	A rendszer blokkvázlata	24
4.2.	Vezérelhető, feszültségcsökkentő DC/DC átalakító	24
4.3.	Akkumulátoros rendszer	25
4.3.1.	Kontaktoros főköri fokozat	25
4.3.2.	Akkumulátor pakk	26
4.4.	A rendszer jelei, működése és leíró egyenletei	28
<b>5.</b>	<b>A Simulink modell</b>	31
5.1.	Számábrázolás	31
5.2.	A modell legfelső szintű leírása	34
5.3.	A Simulink modell részeinek leírása	37
5.3.1.	A bemenetek szinkronizálása	37
5.3.2.	Szinkron DC/DC feszültség átalakító Simulink modellje	37
5.3.3.	A DC kontaktor Simulink modellje	40
5.3.4.	A cabling Simulink modellje	42

5.3.5.	Az akkumulátor pakk Simulink modellje . . . . .	42
5.3.6.	A $\Sigma/\Delta$ modulátor Simulink modellje . . . . .	44
5.4.	A felhasználói felület (HMI) . . . . .	44
5.5.	A mérési eredmények . . . . .	46
<b>6.</b>	<b>Értékelés . . . . .</b>	<b>50</b>



# Ábrák jegyzéke

1.1. Hardware-In-the-Loop szimulátor általános felépítése . . . . .	6
2.1. Akkumulátor cella általános, n-ed rendű villamos helyettesítőképe . . . . .	9
2.2. Akkumulátor modell (Internal Resistance) . . . . .	12
2.3. Akkumulátor modell (One Time Constant) . . . . .	13
2.4. Akkumulátor modell (Two Time Constant) . . . . .	13
2.5. Numerikus integrátor . . . . .	19
2.6. Előrelépő Euler módszer . . . . .	20
2.7. Előrelépő Euler módszer megvalósított modellje . . . . .	20
3.1. HIL rendszer blokkvázlata . . . . .	21
3.2. Másodfokú aluláteresztő szűrő . . . . .	22
3.3. Szintillesztő kapcsolás (5 V/3 V) . . . . .	23
4.1. Az akumulator töltő rendszer blokkvázlata . . . . .	24
4.2. DC/DC konverter kapcsolási rajza . . . . .	24
4.3. Kontaktoros főköri fokozat kapcsolási rajza . . . . .	25
4.4. Kontaktoros főköri fokozat modelljének kapcsolási rajza . . . . .	26
4.5. Lítium-ion akkumulátor üresjáratú feszültsége a töltöttség függvényében . . . . .	27
4.6. A rendszer jelei és referenciairányai . . . . .	28
5.1. Simulink modell legfelső szintje . . . . .	36
5.2. Bemenetek beléptetése . . . . .	37
5.3. DC/DC átalakító Simulink modellje . . . . .	38
5.4. IGBT hídág a $V_L$ segédváltozó kiszámításához . . . . .	38
5.5. IGBT késleltetések állapotgépe . . . . .	39
5.6. Az $i_L$ állapotváltozó számítása . . . . .	39
5.7. A $V_{Charger}$ állapotváltozó számítása . . . . .	40

5.8. A DC kontaktor Simulink modellje . . . . .	40
5.9. A $V_{switch}$ segédváltozó számítása . . . . .	41
5.10. A főköri vezetékezés $R - L$ helyettesítőképek modellje . . . . .	42
5.11. A kombinált akkumulátor cella helyettesítőkép modellje . . . . .	43
5.12. A $\Sigma/\Delta$ modulátor Simulink modellje . . . . .	44
5.13. Chipscope-pro magok és a modell viszonya . . . . .	45
5.14. CC-CV töltés jelleggörbéi . . . . .	47
5.15. A kontaktor mechanikai késleltetései . . . . .	47
5.16. A töltési ciklus . . . . .	48
5.17. CV töltés és a töltési ciklus vége . . . . .	49

# 1. Bevezető

## Hardware-In-The-Loop szimulátorok a teljesítményelektronikában

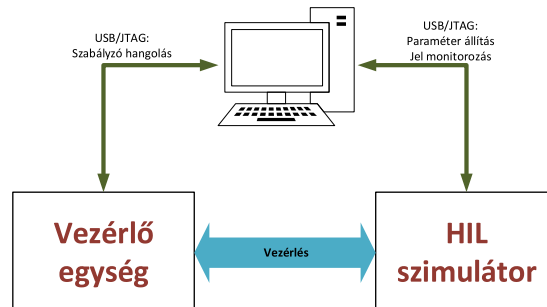
A mai villamos iparban nagy hangsúlyt kapnak a korszerű teljesítmény átalakítók, energiatárolók. Épp emiatt nagy igény mutatkozik azok fejlesztésére és minél nagyobb volumenű gyártására. Ezeknek az eszközöknek a fejlesztése viszont több problémát is rejt magában.

Rendszerint a korszerű, nagy teljesítményű energiatárolók és átalakítók megfelelő működtetéséhez nem elegendő a főkör megtervezése, ezek mellé szükség van szabályzók és vezérlő egységek implementálására is. Ez a két egység rendszerint egymástól nagyon eltérő munkafolyamatokat és tervezési készségeket von maga után, épp emiatt külön munkacsoportok dolgoznak rajtuk párhuzamosan. Ez a párhuzamos munkavégzés viszont azzal jár, hogy a két egység nem azonos időben készül el, rendszerint előbb rendelkezésre áll a vezérlő, mint hozzá maga a teljesítmény fokozat. Az így keletkezett holtidő semmiféle hasznot nem termel a fejlesztőknek, hisz nem tudják az elkészített eszközüket hangolni, tesztelni, illetve validálni[1].

Egy másik probléma szintén a tesztelésből fakad. A mai komplex szabályzókat már nem lehet, illetve nem érdemes analóg módon diszkrét áramköri elemekből felépíteni, ezek rendszerint egy beágyazott rendszert képeznek, melyek központi egységében (pl.: DSP,MCU) fut a szabályzó kódja[2]. Mivel ezek a szabályzók nagy teljesítményű rendszereket irányítanak, fontos a megbízható működésük. Épp ezért egy elgépelés a forráskódban komoly meghibásodásokhoz, túlmelegedésekhez, esetleg egyes alkatrészek felrobbanásához vezethetnek. Ez különösen is igaz a mai akkumulátoros energiatároló rendszerek esetében, mivel ezekben lítium alapú akkumulátor cellákat használnak, melyek nem megfelelő körülmények között súlyos károkat okozhatnak, súlyosabb esetekben emberi sérülésekhez vezethetnek.

Az ilyen és ehhez hasonló problémák kiküszöbölésére megoldást nyújt a vezérlő rendszerek szimulációval történő tesztelése. A közép- és nagy teljesítményű energiatárolókkal és átalakítókkal foglalkozó iparágakban az egyik legelterjedtebb szimulációs eljárás a

Hardware-In-the-Loop (HIL) szimulátorok alkalmazása[3]. A teljesítmény elektronikában használatos HIL szimulátor általános felépítését mutatja az 1.1. ábra.



1.1. ábra. Hardware-In-the-Loop szimulátor általános felépítése

Ebben a szimulációs eljárásban létrehozuk a teljesítmény-átalakító modelljét valamilyen magas szintű nyelven, tipikusan MATLAB/Simulink, esetleg Labview környezetben. Miután létrehoztuk a modellünket, lehetőségünk van azt még implementálás előtt, a fejlesztőkörnyezetben ellenőrizni folytonos idejű, lebegőpontos számábrázolást használó (offline) szimulációval, szükség esetén javíthatjuk a nem megfelelően működő modellünket[4]. Amennyiben ezek az offline tesztek sikerrel záródtak, a modellünk készen áll valamilyen tesztkörnyezeten történő implementálásra.

A megfelelő tesztkörnyezet kiválasztásakor figyelembe kell vennünk több tényezőt. Mivel ezekben a rendszerekben az állapotváltozók száma, melyeket egyszerre kell kiszámítani igen nagy (a rendszer komplexitásától függően 10+ különálló mennyiségről beszélhetünk), a műveletek számítás igényesek. Ezen kívül figyelembe kell vennünk, hogy a teljesítményelektronikai rendszerekben a vezérlés PWM segítségével valósul meg. Ezeknek a jeleknek a frekvenciája igen magas, tipikusan 2-150 kHz lehet[5]. Mivel egy HIL szimulátor legfőbb célja a rendszer szabályzástechnikai szempontokból való modellezése, ezért fontos ezeknek a kapcsolófrekvenciás összetevőknek a szimulálása is, hisz ezek zajként jelennek meg a szabályzási körben, amire a vezérlő egységet fel kell készítenünk. Ahhoz, hogy ezeket a több kHz-es jeleket modellezni tudjuk, igen kis lépésközöket kell választani a szimuláció futtatásához.

Mindezeket figyelembe véve jó megoldást kapunk, ha rendszerünket egy FPGA-ra szintetizáljuk. Ehhez rendelkezésünkre áll a Simulink részeként HDL coder, ami az általunk

elkészített modellt fordítja át az FPGA-ra szintetizálható HDL (VHDL vagy Verilog) nyelvre.

Az FPGA-k nagy előnye a HIL alkalmazásokban, hogy a műveletek párhuzamos lefutásúak, így megoldható, hogy a szimuláció lépésköze 10 – 100 nsec nagyságrendben legyen[6, 7]. Mivel manapság egyre erősebb hardverek érhetőek el, így kézenfekvő lenne, hogy a pontosság javítása érdekében még kisebb lépésközzel működő szimulátorokat tervezzünk. Mindezek ellenére nem feltétlenül előnyös a lépésköz túl kis értékűre választása. Hajtásrendszerekben, a rendszer mechanikai időállandói nem közelítik meg a korábban leírt 2-150 kHz-es frekvenciákat, így a szimulációkhoz nincs szükség igen kis lépésközök használatára.

A teljesítmény-átalakítók vezérlői legtöbb esetben analóg, vagy az ipari környezetre leginkább jellemző, nagy zaj immunitással rendelkező szigma-delta ( $\Sigma/\Delta$ ) modulált jeleket várnak. Az utóbbi esetben a  $\Sigma/\Delta$  modulációt végző IC-k funkcióját kell megvalósítani FPGA-ban, amik tipikusan 10-20 kHz-cel működnek. Ekkor viszont még a lépésköz csökkentésével sem áll rendelkezésünkre több információ, így annak csökkentése feleslegessé válik. Abban az esetben, ha a vezérlő analóg jeleket vár, a komplexitás és hatékonyság szempontjából szintén a  $\Sigma/\Delta$  modulátor használata a legkedvezőbb, amivel az FPGA-ban bitsorozatot hozunk létre, majd a nagyfrekvenciás tartományba kitolt kvantálási zajt egy másodrendű szűrő segítségével elnyomjuk[8].

További előnye ennek a szimulációs eljárásnak, hogy megfelelő számítógépes interfészeknek köszönhetően a modellünk paramétereit valós időben változtathatóak és minden jelet szintén valós időben követhetünk, azokat nem csak utólag vizsgálhatjuk.

Legnagyobb előnye a HIL szimulációnak a többi tesztelési eljárással szemben viszont az, hogy többek között a valóságnak a lehető leginkább megfelelő körülmények és állapotok hozhatóak létre. Figyelembe tudjuk venni a főköri időállandókat és veszteségeket, a kapcsolóelemek és a főköri kontaktorok késleltetését, melyek szabályzástechnikai szempontokból nem elhanyagolhatóak.

A vezérlőegységek minél komplexebb tesztelésének és validációjának érdekében létrehozhatóak extrém állapotok is[9, 10], így megfigyelhető, és befolyásolható a szabályzók vészhelyzetben mutatott viselkedése is, ezzel növelve a teljes rendszer biztonságát.

## 2. Elméleti áttekintés

Ebben a fejezetben bemutatom azokat az alapvető ismereteket, melyek elengedhetetlenek voltak a munkám elvégzéséhez. Az akkumulátoros rendszer szimulációjához meg kell ismernünk az akkumulátor cellák villamos modellezésének alapjait. A szimuláció FPGA-n történő futtatásához szükségünk van diszkrét idejű integrátorok használatára, ahhoz, hogy a tervezés során a feladatnak leginkább megfelelő integrátort implementáljunk, meg kell ismernünk a legelterjedtebb numerikus integrátorokat.

### 2.1. Korszerű lítium-ion akkumulátor cella modellek

Az akkumulátor cellák szimulációjához megfelelő modellt kell választanunk. Ennek a modellnek pontos villamos helyettesítő képeket kell adnia az akkumulátor cellákról. A modellezést több szinten is meg lehet ejteni. A cellák elektrokémia modellezése esetünkben nem megfelelő, hiszen az a cellák fizikai kialakításához, az energiafelhasználás optimalizálásához elengedhetetlen[11, 12]. Ezek rendkívül komplex modellek, melyek rendszerszintű dinamikus viselkedést nem szolgáltatnak.

Lehetséges matematikai modellek használata, ezek viszont sokszor sztochasztikus megközelítést használnak, és bár képesek rendszerszintű viselkedés előállítására, semmiféle I-V karakterisztikát nem eredményeznek, melyek elengedhetetlenek az elektromos tervezéshez [13, 14].

E között a két szint között helyezkedik el az akkumulátor cellák villamos modellezése, mely a rendszer matematikai leírását használja, de az elektrokémiai hatásokat is figyelembe veszi, mint elektrokémiai és koncentrációs polarizációk[11, 15].

Mielőtt foglalkoznánk a cellák legnépszerűbb villamos helyettesítőképeivel, szükséges néhány alapvető fogalmat megismernünk. A szakirodalomban a cellák minőségi leírására több állapotjelző is elterjedt, ezek közül a legfontosabb a State-of-Charge (SoC). Az SoC nem más mint az akkumulátor cella töltöttségi szintje. Értékét rendszerint százalékban, vagy 0 – 1-közötti viszonyzámban adják meg. Ezzel az állapotjelzővel megadható, hogy az adott áramerhelés mellett a cella mikor fog lemerülni, azaz, hogy mikor csökken nullára a felhasználható vagy más névvel kisütési kapacitása. Az SoC fogalmával szorosan össze-

függ a Depth-of-Discharge (DoD), vagyis a cella merültségének mértéke. DoD segítségével megadhatjuk, hogy a névleges kapacitásból mekkora részt használtunk már el. Az SoC értéke kiszámolható a 2.1. képlet alapján,

$$SoC(t) = SoC_{init} - \frac{1}{Q_n} \int_0^t i_d(t) dt \quad (2.1)$$

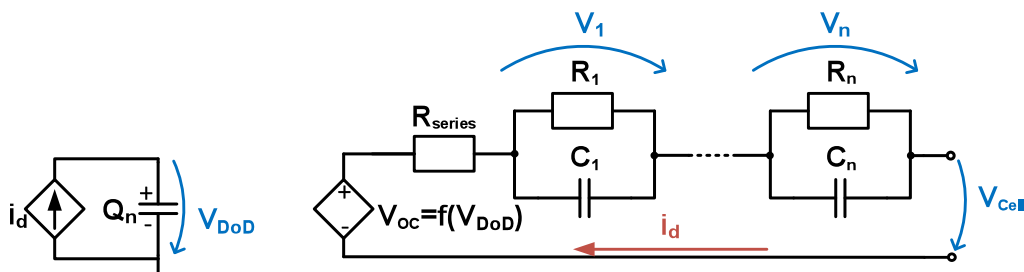
ahol  $Q_n$  az akkumulátor cella nominális kapacitása,  $i_d$  pedig az akkumulátor cella kisütő árama, feltételezve hogy értéke kisütéskor pozitív, töltéskor negatív.

Mivel mind az SoC, mind a DoD a cella töltöttségi állapotát írja le, ezek kifejezhetők egymásból, mégpedig a következőképpen,

$$DoD = 1 - SoC \quad (2.2)$$

feltételezve, hogy mind SoC, mind DoD a 0 – 1 tartományon van kifejezve. A szakirodal-  
 makban egyaránt használják az SoC és DoD kifejezéseket.

Az akkumulátorok villamos modellezéséhez megfelelő villamos helyettesítőképet kell választanunk. A gyakorlatban több főbb helyettesítőkép is elterjedt, mint a Thévenin[16, 17], impedancia[18], illetve futásidejű[19, 20] modellek. Az általam választott modell az ezek alapján létrejött kombinált modell[11], mely tartalmazza a Thévenin és futásidejű modellek részeit. Ezzel a megközelítéssel elérhetjük, hogy egyidejűleg jelezhetjük a cella futásidejű viselkedését, annak tranziensre adott válaszát és valós időben rögzíthetjük dinamikus, villamos karakterisztikáit, mint az üresjáratú feszültséget, vagy SoC-t[21]. Ennek a kombinált modellnek az általános alakját mutatja a 2.1. ábra. Az itt látható helyettesítő-



2.1. ábra. Akkumulátor cella általános, n-ed rendű villamos helyettesítőképe

kép két részre osztható. A kapcsolás baloldala határozza meg az akkumulátor élettartamát

(battery life time), ez a rész felel a modellben a cella DoD értékének a kiszámításáért[11]. Az ábrán látható  $i_d$  áram az akkumulátor cella kisütő árama, feltételezve, hogy ez az áramgenerátor  $i_d$  vezérelt és a  $Q_n$  kapacitást pontosan  $1V$ -ra tölti fel, ahol  $Q_n$  értéke a cella névleges kapacitása, akkor éppen DoD értékét kapjuk meg.

$$V_{DoD}(t) = \frac{1}{Q_n} \int_0^t i_d(t) dt + V_{DoD,init} \quad (2.3)$$

A helyettesítő kép jobb oldala adja a cella feszültség-áram karakterisztikáját. Ez a rész általános esetben felírható egy DoD függő, feszültségvezérelt feszültség generátorral, egy belső ellenállással és  $n$ -darab  $R - C$  taggal. A valóságban a modellel reprezentált összes mennyiség és paraméter több változó nemlineáris függését mutatja. Ezek SoC, hőmérséklet, State-of-Health (SoH), ciklusszám függőek. Szabályozástechnikai szempontból az akkumulátor nemlineáris és dinamikus viselkedése szignifikáns hiba nélkül modellezhető lineáris vagy statikus értékű paraméterezéssel is[11, 15]. Egy töltő rendszer modellezésekor például elhanyagolhatjuk a cellák önkisülési jelenségét, illetve az SoH miatt fellépő hasznos kapacitás csökkenését, hisz ez a töltési mechanizmus szempontjából irreleváns. A továbbiakban az egyszerű implementálhatóság érdekében a cellák paramétereit konstansnak feltételeztem.

Ezek alapján első lépésben határozzuk meg az állapotváltozók és azok deriváltjainak egyenleteit. Az első állapotváltozó a  $V_{DoD}$ , ez a 2.1. egyenletben látható. Ennek deriváltja, ha feltesszük, hogy  $V_{DoD,init} = 0$  a következőképpen néz ki.

$$\dot{V}_{DoD} = \frac{1}{Q_n} i_d \quad (2.4)$$

További állapotváltozók még az  $R-C$  tagok feszültségei. Ezek általános esetben a 2.5. egyenlet alapján írhatók fel.

$$V_n(t) = \frac{1}{C_n} \int_0^t \left( i_d(t) - \frac{V_n(t)}{R_n} \right) dt + V_{n,init} \quad (2.5)$$



Ekkor megint csak felhasználva, hogy  $V_{n,init} = 0$ , a deriváltak a következőképpen felírhatók,

$$\dot{V}_n = \frac{id - \frac{V_n}{R_n}}{C_n} = \frac{id}{C_n} - \frac{V_n}{\tau_n} \quad (2.6)$$

ahol  $\tau_n = R_n C_n$ .

Ekkor az n-ed rendű helyettesítőkép állapotterez leírása a 2.7. egyenletből látszik.

$$\begin{bmatrix} \dot{V}_{DoD} \\ \dot{V}_1 \\ \dot{V}_2 \\ \vdots \\ \dot{V}_n \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & -\frac{1}{\tau_1} & 0 & \cdots & 0 \\ 0 & 0 & -\frac{1}{\tau_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -\frac{1}{\tau_n} \end{bmatrix} \begin{bmatrix} V_{DoD} \\ V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} + \begin{bmatrix} \frac{1}{Q_n} \\ \frac{1}{C_1} \\ \frac{1}{C_2} \\ \vdots \\ \frac{1}{C_n} \end{bmatrix} i_d \quad (2.7)$$

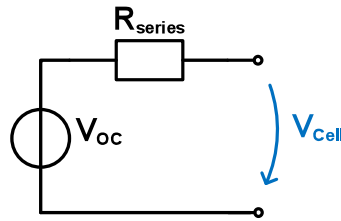
A rendszer válaszát nem lehet egyszerűen állapotteres alakban felírni, mivel az nem csak lineáris elemeket tartalmaz. Ezért a cella kapocsfeszültsége a 2.8. képlet alapján számolható.

$$V_{cell} = V_{oc} - R_{series} I_d - \sum_{i=1}^n V_i \quad (2.8)$$

Az így megfogalmazott általános leírás alapján a gyakorlatban három, a korábban bemutatott helyettesítőképre épülő modell terjedt el. Ezek mind a 2.1. ábrán látható n-ed rendű kapcsolás változatai, ahol n helyére megfelelő fokszámot helyettesítünk. Így a legelterjedtebbek a következők:

**Nulladrendű modell** Az n-ed rendű modell legegyszerűbb változata, az angol szakirodalomban Internal resistance (IR) modell. Ez nem tartalmaz semmiféle dinamikus viselkedést leíró elemet, az akkumulátor cellát csak egy belső ellenállással és feszültség forrással képezi le, kapcsolása a 2.2. ábrán látható.

A gyakorlatban ennek az elrendezésnek kétféle megközelítése is létezik. Egyszerűbb esetben Thévenin helyettesítőképként használjuk, ekkor feltételezünk egy olyan akkumulátor kapacitást, hogy az konstans feszültséggenerátorral közelíthető. Ez sok



2.2. ábra. Akkumulátor modell (Internal Resistance)

esetben elégséges megoldást adhat, de nem veszi figyelembe a cellák SoC-tól nemlineárisan függő üresjáratú feszültségét[15].

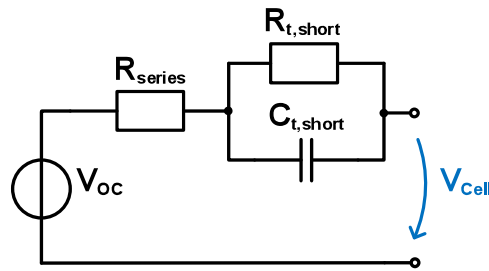
Másik megközelítése - mint esetünkben - a helyettesítőkép kombinált modellből való levezetése. Ekkor a konstans feszültségforrást egy vezérelt forrással helyettesítjük, mely SoC függő. Így a valóságos üresjáratú feszültséghez közelítő megoldást kapunk.

Ennek a modellnek előnye, hogy egyszerűsége miatt könnyen készíthető valósidejű szimuláció róla. Kevés elemet tartalmaz, így elérhető, hogy a szimulációk kis lépésközzel futhassanak a pontosabb modellezés érdekében. Másik nagy előnye a könnyű skálázhatósága. A paraméterek már korábban említett linearizációjával, nemlineáris függéseinek elhanyagolásával könnyedén pakk szintű modelleket alkothatunk.

Nagy hátránya viszont, hogy egyszerűségéből adódóan nem képes az akkumulátor cellák dinamikus viselkedésének pontos leírására. Mivel nem tartalmaz állapotváltozókat, ezért egységugrás jellegű áramterhelések hatására kapocsfeszültségében szintén ugrást tapasztalunk, ami valóságos cellák esetén nem áll fenn. Épp emiatt a dinamikus viselkedés figyelembe vételét igénylő, pontos szimulációkra ez a modell nem alkalmas.

**Elsőrendű modell** A nullad rendű modell kibővítésével, az első állapotváltozó beiktatásával kaphatjuk meg az elsőrendű, vagy One Time Constant (OTC) modellt.

Ez a modell már tartalmaz állapotváltozót, így annak dinamikus viselkedése jobban követi a valós viselkedést, mint a nulladrendű modell. Az  $R - C$  tag paramétereinek megfelelő hangolásával  $\tau_1$  időállandót a cella adott egységugrásra adott válaszána időállandójára hangolhatjuk, így egységugrás szerű terhelés ingadozásokkor a feszültségben nem fog ugrás jelentkezni, jelalakja jobban közelíti a valóságos jel-

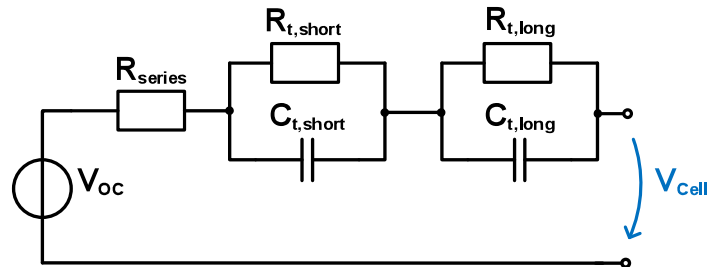


2.3. ábra. Akkumulátor modell (One Time Constant)

alakot, mint a nulladrendű modell esetében. További előnye még, hogy a modell összetettsége nem változott nagy mértékben, így még mindig könnyen lehet valósídejű szimulátorokban alkalmazni, bár skálázhatósága már kevésbé egyszerű, mint a nulladrendű esetben[21].

Hátránya, hogy a modell csupán egytárolós tagként közelíti a valóságos cellákat, így csak gyors tranziensek esetében ad pontosabb képet a nulladrendű modellnél.

**Másodrendű modell** Az első rendű modell további bővítésével kapjuk meg a két állapotváltozót tartalmazó, úgynevezett Two Time Constant (TTC) modellt.

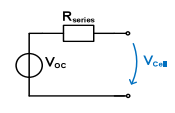
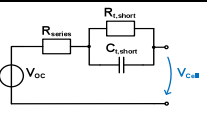
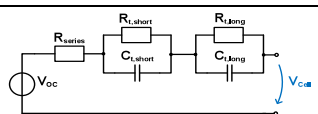


2.4. ábra. Akkumulátor modell (Two Time Constant)

Ennek a modellnek a legnagyobb előnye, hogy mivel két állapotváltozót is tartalmaz, azok időállandóinak megfelelő megválasztásával a terhelésben bekövetkező mind gyors, mind lassú tranzienst okozta ugrásokat is jól tudja követni, így a legtöbb alkalmazáshoz pontos szimulációk végezhetők rajta[22, 23].

Hátránya az eddig tárgyalt modellekhez képest, hogy összetettsége miatt nehezebb alkalmazni valósídejű alkalmazásokban, illetve nehézkes a skálázhatósága és bővíthetősége. Bár megvalósítása nehezebb ennek a modellnek, a korszerű szimulációs

eszközeinkben a szimuláció még mindig lehetséges, ugyanígy skálázható is, bár több erőforrást igényel az OTC és IR modellekhez képest. Pontossága viszont annyival jobb, mint a korábban tárgyalt modelleké, hogy erőforrás igénye ellenére az iparban ez a leginkább elterjedt modell[21]. A továbbiakban ezt a modellt használtam fel a szimulátor tervezése során.

Fokszám	Helyettesítőkép kapcsolása	Hatékonyság	Komplexitás	Skálázhatóság	Bővíthetőség	Identifikáció komplexitása
0		A	A	M	M	A
1		K	K	K	K	K
2		M	M	K	K	M

A = alacsony; K = közepes; M = magas

2.1. táblázat. Helyettesítő kapcsolások összehasonlítása

Mint látható az általános modell tetszőleges számú  $R - C$  taggal bővíthető, így tetszőleges számú, illetve bonyolultságú modell képezhető ezen az elven. Ennek ellenére a gyakorlatban nem használnak kettőnél nagyobb fokszámú modellt, mivel az erőforrás felhasználás és számítási igény tekintetében nem érhető el szignifikánsan kisebb hiba[11].

Bár azt várnánk ezektől a modellektől, hogy az egyre nagyobb rendű változatai egyre pontosabban követnék le egy valóságos cella viselkedését, ez viszont a valóságban nem feltétlenül következik be. Ahhoz, hogy a modellek pontosan kövessék a valóságot, identifikálni kell a modell paramétereit. Ezek a méretezések nagyszámú méréseket eredményeznek valódi cellákon. A mérések általában előre definiált, periodikus áramprofilú terhelések alapján végzett töltési-kisütési ciklusok, melyek alapján számíthatók időállandók, majd ellenállás és kapacitás értékek[15, 24]. Mivel ezeket a méréseket teljes töltésre és teljes kisütésre kell elvégezni, és általában egy paraméter kinyeréséhez több ciklus szükséges, nagyon időigényesek.

Az elkészített HIL szimulátor célja nem a cellák individuális modellezése, hanem annak pakkszintű dinamikus leírása volt. Ezt a viselkedést a cellaparaméterek identifikációja nélkül is megfelelően pontosan el tudtam végezni. Mint azt korábban is leírtam, a paraméterek konstans értékekkel való közelítésével csak kis hibát viszünk a rendszerbe és komplex identifikációs eljárások nélkül felhasználhatók az akkumulátor adatlap értékei.

Mivel a gyártók minden termékükön komoly méréseket végeznek, feltételezhetjük, hogy az általuk kiadott értékek és karakterisztika görbék helyesek, azok a modellezés során felhasználhatók. Ilyen módon könnyen megkaphatjuk pusztán az adatlapból a cellák  $V_{oc} - DoD$  függését és futásidejű viselkedését, mely nagy előrelépés az egyszerű Thévenin helyettesítőképekkel szemben.

## 2.2. Numerikus integrálási eljárások

Annak érdekében, hogy FPGA-n az offline szimulációkkal verifikált modelleket implementálhassuk, át kell alakítani azokat diszkrét idejűre. Meg kell említenünk, hogy az offline modelleket bár folytonosnak tekintjük, állapotváltozóik kiértékelése szintén diszkrét időben, periodikusan zajlik, mivel ezek a szimulációk számítógépeken futnak. A számítógépen futó szimulációk futásidejének optimalizálása érdekében ezeket a szimulációkat változó lépésközű megoldókkal végezzük el. Ezek képesek az állapotváltozók figyelésére és azok változási gyorsaságát figyelembe véve állítják a szükséges felbontást (step time). Lassan változó jeleket nagyobb lépésközzel számolunk, míg gyors változáskor a szimulációs szoftver (esetünkben MATLAB) érzékeli a változást és sűrűbb lépésközzel dolgozik.

Bár az FPGA-k is képesek ezekhez hasonló változó lépésközű megoldókat használni, azok bonyolultak és egy korszerű FPGA esetében implementálásuk felesleges, hisz elég erőforrás áll rendelkezésünkre fix lépésközű megoldók használatához. Ahhoz hogy a szimulációnkat valós idejű jelzővel illethessük, ahhoz fix lépésközű megoldás alkalmazása esetén igen kicsi, 10-100nsec-os nagyságrendű lépésközt kell elérni. Ezzel a módszerrel biztosíthatjuk, hogy minden jel periodikusan előálljon.

A diszkrét idejű integrálási problémák megoldásához numerikus integrálási eljárásokat használnak.

**Előrelépő Euler módszer** A legegyszerűbb numerikus eljárás az előrelépő, vagy explicit Euler módszer. Ez a módszer egy véges differencia módszer, mely segítségével egy függvény integráltja közelíthető. A módszer numerikus integrálásra a következőképpen vezethető le.

Tegyük fel, hogy a keresett függvényünk a következően néz ki,

$$y' = f(t, y(t)) \quad (2.9)$$

ahol  $t \in [0, b]$  és  $y$  kezdeti értéke  $y(0) = 0$ . Osszuk fel a  $[0, b]$ -tartományt  $n$  részre. Ekkor  $y'$  integráltja a tartomány  $t_i$  és  $t_i + h$  pontja között, ahol  $h = \frac{b}{n}$ , vagyis a numerikus számítás lépésköze, a következő,

$$y(t_i + h) = y(t_i) + \int_{t_i}^{t_i+h} f(t, y(t)) dt \quad (2.10)$$

Az integrál közelítésére használjuk a baltéglalap szabályt, majd az eredményt behelyettesítve a 2.10. egyenletbe, megkapjuk az Euler módszer explicit alakját[25].

$$\int_{t_i}^{t_i+h} f(t, y(t)) dt \approx hf(t_i, y(t_i)) \quad (2.11)$$

$$y(t_i + h) = y(t_i) + hf(t_i, y(t_i)) \quad (2.12)$$

Mint a 2.12. egyenletben látható, a módszer az integrál következő értékének kiszámításához mindig csak az előző értéket veszi figyelembe, emiatt ez a módszer könnyen implementálható valósidejű alkalmazásokban.

Hátránya ennek az eljárásnak, hogy könnyen instabillá válhat. Ez akkor következhet be, ha a vizsgált rendszer időállandóihoz képest túl nagy lépésközt állítunk be, ekkor egy egyszerű probléma megoldása is könnyen divergensé válhat. Ez a probléma kiküszöbölhető a számítás lépésközének megfelelően kis értékűre választásával.

**Hátralépő Euler módszer** Az előzőhöz hasonló módszer a hátralépő, vagy implicit Euler módszer. Ez a módszer a következőképp néz ki,

$$y(t_i + h) = y(t_i) + hf(t_i + h, y(t_i + h)) \quad (2.13)$$

Az egyenletet összehasonlítva a 2.12. egyenlettel láthatjuk, hogy a két módszer különbsége, hogy az előrelépő Euler módszer minden lépésben az előző pontban vizsgálja a függvény meredekségét ( $f(t_i, y(t_i))$ ), míg a hátralépő Euler módszer a keresett pontban teszi ugyan azt. Így pontosabbá tehető az integrál közelítése, de elveszti azt a tulajdonságát, hogy az egyenlet két oldalán mindig azonos pontban számolunk. Mivel ennél a módszernél az  $i + 1$ -edik pontban lévő értékhez ismernünk kell az  $i$ -edik pontbeli értéket és az  $i + 1$ -beli érintőt, épp ezért ezt nem lehet explicit módon kifejezni.

Ennek a módszernek a stabilitása kedvezőbb mint az explicit Euler esetében. Általánosságban azt szeretnénk, hogy a lépésközt előre meghatározott specifikációk szerint határozzuk meg, nem pedig a választott numerikus eljárás stabilitása alapján. Ezek alapján az olyan rendszereket, melyek megoldásához a specifikált lépésközhöz képest a stabilitás eléréséhez sokkal kisebb lépésközt kell választanunk stiff rendszereknek nevezzük[26]. Ezeknek a rendszereknek a megoldására jó módszer a hátralépő Euler módszer, mivel annak stabilitása kisebb mértékben függ a lépésköztől.

**Trapéz szabály** A harmadik széles körben elterjedt numerikus módszer a trapézsabály.

Legkönnyebben egy tetszőleges függvény határozott integrálja közelíthető a következőképpen,

$$\int_a^b y(t)dt \approx (b - a) \left( \frac{y(a) + y(b)}{2} \right) \quad (2.14)$$

Ilyen módon a függvény integráltját, vagyis a görbe alatti területet egy trapéz területével közelítjük. A módszer pontosabbá tehető, ha a  $t \in [a, b]$  tartományt felosztjuk  $N$  részre, és minden rész területre kiszámoljuk a hozzá tartozó trapéz területét, majd összegezzük azokat. Így általános esetben a trapéz szabály

$$\int_a^b y(t)dt \approx \frac{h}{2} \sum_{k=0}^N (y(t_{k+1}) + y(t_k)) = \frac{b-a}{2N} (y(t_0) + 2y(t_1) + \dots + 2y(t_N) + y(t_N)) \quad (2.15)$$

ahol  $h = \frac{b-a}{N}$ .

Ezzel a módszerrel a lépésköz ( $h$ ) megfelelően kicsire választásával pontosan közelíthető egy tetszőleges függvény integráltja, és stabilitása nem függ olyan mértékben a lépésköztől, mint az Euler módszer esetében.

Valósídejű alkalmazásokban való használatát nehezíti, hogy egy integráláshoz sok összegzést el kell végeznünk, így ezzel a módszerrel nem lehet gyors számításokat végezni.

**Runge-Kutta módszer** Az eddig említett módszerek közül a legkomplexebb és egyben a legpontosabb numerikus integrálási eljárás a Runge-Kutta módszer. Ezzel a névvel egy módszer családra szoktak utalni, illetve annak legelterjedtebb tagjára, a negyedrendű Runge-Kutta módszerre[26].

A módszer hasonlóan közelíti egy függvény integrálját, mint az implicit és explicit Euler módszerek. Ezek mind két pont között téglalapként tekintenek a függvény alatti területre, a különbség a három között, hogy az implicit (előrelépő) Euler a szakasz elején, az explicit (hátralépő) Euler a szakasz végén, míg a legegyszerűbb Runge-Kutta a szakasz közepén vizsgálja a helyettesítő téglalap magasságát[26].

A negyedrendű módszer is ezt az alapelvet használja fel, de a végső közelítéshez négyféle közelítés átlagát veszi. Ezek alapján egy tetszőleges függvény integráltja az  $N$  részre osztott  $t \in [a, b]$  szakaszon a következően néz ki,

$$\begin{aligned}
 y_k &= y_{k+1} + \frac{h}{6} \left( f(t_{k-1}, Y_1) + 2f\left(\frac{t_{n-1}}{2}, Y_2\right) + 2f\left(\frac{t_{n-1}}{2}, Y_3\right) + f(t_n, Y_4) \right) \\
 Y_1 &= y_{k-1} \\
 Y_2 &= y_{k-1} + \frac{h}{2} f(t_{k-1}, Y_1) \\
 Y_3 &= y_{k-1} + \frac{h}{2} f\left(\frac{t_{k-1}}{2}, Y_2\right) \\
 Y_4 &= y_{k-1} + hf\left(\frac{t_{k-1}}{2}, Y_3\right)
 \end{aligned} \tag{2.16}$$

ahol  $y' = f(t, y)$ ,  $t \in [a, b]$  és  $h = \frac{b-a}{N}$ .

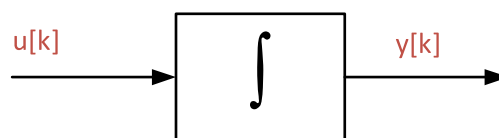


Bár ez a módszer a legstabilabb, a lépésköztől legkevésbé függ a stabilitása, valamint a négy módszer közül azonos lépésközök esetén a legpontosabb eredményt adja, a 2.16. egyenletből is látszik, hogy számítása igen komplex. Mivel egy pontban a módszer négy komplex számítást igényel, ennek a módszernek az alkalmazása nehézkes valósidejű rendszerekben.

A TDK munkám elvégzéséhez az előrelépő Euler módszert választottam. A cél a HIL szimulátor esetén az volt, hogy a rendszer futtatásakor a lehető legkisebb szimulációs lépésközt érzjük el, a lehető legegyszerűbben megvalósítható numerikus eljárást kellett választanom. Mivel a szimulátorban elkészült egy több elemből álló akkumulátor pakk modell, mely több integrátort is tartalmaz, figyelembe kellett venni, hogy ezek mind szintetizálásra kerülnek az FPGA-n, így nem lehetett nagy bonyolultságú integrátorokat használni.

Bár a megemlített numerikus eljárások közül a leginstabilabb módszer az előrelépő Euler, ennek hatása nem érvényesült esetünkben. Figyelembe kell viszont venni, a töltő fokozatban megvalósított kapcsoló üzemi feszültségcsökkentő konvertert is, hiszen ennek jelei 10 kHz nagyságrendűek. A tervezés során a megcélzott lépésköz 100 nsec volt, ami így is nagyságrendekkel kisebb a rendszerben megtalálható kapcsolási frekvenciához tartozó 100 usec-hoz képest.

A kiválasztott matematikai módszerből létre kell hozni a numerikus integrátor modelljét a kiválasztott módszer egyenlete alapján.



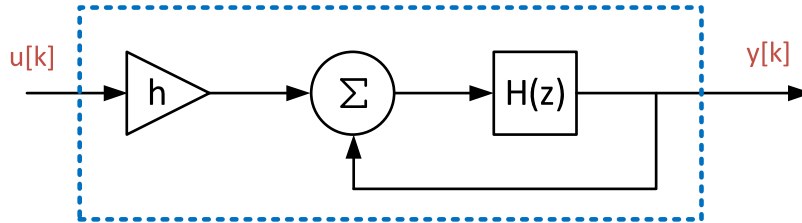
2.5. ábra. Numerikus integrátor

Ehhez felhasználjuk 2.12. egyenletet. Írjuk át ezt az egyenletet oly módon, ahogy azt a modellben is fel fogjuk használni,

$$y[k + 1] = y[k] + hu[k] \quad (2.17)$$

ahol  $u[k]$  a bemenet,  $y[k]$  a bemenet integráltja és  $h$  a szimuláció lépésköze. Ezek alapján

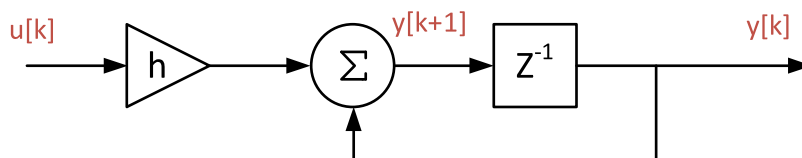
láthatjuk, hogy az integrál következő lépésben előálló értéke meghatározható az adott lépésben az integrál kimenetének és a bemenet lépésközzel súlyozott összegeként.



2.6. ábra. Előrelépő Euler módszer

A 2.5. képen látható doboz tartalma kibontva látható a 2.6. ábrán. Láthatjuk hogy az összegző bemenetén előáll a szükséges  $hu[k]$  és  $y[k]$  is, így a 2.17. egyenlet alapján az összegző kimenetén megjelenik az integrál következő lépése. Hogy ez a megfelelő módon kerüljön ki a kimenetre, vagyis a bemenet és a kimenet azonos ütemben legyen, szükségünk van a jel egy lépésközzel való késleltetésére. Szerencsére ez egy FPGA-n könnyen kivitelezhető. Ezek alapján a következő megvalósítást kapjuk,

$$H(z) = z^{-1} \tag{2.18}$$



2.7. ábra. Előrelépő Euler módszer megvalósított modellje

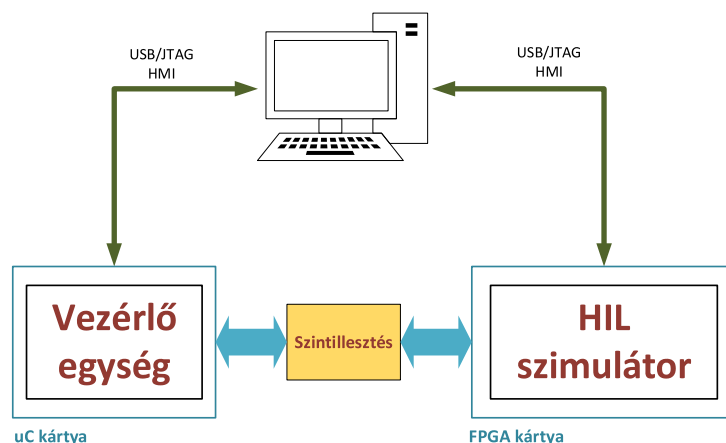
Mint a 2.7. ábrából látszik az Euler módszerrel nagyon kevés komponens felhasználásával képesek vagyunk numerikus integrátort konstruálni, mely kis stabilitása ellenére megfelel az általunk támasztott követelményeknek.

### 3. A megvalósított HIL szimulátor

#### 3.1. A tesztkörnyezet felépítése

A tesztkörnyezet egy mikrokontrolleres vezérlőegységből egy Artix-7 Xilinx FPGA-t tartalmazó fejlesztői kitből, és egy hozzá tervezett viszonylag nagy számú és többféle konfigurációt engedő interfészkartyából, továbbá egy laptopból áll. A laptophoz mind a vezérlő mind a az FPGA kártya USB/JTAG interfésszel csatlakozik USB kábelen, melyen egyrészt a szabályozó- másrészt a HIL paraméterezésére és monitorozására alkalmas HMI-k (felhasználói interfészek) futnak.

Mivel a vezérlést megvalósító kártya és az interfész kártya jelszintjei nem egyeztek meg, szükség volt egy egyszerű szintillesztés beiktatására. A teljes rendszer blokkvázlata a 3.1. ábrán látható.



3.1. ábra. HIL rendszer blokkvázlata

#### 3.2. A vezérlőegység

Ez a kit egy 32 bites lebegőpontos számábrázolást biztosító MCU-t tartalmaz, melyen a töltési protokollt és a szabályozókat megvalósító kód került implementálásra. Ez az egység a Siemens Zrt. egy korábbi projektjének keretein belül készült el, melyet biztosítottak számomra a TDK munkámhoz.

### 3.3. Az FPGA fejlesztőkártya[29]

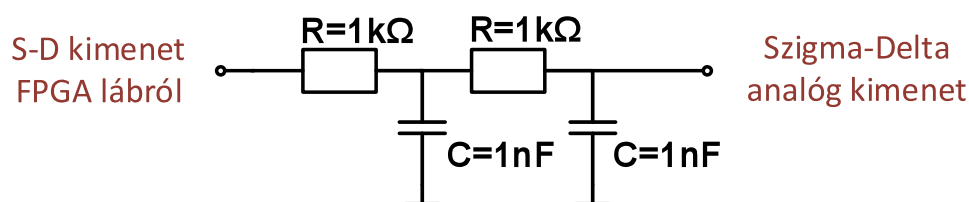
A szimulátor megvalósításához egy a Xilinx cég által gyártott Nexsys 4 fejlesztőkártyát használtam. Ezen a kártyán található egy Artix-7 FPGA modul, mely több mint 15000 logikai cellát tartalmaz és akár 450 MHz-es maximális órajellel képes működni.

A szimulátor tervezésekor a cél a 100 nsec-os lépésköz, vagyis a 10 MHz órajel elérése volt. Ez az FPGA kártya tökéletesen megfelelt a kitűzött céloknak, valamint elegendő teljesítménnyel rendelkezik a modell további bővítéséhez, fejlesztéséhez.

### 3.4. Szintillesztés

A megvalósított rendszerben az FPGA alapú szimulátor és a vezérlést megvalósító vezérlő kártya nem közvetlenül kapcsolódik egymáshoz, ennek több oka is van.

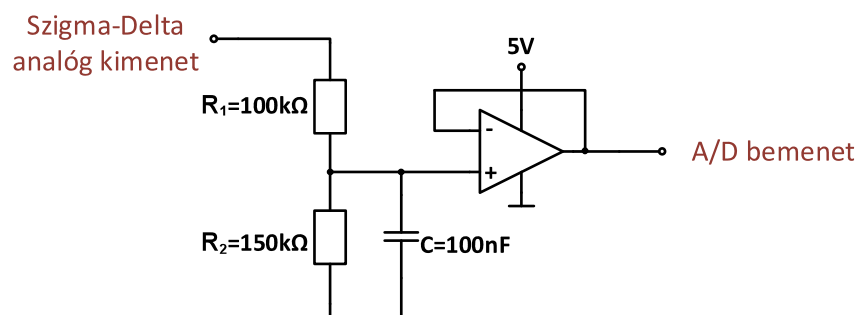
A vezérlőkártyánk, mint a valós akkumulátoros rendszer esetében is analóg jeleket vár a bemenetén. Ezeket a jeleket viszont előtte elő kell állítanunk, hisz a szimulátor fixpontos, digitális jelekkel dolgozik. Mivel az FPGA fejlesztőkártyánk nem tartalmaz analóg kimeneteket, ennek megoldására implementáltam az FPGA-ban a már korábban is említett szigma-delta modulátort (megvalósítása az 5.3.6. fejezetben látható). Ennek köszönhetően egy jel előállításához fizikailag csak egy lábat használtunk fel, viszont ezzel a módszerrel az FPGA lábán egy bitstream-generálunk, amit megfelelő szűréssel analóg jellé tudunk alakítani.



3.2. ábra. Másodfokú aluláteresztő szűrő

A 3.2. ábrán látható másodfokú aluláteresztő szűrő segítségével kiszűrhetjük a  $\Sigma/\Delta$  nagyfrekvenciás kvantálási zaját. Az így konstruált másodfokú szűrő törésponti frekvenciája  $f = 159.15$  kHz. A vezérlőkártyán 10 kHz frekvenciájú A/D átalakítót használunk, így annak bemenetére egy megfelelően szűrt analóg jel kerül.

Az interfész kártyán lett az előbb leírt aluláteresztő szűrő implementálva, így annak értékei adottak voltak. Mivel ez a kártya 5 V-os tápellátásról működik, így a  $\Sigma/\Delta$  kimenetének értékei is 5 V-os értéktartományban mozognak, viszont a vezérlőkártyán található A/D átalakító referencia feszültsége 3 V, így a két kártya között szintillesztésre volt szükség. Ezt az illesztést egy egyszerű feszültségosztóval oldottam meg.



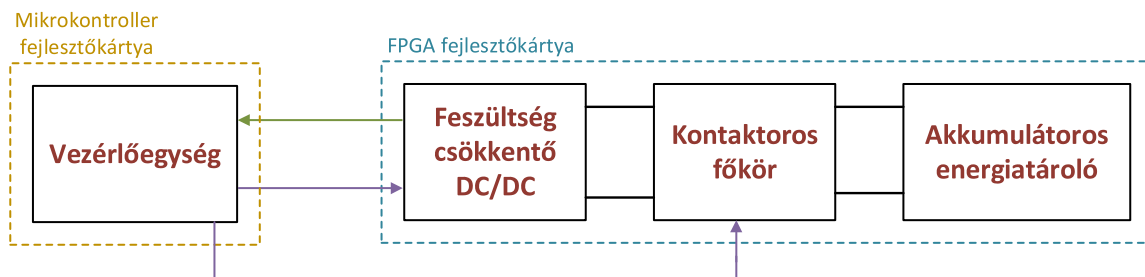
3.3. ábra. Szintillesztő kapcsolás (5 V/3 V)

A feszültségosztó kimenetére egy követőerősítő került, így nem terheljük meg az osztót az A/D átalakítóval, így az osztásarányt nem fogja módosítani a mikrokontroller bemenetének belső ellenállása és kapacitása.

## 4. A modellezett rendszer

### 4.1. A rendszer blokkvázlata

A HIL szimulátorban modellezett akkumulátor töltőfokozatban megtalálható egy vezérelt feszültségcsökkentő DC/DC átalakító, egy akkumulátor pakk és a kettőt összekötő főköri elemek modellje. Ezek blokkvázlata látható a 4.1. ábrán.

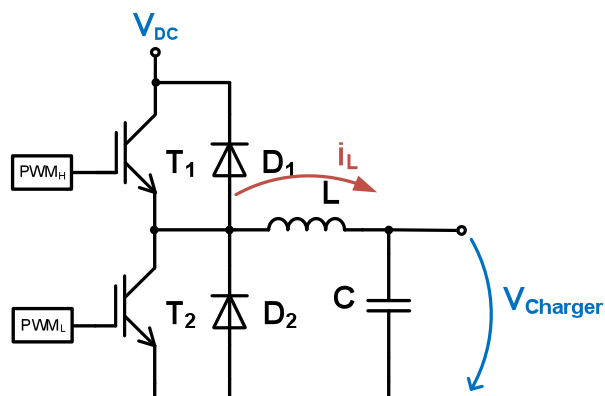


4.1. ábra. Az akkumulátor töltő rendszer blokkvázlata

A vezérlő kártya biztosítja az akkumulátorok töltéséhez szükséges feszültség és áram karakterisztikákat a feszültség átalakító kimenetén.

### 4.2. Vezérelhető, feszültségcsökkentő DC/DC átalakító

A töltőfokozat szimulálásakor nem foglalkoztam a betáplálás oldal kérdéseivel, hiszen ez lehet a fogyasztói hálózat, egy szünetmentes ipari tápegység kimenete, vagy egy üzemi nagyfeszültségű DC busz. Igaz azonban, hogy bármelyik modelljének kimenete feszült-



4.2. ábra. DC/DC konverter kapcsolási rajza

séggenerátoros jellegű, így a modell elkészítésekor feltételeztem  $V_{DC} = 600$  V egyenfeszültséget. Ahhoz, hogy a töltés megvalósuljon, ebből az egyenfeszültségből kell egy az akkumulátorok számára megfelelő feszültség szintet előállítanunk. Ennek a feszültségnek az előállítását látja el a DC/DC átalakító, mely egy szinkron feszültségcsökkentő konverter (szinkron buck). Ennek az áramkörnek a kapcsolási rajza látható a 4.2. ábrán.

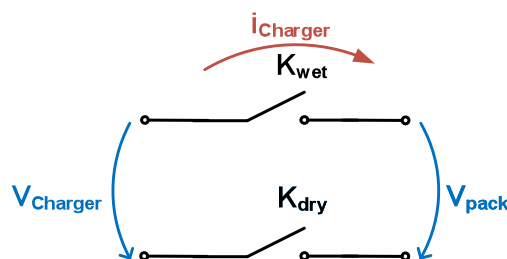
A kapcsolat két vezérelhető IGBT-t tartalmaz, ezeket megfelelő vezérlés hatására szinkronban, ellen fázisban üzemeltetünk. A vezérlőjelek ( $PWM_H$  és  $PWM_L$ ) magas-aktív jelek, vagyis magas logikai jelszinteken vezetnek az IGBT-k.

Megfelelő vezérlőjelek előállításával szabályozni tudjuk az  $L$  fojtótekercs  $i_L$  áramát, és ezáltal a  $C$  kimeneti kondenzátor töltőáramát illetve  $V_{charger}$  feszültségét. Mivel ezeket a kapcsolóelemeket szinkron üzemeltetjük, biztosítanunk kell, hogy azok egyidejűleg ne legyenek bekapcsolt állapotban, hisz ekkor rövidzár alakul ki és az így keletkezett áramot csak a tranzistorok belső ellenállása korlátozza. Ennek az állapotnak az elkerüléséért a vezérlőegység a felelős.

## 4.3. Akkumulátoros rendszer

### 4.3.1. Kontaktoros főköri fokozat

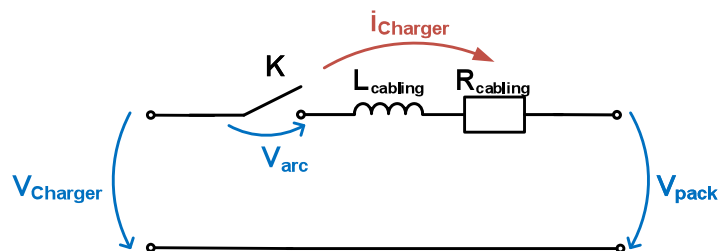
A töltőfokozat főköre tartalmazza azokat az elemeket, melyek összekötik a DC/DC konvertert az akkumulátor pakkal. Ennek a kapcsolását láthatjuk a 4.3. ábrán.



4.3. ábra. Kontaktoros főköri fokozat kapcsolási rajza

A rendszermodellben ennek a fokozatnak a modelljében vettem figyelembe a rendszerben található kábelezést, valamint két DC kontaktort, melyek a pakk biztonságos be- és lekapcsolást végzik. Ezek a kontaktorok vezérelhető eszközök, azok irányítását a vezérlőegység látja el.

A két DC kontaktor közül konvencionálisan a negatív sínen található Dry kontaktor nem kapcsol teljesítményt, azaz bekapcsolási szekvenciában ez az első. A pozitív sínen található kontaktor ellenben már terhelés alatt kapcsolja össze a DC/DC átalakítót az akkumulátor pakkal. A Dry kapcsolóelem a modell szempontjából elhanyagolható, bár a valóságos rendszerben megtalálható, modellünkben egy vezetékkel helyettesíthető. A modellben figyelembe kell venni még a két részt (töltő, akkumulátor) összekötő vezetőket is. Ezeket soros  $R - L$  helyettesítőképpel modelleztem. Ezek alapján a kontaktoros főkör modellezett kapcsolási rajza a 4.4. ábrán látható.



4.4. ábra. Kontaktoros főköri fokozat modelljének kapcsolási rajza

Mivel a kontaktor egy elektromechanikus elem, a modellezés során nem hanyagolhatjuk el annak be- és kikapcsolási késleltetését, valamint a kapcsoláskor fellépő ívhúzás jelenségét. Mivel bekapcsolás előtt nincs terhelés a DC buszon, így a bekapcsolási késleltetés, valamint ív kialakulása nem olyan lényeges, mint a kikapcsolási késleltetés és ívképződés. Ennek az ívnek a kialakulása több tényezőtől is függ, mint az ívhossz, áram és kontaktok anyaga[27]. Ennek modellezése rendkívül bonyolult, a modell kialakítása erősen függ az alkalmazott kontaktor típustól, hisz lehetséges olyan eset, tipikusan DC busz rövidzárlatkor, hogy az adott kapcsolóelem kialakítása miatt nem képes az ívkisülés kioltására[28]. Mivel TDK dolgozatomban nem a kontaktor validálása volt a fő feladat, annak ívfeszültségét ( $V_{arc}$ ) konstans értéként vettem figyelembe.

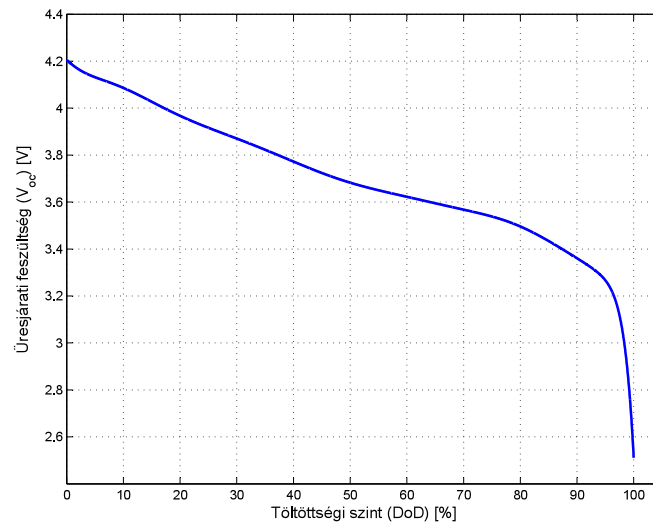
### 4.3.2. Akkumulátor pakk

Az akkumulátor cellákat tartalmazó pakk modell megalkotásához a 2.1. fejezetben tárgyalt helyettesítőképet használtam. Mint azt már korábban említettem, ez az elektromos helyettesítőkép tartalmaz lineáris és nemlineáris elemeket. A lineáris, vagyis a feszültség-



áram karakterisztikát meghatározó elemek modellezése a Simulink könyvtárban megtalálható standard elemekkel történt.

A nemlineáris üresjáratú feszültség modellezéséhez fel kellett használni a cellák adatlapjában megtalálható karakterisztikát (4.5. ábra). Ezt a karakterisztikát az FPGA-n való



4.5. ábra. Lítium-ion akkumulátor üresjáratú feszültsége a töltöttség függvényében

futtatáshoz diszkretizálni kell. Megfelelő felbontást kaptunk a karakterisztika 4096 pontra való felbontásával. Ezeket a diszkrét értékeket egydimenziós look-up-table-ben (LUT) tároltam. Ennek a tömbnek az indexelése DoD érték alapján történik, melyet az akkumulátor kisütő (vagy éppen töltő) áramának integrálásából kapjuk meg.

Ezzel a módszerrel elkészült cella modellekből kell létrehozni a pakk modelljét, ennek két lehetséges módja van. Általános esetben egy ilyen akkumulátor pakk  $N_{row}$  darab párhuzamosított cellából áll, melyek biztosítják a pakk megfelelő áramterhelhetőségét. Ahhoz, hogy az akkumulátor pakknak a kimeneti feszültsége a követelményeknek megfelelő legyen, ezeket a párhuzamosított cellákat sorosítanunk kell, általános esetben  $N_{series}$  darabbal számolva.

A modell készítés egyik lehetséges megoldása, ha feltételezzük, hogy a párhuzamosított cellák kontaktusa ideális, mind azonos SoH és DoD állapotokkal rendelkeznek, valamint belső ellenállásaik a gyártási szórás elhanyagolásával megegyeznek. Ekkor ezek összevonhatók egy cella modellbe, ha ellenállás paramétereiket  $1/N_{row}$ -val, kapacitás paramétereit

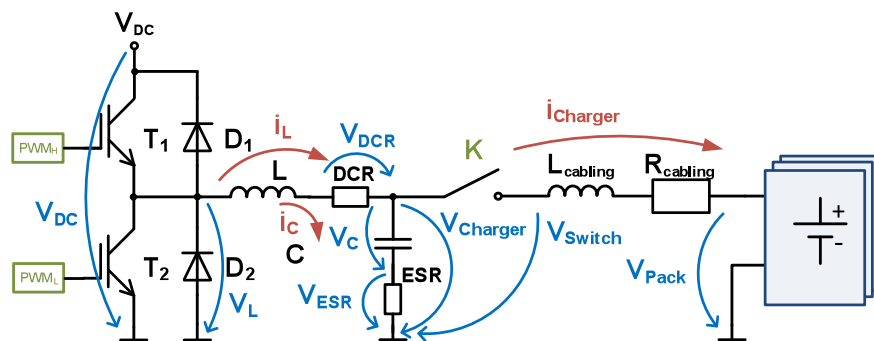
ket  $N_{row}$ -val beszorozzuk. Az így kapott párhuzamos cellákat  $N_{series}$ -szer példányosítva és sorba kapcsolva alakul ki a pakk modell. Ezzel a módszerrel egy nagyon erőforrásigényes modellt kapunk, hisz egy nagyteljesítményű rendszerben az akkumulátorok kapocsfeszültsége több száz volt, így egyszerre akár száz párhuzamosított cellát is le kell helyeznünk, melyek mind tartalmazzák a nemlineáris karakterisztika tábláját, mely nagy mennyiségű blokk RAM-ot igényel az FPGA-n.

A másik módszer segítségével kisebb erőforrás igényű modellt készíthetünk. Ehhez az előbbi módszer feltételezésein kívül feltételeznünk kell még, hogy minden cella kapacitása azonos és azonos árammal terheljük őket. Ebben az esetben nem vétünk nagy hibát, ha nem csak a párhuzamos, de a soros cella kapcsolatokat is összevonjuk egy  $N_{series}$  szorzótényezővé, mely reciprok kapcsolatban áll  $N_{row}$ -val.

Az elkészült szimulátorban én a második módszer szerint jártam el, mivel az első erőforrásigénye túl magas és a szimuláció pontosságát nem segíti elő, a HIL szimulátor célja a vezérlő egység tesztelése, melyhez az akkumulátor pakknak csak a dinamikus rendszerszintű viselkedése a lényeges, nem a pakkon belül az egyes cellák viselkedése. A továbbiakban azonban fejlesztési irány lehet a pakk modelljének átalakítása és a cellák rendszeren bellüli viselkedésének vizsgálata.

#### 4.4. A rendszer jelei, működése és leíró egyenletei

A rendszer Simulink modelljének kialakításához a teljes rendszert leíró állapotteres leírásra van szükség. Az ehhez szükséges jelek elnevezése és a referenciáirányok 4.6. ábrán láthatók.



4.6. ábra. A rendszer jelei és referenciáirányai

A  $V_{DC}$  értéke valós időben megadható paraméter, ez kerül a DC/DC átalakító bemenetére. Mivel az átalakító egy szinkron feszültségcsökkentő kapcsolás, annak IGBT-i felváltva vezetnek, így  $V_L$  értékét hol 0 V-ra, hol  $V_{DC}$ -re kapcsolja. Az így kialakult feszültség és  $V_{Charger}$  feszültség különbségeként előáll az  $L$  tekercsen eső feszültség, mely segítségével kiszámolható a tekercs  $i_L$  árama, mely a rendszerünkben állapotváltozó.

Az így kiszámolt  $i_L$  áramból megkaphatjuk a  $C$  kondenzátor töltőáramát, ezzel közvetten annak  $V_{Charger}$  feszültsége számolható. A feszültségcsökkentő kimenetén folyó áram középértéke állandósult állapotban meg fog egyezni  $i_L$  középértékével. Ezt az  $i_{Charger}$  áramot kapjuk meg a vezetékvezés  $L_{cabling}$  és  $R_{cabling}$  -re felírt egyenletek alapján, ahol a tekercsre kerülő feszültség a töltőfeszültség ( $V_{Charger}$ ) és a  $V_{pack}$  akkumulátorfeszültség és az ellenálláson eső feszültség összegének különbsége.

A tekercsek és a kondenzátor modellezésekor figyelembe vettem azok veszteségeit soros DCR és ESR ellenállásokkal.

Típus	Jelölés	Megjegyzés
Állapotváltozó	$i_L, V_{Charger}, i_{Charger}$	Állapotegyenletek eredményei
Segédváltozó	$V_{DC}, V_L, i_C, V_{switch}, V_{pack}, V_{DCR}, V_{ESR}$	Állapotegyenletek bemenetei
Kimenetek	$i_L, V_{Charger}$	$\Sigma/\Delta$ kimenetek, vezérlőkártyába
Monitorozás	$i_L, V_{Charger}, V_{switch}, i_{Charger}, V_{pack}, PWM_H, PWM_L$	Megfigyelés a HIL HMI-ben

4.1. táblázat. A rendszer jelei

Ezek alapján felírhatjuk az állapot egyenleteket.

$$V_L(t) - (V_{Charger}(t) + i_L(t)DCR) = L \frac{di_L(t)}{dt} \quad (4.1)$$

$$i_L(t) - i_{Charger}(t) = C \frac{dV_C(t)}{dt} \quad (4.2)$$

$$V_{Charger}(t) - (V_{pack}(t) + i_{Charger}(t)R_{cabling}) = L_{cabling} \frac{di_{Charger}(t)}{dt} \quad (4.3)$$

Ezeket az egyenleteket az FPGA-n belül periodikusan értékeljük ki numerikus integrálást használva, át kell alakítani őket az integrális alakjukra. Így a rendszer állapotegyenletei a következők,

$$i_L(t) = \frac{1}{L} \int_0^t V_L(t) - (V_{Charger}(t) + i_L(t)DCR) dt \quad (4.4)$$

$$V_{Charger}(t) = \frac{1}{C} \int_0^t (i_L(t) - i_{Charger}(t)) dt + (i_L(t) - i_{Charger}(t)) ESR \quad (4.5)$$

$$i_{Charger}(t) = \frac{1}{L_{cabling}} \int_0^t V_{Charger}(t) - (V_{pack}(t) + i_{Charger}(t)R_{cabling}) dt \quad (4.6)$$

Továbbá szükségünk van még az IGBT hídág kimenetén megjelenő  $V_L$  és a kontaktor kimenetén lévő  $V_{switch}$  egyenleteire, melyek

$$V_L = \begin{cases} V_{DC} & , PWM_H = 1 \\ V_{DC} & , PWM_L = 0 \quad \text{és} \quad i_L > 0 \\ 0 & , \text{egyébként} \end{cases} \quad (4.7)$$

$$V_{switch} = \begin{cases} V_{Charger} & , K = 1 \\ V_{pack} & , K = 0 \quad \text{és} \quad i_{Charger} < i_{threshold} \\ V_{Charger} - V_{arc} & , k = 0 \quad \text{és} \quad i_{Charger} > i_{threshold} \end{cases} \quad (4.8)$$

## 5. A Simulink modell

### 5.1. Számábrázolás

Mielőtt végigmennénk a rendszer Simulink-ben létrehozott modelljén, meg kell ismerünk az ott használt számábrázolást, annak metódusát. Az FPGA-n belül a lehető leggyorsabb működési sebesség elérése érdekében fixpontos számábrázolást használunk, mely segítségével az egyes mennyiségek összeszorzása kevesebb időbe telik, mint lebegőpontos esetben. Bár veszítünk a lebegőpontos számábrázolás pontosságából, de megfelelő bit-szám választása esetén ez a hiba minimalizálható.

Az ábrázolandó mennyiségeket három fő csoportba oszthatjuk, ezek közül a legelső a változók kezelése. Ebbe a csoportba tartozik minden áram és feszültség mennyiség. A második csoport az állapotegyenletek szorzótényezői, valamint a harmadik csoport a jelek összegzésére használt összeadók számábrázolása.

Az offline modellből történő fixpontosítást a MATLAB Fixed-Point Designer eszköz segítségével oldottam meg. A program által használt fixpontos számábrázolás alakja

$$fixdt(i, i + n + m, m) \quad (5.1)$$

ahol  $i$  az előjel bit,  $n$  az egész bitek és  $m$  a tört bitek száma. Ahhoz, hogy egy mennyiségről el tudjuk dönteni, hogy hány biten lehet ábrázolni ismernünk kell annak maximális és minimális értékét, valamint az általunk elvárt felbontást.

Az első csoportba tartozó változók fixpontosítása a következőképpen lehetséges,

$$i = \begin{cases} 1 & , u_{min} < 0 \\ 0 & , \text{egyébként} \end{cases} \quad (5.2)$$

$$n = \begin{cases} \lceil \log_2 (\max (|u_{max}|, |u_{min}|)) \rceil & , |u_{max}| \geq 1 \quad \text{és} \quad |u_{min}| \geq 1 \\ \lceil \log_2 (\max (|u_{max}|, |u_{min}|)) \rceil & , \text{egyébként} \end{cases} \quad (5.3)$$

$$m = \begin{cases} \lceil -\log_2(q) \rceil & , q < 1 \\ \lfloor -\log_2(q) \rfloor & , \text{egyébként} \end{cases} \quad (5.4)$$

ahol  $u_{min}$  a jel minimum,  $u_{max}$  a jel maximum értéke,  $q$  pedig az elvárt felontás. Az egész részek megállapításához minden esetben a jel legnagyobb abszolút értékének kell venni a kettes alapú logaritmusát, hisz az a szám lesz a bitek száma, amin ez az érték még ábrázolható. Amennyiben a legnagyobb abszolútérték egynél nagyobb, mindenképpen a logaritmus felső egész részét kell vennünk, ezzel biztosítva, hogy a jel teljes tartománya beleférjen a fixpontos alakba.

A törtrészek esetében hasonlóan jártam el, de mivel általános esetben a felbontás egy egynél kisebb szám, annak logaritmusá negatív lenne, ezért az egész értéket  $-1$ -gyel meg kell szoroznunk.

A második csoportba tartozó mennyiségek az állapotegyenletekben lévő szorzótényezők értékei. Ezeknek a fixpontosításához figyelembe kell venni az FPGA-ban található szorzó áramkörök kialakítását. Az általunk használt Artix-7 18x25 bites szorzókat használ, egy maximum 25 bites számot tud egy 18 bites szorzótényezővel gyorsan összeszorozni. Ahhoz, hogy ezt ki tudjuk használni, a szorzótényező alakjában az összes bitek számát fixáltuk, vagyis  $i + n + m = 18$ .

Mivel ezek a szorzatok esetünkben főleg integrál együtthatók, ezek értéke a lépésköz miatt kicsi (Euler módszer, lásd 2.2. fejezet), ezért nem tudjuk a korábban bemutatott fixpontosítást használni. Az integrálás során, ha nem választjuk megfelelő félbontásúra a szorzó ábrázolását, a vétett hiba akkumulálódik, ezzel elrontva az eredményt. Hogy ezt elkerüljük, az integrálás elvégzéséhez ki kell bővítenünk a számaábrázolás tartományát, majd az integrálás elvégzése után vissza alakíthatjuk az eredményt az eredeti formájára. Ezek alapján három részre kell felbontani a fixpontosítást, meg kell adnunk a szorzótényező alakját, hogy azt pontosan ábrázoljuk, a végső szorzat alakját, hogy a hiba ne akkumulálódjon, és az integráláskor használt összegző alakját.

Ezek közül nézzük először a szorzótényező felírását,

$$f = \begin{cases} \lceil \log_2(u) \rceil & , u > 1 \\ \lfloor \log_2(u) \rfloor & , \text{egyébként} \end{cases} \quad (5.5)$$

$$m = n - f - i \quad (5.6)$$

ahol  $f$  segédváltozó, az LSB logaritmusának kiszámításához,  $n$  az egészek,  $m$  a tört bitek,  $i$  pedig az előjel bitek száma. Mivel a szorzótényező egy konstans szám, így annak nincs külön maximum és minimum értéke, a szükséges bitek számát megkapjuk a konstans ( $u$ ) kettes alapú logaritmusaként. Mivel ez rendszerint kis érték, sok tört bitre lenne szükség, a törtbitek száma túllépné az összes bit értékét, így az ábrázolt biteket eltoljuk, tehát az egész bitek száma negatív lesz. Ebben a számábrázolásban az összes bitet és az előjel bitet nekünk kell előre megadnunk.

A második lépés a szorzat végeredményének, a szorzat alakjának a meghatározása. Ehhez először tudnunk kell a szorzat bemenetének fixpontos alakját, melyet a legelső módszer alapján mint változó kiszámíthatunk. Ebből a reprezentációból előáll az összes bit, illetve az előjel bitek száma. Itt törekednünk kell, hogy mivel a szorzó maximum 25 bitet tud fogadni, a változó értéke ne lépje túl azt. Miután a bemenet segítségével előállt az összes bit száma, már csak a tört biteket kell meghatároznunk, ez hasonlóan történik, mint a szorzótényező bitjeinek meghatározásakor, de mivel itt pozitív számot kell kapjunk, a feltételeket megfordítjuk, vagyis,

$$f_{\text{szorzat}} = \begin{cases} -\lceil \log_2(u) \rceil & , u > 1 \\ -\lfloor \log_2(u) \rfloor & , \text{egyébként} \end{cases} \quad (5.7)$$

$$m_{\text{szorzat}} = m_{\text{bemenet}} + f_{\text{szorzat}} \quad (5.8)$$

Ahhoz, hogy az integrátorban a hiba ne akkumulálódjon, a visszacsatolás összeadójának bitszámát is meg kell választanunk. Ehhez tudnunk kell az integrátor kimenetének alakját, amit szintén változóként az első módszer alapján számolhatunk. Ennek előállítása

után megkaphatjuk az összegző felbontását, mégpedig

$$i_{\text{összeadó}} = \begin{cases} 1 & , i_{\text{szorzat}} + i_{\text{kimenet}} > 1 \\ 0 & , \text{egyébként} \end{cases} \quad (5.9)$$

$$n_{\text{összeadó}} = m_{\text{összeadó}} + n_{\text{kimenet}} \quad (5.10)$$

$$m_{\text{összeadó}} = m_{\text{szorzat}} \quad (5.11)$$

vagyis az összegző tört bitjeit a szorzat alakja, míg egész bitjeit a tört bitek és a kimenet határozza meg.

A harmadik fő mennyiség, a modellben használt összeadók méretezése, ez hasonlóan zajlik az integrátor összegzőjéhez. Itt minden esetben a bemenő jelek maximális bitszámával kell számolni, vagyis

$$i_{1+2} = \begin{cases} 1 & , i_1 + i_2 > 1 \\ 0 & , \text{egyébként} \end{cases} \quad (5.12)$$

$$n_{1+2} = \max(n_1, n_2) \quad (5.13)$$

$$m_{1+2} = \max(m_1, m_2) \quad (5.14)$$

ahol  $i_1, n_1, m_1$  az egyik, míg  $i_2, n_2, m_2$  a másik bemenet ismert értékei.

## 5.2. A modell legfelső szintű leírása

Az 5.1. ábrán látható a Simulink modell felső szintje. Ez alapján a modell hat főbb részre osztható, melyek közül a legelső elem az Input sync modul. Itt történik meg a rendszer vezérlő jeleinek és bemeneteinek a beléptetése. A modell az FPGA-n belül szinkron megvalósítást követ, viszont a külvilágból a jelek aszinkron érkehetnek, ezért azokat csak szinkron engedélyezhetjük a modellen belül.

Miután előálltak a rendszer működéséhez szükséges jelek, következik a szinkron DC/DC konverter modellje, a Szinkron buck konverter modell. Ez tartalmazza az IGBT hídágat,

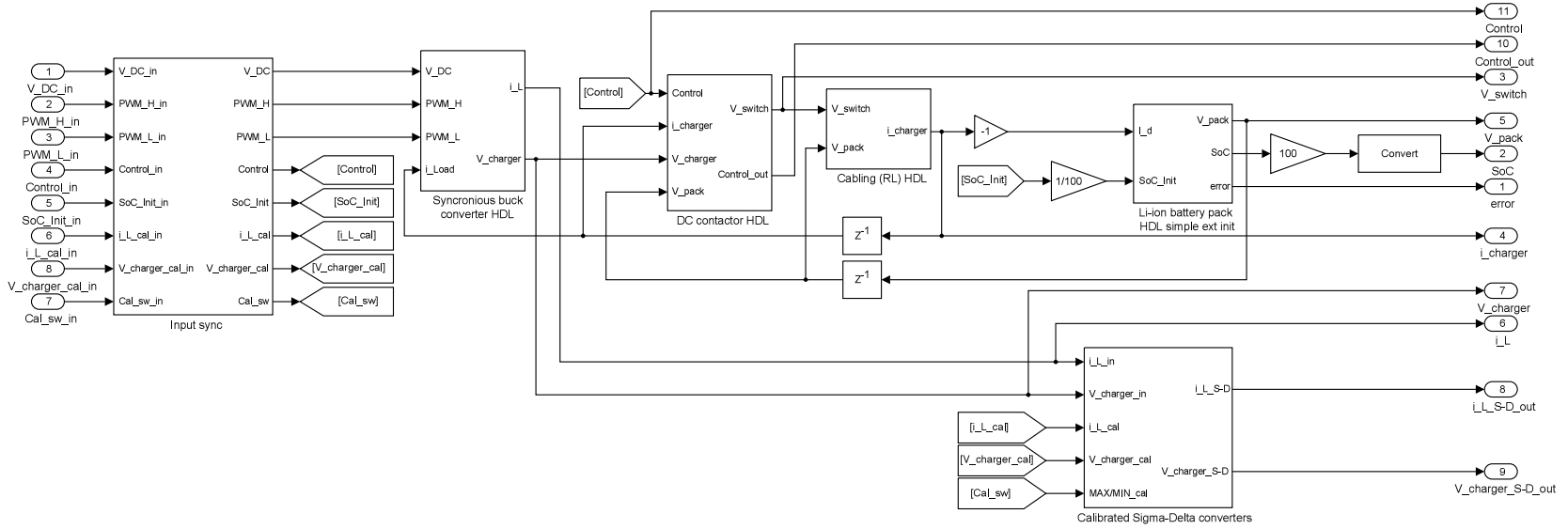


valamint a fojtótekercest és kimeneti DC kondenzátort. A modell kimenetét képezi az  $i_L$  áram és  $V_{Charger}$  feszültség, melyet a kalibrált  $\Sigma/\Delta$  modulátor modellen keresztül az FPGA két lábára küldünk.

A konverter által előállított  $V_{Charger}$  feszültség jut a Kontaktor modellbe, mely tartalmaz elektromos és mechanikus modellezést is. Itt állítjuk elő a kontaktor kimenetére kerülő  $V_{switch}$  feszültséget.

Míndezek után következik az akkumulátor pakk modell. Mivel a modell elkészítésekor pozitív áramnak a pakk kisütő áramát vettem, így a modellbe bekerült egy  $-1$ -es szorzó a megfelelő áramirány előállításához.

Az ábra alsó részén található még a  $\Sigma/\Delta$  modulátorok modellje is. Ezek egy kalibrációs logikát is tartalmaznak a vezérlő egység AD átalakítóinak teszteléséhez és konfigurációjához.



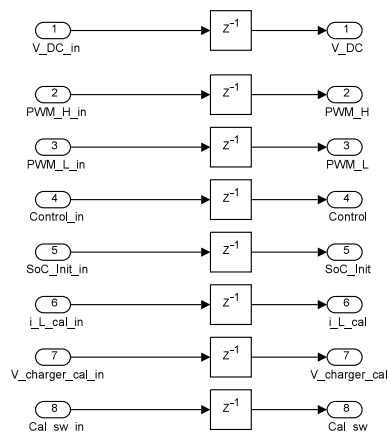
5.1. ábra. Simulink modell legfelső szintje

### 5.3. A Simulink modell részeinek leírása

A következőkben írom le a rendszer előző fejezetében ismertett részeinek pontos megvalósítását, majd a korábban ismertett állapotegyenletek Simulink-be ültetését.

#### 5.3.1. A bemenetek szinkronizálása

Mint azt korábban is írtam az FPGA szinkron ütemezéssel dolgozik. Ezt valósítja meg az 5.2. ábrán látható modell részlet. Ez a legelső blokk, ami a jelek szempontjából az FPGA határán van, tehát minden bemenete az FPGA-n kívülről érkezik.



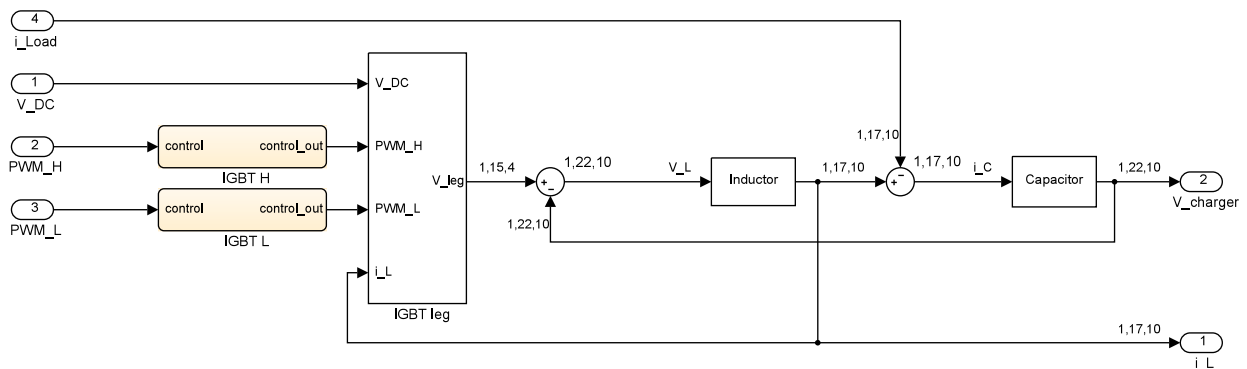
5.2. ábra. Bemenetek beléptetése

A modell ezen részének két feladata van, a bemeneti jelek órajelre történő frissítése és a jelek fixpontossá alakítása. Az első funkciót látják el az egy lépésközt késleltető blokkok. Ezek kimeneteit minden órajelre, tehát egy lépésköz elteltével frissítik, így megakadályozva az aszinkron viselkedést.

#### 5.3.2. Szinkron DC/DC feszültség átalakító Simulink modellje

A rendszer következő részegysége a feszültségcsökkentő DC/DC átalakító, mely előállítja az  $i_L$  és  $V_{Charger}$  állapotváltozókat, valamint a kiszámításukhoz szükséges  $V_L$  segédváltozót. A konverter modellje az 5.3. ábrán látható.

Ez a modell írja le a kapcsolás elektromos viselkedését, és a kapcsolóelemek késleltetését (sárga blokkok). Az IGBT leg nevű egység állítja elő az állapotváltozók számításához

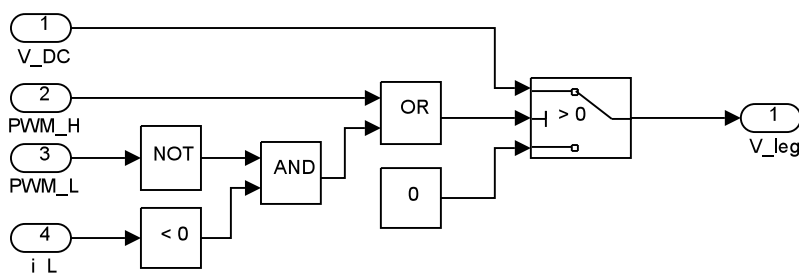


5.3. ábra. DC/DC átalakító Simulink modellje

szükséges  $V_L$  feszültséget. Ahhoz, hogy megértsük a működését írjuk fel újra  $V_L$  egyenletét.

$$V_L = \begin{cases} V_{DC} & , PWM_H = 1 \\ V_{DC} & , PWM_L = 0 \text{ és } i_L > 0 \\ 0 & , \text{egyébként} \end{cases} \quad (5.15)$$

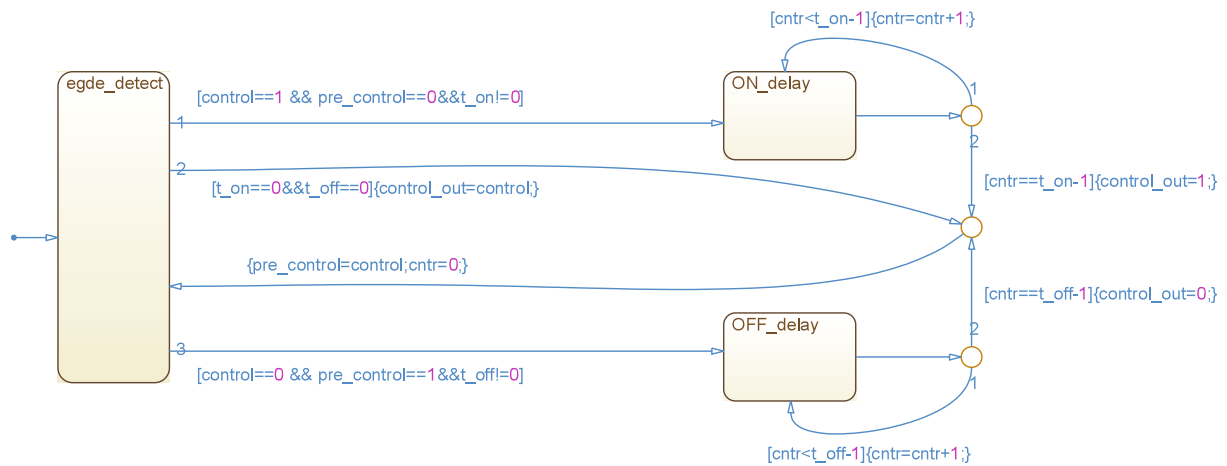
Ez alapján a Simulink modell a következőképpen néz ki,



5.4. ábra. IGBT hídág a  $V_L$  segédváltozó kiszámításához

Az IGBT lábra kerülő  $PWM_H$  és  $PWM_L$  jelek már az IGBT-k be- és kikapcsolási idejével késleltetett jelek. Ezeket a késleltetéseket Simulink-en belül Stateflow[30] segítségével oldottam meg. Ebben kívülről megadható paraméterként kerül be a késleltetés mértéke, mégpedig a lépésköz multiplikátoraként. A Stateflow használatával egy három-állapotú állapotgép segítségével oldottam meg a be- és kikapcsolási késleltetések egyidejű megoldását. A Stateflow az 5.5. ábrán látható.

Az állapotgép éldetektálás segítségével lép át bekapcsolási és kikapcsolási állapotokba.



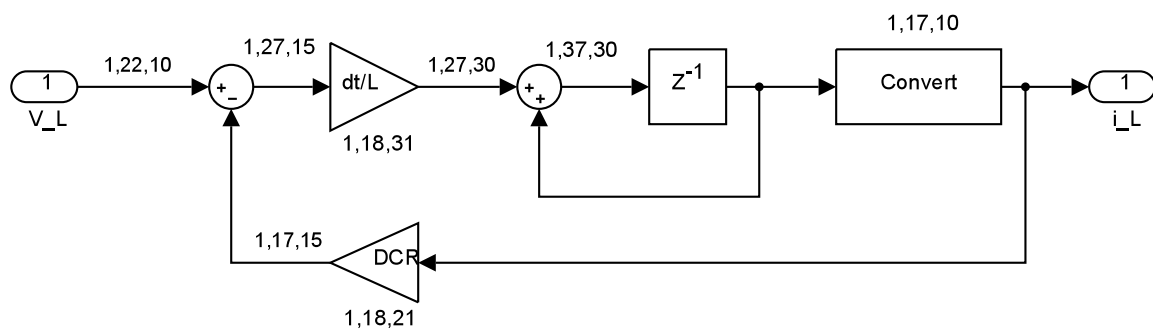
5.5. ábra. IGBT késleltetések állapotgépe

Amennyiben a kontroll jelen pozitív él érkezik, átlépünk bekapcsolási késleltetésbe, míg lefutó él esetén kikapcsolási késleltetésbe. Mindkét esetben egy számláló értékét növeljük lépésenként mindaddig, míg el nem érjük a késleltetés mértékét, ekkor kimenetet megfelelőértékre billentjük, majd várunk a következő vezérlésre.

A DC/DC átalakító további részei tartalmazzák az  $i_L$  és  $V_{Charger}$  állapotváltozók kiszámítását. A tekercs modellezéséhez előállított  $V_L$  és a kondenzátor modell kimenetéről visszacsatolt feszültség különbségeként előálló feszültség segítségével számoljuk  $i_L$ -t az 5.16. egyenlet alapján.

$$i_L(t) = \frac{1}{L} \int_0^t V_L(t) - (V_{Charger}(t) + i_L(t)DCR) dt \quad (5.16)$$

Így  $i_L$  számítása látható az 5.6. ábrán. Látható az ábrán a megvalósított előrelépő Euler módszert használó numerikus integrátor.

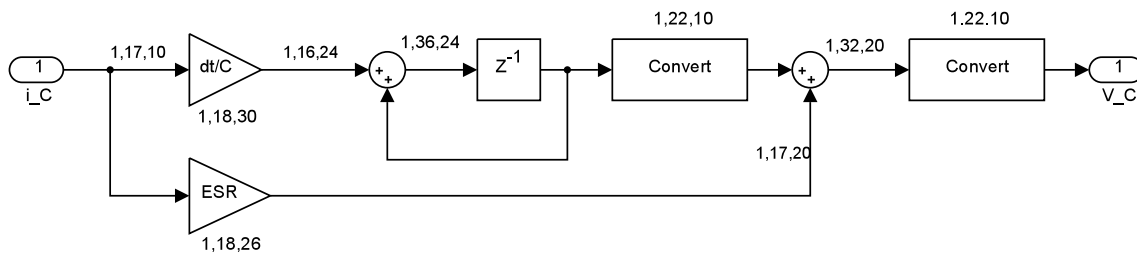


5.6. ábra. Az  $i_L$  állapotváltozó számítása

A  $V_{Charger}$  állapotváltozót a kondenzátor modellben számoljuk. Ennek bemenetén állítjuk elő a kondenzátor töltő áramát, vagyis  $i_C = i_L - i_{Charger}$ . Ez alapján  $V_{Charger}$  egyenlete,

$$V_{Charger}(t) = \frac{1}{C} \int_0^t (i_L(t) - i_{Charger}(t)) dt + (i_L(t) - i_{Charger}(t)) ESR \quad (5.17)$$

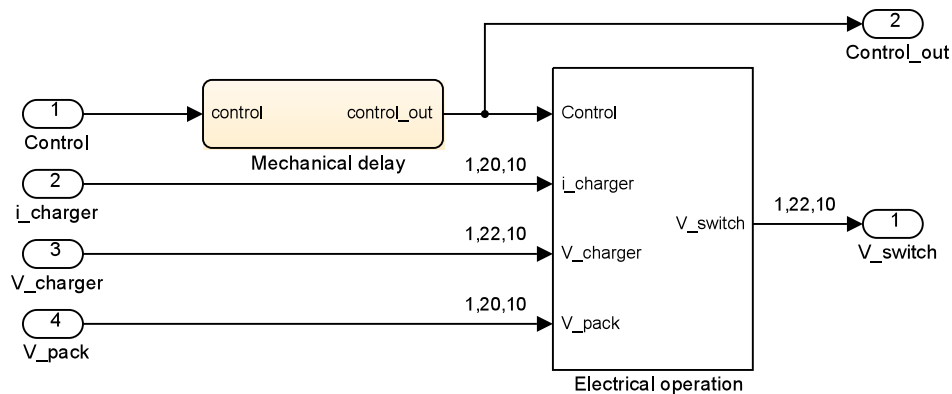
Az állapotegyenlet alapján  $V_{Charger}$  kiszámítása a következőképpen néz ki,



5.7. ábra. A  $V_{Charger}$  állapotváltozó számítása

### 5.3.3. A DC kontaktor Simulink modellje

A kontaktor modellje látható az 5.8. ábrán. A kontaktor modelljében valósult meg



5.8. ábra. A DC kontaktor Simulink modellje

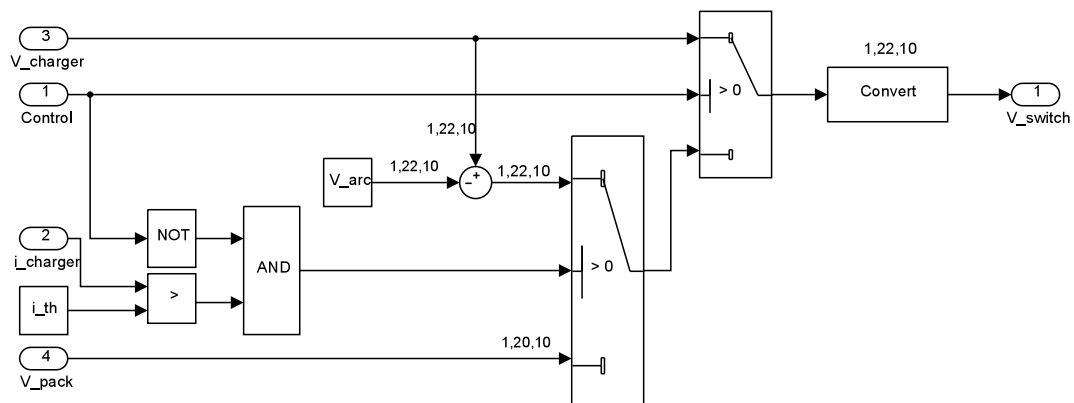
annak elektromos és mechanikus viselkedése. Mivel ez az elem az IGBT-khez hasonlóan kapcsolóelem, így késleltetése van. Bár a késleltetés oka különbözik a két esetben, hisz az IGBT-k félvezető elemek, azok késleltetése a töltéskihúzásból származik, míg a kontaktor elektromechanikus egység, és késleltetése a mechanikából származik, mindkét esetben a

késleltetések modellezése hasonlóan történhet. Épp ezért az IGBT-k és a kontaktorok be- és kikapcsolási késleltetését egyforma Stateflow állapotgépekkel oldottam meg, ami az 5.5. ábrán látható.

Az elektromos viselkedést leíró modell tartalmazza a  $V_{switch}$  segédváltozó leírását. A segédváltozó egyenlete a következő,

$$V_{switch} = \begin{cases} V_{Charger} & , K = 1 \\ V_{pack} & , K = 0 \text{ és } i_{Charger} < i_{threshold} \\ V_{Charger} - V_{arc} & , k = 0 \text{ és } i_{Charger} > i_{threshold} \end{cases} \quad (5.18)$$

Ezek alapján a Simulink modell az 5.9. ábrán látható.



5.9. ábra. A  $V_{switch}$  segédváltozó számítása

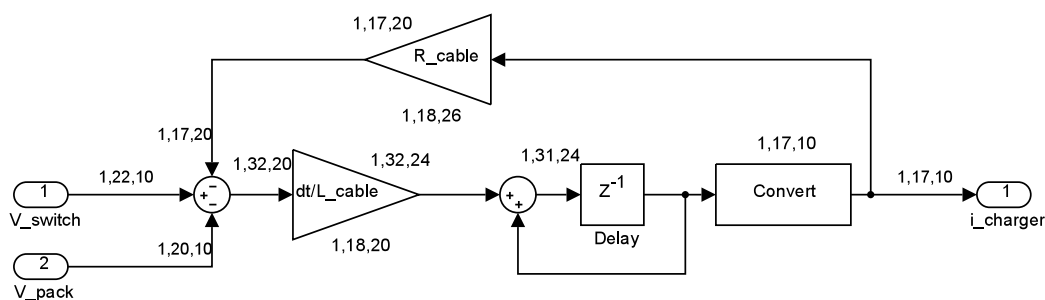
Mint azt már a 4.3.1. fejezetben leírtam, a DC kontaktor induktív körben történő kikapcsolásakor fellépő ívet egy konstans  $V_{arc}$  feszültségként vettem figyelembe, melyet a töltőáram ( $i_{Charger}$ ) értékétől függően kapcsolunk. Abban az esetben, amikor kikapcsolás történik, vagyis  $Control$  értéke 0-ra vált, és  $i_{Charger}$  értéke nagyobb mint egy általunk adott küszöb áram ( $i_{th}$ ), akkor kontaktoron még folyik áram az ívben, így a kontaktor kimenetén lévő feszültséget csökkentjük az ívfeszültséggel ( $V_{arc}$ ).

### 5.3.4. A cabling Simulink modellje

A főkör modelljének másik elemét, a kábelezést reprezentáló  $R - L$  helyettesítőképet tartalmazza a Simulink modell Cabling blokkja. Ez a blokk állítja elő az  $i_{Charger}$  állapotváltozót, mely az akkumulátor pakk töltőárama. Az állapotegyenlete a következő:

$$i_{Charger}(t) = \frac{1}{L_{cabling}} \int_0^t V_{Charger}(t) - (V_{pack}(t) + i_{Charger}(t)R_{cabling}) dt \quad (5.19)$$

Az egyenlet alapján megvalósított modell látható az 5.10. ábrán.



5.10. ábra. A főköri vezetékvezés  $R - L$  helyettesítőképének modellje

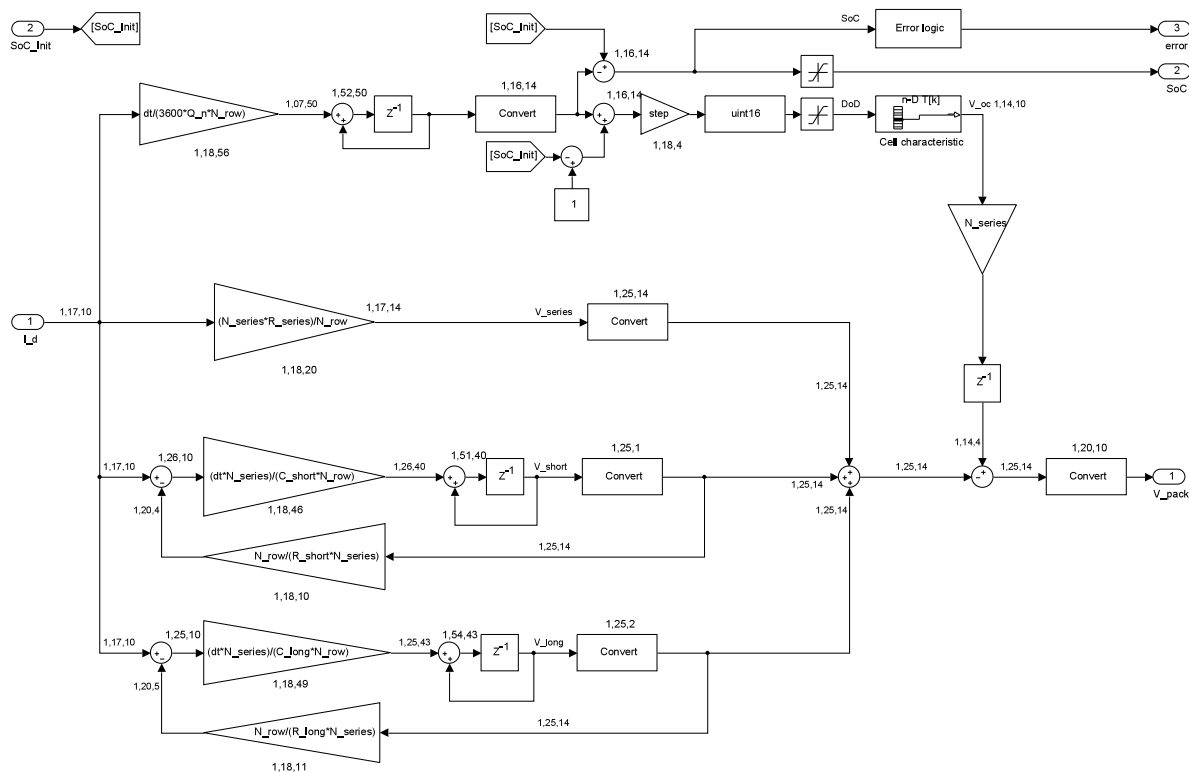
### 5.3.5. Az akkumulátor pakk Simulink modellje

A 2.1. fejezetben leírt és kiválasztott helyettesítőkép megvalósítása látható az 5.11. ábrán. A pakk modellezését a 4.3.2. fejezetben leírtak alapján egy akkumulátor cella modell felskálázásával ( $N_{series}$  és  $N_{row}$ ) értem el.

A modell –mint a helyettesítőkép– két részre osztható, az 5.11. ábra felső része valósítja meg az üresjárat  $V_{oc}$  feszültség nemlineáris karakterisztikájának futásidejű leírását (helyettesítőkép bal oldala, 2.1. ábra), valamint a feszültség-áram karakterisztikát (az 5.11. ábra alsó rész).

A  $V_{oc}$  karakterisztikát a gyártói adatlapból olvastam be, mely a kisütés (DoD) függvényében volt ábrázolva, így a modell elkészítésekor én is ezt alkalmaztam. A karakterisztikát egy egydimenziós LUT-ban tároltam, melyben a karakterisztika 4096 pontra van bontva és azt DoD indexeli. Ahhoz, hogy DoD értékét meghatározhassuk, a kisütőáramot kell integrálnunk a 2.3. egyenlet alapján.





5.11. ábra. A kombinált akkumulátor cella helyettesítőkép modellje

Ezzel a módszerrel bizonyos felhasználási területeken nem tudunk megfelelően pontos DoD számítást végezni, ehhez léteznek Kalman-szűrőt alkalmazó, modell alapú eljárások, viszont esetünkben a pakkszintű dinamikus vizsgálathoz megfelelő módon tudjuk számolni DoD értékét.

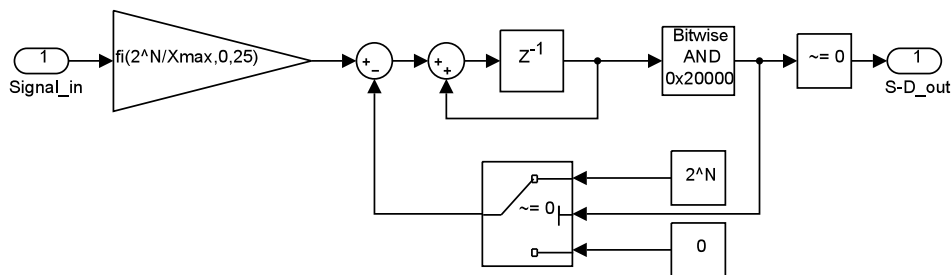
A modellben képesek vagyunk kívülről megadni annak kezdő értékét, de mivel szemléletesebb értékeket kapunk, ha a megmaradt felhasználható töltést nézzük, ezért ezt SoC-ben adhatjuk meg ( $SoC_{init}$ ), illetve az aktuális töltöttségi szintet is SoC értékben vezetjük ki.

Az integrálással előállított DoD-t skáláznunk kell, hogy az a LUT teljes tartományát címezni tudja, majd a LUT kimenetét egy  $N_{series}$  szorzóval szorozzuk, hogy megkaphassuk a pakkszintű üresjáratú feszültséget. A modellnek ez a része tartalmaz még egy error logikát, amivel jelezni tudjuk, ha töltés, illetve kisütés közben túlléptük  $V_{oc}$  helyesen címezhető tartományát. Mivel  $V_{oc}$ -t görbeillesztés segítségével kaptam, az helyes értékeket csak DoD 0 – 1 tartományon mutat, így detektálnunk kell az esetleges félreindexelést.

A modell másik része tartalmazza a feszültség-áram karakterisztikákat, itt került implementálásra a helyettesítőkép belső ellenállása és két  $R - C$  tagja.

### 5.3.6. A $\Sigma/\Delta$ modulátor Simulink modellje

Mint már korábban is írtam a vezérlőegység számára biztosított, visszacsatolásként szolgáló jeleket  $\Sigma/\Delta$  modulátoron és szűrőn keresztül állítjuk elő. Ennek a  $\Sigma/\Delta$  modulátornak a Simulink modellje látható az 5.12. ábrán.



5.12. ábra. A  $\Sigma/\Delta$  modulátor Simulink modellje

A modulátor működéséhez paraméterként előre meg kell adnunk a jel összes bitszámát ( $N$ ), illetve a jel maximális értékét ( $X_{max}$ ). Ez alapján a két érték alapján a  $\Sigma/\Delta$  bemenetén skálázzuk a bejövő jelet, mivel a fix pontos reprezentáció miatt a legnagyobb ábrázolható szám rendszerint nem egyezik meg a fizikai jel maximálisan felvehető értékével. Ezzel a skálázással elérhetjük, hogy a bemenetre kerülő  $X_{max}$  legyen a modulátor full-scale (FS) érték, és a modulátor kimenetén ekkor jelenjen meg a legnagyobb érték.

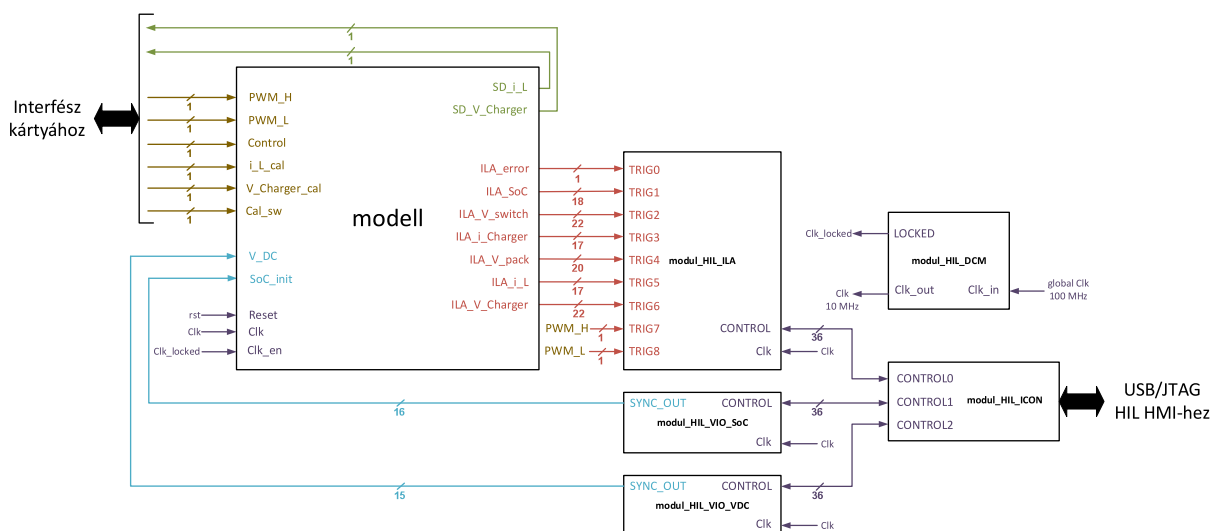
A skálázott jelet rávezetjük egy integrátorra, majd az integrátor kimenetét bit-maszkoljuk az MSB bitre, és ennek függvényében a bemenetből nullát, vagy a legnagyobb ábrázolható számot vonjuk le. A maszkolás eredményét nullára komparáljuk, ennek az eredménye kerül a kimenetre.

## 5.4. A felhasználói felület (HMI)

A megvalósított HIL szimulátor belső jeleit vizsgálni, egyes paramétereit valós időben állítani szeretnénk. Ehhez szükségünk van egy felhasználói felületre (HMI). Ennek megvalósításához az FPGA-ban egy kommunikációs egység implementálására lenne szükség, aminek HDL szintű leírása bonyolult.

Ahhoz, hogy ne kelljen ezt a kommunikációs hardvert az FPGA-ra szintetizálni, ezért a kommunikáció és HMI megvalósításához a Xilinx cég ChipScope-Pro[32] alkalmazását használtam, mely az ISE Design Studio-ban megtalálható. A szimulátor futtatásakor jeleinek monitorozására a ChipScope-Pro felhasználói felületét használtam.

Ahhoz, hogy a modell a számítógépen futó ChipScope HMI-vel kommunikálni tudjon, ChipScope magokat kell szintetizálnunk az FPGA-ra, melyet HDL szinten végezhethünk el. Ezek a magok és azok a modellhez való viszonya látszik az 5.13. ábrán.



5.13. ábra. Chipscope-pro magok és a modell viszonya

A megfelelő működéshez négy különböző mag szintetizálására van szükség, melyből három felelős a HMI-vel való kapcsolat létrehozására, egy pedig a megfelelő órajelet állítja elő. A szimuláció futtatására előre meghatároztuk a 100 nsec-os lépésközt, vagyis 10 MHz órajelet frekvenciát. Ahhoz, hogy ezt elérjük, le kell osztanunk az FPGA globális órajeletét, ami esetünkben 100 MHz. Ezt a funkciót látja el a *modul\_HIL\_DCM* mag (Digital-Clocking-Manager). Az ezzel a maggal előállított órajelet használjuk fel mind a modellünkben, mind a ChipScope magokban.

Az FPGA fejlesztőkártyán lehetőségünk van USB kábel csatlakoztatására, így a kommunikációt a számítógép és a szimulátor között USB/JTAG segítségével célszerű megvalósítani. Ahhoz, hogy ez a kommunikációs csatorna létrejöjjön, implementálnunk kell egy ICON (Integrated-Controller) magot (*modul\_HIL\_ICON*). Ez az egység teremt kapcsolatot az FPGA-ra szintetizált többi maggal 36 bites CONTROL vonalak segítségével.

Minden egyes ChipScope maghoz szükségünk van egy ilyen kétirányú kommunikációt lebonyolító CONTROL vonal létrehozására.

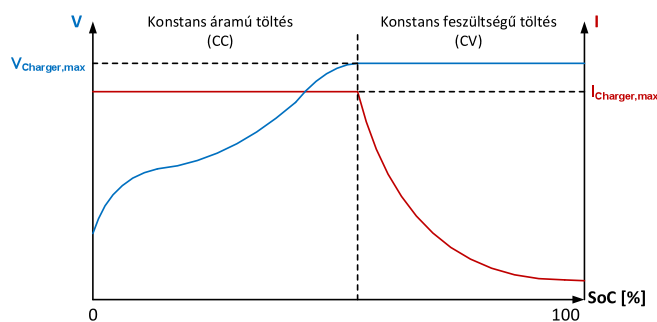
A modell ki- és bemeneteit nem feltétlenül kell fizikai ki- és bemenetekre vezetnünk, használhatunk virtuális ki- és bemeneti portokat (VIO, Virtual-Input-Output), melyek a ChipScope-pal teremtenek egyirányú kapcsolatot ICON-on keresztül. Ezekon a virtuális portokon keresztül tudjuk a modell egyes paramétereit valós időben állítani. Egy ilyen modullal egyszerre egy jel értéke állítható, mely tetszőleges bitszámú lehet. A modellben két paraméter folyamatos állítására van lehetőség, melyek a DC/DC átalakító bemenetére kerülő  $V_{DC}$  feszültség és az akkumulátor pakk kezdeti töltöttségi szintje ( $SoC_{init}$ ). Mindkét jelhez implementálni kellett egy *modul\_HIL\_VIO* magot.

Az utolsó ChipScope mag segítségével végezhető el a modellünk belső jeleinek monitorozása. Ehhez két dologra van szükség. Egyrészt a Simulink modellben a vizsgálandó jeleket egy-egy port segítségével ki kell vezetnünk (5.1. ábra), illetve implementálnunk kell egy ILA (Integrated-Logic-Analyzer) magot (*modul\_HIL\_ILA*). Ennek a magnak a segítségével több jelet tudunk egyszerre vizsgálni, a ChipScope-Pro alkalmazáson belül integrált szkóp segítségével az általunk kiválasztott jelek az időtartományban vizsgálhatók. Az ILA mag generálásakor meg kell adnunk, hogy egyszerre hány mintavételt tároljon el, ezzel meghatározzuk az ábrázolható maximális időintervallumot.

### 5.5. A mérési eredmények

Az elkészített HIL szimulátor validációjához a Siemens Zrt. által biztosított vezérlő segítségével méréseket végeztem. Ezeknek a méréseknek az alkalmával elvégeztem a modellezett akkumulátor pakk teljes töltési ciklusának modellezését.

A vezérlőegységben egy konstans áramú (CC), konstans feszültségű (CV) töltést megvalósító állapotgép és szabályozó fut. A teljes töltési metódus négy részre bontható. Első lépésként elő kell töltenünk a DC/DC konverter kimeneti kondenzátorát, ez az előtöltő szakasz. Ez után a kontaktorok rákapcsolásával megkezdhetjük a tényleges CC-CV töltést. A töltés elő felében konstans áramszabályozást valósítunk meg, az akkumulátorokat konstans árammal töltjük, a feszültség az akkumulátor pakkfeszültségéhez hasonló mó-



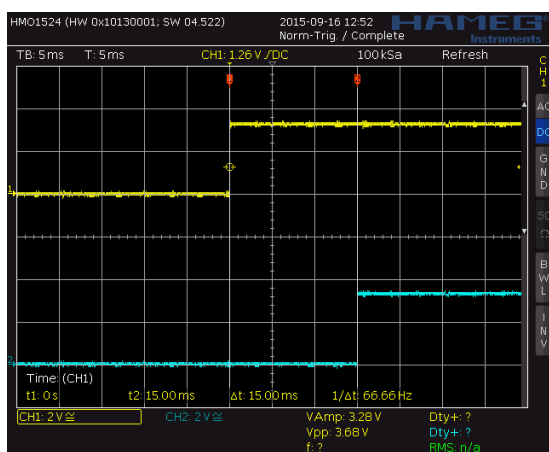
5.14. ábra. CC-CV töltés jelleggörbéi

don emelkedik. Ebben a töltési állapotban maradunk mindaddig, míg el nem érjük az akkumulátorok adatlapjában megadott maximális töltőfeszültséget.

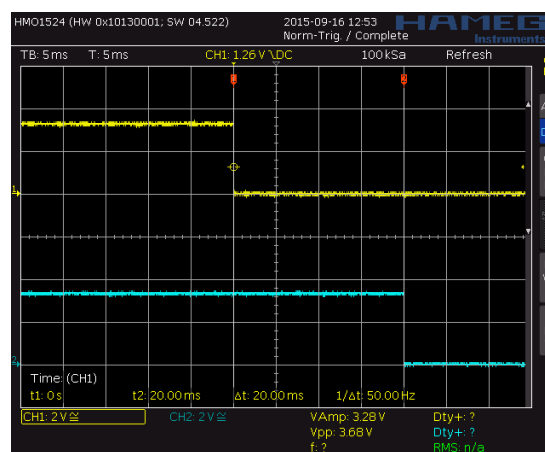
Ekkor átlépünk konstans feszültségű töltésre. Ebben az állapotban a feszültséget végig a maximális töltőfeszültségen tartjuk, míg az áram folyamatosan csökken. Ezt a töltési eljárást folytatjuk mindaddig, míg el nem érünk egy minimális, maradó áramot.

Miután lecsökkent az áram nulla körüli értékre, lekapcsoljuk a kontaktorokat és szabályzott módon kisütjük a töltő fokozat kimeneti kondenzátorát, ezzel befejezzük a töltést.

A teljes töltési ciklus elvégzése előtt teszteltem a kontaktorok működését és a mechanikai késleltetés mértékének helyes beállítását. Ehhez az FPGA fejlesztőkártya két lábára kiveztem a kontaktor késleltetés előtti és az utáni vezérlőjelét.



(a) Bekapcsolási késleltetés



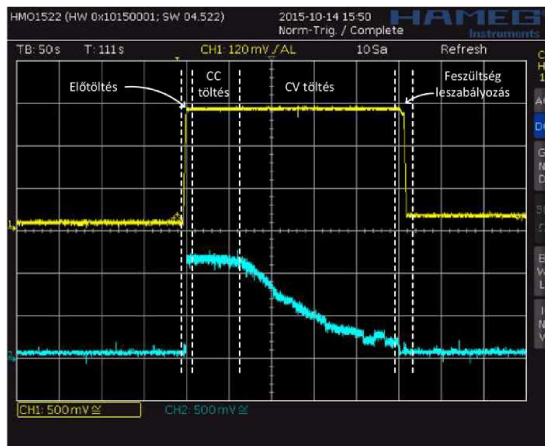
(b) Kikapcsolási késleltetés

5.15. ábra. A kontaktor mechanikai késleltetései

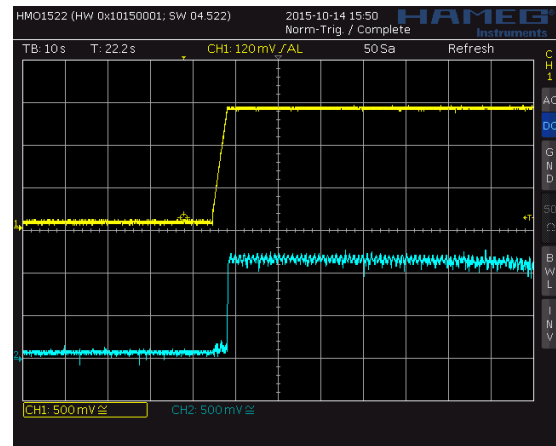
Az 5.15. ábrán látható sárga jel a késleltetés előtti, míg a kék jel a késleltetés utáni vezérlőjel. Az 5.15a ábra kurzorai által mért késleltetés megegyezik az előre beállított 15

msec késleltetéssel, illetve az 5.15b. ábrán mért 20 msec késleltetés is megegyezik az előre beállított értékkel, tehát a Stateflow alapú késleltetés jól működik.

Miután meggyőződtem a kontaktor modell helyes működéséről elvégeztem a teljes töltési ciklusra a tesztelést. Hogy a töltés jelleggörbéi, az állapotok közötti váltás egy oszcilloszkóp ábrán is látható legyen, a mérést elvégeztem egy 95 %-ra feltöltött akkumulátor pakk modellre is, ez látható 5.16a. ábrán.



(a) A töltés teljes ciklusa



(b) Előtöltés és CC töltés

5.16. ábra. A töltési ciklus

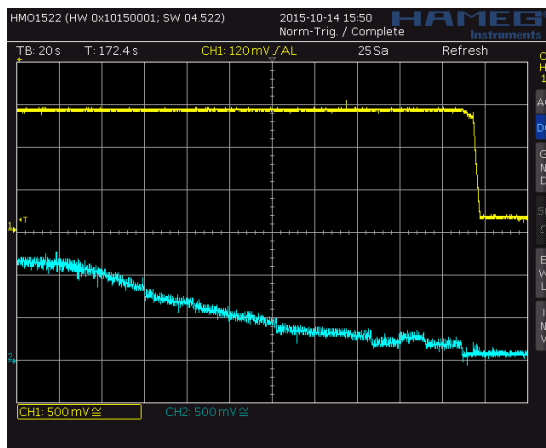
Az előtöltéshez 5 A-es áramreferenciát kapott a szabályzó, majd konstans áramtöltéskor a megengedhető legnagyobb árammal, 25 A-ral végeztük a töltést. Az oszcilloszkóp ábrákon a  $\Sigma/\Delta$  szűrt kimeneti jele látható a szintillesztés után, tehát az értékek feszültség jelek a 0 – +3 V tartományban. Az ábrákon a sárga jel felel meg a DC/DC konverter kimeneti  $V_{Charger}$  feszültségnek, míg a kék jel a konverter  $i_L$  fojtótekercs áramának.

Az 5.16b. ábrán látható, hogy a feszültséget előtöltéskor túllövés nélkül felszabályozzuk kis áramérték mellett, majd a kontaktor rákapcsolása után  $i_L$  áramot gyorsan a maximális értékére szabályozzuk és értékét ott tartjuk, míg a töltő feszültség ( $V_{Charger}$ ) el nem éri a maximális értékét. A szabályzott áram jó közelítéssel konstansnak tekinthető. A megfelelő töltés ellenőrzésére vizsgáltam az akkumulátorok töltöttségét (SoC) az idő függvényében. Mivel ebben az állapotban konstans árammal töltünk, SoC lineárisan változik, így megtehetjük, hogy SoC bizonyos mértékű változásához szükséges időt lemérjük, majd azt viszonyítjuk az elméletben szükséges időhöz. Ebben a mérésben az SoC 10 %-ról 20 %-ra

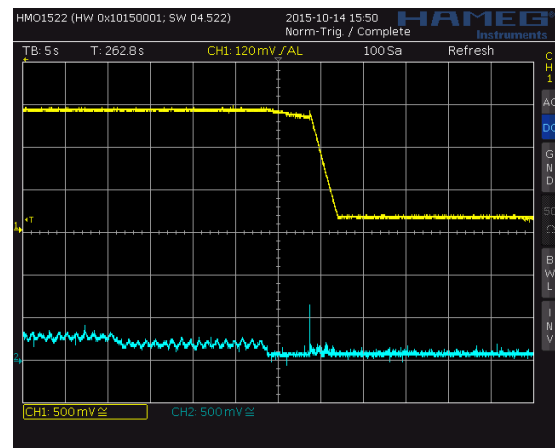
történő változását vizsgáltam. Ehhez a következő képletet írhatjuk fel:

$$t_d = \frac{Q_n[As]}{I_d[A]} = \frac{93600}{25} = 3744 \text{ sec} \quad (5.20)$$

Ezt az értéket célszerű percre átszámítani, ami  $t_d = 62,4$  min, ami megfelel az akkumulátorpakk 0 %-ról 100 %-ra történő feltöltésének. Ennek egy tizede fog megfelelni a  $\Delta SoC = 10$  % eltéréshez szükséges időhöz, ami  $t_{d,10\%} = 6,24$  min. A mért idő ehhez képest  $t_{d,mért} = 6,28$  min, ami kis mértékben tér el az elméleti értéktől, ez a pontatlanság adódhat az SoC értékek pontatlan leolvasásából, illetve az időmérés indításának és leállításának pontatlanságából. Ez alapján a mérés alapján látszik, hogy a töltés jól működik.



(a) A CV töltés



(b) A töltés vége, a töltő feszültség leszabályozása

5.17. ábra. CV töltés és a töltési ciklus vége

Miután  $V_{Charger}$  feszültséggel elértük annak maximális értékét, a vezérlő egység állapotgépe átlép CV töltésbe (5.17a. ábra), ekkor a töltő feszültséget ezen a maximális értéken tartja, míg az áram folyamatosan csökken egészen egy kis maradék áram értékéig. Ennél az áramnál már nem végzünk érdemi töltést, úgy tekinthetjük, hogy a töltési ciklus befejeződött. Ekkor lekapcsoljuk a kontaktort, majd megint csak 5 A-es áramkorláttal leszabályozzuk a kimeneti kondenzátor feszültségét, ez látható az 5.17b. ábrán. Ezzel be is fejeződött a töltési ciklus, a HIL szimulátor és a vezérlő egység helyesen működik.

## 6. Értékelés

A TDK munkám alatt sikerült elkészíteni a megtervezett HIL szimulátort, azon validációs méréseket tudtam végezni. A szimulátor megfelelően, az előre kitűzött célok szerint működött. A TDK munka alatt sikerült elsajátítanom HIL szimulátor tervezési módszereket, melyek segítségével a későbbiekben további irányokban tudok fejleszteni, kutatásokat végezni. Ilyen fejlesztési irány lehet egy akkumulátor felügyeleti rendszer (BMS) tesztelése HIL szimulátor segítségével, valamint az akkumulátorok termikus modelljének integrációja a létező modellbe. Ezekhez szükségesek a cellák identifikációjának elvégzése, melyek eredményeinek felhasználása a termikus modellben az ipari BMS és töltéskiegyenlítő rendszerek fejlesztésében, tesztelésében kiemelt fontosságúak.

Ez a HIL szimulátor lehetőséget nyújt a későbbiekben nagyobb teljesítményű és ezzel veszélyesebb akkumulátoros teljesítmény fokozatok biztonságos tesztelésére. A bemutatott valós idejű HIL tesztelési eljárás és maga a szimulátor tervezése számos további lehetőséget tartogat mind kutatási mind fejlesztési irányokat tekintve, melyeket feltétlenül tovább szeretnék folytatni.



# Irodalomjegyzék

- [1] Debreceni Tibor: "Mini naperőmű FPGA alapú valós idejű Hardware-In-the-Loop szimulátorának tervezése", TDK dolgozat, 2013.
- [2] 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).
- [3] 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.
- [4] 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.
- [5] 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.
- [6] J. C. G. Pimentel, H. Le-Huy: "Hardware emulation for real-time power system simulation", in Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE2006), vol. 2, Jul. 9–13, 2006, pp. 1560–1565.
- [7] Z. Sütő, T. Debreceni, T. Kökényesi, A. Futó, I. Varjasi: "Matlab Simulink Generated FPGA Based Real-time HIL Simulator and DSP Controller: A Case Study ", International Conference on Renewable Energies and Power Quality (ICREPQ'14), ISSN 2172-038 X, No.12, April 2014.
- [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.
- [9] G. G. Parma and V. Dinavahi: "Real-time digital hardware simulation of power electronics and drives", IEEE Transactions on Power Delivery, vol. 22, no. 2, pp. 1235–1246, Apr. 2007.

- [10] Tamás Kökényesi, István Varjasi: "FPGA-Based Real-Time Simulation of Renewable Energy Source Power Converters", *Journal of Energy and Power Engineering* 7 (2013) pp. 168-177.
- [11] Chen, M., Rincón-Mora, G.A: "Accurate Electrical Battery Model Capable of Predicting Runtime and I–V performance", *IEEE Transactions on energy conversion*, Vol. 21 No. 2, 2006, pp.504–511.
- [12] L. Song and J. W. Evans: "Electrochemical-thermal model of lithium polymer batteries", *J. Electrochemical Society*, vol. 147, pp. 2086-2095, 2000.
- [13] D. Rakhmatov, S. Vrudhula, and D. A. Wallach: "A model for battery lifetime analysis for organizing applications on a pocket computer", *IEEE Trans. VLSI Systems*, vol. 11, no. 6, pp. 1019-1030, Dec. 2003.
- [14] Rong and M. Pedram: "An analytical model for predicting the remaining battery capacity of lithium-ion batteries", in *Proc. Design, Automation, and Test in Europe Conf. and Exhibition*, 2003, pp. 1148-1149.
- [15] He, H.; Xiong, R.; Fan, J.: "Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation by an experimental approach", *Energies* 2011, 4, 582–598.
- [16] B. Schweighofer, K. M. Raab, and G. Brasseur: "Modeling of high power automotive batteries by the use of an automated test system", *IEEE Trans. Instrumentation and Measurement*, vol. 52, no. 4, pp. 1087-1091, Aug. 2003.
- [17] L. Gao, S.Liu, and R. A. Dougal: "Dynamic lithium-ion battery model for system simulation", *IEEE Trans. Components and Packaging Technologies*, vol. 25, no. 3, pp. 495-505, Sept. 2002.
- [18] S. Buller, M. Thele, R. W. De Doncker, and E. Karden: "Impedance-based simulation models of supercapacitors and Li-ion batteries for power electronic applications", in *Conf. Rec. 2003 Industry Applications Conf.*, vol. 3, pp. 159601600.
- [19] S. C. Hageman: "Simple pspice models let you simulate common battery types", *EDN*, pp. 17-132, Oct. 1993.
- [20] S. Gold: "A pspice macromodel for lithium-ion batteries", in *Proc. 12th Annual Battery Conf. on Applications and Advances*, 1997, pp. 215-222.
- [21] Long Lam: "A Partical Circuitbased Model for State of Health Estimation of Li-ion Battery Cells in Electric Vehicles", *Master of Science Thesis*, University of Technology Delft, 2011.
- [22] Rui Xiong, Xianzhi Gong, Chunting Chris Mi, Fengchun Sun: "A robust state-of-charge estimator for multiple types of lithium-ion batteries using adaptive extended Kalman filter", *Journal of Power Sources*, vol. 243, 2013, pp. 805-816.

- [23] Shifei Yuan, Hongjie Wu and Chengliang Yin: "State of Charge Estimation Using the Extended Kalman Filter for Battery Management Systems Based on the ARX Battery Model", *Energies* 2013, 6, pp. 444–470.
- [24] Habiballah Rahimi-Eichi, Student Member, IEEE, Federico Baronti, Member, IEEE, and Mo-Yuen Chow, Fellow, IEEE: "Online Adaptive Parameter Identification and State-of-Charge Coestimation for Lithium-Polymer Battery Cells", *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 4, April 2014.
- [25] Kendall E. Atkinson: "An intruduction to numerical analysis", Wiley & Sons, Inc. 1988, ISBN 0-471-62489-6, pp.251-255.
- [26] Uri M. Ascher, Linda R. Petzold: "Computer methods for ordinary differential equations and differential-algebraic equations", Society for Industrial and Applied Mathematics, 1998, ISBN 0-898-71412-5, pp.35-56.
- [27] Ravel F. Ammerman, Senior Member, IEEE, Tammy Gammon, Senior Member, IEEE, Pankaj K. Sen, Senior Member, IEEE, and John P. Nelson, Fellow, IEEE: "DC-Arc Models and Incident-Energy Calculations", *IEEE Transactions on Industry Applications*, Vol. 46, No. 5, Sept/Oct 2010.
- [28] F. M. Uriarte, A. L. Gattozzi, J. D. Herbst, H. B. Estes, T. J. Hotz, A. Kwasinsky, R. E. Hebner: "A DC Arc Model for Series Faults in Low Voltage Microgrids", *IEEE Trans. on Smart Grid*, vol. 3, no. 4, Dec. 2012.
- [29] Digilent: Nexys4<sup>TM</sup> FPGA Board Reference Manual,  
[https://www.digilentinc.com/Data/Products/NEXYS4/Nexys4\\_RM\\_VB2\\_Final\\_5.pdf](https://www.digilentinc.com/Data/Products/NEXYS4/Nexys4_RM_VB2_Final_5.pdf)  
(Oct. 2015)
- [30] Mathworks: Stateflow User's Guide,  
[http://www.mathworks.com/help/pdf\\_doc/stateflow/sf\\_ug.pdf](http://www.mathworks.com/help/pdf_doc/stateflow/sf_ug.pdf) (May 2014)
- [31] Mathworks: HDL Coder User' Guide,  
[http://www.mathworks.cn/help/pdf\\_doc/hdlcoder/hdlcoder\\_ug.pdf](http://www.mathworks.cn/help/pdf_doc/hdlcoder/hdlcoder_ug.pdf) (May 2014)
- [32] Xilinx: ChipScope Pro Software and Cores User Guide,  
[http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/chipscope\\_pro\\_sw\\_cores\\_ug029.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/chipscope_pro_sw_cores_ug029.pdf) (May 2014)