

Miskolci Egyetem



GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR
AUTOMATIZÁLÁSI ÉS INFOKOMMUNIKÁCIÓS INTÉZET

TDK DOLGOZAT

**FPGA platformon megvalósított
RISC-V mikrovezérlő alapú
motorvezérlő tervezése**

KÉSZÍTETTE:

Kocsis Benedek

MSc szintű, villamosmérnök szakos hallgató

KONZULENS:

Dr. habil. Vásárhelyi József

Egyetemi docens

Miskolc, 2023.

A konzulens nyilatkozata

Alulírott, Dr. habil. Vásárhelyi József a Miskolci Egyetem Automatizálási és Infó-kommunikációs Intézetének oktatója a

„FPGA platformon megvalósított RISC-V mikrovezérlő alapú motorvezérlő tervezése”

című TDK dolgozatot beadásra alkalmasnak ítélem.

Miskolc, 2023. április 26.



Dr. habil. Vásárhelyi József
a konzulens aláírása

Eredetiség nyilatkozat

"Alulírott Kocsis Benedek, a Miskolci Egyetem Gépészmérnöki és Informatikai Karának hallgatója büntetőjogi és fegyelmi felelősségem tudatában kijelentem és aláírással igazolom, hogy ezt a dolgozatot saját magam készítettem, a benne leírt vizsgálatokat – ha ezt külön nem jelzem – magam végeztem el, és az ismertetett eredményeket magam értem el. Adatokat, információkat csak az irodalomjegyzékben felsorolt forrásokból használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Miskolc, 2023. május 02.

Kocsis Benedek

.....
a hallgató aláírása

Tartalomjegyzék

1. Bevezetés	4
2. A BLDC motor meghajtó kártya bemutatása	7
2.1. Az áramkör bemutatása	8
3. MikroKon	15
3.1. Kapcsolási rajz	16
4. MotVez	17
4.1. Rendszerkép	17
4.2. Az érzékelő nélküli motor hajtás	19
4.3. SPI Slave	22
4.4. Position Detector (PD)	24
4.5. Commutation Controller (CCON)	27
4.6. PWM Controller (PWMC)	30
4.7. Output Controller (OC)	31
5. LovagSoC	32
5.1. Rendszerkép	32
5.2. RV32I processzor	33
5.3. A Wishbone buszrendszer	37
5.4. Memória	40
5.5. General Purpose Input Output (GPIO)	41
5.6. Serial Peripheral Interface Master (SPI Master)	41
5.7. Universal Asynchronous Receiver Transmitter (UART)	44
5.8. Debug támogatás	47
5.9. Programozói útmutató	49
6. Összefoglalás	53

1.

Bevezetés

Az egyenáramú (DC) motorok a mai ipar és mindennapi élet számos területén jelen vannak, mint elektro-mechanikus energia átalakító eszközök [32], [14]. Az egyenáramú motorok általánosan egy állórészből és egy forgórészből állnak. Az állórész és a forgórész által biztosított mágneses terek interakciójának eredményeként állítanak elő munkára felhasználható nyomatékot [11]. A hagyományos, kefések DC motorok állórésze laminált vasmagú gerjesztőtekercs, vagy állandómágnesek segítségével állíthatja elő a mágneses terét. [11] szerint az állandómágneses DC motorok nagyobb hatásfokra, adott térfogaton nagyobb leadására képesek, mint az elektromosan gerjesztett párjuk, miközben a mechanikai felépítésük egyszerűsödik. A kefések DC motorok forgórésze egyenáramú gerjesztésű. Ahhoz, hogy a kívánt nyomaték leadása folyamatos legyen és a motor forgásban maradjon, a forgórész mezeje egy mechanikus kommutátor segítségével a forgórész pozíciójától függően elforgatásra kerül, így az állórész és forgórész mezők szinkronban maradnak. A kommutátor az állórészre rögzített két szén keféből és a forgórészhez rögzített szegmensekből áll, ezek a szegmensek vannak a forgórész tekercseinek végpontjaira kötve [14]. Ez a kommutátor azonban a súrlódó elemek kopása miatt korlátozott élettartamú, önmagában teljesítmény veszteséget okoz, nagy fordulatszámon pedig szikrázás és megnövekedett elektromágneses kibocsátás fellépését eredményezi [32].

Az utóbbi évtizedekben a félvezető technológiák nagymértékű javulása következtében egyre elterjedtebben kerülnek használatba a kefe nélküli állandómágneses szinkronmotorok, amelyek kommutációja elektronikus kapcsolókkal valósul meg. Ezek közös jellemzője, hogy, a kefések állandómágneses DC motorokkal ellentétben a mágnesek a forgórészen, míg a tekercselés az állórészen található, ezzel a mechanikai súrlódás oka a csapágyakra korlátozódik [11], [32], [14], [17]. A kefe nélküli állandómágneses szinkronmotorok (PMSM - Permanent Magnet Synchronous Motor) között két típust nevezhetünk meg az általuk a motor csatlakozásokon indukált elektromotoros erő alakja szerint, a kefe nélküli váltóáramú AC (BLAC - brushless AC) és a kefe nélküli egyenáramú DC (BLDC - brushless DC) motort. A BLAC motor szinuszos, a BLDC trapéz hullám alakú feszültséget indukál [17], [14]. A BLDC motorok számos helyen kerülnek felhasználásra, többek között ventilátorok, szivattyúk, rádióvezérelt drónok, gépjárművek és elektromos kéziszerszámok hajtására alkalmazzák őket előszeretettel. A BLDC

motorok üzemkor az állandómágneses kefések DC motorokkal azonos okból szükséges az álló- és forgórész mezejének szinkronban tartása. Ezt a feladatot az elektronikus kommutátor a forgórész pozíciójától függően előállított állórész tekercsek gerjesztésével végzi el. Alapvető fontosságú tehát a forgórész pozíciójának ismerete, amely általában különböző elven működő érzékelők segítségével valósul meg (például forgójeladó, Hall elem stb.) [32], [17], [10]. A pozíció érzékelők azonban számos hátrányos tulajdonsággal rendelkeznek. A Hall érzékelők által előállított jel pontossága nagyban függ a Hall elemek szerelési pontosságától [6]. Az érzékelők extra költséget, komplexitást, meghibásodási lehetőséget és megnövekedett méretet adnak a rendszernek [29], [32], [10]. A Hall elemek megbízható működése függ az üzemi hőmérséklettől, használatuk újabb vezetékek, áramköri elemek felhasználását teszi szükségessé [32], [10].

A szenzorok okozta hátrányos tulajdonságok kikerülése érdekében különböző elven működő szenzor nélküli forgórész pozíció meghatározási technikákat fejlesztettek ki [18]. Ezekről bővebben később, a 4.2. "Az érzékelő nélküli motor hajtás" alfejezet ad beszámolót. Az egyes módszerek rájuk jellemző előnyökkel és üzemi korlátokkal bírnak, de a szenzor nélkül történő BLDC kommutációt előszeretettel használják ventilátorok és szivattyúk üzemeltetéséhez is.

A BLDC motorok irányításában gyakran mikrovezérlőket, jelprocesszorokat alkalmaznak. A mikrovezérlő egy olyan egy chip-re integrált rendszer, ami egy (vagy több) processzort, memóriát (legalább adatmemóriát), és egyéb, az adott felhasználási céltól függő perifériát tartalmaz. Az egységek ilyenfajta aggregálása helyet és alkatrészmennyiséget spórol [12]. A mikroprocesszorok utasítás készletének megvalósítása nagymértékben befolyásolja a központi egység CPU utasítás feldolgozási sebességét, és a mikrovezérlő/mikroprocesszor félvezetőben elfoglalt méretét egy adott megvalósítási technológián. A RISC-V processzor architektúra az utóbbi években egyre jobban kezd elterjedni. Köszönhető ez a RISC-V utasításkészlet nyílt és ingyenesen felhasználható architektúrájának (ISA Instruction Set Architecture). A RISC-V architektúra alap utasításkészletének használatakor egy kompakt és viszonylag egyszerű processzor készíthető. Ez az alap utasításkészlet a szabvány alapján további utasítás kiegészítőkkel bővíthető, így a tervezett processzorok a legegyszerűbb mikrovezérlőtől nagyteljesítményű sokmagos processzorokig terjedhetnek [30].

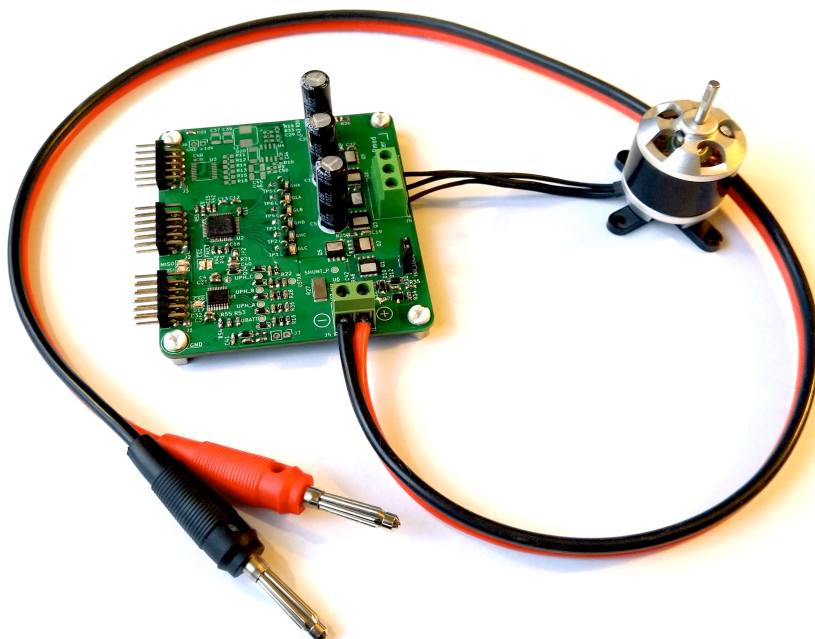
A beágyazott rendszerek tervezése során nagy szerepet kap az egyes funkciók, feladatok megoldásainak minél kisebb helyen, minél kevesebb alkatrész felhasználásával való megvalósítása. Számos integrált áramköri megoldás létezik a BLDC motorvezérlő rendszerek terén. A Texas Instruments a DRV8305-el egy chipbe integrálja a teljesítmény MOSFET-ek meghajtásához szükséges áramköröket, árammérő műveleti erősítőket és biztonsági funkciókat [8]. Az Allegro MicroSystems számos kifejezetten BLDC motor meghajtásra kifejlesztett IC-vel rendelkezik, ezek nemcsak a MOSFET meghajtását végzik, hanem a motor kommutációt és egyéb szabályozási feladatokat is [19]. Infineon az Arm Cortex-M3 processzort tartalmazó TLE987 mikrovezérlőben integrál MOSFET meghajtó áramköröket, kapcsolódó funkciókat és egy árammérő erősítőt, valamint a BLDC-k szenzoros és szenzor nélküli hajtását segítő jelfeldolgozó funkciókat [27].

Jelen dolgozat egy BLDC motor vezérlését megvalósító beágyazott rendszer tervezését mutatja be. A 2. fejezet a BLDC motor meghajtásához szükséges teljesítmény áramkört és jelfeldolgozó áramköröket tartalmazó kártyát mutatja be. A 3. fejezet a fejlesztéshez szükséges RP2040 mikrovezérlő alapú fejlesztői kártya rövid bemutatását szolgálja. A 4. fejezet az FPGA-n megvalósított, Verilog nyelven leírt MotVez motorvezérlő logika bemutatását tartalmazza. A dolgozat központi témájaként az 5. fejezetben bemutatásra kerül a szintén FPGA-n megvalósított, Verilog nyelven leírt RISC-V alapú, a MotVez logikát perifériaként alkalmazó egyszerű mikrovezérlő, melynek a LovagSoC fantázianevet adtam.

2.

A BLDC motor meghajtó kártya bemutatása

A BLDC meghajtó kártya egy szabványos Pmod felületen keresztül csatlakoztatott három fázisú inverter. Az áramkör tervezésére azért volt szükség, mert az FPGA-n történő motor vezérlő logika fejlesztéséhez elengedhetetlen, hogy az valós környezetben ki legyen próbálva. Ezen kívül a tervezett áramkör kompakt kivitelben tartalmaz más, szükséges áramköröket a motor meghajtásához és diagnosztikájához. Az áramkör könnyen csatlakozható és leválasztható az FPGA-ról, az elterjedt Pmod tűsoros kiosztás használatának köszönhetően pedig más FPGA-hoz vagy egyéb fejlesztői eszközhöz is csatlakoztatható. Ez azt jelenti, hogy az áramkör jövőbeni fejlesztési folyamatoknak is része lehet és ez is befolyásolta az áramkörbe tervezett egységek kiválasztását.

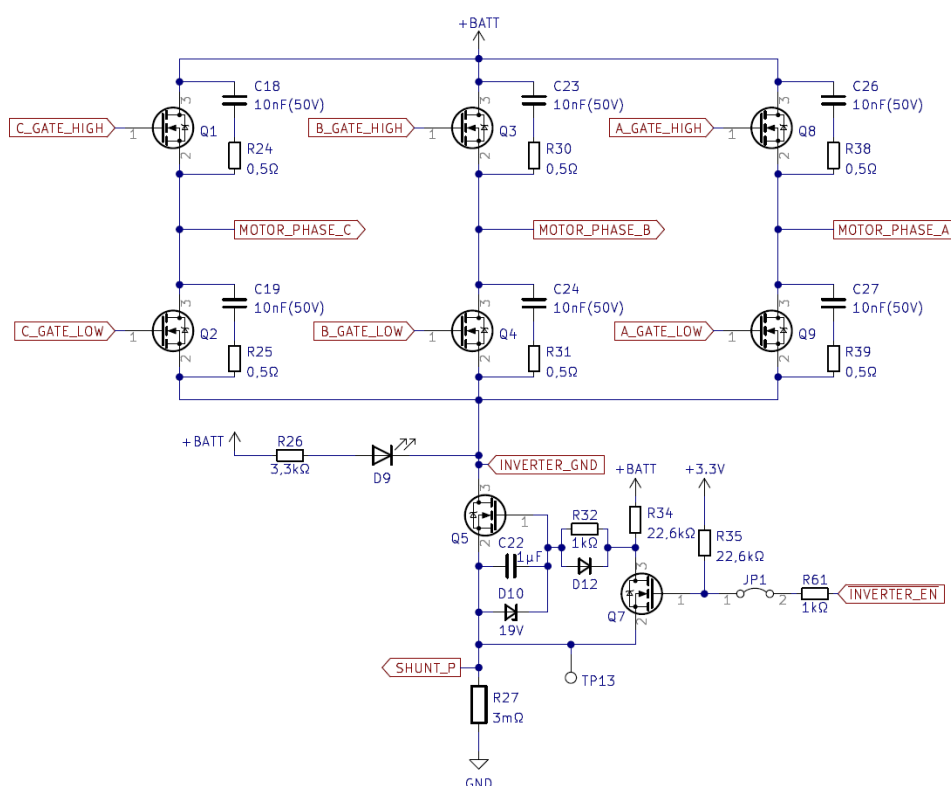


1. ábra: Az FPGA Pmod inverter kártya.

A rendszer főbb egységei Az áramkör legfontosabb egysége a három fázisú H-híd kapcsolás, amely a motor számára előállítja a három fázisú feszültség hullámot. A MOSFET-ek megfelelő kapcsolásához szükséges egy FET meghajtó áramkör. Ez végzi a FET-et kapcsoló jelek szintváltását és erősítését pufferekkel. A motor szenzor nélküli vezérléséhez szükséges jelet a fázisokra csatolt három komparátor adja. A rendszer állapotának felügyeletéhez szükséges analóg jelet egy A-D átalakító továbbítja az FPGA felé.

2.1. Az áramkör bemutatása

Három fázisú H-híd



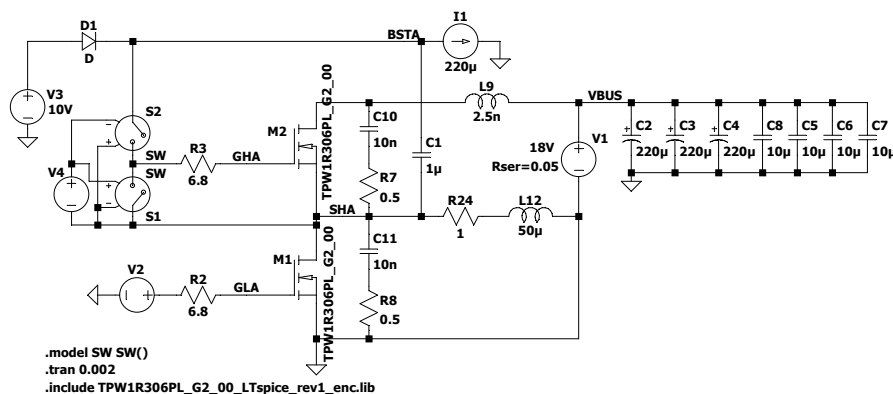
2. ábra: A három fázisú H-híd kapcsolás RC csillapítókkal és az előtöltő áramkör.

A kefe nélküli motorok leggyakoribb kapcsolási topológiája három fázistekercset csillag topológiába kapcsolt tekercs. A motor meghajtása a három fázistekercs kivezetésein keresztül, háromfázisú feszültség táplálásával történik. A teljesítmény átvitelére és kapcsolására többfajta megoldás is lehetséges, ilyen a relé, a bipoláris tranzisztor, az IGBT és a MOSFET. Ezek közül a választott alkatrész a MOSFET, mert magas kapcsolási frekvencián és viszonylag nagy teljesítménnyel üzemeltethető.

A H-híd hat MOSFET-ből áll, ezek három félhidat alkotnak, amelyek külön-külön a tápfeszültségre vagy a földre kötik az adott motor fázist, a FET-ek kapcsolási állapotától függően [9].

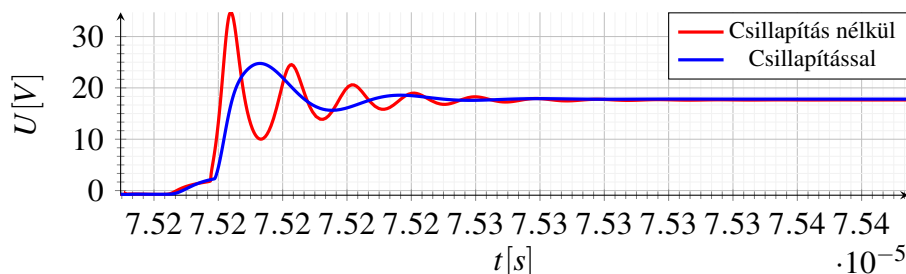
FET csillapító RC kör

A modulált feszültség előállításakor a PWM-el kapcsoló FET-ek kapcsolási idejének minimalizálására kell törekedni a kapcsolási veszteség minimalizálása érdekében. A gyorsan kapcsoló, nagy áramot vezető FET nyitásakor a vezetékkel parazita induktivitása miatt nagyméretű feszültség tüskék és nagyfrekvenciás oszcilláció keletkezhet [23]. Ezek a tüskék kárt tehetnek egyes alkatrészekben, illetve elektromágneses interferenciát okozhat.



3. ábra: A FET csillapító és tápkondenzátorok LTSpice szimulációjának kapcsolása.

A tüskék csillapításához és a nagyfrekvenciás zavarok elkerülésének érdekében széles körben elterjedt megoldás egy RC szűrő használata a FET drain-source lábaival párhuzamosan kötve, a 2. ábrán látható módon.



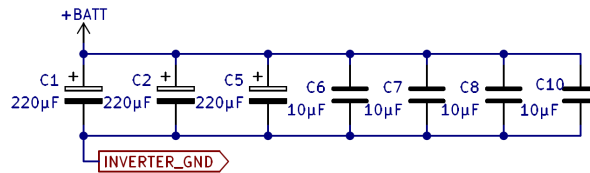
4. ábra: A FET csillapító áramkör LTSpice szimuláció eredménye.

Az ellenállások és kondenzátorok értékeit a 3. ábrán látható szimuláció összeállításával és futtatásával választottam ¹, a fő szempont a feszültségtűskék csökkentése és az RC kör veszteségének minimalizálása volt. A 4. ábra a terhelt félhíd *SHA* pontjának feszültségét mutatja az alsó FET kikapcsolásakor.

Tápkondenzátorok

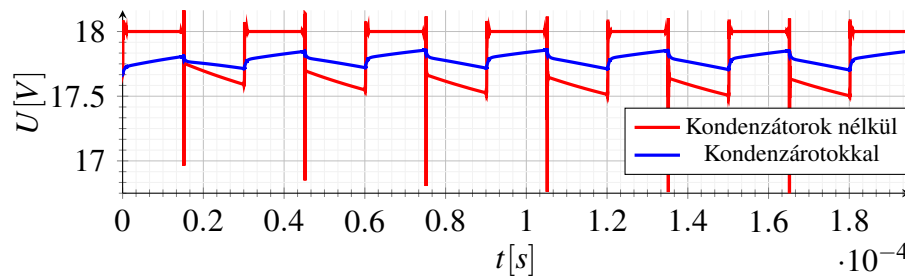
A tápkondenzátorok felelősek a híd kapcsolás nagy frekvenciájú terhelése által okozott feszültség-ingadozások csökkentéséért.

¹A szimulációs áramkör a DRV8300 működési elve alapján készült, az utánhúzó kondenzátor szivárgó áramának 220μ értékét az adatlappól vettem.



5. ábra: A tápkondenzátorok kapcsolása.

A nagy kapacitású elektrolit kondenzátorok feladata a PWM okozta hullámszám csökkentése, a kerámia kondenzátorok feladata pedig a feszültségcsúcsok kiszűrése, az alacsony mértékű belső ellenállásuknak köszönhetően.



6. ábra: A tápkondenzátorok LTSpice szimulációjának eredménye.

A választott kondenzátorok működését szimulációval is ellenőriztem, a 3. ábrán látható kapcsolással. A 6. ábrán látható eredmény a terhelt félhíd busz feszültségét mutatja (VBUS pont). Kondenzátorok használatával nagymértékben csökkent a feszültség hullámszáma.

Híd engedélyező és kondenzátor előtöltő áramkör

A 2. ábrán a Q5 jelölésű FET szerepe kettős: a híd engedélyezését/tiltását, valamint a tápkondenzátorok előtöltését végzi. Az inverter csak akkor üzemelhet, ha ez a hetedik MOSFET bekapcsol. Ez fejlesztés közben hasznos biztonsági funkció, például PWM mintázatok kipróbálásakor, amikor az egyes félhidak esetleg hibás minta miatt egybezárhatnak.

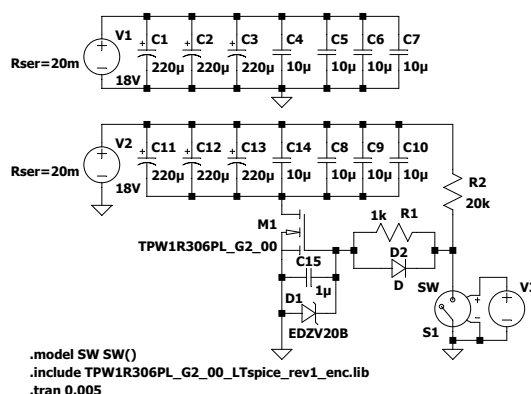
A tápkondenzátorok előtöltése a tápfeszültség csatlakoztatásakor a kondenzátorok töltési áramának csökkentését jelenti. A túl magas áramimpulzus egyes alkatrészek élettartamának csökkenéséhez vezethet, vagy a táp csatlakozók összehegedéséhez [5].

A tervezett előtöltő áramkör a hetedik FET bekapcsolását RC körrel lassítva korlátozza a tápkondenzátorok töltési áramát [4]. A bekapcsolás lassítása miatt a FET a híd engedélyezése után 3,5ms késleltetéssel kapcsol be, a FET-en jelentős, de elfogadható mértékű hő keletkezik.

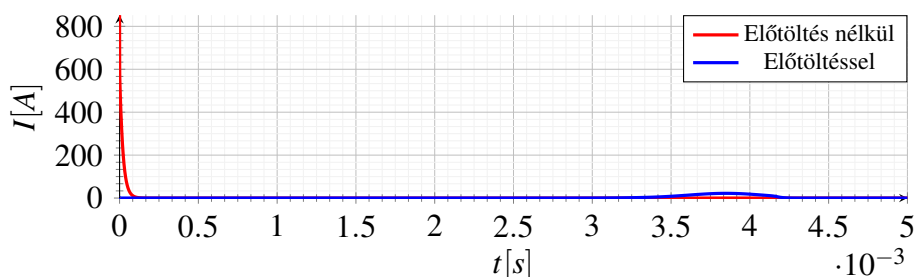
A tervezett áramkört szimulációval is ellenőriztem. A 8. ábrán a DC-link áram a kondenzátor előtöltés használatával nagymértékben csökkent. Az eredmények alapján a kapcsolás megfelelőnek bizonyult.

FET meghajtók

A kiválasztott teljesítmény FET-ek kapcsolásához külső áramkör szükséges. Ennek két fő oka, hogy az FPGA kimenete nem szolgáltatja a megfelelő feszültséget a FET teljes bekapcsolásához,



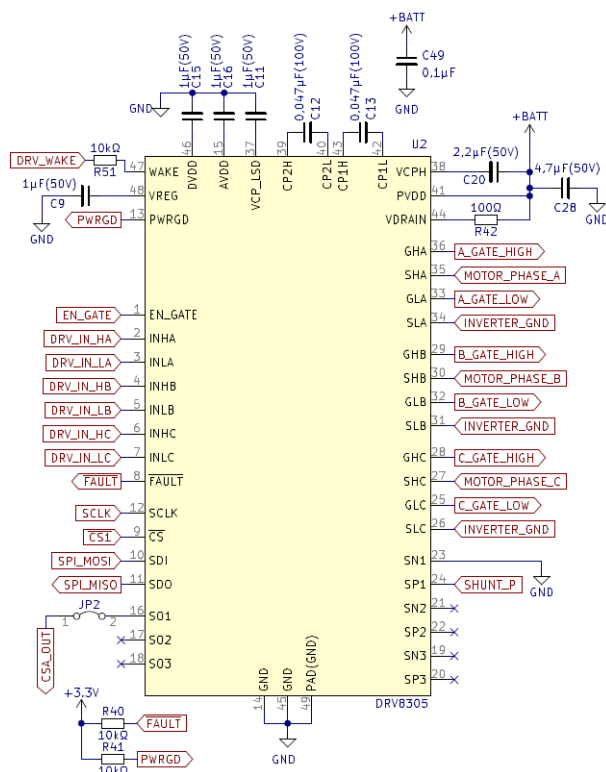
7. ábra: Az előtöltő áramkör LTSpice szimulációjának kapcsolása.



8. ábra: Az előtöltő áramkör LTSpice szimuláció eredménye.

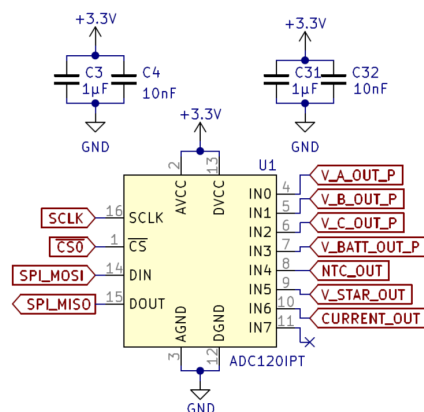
valamint a FET gate láb megfelelő sebességű feltöltéséhez szükséges áramerősséget sem képes leadni. Az áramkörhöz két FET meghajtót is választottam. Erre azért volt szükség, mert a DRV8305-ből csak egy darab volt elérhető, s ha az elromlik, akkor lehetőséget biztosít a második meghajtó. Ezen kívül előnyös, hogy nem csak egyfajta chip-el működik a rendszer, hiszen egyik, vagy másik nehezen elérhetővé válhat a jövőben. A DRV8305 magas szintű integráltsággal rendelkezik, hozzá nem szükséges külső feszültség átalakítás, valamint a három beépített műveleti erősítő kompakt áram mérést tesz lehetővé. Az IC további előnye az SPI kommunikáción keresztül módosítható beállítás regiszterek, ezek tovább finomítják a működést. A felső oldali tranzisztorok bekapcsolásához a DRV8305 töltéspumpa áramkört használ. Ezen meghajtó támogató alkatrészeit az adatlap ajánlásai alapján választottam és kötöttem be. Figyelmet kellett szentelni a kondenzátorok névleges üzemi feszültségére, ezeket jóval a tápfeszültség fölé kellett választani.

A DRV8300 egy jóval kevésbé összetett IC, de a szükséges külső alkatrészek annál számosabbak. A tápellátást külső feszültség átalakítóval kellett biztosítani, mivel az IC táp feszültsége kerül a FET-ek gate lábára. A FET-ek gate áramát ellenállással kellett korlátozni. A DRV8300 meghajtó kondenzátor utánhúzó áramkörrel kapcsolja be a felső oldali FET-et. Az ehhez szükséges dióda beépített a chipben, csak a kondenzátort kellett ehhez biztosítani. Ezen áramkör olcsó, kevés alkotóelemű, hátránya viszont, hogy nem képes 100%-os kitöltési tényezőn üzemeltetni a hidat, melynek oka az utánhúzó kondenzátor szivárgó árama. Az adatlap utasítása alapján [7] az utánhúzó kondenzátort a maximális, $10\mu F$ méretűre választottam, hogy alacsonyabb PWM frekvencián és magasabb kitöltési tényezőn is üzemelhesen a híd.



9. ábra: A DRV8305 FET meghajtó kapcsolása.

2.1.1. Analóg-digitális átalakító

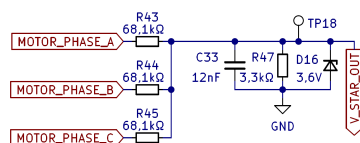


10. ábra: Az ADC120 A-D átalakító kapcsolása.

Az analóg-digitális átalakító feladata, hogy a motor és a rendszer állapotáról információt szolgáltatson az azt feldolgozó elem (jelen esetben az FPGA) részére. A választott vizsgálandó mennyiségek a H-híd árama, a tápfeszültség, hőmérséklet mérés, a motor fázisfeszültségei és a virtuális csillagponti feszültség. A motor feszültségmérésének szükségét az adja, hogy a forgórész pozíció meghatározására akár az A-D átalakító is felhasználható, így a komparátorok nem feltétlenül szükségesek. (Emellett más, komplexebb hajtásoknál szükségessé válhat a motor effajta megfigyelése.) A választott A-D átalakító, az ADC120 rendelkezik a szükséges csatornaszámmal, s ezek mintavétele tetszőleges sorrendű lehet. A maximális mintavételi sűrűsége 16MHz-es SPI órajellel tehetünk szert, így akár 500kHz-el mérhetünk, a mért pont

kimenő impedanciájától függően. A mért mennyiségek SPI kommunikációs protokollon keresztül olvashatók ki, így az ADC120 és a DRV8305 egy SPI buszra csatlakoznak. Közülük chip kiválasztó jelekkel választhatunk. Az ADC120 áramkörét az adatlapi ajánlásoknak megfelelően készítettem el. A tápfeszültséget az FPGA NYÁK-ja biztosítja.

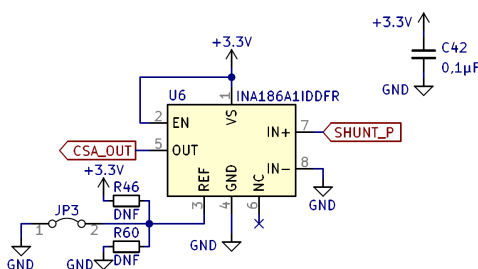
Jel kondicionáló áramkörök



11. ábra: Virtuális csillagpont áramkör feszültségosztóval és RC szűrővel.

Mivel az A-D átalakító csatornáinak maximális bemeneti jelszintje a 3,3V-os tápfeszültséggel megegyezik, szükséges volt a mért motor feszültségek szintváltása. Ezt egyszerű feszültségosztókkal oldottam meg, mert ebben az esetben azok kimenő impedanciája nem befolyásolta jelentősen a mért értékeket. Az inverter áramának mérése söntés ellenálláson keresztül történt, mert az olcsó, egyszerű és viszonylag pontos megoldás. A feszültségosztókra és az erősített áramjelre aluláteresztő RC szűrőket helyeztem, ezzel próbálva javítani a jelminőséget.

Áram mérőerősítő A söntés ellenálláson eső feszültség méréséhez erősítésre van szükség, ezt egy műveleti erősítő végzi. A DRV8305 tartalmaz beépített műveleti erősítőt árammérés céljából, de mivel az választhatóan szerelhető alkatrész, és a beépített erősítő alacsony áramok mérésére viszonylag pontatlan, ezért szükség volt külső OPA hozzáadására.



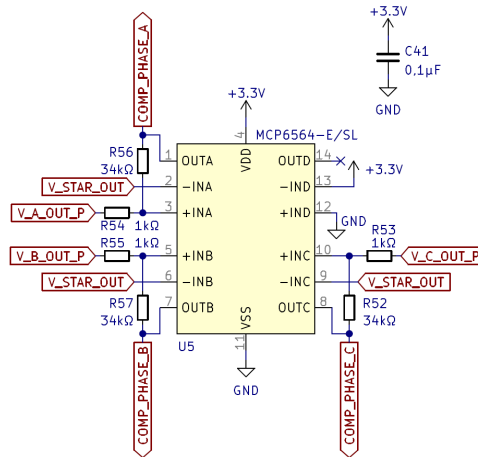
12. ábra: Az árammérő műveleti erősítő áramköre.

A választott alkatrész kis méretű és minimális külső elemet igényel. A referencia láb emelésével két irányú árammérést is végezhetünk, de ez jelen esetben nem szükséges. A választott erősítés $G_{V/V} = 25 \frac{V}{V}$.

Komparátorok

A komparátorok a motor fázisfeszültségeit hasonlítják a virtuális csillagponti feszültségéhez. Az érzékelő nélküli hajtás egyik megoldása ezen komparátorok jeleinek felhasználása, mint pozíció jelek. A jel mintázata hasonló a Hall érzékelőkkel kapott jelhez.

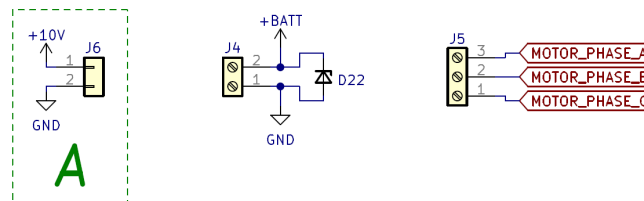
A választott IC négy komparátort integrál egybe, ezzel csökken a NYÁK-on elfoglalt hely. A komparátorok hiszterézishez szükséges visszacsatolással vannak ellátva, ezek csökkenthetik a komparált jelek zajszintje miatti prelegő kimenetet. Ha nem használunk hiszterézist, akkor a bemenettel sorban levő ellenállást át kell hidalni.



13. ábra: A Back-EMF komparátorok kapcsolása.

Csatlakozók

A teljesítmény áramkör tűskecsatlakozókkal kapcsolódik az FPGA kártyához. Ezek a csatlakozók szabványos Pmod kiosztásúak, így az áramkört más FPGA-hoz vagy akár mikrovezérlős áramkörhöz is csatlakozhatjuk. Az átküldött jelek a DRV8305 állapotjelző és vezérlő bemenetei, a gate vezérlő jelek, az SPI busz jelei, a komparátor kimenetek és az invertert engedélyező FET gate jele. Az áramkör táp bemenete és a motor kivezetései csavart terminálokkal csatlakoznak.



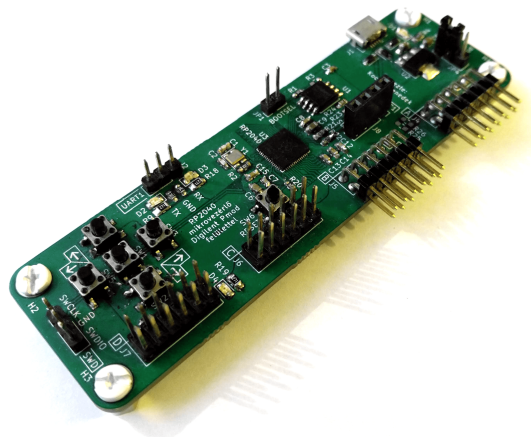
14. ábra: A táp ki- és bemeneti csatlakozók.

A táp terminállal párhuzamosan TVS diódát kötöttem az esetleges feszültségtranziensek elleni védelemhez.

3.

MikroKon

A MikroKon (MikroKontroller) kártya egy RaspberryPi RP2040 alapú fejlesztői kártya Pmod csatlakozókkal.

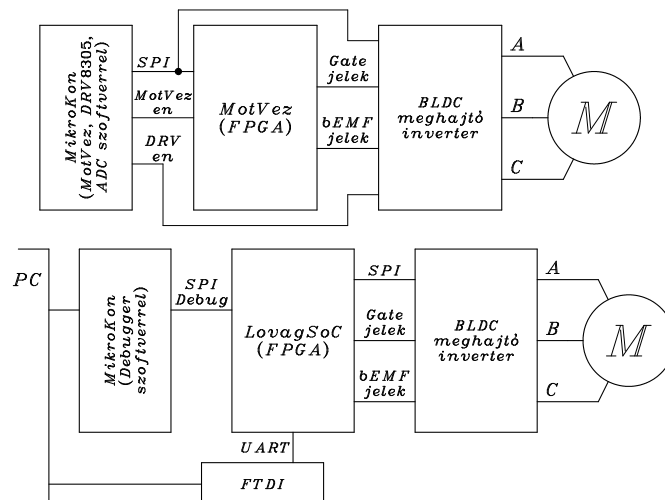


15. ábra: A MikroKon fejlesztői kártya fotója.

Az FPGA-ra való fejlesztés kezdetekor szükséges volt egy külső mikrovezérlőre, ami rendelkezik a teszteléshez szükséges kommunikációs protokollal és kimenetekkel. Lehetséges megoldás volt egy belső, lágymagos processzor használata, de a tervezett MotVez logikának önállóan is használhatónak kell lennie. Ezt csak külső áramkörrel lehet igazolni. A külső áramkört szerelőlapon vagy próbapanelen is elkészíthetjük, de csak a nyomtatott áramköri megvalósítással kapunk kompakt elrendezést, ami egyszerűen csatlakozható és leválasztható. Az áramkör nemcsak FPGA-n történő tervezés során használható, önmagában is beágyazott szoftver fejlesztésekor hasznos. A kártya központi egysége az RP2040 mikrovezérlő. Ennek tápellátását lineáris regulátor végzi. Az RP2040 órajelét külső oszcillátor szolgáltatja, a program memória külső QSPI chipről kerül beolvasásra. A NYÁK-on Pmod kiosztású csatlakozók, UART, I2C és SWD tűskecsatlakozók helyezkednek el. A mikrovezérlő programozása és soros kommunikáció USB csatlakozón történik. A felhasználói bevitel az öt nyomógombon keresztül is történhet.

3.1. Kapcsolási rajz

A kapcsolat alapját a RaspberryPi RP2040 példa hardver adja [13]. Ennek okai a fejlesztés időszükségletének csökkentése. A MikroKon kapcsolása a példa hardver jelentősen módosított változata, a NYÁK teljesen egyedi. Az RP2040 3,3V feszültséget igényel. Mivel a kártyán semmi nem igényel nagy teljesítmény átvitelt, lineáris regulátor is használható. A csatlakoztatott 5V feszültség jelenlétét LED jelzi. A MikroKon kártya nemcsak az USB csatlakozó, hanem a csatlakoztatott FPGA-n keresztül is táplálható. Ez esetben a 3,3V-ot a Pmod csatlakozó biztosítja. A váltást egy jumper elágazás végzi. Az USB csatlakozó bemeneteire ESD diódákat helyeztem, hiszen itt várható folyamatos emberi érintés használat során.



16. ábra: A MikroKon szerepe a MotVez (felül) és a LovagSoC (alul) teszt összeállításában.

RP2040 mikrovezérlő Az RP2040 egy ARM Cortex M0+ processzor alapú mikrovezérlő. A CPU 133MHz-en fut és két maggal rendelkezik. A program memória külső QSPI flash-ből töltődik be. A mikrovezérlő rendelkezik a szükséges kommunikációs protokollokkal és általános kimenetekkel. A chip 264kB belső RAM-al rendelkezik, amely több mint elegendő igen bonyolult alkalmazásokra is. A beépített USB felületnek köszönhetően könnyen programozható külső programozó kártya használata nélkül. Habár az RP2040 rendelkezik belső rezgőkörrel, a komolyabb időzítési követelmények elérésének érdekében külső kristály oszcillátor került alkalmazásra.

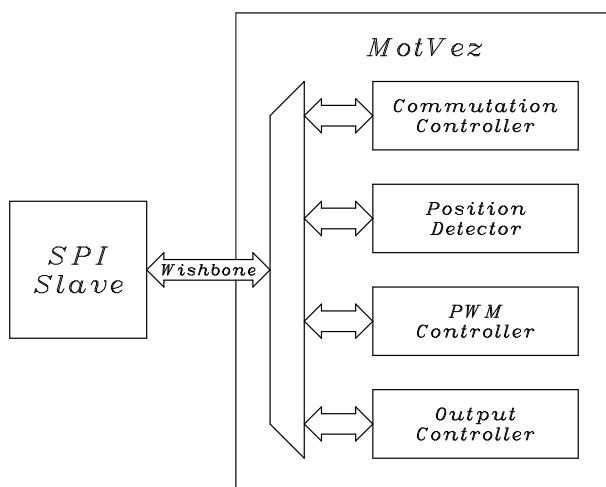
Egyéb csatlakozók, gombok Egyes kommunikációs felületek külön kivezetésre kerültek a könnyebb hozzáférhetőség érdekében. Az I2C felület egy OLED kijelző meghajtását szolgálja jelen esetben. Az SWD (Serial Wire Debug) pedig külső debuggolást tesz lehetővé. A felhasználói bevitelhez öt nyomógomb biztosított. ezek felhasználása általános lehet, de a jelen esetben tervezett szoftvernél a grafikus felhasználói felület léptetését végzik. Az RP2040 GPIO kimeneteit csatlakoztattam a Pmod kiosztású tűkesorokhoz. Ezeken az A csatlakozó kivételével a forraszhíddal köthető az FPGA-hoz.

4.

MotVez

4.1. Rendszerkép

A MotVez rendszer alapvetően négy fő modulból, a Position Detector-ból, a Commutation Controller-ből, a PWM Controller-ből és az Output Controller-ből áll. Ezeknek a moduloknak különböző feladataik és beállításaik vannak. Egymás közt közvetlen jelátvitellel továbbítanak szükséges adatokat, irányító jeleket. Ez a négy modul egy belső buszrendszerre van kötve, ami a MotVez-en kívüli adatfolyamot teszi lehetővé.

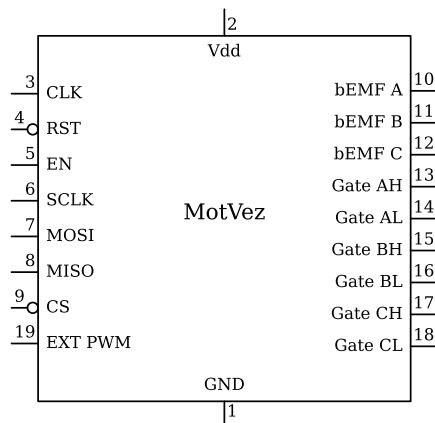


17. ábra: A MotVez rendszerdiagram.

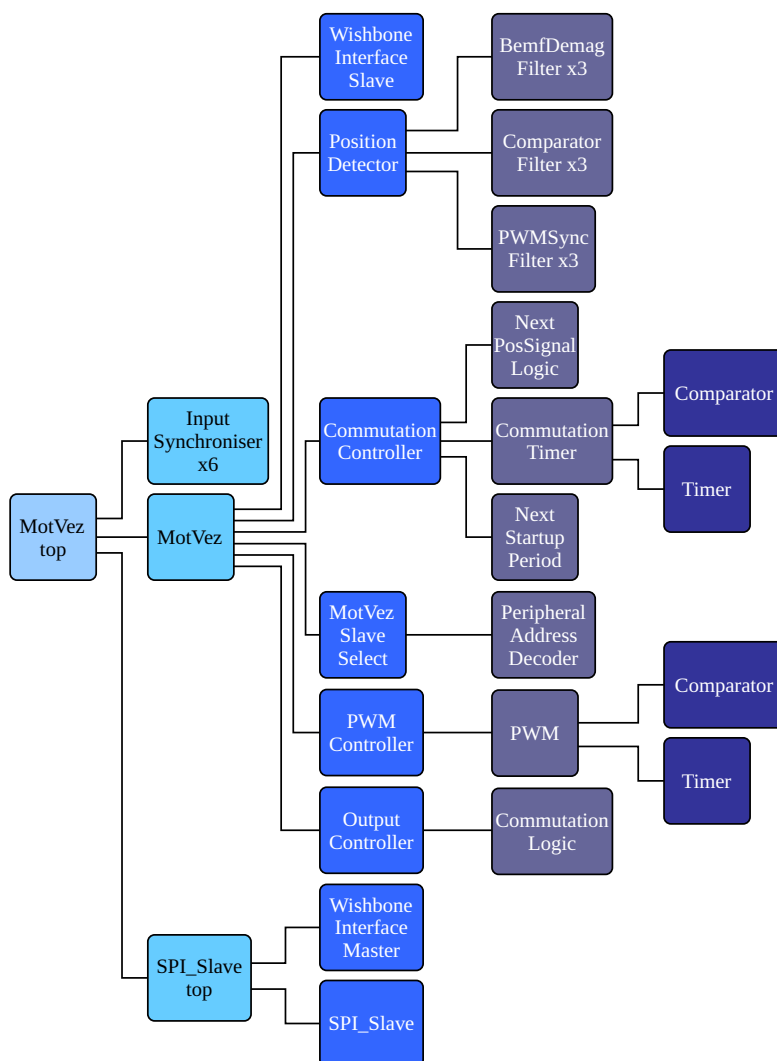
Ez a belső busz aztán egy busz átalakító híd segítségével egy szabványos busz protokollra kapcsolódik. Mivel ez a fejezet közvetlenül a MotVez rendszer felépítésével és tervezésével foglalkozik, ezért a továbbiakban a 17. ábra szerinti rendszer kerül bemutatásra.

Bár az SPI Slave nem része a MotVez rendszernek, annak teszteléséhez és működtetéséhez ez a modul minimális szükséglet, hiszen a külvilággal ez biztosít adatátvitelt. Így ennek a leírása is e fejezet része. Mivel a MotVez egy szabványos busz csomaggal rendelkezik, alkalmazható egy bonyolultabb rendszer egyik tagjaként, perifériájaként is.

A 19. ábra bemutatja, hogy az egyes egységek, modulok hogyan ágyazódnak egymásba. Az



18. ábra: A MotVez + SPI különálló logika áramköri szimbóluma.



19. ábra: A MotVez + SPI különálló logika modul hierarchiájának ábrája.

ábrán az egyes elemek mind egy-egy Verilog modult (forrásfájlt) képviselnek. ¹

¹A MotVez bemutatásának olvasása során érdemes időnként visszatérni ehhez az ábrához eligazításért.

4.2. Az érzékelő nélküli motor hajtás

A MotVez egység fő funkciója egy BLDC motor szenzor nélküli meghajtása. Ez okból az alábbiakban bemutatásra kerül az alkalmazott forgórész pozíció meghatározási elv lényege, működése.

4.2.1. A BLDC motorokról

A kefe nélküli, vagy BLDC motorok az állandómágneses szinkrongépek (PMSM - permanent magnet synchronous motor) csoportjába tartoznak. A BLDC motorokat a PMSM-től megkülönböztető tulajdonság a forgórész és állórész közti légrés fluxussűrűség hullámformája, amely a BLDC-k esetében trapéz alakú. [14] Ez nem szabályszerű kijelentés, hisz sokszor ez a trapéz forma erősen torzult, inkább hasonlít szinuszos hullámhoz.

A BLDC motorok alapvetően egy forgórészből és egy állórészből állnak. A forgórészen, vagy abba beágyazva találhatók az állandó mágnesek, a forgórész anyaga általában laminált vaslemezekből áll. Az állórészen találhatók a pólustekercsek, a BLDC motorok esetében egy tekercs általában csak egy pólustörzset ölel át. A háromfázisú tekercselés általában csillag topológiában van bekötve, de a csillagpont kivezetésére általános hajtás tervezésénél nem érdemes számítani, legtöbbször csak a három fázis vezeték kivezetett [32].

A BLDC motorok hajtása

Az állandómágneses szinkronmotorok forgatásához háromfázisú feszültség szükséges. A feszültség lehet szinuszos, illetve négyzethullám alakú. A BLDC motorok esetében szinuszos hajtás előnyei egyebek közt az egyenletes nyomatékleadás, a csökkentett motor zaj és a hajtási technikától függően a dinamikusabb viselkedés. A négyzethullámú, vagy „blokk”, hatlépéses kommutációs módszer a szinuszos megoldáshoz képest jóval egyszerűbben megvalósítható, nagyobb maximális nyomaték érhető el vele, egyszerűbb feladatokra sokkal elterjedtebben használt, mint a szinuszos kommutáció [1]. A MotVez motorvezérlő logika a hatlépéses kommutáció alapján végzi a motor hajtását, főként az [alkalmazott forgórész pozíció meghatározási módszer](#) miatt.

A háromfázisú feszültség előállításához általában háromfázisú H-hídkapcsolást alkalmaznak. Ez a kapcsolás három félhídból, hat elektromos kapcsolóból áll. Ezek a kapcsolók általában tranzisztorok, ezen belül lehetnek BJT, IGBT vagy MOSFET típusúak. Az egyes tranzisztorokat a forgásiránytól és forgórész pozíciótól függően megfelelő sorrendben nyitjuk, illetve zárjuk, ezek a kommutációs lépések (lásd 1. táblázat). [17] A három fázisú feszültség előállításához tehát hat kimenő jel generálása szükséges, ezt a feladatot látja el a MotVez logika a bemeneti paraméterek és információk alapján.

A DC motorok hatásos működtetéséhez szükséges a DC-AC átalakítás során előállított feszültség fázisának szinkronizálása a forgórész mágneses fluxuseloszlásának függvényéhez.

Szögeltérés a fázisfeszültségek és a fluxuseloszlás között csökkenti a motor nyomatékát, egy bizonyos eltérést meghaladva pedig a kommutáció összeomlásához és a motor megállásához vezet [32]. Ez a probléma hagyományos kefék DC motorok esetében a kommutátor szegmensek és a szénkefék kialakításával mechanikailag oldódik meg. A BLDC motorok esetében azonban a kommutáció elektronikus megoldással valósul meg, tehát a szinkront is az elektronikának kell fenntartania, amihez szükséges a forgórész szöghelyzetének ismerete.

A forgórész pozíció meghatározására többféle megoldás létezik, ezek között a forgójeladó (mechanikus, optikai stb.), a Hall érzékelő és a szenzor nélküli módszerek elterjedtek. A szenzor nélküli módszerek előnye többek között a hosszú távon megbízhatóbb működés (nincs szenzor ami elromolhatna), az olcsóbb kivitel és az alacsonyabb szerelési költségek [10]. A MotVez logika képes fogadott Hall érzékelő jeleket is feldolgozni, de a rendszer fókusza egy szenzor nélküli alapú kommutáció megvalósítása.

A BLDC motorok Hall szenzorok nélküli meghajtására számos megoldás létezik, ezekről [18] készített egy széleskörű áttekintést. [3] a motor által generált back-EMF motor csatlakozáson mért értékét A-D átalakítóval mintavételezve, azt a tápfeszültség feléhez hasonlítva kapott pozíció jeleket. [25] hasonló módszert analóg komparátor alkalmazásával és virtuális csillagpont létrehozásával valósított meg. [26] a virtuális csillagpont és a tápfeszültség felét összehasonlítva kapott kommutációs ütemezést, amely az előző két módszerhez képest szélesebb sebességtartományban biztosíthatott üzemet. [29] olyan módszert mutat be, amely a motor tekercsek indukciós együtthatójának forgórész pozíciófüggésén alapszik. A fázisaiba való nagyfrekvenciás jel beoltásával, s az ennek hatására meginduló áram és a motor paraméterek felhasználásával nyer ki forgórész pozíció információt álló helyzetben és alacsony fordulatszámon. [22] a három fázisú H-híd MOSFET szabadonfutó diódáinak vezetési állapotát vizsgálva kapott pozíció információt, ezzel a szenzor nélküli kommutációt a motor indításakor alacsonyabb fordulatszámon kezdhette. [28] kiterjesztett Kálmán-szűrőt alkalmaz a motor nemlineáris egyenletének felhasználásával a forgórész szöghelyzetének közvetlen meghatározására.

A tervezett logikában alkalmazott módszer a fázis back-EMF nullpontmetszés módszere virtuális csillagpont felhasználásával. A módszer egyszerű, alacsony a számítási igénye, nem igényli analóg-digitális átalakító használatát. A módszer választásának legfőbb oka azonban a fejlesztési idő csökkentése, a rendelkezésre álló idő korlátja miatt. ²

4.2.2. A fázis back-EMF nullpontmetszés elve

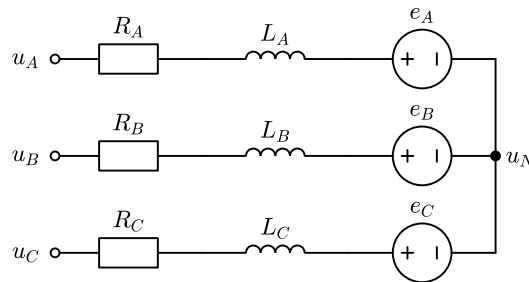
A motor által generált elektromotoros erő (back-EMF - backwards electromotive force) amplitúdója arányos a motor fordulatszámaival, a hullámforma pedig a légrés fluxus eloszlás deriváltjának alakjával azonos, a Faraday-féle indukciós törvény alapján.

Ez a fluxuseloszlás BLDC esetén trapéz hullám alakot vesz fel, melynek fázisa a szöghelyzettől függ. A 4.1. egyenlet a motor egyik fázisfeszültségének egyenletét adja meg [32].

²Későbbi változatokban más módszerek használata is megfontolásra kerül majd.

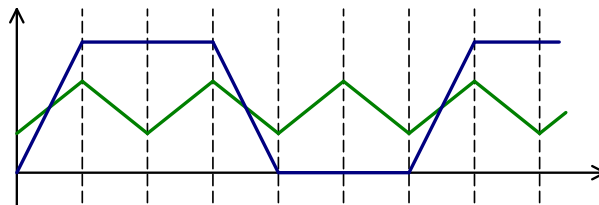
$$u_X = Ri_X + L_X \frac{di_X}{dt} + e_X, \quad (4.1)$$

ahol $X = (A, B, C)$ tekercsek, u_X a motor X fázisának feszültsége, R a fázistekercs ellenállása, i_X az X fázis árama, L_X a fázis indukciós együtthatója és e_X a generált elektromotoros erő (back-EMF). Látható, hogy ha a tekercsben nem folyik áram, a fázisfeszültség megegyezik a back-EMF-el. Ebben az esetben a feszültség vizsgálatával megbecsülhető a forgórész pozíciója. A fázis back-EMF alapú szenzor nélküli pozíció meghatározási módszer a [32]-ben leírtak alapján Hall érzékelőhöz hasonló jelet készít a fázisfeszültség vizsgálatával.



20. ábra: A csillag topológiában kötött háromfázisú motor helyettesítő áramköre.

A MotVez által alkalmazott hatlépéses kommutációs minta alapján minden pillanatban csak két fázis energizált. Mindig a harmadik, semleges fázis terminálon történik a mérés. Mikor a motor mért fázis feszültsége előjelet vált a 21. ábrán látható módon, ez az ún. nullpontmetszés könnyen feldolgozható információt szolgáltat a forgórész pillanatnyi helyzetéről.

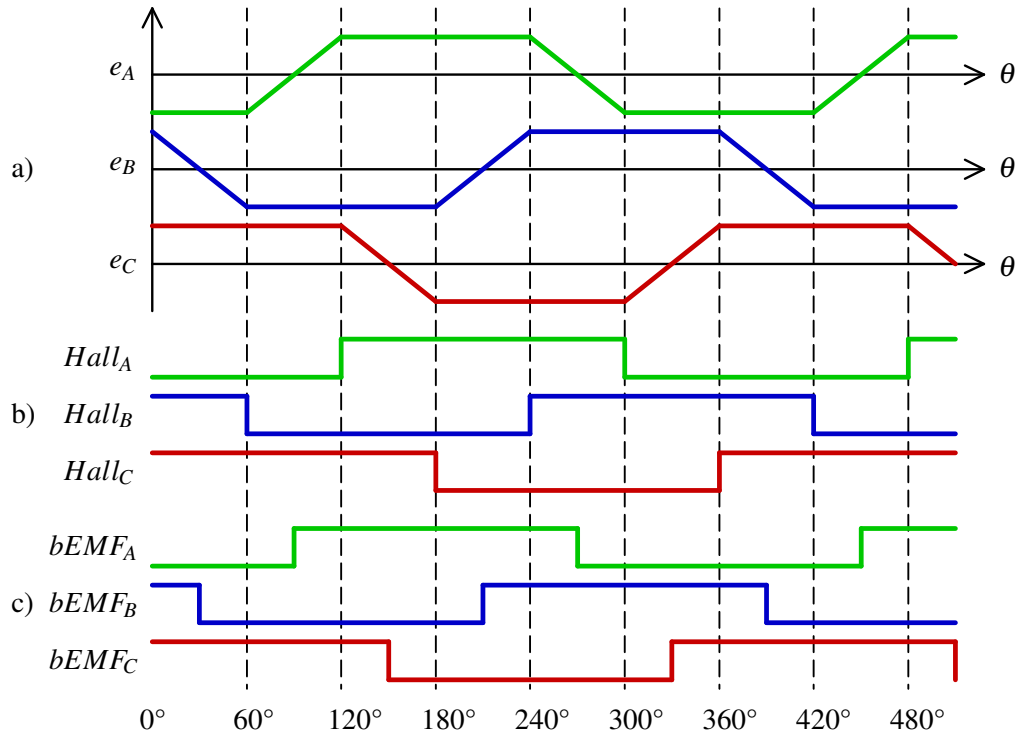


21. ábra: A földhöz vonatkoztatott motor fázis (kék) és virtuális csillagpont feszültségek (zöld).

A 22. ábrán látható, hogy az alkalmazott szenzor nélküli módszer által kapott pozíció jel hasonlít a Hall érzékelő jeleihez, azonban 30 fokkal elcsúsztatott ahhoz képest. A kommutációs pillant meghatározásához tehát 30 fokos késleltetés használata szükséges.

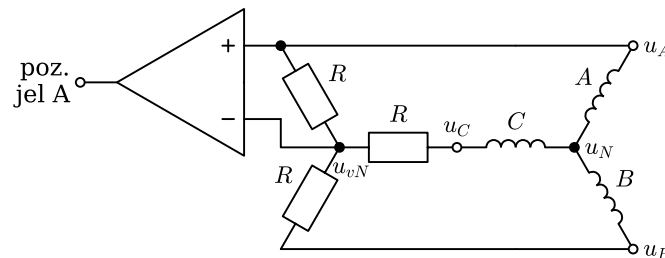
A nullpontmetszés meghatározása a MotVez rendszer használatakor

A szenzor nélküli pozíció meghatározás során szükséges a motor három fázisfeszültségének, vagyis a motor termináljai és a csillagpont közti feszültség vizsgálata. Mint az már említésre került, a BLDC motorok nagy többsége esetén a csillag topológiába kötött tekercsek csillagpontja nem elérhető közvetlenül vizsgálatra. A probléma megoldása a MotVez logika esetén egy egyszerű ellenálláshálózat. A hálózat három azonos értékű ellenállásból áll, melyek a motor



22. ábra: Fázis back-EMF feszültségek (a), Hall szenzor jelek (b), back-EMF nullpontmetszés által generált jelek (c).

tekercseivel azonos csillag topológiában csatlakoznak [24], a 23. ábrához hasonló módon. A létrehozott csillagpont feszültsége a nullpontmetszés pillanatában azonos a motor u_N feszültségével, ez az u_{vN} virtuális csillagpont. A MotVez logika bEMF A, B és C bemenetei (18.



23. ábra: A virtuális csillagpont alapú feszültség komparálás.

22. ábra) a szenzor nélküli pozíció jelek külső generálását igénylik. Ezeket a jeleket három logikai-szintű komparátor állítja elő. Bemeneteik az előállított fázisfeszültségek polaritását vizsgálják, a komparálás eredménye a pozíció jel. (Lásd 22. ábra, c) részlete.)

4.3. SPI Slave

A tervezett motorvezérlő logika egyes üzemi paramétereit SPI protokollal használataival lehet programozni. Ennek szükségességét az adja, hogy az egyes rendszerek, amikben a MotVez használatba állhat, különböző motorokat alkalmaznak különböző felhasználási területeken.

4.3.1. A tervezett SPI felület működése

A tervezett SPI Slave modul egy szolga, vagy „slave” beosztású SPI kommunikációs periféria. Az SPI (Serial Peripheral Interface) soros kommunikációs protokoll egyszerű és viszonylag gyors adatátvitelt tesz lehetővé [12]. Ebben a modulban használt SPI konfiguráció full-duplex, tehát egyszerre történik adat küldés és fogadás. Ennek megfelelően négy kimenet szükséges:

MOSI: Master Out, Slave In. Adatátviteli jel, mester-szolga irányban. Push-pull kimenet.

MISO: Master In, Slave Out. Adatátviteli jel, szolga-mester irányban. Külső felhúzást igényel.

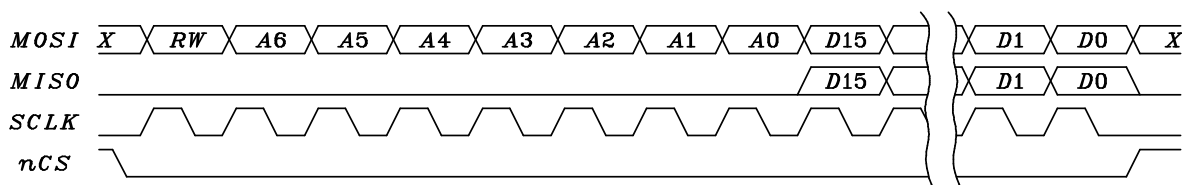
SCLK: Serial Clock. A soros kommunikáció órajele az adatátvitel ütemezéséhez. Push-pull kimenet.

nCS: Negated Chip Select. Szolga periféria választó jel, aktív alacsony. Külső felhúzást igényel.

A tervezett SPI logika órajel polaritása CPOL = 0, tehát alapállapotban az SCLK alacsony. Az átviteli keret felfutó órajel éllel kezdődik és annak lefutó élével ér véget. Az adat bitek írásának fázisa CPHA = 1, tehát minden új bit az SCLK felmenő élén kerül kiírásra, az olvasás pedig a lefutó élen történik.

Egy átviteli keret felépítése

Az adatkeret felépítését a DRV8305 MOSFET meghajtó IC által alkalmazott keret alapján választottam [8], hogy könnyítse a külső MikroKon szoftverének fejlesztését, valamint, mert ez a keret felépítés jól illik a regiszter beállítások elvégzéséhez. Egy adattovábbítás bitszélessége



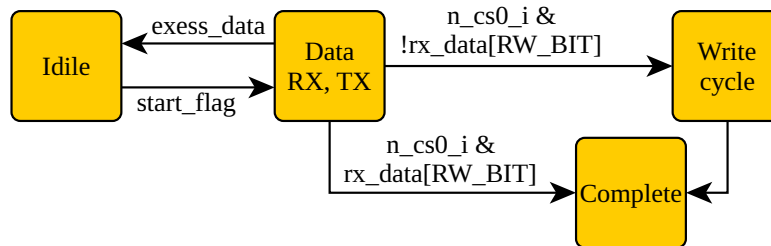
24. ábra: Az SPI Slave adatátviteli keretének idődiagramja.

24bit, ebből a 23-as bit az írás/olvasás választását végzi, [22:16] bitek az adtművelet címét határozzák, [15:0] az átvitt, illetve kapott adat vektora. A MISO kimenet alapállapota alacsony. Az adatkeret kezdetét a mester fél diktálja, az órajelet ő biztosítja. Az adatátvitel kezdetét az nCS jel lefutó éle, és az ezt követő felfutó, majd lefutó SCLK él jelzi. Az átviteli keret végezte az utolsó, D0 adatbit beolvasása (ami egy SCLK lefutó él) utáni nCS felfutó éllel történik. Túl sok, vagy túl kevés bit érkezése után a kapott adat eldobásra kerül.

4.3.2. Az tervezett SPI Slave modul működése

Az SPI Slave modul egy SPI Slave top nevű modulba ágyazódik. Ebben a modulban történik a belső periféria buszról a Wishbone buszra való áthidalás, amit a Wishbone Interface Master modul végez.

Az SPI Slave modulban az adatátvitelt egy állapotgép irányítja. Az „Idle” állapotból a gép



25. ábra: Az SPI Slave állapotgépét bemutató ábra.

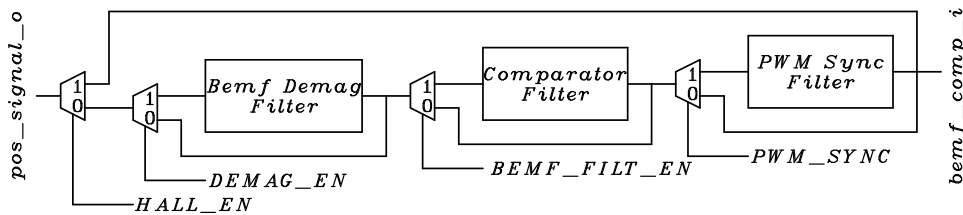
az adatkeret kezdetét jelző jel, a *start_flag* (*nCS* alacsony, majd *SCLK* fel- majd lefutó él) után az adatátviteli, „Data RX, TX” állapotba kerül. Itt az *SCLK* lefutó élein a kapott bitek leolvasásra kerülnek egy shift-regiszterben. A keret elején a shift-regiszter második eleme 1-re állítódik, az első elem az első beolvasott érték, a többi bit törlődik. A második elem megjelölése az utolsó címbit beolvasásának jelzésére szolgál. Ennek történetkor a kapott címre egy olvasási kérelmet küld a buszra az SPI Slave. A kérelmezett adat megérkezéséhez három órajel ciklus szükséges, így ez alapvetően korlátozza a maximális megengedett *SCLK* frekvenciát. A buszművelettel olvasott adat egy másik, TX shift-regiszterbe kerül. Innentől az adat bitek egyszerre kerülnek beolvasásra és kiküldésre a logika által. Az utolsó adat bit olvasása után, ha *rx_data[RW_BIT] = 0*, a beolvasott adat az adott címre írásra kerül a busz használatával. Ha az utolsó olvasott bit után *SCLK* lefutó élt kap az állapotgép, akkor írás művelet nem kerül sorra és az állapotgép alapállapotba áll. Ha a keret olvasása közben az *nCS* idő előtt a keret végét jelzi, akkor az addig olvasott adat eldobásra kerül és a gép alapállapotba áll.

4.4. Position Detector (PD)

A tervezett Position Detector, vagy Pozíció Meghatározó egység feladata a vezérelt motor kommutálásához szükséges pozíció jel feldolgozása. A Position Detector fogadja a MotVez-be bemenő back-EMF komparátor vagy Hall szenzor jeleket, ezeket a felhasználó által beállított feldolgozást elvégezve a kommutáció vezérlő egységnek (CCON) irányítja.

4.4.1. A Position Detector működésének leírása

A Hall érzékelő jelek fogadása esetén a Position Detector modul csupán továbbítja a jelet, azok szűrésére nincs szükség. Az érzékelő nélküli pozíció meghatározás esetén viszont a komparátorok által adott pozíciót számos okból előálló zaj zavarhatja. A zajos jelet a Position Detector annak okozati részeire bontja, azokat külön-külön hardveres alegységekkel távolítja el. Ezek az egységek a komparátor szűrő, a PWM szinkronizáló szűrő és a back-EMF demagnetizációs szűrő. Ezek a szűrők szükség szerint külön-külön be- és kikapcsolhatók a 26. ábrán látható

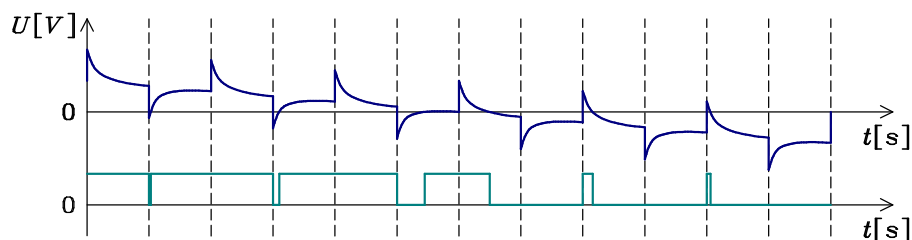


26. ábra: A szűrők logikai kapcsolata.

beállításbitek változtatásával, a szűrési sorrend viszont adott. A szűrés eredményeként kapott pozíció jelet a POS_PTRN bitmező olvasásával egy külső rendszer is felhasználhatja.

A PWM szinkronizáló szűrő

A tervezett PWM szinkronizáló szűrő a komparátor mintavételezését a PWM (Pulse Width Modulation - pulzusszélesség moduláció) éleihez ütemezi, a [2]-ben leírt viselkedés alapján. A MotVez a motor sebességének szabályozásához pulzusszélesség modulációt használ. A FET-ek be- és kikapcsolásakor feszültség tranziensek keletkeznek a motor fázisaiban. Ezek a tranziensek a szenzor nélküli hajtáskor vizsgált semleges fázison is megjelennek. A tranziensek a PWM be- vagy kikapcsolt állapotú periódusának vége felé lecsengnek, ha erre elég idő áll rendelkezésre, a 27. ábrához hasonló módon. Az adott PWM fázisban tehát mindig a fázis végét jelentő él (felfutó vagy lefutó) előtt közvetlenül kaphatjuk a legpontosabb komparálási eredményt. A 27. ábrán a motor egyik fázisának feszültsége látható, a PWM tranziensek tipikusan egy prellegő jelet eredményeznek a komparátor kimenetben. A zaj miatt többszöri nullpont átmenet érzékelés



27. ábra: A PWM zaj (felül) és az eredményezett komparátor jel (alul) illusztrációja.

történik, ami meghamisítja a pozíció becslést. A MotVez-ben elérhető a belsőleg generált, vagy betáplált PWM jel, ami a FET-ek PWM kapcsolását diktálja. A PWM szinkronizáló szűrő ezt a PWM jelet használja fel és ennek a jelnek az élén vizsgálja a bemeneti pozíció jelet. PWM_SYNC bitmező beállításával választható ki a vizsgálandó PWM él: felfutó, lefutó, mindkettő, egyik sem.³ A PWM él detektálása miatt az él után egy órajel ciklusnyi idővel történik a bemenet mintavétele, de ez olyan rövid idő, hogy a FET-ek ezalatt nem kezdenek kapcsolni. Abban az esetben, ha a PWM kitöltési tényezője 100% vagy 0%, a szűrő kikapcsol és átengedi a bemenő jelet⁴, hiszen nincs PWM él, amin mintavételezni lehetne.⁵

³Az "egyk sem" lehetőség a PWM szűrő kikapcsolt állapotát jelöli.

⁴Ha külsőleg generált PWM-et használunk, akkor ez nem történik meg, manuálisan kell kezelni a kikapcsolást ilyenkor.

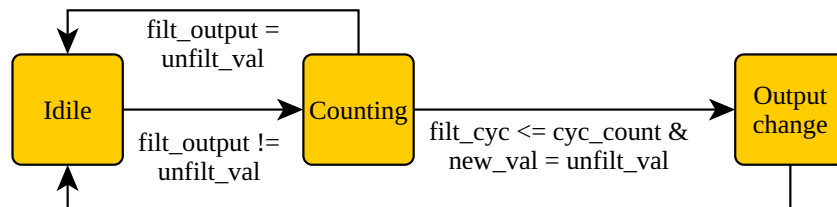
⁵Futás közben a kitöltési tényezőtől függően érdemes a kiválasztott élt váltani, hisz például, ha nagy a kitöltési tényező, akkor előfordulhat, hogy a PWM bekapcsolt periódus végén még mindig nem csengett le a tranziens.

Komparátor szűrő

A komparátor szűrő az általános zajok által okozott hamis komparátor váltások, prellégések kiszűrését szolgálja, a [27]-ben leírtakhoz hasonló elven. ⁶ A szűrő a BEMF_FILT_EN beállítás bittel kapcsolható be. A komparátor szűrő által eltávolított impulzus időtartama a BEMF_FILT_CYCLES bitmezővel állítható be, órajel ciklusok számaként. A szűrő által használt órajel a MotVez rendszer órajele egy kötött előosztóval lassítva. Így a maximum kiszűrt impulzus szélesség:

$$t_{\text{szűrő}} = BEMF_FILT_CYCLES \frac{256}{f_{MotVez_clk}} \quad (4.2)$$

A szűrő ciklusszámának beállításakor figyelembe kell venni, hogy a szűrt jel a bemenethez képest $t_{\text{szűrő}}$ időtartammal késik. A szűrőt egy egyszerű állapotgép vezérli. Alap állapotból bemenő jel változás esetén a „Counting” állapotba kerül, ahol megkezdődik a számlálás. Ha a bemeneti

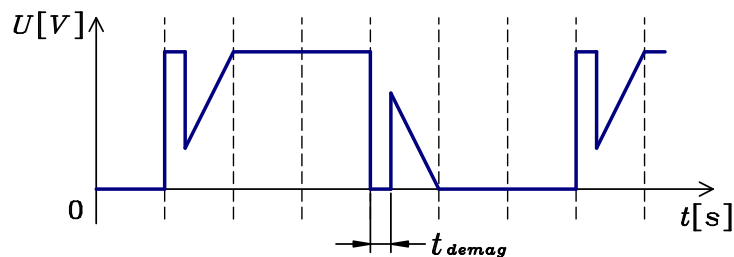


28. ábra: A komparátor szűrő állapotgép ábrája.

érték visszavált az eredeti értékére az időzítő letelte előtt, a gép visszakerül „Idle” állapotba. Máskülönben amikor a számláló eléri a megadott ciklusszámot, a következő „Output change” állapotban a szűrő kimenetén megjelenik az új érték, majd újra alapállapotba áll az állapotgép.

Back-EMF demag pulzus szűrő

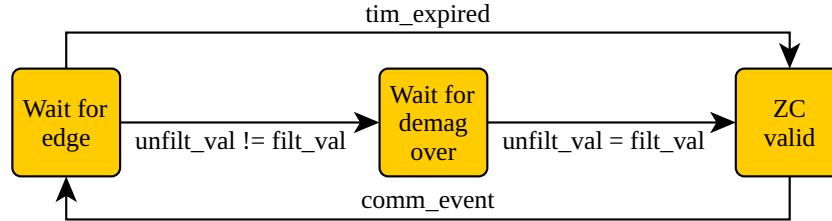
A MotVez által használt kommutáció, a hatlépéses „blokk” kommutáció minden esetben egyszerre csak két tekercset energizál. A harmadik, „lebegő” tekercsen látható a motor által generált feszültség, amit a szenzor nélküli pozíció meghatározás használ. Egy fázis bekapcsolt állapotból lebegő állapotba kerülésekor benne még áram folyik, aminek lecsengéséhez idő szükséges. Ezt nevezzük demagnetizációs időnek. Ezalatt az idő alatt a háromfázisú H-hídban az inaktív fázis



29. ábra: A back-EMF demagnetizációs pulzust szemléltető ábra.

⁶A szűrő adott esetben akár használható a PWM és a demag tranziensek szűrésére is, de azok ilyenfajta szűrése jelentős késést okozhat a pozíció jelben.

egyik FET-jének diódáján fog ez az áram folyni. Tehát az áram irányától függően a lebegő fázison mért feszültség $U_{\text{táp}} + U_{\text{dióda}}$ vagy $-U_{\text{dióda}}$ lesz erre az időre, ezért a back-EMF nem látható ilyenkor, a komparátor pedig hamis nullátmenetet jelez. Ennek a hibás komparátor váltásnak kiszűrését a 30. ábrán látható állapotgép végzi. A tervezett szűrő a [21]-ben bemutatott, időzítő alapú és a [27] által alkalmazott, pusztán a bemenet változását figyelembe vevő módszert kombinálja, ezáltal mindkét módszernél megbízhatóbb üzemet biztosít. A tervezett állapotgép [16]-ban leírtakhoz hasonló viselkedést ér el.⁷ Az állapotgép mindaddig a „ZC Valid” (Nullátmenet



30. ábra: A back-EMF demag pulzus szűrő állapotgépének ábrája.

Érvényes) állapotban marad, amíg a külső Commutation Controller egységtől jelet nem kap kommutáció történtéről (comm_event). Ezután a „Wait for edge” állapotban vár a kommutációs tranziens megtörténte, ezzel együtt figyeli a kommutációs időzítőre kötött tim_expired jelet, ami egy előre beállított, a kommutációtól számított időtartamot jelöl. Ha az időzítő lejártá előtt megjelenik a tranziens, a „Wait for demag over” állapot következik, egyébként a nullátmenetek újra érvényesnek számítanak a következő kommutációig. Az időzítő szükségességét az adja, hogy a demag pulzus nem jelenik meg minden esetben, például alacsony motor áramnál, illetve olykor csak a back-EMF fel- vagy csak lefutó szakaszán látható. A szűrő a DEMAG_EN bittel kapcsolható ki és be. A DEMAG_PERIOD bitmező állítja be, hogy a kommutációtól számítva mennyi ideig figyelje a szűrő a demag tranziens megjelenését.

$$t_{\text{demag}} = \frac{t_{\text{komm}}}{2^{\text{DEMAG_PERIOD}}} \quad (4.3)$$

Ügyelni kell a DEMAG_PERIOD és a komparátor szűrő BEMF_FILT_CYCLES kiválasztásánál, hogy a motor legnagyobb futtatni kívánt sebességénél is kisebb legyen a komparátor szűrő idő, mint a demag időzítő ideje, mert ellenkező esetben a demag pulzus a tim_expired jel után jelenik meg a szűrő bemenő jelén és átjut a szűrőn.

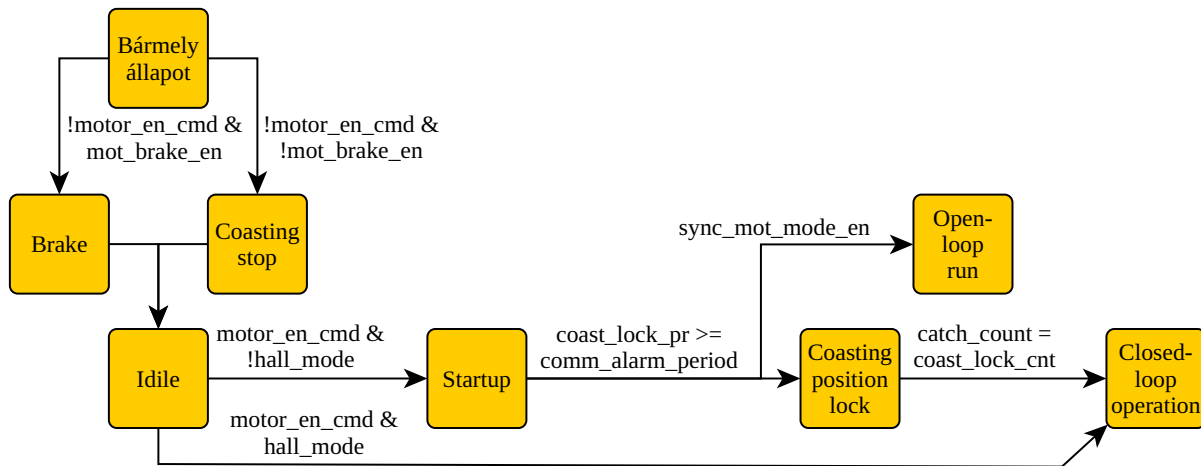
4.5. Commutation Controller (CCON)

A tervezett Commutation Controller vagy „Kommutáció Vezérlő” modul felelős a kefe nélküli motor kommutációjának időzítéséért, a motor futtatásáért. Alapvetően ez az egység egy pozíció bemenetet dolgoz fel amit a Position Detector-tól kap és egy másik pozíció jelet továbbít az Output Controller egység felé. A Commutation Controller modulban található állapotgép vezérli

⁷A hasonlóság mértéke a [16]-ben leírtak homályos megfogalmazása miatt nem egyértelmű.

a motor indítását és leállítását, fékezését a különféle üzemmódokban, melyek a beállítás bitekkel választhatók.

4.5.1. A Commutation Controller működésének leírása



31. ábra: A kommutációs vezérlő állapotgépének ábrája.

A motor indítása vagy engedélyezése kétféleképpen történhet. Alap esetben a motor csak akkor indul, ha az engedélyező bemeneti láb és a MOT_EN bit magas. Azonban az EN_CTRL_MOT_EN bit beállításával elegendő az engedélyező láb magas állapota az indításhoz.

A motor engedélyezése után az állapotgép a „Startup”, vagy indítási állapotba kerül, ha a Hall szenzoros üzem nincs engedélyezve a HALL_EN bittel. Máskülönben a motor meghajtása rögtön zárt hurkú kommutációval történhet, így az állapotgép a „Closed-loop operation” állapotba ugrik.

Szenzor nélküli üzemmódban az álló motort először nyílt hurkú kommutáció használatával kell egy adott, minimális sebességre felpörgetni. Ez azt jelenti, hogy a motor nem indítható jelentős terhelő nyomaték mellett, mert a forgórész és az állórész mágneses mezője kieshet a szinkronból. A Commutation Controller egy S-alakú sebességgörbével indítja a motort, így az nagyobb eséllyel marad szinkronban a felpörgetés ideje alatt, mint lineáris görbe használatával. A felpörgetés paraméterei regiszterbeállításokkal testreszabhatók. Az OL_RAMP_START és az OL_RAMP_FINAL regiszterek a felpörgetési görbe kezdő és végső kommutációs periódusát adják meg.⁸ Az OL_RAMP_CONST regiszter a sebesség görbe meredekségét diktálja. Minél kisebb ez az érték, annál meredekebb a görbe.⁹ A felpörgetés vége akkor következik, amikor a kommutációs periódus eléri, vagy kevesebb lesz, mint a COAST_LOCK_PR regiszterben

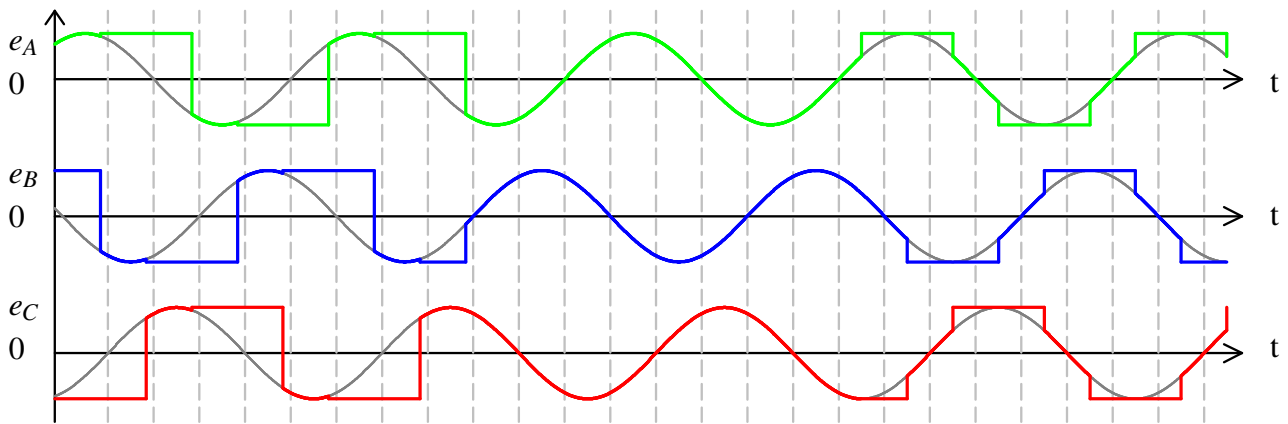
⁸Mindig igaznak kell lennie, hogy $OL_RAMP_START > OL_RAMP_FINAL$, egyébként a felpörgetés meghiúsul.

⁹Praktikus szempontból az $OL_RAMP_CONST < 3$ értékek igen gyors sebesség változást adnak, ezek általában nem biztosítanak megbízható felpörgetést. Általánosan minél nagyobb a forgórész tehetetlenségi nyomatéka, annál nagyobb értékre kell az OL_RAMP_CONST-ot választani.

megadott érték.¹⁰ A használt egyenlet a következő [15]:

$$n_{komm}[k+1] = n_{komm}[k] - \frac{n_{komm}[k] - OL_RAMP_FINAL}{2OL_RAMP_CONST}, \quad (4.4)$$

ahol $n_{komm}[k]$ a jelenlegi kommutációs periódus száma. A felpörgetés végeztével szükséges a forgórész és állórész mágneses tereinek pontos szinkronizálása a zárt hurkú kommutáció működtetéséhez. Ahhoz, hogy a lebegő back-EMF feszültsége láthatóvá váljon, a kommutációs szektornak és a motor által generált feszültségnek szinkronban kell lennie, maximum (elméleti) $\pm 15^\circ$ eltéréssel. A problémát az adja, hogy diktált kommutáció közben a szinkron szögkülönbséget a táp feszültség, a terhelő nyomaték és a diktált sebesség is befolyásolja, így a felpörgetés után nem lehetünk biztosak, hogy a pozíció jel valóban a helyes nullpontmetszést követi. Ennek



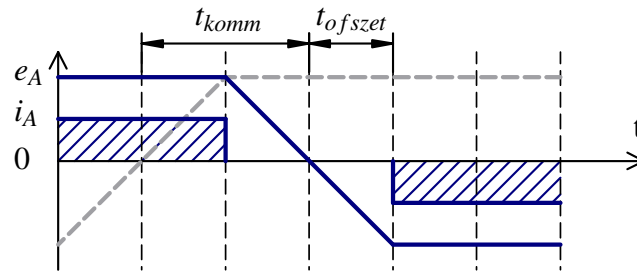
32. ábra: A forgórész szinkronizálás folyamata.

kikerüléséhez a felpörgetés után szükséges a motor pozíciójának pontos meghatározása. Ehhez egy rövid időre mindhárom fázis kikapcsolt, lebegő állapotba kerül, így zavar nélkül láthatóvá válik a generált három fázisú feszültség, tehát az érzékelt forgórész pozíció biztosan helyes [6]. Adott számú nullátmenet után véget ér a szinkronizálás és megkezdődhet a zárt-hurkú kommutáció alapú üzem. A várt nullátmenetek számát a COAST_LOCK_CNT mező állítja be.

Az általános szenzor nélküli üzem során a pozíció változás indítja el a kommutációs időzítőt. Két pozíció változás közti eltelt időt megfigyelve kapjuk a szükséges kommutáció késleltetésének periódusát [32].

Választható módon engedélyezhetjük a szinkronmotor módot a SYNC_MOT_MODE bittel. Ebben a módban felpörgetés után a motor nem lép át zárt hurkú kommutációba, hanem továbbra is diktált kommutációban fut. Ebben a módban a kommutáció időzítése kötött, tehát a motor érzékeny lesz a terhelő nyomaték változására.

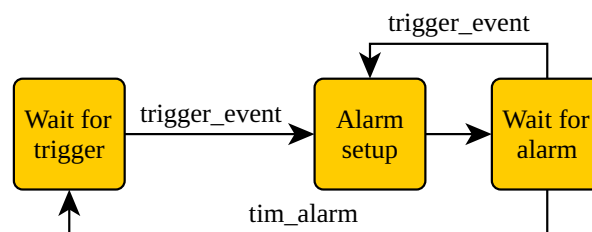
¹⁰Ha $COAST_LOCK_PR < OL_RAMP_FINAL$, a motor felpörgetési állapotban ragad. Az OL_RAMP_FINAL értéket úgy kell megválasztani, hogy az a motor tervezett üzemi sebessége legyen, míg a $COAST_LOCK_PR$ a szenzor nélküli kommutációhoz szükséges legkisebb sebesség.



33. ábra: A kommutáció időzítése a nullpontmetszés alapján.

Commutation Timer egység

A kommutációs időzítő vagy "Commutation Timer" egység felelős a kommutáció időzítésének és ütemezésének elvégzéséért, valamint a demagnetizációs szűrő időzítésének működtetéséért. A modult központilag egy állapotgép vezérli. Pozíció változás érzékelésekor a Commutation



34. ábra: A kommutációs időzítő állapotgép ábrája.

Controller állapotgépe a *trigger_event* jelet küldi az erre várakozó Commutation Timer állapotgépeznek. Ennek hatására az megindítja az időzítőt az adott periódusra beállítva a riasztást az "Alarm setup" állapotban. Az időzítő lejártával (*tim_alarm*) a modul jelez a Commutation Controllernek, hogy megtörténhet a kommutáció. A riasztásra való várakozás közben a demag szűrő kikapcsolásra kerül a számított periódus elérését követően.

A kommutáció időzítéséhez használt órajel a MotVez órajelének osztásával áll elő. Az osztószám állítható, a COMM_TIM_PSC bit segítségével. ¹¹

4.6. PWM Controller (PWMC)

A tervezett PWM Vezérlő, vagy "PWM Controller" egység felelős a motor hajtó feszültségének változtatásához használt pulzusszélesség moduláció jel előállításáért és kezeléséért.

4.6.1. A PWM Controller működésének leírása

A MotVez rendelkezik belső PWM (Pulse Width Modulation) generátorral a H-híd feszültség PWM-el való irányításához. Ez a generált PWM 9 bites felbontású kitöltési tényezővel állítható, a PWM_DC bitmezővel. A PWM frekvenciája a generátor órajelének előosztásával állítható,

¹¹Az osztás azért szükséges, mert alacsony előosztónál alacsony motor sebesség a szektor időt mérő regiszter túlfolyását eredményezheti, nagy előosztó miatt viszont nagy sebességnél a szektor idő pontossága csökken.

a PWM_TIM_PSC segítségével. A belső PWM generálás kikapcsolható, ekkor az adott külső lábon kapott PWM jel kerül felhasználásra. Ez az üzemmód a PWM_INT_GEN bitmezővel állítható. A generált vagy kapott PWM jel az Output Controller és a Position Detector modulnak kerül továbbításra.

4.7. Output Controller (OC)

A tervezett Kimenet Vezérlő, vagy "Output Controller" kezeli a MOSFET gate lábakat vezérlő kimeneteket a kapott kimeneti mintázat, PWM és forgási irány alapján.

4.7.1. Az Output Controller működésének leírása

Az Output Controller egység a CommutationController-től kap egy 3 bites kimeneti mintázatot, amit felhasználva egy kombinációs logika kiválasztja a bekapcsolandó H-híd tranzisztorokat a forgási iránytól függően. Ezután a felső FET-ek bekapcsoló jelére ÉS kapcsolattal csatlakozik az Output Controllerbe betáplált PWM jel.

A kapott mintázat alapján a FET bekapcsoló jelek a következő táblázat szerint alakulnak:

1. táblázat: A kommutációs minta táblázata.

Szektor	DIR = 1						DIR = 0					
	AH	BH	CH	AL	BL	CL	AH	BH	CH	AL	BL	CL
0 1 1	1	0	0	0	1	0	0	1	0	1	0	0
0 1 0	0	0	1	0	1	0	0	1	0	0	0	1
1 1 0	0	0	1	1	0	0	1	0	0	0	0	1
1 0 0	0	1	0	1	0	0	1	0	0	0	1	0
1 0 1	0	1	0	0	0	1	0	0	1	0	1	0
0 0 1	1	0	0	0	0	1	0	0	1	1	0	0
0 0 0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 1	1	1	1	0	0	0	1	1	1	0	0	0

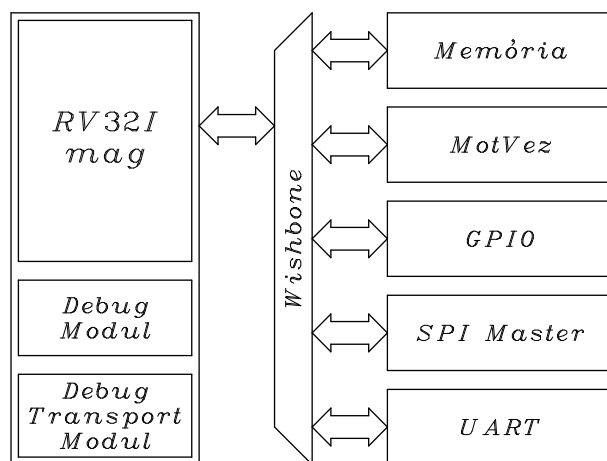
Választható módon a felhasználó felülírhatja a MotVez által válsztott FET gate jeleket a FET_ATOMIC_SET bit segítségével. Ekkor a FET-ek a FET_STATES bitmezővel kapcsolhatók ki és be. A bekapcsolt FET-eken ebben a módban is megjelenik a betáplált PWM jel. Ez a mód hasznos lehet egyedi, megvalósítás függő problémák megoldására.

5.

LovagSoC

5.1. Rendszerkép

A tervezett LovagSoC (Lovag System on Chip) processzora egy 32 bites, szabványos RISC-V utasításkészletű mag, amely az általános felhasználási igényeket látja el. A processzor utasításkészlete az alap RV32I - integer utasításkészlet. Ez azt jelenti, hogy a processzor szükség szerint továbbfejleszthető további kiegészítő utasítások megvalósításával. A mag megfigyeléséhez, debuggoláshoz, valamint a program memóriába való adat külső feltöltéséhez egy, a maghoz hasonlóan szabványos kialakítású debug egység, a "Debug Module" használatos. A Debug Modul



35. ábra: A LovagSoC rendszer diagramja.

és a külső debugger hardver közötti összeköttetést a "Debug Transport Modul" szolgáltatja. Ez az egység a külső debugger által használt kommunikációs protokollt alakítja át a Debug Modul belső buszának hozzáféréséhez. A RISC-V processzor egy belső buszrendszer használatával kezeli a memóriát és az egyes perifériákat. Ez a szabványos Wishbone buszrendszert alkalmazza.

A LovagSoC egy Neumann architektúrájú processzort valósít meg. Ennek megfelelően a programmemória (ROM) és az adatmemória (RAM) egy címtérben helyezkedik el. ¹ A busz

¹Jelen esetben az FPGA-n való megvalósítás miatt a ROM és a RAM fizikailag is egy egységet alkot. Köztük a különbséget csupán a címtérben való szoftveres megkülönböztetés adja (linker script).

műveletek egy kimeneten (porton) keresztül történnek, a perifériák is a memória címtérben vannak. A buszra fűzött egységek között a memória térkép alapján a címdekóder választ.

A rendszer kifejezetten egy motor meghajtása köré lett tervezve. A LovagSoC egy erre a feladatra fejlesztett motorvezérlő perifériával rendelkezik. A MotVez (Motor Vezérlő) egység kefe nélküli (BLDC - Brushless DC) motorok egyszerű meghajtását teszi lehetővé. A MotVez képes a BLDC motor Hall érzékelő nélküli működtetésére is. ²

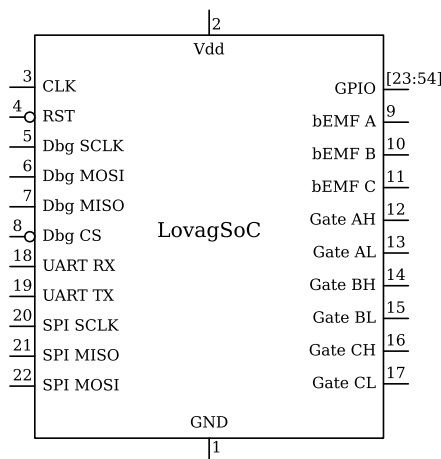
A GPIO egység 32 darab választható irányú ki-/bemenetet biztosít általános célú felhasználásra.

A LovagSoC egy mester konfigurációjú SPI egységgel is rendelkezik. Ez viszonylag nagy sebességű kommunikációt biztosít külső chipekkel. Az SPI Master full-duplex kivitelű, tehát egyszerre biztosít kétirányú adatátvitelt.

Az UART periféria biztosít alternatívát az SPI mellett egy külső rendszerrel való kommunikációhoz. A full-duplex kivitelű UART egymástól függetlenül képes adat küldésére és fogadására.

5.1.1. A LovagSoC lábkiosztása

A LovagSoC a jelenlegi, v0.12. változatában 52 lábbal rendelkezik. Mivel a megvalósítás FPGA-n, ezért a *GND* és az *V_{dd}* lábak nem számítanak ide.

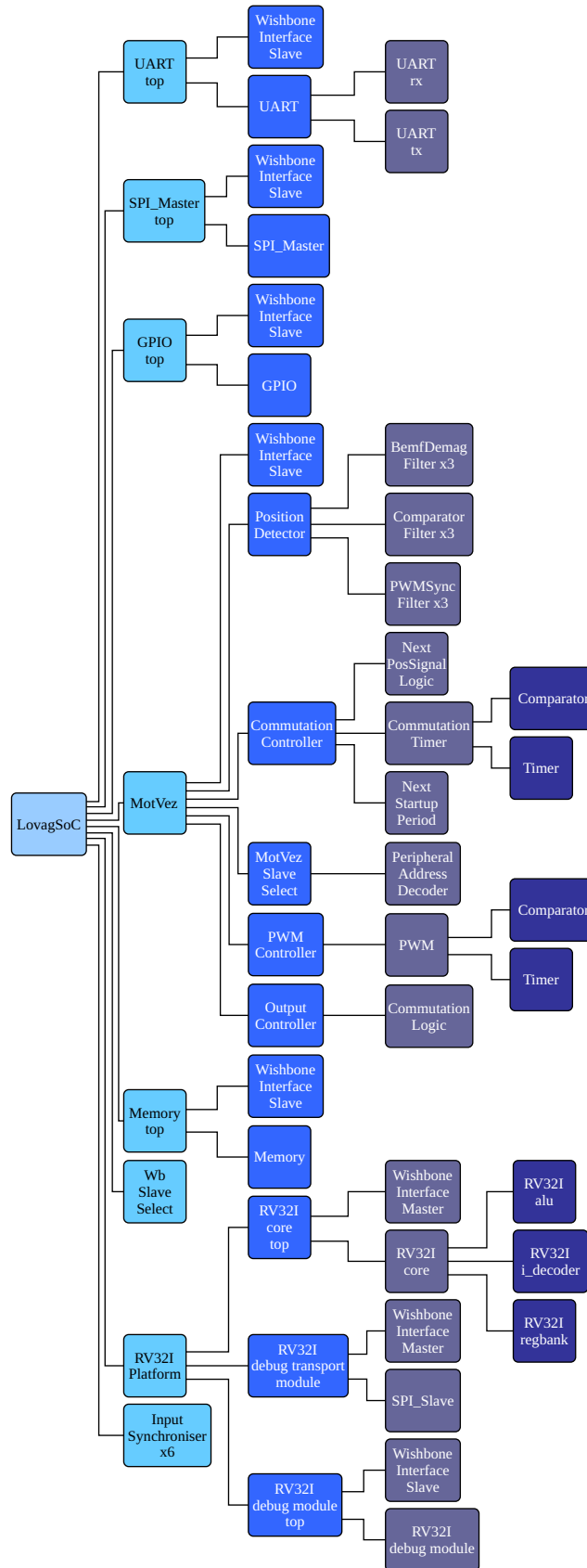


36. ábra: A LovagSoC áramköri szimbóluma.

5.2. RV32I processzor

A LovagSoC mikrovezérlő központi egysége a CPU (Central Processing Unit). Ez az egység felelős a felhasználó által szolgáltatott utasításoknak az adott sorrendben való végrehajtásáért.

²Valójában ez a fő rendeltetése a MotVez-nek, de nem csak szenzor nélküli BLDC-ket képes meghajtani.



37. ábra: A LovagSoC mikrovezérlő logika modul hierarchiájának ábrája.

Az RV32I processzort a RISC-V International által közzétett RISC-V ISA (instruction set archi-

tecture) utasításkészlet szabvány alapján terveztem. A felhasznált dokumentum a "The RISC-V Instruction Set Manual Volume I: Unprivileged ISA", változata a 20191213 [30]. A processzor mag a LovagSoC-n belül az RV32I platform egységben foglal helyet. Itt kap kiegészítést szabványos Wishbone buszrendszerre való csatlakozáshoz, valamint a debug feladatok ellátásához, a Debug Modul egység segítségével.

5.2.1. Az RV32I mikroarchitektúra

A RISC-V szabvány többféle utasításkészletet definiál, az alap készletek kötelezően használandók, ezekre épülnek a kiegészítő utasítások. A megvalósított processzor mag a 32 bites alap utasításkészletet alkalmazza, ez az RV32I, vagy RISC-V 32 bit integer.³

Az alap utasításkészlet regiszterkészlete 32 darab 32 bit széles, általános felhasználású regisztert tartalmaz (GPR - general purpose register), ahol a 0. indexű regiszter (x0) mindig 0 értékű.

A RISC-V architektúra 4 bájtos⁴ határra igazított memória hozzáféréseket támogat⁵, de ezeken a határokon belül 8 bites granularitással címezhető meg adat. A határok átlépése jelen megvalósítás esetén hibás művelet végrehajtást eredményez. A megvalósított RV32I mag nem

0x4				0x0			
32 bit				32 bit			
16 bit		16 bit		16 bit		16 bit	
8 bit	8 bit	8 bit	8 bit	8 bit	8 bit	8 bit	8 bit

38. ábra: Az RV32I memória címzés igazítási határai.

rendelkezik hardver által kezelt szorzás és osztás művelettel, ezeket a fordítónak vagy assembly programozónak kell kezelnie.

A tervezett RISC-V mag Neumann-elv alapján készült. Ennek megfelelően az utasításmemória (programmemória) és az adatmemória hozzáférése egy busz porton keresztül történik, a két egység egy címtérben foglal helyet [12]. A Neumann architektúra alkalmazásának oka az, hogy ezt használva csak egy busz portot kell alkalmazni, ezáltal a rendszer komplexitása csökken, így a fejlesztési idő és a logika mérete is.

Az RV32I mag rendelkezik kezdetleges debug támogatással, a processzor futása felfüggeszthető, újraállítható, a memória és a regiszterkészlet felfüggesztett állapotban vizsgálható és módosítható.

A tervezett processzor mikroarchitektúra szempontjából a multiciklusú processzorok közé tartozik. Egy utasítás végrehajtásához több órajel ciklus idő szükséges.

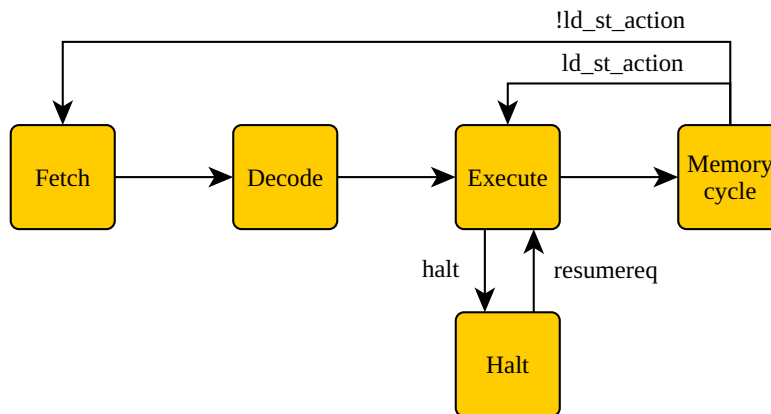
³A szabvány dokumentumban előforduló XLEN a 32 értéket veszi, az ILEN szintén 32 bit.

⁴A "C" utasítás kiegészítéssel 2 bájtos is lehet, ez jelenleg nem használt.

⁵A szabványban az IALIGN 32 bites értéket vesz.

Központi logika

A mag központi logikai egysége felelős az utasítások beolvasásának és végrehajtásának ütemezéséért, a vezérlő jelek generálásáért, a regiszterkészlet és a memória műveletek kezdeményezéséért. A központi logika az 39. ábrán látható állapotgépét tartalmazza. Ez az állapotgép minden



39. ábra: Az RV32I mag állapotgépének ábrája.

felfutó órajel élen lép a következő állapotba.

Egy utasítás végrehajtásának leírása A processzor alaphelyzetben a "Fetch" állapotból indul. Ebben az állapotban az olvasás buszművelet eredményeként kapott adat az elvégzendő utasítást tároló regiszterbe kerül beolvasásra.

A "Decode" állapotban a beolvasott utasítás dekódolása történik. Ez egy különálló egység, az utasítás dekóder kombinációs logika feladata. Az utasítás dekóder felismeri az utasítás fajtáját, hogy milyen műveletet ír elő az utasítás, és, hogy ennek a műveletnek mik a tényezői.

Az "Execute" állapotban végrehajtásra kerül az utasítás elemzés eredményeként kapott művelet. Ez lehet ALU művelet, regiszter írás közvetlen értékkel, memória írás vagy olvasás adott címmel, stb. Ha nincs végrehajtandó memória írás vagy olvasás művelet, ebben az állapotban kezdeményezésre kerül a következő utasítás beolvasása a programmemóriából.

Az alkalmazott buszprotokoll egy adatátvitel elvégzéséhez két órajel ciklust igényel. A következő, "Memory cycle" állapotban a buszművelet elvégzéséhez szükséges ciklust biztosítja, tehát ez egy várakozási állapot. Ha a jelenlegi utasítás memória hozzáférést igényel ($ld_st_action = 1$), akkor az Execute állapot után először ez a buszművelet hajtódik végre, majd az állapotgép visszalép az Execute-ba, ahol eztán már a következő utasítás olvasását kezdeményezi az állapot ($ld_st_action = 0$).

A buszművelet végeztével ismét a Fetch állapot következik.

Indítás utáni inicializálás A processzor indítása (reset, táp bekapcsolása) után az utasítás regiszterben nincs adat. Ezért az első utasítás végrehajtás körben üresen megy végig az állapotgép, a program végrehajtása tehát 5 ciklus késéssel kezdődik ilyenkor.

Működés a felfüggesztett állapotban A tervezett mag debug tevékenység végzését is támogatja. Ennek vezérlését a külső Debug Modul végzi, a debugger utasításainak megfelelően. A debug elvégzése a "Halt" állapotban történik. Futás közben ide a Debug Modul kérelmével kerül az állapotgép. Felfüggesztési kérelem esetén a jelenleg beolvasott utasítás dekódolása megtörténik, az állapotgép az "Execute" állapotba lép, de nem hajtódik végre az utasítás. Innen kerül a "Halt" állapotba az állapotgép.

A felfüggesztett állapotban a Debug Modul hozzáfér a regiszterkészlethez és a processzor memória portjához.

Mivel reszet után az első Execute állapot nem kezel utasítást, az első utasítás beolvasása és végrehajtása előtt be lehet lépni a debug módba.

Regiszterkészlet

A processzor regiszterkészlete 32 darab 32 bites elemből áll. Ezek külön egységben foglalnak helyet, a processzor központi logikája egy szinkronos írás porton és két aszinkron olvasás porton éri el a regisztereket. A két olvasás port a regiszter-regiszter utasítások operandusainak beolvasásához szükséges. A 0. indexű regiszter a szabvány utasításának megfelelően 0-hoz kötött értékű.

Aritmetikai-logikai egység

Az aritmetikai-logikai egység (ALU - Arithmetic-Logic Unit) felelős a számítások, logikai műveletek elvégzéséért. Ez akkor megy végbe, ha az utasítás dekóder az adott művelet jelét generálja a "Decode" állapotban. Az elkészült eredmény a következő, "Execute" állapotban kerül feldolgozásra.

5.3. A Wishbone buszrendszer

Fontos tudnivaló: A jelen v0.12. (fejlesztői) változatban a tervezett busz kommunikációs logika nem garantált, hogy 100%-ban megfelel a szabvány előírásainak.

A LovagSoC az RV32I processzor és a perifériák közti kommunikáció folytatásához busz rendszert használ. Ez a buszrendszer a Wishbone B4 változatú szabvány előírásai alapján valósult meg.

5.3.1. A megvalósított Wishbone rendszer tulajdonságai

Szabvány verzió: A buszrendszer tervezésekor figyelembe vett Wishbone szabvány verziója a B4. A használt dokumentum a "WISHBONE System on Chip (SoC) Interconnection Architecture for Portable IP Cores" volt az OpenCores jóvoltából [31].

Kommunikációs szereposztás: A LovagSoC-ben a Wishbone buszport az RV32I mag és a DTM esetében mester, minden más Wishbone periféria esetén szolga szerepet tölt be.

Busz jel megnevezések: Az alábbiakban felsorolásra kerülnek a LovagSoC perifériái által használt Wishbone jelek megnevezései.

clk: Clock. A busz művelet által használt órajel. Jelen esetben megegyezik a perifériák órajelével.

adr: Address. A busz művelet által használt cím. A DTM 7 bites, az RV32I 32 bites szélességű vonalat használ.

dat: Data. A művelet során átvitt adat. Két külön vonal a két adat irány számára. 32 bit széles a LovagSoC-ben.

we: Write enable. Az írás műveletet engedélyező jel.

sel: Select. Kiválasztja, hogy a *dat* vonal melyik részén található az írott/olvasott adat.

stb: Strobe. Egy aktív busz ciklus folyamatát jelzi.

ack: Acknowledge. A szolga periféria visszajelzése a busz művelet megtörténtéről.

cyc: Cycle. Egy aktív busz ciklus folyamatát jelzi.

Port méret: Az egyszerre átküldhető adatmennyiség maximum a LovagSoC által megvalósított Wishbone portokon 32 bit.

Port granularitás: Az alkalmazott Wishbone portok granularitása mindenütt 8 bites.

Bájtsorrend: A használt bájtsorrend minden porton kis-endián rendezésű.

Az adatátvitel szekvenciája: Nem definiált.

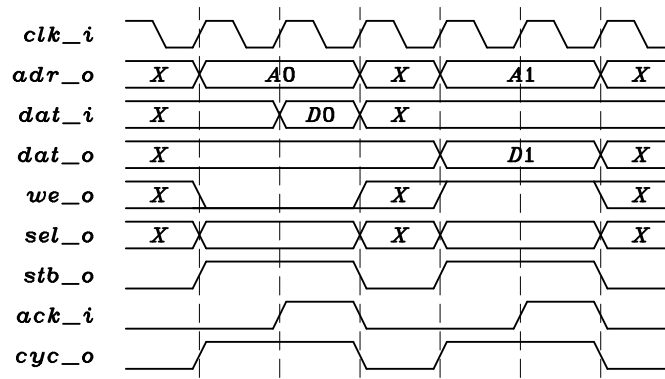
5.3.2. A műveletek felépítése

A LovagSoC-ben a Wishbone szolga busz felületek képesek az alap (standard) és a futószalagos (pipelined) busz protokollok kezelését. A szolgák minden esetben az órajel élével szinkronban válaszolnak a mester által kezdeményezett műveletre. Az alkalmazott buszprotokollok a klasszikus "classic" és egyedüli "single" átvitelt használják. Az alábbi protokoll leírások ábráin az *i* és *o* végződés a mester periféria szemszögéből a jel irányát mutatja.

Alap írás és olvasás

Az alap (a dokumentációban "standard" megnevezésű) protokoll egy műveletet három órajel ciklus alatt végez el.

Olvasás művelet Az első felfutó *clk* élen a mester periféria közlésezi a művelet címét, a *we* vonalat 0-ra húzza, a *sel* vektorral megadja, hogy hova várja az adatot és jelzi az átvitel érvényességét a *cyc* és *stb* jelekkel. A második felfutó élen a szolga periféria válaszol az *ack* felhúzásával és az adott cím által meghatározott adat *dat_i*-re tevésével. A harmadik élen a mester véget vet az átviteli ciklusnak, tehát *cyc* és *stb* 0, a szolga válaszul negálja az *ack*-ot.



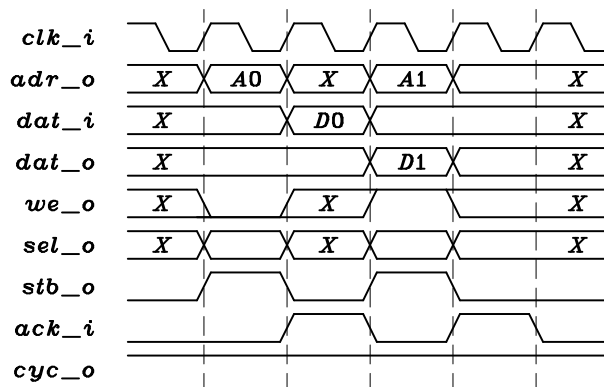
40. ábra: A Wishbone alap olvasás (bal) és írás (jobb) idődiagramja a mester szemszögéből.

Írás művelet Az első felfutó *clk* élen a mester periféria közlésezi a művelet címét és az adatot, a *we* vonalat 1-ra húzza, a *sel* vektorral megadja, hogy hol lesz adat a *dat_o*-n és jelzi az átvitel érvényességét a *cyc* és *stb* jelekkel. A második felfutó élen a szolga periféria válaszol az *ack* felhúzásával és a kapott adatot a cím és a *sel* vektor segítségével feldolgozza. A harmadik élen a mester véget vet az átviteli ciklusnak, tehát *cyc* és *stb* 0, a szolga válaszul negálja az *ack*-ot.

Futószalagos írás és olvasás

A futószalagos (pipelined) protokoll egymás után következő átvitelekhez csak 2 ciklust igényel. A LovagSoC-ben a *cyc* ebben a módban mindig 1-re van kötve.

Olvasás művelet Az első felfutó *clk* élen a mester periféria közlésezi a művelet címét, a *we* vonalat 0-ra húzza, a *sel* vektorral megadja, hogy hova várja az adatot és jelzi az átvitel érvényességét az *stb* jellel. Ezzel egyidőben a szolga az *ack* jelet 0-ra húzza, ha nem ott volt. A második felfutó élen a szolga periféria válaszol az *ack* felhúzásával és az adott cím által meghatározott adat *dat_i*-re tevésével. Ezzel egyidőben az *stb* negálódik, jelezve a művelet végét.



41. ábra: A Wishbone futószalagos olvasás (bal) és írás (jobb) idődiagramja a mester szemszögéből.

Írás művelet Az első felfutó *clk* élen a mester periféria közzéteszi a művelet címét és az adatot, a *we* vonalat 1-re húzza, a *sel* vektorral megadja, hogy hol lesz adat a *dat_o*-n és jelzi az átvitel érvényességét az *stb* jellel. Ezzel egyidőben a szolga az *ack* jelet 0-ra húzza, ha nem ott volt. A második felfutó élen a szolga periféria válaszol az *ack* felhúzásával és a kapott adatot a cím és a *sel* vektor segítségével feldolgozza. Ezzel egyidőben az *stb* negálódik, jelezve a művelet végét.

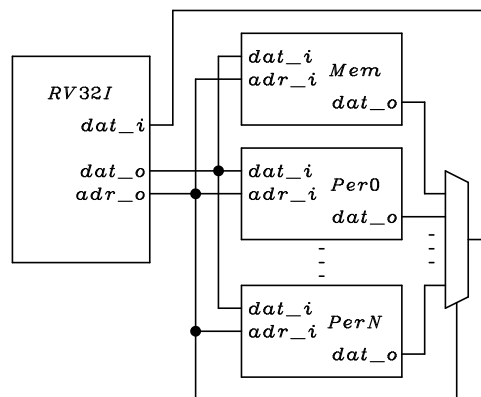
5.4. Memória

A LovagSoC memóriája az FPGA-n való implementáció eredményeként "block RAM"-ként valósul meg. A v0.12. változatú LovagSoC-ben a programmemória mérete 6kB, az adatmemória mérete 1kB. Ez egy fejlesztői változatban elegendő nagy alapszintű feladatok megoldásához, a fejlesztett perifériák teszteléséhez.

A ROM és a RAM jelen esetben egy fizikai egységet alkot, köztük a különbséget szoftver biztosítja. A LovagSoC-ben a ROM, a RAM és a perifériák regiszterei egy címtérben vannak, őket azonos, memória írási és olvasási processzor utasításokkal lehet elérni.

5.4.1. A memória és perifériák bekötése buszrendszerrel

Az alábbi diagram szemlélteti a memória és a perifériák bekötését a CPU-hoz. A perifériák



42. ábra: A memória bekötésének diagramja.

felé tartó buszjelek minden perifériához egyaránt megérkeznek. A perifériák belső logikája felelős a cím dekódolásához, hogy neki szól-e a kezdeményezett buszművelet. Ezzel szemben a processzor felé tartó jelek közül egy multiplexer választja ki a megfelelő perifériát az adott cím segítségével.

5.4.2. Memória térkép

A 2. táblázat tartalmazza a memória egységek és perifériák elérési címeinek térképét.

2. táblázat: A LovagSoC memória térképe.

Címhatárok	Méret	Egység
0x2096 - 0xFFFF FFFF FFFF FFFF	-	Használatlan
0x2090 - 0x2095	6B	UART
0x2088 - 0x208F	8B	SPI Master
0x2080 - 0x2087	8B	GPIO
0x2014 - 0x207F	108B	Használatlan
0x2002 - 0x2013	18B	MotVez
0x1C00 - 0x2001	1026B	Használatlan
0x1800 - 0x1BFF	1kB	Adatmemória
0x0000 - 0x17FF	6kB	Programmemória

5.5. General Purpose Input Output (GPIO)

A tervezett GPIO periféria irányítja az általános célú ki- és bemeneteket. A 32 lábra osztható GPIO jelek egy portot alkotnak, az egyes lábak iránya állítható.

5.5.1. A tervezett GPIO működésének leírása

A GPIO periféria 32 darab választható irányú ki-/bemenetet valósít meg. A kimenetek minden esetben push-pull meghajtásúak. Az egyes lábak irányát a DIR regiszter állítja be. A STATE regiszter meghatározza a kimenő irányú láb feszültségét, vagy tartalmazza a bemenő irányú láb állapotát.

5.6. Serial Peripheral Interface Master (SPI Master)

A tervezett SPI Master egy mester szerepkiosztású, SPI soros kommunikációs protokollt megvalósító periféria, amely külső eszközökkel biztosít SPI kommunikációt [12]. A tervezett SPI periféria full-duplex és simplex kommunikációt bonyolíthat le. A periféria állítható soros órajel frekvenciával, polaritással és adatátviteli fázissal rendelkezik. Az átviteli keret nagysága automatikusan kezelt.

5.6.1. Az SPI Master működésének leírása

Full-duplex és simplex átviteli módok

Az SPI Master periféria által megvalósított full-duplex SPI adatátvitel négy jelet használ. Ezek a következők:

MOSI: Master Out, Slave In. A kiküldött adat vonala. Push-pull kimenet. Alap állapotban 0 értéket tart.

MISO: Master In, Slave Out. A befogadott adat vonala. Külső felhúzást igényel.

SCLK: Serial Clock. A soros kommunikáció ütemjele. Push-pull kimenet.

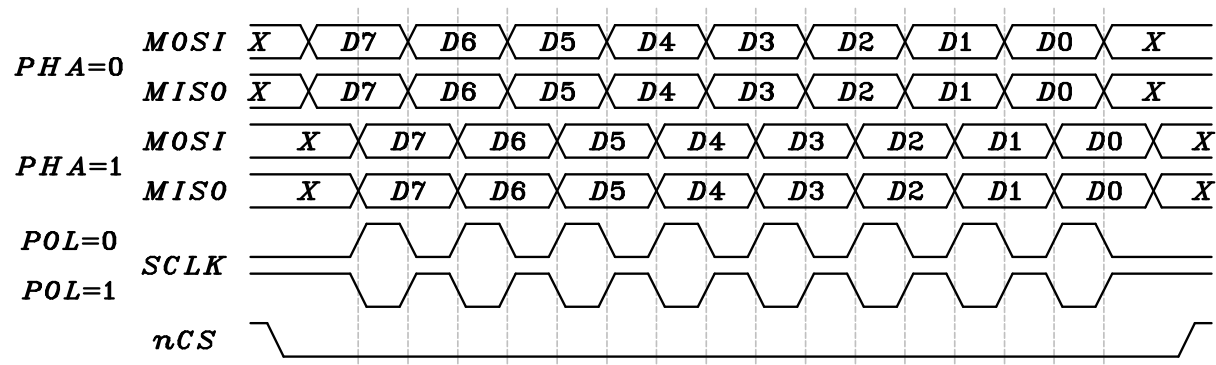
nCS: Negated Chip Select. A cél slave perifériát kiválasztó jel. Jelenleg nincs dedikált láb erre szánva, egy általános GPIO láb és a szoftveres irányítás szükséges.

Full-duplex kommunikáció során a kimenő és bemenő kommunikáció egyszerre történhet. Az SPI Master simplex, egyirányú SPI kommunikációt is támogat, hisz ez csak azt jelenti, hogy csak a MOSI vagy a MISO vonal használt.

Adatkeret beállítások

Különböző külső chippek, slave hardverek különböző SPI konfigurációt igényelnek, az SPI Master lehetővé teszi ezek módosítását a megfelelő beállítás regiszterekkel, az adat átviteli sorrendjét kivéve, ami minden esetben a legnagyobb helyértékű bittől a legkisebb helyértékűig folyik.

Órajel fázis és polaritás beállítása Beállítható az SCLK órajel polaritása, ez a POL bittel történik. Hasonlóképpen az adatátvitel, illetve olvasás fázisa a PHA beállítás bittel állítható. Ezen két bittel négyféle konfiguráció állítható be.



43. ábra: Az SPI Master adatkeret beállítások 8 bites átvitel.

Mód 0: $POL = 0$, $PHA = 0$. SCLK alap állapotban 0. A bitek fogadása az SCLK felfutó élén, a küldése a lefutó élén történik.

Mód 1: $POL = 0$, $PHA = 1$. SCLK alap állapotban 0. A bitek fogadása az SCLK lefutó élén, a küldése a felfutó élén történik.

Mód 2: $POL = 1, PHA = 0$. SCLK alap állapotban 1. A bitek fogadása az SCLK lefutó élén, a küldése a felfutó élén történik.

Mód 3: $POL = 1, PHA = 1$. SCLK alap állapotban 1. A bitek fogadása az SCLK felfutó élén, a küldése a lefutó élén történik.

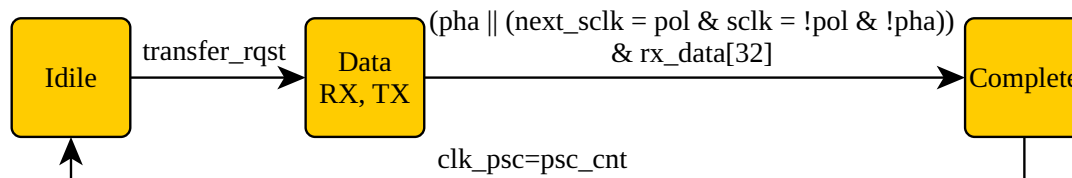
Az átviteli sebesség beállítása Az SPI SCLK ütemjelének frekvenciája a periféria órajelének osztásával áll elő. Az osztószám a CLK_PSC bitmezővel állítható.

$$f_{SCLK} = \frac{f_{LovagSoC_clk}}{CLK_PSC + 1} \quad (5.1)$$

A keretméret beállítása Az SPI Master a keretméretet automatikusan állítja be. Egy átvitel kezdeményezéséhez szükséges a DAT regiszter írása. Ez a regiszter 8, 16 és 32 bites írással/olvasással is hozzáférhető. Az SPI Master a DAT regiszter írásának bitszélességével egyenlő mennyiségű bitet küld és fogad.

Egy adatátvitel részletes leírása

Az SPI Master az adatátvitel folyamatát egy egyszerű állapotgéppel kezeli. Kezdetkor az állapotgép az "Idile" nyugalmi helyzetből indul. Ekkor $SCLK = POL$ és $MOSI = 0$. Egy keret kezdete előtt a felhasználónak szoftveresen kell az nCS vonalat egy GPIO lábbal lehúznia, ha több szolga eszköz is használatban van az SPI buszon.



44. ábra: Az SPI Master modul állapotgépének ábrája.

A keret akkor kezdődik, amikor a felhasználó a DAT regiszterbe írja a küldendő adatcsomagot.⁷ Ekkor az állapotgép átlép a "Data RX, TX" állapotba és a BUSY bit aktívvá válik, megkezdődik az SCLK órajel generálása. Az órajel adott élein a beállításoknak megfelelően történik a bitek küldése, olvasása. A küldéshez és fogadáshoz két shift regiszter használt. A DAT-ba való íráskor az adat a TX shift regiszterbe kerül (tx_data), a fogadott adat pedig az RX-be (rx_data) a transzfer során. A keret kezdetekor az RX-be a 0. helyértékre 1 kerül. A shiftelés során ez jelzi a beolvasott és kiküldött bitmennyiséget. Az átvitel végeztével az állapotgép a "Complete" állapotba kerül, majd egy fél SCLK periódus leteltével visszatér a nyugalmi "Idile" állapotba, a BUSY bit inaktívvá válik. A beolvasott adat a DAT regiszterben található olvasásra. Ezek után a felhasználó az nCS lában ismét felhúzhatja szoftverrel.

⁶ $CLK_PSC = 0$ esetén is a maximum frekvencia $f_{LovagSoC_clk}/2$.

⁷Olvasási szándék esetén 0-t írhatunk a DAT-ba, ekkor a keret végéig a MOSI vonal alap állapotban marad.

Ha egy átvitel közben új adat íródik a DAT-ba, az figyelmen kívül marad, és folytatódik az eddigi művelet.

5.7. Universal Asynchronous Receiver Transmitter (UART)

A tervezett UART (Universal Asynchronous Receiver Transmitter) periféria lehetőséget ad az iparban egyik legelterjedtebb, legegyszerűbb kommunikációs protokoll használatára. Az UART protokoll segítségével külső eszközökkel full-duplex aszinkron módon kommunikálhatunk. A tervezett UART periféria rendelkezik az általánosan elvárt adatkeret beállítási paraméterekkel [12].

5.7.1. Az UART működésének leírása

Kimenetek, full-duplex és simplex átviteli módok

Az UART modul full-duplex mód használata esetén egymástól függetlenül, egyszerre tud adatot fogadni és küldeni, aszinkron módon. Ezt két vonalon végzi:

TX: Transmit (Küldés) vonal. Az UART perifériától külső egység felé küldött adat vonala.

RX: Receive (Fogadás) vonal. A külső egységtől az UART periféria által fogadott adat vonala.

Az RX és TX vonalak egymástól függetlenül, különböző perifériákra is csatlakozhatnak. Így simplex egyirányú kommunikáció is használható.

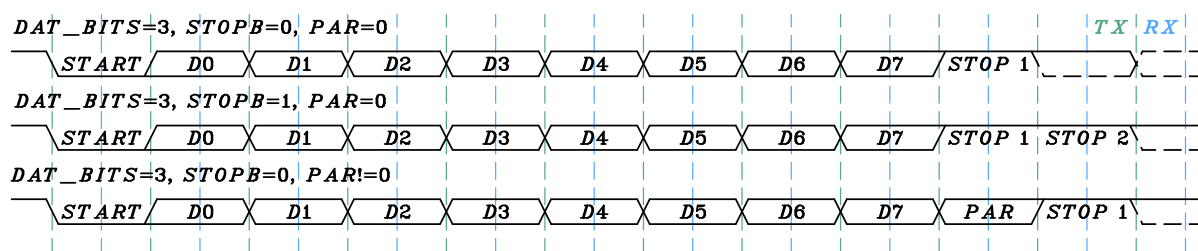
Adatkeret beállítások

Az UART periféria lehetőséget biztosít a leggyakoribb keret Konfigurációk használatára. A választott beállítások a küldött és fogadott keretre egyaránt érvényesek. A beállítások csak akkor módosíthatók, ha $BUSY = 0$, egyébként a kívánt módosítások nem történnek meg.

Az adatméret beállítása Az UART átviteli keret adatszélessége alaphelyzetben 8 bit. Ez állítható 5 és 9 között a DAT_BITS bitmező segítségével. Az egy kezet által átvitt bitmennyiség tehát $5 + DAT_BITS$.

Stop bitek számának állítása A UART protokoll részeként a keret végét 1 értékű STOP bit jelzi, egyes eszközök egy, mások két ilyen bitet várnak. Az UART periféria alaphelyzetben egy STOP bitet használ, ez a STOPB bittel állítható szükség szerint.

Paritás ellenőrzés beállítása A fogadott adatkeret helyességének ellenőrzéséhez az UART protokoll lehetőséget biztosít paritás bit használatára. A paritás bit értéke attól függ, hogy a adatban páros vagy páratlan számú 1 található. Az UART periféria a PAR beállításával ezt a



45. ábra: Az UART átviteli keret, beállítások.

paritás bitet meghatározza, egy keret küldésekor az adatbitek után helyezi. Ugyanez a beállításbit a fogadott adat paritás ellenőrzését is engedélyezi. Ha a fogadott keret paritás bitje nem egyezik a számított paritással, az keret érvénytelen, az olvasott adat eldobásra kerül.

Bitfrekvencia beállítása Az UART periféria lehetőséget biztosít a bitráta vagy bifrekvencia állítására. Fogadáskor a mintavételezés frekvenciáját és a küldés frekvenciáját egy azonos előosztó által meghatározott külön generált órajel alapozza. Ez az előosztó a CLK_PSC bitmezővel állítható 16 bites felbontásban.

$$f_{bit} = \frac{f_{LovagSoC_clk}}{8 \cdot CLK_PSC + 2} \quad (5.2)$$

Az alábbi táblázat foglalja össze a leggyakoribb bitfrekvenciák előállításához szükséges előosztó értékeket.

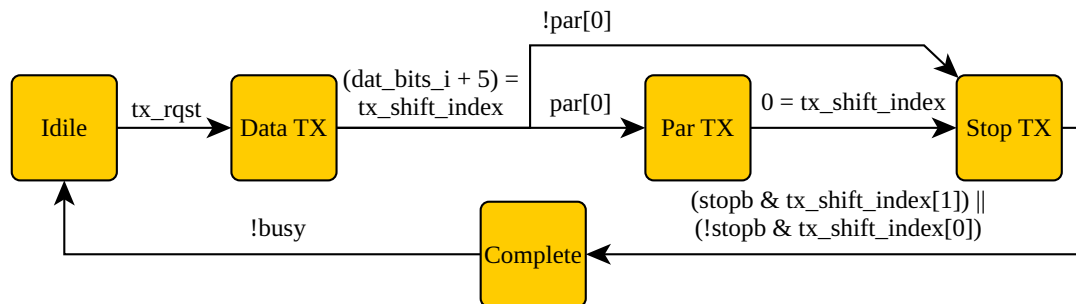
3. táblázat: Szabványos bitfrekvenciák előállítása.

f_{bit} [bit/s]	CLK_PSC	f_{bit} [bit/s]	CLK_PSC
110	56818	19200	325
150	41666	38400	163
300	20833	57600	108
1200	5208	115200	54
2400	2604	230400	27
4800	1302	460800	13
9600	651	921600	7

Egy adatátvitel részletes leírása

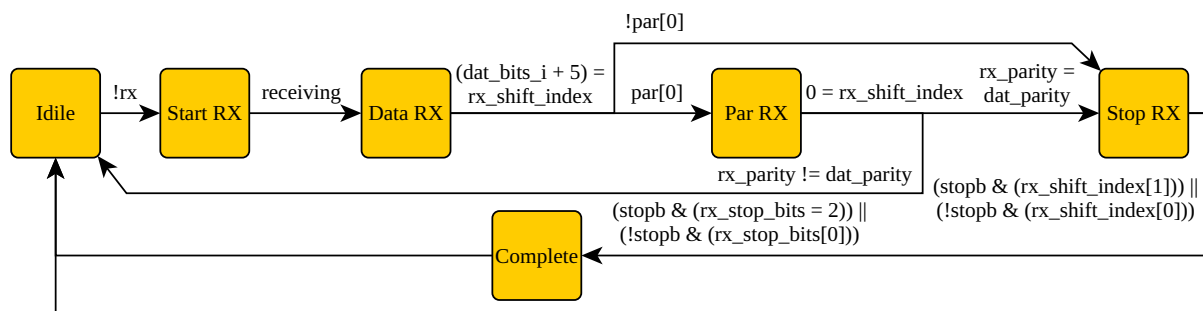
Az UART periféria két alegységgel végzi a kommunikációt. Az UART_rx a fogadást, az UART_tx a küldést kezeli, mind a saját állapotgépével. A két egységnek köszönhetően a küldés és fogadás folyamata egymástól független, egyszerre, aszinkron módon folyhatnak.

Küldés, TX DAT regiszter írását követően a TX láb az alacsony értéket veszi fel, ez jelenti a keret kezdetét mutató START bitet. A BUSY bit jelezni kezdi a folyamatban lévő TX műveletet. Az állapotgép átlép a "Data TX" helyzetbe, megkezdődik a bitfrekvenciának megfelelő ütemjel generálása. A START bit leteltével a küldendő adat kiküldésre kerül, a megadott bitmennyiségben. Közben a küldött adat paritása kiszámolásra kerül. ⁸ Mikor az utolsó adatbit is ki lett küldve,



46. ábra: Az UART_tx modul állapotgép táblázata.

ha a felhasználó választotta, a "Par TX" állapotban kiküldetik a paritás bit is. Ezután a "Stop TX" állapotban a keret befejeztét jelző STOP bit(ek) shiftelődnek ki, mennyiségük 1 vagy 2, a felhasználó szabja meg. A keret végeztével a "Complete" állapotban az utolsó STOP bit periódusának lejártá után a BUSY bit törlődik és az állapotgép az "Idile" állapotba lép, ahol kész a következő adat fogadására. Ha a DAT regiszter a keret közben kap írást, az újonnan kapott adat eldobásra kerül.



47. ábra: Az UART_rx modul állapotgépe.

Fogadás, RX Az állapotgép "Idile" állapotban indul. Az RX láb lehúzását követően a "Start RX" állapotban fél bit periódus elteltét megvárva ellenőrzésre kerül, hogy még mindig lehúzott-e az RX bemenet. Ha igen, akkor megkezdődik az előre meghatározott számú adatbitek beolvasása a belsőleg generált órajel alapján. Közben számolásra kerül a paritás értéke.

Az adat bitek fogadása után, ha a felhasználó beállította, a paritás bit beolvasódik. Ha az olvasott érték nem egyezik meg a számított paritással, a keret eldobásra kerül. Az RX keretet a megadott számú STOP bitek olvasása zárja ha ezek közben a beolvasott érték nem 1, a keret

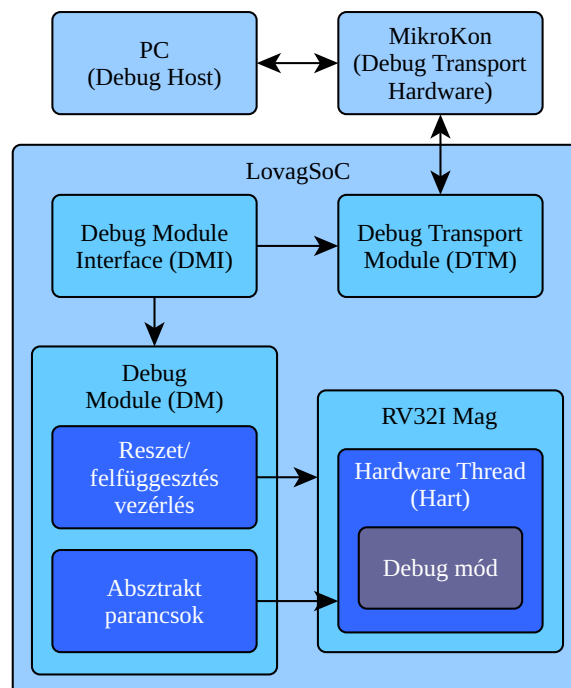
⁸Ez annyiból áll, hogy ha $data[tx_shift_index] = 1$, a paritás értéke invertálódik.

szintén eldobásra kerül. Végül a "Complete" állapotban beállításra kerül a DAT_RDY bit, jelezve az adat megérkezését.

5.8. Debug támogatás

Fontos tudnivaló: A jelen v0.12. (fejlesztői) változatban a tervezett debug rendszer nem felel meg teljesen a szabvány előírásainak. Ezért még nem használhatók a RISC-V debug szabványra épülő szoftverek. Jelenleg saját fejlesztésű szoftver biztosít lehetőséget igen alapszintű debug feladatok elvégzésére.

Egy mikrovezérlőre való szoftver fejlesztéséhez elengedhetetlen a hardveres debug támogatás. A LovagSoC debug rendszere a RISC-V International által közzétett RISC-V processzor szabványhoz tartozó "Minimal RISC-V Debug Specification 0.13.2" külső debug támogatás szabványa alapján készült. [20] A RISC-V debug szabvány használatával a jövőben lehetővé



48. ábra: A LovagSoC debug rendszere.

válhat létező debug szoftver (pl. OpenOCD) használata. A tervezett debug rendszer több fő egységből áll. Ezek belülről kifelé haladva: RV32I processzor a debug funkciókat lehetővé tevő logikai kiegészítésekkel, a debug logikát tartalmazó Debug Modul (DM), a Debug Modul Interface (DMI), a Debug Transport Modul (DTM), kilépve a LovagSoC-n kívülre a Debug Transport Hardver (DTH) és a Debug Host.

A Debug Host egység feladata a debug parancsok generálása, a praktikus, felhasználó által megítélt debug feladat vagy egyéb debug felülettel kapcsolatos feladat elvégzésének részeként. A hardveres követelményei a Debug Transport Hardverrel való csatlakozási és kommunikációs

képesség. Az egyéb hardveres megfontolásokat a fő mérvadó, a szoftveres követelmények adják. A szoftver követelmények részletes elemzése túlmutat ezen dokumentum keretein, de alapvetően szoftverfejlesztői eszközök debug funkciói és szoftver memóriába való feltöltése alkotják. A megvalósított rendszerben a Debug Host szerepét egy személyi számítógép (PC) látja el. A PC a DTH-val USB protokollon keresztül kommunikál. A saját fejlesztésű Debug Host szoftvere a programozási útmutatóban kerül bemutatásra.

A Debug Transport Hardver feladata a DebugHost által küldött debug parancsok továbbítása a DTM felé, és egyéb debuggolással kapcsolatos feladatok ellátása, a Debug Host parancsai alapján. Ennek a hardveres egységnek a hardveres követelményei a Debug Hosttal és a Debug Transport Modullal való kommunikációra való képesség és a DTH szoftver futtatására elegendő számítási teljesítmény. A szoftveres követelmény a Debug Host által adott feladatok elvégzése. A LovagSoC által használt DTH jelen verzióban a Debug Host szoftver egyedisége miatt szintén egyedi megoldás, a MikroKon fejlesztői panel. Más programozható eszköz is használható USB és SPI kommunikációs képességgel, de a jelenlegi DTH szoftver csak a MikroKon kártyán működik. A saját fejlesztésű DTH szoftvere a programozási útmutatóban kerül bemutatásra.

5.8.1. Debug Transport Modul (DTM)

A Debug Transport Modul összeköttetést, áthidalást biztosít a Debug Modul és a rendszeren kívüli Debug Transport Hardver között. A külső hardverrel alkalmazott kommunikáció az SPI protokoll full duplex kivitelben.

A DTM egy SPI Slave egységből és egy Wishbone Master buszfelület egységből áll. A DTM-et és a Debug Modult a Debug Modul Interface (DMI) köti össze, ami jelen esetben egy Wishbone alapú busz.

Az SPI Slave egység azonos a MotVez egyedülálló használatakor alkalmazott modullal, a különbség az adatkeret hosszában található, amely jelen esetben 16 bit helyett 32 bit.

5.8.2. Debug Modul (DM)

A Debug Modul tartalmazza a debuggolási parancsok végrehajtásához szükséges logikát. Feladata a hart-ok (hardware thread - hardver szálak) kezelése a debugger utasításainak megfelelően. A Debug Modul lehetővé teszi a hart futásának felfüggesztését, folytatását, reszettelését, a regiszter készlet és a memória olvasását és írását.

A Debug Modul működésének leírása

A Debug Modul alapvetően a buszműveleteken keresztül kapott parancsokat hajtja végre. A DTM a DMI-n (Debug Modul Interface) keresztül juttatja el a debugger által küldött parancsokat. A DMI szerepét a LovagSoC-ben egy Wishbone busz tölti be, amelyen a Debug Modul szolgál, a DTM pedig a mester szerepét tölti be.

5.9. Programozói útmutató

A dokumentáció következő fejezetében a LovagSoC programozásának módszere és a használt programozói környezet felépítése lesz kifejtve. Ehhez tartozik a mikrovezérlő felprogramozásához használandó MikroKon beágyazott hardver felületei és szoftvere.

5.9.1. A programozói környezet leírása

A LovagSoC-re való szoftver írása és futtatása az assembly, a c és a c++ nyelven támogatott. A jelenlegi szoftveres támogatás a c nyelvre fókuszál, ennek oka, hogy gyorsabb fejlesztés tesz lehetővé, mint az assembly, és könnyebben lehet kis program- és adatmemória felhasználású programokat írni vele, mint a c++-al. Beágyazott c kód fordításához szükséges egy c szabványú fordító, egy startup fájl és egy linker szkript. A szoftver fejlesztéséhez és feltöltéséhez Linux alapú rendszerre van szükség.

A fejlesztői környezet felállítása

A kód fordításához és feltöltéséhez egyes szoftverek megléte szükséges. Ezek:

- Linux operációs rendszer
- Git
- GNU Make
- Python3
- RISC-V GNU Compiler Toolchain

Továbbá szükséges még a felsorolt szoftverek követelményeinek teljesítése. A RISC-V GCC fordító telepítéséhez az alábbi GitHub repozitórium klónozása szükséges.

```
git clone https://github.com/riscv-collab/riscv-gnu-toolchain.git
```

A telepítés előtt követni kell a repozitórium honlapján található utasításokat. A telepítéshez a klónozott mappában szükséges az alábbi két sor futtatása. A konfiguráció beállítja, hogy a *multilib* változat lesz telepítve.

```
./configure --prefix=/opt/riscv --enable-multilib  
make -jn
```

ahol *n* a fizikai processzor magok száma a telepítés gyorsításáért.

Az előre elkészített szoftveres környezet a következő GitHub minta projektben érhető el: https://github.com/KoBenedek/LovagSoC_program, ez a következőképpen is beszerezhető:

```
git clone https://github.com/KoBenedek/LovagSoC_program.git
```

Ez a minta egy példa szoftvert tartalmaz a *Code* mappában, ami segítséget nyújthat saját kód írásában. A *Device* mappa tartalmazza a hardverfüggő, fordításhoz szükséges fájlokat és a LovagSoC periféria regisztereit tartalmazó *LovagSoC.h* header fájlt.

A startup fájl A startup fájl egy assembly program ami a c kódban levő *main* függvényt hívja meg, az előtte szükséges műveleteket végzi el. A LovagSoC jelen állapotában a startup fájl csupán a veremmutatót állítja a RAM végére. A veremmutató értéke a 2. táblázat alapján 7168.

A linker szkript A linker szkript feladata az egyes adatrészek elhelyezése a memória térképen belül. A linker szkript definiálja a létező memóriák méretét és helyét, a ROM 6kB hosszú és a 0. helyről kezdődik. A RAM 1kB-os és a ROM végétől kezdődik.

Program feltöltése a LovagSoC-re

Az elkészült program fordításához a

```
make
```

parancs használandó. Első futtatásnál elkészíti a fordítás kimeneti fájljait és annak *Build* mappáját. Feltöltés előtt csatlakoztatni kell a PC-hez és a LovagSoC-hez a programozó hardvert (jelenleg ez a MikroKon). Ezután a feltöltés az alábbi parancssal történik.

```
make program
```

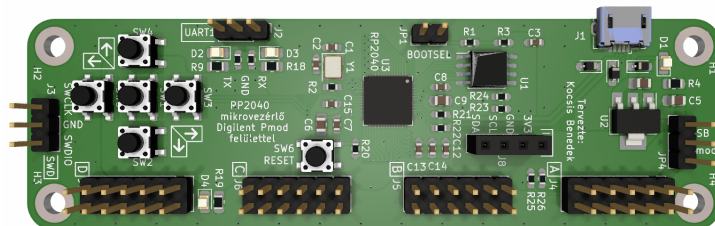
Ez elindít egy szkriptet, ami debug módba helyezi az RV32I processzort, a memóriába írja a programot és reszet után elindítja a program futtatását. Az első feltöltés előtt szükséges a *upload_prog.py* fájlban a programozó hardver csatlakoztatott soros COM portjának megadása.

5.9.2. A MikroKon programozó és debug hardver leírása

A LovagSoC jelenlegi v0.12. változatában a korlátozott debug támogatás miatt a MikroKon mikrovezérlőt használtam a fejlesztett szoftver feltöltésére.

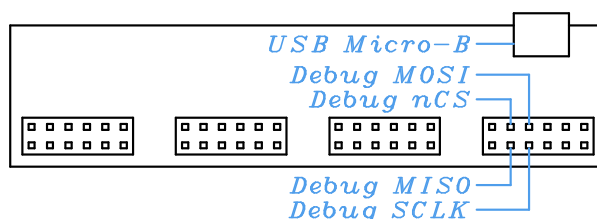
A MikroKon hardver debug felülete

A tervezett MikroKon fejlesztői panel többek között rendelkezik a PC-hez való csatlakozáshoz szükséges USB csatlakozóval és a LovagSoC debug portjához szükséges SPI kimenetekkel. A



49. ábra: A MikroKon fejlesztői kártya.

LovagSoC debug csatlakozás a tűskesorral történik. A 50. ábra szemlélteti, hogy mely tűskesor lábak mely SPI jelet kezelik.



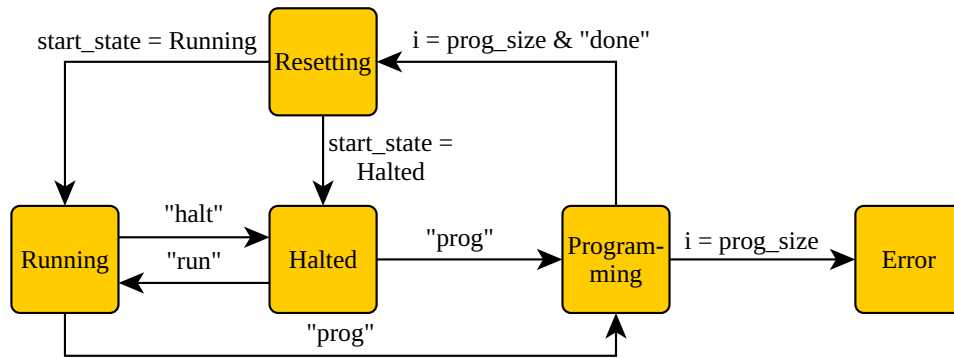
50. ábra: A MikroKon debug csatlakozó felületek.

A MikroKon szoftver debug felülete

A tervezett szoftver bekapcsoláskor inicializálja az USB és SPI kommunikációt és a felhasznált ki- és bemenő lábakat.

A szoftver alapvetően egy állapotgép köré épül, lásd 51. ábra. Ezt az állapotgépet az USB-n keresztül küldött adatok léptetik. Bekapcsolás után az állapotgép a "Running" állapotban ül, ekkor a hozzákötött processzor a jelenleg rajta lévő szoftvert futtatja. Ebben az állapotban a MikroKon a "halt" szövegdarab beolvasása hatására felfüggesztés parancsot ad a LovagSoC-nek és az állapotgép a "Halted"-ba kerül. A Halted állapotból a "run" olvasásával ad folytatás parancsot a LovagSoC-nek, visszalépve a Running-ba. Mind a Running és Halted állapotban "prog" szöveget beolvasva az állapotgép a "Programming" állapotba lép, ahol, ha nincs felfüggesztve a CPU, akkor megteszi.

A Programming állapotban először a program méretét (utasítások száma) olvassa be, ezután egyesével beolvassa a PC-től az utasításokat és feltölti őket a LovagSoC memóriájába, ellenőrzésként pedig visszaolvassa a és kiküldi a PC-nek a memória értéket. Közben egy villogó LED



51. ábra: A MikroKon szoftver debug állapotgépe.

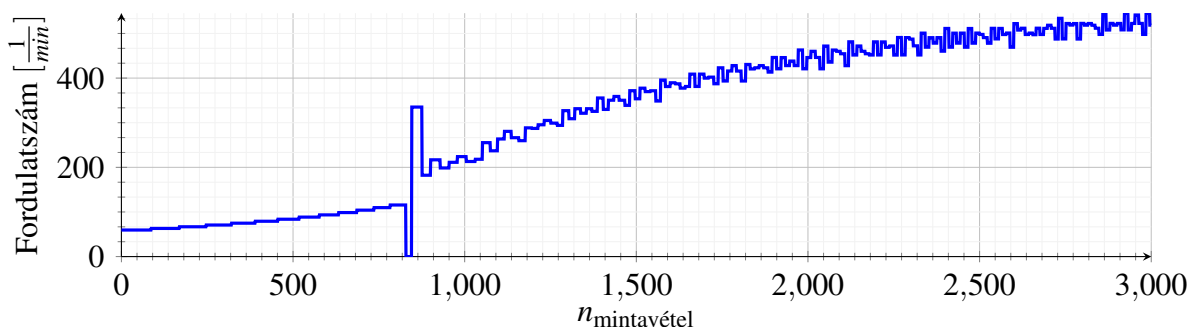
szolgáltat visszajelzést. A programozás végeztével (a megadott utasításszám beérkezése után, $i = prog_size$) a MikroKon "done"-t olvas be, ha nem, akkor hibára fut. Az "Error" állapotból csak reszeteleléssel kerülhet ki az állapotgép.

Ha sikeres a programozás, akkor a "Resetting" állapotban újraindításra kerül a CPU, ezután pedig megkezdődik a kód futtatása ha futás állapotban indult a programozás, egyébként a Halted állapotba kerül vissza az állapotgép. Ha a MikroKon vagy a processzor nem kívánt állapotba kerülne, először a MikroKon-t kell újraállítani, ezután a LovagSoC-t.

6.

Összefoglalás

A dolgozatban bemutatásra kerültek a bevezetőben felvezetett egységek. A 52. ábrán látható egy LovagSoC-t alkalmazó motor (lásd 54. ábra) felpörgetésének sebességgörbéje. A 800. mintavétel környékén vált át a motorvezérlő diktált kommutációról az általános szenzor nélküli működésre. A mérést közvetlenül a LovagSoC végezte, a sebességet a MotVez periféria szolgáltatta, az értékeket UART-on keresztül küldte ki a szoftver a PC-nek. A motor megbízható felpörgetéséhez szükséges volt a felpörgetést testreszabó regiszter értékek kísérleti módon való behangolása.

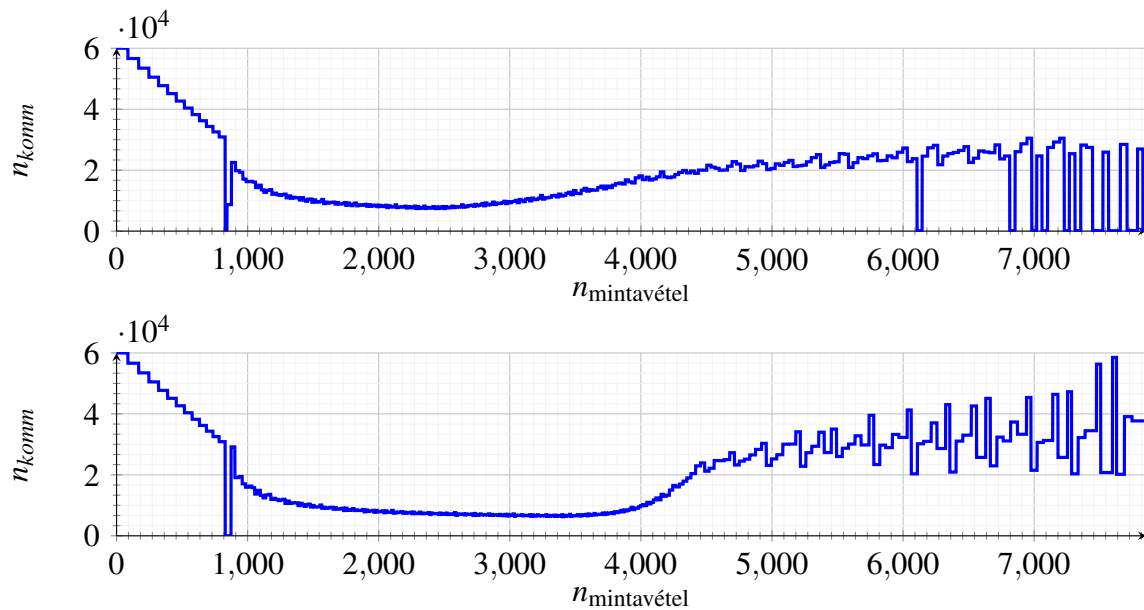


52. ábra: LovagSoC teszt futam, motor felpörgetés.

A 53. ábrán látható egy, a demagnetizációs pulzus szűrőt próbára tevő mérés. A mérés során a futó motort kézzel leterheltem, a LovagSoC pedig folyamatosan küldte UART-on a mért kommutációs periódus időket (amely fordítottan arányos a fordulatszámmal). Látható, hogy szűrés nélkül a demag pulzusok elrontják a mért nullpontmetszések közti időt, a demag szűrőt használva viszont a pulzusok eltűnnek, a sebesség zajos jellege itt az alacsony motor fordulatszám miatti bizonytalan back-EMF komparálás miatt mutatkozik.

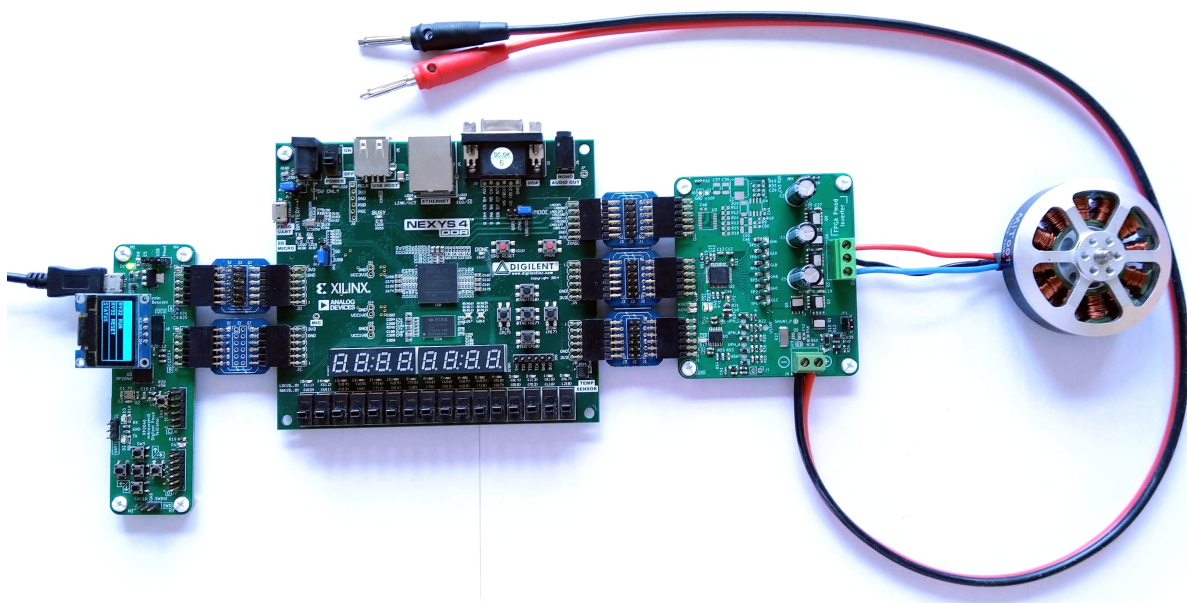
A tervezett BLDC vezérlő logika a szenzor nélküli pozíció szűrési technikáival az alkalmazott szenzor nélküli módszer adta lehetőségeket a gyakorlatban is kihasználnak mutatta. A hajtott motorok tapasztalat szerint megbízhatóan futottak a BLDC meghajtó híd minimális és maximális támogatott feszültségén is. A tervezett motor vezérlő logika ventilátorok és szivattyúk motorjának vezérléséhez kiválóan alkalmas.

A dolgozat utolsó fejezetében bemutatásra került a tervezett mikrovezérlő, melynek részét képezte a MotVez motorvezérlő logika. A gyakorlatban a bemutatott perifériák rendeltetésük



53. ábra: LovagSoC teszt futam, demag pulzus szűrő teszt eredmények. A demag szűrő kikapcsolva alul, bekapcsolva felül. (Az n_{komm} a ZC_PD regiszter kiolvasott értéke.)

szerint viselkedtek, a mikrovezérlő képes volt a feltöltött programokat megfelelően elvégezni. A LovagSoC használatával a rendszer fizikai mérete nagymértékben csökkent, hiszen a külső mikrovezérlő használata szükségtelenné vált. A LovagSoC-ben motor futtatása a rendszer indításakor inicializálás után egyetlen sor kód végrehajtásával történik. A motor futása közben nem vesz el processzor időt más feladatokról. A motor még debuggolás közben is tovább fut, így motor futtatást alkalmazó felhasználás esetén is biztonságosan felfüggeszthető a processzor futása.



54. ábra: Fotó a LovagSoC teszt összeállításáról.

A tervezett mikrovezérlő a piacon elérhető (bár motorvezérlő perifériával nem rendelkező)

alternatívákhoz képest több szempontból is elmarad. A jelenlegi, v0.12. változatában, még szegényes periféria választékot kínál. A mikrovezérlő processzora nem támogat megszakításkezelést, a rendszer nem alkalmaz DMA-t. A processzor utasítás végrehajtása nem futószalagos kivitelű, a RISC-V hardveres szorzás és osztás utasítás kiegészítő (M) nincs megvalósítva.

A MotVez motorvezérlő is hiányosnak tekinthető az egyes piaci alternatívákhoz képest. A MotVez jelenlegi, v1.12. változatában nincs beépített általános üzemhiba kijelzés, így nincs például motor lefulladás érzékelésre lehetőség.

A dolgozat írása közben folyamatban van a Verilog modulok formális módszerrel való igazolása, ami a tervezett rendszert professzionális környezetben is alkalmazhatóvá teszi majd. Az említett hiányosságok pótlása a formális igazolás után következő rendszer verziók feladata lesz.

A formálisan igazolt és kiegészített LovagSoC szilícium alapon is megvalósul majd, (legalább a GDSII gyártási tervek elkészültéig) a nemrégiben nyilvánosan elérhetővé vált SKY130 technológián.

Irodalom

- [1] Mohd Arif Abu Hassan és tsai. „Efficiency comparison of trapezoidal and sinusoidal method for brushless dc motor drive”. *Applied Mechanics and Materials*. 785. köt. Trans Tech Publ. 2015, 248–252. old.
- [2] *AN1946 Sensorless BLDC Motor Control and bEMF Sampling Methods With STMC7*. Techn. jel. STMicroelectronics, 2007.
- [3] *AVR444 Sensorless control of 3-phase brushless DC motors*. Techn. jel. Atmel Corporation, 2005.
- [4] Kocsis Benedek. *Fűnyíró motor vezérlésének tervezése*. 2021.
- [5] Claire Chang és Tilden Chen. *Why Pre-Charge Circuits are Necessary in High-Voltage Systems*. Techn. jel. Texas Instruments, 2021.
- [6] Shaohua Chen, Gang Liu és Lianqing Zhu. „Sensorless startup strategy for a 315-kW high-speed brushless DC motor with small inductance and nonideal back EMF”. *IEEE Transactions on Industrial Electronics* 66.3 (2018), 1703–1714. old.
- [7] *DRV8300: 100-V Three-Phase BLDC Gate Driver Datasheet*. Texas Instruments. 2020. szept.
- [8] *DRV8305 Three Phase Gate Driver With Current Shunt Amplifiers and Voltage Regulator Datasheet*. Texas Instruments. 2016. febr.
- [9] A. Emadi. *Handbook of Automotive Power Electronics and Motor Drives*. Electrical and Computer Engineering. CRC Press, 2017. ISBN: 9781420028157. URL: <https://books.google.hu/books?id=40duBwAAQBAJ>.
- [10] Jose-Carlos Gamazo-Real, Ernesto Vázquez-Sánchez és Jaime Gomez-Gil. „Position and Speed Control of Brushless DC Motors Using Sensorless Techniques and Application Trends”. *Sensors* 10 (2010. júl.). DOI: [10.3390/s100706901](https://doi.org/10.3390/s100706901).
- [11] J.F. Gieras és M. Wing. *Permanent Magnet Motor Technology: Design and Applications, Second Edition*, Electrical and Computer Engineering. Taylor & Francis, 2002. ISBN: 9780824707392. URL: <https://books.google.hu/books?id=rJD-nQEACAAJ>.
- [12] Bettina Weiss Günther Gridling. *Introduction to Microcontrollers*. Courses 182.064 & 182.074. Version 1.4. Vienna University of Technology, 2007.
- [13] *Hardware design with RP2040*. Techn. jel. Raspberry Pi Ltd, 2020.

- [14] A. Hughes. *Electric Motors and Drives: Fundamentals, Types and Applications*. Electrical Engineering. Elsevier/Newnes, 2006. ISBN: 9780750647182. URL: <https://books.google.hu/books?id=IoEYnwEACAAJ>.
- [15] *Ismeretlen forrás, fejlesztés közben elveszett, felkutatása kudarcba fulladt*.
- [16] Kamyar Khosravi és Masahira KURIHARA. *Motor control circuit with degauss filter*. US11374513B2, Allegro Microsystems LLC. 2019. URL: <https://patents.google.com/patent/US11374513B2/en?q=US11374513B2>.
- [17] R. Krishnan. *Permanent Magnet Synchronous and Brushless DC Motor Drives*. CRC Press, 2017. ISBN: 9781420014235. URL: <https://books.google.hu/books?id=U-RxikH3aXEC>.
- [18] Yong Li és tsai. „Rotor position estimation approaches for sensorless control of permanent magnet traction motor in electric vehicles: A review”. *World Electric Vehicle Journal* 12.1 (2021), 9. old.
- [19] Allegro MicroSystems. *Brushless DC Drivers (BLDC)*. URL: <https://www.allegromicro.com/en/products/motor-drivers/bldc-drivers> (elérés dátuma 2023).
- [20] Szerkeztők Tim Newsome és Megan Wachs. *RISC-V External Debug Support Version 0.13.2*. RISC-V Foundation, 2019. URL: <https://riscv.org/wp-content/uploads/2019/03/riscv-debug-release.pdf>.
- [21] Dennis Nolan. *AN4220 Sensorless six-step BLDC commutation*. Techn. jel. STMicroelectronics, 2013.
- [22] Satoshi Ogasawara és Hirofumi Akagi. „An approach to position sensorless drive for brushless DC motors”. *IEEE Transactions on Industry Applications* 27.5 (1991), 928–933. old.
- [23] *Power MOSFET Selecting MOSFETs and Consideration for Circuit Design*. Techn. jel. Toshiba Electronic Devices és Storage Corporation, 2018.
- [24] Jianwen Shao, Dennis Nolan és Thomas Hopkins. „A novel direct back EMF detection for sensorless brushless DC (BLDC) motor drives”. *APEC. Seventeenth Annual IEEE Applied Power Electronics Conference and Exposition (Cat. No. 02CH37335)*. 1. köt. IEEE. 2002, 33–37. old.
- [25] Jianwen Shao és tsai. „A novel microcontroller-based sensorless brushless DC (BLDC) motor drive for automotive fuel pumps”. *IEEE Transactions on Industry Applications* 39.6 (2003), 1734–1740. old.
- [26] Xinda Song és tsai. „High-precision sensorless drive for high-speed BLDC motors based on the virtual third harmonic back-EMF”. *IEEE transactions on Power Electronics* 33.2 (2017), 1528–1540. old.

- [27] *TLE987 Microcontroller with LIN and BLDC MOSFET driver for automotive applications User Manual*. Rev. 1.8. Infineon Technologies AG. 2022. szept.
- [28] P. Vas. *Sensorless Vector and Direct Torque Control*. Monographs in electrical and electronic engineering. Oxford University Press, 1998. ISBN: 9780198564652. URL: <https://books.google.hu/books?id=9ORSAAAAMAAJ>.
- [29] Gaolin Wang, Guoqiang Zhang és Dianguo Xu. „Basic Structure and Mathematical Model”. *Position Sensorless Control Techniques for Permanent Magnet Synchronous Machine Drives*. Singapore: Springer Singapore, 2020, 1–24. old. ISBN: 978-981-15-0050-3. DOI: [10.1007/978-981-15-0050-3_1](https://doi.org/10.1007/978-981-15-0050-3_1). URL: https://doi.org/10.1007/978-981-15-0050-3_1.
- [30] Szerkeztők Andrew Waterman és Krste Asanovic. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 20191213*. RISC-V Foundation, 2019. dec. URL: <https://riscv.org/technical/specifications/>.
- [31] *WISHBONE B4. WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores*. OpenCores, 2010. URL: https://cdn.opencores.org/downloads/wbspec_b4.pdf.
- [32] C. Xia. *Permanent Magnet Brushless DC Motor Drives and Controls*. Wiley, 2012. ISBN: 9781118188361. URL: <https://books.google.hu/books?id=FkRYP7DW09cC>.