Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Bányai Tamás

ANALÓG SZÖGSZENZOR ILLESZTÉSE FPGA-HOZ

Szakdolgozat

KONZULENS

Dr. Balogh András
(ThyssenKrupp Presta Hungary Kft)

Dr. Sujbert László
(BME-MIT)

BUDAPEST, 2014

MŰEGYETEM 1782

SZAKDOLGOZAT-FELADAT

Bányai Tamás (R44P4P)
szigorló villamosmérnök hallgató részére

Analóg szögszenzor illesztése FPGA-hoz

A modern gépjárművek biztonságtechnikai és kényelmi funkcióinak megvalósításában, környezetvédelmi jellemzőinek javításában stb. egyre jelentősebb szerepet kapnak a számítástechnikai megoldások. Az egyik kritikus új funkció az elektronikus kormányrendszer. Ez egy – általában háromfázisú szinkron – motoron keresztül ad rásegítő nyomatékot a kormánygépre vagy a kormányoszlopra. A motor pontos, dinamikus szabályozása komplex, nagy számításigényű feladat, amelyet általában nagyteljesítményű mikrokontrollerek segítségével végeznek.

Érdekes kutatás-fejlesztési feladat a számítások FPGA-n történő implementálása, melynek eredményeként a mikrokontroller terhelése csökkenthető, a valós idejű kód nagy része a programozható áramkörön (későbbiekben sorozatgyártásban ASIC-en vagy ASSP-n) futtatható. A jelölt a megvalósíthatóság vizsgálatába kapcsolódik be, a motorszabályozási hurok egy részét – a szögszenzor illesztését, jelének feldolgozását – valósítja meg. A hallgató feladata magában foglalja az alábbiakat:

- Az alkalmazott GMR elvű szögszenzor megismerése
- Illesztőpanel tervezése a szenzor jelének az FPGA-ba való továbbítására
- A szenzor kimenetén levő sin/cos jelek feldolgozása (normalizálás, ofszetkompenzáció, stb.) FPGA-n
- A sin/cos jelekből szögjel számítása az FPGA-n
- A szükséges FPGA tesztkörnyezet létrehozása
- A feldolgozás redundánssá tételéhez szükséges lépések (két szenzor párhuzamos kezelése, a kimenetek összehasonlítása) megvalósíthatóságának vizsgálata

A feladat megvalósításához szükséges eszközöket a ThyssenKrupp Presta Hungary Kft. biztosítja.

Tanszéki konzulens: Dr. Sujbert László docens
Külső konzulens: Dr. Balogh András (ThyssenKrupp Presta Hungary Kft.)

Budapest, 2014. szeptember 18.

.....
Dr. Jobbágy Ákos egyetemi tanár
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

1117 Budapest, Magyar Tudósok krt. 2. I. ép. I.E.444.
Telefon: 463-2057, Fax: 463-4112
http://www.mit.bme.hu • e-mail: mitadm@mit.bme.hu

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Bányai Tamás**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2014. 12. 10.

...……

Bányai Tamás

Összefoglaló

A mai modern személygépjárművekben már szinte kötelező tartozéknak mondható a kormányzásrásegítés, amelynek segítségével a sofőr biztonságosabban irányíthatja a járművet. A legújabb fejlesztésekben a kormányzásrásegítést elektromos motorokkal valósítják meg, az egyszerűbb fejlesztés és a funkciók bővíthetősége miatt.

Rásegítéskor a motorokkal szemben igen pontos beavatkozási követelmények vannak. A teljesítésükhöz elengedhetetlen egy komplex, nagy számítási igényű szabályozási hurok megvalósítása. A szakdolgozat elkészítése során ebbe a szabályozási körbe nyertem betekintést, és egy részét, a forgórész szöghelyzetének érzékelését és a szenzor jeleinek feldolgozását kellett megvalósítanom FPGA áramkörön.

A dolgozat első felében ismertetem a megszerzett és a feladat megoldásához szükséges elméleti ismereteket. Először rövid áttekintést adok a kormányrendszer felépítéséről és a rásegítés alapvető kérdéseiről. Majd bemutatom a rásegítő elektromos motorok elvárt tulajdonságait, és konkrét típusait. Ezek után részletesen ismertetem a mezőorientált szabályozási kör teljes felépülését, amelybe betekintést nyertem. Kitérek a vezérlés alapelvére, az egyes részegységek feladataira, és egy konkrét példán keresztül is végigkövetem a szabályozási kör működését. Végül részletesen bemutatom a GMR alapú szöghelyzet-érzékelés fizikai alapjait, ami alapján a feladat megvalósult.

A dolgozat második felében a megvalósítás konkrét lépéseit és a döntéseim indoklását írom le. Először ismertetem a méréshez alkalmazott szögszenzor legfontosabb adatait. Majd a szöghelyzet kialakításának egymás utáni lépéseit mutatom be, kiemelve a tervezési megfontolásokat. Kitérek az illesztő nyomtatott áramköri lemez kialakítására, az illesztőkártya és az FPGA áramkör közötti kommunikációra, és az FPGA-ban megvalósított jelfeldolgozási lépésekre, valamint végül ezek tesztelésére is.

A dolgozat végén megvizsgálom a redundáns működés lehetőségét, hogy a szöghelyzet-érzékelő az autóipari szabványoknak is megfeleljen. Végül a jövőbeli fejlesztési lehetőségekre is rávilágítok.

A fejlesztés végeredményeként elkészült az illesztőpanel és az FPGA áramkörbe implementálható jelfeldolgozási lánc. El tudtam végezni az első méréseket is, amelyek alapján az elkészült rendszer képes információt szolgáltatni a forgórész szöghelyzetéről.

Abstract

Nowadays in modern automobiles the power steering function is almost obligatory. This function enables the driver to control the car on a significantly higher safety level. In the newest innovations power steering is achieved with electric motors due to easier development and the possibility for many add-ins.

In electric power steering (EPS) the motors have to meet srict operational requirements. To accomplish these a complex motor control loop is needed, which demands huge amount of computing. I gained insight to a control loop named Field Oriented Control (FOC) and carried a part of it into execution – my exact task was to implement the processing of the signals of the rotor angle sensor on an FPGA circuit.

The first part of my thesis is about the theoretical knowledge required for the task. First, a short overview is given about the steering system and the basics of power steering functionality. The next chapter descibes the requirements of EPS motors and the types of electric motors typically used. Next, the Field Oriented Control is detailed. I have been paying special attention to the basic principle of control and the tasks of every component in the loop – I have also been using a direct example to present the operation of the control loop. Finally, the physical basis of the GMR detection of rotor position is presented, as a GMR based detection has been used in my exact task.

The second part of my thesis is about the particular steps of my solution for the task and the decisions behind them. First, a short summary is given about the specific GMR sensor which has been used. Next, the computing steps of the angle process is described in depth, touching upon the design of the interface panel, the communication between the panel and the FPGA, the process thread in the FPGA and the tests of the process components.

At the end of my thesis the possibility for redundancy in the panel and the process thread is examined in order to meet the regulations of automotive industry. Finally, a short list is given about the opportunities for further development.

As a result of my task, the samples of the designed interface panel have been manufactured and the implementable process thread is made. According to the first measurements the circuits are able to provide information about the rotor angle.

1. Bevezetés

Napjainkban a gépjárműipar egyre szorosabban összefonódik az elektronikai iparral, ugyanis a modern gépjárművekben egyre több elektronikus egység teljesít szolgálatot. Egy-egy elektronikus vezérlőegység (rövidítve ECU – Electronic Control Unit) egy személygépjárműben is igen szerteágazó módon, különféle funkciók megvalósításáért lehet felelős. Például: fékezés vezérlése, menetstabilizálás, motorvezérlés, kormányzásrásegítés stb.

A modern gépjárművekben az elektronikus vezérlőegységek vagy egy régebben más eszközökkel megoldott funkciót váltanak ki, vagy eddig megvalósíthatatlan, új funkcionalitást hoznak létre. Az első esetben a kiváltás oka lehet az autóiparban egyre szigorodó szabványoknak való megfelelés, de akár a környezeti terhelés csökkentése érdekében is szükség lehet erre a lépésre. Kiváltáskor mindig a legfontosabb szempont, hogy az elektronikai megoldás hatékonyabb legyen, mind a fogyasztás terén, mind pedig a költségek terén. Második, új funkció esetében általában kényelmi vagy biztonsági funkciók megvalósítását teszi lehetővé az elektronikus beavatkozás. Ilyenkor az elsődleges cél az innováció (például a fejlett vezetőasszisztens rendszerek). Ezenkívül előfordulhat olyan eset is, amikor egy régi megoldást váltanak ki, azonban az elektronika segítségével a pontosabb beavatkozás vagy érzékelés miatt javíthatnak az adott funkcionalitáson (például az elektronikus kormányrásegítés).

1.1. Témaválasztás

A személygépjárművekben egy ilyen funkció a kormányzás, illetve a kormányrásegítés, amelynél manapság egyre gyakrabban alkalmazzák az elektronikus kormányrendszert, amely egy elektromotor segítségével ad rásegítő nyomatékot a kormányműre, ezzel téve könnyebbé a kormányzást a sofőr számára.

Ezzel a témakörrel foglalkozik a ThyssenKrupp Presta Hungary Kft, amelynél töltött gyakornoki munkám során ismerkedtem meg én is e tématerülettel. Itt kerültem kapcsolatba az ECU-SW osztállyal, ahol ezen elektronikus vezérlőegységek szoftverét fejlesztik. Itt nyerhettem betekintést az elektromos kormányrendszer motorszabályozási körébe, amelynek egy részét – a motor szöghelyzetének érzékelését – kellett FPGA áramkörön megvalósítanom a szakdolgozat elkészítése során.

1.2. A feladat részletezése

A szakdolgozat-készítés közben, nemcsak az eszközök megtervezésére és a működéshez szükséges programok megírására törekedtem, hanem igyekeztem elsajátítani az ezekhez szükséges elméleti háttéranyagot is.

A feladatom a szakdolgozat elkészítése során az volt, hogy ismerjem meg a ThyssenKrupp Presta Hungary Kft-nél alkalmazott kormányrásegítő rendszerek felépítését és működési elvét, különös figyelemmel a mechanikai konstrukcióra, valamint az alkalmazott elektromos motorokra. Ezek után megismerkedtem az ott alkalmazott motorszabályozási kör működésével és fontosabb részegységeivel, funkcióival, külön figyelmet fordítva a szöghelyzetet érzékelő szenzorra és annak kapcsolataira. Ezek ismeretében meg kellett terveznem egy nyomtatott áramköri lemezt, amely illeszkedik a motor konstrukciójához, tartalmazza a szenzort és az érzékeléshez szükséges egyéb alkatrészeket. Végül pedig implementálnom kellett a szenzor jelének feldolgozását egy Xilinx FPGA áramkörön.

2. Kormányrendszer

A kormány a gépjárműben a legfontosabb kapcsolat az út és a sofőr között. Ennek segítségével tudja irányítani a sofőr a járművet, de ezenkívül fontos feladata a vezetési élmény kialakítása is, vagyis hogy a sofőr visszajelzést is kapjon az útfelületről, amin vezet. A kormányrendszer alapvetően befolyásolja a jármű viselkedését és teljesítményét.

2.1. Kormányzásrásegítés

A kormányrásegítő rendszer feladata, hogy a sofőrnek ne kelljen túlzottan nagy erőt kifejtenie a kormánykerékre ahhoz, hogy a jármű elforduljon. A rásegítő rendszer is beavatkozik a kormányzás folyamatába, segít a sofőr erején felül a kerekek elforgatásában, ezzel megvalósítva a könnyebb kormányzást, így téve biztonságosabbá a vezetést.

A rásegítő rendszer célja, hogy minden kormányállapotban – beleértve a statikus és a dinamikus kormányállapotokat is – precízen pozicionálható kormányzást tegyen lehetővé a sofőr számára, úgy, hogy mindig a helyzethez illeszkedő nagyságú, de még nem túl megerőltető mértékű erőt kelljen kifejtenie a kormánykeréken. Emiatt az igény miatt a köznyelvben szervomotornak is szokták hívni.

A rásegítés általában álló helyzetben, vagy lassú haladáskor válik igen fontossá, nagyobb sebesség esetén jóval kisebb rásegítő nyomaték is elegendő. Vagyis a rásegítő nyomaték karakterisztikája az autó sebességének függvényében egy csökkenő görbe. Minden sebességnél fontos azonban, hogy a kormányszerkezet megfelelő visszajelzéseket adjon a sofőrnek az út állapotáról, vagyis a megfelelő vezetési élményt alakítsa ki.

Néhány évvel ezelőtt engedélyezték csak a „drive by wire”, magyarul vezetékkel történő kormányzás megoldásait személygépjárművekben. Ilyenkor a sofőr nem közvetlenül irányítja az autó kerekeit, hanem elektronikus eszközökkel és ezek között létesített adatkapcsolattal megoldott a kormányzás. A korábbi tiltó törvények miatt kötelező volt a mechanikai kapcsolat a kormánykerék és kormányzott kerék között. A nemrégiben történt engedélyezés [1] miatt azonban csak egyetlen gyártó egyetlen

autótípusában érhető el ez a technológia a piacon. Ennél a modelltípusnál is van még azonban tartalék rendszerként egy mechanikai kapcsolat is, arra az esetre, ha az elektronika meghibásodna.

A rásegítésfunkció azonban megengedett és régóta is alkalmazott az autókban, mivel ezzel biztonságosabbá tehető a személygépjárművek kezelése. Figyelembe kell venni azonban azt, hogy ez a funkció is biztonságkritikus, ugyanis az autó alapvető irányításához kapcsolódik a rásegítő rendszer.

Biztonságkritikusnak nevezzük a rendszert, amelynél fontos követelmény, hogy ne veszélyeztethessen emberi életet és egészséget, valamint ne okozhasson gazdasági és környezeti károkat sem. Tehát olyan rendszer, amelynél, ha nem teljesül a szigorúan előírt specifikáció, akkor az balesethez vagy káresethez vezethet.

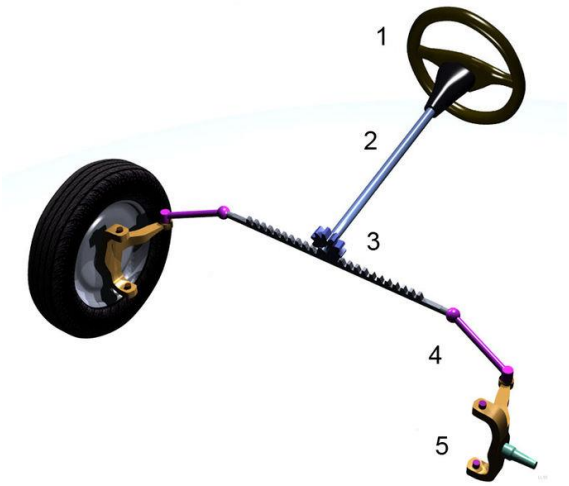
Mivel biztonságkritikusnak kell tekinteni ezt a funkciót, emiatt szigorúbb szabványi feltételeknek kell megfelelnie. Az autóiparban kétféle szabványt alkalmaznak a biztonságkritikus rendszerekre. Az egyik az IEC 61508 szabvány [2], amely egy nemzetközi ágazatfüggetlen üzembiztonsági szabvány, és bármilyen biztonságkritikus alkalmazást támogat (nem csak az autóipart). A másik az ISO 26262 szabvány [3], amely kifejezetten közúti forgalomban működő elektromos és elektronikus rendszerek szabványa. A szabványok teljesítésén kívül ezt a funkciót végrehajtó rendszernek a tervezési folyamata is különleges odafigyelést igényel. Speciális megoldásokra lehet szükség (pl.: redundancia alkalmazása rendszerszinten), és az egész tervezés során a biztonsági analízis elvégzése kiemelt szerepű. Ezenkívül a felelősség miatt hatósági engedély is szükséges lehet az eszköz üzembe helyezéséhez.

Az ilyen biztonságkritikus rendszerek általában hibatűrő rendszerek is, vagyis olyan rendszer, amely fennálló hardver és/vagy szoftver hiba esetén is képes ellátni megfelelően a feladatát. Ez fontos is, hiszen a rendszer ki van téve a környezeti hatásoknak, amelyek károsíthatják, azonban még így sem szabad, hogy emberéletet veszélyeztessen.

Például a kormányzásrásegítés esetében is több ilyen veszélyeztető környezeti hatás lehet. Az egyik a hő, ugyanis a motortérben, a motor közelében is lehet a rásegítő rendszer. Vagy akár a mechanikai vibráció is, ugyanis az útegyenetlenség átadódhatnak a kormányrendszernek is. E hatásokat is ki kell bírnia a rásegítést végző rendszernek,

illetve fatális hiba esetén is olyan állapotba kell kerülnie, hogy az autó kormányozható maradjon, ha a rásegítés esetleg egyáltalán nem is működőképes.

2.2. Kormányszerkezet felépítése



2-1. ábra Kormányszerkezet mechanikus felépítése [1]

1: kormánykerék; 2: Kormányoszlop; 3: kormánymű; 4: nyomtávrúd; 5: tengelycsonk

A 2-1. ábrán látható egy tipikus kormányszerkezet felépítése, és alatta a részeinek a megnevezése. A szerkezet működése a következő [4]. Ha a sofőr elforgatja a kormánykereket, akkor a kormányoszlop is ugyanakkora nyomatékkal elfordul, majd egy fogaskerék segítségével a forgómozgás lineáris mozgássá alakul át a kormányművön (ami kialakításában egy fogasléc), aminek hatására a nyomtávrúdon át a kerekek elfordulnak.

Már a működésen is látszik, hogy a kormányzásra történő rásegítés több helyen is lehetséges. Kettő tipikus alkalmazott rásegítési pont a kormányoszlopra, és a kormányműre való rásegítés, de további típusokat a későbbiekben ismertetek.

2.3. Kormányzásrásegítés típusai

A rásegítéseket a beavatkozás fizikai kapcsolata alapján három nagyobb csoportba lehet osztani [5]. Az első a hidraulikus rásegítés, a második a hibrid rásegítés, és az utolsó pedig a tisztán elektromos rásegítés.

Hidraulikus rásegítés esetén a nyomatékrásegítés hidraulikafolyadék és az autó motorja által szolgáltatott hidraulikanyomás segítségével történik. Ezt a módszert alkalmazták először rásegítés megvalósítására, azonban több hátránya is van. A

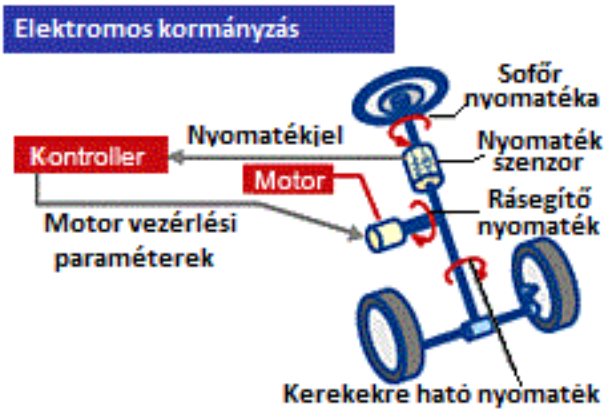
legfontosabbak a bonyolult felépítés, a működéshez szükséges sok vezeték, a kedvezőtlen rásegítési karakterisztika, a pontatlan beavatkozási lehetőség és a környezetszennyező hidraulikaolaj.

Hibrid rásegítés esetén a rásegítés továbbra is hidraulikus rendszer segítségével történik, a szükséges nyomást azonban egy külön elektromos motorral alakítják ki. Ezzel a megoldással kiküszöbölhető a vezetékezés problémája, valamint optimális rásegítési karakterisztika is megvalósítható. Hátrányként továbbra is megmarad a bonyolult felépítés, a környezetszennyező olaj, és a nem teljesen precíz beavatkozás.

A tisztán elektromos rásegítést a magyar nyelvben elektromos kormányzásnak is hívják. Szokás EPS-nek rövidíteni, az angol „Electric Power Steering” kifejezés alapján, illetve EAS-nek, az angol „Electrically Assisted Steering” kifejezés alapján.

Elektromos kormányzás esetén egy elektromotor – általában egy háromfázisú szinkronmotor – van hozzácsatlakoztatva egy fogaskerék segítségével a kormányszerkezethez valamely pontjához, és ez ad rásegítő nyomatékot a kormányszerkezetre a sofőr kormánykerékre kifejtett erején felül.

Az elektronikus kormányzás három főbb részegységből áll össze. Egyrészt az elektromos motorból (amely általában háromfázisú szinkronmotor), amelyet gyakran szervomotornak is neveznek. Másrészt egy nyomatékszenzorból, amely a kormánykerékkel a sofőr által kifejtett nyomatékot érzékeli, az érzékelési pont általában a kormányoszlopon van. Végül pedig az elektronikus vezérlőegységből (ECU), amely az egész rendszert szabályozza, és a különböző bemenő jelekből (pl.: nyomatékszenzor jele, szervomotor pozíció, jármű sebessége, stb.) kialakítja a szervomotor által kifejtendő, a sofőr általi nyomatékra rásegítő nyomatékot.



2-2. ábra EPS egység alkotóelemei [2]

Az elektromos kormányzásnak több előnye is van. Egyrészt teljesen külön egységként kezelhető a rásegítő szerkezet, így könnyen skálázható és variálható a rendszer felépítése, és könnyebben is javítható. Másrészt a külön rásegítő motor miatt akármilyen karakterisztika megvalósítható a rásegítésnél a megfelelő motorvezérlés alkalmazásával, ráadásul ez a megoldás sem csökkenti az autó motorjának hasznosítható teljesítményét. Harmadrészt csak elektronikus egységek alkotják, így nagyon pontos és gyors beavatkozású szabályozást tesz lehetővé, és nem annyira környezetszennyező megoldás, mint a hidraulikus módszer.

Ezek alapján jól látható, hogy a három fő csoport közül az elektromos kormányrendszer nyújtja jelenleg a legjobb megoldást. Emiatt manapság a hidraulikus módszerek egyre jobban kiszorulóban vannak.

A rásegítés helye alapján 4 alcsoportba szokták osztani az elektromos kormányokat [6]:

1. Oszloprásegítő (Column-assist type): ilyenkor az elektromotor a kormányoszlopra fejt ki a rásegítő nyomatékot.
2. Fogaskerék-rásegítő (Pinion-assist type): ilyenkor az elektromotor a kormányoszlop és kormánymű között lévő fogaskerekes kialakításhoz kapcsolódik.
3. Fogasléc-rásegítő (Rack-assist type): ilyenkor az elektromotor a kormányműre, vagyis a fogaslécre fejt ki a rásegítő nyomatékot egy szíj segítségével
4. Direkt vezető (Direct-drive type): ilyenkor egy egységnek tekinthető az elektromotor és a fogasléc, a rásegítés azonban az előző típushoz hasonlóan a fogaslécen valósul meg.

Az oszloprásegítés előnye, hogy felépítése nagyon egyszerű, hátránya azonban, hogy mindenképp közel helyezkedik el az utasfülkéhez, így az általa keltett zaj és vibráció könnyebben zavarhatja a sofőrt. Ezenkívül az oszlop és a különböző kardánok nem viselnek el nagy nyomatékot – illetve túl vastagra kellene őket tervezni nagyobb teljesítmény esetén. Emiatt általában csak kisebb nyomatékú motorokat alkalmaznak ilyen típusnál. A fogaskerék-rásegítő típus előnye, hogy az előző típushoz képest nagyobb rásegítő nyomatékú motor alkalmazható, bonyolult lehet azonban a felépítése.

A fogasléc rásegítő típus esetében is nagy nyomatékot lehet alkalmazni a rásegítésnél, és egyszerű felépítésű is lehet a rendszer.

A ThyssenKruppnál kizárólag csak elektromos kormányszervókat gyártanak és fejlesztenek, ugyanis a modern autókban ez szinte már elvárás. Ezen belül is, a kormányoszlopra történő rásegítést, valamint a fogaslécre történő rásegítést alkalmazzák.

2-3. ábra Kormányoszlop rásegítés, és fogasléc rásegítés [3]

Mindkét megoldás előnye, hogy kis helyen elférő, kompakt szerkezeteket lehet ezzel a módszerrel megvalósítani, és könnyen skálázható is a megoldás a vevői igényeknek megfelelően.

3. Szervomotor felépítése és vezérlése

Az új fejlesztésű autókban szinte kizárólag csak az elektromos kormányzást alkalmazzák rásegítés céljából. Ennél a módszernél mindig egy különálló villamos motor áll rendelkezésre a rásegítő nyomaték kialakításához. Emiatt ehhez a motorhoz egy különálló szabályozási kört is meg kell valósítani, amely teljesíti a rásegítési igény nyomatékkarakterisztikáját, ehhez azonban ismerni kell az alkalmazott motortípusok felépítését és alapvető karakterisztikáit.

3.1. Szervomotorok elvárt tulajdonságai

Általában véve a szervo kifejezés egy olyan rendszert jelent, amelyben a kimenő paramétereket egy negatív visszacsatolású hurok segítségével alakítjuk ki. A legtöbb esetben azonban olyan rendszerekre használják, ahol pontos pozíciószabályozás kialakítása a cél és a pontos szabályozás széles sebességtartományokon át is fenntartható, beleértve az álló helyzetet is. Emiatt leggyakrabban a szervomotor szabályozási köröknél valamilyen szenzor segítségével visszamérik a kialakított szöghelyzetet, és ezt az információt visszacsatolják a vezérlő bemenetére [7].

A szervomotor kialakításakor speciális elvárásoknak kell megfelelni és a motor karakterisztikáját az elérni kívánt céloknak megfelelően kialakítani. Az első legfontosabb elvárás a nagy nyomatéksűrűség, ugyanis nagy nyomatékú motorral nemcsak pontos szöghelyzet szabályozás valósítható meg, hanem gyors beavatkozás is elérhető. A nagy nyomatékra szükség van, mert a rásegítés vagy a kormányoszlopon vagy a fogaslécen történik, ahol viszonylag kicsi az erőkar, emiatt nagy nyomatékigény is keletkezhet a megfelelő rásegítés céljából. Maga a szervomotor azonban a motortérben helyezkedik el, ahol szűkös a hely, emiatt fontos a kis méret is. A második elvárás a kis nyomatékhullámosság, ugyanis csak így valósítható meg pontos szögszabályozás. Továbbá, ha nagy lenne a rásegítő motor nyomatékhullámossága az igen kellemetlen, rázó érzést alakítana ki a kormánykerék forgatásakor, amely akár el is vonhatja a sofőr figyelmét. A harmadik fontos elvárás, hogy a rásegítő motornak nagyon kicsi legyen a tehetetlensége, ugyanis csak így alakítható ki pontos szöghelyzet szabályozás és csak így biztosítható, hogy a szervomotor széles sebességtartományok között is ugyanolyan pontos és nagy dinamikájú beavatkozást tegyen lehetővé.

3.2. Alkalmazott motortípus

Szervomotoros hajtársa igen sokféle villamos géptípust is használnak, manapság a leginkább alkalmazott motortípus azonban a háromfázisú szinkrongép.

Ennek a géptípusnak autóipari szempontból három fő tulajdonságát érdemes megemlíteni [8]. Az első, hogy csak egyetlen fordulatszámon, az állórész mágneses mezejének fordulatszámán, az úgynevezett szinkron fordulatszámon képes csak működni és állandó nyomatékot kifejteni. Minden relatív fordulateltérés során csak zérus középértékű, lengő nyomatékot képes kifejteni. Vagyis ez a géptípus igen mereven kötött a fordulatszámhoz. Az állórész frekvenciájához való merev hozzárendelés volt régebben a legfőbb akadály az elterjedésében, az inverter kapcsolások segítségével azonban ez a probléma megoldható lett. A második fő tulajdonsága az elsőnek a következménye, vagyis, hogy a szinkrongép nem tud indulni. Emiatt járműipari megoldásokra szinte teljesen alkalmatlan is lehetne, de különböző megoldások segítségével (pl.: külön indítómotor alkalmazásával, kalickás indukciós motorvezérléssel, vagy a tápláló inverter megfelelő vezérlésével) elérhető a szinkron fordulatszám. A harmadik fő tulajdonsága az, hogy a legnagyobb teljesítménysűrűségű villamos gép, cos(φ)=1 értékkel. Ennek oka, hogy a szinkrongép forgórészének saját mágneses tere van, amit nem a felvett áramkomponensekből kell kialakítani, így akár kapacitív meddő teljesítményt is képes szolgáltatni, vagyis javulhat a teljesítménytényező. A nagy teljesítménysűrűség, a jó teljesítménytényező és a kevesebb karbantartást igénylő kefe nélküli konstrukciója miatt is ez a legelőnyösebb géptípus, és manapság ezért is terjedt el.

Állandó mágnesek alkalmazásával csökkenthető a gép mérete, ugyanakkora kívánt nyomaték esetén. Ez különösen előnyös lehet az autóipari alkalmazásoknál. Ilyenkor a gép forgórészében az egyenárammal gerjesztett tekercs helyett állandó mágnest alkalmaznak. Ilyen gépek neve állandó mágneses szinkrongép, de a köznyelvben gyakran állandó mágneses gerjesztésű szinkron motornak is hívják. Szokás BPM motorcsaládnak is rövidíteni az angol „Brushless Permanent Magnet” kifejezés alapján.

Felépítését tekintve az állórészen egy általános háromfázisú tekercselés található, amelyet armatúratekercselésnek is szoktak hívni. Ezzel alakítják ki az állórész forgó mágneses mezejét, amelyet armatúramezőnek is neveznek. A forgórészben egy

állandó mágnes található, kettő vagy akár több pólusú elrendezésben. A rögzített pólusokkal ellátott forgórészt szokás póluskeréknek, a mezejét pólusmezőnek is nevezni. Fontos a mágnes északi és déli pólusa közti egyenes, amely a forgórész d tengelyét jelzi, és egyben a főfluxus és a forgórész mágneses erővonalainak iránya is.

A motor nyomatékát a két mágneses mező (a pólusmező és az armatúramező) kölcsönhatása eredményezi. A kialakított nyomaték arányos a mágneses terek főfluxus irányai közötti szög szinuszával. Így állandó nyomaték csak úgy érhető el, ha a két mágneses mező relatív nyugalomban van, vagyis ha együtt forognak. Ezt a nyomatékot elektromágneses nyomatéknak szokás nevezni. A legtöbb motortípus ezen felül reluktancianyomatékot is szolgáltat, amely az állórész és forgórész közti légrés méretének változásából ered. Elektromágneses nyomatékképzéssel egyenletes is, míg reluktancianyomaték-képzéssel csak lüktető nyomaték érhető el [9]. Emiatt a szervomotoroknál minimális reluktancianyomaték-képzésre törekednek.

Fontos megemlíteni, hogy a villamos szög, geometriai szög között eltérés lehetséges. A póluspárszám azt adja meg, hogy a mágneses mező hány teljes hulláma található meg a kerület mentén. A geometriai szög a forgógépen mért fizikai szöget jelenti. A villamos szög pedig azt jelöli, hogy ha a gépet megforgatjuk és mérjük a kapcsain a feszültséget, akkor mekkora forgási szögtartományon alakul ki egy teljes indukált feszültséghullám. A geometriai és a villamos szög között a póluspárszám adja a kapcsolatot. Ha α_v a villamos-, α_g a geometriai szög és p a póluspárok száma, akkor:

$$\alpha_v = p * \alpha_g$$

A későbbiekben, ha külön nem említem, akkor a villamos szög fogalmát értem a szög, és a geometriai szög fogalmát a szög-helyhelyzet kifejezés alatt.

3.3. Állandó mágneses szinkrongépek típusai

Az állandó mágneses szinkrongépeket két típusba lehet sorolni vezérlésük alapján [7]. Az egyik a trapézmezős, a másik a szinuszmezős. Itt a mező kifejezés a mágneses indukció térbeli eloszlására vonatkozik a kerület mentén, a motor légrésében.

A trapézmezős típust szokás kefe nélküli DC motornak is hívni, illetve röviden fogalmazva csak BLDC, az angol „Brushless DC” kifejezés alapján, mert felépítésében olyan, mint egy kifordított kefés DC motor. Ez a kifejezés azonban megtévesztő, ugyanis váltakozó áramú, AC táplálást igényel a motor. Ilyenkor az állórésztekercsek

táplálása négyszöghullámokkal (négyszögjelekkel) történik, ugyanis a mágneses mező is ilyen alakú, így a nyomatéklüktetés csökkenthető. A táplálás folyamata lényegében egy-egy fázis áramának kikapcsolása és ezzel egyidejűleg egy másik fázis áramának bekapcsolása, vagyis a kommutáció itt is megvalósul. Lényegében egy tekercset nézve hol pozitív irányú, hol negatív irányú áram folyik át rajta egymás után (esetleg bizonyos vezérléseknél zérus áram is folyhat). A BLDC motoroknak három előnyük van. Az egyik, hogy a szükséges táplálást egyszerűen meg lehet valósítani mikrokontrollerek segítségével, így a táplálás és vezérlés kisebb költségű lehet. A másik, hogy trapéz alakú mágnesesmező-eloszlást könnyű megvalósítani a forgórészen. A harmadik, hogy nagy, akár 95%-os hatásfok is elérhető használatukkal.

A szinuszmezős típust szokás PMSM-nek is rövidíteni az angol „Permanent Magnet Synchronous Motor" kifejezés alapján. Itt is félrevezető az angol rövidítés, ugyanis a BLDC típus is állandó mágneses szinkronmotor. Az állórésztekercsek táplálása ilyenkor szinusz hullámformával történik, ugyanis a mágneses mező kerületi eloszlása is szinuszos, így a nyomatéklüktetés csökkenthető. Ilyenkor, mint a típus nevéből is adódik, a forgórész mágneses mezejének is szinuszos eloszlásúnak kell lennie. Ezt meg lehet oldani a mágnesek megfelelő elhelyezésével, megvalósítani azonban nehezebb, mint a trapézmezős esetében. Ennek a motortípusnak három nagy előnye van. Az egyik, hogy nagyon jó hatásfokkal bír. A másik, hogy pontosabb pozicionálás valósítható meg, mint a trapézmezős típus esetében. A harmadik pedig, hogy akár hálózati táplálás lehetősége is adott, szintén a mező típusa miatt, valamint ilyenkor a villamos gép kapcsain leadott többfázisú villamos teljesítmény, illetve a villamos gép tengelyén leadott nyomaték (mechanikai teljesítmény) az időben állandó.

Mindkét típus vezérlésénél fontos megemlíteni, hogy csak akkor alakulhat ki egyáltalán forgómozgás, ha ismert a forgórész helyzete. Ugyanis a forgórész mágneses mezeje alapján lehet eldönteni, hogy éppen melyik fázistekercsre milyen irányú áramot kell ráadni, hogy a megfelelő irányú forgatónyomaték ébredjen. Vagyis a vezérlésnél kritikus jelnek számít a szöghelyzet, amely információt vissza kell csatolni a szabályozásban. A szakdolgozat-feladatom nekem is ez volt, a szöghelyzet érzékelése és kiszámítása, vagyis a szabályozási kör egy igen fontos részét valósítottam meg.

A ThyssenKruppnál kizárólag PMSM típusú motorokat alkalmaznak a kormányzásrásegítés céljából. Azonban többféle gyártótól is rendelnek a villamos motorokat, testreszabott kivitelben.

4. Mezőorientált szabályozás

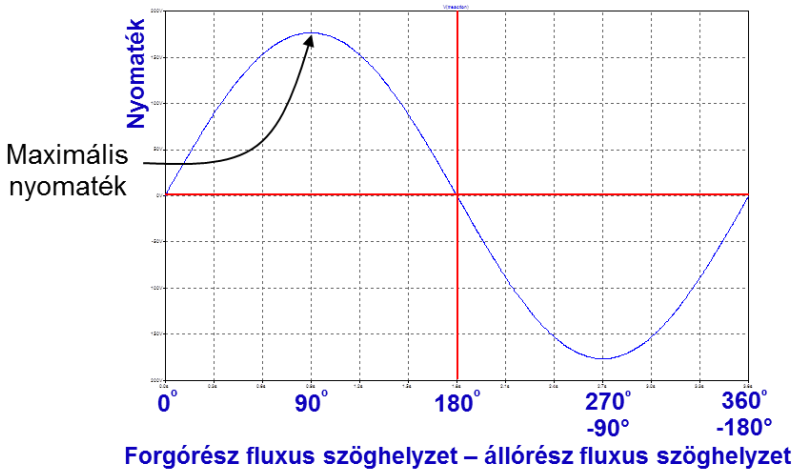
Manapság egy igen közkedvelt és gyakran alkalmazott szabályozási megoldás váltakozó áramú (AC) motorok esetében a mezőorientált szabályozás, röviden FOC, az angol „Field Oriented Control” kifejezés alapján. Ez a megoldás lényegében leírja a teljes szabályozási kör működését, amelynek segítségével a motort irányíthatjuk. A szabályozás alapelve egyszerű, megvalósításához azonban mögöttes matematikai transzformációk (Park- és Clark transzformáció) ismerete is szükséges [7].

Természetesen nem ez az egyetlen szabályozási módszer, amelyet váltakozó áramú (AC) motorok esetén alkalmaznak. Többféle más megoldás is elterjedt a szabályozási kör kialakítására. Ilyen például a Park-vektor moduláció, vagy SVM, az angol „Space Vector Modulation” kifejezés alapján. Erre, illetve egyéb szabályozási megoldások részletesebb tárgyalására, azonban nem szándékozom kitérni, ugyanis a ThyssenKruppnál a mezőorientált szabályozást alkalmazzák, így ebbe a módszerbe tudtam részletesebben betekintést nyerni.

A mezőorientált szabályozás alkalmazásával a villamos motor nyomatékának szabályozása lehetséges. Ha esetleg a motorpozíció, vagy -sebesség alapján kellene szabályozni, akkor ezeket a szabályozási köröket külön kellene kialakítani, hogy ezek a mennyiségek is szabályozottak legyenek. A mezőorientált szabályozás hatalmas előnye ugyanakkor, hogy gyors tranziens lefutású és nagyon pontos szabályozást tesz lehetővé.

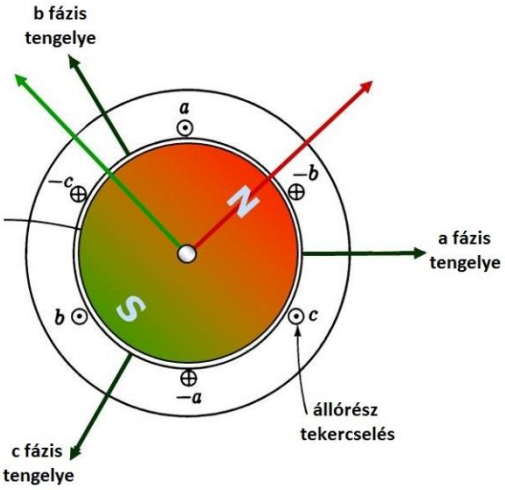
A szabályozás alapötlete az alábbi egyszerű gondolatmenettel foglalható össze. Ha adott egy akár 2 akár 3 fázisú váltakozó áramú motor (pl.: PMSM), akkor létrehozható megfelelő állórésztekercseléssel és -táplálással egy egyenletesen forgó mágneses fluxusvektor, illetve mágneses indukcióvektor. Vagyis ha egy mágnest helyezünk ebbe a mágneses mezőbe (pl.: a forgórész egy állandó mágnes, mint PMSM esetében), akkor az is a mező szögsebességével egyezően forogni fog és követi az állórész mágneses mezejét. Azt részleteztem már, hogy a kialakított nyomaték arányos a mágneses terek főfluxus irányai közötti szög szinuszával, ha csak az elektromágneses nyomatékot vesszük figyelembe. Vagyis ha a forgórész állandó mágnese pontosan egyező irányban áll az állórész mágneses mezejének irányával, akkor a motor zérus nyomatékot képez. Ha pedig a két mágneses mező közti szög éppen 90° akkor a szinuszfüggvény tulajdonságai miatt – miszerint 90°-nál lokális maximuma van a

függvénynek – a motor nyomatéka maximális lesz adott táplálóáram-amplitúdó mellett. Az is könnyen belátható, hogy 270°-os (-90°-os) szög esetében a nyomaték minimális lesz, ugyanakkor abszolút értékben ez maximális nyomatékot jelent. Ebben az esetben is adott táplálóáram-amplitúdó mellett maximális nyomatékú munkapontban van a motor, csupáncsak a másik irányba forog az elrendezés és irányrendszerek választása miatt. Az elektromágneses nyomaték alakulását a mágneses mezők főfluxusai közötti szög függvényében az alábbi ábra szemlélteti.



4-1. ábra Elektromágneses nyomaték alakulása a mágneses mezők közötti szög függvényében [4]

A szabályozás célja az alapelv szerint az, hogy az állórész és forgórész mágneses mezők mindig ±90°-os szögben álljanak egymáshoz képest, attól függően, hogy merre akarjuk forgatni a motort. Mivel a forgórész egy állandó mágnes, emiatt a forgórész szöghelyzetét nem tudjuk szabályozni, az állórész szöghelyzetét azonban igen. Vagyis folyamatosan mérni kell a forgórész mágneses mezejének irányát (a főfluxus irányát). Ezt az irányt szokás d tengelynek is nevezni, az angol „direct-axis” kifejezés alapján. Majd ha ez az információ ismert, akkor ki tudjuk számolni a d tengelyhez képest 90°-kal eltérő – merőleges – irányt. Ezt az irányt szokás q tengelynek is nevezni az angol “quadrature-axis” kifejezés alapján. Végül kiszámoljuk, hogy milyen áramkomponensekre van szükség a tekercsekben, hogy az általuk kialakított mágneses mező iránya a q tengely irányába essen. A jobb átláthatóságot segíti az alábbi ábra, ahol a piros nyíl jelöli a d tengely irányát, a zöld pedig a q tengelyét.



4-2. ábra FOC alapelve [4]

A szabályozás megvalósításához még egy feladatunk van, miszerint folyamatosan frissíteni kell ezeket az értékeket, ugyanis a motor forogni fog, emiatt folyamatosan mérni kell a d tengely szögét, utána folyamatosan ki kell számítani a 90°-os eltolást, és végül az alapján a fázisáramokat.

A mezőorientált szabályozás során mindig 90°-os szöghelyzetkülönbséget akarunk kialakítani a két mágneses tér iránya között. Ha esetleg növelni vagy csökkenteni szeretnénk a kialakítandó nyomatékot, akkor az állórészfluxus nagyságát kell változtatni, amit a fázisáramok amplitúdójának változtatásával tehetjük meg

4.1. Mezőorientált szabályozás folyamata

A szabályozás elméleti megfontolásai után, most ebben a fejezetben végigkövetem a szabályozási kör működését egy háromfázisú PMSM motor vezérlés példáján [7].

A legelején említettem, hogy ez a szabályozási kör a motor nyomatékát tudja szabályozni. Egy PMSM típusú motoron a nyomatékot háromféle módon változtathatjuk. Az egyik mód, a forgórészfluxus nagyságának változtatása, ez azonban gyakran nehezen megvalósítható, emiatt általában konstans értékűnek veszik, mint ahogy ebben a példában is annak veszem. A másik mód az állórészfluxus szögének változtatása és vele együtt az állórész tekercseire adott áram térvektor szögének a változtatása. A mezőorientált szabályozás alapelve azonban az, hogy ezt a szöget konstans ±90°-os értéken tartjuk, így ezt sem változtathatjuk. A harmadik mód az állórész által keltett fluxus mértékének a változtatása, amit azonban az állórész

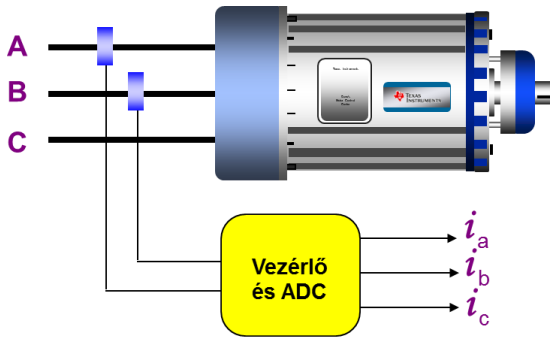
tekercseire adott áram térvektor nagyságával befolyásolni tudunk. Minél nagyobbra választjuk a térvektor amplitúdóját annál nagyobb nyomaték ébred a motorban. A motor nyomatéki egyenlete az alábbi, megfelelően beállított szabályozás esetén [10].

$$M = \frac{3}{2} * p * (\Phi_{dr} * I_{qs})$$

Ahol M a nyomaték nagysága, p a póluspár-szám, Φ_{dr} a forgórész fluxusának d irányú komponens nagysága (ami megegyezik a teljes forgórészfluxus amplitúdóval, ha a szórástól eltekintünk) és I_{qs} az állórész áram térvektor q irányú komponens nagysága (ami megegyezik a teljes áramvektor nagyságával, ha jól működik a szabályozás).

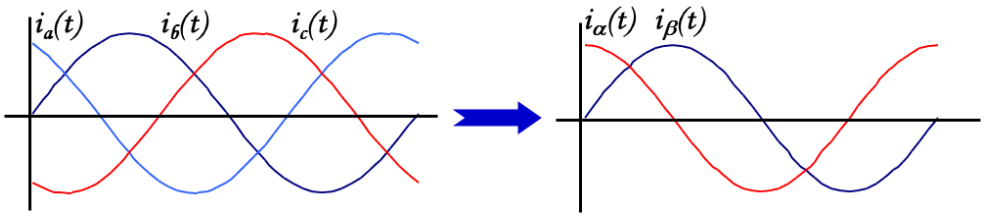
A szabályozás algoritmusa négy lépésből áll össze. Ezeket a lépéseket megfelelő időközönként le kell futtatni. Elég gyakran, hogy a motor mágneses tere ne változzon meg túlságosan a forgás miatt, viszont elég ritkán, hogy a lépések biztosan lefuthassanak. A szabályozási kör robusztussága miatt működés közben tipikusan 5-8°-os szöghelyzet-bizonytalanságot képes tolerálni. Szerencsére a motor mechanikai korlátai miatt a fordulatszám nem mehet egy előre meghatározott érték fölé, így megfelelő idő áll a rendelkezésre mire ez a szögelfordulás bekövetkezik, és a számítások elvégezhetők. Ez a tolerancia a szöghelyzet-érzékelés pontosságát is megszabja, és ez alapján ilyen célú felhasználásra megfelelő szögszenzor választható.

Első lépésként meg kell mérnünk a motor kapcsain befolyó áramokat. Elméletileg elegendő a három fázisáramból csak kettőt megmérni, mert a harmadik az első kettő alapján számítható, ugyanis amennyi áram befolyik a mért két fázis vezetékén ugyanannyi fog kifele is folyni a harmadik vezetéken a csomóponti törvény miatt. A valóságban azonban gyakran mérik mind a három fázisáramot, így lehetőség van hibajelzés kialakítására is. A három fázisérték alapján kiszámítható az aktuálisan az állórész tekercsein folyó áram térvektora.



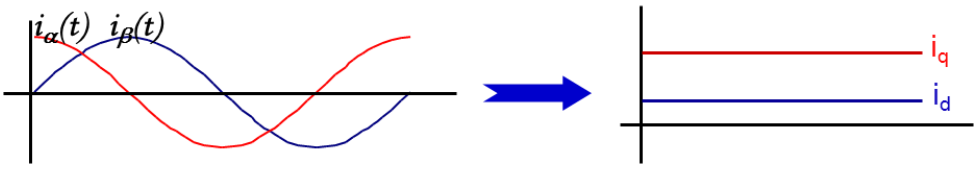
4-3. ábra FOC első lépése [4]

Második lépésként össze kell hasonlítanunk a mért áram térvektorát a kívánt áram térvektorával és kialakítani a hibajelet. Ehhez a lépéshez legelőször meg kell mérni a forgórész – pontosabban a forgórész mágneses mezejének, főfluxusának – szöghelyzetét és irányát. A szakdolgozat feladata során ezzel a méréssel foglalkoztam. A mérés alapja a GMR szenzor, amelynek működését a későbbiekben ismertetem. Ha ismert a forgórész szöghelyzete, akkor vele együtt ismert a d tengely is, ami alapján kiszámítható rá merőleges q tengely iránya is. Ezek után kapnak szerepet a matematikai transzformációk. Ugyanis lehetne azt a megoldást is választani, hogy a három fázisáram alapján három különálló szabályozó egységgel megvalósítjuk, hogy az áram térvektor kövesse q tengelyt, ez azonban feleslegesen sok számítással járna és a szabályozási algoritmus is bonyolult lenne. Ehelyett első műveletként az aktuális áram térvektor koordinátáit ($i_A(t)$, $i_B(t)$ és $i_C(t)$) felírjuk a 120°-os, három koordinátatengelyű (A,B és C), álló, vonatkoztatási rendszer helyett egy kéttengelyes (α és β), álló, derékszögű koordinátarendszerben. Így megkapjuk az aktuális $i_\alpha(t)$ és $i_\beta(t)$ értékeket. Ez a lépés a Clark transzformáció [11], melynek segítségével egyértelmű hozzárendelés valósítható meg a két koordinátarendszer között.



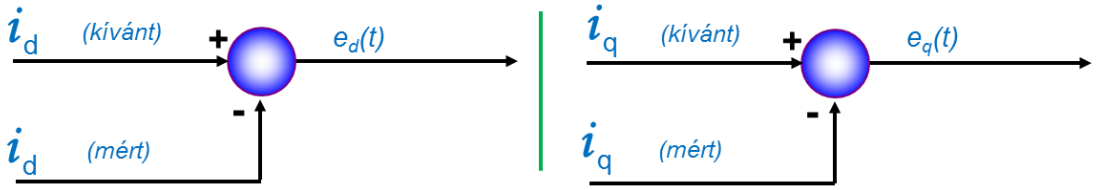
4-4. ábra FOC második lépés - Clark transzformáció [4]

E transzformáció segítségével tulajdonképpen az eredetileg háromfázisú motorból a kétfázisú motort csináltunk, ahol az állórésztekercsek egymáshoz képest 90°-ban állnak. Ezzel máris eliminátunk egy szabályozó egységet, ugyanis innentől kezdve már csak két értékre kell a szabályzó egységet készítenünk. De még az új álló derékszögű koordinátarendszerben az aktuális áram térvektor koordinátái – $i_\alpha(t)$ és $i_\beta(t)$) – időben koszinusz- és szinuszjelként változnak, amelyek szabályozása nehézkes lehet, főleg magasabb fordulatszámok esetén. Emiatt következő számításként kiszámoljuk az aktuális áram térvektor koordinátáit nem az álló, hanem a forgórésszel együtt forgó, kéttengelyű, derékszögű, (d és q) koordinátarendszerben. Így megkapjuk az aktuális $i_d(t)$ és $i_q(t)$ értékeket Ez a lépés a Park transzformáció, melynek segítségével egyértelműen áttérhetünk a két koordinátarendszer között.



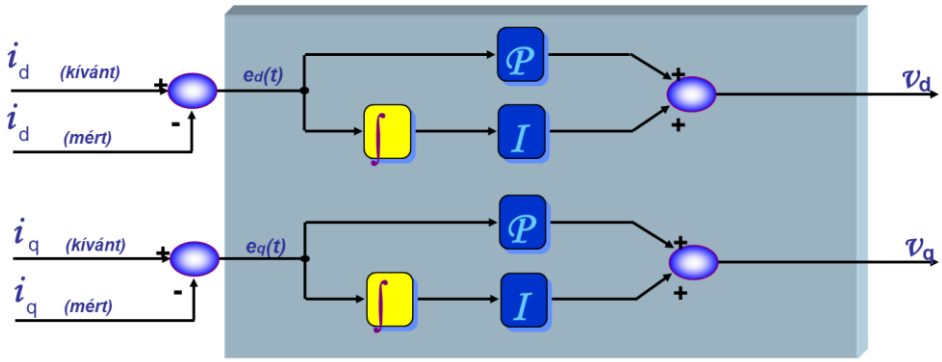
4-5. ábra FOC második lépés - Park transzformáció [4]

Ennek hatására az áramvektor koordinátái időben konstans jelként „változnak”. Ez utóbbi két érték segítségével már könnyen kialakítható a hibajel. A kívánt $i_d(t)$ és $i_q(t)$ értékek kialakításakor azonban külön kezeljük a két komponenst. Normális esetben a forgórész állandó mágnese elég fluxust biztosít a nyomatékképzéshez, így az $i_d(t)$ kívánt értékét zérusnak vesszük. Ezzel érjük el, hogy az állórész mágneses mezeje 90°-os szöggel legyen elforgatva a forgórész mágneses mezejéhez képest. Ha azonban szükséges, az $i_d(t)$ segítségével csökkenthető e szög, és például így megvalósítható a mezőgyengítés. Az $i_q(t)$ komponens nagyságával lehet beállítani a motor nyomatékát. Minél nagyobbra vesszük az $i_q(t)$ a komponenst annál nagyobb nyomaték keletkezik a motorban, illetve ha kisebbre, akkor kevesebb. Így a nyomatékigény alapján kiszámítható a kívánt értéke. Mind a két komponensnél kivonva a kívánt értékből az aktuális, mért értéket megkapjuk mindkét jelre a hibakomponenst, amelyeket e_d és e_q szimbólumokkal jelölök.



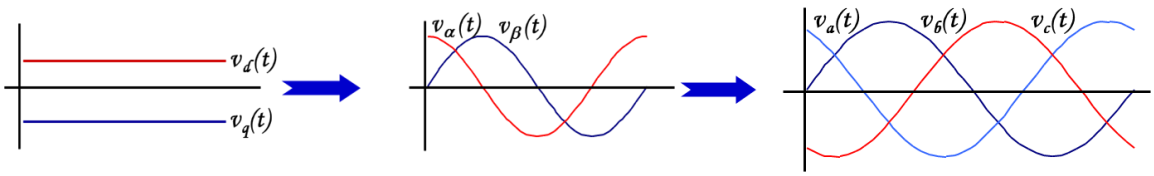
4-6. ábra FOC második lépés - Hibajelek előállítása [4]

Harmadik lépésként a hibakomponensek megfelelő erősítésével kialakítjuk a beavatkozó jeleket. Ez egy-egy PI szabályozó segítségével megoldható feladat mind a két (d és q) komponensre is. A beavatkozó jelek jelen esetben feszültségjelek lesznek, amelyek az állórésszel kialakítani kívánt feszültség térvektor koordinátái a d és q koordinátarendszerben. Jelük $u_d(t)$ és $u_q(t)$.



4-7. ábra FOC harmadik lépés [4]

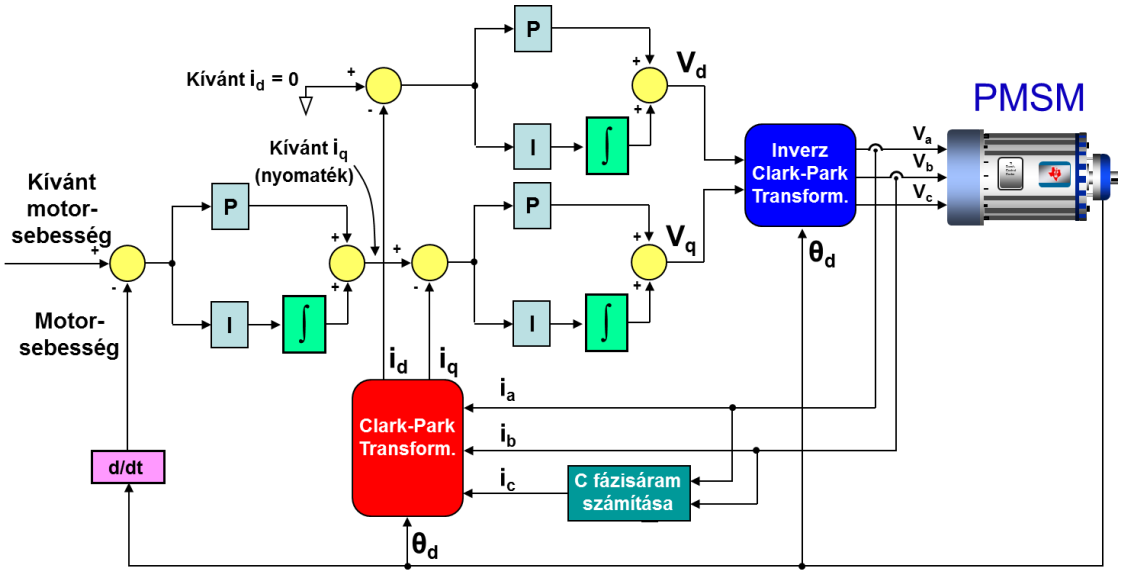
Negyedik lépésként megfelelően modulálni kell a beavatkozó feszültségjeleket és rákapcsolni a motor állórész tekercseire. Ehhez először vissza kell számolnunk a d és q koordinátarendszerben adott $u_d(t)$ és $u_q(t)$ értékeket álló, vagyis az α és β koordinátarendszerbe. Ez az inverz-Park transzformáció segítségével megoldható, és így megkaphatjuk $u_\alpha(t)$ és $u_\beta(t)$ jeleket. Ha kétfázisú motorunk lenne, akkor készen is lennénk, mivel azonban háromfázisú PMSM motort alkalmazunk emiatt még a két koordinátatengelyű, álló, α és β viszonyítási rendszerből vissza kell térnünk a három koordinátatengelyű, álló, A, B és C viszonyítási rendszerbe. Ez az inverz-Clark transzformáció segítségével megoldható, és így megkapjuk $u_A(t)$, $u_B(t)$ és $u_C(t)$ feszültség időfüggvényeket, amelyeket a szabályozás szerint a tekercsek pólusaira kell adnunk, hogy az így létrehozott mágneses mező pontosan derékszögben álljon a forgórész mágneses mezejével, ezáltal biztosítva a megfelelő nyomatékot.



4-8. ábra FOC negyedik lépés - Inverz transzformációk [4]

Valóságban a motor tekercseit tranzisztoros inverterkapcsolások segítségével hajtjuk meg, így nem tudunk tisztán szinuszos jelet generálni, pulzusszélesség moduláció (PMW) segítségével azonban igen jól közelíthető a szinuszjel a motor tekercsein.

A lépések leírása során ábrákkal még a matematikai átalakításokat szemléltettem a jobb megértés érdekében. Azonban a lentebbi, 4-9. ábrán látható az egész szabályozási kör teljes egészében. Az ábrán ugyanezek a lépések láthatók, mint amiket fentebb leírtam, csak most a szabályozási kör szempontjából végigkövetve a jelek útvonalait.



4-9. ábra FOC - A teljes szabályozási kör [4]

A 4-9. ábra alsó és jobboldali részén jól látható az első lépés, vagyis a fázisáramok visszamérése. Legalul és jobboldalt látható a pozíció visszamérése, ami a második lépés eleje. Alul középen látható a Clark és Park transzformációt megvalósító modul piros színnel, amely a második lépés része. Közvetlenül a piros modul felett látható a hibajel kialakítása, amely a második lépés végső szakasza. A kivonóktól jobbra egymás alatt látható a két PI szabályozó modul, amely a harmadik lépés. Majd kék színben az inverz-Park és inverz-Clark transzformációs modul, amely a negyedik lépés része.

Felhívom a figyelmet az ábra bal oldalára, ahol látható, hogy ez a szabályozási kör egy kívánt motorsebesség-jelet kap bemenetként. Majd közvetlenül a bemenő jel után egy PI szabályzó alakítja ki a kívánt i_q áramkomponens értékét. Vagyis jelen szabályozási kör nem a hagyományos mezőorientált szabályozást valósítja meg, hanem a motorsebesség alapján szabályoz. Ezt már a fejezet legelején említettem, hogy a mezőorientált szabályozás a motor nyomatékjelét tudja szabályozni, egyéb szabályozás kialakításához külön egységek kellenek a szabályozási kör kialakítására. Az ábrán látható módon azonban, a hagyományos mezőorientált szabályozás kiegészítésével, ez a feladat is könnyen megvalósítható. Ezt a változtatást leszámítva a 4-9. ábra tökéletesen összefoglalja a mezőorientált szabályozás minden lépését.

5. GMR érzékelés

A GMR kifejezés az angol Giant Magnetoresistance kifejezésből ered, amelyet magyarul óriás mágneses ellenállásnak szokás fordítani. A GMR érzékelés alapelvén a mágneses mező nagyságát lehet megmérni széles tartományon belül. Az ilyen elven működő szenzorok direktben a mágneses mező nagyságát mérik, nem pedig annak a változását, így alkalmasak akár konstans mező érzékelésére is.

Az érzékelés alapelve, hogy bizonyos mágneses anyagok elektromos ellenállása függ a mágnesezettségtől. GMR esetében a G betű, az óriás szó arra utal, hogy a villamos ellenállás nagymértékben változik a mágneses mező függvényében. Vagyis az ilyen szenzorok érzékenyek a mágneses tér kis változásaira is, így kiválóan alkalmasak pozíció és szöghelyzet mérésére lineáris és forgó rendszerekben is [12].

5.1. GMR és AMR érzékelés

A MR érzékelés elvét először Lord Kelvin fedezte fel 1856-ban. A vas vizsgálata során ő fedezte fel, az elektromos ellenállása más, ha az áram ugyanabban az irányban folyik a fémben, mint a mágneses erővonalak iránya, mintha rá merőlegesen. Ezt a hatás később AMR-ként, vagyis anisotropic magnetoresistance kifejezésként vált közismertté.

Az AMR érzékelés alapelve, hogy egy anyag elektromos ellenállása függ az anyagban folyó áram, és az anyagra ható mágneses mező iránya közti szögtől. A változás mechanizmusa a mágneses tér és a vezető elektronok spinjének kölcsönhatásán alapszik, ugyanis az elektronok bizonyos a mágneses térrel bezárt szögek esetén jobban szóródnak, míg más esetben könnyebben haladnak előre. Végeredményként elmondható, hogy az elektromos ellenállás maximális, ha a mágneses mező, és az áram iránya egymással párhuzamos, akár azonos akár ellentétes irányba mutatnak. Míg az ellenállás minimális, ha a kettő egymással 90°-os szöget zár be [13].

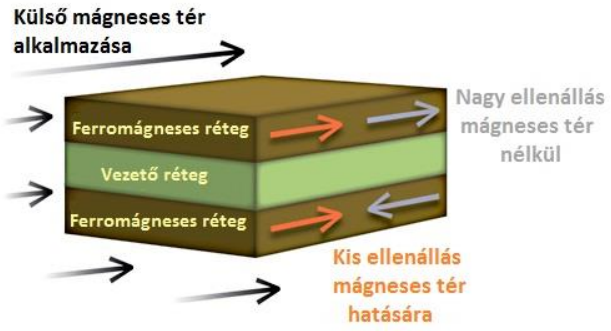
A GMR érzékelés is ugyanazt a hatást használja fel, mint az AMR, speciális kialakítás segítségével azonban az elektromosellenállás-változás sokkal nagyobb lesz GMR érzékelőkben, mint a hagyományos AMR érzékelőkben. Ezenkívül még egy fontos különbség, hogy az AMR szenzorok nem képesek különbséget tenni a kétféle

mágneses pólus között. Vagyis ha folyamatosan forgatjuk az átfolyó áram irányát a mágneses mezőhöz képest, és így vesszük fel az elektromosellenállás-változás függvényét, akkor egyező irány és ellentétes irány esetében ugyanazt az értéket kapjuk, vagyis a karakterisztika 180°-onként ismétlődik. GMR esetében a karakterisztika csak 360°-onként ismétlődik, így a teljes forgás során meghatározható az aktuális pozíció.

5.2. GMR struktúra

A GMR jelenséget először 1988-ban fedezték fel, amikor is nanométer vastagságú mágneses rétegek közé egyenlő vastagságú, ugyanúgy nanométer vastagságú vezető rétegeket helyeztek. Ilyenkor óriási változás következett be az elektromos ellenállásban, ha mágneses térbe helyezték a szerkezetet [14].

Egy GMR érzékelő leegyszerűsített struktúrájában két ferromágneses anyagréteg (pl.: vas, nikkel) közrefog egy nem mágneses, de vezető anyagréteget (pl.: réz). A hatás kialakítása szempontjából fontos, hogy az összes réteg nagyságrendileg nanométer vastagságú legyen. Ilyen kis vastagságú rétegek esetében a méretek gyakran összemérhetőek az atomi méretekkel, emiatt a vastagságokat gyakran angströmben (Å) mérik. 1 Å = 10^{-10}m=0.1nm.



5-1. ábra GMR szenzor szerkezete [5]

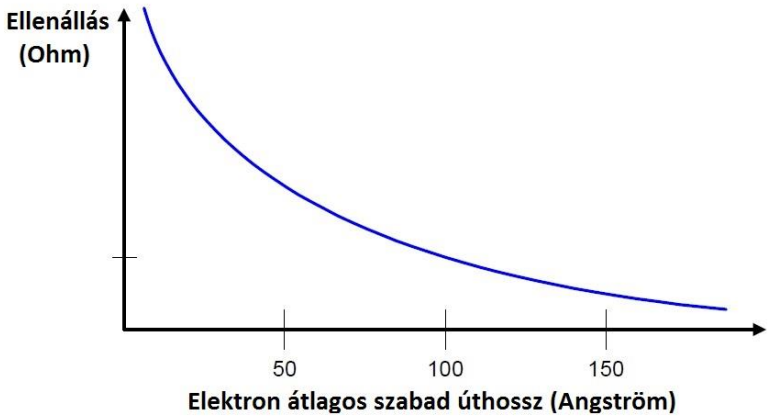
A két mágneses réteg anti-ferromágneses csatolásban van egymással, ha nem alkalmazunk külső mágneses mezőt, vagyis a rétegek mágnesezettsége ellentétes irányú. Ilyenkor az elektromos ellenállás a legnagyobb, mint ahogy azt az 5-1. ábra is mutatja. Azonban ha megfelelően nagy külső mezőt alkalmazunk, akkor az megszünteti az anti-ferromágneses csatolást, és a két mágneses rétegben egyező irányú lesz a mágnesezettség. Ilyenkor az elektromos ellenállás kicsi lesz. Vagyis az ellenállás változás miatt a szerkezet képes a mágneses tér érzékelésére.

5.3. GMR struktúra működési elve

Az előző alfejezetben már említettem, hogy egy GMR szenzor struktúrája a kulcs ahhoz, hogy a GMR érzékelőben kis mágnesesmező-változás hatására is nagy elektromosellenállás-változás jöjjön létre.

Az óriási elektromosellenállás-változás részben annak is köszönhető, hogy egy vezetőréteg ellenállása függ a réteg vastagságától. Ha megközelítjük az atomi méretet a réteg vastagságában, akkor minél vékonyabb a vezető réteg, annál nagyobb az elektromos ellenállása is. Ennek fő oka a vezetési elektronok haladásában, vagy a rácsszerkezetben lévő atommal való ütközés miatti szóródásban keresendő [14].

Homogén anyagokban (pl. színfémben) a vezetési elektronok viszonylag hosszú utat képesek megtenni egyenesen, mielőtt egy másik atomi részecskével ütköznének, és emiatt irányváltozásra kényszerülnének, vagyis szóródnának. Ez a hossz leginkább az anyagi minőségtől függ. Azt az átlagos úthosszt, amelyet az elektronok átlagban megtesznek, mielőtt a szóródás bekövetkezne, átlagos szabad úthossznak nevezzük. Ha azonban nagyon vékony az anyag rétege – vékonyabb, mint az átlagos szabad úthossz – akkor az elektronok nem képesek ugyanakkora távolságot egyenesen haladni, mint azt a nagy kiterjedésű homogén anyag esetében tennék. Sokkal valószínűbb, hogy előbb elérik a réteg határát, minthogy egy másik atomi részecskével ütköznének, és a határátlépés miatt kényszerülnek irányváltoztatásra, vagyis a határon szóródnak. Ennek következtében a vékony réteg miatt csökken az elektronok átlagos szabad úthossza. Így nő a réteg elektromos ellenállása, hiszen az elektronok nehezebben képesek egyenesen haladni az ilyen vékony rétegben. Az 5-2. ábra mutatja az anyagréteg elektromos ellenállását a vastagság függvényében.

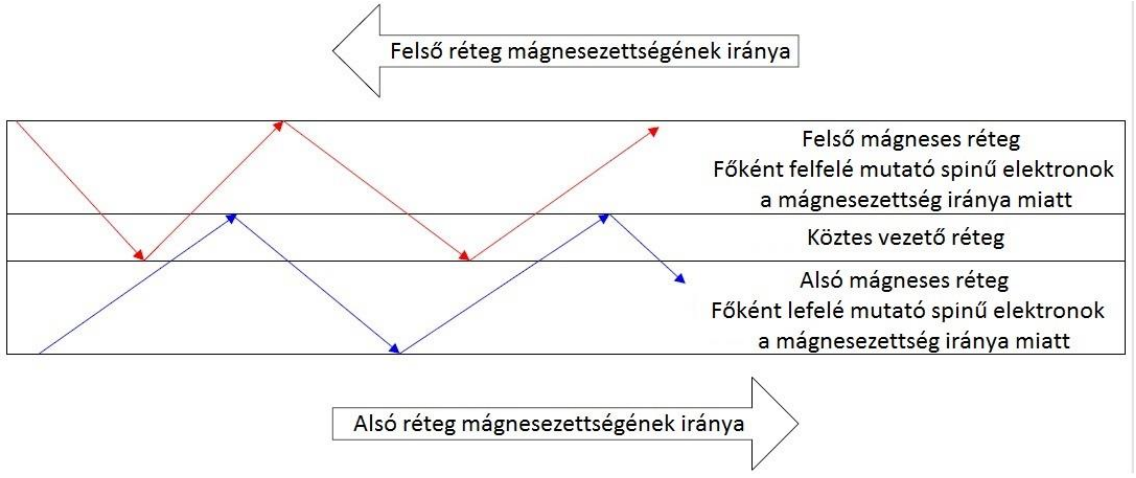


5-2. ábra Mágneses anyagrétegek ellenállása a vastagság függvényében [6]

Ezen hatás lehető legjobb kihasználása miatt szükséges, hogy a mágneses és a vezető anyagrétegek nanométer vastagságúak legyenek a GMR szenzorban.

A megfelelően nagy ellenállásváltozás eléréséhez azonban még egy fizikai tulajdonságot is kihasznál a GMR elvű érzékelő. Ez pedig az elektronok spinjének mágneses mező általi „modulációja". Ferromágneses anyagokban a vezetési elektronok spinjét felfelé mutatónak tekintjük, ha spin iránya egyezik a ferromágneses anyag mágneses momentumvektorának irányával. Míg a spint lefelé mutatónak nevezzük, ha az iránya pont ellentétes a mágneses momentumvektor irányával. Egy nem mágneses anyagban (pl.: a GMR szenzor középső vezető rétegében) kiegyenlített számban vannak a felfelé és lefelé irányban mutató spinű elektronok. A kvantummechanika szerint azonban annak a valószínűsége, hogy egy elektron egyenesen halad egy mágneses anyagban, vagy szóródik benne, az az elektron spinjének irányától függ. Tehát egy mágneses anyagban a felfelé mutató spinű elektronok tovább képesek terjedni irányváltoztatás nélkül [12].

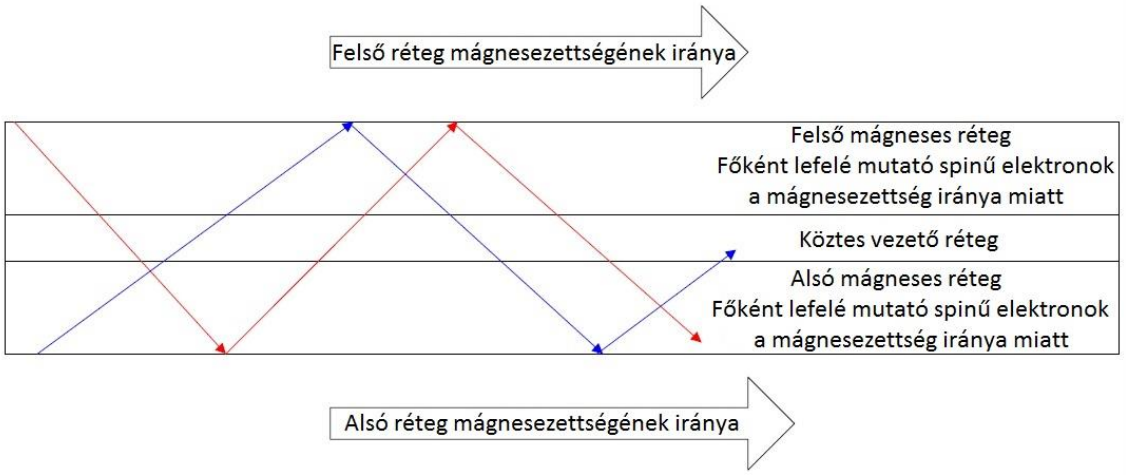
A GMR szenzorban ezt a hatást is ki lehet használni a két mágneses réteg anti-ferromágneses csatolása miatt. Ugyanis amikor nincsen külső mágneses tér, akkor a két mágneses rétegben a mágnesezettség iránya ellentétes. Ilyen esetben az elektronok nagy valószínűséggel a két mágneses réteg határán szóródni fognak. Ugyanis ha a felső rétegben egy felfelé mutató spinű elektron, áthalad a vezető rétegen és megpróbál belépni az alsó mágneses rétegbe, akkor ott már lefelé mutató spinűnek vehetjük, mert az alsó réteg mágneses indukcióvektora pont ellentétes a felső rétegével, így tehát az alsó rétegben nem tud egyenesen haladni majd, hanem szóródni fog. Ugyanez igaz fordított esetben is, ha az alsó réteg elektronja a vezető rétegen át eléri a felső réteget.



5-3. ábra Vezető elektronok haladása a spinjük szerint anti-ferromágneses csatolás esetén [6]

Ezt szemlélteti az 5-3. ábra, ahol a felső mágneses réteg mágneses indukcióvektorát tekintjük a vonatkoztatása alapnak a spin irányának meghatározásakor. Vagyis ilyen esetben az elektronok átlagos szabad úthossza viszonylag kicsinek mondható, tehát a szenzor elektromos ellenállása nagy.

Ha azonban megfelelően nagy külső mágneses térbe helyezzük a rétegeket, ahol a mágneses momentumvektor iránya ellentétes a szélső réteg mágnesezettségének irányával, akkor az legyőzi az anti-ferromágneses csatolást, és az alsó és felső mágneses réteg mágnesezettsége azonos irányú lesz. Ilyenkor azonban mind a két rétegben az azonos spinű elektronok tudnak csak terjedni, viszont azok mindkét rétegen át könnyen terjedhetnek, az azonos mágneses momentumvektor-irány miatt.



5-4. ábra Vezető elektronok haladása a spinjük szerint külső mágneses tér esetén [6]

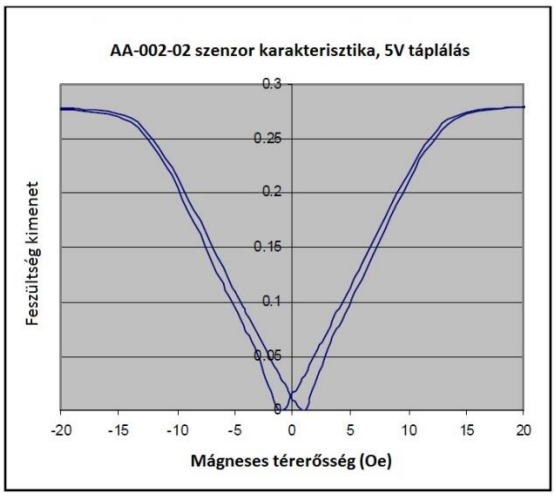
Ezt szemlélteti az 5-4. ábra. Látható, hogy ilyenkor megnőtt az elektronok átlagos szabad úthossza az előző esethez képest. Tehát ilyenkor a szenzor ellenállása kisebb lesz.

GMR szenzorok esetében a szenzor érzékenységét általában százalékban adják meg, vagyis hány százaléknyit képes csökkenni a háromrétegű struktúra ellenállása ahhoz képest, ha nem hat a szenzorra külső mágneses mező. Az érték kiszámítását az ellenállásváltozás és a minimális ellenállásérték aránya alapján szokás százalékban kifejezni.

5.4. GMR szenzorok felépítése

Általában a mágneses terek érzékelésére gyártott GMR szenzorokban 4 darab GMR struktúrát alkalmaznak Wheatstone-híd kapcsolásban. A kapcsolás technikája alapján két konstrukciót lehet megkülönböztetni [14].

Az első a magnetométernél alkalmazott kapcsolás. Ez a szenzor típus az alap a mágneses mező érzékeléséhez. Ilyenkor kettő GMR ellenállást mágnesesen árnyékolnak a hídban, a szubsztrátra felvitt mágneses anyagok segítségével, amelyek elnyelik a külső mágneses tér erővonalait. Ennek alkalmazásával a mágneses anyagok hiszterézishibái csökkenthetőek. Így csak kettő, a híd két ellentétes ágán lévő GMR ellenállás érzékeli a külső mágneses mezőt. Ezek ellenállása csökken külső mágneses tér hatására, míg a két árnyékolté változatlan marad, így a Wheatstone-híd kimenetén mérhető feszültségkülönbség lesz. Ilyen elrendezés esetén lehetőség van a szubsztrátra olyan anyagokat is felvinni, amelyek a mágneses fluxus irányát képesek befolyásolni, és így koncentrálni a mágneses térerősség erővonalait a szabadon lévő GMR ellenállásoknál. Így akár kétszerestől százszoros mértékig is megnövelhető a GMR ellenállást érő mágneses térerősség nagysága, aminek hatására jobban változik a híd kimeneti feszültsége. Egy ilyen szenzor tipikus karakterisztikáját mutatja az 5-5. ábra.



5-5. ábra GMR szenzor tipikus karakterisztika [6]

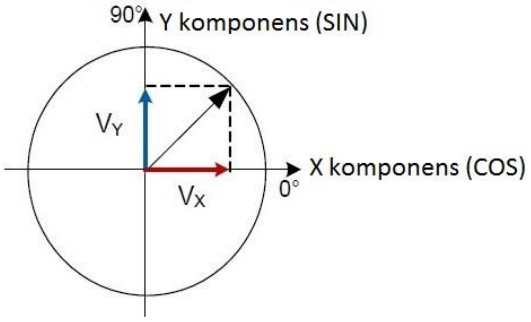
A második a gradiométereknél alkalmazott kapcsolás. Ilyenkor a mágneses tér gradiensével arányos feszültség keletkezik a Wheatstone-híd kimenetén. Ennek a felépítése tulajdonképpel megegyezik az előző típussal, annyi különbséggel, hogy nem alkalmaznak árnyékolást, vagyis a híd mind a négy ellenállása változik a mágneses tér hatására.

5.5. Forgó mágneses mező mérése

Az előző fejezetekben bemutattam, hogy egy GMR struktúra hogyan képes változtatni az ellenállását a mágneses mező nagyságának függvényében. Az eddigiek során azonban mindig azt feltételeztem, hogy a külső mágneses tér iránya mindig megfelelő, és az anti-ferromágneses csatolást próbálja megszüntetni.

Ebből a tényből látszik, hogy egy GMR ellenállás csak a struktúra vonala mentén képes érzékelni a mágneses tér nagyságát, és csak az adott irányú mágneses tér hatására következik be az ellenállás változása.

Munkám során azonban a forgórész szöghelyzetét kellett megmérni. Magát a szöghelyzetet nem a forgórész mágneses mezejének alapján kell mérni, hanem a forgórész tengelyének végén direkt ebből a célból található egy igen erős mágnes, amelynek helyzetét lehet érzékelni a GMR szenzorral, így valósítva meg a mérést. Ahogy a forgórész elfordul, az azonos tengelyen lévő mágnes és annak mágneses tere is forgómozgást végez, vagyis a mágneses tér vektora mindig azonos nagyságú, iránya azonban folyamatosan változik. A GMR struktúra mindig csak a tengelye mentén képes a mágneses térvektor nagyságát érzékelni. Ha végigkövetjük a mágneses térvektor egy fordulatát, és nagyságát levetítjük két egymással derékszöget bezáró tengelyre, amelyek síkjában az ellenállások találhatóak, akkor az egyik tengelyen koszinusz, míg a másik tengelyen szinusz hullámformát kapunk. Ezt használják ki a szenzorok is, és ezt szemlélteti az 5-6. ábra is.



5-6. ábra Körforgó vektor vetületei

6. Analóg szögszenzor

Az analóg szögszenzort, mint minden más alkatrészt, amelyeket a szakdolgozat készítése során felhasználtam a ThyssenKrupp Presta Hungary Kft. biztosította. A felhasznált alkatrészek pontos típusai nem publikusak, azonban a szakdolgozat szempontjából lényeges adataikat bemutatom.

A szenzor adatlapjából is kiolvasható, hogy a szöghelyzet érzékelésére tervezték kifejezetten biztonságkritikus autóipari rendszerekhez. A szenzor a teljes 360°-os szögtartományban képes nagy, tipikusan néhány fokos pontossággal érzékelni a szöghelyzetet, valamint kis késleltetési időkkel rendelkezik, így remekül helytáll olyan alkalmazásokban, amelyek gyors és pontos beavatkozást igényelnek. Az elektromos kormányzás során ez a két tulajdonság kiemelt szerepű, így a szakdolgozat feladatához, illetve az esetleges későbbi gyártáshoz is kiválóan alkalmas.

Az alkalmazott szögszenzor teljesen redundáns érzékelő, amely analóg kimeneti értékekkel rendelkezik. A tokozáson belül található egy GMR szenzor, amellyel teljes 360°-ban lehet érzékelni a szögpozíciót, valamint egy AMR szenzor is, amely egyrészt a biztonságkritikus környezett miatt szükséges, másrészt ennek segítségével még pontosabb eredményt lehet kialakítani. A szenzor a szilíciumszeletek síkjában képes érzékelni a mágneses teret, és annak X és Y irányú komponensét. Forgó mágneses mező esetén ezeket szokás koszinusz és szinusz szögkomponenseknek nevezni. Ezt a két komponenst méri mind a két MR szenzor.

Mind a két szenzor, időben körforgó mágneses mező hatására, kimenetként analóg szinuszos jeleket szolgáltat, amelyekkel jellemezhető a mágneses tér iránya. AMR szenzor esetében 0°-180° között, míg GMR szenzor esetében 0°-360° között adható meg egyértelműen a mágneses tér iránya a kimeneti jelek alapján. A kimeneti jelek függetlenek a mágneses mező nagyságától, vagyis csupán a mágneses tér iránya határozható meg az értékekből. A kimeneti jelek analóg feszültségjelek, amelyek föld és tápfeszültség közötti értékeket vehetnek fel. Így ha a szenzor után következő analóg-digitális konvertert ugyanerre a földre és tápfeszültségre kötjük, akkor a szenzor kimenő szinuszjeleit, megfelelő mintavételezés esetén, könnyen digitalizálhatjuk. Megemlítendő, hogy a szenzor kimeneti jelei alapján aszimmetrikus, és differenciális jelátvitel is megvalósítható, így többféle alkalmazásban is használható.

Az analóg szögszenzor a mezőorientált szabályozásnál a motor aktuális pozíciójának visszamérése használható fel, a forgórész tengelyének végére szerelt mágnes és mágneses mezejének alapján.

A szakdolgozat feladatának megvalósítása során csupán az érzékelő GMR részét használtam ki, ugyanis a vizsgálódás célja főként a feldolgozási algoritmus megvalósíthatósága FPGA áramkörön. Későbbiekben azonban érdemes az AMR részt is használni, ugyanis a két szenzor értékének összehasonlításával tovább növelhető a kapott szöghelyzet pontossága, valamint a szenzor megbízhatósága is növekszik, ami fontos szempont, ugyanis a kormányzásrásegítés biztonságkritikus funkció. A szakdolgozat végén, a feldolgozás redundánssá tételének tárgyalásánál részletesebben is kitérek az AMR érzékelés kihasználására is.

6.1. Analóg szögszenzor kialakítása

Mint már említettem, az előző fejezetben az alkalmazott szögszenzorban egy tokozáson belül található egy AMR és egy GMR érzékelő is.

Az integrált áramkör használata szempontjából a két érzékelő teljesen különválik, olyannyira, hogy a két érzékelőnek külön kell szolgáltatni a tápfeszültséget is, vagyis lehetséges az is, hogy az egyik érzékelő teljesen ki van kapcsolva. Az áramköri kialakítás szempontjából is teljesen különválik a két érzékelő. A közös tokban két külön szilíciumszeleten valósították meg a GMR és az AMR szenzort, úgy, hogy a GMR szelete van felül, közvetlenül alatta kap helyet az AMR szelete. A két érzékelő egymáshoz képest megfelelően elforgatott helyzetben van, hogy a két szenzor által szolgáltatott jelek koordinátarendszerei egybeessenek. Így az elhelyezés garantálja a minimális eltérést a két szenzor által mért szöghelyzetértékek között.

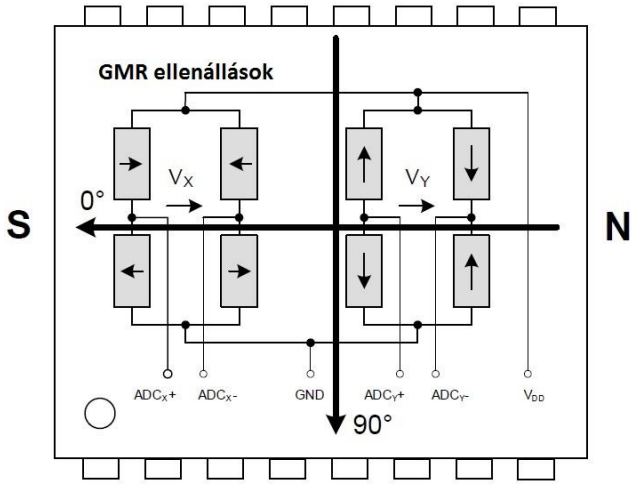
A szeletek kialakításában még függőleges tagolást alkalmaznak, vagyis az adott chipszelet felső rétegében találhatók a mágneses teret érzékelő ellenállások, és alattuk a többi analóg áramköri elem. Így biztosított az analóg jelek megfelelő védelme.

Mind az AMR, mind pedig a GMR érzékelő négy darab különálló, a mágneses terek érzékelő ellenállásból áll össze Wheatstone-híd kialakításban. Mind a két érzékelő méri a mágneses tér X irányú, vagy más néven koszinusz komponensét, valamint az Y irányú, szinusz komponensét is. A teljes hídkapcsolás előnye, hogy az ellenállás-

változás miatti jelkülönbségeket maga a híd kétszeresére erősíti, és a híd kimenete nem függ a hőmérséklettől.

6.2. GMR szenzor

A GMR érzékelőben alkalmazott hídstruktúra kimeneti jelértékei egyértelműek a mágneses tér 180°-os tartományig, vagyis az analóg kimeneti jelek szintjei és a mágneses tér szöghelyzete között egyértelmű hozzárendelés tehető e tartományon belül. Két hídstruktúrát alkalmazva és azokat egymáshoz képest ortogonálisan elhelyezve azonban már a teljes 360°-ban egyértelműek lesznek a kimenetek értékei. Ez az elhelyezés látható a 6-1. ábrán, ahol az ellenállásokon lévő nyilak mutatják, hogy az adott ellenállás melyik irányú mágnesestér-komponens hatására csökkenti az ellenállását.

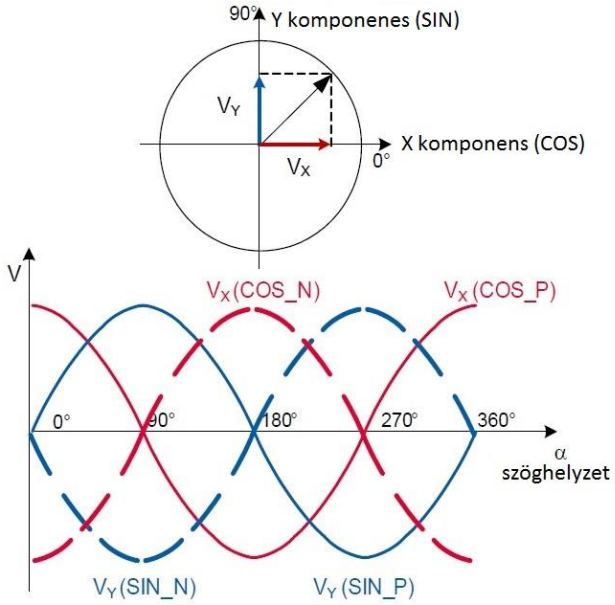


6-1. ábra GMR szenzor felépülése – chip felső szelete (top die)

Látható, hogy a két híd miatt két feszültségérték (V_x és V_y) együtt szolgáltatja a szenzor kimeneti értékét. Ez a két feszültségérték arányos a mágneses tér X illetve Y irányú komponensével. Ezekből az értékekből az atan2 (másképpen rövidítve arctg2) függvény segítségével számíthatjuk ki a szöghelyzet értékét, az alábbi kifejezés szerint:

$$\alpha = atan2(V_x, V_y)$$

Az atan2 az arkusztangens függvény egy általánosítása, amely egy síkvektor két derékszögű koordinátarendszerben adott koordinátája alapján kiszámítja a vektor irányszögét (x tengellyel bezárt szögét). Éppen erre az információra van szükségünk, ahogy az a 6-2. ábrán is látható.

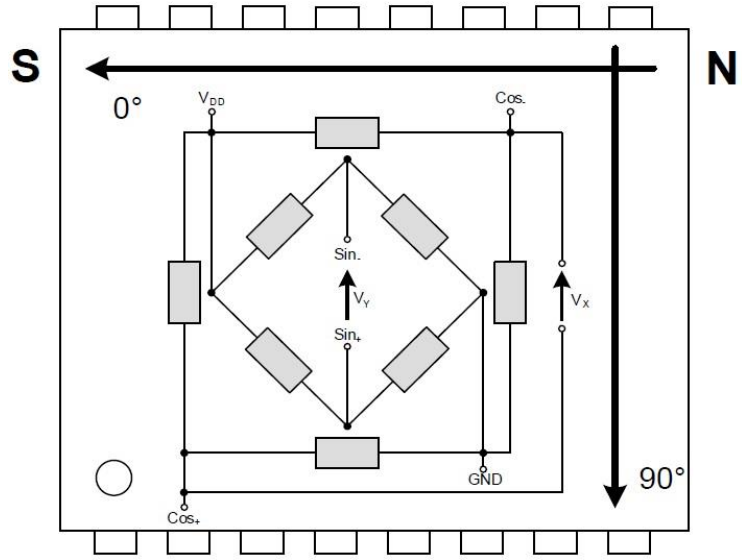


6-2. ábra Ideális GMR szenzor kimeneti jelalak

Az ábrán az is látható, hogy a szenzor valójában nemcsak 2 feszültségértéket szolgáltat, hanem négyet. Ebből is látszik, hogy redundáns az érzékelő. Ugyanis az X és Y irányú koszinusz- és szinuszjelek ponált értékei (COS_P, SIN_P) mellett azok negált értékeit is külön szolgáltatja, így közös módusú zavarások ellen jobban védett a jel. Emiatt a szenzor jeleinek feldolgozása során érdemes mind a négy jelet felhasználni, és képezni a ponált és negált jelek számtani közepéből a szenzor ofszetértékét, majd ezt levonni a ponált jelértékből, és ezeket a jelértékeket felhasználva képezni a szöghelyzet értékét az atan2 függvény segítségével.

6.3. AMR szenzor

Az AMR szenzor kialakításban és viselkedésben is nagyon hasonlít az előzőekben bemutatott GMR-re. Itt azonban az alkalmazott hídstruktúra kimeneti jelértékei csak a mágneses tér 90°-os tartományig egyértelműek, emiatt a két hídstruktúrát 45°-os szögben kell egymáshoz képest elhelyezni, hogy 0°-180°-os szöghelyzet tartományban lehessen érzékelni, lásd 6-3. ábra.

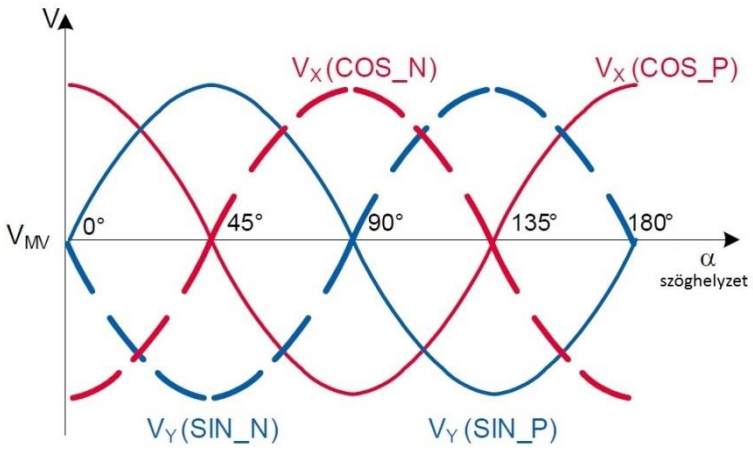


6-3. ábra AMR szenzor felépülése – chip alsó szelete (bottom die)

A szenzor által szolgáltatott Vx és Vy feszültségértékek ugyanúgy a mágneses tér X és Y irányú komponensével arányosak, és majdnem ugyanolyan módon kell kiszámítani a szögértéket, mint az előző esetben. Az AMR kialakítása miatt azonban a számítások végén még kettővel osztani kell az értéket. Így a szöghelyzet függvénye:

$$\alpha = \frac{atan2(V_x, V_y)}{2}$$

A szenzor ugyanúgy, mint az előző esetben is, a ponált és negált értékeket is szolgáltatja, amelyeket a 6-4. ábra szemléltet.



6-4. ábra Ideális AMR szenzor kimeneti jelalak

6.4. Alkalmazott szenzor pontossága

A bevezetőben említettem, hogy a szenzor igen nagy pontosságra képes, azonban az elérhető pontosság sok egyéb paramétertől függ.

A szenzor hibáit négy csoportba lehet osztani. Az első az ofszethiba, vagyis amikor 0°-os szögértéknél nem pontosan 0 értéket mutat a szenzor. A második az egyidejűségi hiba, vagyis az abból eredő hiba, hogy az érzékelés nem nulla idő alatt zajlik le. A harmadik pedig az ortogonalitási hiba, amely csak a GMR szenzornál lép fel, és abból ered, hogy a két hídstruktúra nem tökéletesen 90°-ban áll. Az utolsó a hiszterézishiba, amely a szenzorok szerkezetében lévő mágneses anyagok tulajdonságából származik.

Ha azonban mind a négy jelet (ponált és negált jeleket) is felhasználjuk, és az alapján a már említett ofszetkorrekciót elvégezzük, akkor az első hibatípus eliminálható. Az első használat előtt ajánlott egy megfelelő kalibrációs eljárást végrehajtani, amelynek során egy körülfordulás alatt a szenzor által szolgáltatott értékekből kiszámíthatók a korrekciós paraméterek. Ezek felhasználásával az is elérhető, hogy csak a hiszterézishiba maradjon meg, a többi hibaforrás mind eliminálható.

Ha alkalmazzuk az ofszetkorrekciót és a kalibrációt, akkor az adatlap szerint összességében a maximális szöghelyzet eltérés 3° GMR szenzor esetében, míg az AMR szenzornál a maximális szögeltérés 1.7°. Ez a pontosság alapján már megvalósítható a mezőorientált szabályozás.

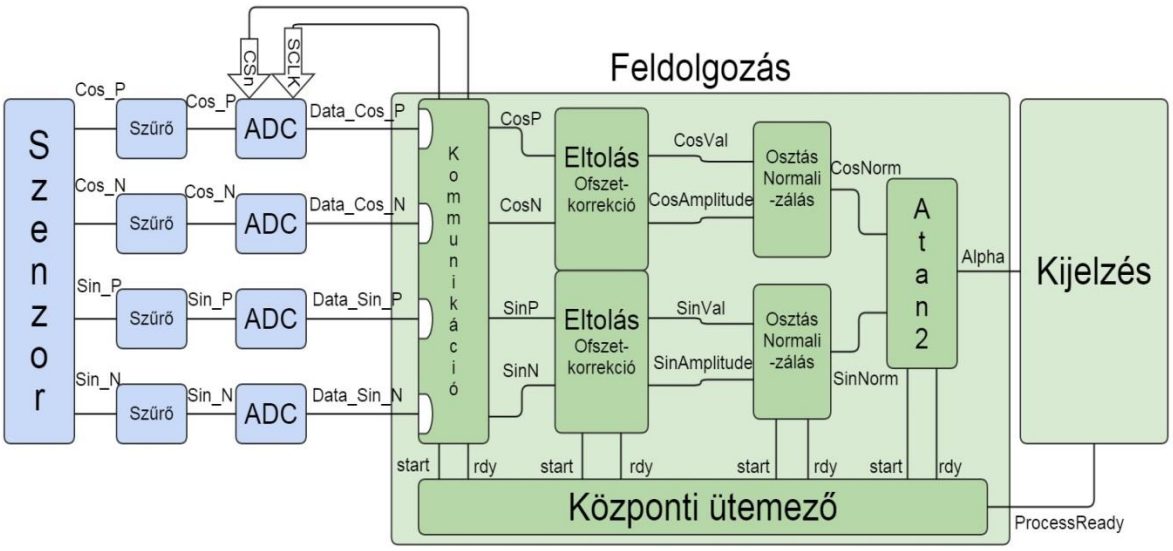
7. Jelfeldolgozás tervezési lépései

Az előző fejezetben bemutattam a szakdolgozat készítése során alkalmazott szögszenzort, ebben a fejezetben pedig a szöghelyzet kialakításának lépéseit, és nehézségeit, a tervezés közbeni problémákat és azok megoldásait ismertetem. Először az alkalmazott modulokról adok egy áttekintő képet. Bemutatom, hogy logikailag, funkcionálisan és fizikai valójában hogyan épül fel a szögszenzor jelének feldolgozási folyamata. Ezek után pedig részletesebben is ismertetem a legfontosabb modulok tervezési megfontolásait.

Minden részletre azonban nem áll módomban kitérni, így a későbbi fejezetekben is csupán a megvalósítás legfontosabb lépéseire térek ki, amelyek nehezen érthetőek, vagy nem egyértelműek a megoldás szempontjából. Emiatt is fontos, hogy az egyes részletek előtt bemutassam a jelfeldolgozás folyamatát.

7.1. Funkcionális blokkvázlat

A teljes jelfeldolgozási folyamat funkcionális blokkvázlata látható a 7-1. ábrán.



7-1. ábra Jelfeldolgozás funkcionális blokkvázlata

Az ábrán bal szélről indulva, jobbra haladva láthatók az egymás utáni lépések. A jelfolyam is lényegében balról jobbra halad, másik irányban csak az analóg-digitális átalakítók (ADC-k) vezérléséhez szükséges jelek haladnak.

Az ábrán kék színnel tüntettem fel azokat a funkcionális feladatokat, amelyek megvalósítása a külön tervezett illesztőpanelen történt. Ide tartozik: a szenzor, az analóg jelek szűrése és az analóg-digitális átalakítás. Zöld színnel jelöltem azokat a feladatokat, amelyek megvalósítása az FPGA áramkörön történt. Ide tartozik: a kommunikáció az analóg-digitális átalakítókkal, az ofszetkorrekció, a normalizálás, az atan2 függvény megvalósítása és a kijelzés.

Az ábrán baloldalt látszik az analóg szögszenzor, amelynek az AMR része ki van kapcsolva és csak a GMR része szolgáltat adatokat. Jól látszik, hogy a szenzor ponált (Cos_P, Sin_P) és negált (Cos_N, Sin_N) jeleinek felhasználásával megvalósul az ofszetkorrekció a jelfeldolgozás során. Ezeket a jeleket ezután egy aluláteresztő RC szűrőn átvezetve a zavarokból adódó (pl.: közeli nagy teljesítményű motor miatti) nagyfrekvenciás jelek kiszűrhetők.

A szűrt jeleket analóg-digitális átalakítók segítségével, 20 kHz-el érdemes mintavételezni, ugyanis ez az optimális mintavételi frekvencia a különböző alkatrészek szempontjából. Az analóg-digitális átalakítók egy-egy soros kommunikációs vonalon szolgáltatják az adatokat az FPGA áramkör felé.

Az FPGA áramkörben minden feldolgozó modul egy start jel hatására beolvassa saját regisztereibe a bemenetén lévő értéket, azon elvégzi a megfelelő átalakítást, feldolgozást néhány belső órajel alatt, majd kimeneti regiszterébe teszi a végeredményt, amelynek véglegességét egy rdy („ready”/„kész”) jellel jelzi. A feldolgozás során a start jeleket egy központi ütemező osztja megfelelő ütemben az egyes modulok számára, illetve figyeli a modulok ready jelét, és az alapján adja a start jelet a következő modul számára. Az egész feldolgozás végén az ütemező szolgáltat egy végső kész jelet (ProcessReady), amellyel jelzi a szabályozási körben következő modulnak, hogy a szöghelyzet értéke előállt és stabil. Az ütemező felelős továbbá a 20 kHz-es mintavételezési frekvencia kialakításáért is.

A kommunikációs modul feladata egyrészről az analóg-digitális átalakítók vezérlése, a megfelelő órajel (SCLK), kiválasztójel (CSn) kialakítása a mintavételi frekvencia és az átalakítók kommunikációs protokollja alapján. Ezenkívül fogadja, az analóg-digitális átalakítók soros vonalain érkező adatokat, és kialakítja belőlük egy regiszterbe az aktuális ponált és negált koszinusz-, illetve szinuszértékeket. Mind a négy értéket egyszerre fogadja, egyszerre választja ki mind a négy analóg-digitális átalakítót, majd innentől kezdve a további feldolgozási lépések is párhuzamosan folynak.

A következő lépésben a ponált és negált jelek alapján az ofszetkorrekciós modul végzi az ofszet kiszámítását, illetve ez a modul érzékeli a koszinusz- és szinuszjelek amplitúdóját is minden periódusban. Az ofszetkompenzáció már itt azonnal meg is valósul az aktuálisan kiszámított ofszet alapján. Az aktuális amplitúdó is fontos információ, ugyanis változhat az idő során, valamint később a normalizálás ehhez az értékhez képest fog megtörténni. Emiatt az ofszetkompenzált jeleknél ennek a modulnak figyelnie kell a maximum-, illetve minimumértékeket, és a legutolsó szélsőértéket kell megjegyeznie, amíg újabb csúcs nem érkezik.

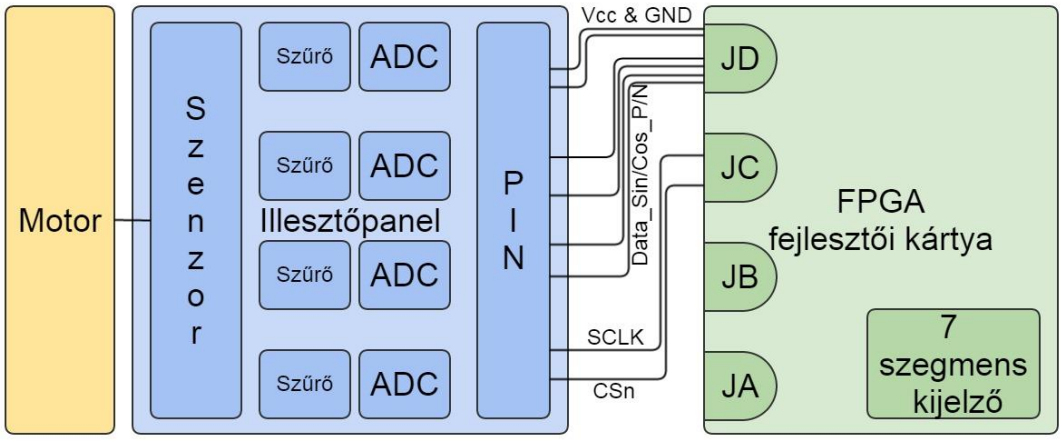
Az ofszetkompenzált jelek amplitúdójukra történő normalizálását a normalizáló modul végzi el, amely átskálázza a jelértékeket, úgy, hogy az előzőleg érzékelt amplitúdóértéket tekinti 1-nek. Ez a művelet egy osztás elvégzésével történik, mellyel elérhető, hogy a normalizált jelek értékei -1 és 1 közé essenek. Ez fontos, mert az atan2 függvény bemeneteként csak ilyen jelet szabad szolgáltatni.

Az utolsó feldolgozási modul a normalizált koszinusz- és szinuszjeleken végrehajtja az atan2 függvényt, a szenzor adatlapjában leírtaknak megfelelően. Ezzel előáll a mechanikai szög, amely a forgórész szöghelyzetét mutatja.

Az egész folyamat legvégén az FPGA fejlesztőkártyán lévő hétszegmenses kijelzőn megjelenik az aktuális szögérték. Ez egy ideiglenes megoldás, későbbiekben éles működéskor ezt a jelet a szabályzó fogja felhasználni.

7.2. Fizikai blokkvázlat

A modulok fizikai megvalósítását és összeköttetését szemlélteti a 7-2. ábra.



7-2. ábra Jelfeldolgozás fizikai komponensei és kapcsolatuk

Az ábrán látható, hogy a motor közelébe helyezett illesztőpanelen található a szenzor, amely méri a motor tengelyén lévő mágnes irányát. Az illesztőpanelen található még a szenzor analóg jeleinek szűrése, valamint mintavételezése és digitalizálása is. Végül az összes jel egy tűsorra kerül, ahonnan továbbítható majd az FPGA felé, 8 vezeték segítségével. Ezek rendre a táp és a föld (amelyeket az FPGA szolgáltat az illesztőpanel számára is), a 4 soros adatvezeték (amelyeket az illesztőpanel hajt meg), valamint az órajel és a kiválasztójel (amelyeket az FPGA szolgáltat).

Az FPGA fejlesztői panelen 4 darab pmod csatlakozó található. Ezek 6 lábú csatlakozók, ahol 2 láb mindig a táp és a föld, a maradék 4 pedig az FPGA bizonyos lábainak a kivezetése. Így 2 ilyen csatlakozót felhasználva könnyen kialakítható a kapcsolat az illesztőpanel felé. Egyik csatlakozón a táp- és föld-, valamint a négy soros adatvonal, míg a másik csatlakozón a két vezérlőjel átvitele valósul meg.

A Xilinx Spartan 3E FPGA áramkört tartalmazó fejlesztői panelen található egy hétszegmenses kijelző is, amelyről ideiglenesen leolvasható a végeredmény.

7.3. Illesztőpanel kialakítása

A mérés megvalósításához szükséges volt egy illesztőpanel, kisméretű nyomtatott áramkör megtervezése, amely mechanikusan illeszkedik a szervomotor kialakításához, méri a motor tengelyén lévő mágnes szöghelyzetét, és valamilyen kommunikációs protokoll szerint a mérési eredményeket továbbítja az FPGA áramkör felé. Feladatom volt az áramkör megtervezése is, a gyártást a ThyssenKrupp szervezte.

Az illesztőpanel több szempontból is központi eleme maga az analóg szögszenzor. Logikailag a szenzor a központi elem, mert az egész panel legfontosabb feladata a forgó mágnes szöghelyzetének detektálása. Másrészről fizikailag is a panel közepén kellett elhelyezni a szenzort, ugyanis a motor tengelyével egy vonalban kell, hogy legyen a szenzor, és a motor konstrukciójának kialakításából adódik, hogy a motor tengelye a nyomtatott áramköri panel közepére esik. A szenzor középpontba történő helyezése azzal az előnnyel is jár, hogy az általa szolgáltatott analóg jelek elvezetése az analóg-digitális átalakítókig sugárszerűen megoldható, emiatt az analóg jelvezetékek igen rövidek. A szenzornak csak a GMR érzékelője került felhasználásra – az AMR része tápot sem kap – viszont mind a négy kimeneti analóg (ponált és negált koszinusz illetve szinusz) jelet is felhasználják a későbbi modulok, emiatt mindegyiket továbbítani kell az FPGA áramkör felé.

Fontos megemlíteni, hogy ezek az analóg jelek viszonylag kis frekvenciájúak lesznek, ugyanis a motor forgási sebessége maximálisan 4500 fordulat/perc. Ezzel azonos frekvenciájúak lesznek a szenzor által szolgáltatott analóg szinuszos jelek is. Ezeknek a jeleknek a mintavételezése 20 kHz-es frekvenciával történik. A mintavételi frekvenciát és ezzel együtt a feldolgozás sebességét is érdemes lenne növelni, hogy a különböző zajok hatásait ki lehessen küszöbölni, a beavatkozásnál azonban a végfok kapcsolóinak kapcsolási veszteségei ekkor növekednének. Így ez a ThyssenKruppnál alkalmazott 20 kHz az optimális mintavételi frekvencia a kapcsolási veszteségek, a zaj és a vezetési veszteségek figyelembevételével.

Az illesztőpanelen valósul meg a szenzor által szolgáltatott analóg jelek szűrése is. Erre azért van szükség, hogy a különböző (pl. a közeli nagy teljesítményű motorból származó) zajok ne módosíthassák a szenzor analóg jeleit, és az analóg-digitális átalakítókra már csak a kisfrekvenciás hasznos jel kerülhessen. Egy elsőfokú RC aluláteresztő szűrővel megvalósítható megfelelő zajszűrés, ahol az ellenállás értéke 10 kΩ, míg a kondenzátoré 330 pF. Ugyanis az elsőfokú RC aluláteresztő szűrő vágási frekvenciája:

$$f_0 = \frac{1}{2\pi RC}$$

ahol „R” jelöli az ellenállás értékét, „C” pedig a kapacitását. Behelyettesítve 48.2 kHz adódik. Ez az érték egy megfelelő vágási frekvencia, egyrészt, mert a szervomotor korábbi EMC vizsgálatai során megmérve a zavarójeleket azok ezen érték felettiek voltak, másrészt, mert a panel digitális adatvonalain a változások már MHz nagyságrendbe esnek, így mindkét zavartípust a szűrő megfelelően kiszűri. Továbbá az is fontos, hogy az analóg-digitális átalakító mintavételi frekvenciája és a mintavételi törvény alapján az átalakító megfelelő módon tudja mintavételezni a jeleket a szűrő után.

A panelen található még a négy analóg-digitális átalakító. A választott átalakító egy 12 bites szukcesszív approximációs nagy sebességű analóg-digitális átalakító. Képes akár 1 MSPS mintavételi sebességre is, így a 20 kHz-es mintavételezésre bőven megfelel. Egyszerű kommunikációs protokollja indokolja ezen átalakító választását, ugyanis csak egy kiválasztójelre (CSn), egy órajelre (SCLK), táp- és földvezetékre van szüksége. Egy soros adatvonalon szolgáltatja az analóg bementén mintavételezett jel digitális értékét az órajel ütemében, ha kiválasztották. A négy átalakító egy időben

mintavételezi az analóg bemenetén a szenzor négy analóg jelét, ugyanis közös a kiválasztó- és az órajel. Tápja és földje közös mind a négynek, a szenzorral együtt, mint ahogy azt a szenzor adatlapjában is ajánlották, mert így lehet a legkönnyebben digitalizálni az analóg jeleket. Ráadásul ilyenkor az esetleges kisebb tápingadozás sem okoz hibát, ugyanis mind a szenzort és az analóg-digitális átalakítókat is egyformán érinti. Emiatt azonban a tervezés során hosszú tápvezeték adódott, amely keresztülmegy az egész panelen, így a panel mindkét végén megfelelő tápszűrő kondenzátorok alkalmazása szükséges a megfelelő tápfeszültség kialakításához.

A panel szélén egy tűsorra kerül ki az összes FPGA felé szükséges jel, vagyis sorrendben a táp-, a földvezeték, a négy analóg-digitális átalakító 4 soros adatvonala, a közös kiválasztó- és a közös órajel.

Fontos megemlíteni, hogy a panel készítésekor minden jelútban található egy tesztpont is, ahol mérőműszerrel meg lehet mérni a vezetéken a feszültségszintet. Ez igaz a négy analóg jelre is, és az előbb felsorolt és kivezetett digitális jelekre is. Utóbbiaknál azért kell külön tesztpont is, mert a csatlakozó kialakítása miatt nem biztos, hogy a tűsoron hozzá lehet férni a jelekhez.

Az alkatrészek kiválasztásán kívül még kettő fontos tervezési szempontot kellett figyelembe venni. Az egyik szempont a panel alakja. Ugyanis maga a motor és a kialakítása már adott volt, így a hozzá illeszkedő áramköri paneleknek is adott helye van fix mérettel. Fontos, hogy nemcsak a panel alakjára kellett odafigyelni, hanem a rögzítő csavaroknak is kellett furatot tervezni a panelben megadott helyekre. A tervezés során a panel alakjával és az alkatrészek elhelyezésénél tükörképben kellett gondolkodni, ugyanis a szenzornak a motor felé kell néznie, vagyis a motor tetején lévő panel lefelé fog nézni. A másik tervezési szempont, hogy a hátoldalon ki kellett alakítani egy nagy felületű egybefüggő földpotenciálú réteget. Ennek kettős oka van. Az egyik, hogy az analóg-digitális átalakító kialakítása miatt nem lehetett különválasztani az analóg és a digitális földet, így a nagy felület segítségével kerülhető el a digitális jelek váltása miatti zavarok. Az egybefüggő földréteg másik célja az árnyékolás, hogy sem a motor sem maga a panel ne keltsen zavarhatásokat.

Magát a nyomtatott áramköri terveket egy könnyen kezelhető internetes áramkörtervező programmal készítettem el, amit a konzulensem, Balogh András ajánlott. A program neve Upverter [15], amely ingyenesen használható, és jó támogatást ad a megismeréséhez.

7.4. Központi ütemező

Az FPGA-ban implementált feldolgozó modulok vezérlése – egy erre a célra külön implementált FPGA modul – a központi ütemező segítségével valósul meg. A modul feladata egyrészt a 20 kHz-s mintavételi frekvencia előállítása, másrészt a feldolgozási folyamat irányítása.

A 20 kHz-es mintavételi frekvenciát, az FPGA 50 MHz-es órajeléből lehetséges előállítani egy egyszerű engedélyező jel kialakításával. Vagyis a központi ütemező minden T=1/(20 kHz)=50 μs időközönként kiad egy 1 órajel szélességű start impulzust.

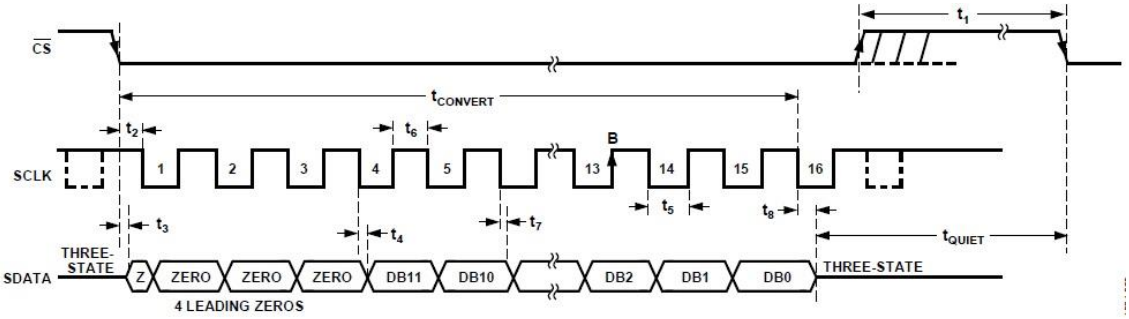
A feldolgozási szál vezérlése megoldható egy állapotgéppel. Az állapotgép a központi ütemezőn belül működik, és 5 darab 1 bites regiszterből épül fel. Az állapotgép alapja, hogy figyeli az egyes modulok kész (ready) jelzését, és ha az megérkezett, akkor lép a következő állapotba, valamint a központi ütemező is ekkor adja az őt követő modulnak a startjelet. Az állapotgép regiszterek értékei szolgáltatják a start jeleket a feldolgozási lánc egyes moduljai számára. Ezek a start jelek mindig 1 órajel szélességű impulzusok. Az első modulnak – a kommunikációs modulnak – a 20 kHz-es frekvencia alapján ad startjelet az ütemező. Utána a többi modul akkor kapja meg a startjelet, ha az előző elkészült a feldolgozással. Ha az utolsó modul is elkészült, és ezt jelezte a központi ütemezőnek, akkor maga az ütemező szolgáltatja a külvilág felé a kész (ready) jelzést.

7.5. Kommunikációs modul

A kommunikációs modul az első modul a feldolgozási láncban, amely már az FPGA áramkörben került megvalósításra. Feladata, hogy összekösse az FPGA fejlesztői kártyát és az illesztőpanelt egy egyszerű kommunikációs protokoll segítségével. A megfelelő kommunikációhoz szükséges, hogy a modul kialakítsa a panelen lévő analóg-digitális átalakítók vezérlőjeleit, valamint, hogy az átalakítók által szolgáltatott digitális jeleket fogadja, és elérhetővé tegye az FPGA áramkörben lévő, a feldolgozási láncban következő modul számára.

A választott analóg-digitális átalakító kommunikációs protokollja igen egyszerű, ez indokolja az alkalmazását az illesztőpanelen. A protokoll egy soros kommunikációt ír elő, mert az átalakítónak egy digitális kimenete van, vagyis egy lábán jelenik meg egymás után az átalakítás 12 bites értéke. Az átalakító analóg bemenetének

mintavételezése és a digitális kimeneten a bemeneti jel digitális értékének előállítása két vezérlőjellel történik. Az egyik a kommunikációs órajel (SCLK), a másik a kiválasztójel (CSn). Az analóg csatorna mintavételezése a kiválasztójel lefutó élére kezdődik meg. Ezután a kommunikációs órajel ütemében, 16 órajel alatt a digitális kimeneten megjelenik négy vezető nulla érték, majd a 12 bites jelérték, a legmagasabb helyi értékű bittel kezdve. Egészen pontosan a kiválasztójel lefutó élének hatására az adatkimenet kikerül a nagyimpedanciás állapotból, és megjelenik az első vezető nulla érték. Onnantól kezdve minden órajel lefutó élre kerül a kimenetre a következő bit. Emiatt az utolsó bit a 15. órajel lefutó él hatására kerül ki, míg a 16. órajellefutás hatására az adatvonal ismét nagyimpedanciás állapotba kerül. Ez látható a 7-3. ábrán.



7-3. ábra Analóg-digitális átalakító soros kommunikációs protokollja

A kommunikációs protokoll előírja, hogy a kiválasztójel lefutó éle után az órajelnek is lefutó éle következzen, és a két él között legalább 10 ns időnek kell eltelnie. Ezenkívül a 16 órajel alatt a kiválasztójelnek alacsony értéken kell maradnia, különben a vett mintát az átalakító eldobja, és azonnal leállítja az adatszolgáltatást. Így a 16. órajel után lehet magasba húzni a kiválasztó jelet, aminek hatására a digitális adatkimenet nagyimpedanciás állapotba kerül, és az órajel értéke is közömbös, akár le is állhat.

A kommunikációs modul feladata, hogy a központi ütemezőtől kapott start jel hatására a fent ismertetett módon kialakítsa a kommunikációt. Fontos megemlíteni, hogy a panelen lévő négy analóg-digitális átalakítót közös órajellel és közös kiválasztójellel kell vezérelni, ugyanis így közel egyszerre vehető minta mind a négy analóg jelből, és az FGPA párhuzamosan dolgozza fel az átalakítók által szolgáltatott digitális jeleket.

Az analóg-digitális átalakító adatlapjából kiolvasható, hogy órajelként képes akár 20 MHz-es jelet is fogadni. Az FPGA fejlesztői kártyán az órajel generátor 50

MHz-es órajelet állít elő. Az FPGA órajelét néggyel leosztva SCLK = 12.5 MHz-es órajel áll elő, amely megfelelő az analóg-digitális átalakítónak is. A tervezés során nem volt szempont a lehető leggyorsabb kommunikáció megvalósítása, emiatt lehetett alkalmazni ezt az egyszerű leosztást. A kommunikáció órajele (SCLK) az alaphelyzetbe állítás után folyamatosan elérhető az FPGA kimenetén, ugyanis ha nincsen kiválasztva az átalakító, akkor közömbös az órajel értéke. Így az órajel kialakítása a lehető legegyszerűbb lehet. Fontos megemlíteni, hogy ez a 12.5 MHz-es órajel csupán az analóg-digitális átalakító és az FPGA közötti kommunikációt vezérli. Maga a mintavételezés továbbra is 20 kHz-cel történik. Tehát a kommunikációs modul minden T = 50 µs időközönként kap egy start jelet a központi ütemezőtől, és $T_{convert}$ = 1,28 µs idő alatt kialakítja az eredményt, amely az alábbi módon számított érték:

$$T_{convert} = 16\left(\frac{1}{12.5\ \text{MHz}}\right) = 1{,}28\mu\text{s}$$

A kommunikációs protokoll leírásakor említettem, hogy fontos, hogy a kiválasztójel lefutó éle után a kommunikációs órajel is lefutó éllel folytatódjon, és legalább 10 ns idő elteljen a két élváltás között. Mivel a kommunikációs órajel folyamatos, emiatt a kiválasztójel megfelelő időzítésével kell biztosítani ezt az időt. A modul érzékeli a kommunikációs órajel felfutó élét, start jel esetén pedig nem azonnal húzza le a kiválasztójelet alacsony értékbe, hanem a kommunikációs órajel felfutása után. A metódus pontos megvalósítása az, hogy egy regiszterbe az FPGA órajelével vételezve a modul folyamatosan elmenti a kommunikációs órajel előző értékekét. Ha a regiszter értéke 0, míg az aktuális kommunikációs órajel érték 1, akkor volt felfutó él, és csak ezek után vált nullába a kiválasztójel. Ezzel mindkét feltétel teljesül. Mert így biztosan előbb lesz lefutása a kiválasztójelnek, és csak utána a kommunikációs órajelnek. Továbbá a kommunikációs órajel magas értéke alatt két teljes FPGA órajel periódus is lezajlik. Így az első FPGA órajel után lehet érzékelni a váltást, és azonnal nullába húzni a kiválasztójelet. Ezzel a két jelváltás között biztosan lesz 1 FGPA órajelnyi idő, ami t=1/(50 MHz) = 20 ns. A kiválasztójel természetesen a kommunikáció 16. órajele után visszaáll magas értékbe.

A 16 bites adat beléptetését egy állapotgéppel célszerű megvalósítani. Az állapotváltozó egy számláló, amely a kommunikációs órajel (SCLK) lefutó élére számol, tehát azt jelzi, hogy a kommunikációban hány lefutó él volt eddig. A számláló nullába áll, ha az alaphelyzetbe állítás aktív, vagy ha kiválasztójel magas értékű.

Egyébként minden SCLK lefutó él hatására növekszik eggyel. Fontos átlátni, hogy a számlálót a kiválasztójel lefutása indítja és a felfutása állítja nullába ismét, míg a kiválasztójel a start jel hatására megfelelő időben húzódik alacsony értékbe, és a számláló 16-os értékére megy vissza magas állapotba.

Az adatok beléptetése a kommunikációs órajel (SCLK) felfutó élére történik, ugyanis az analóg-digitális átalakító a lefutó élkor adja ki az új értéket, amely stabilan megjelenik az adatvonalon a felfutó élkor. Emiatt a kommunikációs modul az állapotgép 0 és 16 közötti állapotaikor egy shiftregiszter segítségével folyamatosan hozzáilleszti az eddig beérkezett adatokhoz az aktuális bit értékét. Említettem, hogy az első vezető nulla a kiválasztójel lefutásakor jelenik meg az átalakító adatkimenetén, és az első órajel lefutáskor már a második vezető nulla kerül rá. Az egyszerűség kedvéért az első vezető nullát nem fogadja a kommunikációs modul, hanem csak 3 nulla értéket és a 12 bites eredményt fogadja. Így elegendő egy 15 bites shiftregiszter az adatok fogadására, valamint megfelelő a kommunikációs órajel felfutó órajelére történő beléptetés is.

Legvégül a 15. állapot végén egy 12 bites kimenőregiszterbe kerül a vezető nullák nélküli adat, és ezzel együtt a kész (ready) jelzés is kialakításra kerül.

7.6. Ofszetkorrekciós modul

A kommunikációs modul egy 12 bites értéket szolgáltat a kimeneti regiszteréből, amelyeket még tovább kell alakítani, hogy a szenzor adatlapjában írt atan2 függvény segítségével a szöghelyzet kialakuljon. Ennek első lépése az ofszetkorrekció.

Az analóg-digitális átalakító négy 12 bites értéket szolgáltat. Ennyi bitet feldolgozni azonban felesleges lenne, hiszen a szenzor hibája miatt a már a mérési hiba is maximum 3° lehet. Ez a 3°-os eltérés az analóg jeleknél ott okozza a legnagyobb tévedési lehetőséget, ahol a szinuszfüggvény a leglaposabb, hiszen itt kis feszültségeltérés is nagy szögeltérést okozhat a számítás során. A tápfeszültség 3.3 V. Ez alapján kiszámítható, hogy 3° eltérés kb. 4.5 mV hibát okoz az analóg jelekben. A legkisebb bit pedig 0.806 mV különbség hatására már változik. Az értékeket az alábbi módon kaptam:

$$(\sin(90°) - \sin(87°)) \cdot 3.3\text{V} = 4.523\text{mV}$$

$$\frac{3.3V}{2^{12}} = 0.806mV = LSB \; ; \; 2LSB = 1.611mV \; ; \; 3LSB = 3.22mV$$

Ezek alapján látható, hogy az alsó két bit elhagyásával még a számítások során megjelenő kerekítési hibák is elhanyagolhatók lesznek. Emiatt a kommunikációs modul által szolgáltatott szám alsó két bitje a feldolgozási lánc elején elhagyható, és csak a felső 10 bit értéke alapján elegendő kiszámolni a szöghelyzet értékét.

A szenzor által szolgáltatott koszinusz- és szinuszkomponenseknek a ponált és a negált értékét is továbbítja a kommunikációs modul, amelyek immáron 10 bites adatként betöltődnek az ofszetmodul bemeneti regisztereibe, ha a központi ütemező start jelet szolgáltat az ofszetkorrekciós modul számára. A szenzor által mért koszinusz- és szinuszkomponens is ofszethibával rendelkezhet, ráadásul az ofszet mértéke hőmérsékletfüggő, emiatt a pontos mérési eredményhez elengedhetetlen e hiba eliminálása. A ponált és negált értékek segítségével azonban az ofszet kiszámítható. Az ofszet aktuális értéke az aktuális ponált és negált jel számtani közepe. Az ofszet értékét minden minta esetén elő kell állítani mind a két komponensre.

A bemeneti regiszterekbe beolvasott számokat 10 bites előjel nélküli jelként kell értelmezni. A beolvasáskor az átláthatóság miatt 11 bites regisztereket célszerű használni, és a felső bit értékét 0-val kitölteni, hogy a későbbiekben kettes komplemens formátumra való áttéréskor ne lehessen probléma ebből. Egy koszinusz- és egy szinuszjel azonban felvehet negatív értékeket is. Így ezeket a jeleket el kell tolni a függőleges tengely mentén, hogy a szinuszos jelek középértéke zérus lehessen. Ehhez az aktuális jelértékeket pontosan az aktuális ofszet értékével kell eltolni. Emiatt minden minta esetén a ponált jelértékből kivonva az aktuálisan számított ofszet értékét, megkaphatók a komponensek valódi értékei, amelyek a valódi szinuszos jellegű eredményeknek tekinthetők. Ezek az értékek immáron 11 bites kettes komplemens formátumú egész számértékek.

Ez a modul szolgáltatja még a feldolgozási láncban következő modulnak a koszinusz- és szinuszjelek legutóbb mért amplitúdóértékét is. Tehát minden fél szinuszhullám végén figyeli az abszolút értékben maximálisan előforduló értéket. Megfelelő működés esetén a szenzor által szolgáltatott koszinusz- és szinuszjelek 90°-os fáziskülönbségben vannak. Tehát amikor a koszinuszkomponensben nullátmenet van, akkor a szinuszkomponens abszolút értéke maximális, és ez fordítva is igaz. Emiatt a tulajdonság miatt az amplitúdómérés úgy is meg valósítható, hogy a koszinusz- vagy a szinuszkomponens nullátmenetekor a másik komponens értékét tekintjük az amplitúdó értékének. Ez a kettes komplemens számábrázolás miatt könnyen megvalósítható a

legmagasabb helyi értékű bit változásának figyelésével. Tehát mindig eltárolva egy regiszterben az előző mintánál mindkét komponens legmagasabb bitértékét, ha az az egyik komponens esetében más, mint az aktuális bit értéke, akkor a másik komponens teljes aktuális értéke lesz a modul által szolgáltatott amplitúdóérték, amíg nem érkezik egy újabb váltás.

Ez az amplitúdómérés nem a méréstechnikában szokványos módszer, mert ott néhány periódus alatt vett több minta segítségével állapítható meg a jel amplitúdója. A nullátmenet figyelésének módszerével szinte azonnal előállítható az amplitúdóérték, ráadásul a módszer kisebb számítási igényű is. Ez a módszer azonban a mintavételi pont bizonytalansága miatt hibakomponenst okoz az érzékelésben. Szerencsére jelen esetben a hiba mértéke elhanyagolható, mert a szervomotor maximális fordulatszáma 4500 ford./perc, ami átszámítva 75 Hz-es frekvenciának felel meg, ami egyben a szenzor által szolgáltatott analóg szinuszos jelek maximális frekvenciája is. A mintavételi frekvencia 20 kHz, tehát a legrosszabb esetben is, amikor a motor maximális fordulaton működik a szinuszjel egy periódusában kb. 266 minta áll rendelkezésre, mert:

$$\frac{t_{szinusz}}{t_{mintavétel}} = \frac{\frac{1}{75\ Hz}}{\frac{1}{20000\ Hz}} = \frac{800}{3} \cong 266.667$$

Az amplitúdómérés hibája maximális, ha a szinuszos jel amplitúdója két mintavételi időpont között félúton lenne. Ilyenkor azonban az eredő bizonytalanság a periódusnak csak kb. 533-ad része lenne, amelynek hibahatása elhanyagolható, mert a legrosszabb esetben is az amplitúdómérés hibája a szinuszos jelre:

$$\left|\cos\left(\frac{360°}{533}\right) - 1\right| \approx 0.00007$$

Végül a modul a kiszámított valódi jel- és amplitúdóértékeket átalakítja a 11 bites kettes komplemens egész formátumból, 1 előjelbit és 10 előjel nélküli egészrész bit formátumba, ugyanis a feldolgozásban következő modul számára ebben a formátumban szükségesek az adatok.

Ez a modul összesen három órajel alatt végez a teljes feladatával. Az első órajel alatt beolvassa a kommunikációs modultól kapott értékeket a bemeneti regisztereibe. A második órajel alatt kiszámítja az ofszetet és az ofszetkompenzált jelértékeket is. Egy harmadik órajel szükséges az amplitúdó vizsgálatához. Így a modul kész (ready) jelzése kialakítható a start jel 3 órajellel történő késleltetésével.

7.7. Normalizáló modul

A szenzor adatlapjából kiolvasható, hogy atan2 függvény segítségével számítható ki a szöghelyzet a koszinusz- illetve a szinuszkomponensből. Az atan2 függvény bemenete azonban csak -1 és 1 közötti érték lehet, emiatt az előzőekben kapott komponensértékeket tovább kell alakítani, hogy e két szám közé essenek, vagyis normalizálni kell őket.

Az előző modul mindkét komponens esetén előállította az aktuális jelértéket, és az előző fél szinuszhullám amplitúdóértékét is. Ez utóbbi értéknél nem fordulhat elő abszolút értékben véve nagyobb érték a fél szinuszhullám alatt, vagyis ha az amplitúdó értéke egységnyinek tekinthető, akkor a többi jelérték -1 és 1 közötti szám lesz. Így normalizálható a jelérték. A normalizálás megvalósításához, tehát minden minta esetén, mindkét komponensnél el kell osztani az aktuális jelértéket az amplitúdó értékével, és így előáll a két normalizált eredmény.

Normalizáláshoz létre kell hozni egy osztó modult, és ebből egy-egy modult példányosítani is kell a koszinusz- és a szinuszkomponens feldolgozási láncába külön-külön. A két példány ugyanazt a műveletsort hajtja végre, de az átláthatóság kedvéért mégis inkább célszerű különválasztani a kettőt, így külön kapnak start jelet, és adnak kész jelet a központi ütemező felé.

Az osztó modul bemenete két 11 bites szám, 1 előjelbit és 10 egészrész bit formátumban. Kimenete egy 11 bites szám 1 előjelbit, 1 egészrész bit, és 9 törtrész bit formátumban. Minden szám fixpontosnak tekinthető. Ha a hányados előjelét külön állapítjuk meg, akkor elegendő egy fixpontos pozitív számok osztását végző aritmetikai modul megvalósítása. Ezen osztó algoritmus alapja, hogy kezdésként megfelelő hosszúságú regiszterek tetejébe kell másolni az osztót és az osztandót, majd ki kell nullázni a hányados regiszterét és megfelelő értéket kell betölteni a helyiérték-mutatóba. Ezután minden ciklus első lépéseként meg kell vizsgálni az osztandó és az osztó egymáshoz viszonyított nagyságát. Ha az osztandó kisebb, mint az osztó, akkor az első lépésben nincs teendő. Ha az osztandó nagyobb vagy egyenlő, mint az osztó, akkor az osztandóból ki kell vonni az osztót, és a hányados aktuális helyi értékét 1-be billenteni. A ciklus második lépéseként el kell tolni egy értékkel jobbra az osztót, és csökkenteni kell a helyiérték-mutatót. A ciklust egészen addig kell ismételni, amíg a helyiérték-mutató az hányados regiszterének végére nem ér.

Hibátlan működés esetén az osztás végeredménye nem lehet nagyobb 1-nél. Emiatt nem szükséges a teljes osztó modul alkalmazása, mert gyorsabban és kevesebb erőforrás felhasználásával is megvalósítható a művelet. A bejövő számok egészrésze 10 bit, a kimenet 1 egészrész bit és 9 törtrész bit. Így az osztandót és az osztót egy-egy 19 bites regiszter felső 10 bitjére kell átmásolni, vagyis azonos pozíciókba lehet helyezni mindkettőt. Nem kell eltolt pozíciót alkalmazni kezdéskor, mert legfeljebb azonos lehet a két szám, az osztandó nem lehet nagyobb, mint az osztó. A jobb átláthatóság kedvéért célszerűnek tűnt egy 20 bites regisztert lefoglalása, emiatt felső bitnek 0 értéket kellett adni, ezzel jelezve, hogy pozitív számok körében történik a további műveletvégzés. A végeredménynek egy 11 bites regisztert szükséges, mert így elfér benne az 1 előjelbit, az 1 egészrész bit és 9 törtrész bit is.

Start jel hatására a normalizáló modul először bemásolja a bemeneten lévő osztót és osztandót két 20 bites regiszterbe, a 19. pozíciótól kezdve lefelé, és a regiszterek többi részét, valamint az eredményregisztert is kinullázza. Ezután a helyiérték-mutatót állítja 9-re, ugyanis az osztandót 9 alkalommal kell majd eltolni ahhoz, hogy a regiszterek végére érjen. Ezzel egyidejűleg kialakítja az eredmény előjelértékét is. Ha mind a két bemenet előjelbitje azonos, az azt jelenti, hogy mindkét szám pozitív vagy mindkét szám negatív, tehát az osztás végeredménye pozitív, emiatt a modul az eredmény regiszter legfelső bitjét 0-ban hagyja. Ha különböző a két előjel, akkor a negatív eredménynek megfelelően az eredményregiszter legfelső bitjét 1-be váltja. Ezek után az előbb leírt ciklikus algoritmussal kialakítható az eredmény többi része.

Legvégül, ha a végeredmény valamilyen ok miatt mégis 1-nél nagyobb értéket mutatna, akkor az előjelet megtartva a modul felülírja az eredményregiszter tartalmát az 1.0-nak megfelelő számértékkel. Egyéb esetben pedig a végeredményként kapott számot megtartja. Ezzel biztosítható, hogy az atan2 modul bemenetére véletlenül se kerülhessen 1-nél nagyobb érték.

Látszik az előzőekből, hogy az osztás hardveres megvalósítása bonyolult művelet, így az FPGA áramkörnek is 10 órajelciklusra van szüksége, mire az eredmény előáll. A kész (ready) jel a helyiérték-mutató 0-ba állásának figyelésével alakítható ki.

7.8. Atan2 modul

A feldolgozási lánc utolsó lépése, hogy a szenzor jelének két komponense alapján – amelyek már ofszetkompenzált és normalizált értékek – atan2 függvény segítségével kiszámítható a forgórész szöghelyzete.

$$\alpha = atan2(V_x, V_y)$$

ahol V_x az x irányú komponens, amit eddig koszinuszjelnek neveztem, és V_y az y irányú komponens, amit eddig szinuszjelnek neveztem.

Bemenetként a két komponens értékét kapja 1 előjelbit, 1 egészrész bit, és 9 törtrész bit formátumban. Ezeket a modul első lépésként visszaalakítja kettes komplemens formátumba, 2 előjeles egészrész bit és 9 törtrész bit felosztásban.

Maga az atan2 függvény, vagy akár bármelyik egyéb trigonometrikus függvény megvalósítása igen bonyolult, sok erőforrást igénylő. A CORDIC algoritmus segítségével azonban megvalósítható a legtöbb trigonometrikus függvény. A CORDIC az angol „Co-ordinate Rotational Digital Computer” kifejezés rövidítése [16]. Az algoritmus eredetileg iteratív módon, koordinátarendszerek forgatásával képes volt meghatározni egy adott vektor adott szöggel történő elforgatása után a vektor új koordinátáit. Az algoritmus nagy előnye, hogy csupán összeadás és eltolás műveleteket alkalmaz, amelyre minden processzor képes. Az algoritmus alapján CORDIC segítségével a legtöbb trigonometrikus függvény is kiszámolható, és ezek metódusai beépültek a CORDIC algoritmusba [17].

A szakdolgozat során mégsem kellett programozási módszerekkel megvalósítani ezt az algoritmust, ugyanis a Xilinx fejlesztőkörnyezete bizonyos modulokat készként tartalmaz, amelyeket a fejlesztés során fel lehet használni, ezzel gyorsítva és segítve a fejlesztést. E kész modulok összefoglaló neve IP, amely az angol „Intellectual Property” kifejezés rövidítése. Ezek a felhasználó számára fekete dobozként jellemezhetők, amelynek csak a bemeneteit, kimeneteit és néhány belső paraméterét állíthatja a felhasználó, de magát a modult leíró kódot nem láthatja. A Xilinx ISE 14.2 fejlesztőkörnyezetben az IP modulok között található CORDIC néven egy ilyen mag, amely atan2 függvény megvalósítására is képes. Az általam használt modul a Xilinx LogiCORE™ IP CORDIC, 4.0-ás verziója, amelyet 2012. április 24-én adtak ki.

Az egyszerűen felhasználható IP modul azonban a CORDIC algoritmusra jellemző módon erőforrás-igényes. Emiatt a beállítások választása során külön figyelni kell a lehető legkisebb erőforrás-igényű lehetőségek kiválasztására, amelyek még nem okoznak nagyon pontatlan számításokat. Így a felhasznált IP CORDIC modul beállítása: Arc Tan függvény megvalósítása, párhuzamos architektúrával, pipeline alkalmazása nélkül, radián mértékegységben számolva, 11 bites regiszterezett be- és kimenetekkel, az eredmény levágással történő kerekítésével, 11 ciklusú iterációval számolja ki az eredményt, amely csak a szöghelyzet értéke. Ezzel elérhető lett, hogy a modul a százezer kaput tartalmazó Xilinx Spartan 3E FPGA logikai cella erőforrásainak csak 23%-át használja fel.

AZ IP CORDIC modul adatlapjából kiolvasható, hogy Arc Tan mód esetén a bemeneteknek -1 és +1 közé kell esniük, és a formátuma 1QN kell, hogy legyen, vagyis kettes komplemens ábrázolásban 2 egészrész bittel kell, hogy rendelkezésre álljanak. Ez a feltétel indokolja az előző modulok során választott ki- és bemeneti bitszámokat, valamint a számábrázolási formátumokat is. A modul kimenetként 2QN formátumú számot ad, vagyis kettes komplemens ábrázolásban 3 egészrész bittel, és a kimenet értéke $-\pi$ és $+\pi$ közötti, amely pontosan lefedi a teljes 360°-ot.

Az adatlapból az is kiolvasható, hogy a modul N bites bemenet esetén N órajel múlva szolgáltatja a kimeneti értéket, vagyis jelen esetben, a start jel 11 órajellel történő késleltetésével kialakítható a kész (ready) jelzés.

7.9. Elkészült modulok tesztelése

A fejlesztés folyamata során célszerű először az egyes elkészült modulokat külön-külön tesztelni a Xilinx iSIM szimulációs programmal. E fázis során az egyes modulokban akár mélyebb módosításokat is szükséges lehet alkalmazni, hogy azok kívánt eredményt szolgáltassák. Érdemes minden modul esetében végrehajtani ezt a fázist és figyelni a kimeneti jelformátumokat. A szimulációs program használatával könnyen megtalálhatók és javíthatók a hibák. A program segítségével a modul bemenő jeleit le lehet generálni, majd a program a megvalósított modul működésének szimulációjával kiadja a modul kimeneti jelformáját. A program segítségével az egyes belső változók állapotai is minden időpontban nyomon követhetők, így segítve a hibakeresést.

A tesztelés következő fázisa a modulok összeillesztése és együttes szimulációja. Ennek során a külön-külön már letesztelt modulok összeillesztésével, meg lehet győződni a teljes rendszer működőképességéről, valamint az esetlegesen itt felmerülő hibák még könnyebben javíthatók a szimulációs program adta lehetőségek miatt. Jól átgondolt generált bemeneti hullámformákkal a kritikus működési lépések helyes végrehajtását érdemes ellenőrizni, és hiba esetén módosítani a terveken.

Tesztelés utolsó fázisa az implementáció, amelynek során a helyesnek vélt kóddal fel kell programozni az FPGA áramkört. Ha az implementáció sikeres, akkor átgondolt lépések szerint, különböző teszteket kell végrehajtani a valós áramkörön és ellenőrizni annak helyes működését.

A modulok különálló tesztelését a jelfeldolgozási láncban hátulról kezdtem, ugyanis az atan2 függvényt megvalósító modul bemeneti követelményei a legszigorúbbak, és ehhez kell igazodni a többi modul megvalósításánál. Az előbbi megfontolások alapján a fejlesztés során minden modult leteszteltem ilyen módon.

Modulok összeillesztésekor a bemeneti jelformákat az analóg-digitális átalakító adatlapjában leírtak szerint gondoltam végig. A kimeneti hullámforma helyességéről a Xilinx IP CORDIC modul dokumentációja alapján győződtem meg. Az illesztőpanel elhúzódott gyártási ideje miatt lehetőségem volt a feldolgozási lánc minden kritikus lépését tesztelni.

Miután a panel elkészült az implementáció során csak a kommunikációs modulban volt szükség néhány változtatásra, a feldolgozási lánc elsőre működött. Az implementáció után egy mágnes kézi forgatásával tudtam tesztelni az eszköz működését, és a kijelzés segítségével nyomon követni az eredményeket. Természetesen a későbbiekben szükségesek lesznek magán a motoron végzett tesztek is, ahol a kijelzés helyett valamilyen egyéb mód szükséges az eredmények megmutatására, akár adattárolással is egybekötve a gyors működési sebesség érdekében.

8. A jelfeldolgozás redundánssá tétele

Az autóban a kormányzás, valamint a hozzá kapcsolódó rásegítés is biztonságkritikus funkció, ugyanis meghibásodása esetén emberi életek kerülhetnek veszélybe. A biztonságkritikus rendszereknek szigorúbb előírásokat kell teljesítenie az átlagos elvárásoknál. Az autóiparban egy pont meghibásodás kivédésére törekednek, ugyanis az első hibát és a hiba súlyosságát azonnal lehet jelezni a sofőrnek, és ha szükséges, akár azonnal meg is állíthatja a személygépjárművet. A szigorú előírások teljesítését azonban gyakran redundancia alkalmazásával lehet csak elérni.

A szervomotor szöghelyzetének érzékelése része a kormányzásrásegítésnek, emiatt ebben a részfunkcióban is megfelelő biztonsági szintet kell elérni, ami megfelelő redundanciával lehetséges. Ez jelen esetben azt jelenti, hogy a forgórész szöghelyzetét két mérőberendezéssel is mérni kell, külön-külön fel kell dolgozni mind a két érzékelő jelét, majd a kapott eredményeket egy bizonyos toleranciával össze kell vetni. Ha a két kapott érték között túl nagy az eltérés, az egyértelműen meghibásodást jelent valamelyik feldolgozási folyamatban. Ezt jelezni kell a vezérlő felé, amely majd dönt a hiba súlyosságáról, és a további biztonságos működés módjáról.

8.1. Redundancia az érzékelésben

Említettem, hogy maga a szenzor kifejezetten autóipari biztonságkritikus rendszerekbe ajánlott. Ennek oka, hogy a szenzor tokjában két érzékelő is található, egy GMR és egy AMR szenzor. Tehát, ha nem csak a GMR szenzor jele kerül felhasználásra, mint ahogy azt a szakdolgozat feladat készítése során történt, hanem az AMR jele is, akkor a kétszeres mérés kritériuma teljesíthető. Ráadásul ez a lépés további előnyökkel jár, ugyanis az AMR szenzor pontosabb, mint a GMR szenzor, így a nagyobb megbízhatóságon felül pontosabb eredmény is kapható az alkalmazásával.

Az AMR szenzorrész bekapcsolásához szükséges, hogy egy külön tápvezeték és egy külön földvezeték is bevezetésre kerüljön a szenzor megfelelő lábaira is az eddigi GMR részt bekapcsoló táp mellett. A külön táphozzávezetés a teljes redundancia miatt kell. Ha közös lenne a két szenzorrész tápja, az közös egypontú hibaforrás lenne, ugyanis egy esetleges táphiba esetén egyik érzékelő sem szolgáltatna jeleket, így teljes mértékben elveszne a forgórész szöghelyzetének jele. A külön táp nemcsak a panel

huzalozásában fontos, hanem a valóságban fizikailag különálló tápáramkörrel is kell meghajtani a két külön vonalat.

Emellett az illesztőpanelen további áramköri elemeket kell még elhelyezni, hogy az AMR szenzorrész által szolgáltatott jeleket fel lehessen dolgozni. A GMR bekötéshez hasonlóan, az AMR mind a négy analóg adatkimenetét megfelelően szűrni kell, és mind a négy jelet mintavételezni és digitalizálni kell 4 darab analóg-digitális átalakító segítségével. Végül ezek digitális jeleit ugyanúgy továbbítani kell az FPGA felé. Az AMR szenzorrész elektromos paraméterei egyeznek a GMR szenzorrész paramétereivel, így a teljes analóg jelfeldolgozást és a jelillesztés fizikai elemeit nem kell újratervezni, elég csak megismételni az illesztőpanelen. Az új alkatrészeknek mind az újonnan kialakítandó AMR szenzorrész tápját és földjét kell megkapniuk a különálló redundáns működés miatt.

A panelen található analóg-digitális átalakítók vezérlése is külön figyelmet igényel a redundáns viselkedés miatt. A GMR, valamint az AMR szenzorrészhez tarozó analóg-digitális átalakítók közös vezérlése esetén ugyanis közös egypontú hibaforrás keletkezne a rendszerben, vagyis ha bármelyik vezérlőjel megsérülne, akkor az összes, a panelen megvalósított funkció működésképtelen lenne. Az átalakítók vezérlőjelei az órajel (SCLK) és a kiválasztójel (CSn). Mindkét jelnél külön kivezetések kellenek a két szenzorrész átalakítóitól az FPGA felé.

Tehát arra van szükség, hogy a panelen a szenzor mindkét része és az hozzájuk tartozó négy-négy analóg-digitális átalakító megkapja a megfelelő tápot. Ezek után mind a nyolc analóg kimenetére egyformán kell egy-egy RC aluláteresztő szűrő, majd egy-egy analóg-digitális átalakító és a digitális jeleket egy-egy csatlakozósorra kell kivezetni. Ezek paraméterezése és az alkatrészek típusai azonosak lehetnek a szakdolgozat során megvalósítottakkal és használtakkal.

8.2. Redundancia a feldolgozásban

A jelfeldolgozás további lépései (kommunikáció az illesztőpanel felé, ofszetkorrekció, normalizálás, atan2 függvény), amelyek az FPGA-ban valósultak meg, lényegében teljes egészében duplikálni kellene a redundancia megvalósításához. Ugyanis az újonnan becsatolt, az AMR szenzorrész által szolgáltatott négy jelfolyam pontosan ugyanolyan, mint a GMR által szolgáltatott.

Az FPGA működési sebessége sokkal nagyobb, mint a 20 kHz-es mintavételezés, így képes lenne egymás után ugyanazon az útvonalon feldolgozni a rendelkezésre álló idő alatt mind a két szenzorrész által szolgáltatott jeleket. Hiába elég gyors működésű azonban az FPGA, mégsem járható ez az út több ok miatt is. Az egyik ok, hogy a kétféle szenzor koszinusz- és szinuszjelei között ofszeteltérés lehetséges, emiatt az ofszetkorrekciós modult eleve külön kellene választani a jeleknél. A másik ok pedig az, hogy ha a közös feldolgozási modulok bármelyikében keletkezne egy hiba, akkor az megint egypontú közös hiba lenne. Ugyanis mind a két jelfolyamot egyformán torzítaná, így mind a két érték hibás lenne, ráadásul ugyanolyan módon keletkezne a hiba, így az összehasonlításnál valószínűleg észre se lehetne venni a hibát.

E megfontolások miatt a teljes feldolgozási lánc összes modulját teljes egészében, még egyszer implementálni kell az FPGA áramkörbe a szükséges redundancia megvalósításához. Az AMR és a GMR szenzorrész jeleinek feldolgozása lényegében megegyezik. Egy ponton szükséges változtatni a szakdolgozat során elkészített feldolgozáson. A szenzor adatlapjából kiolvasható, hogy a GMR szenzorrész által szolgáltatott koszinusz- és szinuszkomponensekből atan2 függvény segítségével azonnal a szöghelyzet kapható meg. Az AMR szenzorrész által szolgáltatott jeleknél azonban ezzel a módszerrel a szöghelyzet kétszerese kapható meg. Vagyis a GMR rész feldolgozása maradhat ugyanaz, mint ami a szakdolgozat feladata során megvalósult. Az AMR rész feldolgozásánál azonban ugyanazokat a modulokat újra le kell másolni és implementálni, csak a feldolgozási lánc végén osztani kell a kapott szögértéket kettővel (ami bináris logikánál az érték léptetését jelenti eggyel).

Fontos elem a feldolgozási lánc egyes moduljait vezérlő központi ütemező. Ezt a modult is implementálni kell redundánsan, hogy a két különálló feldolgozási láncot két különálló ütemező vezérelhesse, így kerülve el a közös egypontú meghibásodás lehetőségét. Az nem elég azonban, ha a két ütemező egymástól teljesen különálló módon működik, hanem ezeknek ellenőrizni is kell egymást. Ez fontos, ugyanis ha az egyik feldolgozási lánc meghibásodik, akkor nincs is értelme a végén a két jelet összehasonlítani. Annak a megvalósítása, hogy a két központi ütemező egymást ellenőrizze, egyszerű módon, az ütemezők által szolgáltatott kész („ready”) jelzések összehasonlításával is könnyen megvalósítható. Ha szükséges újabb jelek bevezetésével nagyobb megbízhatóságú ellenőrzési mechanizmusok is megvalósíthatók. Hiba esetén a hibajelzésről is gondoskodni kell.

8.3. Összehasonlítás toleranciával

A redundancia megvalósításának utolsó lépése a két úton számított értékek összehasonlítása. Ha a két külön úton érzékelt és feldolgozott jel eredménye hasonló, akkor a mért érték megfelelő, ha azonban az eltérés jelentős, akkor hibásnak kell tekinteni a kapott értékeket. Nem kívánunk meg az aktuálisan kapott eredményektől teljes egyenlőséget, ugyanis a kétféle szenzor pontossága nem azonos, ráadásul az sem biztosított, hogy a két szenzor jeléből azonos időpontban történik a mintavétel, és az analóg jelutakban különböző alkatrészek találhatók, amelyek nem tökéletesen egyformák. Az ezekből adódó eltéréseket még nem szabad hibának érzékelni.

Első lépésként a két szenzor által szolgáltatott jel értékét egyeztetni kell az összehasonlításhoz. Ugyanis az AMR szenzor csak 0-180° között, míg a GMR szenzor a teljes 0-360° között szolgáltat értéket. Vagyis amikor a GMR szenzor 180-360° közötti értéket mér, akkor az AMR szenzor biztosan újból 0-180° közötti értéket mutat. Tehát ilyen esetben az AMR eredményéhez hozzá kell adni 180°-ot, hogy egyáltalán összehasonlítható legyen a két feldolgozási útvonalon kapott eredmény.

Az eredményeket azonban ezeken túl is egy bizonyos toleranciával kell összehasonlítani. A GMR szenzor pontossága összességében ±3° maximálisan a szögeltérés végeredményére vetítve, míg az AMR szenzor pontossága ±1.7° maximálisan. Így legrosszabb esetben a szenzorok hibái miatti maximális eltérés 4.7° lehet. Ezt az értéket azonban még felfelé kell kerekíteni, ugyanis nem garantált a különválasztott vezérlőjelek miatt, hogy egyszerre történik a mintavételezés a kétféle szenzor analóg kimenetein, valamint az analóg jelútban lévő szűrők alkatrészeinek toleranciája miatt is lehet eltérés az értékekben. Szerencsére az analóg jelek kisfrekvenciásak, így ezek a hibakomponensek már nem ilyen jelentősek. Emiatt összességében ±5°-os különbség megengedhető maximálisan a két jel között. Ez még nem okoz különösebb problémát a mezőorientált szabályozásban sem.

Az AMR szenzor szolgáltatja a pontosabb jelértéket, emiatt érdemes ezt az értéket eredménynek tekinteni – ha megfelelően hozzáadtuk vagy sem a 180°-ot – kiegészítve ezzel a működést a teljes körülfordulásra. Így az előbb leírt megfelelő összehasonlítás után egyértelműen eldönthető az eredményül kapott szögértékről, hogy helyes-e vagy esetleg valamilyen hiba van az érzékelésében. Utóbbi esetben gondoskodni kell a megfelelő hibajelzésről is a vezérlő felé.

9. Későbbi fejlesztési lehetőségek

A szakdolgozat elkészítés elején röviden feltérképeztem a modern személygépjárművekben alkalmazott kormányzásrásegítési módszereket, beleértve az alapvető mechanikai konstrukciókat, illetve a rásegítéshez alkalmazott motortípusokat. Ezek után részletesebben tanulmányoztam a mezőorientált szabályozást. Betekintést nyertem a kormányzásrásegítésnél alkalmazott teljes szabályozási körbe, és az egyes részegységek főbb feladataiba is.

Ezen ismeretek segítségével átláthatóvá vált számomra, hogy e témakörön belül pontosan hol is helyezkedik el a motor szöghelyzetének érzékelése, amely a szakdolgozatom témája volt. A feladat megvalósítása során megismertem az analóg szögszenzor mérési elvét, és az alkalmazott szenzortípus paramétereit is. A kutatási feladat megvalósítása két fejlesztési fázisra osztható. Az egyik az illesztőpanel megtervezése, melynek során nem merültek fel nagyobb akadályok, azonban a panel gyártása kissé elhúzódott. Szerencsére maradt idő a panel és az FPGA közötti kommunikáció élesztésére és a teljes rendszer implementációjára. A másik fázis a szöghelyzet értékének kialakítása az FPGA áramkörön amelyet, egymást követő modulok láncba fűzésével oldottam meg. A feldolgozási láncon belül külön nehézséget okozott egyrészt, az adatátvitelhez szükséges kommunikációs modul soros adatfeldolgozása, másrészt az atan2 modul részben hiányos dokumentációja.

Feladatom volt továbbá a redundancia megvalósíthatóságának vizsgálata, hogy a biztonságkritikus követelményeket teljesítse a rendszer. A megvalósításhoz szükséges alapelveket a szakdolgozatban leírtam, ha a szükséges eszközök rendelkezésre állnak, akkor kevés munkával ez a fejlesztési lépés is megoldható.

A feladatkiírásban megfogalmazott elvárásoknak megfelelően, megismerkedtem és bemutattam az alkalmazott szögszenzor felépítését, elkészült az illesztőpanel, amely továbbítja a szenzor jeleit digitális formában az FPGA felé, implementáltam az FPGA áramkörbe a szenzor GMR rész koszinusz- és szinuszjeleinek feldolgozását. Legvégül a szimulációs tesztek után, el tudtam végezni a teljes feldolgozási láncot megvalósító kész áramkörökön az első méréseket is. Fejlesztési végeredményként elkészült az szögszenzor jelének illesztése és feldolgozása egy áramköri panel és egy FPGA együttes rendszereként.

Egy fejlesztési lehetőség az alkalmazott szenzor minkét részének felhasználása a feldolgozási láncban a leírt megvalósíthatósági vizsgálatok alapján. Az így elkészült szöghelyzet-érzékelő rendszeren először kísérleti motorteszteket is érdemes végrehajtani. Ha a tesztek alapján a rendszer hatékonyabb és gyorsabb beavatkozást tesz lehetővé, mint az eddigi rendszer, és ha a költséghatékonysági kritériumoknak is megfelel, akkor érdemes lehet a terveket – egy illesztőpanel és FPGA együttese helyett – teljes egészében közvetlenül ASIC, (application-specific integrated circuit) áramkörön megvalósítani. Ilyenkor néhány külső alkatrész elhagyható lenne (pl.: a tűsor és kivezetések), valamint a drága FPGA áramkört le lehetne cserélni a sorozatgyártásban olcsóbb egyedi tervezésű áramkörre.

A mezőorientált szabályozás leírásánál is említettem, hogy a szöghelyzet érzékelése csupán egy kis része csak a teljes szabályozási körnek. Fontos további elem még a végfokok meghajtása, vagyis a beavatkozás lépése. A szabályozási kör e részének FPGA áramkörre történő implementálását szintén szakdolgozat feladatként kiadta a ThyssenKrupp Presta Hungary Kft, így ha az is elkészül, akkor a két rendszer együttesen implementálható egy FPGA áramkörben, és ilyenkor már csak a szabályozás algoritmusát kell külön mikrokontrolleren futtatni.

Magát a motorszabályozást is érdemes lehet áttenni FPGA áramkörbe, hiszen ezzel a futási idő jelentősen csökkenthető, ezáltal a beavatkozás is gyorsabban megvalósulhat. Így lehetővé válhat a holtidők miatti hibák csökkentése, emiatt pontosabb beavatkozásra is képes lehet együttesen a rendszer. Azonban ennek megvalósítása igen sok feladatot jelent, rengeteg áramköri elemet igényelne, emiatt itt már a költséghatékonyság is fontos szerepet játszana.

Irodalomjegyzék

[1] United Nations Economic Commission for Europe UN Vehicle Regulations: *Regulation No. 79 - Rev.2 - Steering equipment, Section 2.6.3*, http://www.unece.org/fileadmin/DAM/trans/main/wp29/wp29regs/r079r2e.pdf (megtekintés: 2014. December 07. 16:58)

[2] International Electrotechnical Commission: *IEC 61508 - Functional Safety*, http://www.iec.ch/functionalsafety/ (megtekintés: 2014. December 07. 16:58)

[3] International Organization for Standardization: *ISO 26262 - Road vehicles — Functional safety, 2011*, https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-1:v1:en (megtekintés: 2014. Decemeber 07. 16:58)

[4] Duka Gyula, Keller Ervin, Kiss István, Virágh Sándor.: *A járművezetői vizsga tankönyve B*, ISBN 963-89201-6-1, BertelsmassSpringer Magyarország Kft Transport Média Diviziója, 2002, 379-382. oldalak

[5] Wikipedia: *Power Steering*, http://en.wikipedia.org/wiki/Power_steering (megtekintés 2014. Október 21. 15:03)

[6] CDX Online E-Texbook: *Electric power assisted steering*, http://www.cdxetextbook.com/steersusp/steer/boxesColumns/elecpwrassist.html (megtekintés 2014 Október 21, 15:03)

[7] Dave Wilson (Texas Instruments): *Motor Control Compendium 2010-2011*, http://focus.ti.com/docs/training/catalog/events/event.jhtml?sku=OLT210201 (megtekintés 2014. Október 26. 17:53)

[8] Dr. Puklus Zoltán, Dr. Szénásy István: *Villamos hajtások*, Széchenyi István Egyetem, 1. modul: 1-4. lecke, http://www.tankonyvtar.hu/hu/tartalom/tamop425/0013_13_Villamos_hajtasok-hu/index.html (megtekintés 2014. Október 26. 17:53)

[9] Halász Sándor: *Villamos hajtások*, ISBN 963 450 5171 , ROTEL Kft, 1993, 287-345. oldalak

[10] J. Rais, M. P. Donsión: *Permanent Magnet Syncgronous Motors (PMSM). Parameters influence on thy synchronization process of PMSM*, http://www.icrepq.com/icrepq-08/409-rais.pdf, (megtekintés 2014. Október 27. 18:37)

[11] Open Electrical Wiki: *Clarke Transform*, http://www.openelectrical.org/wiki/index.php?title=Clarke_Transform, (megtekintés 2014. Október 27. 18:37)

[12] NVE Corporation: *How GMR Works*, http://www.gmrsensors.com/gmr-operation.htm , (megtekintés 2014. November 1. 11:38)

[13] Wikipedia: *Magnetoresistance*, http://en.wikipedia.org/wiki/Magnetoresistance , (megtekintés 2014. November 1. 11:38)

[14] NVE Corporation: *Application Notes for GMR Sensors*, http://www.nve.com/Downloads/apps.pdf, (megtekintés 2014. November 1. 11:38)

[15] Upverter tervezőprogram, https://upverter.com/ (megtekintés: 2014. December 09. 8:42)

[16] Srinivasa Murthy H N, Roopa M: *FPGA Implementation of Sine and Cosine Generators using CORDIC Algorithm*, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-1, Issue-6, November 2012

[17] BME MIT: *CORDIC*, http://www.mit.bme.hu/system/files/oktatas/targyak/8497/CORDIC_ppt.pdf (megtekintés: 2014. November 29. 18:02)

Ábrajegyzék

[1] Wikipedia: *Steering System*, http://en.wikipedia.org/wiki/File:Steer_system.jpg (megtekintés: 2014. Október 21, 15:03)

[2] Mitsubishi electric: *EPS Motor*, http://www.mitsubishielectric.com/company/environment/policy/product/automation/eps/images/img_01.gif (megtekintés: 2014. Október 21, 15:03)

[3] Team-bhp forum: *How does electric power steering work topic*, http://www.team-bhp.com/forum/attachments/technical-stuff/903093d1331789249-how-does-electric-power-steering-work-tu9.jpg (megtekintés: 2014. Október 21, 15:03)

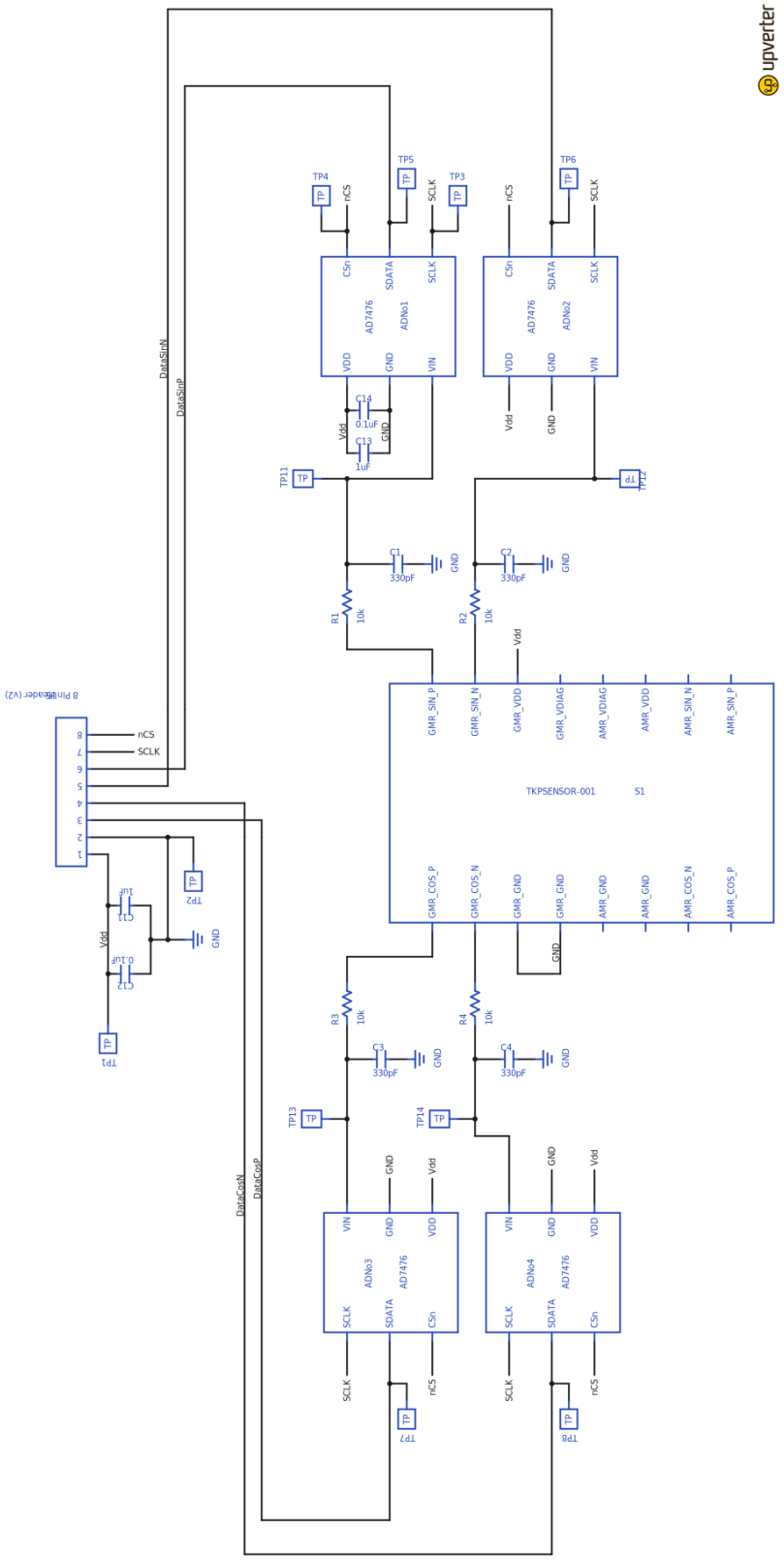
[4] Dave Wilson (Texas Instruments): *Motor Control Compendium 2010-2011*, http://focus.ti.com/docs/training/catalog/events/event.jhtml?sku=OLT210201 (megtekintés: 2014. Október 26, 17:53)

[5] Design World: *Choose Your Best Pneumatic Cylinder Sensor Here*, http://www.designworldonline.com/uploads/Imagegallery/gmr-technology.jpg, (megtekintés: 2014. November 1. 11:38)

[6] NVE Corporation: *Application Notes for GMR Sensors*, http://www.nve.com/Downloads/apps.pdf, (megtekintés 2014. November 1. 11:38)

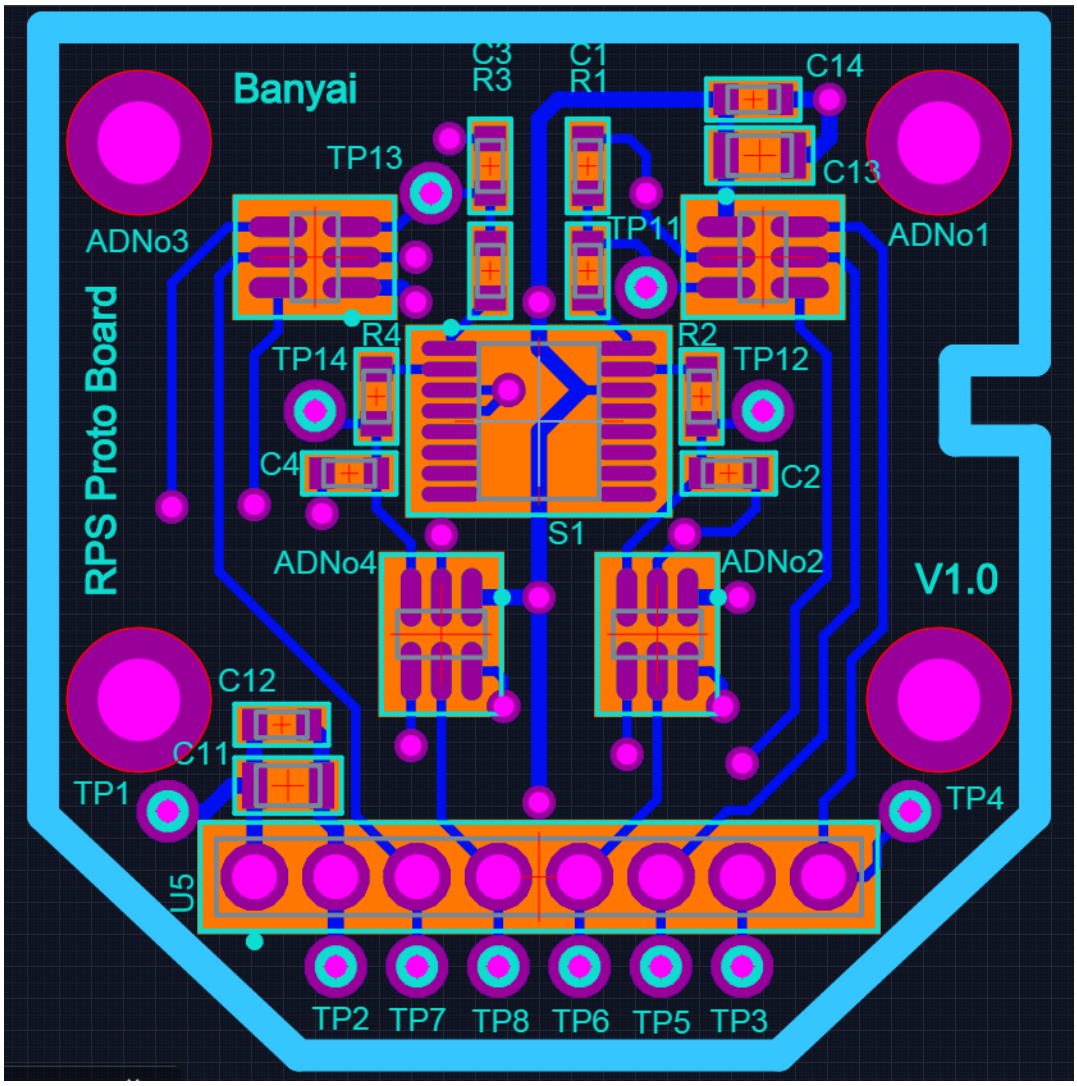
Függelék

A megtervezett illesztőpanel kapcsolási rajza

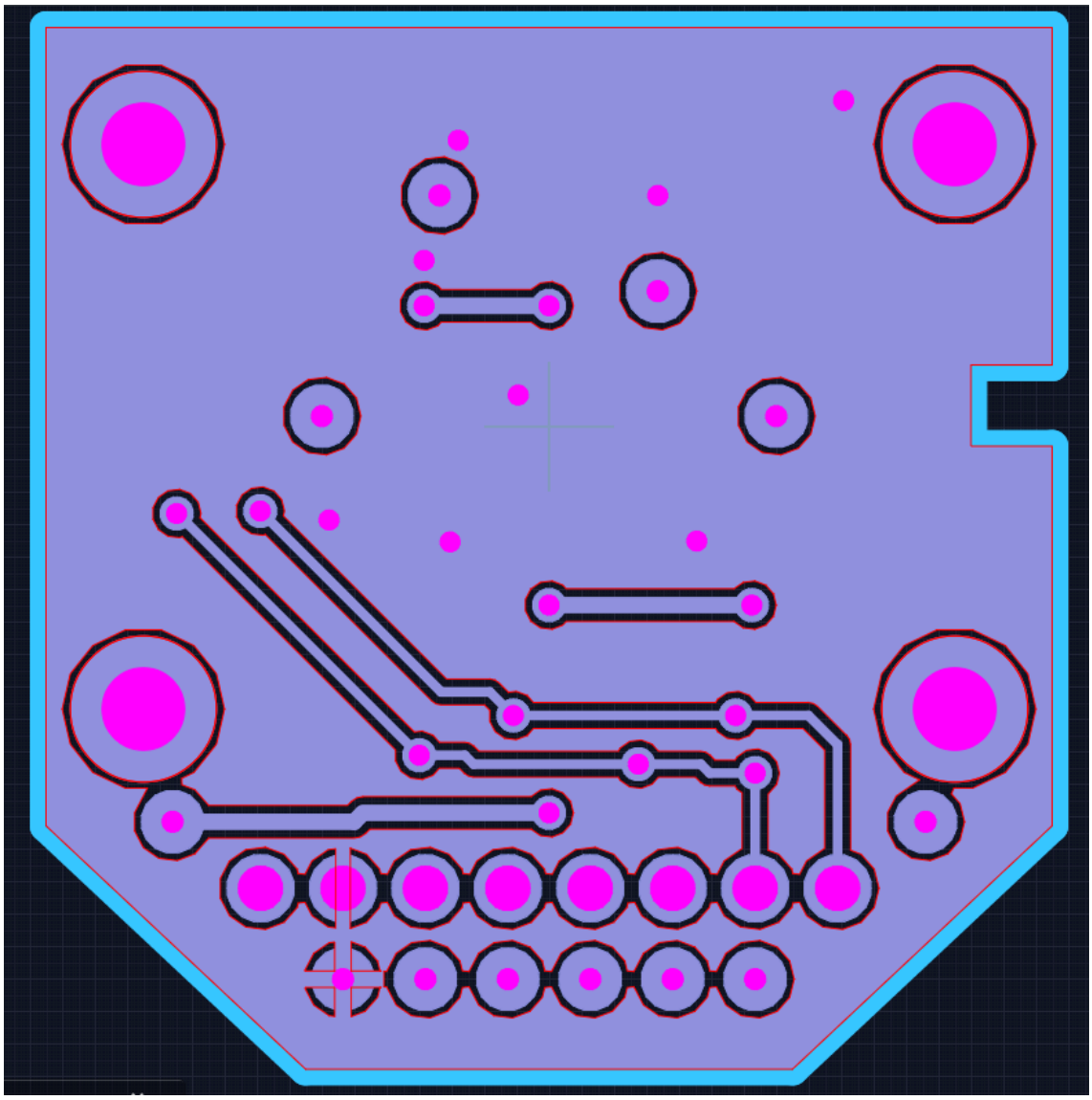


A megtervezett illesztőpanel áramköri rajzolata

Az alábbi ábrákon látható az elkészült illesztőpanel áramköri elrendezése. A két képen a kétoldalas nyomtatott áramköri lemez felső és alsó oldali rajzolata látható külön-külön. Az ábrákon a világoskék szín jelöli a panel mechanikai szélét, rózsaszín a furatokat, sötétkék a felsőoldali rézréteget (vezetékezés), sötétlila a felsőoldali forrasztásgátló lakk ablakrétegét, narancssárga az alkatrészek – kivezetések miatti – felületigényét, a szürke keret az alkatrészek fizikai méretét és a világoslila szín pedig az alsó oldali rézréteget (vezetékezés).

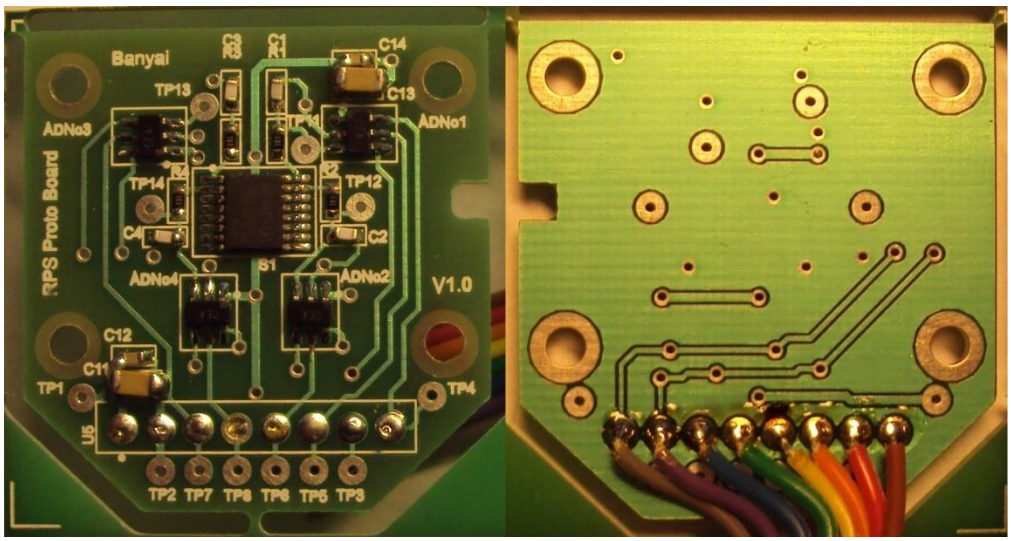


F1 ábra: A panel felsőoldali (top) rajzolata



F2 ábra: A panel alsóoldali (bottom) rajzolata

Az elkészült illesztőpanel:



A feldolgozó modulokat leíró kód

A Xilinx Spartan 3E-100 FPGA-ba implementált feldolgozási lánc moduljait megvalósító kódok lentebb olvashatók. A modulok leírása Verilog nyelven történt. Tanulmányozásuk megkönnyítheti a 7.4.-7.8. alfejezetekben leírt működés megértését. Az egyes modulok kódjai egymás után ezen alfejezetekben elmagyarázott működés sorrendjében olvashatók.

Központi ütemező (CentralTimer.v):

```
module CentralTimer(
   input clk,                  //input:from FPGA
   input rst,
   input ADComReady,           //input:from modules
   input ConvertReady,
   input Division1Ready,
   input Division2Ready,
   input Atan2Ready,
   output StartADCom,          //output:to control modules
   output StartConvert,
   output StartDiv1,
   output StartDiv2,
   output StartAtan,
   output FullComplete         //output:indicates the all conversation ready
    );

   /* Registers */
   reg regStartADCom;          //register for storing output data
   reg regStartConvert;        //register for storing output data
   reg regStartDiv;            //register for storing output data
   reg regStartAtan;           //register for storing output data
   reg regFullComplete;        //register for storing output data
   reg [2:0] state;            //register for store that which state are in
                               //state=0: reseted;
                               //state=1: Communicating with AD;
                               //state=2: converting inputs;
                               //state=3: dividing;
                               //state=4: doing atan2;

   /* Clockdivision for 20kHz */
   /* The sampling will be at this frequency */
   reg [11:0] cntr;
   wire ce;
   always @(posedge clk)
       begin
           if(rst|ce)
               cntr <= 0;
           else
               cntr <= cntr + 1;
       end
   assign ce = (cntr == 2499);    //for 20kHz output -> cntr[11:0] needed
   //assign ce = (cntr == 49999999);//for 1Hz output -> cntr[25:0] needed
   //assign ce = (cntr == 30);    //for iSim
   /* CE MUST BE MORE RARE THAN 3+10+11 CLK!!!*/
```

```
/* main function: sending start signals to modules as a state machine*/
always @(posedge clk)
    begin
        if(rst)                                 //at reset no starting
            begin
                regStartADCom <= 0;
                regStartConvert <= 0;
                regStartDiv <= 0;
                regStartAtan <= 0;
                regFullComplete <= 0;
                state <= 0;             //state=0: reseted
            end
        else if((ce) && (state == 0))

            begin
                regStartADCom <= 1;
                regStartConvert <= 0;
                regStartDiv <= 0;
                regStartAtan <= 0;
                regFullComplete <= 0;
                state <= 1;             //state=1: Comminicating with AD
            end
        else if((ADComReady) && (state == 1))
            begin
                regStartADCom <= 0;
                regStartConvert <= 1;
                regStartDiv <= 0;
                regStartAtan <= 0;
                regFullComplete <= 0;
                state <= 2;             //state=2: Converting inputs
            end
        else if((ConvertReady) && (state == 2))
            begin
                regStartADCom <= 0;
                regStartConvert <= 0;
                regStartDiv <= 1;
                regStartAtan <= 0;
                regFullComplete <= 0;
                state <= 3;             //state=3: dividing
            end
        else if((Division1Ready) && (Division2Ready) && (state == 3))
            begin
                regStartADCom <= 0;
                regStartConvert <= 0;
                regStartDiv <= 0;
                regStartAtan <= 1;
                regFullComplete <= 0;
                state <= 4;             //state = 4: doing atan2
            end
        else if((Atan2Ready) && (state == 4))
            begin
                regStartADCom <= 0;
                regStartConvert <= 0;
                regStartDiv <= 0;
                regStartAtan <= 0;
                regFullComplete <= 1;
                state <= 0;         //processing finished, back to reset
            end
```

```
            else if((state == 1)||(state == 2)||(state == 3)||(state == 4))
                begin //achieve that start signal are high only for one clk.
                    regStartADCom <= 0;
                    regStartConvert <= 0;
                    regStartDiv <= 0;
                    regStartAtan <= 0;
                    regFullComplete <= 0;
                end
        end

    /*Drive outputs*/
    assign StartADCom = regStartADCom;
    assign StartConvert = regStartConvert;
     assign StartDiv1 = regStartDiv;
    assign StartDiv2 = regStartDiv;
    assign StartAtan = regStartAtan;
    assign FullComplete = regFullComplete;

endmodule
```

Kommunikációs modul (ADCommunication.v):

```
module ADCommunication(
    input clk,
    input rst,
    input start,            //in from FPGA start
    output CSn,             //out to ADC CSn
    output SCLK,            //out to ADC SCLK
    input DATASINP,         //in from ADC
    input DATASINN,         //in from ADC
    input DATACOSP,         //in from ADC
    input DATACOSN,         //in from ADC
    output rdy,             //out to FPGA: comm&data ready
    output [11:0] SinFinP, //out to FPGA: sinPvalue
    output [11:0] SinFinN, //out to FPGA: sinNvalue
    output [11:0] CosFinP, //out to FPGA: cosPvalue
    output [11:0] CosFinN  //out to FPGA: cosNvalue
     );

    /*Clock division: Provide 12.5 MHz -> ADC OK + 4x division from clk*/
    reg clkcntr;
    reg rSCLK;
    always @(posedge clk)
        begin
            if(rst)
                clkcntr <= 0;
            else
                clkcntr <= clkcntr + 1;
        end
    always @(posedge clk)
        begin
            if(rst)
                rSCLK <= 1;
            else if(clkcntr == 1)
                rSCLK <= ~rSCLK;
        end
    assign SCLK = rSCLK;

    /*Provide CSn for AD*/
```

```
reg wasStart;
reg rCSn;
reg [4:0] state;
reg lastSCLK;
// remember last sclk, because CSn must fall after posedge SCLK
always @(posedge clk)
    begin
        if(rst)
            lastSCLK <= 1;
        else
            lastSCLK <= SCLK;
    end

//create nCS: fall after a posedge SCLK, rise after 16th state = 16 SCLK
always @(posedge clk)
    begin
        if(rst)
            begin
                wasStart <= 0;
                rCSn <= 1;
            end
        else if(start)
            begin
                wasStart <= 1;
            end
        else if(wasStart &&(SCLK == 1)&&(lastSCLK == 0)&&(state == 0))
            begin
                rCSn <= 0;
            end
        else if((state == 16) && (SCLK == 1))
            begin
                rCSn <= 1;
                wasStart <= 0;
            end
    end
assign CSn = rCSn;

//create state: counting falling edges on sclk duiring a CS
always @(negedge SCLK)
    begin
        if(CSn == 1)
            begin
                state <= 0;
            end
        else
            begin
                state <= state + 1;
            end
    end

/*Reading data inputs*/
reg [14:0] rSinInP;
reg [14:0] rSinInN;
reg [14:0] rCosInP;
reg [14:0] rCosInN;
always @(posedge SCLK)
    begin
        if(CSn == 1)
            begin
```

```
                rSinInP <= 0;
                rSinInN <= 0;
                rCosInP <= 0;
                rCosInN <= 0;
            end
        else if(state < 16)
            begin
                rSinInP <= {rSinInP[13:0] , DATASINP};
                rSinInN <= {rSinInN[13:0] , DATASINN};
                rCosInP <= {rCosInP[13:0] , DATACOSP};
                rCosInN <= {rCosInN[13:0] , DATACOSN};
            end
    end
  /*Remember inpust until next CSn and create ready signal*/
  reg [11:0] rSinFinP;
  reg [11:0] rSinFinN;
  reg [11:0] rCosFinP;
  reg [11:0] rCosFinN;
  reg ready;
  always @(posedge clk)
    begin
        if(rst)
            begin
                rSinFinP <= 0;
                rSinFinN <= 0;
                rCosFinP <= 0;
                rCosFinN <= 0;
                ready <= 0;
            end
        else if(start)
            begin
                ready <= 0;
            end
        else if((state == 15) && (SCLK == 1))
            begin
                rSinFinP <= rSinInP[11:0];
                rSinFinN <= rSinInN[11:0];
                rCosFinP <= rCosInP[11:0];
                rCosFinN <= rCosInN[11:0];
                ready <= 1;
            end
    end

  /*Drive outputs*/
  assign rdy = ready;
  assign SinFinP = rSinFinP[11:0];
  assign SinFinN = rSinFinN[11:0];
  assign CosFinP = rCosFinP[11:0];
  assign CosFinN = rCosFinN[11:0];

endmodule
```

Ofszetkorrekciós modul (ConvertSensorData.v):

```
module ConvertSensorData(
  input clk,
  input rst,
  input start,           //in: 1 bit to show that process should start
  input [9:0] sinP,      //in: 10 bit unsigned integer from ADC
```

```
input [9:0] cosP,       //in: 10 bit unsigned integer from ADC
input [9:0] sinN,       //in: 10 bit unsigned integer from ADC
input [9:0] cosN,       //in: 10 bit unsigned integer from ADC
output [10:0] sin,      //out: 11 bit signed integer number:
                        //1sing+10integer showing current sin value
output [10:0] cos,      //out: 11 bit signed integer number:
                        //1sing+10integer showing current cos value
output [10:0] sinAmpl, //out: 11 bit signed integer number:
                    //1sing+10integer showing sin's measured amplitude
output [10:0] cosAmpl, //out: 11 bit signed integer number:
                    //1sing+10integer showing cos' measured amplitude
output dataReady    //out: 1 bit for show that all output data is ready
 );

/* Registers for store sensors preprocessed values */
reg [10:0] sinPreg;    //Register inputs as signed integer
reg [10:0] cosPreg;
reg [10:0] sinNreg;
reg [10:0] cosNreg;

reg [10:0] sinOffset;  //storing sampled neg-sin and pos-sin offset
reg [10:0] cosOffset;  //storing sampled neg-cos and pos-cos offset
reg [10:0] realSin;    //store the real sin value without offset
reg [10:0] realCos;    //store the real cos value without offset
reg [10:0] lastAmplSin;//store the sin's amplitude (when cos = 0)
reg [10:0] lastAmplCos;//store the cos's amplitude (when sin = 0)

reg signSin; //store the sign bit of the last realSin, for 0-transition
reg signCos; //store the sign bit of the last realCos, for 0-transition

reg [1:0] delayDone;   //counter for delaying the start signal

/* register inputs */
always @(posedge clk)
    begin
        if(rst)                    //reset all to default
            begin
                sinPreg <= 512;    //P&N can’t be equal at rst ->
                cosPreg <= 512;    //realsin&realcos would be 0 ->
                sinNreg <= 510;    //realsin&realcos can’t be 0 at rst->
                cosNreg <= 510;    //Overwriting LastAmp is not OK*/
            end
        else if(start)
            begin
                sinPreg <= {1'b0, sinP};
                cosPreg <= {1'b0, cosP};
                sinNreg <= {1'b0, sinN};
                cosNreg <= {1'b0, cosN};
            end
    end

/* main functionality to calculate the real values from inputs */
always @(posedge clk)
    if(rst)
        begin
            sinOffset <= 511;  //10 bit-> 1024 -> half: 512
            cosOffset <= 511;
            realSin <= 1;      //realsin&realcos cannot be 0 at rst ->
            realCos <= 1;      //Overwrites LastAmp is not OK*/
```

```
        end
    else
        begin
            /* calculate offset */
            sinOffset <= (sinPreg+sinNreg)/2;         // /2 is handled
            cosOffset <= (cosPreg+cosNreg)/2;
            /* calculate real values */
            realSin <= sinPreg-sinOffset;
            realCos <= cosPreg-cosOffset;
        end

/* remember realCos and realSin sign bit */
always @(posedge clk)
    if(rst)
        begin
            signSin <= 0;
            signCos <= 0;
        end
    else
        begin
            signSin <= realSin[10];
            signCos <= realCos[10];
            /*MSB indicates the sing of a 2-complement number*/
        end

/* remember the amplitude when the other trig has a zero-transition*/
always @(posedge clk)
    begin
        if(rst)                //3.3V=Vdd on ADC and 1.05V a main -> 326
            begin
                lastAmplSin <= 326;    //((1.05/3.3)*1024)
                lastAmplCos <= 326;
            end
        else
            begin
                //zero-transition on sin -> remember cos amplitude
                if((signSin != realSin[10]) || (realSin == 0))
                    begin
                        //the cos' ampl is positive -> just remember
                        if(realCos[10] == 0)
                            begin
                                lastAmplCos <= realCos;
                            end

                        //the cos' amplitude is negative -> convert it
                        //to positive and remember it
                        else
                            begin
                                lastAmplCos <= ((~realCos)+1);
                            end
                    end
                //zero-transition on cos -> remember sin amplitude
                if((signCos != realCos[10]) || (realCos == 0))
                    begin
                        //the sin's ampl is positive -> just remember
                        if(realSin[10] == 0)
                            begin
                                lastAmplSin <= realSin;
                            end
```

```
                        //the sin's amplitude is negative -> convert it
                        //to positive and remember it
                        else
                            begin
                                lastAmplSin <= ((~realSin)+1);
                            end
                    end
            end
        end

    /* Create DataReady signal */
    // DataReady is the start signal delayed with 3 clk.
    always @(posedge clk)
        begin
            //at rst was'nt start signal, counter 0 indicating reset status
            if(rst)
                begin
                    delayDone <= 0;
                end
            //at start signal, remember it, and start delay counter
            else if(start)
                begin
                    delayDone <= 1;
                end
            //after a start signal count up
            else if((delayDone != 0) && (dataReady != 1))
                begin
                    delayDone <= delayDone + 1;
                end
        end
    assign dataReady = ((delayDone == 3) && (start == 0));


    /* Drive the outputs */
    /* Create correct format:
            1 singbit + integer value (in positive) in 10 bit */
    assign sin[10] = ((realSin[10] == 1) ? 1'b1 : 1'b0);
    assign sin[9:0] = ((realSin[10] == 1) ? ((~realSin)+1) : realSin[9:0] );

    assign cos[10] = ((realCos[10] == 1) ? 1'b1 : 1'b0);
    assign cos[9:0] = ((realCos[10] == 1) ? ((~realCos)+1) : realCos[9:0] );

    assign sinAmpl[10] = ((lastAmplSin[10] == 1) ? 1'b1 : 1'b0);
    assign  sinAmpl[9:0]  =  ((lastAmplSin[10]  ==  1)  ?  ((~lastAmplSin)+1)  :
lastAmplSin[9:0] );

    assign cosAmpl[10] = ((lastAmplCos[10] == 1) ? 1'b1 : 1'b0);
    assign  cosAmpl[9:0]  =  ((lastAmplCos[10]  ==  1)  ?  ((~lastAmplCos)+1)  :
lastAmplCos[9:0] );

endmodule
```

Normalizáló modul (Division.v):

```
module Division(
    input [10:0] dividend,      //input: 1sign bit + 10 integer bit
    input [10:0] divisor,       //input: 1sign bit + 10 integer bit
    input start,                //input: start div
    input clk,
```

```
input rst,
output [10:0] quotient,
                    //output: 1sign bit + 1 integer bit + 9 fraction bit
output complete             //output: indicates that division is ready
);

/*Registers for storing the inputs, inner variables, and last output*/
reg [10:0] quotient_copy;  //store the result's bit while calculating it
reg [19:0] dividend_copy;  //store the dividend input while calculating
reg [19:0] divider_copy;   //store the divider input while calculating

reg [5:0] bit;             //store the position where just calculating
reg done;                  //showing if calculation is done
reg wasStart;              //show that at reset teh data is not valid

/*main part of the calculation*/
always @( posedge clk )
begin
    //At reset event clear all inner registers
    if(rst)
        begin
            quotient_copy <= 0;
            dividend_copy <= 0;
            divider_copy <= 0;
            bit <= 0;
            done <= 1;
            wasStart <= 0;
        end

    //At first clk read the inputs to inner registers, and set done to
    //undone, and set position to the top, and calculate sing value
    if( done && start )
        begin
            done <= 1'b0;
            wasStart <= 1;
            bit <= 9;
            quotient_copy <= 0;

            dividend_copy[19] <= 0;
            dividend_copy[18:9] <= dividend[9:0];
            dividend_copy[8:0] <= 0;

            divider_copy[19] <= 0;
            divider_copy[18:9] <= divisor[9:0];
            divider_copy[8:0] <= 0;

            //set sign bit
            if((dividend[10] == 1 && divisor[10] == 0) || (dividend[10]
== 0 && divisor[10] == 1))
                quotient_copy[10] <= 1;
            else
                quotient_copy[10] <= 0;
        end

    //From second clk to finish substract the divider from devidend if
    //it's greater and alays shift(reduce) the dividor,
    //and always check the position, and reduce it
    else if(!done)
        begin
```

```
                //compare divisor/dividend
                if(dividend_copy >= divider_copy)
                    begin
                        //subtract
                        dividend_copy <= dividend_copy - divider_copy;
                        //set quotient
                        quotient_copy[bit] <= 1'b1;
                    end

                //reduce divisor
                divider_copy <= divider_copy >> 1;

                //stop condition
                if(bit == 0)
                    begin
                        done <= 1'b1;
                    end

                //reduce bit counter
                bit <= bit - 1;
            end
    end

    /*Drive outputs*/
    //quotient should be between +1 and -1, so if the result over this cut
    //down. Otherwise the previous detection has errors, and atan modul
    //should get numbers like the above

    assign quotient = ((quotient_copy[9] == 1) ? {quotient_copy[10],
10'b1000000000} : quotient_copy);

    assign complete = (done && wasStart && (start == 0));

endmodule
```

Atan2 modul (Atan2Wrap.v):

Maga az atan2 modul egy Xilinx IP modul, amelynek csak a paraméterei állíthatók, emiatt ezt a modult körülfoglalja az Atan2Wrap modul, amely biztosítja a megfelelő be és kimeneteket.

```
module Atan2Wrap(
    input clk,
    input rst,
    input start,
    input [10:0] xCoordinate,
    //input format: 1sign bit + 1 integer bit + 9 fraction bit
    input [10:0] yCoordinate,
    //input format: 1sign bit + 1 integer bit + 9 fraction bit
    output [10:0] angle,
    //output format: 3 integer bit + 8 fraction bit in 2complement format
    output ready
    //output indicating that angle is ready and stable
     );

    /* Registers */
    //storing input as a 2-complement real number (2integer + 9fraction)
```

```
   reg [10:0] xCoordReal;
   reg [10:0] yCoordReal;

   /* Register inputs in a correct form */
   always @(posedge clk)
       begin
           if(rst)                        //at reset inputs are 1
               begin
                   xCoordReal <= 1;
                   yCoordReal <= 1;
               end
           else if(start)                 //at start remember inputs
               begin
                   xCoordReal[10] <=((xCoordinate[10] == 1) ? 1'b1 : 1'b0);

                   xCoordReal[9:0]    <=    ((xCoordinate[10]    ==    1)    ?
((~xCoordinate[9:0])+1) : xCoordinate[9:0] );

                   yCoordReal[10] <=((yCoordinate[10] == 1) ? 1'b1 : 1'b0);

                   yCoordReal[9:0]    <=    ((yCoordinate[10]    ==    1)    ?
((~yCoordinate[9:0])+1) : yCoordinate[9:0] );
               end
       end

   /* Main function is: atan2 functionality */
   Atan2  domath  (.x_in(xCoordReal),  .y_in(yCoordReal),  .phase_out(angle),
.clk(clk));

   /* counting for delay -> create ready signal */
   reg[3:0] cntr;
   always @(posedge clk)
       begin
           //at rst no start signal, counter is 0 indicating reset status
           if(rst)
               begin
                   cntr <= 0;
               end
           //at start signal, remember it, and start delay counter
           else if(start)
               begin
                   cntr <= 1;
               end
           //after a start signal count up
           else if((cntr != 0) && (ready != 1))
               begin
                   cntr <= cntr + 1;
               end
       end
   assign ready = ((cntr == 11) && (start == 0));

endmodule
```