Debreceni Egyetem

Informatikai Kar

FPGA alapú robotkarvezérlés megvalósítása

Témavezető:

Dr. Végh János

Egyetemi tanár

Készítette:

Tóthfalusi Tamás

Mérnök informatikus

Külső konzulens:

Nagy Gábor

Villamosmérnök

Debrecen

2010

„Egyetlenegy probléma sem oldható meg azon a tudatossági szinten,
amelyen az keletkezett.”

- Albert Einstein

Tartalomjegyzék

I. Bevezetés

Az FPGA-alapú eszközök mára igen elterjedté váltak, az ipari és informatikai eszközökben való integráltságuk szinte határtalan megoldási, alkalmazási lehetőséget nyújt. A párhuzamos gondolkodás, a növekvő teljesítmény és a csökkenő ár mindezt tovább fogja fokozni a közeljövőben. Teljesítményük és határtalan felhasználási módjuk következtében több vállalat is fejleszti és gyártja ezeket az eszközöket, folyamatos versenyben egymással. Más vállalatok azonban a már kész FPGA-kat használják fel saját eszközeikben, növelve azok teljesítményét és rugalmasságát.

Az FPGA-k és a rájuk épülő berendezések előállítása mellett azonban az ezzel foglalkozó cégeknek másra is figyelniük kell, ez pedig az a programozó réteg, amely majd esetleg az ő termékeiket fogja használni. Ennek érdekében egyre több cég kapcsolódik be az oktatásba, saját bemutatókat, oktatási kurzusokat tartva mutatják be az általuk gyártott hardvereket.

A National Instruments vállalat szintén több éve foglalkozik olyan eszközök gyártásával, amelyek FPGA-ra épülnek. Mára termékeik nagy része ebbe a kategóriába sorolható. Korábban azonban nem foglalkoztak ezeknek az eszközöknek a felprogramozásával, másfél évvel ezelőtt azonban FPGA témájú szakdolgozatcímeket hirdették meg. Ezzel a legfőbb céljuk az volt, hogy ismertebbé tegyék ezt a technológiát, és annak jelentőségét. A témák meghirdetésével megteremtették a lehetőséget a hallgatók számára, hogy egy olyan technikai újdonságot ismerjenek meg, amely a kötelező oktatási tervben nem szerepel. Eszközök és képzett szakemberek biztosításával segítették az FPGA programozás oktatásba való bevezetését. Másfél évvel ezelőtt meg is történt az FPGA-alapú szakdolgozattémák kiírása és egyeztetése a megfelelő oktatókkal.

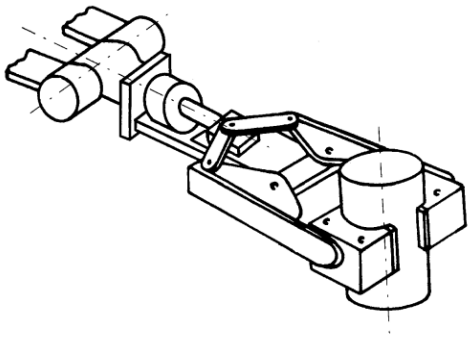
A cég által meghirdetésre kerülő szakdolgozatcímek közül választottam ki én is a dolgozatom témáját. Ennek oka az volt, hogy komolyabban meg szerettem volna ismerkedni az FPGA eszközök programozásával, és a bennük rejlő lehetőségekkel. Véleményem szerint ez a technológia újabb informatikai területeket fog meghódítani, és ennek én is szeretnék a részesévé válni. A robotkar működése pedig szintén egy másik általam kedvelt, de még nem ismert informatikai terület hozzátartozója. Így lehetőségem nyílt nemcsak a robotikába, hanem az FPGA programozásba is betekintést nyerni.

II. Elméleti áttekintés

Az ipari robot, ISO szabvány szerinti hivatalos megfogalmazásban, egy automatikusan vezérelt, újraprogramozható, többfunkciós manipulátor három, vagy több szabadságfokkal. A robotok nagy előnye, hogy nem fáradnak el, így akár 24 órán át képesek végezni ugyanazt a mozgássorozatot éveken keresztül. Ez az oka annak, hogy mára az ipari felhasználásuk elterjedté vált.

Mivel a robotok általában az emberi mozgást próbálják utánozni, ezért felépítésükben is gyakran próbálnak az emberi karra hasonlítani. Ennek a kivitelezése azonban nehéz, a mozgó részeket általában csuklókkal, csúszkákkal, szervó motorokkal, stb. működtetik. A nagy teherbírás érdekében a szervó motorok hajtóműveket működtetnek a kívánt pozíció eléréséhez, illetve a célhelyzet megtartását fékekkel biztosítják. A mozgató részen kívül fontos elemük a megfogó rész, illetve rendelkezhetnek szenzorokkal is.

A robotok fogóeszközei az általuk végzett munkafolyamatra specializálódnak. A különböző tárgyak, eszközök, munkadarabok megfogását általában alakzáró szorítópofákkal, vagy ujjakkal valósítják meg.



2.1. ábra – Kétpofás alakzáró megfogó [2]

A fejlettebb robotok a környezetükről különböző információkkal rendelkeznek, amelyeket a működésükhöz felhasználnak. Ezeket az információkat szenzorok, érzékelők segítségével gyűjtik be és használják fel. A szenzorokat két nagyobb csoportra oszthatjuk. Megkülönböztetünk belső- és külső érzékelőket.

- A belső érzékelők a mozgatórendszerhez és a megfogó szerkezethez illeszkednek. Funkciójuk szerint lehetnek útmérők, szögsebességmérők, nyomatékérzékelők.

- A külső érzékelő a nevéből adódóan a robot és a külső környezet között teremt kapcsolatot. Két nagy csoportjuk a tapintó és a látóérzékelő. A tapintó szenzor a közvetlen tárggyal való kapcsolatot érzékeli és elemzi, a látó szenzor fogalomkörébe azonban már a kamera és a képfeldolgozás témaköre tartozik.

Az érzékelők használatával lehetőség nyílik további munkafeladatok ellátására, mint például az elkészült munkadarabok vizsgálata és ellenőrzése. Erre csak a fejlett érzékelőkkel és mesterséges intelligenciával rendelkező ipari robotok képesek.

A robot az érzékelőkön kívül más be- kimeneti eszközökön keresztül is kommunikálhat a külvilággal, vagyis a kezelőjével vagy programozójával. Bemeneti eszköz lehet egy Joystick, egér, funkciógombok, pozícionáló gömb, vagy akár billentyűzet. Ha a robot nem önállóan végzi a feladatát, esetleg ember irányítja az adott munkafolyamatot, akkor az előbb említett bemeneti eszközökkel mindez megtehető. Szintén hasznosak a programozási eljárásnál, új útvonal beállításánál vagy betanításánál. A kimeneti eszközök közé a kijelzőt sorolnám, amely a robotkar adott állapotát, a végzett tevékenységet jellemző mennyiségeket jelenítheti meg, de hasznos lehet hibakeresésnél is. Érintőkijelző esetén nemcsak kimeneti, de bemeneti eszközzé is válhat.

A robotok talán legfontosabb tulajdonsága, hogy újraprogramozhatóak, így egy másik feladat elvégzésére is alkalmassá tehetőek. Felprogramozásukra többféle mód ismert, amelyeket két nagyobb csoportra oszthatunk: On-line és Off-line.

On-line programozás során közvetlenül magát a robotot programozzuk. Ennek a módszernek az előnye, hogy a programozó a munkaterület fizikai felépítését figyelembe véve tudja ellátni a feladatot, így nem fordulhat elő a munkaterületen kívülre eső célpont. Hátránya viszont, hogy ebben az időszakban a robot nem üzemelhet. Ide tartozik többek között az átvezető, illetve az átsétáló programozási módszer.

Átvezető programozás esetén a mozdulatok eltárolása egy vezérlőpult segítségével történik. Egy kezelő szakember lépésenként eltárolja a mozgásokat egy kezelőpult segítségével. Az egyes lépések külön-külön kerülnek eltárolásra.

Átsétáló programozás esetén a robotkart a kezelő, vagy programozó kézzel vezeti végig egy adott útvonalon, és az útvonal fontosabb paramétereit a robotkar eltárolja. Ezekből a

paraméterekből később vissza tudja játszani a mozgássorozatot. A módszer hatékonyságának érdekében a programozónak az adott munkaterületen jártasnak kell lennie, hogy optimális útvonal kerüljön tárolásra.

Off-line programozás során a vezérlést végző kód a robottól függetlenül kerül kifejlesztésre. Ennek a módszernek az előnye, hogy a robot a programozási időszakban folyamatosan üzemelhet, de így a munkaterület fizikai adottságaiból adódó korlátok problémát okozhatnak.

A robotok csoportosítása irányításuk szerint a következőképpen lehetséges:

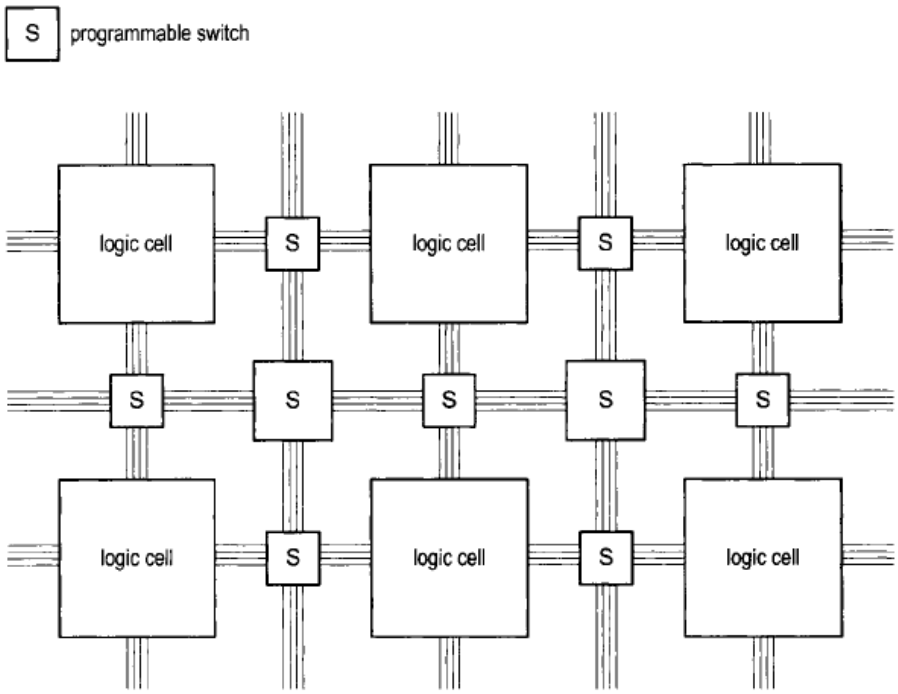
- Ipari manipulátorok
    - Programozható manipulátor. Megfogó szerkezettel rendelkezik, amelyet egy egyszerű program vezérel. Adott mozgássorozatot ismétel, tárgyak, anyagok, árukészletek áthelyezésére használható.
    - Teleoperációs manipulátor. Ez lényegében egy emberi irányítású manipulátor, valamilyen beviteli eszköz segítségével a kezelő egy biztonságos helyről irányítja a munkafolyamatot.
- Ipari robotok
    - Pont-szakasz vezérlésű robot. Az adott koordinátákat, illetve célpontokat a robot a felépítésének megfelelően éri el. A robot egy adott mozgássorrenden halad végig.
    - Pályavezérlésű robot. A mozgás során két koordinátapont között egy általa meghatározott pályán halad.
    - Intelligens robot. A robot szenzorokkal rendelkezik és egyes mozgássorozatokat ezek segítségével határoz meg.

A robotika fejlődése maga után vonja az ipari robotok terjedését is, egyre többféle munkafolyamat elvégzésére válnak alkalmassá. A fejlettebb érzékelők és a különféle bonyolult algoritmusok segítségével manapság már nemcsak a feladatot végezhetik el, de ellenőrizhetik a készterméket is. Felhasználásuk a jövőben egyre nagyobb teret fog hódítani.

III. A felhasznált eszközök és technológiák bemutatása

III. 1. Az FPGA

Az FPGA (Field-Programmable Gate Array) egy olyan logikai eszköz, amely egy kétdimenziós tömbben elhelyezett logikai cellákból és programozható kapcsolókból áll.



3.1. ábra – FPGA eszköz elméleti felépítése [4]

Az egyes logikai cellákat alakíthatjuk ki egy egyszerű funkció elvégzésére, a programozható kapcsolókkal pedig testre szabhatjuk az egyes logikai cellák közötti összeköttetéseket.
A logikai cellák egy úgynevezett konfigurálható logikai blokk (CLB – Configurable Logic Block) erőforrásai. A logikai cella (LC – Logic cell) további részegységekre bontható. Ezek a LUT (Look-Up Table) és a D-típusú Flip-Flop. A LUT olyan 1 bit széles memória, amelynek mélysége, tehát a címvonalainak a száma változó. Tipikusan ez a szám 4, így egy tetszőleges 4 változós függvény valósítható meg vele. A D-típusú Flip-Flop pedig egy élérzékeny tároló, amely órajel és Set/Reset bemenettel rendelkezhet.
Egy általunk elkészített terv tehát a logikai cellák funkciójának pontosításából, és a programozható kapcsolók összeköttetéseinek beállításából áll. Ha a tervünk és az

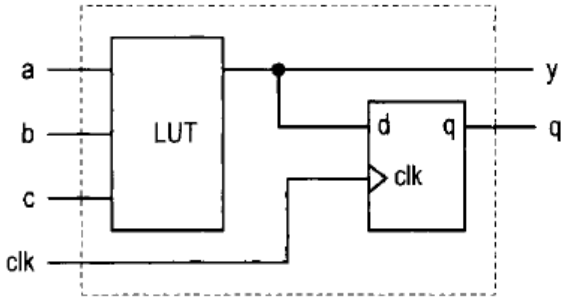
implementáció, vagyis a használt szoftver által készített huzalozási lista, elkészült, egy megfelelő illesztőkábel segítségével rátölthetjük a konfigurációnkat az FPGA eszközre.
Az illesztőkábel gyártónként és típusonként eltérhet. Pl.: USB, JTAG, Ethernet kábel, stb.
A technológia előnye, hogy az eszköz hardver szinten programozható. Csak modulszintű leírást kell elvégeznünk, a részletek kidolgozását és a huzalozás optimalizálását a tervezőrendszerre hagyjuk.
Felhasználó általi újraprogramozhatóság szempontjából többféle FPGA-t különböztethetünk meg:

- OTP FPGA: csak egyszer programozható, ennek köszönhetően azonban a kód biztonságban van. Gyártók közé sorolható az Actel és a Quicklogic.
- Flash FPGA: sokszor újraprogramozható, de csak törlés után. A kód tehát állandó, viszont szükség esetén módosítható. Előnye a kis fogyasztás, így főleg hordozható eszközöknél jelenik meg (MP3 lejátszó, kamera). Gyártó pl.: Actel.
- SRAM FPGA: hagyományos, CMOS technológiára épül. Az információ memóriacellákban található, ezért bármikor átprogramozható. Hátránya, hogy minden induláskor újra kell konfigurálnunk, ugyanis kikapcsoláskor elvész a kód. Előnye azonban, hogy olcsó. Főbb gyártók közé tartozik a Xilinx és az Altera.

Az FPGA-k tehát általános célú logikai elemekből épülnek fel, melyekhez programozható huzalozás tartozik. Alapvetően minden felhasználói láb Input/Output, vagyis bemeneti/kimeneti. Alkalmazhatóságuk rugalmas a felhasználó által módosítható homogén belső szerkezetének köszönhetően, melyhez párosul a viszonylag gyors tervezési lehetőség. Az alkalmazott szoftver programfájl generálása azonban az áramkör bonyolultságától függően több időbe is telhet. A szoftverek előnye azonban, hogy elvégzik helyettünk az áramköri elemek huzalozását, elhelyezését és optimalizálását. Az esetleges tervezési hibák esetén pedig a sokszoros módosíthatóságnak köszönhetően az áramkör tényleges megépítése nélkül újra és újra tervezhetjük és tesztelhetjük a munkánkat.

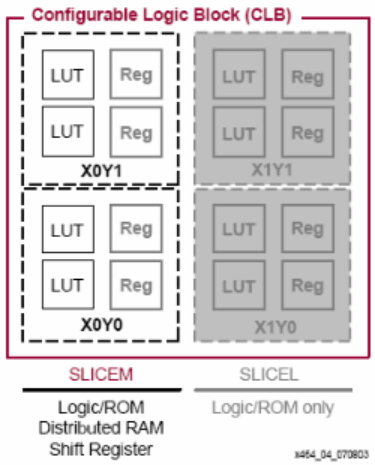
III. 1. 1. Xilinx Spartan-3E

Az általam használt FPGA-t a Xilinx cég gyártotta és a Spartan-3E családba tartozik.
A Spartan-3 eszközök legalapvetőbb eleme szintén a logikai cella (LC – Logic Cell), amely egy 4 bemenetű LUT-ot és egy D-típusú Flip-Flopot tartalmaz, hasonlóan az 3.2 ábrához.



3.2. ábra – Az FPGA logikai cellájának általános felépítése [4]

Összegezve a logikai cella egy átviteli áramkört tartalmaz, ami az aritmetikai funkciók implementálásához használatos. Tartalmaz továbbá egy áramkört a multiplexerek implementálására. A LUT-ot használhatjuk továbbá egy 16bit mélységű és 1 bit szélességű SRAM-ként, vagy egy 16 bites léptetőregiszterként. A rugalmasság növelése és a teljesítmény tökéletesítése érdekében 8 logikai cella kapcsolódik egy speciális belső huzalozási struktúrával. A Xilinx terminológiája szerint két logikai cellát csoportosítanak egy úgynevezett „slice”-ba, a négy slice együtt pedig egy CLB-t alkot.



3.3. ábra – Spartan-3E FPGA Configurable Logic Block felépítése [3]

A Spartan-3 eszközök négy féle funkcionális blokkot tartalmaznak. Ezek a szorzó, a blokk RAM, az órajel kezelő egység (DCM – Digital Clock Manager), és a be/kimeneti blokk (IOB – Input/Output Block). A szorzó két 18 bites számot fogad el bemenetként és ebből állítja elő a végeredményt. A blokk RAM egy 2kByte-os szinkron SRAM, amelyet független órajeltartományok közötti szinkronizációra, vagy adatcserére használhatunk. Az órajel kezelő egység digitális késleltető ciklust használ az órajeleltérések csökkentésére, valamint az órajel frekvenciájának vezérlésére és fáziseltolására. A negyedik funkcionális blokk a be/kimeneti blokk, amely az adatforgalmat vezérli a külső portok és a belső logika között.

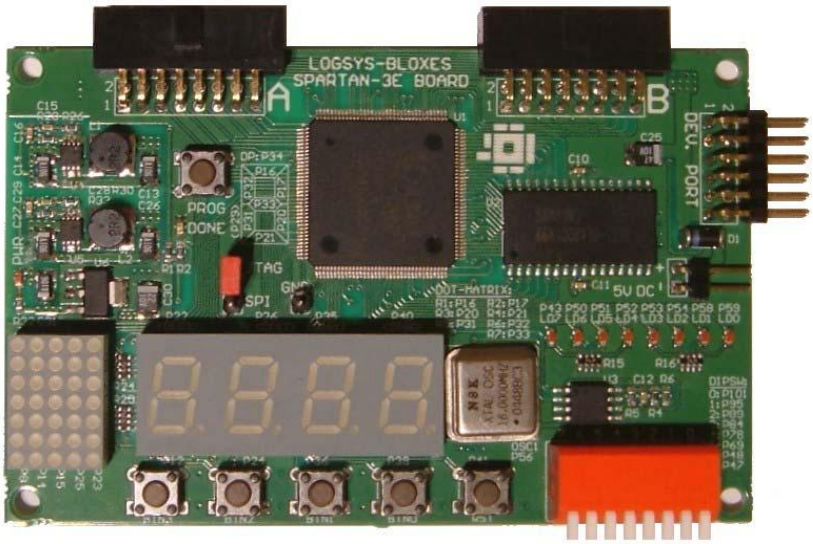
A Spartan-3E család csökkenti a rendszerköltségeket azzal, hogy a legalacsonyabb logikai egység költségarányt nyújtja az FPGA családok között. Támogatja a legalacsonyabb költségű konfigurációs megoldásokat, amelybe beletartozik többek között a párhuzamos flash memória és az SPI.

A Spartan-3E és a Spartan-3 FPGA családok szolgáltatják a legalacsonyabb költségű programozható megoldásokat a tervünk elkészítéséhez, 50.000 – 5.000.000 terjedő kapuszámmal, valamint 784-ig terjedő be/kimenettel. A Spartan-3E a kapu centrikus, míg a Spartan-3 az I/O centrikus alkalmazások számára lett optimalizálva. Komolyabb tervek elkészítése előtt figyelembe kell vennünk az FPGA erőforrásait. Ha egy összetettebb, és sok belső áramköri elemet igénylő projektet készítünk, megfelelően kell kiválasztanunk hozzá a programozható eszközünket. A gyártók a típusadatokat különféle összehasonlító-táblázatokban biztosítják számunkra. Egy ilyen táblázatrészlet ábrázol a következő táblázat, amelyben a Spartan-3E családon belül elérhető néhány típus főbb adatait tekinthetjük meg.

Jellemzők/Típus	XC3S100E	XC3S250E	XC3S500E	XC3S1200E	XC3S1600E
Kapuszám	100.000	250.000	500.000	1.200.000	1.600.000
Logikai cellák száma	2.160	5.508	10.476	19.512	33.192
Blokk Ram bitek száma	72.000	216.000	360.000	504.000	648.000
Órajel kezelő egységek száma	2	4	4	8	8

III. 1. 2. Logsys Panel

A szakdolgozatom elkészítéséhez a Logsys Spartan-3E FPGA kártyát használtam. A 3.4. ábra a panelt ábrázolja, amely a Felhasználói útmutatójából származik.



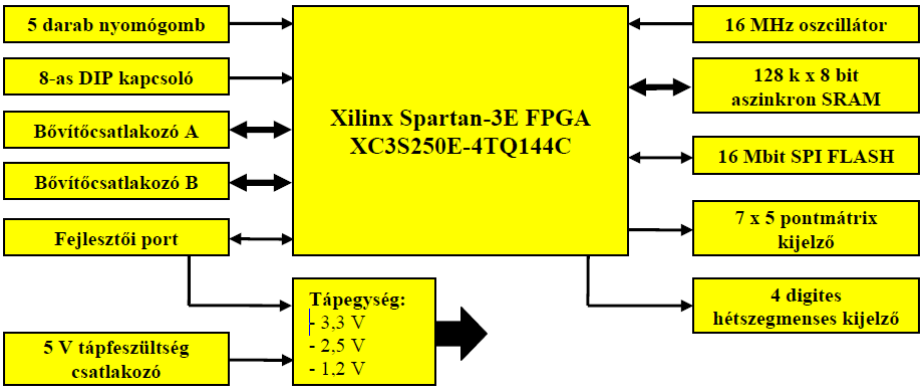
3.4. ábra – Logsys panel

A kártyán az alábbi komponensek találhatók:

- Xilinx XC3S25E-4TQ144C típusú FPGA, melynek főbb jellemzői:
  - 250 ezer kapu (4896 LUT és flip-flop)
  - 12 darab 18 kBites blokk-RAM
  - 12 darab 18x18 bites előjeles szorzó
  - 4 darab DCM (Digital Clock Manager) modul
- Memóriák a program és az adatok tárolására:
  - Egy 128kx8 bites, 10ns-os aszinkron SRAM (Samsung K6R1008V1D-TI10)
  - Egy 16 Mbites SPI buszos soros FLASH memória (Winbond W25P16)
  - A soros FLASH memória konfigurációs memóriaként is szolgál az FPGA számára
- Megjelenítő eszközök:
  - 8 darab LED
  - 4 digites hétszegmenses kijelző

    - 7 x 5 pontmátrix kijelző
- Beviteli eszközök:
    - 5 darab nyomógomb
    - 8-as DIP kapcsoló
- Egy 16 MHz-es oszcillátor
- Csatlakozó a Logsys fejlesztői kábel számára
- 2 darab csatlakozó a kiegészítő modulok számára:
    - 13 FPGA I/O láb (11 kétirányú, 2 csak bemenet)
    - 5V és 3,3V tápfeszültség kimenet

A kártya összefoglaló blokkvázlatát a 3.5. ábrán láthatjuk, melynek forrása szintén a Logsys panel Felhasználói útmutatója.



3.5. ábra – A Logsys Panel blokkvázlata

A kijelzők vezérlése időmultiplexelt módon lehetséges. A két kijelző esetén hét vezérlőjel közös, ezekkel lehet a hétszegmenses kijelző egyes szegmenseit, illetve a pontmátrix kijelző oszlopaiban található LED-eket bekapcsolni. Minden egyes karakter, illetve oszlop külön kiválasztó jellel rendelkezik. A két kijelző természetesen önállóan is használható, ebben az esetben a hétszegmenses kijelzőhöz csak 4 ütemű, a pontmátrix kijelzőhöz csak 5 ütemű időmultiplexelt vezérlést kell használni.

A DIP kapcsolók 0-tól 7-ig vannak számozva, a bal szélső kapcsoló sorszáma a 7, a jobb szélsőé pedig a 0. A kapcsolók logikai értéke alsó állásban „0”, felső állásban pedig „1”.

Az FPGA kártya nem csak a rajta elhelyezkedő 16 MHz-es oszcillátortól, hanem a fejlesztői port CLK vonaláról is kaphat órajelet.

Mindkettő az FPGA egy-egy órajel bemeneti lábára csatlakozik. Az oszcillátor 16 MHz-es órajeléből az FPGA-ban található DCM (Digital Clock Manager) modulok segítségével egyéb frekvenciák is előállíthatók.

A Logsys panel felkonfigurálására kétféle mód lehetséges. Az egyik lehetőség a fejlesztői port JTAG interfészén keresztül. Ezen kívül az eszköz képes magát a kártyán lévő SPI buszos soros FLASH memóriából is felkonfigurálni. A konfigurációs mód egy jumperrel választható ki. A JTAG interfész a kiválasztott módtól függetlenül mindig rendelkezésre áll.

Az FPGA kártya 5 V-os tápfeszültséget igényel. A tápellátás alapvetően a fejlesztői kábelről történik, de lehetőség van egyéb külső 5 V-os egyenfeszültség forrás csatlakoztatására is. A kártyán található összes periféria és az I/O vonalak 3,3 V-ról működnek.

A kiegészítő modulok illesztését két 16 pólusú csatlakozó teszi lehetővé. A bővítőportok lábkiosztását a 3.6. ábra mutatja, melynek forrása a kártya Felhasználói útmutatója.

(15) Input	(13) I/O	(11) I/O	(9) I/O	(7) I/O	(5) I/O	(3) +3,3V	(1) GND
(16) Input	(14) I/O	(12) I/O	(10) I/O	(8) I/O	(6) I/O	(4) I/O	(2) +5V

3.6. ábra – A Logsys panel bővítőportjainak lábkiosztása

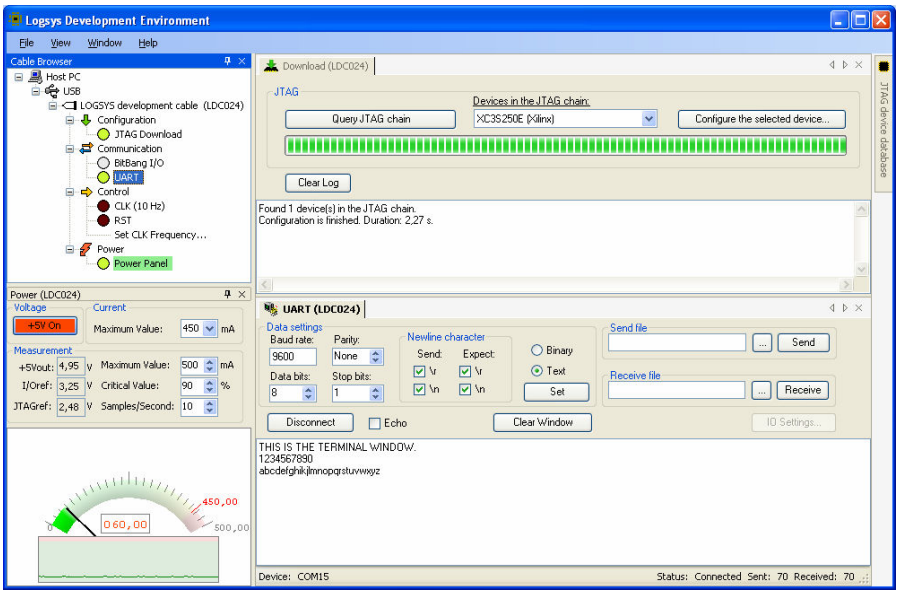
A csatlakozókra ki van vezetve a 3,3 V-os és az 5 V-os tápfeszültség is, azonban az adatvonalak 3,3 V-ról működnek és nem 5 V toleránsak. A bővítőportok segítségével ez az FPGA kártya nagyon sokféle feladat elvégzésére válik alkalmassá. A Budapesti Műszaki Egyetemen több bővítőmodult is készítettek hozzá, amelyek segítségével egyszerűen csatlakoztathatunk például egy PS/2 billentyűzetet, vagy egy VGA csatlakozóval rendelkező monitort. Így az FPGA-hoz még több bemeneti/kimeneti eszközt csatlakoztathatunk a kártyára szerelteken kívül. Ugyanakkor természetesen nemcsak eszközöket, hanem áramköröket is csatlakoztathatunk, sőt több FPGA-t köthetünk össze egymással. Így egyes bonyolult számolások, folyamatok elvégzését több eszköz is elvégezheti egymással párhuzamosan, folyamatosan adatot cserélve egymással. Ezt a módszert használják például kriptográfiai algoritmusok eredményének visszafejtésénél.

A robotkar vezérléséhez mind a két bővítőcsatlakozót használtam. Az egyikhez egy kiegészítő panelt csatlakoztattam, melyet a National Instruments készített. Ez a panel külső 5V-os tápfeszültséget igényel, melyet 10 vonalra bont le a szervók részére.

További 10 bemeneti vonalat biztosít a szervók csatlakoztatására, melyek a megfelelő bővítőcsatlakozó-lábakhoz kapcsolódnak. Így lehetőségünk van akár 10 szervó egyidejű vezérlésére, de az általam használt robotkar csak 6 motort használ a mozgáshoz.
A másik csatlakozóra egy AT billentyűzetet csatlakoztattam, melynek működését a „Billentyűzetek” című részben ismertetek.

III. 1. 3. Grafikus fejlesztői környezet

Az elkészített program FPGA-ra való feltöltésére, illetve a Logsys panel monitorozására és vezérlésére egy jól áttekinthető és testreszabható grafikus program áll a rendelkezésünkre, amelyet a Budapesti Műszaki Egyetem készített. Ez biztosítja a fejlesztői kábel funkcióinak elérését. A programozható eszköz konfigurálására a JTAG interfész áll rendelkezésünkre. A grafikus felhasználói felület a 3.7. ábrán látható.



3.7. ábra – A Logsys panel felprogramozását segítő grafikus program képernyőképe

Ha elkészítünk egy tervet és generálunk egy az FPGA-ra feltölthető programot, akkor egy SVF, vagy Xilinx eszközök esetén egy BIT fájlt kapunk. A Logsys rendszer az SVF fájlok kezelése mellett, Xilinx eszközök esetén, közvetlenül támogatja a BIT és a JEDEC fájlok használatát is. A fejlesztői kábel többféle szinkron és aszinkron soros kommunikációs protokollt támogat. Ezek közé tartozik az UART. Egyszerű tesztelési feladatok elvégzéséhez rendelkezésünkre áll a BitBang I/O kommunikációs mód, melyet én is használtam.

Az I/O műveletek eredményeit kimenthetjük fájlokba, illetve visszatölthetjük azokat onnan. Lehetőségünk van továbbá órajel generálására az FPGA eszköz számára, melynek értéke 1 és 10.000 Hz közötti érték lehet. Fontosabb funkció még a szinkron Reset jel küldési lehetőség. Külön panel szolgál a fogyasztási állapot megjelenítésére.

III. 2. Hardverleíró nyelvek

A HDL (Hardware Description Language), vagyis hardverleíró nyelvek mára igen jelentőssé váltak. A HDL nyelv, amint az a nevéből is kiderül, bonyolult áramkörök leírására szolgál. Régebben ezek a nyelvek csak az áramkörök tesztelésére szolgáltak, mára azonban olyan fejlődéseken mentek keresztül, amelyek lehetővé teszik őket bonyolult kapcsolási rajzok előállítására is. Ennek a nagy előnye abban rejlik, hogy nem kell elkészítenünk a kapcsolási rajzunkat kézzel, papíron, és nem kell megépítenünk az adott áramkört. A HDL nyelvek használatával leprogramozhatjuk a kívánt működést, egy alkalmas program (úgynevezett szintetizáló program) pedig elvégzi helyettünk az áramkör megtervezését és optimalizálását.

A gyors és rövid kódokkal ellentétben itt a fő hangsúly a szerkezeten van, vagyis szintetizálhatónak, hardverrel ábrázolhatónak kell lennie. Ennek oka az, hogy az adott kódrészletből a fordítóprogramnak fel kell ismernie, hogy az egy összeadó, vagy esetleg egy multiplexer.

Ha pedig kifelejtettünk valamit a tervünkből, vagy esetleg nem úgy működik, ahogyan azt eredetileg elterveztük, javíthatjuk azt a programkód módosításával. Az esetleges hibák kiszűrésére készíthetünk teszteket, vagy lehetőség van a generált áramkör felépítésének és áramköri elemeinek a vizsgálatára. A tesztek segítségével egy adott áramköri elem bemenetét, illetve kimenetét vizsgálhatjuk a lehetséges állapotokban. Lehetőség van a kimenetek értékeinek monitorozására a bemenetek változásával azonos időben. Ezt az informatikai lehetőséget használják manapság a félvezető gyárak is, ahol a bonyolult áramköröket nem kapcsolási rajz alapján készítik, hanem hardverleíró nyelveken írják le a működést. Összefoglalva tehát egy számítógép és egy olcsó FPGA kártya segítségével egy modern, összetett digitális rendszert építhetünk.

A két legelterjedtebb és legjobban támogatott hardverleíró nyelv a Verilog és a VHDL, melyek nagyon hasonlóak egymáshoz, a szintaktikai különbséget leszámítva. Mindkét nyelv megfelel az IEEE szabványoknak. A szakdolgozatom elkészítéséhez a Verilog nyelvet választottam, melynek oka, hogy ezzel a nyelvvel már korábbi tanulmányaim során találkoztam, és ennek az ismeretét szerettem volna komolyabban elsajátítani.

III. 2. 1. Verilog

A Verilogot [12] az 1980's években fejlesztették ki, formailag az IEEE szabványoknak megfelel. 1995-ben hagyták jóvá a szabványt (ennek a neve a Verilog-1995), majd 2001-ben újra átdolgozták, és kialakult egy újabb változata (ez az úgynevezett Verilog-2001). Az újabb változatában sok hasznos újdonságot adtak hozzá, amely megkönnyíti vele a munkát. A Verilog nyelvet különböző szintű digitális rendszerek leírására és modellezésére fejlesztették ki, ezért egy nagyon összetett nyelv. A szintaktikája hasonlít a C nyelvre, szemantikailag azonban az egyidejű, párhuzamos hardverműveleteken alapul. Ez azt jelenti, hogy a programkód blokkokból építhető fel, melyek párhuzamosan futnak. Természetesen érzékenységi listájuk eltérhet.

III. 2. 2. Xilinx ISE Webpack

Az általam írt kód fordítására és tesztelésére a Xilinx cég ingyenes ISE Webpack programját használtam [11]. A fejlesztőkörnyezet használata egyszerű, és sok funkcióval rendelkezik. Ezek közé tartozik:

- az előre tárolt példák, a használt nyelvnek megfelelően,
- sematikus ábrák készítése, amely a programozott kód hardver szinten történő megjelenését ábrázolja,
- felhasználói megszorítások könnyű kezelése,
- a generált programfájlt lehetőségünk van feltölteni a program által támogatott eszközökre,
- a teszteléshez szimulációk, szimulációs fájlok hozhatók létre.

III. 3. Billentyűzetek

A billentyűzet az adatbevitel egyik legkönnyebb eszköze. Az első IBM PC megjelenése óta a billentyűzet elektronikai és szabványfelépítése nem változott. Az IBM PS/2 sorozatának köszönhetően kialakult az általános tervrajza és csatlakozója, és ezután már a világ PS/2 billentyűzet néven kezdte hívni. Annak a ténye, hogy a szabvány nem változott közel 20 évig, 2 pozitív következményt von maga után:

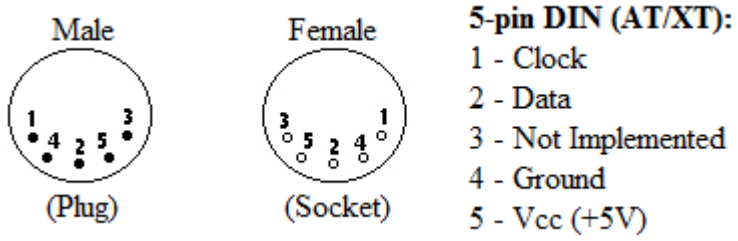
- Alacsony ár.
- Más bemeneti eszközök is használják a PS/2 szabványt.

A szabványnak két fontos tulajdonsága van:

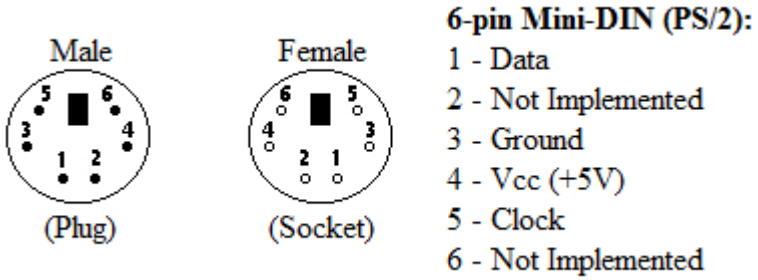
1. A kommunikáció bináris, szinkron és soros alapú. Csak két csatornára van szükség, az egyik az órajel, a másik az adatcsomag számára.
2. A kommunikáció kétoldali. Azon kívül, hogy a billentyűzet adatfolyamot továbbít a számítógép (host) felé, a host is küldhet szabályozó, beállító parancsokat a bemeneti eszköznek. Pl.: ezek használatával irányíthatók a ledek (Num/Caps/Scroll Lock). Egyoldali kommunikációnál ezekre természetesen nincs szükség.

III. 3. 1. Csatlakozás és a PS/2 szabvány

A fizikai PS/2 port használatára kétféle csatlakozási lehetőség áll a rendelkezésünkre. Az egyik az úgynevezett 5-pólusú DIN (5-pin DIN), a másik a 6-pólusú mini-DIN (6-pin mini-DIN). Mindkét csatlakozó teljesen hasonló, a gyakorlati különbség a felépítésében rejlik.
A 6-pin mini-DIN csatlakozóval rendelkező billentyűzetet gyakran PS/2 billentyűzetnek, az 5-pin DIN csatlakozóval rendelkezőt pedig AT billentyűzetnek hívják. A 3.8. és a 3.9. ábra a kétféle csatlakozót mutatja be.



3.8. ábra – 5 pólusú DIN csatlakozó felépítése [8]



3.9. ábra – 6 pólusú Mini-Din csatlakozó felépítése [8]

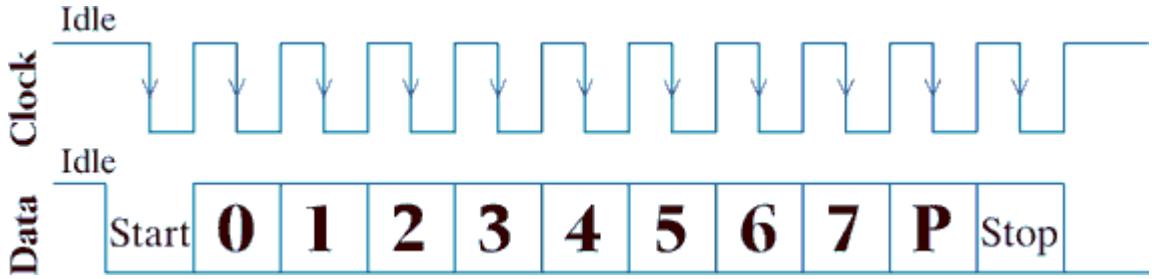
Az előkészítésnél két fontos dologra kell figyelnünk, mielőtt csatlakoztatunk egy billentyűzetet egy vezérlő egységhez:

1. A billentyűzet működéséhez biztosítani kell a megfelelő vezetékeken a +5V áramot, illetve a földelést (GND). Megjegyzem, én a működéshez a Logsys Panel +3V-os kimenetét használtam, amellyel megfelelően működött az eszköz.
2. A logikai „0”-nak a +5V, a logikai „1”-nek pedig a 0V felel meg.

Minden gombnyomás alkalmával a bemeneti eszközben lévő kódoló egység egy adatcsomagot küld. Minden adatcsomag 11 órajel ciklusig tart, amely alatt 11 bit átvitelére kerül sor. Ebből azonban csak 8 bit a tényleges adat. Az első bit az úgynevezett start bit,

amelynek az értéke logikai nulla. Az adatcsomagok küldése között az órajel vonalon végig logikai 1-es szint van, így amikor elkezdődik az adatküldés, nullára vált, ezzel jelezve, hogy adatot küld. Az adatvonalon tehát megjelenik egy nulla, amely után az órajel vonalra is nulla érték kerül, és 11 órajel ciklus fut le rajta. Az adatok olvasása leszálló órajelekre történik. Elküldésre kerül a 8 darab adat bit, melyet egy paritás bit követ. Ennek szerepe az adat sikeres megérkezésének ellenőrzésében van. A 8 adatbit és az 1 paritás bit együtt összesen páratlan számú egyest tartalmaznak. Végül pedig az utolsó adatbit, az úgynevezett stop bit elküldésével jelzi a billentyűzet, hogy a küldés véget ért. A stop bit értéke logikai egyes szintű, mely után az órajel vonalon is logikai 1 jelenik meg mindaddig, amíg újra nem kerül sor adatküldésre.

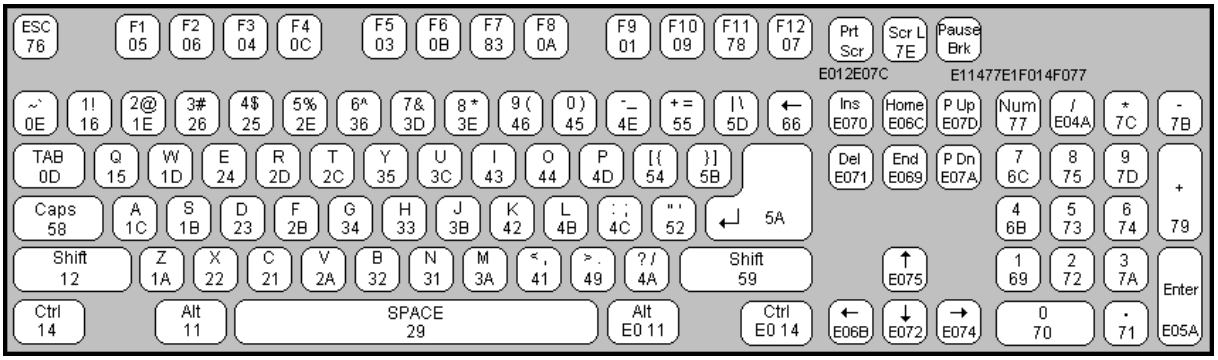
Összefoglalva tehát az órajel csak az adatátviteli periódus alatt generálódik, frekvenciája 10 és 30 kHz közötti érték.



3.10. ábra – Az adat és órajel vonalak értékei adatküldés közben, PS/2 szabvány szerint [10]

III. 3. 2. Billentyűkódok

Ha lenyomunk egy billentyűt, az eszköz elküldi az adott gombhoz tartozó kódot. Minden billentyűhöz más kód tartozik, jelentése függ a billentyűzetkiosztástól. Például egy angol billentyűzet „Q” betűjének a kódja egy török billentyűzet „F” betűjének felel meg. Az 3.11. ábra egy angol billentyűzet kódjait tartalmazza.



3.11. ábra – Egy adott billentyű és a hozzá tartozó billentyűkód

Ha lenyomjuk az „A” betűt, a hozzá tartozó „1C” hexadecimális kód kerül elküldésre. Ha nem engedjük fel, és folyamatosan nyomva tartjuk, hosszabb ideig, mint az úgynevezett „typematic” késleltetés, akkor egy újabb „1C” kerül elküldésre. A „typematic” késleltetés értéke beállítható. Ez az „1C” érték addig fog ismétlődni, ameddig el nem engedjük a billentyűt. Felengedés után az eszköz egy újabb hexadecimális értéket küld nyugtázásképpen, ez az „F0”, majd pedig újra megismétli a lenyomott billentyű kódját, tudatva velünk, hogy melyik billentyűnek az értéke került elküldésre az előbb.

Az „A” gomb lenyomása után tehát a következő 3 hexadecimális számérték kerül elküldésre a következő sorrendben: „1C” „F0” „1C”.

Ha folyamatosan nyomva tartjuk az „A” billentyűt, az „1C” kód addig ismétlődik, ameddig fel nem engedjük, lezárásképpen pedig szintén az „F0” és végül az „1C” következik. Tehát „A A … A ” => „1C 1C … 1C F0 1C”.

III. 4. Szervók

A szervók olyan motorok, amelyek vezérlése úgynevezett impulzusszélesség modulált jelek segítségével történik. Ezen jelek vezérlési elvét bővebben kifejtem a „III.4.1. Szervók vezérlése” című fejezetben. A motorok általános felépítése, hogy egy kimeneti tengellyel és 3 bemeneti vezetékkel rendelkeznek.

Az eszköz szerkezete több részből tevődik össze. Ezek közé tartozik az úgynevezett DC motor, amely egyenárammal működő elektronikus motor.

A forgás alapja egy a kimeneti tengelyhez kapcsolódó, és a megfelelő pozíció beállítását elvégző potenciométer.

Működésükhöz elengedhetetlen továbbá egy fogaskerék-sorozat, mely készülhet műanyagból vagy a komolyabb szervók esetében fémből.

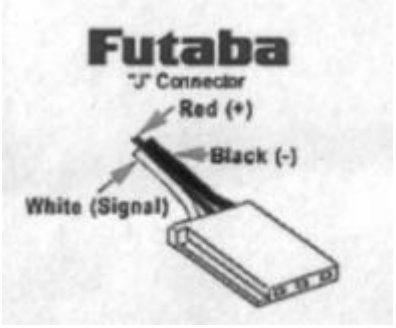
A szervók használatának fő oka, hogy alacsony áramú eszközök, melyek tengelyének szögelfordulása precízen irányítható.

Robotoknál elterjedt alkalmazási területei közé tartozik:

- érzékelők mozgatása
- robotok végtagjainak mozgatása
- kerekek mozgatása.

A csatlakoztatásukra általánosan elterjed úgynevezett „J” csatlakozó 3 vezetékkel rendelkezik. Az egyik a 4V-6V közötti áramellátásért felel, egy másik a földelésért, a harmadik pedig a bemeneti vezérlőjelet továbbítja.

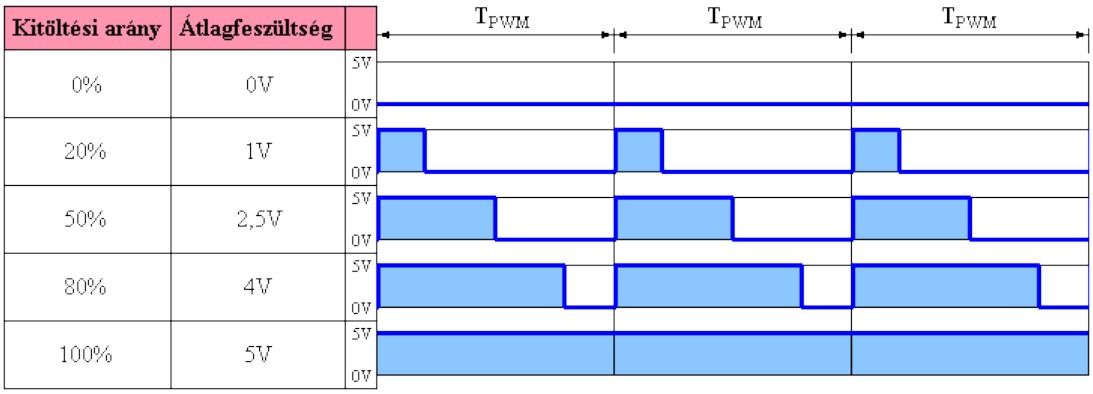
A 3.12. ábrán a „J” típusú csatlakozó látható, amely a Futaba S3003 szervó adatlapjáról származik.



3.12. ábra – Szervók általános csatlakozójának felépítése

III. 4. 1. Szervók vezérlése – PWM jel

Ha információt szeretnénk közölni a szervókkal, akkor erre egy speciális eszköz áll a rendelkezésünkre, a jel moduláció. Ez a folyamat azt jelenti, hogy az általunk elküldött négyszögjelet olyan mértékben megváltoztatjuk az eredetihez képest, hogy az információt tudjon hordozni. Így a vevő oldalon a módosított jel már jelentéssel bír, amely a szervók esetén egy adott irányba való elmozdulást fog jelenteni. Többféle modulációs eljárás ismert, amelyek a kimeneti jel adott jellemzőinek változtatását eredményezik. A szervó motorok vezérlésére elterjedt modulációs eljárás az impulzusszélesség moduláció (PWM – Pulse Width Modulation), amely során olyan digitális jelek állíthatók elő (továbbiakban PWM jel), melyek periódusa állandó, az átlagfeszültség beállítása pedig egy úgynevezett kitöltési tényező változtatásával történik. A PWM jelek pontosságát a kitöltési tényező beállíthatóságának pontossága határozza meg. Jellemzője a frekvencia, mely a jelek ismétlését szolgálja. A 3.13. ábra néhány különböző kitöltési tényezőjű PWM jelet ábrázol.



3.13. ábra – PWM jelek átlagfeszültsége különböző kitöltési arány mellett [14]

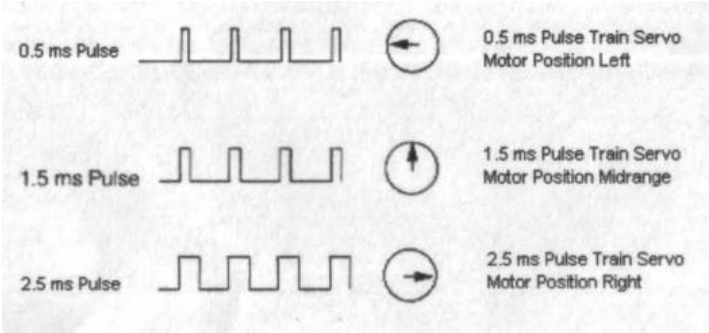
A szervó motorok irányításának alapja tehát egy PWM jel, melynek a periódusidőn belüli kitöltési tényezője egy előre meghatározott és típusonként eltérő minimum és maximum érték között változhat. A jel frekvenciája általában 50-60Hz közötti érték. Én a vezérlésre 20ms-es periódust használtam, ami 50Hz-nek felel meg.

A szervók működésében szintén megfogalmazható egy általános érték a kitöltési tényező tekintetében. 1,5ms érték a tengely középre állását eredményezi. Az alsó, illetve felső határ azonban típusonként, sőt azonos típusonként is eltérhet. A 180^o mozgásterű motoroknál az alsó érték nagyrészt 0,5ms, ez a 0^o, a felső érték pedig 2,5ms, ez a 180^o, körül mozog.

A szervó motor nem a bemenetére kapott PWM jelből előálló feszültséggel lesz működtetve, ezek csak referenciafeszültségek. Ez a bemeneti feszültség lesz a jövőbeni állapot, tehát összehasonlításra kerül egy másik referenciafeszültséggel, ami az aktuális állapothoz tartozik. Mivel a szervó tengelye egy potenciométer, a két feszültség kiegyenlítése érdekében elmozdul, ebből adódik a tengelyelfordulás, így juthatunk el a kívánt pozícióra. Ezt a helyzetet megtartják, esetlegesen emelve és folyamatosan tartva a rájuk helyezett súlyt.
A bemeneti jel ismétlésének igénye szempontjából megkülönböztethetünk analóg és digitális szervókat.

- Analóg szervó esetén a vezérlőjelet a már említett 50-60Hz-es periódussal kell továbbítanunk az eszköznek. Ennek oka, hogy csak nagyon rövid ideig kapja meg a vezérlő vonalon a feszültséget, és ezalatt az idő alatt nem biztos, hogy képes beállni a számunkra megfelelő pozícióra.
- Digitális szervó esetén elegendő egy vezérlőjel, ugyanis az eszköz addig fordul, ameddig el nem éri a számára előírt helyzetet.

Az első teszteléseket egy Futaba S3003 szervón végeztem.
Amint az a Futaba S3003 adatlapjában is szerepel, az optimális működési frekvenciája 50Hz. A szervó működését először a Logsys panellel teszteltem, amely a „B” portjának kimenetén 20ms-os periódust generálva, tehát 50Hz-et, küldte a vezérlőjeleket. A logikai 1-es szintben lévő impulzus ideje ezen perióduson belül 0,5ms, illetve 2,4ms értéket vehetett fel. A tengely 0 és 180fok között mozoghat.
A 3.14. ábrán láthatóak a szervó adatlapjában szereplő kitöltési tényezők, illetve ezek függvényében a tengely helyzete. A kép szintén a Futaba S3003 adatlapjáról származik.



3. 14. ábra – Futaba S3003 szervó tengelyének iránya különböző PWM jelek esetén

Az általam használt robotkaron más típusú szervó motorokat vezéreltem. A robotkar működését 3 különböző típusú, azonos gyártótól származó motor látja el:

- TowerPro MG-995
- TowerPro SG-5010
- TowerPro SG-91R.

A 3 különböző eszköz oka az általuk maximálisan elbírt súly, ugyanis az alul lévőknek nagyobb terhet kell felemelniük, mint a legfelsőknek. Természetesen ezen okok miatt árban is eltérnek egymástól, ezért költséghatékonyság szempontjából a felső részekre gyengébb szervók is elegendőek voltak.

A TowerPro MG-995 fémfogaskerekes szervó, melynek nyomatéka 4,8V-al vezérelve 11kg/cm. Az SG-5010 típusú eszköz már 5,2kg/cm, míg az SG-91R műanyagfogaskerekes már csak 1,8 kg/cm ugyanezen árammal működtetve. Bolti áruk ebben a sorrendben feleződik.

Ezek a szervók 4,8V és 6V közötti árammal vezérelhetőek. Én az eszközöket 5V-al működtettem, így az előbb említett adatok mellett dolgoztam. Kisebb tárgyak felemelése nem okozott problémát. A feszültség növelésével nagyobb nyomaték érhető el, ez azonban hosszútávon csökkenti a szervók élettartamát.

A 3.15. ábrán az általam használt szervók láthatóak. Az alábbi képek a http://www.google.com képkeresési találataiból származnak.

3.15. ábra – TowerPro MG-995, SG-5010 és SG-91R típusú szervók

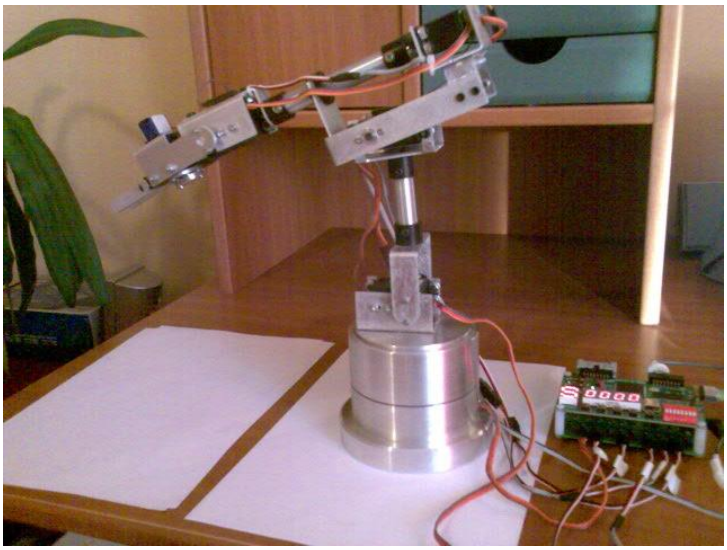
III. 5. A robotkar

A szakdolgozatom elkészítéséhez használt robotkart a National Instruments vállalat bocsátotta a rendelkezésemre. A robotkar nem tényleges ipari működésre készült, amely fizikai korlátaiban is megmutatkozik. Az eszköz az ipari robotok fizikai felépítését és működési elvét modellezi, lehetőséget nyújtva az ipari technológia és a robotkarok vezérlési elvének a megismerésére.

Felépítését tekintve megegyezik az általános ipari robotkarok szisztémájával:

- a mozgásért felelős motorokat szervók látják el,
- a megfogó részt két szorítópofa modellezi.

A robotkarhoz szintén biztosított a vállalat egy olyan kiegészítő-panelt, amely segítségével csatlakoztatni tudtam az eszközt a Logsys panelhez. A kiegészítő-panelre maximum 10 darab szervót csatlakoztathatunk, amelyek a Logsys panel megfelelő I/O portjára csatlakoznak. Az áramellátásról pedig egy külső tápegység gondoskodott, amelyet szintén a kiegészítő-panel osztott szét a megfelelő kimeneteken a szervók részére.
A robotkart a 3.16. ábrán láthatjuk összekötve a kiegészítő-panellel, amely a Logsys kártyához csatlakozik.



3.16. ábra

A dolgozat elkészítéséhez felhasznált robotkar, a hozzá csatlakozó kiegészítő-panel, valamint a Logsys kártya.

A robotkar mozgását 6 szervó motor látja el, amelyek segítségével mozgathatjuk vízszintes, illetve függőleges irányban, valamint az elején található „fogó” részt irányíthatjuk. Mindegyik szervó motort külön vezetéken keresztül csatlakoztathatjuk a kiegészítő panelhez, tehát nincs egységesen kötegelve. A robotkar a szervók erősségét figyelembe véve 3 részre különíthető el. Az alsó részen lévő motorok tartják a legtöbb súlyt, így ezek a szervók a legerősebbek, valamint ezen eszközök igénylik a legnagyobb áramerősséget. Ehhez a részhez 2 motor tartozik. Az alsó rész segítségével mozgathatjuk a robotkart vízszintes irányban, tehát jobbra és balra, illetve függőleges irányban, tehát fel és le. A második részen már gyengébb erősségű mozgatómotorok találhatóak, szám szerint három. Itt már ugyanis kevesebb súlyt kell tartania, kevesebb alkatrészt kell felemelnie vagy elforgatnia. Itt szintén a függőleges mozgást látják el, valamint a robotkar „fogó” részét mozgathatjuk közvetlenül le/fel, vagy forgathatjuk jobbra, illetve balra. Az utolsó rész a kar lényege, két fémpofát zárhatunk össze, mely a tárgyak megfogására szolgálhat.

Magát a robotkart a talpán lévő súly tartja meg, hogy a mozgássorozatok közben el ne billenjen oldalra. Nagyobb ívű mozgásoknál azonban az eszköz néha elbillent az egyensúlyából, így a rögzítésre is rá kellett segíteni. További fizikai korlátot jelentett a mozgatás során, hogy az alsó szervó közel vízszintes állapotból már nem volt képes felemelni a rá nehezedő alkatrészek súlyát.

IV. Az elkészült munka, a robotkar irányítása

A szakdolgozatom címe egy nagyobb témát jelent, amely magában foglalja egy ipari robotkar mozgatásának és vezérlésének a megvalósítását FPGA segítségével. Mivel az eszköz felépítése és működési elve hasonló az iparban használt rendszerekhez, így a mozgatás leprogramozásával és az elvi irányítás megértésével betekintést nyerhettem az iparban használatos technológiákba.

Az általam elkészített munka egy nagyobb terv része. Előttem már foglalkoztak a robotkar vezérlésével, de az eszköz csak egy előre letárolt mozgássorozatot hajtott végre. Az FPGA eszköz felprogramozása után a szervók egy előre letárolt kódsorozatot kaptak meg, így a robotkar ugyanazt a mozgást végezhette ismétlődően. Ezzel szemben én egy olyan dolgozatot készítettem, amelyben a kezelőnek lehetősége van a robotkar tényleges vezérlésére egy külső irányítópult segítségével.

A mozgatáshoz először egy olyan bemeneti eszközt kellett keresnem, amely a könnyű használat mellett alkalmas a szervók valós idejű irányítására. Így egy billentyűzet mellett döntöttem, amely segítségével egy kézi vezérlésű ipari robotkar-modellt készítettem. Mivel a robotkar működését 6 szervó látja el, így ha minden szervót mozgatni szeretnénk mindkét irányba a lehetőségeinek megfelelően, 12 gombra van szükségünk. A billentyűzet gombjai egyszerre több szervót is vezérelhettek volna, de így részletesebb a mozgás, és nem maradnak ki fizikailag elérhető részek a mozgástérből. Figyelembe kellett azonban vennem, hogy a Logsys panel nem rendelkezik USB, RS232 valamint PS/2 porttal sem. Így egy régebbi AT billentyűzetet csatlakoztattam a Logsys panel „A” portjára, melyhez egy vezetékkel külön csatlakozót készítettem. Mivel az AT billentyűzet PS/2 szabványt használ a kommunikációra, csak egy adat- és egy órajel-vonalra volt szükségem.

Azok az ipari robotok, amelyek folyamatosan ugyanazt a mozgássorozatot végzik egy adott pályán, a munkavégzés előtt felprogramozásra kerülnek. A beprogramozást általában egy, az elvégzendő feladathoz értő szakember végzi, ezzel optimális beállítást eredményezve. Ehhez rendelkezésre áll egy beviteli eszköz, amely általában egy irányító pult.

Mivel rendelkezésemre állt a billentyűzet, és az így elkészített modell megfelelt az előbb említett ipari szisztémának, ezért úgy döntöttem, hogy készítek egy tárolási funkciót is. A billentyűzet segítségével lehetőségünk van tehát eltárolni egy mozgássorozatot a Logsys panel memóriájában, majd később onnan visszaolvasva azt újra lefuttatni.

Így egy olyan általános ipari eljárást programoztam le, amelyben egy kezelő a robotkar mozgási pályáját kézzel állítja be úgy, hogy egy beviteli eszköz segítségével végigvezeti a robotkart az elvégzendő feladat lépésein. Egy ilyen módon eltárolt mozgássorozatot a tárolás végeztével újból folytathatunk, ha esetleg még kifelejtettünk volna pár billentyűt. Természetesen törölhetjük is a memóriát, ám ilyenkor nem lesz a memóriában lefuttatható mozgássorozat, így ha le akarjuk játszani, a robotkar nem fog megmozdulni. Billentyűzetről azonban továbbra is vezérelhetjük, kivéve akkor, ha előre letárolt billentyűsorozatot játszunk vissza. Alaphelyzetbe a Logsys panel Reset gombjával hozhatjuk, ekkor visszaáll a robotkar kezdőállapotba. Reset gomb megnyomására a memória nem törlődik.

Az ipari robotokat nemcsak ilyen módon lehet beprogramozni, hanem lehetőség van pálya koordinátáinak megadásával is vezérelni. Ennek a módszernek a megvalósítása szerepel a további tervek között, mivel a szakdolgozatomban erre már nem került sor. Az eddig elvégzett munkám tehát folytatódni fog, kiegészülve grafikus felhasználó felülettel és a mozgások finomításával. Ezt azonban már más hallgatók fogják elvégezni.

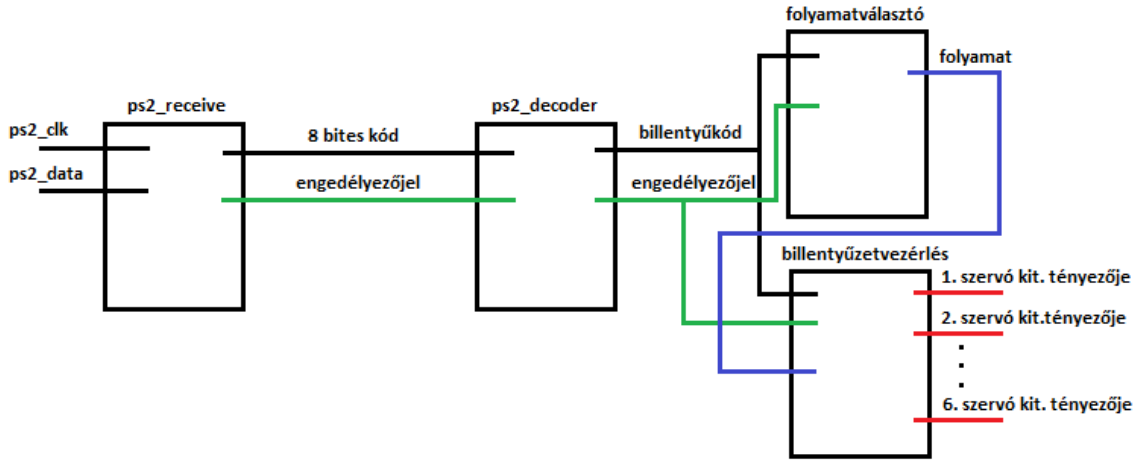
Mivel a National Instruments vállalat eszközeinek nagy része FPGA alapú termék, ezért a robotkar szakdolgozattémájának kiírásakor szintén FPGA-t választottak a vezérlésre. A robotikában működési alapelv a párhuzamosság, az egyes működési egységeknek folyamatos összhangban kell lenniük egymással, és mindig az aktuális információval kell rendelkezniük egy adott helyzetről a következő mozgás kiértékelése végett. Ezért is volt ideális választás egy FPGA-val vezérelni a robotkart, ugyanis ez az eszköz a párhuzamos rendszerek tervezését teszi lehetővé. Ezenkívül gyors és nagyteljesítményű számításokat igénylő rendszereknél is alkalmazhatóak, amely fontos szempont a robotikában egy helyzet gyors kiértékelésénél.

Az FPGA-ra készített kódot Verilog nyelven írtam, a kódot a nyelv lehetőségeit kihasználva több modulra bontottam. A Xilinx ISE Webpack modulszintű sematikus ábráját, valamint a saját, eltervezett modulfelépítést használva mutatom be a program működését. A kód három nagyobb részre osztható. Az első a billentyűzet kezelését, a második a memória kezelését, a harmadik pedig a szervók vezérlését látja el.

IV. 1. A program részegységeinek működése

IV. 1. 1. Billentyűzetvezérlés

A billentyűzet, amelyet használtam, a már korábban említett és bemutatott PS/2 kommunikációs szabványt használja adatközlésre. Csak egyirányú kommunikációt valósítottam meg, tehát a billentyűzet számára nem küldök ki adatokat. Az adatfogadásra, illetve ezek átalakítására, a szervó motorok számára értelmezhető kitöltési tényező előállítására, 3 modult programoztam. Ebben a részben ismertetnék továbbá egy folyamatválasztó modult. A következőkben bemutatásra kerülő programrészek sematikus ábráját a 4.1. ábra szemlélteti. Az ábra egyszerűsítése érdekében csak a fontosabb adatvonalakat jelenítetem meg.



4.1. ábra

A ps2_receive modul bemenetként a Logsys panel 16MHz-es órajelét, egy Reset jelet, valamint a billentyűzet órajelvonalát, illetve adatvonalát várja. A modul feladata, hogy egy adott billentyű lenyomásakor a billentyűzet által küldött 8 bites adatsorozatot fogadja, és eltárolja. Mivel a 8 db bit egymás után érkezik az adatvonalon, egy shift regiszter fogadja és tárolja el azokat. A megérkezett adat helyessége kezelve van a paritás bit figyelésével. A modul kimenetként a 8 bitet szolgáltatja, valamint egy engedélyező jelet. Az engedélyező jel „1” szintje jelzi, ha sikeresen megérkezett és eltárolásra került az információ, különben „0” logikai szintű.

A ps2_decoder modul a ps2_receive modul által szolgáltatott 8 bitet és az engedélyező jelet várja. A részegység feladata, hogy a bejövő kódokból előállítsa a lenyomott billentyű kódját, azaz elkülönítsen 1 darab 8 bites kódot a beérkező adatokból. Az elkülönítés alatt azt értem, hogy egy billentyű lenyomásakor 3x8bit érkezik az adatvonalon. Ez a három információ a billentyű kódja, egy hexadecimális „F0” számpár, ami a billentyű felengedését jelzi, valamint ismételten a billentyű kódja. Ez a modul tehát figyeli a billentyű felengedését, az „F0”-t, és a 3x8bitből meghatározza a billentyűkódot. Szintén kezelve van a hosszan tartó nyomva tartás, azaz ha nem engedjük fel a billentyűt egy bizonyos ideig. Ekkor ugyanis a billentyű kódját nem az F0 követi, hanem folyamatosan ismétlődik egy bizonyos késleltetéssel. A modul ekkor ugyanúgy elkészíti a billentyűhöz tartozó 8 bites adatot, és egy engedélyező vonal segítségével jelzi, hogy billentyűnyomás történt, valamint elkészült a kód.

Ezt az elkészített billentyűkódot több modul is megkapja a hozzá tartozó engedélyező jellel együtt. Többek között ide tartozik egy folyamatválasztó programrész, amely a lenyomott billentyű függvényében eldönti, hogy a szervó motorok honnan kapják a vezérlést, a billentyűzetről vagy a memóriából. Szintén itt kerül eldöntésre, hogy a memóriába tárolunk-e, vagy a billentyűzetről irányítunk, esetleg az eltárolt mozgássorozatot olvassuk-e vissza. A kimenete tehát egy 3 bites adatvonal, melynek néhány fontosabb decimális értéke a következőket jelenti:

- 1, ha a billentyűzetről irányítunk, ez az alapértelmezett
- 3, ha elindítjuk a memóriába való tárolást
- 4, ha kiolvassuk a tárolt kódot
- 5, ha töröljük a memóriát.

A folyamatválasztó modul a billentyűkódon kívül más jelzővonalakat is vár bemenetként, amelyek segítségével dönteni tud. Ilyenek például a memóriából való kiolvasás végét jelző, illetve a memória törlésének végét jelző adatvonalak. A kimenetét az összes irányítással kapcsolatos modul megkapja. Ide tartozik többek között az a programrész, amely a lenyomott billentyűnek a kódját megvizsgálva az adott szervó kitöltési tényezőjét egy fix értékkel változtatja pozitív vagy negatív irányban. Ez a 4.1 ábrán a billentyűvezérlés nevet kapta.

Kezdeti problémát jelentett, hogy az általam használt billentyűzet első gombnyomásra nem az előbb említett kódsorozatot küldte ki a megfelelő adatvonalon. Második, illetve további billentyűk lenyomásakor azonban már a megfelelő kód került kiolvasásra. A problémát nem a PS2 protokollt feldolgozó kódrészlet jelentette, hanem a billentyűzet küldött rossz kódsorozatot. Ennek okát nem sikerült megfejtenem. A problémát úgy oldottam meg, hogy a billentyűzetről való vezérlést egy engedélyező billentyűlenyomástól tettem függővé. Erre a feladatra a „Space" billentyűt választottam. A hiba felderítésére külön kódot írtam, ami kijelzi a fogadott kódsorozatokat, így ellenőrizni tudtam azokat.

Az először leütött gomb kódja a következő eltéréseket mutatta (a kódok 8 biten ábrázolt hexadecimális értékeket jelentenek):

- Az „A" billentyű kódja első lenyomásra:
  - 70 C1 1C
- Az „A" billentyű kódja már egy előzőleg megnyomott gomb után:
  - 1C F0 1C
- A „B" billentyű kódja első lenyomásra:
  - C8 C1 32
- A „B" billentyű kódja már egy előzőleg megnyomott gomb után:
  - 32 F0 32

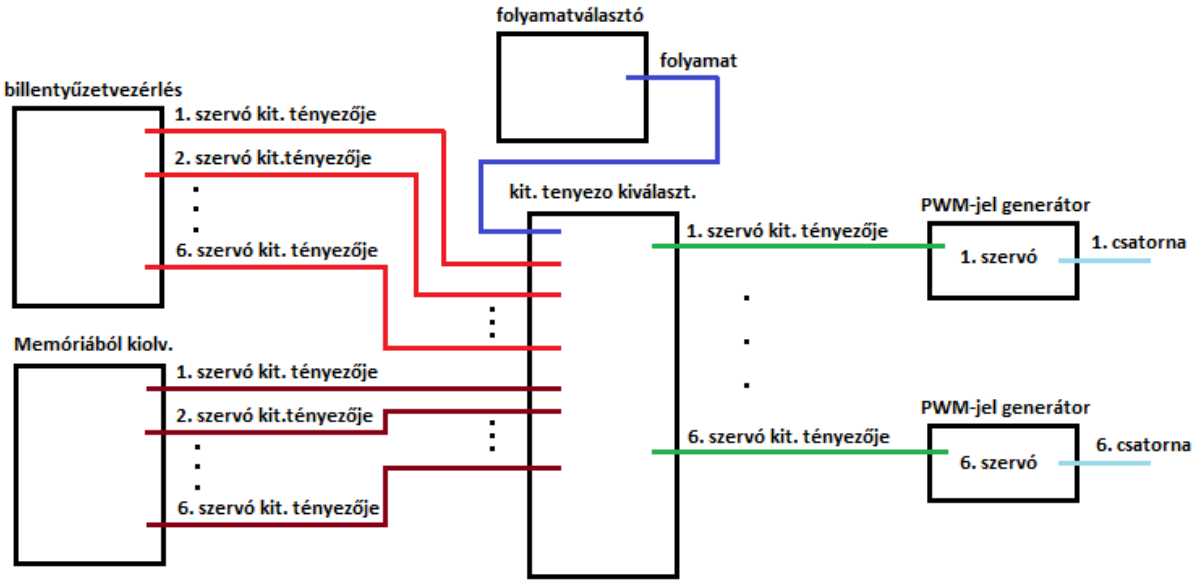
Ebből a példából látszik, hogy a kódok valóban eltértek egymástól, azonban a záró 8 bit megegyezett.

Mivel már az elkészült kódfeldolgozó modul az eredeti, 3.11. ábrán szereplő hexadecimális értékeket dolgozta fel, ezért a problémát úgy oldottam meg, hogy csak akkor dolgozza fel az adatokat, ha már történt billentyűnyomás, és az a billentyű a „Space" volt. A 3. bitsorozat vizsgálatával így ki tudtam szűrni, hogy „Space" került-e lenyomásra, az őt követő billentyűkódok feldolgozása pedig a szabvány szerint folytatódhatott.

IV. 1. 2. Szervóvezérlés

A szervók vezérlésének alapja a PWM jel. Az egyik és legfontosabb modul tehát egy PWM jel generáló programrész, amely bemenetként megkapja a kimeneti csatornát, tehát, hogy melyik szervót fogja irányítani. Szintén bemenetként adjuk át neki a kitöltési tényezőt, amelynek változtatásával érjük el a motor mozgását. Ez a programrész hatszor kerül példányosításra a főmodulban, így a 6 szervómotor párhuzamosan kapja meg a számára szükséges adatot. Biztosítva van továbbá az is, hogy a motorok 20ms-onként értesülnek a PWM jel kitöltési tényezőjéről. Ez azért szükséges, mert a használt eszközök analóg elven működnek, és ahhoz, hogy a motor elérje a kívánt pozíciót és meg is tartsa azt, erre a periódusidőre van szükség.

A következő fontos kódrész az impulzusszélesség modulált jelek kiválasztásáért felelős. Ez a kitöltési tényezőt kiválasztó modul bemenetként a folyamatválasztó modul kimenetét kapja meg. Ennek segítségével eldönti, hogy a PWM jel generáló modul bemenetként milyen kitöltési tényezőt használjon fel. Ha billentyűzetről irányítunk, akkor a billentyűzetről való vezérléshez tartózó értékek, ha a memóriába tárolunk, akkor az ehhez tartozó adatatok, ha pedig a memóriából olvassuk ki az értékeket, akkor annak megfelelő jeleket fog továbbítani kimenetként. A 4.2. ábra a szervóvezérléshez kapcsolódó fontosabb modulok sematikus rajzát, valamint a fontosabb adatvonalakat jeleníti meg.



4.2. ábra

IV. 1. 3. Memóriakezelés

Az adatok tárolására a Logsys panelen található 128Kx8 bites SRAM-ot használtam. A gyors, vagyis 10ns-os címhozzáférési idő miatt az adatok írásához és olvasásához nem volt szükség külön késleltetésre. Ennek oka az, hogy a Logsys panel belső 16MHz-s oszcillátora 62.5 ns-os órajelet generál, így ezt használtam az írási/olvasási ciklusnál. A memória három vezérlő adatvonallal rendelkezik, melyek segítségével irányíthatjuk az írási/olvasási folyamatokat:

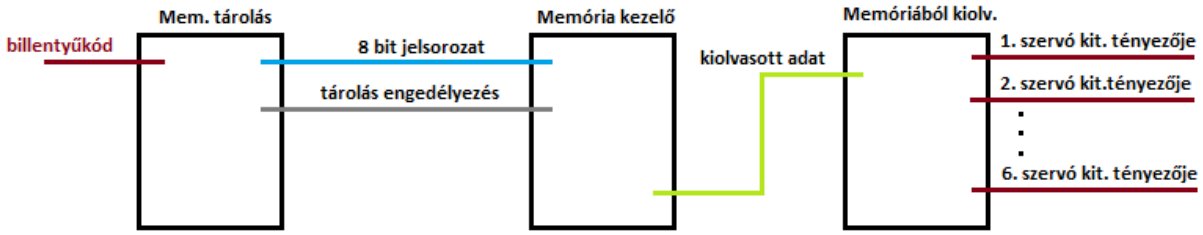
- CS – Chip Select. Ezt akkor használhatjuk, ha több memória áll a rendelkezésünkre, és így válthatunk közöttük az adott eszköz tiltásával, illetve engedélyezésével.
- WE – Write Enable. Ezzel a vonallal engedélyezzük az írást.
- OE – Output Enable. Írás esetén ennek a vonalnak a logikai értéke „Don't care", azaz független tőle az írás. A memóriából való kiolvasás esetén viszont ezen jel segítségével jelezhetünk.

A következő, 4.3 táblázat az adatvonalak lehetséges logikai értékeit ábrázolja a hozzájuk kapcsolódó állapotokkal. Az „X" azt jelenti, hogy értéke mindegy a vezérlés szempontjából. Mivel a memória negatív logikát használ a vezérlésre, ezért a táblázatban a negált értékeket jelenítem meg. Az írási és olvasási műveletek elvégzésekor, a következő logikai értékeket kiküldve az adatvonalakon, elérhetjük a kívánt funkciókat.

CS – Chip Select	WE – Write Enable	OE – Output Enable	Funkció
1	X	X	A memória nincs kiválasztva.
0	1	1	A memória kimenete tiltva van, így nem tudunk olvasni.
0	1	0	Engedélyezve van az olvasás.
0	0	X	Engedélyezve van az írás.

4.3 táblázat

A memóriakezeléssel kapcsolatos fontosabb modulok és adatvonalak sematikus rajzát a 4.4. ábra szemlélteti.



4.4. ábra

A három modul közül a „memóriakezelő” elnevezésű látja el a tényleges memóriakezelést. Itt történik a memória adatvonalainak a vezérlése. A tárolóeszközhöz szintén tartozik egy 8 bites adatbusz, amin keresztül egy adott címre adatot írhatunk, illetve onnan adatot olvashatunk ki. Az adatbusz egy kétirányú vonal, tehát viselkedhet bemenetként vagy kimenetként is. Fontos tulajdonsága, amit a memória megkövetel, hogy ha nem mozgatunk rajta adatot egyik irányban sem, akkor nagyimpedanciás állapotban kell lennie. Ez a programrész ezt a követelményt is kielégíti. Mivel csak egy memóriát használtam, így a „Chip Select” vonallal nem kellett foglalkoznom, fix logikai „0” szintre állítottam. Az írás, olvasás és a törlés művelete szintén ebben a modulban történik. Az ezekhez szükséges megfelelő adatokat a tárolás és a kiolvasás modultól kapja meg a programrész. Szintén bemenetként jelenik meg a folyamatválasztó kódrész kimenete, ugyanis ennek segítségével dönti el, hogy éppen milyen műveletet végzünk.

A tárolás programrész bemenetként a ps2_decoder modul kimeneti billentyűkódját várja, amelyet átalakítottam a következő 8 bites formára: X_XXXX_XXX. Ez a modul szintén szolgáltat kitöltési tényezőt a szervó motorok számára. Ennek oka, hogy miközben tároljuk a mozgássorozatot, megjelenítjük a mozgást is. Így látjuk azt, hogy mit fogunk kapni a visszaolvasáskor. A tárolás befejeztével pedig a billentyűzetről való vezérlés az adott állapotból folytatható. A 8 bit jelentése a következő, balról jobbra haladva:

- Az első helyen szerepelhet 0 vagy 1, melynek jelentése:
    - 0 – a kitöltési tényezőt csökkentjük
    - 1 – a kitöltési tényezőt növeljük

- A 2-5 helyen szereplő középső 4 bit az értéket jelenti, vagyis hogy mennyivel kell csökkenteni, illetve növelni.
- Az utolsó 3 bit pedig a szervó sorszámát jelenti, vagyis hogy melyik motorra vonatkoznak az előbb említett adatok.

Az így előállított jelsorozat tehát a billentyűzet kódjától függ, amely átadásra kerül a memória modulnak. Ez kerül tárolásra egy adott címre. Szintén ebben a modulban tartunk nyílván egy számlálót, amelynek értéke 200-ról számol vissza, és a letárolható billentyűk számát jelenti. Értéke természetesen növelhető, de a tesztelésekhez ez az érték megfelelő volt. Miután az adatok eltárolásához szükséges értékek előálltak és átadódtak a memória modulnak, megkezdődhet az adatok beírása egy adott címre. A memóriacímek 0-ról növekszenek, tehát ez lesz az első címünk. A memória adatlapján többféle lehetőség áll rendelkezésünkre az írási ciklust illetően, melyek közül én az „Output Enable” vonal fix logikai 0 szintre állításával működőt választottam. Ez a következő sorrendben végzi az írást a megfelelő késleltetésekkel:

1. Beállítjuk a kiolvasandó memóriacímet a címvonal segítségével.
2. Kiválasztjuk az eszközt a „Chip Select” vonallal. (Több eszköz esetén).
3. A „Write Enable” vonalat logikai 0 szintre állítjuk.
4. Az adatbuszon megjelenítjük a beírandó adatot.

Törlés esetén az összes felhasznált címre decimális 0 érték íródik 8 biten ábrázolva.

Ha eltároltuk a megfelelő értékeket, az adatok visszaolvasása a kiolvasás modul segítségével történik. Ez a modul elküld egy olvasási engedélyezőjelet a memória modulnak, amely után visszakapja a kiolvasott 8 bites adatokat. Ezután visszafejti a kód jelentését, és elvégzi a kitöltési tényezőkön a megfelelő változtatásokat. A memória adatlapján szintén több kiolvasási időzítést programozhatunk le, én a címvezéreltet választottam. Ennek megfelelően az olvasás a következő sorrendben történik:

1. Az eszközt kiválasztjuk a „Chip Select” vonallal. (Több eszköz esetén).
2. Az „Output Enable” vonalat logikai 0 szintre, a „Write Enable” vonalat logikai 1 szintre állítjuk.
3. Beállítjuk az olvasandó memóriacímet a címvonal segítségével.
4. Az adatbuszról beolvassuk a 8 bitet.

A kiolvasás fix gyorsasággal történik, így a robotkar mozgási sebessége is kötött lesz. Az így folyamatosan változó kitöltési tényezők értékeit megkapja a már korábban ismertetett „kitöltési tényezőket kiválasztó modul”, a szervók pedig működésbe lépnek. Kiolvasás közben nincs lehetőség a billentyűzetről irányítani, maga a folyamat pedig csak a Reset gomb segítségével szakítható meg. Ilyenkor alapállapotba áll vissza a robotkar. Itt jegyezném meg, hogy a kiolvasás induló állapota mindig egy előre meghatározott érték, törlés után a tárolás megkezdését mindig ebből az előre beállított állapotból tehetjük meg.

IV. 1. 4. Kijelzők

A Logsys panel elérhető kijelzői állapot-visszajelzőként szolgálnak. A 8 db Led kijelzőn a billentyű kódjának 8 bitje jelenik meg. A 4 db 7 szegmenses kijelzőn a memóriába való tároláskor a még tárolható billentyűk száma jelenik meg, értéke egy-egy új bevitelnél értelemszerűen csökken. Visszaolvasáskor pedig értéke 0-ról nő a bevitt adatok darabszámáig. Ha billentyűzetről vezéreljük a robotkart, akkor a 7 szegmenses kijelzőn nem történik értékváltozás, mind a 4 darab 0 értéket jelez. A pontmátrix-kijelzőt betűk megjelenítésére használtam. Ha billentyűzetről irányítunk, akkor egy „B” betű, ha a memóriába írunk, akkor egy „M”-betű, ha pedig a memóriából kiolvasunk, akkor egy „K” betű jelenik meg rajta.

V. Köszönet

Ezúton szeretném megköszönni Nagy Gábornak és a National Instruments vállalatnak, hogy rendelkezésemre bocsátották a Gábor közreműködésével készült eszközöket. Eszközök alatt a robotkart és a kiegészítő panelt értem.

Szintén köszönöm Dr. Végh János Tanár Úrnak, hogy a téma kiírásával és a külső kapcsolatok megteremtésével lehetőséget kaptam FPGA eszközökön fejleszteni, illetve megismerni azokat.

És legvégül köszönet jár Vitéz László csoporttársamnak is, aki a kezdeti nehézségeken segített át, és biztosította a működéshez szükséges áramellátó eszközt.

VI. Összegzés

A szakdolgozatom egy olyan összetett témát foglal magában, amely egy ipari robotkar működésének a modellezését tűzte ki célul. Először egy kezdeti tervet készítettem, amelyben meghatároztam a megoldandó feladatokat és az elvégzendő munkafolyamatokat. Mivel kiderült, hogy a teljes munka elvégzésére nem elegendő a szakdolgozatírásra szánt idő, ezért a témavezetőmmel egyeztetve meghatároztuk az általam elvégzendő feladatrészt.

Már előttem is foglalkoztak a vezérlés megoldásával, így dolgozatom elkezdése előtt már egy olyan vezérlési mód készült el, amelyben a robotkar egy előre letárolt mozgássorozatot futtatott le folyamatosan [4]. Az ezt követő lépés célja az volt, amelyet a munkám során készítettem el, hogy egy olyan résszel bővüljön a terv, melyben lehetőség van a robotkart irányítani valamilyen külső vezérlőeszköz segítségével. Erre a feladatra egy billentyűzetet választottam, de mivel a Logsys FPGA kártya billentyűzet csatlakoztatására nem volt felkészítve, ezért egy erre alkalmas csatlakozót készítettem. Az így elkészült munka egy ipari robotkar kézi vezérlését modellezi, ugyanis a billentyűzet minden gombjához a robotkar egy adott részének mozgatása tartozik adott irányba. Ezenkívül a kezelőnek lehetősége van egy mozgássorozatot letárolni, majd később visszajátszani azt. Így a kézi vezérlés mellett önálló, folyamatos működésre is képes. Ez az elv hasonlít az iparban alkalmazott és bevált eljárásokhoz, ahol a folyamatos, ismétlődő munkavégzést elvégző robotokat először felprogramozzák az elvégzendő feladatnak megfelelően, majd a robotkar az előzőleg letárolt értékeket felhasználva látja el a feladatát.
Az iparban a mozgássorozatokat többféleképpen állíthatják be, melyek általában a mozgás pályájának meghatározásából állnak. Ilyen elterjedt módszer a robotkar végigvezetése az adott pályán, amelyet az én munkám is modellez.

A terv végleges célja egy olyan ipari robot vezérlésének a modellezése, amelynek felprogramozása nemcsak irányítóeszközről, hanem egy grafikus felületű programból is lehetséges, de erre a szakdolgozatomban már nem került sor. A program lehetőséget nyújtana a mozgási pálya egy másik módszer szerinti beállítására, amely a koordináták megadásával való felprogramozást jelenti. Ezenkívül a pontosságot igénylő munkafolyamatok elvégzése miatt a robotkar mozgásának finomítása, illetve egy elterjedtebb külső vezérlőeszköz használata (botkormány) is a további elvégzendő feladatok közé tartozik.

VII. Irodalomjegyzék

1. Dr. Csáki Tibor – Robottechnika
2. Makó I. – Robottechnika előadásvázlat
3. Fehér Béla – Digitális rendszerek tervezése FPGA áramkörökkel
4. Pong P. Chu – FPGA Prototyping by Verilog Examples
5. Vitéz László – FPGA alapú robotkarvezérlés megvalósítása
6. http://en.wikipedia.org/wiki/Industrial_robot
7. http://www.villanyszaklap.hu/cikkek.php?id=397
8. http://www.computer-engineering.org/ps2protocol/
9. http://www.beyondlogic.org/keyboard/keybrd.htm
10. http://pcbheaven.com/wikipages/The_PS2_protocol/
11. www.xilinx.com
12. http://en.wikipedia.org/wiki/Verilog
13. http://en.wikipedia.org/wiki/Field-programmable_gate_array
14. http://www.freeweb.hu/t-t/elokep/pic/felhkk/kk/sz1604.htm
15. http://en.wikipedia.org/wiki/Hardware_description_language
16. http://myprojects.hu/pic/szervo-motorok-vezerlese.html
17. http://www.seattlerobotics.org/guide/servos.html
18. http://sites.google.com/site/robotsaustralia/servosforrobotics2
19. http://www.best-microcontroller-projects.com/servo-motor.html
20. http://logsys.mit.bme.hu

VIII. Melléklet

I. sz. melléklet: Használati útmutató

II. sz. melléklet: Képek a robotkarról

I. Használati útmutató

Billentyű	Vezérlő funkció
Reset (Logsys panel)	Alapállapotba állítás, a szervó motorok felveszik a kezdőállapotot. A memória nem törlődik.
Space	Billentyűzetről való vezérlés engedélyezése, Reset után szintén alkalmazandó.
F9	A memóriába való tárolást indítja el.
F10	A memóriába való tárolást állítja meg, a vezérlés tovább folytatódhat a billentyűzetről az adott pozícióból.
F11	A memóriában tárolt kódsorozat visszaolvasása és a tárolt mozgássorozat visszajátszása adott sebességgel. Alapállapotból indul.
F12	A memóriában tárolt kódsorozat törlése.

Megjegyzés: A memóriában tárolt kódsorozatot lehetőségünk van folytatni az F9 gomb újbóli megnyomásával.

Billentyű	Vezérlő funkció
A	3. szervó mozgatása balra
D	3. szervó mozgatása jobbra
W	2. szervó mozgatás fel
S	2. szervó mozgatása le
Q	1. szervó által vezérelt szorítópofák nyitása
E	1. szervó által vezérelt szorítópofák zárása
J	6. szervó mozgatása balra
L	6. szervó mozgatása jobbra
K	4. szervó mozgatása le
I	4. szervó mozgatása fel
U	5. szervó mozgatása le
O	5. szervó mozgatása fel

II. Képek a robotkarról