Debreceni Egyetem

Informatikai Kar

FPGA alapú robotkarvezérlés megvalósítása

Témavezető:

Dr. Végh János

Egyetemi tanár

Készítették:

Fórizs Zoltán

Kócsi György

Mérnök informatikus hallgatók

Külső konzulens:

Nagy Gábor

Villamosmérnök

Debrecen

2010

„A robotika három törvénye:

1. *A robotnak nem szabad kárt okoznia emberi lényben, vagy tétlenül tűrnie, hogy emberi lény bármilyen kárt szenvedjen.*
2. *A robot engedelmeskedni tartozik az emberi lények utasításainak, kivéve, ha ezek az utasítások az első törvény előírásaiba ütköznének.*
3. *A robot tartozik saját védelméről gondoskodni, amennyiben ez nem ütközik az első vagy második törvény bármelyikének előírásaiba."*

(Isaac Asimov)

Tartalomjegyzék

1. Bevezetés, témaválasztás indoklása

Napjaink informatikai rendszereinek teljesítményét már nem csupán a nyers számítási teljesítménye jellemzi, hanem a párhuzamosíthatóság lehetősége is. A személyi számítógépek piacán is komoly térhódítást nyertek a többmagos processzorokat tartalmazó rendszerek. Párhuzamos működés szempontjából egy másik alkalmas eszköz az FPGA, melyet a Xilinx cég fejlesztett ki, és dobta piacra először 1984-ben. (Az FPGA-ról részletesebben a 2.1 fejezetben számolunk be.)

Az Informatikai Karon először a Baross program keretein belül Dr. Végh János tanár úr biztosította a lehetőséget a hallgatóknak, hogy megismerkedhessenek ezzel a technológiával. Az órák során a Budapesti Műszaki és Gazdaságtudományi Egyetem Méréstechnika és Információs Rendszerek tanszék docense, Dr Fehér Béla segítségével elsajátíthattuk egy ilyen rendszer működési elvét, valamint lehetőségünk nyílt egy oktatási célra készült fejlesztői panelen kisebb projektek megvalósítására. A csoporton belül sok hallgatónak felkeltette az érdeklődését ez a technológia, ezért úgy döntöttünk, hogy a Szakdolgozat tantárgy keretein belül még jobban szeretnénk elmélyülni eme technológia rejtelmeiben. Ebben nagy segítségünkre volt Dr. Végh János, aki többféle szakdolgozat-téma kiírásával biztosította a lehetőséget a hallgatók számára FPGA-témájú diplomamunka készítésére, ezen kívül saját ötletek megvalósítására is biztosított lehetőséget. Közösen jelentkeztünk az „FPGA alapú robotkarvezérlés megvalósítása” című szakdolgozatra, mivel érdekelt minket a robotika világa is, valamint az, hogy egy ilyen rendszert miként lehet FPGA-technológiával vezérelni. A feladatot csoportmunkában valósítottunk meg.

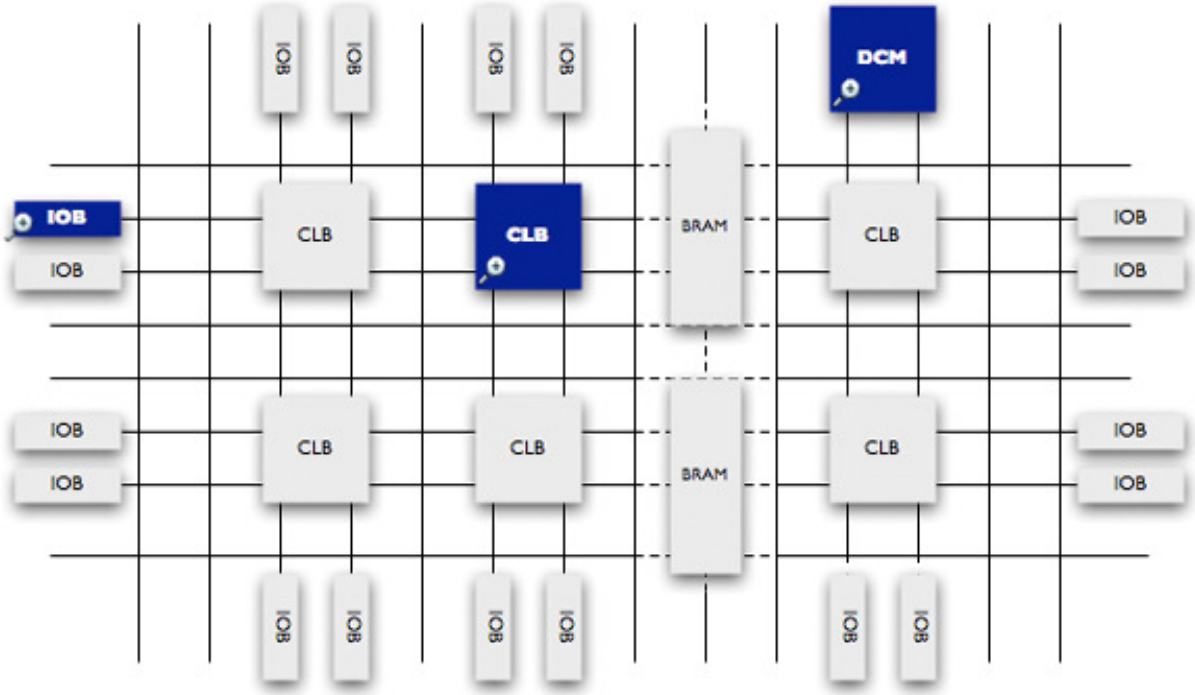
A munka kezdetekor a tárgyi feltételek közül az FPGA fejlesztői kártyát az Informatikai Rendszerek és Hálózatok tanszék, a robotkart pedig a National Instruments biztosította számunkra. Ez a vállalat már évek óta foglalkozik FPGA-rendszerek fejlesztésével, így ott tapasztalt villamosmérnökök segítettek bennünket a felmerülő problémák megoldásában.

Mivel ez a tudományterület új volt számunkra, így alapos elméleti és gyakorlati utánajárás volt szükséges a munka kezdetekor. Dolgozatunk első részében összefoglaljuk a témához szükséges elméleti ismereteket, később bemutatjuk az elkészült projektet.

2. Elméleti háttér

2.1 FPGA

Az FPGA (Field-Programmable Gate Array = helyszínen programozható, logikai kapukat tartalmazó tömb) egy olyan félvezető eszköz, amelyik "logikai blokk"-oknak nevezett programozható logikai komponenseket(**CLB**), programozható összeköttetéseket, I/O blokkokat(**IOB**), órajelkezelő-egységeket(**DCM**), illetve memóriablokkokat tartalmaz. Ezek az áramköri elemek a felhasználó által szabadon programozhatóak. Olyan általános célra programozható, mely digitális logikát valósít meg, mivel analóg jelet önmagában nem képes kezelni. Előnye, hogy az áramkörön belül párhuzamos a jelterjedés, így alacsony frekvencián (~10-100MHz) is nagy a számítási teljesítménye. Jól használható beágyazott rendszerekben. A logikai blokkokon belül logikai kapuk találhatóak, részben ezek száma, valamint a memóriablokkok mérete jellemzi az FPGA teljesítményét.



1. ábra: FPGA általános felépítése (forrás: http://www.xilinx.com)

Ezek az eszközök programozhatóság alapján a következőképpen csoportosíthatóak:

Egyszer programozható FPGA:

- OTP FPGA *(One Time Programmable FPGA)*: Az ilyen eszközök a legyártásuk után csak egyszer programozhatóak fel, így a beleírt program véglegesen megmarad, nem törölhető. Egyes típusaiknál elérhető az inkrementális javítás, vagyis a már bent levő program nem változtatható meg, de bizonyos esetekben további funkcionalitással bővíthető. Előnyük, hogy kisméretűek, alacsony fogyasztásúak, és mivel a program nem törölhető, így biztonságosan ki lehet olvasni.

Többször programozható FPGA:

- FLASH FPGA: Ebben az esetben a program egy Flash-memóriába kerül, az FPGA minden indulásakor innen olvassa be a programot. A program a tápfeszültség megszűnése esetén nem vész el. Ezeknél a típusoknál már megvan az újraprogramozhatóság lehetősége, így előnyük, hogy törlés után újra lehet programozni, ezáltal az eszköz más feladatok végrehajtására is használható.
- SRAM FPGA: A legelterjedtebb típus, CMOS technológiával. Előnye az alacsony ár, valamint bármikor átprogramozható. Hátránya, hogy a tápfeszültség megszűnése esetén a memória törlődik, valamint a benne tárolt program könnyebben megsérülhet.

FPGA eszközök legnagyobb gyártói:

- Xilinx (http://www.xilinx.com)
- Altera (http://www.altera.com)
- Actel (http://www.actel.com)

Szakdolgozatunkban egy, a Xilinx cég termékét tartalmazó panelt használunk fel, így a továbbiakban az ő eszközeit ismertetjük.

A Xilinx céget két villamosmérnök, Ross Freeman és Bernard Vonderschmitt alapította 1984-ben. Itt fejlesztették ki, és dobták piacra az első FPGA-t. Jelenleg az Altera mellett a Xilinx számít az egyik legnagyobb beszállítónak. Kínálatukban két fő termékcsalád található meg:

- Virtex-családba tartozó FPGA-k: Teljesménycentrikus, ipari felhasználásra szánt eszközök, melyeken nagyságrendekkel több erőforrás (CLB,IOB,memória)

található, valamint működési frekvenciájuk is lényegesen magasabb. Egyes típusaik beépített PowerPC processzort is tartalmaznak.

- Spartan-családba tartozó FPGA-k: Költséghatékony eszközök, általános felhasználási célokra optimalizált erőforrásokkal, és alacsonyabb működési frekvencia jellemzi őket.

Xilinx FPGA-k azonosítása típusszám alapján:

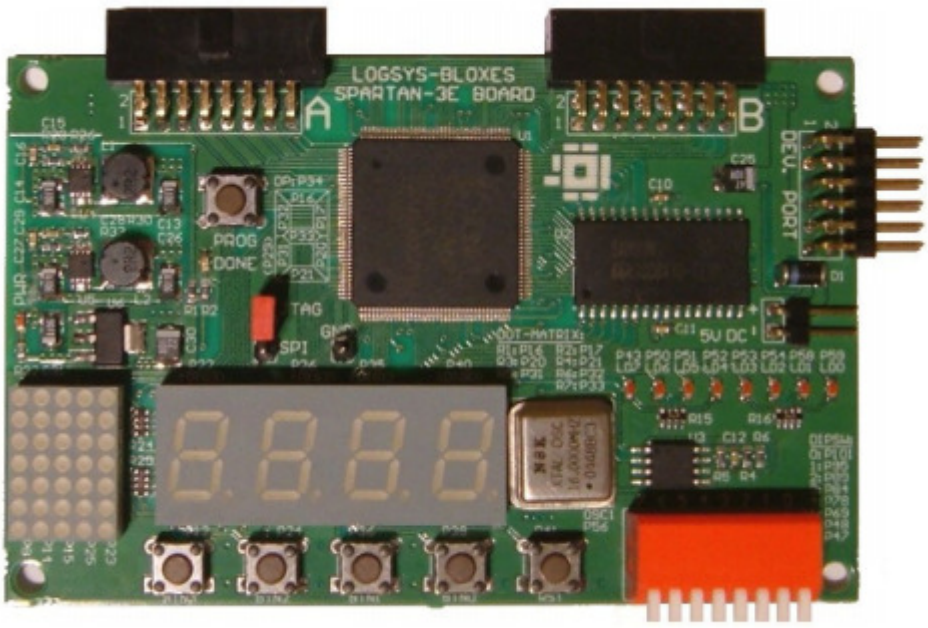
XC3S250E

- **XC**: Minden FPGA eszköz típusszáma ezzel a 2 betűvel kezdődik.
- **3S**: A Xilinx-termékcsalád nevét jelöli. (Esetünkben: **S**partan**3**)
- **250**: Az elérhető kapuszám mennyisége ezres nagyságrendben.(Esetünkben 250000 db)
- **E**: Opcionális betűjel, ha szerepel akkor az FPGA kapuszám-orientált, ha nem, akkor I/O-blokk-orientált

2 ábra: Xilinx Spartan XC3S250E típusú FPGA(forrás: Google képkeresés találat)

2.1.1 Logsys – panel

A szakdolgozatunk gyakorlati megvalósításához a LOGSYS Spartan-3E kártyáját használtuk:



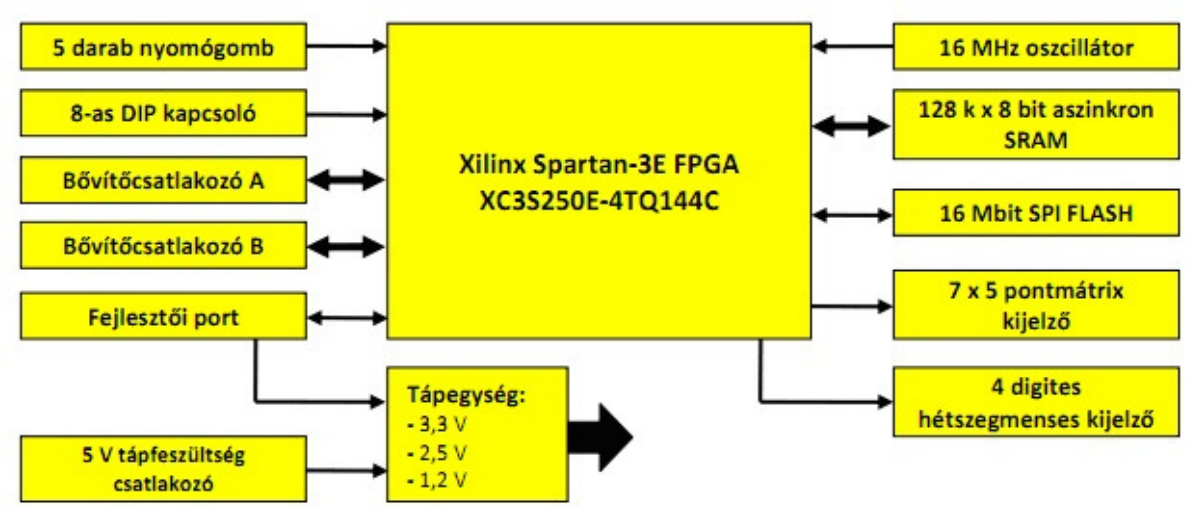
3. ábra: Logsys-panel (forrás: Logsys Felhasználói útmutató)

A kártya egyszerű felépítésű, kezdő felhasználók számára és oktatási célokra készült FPGA kártya, de összetettebb tervek kivitelezésére is alkalmas.

A kártyán az alábbi komponensek találhatók:

- Xilinx XC3S250E - 4TQ144C típusú FPGA
  - 250 ezer kapu (4896 LUT és flip−flop)
  - 12 darab 18 kbites blokk−RAM
  - 12 darab 18 x 18 bites előjeles szorzó
  - 4 darab DCM (Digital Clock Manager) modul
- Memóriák a program és az adatok tárolására:
  - Egy 128 k x 8 bites, 10 ns-os aszinkron SRAM
  - Egy 16 Mbites SPI buszos soros FLASH memória
- Megjelenítő eszközök:
  - 8 darab LED

  - 4 digites hétszegmenses kijelző
  - 7 x 5 pontmátrix kijelző
- Beviteli eszközök:
  - 5 darab nyomógomb
  - 8 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)
  - 5 V és 3,3 V tápfeszültség kimenet



4. ábra: Logsys-panel blokkvázlata (forrás: Logsys Felhasználói útmutató)

A kártyán található FLASH memória konfigurációs memóriaként is szolgál.

A LED-ek LD0-LD7-ig vannak számozva. A vezérlő jelei aktív magas szintűek.

A hétszegmenses kijelző karakterei DIG0-tól DIG3-ig vannak számozva. A kijelző minden vezérlő jele aktív alacsony szintű.

A pontmátrix kijelző sorainak jelölése felülről lefelé rendre ROW1-ROW7. Az oszlopok COL0-tól COL4-ig terjednek. A kijelző minden vezérlő jele aktív alacsony szintű.

A kijelzőket időmultiplexelt módon kell vezérelni. A hét vezérlőjel közös a két kijelző esetében. Minden egyes karakternek és oszlopnak különböző kiválasztó jele van. A két kijelző külön-külön is használható. Ilyenkor 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 az alsó állásban jelentenek logikai 0 értéket.
A nyomógombok balról jobbra BTN3-BTN0 sorszámúak. Az ötödik a RESET gomb.
A kártya órajel forrásaként felhasználhatjuk a panelen lévő 16MHz-es oszcillátort vagy a fejlesztői kábel CLK vonalát is. Az FPGA-ban található DCM (Digital Clock Manager) segítségével egyéb frekvenciák is előállíthatóak.

A kártya kétféle módon lehet felkonfigurálni:
- a fejlesztői port JTAG interfészén keresztül
- a kártya képes magát is felkonfigurálni a rajta lévő FLASH memóriából.
A konfigurációs mód egy jumperrel választható ki.

A fejlesztői port a következő interfészekkel rendelkezik:

(11) 5 V	(9) V_{ref} I/O	(7) MOSI	(5) CLK	(3) TCK	(1) TDO
(12) V_{ref} JTAG	(10) GND	(8) MISO	(6) RST	(4) TMS	(2) TDI

5. ábra: A Logsys-fejlesztői kábel csatlakozó lábkiosztása (forrás: Logsys Felhasználói útmutató)

- JTAG interfész: TDI, TDO, TCK és TMS vonalak
- Vezérlő interfész:
  - CLK órajel bemenet
  - RST reset bemenet
- Soros kommunikációs interfész:
  - MOSI soros adat bemenet
  - MISO soros adat kimenet
- Tápellátás:
  - 5V tápfeszültség
  - Referenciafeszültség kimenetek a fejlesztői kábel számára:
    V_{ref} I/O, V_{ref} JTAG

Az FPGA kártya 5V-os tápfeszültséget igényel. A tápellátás a fejlesztői kábelen keresztül történik, de a panelen lévő 5V DC csatlakozóra is köthetünk külső áramforrást. A kártyán található továbbá két bővítőcsatlakozó (A és B) amelyhez kiegészítő paneleket köthetünk:

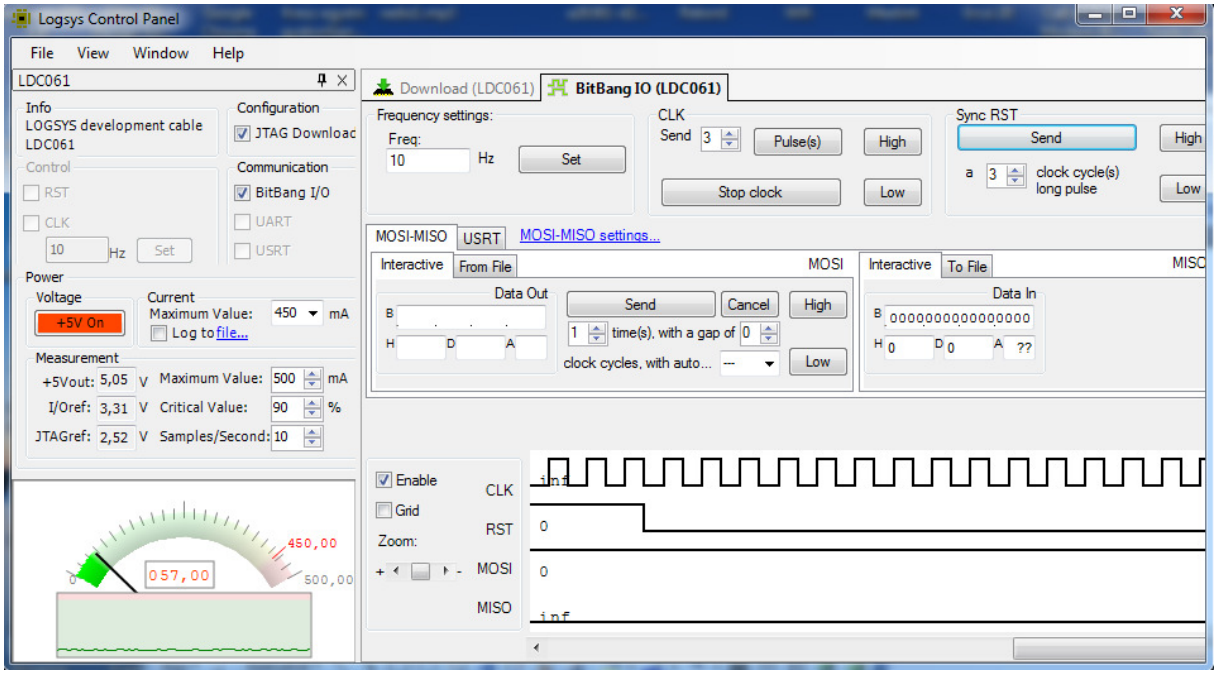
(15) Input	(13) I/O	(11) I/O	(9) I/O	(7) I/O	(5) I/O	(3) +3,3V	(1) GND
(16) Input	(14) I/O	(12) I/O	(10) I/O	(8) I/O	(6) I/O	(4) I/O	(2) +5V

6. ábra: A Logsys-panel bővitő csatlakozója (forrás: Logsys Felhasználói útmutató)

Mindkettő 16 pólusú, azonos lábkiosztású. 2 input és 11 I/O porttal rendelkeznek. Ki van vezetve rájuk a 3,3V-os és az 5V-os tápfeszültség is. Az adatvonalak 3,3V-ról müködnek.

2.1.2 Logsys GUI

A felkonfigurálás a fejlesztői kábelen keresztül történik, amit a Budapesti Műszaki Egyetem által készített grafikus felhasználói felület, a LOGSYS GUI segítségével lehet kezelni.



7. ábra: A Logsys GUI

A program kezelői felülete szabadon áthelyezhető modulokból épül fel. Minden kábel funkciót egy külön panelen érhetünk el. A rendszer több kábel egyidejű használatát is támogatja. Ezek bármikor csatlakoztathatók és eltávolíthatók a program futása közben is, mert a GUI követi a változásokat. Minden kábelhez egy-egy saját panel tartozik ahol elérhetőek az egyes kábelek funkciói. Az egyszerűen használható funkciók a kábel panelról, míg az összetettebbek külön ablakokból érhetők el.

A panel bal felső sarkában az adott kábel típusa és sorozatszáma látható. Utóbbi a kábel egyedi azonosítójaként szolgál.

A képernyő bal oldalán az egyszerűbb kapcsolókat és visszajelző részeket találjuk:

Az *ON/OFF* gombbal be illetve ki kapcsolhatjuk a kártyát.

A *JTAG Download* kapcsolóval konfigurációs fájlokat tölthetünk az eszközünkre. A program támogatja az SVF fájlformátumot, ami ipari szabvány. Emellett ha Xilinx termékkel rendelkezünk, lehetőség van a BIT file feltöltésére is.

Megadhatunk kártyánknak külső órajelet is a CLK kapcsoló segítségével. Az RST kapcsolóval pedig alapállapotba állíthatjuk.

Itt találjuk még a tápfeszültség beállító és fogyasztás indikátor részeket.

A különböző kommunikációs kapcsolók segítségével többfajta mód közül is választhatunk:

- BitBang I/O – Ebben a módban manuálisan adhatjuk meg a különböző vezérlő és kommunikációs órajeleket, amit egy visszajelző grafikonon követni is tudunk.
- UART – Itt aszinkron soros kommunikációt tudunk megvalósítani a MOSI/MISO vonalak segítségével.
- USRT – Szinkron soros átvitelre szolgál.

A program képes fájlból kiolvasni az átküldendő és fájlba menteni a kapott adatokat.

2.2 Hardverleíró nyelvek

Napjainkban a HDL (Hardware Description Language) nyelvek jelentősége igen nagy. Ezek összetett áramkörök leírására készültek. Egy-két évtizeddel ezelőtt a tradicionális tervezői módszereket, amelyek a kapuszintű, modulszintű kapcsolási rajzokon alapultak fokozatosan felváltják a hardverleíró nyelveken alapuló regisztertranszfer szintű (RTL Register Transfer Level) leírások, amelyek alapján a szintézis eszközök a kívánt funkciónak megfelelő áramköröket automatikusan előállítják. A HDL nyelvnek a hardver működésének modellezésén túl olyan feltételeket kell biztosítania, amelyek a tervezői munka minőségét és hatékonyságát növelik.

A HDL nyelv egy speciális alkalmazási területhez kifejlesztett számítógépes programozási nyelv. Speciális szerkezeteket és kifejezéseket alkalmaznak a hardver eszközök időben párhuzamos, konkurens működésének leírására és az időbeli késleltetések és jelalakok modellezésére.

Előnyei:

- Támogatják a strukturált tervezési módszereket
- Rugalmasan particionálhatók
- A nyelvi leírás alapján kapcsolási rajz generálható, ha szükséges
- Magas szintű absztrakció
- Felkínálja alternatívák összehasonlítását
- Módosítások gyorsan végrehajthatóak
- Javítja a hatékonyságot és a minőséget
- Gyors prototípus készítés
- Kihasználja a szintézis eszközöket

A két legelterjedtebb hardverleíró nyelv a Verilog és a VHDL.

A szakdolgozatunk kódjának elkészítéséhez a Verilog nyelvet választottuk, mert a korábbi FPGA-val kapcsolatos kurzusokon ezzel ismerkedtünk meg.

2.2.1 Verilog

A Verilog egy hardverleíró nyelv. Elektronikus áramkörök és rendszerek leírására való szöveges formátum. Elektronikus áramkörök esetén a funkcionális működés Verilog szimulációval való ellenőrzésére, az időzítések ellenőrzésére, tesztelésre és logikai szintézisre használják.
A nyelv története az 1980-as évekre nyúlik vissza, amikor egy Gateway Design Automation nevű cég kifejlesztett egy Verilog-XL nevű logikai szimulátort, és vele együtt egy hardverleíró nyelvet.
1989-ben a Cadence Design Systems megvásárolta a Gateway-t.
1990-ben a Cadence nyilvánossá tette a nyelvet azzal a szándékkal, hogy az standard, céghez nem kötődő nyelv legyen.
1995-ben a Verilog HDL lett az IEEE 1364 számú standardje.
2001-ben nyilvánosságra hozzák a „Verilog 2001" –et.

A nyelv felépítése hasonlít a C nyelvre, viszont szemantikailag a párhuzamosság jellemzi. Bizonyos kódrészletek egyidejű végrehajtása az áramkörben lévő párhuzamos jelterjedést modellezi. Hierarchikus, funkcionális egységeken alapuló tervezői megközelítést alkalmaz. A teljes terv több, kisebb modulból áll össze, melyek részfunkciókat valósítanak meg. A modul egy hardver blokkot ábrázol, jól definiált kimenetekkel és bemenetekkel, valamint meghatározott feladattal. A modul két részből áll. A port deklarációk a modul külső interfészét reprezentálják. A modul többi része egy belső leírás: a viselkedése, a szerkezete, vagy e kettő keveréke.

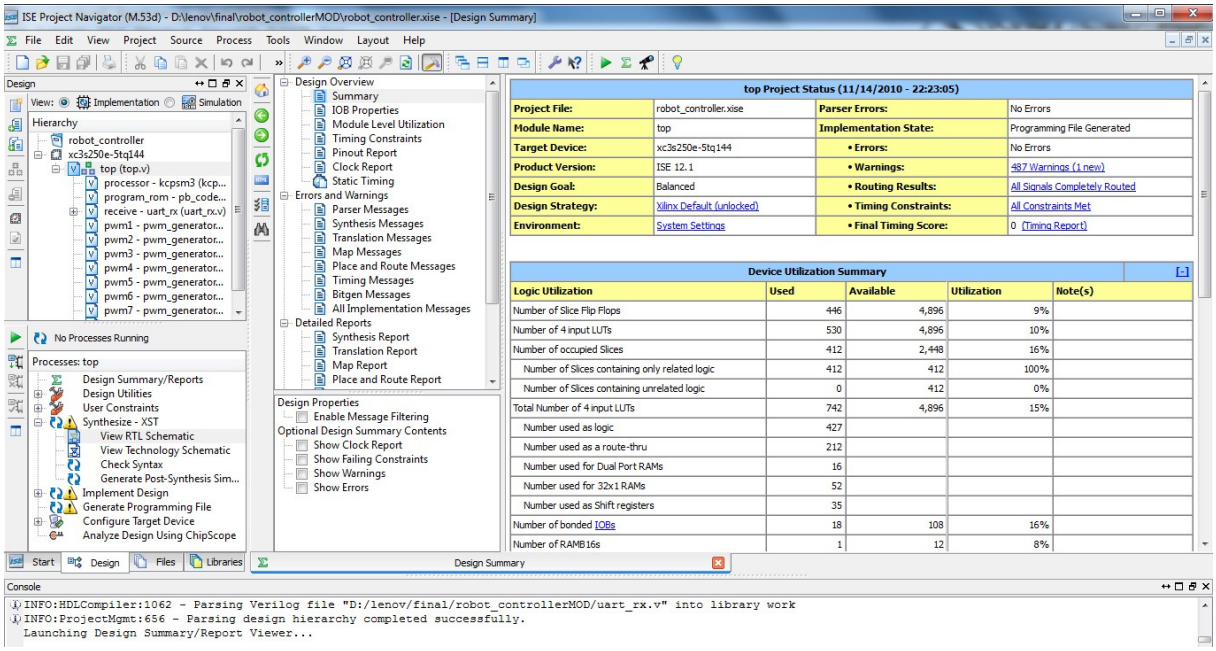
2.2.2 Xilinx ISE WebPACK

A Xilinx cég, a programozható logikai eszközök (PLD és FPGA) egyik jelentős gyártója, kidolgozott egy számítógépes tervező rendszert ezeknek az eszközöknek a használatához. Ennek a rendszernek a neve: Xilinx ISE Logic Design Tools. A cég az ISE rendszer egyszerűbb, de funkcionálisan komplett változatát is összeállította, ami a WebPACK nevet kapta. A WebPACK a Xilinx cég honlapjáról (www.xilinx.com) ingyenesen letölthető regisztráció után.

Szakdolgozatunk kódja megírásához, fordításához, teszteléséhez ezt a fejlesztőkörnyezetet használtuk, mert számos hasznos funkcióval rendelkezik a program illetve a kezelőfelülete is átlátható.

Előnyök:

- A felhasználó életét rengeteg beépített példaprogram (template) könnyíti.
- Lehetőség van a kód verifikálására, szimulációjára.
- Felkonfigurálhatjuk az FPGA-t ha az ISE által támogatott eszközzel rendelkezünk.
- RTL sematikus ábrát hozhatunk létre, ami átláthatóbb, mint a kód; ezáltal hibakeresésre tökéletes megoldás
- Egyértelmű hibaüzenetek → Könnyű debuggolás!



15. ábra: A Xilinx ISE kezdőképernyője

2.3 Szoftver – processzor, PicoBlaze

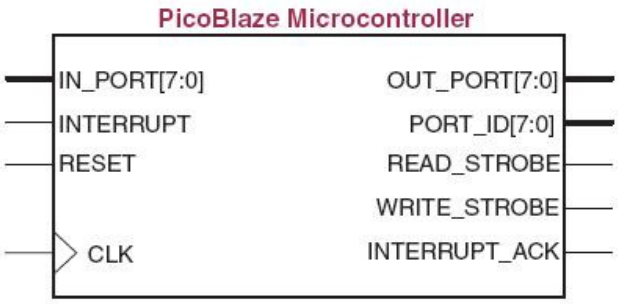
A szoftver – processzor egy, az FPGA-ba fordítható mikrokontroller. Ugyanúgy rendelkezik például aritmetikai-logikai egységgel (ALU), program-számlálóval, megszakításvezérlővel, bemeneti/kimeneti vonalakkal, mint egy valódi egység, csak az FPGA-ban, a hardverleíró nyelv segítségével van felépítve. Ilyen szoftver-processzor például a PicoBlaze, mely a Xilinx licence alatt szabadon felhasználható.

A PicoBlaze tulajdonképpen egy 8-bites mikrokontroller, amelyet a Xilinx cég mérnöke, Ken Chapman kifejezetten a Spartan-, illetve Virtex-szériájú FPGA-khoz készített. Esetünkben a PicoBlaze KCPSM3 (Ken Chapman Programmable State Machine 3) nevű szoftver-processzort használtuk, mivel ez van a Spartan3 termékcsaládhoz optimalizálva. A PicoBlaze processzor mag egyszerűen realizálható Xilinx FPGA áramkörrel, egyszerűen le kell tölteni a megfelelő konfigurációs állományokat. A letöltött PicoBlaze mag mellett még jelentős hely marad saját áramköri egységek létrehozására, mivel egy mag 96 slice helyet foglal el, ez nagyjából a felhasznált FPGA kb. 5%-a. Természetesen akár több ilyen processzor is tölthető az FPGA-ra, a feladat bonyolultságától függően. Ennek segítségével lehetőség nyílik közvetett módon egyszerű assembly program írására FPGA eszközre.

A PicoBlaze mikrokontroller főbb tulajdonságai:

- 16 bit széles adat-regiszter
- 1024 utasítás méretű programtár
- 8 bit széles aritmetikai – logikai egység, CARRY és ZERO jelzőflagekkel
- 64 byte széles Scratchpad RAM („vázlatfüzet” a gyakran használt adatok számára)
- 256 input és 256 output port

A processzor felépítése a következő:



8. ábra: A PicoBlaze modul sematikus ábrája (forrás: PicoBlaze User Guide)

A vonalak jelentése:

- IN_PORT[7:0]: **Bemeneti port:** Ide érkeznek a bemeneti adatok. Ezek az adatok az órajel felfutó élével egyidejűleg kerülnek kiolvasásra.
- INTERRUPT: **Megszakítás:** Megszakítást generál a processzor, ha az utolsó két órajelciklus alatt pozitív értéket kap.
- RESET: **Reset bemenet:** A processzoron belül generálódik egy Reset-Event, ha ezen a vonalon 1 órajelciklusban magas érték található.
- CLK: **Órajel bemenet:** Bemeneti órajel, mely lehet az FPGA-panel saját órajele, vagy a Xilinx ISE által generált.
- OUT_PORT[7:0]: **Kimeneti adatport.** A kimenő adatok itt olvashatók ki két órajelcikluson keresztül, ha OUTPUT utasítás érkezett, és a WRITE_STROBE vonal magas értéken van. A kiolvasás a órajel felfutó élével egy időben történik meg.
- PORT_ID[7:0]: **Portcím:** Az I/O művelet az itt megadott porton történik meg INPUT vagy OUTPUT utasítás esetén.
- READ_STROBE: **Olvasás-jelzés:** Ha magas értéken van, akkor azt jelöli, hogy az adat az IN_PORT vonalon beolvasásra került az INPUT parancsban megadott regiszterbe.
- WRITE_STROBE: **Írás-jelzés:** Ha magas értéken van, akkor azt jelöli, hogy az adat az OUT_PORT-ra került, az OUTPUT parancs hatására.
- INTERRUPT_ACK: **Megszakítás nyugtázása:** ha magas értéken van, akkor jelzi, hogy az INTERRUPT vonalon megszakítás történt.

Szakdolgozatunkban a PicoBlaze feladata a számítógép soros portjáról érkező utasítások fogadása, értelmezése, illetve továbbítása a robotkar szervó motorja számára. A soros-porti kommunikáció egyik leggyakoribb és általunk is használt eszköze az **UART**.

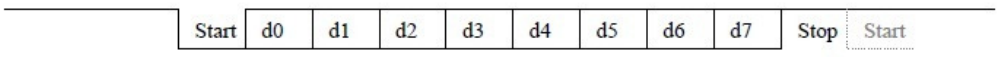
2.4 UART

2.4.1 Az UART általános definíciója

Az UART (Universal Asynchronous Receiver/Transmitter), magyarul univerzális aszinkron adóvevő a legelterjedtebb módja a soros interfészek közötti adatátvitelnek. Az UART egy olyan célhardver, mely a soros portra küldött adatok bájtjait aszinkron start-stop bitfolyammá alakítja át, majd ezeket elektromos impulzusok segítségével továbbítja. Az adatátvitelt azért nevezzük aszinkronnak, mivel az átvitel egy startbit segítségével bármikor kezdődhet. Előnye, hogy nem kell szinkronizált órajelet átvinni, így az átvitel módja egyszerű, egyetlen vezetéken megvalósítható. Létezik szinkron adóvevő is, ekkor egy további vezeték szükséges az órajel átvitelére, így az átvitel bonyolultsága nő.

Az UART két fő része az adó (Transmitter), és a vevő (Receiver). Az adó tulajdonképpen egy speciális shift-regiszter, amely párhuzamos módon betölti az adatokat, majd bitenként kiküldi azokat egy megadott sebességgel. A vevő a másik oldalon pedig bitenként beolvassa az adatokat, majd a fogadott bitekből összeállítja az adatot.

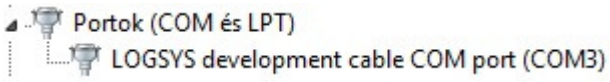
A soros vonal alaphelyzetben mindig magas (logikai 1) értéken áll. Az átvitel mindig egy startbittel kezdődik, ami alacsony (logikai 0) értékű. Ezt követik az adatbitek, melyeket beállítástól függően paritásbit követhet, majd az átvitel egy stopbittel fejeződik be, ami logikai 1 értékű. Az adatbitek számát szintén előre be kell állítani, ez lehet 6, 7, vagy 8. A paritásbitet hibaellenőrzésre használjuk. Páratlan számú 1-es átvitele esetén a paritás logikai 0, páros számú 1-es átvitele esetén a paritás logikai 1 értéket vesz fel. A stopbitek száma a beállítástól függően lehet 1, 1.5, vagy 2. Az átvitel sebességét baud-ban adjuk meg, mely a másodpercenként átvitt bitek számát jelöli. A 9. ábrán egy olyan átvitelt láthatunk, ahol 8 adatbitet továbbítunk paritásbit nélkül, és 1 stopbittel.



9. ábra: Az UART adatátvitel (forrás: PicoBlaze User Guide)

Az átvitel során nem küldünk órajel információkat. Ezért az átvitel megkezdése előtt az adó és a vevő beállításait egyeztetni kell, vagyis az átvitel sebességét baudban, az adatbitek számát, a stopbitek számát, valamint a paritásbitre vonatkozó információkat.

A mai számítógépek és laptopok többsége már nem rendelkezik soros porttal. Ezért mi a kommunikációhoz a Logsys-panel fejlesztői kábelét használjuk.



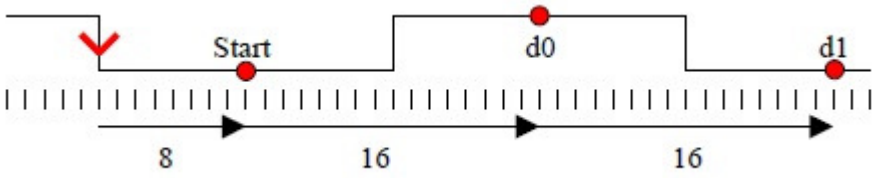
10. ábra: A Logsys fejlesztői kábel COM portja a Windows Eszközkezelőben

A fejlesztői kábel segítségével megvalósítható a soros porti kommunikáció, mivel az emulál egy soros portot fix adatátviteli beállításokkal. Ilyenkor a kábel MOSI érintkezője az adó, a MISO érintkező pedig a vevő. Kommunikáció során használandó beállítások: 9600 baud sebesség 8 adatbittel, paritásbit nélkül, 1 stopbittel. A fejlesztői kábel USB interfészen kommunikál a számítógéppel, mely minden ma használatos PC-n megtalálható.

2.4.2 Az UART megvalósítása PicoBlaze segítségével

A Xilinx cég honlapjáról regisztráció után szabadon letölthető PicoBlaze – csomag tartalmazza az UART kommunikációhoz szükséges modulokat, mellékelik hozzá a teljes dokumentációt, valamint biztosítanak egy mintaprogramot is, mely segítségével az UART interfész használata könnyen elsajátítható. Az UART vevő a következő karakterisztikára van optimalizálva: 1 startbit, 8 adatbit, nincs paritás, 1 stopbit. Ez megfelel a Logsys Fejlesztői kábel tulajdonságainak is, így módosítás nélkül felhasználható. Az adatátvitel sebességét viszont a legfelső, top – modulban be kell állítani. Mivel a robotkar analóg szervó motorokkal van ellátva, így nem képes állapotáról információt küldeni. Ezért a PicoBlaze csak a számítógéptől érkező utasítások feldolgozására van felkészítve, ő maga adatot nem küld, tehát az UART-adó nem került implementálásra.

A vevő a beérkező adatot a megadott baud-sebességnél gyorsabban, úgynevezett túlmintavételezési-módszerrel olvassa ki. A leggyakrabban használt sebesség az eredeti 16-szorosa. Ez azt jelenti, hogy minden egyes bitet 16-szor mintavételezünk.

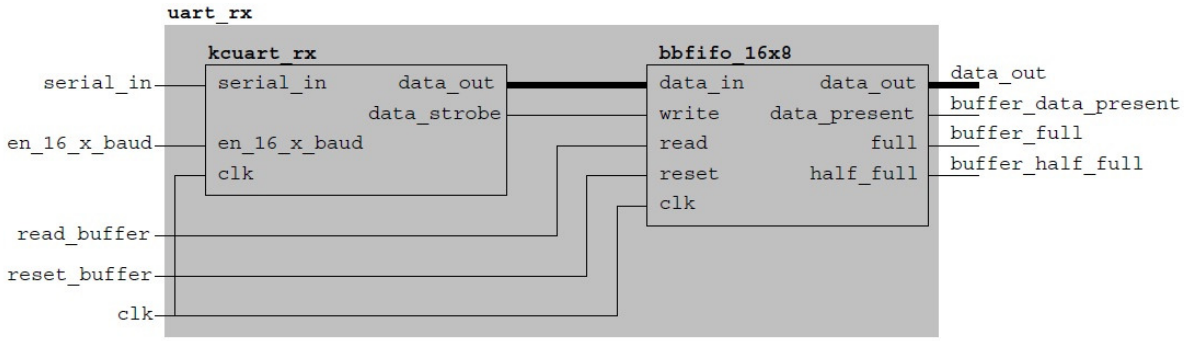


11. ábra: Az UART mintavételezés menete (forrás: PicoBlaze UART Manual)

A módszer menete:

1. Ha a jel logikai 0 értéket vesz fel, elindul a mintavételezés számlálója.
2. Ha a számláló eléri a 7-et, akkor épp a jel közepén tartunk. A számlálót nullázzuk.
3. Ha a számláló eléri a 15-öt, elértünk az első adatbit közepére. A mintavételezett értéket eltároljuk egy regiszterbe, a számlálót újraindítjuk.
4. Ismételjük meg a 3. lépést N-1 alkalommal.
5. Ismételjük meg a 3. lépést M alkalommal, míg elérjük a stopbitet.

Az UART vevő modulja összesen 3 fájlból áll. A vevő top-modulja az ’uart_rx.v’ fájl, melyhez tartozik még a vételt megvalósító ’kcuart_rx.v’ fájl, valamint valamint a hozzá tartozó FIFO buffert megvalósító ’bbfifo_16x8.v’ fájl.



12. ábra: Az UART modul sematikus felépítése (forrás: PicoBlaze UART Manual)

Az UART vevő modul jelei:

- serial_in: Ezen a vonalon érkezik be a soros adat, először a startbit, utána a 8 adatbit, végül a stopbit. Az en_16_x_baud segítségével történik a mintavételezés. Egy érvényes adatátvitel a stopbit vételével zárul, ekkor az adat a FIFO bufferbe kerül (ha nincs tele).
- en_16_x_baud: A soros átvitel mintavételezés időzítési referenciája.
- read_buffer: Logikai 1 értékkel jelzi, ha a data_out vonalon az adat ki lett olvasva, ekkor a FIFO a következő elérhető adatot küldi ki.
- reset_buffer: Magas érték esetén a 16 bájtos puffer nullázódik, a benne levő adatok elvesznek.
- clk: Globális órajel bemenete.
- data_out: A byte-adatok innen kerülnek kiolvasásra. Az itt lévő adat érvényes, ha a ’buffer_data_present’ vonal magas értéken van.

- buffer_data_present: Ha a puffer tartalmaz legalább 1 byte fogadott adatot, akkor ez a vonal magas értékű és az adat elérhető a ’data_out’ porton.
- buffer_full: Ha a 16 bájtos FIFO megtelt, akkor logikai 1 értéket vesz fel.
- buffer_half_full: Ha a 16 bájtos FIFO 8 vagy több bájt adatot tartalmaz, akkor logikai 1 értéket vesz fel.

Mint látható, az UART modul rendelkezik a fogadáshoz szükséges FIFO buffer tárolóval, és jelzővonalakkal, így működéséhez további áramköri elem nem szükséges. INPUT parancs hatására a PicoBlaze in_port vonalára kerül a FIFO következő eleme, majd az eltávolításra kerül.

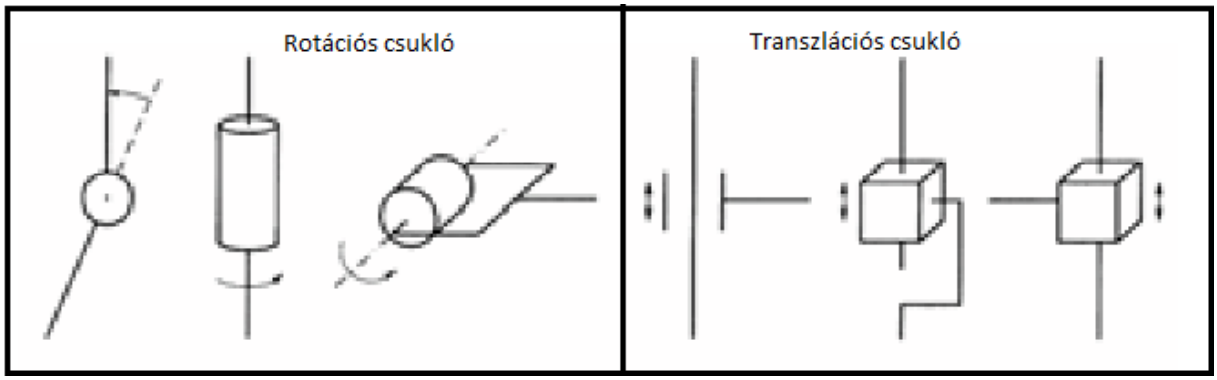
2.5 A robotok

A robot (a szláv robota szóból ered, jelentése: munkaság, szolgaság) elektromechanikai szerkezet, amely előzetes programozás alapján képes különböző feladatok végrehajtására. Lehet közvetlen emberi irányítás alatt, de önállóan is végezheti munkáját egy számítógép felügyeletére bízva. A robotokkal rendszerint olyan munkákat végeztetnek, amelyek túl veszélyesek, nehezek vagy nagy pontossággal végrehajtandó feladatok.

Manapság a robotok szinte az élet minden területén jelen vannak hol kisebb, hol nagyobb komplexitásban. (pl.: nehézipar, autógyártás, orvostudomány, háztartási „robotok", űreszközök, játékok, katonai robotok, humanoid robotok).

Az *ipari robot* mechanikai struktúra vagy manipulátor, amely merev testek (szegmensek) sorozatából áll, melyeket összeillesztések (csuklók, ízületek) kapcsolnak össze. A manipulátor szokásos részei: kar (mozgatás), kézcsukló (kézi funkciók), végberendezés (kívánt feladat elvégzése). Az *aktuátorok* a manipulátor *ízületeiben, csuklóin* elhelyezkedő mozgatóegységek. Ezek elektronikus, hidraulikus, vagy pneumatikus elven működő eszközök. A robotokon *szenzorok* is lehetnek: Ezek a manipulátor állapotának és a környezet jellemzőinek mérésére szolgálnak. Az *irányítórendszer* lehet számítógép, mikrokontroller vagy akár egy FPGA is (mi is ezzel vezéreljük). A manipulátorok szerkezete lehet nyílt vagy zárt kinematikai lánc. Ízületeik lehetnek transzlációs vagy rotációs csuklók.

A *rotációs csukló* forgómozgásra, míg a *transzlációs csukló* egy tengely menti mozgásra képes a szegmensek között.



A *mozgás szabadságfoka* a működtetett ízületek számával egyenlő.

A *szabadságfok* egy adott feladat végrehajtásához szükséges független paraméterek száma. Például egy háromdimenziós objektum tetszőleges pozícionálásához és orientálásához 6

szabadságfok szükséges. *Kinematikailag redundáns* a manipulátor, ha a mozgás szabadságfoka nagyobb, mint a manipulátor szabadságfoka.

A *munkatér* a környezet azon része, amit a manipulátor el tud érni. Alakja és térfogata függ a manipulátor szerkezetétől és a csuklók mechanikai korlátozásaitól.

A kar mozgásának szabadságfoka szerinti csoportosítás:

- Descartes
- henger
- gömb
- SCARA
- antropomorf

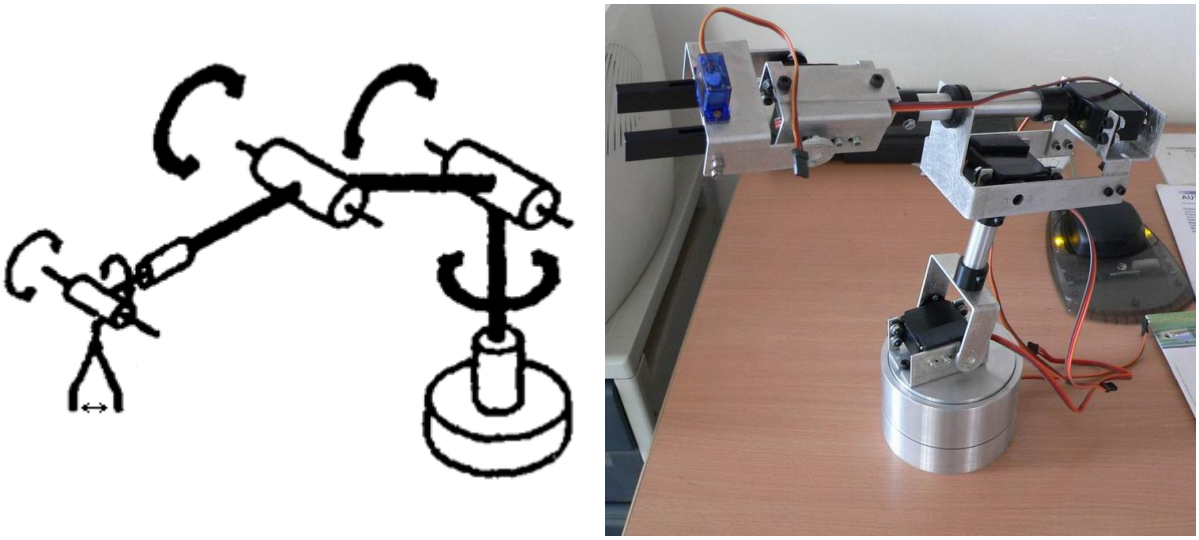
Felosztásukra sokféle szempont létezik:

- ipari manipulátor (kevés szabadság)
    - programozható manipulátor
        - egyszerű mozgás, kötött program
    - teleoperációs manipulátor
        - tetszőleges mozgás, nincs program
- ipari robot (okos vezérlés, szabadon programozható)
    - pont-szakasz vezérlésű robot
        - PS (point-straight line mozgás) a pálya paraméterei nem adhatók meg, csak a célpont programozható
    - pályavezérlésű robot
        - szervorobot, CP (contiunuous path) robot, a pálya paraméterei megadhatók, programozhatók (pályatípus, sebesség, gyorsulás, stb.)
    - intelligens, szenzorvezérelt robot
        - SC (sensor controlled) a programozott pályától eltérő pályán is mozoghat, programozható paraméterek, pl. erő függvényében.

A robotok osztályozhatók még mozgásuk, munkaterük, vezérlésük, feladatuk, energiaforrásuk, méretük szerint is.

2.5.1 Robotkar

A szakdolgozathoz használt robotkar a National Instruments vállalatnál készült az ipari robotok modellezése céljával. Felépítése az ipari robotokéhoz hasonló:



13. ábra: A robotkar vázlatos és valódi képe

A kar szerkezete egy nyílt kinematikai lánc. A mozgásért az ízületekben (5db rotációs csukló) elhelyezkedő 5db analóg szervó motor felel (aktuátorok). Az egész robot körbe forgását a talapzatban lévő motor biztosítja, amely síkja merőleges a karban lévő két másik motor síkjára és a végberendezést forgató motor síkjára is. A végberendezés két darab merev ujjból álló megfogó szerkezet: két szorítópofa, amit a két pofa között lévő hatodik szervó húz össze illetve távolít szét.

Az alsó két motornak van szüksége a legnagyobb áramerősségre, mert ezek a legerősebbek. Ezeknek a motoroknak kell a legnagyobb tömegeket megmozgatniuk. Fölfelé haladva a karon, a következő három már gyengébb teherbírású. A legutolsó és egyben leggyengébb motor a végberendezésben található. A szorítópofáknak nem kell hatalmas erőt kifejtenie, ezért ide ez a motor is elegendő volt.

Mozgásának szabadságfoka 5 mert 5db szervó található a karban. Szabadsági foka azonban 3, mert itt már csak a független paramétereket kell figyelembe venni. Több rotációs csuklója is ugyanabban a síkban mozog tehát ez egy redundáns manipulátor. A kar mozgásának szabadságfoka szerint ez egy antropomorf manipulátor.

A kar nem rendelkezik szenzorokkal, irányítórendszere a mi esetünkben egy FPGA.

A robotkart egy – a LOGSYS panelhez gyártott – kiegészítő-panel segítségével csatlakoztattuk az FPGA I/O portjához. Ez egy speciális panel, amely 10 egymástól független vezérlő csatornát biztosít számunkra és lehetőség van külső áramforrás biztosítására, mert a szervó motorok együtt igen nagy áramfelvételre képesek. Erre egy – speciálisan átalakított – számítógép tápegységet használtunk.

2.6 A szervómotorok

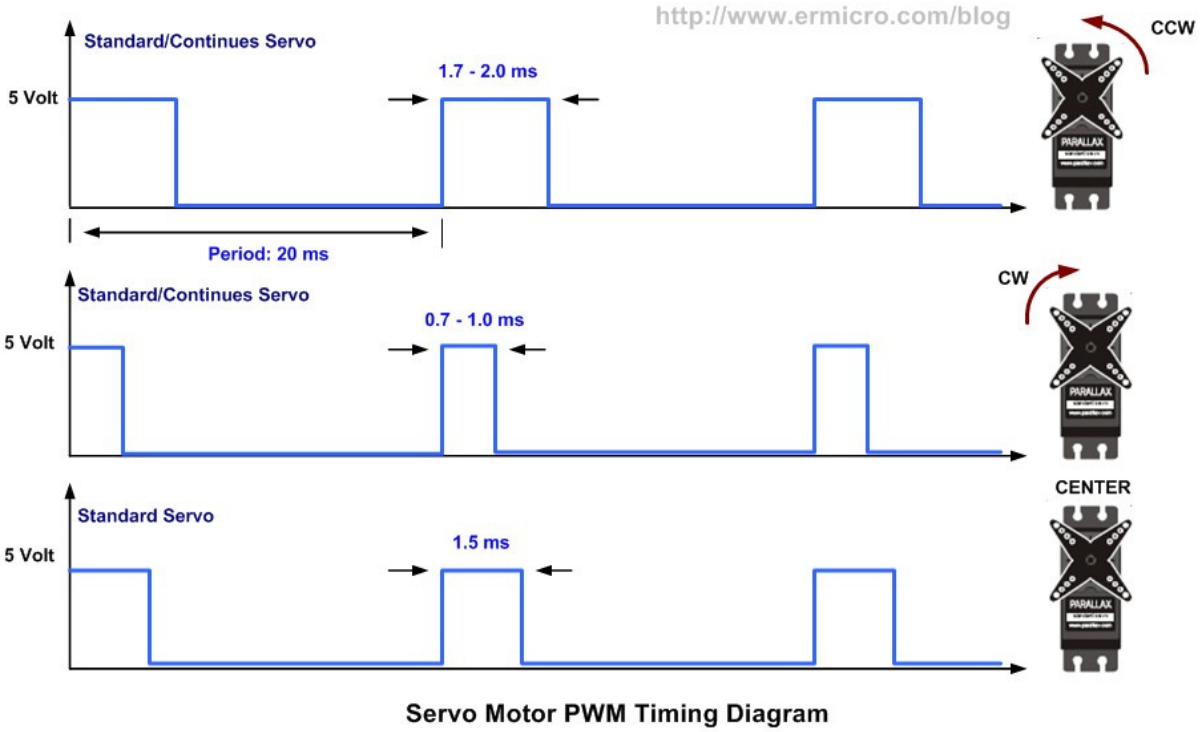
A robotkar mozgatása szervó motorokkal valósul meg. Ezek a speciális eszközök impulzusszélesség (PWM - Pulse-width modulation) vezérelt motorok. Ez a gyakorlatban azt jelenti, hogy a motoroknak impulzusszélességgel lehet megadni, hogy milyen pozíciót vegyenek fel. A motor vezérlőjének első része egy impulzusszélesség – feszültség konverter. Ez megméri a kapott impulzus szélességét, majd annak alapján egy adott feszültséget generál. A vezérlőbe be van égetve a minimum impulzus szélesség (minIW) és a maximum impulzus szélesség (maxIW). A generált feszültség maximális értéke közel megegyezik a bejövő tápfeszültséggel. Ezt a maxIW elérésekor adja ki magából. A motor nem ezzel a feszültséggel lesz meghajtva, ez csak egy referencia feszültség. A szervó tengelyén egy potmétert helyeztek el. Ahogy fordul a tengely, úgy változik ellenállása. Egyik vége V+ a másik pedig GND-re van kötve. A középső lábán pedig egy referencia feszültséget kapunk. Az egyik referencia feszültéség a motor aktuális pozícióját jelzi, a másik pedig a jövőbeni pozícióját. A cél, hogy a két referencia feszültség egyenlő nagyságú legyen, tehát a motor elérje a kívánt pozíciót. Ehhez ez a két feszültség rá van engedve egy harmadik egységre, ami összehasonlítja a két feszültséget. Ez a harmadik egység képes a feszültségkülönbségek alapján meghatározni, hogy a motort jobbra vagy balra kell forgatni ahhoz, hogy a két referenciafeszültség egyforma legyen.

Minden szervón 3 vezeték található. Ebből kettő a tápellátásért felel. Ez általában 4,8-7,2 V között lehet a hasonló méretű motoroknál. A feszültség növelésével növekszik a szervó nyomatéka, sebessége viszont csökken az élettartama. A harmadik vezeték pedig a vezérlésért felelős. Erre tipikusan 5V feszültség kerül.

2.6.1 A PWM vezérlés

PWM - Pulse-width modulation

Az impulzus szélességek szervónként eltérőek. A legtöbb szervó 1,5 ms széles impulzus hatására áll középállásba. Vannak olyan szervók, amik -60° és +60° között mozognak. Ezek általában 1,2-1,8 ms közötti impulzust kapnak. A nagy mozgásterű szervók -90° és +90° közötti tartományban képesek mozogni. Ezek 0,5 ms és 2,5 ms közötti impulzusszélességgel vezérelhetőek. Az impulzust nem elég egyszer kiadni, hiszen analóg elven működnek. A motor csak egy nagyon rövid ideig kap tápfeszültséget. Ez a rövid idő alatt nem biztos, hogy a szervó képes elérni a kívánt pozíciót. Ha az impulzus konverter nem kap jelet, akkor a motort sem gerjeszti, tehát a rendszer "elernyed". A digitális szervóknak általában elég egy impulzus, hiszen a digitális vezérlés addig hajtja a motort, amíg el nem éri a kívánt pozíciót. Az impulzushiányos időben lévő "elernyedés" ezekre is ugyanúgy jellemző. Ezért 50Hz vagy 60Hz-cel kell küldenünk a jelet a szervónak.



14. ábra: A PWM jelgenerálás idődiagramja

Az NI robotkarában 3 különböző típusú szervó van:

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

Az MG-995-ös fémfogaskerekes szervó a legerősebb a három közül ezért ő helyezkedik el legalul a robotkarban. Itt van szükség a legnagyobb nyomatékra (1.08 Nm).

Az SG-5010-es szervók fölfelé haladva középen helyezkednek el a karon. Ezek már gyengébbek előző társuknál (0.51 Nm).

A szorítópofákat pedig egy SG-91R mozgatja. Neki van a legkisebb nyomatéka (0.18 Nm) mégis tökéletesen elég erre a feladatra.

Mi a robotkar szervóit 5V-os feszültségen és 50Hz-es frekvencián vezéreltük, ami 20 ms-os periódusidőt jelent. 0,5 ms és 2,5 ms között változtattuk a kitöltési tényezőt, ami a szervók két végállásának felelt meg. A kar indulási pozíciójában a motorok 1,5 ms-os kitöltési tényezőt kapnak tehát középállásba helyeződnek.

2.7 Fejlesztői környezetek

2.7.1 Java

A robotkart irányító grafikus felület Java nyelven, a Netbeans IDE szoftver segítségével lett írva.

A Java egy általános célú, objektumorientált programozási nyelv, amelyet a Sun Microsystems fejleszt a ’90-es évek elejétől kezdve napjainkig. A Java alkalmazásokat bytecode formátumra alakítják, amely futtatása a Java virtuális géppel (JVM – Java Virtual Machine) történik. A Java nyelv szintaxisát főleg a C és a C++ nyelvektől örökölte, viszont a Java sokkal egyszerűbb objektummodellel rendelkezik, mint a C++.

A Java nyelvet kávézás közben találták ki, innen ered a kávéscsésze ikon. Négy fontos szempontot tartottak szem előtt, amikor a Javát kifejlesztették:

- objektum-orientáltság
- függetlenség az operációs rendszertől, amelyen fut (többé-kevésbé)
- olyan kódokat és könyvtárakat tartalmazzon, amelyek elősegítik a hálózati programozást
- távoli gépeken is képes legyen biztonságosan futni

A nyelv első tulajdonsága, az objektum-orientáltság („OO”), a programozási stílusra és a nyelv struktúrájára utal. Az OO fontos szempontja, hogy a szoftvert „dolgok” (objektumok) alapján csoportosítja, nem az elvégzett feladatok a fő szempont. Ennek alapja, hogy az előbbi sokkal kevesebbet változik, mint az utóbbi, így az objektumok (az adatokat tartalmazó entitások) jobb alapot biztosítanak egy szoftverrendszer megtervezéséhez.

A második legfontosabb tulajdonság pedig a platformfüggetlenség/hordozhatóság. Ez azt jelenti, hogy az ezen a nyelven íródott programok hasonlóan fognak futni különböző hardvereken. A Java fordítóprogram bájtkódra fordítja le a forráskódot, ami aztán futtatva lesz a virtuális gépen (JVM). Ez jelentős költségcsökkenést eredményez, mert a kódot csak egyszer kell megírni.

2.7.2 NetBeans

A NetBeans egy integrált fejlesztői környezet, ami a Java nyelven alapul. A program grafikus fejlesztőfelületet kínál a különböző alkalmazások, Appletek vagy akár JavaBeanek elkészítéséhez, amelynek segítségével könnyebben, gyorsabban tudjuk fejleszteni saját programjainkat. A szoftver ingyenesen letölthető a www.netbeans.org honlapról.

A NetBeans cseh eredetű, a szoftver Xelfi néven látta meg a napvilágot 1997-ben a prágai Károly Egyetem matematika és fizika tanszékén, majd később egy cég alakult a Xelfi útjának egyengetésére, amit 1999-ben vett meg a Sun Microsystems, amely még ugyanabban az évben nyilvánossá tette a szoftver forrását. A NetBeans egy Java-fejlesztői környezetnek indult, azonban a 6.0 verzió óta már a Ruby on Railst, illetve a JavaScriptet is támogatja a C és a C++, illetve a Java Enterprise Edition mellett, azóta pedig több más nyelvvel (pl. PHP) bővült a kínálata.

Előnyök:

- Szabadon használható
- Támogatott technológiák: Ajax , C/C++, Databases, Debugger, Desktop. Editor, Groovy, GUI Builder, Hudson, Java EE, JavaFX, Java ME, Java SE, JavaScrip, Kenai, Maven, Mobile, PHP, Profiler, Python, Refactor, REST, Rich Client Platform, Ruby, SOAP, Web
- Pluginek/kiegészítők támogatása
- Alap támogatások: kód kiegészítés (code completion) / szinezés (highlight) / hajtás (folding) / formázás / sablonok (templates)
- Projekt támogatások: verziókövetés (CVS, Mercurial, SVN) / Kenai Projektek natív támogatása
- MySQL adatbázisok kezelése, grafikus felülettel

2.7.3 PicoBlaze C Compiler

Az FPGA-ban lévő PicoBlaze szoftver processzor utasításait C nyelven adtuk meg. A C forráskódból a PicoBlaze C Compiler nevű fordítóprogrammal készítettük el az assembly utasításokat tartalmazó PSM fájlt, majd ebből a PicoBlaze KCPSM3 nevű fordítóprogramja segítségével létrehoztuk a PicoBlaze processzor utasításait tartalmazó memória kódot.

A PicoBlaze C Compilert Francesco Poderico, FPGA fejlesztő hozta létre azzal a céllal, hogy a PicoBlaze programozható legyen C nyelven is, megkönnyítve ezzel a PicoBlaze-t használó programozók életét.

A PicoBlaze processzor C-ben való programozásának több előnye is van:

- A C könnyű, szinte mindenhol ismerik a világon
- Sokkal átláthatóbb, mint az assembly
- A forráskód sokkal rövidebb C-ben, hamarabb meg lehet írni
- A hibakeresés sokkal könnyebb

Hátrányok:

- A C-ben megírt, aztán assemblyre átfordított kód sokkal hosszabb lesz mintha eleve assemblyben írtuk volna
- Emiatt beleütközhetünk a PicoBlaze utasítás korlátjába, ami maximum 1024 utasítást enged meg
- Oda kell figyelni a változók deklarálásánál a kevés memória miatt → célszerű globális változók használata lokálisok helyett
- A fordító hatékony, de sajnos kód optimalizációt nem végez: a kódot nem rövidíti, a futási időt nem csökkenti

A fordító a C szintaktikáját engedi meg kisebb megszorításokkal. A program írása közben akármikor átválthatunk assemblyre a *#asm*, és vissza a *#endasm* direktívák használatával.
A *#include* segítségével saját header fájlokat hívhatunk meg. Ezek segítségével saját függvényeinket is megírhatjuk, felhasználhatjuk őket a kódban. Nevesített konstansok definiálására is van lehetőség a *#define* paranccsal. A legfontosabb változók az *int* és a *char*.

Lehetőség van mindkettőből előjel nélküli, (*unsigned*) típusok használatára. Itt is, ahogyan az ANSI C-ben, használhatunk tömböket és mutatókat is viszont a mutatóra mutatók (pointer to pointer) használata nem megengedett.
Az elágaztató utasítások és a ciklusok a megszokottak: *IF, WHILE, DO-WHILE, SWITCH*
A fordító segítségével lehetőség van a megszakítások kezelésére. Saját függvényt adhatunk meg ami lefut egy megszakítás beérkezésekor.

Mi a program alpha 1.7.x verzióját használtuk. A 2005-ös Felhasználói kézikönyv szerint Francesco Poderico folyamatosan fejleszti a fordítót és a következő verziók már kód optimalizációt is fognak végezni.

Összevetve az előnyöket és a hátrányokat szerintünk ez egy nagyon hasznos program, rendkívül nagy segítséget nyújt azoknak a programozóknak. akiknek nincsenek assembly ismereteik viszont a C programozási nyelvben otthon érzik magukat.

3. A megvalósult projekt

Szakdolgozati témánk egy olyan nagyobb feladat része, mely során a végleges cél egy ipari robotkar vezérlésének megvalósítása FPGA segítségével. Azért is választottunk FPGA-t a vezérléshez, mivel az eszköz előnye, a párhuzamosság remekül kihasználható. Ez a robotikában igen fontos, a részegységeknek folyamatosan összhangban kell lennie egymással, az információnak pedig a lehető leggyorsabban kell kiértékelődnie, hogy a robotkar a szükséges mozgást időben valósítsa meg. Ez az iparban különösen fontos, és mivel ezt ki is használják, úgy gondoltuk, hogy egy olyan rendszer működését modellezzük le, ami valóban használatos és bevált. Szakdolgozatunkban egy olyan robotkarvezérlés elkészítését tűztük ki célul, mely segítségével a felhasználó a robotkart valós időben, egy grafikus felületű program segítségével irányíthatja. Egyeztetve a témavezetővel, és a külső konzulenssel kiderült, hogy a teljes feladathoz nem elegendő a szakdolgozat elkészítésére szánt idő, ezért meghatároztuk az általunk elvégzendő feladatot.

A mi munkánk előtt már többen is foglalkoztak a robotkar vezérlésének elkészítésével. Vitéz László egy kiegészítő panel segítségével számítógépen előre letárolt mozgássorozatok végrehajtását valósította meg. Tóthfalusi Tamás egy a Logsys-panelre illesztett billentyűzet segítségével valósította meg a robotkar irányítását. Mindketten nagy segítséget nyújtottak számunkra a kezdeti időszakban. Az elkészült vezérlésünk képes a robotkar 6 szervómotorját akár egyszerre is működtetni, lehetőséget adva arra, hogy a robotkar komplex mozgást is végrehajtson. A megvalósításhoz írtunk JAVA nyelven egy egyszerű vezérlőprogramot, ahol egér segítségével lehet a robotkart mozgatni. Mivel az egér 2 dimenziós beviteli eszköz, ezért egyszerre 2 kitöltési tényezőt állíthatunk be a szervó motoroknak, melyeket a Logsys-panel DIP-kapcsolóján kell kiválasztani.

A projektünk két fő része a számítógépen futó ’robot_interface’, valamint az FPGA kártyára feltöltött ’robot_controller’. A ’robot_interface’ egy olyan alkalmazás, melyen egy célkereszt segítségével a kijelölt területen kattintva megadhatjuk a kívánt elmozdulás mértékét. Ekkor a program eltárolja a koordinátákat, majd kiküldi azokat a Logsys Fejlesztői kábel soros portján. A program lehetőséget biztosít a megfelelő port kiválasztására is. A ’robot_controller’ az FPGA eszközt programozza fel a soros kommunikáció megvalósítására PicoBlaze segítségével, majd a megadott koordináták alapján a kiválasztott szervó motorokon megtörténik a jelgenerálás.

A robotkar a National Instruments vállalat tulajdona, mely szintén FPGA eszközök fejlesztésével is foglalkozik, ezért a szakdolgozat készítése során szintén nagy segítségünkre voltak.

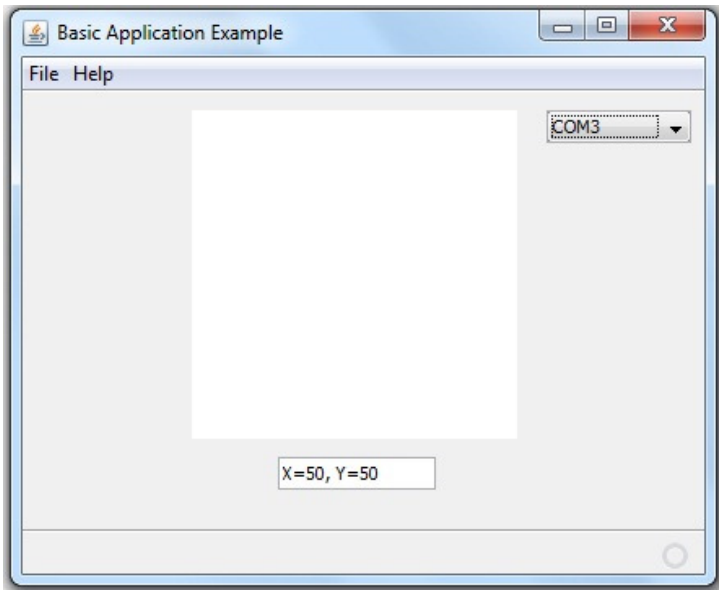
A számítógépen futó program JAVA nyelven íródott, amely elkészítéséhez a NetBeans fejlesztői környezetet használtuk. Az alap JAVA-csomag nem tartalmazza a soros kommunikációhoz szükséges állományokat, de ezek szabadon letölthetőek, majd őket a számítógépen megfelelő helyre másolva programunk alkalmassá válik a kommunikáció létrehozására.

Az FPGA-ra töltött program Verilog nyelven íródott. Elkészítésekor a Xilinx cég saját, Xilinx ISE Webpack fejlesztői környezetét használtuk fel, mely a gyártó honlapjáról egy ingyenes regisztrációt követően letölthetünk. A ’robot_controller’ projekt három fő részre osztható, melyek az UART-vezérlés, a PicoBlaze-mikrokontroller, valamint szervó motorok vezérlője.

3.1 A program részeinek működése

3.1.1 A felhasználói interfész

A számítógépen futó alkalmazás használatához rendelkezni kell a NetBeans fejlesztői környezettel, valamint a soros port kezelés fájljait a megfelelő helyre kell másolni. E fájlok elhelyezéséről a mellékelt Használati útmutatóban beszélünk



16. ábra: A grafikus interfész ablakképe

A felhasználó a program jobb oldalán található lenyíló menüvel választhatja ki, hogy melyik portot akarja használni a kommunikáció során. Ez általában a COM3 port, de ha olyan számítógépen használjuk, amely rendelkezik valós, vagy további emulált soros portokkal (például Bluetooth-kapcsolat), akkor a számozás megváltozik, így biztosítunk lehetőséget a megfelelő port kiválasztására.

A program középső részén található az irányításhoz használt felület, itt az egérkurzor célkereszttté változik. Ez egy negyed-koordináta-rendszernek feleltethető meg, vagyis a bal felső sarka a (0,0) pont, a jobb alsó sarka a (100,100) pont. Mindkét érték a két kiválasztott motor elfordulását szimbolizálja százalékban. Így 0 és 100 között tetszőleges (x,y) kitöltési tényező küldhető ki. Az x-értéket választástól függően az 1-es, 3-as, 6-os szervó motor kapja meg. Az y-értéket szintén választástól függően a 2-es, 4-es, 5-ös szervó motor kapja meg. Normál esetben két szervó motor kiválasztása javasolt, de mivel a jelterjedés párhuzamos, lehetőség van ennél kevesebb, vagy akár több motor kiválasztására.

A választófelület mellett egy ablakban a program minden esetben kiírja az utoljára kiválasztott (x,y) kitöltési tényezőt. Ez alól kivétel közvetlenül a program indítása utáni szakasz, amikor még nem lett egy érték sem kiválasztva, ekkor az ablak üres.

A program a NetBeans kimeneti ablakán folyamatosan informálja a felhasználót a kommunikáció állapotáról. Jelzi a kiválasztott portot, a kiküldött koordinátát, valamint értesít, ha nem sikerült kapcsolatot létesíteni (például ha a portot már használja egy másik program). Ha kiválasztottunk egy értéket, akkor a program megnyitja a portot, elküldi az adatot, majd zárja a kapcsolatot. Tehát csak akkor épül fel a kapcsolat, ha van küldendő adat, és csak addig él, amíg elküldi az adatot.

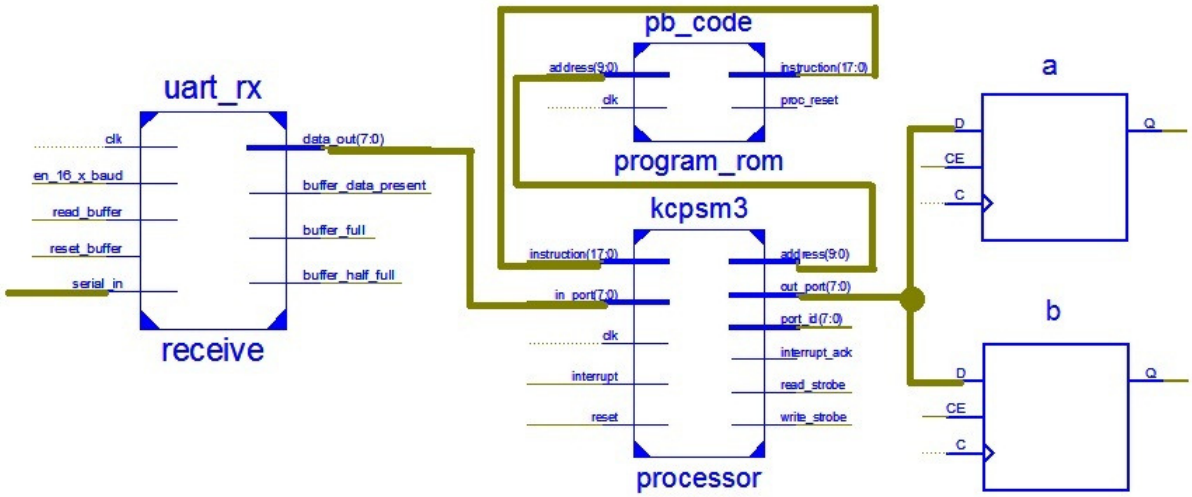
Mivel a kommunikáció létrehozása és bezárása adatforgalommal jár, ezek negatív hatással vannak a robotkar helyzetére, mozgása kiszámíthatatlanná válhatna. Ezért a kommunikáció fix formátumban, öt értékkel zajlik, mely az alábbi:

{ x | y }

ahol x az első kitöltési tényező értékét jelöli, y pedig a második kitöltési tényező értékét. Így a kommunikáció további adatforgalma nem befolyásolja a robotkar működését, a PicoBlaze figyeli a formátum helyességét. A soros kommunikáció egyszerűsítése érdekében a kitöltési tényező értékei – amelyek minden esetben a decimális 0 és 100 közé eső értékek – sztringként kerülnek kiküldésre. Így például a 0 az ASCII „null” kódjának felel meg, az 50 a „2” szám, a 100 pedig a „d” betű kódja.

3.1.2 PicoBlaze UART kommunikáció

Az FPGA-n futó program három bemeneti vonallal (rx, kapcsoló, órajel), és egy kimeneti vonallal rendelkezik. Ennek egyik legfontosabb része a PicoBlaze szoftver-processzor, valamint a segítségével megvalósított UART kommunikáció. A processzorhoz csatlakozik az általa végrehajtandó utasításokat tartalmazó program-memória, melynek neve 'program_rom'. Az UART vevő modul fogadja a bemeneten érkező adatokat a soros bemenetén. Itt visszaállítja az eredeti 8-bites adatot, melyet a 'data_out' kimenetén továbbít a PicoBlaze processzor 1-es számú bemeneti portjára. A PicoBlaze feladata a beérkező adatok feldolgozása, majd továbbítása a megfelelő portra. A 17. ábrán az UART modul és a PicoBlaze sematikus ábrája látható



17. ábra: A PicoBlaze és az UART modul sematikus rajza

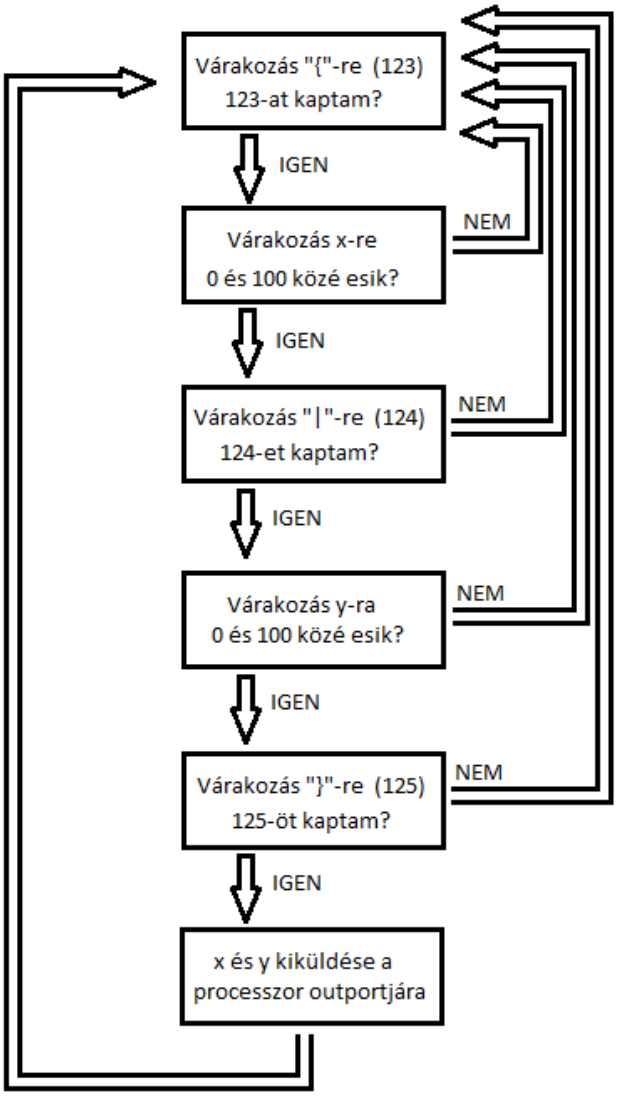
A feldolgozás mindig egy vizsgálattal kezdődik, vagyis ellenőrzi, hogy a bemeneti adat érvényes-e (vagyis a decimális kódja nagyobb 0-nál). Ezt követően vizsgálja, hogy a bejövő adatok megfelelnek-e az utasítás-formátumnak, ami a következő: **123 x 124 y 125 =>{ x | y }**

Tehát:

- Első lépésben 123-nak kell érkeznie. Siker esetén az utasítás-számláló az 1 értéket veszi fel. Ellenkező esetben az utasítás-számláló 0.
- Második lépésben 0-100 közötti számnak kell érkeznie, ezt eltárolja egy 'a' változóba. Siker esetén az utasítás-számláló a 2 értéket veszi fel. Ellenkező esetben az utasítás-számláló 0.

- Harmadik lépésben 124-nek kell érkeznie. Siker esetén az utasítás-számláló a 3 értéket veszi fel. Ellenkező esetben az utasítás-számláló 0.
- Negyedik lépésben 0-100 közötti számnak kell érkeznie, ezt eltárolja egy ’b’ változóba. Siker esetén az utasítás-számláló a 4 értéket veszi fel. Ellenkező esetben az utasítás-számláló 0.
- Ötödik lépésben 125-nek kell érkeznie. Siker esetén az utasítás-számláló nullázódik.
- Bármely lépésnél, ha a beérkező adat nem felel meg a feltételnek, az utasítás-számláló nullázásra kerül, és a teljes utasítást meg kell ismételni.

A feldolgozás folyamatábrája:

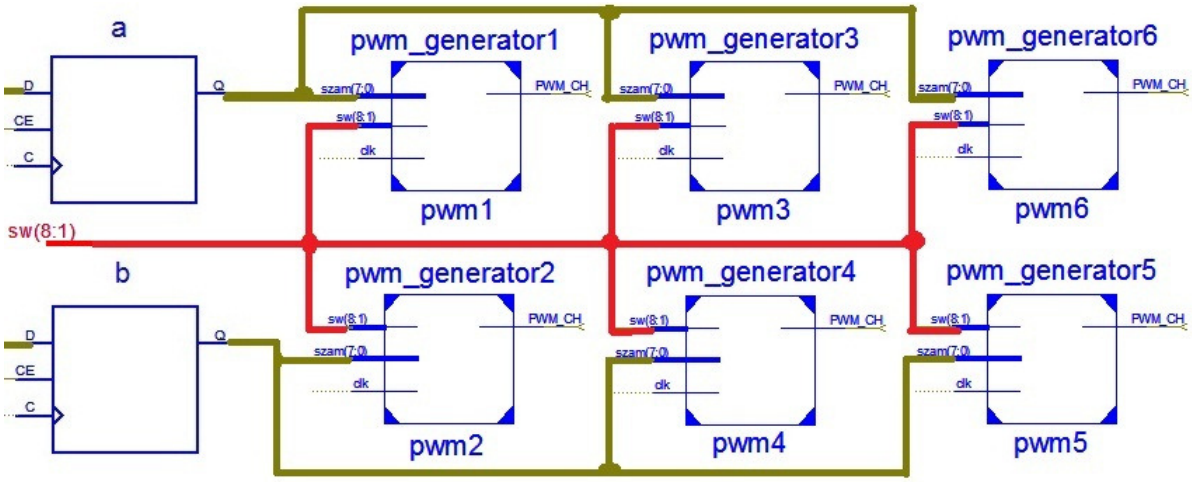


18. ábra: Az UART kommunikáció folyamatábrája

Ezek után az ’a’ változó értéke, vagyis az x-kitöltési tényező értéke kiküldésre kerül a processzor 1-es számú portján, az ’out_port’ vonalon, valamint a ’b’ változó, vagyis az y-kitöltési tényező értéke a processzor 2-es számú portján, szintén az ’out_port’ vonalon. Az értékek csak abban az esetben kerülnek kiküldésre, ha mindkét kitöltési tényező érvényes, és az utasítás formátuma helyes volt. Így elkerülhető, hogy a robotkar olyan esetben is megmozduljon, amikor az utasítás nem teljes, mert az hibás, vagy hiányos. A kiküldött értékeket a szervó motorok PWM generátora dolgozza fel.

3.1.3 PWM jelgenerálás

A Logsys-panelen futó program másik fontos része a szervó motorok irányítását megvalósító PWM jelgeneráló modulok. Mivel mindegyik szervó motornak kisebb-nagyobb mértékben eltérő jelek szükségesek, ezért minden motornak saját modulja van. Alaphelyzetben, mikor a program feltöltésre kerül a kártyára és még nem érkezett egy utasítás sem, mindegyik szervó motor a saját kitöltési tényezőjének az 50%-át veszi fel, így a robotkar középállásban van. Mindegyik jelgenerátor három bemeneti (kitöltési érték, kapcsoló, órajel) és egy kimeneti vonallal rendelkezik. A két bemenet közül az egyik a kapott szám, a másik pedig az adott szervó motorhoz rendelt kapcsoló állapota. A kimeneti vonalon természetesen a generált PWM jel kerül továbbításra. Az alábbi ábrán PWM jelgenerálás sematikus ábrája látható.



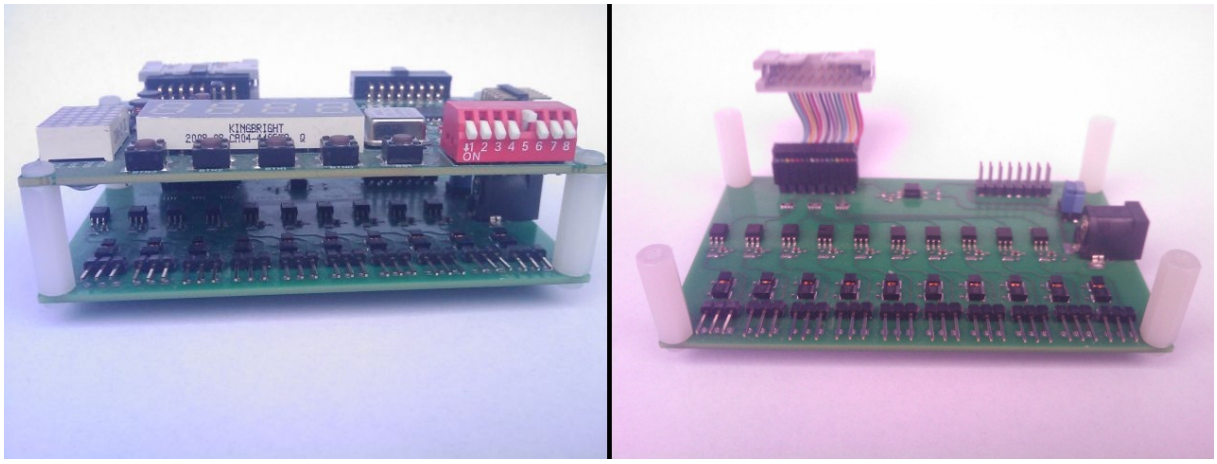
19. ábra: A PWM jelgenerálás sematikus rajza

Amennyiben egy megadott kapcsoló logikai 0 állapotban van, úgy a hozzá tartozó szervó motorhoz beérkezett érték nem módosítja a generált PWM jelet. Ha a kapcsoló logikai 1 értéken van, úgy megvizsgálja a modul, hogy a bejövő érték 0 és 100 közé esik. Ha igen, akkor a modul a $Generált\_PWM = 0°kitöltés + (érték * 1°kitöltés)$ képletnek megfelelően generál egy jelet. Bizonyos szervó motoroknál konstrukciós okokból az értéket 100-ból kell kivonni. Így a szervó motor mozgása ellentétes lesz, viszont a kapott elmozdulás megfelelő lesz az interfészben elvárttal.

A jelgenerálás első lépése egy 50Hz frekvenciájú jel létrehozása. A Logsys-panel oszcillátora 16MHz-es frekvenciával rendelkezik, ezért az órajel osztására van szükség. Ehhez a bemeneti órajelet el kell osztani 320000-el, mivel $\frac{16000000}{320000} = 50$.

A következő lépésben beállítjuk a kiküldendő impulzus szélességet a fent ismertetett egyenlet segítségével, majd kiküldjük a Logsys-panel 'A' jelzésű bővítőcsatlakozóján, melyre a 10 csatornás PWM kiegészítő panel csatlakozik. A panel külső 5V-os tápfeszültséget igényel, melyet szétoszt mind a 10 csatornára, valamint további 10 vonal csatlakozik a Logsys-panelre. A bővítőcsatlakozón keresztül a kiegészítő panelre továbbítódik a generált impulzus, mely a robotkar esetében 6 csatornát jelent a rendelkezésre álló 10-ből. Mivel mindegyik csatorna egy meghatározott FPGA I/O lábra csatlakozik, ezért ügyelni kell a a robotkar megfelelő csatlakoztatására.

Az alábbi ábrán a kiegészítő panel látható a Logsys-panellel együtt, valamint önállóan:



20. ábra: A kiegészítő panel

3.2 Tapasztalatok

Az általunk megvalósított robotkarvezérlés a végső finomhangolások után könnyen használható. Az egér beviteli eszközként történő alkalmazása növelte a robotkar pontosságát. Az egyszerű felhasználói interfész előnye a könnyű kezelhetőség, nem szükséges mély elektronikai ismeretekkel rendelkeznie felhasználójának. Kézügyesség és gyakorlás segítségével a robotkar (lehetőségeihez mérten) finom mozgásokra is képes. Azonban figyelembe kell venni a robotkar konstrukciós adottságait, vagyis bizonyos pozíciókból a terhelés miatt nem képes megmozdulni, ilyenkor kézzel rá kell segíteni. Ezen kívül érdemes kerülni a nagyobb ívű mozgásokat, mivel ilyenkor a robotkar bebillen, és akár fel is borulhat. Ilyen esetekben célszerű előre gondoskodni a robotkar megfelelő rögzítéséről. Első használat előtt célszerű elolvasni a használati útmutatót, mivel a felhasználói interfész, és a Logsys-panel használatához külön szoftverek szükségesek.

3.3 Fejlesztési lehetőségek

Szakdolgozatunkban sikerült egy olyan robotkarvezérlést megvalósítani, ahol a felhasználó a számítógépen egy egyszerű 2 dimenziós felület segítségével irányítani tudja a robotkart, az kisebb feladatok végrehajtására is alkalmassá teheti. A szakdolgozat elkészítésére megadott idő alatt egy ilyen komplett vezérlés elkészítését határoztuk meg. Mivel mások is fognak ebben a témában szakdolgozatot készíteni, így célszerű munkánkat olyan irányban fejleszteni, mely során a felhasználói interfész interaktívabbá válik. Erre jó példa az MTA SZTAKI által fejlesztett VirCA, ahol 3 dimenziós térben a robotkar modellje segítségével interaktív módon lehet irányítani az eszközt, akár távolról, interneten. Ezenkívül napjainkban válnak elérhetővé olyan beviteli eszközök, melyek térbeli mozgásokat is képesek érzékelni, ezek használata szintén izgalmas lehet a robotika témakörében.

4. Köszönetnyilvánítás

Ezúton szeretnénk köszönetünket kifejezni Dr. Végh János Tanár Úrnak, hogy a téma kiírásával lehetővé tette számunkra, hogy szakdolgozat keretein belül megismerkedhessünk a robotika és az FPGA technológiáival. A megvalósítás során ötleteivel segítette a munkánkat.

Továbbá köszönjük a National Instruments vállalatnak, azon belül Nagy Gábornak, hogy rendelkezésünkre bocsátotta a robokart és a használatához szükséges PWM-interfészt, valamint Mácsi Zoltánnak, aki szakértelmével nagyban hozzájárult a szakdolgozat sikeréhez.

Végül, de nem utolsó sorban köszönetünket fejezzük ki Dr. Fehér Béla Tanár Úrnak az FPGA technológia, és Dr. Oniga Istvánnak a PicoBlaze processzor megismertetéséért.

5. Összefoglalás

Szakdolgozatunkban egy olyan komplex robotkarvezérlés elkészítését tűztük ki célul, amelyben a felhasználó egy számítógépen futó grafikus program segítségével, egér használatával képes mozgásra bírni egy robotkart, melyet a National Instruments vállalat fejlesztett, és bocsátott a rendelkezésünkre. A feladatot csoportmunkában valósítottuk meg.

Előttünk már készítettek a robotkarhoz olyan vezérlést, amely előre letárolt mozgássorozatot hajtott végre, később egy olyat, amiben egy billentyűzet segítségével valósították meg az irányítást. Már akkor felmerült az igény egy grafikus program használatára. Miután egyeztettünk a külső konzulenssel, valamint a témavezetővel az elkészítendő munkáról, elkezdtünk megismerkedni a megvalósításhoz szükséges technológiákkal.

Munkánk során megismerkedtünk PC-n a Java alkalmazásfejlesztéssel és a soros port kezeléssel, az FPGA-n a szoftver-processzorral, az UART kommunikációval és a PWM vezérléssel. Ezen kívül betekintést nyertünk a robotika világába.

A grafikus interfész elkészítése után a kommunikáció megvalósítása érdekében szükséges volt az FPGA-n egy olyan vezérlés elkészítése, mely segítségével létrejön a kapcsolat az FPGA és a PC között. Mivel a felhasznált Logsys-panelen Xilinx Spartan 3 típusú FPGA található, így lehetőségünk nyílt a Xilinx cég PicoBlaze szoftver-processzor, valamint az UART kommunikációs modul használatára. Ezután elkészítettük a robotkar mozgatását megvalósító PWM vezérlést. Végül összehangoltuk a grafikus programot a vezérléssel, hogy a mozgatás irányhelyes és minél pontosabb legyen.

Az elkészült projekttel sikerült a robotkar pontosságát növelni, mivel itt a felhasználó szabja meg, hogy mekkora mértékű legyen az elmozdulás. Ez lehet egész kicsi (1.8°), de lehet akár 180°-os is. Előnyös a grafikus felület használata is, mert így egyszerűen kezelhető.

Munkánk a későbbiekben továbbfejleszthető, mivel más interfész is használható, a vezérlés változtatása nélkül. A hétköznapjainkban egyre népszerűbb virtuális terek, valamint térbeli mozgások lekövetésére használt beviteli eszközök alkalmazása további izgalmas lehetőséget kínálnak a robotika világában.

6. Irodalomjegyzék

1. Dr. Végh János: Bevezetés a Verilog hardverleíró nyelvbe (egyetemi jegyzet)
2. Fehér Béla: Digitális rendszerek tervezése FPGA áramkörökkel
3. Nyékiné Gaizler Judit: Java 2 útikalauz programozóknak 1.3
4. Tóthfalusi Tamás: FPGA alapú robotkarvezérlés megvalósítása
5. Pong P. Chu: FPGA Prototyping by Verilog Examples
6. Logsys Spartan 3E FPGA kártya felhasználói útmutató: http://logsys.mit.bme.hu/sites/default/files/page/2009/09/LOGSYS_SP3E_FPGA_Board.pdf
7. PicoBlaze 8-bit Embedded Microprocessor User Guide: http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf
8. http://en.wikipedia.org/wiki/Pulse-width_modulation
9. http://hu.wikipedia.org/wiki/UART
10. http://www.szgt.uni-miskolc.hu/~csaki/robot.pdf
11. http://www.fzolee.hu/framework/files/Rob1.pdf
12. http://www.mestersegesintelligencia.hu/doc/ipari%20robotok.php
13. http://hu.wikipedia.org/wiki/Robot
14. http://myprojects.hu/pic/szervo-motorok-vezerlese.html
15. http://home.mit.bme.hu/~feher/verilog/P1hu.pdf
16. http://www.inf.unideb.hu/~jvegh/edu/hw/verilog/handout/VerilogIntro.pdf
17. http://home.mit.bme.hu/~feher/fpgasopc/wpack112_bev.pdf
18. http://hu.wikipedia.org/wiki/Java_(programoz%C3%A1si_nyelv)
19. http://www.hwsw.hu/hirek/44766/netbeans-java-javafx-osgi-szoftver-fejlesztes.html
20. http://rtime.felk.cvut.cz/~krakorj/lib/exe/fetch.php?id=projekty%3Afpga_ml403&cache=cache&media=projekty:pccomp_manual.pdf
21. http://digitus.itk.ppke.hu/~marbe/2005_X_robot_manipulatorok.pdf

7. Melléklet

- Használati útmutató
- Képek a robotkarról működés közben

Használati útmutató

A robotkarvezérlés használatához szükséges szoftver-követelmények:

- NetBeans fejlesztői környezet (letölthető a http://www.netbeans.org oldalról)
- Logsys GUI az FPGA állomány feltöltéséhez
- Xilinx ISE Webpack programcsomag az FPGA állomány konverziójához

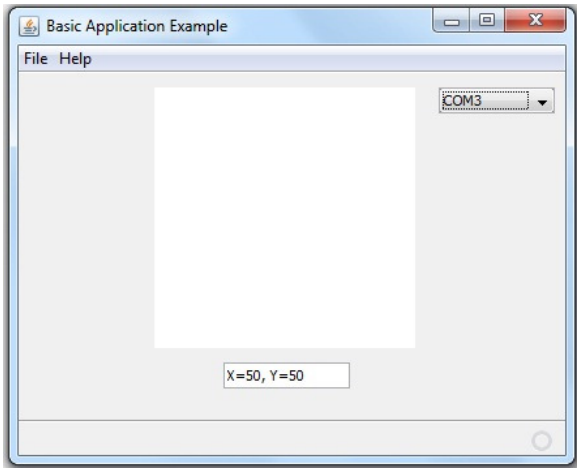
A soros porti kommunikációhoz a megadott fájlokat az alábbi helyekre kell másolni:

- win32com.dll: Másoljuk a fájlt a \Java\jdk\bin mappába.
- comm.jar: Másoljuk a fájlt a \Java\jdk\jre\lib\ext mappába.
- javax.comm.properties: Másoljuk a fájlt a \Java\jdk\jre\lib mappába

A program indítása:

- A Logsys GUI segítségével töltsük a 'top.bit' fájlt az FPGA-ra.
- Importáljuk a 'robot_interface' projektet a NetBeans programba, majd indítsuk el.
- A vezérlés használatra kész.

A Logsys-panelen a DIP-kapcsolók segítségével válasszuk ki a vezérelni kívánt szervó motorokat. A kapcsolók számozása megegyezik a szervók számozásával. A kapcsoló felbillentett állapota jelenti a kiválasztott állapotot.



A program jobb oldalán válasszuk ki a használni kívánt portot, majd a fehér négyzetben kattintva irányíthatjuk a robotkart.

Képek a robotkarról működés közben