LabVIEW

Mik a beágyazott rendszerek tervezésének problémái? Milyen egy teljes tervezési ciklus folyamata? V-modell ismertetése.

Röviden: A HW és SW problémák alapja egyaránt az egyre növekvő komplexitás. A tervezés folyamata egy absztrakcióban "lefelé" és egy "felfelé" tartó szakaszból áll (V-modell): a tervezés során magas szintű tervekből és követelményekből alacsonyabb szintűeket majd implementációt készítünk, majd a tesztelést alulról felfelé haladva egyre magasabb szinten végezzük.

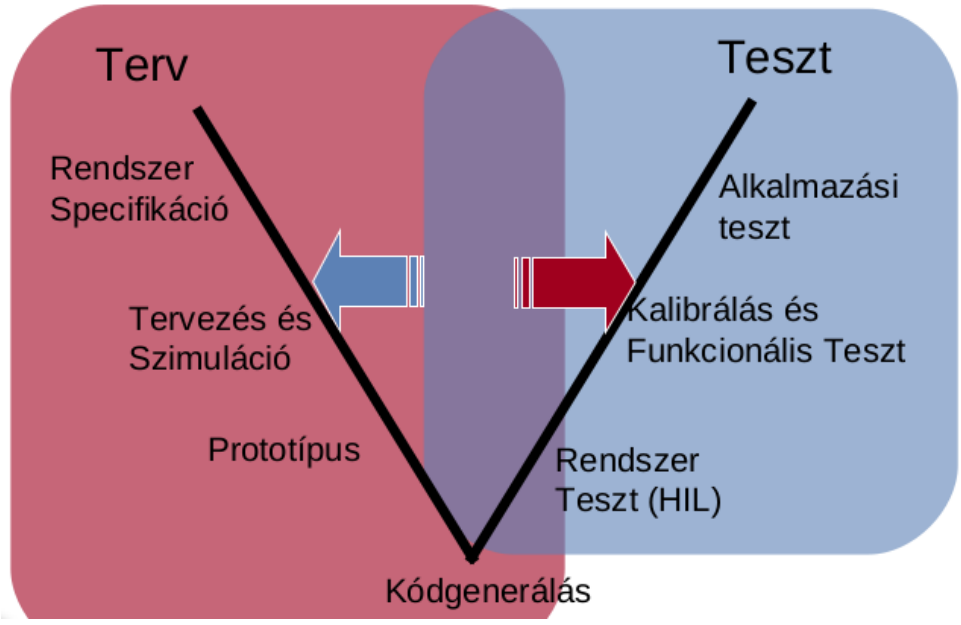
A beágyazott rendszerek tervezésének problémái hardveres és szoftveres problémák egyaránt, azonban mindegyikben közös, hogy a beágyazott rendszerek bonyolultságának folyamatos és egyre gyorsuló növekedése a kihívások alapja.

Hardveres problémák:

- Komplex megoldások és szabványok megvalósítása: egyre nagyobb sebességű Ethernet, bonyolult USB szabvány, multimédiás (audio/video) jelfeldolgozás HW támogatásának igénye és összetettsége

Szoftveres problémák:

- A szoftverméret exponenciális növekedése
- A termékfejlesztés költségeinek 50%-a a szoftverhez köthető
- A legtöbb beágyazott rendszer fejlesztése szoftverproblémák miatt csúszik



1. ábra. A V-modell

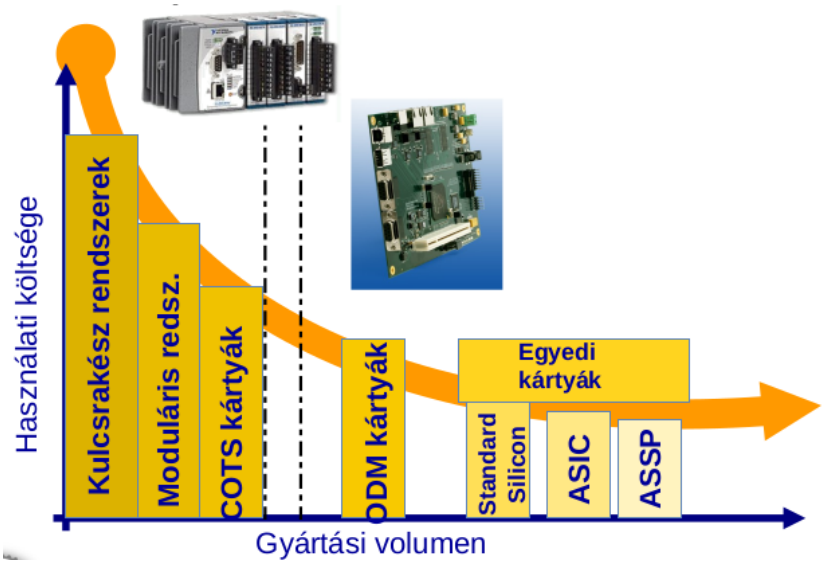
A fejlesztés menete:

A fejlesztés a legmagasabb absztrakciós szintről kezdődik, ahol a termék, mint ötlet megszületik. A termékkel, alkalmazással kapcsolatos elvárásokat itt rögzítik. Ezt követi a termék, mint rendszer technikai specifikációja, a teljes rendszerrel szemben támasztott, mérhető követelmények leírása. A rendszert ezután fokozatosan részletezik, az egyes alegységeket tervezik és működésüket szimulálják, majd ezekkel a részegységekkel prototípust állítanak össze. A prototípus sikeressége esetén megkezdődhet a legalacsonyabb szintű implementáció, a kézi vagy automatizált kódgenerálás. A megvalósítás végeredményét a legkisebb, legegyszerűbb egységektől felfelé, a teljes rendszer szint felé haladva tesztelik és verifikálják, melynek végső lépése a kész termék megfelelőségének vizsgálata.

A V-modell tervezési és tesztelési ágai hatással vannak egymásra. A tervezés minden szintjén specifikációkat, követelményeket fogalmazunk meg a rendszerrel és alegységeivel szemben, melyeket később a tesztelési fázisban a működés helyességének megítélésére használunk. A tesztelési fázis eredményei pedig visszahathatnak a tervezési lépésekre oly módon, hogy a helytelenül vagy nem elég jól működő rendszer vagy alegység hibáit felfedik.

Milyen módon redukálható a tervezési idő/tervezési költség? Mikor melyik választás adja a legkedvezőbb eredményt?

A tervezési idő és -költség egymással összefüggésben lévő tényezők: a hosszabb ideig tartó fejlesztések költségben is könnyedén nagyra nőnek. Cél tehát a fejlesztési idő és ezáltal a költségek csökkentése. A fejlesztés egyértelműen lerövidül, ha a terméket nem "nulláról" kezdjük el megtervezni vagy gyártani, hanem ehhez felhasználunk meglévő félkész vagy akár kész megoldásokat, melyeket már csak az alkalmazásukra kell szabni. A kész egységek vagy akár rendszerek alkalmazásának persze ára van, hiszen azt valaki másnak el kellett készítenie, a kérdés az, mennyit vagyunk hajlandóak fizetni ezért a kész megoldásért. A választ nagyban befolyásolja, hogy milyen módon épül be ez a költség magába a termékbe és hány terméket szeretnénk forgalomba hozni.



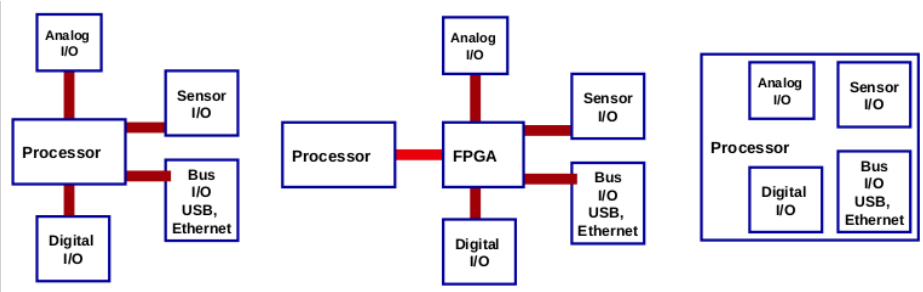
2. ábra. Termékek fejlesztésének ára a darabszám függvényében

Az egyedi termékek magukon viselik a teljes fejlesztési folyamat költségeit, ezért jellemzően drágábbak, mint a tömeg-gyártott termékek, melyek fejlesztési költsége sok száz, ezer vagy akár millió termék árában oszlik szét.

Egyedi termék fejlesztése esetén, mivel annak ára eleve magas lenne, érdemes kész megoldásokat alkalmazni, melyek drasztikusan csökkentik a fejlesztési időt, ezzel a lehető legalacsonyabban tartva az egyetlen vagy néhány termék árát. Nagyobb darabszámban gyártott termékek esetén a

fejlesztés költségei jobban megoszlanak, ezért csak moderált mértékben érdemes kész, relatíve drága megoldásokat bevetni.

Általános rendszer-felépítési modellek. A LabVIEW által kínált heterogén rendszerfelépítés (Host - Real-time - FPGA) ismertetése, az egyes rétegek/egységek szerepe, alkalmazási területe, funkciói.



3. ábra. Rendszerfelépítések: a) processzorközpontú, b) FPGA központú, c) integrált

Az FPGA használatának szempontjai a LabVIEW környezetben.

A LabVIEW grafikus, adatfolyamgráf alapú fejlesztési környezete az FPGA hardver felépítéséhez jól illeszkedik: a fejlesztőkörnyezet elősegíti a hardveres szemléletet, a párhuzamos programszervezést, a hierarchikus felépítést.

Ez a rész egyelőre erősen csonk...

Mi a jellemzője a közvetlenül az FPGA-n futó ill. a Windows rendszerben emulált működésnek? Mi az értelme ezen utóbbi (Windows Target mód) használatának?

A LabVIEW-ban fejlesztett adatfeldolgozó láncot bitfájllá lehet fordítani, majd valós FPGA hardverre letölteni. Ennek a folyamatnak az előnye, hogy valós körülmények között tesztelhetjük az alkalmazást, a késztermékhez hasonlóan a műveletvégzés FPGA-ban történik. A LabVIEW ehhez a fajta teszteléshez interaktív felületet is biztosít, az FPGA konfigurációját, mint VI-t egy hozzá tartozó Front Panel-en keresztül érhetjük el, ekkor azonban a valós idejű végrehajtás miatt hibakeresésre nincs lehetőség.

A Windows Target mód lényege, hogy az FPGA számára generált VI-t nem tesszük tényleges hardverbe, hanem szoftveresen emuláljuk, ezzel lehetőséget nyerünk a műveletvégzés megállítására és a hibakeresésre. Az így emulált működés viszont nem rendelkezik a valós hardverre jellemző időzítésekkel.

Az FPGA használati jellemzői: sebesség, párhuzamosíthatóság, erőforrásigény. Mit jelent a megosztott erőforrás és mik tartoznak ebbe a kategóriába?

A National Instruments FPGA hardverének működési órajele 40 MHz. A 40 MHz biztos betartásához a LabVIEW automatikusan pipeline regisztereket illeszt be az adatfolyamgráfba.

Az adatfolyamgráf jellegű programozás jól illeszkedik az FPGA-khoz, a párhuzamosítás természetes velejárója a fejlesztőkörnyezet gondolatvilágának, a feladatok párhuzamosítása azonban erőforrás-igényes.

Azokban az esetekben, amikor egy erőforrást nem foglal le teljesen egy adott feladat végzése vagy véges száma miatt nem jut minden feladatnak saját belőle, megosztható, időben multiplexálható több feladat között, az ilyen erőforrásokat nevezzük megosztott erőforrásoknak.

Megosztott erőforrások esetén fontos, hogy egyszerre csak egy feladat vegye igénybe az erőforrást, a konkurens hozzáférések a feladatok interferálásához és hibás eredményekhez vezetnek.

Megosztott erőforrások lehetnek:

- digitális kimenetek,
- memóriák, FIFO tárolók,
- nem újra-beépülő VI-ok,
- lokális változók.

Mi a különbség az újra beépülő és a nem újra beépülő FPGA VI-ok között?

A LabVIEW-ban alapértelmezésben minden VI nem újra-beépülő, vagyis egyetlen példányban képződik le az FPGA-ban. Erőforrásokkal kapcsolatos beállítások közt ez megváltoztatható, ekkor létrehozhatók újra beépülő VI-ok, melyek annyi példányban jönnek létre az FPGA-n, ahány példányban lehelyeztük őket az adatfolyamgráfban.

Milyen műveleti javaslatok/korlátozások érvényesek az FPGA-kra és miért?

A legfontosabb korlátozás, hogy nincs FPGA-ra való tervezés esetén lebegőpontos műveletvégzés, sem egyszeres-, sem dupla pontosságú. Ennek oka, hogy a speciális számábrázolás bonyolult logikát igényel, melyre nincsenek beépített erőforrásai az FPGA-knak, így nagy mennyiségű univerzális logika kerülne felhasználásra. Ehelyett javasolt az egész- vagy az ezzel bitszinten egyenértékű fixpontos számábrázolást használni, mert ilyen formátumú számok támogatására szabottak például az FPGA-k DSP szeletei is és a számítások kevésbé költségesek (erőforrás-igényesek). Fixpontos számábrázolás esetén a szorzás skálázással, kettő hatvánnyal való szorzás/osztás pedig léptetéssel valósítható meg, amely tovább csökkenti a szükséges erőforrások mennyiségét.

Mivel az egész/fixpontos aritmetika dinamikatartománya a lebegőpontos számábrázoláshoz képest meglehetősen kicsi, ezért könnyedén előfordulhat túlcsordulás. A túlcsordulással kapcsolatos problémák mérséklése érdekében javasolt szaturációt alkalmazni átfordulás helyett.

Soroljon fel problémás műveleteket és ezek elkerülésének lehetőségeit!

Milyen alapvető digitális kommunikációs lehetőségeket biztosítanak az FPGA-k a LabVIEW rendszerben?

Az FPGA lábai (I/O blokkok) más egységekhez hasonlóan egyszerűen beilleszthetők és konfigurálhatók.

Rendelkezésre álló I/O típusok:

- digitális I/O: egyszerű boolean változók írhatók ki, olvashatók be rajta,
- ADC/DAC I/O: összetett értékek írására, olvasására használhatók (hardveres támogatás az R-szériás kártyákon)
- Rögzített, előre kiosztott FPGA lábak az adott fejlesztőkártyán: ezek használatához célszerű mintaalkalmazások alapján dolgozni.

A digitális I/O-k által megvalósítható protokollok:

- Komponensek, IC-k közötti kommunikáció: PWM, SPI, I2C, JTAG, PS/2, ...
- Rendszer szintű kommunikáció: MIL-STD-1553, CAN, MOST, ...
- Távközlés, űrkutatás: PCM, telemetria
- Fogyasztói elektronika: S/PDIF, I2S
- Egyedi, eszközspecifikus protokollok

Milyen ütemezési, időzítési lehetőségek vannak az LabVIEW FPGA rendszerben? Hogyan lehet ciklusokat időzíteni?

A LabVIEW FPGA rendszerben több, konfigurálható időzítési függvény is a rendelkezésünkre áll:

- Hurok időzítő: a ciklusvégrehajtás során az iterációk közötti várakozás hosszát adja meg system tick-ben.
- Várakozás: a beállított mennyiségű system tick megtörténtéig várakoztatja a végrehajtást.
- Ütemszámláló: az indulása után minden egyes system tick-nél növeli az értékét; jellemzően több ütemszámláló értékének különbségét használják pl. futási idő mérésére.

Ciklusok időzítésére a fent felsoroltak közül az első kettő a leginkább alkalmas: az első kifejezetten erre a célra létrehozott eszköz, míg a második egyszerű várakozás. Alkalmazásuk között különbséget jelent, hogy míg a hurok időzítő csak az egyes iterációkat választja el egymástól, de az első iterációt nem "várakoztatja", addig az egyszerű várakozás az első iterációra is hat.

Mit jelent az egy órajelciklus alatti műveletvégzés?

A LabVIEW FPGA rendszere alapértelmezetten 40 MHz-es órajel-frekvenciát alkalmaz. Ahhoz, hogy ezt a 40 MHz-et biztosan tartani lehessen, a LabVIEW automatikusan pipeline regisztereket épít be az adatfolyamgráfba, ezek azonban megnövelik a gráf késleltetését. Abban az esetben, ha egy összetettebb művelet elvégzését is szeretnénk egyetlen órajel-periódus alatt elvégezni, utasíthatjuk a LabVIEW környezetet a pipeline regiszterek elhagyására, ekkor persze a mi feladatunk azt biztosítani, hogy a létrejövő kombinációs logika késleltetése ne haladja meg a két felfutóél között rendelkezésünkre álló időt.

Hogyan lehet interfészeket kialakítani az FPGA VI-okhoz?

Interfészeket felügyelő VI-ok létrehozásával építhetünk be, melyek az FPGA VI és a Host VI közötti kommunikáció megvalósítására hivatottak. Ezek a VI-ok az *Open FPGA VI*, *Read/Write Control*, *Invoke Method* és *Close FPGA VI* elemekből építhetők fel.

- Open FPGA VI: az FPGA-val való kapcsolat megnyitására szolgál, a VI vagy a bitfájl kiválasztása és megnyitása után egy referenciát kapunk az eszközre. A kezelőfelület kijelző- és beavatkozó elemei ezen a referencián keresztül kommunikálnak az eszközzel.
- Read/Write Control: az FPGA-val való adatcserét képes lebonyolítani, kétirányú kommunikációt tesz lehetővé, melyben a kezelőfelületnek szigorúan előbb a vezérlő elemek adatinak írása, majd ezután a kijelző elemek adatainak beolvasása történik. A kommunikációhoz egyszerű és összetett adatok, adatstruktúrák is alkalmazhatók.
- Invoke Method: az esemény alapú vezérlés eszköze, tulajdonképpen szubrutinhívás (futtatási parancs, letöltés, várakozás, megszakításkérés stb.).
- Close FPGA VI: a kommunikációs interfész lezárására szolgál, a kommunikáció / interfész-definíció végét jelzi.

Adatátvitelek szinkronizálásának jellemzői

A LabVIEW környezetben alapvetően minden aszinkron: az FPGA és a Host egymáshoz képest aszinkron módon működik, de az egymástól független VI-ok, adatfolyamok működése is aszinkron. Ennek megfelelően a Host és az FPGA (a vezérlés és az adatfeldolgozás) közötti kommunikáció ennek megfelelően aszinkron, mely igény esetén szinkronizálható megszakítások alkalmazásával.

Az FPGA-ba beépíthető megszakításkérő VI, a Host oldalon pedig várakozhatunk erre a megszakításra (opcionálisan az FPGA is várhat a megszakításkérés nyugtázására, ez a kézfogásos szinkronizáció). Fizikai megszakításvonalból egy darab van, de 32 logikai megszakítás alkalmazása lehetséges.

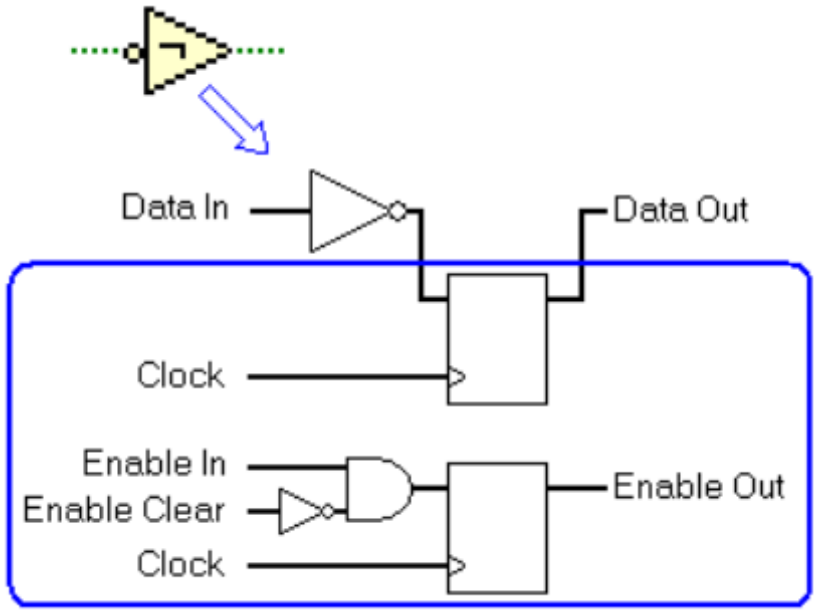
DMA tulajdonságok

A DMA adatátvitel háromféle lehet (az NI R-szériás HW-ein):

- Blokkoló: az adatátvitel elején be kell állítanunk az átvinni kívánt adat mennyiségét és az erre szánt maximális időkeretet (timeout). A blokkoló DMA átvitel processzor intenzív, de nagyon gyors és a megadott időkorláton biztosan nem lóg túl akkor sem, ha nem sikerült minden adatot átvinni.
- Lekérdezéses: az átvitel előtt lekérdezzük az elérhető adatok számát, majd aztán ezt az adatmennyiséget DMA-val beolvassuk. Kevésbé processzor intenzív módszer, de még mindig gyors.
- Megszakításos: az FPGA-ban halmozódó adat egy előre megadott mennyiség elérésekor megszakítást generál, a Host pedig ezt a megszakításkérést feldolgozva olvassa be a felgylt adatmennyiséget. Ez a módszer terheli legkevésbé a processzort, ugyanakkor az adatok felgyülemlésére való várakozás miatt lassú is.

Hogyan működik a LabVIEW FPGA rendszerben az elemi műveletek HW megvalósítása? (Alapfunkció + regiszterezés + engedélyezés)

A LabVIEW FPGA rendszerben minden művelet regiszterezett, illetve a műveletek helyes sorrendjének megtartásához az adatfolyam minden eleme egy engedélyezőláncra van fűzve. Ennek megfelelően minden elemi művelet az alapfunkcióján kívül tartalmaz egy regisztert a kimenetén és az engedélyezőlánc rá eső készét.



4. ábra. Elemi művelet és annak kiegészítései

Milyen lehetőségek vannak a sebesség optimalizálására?

A sebességoptimalizálás aspektusai:

- Párhuzamosítás, párhuzamos ciklusvégrehajtás: az adatfolyamgráf műveleteinek párhuzamosításával a teljes algoritmus lefutásához szükséges idő csökkenthető az erőforrás-igény növekedése árán. A LabVIEW grafikus programozási felülete ösztönzi a párhuzamosítást és valódi párhuzamos végrehajtást valósít meg.
- Pipeline-osítás: az órajel-frekvencia növelése érdekében az egyes műveleteket szakaszokra bonthatjuk, melyek a köztük lévő regiszterezés által időben átlapolódva működhetnek.
- Egy óraütem időzítésű ciklusok (Single Clock Tick Loops, SCTL): a ciklustörzsbe foglalt művelet egyetlen ütem alatt elvégezhető, ezzel a végrehajtáshoz szükséges szinkronizáció és az engedélyezés okozta többlet hardver minimalizálható. Értelemszerűen nem alkalmazhatók olyan elemek a ciklustörzsben, melyek bármilyen okból tovább tartanának, mint egyetlen órajel-ciklus. SCTL megvalósításához általában külön jelezni kell a LabVIEW-nak, hogy ne építsen be a logikába pipeline regisztereket.
-

Mi az előnye az egyetlen órajel alatt végrehajtott ciklusoknak? Milyen feltételek mellett használható ez a módszer? Milyen korlátozások vannak a módszer használatára? Miért lesz kisebb az erőforrásigény egy ilyen előírás után?

A LabVIEW FPGA rendszer alapértelmezetten 40 MHz-es órajelen fut. Ahhoz, hogy ezt biztosan lehessen is tartani, a LabVIEW automatikusan regisztereket illeszt az elemi VI-ok kimenetére, amely azonban indokolatlanul megnöveli az egyszerű logikák késleltetését. Ha a megvalósítandó művelet biztosan beleférne a 40 MHz-es órajel egyetlen ciklusába, a LaBVIEW utasítható a regiszterek elhagyására, ezzel a műveletvégzés is gyorsabb lesz, valamint az elhagyott hardver miatt kisebb erőforrás-igényű. A művelet órajel-perióduson belül tartása ekkor a mi felelősségünk.

A módszernek természetes korlátozása, hogy nem alkalmazható olyan VI-okra, amelyek bármi okból kifolyólag hosszabb műveletvégzéssel rendelkeznek, mint egy órajel. Ilyen VI-ok például az időzítési funkciók, egymásba ágyazott ciklusok vagy a megosztott erőforrások.