4. Digitális rendszertervezés - Nios II beágyazott processzoros rendszerek

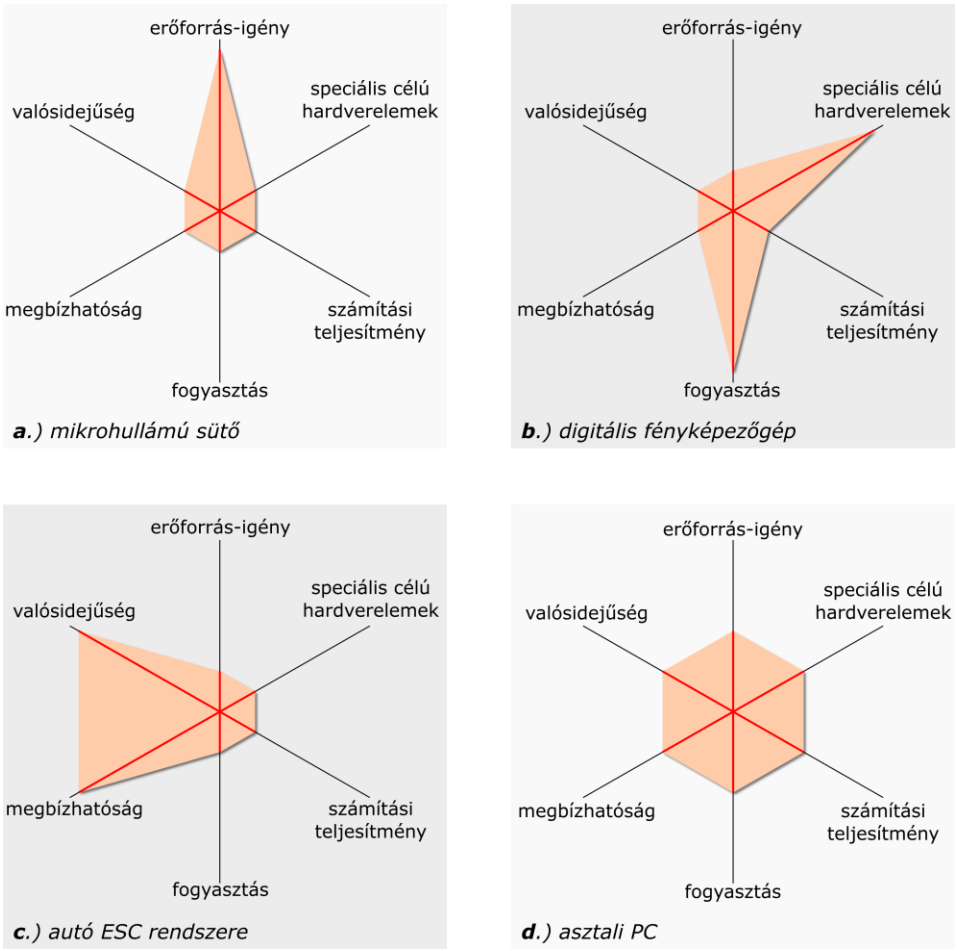
Szerző: Horváth Péter

4.1. Beágyazott rendszerek tervezési szempontjai

A beágyazott rendszerek olyan **alkalmazás-specifikus számítógépek**, amelyek a bennük megtalálható hardver erőforrások tekintetében egy specifikus feladatra optimalizáltak. Ellentétben egy általános célú számítógépes rendszerrel (pl. PC) nem önállóan használható funkcionális egységek, hanem valamilyen **nagyobb rendszer részét képezik** (innen az elnevezés). Tervezésük során az elsődleges célkitűzés, hogy **a szükséges számítási kapacitást a lehető legkevesebb erőforrás felhasználásával biztosítsák**, azonban a számítási kapacitáson és a felhasznált erőforrások mennyiségén túlmenően e rendszerek megtervezése során egyéb szempontokat is figyelembe kell venni:

- **Speciális célú hardverelemek**: Van-e a rendszerben olyan hardver egység, amely kifejezetten valamilyen szűk alkalmazási terület igényeinek megfelelően képes többlet számítási teljesítményt biztosítni (pl. DSP, titkosítás, tömörítés, mintafelismerés stb.).
- **Valósidejűség**: A rendszer válaszideje a külvilág eseményeire egyes esetekben kritikus szempont. Ezek az ún. valós idejű rendszerek *(real-time system)*, amelyek speciális hardver- és szoftvermegoldásokat igényelnek.
- **Megbízhatóság**
- **Fogyasztás**

Természetesen azt szeretnénk elérni, hogy a felsorolt tulajdonságok mindegyike szempontjából optimális rendszert állítsunk elő, de **ezek a tulajdonságok csak egymás rovására javíthatók**. Pl. egy mikroprocesszor számítási teljesítménye növelhető az órajel frekvenciájának megnövelésével, ez azonban a fogyasztást is megnöveli. A megbízhatóság redundáns hardverelemek hozzáadásával javítható, de ez az erőforrásigény megnövekedését is jelenti. Az egyes tervezési szempontok viszonylagos fontosságát - amelyet ún. radardiagramokkal szemléltethetünk - a beágyazó környezet határozza meg (4-1. ábra).

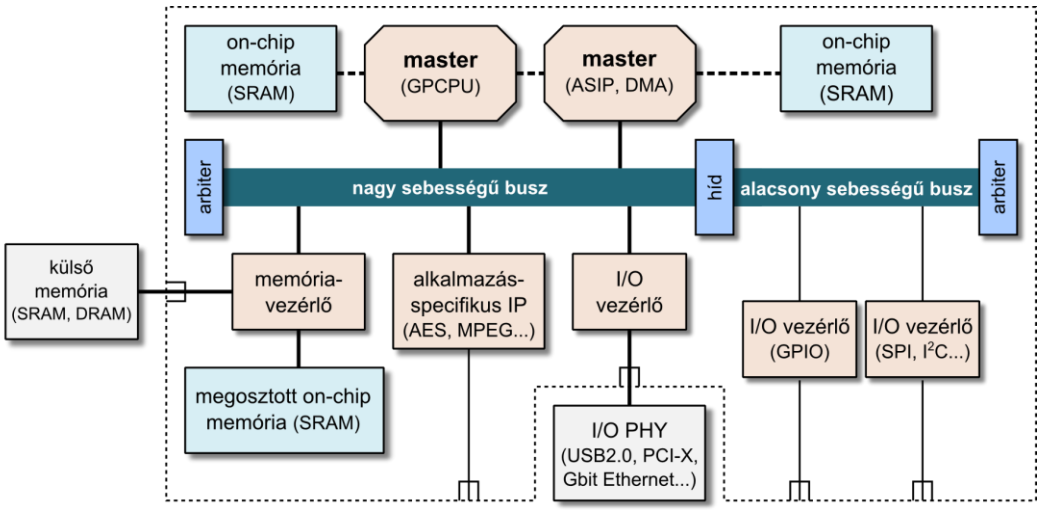


4-1. ábra Radar diagramok a beágyazott rendszerekkel szemben támasztott követelmények szemléltetésére

Egy mikrohullámú sütő beágyazott processzoros rendszerében a valós idejű működés nem követelmény, azonban a termék ára minimalizálandó (4-1. ábra: *a*). Egy digitális fényképezőgép esetén - annak hordozható jellegéből adódóan - fontos szempont a fogyasztás, és a képfeldolgozási és adattárolási feladatok miatt a speciális célú hardverelemek számítási teljesítménye is meghatározó jelentőségű (4-1. ábra: *b*). Egy autó ESC *(Electronic Stability Control)* rendszere esetén a valós idejű működés elengedhetetlen, és a megbízhatóság is kulcsfontosságú, míg a fogyasztás nem kritikus szempont (4-1. ábra: *c*). Egy általános célú számítógép esetén (pl. asztali PC) nincs kiemelkedő jelentőségű, a többinél fontosabb optimalizációs szempont 4-1. ábra: *d*).

4.2. A beágyazott rendszerek komponensei

A beágyazott rendszerek központi adatfeldolgozó alrendszerei alkalmazás-specifikus áramkörök, de szerkezetük általában követi a 4-2. ábrán látható mintát.



4-2. ábra A beágyazott rendszerek hardverelemei

Master egységek

- A különböző hardverelemek egy központi mikroprocesszor köré vannak szervezve. Ezek az **általános célú mikroprocesszorok** (*General Purpose Central Processing Unit*, GPCPU) a mikrokontroller-alapú rendszereknél megszokottakhoz képest nagyobb számítási teljesítményű RISC *(Reduced Instruction Set Computer)* magok. A leggyakoribb ad-hoc ipari szabványok az ARM, a PowerPC és a MIPS architektúrákra épülnek. Feladatkörük a beágyazott **operációs rendszer** (ha szükséges) **futtatása,** a többi hardverelem **vezérlése** és egyes, **kevésbé számításigényes adatfeldolgozási feladatok** ellátása.
- **Alkalmazás-specifikus mikroprocesszorok** (*Application-Specific Instruction Set Processor*, ASIP): Tárolt programú mikroprocesszorok, amelyek utasításkészlete és /vagy mikroarchitektúrája egy adott alkalmazás igényei szerint optimalizált.
- **DMA-vezérlő** *(Direct Memory Access)*: Közvetlen, blokkos adatátvitelt biztosít a memória és a slave egységek között a mikroprocesszor tehermentesítése céljából.
- **Memória**. Az on-chip memória általában **hierarchikus, többszintű gyorsítótár** *(cache)*, amelyet a gyors működés szükségessége miatt SRAM vagy beágyazott DRAM *(eDRAM)* cellákkal valósítanak meg. Az operatív- és/vagy háttértárként működő külső memória SRAM- és klasszikus DRAM-alapú egyaránt lehet.

Chipen belüli kommunikáció

- **Rendszerbusz**. Az újrafelhasználhatóság érdekében a belső összeköttetéseket megvalósító **buszrendszerek szabványos megoldások**. A nagy sávszélesség-igényű és a kevésbé sebességkritikus egységek általában két külön buszon keresztül érhetők el. A két buszrendszert ún. híd *(bridge)* modulok kapcsolják össze A legelterjedtebb on-chip rendszerbusz-szabványok az ARM AMBA, a MIPS EC és az IBM CoreConnect.

- **Pont-pont interfészek**. Közvetlen összeköttetések, amelyek időzítéskritikus alkalmazásokban a lehető legnagyobb sávszélességet, vagy kis adatforgalom esetén a közös busz tehermentesítését biztosítják (a 4-2. ábrán szaggatott vonallal jelölve).

Slave egységek

- Általános célú be- és kimeneti portok (*General Purpose I/O*, GPIO).
- Nagysebességű soros kommunikációs csatornák (pl. RocketIO).
- Szabványos kommunikációs interfészek (pl. USB, SPI, I2C, UART, PCI-X stb.) fizikai rétegének és ad-hoc vagy szintén szabványos protokolljainak implementációja.
- Alkalmazás-specifikus funkcionális egységek. Tipikusan nagy számításigényű, nagyfokú párhuzamosságot igénylő algoritmusok hardver implementációi, pl. video codec (JPEG, MPEG), titkosítás (AES, DES).

4.3. Beágyazott rendszerek újrakonfigurálható technológián

A beágyazott rendszerek **egyetlen chipre integrált megvalósításait *System-on-Chip*** (SoC) áramköröknek nevezzük. Az integráltság fokának növelése, a lehető legtöbb komponens egyazon chipen való megvalósítása a **számítási teljesítmény** szempontjából előnyös, hiszen a chipen kívüli kommunikáció sokkal lassabb technológiákat igényel, mint a chipen belüli. Azonban az integráció elsődleges hajtóereje nem a számítási kapacitás, hanem a **megbízhatóság** növelése. Az integrált áramköri technológia lényegesen megbízhatóbb, mint az egyes chipeket körülvevő, az azok közötti összeköttetéseket megvalósító áramköri környezet technológiája (PCB).

A mai modern újrakonfigurálható áramkörök, FPGA-k kapacitása és sebessége lehetővé teszi, hogy egy beágyazott rendszer minden komponensét az általános célú erőforrások (BLE-k) és heterogén architekturális elemek (blokk szorzók, blokk RAM-ok, hard-core mikroprocesszorok, kommunikációs interfészek áramkörei) segítségével valósítsuk meg. Az újrakonfigurálható technológián megvalósított beágyazott processzoros rendszereket ***System on a Programmable Chip*** (SoPC) rendszereknek nevezzük. Ezek fő előnye, hogy a klasszikus SoC rendszerekhez képest, a mikroprocesszor(ok)on futó szoftveren kívül **maga a hardverplatform is rugalmasan módosítható** az esetlegesen változó specifikáció igényeinek megfelelően.

4.3.1. Soft-core mikroprocesszorok

Ahhoz, hogy egy FPGA-n belül egy komplett mikroprocesszoros rendszert hozhassunk létre, nem szükséges, hogy az adott FPGA tartalmazzon hard-core mikroprocesszor magot. A nagyobb FPGA gyártók mindegyike kínál a saját eszközeire optimalizált, ún. soft-core mikroprocesszorokat a hozzá tartozó buszrendszerrel, I/O komponensekkel, hardver gyorsítókkal és az ilyen összetevőkből álló hardverplatform hatékony kifejlesztéséhez szükséges szoftver eszközökkel együtt. A **soft-core** kifejezés azt jelenti, hogy a mikroprocesszor és az egyéb komponensek **fizikai tervei nem állnak rendelkezésre**, az FPGA gyártó azoknak csak **szintetizálható**,

nagymértékben **generikus RTL modelljeit** bocsátja a felhasználó rendelkezésére. A felhasználó feladata az általa szükségesnek vélt architekturális erőforrások kiválasztása és az RTL modellek szintetizálása egy konkrét FPGA eszközre. A két legnagyobb FPGA gyártó, a Xilinx és az Altera soft-core rendszerei a MicroBlaze (Xilinx) és a Nios II (Altera). A laboratórium során Altera eszközt használunk, ezért az alábbi pontban a Nios II-re épülő rendszert mutatjuk be részletesen.

4.3.2. A Nios II mikroprocesszoros rendszer

A Nios II mikroprocesszor

A Nios II az Altera által fejlesztett, a saját FPGA eszközeire optimalizált soft-core mikroprocesszor. **Load-store jellegű RISC utasításkészlete** fix, 32-bites utasításokból áll. Legfontosabb architekturális jellemzői a következők:

- 32-bites adatút
- 32-bites lineáris címtartomány
- a memória-címtartományra leképzett I/O rendszer *(memory-mapped I/O)*
- 32 szintű megszakítási rendszer
- 32x32-bites általános célú regisztertömb
- utasítás- és adatcache (opcionális)
- MMU *(Memory Management Unit)*: az operációs rendszerekben a virtuális tárkezelés hatékony megvalósítátához szükséges hardverelem
- MPU *(Memory Protection Unit)*: ugyancsak az operációs rendszerekben a különböző privilégiumszintek implementálására
- JTAG debug modul (opcionális)

A Nios II mikroprocesszor opcionális hardverelemei a szintézist megelőző konfigurációs lépésben választhatók ki az adott feladat igényeinek megfelelően. Magának a mikroprocesszornak **három, erőforrásigény, számítási teljesítmény és fogyasztás szempontjából lényegesen eltérő változata áll rendelkezésre**:

- ***Nios II / f*** *(fast)*: A Nios II utasításkészlet legnagyobb számítási teljesítményű, ugyanakkor legdrágább és legnagyobb fogyasztású megvalósítása. Legfontosabb hardverelemei:
    - 6 fokozatú pipeline
    - utasításcache (0,5 KB - 64 KB)
    - adatcache (0,5 KB - 64 KB)
    - dinamikus elágazásbecslés
    - hardver szorzó (1 órajelciklus késleltetés)
    - hardver osztó
    - barrel shifter (1 órajelciklus késleltetés)
    - opcionális MMU/MPU
- ***Nios II / s*** *(standard)*: Erőforrásigénye kb. 20%-kal, számítási teljesítménye mintegy 40%-kal kisebb, mint az “f” verzióé. Legfontosabb hardverelemei:
    - 5 fokozatú pipeline

    - utasításcache (0,5 KB - 64 KB)
    - statikus elágazásbecslés
    - hardver szorzó (3 órajelciklus késleltetés)
    - hardver osztó
    - barrel shifter (3 órajelciklus késleltetés)
- ***Nios II / e*** *(economy)*: A Nios II utasításkészlet legolcsóbb megvalósítása. Erőforrásigénye csupán fele az “s” verzióénak. Az adatút nem pipeline szervezésű.

A három verzió pontosan ugyanazt az utasításkészletet valósítja meg, **programozói nézetük** tehát teljesen **azonos**, tehát az egyik verzióra megírt szoftver változtatás nélkül képes futni bármelyik másik verzión.

Memória

A Nios II rendszer lehetővé teszi cache memória implementálását, amely **a mikroprocesszorhoz közel elhelyezkedő, kis méretű ideiglenes tároló a gyakran használt utasítások és adatok számára**. Hozzáférési ideje lényegesen alacsonyabb, mint az operatív memóriáé, mivel fizikailag az FPGA eszköz belső **blokk RAM** moduljai valósítják meg. A cache memória a címzési mód szempontjából **transzparens**, vagyis a futó alkalmazás a fizikai - vagy operációs rendszer esetén a virtuális - memóriabeli címeket használja.

A transzparens cache memória alkalmazásának hátránya, hogy egy konkrét adathoz való **hozzáférés ideje nem állandó, a program végrehajtása során változik**. Ha egy adat nem található meg a cache memóriában, akkor a hozzáférés lényegesen lassabb és a transzparens címzés miatt a futó alkalmazás nem tudhatja előre, hogy mikor melyik adat van benne a cache-ben. Időzítéskritikus alkalmazások esetén ez a bizonytalanság nem mindig engedhető meg, ezért ilyen esetekben ún. **szorosan csatolt memóriát** alkalmazunk, amely fizikailag a cache-hez hasonlóan **blokk RAM**-ban valósul meg, de címzését tekintve **nem transzparens**. A címtartománynak egy a futó alkalmazás által is ismert tartománya van leképezve erre a memóriára, amelynek válaszideje állandó, így **az időzítéskritikus kódrészletek futásideje jól prediktálható**.

I/O szervezés

A Nios II rendszerben a perifériavezérlők a memória-címtartományra vannak leképezve. Ez az ún. **memory-mapped I/O** szervezés azt jelenti, hogy az egyes perifériaelemek ugyanazokkal a

load és store utasításokkal érhetők el, mint a memória[1]. Az egyes perifériák interfésze általában néhány regiszterből áll, amelyek vezérlő- és státuszmezőkből, valamint adatmezőkből állnak. **A perifériára jellemző regiszterhalmaz minden eleme a lineáris címtartomány egy meghatározott címén írható és olvasható**.

Az Avalon rendszerbusz

A Nios II rendszerben a mikroprocesszor és a perifériavezérlő áramkörök egy közös buszon kommunikálnak egymással. Ez a rendszerbusz az Avalon, amely a Nios II processzorhoz hasonlóan az Altera saját fejlesztésű buszrendszere. Az Avalon hat különböző interfészt specifikál, amelyek vezérlési és adatvonalak halmazát és a hozzájuk tartozó időzítési kritériumokat jelentik. Bármely egyedi hardver egység, amely kompatibilis e hat interfészspecifikáció valamelyikével, integrálható a Nios II mikroprocesszoros rendszerbe:

- ***Avalon MM*** *(Memory Mapped)*. Egy memóriacím-alapú master-slave kapcsolatot definiál. A master egység a cím segítségével azonosítja a megszólítandó slave egységet, amelynek aztán adatot küldhet, vagy onnan adatot olvashat.
- ***Avalon-ST*** *(streaming)*. Dedikált, egyirányú adatkapcsolat két egység között. Az *Avalon-ST source* (forrás) folyamatos adatfolyamot küld az *Avalon-ST sink*-nek (nyelő).
- ***Avalon MM tristate***. Az Avalon MM interfész kiterjesztése az FPGA-n kívüli, háromállapotú buszrendszerre kapcsolt komponensek felé.
- ***Avalon clock***. A globális jelek (órajel és reset) továbbításáért felelős interfész. Az *Avalon clock output* interfész órajelet és reset-et generál, míg az *Avalon clock input* interfész fogadja azokat.
- ***Avalon interrupt***. Ez az interfész teremti meg a kapcsolatot a megszakítást generáló *(Avalon interrupt sender)* és azt fogadó *(Avalon interrupt receiver)* egységek között.
- ***Avalon conduit***. Lehetővé teszi az interfészjelek kivezetését a mikroprocesszoros rendszeren kívülre. Ez az az interfész, amely segítségével a mikroprocesszoros rendszer (beleértve a processzoron kívül az összes könyvtári és egyedi perifériavezérlőt) kommunikálhat beágyazó környezetével.

Megszakítások és kivételek kezelése

A megszakítás és kivétel kifejezéseket sokféle értelemben használja a szakirodalom. E jegyzetben **megszakítás** alatt azt a mechanizmust értjük, amikor egy külső forrás egy

[1] Ha a rendszerben van cache memória, akkor gondoskodni kell arról, hogy a perifériákhoz való hozzáférés esetén a cache memóriát megkerülve, közvetlenül a megcímzett periféria regisztereihez férjünk hozzá. Ezért az utasításkészlet külön a perifériákra vonatkozó load és store utasításokkal is rendelkezik, amelyeket azonban a C fordító eltakar a programozó elől. Assembly szinten tehát van különbség az egyszerű memória írás/olvasás és a periféria írás/olvasás között.

megszakítási vonalon (vezetéken) jelzi a processzornak egy **esemény bekövetkeztét**, amelyre a rendszernek reagálnia kell. **Kivétel** alatt azt értjük, amikor a mikroprocesszoron futó **szoftver idéz elő eseményt** (pl. nullával osztás, ismeretlen utasítás beolvasása stb.). A rendszerhívás (a felhasználói szoftver által egy magasabb privilégiumszintű, pl. az operációs rendszer részét képező rutin hívása), vagy más néven szoftver-megszakítás a kivételek közé tartozik.

A Nios II mikroprocesszor 32 külső, szintérzékeny megszakítási vonallal rendelkezik. A megszakítást előidéző esemény azonosítása szempontjából **a megszakítási rendszer lekérdezéses**, vagyis bármely megszakítási vonal aktiválódása esetén a végrehajtás egy **közös megszakításkiszolgáló rutin** (*Interrupt Service Routine*, ISR) első utasítására ugrik. E rutin lekérdezi a megszakítás generálására képes perifériaelemek megfelelő flag-jeit, ami alapján eldöntheti, hogy melyik periféria kért megszakítást. Azután a végrehajtás az immár azonosított megszakítási forrás kiszolgáló rutinjának adódik át.

A JTAG debug modul

A debug modul feladata, hogy **monitorozza a mikroprocesszor belső erőforrásainak állapotát és kapcsolatot teremtsen egy host számítógéppel**, amelyen a fejlesztés és hibakeresés folyik. A Nios II rendszerben a debug modul az FPGA eszköz **JTAG interfészét** használja e kommunikációs csatorna megvalósítátára. Az összes többi hardverelemhez hasonlóan a debug modul is opcionális és konfigurálható. Alkalmas a programtár feltöltésére, töréspontok elhelyezésére a programkódban, a regiszterek és az adatmemória tartalmának monitorozására és a programvégrehajtással kapcsolatos egyéb adatok gyűjtésére.

4.3.3. A fejlesztés folyamata

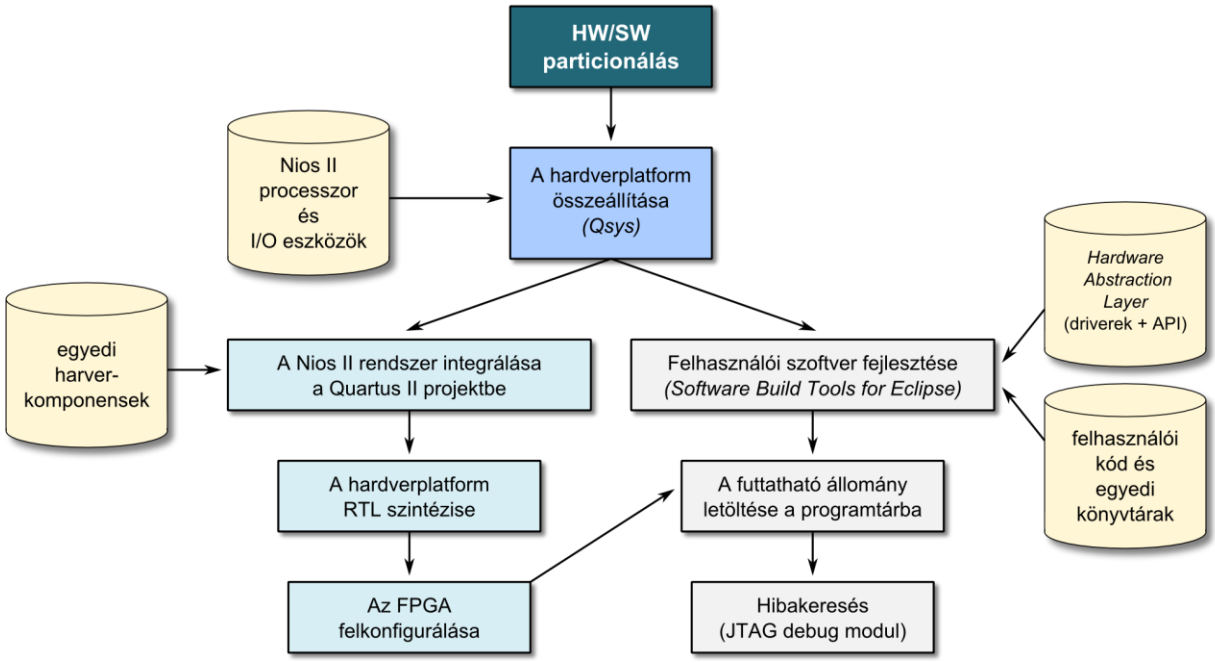
A Nios II beágyazott processzoros rendszerek fejlesztése az alábbi részfeladatokat foglalja magában:

- ***Hardver/szoftver particionálás***. Egy beágyazott rendszer sokféle eltérő feladatot végez párhuzamosan. A specifikációban lefektetett, főként az erőforrásigényre és a számítási teljesítményre vonatkozó követelmények alapján eldöntendő, hogy melyik részfunkciókat végzi a központi mikroprocesszor (szoftver) és melyik az, amelyik a szigorú teljesítménykritériumok miatt hardveres megvalósítást igényel.
- ***Hardverplatform fejlesztése***. A hardver/szoftver particionálás után kerül sor a hardverkörnyezet kifejlesztésére, amelyhez az FPGA gyártók egyedi, integrált fejlesztői környezetet kínálnak. Az Altera Nios II rendszere esetén ez a fejlesztői környezet a Qsys, amely a *Szintézis FPGA technológiára* c. laboratóriumi gyakorlat során megismert Quartus II környezetbe épül be. A hardverplatform kifejlesztése magában foglalja a Nios II mikroprocesszor megfelelő verziójának, valamint a szükséges perifériaelemeknek a kiválasztását és felkonfigurálását, a szintetizálható HDL modellek generálását, beleértve az egyes komponensek összekapcsolását megvalósító Avalon rendszerbuszt is. A hardverplatform generálása során állnak elő a felhasznált perifériaelemekhez tartozó

szoftver könyvtárak (*Hardware Abstraction Layer*, HAL) és fejállományok is, amelyek az adott periféria kezeléséhez szükséges alacsony szintű drivereket és magasabb szintű API *(Application Programming Interface)* függvényeket tartalmazzák.

- ***Szoftveralkalmazás fejlesztése***. A hardverplatform leíró fájljainak generálása után - az RTL szintézissel párhuzamosan - elkezdődhet a szoftver alkalmazás fejlesztése, amelyhez ugyancsak egyedi, a Quartus II-be beépülő, integrált fejlesztői környezet áll rendelkezésre *(Nios II Software Build Tools for Eclipse)*.
- ***Funkcionális verifikáció***. Az FPGA felkonfigurálása után az integrált szoftverfejlesztői környezetből a JTAG debug modulon keresztül a szoftver alkalmazás a programtárba tölthető és teljeskörű hibakeresés végezhető.

A Nios II rendszer fejlesztésének folyamatát a 4-3. ábra szemlélteti.



4-3. ábra Nios II design flow

4.4. Ajánlott irodalom

[1] Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Naraig Manjikian, **Computer Organization and Embedded Systems**, 6th Edition, McGraw-Hill Companies, Inc., 2012

[2] Pong P. Chu, **Embedded SoPC Design with Nios II Processor and VHDL Examples**, John Wiley and Sons, Inc., 2011

[3] Michael Keating, Pierre Bricaud, **Reuse Methodology Manual for System-on-Chip Designs**, 3rd Edition, Kluwer Academic Publishers, 2002