BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

FPGA alapú rendszerek fejlesztése

8. előadás

Raikovich Tamás

Tematika

A tárgy hátralevő részének fő témája az FPGA alapú (szoft) processzoros rendszerek

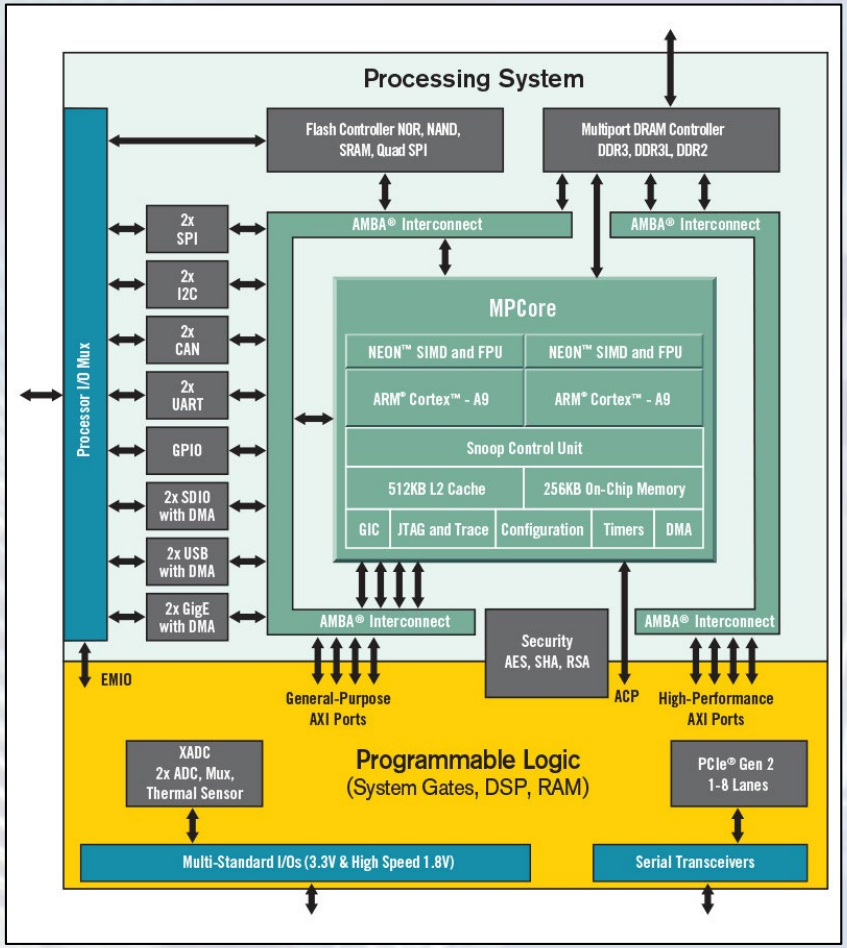
- **Előadás:** alapfogalmak, MicroBlaze CPU, standalone szoftverfejlesztés
- **Gyakorlat:** MicroBlaze alaprendszer, egyszerű szoftver
- **Előadás:** áramkörön belüli buszrendszerek (AMBA AXI), megszakítás- és kivételkezelés
- **Gyakorlat:** saját periféria illesztése
- **Előadás:** PCI és PCI Express busz, külső memóriák, DMA
- **Gyakorlat:** megszakításkezelés, DMA
- **Előadás:** FPGA-k konfigurációja, HW-SW debug

Tartalom

- ***A processzoros rendszerekkel kapcsolatos fontosabb alapfogalmak***
- **Xilinx MicroBlaze processzor**
- **Standalone szoftverfejlesztés**

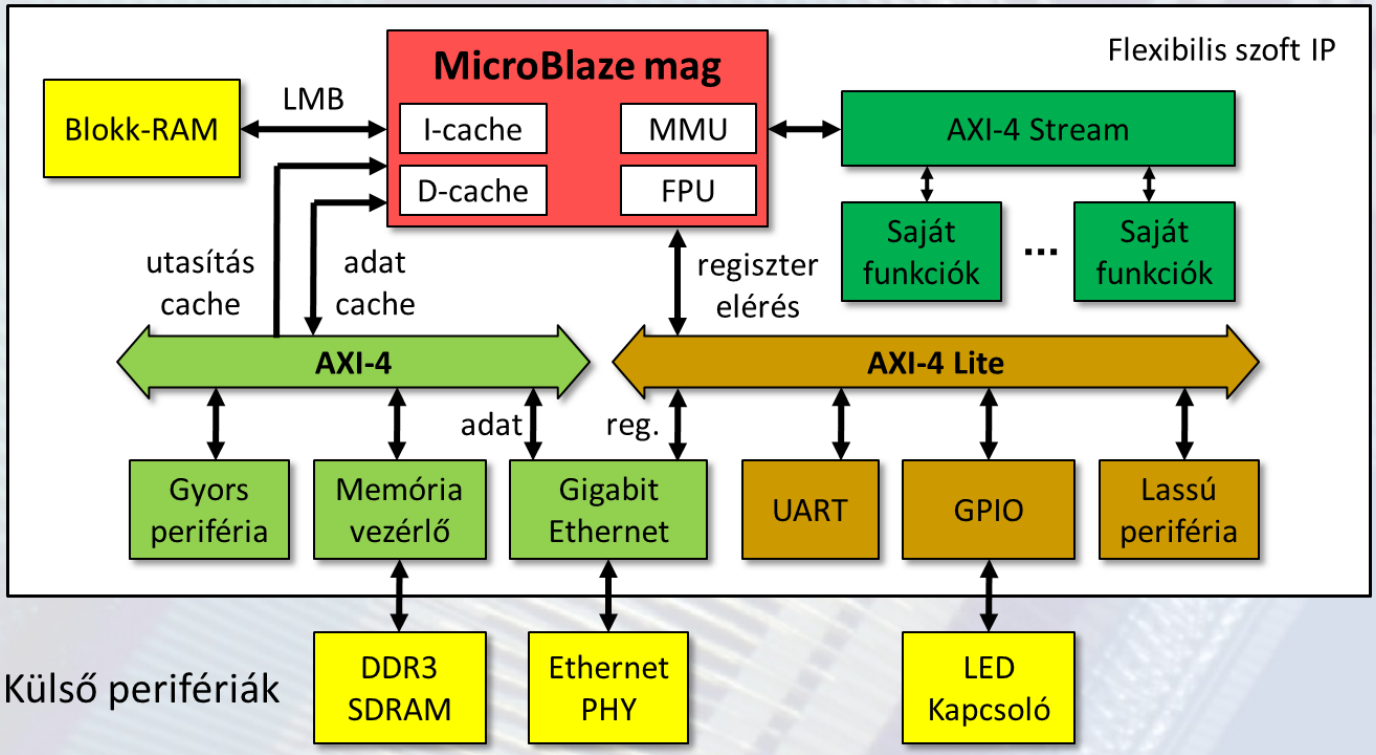
Hard processzoros rendszer

- **A chip-en van megvalósítva**
  - Fix funkció, nem flexibilis
  - Gyors
- **Példa: Xilinx Zynq-7000 SoC**
  - Processzoros rendszer
    - 2 ARM Cortex-A9 CPU
    - Perifériák
  - FPGA
- **Részletek a Heterogén SoC rendszerek tárgyban**



Szoft processzoros rendszer

- **FPGA erőforrásokból épül fel**
  - Nagymértékben konfigurálható, flexibilis, lassabb
- **Példa: Xilinx MicroBlaze processzor (ezzel foglalkozunk)**



CISC processzorok

- **Complex Instruction Set Computer (↔ RISC)**
- **Ez a típus volt előbb**
- **Egy utasítás több alacsonyszintű műveletből állhat**
  - Pl. memóriában tárolt érték megnövelése:
    - Memória olvasás, aritmetikai művelet, memória írás
  - Változó órajelciklus igény
  - Magasszintű nyelvi elemek közvetlen támogatása
- **Sokféle és összetett címzési mód**
  - Pl.: regiszter + (regiszter * skála) + offszet indirekt
- **Nagyszámú és változó méretű utasításkészlet**
- **Kevés belső regiszter (akkumulátor alapú)**
- **Neumann architektúra: egyetlen közös memória interfész**
- **Mikroprogramozott vezérlő (mikrokód)**
- **Példák: MOS 6502, Intel 8051, Intel x86, Zilog Z80**
  - A mai x86 processzorok belül már RISC felépítésűek!

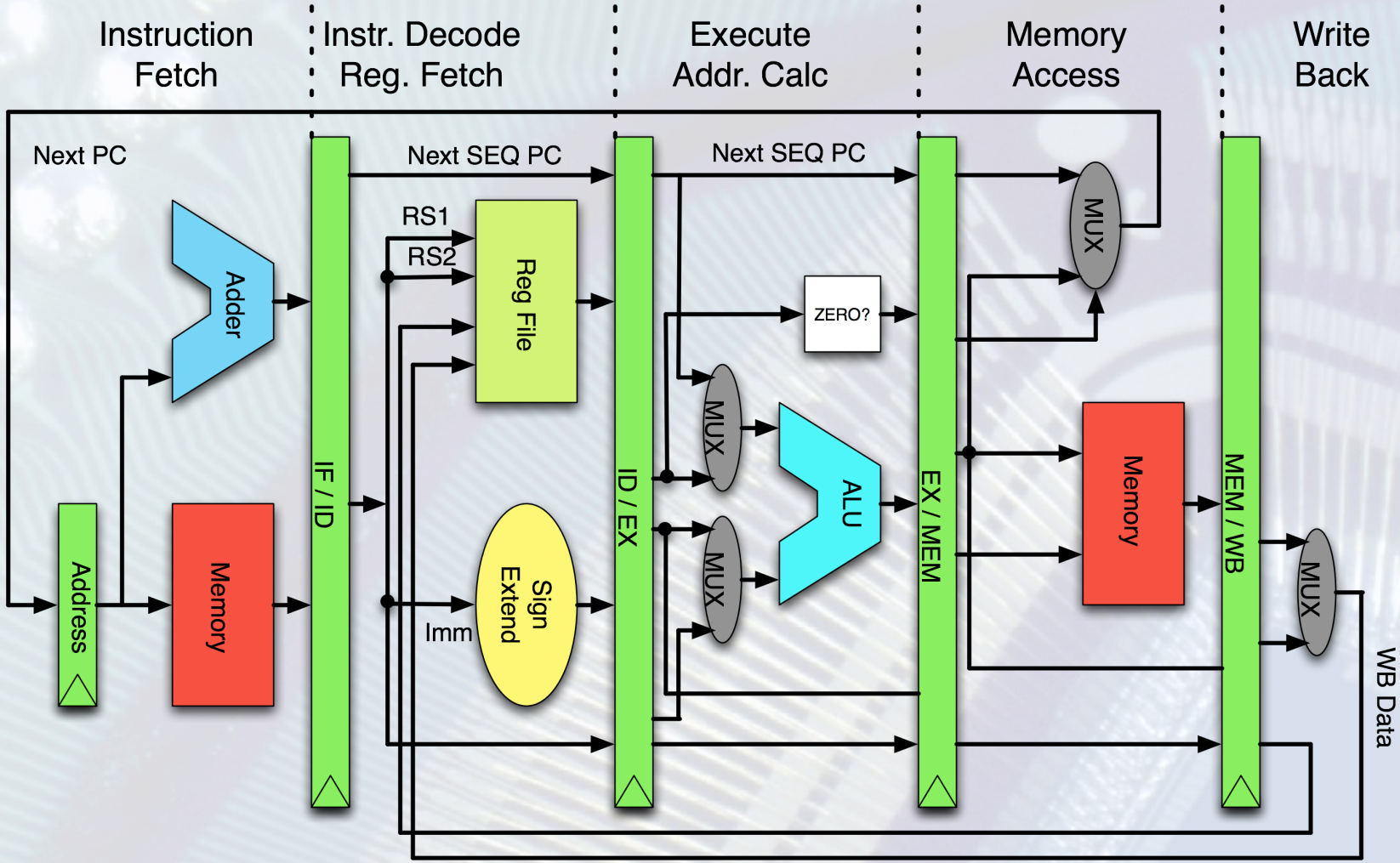
RISC processzorok

- **Reduced Instruction Set Computer (↔ CISC)**
- **A CISC utasítások helyettesítése egyszerűbb műveletekkel**
  - 1 órajel alatt végrehajthatók
  - Pipeline felépítés és utasítás végrehajtás
- **Kevés címzési mód, ortogonális utasításkészlet**
  - Az utasításoknál használható az összes címzési mód
- **Fix méretű utasításkészlet**
- **Load-store architektúra**
  - Műveletvégzés csak belső regisztereken
  - CISC: memóriában tárolt érték növelése → RISC: 3 utasítás
- **Sok belső regiszter**
- **Harvard architektúra: külön program- és adatmem. interfész**
- **Példák: ARM, RISC-V, IBM PowerPC, MIPS, Xilinx MicroBlaze**

Pipeline utasítás végrehajtás

- **Az utasítások végrehajtása során az egyes lépésekhez egy-egy pipeline fokozat tartozik**
  - Jó esetben minden órajelciklusban elkezdhető egy újabb utasítás feldolgozása
- **Pl. 3 fokozatú pipeline**
  - Utasítás elővétel (IF, fetch)
    - Programmemória olvasás
    - Beírás az utasítás regiszterbe
    - A programszámláló növelése
  - Utasítás dekódolás (DC, decode)
    - Az elvégzendő művelet meghatározása
    - A műveleti operandusok meghatározása, beolvasása
  - Utasítás végrehajtás (EX, execute)
    - A művelet végrehajtása és az eredmény visszaírása

Pipeline utasítás végrehajtás



Pipeline utasítás végrehajtás

- **Problémák – külső memória hozzáférés**
  - Az adat rendelkezésre állásáig a pipeline leáll
  - Gyorsítótár (cache) alkalmazása
- **Problémák – adat versenyhelyzet**
  - Az előző eredmény még nincs visszaírva, de operandusa a következő utasításnak
  - Megoldás
    - Az utasítások átrendezése (assembler)
    - A végrehajtás felfüggesztése, amíg az eredmény nem érhető el (kihasználatlan órajel ciklusok)
    - Adat előrecsatolás (egy pipeline fokozat átugrása)

Pipeline utasítás végrehajtás

- **Problémák – ugrás**
  - A PC új értéket kap, emiatt a már lehívott utasítások érvénytelenek lesznek
  - Megoldás
    - Pipeline kiürítése: kihasználatlan órajel ciklusok
    - Delay slot: a már lehívott utasítás(ok) végrehajtása, programszervezés kérdése (a fordító feladata)
    - Ugrásbecslés: a CPU megpróbálja megjósolni az ugrási címet, helyes becslés esetén nincs veszteség
- **Problémák – megszakítás, kivétel**
  - Nem tudni, hogy mikor történik
  - Pipeline kiürítése az egyetlen lehetőség

Gyorsítótár (cache)

- **Miért kell?**
  - DRAM hozzáférés tulajdonságai
  - Kis késleltetés
- **Cache-ben**
  - Tárolt memória cím (TAG): cache találat megállapítás
  - Adat: cache line méretű blokkok
- **Memória írás**
  - Write-through: minden cache írás hatására frissül a RAM-ban az adat
  - Write-back: adat kiírás a cache frissítése esetén
- **Cache koherencia**
  - Több CPU mag, DMA képes HW (cache invalidálás)

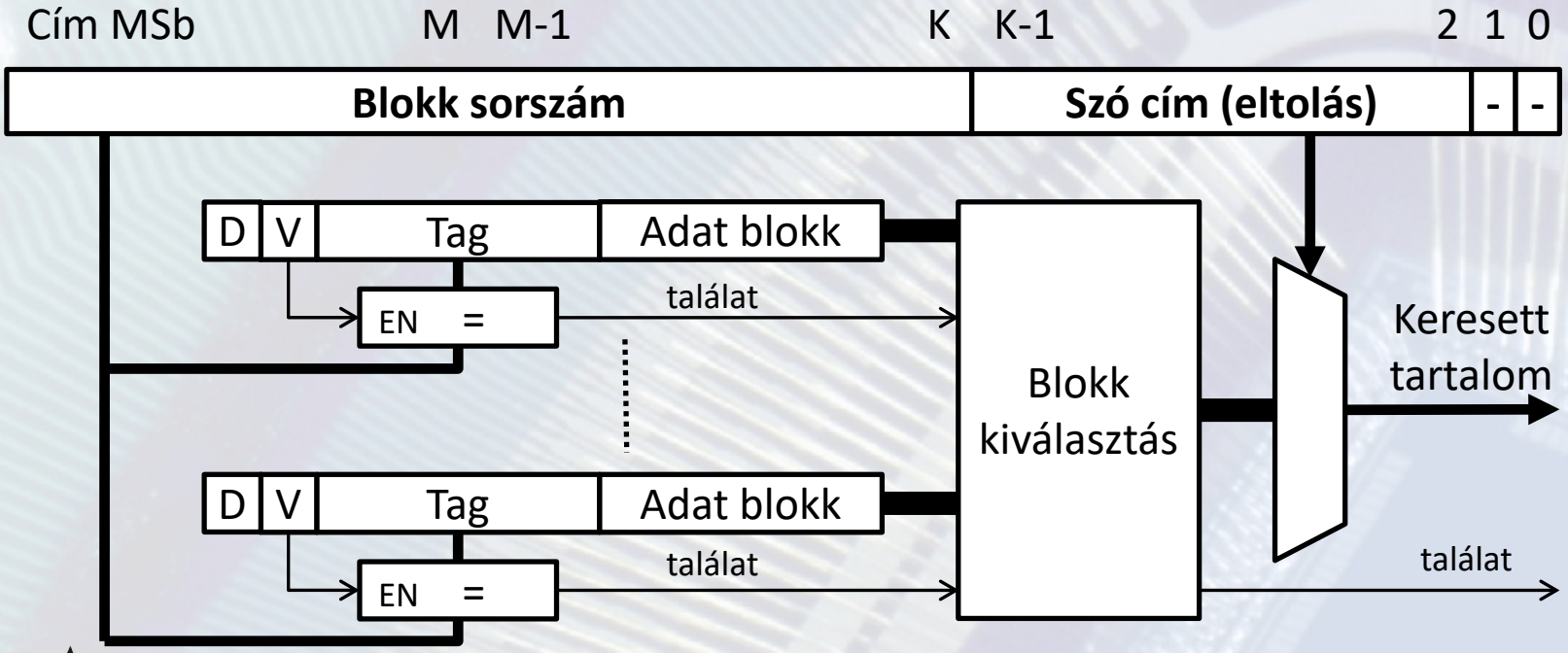
Gyorsítótár (cache)

- **Az adatok tárolása 2^K byte méretű blokkokban (cache line) történik, a cache mérete 2^M byte (M>K)**
  - A cím [K-1:0] bitjei adják a blokkon belüli eltolást
  - A cím felső bitjei adják a blokk sorszámát
- **Minden blokkban az adatok mellett tárolni kell még**
  - Tag: a memória hányadik blokkja tartozik a tárolt adathoz (a cím felső bitjei)
  - Valid bit: a blokk érvényességének jelzése
  - Dirty bit: történt-e írás erre a blokkra

Gyorsítótár (cache)

Teljesen asszociatív gyorsítótár (az egyik véglet)

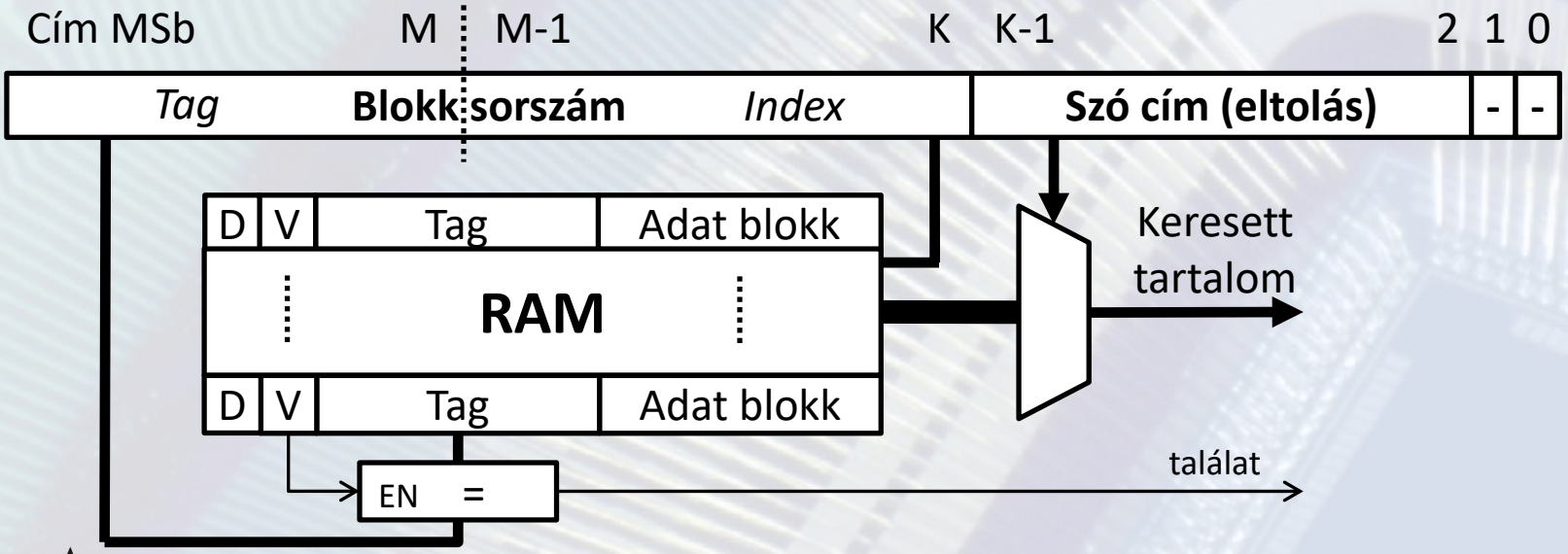
- **A memória blokkok bárhová elhelyezhetők a cache-ben**
- **Tartalom szerint címezhető memória (CAM)**
- **Nagy erőforrás- és energiaigény: sok széles komparátor**



Gyorsítótár (cache)

Direkt leképzésű gyorsítótár (a másik egyik véglet)

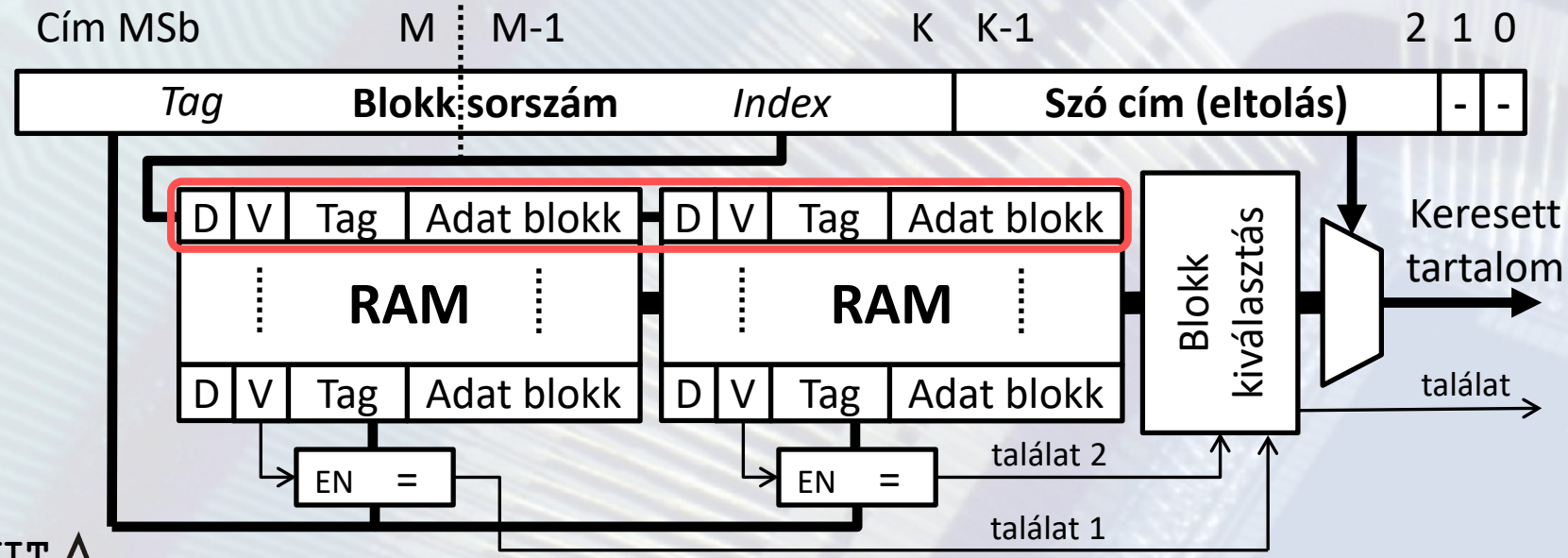
- **A memória blokkok csak egy helyre kerülhetnek**
  - Pl. a blokk sorszám alsó bitjei (index) döntik ezt el
- **Jóval kisebb erőforrásigény: memória és 1 komparátor**
- **Hátrány: gyakori cache miss nem megfelelő program vagy adatpuffer szervezésnél a működés jellege miatt**



Gyorsítótár (cache)

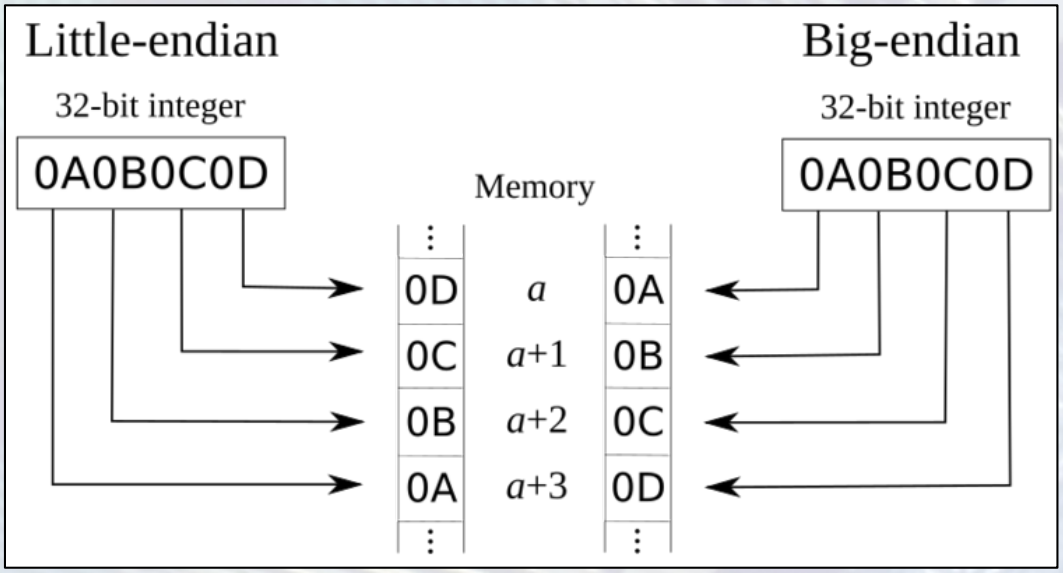
N-utas asszociatív gyorsítótár (a kettő között)

- **A memória blokkok N helyre kerülhetnek (tipikusan N=2-16)**
  - Az index bitek egy N elemű halmazt jelölnek ki
- **Erőforrásigény: N db memória és N db komparátor**
- **Ritkább cache miss, mérsékelt komplexitás és fogyasztás**



Bájtsorrend (endianness)

- **Little-endian bájtsorrend**
  - A kisebb helyiértékű bájt a kisebb címen található
- **Big-endian bájtsorrend**
  - A kisebb helyiértékű bájt a nagyobb címen található
- **Bi-endianness**
  - Változtatható bájtsorrend, mindkettő támogatott



Tartalom

- A processzoros rendszerekkel kapcsolatos fontosabb alapfogalmak
- *Xilinx MicroBlaze processzor*
- Standalone szoftverfejlesztés

A MicroBlaze processzor

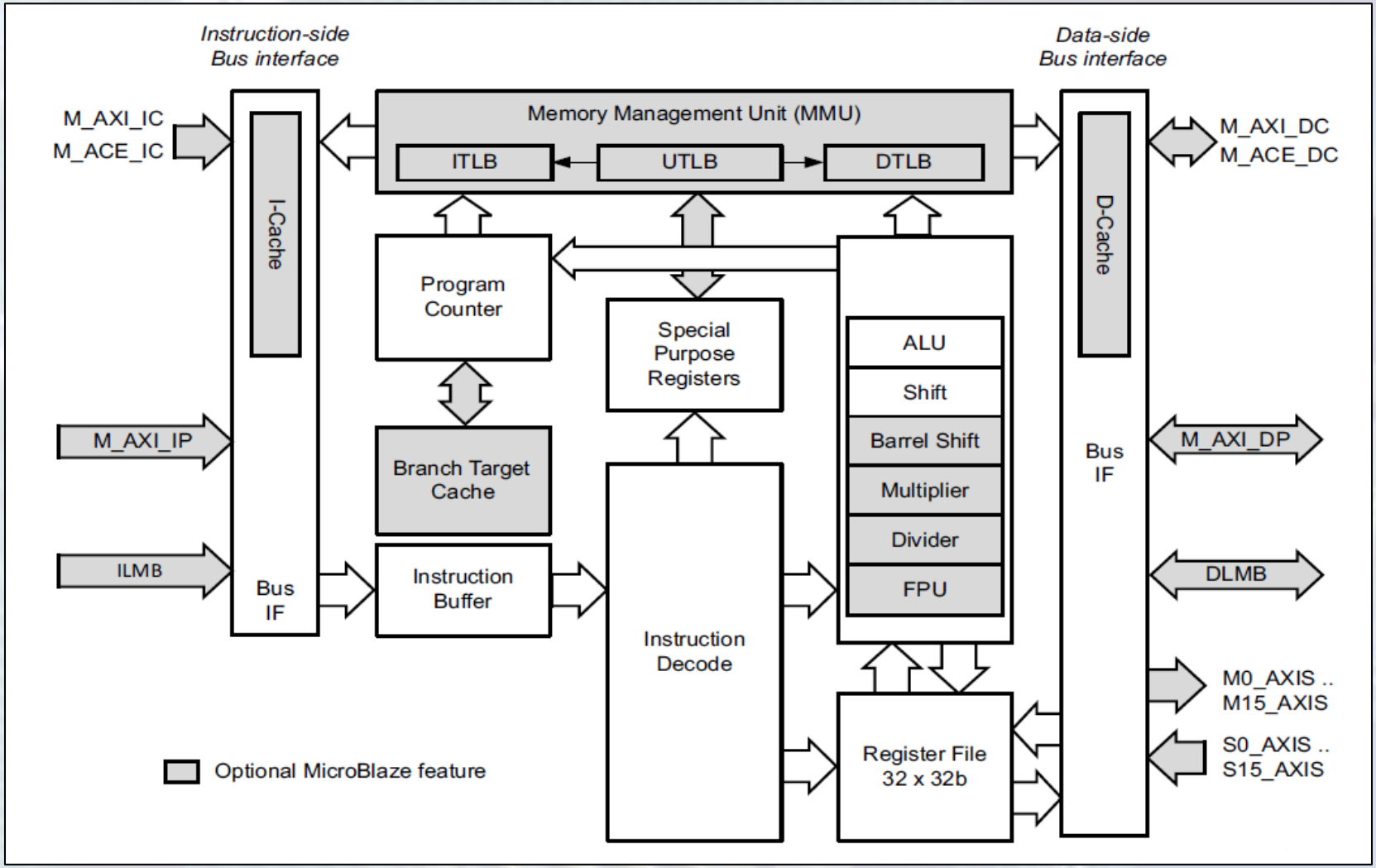
- **32/64 bites általános célú RISC processzor**
  - Alkalmas önálló feladatok végrehajtására
  - Kiegészítő egység a nagyteljesítményű ARM SoC-s rendszerekben → egységes interfész felület: AXI
- **Lágy processzor mag**
  - Az FPGA programozható erőforrásaiból épül fel
- **Rugalmas rendszerkialakíthatóság**
  - Minimális verzió lokális belső memóriával
  - Nagy rendszer külső memóriával, sok perifériával
- **MicroBlaze Processor Reference Guide (UG984)**

A MicroBlaze processzor

A MicroBlaze processzor jellemzői

- **Harvard architektúra**
  - Szétválasztott utasítás és adat oldali busz interfész
- **32-64 bites címbusz, 32/64 bites adatbusz**
- **32 általános célú regiszter**
- **Ortogonális utasításkészlet**
  - 32 bites utasítások
  - 3 operandus, 2 címzési mód (abszolút, regiszter indirekt)
- **Egyszeres kibocsátású pipeline**
  - Kisebb erőforrás igényű változat: 3 fokozatú pipeline
  - Nagyteljesítményű változat (1): 5 fokozatú pipeline
  - Nagyteljesítményű változat (2): 8 fokozatú pipeline
- **Nagymértékben konfigurálható**

A MicroBlaze processzor



A MicroBlaze processzor

A MicroBlaze főbb konfigurációs beállításai

- **Utasítás és adat oldali busz interfészek**
  - AXI-4, AXI-4 Lite, AXI-4 Stream, Local Memory Bus (LMB)
- **Cím méret (32-64 bit), adatméret (32 vagy 64 bit)**
- **Utasítás és adat cache**
- **MMU (Linux operációs rendszer futtatásához)**
- **Opcionális utasítások**
  - Egész szorzás (hardveres szorzó) és osztás (hardveres osztó)
  - Barrel shifter, pattern compare
  - Lebegőpontos műveletek támogatása (FPU)
- **Hardveres debug modul**
  - Debug képességek, HW utasítás és adat töréspontok száma
- **Hardveres kivételek támogatása**
- **Pipeline fokozatok száma**

A MicroBlaze processzor

A MicroBlaze főbb konfigurációs beállításai

- **Lebegőpontos egység (FPU)**
  - IEEE 754 egyszeres pontosságú formátum
  - Műveletek: összeadás, kivonás, szorzás, osztás, összehasonlítás, konverzió, négyzetgyök
- **Utasítás és adat cache**
  - Gyorsítótárak mérete
  - Cacheline mérete
  - Gyorsítótárazott memória címtartomány
  - Írás engedélyezése
    - Sorok érvénytelenítése, kiürítése, törlése
  - Adat cache esetén write-through és write-back mód
    - Write-through: írás esetén mindig van memória írás is (lassabb)

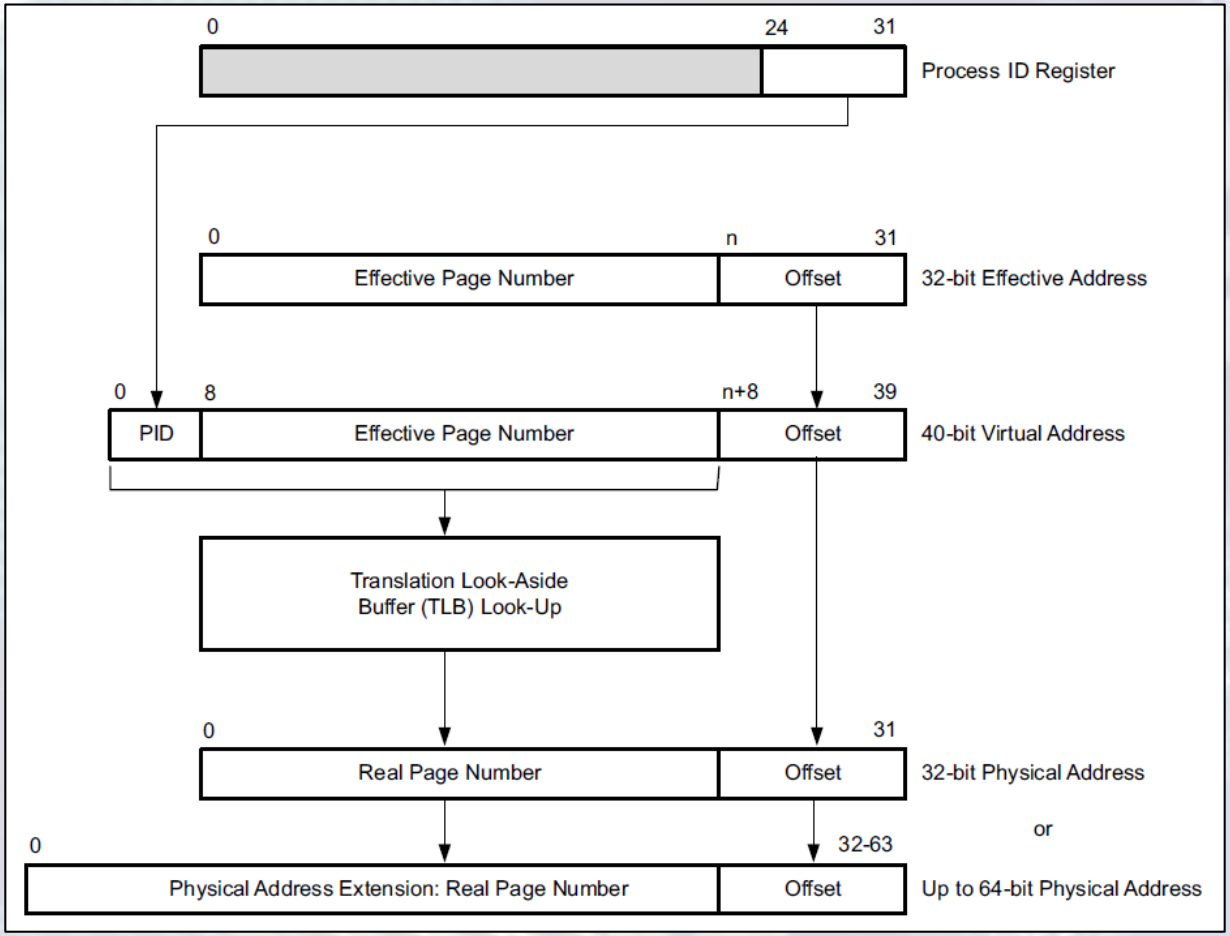
A MicroBlaze processzor

A MicroBlaze főbb konfigurációs beállításai

- **Memory Management Unit (MMU)**
  - A PowerPC 405 processzor MMU-ján alapul
  - Valós mód: a virtuális (effektív) cím közvetlen használata a memória elérésnél
  - Virtuális mód: a virtuális (effektív) címről címfordítás történik a fizikai címre az MMU által
    - Lap méretek: 1 kB, 4 kB, 16 kB, 64 kB, 256 kB, 1 MB, 4 MB, 16 MB
    - A lapok áthelyezhetőek a teljes fizikai címtartományon belül
    - Az inaktív lapok kivehetőek a fizikai memóriából
    - A lapok védelme a nem engedélyezett hozzáféréstől
    - Privilegizált utasítások
    - Opcionális fizikai cím kiterjesztés maximum 64 bitig (PAE)
    - Szoftveres Translation Look-Aside Buffer (TLB) kezelés
  - Multitasking támogatás biztosítása

A MicroBlaze processzor

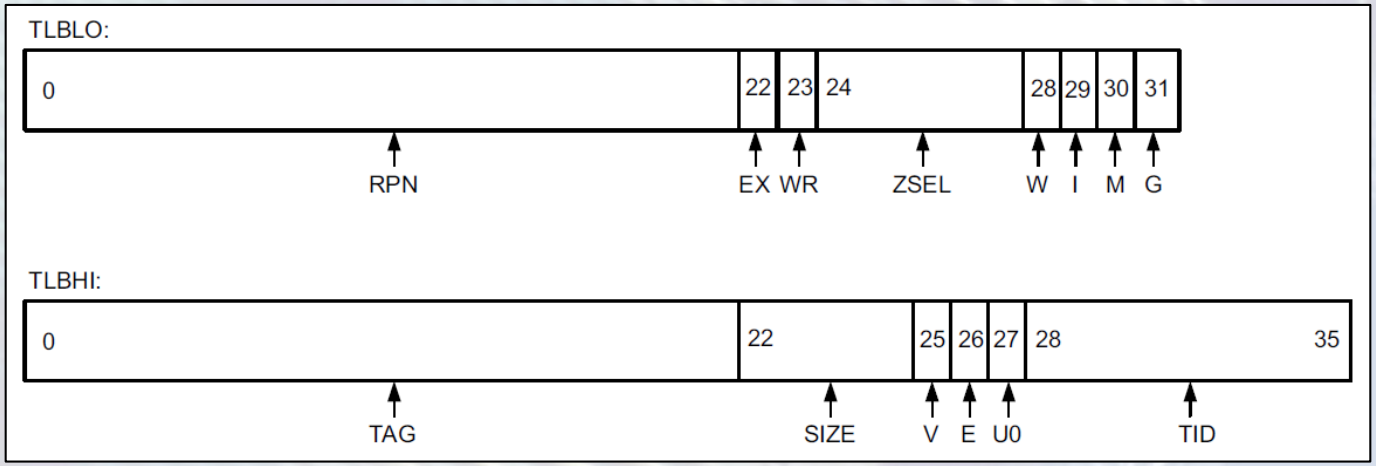
Memory Management Unit (MMU) címfordítás



A MicroBlaze processzor

MMU TLB bejegyzés

- **Virtuális lap azonosítás: TAG, SIZE, V, TID**
  - A cím átfordítás során a virtuális lap számmal kerül összehasonlításra
- **Fizikai lap azonosítás: RPN, SIZE**
  - Az átfordított lapot azonosítja a fizikai memóriában
- **Elérés és hozzáférés vezérlés: EX, WR, ZSEL**
- **Tárolási jelzések: W, I, M,G, E, U0**
  - Gyorsítótárazás engedélyezése és annak jellege, bájtsorrend, stb.



A MicroBlaze processzor

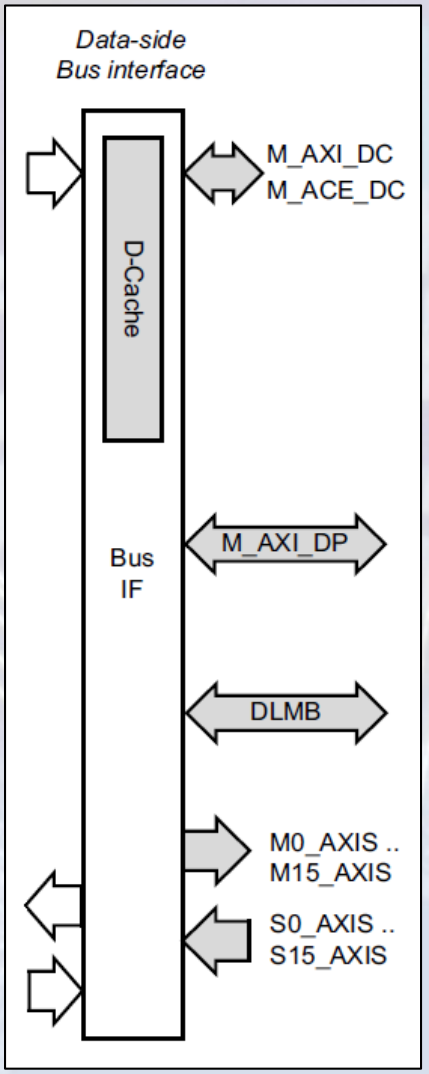
Az ARM Cortex (hard) és a Xilinx MicroBlaze (szoft) processzorok az ARM AMBA szabványból az AXI buszrendszert használják

- **AXI-4 (illetve AXI-3)**
  - Memóriába leképzett
  - Nagysebességű, burst-ös adatátvitelt biztosít
- **AXI-4 Lite**
  - Memóriába leképzett
  - Lassabb, nincs burst-ös adatátvitel
- **AXI-4 Stream**
  - Csak adatok átvitele burst-ös módon
- **A részletekről külön előadás lesz**

A MicroBlaze processzor

A MicroBlaze processzor busz interfészei

- **Külön utasítás és adat oldali busz interfészek**
- **AXI-4 (M_AXI_IC, M_AXI_DC)**
  - Az utasítás és adat cache számára biztosít nagysebességű, burst-ös memória hozzáférést
- **AXI-4 Lite (M_AXI_IP, M_AXI_DP)**
  - Periféria regiszterek elérése
  - Az utasítás oldalon is engedélyezhető
- **Local Memory Bus (LMB)**
  - Egyszerű szinkron busz
  - Elsősorban a belső Blokk-RAM-ok illesztéséhez
  - Periféria illesztési lehetőség
- **AXI-4 Stream (Mn_AXIS, Sn_AXIS, n=0-15)**
  - FIFO alapú interfész saját funkciók illesztéséhez
  - Processzorok közötti kommunikáció



A MicroBlaze processzor

A MicroBlaze processzor adattípusai

- **Bájt (8 bit), félszó (16 bit) és szó (32 bit) adattípusok**
- **Little-Endian: LSB a kisebb, MSB a nagyobb címen**
  - Ez az alapértelmezett
  - Bit indexelés: az LSb sorszáma 0 (az LMB busz és a CPU regiszterek esetén fordított)
- **Big-Endian: MSB a kisebb, LSB a nagyobb címen**
  - Konfigurációs paraméterrel engedélyezhető
  - Bit indexelés: az MSb sorszáma 0
- **Opcionális fordított memória elérési utasítások**

A MicroBlaze processzor

Bájt kezelés írásnál: a CPU a cím alsó két bitjének megfelelő bájt(ok)ra helyezi az adatot

- Példa: little-endian bájtsorrend (írás az rD[0:31]-ből)

Address [LSB-1:LSB]	Byte_Enable [0:3]	Transfer Size	*MSB* Write Data Bus Bytes			*LSB*
			Byte3	Byte2	Byte1	Byte0
11	1000	byte	rD[24:31]			
10	0100	byte		rD[24:31]		
01	0010	byte			rD[24:31]	
00	0001	byte				rD[24:31]
10	1100	halfword	rD[16:23]	rD[24:31]		
00	0011	halfword			rD[16:23]	rD[24:31]
00	1111	word	rD[0:7]	rD[8:15]	rD[16:23]	rD[24:31]

A MicroBlaze processzor

Bájt kezelés olvasásnál: a CPU jobbra rendez

- Példa: little-endian bájtsorrend (írás az rD[0:31]-be)
  Adatbusz: Byte3 (MSB) | Byte2 | Byte1 | Byte0 (LSB)
  Bájt cím: 11 10 01 00

Address [LSB-1:LSB]	Byte_Enable [0:3]	Transfer Size	*MSB* Register rD Data *LSB*			
			rD[0:7]	rD[8:15]	rD[16:23]	rD[24:31]
11	1000	byte				Byte3
10	0100	byte				Byte2
01	0010	byte				Byte1
00	0001	byte				Byte0
10	1100	halfword			Byte3	Byte2
00	0011	halfword			Byte1	Byte0
00	1111	word	Byte3	Byte2	Byte1	Byte0

A MicroBlaze processzor

A szokásos pipeline kérdések

- **Elágazások a programban**
  - Ugrási utasítások, szubrutinhívás
  - Megszakításkezelés, kivételkezelés
- **Következmény**
  - A lehívott és dekódolt utasítást nem kell végrehajtani
  - Pipeline kiürítés és újratöltés: ciklusok vesznek el
- **A hatékonyság javítása: delay slot**
  - A processzor végrehajtja a már lehívott utasításokat
  - Az ugrás csak ezután következik be
  - Igazából programszervezés kérdése
- **A hatékonyság javítása: ugrásbecslés (Branch Target Cache)**
  - Helyes becslés esetén nincs pipeline kiürítés

A MicroBlaze processzor

Regiszterkészlet

- 32 darab 32/64 bites általános célú regiszter
- Speciális regiszterek
- Hardveres használati megkötések
- C/C++ programok esetén használati konvenciók

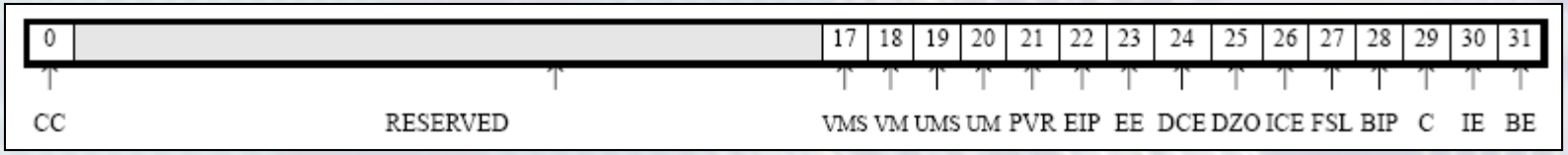
A MicroBlaze processzor

Register	Type	Enforcement	Purpose
R0	Dedicated	HW	Value 0
R1	Dedicated	SW	Stack Pointer
R2	Dedicated	SW	Read-only small data area anchor
R3-R4	Volatile	SW	Return Values/Temporaries
R5-R10	Volatile	SW	Passing parameters/Temporaries
R11-R12	Volatile	SW	Temporaries
R13	Dedicated	SW	Read-write small data area anchor
R14	Dedicated	HW	Return address for Interrupt
R15	Dedicated	SW	Return address for Sub-routine
R16	Dedicated	HW	Return address for Trap (Debugger)
R17	Dedicated	HW/SW	Return address for Exceptions HW, if configured to support hardware exceptions, else SW
R18	Dedicated	SW	Reserved for Assembler/Compiler Temporaries
R19	Non-volatile	SW	Must be saved across function calls. Callee-save
R20	Dedicated or Non-volatile	SW	Reserved for storing a pointer to the global offset table (GOT) in position independent code (PIC). Non-volatile in non-PIC code. Must be saved across function calls. Callee-save.
R21-R31	Non-volatile	SW	Must be saved across function calls. Callee-save.

A MicroBlaze processzor

Regiszterkészlet: speciális regiszterek

- Írásukhoz és olvasásukhoz külön utasítások
- RPC: programszámláló (32 bites, csak olvasható)
- MSR: státusz regiszter



- Regiszterek a kivételkezeléshez: EAR, ESR, BTR, EDR
- FSR: lebegőpontos státusz regiszter
- MMU regiszterek: PID, ZPR, TLBLO, TLBHI, TLBX, TLBSX
- PVR: processzor verzió regiszter (PVR0 – PVR10)

A MicroBlaze processzor

MicroBlaze utasításkészlet

- **Viszonylag egyszerű utasításrendszer**
- **RISC processzor → műveletek csak regiszterben tárolt adatokon**
- **„A” típusú utasítások:**
  - Műveleti kód + cél regiszter + 2 forrás regiszter

Opcode	Destination Reg	Source Reg A	Source Reg B	0	0	0	0	0	0	0	0	0	0	0
0	6	11	16	21										31

- **„B” típusú utasítások:**
  - Műveleti kód + cél regiszter + 1 forrás regiszter + 16 bites konstans

Opcode	Destination Reg	Source Reg A	Immediate Value
0	6	11	16 31

  - IMM utasítás: 32 bites konstansok, a felső 16 bitet tárolja
  - (2x) IMML utasítás: 48 (64) bites konstansok, felső 24 (48) bitet tárolja
  - Az IMM/IMML és az azt követő utasítás atomi, nem szakíthatók meg

Tartalom

- A processzoros rendszerekkel kapcsolatos fontosabb alapfogalmak
- Xilinx MicroBlaze processzor
- ***Standalone szoftverfejlesztés***

Asztali vs. beágyazott SW fejlesztés

Asztali rendszerek

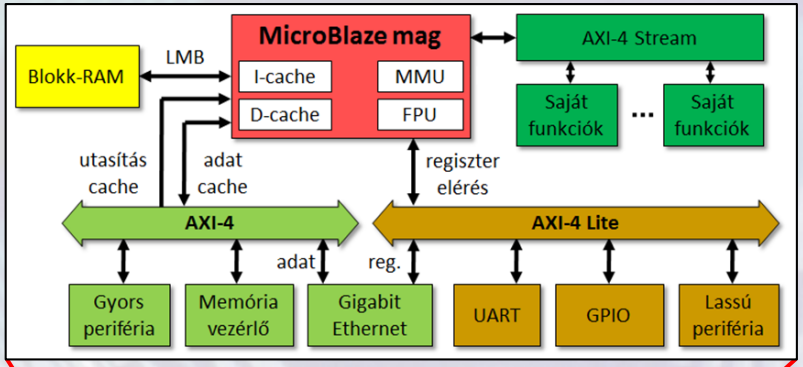
- **Fejlesztés, hibakeresés és futtatás ugyanazon a gépen**
- **Az OS akkor tölti be a programot a memóriába, ha a felhasználó ezt kéri**
- **Címek feloldása**
  - Az alkalmazás betöltésekor
  - A betöltő az OS része

Beágyazott rendszerek

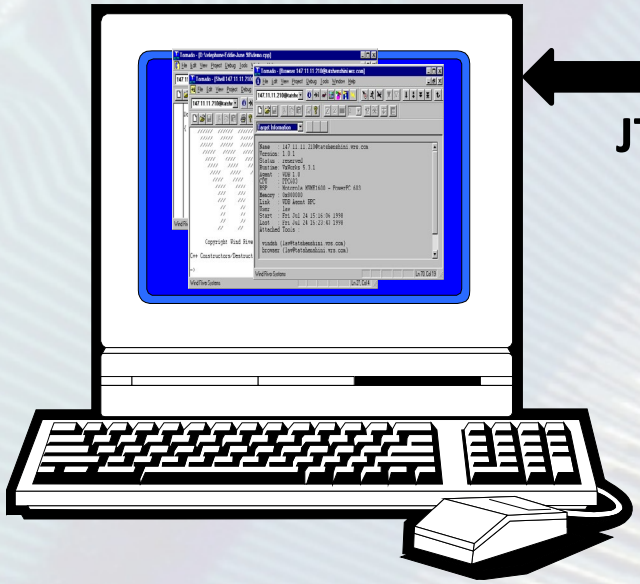
- **Fejlesztés: a host gépen**
- **Futtatás: a célrendszeren**
- **Egy futtatható állomány**
  - ELF fájl
  - Bootloader, alkalmazás, ISR, (operációs rendszer)
  - Címek feloldása linkeléskor
- **Futtatható kód letöltése a célrendszerre**
  - JTAG, Ethernet, soros port
  - Flash programozó

Asztali vs. beágyazott SW fejlesztés

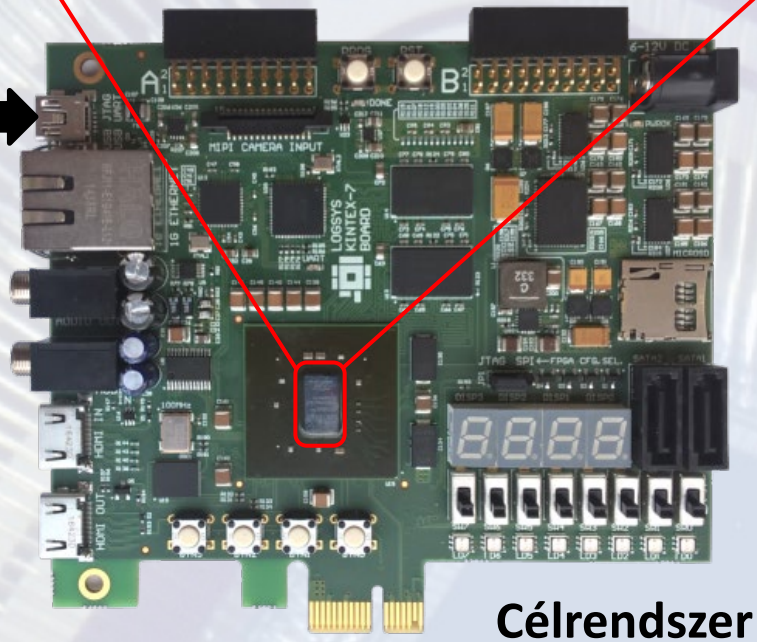
- A fejlesztés külön számítógépen (host) történik, az alkalmazást le kell tölteni a célrendszerre
- A keresztfordító a host gépen fut



Host számítógép



JTAG, UART



Célrendszer

Asztali vs. beágyazott SW fejlesztés

Különféle problémák

- **Minden terv esetén egyedi a hardver**
- **Megbízhatóság**
- **Valósidejű válasz megkövetelése**
  - RTOS ↔ normál OS
- **Kis méretű, kompakt kód**
- **Magasszintű nyelvek (C/C++) ↔ assembly**

Vivado fejlesztői környezet

Integrált fejlesztői környezet a Xilinx 7-es sorozatú vagy újabb eszközökhöz

ISE Design Suite

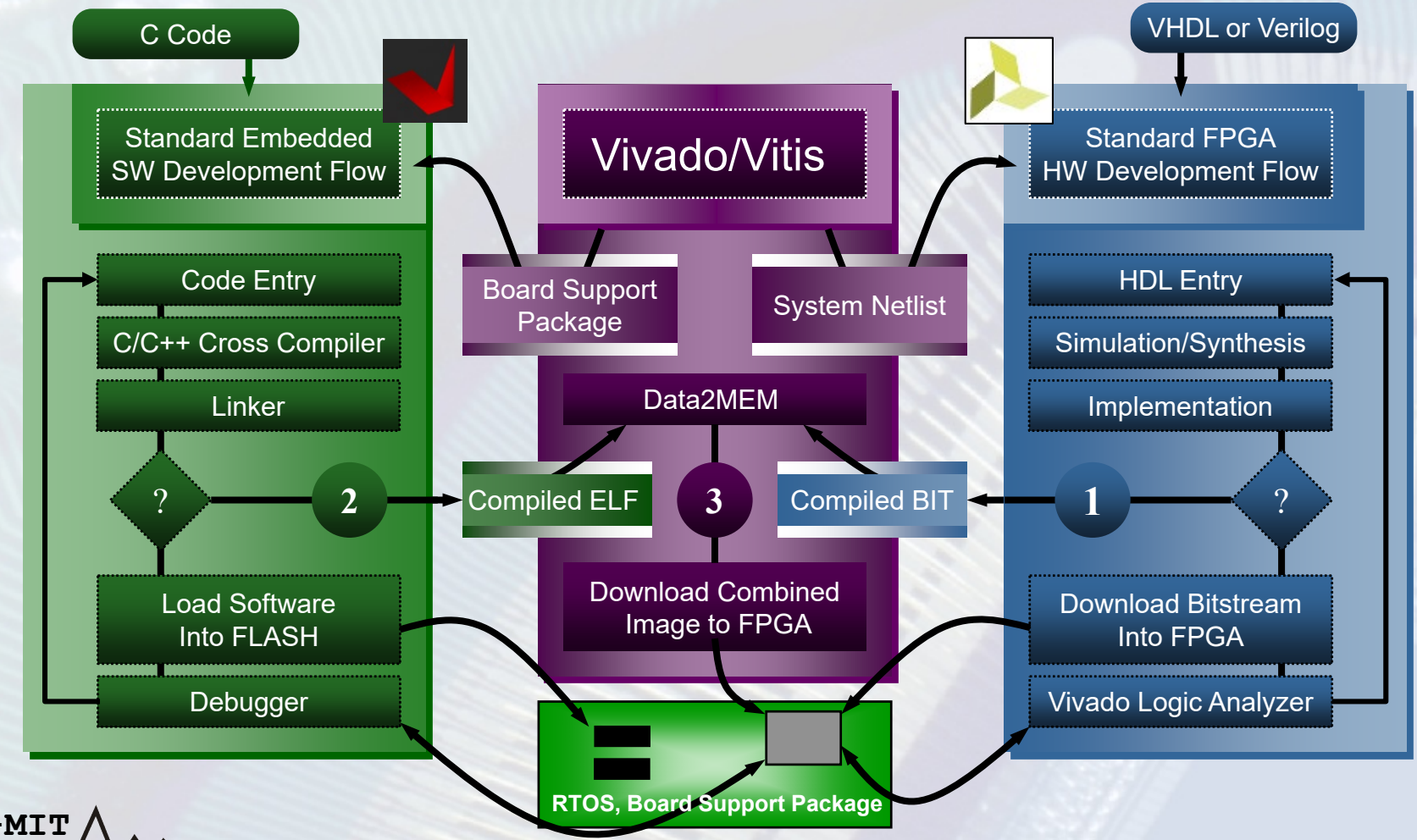
- **ISE: (elsősorban) HDL alapú fejlesztés**
- **EDK: IP alapú fejlesztés (processzoros rendszerek)**
- **ChipScope: belső logikai analizátor**
- **SDK: szoftverfejlesztés**

Vivado Design Suite

- **Vivado: HW fejlesztés**
  - HDL alapú fejlesztés
  - IP Integrator
  - Vivado Logic Analyzer
- **Vitis: szoftverfejlesztés**
- **Vitis HLS: magasszintű szintézis (C → hardver)**

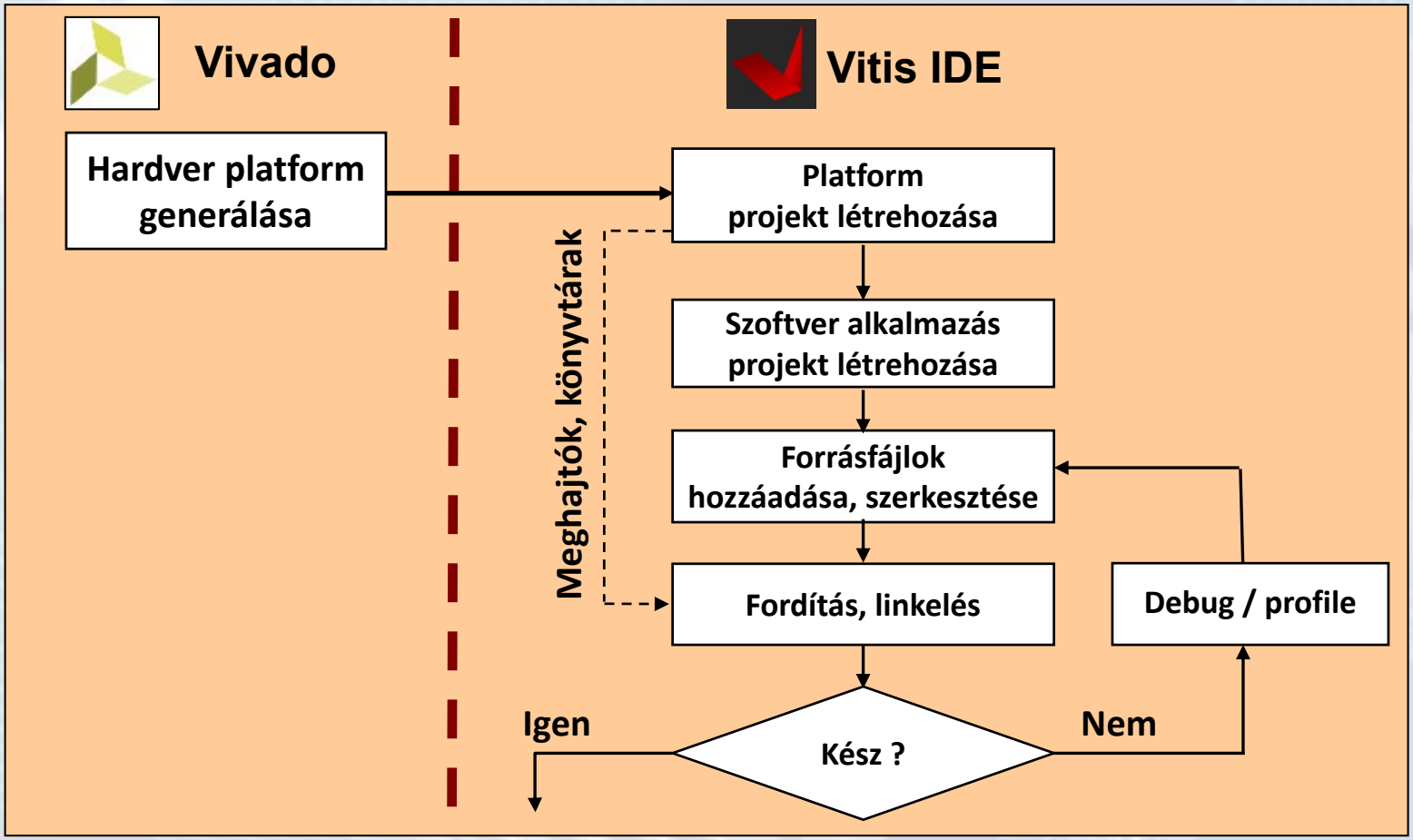
Vivado fejlesztői környezet

A fejlesztési folyamat áttekintése



Vivado fejlesztői környezet

A fejlesztési folyamat áttekintése



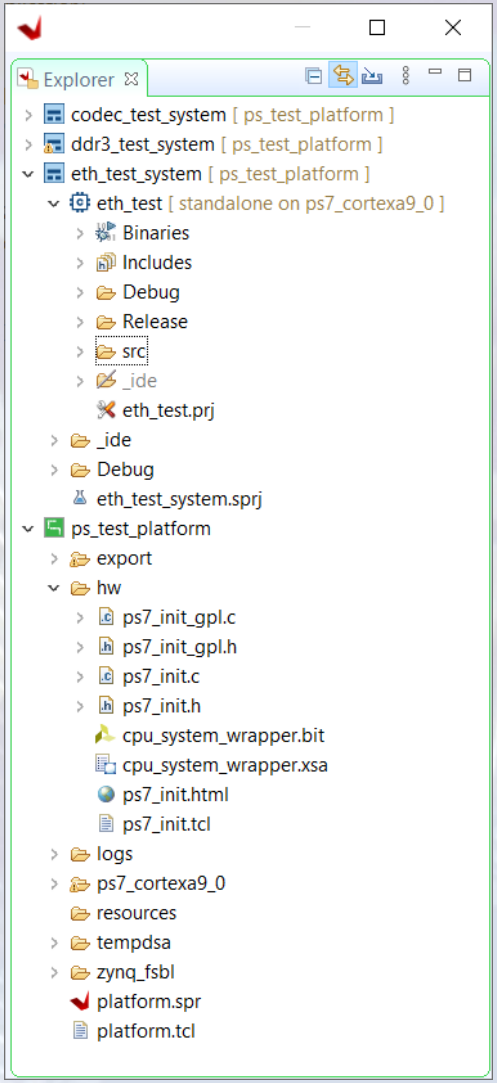
Vitis projekt típusok

1. Platform projekt

- Hardver komponensek
  - HW inicializálás, perifériák, BIT fájl
- Szoftver komponensek
  - Domain-ek: OS, CPU, futtatókörnyezet
  - Board support package (BSP): eszközmeghajtók, szoftver könyvtárak

2. Alkalmazás projekt(ek)

- Az alkalmazások végső tárolója
- Egy domain-hez van társítva
- Fordítás eredménye: ELF fájl

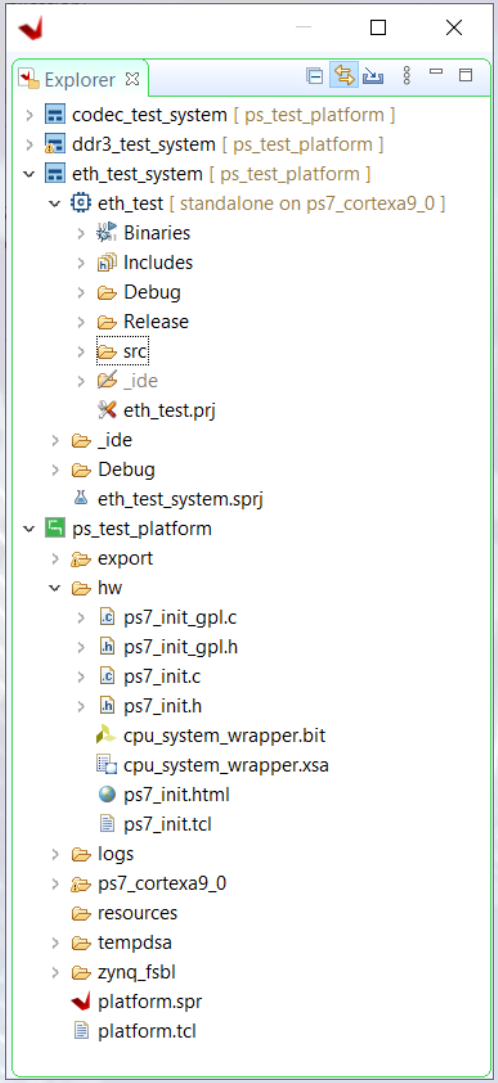


Vitis projekt típusok

3. Rendszer (system) projekt

- Azonos hardveren futó, különböző domain-ekhez társított alkalmazásokat csoportosítja
- Célja a boot image létrehozása

Domain „konfigurációk”		
Operációs rendszer	***Processzor***	***Futtatókörnyezet***
Standalone	ARM CPU magok MicroBlaze	C/C++
Linux		C/C++, OpenCL
FreeRTOS		C/C++



Board Support Package

Eszközmeghajtó programok

- **Rétegzett architektúra**
- **2. réteg: RTOS alkalmazási réteg**
- **1. réteg: Magas szintű eszközmeghajtók**
  - Teljes funkcionalitás
  - Többféle processzor és operációs rendszer támogatott
- **0. réteg: Alacsony szintű eszközmeghajtók**

2. réteg: RTOS
1. réteg: magas szintű meghajtók
0. réteg: alacsony szintű meghajtók

Board Support Package

0. réteg: alacsony szintű eszközmeghajtó programok

- **Makrók és függvények, melyek lehetővé teszik a kisméretű rendszerek megvalósítását**
- **Tulajdonságok:**
  - Kis memóriaigény
  - Egyáltalán nincs vagy csak kevés hibaellenőrzés
  - Csak az eszköz alapfunkcióit támogatják
  - Nem támogatják az eszköz konfigurációs paramétereket
  - Több eszközpéldány támogatása: báziscímek megadásával
  - Csak lekérdezéses I/O
  - Blokkoló hívások
  - Header fájlok végződése “_l” (például xuartlite_l.h)

Board Support Package

1. réteg: magas szintű eszközmeghajtó programok

- **Makrók és függvények, melyek lehetővé teszik az eszközök minden tulajdonságainak kihasználását**
- **Tulajdonságok:**
  - Absztrakt API, amely leválasztja a programozói interfészt a hardver rendszer változásairól
  - Támogatják az eszköz konfigurációs paramétereket
  - Több eszközpéldány támogatása
  - Lekérdezéses és megszakításos I/O
  - Nem blokkoló hívások a komplex alkalmazások támogatásához
  - Nagy memóriaigény lehetséges
  - Tipikusan pufferelt adatátvitel a bájtos adatátvitel helyett
  - Header fájlok végződése nem “_l” (például xuartlite.h)

Board Support Package

- **Uartlite magas szintű meghajtó**
  - XStatus XUartLite_Initialize(XUartLite *InstancePtr, Xuint16 DeviceId)
  - void XUartLite_ResetFifos(XUartLite *InstancePtr)
  - unsigned int XUartLite_Send(XUartLite *InstancePtr, Xuint8 *DataBufferPtr, unsigned int NumBytes)
  - unsigned int XUartLite_Recv(XUartLite *InstancePtr, Xuint8 *DataBufferPtr, unsigned int NumBytes)
  - Xboolean XUartLite_IsSending(XUartLite *InstancePtr)
  - void XUartLite_GetStats(XUartLite *InstancePtr, XUartLite_Stats *StatsPtr)
  - void XUartLite_ClearStats(XUartLite *InstancePtr)
  - XStatus XUartLite_SelfTest(XUartLite *InstancePtr)
  - void XUartLite_EnableInterrupt(XUartLite *InstancePtr)
  - void XUartLite_DisableInterrupt(XUartLite *InstancePtr)
  - void XUartLite_SetRecvHandler(XUartLite *InstancePtr, XUartLite_Handler FuncPtr, void *CallBackRef)
  - void XUartLite_SetSendHandler(XUartLite *InstancePtr, XUartLite_Handler FuncPtr, void *CallBackRef)
  - void XUartLite_InterruptHandler(XUartLite *InstancePtr)
- **Uartlite alacsony szintű meghajtó**
  - void XUartLite_SendByte(Xuint32 BaseAddress, Xuint8 Data)
  - Xuint8 XUartLite_RecvByte(Xuint32 BaseAddress)

Board Support Package

Szoftver könyvtárak

- **Matematikai könyvtár (*libm*)**
- **Standard C könyvtár (*libc*)**
  - A könyvtár függvényei automatikusan rendelkezésre állnak
- **Xilinx C nyelvű meghajtók és könyvtárak (*libxil*)**
  - FAT fájlrendszer: ***xilffs***
  - Memória fájlrendszer: ***xilmfs***
  - TCP/IP hálózati kommunikáció: ***lwip***
  - Párhuzamos flash memória támogatás: ***xilflash***
  - Soros flash memória támogatás: ***xilisf***
  - Stb.

Board Support Package

xparameters.h header fájl

- A rendszerben lévő hardver egységek paramétereit tárolja
- Elnevezési konvenció: ***XPAR_[periféria_név]_[paraméter_név]***

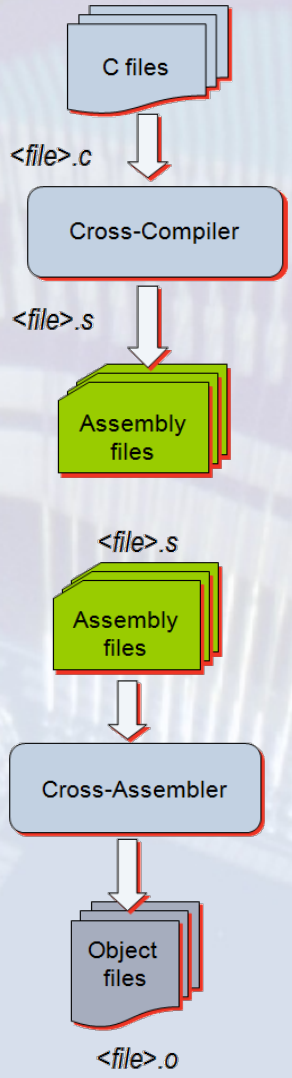
```
/* Definitions for driver GPIO */
#define XPAR_XGPIO_NUM_INSTANCES 3

/* Definitions for peripheral AXI_GPIO_EXP_STAT */
#define XPAR_AXI_GPIO_EXP_STAT_BASEADDR 0x46000000
#define XPAR_AXI_GPIO_EXP_STAT_HIGHADDR 0x4600FFFF
#define XPAR_AXI_GPIO_EXP_STAT_DEVICE_ID 0
#define XPAR_AXI_GPIO_EXP_STAT_INTERRUPT_PRESENT 1
#define XPAR_AXI_GPIO_EXP_STAT_IS_DUAL 1

/* Definitions for peripheral AXI_GPIO_LED_DISP */
#define XPAR_AXI_GPIO_LED_DISP_BASEADDR 0x44000000
#define XPAR_AXI_GPIO_LED_DISP_HIGHADDR 0x4400FFFF
#define XPAR_AXI_GPIO_LED_DISP_DEVICE_ID 1
#define XPAR_AXI_GPIO_LED_DISP_INTERRUPT_PRESENT 0
#define XPAR_AXI_GPIO_LED_DISP_IS_DUAL 1
```

Linker script

- **A tárgykód és a végrehajtható fájl szekciókból áll**
  - ***.text***: végrehajtható kód
  - ***.rodata***: csak olvasható adatok
  - ***.sdata2***: kis méretű (max. 7 byte), csak olvasható adatok
  - ***.sbss2***: kis méretű, nem inicializált, csak olvasható adatok
  - ***.data***: írható/olvasható adatok
  - ***.sdata***: kis méretű, írható/olvasható adatok
  - ***.sbss***: kis méretű, nem inicializált adatok
  - ***.bss***: nem inicializált adatok
  - ***.heap***: szekció a dinamikus memória foglaláshoz
  - ***.stack***: verem szekció
- **Linker script: memória területek címe és mérete, a szekciók tulajdonságai és elhelyezkedésük a memória területeken**

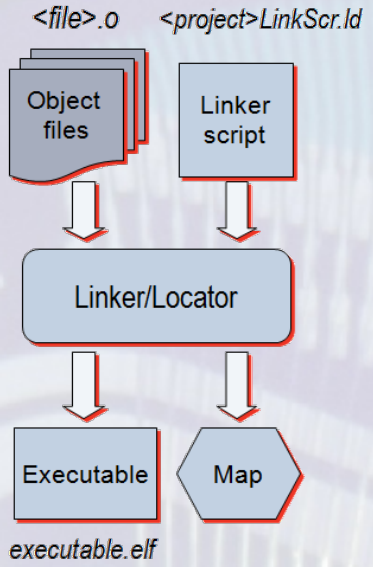
GNU fejlesztőeszközök

- **Az SDK a GNU fejlesztőeszközöket használja**
  - Parancssoros eszközök
- **GNU C/C++ fordító**
  - Négy programot futtat le egymás után
  - ***Preprocesszor***: feldolgozza a preprocesszor direktívákat és a makrókat (helyettesítés)
  - ***Nyelvspecifikus fordító***: assembly forrásfájlt készít a C vagy C++ kódból (.c, .cpp, .cc → .s)
  - ***Assembler***: az assembly forrásfájlból tárgykódot (object fájl) készít (.s → .o)
  - ***Linker***: a tárgykód fájlokból és a könyvtárakból elkészíti a futtatható fájlt (.o, .a → .elf)



GNU fejlesztőeszközök

- **Az assembler és a linker külön is futtatható**
- **A tárgykód (object) fájl tartalma**
  - Az assembler által előállított gépi kód
  - Konstans adatok, külső referenciák
  - Debug információk
- **Archiver:**
  - Szoftver könyvtár létrehozása tárgykód fájlokból
  - Szoftver könyvtár tartalmának módosítása
  - Tárgykód fájlok kinyerése szoftver könyvtárból
  - Az SDK pl. a BSP könyvtár létrehozásához használja



Vitis IDE – Debug funkciók

A debugger funkciói

- **Töréspontok elhelyezése és eltávolítása:**
  - Dupla kattintás a sor száma mellett a szürke területen

```
32      //Clear the timer interrupt flag.
33      csr = XTmrCtr_GetControlStatusReg(XPAR_AXI_TIMER_0_BASEADDR, 0);
34      XTmrCtr_SetControlStatusReg(XPAR_AXI_TIMER_0_BASEADDR, 0, csr);
```

- **A program végrehajtás vezérlése**
  - ***Resume***: a program futásának folytatása
  - ***Suspend***: a program futásának megállítása
  - ***Step Into***: az aktuális forráskód sor végrehajtása
    - Függvényhívás esetén belép a függvénybe
  - ***Step Over***: az aktuális forráskód sor végrehajtása
    - Függvényhívás esetén lefut a függvény, nem lép be a függvénybe
  - ***Step Return***: a futás leáll a függvényből való kilépéskor
  - ***Run to Line***: futtatás a kijelölt forráskód sorig

Vitis IDE – Debug funkciók

A debugger funkciói

- **Forráskód nézet (C/C++, disassembly)**
  - Töréspontok elhelyezése, eltávolítása
  - Változók értékeinek megtekintése
    - Vigyük az egérkurzort a változó fölé
- **Debug nézet**
  - Veremkeret
  - A program végrehajtás vezérlése
- **Variables: a lokális változók listája, értékeik módosítása**
- **Breakpoints: töréspontok engedélyezése, tiltása**
- **Registers: a CPU regiszterek listája, értékeik módosítása**
- **Expressions: kifejezések értékének figyelése (watch)**
- **Memory: memóriatartalom megjelenítése, módosítása**