



EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének
együttes javítása a Pannon Egyetemen

EMBEDDED SYSTEM DEVELOPMENT (MISAM154R)



Created by Zsolt Voroshazi, PhD
voroshazi.zsolt@virt.uni-pannon.hu

Updated: 25. Sept. 2020.

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok



BEFEKTETÉS A JÖVŐBE

2. XILINX VIVADO

Embedded System development environment

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok



BEFEKTETÉS A JÖVŐBE

Topics covered

1. Introduction – Embedded Systems
2. FPGAs, Digilent ZyBo development platform
3. **Embedded System - Firmware development environment (Xilinx Vivado – „EDK” Embedded Development)**
4. Embedded System - Software development environment (Xilinx VITIS – „SDK”)
5. Embedded Base System Build (and Board Bring-Up)
6. Adding Peripherals (from IP database) to BSB
7. Adding Custom (=own) Peripherals to BSB
8. Development, testing and debugging of software applications – Xilinx VITIS (SDK)
9. Design and Development of Complex IP cores and applications (e.g. camera/video/audio controllers)

Pre-requirities

- Digital Circuits and Technics I-II.
- Design Methods: Learn how to use the Xilinx Vivado Design Suite (HLx) development environment
- Basic C / C ++ knowledge
 - Programming I. / II.
- Basics of Digital Systems and Computer Architectures
- Basics of HDL knowledge (e.g. VHDL language)
 - Design Methods with Programmable Logic Devices (BSc in Informatics, BSc in Electrical Engineering)

Xilinx product portfolio (2020)



- Previously: ISE and XPS Platform Studio (till 2013)
 - Support up to FPGA Series-6 and older devices. Obsoleted!
 - Switching to newer Vivado/VITIS 2020.1 is recommended!



- **Vivado Design Suite HLx** (support APSoCs, UltraScale MPSoCs and FPGA Series-7

- Latest version: 2020.1
 - We use Vivado 2020.1 in the laboratory work



- Vivado HLS: High Level Synthesis

- Support C/C++, OpenCV languages



- ***VITIS** = Unified Software Platform 2020.1

- SDx: Software Defined

- SDAccel, SDSoC (reVision), SDNet (fully integrated into the new VITIS environment!)

- Support Cloud, C/C++, OpenCL

Xilinx Vivado + VITIS

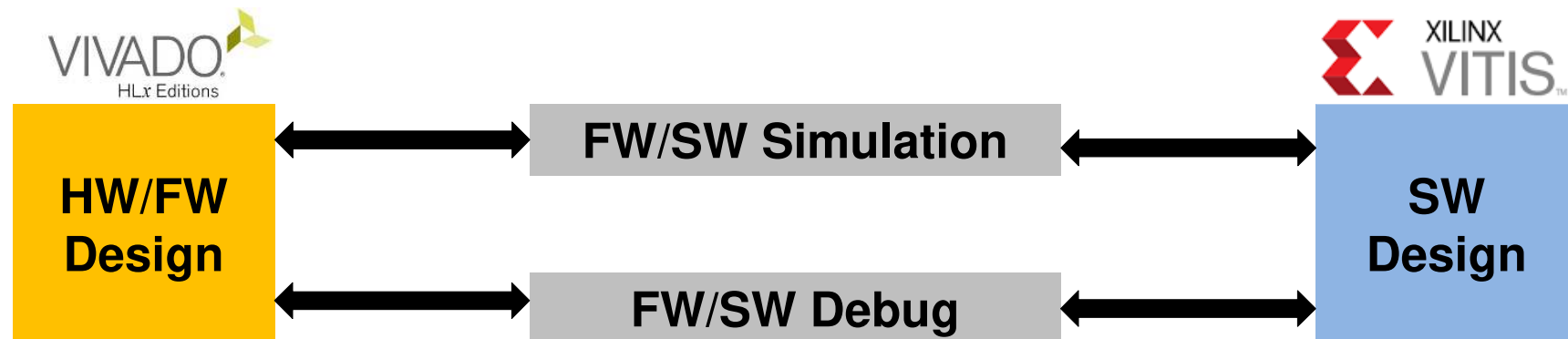
We use **Vivado Design Suite HLx 2020.1** in the laboratory work.

- Vivado Project Navigator: framework - IDE, various modules can be invoked (aspects of project management as PlanAhead)
 - **Embedded designer / IPI = IP integrator (~EDK) – designing and developing Embedded firmware ecosystem (FPGA logic + ARM cores)**
 - Xilinx IP Catalogue (free, or purchased IPs)
 - Custom-own, or 3rd Party IPs
 - Timing Analyzer
 - FPGA Editor
 - **HW Manager– Download Bitstream (FPGA configuration)**
 - **ILA: Integrated Logic Analyzer (~oscilloscope)**
- **VITIS as Unified Software Platform – Embedded software development (~SDK)**
 - Eclipse based IDE environment

Xilinx Vivado + VITIS

Two main modules in the IDE framework:

- **Vivado Embedded Designer / IP Integrator** (~ EDK): Xilinx FPGA-based integrated development environment for assembling and parameterizing embedded systems (at **HW / FW** level) for rapid prototype development
- **VITIS** as **SDK**: Eclipse-based integrated **SW** development environment to support embedded processors, and accelerators (e.g. MicroBlaze™, ARM™):
 - SW: C / C ++ applications,
 - GCC, GPP (built-in/external) compilers,
 - GNU debugger
 - Support for embedded OS (Linux distributions).





XILINX VIVADO AS EMBEDDED SYSTEM DEVELOPMENT

General description – design flow

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok



BEFEKTETÉS A JÖVŐBE

References



Xilinx Vivado Design Suite development tool:

- <https://www.xilinx.com/products/design-tools/vivado.html#documentation>



Vivado Getting Started Guide (UG-910)

- https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug910-vivado-getting-started.pdf



Zynq-7000 All Programmable SoC Software Developers Guide (UG-821)

- https://www.xilinx.com/support/documentation/user_guides/ug821-zynq-7000-swdev.pdf



Zynq-7000 All Programmable SoC Embedded Design tutorial (UG-1165)

- https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug1165-zynq-embedded-design-tutorial.pdf



Zynq-7000 APSoC Concepts, Tools, and Techniques (Hands-on Guide)

- https://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_4/ug873-zynq-ctt.pdf

Xilinx Vivado

Aims:

- Development of embedded systems (firmware) for Xilinx FPGAs (eg Spartan-7, Virtex-7, Kintex-7) or APSoCs (e.g. Zynq-7000 family),
 - Description and integration of components (IP), parameterization,
 - Methods and steps of HW / FW design implementation,
 - Demonstration and learning of development tools,
-
- VITIS: Introduce SDK as SW development environment, hardware debugging.


Elaboration of Embedded System (FW) in Vivado

- Integrate processor system in FPGA
 - Xilinx MicroBlaze: soft-core architectures, or
 - ARM: hard-core processors
- Design and instantiate IP blocks
- Connect Reset / Clock / Debug ports
- Set peripheral interconnections
 - AMBA AXI v.4 busz interfész (supported from Vivado 2012.x and EDK 12.x)
- Map addresses in the address space
- Validate layout (=block diagram)

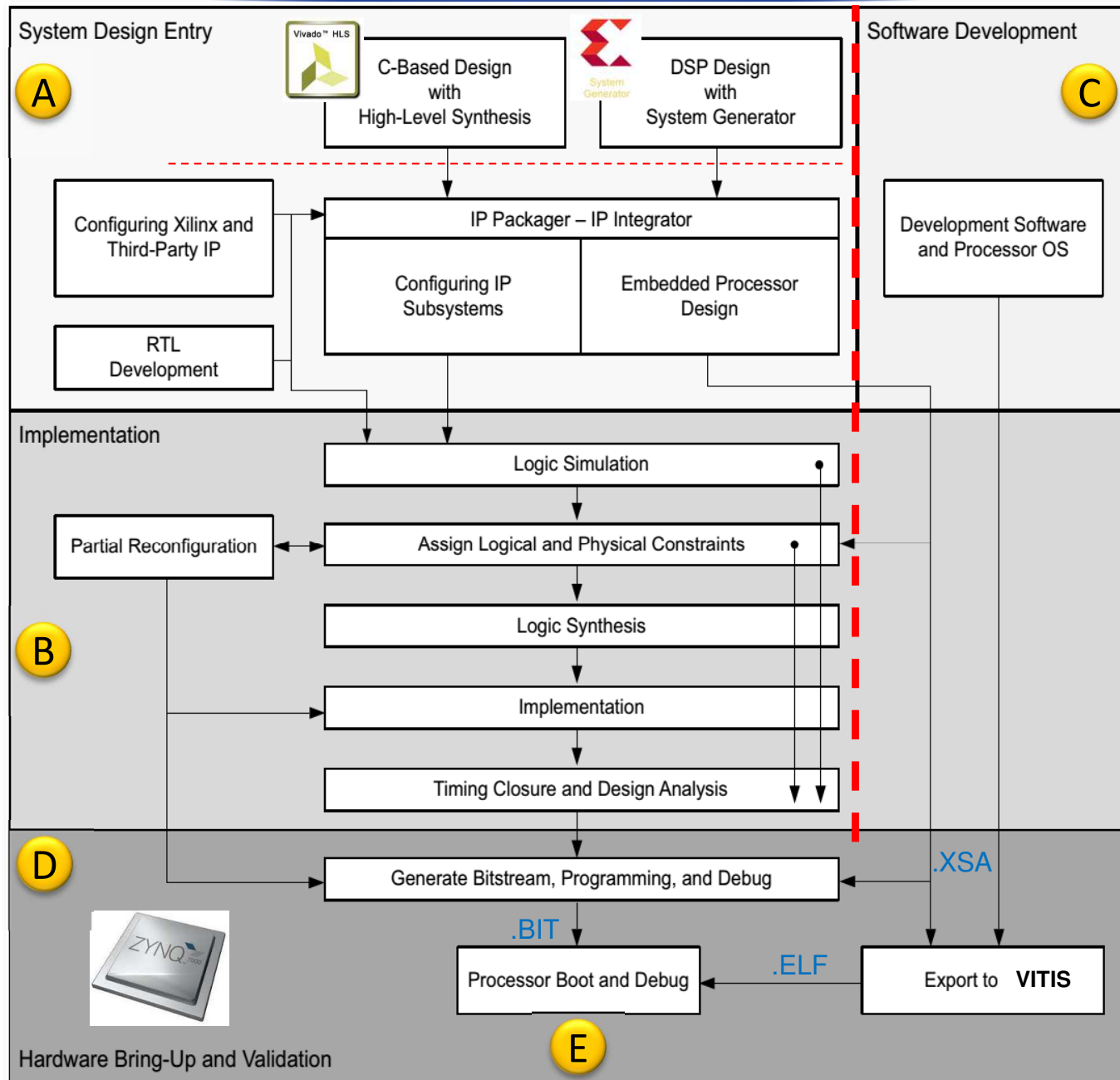
Elaboration of Embedded Software (SW/OS) in VITIS

- Generate SW libraries and drivers
 - C / C ++ language support
 - Compiling a SW application
 - SW drivers (low-high level), routines (interrupt), pre-defined API functions, SW stacks
- Operating system (OS or RTOS)
 - Stand-alone (non-OS) vs. Bare-metal, resp.
 - Embedded Linux OS (PetaLinux), vs. VxWorks RTOS distros, etc.
- Booting

Embedded Systems - Development Flow

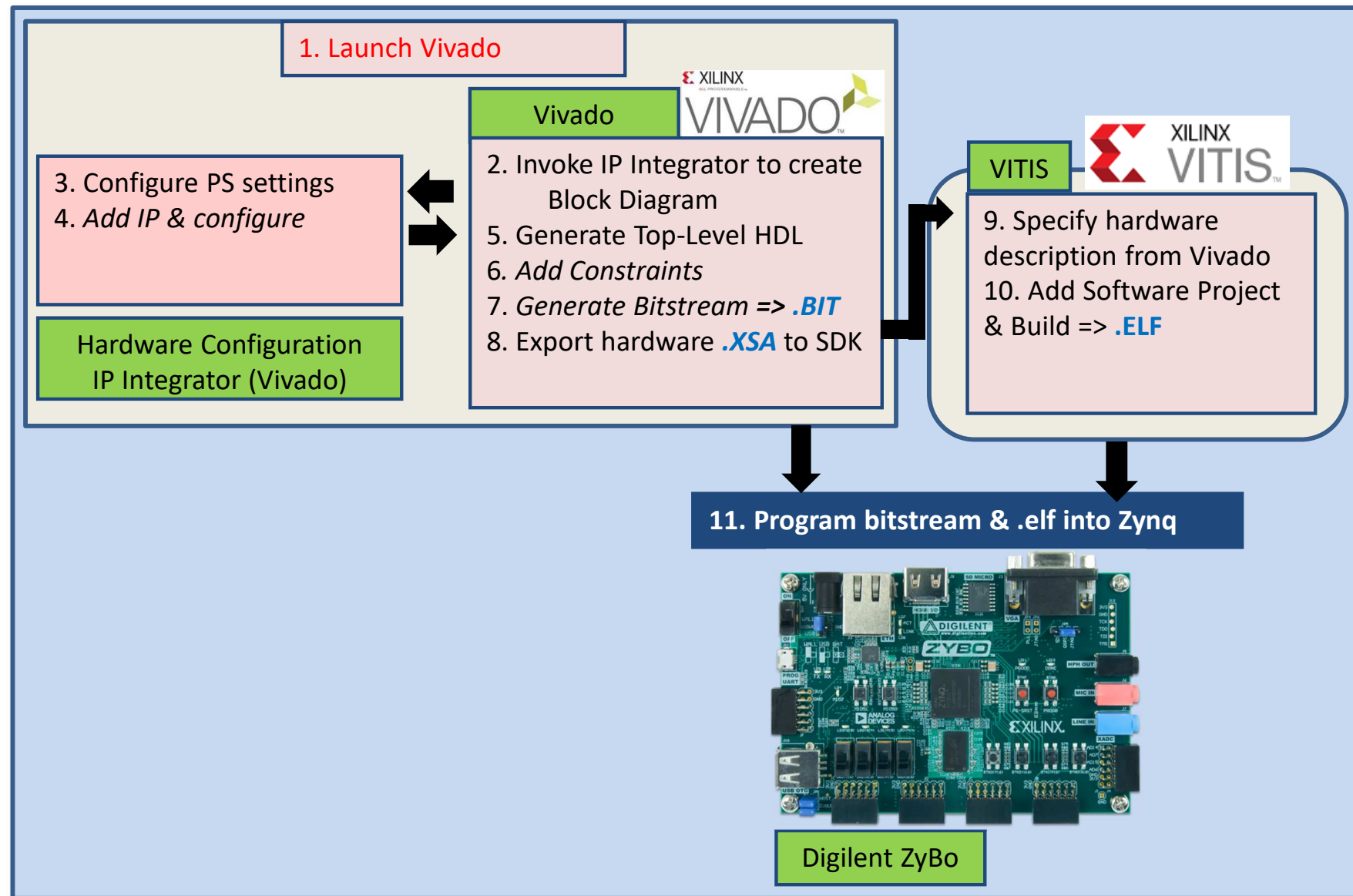
- 
- A** Embedded HW/FW development (**VIVADO**)
 - IP Integrator: rapid prototype development for target hardware
 - Extension of an existing embedded HW-FW system
 - Integrating peripherals in the IP catalog, or using custom (self-made) IP peripherals,
 - Generating HDL netlist (description)
 - B** Implementation
 - Synthesis, MAP, PAR, and timing analysis (user constraints)
 - C** Development of embedded SW in SDK (**VITIS**)
 - BSP: Board Support Package / Domain
 - Generating device libraries and drivers
 - Creating SW Application and Debugging
 - Optional: debug application using Xilinx Microprocessor Debug (XMD) and GNU debugger (gdb) on ARM / MicroBlaze cores
 - D** Configuration and testing
 - bitstream (.BIT) generation and configuration on FPGA
 - E** Load configuration (optional)
 - Initialization of external Flash memory, resp. boot procedure

Embedded Systems - Development Flow



Embedded Systems - Development Flow

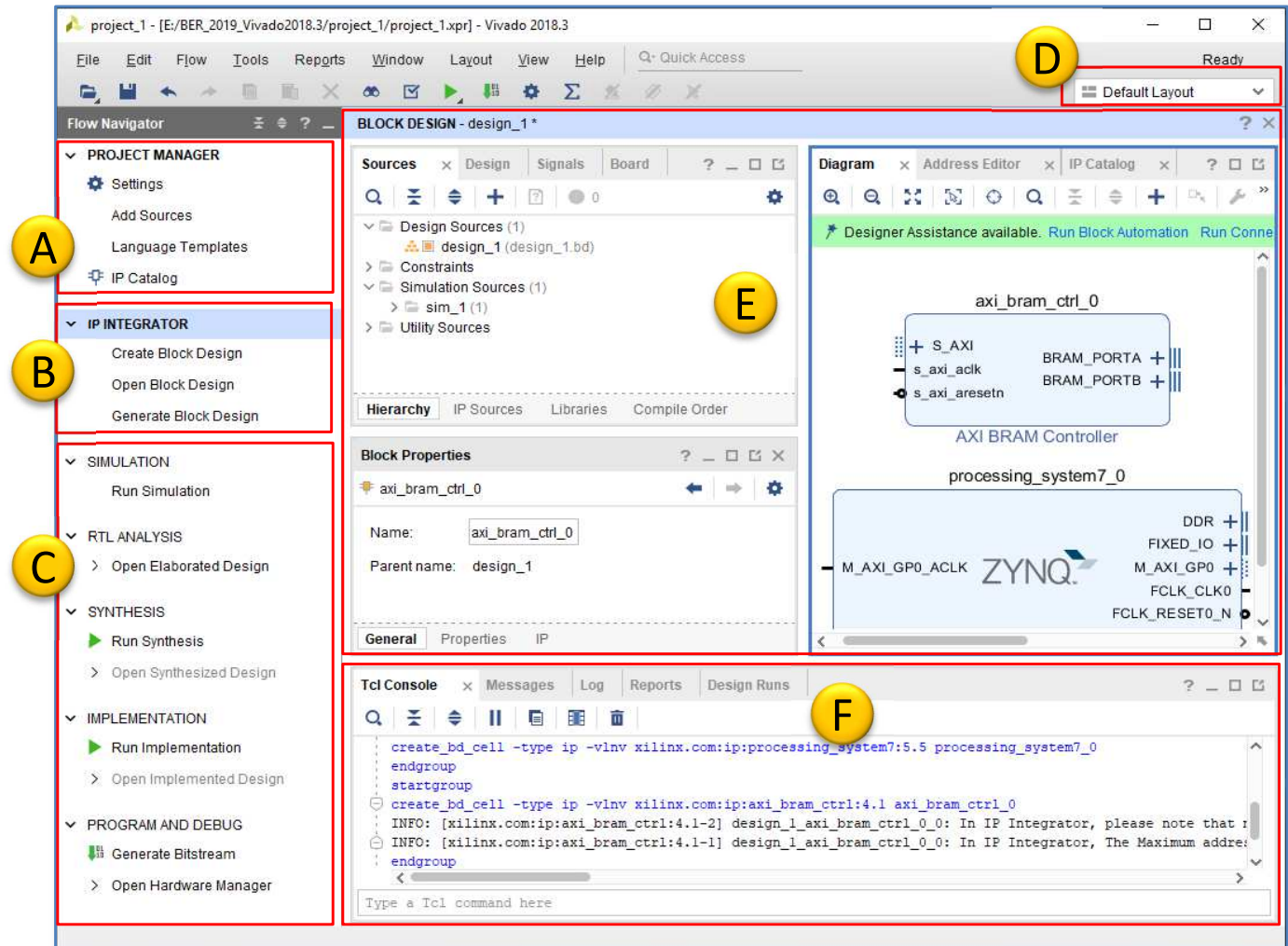
A demonstration example:



Vivado – GUI*



- **A:** Project Management
- **B:** IP Integrator
- **C:** FPGA Flow
- **D:** Layout Selection
- **E:** Project view/Preview Panel
- **F:** Console, Messages, Logs

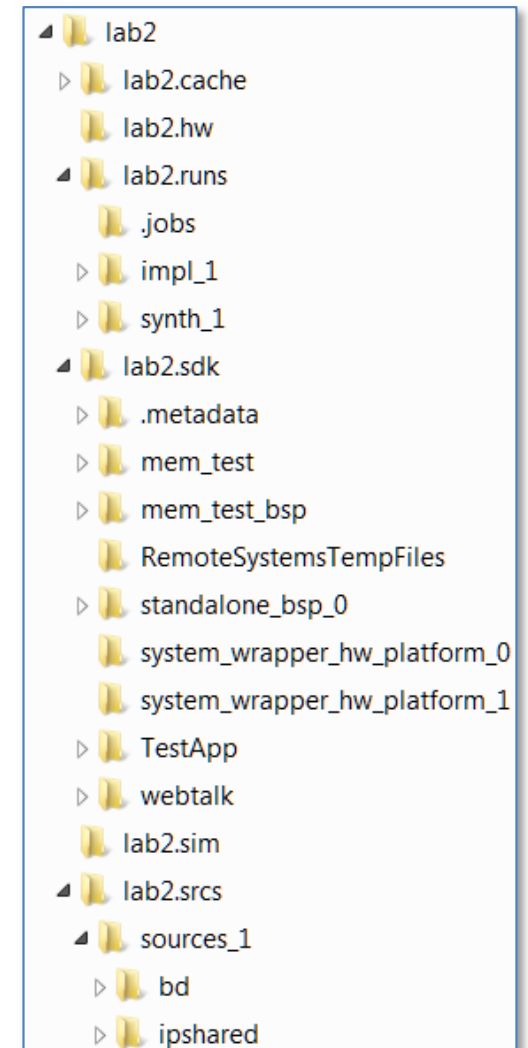


* Note: Vivado can be used in two ways:

- **Project mode:** project management approach with GUI (we use this!)
- **Non-project:** script command approach, without GUI, command line

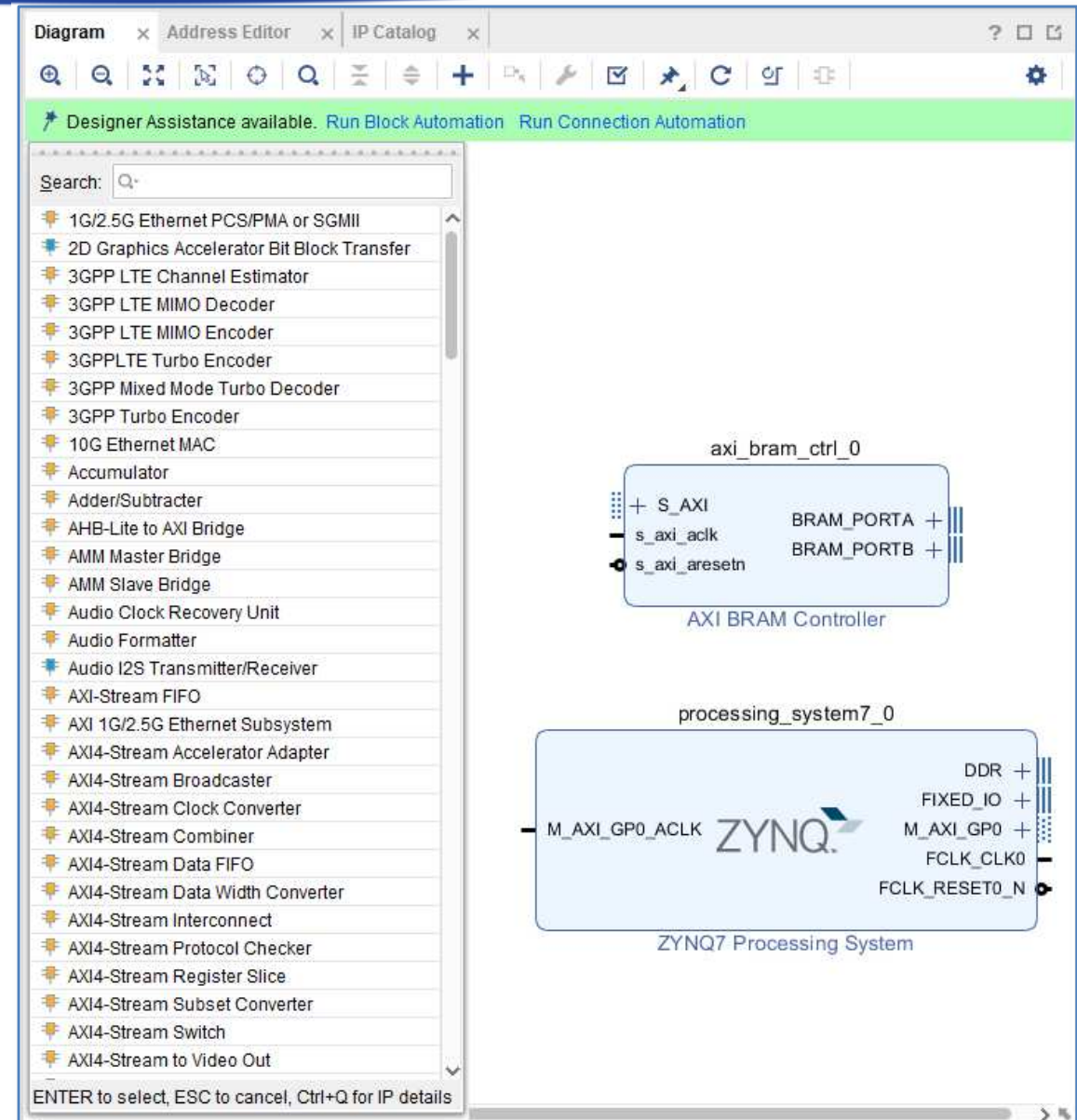
Vivado - project directory structure

- in Top-level directory
 - **.xpr** = Vivado Project File (xml), log, journal
- **/<name>.srcs**
 - Project source files (BD, XDC), IP Integrator files
- **/<name>.sim**
 - Szimulációs fileok
- **/<name>.runs**
 - szintézis, implementációs futtatások
- **/<name>.sdk**
 - SDK Export directory, Hardware Platform (xml)
- **/<name>.cache**
 - Temporary files

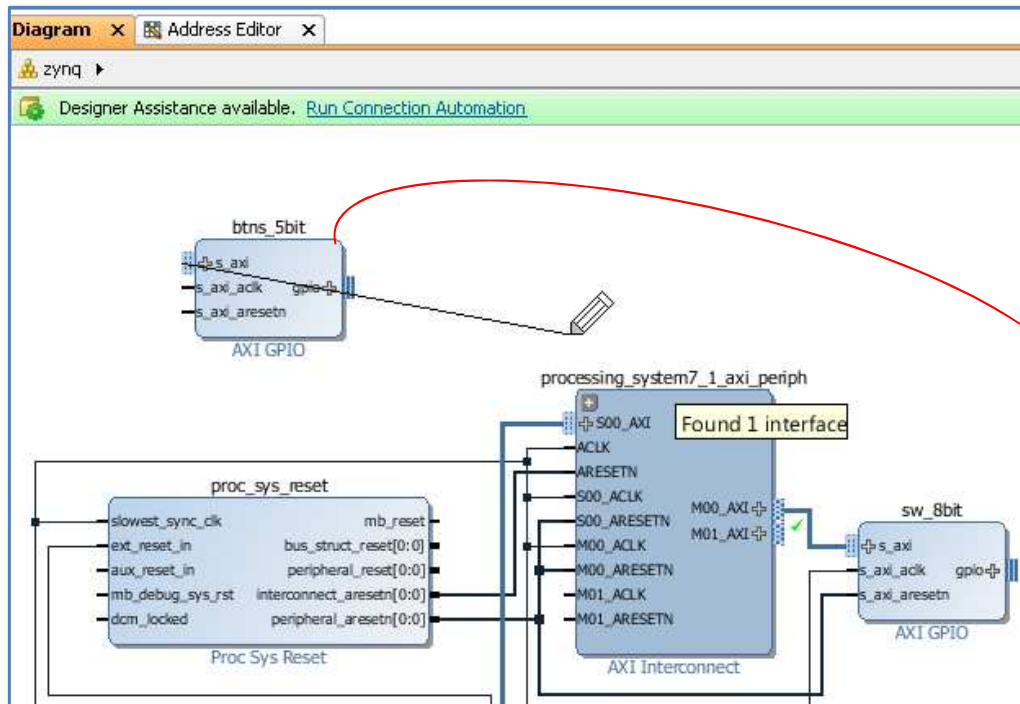


IPI – IP Integrator

- **IP – Intellectual Property**
 - „szellemi termék”
- **IP Catalogue**
 - Drag&Drop
- **Efficient GUI**
 - Auto router
 - Redraw
 - Yielding wrong connections
 - Block grouping, hierarchy handling
- **IP Packager**
 - Import custom, or external partner / 3rd party IPs



IPI – IP Integrator



Re-customize IP

AXI GPIO (2.0)

Documentation IP Location

☒ Show disabled ports

Component Name **axi_gpio_0**

Board IP Configuration

GPIO

☐ All Inputs

☐ All Outputs

GPIO Width **32** [1 - 32]

Default Output Value **0x00000000** [0x00000000,0xFFFF]

Default Tri State Value **0xFFFFFFFF** [0x00000000,0xFFFF]

☐ Enable Dual Channel

GPIO 2

☐ All Inputs

☐ All Outputs

GPIO Width **32** [1 - 32]

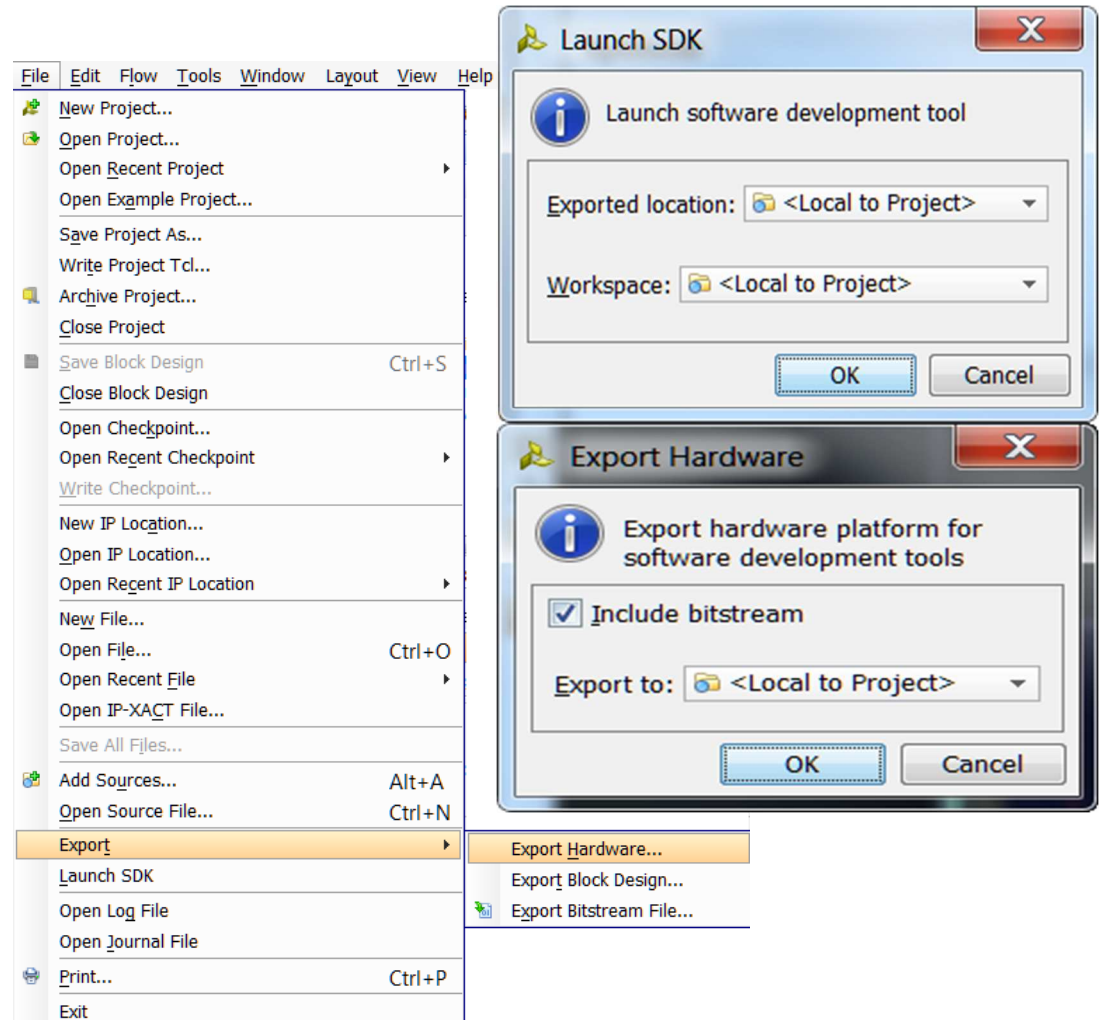
Default Output Value **0x00000000** [0x00000000,0xFFFF]

☐ Enable Interrupt

OK Cancel

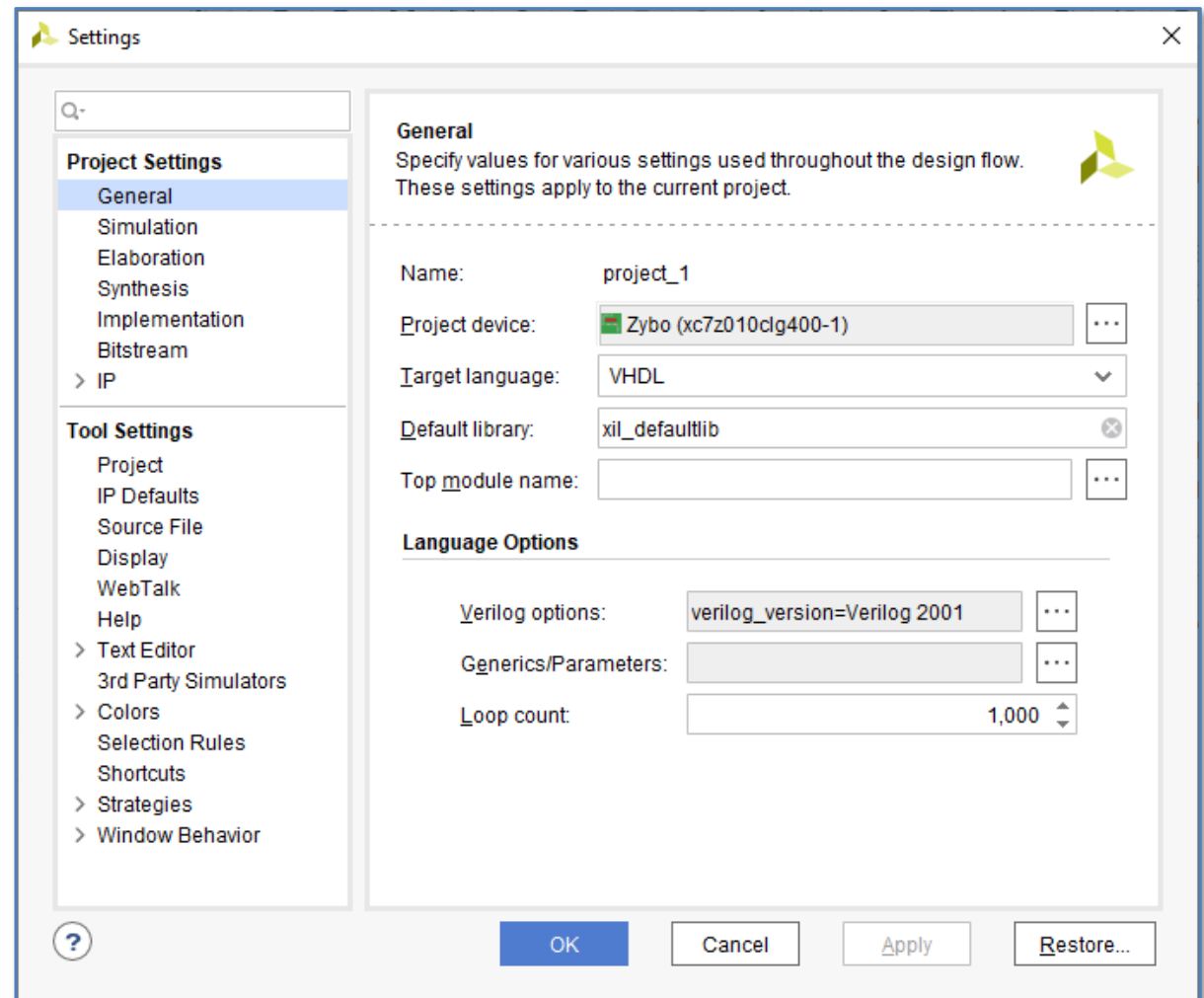
Exporting HW/FW → VITIS

- **.XSA:** Xilinx Support Archive file – HW description of embedded system
- **.BIT:** Bitstream exported (optional, if PL side configured)
- **VITIS SDK launch**



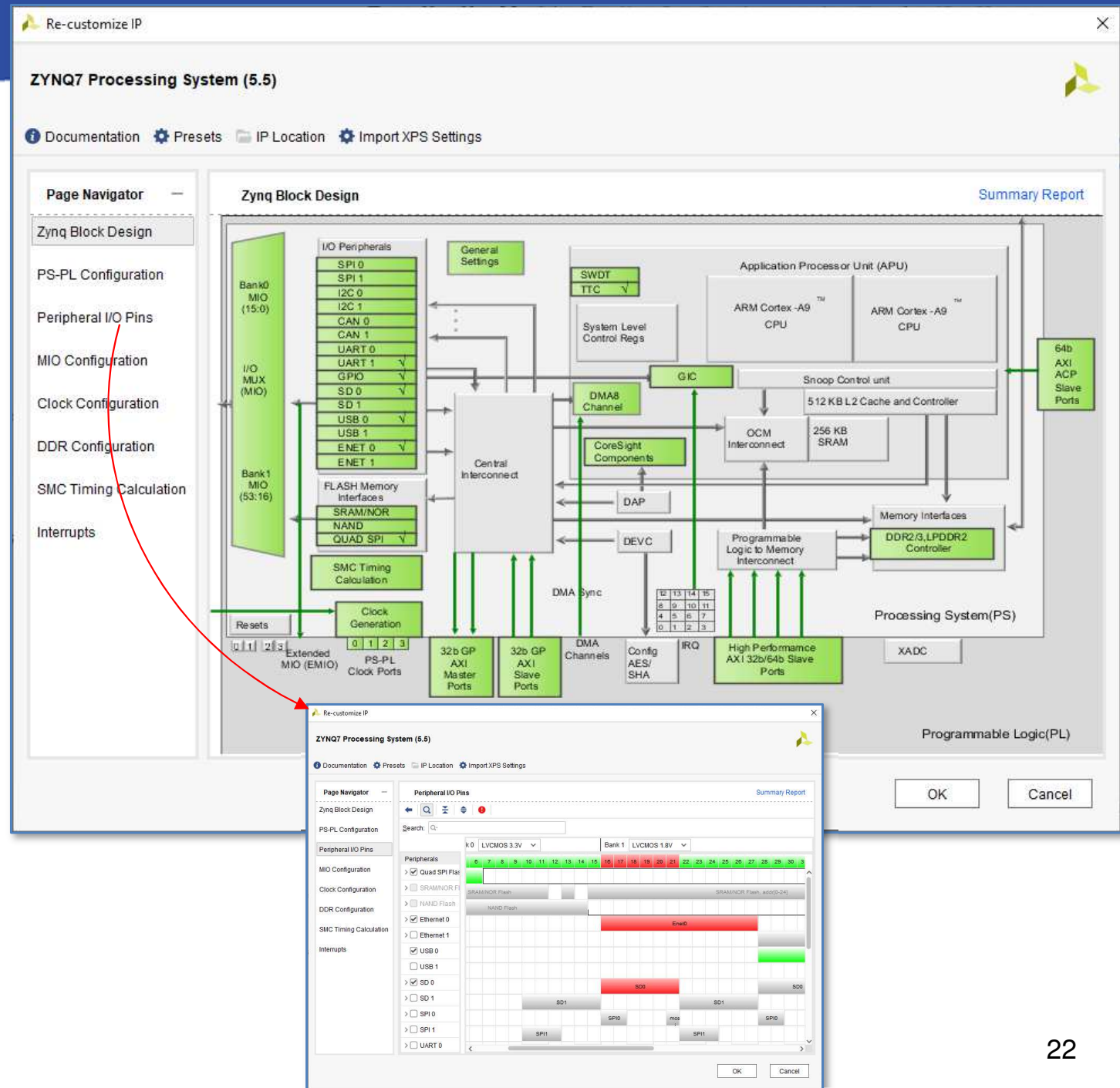
Vivado – Project settings

- Target device settings:
 - FPGA architecture,
 - Device size,
 - Package, pin count,
 - Speed grade
- Simulation, Synthesis, Implementation, Bitstream settings
- IP repository – search path
 - Path for external Ips
- Tool settings



GUI – Zynq PS

- Zynq PS customization
 - ARM cores
 - I/O peripherals
 - Memory system, DDR controller
- I/O partitioning:
 - PS, PL parts
 - MIO configurations
- MemoryMapped configuration registers



Clock configuration

- Clock Configuration (source from ARM_PLL, DDR_PLL, IO_PLL)
 - Changeable Input frequency (Processors, DDR)
 - Frequency for all IO Peripherals can be set
 - Enable/disable PL-side clocks
 - Set Timers

The screenshot displays the 'Clock Configuration' tool interface. On the left is a 'Page Navigator' with links to Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration (selected), DDR Configuration, SMC Timing Calculation, and Interrupts. The main area is titled 'Clock Configuration' and has tabs for 'Basic Clocking' and 'Advanced Clocking'. Under 'Basic Clocking', the 'Input Frequency (MHz)' is set to 50.000000 and the 'CPU Clock Ratio' is 6:2:1. Below this is a table of clock components.

Component	Clock Source	Requested Frequ...	Actual Frequency(...)	Range(MHz)
▼ Processor/Memory Clocks				
CPU	ARM PLL	650	650.000000	50.0 : 667.0
DDR	DDR PLL	525	525.000000	200.000000 : 534.000...
▼ IO Peripheral Clocks				
SMC	IO PLL	100	10.000000	10.000000 : 100.000000
QSPI	IO PLL	200	200.000000	10.000000 : 200.000000
ENET0	IO PLL	1000 Mbps	125.000000	
ENET1	IO PLL	1000 Mbps	10.000000	
SDIO	IO PLL	50	50.000000	10.000000 : 125.000000
SPI	IO PLL	166.666666	10.000000	0.000000 : 200.000000
> CAN				
▼ PL Fabric Clocks				
<input checked="" type="checkbox"/> FCLK_CLK0	IO PLL	100	100.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK1	IO PLL	50	10.000000	0.100000 : 250.000000

Abbreviations

- Major files (extensions) used or generated by Vivado IPI (FW) embedded system development tool:
 - **.XPR** = Xilinx Project File
 - **.XSA** = Xilinx Support Archive file (XML descriptor → input for VITIS SDK)
 - **.XDC** = Xilinx Constraint File (~ Synopsalapúis .SDC)
 - **.BIT** = Bitstream (FPGA configuration) file
- VITIS SDK (SW) fejlesztő eszköz által generált fontosabb fájlok:
 - **.ELF** = Extensible and Linkable Format
 - **.LD** = Linker Script Download file
 - **.BSP** = Board Support Package (Domain = SW app + OS)

MICROBLAZE™ EMBEDDABLE / SOFT PROCESSOR CORE

Brief description – **NOT DISCUSSED**

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok



BEFEKTETÉS A JÖVŐBE

References



MicroBlaze soft-processor

- <https://www.xilinx.com/products/design-tools/microblaze.html>



MicroBlaze Processor reference guide (Vivado 2020.1):

- https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug984-vivado-microblaze-ref.pdf



MicroBlaze quick start with Vitis:

- https://www.xilinx.com/support/documentation/quick_start/microblaze-quick-start-guide-with-vitis.pdf



MicroBlaze – Design Hub:

- <https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0020-microblaze-hub.html>

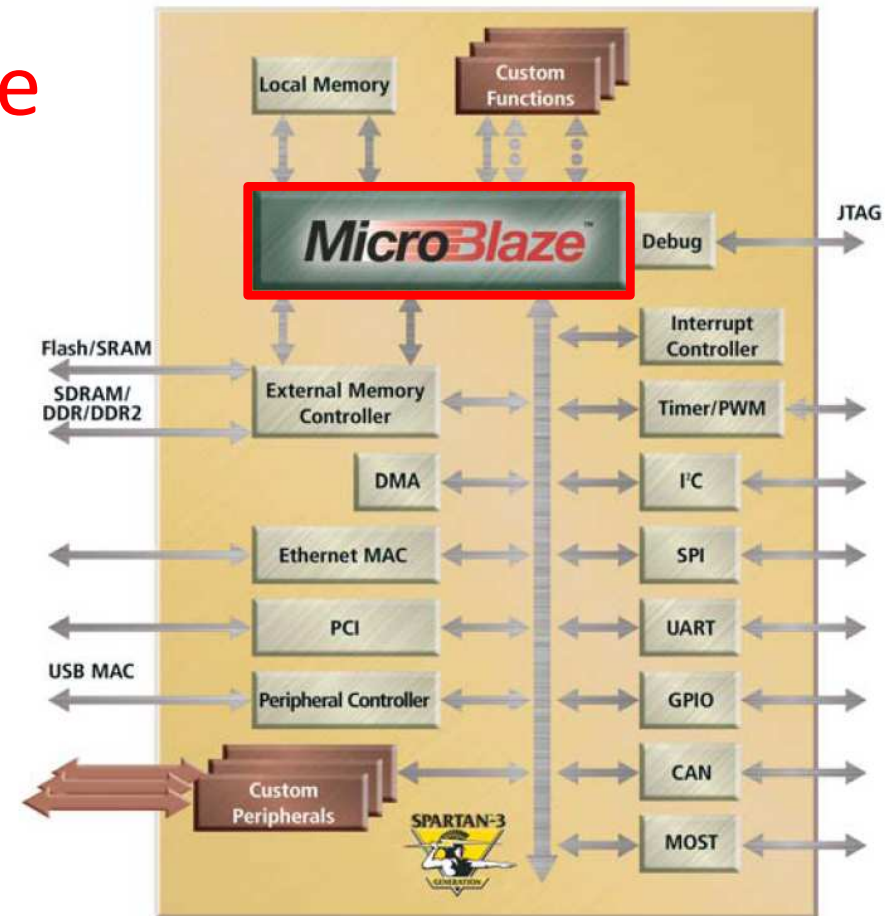
Processor cores on Xilinx FPGA/APSoC

- „Embed~~able~~“ = „soft-core“ processor = variable/non-dedicated
 - Xilinx *PicoBlaze*: 8-bit (build up from VHDL, Verilog HDL sources)
 - **Xilinx *MicroBlaze*: 32-bit (VITIS/Vivado SDK support!)**
 - It can be connected to AMBA-AXI in Vivado or PLB/OPB (deprecated) in former EDK dev. systems
 - 3rd Party: non-Xilinx manufactured (HDL source)
 - Pl. www.opencores.org
 - NEW ARM Cortex M1/M3 (2019): licensable cores (HDL source)
- Embed~~ded~~ = „hard-core“ processor = dedicated
 - IBM *PowerPC* 405/450 processor 32-bit (deprecated)
 - Exclusively in Virtex-II Pro, Virtex-4 FX, Virtex-5 FXT FPGAs!
 - *ARM Cortex-A9* processor (32-bit): connected to the ARM AMBA-AXI interface
 - Xilinx Zynq APSoC-n integrated ARM cores
 - ARM Cortex A53/A72 (64-bit)
 - Xilinx Zynq APSoC, UltraScale+ MPSoC
 - ARM Cortex-R5 – real-time (32-bit)
 - Xilinx Zynq Ultrascale RFSoc – radio chips


MicroBlaze – Soft-core processor

"Embeddable" processor core

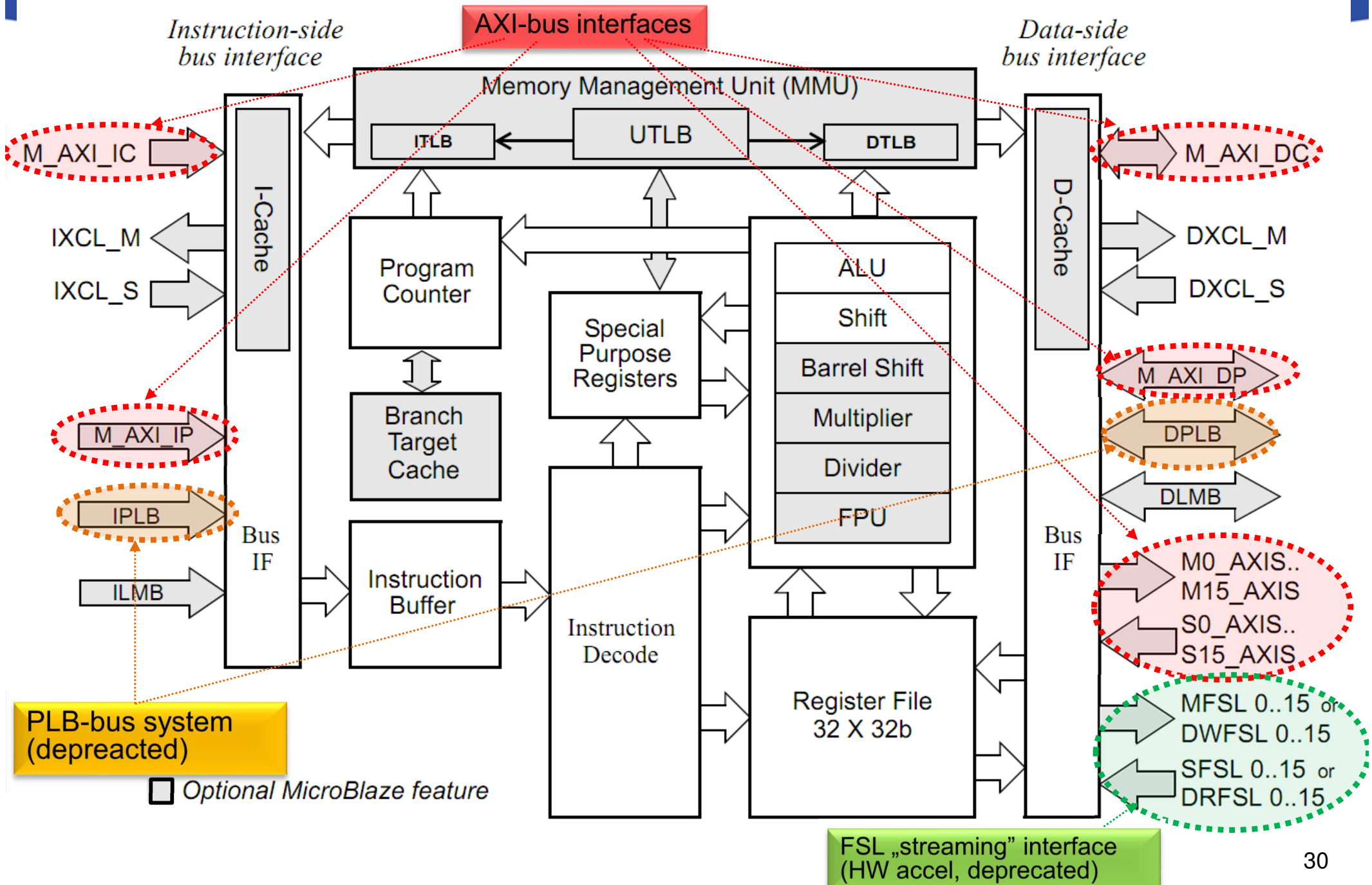
- RISC instruction set architecture
- 32-bit soft processor core
- 133+ MHz clock signal (PLB* / AXI bus)
- Harward memory architecture
 - Instruction Cache / Data Cache
- Low power consumption: ~ mW / MHz
- 3-/5-stages of data line pipe-line
- #32 32-bit general purpose registers
- Timing options (timer/counter)
- Many peripherals, communication interfaces can be connected (IP cores)
- It can be implemented on **any** Xilinx FPGA that has sufficient programmable logic resources and is supported by the development software!



MicroBlaze - Performance

- Instruction set (#85): run over 1 clock cycle with except of:
 - Load & Store (2 cycles): memory access
 - Multiplication (2 cycles)
 - Branches (1-3 cycles)
 -  **Details:** `\doc\microblaze.pdf` (installed DocNav directory)
- Operational frequency – fast speed grade, 5 stage-pipeline
 - 307 MHz on the Virtex-6 (-3) FPGA
 - 154 MHz on the Spartan®-6 (-3) FPGA
 - 119 MHz on the Spartan-3 (-5) FPGA
- Computing performance 1.15 **DMIPS**/MHz (=Dhrystone MIPS means a raw program-instruction execution speed/ sec)
- Resource utilization – num of LUTs in case of area vs. speed optimized version
 - 779/1,134 LUTs in the Virtex-6 FPGA (LUT-6)
 - 770/1,154 LUTs in the Spartan-6 FPGA (LUT-6!!)
 - 1,258/1,821 LUTs in the Spartan-3 FPGA (LUT-4!!)

MicroBlaze - Microarchitecture



MicroBlaze – general features

„Embeddable” processor core

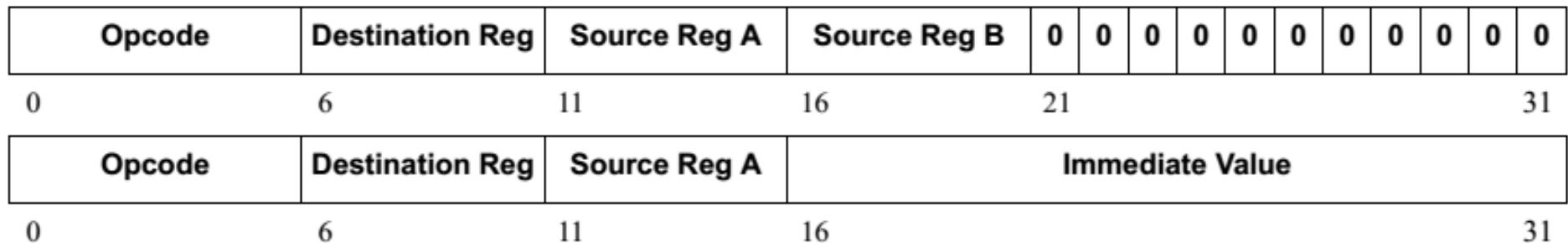
- **32-bit soft processor core**
- **RISC instruction set** architecture
 - ~ #85 instructions and their subtypes
- **100+ MHz** scalable clock (depending on FPGA architecture and bus system: AXI vs. older PLB)
- **Harvard** architecture
 - Instruction Cache / Data Cache
- Low power consumption: ~ mW / MHz
- #32 32-bit General Purpose Registers
- Timing options (timer/counter)
- Many types of peripherals, communication interface can be connected (in form of IP cores)
- It can be implemented on any Xilinx FPGA that has sufficient resources and is supported by the development software!

MicroBlaze – Parts I.

- *Flexible 32-bit processor core*
 - Current latest version v.11.0 (on Xilinx Vivado 2020.1)
 - Configurable 3-stage (resource optimized) or 5-stage data line pipeline (performance optimized version)
- Optionally configurable **instruction** and **data** cache (**I**-Cache, **D**-Cache)
- Direct mapping (1-way, group associative cache)
- It has #32 General File registers (32-bit each) (Register File)
- **Scalable 32-, 64-, or 128-bit data bus**
- **32-bit address / instruction bus**

MicroBlaze – Parts II.

- **32-bit instruction bus** (supports up to *3-address* instructions, and *two different addressing modes: Register and Register-Imm*)



- arithmetic,
- logic,
- Load & Store (Memory register for transfer operations - RISC feature),
- Program organizer instructions (e.g. Branch - conditional execution),
- Special instructions.

MicroBlaze – Parts III.

- optional **Memory Management and Memory Protection Unit (MMU)**
 - Required to run an embedded Linux OS on the MicroBlaze core with advanced memory management
- optional processor core **floating point arithmetic unit (FPU)**
 - Support for standard IEEE-754 format

MicroBlaze – Parts IV.

- **Barrel Shifter:** arbitrarily scrollable register (optional),
- **Hardware multiplier** (optional): when used, uses 3 MULT18x18 / DSP dedicated blocks.
 - This is a platform specific operation unit: multiplying the 32×32 -bit operands of the processor → with a 64-bit result,
- **Hardware divider** circuit (optional): includes multiplier circuits, logic cells and possibly dedicated BRAM cells,
- **Hardware-supported debugging** (with extensive debug function)
 - Xilinx MDM - MicroBlaze Debug Module,
 - Xilinx ILA, embedded logic analyzer module (s) for professional external measuring instruments
 - supported by another manufacturer (e.g. Agilent Trace Module)

MicroBlaze – Parts V.

- It basically follows the Big-Endian, i.e. bit / byte-reversed storage format (but since the EDK 13.x system, endianness can be configured in the two different bit / byte orders):
 - **Big-endian**: to support I/O peripherals that can be connected to the previous PLB (Processor Local Bus) bus
 - **Little-endian** (optional): to support the latest ARM hard-core / MicroBlaze soft-core AXI bus systems (in the latter case configurable)

The ARM logo is displayed in a white square in the top right corner. It consists of the letters "ARM" in a bold, blue, sans-serif font.

ARM™ EMBEDDED / HARD PROCESSOR CORE

General description

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok

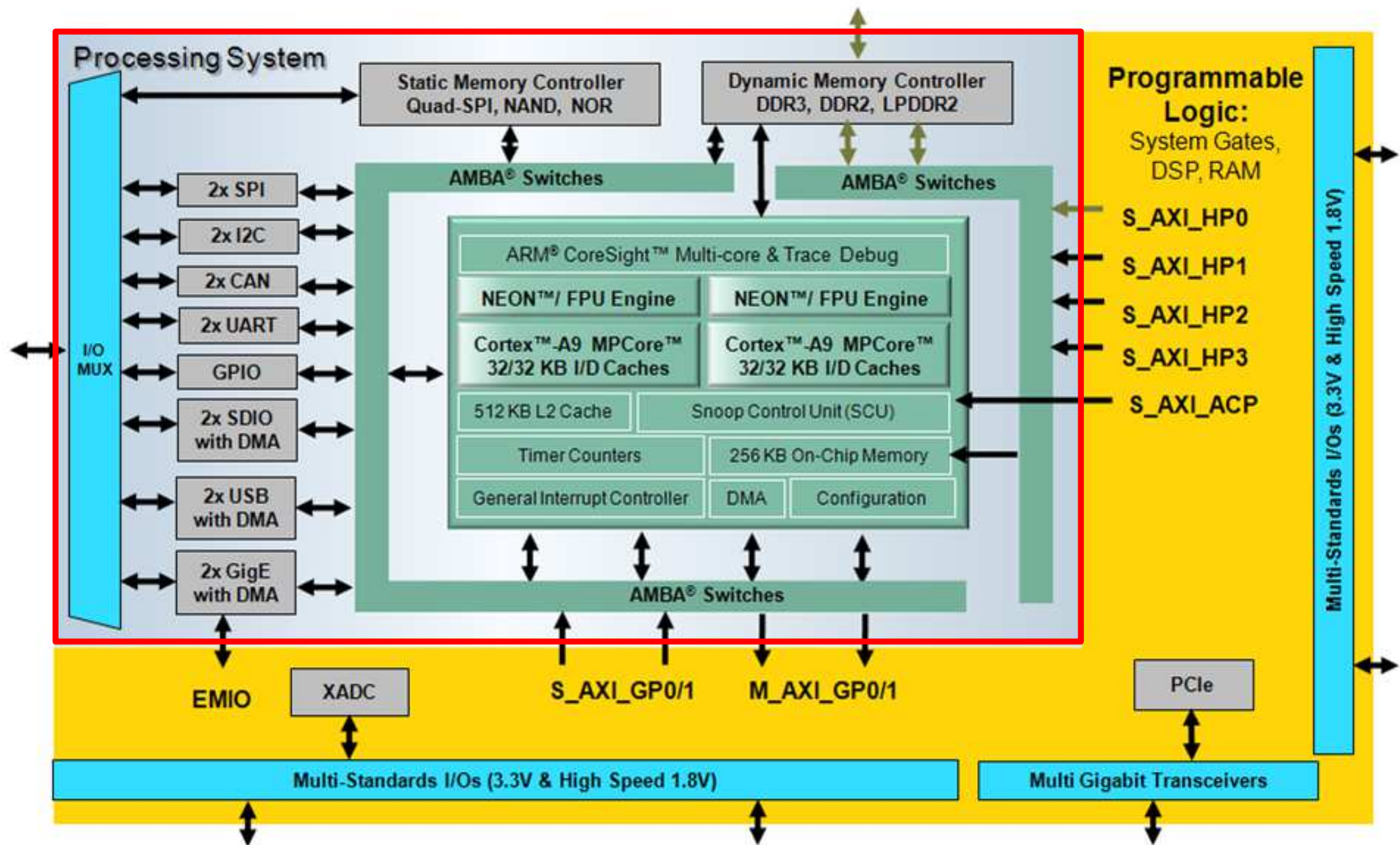


BEFEKTETÉS A JÖVŐBE

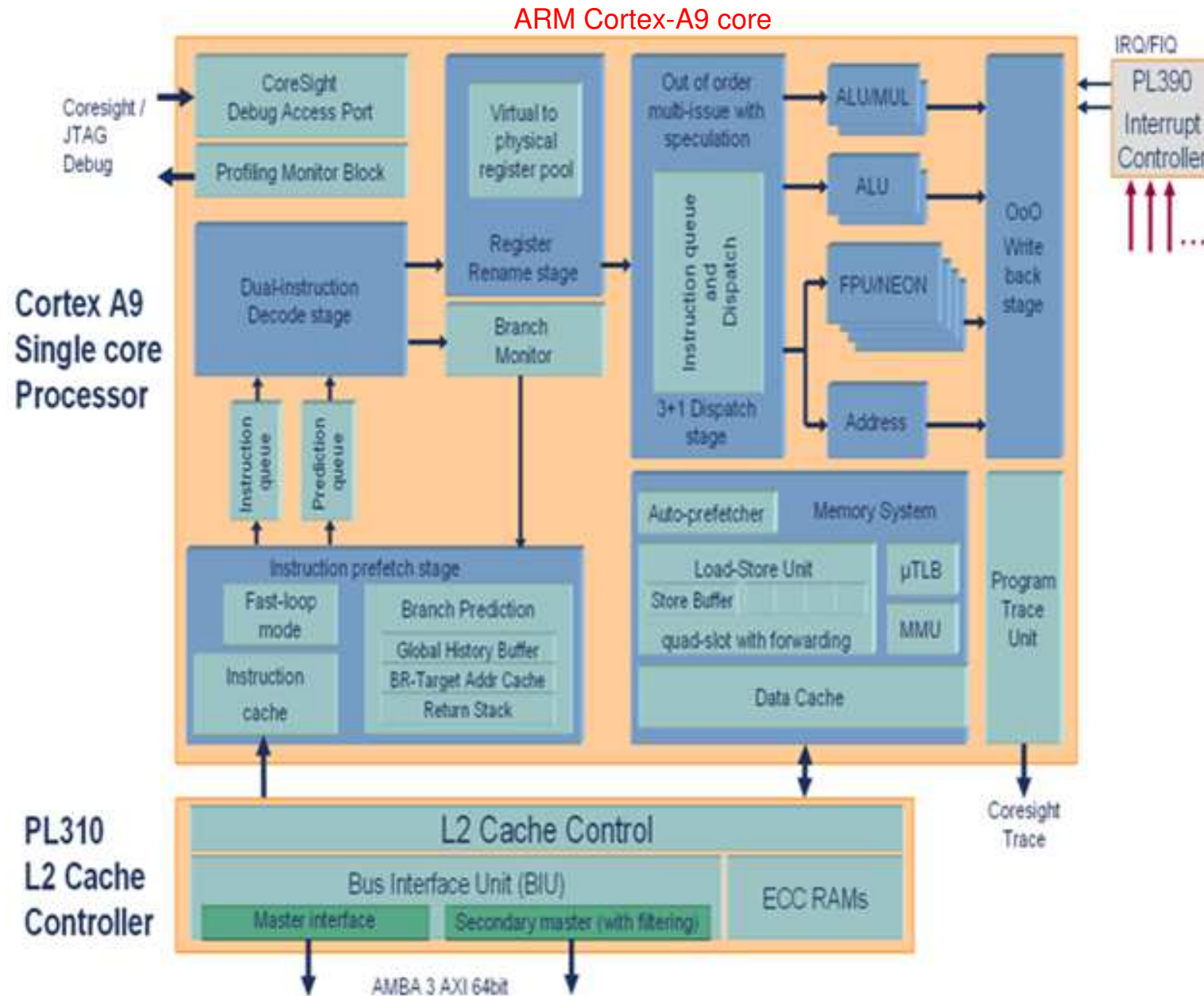
ARM processor architecture I.

- Integrated into Zynq PS side
- ARM Cortex-A9 processor based on **ARMv7-A architecture**
 - ARMv7 - ARM Instruction Set Architecture (ISA)
 - ARMv7-A: Application Memory Management Unit (MMU)
 - Called also as APU (Application Processor Unit)
 - ARMv7-R: Real-time Memory Protection Unit (MPU)
 - ARMv7-M: Microcontroller: M1, M3 (since 2019*, free but license required)
- ARMv7 ISA - instruction set architecture
 - 16 bits/32 bits
 - NEON: ARM's Single-Instruction Multiple-Data (SIMD) extension
- ARM Advanced Microcontroller Bus Architecture (AMBA®) protocol
 - AXI3: v3 ARM „bus“ interface
 - AXI4: v4 (extended bursts)
- „Cortex“ newest processor families: A9, A53, A72, and R5 (radio)

Zynq APSoC architecture



ARM processor architecture II.



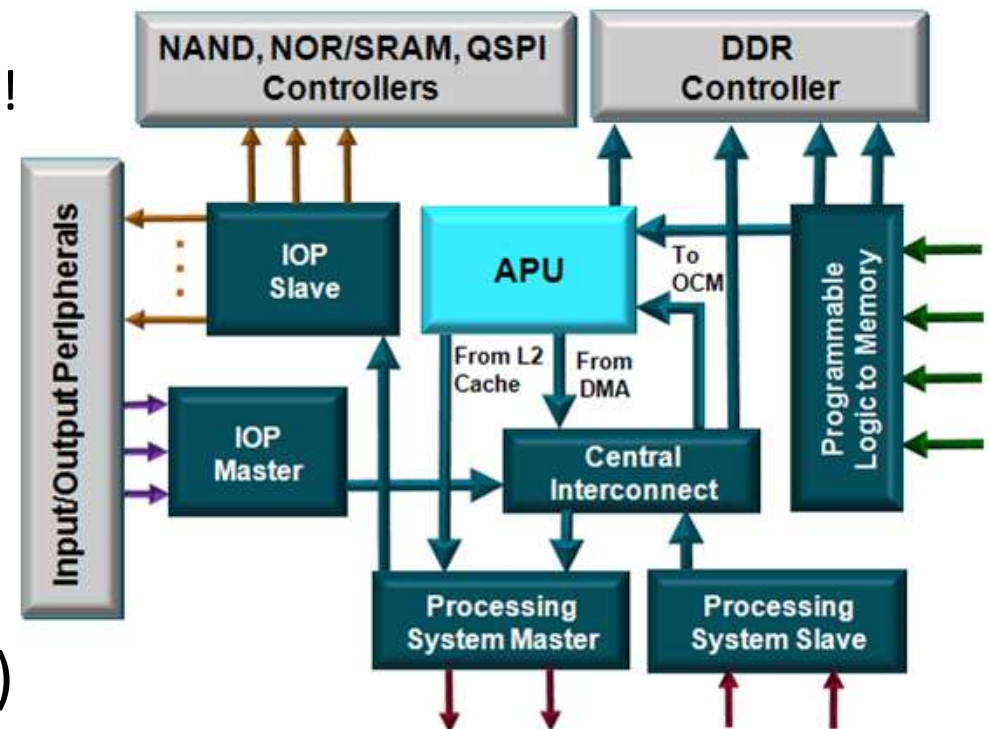
ARM PS (PL) memory map

- The Cortex-A9 processor uses 32-bit addressing (4Gbyte)
- All **PS** peripherals and PL peripherals are memory mapped to the Cortex-A9 processor cores
- All slave **PL** peripherals will be located between 4000_0000 and 7FFF_FFFF (connected to **GP0**) and 8000_0000 and BFFF_FFFF (connected to **GP1**)

FFFC_0000 to FFFF_FFFF	OCM
FD00_0000 to FFFB_FFFF	Reserved
FC00_0000 to FCFF_FFFF	Quad SPI linear address
F8F0_3000 to FBFF_FFFF	Reserved
F890_0000 to F8F0_2FFF	CPU Private registers
F801_0000 to F88F_FFFF	Reserved
F800_1000 to F880_FFFF	PS System registers,
F800_0C00 to F800_0FFF	Reserved
F800_0000 to F800_0BFF	SLCR Registers
E600_0000 to F7FF_FFFF	Reserved
E100_0000 to E5FF_FFFF	SMC Memory
E030_0000 to E0FF_FFFF	Reserved
E000_0000 to E02F_FFFF	IO Peripherals
C000_0000 to DFFF_FFFF	Reserved
8000_0000 to BFFF_FFFF	PL (MAXI_GP1)
4000_0000 to 7FFF_FFFF	PL (MAXI_GP0)
0010_0000 to 3FFF_FFFF	DDR(address not filtered by SCU)
0004_0000 to 000F_FFFF	DDR(address filtered by SCU)
0000_0000 to 0003_FFFF	OCM

Important parts of PS

- Application processing unit (APU)
- I/O peripherals (IOP)
 - Multiplexed I/O (MIO), extended multiplexed I/O (EMIO). Max 54 pin!
- Memory interfaces
- PS interconnect
- DMA
- Timers
 - Public and private
- **G**eneral **i**nterrupt **c**ontroller (GIC)
- **O**n-**c**hip **m**emory (OCM): RAM
- Debug controller: CoreSight

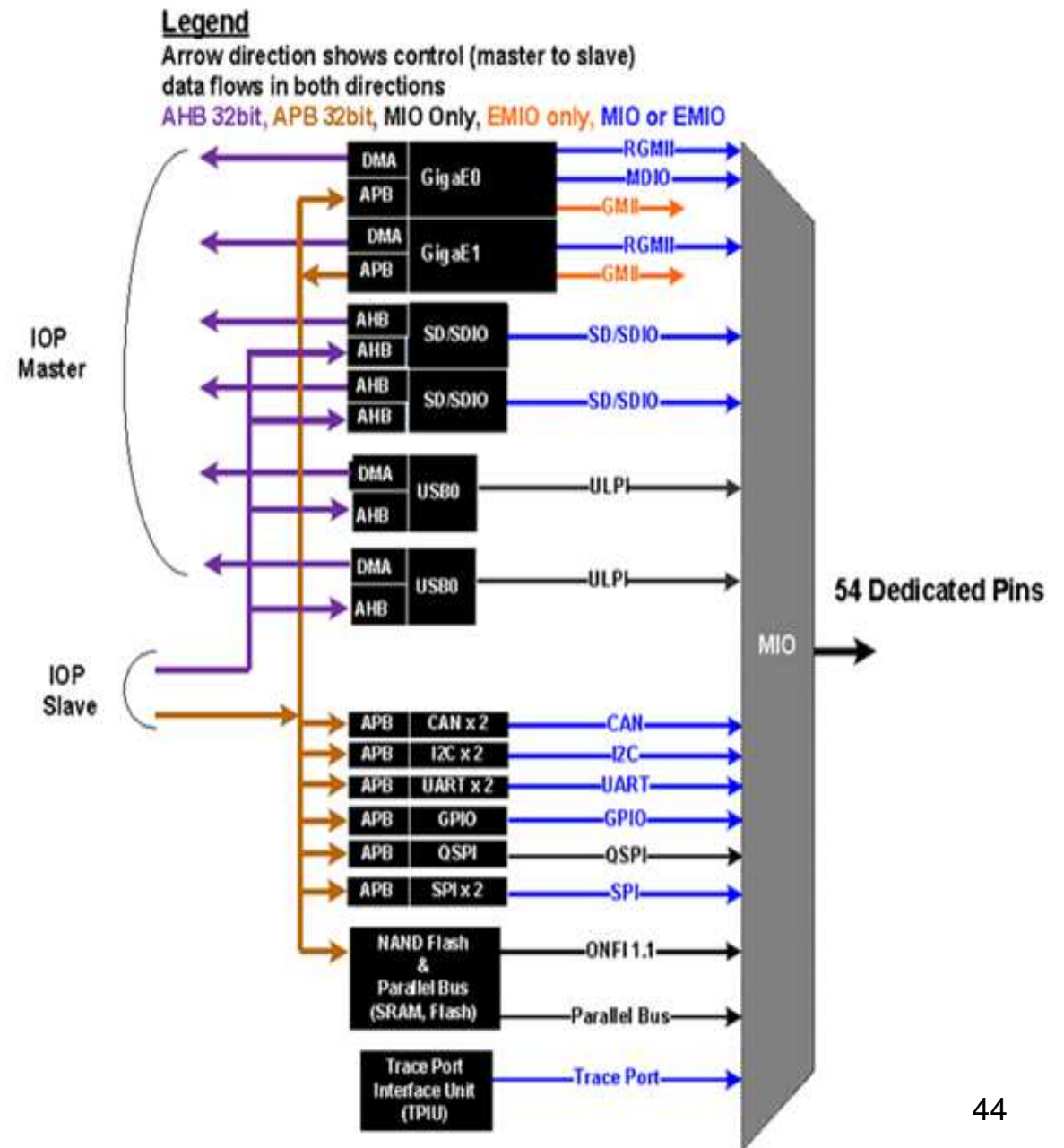


Boot order

- CPU0 boots from OCM ROM; CPU1 goes into a sleep state
- On-chip boot loader in OCM ROM (Stage 0 boot)
- Processor loads **First Stage Boot Loader (FSBL)** from external flash memory:
 - NOR
 - NAND
 - **Quad-SPI**
 - **SD Card**
 - JTAG; not a memory device—used for debug only
 - (Boot source selected via package bootstrapping pins)
- Optional secure boot mode allows the loading of encrypted software from the flash boot memory

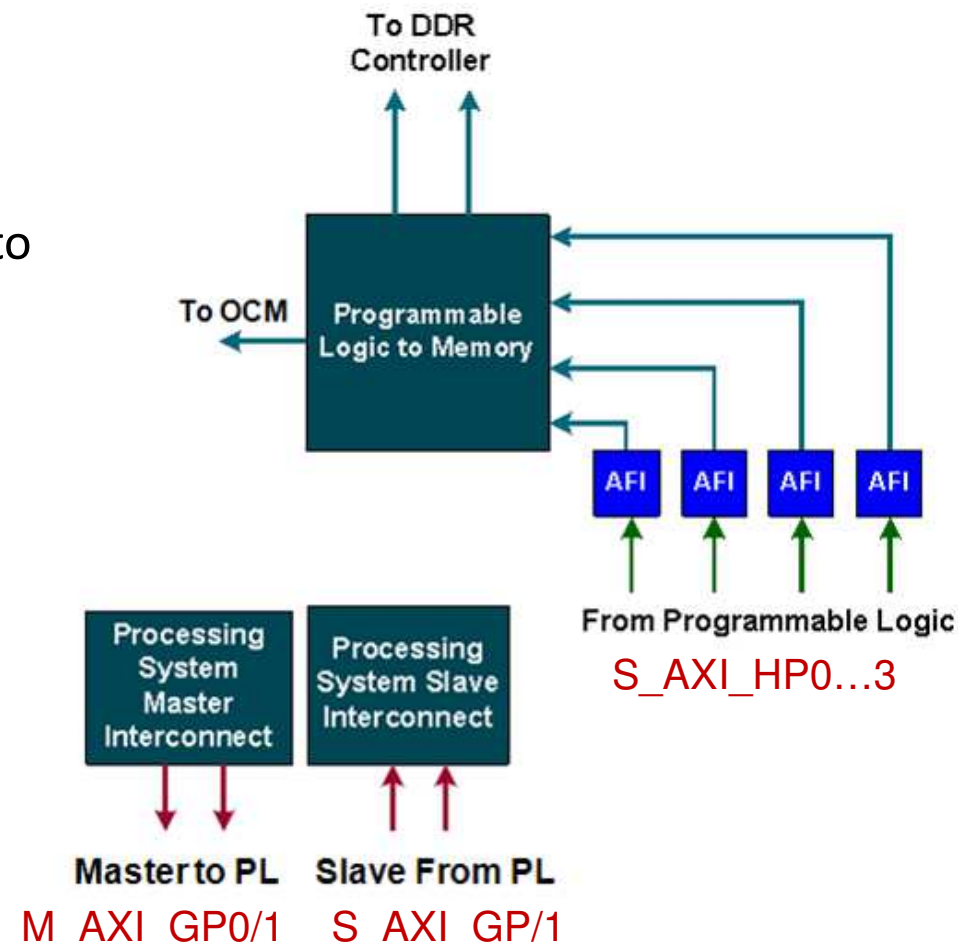
IOP – IO peripherals

- 2 GigE
- 2 USB
- 2 SPI
- 2 SD/SDIO
- 2 CAN
- 2 I2C
- 2 UART
- 4 db 32-bit GPIOs
 - NAND, NOR/SRAM,
 - Quad SPI
- Trace ports



PS-PL interconnections („bridges“)

- **AXI high-performance slave portok (HP0-HP3)**
 - 4 Configurable 32-bit or 64-bit data width
 - Access to OCM (1) and DDR (2) only
 - Conversion to processing system clock domain
 - AXI FIFO Interface (AFI) are FIFOs (1KB) to smooth large data transfers
- **AXI general-purpose portok (GP0-GP1)**
 - 2masters from PS to PL
 - 2slaves from PL to PS
 - 32-bit data width
 - Conversation and sync to processing system clock domain



BUS INTERFACES

General description

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok



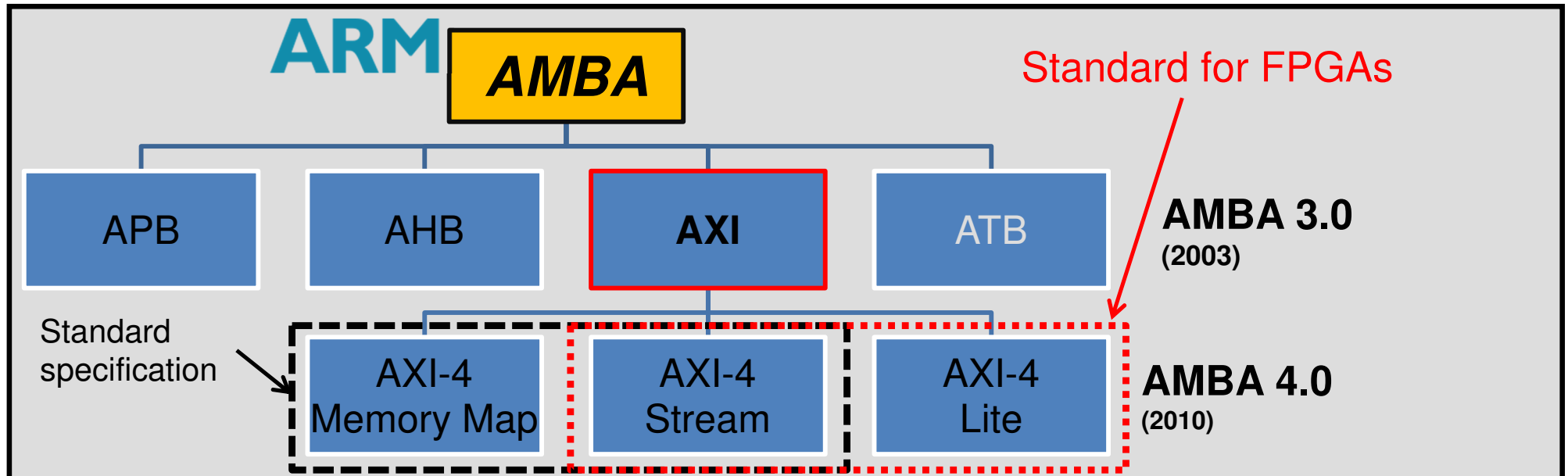
BEFEKTETÉS A JÖVŐBE

Buses – „1x1”

- Bus master (**M**): initiate a bus transaction (R/W)
- Bus slave (**S**): only responds to a bus request
- Bus arbitration (**A**): 3-step decision process
 - The master (M) device sends its bus reservation request to the Arbitrator (A) via the BR (bus request) line for the next transaction
 - The Arbiter (A) continuously monitors (polls) the demands and it sends a BG (bus grant) signal to the master unit (if its priority was the highest from several Masters at the same time)
 - The bus requesting master (M) unit receives the BG signal and then initiates the **transaction** (read-write) to the slave (S) unit when the previous master (M) unit finished its current transaction
- **Arbitration** = decision mechanism. Who should be the Master for the next transaction? The decision can be based on:
 - Fixed-priority, or FIFOed
 - Round-robin, or
 - Hybrid method

ARM AMBA – AXI interface

- AMBA: Advanced Microcontroller Bus Architecture



Interfész	Tulajdonság	Hasonló busz
AXI4-MM: Memory Mapped / Full	Traditional address/data Burst – „löket” (SAMD = single address → multiple data). Burst: max 256 unit	PLBv46, PCI
AXI4-Streaming	No address, only data transfer — Burst-like (unlimited burst)	Local Link / DSP interfaces / FIFO / FSL
AXI4-Lite	Hagyományos cím/adat — Non burst-like (SASD = single address → single data) Data can be 32 / 64 bits *	PLBv46-single (deprecated) OPB (deprecated)

AXI – Advanced Extensible Interface

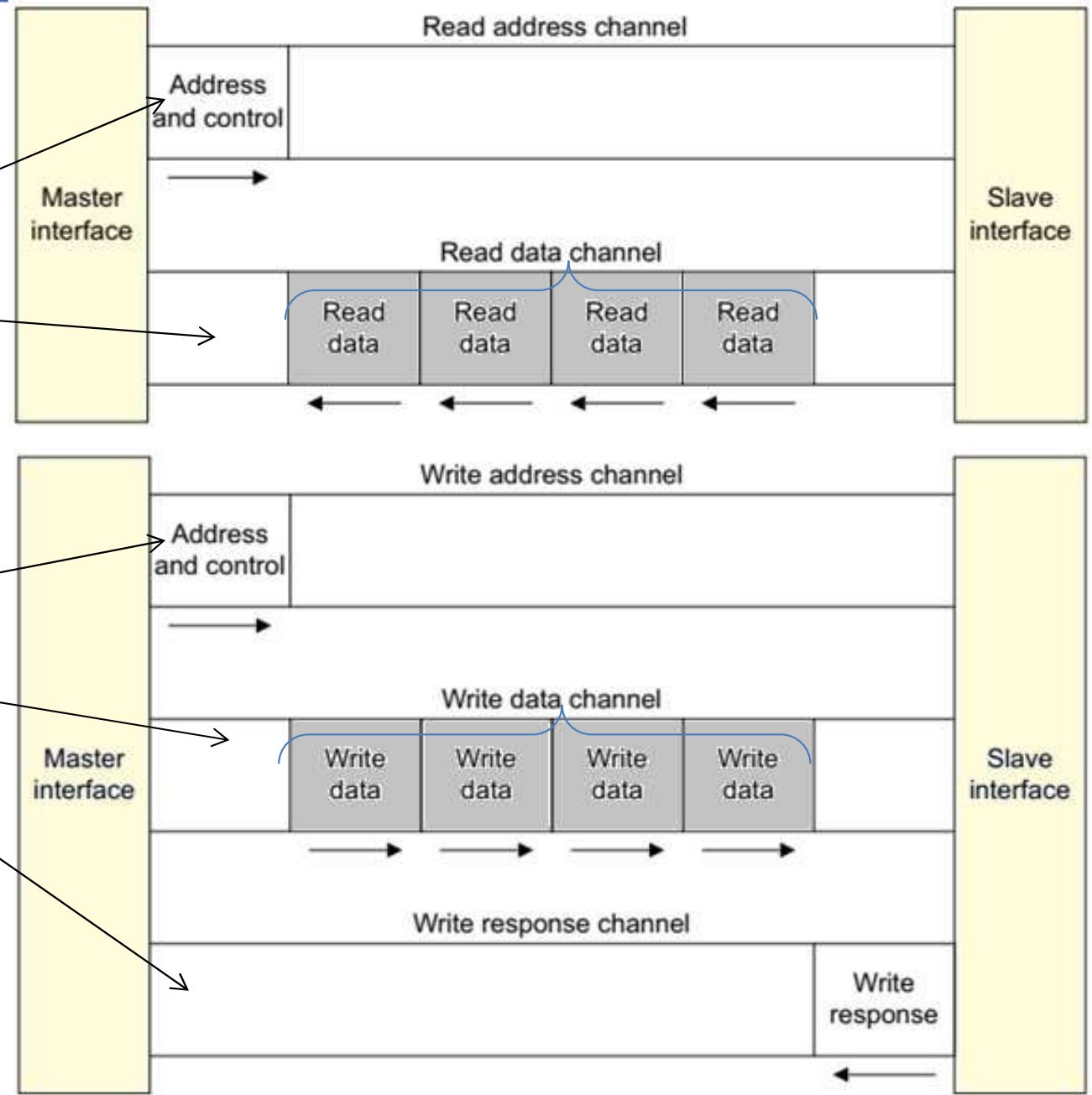
- Standard **interface** and **protocol** definition,
- Widely used,
- Not considered to a „bus” connection!
- The AXI specifications define it more as an „**interface**” (rather than a bus connection!):
 - It does not specify how to connect the IP peripherals in the system, but
 - Defines the inputs and outputs of an IP peripheral.

AXI4 basic mode (memory mapped)

5 communication channels:

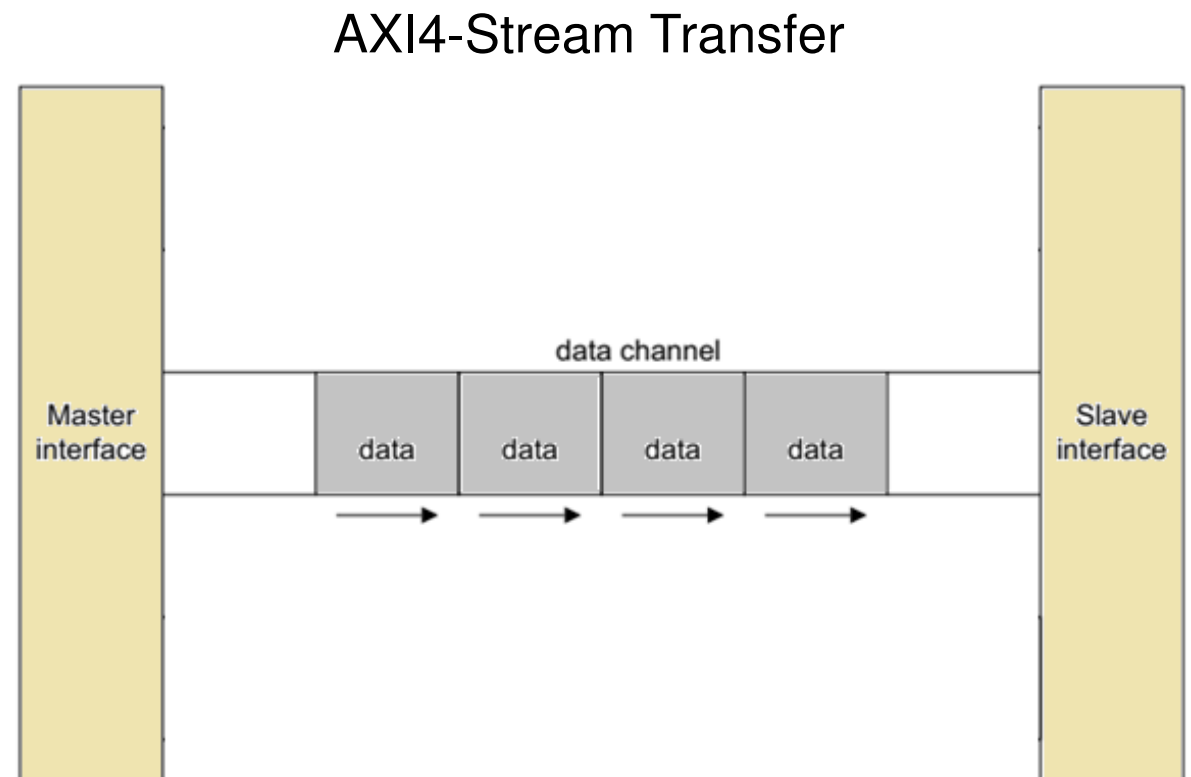
1. Read Address Channel
2. Read Data Channel

3. Write Address Channel
4. Write Data Channel
5. Write Response Channel



AXI4 Stream mode

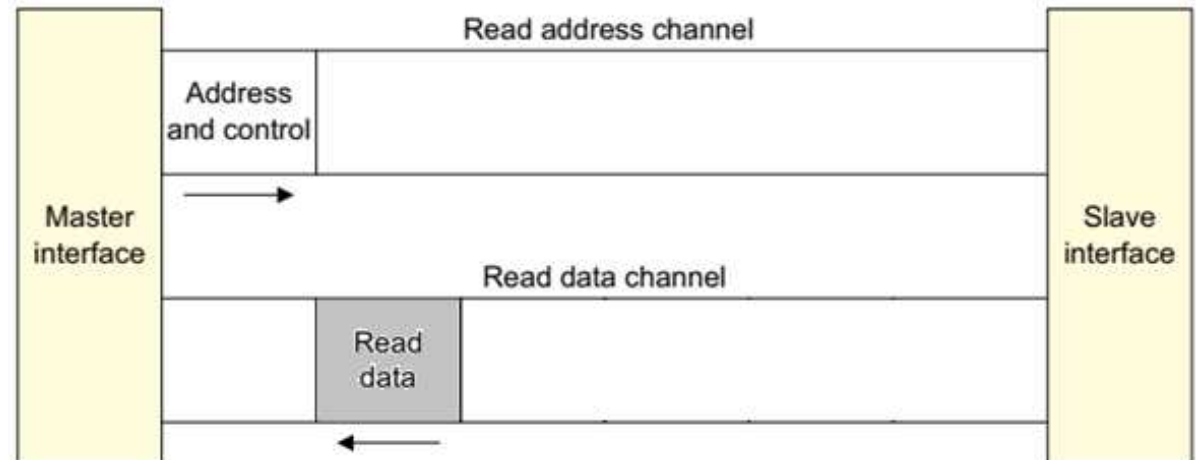
- **No address channel**, no read and write, always just master to slave
 - Effectively an AXI4 “write data” channel
- Unlimited burst length
 - AXI4 max 256
 - AXI4-Lite does not burst
- Virtually same signaling as AXI Data Channels
 - Protocol allows merging, packing, width conversion
 - Supports sparse, continuous, aligned, unaligned streams



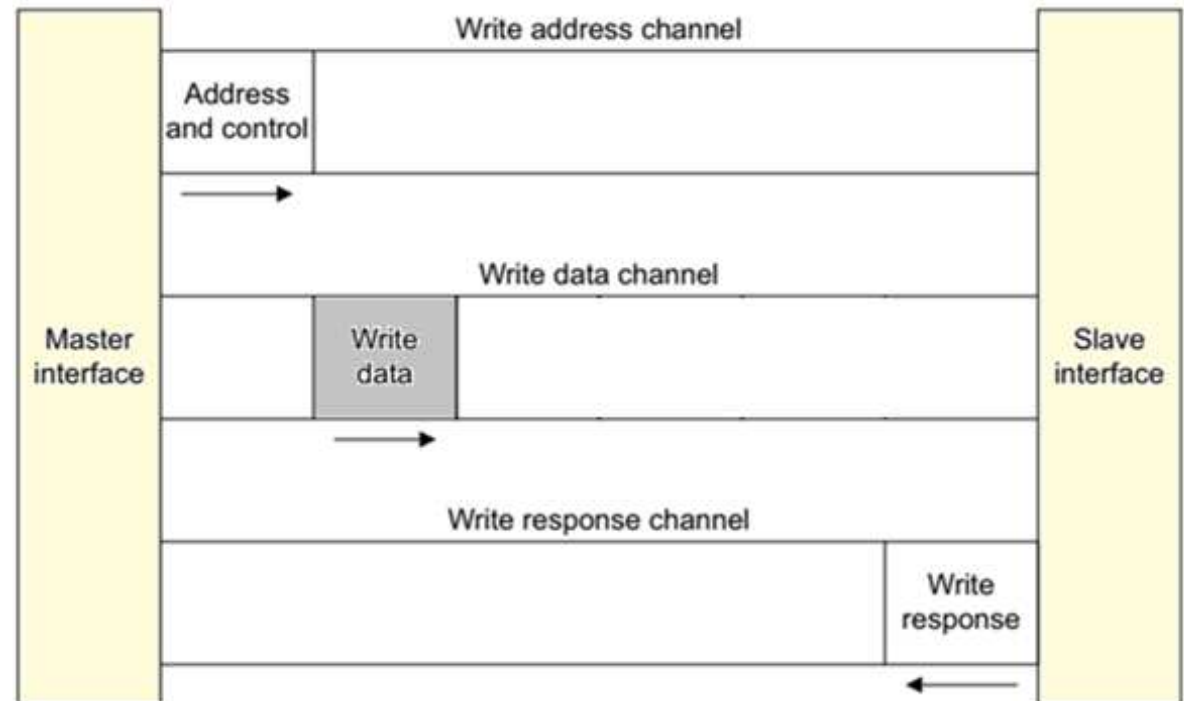
AXI4 Lite mode

- No burst
 - Ex. pass control information
- Data width 32 or 64 only
 - Xilinx IP only supports 32-bits
- Very small footprint
- Bridging to AXI4 handled automatically by AXI_Interconnect (if needed)

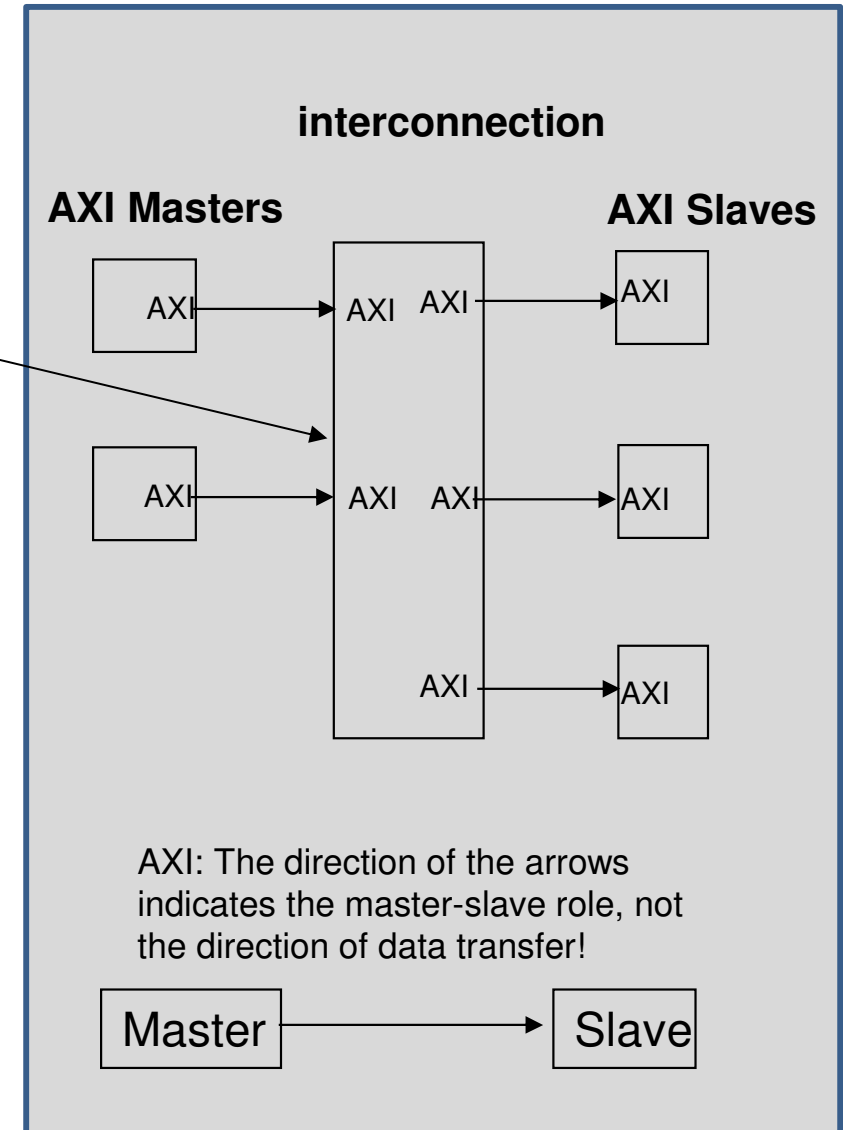
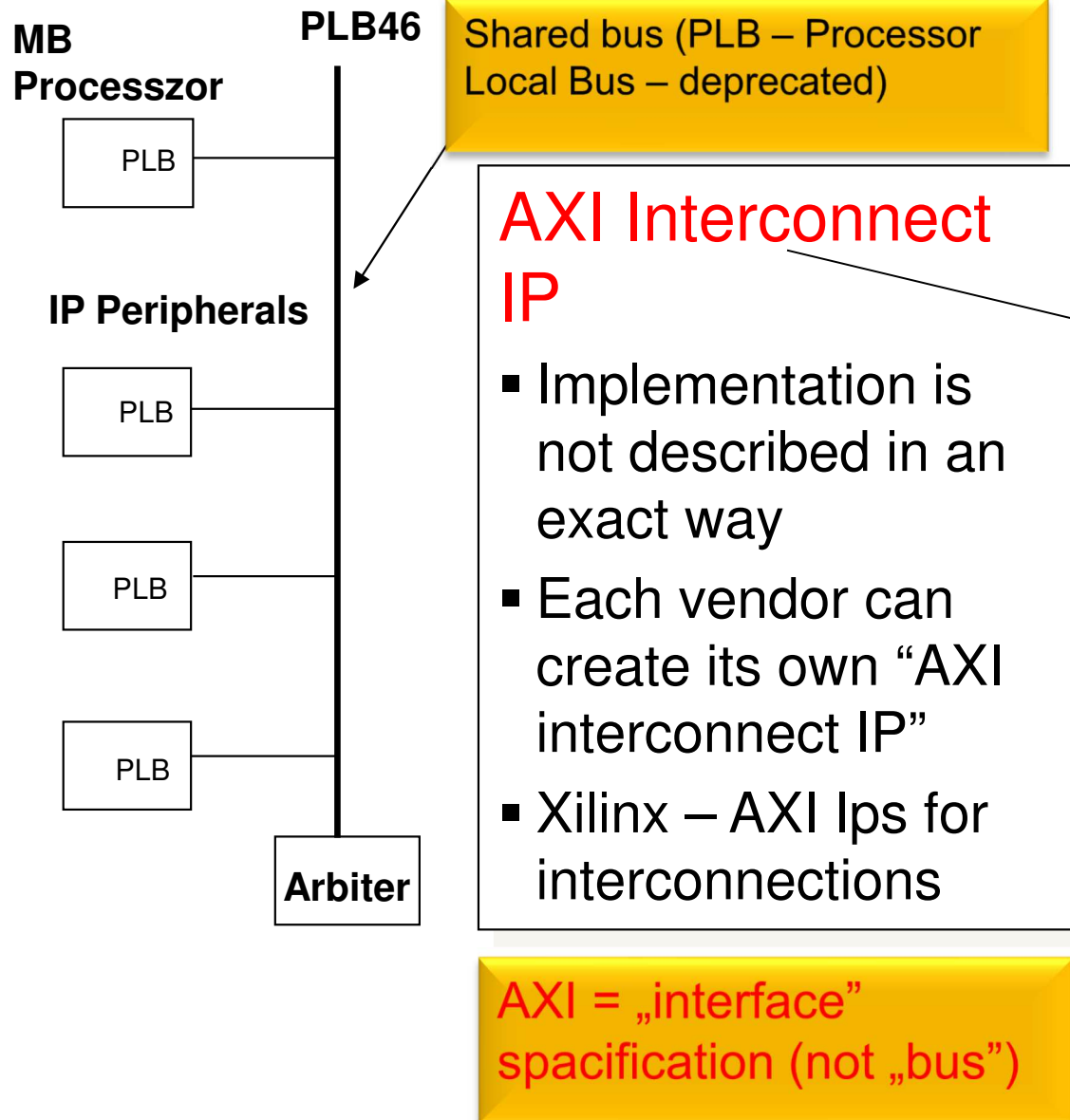
AXI4-Lite Read



AXI4-Lite Write



AXI as an interface



LMB – Local Memory Bus

- The **LMB** provides data / instruction access to the MicroBlaze processor @ 1 clock cycle
 - on-chip, dual-port memories (BRAMs)
 - own dedicated data / instruction bus
- Simple synchronous protocol
 - **DLMB**: Data interface LMB (BRAM only)
 - **ILMB**: Instruction interface LMB (BRAM only)

Summary – buses & interfaces

- **MicroBlaze/ARM** : AXI interface supported in newest Vivado/VITIS development environment
 - **AXI**: interface for high-speed, point-to-point interconnection
 - MicroBlaze:
 - **AXI**: supports both Little and Big-endian format
- **LMB**: processor's local memory bus (for data, stack and interrupt handler routines) – only MicroBlaze processors
 - Fix and deterministic latency

EMBEDDED SW DEVELOPMENT ENVIRONMENT (XILINX VITIS)

Brief description

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok

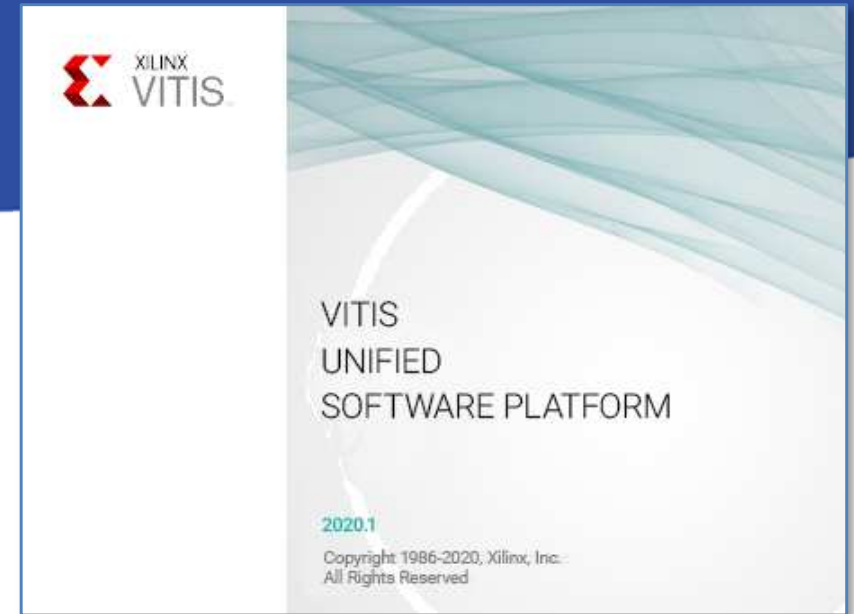


BEFEKTETÉS A JÖVŐBE

VITIS - SDK

Key features:

- Full-featured IDE for SW dev.
 - Eclipse-based, open-source
- Independently works from Vivado
 - Embedded SW development
- Support Embedded HW/FW system debug
 - Multi-processor platform (MPSoC/APSoC)
 - Multi-BSP / domain SW systems
 - Multi-SW application
- Efficient C/C++ source editor and debugger (navigator)



*Utilization of VITIS SDK will be introduced in Laboratory-04

VITIS SDK

- **Builder module:**
 - Compiling and linking source files (linker script: **.LD**)
 - Generating compilation settings (option) for embedded SW application: select between Debug, Release and Profile configurations
 - These settings can be changed, refined later (custom build)
 - Compilation types: Standard Make, Managed Make (**makefile.**)
- **Run module:**
 - Application run (Run /+ Profile) vs. Debug
 - Target Connection settings
 - Executable/Run file: **.ELF + Executable Load File**
- **Debug module:**
 - GNU debugger (gdb), loading application, and HW debugging
 - Information about the status of debugging (Debug view)
- **Search module/Help module:**
 - Help in application development

VITIS SDK GUI

Project Explorer (3 level):

- HW Platform (design_1_wrapper)
- BSP: Board Support Package
- Application project

Source editor

Debug window

The screenshot displays the Vitis IDE interface with the following components:

- Project Explorer (left):** Shows a hierarchical view of the project structure. The 'MemTest' system is expanded, showing its components: 'Binaries', 'Includes', 'Debug', and 'src'. The 'src' folder contains files like 'memory_config_g.c', 'memory_config.h', 'memorytest.c', 'platform_config.h', 'platform.c', 'platform.h', 'lscript.ld', and 'Xilinx.sner'. Below this, the 'Assistant' pane shows the project hierarchy: 'App_project_system [System]' containing 'App_project [Application]', 'MemTest_system [System]' containing 'MemTest [Application]', and 'design_1_wrapper [Platform]'.
- Source Editor (center):** Displays the 'memorytest.c' file. The code includes headers, defines, and functions for testing memory regions (32-bit, 16-bit, and 8-bit). It uses 'Xil_TestMem' functions and prints status messages like 'PASSED!' or 'FAILED!'.
- Outline (right):** Shows a list of files and functions in the current project, including 'stdio.h', 'xparameters.h', 'xil_types.h', 'xstatus.h', 'xil_testmem.h', 'platform.h', 'memory_config.h', 'xil_printf.h', 'putnum(unsigned int) : void', 'test_memory_range(struct r', and 'main() : int'.
- Console / Message window (bottom):** Shows the build output for 'MemTest, Debug'. It includes the command 'Invoking: ARM v7 Print Size', the output of 'arm-none-eabi-size MemTest.elf', and the final message 'Finished building: MemTest.elf.size'. The build finished at 13:54:03 and took 25.22ms.

At the bottom of the IDE, there are tabs for 'Writable', 'Smart Insert', and '100 : 1'.

Console / Message window



EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének
együttes javítása a Pannon Egyetemen

THANK YOU FOR YOUR KIND ATTENTION!

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Strukturális
és Beruházási Alapok



BEFEKTETÉS A JÖVŐBE