Dr. Kincses Zoltán, Dr. Vörösházi Zsolt:

Beágyazott rendszerek fejlesztése

A felsőfokú informatikai oktatás minőségének fejlesztése, modernizációja

TÁMOP-4.1.2.A/1-11/1-2011-0104

Főkedvezményezett:
Pannon Egyetem
8200 Veszprém
Egyetem u. 10.

Kedvezményezett:
Szegedi Tudományegyetem
6720 Szeged
Dugonics tér 13.

2014

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Frissítve: 2018. november 24.

Beágyazott rendszerek fejlesztése (FPGA)

Dr. Vörösházi Zsolt

6. XADC – Xilinx Analog/Digital Converter beépítése és tesztelése

Tárgyalt ismeretkörök

1. Bevezetés – Beágyazott rendszerek, FPGA-k, Digilent ZYBO fejlesztő kártyák és eszközök
2. Beágyazott Rendszer fejlesztő szoftverkörnyezet (Xilinx Vivado Embedded Development) áttekintése
3. Beágyazott alap tesztrendszer (BSB - Base System Builder and Board Bring-Up) összeállítása
4. Perifériák hozzáadása (IP adatbázisból) az összeállított beágyazott alaprendszerhez. Saját periféria hozzáadása az összeállított beágyazott alaprendszerhez
5. Szoftver alkalmazások fejlesztése, tesztelése: PMOD TMP hőmérséklet mérő
6. **XADC – Xilinx AD converter, mint System Monitor.**

Irodalomjegyzék

- 📖 XADC - Xilinx Analog-Digital Converter:

https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf

- 📖 Vivado – AXI XADC IP leírása

https://www.xilinx.com/products/intellectual-property/axi_xadc.html

- 📖 XADC Wizard (paraméterezhetőség):

https://www.xilinx.com/support/documentation/ip_documentation/xadc_wiz/v3_3/pg091-xadc-wiz.pdf

- 📖 XADC Use-cases:

https://www.xilinx.com/support/documentation/application_notes/xapp795-driving-xadc.pdf

- 📖 Adam Taylor – MicroZed Chronicles Part 7-8:

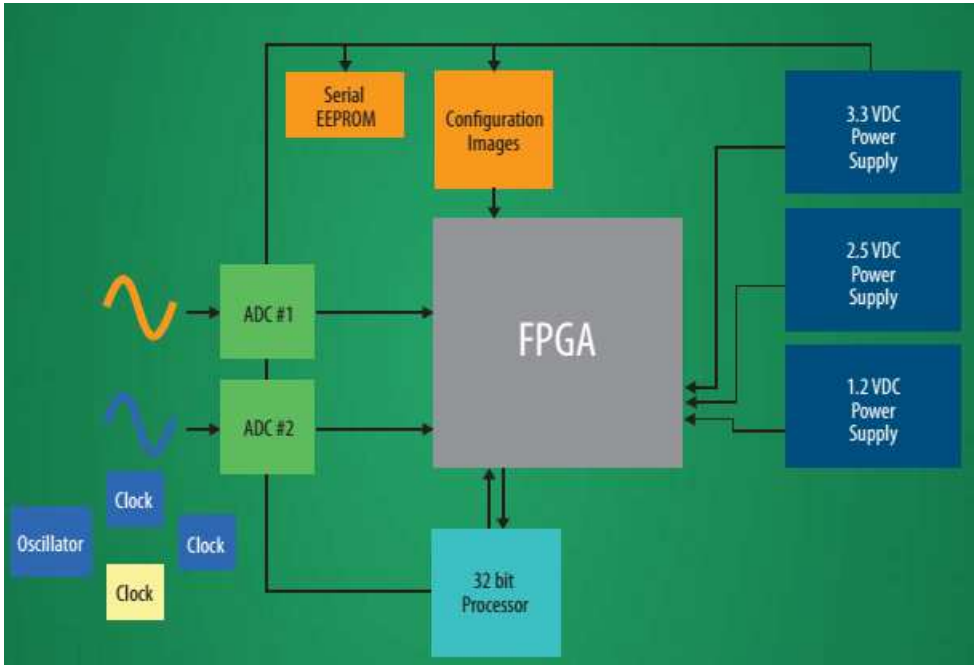
Part7: https://forums.xilinx.com/t5/Xcell-Daily-Blog/Getting-the-XADC-Running-on-the-MicroZed-Adam-Taylor-s-MicroZed/ba-p/380989

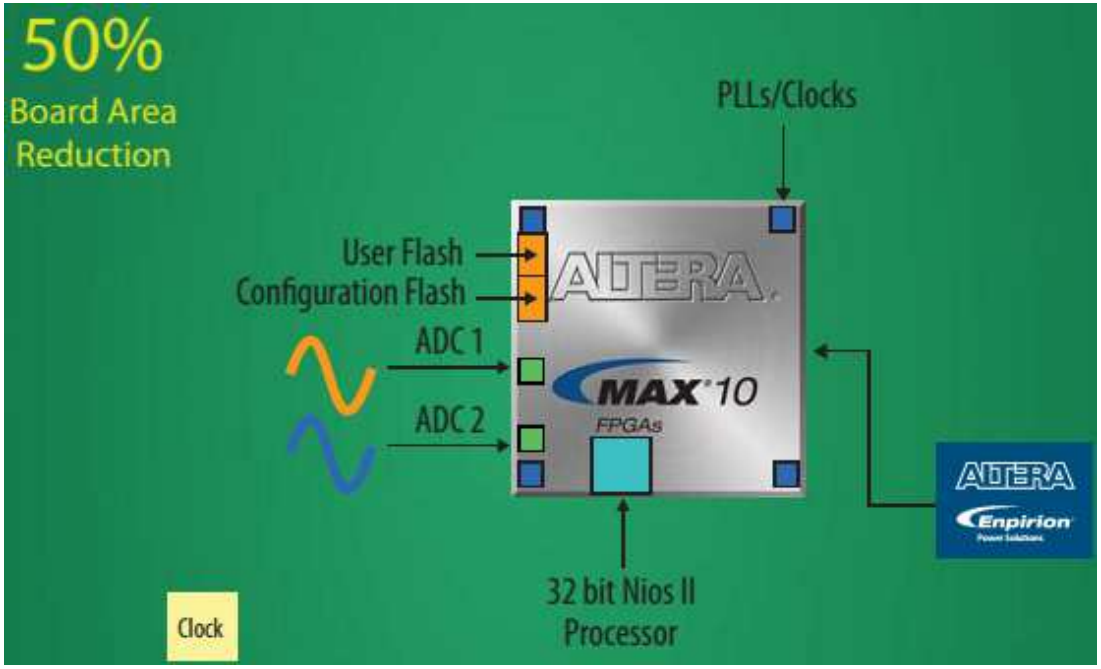
Part-8: https://forums.xilinx.com/t5/Xcell-Daily-Blog/MicroZed-XADC-Software-Adam-Taylor-s-MicroZed-Chronicles-Part-8/ba-p/383861

ADC – Analóg/Digitális konverzió

- Milyen lehetőségek vannak analóg jelek feldolgozására FPGA-val?
  - Külső ADC (FPGA-s fejlesztő kártyán nincs): **hagyományosan** egy külső bővítő kártyán lévő funkció
    - Minden esetben megvalósítható
  - **Belső-integrált** ADC (FPGA-n belüli „on chip” funkció)
    - Csak bizonyos FPGA-kon, ill. CPLD-ken
      - Pl. Xilinx XADC IP core (Xilinx 7-es sorozatokon)

Hagyományos vs. integrált megoldás

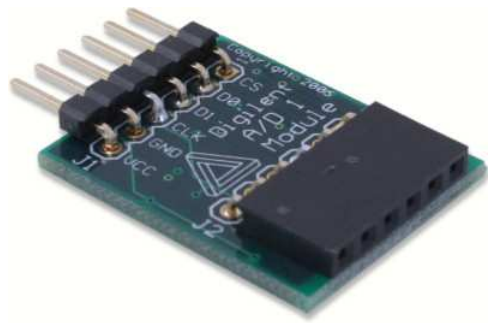




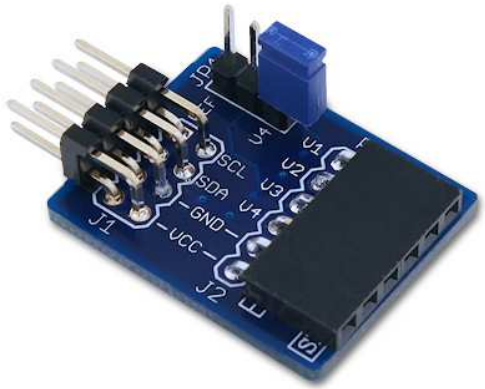
Hagyományos: ADC kártya + FPGA

pl. Digilent PMOD ADs

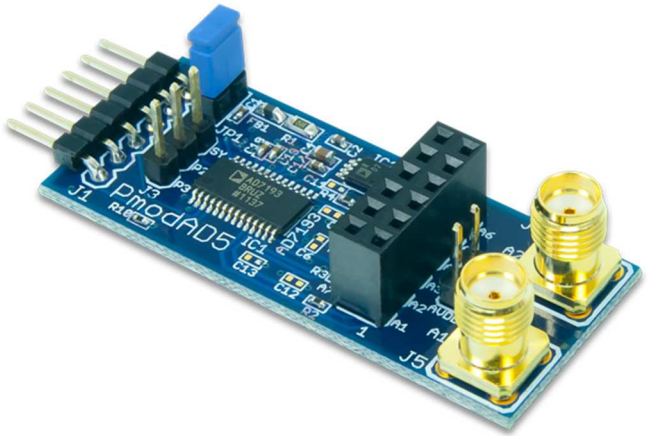
- **PDMOD AD1:** 2 csatornás Analog Devices AD7476 12-Bit, 1 MSPS, Low-Power A/D Converter



- **PDMOD AD2:** 2 db Analog Devices AD7476 12-Bit, 1 MSPS, Low-Power A/D Converter (+I2C)

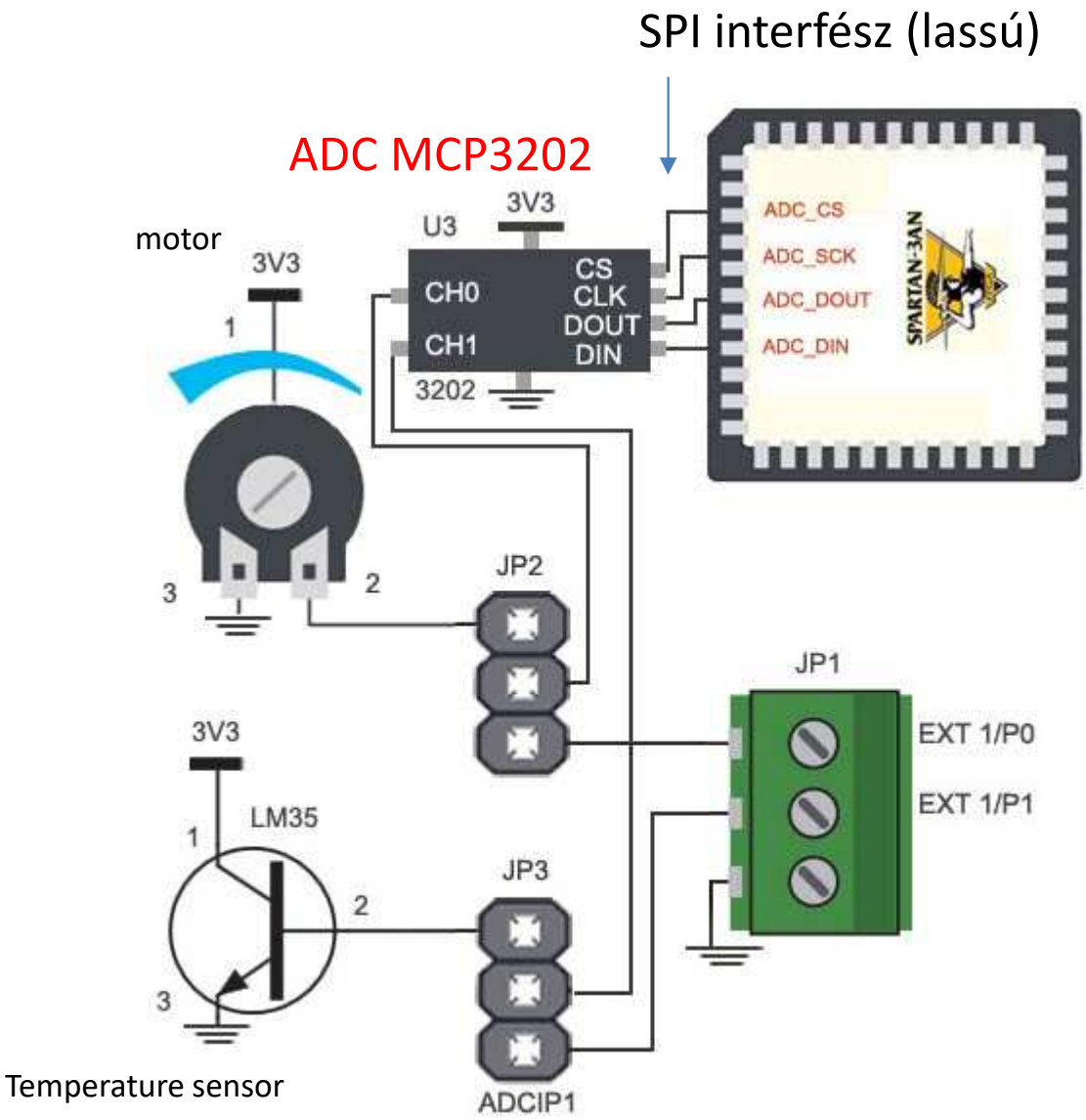


- **PDMOD AD5:** Analog Devices AD7991 4-csatornás, 24-Bit ADC, 1MSPS

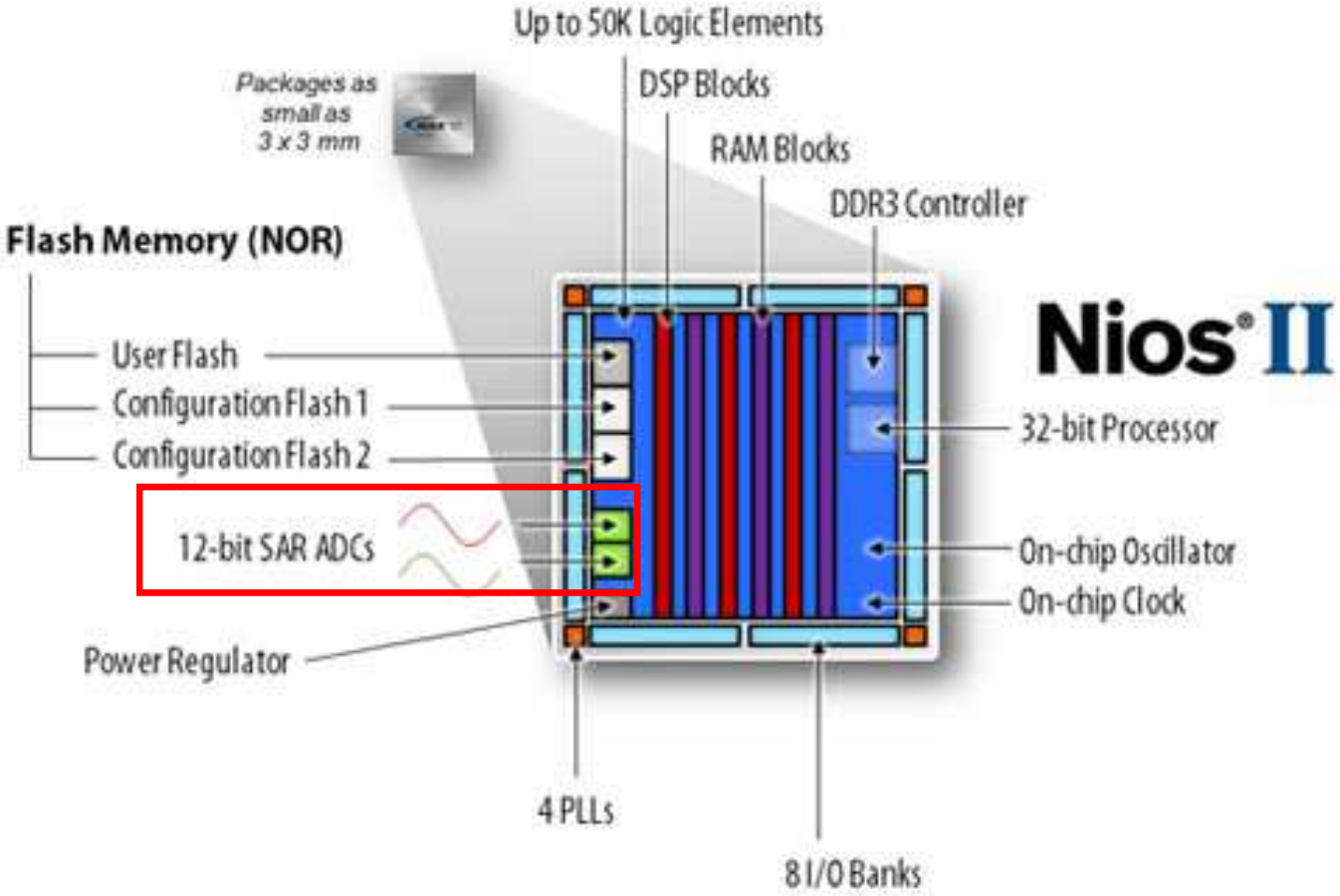


SPI ADC – Xilinx Spartan3-AN

- https://www.pantechsolutions.net/project-kits/how-to-interface-spi-adc-with-spartan-3an



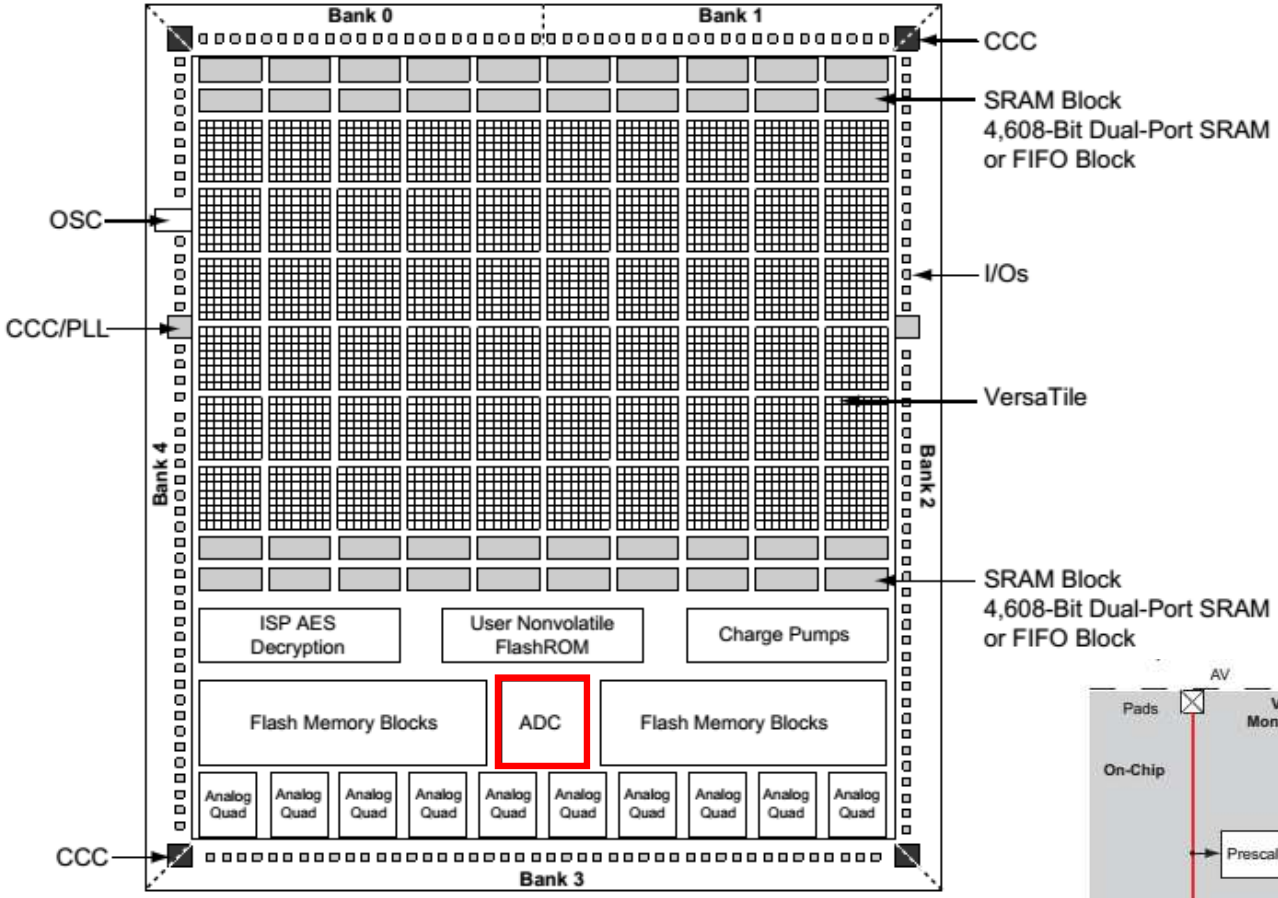
Altera MAX 10 FPGA

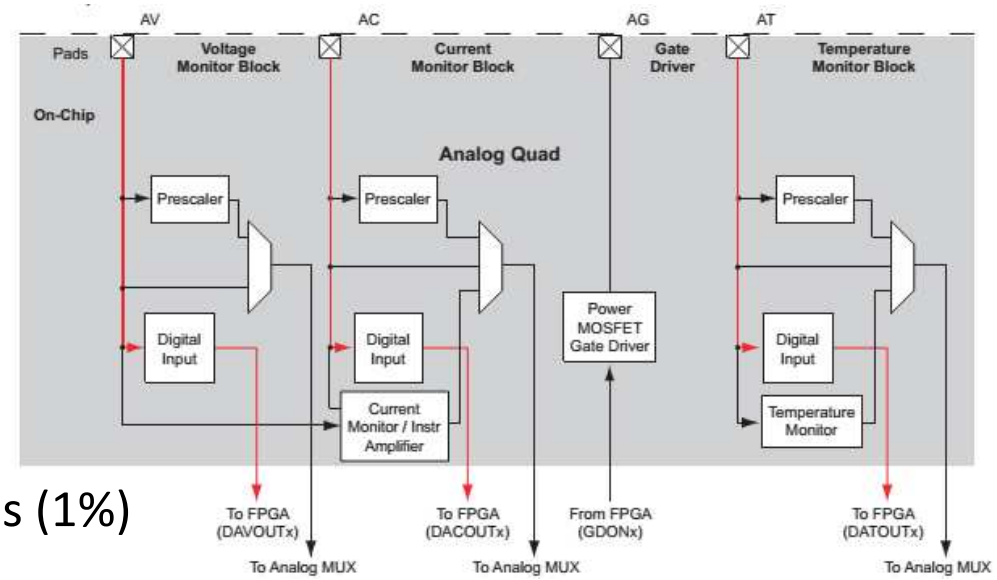


2 db beágyazott SAR ADC – 12-bit, 1 MSPs

- max. 18 analóg bemeneti csatorna
- hőmérséklet mérő szenzor

MicroSemi Fusion (mixed-signal FPGA)



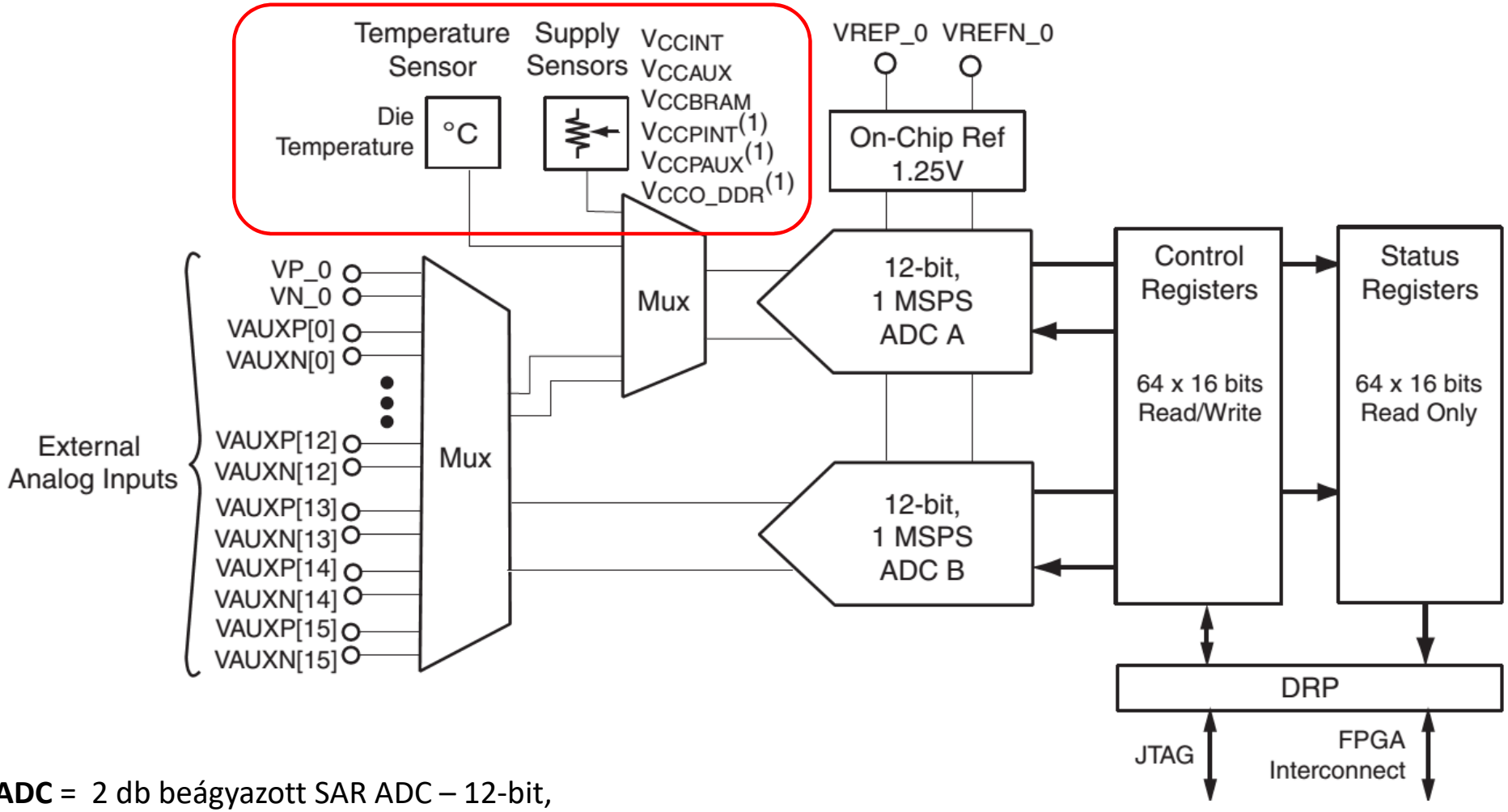


Beágyazott SAR ADCs – 8/10/12-bit, max. 0.6 Msps (1%)

- max. 30 analóg bemeneti csatorna
- Analóg quad: feszültség, áram-, hőmérséklet monitor

Xilinx-”7” FPGA-k - XADC

(Artix-7, Kintex-7, Virtex-7, Zynq 7000)



1 **XADC** = 2 db beágyazott SAR ADC – 12-bit, 1 MSPs (1%)

- max. 17 analóg bemeneti csatorna
- hőmérséklet mérő szenzor

Xilinx Vivado használata

XADC HOZZÁADÁSA A BEÁGYAZOTT RENDSZERHEZ

Mit is jelent az XADC?

- **Xilinx Analóg-Digitális Konverter** blokk
  - FPGA/APSoC hőmérséklet, feszültség szintek mérése
  - Alarm funkciók – vészjelzések beállítható intervallumokon
  - **AXI4-Lite**/Stream interfész v. felhasználói interfész
  - Könnyű konfigurálhatóság (Wizard ~ GUI)
  - Több csatorna kiválasztása/használata

XADC – mint System Monitor

Miket képes mérni? „Mindent az FPGA/APSoC állapotáról”.

- Hőmérséklet: On-chip temperature sensor

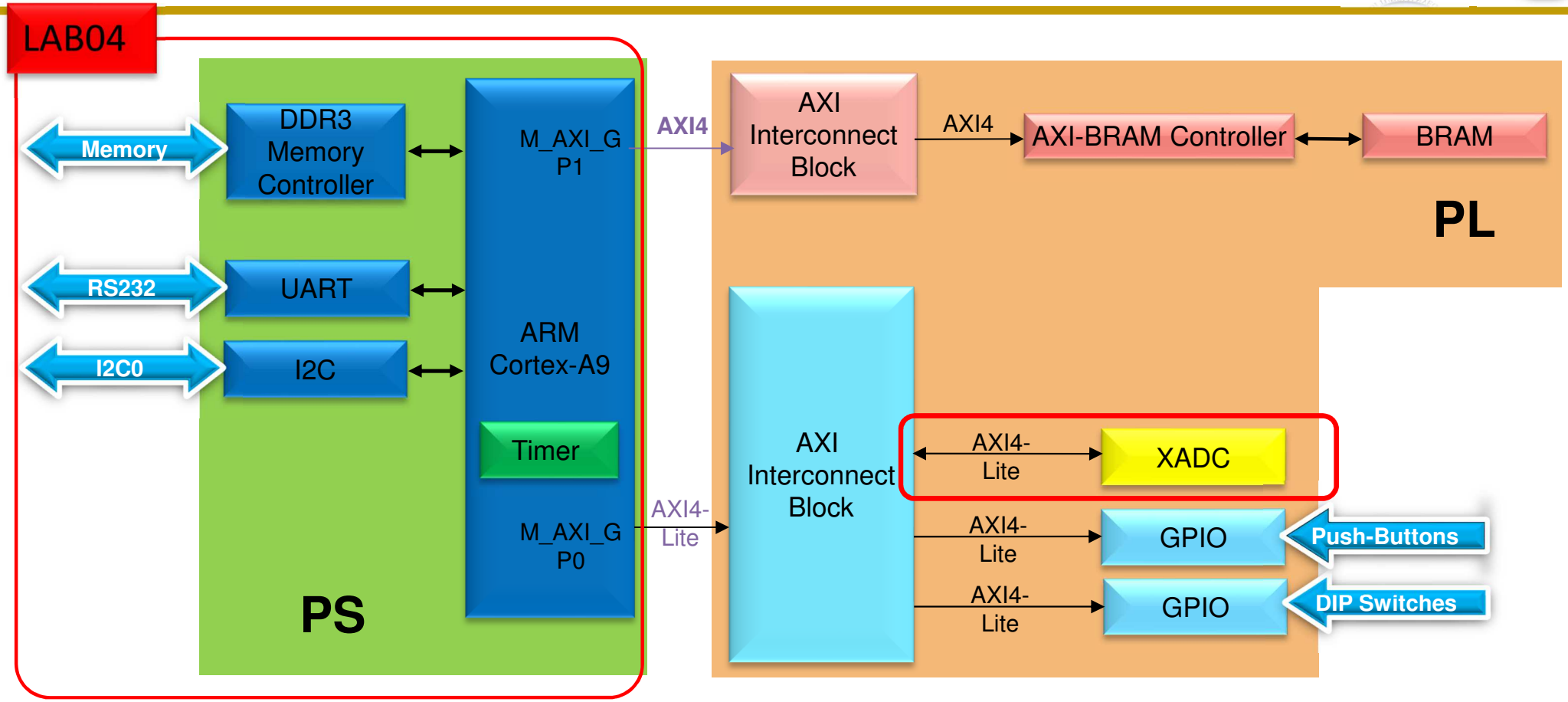
- VCCInt: The internal PL core voltage
- VCCAux: The auxiliary PL voltage
- VRefP: The XADC positive reference voltage
- VRefN:The XADC negative reference voltage
- VCCBram: The PL BRAM voltage

- VCCPInt: The PS internal core voltage
- VCCPAux: The PS auxiliary voltage
- VCCDdr: The operating voltage of the DDR RAM connected to the PS

A feladat megoldásának lépései

- A korábbi (5. fólia) ismeretkör elsajátítása során létrehozott projekt archiválása (***Save Project as…***):
  - `LAB03` → `LAB04` (majd nyissuk is meg)
- XADC IP hozzáadása a XADC IP Wizard segítségével a Xilinx Vivado-ban,
- XADC felparaméterezése és összekapcsolása a beágyazott rendszerrel.

A bővítendő tesztrendszer



PS oldal:

- ARM hard-processzor mag
- belső OnChip-RAM vezérlő
- RS232 soros interfész
- külső DDR3 memória vezérlő
- I2C vezérlő

PL oldal:

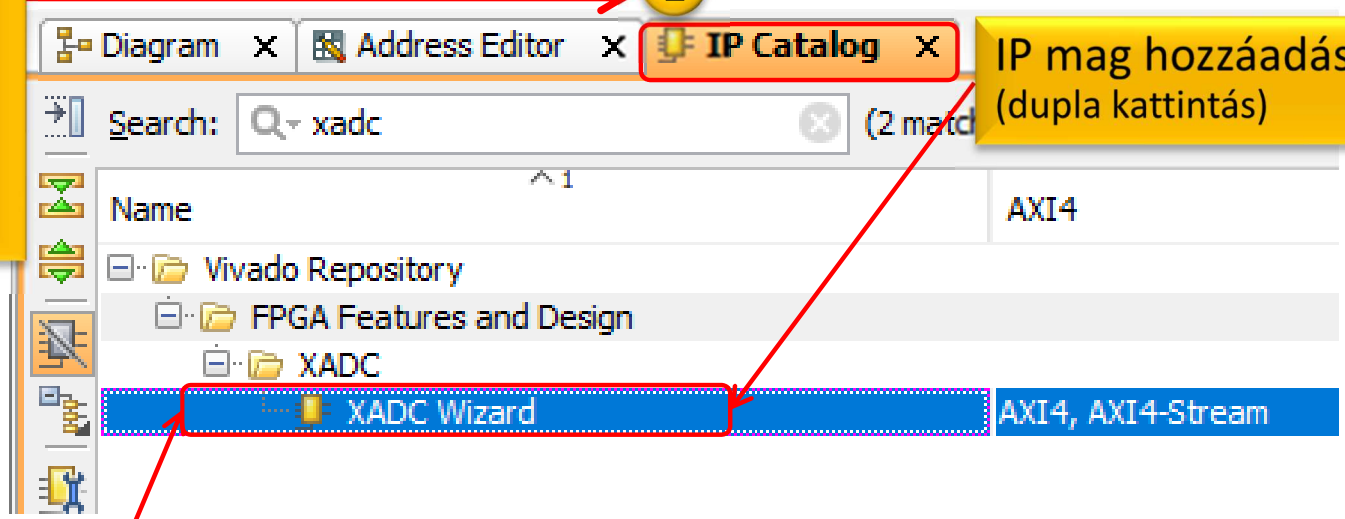
- LAB04: XADC IP hozzáadása

PL oldali XADC IP hozzáadása és összekötése az alaprendszerrel I.

- Ezután már az új IP mag hozzáadható a Vivado-ban (két lehetőség):
  - a.) Block Diagram View -> Add IP
  - b.) IP katalógus megnyitása –> IP kiválasztása –> Dupla kattintással –> Add IP to Block Design
- Adjunk a processzor rendszerhez egy PL oldali **XADC** blokkot

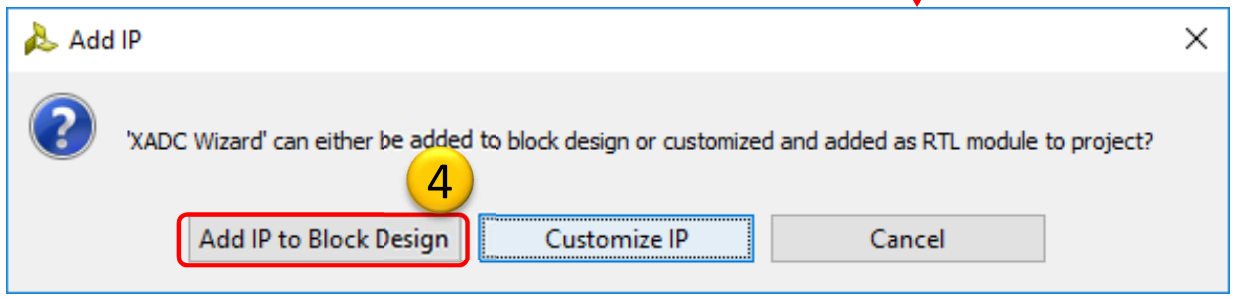
Váltás a az *IP katalógus* nézetre, majd szűrés az „xadc”-re

1



IP mag hozzáadása (dupla kattintás)

3



2

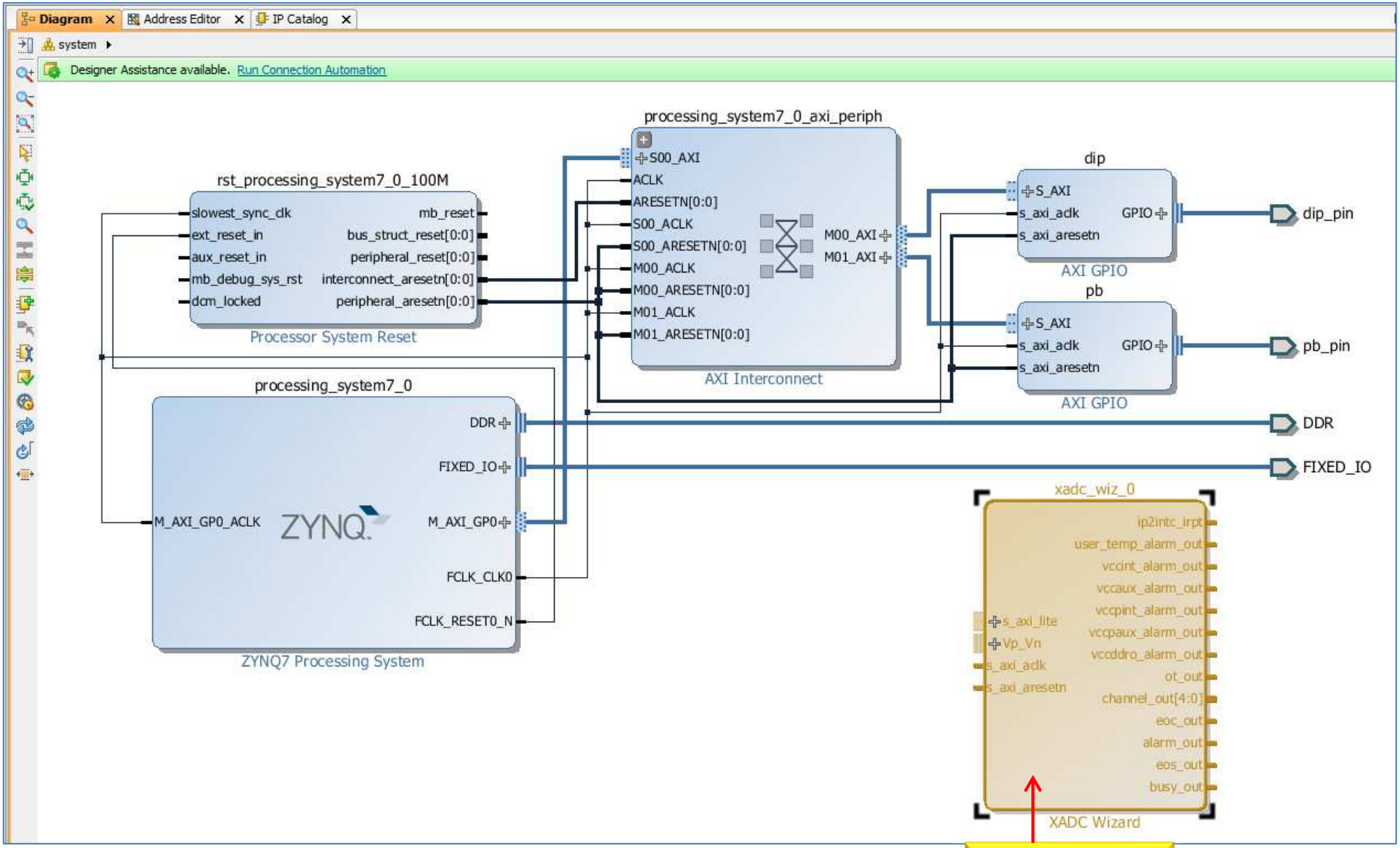
XADC kiválasztása

PL oldali XADC periféria hozzáadása és összekötése az alaprendszerrel II.

- Ezek után az XADC IP modulhoz a Vivado-ban következőket szükséges beállítani (lehet manuális / automatikus is!):
  - a.) ***interfész kapcsolatot*** az IP modul és buszrendszer (AXI Lite) között,
  - b.) IP modul ***címtartományhoz*** rendelését (Base-High Addresses),

Block Diagram

Kattintsunk duplán az XADC-re, majd paraméterezzük fel.



XADC_Wizard (GUI): paraméterek

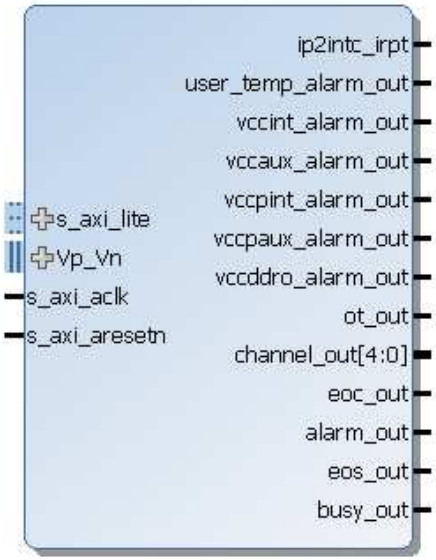
Minden default beállításon marad.

Re-customize IP

XADC Wizard (3.1)

Documentation | IP Location

Show disabled ports



Component Name: system_xadc_wiz_0_0

Basic | ADC Setup | Alarms | Channel Sequencer | Summary

Interface Options

AXI4Lite | DRP | None

Startup Channel Selection

- Simultaneous Selection
- Independent ADC
- Single Channel
- Channel Sequencer

AXI4STREAM Options

Enable AXI4Stream

FIFO Depth: 7 [7 - 1020]

Control/Status Ports

reset_in | Temp Bus | JTAG Arbiter

Event Mode Trigger

convst in | convstclk in

Timing Mode

Continuous Mode | Event Mode

DRP Timing Options

Enable DCLK

DCLK Frequency(MHz): 100 [8.0 - 250.0]

ADC Conversion Rate(KSPS): 1000 [154.0 - 1000.0]

Acquisition Time (CLK): 4

Clock divider value = 4

ADC Clock Frequency(MHz) = 25.00

Analog Sim File Options

Sim File Selection: Default

Analog Stimulus File: design

Sim File Location: ./

Waveform Type: CONSTANT

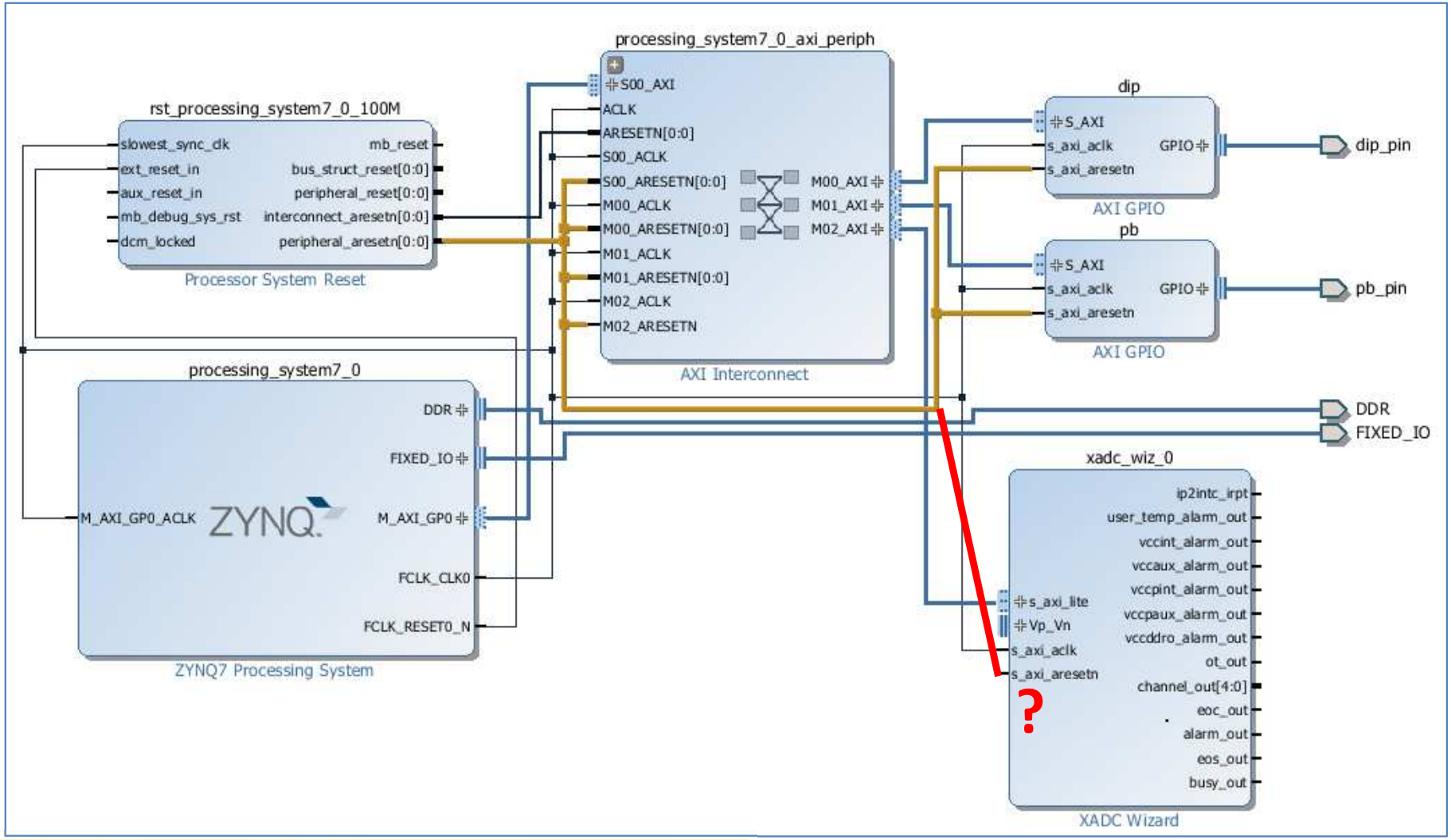
OK | Cancel

XADC bekötése (autorouter)

Diagram –> Run Connection Automation indítása

Xadc_wiz_0 –> S_AXI_Lite interfész kiválasztása automata routoláshoza

Blokk diagram: XADC manuális bekötése



Autorouter sem köt be mindent!

Peripheral_aresetn[] összekötése az s_axi_aresetn-el manuálisan szükséges (ceruza ikon)

XADC - Memória címtartományok beállítása

- Block Design -> „Address Editor” nézet kiválasztása
- „Map”-eletlen („unmapped”) IP perifériák memória-címtartományhoz rendelése:
  - a.) automatikusan - címgenerálással vs. b.) manuálisan

0X4000_0000 mivel GP0 port van beállítva!

1

Diagram | **Address Editor** | IP Catalog

Cell	Slave Interface	Base Name	Offset Address		
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
dip	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
pb	S_AXI	Reg	0x4121_0000	64K	0x4121_FFFF
xadc_wiz_0	s_axi_lite	Reg	0x43C0_0000	64K	0x43C0_FFFF

a.) Automatikus címgenerálás (jobb gomb -> Auto Assign Address)

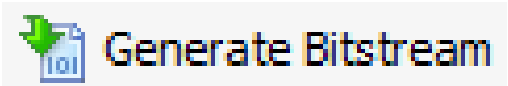
2

b.) *Base address* beállítása*

XADC_wiz_0: **0x43C0_0000 (64K)**

*Címtartományoknak 2^n méretűeknek kell lennie és nem lapolódhatnak át!

Block Design – Layout szintézis

- Ezek után egy-egy a Block Design-t frissíteni kell:
  - Regenerate Layout
  - Validate Design (DRC)
  - Flow Navigator -> Generate Bitstream 

(külső portokhoz FPGA lábakat most nem fogunk rendelni, nem kell az .XDC-t módosítani !)

Reports (implementáció után)

- Kérdés – Mennyi erőforrást foglalunk el PL oldalon?
  - Reports -> Place Design -> Utilization Report (vagy Project Summary)

```
1. Slice Logic
+----------------------------+------+-------+-----------+-------+
|          Site Type         | Used | Fixed | Available | Util% |
+----------------------------+------+-------+-----------+-------+
| Slice LUTs                 |  835 |     0 |     17600 |  4.74 |
|   LUT as Logic             |  771 |     0 |     17600 |  4.38 |
|   LUT as Memory            |   64 |     0 |      6000 |  1.07 |
|     LUT as Distributed RAM |    0 |     0 |           |       |
|     LUT as Shift Register  |   64 |     0 |           |       |
| Slice Registers            | 1063 |     0 |     35200 |  3.02 |
|   Register as Flip Flop    | 1063 |     0 |     35200 |  3.02 |
|   Register as Latch        |    0 |     0 |     35200 |  0.00 |
| F7 Muxes                   |    0 |     0 |      8800 |  0.00 |
| F8 Muxes                   |    0 |     0 |      4400 |  0.00 |
+----------------------------+------+-------+-----------+-------+
…
7. Specific Feature
+-------------+------+-------+-----------+--------+
|  Site Type  | Used | Fixed | Available |  Util% |
+-------------+------+-------+-----------+--------+
| XADC        |    1 |     0 |         1 | 100.00 |
+-------------+------+-------+-----------+--------+
```

Xilinx Vivado SDK

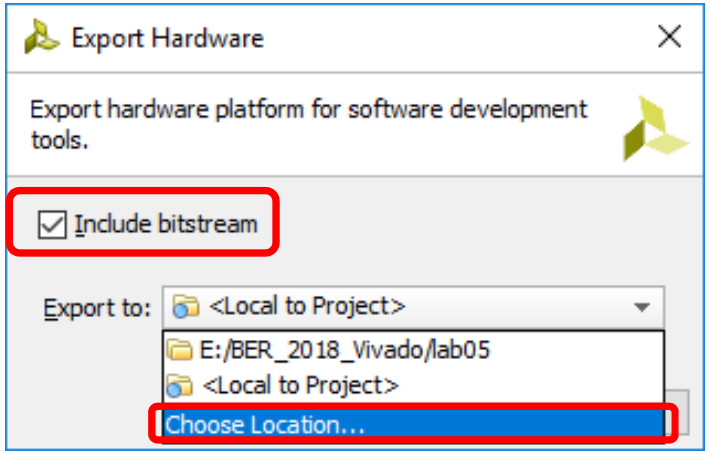
SZOFTVER TESZT ALKALMAZÁS ÖSSZEÁLLÍTÁSA (XADC)

SDK – alkalmazás-fejlesztés általános lépései

1. Vivado projekt létrehozása, majd Export → **SDK**,
2. **Új,** vagy C/C++ template-ből generálható alkalmazás létrehozása (XADC alkalmazás):
   a) **BSP** (**B**oard **S**upport **P**ackage) generálása és fordítása,
   b) **Linker Script** generálása (memória szekciók megadása, .ld),
   c) **SW** alkalmazáskód megírása/generálása, és fordítása.
3. Soros terminál beállítása (USB-soros port beállítása),
4. JTAG-USB programozó csatlakoztatása és beállítása,
5. *FPGA konfigurálása!! (.bit, van PL oldal)*
6. *‚debug' – hardveres hibakeresés* beállítása
7. Debug (breakpoint-ok beszúrása, léptetés, stb.)

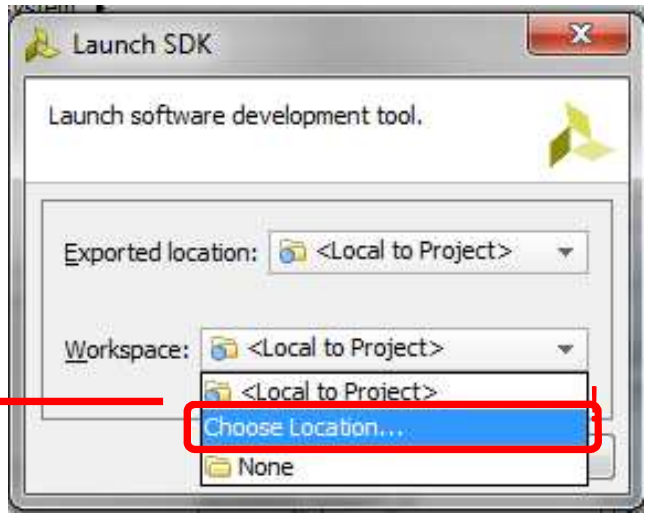
Export HW -> Vivado SDK

- **File -> Export -> Export hardware**



Mivel PL (FPGA) oldal is konfigurálva lett, ezért most már kell a bitstream is az SDK-hoz. **Choose Location... lab04** kiválasztása! (–> XADC)

- **File -> Launch SDK, OK.**
  - Vivado SDK indítása a keretrendszerből (ezt kívülről indítva is meg lehetne tenni)



Fontos, hogy a workspace-t lehetőleg ne az előző projektbe hozza létre - > Choose location... – és mostani **lab04** kiválasztása (-> XADC)

Vivado SDK GUI

.hdf = Hardware description file = hardver konfigurációs csomag file.
(Valójában egy zip, amely két leíró xml-t tartalmaz!) Ebből olvassa ki az SDK a beállításokat.

C/C++ - system_wrapper_hw_platform_0/system.hdf - Xilinx SDK

File Edit Source Refactor Navigate Search Project Xilinx Tools Run Window Help

Project Explorer

- system_wrapper_hw_platform_0
  - ps7_init_gpl.c
  - ps7_init_gpl.h
  - ps7_init.c
  - ps7_init.h
  - ps7_init.html
  - ps7_init.tcl
  - system_bd.tcl
  - system_wrapper.bit
  - system.hdf

HW platform: Vivado-ban létrehozott beágyazott rendszer – „platform” leírása

system.hdf

ps7_dma_s	0xf8003000	0xf8003fff
ps7_globaltimer_0	0xf8f00200	0xf8f002ff
ps7_gpv_0	0xf8900000	0xf89fffff
ps7_intc_dist_0	0xf8f01000	0xf8f01fff
ps7_iop_bus_config_0	0xe0200000	0xe0200fff
ps7_l2cachec_0	0xf8f02000	0xf8f02fff
ps7_ocmc_0	0xf800c000	0xf800cfff
ps7_pl310_0	0xf8f02000	0xf8f02fff
ps7_pmu_0	0xf8891000	0xf8891fff
ps7_pmu_0	0xf8893000	0xf8893fff
ps7_ram_0	0x00000000	0x0002ffff
ps7_ram_1	0xffff0000	0xffffdfff
ps7_scuc_0	0xf8f00000	0xf8f000fc
ps7_scugic_0	0xf8f01000	0xf8f01fff
ps7_scugic_0	0xf8f00100	0xf8f001ff
ps7_scutimer_0	0xf8f00600	0xf8f0061f
ps7_scuwdt_0	0xf8f00620	0xf8f006ff
ps7_slcr_0	0xf8000000	0xf8000fff
ps7_uart_1	0xe0001000	0xe0001fff
ps7_xadc_0	0xf8007100	0xf8007120
xadc_wiz_0	0x43c00000	0x43c0ffff

!

Memória térkép (PS)

Address Map for processor ps7_cortexa9_1

Overview

Outline — An outline is not available.

Make Target

Target Connections

- Hardware Server
- Linux TCF Agent
- QEMU TcfGdbClient

Felhasznált PS oldali perifériák listája és verziói (lejjebb)

IP blocks present in the design

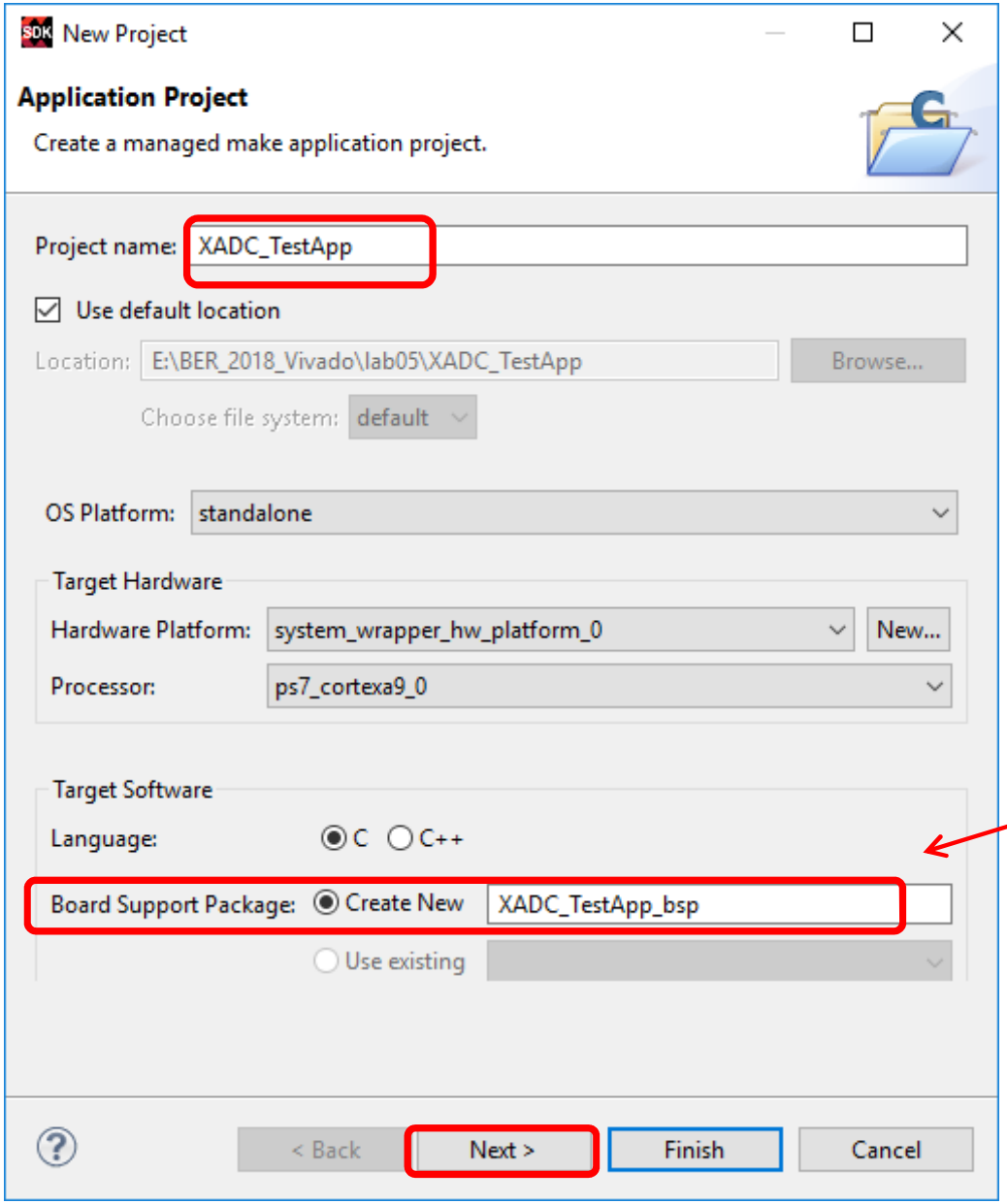
dip	axi_gpio	2.0
pb	axi_gpio	2.0
processing_system7_0	processing_system7	5.5
processing_system7_0_axi_periph	axi_interconnect	2.1
rst_processing_system7_0_100M	proc_sys_reset	5.0
xadc_wiz_0	xadc_wiz	3.1

SDK Log

```
00:14:54 INFO  : Launching XSDB server: xsdb.bat -s C:/Xilinx/SDK/2
00:14:55 INFO  : Processing command line option -hwspec E:/BER_2018
00:14:56 INFO  : XSDB server has started successfully.
```

SDK szoftver projekt létrehozása I./a

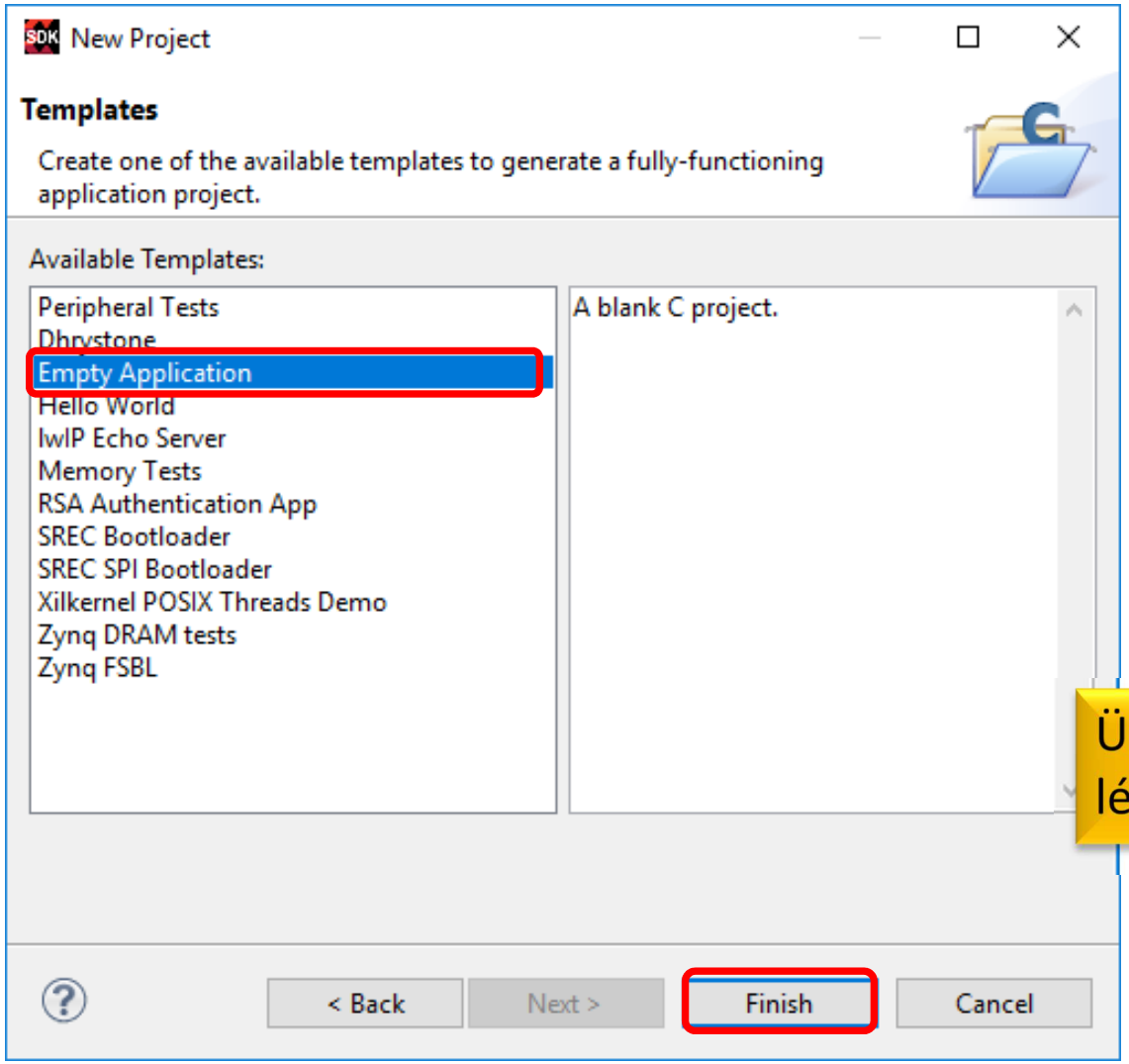
File → New → Application Project (sw alk. neve legyen pl. `XADC_TestApp`**)**



Új (vagy meglévő) BSP fájl létrehozása/ hozzárendelése

NEXT >

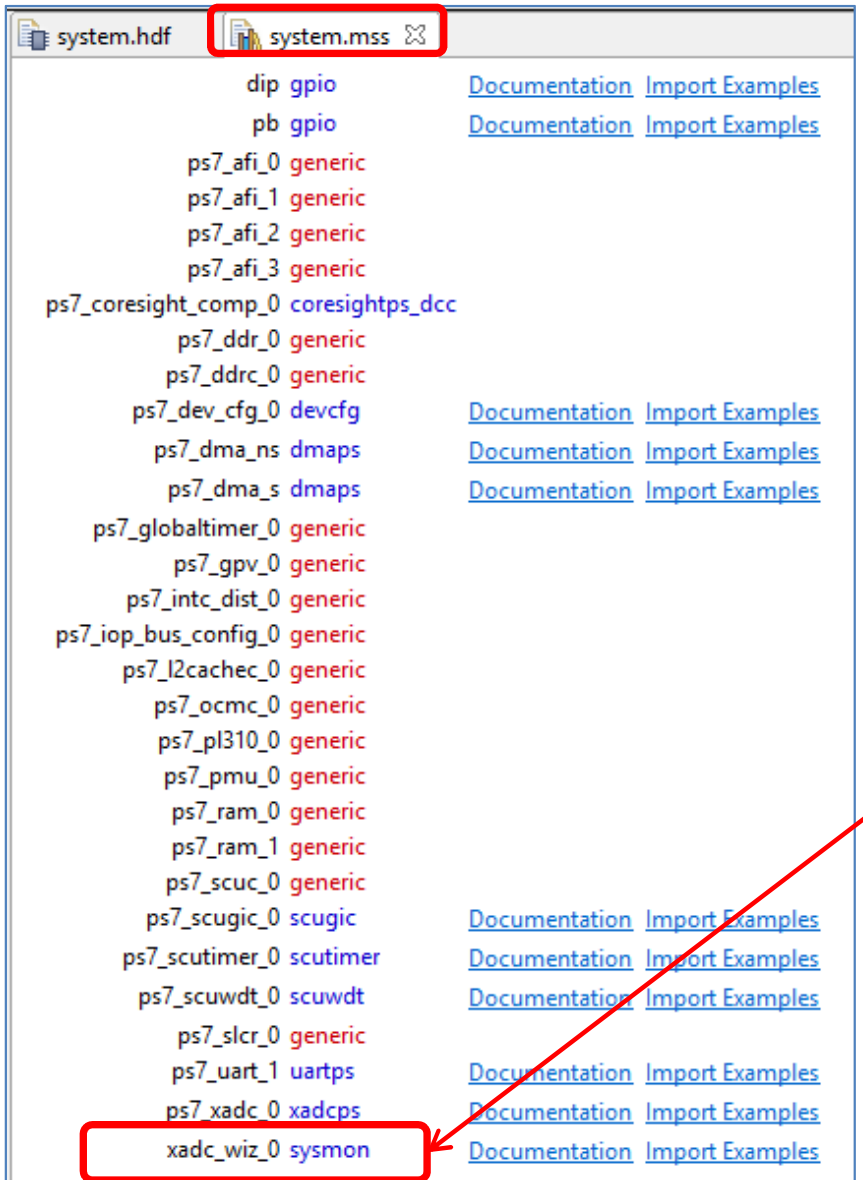
SDK - BSP létrehozása I./b



Üres alkalmazás létrehozása.

SDK – MSS ellenőrzése I./c

- SDK: Válasszuk ki a `system.mss` fület
- Vizsgáljuk meg az „Overview”, ill. „Source” nézeteket!



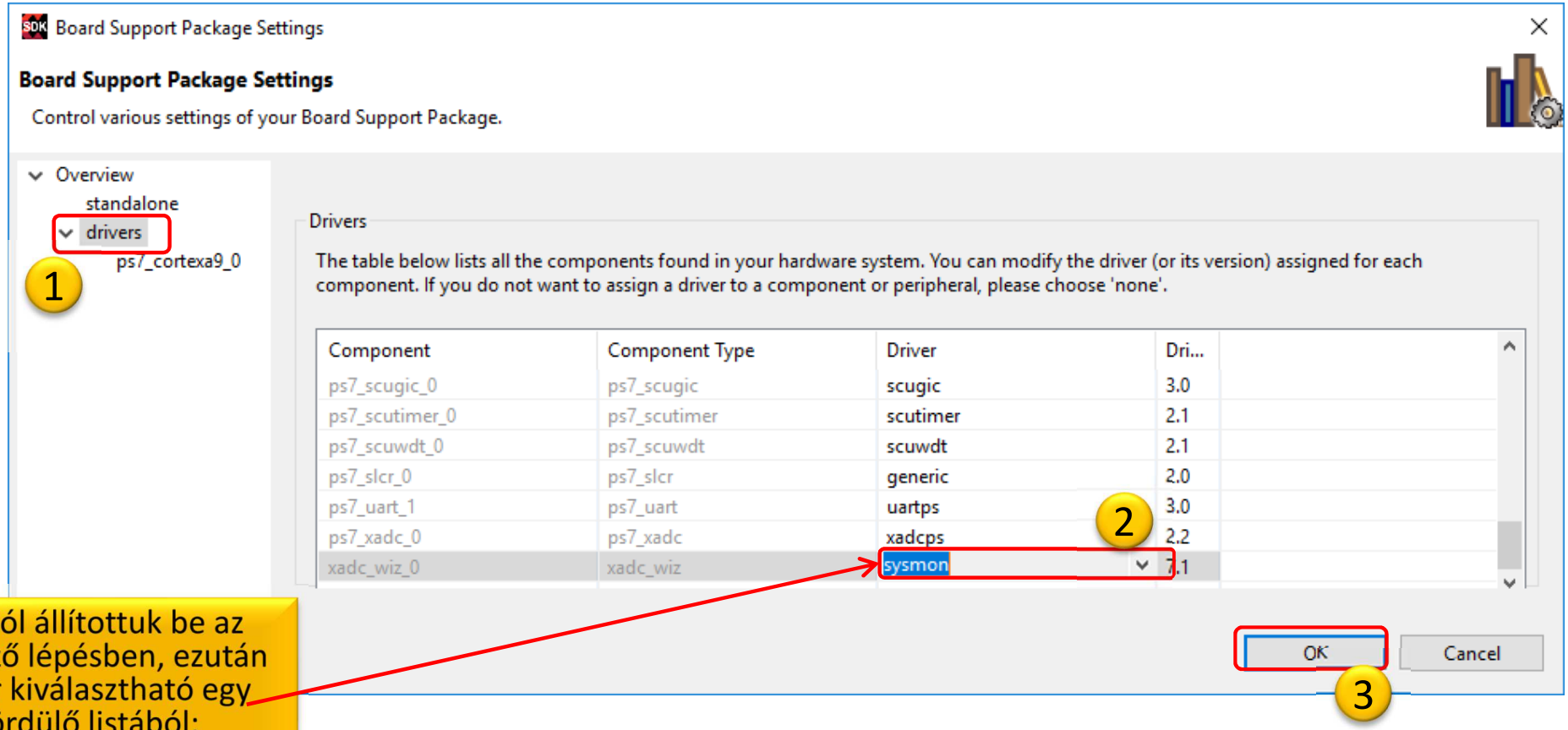
MSS: **M**icroprocessor **S**oftware **S**pecification

sysmon: ha ez látható, akkor a *`xadc_wiz_0`*-hoz már hozzá van rendelve a megfelelő driver (egyébként **generic**)

Választhatunk grafikus „overview” vagy „source” (driver forrás .mss) nézetek között.

SDK – BSP XADC driver hozzáadása I./d

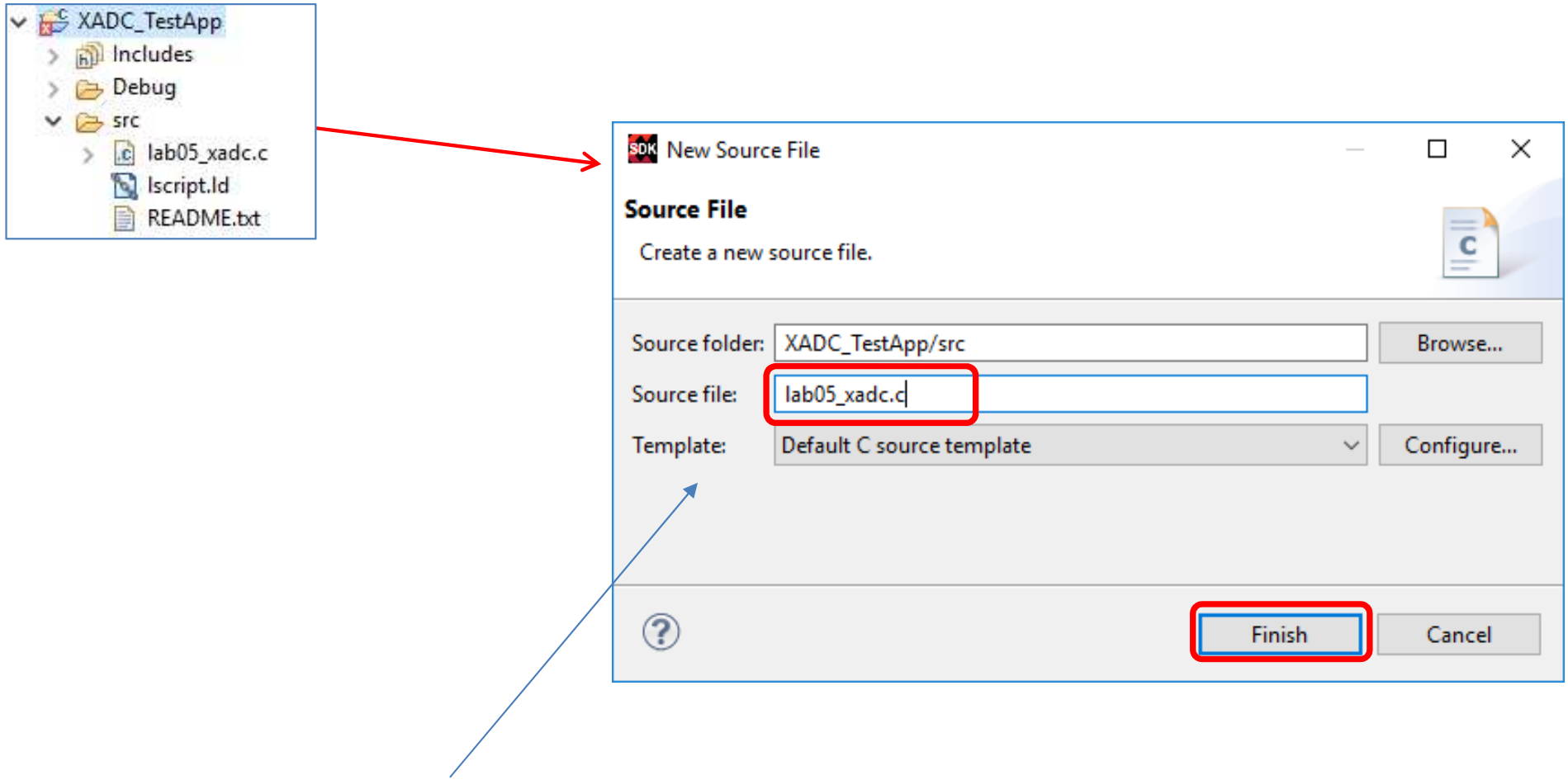
- **Project Explorer → Jobb kattintás a** `XADC_TestApp_bsp` **→ Board Support Package Settings**



Ha jól állítottuk be az előző lépésben, ezután már kiválasztható egy legördülő listából: „***sysmon***” nevű driver.

SDK - C forráskód létrehozása II./a

File → New → Source File (neve legyen `lab05_xadc.c`)



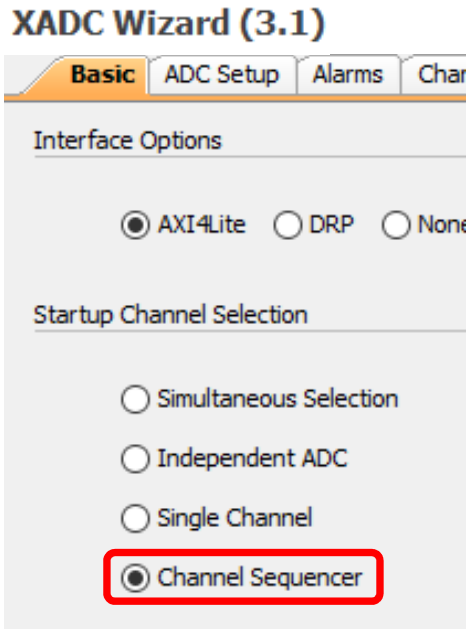
forrás kód: BER_lab_xadc.zip (másoljuk bele a tartalmát)

12.	máj. 4.	XADC - System monitor BER_08_TAMOP_LAB05_XADC_2018	BER_lab5_xadc.zip

C forráskód I.

```
xil_printf("-- PL XADC_WIZARD demonstration on ZyBo -- \n\r");
xil_printf("XADC configuration started...\n\r");

//XADC initilization

 ConfigPtr = XAdcPs_LookupConfig(XPAR_AXI_XADC_0_DEVICE_ID);
        if (ConfigPtr == NULL) {
                    return XST_FAILURE;
        }

 Status_ADC = XAdcPs_CfgInitialize(XADCInstPtr,ConfigPtr,ConfigPtr->BaseAddress);
 if(XST_SUCCESS != Status_ADC){
     print("ADC INIT FAILED\n\r");
     return XST_FAILURE;
  }

 //self test
 Status_ADC = XAdcPs_SelfTest(XADCInstPtr);
        if (Status_ADC != XST_SUCCESS) {
                    return XST_FAILURE;
        }

        //stop sequencer
        XAdcPs_SetSequencerMode(XADCInstPtr,XADCPS_SEQ_MODE_SINGCHAN);

        //disable alarms
        XAdcPs_SetAlarmEnables(XADCInstPtr, 0x0);

        //configure sequencer to just sample internal on chip parameters
        XAdcPs_SetSeqInputMode(XADCInstPtr, XADCPS_SEQ_MODE_SAFE);

        //configure the channel enables we want to monitor
        XAdcPs_SetSeqChEnables(XADCInstPtr,XADCPS_CH_TEMP|XADCPS_CH_VCCINT|XADCPS_CH_VCCAUX|XADCPS_CH_VBRA
M|XADCPS_CH_VCCPINT|
                    XADCPS_CH_VCCPAUX|XADCPS_CH_VCCPDRO);
```

Xadcps.c és xadcps.h fileok-ban deklaráltak.



C forráskód II.

```
while(1){
                xil_printf("\n\r Next iteration - read out XADC registers ...\n\r");

                TempRawData = XAdcPs_GetAdcData(XADCInstPtr, XADCPS_CH_TEMP);
                TempData = XAdcPs_RawToTemperature(TempRawData);
                printf("Raw Temp %lu -> Real Temp %f [C°]\n\r", TempRawData, TempData);

                VccIntRawData = XAdcPs_GetAdcData(XADCInstPtr, XADCPS_CH_VCCINT);
                VccIntData = XAdcPs_RawToVoltage(VccIntRawData);
                printf("Raw VccInt %lu -> Real VccInt %f [V]\n\r", VccIntRawData, VccIntData);

                VccAuxRawData = XAdcPs_GetAdcData(XADCInstPtr, XADCPS_CH_VCCAUX);
                VccAuxData = XAdcPs_RawToVoltage(VccAuxRawData);
                printf("Raw VccAux %lu -> Real VccAux %f [V]\n\r", VccAuxRawData, VccAuxData);

                VBramRawData = XAdcPs_GetAdcData(XADCInstPtr, XADCPS_CH_VBRAM);
                VBramData = XAdcPs_RawToVoltage(VBramRawData);
                printf("Raw VccBram %lu -> Real VccBram %f [V]\n\r", VBramRawData, VBramData);

                VccPIntRawData = XAdcPs_GetAdcData(XADCInstPtr, XADCPS_CH_VCCPINT);
                VccPIntData = XAdcPs_RawToVoltage(VccPIntRawData);
                printf("Raw VccPInt %lu -> Real VccPInt %f [V]\n\r", VccPIntRawData, VccPIntData);

                VccPAuxRawData = XAdcPs_GetAdcData(XADCInstPtr, XADCPS_CH_VCCPAUX);
                VccPAuxData = XAdcPs_RawToVoltage(VccPAuxRawData);
                printf("Raw VccPAux %lu -> Real VccPAux %f [V]\n\r", VccPAuxRawData, VccPAuxData);

                VDDRRawData = XAdcPs_GetAdcData(XADCInstPtr, XADCPS_CH_VCCPDRO);
                VDDRData = XAdcPs_RawToVoltage(VDDRRawData);
                printf("Raw VccDDR %lu -> Real VccDDR %f [V]\n\r", VDDRRawData, VDDRData);
```

Eredmény – kiíratás (console)

Console | Tasks | Terminal | Problems | Exe

XADC_TestApp Debug [Xilinx C/C++ application (GDB)] E:\BER_

```
-- PL XADC_WIZARD demonstration on ZyBo --
XADC configuration started...

 Next iteration - read out XADC registers ...
Raw Temp 41048 -> Real Temp 42.512512 [C°]
Raw VccInt 21686 -> Real VccInt 0.992706 [V]
Raw VccAux 39237 -> Real VccAux 1.796127 [V]
Raw VccBram 21682 -> Real VccBram 0.992523 [V]
Raw VccPInt 21645 -> Real VccPInt 0.990829 [V]
Raw VccPAux 39230 -> Real VccPAux 1.795807 [V]
Raw VccDDR 32564 -> Real VccDDR 1.490662 [V]


 Next iteration - read out XADC registers ...
Raw Temp 41022 -> Real Temp 42.312592 [C°]
Raw VccInt 21686 -> Real VccInt 0.992706 [V]
Raw VccAux 39237 -> Real VccAux 1.796127 [V]
Raw VccBram 21682 -> Real VccBram 0.992523 [V]
Raw VccPInt 21645 -> Real VccPInt 0.990829 [V]
Raw VccPAux 39230 -> Real VccPAux 1.795807 [V]
Raw VccDDR 32564 -> Real VccDDR 1.490662 [V]
```

Kérdés:

- Mekkora a letöltendő .elf file mérete?
  - Megoldás:

```
'Invoking: ARM Print Size'
arm-xilinx-eabi-size XADC_TestApp.elf  |tee "XADC_TestApp.elf.size"
   text    data     bss     dec     hex filename
  68004    2328   22668   93000   16b48 XADC_TestApp.elf
'Finished building: XADC_TestApp.elf.size'
```

- Miért ilyen nagy a mérete?
  - Megoldás: Printf() !? (formátumvezérelt kiíratás)