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: 2019. október 25.

Beágyazott rendszerek fejlesztése (FPGA)

Dr. Vörösházi Zsolt

5. Periféria modul illesztése (PMOD TMP) hőmérséklet méréssel

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**

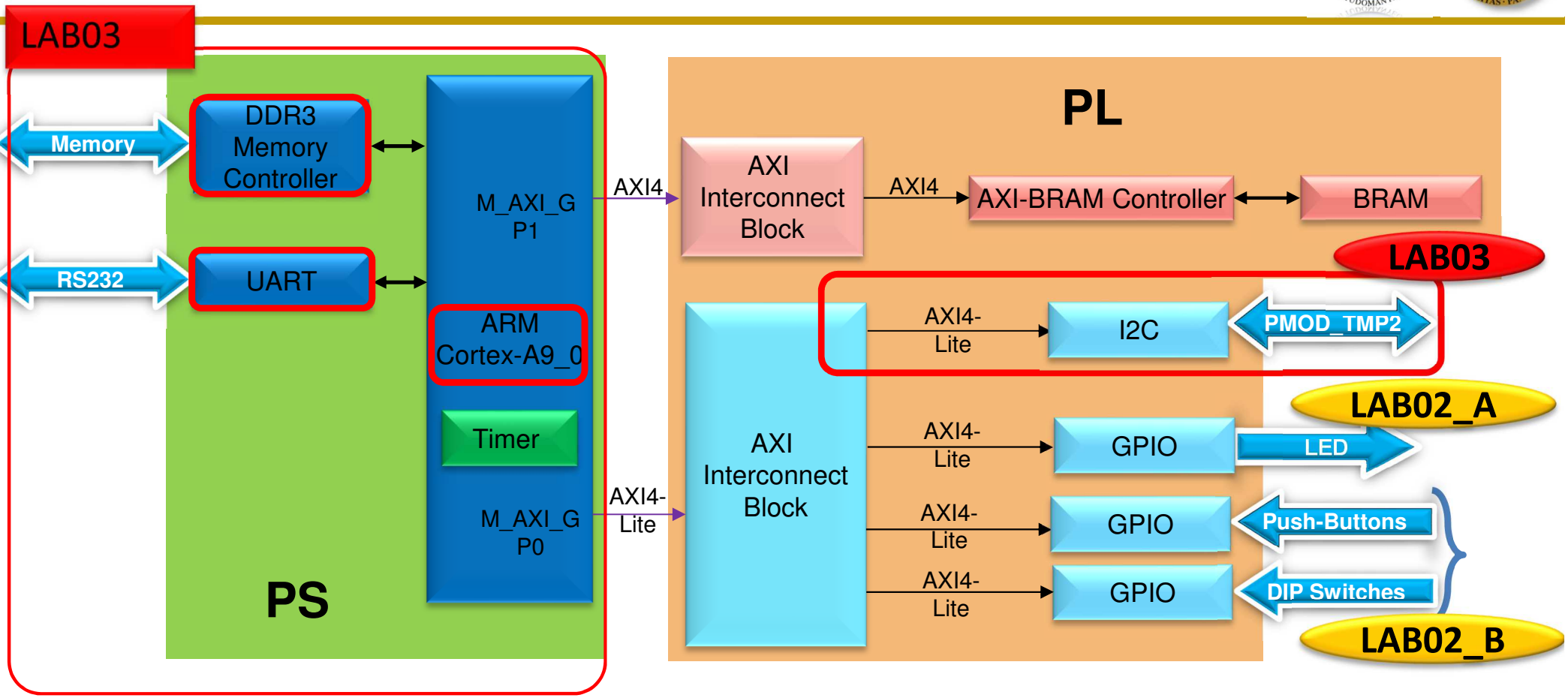
A feladat megoldásának lépései

- Az előző (04. fólia) ismeretkör elsajátítása során létrehozott
  - **File -> Project -> Save As…**
  - Alternatív módszer lehet: projekt archiválása (`LAB02B` → `LAB03` néven), majd pedig a \`LAB03` megnyitása Vivado-ban,

- Az IP katalógusból kiválasztott I^2C periféria integrálása és összekötése az alaprendszerrel,
- Külső I^2C portok létrehozása (.XDC),

- Egy Periféria teszt-alkalmazás (PMOD_Test) készítése az SDK-környezetben,
- A FW+SW tervek teszt verifikálása a Digilent ZyBo kártyán.

A bővítendő tesztrendszer



PS oldal:

- ARM hard-processzor mag
- belső OnChip-RAM vezérlő
- RS232 soros interfész
- külső DDR3 memória vezérlő
- (I^2C vezérlő opcionális)
- Timer / interrupt vezérlő (ARM)

PL oldal:

- LAB02_A: GPIO bemenetek
  - PBSs: Push Button (nyomógomb kezelő)
  - DIPs: Switches (kapcsoló kezelő)
- LAB02_B: GPIO kimenet
  - LEDs: LED kijelző

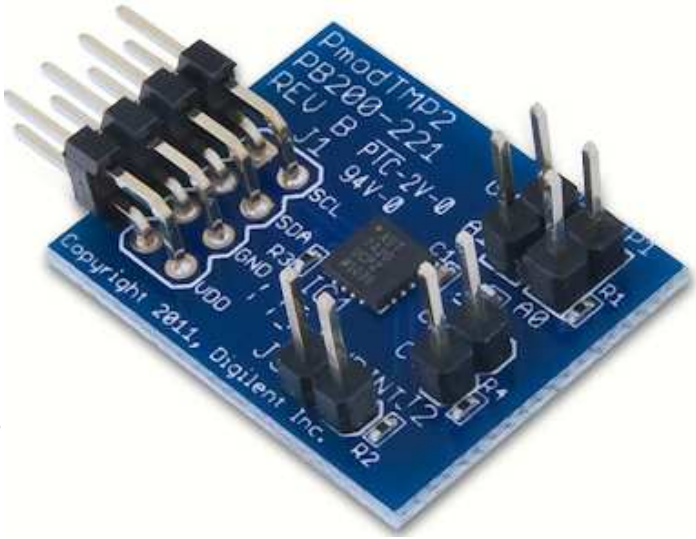
- **LAB03: I^2C vezérlős hőmérsékletmérő modul (PMOD_TMP2)**
  - **ARM PS/ vagy PL oldalra is csatlakoztatható**

Irodalmi háttér

- Digilent PMOD TMP2 hőmérséklet szenzor modul (I^2C):
  - 🕮 https://store.digilentinc.com/pmod-tmp2-temperature-sensor/
  - 🕮 https://reference.digilentinc.com/reference/pmod/pmodtmp2/reference-manual
- Analog Devices ADT7420 hőmérséklet érzékelő IC:
  - 🕮 http://www.analog.com/media/en/technical-documentation/data-sheets/ADT7420.pdf
- Analog Devices – Digilent közös Wiki oldala:
  - 🕮 http://wiki.analog.com/resources/alliances/digilent

  Referencia terv FPGA-ra (SW driver-ekkel)!
  - 🕮 http://wiki.analog.com/resources/fpga/xilinx/pmod/adt7420
- I^2C szabványról általánosan:
  - 🕮 Dr. Fodor Attila, Dr. Vörösházi Zsolt: Beágyazott rendszerek, TÁMOP 4.1.2 (PE MIK, Villamosmérnöki és Információs Rendszerek Tanszék) 2011. http://tananyagfejlesztes.mik.uni-pannon.hu/
  - 🕮 I2C Wikipedia: https://en.wikipedia.org/wiki/I%C2%B2C

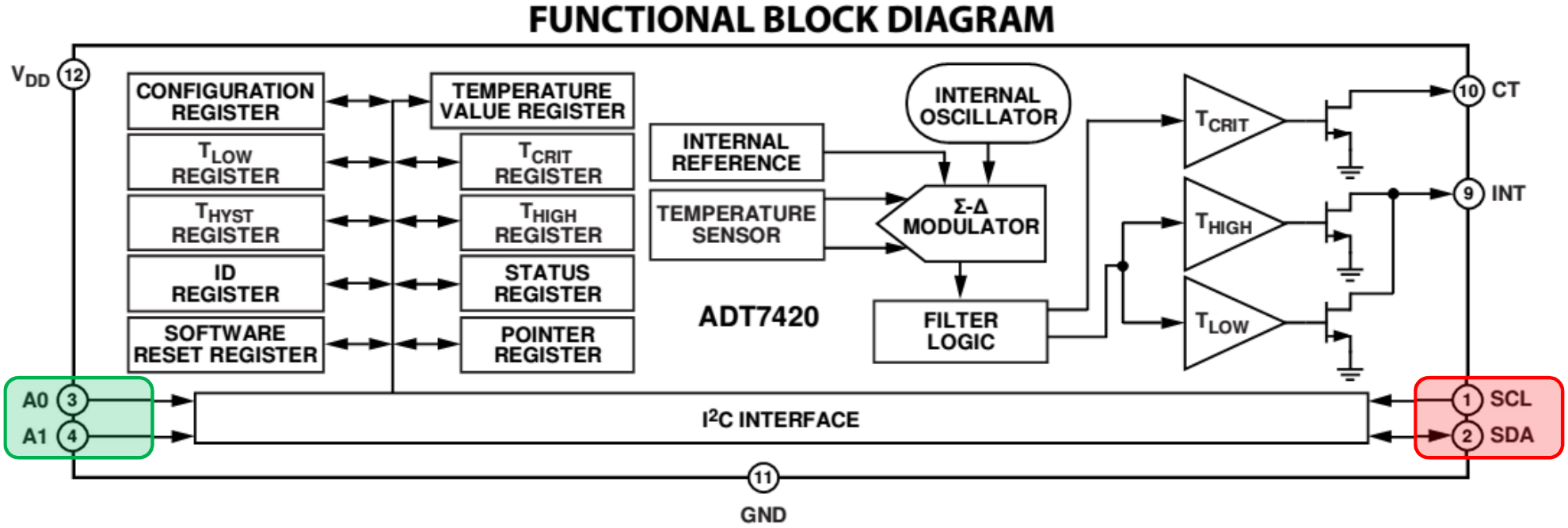
Digilent PMOD_TMP2



- I^2C alapú **hőmérséklet érzékelő és hőfokszabályzó periféria modul**
  - T_A= –40°C … +150°C,
  - max. 16-bit felbontásig skálázható,
  - Átlagos pontossága jobb, mint 0.25 °C,
    - 13 = 9+4 bites módban: $1/2^4$ = 0.0625 °C,
    - 16 = 9+7 bites módban: $1/2^7$ = 0.0078 °C,
  - **I^2C interfész**, 4 választható (jumper) I^2C címmel (A_1-A_0)
    - „Daisy Chain” lehetőség (7/10 bites címzés)
  - Folyamatos konverzió 240ms-ként,
  - Programozható küszöbértékek (max/min - CT), külső lábak, mint threshold (INT),
  - **3.3**V vagy 5V interfész támogatás,
  - Kalibrálást nem igényel!

ADT 7420

- Az áramkör blokkszintű kapcsolási rajza:



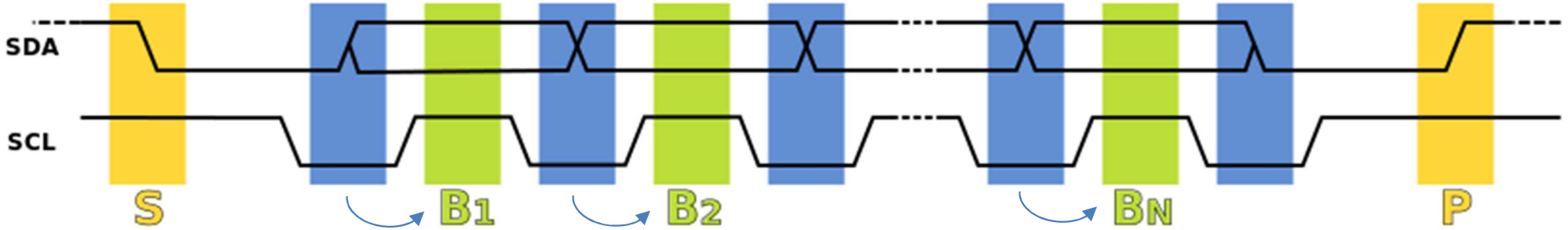
- PMOD_TMP2 jelei / választható I²C címek:

Connector J1 – I2C Communications		
Pin	**Signal**	**Description**
1, 2	SCL	I2C Clock
3, 4	SDA	I2C Data
5, 6	GND	Power Supply Ground
7, 8	VCC	Power Supply (3.3V/5V)

Addresses		
JP2	**JP1**	**Address**
Open	Open	0x4B (0b1001011)
Open	Shorted	0x4A (0b1001010)
Shorted	Open	0x49 (0b1001001)
Shorted	Shorted	0x48 (0b1001000)

Philips I²C szabványról (1982)

Forrás: https://en.wikipedia.org/wiki/I%C2%B2C



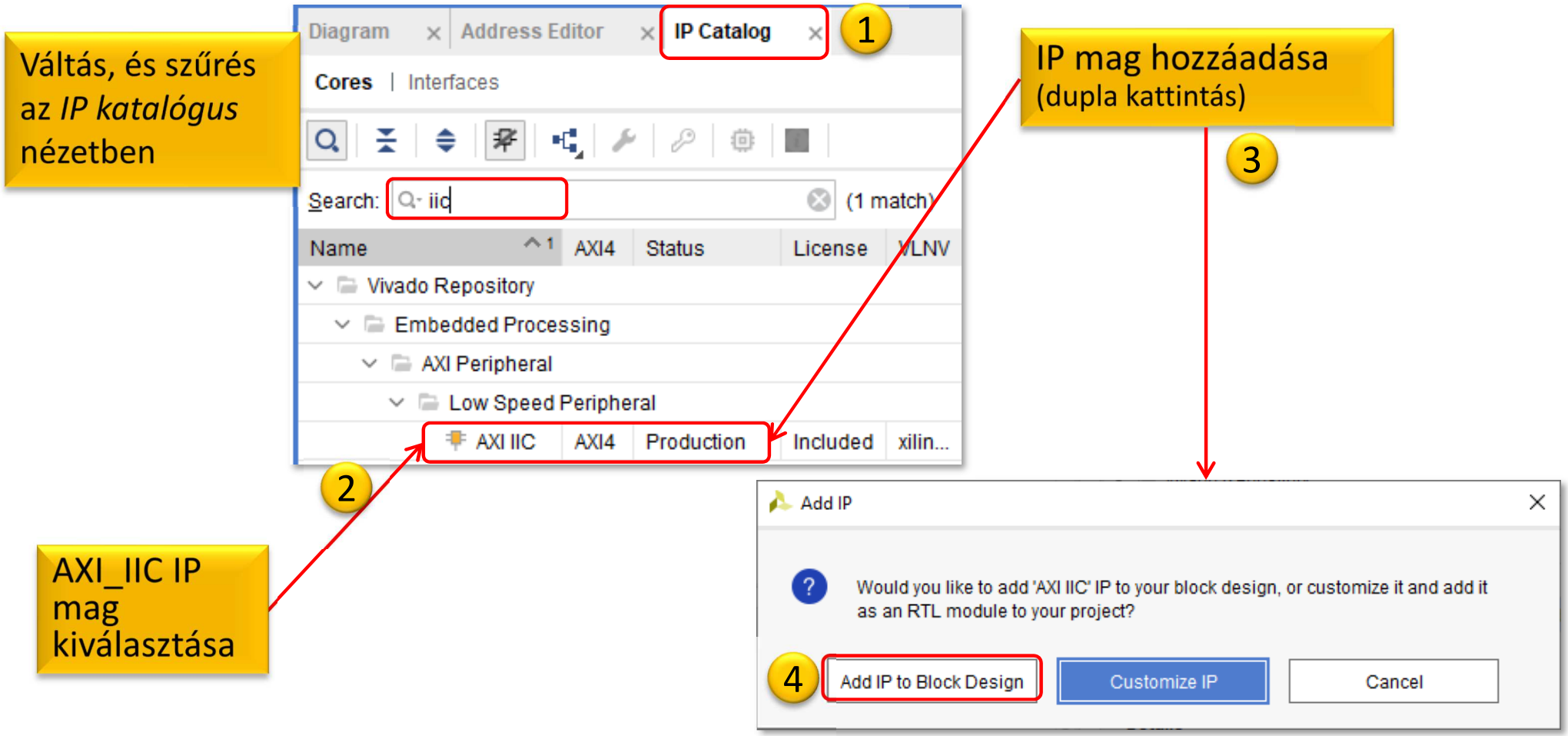
1. Data Transfer is initiated with a *START bit* (S) signaled by **SDA** being pulled **low** while SCL stays high (pull-down).
2. SDA sets the 1st data bit level while keeping **SCL low** (during blue bar time) .
3.The data is sampled (received) when **SCL rises** (green) for the first bit (B1).
4. This process repeats, SDA transitioning until **SCL is low** again, and the data being read while **SCL is high** (B2, Bn).
5. A *STOP bit* (P) is signaled when SDA is pulled high while **SCL** is **high**.

Reserved Address Index	8 Bit Byte			Description
	7 bit Address		R/W Value	
	MSB (4bit)	LSB (3bit)	1 bit	
1	0000	000	0	General Call
2	0000	000	1	START BYTE
3	0000	001	X	CBUS Address
4	0000	010	X	Reserved For Different Bus Format
5	0000	011	X	Reserved For Future Purpose
6	0000	0XX	X	HS-mode Master Code
7	1111	1XX	1	Device ID
8	1111	0XX	X	10-bit slave addressing

„Master write(0) to / read(1) from the Slave”

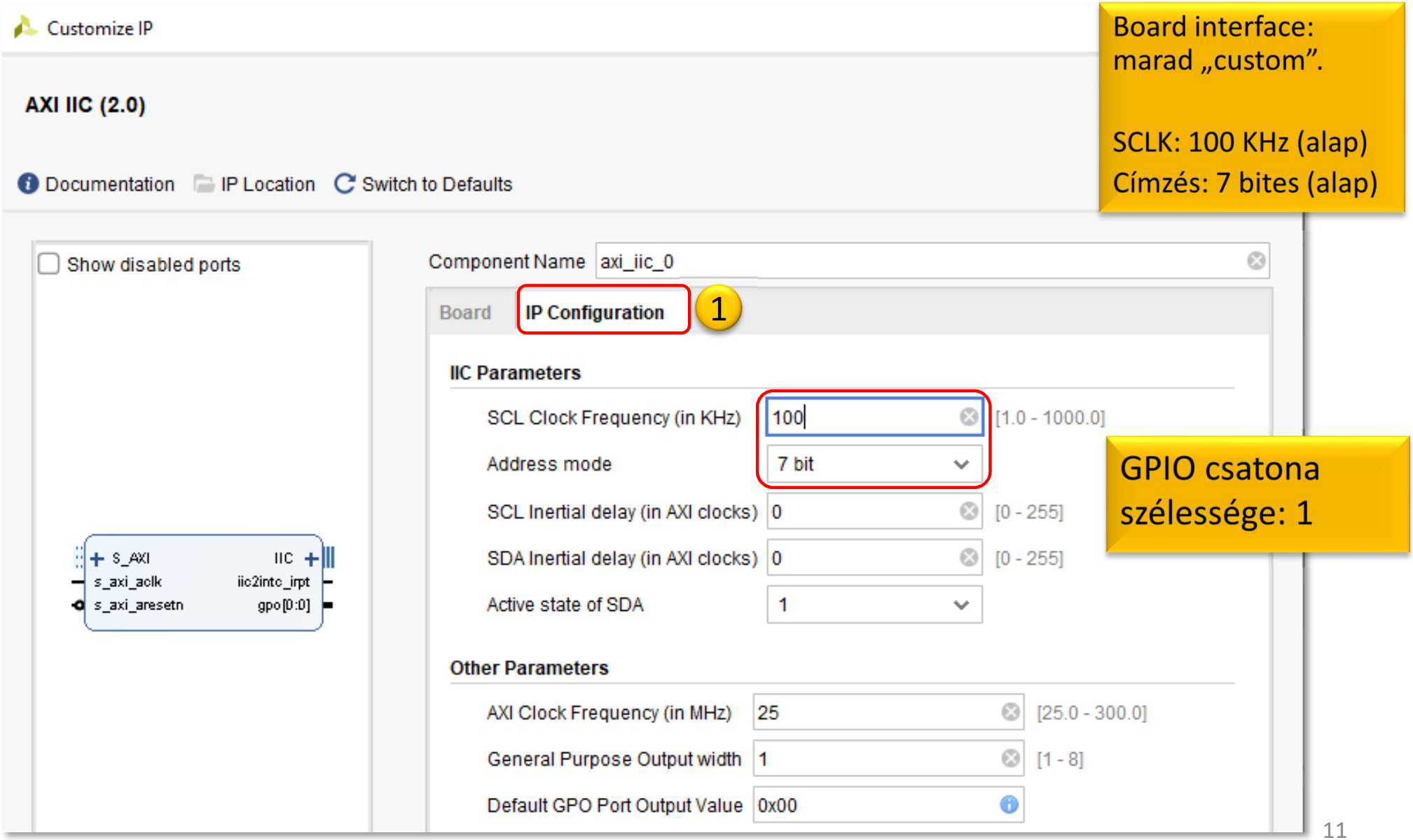
I²C vezérlő hozzáadása és összekötése az alaprendszerrel I.

- I²C vezérlő csatlakoztatására két lehetőség van
  - a.) **PL-oldali AXI_I2C IP mag hozzáadása, vagy**
  - b.) PS-oldali PS_I2C0/1 periféria vezérlő engedélyezése Zynq alrendszerben.
- Adjunk a processzor rendszerhez egy `AXI_IIC` perifériát a Block Diagram segítségével (IP katalógusból)



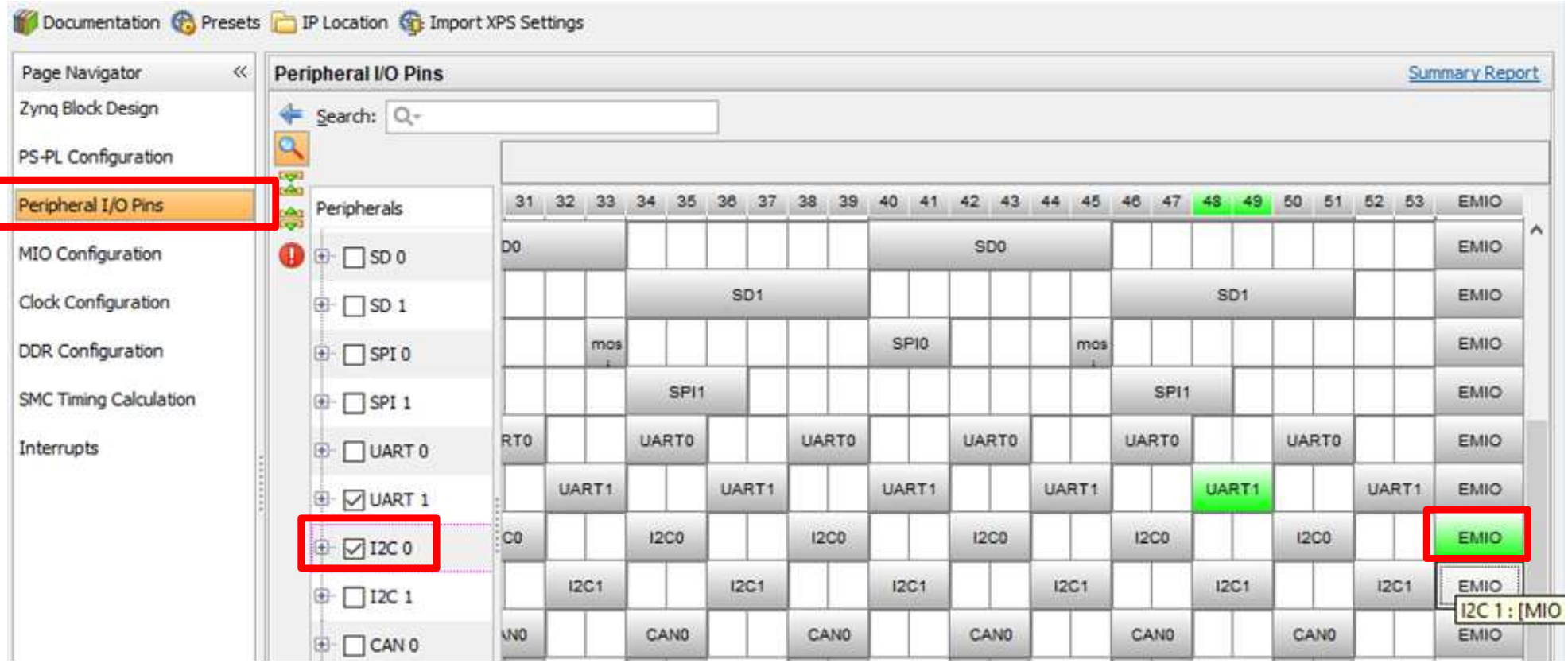
AXI IIC periféria hozzáadása az alaprendszerhez II.

Dupla kattintással vizsgáljuk meg az AXI_IIC-t:
(minden maradjon alap beállításon)



Alternatív megoldás: PS I2C

- Nem ezt használjuk, ez csak lehetőség.
- PL (FPGA) oldali I2C blokkot használjuk!



AXI GPIO-k bekötése (autorouter)

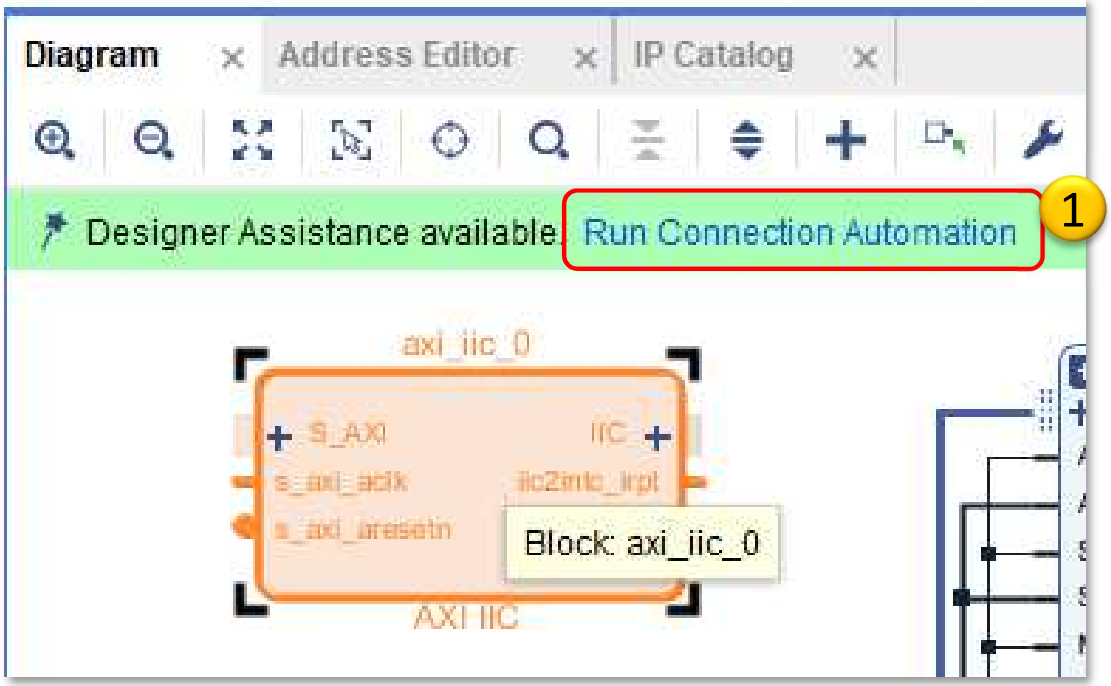
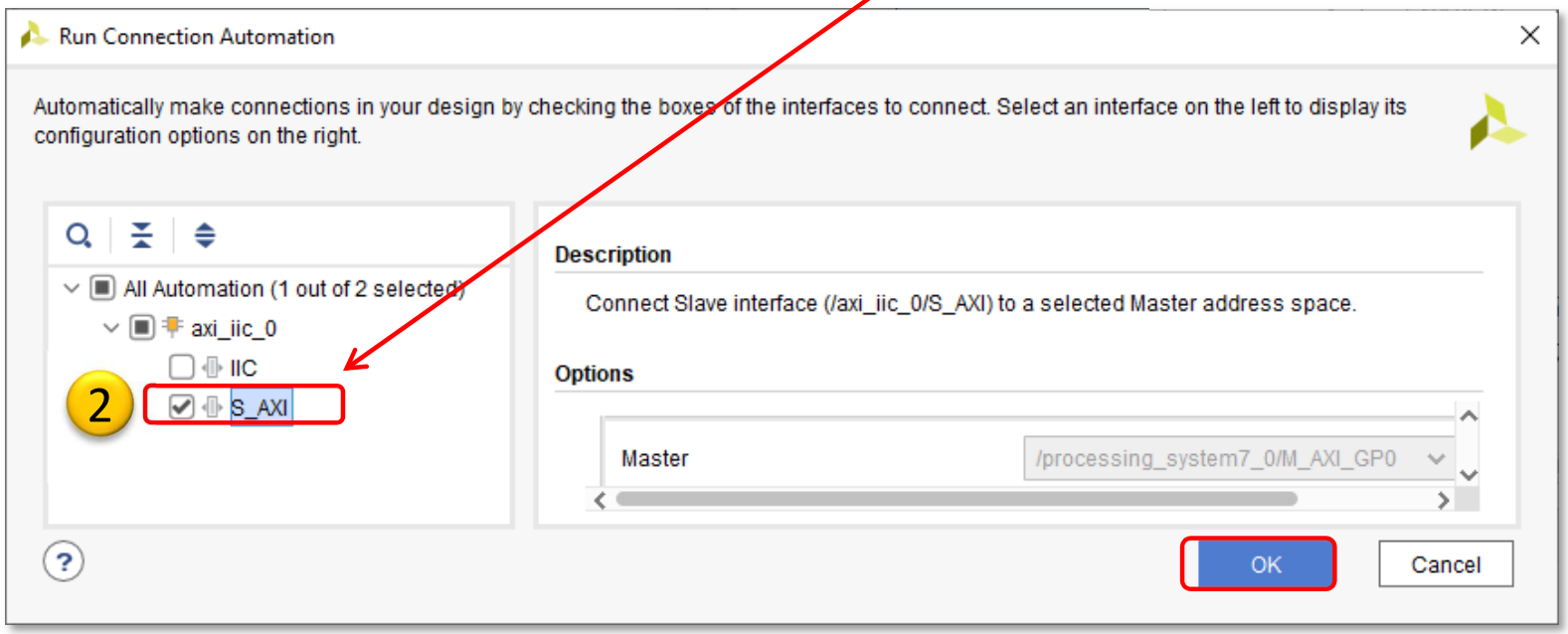


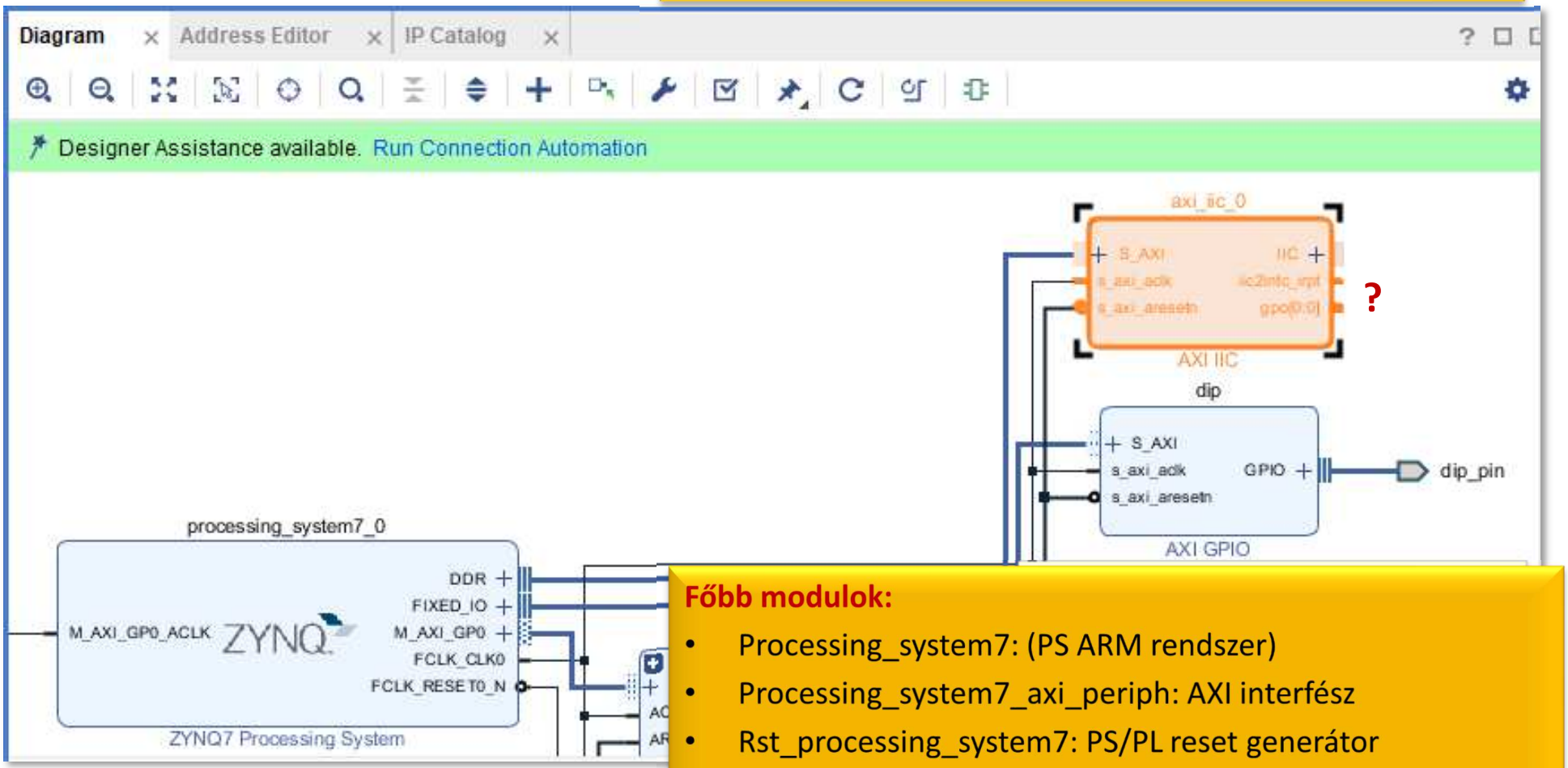
Diagram -> Run Connection Automation indítása

AXI_IIC_0 -> S_AXI interfész kiválasztása, automata routoláshoz (minden más alapértéken marad)



Block design – teljes nézet

Vizsgáljuk meg az egyes blokkokat, illetve összeköttetéseiket!

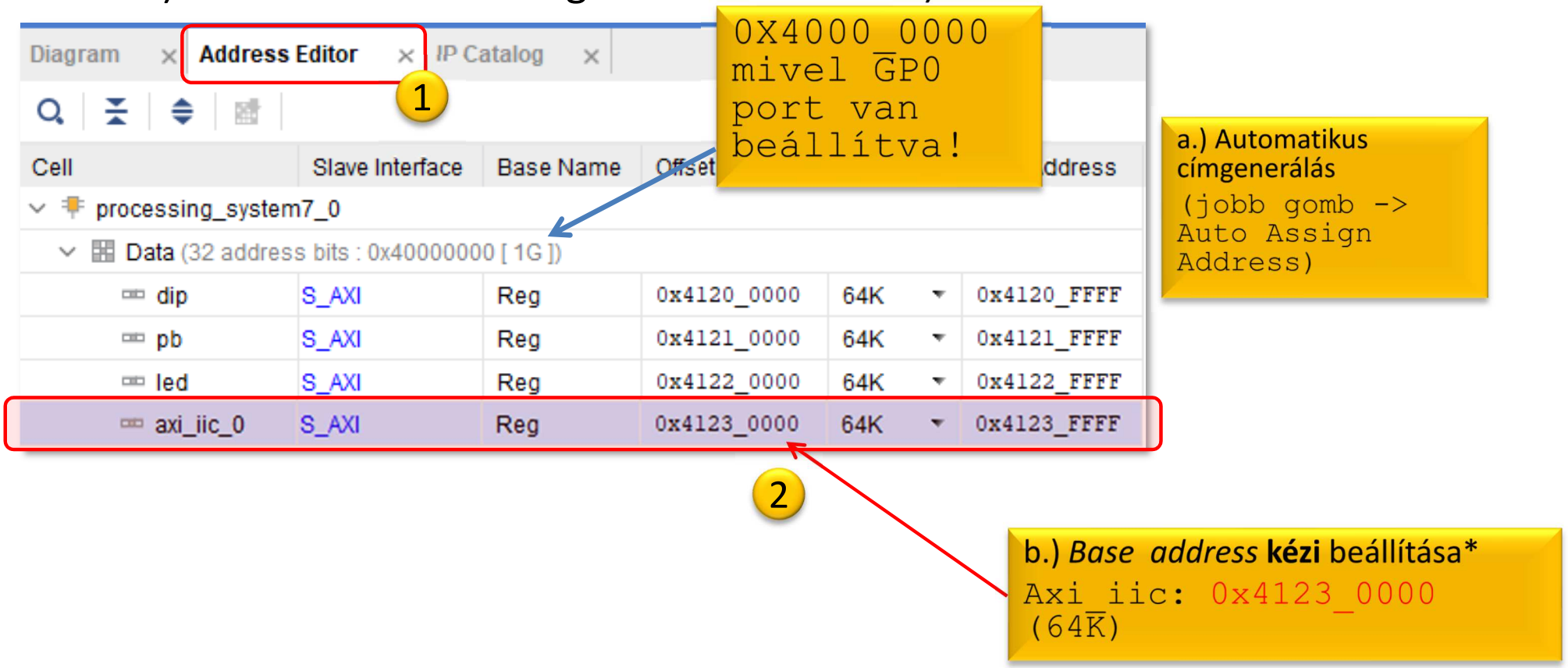


Főbb modulok:

- Processing_system7: (PS ARM rendszer)
- Processing_system7_axi_periph: AXI interfész
- Rst_processing_system7: PS/PL reset generátor
- dip: AXI GPIO a DIP kapcsolóknak (PL oldali)
- Pb: AXI GPIO a PB nyomógomboknak (PL oldali)
- Led: AXI GPIO a LED kijelzőknek (PL oldali)
- **Axi_iic_0: AXI_IIC periféria vezérlő** (PL oldali)

AXI IIC - Memória címtartományok beállítása

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



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

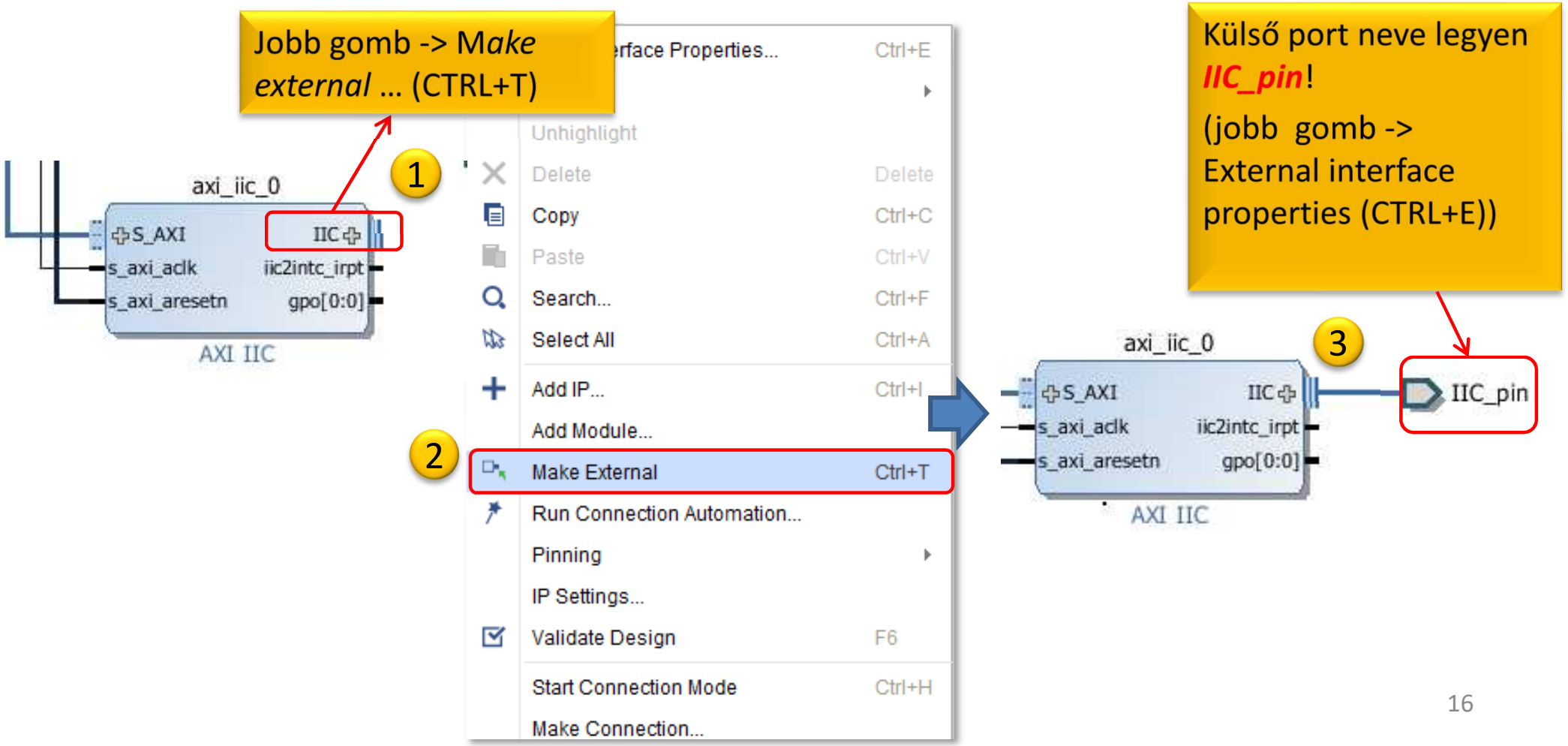
AXI GPIO - Külső portok hozzárendelése

Az AXI_IIC példányhoz hozzá kell még kapcsolni a ZyBo kártyán található (**PMOD JE 3,4 jeleket**) az FPGA (PL-oldali) lábakhoz:

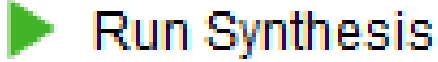
1.) Az AXI_IIC port-jait a külső fizikai FPGA lábakra (pin) is kell kötni,

2.) Ha kell, a külső portok neveit is definiáljuk (pl: „`_pin`” végződésűre), majd

3.) Az <system>.XDC fájl-ban meg kell majd adni az adott FPGA pin azonosítóját.

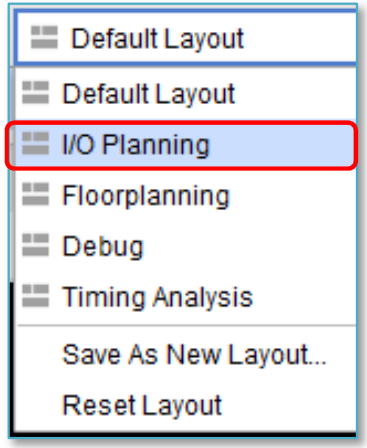


Block Design – Layout szintézis

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

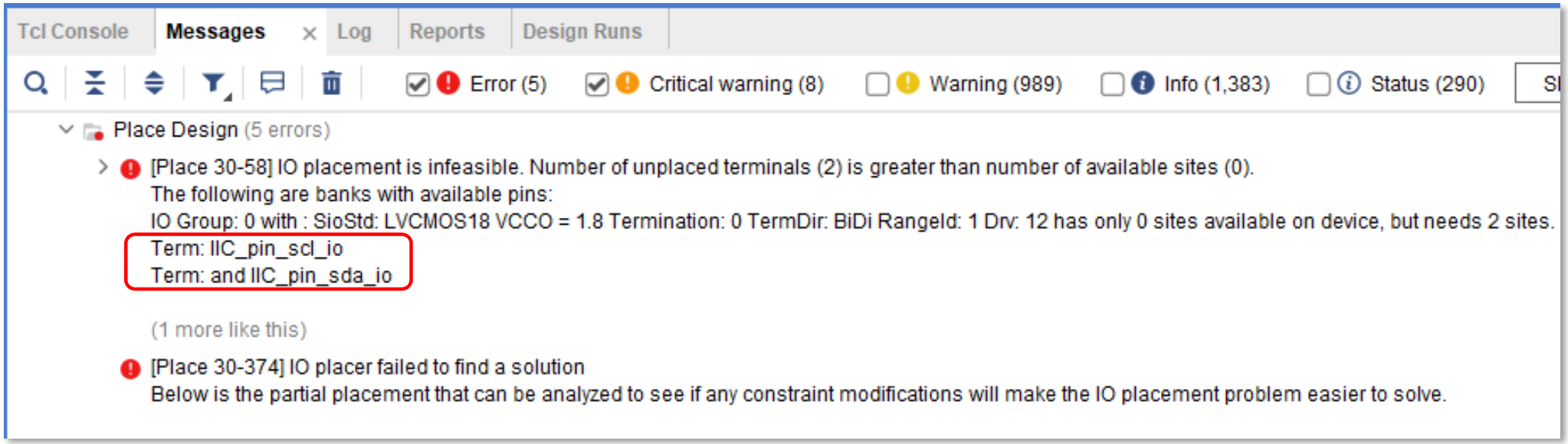
    - Majd - **Open Synthesized Design** , OK

- Utolsó lépésként a külső porthoz (`iic_pin`) hozzá kell rendelni az FPGA IO lábakat!
  - Layout menü -> I/O Planning layout nézet



IO planning – hibás lábkiosztás

- Amennyiben az előző lépésben nem a szintetizált tervet nyitjuk meg, hanem elindítjuk a „*Run Implementation*”-t akkor az alábbi hibaüzenetet kapjuk:



- Oka: a Vivado megpróbálja elhelyezni a két I^2C jelet.
  - Hibás pin location-t és IO Standard-et rendelne (LVCMOS 1.8V) hozzá.

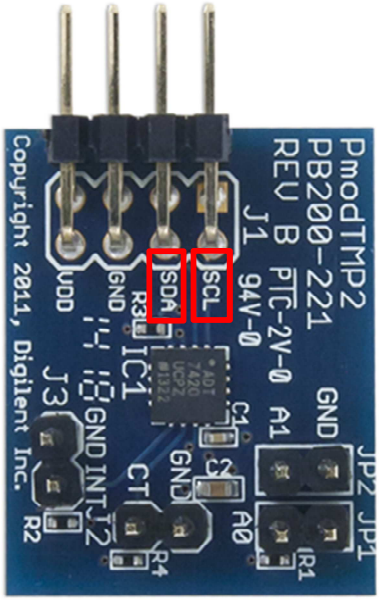
ZyBo - PMOD konnektorok

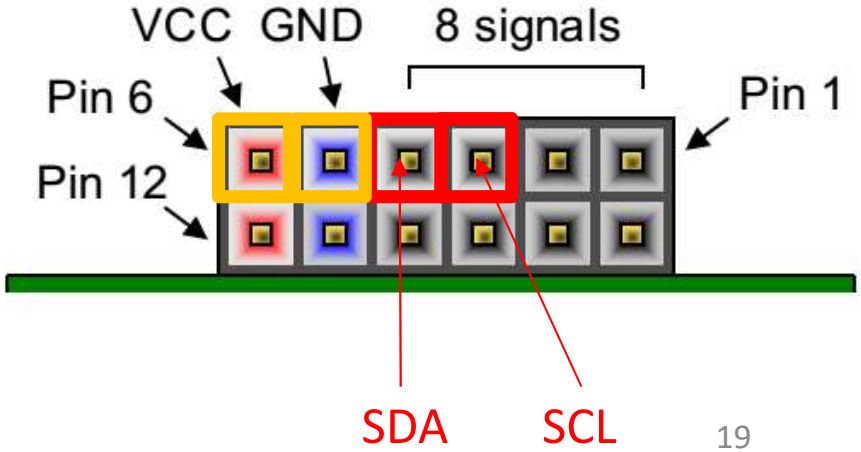
Pmod JA (XADC)	Pmod JB (Hi-Speed)	Pmod JC (Hi-Speed)	Pmod JD (Hi-Speed)	Pmod JE (Std.)	Pmod JF (MIO)
JA1: N15	JB1: T20	JC1: V15	JD1: T14	JE1: V12	JF1: MIO-13
JA2: L14	JB2: U20	JC2: W15	JD2: T15	JE2: W16	JF2: MIO-10
JA3: K16	JB3: V20	JC3: T11	JD3: P14	JE3: J15	JF3: MIO-11
JA4: K14	JB4: W20	JC4: T10	JD4: R14	JE4: H15	JF4: MIO-12
JA7: N16	JB7: Y18	JC7: W14	JD7: U14	JE7: V13	JF7: MIO-0
JA8: L15	JB8: Y19	JC8: Y14	JD8: U15	JE8: U17	JF8: MIO-9
JA9: J16	JB9: W18	JC9: T12	JD9: V17	JE9: T17	JF9: MIO-14
JA10: J14	JB10: W19	JC10: U12	JD10: V18	JE10: Y17	JF10: MIO-15

Legyen a **JE** jelű standard PMOD konn.

i2c_pin_scl : J15

i2c_pin_sda: H15





https://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPZYBO/documentation/ZYBO_RM_B_V6.pdf

IO planning – helyes lábkiosztás I.

- a.) Egyik lehetőség az I/O planning használata (GUI)

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew
IIC_58957 (2)	INOUT			☐		default (LVCMOS18)	1.800		12	SLOW
Scalar ports (2)										
IIC_pin_scl_io	INOUT			☐		default (LVCMOS18)	1.800		12	SLOW
IIC_pin_sda_io	INOUT			☐		default (LVCMOS18)	1.800		12	SLOW

Zybo_master.xdc alapján is megadható lenne a helyes lábkiosztás, melyet a következőkkel kell kiegészíteni:

- Site:
  - **i2c_pin_scl : J15**
  - **i2c_pin_sda: H15**
- IOSTANDARD: **LVCMOS33**
- Pull type: **PULLUP**
- OCT (On-Chip-Termination): **NONE**

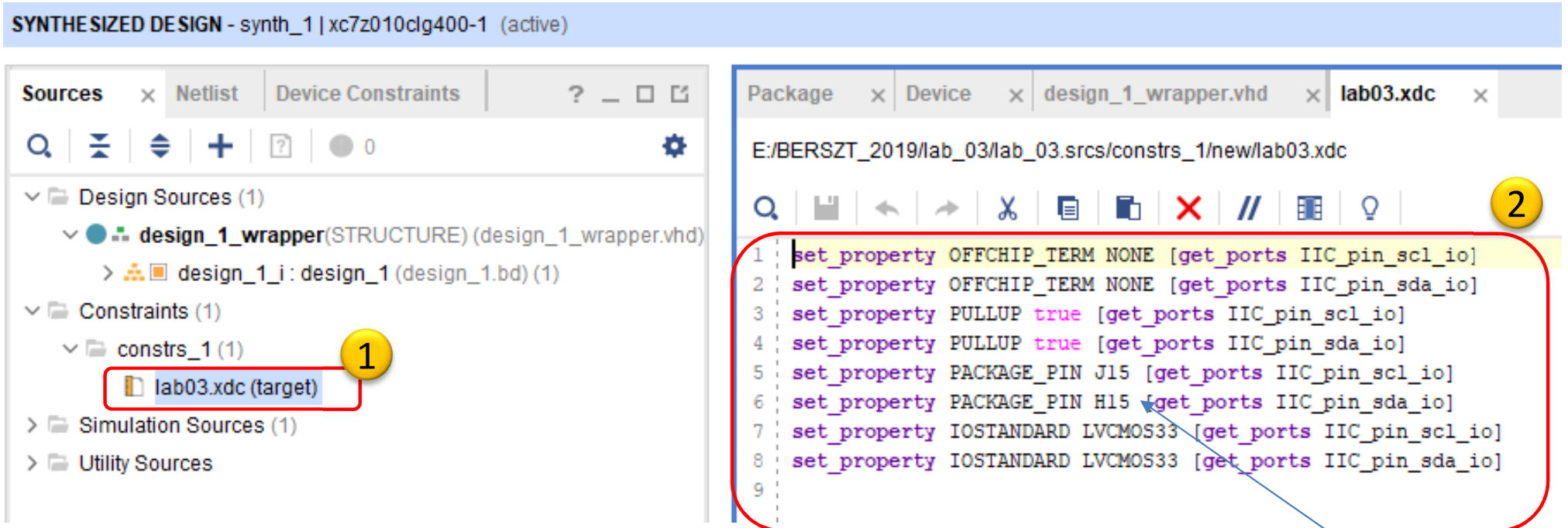
Tcl Console | Messages | Log | Reports | Design Runs | Package Pins | I/O Ports

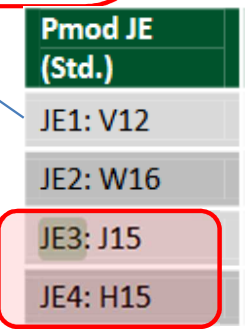
Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination
IIC_58957 (2)	INOUT		1	☑	35	LVCMOS33* 2	3.300		12	SLOW	PULLUP	NONE
Scalar ports (2)												
IIC_pin_scl_io	INOUT		J15	☑	35	LVCMOS33*	3.300		12	SLOW 3	PULLUP	NONE 4
IIC_pin_sda_io	INOUT		H15	☑	35	LVCMOS33*	3.300		12	SLOW	PULLUP	NONE

Végül ***File -> Save Constraints*** vagy **CTRL+S**. Ezután rákérdez az XDC file elmentésére. Adjunk neki nevet: pl. „`lab03.xdc`”

IO planning – helyes lábkiosztás II.

- b.) Másik lehetőség az .xdc manuális szerkesztése
  - File -> Add Sources… -> Add or Create constraints
  - Create File
  - Adjunk neki nevet: pl. „`lab03.xdc`”





Végül ***File -> Save Constraints*** vagy **CTRL+S**. Ezután rákérdez az XDC file elmentésére.

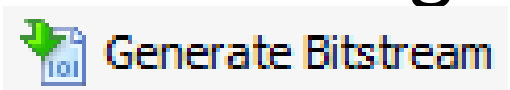
Implementáció és bitstream generálás

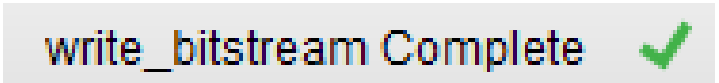
- Flow Navigator menü → Run Implementation

  - Kiszűri az esetleges elkötéseket, hibákat,
  - Figyelmeztető (warning) jellegű üzenetek megengedettek (implementálható a terv),
  - Legtöbb lebegő (floating) vezetékkel sem kell foglalkozni (pl. Peripheral Reset, stb).
  - Miközben dolgozik – hosszabb idő - érdemes megnézni a fordítási riportokat!

Ezután indítható el a Bitstream generálás.

- Flow Navigator → Generate Bitstream.





Xilinx Vivado SDK

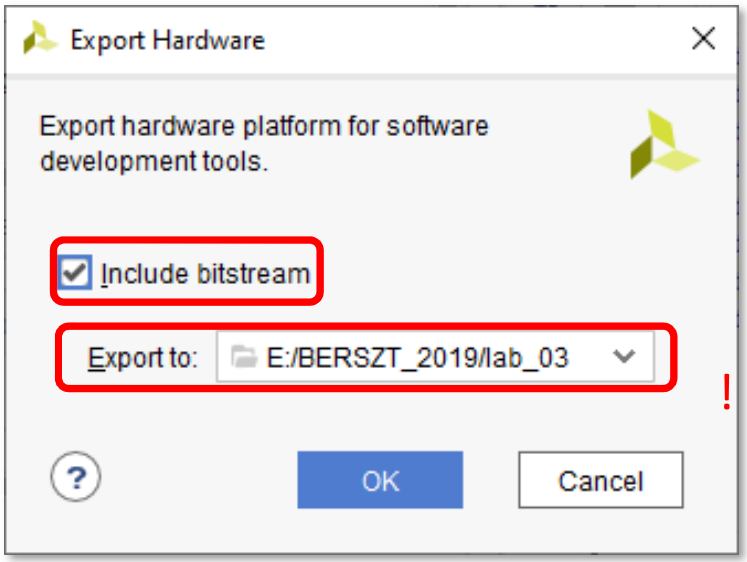
SZOFTVER TESZT ALKALMAZÁS ÖSSZEÁLLÍTÁSA (PMOD_TMP2)

SDK – alkalmazás-fejlesztés általános lépései

1. Vivado projekt létrehozása, majd Export → **SDK**,
2. **Új,** vagy C/C++ template-ből generálható alkalmazás létrehozása (TestApp PMOD 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. (zip!)
3. Soros terminál beállítása (USB-soros port beállítása),
4. JTAG-USB programozó csatlakoztatása és beállítása,
5. *FPGA konfigurálása!! (.bit, mivel PL oldal is használt)*
6. *‚debug' – hardveres hibakeresés* beállítása
7. Debug (breakpoint-ok beszúrása, léptetés, stb.)

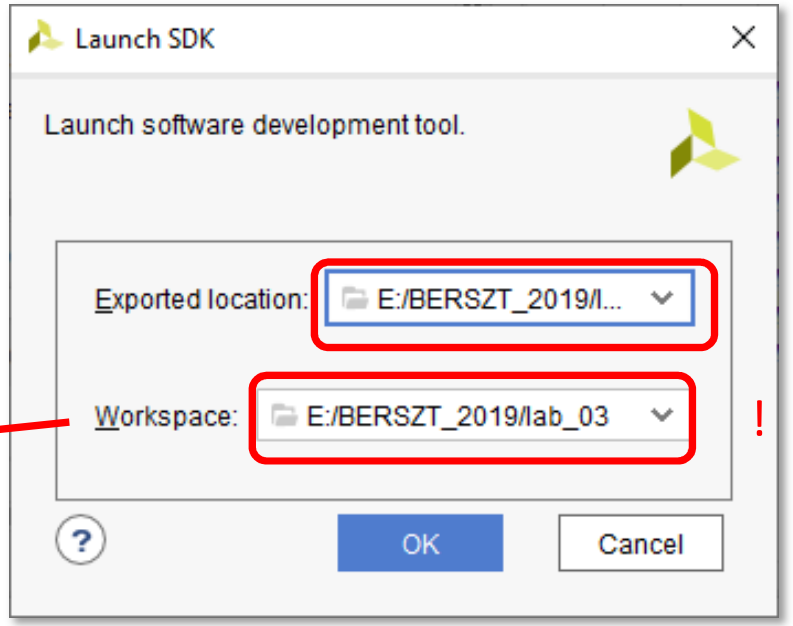
Export HW -> Vivado SDK

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



Mivel PL (FPGA) oldal is konfigurálva lett, ezért most már kell a bitstream is az SDK-hoz. *Choose Location*... **lab03** kiválasztása!

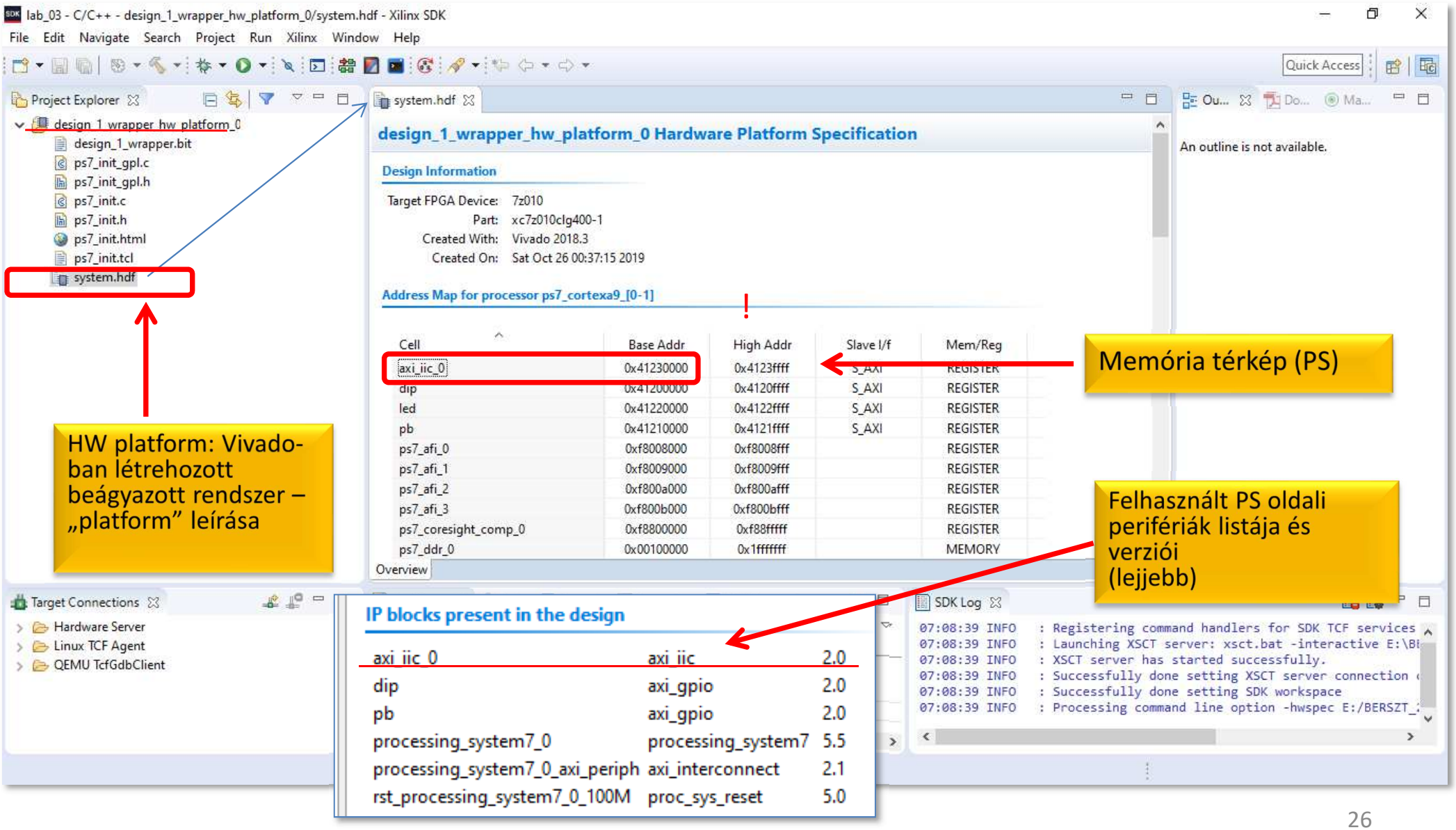
- **File -> Launch SDK, OK.**
  - Vivado SDK indítása a keretrendszerből (ezt kívülről indítva is meg lehetne tenni)



Fontos, hogy a workspace-t lehetőleg ne az előző projektbe hozza létre! - > *Choose location*... – és mostani **lab03** kiválasztása

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.

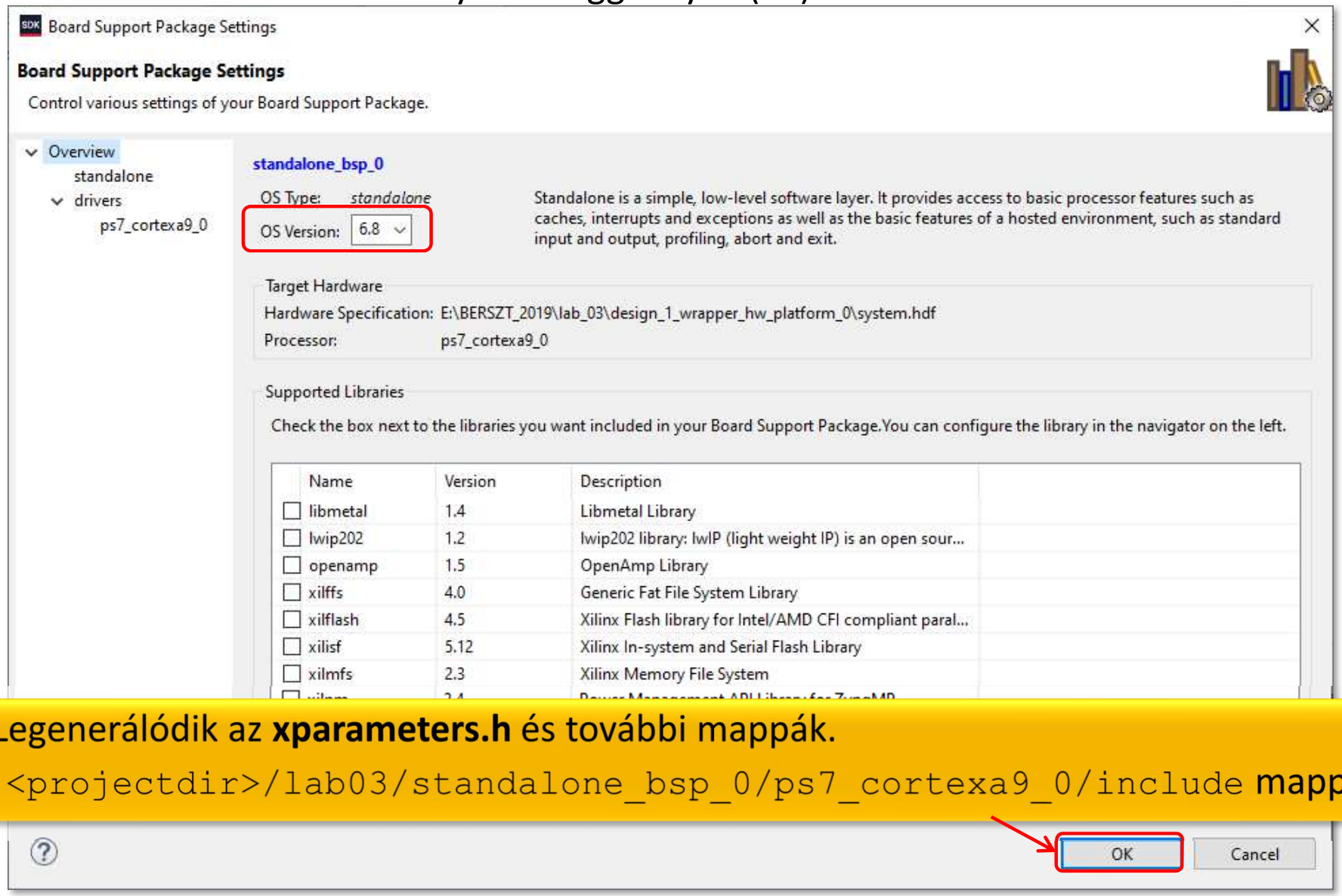


Tesztalkalmazás (TestApp) készítése I.

- 1.) A Launch SDK - **Choose Location…** opciójával válasszuk ki a `<project_dir>/LAB03/` → majd kattintsunk az **OK**-ra.
- 2.) Az SDK-ban: **File** → **New** → **Board Support Package** → kattintsunk a **Finish**-re.
  - Új BSP neve `standalone_bsp_0` is maradhat,
  - Hagyjunk mindent alap értéken.

Tesztalkalmazás (TestApp) készítése II./a

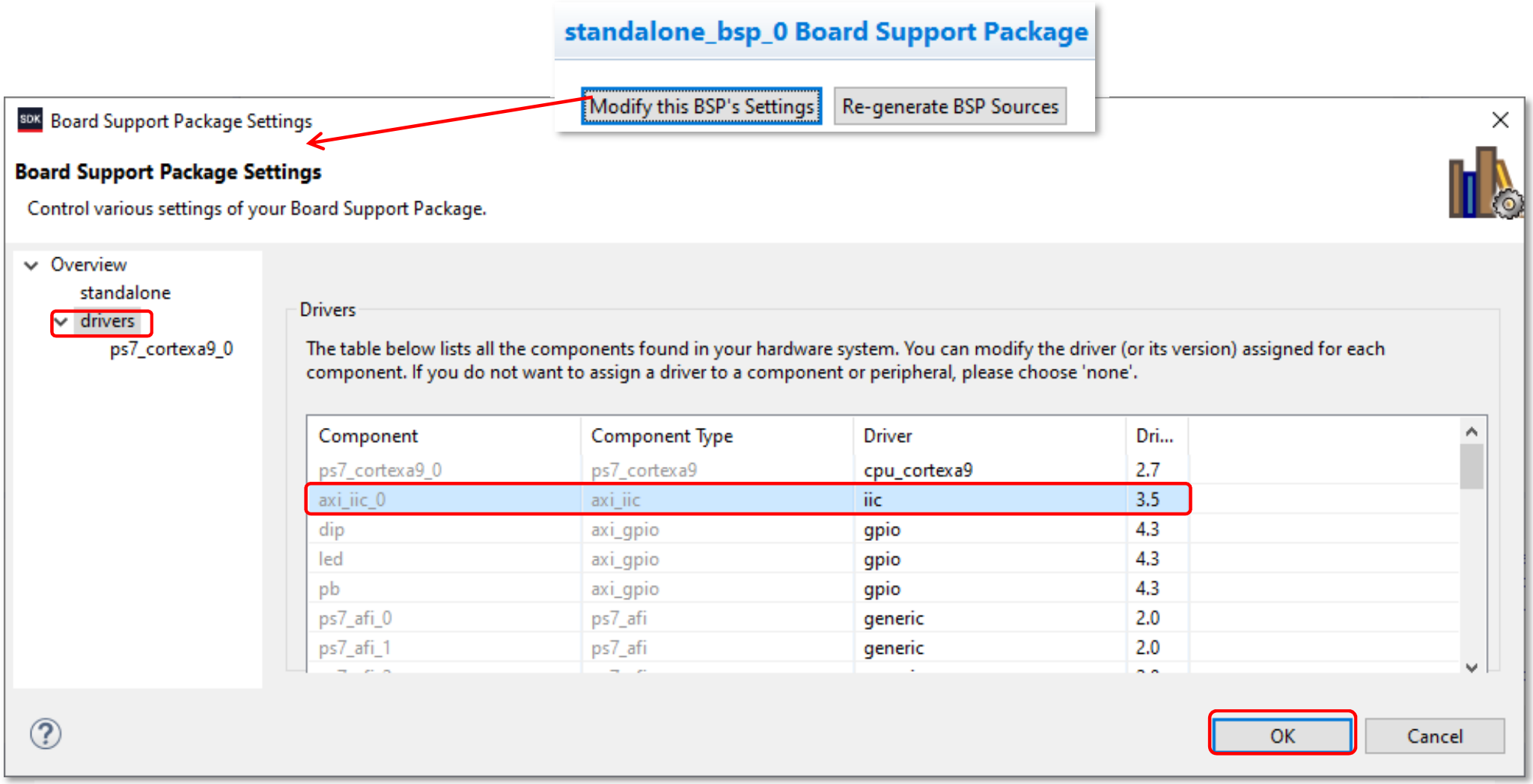
- **Software Platform Settings**
  - Operációs rendszer kiválasztása: *standalone* vs. *xilkernel* (esetleg 3rd Party OS)
  - Rendelkezésre álló könyvtári függvények (lib) kiválasztása



OK -> Legenerálódik az **xparameters.h** és további mappák.

Helye: `<projectdir>/lab03/standalone_bsp_0/ps7_cortexa9_0/include` mappa

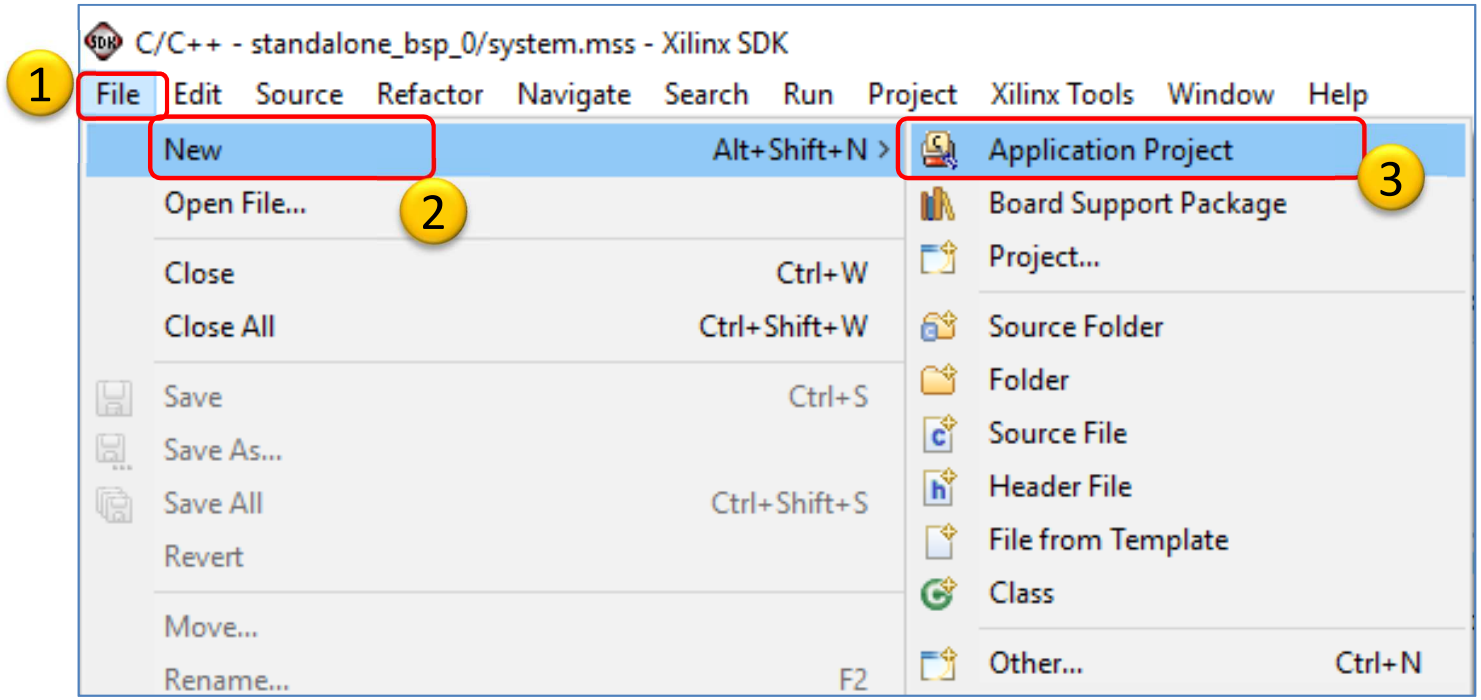
Tesztalkalmazás (TestApp) készítése II./b



Fontos: axi_iic_0 perifériához a none/generic driver helyett az „iic” drivert kell kiválasztani a helyes működéshez (ha nem ez lenne!)

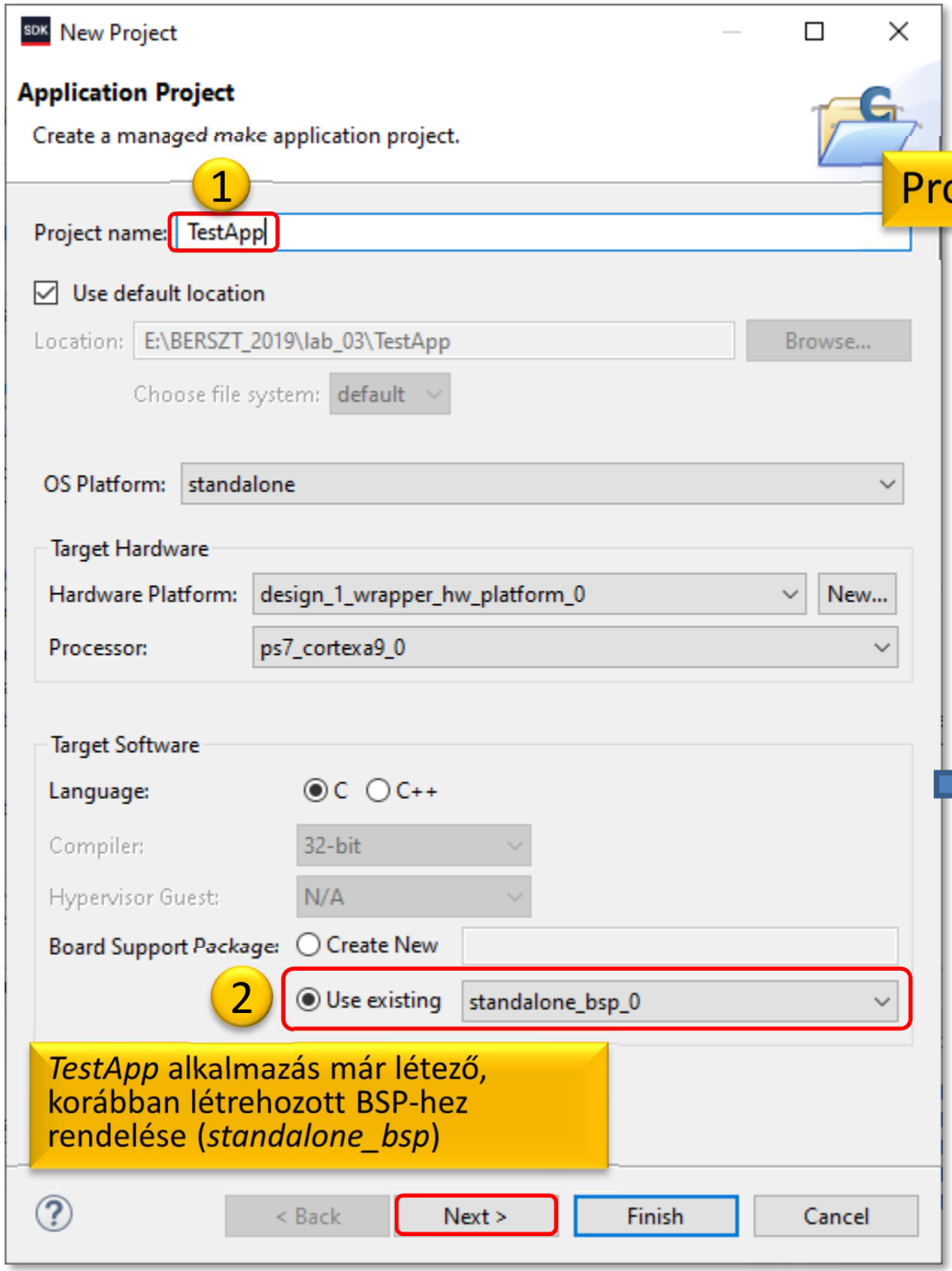
Tesztalkalmazás (TestApp) készítése III.

- Új projekt alkalmazás létrehozása (File –> New –> Application Project)



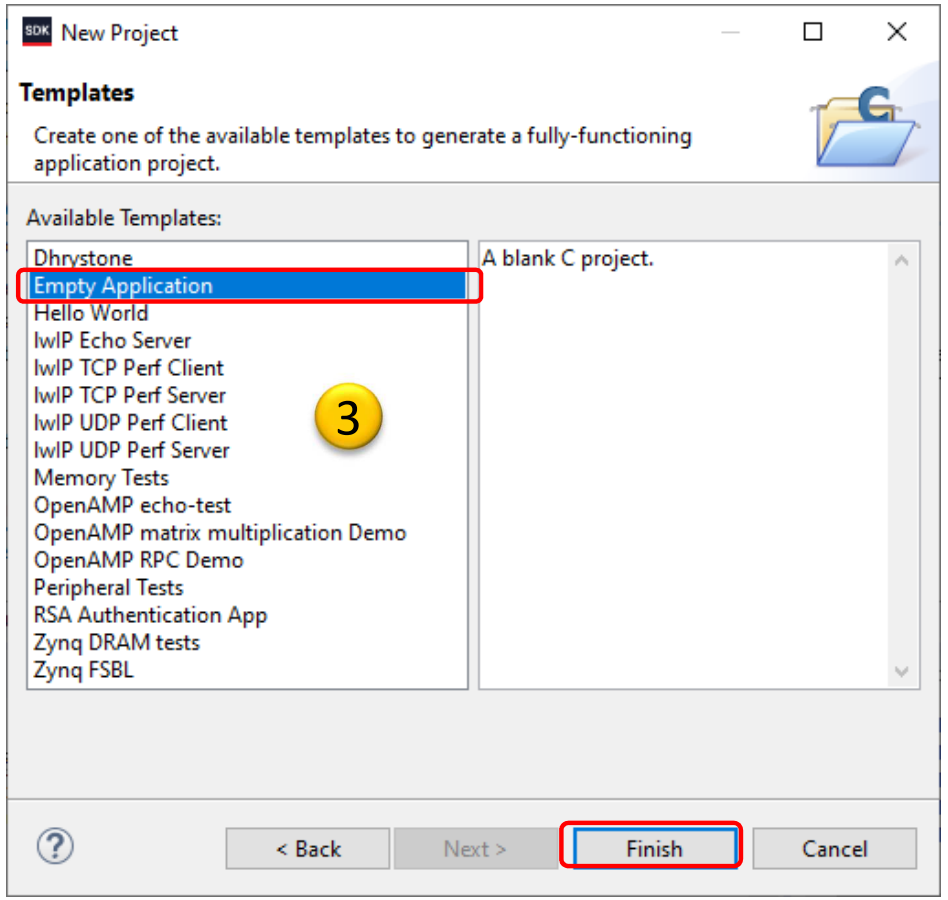
BSP: *standalone_bsp_0* (alapértelmezett)

Tesztalkalmazás (TestApp) készítése IV.



Projekt neve: *TestApp*

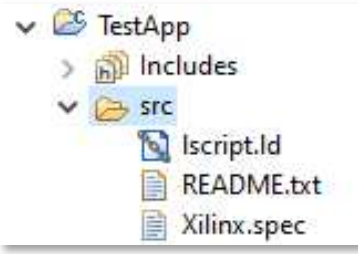
TestApp alkalmazás már létező, korábban létrehozott BSP-hez rendelése (*standalone_bsp*)

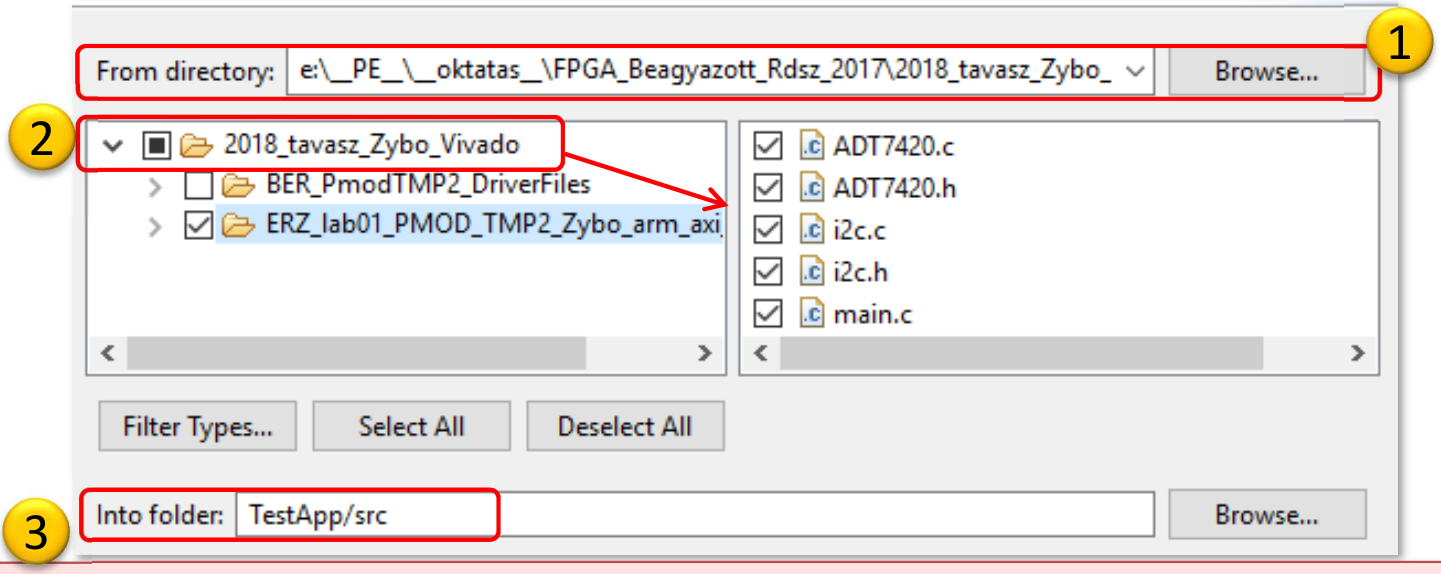


Tesztalkalmazás (TestApp) készítése V.

Forrás fájl(ok) importálása: TestApp/src kijelölése!

- File -> Import … -> General -> Archive File-> NEXT!





BER_PmodTMP2_DriverFiles.zip * -ből hozzáadni:
- ADT7420.c / .h
- I2c.c / .h
- Main.c

* Tárgy oldaláról letölthető: BER_PmodTMP2_DriverFiles.zip

- Linker script generálása (.ld): OnChip RAM vs. külső MEM.
- SW alkalmazás fordítása (.elf)
- Tesztverifikáció: bitfile generálás + tesztelés FPGA-n.

*Megjegyzés: az i2c.c és i2.h fájlok az Analog Devices támogató csomagjából lettek felhasználva. Alternatív megoldásként a Xilinx iic.c és iic.h fájlokat is használhatjuk:
Lásd: <Xilinx install_dir>/…/sw/XilinxProcessorIPLib/drivers/iic_vX_YZ_a/doc/html/api/index.html

TestApp forráskód hibáinak javítása

1.)

```
In file included from ../src/ADT7420.c:49:0:
../src/ADT7420.c: In function 'ADT7420_Init':
../src/ADT7420.h:49:22: error: 'XPAR_AXI_IIC_BASEADDR' undeclared (first use in this function)
 #define IIC_BASEADDR XPAR_AXI_IIC_BASEADDR
                      ^
```

– Megoldás: adjuk meg az XPAR_AXI_IIC_BASEADDR #define helyes nevét az `xparameters.h` file alapján

2.)

```
../src/ADT7420.h:53:28: error: expected expression before '<' token
 #define ADT7420_IIC_ADDR   <address>
                            ^
```

– Megoldás: nézzük meg a ADT7420_IIC_ADDR helyes címét (korábbi fólia, v. kártya adatlap)

3.)

```
../src/main.c: In function 'main':
../src/main.c:78:12: error: 'XPAR_RS232_UART_1_BASEADDR' undeclared (first use in this function)
  Xil_Out32(XPAR_RS232_UART_1_BASEADDR+0x0C, (1 << 4));
```

– Megoldás: keressünk ilyen nevű define-t az `ADT7240.h` file-ban.

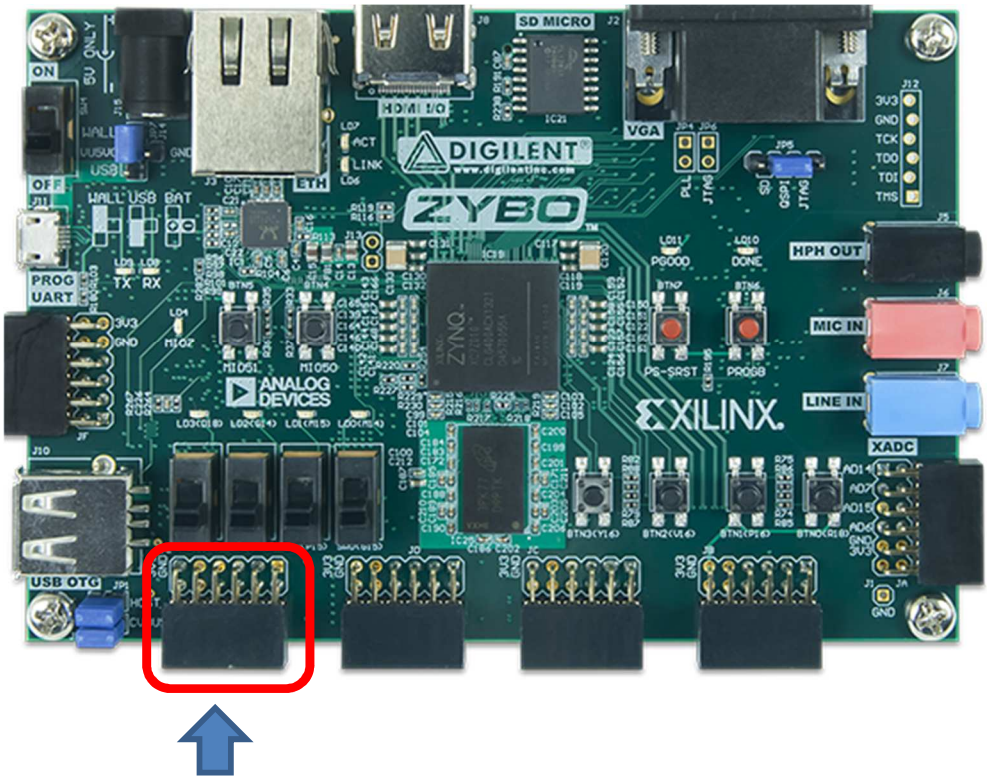
Kérdés:

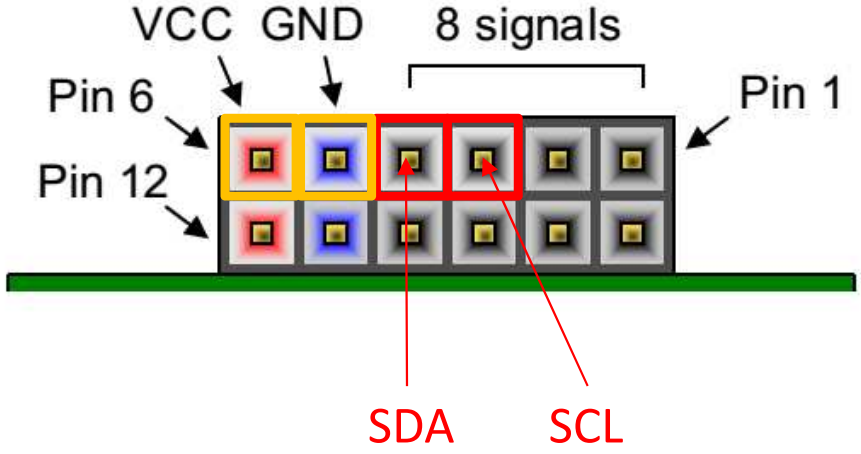
- Mekkora a letöltendő .elf file mérete?
- Megoldás:

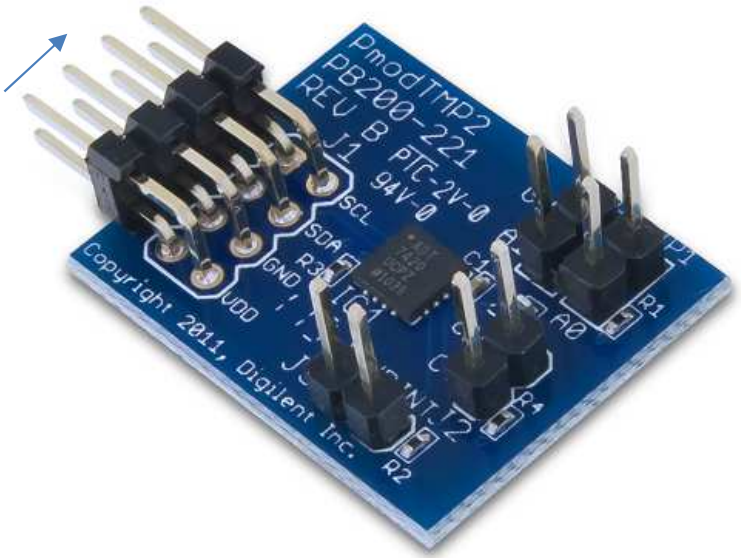
```
'Invoking: ARM Print Size'
arm-xilinx-eabi-size TestApp.elf  |tee
"TestApp.elf.size"
   text    data     bss     dec     hexfilename
  27780    1160   22580   51520    c940TestApp.elf
'Finished building: TestApp.elf.size'
```

Teszt verifikáció I.

- Kártya helyes csatlakoztatása PMOD JE-re.





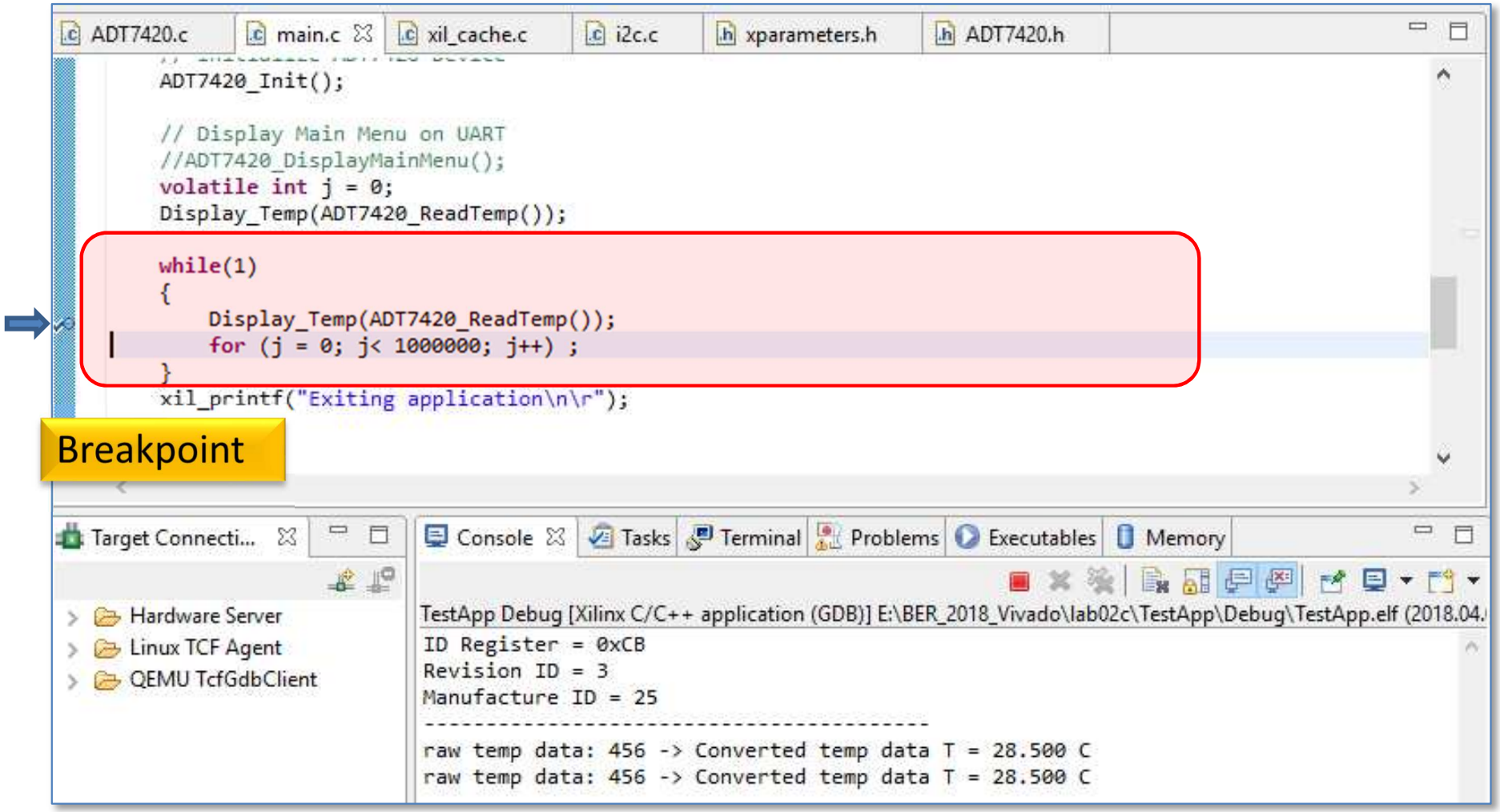


Teszt verifikáció II.

- 1.) Digilent PMOD_TMP2 hőmérséklet mérő kártya megfelelő illesztése a Digilent ZyBo kártyához:
  - **Standard JE** PMOD foglalat felső sorának jobbról 3-4-es lábaira (SCL – SDA)
  - PMOD_TMP2-es kártyán a JP1/JP2 jumperek rövidre zártak? (A0/A1 címek kiosztása ennek megfelelően !!)
- 2.) A JTAG-USB programozó/táp (USB-soros port) csatlakoztatása a kártyához és a számítógéphez,
- 3.) Digilent ZyBo kártya bekapcsolása,
- **4.) Xilinx Tools → Program FPGA**
  - **.bit fájl kiválasztása,**
  - **FPGA programozása,**
- **5.) Debug konfiguráció: környezet beállítása és indítása:**
  - Consol-ra az USB soros port beállítása (COMx),
  - Debug config-ban megadható akár az FPGA .bit file konfigurációja is (Entire reset).

Teszt verifikáció III.

- A forráskódból a következő rész felel az aktuális hőmérséklet kiíratásáért:
  - `Display_Temp(ADT7420_ReadTemp());`
  - Javaslat: tegyük a függvényt végtelen ciklusba a folyamatos kiíratáshoz.



Eredmény (debug log)

- Készítsük el annak Debug Configuration-ját, majd Debug-oljuk az alkalmazást.

```
ID Register = 0xCB
Revision ID = 3
Manufacture ID = 25
---------------------------------------
raw temp data: 456 -> Converted temp data T = 28.500 C
raw temp data: 456 -> Converted temp data T = 28.500 C
...
```