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

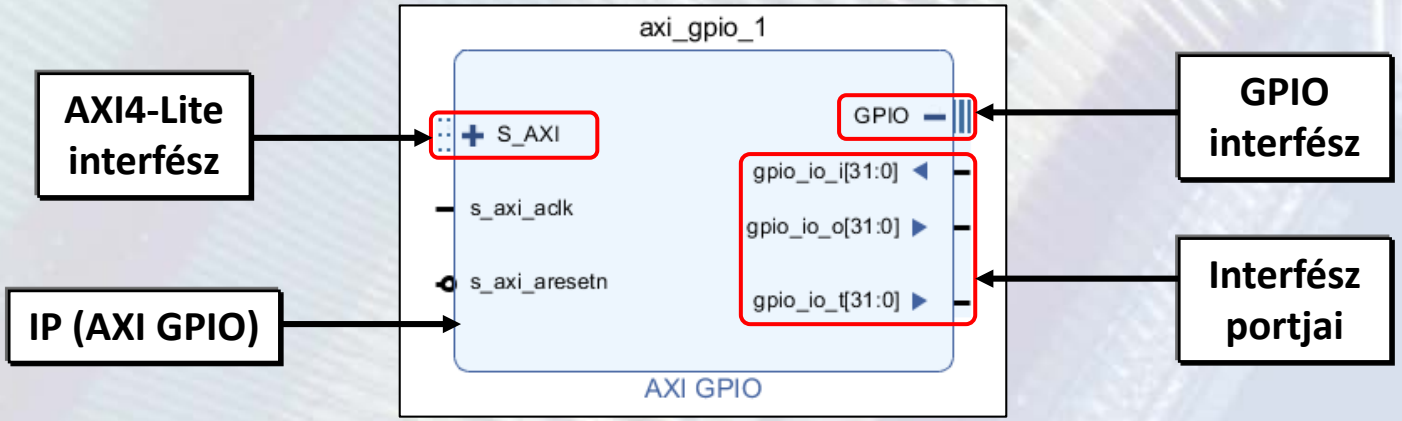
Mikrorendszerek tervezése

Saját IP készítése, periféria illesztés

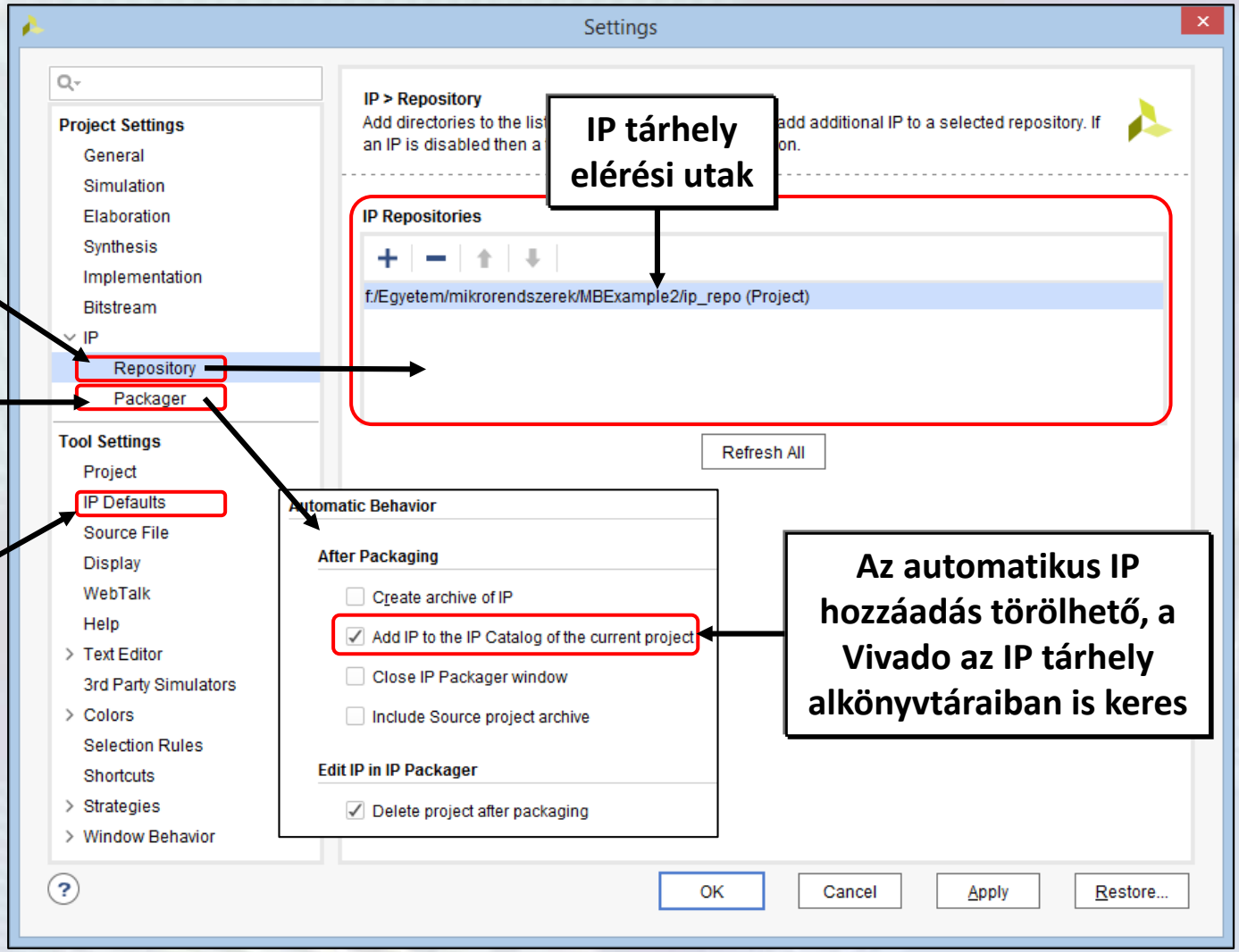
Fehér Béla
Raikovich Tamás

Vivado IP Repository beállítások

- **IP Repository: az IP-ket és az interfészeket tárolja**
- **Gyári IP-k és interfészek helye:**
  [telepítési könyvtár]\Vivado\[verzió]\data\ip
- **Projekt IP tárhely beállítások**
  - Flow Navigator panel: Project Manager → Settings
  - Saját IP tárhely megadása
  - Alapértelmezett IP tárhely (új projekthez)



Vivado IP Repository beállítások



Projekt IP tárhely beállítások

IP Packager beállítások

Alapértelmezett IP beállítások (új projekthez)

IP tárhely elérési utak

Az automatikus IP hozzáadás törölhető, a Vivado az IP tárhely alkönyvtáraiban is keres

Periféria illesztési feladat

- **A periféria típusa alapján az igények felmérése**
  - Regiszterek száma és elérése (írható, olvasható)
    - Parancs regiszter, státusz regiszter
    - Üzemmód regiszter, adatregiszter
    - Megszakítás engedélyező és flag regiszterek, stb.
  - Esetleg FIFO vagy kisebb memória blokk
- **Bonyolultabb perifériák esetén**
  - Burst képes slave interfész
  - Burst képes master interfész
  - Ezekkel itt nem foglalkozunk

Periféria illesztési feladat

- **A címtartomány használatának megtervezése**
  - Általában 2^N bájt méretű címtartomány
  - Regiszterek és memória blokkok elhelyezése
- **Példa: 4 bites bemeneti periféria megszakítással**
  - Három 32 bites regiszter → 16 bájtos címtartomány

Adatregiszter
BÁZIS+0x00

31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
0	0	.....	0	IN3	IN2	IN1	IN0
R	R	R	R	R	R	R	R

Megszakítás eng. reg.
BÁZIS+0x04

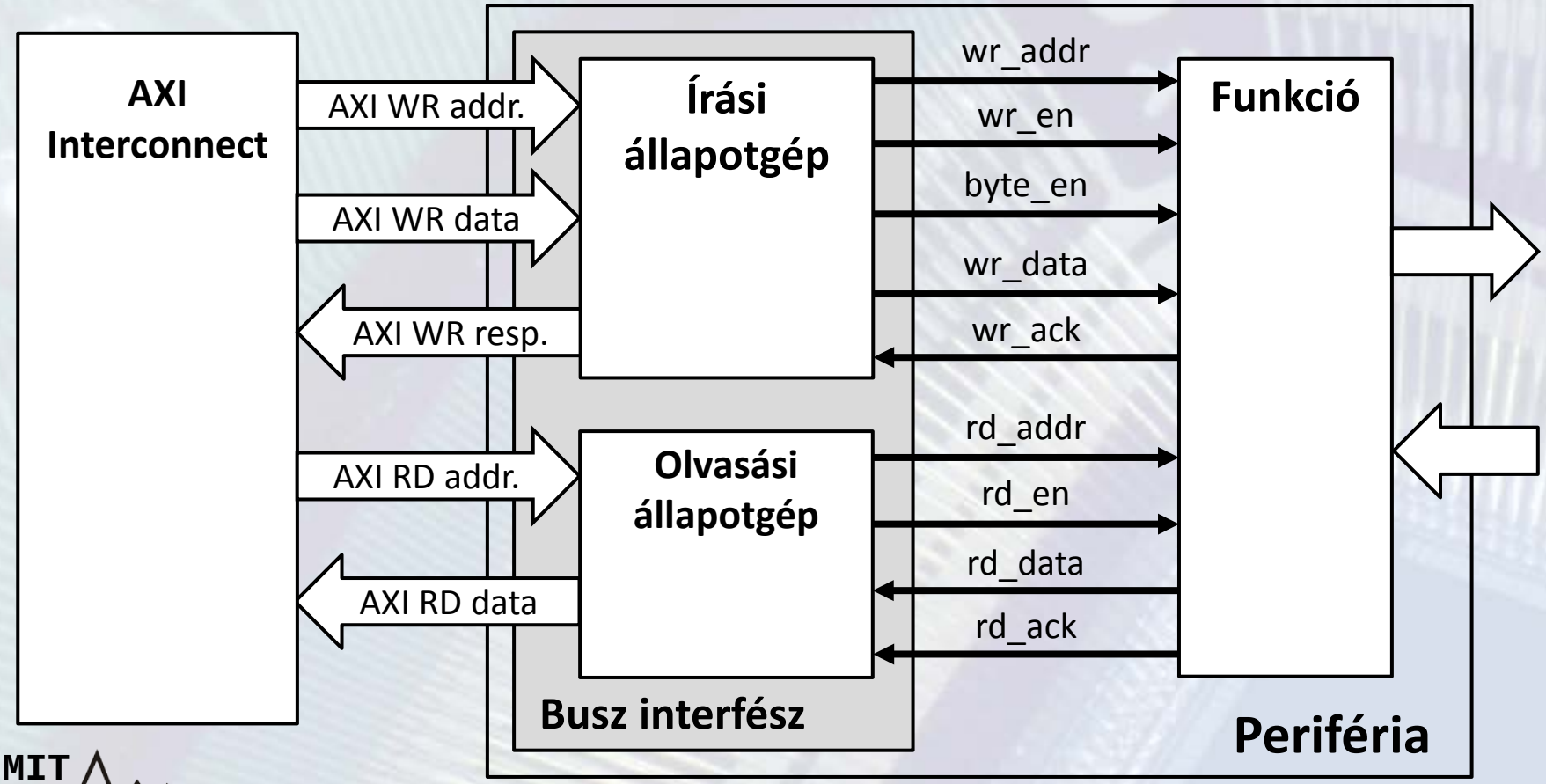
31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
0	0	.....	0	IE3	IE2	IE1	IE0
R	R	R	R	R/W	R/W	R/W	R/W

Megszakítás flag reg.
BÁZIS+0x08

31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
0	0	.....	0	IF3	IF2	IF1	IF0
R	R	R	R	R/W1C	R/W1C	R/W1C	R/W1C

Periféria illesztési feladat

- Példa: periféria egyedi busz interfész megvalósítással
- A Vivado nem ilyen AXI busz interfészt generál



Periféria illesztési feladat

Busz protokolltól független írási interfész:

- **2^N bájtos címtartomány, 2^M bájtos memória ($M \leq N$)**
- **AXI4-Lite busz interfész → 32 bites adatok**
- ***wr_addr[N-1:2]*: írási cím (wr_addr[1:0] nem használt)**
- ***wr_en*: írási adatátvitel jelzése**
  - Egyedi írás engedélyező jelek előállítása
    - Regiszter: ***reg_wr[i] = wr_en & (wr_addr[N-1:2]==(ADDRESS1 >> 2)) & …***
    - Memória: ***mem_wr[j] = wr_en & (wr_addr[N-1:M]==(ADDRESS2 >> M))***
- ***byte_en[3:0]*: bájt engedélyező jelek**
  - Vizsgálandó a 8, 16 vagy 32 bites regiszter írás eldöntéséhez
  - A memória rendelkezik bájt engedélyező bemenettel
- ***wr_data[31:0]*: írási adat**
- ***wr_ack*: írás nyugtázása (opcionális várakozás)**

Periféria illesztési feladat

Busz protokolltól független olvasási interfész:

- **2^N bájtos címtartomány, 2^M bájtos memória (M ≤ N)**
- **AXI4-Lite busz interfész → 32 bites adatok**
- ***rd_addr[N-1:2]*: olvasási cím (rd_addr[1:0] nem használt)**
- ***rd_en*: olvasási adatátvitel jelzése**
  - Egyedi olvasás engedélyező jelek előállítása
    - Regiszter: ***reg_rd[i] = rd_en & (rd_addr[N-1:2] == (ADDRESS1 >> 2))***
    - Memória: ***mem_rd[j] = rd_en & (rd_addr[N-1:M] == (ADDRESS2 >> M))***
    - A kimeneti multiplexer vezérléséhez
    - Kell még, ha az olvasás állapotváltozást okoz (pl. FIFO, bit törlés, stb.)
- ***rd_data[31:0]*: olvasási adat**
- ***rd_ack*: olvasás nyugtázása (opcionális várakozás)**

Saját periféria készítése

- **Specifikáció**
  - Általános célú I/O (GPIO) periféria
  - Paraméterrel megadható port méret (1 – 32 bit)
  - Bemenet változása esetén megszakításkérés
- **Regiszterkészlet, programozói interfész**
  - Kimeneti adatregiszter: BÁZIS+0x00, 32 bit, R/W
    - A (kimenetbe kapcsolt) port biteken megjelenő érték

31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
OUT31	OUT30	.....	OUT4	OUT3	OUT2	OUT1	OUT0
R/W	R/W		R/W	R/W	R/W	R/W	R/W

  - Bemeneti adatregiszter: BÁZIS+0x04, 32 bit, RD
    - A GPIO port aktuális állapotát tartalmazza

31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
IN31	IN30	.....	IN4	IN3	IN2	IN1	IN0
R	R		R	R	R	R	R

Saját periféria készítése

- **Regiszterkészlet, programozói interfész**
  - Irányregiszter: BÁZIS+0x08, 32 bit, R/W
    - A port bitek irányát adja meg (0: bemenet, 1: kimenet)

31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
DIR31	DIR30	.....	DIR4	DIR3	DIR2	DIR1	DIR0
R/W	R/W		R/W	R/W	R/W	R/W	R/W

  - Megszakítás eng. regiszter: BÁZIS+0x0C, 32 bit, R/W
    - Megszakításkérés engedélyezése a bemeneti port bitekre

31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
IE31	IE30	.....	IE4	IE3	IE2	IE1	IE0
R/W	R/W		R/W	R/W	R/W	R/W	R/W

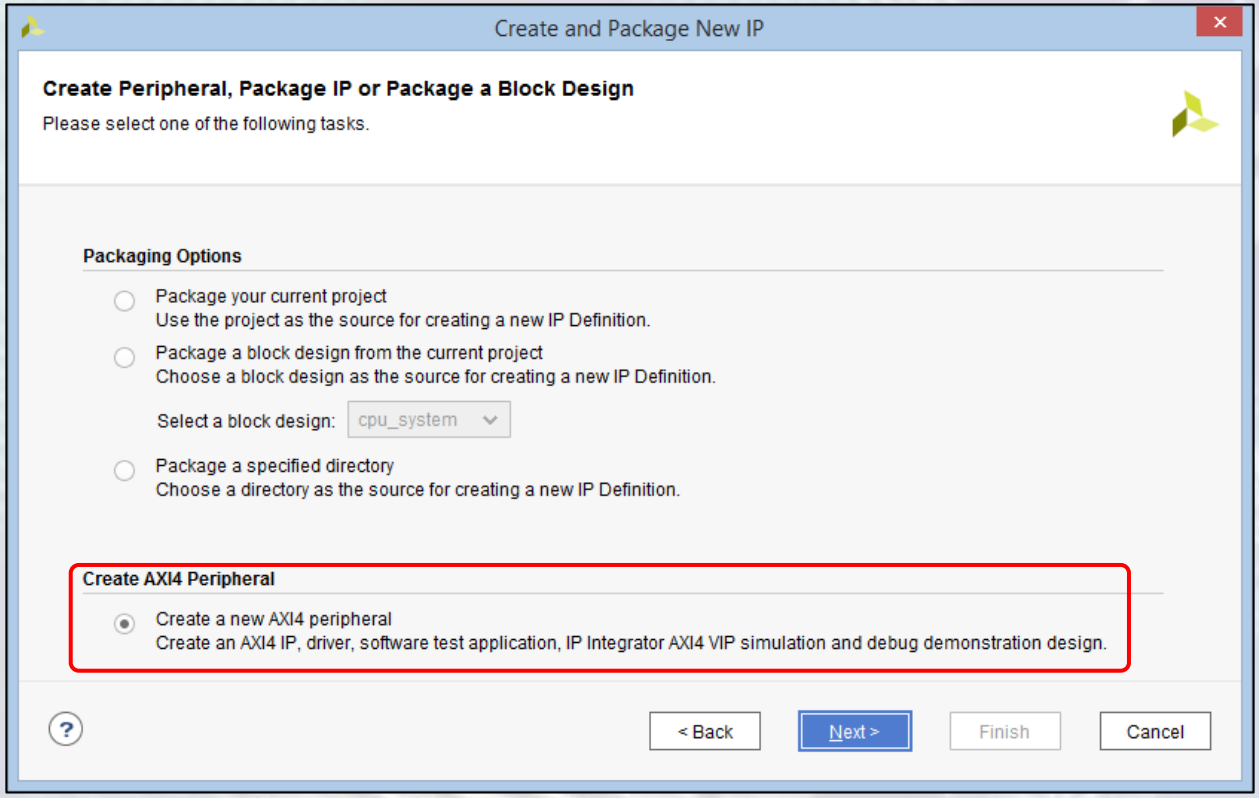
  - Megszakítás flag regiszter: BÁZIS+0x10, 32 bit, R/W
    - A bemeneti port bitek változását jelzi, 1 beírásával törölhető
    - A jelzés független a megszakítás engedélyezettségétől

31. bit	30. bit	.....	4. bit	3. bit	2. bit	1. bit	0. bit
IF31	IF30	.....	IF4	IF3	IF2	IF1	IF0
R/W1C	R/W1C		R/W1C	R/W1C	R/W1C	R/W1C	R/W1C

Saját periféria készítése

Új AXI4 periféria létrehozása

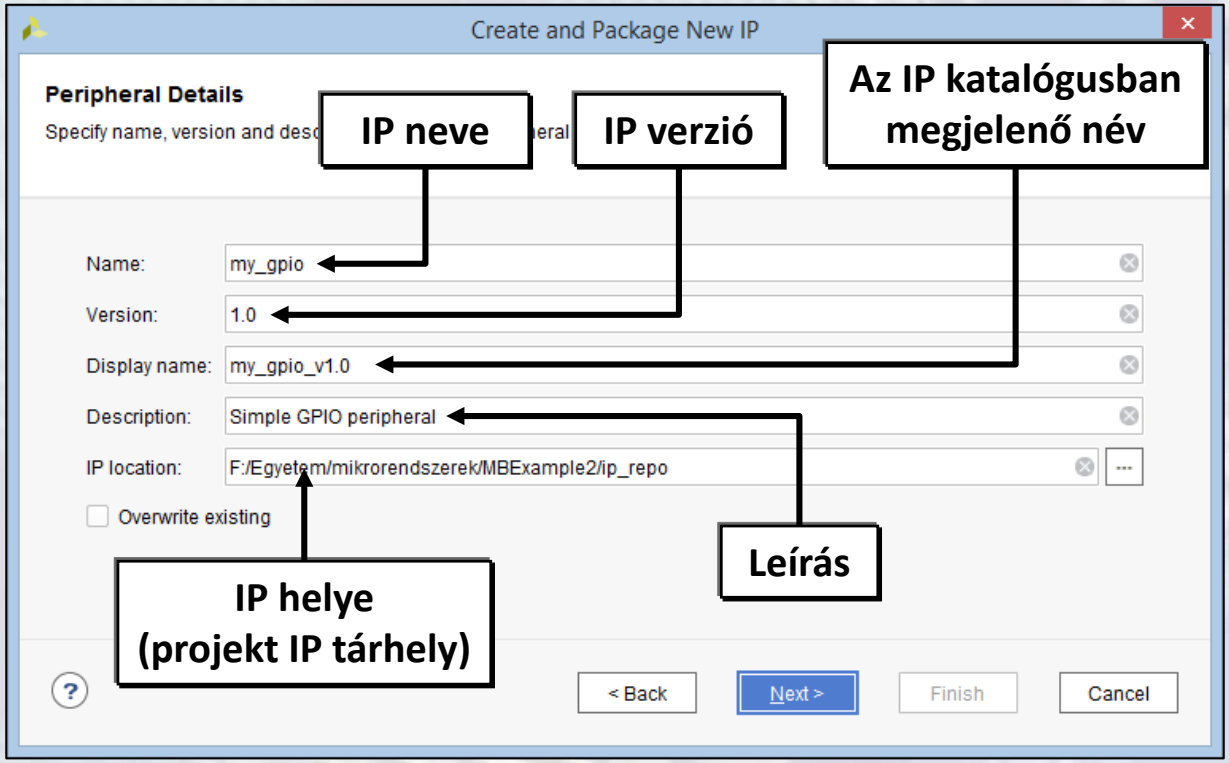
- Tools → Create and Package New IP…



Saját periféria készítése

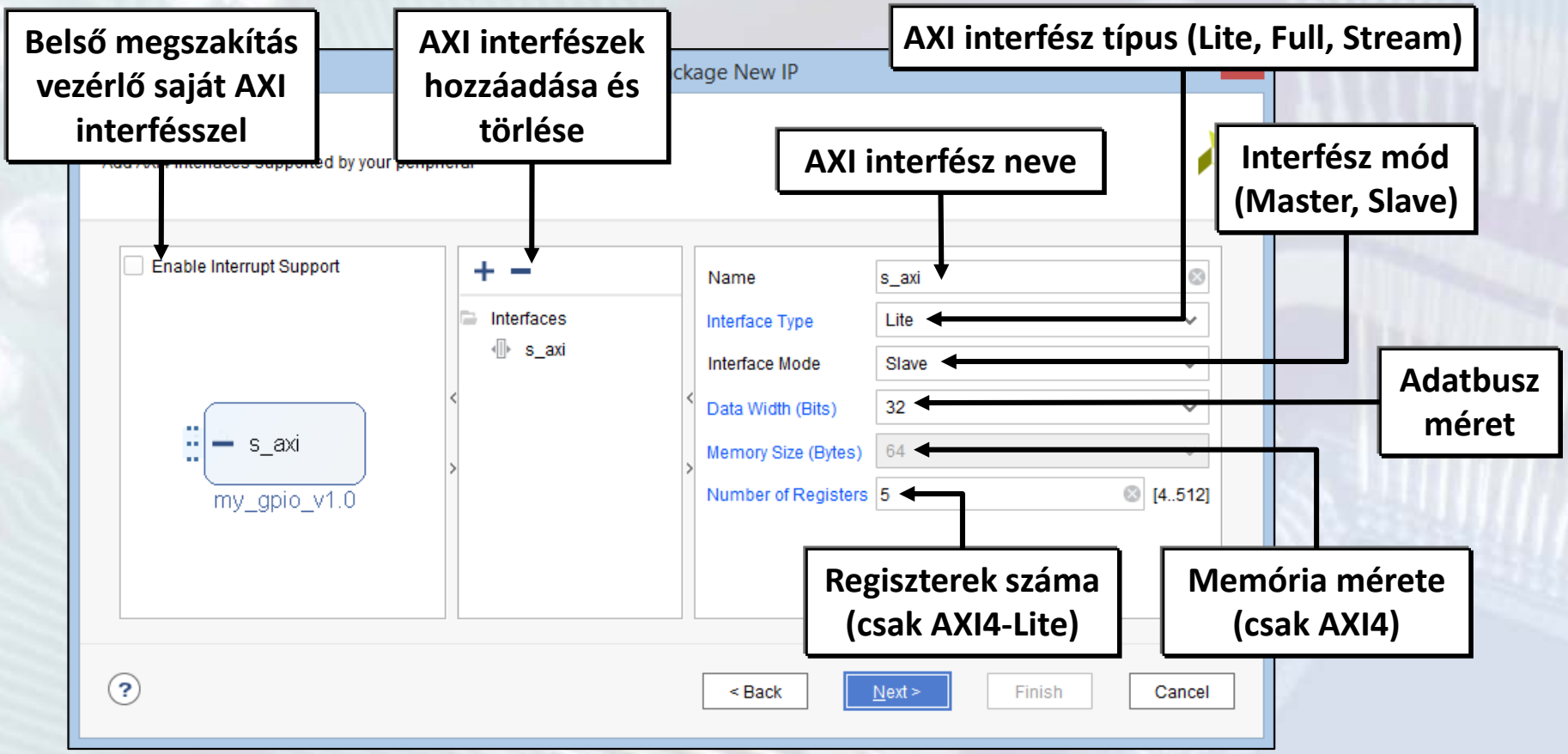
A periféria adatainak megadása

- IP neve: ***my_gpio***, verziója: ***1.0***
- Az IP könyvtára: [IP location]\[name]_[version]



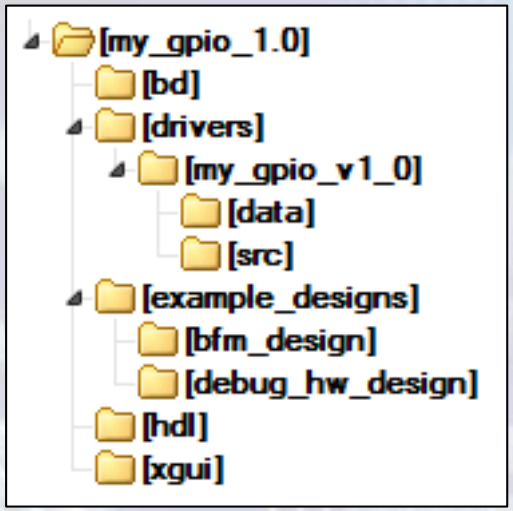
Saját periféria készítése

AXI interfész hozzáadása: AXI4-Lite slave, 5 regiszter



Saját periféria készítése

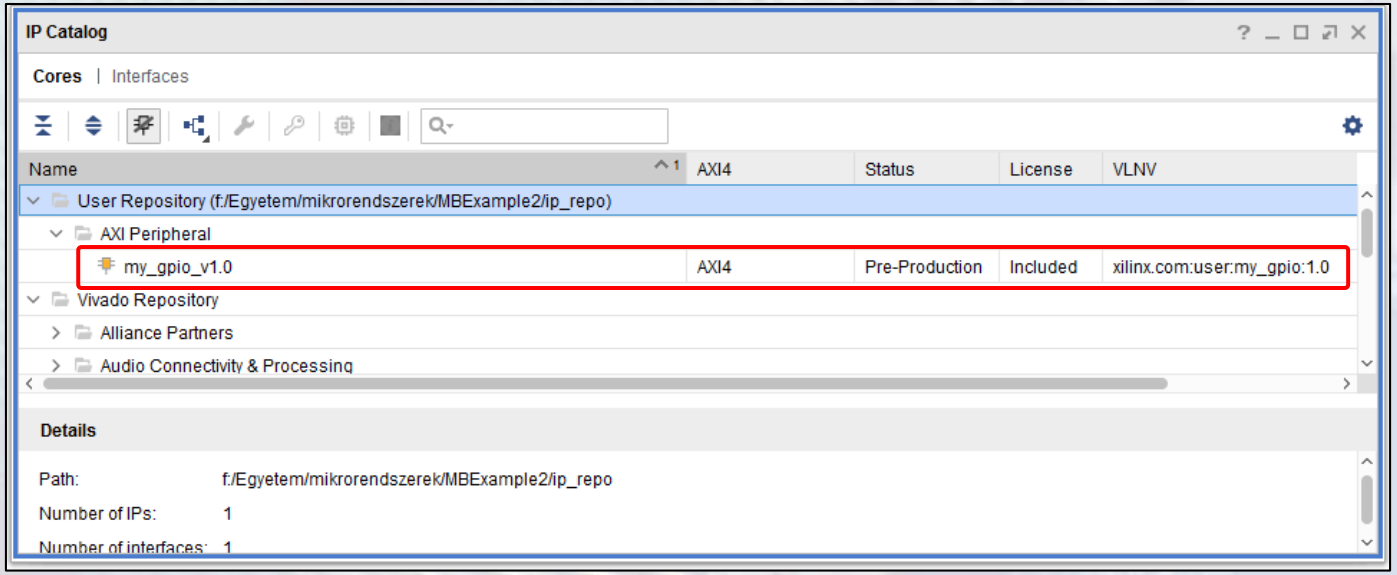
Az IP-hez tartozó könyvtár tartalma



- ***component.xml*** **fájl: IP-XACT leírás**
- ***bd*** **könyvtár: blokk diagram TCL szkript**
- ***drivers*** **könyvtár**
  - SDK és szoftver forrásfájlok (C kód)
  - Egyszerű regiszter és memória elérés
  - Egyszerű önteszt
- ***example_designs*** **könyvtár: példák**
  - Nem minden típusú perifériához jön létre
- ***hdl*** **könyvtár: HDL forrásfájlok**
- ***xgui*** **könyvtár: konfigurációs GUI TCL szkript**

Saját periféria készítése

- **Az új periféria bekerül az IP katalógusba**
- **A periféria szerkesztése, módosítása**
  - Jobb klikk az IP-n → Edit in IP Packager
  - Új projekt jön létre a szerkesztéshez és megnyitásra kerül egy új Vivado ablakban



Saját periféria készítése

- **A generált forrásfájlok (a név függ a megadott adatoktól)**
  - ***my_gpio_v1_0.v***: top-level modul
  - ***my_gpio_v1_0_S_AXI.v***: AXI4-Lite slave interfész
- **Az AXI interfész modul módosítása**
  - Paraméterek és portok hozzáadása, szükségtelen jelek törlése

```
// Users to add parameters here
// A GPIO port bitszáma.
parameter C_GPIO_WIDTH = 32,
// User parameters ends
```

```
// Users to add ports here
output reg  [C_GPIO_WIDTH-1:0] gpio_dout,
input  wire [C_GPIO_WIDTH-1:0] gpio_din,
output reg  [C_GPIO_WIDTH-1:0] gpio_dir,
output reg  [C_GPIO_WIDTH-1:0] int_enable,
output reg  [C_GPIO_WIDTH-1:0] int_flag,
input  wire [C_GPIO_WIDTH-1:0] int_flag_set,
// User ports ends
```

```
reg […] slv_reg0;
reg […] slv_reg1;
reg […] slv_reg2;
reg […] slv_reg3;
reg […] slv_reg4;

integer byte_index;
```

Saját periféria készítése

Az AXI interfész modul módosítása

- Írható regiszterek, csak 32 bites írás lehetséges
- Az eredeti regiszter implementáció helyére kerülnek

```
assign slv_reg_wren = …… ;
wire [2:0] reg_sel  = axi_awaddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB];
wire       dword_wr = (S_AXI_WSTRB == 4'b1111);
integer    i;

always @(posedge S_AXI_ACLK)      //Kimeneti adatregiszter.
  if (S_AXI_ARESETN == 1'b0)
    gpio_dout <= {C_GPIO_WIDTH{1'b0}};
  else if (slv_reg_wren && (reg_sel == 3'd0) && dword_wr)
    gpio_dout <= S_AXI_WDATA[C_GPIO_WIDTH-1:0];

always @(posedge S_AXI_ACLK)      //Irányregiszter.
  if (S_AXI_ARESETN == 1'b0)
    gpio_dir <= {C_GPIO_WIDTH{1'b0}};
  else if (slv_reg_wren && (reg_sel == 3'd2) && dword_wr)
    gpio_dir <= S_AXI_WDATA[C_GPIO_WIDTH-1:0];
```

Saját periféria készítése

Az AXI interfész modul módosítása

- Írható regiszterek, csak 32 bites írás lehetséges
- Megszakítás flag regiszter: bemenet változása beállítja a jelzést, 1 beírása pedig törli

```
always @(posedge S_AXI_ACLK)      //Megszakítás engedélyező regiszter.
  if (S_AXI_ARESETN == 1'b0)
    int_enable <= {C_GPIO_WIDTH{1'b0}};
  else if (slv_reg_wren && (reg_sel == 3'd3) && dword_wr)
    int_enable <= S_AXI_WDATA[C_GPIO_WIDTH-1:0];

always @(posedge S_AXI_ACLK)      //Megszakítás flag regiszter.
  for (i = 0; i < C_GPIO_WIDTH; i = i + 1)
    if (S_AXI_ARESETN == 1'b0)
      int_flag[i] <= 1'b0;        //Reset törli a jelzést.
    else if (int_flag_set[i])
      int_flag[i] <= 1'b1;        //Változás esetén a jelzés beállítása.
    else
      if (slv_reg_wren && (reg_sel == 3'd4) && dword_wr && S_AXI_WDATA[i])
        int_flag[i] <= 1'b0;      //1 beírása törli a jelzést.
```

Saját periféria készítése

- **Az AXI interfész modul módosítása**
  - A regiszter olvasás megvalósítása

```
always @(*)
begin
  // Address decoding for reading registers
  case ( axi_araddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB] )
    3'h0   : reg_data_out <= slv_reg0;        reg_data_out <= gpio_dout;
    3'h1   : reg_data_out <= slv_reg1;        reg_data_out <= gpio_din;
    3'h2   : reg_data_out <= slv_reg2;        reg_data_out <= gpio_dir;
    3'h3   : reg_data_out <= slv_reg3;        reg_data_out <= int_enable;
    3'h4   : reg_data_out <= slv_reg4;        reg_data_out <= int_flag;
    default: reg_data_out <= 0;
  endcase
end
```

- **A top-level modul módosítása**
  - Saját paraméterek hozzáadása

```
// Users to add parameters here
parameter C_GPIO_WIDTH = 32,
// User parameters ends
```

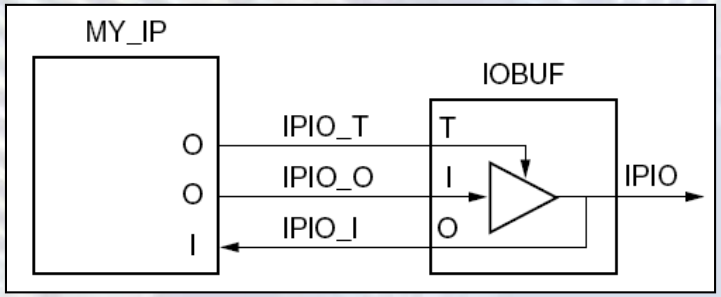
Saját periféria készítése

A top-level modul módosítása

- **Saját portok hozzáadása**

```
// Users to add ports here
input  wire [C_GPIO_WIDTH-1:0] gpio_I,
output wire [C_GPIO_WIDTH-1:0] gpio_O,
output wire [C_GPIO_WIDTH-1:0] gpio_T,
output reg                     irq,
// User ports ends
```

- **Háromállapotú kimenet vagy I/O vonal megadása**
  - Kimenet (***_O***), bemenet (***_I***), kimenet engedélyezés (***_T***)
  - A ***_T*** jel aktív alacsony
  - Az IP és az I/O buffer portok összerendelése a név alapján történik



Saját periféria készítése

A top-level modul módosítása

- Az AXI interfész modul új portjainak bekötése

```
wire [C_GPIO_WIDTH-1:0] gpio_din, gpio_dir, int_enable;
wire [C_GPIO_WIDTH-1:0] int_flag, int_flag_set;

// Instantiation of Axi Bus Interface S_AXI
my_gpio_v1_0_S_AXI # (
  .C_GPIO_WIDTH(C_GPIO_WIDTH),
  .C_S_AXI_DATA_WIDTH(C_S_AXI_DATA_WIDTH),
  .C_S_AXI_ADDR_WIDTH(C_S_AXI_ADDR_WIDTH)
) my_gpio_v1_0_S_AXI_inst (
  .gpio_dout(gpio_O),
  .gpio_din(gpio_din),
  .gpio_dir(gpio_dir),
  .int_enable(int_enable),
  .int_flag(int_flag),
  .int_flag_set(int_flag_set),
  .S_AXI_ACLK(s_axi_aclk),
            ⋮
);
```

Saját periféria készítése

A top-level modul módosítása

- Mintavételezés, változás detektálás, megszakításkérés

```
//Add user logic here
//A bemenetek mintavételezése és szinkronizálása az órajelhez.
reg [C_GPIO_WIDTH-1:0] din_reg1, din_reg2, din_reg3;

always @(posedge s_axi_aclk)
  if (s_axi_aresetn == 1'b0)
    {din_reg3, din_reg2, din_reg1} <= {3*C_GPIO_WIDTH{1'b0}}
  else
    {din_reg3, din_reg2, din_reg1} <= {din_reg2, din_reg1, gpio_I};

assign gpio_T       = ~gpio_dir;
assign gpio_din     =  din_reg2;
assign int_flag_set = (din_reg2 ^ din_reg3) & ~gpio_dir;

//A megszakításkérő kimenet meghajtása (aktív magas).
always @(posedge s_axi_aclk)
  if (s_axi_aresetn == 1'b0) irq <= 1'b0;
  else                       irq <= |(int_enable & int_flag);
```

Saját periféria készítése

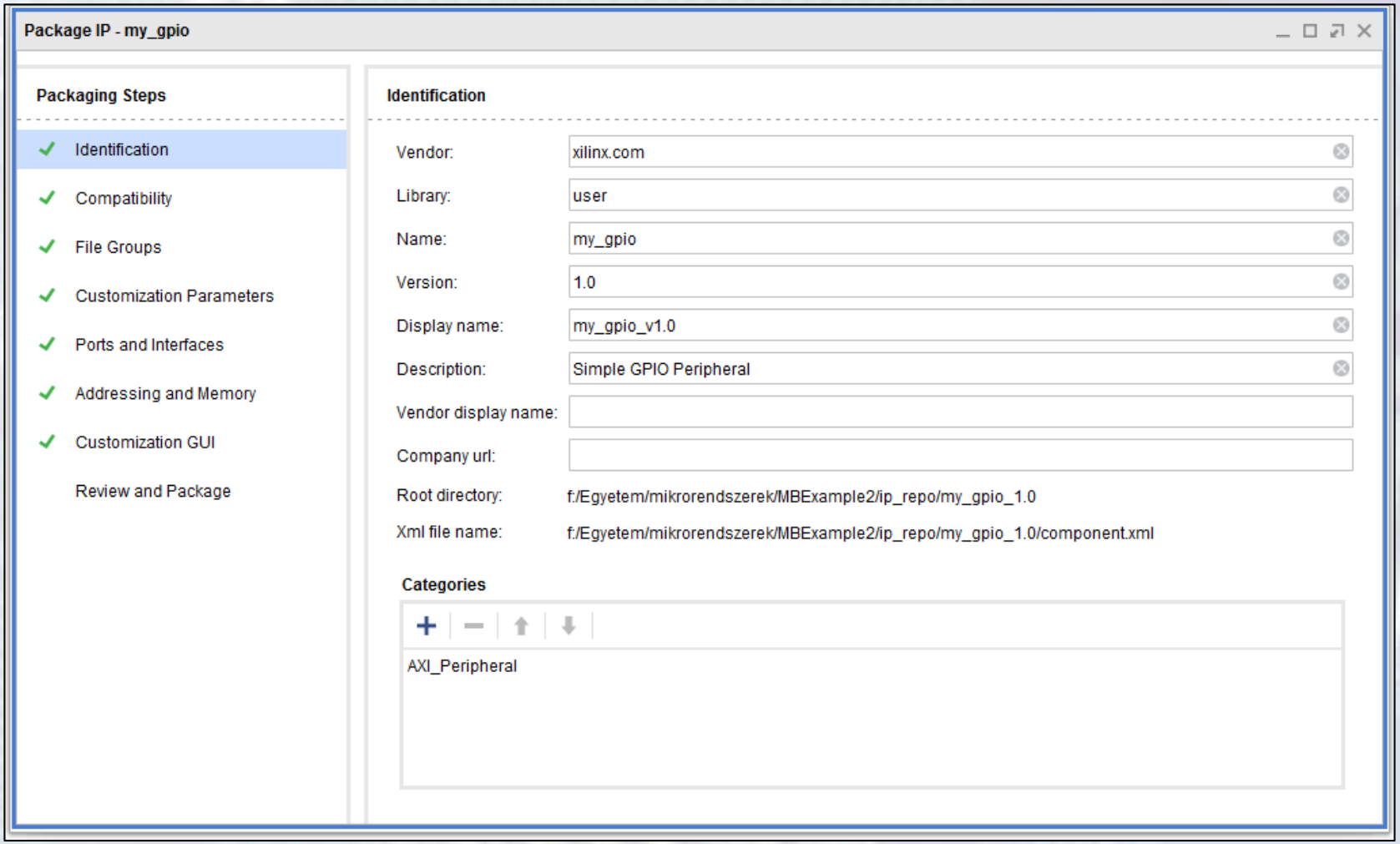
- A generált AXI4-Lite slave interfész modul egyszerű írható és olvasható regiszterek megvalósításához jól használható
- Bonyolultabb regiszterek (például megszakítás flag regiszter) és memória blokk illesztéséhez nem elég rugalmas, több módosítást igényel
- Ha erre van szükségünk, akkor érdemes lehet saját AXI interfészt készíteni a korábbi példák alapján
- A cím vonalak bitszáma a megfelelő paraméterrel (a példában C_S_AXI_ADDR_WIDTH) beállítható az igények szerint

IP Packager

- **Az IP-hez tartozó IP-XACT leírás szerkeszthető vele**
- **Az IP-XACT egy IEEE ipari szabvány XML formátum az IP-k metaadatokkal történő leírásához**
  - Portok, interfészek
  - Konfigurálható paraméterek
  - Fájlok, dokumentációk
- **Magasszintű leírás, tehát nem helyettesíti a HDL-t**
- **Lehetővé teszi**
  - Az automatikus összeköttetéseket, konfigurációt
  - Mástól származó IP integrálását
  - Saját IP exportálását

IP-XACT™

IP Packager


Package IP - my_gpio
Packaging Steps
Identification
Compatibility
File Groups
Customization Parameters
Ports and Interfaces
Addressing and Memory
Customization GUI
Review and Package
Identification
Vendor: xilinx.com
Library: user
Name: my_gpio
Version: 1.0
Display name: my_gpio_v1.0
Description: Simple GPIO Peripheral
Vendor display name:
Company url:
Root directory: f:/Egyetem/mikrorendszerek/MBExample2/ip_repo/my_gpio_1.0
Xml file name: f:/Egyetem/mikrorendszerek/MBExample2/ip_repo/my_gpio_1.0/component.xml
Categories
AXI_Peripheral

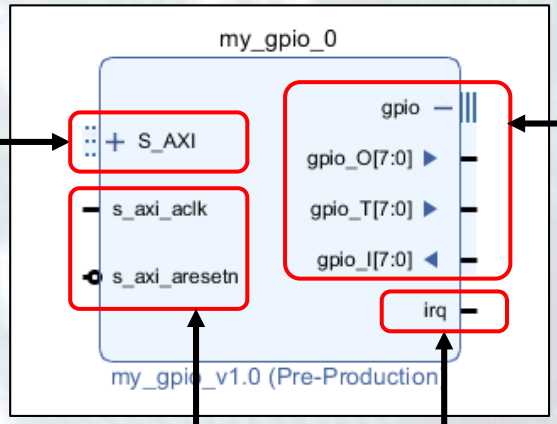
IP Packager

- **Elemzi a HDL forrásfájlokat, így egyes adatok és beállítások automatikusan kitöltésre kerülnek**
- ***Identification*: az IP-hez tartozó azonosító adatok**
- ***Compatibility*: FPGA-k és HDL szimulátorok megadása, melyekkel az IP használható**
- ***File Groups*: az IP-hez tartozó fájlok**
  - HDL forrás, szoftver meghajtó, dokumentáció, stb.
- ***Customization Parameters*: módosítható, beállítható IP paraméterek (HDL) tulajdonságai**
  - GUI megjelenés, típus, értékkészlet
  - A példában a *C_GPIO_WIDTH*: GUI-ban jelenjen meg, long típusú, minimális értéke 1, maximális értéke 32

IP Packager

- ***Ports and Interfaces*: IP port és interfész beállítások**
  - Lehetőleg használjuk a már meglévő interfészeket
  - Saját interfész: Tools → Create Interface Definition…
  - A példában az *irq* port automatikusan hozzárendelődik egy *irq* interfészhez (aktív magas → SENSITIVITY=LEVEL_HIGH)
  - A példában a *gpio_I*, *gpio_O* és *gpio_T* portokat rendeljük egy *gpio* interfészhez
- ***Addressing and Memory*: memória címtartomány adatok**
- ***Customization GUI*: az IP konfigurálásához tartozó grafikus felület szerkesztése**
  - A példában a *C_GPIO_WIDTH* paraméter legyen látható, húzzuk át a *Page 0* csomópont alá
- ***Review and Package*: összefoglalás, IP leírás generálása**
- **Vivado UG – Creating and Packaging Custom IP (UG1118)**

Az elkészült GPIO IP

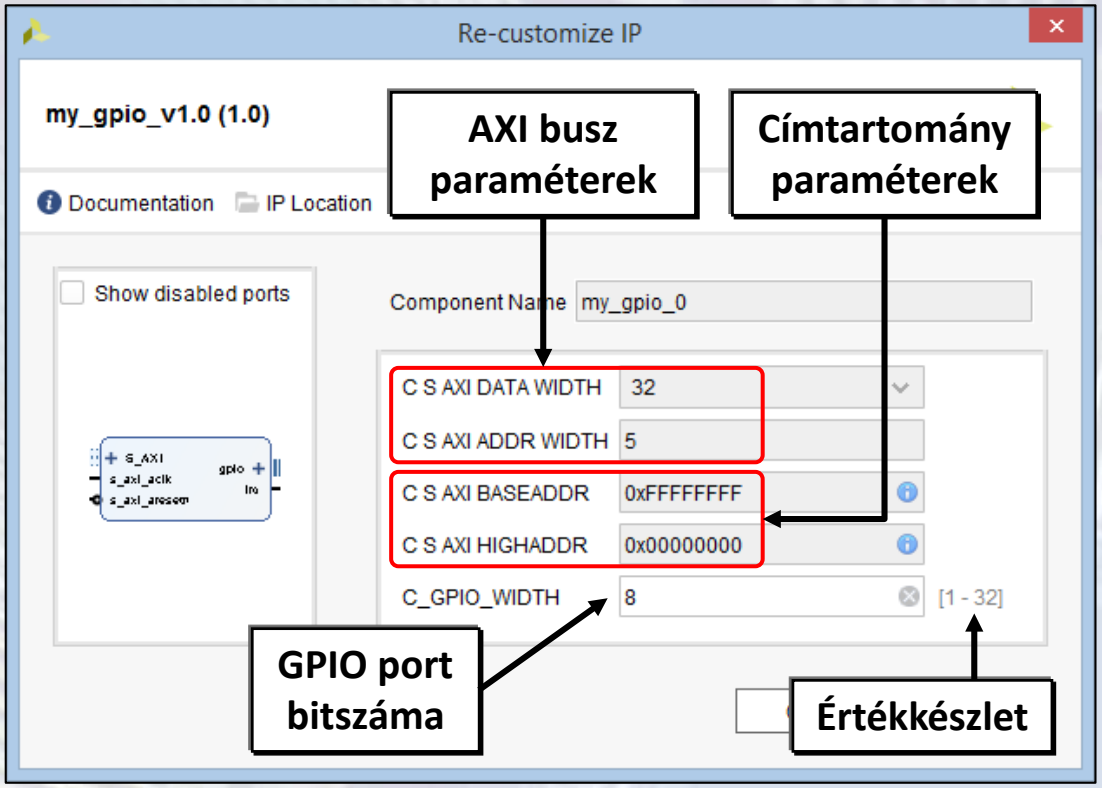


A GPIO interfész és a hozzá tartozó gpio_O, gpio_T és gpio_I portok

AXI4-Lite interfész

Megszakításkérő kimenet

AXI órajel és reset



AXI busz paraméterek

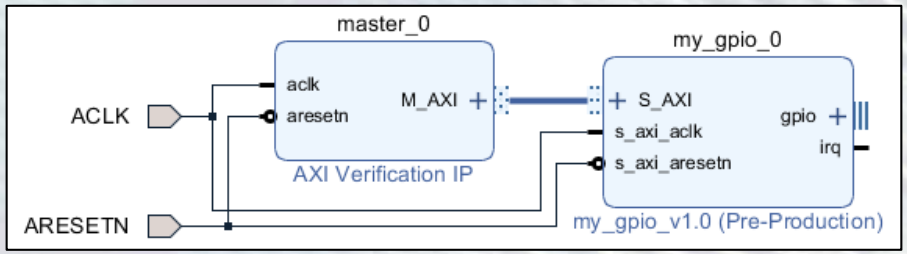
Címtartomány paraméterek

GPIO port bitszáma

Értékkészlet

IP tesztelése – Szimuláció (BFM)

- **Bus Functional Model (BFM): nem szintetizálható, csak a szimulátorral használható modellje a busznak**
  - Taszkokat biztosít a busz tranzakciók kezeléséhez
  - Ellenőrzi a busz protokoll betartását
- **A BFM szimulációs környezetet generáló TCL szkript: [IP_könyvtára]\example_designs\bfm_design\design.tcl**
  - Az AXI Verification IP-re épül (részletek: PG267)
  - SystemVerilog teszt környezet




```
149      result_slave = 1;
150     mtestWDataL[31:0] = 32'h00000001;
151     for(int i = 0; i < 4;i++) begin
152     S00_AXI_test_data[i] <= mtestWDataL[31:0];
153     mst_agent_0.AXI4LITE_WRITE_BURST(
154     mtestADDR,
155     mtestProtectionType,
156     mtestWDataL,
157     mtestBresp
158     );
159     mtestWDataL[31:0] = mtestWDataL[31:0] + 1;
160     mtestADDR = mtestADDR + 64'h4;
161     end
162        $display("Sequential write transfers exam
163        $display("Sequential read transfers examp
164        mtestID = 0;
165        mtestADDR = 64'h00000000;
```

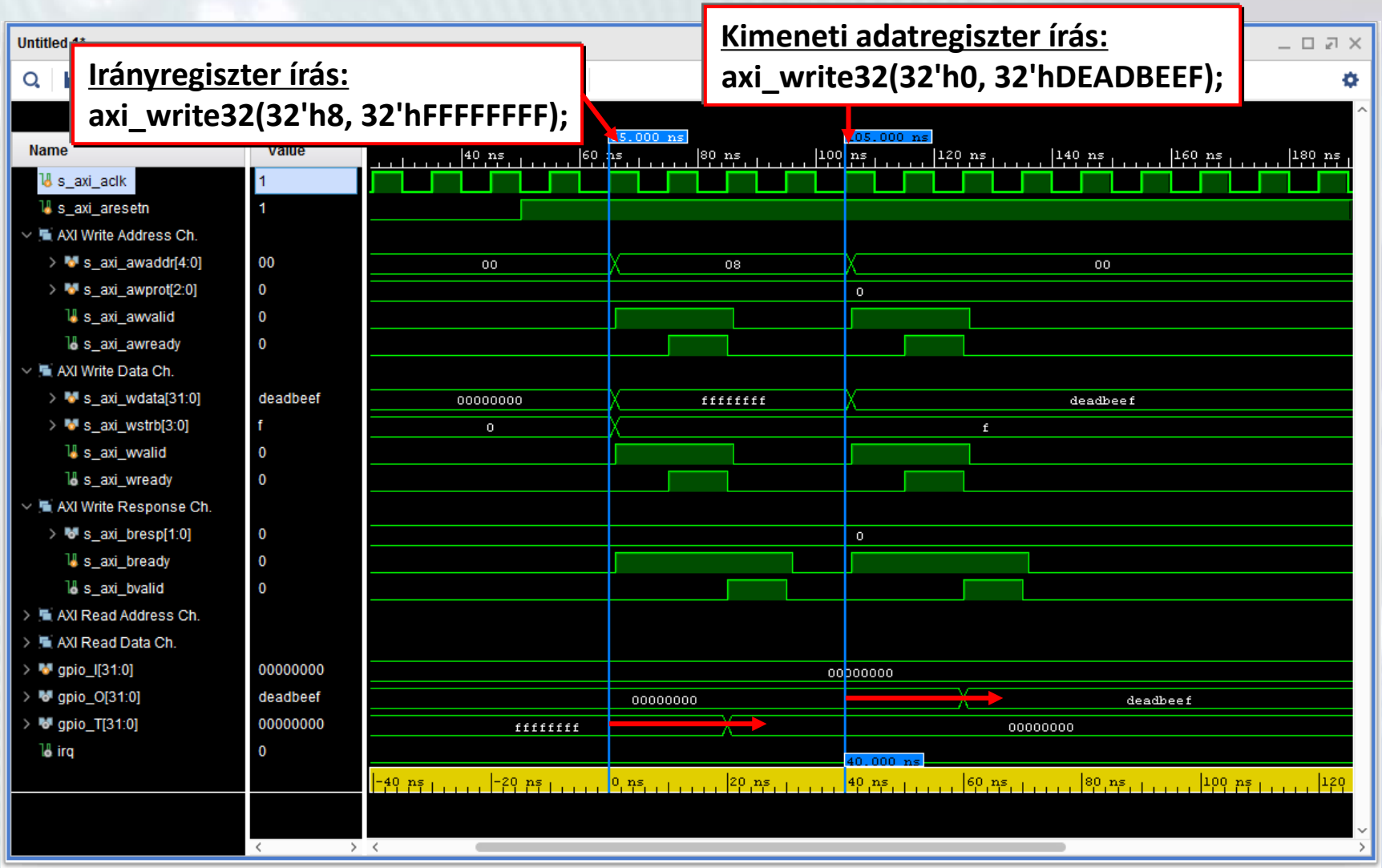
- **Saját, egyszerűbb BFM készítése**

IP tesztelése – Szimuláció (BFM)

Példa: egyszerű AXI4-Lite 32 bites írási taszk

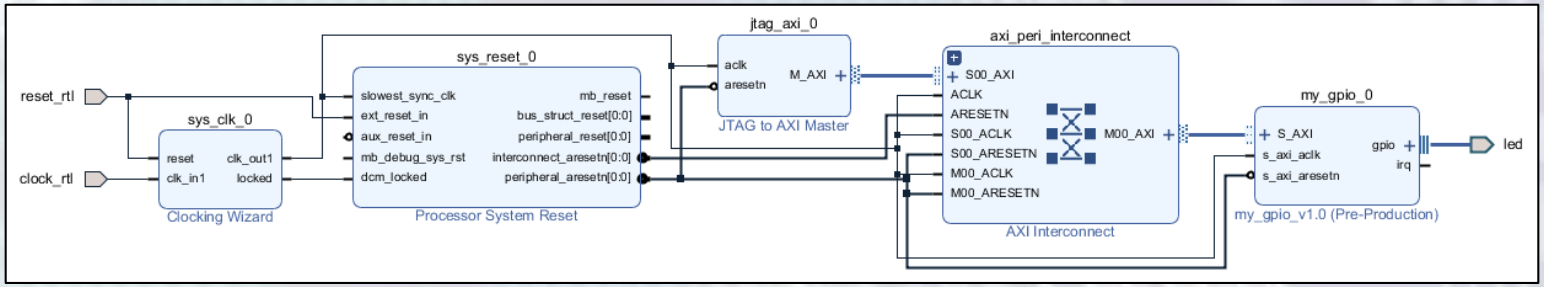
```
task axi_write32;
input [31:0] address;
input [31:0] data;
begin
  address = address & 32'hfffffffc;
  @(posedge s_axi_aclk);
  $display("%t: AXI write - address=0x%h,
     data=0x%h", $time, address, data);
  fork
    //Írási cím csatorna.
    begin
      #1 s_axi_awaddr  = address;
      s_axi_awprot     = 3'b000;
      s_axi_awvalid    = 1'b1;
      wait (s_axi_awready == 1'b1);
      @(posedge s_axi_aclk);
      #1 s_axi_awvalid = 1'b0;
    end
    //Írási adat csatorna.
    begin
      #1 s_axi_wdata  = data;
      s_axi_wstrb     = 4'b1111;
      s_axi_wvalid    = 1'b1;
      wait (s_axi_wready == 1'b1);
      @(posedge s_axi_aclk);
      #1 s_axi_wvalid = 1'b0;
    end
    //Írási válasz csatorna.
    begin
      #1 s_axi_bready = 1'b1;
      wait (s_axi_bvalid == 1'b1);
      $display("%t: AXI write –
         resp=0x%h", $time, s_axi_bresp);
      @(posedge s_axi_aclk);
      #1 s_axi_bready = 1'b0;
    end
  join
end
endtask
```

IP tesztelése – Szimuláció (BFM)



IP tesztelése – HW (JTAG)

- **A hardveres teszt rendszert generáló TCL szkript: [IP_könyvtára]\example_designs\debug_hw_design\design.tcl**
  - A JTAG to AXI Master IP-t használja (részletek: PG174)
  - FPGA konfigurációs fájl generálása szükséges



- **Csak TCL parancsokkal vezérelhető**
  - Példa: irányregiszter → 0xFF, kimeneti adatregiszter → 0xAA

```
create_hw_axi_txn dir_reg_wr [get_hw_axis hw_axi_1] -type WRITE –address
   44a00008 -len 1 -data 000000ff
create_hw_axi_txn dout_reg_wr [get_hw_axis hw_axi_1] -type WRITE -address
   44a00000 -len 1 -data 000000aa
run_hw_axi [get_hw_axi_txns dir_reg_wr]
run_hw_axi [get_hw_axi_txns dout_reg_wr]
```

IP készítése egyéb forrásból

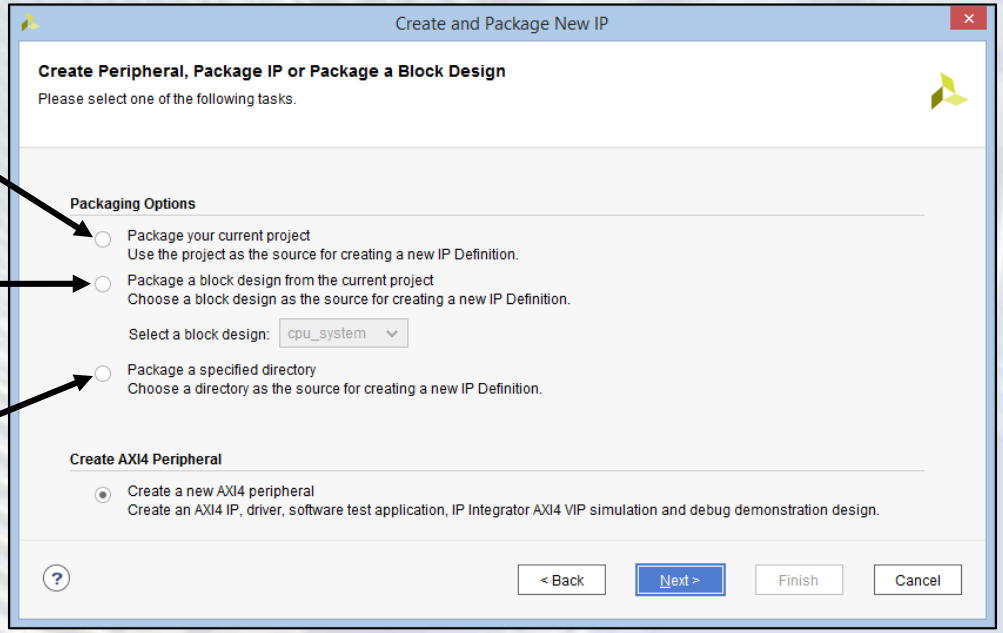
Az eddigiekhez hasonlóan IP készíthető még

- **A jelenleg megnyitott projekt forrásainak felhasználásával**
- **A projektben lévő Block Design-okból**
- **Adott könyvtár tartalmának felhasználásával**
  - *Package as a library core* opció: az IP a katalógusban nem lesz látható

Jelenleg megnyitott projekt becsomagolása

Block Design becsomagolása

Könyvtár tartalmának becsomagolása



Veilog modul felhasználása IP-ként

- **Ha teljesen egyedi perifériát szeretnénk készíteni és nem akarjuk használni a generált forráskód vázat**
- **Verilog modul beilleszthető a Block Design-ba:**
  - Jobb klikk → Add Module…
- **Modul port társítása interfészhez:**
  - ***X_INTERFACE_INFO*** Verilog direktívával
- **Modul portjához tartozó paraméterek megadása:**
  - ***X_INTERFACE_PARAMETER*** Verilog direktívával
- **Példák találhatóak a Vivado fejlesztői környezet kód sablonjai (Language Templates) között**
  - Verilog → IP Integrator HDL

Veilog modul felhasználása IP-ként