



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

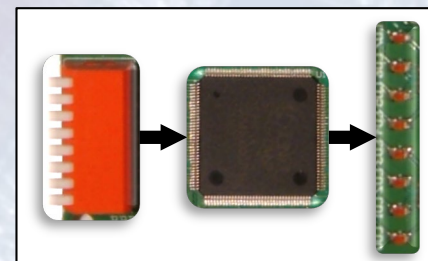
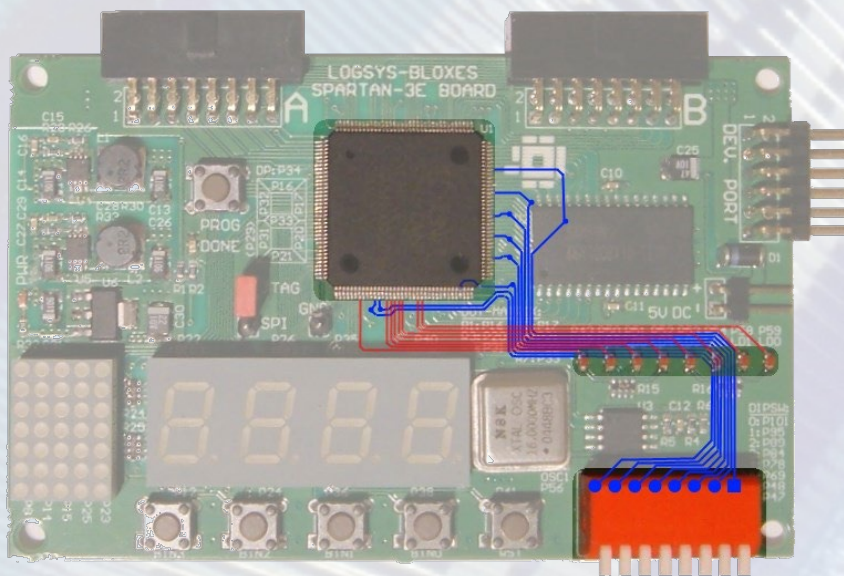
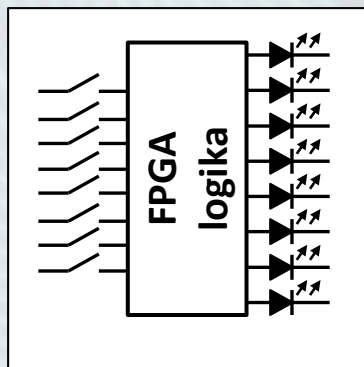
# **Digitális technika (VIMIAA03)**

## **2. gyakorlat és laboratórium**

**Raikovich Tamás**  
**BME MIT**

# 1. feladat: HW „Hello World!”

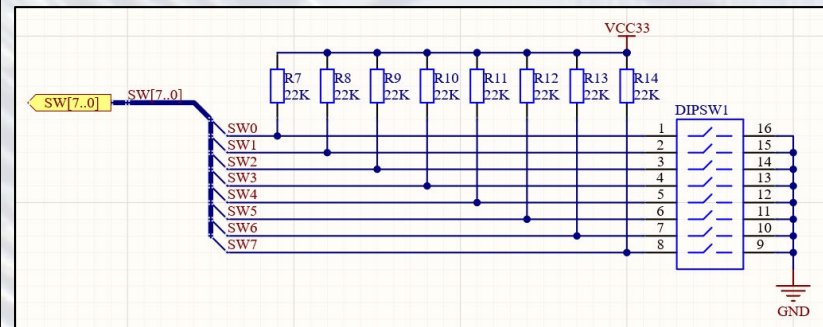
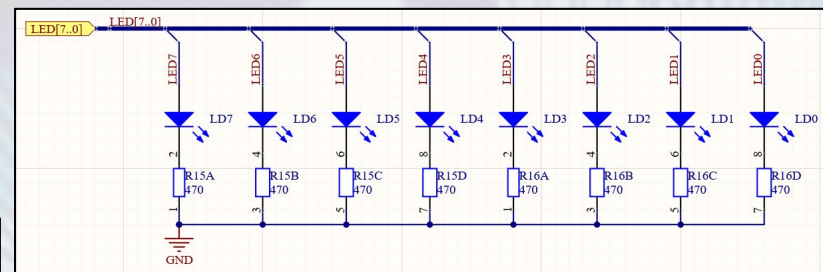
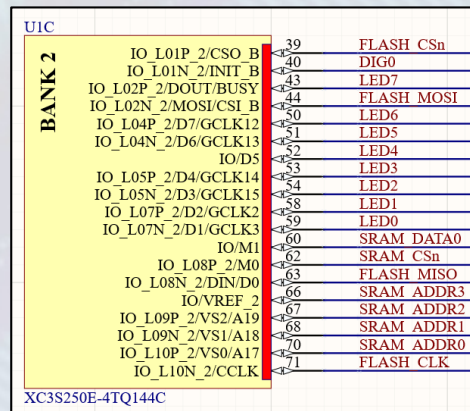
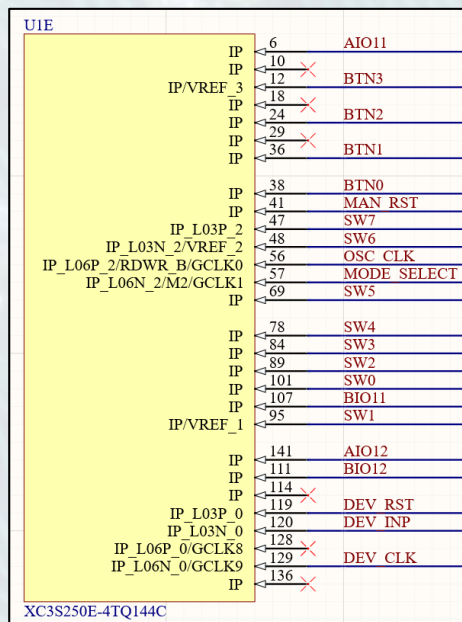
- 8 db LED vezérlése a 8 bites DIP kapcsolóval a LOGSYS Spartan-3E FPGA kártyán
  - DIP kapcsoló → FPGA bemenet: **kék** huzalozás
  - FPGA kimenet → LED: **piros** huzalozás
  - Mi legyen megvalósítva az FPGA logikában?
- Mint az előző alkalommal, de most Verilog HDL-ben





# 1. feladat: HW „Hello World!”

Az elvi kapcsolási rajz a szükséges paraméterekkel  
(nem tananyag, csak érdeklődőknek !)



LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA láb	P43	P50	P51	P52	P53	P54	P58	P59

Kapcsoló	7	6	5	4	3	2	1	0
FPGA láb	P47	P48	P69	P78	P84	P89	P95	P101

# 1.a feladat – A projekt létrehozása

- A Xilinx ISE Design Suite 14.6 fejlesztői környezetet használjuk az FPGA fejlesztéshez
- Új projekt létrehozása: **File** → **New Project...**

New Project Wizard

← Create New Project  
Specify project location and type.

Enter a name, locations, and comment for the project

Name: lab02

Location: E:\Egyetem\vimiaa03\labor\2\lab02

Working Directory: E:\Egyetem\vimiaa03\labor\2\lab02

Description:

Select the type of top-level source for the project

Top-level source type:  
HDL

More Info Next > Cancel

Projekt név:  
lab02

A projekt  
elérési útja

HDL, mert  
Verilog-ot  
használunk

# 1.a feladat – A projekt létrehozása

A használt eszköz beállítása: ***XC3S250E-4TQG144C***

New Project Wizard

← Project Settings  
Specify device and project properties.

Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S250E
Package	TQ144
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

More Info < Back Next > Cancel

Family: Spartan3E

Device: XC3S250E

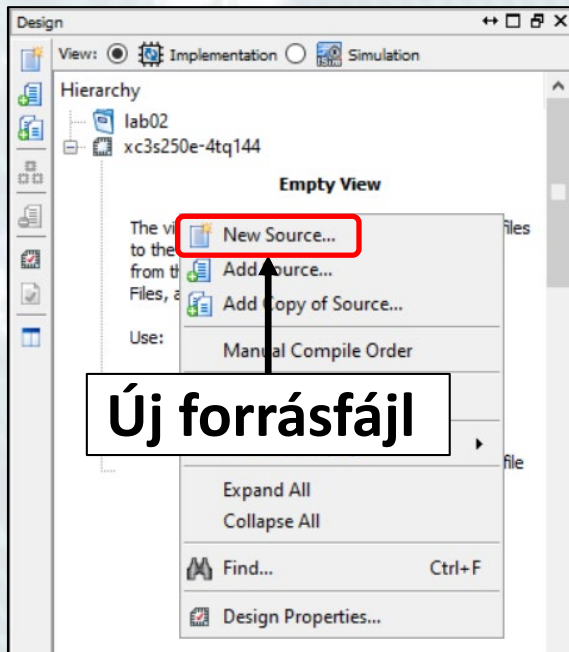
Package: TQ144

Speed: -4

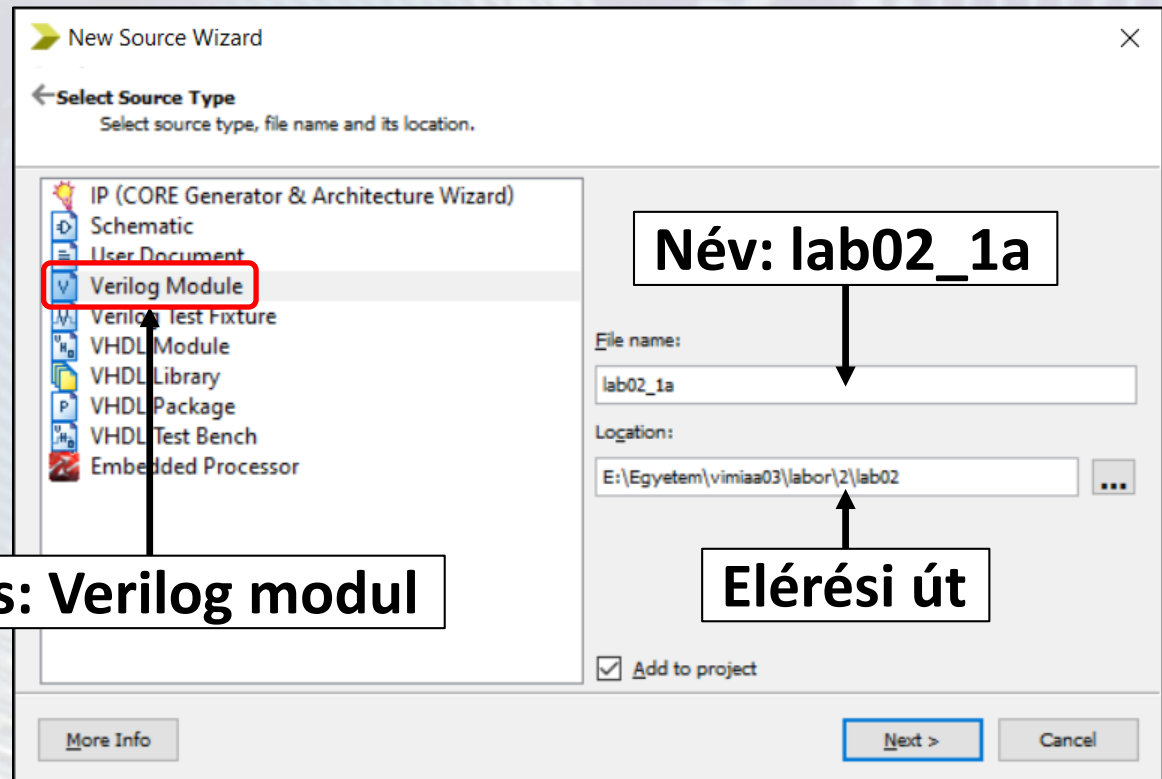


# 1.a feladat – Verilog modul hozzáadása

Adjunk az üres projekthez egy Verilog modult: **lab02\_1a**



Új forrásfájl



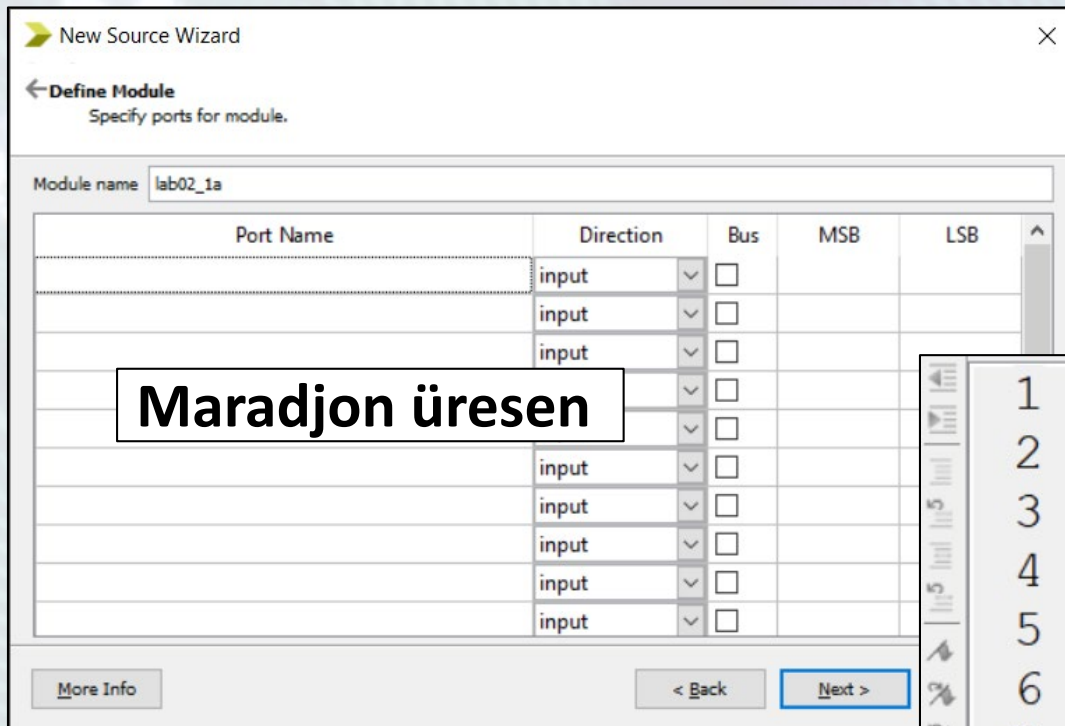
Típus: Verilog modul

Név: lab02\_1a

Elérési út

# 1.a feladat – Verilog modul hozzáadása

- A modul portjait majd begépeljük
- Létrejön a Verilog modul váza, amit ki kell egészíteni



The dialog box is titled "New Source Wizard" and "Define Module". It prompts the user to "Specify ports for module." The "Module name" field contains "lab02\_1a". Below this is a table for defining ports.

Port Name	Direction	Bus	MSB	LSB
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		

Buttons at the bottom: "More Info", "< Back", and "Next >".

Maradjon üresen

A Verilog  
modul váza

```
1 `timescale 1ns / 1ps
2
3 module lab02_1a (
4 );
5
6 endmodule
7
```

# Verilog HDL ismeretek

## Verilog modul felépítése, részei

- A Verilog nyelv hierarchikus, egymásba ágyazott funkcionális egység (modul) alapú tervezési megközelítést használ
  - A legfelső (top-level) modul kapcsolódik az eszköz lábaihoz
- A modul deklarálásának gyakorlatban használt szintaxisa

```
      A modul neve
      └──────────┘
module SomeFunction(
    input  wire [7:0] op1,
    input  wire [7:0] op2,
    output wire [7:0] result
);

    assign result = op1 & op2;

endmodule
```

A portok deklarálása a modul fejlécében, a port listában. A „külvilággal” való kapcsolat.

A funkcionális leírása a modul „törzsében”



# Verilog HDL ismeretek

## Modul portjainak megadása

- A portok deklarálásának szintaxisa

**input**    **wire**    [7:0]    **op1**  
*irány*    *típus*    *méret*    *név*

- **Irány**
  - Bemeneti port: **input**
  - Kimeneti port: **output**
- **Típus**
  - **wire**: vezeték (alapértelmezett, ha nem adjuk meg)
  - **reg**: tároló is lehet belőle (csak kimeneti portra adható meg)
- **Méret: [j : i]** → a port mérete  $|j - i| + 1$  bit
  - **i** a legkisebb helyiértékű bit (LSb) sorszáma
  - **j** a legnagyobb helyiértékű bit (MSb) sorszáma
  - Jellemzően N-1:0 (pl. 7:0), de lehet más is
  - Ha nem adjuk meg, akkor a port mérete 1 bit lesz

# 1.a feladat – A portok megadása

- A modul fejlécében adjuk meg a portokat
- Most 1 bites portokat használjunk
  - 8 darab bemenet: ***sw0, sw1, ..., sw7***
  - 8 darab kimenet: ***ld0, ld1, ..., ld7***

```
module lab02_1a(  
    input  wire sw0,  
    input  wire sw1,  
    :  
    :  
    output wire ld0,  
    :  
    :  
) ;
```

# Verilog HDL ismeretek

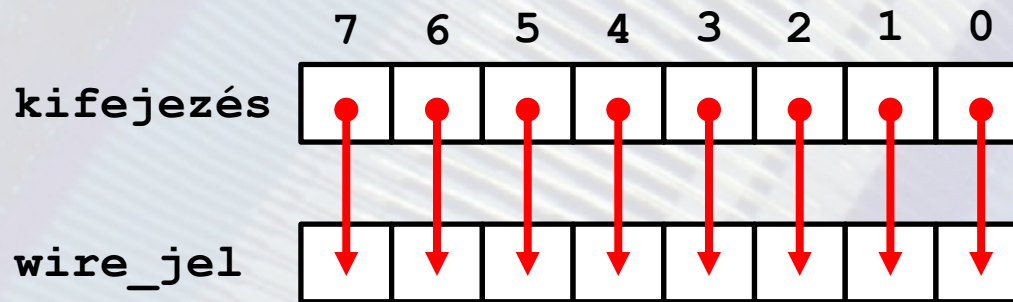
## Folytonos értékadás

Logikai kapcsolat megadása *wire* típusú jelek esetén

```
assign result    = op1 & op2;
```

```
assign wire_jel = kifejezés;
```

- A *wire\_jel* által reprezentált értéket a *kifejezés* minden pillanatban meghatározza (folytonos értékadás), ez a *kombinációs logika* leírásának felel meg
- Az értékadás az LSb-től kezdve bitenként történik





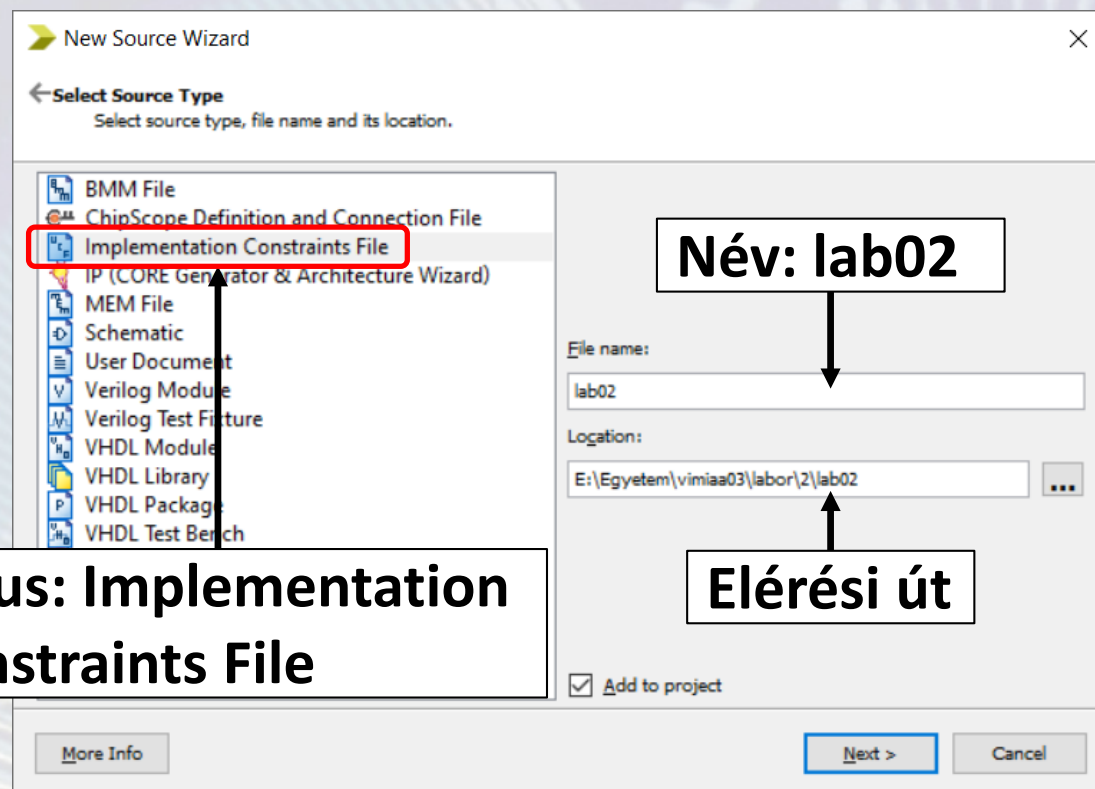
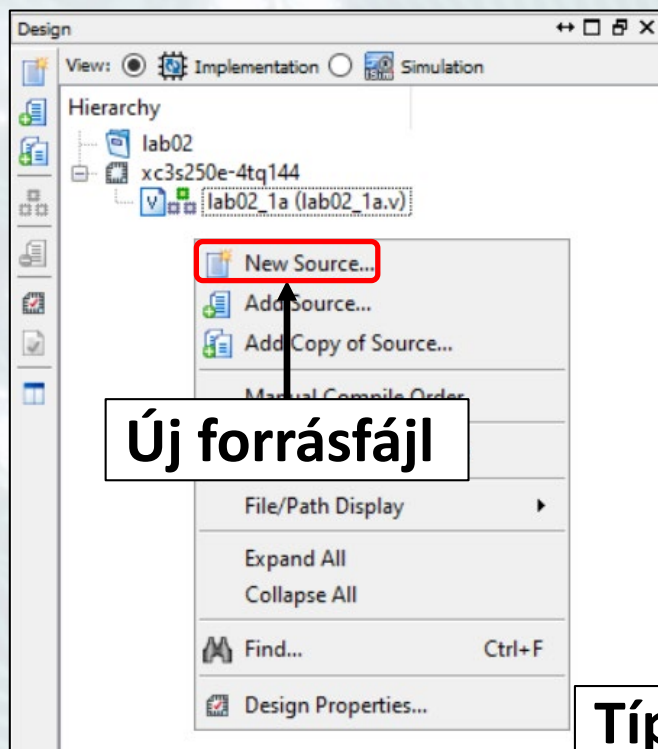
# 1.a feladat – A modul funkciójának leírása

- A modul törzsében írjuk le a megvalósítandó funkciót
- Az adott **ld** port kapja meg a megfelelő **sw** port értékét
  - Ehhez 8 darab **assign** utasítás szükséges

```
module lab02_1a(  
    ⋮        ⋮        ⋮  
);  
  
assign ld0 = sw0;  
assign ld1 = sw1;  
    ⋮        ⋮        ⋮  
endmodule
```

# 1.a feladat – A portok és az FPGA lábak összerendelése

Adjunk a projekthez egy UCF fájlt: **lab02**



# 1.a feladat – A portok és az FPGA lábak összerendelése

Az UCF fájlban adjunk meg minden porthoz egy FPGA lábat

- **NET "port bit" LOC="FPGA láb";**
- A kapcsoló bitek az UCF fájlban: **sw0 ... sw7**
- A LED bitek az UCF fájlban: **ld0 ... ld7**

**NET "sw0" LOC="P101";**

⋮ ⋮ ⋮ ⋮

**NET "ld0" LOC="P59";**

⋮ ⋮ ⋮ ⋮

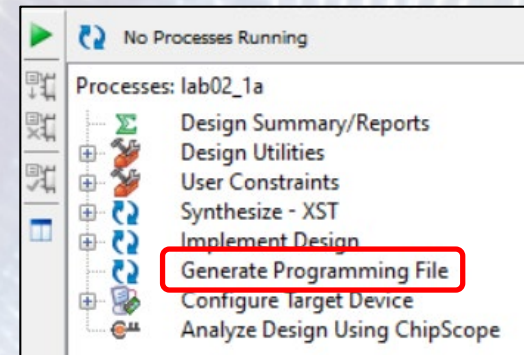
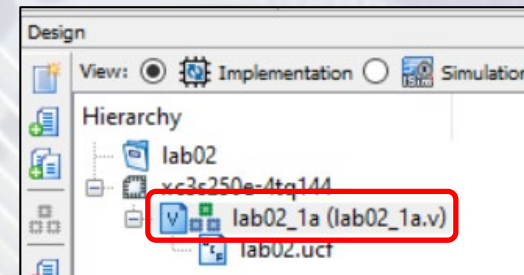
LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA láb	P43	P50	P51	P52	P53	P54	P58	P59

Kapcsoló	7	6	5	4	3	2	1	0
FPGA láb	P47	P48	P69	P78	P84	P89	P95	P101



# 1.a feladat – Kipróbálás a hardveren

- A fejlesztői környezetben generáljuk az FPGA konfigurációs fájlt (BIT fájl)
- A Logsys GUI-val programozzuk fel az FPGA-t
  1. Az 5 V-os tápfeszültség bekapcsolása
  2. A JTAG funkció megnyitása
  3. Az eszközök felderítése a kártyán
  4. A BIT fájl letöltése az eszközre
- Próbáljuk ki a működést a hardveren



# 1.a feladat – Kipróbálás a hardveren

The screenshot shows the Logsys Control Panel software interface. The window title is "Logsys Control Panel". The menu bar includes "File", "View", "Window", and "Help".

The left sidebar contains several sections:

- Info:** LOGSYS development cable
- Control:** ☐ RST, ☐ CLK, 10 Hz, Set button
- Power:** Voltage section with a red "+5V On" button (indicated by red arrow 1); Current section with "Maximum Value: 450 mA" and "Log to file..." checkbox
- Measurement:** +5Vout: 4,94 V, Maximum Value: 500 mA, I/Oref: 3,31 V, Critical Value: 90 %, JTAGref: 2,51 V, Samples/Second: 10
- Configuration:** ☒ JTAG Download (indicated by red arrow 2)
- Communication:** ☐ BitBang I/O, ☐ UART, ☐ USRT

The main area is titled "Download ( DC023)". It contains a "JTAG" section with a "Query JTAG chain" button (indicated by red arrow 3) and a "Clear Log" button. Below this, it says "Found 1 device(s) in the JTAG chain." and lists "XC3S250E (Xilinx)" in a dropdown menu (indicated by red arrow 4). A "Configure the selected device..." button is also present.

At the bottom left, there is a graphical representation of a gauge showing a value of 045,00, with a scale from 0 to 500,00.

# 1.b feladat – Bitvektorok használata

A leírás nagymértékben egyszerűsíthető, ha egybites portok helyett több bites portokat (azaz bitvektorokat) használunk. Az előzőekhez hasonlóan végezzük el az alábbiakat:

- Adjunk a projekthez egy új Verilog modult: **lab02\_1b**
  - Legyen ez a top-level modul: **jobb klikk** → **Set as Top Module**
  - Az UCF fájl az új modul alá kerül (ha nem: eltávolítás, majd hozzáadás)
- A modul portjai legyenek most 8 bitesek

```
input  wire [7:0] sw,
output wire [7:0] ld
```
- A funkció leírásához most egyetlen **assign** elegendő

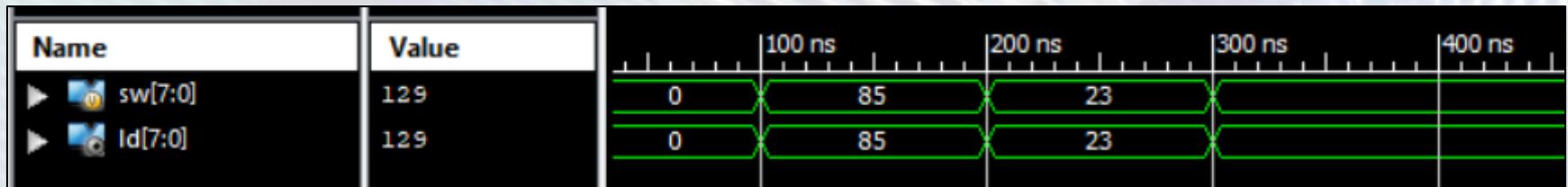
```
assign ld = sw;
```
- Az UCF fájlban a sorszámukkal hivatkozhatunk a port bitekre
  - **sw<0>**, ..., **sw<7>** és **ld<0>**, ..., **ld<7>** (pl. NET "ld<0>" LOC="P59")



# 1.b feladat – Tesztelés, szimuláció

A szimulátor segítségével ellenőrizhető az elkészült rendszer egészének vagy egy részének megfelelő működése a hardver nélkül is

- Bemeneti adatok: a tesztelendő modul bemenetei
  - A tesztkörnyezetben adjuk meg, hogy egymás után milyen értékek kerüljenek a bemenetekre
- Eredmény: idődiagram formájában
  - A tesztkörnyezetben (és kiegészítéssel az almodulokban) lévő belső jelek időbeli változását mutatja grafikusan



# 1.b feladat – Tesztelés, szimuláció

- **A szimuláció nagyon fontos**, de mi idő hiányában többször is el fogunk tekinteni ettől
- Most szimulálni fogjuk a **lab02\_1b** modult
- Váltunk át implementációs nézetről szimulációsra
- Tesztkörnyezet hozzáadása: **lab02\_1b\_test**

The screenshot shows the Xilinx ISE IDE interface. On the left, the 'Design' pane shows the project hierarchy with 'lab02' expanded, showing 'lab02\_1a (lab02\_1a.v)' and 'lab02\_1b (lab02\_1b.v)'. The 'Simulation' view is selected. A red box highlights the 'New Source...' button. A callout box points to it with the text 'Új forrásfájl'. In the center, the 'New Source Wizard' dialog is open. The 'Select Source Type' tab is active, and 'Verilog Test Fixture' is selected. A callout box points to it with the text 'Típus: Verilog Test Fixture'. The 'File name' field contains 'lab02\_1b\_test', and the 'Location' field contains 'E:\Egyetem\vimia03\labor\2\lab02'. A callout box points to the 'File name' field with the text 'Név: lab02\_1b\_test'. Another callout box points to the 'Location' field with the text 'Elérési út'. On the right, the 'Associate Source' tab is active, showing a list of sources with 'lab02\_1b' selected. A callout box points to it with the text 'Tesztelendő modul: lab02\_1b'.

Új forrásfájl

Név: lab02\_1b\_test

Típus: Verilog Test Fixture

Elérési út

Tesztelendő modul: lab02\_1b

# 1.b feladat – Tesztelés, szimuláció

Várakozási  
időegység

A tesztelt modul  
bemenetei *reg*  
típusú változók,  
mert a következő  
értékadásig  
tartaniuk kell az  
értéküket

Itt fogjuk megadni  
a tesztelt modul  
bemeneteire  
kerülő értékeket

```
1  `timescale 1ns / 1ps
2
3  module lab02_lb_test;
4
5      // Inputs
6      reg [7:0] sw;
7
8      // Outputs
9      wire [7:0] ld;
10
11     // Instantiate the Unit Under Test (UUT)
12     lab02_lb uut (
13         .sw(sw),
14         .ld(ld)
15     );
16
17     initial begin
18         // Initialize Inputs
19         sw = 0;
20
21         // Wait 100 ns for global reset to happen
22         #100;
23
24         // Add stimulus here
25
26     end
27
28 endmodule
```

Szimuláció  
felbontása

A tesztkörnyezet egy portok nélküli modul

A tesztelt modul kimenetei

A tesztelt modul példányosítása  
(részletek a következő laboron)

Az *initial* blokk tartalma  
a szimuláció kezdetekor  
egyszer hajtódik végre

Várakozás (itt most 100 ns ideig):  
#[idő az adott egységben értelmezve]



# Verilog HDL ismeretek

## Numerikus konstansok megadása

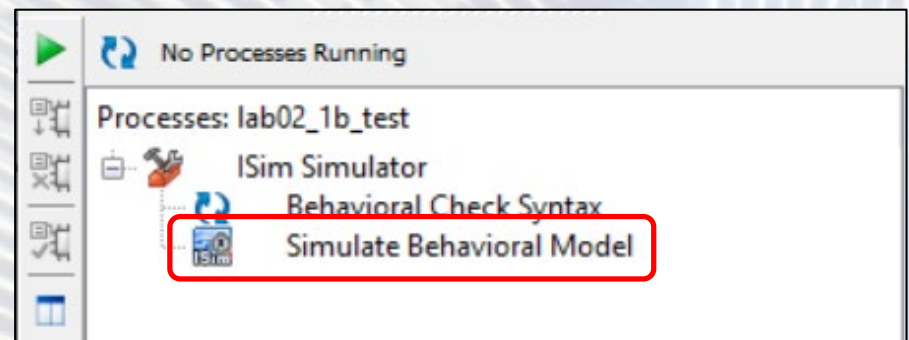
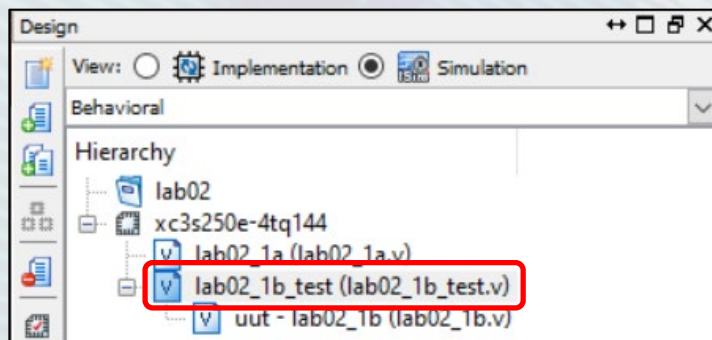
- A numerikus konstansok megadásának szintaxisa **<bitek száma>'<számrendszer><numerikus konstans>**
- **Bitek száma:** a konstans mérete bitekben
  - Az alapértelmezett méret 32 bit, ha nem adjuk meg
- **Számrendszer:** **decimális az alapértelmezett, ha nincs megadva**
  - Bináris: **b**, oktális: **o**, decimális: **d**, hexadecimális: **h**
- **A numerikus konstans a számrendszer digitjeivel adható meg**
  - A **'\_'** karakter használható a számjegyek szeparálásához
- **Példák**
  - **8'b0000\_0100**: 8 bites bináris konstans, értéke 4
  - **6'h1f**: 6 bites hexadecimális konstans, értéke 31
    - Binárisan: 6'b01\_1111
  - **128**: 32 bites decimális konstans
    - Binárisan: 32'b00000000\_00000000\_00000000\_10000000

# 1.b feladat – Tesztelés, szimuláció

- Egészítsük ki a tesztkörnyezetet néhány bemeneti adattal (tesztvektorok) az **initial** blokkban, az értékadások között várjunk 100 ns ideig

```
sw = 8'h55;           //Decimális értéke 85  
#100;  
sw = 8'd23;  
#100;  
sw = 8'b1000_0001;    //Decimális értéke 129
```

- Indítsuk el a szimulációt a tesztkörnyezetre



# 1.b feladat – Tesztelés, szimuláció

The screenshot shows the ISim (P.68d) - [Default.wcfg\*] window. The interface includes a menu bar (File, Edit, Help), a toolbar, a project tree on the left, a signal list in the center, and a waveform viewer on the right. The waveform viewer displays a digital signal with values 0, 85, 23, and 129 over time. The time scale is set to 1.00us. The console at the bottom shows the simulation status and time resolution.

**Idődiagram**

**Az idődiagram nagyítása és kicsinyítése**

**Számrendszer módosítása**

**A szimuláció futtatása adott ideig**

**A szimuláció újraindítása**

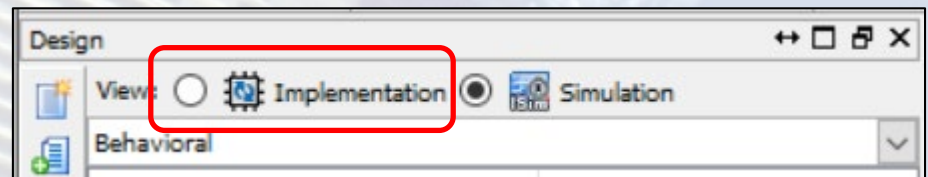
Changes the radix of the selected signal(s)

Sim Time: 1,000,000 ps



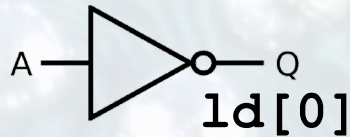
# 1.b feladat – Tesztelés, szimuláció

- A jelek sorrendjének módosítása az idődiagramon
  - A bal egérgombot lenyomva tartva húzzuk a jelet a kívánt pozícióba
- Helyes a működés, ha az **ld** kimeneten ugyanazt az értéket látjuk, mint az **sw** bemeneten
- Az implementációs nézetre váltva az előzőekhez hasonlóan generálható az FPGA konfigurációs fájl
- Próbáljuk ki ezt a változatot is a hardveren



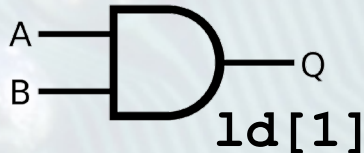
## 2. feladat – Logikai műveletek

### NOT



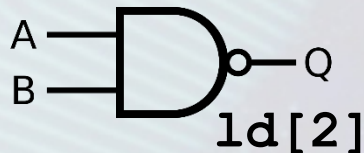
A	Q
0	1
1	0

### AND



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

### NAND



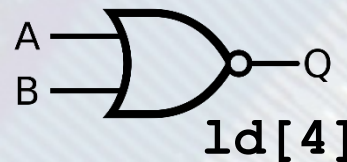
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

### OR



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

### NOR



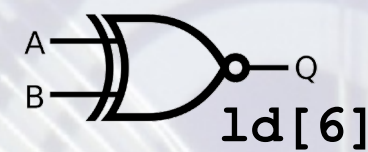
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

### XOR



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

### XNOR



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

**Feladat az adott logikai kapuk megvalósítása**

- Bemenetek: BTN0 (A) és BTN1 (B) gombok
- Kimenetek: LED-ek

# Verilog HDL ismeretek

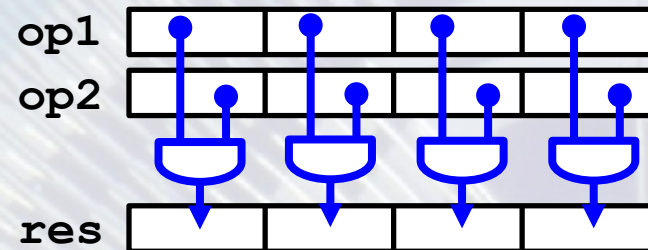
## Bitenkénti és bitredukciós operátorok

### Logikai műveletek leírása a bitenkénti és bitredukciós operátorokkal

Művelet	Bitenkénti operátor	Bitredukciós operátor
NOT	~	nincs ilyen
AND	&	&
NAND	nincs ilyen	~&
OR		
NOR	nincs ilyen	~
XOR	^	^
XNOR	nincs ilyen	~^

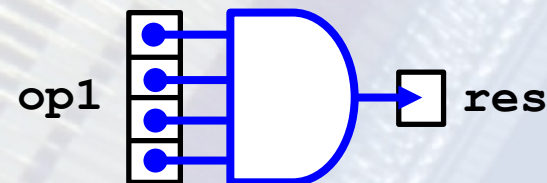
Bitenkénti operátorok: vektorok esetén bitenként hajtódik végre

```
assign res = op1 & op2;
```



Bitredukciós operátorok: egyetlen vektor bitjein hajtanak végre műveletet, az eredmény 1 bites

```
assign res = &op1;
```





# Verilog HDL ismeretek

## Indexelő operátor

Több bitből álló jel (vektor) egy részének kiválasztása az indexelő operátorral lehetséges

**vektor\_jel[i], vektor\_jel[j:i]**

- **[i]** kiválasztja a vektor *i*-edik bitjét
- **[j:i]** kiválasztja a vektor *j*-edik és *i*-edik bitje közötti részét (a határokat is beleértve)

## 2. feladat – Logikai műveletek

- Adjunk a projekthez egy új Verilog modult: **lab02\_2**
  - Legyen ez a top-level modul: **jobb klikk** → **Set as Top Module**
  - Az UCF fájl az új modul alá kerül (ha nem: eltávolítás, majd hozzáadás)
- Adjuk meg a modul portjait
  - **bt**: 2 bites **wire** típusú bemenet
  - **ld**: 7 bites **wire** típusú kimenet
- Adjuk meg a bemenetek és kimenetek közötti kapcsolatokat
  - A bitenkénti operátorokkal: **assign ld[1] = bt[0] & bt[1];**
  - Vagy a bitredukciós operátorokkal: **assign ld[1] = &bt;**
- Módosítsuk az UCF fájlt
  - A megjegyzések (kommentek) kezdetét a # karakter jelzi
  - Az **sw** portbitek nem kellene, kommentezzük ki
  - Az **ld<7>** portbit nem kell, kommentezzük ki
  - Adjuk hozzá a **bt** portbiteket (bt<0>: P38, bt<1>: P36)

## 2. feladat – Logikai műveletek

- Generáljuk az FPGA konfigurációs fájlt (BIT fájl)
- A Logsys GUI-val programozzuk fel az FPGA-t
- Próbáljuk ki a működést a hardveren
- Az első két labor alapján a kapcsolási rajz vagy a Verilog HDL használata volt könnyebb?