



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

Digitális technika

Xilinx ISE GUI használata

BME MIT
Fehér Béla
Raikovich Tamás

Xilinx ISE használata



A fejlesztőkörnyezet különböző módokon használható

1. Lokálisan a számítógépre telepítve: IE226 és IE321

Az otthoni használathoz szükséges a telepítőkészlet letöltése és ~12GB lemezterület

2. Virtuális gépen futtatva IE413

3. A BME VIK kari felhőben <https://cloud.bme.hu>

Telepítés és lemezigény nélküli, de hálózati kapcsolatot igényel (VIK Cloud, KIFÜ-NIIF Cloud)

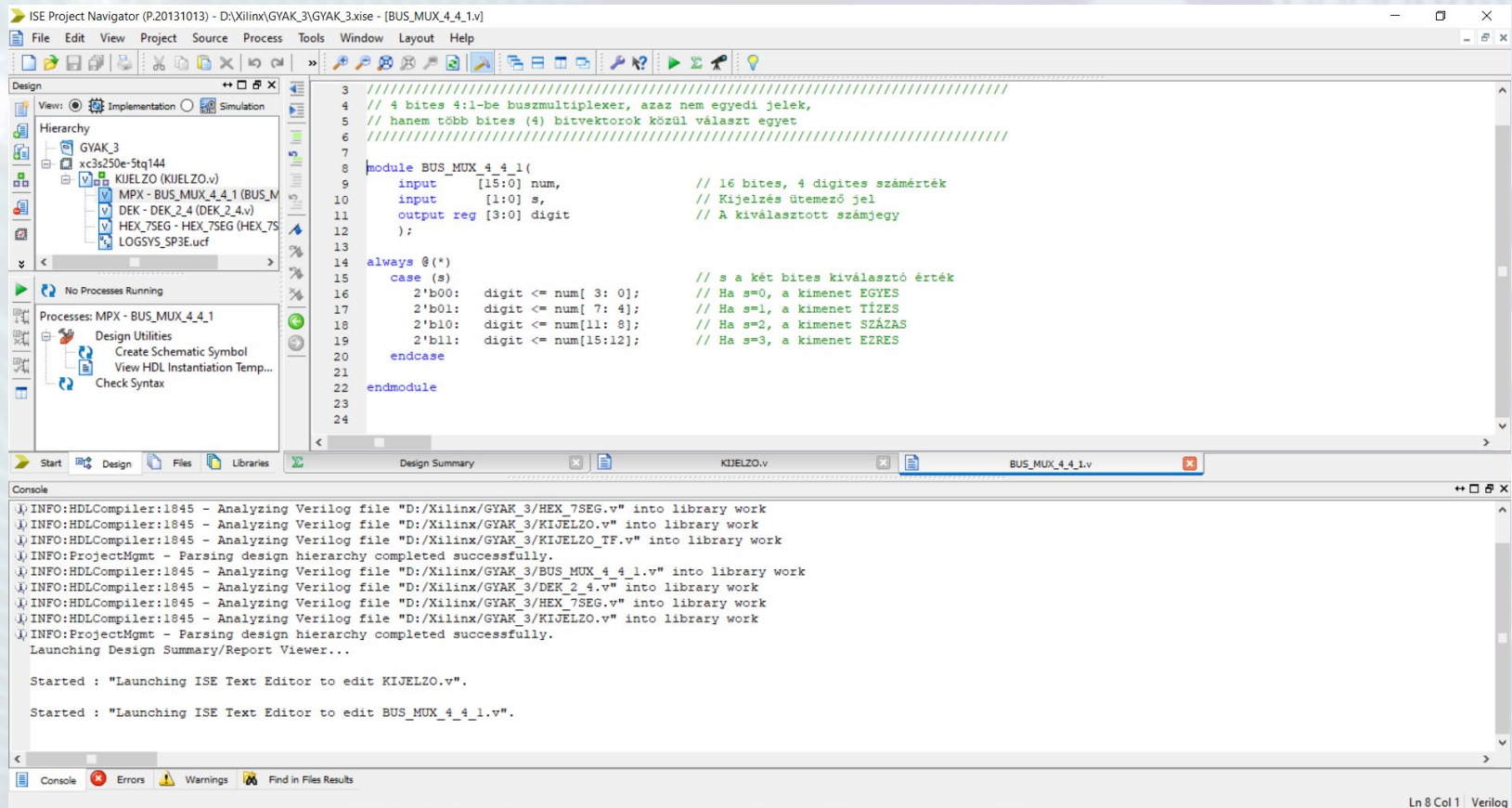
Bejelentkezés címtáron keresztül: Windows 10 ISE V2 VM

Használat távoli asztal kapcsolaton keresztül vagy konzolban

4. KSZK SCHAcc távoli hozzáférés RemoteApp

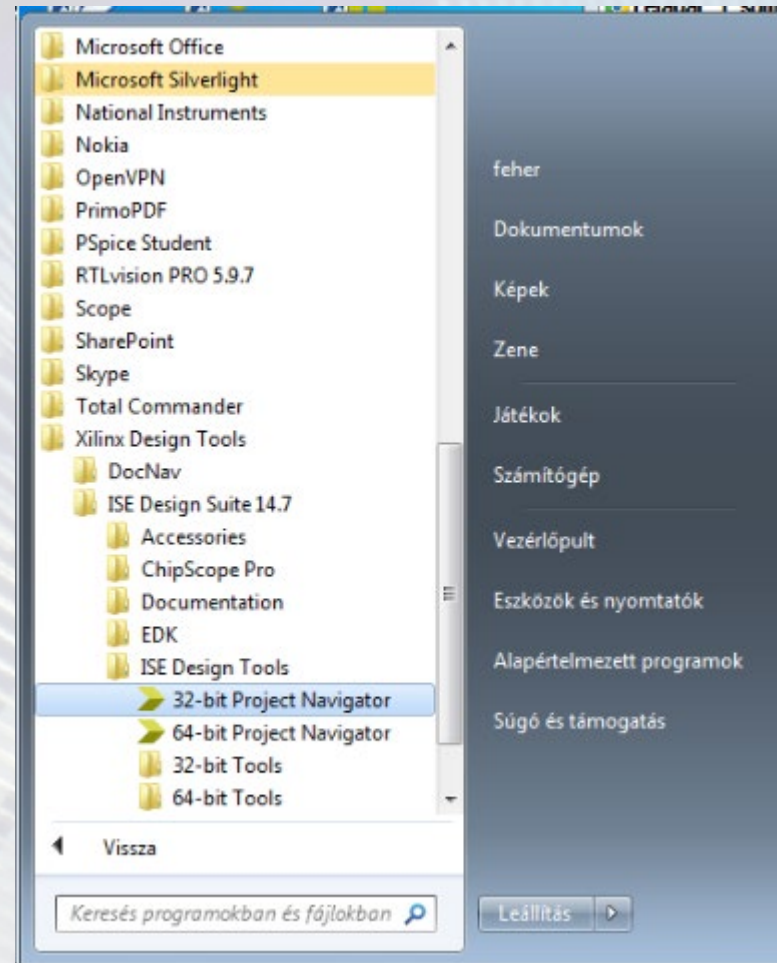
Xilinx ISE

- **Xilinx ISE: Összetett tervezői környezet, mi csak néhány egyszerűbb szolgáltatását használjuk**



Projekt létrehozása (1)

- **Xilinx ISE 14.7/6 elindítása**
 - Asztal → Ikon
 - vagy
 - Start → All Programs →
 - Xilinx Design Tools →
 - ISE Design Suite 14.7 →
 - ISE Design Tools →
 - 32/64-bit Project Navigator



Projekt létrehozása (2)

- Az új projekt létrehozása
File → New Project
- Projekt neve: Lab1
- Helye pl. D:\DTLab\
• Projekt típusa: HDL

New Project Wizard

Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name: Lab1

Location: D:\DTLab\Lab1 ...

Working Directory: D:\DTLab\Lab1 ...

Description:

Select the type of top-level source for the project

Top-level source type: HDL

More Info Next Cancel

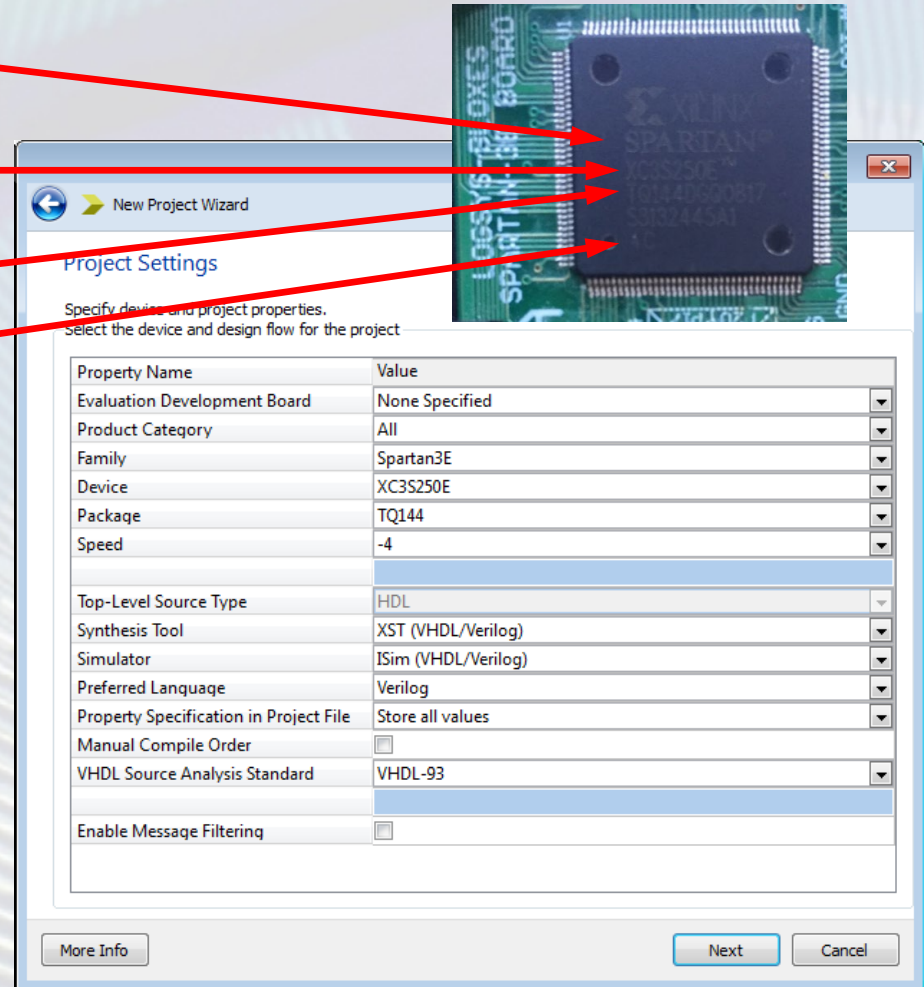
Projekt létrehozása (3)

- **Ami fontos: A fizikai alkatrész specifikálása**

- Family: Spartan3E
- Device: XC3S250E
- Package: TQ144
- Speed: -4

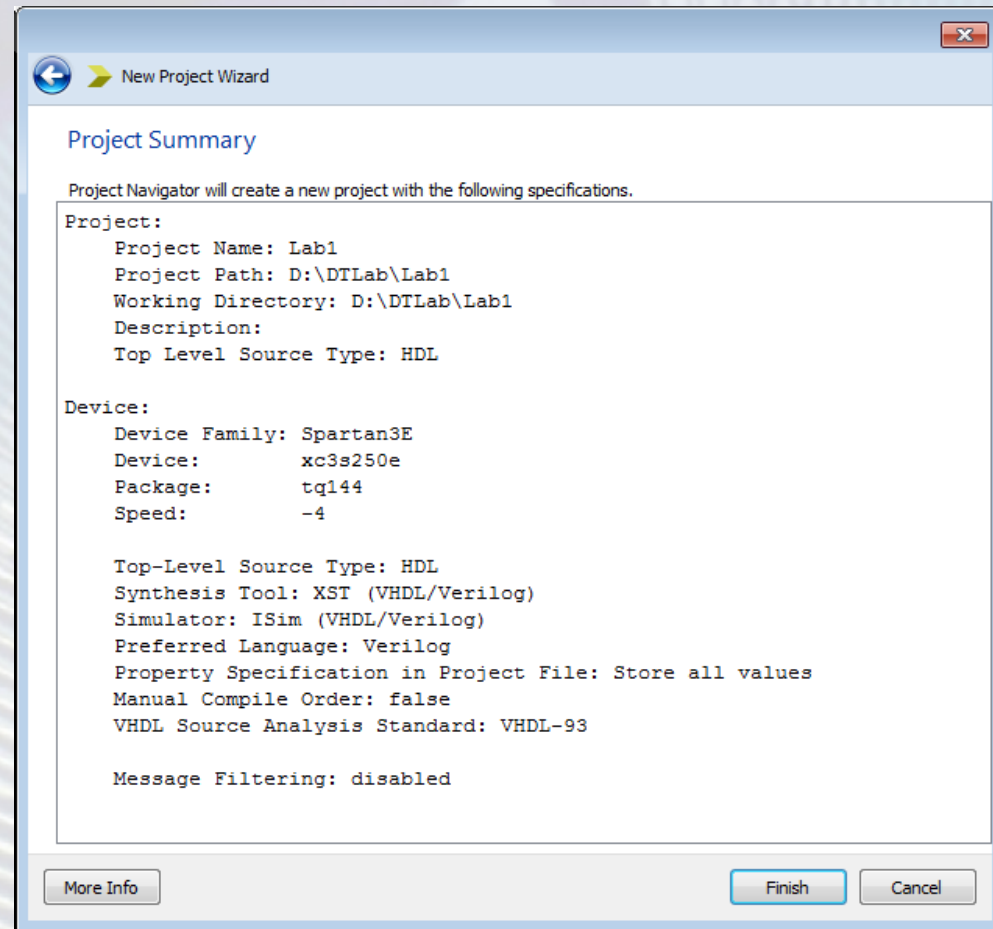
- **A feldolgozási technológia specifikálása**

- Synthesis tool: XST
- Pref. lang: Verilog



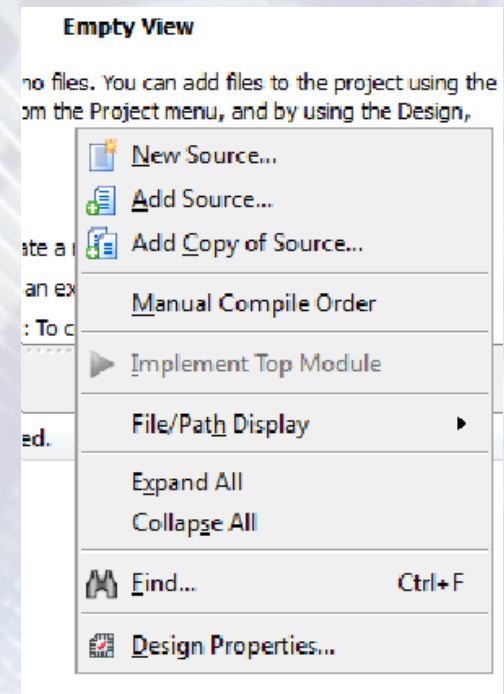
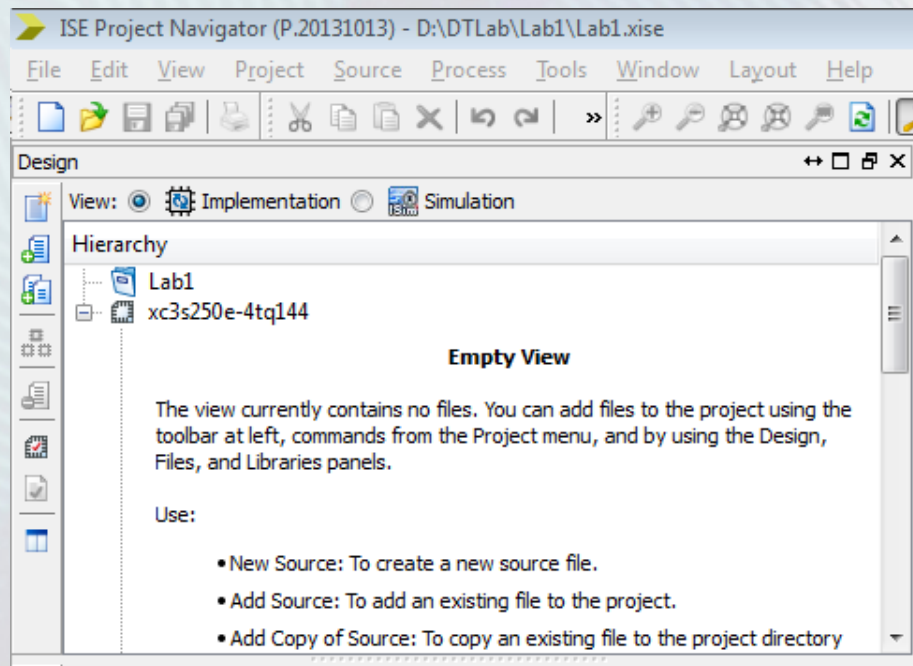
Projekt létrehozása (4)

- Összefoglalás a beállításokról
- A Finish után a projekt struktúra létrejön
- Ezután következik a projekt forrásfájlok előkészítése



Projekt források létrehozása (1)

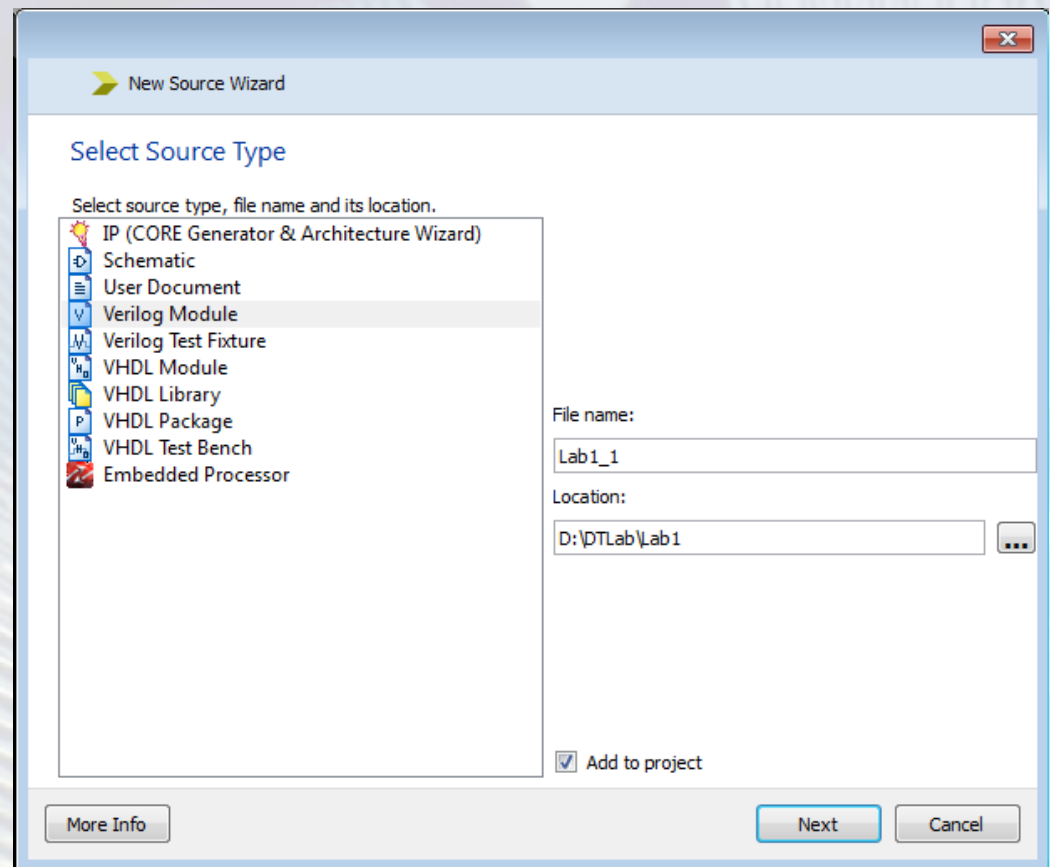
- A tervet a HDL forrásfájlok specifikálják
 - Project → New Source
 - vagy a Design ablakban jobb gomb → New Source



Projekt források létrehozása (2)

- **Az első forrásfájl típusa:**

- Verilog Module
- Fájl neve: Lab1_1
- (kiterjesztés .v)
- Helye: a projekt könyvtár
- ✓ Add to project



Projekt források létrehozása (3)

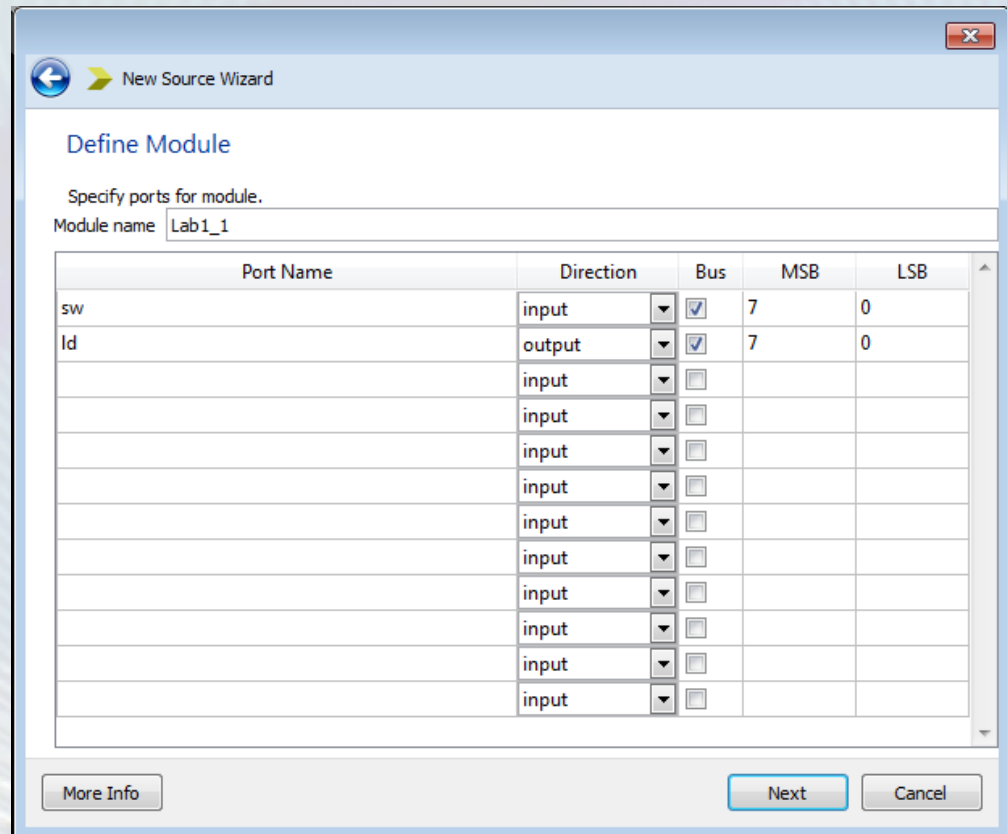
- **Definiáljuk a bemeneti és kimeneti jeleket**

- **Kapcsolók:**

- sw 8 db
 - Input
 - Bus \checkmark
 - MSB 7 LSB 0

- **LED-ek**

- ld 8 db
 - Output
 - Bus \checkmark
 - MSB 7 LSB 0



New Source Wizard

Define Module

Specify ports for module.


Module name: Lab1_1

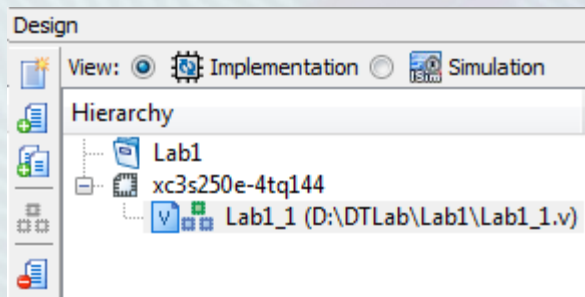
Port Name	Direction	Bus	MSB	LSB
sw	input	<input checked="" type="checkbox"/>	7	0
ld	output	<input checked="" type="checkbox"/>	7	0
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		
	input	<input type="checkbox"/>		

More Info Next Cancel

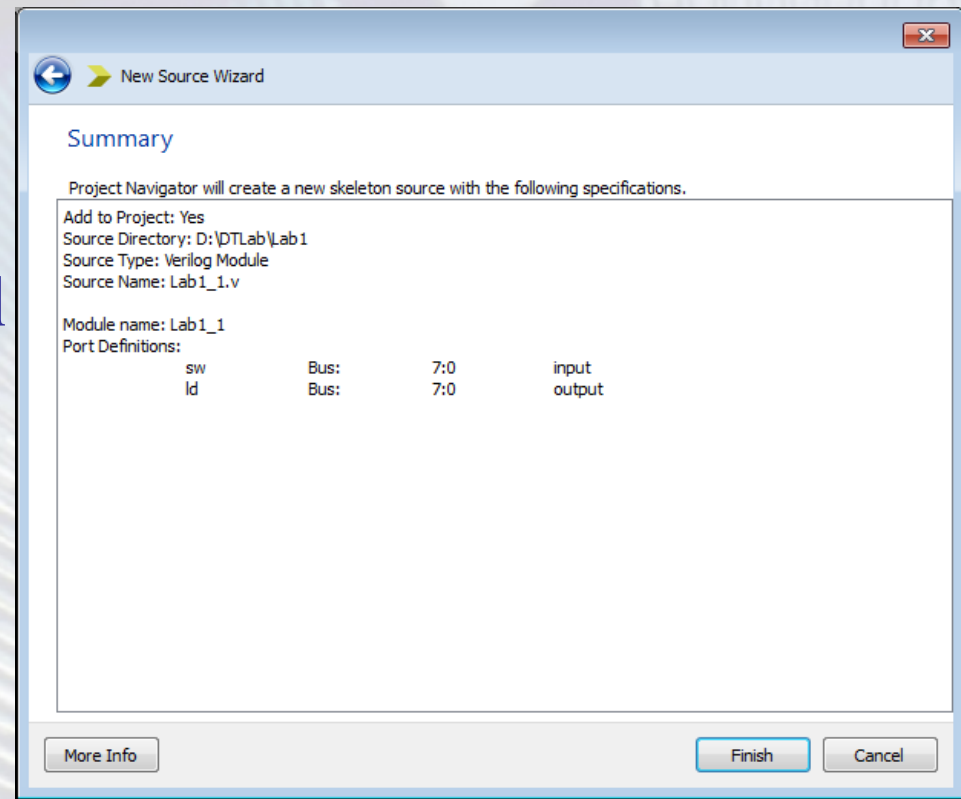
Lehetne bitenkénti megadás is (sw0, sw1,..sw7)

Projekt források létrehozása (4)

- Összefoglalás a beállításokról
- Finish után létrejön a Lab1_1.v Verilog HDL forrásfájl
- A  ikon jelzi, hogy ez a projekt hierarchia csúcsán lévő, „Top Module” forrás fájl



- Minden más projektfájl ez alá fog rendeződni



Projekt források létrehozása (5)

- A Lab1_1.v Verilog minta szövegfájl tartalma

- ``timescale 1ns/1ps`: A szimuláció során az időfelbontás 1ps, az értékek ns-ban értendők $1,23456789\text{us} = 1234,568\text{ns}$
- A (zöld) megjegyzés mező nem lényeges
- A Verilog fájl `moduleendmodule` törzse tartalmazza az általunk megadott `input output` paramétereket, továbbá ide kerül majd a `forráskód` lényeges, a működést specifikáló része, azaz a funkciót leíró kódsorok (lásd később)

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:      23:5
// Design Name:
// Module Name:      Lab1
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File
// Additional Comments:
//
////////////////////////////////////////////////////////////////
module Lab1_1(
    input [7:0] sw,
    output [7:0] ld
);

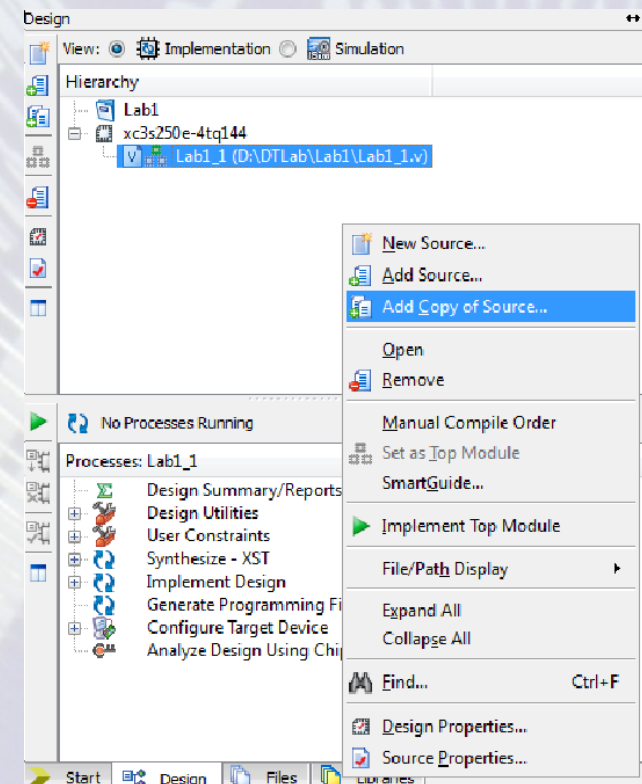
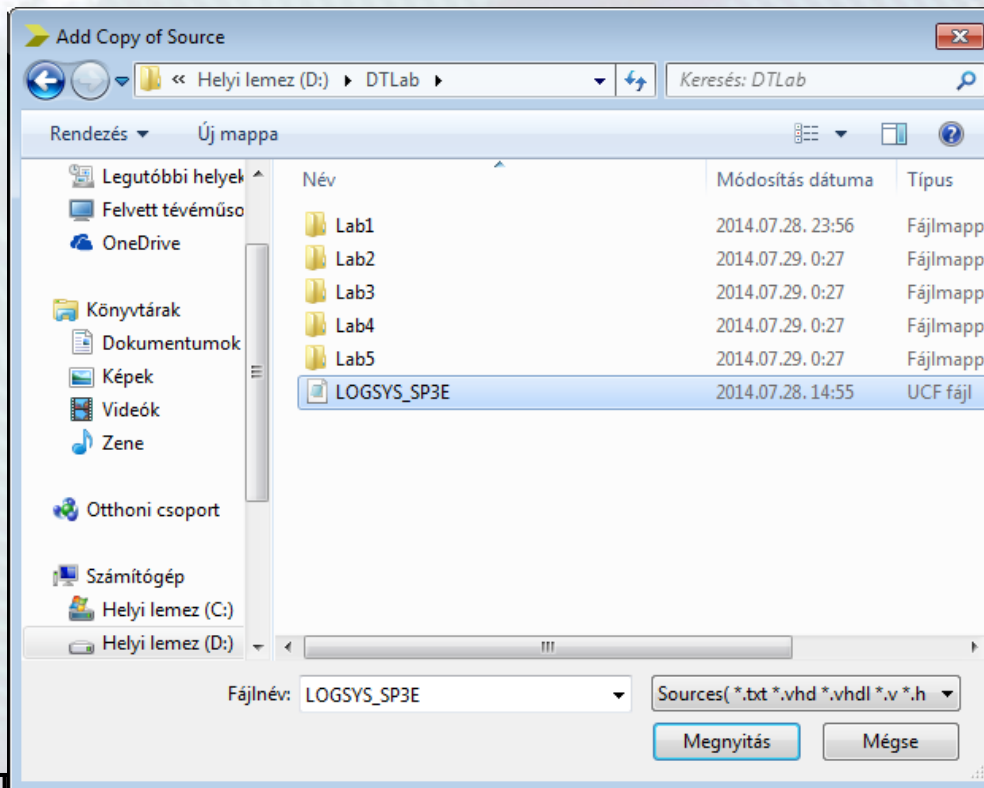
endmodule
```

Projekt források létrehozása (6)

- **Összetett feladatok esetén a terv részleteit önálló modulokba érdemes elhelyezni**
- **A modulok legyenek önálló fájlok**
- **A részekre osztás (partícionálás) finomsága (az egyes modulok komplexitása) egyéni döntés kérdése**
- **A lényeg, hogy segítse a terv megértését**
- **Sok esetben használhatunk már meglévő, könyvtári modulokat. Ezek vagy csak olvashatók, vagy ha nem, akkor csak a másolatukat adjuk hozzá a projekthez.**
- **Léteznek paraméterezhető modulok is (lásd később)**

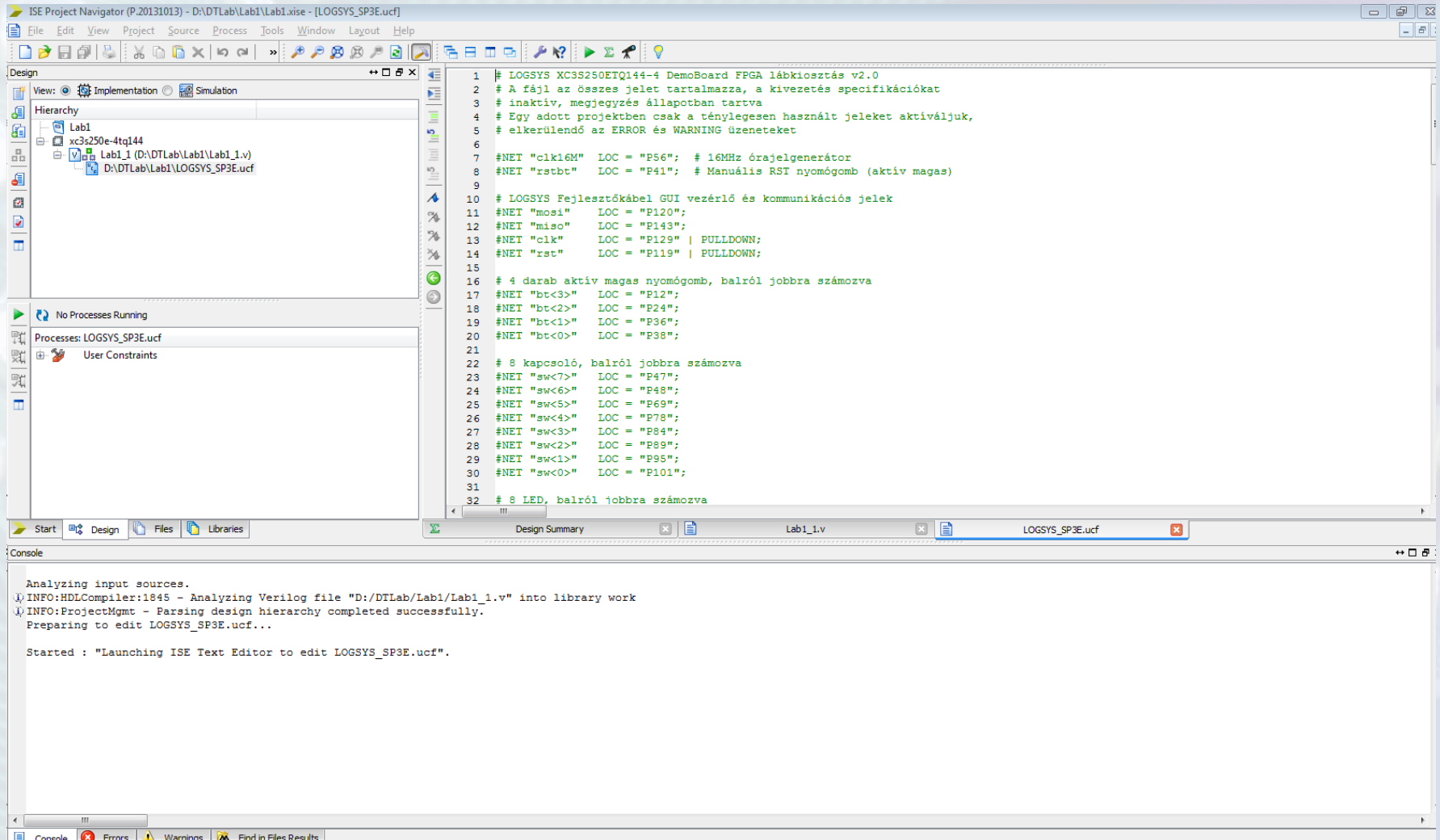
Projekt források létrehozása (7)

- A külső jelek bekötésének megadása (.UCF)
 - Projekt → Add Copy of Source ...
 - Jobb gomb, és → Add Copy of Source ...



Projekt források létrehozása (8)

- A projektstruktúra elkészült



Projekt források létrehozása (9)

- A külső jelek bekötésének specifikációja
 - A LOGSYS_SP3E.UCF fájlt a projektkönyvtárba másoltuk és a használandó jeleket aktiváljuk (a projektre szabjuk)

```
#NET "bt<1>" LOC = "P36";
#NET "bt<0>" LOC = "P38";

# 8 kapcsoló, balról jobbra számozva
#NET "sw<7>" LOC = "P47";
#NET "sw<6>" LOC = "P48";
#NET "sw<5>" LOC = "P69";
#NET "sw<4>" LOC = "P78";
#NET "sw<3>" LOC = "P84";
#NET "sw<2>" LOC = "P89";
#NET "sw<1>" LOC = "P95";
#NET "sw<0>" LOC = "P101";

# 8 LED, balról jobbra számozva
#NET "ld<7>" LOC = "P43";
#NET "ld<6>" LOC = "P50";
#NET "ld<5>" LOC = "P51";
#NET "ld<4>" LOC = "P52";
#NET "ld<3>" LOC = "P53";
#NET "ld<2>" LOC = "P54";
#NET "ld<1>" LOC = "P58";
#NET "ld<0>" LOC = "P59";

# 4 digités kijelző aktív ALACSONY szegmens vezérlésére is, a row<i> = seg<i>, megfelelő ahol 0<=i<=6 és a felső sort jelöli a 0 index
# --0--
#NET "seg_n<7>" LOC = "P34"; #7 | |
#NET "seg_n<6>" LOC = "P33"; #6 5 1
#NET "seg_n<5>" LOC = "P32"; #5 | |
```

(Uncomment)



```
#NET "bt<2>" LOC = "P24";
#NET "bt<1>" LOC = "P36";
#NET "bt<0>" LOC = "P38";

# 8 kapcsoló, balról jobbra számozva
NET "sw<7>" LOC = "P47";
NET "sw<6>" LOC = "P48";
NET "sw<5>" LOC = "P69";
NET "sw<4>" LOC = "P78";
NET "sw<3>" LOC = "P84";
NET "sw<2>" LOC = "P89";
NET "sw<1>" LOC = "P95";
NET "sw<0>" LOC = "P101";

# 8 LED, balról jobbra számozva
NET "ld<7>" LOC = "P43";
NET "ld<6>" LOC = "P50";
NET "ld<5>" LOC = "P51";
NET "ld<4>" LOC = "P52";
NET "ld<3>" LOC = "P53";
NET "ld<2>" LOC = "P54";
NET "ld<1>" LOC = "P58";
NET "ld<0>" LOC = "P59";

# 4 digités kijelző aktív ALACSONY szegmens vezérlésére is, a row<i> = seg<i>, megfelelő ahol 0<=i<=6 és a felső sort jelöli a 0 index
# --0--
#NET "seg_n<7>" LOC = "P34"; #7 | |
#NET "seg_n<6>" LOC = "P33"; #6 5 1
```

Projekt terv megvalósítása (1)

- **A tervezési feladat alapján megírjuk a „Top Module” és az esetleges egyéb modulok funkcionalitását realizáló kódrészleteket**
 - Lásd pl. Lab1_1.v **1, 2, 3** feladatok
- **Mentés, szintaktikai ellenőrzések, javítások**
- **Ha minden rendben, akkor**
 - Az elvi (funkcionális) terv ellenőrzése szimulációval
 - A terv realizálása és a generált konfiguráció letöltése a kártyára, és tesztelése a működő hardveren

Példa: Lab1_1 tervezési feladatok

- A létrehozott Lab1_1.v Verilog HDL modul üres vázába készítjük el az első tervspecifikációt

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Digitális Technika Laboratórium 1. hét
// 1. bemutató projekt: A környezet használatának bemutatása
// LED-ek vezérlése DIP kapcsolókkal
// Részfeladatok:
// 1_1_1   Egyszerű vezetékezés, műveletvégzés nélkül, 8 bites vektor jelekkel
// 1_1_2   Kettes komplement képzés
// 1_1_3   Aritmetikai műveletek vizsgálata (+, *, /, %, **) 4 bites operandusokon
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
module Lab1_1(
    input    [7:0] sw,
    output   [7:0] ld
);

// IDE ÍRANDÓ A VERILOG KÓD
// ....
// ....

endmodule
```

A terv ellenőrzése szimulációval

- A szimulátor egy számítógépes program, amely a terv logikai működését ellenőrzi
- Funkcionális szimulációnál csak a modulok bemeneti – kimeneti összefüggéseit szimulálja, a valós fizikai hatásokat, a végleges terv valódi belső paramétereit (időzítés, terhelés) nem kezeli
- A szimulátor egy tesztkörnyezetet ad, amelyben a bemeneteket jelforrásokkal, „generátorokkal” vezéreljük és vizsgáljuk „monitorozzuk” a kimenetek állapotát (mintha valóban működtetnénk)



A szimulációs környezet

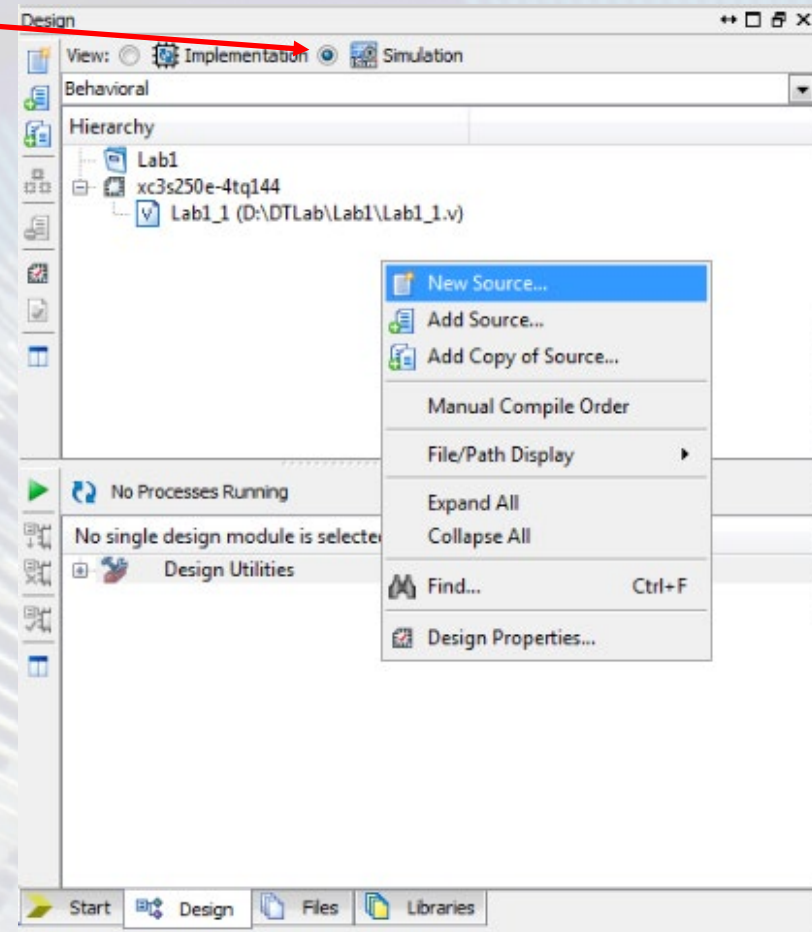
- **A Xilinx ISE beépített szimulátora az ISim**
 - Verilog és VHDL tervek funkcióját szimulálja
 - Gyári alkatrészmodell könyvtárakat használ
- **A tesztkörnyezet neve Verilog Test Fixture**
 - Speciális modul, nincsenek bemenetei/kimenetei, azaz minden belül van, ami a szimulációhoz kell
 - Persze mindent nekünk kell beletenni, mert kezdetben üres
 - Specifikáljuk a meghajtó jeleket, tesztvektorokat, gondoskodunk az alaphelyzetbe állításról és a kimeneti eredmények kiértékelési módjáról.

A szimulációs környezet

- **A tesztkörnyezet hierarchia felépítése**
 - Szimulációs környezet = (Verilog Test Fixture)
 - A beágyazott Verilog module, a tesztelt tervfájl azonosítója UUT, „Unit Under Test”
 - Ez egy általános azonosító a tesztkörnyezetbe beillesztett tetszőleges tervezési fájlra (pl. Lab1_1.v)
 - **Megjegyzés:** A szimulációs környezet felépítése általában összetettebb munka, mint a tervfájl elkészítése. Sokan nem is szeretik ezt a feladatot.
 - **De** ellenőrzés nélkül a terv ritkán működik helyesen! Ez jellemzően minden területen igaz!

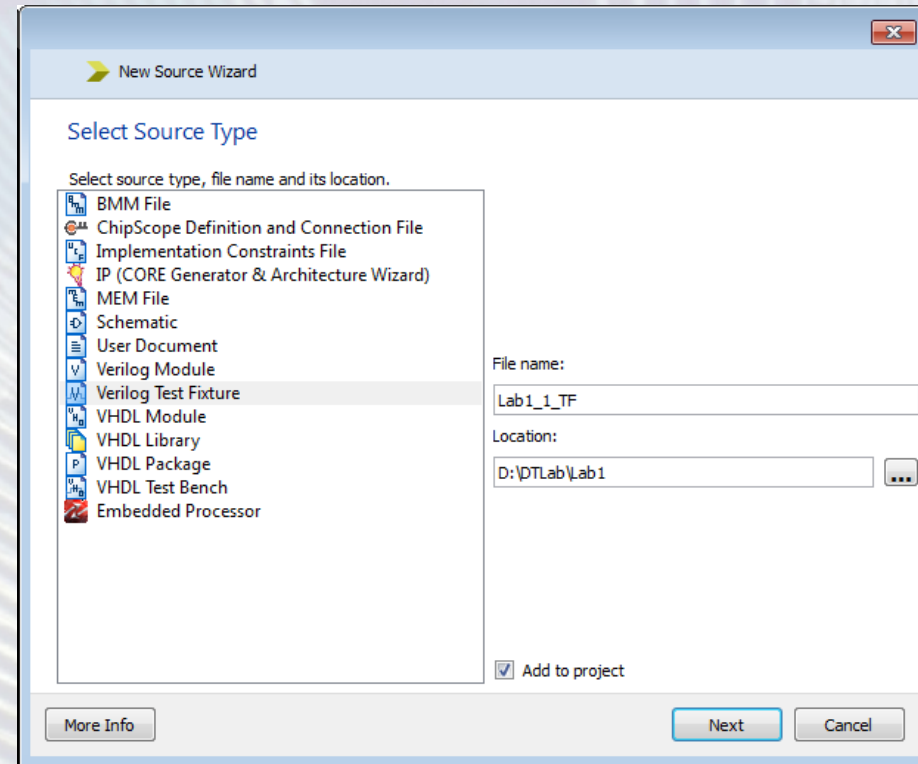
A szimuláció előkészítése

- A projekt nézetet átváltjuk szimulációs módba
- **View → Simulation**
 - Hatás: UCF nem látható
 - Alul Process ablak „kiürül”
 - Csak a legfelső szintű (jelen esetben egyetlen) Lab1_1.v tervfájl marad látható
 - Ehhez rendeljük hozzá új forrásként a teszt-környezetet (az ismert módokon...)



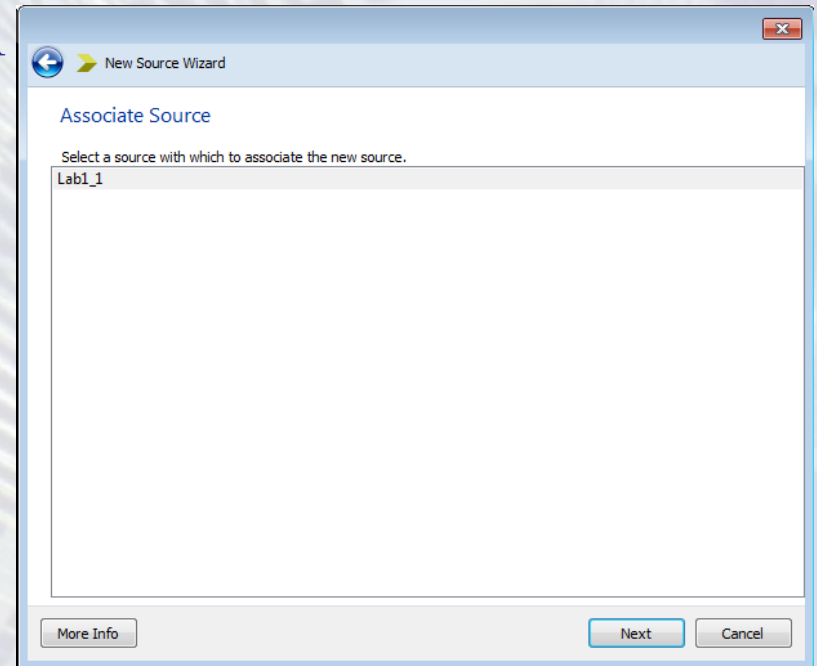
A szimuláció előkészítése

- **Az új forrás típusa Verilog Test Fixture**
 - Neve legyen a terv-hez kapcsolódó, _TF kiegészítéssel. Pl. Lab1_1_TF.v
 - Ez is egy Verilog HDL fájl, csak a használata speciális
 - Csak az ellenőrzés során használjuk



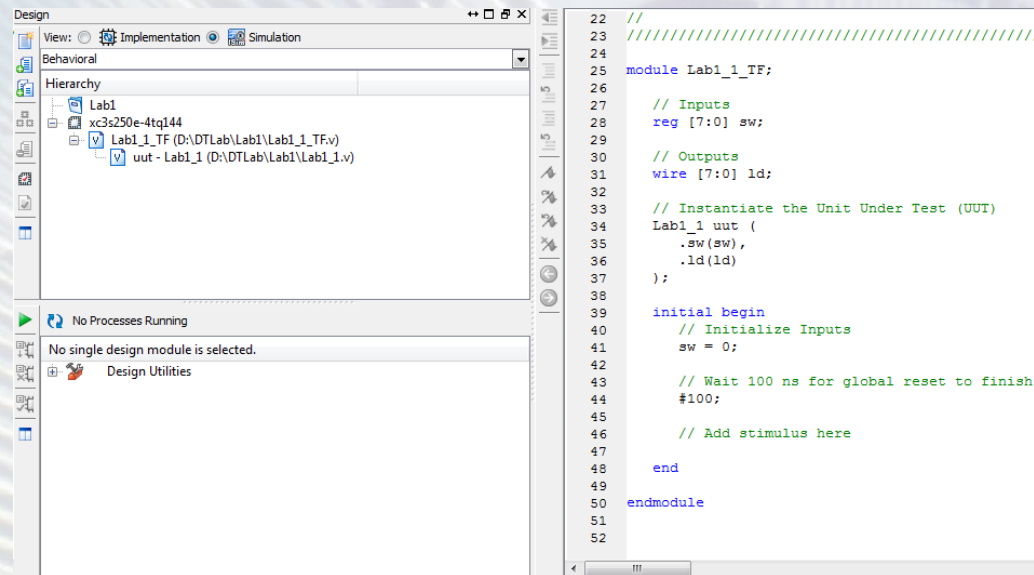
A szimuláció előkészítése

- A Verilog Test Fixture mintafájl generálása a tesztelendő fájl (UUT) interfészjellemzőitől (a bemeneti és a kimeneti jelektől) függ, ezért meg kell adni a fájlt.
 - A példában csak egy fájl van, ezért ez egyértelmű
 - Összetettebb projektnél kiválasztható, mi legyen a teszt célpontja



A szimuláció előkészítése

- A létrehozott Lab1_1_TF a szimulációs üzemmód beágyazó környezetét biztosítja
- Vezérli a bemeneteket, megfigyeli a kimeneteket
- Beépíti a Lab1_1.v tervet, mint UUT
- Inicializálja a változókat, és futtat 100ns idejű szimulációt
- Ezután várja a saját tesztvektorok, teszt-előírások megadását az **initial begin** **end** blokkon belül

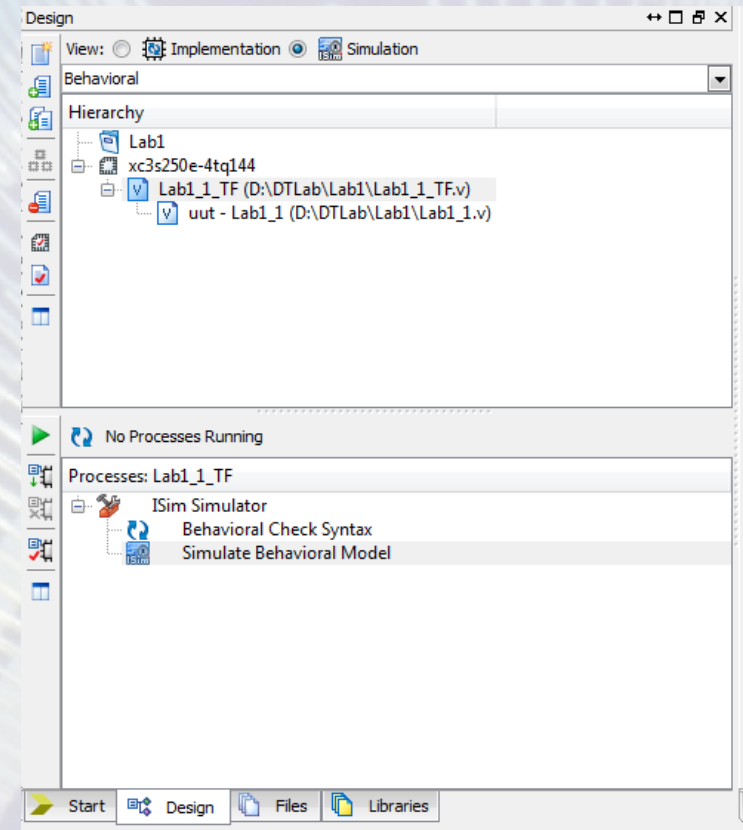


The screenshot displays the Xilinx ISE Design Suite interface. The 'Design' window on the left shows the 'Behavioral' view with a hierarchy tree containing 'Lab1', 'xc3s250e-4tq144', 'Lab1_1_TF (D:\DTLab\Lab1\Lab1_1_TF.v)', and 'uut - Lab1_1 (D:\DTLab\Lab1\Lab1_1.v)'. The 'Simulation' window on the right shows the Verilog code for 'Lab1_1_TF'. The code includes input and output declarations, instantiation of the Unit Under Test (UUT), and a testbench structure with 'initial begin' and 'end' blocks. The testbench initializes inputs, waits for 100 ns, and adds a stimulus.

```
22 //  
23 //////////////////////////////////////  
24  
25 module Lab1_1_TF;  
26  
27 // Inputs  
28 reg [7:0] sw;  
29  
30 // Outputs  
31 wire [7:0] ld;  
32  
33 // Instantiate the Unit Under Test (UUT)  
34 Lab1_1 uut (  
35     .sw(sw),  
36     .ld(ld)  
37 );  
38  
39 initial begin  
40     // Initialize Inputs  
41     sw = 0;  
42  
43     // Wait 100 ns for global reset to finish  
44     #100;  
45  
46     // Add stimulus here  
47  
48 end  
49  
50 endmodule  
51  
52
```

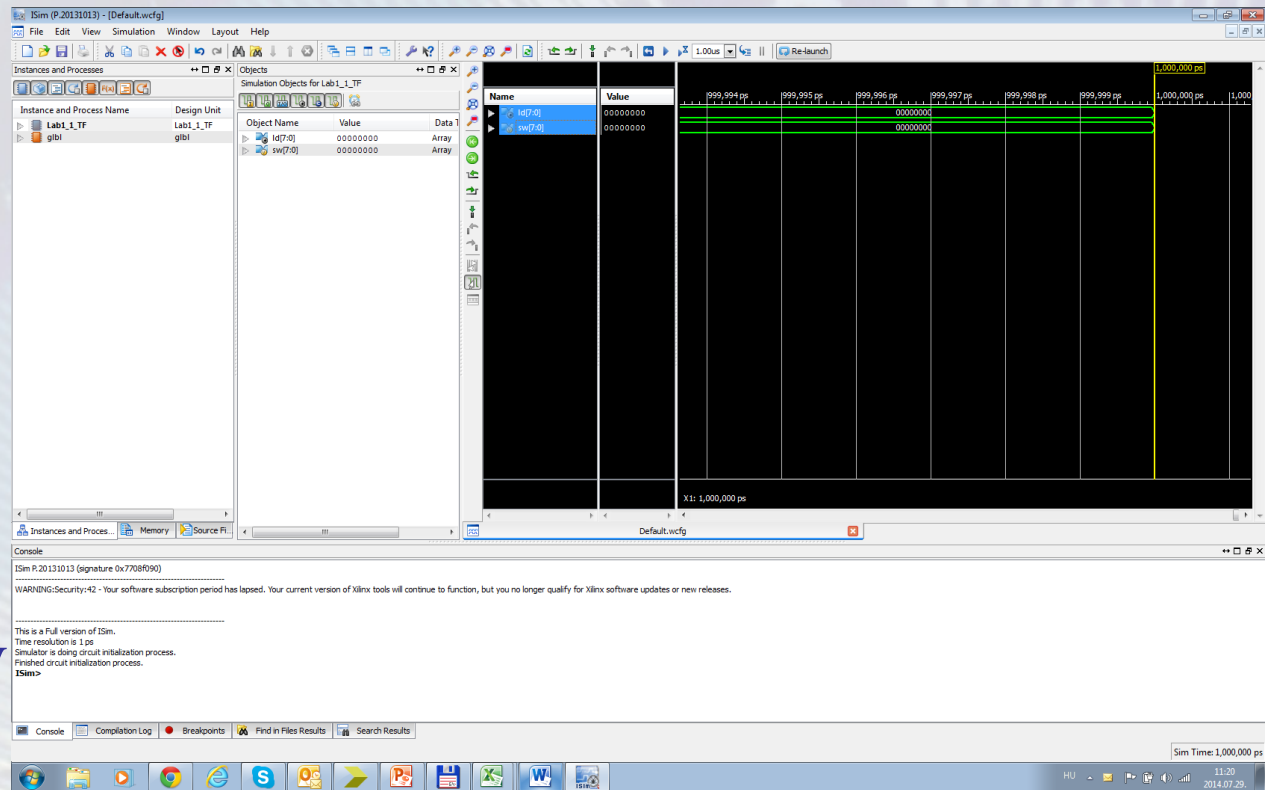
A szimuláció indítása

- A szimuláció indítása a Lab1_1_TF.v Verilog Test Fixture fájl kiválasztásával és a Process ablakban a Simulate Behavioral Model parancs kiadásával lehetséges
- Ekkor egy önálló program, az ISim szimulátor indul el
- Ez beolvassa a forrásokat és szintaktikai ellenőrzés után a szimulációs projekt futatható modelljét generálja, amit az ISim GUI-ban tesztelhetünk



A Xilinx ISim szimulátor

- A szimulátor egy összetett program környezet
 - Legfontosabb részlete a hullámforma ablak, ahol az idő függvényében látjuk, hogy a bemeneti vezérlésre hogyan reagál a tesztelt UUT.
 - Általában programozott tesztekkel dolgozunk
 - De ehhez kell a Verilog nyelv ismerete



Az ISim használata

- A HDL alapú tesztelés a programozott tesztvektor generálással használható
 - A Test Fixture fájlban előírjuk a teszt időbeli lefutását.
 - Az `initial begin end` egyfajta ütemezett lefutást biztosít, azaz 100ns ütemezéssel (``timescale`) kiadja/aktiválja/végrehajtja az új tesztvektorokat, majd leáll
- A Test Fixture újrafordítás után használható az Isim újraindításával vagy 

```
`timescale 1ns / 1ps
||
module Lab1_1_TF;

    // Inputs
    reg [7:0] sw;

    // Outputs
    wire [7:0] ld;

    // Instantiate the Unit Under Test (UUT)
    Lab1_1 uut (
        .sw(sw),
        .ld(ld)
    );

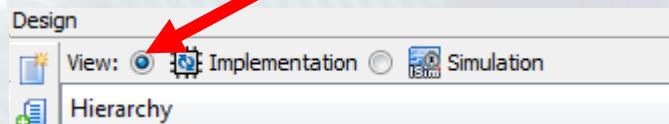
    initial begin
        // Initialize Inputs
        sw = 0;

        // Wait 100 ns for global reset to finish
        #100;

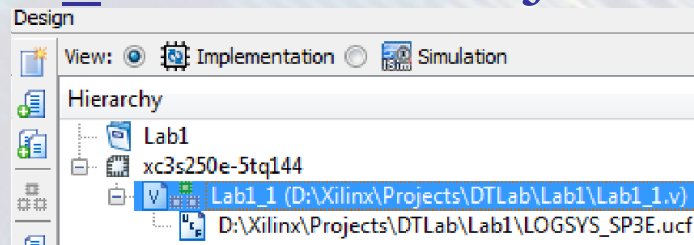
        // Add stimulus here
        #100 sw = 8'b01010101;
        #100 sw = 8'b01111000;
        #100 sw = 8'b11001100;
        #100 sw = 8'b10001011;
    end
endmodule
```

Lab1_1_1 feladat

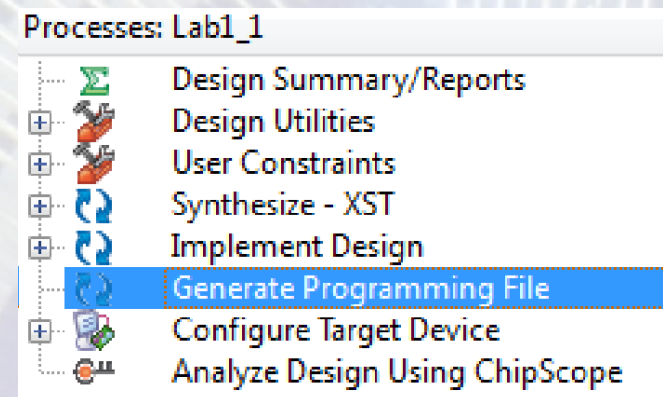
- A szimuláció befejezése után generáljuk a specifikációhoz tartozó konfigurációs adatfájlt
 - Implementációs mód (NEM SZIMULÁCIÓS!)



- A Lab1_1.v tervezői fájl aktív (+ az UCF is)

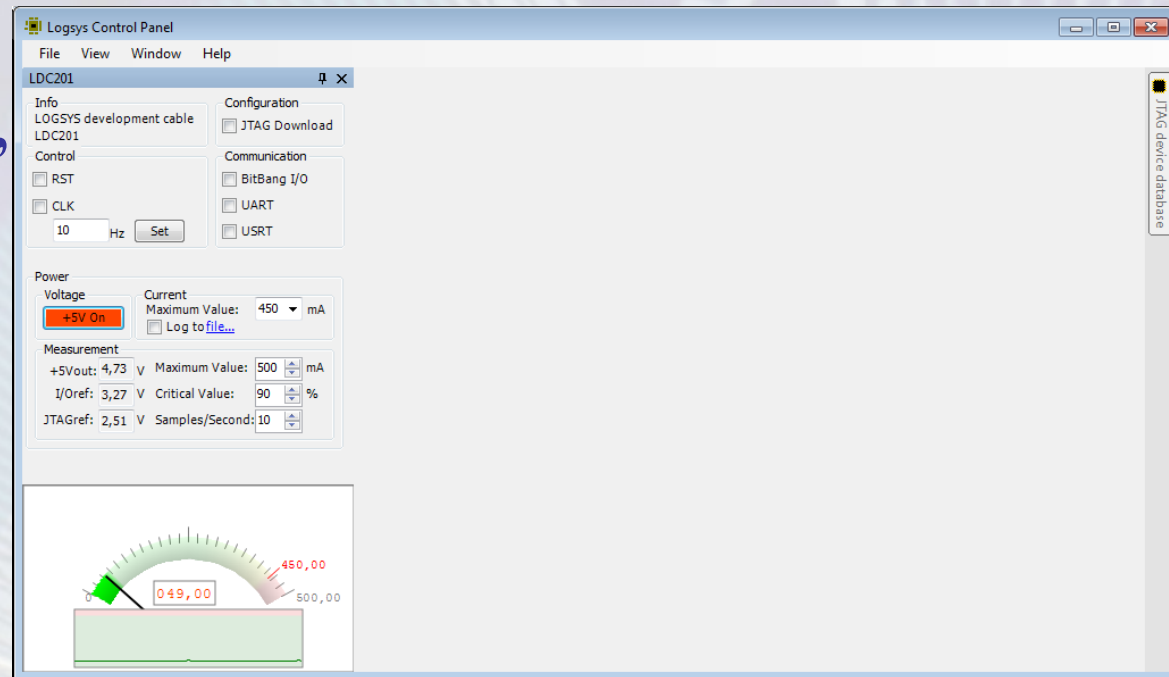
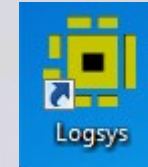


- Konfigurációs fájl generálás



Az FPGA felkonfigurálása

- Indítsuk el a LOGSYS alkalmazást
- Csatlakoztassuk a fejlesztői kábelt a számítógéphez és az FPGA kártyához
- Kapcsoljuk be a +5V tápfeszültséget, ellenőrizzük a mért adatokat:
 - V_{out} : 5V;
 - V_{ref} I/O: 3,3V;
 - V_{ref} JTAG: 2,5V;
 - I_{out} : kb. 50mA



Az FPGA felkonfigurálása

- Nyissuk meg a JTAG Download konfigurációs interfészt
- Azonosítsuk az elérhető eszközöket → Query JTAG Chain
- Válasszuk ki az FPGA-t a listából (csak egy eszköz van)
- Töltsük le az FPGA-ra a projekt könyvtárból a lab1_1.bit fájlt (Configure...)
- A zöld színű DONE LED kigyulladás jelzi a sikeres konfigurálást
- Teszteljük a működést!

