Vásárhelyi József:

Oktatási Segédlet a Xilinx ISE 14.7 szoftver használatához

VERILOG verzió

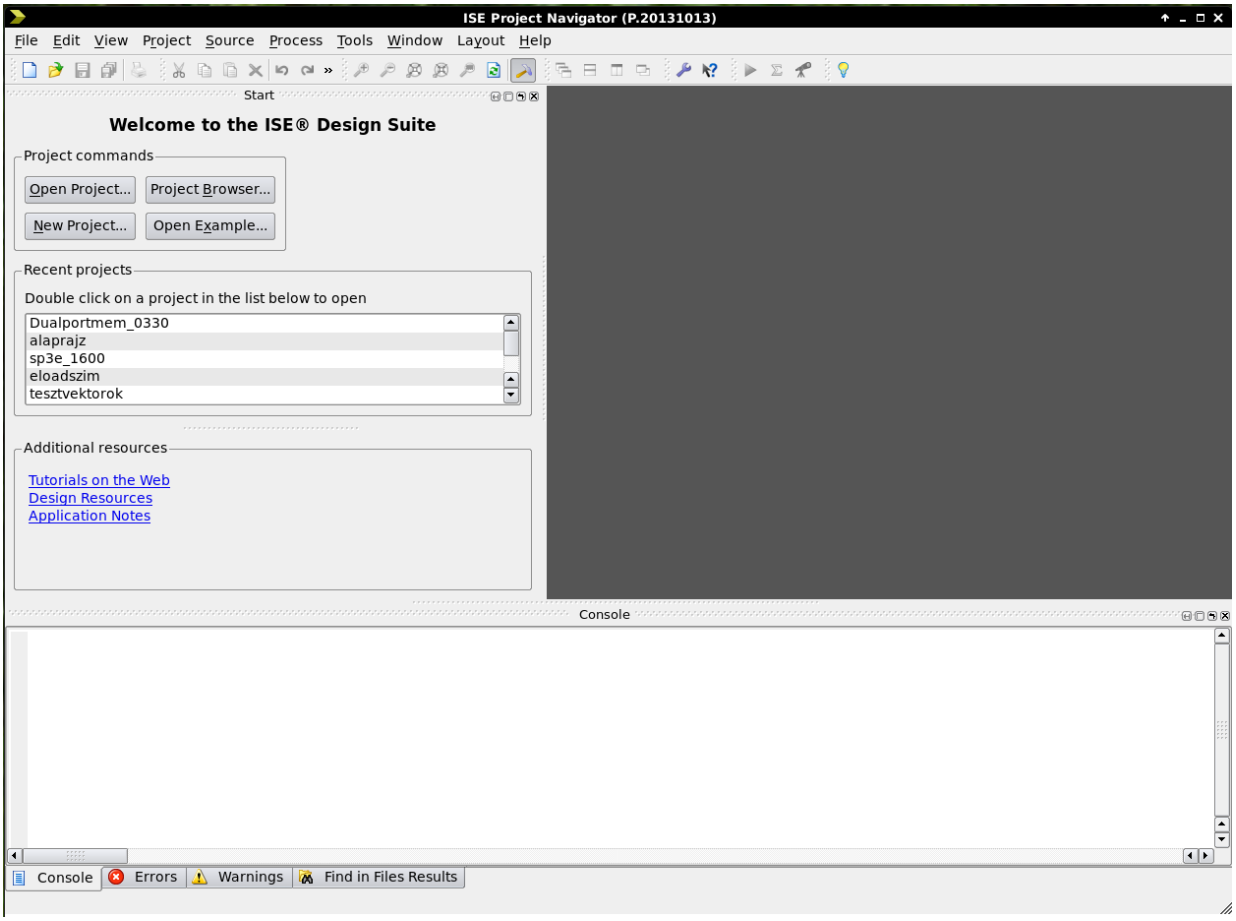
Tartalomjegyzék

Alapok

A ISE 14.7 a Xilinx cég FPGA integrált fejlesztői környezete, amely a támogatja az új típusú és régebbi FPGA fejlesztést (pl. Spartan3E, Virtex 4, Virtex 5, Artix, Zynq). A segédlet célja megismertetni az ISE alapfokú használatát (tervek kezelése, kapcsolási rajz szerkesztés, VERILOG állomány és szimuláció).
A Xilinx ISE 14.7 IDE (Integrated Development Environment – Integrált Fejlesztő Környezet) segítségével hozzunk létre egy egyszerű tervet, amelyet a gyakorló kártyán (Nexsys2) található kapcsolók, LED-ek, illetve a hétszegmenses kijelző tesztelésére használunk.
Indítsuk el a Xilinx ISE 14.7 programot: Start menüből, vagy az asztalon található ikon segítségével:

1. ábra. Xilinx ISE 14.7 indító kép

A program elindítása után a következő felülettel találkozunk:

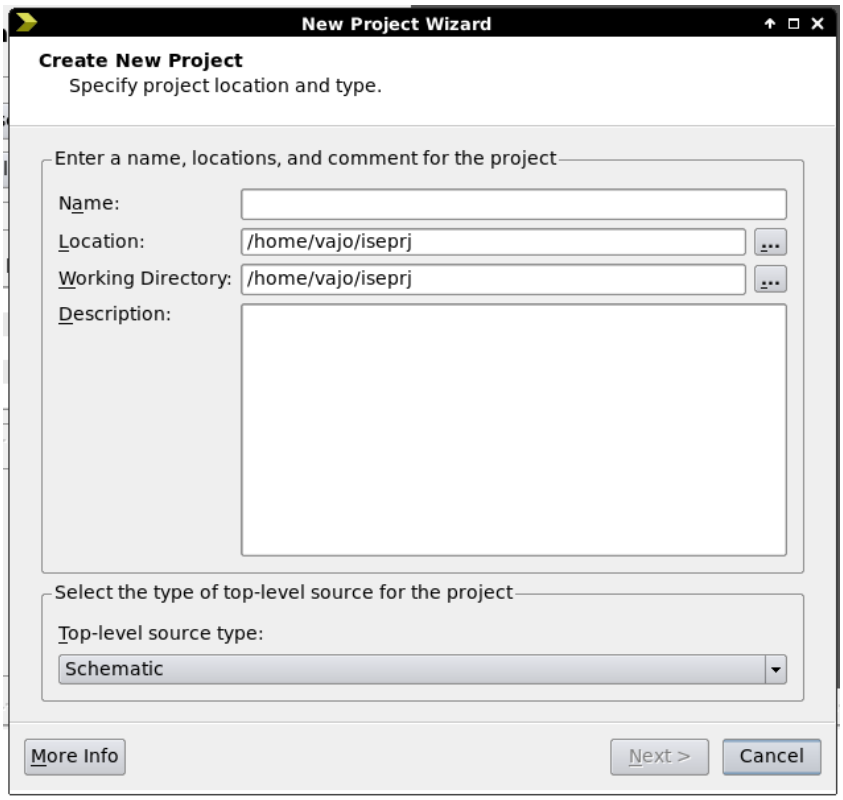


2. ábra. Xilinx ISE 14.7 kezelőfelülete

Új terv létrehozása

Első lépésként készítsünk el egy új tervet (projekt).

Menüparancs: File --> New Project
Amennyiben rendelkezünk már egy létrehozott tervvel nyissuk meg azt.
Menüparancs: Open Project
A fejlesztőkörnyezet minden tervnek külön könyvtárat hoz létre, és a tervet a megadott mappába menti.
Megjegyzés: *A gyakorlatok során minden hallgatói munkát az S lemezegységre mentsünk (\\193.6.4.39\student)! Az S lemezegységen hozzunk létre egy saját könyvtárat (neptun kód vagy ékezet nélküli név) A terv neve legyen „elsoterv”, legmagasabb szintű forrásként pedig schematic (kapcsolási rajz alapú) típust adjunk meg!*

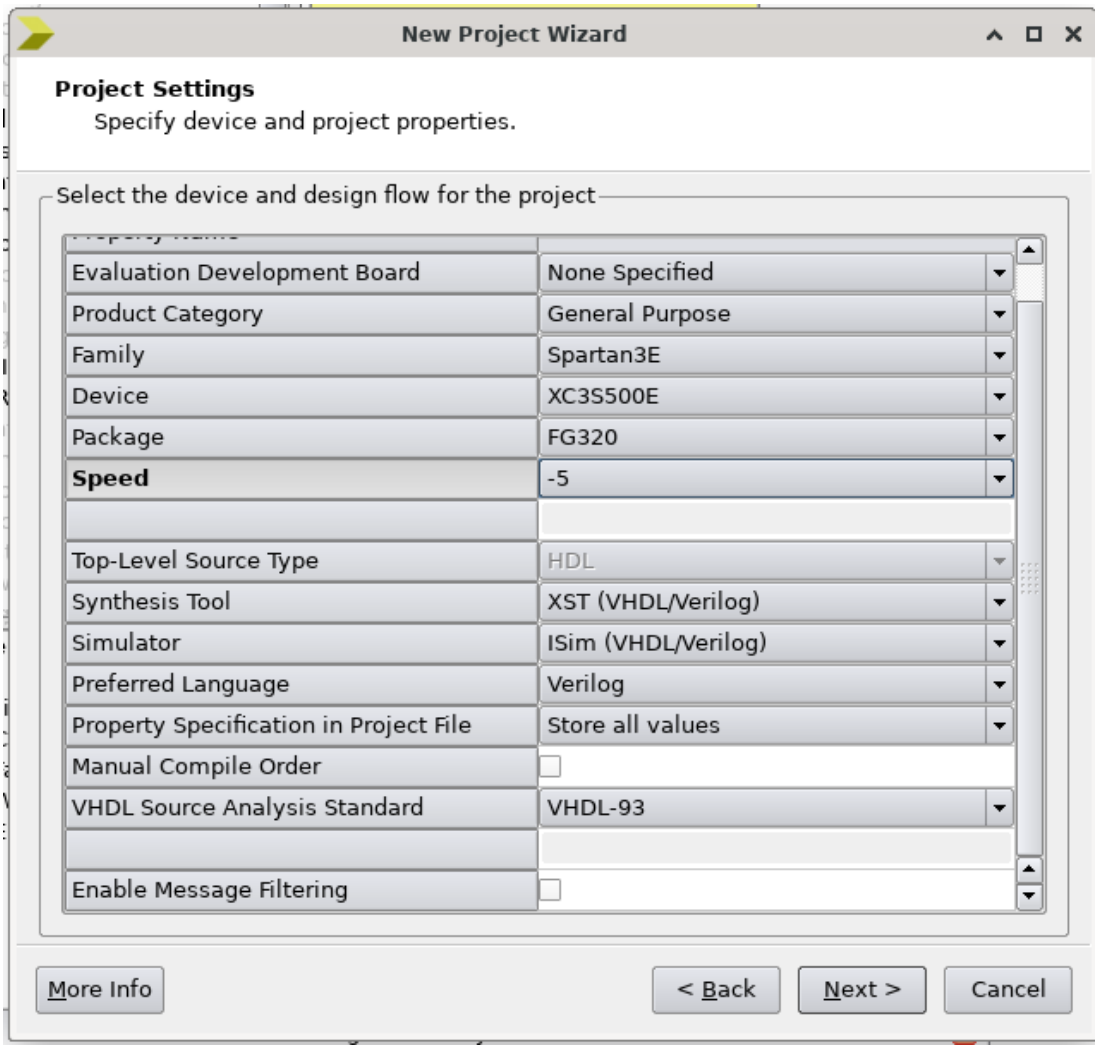


3. ábra. Új terv létrehozása

Project elsődleges forrástípusának kiválasztása
Az általunk elkészíteni kívánt terv modulokból épül fel, minden modulnak 2 különféle forrása lehet:

- Schematic: kapcsolási rajz,
- Verilog: hardverleíró nyelv,

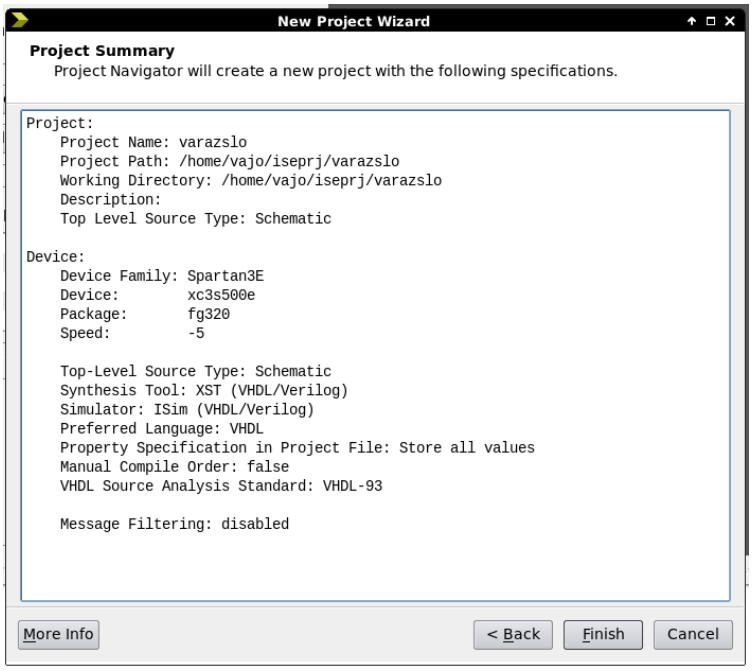
A legfelső úgynevezett „TOP” forrásban fogjuk összeállítani modulunkat – ezt a legegyszerűbben kapcsolási rajz (schematic) típus esetén tudjuk megtenni! A következő ablakban a fejlesztéssel kapcsolatos eszköz fontosabb tulajdonságait állíthatjuk be:



4. ábra. A tervben használt FPGA tulajdonságainak beállítása

A 4. ábra a használt áramkör beállításait mutatja. Ennek elemei a következők:
Product Category – Termék kategória: ***All*** (minden)
Family – Eszközcsalád: Spartan–3***E***
Device – Eszköztípus: ***XC3E500***
Package – Tokozás: ***FG320***
Speed – Sebesség: -5
Top Level Source Type – Elsődleges forrás: ***Schematic*** (később módosítható)
Synthesis Tool – Szintézis eszköze: ***XST*** (VHDL/Verilog)
Simulator – Szimulátor: ***ISE Simulator*** (VHDL/Verilog)
Preferred Language – Előnyben részesített program nyelv: ***VERILOG***

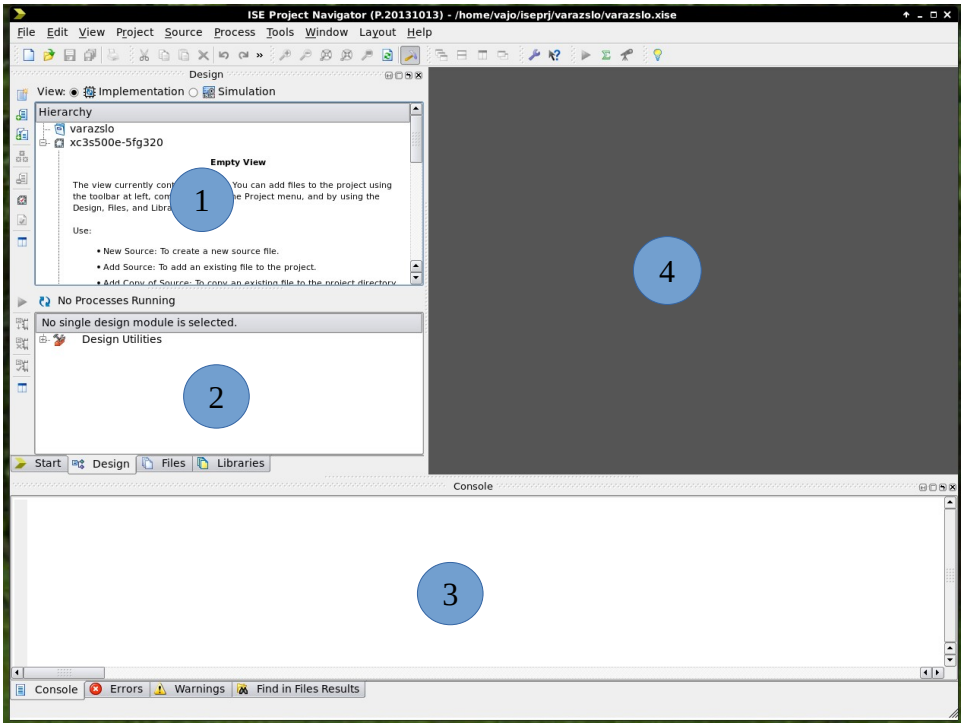
Kitöltés után kattintás a ***Next*** gombra és kattintás a **Finish** gombra miután még egyszer áttekintjük a létrehozandó terv adatait (5. ábra)



5. ábra. Az új terv paramétereinek áttekintése.

Új forrás állomány létrehozása/hozzáadása a tervhez

Új kapcsolási rajz/Verilog állomány létrehozása **Project --> New Source** menüparanccsal történik! Egy létező forrásállomány hozzáadása pedig **Project -> Add Source** menüparanccsal történik**!** Egy létező forrásállomány másolatának hozzáadása a **Project -> Add Copy of Source** menüparanccsal történik**!**



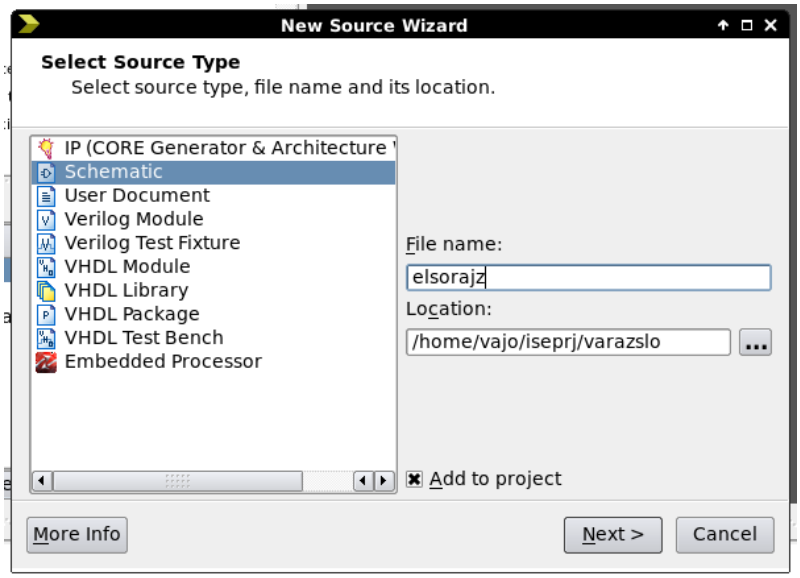
6. ábra. Kezelői felület elemei: 1 – források; 2 – folyamatok; 3 – üzenetek; 4 – munkatér.

Forrás hozzáadása a projekthez

Hozzuk létre tervünk első kapcsolási rajzát! Első tervünk a következő funkciókat fogja megvalósítani: Egy kapcsoló hatására bekapcsolja egymás után a gyakorlópanelen (Nexsys2) lévő

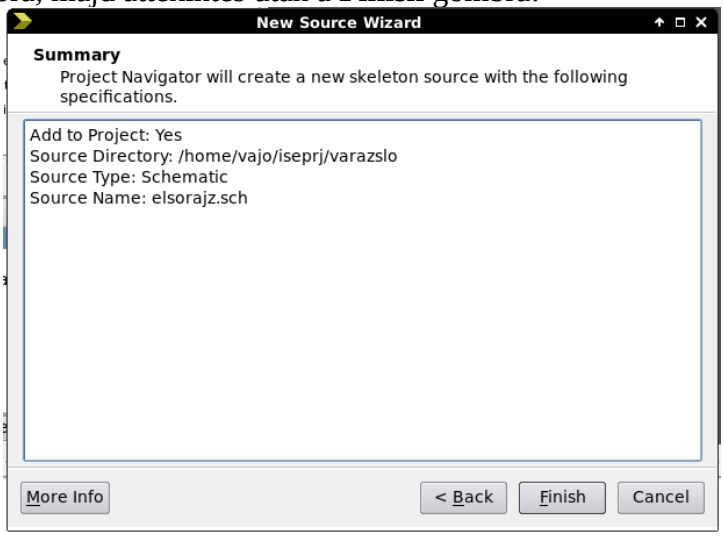
LED-eket, valamint az egyes szegmenseket a kijelzőkön – amennyiben a kapcsolót kikapcsoljuk működés közben, a kijelző őrzi az előző állapotot.
Ehhez szükségünk van egy időzítő áramkörre, amely az egyes LED-ek ill. szegmensek bekapcsolásának időközét állítja be, a láthatóság érdekében (amennyiben csak szimulációt kívánunk végezni eltekinthetünk az órajel-osztó áramkör megépítésétől)!
Megjegyzés: Az Nexys2 kártya órajelének frekvenciája 50 MHz. Ezt az órajelet osztjuk az idozito.sch – modullal kisebb vezérlő frekvenciákra. Az osztott órajellel vezéreljük az áramköröket. Így a panelen láthatók lesznek a változások.

Első kapcsolási rajz létrehozása:
Adjunk hozzá projektünkhöz egy új kapcsolási rajzot.
Menüparancs Project -> New Source
A parancs kiadása után a következő felülettel találkozunk (7. ábra):



7. ábra. Új forrás állomány hozzáadás.

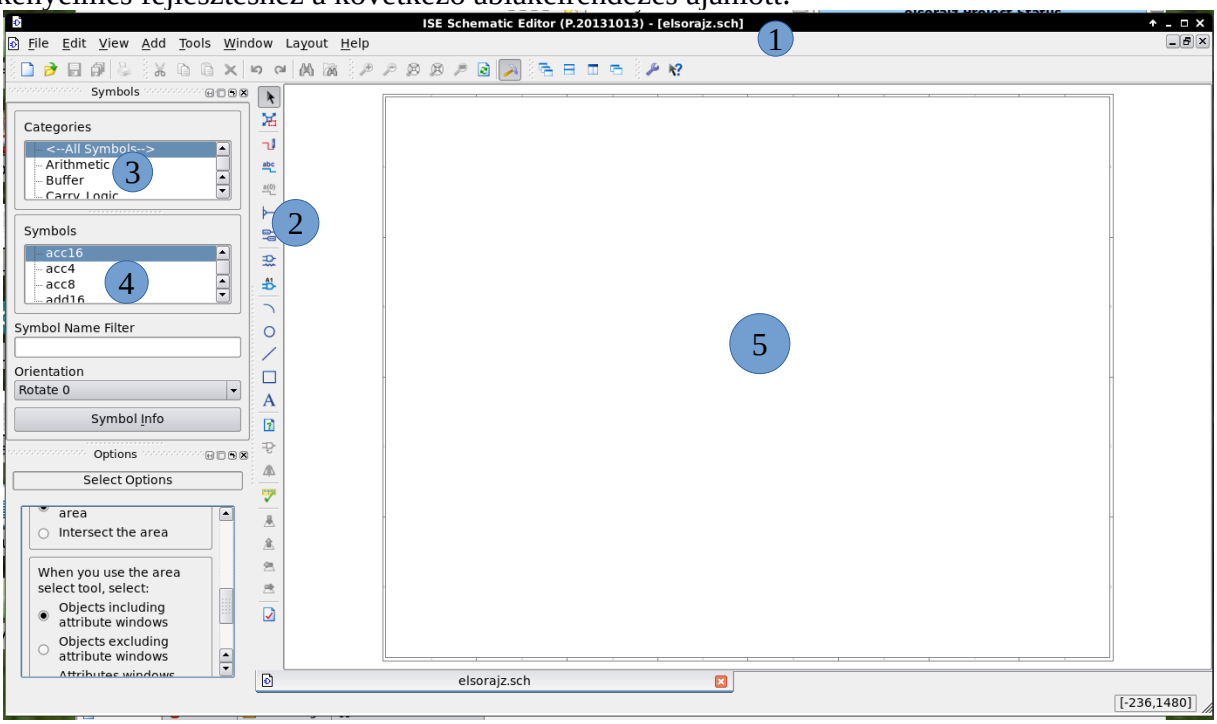
Legyen a forrásunk típusa **schematic** (kapcsolási rajz), a neve **idozito**!
Kattintás a **Next** gombra, majd áttekintés után a **Finish** gombra!



8. ábra. Az új rajz létrehozásának befejező lépése.

Az ISE létrehozza az új kapcsolási rajzot és megnyitja a grafikus felületet a rajzoláshoz.
Kezdjünk hozzá a terv megvalósításához!

A kényelmes fejlesztéshez a következő ablakelrendezés ajánlott:



9. ábra. Kapcsolási rajz nézet

1: Menüsor
2: Fontosabb ikonok
3: Alkatrész könyvtárak (Categories) ablak
4: Áramköri szimbólumok (Symbols) ablak
5: Munkatér

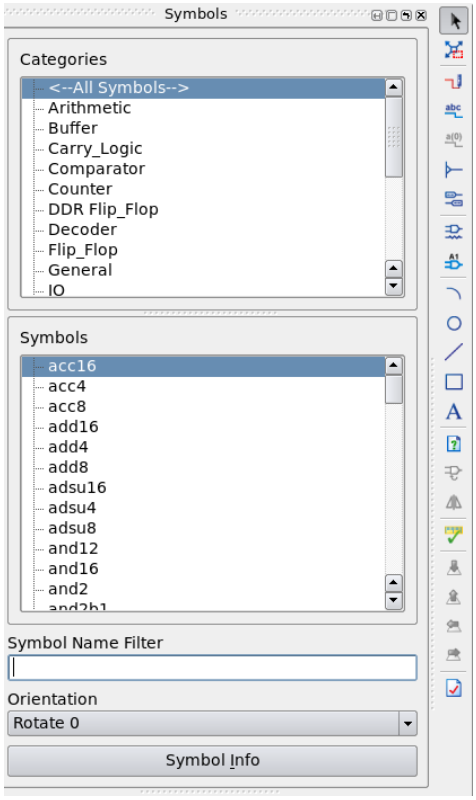
Megjegyzés: A fejlesztő lehetővé teszi az egyes ablakok kiemelését a környezetből. Például a kacsolási rajz szerkesztése így kényelmesebb. Ügyeljünk rá, hogy az **idozito.sch** fül legyen aktív! Ha az **idozito.sch** fülre egér jobb gombbal kattintunk a **Float** parancs kiválasztásával teljes képernyős ablakra válthatunk!

Kapcsolási rajz megvalósítása

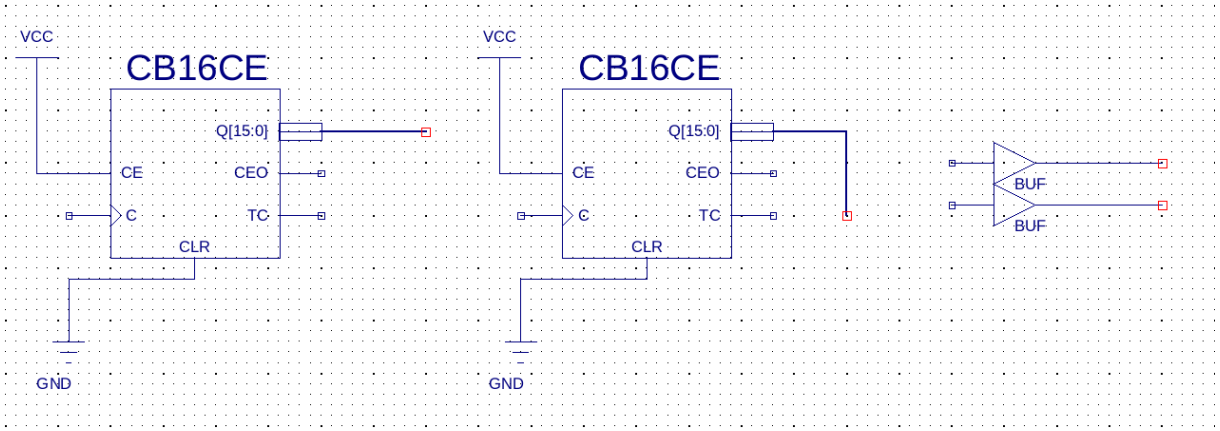
Válasszuk ki a szükséges alkatrészeket a bal oldalon található **Sources** ablak **Symbols** fülének segítségével.

Alkatrészek kiválasztása

Az alkatrész keresés történhet kategóriák alapján, de használhatjuk a névkeresőt is (Symbol Name Filter) – lásd 9. ábra
Mint említettük órajelünk 50MHz-es ezért a kapcsolók és nyomógombok pergésmentesítéshez és az időzítésekhez létre kell hoznunk egy frekvencia osztó áramkört, amit két 16 bites számlálóval valósítunk meg. Ezzel előállítjuk az időzítéshez szükséges belső órajelet is. Hozzuk létre a 11. ábrán látható kapcsolást.



10. ábra. Alkatrészek kiválasztása.



11. ábra. Első rajz részlet

A felhasznált alkatrészek a következők:

- CB16CE: 16 bites számláló, működés – órajel – engedélyezéssel, itt frekvenciaosztóként működik
- BUF: Nem invertáló puffer
- VCC: Pozitív tápfeszültség (+5V; magas szint (H); 1)
- GND: Földpont (0V; alacsony szint; 0)

A kapcsolási rajzot az oldalsávon található ikon-parancsokkal – vagy a menüben az „**Add”** legördülő menüben található ugyanazon parancsokkal tudjuk megvalósítani.

Lehetséges parancsok:

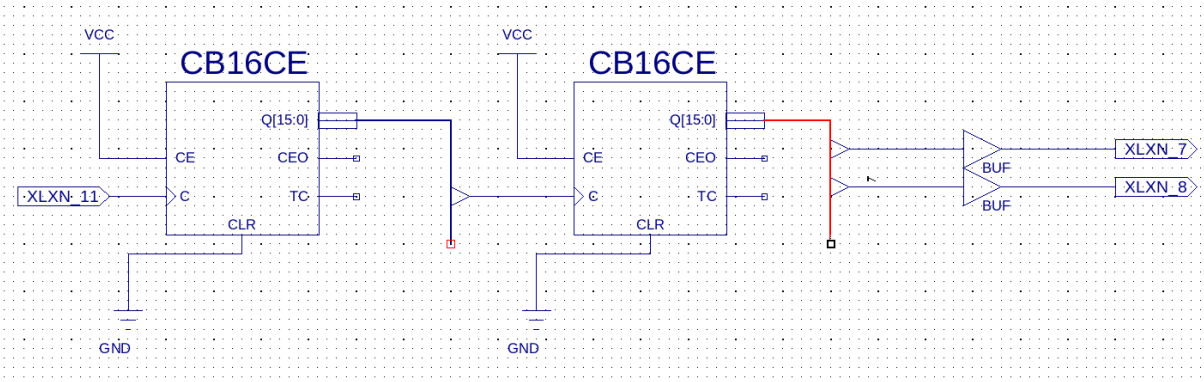
- **Select** Kiválasztás/mozgatás.
- **Add Wire** Vezeték rajzolása
- **Add Net Name** Vezetékezés elnevezése.
- **Rename Selected Bus** A kijelölt busz átnevezése.
- **Add Bus Tap** Sin – busz – csatlakozás hozzáadása
- **Add I/O Marker** Be-, kimeneti port/láb hozzáadása.
- **Add Symbol** Szimbólum – könyvtári alkatrész vagy saját alkatrész hozzáadása.
- **Check Schemtic** Kapcsolási rajz ellenőrzése

Megjegyzés: A kapcsolási rajz elkészítése a fenti parancsokkal történik. Próbálkozzunk bátran!

Jól látható, hogy a számláló kimenetéhez csatlakoztatott vezeték sínrendszer ezért vastagabb a többinél. A sínrendszerhez egy vezeték csatlakozása az **Add Bus Tap** paranccsal történik.

Rajzoljunk az első számláló kimeneti sínjéhez egy és a a második számláló kimeneti sínjéhez két darab busz csatlakozót.

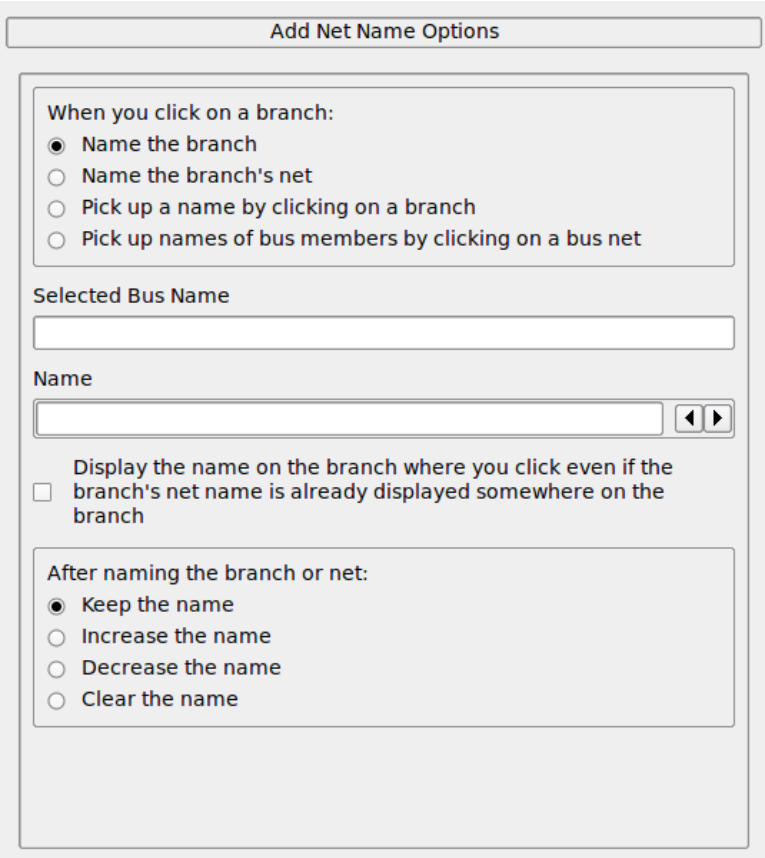
Végezzük el a hálózat teljes bekötését, és helyezzük el a be és kimenetekre az I/O markereket. Be – kimenet hozzáadása: **Add I/O Marker**



12. ábra. I/O markerek elhelyezése.

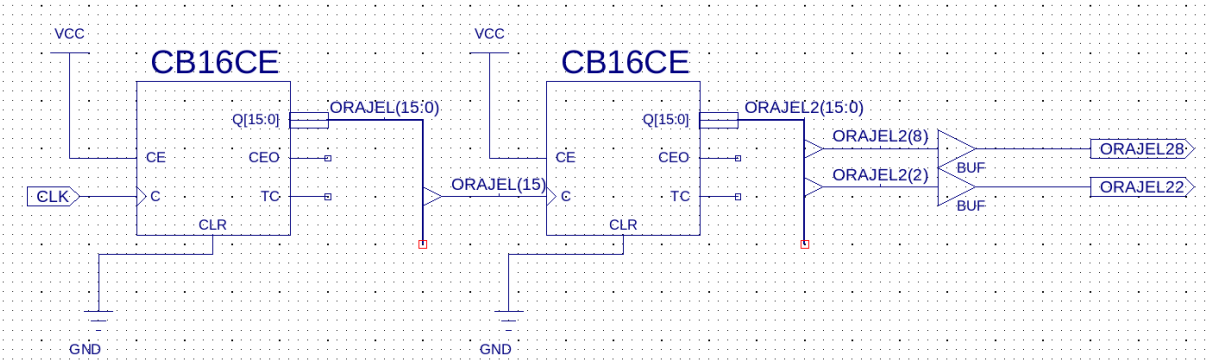
I/O markerek, vezetékek, buszok elnevezése

Nevezzük el a különböző buszokat és I/O markereket.
Add Net Name – a bal alsó ablakban az **Options** fülnél a **Name** sorba írjuk a nevet, majd kattintsunk az elnevezni kívánt vezetékre, ügyeljünk, hogy ez a vezeték elég hosszú legyen a felirat elhelyezéséhez (lásd 13. ábra)! I/O marker elnevezése: dupla kattintás a marker-en!



13. ábra. Vezetékek elnevezése ablak.

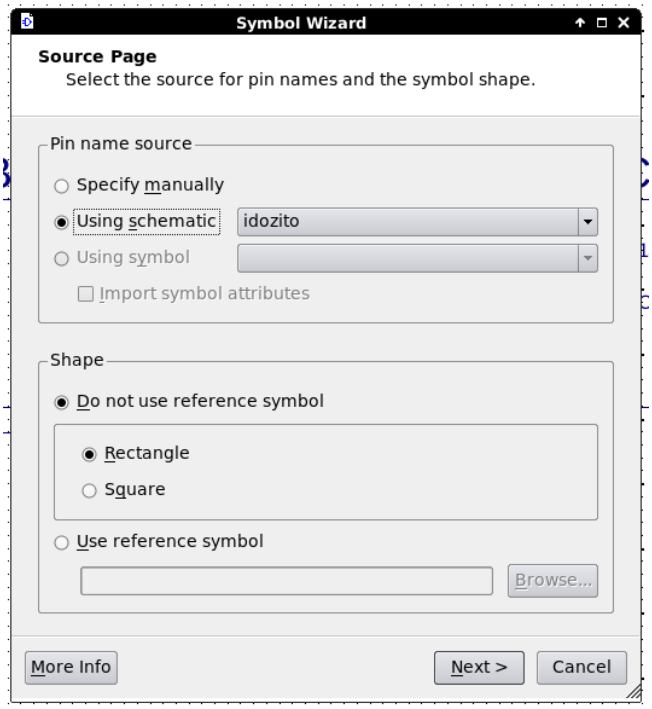
A buszok, illetve leágazásaik elnevezése utal arra, hogy melyik vonalról (bitről) van szó, míg a markerek elnevezése tetszőleges
CLK: clock-órajel – 50 MHz;
ORAJEL(15) az órajel frekvenciájának 16-od része;
ORAJEL2(2): az órajel frekvenciájának 18-Ad része;
ORAJEL2(8): az órajel frekvenciának 24-ed része.
Az ORAJEL22 jelét a későbbiekben használhatja a kapcsolók, vagy nyomógombok pergésmentesítésére (prellezés: a mechanikus kapcsolók nyomógombok nem adnak rögtön stabil jelet, két állapot között „peregnek”, s bár ez emberi mértékkel gyorsnak tekinthetjük, figyelembe véve a digitális rendszerek órajelét (n*MHz) ez komoly gondot okozhat)!



14. ábra. Az időzítő áramkör kapcsolási rajza

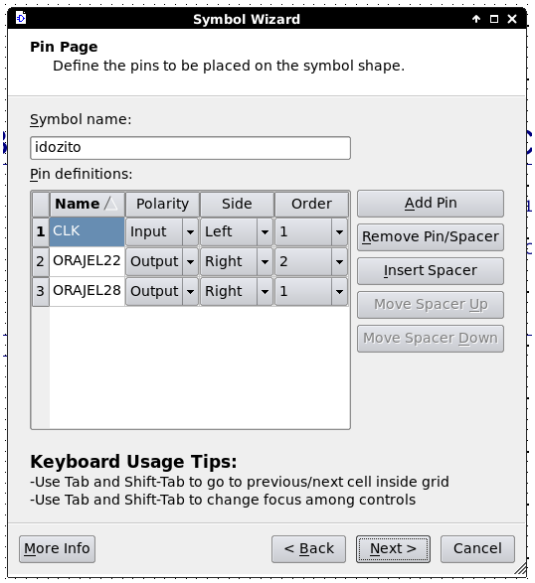
Saját alkatrész létrehozása

A fejlesztőkörnyezet lehetőséget nyújt saját alkatrész definiálására az átláthatóság megkönnyítése érdekében – használjuk ki ezt a lehetőséget!
A **Tools** menüben válasszuk ki a **Symbol Wizard**ot menüparancsot!



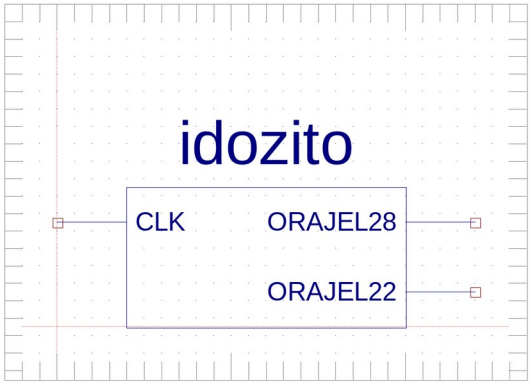
15. ábra. A saját alkatrész definiálása a „szimbólum varázsló” segítségével történik.

Megjegyzés: Fontos, hogy a **Using Schematic** (így az általunk megadott kapcsolási rajzból hozunk létre szimbólumot!), illetve a **Rectangle** legyen bejelölve (az alkatrész téglalap alakú lesz). A következő ablakban az alkatrész be és kimenetei lesznek megjelenítve, ezen ne változtassunk (lásd 16. ábra)

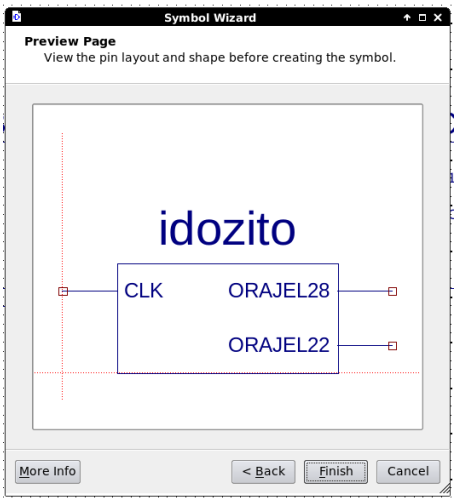


16. ábra. Saját alkatrészünk I/O kivezetései

A Next-re kattintva alkatrészünk geometriai tulajdonságait módosíthatjuk (méretezhetjük) – lásd 17. ábra.



17. ábra. Geometriai tulajdonságok.

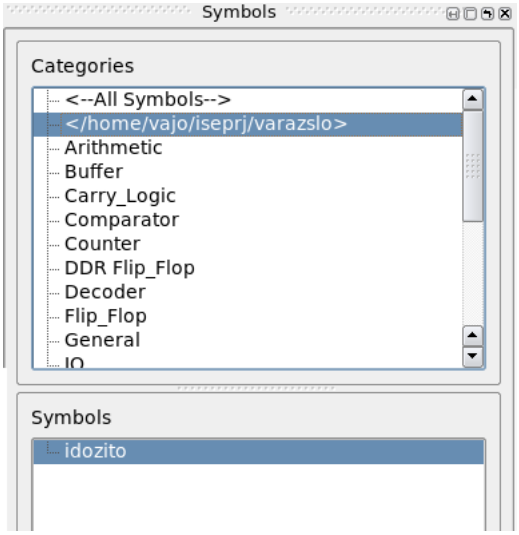


18. ábra. Alkatrész előnézet.

Az utolsó ablakban megtekinthetjük alkatrészünk elő nézetét. A befejezéshez kattintsunk a **Finish** gombra.

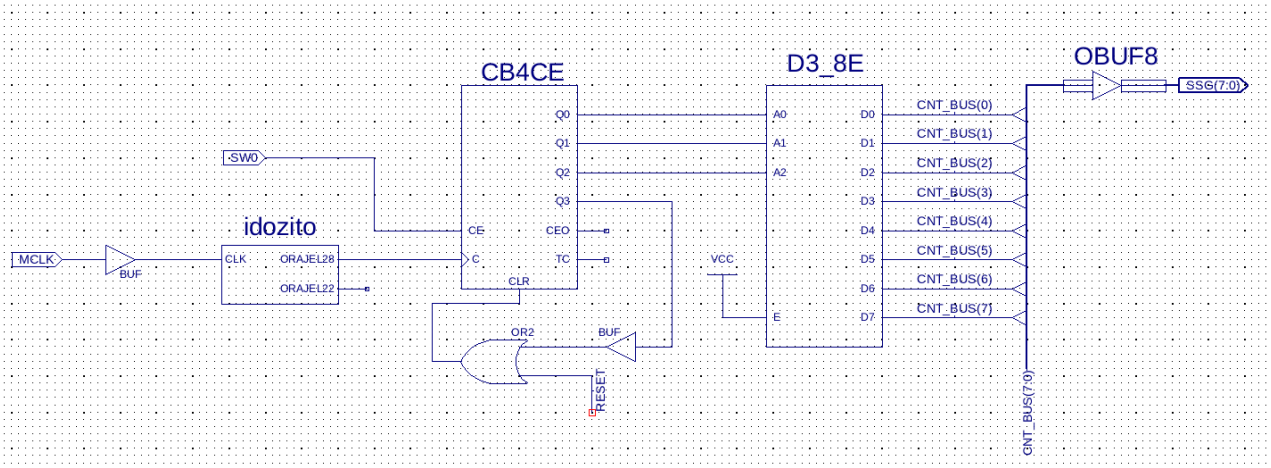
Első alkalmazás megvalósítása

Hozzunk létre egy új kapcsolási rajzot. Mivel ez lesz az első megvalósításunk neve legyen „***proba***” típusa kacsolási rajz. Váltsunk a proba.sch fülre!
Amennyiben minden lépést helyesen végeztünk „idozito” alkatrészünknek látszania kell a szimbólumok között:

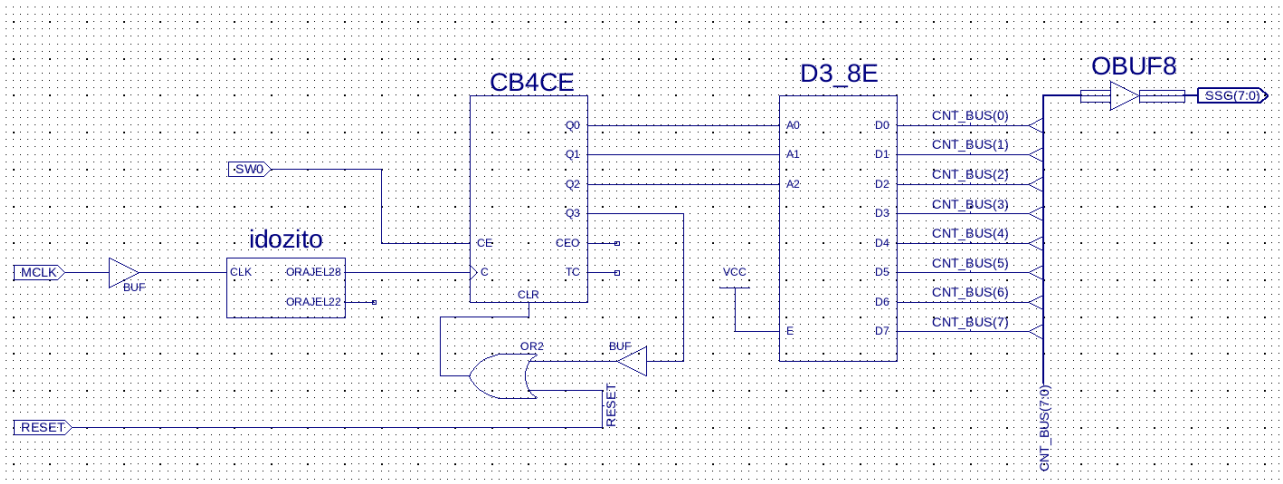


19. ábra. A saját alkatrészünk helye az alkatrész könyvtárban.

Folytassuk a tervbevitelt és készítsük el a következő áramkört (lásd 20. ábra, befejezett rajz – lásd 21. ábra):



20. ábra. Első alkalmazás kapcsolási rajza.



21. ábra. Elkészített proba.sch

A felhasznált elemek az 20. ábrán:

I/O markerek:

- **MLCK** (órajel),
- **SSG(7:0)** (7szegmens)
- **SW0** (kapcsoló)

Alkatrészek:

- IDOZITO (saját alkatrész)
- BUF; OBUF8 (pufferek)
- CB4CE (számláló)
- D3_8E (dekóder)
- OR2 (két bemenetű VAGY kapu)
- VCC (pozitív tápfeszültség; High szint; 1)

Kapcsolási rajz ellenőrzése

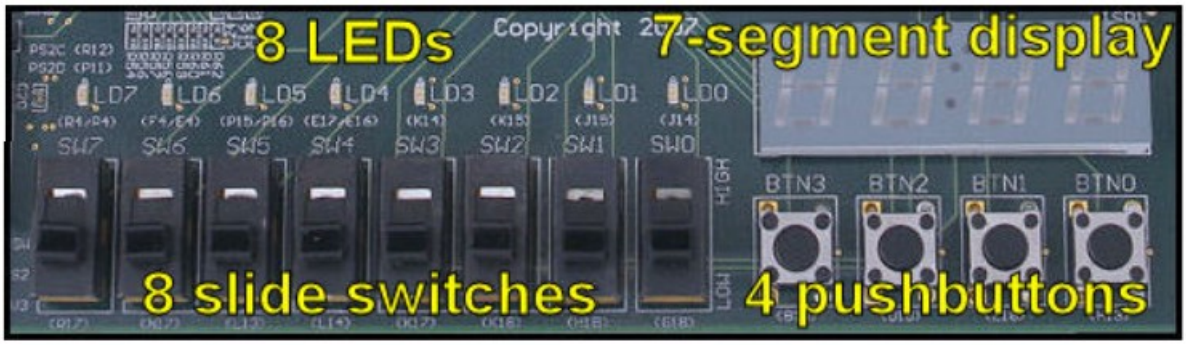
A kapcsolási rajz ellenőrzése sokat segít az elvi hibák kiküszöbölésében. Fordítás előtt mindenképpen ajánlott elvégezni. A rajz elellenőrzését vagy a **Check Schemtic** ikonra kattintva vagy pedig a menüből **Tools Check Schemtic** paranccsal tudjuk megtenni. Amennyiben az ellenőrzés sikeres úgy a következő üzenet jelenik meg:

Start DRC...
No errors or warnings were detected

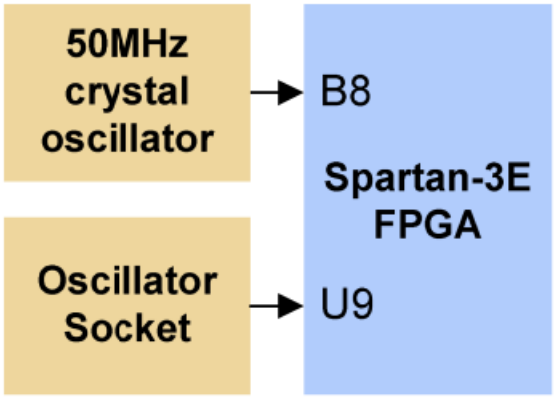
Ha az ellenőrzés hibát észlel úgy a hiba helyét megtaláljuk ha rákattintunk a hiba üzenetre (Figyelem nem az ***Error –*** szóra hanem az üzenetben lévő hivatkozásra!).

Lábkiosztás megvalósítása

Az elkészített kapcsolási rajz fizikailag az FPGA belsejében található és a külvilághoz az áramkör lábain keresztül csatlakozik. Ezért meg kell adnunk, hogy mely integrált áramköri lábakra vezetjük ki a jeleket. Ezt kétféle módon tudjuk megtenni. Szöveges és grafikus szerkesztői felületen.
A Nexsys2 kártyán található kapcsolók, nyomógombok, LED-ek és hétszegmenses kijelző lábkiosztása a mellékelt ábrákon láthatók (22. ábra – Nexsys2 fénykép részlet; 24. ábra – lábkiosztás kapcsolók, LED-ek, nyomógombok, hétszegmenses kijelzők; 23. ábra oszcillátor).



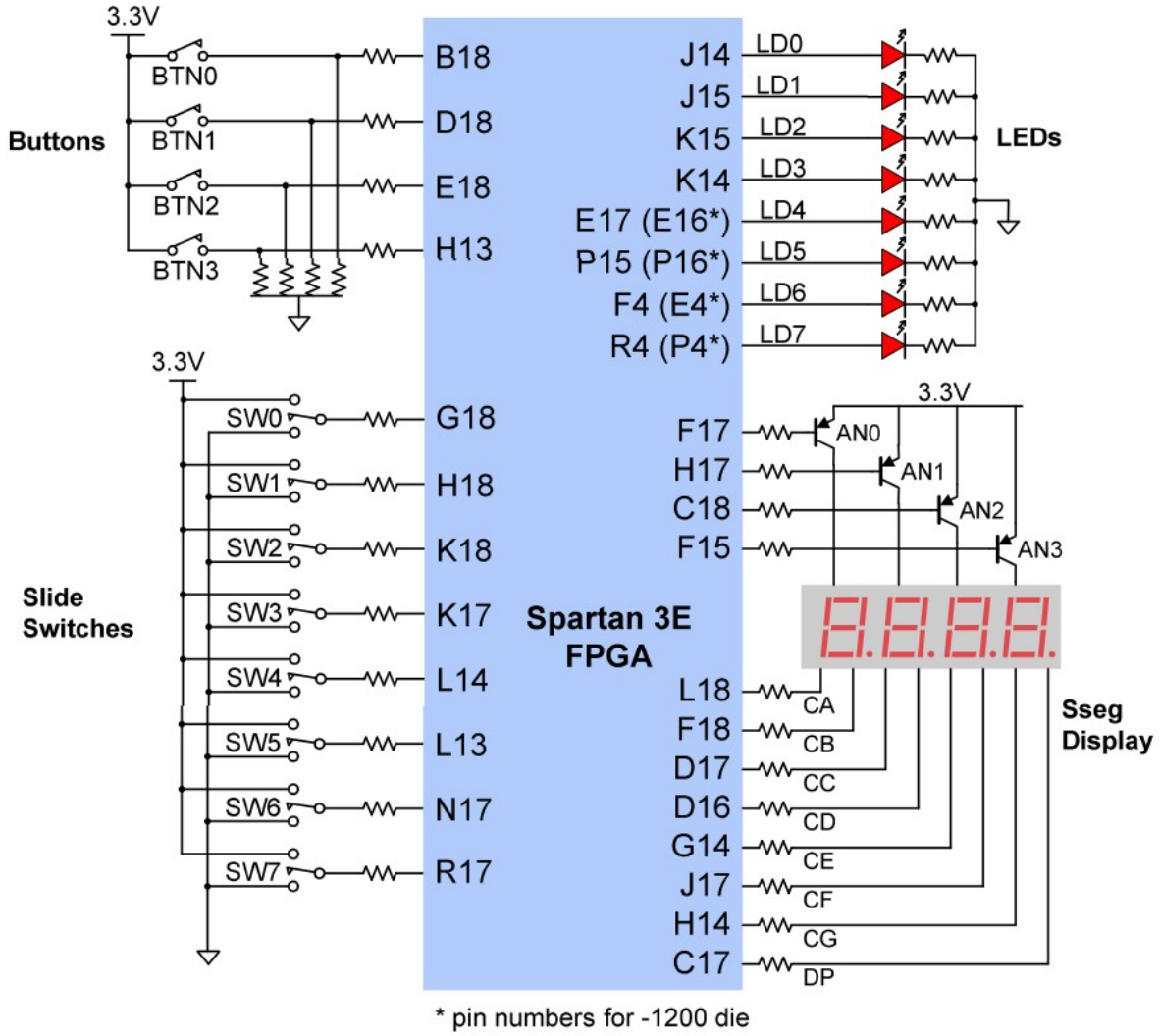
22. ábra. Kapcsolók, LED-ek, nyomógombok, hét szegmenses kijelzők a Nexsys2 kártyán.



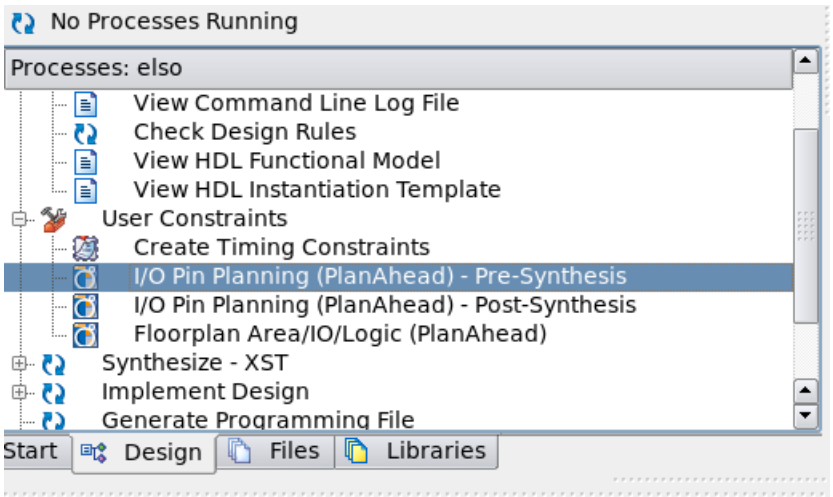
23. ábra. 50MHz-es oszcillátor csatlakozása a Spartan–3E kártyához (B8-as láb)

Lábkiosztás grafikus szerkesztővel

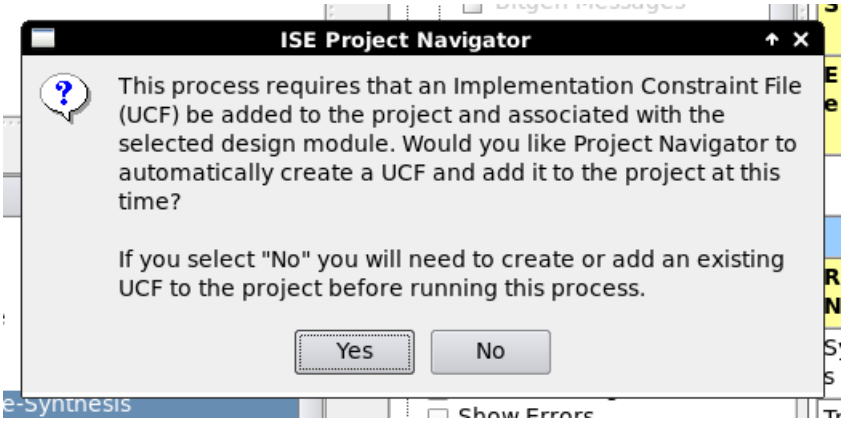
Adjunk a projektünkhöz egy paraméterező állományt (Implementation Constraints File-t)! Ebben a állományban fogjuk a lábkiosztást definiálni! A következő lépéseket kell megvalósítanunk:
A forrás ablakban kiválasztjuk a proba.sch állományt. A Processes ablakban válasszuk ki az **IO Planing (PlanAhead) – Pre-Synthesis** parancsot (25. ábra).
A parancs elindítása után a rendszerek figyelmeztet minket, hogy egy új „Implementation Constraints File” fog hozzáadni! Ezt engedélyeznünk kell (26. ábra)!



24. ábra. Kapcsolók, LED-ek, nyomógombok, hét szegmenses kijelzők kapcsolódása a Spartan-3E FPGA áramkörhöz a Nexsys2 kártyán.



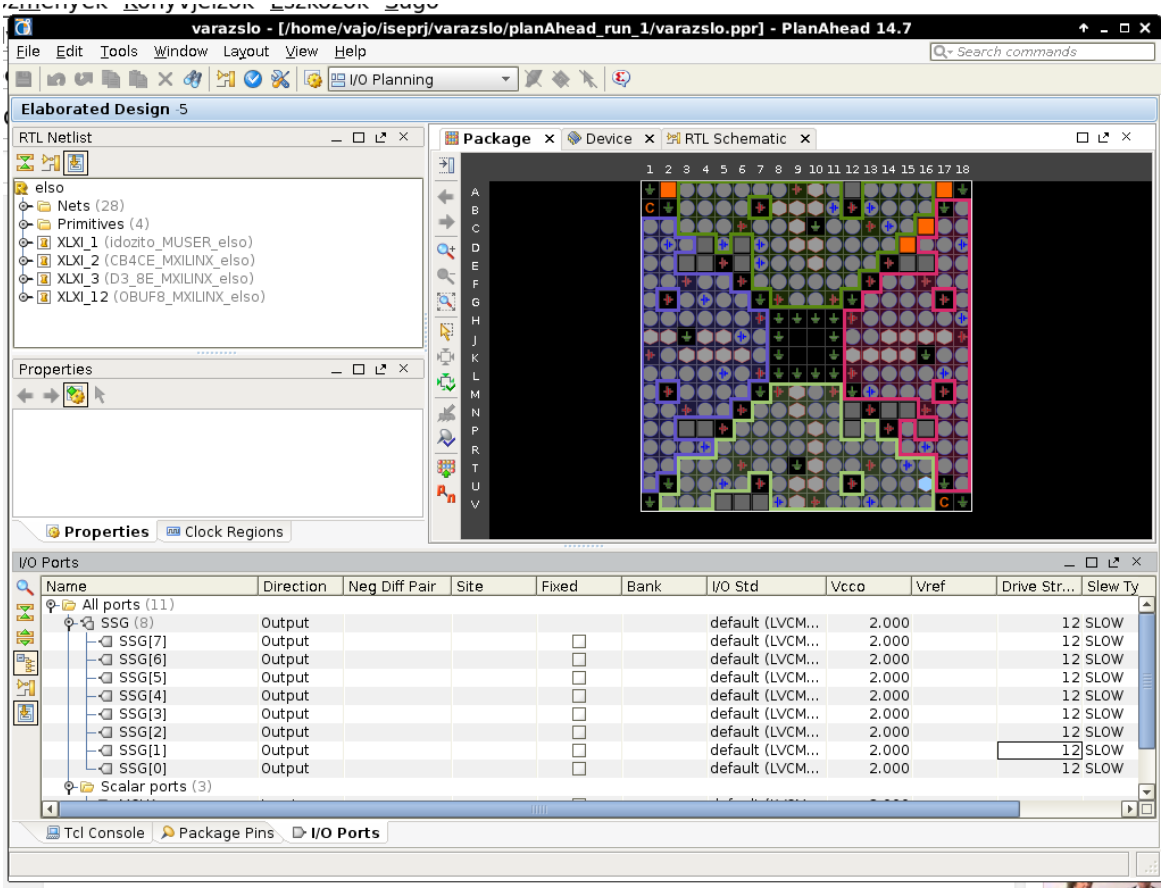
25. ábra Szintézis előtti lábkiosztás megvalósítása I/O Pin Planning paranccsal.



26. ábra Paraméterező állomány hozzáadásának engedélyezése

Lábkiosztás grafikus felületen

A grafikus felület megjelenése előtt a fordító elvégzi a felépített áramkör szintaktikai ellenőrzését, ezek után az 27. ábrán lévő felület jelenik meg.



27. ábra Plan Ahead kép

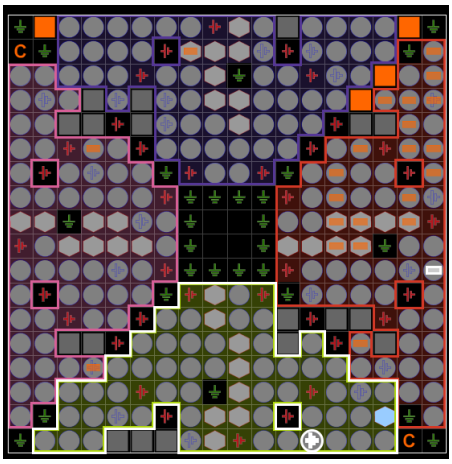
A 27. ábra I/O Ports ablakban beírhatjuk a ***„Site”*** oszlopba a lábakat. Másik módszer az I/O Ports ablakból „húzd és vidd” módszerrel vigyük át megfelelő helyre minden markert jobb oldalon található ***„Package”*** grafikus IC tokozás ábrájára!

I/O Ports

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std
ssg[2]	Output		D17	☑	1	LVTTL*
ssg[1]	Output		F18	☑	1	LVTTL*
ssg[0]	Output		L18	☑	1	LVTTL*
Scalar ports (8)						
AN0	Output		F17	☑	1	LVTTL*
AN1	Output		H17	☑	1	LVTTL*
AN2	Output		C18	☑	1	LVTTL*
AN3	Output		F15	☑	1	LVTTL*
BTN0	Input		B18	☑	1	LVTTL*
BTN1	Input		D18	☑	1	LVTTL*
MCLK	Input		B8	☑	0	LVTTL*
SW0	Input		G18	☑	1	LVTTL*

Tcl Console | Package Pins | I/O Ports

28. ábra Lábak elhelyezése.

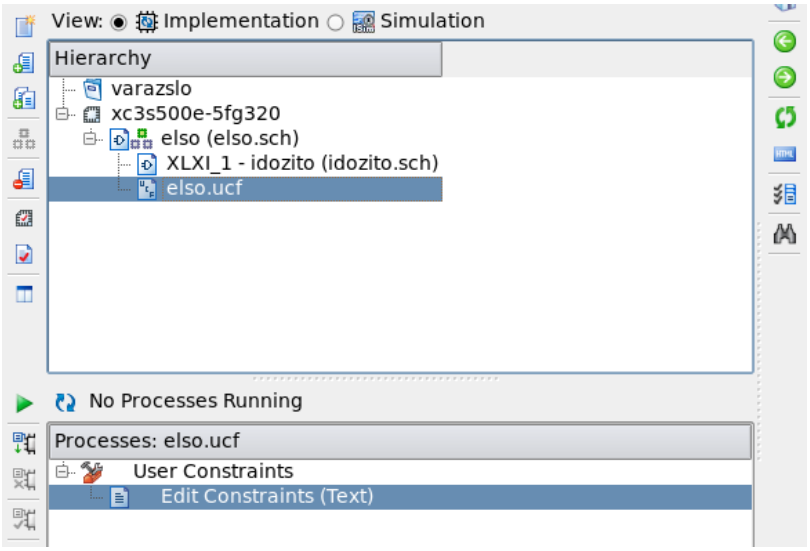


29. ábra Lábak elhelyezése az IC felülnézeti képén (sárga téglalappal megjelölt lábak).

Miután az összes markert elhelyeztük, mentsük el a fájlt, majd zárjuk be a Plan Ahead felületet!

Lábkiosztás szöveges módszerrel

A lábkiosztás elvégzésére lehetőséget nyújt egy szöveg szerkesztő (text editor) is, melyet az **Edit** Constraints paranccsal indíthatunk el (kettőt kattintva) – lásd



30. ábra Lábkiosztás szerkesztése (Edit Constraints)

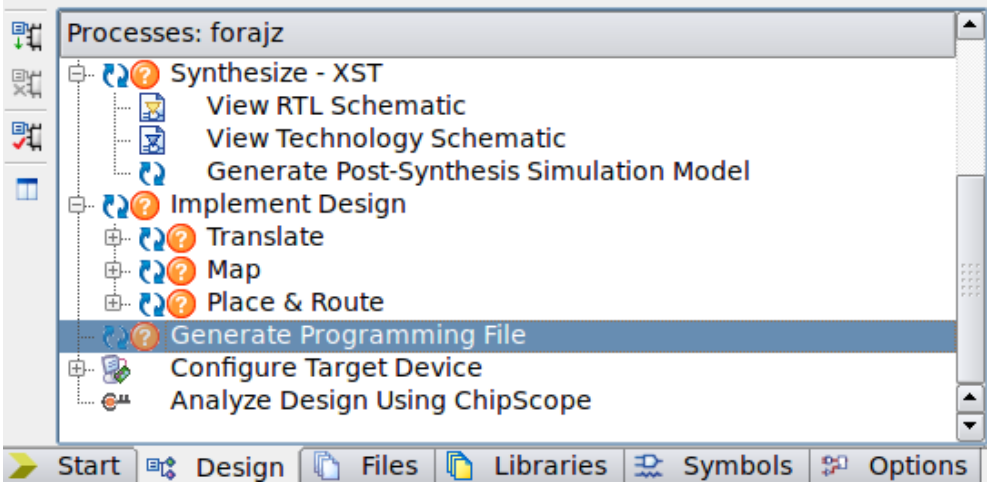
```
1  # lábkiosztás; technologia illesztés
2  NET "MCLK" LOC = B8;
3  NET "MCLK" IOSTANDARD = LVTTL;
4  NET "RESET" LOC = B18;
5  NET "RESET" IOSTANDARD = LVTTL;
6  NET "SW0" LOC = G18;
7  NET "SW0" IOSTANDARD = LVTTL;
8  NET "LED[7]" IOSTANDARD = LVTTL;
9  NET "LED[6]" IOSTANDARD = LVTTL;
10 NET "LED[5]" IOSTANDARD = LVTTL;
11 NET "LED[4]" IOSTANDARD = LVTTL;
12 NET "LED[3]" IOSTANDARD = LVTTL;
13 NET "LED[2]" IOSTANDARD = LVTTL;
14 NET "LED[1]" IOSTANDARD = LVTTL;
15 NET "LED[0]" IOSTANDARD = LVTTL;
16 # PlanAhead Generated physical constraints
17 NET "LED[7]" LOC = R4;
18 NET "LED[6]" LOC = F4;
19 NET "LED[5]" LOC = P15;
20 NET "LED[4]" LOC = E17;
21 NET "LED[3]" LOC = K14;
22 NET "LED[2]" LOC = K15;
23 NET "LED[1]" LOC = J15;
24 NET "LED[0]" LOC = J14;
25
```

31. ábra A lábkiosztás szöveges formája

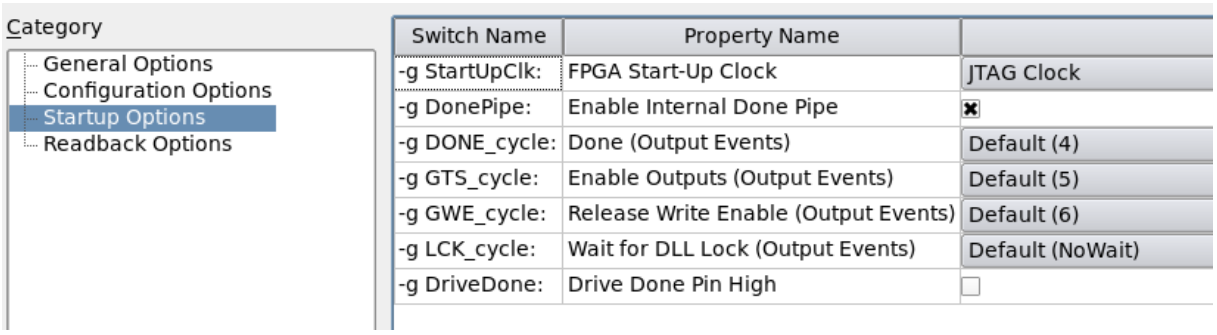
A #-al kezdődő sorokat a fordító nem veszi figyelembe – ezek megjegyzések, így lehetőségünk van univerzális lábkiosztást írni, melyben csak az adott programban használt perifériákat használjuk, a többit megjegyzésként kezeljük („kommentezzük”)!
Mentsük el a szerkesztett állományt.

Terv fordítás

A bittérkép generálásakor (**Generate Programming File**) be kell állítani, hogy az FPGA áramkör működése a konfigurációs órajele alapján induljon (***FPGA Start-Up Clock – 33. ábra***).
A beállításhoz kattintsunk jobb egérgombbal a **Generate Programming File**-ra (konfigurációs állomány létrehozása), majd válasszuk ki a ***Properties*** (tulajdonságok) – lásd 32. ábra.

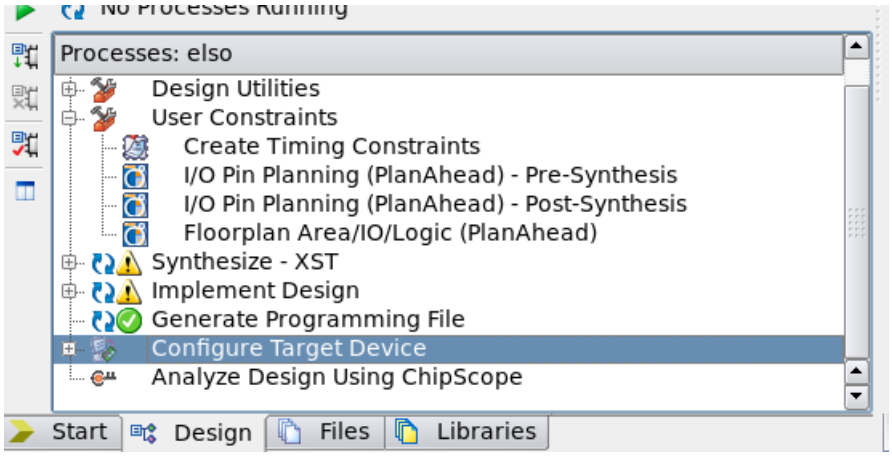


32. ábra FPGA bittérkép generálás



33. ábra FPGA élesítés szinkronizálás JTAG clock (órajel)

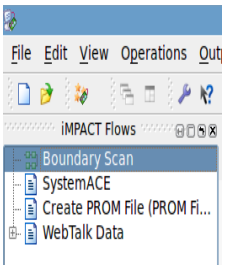
Majd nyomjuk meg az **OK** gombot! Kattintsunk duplán a **Generate Programming File** parancsra.



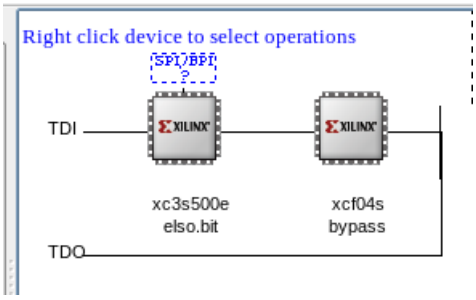
34. ábra FPGA konfiguráció (letöltés) indítása.

PC csatlakoztatása a panelhez

A bittérkép letöltéséhez kattintsunk a **Configure Device (iMPACT)**-re (34. ábra)! A program figyelmeztet, hogy egy új konfigurációs tervet fog hozzáadni a tervünkhöz. Az ablakban kattintsunk a **OK** gombra. Ha a konfiguráló program (iMPACT) elindul, akkor a 35. ábra szerinti parancsot válaszuk ki.



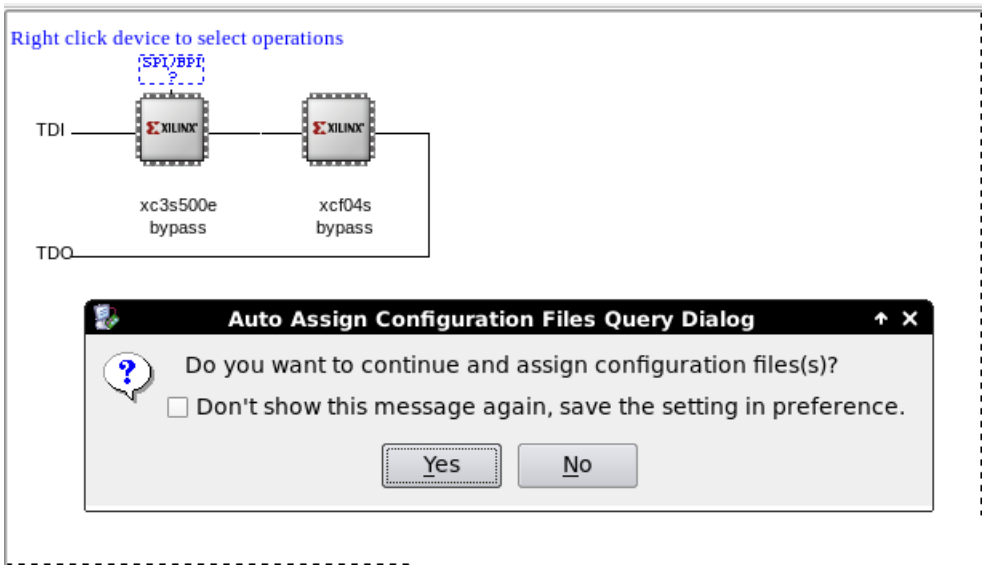
35. ábra Boundary Scan – JTAG konfiguráció kiválasztása.



36. ábra „JTAG kapcsolat a kártyával” ábrája.

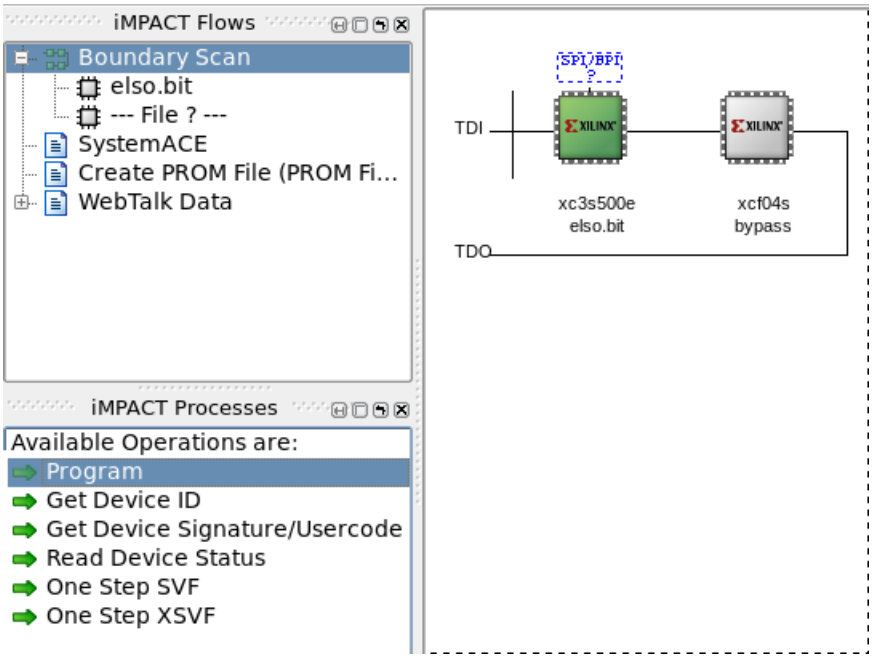
Csatlakoztatás Boundary Scan módszerrel.

A Boundary-Scan (JTAG) parancs elindulásakor automatikusan megnyílik a következő ablak (ha mégse, akkor jobb egérgombbal kattintsunk az IC-re, majd válasszuk az **Assign New Configuration File** parancsot):



37. ábra Program kiválasztása a konfiguráláshoz

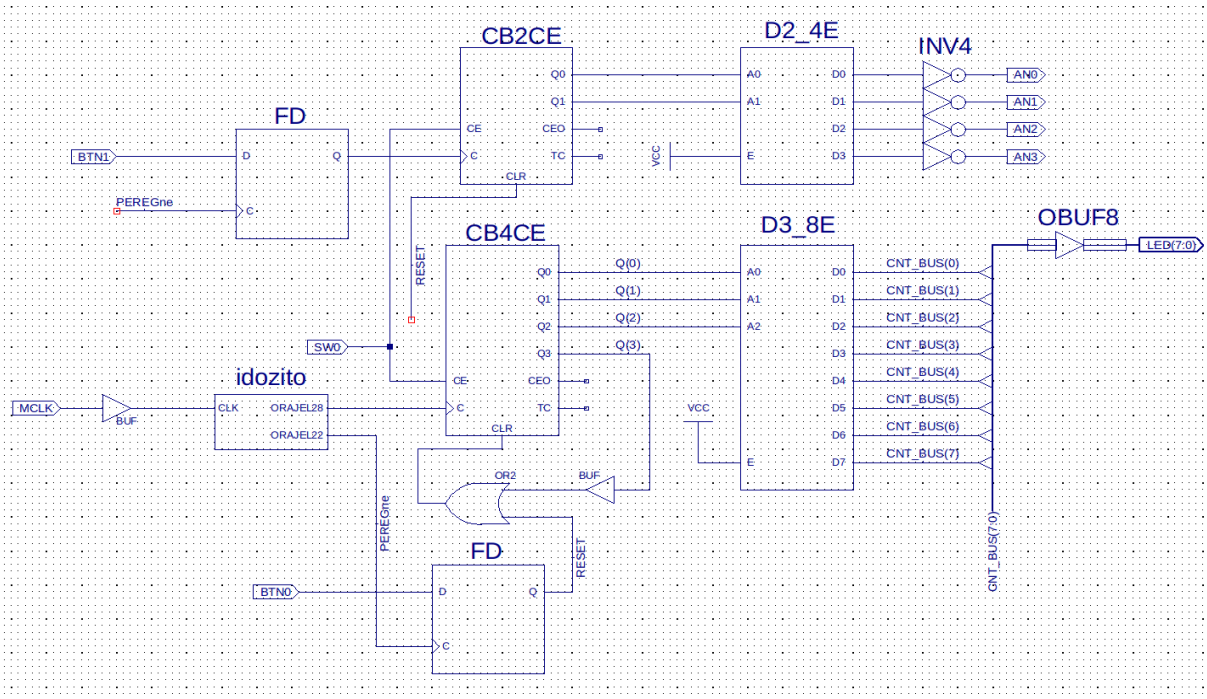
Válasszuk ki az állományt, majd **Open**!
Mivel a FLASH memóriát nem akarjuk konfigurálni ezért itt a válasszuk a BYPASS **utasítást!** (lásd 38. ábra. A bal oldali IC-n kattintva töltsük programunkat az FPGA áramkörbe! Amennyiben változik a kapcsolási rajz vagy a lábkiosztás mindig mentsük el és generáljuk újra a programozó állományt (.bit)!



38. ábra Program letöltése az FPGA áramkörbe.

1. Feladat

Tervezzük át a programot úgy, hogy a BTN3 nyomógomb hatására a program reszeteljen!
A nyomógombot pergésmentesítse; a lábkiosztást is változtatni kell!
A feladat egy lehetséges megoldása a 39. ábrán látható.
Alaphelyzetben minden kijelző be van kapcsolva. Bővítse úgy a kapcsolási rajzot, hogy egyszerre mindig csak egy kijelző legyen bekapcsolva. A kiválasztást a BTN2 nyomógombbal lehessen megtenni (a lábkiosztást is változtatni kell mivel az hétszegmenses kijelző anódjait is vezérelni kell (AN0-3)! A megoldás 39. ábrán látható.



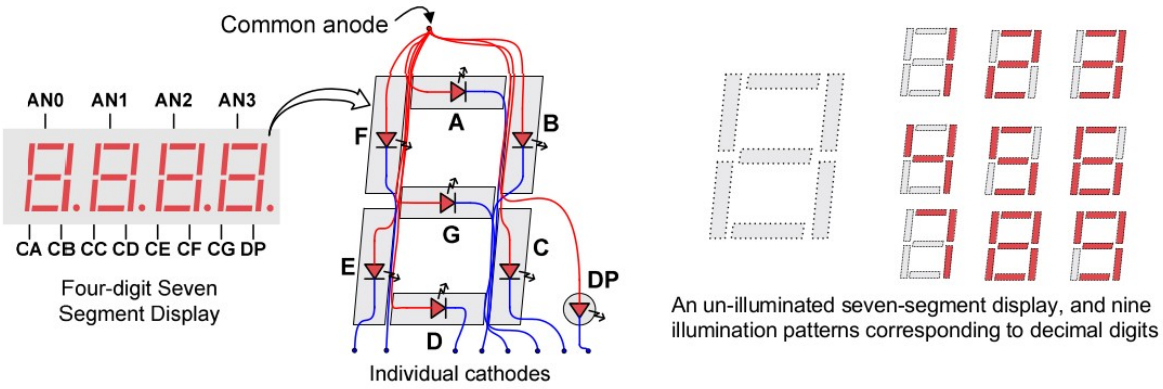
39. ábra Az első feladat egy lehetséges megoldása

2. Feladat:

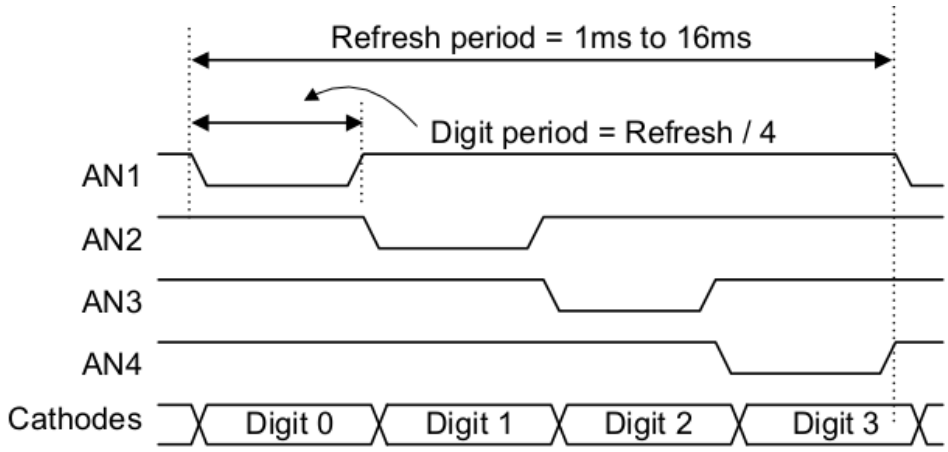
Írja át a programot, hogy a fény oda-vissza fusson, és az irányt az SW2 kapcsolóval lehessen beállítani! Fusson oda-vissza a fény, de azok a LED-ek ne világítsanak, melyekhez rendelt SW kapcsoló aktív (pl.: SW3 és SW4 aktív, akkor LD3 és LD4 nem világít) Az előző két programot egyesítse, úgy, hogy a BTN2 nyomógombbal lehessen váltani közöttük!
Valósítson meg „Knight-Rider” futófényt a LED-eken!

Szimbólum rendelése VERILOG forráshoz

Készítsünk a 7-szegmens kijelzőkön egyszerű számláló programot!
Az előző ismereteinket bővítsük egy kis VERILOG programozással! Írjunk egy algoritmust, mely a bemenetére érkező 4 bites bináris számból előállítja annak 7-szegmenses kódját! 7-szegmens kódtáblázat közös anódos kijelzőre. A kártyán található hétszegmenses kijelző 4 digites, az egyes szegmensek pedig közös anóddal rendelkeznek (lásd 40. ábra). Az egyes digiteket multiplexelve kell vezérelnünk ahhoz, hogy egy-egy digitre a megfelelő hexadecimális számjegyet (0-F) ábrázoljuk. A vezérlési diagram a 41. ábrán látható.



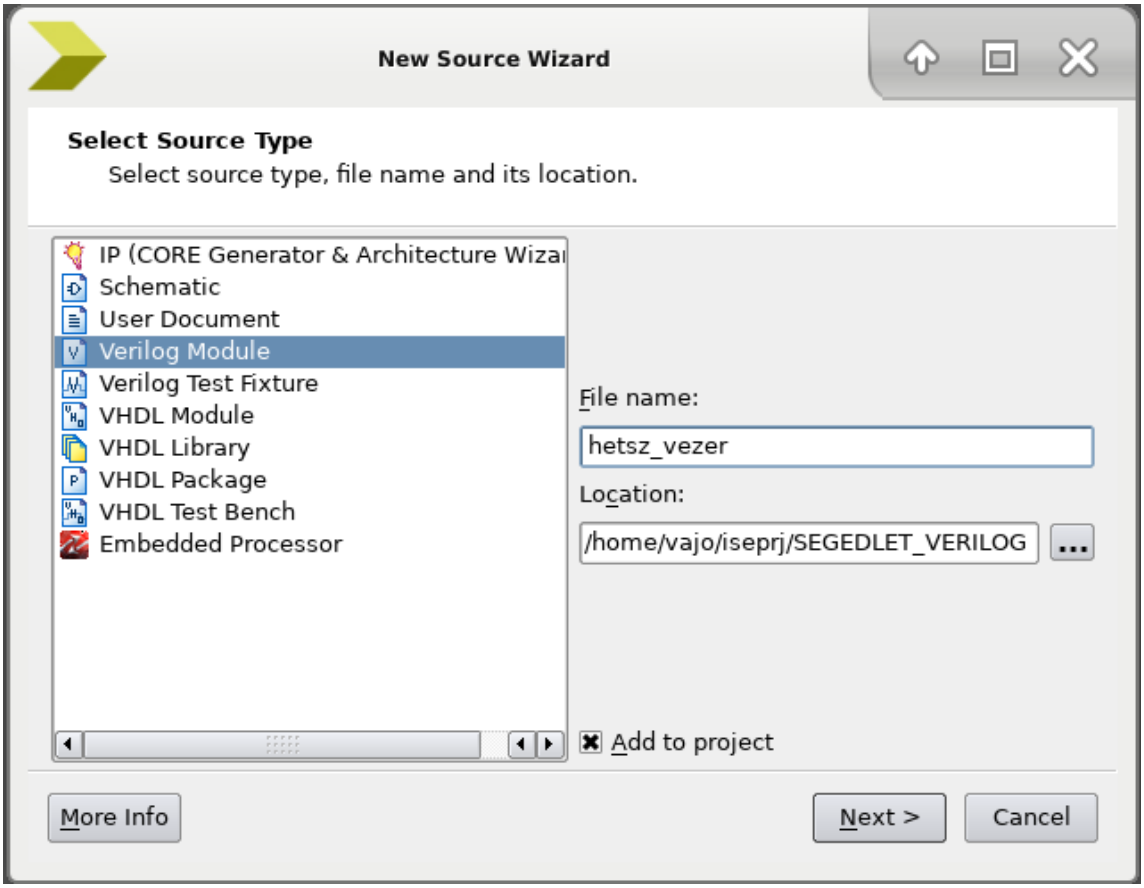
40. ábra Hétszegmenses kijelző felépítésének ábrázolása.



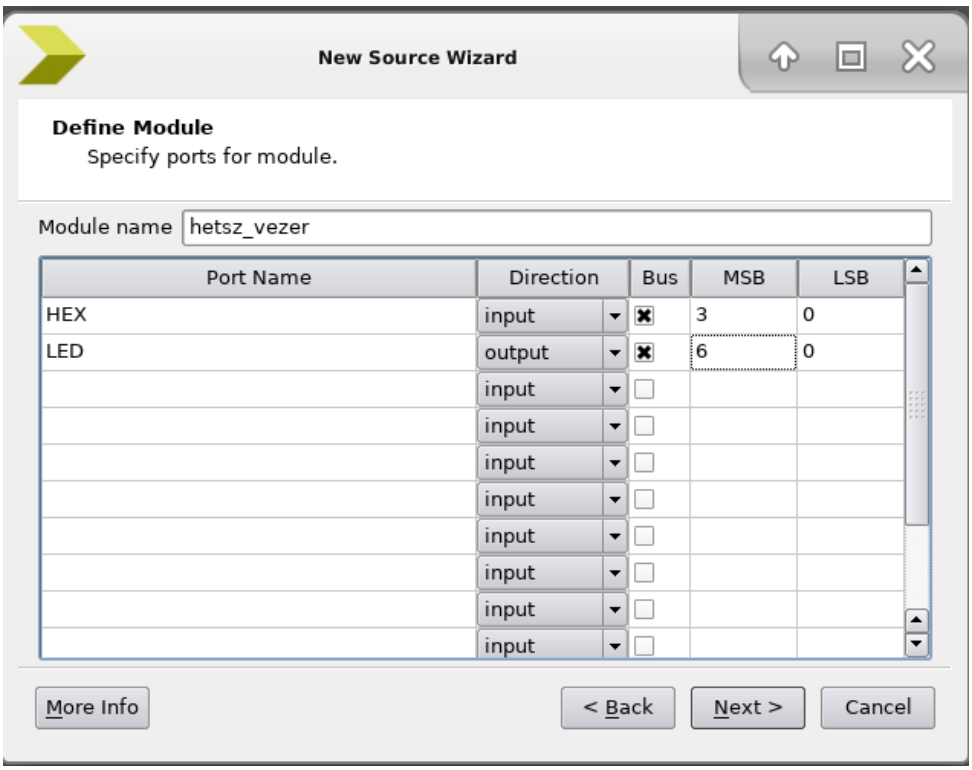
41. ábra A hétszegmenses kijelző vezérlésének idő diagramja.

HDL modul létrehozása

Új forrásként adjunk a projektünkhöz egy VERILOG típusú állományt a File -> New **Source** paranccsal. Az állomány típusa legyen ***„VERILOG Module”*** (42. ábra). Adjuk meg a VERILOG forrás be- és kimeneteit (sín - bus), valamint a sínrendszerek szélességét (HEX[3:0]; LED[6:0]) – lásd 43. ábra.



42. ábra VERILOG állomány hozzáadása a tervhez.



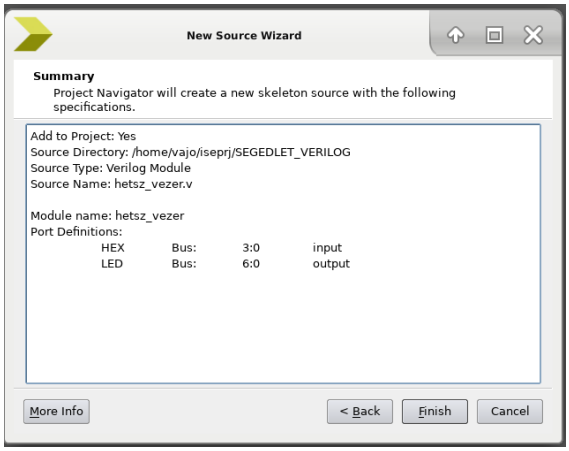
43. ábra VERILOG modul bemeneti és kimeneti változóinak meghatározása.

VERILOG modul inicializálása

Bemenet: bináris kód (4 bites).
Kimenet: 7-szegmens vezérlőkód (7 bites, jelenleg a tizedespontot nem használjuk)
A következő ábrán (45. ábra) egy összegzés látható a beállításainkról, ennek alapján a program elkészítette a VERILOG modul inicializálását. Nekünk már csak a funkcionális programot kell megírnunk (44. ábra). A VERILOG kódot a modul meghatározása után

module(…);
// függvények
endmodule

közé írjuk be. A megjegyzéseket „//” karakterek után írjuk. Ezt a szöveget a fordító nem veszi figyelembe (kommentárnak tekinti)!



44. ábra A fejlesztőkörnyezet által generált VERILOG module összefoglalásának ábrája.

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    18:11:46 01/13/2020
7  // Design Name:
8  // Module Name:    hetsz_vezer
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21 module hetsz_vezer(
22     input [3:0] HEX,
23     output [6:0] LED
24     );
25
26
27 endmodule
28
```

45. ábra A fejlesztőkörnyezet által generált VERILOG kód.

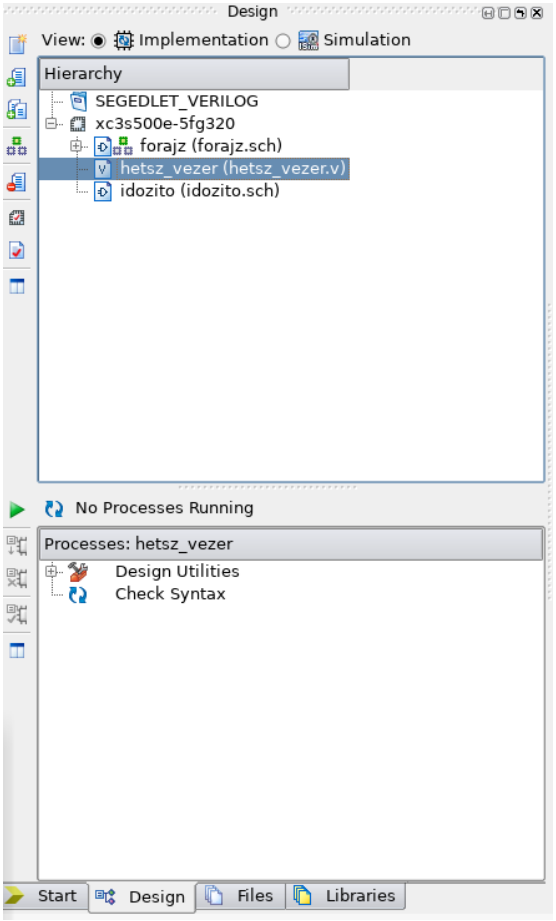
A hétszegmenses kijelző vezérlő VERILOG programja a következő:

```
21
22 module hetsz_vezer(
23     input [3:0] HEX,
24     output [6:0] LED
25     );
26 always @(HEX)
27       case (LED)
28           4'b0001 : LED = 7'b1111001;   // 1
29           4'b0010 : LED = 7'b0100100;   // 2
30           4'b0011 : LED = 7'b0110000;   // 3
31           4'b0100 : LED = 7'b0011001;   // 4
32           4'b0101 : LED = 7'b0010010;   // 5
33           4'b0110 : LED = 7'b0000010;   // 6
34           4'b0111 : LED = 7'b1111000;   // 7
35           4'b1000 : LED = 7'b0000000;   // 8
36           4'b1001 : LED = 7'b0010000;   // 9
37           4'b1010 : LED = 7'b0001000;   // A
38           4'b1011 : LED = 7'b0000011;   // b
39           4'b1100 : LED = 7'b1000110;   // C
40           4'b1101 : LED = 7'b0100001;   // d
41           4'b1110 : LED = 7'b0000110;   // E
42           4'b1111 : LED = 7'b0001110;   // F
43           default : LED = 7'b1000000;   // 0
44       endcase
45
46 endmodule
```

46. ábra Az általunk írt VERILOG kód

A fenti kódsorozat egy switch-case szerkezet (bal oldalt a vezérlés – változók értéke, jobb oldalt az erre adott válasz látható).

Készítsünk sematikus szimbólumot VERILOG forrásunkból. Először ellenőrizzük az általunk létrehozott program helyességét (47. ábra) a ***Check Syntax*** parancs futtatásával. Amennyiben a leírás szerint haladtunk, az ellenőrzés szintaktikai hibát ad (lásd 48. ábra). A hibaüzenet tartalmára kattintva a rendszer megmutatja a hiba helyét.
FIGYELEM! Ha a LED-et bemenetként (IN) határoztuk meg a rendszer nem jelez hibát, viszont majd a fordítás során hibakódot ad. A VERILOG a bemeneti jeleket csak olvasni tudja, viszont a LED az kimenet (OUTPUT) – A VERILOG a kimenetet csak írni tudja. Javítsuk ki a port () meghatározásnál a LED típusát OUTPUT típusra, mentsük el és még egyszer ellenőrizzük a program helyességét. Most már az ellenőrzés szintaktikailag (VERILOG nyelvtan szempontjából) hibátlan programot jelent.



47. ábra VERILOG program ellenőrzése – „Check Syntax”

```
=========================================================================
*                          HDL Compilation                              *
=========================================================================
Compiling verilog file "hetsz_vezer.v" in library work
Module <hetsz_vezer> compiled
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 27 Reference to vector wire 'LED' is not a legal reg or variable lvalue
ERROR:HDLCompilers:44 - "hetsz_vezer.v" line 27 Illegal left hand side of blocking assignment
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 28 Reference to vector wire 'LED' is not a legal reg or variable lvalue
ERROR:HDLCompilers:44 - "hetsz_vezer.v" line 28 Illegal left hand side of blocking assignment
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 29 Reference to vector wire 'LED' is not a legal reg or variable lvalue
ERROR:HDLCompilers:44 - "hetsz_vezer.v" line 29 Illegal left hand side of blocking assignment
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 30 Reference to vector wire 'LED' is not a legal reg or variable lvalue
ERROR:HDLCompilers:44 - "hetsz_vezer.v" line 30 Illegal left hand side of blocking assignment
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 31 Reference to vector wire 'LED' is not a legal reg or variable lvalue
ERROR:HDLCompilers:44 - "hetsz_vezer.v" line 31 Illegal left hand side of blocking assignment
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 32 Reference to vector wire 'LED' is not a legal reg or variable lvalue
ERROR:HDLCompilers:44 - "hetsz_vezer.v" line 32 Illegal left hand side of blocking assignment
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 33 Reference to vector wire 'LED' is not a legal reg or variable lvalue
ERROR:HDLCompilers:44 - "hetsz_vezer.v" line 33 Illegal left hand side of blocking assignment
ERROR:HDLCompilers:247 - "hetsz_vezer.v" line 34 Reference to vector wire 'LED' is not a legal reg or variable lvalue
```

48. ábra Ha hibás az ellenőrzés eredménye példa.

Amennyiben ellenőrzéskor a 48. ábra szerinti hibákat kapjuk, abban az esetben ki kell javítanunk az LED kimenetek típusát: reg – regiszter típusra (lásd 49. ábra).

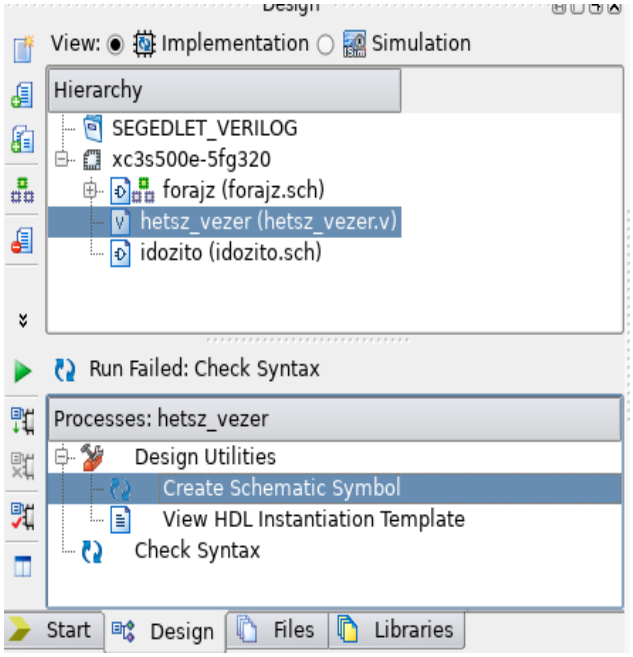
```
22 module hetsz_vezer(
23     input [3:0] HEX,
24     output reg [6:0] LED
25     );
```

49. ábra LED típusa REGiszter szemléltetése.

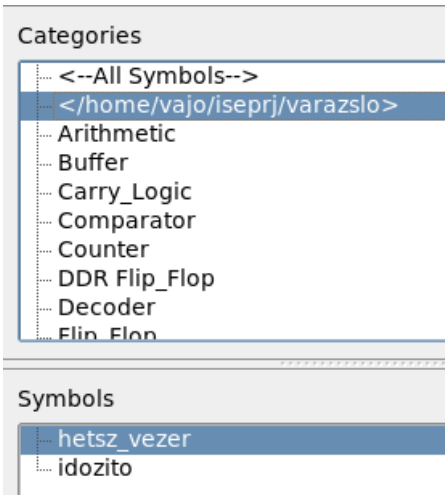
Rajz szimbólum létrehozása VERILOG forrásból

A helyes VERILOG modulból kapcsolási rajz szimbólumot tudunk létrehozni a „Create Schematic Symbol” paranccsal – ahogyan azt a 50. ábra mutatja.

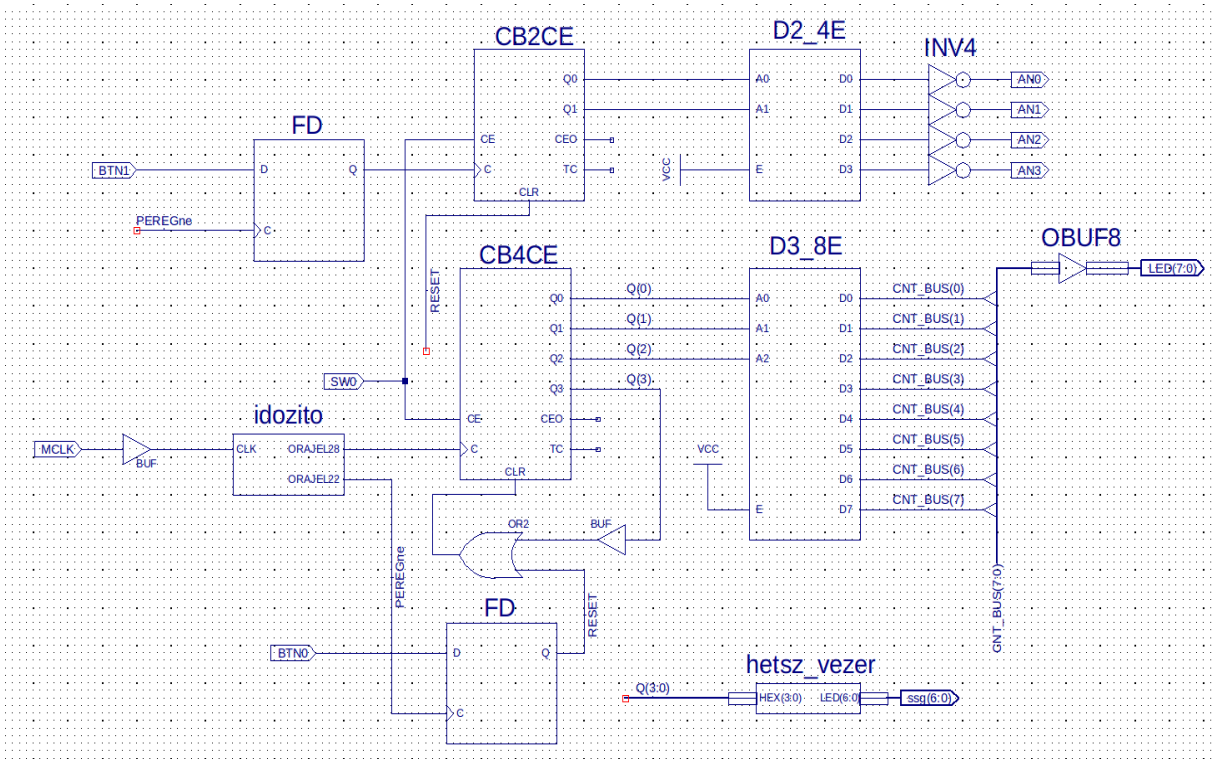
Ellenőrizzük, hogy alkatrészünk megjelent-e a szimbólum könyvtárban a saját szimbólumok között (51. ábra). Az elkészített hétszegmenses vezérlő alkatrész felhasználásával feladatunk egy lehetséges megvalósítása a az 52. ábrán látható.



50. ábra Rajz szimbólum létrehozása VERILOG forrásból



51. ábra VERILOG forrásból elkészített alkatrész a projekt szimbólumai között.



52. ábra A feladat megoldása.

```
16  # NET SSG
17  NET "ssg[0]" LOC = L18 | IOSTANDARD = LVTTL;
18  NET "ssg[1]" LOC = F18 | IOSTANDARD = LVTTL;
19  NET "ssg[2]" LOC = D17 | IOSTANDARD = LVTTL;
20  NET "ssg[3]" LOC = D16 | IOSTANDARD = LVTTL;
21  NET "ssg[4]" LOC = G14 | IOSTANDARD = LVTTL;
22  NET "ssg[5]" LOC = J17 | IOSTANDARD = LVTTL;
23  NET "ssg[6]" LOC = H14 | IOSTANDARD = LVTTL;
24  #NET "ssg[7]" LOC = C17 | IOSTANDARD = LVTTL;
25  # NET AN
26  NET "AN0" LOC = F17 | IOSTANDARD = LVTTL;
27  NET "AN1" LOC = H17 | IOSTANDARD = LVTTL;
28  NET "AN2" LOC = C18 | IOSTANDARD = LVTTL;
29  NET "AN3" LOC = F15 | IOSTANDARD = LVTTL;
30  # NET btn
```

53. ábra A hétszegmenses kijelző lábkiosztásának meghatározása.

Módosítsuk vagy egészítsük ki az elso.ucf állományban a lábkiosztást a hétszegmenses kijelző szegmenseivel és anódjaival (53. ábra)

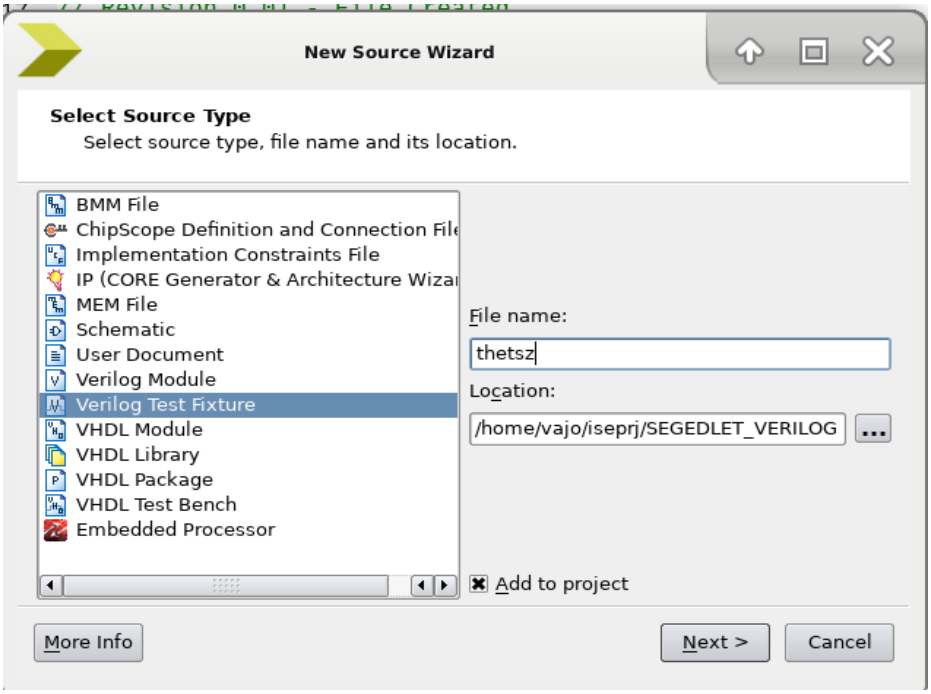
Megjegyzés: Ellenőrizzük az .ucf állományban, hogy a többi be/ki-menetünk is a dokumentáció szerinti (lásd [1.]).

A hétszegmenses kijelző vezérlő ellenellenőrzése szimulációval

Kapcsolási rajzaink, VERILOG programjaink funkcionálisan helyes működését szimulációval is tudjuk ellenőrizni. Az alábbiakban a hétszegmenses kijelző vezérlő működését ellenőrizzük szimuláció segítségével.

Tesztvektor állomány hozzáadása a tervhez

Ehhez egy új állományt kell hozzáadnunk a tervünkhöz (**File → New Source**), amelynek típusa VERILOG Test Bench (tesztelő állomány) lesz. Nevezzük el thetsz, hogy a névadással is kössük a hetsz_vezer.v állományhoz (54. ábra). A következő lépéshez kattintsunk a **Next**-re
Válasszuk ki a tesztelendő állományt, a mi esetünkben a hetsz_vezer (55. ábra), majd **Next**.



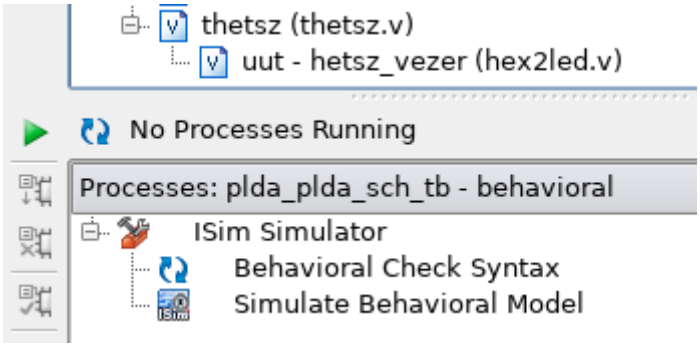
54. ábra Tesztvektorok állomány kiválasztása.

55. ábra Tesztelendő állomány kiválasztása.

A thetsz.v állományt ki kell egészítenünk a gerjesztésekkel/tesztvektorokkal (56. ábra). Ehhez az egyes jeleket súlyozásuk szerint (2,4,8,16) tartjuk egy-egy félperiódusig 0 majd 1 állapotban. Mivel ez a jelváltozás megegyezik egy 4 bites számláló kimenő jeleivel ezért a tesztvektorokat számlálóval állítjuk elő. Ezáltal létrehozunk megfelelő periódusú jeleket és így a rendszer előállítja számunkra az összes bemeneti kombinációt, amelyet a HEX jel felvehet (0000 … 1111).

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////

module thetsz;

   // Inputs
   reg [3:0] HEX=0;
   wire [6:0] LED;

   // Instantiate the Unit Under Test (UUT)
   hetsz_vezer uut (
      .HEX(HEX),
      .LED(LED)
   );
   always #1 HEX=HEX+1;


   initial begin
      // Initialize Inputs

      // Wait 100 ns for global reset to finish
      #16 $finish;

      // Add stimulus here

   end

endmodule
```

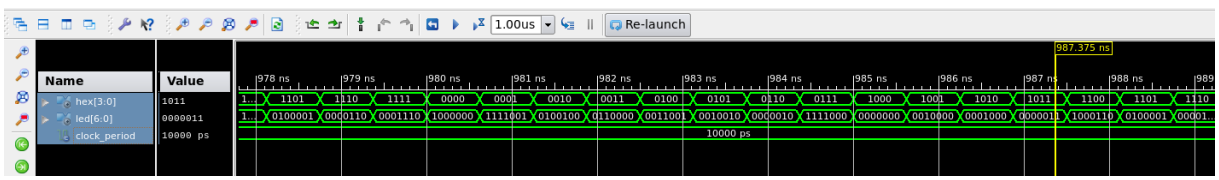
56. ábra Tesztvektorok a hétszegmenses kijelzőhöz.



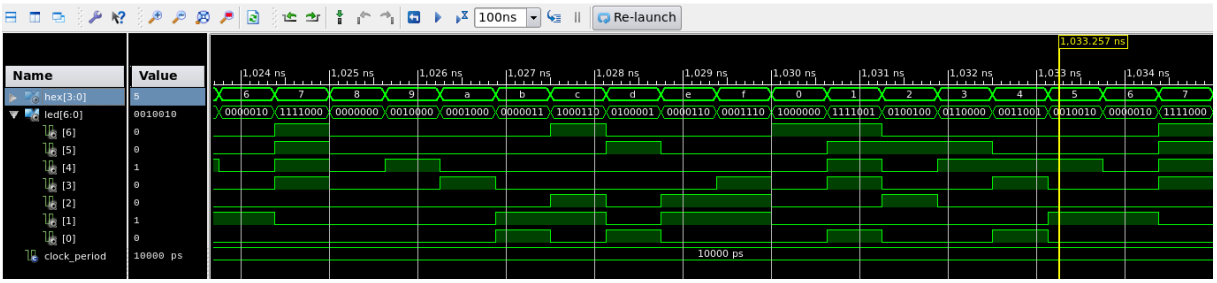
57. ábra Szimulátor elindítása az ISE-ben

Szimuláció

A szimulátort az 57. ábra szerinti paranccsal indítható el. A szimuláció az áramkör ideális viselkedését (funkcionalitását) fogja vizsgálni.
Az 58. ábrán a szimuláció eredménye látható, közvetlenül a szimulátor élindítása után. Az 1.00 us szimulációs időt állítsuk 100 ns-ra, ahogyan az az 58. ábrán látható. Kattintsunk a LED[6:0] névre az idődiagramon, hogy láthatóvá tegyük a szegmensek időbeni változását. A bemeneti értékek alapján vizsgáljuk a kimeneti LED-ek értékét. Megállapítható, hogy a hétszegmenses kijelző vezérlő áramköre helyesen működik.



58. ábra Szimulátor kép bejelentkezés után. Grafikus kép.



59. ábra Hétszegemses kijelző szimulációs eredmények. Grafikus kép.

Egyéb Feladatok

Oldja meg, hogy egy kapcsolóval (SW0) lehessen kiválasztani, hogy felfelé, vagy lefelé számlálás történjen!
Egy másik kapcsolóval (SW1) át lehessen állítani a számlálást decimálisra!
SW2 aktív esetén a számlálás „helyiérték-helyes” legyen, vagyis készítsen valódi számlálót a hétszegmenses kijelzőre (a tesztelés rövidítése érdekében növelje meg az órajelet)!.

Melléklet

Az általunk használt gyakorlópanel lábkiosztása a következő:

Periféria	**Lábszám**	**Megjegyzés**
CLK	B8	Órajel
BTN0	B18	Nyomógomb
BTN1	D18	Nyomógomb
BTN2	E18	Nyomógomb
BTN3	H13	Nyomógomb
SW0	G18	Kapcsoló
SW1	H18	Kapcsoló
SW2	K18	Kapcsoló
SW3	K17	Kapcsoló
SW4	L14	Kapcsoló
SW5	L13	Kapcsoló
SW6	N17	Kapcsoló
SW7	R17	Kapcsoló
LED0	J14	LED
LED1	J15	LED
LED2	K15	LED
LED3	K14	LED
LED4	E17	LED
LED5	P15	LED
LED6	F4	LED
LED7	R4	LED
DISP_LED0	L18	a
DISP_LED1	F18	b
DISP_LED2	D17	c
DISP_LED3	D16	d
DISP_LED4	G14	e
DISP_LED5	J17	f
DISP_LED6	H14	g
DISP_LED7	C17	dp
AN0	F17	DISP0 ENABLE
AN1	H17	DISP1 ENABLE
AN2	C18	DISP2 ENABLE
AN3	F15	DISP3 ENABLE

Felhasznált irodalom

[1.] ***: *Nexsys2 dokumentáció*, 2011, pp. 17. http://mazsola.iit.uni-miskolc.hu/d/nexys2

[2.] Vásárhelyi J.: Hardver leíró nyelvek: *VHDL segédlet, http://mazsola.iit.uni-miskolc.hu/d/vhdl, 2013, pp. 54.*

[3.] *Vásárhely J.: Tesztvektorok négyváltozós kombinációs hálózatok szimulációjához***,** 2013. VHDL állomány.