

9. Digitális rendszertervezés - Szintézis FPGA technológiára

Szerző: Horváth Péter, 2016. Verilog HDL-re átdolgozta: Ress Sándor, 2019.

Utolsó módosítás: 2023. 05. 11.

A *Szintézis FPGA technológiára* c. laboratóriumi gyakorlat során az *Altera Quartus II* fejlesztői környezetét és a *DE0* fejlesztőkártyát használjuk. A szintézis lépéseit az *RTL modellezés és szimuláció* c. laboratóriumi gyakorlat elméleti összefoglalójában bemutatott fel/le számláló egység módosított változatának szintézisén keresztül vizsgáljuk. A 9-1. ábra összefoglalja a számláló Verilog modelljén elvégzett módosításokat.

```
2
3 module counter(
4     input          clk,
5     input          reset_n,
6     input          enable,
7     input          direction,
8     input[7:0]     parallel_in,
9     input          load_n,
10    output reg[7:0] cout
11);
12 reg[25:0] delay_counter;
13 reg enable_increment;
14
15 always @(posedge clk)
16 begin: PRE_SCALER
17     if (delay_counter == 50000000)
18     begin
19         delay_counter <= 0;
20         enable_increment <= 1;
21     end
22     else
23     begin
24         delay_counter <= delay_counter + 1'b1;
25         enable_increment <= 0;
26     end
27 end
28
29 always @(posedge clk or negedge reset_n)
30 begin: COUNTER
31     if (reset_n == 0)
32         cout <= 8'b0;
33     else
34         if (enable)
35         begin
36             if (load_n == 0)
37                 cout <= parallel_in;
38             else if (enable_increment)
39             begin
40                 if (direction == 1)
41                     cout <= cout + 1'b1;
42                 else
43                     cout <= cout - 1'b1;
44             end
45         end
46     end
47 endmodule
```

A számláló 8 bites a korábbi 4 helyett

A párhuzamos betöltő bemenetet alacsony aktívra változtattuk, mert a DE0 fejlesztőkártyán lévő nyomógombok is ilyen működésűek

Beiktattunk egy ún. pre-scaler áramkört, amely a bejövő 50MHz-es órajelet leosztva a számlálás ütemét kb. 1 másodpercre állítja be

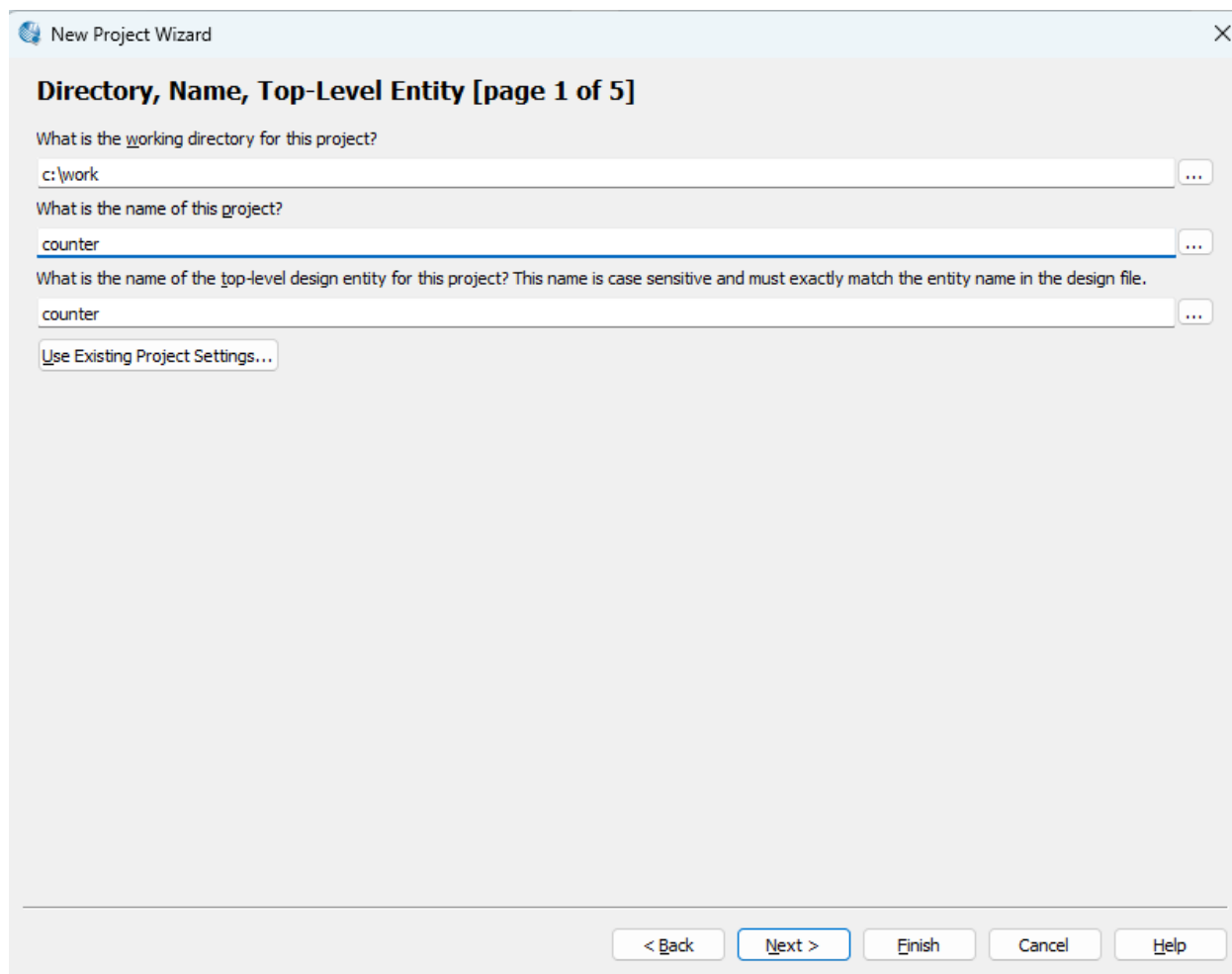
9-1. ábra A módosított számláló Verilog modellje

A számláló Verilog modelljét az *Altera DE0* fejlesztőkártyán rendelkezésre álló hardverkörnyezethez illesztettük a következő módon:

- A fejlesztőkártyán 10 db kétállású kapcsoló van, amelyekből egy a globális engedélyezés, egy a számlálás iránya. A fennmaradó 8 kapcsolót felhasználhatjuk egy 8-bites buszként a számláló inicializálására a párhuzamos betöltés funkcióval. A számláló tehát immár 8-bites.
- A számláló értékét a kártyán rendelkezésre álló 10 db LED közül az alsó nyolcon fogjuk megjeleníteni.
- A párhuzamos betöltést a rendelkezésre álló 3 nyomógomb közül az egyik valósítja meg, negált logikával. A lenyomott gomb logikai alacsony szintet kényszerít a megfelelő I/O lábra.
- A kártyán egy 50 MHz-es kristályoszillátor áll rendelkezésre. Annak érdekében, hogy a számlálás jól megfigyelhető legyen, egy ún. pre-scaler áramkört iktatunk be, amely a számláló órajelének frekvenciáját látszólag kb. 1 Hz-esre módosítja. Valójában nem órajel-leosztásról van szó, csak a számlálást egy olyan belső engedélyező jelhez kötöttük, amely másodpercenként csupán egyszer aktív.

Az *Altera Quartus II* egy integrált fejlesztőkörnyezet, amely hatékony felületet nyújt a szintézis lépéseinek végrehajtásához. A fejlesztés során minden lépés elvégezhető a grafikus felhasználói felületen keresztül, de lehetőség van parancssorból történő használatra is. A szintézis lépéseit ebben az anyagban is bemutatjuk, hogy ne kelljen az előző laborok anyagait is böngészni.

A szintézist egy új projekt létrehozásával kezdjük. Új projektet a **File** menü **New Project Wizard...** parancsával hozhatunk létre. Ekkor egy párbeszédablak-sorozat segítségével beállíthatjuk a projekt legfontosabb részleteit. Az első ablak egy bevezetés (*Introduction*), amely bemutatja, hogy a varázsló milyen beállításokat tesz lehetővé. A **Next** gomb segítségével lépünk át a következő ablakra, ahol a projekt helyét és nevét adhatjuk meg, valamint azt, hogy a HDL forrás hierarchiájában melyik a legfelső szintű (*toplevel*) modul, amelyiket szintetizálni szeretnénk (9-2. ábra).



New Project Wizard

Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

c:\work

What is the name of this project?

counter

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

counter

Use Existing Project Settings...

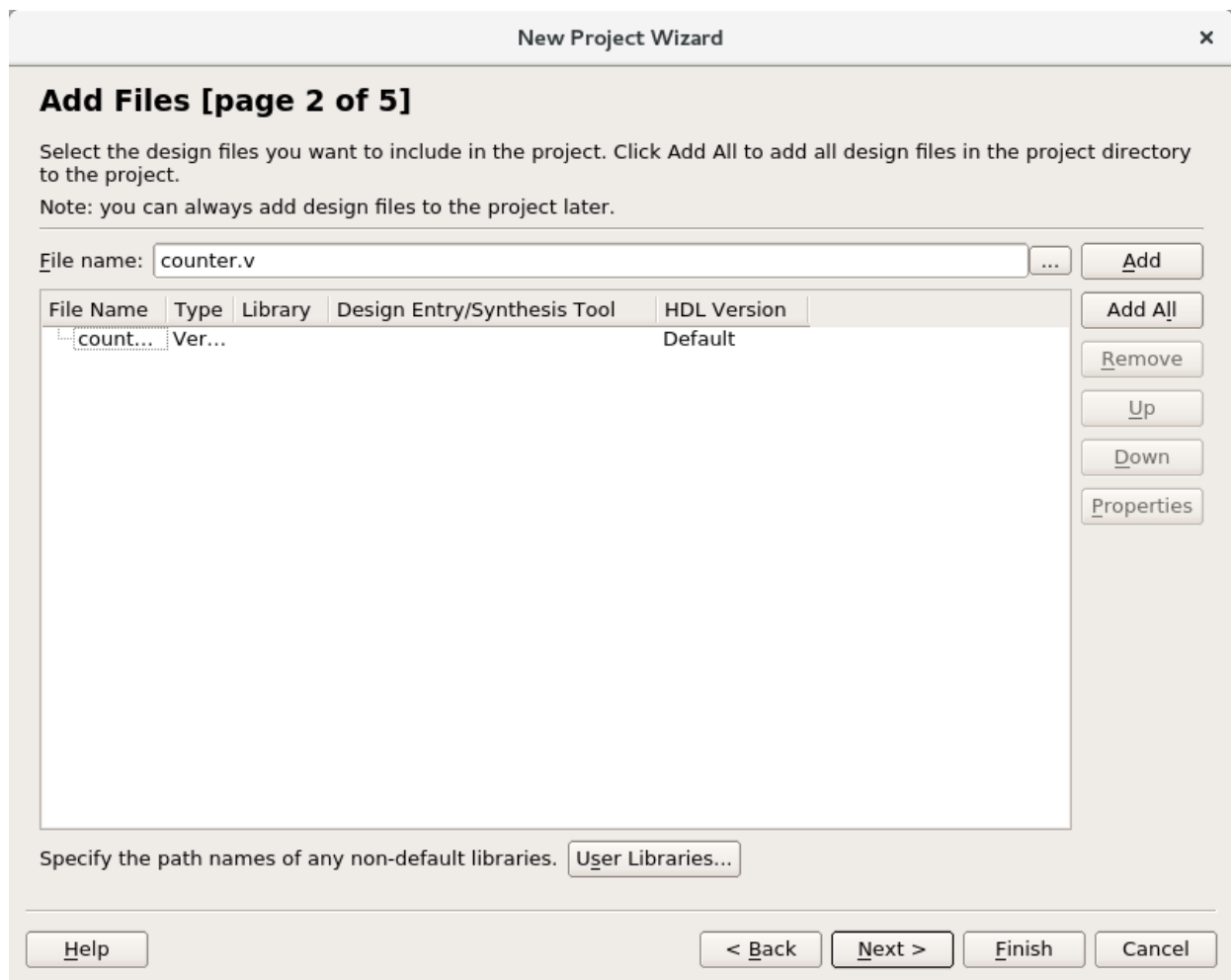
< Back Next > Finish Cancel Help

9-2. ábra Az új projekt helye, neve és a toplevel modul neve

Az új projektet érdemes a C:\Work könyvtárban kell létrehozni. Ha nem létező könyvtárat adtunk meg, akkor a program rákérdez, hogy létrehozhatja-e azt. Hagyjuk jóvá az új könyvtár létrehozását!

Ügyeljünk arra, hogy a top-level design entity-t, **counter** helyesen adjuk meg. Itt a fő modul nevét kell megadni, nem pedig a forrásfájl nevét. Természetesen ezt később megváltoztathatjuk.

A következő ablakban lehetőségünk van forrásfájlok hozzáadására. A *counter.v* forrásfájl betallóztatása után ne felejtsünk el az **Add** gombra kattintani (9-3. ábra)! Forrásfájlokat természetesen a projekt létrehozása után is tudunk hozzáadni.



9-3. ábra Forrásfájlok hozzáadása

A forrásfájlok hozzáadása után a **Next** gombra kattintva az FPGA eszközcsalád és azon belül a konkrét eszköz kiválasztására van lehetőség. A DE0 fejlesztőkártyán egy Cyclone III FPGA kapott helyet, amelynek típusazonosítója a tokról leolvasható: EP3C16F484C6. A grafikus felület az eszköz gyorsabb kikeresése céljából szűrők beállítását teszi lehetővé. Az eszközcsalád (*Family*) mezőben válasszuk a **Cyclone III**-at, a tokozásnál (**Package**) **FBGA**-t (*Flip-chip Ball Grid Array*), a kivezetések számánál (**Pin count**) **484**-et és a sebességnél (**Speed grade**) - amely a gyártástechnológiára utal - **6**-ot. Ezeknek a szűrőbeállításoknak négy különböző eszköz felel meg, ezek közül válasszuk az elsőt (9-4. ábra)!

New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

Device family

Family: Cyclone III
Devices: All

Target device

☐ Auto device selected by the Fitter
☒ Specific device selected in 'Available devices' list
☐ Other: n/a

Show in 'Available devices' list

Package: FBGA
Pin count: 484
Speed grade: 6
Name filter:
☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit el
EP3C16F484C6	1.2V	15408	347	516096	112
EP3C40F484C6	1.2V	39600	332	1161216	252
EP3C55F484C6	1.2V	55856	328	2396160	312
EP3C80F484C6	1.2V	81264	296	2810880	488

Help **< Back** **Next >** **Finish** **Cancel**

9-4. ábra Eszközcsalád és eszköz kiválasztása

Az eszköz kiválasztása után már csak egy beállítás van hátra. Bár a számláló áramkörünk működését a fejlesztőkártyán fogjuk ellenőrizni, a teljesség kedvéért bemutatjuk, hogy miként lehet előállítani azokat a fájlokat, amelyek az időzítési szimulációhoz szükségesek, nevezetesen a post-layout HDL modellt (amelyet FPGA technológia esetén post-place&route modellnek hívnak) és az időzítési viszonyokat leíró SDF fájlt. Ahhoz, hogy ezek a fájlok a nekünk megfelelő formátumban generálódjanak, ki kell választanunk a verifikációs környezetet és a generált kapusintű modell nyelvét. Ehhez az **EDA Tool Settings** ablak **Simulation** sorának **Tool Name** mezőjében állítsuk be a **ModelSim**-et, a **Format(s)** mezőben pedig válasszuk a **Verilog HDL**-t (9-5. ábra).

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/S...	<None>	<None>	<input checked="" type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	ModelSim	Verilog HDL	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Formal Verifica...	<None>		<input type="checkbox"/>
Board-Level	Timing	<None>	<input type="checkbox"/>
	Symbol	<None>	<input type="checkbox"/>
	Signal Integrity	<None>	<input type="checkbox"/>
	Boundary Scan	<None>	<input type="checkbox"/>

9-5. ábra A kimeneti modell-generáláshoz szükséges beállítások

Ezek után a **Next** gombra kattintva egy összegzést kapunk, amelyet a **Finish** gombbal hagyhatunk jóvá.

Mielőtt elkezdenénk végrehajtani a szintézis lépéseit, másoljuk az órajel és időzítési követelményeket tartalmazó SDC fájlt a projektkönyvtárba!

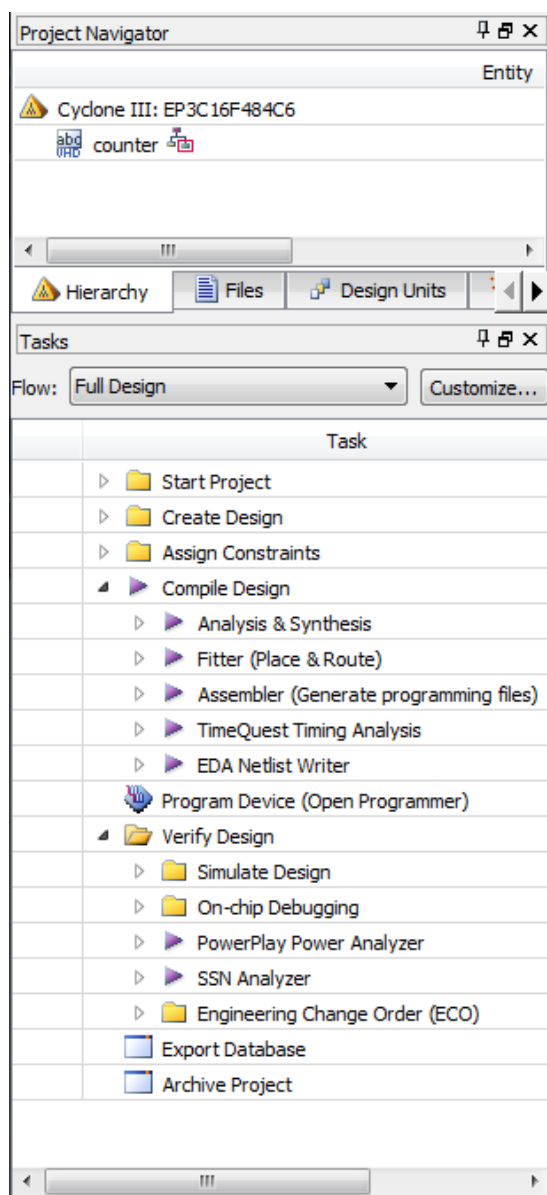
A funkcionalitáson kívül a specifikáció részét képezik a leendő áramkör működési feltételei is. Ezek közül elsődleges jelentőségűek az időzítéssel kapcsolatos megkötések (órajel-frekvencia, amelyen az áramkörnek működnie kell). E megkötések formális megfogalmazására az ún. SDC (*Synopsys Design Constraints*) formátumot használjuk. A 9-6. ábra egy SDC fájlt mutat, amely egy 50 MHz-es órajelet és 0,2 ns-os be- és kimeneti késleltetéseket definiál.

Az SDC fájl szerepe kettős: az ún. *timing-driven szintézis* során - amikor az RTL szintézis célja nem a lehető legkisebb és nem is a lehető leggyorsabb áramkör, hanem az az áramkör, ami minimális erőforrásigénnyel képes teljesíteni az időzítési követelményeket - peremfeltételként szolgál a szintézist végző szoftver számára, másrészt felhasználható az ún. statikus időzítés-analízis (*Static Timing Analysis, STA*) során a szintetizált áramkör időzítési viszonyainak ellenőrzésére. A szintetizált áramkör kapuinak - és a fizikai szintézis előrehaladottságától függően a vezetékezésnek - a késleltetése ismert. Ezt az ismert késleltetést összevetve az SDC fájlban megfogalmazott megkötések által indikált maximálisan megengedhető késleltetésekkel könnyen megállapítható, hogy a leszintetizált áramkör megfelel-e az elvárásoknak. E módszer előnye, hogy sokkal gyorsabb, mint az időzítési szimuláció, hátránya, hogy csak azokat a lokális adatutakat képes ellenőrizni, amelyeket az SDC fájl "lefed". Ha a leszintetizált áramkörben olyan lokális adatúton van időzítési hiba, amelyről az SDC fájl "nem nyilatkozott", akkor a hiba rejtve marad.

```
1 create_clock -period 20 -name clk [get_ports clk]
2
3 set_input_delay -clock clk 0.2 [all_inputs]
4 set_output_delay -clock clk 0.2 [all_outputs]
5
6 derive_clock_uncertainty
```

9-6. ábra: Példa SDC fájl

A grafikus felület bal oldalán található **Project Navigator** ablak tetején a HDL modell szerkezetét, alján pedig a szintézis elvégzendő részfeladatainak felsorolását láthatjuk. A **Flow** nevű legördülő menüben válasszuk a **Full Design** lehetőséget (9-7. ábra)!



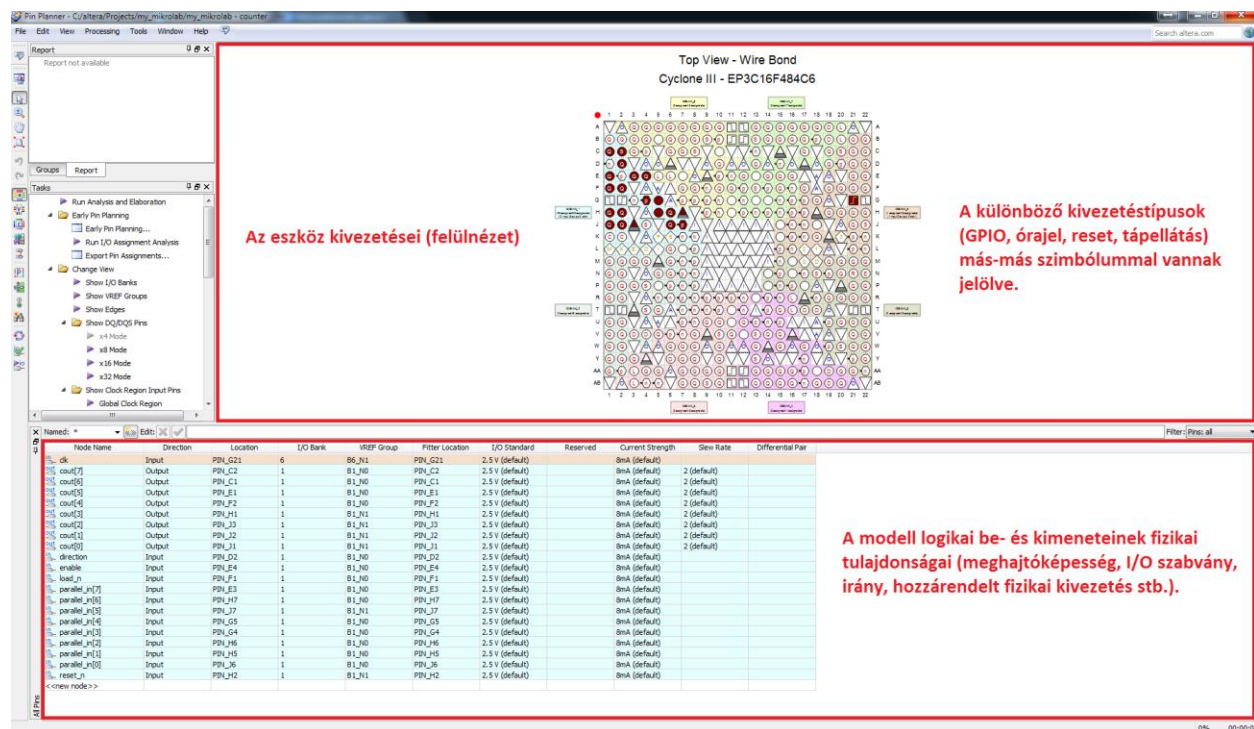
9-7. ábra A Project Navigator ablak

A következő lépés a HDL modell szintaktikai ellenőrzése és szerkezetének feltérképezése. A **Project Navigator** ablakban a **Compile Design** menüpont alatt található **Analysis & Synthesis** almenüben kattintsunk duplán az **Analysis & Elaboration** parancsra! Ha a teljes kódbázis szintaktikailag helyesnek bizonyul, akkor a szintézis eszköz egy belső reprezentációt készít a HDL modellről (*elaborate*), amely során további, immár szemantikai ellenőrzésekre is sor kerül. E lépés eredménye a HDL modell struktúráját tükröző elvont, a szintézis eszközre jellemző

modell, amely egy könyvtárszerkezetre emlékeztető formában tárolja a terv egyes elemeinek, tervezési egységeinek hierarchiáját.

A HDL modell beolvasása és elemzése után már ismertek a modell logikai kivezetései, amelyeket az FPGA eszköz általános célú kivezetéseéhez kell hozzárendelnünk. Ez a lépés az ún. *kivezetés-hozzárendelés (pin assignment)*. Ha egy már elkészült PCB-n (*Printed Circuit Board*) helyet foglaló FPGA-ról van szó (ahogy esetünkben is), akkor a logikai kivezetések fizikai megfelelőinek kiválasztása nem teljesen önkényes. Egy logikai kimenetet értelemszerűen egy meghajtást igénylő, míg egy logikai bemenetet egy meghajtó jellegű erőforrásra kell kapcsolnunk. A felhasznált fejlesztőkártya felhasználói dokumentációja tartalmazza azt az információt, hogy a PCB egyes erőforrásai (LED-ek, kapcsolók stb.) az FPGA melyik kivezetéséhez vannak hozzákapcsolva.

A kivezetés-hozzárendelés a Quartus II környezetben az **Assignments** menü **Pin Planner** nevű alprogramjával végezhető el (9-8. ábra).



Az eszköz kivezetései (felülnézet)






















A különböző kivezetéstípusok (GPIO, órajel, reset, tápellátás) más-más szimbólummal vannak jelölve.

Node Name	Direction	Location	I/O Bank	VRFP Group	Pin Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair
clk	Input	PP1_G21	6	B6_N11	PP1_G21	2.5 V (default)		8mA (default)	2 (default)	
cout[7]	Output	PP1_C2	1	B1_N0	PP1_C2	2.5 V (default)		8mA (default)	2 (default)	
cout[6]	Output	PP1_C1	1	B1_N0	PP1_C1	2.5 V (default)		8mA (default)	2 (default)	
cout[5]	Output	PP1_E1	1	B1_N0	PP1_E1	2.5 V (default)		8mA (default)	2 (default)	
cout[4]	Output	PP1_F2	1	B1_N0	PP1_F2	2.5 V (default)		8mA (default)	2 (default)	
cout[3]	Output	PP1_H1	1	B1_N11	PP1_H1	2.5 V (default)		8mA (default)	2 (default)	
cout[2]	Output	PP1_J3	1	B1_N11	PP1_J3	2.5 V (default)		8mA (default)	2 (default)	
cout[1]	Output	PP1_J2	1	B1_N11	PP1_J2	2.5 V (default)		8mA (default)	2 (default)	
cout[0]	Output	PP1_J1	1	B1_N11	PP1_J1	2.5 V (default)		8mA (default)	2 (default)	
direction	Input	PP1_D2	1	B1_N0	PP1_D2	2.5 V (default)		8mA (default)		
enable	Input	PP1_E4	1	B1_N0	PP1_E4	2.5 V (default)		8mA (default)		
test_0	Input	PP1_F1	1	B1_N0	PP1_F1	2.5 V (default)		8mA (default)		
parallel_in[7]	Input	PP1_E3	1	B1_N0	PP1_E3	2.5 V (default)		8mA (default)		
parallel_in[6]	Input	PP1_H7	1	B1_N0	PP1_H7	2.5 V (default)		8mA (default)		
parallel_in[5]	Input	PP1_J7	1	B1_N11	PP1_J7	2.5 V (default)		8mA (default)		
parallel_in[4]	Input	PP1_G5	1	B1_N0	PP1_G5	2.5 V (default)		8mA (default)		
parallel_in[3]	Input	PP1_G4	1	B1_N0	PP1_G4	2.5 V (default)		8mA (default)		
parallel_in[2]	Input	PP1_H6	1	B1_N0	PP1_H6	2.5 V (default)		8mA (default)		
parallel_in[1]	Input	PP1_H5	1	B1_N0	PP1_H5	2.5 V (default)		8mA (default)		
parallel_in[0]	Input	PP1_H6	1	B1_N0	PP1_H6	2.5 V (default)		8mA (default)		
reset_in	Input	PP1_H2	1	B1_N11	PP1_H2	2.5 V (default)		8mA (default)		
<-new node>										

A modell logikai be- és kimeneteinek fizikai tulajdonságai (meghajtóképesség, I/O szabvány, irány, hozzárendelt fizikai kivezetés stb.).

9-8. ábra Pin Planner

A kivezetésekre vonatkozó táblázatban elegendő a **Location** oszlopot kitölteni a 9-9. szerint. A kitöltés során a pin nevéből csak a helyzetét rögzítő betű-szám kombinációt (pl. G21) kell beírni, a többit a szoftver automatikusan kiegészíti.

Node Name	Direction	Location
 clk	Input	PIN_G21
 cout[7]	Output	PIN_C2
 cout[6]	Output	PIN_C1
 cout[5]	Output	PIN_E1
 cout[4]	Output	PIN_F2
 cout[3]	Output	PIN_H1
 cout[2]	Output	PIN_J3
 cout[1]	Output	PIN_J2
 cout[0]	Output	PIN_J1
 direction	Input	PIN_D2
 enable	Input	PIN_E4
 load_n	Input	PIN_F1
 parallel_in[7]	Input	PIN_E3
 parallel_in[6]	Input	PIN_H7
 parallel_in[5]	Input	PIN_J7
 parallel_in[4]	Input	PIN_G5
 parallel_in[3]	Input	PIN_G4
 parallel_in[2]	Input	PIN_H6
 parallel_in[1]	Input	PIN_H5
 parallel_in[0]	Input	PIN_J6
 reset_n	Input	PIN_H2

9-9. ábra A logikai kivezetések összerendelése a fizikai lábakkal

Ha a kivezetés-hozzárendelést a 9-9. szerint végezzük el, akkor a számláló kivezetései az alábbi, a fejlesztőkártyán rendelkezésre álló erőforrásokkal lesznek összekapcsolva:

- órajel (**clk**): 50 MHz-es kristályoszillátor
- reset (**reset_n**): BTN0 (nyomógomb)
- párhuzamos betöltés (**load_n**): BTN2 (nyomógomb)
- számlálás engedélyezése (**enable**): SW8 (kapcsoló)
- számlálás iránya (**direction**): SW9 (kapcsoló)
- 8-bites párhuzamos bemenet (**parallel_in**): SW0-SW7 (kapcsolók)
- számláló értéke (**cout**): LEDG0-LEDG7 (LED-ek)

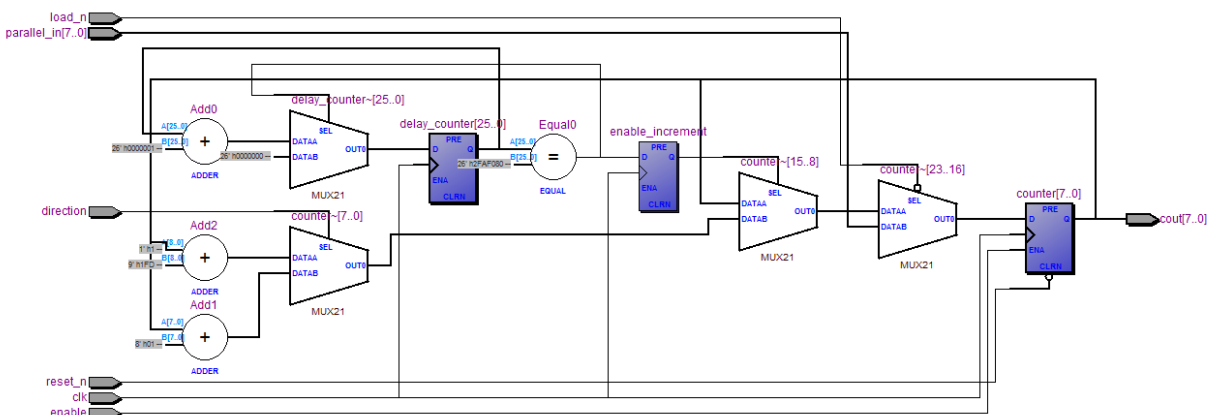
A **Pin Planner** a **File** menü **Close** parancsával zárható be.

Ezután a **Project Navigator** ablak **Compile Design** parancsára duplán kattintva a program a következő lépéseket automatikusan elvégzi:

- *Analysis and Synthesis*: Generikus szintézis és mapping.
- *Fitter (Place & Route)*: Csoportképzés, elhelyezés és huzalozás.
- *Assembler (Generate programming files)*: Bitfolyam előállítás.
- *TimeQuest Timing Analysis*: Statikus időzítés-analízis (lásd: Szintézis standard cellás ASIC technológiára c. laboratórium elméleti összefoglaló: 7.4.1. pont).
- *EDA Netlist Writer*: Post-Place&Route modell és SDF fájl generálása.

A szintézis eredménye a következőképpen ellenőrizhető: nyissuk le a **Project Navigator** ablak **Analysis and Synthesis** pontját, azon belül pedig keressük a **Netlist Viewers** mappát, amelyben

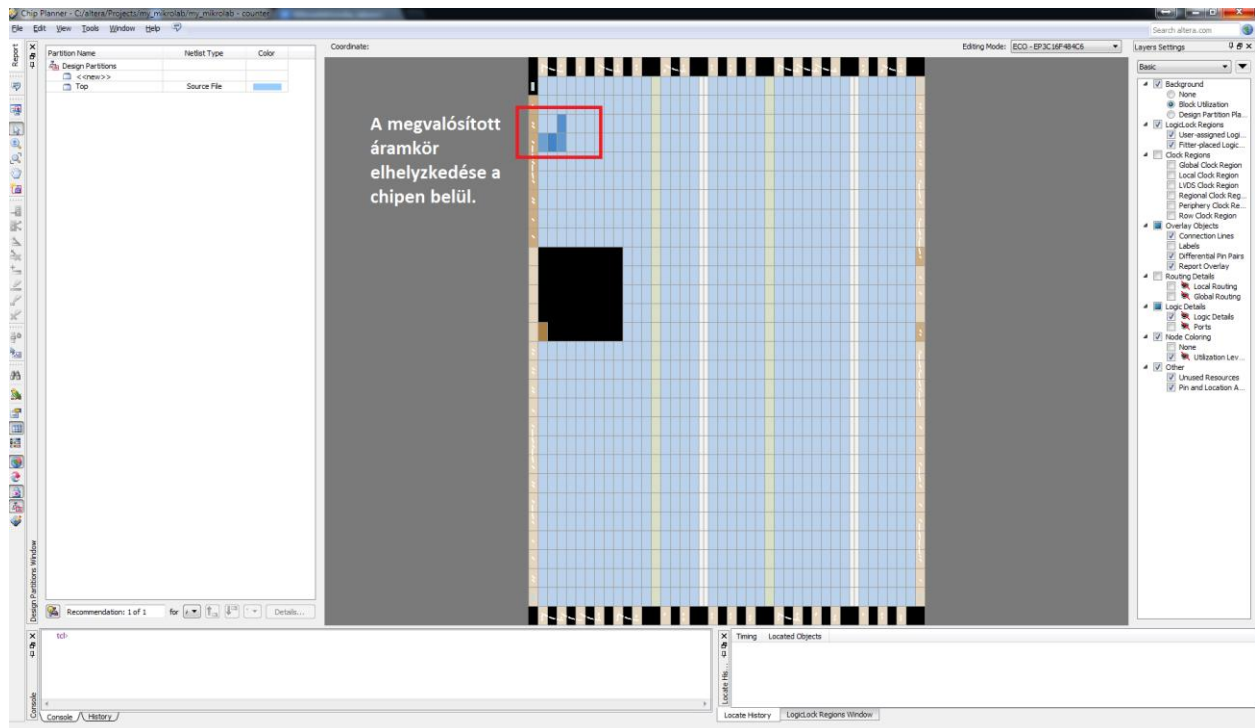
három különböző alkalmazás áll rendelkezésre a szintézis ellenőrzésére. Az **RTL viewer** - ahogy neve is utal rá - az RTL modell alapján elkészített, RTL és kapusintű erőforrásokat tartalmazó netlista (9-10. ábra).



9-10. ábra A számláló áramkör kapcsolási sémája

Bár összetett rendszerek esetén egy ilyen kapcsolási rajz nehezen átláthatónak tűnhet, arra kiválóan alkalmas, hogy az RTL tervező eldöntse, hogy a szintézis szoftver valóban azt az áramkört valósította meg, amelyet ő elképzelt. A **State Machine Viewer** hasonló célú alkalmazás az állapotgépek megvalósításának ellenőrzésére, a **Technology Map Viewer** segítségével pedig ellenőrizhető a post-mapping modell kapcsolási rajza.

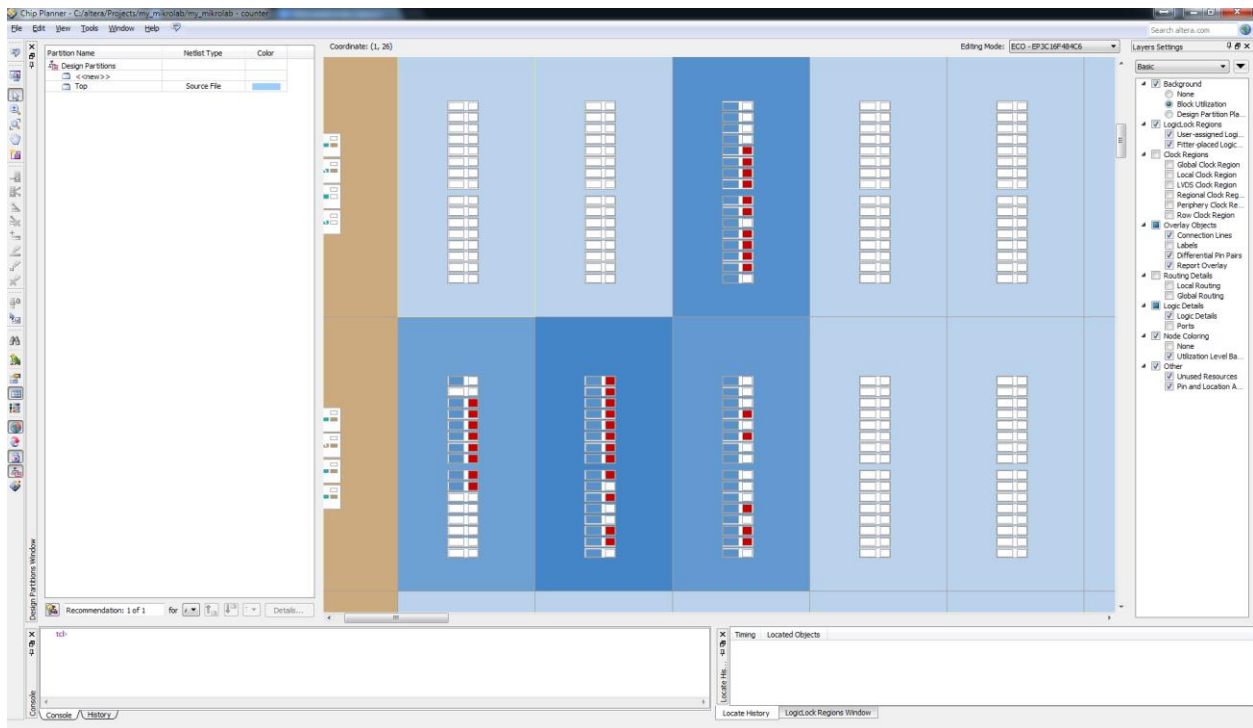
Az automatizált elhelyezés és huzalozás eredményét is megtekinthetjük a **Chip Planner** alkalmazás segítségével, amely a **Tools** menü **Chip Planner** parancsával indítható el (9-11. ábra).



9-11. ábra Chip Planner

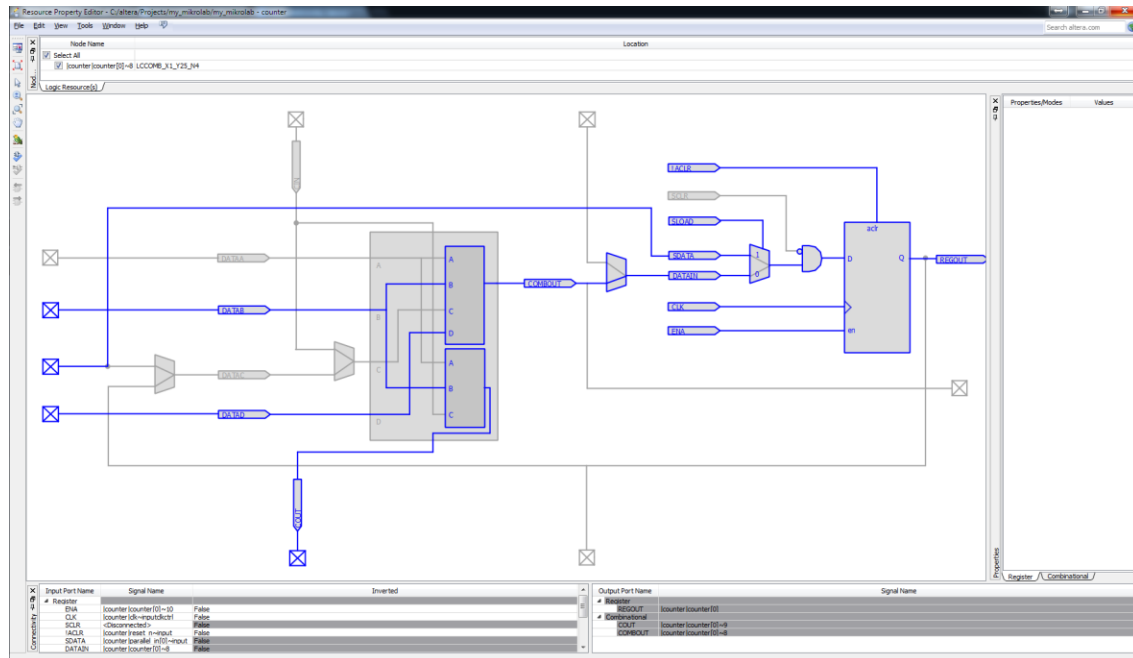
A *Chip Planner* az eszköz felülnézeti képét mutatja, kiemelve az egyes logikai blokkokat és heterogén architektúrális elemeket. Az ábrán világoskék színnel jelölt LAB-ek (Logic Array Block) árnyalata kihasználtságuktól függően változik; minél több BLE foglalt az adott LAB-en belül, annál sötétebb a hozzá tartozó téglalap. Látható, hogy a számláló áramkör nem igényel sok erőforrást, egészen kis részét foglalja csak el a rendelkezésre álló LAB-eknek. Ha ráközelítünk a foglalt területekre, az áramkör szerkezete a legapróbb részletekig megfigyelhető.

Noha ugyanazt az RTL modellt szintetizálta, az elrendezés nem biztos, hogy megfelel az itt találhatóknak. FPGA-k esetében tipikusan véletlenszámmal segített (random constrained) optimalizációt használnak mind az elhelyezés, mind a huzalozás esetén.



9-12. ábra A számláló áramkör által felhasznált LAB-ek

A 9-12. ábrán megfigyelhető a LAB-ekben helyet foglaló 16 BLE. A BLE-k egy-egy kék és piros téglalappal vannak jelölve. A piros téglalapok az adott BLE regiszterének, a kék pedig LUT-jának a kihasználtságát jelzik. Egy BLE-re duplán kattintva azt is megfigyelhetjük, hogy az adott BLE konfigurációja milyen áramköri részletet valósít meg (9-13. ábra).

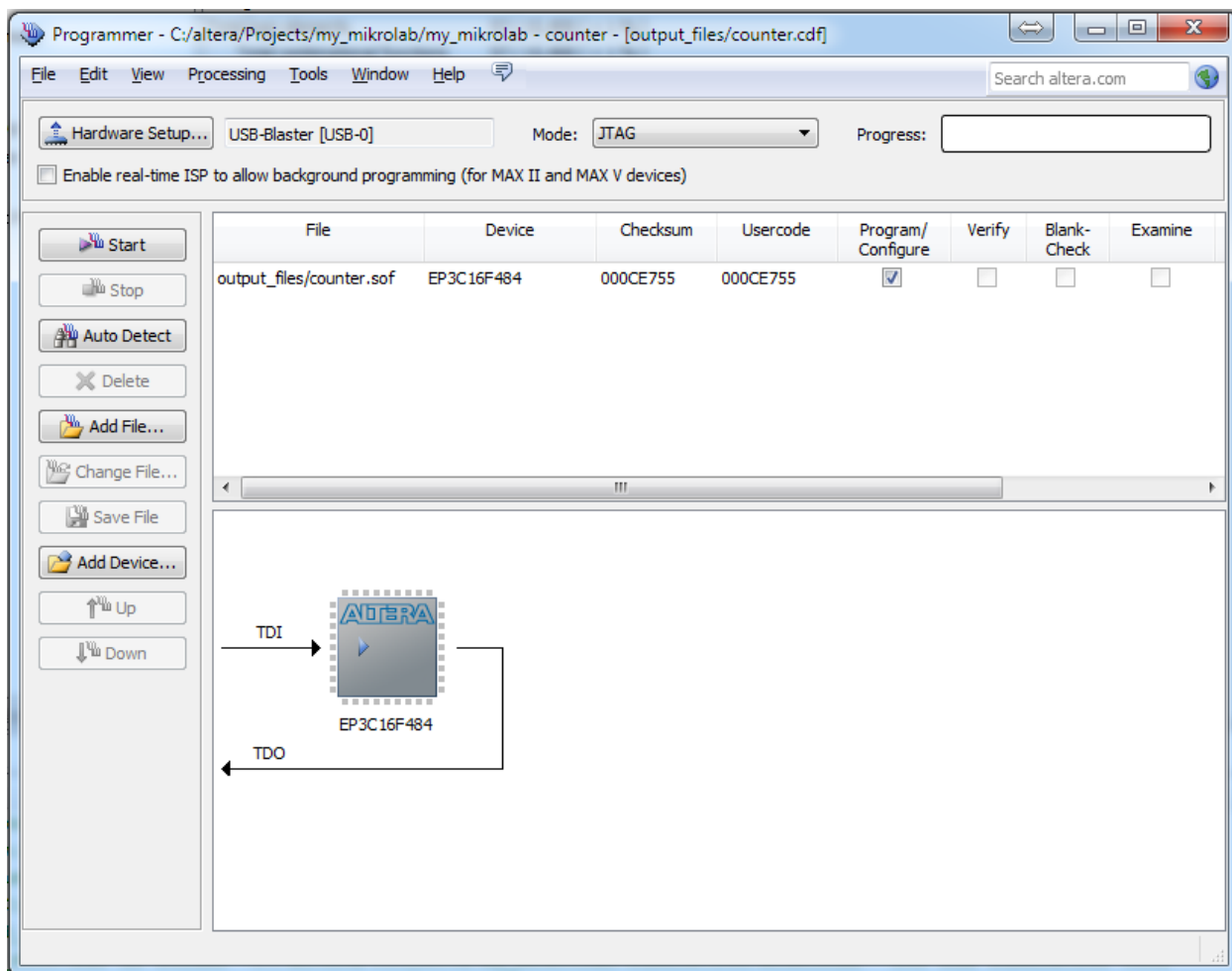


9-13. ábra BLE konfigurációja

A szintézis minden lépését elvégeztük, így nincs más hátra, mint kipróbálni az áramkört élesben.

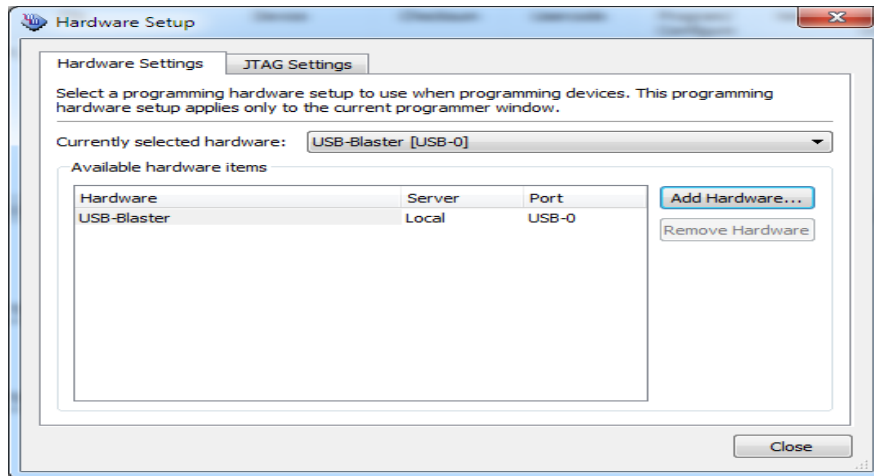
Ha még nem tettük meg, csatlakoztassuk a kártyát USB-n keresztül! Ne felejtsük el bekapcsolni a piros nyomógombbal 😊 . Szép demo villogást kell látnunk. Az USB elegendő tápfeszültséget ad, a külső tápra nincs szükség. A bal alsó SW11 kapcsoló maradjon *“RUN”* állásban! A *“PROG”* állás a konfiguráló FLASH EEPROM írására szolgál, resetkor innen íródik be automatikusan az FPGA konfiguráció. Erre most nem lesz szükség, működés közben fogjuk újrakonfigurálni.

A konfigurációs memória tartalmának feltöltése az eszközre a **Tools** menü **Programmer** parancsával indítható alkalmazással lehetséges (9-14. ábra).



9-14. ábra Programmer

Ha a bal felső sarokban lévő **Hardware Setup** mezőben **No Hardware** feliratot látunk, akkor a **Hardware Setup** gombra kattintva a lehetséges hardverelemek közül válasszuk az **USB Blaster**-t (9-15. ábra).



9-15. ábra USB Blaster kiválasztása

Ezután a **Programmer** ablak felső mezőjében megjelenik az aktuális projektünkben már rendelkezésre álló programozófájl (*output_files/counter.sof*), alsó mezőjében pedig látni fogjuk azt a JTAG láncot, amelyet a program az adott programozófájjal kompatibilisnek ismert fel. Ha a programozófájl nem jelent meg automatikusan, akkor bal oldali gombsor **Add File...** gombjával kézzel is kijelölhetjük azt. A bal oldali gombsor start gombjára kattintva a letöltés elkezdődik. A sikeres letöltést a jobb felső sarokban elhelyezkedő **Progress** mező **"100% (Successful)"** felirata jelzi.

9.1. Elvégzendő feladat

Az előző két laboron megismert tervezési módszerek és a demo projekt felhasználásával alkosson valami látványosat! (de legalábbis ami villog 😊) Fotót tegyen a jegyzőkönyvbe!

Ötletek: (a feladat nehézségét a csillagok száma próbálja szemléltetni, de ez szubjektív)

- Knight Rider 8 LED-del, egy LED mozog jobbra balra (*)
- Knight Rider extra, mind a 10 LED használatával, 3 LED mozog jobbra balra és a két végén „kimegy” a LED. (**)
- Knight Rider full extra: a középső LED teljes, jobbra-balra pedig fele fényerővel világítson, sebessége nyomógombokkal szabályozható legyen! (***)
- LED fényerő szabályozó a nyomógombokkal (**)
- A számláló értékének kijelzése a hétszegmenses kijelzőn, hexadecimálisan (*)
- A számláló átalakítása BCD számlálóvá és kijelzése a hétszegmenses kijelzőn (**)
- Dobókocka. Valamelyik nyomógomb lenyomásakor pörög a hétszegmenses kijelző, ahogy csak tud, elengedéskor megáll (ez elég véletlen lesz) és kijelzi az értéket 1-6 között. (**)
- Félkarú rabló a 4 hétszegmenses kijelzőn (***)
- Hexspeak scroll a hétszegmenses kijelzőkön (**)
- Jöhet saját ötlet is, ha látványos, jövőre bekerül ebbe a syllabuszba is.

9.2. Jótanácsok

1. Ez nem egyszerű, pontosan követni kell a sillabuszt, a legkisebb hiba is nem működő áramkörhöz vezethet. Tapasztalatok alapján a páros munka hatékonyabb.
2. A prescaler áramkört ne vegye ki a tervből, ha valamilyen látható LED-es villogtatás a feladat. 50MHz-en valószínűleg lassú lesz a szeme, hogy ellenőrizni tudja ☹️. Egy tízszeres gyorsítás viszont hasznos lehet.
3. Módosítás után a teljes szintézist meg kell csinálni, nem csak az *Analyze...* stb-t. A legegyszerűbb a felső menüsoron a „*Start Compilation*” gomb nyomkodása (ugyanitt a *Programmer...* ez a kettő kell, más nem ☹️)



4. Csak nem blokkoló értékadást (\leq) használjon.
5. Ne felejtse el, hogy szinkron hálózatot tervez. Az értékadások nem sorban következnek, mint a programozásban, hanem egyszerre, hiába írtuk a forrásban egymás után. Az értékadás bal oldala egy flip-flop bemenete, emiatt csak a következő órajelben kapja csak meg az értéket.
6. Ha a modul be és kimeneteit megváltoztatja, a változást a Pin Plannerben is át kell vezetni. Különösen, ha egy jelet eltávolít, mert ilyenkor a Quartus az innovatív „Can't fit in device” hibaüzenetet adja.
7. A pin kiosztás a mellékelt DE0_User_manual.pdf file-ban található.