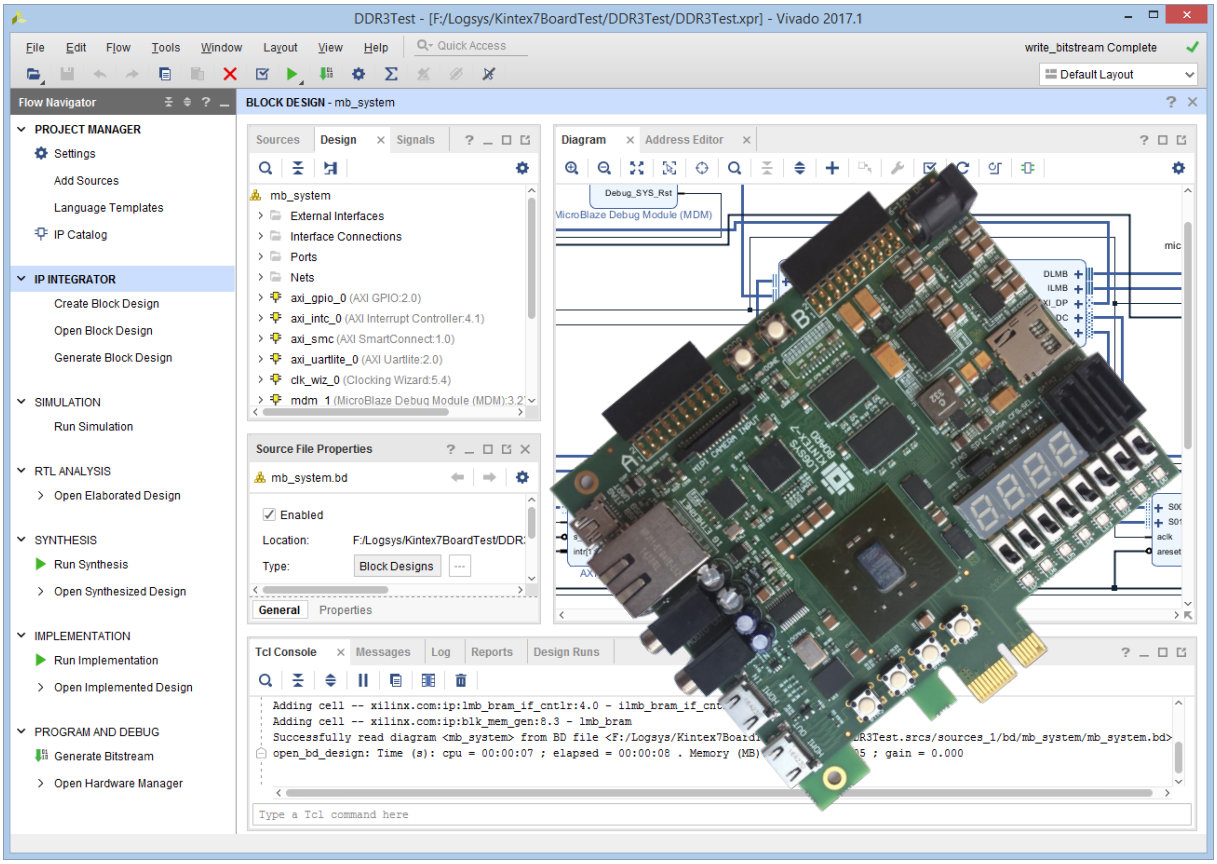
LOGSYS

Egyszerű alkalmazás készítése a LOGSYS Kintex-7 FPGA kártyára a Xilinx Vivado fejlesztői környezet használatával



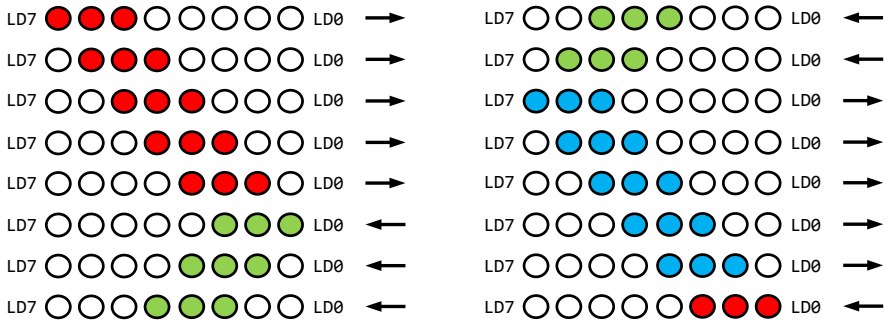
2017. június 16.

Verzió 1.0

http://logsys.mit.bme.hu

A dokumentum célja egy egyszerű alkalmazás elkészítésének bemutatása a LOGSYS Kintex-7 FPGA kártyára a Xilinx Vivado 2017.1 fejlesztői környezet használatával. A leírtak megértéséhez alapfokú digitális technika és Verilog HDL ismeret szükséges.

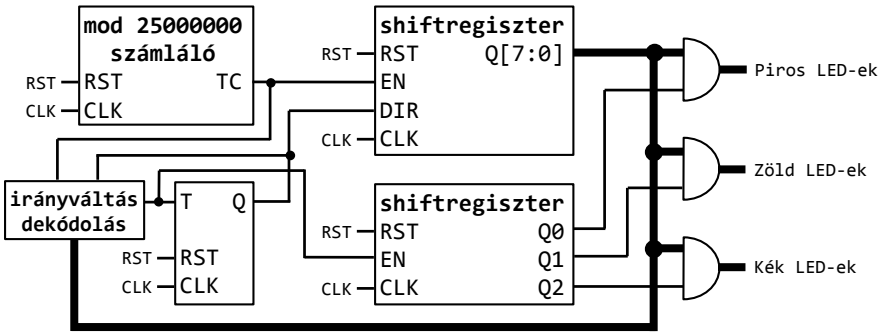
Az elkészítendő rendszer egy Knight Rider futófényt valósít meg három LED kigyújtásával, azaz három bekapcsolt szomszédos LED adott ütemben folyamatosan vándorol, a szélső helyzet elérésekor irányt váltva. Mivel az FPGA kártyán RGB LED-ek vannak, ezért a feladatot bonyolítsuk azzal, hogy minden irányváltáskor a LED-ek színe a következőre vált (1. ábra). A léptetés üteme legyen 250 ms. Szinkron rendszert valósítsunk meg, minden tároló elem a kártyán lévő 100 MHz-es oszcillátor által biztosított órajelről járjon.



1. ábra: A futófény állapotai.

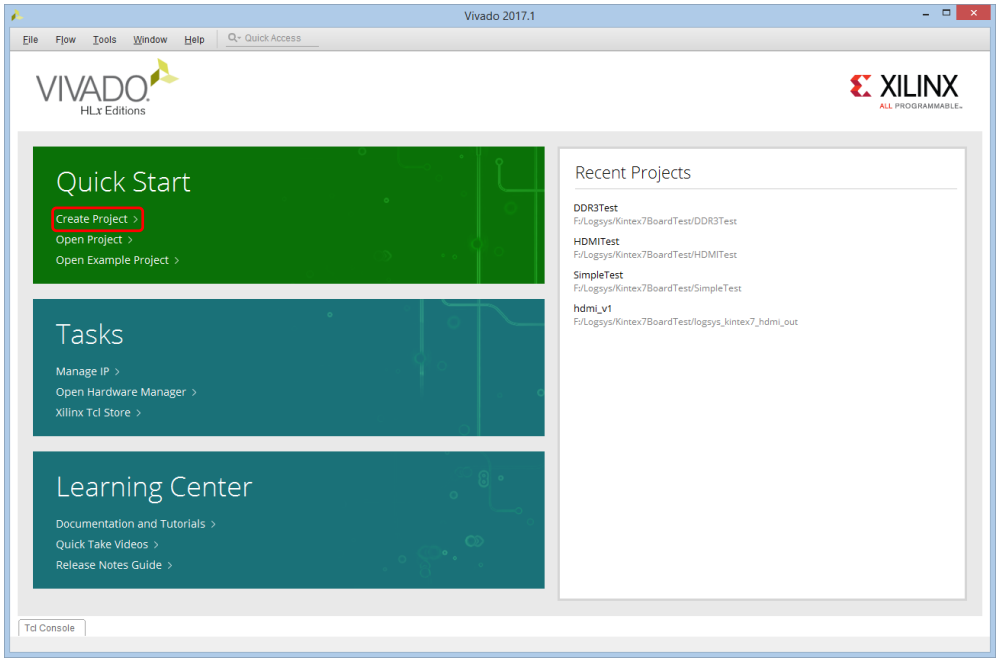
A következőkben megnézzük, hogy milyen funkcionális elemek szükségesek a kívánt rendszer megvalósításához. A felsorolt elemeket tartalmazó blokkvázlat a 2. ábrán látható, ez alapján fogjuk elkészíteni a rendszer Verilog HDL kódját.

- A 4 Hz-es ütemező jelet egy számlálóval állítjuk elő, melynek 25000000 állapota kell, hogy legyen. Ehhez 25 bites számlálóra van szükség.
- A 8 darab LED-en megjelenítendő mintát egy kétirányú és engedélyezhető shiftregiszterrel állítjuk elő, amely kezdetben 11100000 bináris értéket vesz fel.
- A shiftregiszter irány kiválasztó bemenetének értékét a következő végállapot eléréséig azonos értéken kell tartani, így ide egy flip-flop szükséges. Az új irány beállítását a végállapot előtti állapotban kell kezdeményezni, hogy a végállapotban a következő ütemező jel hatására a léptetés már az ellenkező irányba történjen.
- A LED színének kiválasztása egy hárombites, egyirányú és engedélyezhető shiftregiszterrel oldható meg, amelyben egyetlen egy „1” értékű bit forog. Az engedélyezése az irányváltáskor történik.
- A LED-ek meghajtása ÉS kapukon keresztül történik, melyek segítségével a mintát tároló shiftregiszter kimenetét a szín kiválasztó shiftregiszter bitjei maszkolni tudják.



2. ábra: A megvalósítandó rendszer blokkvázlata.

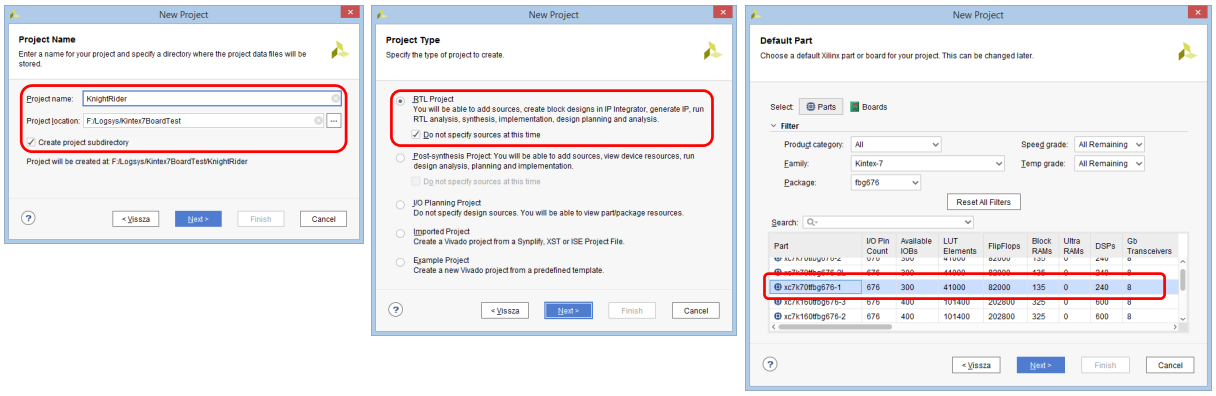
A rendszer megtervezése után indítsuk el a Xilinx Vivado 2017.1 fejlesztői környezetet. A kezdőképernyőt a 3. ábra mutatja. Az új projekt létrehozásához kattintsunk a ***Quick Start*** alatti ***Create Project*** feliratra.



3. ábra: A Xilix Vivado 2017.1 fejlesztői környezet kezdőképernyője.

A projekt létrehozásakor többféle adatot is meg kell adnunk, az egyes lépéseket a 4. ábra szemlélteti.

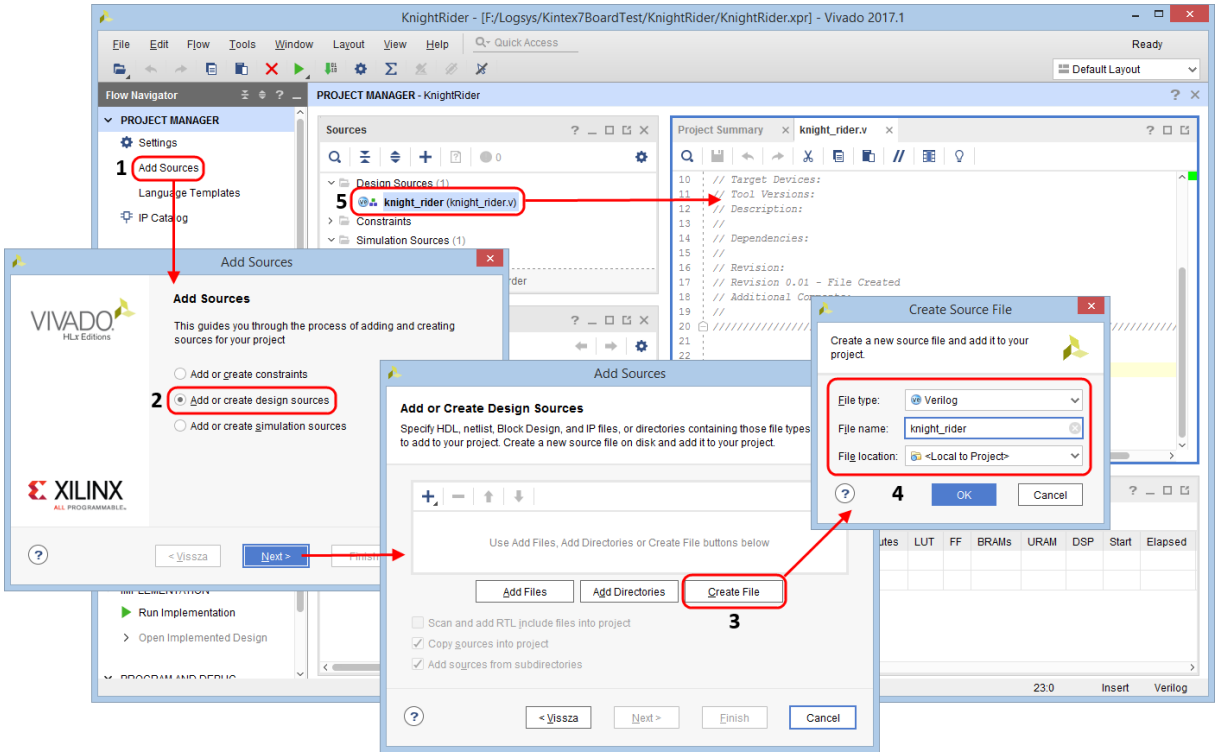
- Az első ablakban meg kell adni a projekt nevét (pl. ***KnightRider***) és az elérési útját. Ha be van jelölve a ***Create project subdirectory*** opció, akkor a megadott könyvtárban létrejön a projekt nevével egyező nevű alkönyvtár.
- A második ablakban kiválasztható a projekt típusa. Mivel Verilog nyelven szeretnénk leírni a rendszert, ezért válasszuk az ***RTL Project*** típust a listából. A ***Do not specify sources at this time*** opciót jelöljük be, mert nem már meglévő forrásfájlt használunk. Az új forrásfájlt később adjuk hozzá a projekthez.
- A harmadik ablakban meg kell adnunk az FPGA típusát, melyre fejlesztünk. A LOGSYS Kintex-7 FPGA kártyán egy XC7K70T-1FBG676I típusú eszköz van, ennek megfelelően a listából válasszuk ki az ***XC7K70TFBG676-1*** elemet.
- Az utolsó ablakban kapunk egy összegzést a megadott beállításokról. A ***Finish*** gombra kattintva létrejön az új projekt.



4. ábra: A project létrehozásának lépései.

Mivel üres projektet hoztunk létre, ezért hozzá kell ahhoz adnunk egy új Verilog forrásfájlt. Ennek lépéseit az 5. ábra szemlélteti.

- A bal oldalon található ***Flow Navigator*** panelen kattintsunk az ***Add Sources*** menüpontra.
- A megjelenő ablakban válasszuk ki a második, ***Add or create design sources*** opciót.
- A következő ablakban lehetőség van meglévő forrásfájlok hozzáadására vagy új forrásfájlok létrehozására. Mivel mi az utóbbit szeretnénk, ezért kattintsunk a ***Create File*** gombra.
- A megjelenő ablakban nevezzük el a forrásfájlt például ***knight_rider*** néven. A fájl típusa legyen ***Verilog*** (a típusnak megfelelő kiterjesztés automatikusan hozzáadódik a fájl nevéhez), a helye pedig legyen ***Local to Project***.
- A ***Finish*** gombra kattintva megjelenik egy ablak, ahol megadhatjuk az új Verilog modul portjait. Itt kattintsunk az ***Ok*** gombra, mert a portokat a forráskód szerkesztésével, manuálisan adjuk majd meg.
- Miután létrejött az új Verilog forrásfájl, annak neve megjelenik a **Sources** panelen a ***Design Sources*** csomópont alatt. Ide duplán kattintva tudjuk azt szerkeszteni.



5. ábra: Új forrásfájl hozzáadása a projekthez.

Mivel az elkészítendő rendszer nagyon egyszerű, így egyetlen Verilog modulban írjuk azt le. A modul fejlécében megadjuk a portokat, melyeken keresztül a modul a külvilággal érintkezik. Két egybites bemeneti portra (100 MHz-es órajel és reset jel), valamint három nyolcbites kimeneti portra (piros, zöld és kék LED-ek vezérlése) van szükség.

```
//*****************************************************************************
//* Knight Rider futófény.                                                    *
//*****************************************************************************
module knight_rider(
   input  wire       clk100M,        //100 MHz-es rendszerórajel.
   input  wire       rstbt,          //Reset jel (RST nyomógomb).
   output wire [7:0] led_r,          //Piros LED-ek.
   output wire [7:0] led_g,          //Zöld LED-ek.
   output wire [7:0] led_b           //Kék LED-ek.
);
```

A modul fejlécének megadása után leírjuk a blokkvázlatban szereplő funkcionális elemeket. Egy 25 bites, 25000000 állapottal rendelkező számláló (***clk_div***) végállapot jelzése (***clk_div_tc***) adja a 4 Hz-es ütemező jelet.

```
//*****************************************************************************
//* Órajel és reset jel.                                                      *
//*****************************************************************************
wire clk = clk100M;
wire rst = rstbt;

//*****************************************************************************
//* A 4 Hz-es ütemező jel előállítása órajel osztással.                       *
//*****************************************************************************
reg  [24:0] clk_div;
wire        clk_div_tc = (clk_div == 25'd0);

always @(posedge clk)
begin
   if (rst || clk_div_tc)
      clk_div <= 25'd24999999;
   else
      clk_div <= clk_div - 25'd1;
end
```

A LED-eken megjelenített mintát a 8 bites, engedélyezhető, kétirányú ***pattern_shr*** shiftregiszter állítja elő. Reset hatására betöltődik az 11100000 kezdőállapot. Ha az ütemező jel engedélyezi a léptetést, akkor az aktuális mintát balra vagy jobbra léptetjük. A léptetés irányát meghatározó ***dir*** jel regiszter típusú, mert ennek állapotát a következő váltásig tartani kell.

```
//*****************************************************************************
//* A megjelenítendő mintát előállító kétirányú shiftregiszter.               *
//*****************************************************************************
reg [7:0] pattern_shr = 8’b1110_0000;
reg       dir;

always @(posedge clk)
begin
   if (rst)                                          //Reset esetén betöltjük a
      pattern_shr <= 8'b1110_0000;                   //kezdőállapotot.
   else
      if (clk_div_tc)                                //Ha a működés engedélyezett
         if (dir)
            pattern_shr <= {pattern_shr[6:0], 1'b0}; //balra léptetünk  (dir=1)
         else                                        //vagy
            pattern_shr <= {1'b0, pattern_shr[7:1]}; //jobbra léptetünk (dir=0)

end
```

A következő forráskód részlet írja le a léptetési irányt kiválasztó ***dir*** regiszter működését. Az irány megváltoztatásának szükségességét a ***change_to_l*** (balra váltás) és a ***change_to_r*** (jobbra váltás) jelek jelzik, hatásuk csak az ütemező jel aktív állapota esetén jut érvényre. Az 1. ábra alapján ezek előállítása egyértelmű. Reset hatására a jobbra léptetés kerül kiválasztásra.

```
//*****************************************************************************
//* Az irányt kiválasztó regiszter.                                           *
//*****************************************************************************
wire change_to_l = ~dir & (pattern_shr[1:0] == 2'b10);
wire change_to_r =  dir & (pattern_shr[7:6] == 2'b01);

always @(posedge clk)
begin
   if (rst)                                          //Reset esetén a kezdetben
```

```
      dir <= 1'b0;                                    //jobbra léptetünk.
   else
      if (clk_div_tc)                                 //Ha a működés engedélyezett
         if (change_to_l)
            dir <= 1'b1;                              //balra módosítjuk az irányt
         else
            if (change_to_r)                          //vagy
               dir <= 1'b0;                           //jobbra módosítjuk az irányt
end
```

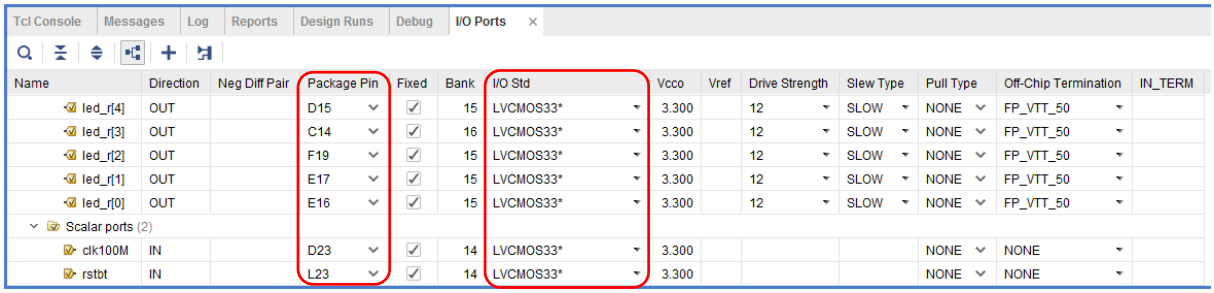
Az utolsó forráskód részlet a szín kiválasztó jel előállítását és a LED-ek meghajtását írja le. Az aktuális színt a 3 bites ***color_shr*** shiftregiszter választja ki, melyben egyetlen egy „1” értékű bitet forgatunk, ha a léptetési irány megváltozik. Reset hatására a piros szín kerül kiválasztásra. A LED-ekre menő mintát ÉS művelettel maszkolja a szín kiválasztó shiftregiszter egy-egy bitje.

```
//*****************************************************************************
//* A színt kiválasztó shiftregiszter.                                        *
//*****************************************************************************
reg [2:0] color_shr = 3’b001;

always @(posedge clk)
begin
   if (rst)                                           //Reset esetén a piros
      color_shr <= 3'b001;                            //színt választjuk ki.
   else
      if (clk_div_tc && (change_to_l || change_to_r)) //Irányváltás esetén
         color_shr <= {color_shr[1:0], color_shr[2]}; //színt is váltunk.
end

//*****************************************************************************
//* A LED-ek meghajtása.                                                      *
//*****************************************************************************
assign led_r = pattern_shr & {8{color_shr[0]}};       //Piros LED-ek.
assign led_g = pattern_shr & {8{color_shr[1]}};       //Zöld LED-ek.
assign led_b = pattern_shr & {8{color_shr[2]}};       //Kék LED-ek.

endmodule
```

A forráskód begépelése után mentsük el a forrásfájlt, majd futtassuk le a szintézist a ***Flow Navigator*** panelen a ***Run Synthesis*** menüpontra kattintva. Az esetleges hibákat javítsuk ki. Ha a szintézis hiba nélkül lefutott, akkor a felugró ablakban válasszuk ki az ***Open Synthesized Design*** opciót a szintetizált terv megnyitásához.

A Verilog forráskód nem tartalmazza azt az információt, hogy a modul portjai mely FPGA lábakkal kerüljenek összeköttetésbe. Ezt az ***I/O ports*** ablakban tudjuk megadni (6. ábra), amely a ***Window*** menü ***I/O Ports*** menüpontjára kattintva jeleníthető meg. A felhasznált perifériák bekötését a kártya felhasználói útmutatója tartalmazza vagy ez leolvasható a kapcsolási rajzról is. A perifériák 3,3 V-ról működnek, ennek megfelelően mindegyik I/O porthoz LVCMOS33 I/O szabványt állítsunk be.

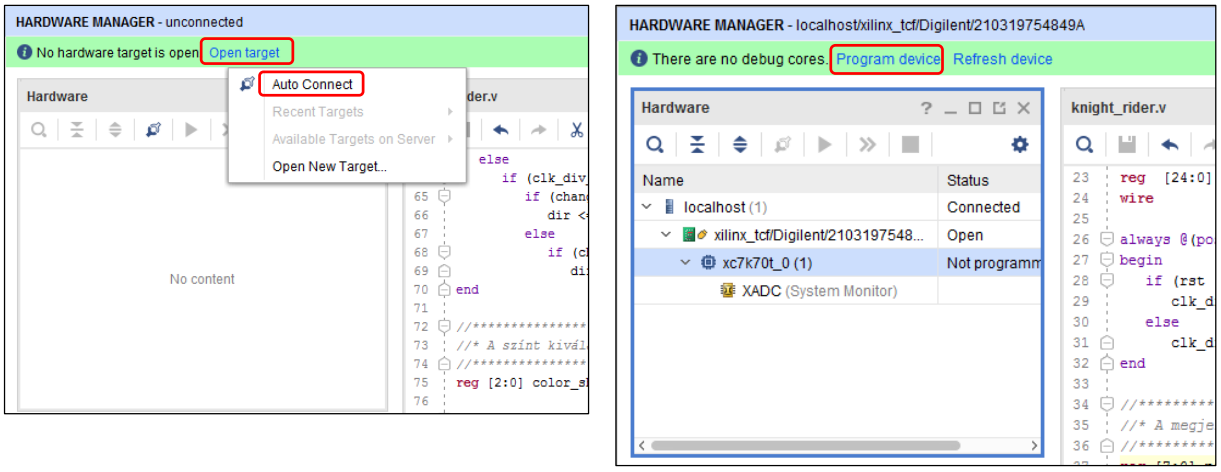
- 100 MHz-es oszcillátor: D23 I/O láb
- RST nyomógomb: L23 I/O láb
- Piros LED-ek: LD0-tól LD7-ig rendre az E16, E17, F19, C14, D15, C16, D18 és C18 I/O lábak
- Zöld LED-ek: LD0-tól LD7-ig rendre a G17, E18, G19, D14, D16, C17, D19 és C19 I/O lábak
- Kék LED-ek: LD0-tól LD7-ig rendre az F17, F18, F20, B15, B16, B17, B19 és D20 I/O lábak



6. ábra: Az I/O portok beállításai.

Miután az I/O portok beállításaival végeztünk, mentsük el az ezekre vonatkozó megkötéseket például ***knight_rider*** néven (az XDC kiterjesztés automatikusan hozzáadódik a névhez). A létrejött XDC fájl megjelenik a ***Sources*** panelen a ***Constraints*** csomópont alatt. A korábban leírt módon ismét futtassuk le a szintézist, majd ezután az implementációt és a konfigurációs fájl generálását az egyes fázisok végeztével felugró ablakban a ***Run Implementation*** és a ***Generate Bitstream*** opciók kiválasztásával. Amennyiben elrontottunk valamit és hibaüzenetet kapunk, akkor javítsuk ki a jelzett hibát.

Az FPGA konfigurációs fájl generálása után a felugró ablakban válasszuk ki az ***Open Hardware Manager*** opciót. A ***Hardware Manager*** (7. ábra) biztosít lehetőséget az FPGA eszköz felprogramozására. Miután megnyílt, csatlakoztassuk a kártyát USB porton keresztül a számítógéphez és kattintsunk az ***Open target*** feliratra, majd a megjelenő menüben válasszuk ki az ***Auto Connect*** menüpontot. Ekkor megtörténik a kártyán lévő eszköz azonosítása és elérhetővé válik annak felprogramozása a ***Program device*** feliratra kattintva. A megjelenő ablakban a konfigurációs fájl neve már megfelelően ki van töltve, itt csak kattintsunk a ***Program*** gombra.



7. ábra: A Hardware Manager.

Az FPGA sikeres felprogramozását a kártyán lévő zöld DONE LED kigyulladása jelzi és az RGB LED-eken megjelenik a futófény.