

1 FPGA áramkörök tesztelése

A laborgyakorlat célja az FPGA-n elkészített áramkörök tesztelése az FPGA-ba integrált analizátor alkalmazásával.

1.1 Bevezető

FPGA –ra készített áramkörök (kapcsolási rajzok) tesztelésére több lehetőség van:

- Áramkör működésének szimulációja, valamely szimulációs környezetben
- Külső méréstechnikai eszközök (oszilloszkóp, külső logikai analizátor) alkalmazása
- Az FPGA áramkörbe integrált logikai analizátor alkalmazása (áramkör működés közben való tesztelése)

Az áramkör szimulációjára több szimulációs környezet is alkalmazható. Az FPGA-ra elkészített áramkörrel a tervezési fázis egyes lépései után (minden fázisnak megfelel egy szimulációs modell) szimuláció végezhető el.

- behavioral simulation- viselkedési szimuláció szintezést követően
- functional simulation-funkcionális szimuláció fordítást követően
- static timing analysis- statikus időanalízis leképzési fázis után
- timing analysis- az elhelyezést és huzalozást követően
- a szimuláción kívül elvégezhető az áramkör teljesítmény analízise

A régebbi rendszerekre: VIRTEX V, Spartan 6 és előbbi FPGA családhoz tartozó FPGA áramkörökre a XILINX ISE Design Suite tervezőkörnyezetben található ISIM modult alkalmazzuk. A régebbi verziókban az ISIM szimulátorban a gerjesztő jeleket egy grafikus felületen is meg lehetett határozni, rajzolni. Az újabb verziókban nincs meg ez a lehetőség. A gerjesztő jeleket VHDL utasításokkal kell meghatározni. Az ISE környezetben a tervhez kell adni egy szimulációs modellt, amely egy minta VHDL állomány létrehozását eredményezi, amelyhez egyrészt hozzá van csatolva a szimulálandó modul, másrészt a gerjesztő jelek. Szekvenciális áramkör esetén az órajel generálásáért egy külön process a felelős. A gerjesztő jeleknek az időbeli változását, alakulását a szimulációt végző felhasználónak kell elkészítenie. Egyes esetekben lehetőség van a tesztjelek (próbapad) automatikusan történő létrehozására.

Az egymást követő jelek időbeli ütemezéséhez a wait on utasítás alkalmazására van szükség. Meghatározzuk a gerjesztő jeleket, majd wait on utasítással bevárjuk a szimulációs idődiagramnak megfelelő időt.

Az alábbi példa szemlélteti egy áramkör szimulációjának lépéseit.

Sok esetben az FPGA-ra elkészített áramkör szimulációja során helyes műkést tanúsít, azonban az FPGA áramkörbe való betöltést követően a tervezett áramkör hibás működését észleljük. A hibás modul (alegység), vagy hibásan működő rész felderítésében az FPGA-ba betöltött áramkör jeleinek működés közbeni analízálása vezet eredményre.

Az áramkör belső jeleinek a tanulmányozására, megjelenítésére használhatóak külső méréstechnikai eszközök, mint például oszcilloszkóp vagy külső logikai analizátor. Az oszcilloszkópon általában kevés számú jel (az oszcilloszkóp csatornáinak megfelelően) tanulmányozható, és általában a jelnek egy pillanatnyi szakaszában (keves mintavétel), az oszcilloszkóp kijelzőjén megjeleníthető.

Külső logikai analizátor alkalmazásával több csatornán, hosszabb ideig történik a mérés, az adatok egy tampon memóriába kerülnek. A mérés leállítását követően a jelek vagy a dedikált logikai analizátor kijelzőjén, vagy a számítógép kijelzőjén grafikusán vannak ábrázolva. A felhasználó a mért jel különböző részeit analizálhatja, kereshet a jelformában, több jelet csoportosíthat, amelyeket egységesen kezel, különböző formátumokban jelenítheti meg a mérési eredményeket. A külső logikai analizátor alkalmazásakor a vizsgálandó jeleket ki kell vezetni az FPGA áramkörből a fejlesztőrendszer megfelelő kimeneteire. Az analizátor csatlakozóját rá kell kötni az FPGA áramkör megfelelő kivezetéseire.

A következő és talán a leghatékonyabb lehetőség a XILINX tervezőkörnyezetben elérhető, hibaelhárításra szolgáló modulok alkalmazása és a számítógép képernyőjén e jelek megjelenítésére szolgáló eszközpár. Az ISE alapú tervezőeszközökben a Chip Scope Pro segítségével volt lehetőség a jeleknek a számítógépen való megjelenítésére.

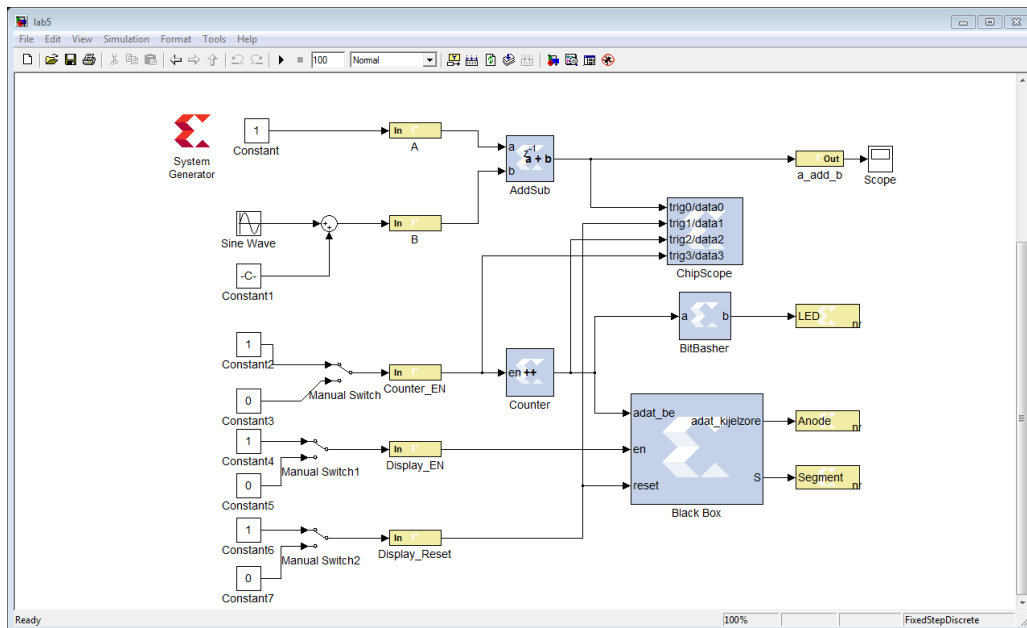
A Vivado tervezőeszközökben lényegében a ChipScope Pro modult a Vivado programba integrálták.

Ez a módszer egy Chipscope IP magnak az áramkörbe való integrálását jelenti. Ez esetben nem szükséges az analizálandó jeleket kivezetni az FPGA áramkörből, csak rá kell kapcsolni az analizátor modul bemeneteire.

Az adatok az FPGA áramkörből a JTAG láncon vannak kivezetve. A fejlesztőrendszertől függően a JTAG lánc általában USB protokollon kapcsolódik a számítógépre. A komplexebb FPGA fejlesztőpanelek esetében Ethernet csatlakozás is előfordul. A számítógépen az analizátor grafikusán megjeleníti a jeleket.

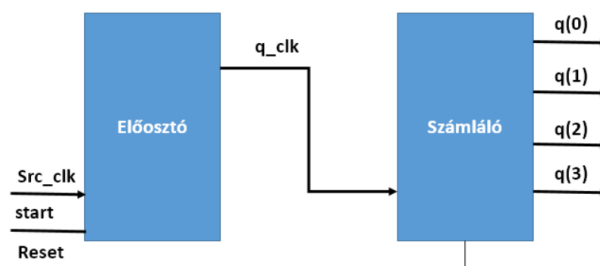
Az ISE alapú System Generator programban lehetőség volt integrálni a tervbe ChipScope Pro analizátor modult, és az analizálandó, tesztelendő jeleket rávezetni a tesztegységre.

ISE alapú System Generator 14.7-es változatában az alábbi példa szemlélteti a tesztelendő jeleknek a Chip Scope modulra való csatlakozását.



1.2 Tesztelendő áramkör

Tesztelendő modul- egy előosztóból és egy számlálóból van kialakítva a következő tömbvázlat szerint.



A VHDL forráskód az előosztó és számláló modulokra egy-egy processel van magvalósítva.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;
```

```
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
```

```
--use UNISIM.VComponents.all;
```

```
entity top_level_2 is
generic (
    div_val : natural:=256);
Port ( src_clk : in STD_LOGIC;
    --src_ce : in STD_LOGIC;
    reset : in STD_LOGIC;
    start : in STD_LOGIC;
    q : out STD_LOGIC_VECTOR (3 downto 0));
end top_level_2;
```

```
architecture Behavioral of top_level_2 is
```

```
signal q_clk: std_logic;
--tesztelésre a jelek megjelölhetőek a mark_debug attributummal
-- attribute mark_debug : string;
-- attribute mark_debug of reset: signal is "true";
-- attribute mark_debug of start: signal is "true";
-- attribute mark_debug of q: signal is "true";
-- attribute mark_debug of q_clk: signal is "true";
--attribute mark_debug of q_div : signal is "true";
```

```
begin
```

```
--számláló VHDL programkódja
szamlalo: process(q_clk, reset)
variable sz: std_logic_vector(3 downto 0);
begin
if reset='1' then
    sz:=(others=>'0');
elsif q_clk'event and q_clk='1' and start='1' then
    sz:=sz+1;
end if;
q<=sz;
end process szamlalo;
```

```
--előosztó VHDL programkódja
modulo:process(src_clk,reset)
```

```
variable x: integer range 1023 downto 0 := 0;
variable q_div: STD_LOGIC:= '0';
```

```

begin
if src_clk'event and src_clk='1' then
    if x<div_val then
        x:=x+1;
        q_div:=q_div;
    else
        x:=0;
        q_div:=not(q_div);
    end if;
    q_clk<=q_div;
end if;
end process modulo;
end Behavioral;

```

1.3 Működés közbeni tesztelés Vivado környezetben

A Vivado tervezőeszközben több lehetőség van, több módon lehet beállítani, konfigurálni a tesztkörnyezetet egy FPGA-ba betöltött áramkör működés közben való tesztelésére.

- Set Up Debug varázsló alkalmazása a teszteléshez szükséges tesztmodul (Debug Cores) tervbe való integrálására
- XDC utasítások alkalmazása a tesztelő modul (Debug Core) tervhez való kapcsolásához és konfigurálásához

Az áramkörben működés közben való tesztelés több munkafázist tartalmaz:

- 1) Tesztelendő jelek kijelölése: az áramkörből a tesztelendő jelek kiválasztása, analizálása, és tesztelésre való megjelölése.
 - a. A tesztelendő jeleknek a HDL forrásodban való megjelölése
 - b. A tesztelendő jeleknek a szintetizált áramkörben való megjelölése
 - c. a tesztelendő jeleknek tcl paranccsal való megjelölése
- 2) Implementációs fázis: a tesztelésre alkalmazott IP magokat és az IP magra csatlakoztatott tesztjeleket is tartalmazó terv implementációja és konfigurációs fájl létrehozása
- 3) Analízis fázis: az FPGA áramkörbe betöltött terv debug interfészére csatlakozva működés közben kiolvasásra és megjelenítésre kerülnek a tesztelésre kijelölt jelek. Analizálva a jelformákat, ellenőrizzük az áramkör működését

1.4 Tesztkörnyezet konfigurálása

1.4.1 Set Up Debug varázsló alkalmazása a teszteléshez szükséges tesztmodul (Debug Cores) tervbe való integrálására

Tesztelésre szánt jelek megjelölését követően a jeleket hozzá kell rendelni tesztelő IP modulhoz. A Vivado tervezőeszköz a **Set Up Debug** varázslóval egy egyszerű felületet biztosít a tesztelendő jeleknek a tesztelendő modulhoz való csatolásában, automatikusan létrehozva a tesztelő magokat, hozzárendelve a tesztelendő jeleket a teszt IP mag bemenetére.

A teszt modul hozzáadás több fázisban történik:

- a szintézist követően a tesztelésre szánt vezetékek kijelölése
 - a. az unassigned nets listából a jelek kiválasztása
 - b. a szintetizált áramkörben a jelek megjelölése (jobb klikk a vezetékekre a Synthesized Design nézetben, vagy a Netlist listából és a Mark Debug paranccsal megjelöljük tesztelésre)
- Set Up Debug parancs végrehajtása
 - a. A Flow navigator ablakban, Synthesis -> Synthesized Design majd Set Up Debug vagy
 - b. Tools menüpontban Set Up Design
 - c. Window menüpontból a Debug nézetben a tesztelendő jelek kijelölése az Unassigned Debug Nets listából és a Set Up Debug parancs kiadása (jobb klikk a kijelölt jelekre, majd Set Up Debug)
- Ha szükséges, újabb jeleket csatlakoztathatunk a tesztmodul bemenetéhez, vagy eltávolíthatunk a Find Nets to ADD-re kattintva
- Az órajel kiválasztása, amelyet majd alkalmaz a tesztmodul a jel mintavételezésére
- Speciális trigger mód és capture mode beállítása
- Ha úgy látjuk, hogy helyesen kijelöltük a tesztjeleket, a Finish gombra kattintva hozzáadjuk és csatoljuk az ILA modult a tervhez
- ILA maggal kapcsolatos paraméterek beállítása
 - a. C_DATA_DEPTH
 - b. C_INPUT_PIPE_STAGES
 - c. C_EN_STRG_QUAL
 - d. C_ADV_TRIGGER
- A tesztelésre szánt jelek (debug nets) a rajz szerint hozzá vannak csatolva a tesztmodulhoz

1.4.2 XDC utasítások alkalmazása a tesztelő modul (Debug Core) tervhez való kapcsolásához és konfigurálásához

#TCL parancsokat alkalmazva a terv a következőképpen hozható létre:

```
set terv_nev teszt_7
set konyvtar C:/Munka/I_Felev/UKDA_2017/L6
start_gui
create_project ${terv_nev} ${konyvtar}/${terv_nev} -part xc7z010clg400-1
```

#VHDL hardver leíró nyelv beállítása

```
set_property target_language VHDL [current_project]
```

```
#VHDL forráskódnak a tervhez való csatolása
```

```
add_files -norecurse ${konyvtar}/top_level_2.vhd
```

```
#Megkötés állománynak a tervhez való csatolása
```

```
add_files -fileset constrs_1 -norecurse ${konyvtar}/system_4.xdc
```

```
update_compile_order -fileset sources_1
```

```
update_compile_order -fileset sim_1
```

```
#Szintézis futtatása
```

```
launch_runs synth_1
```

```
#Várakozás a szintézis befejezésére
```

```
wait_on_run synth_1
```

```
#Synthesized Design megnyitása
```

```
open_run synth_1
```

```
#Debug core magnak a tervhez való csatolása
```

```
create_debug_core u_ila_0 ila
```

```
#ILA modul konfigurálása
```

```
set_property C_DATA_DEPTH 65536 [get_debug_cores u_ila_0]
```

```
set_property C_TRIGIN_EN true [get_debug_cores u_ila_0]
```

```
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
```

```
set_property C_ADV_TRIGGER true [get_debug_cores u_ila_0] #enable advanced trigger mode
```

```
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
```

```
set_property C_EN_STRG_QUAL true [get_debug_cores u_ila_0] #enable Basic capture mode
```

```
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
```

```
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
```

```
set_property port_width 1 [get_debug_ports u_ila_0/clk]
```

```
#ILA modul órajelének meghatározása
```

```
connect_debug_port u_ila_0/clk [get_nets [list src_clk_IBUF_BUFG]]
```

```
#A Test modulon probe0 sínszélességének meghatározása
```

```
set_property port_width 3 [get_debug_ports u_ila_0/probe0]
```

```
#probe0 bemenetre a start, reset és q_div jelek csatolása.
```

#A forráskódban meghatározott jelek nem érhetőek el tesztelésre a szintézist követően, hanem a bemenetek esetében a start_IBUF reset_IBUF jeleket alkalmazzuk, kimenetek esetében pedig a q_OBUF jeleket. .

connect_debug_port u_ila_0/probe0 [get_nets [list start_IBUF reset_IBUF q_div]] jeleket kell alkalmazni

#Egy újabb bemenet (probe1) létrehozása a tesztmodul bemenetére

create_debug_port u_ila_0 probe

#Sínszélesség beállítása

set_property port_width 4 [get_debug_ports u_ila_0/probe1]

#q_OBUF[0], q_OBUF[1], q_OBUF[2], q_OBUF[3] kimeneteknek a probe1 bemeneti portra való csatolása

connect_debug_port u_ila_0/probe1 [get_nets [list q_O*]]

#Megkötés fájl mentése

save_constraints_as system_\${terv_nev}.xdc

set_property constrset system_teszt_6.xdc [get_runs synth_1]

set_property constrset system_teszt_6.xdc [get_runs impl_1]

#Implementáció lefuttatása

launch_runs impl_1

wait_on_run impl_1

#Konfigurációs fájl generálása

launch_runs impl_1 -to_step write_bitstream

wait_on_run impl_1

#Hardver megnyitása

open_hw

#Szerver indítása az FPGA áramkörre való kapcsolódás céljából

connect_hw_server

#Célhardver megnyitása

open_hw_target

#Konfigurációs .bit fájl és debug_net.ltx beállítása

set_property PROGRAM.FILE

\${konyvtar}/\${terv_nev}/\${terv_nev}.runs/impl_1/top_level_2.bit [lindex [get_hw_devices] 1]

set_property PROBES.FILE

\${konyvtar}/\${terv_nev}/\${terv_nev}.runs/impl_1/debug_nets.ltx [lindex [get_hw_devices] 1]

current_hw_device [lindex [get_hw_devices] 1]

#hardver frísítése

refresh_hw_device [lindex [get_hw_devices] 1]

#Hardver programozása

```
program_hw_devices [lindex [get_hw_devices] 1]
```

```
refresh_hw_device [lindex [get_hw_devices] 1]
```

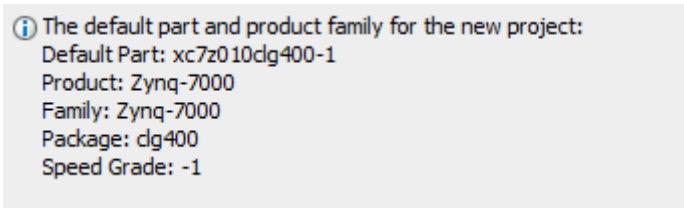
A tcl szkript nyelven megadott programrészt lefuttatjuk az első fázistól az utolsóig, minden lépés automatikusan végrehajtódik, az FPGA áramkörre feltöltődik a konfigurációs fájl és csatlakozik a grafikus vizualizációs felülethez. A felprogramozott eszköz készen áll a tesztelés elvégzésére.

Tesztkörnyezet szemléltetése Vivado 2016.2 tervezőprogram alkalmazására egy egyszerű VHDL alapú áramkör tesztelésére.

1.5 Áramkör működés közben való tesztelése és ennek lépései

A példa során egy egyszerű áramkör (mely két számlálót tartalmaz) működés közben való tesztelését vezetjük végig.

- 1) A top_level.vhd modul alapján egy terv létrehozása (2. labor alapján)
 - a. új terv létrehozása
 - b. könyvtár kiválasztása
 - c. top_level.vhd állománynak a tervhez való csatolása
 - d. system.xdc megkötés állománynak a tervhez való csatolása
 - e. újrakonfigurálható áramkör típusának kiválasztása



The screenshot shows a dialog box titled "The default part and product family for the new project:". It lists the following information: Default Part: xc7z010clg400-1, Product: Zynq-7000, Family: Zynq-7000, Package: clg400, and Speed Grade: -1.

- 2) A tesztelendő jelek kiválasztása, a debug interfésznek a tervbe való csatolása, a jeleknek a tesztmodulba való csatolása
 - a. Kijelölt tesztelendő jelek
 - i. reset
 - ii. start
 - iii. q-kimenet
 - iv. q_div –előosztóban alkalmazott számláló
 - v. q_clk –leosztott órajel
 - b. a tesztelendő jelek bejelölése a VHDL forráskódban szintézis előtt
 - -attribútumokkal való bejelölés

```

attribute mark_debug : string;
attribute mark_debug of reset: signal is "true";
attribute mark_debug of start: signal is "true";
attribute mark_debug of q: signal is "true";
attribute mark_debug of q_clk: signal is "true";

```

- a tesztelésre kijelölt jelekkel az áramkör szintézise (a példa szerint a Tcl parancssorból indítva)

```

launch_runs synth_1
wait_on_run synth_1

```

- Setup Debug parancs kiadása
 - A Flow navigator ablakban, Synthesis, Synthesized Design majd Set Up Design vagy
 - Window menüpontból a Debug nézetben a tesztelendő jelek kijelölése az Unassigned Debug Nets listából és a Set Up Debug parancs kiadása

c. Set Up Debug parancs kiadása a Debug nézetből

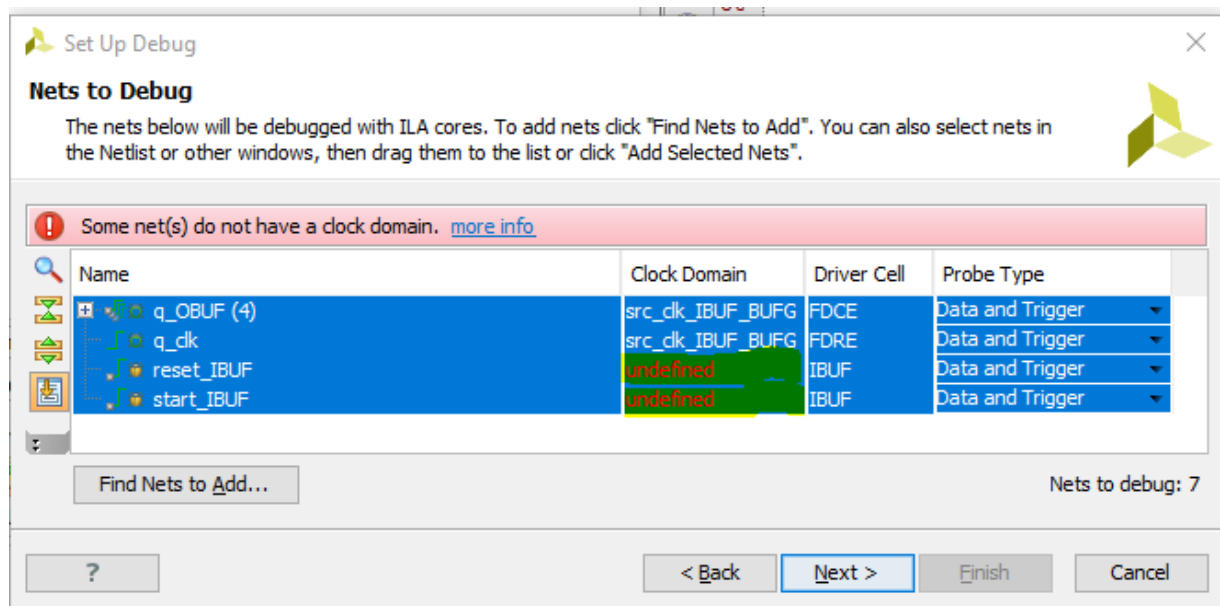
jobb kattintás az Unassigned Debug Nets listából a kijelölt tesztelendő jelekre

Name	Driver Cell	Driver ...	Probe Type
Unassigned Debug Nets (7)			
q_OBUF (4)	FDCE	Q	
q_clk	FDRE	Q	
reset_IBUF	IBUF	O	
start_IBUF	IBUF	O	

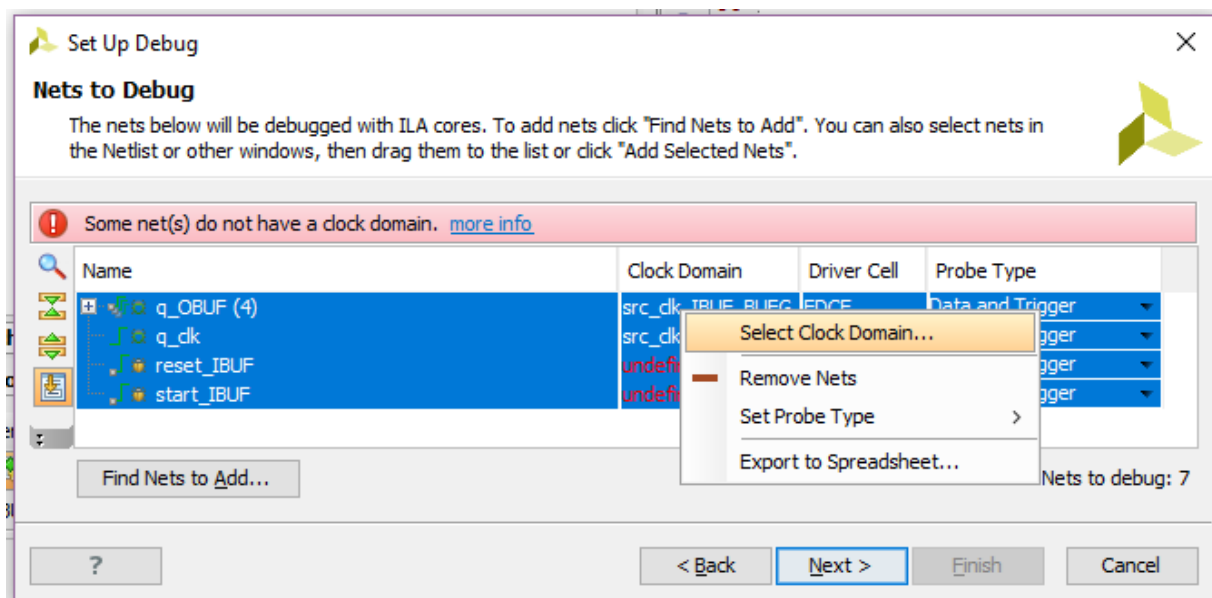
-Set Up debug parancs végrehajtása, majd első lépést követően ugrás a következő fázisra (Next)

3) Órajel tartományok kiválasztása

Minden egyes tesztelendő jelre meg kell határozni, hogy melyik órajel tartományhoz tartozik. Ebben az esetben a start és a reset jelekre automatikusan nem sikerült a tervezőeszköznek órajelet hozzárendelni, ezt a lépést most el kell végezni. Az órajel határozza meg, hogy milyen gyakran mintavételezzük a tesztelés során a jeleket.

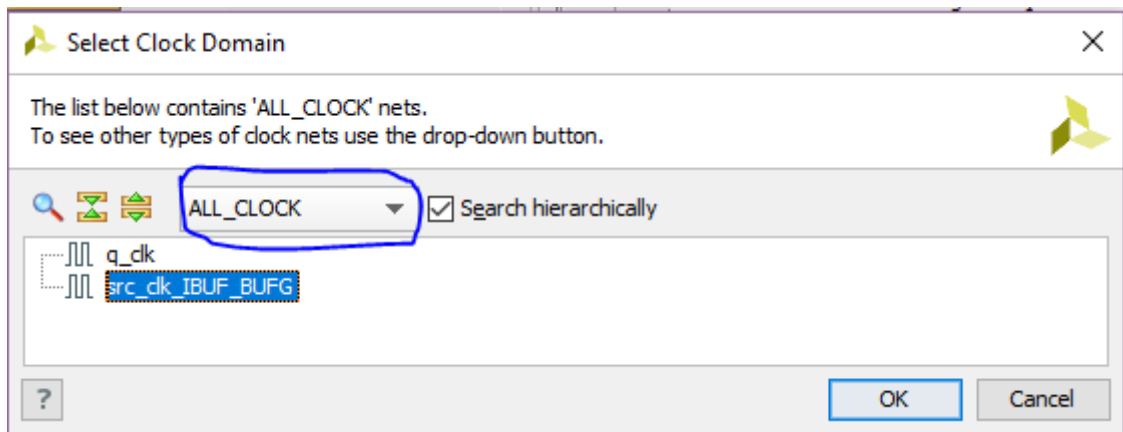


Azon jelek esetében, amelyekhez nincs órajelel hozzárendelve, vagy azon jelek esetében, amelyekre egy új órajelet szeretnénk kiválasztani, jobb kattintás a jelnek megfelelő Clock Domain részre, majd a gyorsmenüből a Select Clock Domain kiválasztása.



A Select Clock Domain ablakban egy listából ki lehet választani, hogy melyik órajelet szeretnénk alkalmazni. A bejelölt legördülő listából ki lehet választani, hogy milyen típusú órajelet szeretnénk alkalmazni (GLOBAL_CLOCK, LOCAL_CLOCK, REGIONAL_CLOCK, ALL_CLOCK)

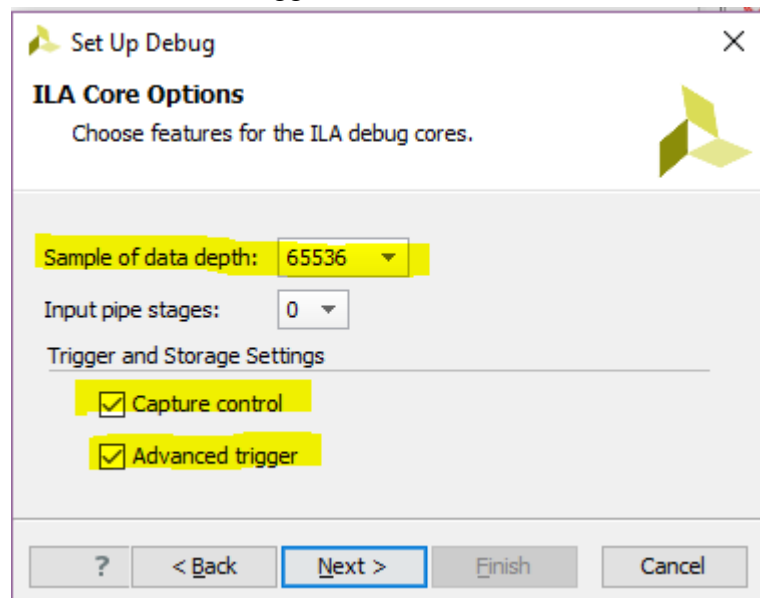
A listában a src_clk_IBUF_BUFG a rendszer globális órajele, míg a q_clk a tesztelendő áramkör két alegységét összekötő lokális órajele.



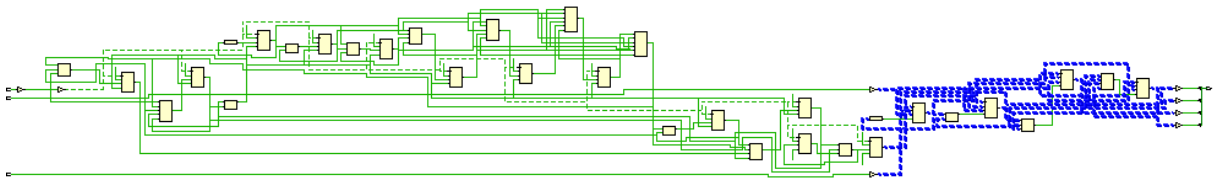
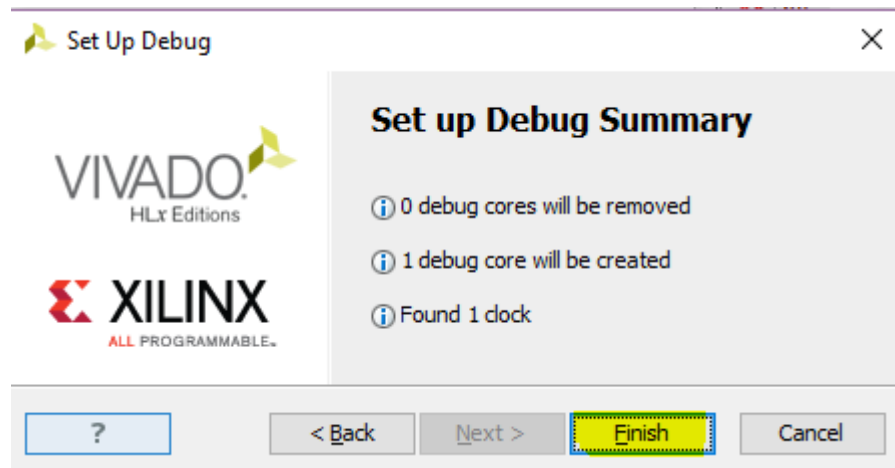
4) A tesztelő ILA IP modul paraméterezése

A következő ablakban (ILA Core Optins) a tesztelésre alkalmazott ILA IP mag paraméterezhető:

- egy mérés során beolvasható minták száma (sample of data depth)
- Input pipe stages,
- Capture control
- Advanced trigger



Az utolsó ablakban a terv jelenlegi állapotáról, a hozzáadott debug core modulokról, illetve az alkalmazott órajelek számáról jelenít meg információt. A Finish gombra kattintva befejezzük a Set Up Debug parancs konfigurálását.



5) A terv implementálása

Implementáció elindítása a tcl parancsablakból a következő utasítással:

```
launch_runs impl_1
```

6) Konfigurációs fájl létrehozása

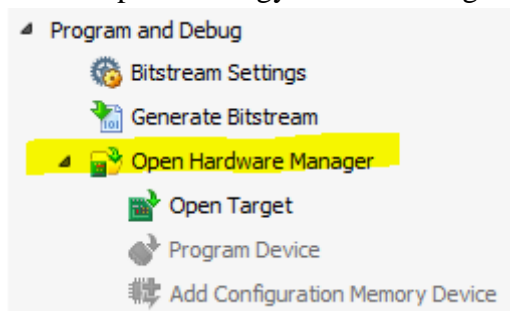
A konfigurációs fájl a tcl parancsablakból a következő utasítással hajtható végre:

```
launch_runs impl_1 -to_step write_bitstream
```

7) Hardver menedzser megnyitása. Ezen a felületen van lehetőség kapcsolódni a hardvereszközhöz.

Hardver manager megnyitása

open_hw vagy a Flow Navigator panelen az Open Hardware Manager menüponttal



A Logikai terv hardverben való tesztelése, alkalmazva a Vivado Logic Analyzer analízátort

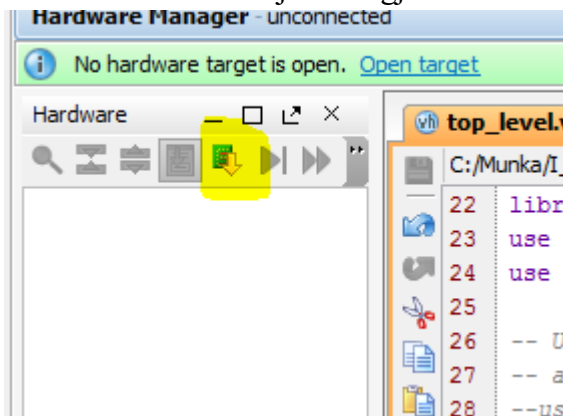
A Vivado Analyzer eszköz kapcsolódik a tervben elhelyezett tesztelő modulokkal: ILA, VIO és JTAG-to AXI. A logikai analizátor funkciók a Flow Navigator ablakon Program and Debug, majd Hardware manager menüpontra kattintva érhető el.

1.6 A tesztelés lebonyolítása

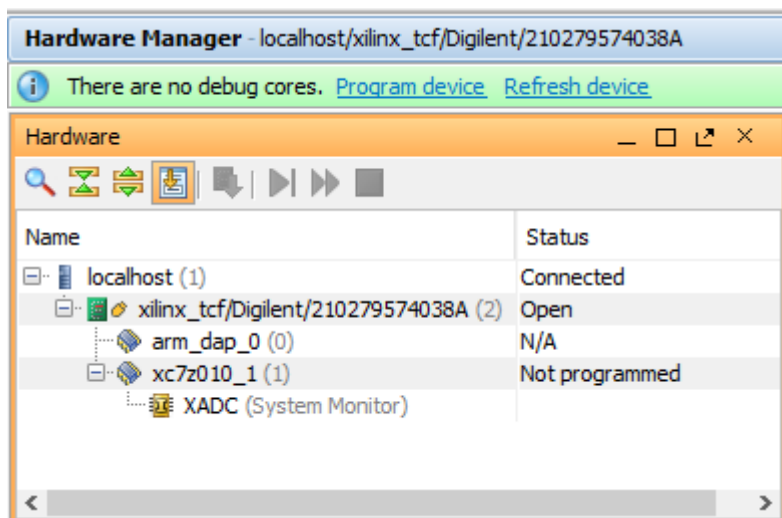
A fontosabb lépések, amelyek szükségesek a hardver teszt megvalósításához:

1.6.1 -a cél FPGA áramkörnek a rendszerhez való csatolása

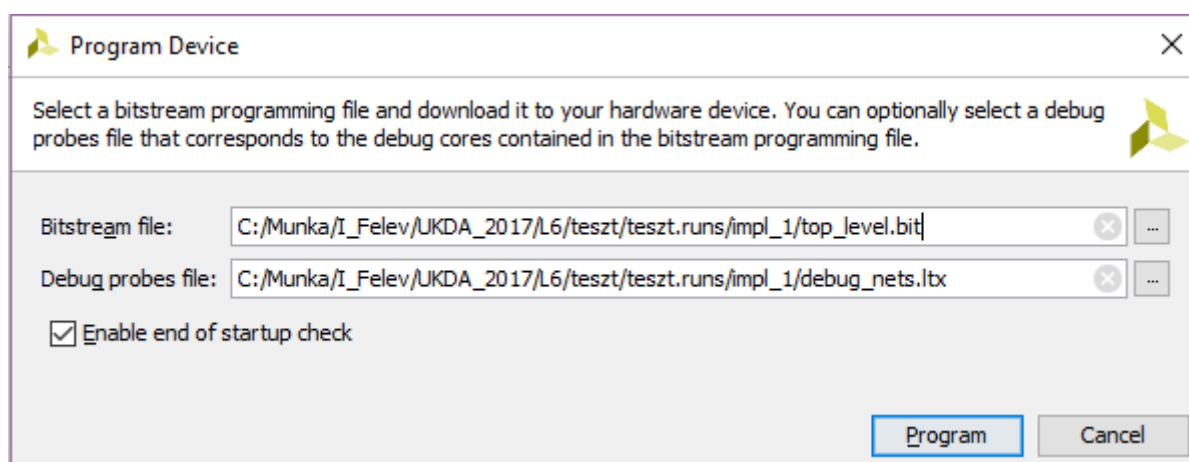
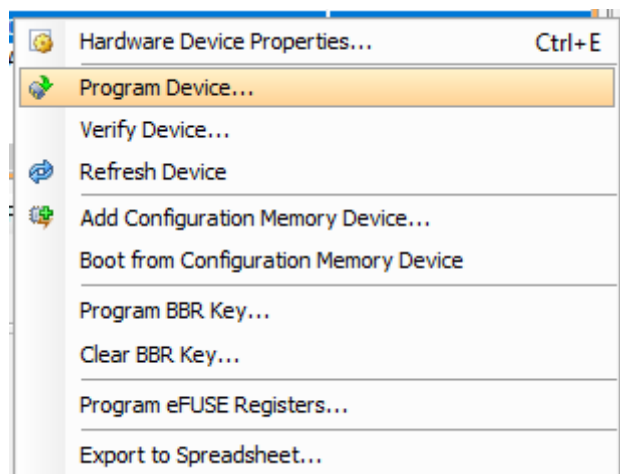
Jelen esetben a Zybo fejlesztőlapot az USB porton kapcsoljuk a számítógéphez, majd kapcsolódunk az FPGA áramkörre az alábbi rajzon megjelölt **Auto Connect** gombra kattintva



Ha sikeres a kapcsolódás az alábbi rajz alapján, a rendszer felismeri a fejlesztőlapot, az FPGA áramkör típusát, valamint az FPGA áramkörön belül az XADC analóg-digitális átalakítót, amelynek segítségével monitorizálni lehet például az áramkör hőmérsékletét.

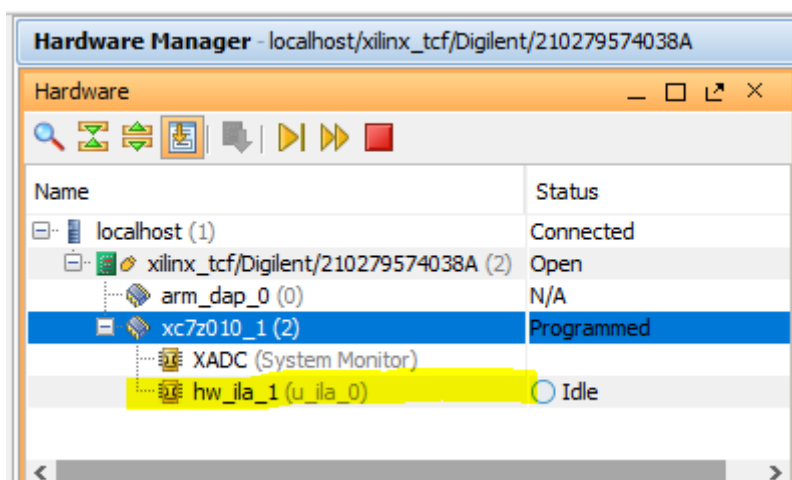


A következő lépésben fel kell programozni az FPGA áramkört Program Device utasítással. Az FPGA áramkör típusára kattintva (jobb klikk) a gyorsmenüből is ki lehet választani a **Program Device** parancsot.



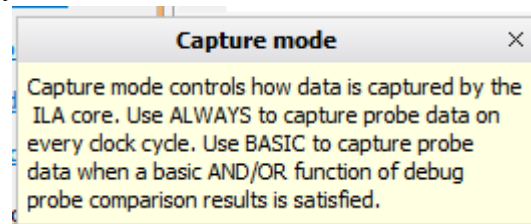
Programozáskor meg kell határozni a konfiguráció bit fájlt, amelyet be szeretnénk tölteni az FPGA áramkörre, valamint a tesztelésre kijelölt vezetékek a debug_nets.ltx állományban érhetőek el.

Az FPAG áramkör programozását követően, ha helyesen konfiguráltuk tesztelésre a tervet, az FPGA áramkör alatt elérhetők a teszt modulok



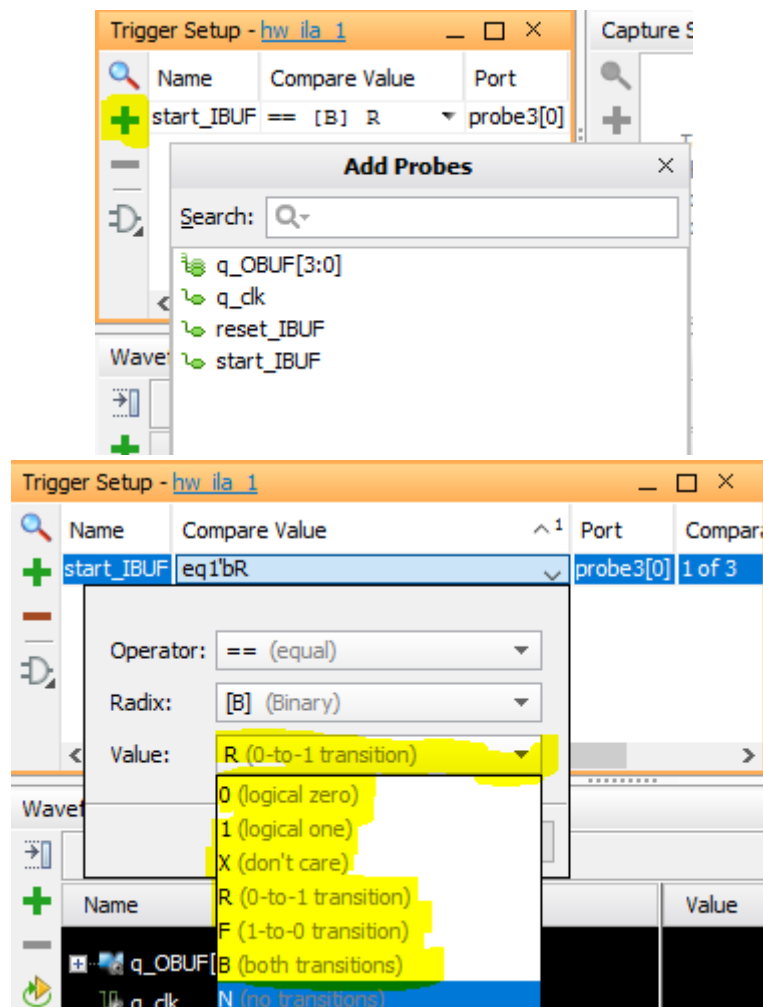
1.6.2 Az ILA tesztmodul trigger módjának beállítása illetve capture control beállításai

Ki kell választani, hogy mely események szeretnénk, hogy élesítsék, triggereljék az ILA tesztmodult, valamint milyen feltételek esetében történik mintavételezés

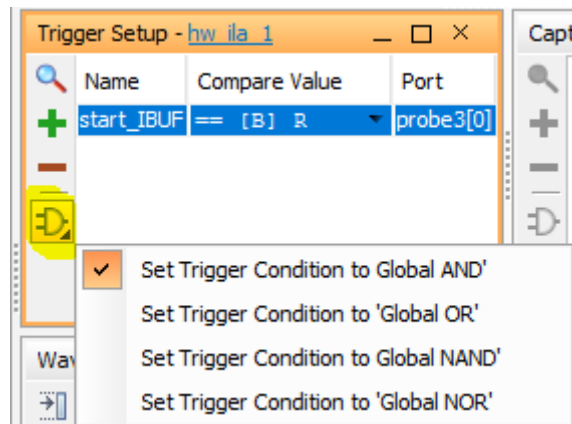


Az ILA modult aktiváló jelek a Trigger Setup panelen belül konfigurálhatók:

A tesztelésre szánt jelek kijelölésekor meg kell határozni, hogy milyen formában szeretnénk alkalmazni a tesztjelet: Trigger, Data vagy Trigger and data. A + ADD Probes gombra kattintva a megjelenített listából ki lehet választani azokat a jeleket, amelyeket szeretnénk, hogy élesítsék az ILA modult, valamint milyen esetben történjen az élesítés: logikai 0 szint, logikai 1 szint, felfutó élre, lefutó élre, vagy mindkét élre (lefutó, felfutó)



A **Set trigger condition** gombra kattintva, abban az esetben, ha több jel kombinációjával szeretnénk aktiválni az ILA modult, meg lehet határozni, hogy milyen logika szerint származtatjuk az aktiváló eseményt.



Az adatgyűjtés elkezdése

A mérési folyamat a következő fázisokból áll:

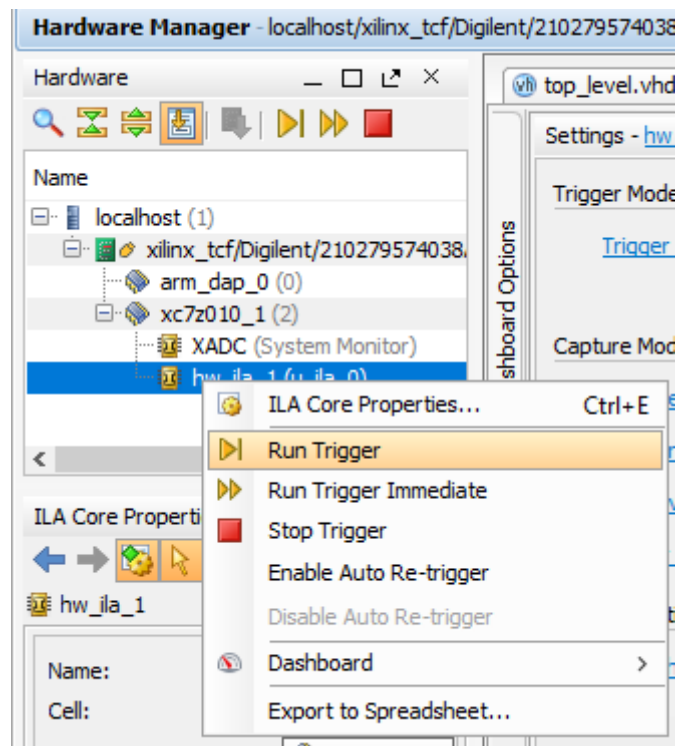
1.6.3 A mérés elkezdése az ILA modulon IDLE állapotban

Az adatgyűjtési szakasz fázisai:

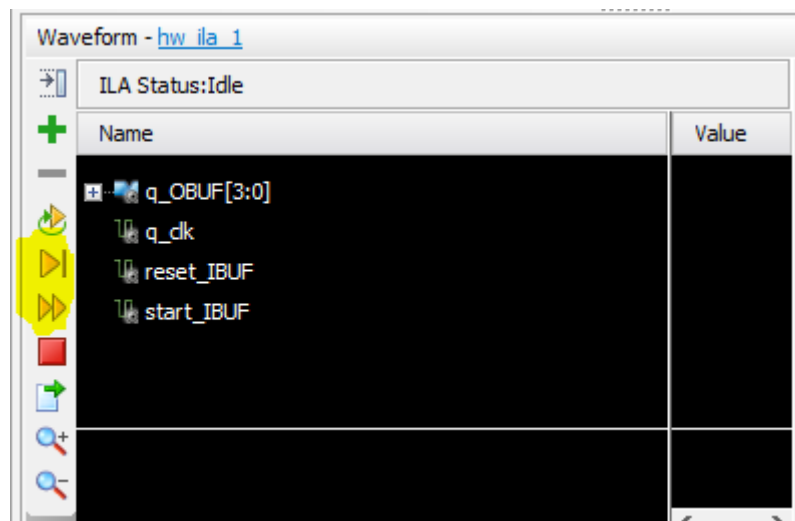
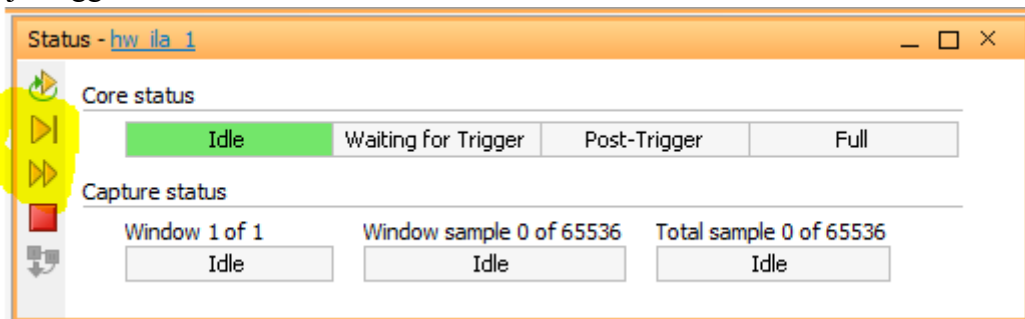
- A mérőmodul készenléti állapotban várakozik a mérés elkezdésére.
- Wait for trigger. A mérőmodul működik és várakozik, hogy az élesítő események bekövetkezzenek.
- Post-trigger. Teljesültek az élesítő feltételek és elkezdődik az adatgyűjtés.
- Full. Az adatok megvannak jelenítve a Waveform ablakban.

A ILA modul IDLE készenléti állapotban várja az adatgyűjtés engedélyezését. A mérés indítására több lehetőség van:

- a) a hardware manager részben jobb klikk a hw_ila_1 modulra, majd Run Trigger, vagy Run Trigger Immediate. Az Enable Auto Re-trigger opció bekapcsolásával automatikusan újraindul az ILA modul.

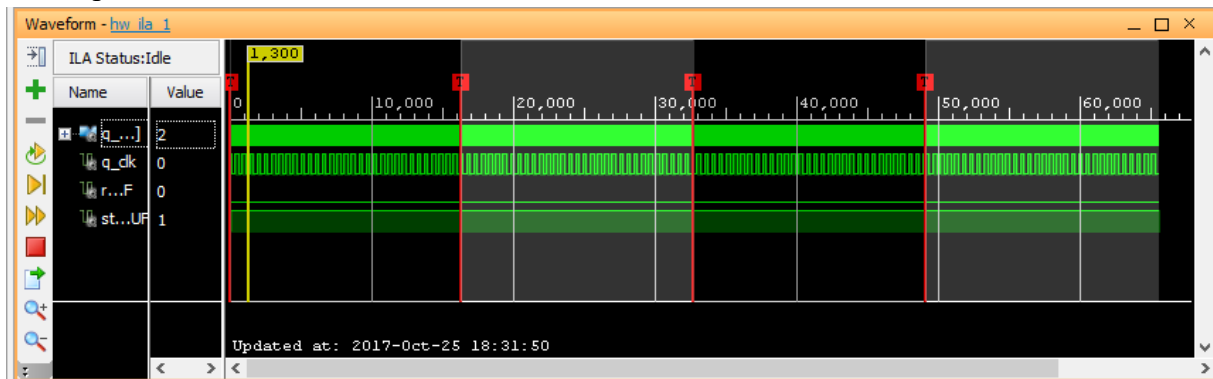


- b) Status vagy waveform panelekből a **Run Trigger**, vagy **Run Trigger Immediate** parancsokra kattintva. A Run Trigger Immediate esetben azonnal megtörténik a mintavételezés, míg a Run Trigger esetben várja, hogy teljesüljön az élesítő feltétel. A Status és Waveform ablakokból Toggle Auto Re-trigger mode opcióval lehet engedélyezni az újratriggerezést.



Az adatgyűjtést követően az ablakok meg vannak jelenítve a Waveform ablakban.

A puffer több kisebb méretű ablakra osztható, mindenik ablak a saját trigger markerével. Az alábbi példában 4 ablakban történik a mintavételezés



Analizálva az elért eredményeket, le kell vonni következtetéseket a tesztelt áramkör működésével kapcsolatosan. A feltárt hibákat ki kell javítani, és a teszteléseket újra elvégezni, mindaddig, amíg az elvárt működés megvalósul