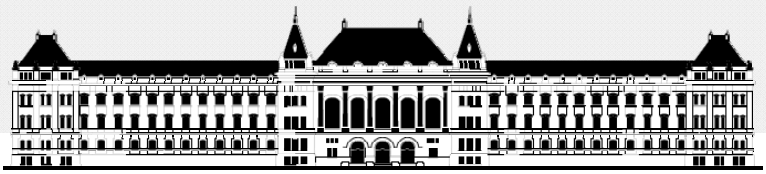


BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Rendszerarchitektúrák labor
Xilinx EDK

Raikovich Tamás

BME MIT

Labor tematika (Xilinx EDK)

- **1. labor:**
  - A Xilinx EDK fejlesztői környezet ismertetése
- **2. labor:**
  - Egyszerű processzoros rendszer összeállítása
  - Egyszerű szoftver alkalmazások készítése
- **3. labor:**
  - Saját periféria illesztése
  - Megszakításkezelés
  - HW/SW együttes fejlesztés (debugger, ChipScope)

Témakörök

- **Beágyazott rendszerek**
- **MicroBlaze processzor**
- **EDK alapok**
- **Gyári és saját IP-k hozzáadása**
- **Szoftverfejlesztés**
- ***HW és SW együttes fejlesztése***

A ChipScope logikai analizátor

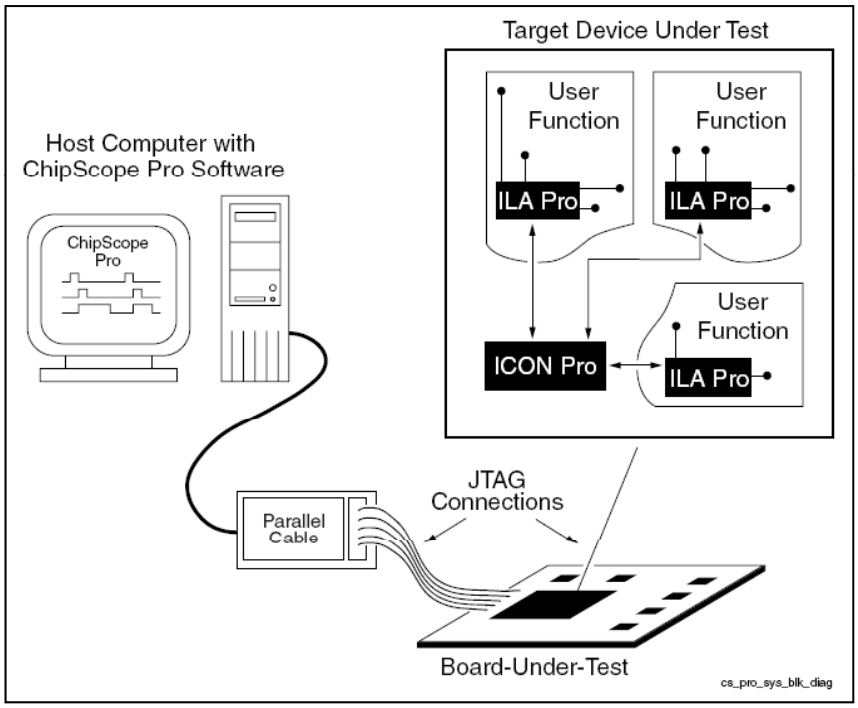
A ChipScope egy, az FPGA tervbe integrálható logikai analizátor, amely az FPGA terv belső jeleinek vizsgálatára használható

- **Előnye a normál logikai analizátorhoz képest**
  - Az igényeknek megfelelően konfigurálható
  - Nem kell kivezetni a jeleket FPGA I/O lábakra
    - Nem mindig van erre a célra elegendő I/O láb
- **Hátránya a normál logikai analizátorhoz képest**
  - FPGA erőforrásokat használ
  - Kevesebb erőforrás marad a terv számára
    - Kisebb kapacitású eszközök esetén korlátozottabb funkció

A ChipScope logikai analizátor

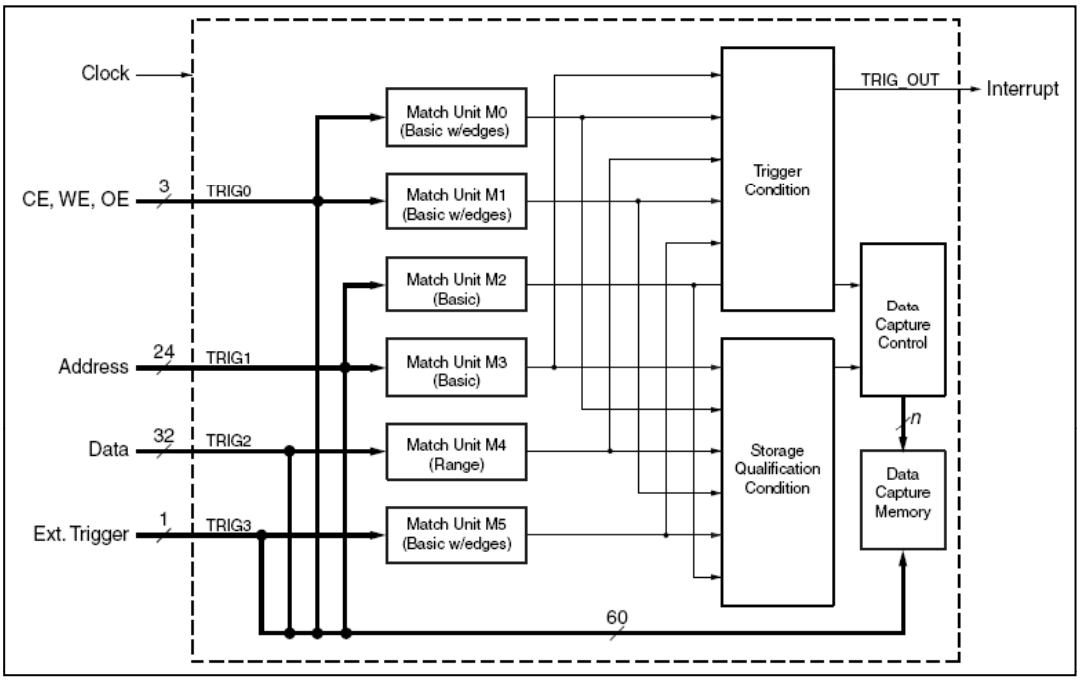
A ChipScope rendszer felépítése:

- **ICON**
  - JTAG kommunikáció
  - Vezérlés
- **ILA**
  - Logikai analizátor
- **ChipScope Pro Analyzer SW**



A ChipScope logikai analizátor

A ChipScope ILA bekötési példa (busz monitorozás):



A ChipScope logikai analizátor

Match Unit: a vizsgálandó jeleket összehasonlítja a megadott feltételekkel

- **Basic: =, <> műveletek**
- **Advanced: =, <>, >, >=, <, <= műveletek**
- **Range: az Advaced műveleteken felül**
  - "tartományon belül esik" művelet (in range)
  - "tartományon kívül esik" művelet (not in range)
- **Mindhárom típus detektálhat éleket is**
  - Basic with edges, Advanced with edges, Range with edges
- **Eseményszámlálók:**
  - Pontosan n előfordulás
  - Legalább n előfordulás
  - Legalább n egymást követő (folyamatos) előfordulás

A ChipScope logikai analizátor

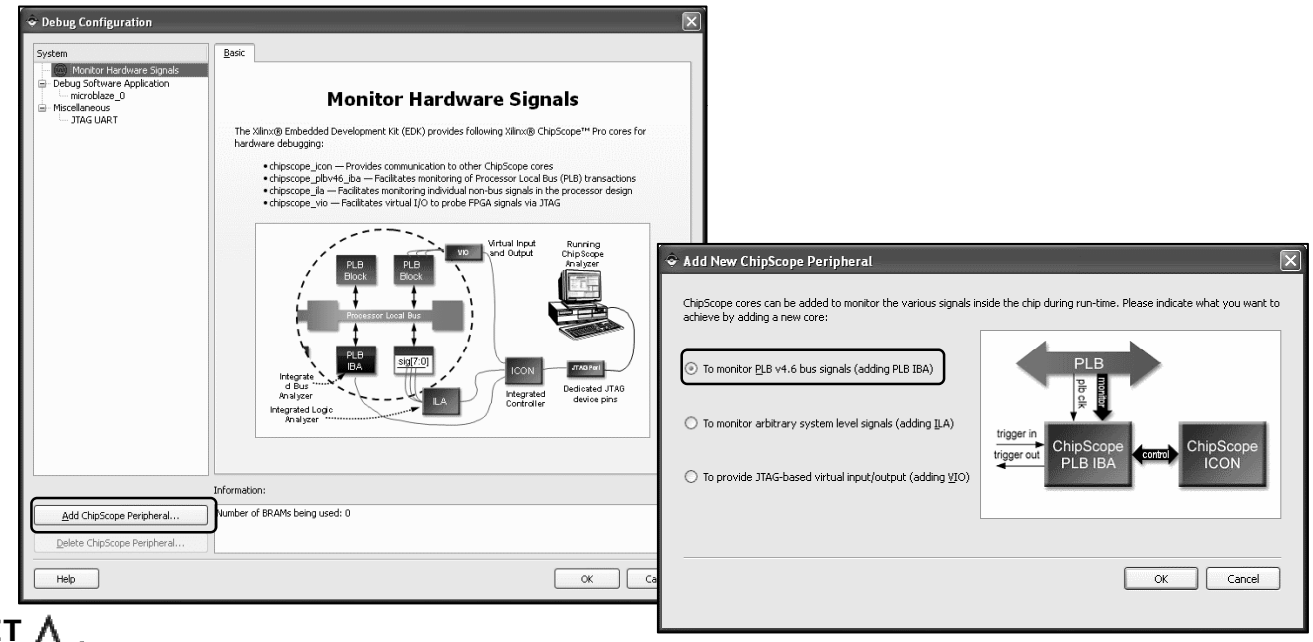
- **Trigger feltétel (Trigger Condition)**
  - A Match Unit események logikai vagy sorrendi kombinációja
  - Kijelöli a kezdőpontot a mintavételezett adatokat tartalmazó ablakban
- **Tárolási feltétel (Storage Qualification Condition)**
  - A Match Unit események logikai vagy sorrendi kombinációja
  - Eldönti, hogy kell-e tárolni az adott mintát

BME-MIT
FPGA labor

A ChipScope logikai analizátor

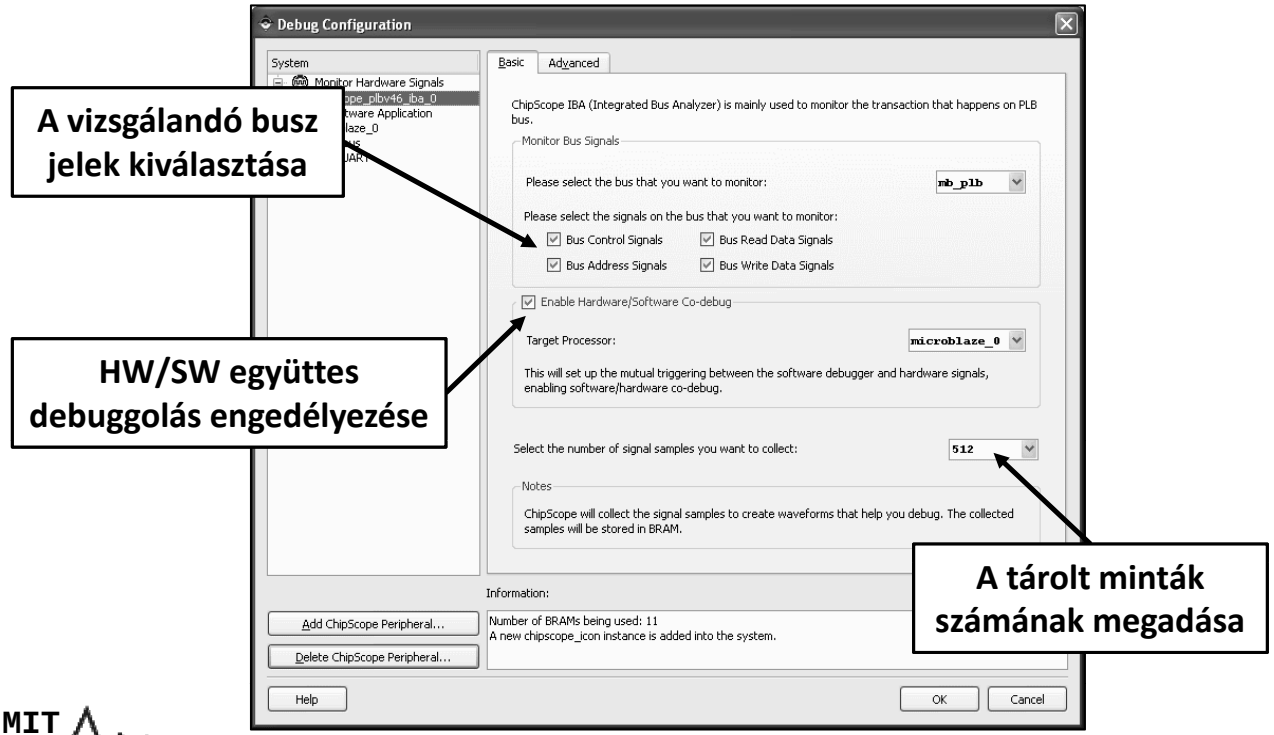
ChipScope beillesztése a processzoros rendszebe:

- ***Debug* menü → *Debug configuration...* → *Add ChipScope Peripheral...***
- **A PLB busz monitorozására fogjuk használni: PLB IBA hozzáadása**



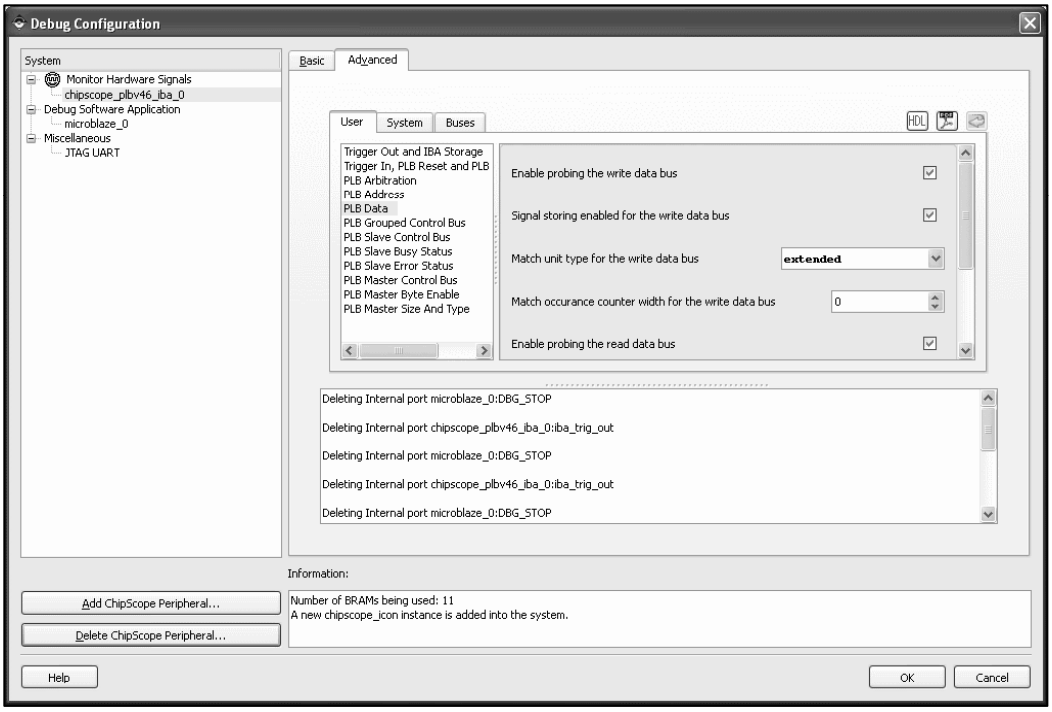
A ChipScope logikai analizátor

ChipScope alap beállítások:



A ChipScope logikai analizátor

ChipScope haladó beállítások:

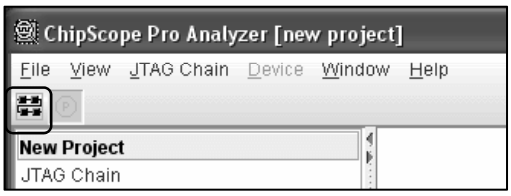


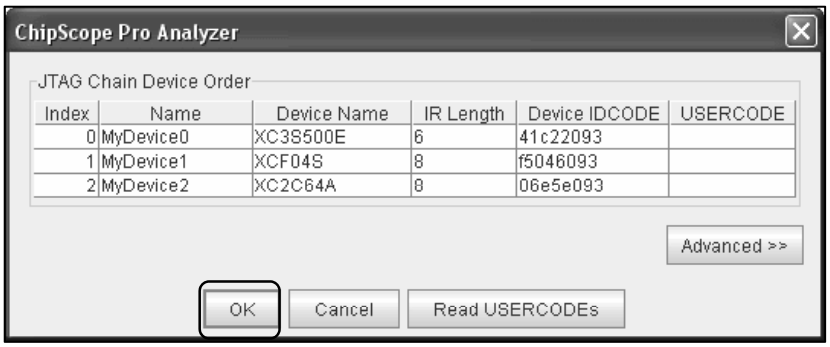
A ChipScope logikai analizátor

- **HW és SW együttes debuggolása**
  - A ChipScope trigger kimenete le tudja állítani a program futását: töréspont a debuggerben
  - A debuggerben beállított töréspontra futás trigger eseményként szolgál a logikai analizátornak
- **Új elemek a rendszerben**
  - ***chipscope_icon_0***: JTAG kommunikáció, vezérlés
  - ***chipscope_plbv46_iba_0***: PLB busz analizátor
- **A ChipScope hozzáadása után újra kell generálni a huzalozási listát és a konfigurációs bitfolyamot**

A ChipScope logikai analizátor

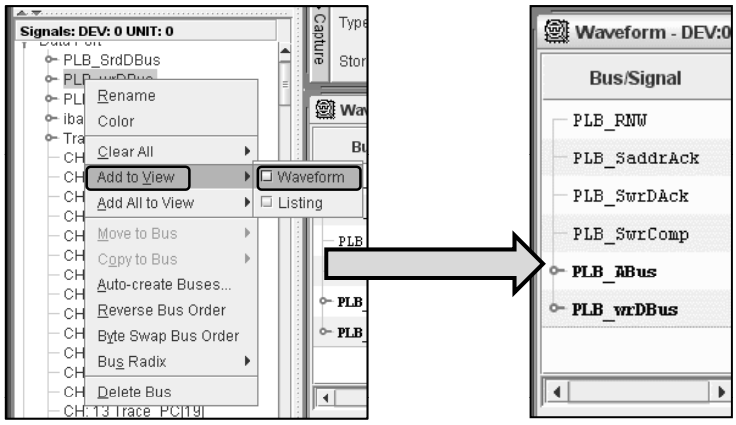
- **Indítsuk el a ChipScope Pro Analyzer programot**
- **Kattintsunk az *Open Cable/Search JTAG Chain* gombra**
  - A JTAG láncban található eszközök azonosítása
- **Kattintsunk az *OK* gombra: elindul az analizátor**
  - Az alapértelmezett trigger beállításokkal
  - Az alapértelmezett hullámforma beállításokkal





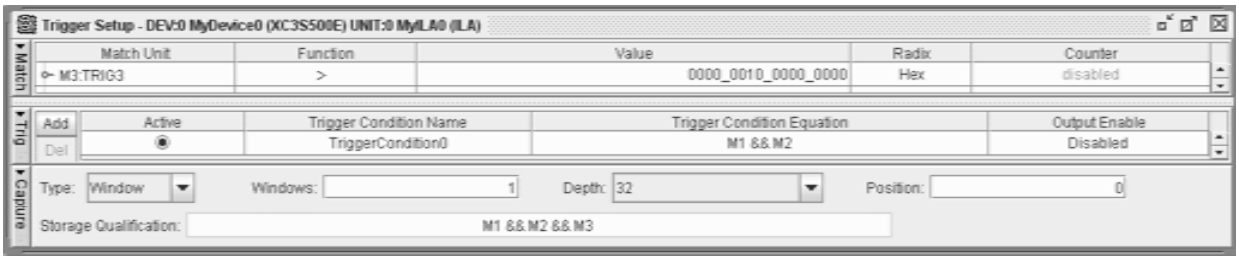
A ChipScope logikai analizátor

- **A PLB busz jelek importálása**
  - ***File*** menü **→ *Import* → *Select New File***
  - ***implementation\chipscope_plbv46_iba_0_wrapper\chipscope_plbv46_iba_0.cdc*** fájl megnyitása
- **Töröljük a meglévő jeleket a hullámforma ablakban**
- **Adjuk hozzá a szükséges PLB jeleket a hullámformához**
  - Jobb klikk a jelen **→ *Add to View* → *Waveform***



A ChipScope logikai analizátor

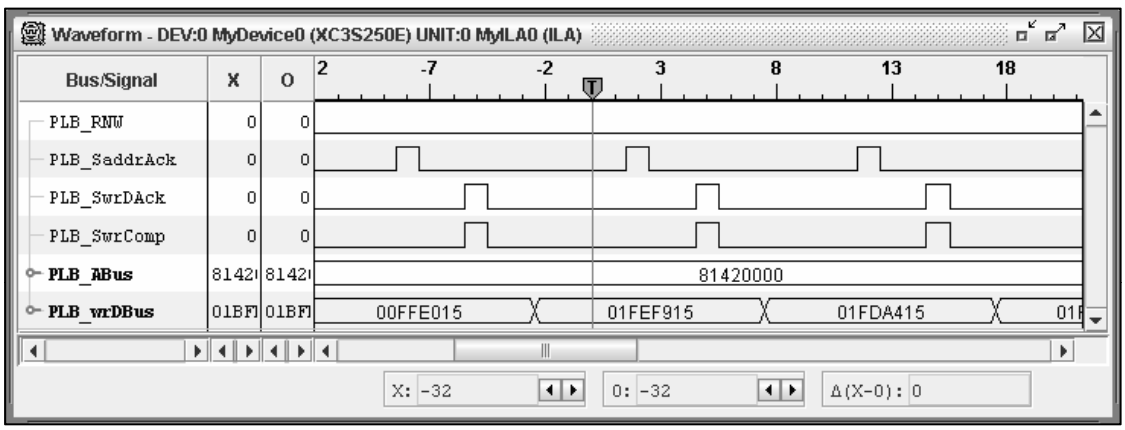
- **A trigger és a tárolási feltételek beállítása**



- **A trigger élesítése**
  - ***Trigger Setup*** **menü → *Run* vagy a ▶ gomb**
- **A debugger elindítása és várakozás a beállított feltételek teljesülésére**

A ChipScope logikai analizátor

- **A trigger feltétel teljesülése után a megadott számú minta tárolásra kerül és megjelenik a hullámforma ablakban**



BME-MIT

FPGA labor

Példa

Példa a ChipScope és a debugger együttes használatára:

- **Timer megszakításos alkalmazás: álljon le a program futása, ha a LED-ekre legalább 0x10 értéket írunk ki**
- **HW/SW Co-Debug legyen engedélyezve (XPS: debug beállítások)**
- **Töröljük a hullámforma ablakban az összes jelet**
- **A hullámformához adjuk hozzá a következő jeleket:**
  - ***PLB_RNW***: írás (0) / olvasás (1) kiválasztó jel
  - ***PLB_PAValid***: a cím érvényességét jelzi
  - ***PLB_SaddrAck***: a cím nyugtázó jel
  - ***PLB_SwrDAck***: az írási adatok nyugtázó jele
  - ***PLB_SwrComp***: az írási adatátvitel végét jelzi
  - ***PLB_ABus***: címbusz
  - ***PLB_wrDBus***: írási adatbusz

Példa

A Match Unit-ok beállítása:

- **Match Unit 0 (M0)**
  - PLB hibajelek és PLB reset jel
  - MicroBlaze programszámláló és halted jelzés
  - Művelet: ==
  - Mindegyik bit legyen X
- **Match Unit 1 (M1)**
  - PLB vezérlő és nyugtázó jelek
  - Művelet: ==
  - Akkor legyen trigger, ha a címet nyugtázta a periféria
    - PLB_RNW: 0, PLB_PAValid: 1, PLB_SaddrAck: 1, a többi bit legyen X
- **Match Unit 2 (M2)**
  - PLB címbusz
  - Művelet: ==
  - Érték: a GPIO LED periféria címe (a formátumot állítsuk HEX-re)

Példa

A Match Unit-ok beállítása:

- **Match Unit 3 (M3)**
  - PLB írási adatbusz
  - Művelet: >=
  - A formátumot állítsuk HEX-re
  - Érték: 0000_0010 (64 bites esetben: 0000_0010_0000_0000)
- **Match Unit 4 (M4)**
  - PLB olvasási adatbusz
  - Művelet: ==
  - A formátumot állítsuk HEX-re
  - Érték: XXXX_XXXX (64 bites esetben: XXXX_XXXX_XXXX_XXXX)

Példa

A trigger és a tárolási feltételek beállítása:

- **Trigger feltétel: M1 && M2 && M3**
  - A címet nyugtázta a periféria (M1) és
  - A címbuszon a GPIO LED periféria címe van (M2) és
  - Az adat értéke nagyobb vagy egyenlő mint 0x10 (M3)
  - ***Break eseményhez***: **Output Enable** legyen **Pulse (high)**
- **Capture**
  - Type: Window
  - Windows: 1
  - Depth: a minták száma (legyen például 64)
  - Position: a trigger esemény pozíciója (legyen például 20)
  - ***Tárolási feltétel***: **All Data**

Példa

- **Indítsuk el a debuggert, majd pedig élesítsük a triggert**
- **A beállított trigger feltétel teljesülésekor**
  - A hullámforma ablakban megjelennek a minták
  - Leáll a program futása (break)