# Using ChipScope
# with
# Xilinx Platform Studio (XPS)

Kazi Asifuzzaman

asifuzzaman.kazi.186 @ student.lu.se

Lund University, Sweden – August 19, 2010

**Introduction :**

A step by step procedure to use ChipScope with a basic system developed in Xilinx Platform Studio (XPS).

**Requirements:**

i. Xilinx ISE Design Suite 12.1
   A. Xilinx Platform Studio (XPS)
   B. ChipScope Pro
ii. Digilent Adept 2.4 or Higher (Windows)
iii. Digilent Nexys 2 Board
iv. Digilent Plug-in for ISE 12.x
v. Download Cable

**Work Flow:**

1. Xilinx Platform Studio (XPS)
   A. Creating a system from the scratch using Base System Builder (BSB)
   B. Getting used to with different parts of XPS
   C. Adding debugging cores into the existing design

2. Digilent
   A. Installing necessary software and plug-in for the target board to be properly connected
   B. Using Digilent Adept for downloading bitstream to the board.

3. Using HyperTerminal to 'talk' with the board.

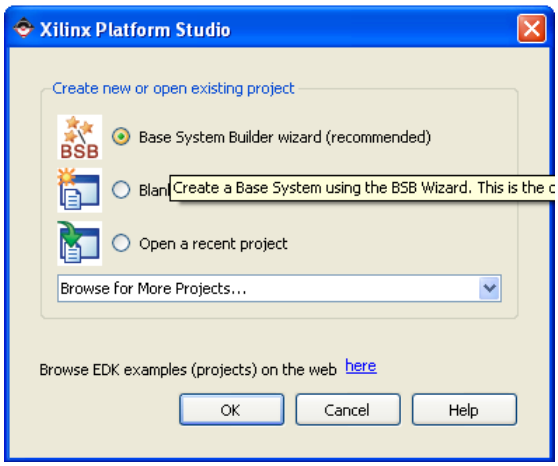
4. Chip Scope Pro
   A. Synchronize an existing design to a new ChipScope project.
   B. Observing bus transactions.

## 1. Xilinx Platform Studio (XPS):

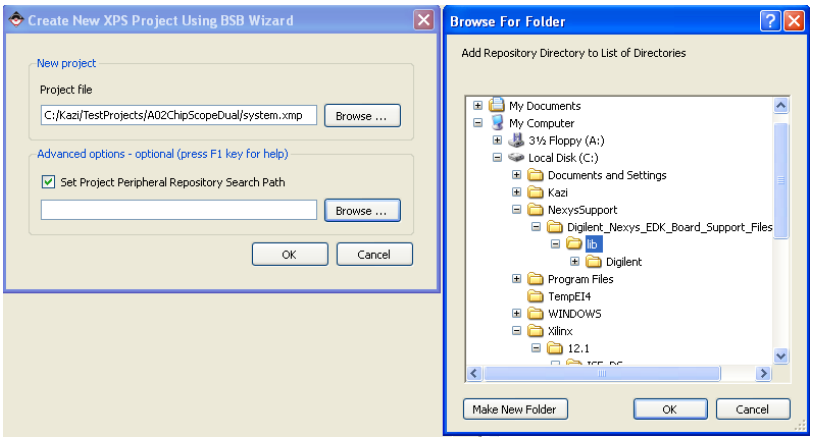
### 1.1 Creating a new design with BSB:

To start, open **Start > All Programs > Xilinx ISE Design Suite 12.1 > Xilinx Platform Studio>**

i. As soon as XPS starts 'Create New or Open Existing Project' wizard shows up.

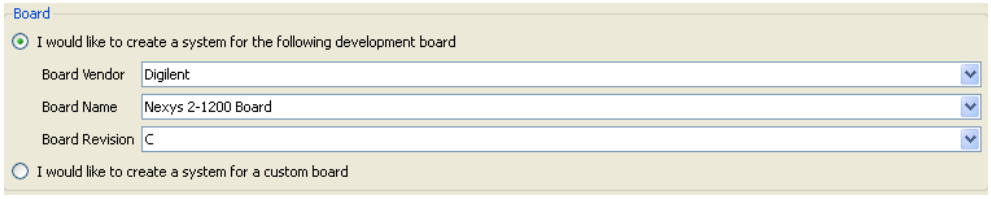



ii. Select the 'Base System Builder' (BSB) option. This would allow you to build a basic working design only with few clicks. However, you have to check/change default settings according to your design needs. Click 'OK'.

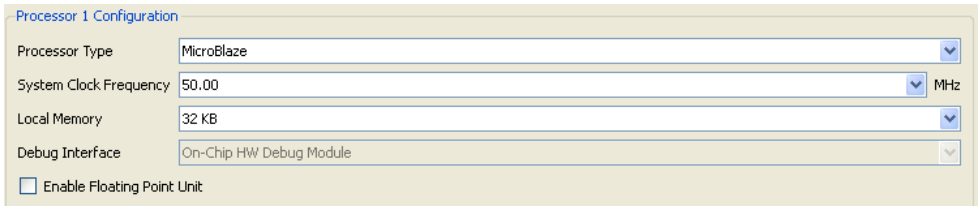



iii. In this window you specify your 'project folder path' and 'project peripheral repository search path'. It is always wise to make a new folder with a relevant name that best describes the project and place the system.xmp file in that folder. It is also recommended that you make the project folder on the hard drive of the computer rather than on a USB memory or any other removable storage device. In this window also locate the support files for the target board as shown. Click 'OK'.

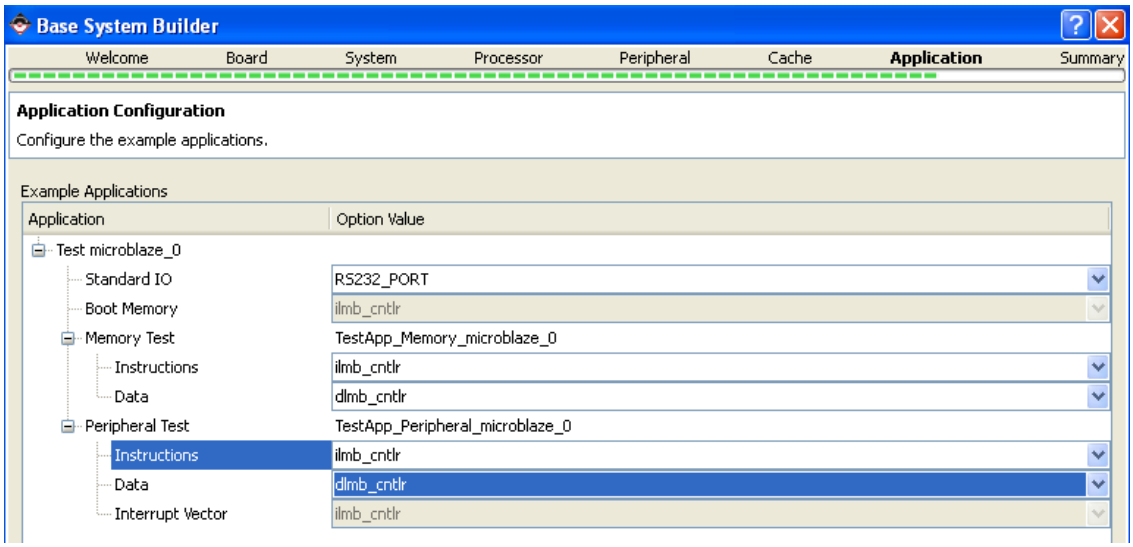
iv. The Base System Builder (BSB) wizard starts. It will take you through rest of the steps of creating a new embedded system design. At the first step select 'I would like to create a new design'. Click 'Next'.




v. Select the board information from the drop down menus as shown. Click 'Next'.
vi. On the system configuration step select 'Single Processor System'. Click 'Next'.
vii. Configure processor 1 with setting shown. Click 'Next'.




viii. Keep default values for 'Peripheral' and 'Cache' steps.
ix. In the 'Application' step, under 'Peripheral Test' submenu change the operation values for 'Instructions' and 'Data' to 'ilmb_cntlr' and 'dlmb_cntlr' respectively.




x. Have a look at the System Summary. Notice the components listed their addresses and location of project files. Click 'Finish'.

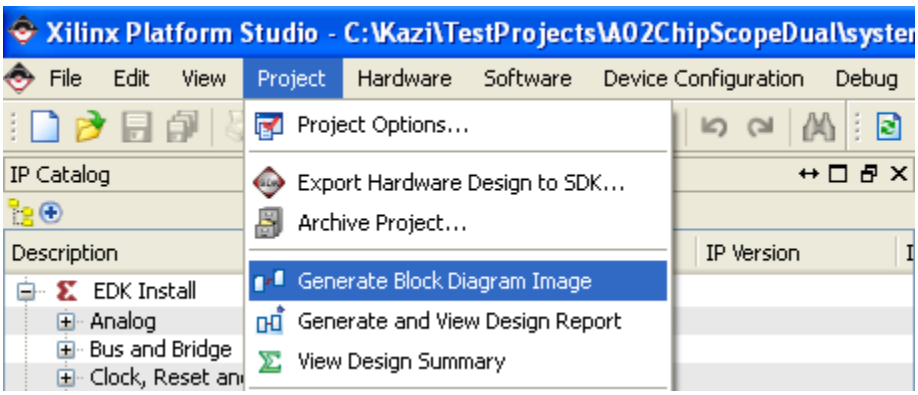
xi. Your initial system has been created. Select 'Start using Platform Studio' in the next wizard.

## 1.2 Exploring Xilinx Platform Studio:

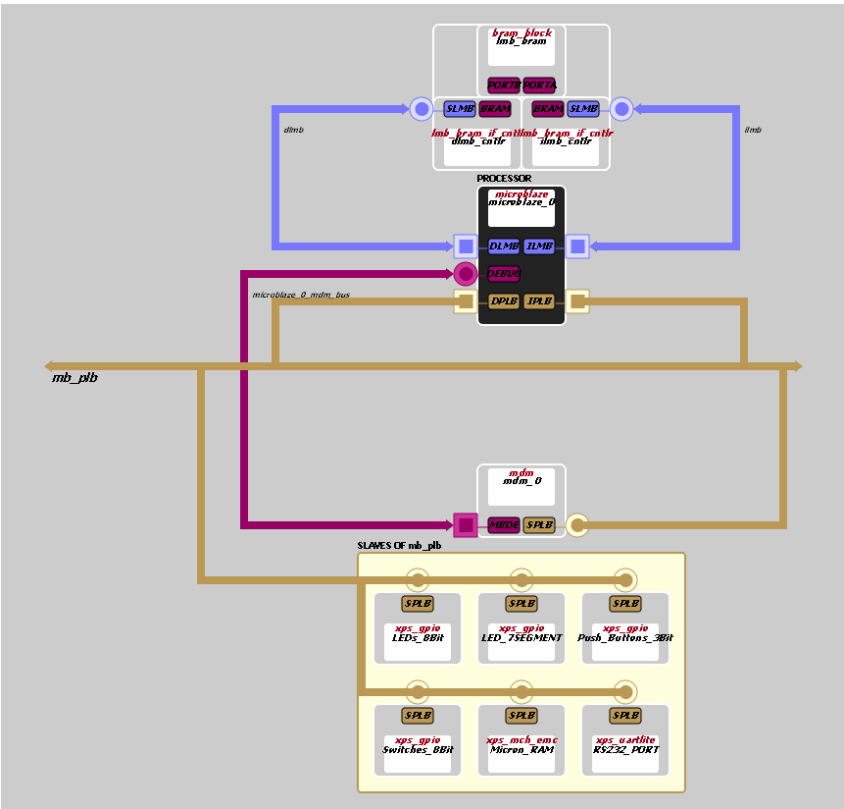
It is good to know few things before actually start working with XPS.

i. Block Diagram:

A block diagram of your new design should already been generated otherwise you can generate it from **Project > Generate Block Diagram Image**.




A Block diagram shows an overview of the components and how they are connected in the system.




Spend few minutes on the block diagram and notice the different parts of the design like BRAM, microblaze_0, FSL, PLB, LMB, GPIO, mdm_0 etc.

ii. Left Panel – Project, Applications & IP Catalog:

- Project Tab: The project references to project related files. Information is grouped in three categories:
  a. Project Files: Project specific files such as the Microprocessor Hardware Specification (MHS) files, Microprocessor Software Specification (MSS) files, User constraint file (UCF) files etc. Have a look at each of the files to have some general idea on these.
  b. Project Options: Project specific options such as device, Netlist, Implementation, Hardware Description Language (HDL) and Sim Model Options.
  c. Design Summary: A graphical display of the state of your embedded design and gives you easy access to system files.

- Applications Tab: This tab lists the software application option settings, header files, and source files that are associated with each application project. With this tab selected you can:
  a. Create and add a software application project, build the project, and load it into the block RAM.
  b. Set compiler options
  c. Add source and header files to the project.

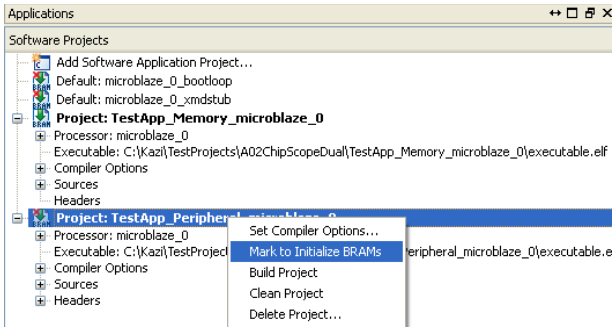
- IP Catalog tab: This tab lists all available IP cores (including their licensing status, release version, supported processors and classification)those can be added to a embedded design.

iii. Right Panel – System Assembly View:

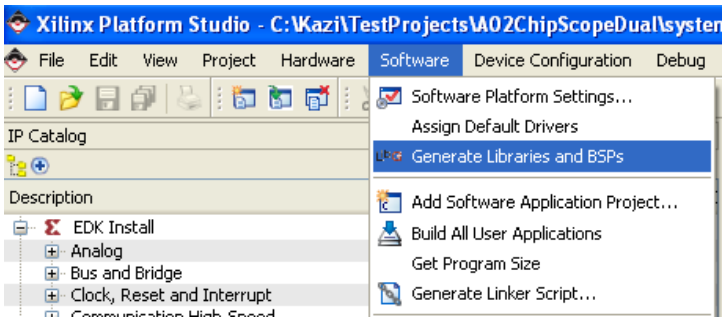
System Assembly View allows you to view and configure system block elements. It has three tabs:

- Bus Interface: Displays the buses in your design. Use this view to modify the information and connections for each bus.
- Ports: Displays ports in your design. Use this view to modify the details for each port.
- Addresses: Displays the address range for each IP instance in your design . Click 'Generate Address' to automatically generate the system address map.

iv. Activate the other Software Project (which activates and uses LEDs, Switches, 7 Segment LED etc. of the FPGA board) by Initializing BRAM for it.



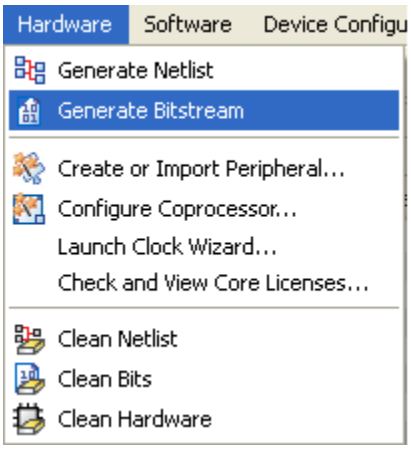
v. To automatically configure and generate library code for your system click **Software > Generate Libraries and BSPs.**




This step also generates header files containing symbolic names for memory mapped peripherals. Run this step and look at your software projects (Applications tab in the left frame). Click on the + left of *processor:microblaze_0* in your active project. There you will find a header file (xparameters.h) with symbolic names of all memory mapped peripherals.

vi. The standard C function **printf** generate huge libraries which will not fit into the small on-chip memory. Instead use **print** or **xil_printf** function to output text. You can compile your programs by selecting **Software > Build All user Applications.**

vii. Click **Hardware > Generate Bitstream.** It might take long to finish (even more than 10 minutes).



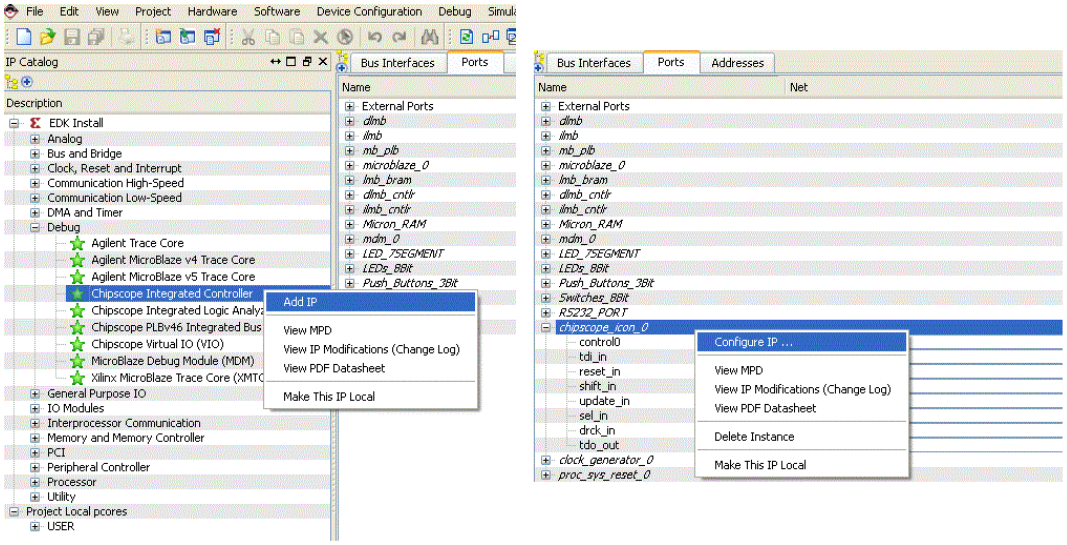

Fortunately the hardware is independent of the software, so whenever you change your C files, this step will not rerun. During the hardware synthesis, the tools use a User Constraint File (UCF), which can be found in the left frame in the Project tab. This file binds external ports in your design to physical pins on the FPGA. It is important that the names in the design are the same as in this file, so do not rename the external ports. Also if you add or remove external pins this file must be edited, i.e. removing a GPIO peripheral.

viii. The last step before downloading the configuration to the FPGA is to merge software binaries and the bit stream from the hardware synthesis. Run **Device Configuration > Update Bitstream.** This places the executable in the on chip memory making the configuration file ready to be downloaded in the FPGA.

## 1.3 Adding Debugging Cores into Design:

i. ChipScope Integrated Controller (ICON): It provides communication with other ChipScope cores. You must use ChipScope ICON core to use any of the other ChipScope cores, because it provides the JTAG connectivity for all other ChipScope cores. One ICON can connect to one or more ChipScope cores via one or more 36-bit control connections. The On-chip Peripheral Bus (OPB) or Processor Local Bus (PLB) Integrated Bus Analyzer (IBA) must be connected to the bus using a monitor (Bus Analyzer) connection.

a. To add a ICON core to the design, open the '**IP Catalog**' tab. Unfold the '**Debug**' menu and locate '**ChipScope Integrated Controller**'. Select and right click on the IP, click '**Add IP**'.




b. Notice that, in the '**System Assembly Panel**', under **Ports** tab a new peripheral is added named '**chipscope_icon_0**'. Right click on the instance and click '**Configure IP**'.

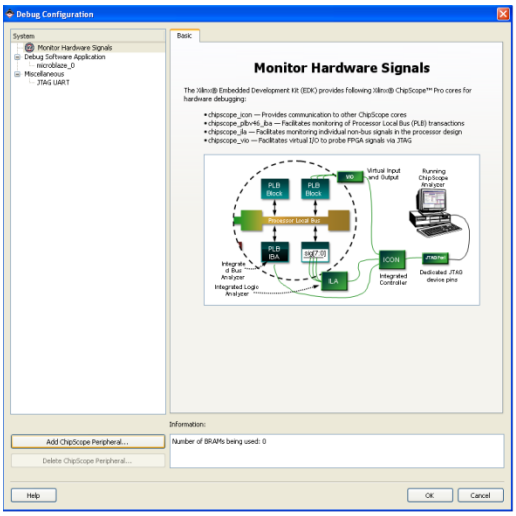
c. In the appearing configuration wizard set the value of '**Number of Control Ports**' as 2. This means this ICON can now be connected to two other ChipScope cores.

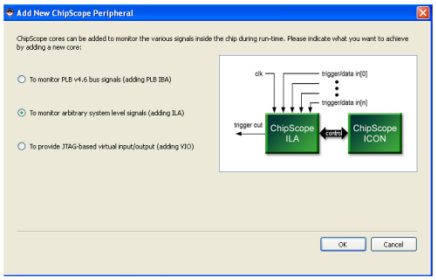
d. You can check the **system.mhs** file at **Project** tab has now been updated with following modifications:

```
BEGIN chipscope_icon
 PARAMETER INSTANCE = chipscope_icon_0
 PARAMETER HW_VER = 1.04.a
 PARAMETER C_NUM_CONTROL_PORTS = 2
END
```

ii. Integrated Logic Analyzer (ILA): ILA is used to monitor individual signals in a processor design. When the ChipScope ILA core is used in Platform Studio, only signals at the top level of the processor design (at the Microprocessor Hardware Specification (MHS) level) can be monitored using the ILA. Using the ILA connection feature in the FPGA Editor, you can monitor signals at level of hierarchy using this ILA.

a. To add a ILA core to the design click **Debug > Debug Configuration.**

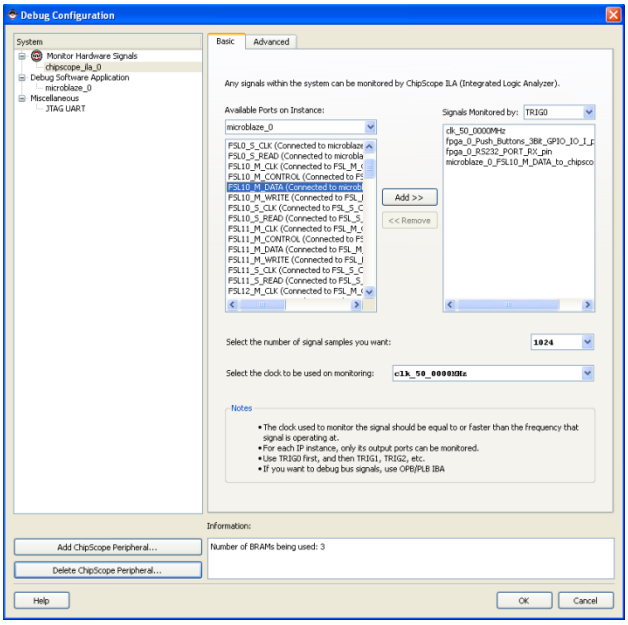



b. Notice the diagram on the appeared window. It may help you to understand the connection and organization between ChipScope cores. Click 'Add ChipScope Peripherals'.

c. Select 'To monitor arbitrary system level signals (adding ILA)' from the list.

d. Configure ILA and select which signals you wish to monitor with it.



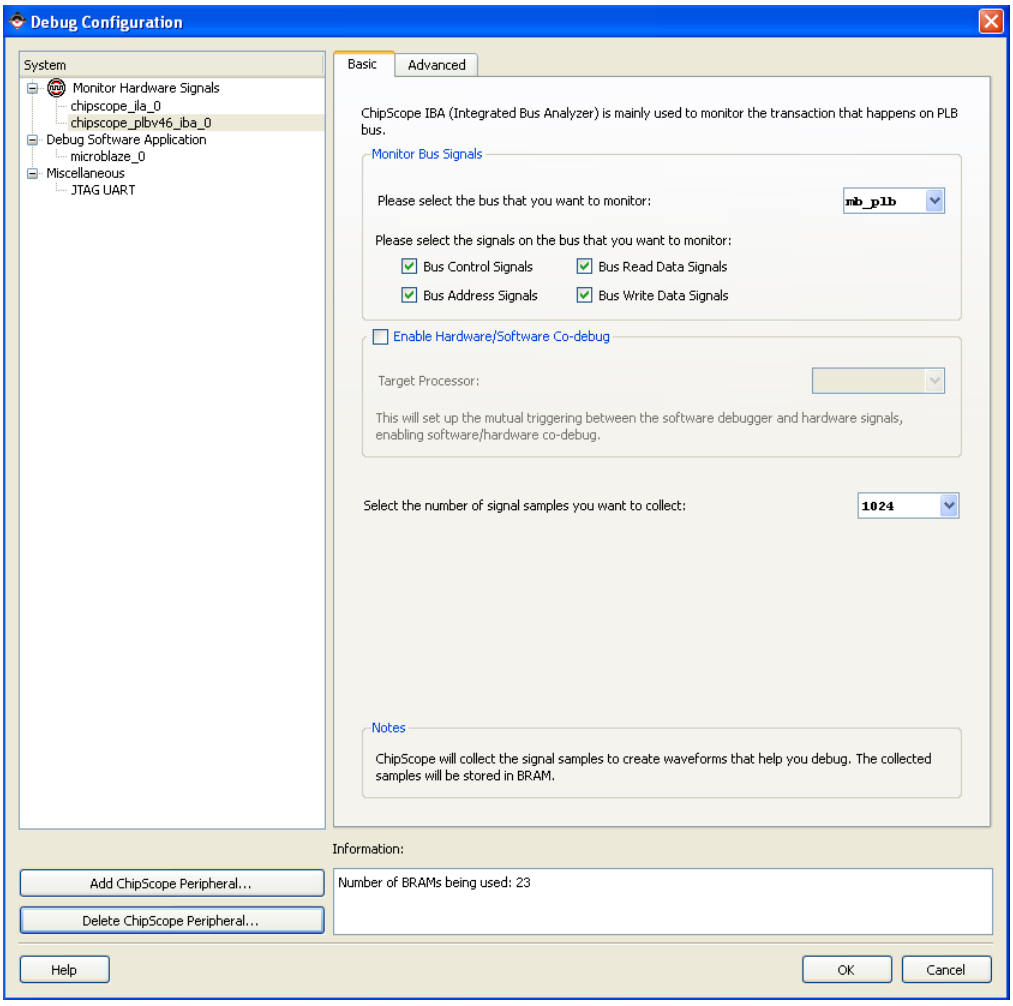
iii. Integrated Bus Analyzer (IBA): Facilitates monitoring of Processor Local Bus (PLB) transactions.

a. To add a IBA core to the design click **Debug > Debug Configuration.**

b. In the window appeared Click 'Add ChipScope Peripherals'.

c. Select the option 'To monitor PLB v4.6 bus signals (adding PLB IBA)'.

d. Configure IBA as shown in the following figure:




e. Click 'OK'.

f. You can also check and modify configurations setting for these cores directly into **system.mhs** file under project tab.

g. Always remember to perform **Device Configuration > Update Bitstream** after doing any modification in the design to ensure that FPGA gets the updated configuration.

## 2. Digilent:

2.1 Initializing Cable Connection to the Board:

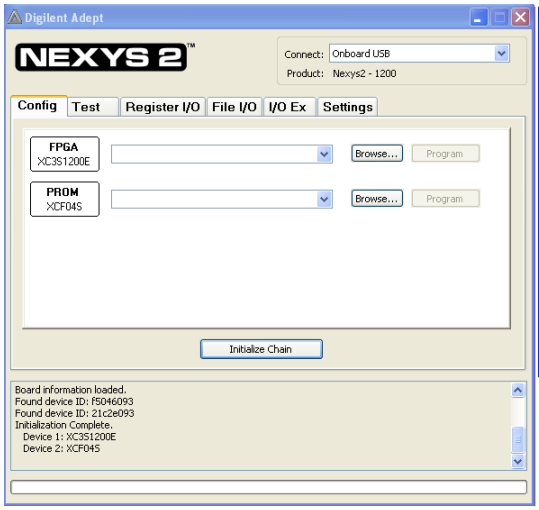
i. Connect the board properly with both USB and RS232 cables. And switch the power on.
ii. Microsoft 'Found New Hardware' starts. Select 'Yes, this time only'. Click 'Next'.
iii. Select 'Install the Software Automatically (Recommended)'. Windows will now import some files automatically to install the new device.
iv. Install Digilent Adept 2.4 (If not already installed)

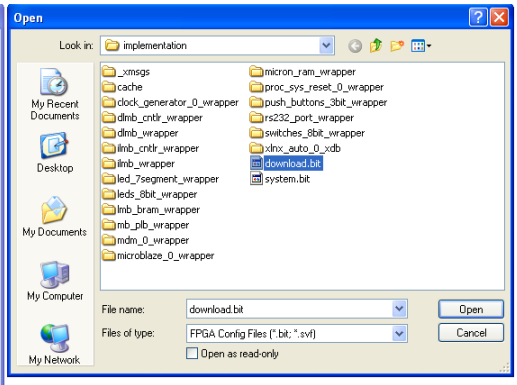
2.2 Installing Digilent Plug-in for ISE 12.x:

i. Copy the plug-in files "**libCseDigilent.dll**" and "**libCseDigilent.xml**" into the ISE Design Suit Installation.
ii. For the ISE Design Suite, the typical location is
**C:\Xilinx\12.1\ISE_DS\ISE\lib\nt\plugins\Digilent\libCseDigilent**
Note: For 64-bit Windows, use **nt64** in place of **nt.**


2.3 Downloading the bitstream to FPGA using Digilent Adept :

i. Start 'Digilent Adept'. Select '**Onboard USB**' from the drop down menu in 'Connect:' field.
ii. Browse the option beside its written FPGA XC3S1200E and select the **download.bit** configuration file from your project directory home under implementation/. Click '**Program'.**
iii. Your FPGA is now programmed with your design configuration.

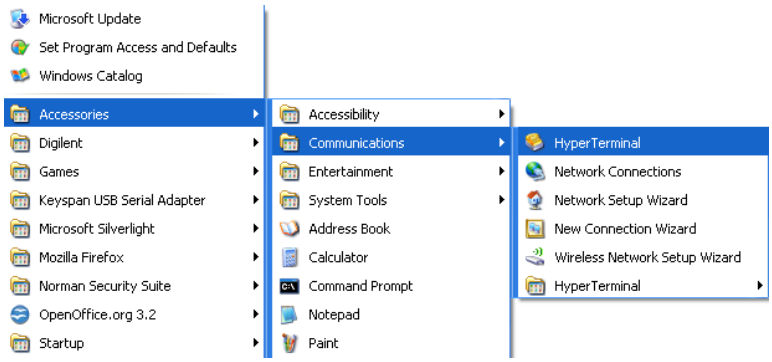


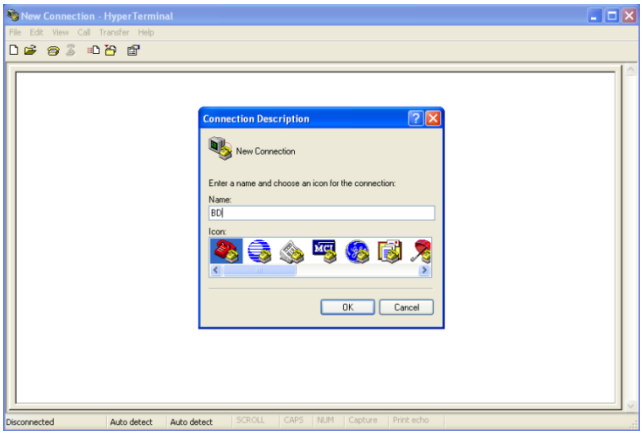
### 3. Communicating with the Board Through HyperTerminal:

To make sure that everything went right in all previous steps, we would try to communicate with the board through HyperTerminal. At this stage the board should be programmed with your design configuration (**download.bit** file has been downloaded in the FPGA board).

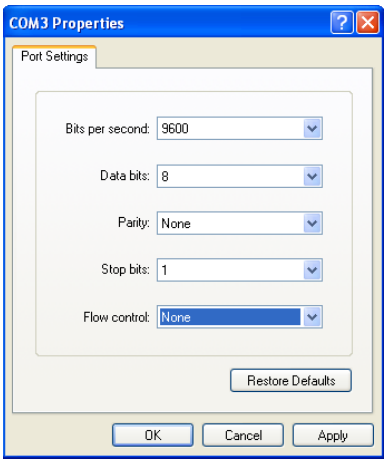
- Start HyperTerminal from Start Menu.



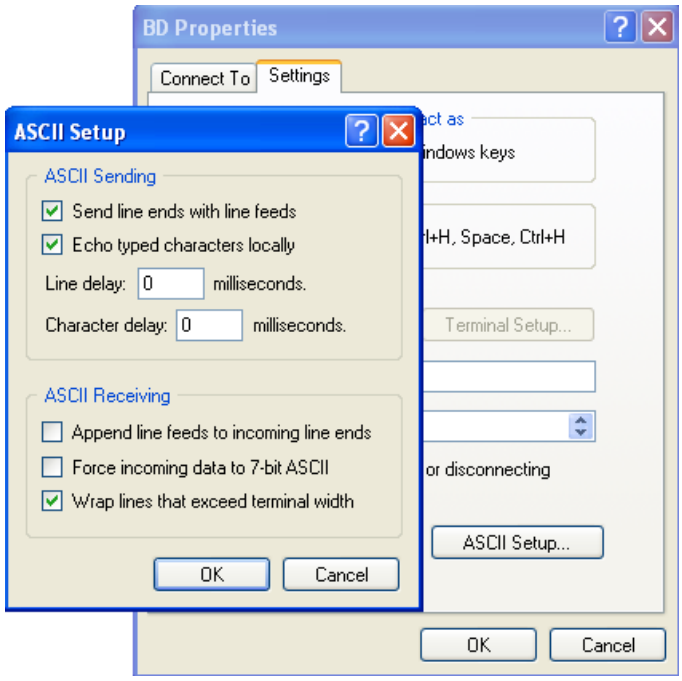

- Give any name to the new connection.



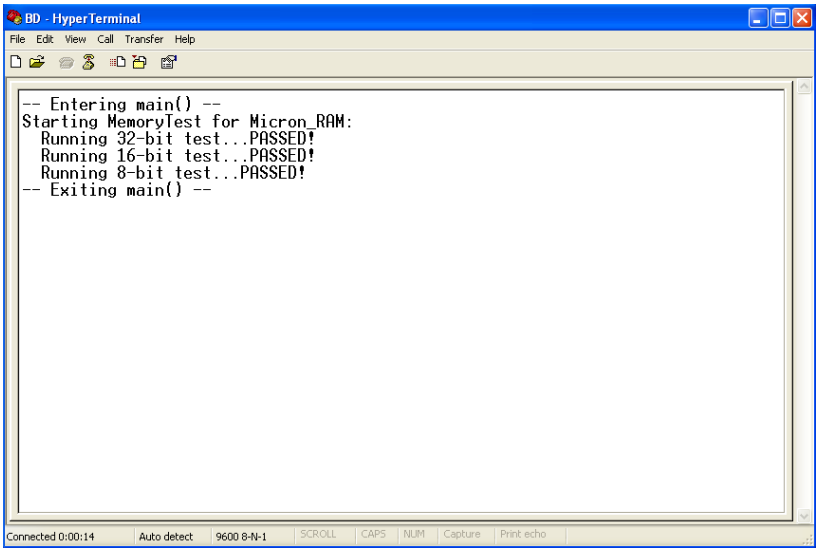

- Select appropriate port of communication (COM1/COM3 etc.)
- Define port setting as below. Click OK.



- You can check ‘Send line ends with line feeds’ and ‘Echo typed characters locally’ at ASCII setup in connection properties to view sent data in the Hyper Terminal window. (Optional)




- Now, press the BTN0 button on the FPGA board. HyperTerminal window should show following output:

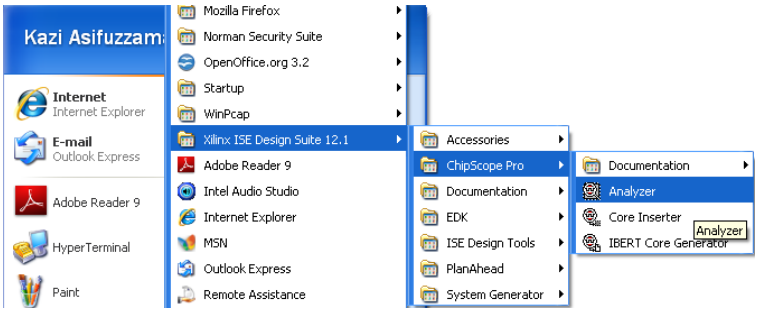



- If you don’t get this output on HyperTerminal window check back or redo your previous steps.

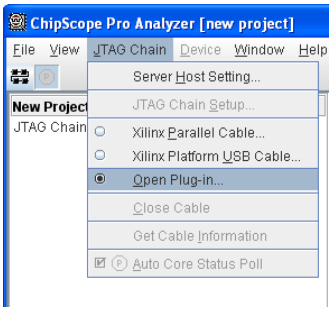
## 4. Chip Scope Pro:

Once you are done with all the previous steps of this manual then you can proceed using the ChipScope Pro analyzer. At this point it is assumed that you have the target board connected with the system and it has been programmed with your configuration (**download.bit** file has been downloaded in the FPGA board).

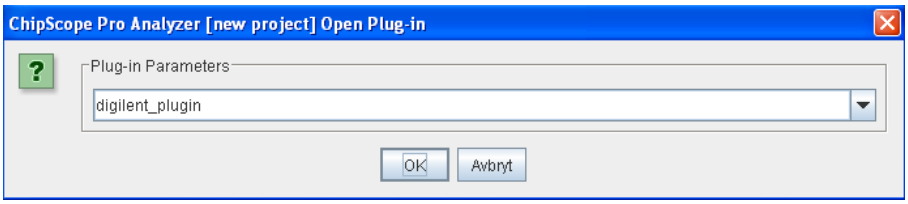
- Start ChipScope Pro Analyzer from Start Menu.



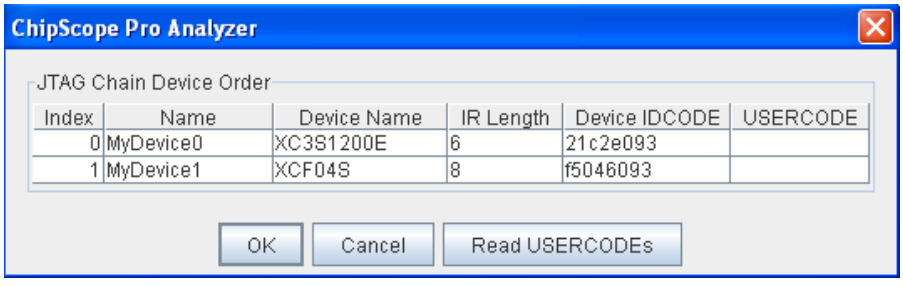

- To connect the board select **JTAG Chain > Open Plug-in.**



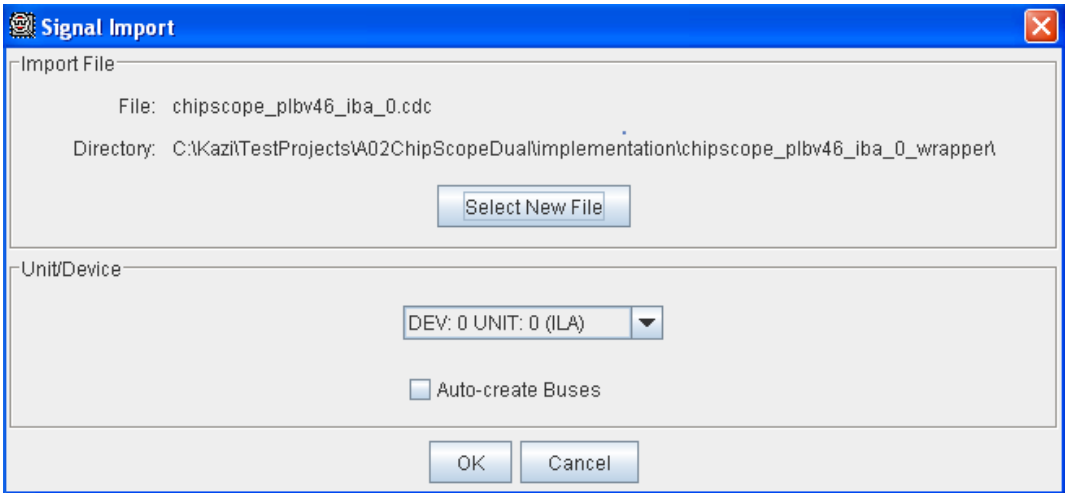

- In the appeared window write down '**digilent_plugin**' in the text box for Plug-in Parameters. Click 'OK'.



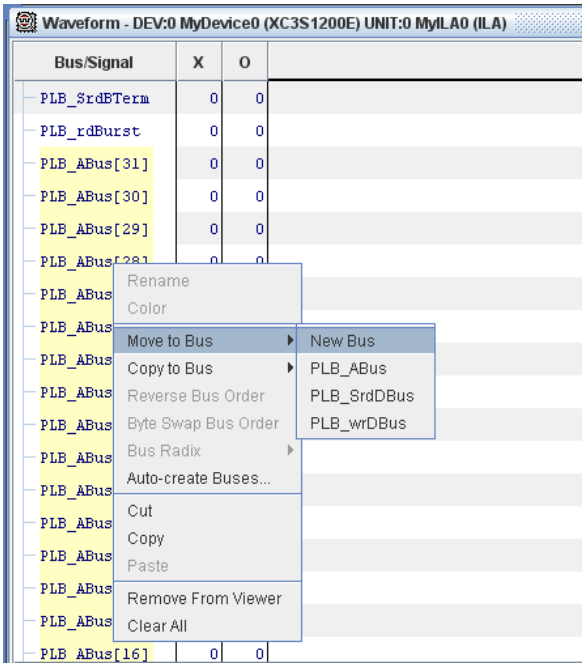

- Check if the JTAG devices appeared correct in the new window. Click 'OK'.



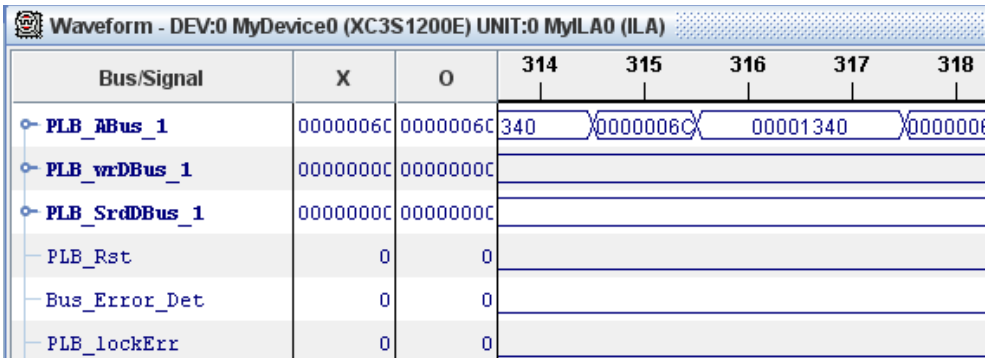
- Import .cdc file for IBA core by **File > Import** and locating corresponding address of the file under implementation/ of the project directory; as shown in the following figure. Click OK.



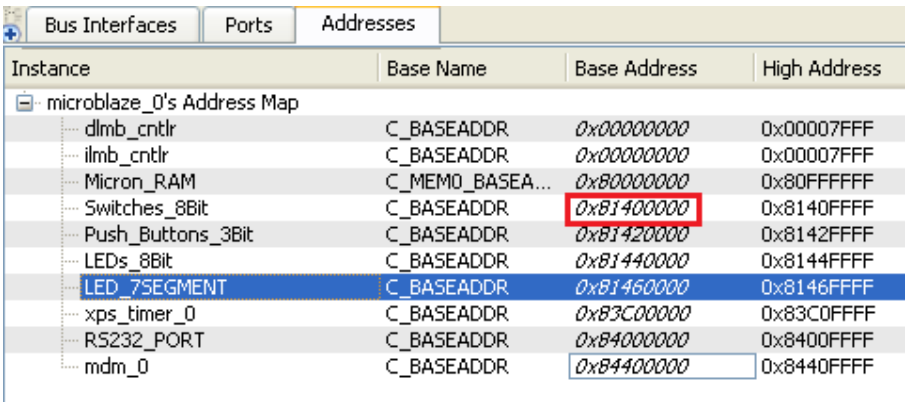

- You can now see array of buses from your design in the waveform window. Group the array of same bus as one single bus (select all > right click > move to bus > new bus). Do the same for every multi-bit signals.



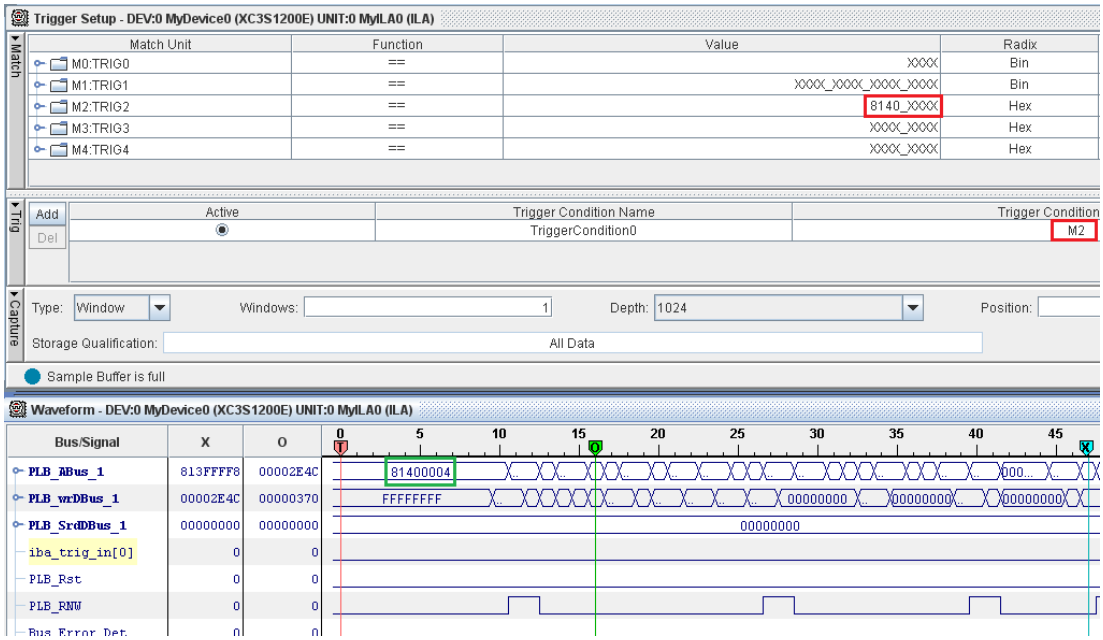

- After grouping all the signals press the **T!** (Trigger Immediately) button to see the data transaction in the buses.




- Now we will move on to trace data on a specific location using ChipScope. To do that we have to manually configure a trigger with our desired function. Let's say we want to monitor transactions regarding **8 Bit Switches**. Then we have figure out the memory addresses being used for this core from the '**Addresses**' tab of the '**System Assembly View**' of XPS.

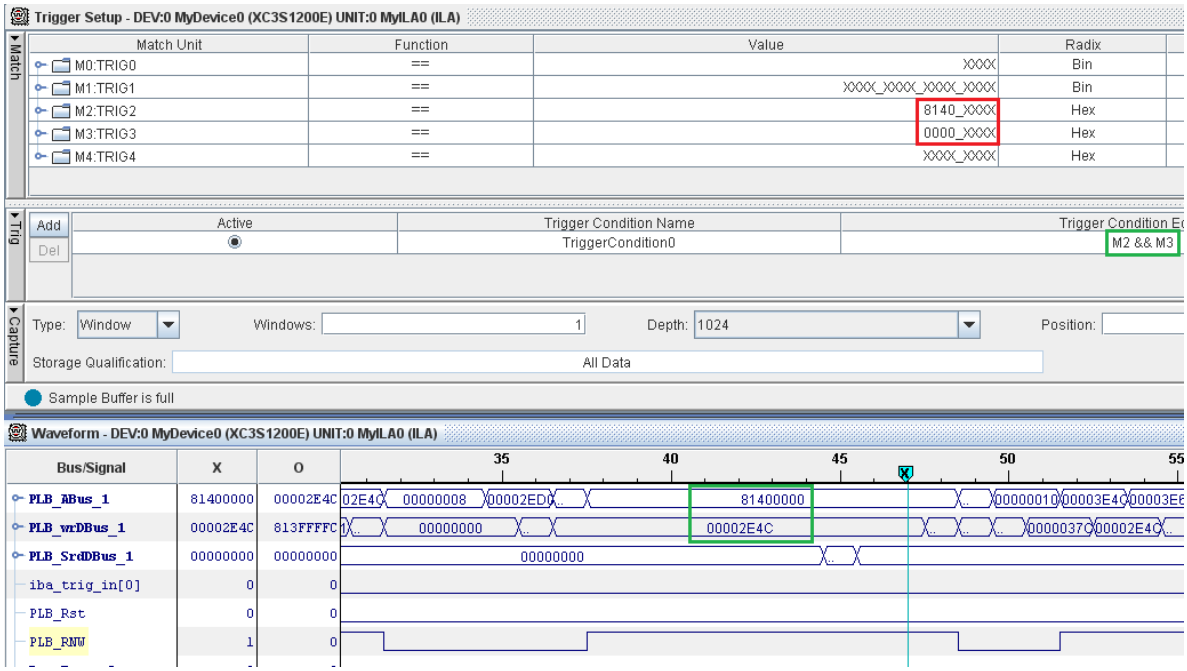



- Notice that **PLB_ABus_1** handles the transactions of addresses being used for the cores. So to observe bus transactions regarding **8 Bit Switches** we have to modify the corresponding match unit value (**M2:TRIG2** in this case; checks PLB_ABus_1) in the following way:



- Click **Trigger Setup > Run** and push the **BTN0** button of the FPGA board once; which resets the program on the board allowing ChipScope to trigger and capture the desired transaction . After few seconds the waveform is displayed where the bus transactions of your target core is captured.

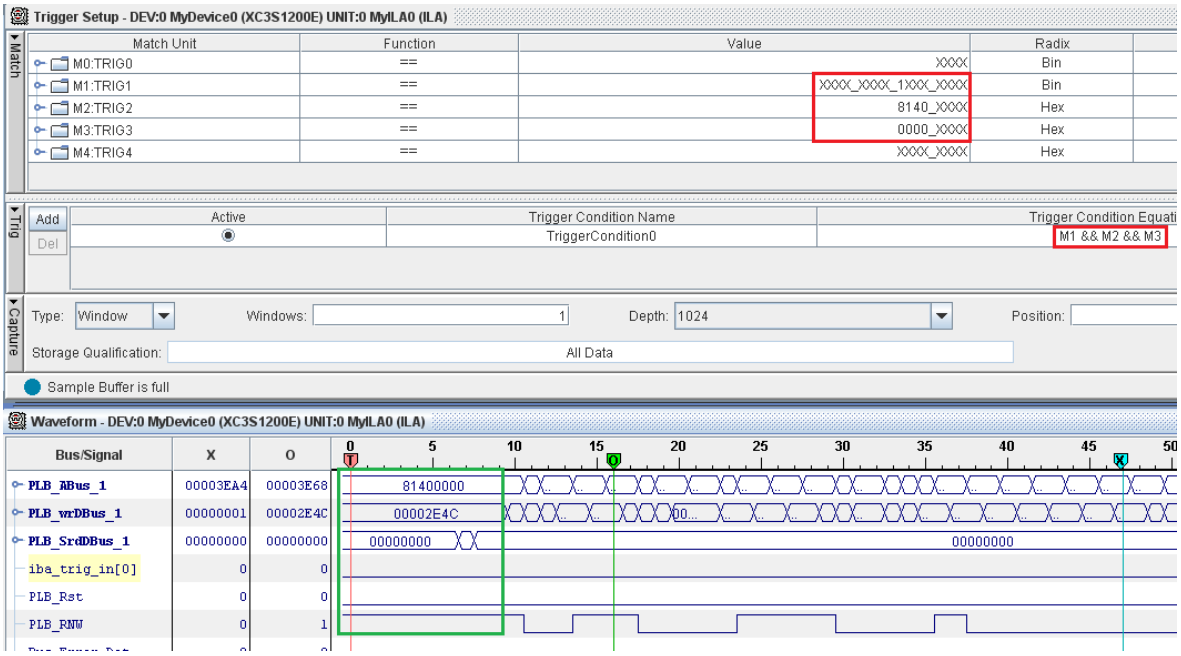
- Now, you can impose other conditions to examine an exact scenario. For example you want to see bus transactions for the same core but for specific dataset on **PLB_wrDBus_1.** First, you have to figure out which match unit value you need to change for **PLB_wrDBus_1** (In this case it is **M3:TRIG3**). Let's say we are looking for a bus transaction on **8 Bit Switches** when **PLB_wrDBus_1** has a value of 0000_XXXX. Don't forget to include **M3** in the Trigger Condition Equation. Click **Trigger Setup > Run** and push the **BTN0** button of the FPGA board once.



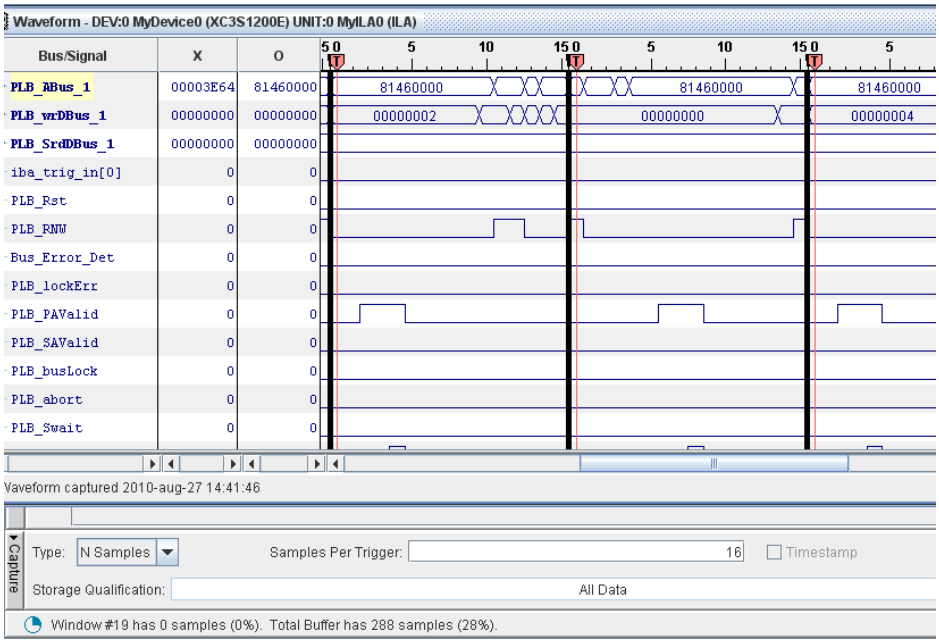

- You can see that an event in waveform has been captured with your desired bus transaction (if any).

- You can further enhance your trigger condition by adding more criteria. For example, if you want to see a bus transaction where the signal **PLB_RNW** is high in addition to previous conditions. Find out which match unit is handling the particular signal (**M1** in this case, you may have to unfold the menu to see which buses are in there). Change it as you like it to see and include the condition in Trigger Condition Equation. Click **Trigger Setup > Run** and push the **BTN0** button of the FPGA board once.

- The captured waveform would show the bus transaction(s) where all conditions that you defined are satisfied.




- However, depending on your purpose you can also combine the conditions with an '**OR**' function instead of an '**AND**'. Where the waveform would have outputs if any of the conditions satisfy.
- You can also get finite number of samples of desired set of transactions by changing the 'Type' field in the 'Capture' tab to 'N Samples' from 'Window'. Now you will see different occurrences of your desire transaction instead of an regular waveform with all bus transactions.




- In this case the buffer may not get 'full' as you can see in the bottom of this pane. It may keep waiting for more samples even if there is none. But you can still see the samples by pushing 'Stop'.

**Abbreviations:**

| | |
|---|---|
| **GPIO** | **- General Purpose IO** |
| **LMB** | **- Local Memory Bus** |
| **PLB** | **- Processor Local Bus** |
| **RS232** | **- Serial Port** |
| **BRAM** | **- Block RAM** |
| **VGA** | **- Video Graphics Array** |
| **OPB** | **- On Chip Peripheral Bus** |
| **MDM** | **- Microblaze Debug Module** |
| **IP Core** | **- Intellectual Property Core** |
| **FSL** | **- Fast Simplex Link** |

**References:**


**[1]** http://www.xilinx.com/

**[2]** http://digilentinc.com/

**[3] Xilinx Platform Studio Tutorial, Flavius Gruian & Per Anderson**