Basic IP guidelines

An IP block is created outside a Vivado project. It is created as stand alone IP so that it can be use multiple times. An IP block is created under */Libraries* as in a directory *_Lib*.
The directory */Libraries/IpCoreName_Lib* under a newly created project is there is placeholder for this *readme_Ip* file.

Create IP

- Open a terminal or command window.
- Change directory (cd) to: */Libraries*
- Create under */Libraries* a directory for the IP core, like : *mkdir _Lib*
- Change directory (cd) to: */Libraries/_Lib*
- Start here Vivado by typing: Vivado & (& is for Linux users to free the terminal).
- In the main windows select:
    - Tasks - Manage IP >
    - Select [New IP Location]
    - Click [Next].
    - In the new popped up window, provide:
        - Part : Click the ... square at the right and select from th menus the sued component.
        - Target Language : In this case VHDL.
        - Target Simulator : In this case the Questa Advanced Simulator
        - Simulator Language : Mixed
        - IP Location : This should already point to the directory where Vivado is started, being : *.../Libraries/_Lib*
    - Click [Finish]
    - Vivado will create the project. In the */Libraries/_Lib* a couple of extra directories are created (.Xil, ip_user_files and managed_ip_project). A new Vivado window is started, this is the IP Manger window.
    - In the right IP Catalog pane select what IP to create and double click it.
        - Example : Under "debug & Verification" select "debug" and double click "VIO (Virual input/output)".
        - A new popup window appears, Customize IP.
        - Click in the top-right the "Component Name" box and give the IP to create a useful name.
        - Start configuring the IP and when done, click [OK].
    - The IP manager / Customizer shows a new popup, simply click [Generate]
    - The IP block we need and customized is now generated.
        - It is generated in its own directory named as the name entered in the "Component Name" box.
    - Close Vivado.

Use the created IP:

- Open the .vhd file where the IP must be instantiated. In case of VIO and ILA IP this is most of the time the Top level .vhd file of the design.
- Enter the library of the IP in the top of the file, above the top level entity.
  library *Lib;*
  use Lib.all;
- At the position where the IP must be instantiated enter:
  - "label" : entity _Lib.
    port map
  - get the ports for the .vho file in the location where the IP is created.

Modification of the IP:

If it is necessary to change something to the IP core, do it this way:

- Open a terminal/command windows
- cd to: ./Libraries/_Lib
- Start Vivado by typing: Vivado & (& is for Linux users to free the terminal).
- In the main windows select:
  - "Manage IP"
  - "Open IP Location"
  - Select/browse in the pop-up file selector screen the */Libraries/_Lib* folder.
  - Press [Select]
- The IP project manager starts and displays in the [Sources] window the IP.
  - Click on .
  - It should now display right of a value equal to the files generated for the Ip and it should also display a list of files below .
- Right click on .
  - Select "Re-customize IP".
  - Perform the modifications needed for the design.
  - Click [OK].
  - In the popup window click on [Generate]
  - Click [OK] on the second pop-up about Out-Of-Context running.
  - Wait till the "Design Runs" window shows a 'V' in front of the IP core.
- Close Vivado
- The IP core is now generated and ready for instantiation.
  - Use the ".vho" file, part from 'your_instance_name : to
    instantiate the modified IP core in the design.

###REMARKS for VIO and ILA cores.
It is possible that Vivado issues warnings about the VIO or the dbg_hub component.
The warning will be about the difference between the clock frequency set for the
VIO or ILA/dbg_Hub and the clock frequency set for the VIO or ILA clock in the design XDC file.
To solve this do (example given for VIO):

- Open a terminal/command windows
- cd to: ./Libraries/Vio_Lib
- Start Vivado by typing: vivado & (& is for linux users to free the terminal).

- In the main windows select:
    - "Manage IP"
    - "Open IP Location"
    - Select/browse in the pop-up file selector screen the "./Libraries/Vio_Lib" folder.
    - Press [Select]
- The IP project manager starts and displays in the [Sources] window the "VioTop" IP.
    - Click on "VioTop".
    - It should now display right of "VioTop" 22 (= files generated).
    - It should now display a list of files below "VioTop".
- Right click on "VioTop".
    - Select "Reset Output Products"
- DO NOT CLOSE vivado, jsut leave it hanging.
- Open now a file browser (Windows: Windows Explorer, Linux : Nautilus).
- Browse into: ./Libraries/Vio_Lib/VioTop
- Open with a text editor: "VioTop.xci".
    - Find the text line: 'BUSIFPARAM_VALUE.SIGNAL_CLOCK.FREQ_HZ'.
    - Enter after the > character the frequency used as VIO clock.
    - Save the file.
    - Open with a text editor: "VioTop.xml".
        - Find the text line: '"BUSIFPARAM_VALUE.SIGNAL_CLOCK.FREQ_HZ"'.
        - Enter after the > character the frequency used as VIO clock.
        - Save the file.
- Return to Vivado and the IP manager.
    - Right click "VioTop".
    - Select "Generate Output Products".
    - In the popup window click on [Generate]
    - Click [OK] on the second pop-up about Out-Of-Context running.
    - Wait till the "Design Runs" window shows a 'V' in front of the IP core.
- Close Vivado
- The IP core is now generated for the frequency just entered in the two files.
- This can be checked by opening: "./Libraries/Vio_Lib/VioTop/VioTop_ooc.xdc"
    - The create_clock constraint should show the frequency entered in both files.

Example of VIO IP usage

In source code files where this VIO is going to be instantiated, use library setting:

```
    library xil_defaultlib;
        use xil_defaultlib.all;
```

This is the base VIO, modify it and/or instantiate the VIO as:
The VIO component MUST then be instantiated as:

```
    <your_instance_name> : entity xil_defaultlib.VioTop
    port map (
        clk         => <Connected_Clock>,
        probe_in0   => <Connected_Signal_or_Bus>,
        probe_in1   => <Connected_Signal_or_Bus>,
        probe_in2   => <Connected_Signal_or_Bus>,
        probe_in3   => <Connected_Signal_or_Bus>,
        probe_in4   => <Connected_Signal_or_Bus>,
        probe_in5   => <Connected_Signal_or_Bus>,
        probe_in6   => <Connected_Signal_or_Bus>,
        probe_in7   => <Connected_Signal_or_Bus>,
        probe_in8   => <Connected_Signal_or_Bus>,
        probe_in9   => <Connected_Signal_or_Bus>,
        --
        probe_out0  => <Connected_Signal_or_Bus>,
        probe_out1  => <Connected_Signal_or_Bus>,
        probe_out2  => <Connected_Signal_or_Bus>,
        probe_out3  => <Connected_Signal_or_Bus>,
        probe_out4  => <Connected_Signal_or_Bus>,
        probe_out5  => <Connected_Signal_or_Bus>,
        probe_out6  => <Connected_Signal_or_Bus>,
        probe_out7  => <Connected_Signal_or_Bus>,
        probe_out8  => <Connected_Signal_or_Bus>,
        probe_out9  => <Connected_Signal_or_Bus>
    );
```