VisualSim® Interface with FPGA – User Guide

MIRABILIS DESIGN INC.

1159 Sonora Ct,

Suite 116, Sunnyvale,

CA 94086, USA

MIRABILIS
design

The information contained herein is subject to change without notice. While every reasonable effort was made to ensure the completeness and correctness of this document, Mirabilis Design Inc. makes no warranty of any kind with regard to this material, including but not limited to any implied warranties. Mirabilis Design Inc. shall not be liable for errors or omissions contained herein or for any damages relating to the use of this material.

VisualSim and VisualSim Architect are registered trademark of Mirabilis Design Inc.

Java and all Java-related titles are trademarks or registered trademarks of Sun Microsystems in the United States and other countries. All other brand or product names may be trademarks of their respective holders.

Mirabilis Design Inc.

1159 Sonora Ct, Suite 116

Sunnyvale, CA 94086

Revision History

The following table shows the revision history for this document

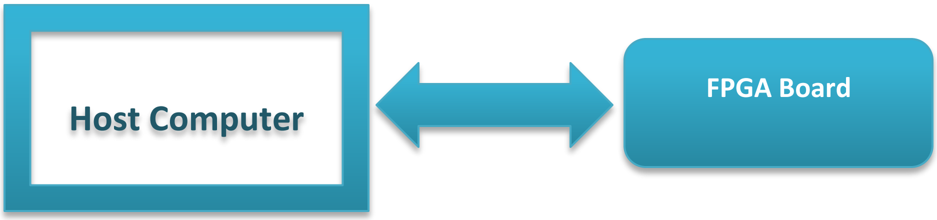
Date	Version	Revision
Feb 20, 2015	2.0	1
Apr 7, 2015	2.1	2

Table of Contents

Introduction

VisualSim® is a system level modeling and simulation tool from Mirabilis Design for conducting Architecture Exploration, power and Performance analysis of proposed or existing systems early in the design flow. This document explains how to use FPGA in loop with VisualSim®. Here we have considered Xilinx Zynq 7000 platform board as the example device that communicates with VisualSim®, however the source code and the setup is completely generic. This interface has been demonstrated on Xilinx Zynq 7000 platform board (Zybo board), using the Vivado tool suite from Xilinx and the accompanying Xilinx SDK. The version of Xilinx tool used is Vivado 10.2.

Using this interface one can communicate with FPGA boards from the host computer and perform processing using the data transferred from VisualSim on the target FPGA and also VisualSim can receive processed data back from FPGA for further analysis.



VisualSim FPGA Interface applications

VisualSim FPGA Interface can be used for System level analysis of Embedded Systems including performance, protocol checking, power and for generating test scenarios for the target FPGA. VisualSim can generate verities of test scenarios very quickly and enables the user to validate application on FPGA covering all corner cases.

VisualSim FPGA interface allows user to adopt below mentioned applications but it is not restricted.

- Generating and running system level scenarios
    - VisualSim can be used to generate system level scenarios from the system definition.
    - Transferring the data to the FPGA to run actual data through the system
    - Transfer the data back to VisualSim to perform analysis and display the information in user readable format
- Run performance simulations on the actual hardware system
    - VisualSim to generate complex use case scenarios that define performance bottle necks
    - Run these scenarios on the actual hardware system
    - Collect process and display the results, so system level optimizations can be performed
- Run power scenarios on the actual hardware
    - Generate complex power scenarios for the system
    - Run these scenarios on the actual hardware

  - Collect process and display the results, so system issues can be identified and fixed

Hardware and Software Requirements

VisualSim can communicate with FPGA's having Micro-Controller/Processor, Ethernet port and FPGA design software's along with SDK. Example Hardware and Software's are as follows; Xilinx Zynq 7000, SmartFusion2, Xilinx Vivado.

VisualSim FPGA Interface Kit

VisualSim FPGA Interface Kit provides all the interface files and demo examples. As VisualSim FPGA Interface Kit is provided as completely configurable user can compile the interface based on the requirements.

How Interface Works

VisualSim Interface with Xilinx Zynq 7000

The Zynq platform has an ARM processor and an FPGA as part of its architecture. This interface uses the ARM processor to enable the interface, so the users can use all of the FPGA for their logic, except for a small layer to communicate with the interface.

The software running on the ARM processor is a lwIP Echo Server. The lwIP Echo server application starts an echo server at port 7. Any data sent to this port is consumed by the FPGA processed and sent back through this interface.

By default, the program assigns the following settings to the board:

IP Address: 192.168.1.10

Netmask : 255.255.255.0

Gateway : 192.168.1.1

MAC address: 00:0a:35:00:01:02

These settings can be changed in the file main.c.

platform.c implements certain processor and platform dependent functions.

The file platform_config.h is generated based on the hardware design. It makes two assumptions: The timer has its interrupt line connected to the interrupt controller, and all the ethernet peripherals (xps_ethernetlite or xps_ll_temac) accessible from the processor can be used with lwIP.

Procedure to Run Example

- Import the Vivado project into the Vivado tool suite

- Make the required changes to the project to suit the hardware being used for the demo.
- Compile the project in Vivado
- Launch the SDK from Vivado, so all the required board support packages are exported to the SDK
- Import the SDK project into the SDK
- Make the required changes in the SDK to match the paths of the new packages that were generated for the board being used
- Compile the software
- Program the FPGA with the generated bit stream
- Then launch the software to run on the FPGA and monitor using a telnet from the host PC

To communicate with FPGA, VisualSim uses UDP socket ports and the user can send packet data from VisualSim to FPGA using VisualSim's "Datagram Writer" library block and receives packet from FPGA using "Datagram Reader" library block. Sample VisualSim model and configuration of Datagram Writer and Datagram Reader is shown in below figure.

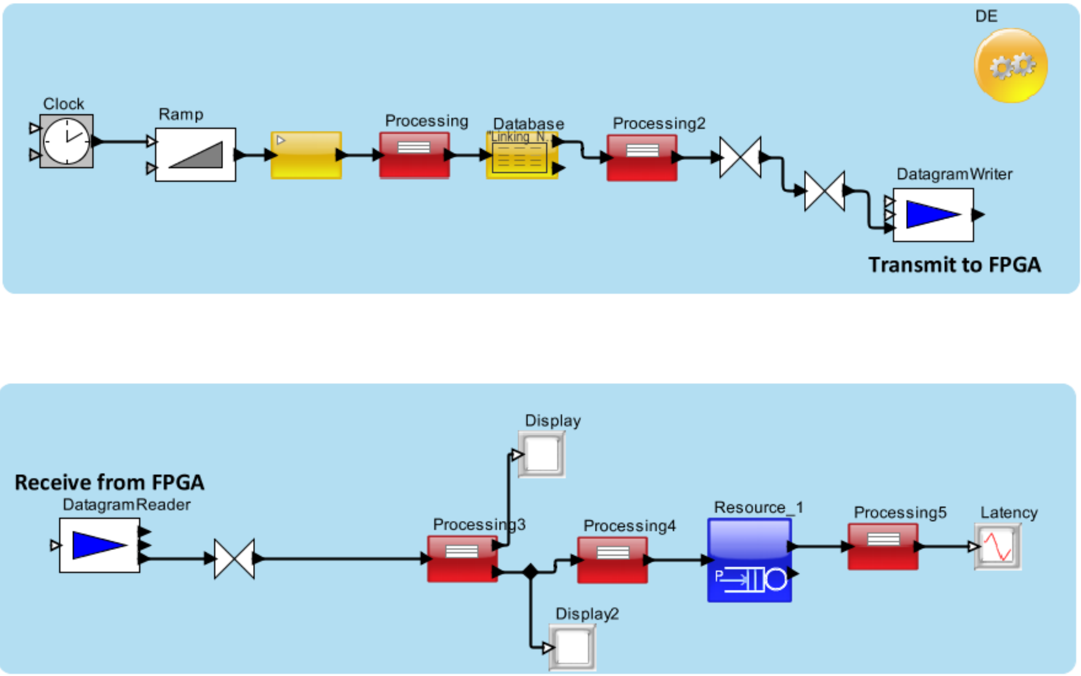
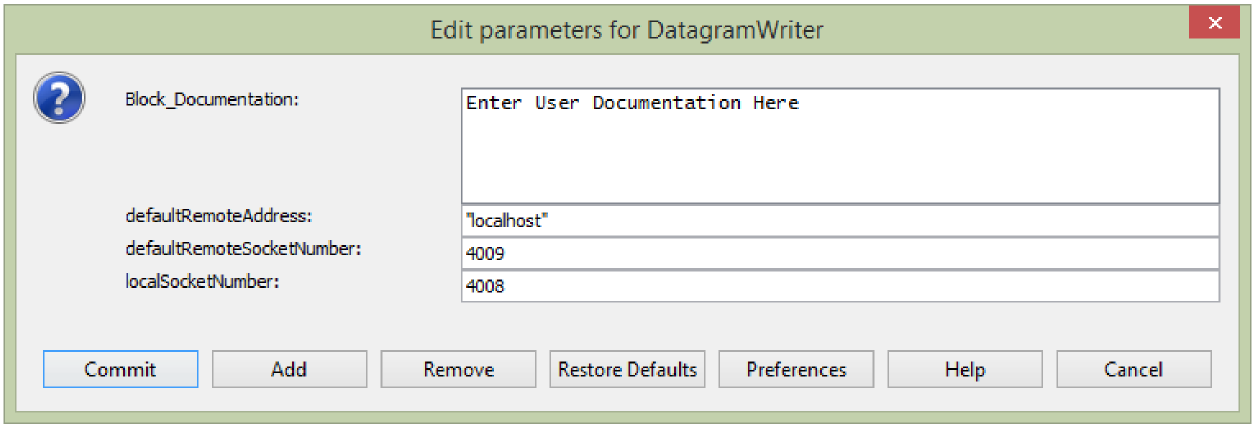


Figure 1: VisualSim Model

VisualSim Datagram Writer and Datagram Reader play a very important role in VisualSim FPGA interface. Datagram Writer and Datagram Reader should be configured with right Socket number and Remote/Local address.



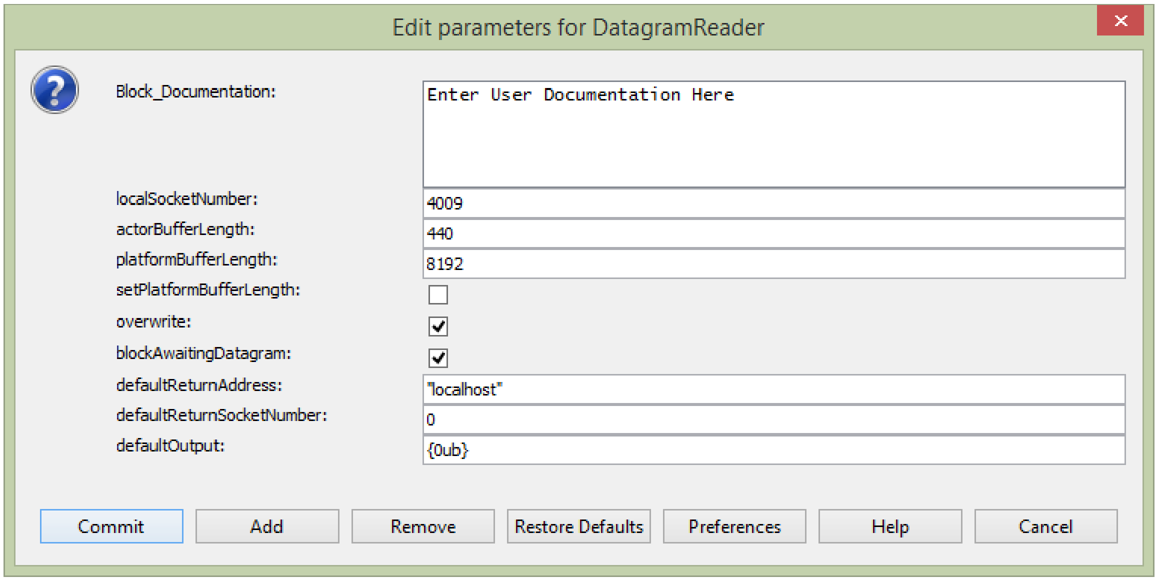
Block_Documentation: Enter User Documentation Here

defaultRemoteAddress: "localhost"

defaultRemoteSocketNumber: 4009

localSocketNumber: 4008

Figure 2: Configuring Datagram Writer block



Block_Documentation: Enter User Documentation Here

localSocketNumber: 4009

actorBufferLength: 440

platformBufferLength: 8192

setPlatformBufferLength: ☐

overwrite: ☑

blockAwaitingDatagram: ☑

defaultReturnAddress: "localhost"

defaultReturnSocketNumber: 0

defaultOutput: {0ub}

Figure 3: configuring Datagram Reader

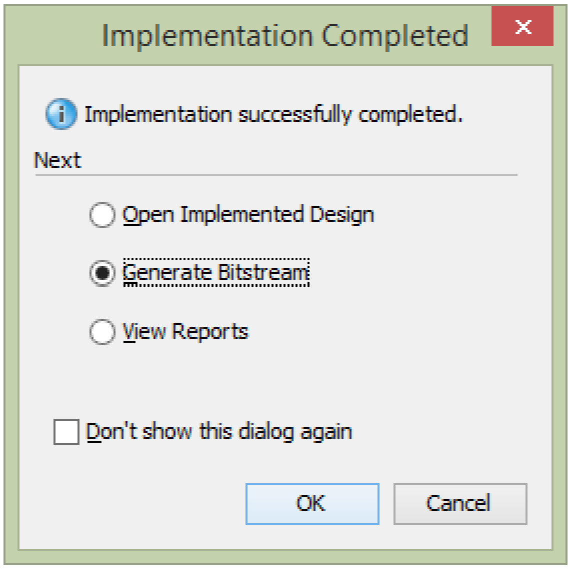
The blocks connected after Datagram Reader corresponds to the subsystems that are connected to FPGA platform and does processing based on the data from FPGA. In this example we have connected a Smart Timed Resource that models a simple buffer and it service that data based on the incoming data size and buffer speed MHz.

Configuring FPGA and about Source Files

We have created a sample Vivado project that works with VisualSim. Please import the Vivado project into Xilinx Vivado tool suite. To Import the project, please click on **File → Open Project** and select **mirab_eth.xpr** and **click Ok**. Now the project has been loaded into Vivado design suite.

Now we need to generate bit stream for programming FPGA. Click on Run Implementation from the menu bar **Flow → Run Implementation** or click on the green arrow button on the tool bar.

After successful Implementation Run generate bit stream by selecting Generate Bitstream Option as shown below



Implementation Completed

Implementation successfully completed.

Next

○ Open Implemented Design

● Generate Bitstream

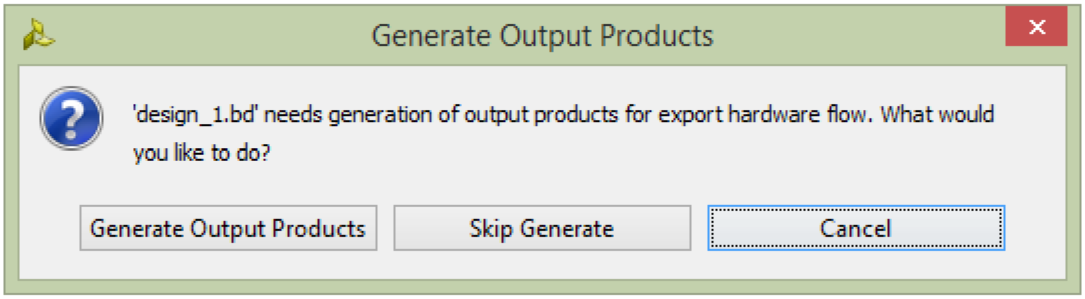
○ View Reports

□ Don't show this dialog again

Figure 4: Generate Bitstream

After successful Bitstream generation, we need to export hardware description file for SDK. To perform this, please click on **File → Export Hardware.** As the original project was compiled using Vivado 10.2, users with latest version of Vivado needs to upgrade IP and generate a new block diagram. Please refer appendix 1 for more details on updating IP and block diagram generation.

When you click on **Export Hardware,** Vivado Suite will as if you want to generate output products or to Skip generation of output products



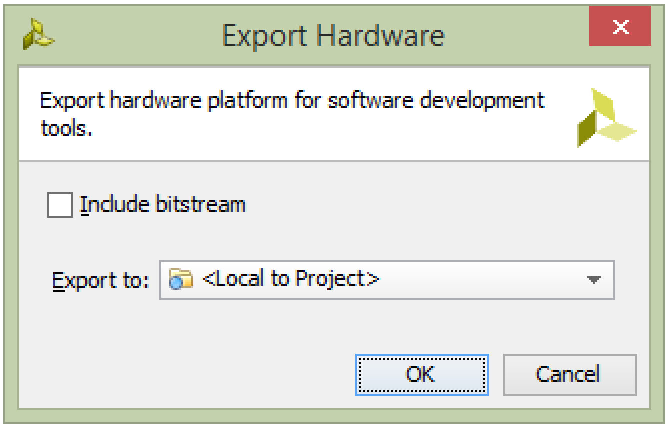
Generate Output Products

'design_1.bd' needs generation of output products for export hardware flow. What would you like to do?

Figure 5: Generate Output Products

Click on Generate output Products

After generating output products for hardware, select the export location as local to project


Export Hardware

Export hardware platform for software development tools.

□ Include bitstream

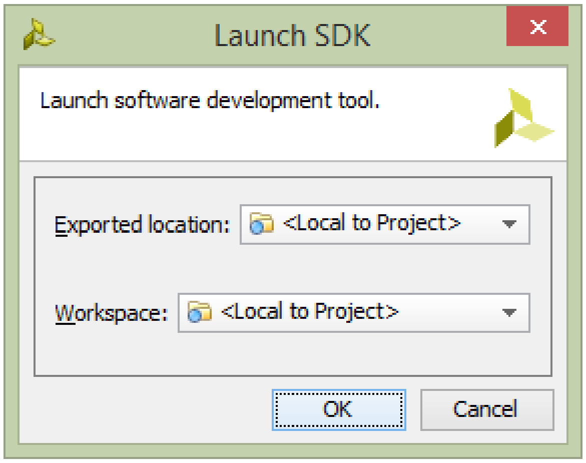
Export to: <Local to Project>

Figure 6: Export Hardware

Click on OK.

Now we need to launch Xilinx SDK, Please make sure that you have installed Xilinx SDK package during installation procedure.

As we need all the board support packages in SDK, launch Xilinx SDK from Vivado Design suite. Click on **File → Launch SDK**. This will open Xilinx SDK, Make the required changes in the SDK to match the paths of the new packages that were generated for the board being used. By default you can select the exported location as Local to project as shown below


Launch SDK

Launch software development tool.

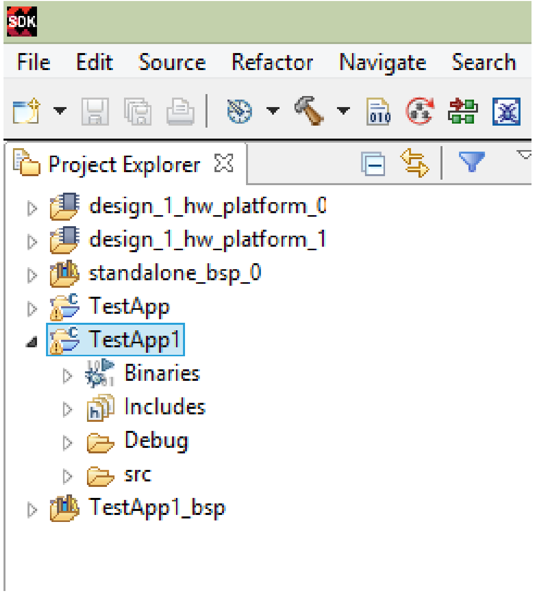
Exported location: <Local to Project>

Workspace: <Local to Project>

Figure 7: Launch SDK

Click on Ok.

The source files are listed under TestApp1, please select TestApp1 in Project Explorer as shown below.



To compile source file click on Run menu from the Menu bar or Run button on the Tool Bar.

Now program the FPGA with the generated bitstream by clicking on **Tools → Program Device**

Launch the software to run on the FPGA and monitor using a telnet from the host PC.

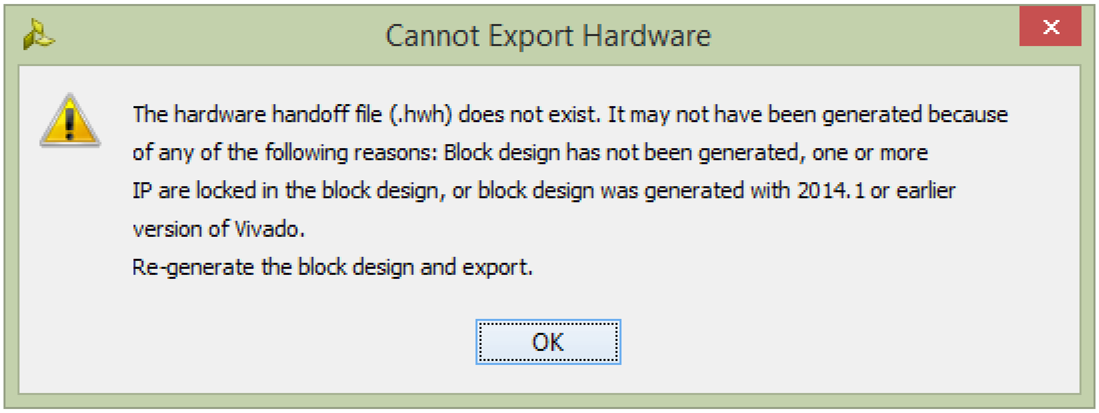
Possible Errors

1. Failed to create a new socket on port {Port Number}
   Solution: This error appears if the Datagram Writer local socket number and Datagram Reader local socket number are same. Please make sure that both are different
2. Blank Report display
   Solution: This is due to incorrect remote address/IP address or Remote socket number defined in Datagram Writer.

Appendix

1. For the users with Vivado design suite above 10.2, please follow the steps mentioned below to upgrade IP and to generate new block diagram.

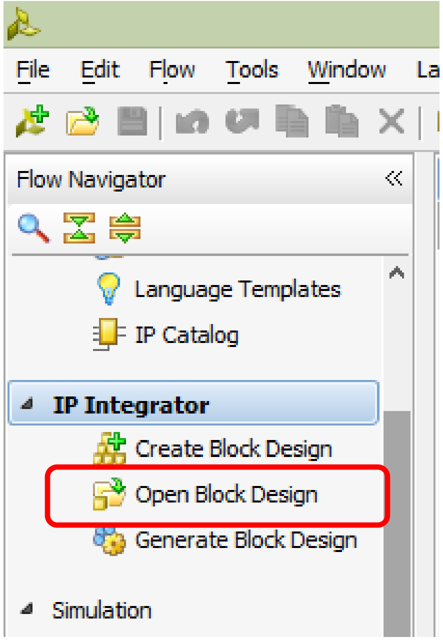
   If you are running Vivado design suite above 10.2 you may receive a message as mentioned below while performing **Export Hardware.**



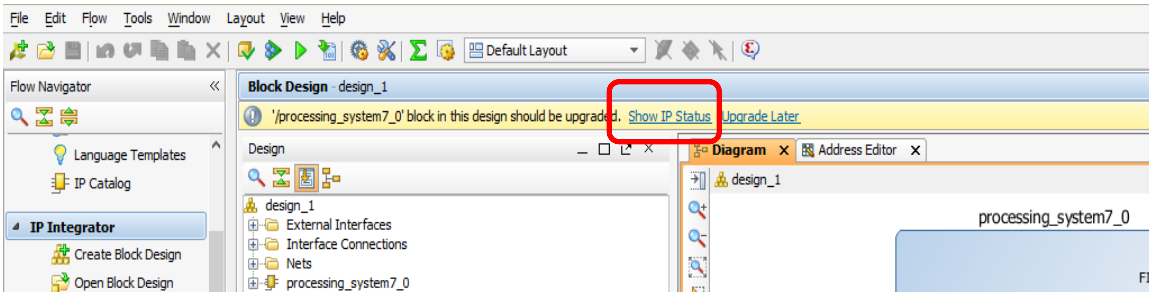
Cannot Export Hardware

The hardware handoff file (.hwh) does not exist. It may not have been generated because of any of the following reasons: Block design has not been generated, one or more IP are locked in the block design, or block design was generated with 2014.1 or earlier version of Vivado.
Re-generate the block design and export.

   At this point click on Open Block Design from the flow navigator as shown below

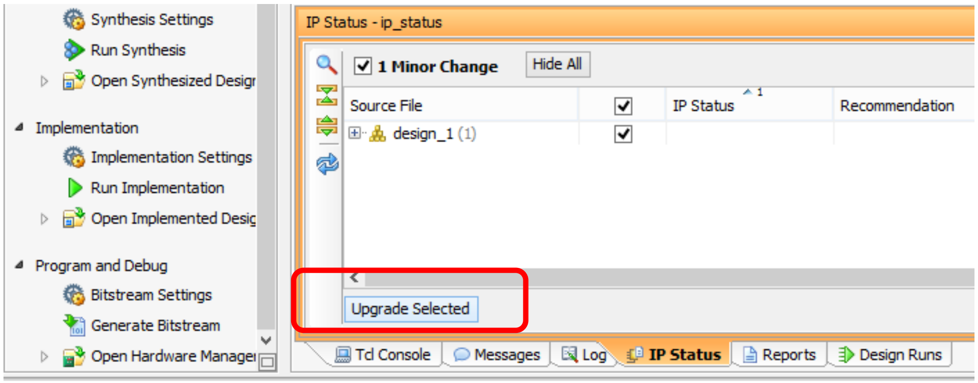


   Click on **Show IP Status**



'/processing_system7_0' block in this design should be upgraded. Show IP Status Upgrade Later

Click on **Update Selected**



IP Status - ip_status

Source File	☑	IP Status	Recommendation
design_1 (1)	☑		

This will update the existing IP's