Ultra96: Creating a Zynq UltraScale+ MPSoC Hardware Platform in Vivado

Overview

With a traditional processor, the hardware platform is pre-defined. The manufacturer selected the processor parameters and built-in peripherals when the chip was designed. To make use of this pre-defined processor, you need only target that specific hardware platform in the software development tools.

The Zynq® UltraScale+™ MPSoC is different. ZU+ provides multiple building blocks and leaves the definition to you as the design engineer. This adds flexibility, but it also means that a bit of work needs to be done up front before any software development can take place.

The first step in completing a ZU+ design is to define and build the hardware platform in the Vivado® Design Suite. The purpose of this tutorial is to show you how to quickly and easily create a base hardware platform for Ultra96.

Objectives

When this tutorial is complete, you will be able to:

- Create a new project in Vivado, targeting Ultra96 and the ZU+ MPSoC
- Create a block based design to insert an ARM processing system
- Import the Ultra96 Preset settings
- Build and export the hardware platform

Experiment Setup

Software

The software used to test this reference design is:

- Windows-7 64-bit
- Xilinx Vivado 2018.2
- Board Definition File (BDF) Install from the Avnet BDF Repository on GitHub.
  - GitHub: https://github.com/Avnet/bdf
  - Instructions: http://www.zedboard.org/support/documentation/24166

Hardware

The hardware setup used to test this reference design includes:

- Win-7 PC with the following recommended memory[1]
  - 4 GB Typical and 5 GB Peak RAM available for the Xilinx tools to complete a XCZU3EG design

[1] Refer to https://www.xilinx.com/products/design-tools/vivado/memory.html

Experiment 1: Create a New Ultra96 Project in Vivado

The Ultra96 development board is supported by Vivado WebPack (which is free). Ultra96 also ships with a Design Edition and SDSoC license voucher. The Processing System (PS) may be used without anything programmed in the Programmable Logic (PL). This PS-only style is the simplest way to use Zynq, so that is what we will do during this lab. However, the power of ZU+ is found in using soft IP in the PL, interconnecting PS to PL, and routing extra PS built-in peripherals through EMIO to PL I/Os, and then programming of the PL is required.

This tutorial will take advantage of built-in 3rd-party board definition files. You can find the instructions here:

- www.Ultra96.org → Support → Documentation → Ultra96 → Installing Board Definition Files

1. If not previously completed, download the Board Definition Installation instructions to install the board definitions. An Ultra96 board definition is built into Vivado 2018.2, but it has a bug. Make sure that your Vivado Install shows Ultra96's 1.2 folder:

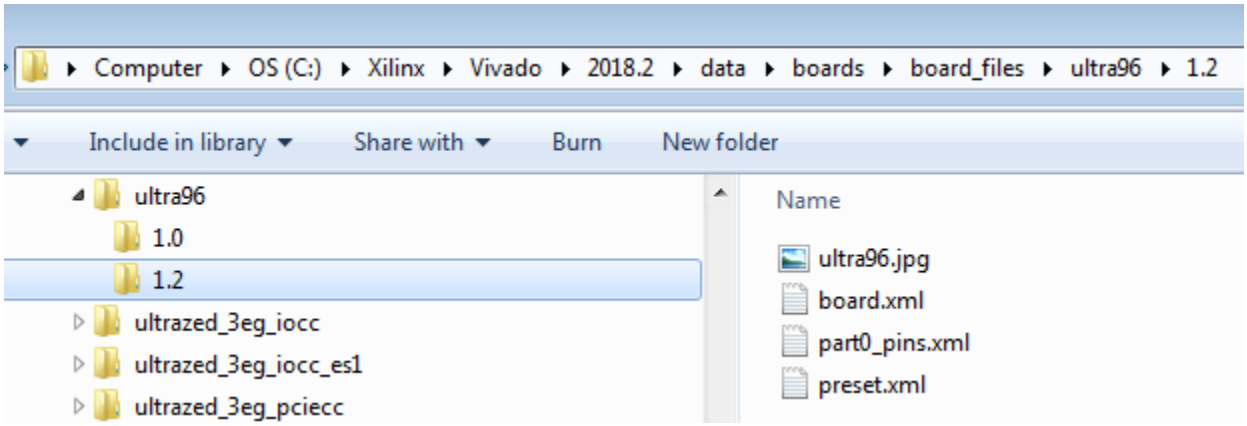

Figure 1 – Ultra96 v1.2 BDF Installed

2. Launch Vivado by selecting **Start → All Programs → Xilinx Design Tools → Vivado 2018.2 → Vivado 2018.2**.

3. Select **File → New Project** or click on **Create New Project** under *Quick Start*.

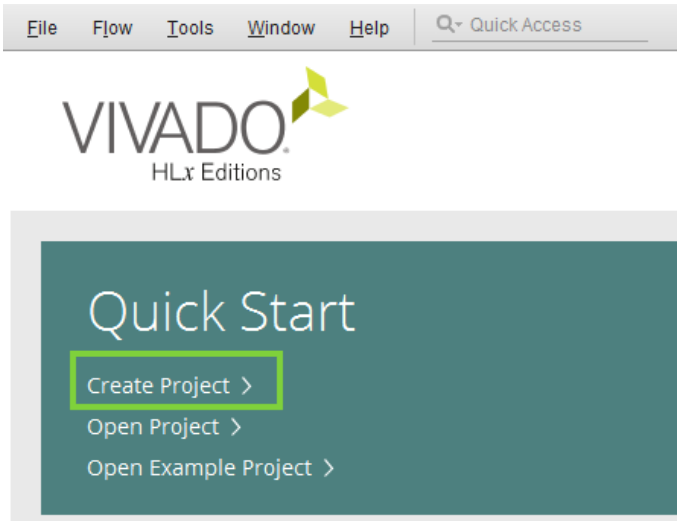

Figure 2 – Create New Vivado Project

4. Click **Next >**.

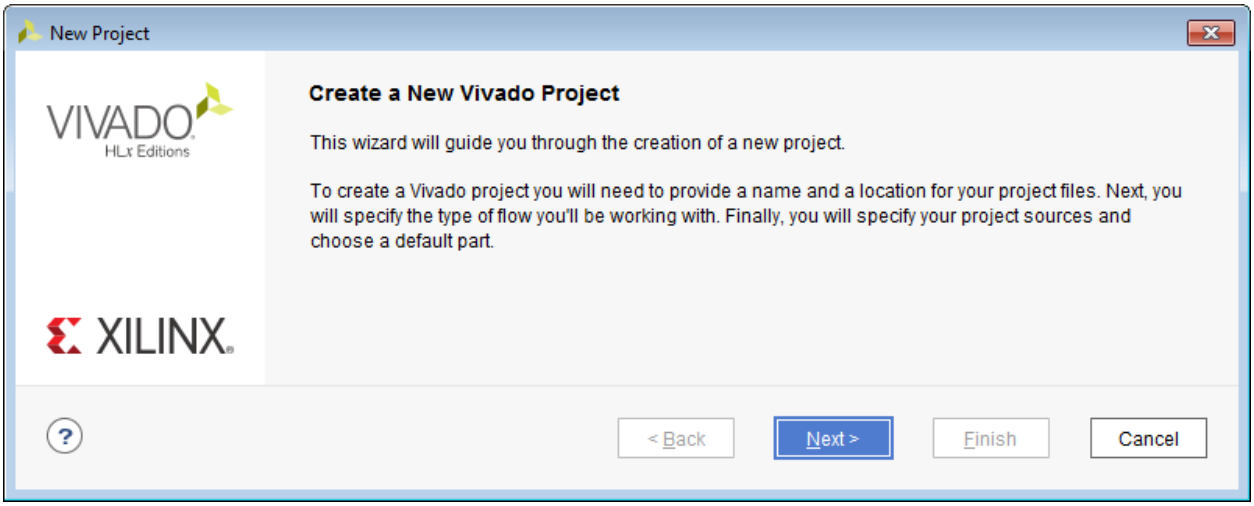

Figure 3 – New Vivado Project Wizard Launched

5. Click the browse icon  . Browse to set the *Project location* to your desired project location and click **Select**. Set the *Project name* to **Ultra96_Basic_System**. Also verify the ***`Create project subdirectory`*** checkbox is selected. Click **Next >**.

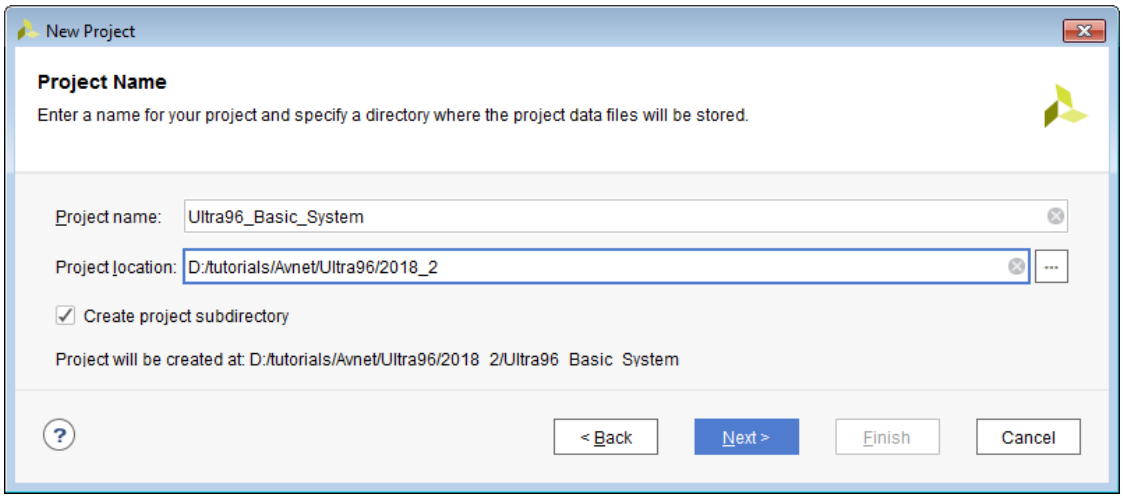

Figure 4 – Set Project Name and Location

6. The project will be RTL based, so leave the radio button for *RTL Project* selected. Since this is a brand new project, check the box for ***Do not specify sources at this time***. Click **Next >**.

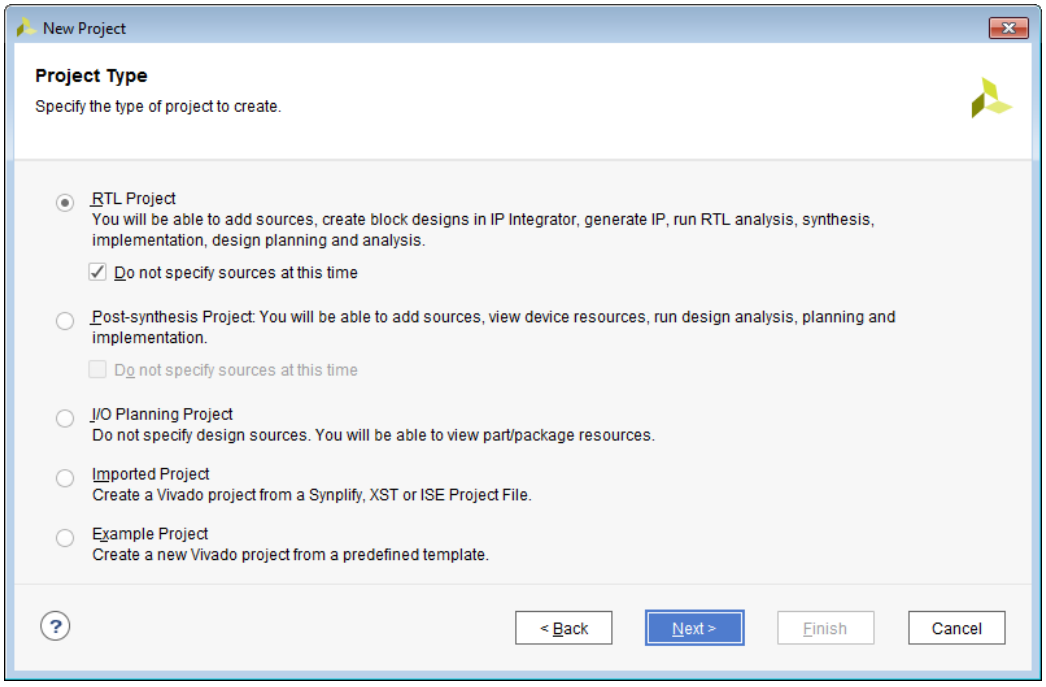

Figure 5 – Set Project Type

Next, the **Default Part** is selected. This can be done by specifying a specific part or by selecting a board. If you have installed the Board Definition archive correctly, you will have access to the Ultra96 BDF.

7. In the *Select* area, select **Boards**.

8. Set the *Vendor* to **em.avnet.com**. Scroll until you find **Ultra96 Evaluation Platform**. The *File Version* should be **1.2** and the *Board Rev* should be Rev 1.

9. Single-click the **Ultra96** Click **Next >**.

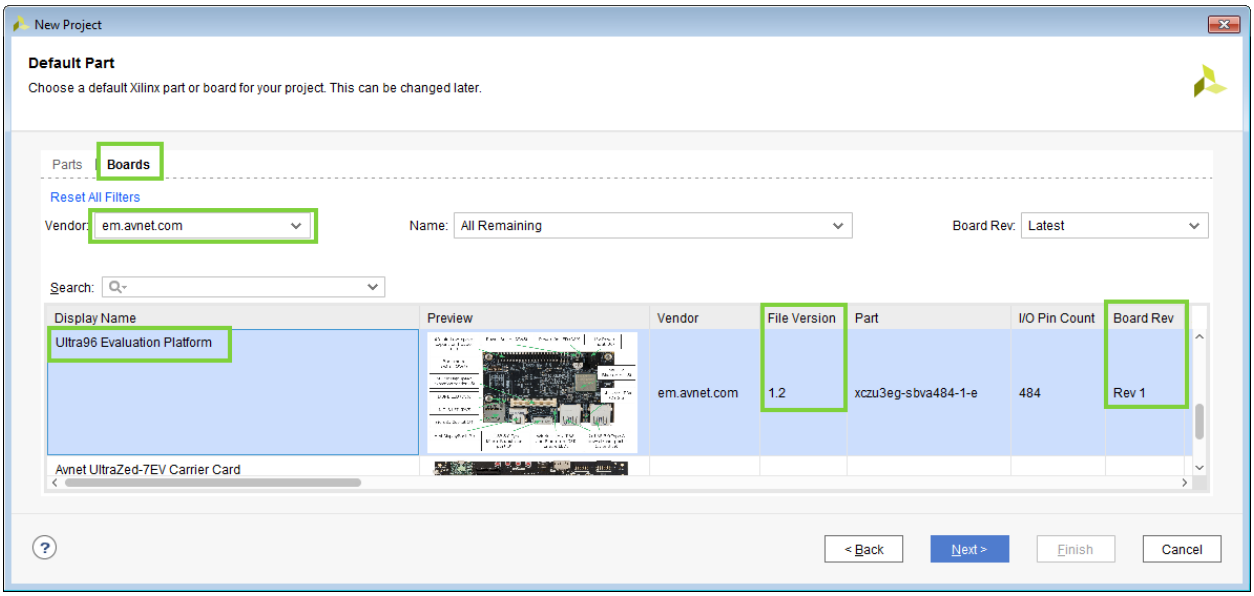

Figure 6 – Select the Target Board

10. A project summary is displayed. Click **Finish**. The Vivado cockpit is now displayed.

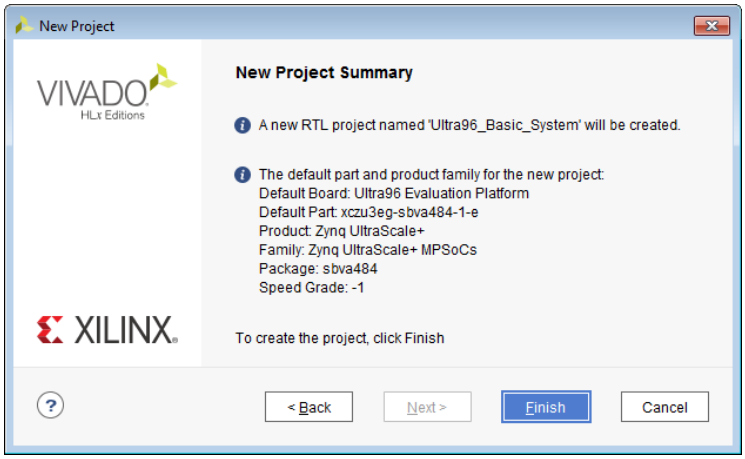

Figure 7 – New Project Summary

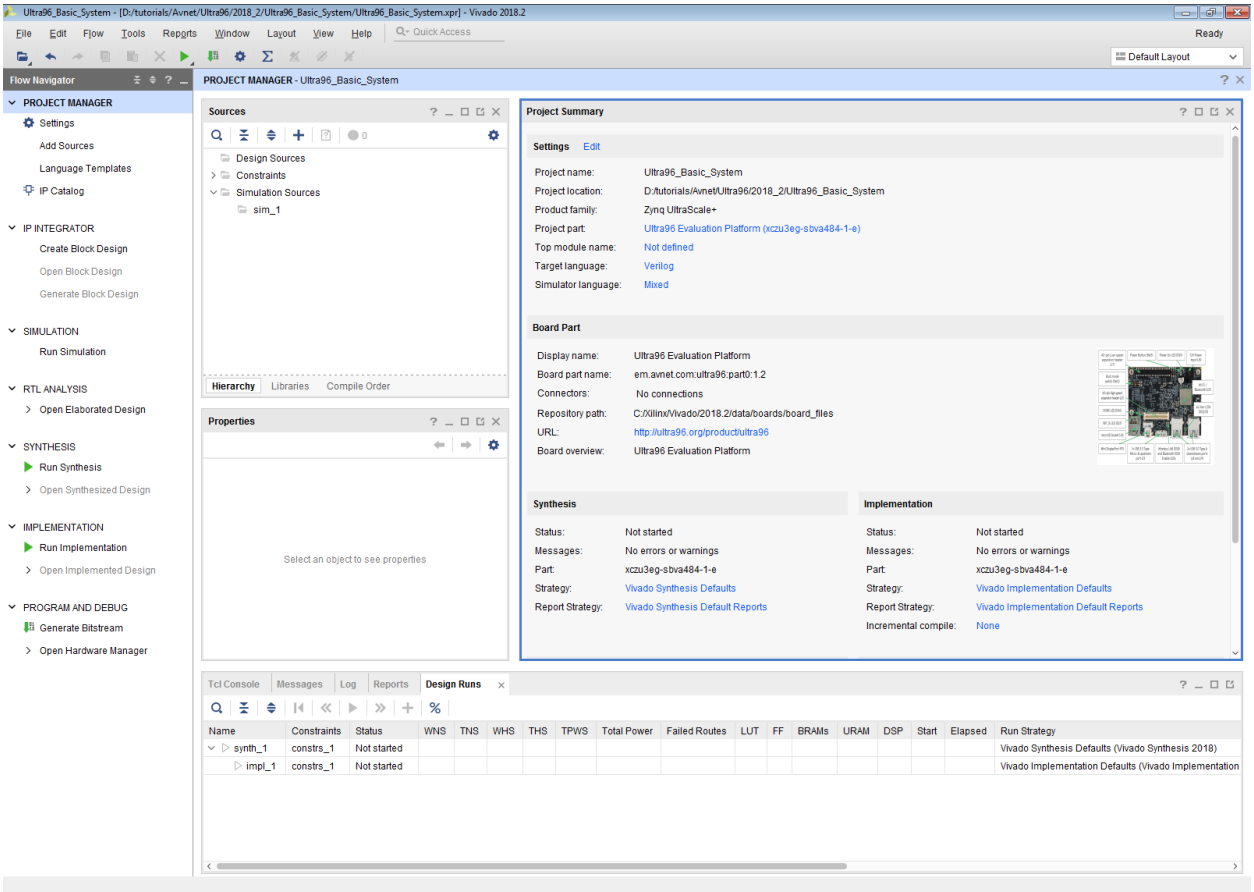

Figure 8 – Vivado Cockpit

Experiment 2: Create and Edit a Block Design

The current project is blank. To access the ARM processing system, we will add an embedded source to the Vivado project using IP Integrator.

1. The recommended way to add an embedded processor is through the Block Design method via IP Integrator. Select **Create Block Design**.

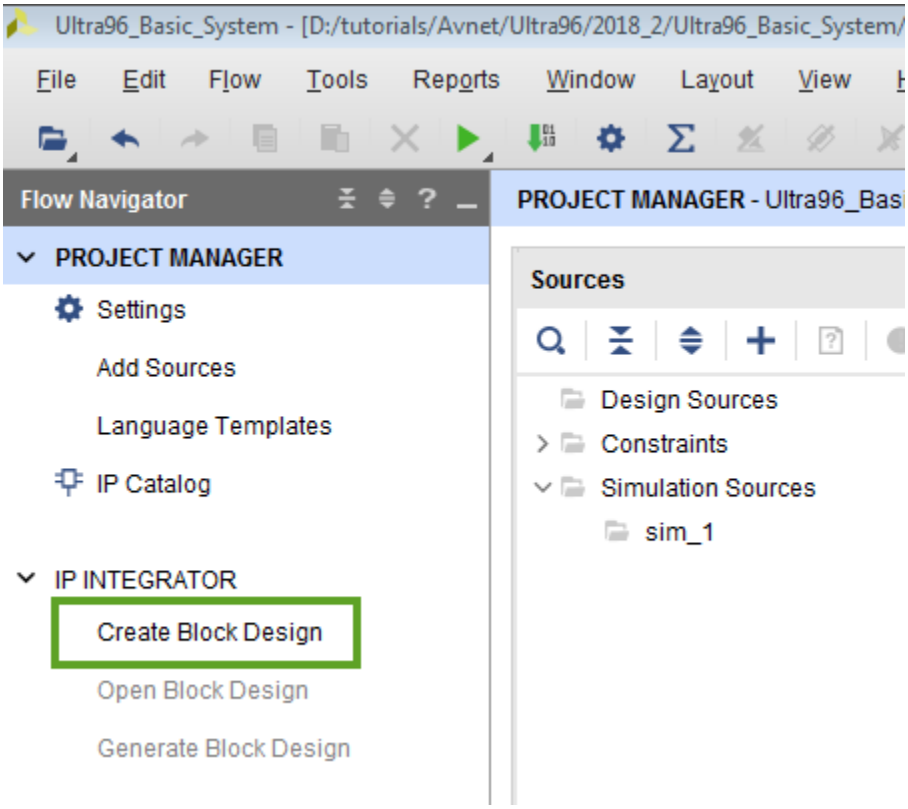

Figure 9 – Create Block Design

2. Give the Block Design a name or use the default name of *design_1*. Click **OK**.

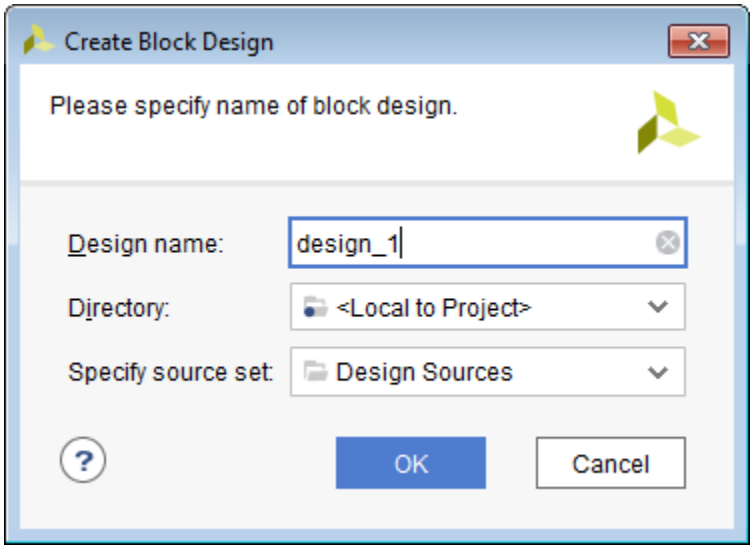

Figure 10 – Block Design Name

3. In the Diagram window, click the **Add IP** text icon + in either location.

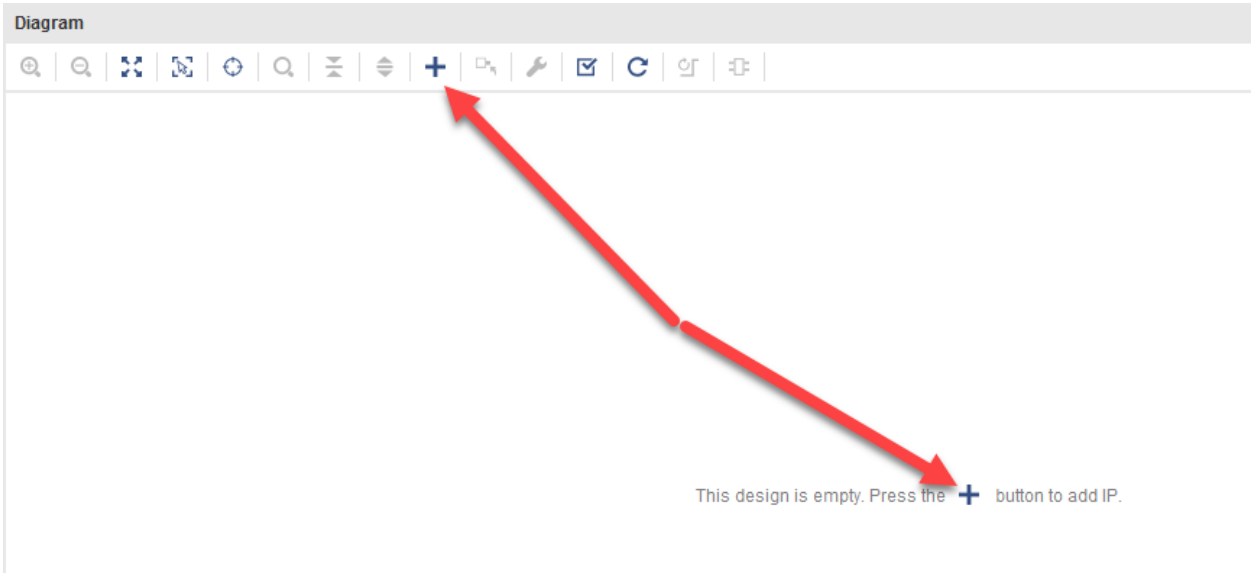

Figure 11 – Add IP to the Block Design

4. The *Add Sources* window opens. Start typing "Zynq" in the search window. Find the **ZYNQ UltraScale + MPSoC** IP. Either double-click this or drag and drop to the *Diagram* window.

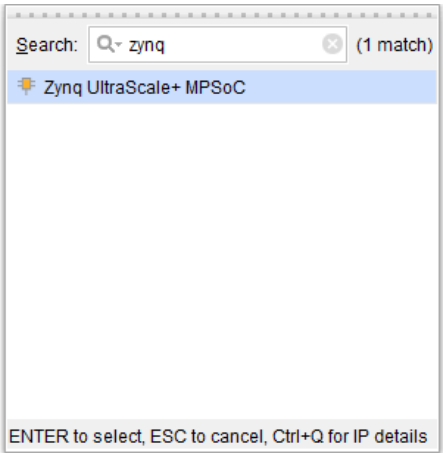

Figure 12 – Add IP Window

The Zynq Processing system will appear in the *Diagram* window. Also a new tab will appear labeled **Address Editor**.

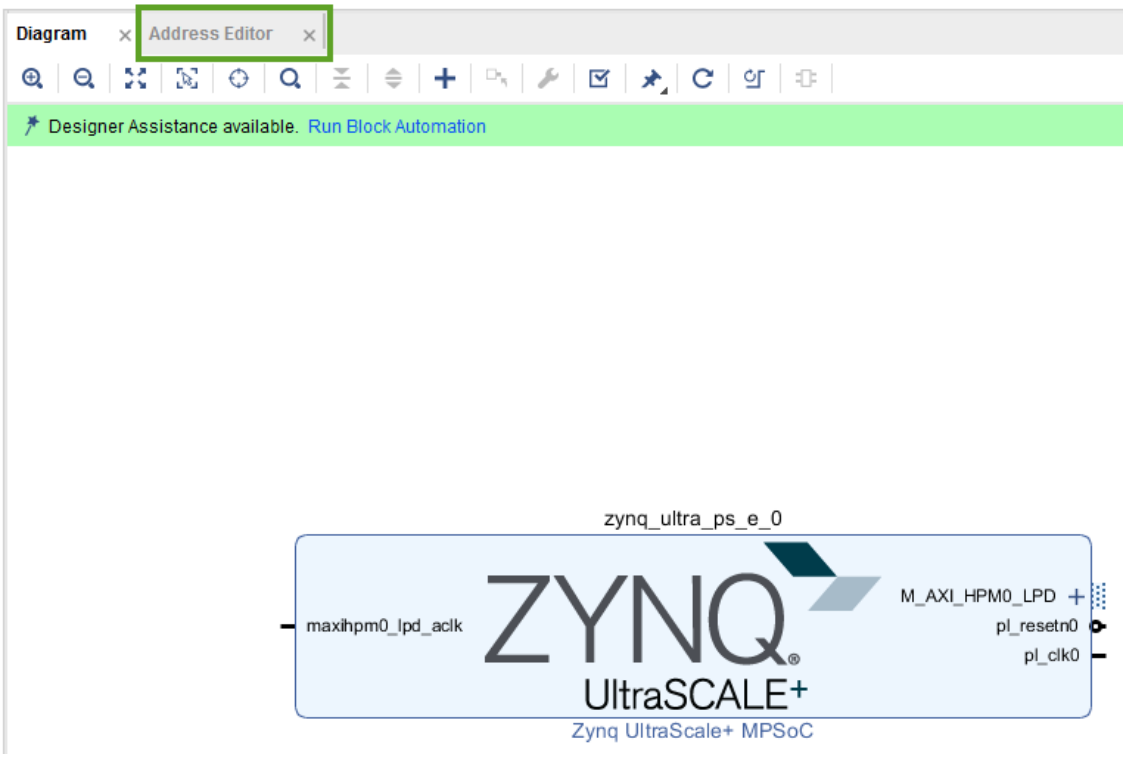

Figure 13 – Updated Block Diagram

5. Similar to the *Add IP* prompt in the previous step, notice now that the *Designer Assistance* has provided the hint to *Run Block Automation*. Click the **Run Block Automation** link at the top of the window.

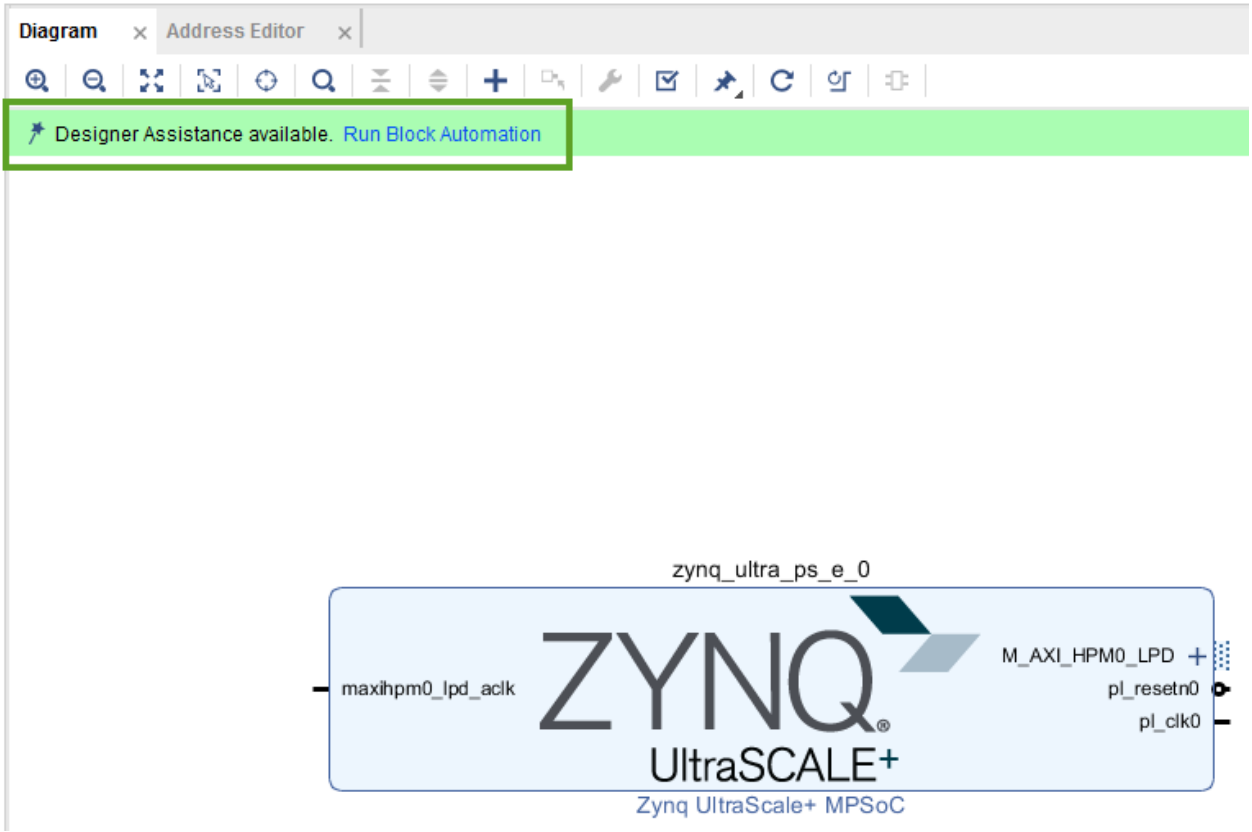

Figure 14 - Run Block Automation

6. The *Apply Board Preset* checkbox applies the Preset TCL that was included as part of the board definition archive. Leave this checked. For details about how to build a system manually, please see the *Developing Zynq Hardware Speedway*. Click **OK**.

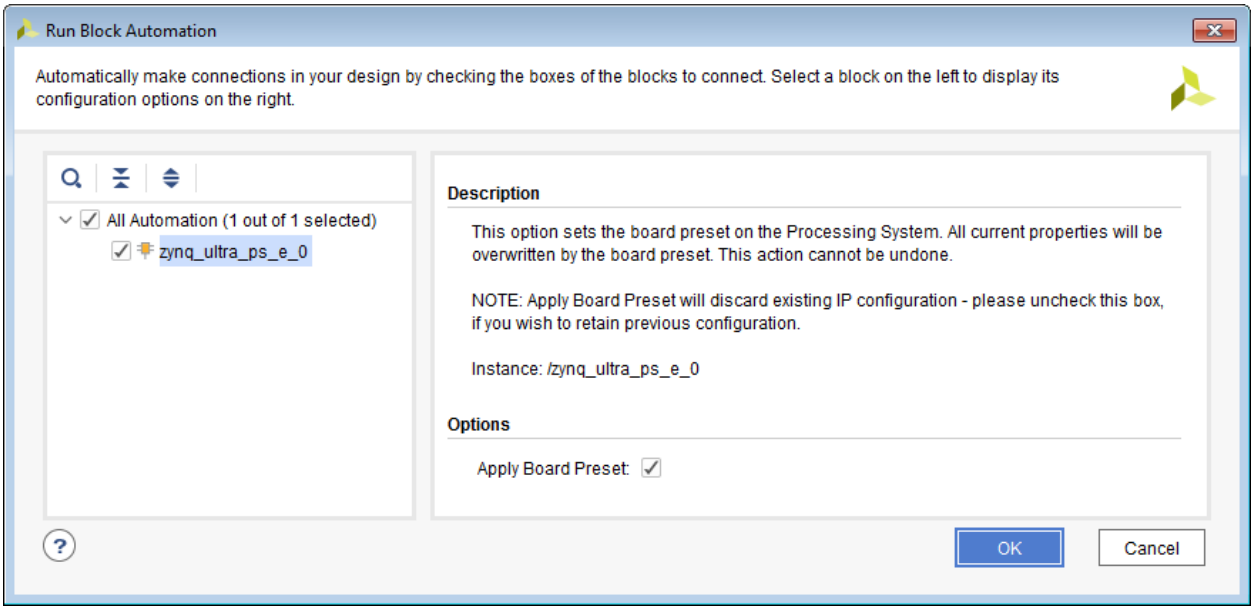

Figure 15 – Run Block Automation

You will now see the ZU+ MPSoC block.

Figure 16 - Zynq Block Diagram for Ultra96 with External I/O

7. The maxihpm clocks must be connected. You can do this by clicking and dragging from the pl_clk0 pin to each maxihpmx_fpd_aclk pin, or you can run the commands below in the Tcl Console:

- `connect_bd_net [get_bd_pins zynq_ultra_ps_e_0/pl_clk0] [get_bd_pins zynq_ultra_ps_e_0/maxihpm0_fpd_aclk]`
- `connect_bd_net [get_bd_pins zynq_ultra_ps_e_0/pl_clk0] [get_bd_pins zynq_ultra_ps_e_0/maxihpm1_fpd_aclk]`

Figure 17 – Clocks Connected

8. At this point, we can **validate** our design. Click the Validate Design icon . A successful validation window will appear. Click **OK**.

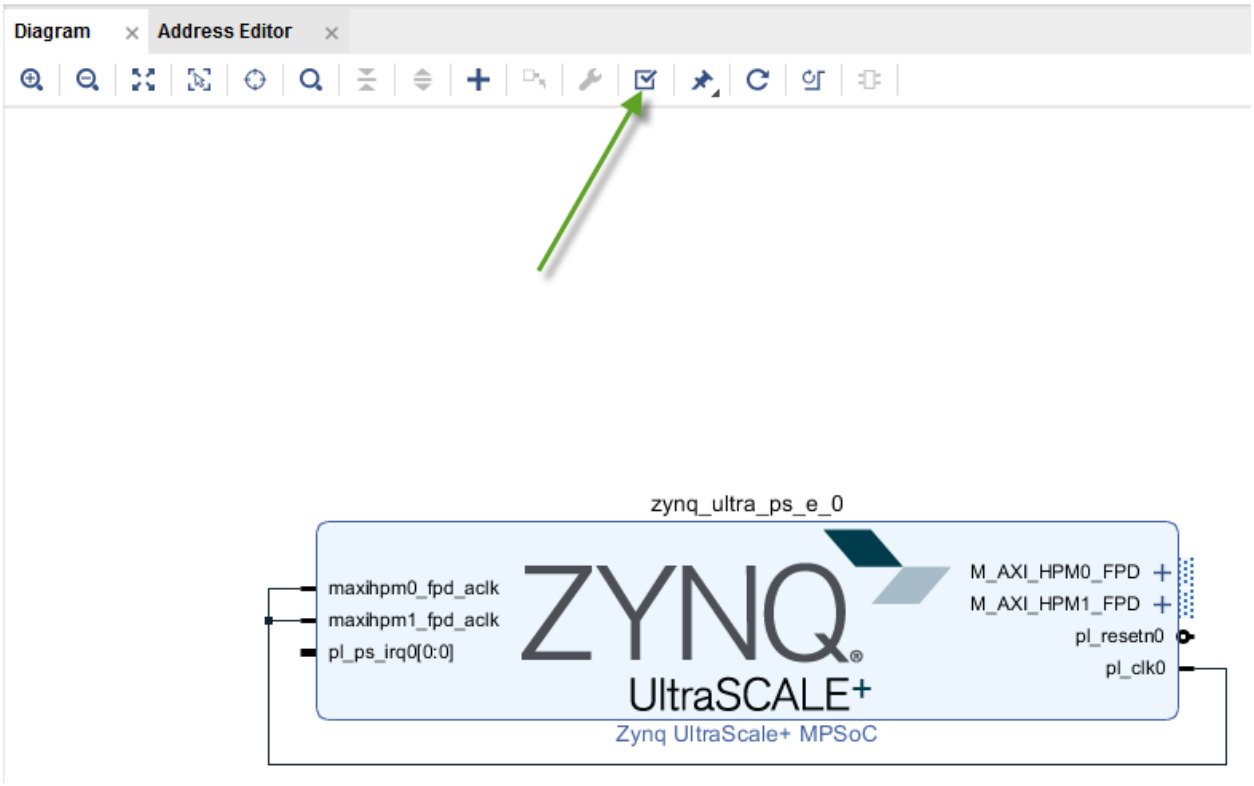

Figure 18 - Validate Zynq Block Design

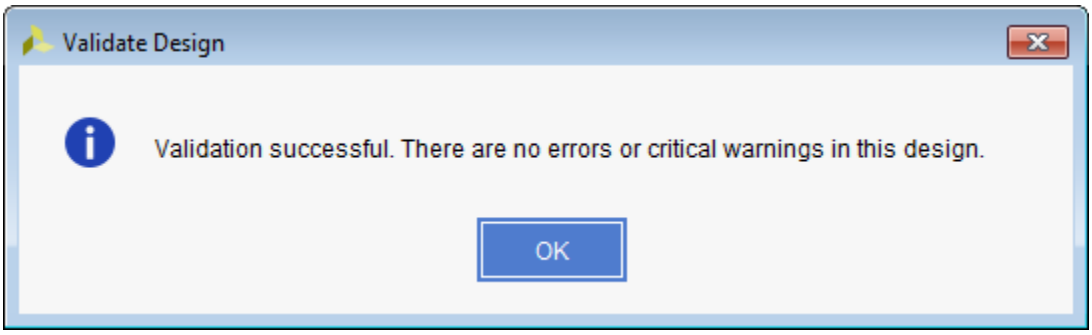

Figure 19 – Validation Successful

9. Click **Save Block Design** icon, , to save the project.

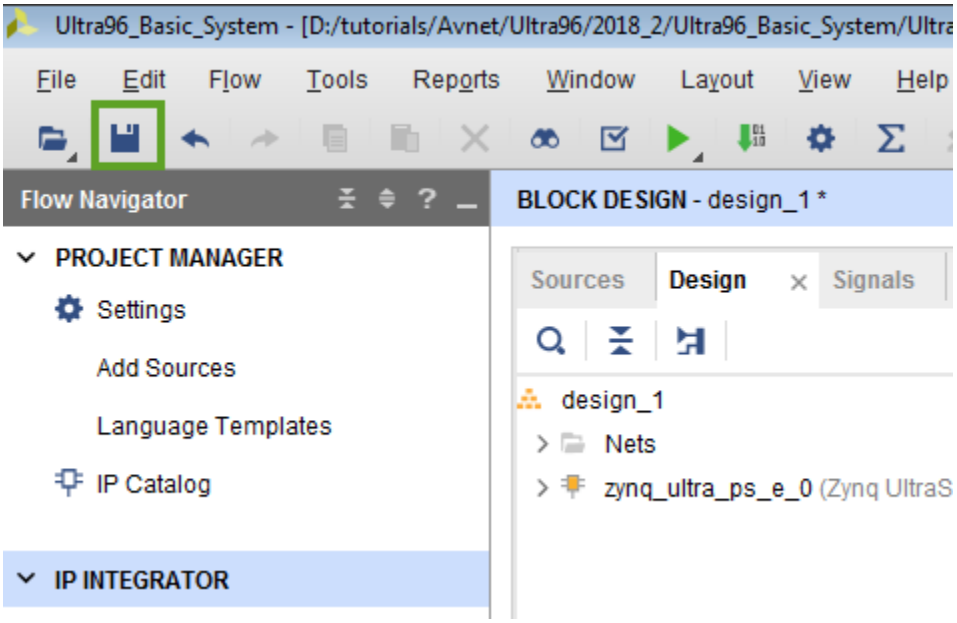

Figure 20 - Save Block Design

10. Switch to the **Sources** tab by clicking on it.

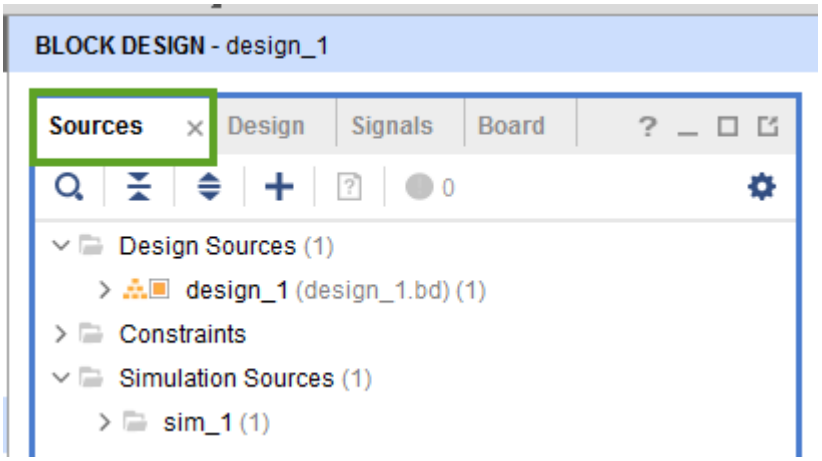

Figure 21 – Sources Tab

11. Right-click on **System (System.bd)** and select **Create HDL wrapper**.

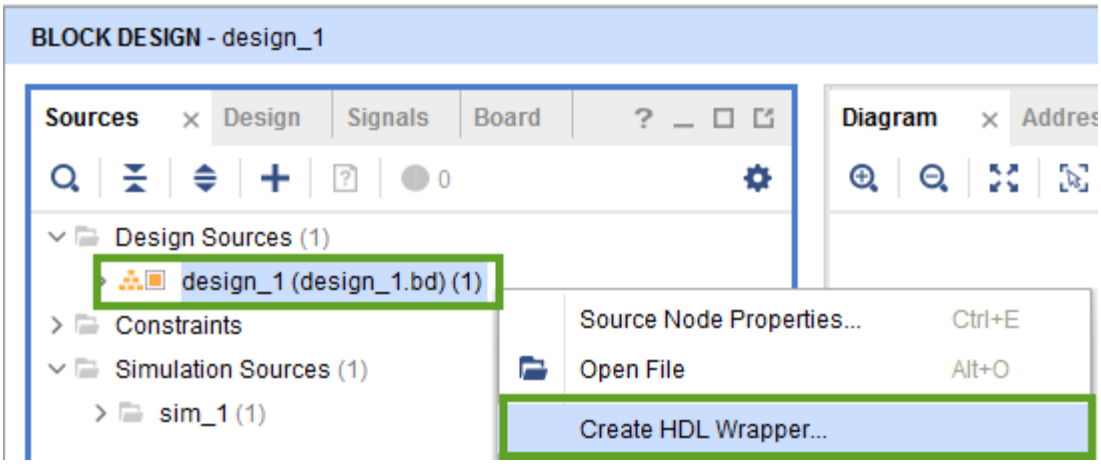

Figure 22 - Create Top Level HDL Wrapper

12. For now, leave the option selected to *Let Vivado manage wrapper and auto-update*. Click **OK**.

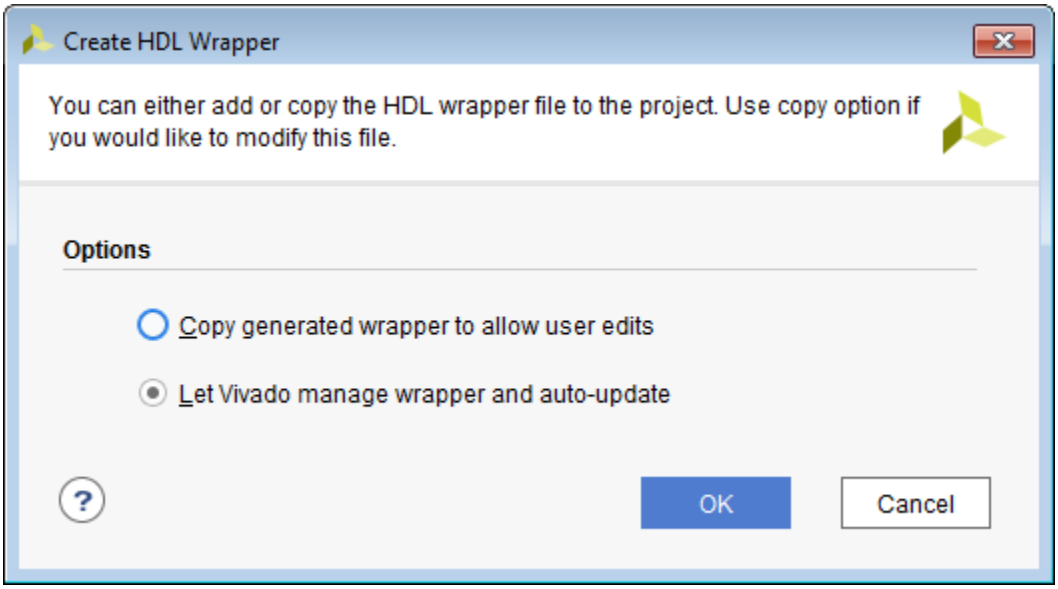

Figure 23 – Let Vivado Manage Wrapper

13. Once the top-level wrapper is created, you can see the design hierarchy in the *Sources* tab. Notice that **design_1_wrapper.v** is the top-level HDL wrapper that was created. **design_1.bd** is the Block Design.

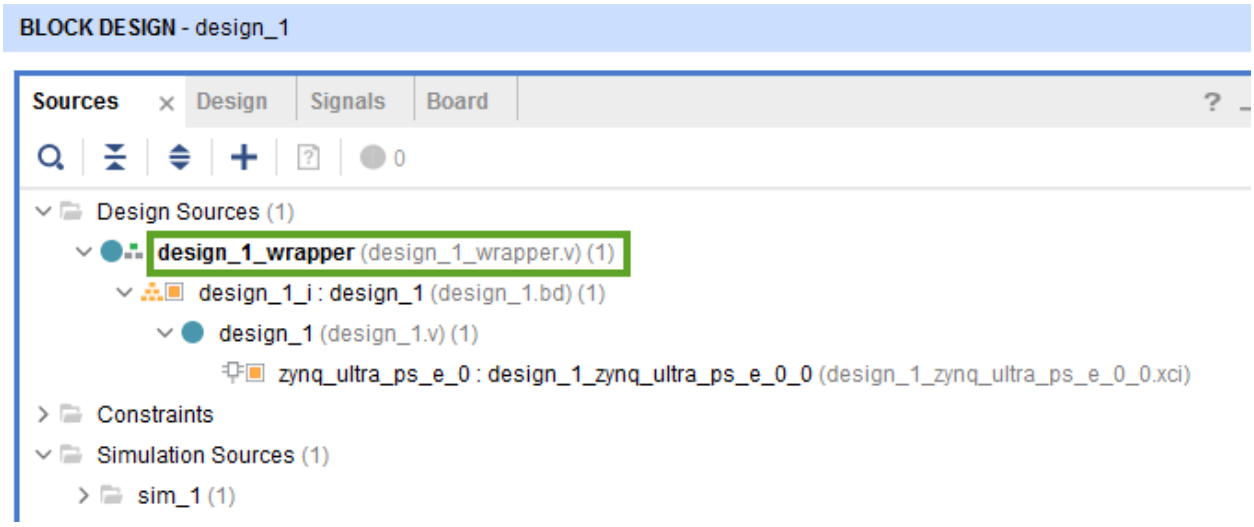

Figure 24 – System_wrapper.vhd Generated

14. Click **Generate Bitstream** in the *Flow Navigator* window.

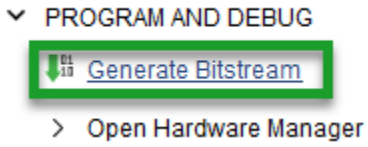

Figure 25 – Generate Bitstream

15. Click yes to start Synthesis and Implementation flows.

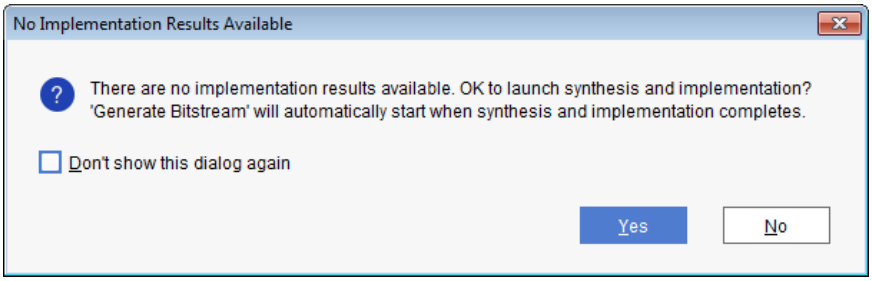

Figure 26 -- Launch Synthesis and Implementation

16. You can adjust your *Number of jobs* to as many processing cores as your host has. Then click **OK**.

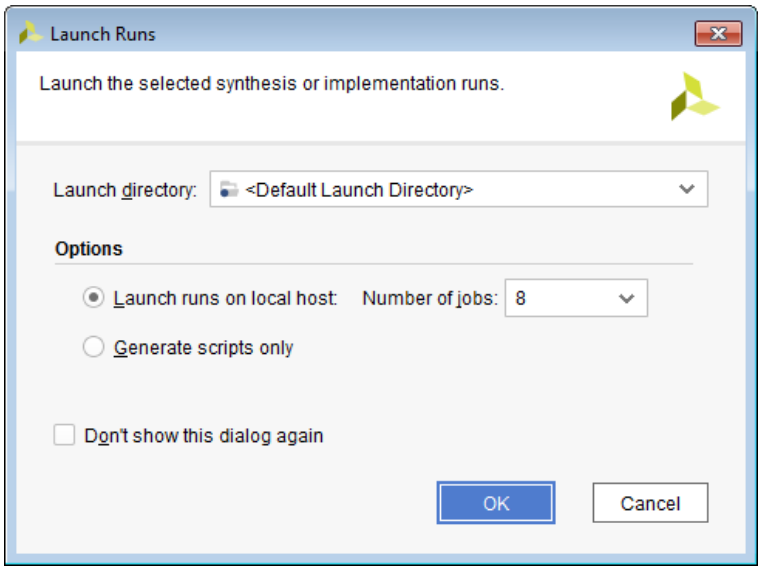

Figure 27 – Launch Runs

17. A large status window will appear. When that disappears, you can continue to monitor the status in the upper right corner of the Vivado cockpit.

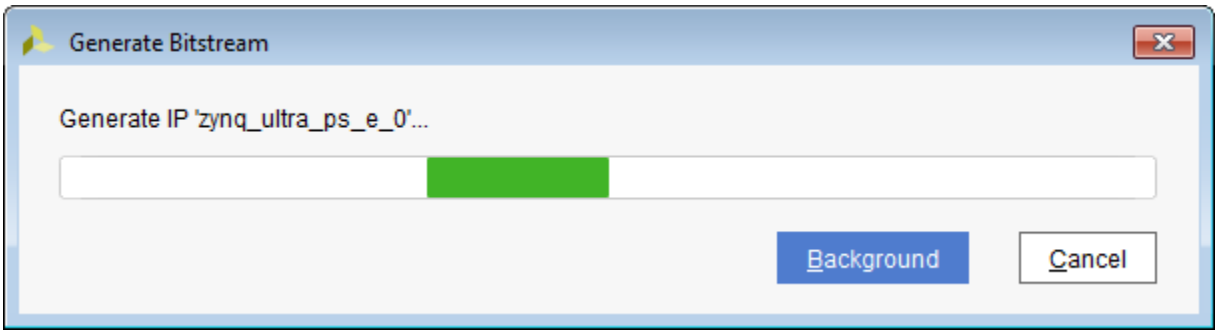

Figure 28 – Status of Generate Bistream

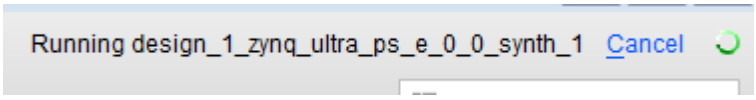

Figure 29 – Progress Status

18. When bitstream generation is completed, click **OK** to *Open Implemented Design*.

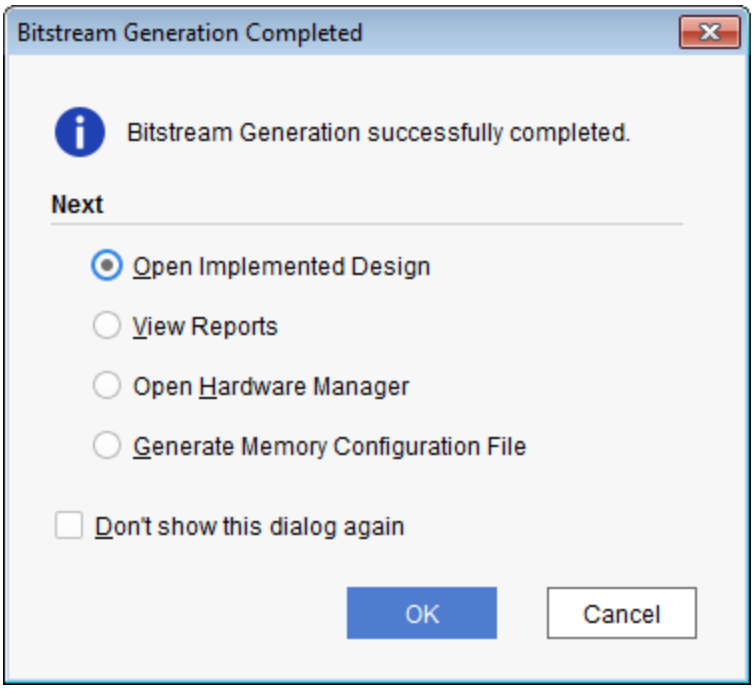

Figure 30 – Open Implemented Design

Experiment 3: Export Hardware Platform to SDK

Now that we've created an embedded system, we must make this platform available to the Software Development Kit (SDK). This is done by exporting the hardware platform.

1. In the Vivado tool, select **File → Export → Export Hardware**. Check the box to ***Include bitstream***. Click **OK**. You could specify a different directory, but for now, leave it as *Local to Project*.

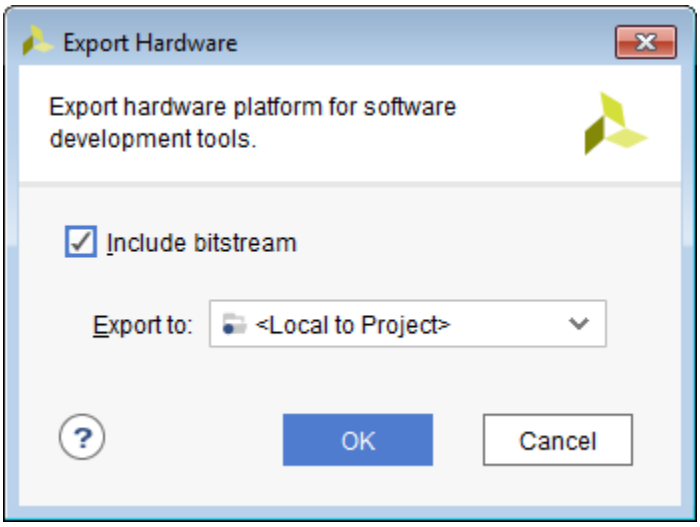

Figure 31 – Export Zynq Hardware Platform

2. We will now explore what you have created. In Windows Explorer, browse to your project directory.

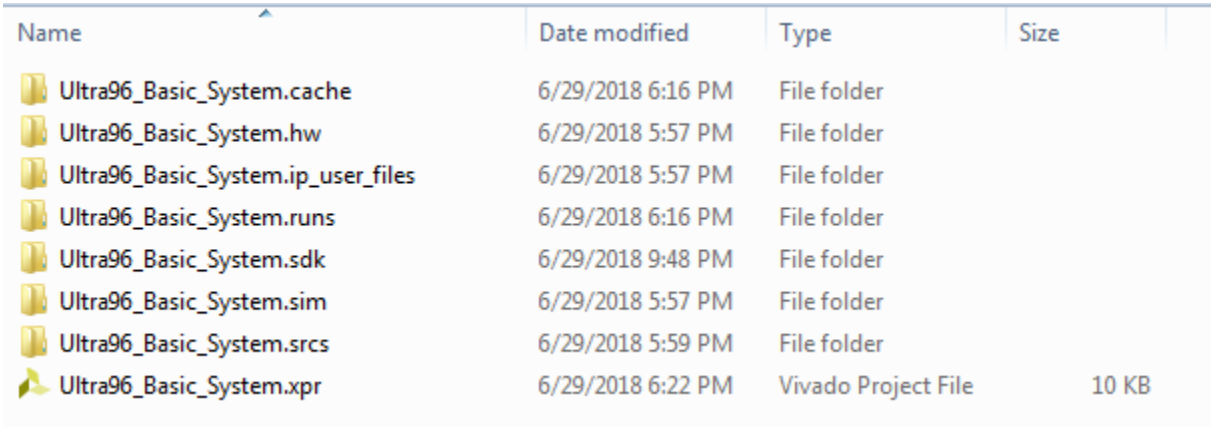

Figure 32 – Project Directory Contents after Export to SDK

You will notice seven directories and one file here. The .xpr file is your Vivado Project File and can be used to re-launch your project when you come back to work on it some more.

The .cache, .hw, .runs, .sim, and .srcs directories contain everything related to the hardware design, including the block design source, wrapper HDL, and synthesis/implementation results.

The .sdk folder is the result of the ***Export Hardware*** operation. Everything required for SDK to import the hardware platform is contained inside one file inside this directory. A hardware engineer looking to share the design with the software team could provide this one file. This provides a very compact and portable method to send a ZU+ Hardware Platform to a colleague.

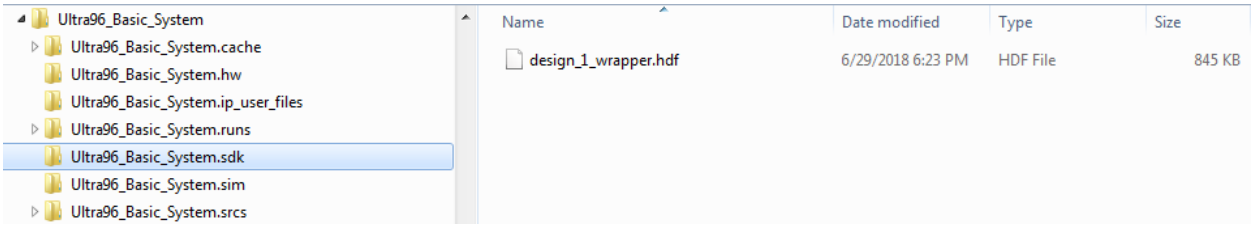

Figure 33 – ZU+ Hardware Platform Export for SDK

The next tutorial will show you how to open SDK, import a hardware platform, and run Hello World.

Revision History

Date	Version	Revision
29 Jun 2018	2018_2.01	Initial Avnet release for Vivado 2018.2