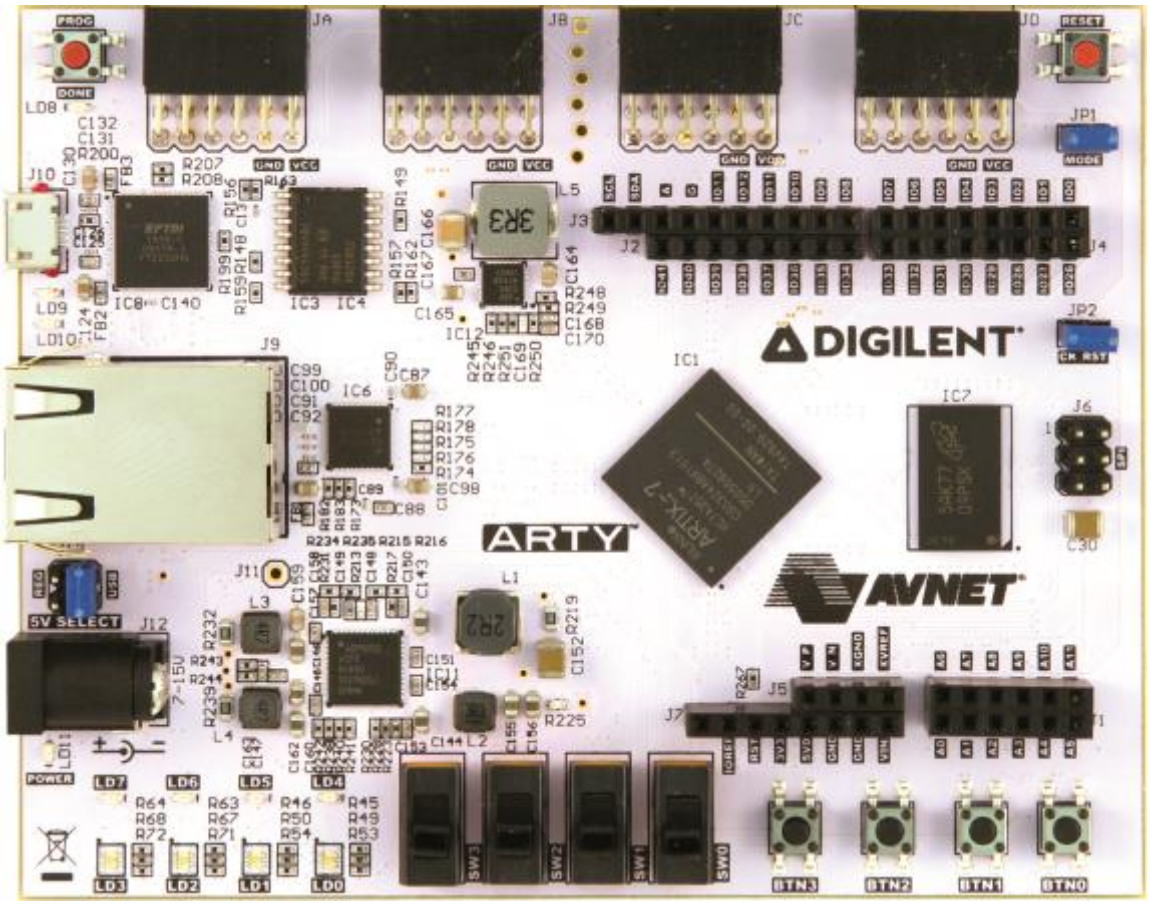


Out-Of-Box GPIO Demo Example Design for the Arty Evaluation Board

Version 1.0

LIT# 5184-7A35T_Arty_OOB_GPIO_demo_V1

Document Control

Document Version: 1.0

Document Date: 8/12/2015

Prior Version History

Version	Date	Comment
1.0	8/12/2015	Initial release for Vivado 2015.2

Contents

1 Overview

This document describes a simple MicroBlaze™ design implemented and tested on the Avnet/Digilent Arty Evaluation Board. This example design integrates pushbutton and DIP switch user input with a custom Pulse Width Modulator (PWM) peripheral to manipulate the brightness and display pattern of LEDs on the board This example design demonstrates continuously reading the DIP switches and using that value to manipulate the brightness of LEDs on the board via a custom PWM peripheral. The lower the value read from the DIP switches, the dimmer the LEDs become.

This tutorial shows how to build a MicroBlaze Hardware Platform and then create, build, and run a software application on the Avnet/Digilent Arty Evaluation Board.

2 Objectives

When you have completed this tutorial, you will know how to do the following:

- Build a MicroBlaze hardware platform integrating a custom IP peripheral.
- Set up an SDK workspace.
- Add an example software application.
- Run the example hardware and software design to manipulate the LED brightness.
- Program the QSPI Flash memory.

3 Reference Design Requirements

The following items are required for proper completion of this tutorial.

3.1 Software

The software requirements for this reference design are:

- Linux, Windows 7, Windows 8.1 (ug973-vivado-release-notes-install-license.pdf)
- Xilinx Vivado Design Edition 2015.2, with SDK

3.2 Hardware

The hardware setup used by this reference design includes:

- Computer with 1 GB RAM and 1 GB virtual memory (recommended) (www.xilinx.com/design-tools/vivado/memory.htm)
- Avnet/Digilent Arty Evaluation board
- USB-A to USB-micro B cable
- Power supply (optional)

4 Supplied Files

The following directory structure is included with this reference design:

demo: Contains the files to run receive test program:

config_fpga.bat: Batch file to configure the FPGA with the "bootloop" bitstream of the MicroBlaze processor system.

cp_from_sdk.bat: Batch file to copy hardware bitstream and software executables from the SDK workspace.

demo_gpio_app.bat: Batch file to run the commands to load the FPGA bitstream of the hardware design and download the software application.

design_1_wrapper.bit: The golden FPGA bitstream of the hardware design.

design_1_wrapper.mmi: The golden BRAM map file to integrate the bootloop executable with the FPGA bitstream.

download.bit: The golden FPGA bitstream integrated with the bootloop application.

download_bit.tcl: TCL command file for downloading the bitstream to the board.

gpio_demo.elf: The golden MicroBlaze executable for the GPIO demo application.

load_bits.tcl: TCL command file for downloading the bitstream to the board.

sleep.exe: Utility to pause execution of the batch file.

xmd.ini: Command file used by XMD to download start execution of the software application.

doc: Contains the documentation for this reference design.

7A35T_Arty_OOB_GPIO_demo_VIV2015_2.pdf: This document.

IPI: Batch file and Vivado TCL scripts to create new Vivado project and create MicroBlaze system block design.

IPI_repo: Repository of files and IP needed to create the MicroBlaze hardware platform.

program_flash: Batch file and Vivado TCL scripts to program the QSPI Flash memory.

sdk_repo: Repository for the GPIO demo software application source files and BSP settings.

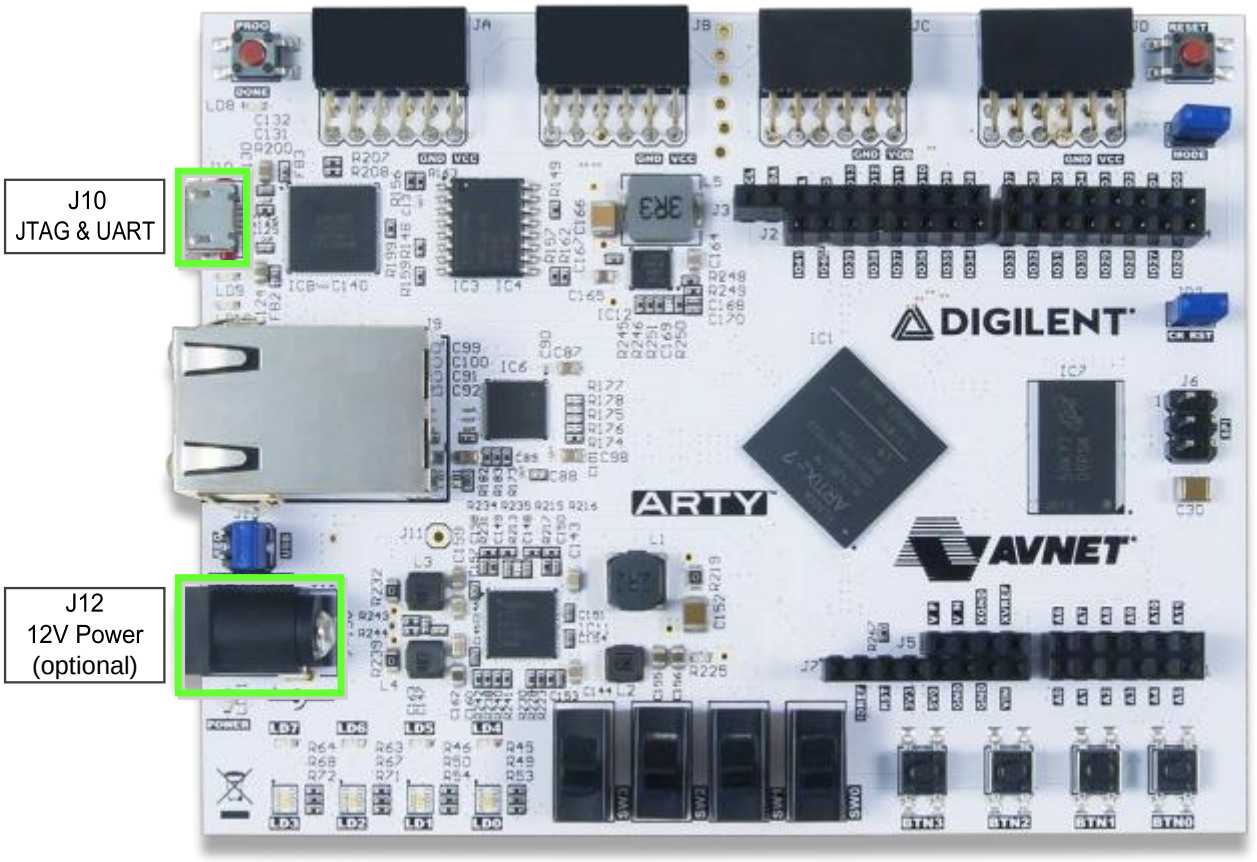
sdk_workspace: Empty folder to use to create new SDK workspace for this design.

IPI_solution.zip: Contains pre-built Vivado IPI project for this design.

sdk_solution.zip: Contains pre-built SDK project for this design.

5 Setting Up the Arty Evaluation Board

Refer to the following figure and perform the following steps to set up the board for running the example design.



1. Plug a USB cable into the PC and the combination JTAG & UART port (J10) (this will also power the board). If the USB port on the hub or host PC cannot supply enough power you may alternatively power the board via the 12V power barrel jack (J12).

6 PC Setup

6.1 Installing the UART Driver and Virtual COM Port

The USB UART driver is built into the device driver for the JTAG interface and is included with the Xilinx Vivado tools installation. As long as the Vivado tools are installed, the USB UART will be recognized when the board is plugged into the host PC.

See Appendix I: Determining the Virtual COM Port for information on identifying the COM port in use on the host PC.

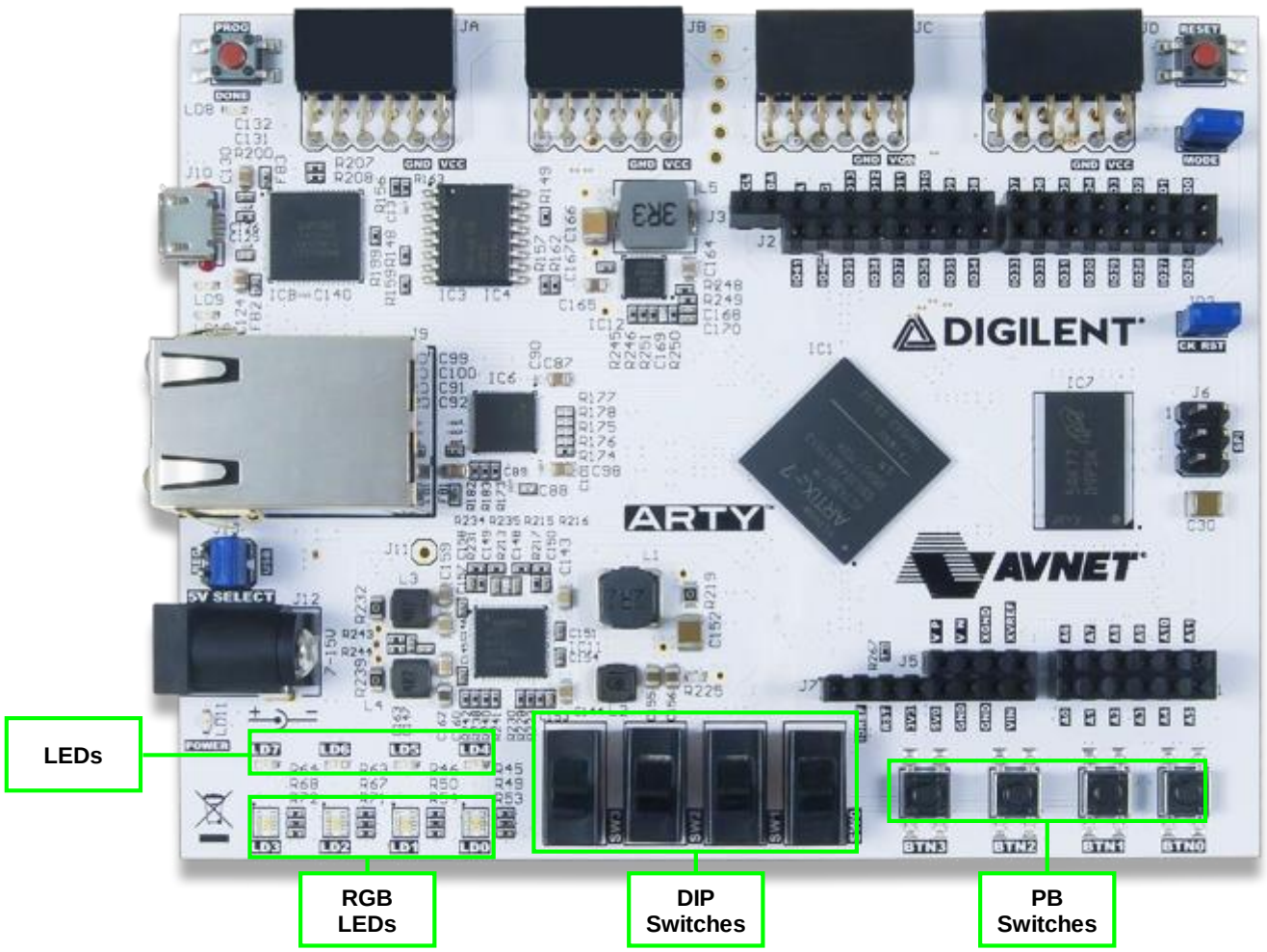
6.2 Installing a Serial Console on a Windows 7 Host

Starting with Windows 7, Microsoft no longer includes the HyperTerminal terminal emulator software. However, this example design requires use of terminal emulation software for a serial console connection to the Arty board. A suitable free and open-source replacement for HyperTerminal is TeraTerm. Download and install instructions for TeraTerm can be found at http://en.sourceforge.jp/projects/ttssh2. As an alternative the Terminal applet in the Xilinx SDK may also be used.

7 Running the Demo Files

You can load the FPGA and run the software application without building the design by using the demo scripts and the pre-built bitstream and elf files. You must have the Xilinx tools (including the SDK) installed on your host, and have the hardware set up and connected as described in Setting Up the Arty Evaluation Board.

Refer to the figure below when running the GPIO demo software application. Note the location of the pushbutton switches, DIP switches, and LEDs.

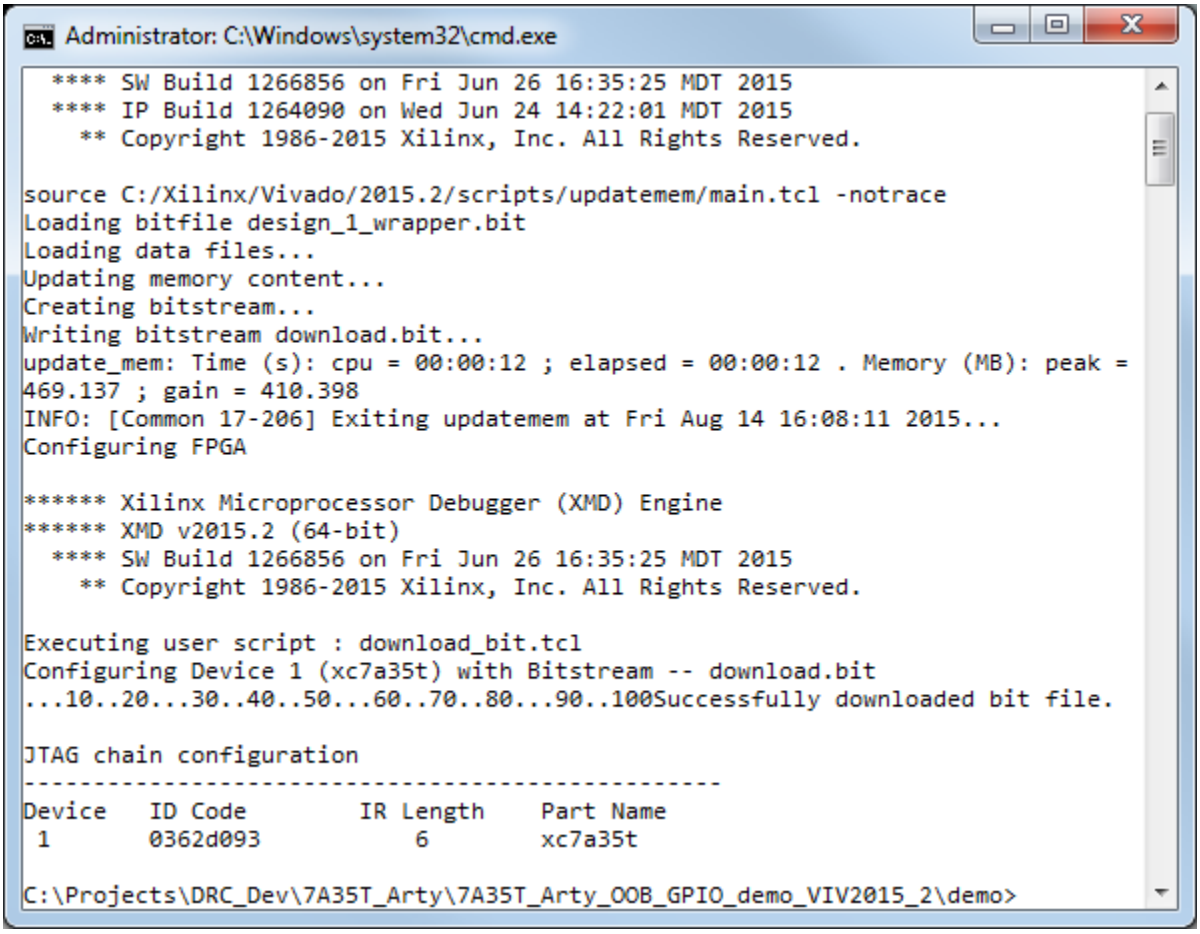


7.1 Applications Download

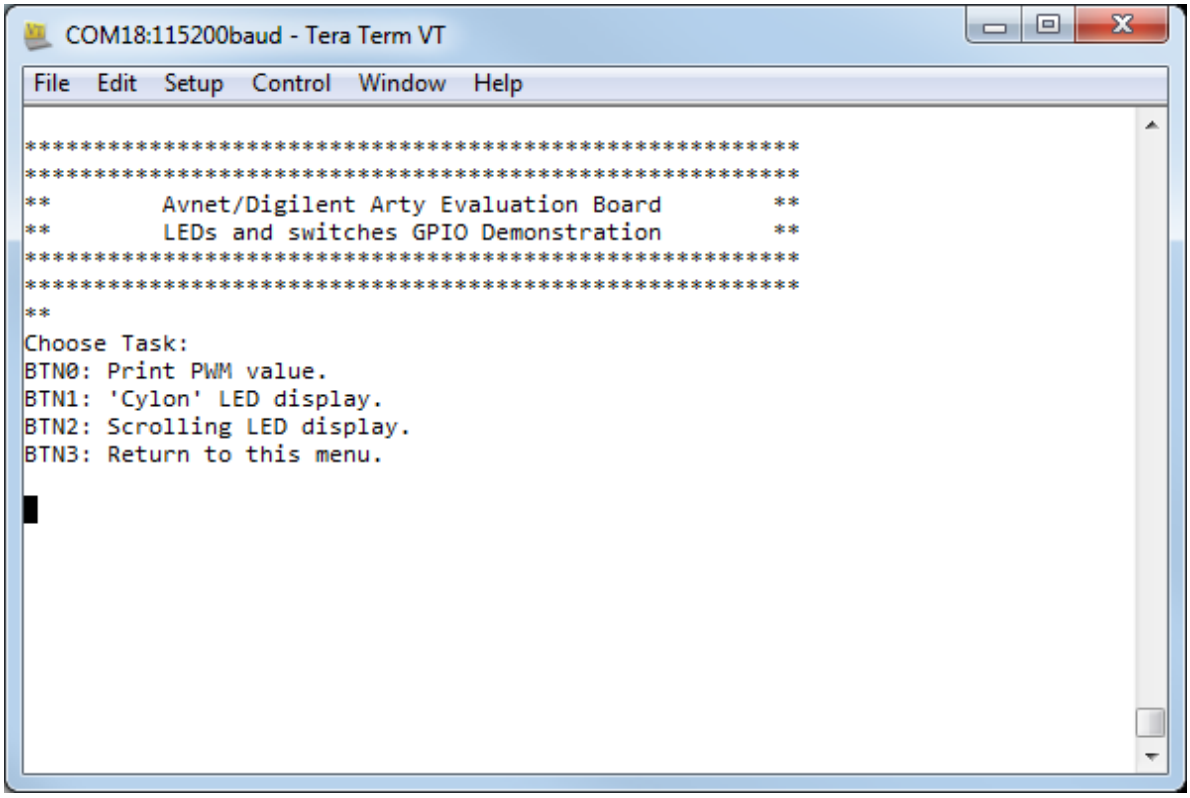
1. Start a serial terminal session and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.
2. Open a command window in the **<installation>\demo** folder and enter:

demo_gpio_app.bat

3. The FPGA bitstream will be downloaded, followed by the executable file for the software application. Do not close the command window.



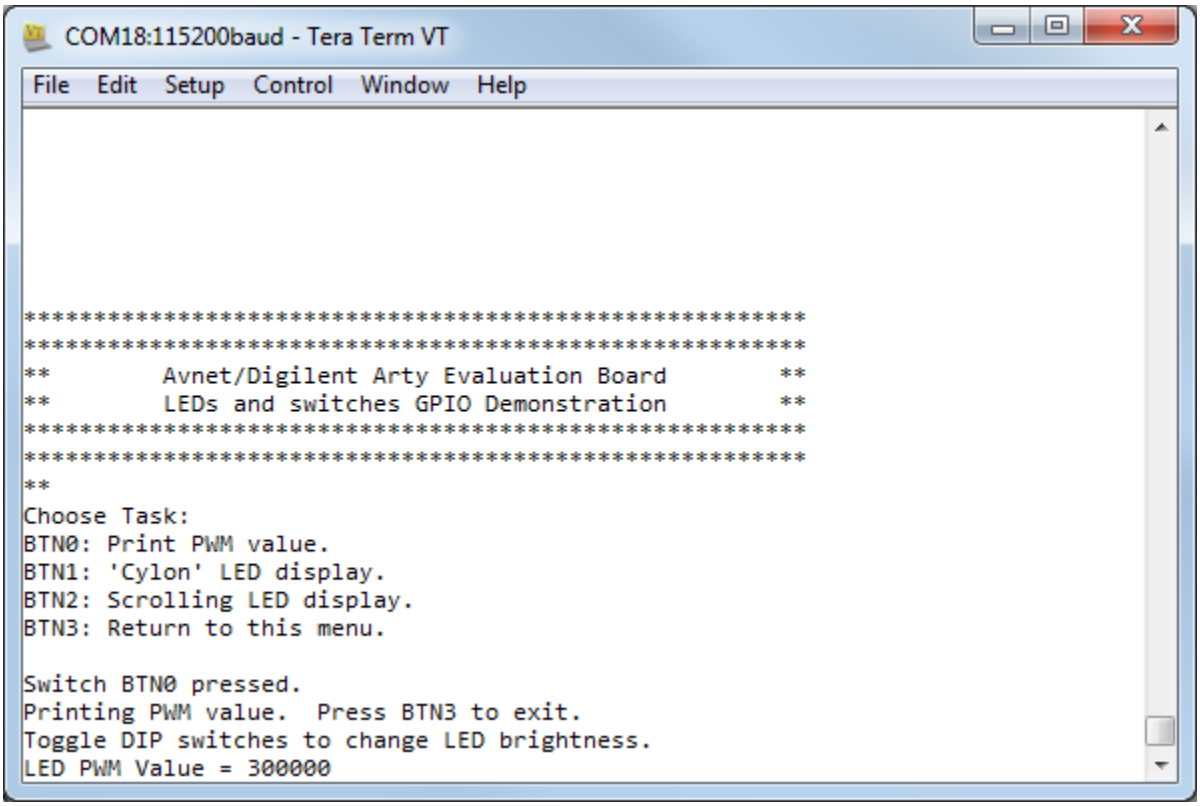
4. When the executable has finished loading and is ready to run you should see the following in your serial terminal window:



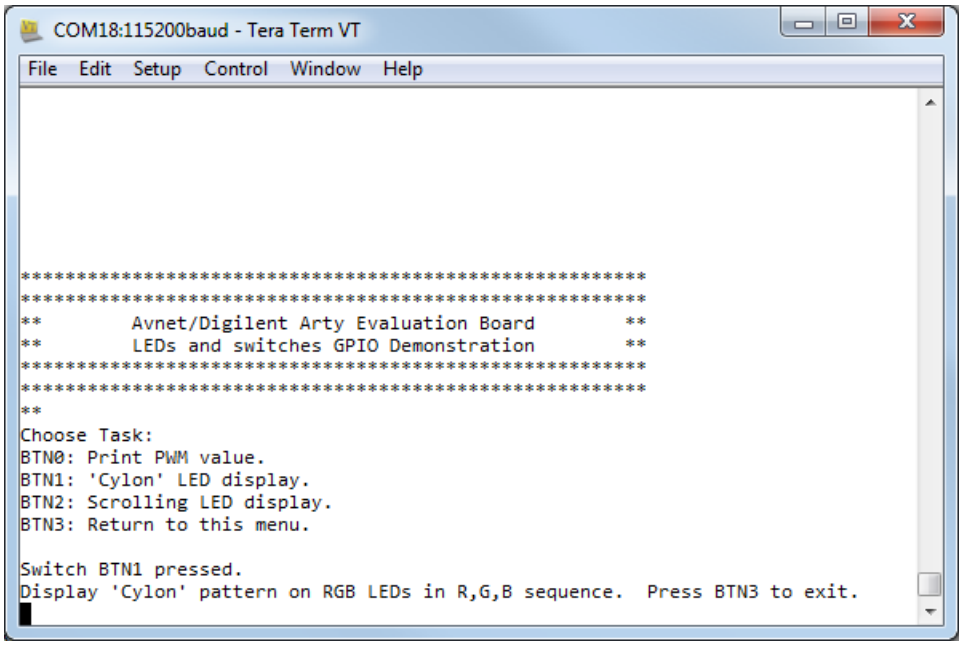
7.2 GPIO Demo

The GPIO demo software application allows the user to interact with the pushbutton and DIP switches on the board to change the display on the LEDs.

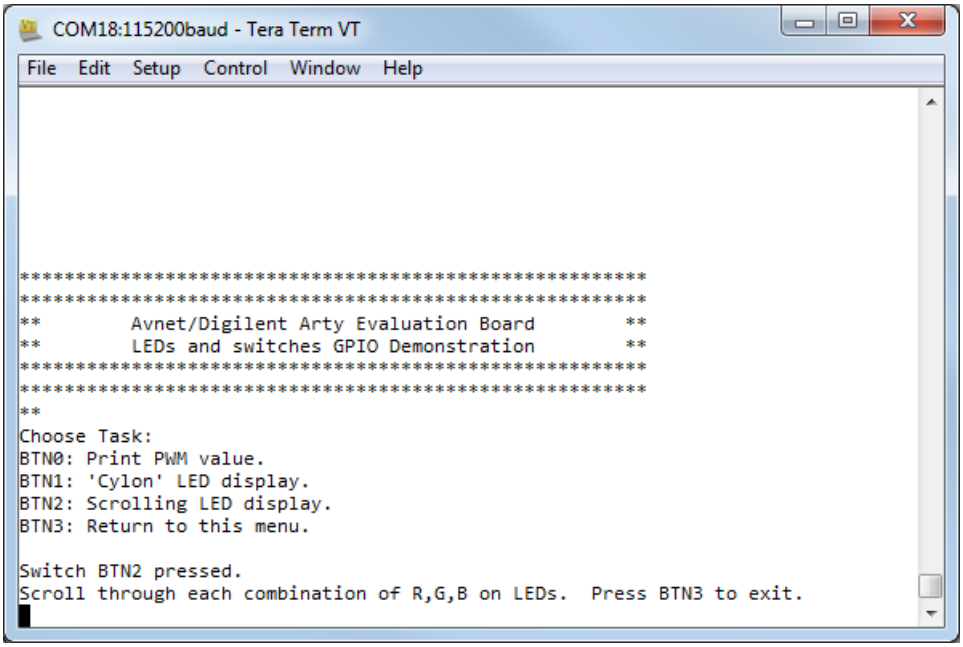
1. Press the **BTN0** pushbutton switch to select the option to print the PWM value that controls the brightness of the LEDs. The PWM value is a calculation of the value read from the DIP switches (SW0-SW3). The DIP switches represent a hexadecimal value. The computed result is used as input to the PWM that controls the brightness of other LEDs on the board (LD4-LD7). The software prints the new PWM value each time the value read from the DIP switches changes. Notice that the LEDs on the board get dimmer and brighter as the DIP switches are toggled. The DIP switch value is multiplied by 20,000 and written to the PWM peripheral to modulate the brightness of the LEDs. The greater the computed value the brighter the LEDs will be. Do not close the command window. Press the **BTN3** switch to return to the main menu.



2. Press **BTN1** to select the pattern to display a ‘Cylon’ (walking back and forth) pattern on the RGB LEDs (LD0-LD3). The color will also change as the LEDs cycle back and forth. Do not close the command window. Press the **BTN3** switch to stop the display and return to the main menu.



3. Press **BTN2** to select the scrolling color LED display on the RGB LEDs (LD0-LD3). The RGB LEDs will scroll through the combination of red, green, and blue for each LED. Press the **BTN3** switch to return to the main menu.



8 Create the MicroBlaze System

8.1 Hardware Design Block Diagram

The following figure shows a high-level block diagram of the hardware design. The design requires:

- MicroBlaze processor
- 32KB of BRAM
- UART Port
- GPIO for LEDs and switches
- QSPI Flash
- Interrupt Controller
- Custom PWM peripheral

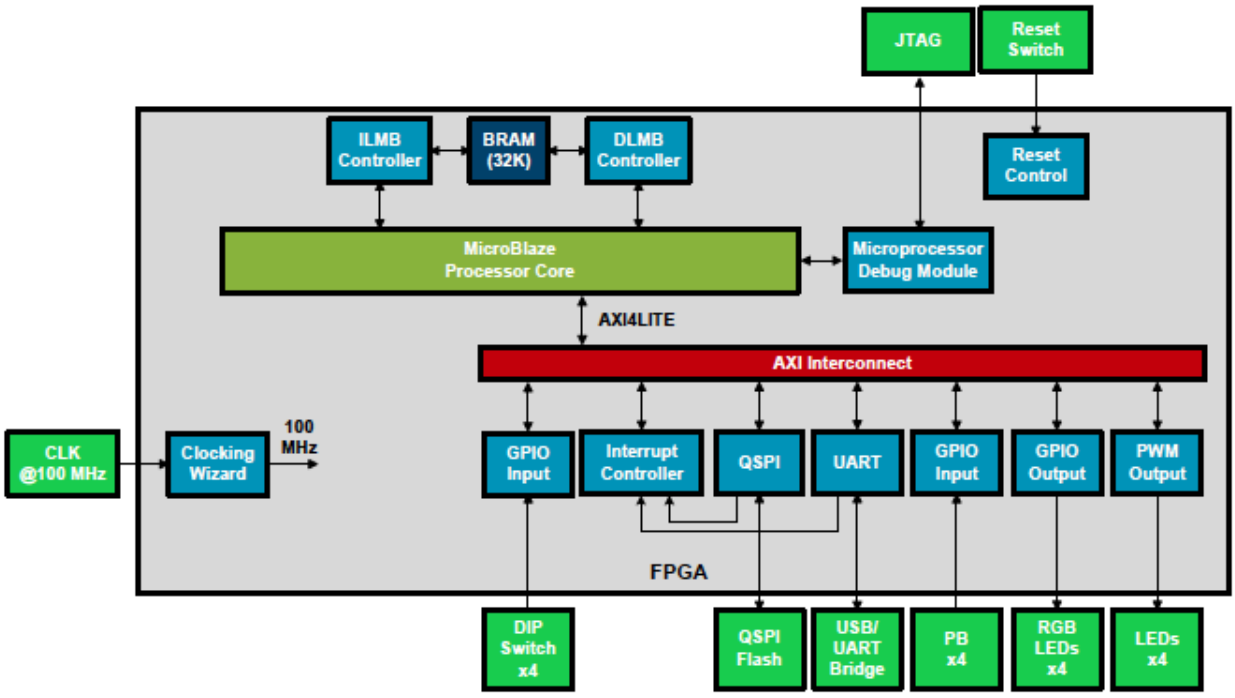


Figure 1 – Reference Design Block Diagram

8.2 Description of Hardware Modifications

This design was created using the Vivado IP Integrator Block Design flow with the following modifications:

- MicroBlaze processor settings:
  - Select the Typical Predefined Configuration
  - Specify 32KB for local memory size
- AXI_QSPI settings
  - Enable Quad mode
  - Set the slave device to Micron (Numonyx)
  - Disable the STARTUP2E primitive
- Clocking Wizard settings
  - Set clk_out1 to 100MHz (system clock)
  - Disable the reset input
- AXI_UARTLite
  - Set baud rate to 115200

8.3 Vivado IP Integrator

Vivado IP Integrator (IPI) provides a rich graphical environment in which to create and customize MicroBlaze processor systems. The integrated TCL command window allows for running simple commands. In fact, most functions and tasks in the Vivado GUI are run as TCL commands. The TCL command window can also be used to automate complex tasks like creating a MicroBlaze system from scratch that is capable of running the software application. Follow the steps below to import and implement a pre-built known-good MicroBlaze system block design.

8.3.1 Install Board Definition Files

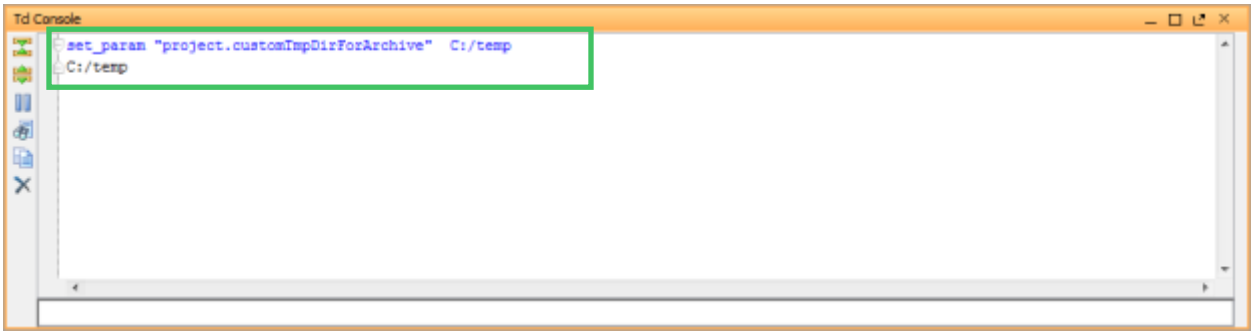
1. If you haven't already done so, please follow the instructions in Appendix II to install the board definition XML file set for the Arty evaluation board. The board definition file identifies the interfaces on the Arty board and allows us to target the board directly in the Vivado tools by selecting it from a menu.

8.3.2 Create the Vivado IPI Block Design Project

1. Navigate to the Windows Desktop or Start menu and launch Vivado.

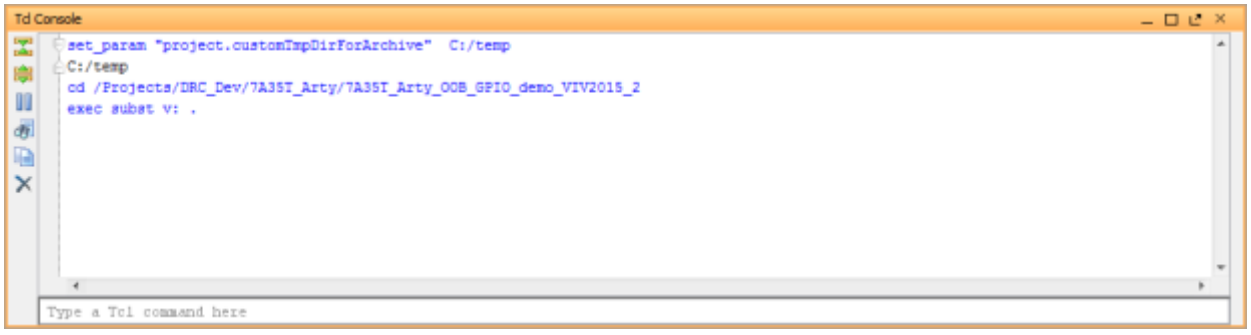
2. If you are using the Vivado Design Suite on a Windows®-7 host, there is a known issue that Vivado file path names often exceed the maximum allowed by Windows. Please see Appendix III for more details. A workaround for this is to change the temp folder that Vivado uses. Create your own temporary directory named C:\temp, and force Vivado to use the new folder with the following TCL command:

   ```
   set_param "project.customTmpDirForArchive"      C:/temp
   ```



2. Another workaround for the 260 character path length limitation is to use the Windows subst command to substitute a drive letter for a long file path. Substitute the drive letter V: (or any other unused drive letter on your Windows PC) for the path where the zip archive of this example design was extracted. This can be done in the Vivado TCL Console:

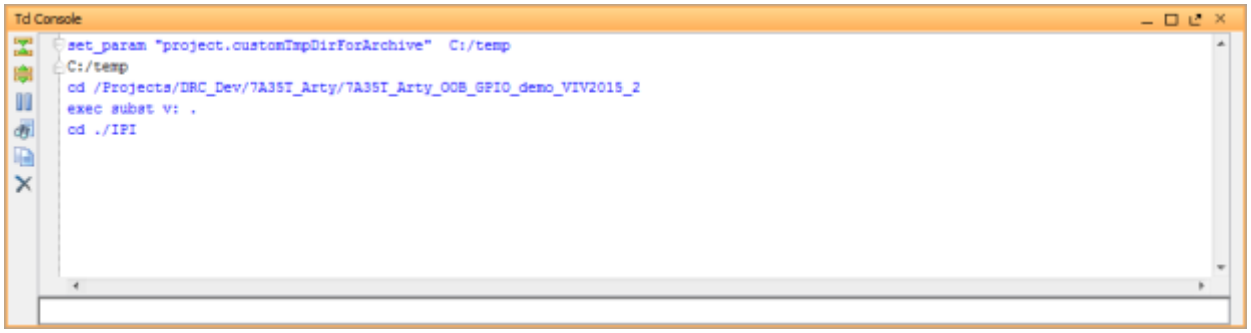
   ```
   cd <installation>
   exec subst V: .
   ```



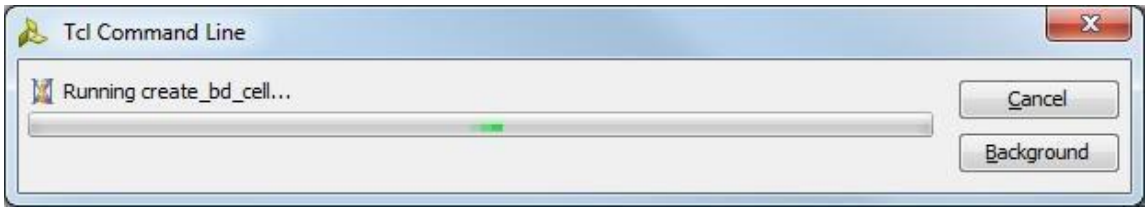
8.3.3 Import and Recreate the Block Design

1. At this point we can create the Vivado project and import the pre-built MicroBlaze processor system block design. Open the **TCL Console** by selecting the tab at the bottom of the Vivado window and enter the following command to import the block design:

```
cd ./IPI
source ./design_1.tcl
```



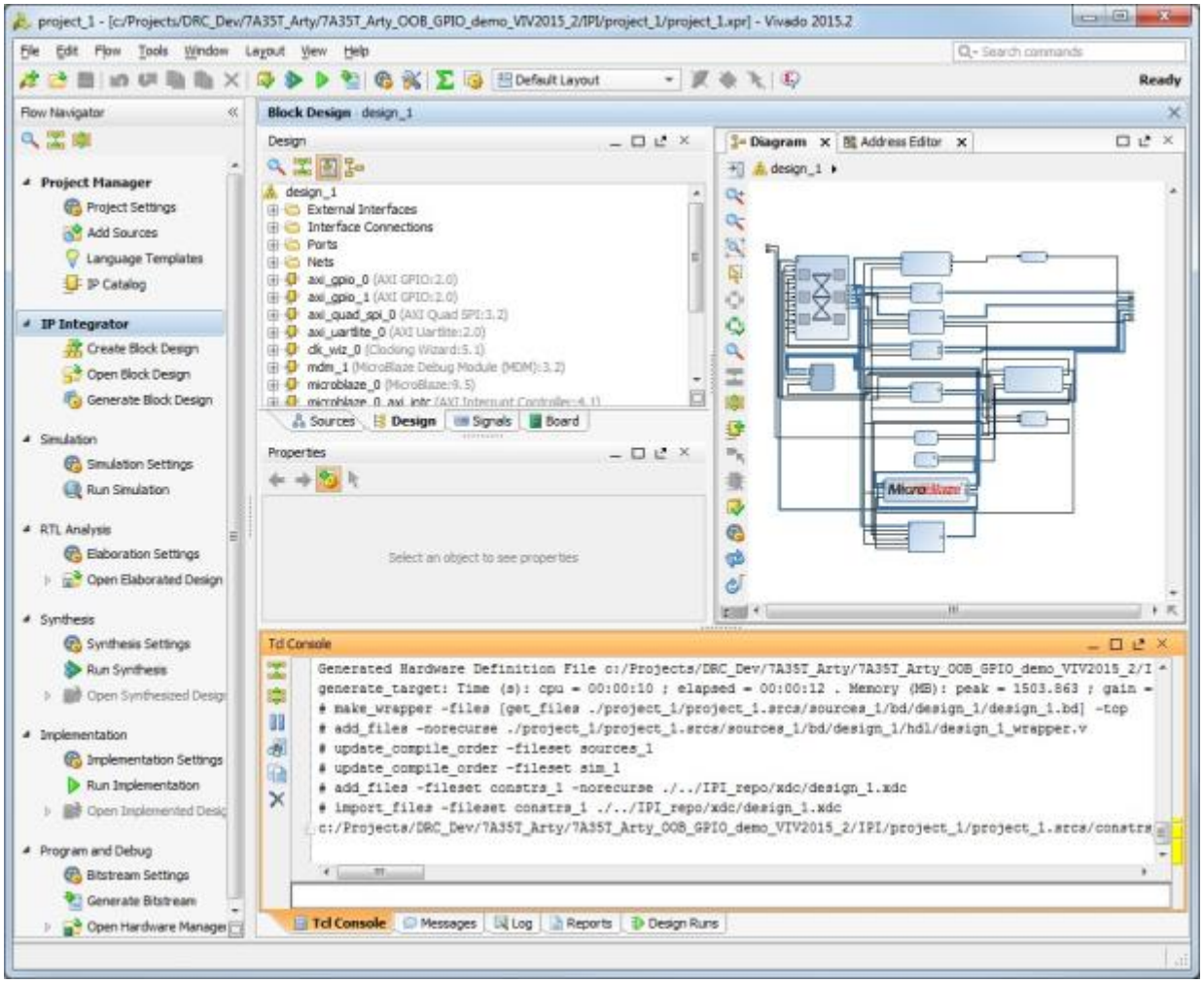
2. You will see the following window as the block design is imported and recreated. This process may take a few minutes.



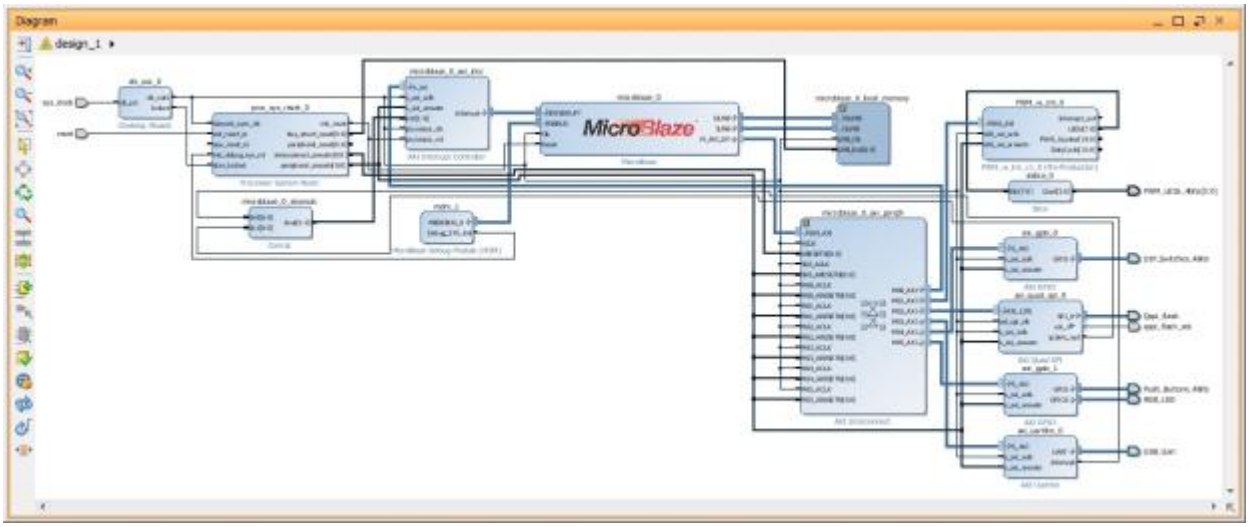
3. Open the **design_1.tcl** file in your favorite text editor and identify the important Vivado tasks that the script automates for us:
   - Create the new project in the **project_1** folder and select the **XC7A35T** FPGA device
   - Select the **Arty** as the target board
   - Set the custom IP repository folder location to **../IPI_repo/ip_repo**
   - Build the MicroBlaze processor system block design
   - Generate the top-level HDL wrapper for the block design
   - Add the **design_1.xdc** constraints file to the design

```
# Create the new project in the project_1 folder and select the XC7A35T FPGA device
create_project project_1 -force ./project_1 -part xc7a35ticsg324-1L

# Select the Artix-7 XC7A35T 'Arty' as the target board
set_property board_part digilentinc.com:arty:part0:1.1 [current_project]

# Set the custom IP repository folder location to ../IPI_repo/ip_repo
set_param bd.enableIpSharedDirectory true
set repos_local "./../IPI_repo/ip_repo"
set_property ip_repo_paths  "${repos_local}" [current_fileset]
update_ip_catalog -rebuild

# Build the MicroBlaze processor system block design
source design_1_bd.tcl
generate_target  {synthesis simulation implementation}  [get_files  ./project_1/project_1.srcs/sources_1/bd/design_1/design_1.bd]

# Generate the top-level HDL wrapper for the block design
make_wrapper -files [get_files ./project_1/project_1.srcs/sources_1/bd/design_1/design_1.bd] -top
add_files -norecurse ./project_1/project_1.srcs/sources_1/bd/design_1/hdl/design_1_wrapper.v
update_compile_order -fileset sources_1
update_compile_order -fileset sim_1

# Add the design_1.xdc constraints file to the design
add_files -fileset constrs_1 -norecurse ./../IPI_repo/xdc/design_1.xdc
import_files -fileset constrs_1 ./../IPI_repo/xdc/design_1.xdc
```

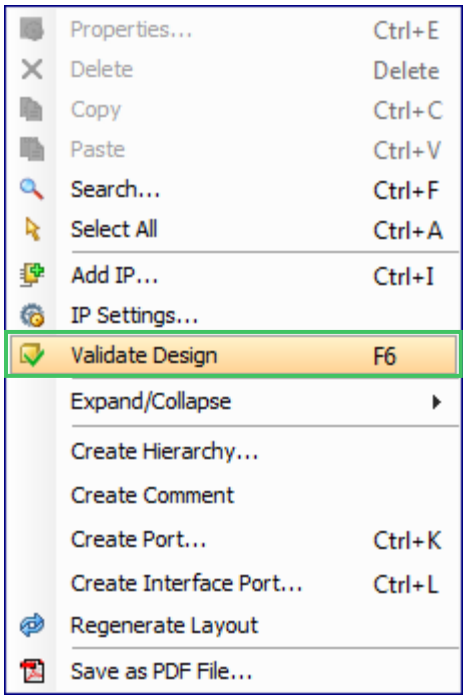
4. When the design is done building you should see a Vivado window similar to this:



5. The Block Design should look similar to this:

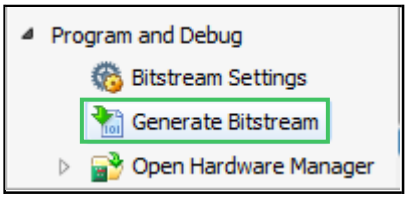


6. Right click in the block design **Diagram** window and select **Validate Design**. There should not be any errors or warnings. Click **OK** to continue.

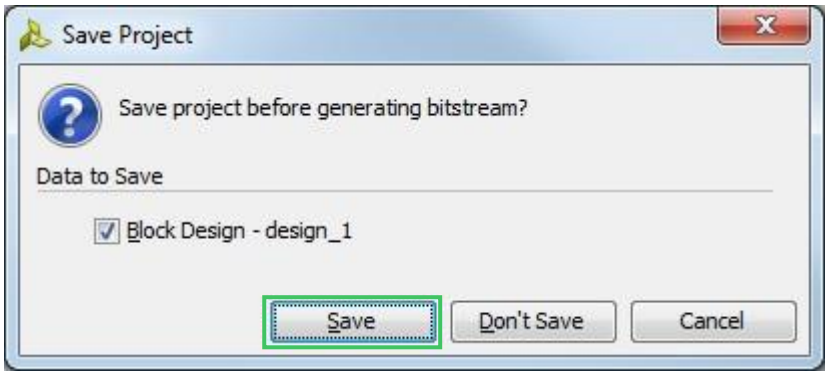


8.3.4 Implement the Design

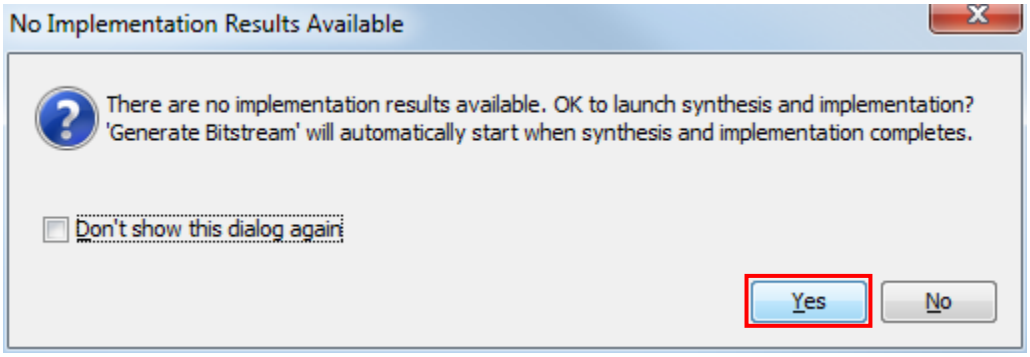
1. We are now ready to build the design. Navigate to the **Project Manager** pane and click **Generate Bitstream**. This will run all previous steps to generate the IP netlists, synthesize the design, and run implementation. You will see a window reminding you that synthesis hasn't been completed yet. Click **OK** to continue.



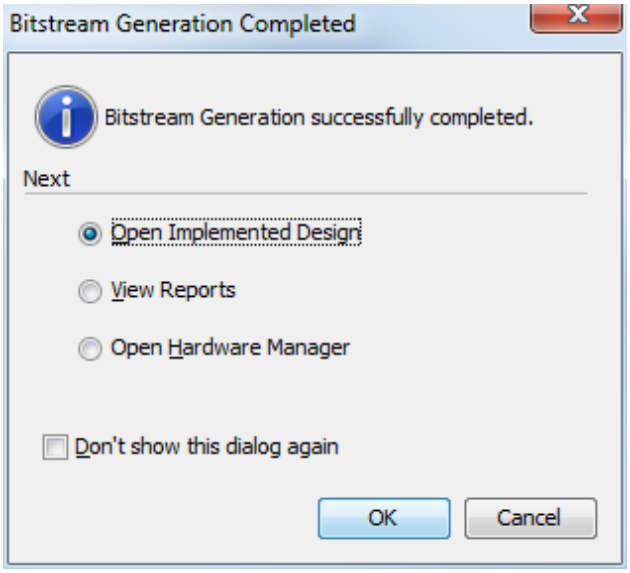
2. Click **Save** if you are prompted to save the block design.



3. You will see a window reminding you that synthesis and implementation hasn't been completed yet. Click **Yes** to continue. This will take a few minutes.



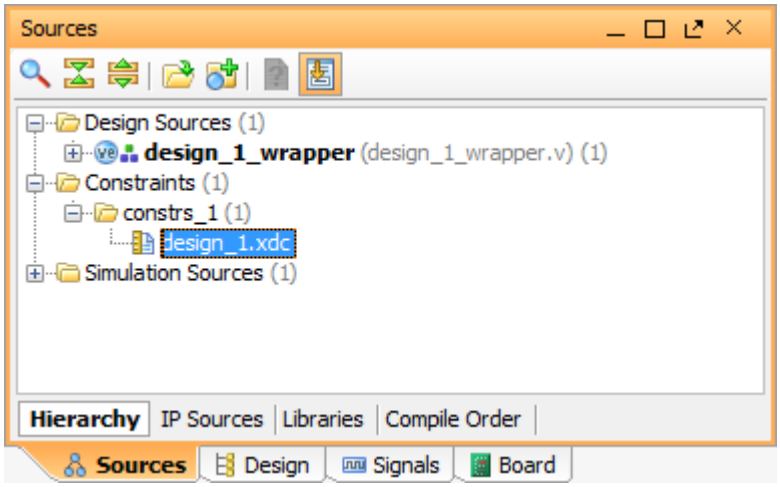
4. When prompted select **Open Implemented Design** and click **OK**:



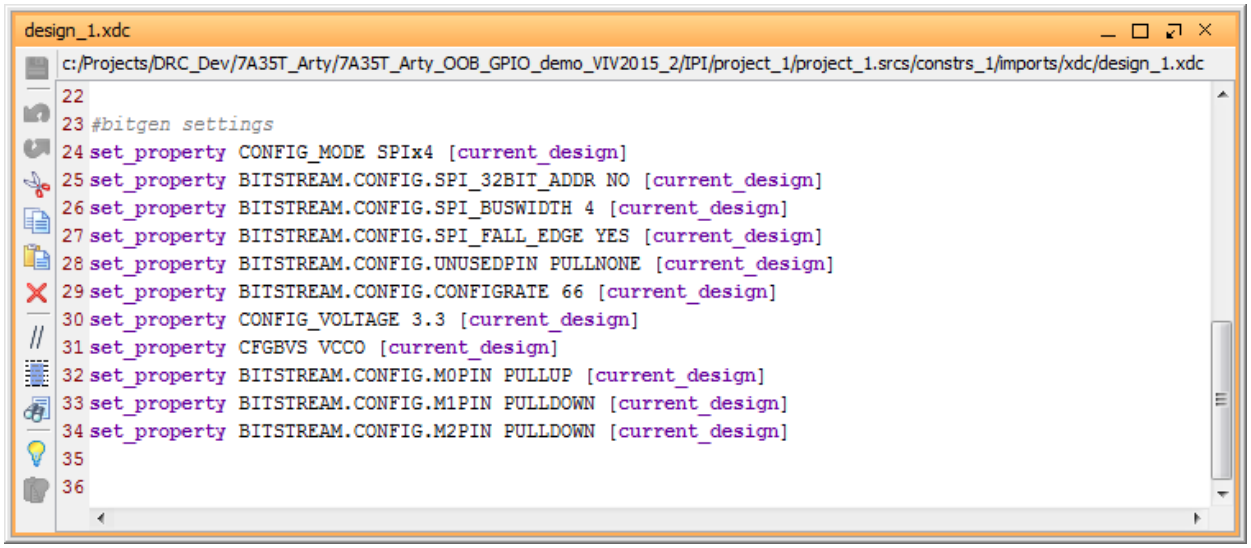
8.3.5 Review the Bitstream Settings

In addition to specifying the system I/O and timing constraints, the system constraints file, design_1.xdc, also specifies the parameters for creating the FPGA configuration bitstream. Because the Arty Evaluation board is configured via a Quad SPI flash memory we must be sure to create a properly formatted bitstream for this interface. Also, the default behavior of the bitstream generation tool is to create an internal pulldown on all unused pins. This can sometimes cause unintended consequences for board interfaces like the DDR3 memory. Follow the steps below to review the bitstream generation options.

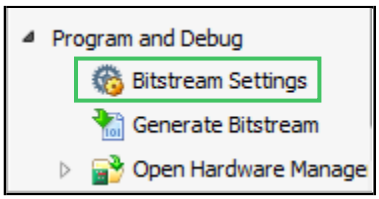
1. Navigate to the **Sources** tab and expand the **Constraints** branch until the **design_1.xdc** constraints file is visible. Double-click to open the file.



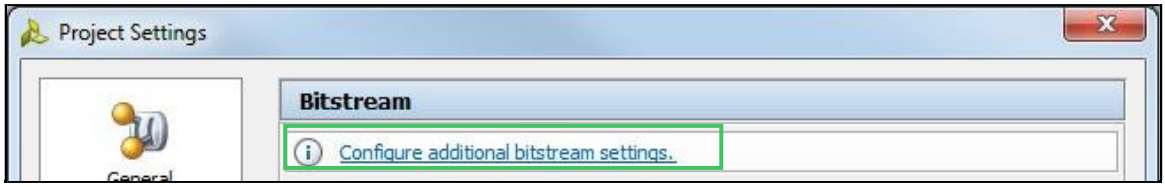
2. Scroll down to near the end of the file and locate the **#bitgen settings** section. These are the constraints and parameters that instruct Vivado to create a properly formatted bitstream for the Arty Evaluation board.



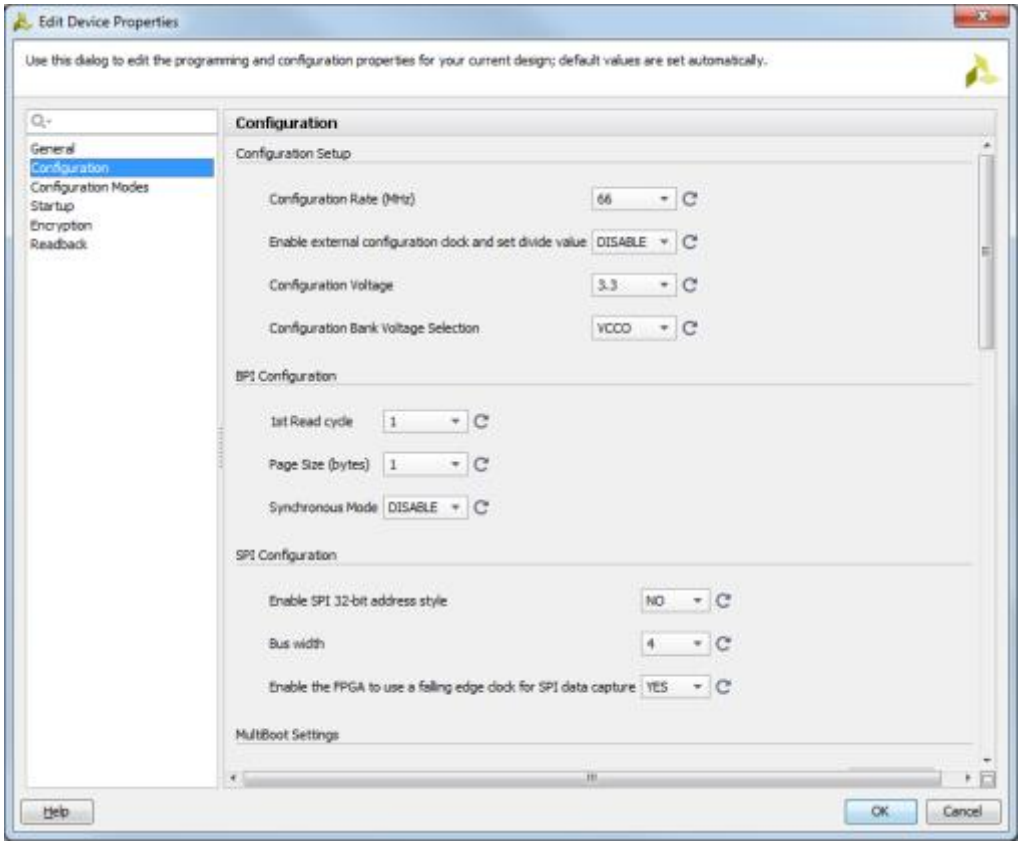
3. These constraints are actually created by selecting options in the Bitstream Settings GUI. With the implemented design open in memory, go to the **Flow Navigator** pane and scroll to the bottom to find and select **Bitstream Settings** under **Program and Debug.**



4. In the Project Settings window click on **Configure additional bitstream settings**.

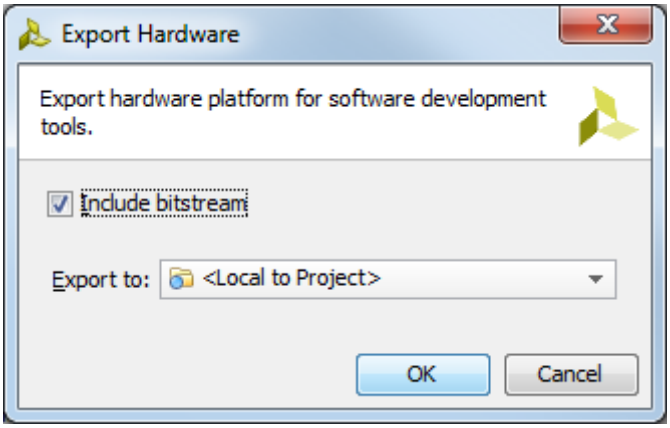


5. In the **Edit Device Properties** window select the **Configuration** tab and compare the settings set in the GUI to those specified in the system constraints file. Click **OK** when done:

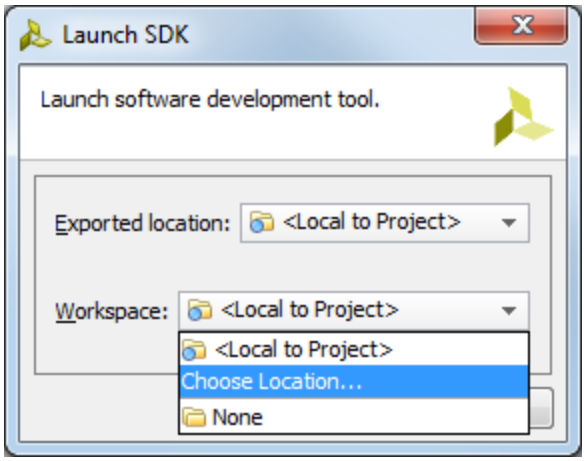


8.3.6 Export Vivado Hardware Design to the SDK

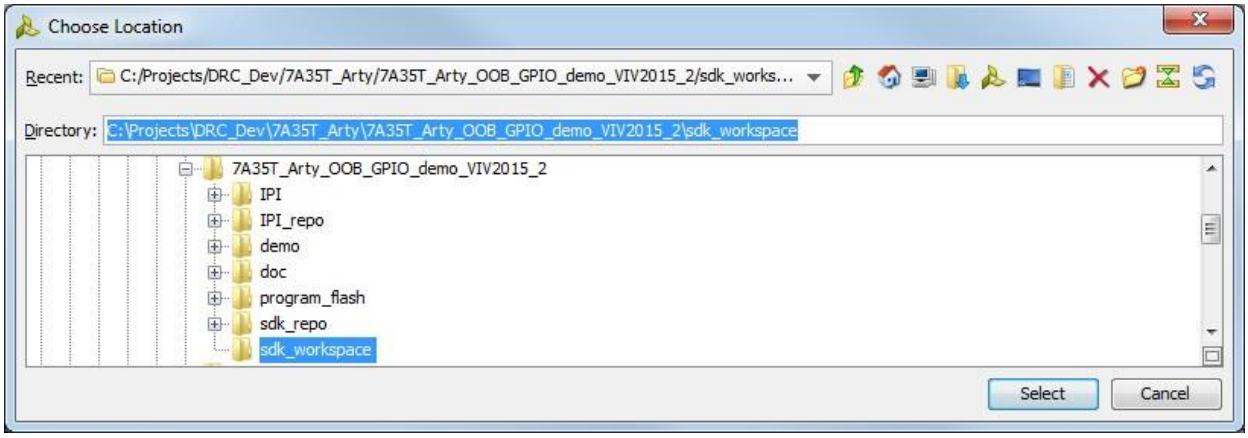
1. All software aspects of the design are performed inside a SDK Workspace. To generate an empty workspace based on the hardware platform built in Vivado navigate to **File → Export → Export Hardware** from the Vivado GUI. Accept the default **<Local to Project>** export directory location and check the box to **Include bitstream** then click **OK** to continue.



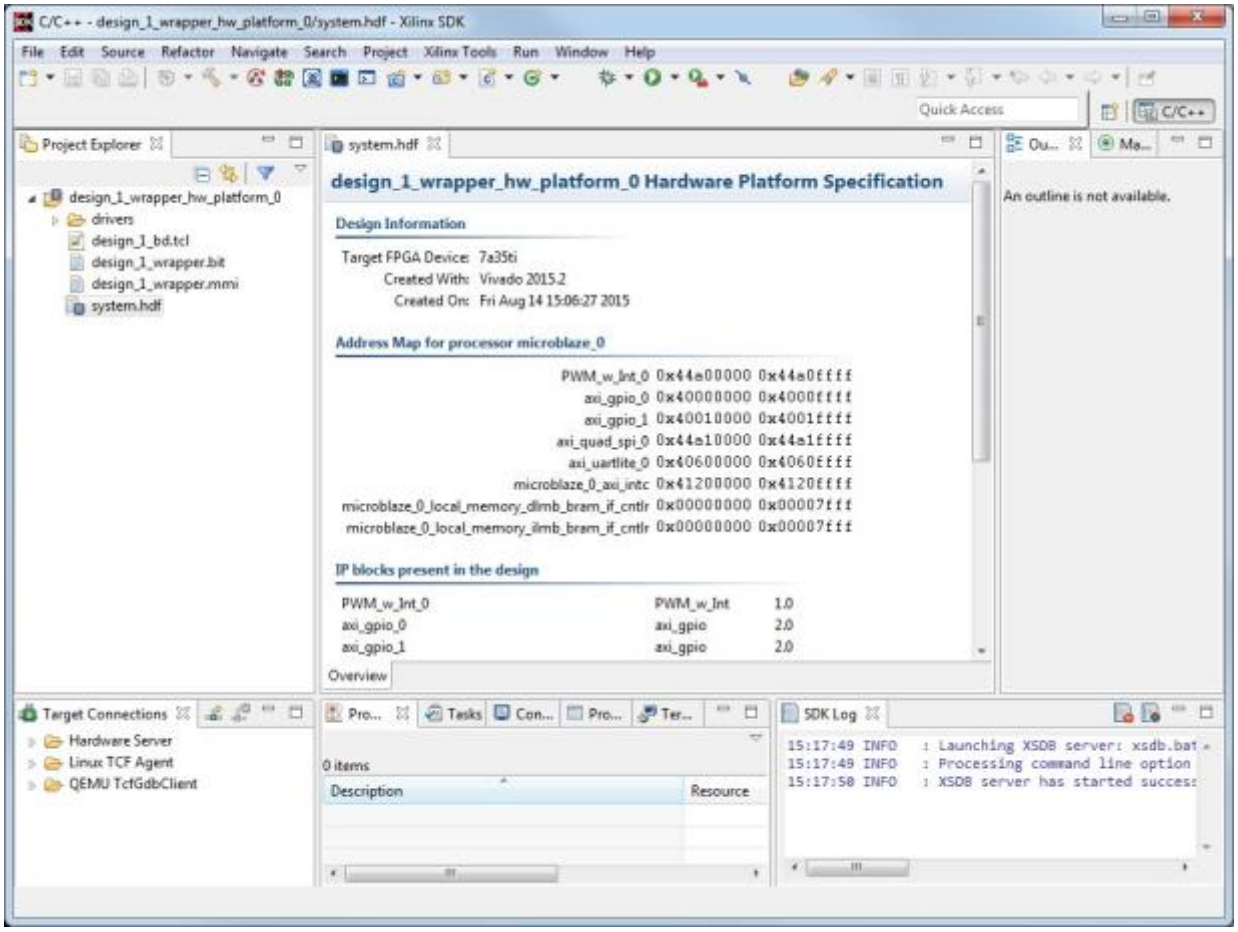
2. Navigate to **File → Launch SDK** from the Vivado GUI. In the Launch SDK window click on the Workspace drop list and select **Choose Location**.



3. Navigate to the **<installation>\sdk_workspace** folder to create a new SDK workspace. **Make sure there are NO SPACES in this path.** The Xilinx SDK does not tolerate spaces in this file path. Click **Select** and then click **OK** to continue.



4. After a few seconds the SDK will show a GUI similar to the one shown below:



9 Create the Software Applications

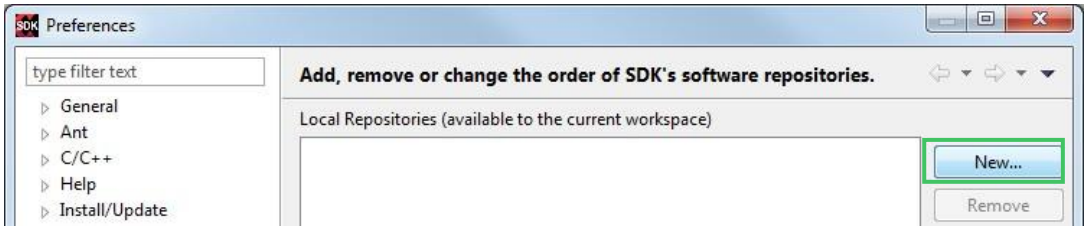
The next steps are to create the GPIO demo and SPI SREC bootloader software applications.

9.1 Add Custom Repository

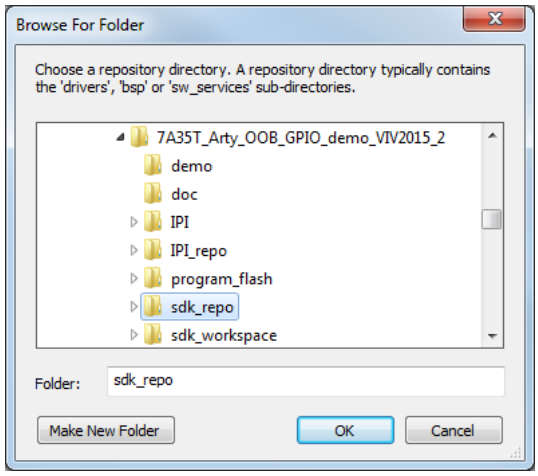
Adding a custom repository to the SDK workspace is a convenient way to integrate software libraries and applications into your software development. We need to add a repository to the SDK workspace for the GPIO demo software application. Including the GPIO demo application in a repository greatly simplifies the task of adding new software sources or application projects to the SDK workspace.

1. In the SDK GUI navigate to **Xilinx Tools → Repositories.**

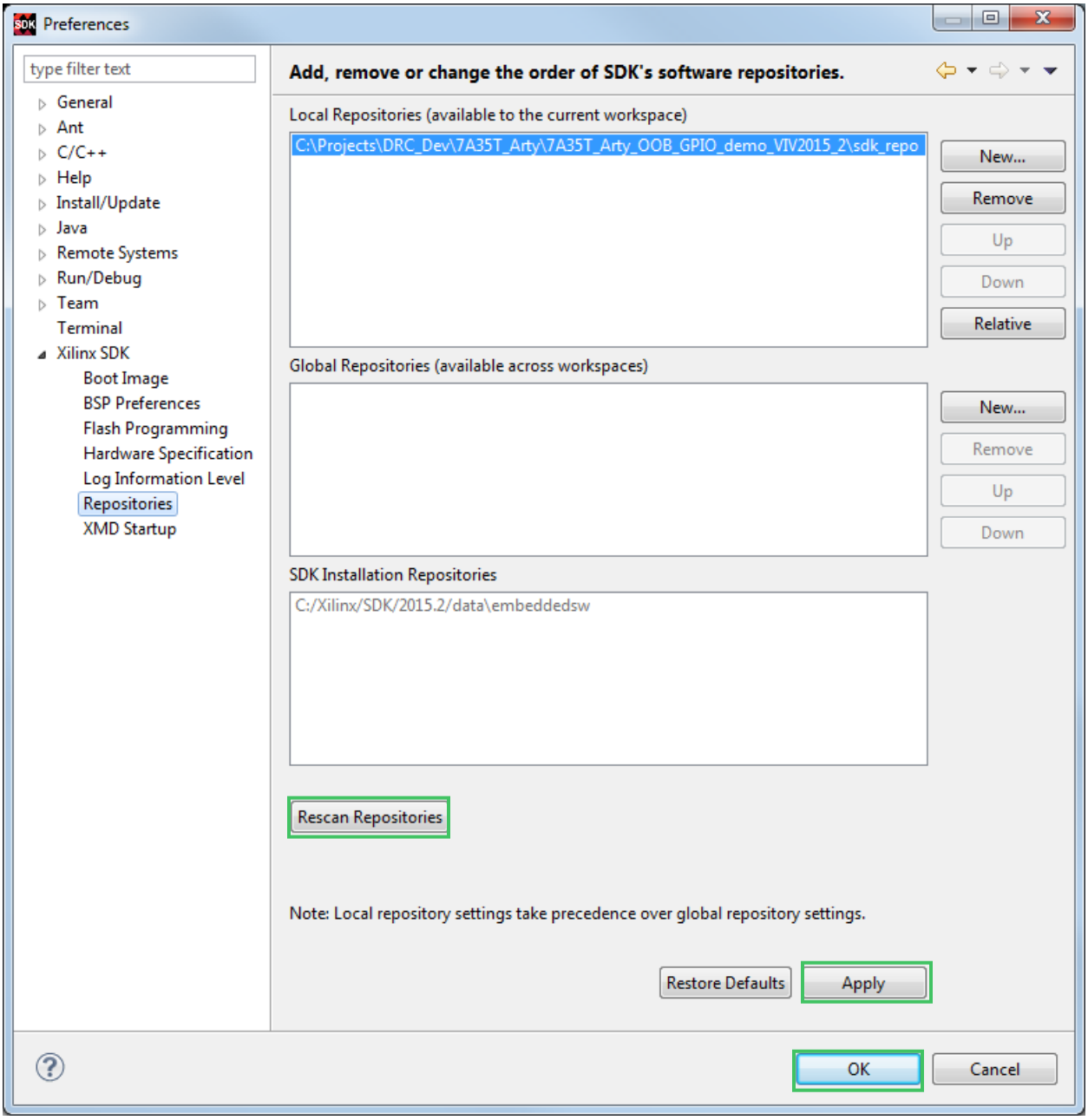
2. Under **Local Repositories** click on **New** to create a repository that will only be visible and usable by this particular SDK workspace.



3. Navigate **to <installation>\sdk_repo** and click **OK**.

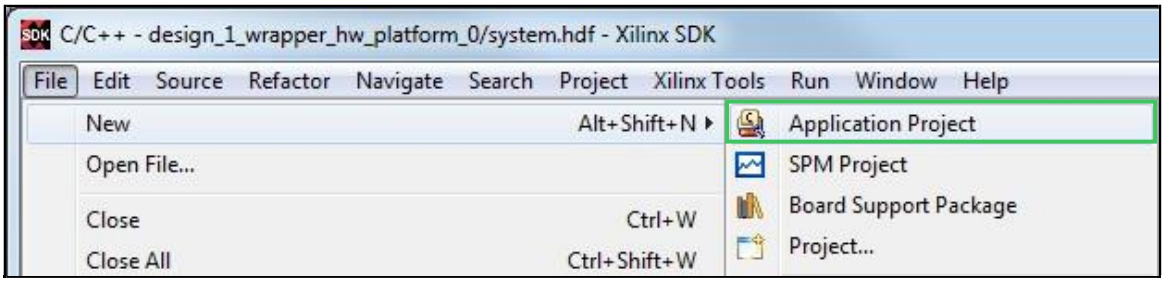


4. Click on **Rescan Repositories**, then click **Apply**, then click **OK**. The new repository is now ready to use.

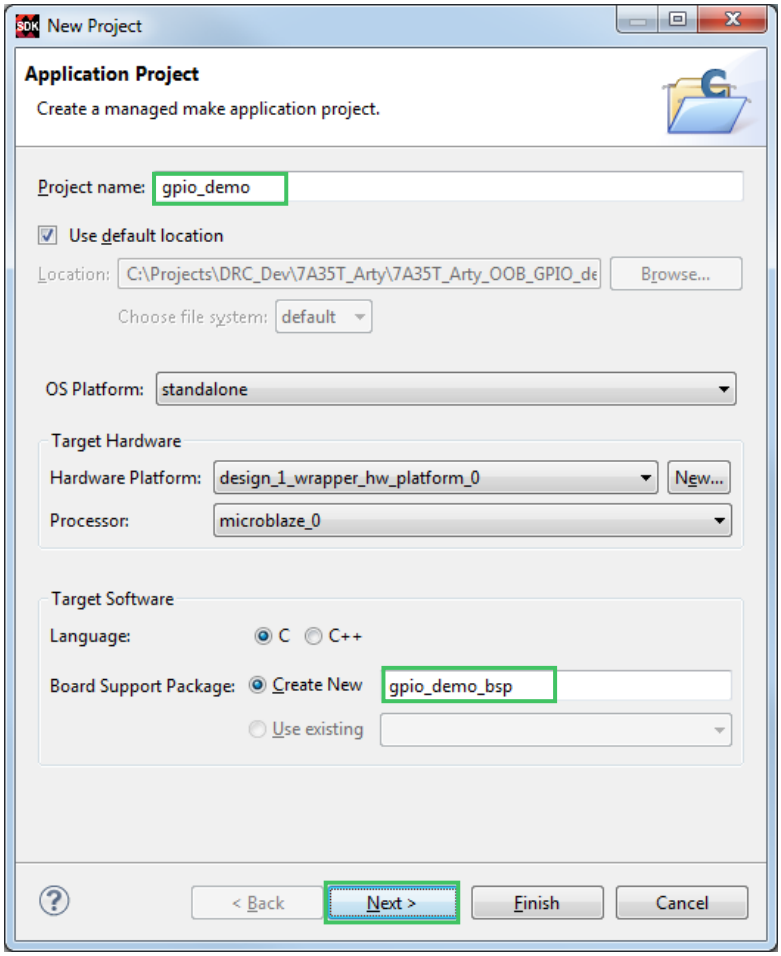


9.2 Create the GPIO Demo software application

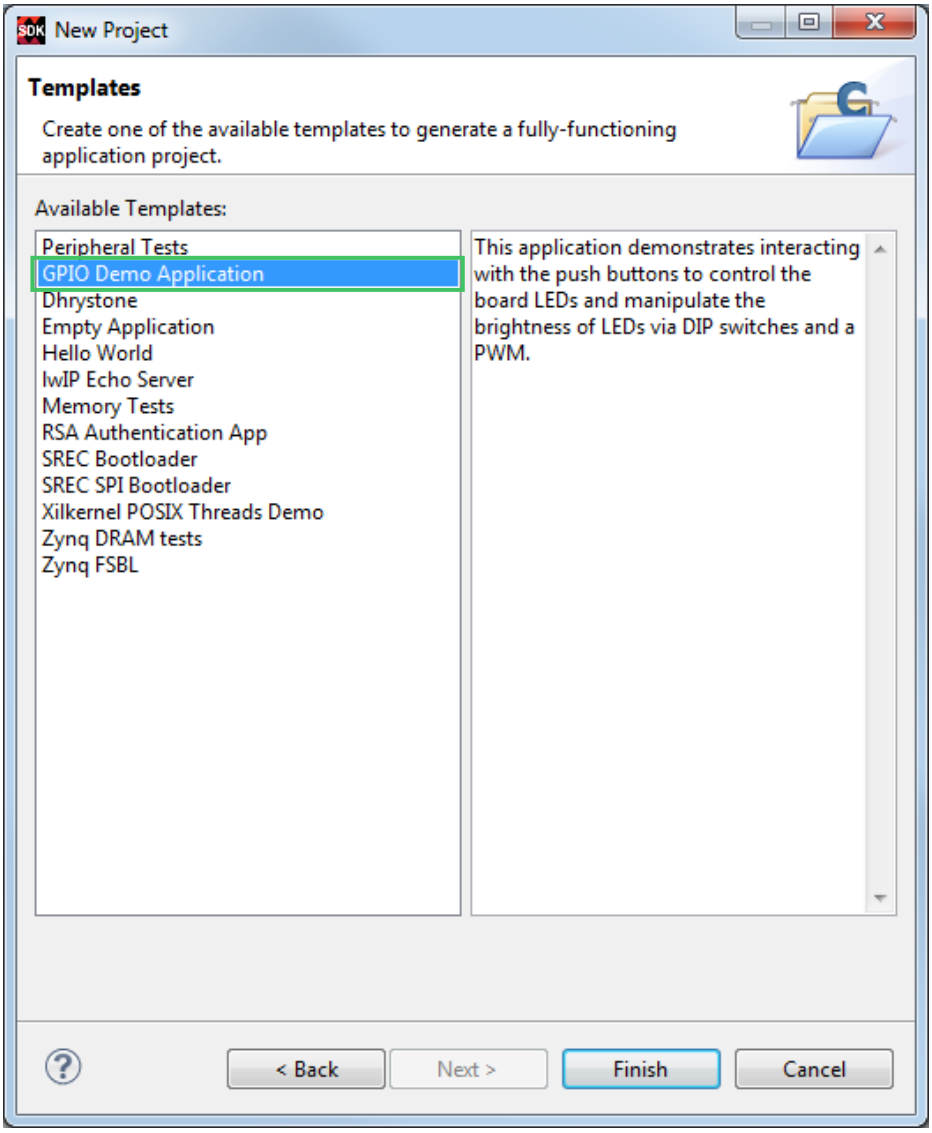
1. Go to **File → New → Application Project**



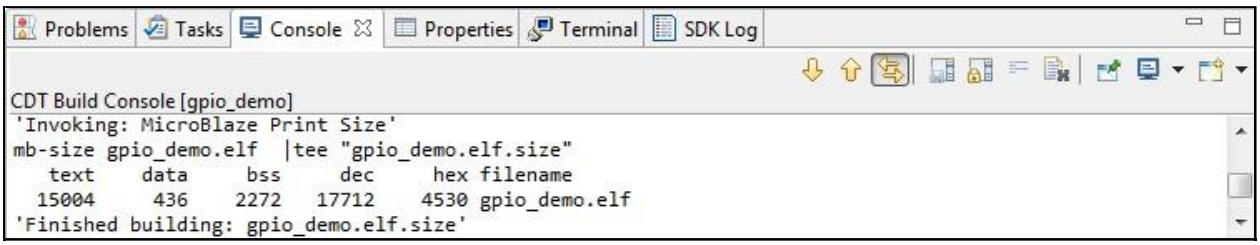
2. Name the project **gpio_demo** and accept the **default location** and **Target Hardware** settings. Accept the default **OS Platform** and **Language**. Accept the default to let the SDK Create New board support package named **gpio_demo _bsp**. Click **Next** to continue.



3. Select the **GPIO Demo Application** template and click **Finish** to continue. This application appears in this list because it is in the repository we added earlier. The source code for the applications and BSP, along with the BSP settings, will be added to the workspace and compiled.

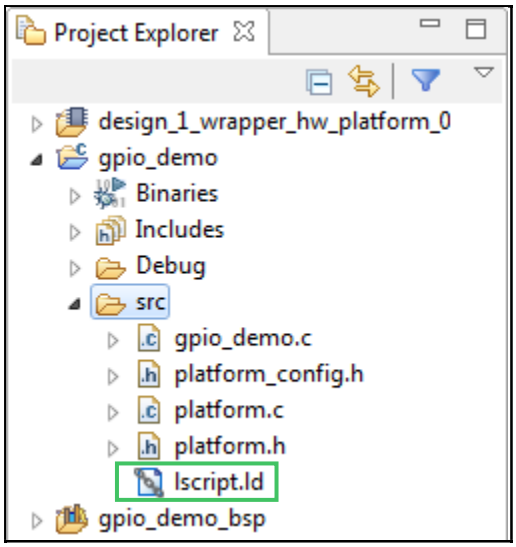


4. When compilation is complete the SDK should display a Console tab similar to this:

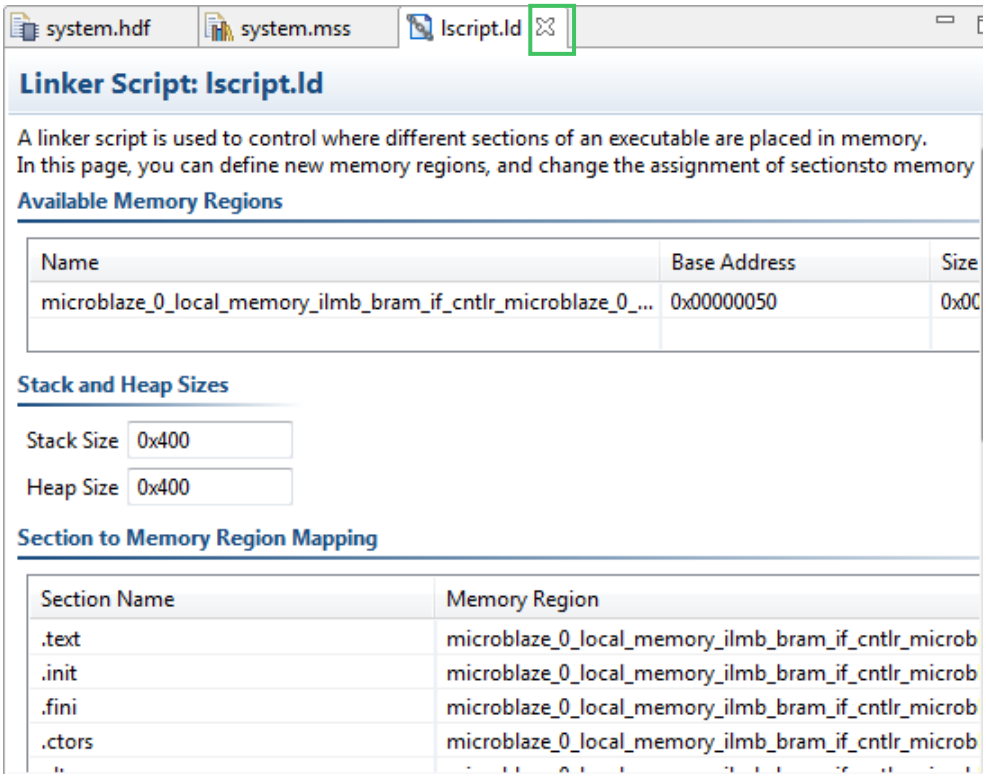


9.2.1 Examine the GPIO Demo Linker Script

1. Navigate the **gpio_demo** software application project source tree in the **Project Explorer** pane and double-click on the **lscript.ld** linker script:



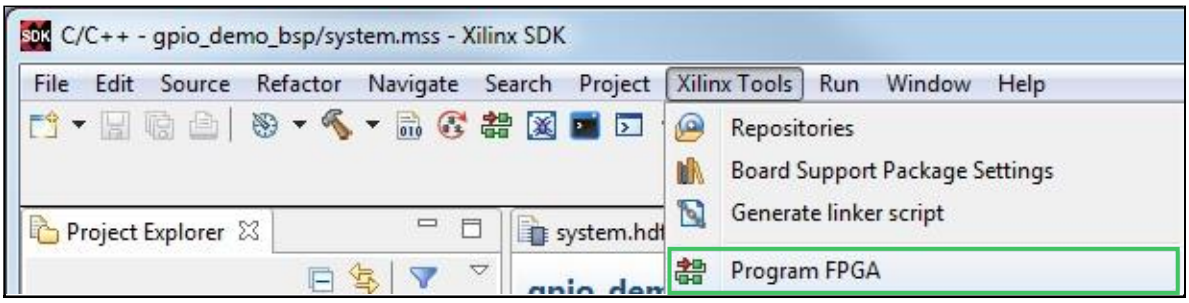
2. Notice that the **Stack Size** and Heap Size are both set to **0x400 (1KB)** and that all of the code sections are mapped to be located in **BRAM** (you will need to scroll the window to see all of the code sections). Click the  on the file tab to close the file.



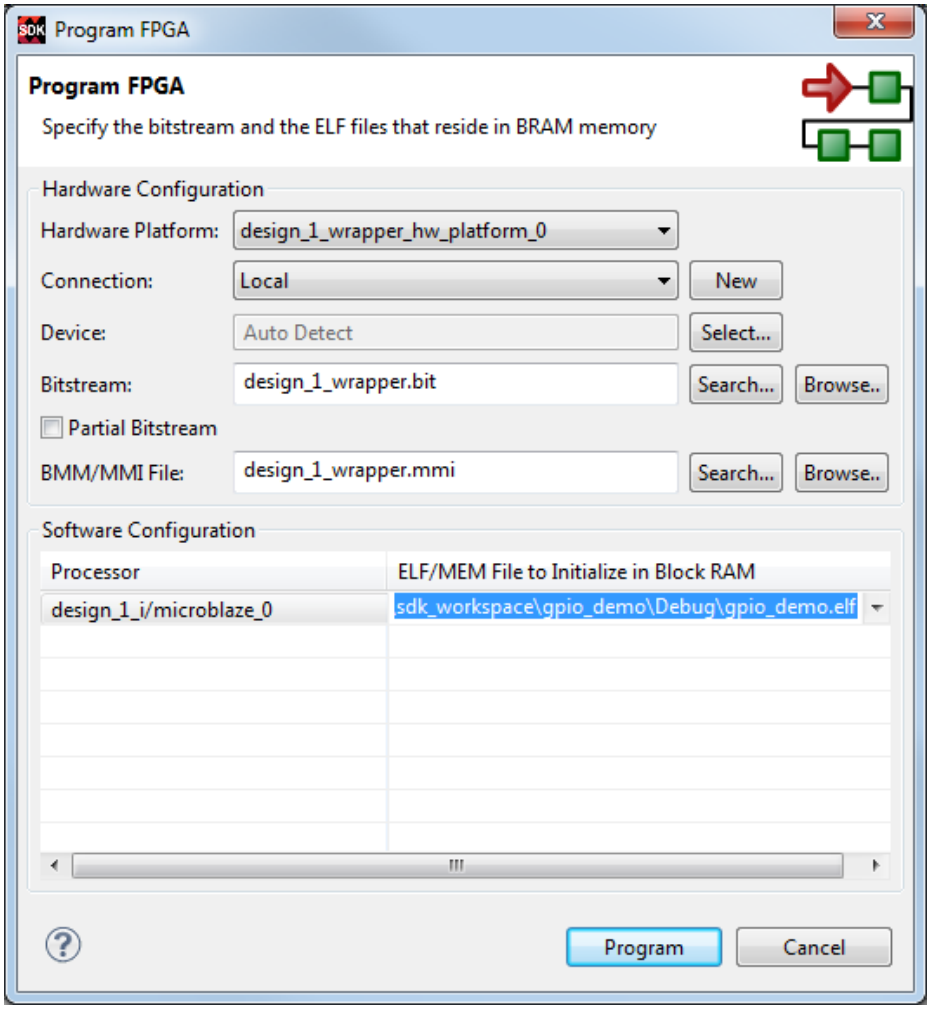
9.2.2 Running the GPIO Demo Software Application

Because the MicroBlaze processor system is configured with 32KB of BRAM, this is enough to hold the gpio_demo software application. The FPGA can be configured with the hardware bitstream merged with this software application, and immediately begin software execution. No need for an intermediate bootloader. We will configure the FPGA now with our hardware bitstream that includes the gpio_demo software application.

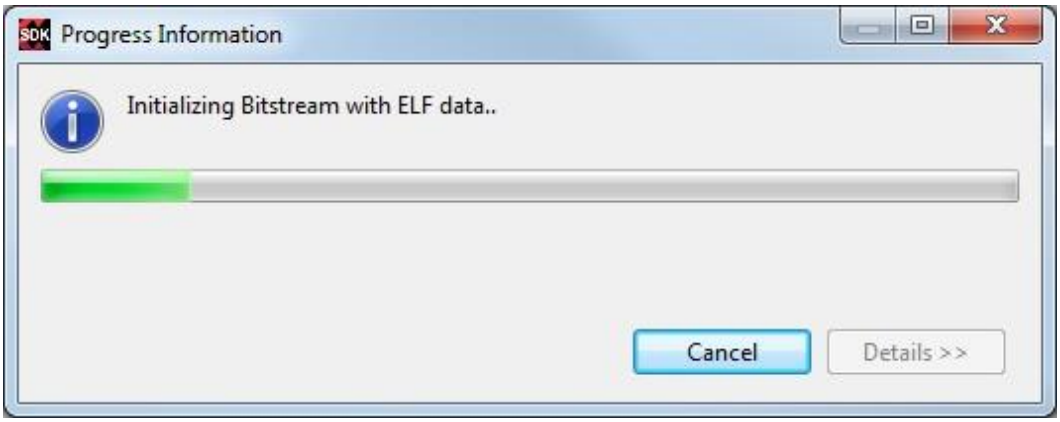
1. If not already done, connect the Arty Evaluation board to the host PC as described earlier in Setting Up the Arty Evaluation Board.
2. Start a serial terminal session using your terminal software of choice and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.
3. In the SDK main menu, select **Xilinx Tools → Program FPGA** or click on the  on the SDK toolbar:



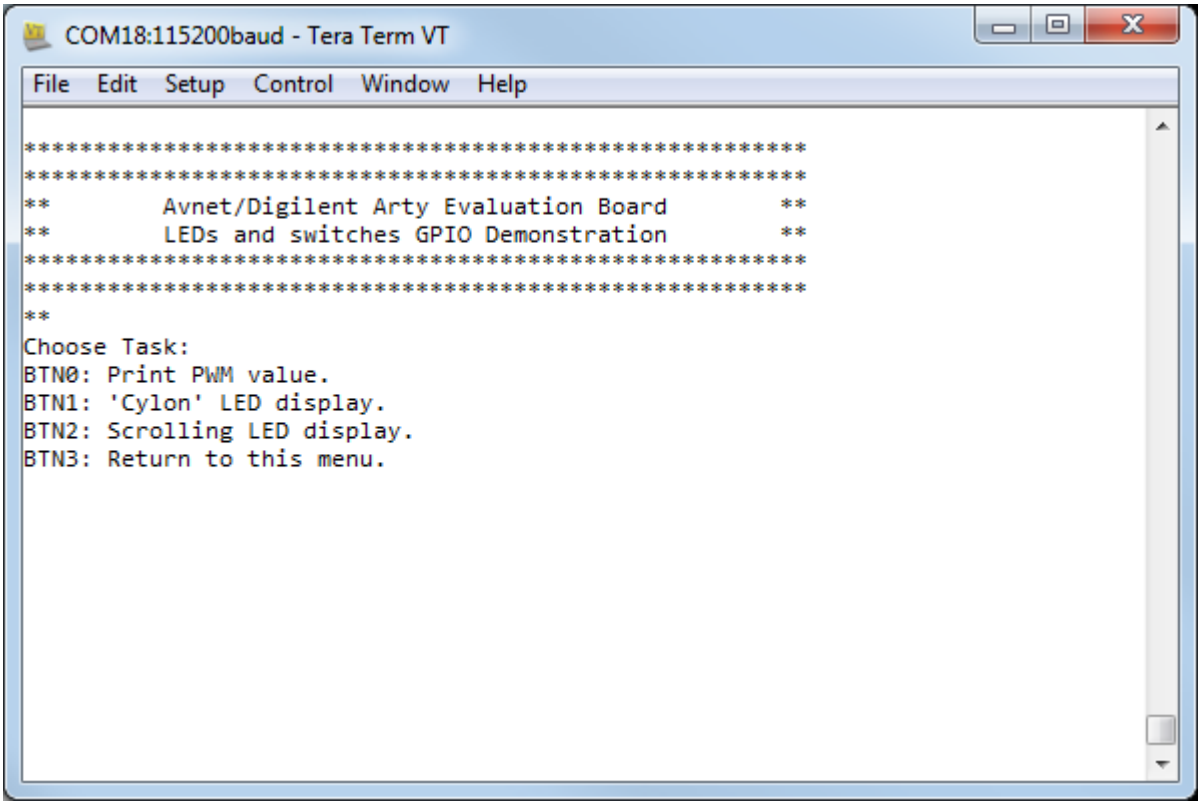
4. Accept the defaults for the **Hardware Platform** and **Connection** and verify that the **gpio_demo** application is selected as the **ELF/MEM File to Initialize in Block RAM**. Click **Program** to continue:



5. You will see the following window while the FPGA bitstream is downloaded:



6. You should see the following on the serial console:



7. You are now ready to run the GPIO demo software application. The steps to run the application are the same as running the demo you probably used earlier, except the steps of downloading the bitstream and application executable are already completed.

10 Program the GPIO Demo Application in QSPI Flash

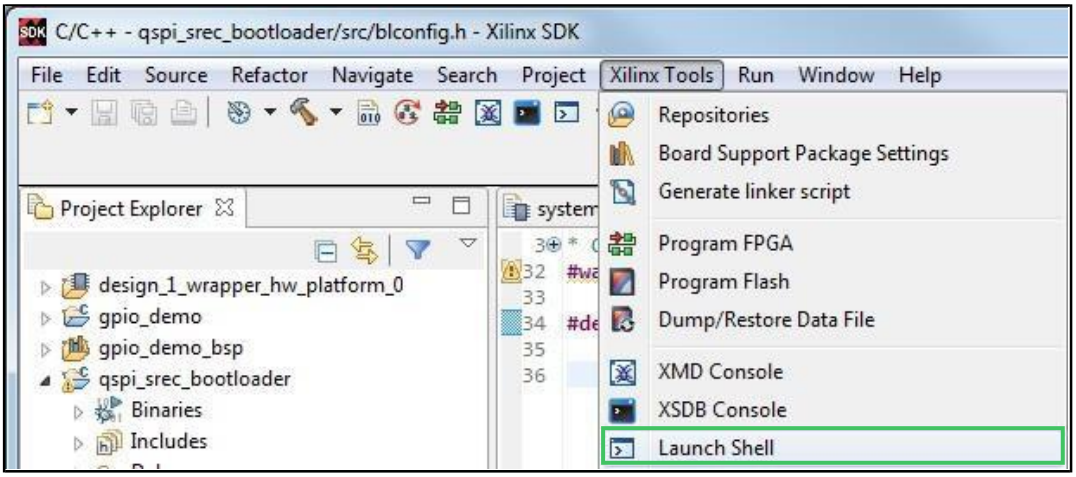
Starting in version 2015.1, the Xilinx SDK now has the capability to program the QSPI flash from the SDK GUI. Previously the Vivado Hardware Manager had to be used for Flash memory programming tasks. Both methods are described here for reference.

10.1 Program the QSPI Flash Using the Vivado Hardware Manager

10.1.1 Prepare the QSPI Programming File

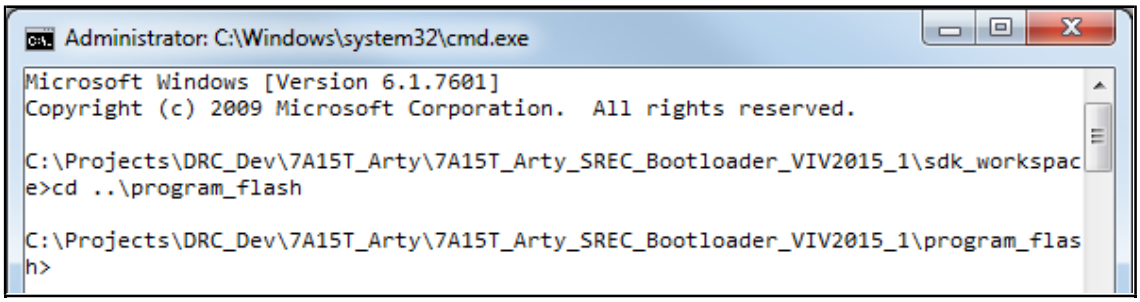
All the components have been assembled at this point, and the last step is to create a FPGA boot image. We will use the Vivado tools to create a boot image for the Artix-7 FPGA combining the FPGA bitstream (with integrated SPI SREC bootloader executable in BRAM) and GPIO demo executable into a single **boot.mcs** Flash programming file.

1. Go back to the command window you opened earlier for running the demo. If you closed that command window you can open another from the SDK. In the SDK GUI navigate to **Xilinx Tools → Launch Shell** or click the  icon on the toolbar.



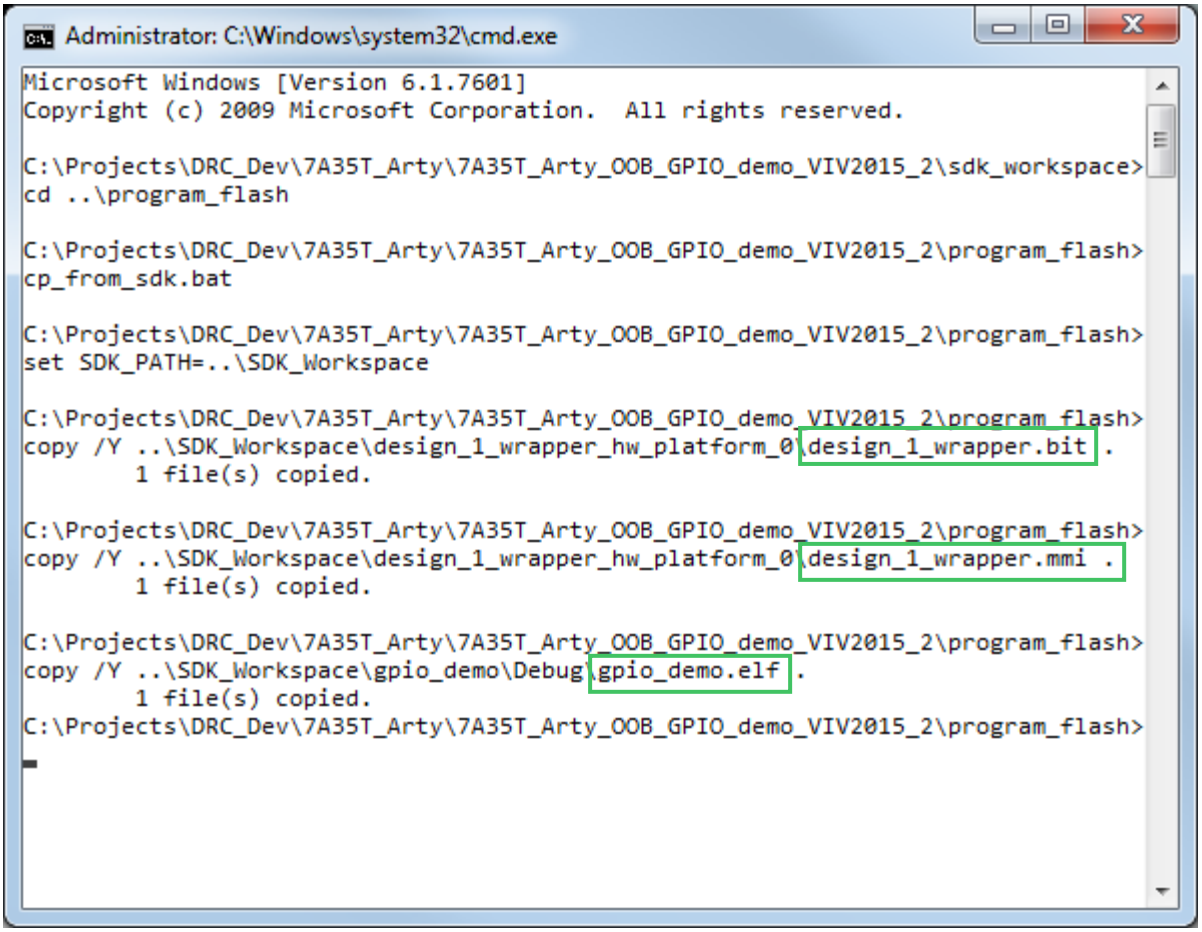
2. Change directories to the **program_flash** folder:

```
cd ..\program_flash
```



3. Run the **cp_from_sdk.bat** batch file to copy the needed files from their locations in the folders of the SDK workspace:

cp_from_sdk.bat



4. Run the **make_mcs.bat** batch file to create the boot.mcs file needed to program into QSPI Flash. This batch file assumes the Vivado tools have been installed in their default locations and checks to see if the required system environment variables have been set. You will need to edit the batch file if your installation of the Vivado tools is in a different folder. The batch file will run the following commands:

   – Merge the gpio_demo.elf with the FPGA bitstream of the MicroBlaze system:

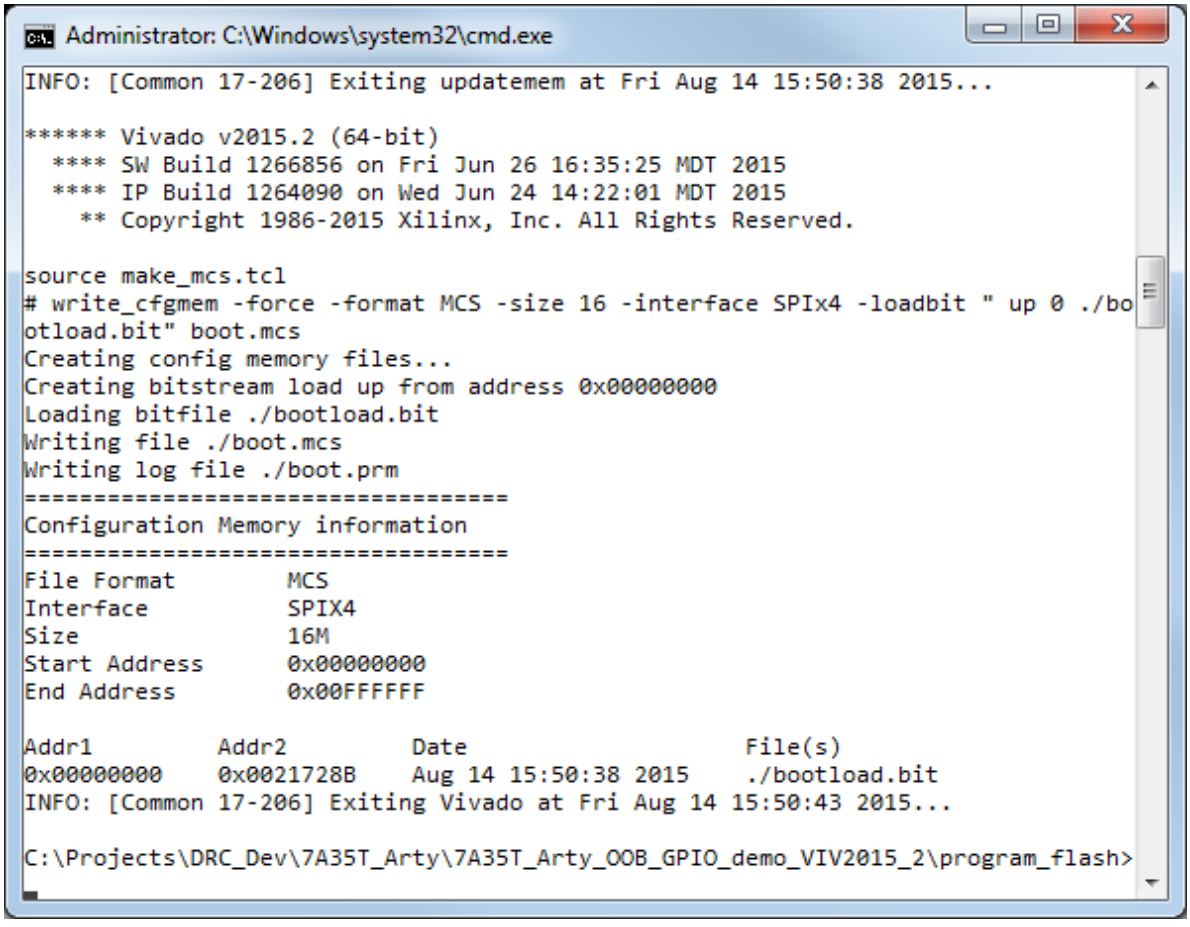
   ```
   cmd /C updatemem -force -meminfo design_1_wrapper.mmi -bit
   design_1_wrapper.bit -data gpio_demo.elf -proc
   design_1_i/microblaze_0 -out bootload.bit
   ```

   – Start Vivado in batch mode to run the TCL script to create the boot.mcs QSPI programming file:

   ```
   call C:\\Xilinx\\Vivado\\%VIVADO_VER%\\.\\bin\\vivado.bat -
   mode batch - source make_mcs.tcl
   ```

   – The **make_mcs.tcl** file:

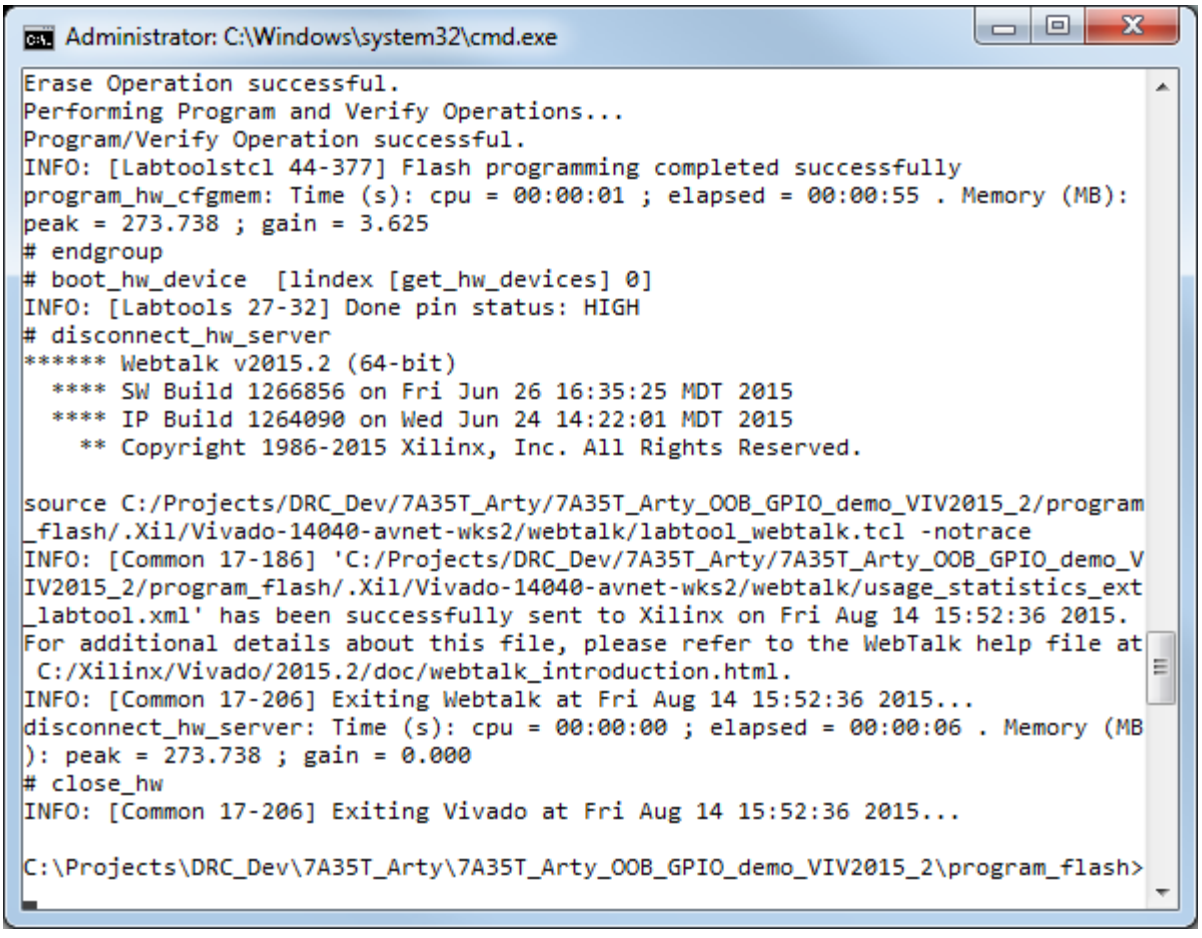
   ```
   write_cfgmem -force -format MCS -size 16 -interface SPIx4 -
   loadbit " up 0 ./bootload.bit" boot.mcs
   ```



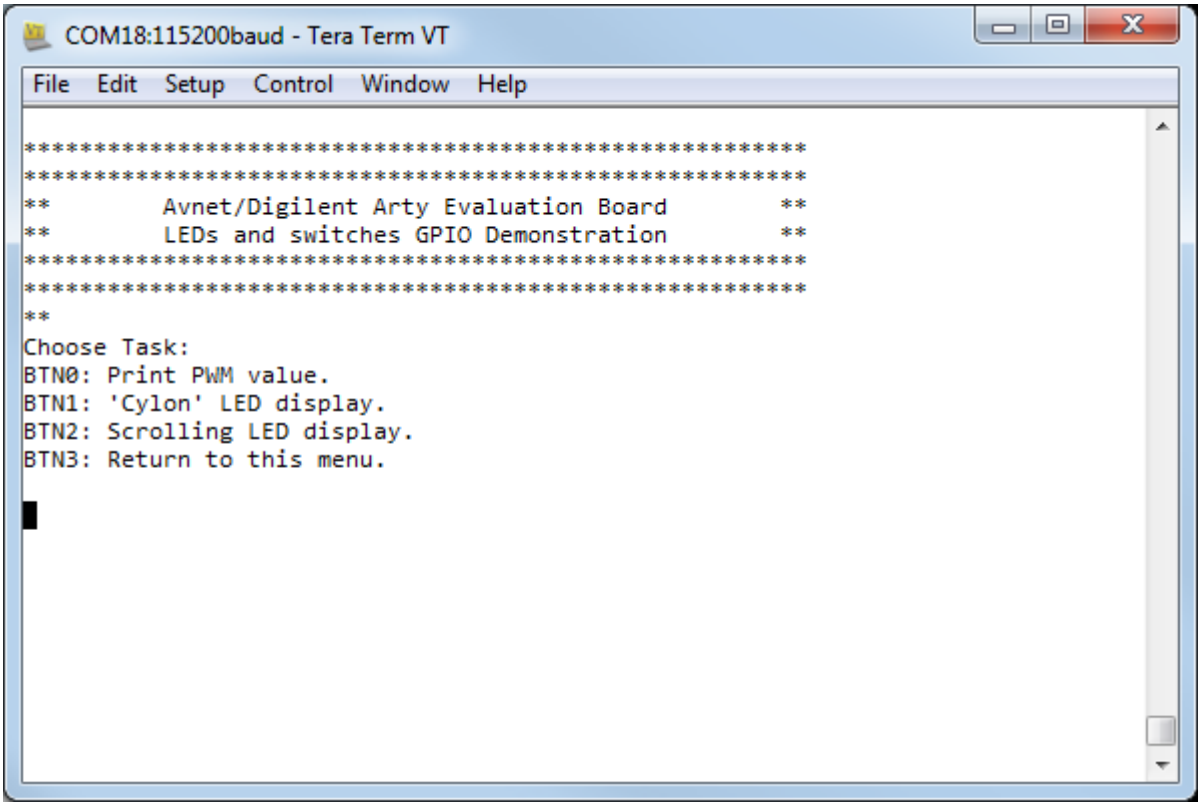
10.1.2 Program the QSPI Flash Using a TCL Script

1. If not already done, connect the Arty Evaluation board to the host PC as described earlier in Setting Up the Arty Evaluation Board.
2. Start a serial terminal session using your terminal software of choice and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control
3. Run the **program_qspi.bat** batch file to program the QSPI Flash. This batch file assumes the Vivado tools have been installed in their default locations and checks to see if the required system environment variables have been set. You will need to edit the batch file if your installation of the Vivado tools is in a different folder. The batch file will run the following command:

   – Start Vivado in batch mode to run the TCL script to program the boot.mcs file to QSPI Flash:
   `call C:\\Xilinx\\Vivado\\%VIVADO_VER%\\.\\bin\\vivado.bat -mode batch - source program_qspi.tcl`

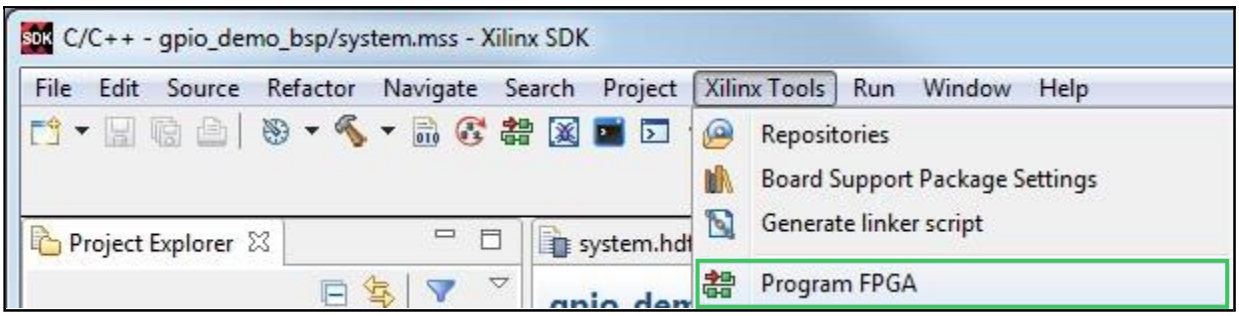


4. When the QSPI Flash programming has completed you should see the Arty board reconfigure and launch the GPIO Demo application. This can also be tested by pressing the **PROG** switch on the Arty board (near LD8 and the USB JTAG/UART port). You are now ready to run the GPIO Demo software application. The steps to run the application are the same as running the demo you probably used earlier, except the steps of downloading the bitstream and application executable are already completed. Open a command window in the **<installation>\demo** folder and run the **cp_from_sdk.bat** batch file script to copy the new bitstream and software ELF files to the demo folder. Go back and rerun the GPIO Demo. This concludes this design tutorial.

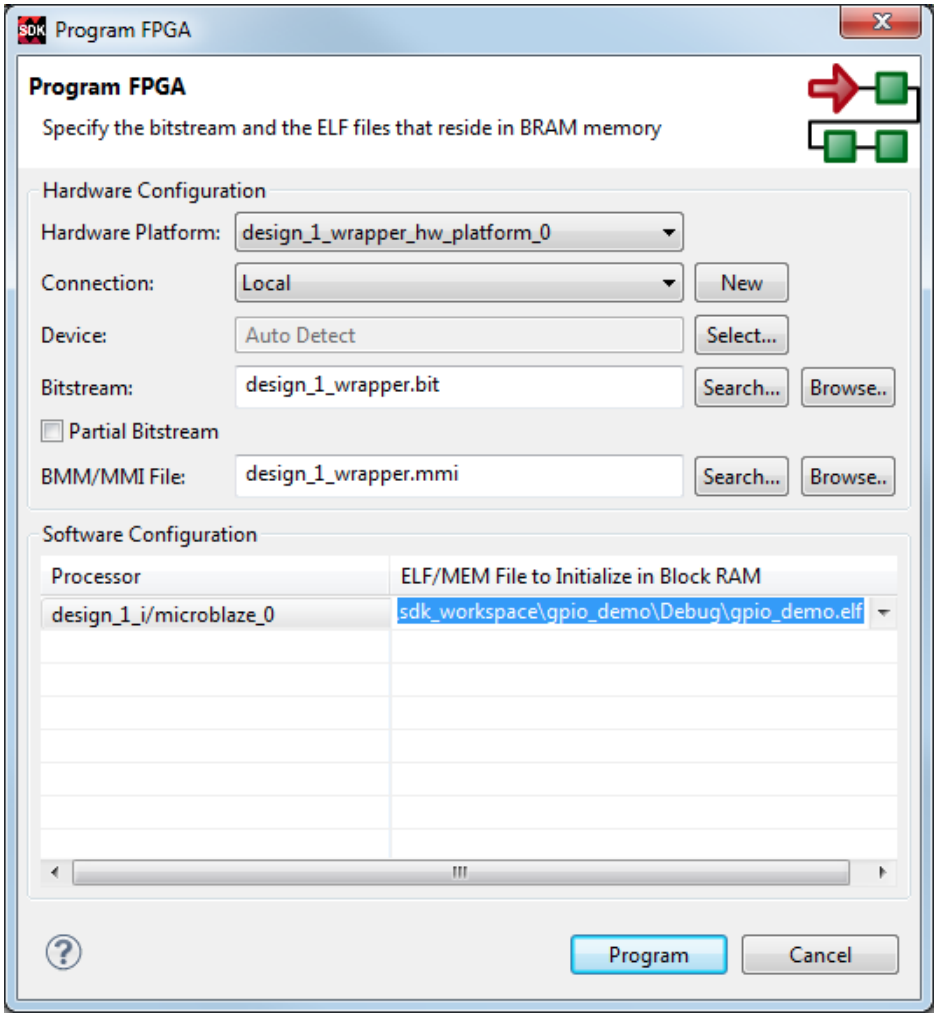


10.2 Program the QSPI Flash Using the SDK

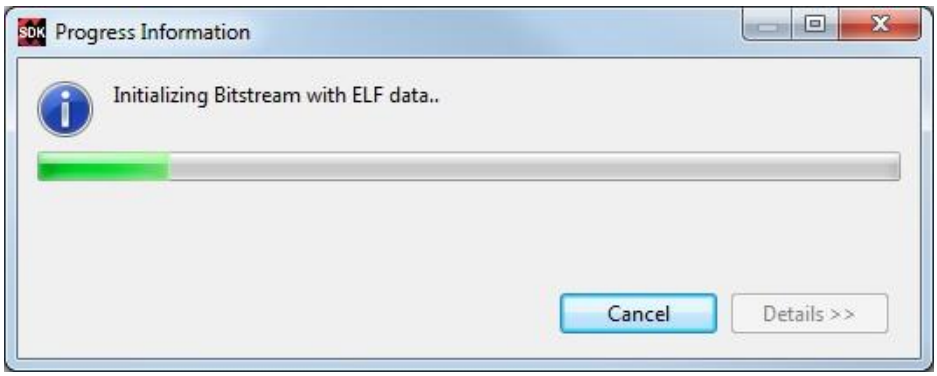
1. If not already done, connect the Arty Evaluation board to the host PC as described earlier in Setting Up the Arty Evaluation Board.
2. Start a serial terminal session using your terminal software of choice and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.
3. We must first create a bitstream with the QSPI SREC bootloader application merged with the FPGA bitstream. In the SDK main menu, select **Xilinx Tools → Program FPGA** or click on the  on the SDK toolbar:



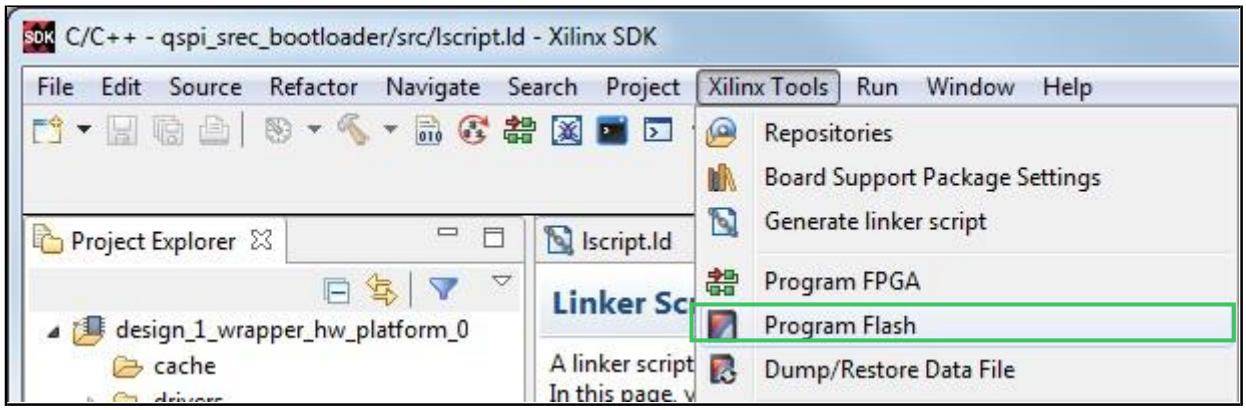
4. Accept the defaults for the **Hardware Platform** and **Connection** and verify that the **gpio_demo** application is selected as the **ELF/MEM** File to Initialize in **Block RAM**. Click Program to continue:



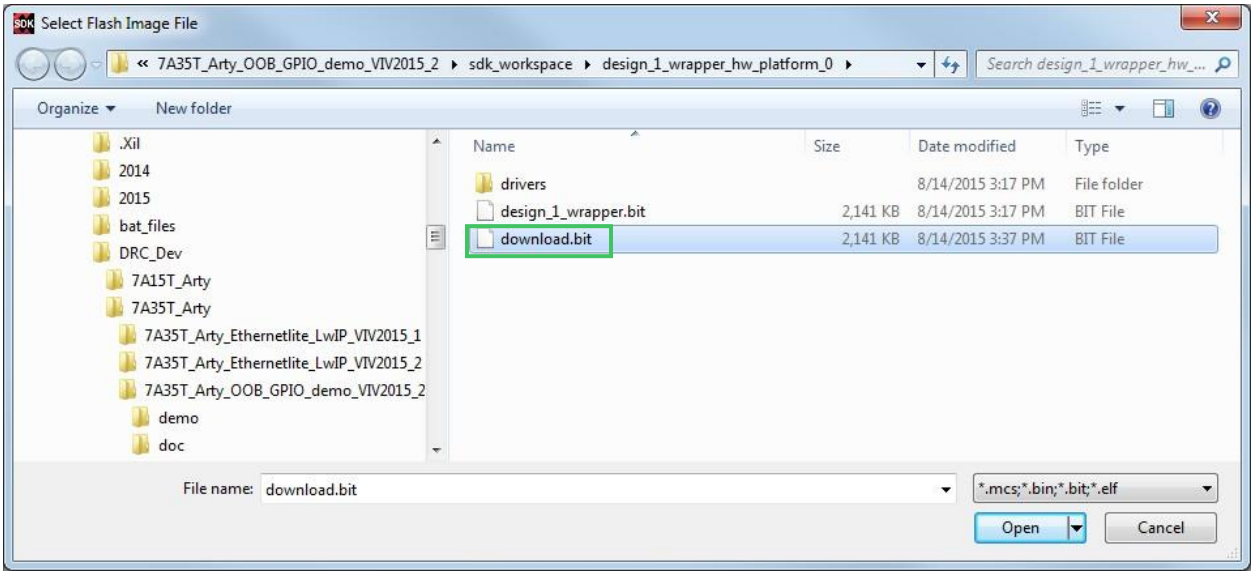
5. You will see the following window while the FPGA bitstream is downloaded:



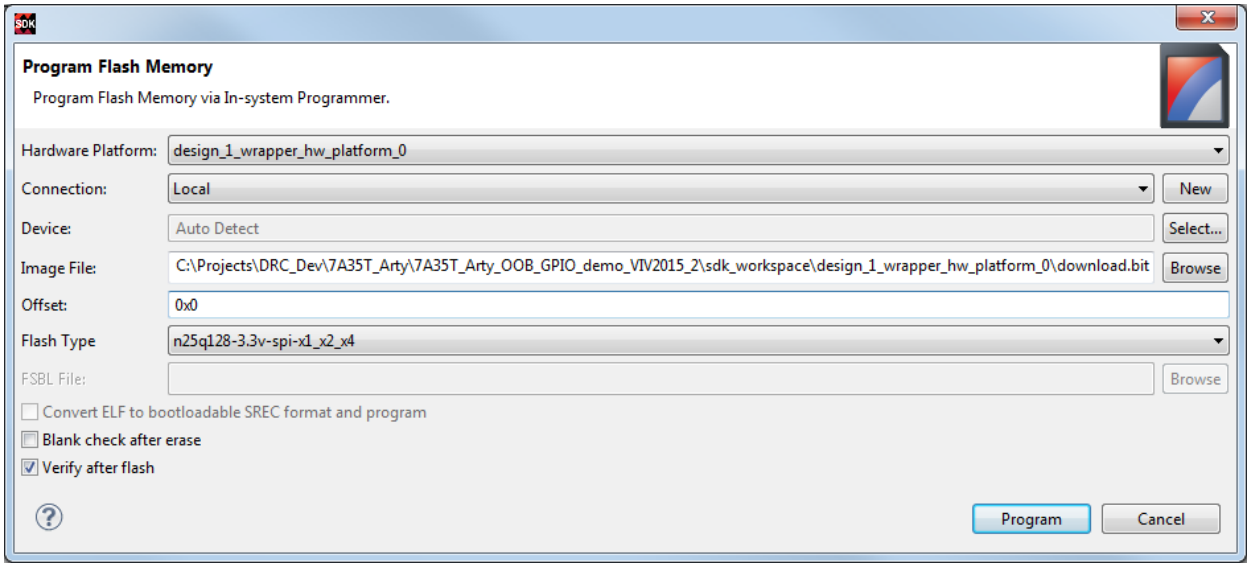
6. In the SDK GUI navigate to **Xilinx Tools → Program Flash**.



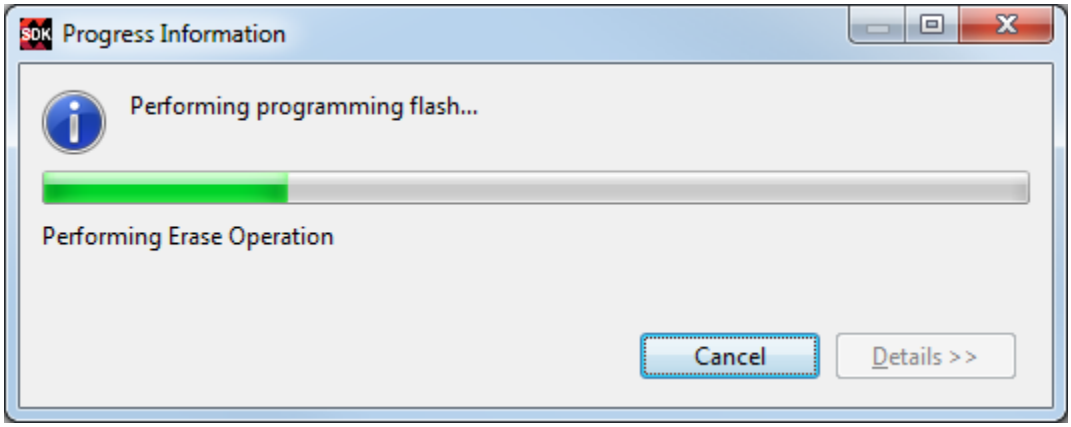
7. Programming the QSPI using the SDK requires two steps. The first step is to program the bitstream with the integrated QSPI SREC bootloader application to the QSPI Flash. Click **Browse** and navigate to the **<installation>\sdk_workspace\design_1_wrapper_hw_platform_0** folder and select the **download.bit** bitstream file. Click **Open** to continue:



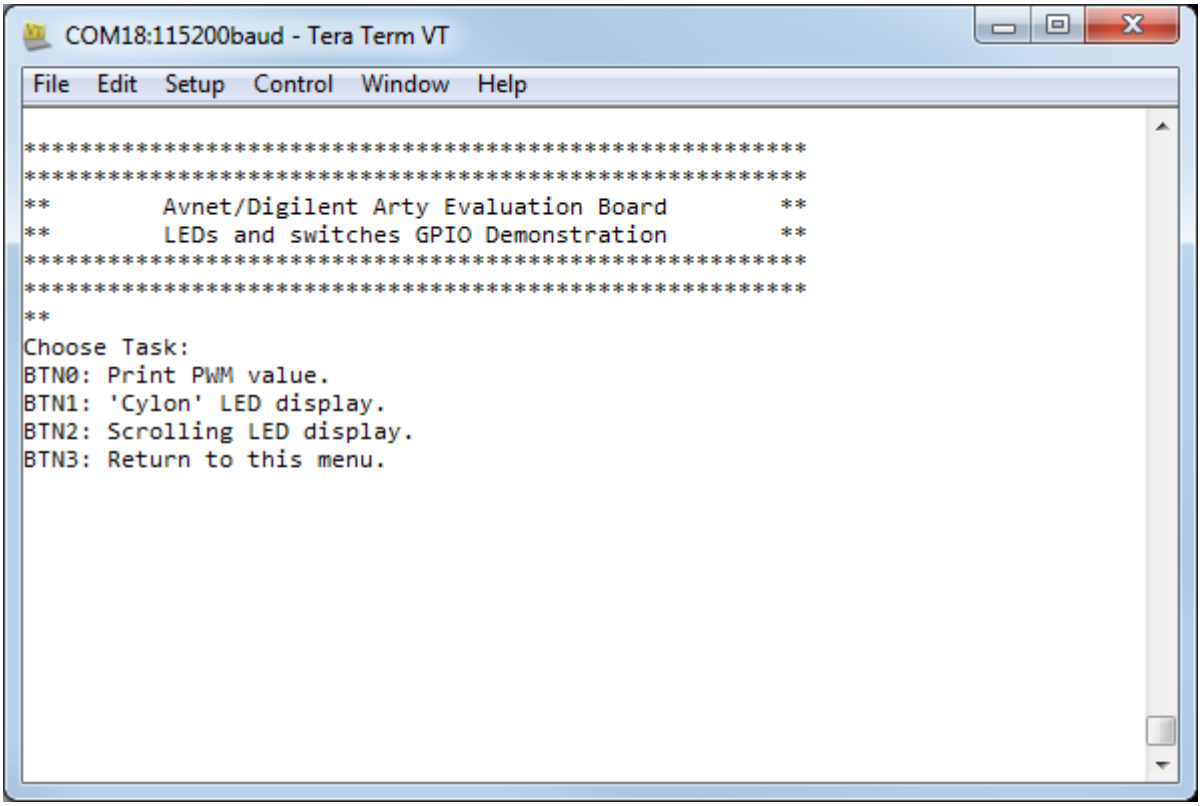
8. Select the following settings and click **Program** to continue:
   - Set the **Offset** to **0x00000000**
   - Set the **Flash Type** to **n25q128-3.3v-spi-x1_x2_x4**
   - Enable **Verify after flash**



9. You will see a status window while the bitstream is programmed to Flash:



10. Test that the QSPI Flash programming was successful by pressing the **PROG** switch on the Arty board (near LD8 and the USB JTAG/UART port). You should see the Arty board reconfigure, launch the SREC bootloader, and boot the GPIO Demo application. You are now ready to run the GPIO Demo software application. The steps to run the application are the same as running the demo you probably used earlier, except the steps of downloading the bitstream and application executable are already completed. Open a command window in the **<installation>\demo** folder and run the **cp_from_sdk.bat** batch file script to copy the new bitstream and software ELF files to the demo folder. Go back and rerun the GPIO Demo. This concludes this design tutorial.



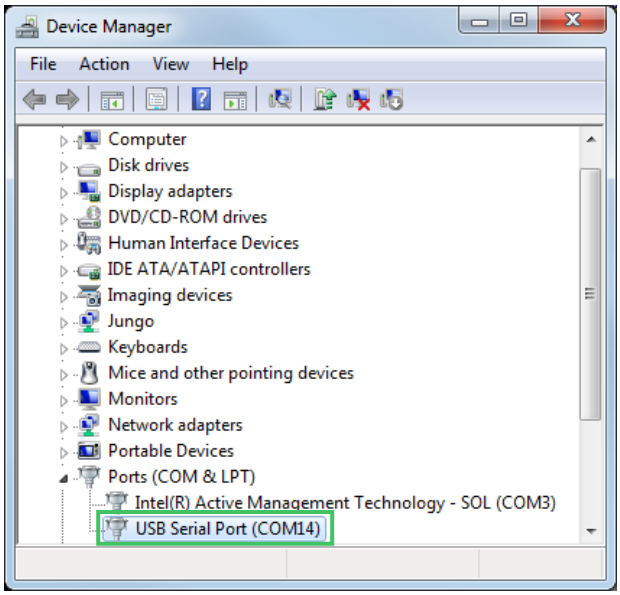
11 Appendix I: Determining the Virtual COM Port

Now you can connect the evaluation board's USB-to-UART port to one of the USB ports on your PC. The new hardware detection will pop up and enumeration of the driver will be started. Once finished a virtual COMx port is created and you are ready to setup a connection using Windows HyperTerminal or comparable serial terminal emulation utility. Follow these instructions to determine the COMx port assigned to the USB-to-UART bridge:

1. Open the Device Manager by right-clicking on 

, select **Properties**, then click on the **Device Manager**.

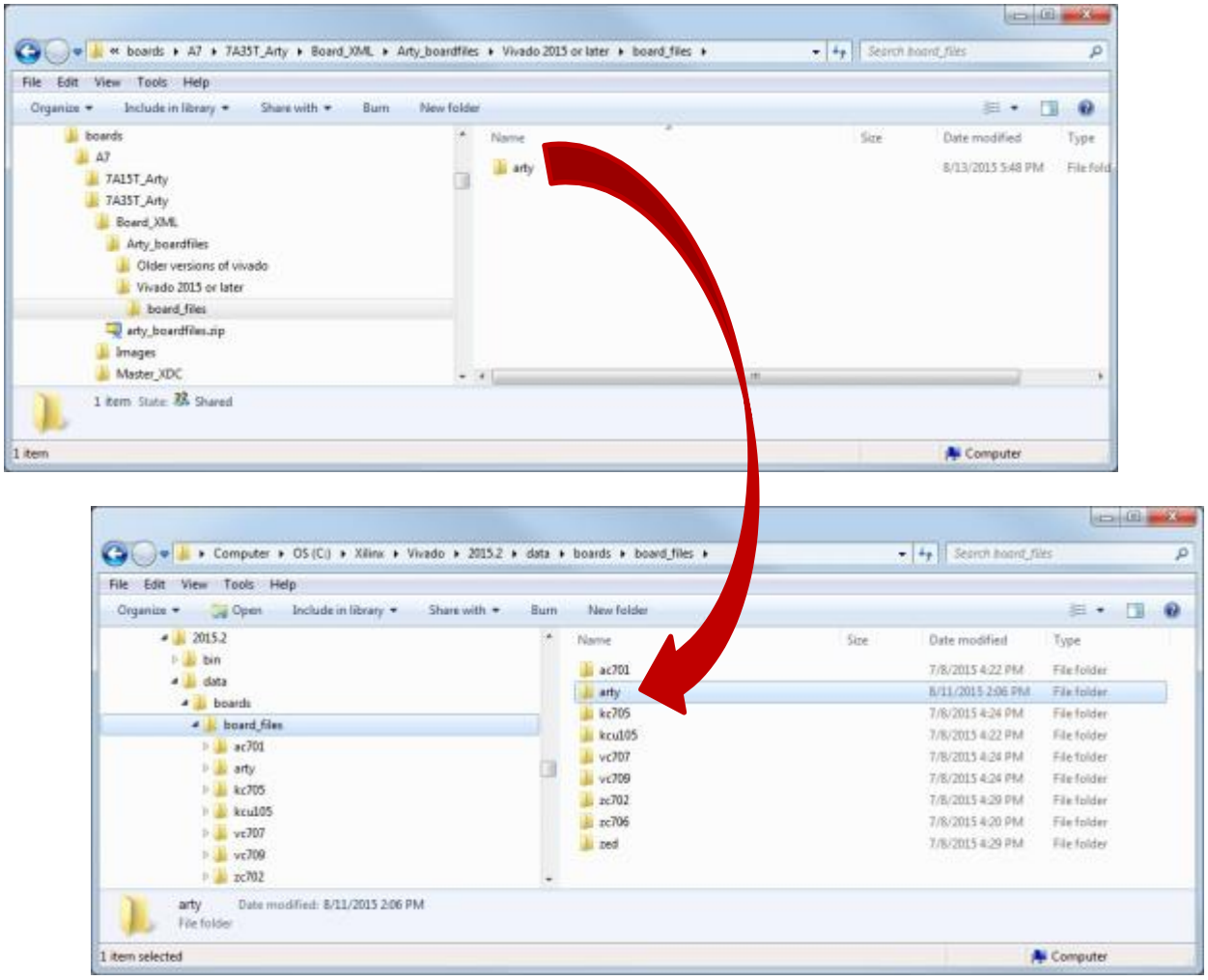
2. In the Device Manager, scroll down to Ports and expand the list. You will see the USB Serial Port and its assigned COM port. In the example below, it is COM14. Make note of this COM port number for use with the serial terminal you will use elsewhere in this design tutorial. This concludes these USB UART driver and virtual COM port installation instructions.



12 Appendix II: Installation of Board XML Files

Many Avnet evaluation boards can now be targeted directly in the Xilinx Vivado tools. This greatly simplifies the task of building the MicroBlaze processor system and integrating system peripherals and board interfaces into your own custom designs. Follow the instructions below to install the board XML files for the Arty board into your Vivado installation folders:

1. Using your favorite web browser, download the zip archive of board XML files from the Digilent wiki site for the Arty Evaluation Board: https://reference.digilentinc.com/_media/arty_boardfiles.zip
2. Extract the zip file in the download folder and in Windows Explorer, copy the **arty** folder from the **<installation>\Arty_boardfiles\Vivado 2015 or later\board_files** folder to **<Vivado_install>\Xilinx\Vivado\2015.2\data\boards\board_files**:



13 Appendix III: Windows 260 Character Path Limit

If you are using the Vivado Design Suite on a Windows-7 host, you may run into issues resulting from Vivado pathnames exceeding the maximum allowed. Vivado projects create a very deep file hierarchy, and it becomes very easy to violate the Windows limit if the project is not extracted near the root of the drive. This can even happen when the archive is decompressed, depending on where you choose to place the project in your existing file hierarchy.

It is not always convenient to place every Vivado project at the root of a drive. To work around this limitation, the recommended procedure is to place the Vivado archive in a shared folder on your host machine, then use Windows Explorer to map a network drive to the directory where the archive will be decompressed. This allows the Vivado project to be mapped to the root of the virtual (mapped) directory, eliminating any path issues.

Vivado also makes use of the Windows temp folder, which may be located several folders deep from the root drive, and this can also cause problems. You can create your own temporary directory in C:\temp, and force Vivado to use the new folder with the following TCL:

```
set_param "project.customTmpDirForArchive"      C:/temp
```

For further information, see the Xilinx answer record at: http://www.xilinx.com/support/answers/52787.html.

14 Appendix IV: Getting Help and Support

14.1 Avnet Website

- Evaluation Kit home page with Documentation and Reference Designs
  http://em.avnet.com/arty
- Avnet support forums
  http://community.em.avnet.com/

14.2 Xilinx Website

- Details on the Artix-7 FPGA family are included in the following Xilinx documents:
  - *Artix-7 Family Overview (DS180)*
  - *Artix-7 FPGA Data Sheet (DS181)*
  - *Artix-7 FPGA Configuration User Guide (UG470)*
- For more detailed information about xilisf, please refer to the Xilinx OS and Libraries Document Collection located here:
  www.xilinx.com/support/documentation/sw_manuals/xilinx2015_2/oslib_rm.pdf
- For more detailed information about SDK, please refer to the SDK Help HTML:
  www.xilinx.com/support/documentation/sw_manuals/xilinx2015_2/SDK_Doc/index.html
- Xilinx support forums
  http://www.xilinx.com/support.html

14.3 Digilent Website

- Arty home page
  http://www.artyboard.com
- Arty Wiki
  https://reference.digilentinc.com/arty