Introduction

The ZedBoard can be programed from multiple sources including a JTAG port, QSPI flash chips or an SD card. Once a project has been completed and debugged using the JTAG port it can be packaged and placed in the QSPI flash chip or SD card so that the ZedBoard will self-program when power is applied.

The procedure for the QSPI chip will be demonstrated.

Design

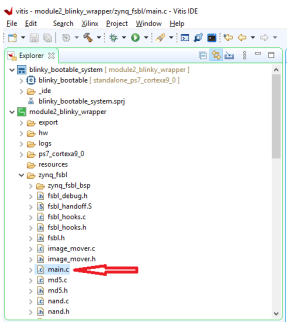
If the completed design contains only a BIT file (FPGA PL, the PS is not used) the process is well documented using the Xilinx Impact software. But if the project is a combination of PL and PS there will be two additional files required, the ELF file for the application and a second ELF file configured as a FSBL (First Stage Boot Loader).

For this tutorial, the design will be based on the final code developed in Tutorial Module 2. One LED will blink under control of the PL while the second LED will blink under control of the PS while the PS continues to send messages out its UART and accept a single byte command to turn on or off the blinking of the PS controlled LED.

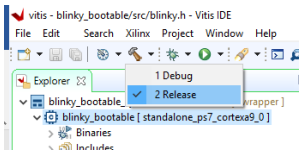
Procedure

FPGA chips in general do not remember their configuration when the power is removed. There are variants available with internal FLASH memory which will self-program when power is applied but typically the FPGA image is held in an external FLASH chip. When power is applied, the FPGA will check for the presence of the external FLASH, then check that a properly formatted file resides within the FLASH. Then the FPGA loads the image, does any necessary housekeeping and completes its boot.

In the case of the ZYNC the boot image contains three parts, the PL image, and two PS images/programs. The first program is known as the FSBL or First Stage Boot Loader. This program contains just enough executable code to know how to read in the larger application program, unpack it, configure the FPGA and finally jump the execution to the application. FSBLs can be made smart and able to check various FPGA aspects to determine exactly which application is to be run. But in our case, we want the simplest form possible. Prior to Vivado 2020, the designer was required to include an FSBL project in their design. Vivado 2020 automatically includes the FSBL project under the module wrapper portion of the project. Do not modify this code unless you are absolutely certain that it is necessary.

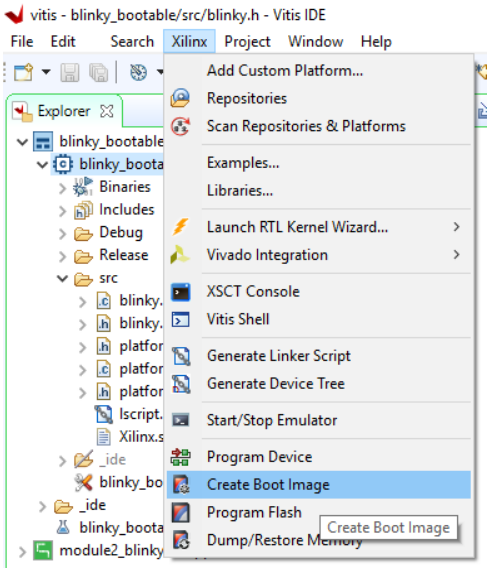


Change from the Debug to Release configuration using the menu bar, the tool bar or by right-clicking on the application in the Assistant tab and selecting “Release”.



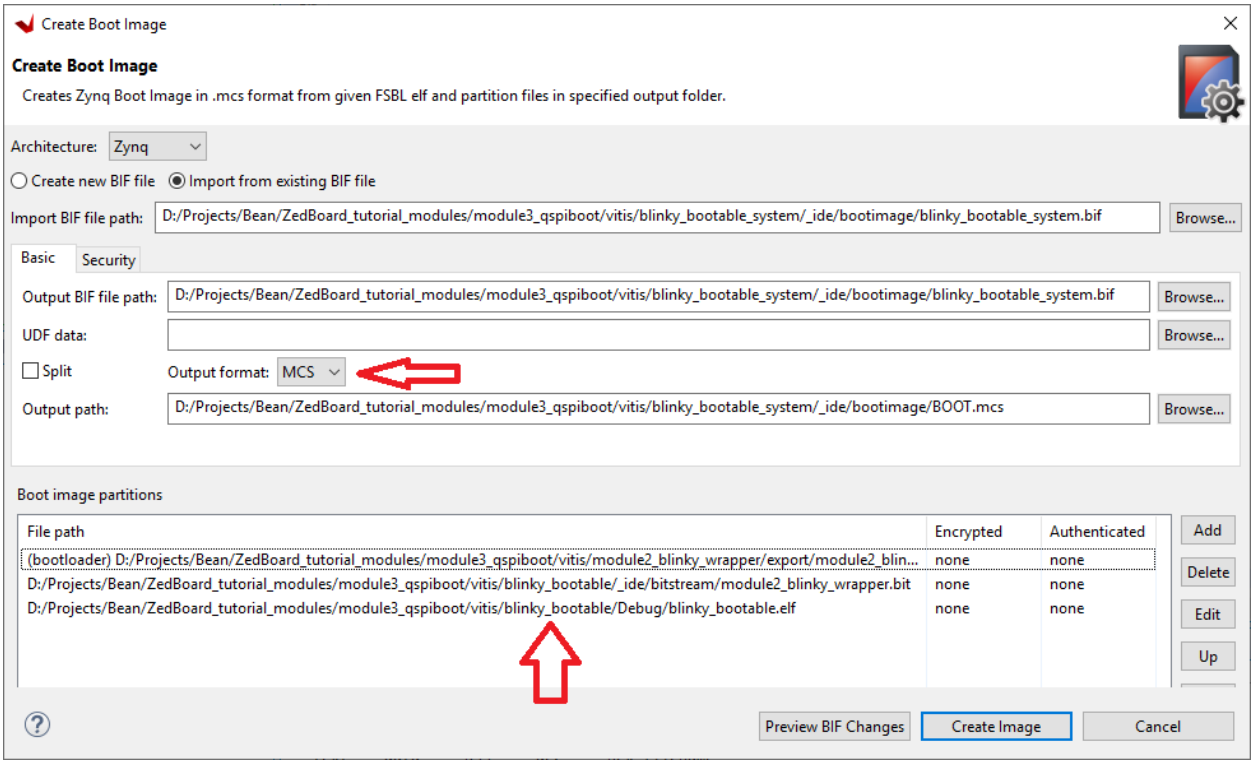
Build the project. You may notice that the overall size of the resultant ELF file has decreased. This is because the debug mode includes code that while mostly hidden to you, facilitates the operation of the debugger. Typically, code deployed in a device will not contain the debug code hooks.

The final step before programming is to create a BIF (Boot Image Format) and either a BIN (binary) or MCS (based on the Intel MCS-86 hexadecimal object file format) image of the entire FPGA programming data. The MCS is a better choice if the FPGA is to be booted from a small serial FLASH chip such as is possible on the ZedBoard board. To create the BIF and MCS file, select "Create Boot Image" from the Xilinx menu at the top of Vitis.



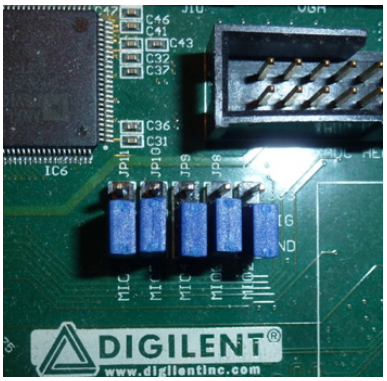
The BIF file can be created as new (this is automatically selected the first time) or from settings found in a previously created BIF file (the default for subsequent file creations). The file paths for this operation get quite long so you may need to re-size the dialog or use the "Browse" button to see the full path names. Accept the default paths for these files.

The "Output format" should be set to MCS. And at the bottom of the dialog there should be three files listed for the boot image partitions, the bootloader, the BIT file which is the PL and PS configuration from Vivado and the ELF file which is the result of your programming for the PS.

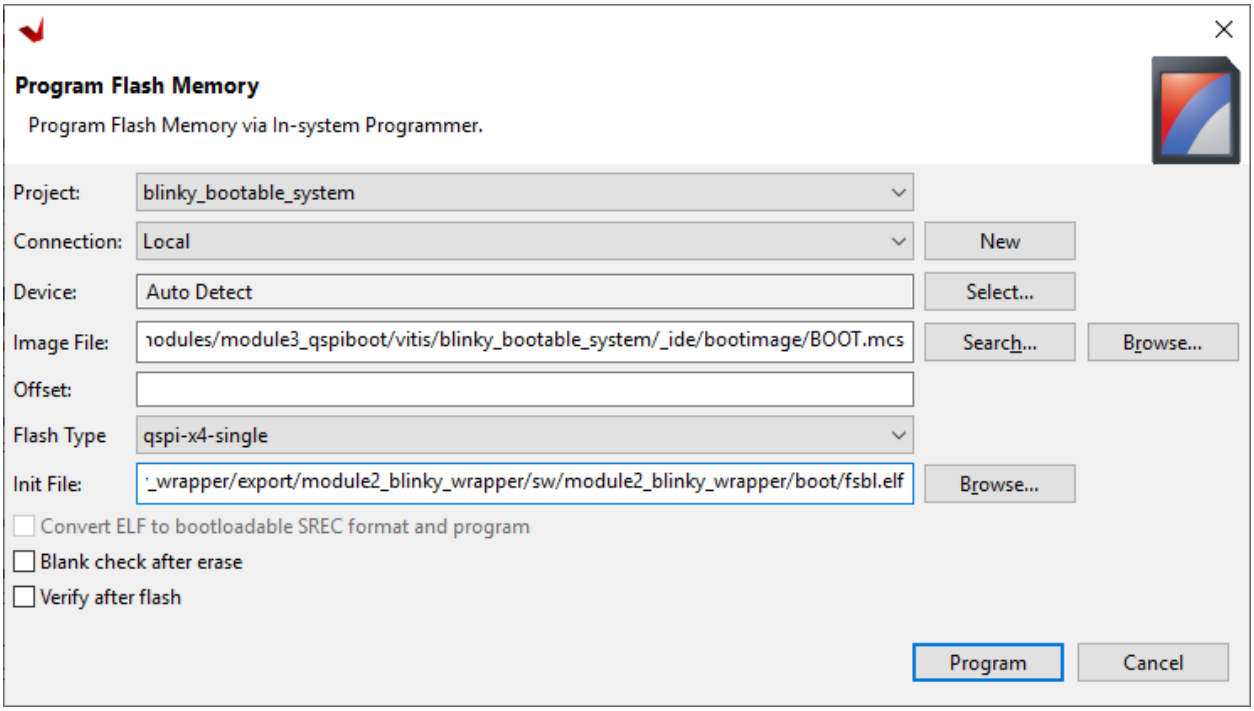


It is possible to include encryption to protect a file from being copied or otherwise easily hacked. This would be a consideration for commercial product development but isn't important in this example. So The Encrypted and Authenticated columns will remain as "none".
Finally create the image by clicking on "Create Image". There will be a short pause while an MCS file is produced. The file will live in a directory under your Vitis directory as ./project_name/_ide/bootimage/ and has a default name of BOOT.MCS.

The final step is to program the ZedBoard's serial flash. Before programming the ZedBoard's flash, it is important that the jumpers on JP7, JP8, JP9, JP10 and JP11 all be set to the GND (lower) position.



Return to the Xilinx menu and select Program Flash Memory. The file listed in Image File should be the BOOT.MCS file you just created. If the default path locations were left alone in the BIF/MCS creation step this should already appear in the dialog box. The Flahs Type should e set to "qspi-x4-single", again this should be pre-selected. The options for a blank check and verify after flash will be unselected and can be selected for extra levels of verification. Clicking "Program" starts the process. This can take several seconds to complete.



To test the programming has worked, turn off the power to the ZedBoard and change the position of JP10 from the GND position (down) to the 3V3 (up) position. Then turn on power. After about ½ second the blue LED labeled DONE should light and LD0 and LD1 should start to blink. If you have a terminal program such as TeraTerm running the serial port should be capturing the output as in Module 2.

Near the blue DONE LED is a button labeled PS-RST which will send a reset command to the PS and restart your program. The TeraTerm terminal should reflect this reboot of the software.

Updates and Debugging

Even after programming the external FLASH, it is possible to continue to use Vitis as before in a debugging mode with JP10 set to 3V3. Sending a new ELF and running the debugger through the JTAG connection will work as before. However unless the MCS file is update and re-flashed, when the power is next cycled, the older version of software will again boot from the FLASH.