Using the Spansion Parallel NOR Flash with MicroBlaze on the Spartan-3A Evaluation Kit

Version 10.1.01

Revision History

Version	Description	Date
10.1.00	Initial draft release	July 9, 2008
10.1.01	Added location of board forum	Sept 5, 2008

Overview

The *Creating Your First MicroBlaze Design with the Spartan-3A Evaluation Kit* tutorial showed how to make use of XPS, BSB, and an XBD file to create and use a MicroBlaze soft processor system for the Avnet Spartan-3A Evaluation Kit. The final design was exercised using sample code generated by BSB.

This tutorial expands on that tutorial to show how to use a custom application with unique code to exercise and test the parallel Flash. Once the parallel Flash is verified, then read-only code can be placed in Flash and executed in place. This tutorial outlines the steps to accomplish this.

Objectives

This tutorial will demonstrate how to do the following:

- Create a new software application
- Add code to the application
- Build the design and initialize the application into BRAM
- Run a parallel Flash test
- Partition and store a piece of the design in Flash
- Run a design while executing the read-only portion of the application in Flash

Requirements

The following items are required for completion of this tutorial.

Software

The following software setup was used to test this reference design:

- WindowsXP 32-bit Service Pack 2
- Xilinx ISE software, version 10.1 with Service Pack 2[1]
- Xilinx EDK software, version 10.1 with Service Pack 2
- Avnet Programming Utility (AvProg) v3.3.7

Hardware

The hardware setup used by this reference design includes:
Computer with a minimum of 330-540 MB (depending on O/S) to complete an XC3S400A design[2]

- Avnet Spartan-3A Evaluation Kit
  - Avnet Spartan-3A Evaluation board
  - USB cable

Setup

- Install ISE and EDK software. Apply service packs.
- Install AvProg software and the Sp3A Eval driver as described in the *Avnet Programming Utility User Manual*.

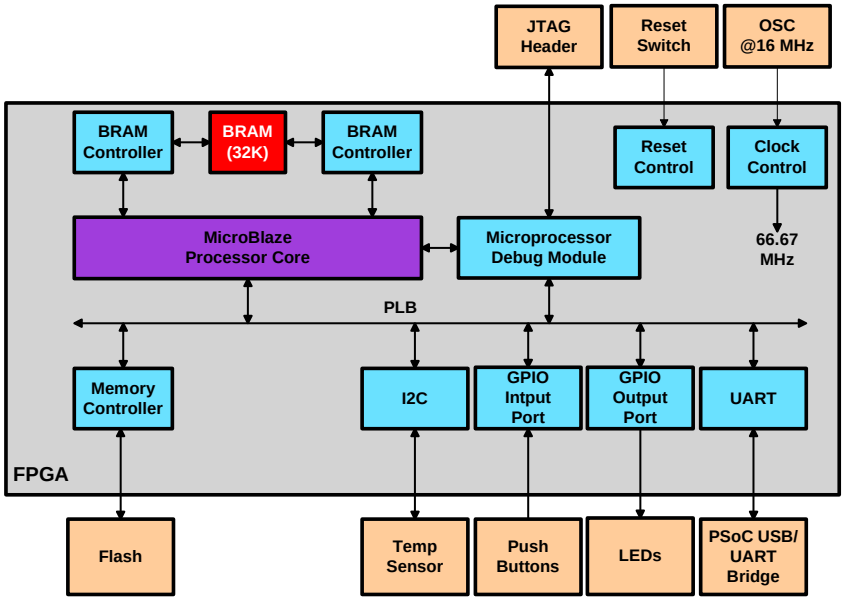
Recommended Reading

- All Avnet documents are available at www.em.avnet.com/spartan3a-evl
- The hardware used on the Spartan-3A Evaluation board is described in detail in Avnet document *Spartan-3A Evaluation Kit User's Guide*.
- BPI configuration for this board is described in detail in Avnet document *Configuring an FPGA from Parallel Flash with the Spartan-3A Evaluation Kit*
- Details on the Spartan-3A FPGA family are included in the following Xilinx documents:
  - *Spartan-3A FPGA Family: Data Sheet* (DS529)
  - *Spartan-3 Generation FPGA User Guide* (UG331)
  - *Spartan-3 Generation Configuration User Guide* (UG332)
- An explanation of the process for executing MicroBlaze code directly from Flash is contained in a Xilinx Answer Record, Application note, and Reference Manual:
  - *AR #23748 - 10.1 EDK - How can I execute MicroBlaze code directly from flash?* (http://www.xilinx.com/support/answers/23748.htm)
  - XAPP983 (http://www.xilinx.com/support/documentation/application_notes/xapp983.pdf)
  - *Embedded System Tools Reference Manual* (in %XILINX_EDK%/doc directory). See the *Gnu Compiler Tools* chapter
- Spansion S29GL Flash datasheet
  - http://www.spansion.com/products/S29GL032N.html
  - http://www.spansion.com/datasheets/s29gl-n_01_10_e.pdf

[1] ISE latest Service Pack is available at www.xilinx.com/download
[2] Refer to www.xilinx.com/ise/products/memory.htm

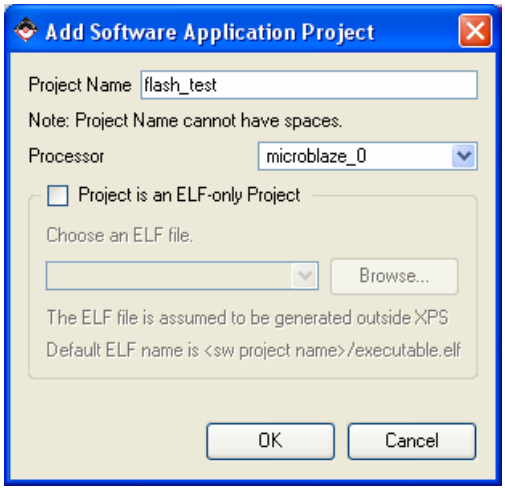
Experiments

The starting point for this tutorial is the project created as part of *Creating Your First MicroBlaze Design with the Spartan-3A Evaluation Kit.* Recall that the hardware platform built during that tutorial has a block diagram as shown below.

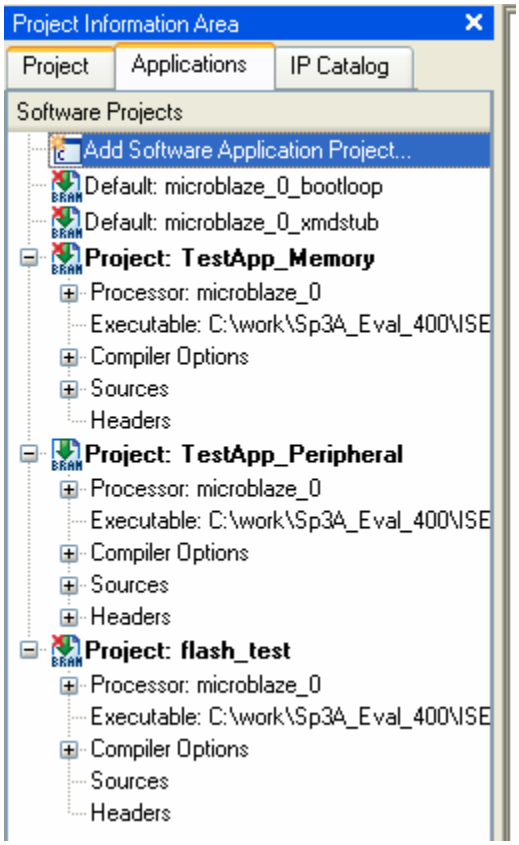


Create a new software application

1. Launch Xilinx Platform Studio (XPS) by selecting **Start → All Programs → Xilinx ISE Design Suite 10.1 → EDK → Xilinx Platform Studio**
2. Open the design at **`Avt3S400A_Eval_MB_parallel_flash_v10_1_00/system.xmp`**
3. Select the *Applications* tab.
4. Double-click on **Add Software Application Project...**
5. Call the new Project Name "flash_test" as shown below. Then click **OK**.



The new software application is now created. See the new entry in the Software Projects area.



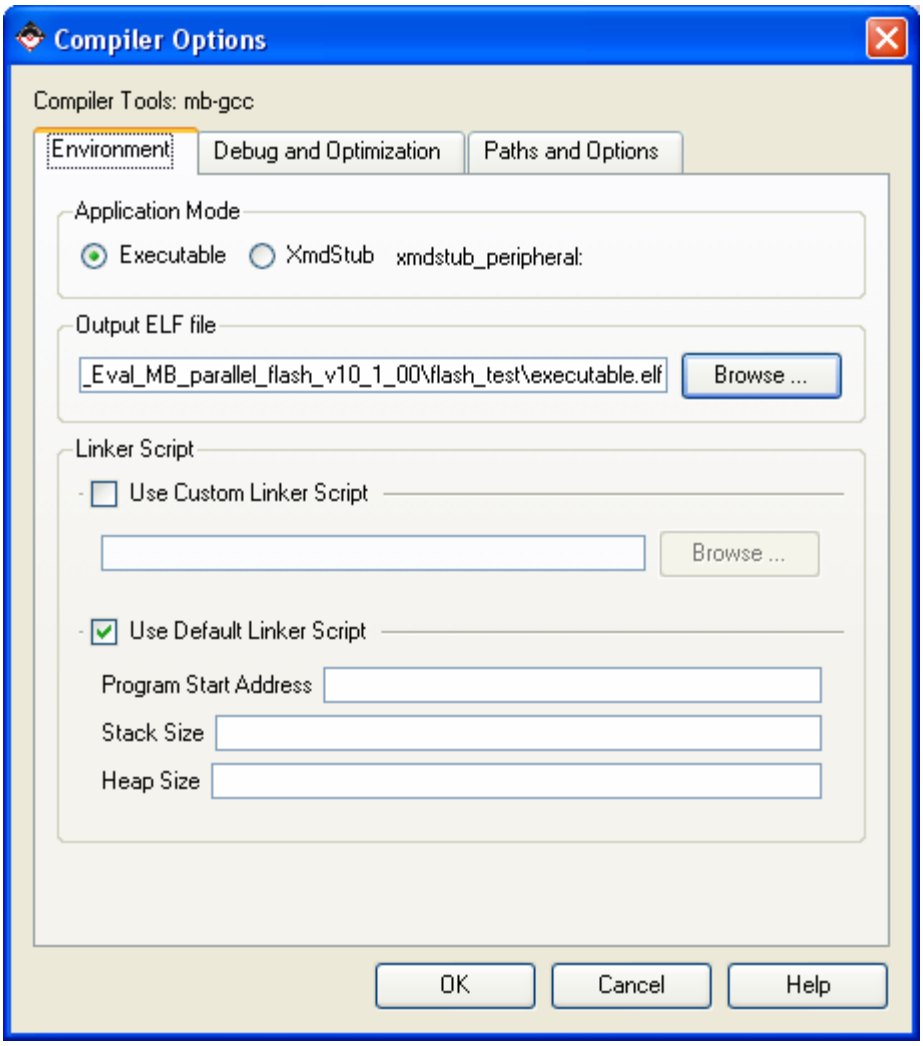
Add code to the application

The code for testing the flash was previously added to the archive.

6. Double-click on **Sources** underneath **Project: flash_test**
7. Browse to the **`code/flash_test`** directory, then select and open **`system.c`**

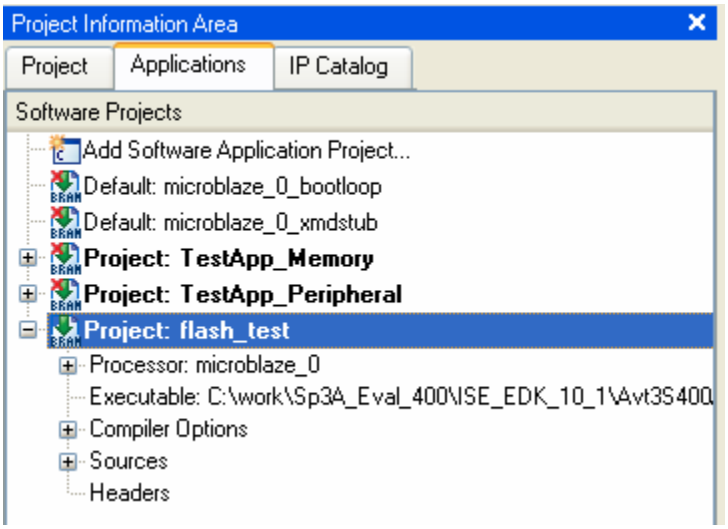
There are no header files for this application. If there were, clicking on **Headers** allows the addition of header files in the same manner as source files.

Also, the default linker script is used, which places this application in BRAM. This is verified by double-clicking on **Project: flash_test** to bring up the Compiler Options for this project.


Compiler Options
Compiler Tools: mb-gcc
Environment
Debug and Optimization
Paths and Options
Application Mode
Executable
XmdStub
xmdstub_peripheral:
Output ELF file
_Eval_MB_parallel_flash_v10_1_00\flash_test\executable.elf
Browse ...
Linker Script
Use Custom Linker Script
Browse ...
Use Default Linker Script
Program Start Address
Stack Size
Heap Size
OK
Cancel
Help

Build the design and initialize the application into BRAM

8. Right-click on **Project: flash_test** and select **Mark to Initialize BRAMs**. The Applications area should now look as follows.



9. Select **Device Configuration → Update Bitstream**. This process should happen very quickly as the project was previously built. This process creates the implementation/download.bit file, which is the hardware bitstream with our software application initialized into BRAM

Run a parallel Flash test

For those using a Xilinx programming cable attached to the target board, the menu option *Device Configuration → Download Bitstream* could be used directly from XPS to download the configuration to the FPGA via JTAG. However, this tutorial makes use of the Avnet AvProg utility and the USB to PSoC interface to configure the FPGA using Slave Serial mode.

10. Set the jumpers on the Avnet Spartan-3A Evaluation board as follows:
    - JP2:USB
    - JP4:M1
    - JP5:SUSPEND OFF
    - JP6
    - JP7:PWR

11. Attach the USB cable to the Avnet Spartan-3A evaluation board and the computer. The D1 LED lights, indicating the board has power.
12. Launch AvProg. To do this, select **Start → All Programs → Avnet → AvProg**. A GUI similar to the following appears. Assuming the Spartan-3A Eval driver was previously installed, AvProg auto-detects the correct COM port for the board (in this case, COM5).

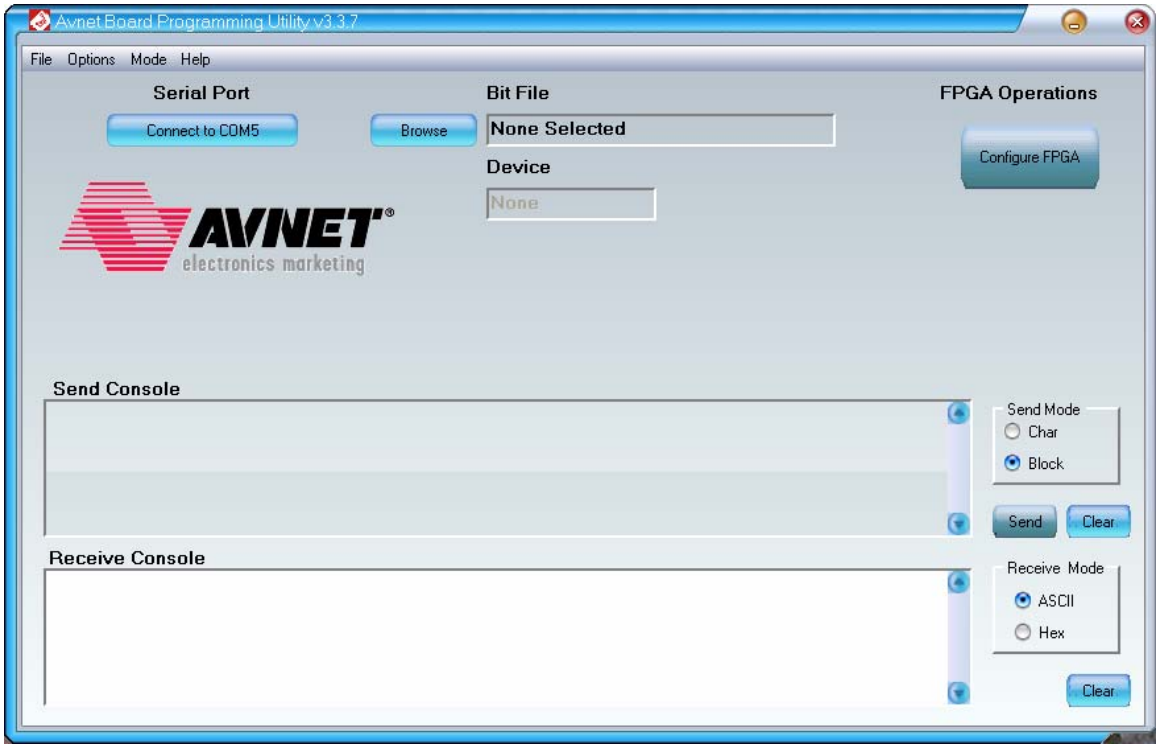


Figure 1 – AvProg

13. To verify the COM port connection settings, select **Options → Comm**. Set the Comm Port to match the Spartan-3A Evaluation board port (in this example, COM5). Select the remaining parameters as shown below, which match the parameters for the UART in the MicroBlaze design. Click **OK** when finished.

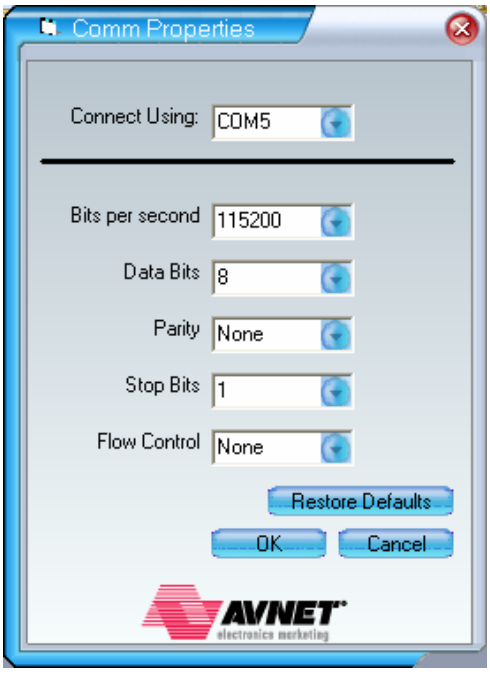
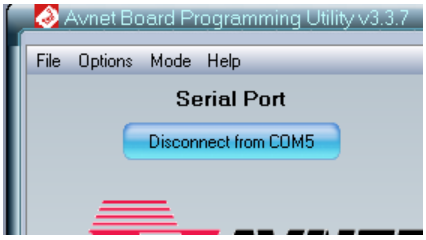
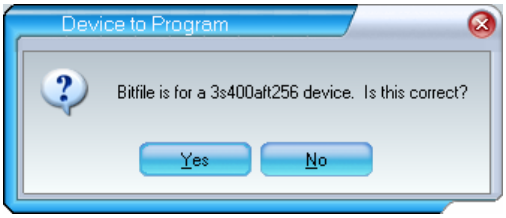


Figure 2 – Comm Properties

14. Click ‘Connect to COMxx’ – if successful the button will change to ‘Disconnect from Comxx’.



15. Browse to the FPGA bitstream just created. This is in the project directory at **`./implementation/download.bit`**.
16. Click **Configure FPGA** to download the bit file for the flash test to the eval board. Click **Yes** to verify the device as a 3s400aft256.



If everything worked correctly, the flash test will begin on the board. This takes a few minutes to complete. If successful, the Receive Console will show the words “Flash Test PASSED!” as shown below.

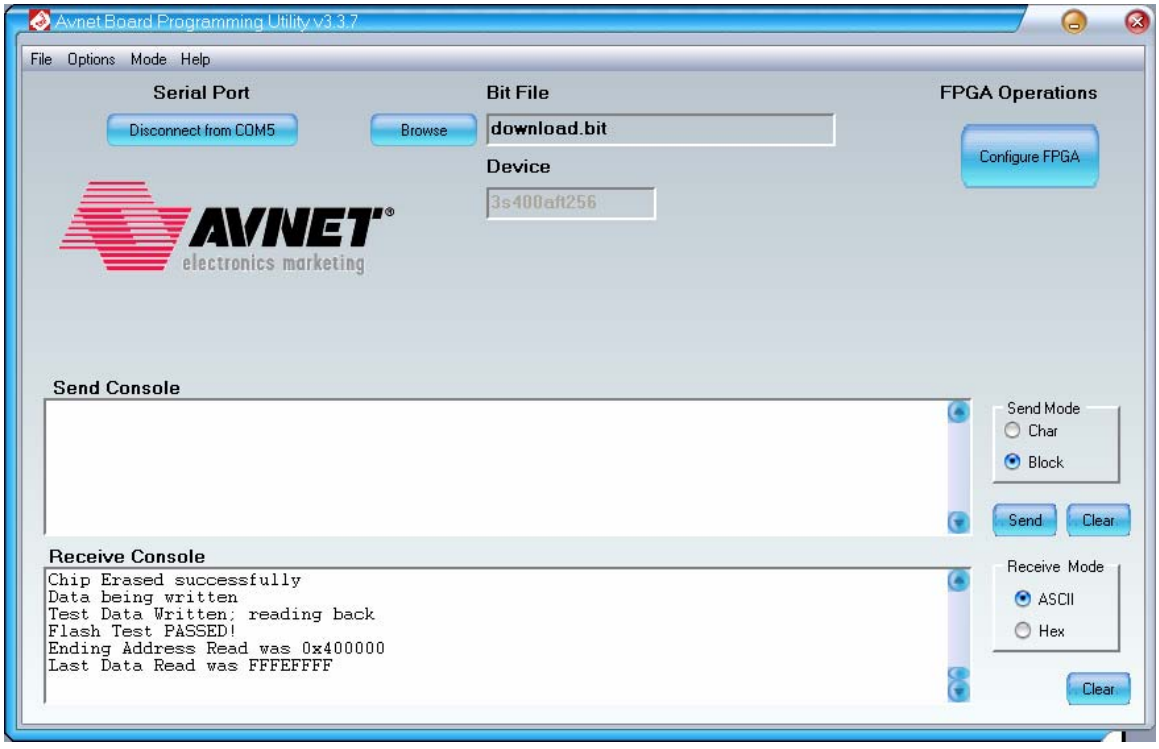


Figure 3 – MicroBlaze TestApp_Peripheral Application Configured to FPGA

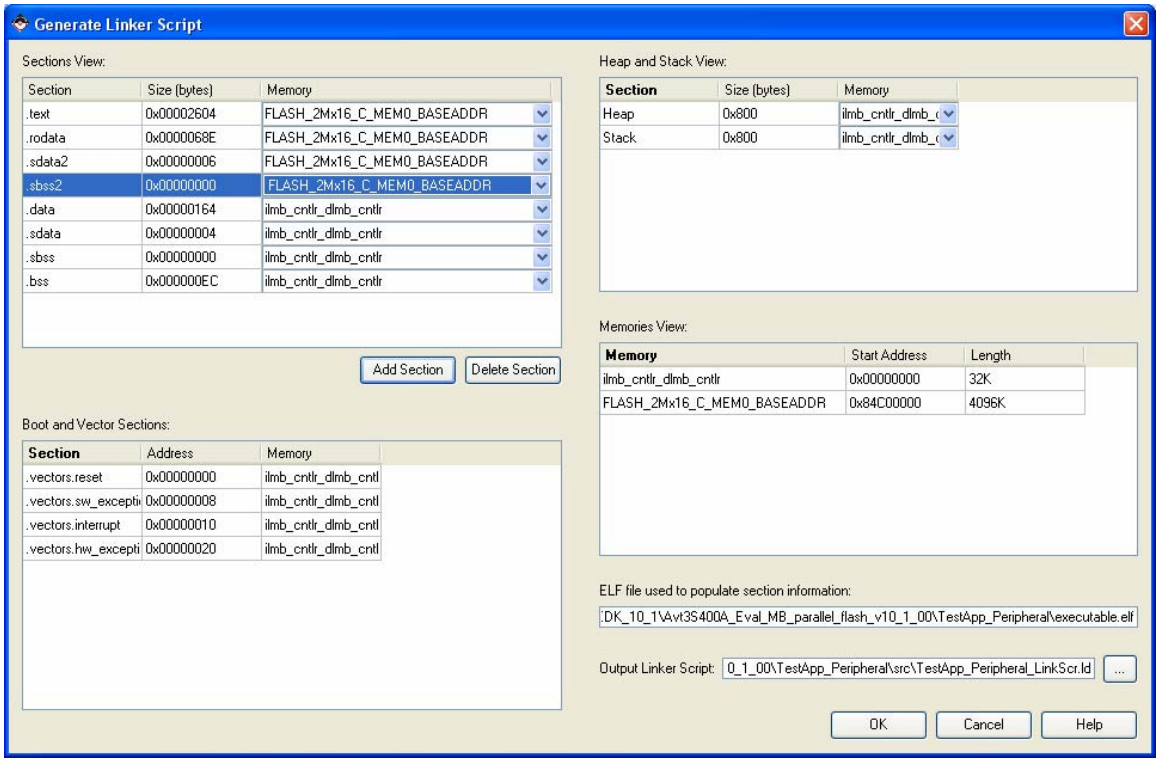
Partition and store a piece of the design in Flash

This flash test and the TestApp_Peripheral application from the original tutorial were both stored and executed entirely from BRAM inside the FPGA. However, some users may prefer to put the read-only code in flash and execute in place. This reduces the demand for BRAM in the system. In this section, the TestApp_Peripheral application is modified to execute partially from flash.

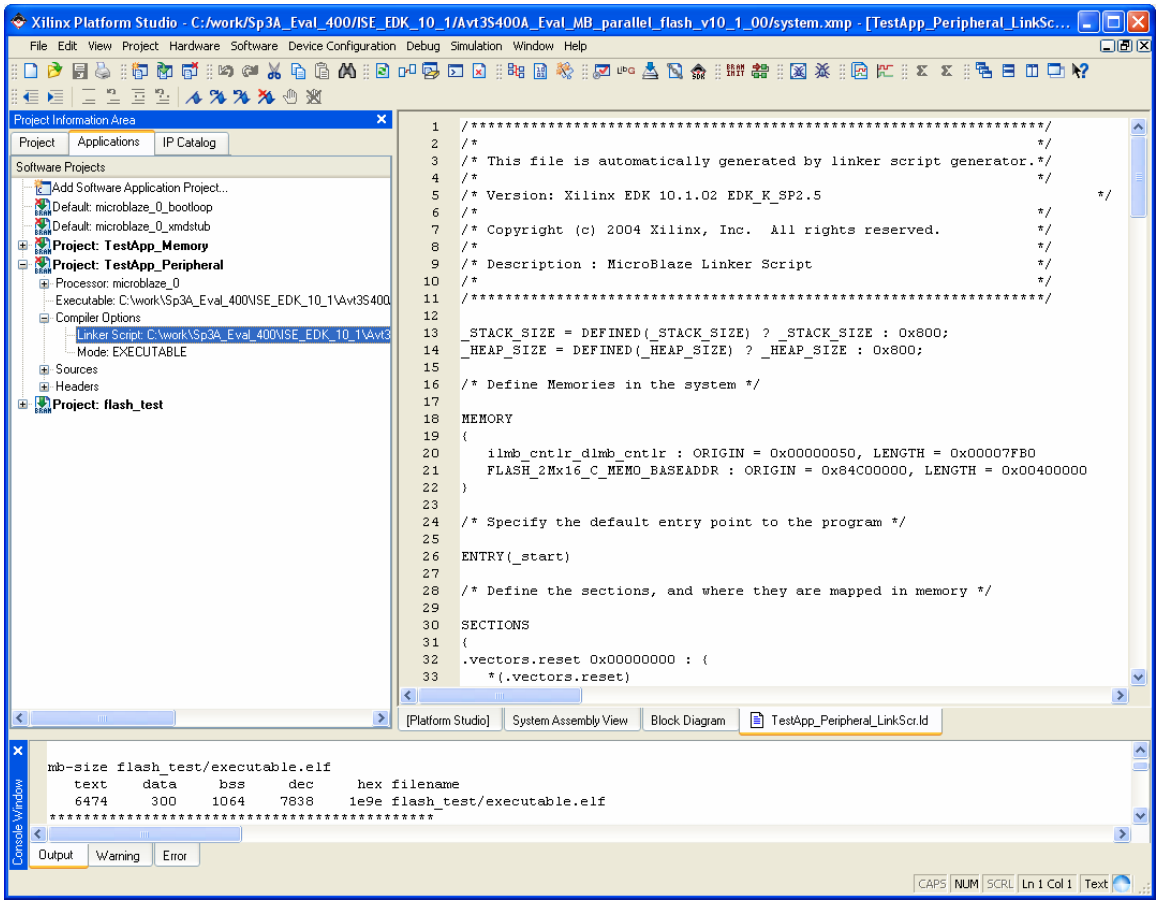
To do this requires a modification to the linker script to partition the software application into two different pieces of memory. The read-only code must be stored in the flash while the writable portions of the application must remain in BRAM.

In addition to storing the MicroBlaze read-only application code in flash, the FPGA configuration image is also stored in flash. The FPGA configuration image must reside at offset 0x0 inside the flash. The uncompressed image is 235,820 bytes, which consumes addresses 0x0 to 0x3992C. The read-only application code could be placed anywhere above this address. For this tutorial, the read-only application code is assigned to offset 0x300000.

17. Select **Software → Generate Linker Script...** then select TestApp_Peripheral then click **OK**.
18. Click on the pull-down menus for the following sections and change to FLASH_2Mx16_C_MEM0_BASEADDR. All of these sections are read-only which is why they can be stored and executed from flash.
    - .text
    - .rodata
    - .sdata2
    - .sbss2



19. Click **OK** when finished. Click **OK** to the warning regarding overwriting the existing linker script.
20. With the *Applications* tab selected, expand **Project:TestApp_Peripheral → Compiler Options**. Double-click on the Linker Script item to open the newly generated linker script in the text editor.



In the new Linker Script, notice that the various sections are now targeted at Flash, as in the .text excerpt shown below (emphasis added):

```
.text : {
   *(.text)
   *(.text.*)
   *(.gnu.linkonce.t.*)
} > FLASH_2Mx16_C_MEM0_BASEADDR
```

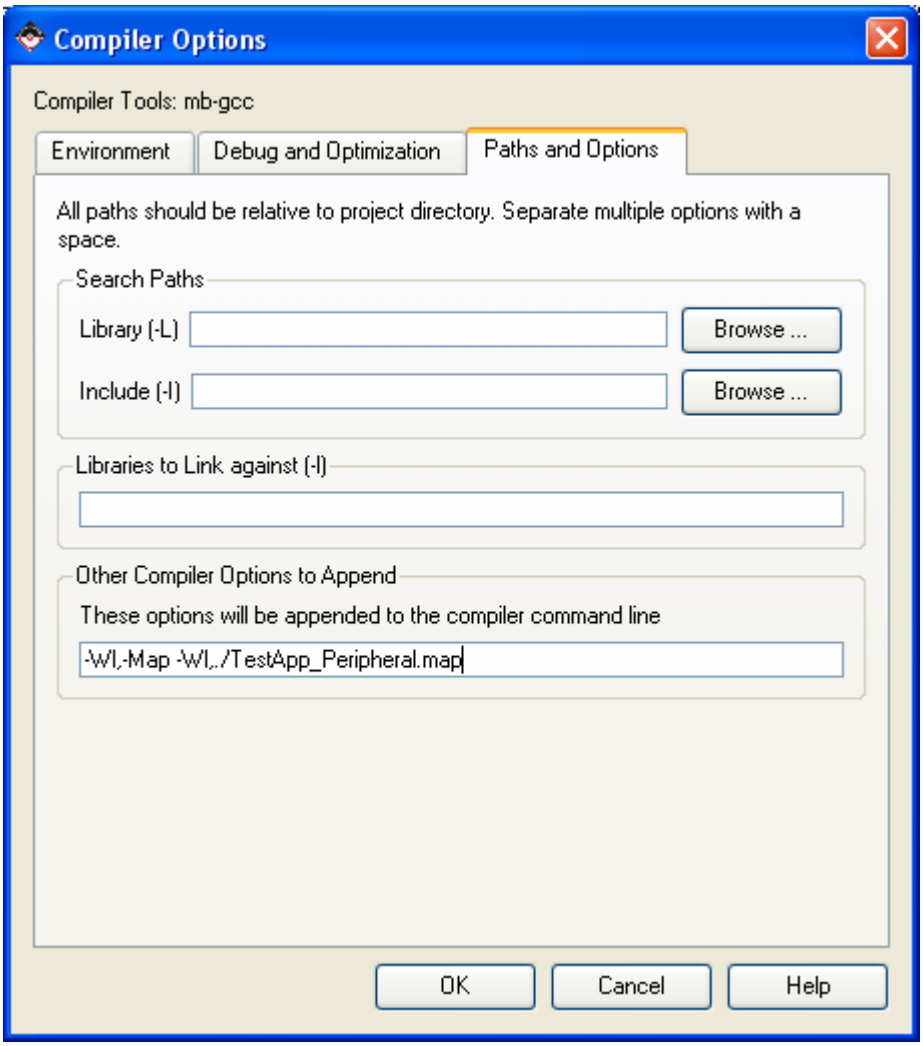
Currently, the Linker Script is set to initialize the read-only sections to offset 0x0 in the Flash. As stated earlier, the configuration image must reside at offset 0x0. The variable in the linker script must be changed to target offset 0x300000 for the read-only code.

21. Modify the FLASH_2Mx16_C_MEM0_BASEADDR variable in the MEMORY section of the Linker script. Increase the ORIGIN by 0x300000 and decrease the LENGTH by 0x300000. The new line should read as follows:

```
FLASH_2Mx16_C_MEM0_BASEADDR : ORIGIN = 0x84F00000, LENGTH = 0x00100000
```

22. Save and close the file.
23. Special linker options are now added to produce a map file of the application. Double-click on **Project: TestApp_Peripheral** to bring up the Compiler Options. Switch to the *Paths and Options* tab. Add the following text into the *Other Compiler Options to Append* field. Click **OK**.

```
-Wl,-Map -Wl,./TestApp_Peripheral.map
```



24. Mark TestApp_Peripheral for BRAM Initialization.
25. Right-click on **Project: TestApp_Peripheral** and select **Build Project**. Once complete, review the `TestApp_Peripheral.map` file in the project directory to verify where each segment was mapped. The ELF file (./TestApp_Peripheral/executable.elf) is now created.
26. The read-only sections must be extracted from the ELF so they can be stored in the Flash. The mb-objcopy command is used to do this. Launch an EDK shell be selecting **Project → Launch EDK Shell**. A script file is provided to simplify entering the command. In the EDK Shell, type "./extract_ro.sh" and hit <Enter>. This script file

produces the flash.bin file in the TestApp_Peripheral directory. The command it executes extracts all the read-only sections. Type 'exit' to close the shell.

```
mb-objcopy -O binary \
-j .text \
-j .init \
-j .fini \
-j .rodata \
-j .sdata2 \
-j .sbss2 \
./TestApp_Peripheral/executable.elf \
./TestApp_Peripheral/flash.bin
```

27. Next the hardware bitstream is initialized with the read-write data in the ELF file. Select **Device Configuration → Update Bitstream** to create a new download.bit file. During this process, the tools extract the read-write sections from the ELF file and use them to create initialization vectors for the FPGA BRAM. The hardware-only system.bit bitstream is then merged with the BRAM initialization vectors to create download.bit, which contains both the hardware platform and read-write application. The tools will analyze the ELF file and discover that the program is not completely self-contained within the BRAM address space and produce the following expected warning.

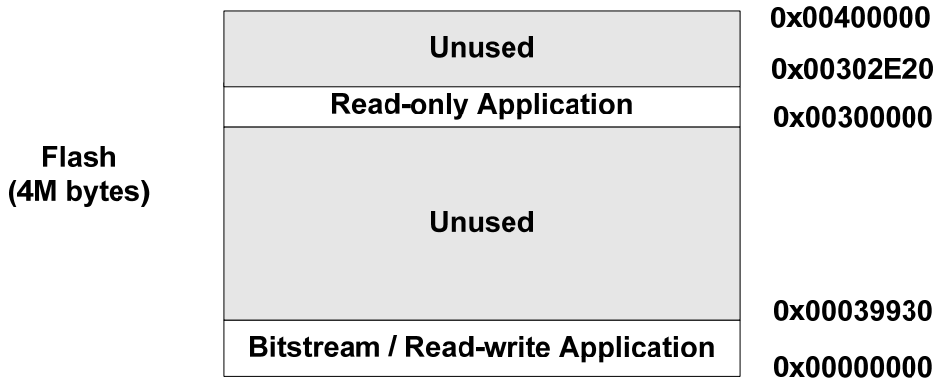
```
WARNING:MDT - Elf file TestApp_Peripheral/executable.elf does not reside
   completely within BRAM memory of processor microblaze_0.
```

28. The download.bit file must be converted to a format compatible with the Flash. In Windows Explorer, browse to the ./FLASH_BURN directory. Double-click on the `make_bpi_image_x16.bat` batch file. This converts the download.bit file into a BPI-compatible binary. Press any key to close the window.

The Flash is now programmed in two separate steps. Two pieces have been created:

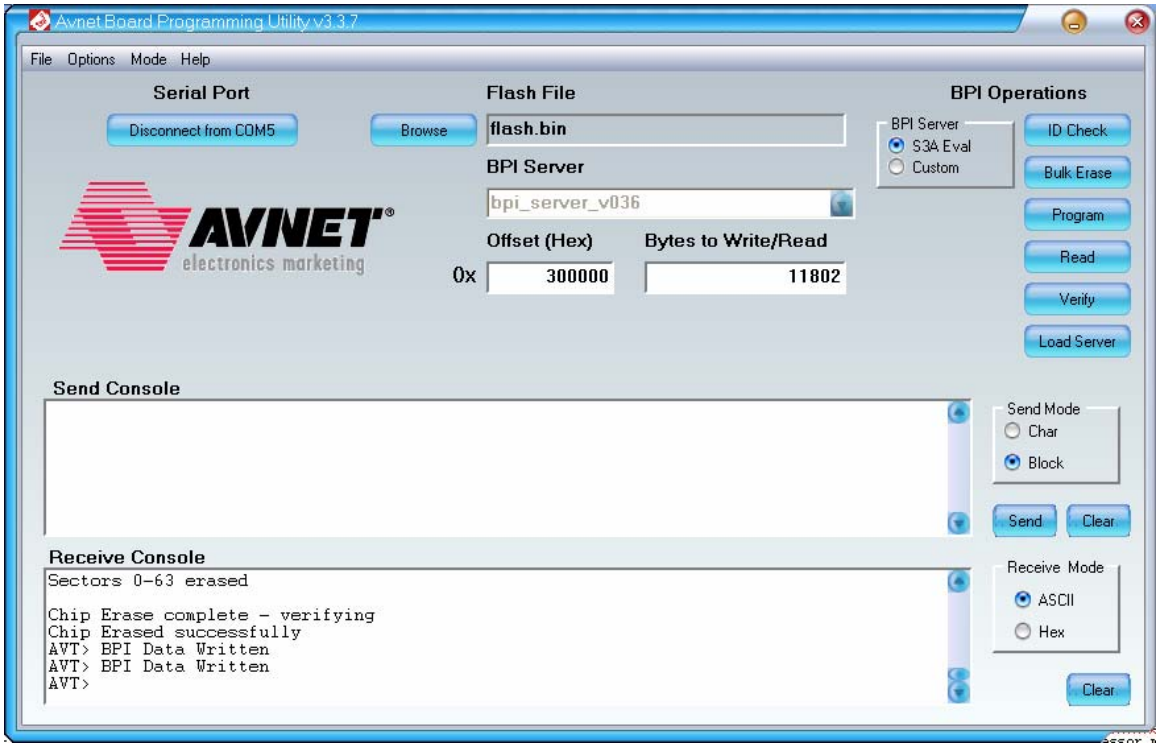
- periph_test_xipFlash.bin – this is download.bit converted to a Flash compatible file. This is a bitstream with the read-write application initialized into the BRAM vectors.
- flash.bin – this is the read-only application

The bitstream must reside at offset 0x0. The read-only application location is chosen to be 0x300000. A mapping of the flash contents is shown in the diagram below.



The following steps outline how to program these two pieces to the Flash.

29. If not already done, launch AvProg and connect to the board.
30. Select **Mode → Program Parallel Flash**.
31. Click the button **Load Server**. Click **Yes** to verify the 3s400 device.
32. Click the button **Bulk Erase**.
33. Click the **Browse** button. Browse and open `./FLASH_BURN/periph_test_xipFlash.bin`. Leave the Offset as 0. Click **Program**. Click **Yes** to confirm that the sections are erased.
34. Click the **Browse** button again. Browse and open `./TestApp_Peripheral/flash.bin`. Change the offset to 0x300000. Click **Program**. Click **Yes** to confirm that the sections are erased.



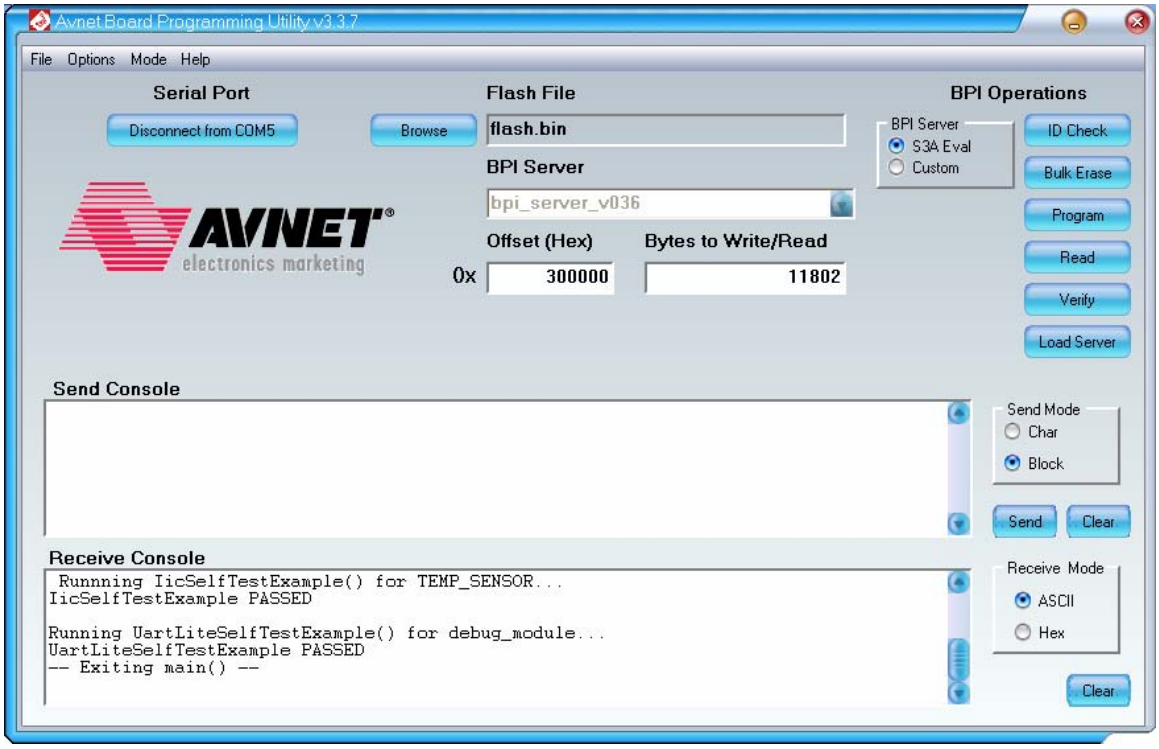
Leave AvProg open and the board powered on to maintain the USB connection.

Run a design while executing the read-only portion of the application in Flash

Now that the bitstream (with read-write application segments initialized to BRAM) and read-only code are programmed into flash, the TestApp_Peripheral application can be exercised.

35. To enable BPI configuration, change the JP4 MODE jumpers to M0 and M2.
36. In AvProg, click the **Clear** button on the Receive Console.
37. Press the PROG button (SW1) on the board to initiate a reconfiguration from BPI.

The FPGA configures from the parallel Flash. MicroBlaze executes the TestApp_Peripheral application with read-write application sections in BRAM and read-only application sections at offset 0x300000 in the Flash. See the results of the program in the Receive Console of AvProg.



Congratulations! You have completed this tutorial exercising the parallel Flash with MicroBlaze.

Additional Support

For additional support, please review the discussions and post your questions to the Spartan-3A Evaluation Forum at http://groups.google.com/group/avnet-spartan-3a-eval-kit.

You can also contact your local Avnet/Silica FAE.