# Memec 3SMB MicroBlaze Using Memory Reference Designs

July 7, 2006
Version 8.1

## Overview

This document describes several reference designs using the Xilinx MicroBlaze™ soft processor core running at 50 MHz on the Memec Spartan-3 MB (3SMB) development board. The 3SMB development board is designed with DDR SDRAM and Flash to give the MicroBlaze access to memory and non-volatile storage.

## Experiment Setup

### Software

The recommended software setup for this reference design is:

- Windows2000 or WindowsXP
- Xilinx ISE 8.1i (Foundation or BaseX) with latest Service Pack[1]
- Xilinx EDK 8.1 with latest Service Pack[1]

### Hardware

The hardware setup used by this reference design includes:

- Computer with a recommended minimum of 1GB RAM and 1 GB Virtual Memory[2]
- Memec Spartan-3 MB Development Kit
- Platform USB Cable or JTAG Programming Cable IV
- Serial Cable

### MicroBlaze Platform

The design and optimization of the MicroBlaze platform used in this reference design is described in *Building and Optimizing a MicroBlaze BSB System for Spartan-3 MB*. Please refer to that document for a detailed description of how to create the platform. The hardware platform in this design was built based on that tutorial.

In addition to the two applications automatically generated by Base System Builder (BSB), the software platform has had four software applications added to it:

- Running from DDR with no linker
- Running from DDR with a linker
- Testing the Flash
- WebServer

[1] Latest Service Packs are available at www.support.xilinx.com/swupdate
[2] Refer to the *ISE 8.1i Release Notes and Installation Guide* http://toolbox.xilinx.com/docsan/xilinx8/books/docs/irn/irn.pdf

The DDR and Flash applications are described in this document. Additionally, this document also describes how to create a bootloader. The WebServer application is described in a separate document.

Additionally, the XilNet and XilMFS libraries have been added to the software platform for the WebServer.

## Understanding the Hardware Platform

A block diagram of the hardware platform is shown in Figure 1. Besides the MicroBlaze processor and busses, the design consists of 16 KB local memory, 32 MB DDR, 4 MB Flash, Ethernet MAC, timer, interrupt controller, two UARTs, a debug peripheral (MDM) and multiple GPIOs.

An asynchronous DDR memory controller is used in this design. This allows the MicroBlaze system to run at a lower speed of 50 MHz, which is more reasonable for Spartan-3, while the DDR is running at 75 MHz, which is the minimum required frequency for the Micron DDR. The on-chip Digital Clock Manager (DCM) is used to create the various clock frequencies and phases required to make this system work, all based on the 75 MHz oscillator on the 3SMB board.

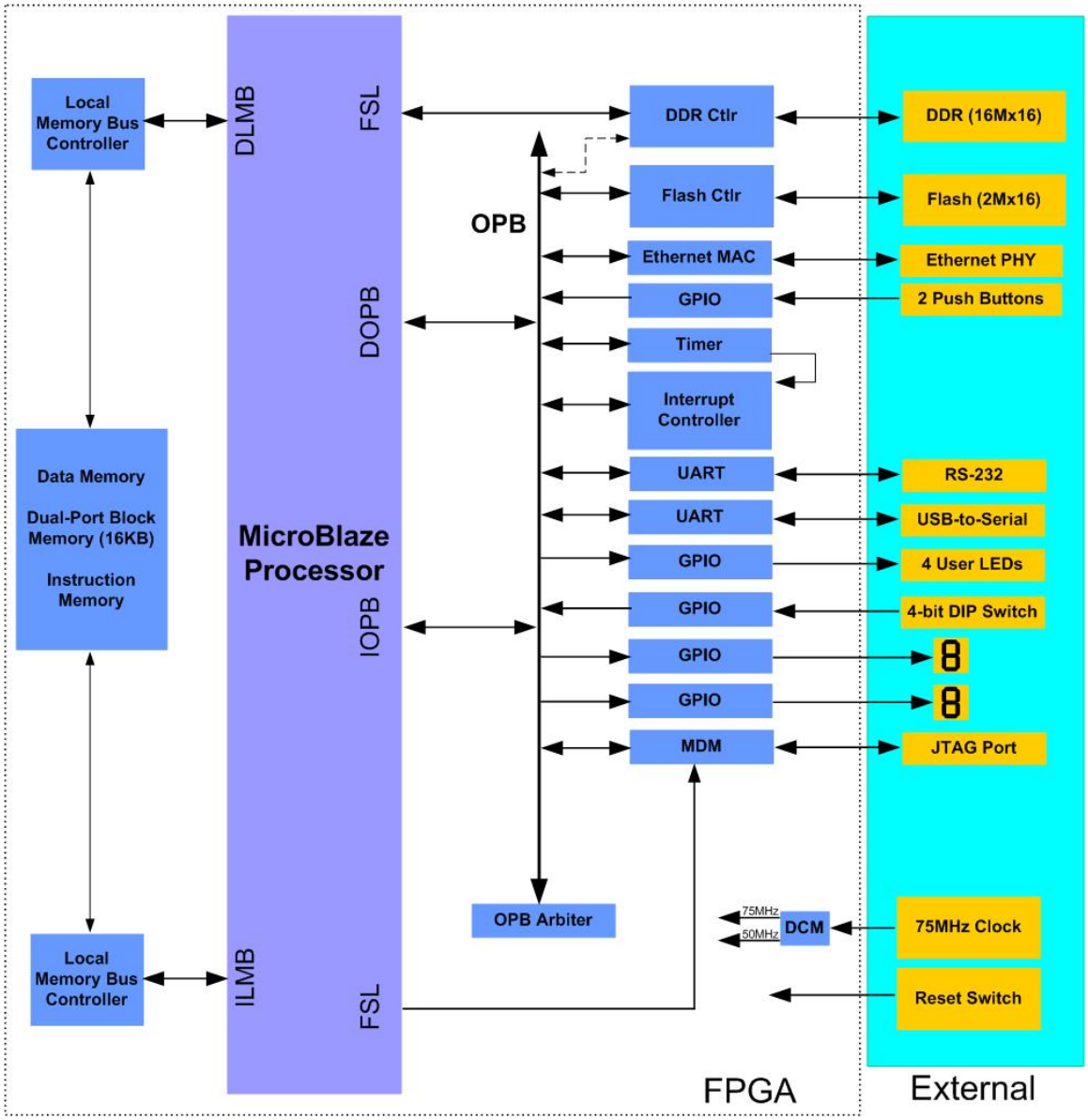


**Figure 1 – Block Diagram of BSB-Generated System**

*Note on licensing*: The 10/100 Ethernet MAC OPB peripheral is not a free core. An evaluation license is required to use the core. The evaluation version of the core includes built-in timeout circuitry that will disable the core after a period a time. The evaluation version of the core is included in EDK. This example will work for some time with the default EDK installation. However, those intending to use the core in production should contact their local Avnet distributor to purchase a full license for the core.

### Memory Map

The memory map is auto-generated by BSB. This memory map can be edited manually, re-generated, or simply viewed by selecting the *Addresses* radio button filter in the System Assembly view. The memory map for this system is shown in Figure 2.

Filters: Bus Interface | Ports | Addresses — Generate Addresses

| Instance | Base Address | High Address | Size |
|---|---|---|---|
| mb_opb | | | U |
| Ethernet_MAC | 0x40c00000 | 0x40c0ffff | 64K |
| dlmb_cntlr | 0x00000000 | 0x00003fff | 16K |
| ilmb_cntlr | 0x00000000 | 0x00003fff | 16K |
| LEDs_4Bit | 0x40000000 | 0x4000ffff | 64K |
| LED_7Segment1 | 0x40020000 | 0x4002ffff | 64K |
| LED_7Segment2 | 0x40040000 | 0x4004ffff | 64K |
| Push_Buttons_2Bit | 0x40060000 | 0x4006ffff | 64K |
| DIP_Switches_8Bit | 0x40080000 | 0x4008ffff | 64K |
| RS232 | 0x40600000 | 0x4060ffff | 64K |
| USB_UART | 0x40620000 | 0x4062ffff | 64K |
| opb_intc_0 | 0x41200000 | 0x4120ffff | 64K |
| debug_module | 0x41400000 | 0x4140ffff | 64K |
| opb_timer_1 | 0x41c00000 | 0x41c0ffff | 64K |
| FLASH_2Mx16 | 0x21000000 | 0x213fffff | 4M |
| DDR_SDRAM_16Mx16 | 0x22000000 | 0x23ffffff | 32M |

**Figure 2 – BSB-generated Memory Map**

## 3S1500MB Board Setup

The Memec Spartan-3 MB board should be configured as follows:

1. Uninstall all MODE jumper on J1.
2. Install a jumper on JP9 in the “BOARD” position (pins 1-2).
3. Install a jumper on JP10, pins 1-3.
4. Install two jumpers on JP8.
5. Install a jumper on JP7 in the “PROM ENABLE” position (pins 1-2).
6. Install a jumper on JP6 in the 3.3V position (pins 1-2).
7. No other jumpers should be installed.
8. Connect a straight through RS232 cable to the board DB-9 connector (JD1) and the serial port of the PC.
9. Connect an ethernet cable to the board (JM1) and the PC (required for the Peripheral Test to work – not required for other demonstrations).
10. Verify the Power switch (SW1) is in the OFF position.
11. Connect the AC/DC adapter to JP1.

Refer to Figure 3 for jumper locations.

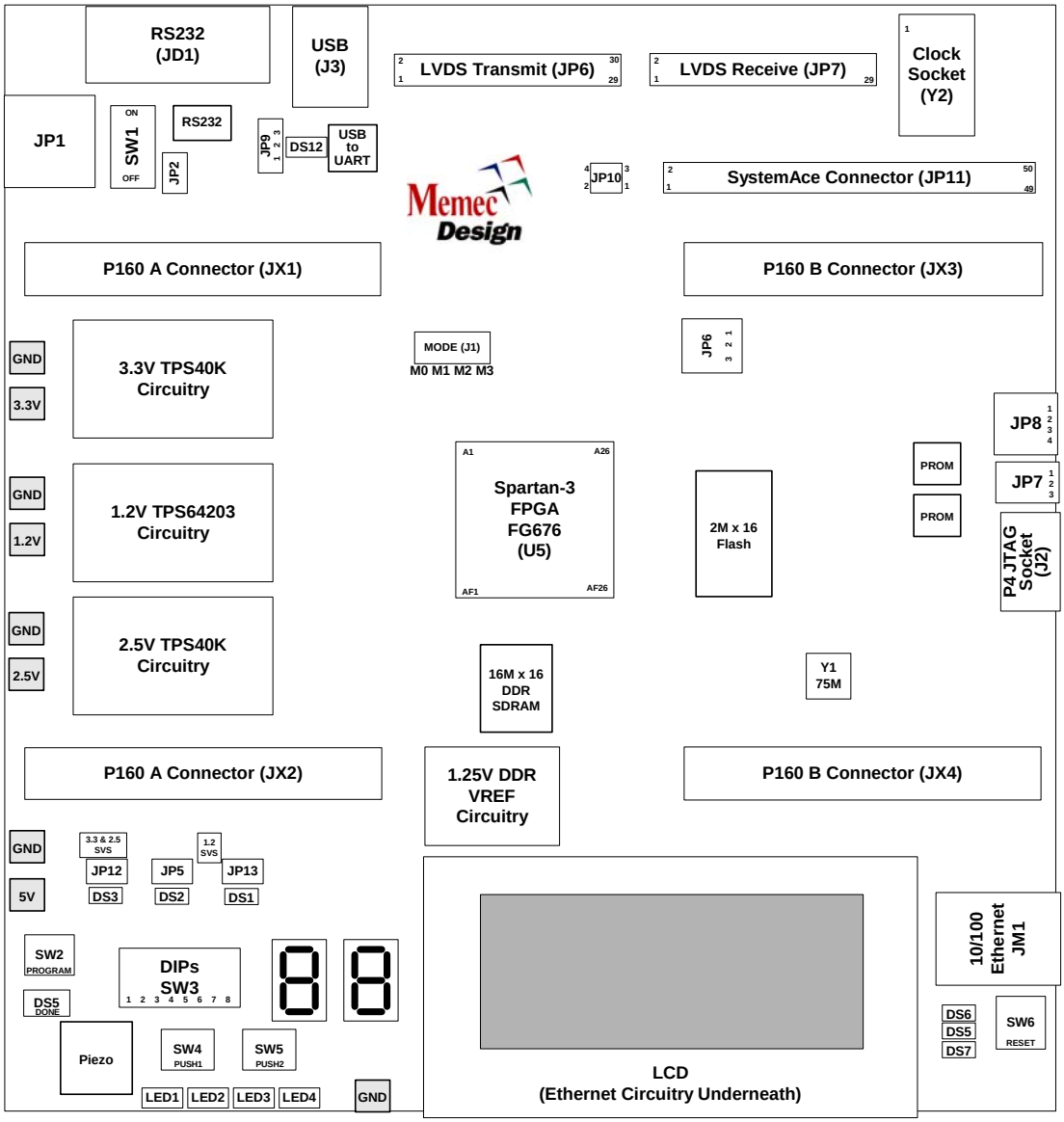


**Figure 3 – Memec Spartan-3 MB Jumper Locations**

# Experiments

Four different memory experiments are included. The first one uses the BSB-generated Test Application to test the DDR memory. The second one runs a simple program from DDR using a compiler setting. The third one runs the same simple program from DDR using a linker script. The fourth application tests the flash. All examples use a 115200 HyperTerminal for RS232 UART output. Refer to Appendix A: Starting a HyperTerminal Session for instructions on starting the HyperTerminal, or simply click the shortcut **com1_115200_8n1n.ht** provided with the reference designs.

## Test DDR

The BSB-generated Memory Test Application tests all memory interfaces which are not running code. In this case, the DDR will be tested since the code has been targeted to run from BRAM. The UART is used to print status messages via stdin/stdout.

1. Turn power on to the board.
2. Open HyperTerminal.
3. Launch XPS.
4. Open the **system.xmp** project located at `.\Memec_3SMB_MicroBlaze_Base\3SMB_BSB_EDK81`
5. The source code for this design is located in the **TestApp_Memory** project under the *Applications* tab. This project should be marked to have the compiled code initialized to BRAM since the **BRAM Initialization** icon in front of the project name, , does not have a red X through it. This software project should be the only one marked this way, as shown in Figure 4.

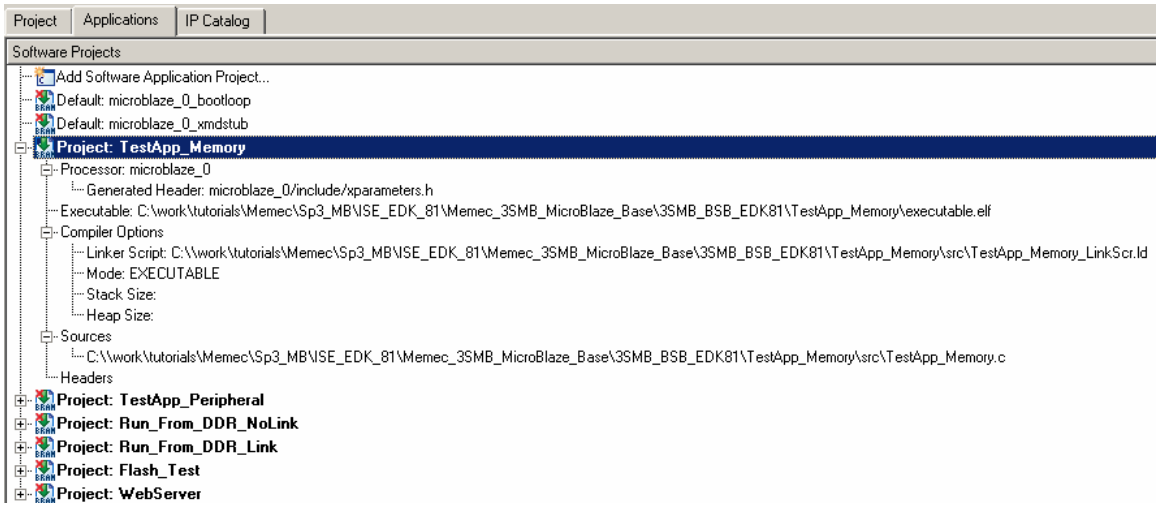


**Figure 4 – TestApp_Memory Software Project Marked for BRAM Initialization**

6. Run **Device Configuration → Download Bitstream**. This first time compiling the project will take 15 minutes or more since the hardware platform must be built. Additionally, the libraries and board support package (BSP) is generated. The TestApp code is then compiled since it is being initialized to BRAM. Lastly, XPS downloads the bitstream with the code embedded into the BRAM.

A passing test will do the following:

- Print a short statement to the HyperTerminal
- Test the DDR in 32-, 16-, and 8-bit modes
- Print another short statement to the HyperTerminal

Successful test results are shown in the HyperTerminal as seen in Figure 5.

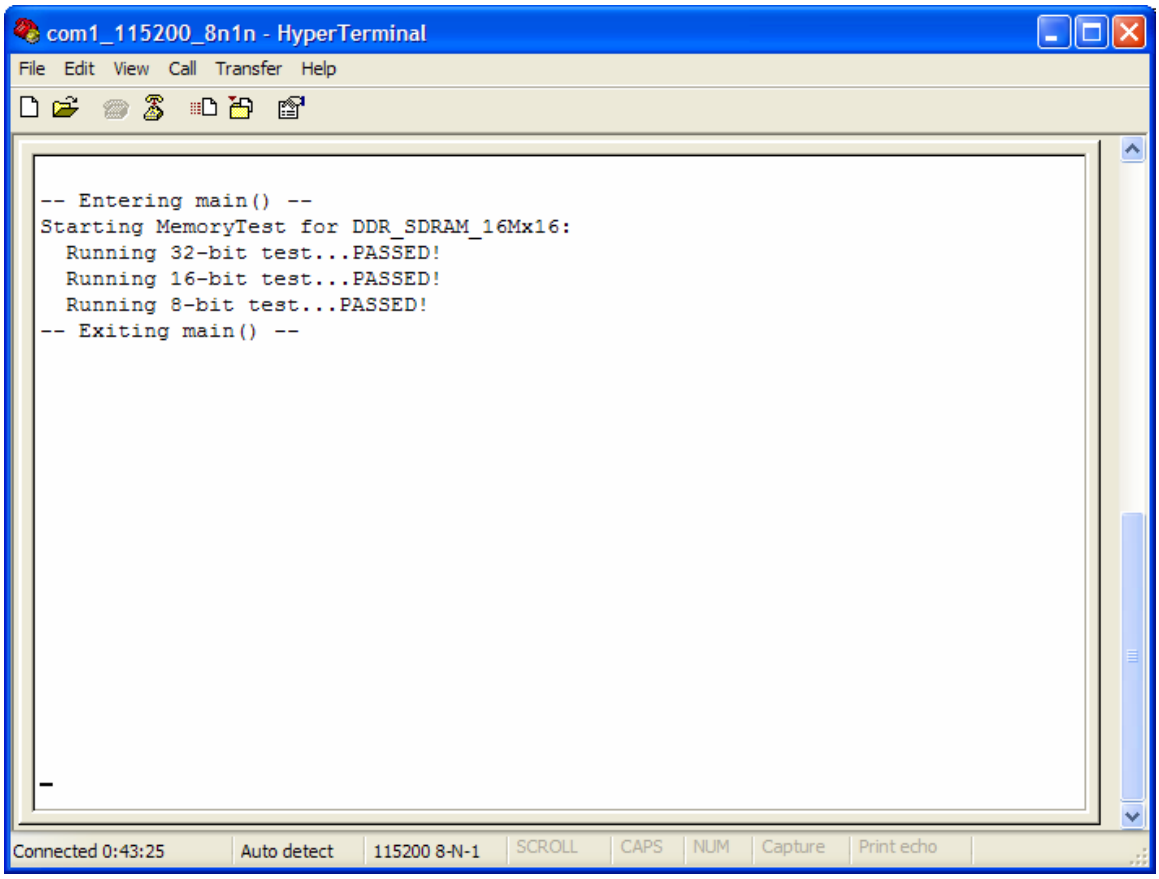


**Figure 5 – DDR Test Passed**

## Running from DDR – No Linker

This program writes a welcome banner to the UART and loops a pattern to the user LEDs. The program runs from DDR based on a compiler setting (not a user linker script), using XMD. (Note that a linker script is always used. If a user does not specify a linker script, a built-in, default EDK linker script is used.) This project uses the MicroBlaze Debug Module (MDM) to connect to XMD for downloading the code to DDR.

1. HyperTerminal and XPS should still be open from the **TestApp_Memory** experiment.
2. Under the *Applications*, right-click on **Project: TestApp_Memory** and uncheck the **Mark to Initialize BRAMs** line.
3. Since this experiment runs the code from DDR, the bootloop is marked for BRAM initialization. This guarantees to keep the processor in a good state waiting for the actual code to be downloaded from XMD. Right-click on **Default: microblaze_0_bootloop** and check the **Mark to Initialize BRAMs** line
4. Open the compiler options by right-clicking on **Project: Run_From_DDR_NoLink** and selecting **Set Compiler Options**. Note the

starting address points to the first location in DDR as shown in Figure 6. Click OK to close the window.

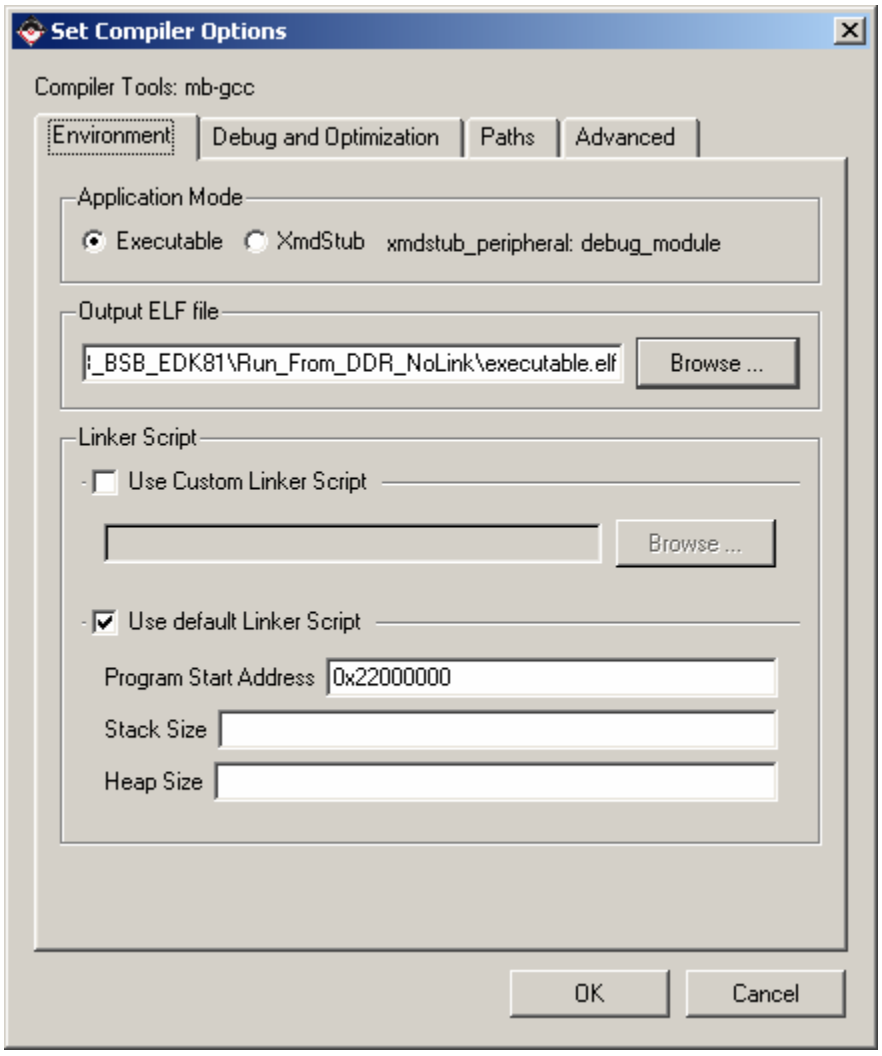


**Figure 6 – Program Start Address Points to DDR**

5. Run **Device Configuration → Download Bitstream**. The DONE LED should light on the 3SMB board. The LEDs and 7-segment displays will also light since the default state for these I/Os is in the ON state. The processor at this point is looping harmlessly, essentially doing nothing.
6. Right-click on **Project: Run_From_DDR_NoLink** and select **Build Project**.
7. Launch XMD: **Debug → Launch XMD**. If not done previously, a dialog box will remind you to set the XMD options. Select OK, then **Save** the default options. After saving, XMD will automatically launch and connect to the MicroBlaze processor.

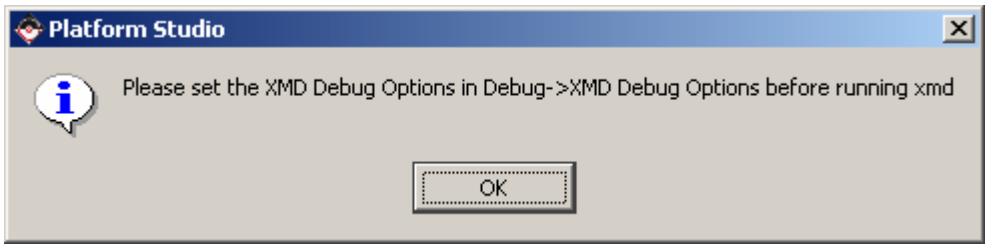


**Figure 7 – XMD Debug Options Must Be Set**

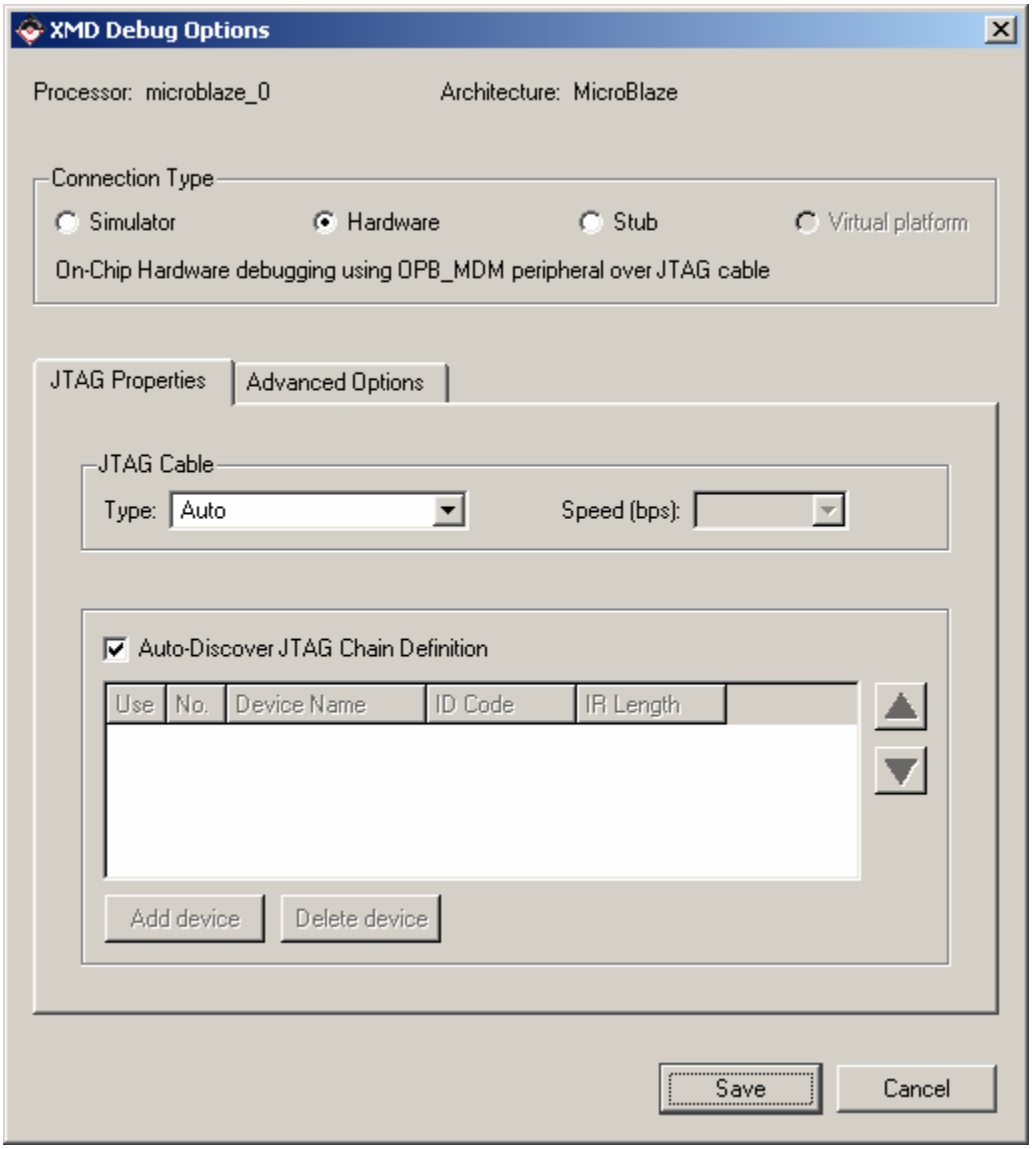


**Figure 8 – Default XMD Debug Options**

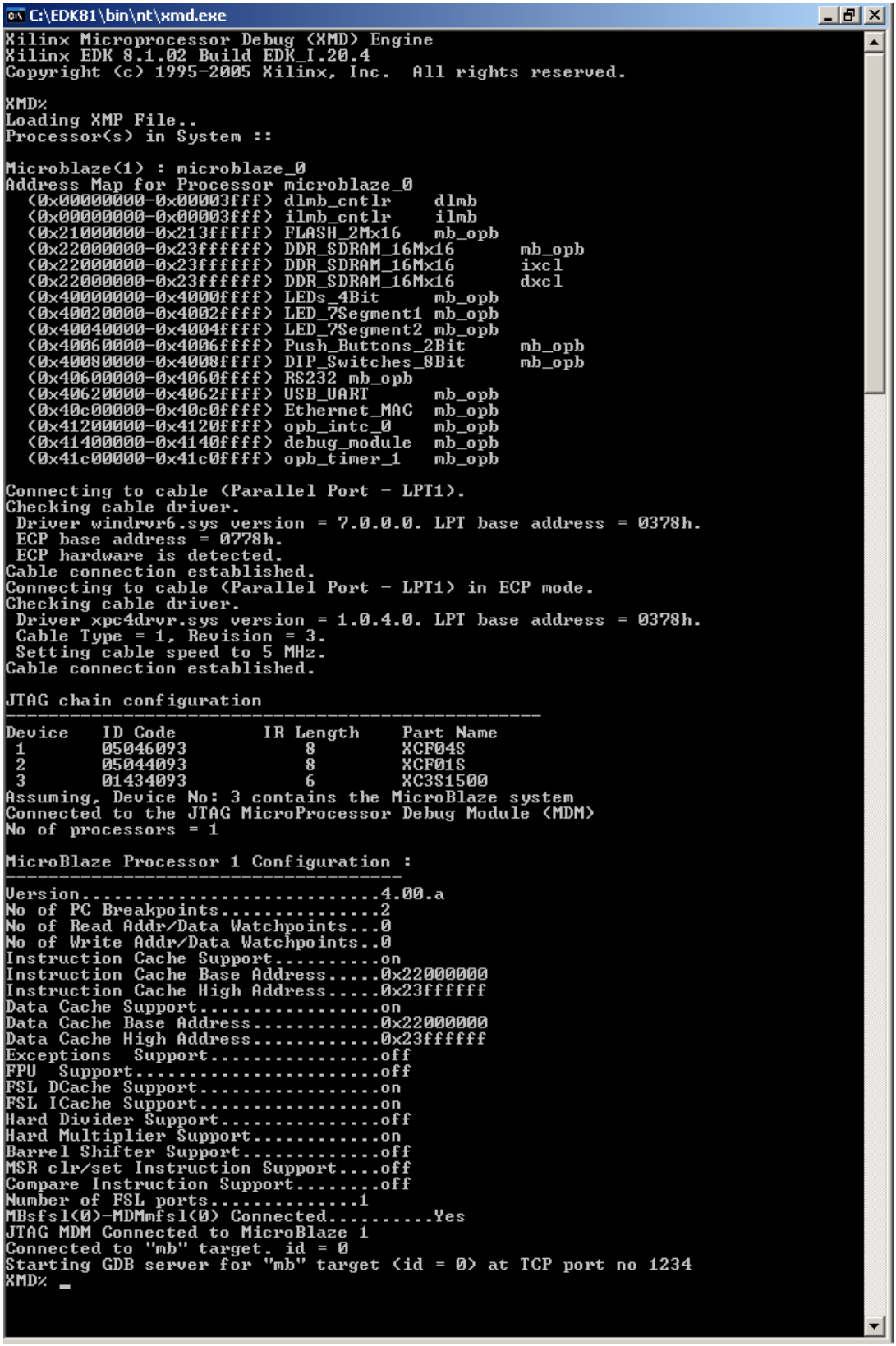


**Figure 9 – XMD Launched and Connected**

8. In XMD, type the following commands

```
dow Run_From_DDR_NoLink/executable.elf
con
```

The LEDs on the board should start flashing, and the HyperTerminal output is shown in Figure 10. Do not close the XMD window at this time as it will be used in the next experiment.

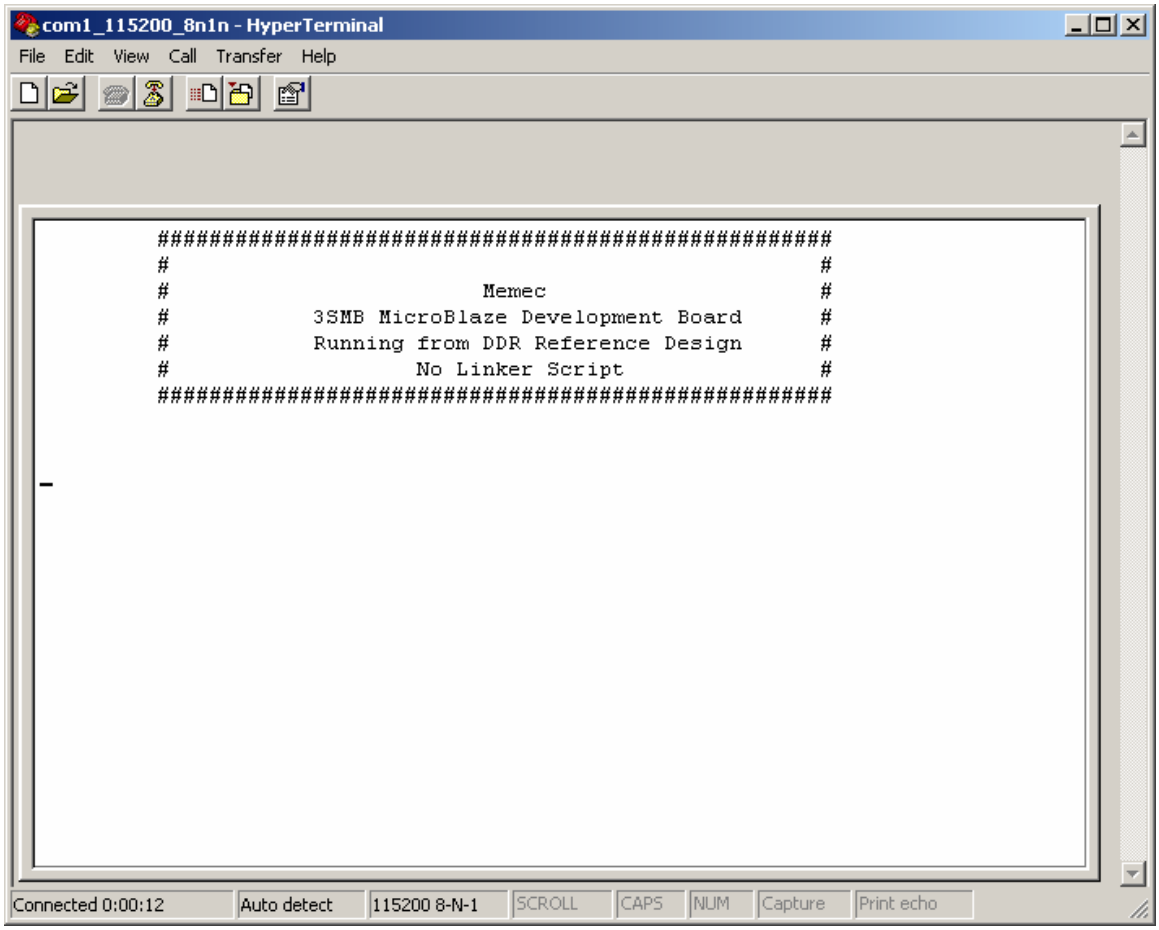


**Figure 10 – Running from DDR using Compiler Options**

As an interesting experiment, try commenting out the lines that enable the cache and re-run the experiment. Notice the difference in LED blink speed, which is attributable to the faster code operation with cache.

## Running from DDR using a Linker Script

The software for this experiment is identical to the previous experiment except a linker script is used rather than a compiler option to place the program in DDR.

1. Open the *Set Compiler Options* window for **Project: Run_From_DDR_Link**. Under the *Environment* tab, notice that the checkbox for *Use Custom Linker Script* is checked and a linker script located in the code directory has been specified. Close the Option window.
2. Expand the browse view for **Project: Run_From_DDR_Link**. Under *Compiler Options*, double-click on the linker script entry to open the linker script in the editor. After viewing the linker script, close it.
3. Right-click on **Project: Run_From_DDR_Link** and select **Build Project**.
4. In the XMD window, type "`stop`". The LEDs stop flashing.
5. Now the code linked with the user linker script will be downloaded to the DDR. Type the following commands into XMD:

```
dow Run_From_DDR_Link/executable.elf
con
```

6. The LEDs on the board should start flashing as before. However, the text displayed in the HyperTerminal is slightly different as shown in Figure 10.
7. Close the XMD window by typing "`exit`" and <return>.

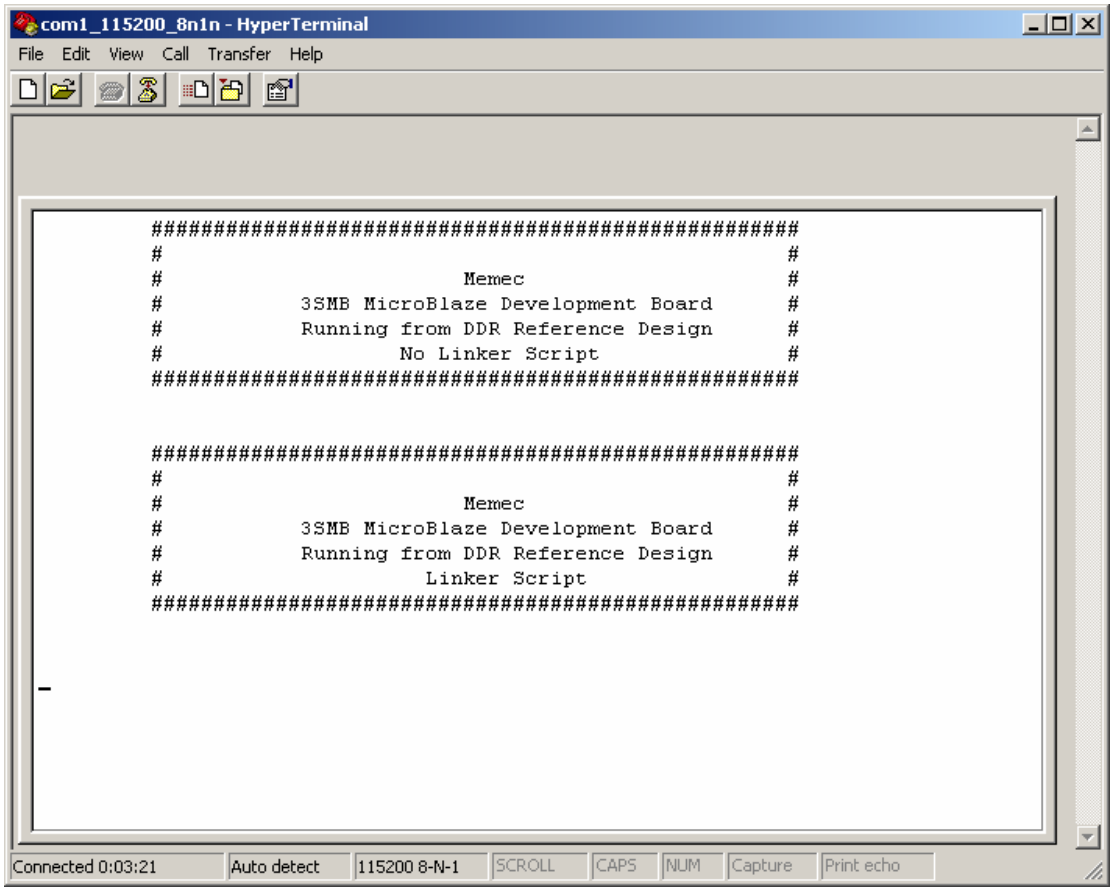


**Figure 11 – Running from DDR using Linker Script**

## Flash Test

The next design tests the flash. First, the manufacturer and device ID codes are read and verified. Next, the entire flash device will be erased. Then, a simple test pattern is written to the flash. Lastly, the flash is read and compared with the original pattern.

1. Unmark **Default: microblaze_0_bootloop** for BRAM initialization.
2. Mark **Project: Flash_Test** for BRAM initialization.

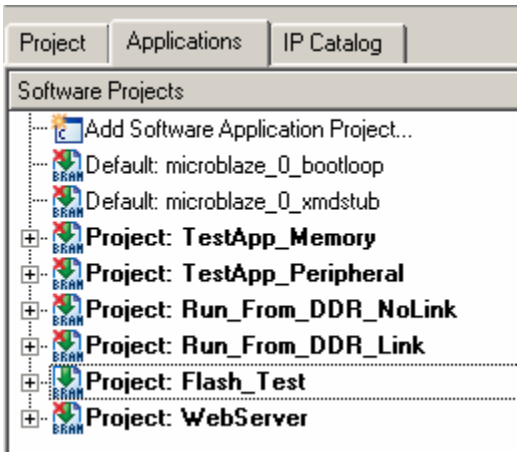


**Figure 12 – Flash_Test Marked for BRAM Init**

3. Build, download, and run the test by selecting **Device Configuration → Download Bitstream**. The results for this test are shown in Figure 13.

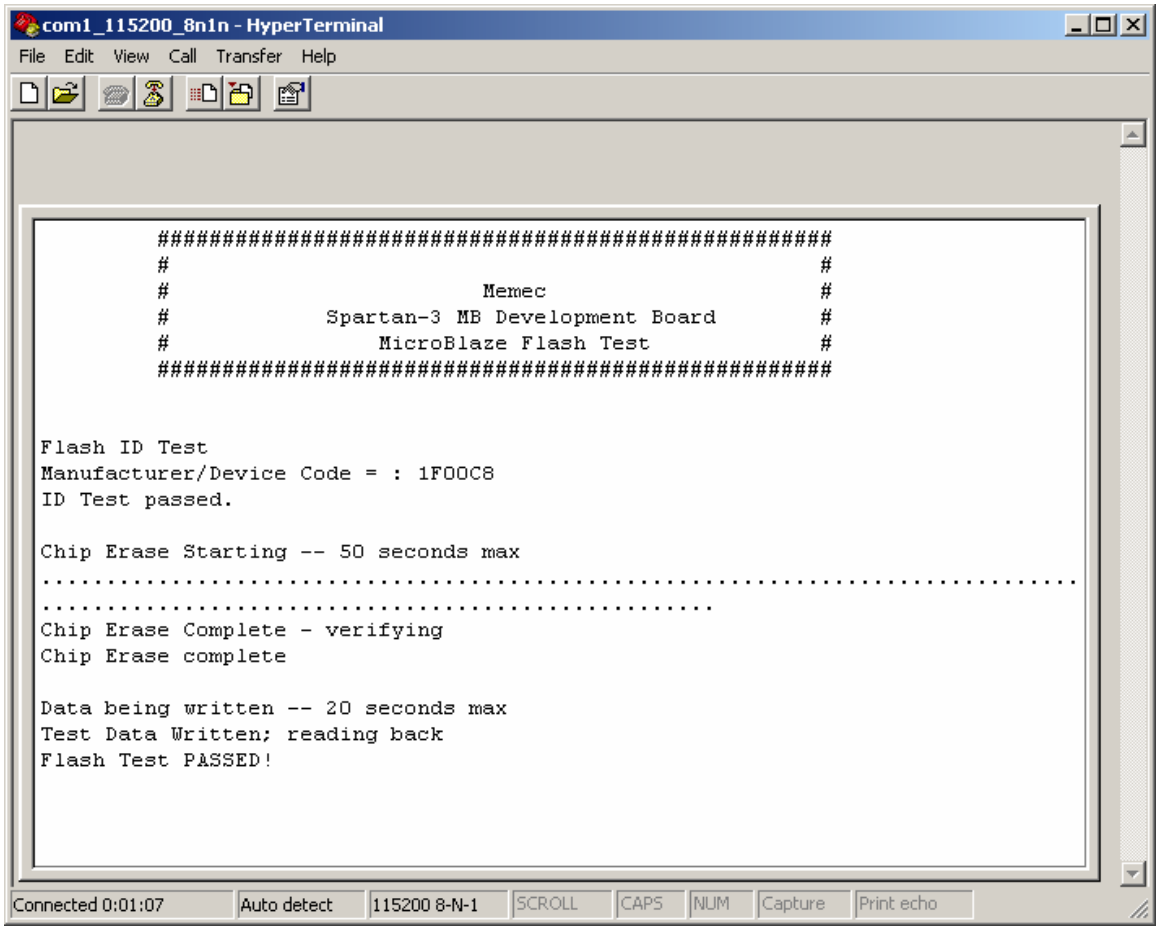


**Figure 13 – Flash_Test Results**

This completes all the memory tests that were included in this project. The next experiment shows how a user creates a bootloader, burns the flash, and bootloads the project with no cables attached and without using XMD.

## Bootloader

A production embedded system does not have a programming cable attached to it. For production, the system must be able to stand on its own, to pull itself up by its own "bootstraps." This experiment will show how to create a bootloading, production system by doing the following:

- Using XPS Flash Programmer to store an SREC copy of a software application targeted to run from DDR and also to create an SREC bootloader which runs from BRAM.
- Creating and programming an MCS PROM file with the MicroBlaze hardware and bootloader image.
- Disconnect the programming cable. Allow the FPGA to self-configure from PROMs, which will run a bootloader to copy the SREC application from Flash to

SRAM, and then jump to SRAM to execute the application from SRAM – no programming cable attached!

## Program the Flash

XPS has a built-in flash programmer which uses an EMC peripheral to access flash. The flash programmer works for all CFI-compliant flash devices, of which the Atmel flash device is one. The default option for the XPS Flash Programmer is a top-boot flash device. The flash on the 3SMB is bottom-boot. Therefore, the flashwriter.tcl file has been copied from %XILINX_EDK%\data\xmd to the project directory with the following command added:

```
set FLASH_BOOT_CONFIG BOTTOM_BOOT_FLASH
```

1. To launch the flash programmer, select **Device Configuration → Program Flash Memory**. (Note that the hardware platform must have previously been downloaded. If you are not following this reference design in sequence, select **Device Configuration → Download Bitstream** first to program the MicroBlaze hardware platform.) The Flash Programmer overrides the default flashwriter.tcl file that is included in the EDK installation with the one it finds in the project directory.
2. Set the parameters as follows (see Figure 14):
   - File to Program: Browse, change File Type to *.elf, and open `Run_From_DDR_Link/executable.elf`
   - Select the check box to Auto-convert the file to SREC
   - Flash Memory Instance Name: Select the flash. The Flash Memory Base Address should be 0x21000000.
   - Scratch Pad Memory properties: Select the DDR. The Scratch Pad Base Address should be 0x22000000.
   - Select the check box to Create the Flash Bootloader

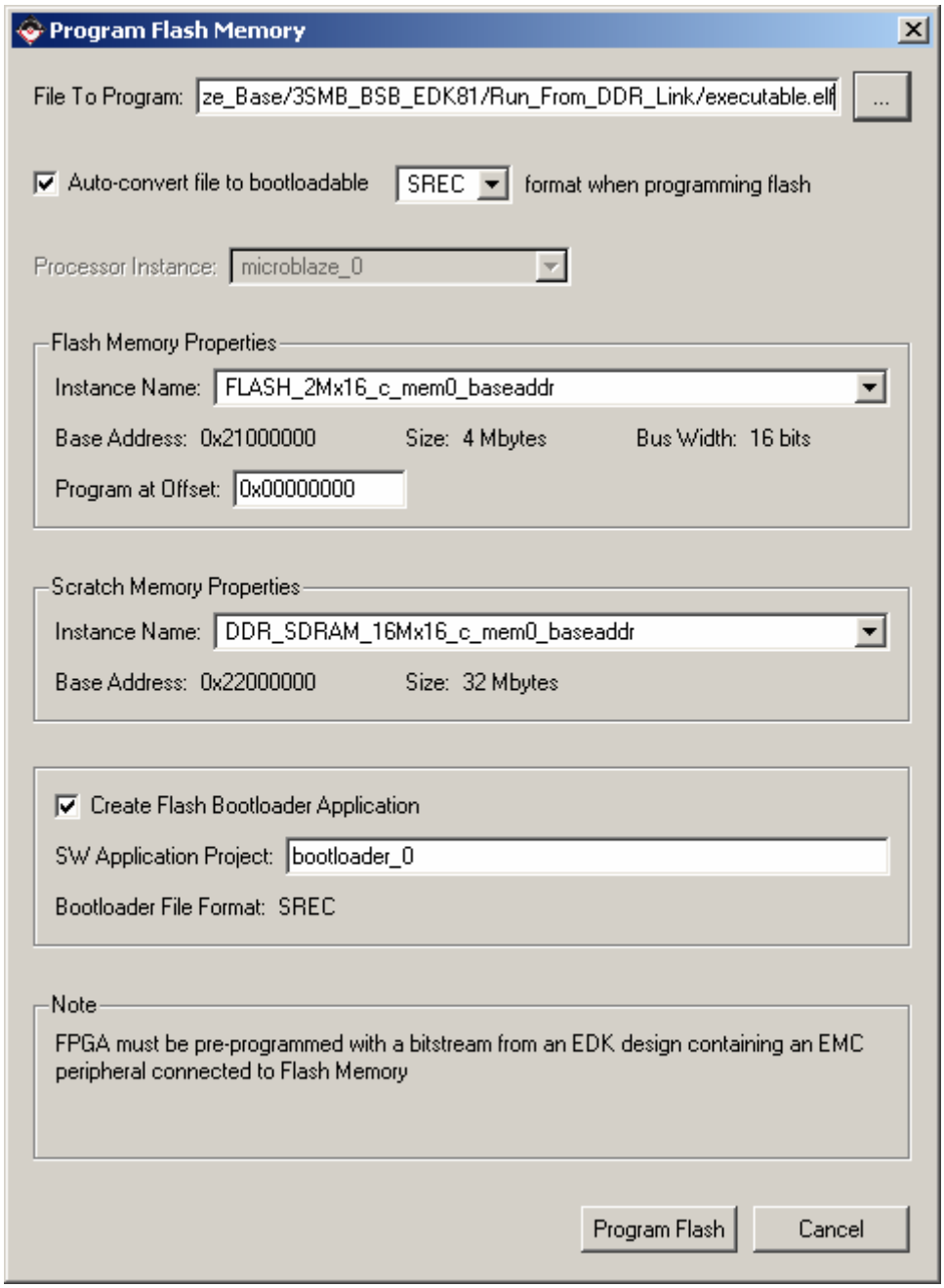


**Figure 14 – Program Flash Memory**

3. Once the parameters are set, click **Program Flash**. First, XPS creates the SREC bootloader application. Click OK to close this window.

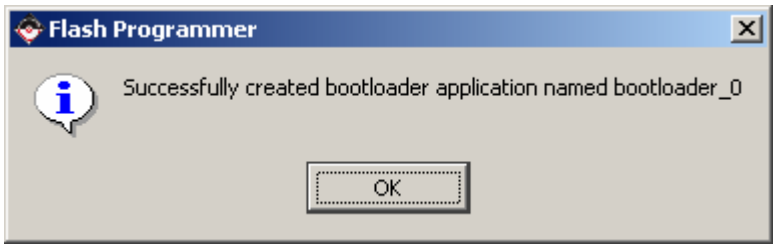


**Figure 15 – SREC Bootloader Created**

4. Next, the programmer will stop whatever is currently executing on the processor and use XMD to run the flash programmer. A copy of the output to the console window is shown below as an example of a successful run.

```
At Local date and time: Mon Jul 03 17:20:57 2006
 xmd -tcl flashwriter.tcl started...
Xilinx Microprocessor Debug (XMD) Engine
Xilinx EDK 8.1.02 Build EDK_I.20.4
Copyright (c) 1995-2005 Xilinx, Inc.  All rights reserved.
Executing user script : flashwriter.tcl
Bottom-boot Flashwriter starting...
Connecting to target...
Connecting to cable (Parallel Port - LPT1).
Checking cable driver.

 Driver windrvr6.sys version = 7.0.0.0. LPT base address = 0378h.
 ECP base address = 0778h.
 ECP hardware is detected.
Cable connection established.

Connecting to cable (Parallel Port - LPT1) in ECP mode.
Checking cable driver.
 Driver xpc4drvr.sys version = 1.0.4.0. LPT base address = 0378h.
 Cable Type = 1, Revision = 3.
 Setting cable speed to 5 MHz.
Cable connection established.

JTAG chain configuration
--------------------------------------------------
Device   ID Code        IR Length    Part Name
 1       05046093           8        XCF04S
 2       05044093           8        XCF01S
 3       01434093           6        XC3S1500
Assuming, Device No: 3 contains the MicroBlaze system
Connected to the JTAG MicroProcessor Debug Module (MDM)
No of processors = 1

MicroBlaze Processor 1 Configuration :
-------------------------------------
Version.............................4.00.a
No of PC Breakpoints................2
No of Read Addr/Data Watchpoints...0
No of Write Addr/Data Watchpoints..0
Instruction Cache Support.........on
Instruction Cache Base Address.....0x22000000
Instruction Cache High Address.....0x23ffffff
Data Cache Support.................on
Data Cache Base Address............0x22000000
Data Cache High Address............0x23ffffff
Exceptions  Support................off
FPU  Support.......................off
FSL DCache Support.................on
FSL ICache Support.................on
Hard Divider Support...............off
```

```
Hard Multiplier Support.....on
Barrel Shifter Support.....off
MSR clr/set Instruction Support....off
Compare Instruction Support.....off
Number of FSL ports.....1
MBsfsl(0)-MDMmfsl(0) Connected.....Yes
JTAG MDM Connected to MicroBlaze 1

Connected to "mb" target. id = 0
Starting GDB server for "mb" target (id = 0) at TCP port no 1234
done.
Opening image file to be programmed...done.
Setting up Flashwriter sources for compilation --
Copying flashwriter sources from C:/EDK81/data/xmd/flashwriter/src to
./etc/flashwriter/src...done.
Compiling flashwriter (CFI Query Mode)...
done.
Target reset successfully
Downloading flashwriter to memory... section, .text: 0x22000000-0x220012fc
        section, .data: 0x22001300-0x22001500
        section, .bss: 0x22001500-0x22001890

Downloaded Program ./etc/flashwriter/flashwriter.elf

Setting PC with program start addr = 0x22000000
done.

./etc/flashwriter/flashwriter.elf :: rcvptr(0x22001314):    0x22001318

./etc/flashwriter/flashwriter.elf :: sndptr(0x22001310):    0x2200132c

./etc/flashwriter/flashwriter.elf :: membufptr(0x22001300):    0x22001500

./etc/flashwriter/flashwriter.elf :: membufsiz(0x22001304):    0x00000004
Setting breakpoint at 0x2200009c
Performing CFI Query on the flash part(s)...Processor started. Type "stop" to stop
processor

XMD% done.

./etc/flashwriter/flashwriter.elf :: devinfop(0x2200130c):    0x22001340
./etc/flashwriter/flashwriter.elf :: devinfosz(0x22001308):    0x000001c0

Flash part information:
Device Algorithm       : AMD/Fujitsu Standard
Number of flash parts  : 1
Mode of each flash part: 16 bits wide
Size of each flash part       : 4 MB
Flash boot config      : BOTTOM_BOOT_FLASH
Compiling flashwriter (Block Erase Mode)...
done.
Target reset successfully
Downloading flashwriter to memory... section, .text: 0x22000000-0x22000d20
        section, .data: 0x22000d20-0x22000f34
        section, .bss: 0x22000f38-0x220012e0
Downloaded Program ./etc/flashwriter/flashwriter.elf

Setting PC with program start addr = 0x22000000
done.
./etc/flashwriter/flashwriter.elf :: rcvptr(0x22000d34):    0x22000d38
./etc/flashwriter/flashwriter.elf :: sndptr(0x22000d30):    0x22000d4c

./etc/flashwriter/flashwriter.elf :: membufptr(0x22000d20):    0x22000f38
./etc/flashwriter/flashwriter.elf :: membufsiz(0x22000d24):    0x00000004
./etc/flashwriter/flashwriter.elf :: devinfop(0x22000d2c):    0x22000d60

Setting breakpoint at 0x2200009c
Erasing appropriate flash block(s)...Processor started. Type "stop" to stop processor

XMD% Processor started. Type "stop" to stop processor
```

```
XMD% done.
Compiling flashwriter (Stream Buffer Calibration)...
done.
Calculating flashwriter application size...
done.
Calibrated stream buffer size: 7346 bytes
Compiling flashwriter (Program Mode)...
done.
Target reset successfully
Downloading flashwriter to memory...  section, .text: 0x22000000-0x22000a28
        section, .data: 0x22000a28-0x22000c3c
        section, .bss: 0x22000c40-0x22002c98
Downloaded Program ./etc/flashwriter/flashwriter.elf
Setting PC with program start addr = 0x22000000
done.

./etc/flashwriter/flashwriter.elf :: rcvptr(0x22000a3c):    0x22000a40
./etc/flashwriter/flashwriter.elf :: sndptr(0x22000a38):    0x22000a54
./etc/flashwriter/flashwriter.elf :: membufptr(0x22000a28):    0x22000c40
./etc/flashwriter/flashwriter.elf :: membufsiz(0x22000a2c):    0x00001cb2

./etc/flashwriter/flashwriter.elf :: devinfop(0x22000a34):    0x22000a68

Setting breakpoint at 0x2200009c
Processor started. Type "stop" to stop processor

XMD% Programming of the flash part with the image starts...
Transferring data to flashwriter...
done.
Processor started. Type "stop" to stop processor


XMD%
Progress: 100.00%
Programming completed successfully.
Processor started. Type "stop" to stop processor
Flashwriter terminating...

Done!
```

## Create the PROM Files

Next, an MCS PROM image containing a bootloader is programmed to the 3SMB board.

5. In XPS, mark software application **Project: Bootloader** for BRAM initialization. Unmark any other programs.
6. Select **Device Configuration → Update Bitstream**. This will create `implementation/download.bit` which will then be converted to the MCS format.
7. Launch iMPACT from the Start Menu by selecting **Start → All Programs → Xilinx ISE 8.1i → Accessories → iMPACT**
8. In the wizard, select **create a new project file**, then browse to the project directory and select a name for the iMPACT session. Click **OK**.
9. Select **Prepare a PROM File**, then **Next**.

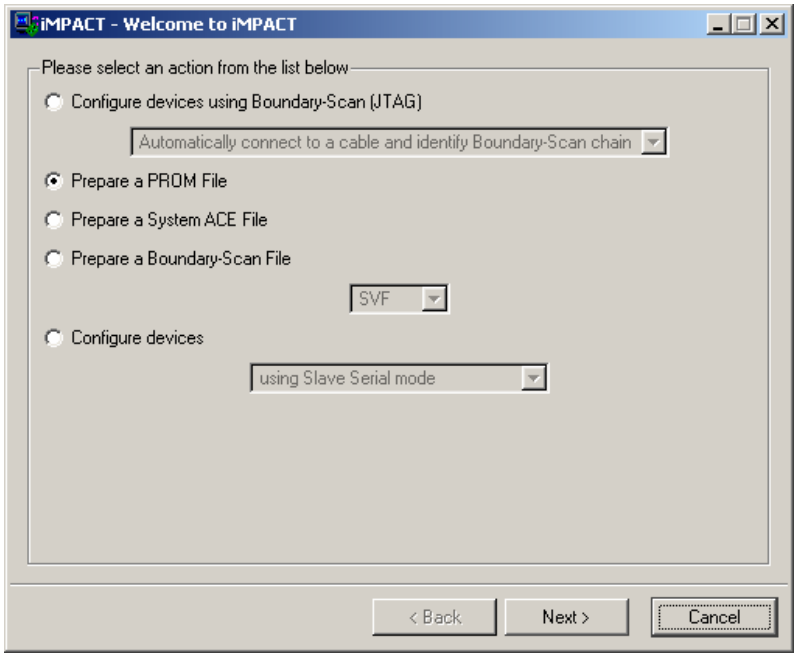


**Figure 16 – iMPACT Prepare PROM File**

10. Change the *PROM File Name* to '**bootload**' and also browse to the project location. Click **Next >**.

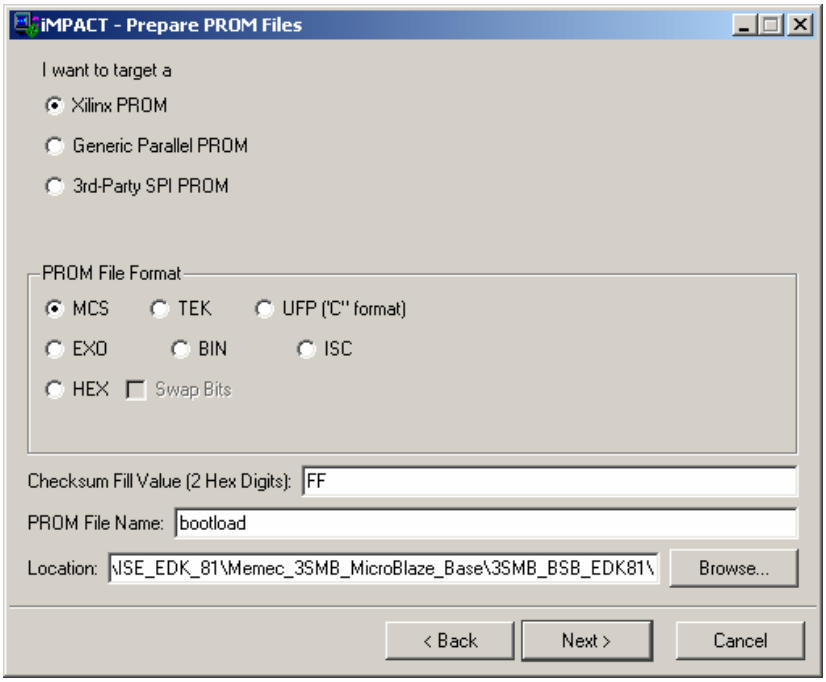


**Figure 17 – PROM File Name and Location**

11. Select the **xcf** family and the **xcf04s** PROM; click **Add**; Now select the **xcf01s** and click **Add** again. Click **Next**.

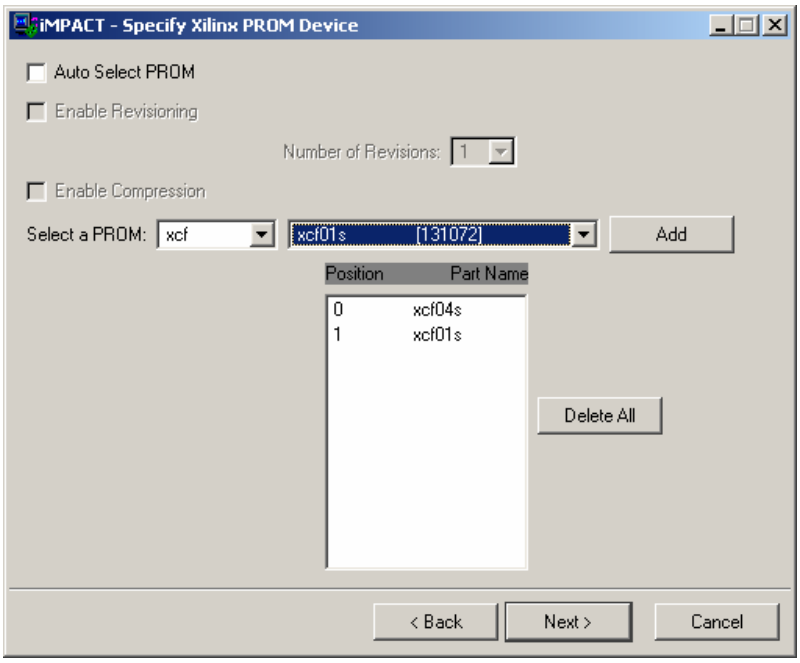


**Figure 18 – 3SMB PROMs Selected**

12. At the summary page, click **Finish**.

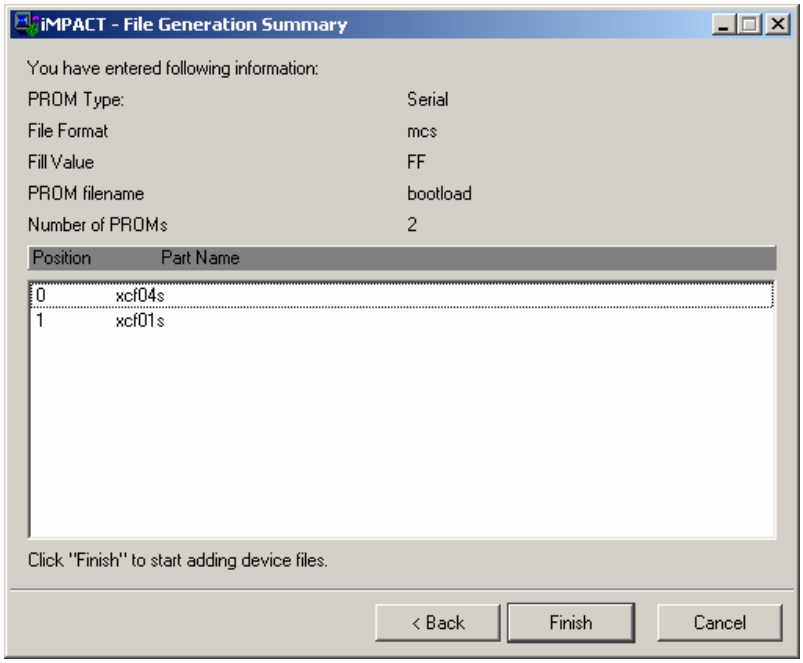


**Figure 19 – PROM Generation Summary**

13. iMPACT prompts you to *Start adding device file to Data Stream: 0* . Click **OK**.


**Figure 20 – Start Adding Device Files**

14. Browse and select `implementation/download.bit` and select **Open**. A warning will be displayed that iMPACT is changing the Startup Clock to CCLK. This is expected since a PROM configures an FPGA using CCLK rather than the JTAG clock. Click **OK**. Click **No** when asked if any more design files are to be added.

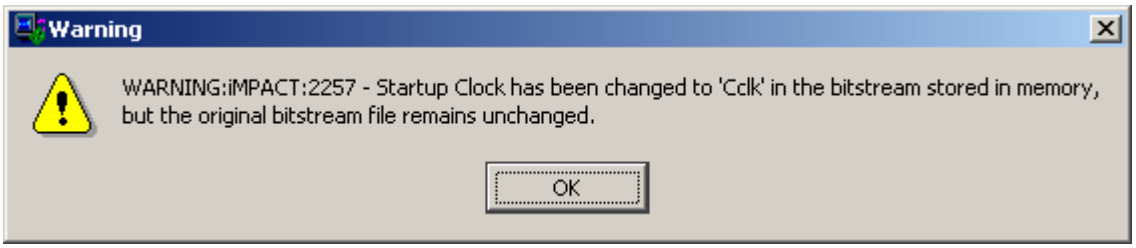


**Figure 21 – Startup Clock Changed to CCLK**

15. Click **OK**.
16. In iMPACT, select **Operations → Generate file…** . If successful, Figure 22 will be displayed. The .mcs files now exist in the project directory.

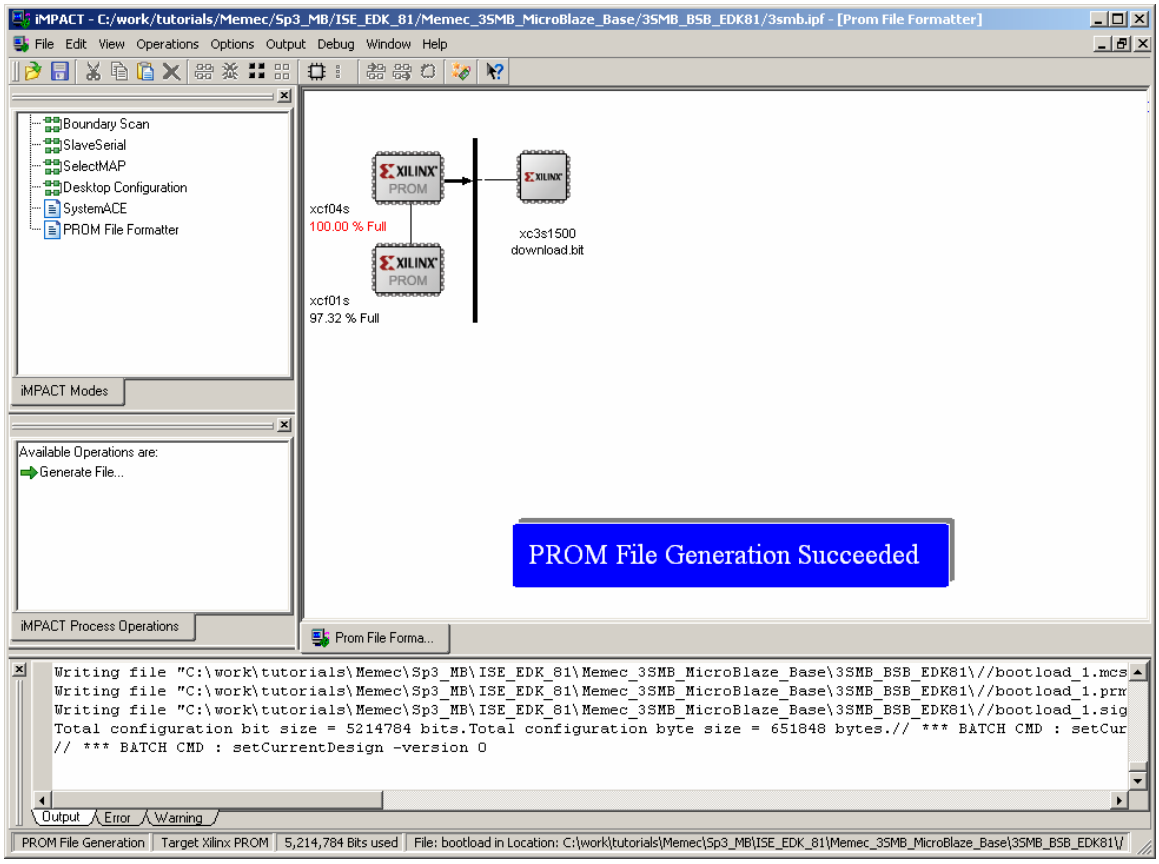


**Figure 22 – PROM File Generation Succeeded**

## Burning the PROM

To burn the .mcs PROM images into the PROMs, do the following:

17. In iMPACT, double-click on **Boundary Scan** under the *iMPACT Modes* tab..
18. Right-click in the open space and select **Initialize Chain**. iMPACT will show a view of the JTAG chain for the 3SMB board and launch into a file selection wizard.
19. For the XCF04S PROM, browse to and select the newly generated **bootload_0.mcs** PROM file. Click **Open**. Repeat the process for the XCF01S PROM, this time selecting the **bootload_1.mcs** PROM file. Cancel the file selection for the FPGA.

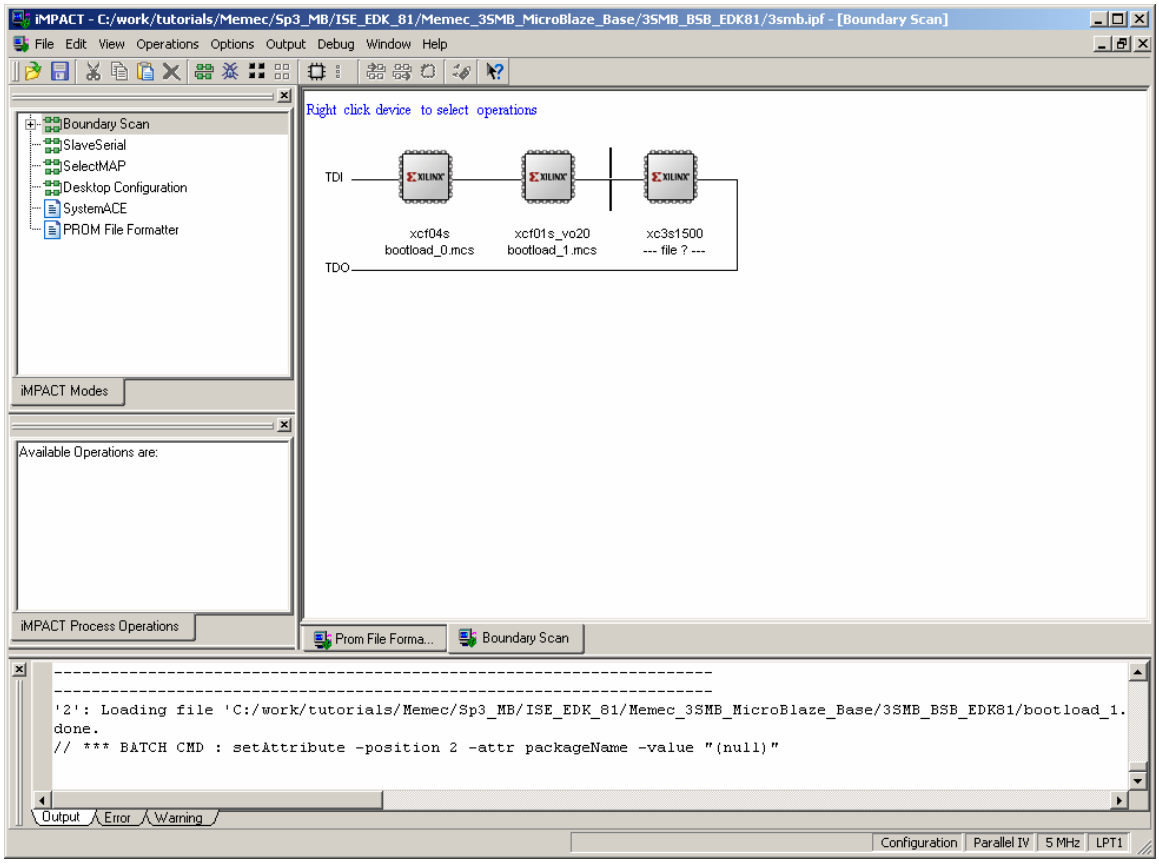


**Figure 23 – .mcs Assigned to PROM**

20. Right-click on the xcf04s icon and select **Program**.
21. Check the options **Erase Before Programming** and **Verify**. Click **OK**.

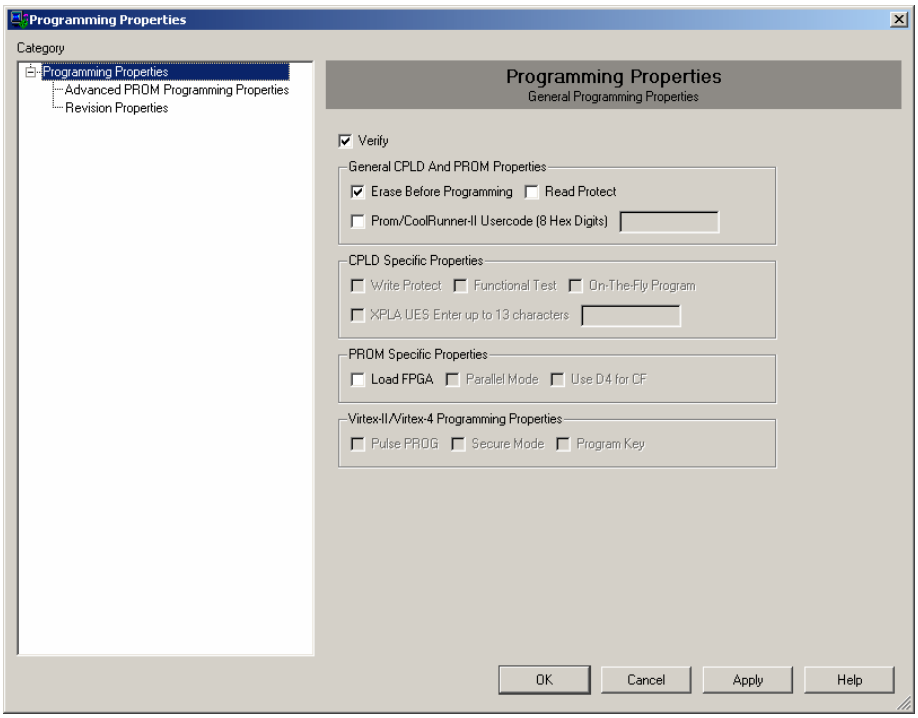


**Figure 24 – PROM Programming Options**

22. After a slight delay while the PROM is erased, iMPACT programs and verifies the PROM. A successful download of the PROM is shown in Figure 25.

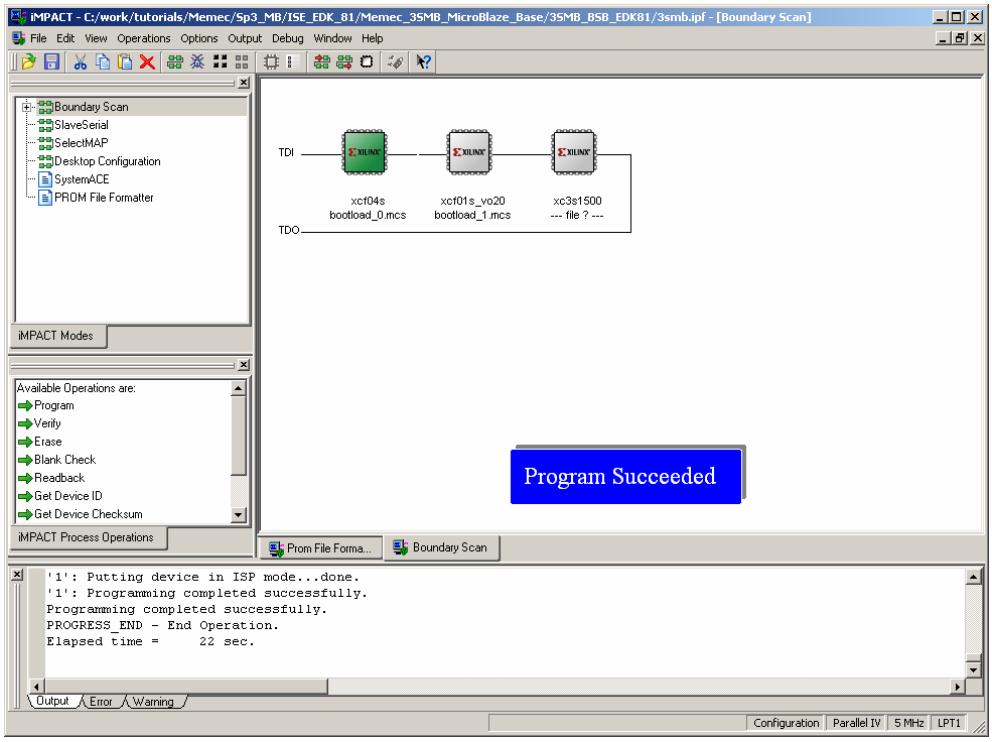


**Figure 25 – PROM Download Successful**

23. Repeat the programming process for the xcf01s PROM.
24. When both PROMs are programmed, close iMPACT.
25. Turn off the board by moving SW1 to OFF. Place three jumpers on MODE pins M0, M1, and M2. The JTAG cable can be disconnected. Turn on power. The PROMs will automatically configure the FPGA on power-up. The bootloader application will copy the flash contents to DDR then jump to DDR. The results are shown in Figure 26.

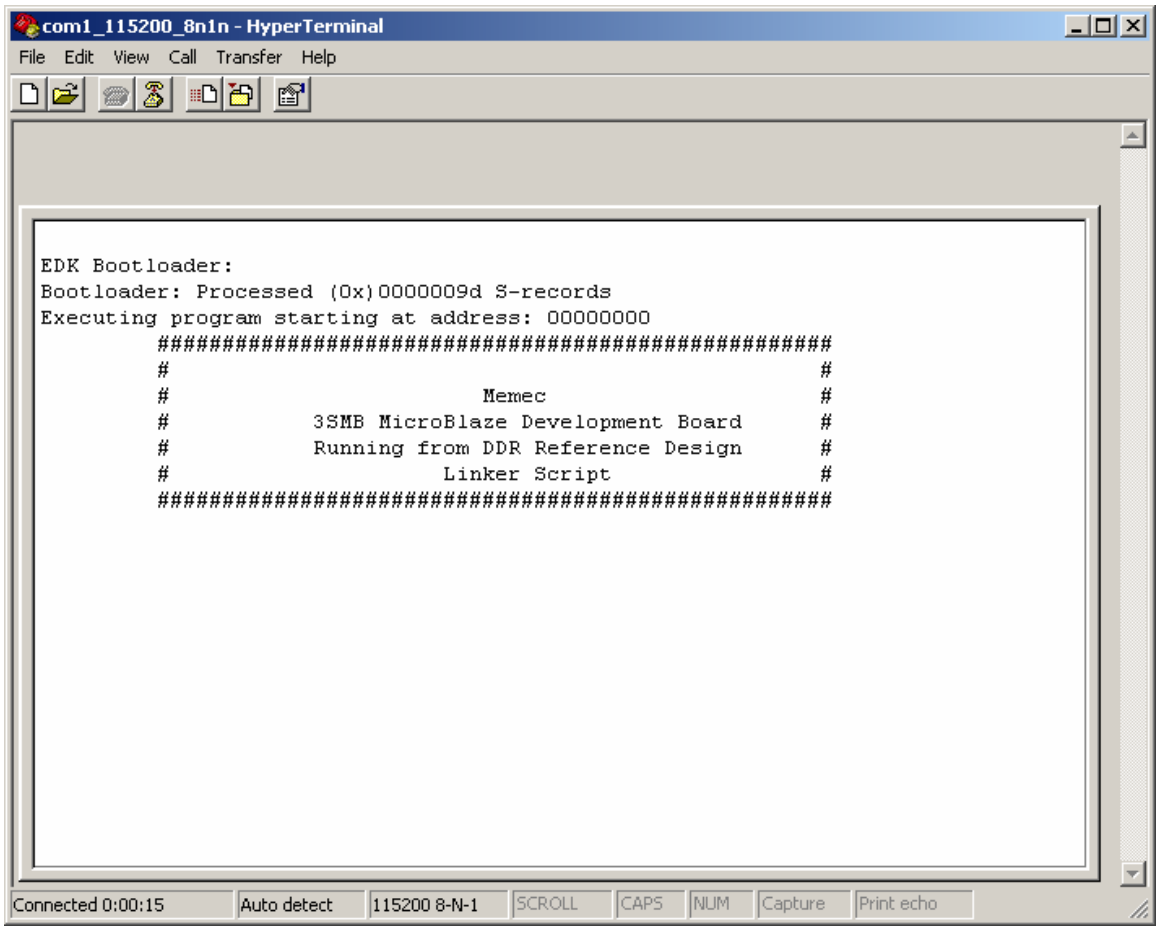


**Figure 26 – Bootloaded Application**

## Revision History

| Date | Version | Revision |
|---|---|---|
| 12/07/04 | 6.3 | Initial Memec release. |
| 12/21/04 | 6.3a | Added bootloader experiment |
| 09/29/05 | 7.1 | Updated to EDK 7.1, including built-in XPS bootloader generation. |
| 07/07/06 | 8.1 | Updated to EDK 8.1, including addition of CacheLink DDR. |

# Appendix A: Starting a HyperTerminal Session

1. Start a HyperTerminal session by double-clicking **com1_115200_8n1n.ht**. Or to start HyperTerminal manually, follow steps 2 through 5:
2. **Start→Programs→Accessories→Communications→HyperTerminal**


**Figure 27 – HyperTerminal Connection Name**

3. Enter edk_demo in the "Name" field and select OK.


**Figure 28 – HyperTerminal COM**

4. Select "COM1" from "Connect using" drop down menu and select OK.

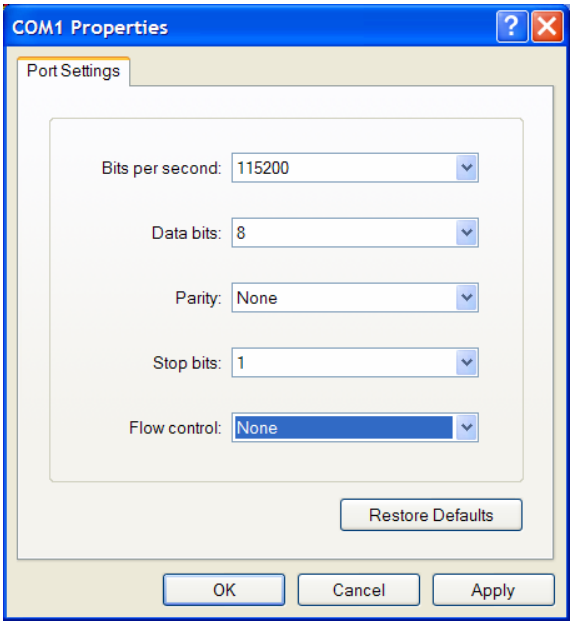


**Figure 29 – HyperTerminal Settings**

5. Enter the above port settings and then select OK. You should see the following HyperTerminal window.

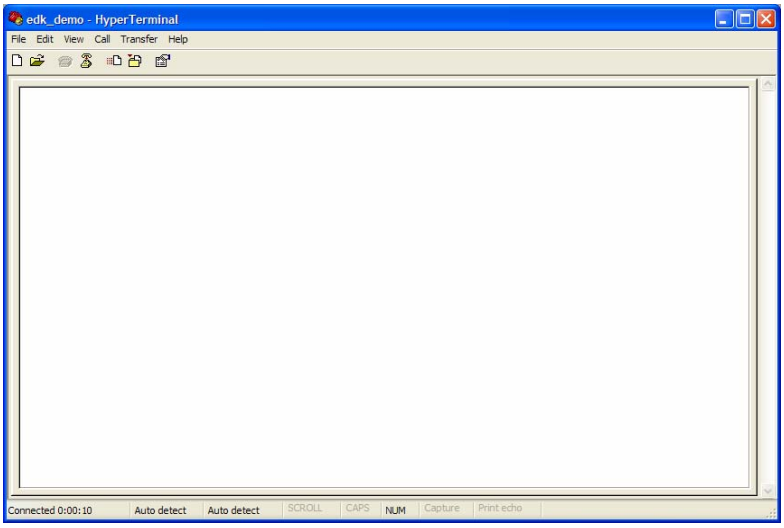


**Figure 30 – HyperTerminal Launched**