Building and Optimizing a MicroBlaze BSB System for Spartan-3 MB

June 29, 2006
Version 8.1

Overview

This document describes how to build an optimized system using the Xilinx MicroBlaze™ soft processor core running on the Memec Spartan-3 MB (3SMB) development board. Starting with Base System Builder (BSB) in Xilinx Platform Studio (XPS), a system is defined and then modified to run the MicroBlaze system at 50 MHz with the DDR running at 75 MHz. The multi-channel, FSL cache connects the DDR to the MicroBlaze system.

The system built in this tutorial is the basis for the following Memec reference designs:

Memec 3SMB MicroBlaze Using Memory Reference Designs
Memec 3SMB MicroBlaze WebServer Reference Design

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

Memec Spartan-3 MB Development Board Introduction

A photograph of the Memec 3SMB development board is shown in Figure 1. Various features and circuits are pointed out. In this exercise, the following features will be utilized:

- 75 MHz oscillator
- 8-bit DIP switch

[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

- 4 user LEDs
- 2 user and 1 reset push button switches
- Two 7-segment displays
- JTAG header
- DDR SDRAM
- 10/100 Ethernet PHY
- Flash
- RS232 & USB UARTs

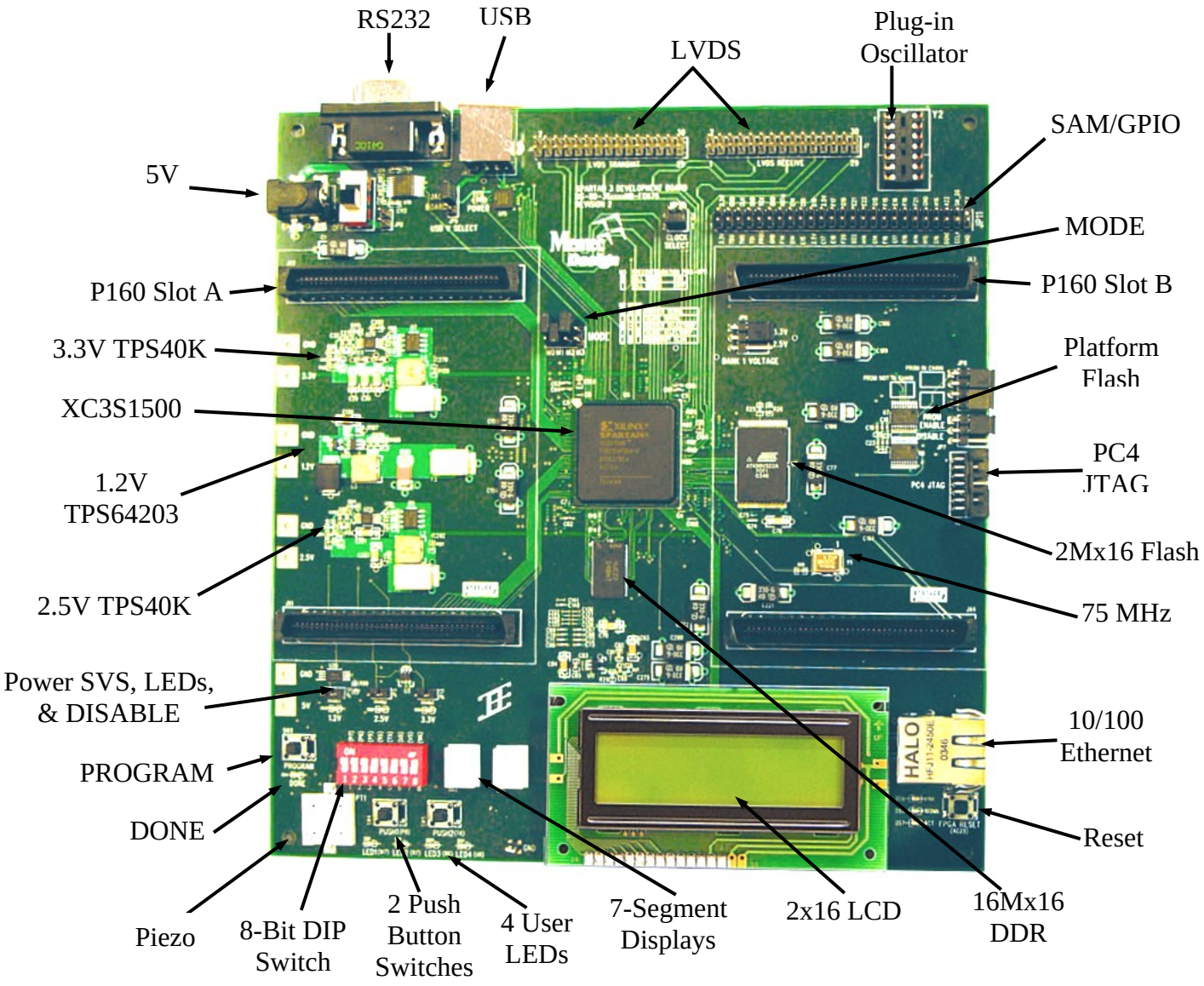

Figure 1 – Spartan-3 MB Development Board

A complete block diagram of the board is shown in Figure 2.

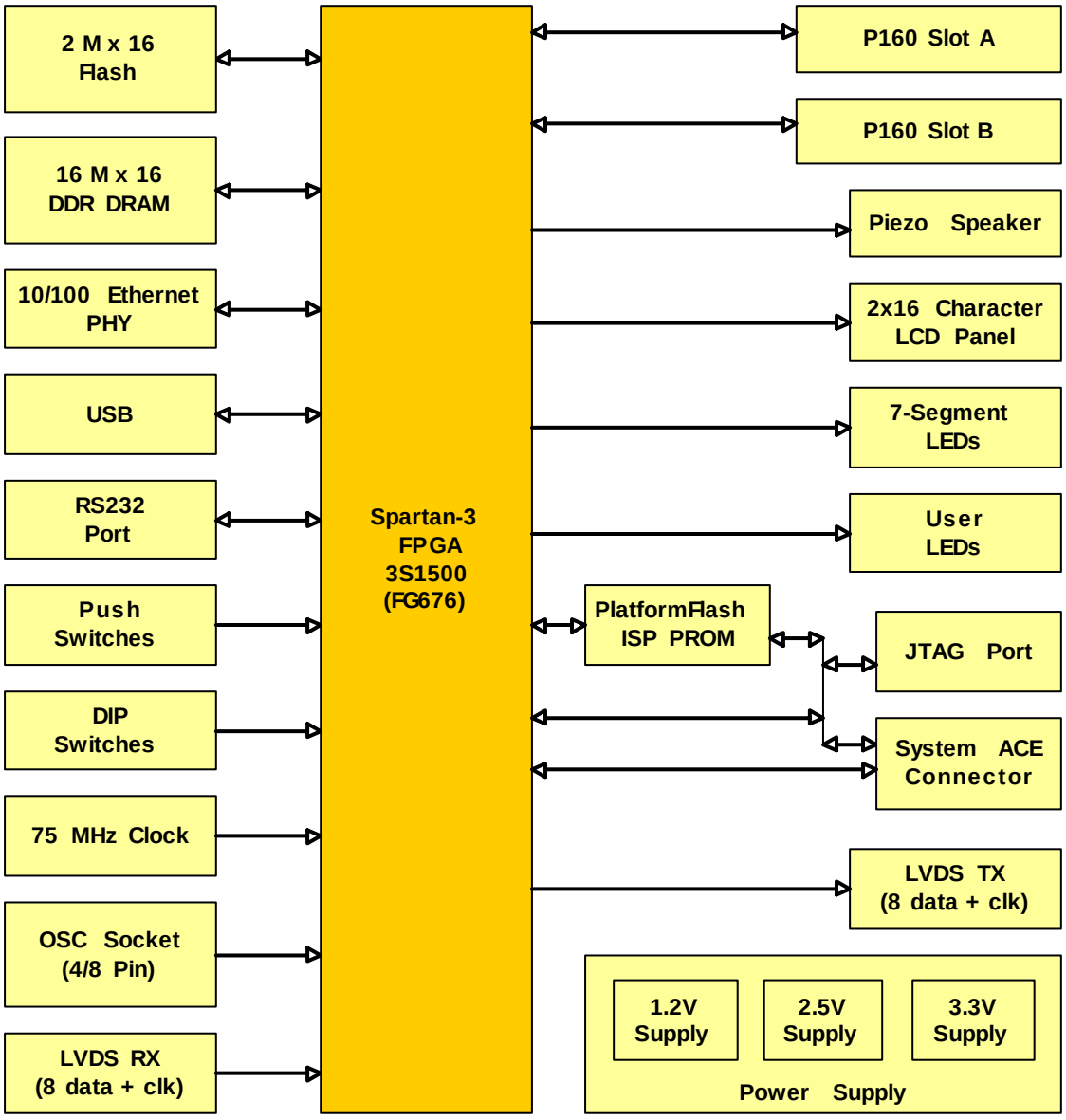

Figure 2 – Spartan-3 MB Block Diagram

Understanding the Hardware Platform

A block diagram of the hardware platform to be constructed is shown in Figure 3. Besides the MicroBlaze processor and busses, the design consists of 32 KB local memory, 32 MB DDR, 4 MB Flash, Ethernet interface, timer, interrupt controller, two UARTs, a debug peripheral (MDM) and multiple GPIOs.

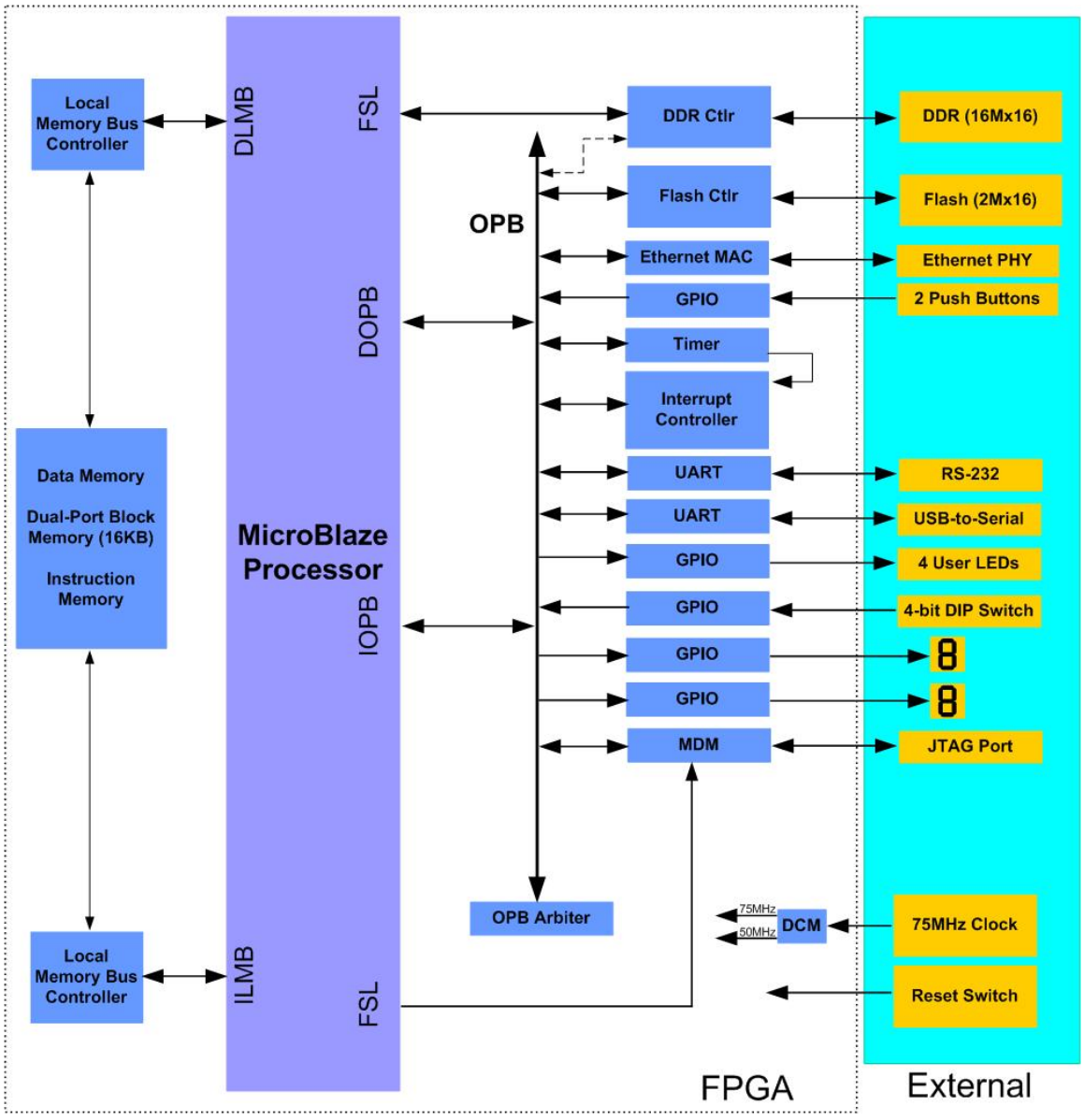

Figure 3 – Block Diagram of BSB-Generated System

Creating the Hardware Platform Using BSB

The following steps discuss how to launch the BSB in XPS and how to create the 3SMB system. Please note that the XBD files for the Memec development boards from Avnet, including the 3SMB board, must be previously installed. The XBD install file and instructions are available at the Avnet Design Resource Center:

www.em.avnet.com/drc

1. Launch XPS. To do this, select **Start → Programs → Xilinx Platform Studio 8.1i → Xilinx Platform Studio**.
2. Select **File →New Project** and then select the radio button for **Base System Builder…** Press **OK**. This will launch the *Base System Builder Wizard*.

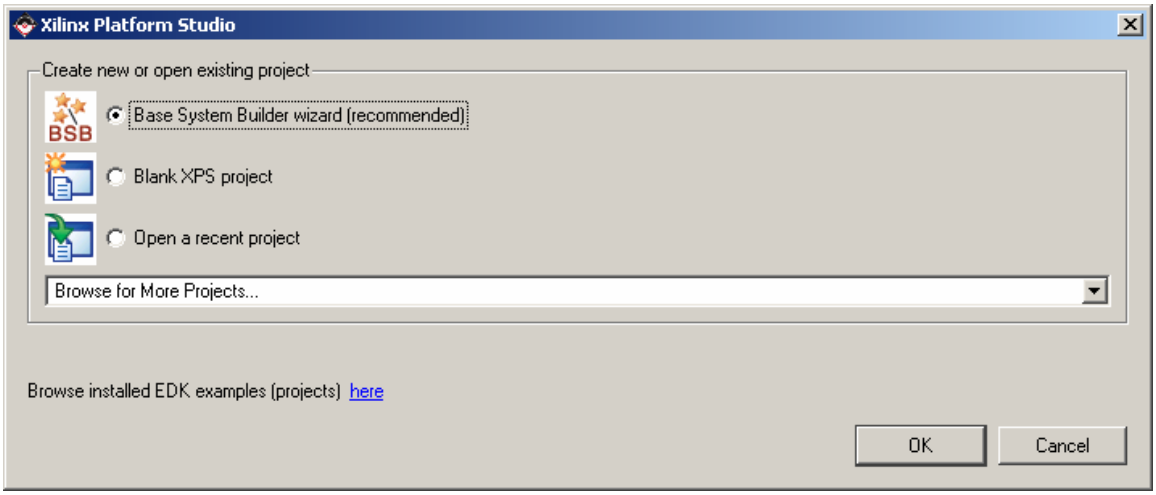

Figure 4 – New Base System Builder Project

3. Browse to an appropriate folder location. Leave the project name as `system.xmp`. If your XBD files were installed to a User Repository, check the Peripheral Repository box and browse to its location. Click **OK**.

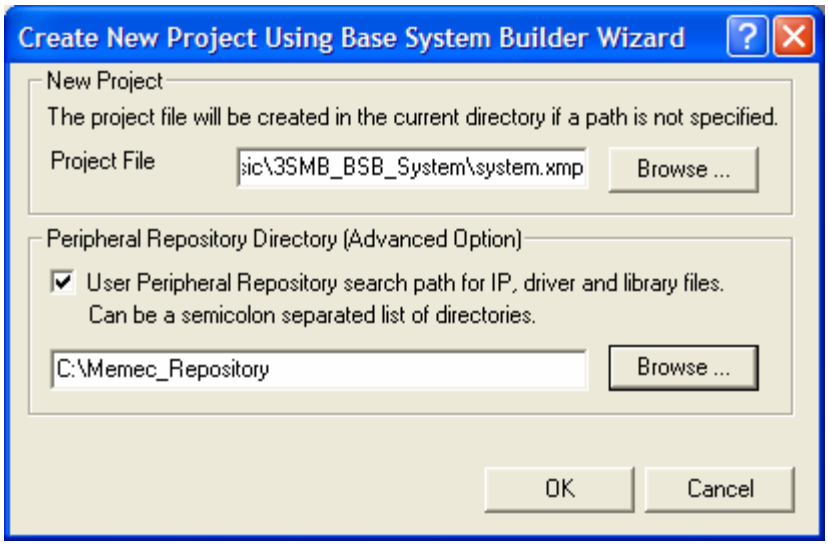

Figure 5 – New Project Location

4. Select the radio button to create a new design. Click **Next>**.

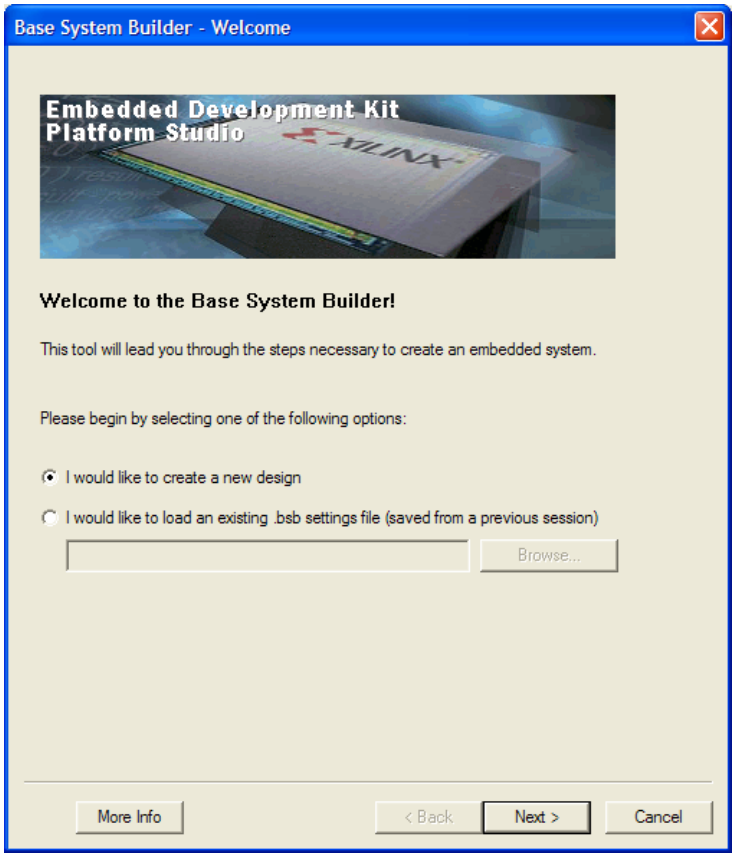

Figure 6 – Create a New BSB Design

5. Specify the target development board: **Memec**, **Spartan-3 3S1500MB Development Board**, **Rev 2**, as shown in Figure 7. Click **Next>**. (Note: if the Memec 3S1500MB board is not an available choice, then the Memec XBD files need to be installed.)

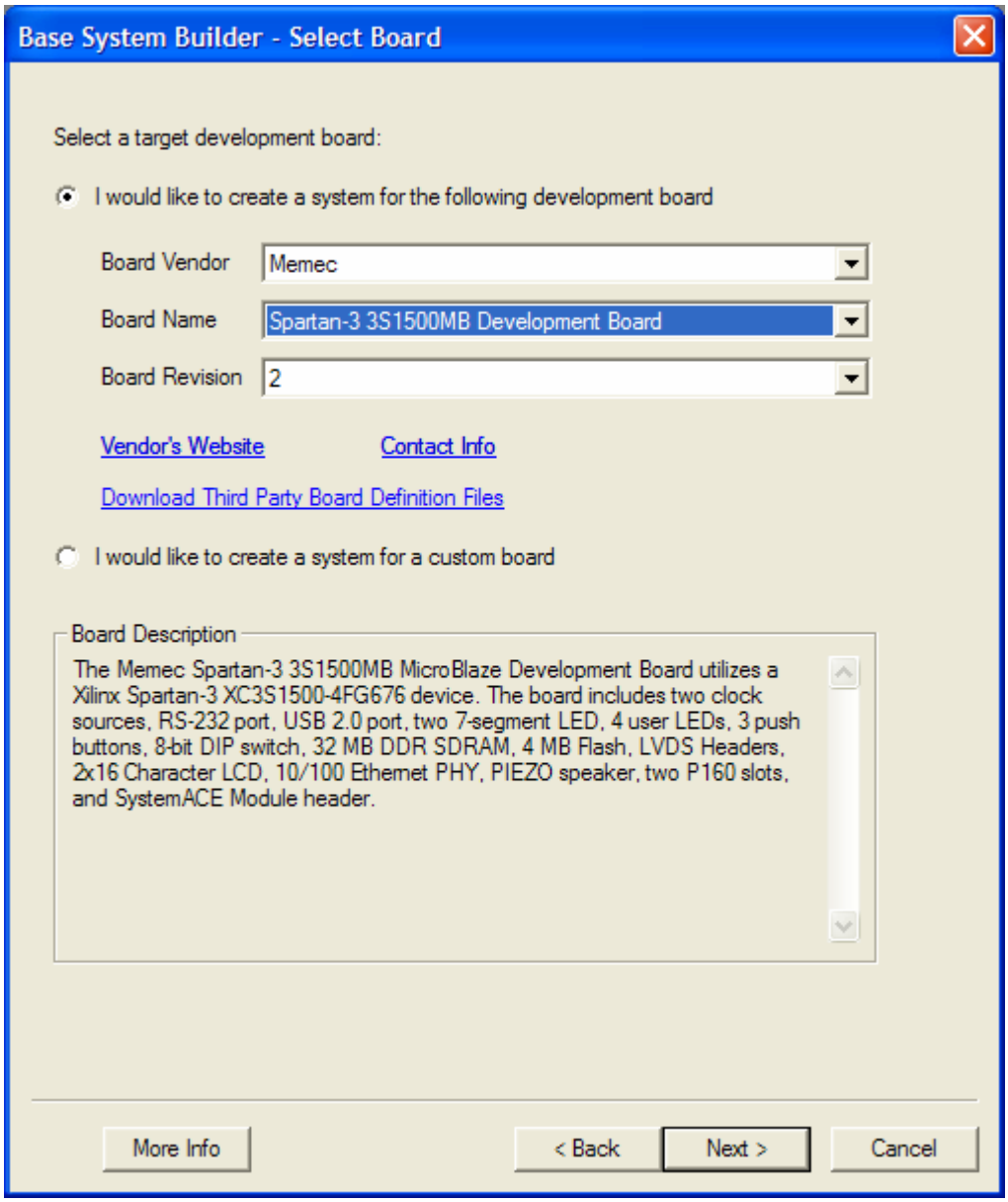

Figure 7 – Specify Target Development Board

6. Since MicroBlaze is the only EDK-supported processor available on Spartan-3, no changes need to be made to the next dialog. Note that the FPGA selected is the 1.5 Million gate Spartan-3 in the FG676 package. Click **Next>**.

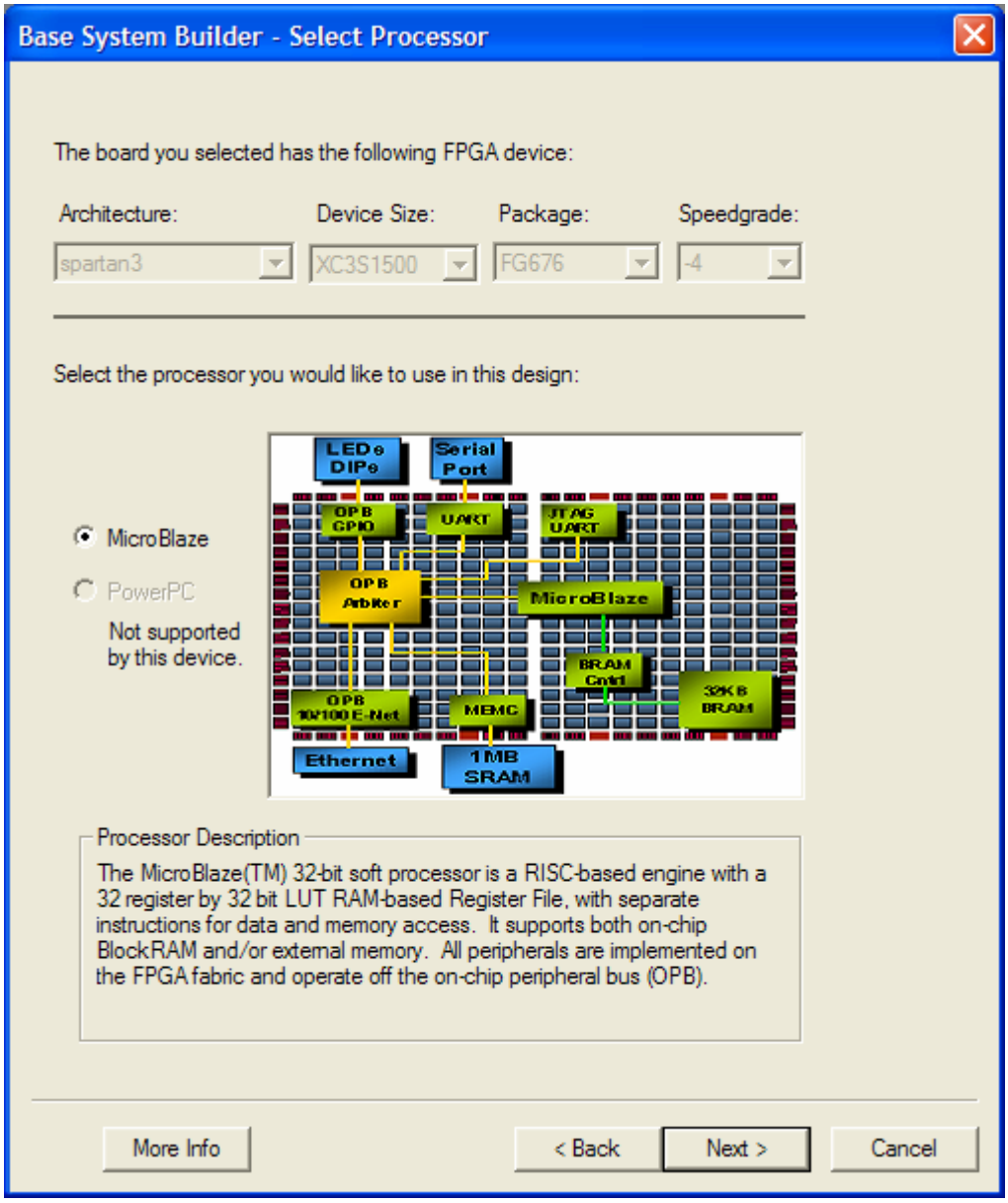

Figure 8 – BSB Processor Selection

7. Reduce the *Processor Bus clock frequency* to 50.00 MHz.
8. Increase *Local Data and Instruction Memory* to **16 KB**. The maximum allowed is determined by the amount of BlockRAM contained on a given chip. The XC3S1500 has a total of 32 BlockRAMs, each of which is 2KB. Therefore, using all 32 BlockRAMs would allow 64KB of local memory, as long as no other devices need BlockRAM. However, in this design, only 16 KB is used for local memory since both the cache and ethernet controller will also consume BlockRAM.
9. Select the radio button in the *Cache setup* section to *Enable cache link*. Change the *Cached memory* to **DDR_SDRAM_16Mx16**

10. Note that the **On-chip H/W debug module**, which is the MicroBlaze Debug Module (MDM), is included by default. Click **Next>**.

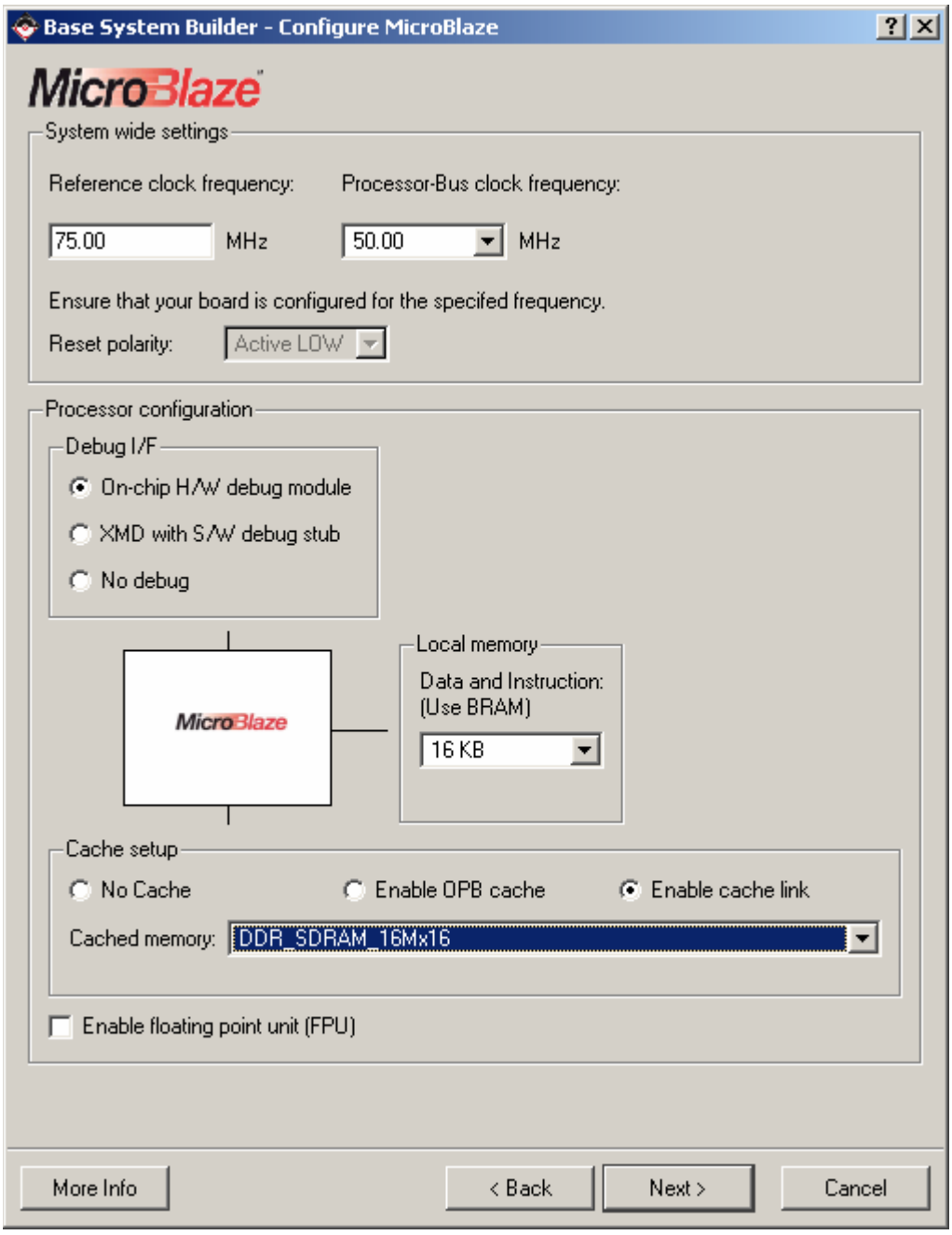

Figure 9 – BSB Configure Processor

The next several screens may not look identical to what you see on your computer, depending on the screen resolution.

11. Change the RS232 and USB_UART Baudrates to 115200. Click **Next>**.

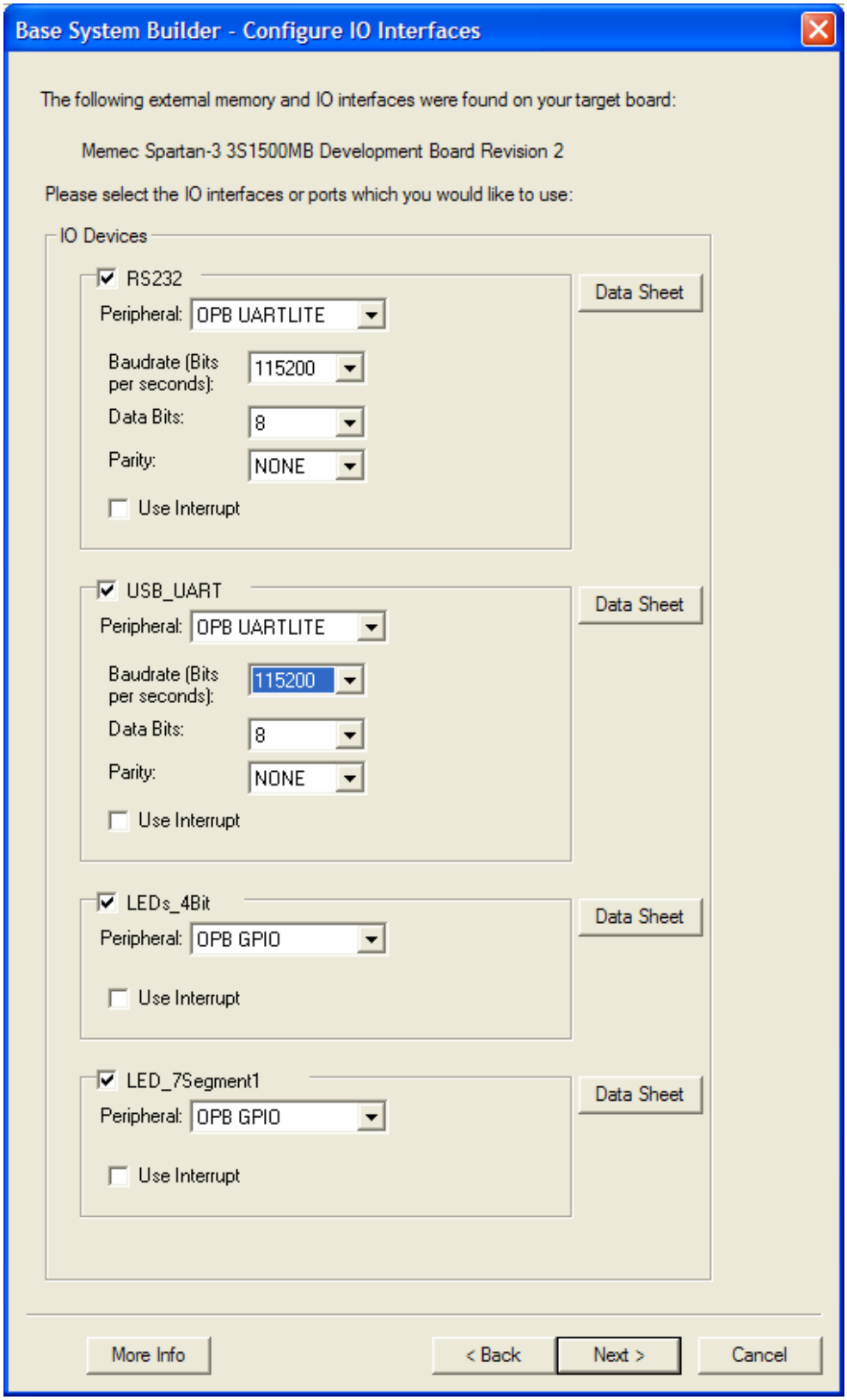

Figure 10 – BSB Configure UARTs

12. Unselect the **FLASH_READY** and **SysACE_CompactFlash** Peripherals. Click **Next>**.

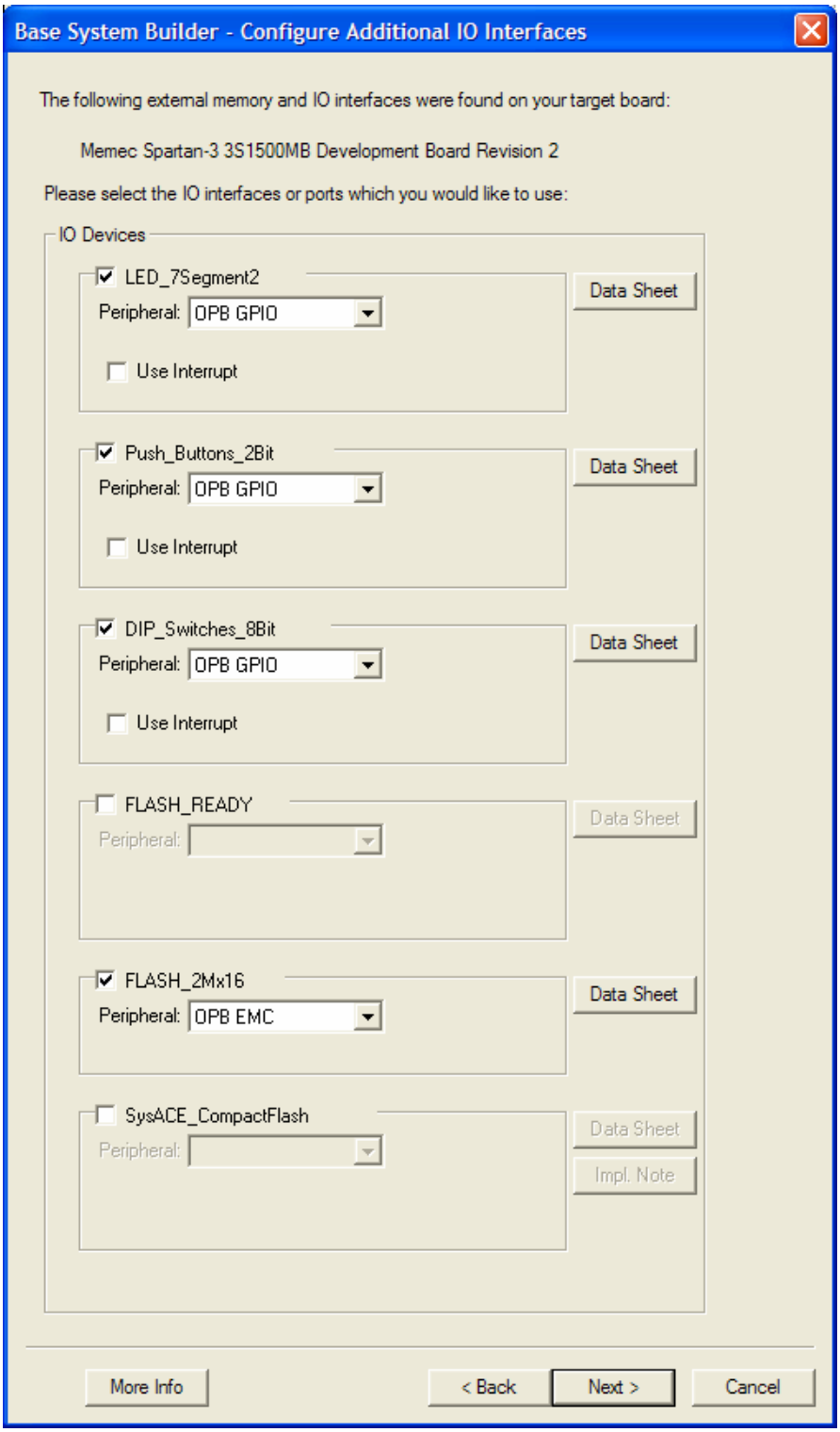

Figure 11 – BSB Configure Peripherals

13. The defaults for the **DDR_SDRAM_16Mx16** and **Ethernet_MAC** Peripherals are acceptable. Click **Next>**.

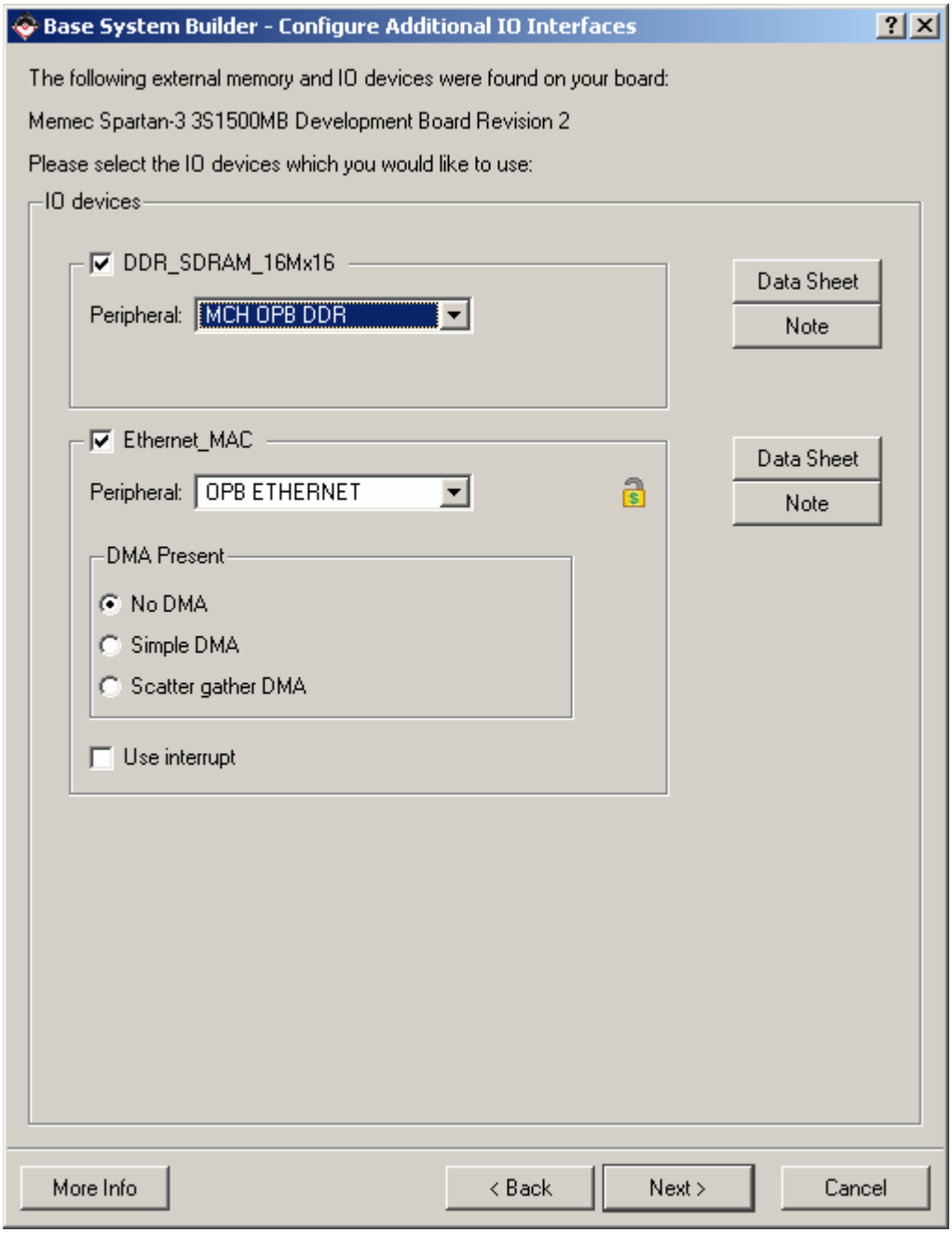

Figure 12 – BSB Configure DDR & Ethernet

14. In the *Add Internal Peripherals* dialog, select **Add Peripheral**.

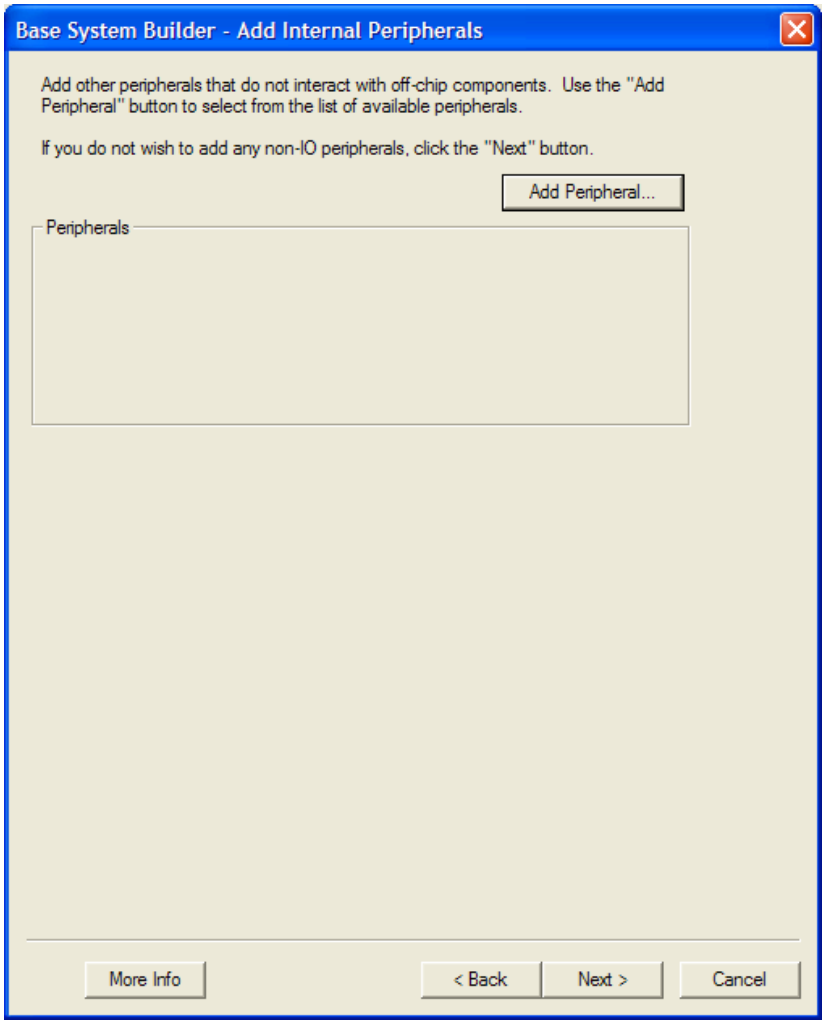

Figure 13 – BSB Add Internal Peripherals

15. Add the **OPB TIMER** peripheral, as shown in Figure 14. Click **OK**.

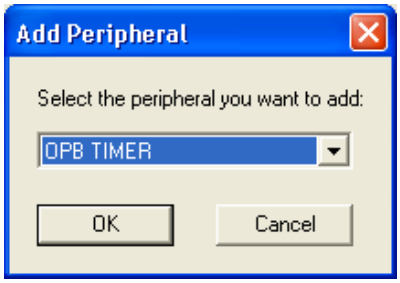

Figure 14 – Add OPB TIMER Internal Peripheral

16. For the OPB TIMER instance, select the radio button for *One timer is present*, and check the box to *Use Interrupt*. See Figure 15. Click **Next >**.

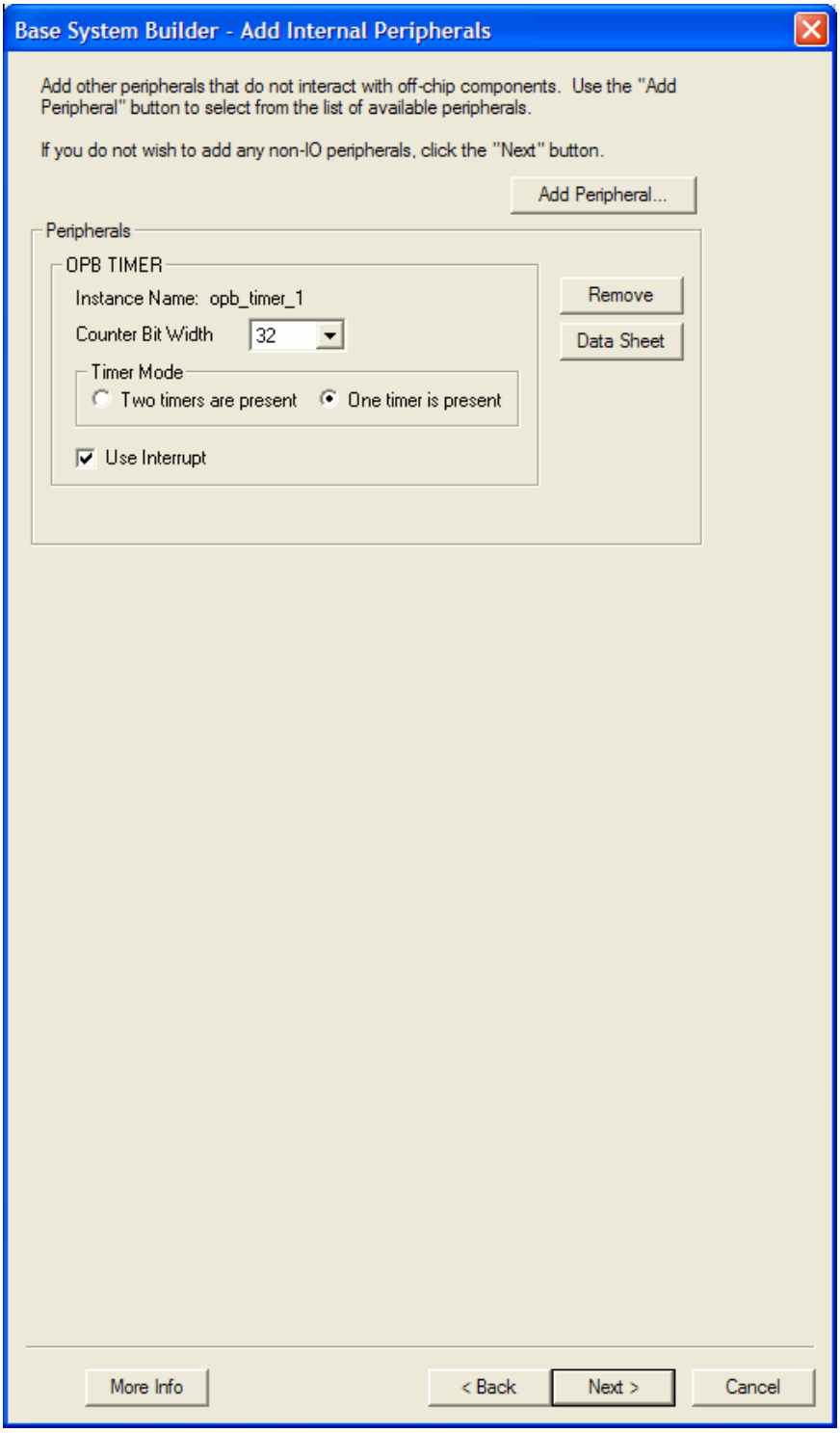

Figure 15 – Set Timer Parameters

17. Select both the Instruction and Data Cache sizes to be 8 KB. Select the check boxes for both ICache and DCache. Click **Next>**.

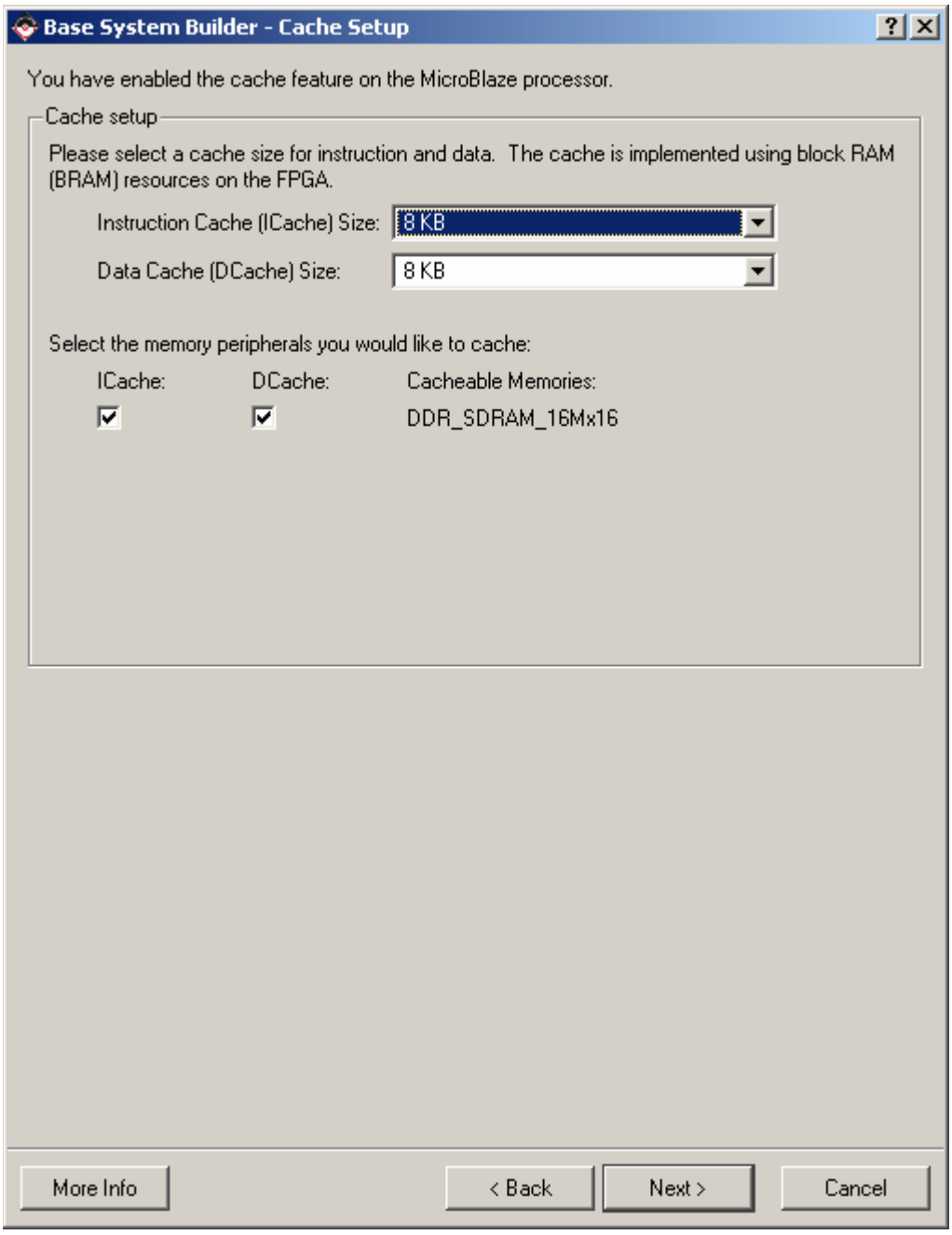

Figure 16 – Cache Setup

18. The default software configuration settings is used for STDIN and STDOUT. The two default sample applications are selected. Click **Next>**.

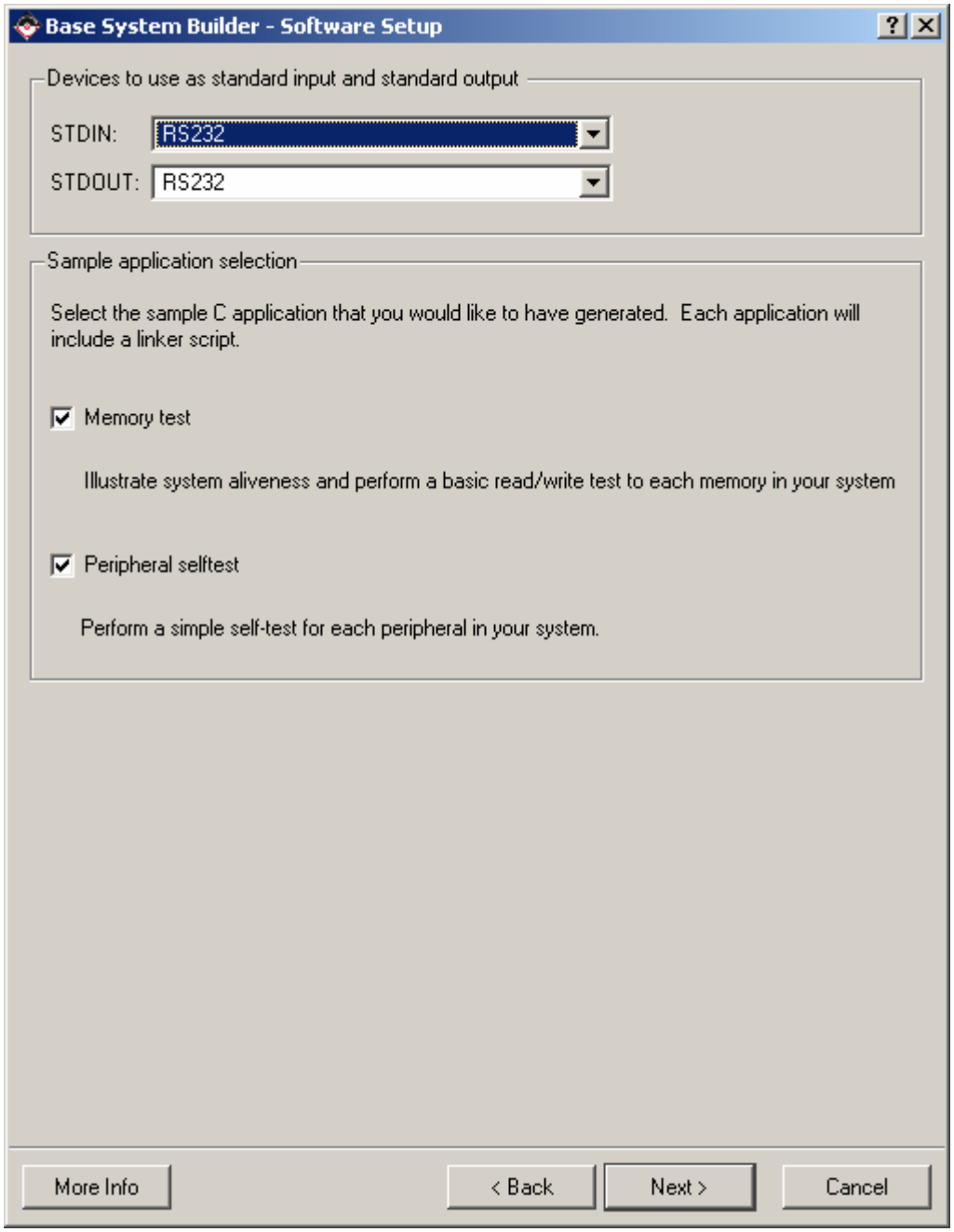

Figure 17 – BSB Software Setup

19. The Memory Test sample application code is designated to reside in the default location of local memory. Click **Next>**.

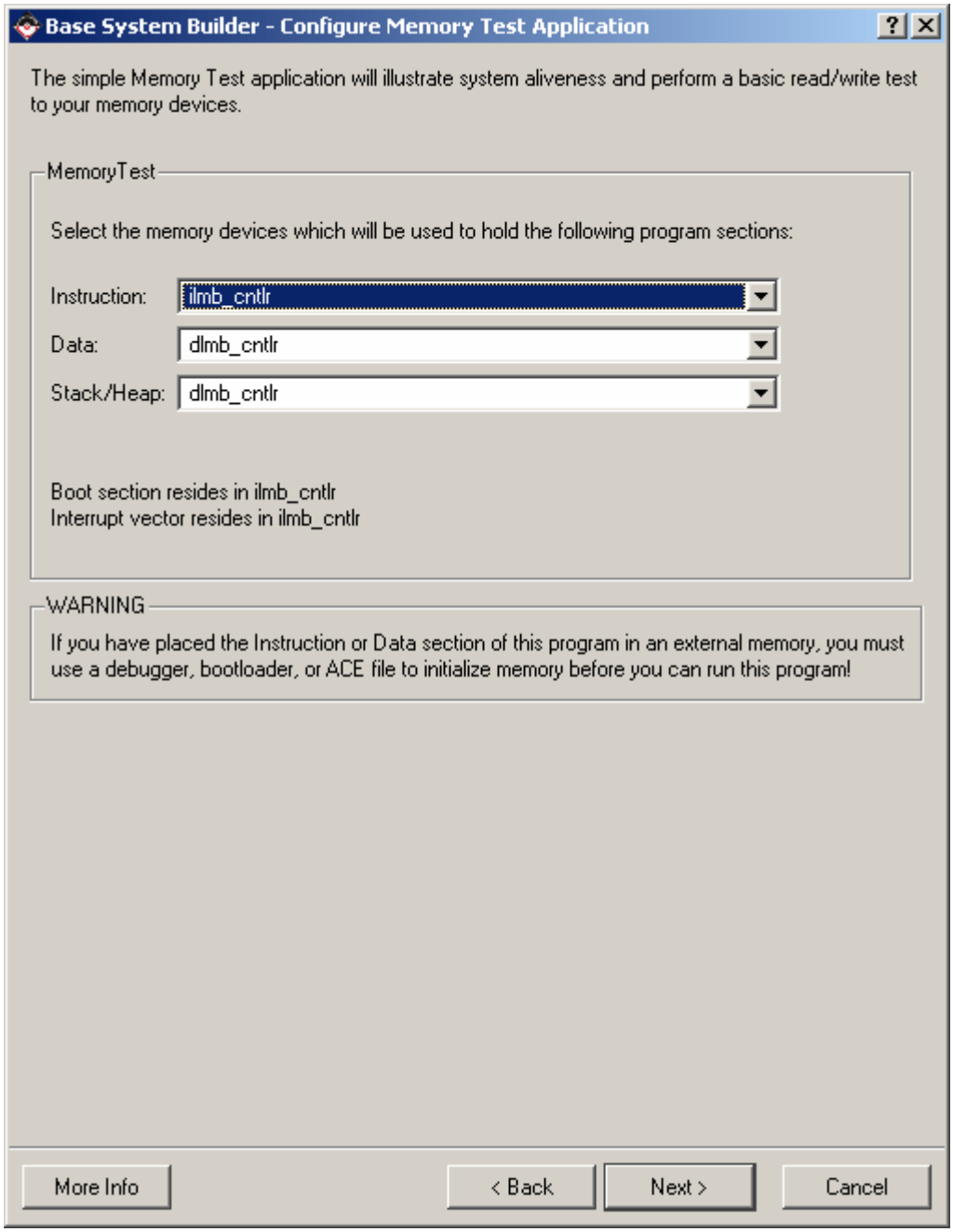

Figure 18 – BSB Memory Test Program Location

20. The Peripheral Test sample application code is designated to reside in the default location of DDR memory. Click **Next>**.

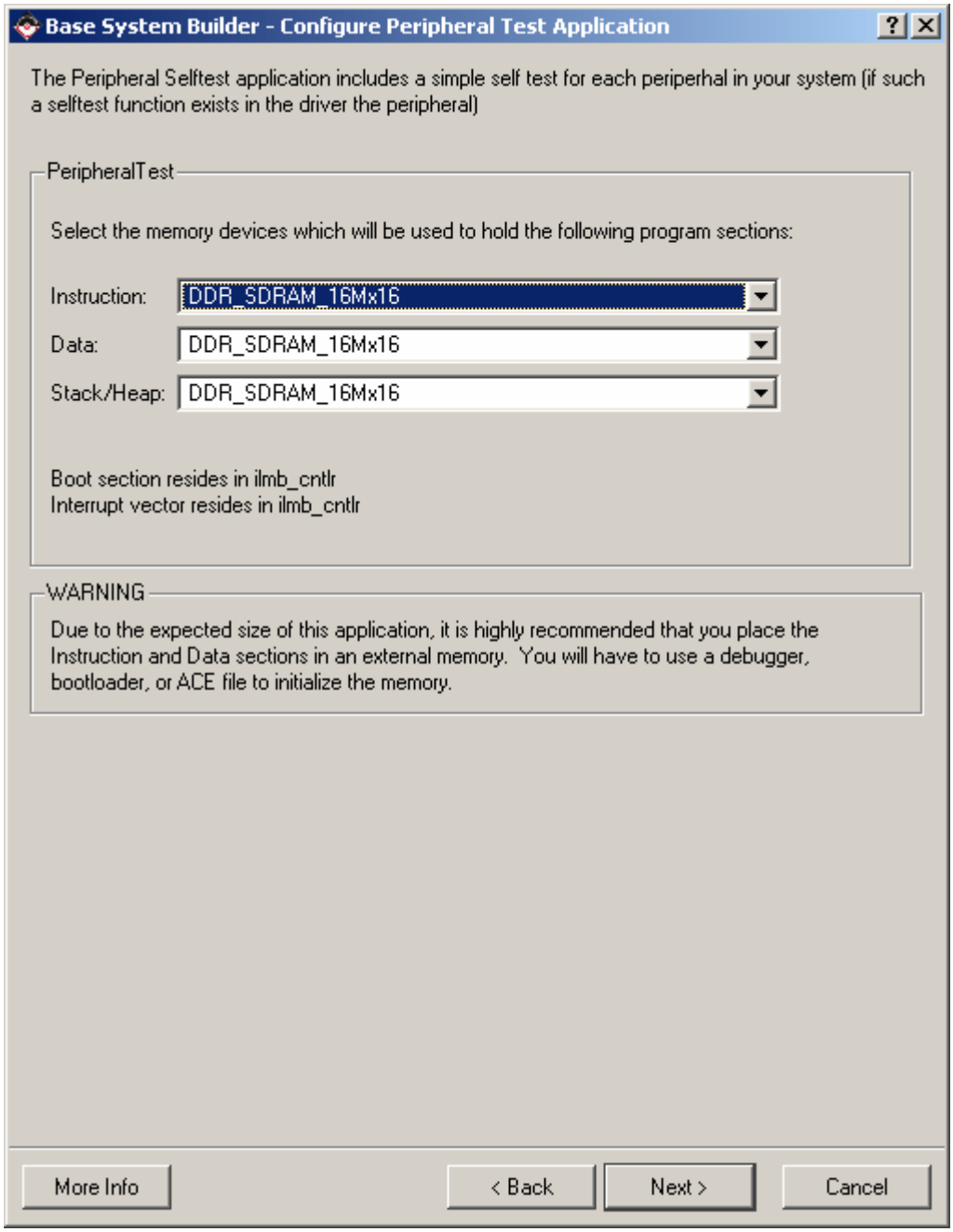

Figure 19 – BSB Peripheral Test Program Location

21. The BSB produces a summary of the created system, including the pre-selected hardware memory map. After reviewing, click **Generate**.

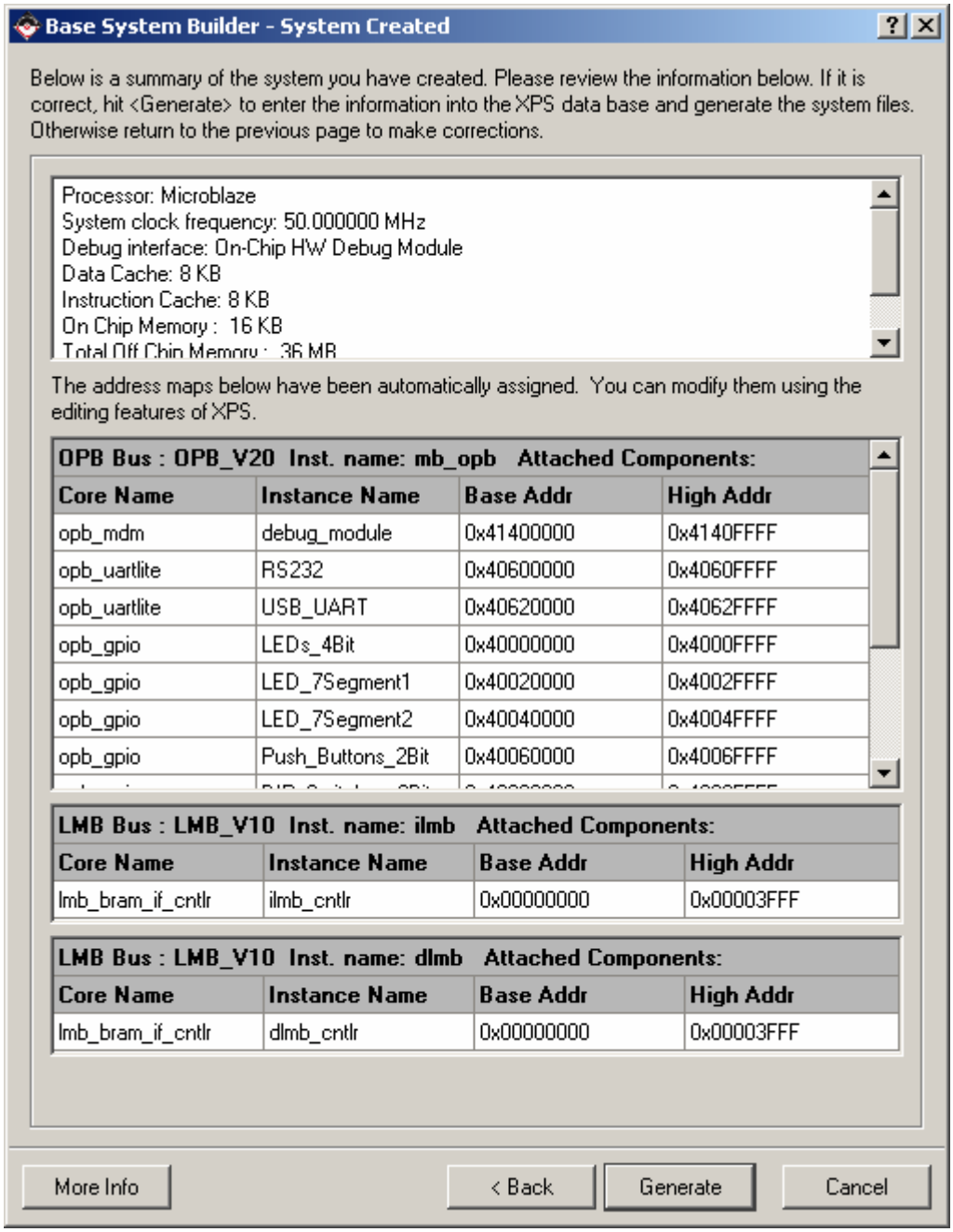

Figure 20 – BSB System Summary

22. Click **Finish**.

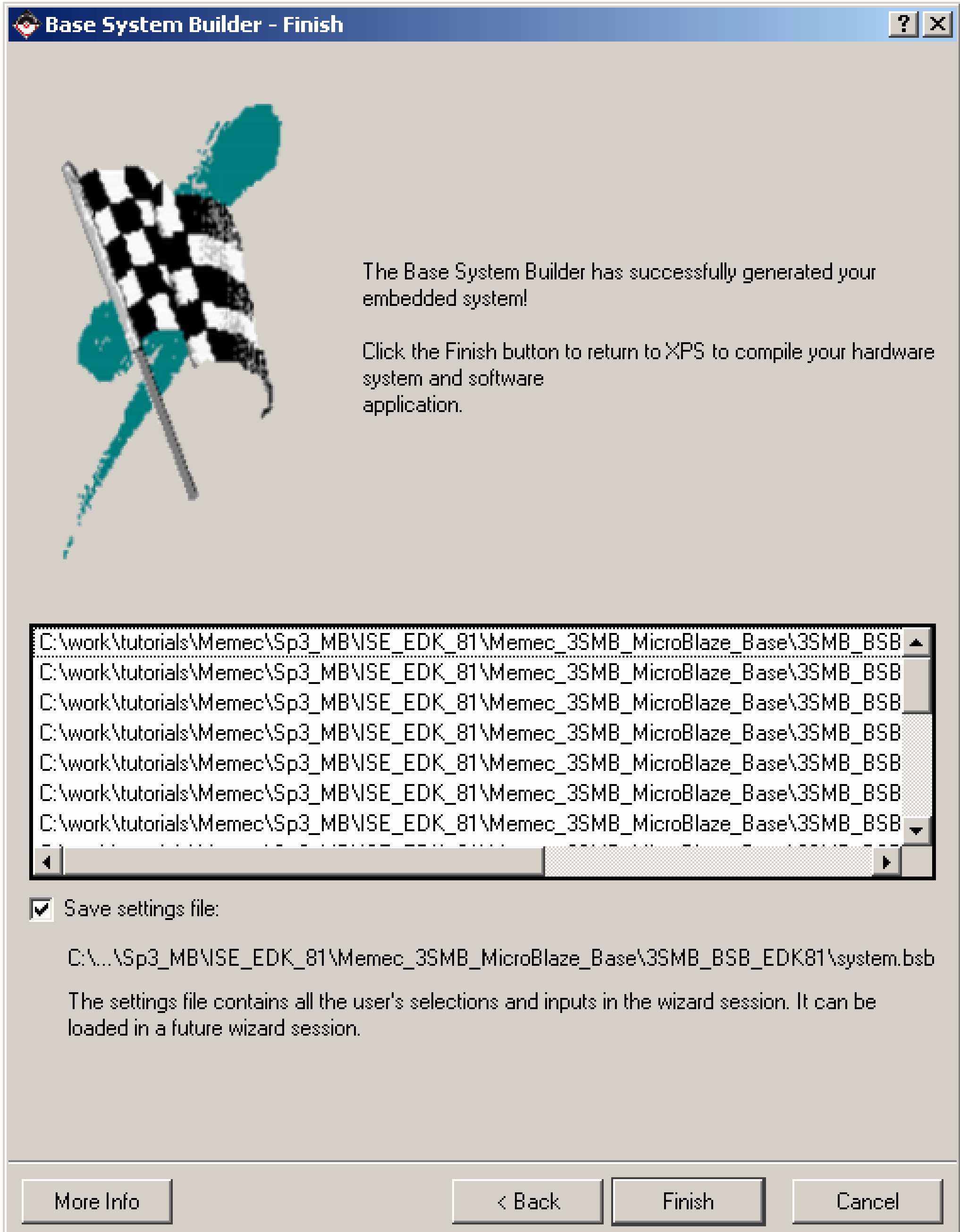

Figure 21 – BSB Finished

23. The BSB project is finished. XPS asks what you would like to do next. Select the radio button for *Start Using Platform Studio* and click **OK**.

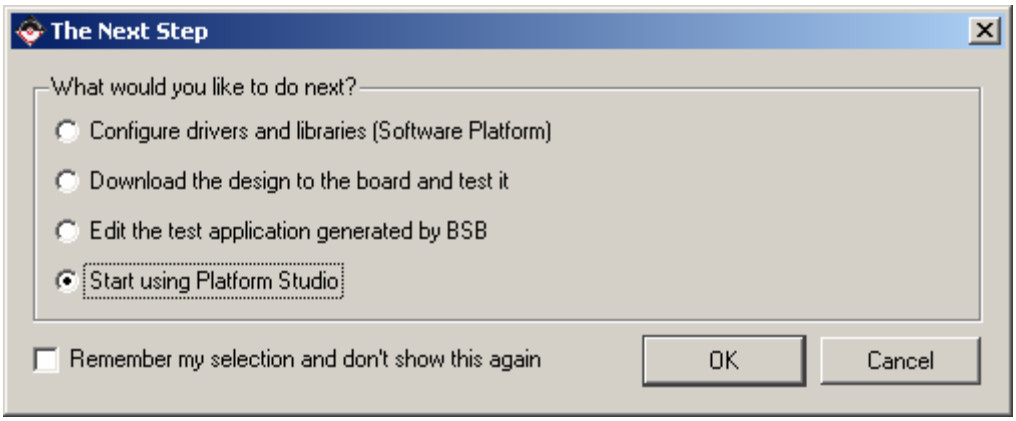

Figure 22 – The Next Step

The newly created system is now viewable in XPS, as shown in Figure 23. The BSB has created several hardware-related design elements, including: the hardware platform specification (MHS), the FPGA pin-mapping and constraints (UCF), and the iMPACT Command File for downloading to the FPGA.

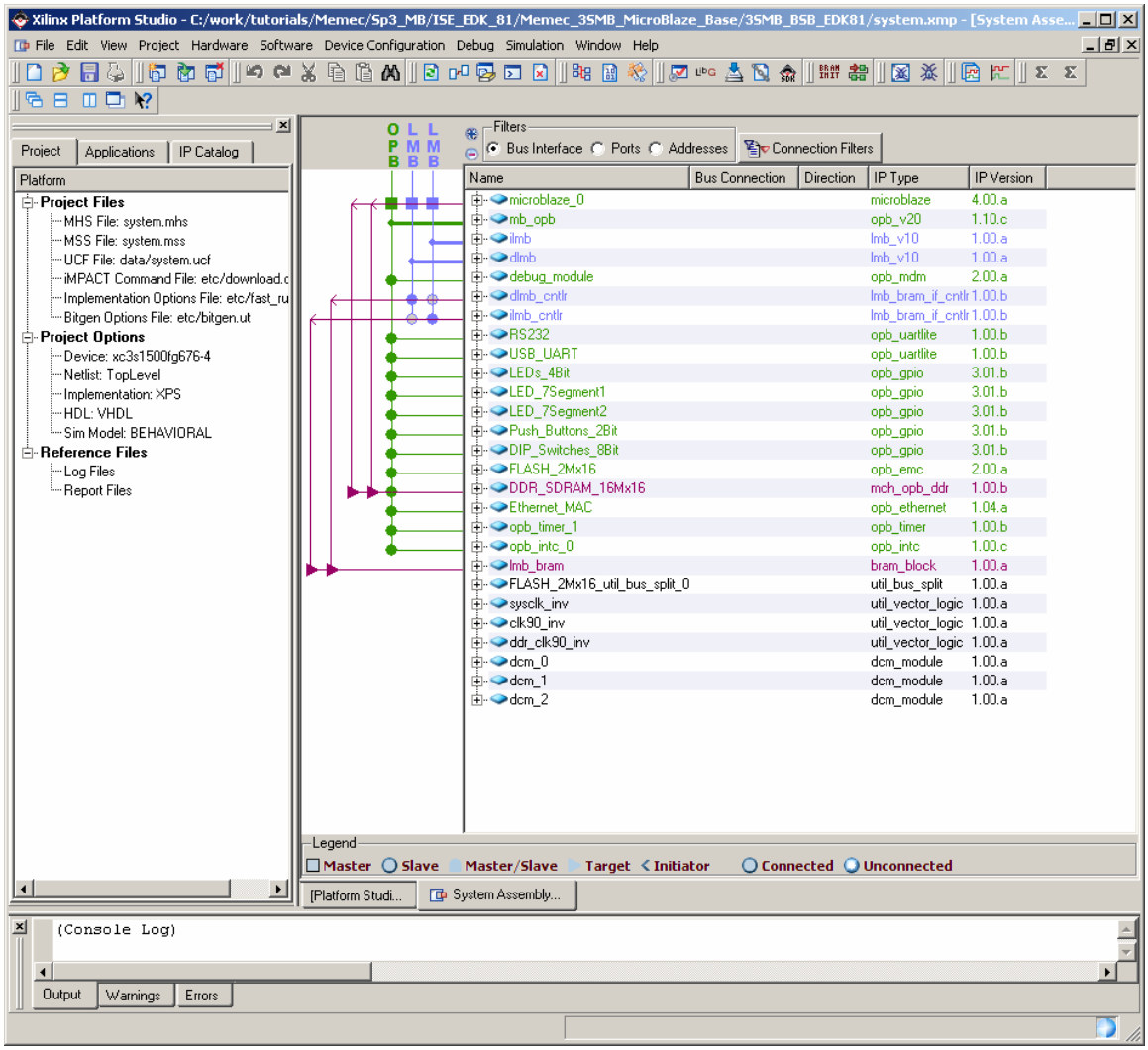

Figure 23 – BSB-Generated System in XPS

Several software elements have also been automatically created, such as the MSS and two test applications and linker scripts (as shown under the Applications tab (see Figure 24)).

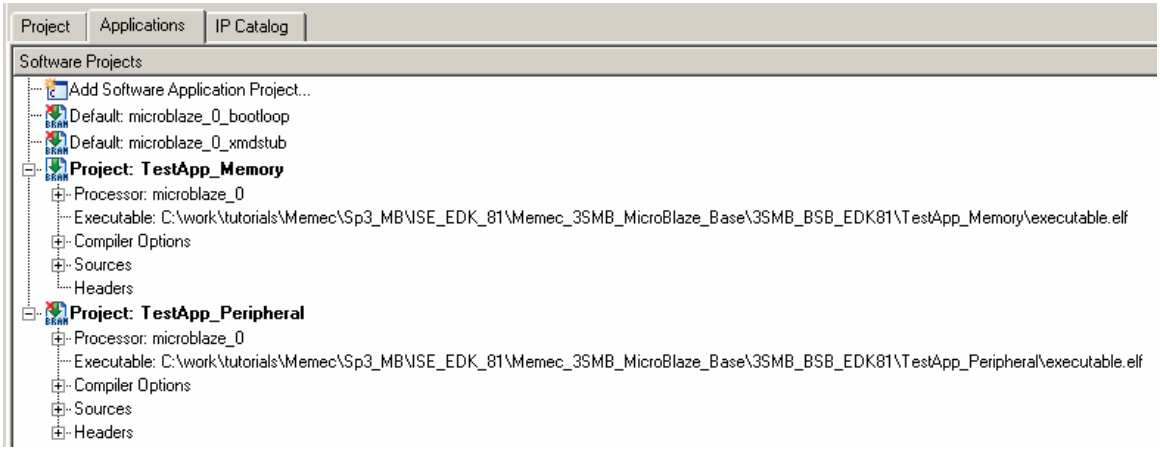

Figure 24 – XPS Applications Tab

The system is now ready to build, a process that typically takes 15-20 minutes but can take considerably longer depending on the complexity of the system and the processing capabilities of the host system. However, we will be making several more modifications and enhancements manually.

Modifications and Optimizations to the BSB Design

Although this BSB-generated system will build successfully, it will not meet the hardware timing requirements and the DDR SDRAM memory will not work. Additional modifications and optimizations are required to fully utilize this system.

MHS Changes

Clocks

The BSB is not capable of setting up the clocking system required for this design to work. The DDR must run at a minimum of 75 MHz. However, getting a full embedded processor system to run at 75 MHz in Spartan-3 is extremely difficult. Therefore, the DDR Controller is used in asynchronous mode with the DDR running at 75 MHz while the remainder of the system runs at 50 MHz.

24. Open the MHS file by double-clicking on **MHS File: system.mhs** under the *Project* tab.
25. In the MHS, replace the clock connections in the OPB_DDR instantiation (Lines 342-347) with the following lines:

```
 PORT Device_Clk90_in = ddr_dev_clk_90
 PORT Device_Clk90_in_n = ddr_dev_clk_270
 PORT Device_Clk = ddr_dev_clk_0
 PORT Device_Clk_n = ddr_dev_clk_180
 PORT DDR_Clk90_in = ddr_fb_clk_90
 PORT DDR_Clk90_in_n = ddr_fb_clk_270
```

26. In the MHS, replace the three util_vector_logic instantiations and the three dcm_module instantiations (Lines 409-486) with the following three util_vector_logic instantiations and two dcm_module instantiations. This preserves the 50 MHz processor clock, but creates a 75 MHz clock that is used as the device clock. The feedback clock is also set up as an 75 MHz clock.

```
BEGIN util_vector_logic
 PARAMETER INSTANCE = sysclk_inv
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE = 1
 PARAMETER C_OPERATION = not
 PORT Op1 = ddr_dev_clk_0
 PORT Res = ddr_dev_clk_180
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = clk90_inv
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE = 1
 PARAMETER C_OPERATION = not
 PORT Op1 = ddr_dev_clk_90
 PORT Res = ddr_dev_clk_270
```

```
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = ddr_clk90_inv
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE = 1
 PARAMETER C_OPERATION = not
 PORT Op1 = ddr_fb_clk_90
 PORT Res = ddr_fb_clk_270
END

BEGIN dcm_module
 PARAMETER INSTANCE = dcm_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_CLK0_BUF = TRUE
 PARAMETER C_CLK90_BUF = TRUE
 PARAMETER C_CLKIN_PERIOD = 13.333333
 PARAMETER C_CLK_FEEDBACK = 1X
 PARAMETER C_EXT_RESET_HIGH = 1
 PARAMETER C_CLKFX_BUF = TRUE
 PARAMETER C_CLKFX_DIVIDE = 3
 PARAMETER C_CLKFX_MULTIPLY = 2
 PORT CLKFX = sys_clk_s
 PORT CLKIN = dcm_clk_s
 PORT CLK0 = ddr_dev_clk_0
 PORT CLK90 = ddr_dev_clk_90
 PORT CLKFB = ddr_dev_clk_0
 PORT RST = net_gnd
 PORT LOCKED = dcm_0_lock
END

BEGIN dcm_module
 PARAMETER INSTANCE = dcm_1
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_CLK0_BUF = TRUE
 PARAMETER C_CLK90_BUF = TRUE
 PARAMETER C_CLKIN_PERIOD = 13.333333
 PARAMETER C_CLK_FEEDBACK = 1X
 PARAMETER C_EXT_RESET_HIGH = 0
 PARAMETER C_DESKEW_ADJUST = SOURCE_SYNCHRONOUS
 PORT CLKIN = ddr_feedback_s
 PORT CLK90 = ddr_fb_clk_90
 PORT CLK0 = dcm_1_FB
 PORT CLKFB = dcm_1_FB
 PORT RST = dcm_0_lock
 PORT LOCKED = dcm_1_lock
END
```

FSL Accelerated Download Channel

To additionally add the FSL accelerated download link to the MDM, follow these additional steps.

27. Add the following parameter and bus_interface to the MicroBlaze instance:

```
PARAMETER C_FSL_LINKS = 1
BUS_INTERFACE SFSL0 = fsl_v20_0
```

28. Add the following parameter and bus_interface to the MDM:

```
PARAMETER C_WRITE_FSL_PORTS = 1
BUS_INTERFACE MFSL0 = fsl_v20_0
```

29. Add the FSL bus:

```
BEGIN fsl_v20
 PARAMETER INSTANCE = fsl_v20_0
 PARAMETER HW_VER = 2.00.a
 PARAMETER C_EXT_RESET_HIGH = 0
 PORT FSL_Clk = sys_clk_s
 PORT SYS_Rst = sys_rst_s
END
```

30. Save and close the MHS.

UCF Changes

Constraints are added to give period and phase information to each clock. Detailed I/O timing statements are added. Ignore statements are also placed on unrelated clock domains and resets.

31. Open the UCF by double-clicking on **UCF File: data/system.ucf** under the *Project* tab.
32. Remove the following lines from the UCF (we'll constrain all the clocks in the next step):

```
Net sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 20000 ps;
```

33. Add the following lines to the UCF to fully constrain all clock phases in the design (note that several lines below wrap to the next line in this document even though they are one continuous line of text):

```
Net sys_clk_s            TNM_NET = OPB_CLK;
Net ddr_dev_clk_0        TNM_NET = DDR_DEVICE_CLOCK_PHASE_0;
Net ddr_dev_clk_90       TNM_NET = DDR_DEVICE_CLOCK_PHASE_90;
Net ddr_dev_clk_180      TNM_NET = DDR_DEVICE_CLOCK_PHASE_180;
Net ddr_dev_clk_270      TNM_NET = DDR_DEVICE_CLOCK_PHASE_270;
Net ddr_fb_clk_90        TNM_NET = DDR_FEEDBACK_CLOCK_PHASE_90;
Net ddr_fb_clk_270       TNM_NET = DDR_FEEDBACK_CLOCK_PHASE_270;

TIMESPEC "TS_FIFO_WRITE" = FROM "OPB_CLK" TO "DDR_DEVICE_CLOCK_PHASE_0"  TIG;
TIMESPEC "TS_FIFO_READ"  = FROM "DDR_DEVICE_CLOCK_PHASE_0" TO "OPB_CLK"  TIG;

TIMESPEC TS_OPB_CLK                     = PERIOD OPB_CLK                     20000 ps;
TIMESPEC TS_DDR_DEVICE_CLOCK_PHASE_0    = PERIOD DDR_DEVICE_CLOCK_PHASE_0    13332 ps;
```

```
TIMESPEC TS_DDR_DEVICE_CLOCK_PHASE_90    = PERIOD DDR_DEVICE_CLOCK_PHASE_90
TS_DDR_DEVICE_CLOCK_PHASE_0 * 1 PHASE + 3333 ps;
TIMESPEC TS_DDR_DEVICE_CLOCK_PHASE_180   = PERIOD DDR_DEVICE_CLOCK_PHASE_180
TS_DDR_DEVICE_CLOCK_PHASE_0 * 1 PHASE + 6666 ps;
TIMESPEC TS_DDR_DEVICE_CLOCK_PHASE_270   = PERIOD DDR_DEVICE_CLOCK_PHASE_270
TS_DDR_DEVICE_CLOCK_PHASE_0 * 1 PHASE + 10000 ps;
TIMESPEC TS_DDR_FEEDBACK_CLOCK_PHASE_90  = PERIOD DDR_FEEDBACK_CLOCK_PHASE_90  13332 ps;
TIMESPEC TS_DDR_FEEDBACK_CLOCK_PHASE_270 = PERIOD DDR_FEEDBACK_CLOCK_PHASE_270
TS_DDR_FEEDBACK_CLOCK_PHASE_90 * 1 PHASE + 6666 ps;
```

34. Save and close the UCF.

Implementation Options

The default implementations options should achieve timing. No changes are required.

Build the Hardware Platform

The hardware platform can now be built. The hardware specification is used to construct and synthesize the hardware platform, after which the Xilinx implementation tools perform place and route.

35. Select **Hardware → Generate Bitstream**.

Code Changes

While the hardware is building, the code can be modified. The default BSB code can be modified to test more than the first 4K of memory.

36. Under the Applications tab, open the TestApp_Memory source code by double-clicking on TestApp_Memory.c

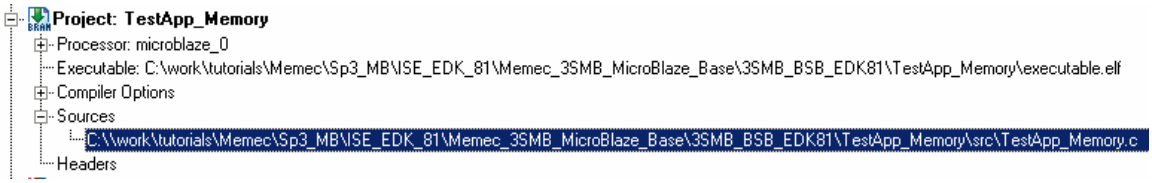

Figure 25 – TestApp_Memory.c Source Code

37. Add the following to the top of the code to properly define the size of the DDR SDRAM memory:

```
#define DDRBYTES     (XPAR_DDR_SDRAM_16MX16_MEM0_HIGHADDR - XPAR_DDR_SDRAM_16MX16_MEM0_BASEADDR + 1)
#define DDRHWORDS    (DDRBYTES / 2)
#define DDRWORDS     (DDRBYTES / 4)
```

38. The entire DDR can be tested by doing the following.

- Replace “1024” in the 32-bit test with “DDRWORDS”
- Replace “2048” in the 16-bit test with “DDRHWORDS"
- Replace “4096” in the 8-bit test with “DDRBYTES”

39. Save and close the code.

No changes are necessary for the TestApp_Peripheral application.

Compile the Applications

After the hardware has finished building, the software platform (Board Support Package) must be built and the applications must be compiled.

40. Select **Software → Build All User Applications**

Verify the Results

41. Verify the hardware timing results by looking at the PAR results, either in the XPS console window or in the `implementation/system.par` file. All timing constraints should pass.
42. Verify the two applications compiled without warning or error.
43. To run the Memory Test, open a HyperTerminal set to 115200 baud. Next connect power, JTAG cable, and RS232 to the board. Now select **Device Configuration → Download Bitstream.** Results are shown below. Note that if you change the code to test the entire 32 MB of DDR, the entire test will take about 80-90 seconds to run. The 32-bit test takes approximately 10-15 seconds to complete, the 16-bit test takes 20-30 seconds, and the 8-bit test takes 40-50 seconds.

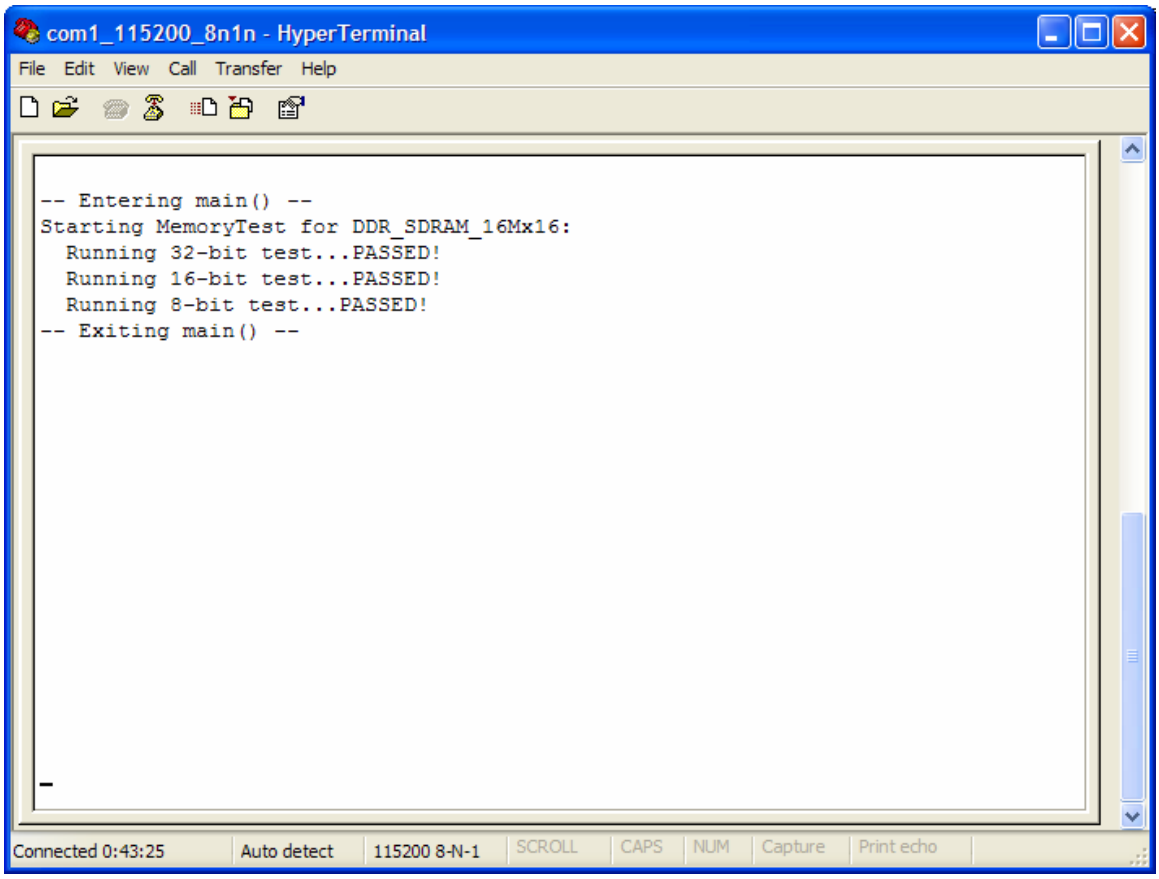

Figure 26 – TestApp_Memory Results for 3SMB Board

The TestApp_Peripheral project is targeted to execute from DDR memory. Therefore, XMD must be used to bootload the code into the memory prior to launching. This could be done using the GDB debugger, but, in this case, is done directly in XMD.

44. To run the Peripheral Test (after previously running the Memory Test to download the hardware platform), launch XMD: **Debug → Launch XMD**. A dialog box will remind you to set the options. Select OK, then **Save** the default options. After saving, XMD will automatically launch and connect to the MicroBlaze processor.

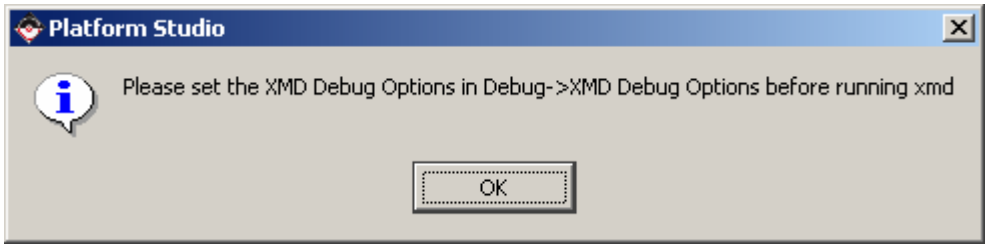

Figure 27 – XMD Debug Options Must Be Set

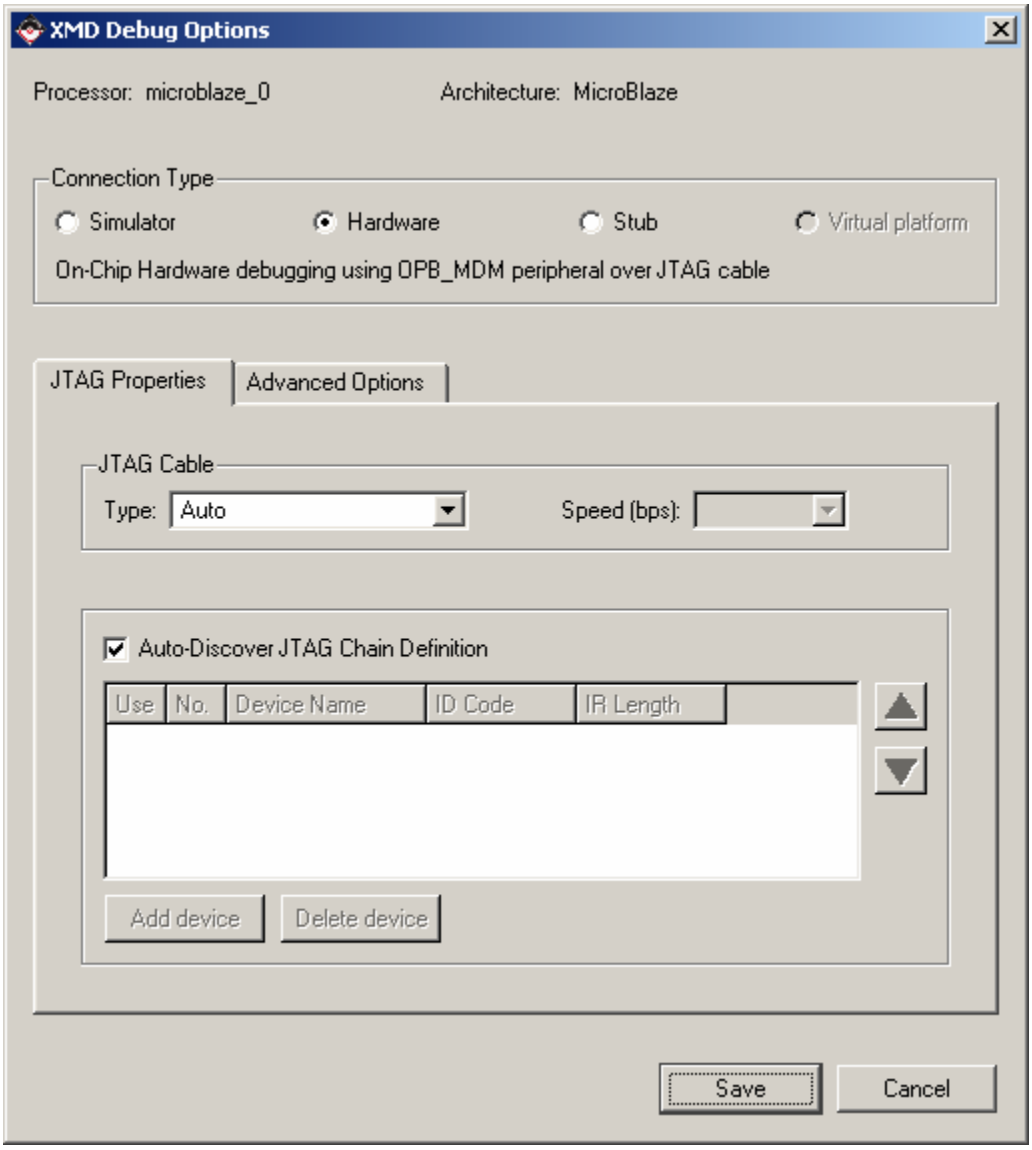

Figure 28 – Default XMD Debug Options

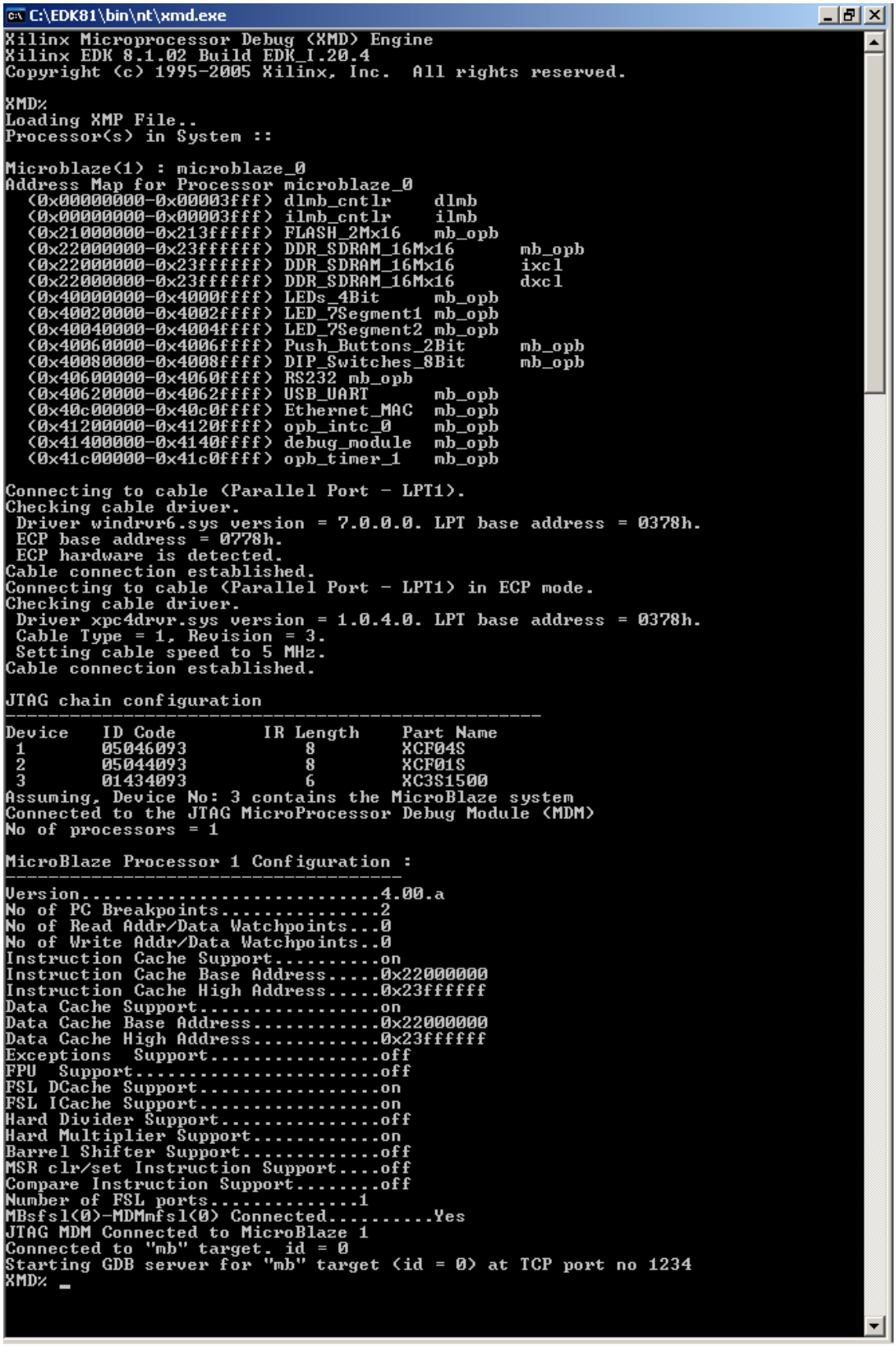

Figure 29 – XMD Launched and Connected

45. In XMD, type the following commands

```
dow TestApp_Peripheral/executable.elf
con
```

The Peripheral Test tests the various peripherals, including flashing the LEDs and 7-segment displays and reading the push buttons and dip switches.

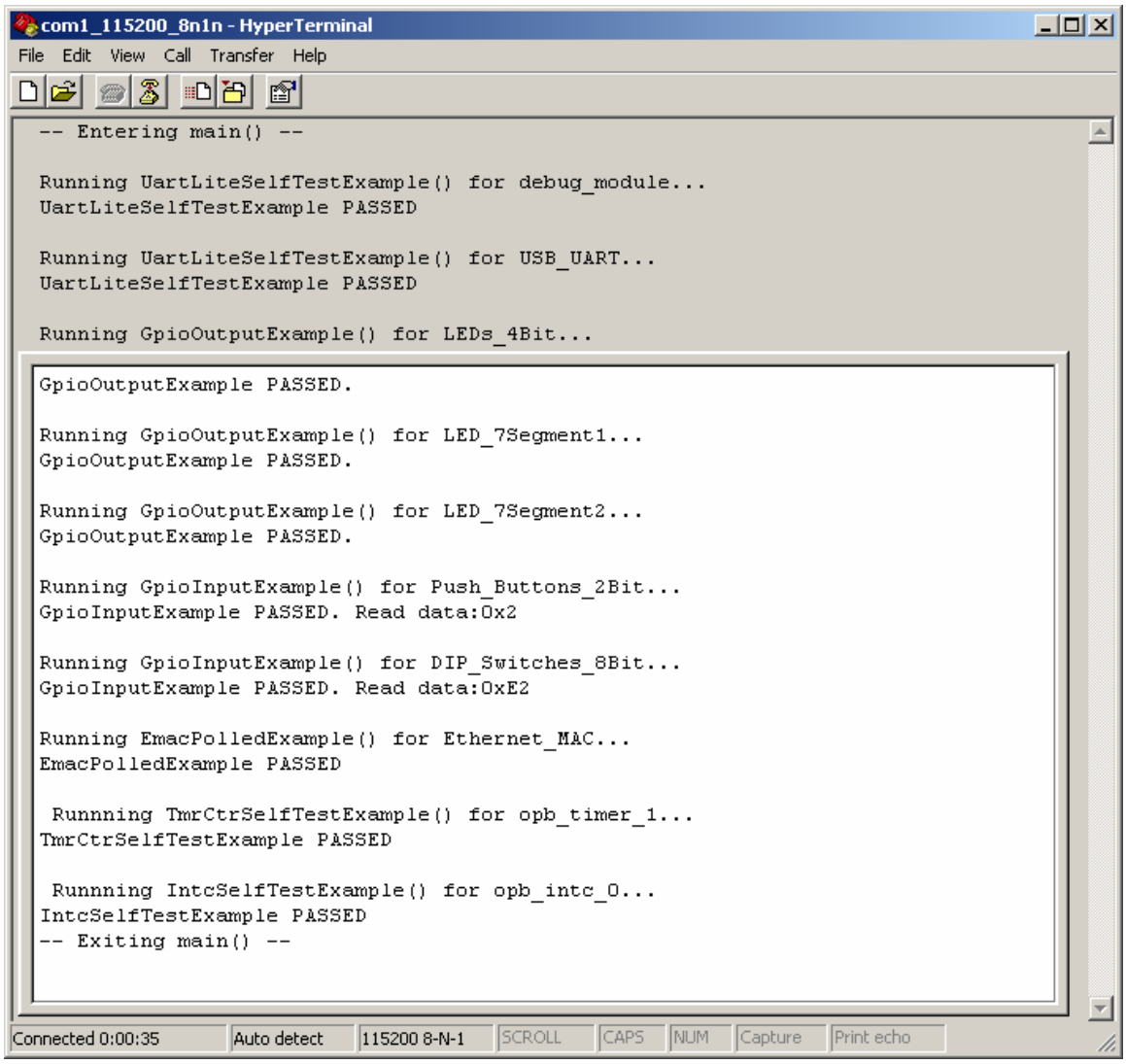

Figure 30 – TestApp_Peripheral Results for 3SMB Board

Additional reference designs which use this platform are:

Memec 3SMB MicroBlaze Using Memory Reference Designs
Memec 3SMB MicroBlaze WebServer Reference Design

Revision History

Date	Version	Revision
12/07/04	6.3	Initial Memec release. (bhf)
12/20/04	6.3a	Updated to 6.3a XBD; added bootloader (bhf)
09/01/05	7.1	Updated to 7.1; changed to Async OPB_DDR (bhf)
06/29/06	8.1	Updated to 8.1; added FSL cache MCH_OPB_DDR (bhf)