Xilinx EDK 9.2 Introduction - Tutorial 4

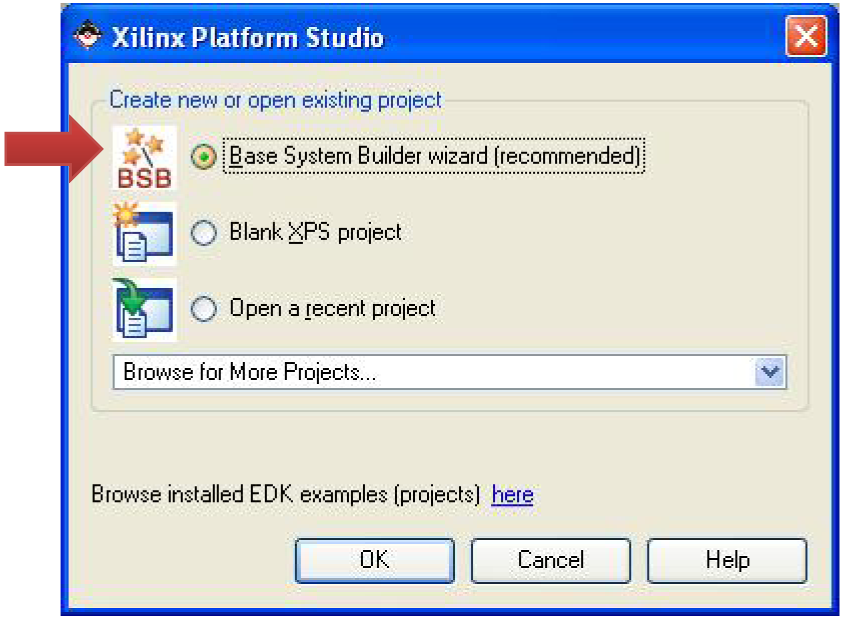
Creating design with use of MicroBlaze

Writing simple processing application in C for MicroBlaze

Overview of Tutorial 4

- Create an EDK project.
- Write sample C program that performs 8x8 Matrix multiplication and outputs to serial port.
- Compile the project and load it to the target platform.
- Verify operation with the output on to the serial terminal application.

Creation of Xilinx EDK Project



Xilinx Platform Studio

Create new or open existing project

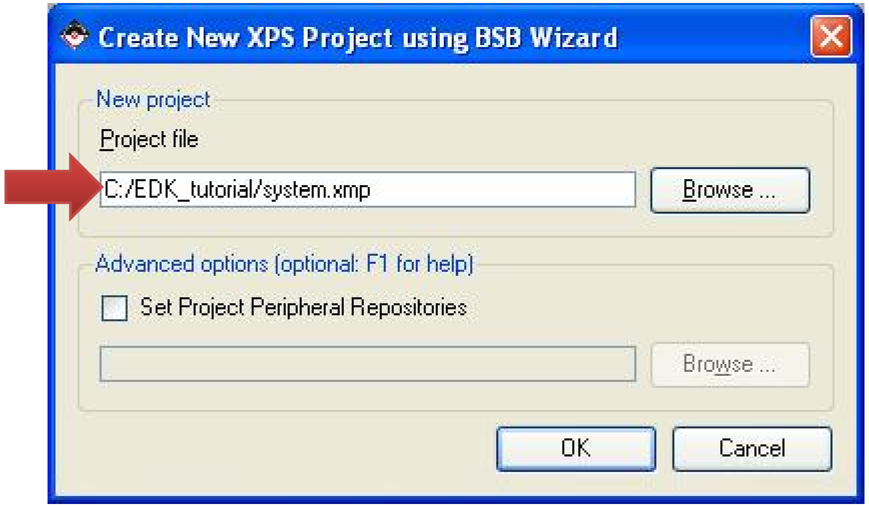
● Base System Builder wizard (recommended)

○ Blank XPS project

○ Open a recent project

Browse for More Projects...

Browse installed EDK examples (projects) here



Create New XPS Project using BSB Wizard

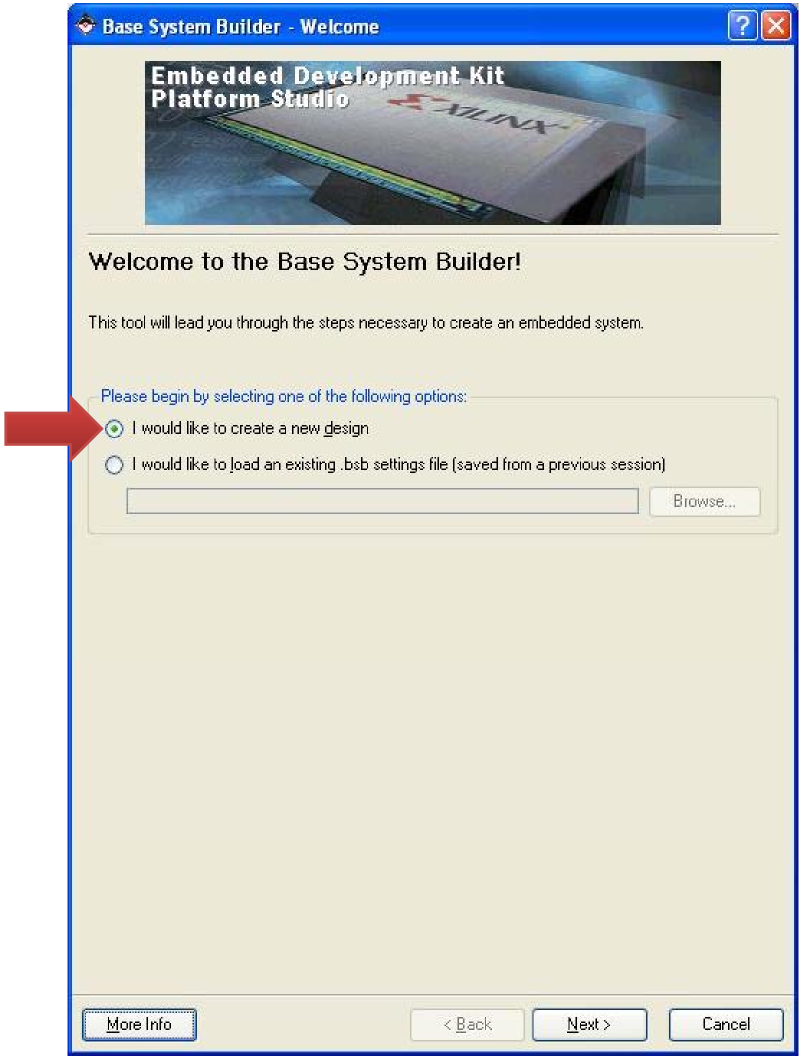
New project

Project file: C:/EDK_tutorial/system.xmp

Advanced options (optional: F1 for help)

□ Set Project Peripheral Repositories

Load Xilinx Platform Studio by clicking on ***Xilinx EDK 9.2.***
Select ***Base System Builder wizard*** and press OK.
Specify the directory for your project.



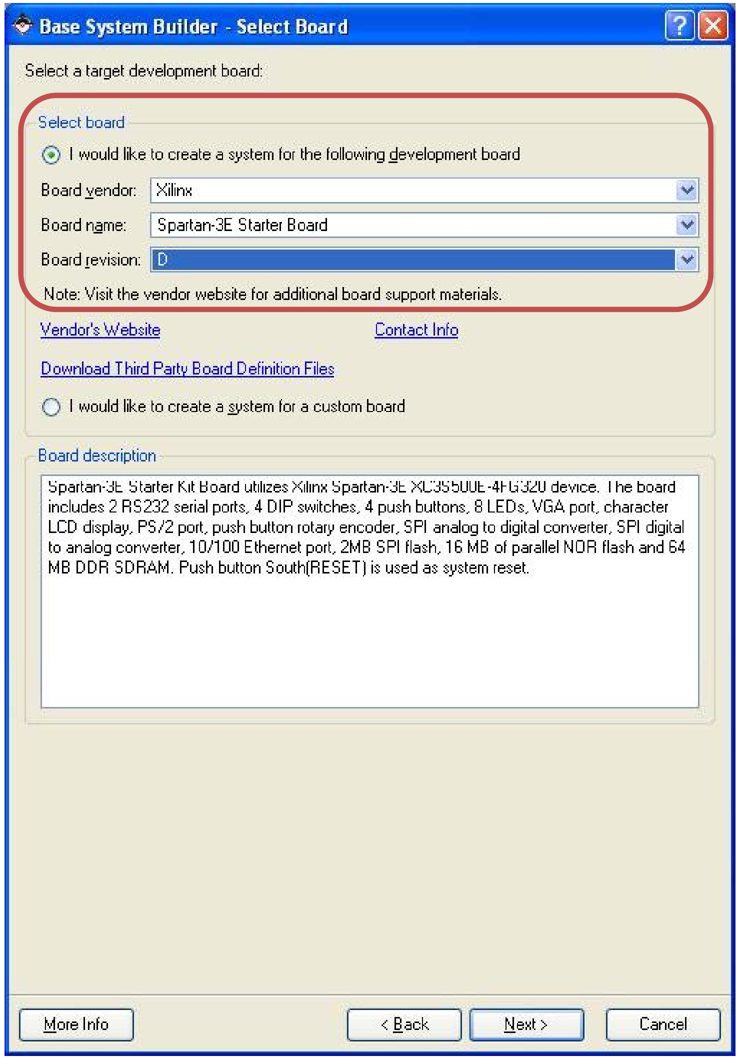
Welcome to the Base System Builder!

This tool will lead you through the steps necessary to create an embedded system.

Please begin by selecting one of the following options:

● I would like to create a new design

○ I would like to load an existing .bsb settings file (saved from a previous session)


Base System Builder - Select Board

Select a target development board:

Select board

● I would like to create a system for the following development board

Board vendor: Xilinx

Board name: Spartan-3E Starter Board

Board revision: D

Note: Visit the vendor website for additional board support materials.

Vendor's Website
Contact Info
Download Third Party Board Definition Files

○ I would like to create a system for a custom board

Board description

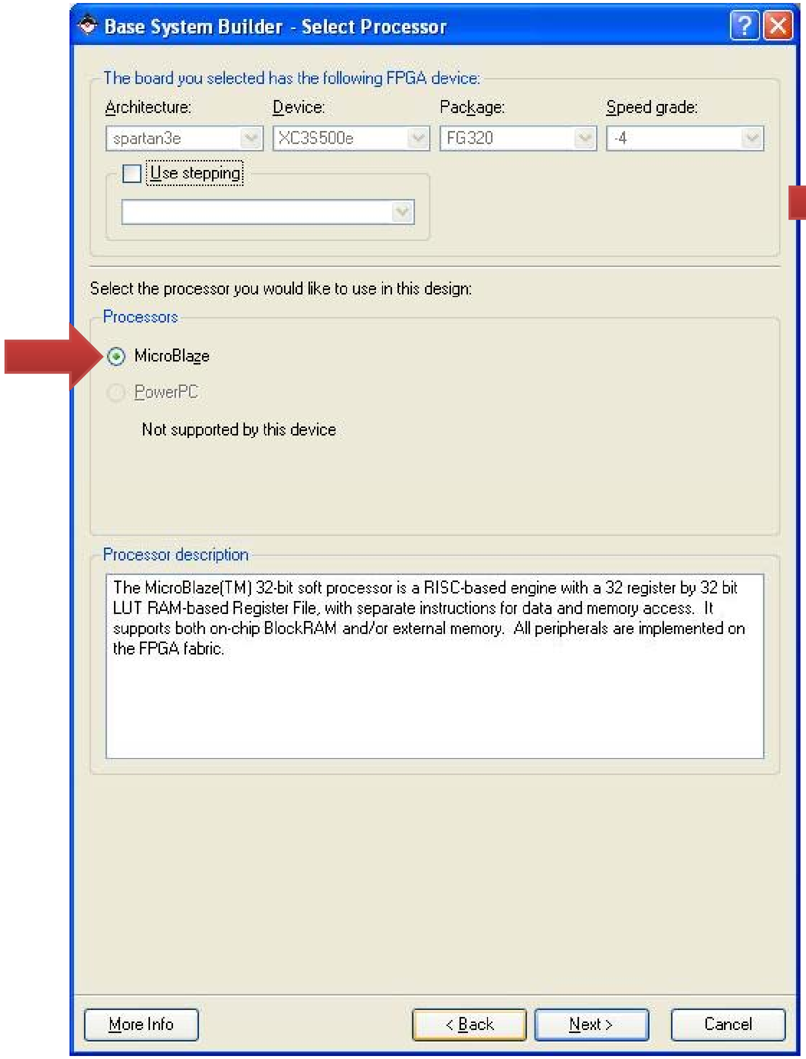
Spartan-3E Starter Kit Board utilizes Xilinx Spartan-3E XC3S500E-4FG320 device. The board includes 2 RS232 serial ports, 4 DIP switches, 4 push buttons, 8 LEDs, VGA port, character LCD display, PS/2 port, push button rotary encoder, SPI analog to digital converter, SPI digital to analog converter, 10/100 Ethernet port, 2MB SPI flash, 16 MB of parallel NOR flash and 64 MB DDR SDRAM. Push button South(RESET) is used as system reset.

Select ***I would like to create a new design*** and press ***Next>***

Select ***I would like to create a system for the following development board***

Specify Spartan 3E Starter Kit board parameters:

Xilinx | ***Spartan-3E Starter Board*** | ***D***



Base System Builder - Select Processor

The board you selected has the following FPGA device:

Architecture:	Device:	Package:	Speed grade:
spartan3e	XC3S500e	FG320	-4

□ Use stepping

Select the processor you would like to use in this design:

Processors

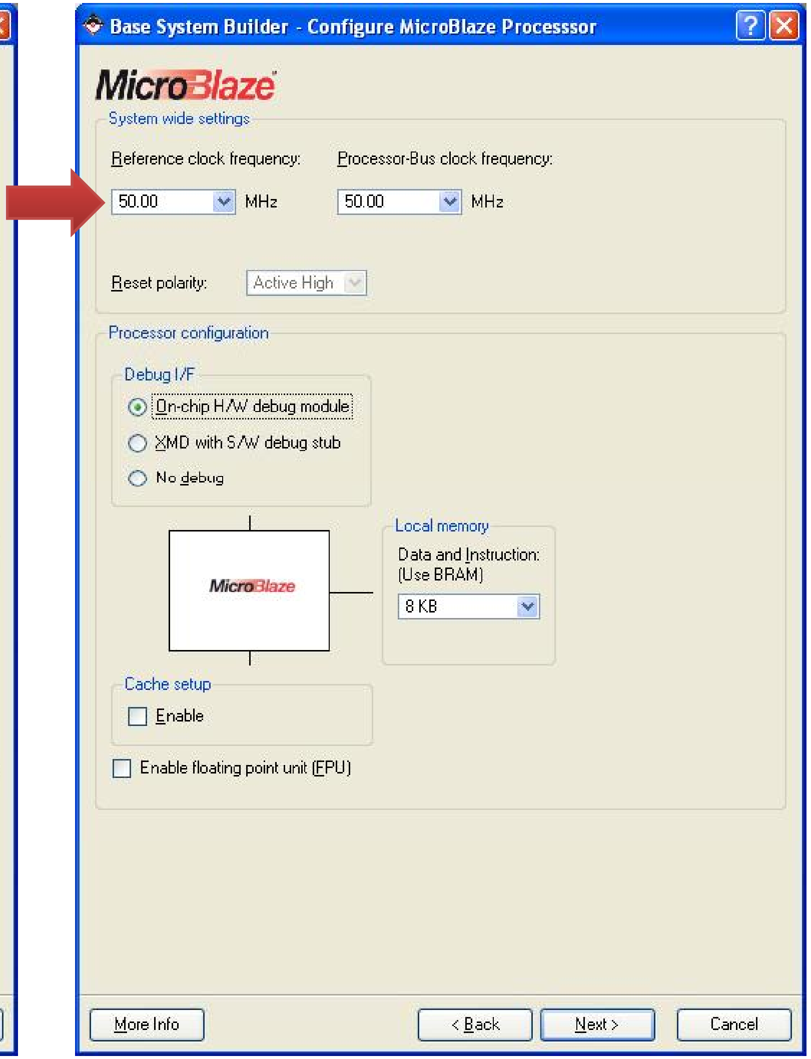
● MicroBlaze

○ PowerPC

Not supported by this device

Processor description

The MicroBlaze(TM) 32-bit soft processor is a RISC-based engine with a 32 register by 32 bit LUT RAM-based Register File, with separate instructions for data and memory access. It supports both on-chip BlockRAM and/or external memory. All peripherals are implemented on the FPGA fabric.



Base System Builder - Configure MicroBlaze Processsor

MicroBlaze

System wide settings

Reference clock frequency: 50.00 MHz

Processor-Bus clock frequency: 50.00 MHz

Reset polarity: Active High

Processor configuration

Debug I/F

● On-chip H/W debug module

○ XMD with S/W debug stub

○ No debug

Local memory

Data and Instruction: (Use BRAM) 8 KB

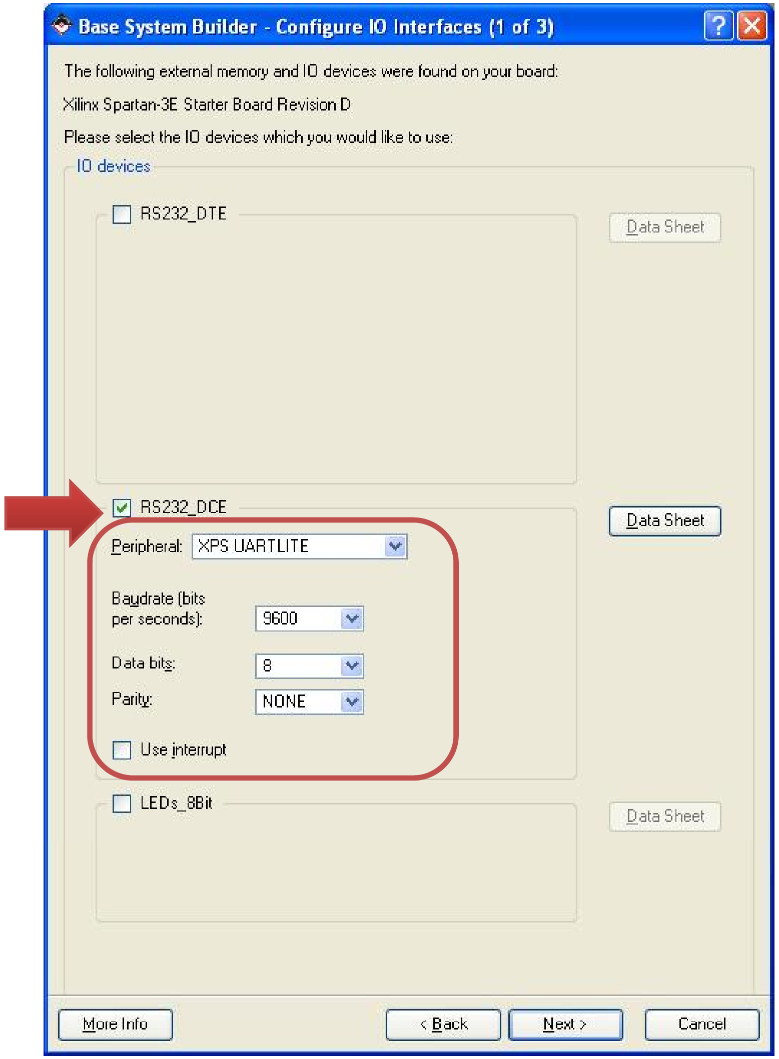
Cache setup

□ Enable

□ Enable floating point unit (FPU)

Since the platform has does not have PowerPC only option is MicroBlaze, therefore no modification is needed. Press ***Next>***

System clock is ***50 MHz***, and the rest of the parameters have to be left as is. Press ***Next>***



Base System Builder - Configure IO Interfaces (1 of 3)

The following external memory and IO devices were found on your board:

Xilinx Spartan-3E Starter Board Revision D

Please select the IO devices which you would like to use:

IO devices

□ RS232_DTE

☑ RS232_DCE

Peripheral: XPS UARTLITE

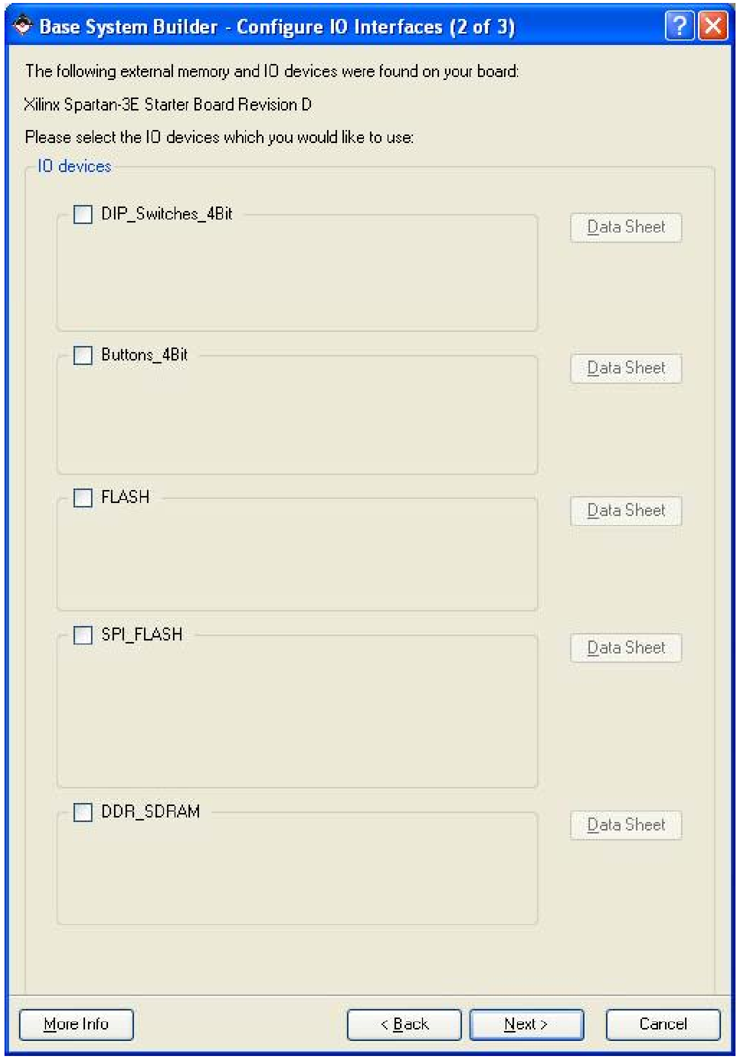
Baudrate (bits per seconds): 9600

Data bits: 8

Parity: NONE

□ Use interrupt

□ LEDs_8Bit



Base System Builder - Configure IO Interfaces (2 of 3)

The following external memory and IO devices were found on your board:

Xilinx Spartan-3E Starter Board Revision D

Please select the IO devices which you would like to use:

IO devices

□ DIP_Switches_4Bit

□ Buttons_4Bit

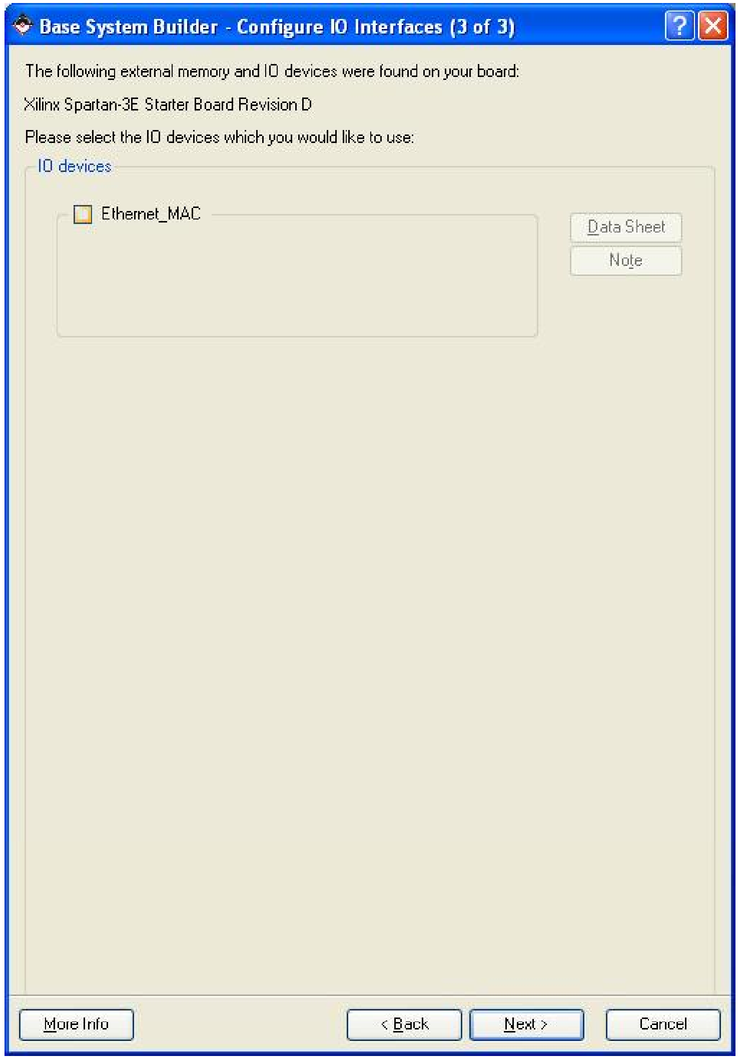
□ FLASH

□ SPI_FLASH

□ DDR_SDRAM

At this step interfaces are selected. For this project only ***RS232_DCE*** serial port is used. All of the other IO devices should be deselected.

Press ***Next>***

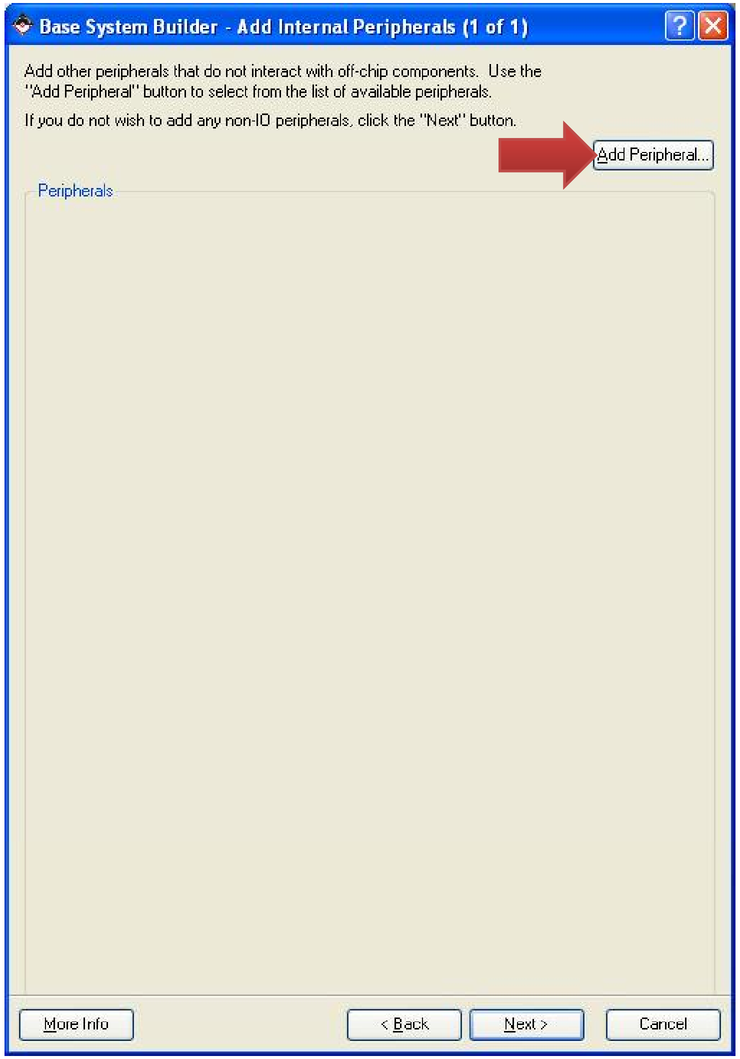


Base System Builder - Configure IO Interfaces (3 of 3)

The following external memory and IO devices were found on your board:

Xilinx Spartan-3E Starter Board Revision D

Please select the IO devices which you would like to use:

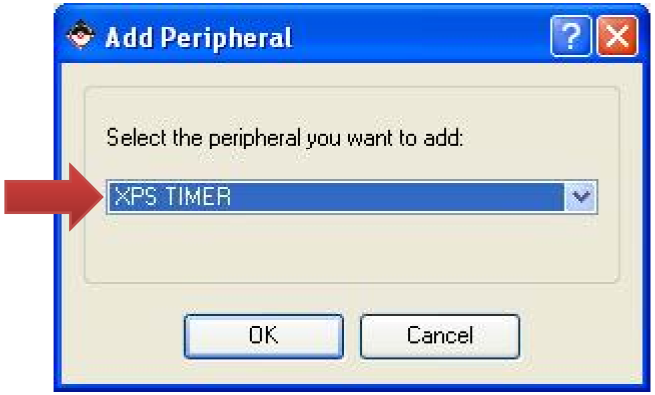


Base System Builder - Add Internal Peripherals (1 of 1)

Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

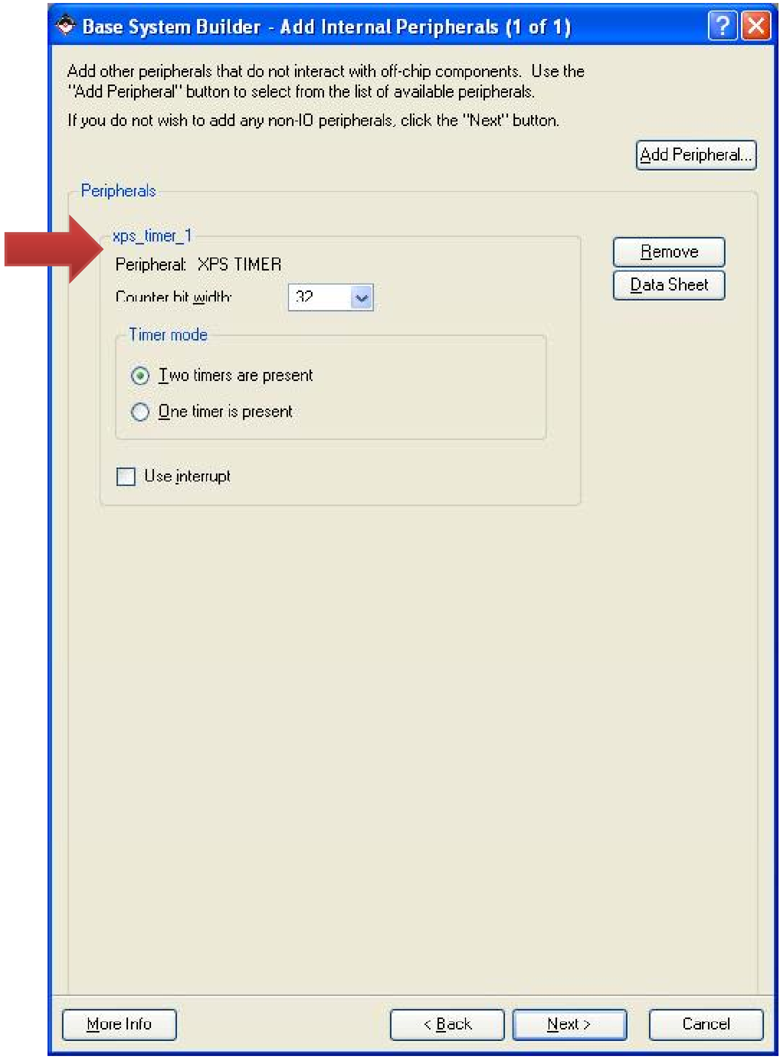
If you do not wish to add any non-IO peripherals, click the "Next" button.

Deselect ***Ethernet_MAC*** and press ***Next> .***
For this project we will also insert a timer internal peripheral. Timer will be used to count execution clock cycles. Press ***Add Peripheral*** to bring up the ***Add Peripheral*** window.



Select the peripheral you want to add:

XPS TIMER



Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

If you do not wish to add any non-IO peripherals, click the "Next" button.

Peripherals

xps_timer_1

Peripheral: XPS TIMER

Counter bit width: 32

Timer mode

● Two timers are present

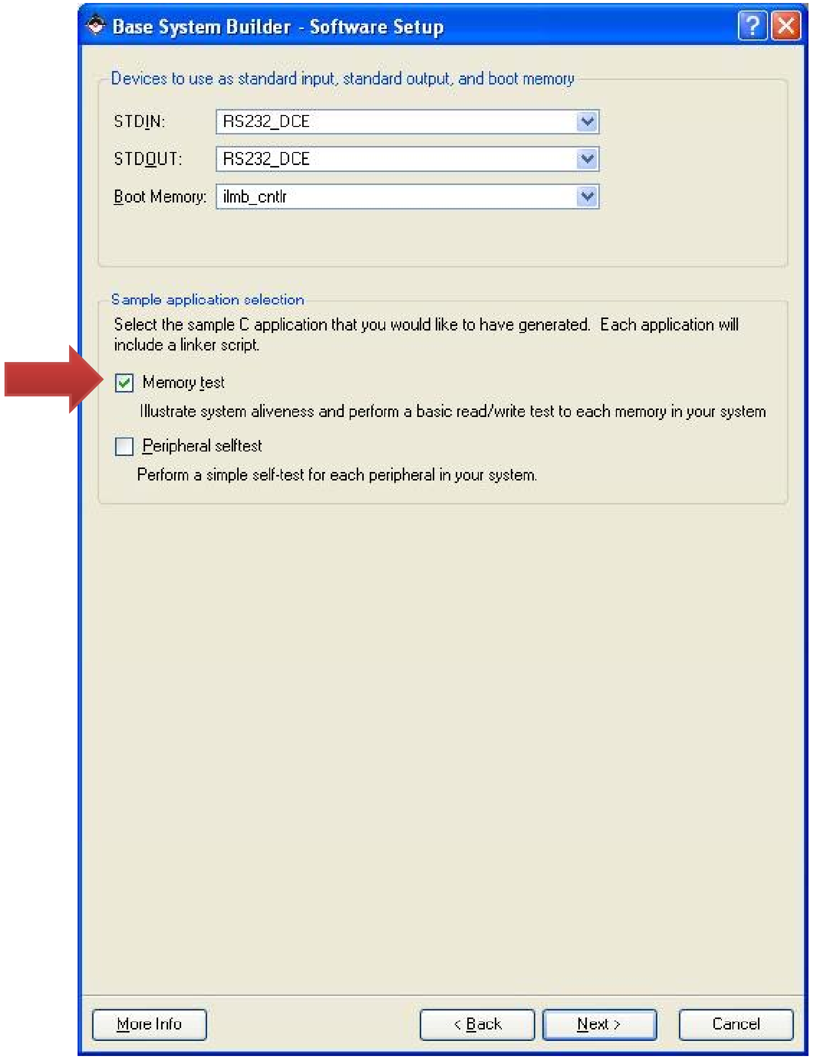
○ One timer is present

□ Use interrupt

Select ***XPS TIMER*** from the drop down menu and press OK**.**

A ***xps_timer_1*** peripheral will appear in the window. In this case leave the default parameters.

Press ***Next>***



Base System Builder - Software Setup

Devices to use as standard input, standard output, and boot memory

STDIN: RS232_DCE

STDOUT: RS232_DCE

Boot Memory: ilmb_cntlr

Sample application selection

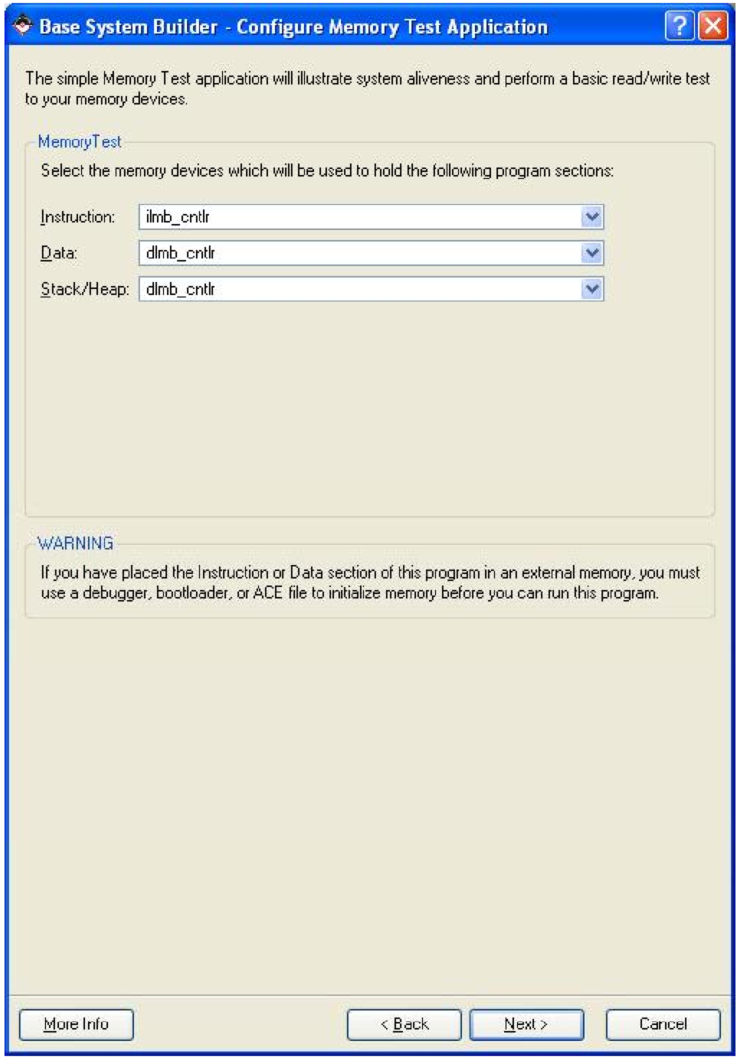
Select the sample C application that you would like to have generated. Each application will include a linker script.

☑ Memory test

Illustrate system aliveness and perform a basic read/write test to each memory in your system

□ Peripheral selftest

Perform a simple self-test for each peripheral in your system.



Base System Builder - Configure Memory Test Application

The simple Memory Test application will illustrate system aliveness and perform a basic read/write test to your memory devices.

MemoryTest

Select the memory devices which will be used to hold the following program sections:

Instruction: ilmb_cntlr

Data: dlmb_cntlr

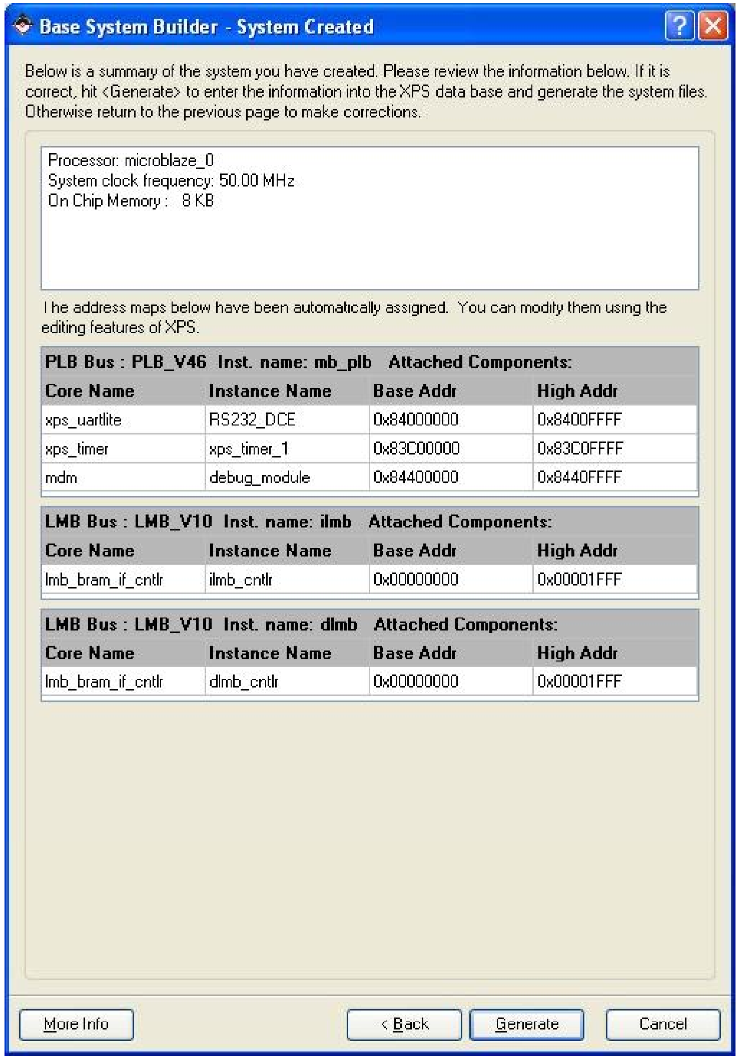
Stack/Heap: dlmb_cntlr

WARNING

If you have placed the Instruction or Data section of this program in an external memory, you must use a debugger, bootloader, or ACE file to initialize memory before you can run this program.

In sample application selection select ONLY ***Memory test.*** This will create a sample C program with all the appropriate header files and components. This sample program can be modified or removed later.

Press ***Next>*** for both of the screens.



Below is a summary of the system you have created. Please review the information below. If it is correct, hit <Generate> to enter the information into the XPS data base and generate the system files. Otherwise return to the previous page to make corrections.

Processor: microblaze_0
System clock frequency: 50.00 MHz
On Chip Memory : 8 KB

The address maps below have been automatically assigned. You can modify them using the editing features of XPS.

PLB Bus : PLB_V46 Inst. name: mb_plb Attached Components:

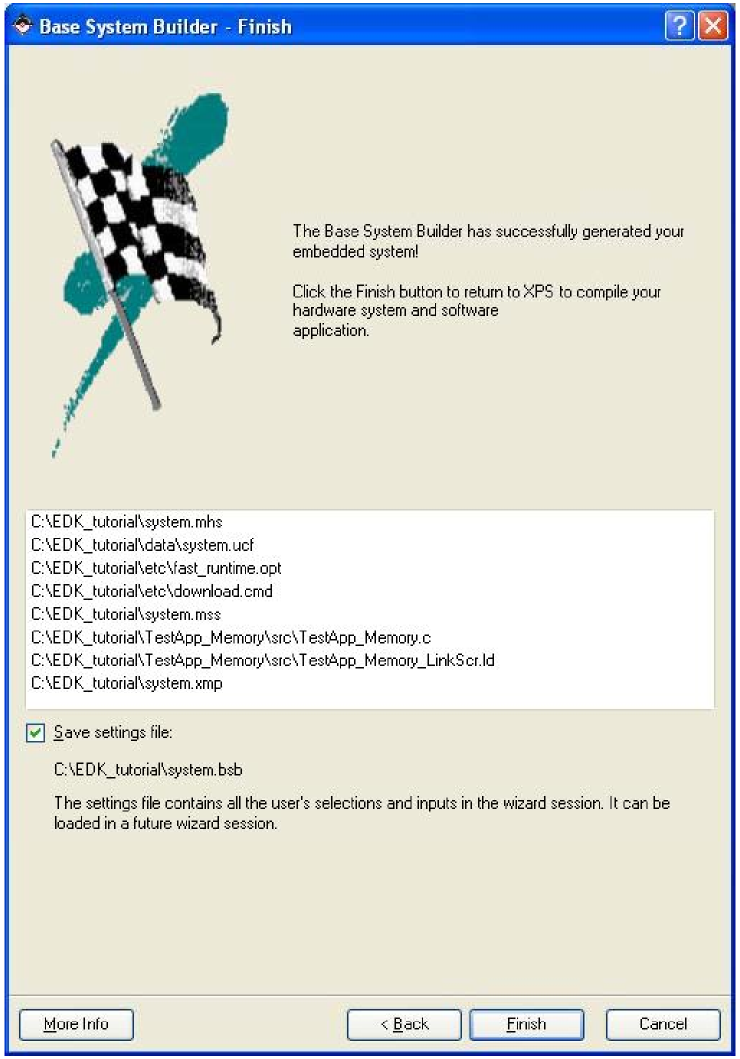
Core Name	Instance Name	Base Addr	High Addr
xps_uartlite	RS232_DCE	0x84000000	0x8400FFFF
xps_timer	xps_timer_1	0x83C00000	0x83C0FFFF
mdm	debug_module	0x84400000	0x8440FFFF

LMB Bus : LMB_V10 Inst. name: ilmb Attached Components:

Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00001FFF

LMB Bus : LMB_V10 Inst. name: dlmb Attached Components:

Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	dlmb_cntlr	0x00000000	0x00001FFF



The Base System Builder has successfully generated your embedded system!

Click the Finish button to return to XPS to compile your hardware system and software application.

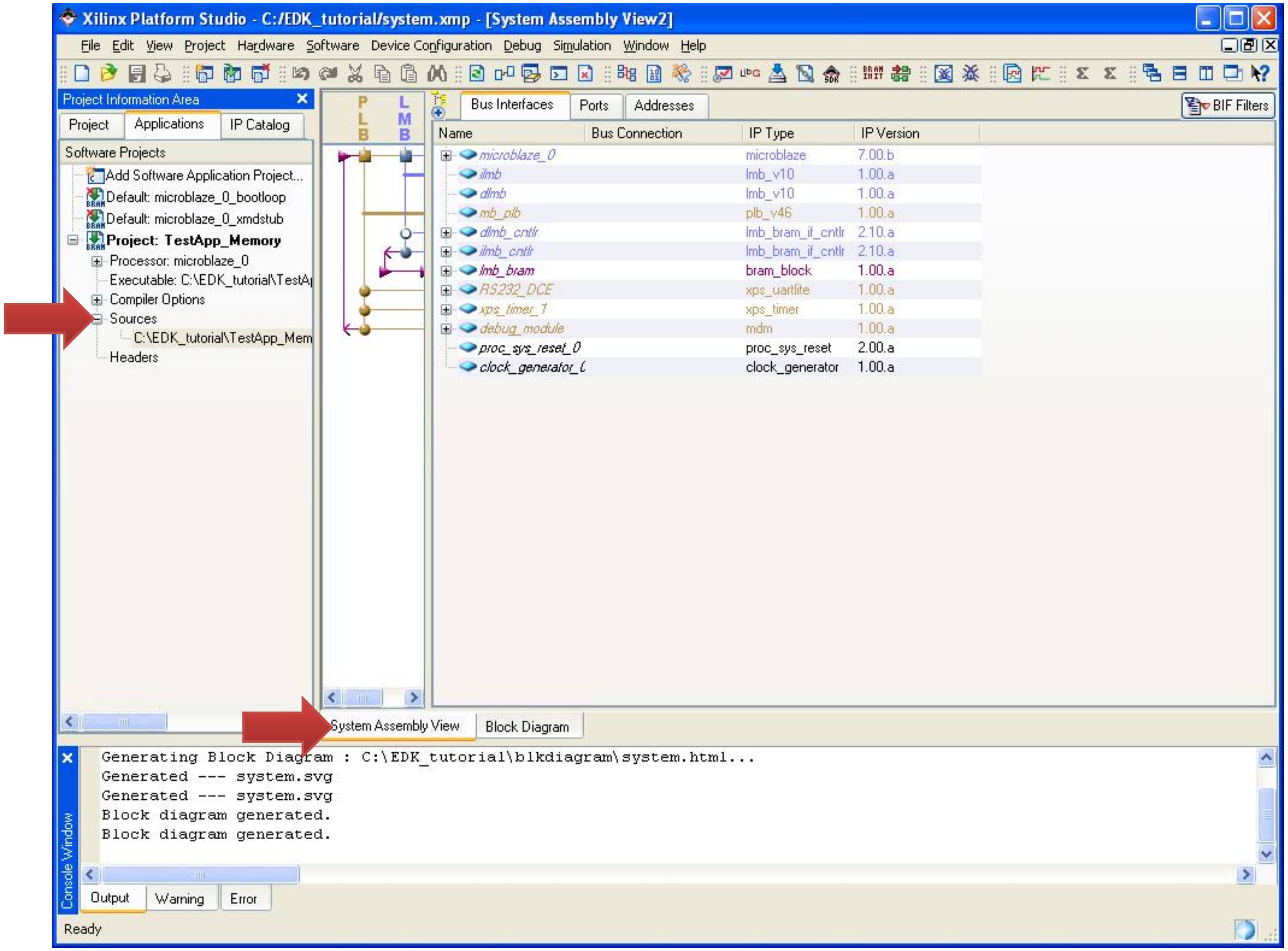
```
C:\EDK_tutorial\system.mhs
C:\EDK_tutorial\data\system.ucf
C:\EDK_tutorial\etc\fast_runtime.opt
C:\EDK_tutorial\etc\download.cmd
C:\EDK_tutorial\system.mss
C:\EDK_tutorial\TestApp_Memory\src\TestApp_Memory.c
C:\EDK_tutorial\TestApp_Memory\src\TestApp_Memory_LinkScr.ld
C:\EDK_tutorial\system.xmp
```

☑ Save settings file:

C:\EDK_tutorial\system.bsb

The settings file contains all the user's selections and inputs in the wizard session. It can be loaded in a future wizard session.

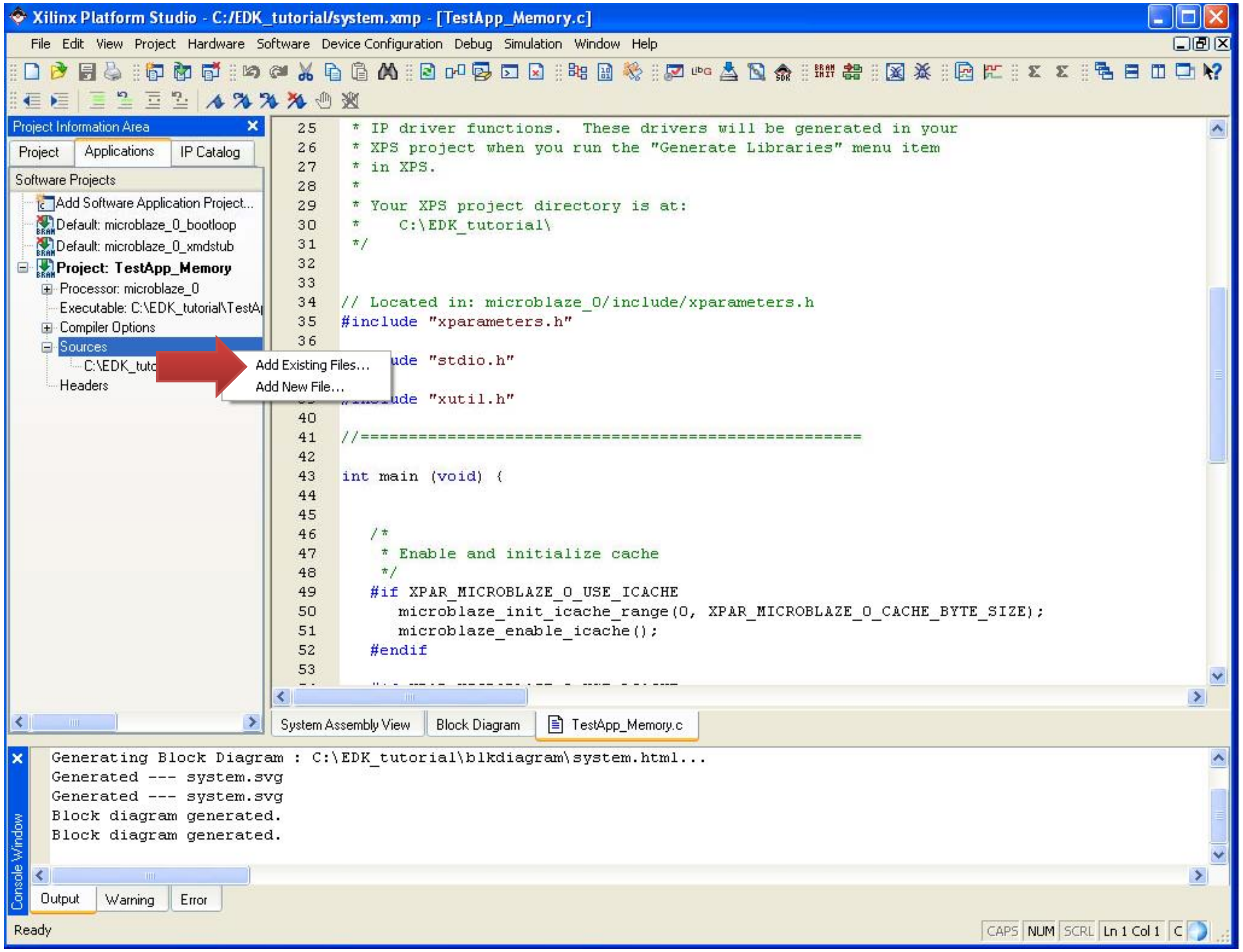
These windows show the base addresses of the actual peripheral devices. Based on these addresses these peripherals are accessed from the application. Press ***Generate*** to generate the necessary files. In the next window all generated files are listed. Press ***Finish***.


Xilinx Platform Studio - C:/EDK_tutorial/system.xmp - [System Assembly View2]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Information Area
Project
Applications
IP Catalog
Software Projects
Add Software Application Project...
Default: microblaze_0_bootloop
Default: microblaze_0_xmdstub
Project: TestApp_Memory
Processor: microblaze_0
Executable: C:\EDK_tutorial\TestAp
Compiler Options
Sources
C:\EDK_tutorial\TestApp_Mem
Headers
PLB
LMB
Bus Interfaces
Ports
Addresses
BIF Filters
System Assembly View
Block Diagram
Console Window
Output
Warning
Error
Ready

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.00.b
ilmb		lmb_v10	1.00.a
dlmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.00.a
dlmb_cntlr		lmb_bram_if_cntlr	2.10.a
ilmb_cntlr		lmb_bram_if_cntlr	2.10.a
lmb_bram		bram_block	1.00.a
RS232_DCE		xps_uartlite	1.00.a
xps_timer_1		xps_timer	1.00.a
debug_module		mdm	1.00.a
proc_sys_reset_0		proc_sys_reset	2.00.a
clock_generator_(		clock_generator	1.00.a

```
Generating Block Diagram : C:\EDK_tutorial\blkdiagram\system.html...
Generated --- system.svg
Generated --- system.svg
Block diagram generated.
Block diagram generated.
```

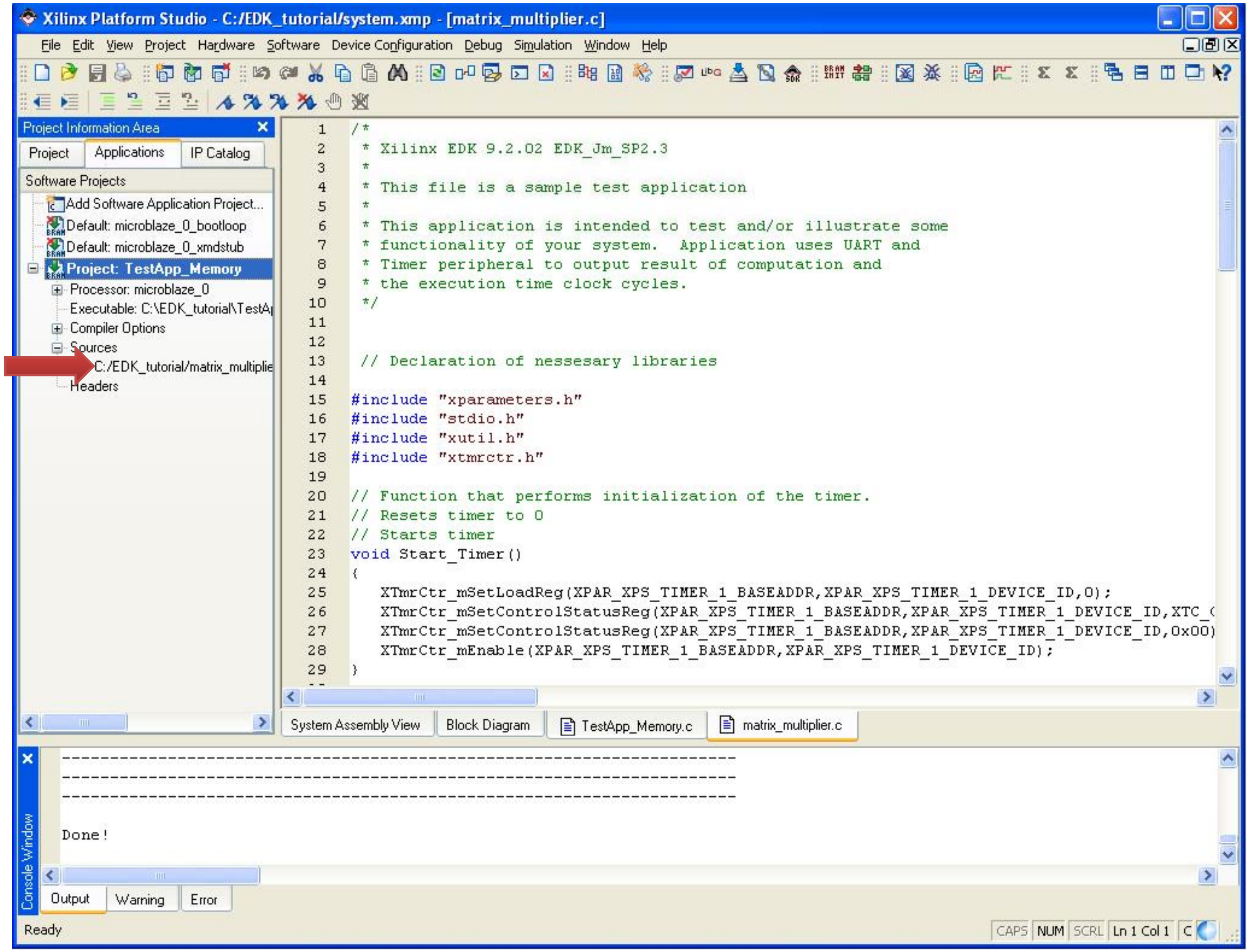
Upon completion of the wizard you will be presented with the Xilinx Platform Studio and generated project name ***TestApp_Memory***. All the selected peripherals are displayed in the ***System Assembly View*** window. Initial generated C file is shown in ***Sources.***



```
 * IP driver functions.  These drivers will be generated in your
 * XPS project when you run the "Generate Libraries" menu item
 * in XPS.
 *
 * Your XPS project directory is at:
 *    C:\EDK_tutorial\
 */


// Located in: microblaze_0/include/xparameters.h
#include "xparameters.h"

          ude "stdio.h"

          ude "xutil.h"

//====================================================

int main (void) {


   /*
    * Enable and initialize cache
    */
   #if XPAR_MICROBLAZE_0_USE_ICACHE
      microblaze_init_icache_range(0, XPAR_MICROBLAZE_0_CACHE_BYTE_SIZE);
      microblaze_enable_icache();
   #endif
```

```
Generating Block Diagram : C:\EDK_tutorial\blkdiagram\system.html...
Generated --- system.svg
Generated --- system.svg
Block diagram generated.
Block diagram generated.
```

At this point sample file can be edited to implement your functionality. However, we will simply insert provided C file. Please make sure ***matrix_multiplier.c*** is downloaded to your project directory. Then right click on ***Sources*** and select ***Add New File…*** and select ***matrix_multiplier.c***


Xilinx Platform Studio - C:/EDK_tutorial/system.xmp - [matrix_multiplier.c]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Information Area
Project Applications IP Catalog
Software Projects
Add Software Application Project...
Default: microblaze_0_bootloop
Default: microblaze_0_xmdstub
Project: TestApp_Memory
Processor: microblaze_0
Executable: C:\EDK_tutorial\TestAp
Compiler Options
Sources
C:/EDK_tutorial/matrix_multiplie
Headers
System Assembly View
Block Diagram
TestApp_Memory.c
matrix_multiplier.c
Console Window
Output Warning Error
Ready
CAPS NUM SCRL Ln 1 Col 1 C

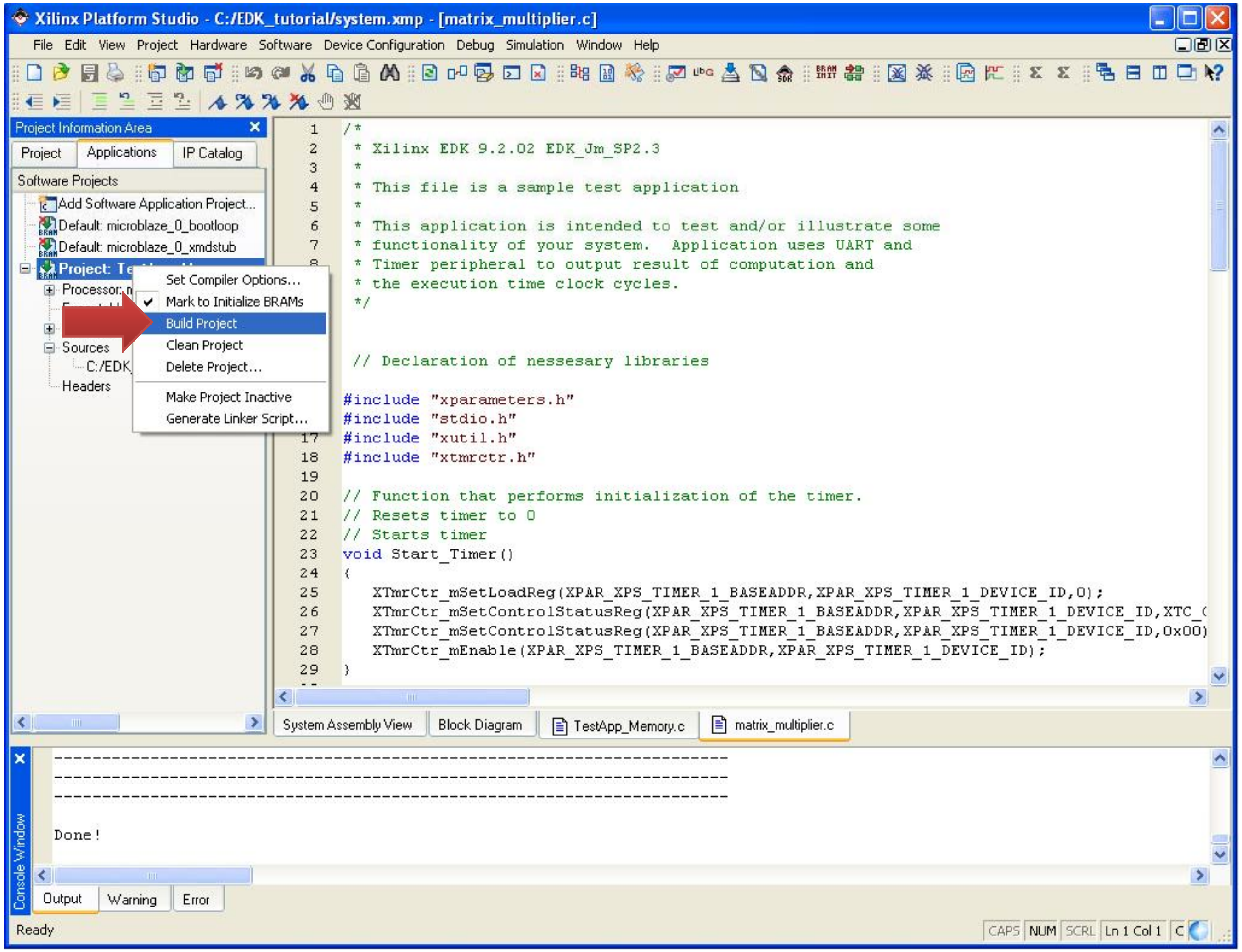
```
/*
 * Xilinx EDK 9.2.02 EDK_Jm_SP2.3
 *
 * This file is a sample test application
 *
 * This application is intended to test and/or illustrate some
 * functionality of your system.  Application uses UART and
 * Timer peripheral to output result of computation and
 * the execution time clock cycles.
 */


 // Declaration of nessesary libraries

#include "xparameters.h"
#include "stdio.h"
#include "xutil.h"
#include "xtmrctr.h"

// Function that performs initialization of the timer.
// Resets timer to 0
// Starts timer
void Start_Timer()
{
   XTmrCtr_mSetLoadReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,0);
   XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,XTC_(
   XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,0x00)
   XTmrCtr_mEnable(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID);
}
```

```
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------

Done!
```

After you added the matrix_multiplier.c your screen should look like the figure above. This simple program is listed and explained in the next couple of slides.

```
// Declaration of nessesary libraries

#include "xparameters.h"
#include "stdio.h"
#include "xutil.h"
#include "xtmrctr.h"

// Function that performs initialization of the timer.
// Resets timer to 0
// Starts timer
void Start_Timer()
{
                XTmrCtr_mSetLoadReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,0);
                XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,XTC_CSR_LOAD_MASK);
                XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,0x00);
                XTmrCtr_mEnable(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID);
}


// This function stops timer and returns final value of the timer.
int Stop_Timer()
{
                XTmrCtr_mDisable(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID);
                return XTimerCtr_mReadReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID, XTC_TCR_OFFSET);
}


int Get_Timer()
{
                return XTimerCtr_mReadReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID, XTC_TCR_OFFSET);
}
```

```
int main (void)
{
        // Initialization of the nessesary variables
        int i,j,k,start_timer_value,end_timer_value;

        // Initialization of source A and B 8x8 matricies and resultunt C matrix
        int a[8][8]={                                           {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8}};

        int b[8][8]={                                           {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8},
                                                                {1,2,3,4,5,6,7,8}};

        int c[8][8]={                                           {0,0,0,0,0,0,0,0},
                                                                {0,0,0,0,0,0,0,0},
                                                                {0,0,0,0,0,0,0,0},
                                                                {0,0,0,0,0,0,0,0},
                                                                {0,0,0,0,0,0,0,0},
                                                                {0,0,0,0,0,0,0,0},
                                                                {0,0,0,0,0,0,0,0},
                                                                {0,0,0,0,0,0,0,0}};
        // Output to the terminal beginning of processing
        xil_printf("-- Entering main() --\r\n");

        Start_Timer();
        start_timer_value=Get_Timer(); // record starting time

        // Perform ordinary matrix multiplication using three nested loops
        for (i=0; i<8; i++ )
        {
                for (j=0; j<8; j++)
                {
                        for(k=0; k<8; k++)
                        {
                                c[i][j]=c[i][j]+a[i][k]*b[k][j];
                        }
                }
        }
        // Stop the timer and record its final value
        end_timer_value=Stop_Timer();

        // print out to the terminal starting and final value of the clock cycle counter
        xil_printf("\n\rStart Timer Value=> %d\n\r",start_timer_value);
        xil_printf("\n\rEnd Timer Value=> %d\n\r",end_timer_value);

        // print out final resulting matrix
        for (i=0; i<8; i++ )
        {
                for (j=0; j<8; j++)
                {
                        xil_printf("%d ",c[i][j]);
                }
                xil_printf("\n\r");
        }

  return 0;
}
```



```
/*
 * Xilinx EDK 9.2.02 EDK_Jm_SP2.3
 *
 * This file is a sample test application
 *
 * This application is intended to test and/or illustrate some
 * functionality of your system.  Application uses UART and
 * Timer peripheral to output result of computation and
 * the execution time clock cycles.
 */


 // Declaration of nessesary libraries

#include "xparameters.h"
#include "stdio.h"
#include "xutil.h"
#include "xtmrctr.h"

// Function that performs initialization of the timer.
// Resets timer to 0
// Starts timer
void Start_Timer()
{
   XTmrCtr_mSetLoadReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,0);
   XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,XTC_(
   XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID,0x00)
   XTmrCtr_mEnable(XPAR_XPS_TIMER_1_BASEADDR,XPAR_XPS_TIMER_1_DEVICE_ID);
}
```

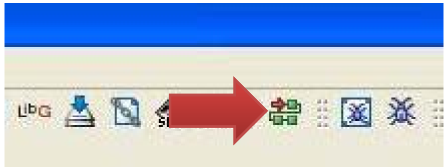
```
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------


Done!
```

To compile the project, right click on the ***Project*** and select ***Build Project***.
Also, make sure that the ***Mark to Initialize BRAMs*** is selected.
After successful compilation bitstream is ready to be loaded on to the platform.

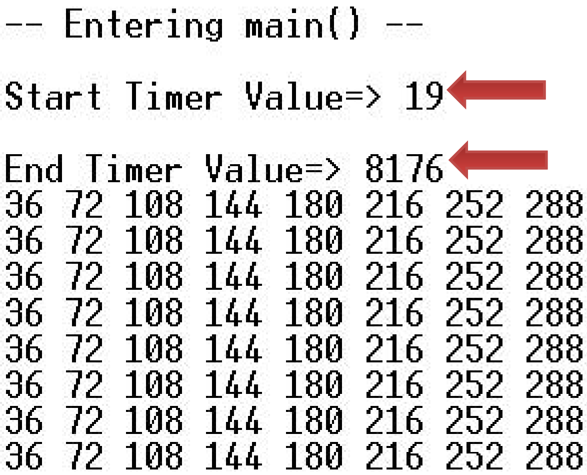
To test the operation of the matrix multiplier we have to open Serial terminal such as Hyper Terminal/Terraterm/Realterm/Minicom to be able to see the output of the platform. Settings should be 9600 Baud, 1 Stop bit, No parity, No hardware handshaking , as was set up on the Slide #5 .

In this tutorial we use Minicom which has initial settings mentioned above. To run it open ***Terminal*** window and type ***minicom*** and press ***Enter.***
This will bring up the Minicom application.

To load the bitstream on the platform click on ***Download Bitstream*** icon
First generation of the bitstream will take a **WHILE!**



On the successful load terminal will output all of the application print statements which show: starting timer value, final timer value, and the result of the matrix multiplication operation.



```
-- Entering main() --

Start Timer Value=> 19

End Timer Value=> 8176
36 72 108 144 180 216 252 288
36 72 108 144 180 216 252 288
36 72 108 144 180 216 252 288
36 72 108 144 180 216 252 288
36 72 108 144 180 216 252 288
36 72 108 144 180 216 252 288
36 72 108 144 180 216 252 288
36 72 108 144 180 216 252 288
```

Conclusion

This completes tutorial 4 which included:

- Creation of a new EDK project.
- Initialization of the required peripherals.
- Setup of a test program for testing:
    - Matrix multiplication
    - Timing of computation
    - Output to the serial port
- Compilation and upload of the bitstream on to the target platform
- Verification of operation with ***minicom*** terminal