Notes on Xilinx EDK 9.1 for Spartan 3E 1500 Development Kit : Basic

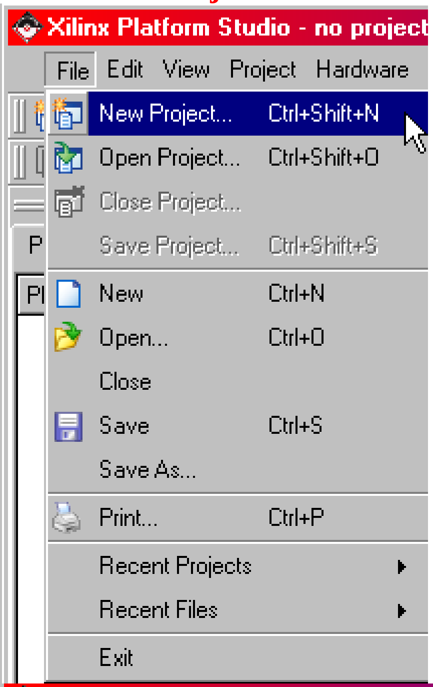
© Duy-Ky Nguyen **2007 July 01**

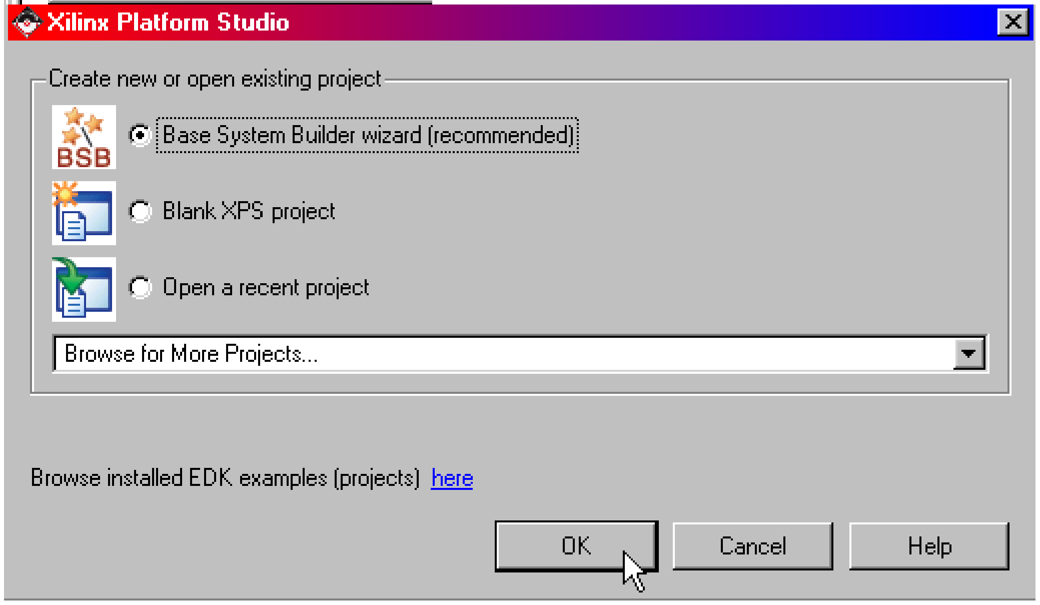
Note :

It is 1500 because it's modified from 500, the only change is FPGA from XC3S500E to XC3S1600E. The reference board 500E-RevD is copied as 1600E-Rev B, FPGA changes from 500 to 1600. That's it. While there's a refenece board 1600E-RevA.

1. Create New Design

Create **New Project** based on **BSB** (Base System Builder) wizard

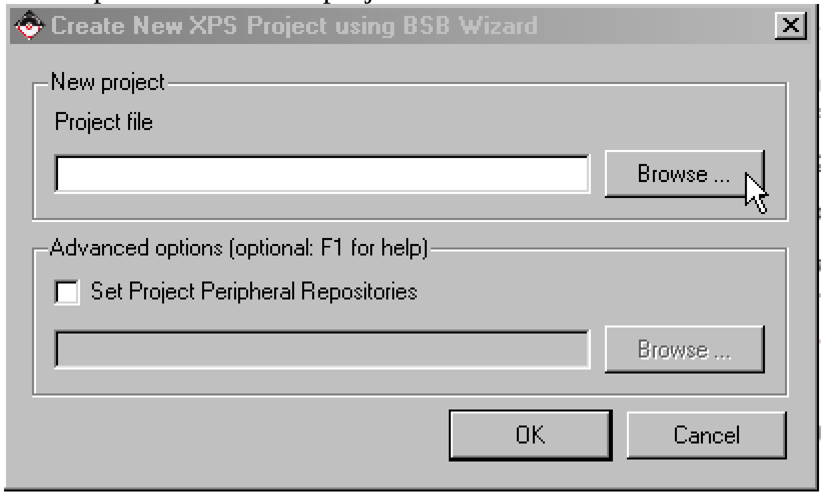




[✓ Base System Builder wizard (recommended)]
Blank XPS project
Open a recent project
Browse for More Projects...

Browse installed EDK examples (projects) here

Find a place for the new project



Create New XPS Project using BSB Wizard

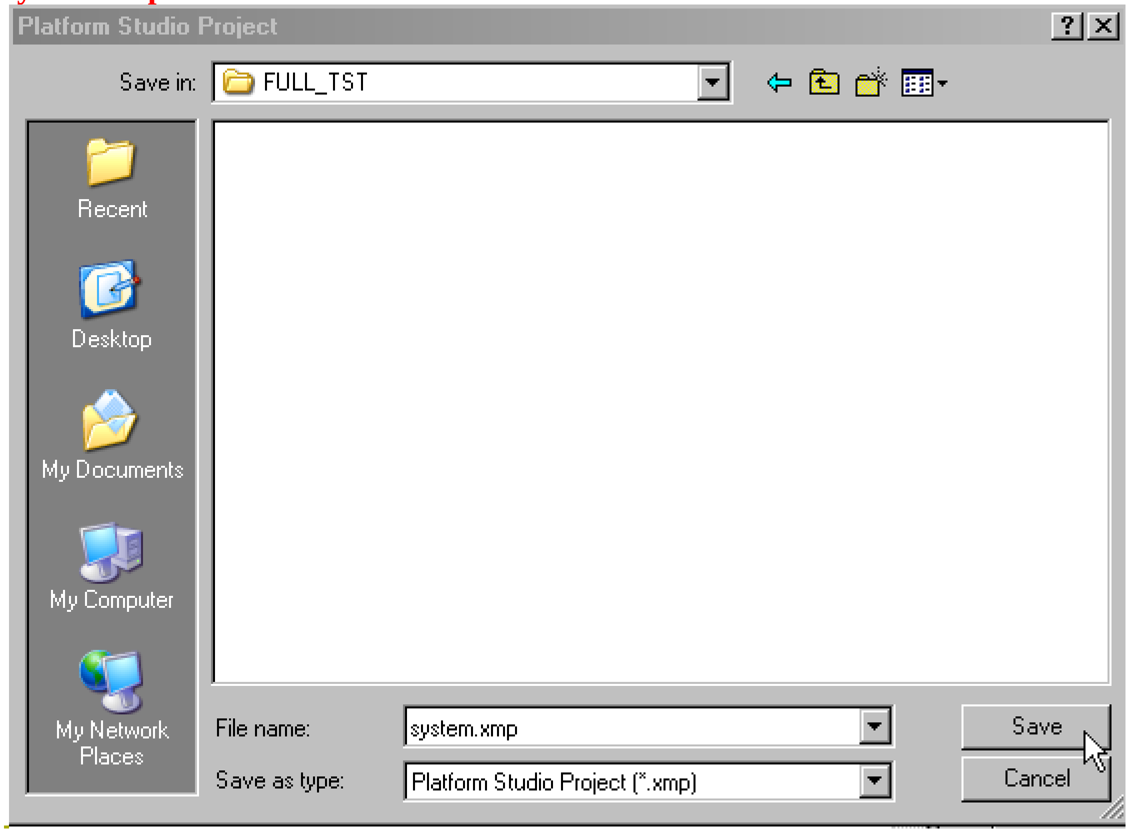
New project

Project file

Advanced options (optional: F1 for help)

[ ] Set Project Peripheral Repositories

system.xmp



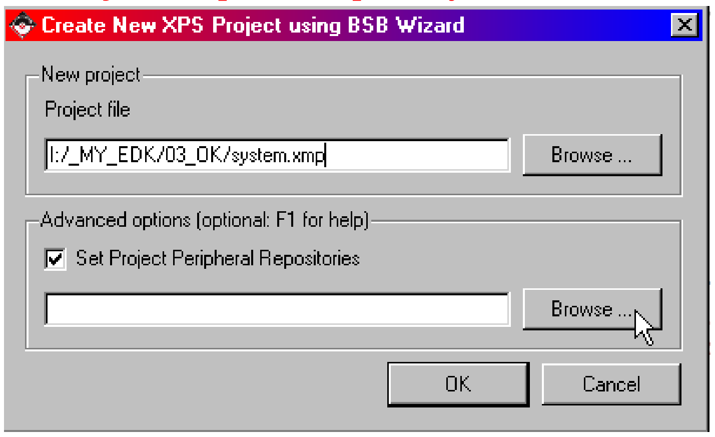
Platform Studio Project

Save in: FULL_TST

File name: system.xmp

Save as type: Platform Studio Project (*.xmp)

Set **Project Peripheral Repository** for Global IP_core


Create New XPS Project using BSB Wizard

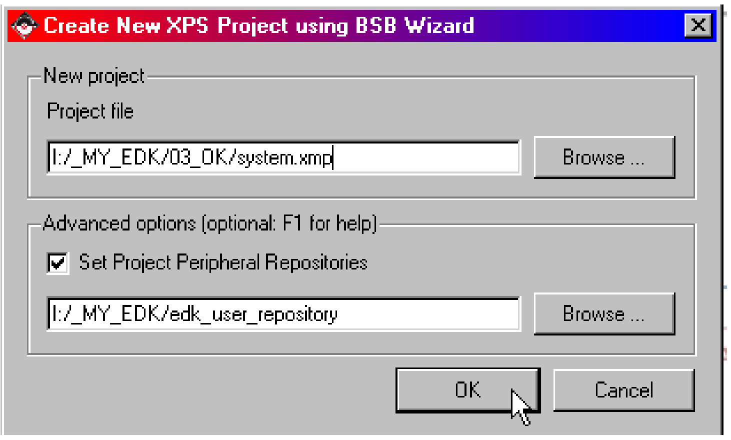
New project

Project file

I:/_MY_EDK/03_OK/system.xmp

Advanced options (optional: F1 for help)

☑ Set Project Peripheral Repositories


Create New XPS Project using BSB Wizard

New project

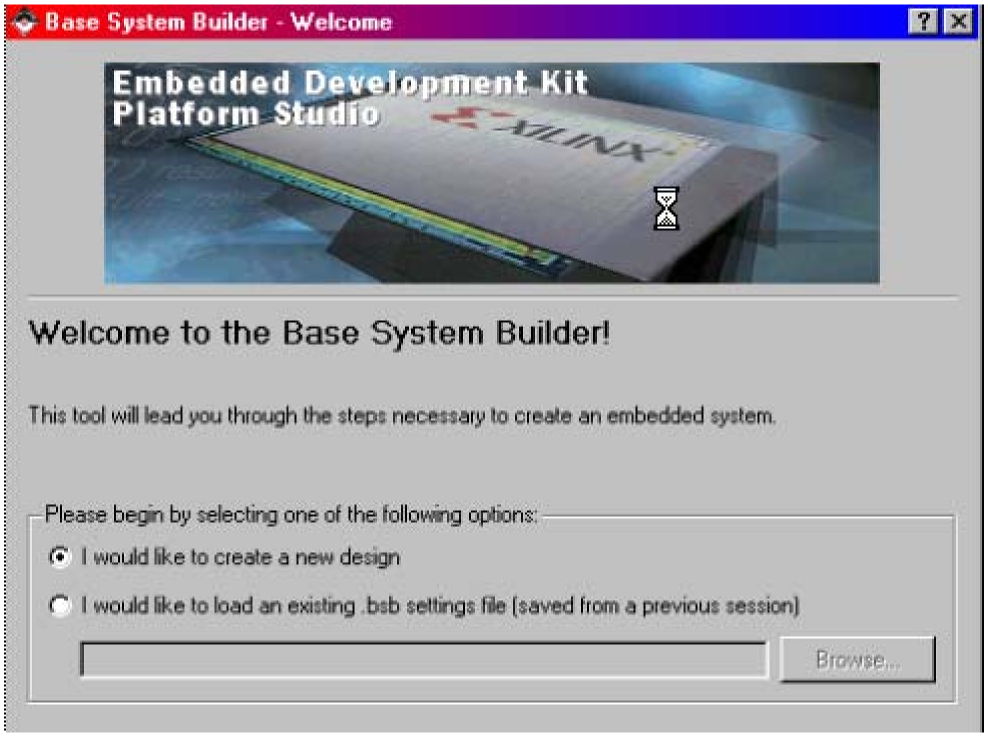
Project file

I:/_MY_EDK/03_OK/system.xmp

Advanced options (optional: F1 for help)

☑ Set Project Peripheral Repositories

I:/_MY_EDK/edk_user_repository


Base System Builder - Welcome

Welcome to the Base System Builder!

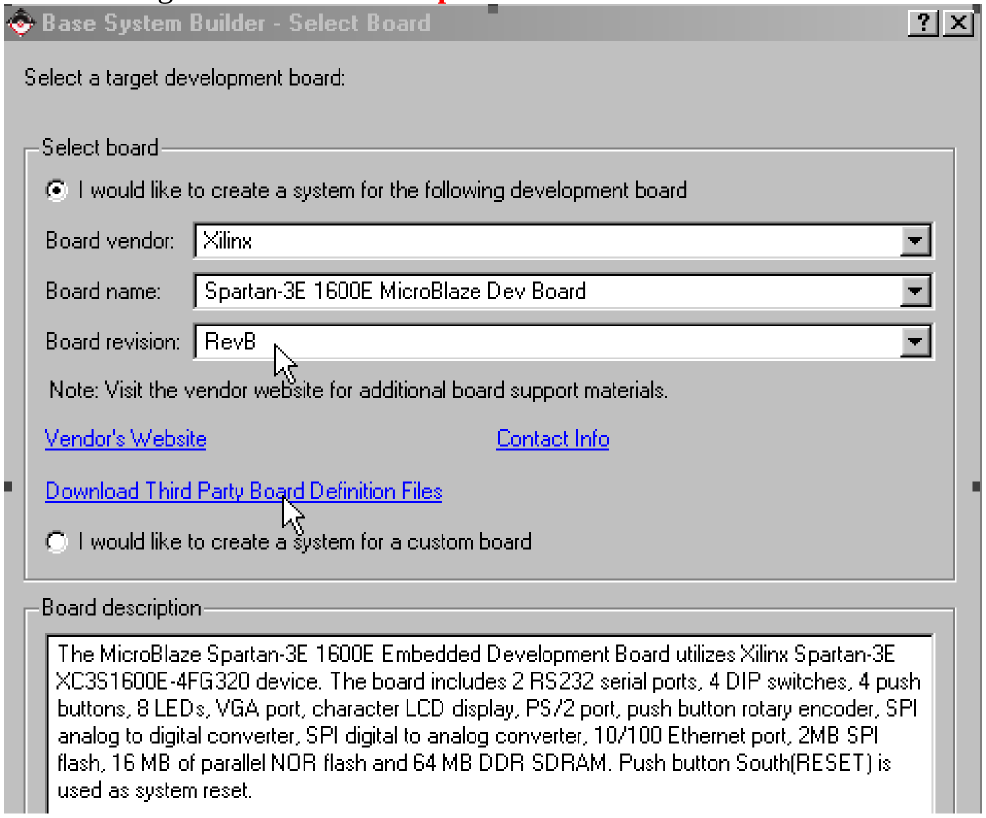
This tool will lead you through the steps necessary to create an embedded system.

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)

Select the right board : **Xilinx – Spartan-3E Starter Board – D**


Base System Builder - Select Board
Vendor's Website
Contact Info
Download Third Party Board Definition Files

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 1600E MicroBlaze Dev Board

Board revision: RevB

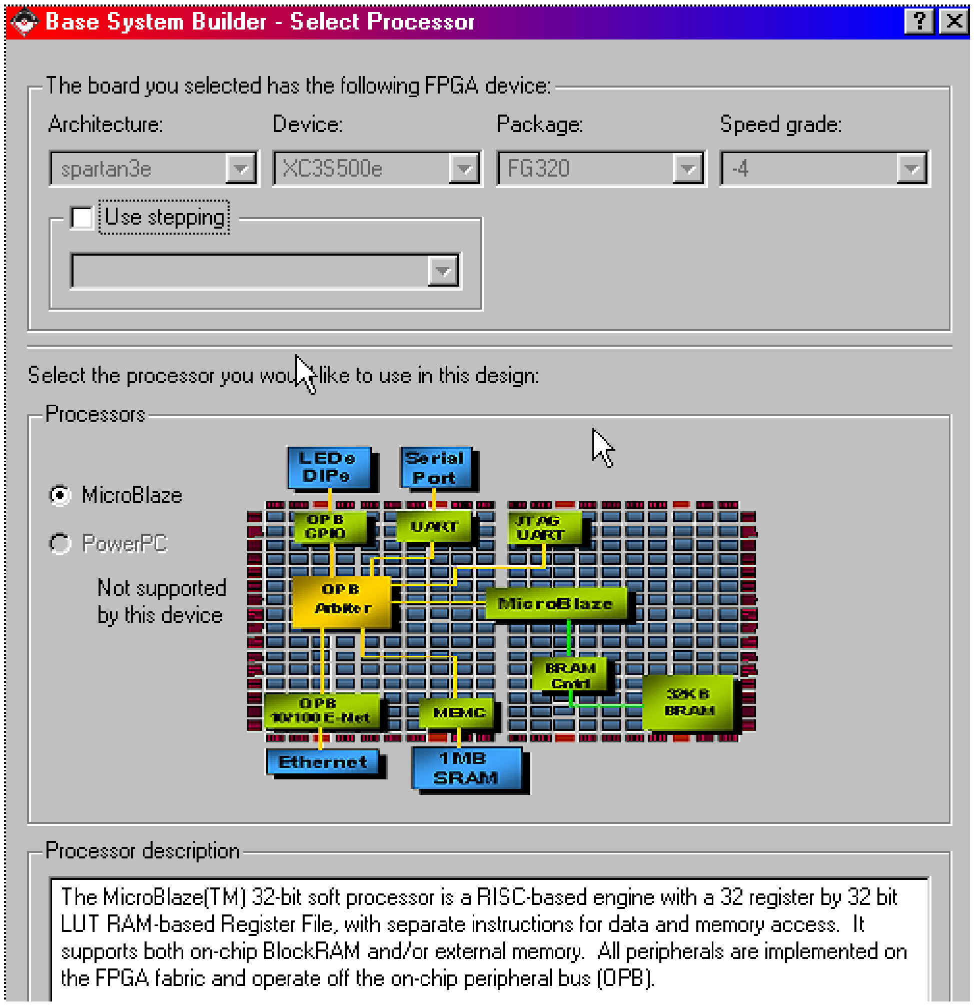
Note: Visit the vendor website for additional board support materials.

( ) I would like to create a system for a custom board

Board description

The MicroBlaze Spartan-3E 1600E Embedded Development Board utilizes Xilinx Spartan-3E XC3S1600E-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 soft core processor **MicroBlaze**



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

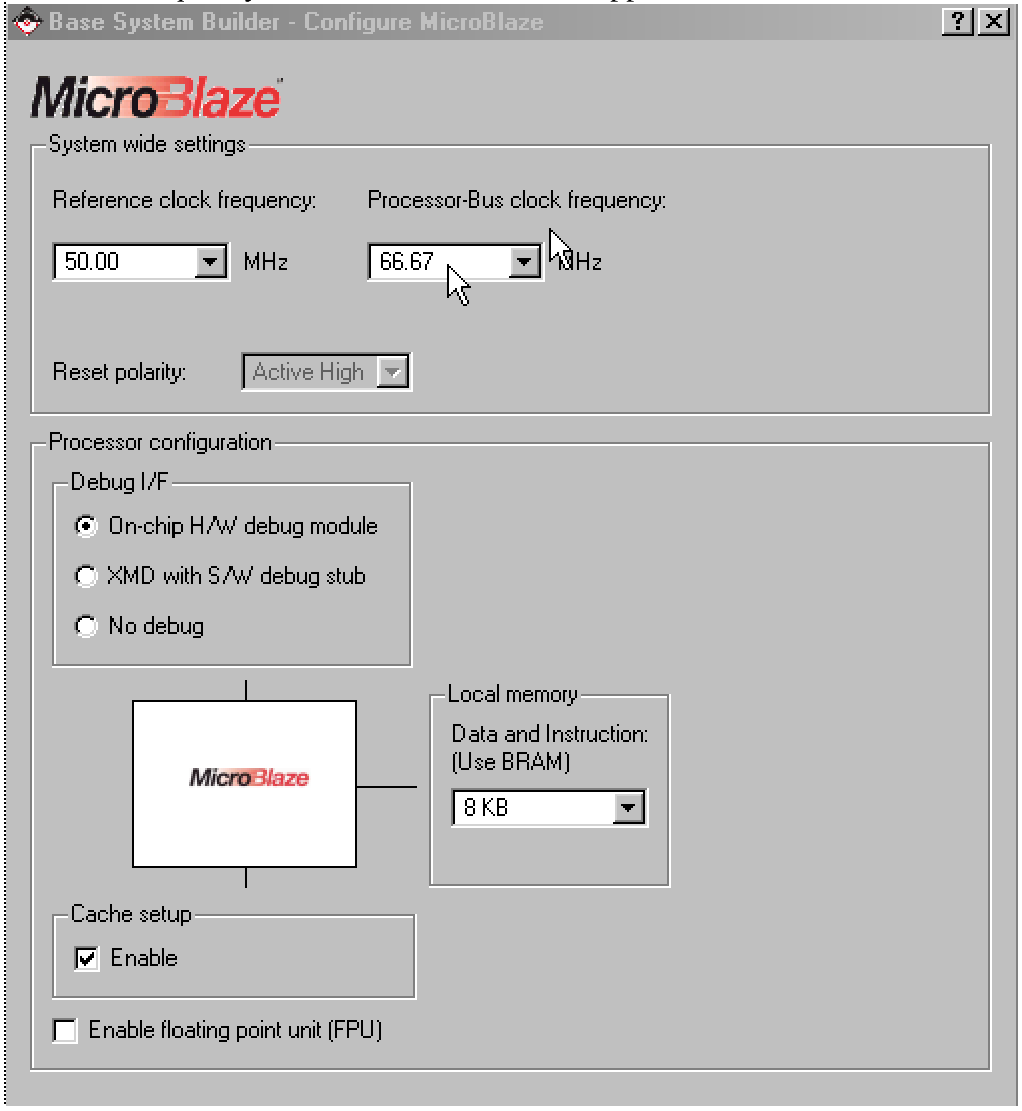
○ PowerPC

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 and operate off the on-chip peripheral bus (OPB).

Select operating frequency of **50 MHz, HW debug module**. **Enable cache setup**. Note that the cache of 8KB from BRAM is set here, but it will be elaborated later for D/I-cache. Sparatn 500E has 20 BRAM of 2KB each. The CPU frequency is chosen 66.67 MHz to support Ethernet 100 BaseT


Base System Builder - Configure MicroBlaze

System wide settings

Reference clock frequency: 50.00 MHz

Processor-Bus clock frequency: 66.67 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

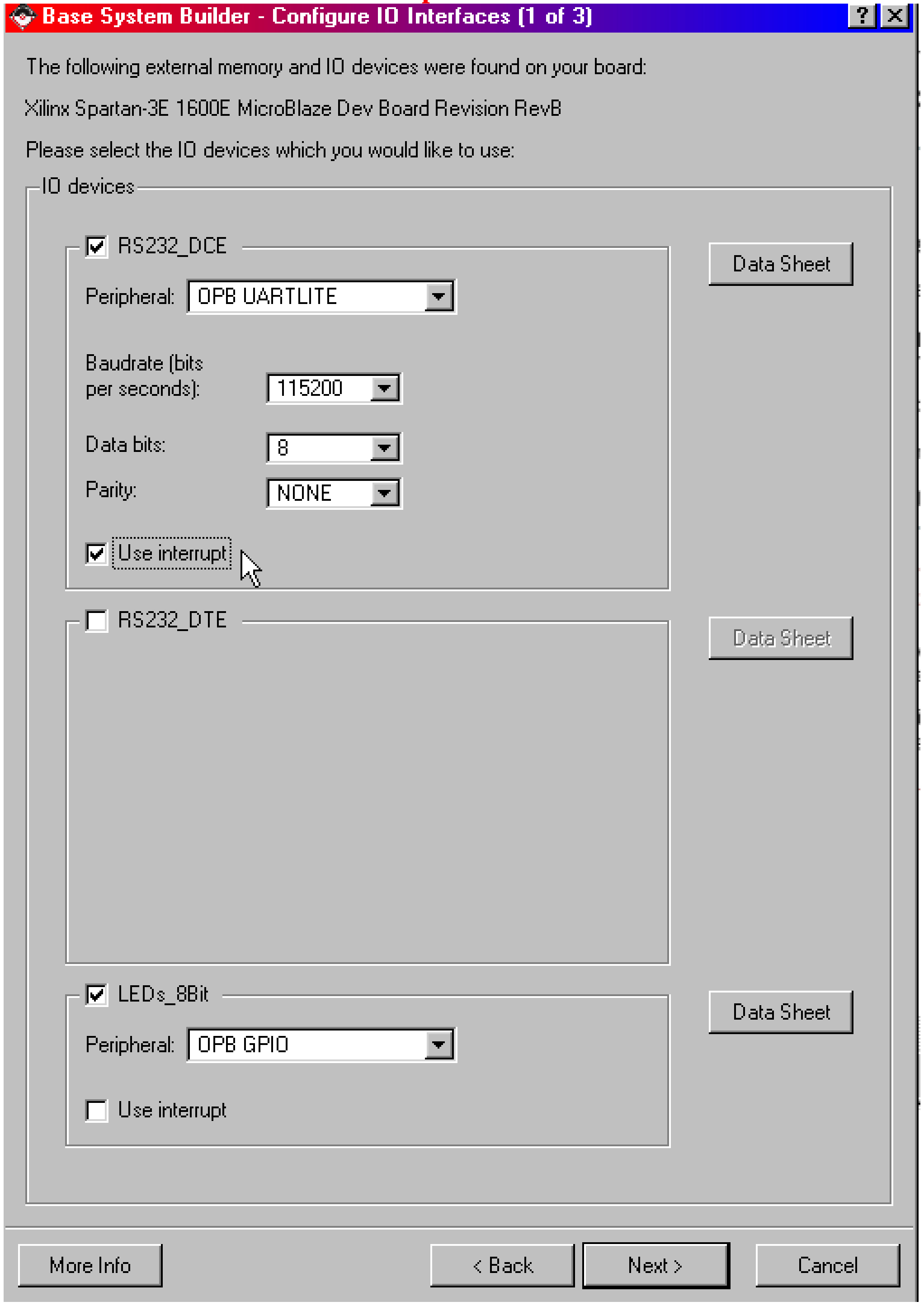
Data and Instruction: (Use BRAM) 8 KB

Cache setup

[✓ Enable]

☐ Enable floating point unit (FPU)

Select baudrate **115200** and **Interrupt** for for both DCE. NO DTE.



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

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

Xilinx Spartan-3E 1600E MicroBlaze Dev Board Revision RevB

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

IO devices

[✓] RS232_DCE

Peripheral: OPB UARTLITE

Baudrate (bits per seconds): 115200

Data bits: 8

Parity: NONE

[✓] Use interrupt

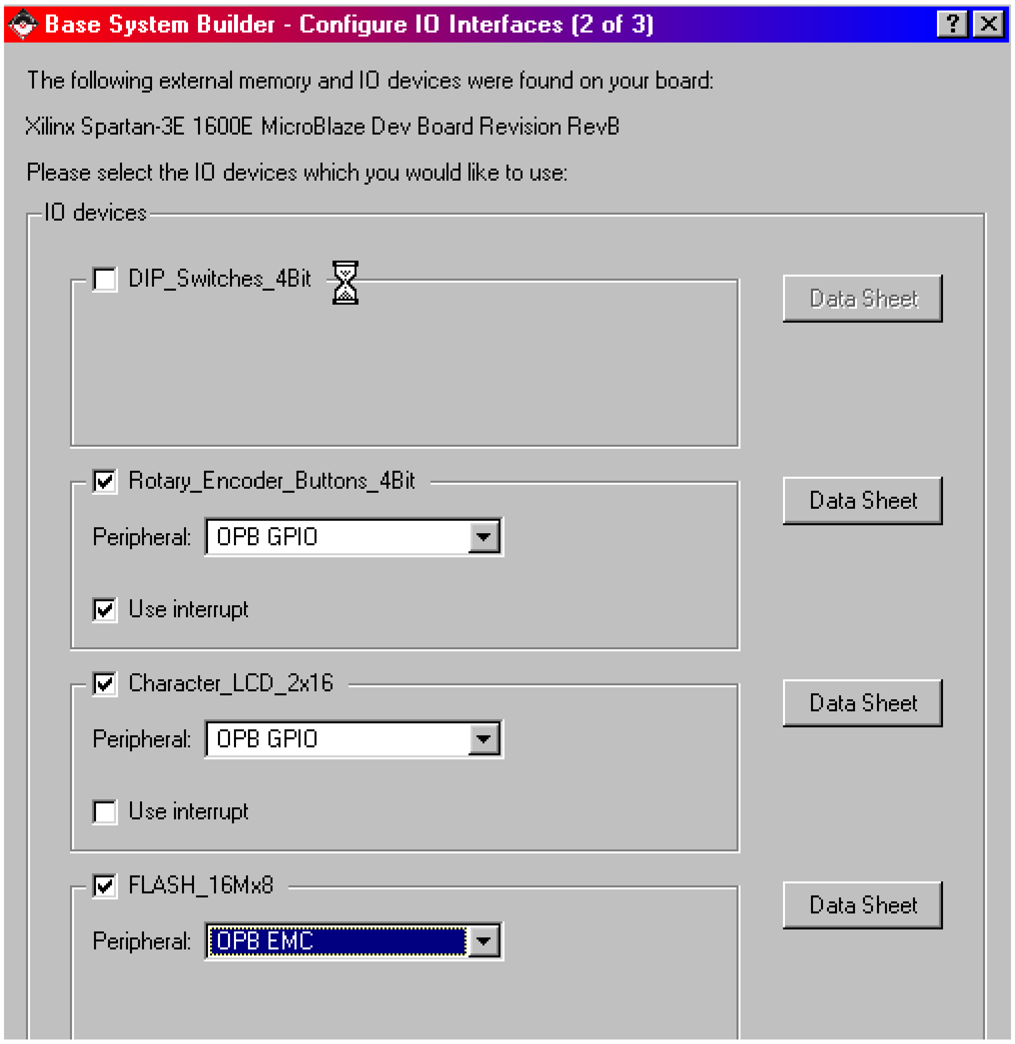
[ ] RS232_DTE

[✓] LEDs_8Bit

Peripheral: OPB GPIO

[ ] Use interrupt

Select Peripheral : OPB EMC, not MCH one, for Flash. The DIP_Switches_4Bit is NOT selected as it is included in HW as selection switch.



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

Xilinx Spartan-3E 1600E MicroBlaze Dev Board Revision RevB

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

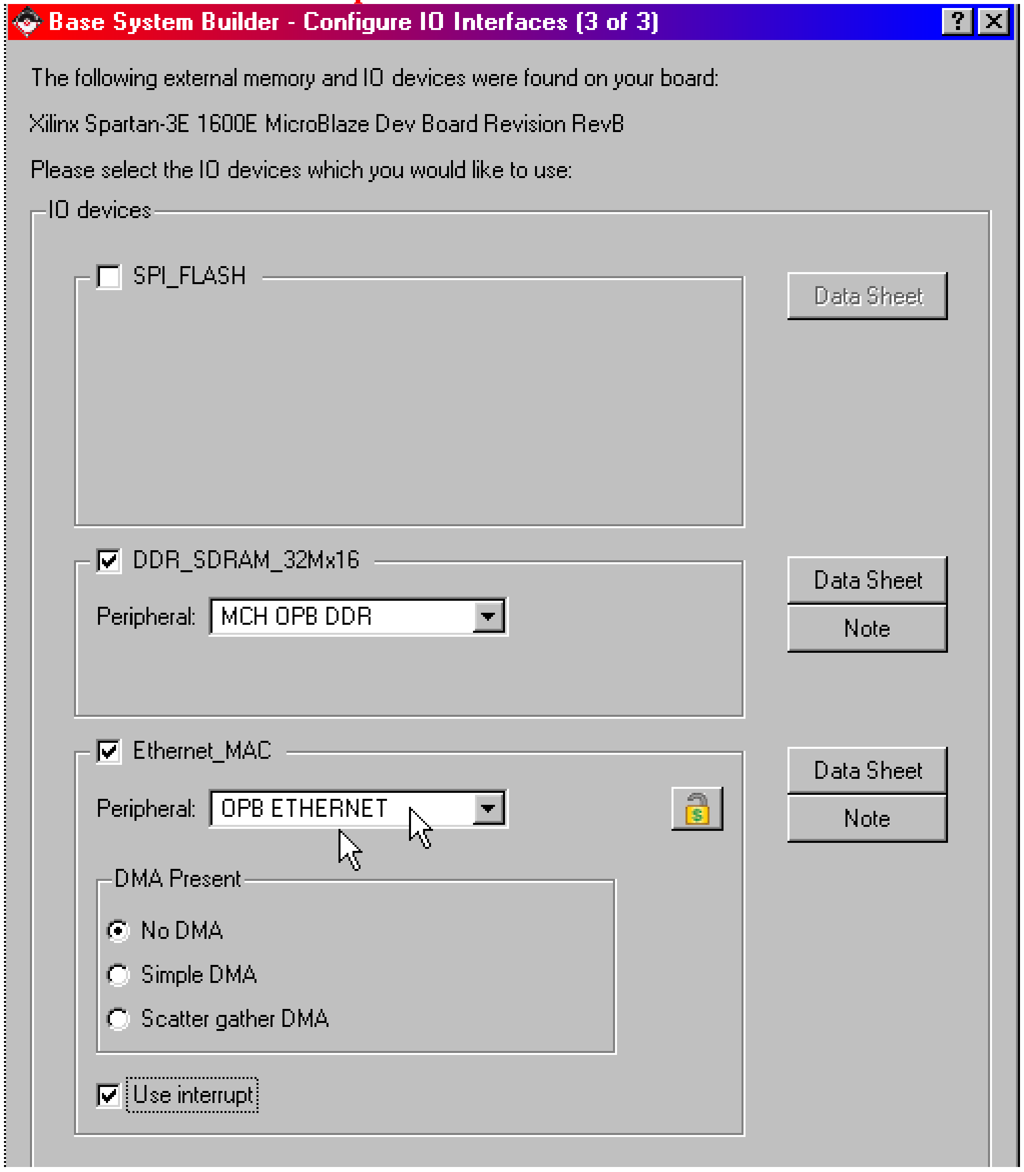
☐ DIP_Switches_4Bit

☑ Rotary_Encoder_Buttons_4Bit
Peripheral: OPB GPIO
☑ Use interrupt

☑ Character_LCD_2x16
Peripheral: OPB GPIO
☐ Use interrupt

☑ FLASH_16Mx8
Peripheral: OPB EMC

Use **Ethernet** with **interrupt**.



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

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

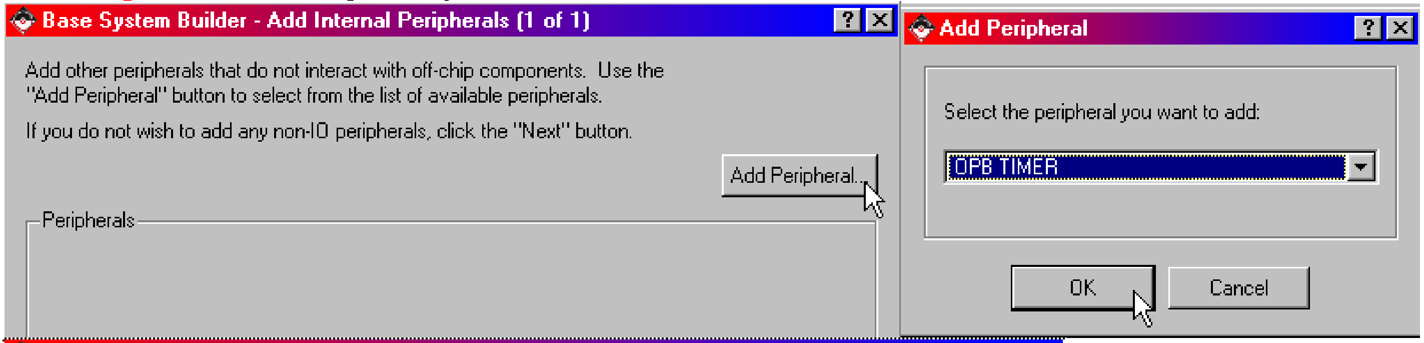
Xilinx Spartan-3E 1600E MicroBlaze Dev Board Revision RevB

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

IO devices

- [ ] SPI_FLASH
- [x] DDR_SDRAM_32Mx16 — Peripheral: MCH OPB DDR
- [x] Ethernet_MAC — Peripheral: OPB ETHERNET
  - DMA Present: [✓ No DMA] Simple DMA, Scatter gather DMA
  - [x] Use interrupt

Add **Peripheral Timer** required by uCLinux.



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

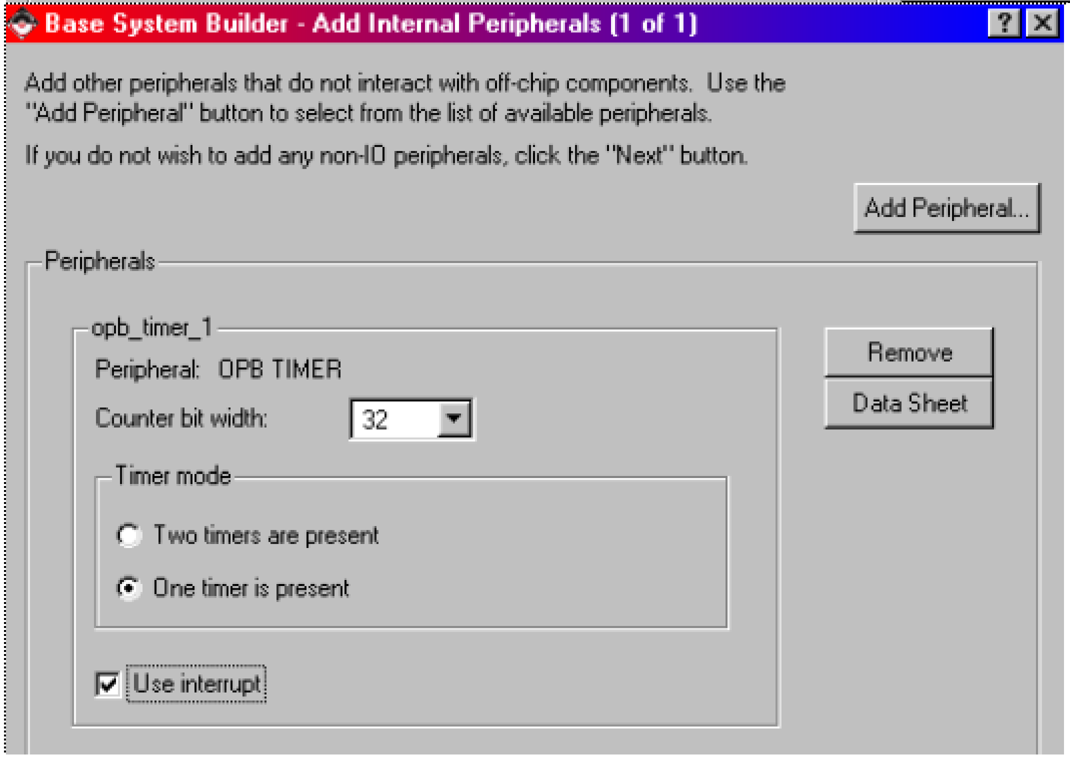
Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

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

Add Peripheral

Select the peripheral you want to add:

OPB TIMER



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.

Peripheral: OPB TIMER

Counter bit width: 32

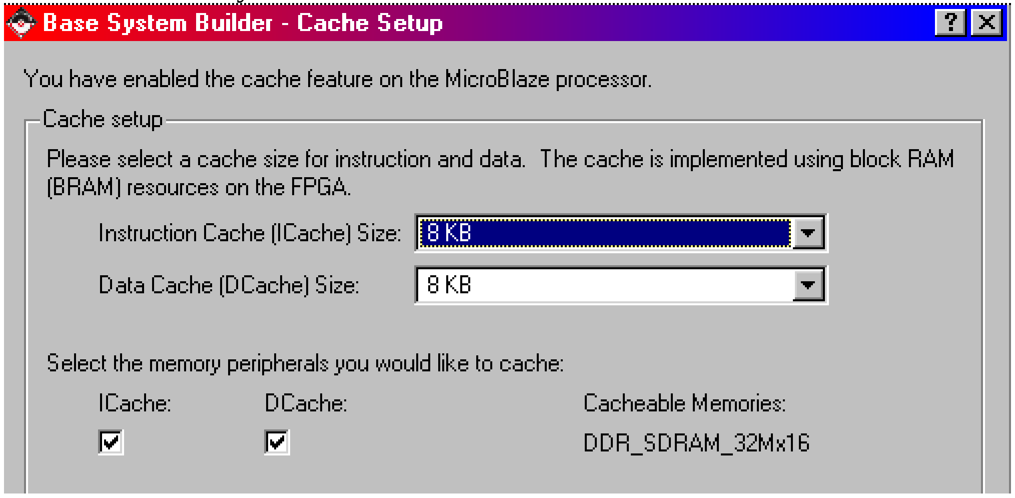
Timer mode

( ) Two timers are present

(●) One timer is present

[✓] Use interrupt

Now select size **8KB** for **each of D/I-cache** and source of cache is SDRAM as SW normally is loaded from Flash into RAM by bootloader and runs out of RAM.



Base System Builder - Cache Setup

You have enabled the cache feature on the MicroBlaze processor.

Cache setup

Please select a cache size for instruction and data. The cache is implemented using block RAM (BRAM) resources on the FPGA.

Instruction Cache (ICache) Size: 8 KB

Data Cache (DCache) Size: 8 KB

Select the memory peripherals you would like to cache:

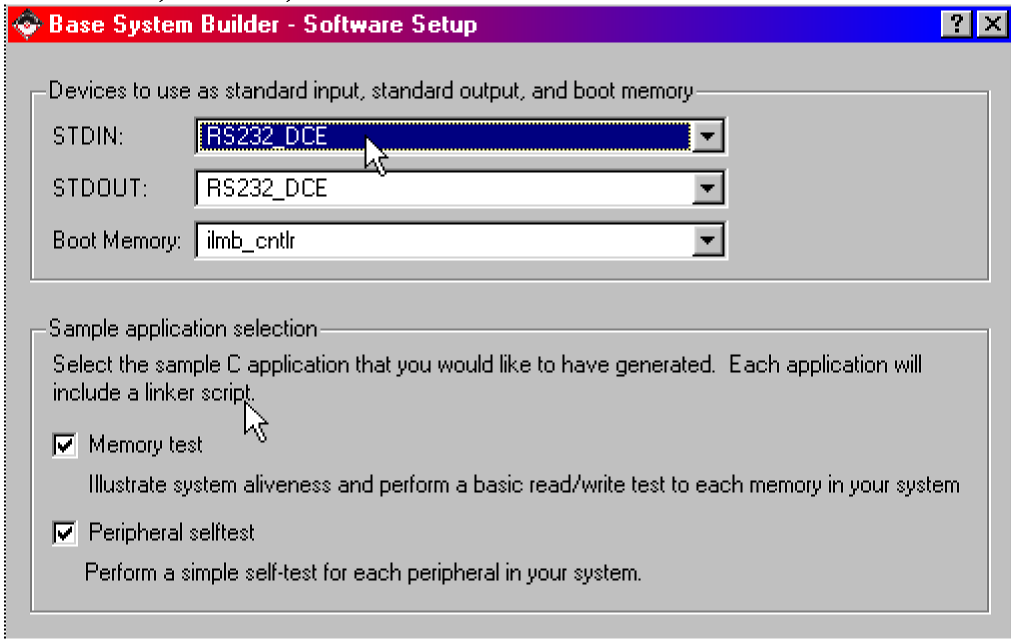
ICache:	DCache:	Cacheable Memories:
✓	✓	DDR_SDRAM_32Mx16

Now, BRAM has been used as below

16 KB for start up code (8 BRAM)

8KB + 8KB = 16 KB for D/I-cache (8 BRAM)

Select **DCE**, not DTE, for STD-IN/OUT



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.

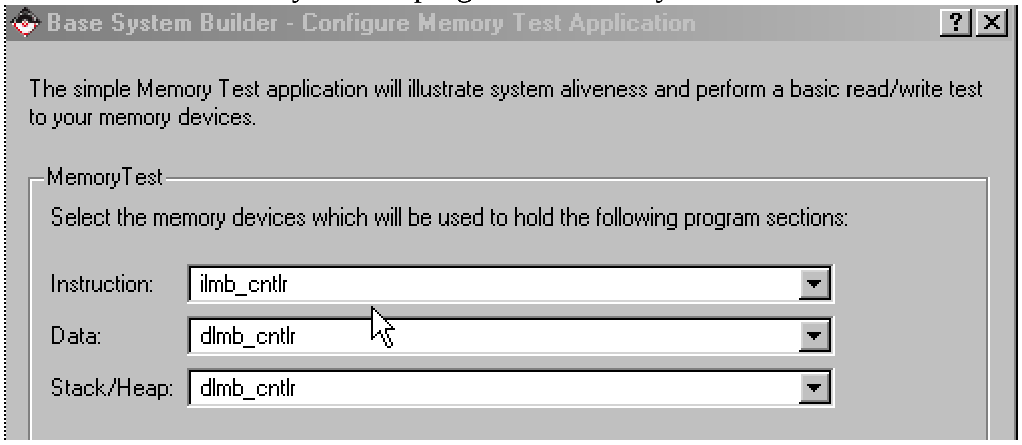
[✓] Memory test

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.

Use **default** fo memory to hold program of memory test



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

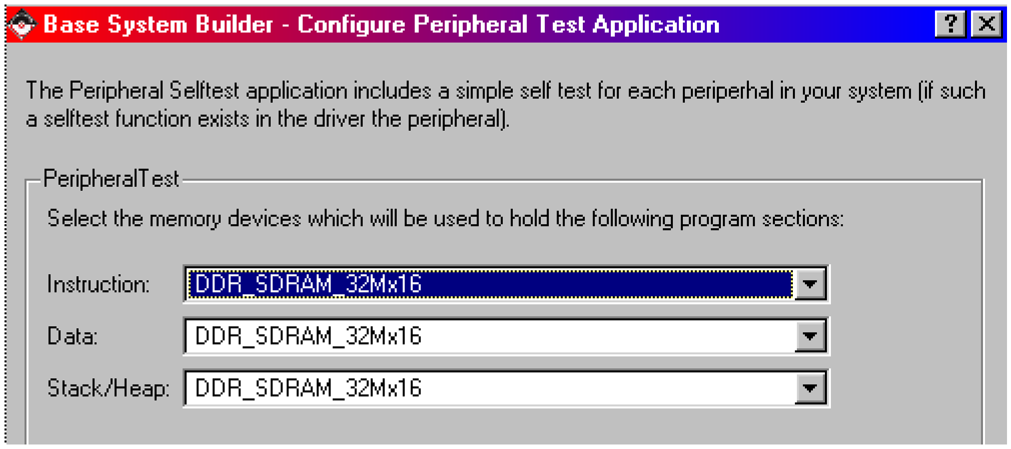
Select the memory devices which will be used to hold the following program sections:

Instruction: ilmb_cntlr

Data: dlmb_cntlr

Stack/Heap: dlmb_cntlr

Use **default**


Base System Builder - Configure Peripheral Test Application

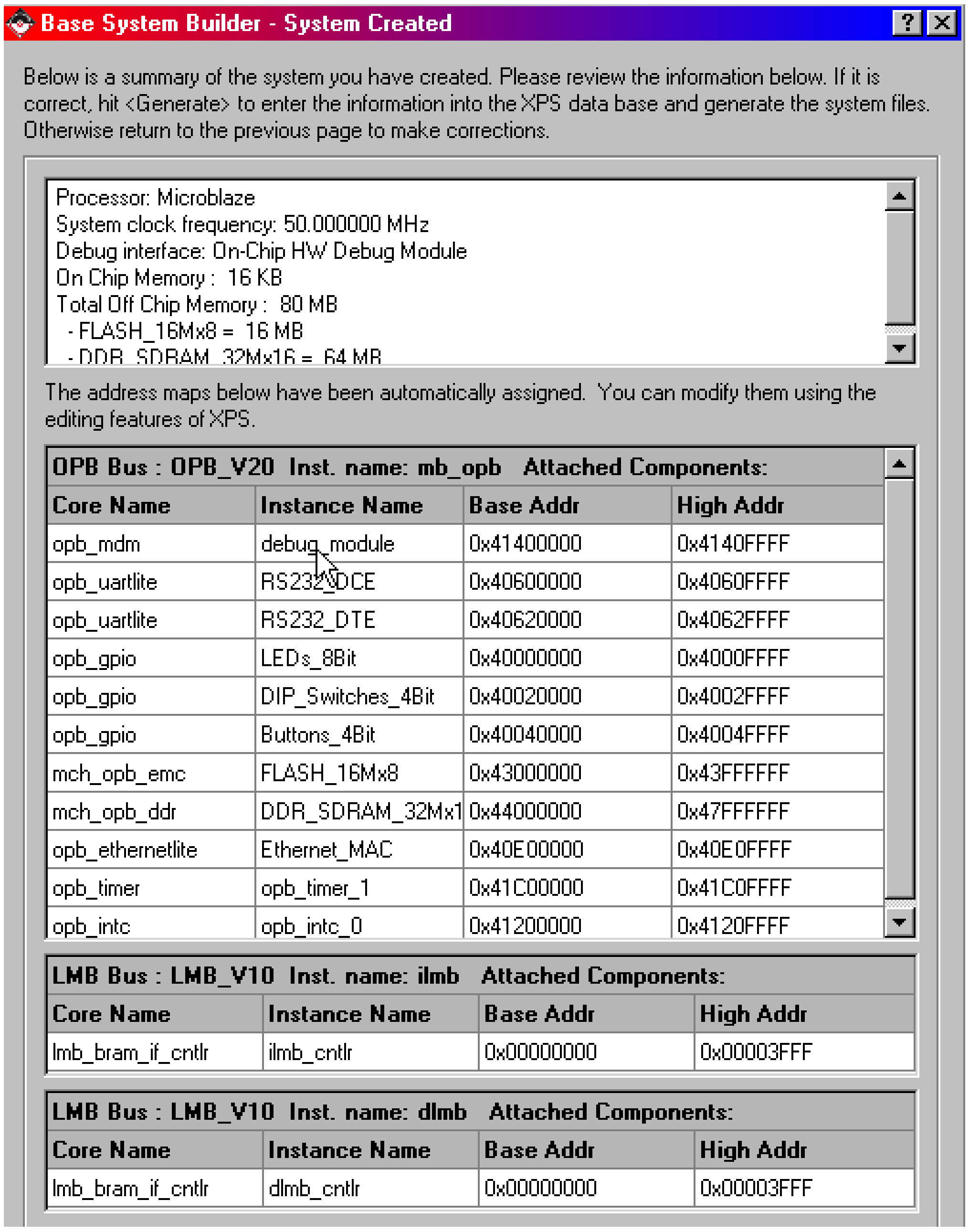
The Peripheral Selftest application includes a simple self test for each periperhal in your system (if such a selftest function exists in the driver the peripheral).

PeripheralTest

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

Instruction:	DDR_SDRAM_32Mx16
Data:	DDR_SDRAM_32Mx16
Stack/Heap:	DDR_SDRAM_32Mx16

Click **Generate**


Base System Builder - System Created

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
System clock frequency: 50.000000 MHz
Debug interface: On-Chip HW Debug Module
On Chip Memory :  16 KB
Total Off Chip Memory :  80 MB
 - FLASH_16Mx8 =  16 MB
 - DDR_SDRAM_32Mx16 =  64 MB
```

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

OPB Bus : OPB_V20 Inst. name: mb_opb   Attached Components:

Core Name	Instance Name	Base Addr	High Addr
opb_mdm	debug_module	0x41400000	0x4140FFFF
opb_uartlite	RS232_DCE	0x40600000	0x4060FFFF
opb_uartlite	RS232_DTE	0x40620000	0x4062FFFF
opb_gpio	LEDs_8Bit	0x40000000	0x4000FFFF
opb_gpio	DIP_Switches_4Bit	0x40020000	0x4002FFFF
opb_gpio	Buttons_4Bit	0x40040000	0x4004FFFF
mch_opb_emc	FLASH_16Mx8	0x43000000	0x43FFFFFF
mch_opb_ddr	DDR_SDRAM_32Mx1	0x44000000	0x47FFFFFF
opb_ethernetlite	Ethernet_MAC	0x40E00000	0x40E0FFFF
opb_timer	opb_timer_1	0x41C00000	0x41C0FFFF
opb_intc	opb_intc_0	0x41200000	0x4120FFFF

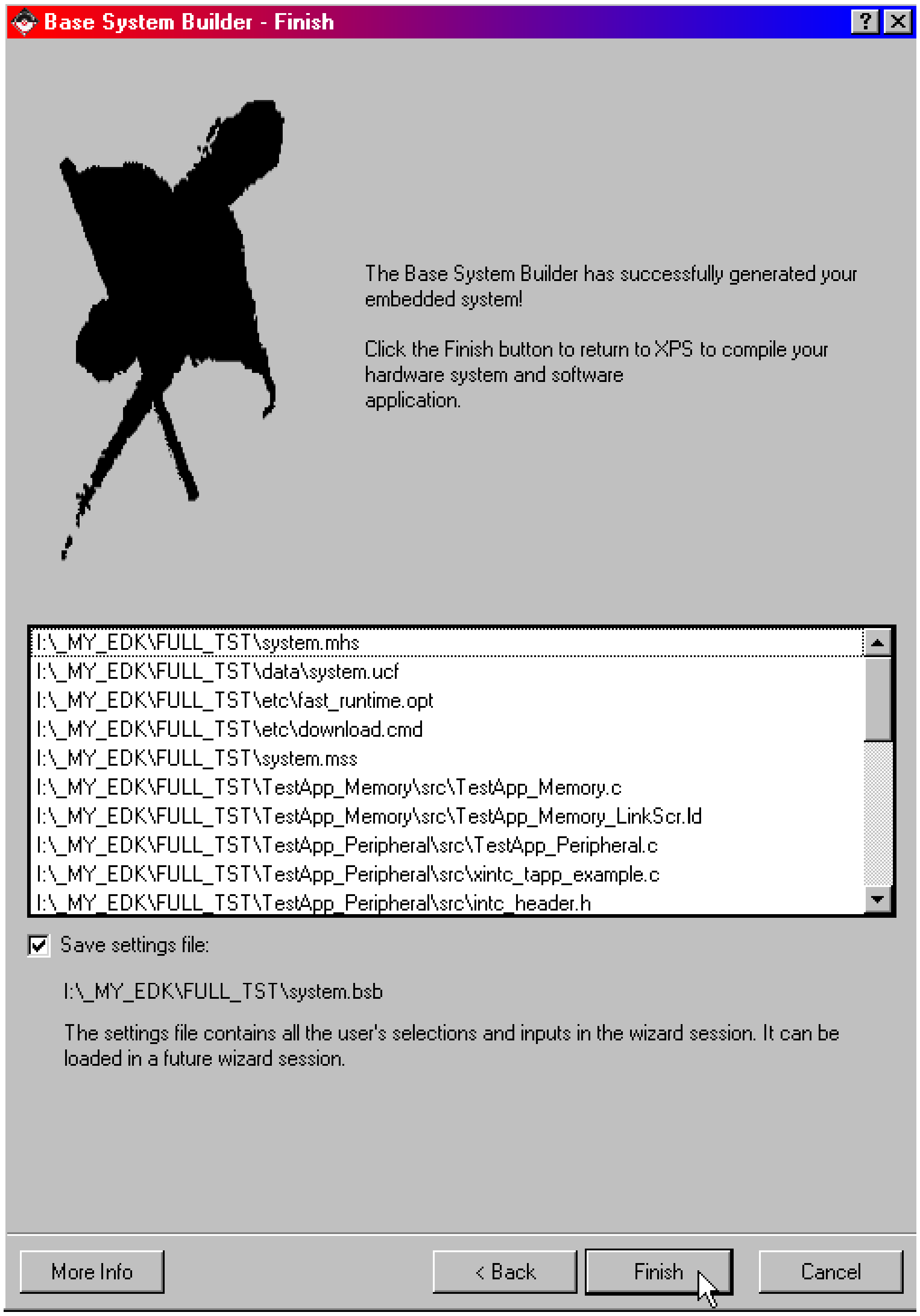
LMB Bus : LMB_V10 Inst. name: ilmb   Attached Components:

Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00003FFF

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

Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	dlmb_cntlr	0x00000000	0x00003FFF

Click **Finish**



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.

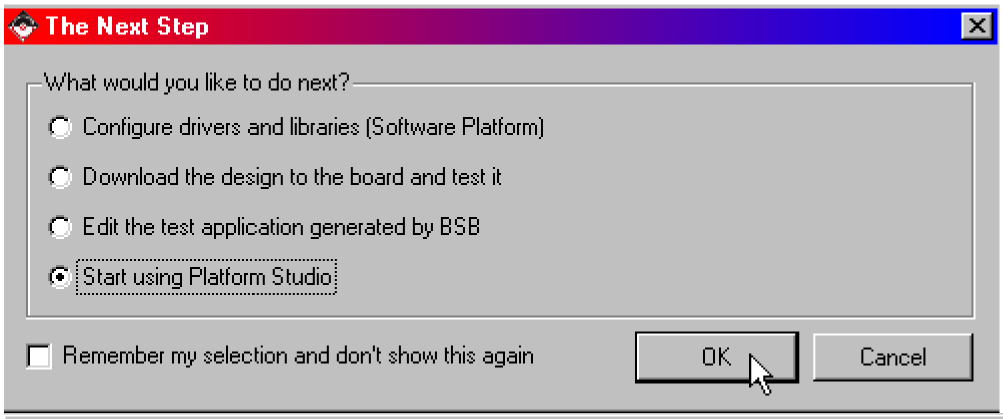
```
I:\_MY_EDK\FULL_TST\system.mhs
I:\_MY_EDK\FULL_TST\data\system.ucf
I:\_MY_EDK\FULL_TST\etc\fast_runtime.opt
I:\_MY_EDK\FULL_TST\etc\download.cmd
I:\_MY_EDK\FULL_TST\system.mss
I:\_MY_EDK\FULL_TST\TestApp_Memory\src\TestApp_Memory.c
I:\_MY_EDK\FULL_TST\TestApp_Memory\src\TestApp_Memory_LinkScr.ld
I:\_MY_EDK\FULL_TST\TestApp_Peripheral\src\TestApp_Peripheral.c
I:\_MY_EDK\FULL_TST\TestApp_Peripheral\src\xintc_tapp_example.c
I:\_MY_EDK\FULL_TST\TestApp_Peripheral\src\intc_header.h
```

[✓] Save settings file:

I:_MY_EDK\FULL_TST\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.

Click **OK**



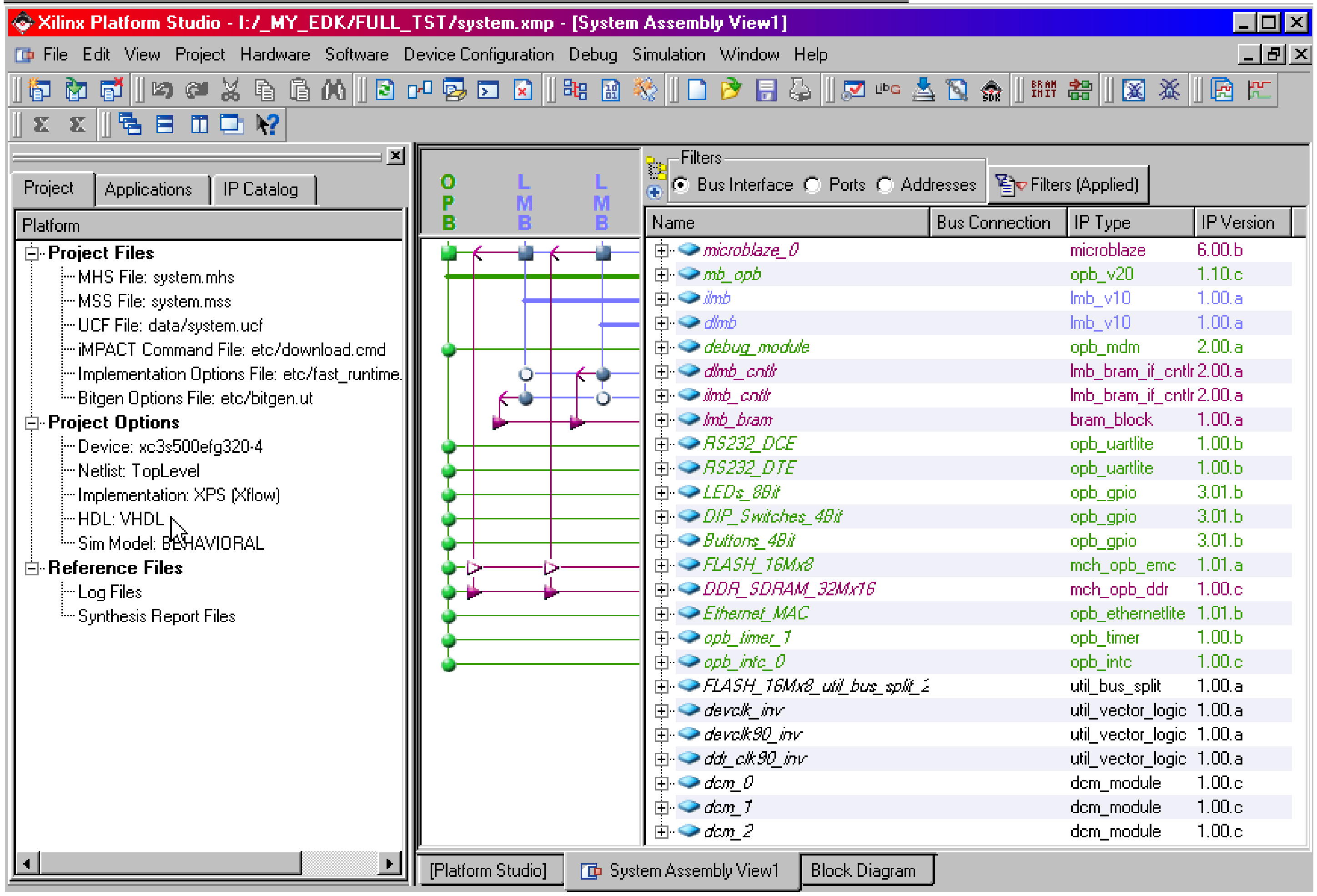
The Next Step

What would you like to do next?

- ( ) Configure drivers and libraries (Software Platform)
- ( ) Download the design to the board and test it
- ( ) Edit the test application generated by BSB
- [✓ Start using Platform Studio]

☐ Remember my selection and don't show this again

OK | Cancel



Xilinx Platform Studio - I:/_MY_EDK/FULL_TST/system.xmp - [System Assembly View1]

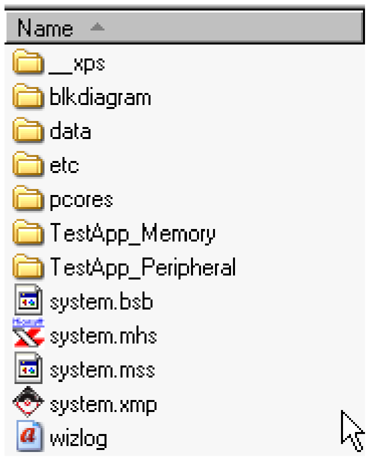
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project
Applications
IP Catalog
OPB
LMB
LMB
[Platform Studio]
System Assembly View1
Block Diagram

Platform
- Project Files
  - MHS File: system.mhs
  - MSS File: system.mss
  - UCF File: data/system.ucf
  - iMPACT Command File: etc/download.cmd
  - Implementation Options File: etc/fast_runtime.
  - Bitgen Options File: etc/bitgen.ut
- Project Options
  - Device: xc3s500efg320-4
  - Netlist: TopLevel
  - Implementation: XPS (Xflow)
  - HDL: VHDL
  - Sim Model: BEHAVIORAL
- Reference Files
  - Log Files
  - Synthesis Report Files

Filters: [✓ Bus Interface] Ports Addresses — Filters (Applied)

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	6.00.b
mb_opb		opb_v20	1.10.c
ilmb		lmb_v10	1.00.a
dlmb		lmb_v10	1.00.a
debug_module		opb_mdm	2.00.a
dlmb_cntlr		lmb_bram_if_cntlr	2.00.a
ilmb_cntlr		lmb_bram_if_cntlr	2.00.a
lmb_bram		bram_block	1.00.a
RS232_DCE		opb_uartlite	1.00.b
RS232_DTE		opb_uartlite	1.00.b
LEDs_8Bit		opb_gpio	3.01.b
DIP_Switches_4Bit		opb_gpio	3.01.b
Buttons_4Bit		opb_gpio	3.01.b
FLASH_16Mx8		mch_opb_emc	1.01.a
DDR_SDRAM_32Mx16		mch_opb_ddr	1.00.c
Ethernet_MAC		opb_ethernetlite	1.01.b
opb_timer_1		opb_timer	1.00.b
opb_intc_0		opb_intc	1.00.c
FLASH_16Mx8_util_bus_split_2		util_bus_split	1.00.a
devclk_inv		util_vector_logic	1.00.a
devclk90_inv		util_vector_logic	1.00.a
ddr_clk90_inv		util_vector_logic	1.00.a
dcm_0		dcm_module	1.00.c
dcm_1		dcm_module	1.00.c
dcm_2		dcm_module	1.00.c

NOTE : VHDL is recommended even it's possible to use Verilog, but unpreditable behavior, especially for user IP_core!!!



It creates
System file XMP, HW file MHS, SW file MSS,
./data/system.ucf
(i) .etc/fast_runtime.opt for options in creating bitstream and (ii) ./etc/ download.cmd for downloading bitstream

2. Include an IP core

FPGA has a limited number of clock buffers. A core of input_buf is used to prevent a clock signak using clock buffer. This job requires 2 steps
The core must be either in global or local **pcores** repository
The core info must be included into **MHS** file with some signal adaptions, eg external signal goes to the input_buf and then to the system module.

```
BEGIN opb_ethernetlite
 PARAMETER INSTANCE = Ethernet_MAC
 PARAMETER HW_VER = 1.01.b
 PARAMETER C_OPB_CLK_PERIOD_PS = 20000
 PARAMETER C_BASEADDR = 0x40e00000
 PARAMETER C_HIGHADDR = 0x40e0ffff
 BUS_INTERFACE SOPB = mb_opb
 PORT IP2INTC_Irpt = Ethernet_MAC_IP2INTC_Irpt
# PORT PHY_tx_clk = fpga_0_Ethernet_MAC_PHY_tx_clk
 PORT PHY_tx_clk = fpga_0_Ethernet_MAC_PHY_tx_clk_ibuf
# PORT PHY_rx_clk = fpga_0_Ethernet_MAC_PHY_rx_clk
 PORT PHY_rx_clk = fpga_0_Ethernet_MAC_PHY_rx_clk_ibuf
 PORT PHY_crs = fpga_0_Ethernet_MAC_PHY_crs
 PORT PHY_dv = fpga_0_Ethernet_MAC_PHY_dv
 PORT PHY_rx_data = fpga_0_Ethernet_MAC_PHY_rx_data
 PORT PHY_col = fpga_0_Ethernet_MAC_PHY_col
 PORT PHY_rx_er = fpga_0_Ethernet_MAC_PHY_rx_er
 PORT PHY_tx_en = fpga_0_Ethernet_MAC_PHY_tx_en
 PORT PHY_tx_data = fpga_0_Ethernet_MAC_PHY_tx_data
END

# # Add input_buf to use IBUF, instead of IBUFG, for clock
BEGIN input_buf
 PARAMETER INSTANCE = input_buf_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER DWIDTH = 1
 PORT I = fpga_0_Ethernet_MAC_PHY_rx_clk
 PORT O = fpga_0_Ethernet_MAC_PHY_rx_clk_ibuf
END
```

```
BEGIN input_buf
 PARAMETER INSTANCE = input_buf_1
 PARAMETER HW_VER = 1.00.a
 PARAMETER DWIDTH = 1
 PORT I = fpga_0_Ethernet_MAC_PHY_tx_clk
 PORT O = fpga_0_Ethernet_MAC_PHY_tx_clk_ibuf
END
```

3. UCF File

UCF file **system.ucf** is also copied from **data** folder to **implementation** one. So all changes must be in **data/system.ucf**

The following constraints must be included into UCF for DDR-SDRAM; otherwise therer is failure in memory test

```
## Added for DDR_SDRAM
NET "*/DDR_DQ_I*" IOBDELAY=NONE;
NET "*/DDR_DQ_O*" IOBDELAY=NONE;
NET "*/DDR_DQ_T*" IOBDELAY=NONE;
NET "*/DDR_DQS_I*" IOBDELAY=NONE;
NET "*/DDR_DQS_O*" IOBDELAY=NONE;
NET "*/DDR_DQS_T*" IOBDELAY=NONE;
```

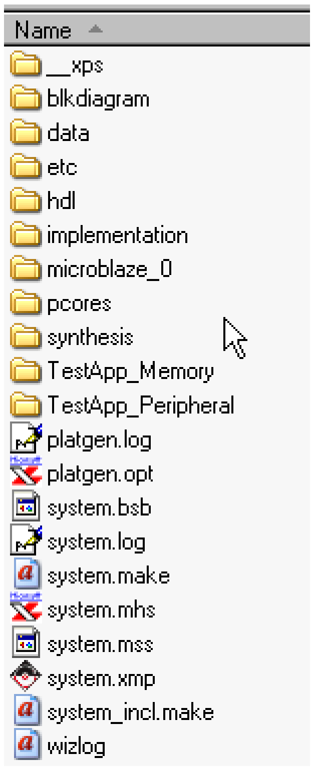
4. BitGen.ut

Change JTAGCLK to CCLK in file bitge.ut under folder etc as CCLK works for both JTAG and standalone. but JTAGCLK does not work in standalone mode.

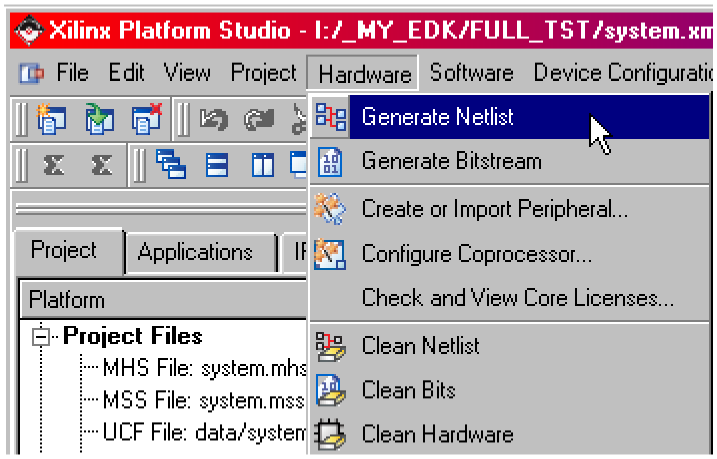
```
-g CclkPin:PULLUP
-g TdoPin:PULLNONE
-g M1Pin:PULLDOWN
-g DonePin:PULLUP
-g StartUpClk:JTAGCLK - CCLK
-g M0Pin:PULLUP
-g M2Pin:PULLUP
-g ProgPin:PULLUP
-g TckPin:PULLUP
-g TdiPin:PULLUP
-g TmsPin:PULLUP
-g LCK_cycle:NoWait
-g Security:NONE
#-m
-g Persist:No
```

5. Create NetList

5.1. GUI - 65 sec


Name
__xps
blkdiagram
data
etc
hdl
implementation
microblaze_0
pcores
synthesis
TestApp_Memory
TestApp_Peripheral
platgen.log
platgen.opt
system.bsb
system.log
system.make
system.mhs
system.mss
system.xmp
system_incl.make
wizlog

It creates system.make and system_incl.make, platgen.opt
./hdl/ for HDL codes like VHDL and V files
./synthesis/ for scripts like PRJ and SCR files
./implementation/ for netlist files like EDN file
SW test files like TestApp_Memory, TestApp_Peripheral



NOTE : all output data are saved into **system.log**

5.2. Cmd-Line – 54 sec

In console windowGo to Design folder to run XPS in NO-WINDOW mode : (1) system first and then (2) create netlist

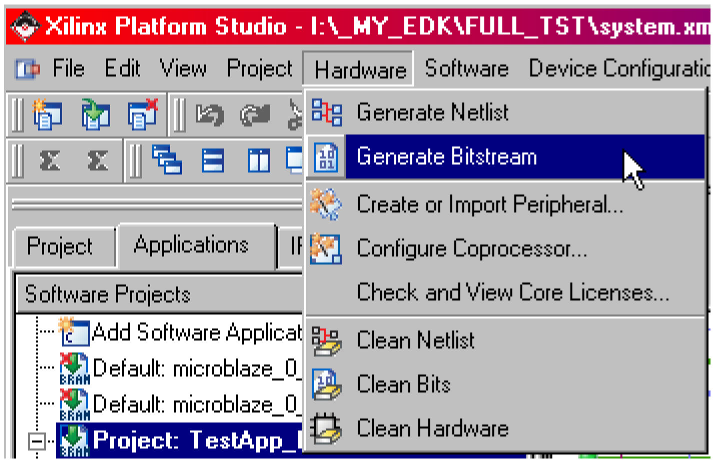
```
>> xps –nw
%% xload xmp system.xmp
%% run netlist
```

NOTE : The cmd-line way finishes the job sooner, but there's no data saved into **system.log** !!!

NOTE : Use **GUI** for detailed info of the process from **system.log**

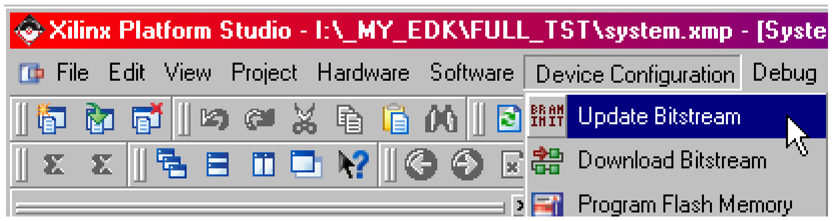
6. Generate system.bit – 358 secs using Make script

It creates **system.bit** after mapping and P&R. with info in **system_map.mrp** and **system.par**



7. Generate download.bit – 95 secs using Make script

It compiles startup code and some required driver for BRAM and comdines with **system.bit** to create **download.bit** ready going to the target

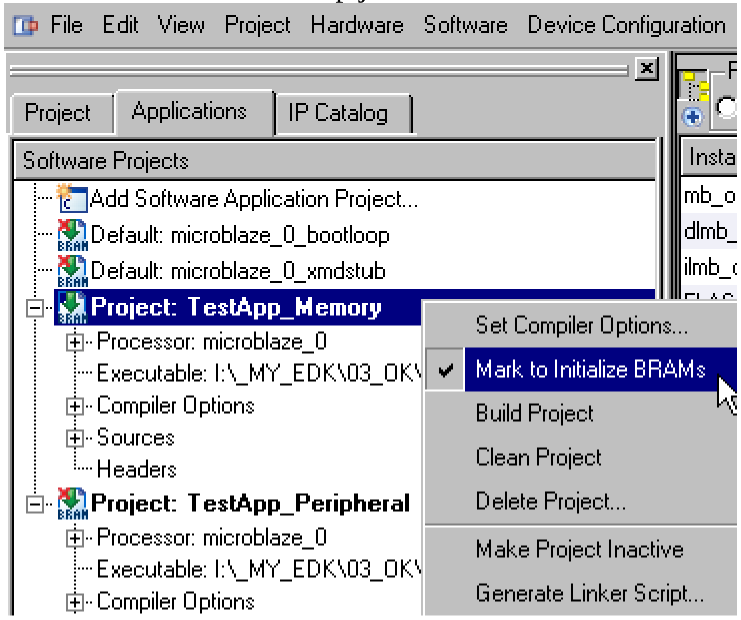


8. Download Bitstream – 95 secs using Make script

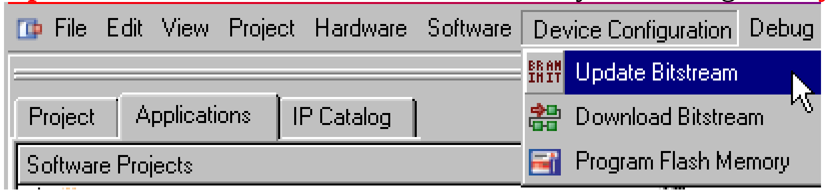
9. HW Test

The **download.bit** file going to the target is comprised of 2 components (1) HW system.bit (2) SW startup always in BRAM. The TestApp_Memory will be initialized in BRAM, it's executed right out of reset

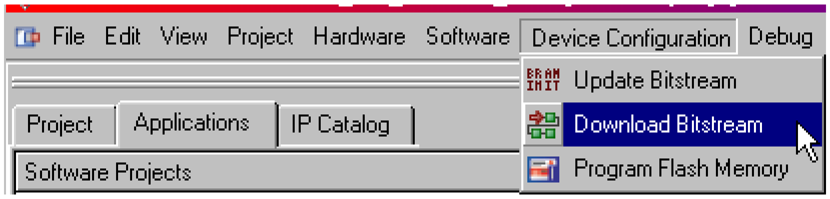
Mark BRAM for a startup job stored in BRAM


File Edit View Project Hardware Software Device Configuration
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: I:_MY_EDK\03_OK\
Compiler Options
Sources
Headers
Project: TestApp_Peripheral
Processor: microblaze_0
Executable: I:_MY_EDK\03_OK\
Compiler Options
Set Compiler Options...
Mark to Initialize BRAMs
Build Project
Clean Project
Delete Project...
Make Project Inactive
Generate Linker Script...

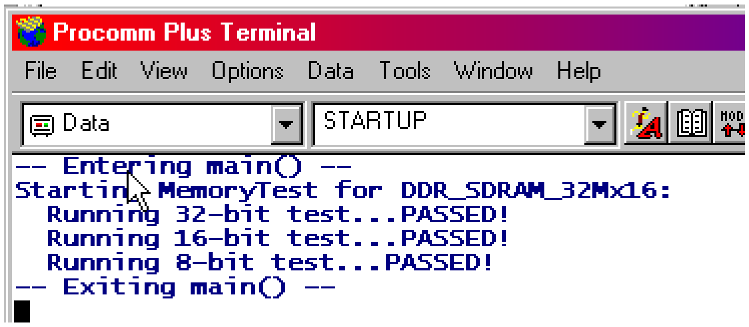
Update Bitstream to create **download.bit** by combining HW **system.bit** and SW startup **Initialize BRAM**


File Edit View Project Hardware Software Device Configuration Debug
Project Applications IP Catalog
Software Projects
Update Bitstream
Download Bitstream
Program Flash Memory

Download bitstream

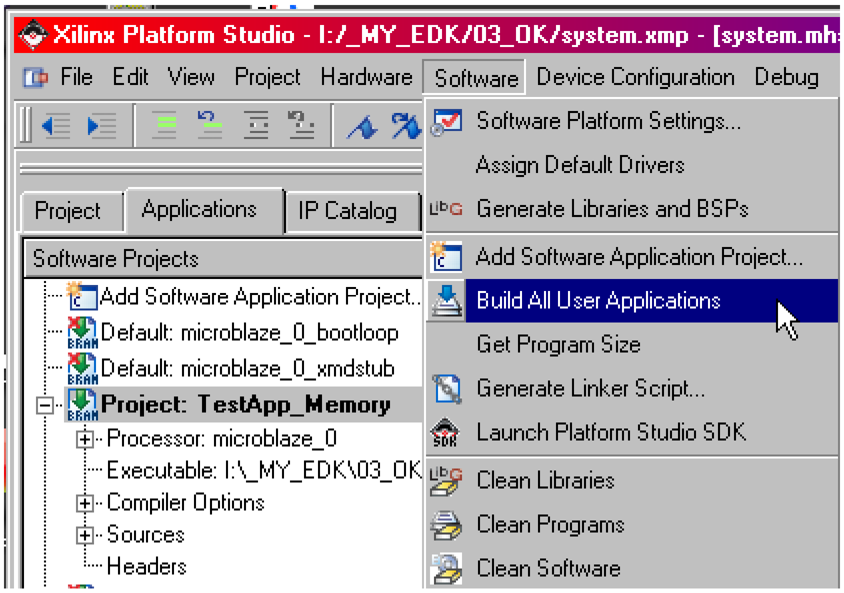

File Edit View Project Hardware Software Device Configuration Debug
Project Applications IP Catalog
Software Projects
Update Bitstream
Download Bitstream
Program Flash Memory

Test Result

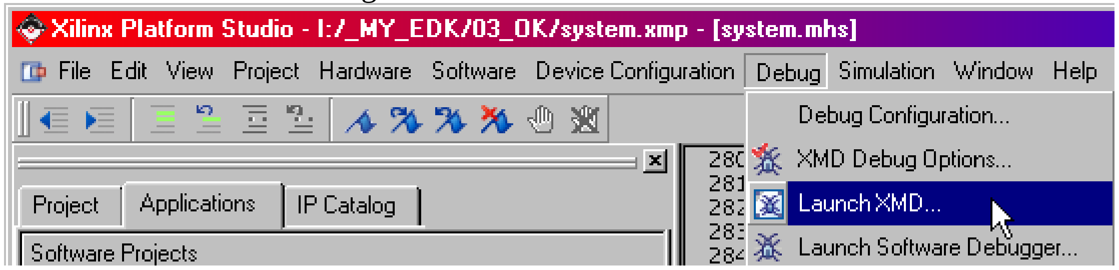


```
-- Entering main() --
Starting MemoryTest for DDR_SDRAM_32Mx16:
  Running 32-bit test...PASSED!
  Running 16-bit test...PASSED!
  Running 8-bit test...PASSED!
-- Exiting main() --
```

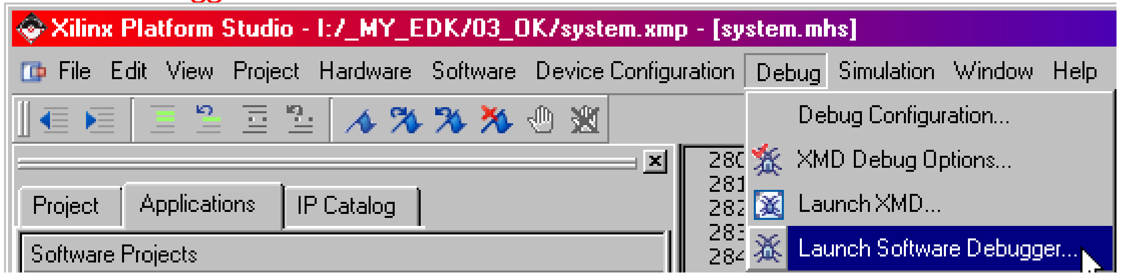
Build TestApp_Peripheral


Xilinx Platform Studio - I:/_MY_EDK/03_OK/system.xmp - [system.mh
File Edit View Project Hardware Software Device Configuration Debug
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: I:_MY_EDK\03_OK
Compiler Options
Sources
Headers
Software Platform Settings...
Assign Default Drivers
Generate Libraries and BSPs
Add Software Application Project...
Build All User Applications
Get Program Size
Generate Linker Script...
Launch Platform Studio SDK
Clean Libraries
Clean Programs
Clean Software

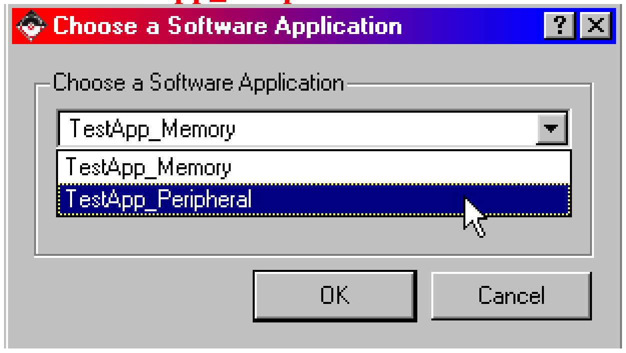
Launch XMD before Debug


Xilinx Platform Studio - I:/_MY_EDK/03_OK/system.xmp - [system.mhs]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Applications IP Catalog
Software Projects
Debug Configuration...
XMD Debug Options...
Launch XMD...
Launch Software Debugger...

Launch Debugger


Xilinx Platform Studio - I:/_MY_EDK/03_OK/system.xmp - [system.mhs]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Applications IP Catalog
Software Projects
Debug Configuration...
XMD Debug Options...
Launch XMD...
Launch Software Debugger...

Select TestApp_Peripheral

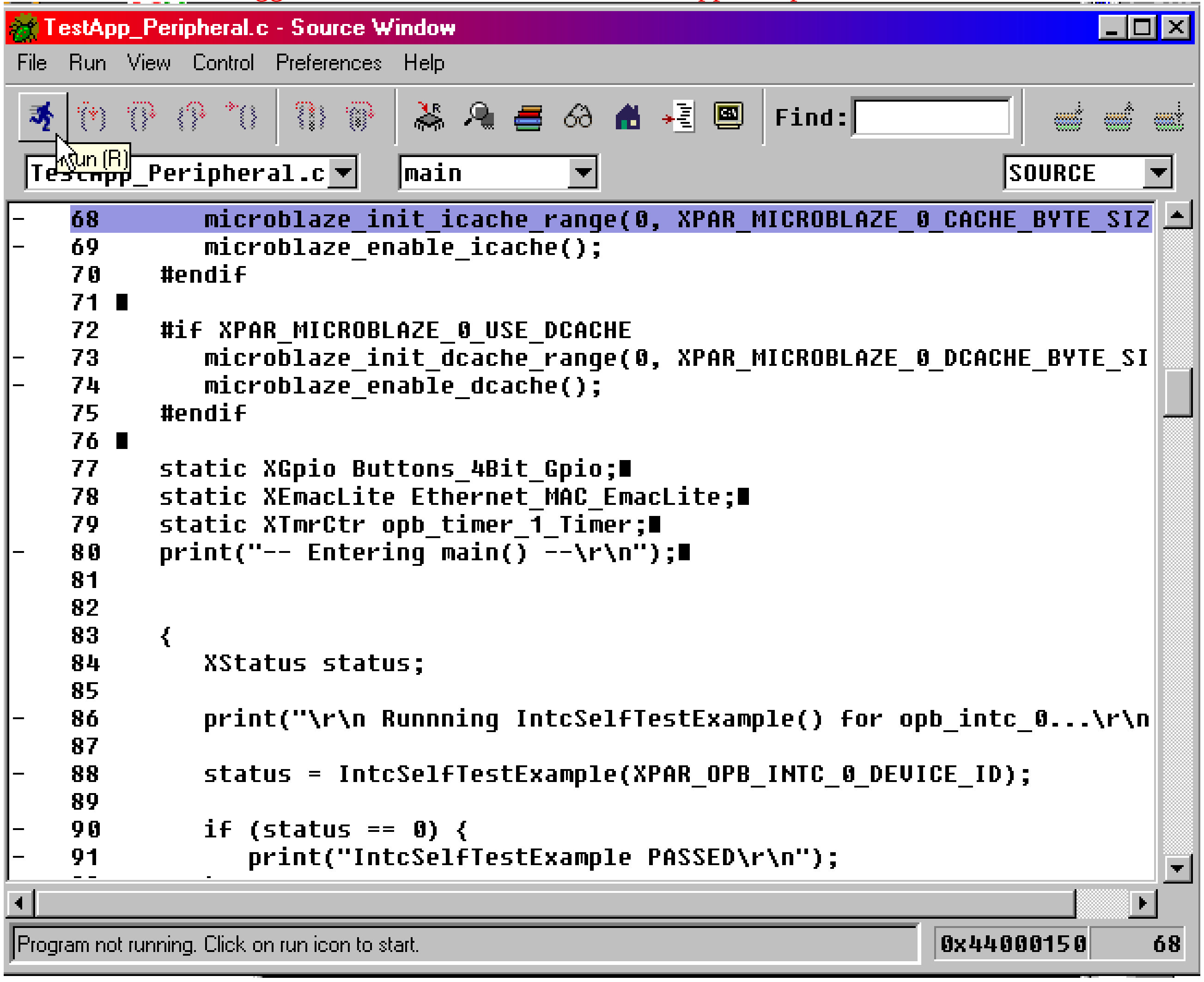


CRASH !?!? It works only for simple design

So, we have to run **XMD in command mode**

```
??> xmd
XMD% connect mb mdm
XMD% exit
??>
```

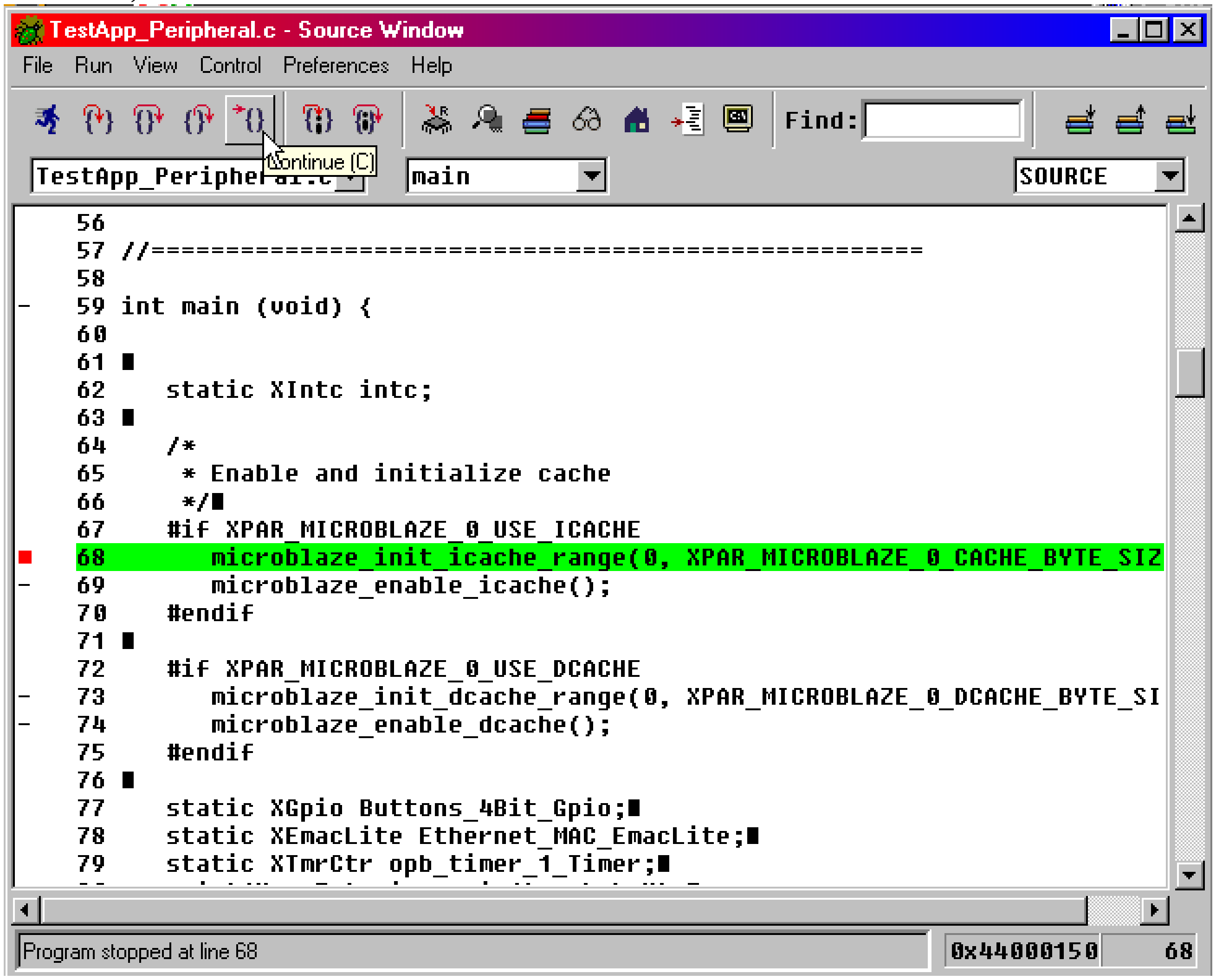
Now we launch debugger from GUI XPS and select TestApp_Peripheral, This time, it works


TestApp_Peripheral.c - Source Window
File Run View Control Preferences Help
Find:
Run (R)
TestApp_Peripheral.c
main
SOURCE

```
        microblaze_init_icache_range(0, XPAR_MICROBLAZE_0_CACHE_BYTE_SIZ
        microblaze_enable_icache();
     #endif

     #if XPAR_MICROBLAZE_0_USE_DCACHE
        microblaze_init_dcache_range(0, XPAR_MICROBLAZE_0_DCACHE_BYTE_SI
        microblaze_enable_dcache();
     #endif

     static XGpio Buttons_4Bit_Gpio;
     static XEmacLite Ethernet_MAC_EmacLite;
     static XTmrCtr opb_timer_1_Timer;
     print("-- Entering main() --\r\n");


     {
        XStatus status;

        print("\r\n Runnning IntcSelfTestExample() for opb_intc_0...\r\n

        status = IntcSelfTestExample(XPAR_OPB_INTC_0_DEVICE_ID);

        if (status == 0) {
           print("IntcSelfTestExample PASSED\r\n");
```

Program not running. Click on run icon to start. 0x44000150 68

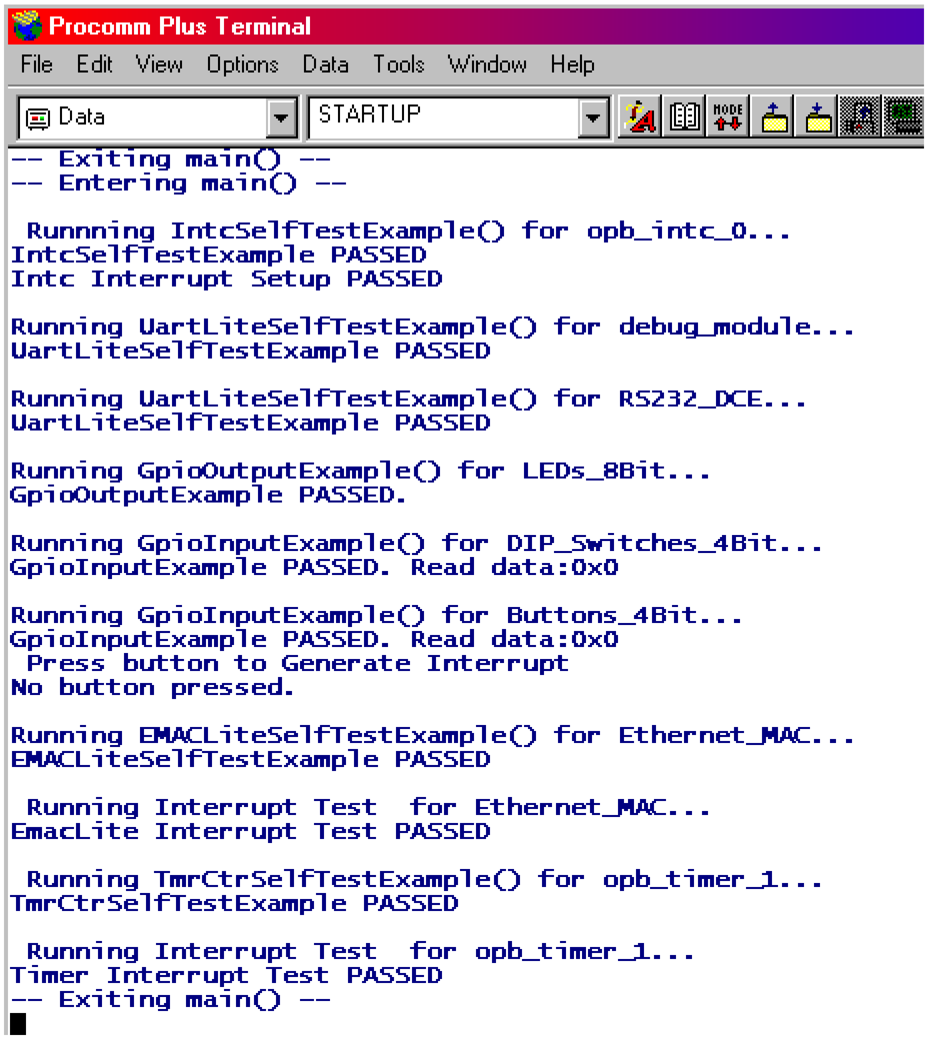
Click **Run**, to have


TestApp_Peripheral.c - Source Window
File Run View Control Preferences Help
Continue (C)
Find:
TestApp_Peripheral.c
main
SOURCE

```
//=====================================================

int main (void) {

   static XIntc intc;

   /*
    * Enable and initialize cache
    */
   #if XPAR_MICROBLAZE_0_USE_ICACHE
      microblaze_init_icache_range(0, XPAR_MICROBLAZE_0_CACHE_BYTE_SIZ
      microblaze_enable_icache();
   #endif

   #if XPAR_MICROBLAZE_0_USE_DCACHE
      microblaze_init_dcache_range(0, XPAR_MICROBLAZE_0_DCACHE_BYTE_SI
      microblaze_enable_dcache();
   #endif

   static XGpio Buttons_4Bit_Gpio;
   static XEmacLite Ethernet_MAC_EmacLite;
   static XTmrCtr opb_timer_1_Timer;
```

```
Program stopped at line 68        0x44000150        68
```

Click **Continue**, to have HW test pass successfully



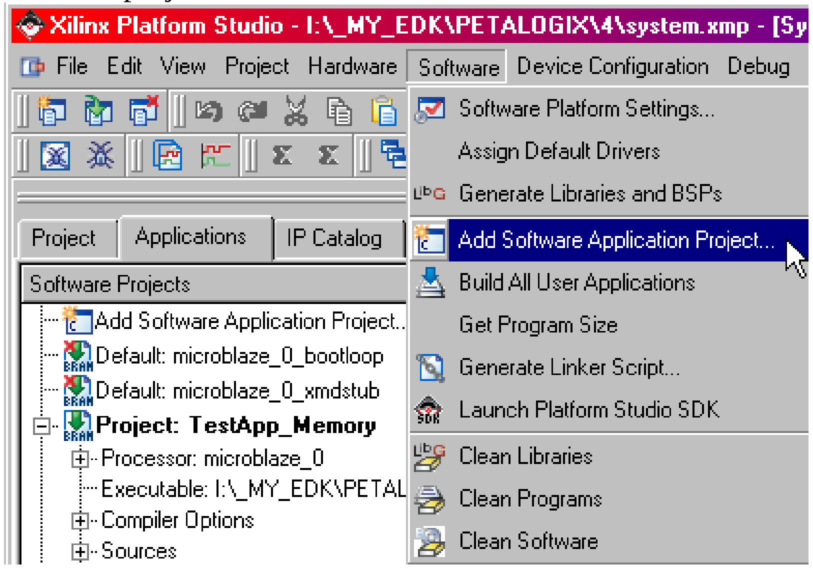
```
-- Exiting main() --
-- Entering main() --

 Runnning IntcSelfTestExample() for opb_intc_0...
IntcSelfTestExample PASSED
Intc Interrupt Setup PASSED

Running UartLiteSelfTestExample() for debug_module...
UartLiteSelfTestExample PASSED

Running UartLiteSelfTestExample() for RS232_DCE...
UartLiteSelfTestExample PASSED

Running GpioOutputExample() for LEDs_8Bit...
GpioOutputExample PASSED.

Running GpioInputExample() for DIP_Switches_4Bit...
GpioInputExample PASSED. Read data:0x0

Running GpioInputExample() for Buttons_4Bit...
GpioInputExample PASSED. Read data:0x0
 Press button to Generate Interrupt
No button pressed.

Running EMACLiteSelfTestExample() for Ethernet_MAC...
EMACLiteSelfTestExample PASSED

 Running Interrupt Test  for Ethernet_MAC...
EmacLite Interrupt Test PASSED

 Running TmrCtrSelfTestExample() for opb_timer_1...
TmrCtrSelfTestExample PASSED

 Running Interrupt Test  for opb_timer_1...
Timer Interrupt Test PASSED
-- Exiting main() --
```

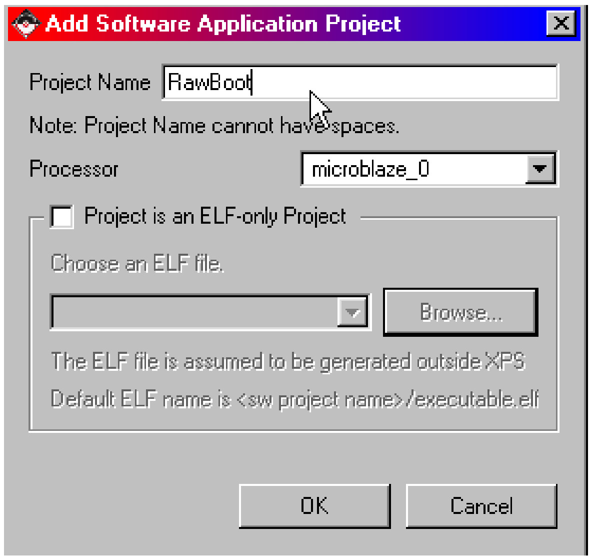
10. Standalone Flash-Based System

We will load TestApp_Peripheral into flash and have a simple bootloader RawBoot to load it into RAM and run out of RAM, after power up.

Add a SW project RawBoot as BootLoader. Bu default, TestApp_Memory is initialized in BRAM, so uncheck it, then nitialize RawBoot in BRAM, ie all others but RawBoot has **red mark**..

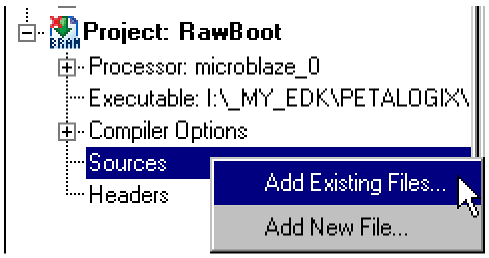
Add SW project RawBoot


Xilinx Platform Studio - I:_MY_EDK\PETALOGIX\4\system.xmp - [Sy
File Edit View Project Hardware Software Device Configuration Debug
Software Platform Settings...
Assign Default Drivers
Generate Libraries and BSPs
Add Software Application Project...
Build All User Applications
Get Program Size
Generate Linker Script...
Launch Platform Studio SDK
Clean Libraries
Clean Programs
Clean Software
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: I:_MY_EDK\PETAL
Compiler Options
Sources

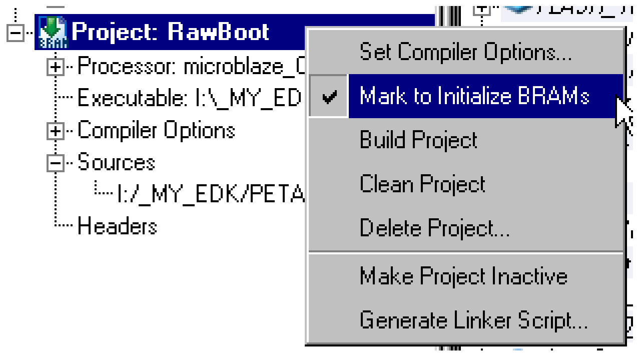

Add Software Application Project
Project is an ELF-only Project
Choose an ELF file.
Browse...
The ELF file is assumed to be generated outside XPS
Default ELF name is <sw project name>/executable.elf
OK
Cancel

Project Name: RawBoot
Note: Project Name cannot have spaces.
Processor: microblaze_0

Add source



Mark BRAM



Change FLASHBASE_ADDR to 0x4306000 in the file **RawBoot/src/flash_bootloader_mod.c**

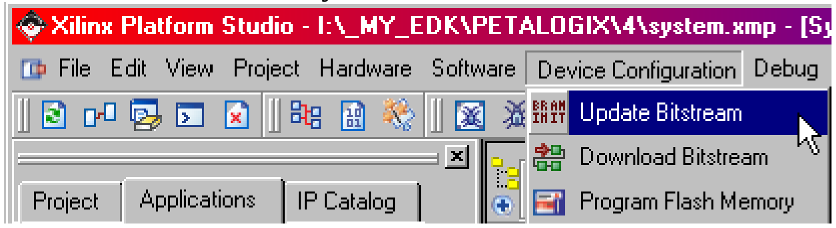
```
#define FLASH_BASE_ADDR 0x43060000
```

This is absolute address is where TestApp_periphearal starts in flash

Build the project TestApp_Peripheral to get ELF file and manualy convert into binary format using the command below

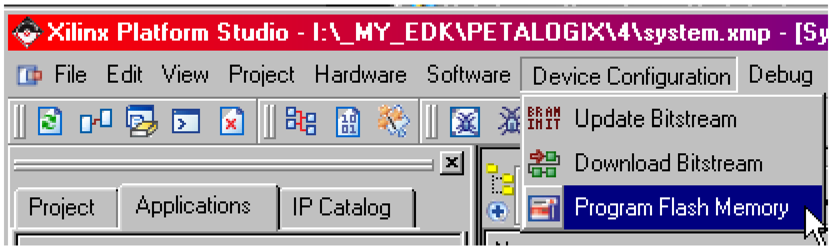
```
mb-objcopy –O bin executeable.elf TestApp_Peripheral.bin
```

Create **download.bit** = system.bit + RawBoot


Xilinx Platform Studio - I:_MY_EDK\PETALOGIX\4\system.xmp - [Sy
File Edit View Project Hardware Software Device Configuration Debug
Update Bitstream
Download Bitstream
Program Flash Memory
Project Applications IP Catalog

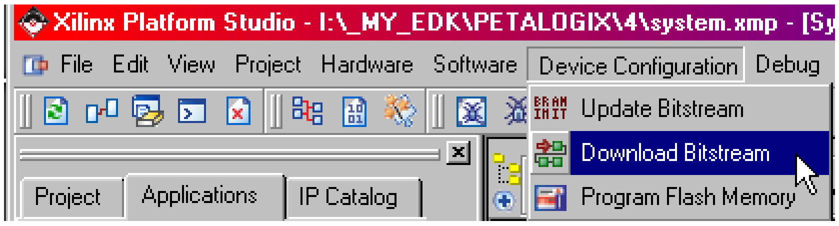
Convert download.bit to binary image for BPI-Up config mode

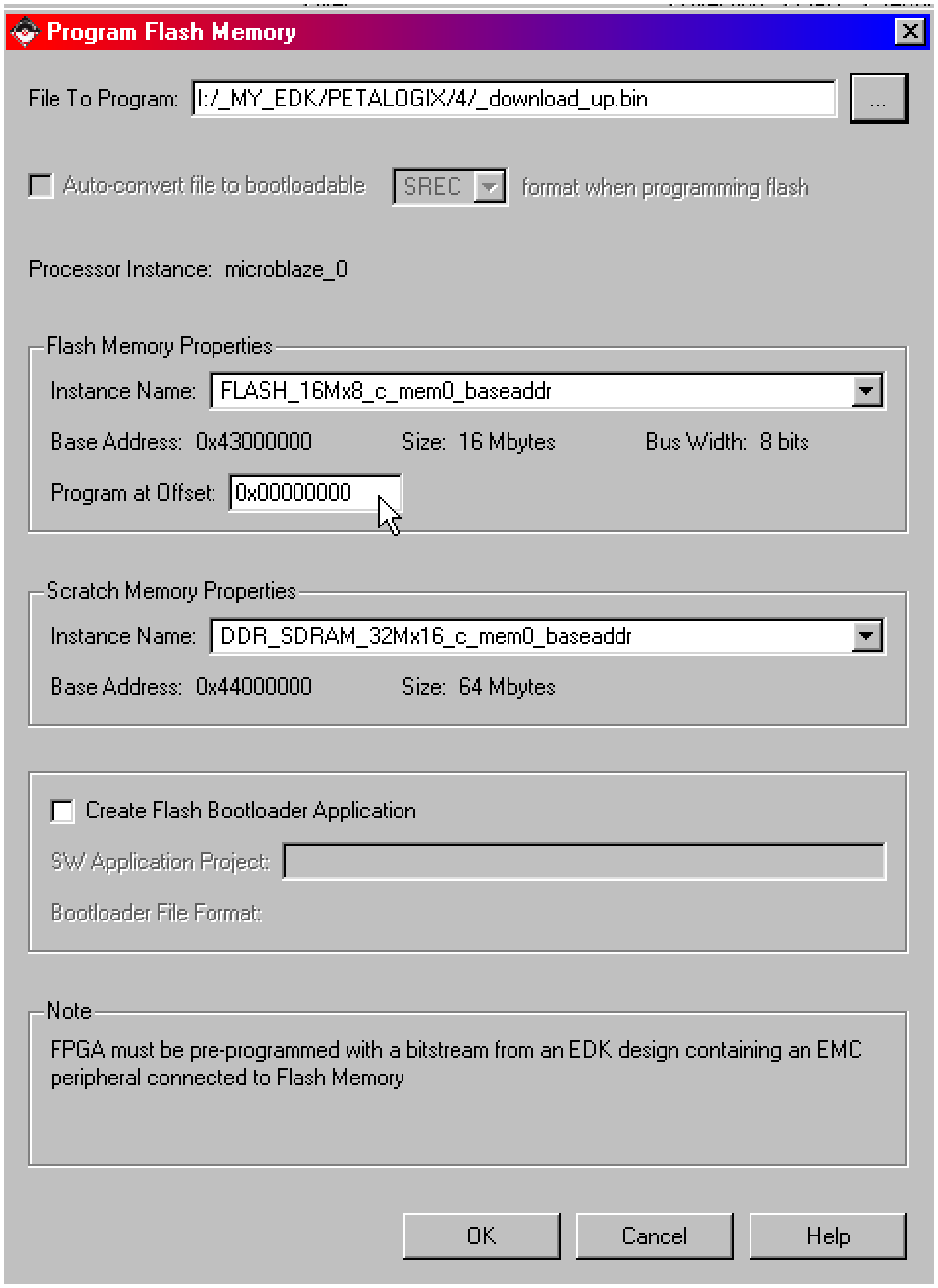
```
promgen -w -u bin -d 0 download.bit -o  downloadup
```


Xilinx Platform Studio - I:_MY_EDK\PETALOGIX\4\system.xmp - [Sy
File Edit View Project Hardware Software Device Configuration Debug
Update Bitstream
Download Bitstream
Program Flash Memory
Project Applications IP Catalog

We will download bitstream into FPGA to employ MicroBlaze in programming the flash

Restore TestApp_Memory to initialize in BRAM
Uncheck RawBoot.
Update bitstream


Xilinx Platform Studio - I:_MY_EDK\PETALOGIX\4\system.xmp - [Sy
File Edit View Project Hardware Software Device Configuration Debug
Update Bitstream
Download Bitstream
Program Flash Memory
Project Applications IP Catalog



Program Flash Memory

File To Program: I:/_MY_EDK/PETALOGIX/4/_download_up.bin

Auto-convert file to bootloadable SREC format when programming flash

Processor Instance: microblaze_0

Flash Memory Properties

Instance Name: FLASH_16Mx8_c_mem0_baseaddr

Base Address: 0x43000000 Size: 16 Mbytes Bus Width: 8 bits

Program at Offset: 0x00000000

Scratch Memory Properties

Instance Name: DDR_SDRAM_32Mx16_c_mem0_baseaddr

Base Address: 0x44000000 Size: 64 Mbytes

Create Flash Bootloader Application

SW Application Project:

Bootloader File Format:

Note

FPGA must be pre-programmed with a bitstream from an EDK design containing an EMC peripheral connected to Flash Memory

Image	Flash Offset	
Download_up.bin	0x0	
TestApp_Peripheral.bin	0x60000	

11. Summary

11.1. HW Platform Config

Below is a procedure to create HW platform to support PetaLinux where defaults are implied

Basic

- User repository for HW IP, like in_buf, and SW OS, like petalinux
- Spartan 3E 500 - D
- Cache Enable with 8 KB
- DCE/DTE : 115200, interrupt for both
- Flash : OPB EMC, not MCH
- EthernetLite with interrupt
- Peripheral Timer with interrupt
- 8 KB D/I-cache for DDR-SDRAM

Bug Fix

Modify MHS to use input_buf to prevent ethernet clk from using global buffers
Add DDR_SDRAM constraints to UCF
Connect to System

- IP Calatlog - Project Local pcores : Double click on USER_IP to add to System
- Connect to system
- sys_clk
- Set address

11.2. Download to Target

- Change JTAGCLK to CCLK in the file etc/bitgen.ut
- Generate Bitstream : system.bit for FPGA only
- Update Bitstream : download.bit comprised of system.bit and startup code TestApp_Mem BRAM
- Download bitstream : config FPGA and have microblaze to test memory
- Launch Debug and run TestApp_Peripheral

11.3. Programming Flash

- RawBoot initialized in BRAM. RawBot must have FLASH_BASE_ADDR = 0x43060000 for TestApp_Peripheral.bin
- Update Bitstream
- Convert download.bit to download_up.bin
- Re-initialize TestApp_Mem into BRAM. RawBoot must NOT in BRAM
- Update bitstream to get new download.bin
- Download bitstream to get MB ready
- Program download_up.bin
- Build TestApp_Peripheral to get execute.elf and convert it into binary file
- Load into flash offset 0x60000

12. Command Line Script

Create Netlist – Bits – Clean : ***_myedk.sh*** with choice therein

```
make -f system.make netlist | tee -a system.log
```

Download : **_DnLd_Bit.bat**

```
impact -batch etc/download.cmd
```

This download.bit has TestMemory marked in BRAM.

Flashing : **_mk_bpi_dn.bat, _do_Flash.bat, flashwriter.tcl etc/flashwriter**

The **Raw_Boot** is marked BRAM as boot loader. **Update** bitstream is used to create a flash version of download.bit which will be converted into flash image of BPI_Down mode using **_mk_bpi_dn.bat**, ie it resides in bottom of the flash where the first byte of the image is the last location of the flash. Therefore, the starting address in the flash is calculated from the file size which is a consatnt for a specific FPGA part, eg XC3S1600E is 746,212 bytes always.

A RAM download.bit must be downloaded to bring up MB using _DnLdBit.bat, then _doFlash.bat is employed for flashing. All flashing parameters are in the TCL script flash_params.tcl , like file name, programming address, …