EDK Base System Builder (BSB) support for XUPV2P Board

Xilinx University Program

May 2005

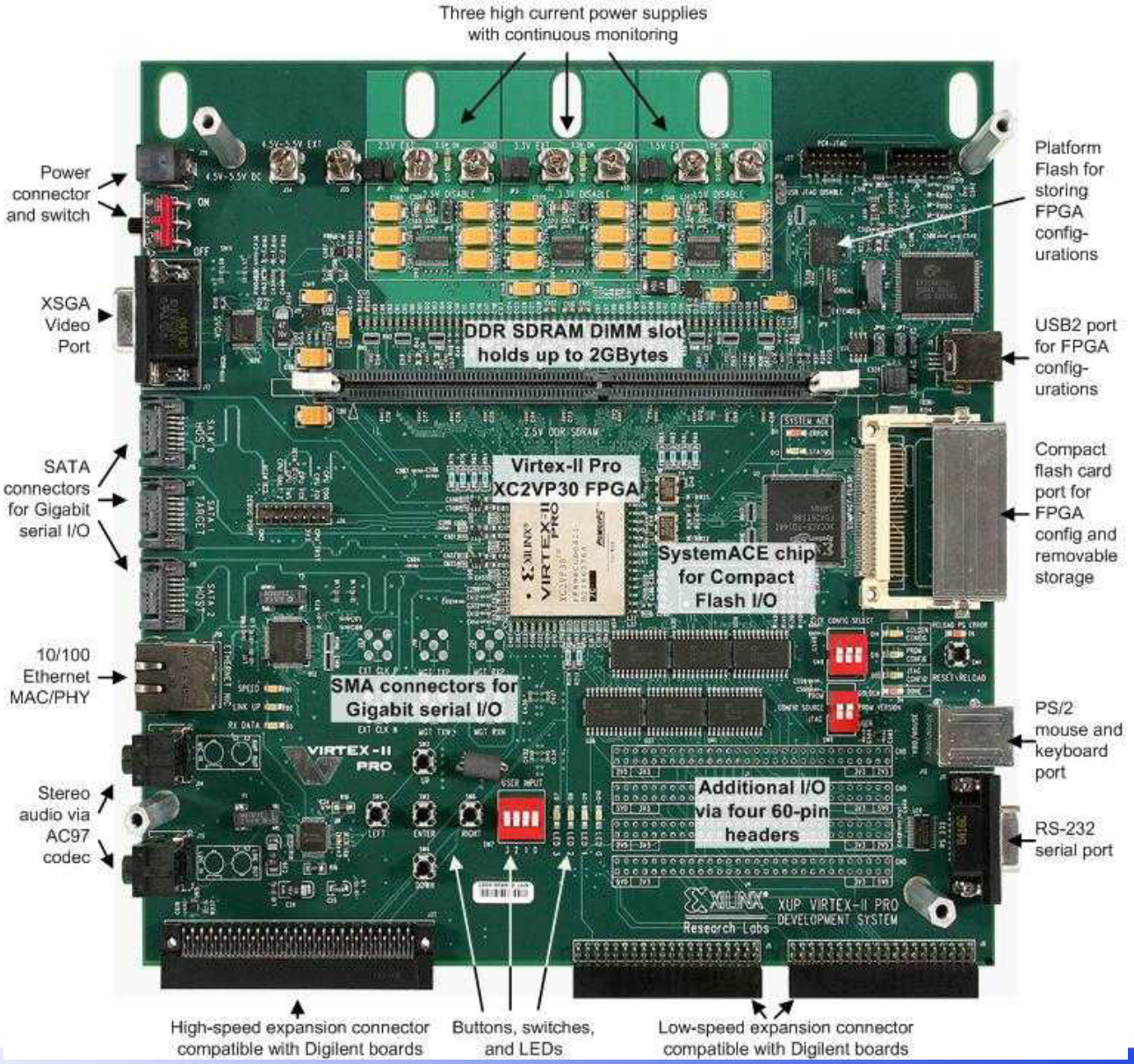
What is BSB?

- The Base System Builder (BSB) wizard is a software tool that help users quickly build a working system targeted at a specific development board.
- Based on the user's board selection, BSB will offer the user a number of options for creating a basic system on that board. These options include processor type, debug interface, cache configuration, memory type and size, and peripheral selection. For each option, functional default values will be preselected in the GUI.
- Upon exit of BSB, a hardware specification (MHS) file and software specification (MSS) file will be created and loaded into the user's XPS project. The user may then optionally further enhance the design in the Xilinx Platform Studio (XPS) GUI.
- The Base System Builder will also optionally generate a software project called "TestApp" which contains a sample application and linker script and can be compiled and run on the hardware on the target development board. Note that XPS supports multiple software projects for every hardware system, each of which may contain its own set of source files and linker script.
- Chapter 3 of the EDK System Tools Manual is good reference:
  - http://www.xilinx.com/ise/embedded/est_rm.pdf

Objective

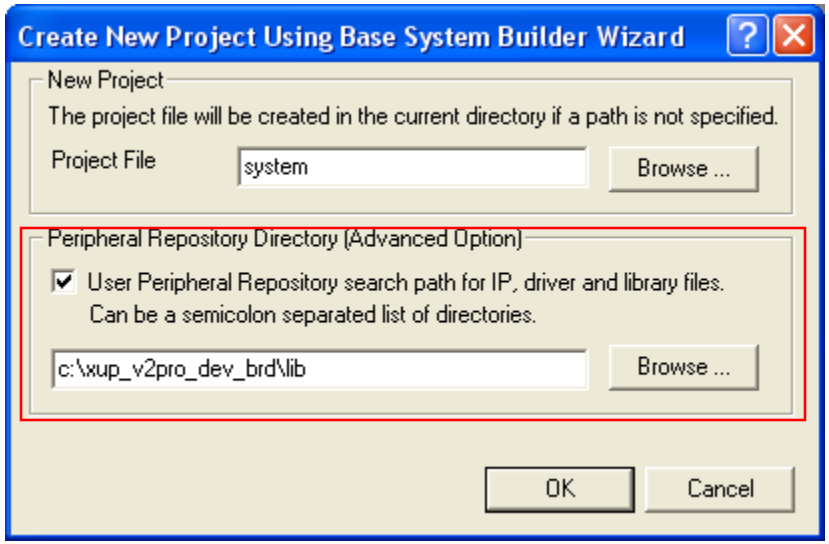
- Use a BSB design (or derivative) as the basis for:
  - Standalone processor based designs
  - Board Support Packages for PP405 Linux and Microblaze uCLinux
- Since it is a general tool, BSB designs are not optimum for every configuration but provide a starting point for further development since it provides reasonable defaults for all parameters not changed by the user

XUPV2P Development System



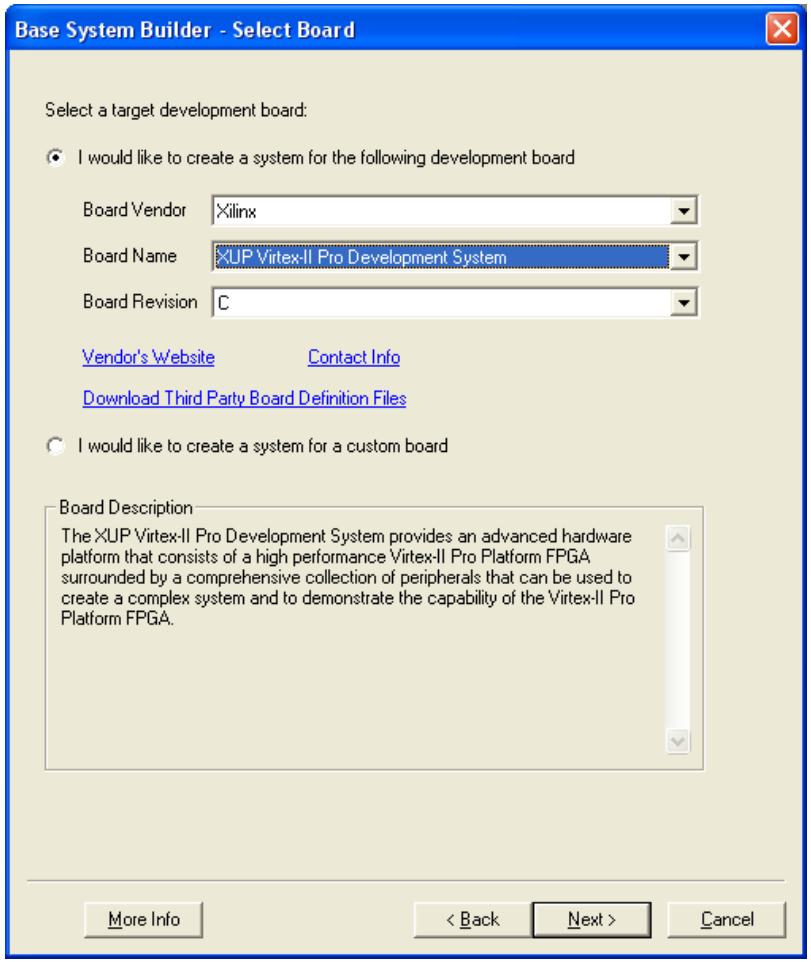
Mini-Howto

- Use EDK 7.1 SP1 (H.11.3) and ISE 7.1 SP2 (H.40)
- Launch EDK Platform Studio (XPS) and select BSB flow
  - Point the User Peripheral Repository Directory to the EDK XUP-V2P support files



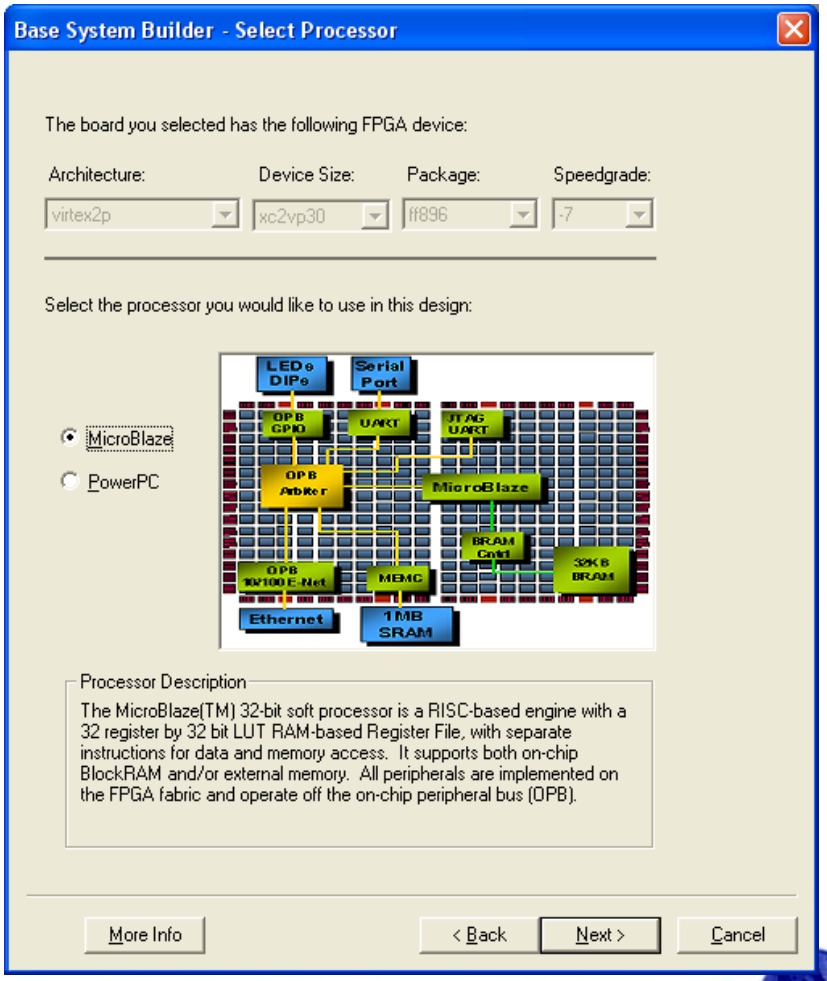
BSB Board Selection

- Select “I would like to create a new design” versus using a previous BSB session as a starting point
- The “XUP Virtex-II Pro Development System” should be listed under the Xilinx board vendor



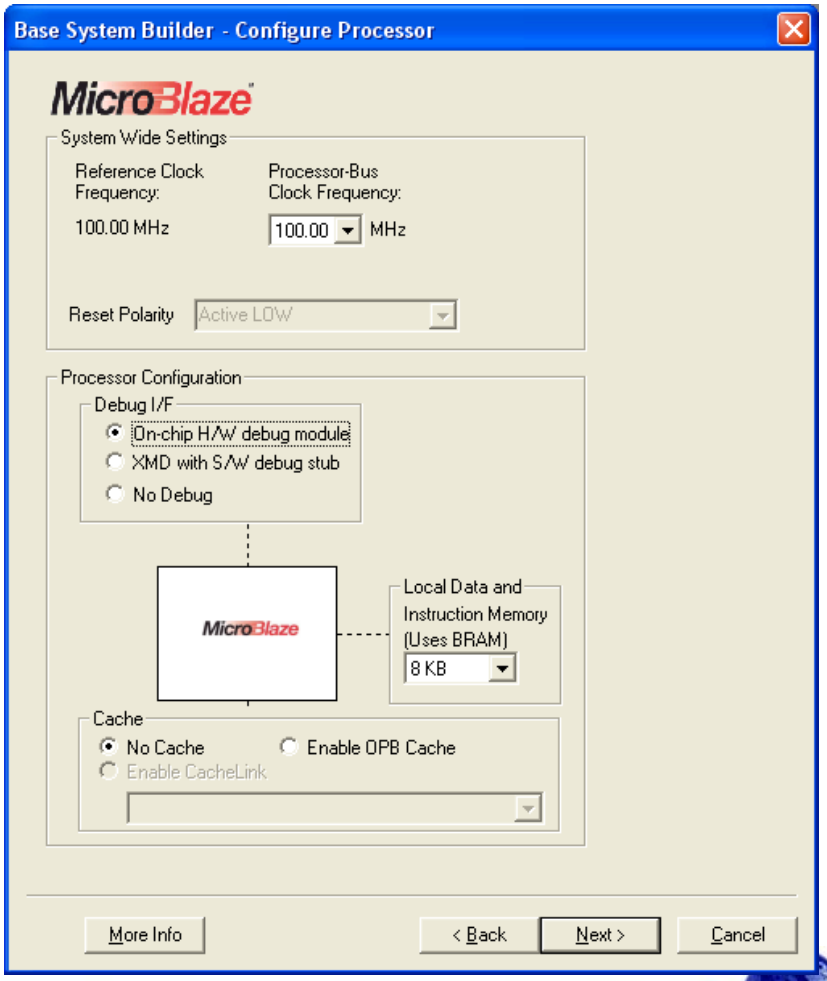
BSB Processor Selection

- BSB supports both the PowerPC405 and Microblaze processors, select the Microblaze processor for now



BSB Processor Options

- Accept the default Microblaze Processor Options



BSB Peripheral Selection

- The user can now include various peripherals provided on the board and select among parameters for each peripheral
  - Select to include RS232_UART_1, LEDs_4Bit, DIPSWs_4Bit, and PushButtons_5Bit
- BSB will optionally create example software applications (TestApps)
  - Accept default options

BSB System Overview

- Finally BSB lists the system configuration summary for the generated design

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: 100.000000 MHz
Debug interface: On-Chip HW Debug Module
On Chip Memory : 8 KB

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_Uart_1	0x40600000	0x4060FFFF
opb_gpio	LEDs_4Bit	0x40020000	0x4002FFFF
opb_gpio	DIPSWs_4Bit	0x40040000	0x4004FFFF
opb_gpio	PushButtons_5Bit	0x40000000	0x4000FFFF

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

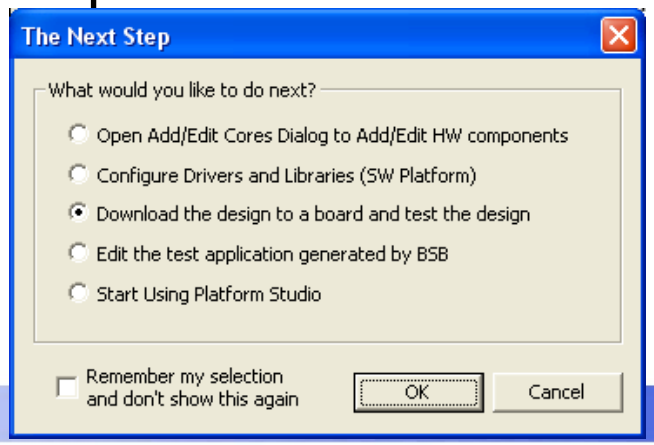
Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00001FFF

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

EDK Xilinx Platform Studio (XPS)

- After BSB finishes, XPS provides several options for the next path → Select Download the design
  - Any of these operations can be performed from the main XPS window
  - However, before the design can be downloaded, it must first be implemented

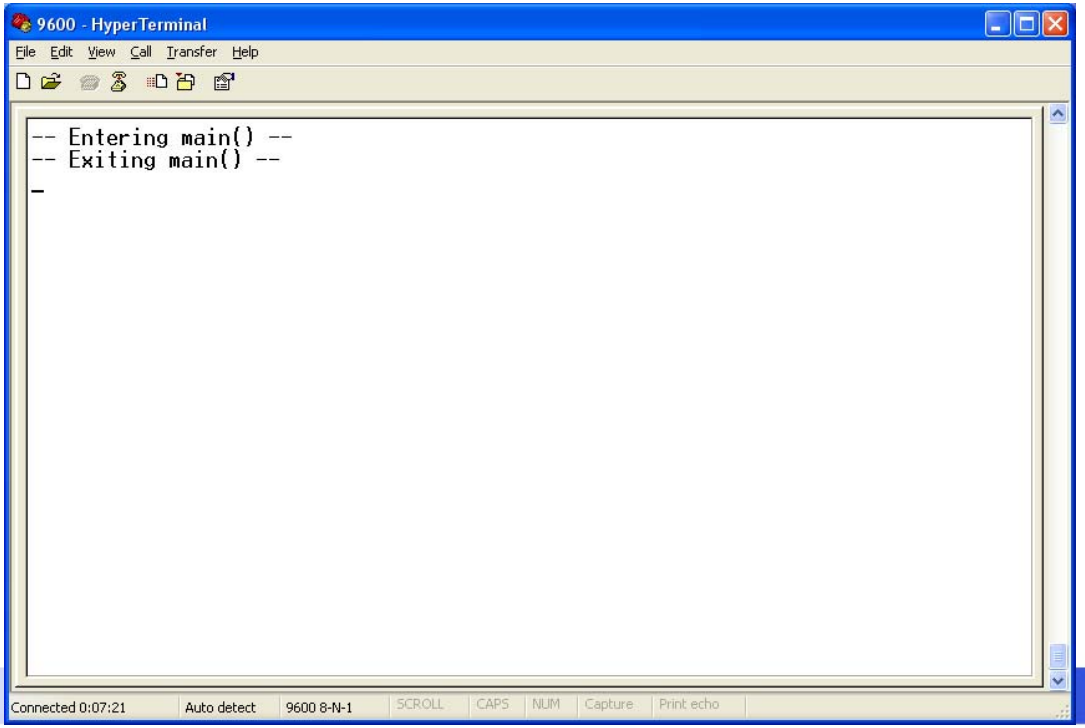


Design Implementation

- A bunch of things happen under the hood!!
  - An HDL representation of the design will be created in the hdl directory
  - Each submodule is synthesized into netlists stored in subdirectories under implementation
  - Ngdbuild combines the netlists and performs DRCs
  - The netlist is placed and routed (par) and a bitfile is generated
  - The software device drivers are compiled into libraries
  - The user application is compiled and linked against the libraries to created an executable elf file
  - Finally, the BRAMs in the bit file which comprise the program memory are configured with the contents of the elf file
  - Implementation/download.bit is the file to program the FPGA!

Serial Output

- Connect a RS232 serial cable from the XUP-V2P board serial port to the PC
- Open a terminal (i.e., HyperTerminal) for 9600baud, 8data bits, No Parity, 1 Stop bit and No flow control



You've Done It!

Congratulations!

You have just implemented a System-on-Chip Design using a Xilinx FPGA!

TestApplication

- BSB also generates a simple test application
  - Each IO EDK peripheral is wiggled by software
  - LEDs are flashed and the Switches/Pushbuttons are read
- First, enable the TestApp_Peripheral
- Then, uncomment the xil_printfs so that the state of the switches/pushbuttons are displayed

Interfacing with Peripherals

1) Disable "Mark to Initialize BRAMs"

1) Enable "Mark to Initialize BRAMs"

Xilinx Platform Studio - H:\mlxup\temp_tests\remote_xps\system.xmp - [(definition) [Platform Studio]]

File Edit View Project Tools Options Window Help

Applications

Right Click for Options

Software Projects
Default: microblaze_0_bootloop
Default: microblaze_0_xmdstub
Project: TestApp_Memory
Processor: microblaze_0
Executable: H:\mlxup\temp_
Compiler Options
Sources
Headers
Project: TestApp_Peripheral
Processor: microblaze_0
Executable: H:\mlxup\ter
Compiler Options
Sources
Headers

Set Compiler Options...
Mark to Initialize BRAMs
Build Project
Clean Project
Delete Project...
Make Project Inactive
Generate Linker Script...

Show

Platform Studio

Documentation | Examples | Tech Tips

START

Software Design | System Creation (blank project / base system builder) | Hardware Design

- Using XPS

Customize System
- Add processor and peripherals
- Create custom IP
- Component configuration and connection

Hardware verification using

(definition) [P...

```
INFO:iMPACT:580 - '3':Checking done pin ....done.
'3': Programmed successfully.
Elapsed time =      4 sec.
// *** BATCH CMD : quit
```

Output | Warning | Error

TestApplication Modification

Xilinx Platform Studio - H:\mlxup\temp_tests\remote_xps\system.xmp - [H:\\mlxup\temp_tests\remote_xps\TestApp_Peripheral\src\TestAp...

File Edit View Project Tools Options Window Help

System | Applications

Right Click for Options

- Software Projects
- Default: microblaze_0_bootloop
- Default: microblaze_0_xmdstub
- **Project: TestApp_Memory**
  - Processor: microblaze_0
  - Executable: H:\mlxup\temp_
  - Compiler Options
  - Sources
  - Headers
- **Project: TestApp_Peripherа**
  - Processor: microblaze_0
  - Executable: H:\mlxup\temp_
  - Compiler Options
  - Sources
    - H:\\mlxup\temp_tests\re
    - H:\\mlxup\temp_tests\re
  - Headers

```
101             print("UartLiteSelfTestExample FAILED\r\n");
102         }
103     }
104
105     /*
106      * Peripheral SelfTest will not be run for RS232_Uart_1
107      * because it has been selected as the STDOUT device
108      */
109
110
111     WriteToGPOutput(XPAR_LEDS_4BIT_BASEADDR, 4);
112
113     {
114         Xuint32 data = ReadFromGPInput(XPAR_DIPSWS_4BIT_BASEADDR);
115         //xil_printf("Data read from DIPSWs_4Bit: 0x%x\r\n", data);
116     }
117
118     {
119         Xuint32 data = ReadFromGPInput(XPAR_PUSHBUTTONS_5BIT_BASEADDR);
120         //xil_printf("Data read from PushButtons_5Bit: 0x%x\r\n", data);
121     }
122
123     print("-- Exiting main() --\r\n");
124     return 0;
125 }
126
127
```

(definition) [... | H:\\mlxup\te...

```
// *** BATCH CMD : quit
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Done.
```

Output | Warning | Error

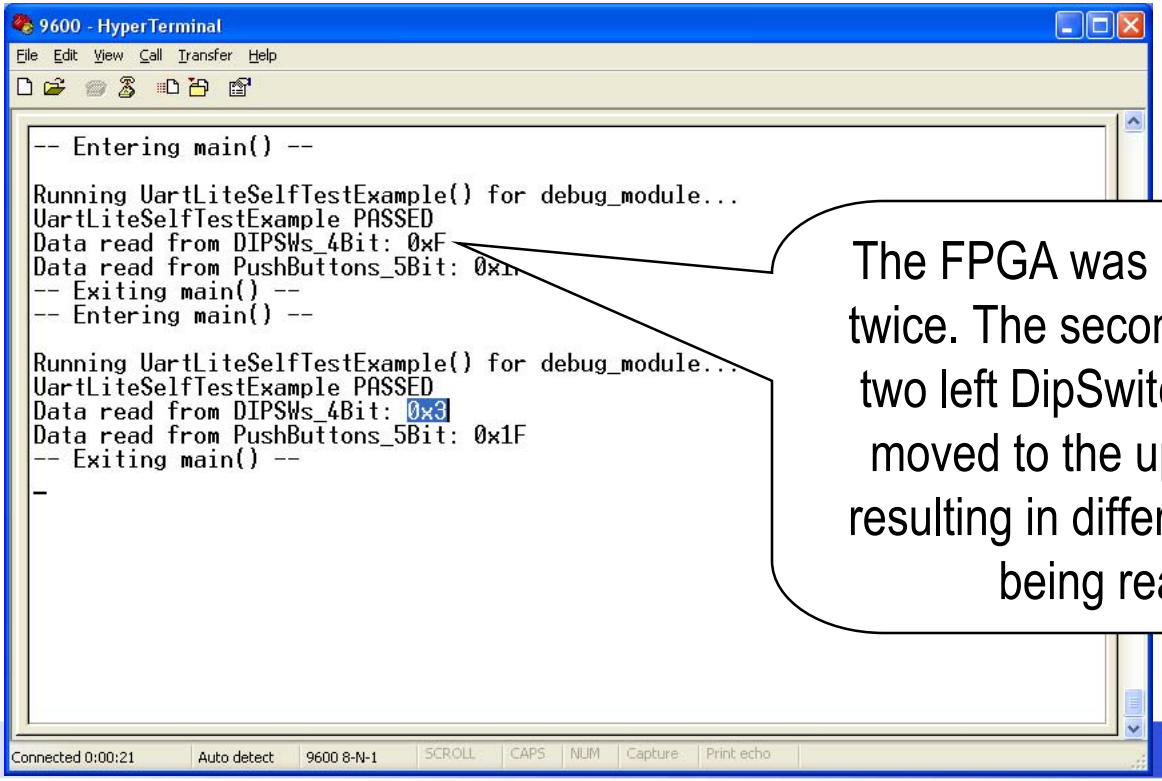
Ready

Uncomment to print values

Uncomment to print values

Downloading the TestApp

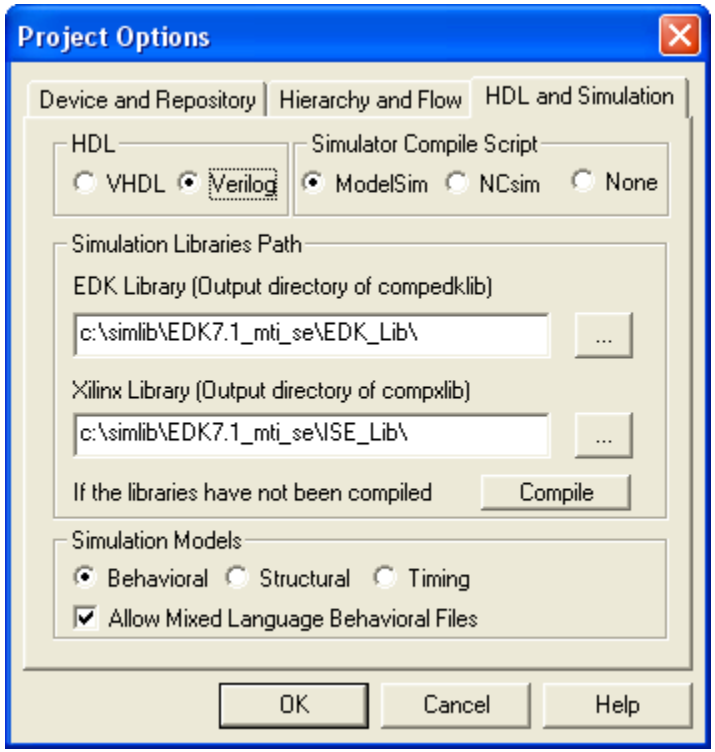
- Select Tools→Download and the software is recompiled, programmed into the FPGA bitstream, and then downloaded to the board



The FPGA was configured twice. The second time the two left DipSwitches were moved to the up position resulting in different values being read.

Simulation Setup

- Select Options→Project Options and either point to already compiled libraries or compile them new
  - Note: For PowerPC405 simulations, also need to set up SmartModel SWIFT support. See “Simulation in EDK” in the “Platform Studio User Guide”



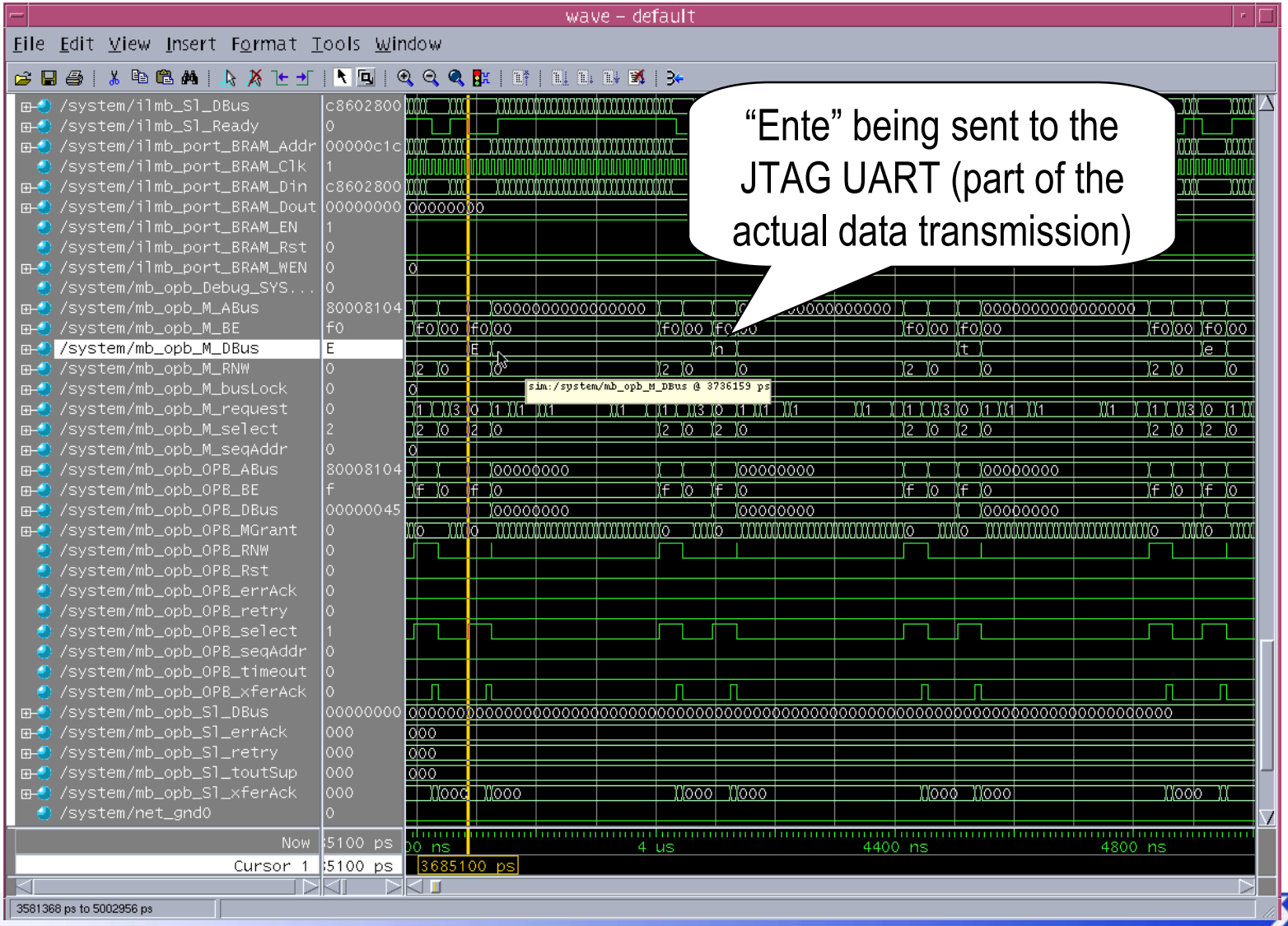
Simulation Script

- Select Tools→Start HDL Simulator
  - Enter the following commands (or as a do script) at the modelsim prompt
  - Note: Do not have external peripherals modeled

Modelsim Simulation Commands

```
# Make sure to set EDK to compile for verilog

do system.do
vsim system system_conf glbl
add wave -radix hexadecimal /*

# Enable the viewing of the Microblaze registers ...
set mb_register_path "/system/microblaze_0/microblaze_0/data_flow_i"

# ... add them to the modelsim view window
quietly WaveActivateNextPane
add wave -noupdate -divider {CPU Registers}
add wave -radix hexadecimal -label PC $mb_register_path/pc_ex
add wave -radix hexadecimal -label MSR $mb_register_path/msr

force sys_rst_pin 0, 1 1us
force sys_clk_pin 0, 1 5ns -r 10ns
run 100us
```

Waveform Output



"Ente" being sent to the JTAG UART (part of the actual data transmission)

Design Size and Implementation Time

- Design Size: Currently only 3% of 2VP30 FPGA for entire Microblaze design
  - There is a lot of room for more peripherals!
- Implementation Time from start to finish:

  Half Hour!