The Xilinx EDK Toolset: Xilinx Platform Studio (XPS)

Building a Base System Platform

By Jason Agron

What is Xilinx EDK?

- EDK = **E**mbedded **D**evelopment **K**it.
- It is a set of tools used to build embedded processing systems.
  - i.e. Systems-On-Chip (SoCs).
    - Processors (MicroBlaze, PowerPC).
    - Interconnect (PLB, OPB, FSL, Custom, etc.).
    - Memories (BRAM, DDR).
    - Peripherals (UART, GPIO, Ethernet, Custom, etc.).
- Provides a single environment for…
  - Simulation
  - Synthesis.
  - Compilation.

How Do I Use Xilinx EDK?

- Xilinx Platform Studio (XPS) - the actual tool.
  - Design flow…
    - First, create the hardware platform.
      - ➢ Select all of the peripherals.
      - ➢ Connect all of the peripherals.
    - Second, create the software for the platform.
      - ➢ Write SW to “make things work”.
    - Iterate if needed.
- The FPGA has a malleable fabric…
  - So both SW and HW are flexible and can be changed…
    - At “compile-time”.
    - At “run-time” (dynamic reconfiguration).

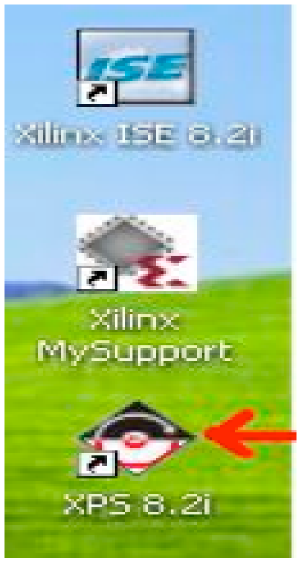
Important EDK Files

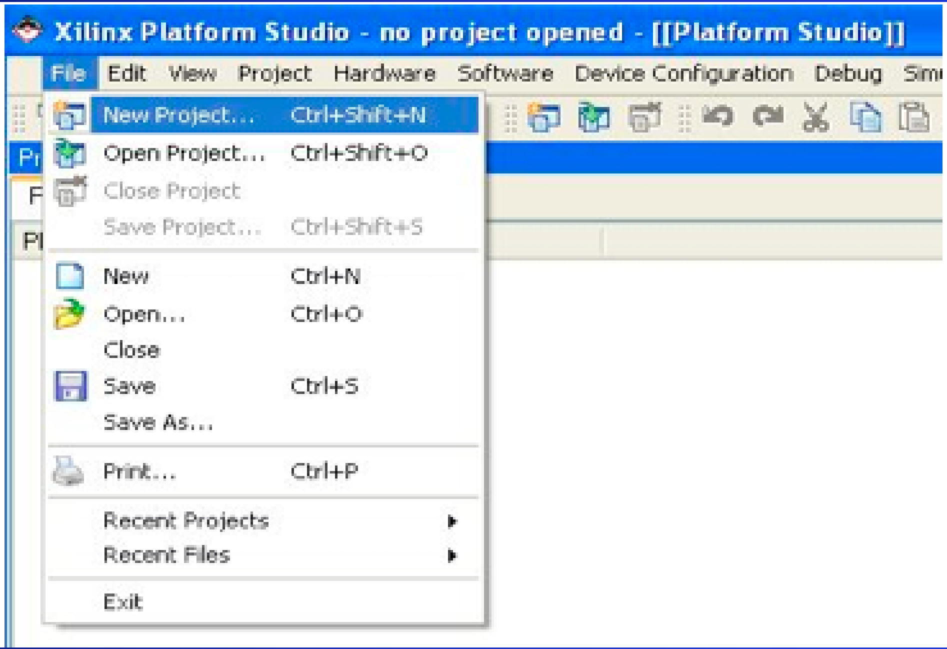
- MHS File:
  - Describes all components and connections in a system.
- MSS File:
  - Describes all SW drivers associated with components of a system.
- UCF File:
  - Describes the connections of all top-level ports.
  - All top-level ports have connections to specific physical pins on the FPGA.

How To Get Started

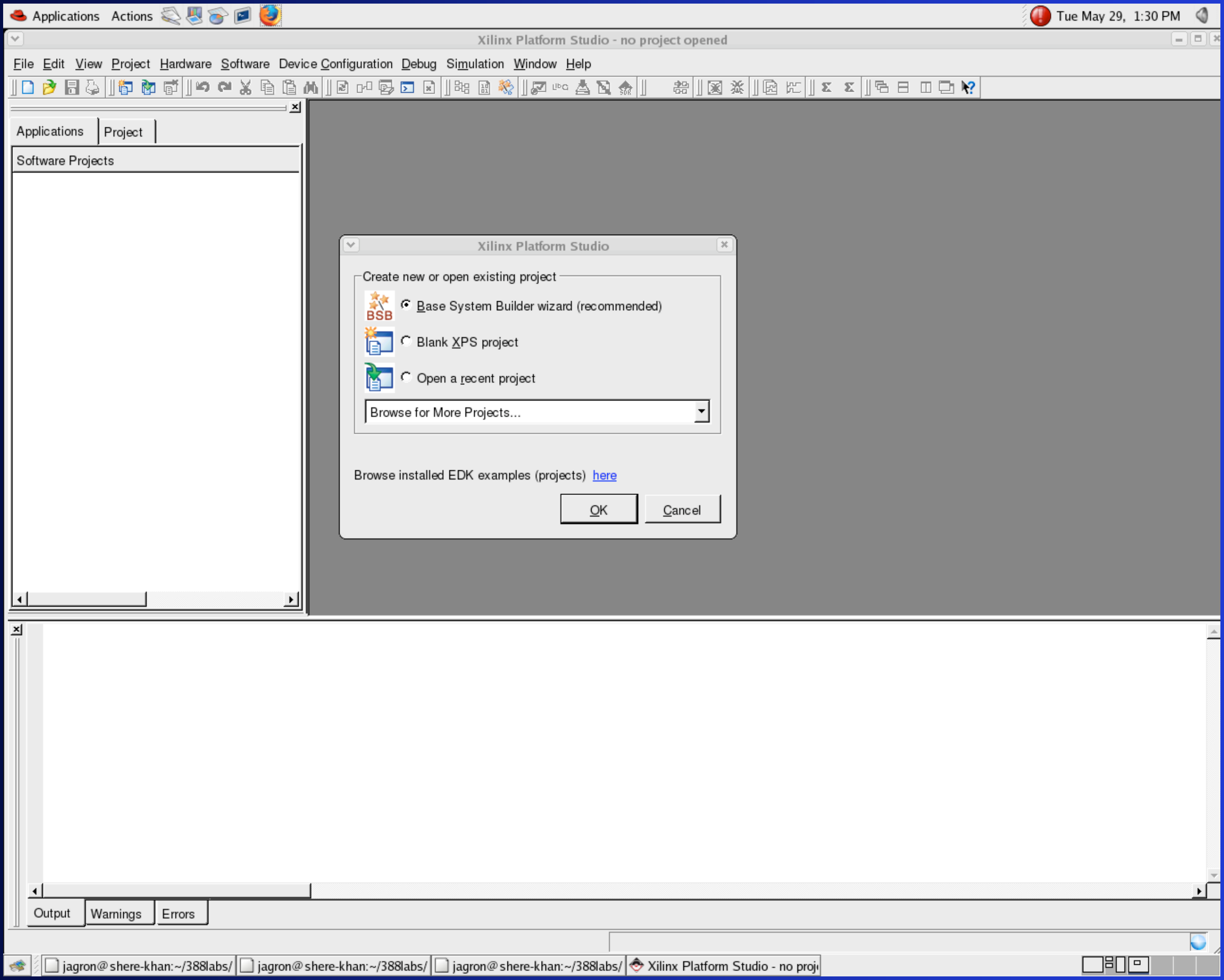
- Open up XPS.
- Create a new project.
  - Select “File”, “New Project”
  - Select “Base System Builder…”
    - Provides a wizard to help get basic system established.
  - Click OK.

XPS - Getting Started





XPS - New Project Creation


Applications Actions
Tue May 29, 1:30 PM
Xilinx Platform Studio - no project opened
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Applications
Project
Software Projects
Output
Warnings
Errors
jagron@shere-khan:~/388labs/
jagron@shere-khan:~/388labs/
jagron@shere-khan:~/388labs/
Xilinx Platform Studio - no proj

Xilinx Platform Studio

Create new or open existing project

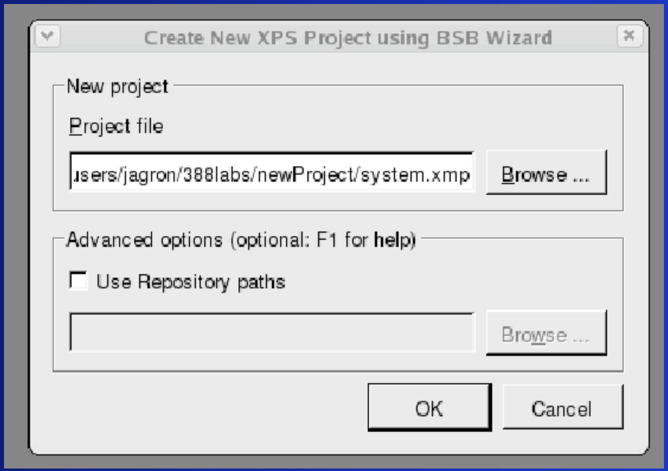
- (●) Base System Builder wizard (recommended)
- ( ) Blank XPS project
- ( ) Open a recent project

Browse for More Projects...

Browse installed EDK examples (projects) here

OK Cancel

XPS - Creating The Base System


Create New XPS Project using BSB Wizard
New project
Project file
Browse ...
Advanced options (optional: F1 for help)
Use Repository paths
Browse ...
OK
Cancel

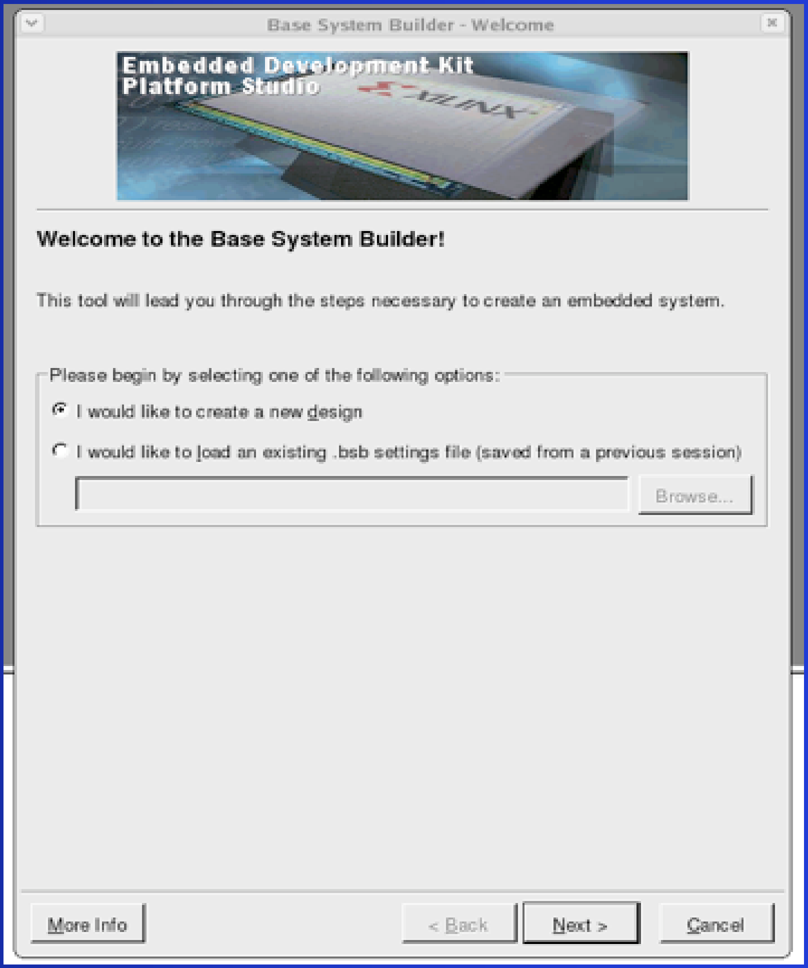
Project file: ısers/jagron/388labs/newProject/system.xmp

☐ Use Repository paths

- Now, create a directory for this EDK project.
  - Saved it as "system.xmp".
- IMPORTANT NOTE!!!!
  - Make sure that the absolute path contains no spaces!!!!
  - Make sure that this project is in it's own directory!!!

XPS - Base System Builder

- The Base System Builder window will open.
- Select "Create a New Design…".
- Now we can select the base components of our custom SoC.



Welcome to the Base System Builder!

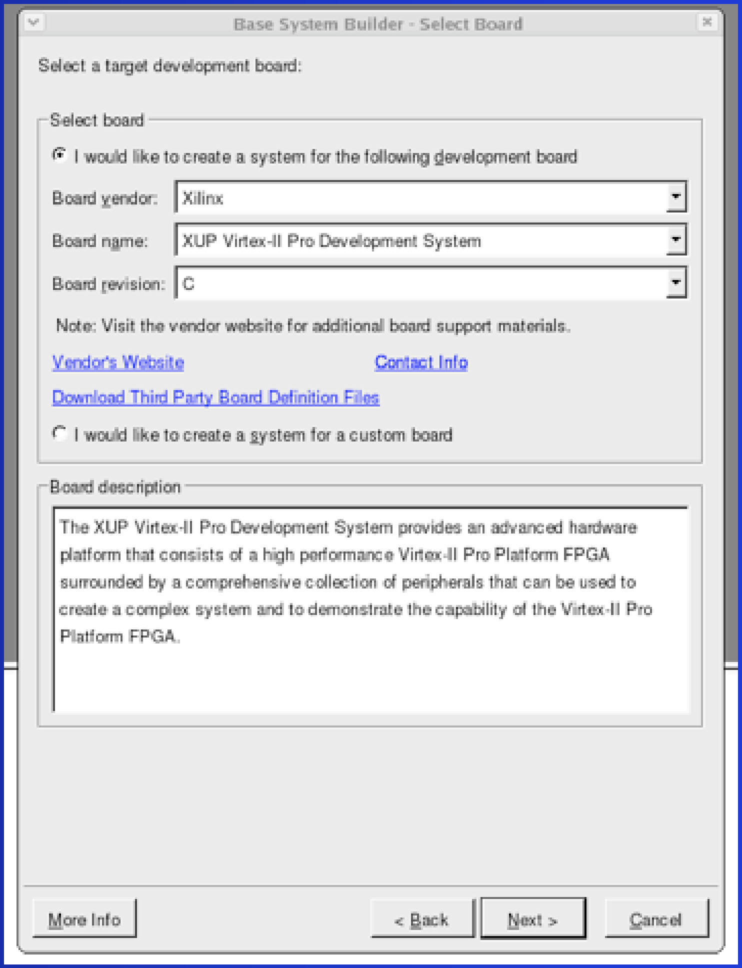
This tool will lead you through the steps necessary to create an embedded system.

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)

XPS - Board Selection

- We must select the platform we wish to use.
- In our case it is…
  - Vendor = Xilinx.
  - Board = XUP.
  - Rev # = C


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: XUP Virtex-II Pro Development System

Board revision: C

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

Vendor's Website   Contact Info

Download Third Party Board Definition Files

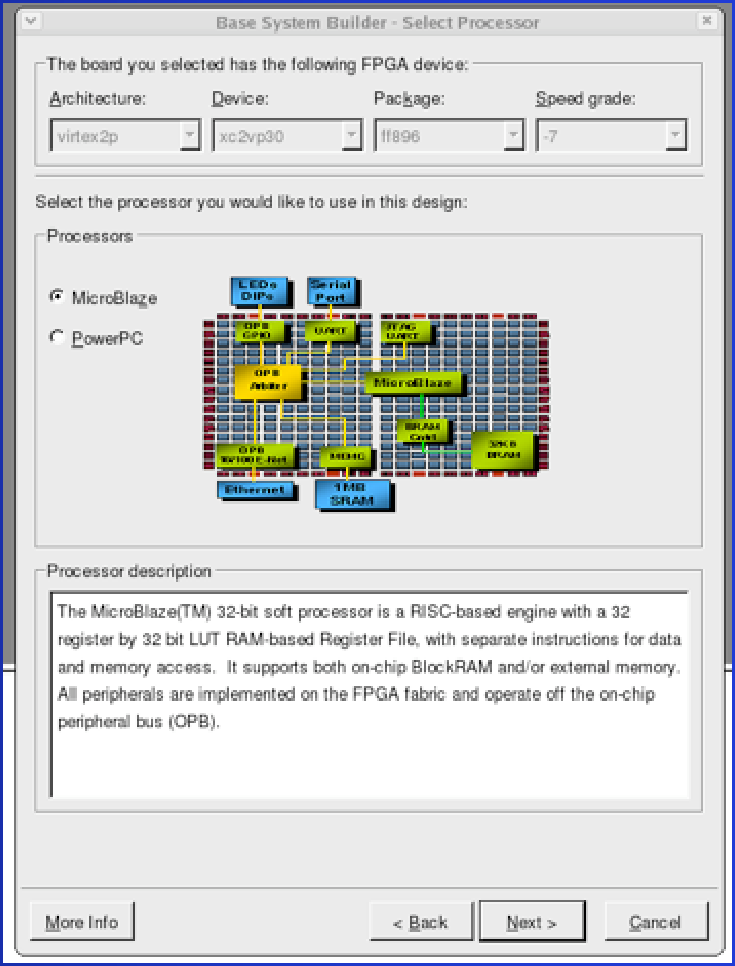
( ) I would like to create a system for a custom board

Board description

The XUP Virtex-II Pro Development System provides an advanced hardware platform that consists of a high performance Virtex-II Pro Platform FPGA surrounded by a comprehensive collection of peripherals that can be used to create a complex system and to demonstrate the capability of the Virtex-II Pro Platform FPGA.

XPS - Processor Selection

- Choice as to which processor to use in our SoC.
- PowerPC:
  - PPC405 Hard Core.
  - Physical CPU embedded within FPGA fabric.
- MicroBlaze:
  - Soft core.
  - Must be synthesized (implemented using the FPGA fabric).
- We will use the MicroBlaze.



The board you selected has the following FPGA device:

Architecture:	Device:	Package:	Speed grade:
virtex2p	xc2vp30	ff896	-7

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

Processors

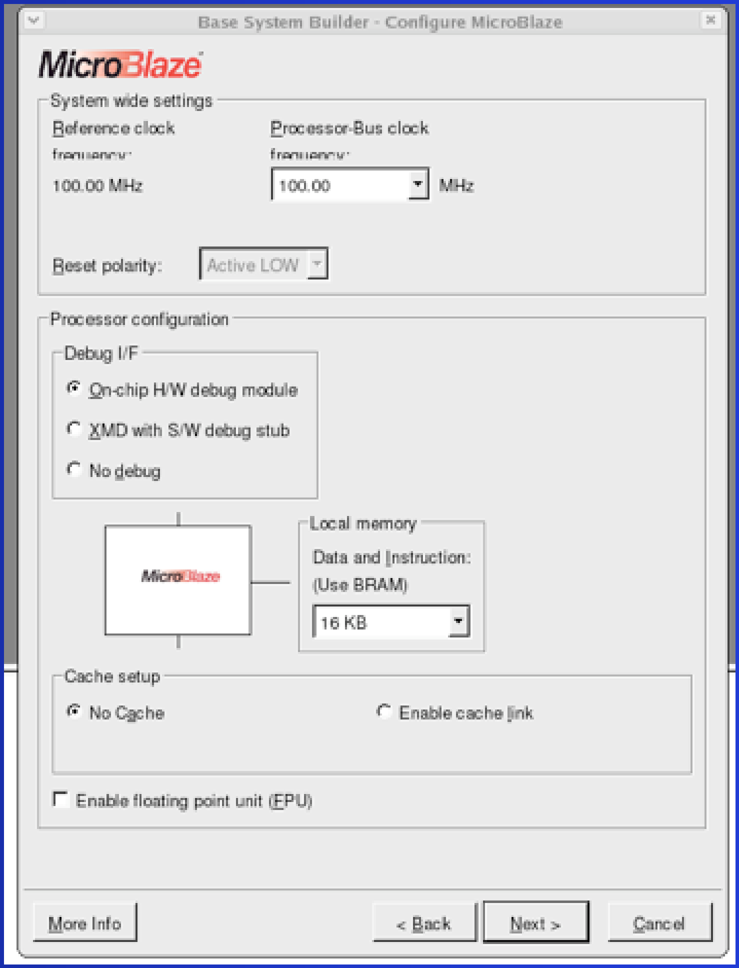
(● MicroBlaze) ○ PowerPC

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).

XPS - Processor Configuration

- Choose customizable CPU settings.
- In our case…
  - Bus freq. = 100 MHz.
  - Debug I/F = On-chip.
  - Local mem. = 16 KB.
  - No cache.
  - Disabled FPU.
- Simple, but highly effective.



Base System Builder - Configure MicroBlaze

MicroBlaze

System wide settings

Reference clock frequency: 100.00 MHz

Processor-Bus clock frequency: 100.00 MHz

Reset polarity: Active LOW

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) 16 KB

Cache setup

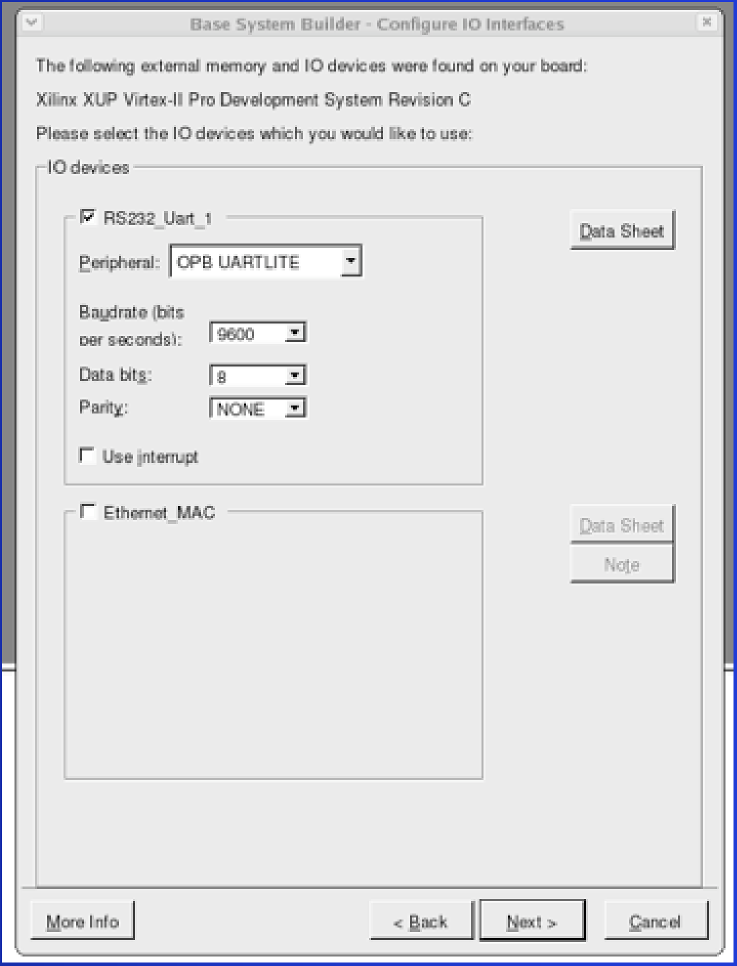
[✓ No Cache]

Enable cache link

Enable floating point unit (FPU)

XPS - I/O Interface Configuration

- Choose from available I/O interfaces.
- Ethernet.
  - allows boards to be networked.
- RS232 UART.
  - Serial protocol I/O for the board.
- In our case…
  - No Ethernet.
  - RS232 + UARTLite.
    - 9600 Baud. No parity
    - 8 Data bits.


Base System Builder - Configure IO Interfaces

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

Xilinx XUP Virtex-II Pro Development System Revision C

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

IO devices

☑ RS232_Uart_1

Peripheral: OPB UARTLITE

Baudrate (bits per seconds): 9600

Data bits: 8

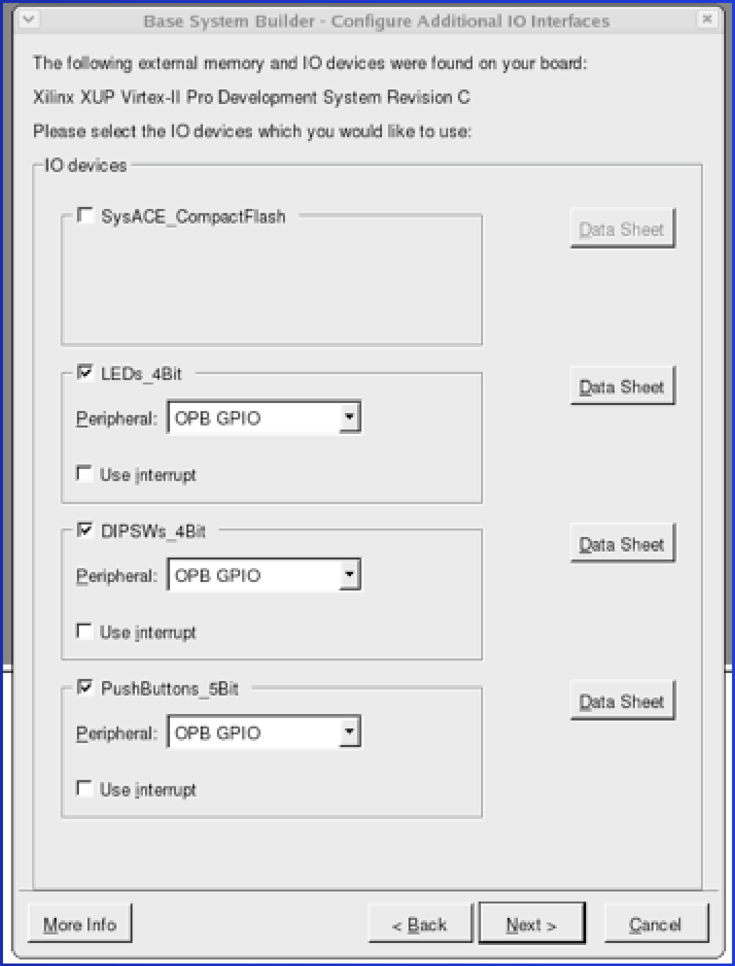
Parity: NONE

☐ Use interrupt

☐ Ethernet_MAC

XPS - I/O Interface Configuration

- Configure additional I/O interfaces.
- SysACE.
  - Allows for file storage.
- General Purpose I/O.
  - GPIO.
  - Used for LEDs, switches, and push buttons.
- In our case…
  - Only use the GPIOs for LEDs, DIPSWs, and PushButtons (No SysACE).


Base System Builder - Configure Additional IO Interfaces

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

Xilinx XUP Virtex-II Pro Development System Revision C

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

IO devices

☐ SysACE_CompactFlash

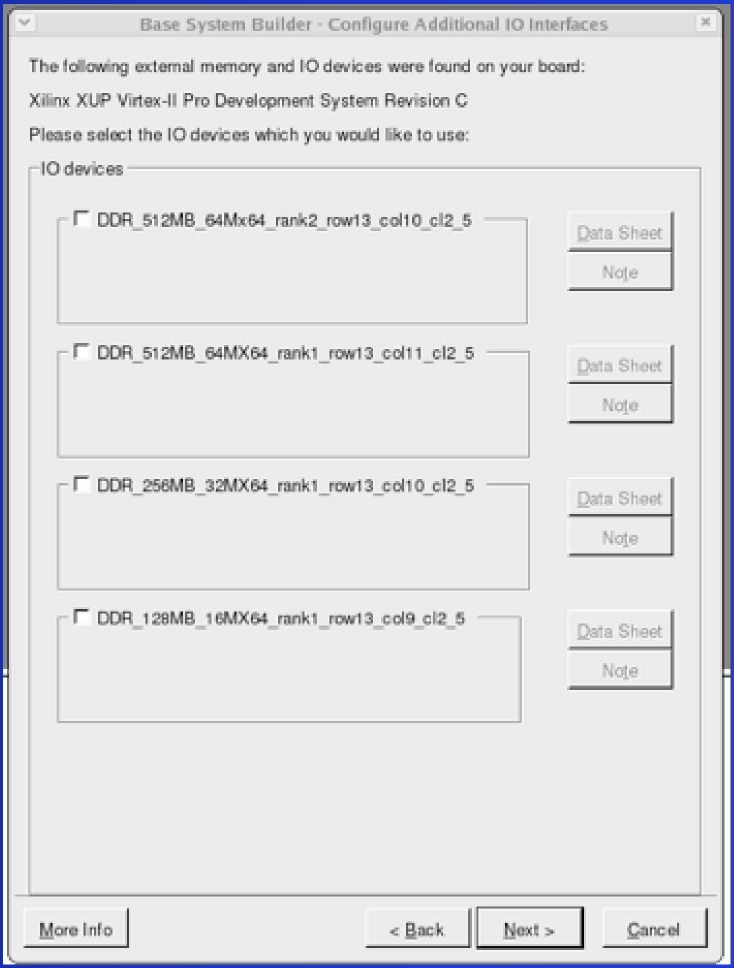
☑ LEDs_4Bit
Peripheral: OPB GPIO
☐ Use interrupt

☑ DIPSWs_4Bit
Peripheral: OPB GPIO
☐ Use interrupt

☑ PushButtons_5Bit
Peripheral: OPB GPIO
☐ Use interrupt

XPS - I/O Interface Configuration

- Configure additional memory interfaces.
- Different types of external memory.
- Often times DDR.
  - Large amount of storage.
  - Cheap.
  - Fast.
- For this system…
  - No external memory.


Base System Builder - Configure Additional IO Interfaces

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

Xilinx XUP Virtex-II Pro Development System Revision C

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

IO devices

☐ DDR_512MB_64Mx64_rank2_row13_col10_cl2_5

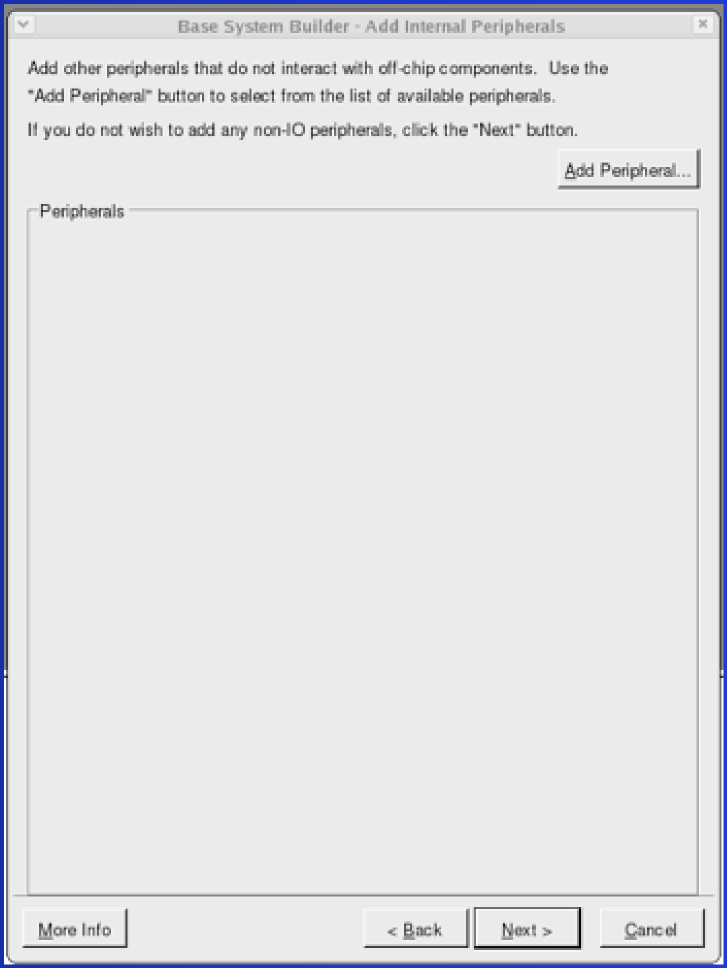
☐ DDR_512MB_64MX64_rank1_row13_col11_cl2_5

☐ DDR_256MB_32MX64_rank1_row13_col10_cl2_5

☐ DDR_128MB_16MX64_rank1_row13_col9_cl2_5

XPS - Add Internal Peripherals

- Xilinx provides a large library of peripherals:
  - I/O
  - Debug
  - Busses
  - Memory
  - Timers
  - Interrupts
  - A/D
- In our case…
  - We will add internal peripherals at a later time.

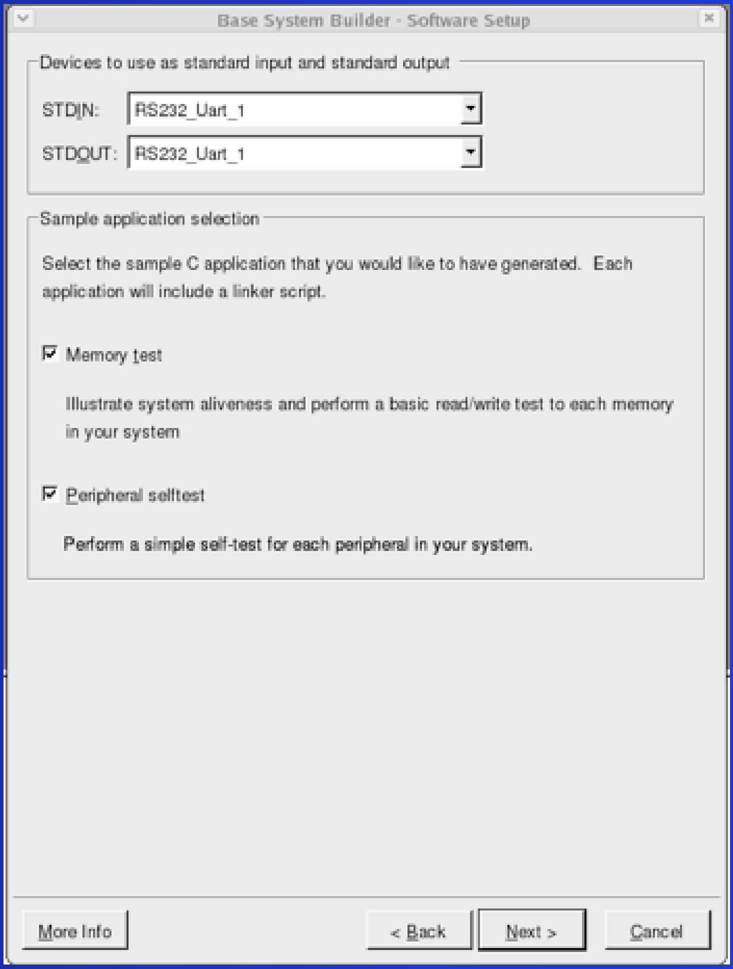


Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.

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

XPS - Software Setup

- Configuration of software-related properties of the system.
- I/O Configuration:
  - STDIN = UART.
  - STDOUT = UART.
- Include sample applications:
  - Memory test.
  - Peripheral self-test.
- In our case…
  - Use the defaults.


Base System Builder - Software Setup

Devices to use as standard input and standard output

STDIN: RS232_Uart_1

STDOUT: RS232_Uart_1

Sample application selection

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

☑ Memory test

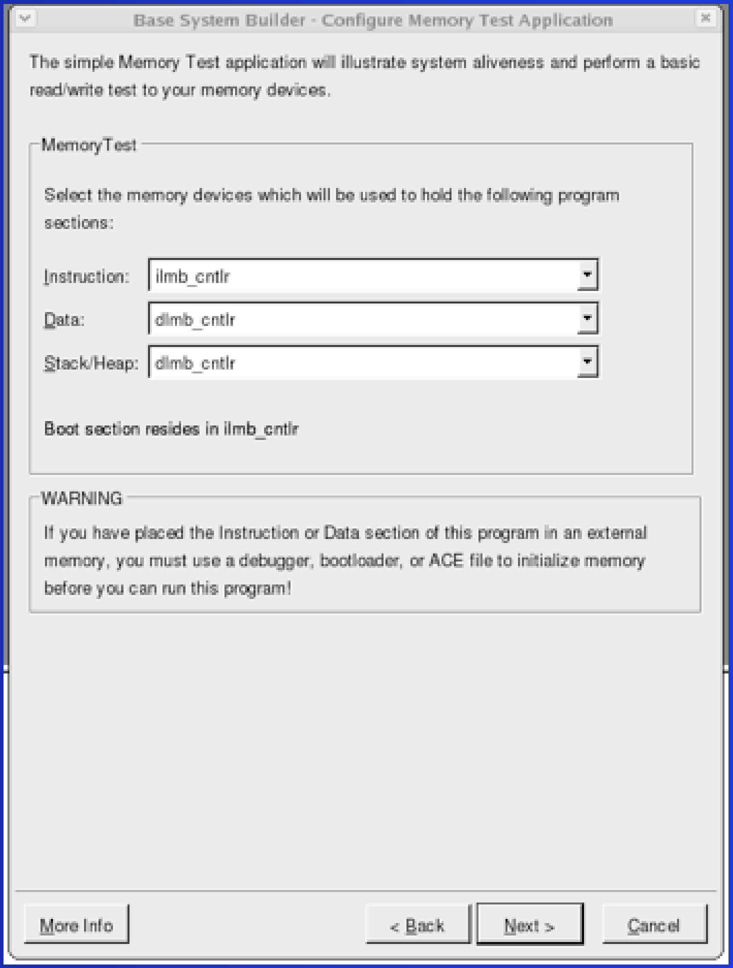
Illustrate system aliveness and perform a basic read/write test to each memory in your system

☑ Peripheral selftest

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

XPS - Application Configuration

- Configuration of application-related properties of the system.
- Choose where instructions and data are stored in the system.
- In our case…
  - Our system is simple a single memory for instructions and a single memory for data.
  - Use the defaults.



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

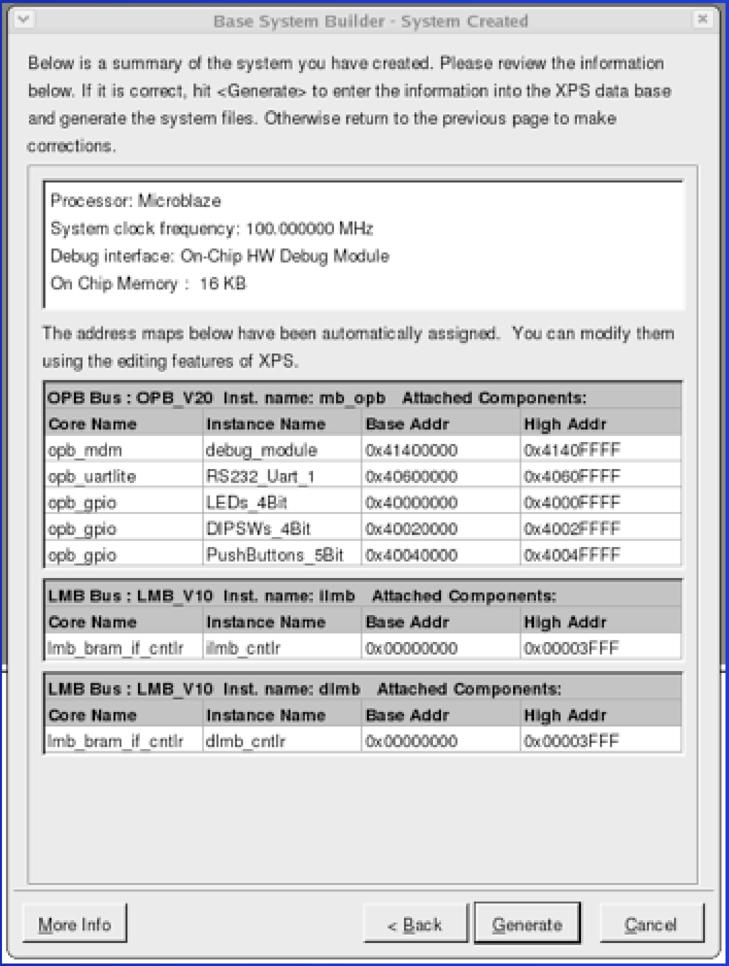
Boot section resides in ilmb_cntlr

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!

XPS - System Created

- System configuration is complete.
- Displays all of the included components and their associated address spaces.
- What are *address spaces*?
- Why is knowing the address space of a device useful?
- 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: 100.000000 MHz
Debug interface: On-Chip HW Debug Module
On Chip Memory : 16 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	0x40000000	0x4000FFFF
opb_gpio	DIPSWs_4Bit	0x40020000	0x4002FFFF
opb_gpio	PushButtons_5Bit	0x40040000	0x4004FFFF

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

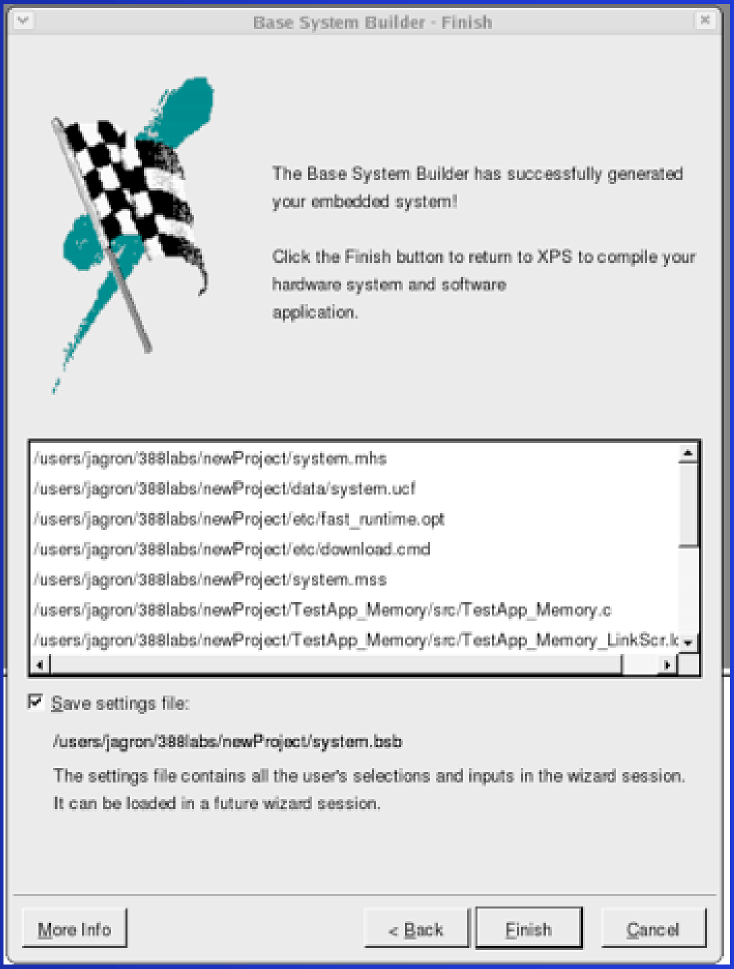
Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00003FFF

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

XPS - BSB Complete

- Congratulations!!!!
  - You have just created a custom SoC!
- Now click on “Finish”, and you can begin…
  - Using the system.
  - Developing custom HW and SW for the system.
- BSB has just generated a .mhs file for your system.
  - A file that lists all components and how they are configured and connected.
  - This file can be translated directly to VHDL or Verilog, and synthesized to the FPGA.



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.

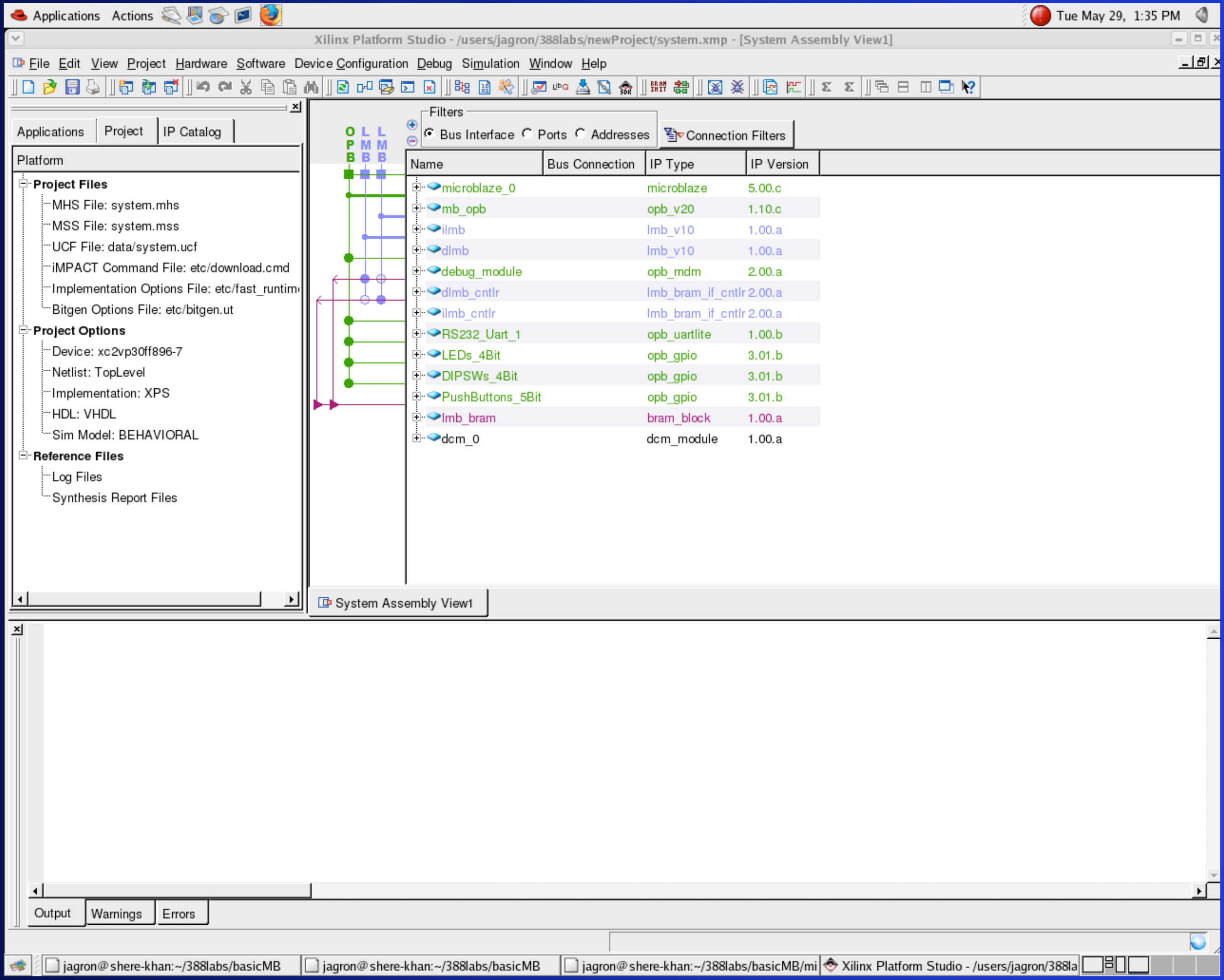
```
/users/jagron/388labs/newProject/system.mhs
/users/jagron/388labs/newProject/data/system.ucf
/users/jagron/388labs/newProject/etc/fast_runtime.opt
/users/jagron/388labs/newProject/etc/download.cmd
/users/jagron/388labs/newProject/system.mss
/users/jagron/388labs/newProject/TestApp_Memory/src/TestApp_Memory.c
/users/jagron/388labs/newProject/TestApp_Memory/src/TestApp_Memory_LinkScr.k
```

[✓] Save settings file:

/users/jagron/388labs/newProject/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.

XPS - Project Tab



Xilinx Platform Studio - /users/jagron/388labs/newProject/system.xmp - [System Assembly View1]

File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help

Applications | Project | IP Catalog

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_runtim
  - Bitgen Options File: etc/bitgen.ut
- Project Options
  - Device: xc2vp30ff896-7
  - Netlist: TopLevel
  - Implementation: XPS
  - HDL: VHDL
  - Sim Model: BEHAVIORAL
- Reference Files
  - Log Files
  - Synthesis Report Files

Filters: Bus Interface, Ports, Addresses — Connection Filters

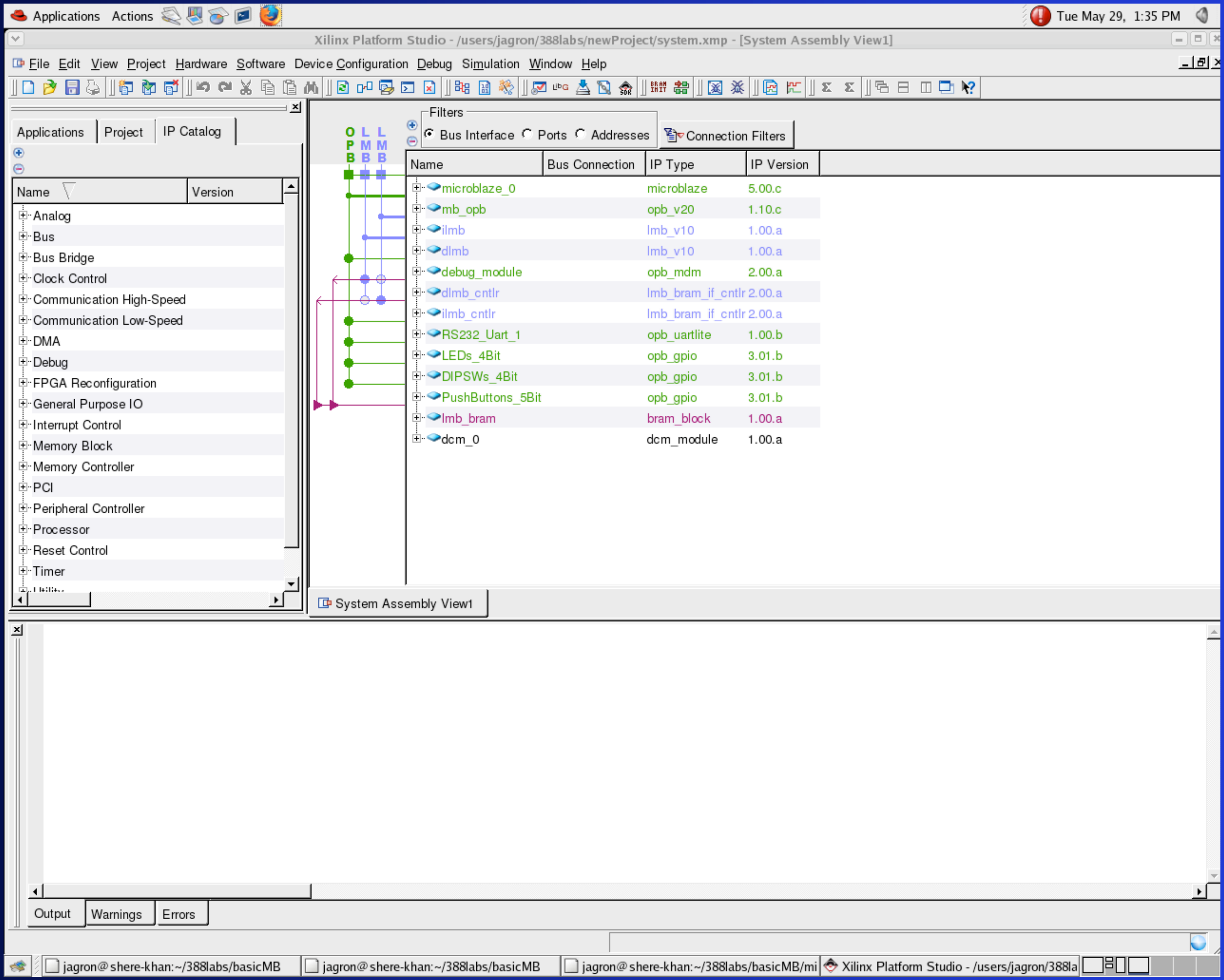
Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	5.00.c
mb_opb		opb_v20	1.10.c
ilmb		lmb_v10	1.00.a
dlmb		lmb_v10	1.00.a
debug_module		opb_mdm	2.00.a
dlmb_cntlr		lmb_bram_if_cntlr	2.00.a
ilmb_cntlr		lmb_bram_if_cntlr	2.00.a
RS232_Uart_1		opb_uartlite	1.00.b
LEDs_4Bit		opb_gpio	3.01.b
DIPSWs_4Bit		opb_gpio	3.01.b
PushButtons_5Bit		opb_gpio	3.01.b
lmb_bram		bram_block	1.00.a
dcm_0		dcm_module	1.00.a

System Assembly View1

Output | Warnings | Errors

Applications
Actions
Tue May 29, 1:35 PM
OPB
LMB
LMB
jagron@shere-khan:~/388labs/basicMB
jagron@shere-khan:~/388labs/basicMB
jagron@shere-khan:~/388labs/basicMB/mi
Xilinx Platform Studio - /users/jagron/388la

XPS - IP Catalog Tab

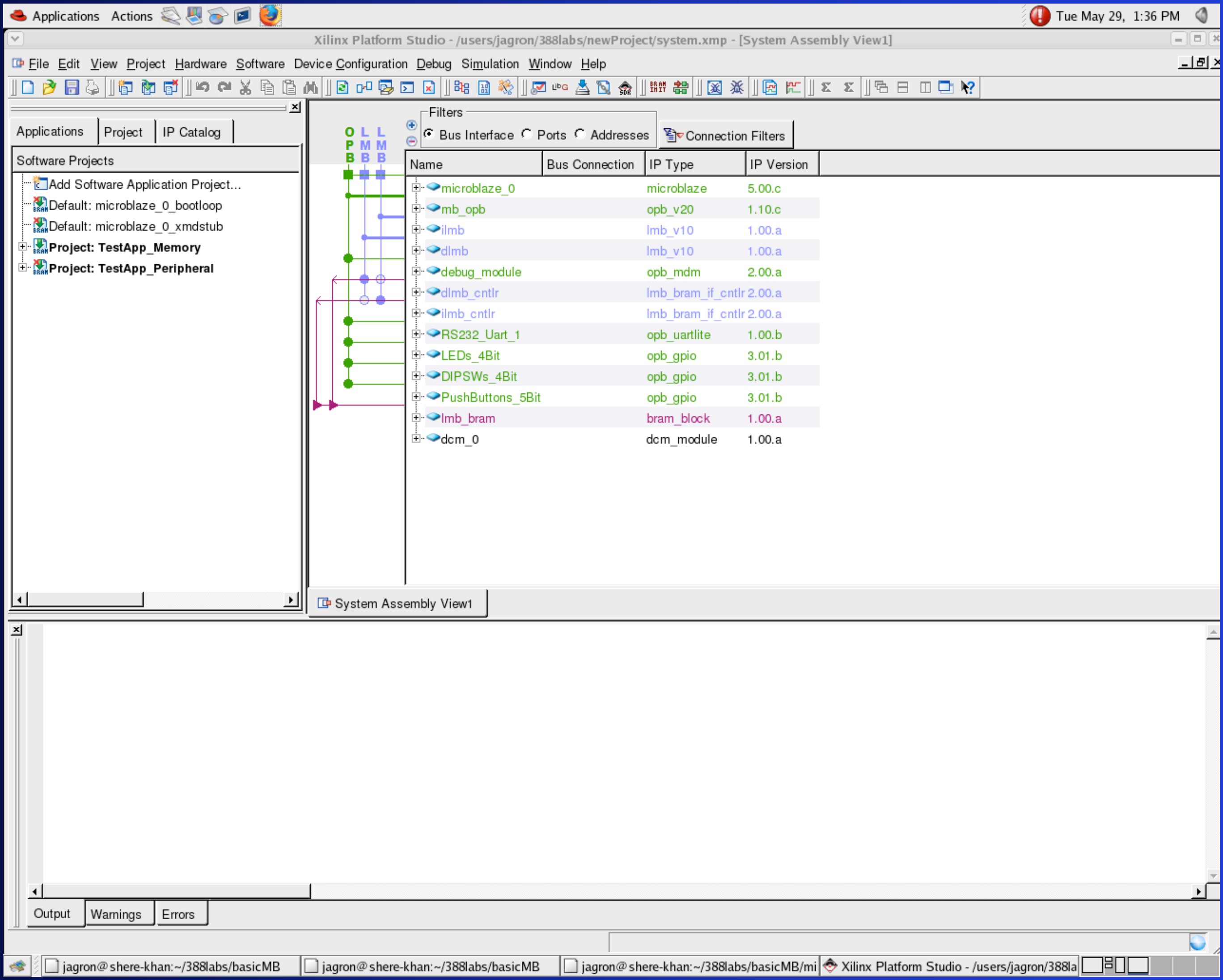

Applications Actions
Tue May 29, 1:35 PM
Xilinx Platform Studio - /users/jagron/388labs/newProject/system.xmp - [System Assembly View1]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Applications Project IP Catalog
Filters
Bus Interface Ports Addresses
Connection Filters
OPB LMB LMB
System Assembly View1
Output Warnings Errors
jagron@shere-khan:~/388labs/basicMB
jagron@shere-khan:~/388labs/basicMB
jagron@shere-khan:~/388labs/basicMB/mi
Xilinx Platform Studio - /users/jagron/388la

Filters: [✓ Bus Interface] Ports Addresses

Name	Version
Analog	
Bus	
Bus Bridge	
Clock Control	
Communication High-Speed	
Communication Low-Speed	
DMA	
Debug	
FPGA Reconfiguration	
General Purpose IO	
Interrupt Control	
Memory Block	
Memory Controller	
PCI	
Peripheral Controller	
Processor	
Reset Control	
Timer	

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	5.00.c
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
RS232_Uart_1		opb_uartlite	1.00.b
LEDs_4Bit		opb_gpio	3.01.b
DIPSWs_4Bit		opb_gpio	3.01.b
PushButtons_5Bit		opb_gpio	3.01.b
lmb_bram		bram_block	1.00.a
dcm_0		dcm_module	1.00.a

XPS - Applications Tab



Xilinx Platform Studio - /users/jagron/388labs/newProject/system.xmp - [System Assembly View1]

Software Projects
- Add Software Application Project...
- Default: microblaze_0_bootloop
- Default: microblaze_0_xmdstub
- **Project: TestApp_Memory**
- **Project: TestApp_Peripheral**

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	5.00.c
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
RS232_Uart_1		opb_uartlite	1.00.b
LEDs_4Bit		opb_gpio	3.01.b
DIPSWs_4Bit		opb_gpio	3.01.b
PushButtons_5Bit		opb_gpio	3.01.b
lmb_bram		bram_block	1.00.a
dcm_0		dcm_module	1.00.a

XPS Interface

- System Assembly View:
  - Graphical view of system.
  - Can edit configurations, port connections, bus connections, and memory spaces for all components.
- Tabs:
  - Project Tab.
    - Project info (.mhs, logs, etc.).
  - IP Catalog Tab.
    - Available peripherals that can be added to the system.
  - Application Tab.
    - Available SW projects that can be run on the system.

How To Run An Application

- If multiple applications are available, then one must be selected.
- This is done on the “Applications” Tab by…
  - Right-clicking on the application of choice and selecting “Mark to initialize BRAMs”.
- IMPORTANT:
  - Select the program you would like to run (now it will have a green BRAM symbol next to it.
  - Now you must always DE-SELECT the other applications by clicking “Mark to initialize BRAMs” (now it will have a red X on it’s BRAM symbol).

How To Run An Application

- Select the application of choice.
  - Compile the sources for the application.
    - Right-click and select “Build Application”.
- Execute the test on the base system platform.
  - This requires the following to be combined…
    - Hardware bitstream (.bit)
    - Software executable (.elf)
  - This is done by selecting “Device Configuration”.
    - “Update Bitstream” - combines HW/SW (.bit + .elf).
    - “Download Bitstream” - downloads the configuration to the board.
      - ➢ Over the USB-based JTAG connection.

Build Process (Start To Finish)

- Build your HW system.
  - Done for us using the base system builder.
- Use a default SW project.
- Click on “download” bitstream.
  - This will go through the entire HW/SW build process.
  - Build HW
    - Synthesize, ngdbuild, map, par, and bitgen.
  - Build SW
    - Compile and link.
  - Then it will combine the SW executable and the HW bitstream and download it to the FPGA.

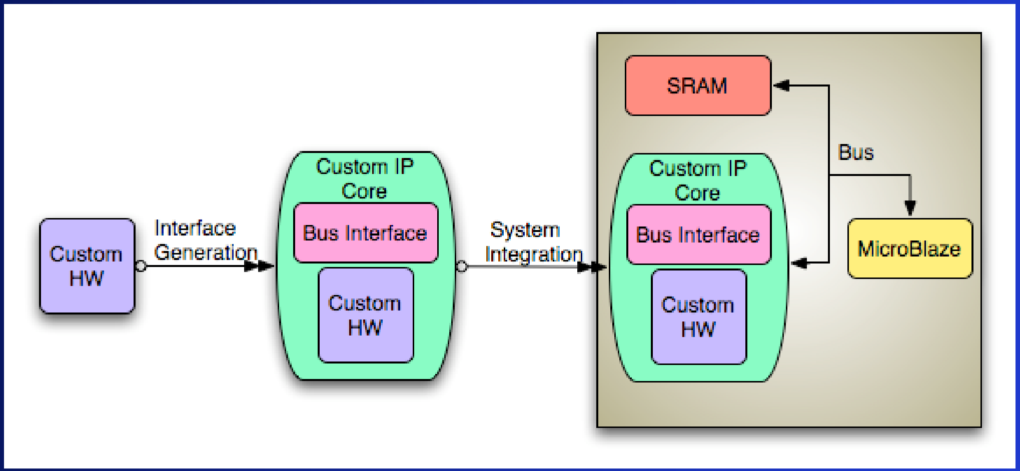
Monitoring Software Execution On The FPGA

- How do you see what is happening on the FPGA?
  - Normally in software you use print() statements.
  - The output goes to the screen.
- In this system, STDIN/STDOUT are routed to the serial port.
  - We must monitor the serial port from an external host to see what is happening.
- In order to "see" what is executing…
  - Open up a terminal window.
    - Minicom (Linux) or Hyperterminal (Windows).
  - Setup the correct communication parameters
    - Baud rate = 9600.

Creating New SW Applications

- Select “Software”…
  - Click on “Add Software Application Project”.
- Enter the new project name.
  - Also choose which CPU to run the application on.
- Now a new application tab entry will appear.
  - You can now add/edit sources for this application.
- In order to run this new application…
  - Right-click on it, Select “Mark to initialize BRAMs”.
    - Instructs the tool that this application is to be “combined” with the bitstream.
    - Make sure to de-select all other applications!!!
  - Now, when updating the bitstream, this application will be used.

Integration of IP Cores



- EDK allows one to add IP cores to a system.
  - Pre-built cores from Xilinx.
  - Custom cores.
- Additionally, there is a Create/Import Core Wizard that…
  - Allows one to quickly create bus interfaces for custom IP cores.
    - PLB, OPB, FSL interfaces.
  - Imports cores into an EDK repository.
    - So that the IP cores can be added into a system.

How to Create/Import Cores

- Select “Hardware”…
  - Click on “Create/Import Peripheral”.
- Creating a new peripheral…
  - Peripheral name.
  - Bus interface type (PLB, OPB, FSL).
  - Interface features:
    - Master/Slave.
    - Interrupts and Resets.
    - Registers.
    - Accessible Signals.
- Importing a peripheral…
  - Not recommended, much easier to do via cut/copy from the command line.

Initial Assignment

- Create a new SW project…
  - Make it do the following…
    - Print “Hello <yourName>!” 5 times.
- Run the SW on the board and demonstrate that it works correctly.
- This may not seem like much, but…
  - You just created a SoC (System-On-Chip).
  - You just cross-compiled a program to run on “bare-metal” (no OS).

HINTS

- To send output over the UART there are 3 types of “print” statements.
- print
  - Usage: print(“hello\r\n”);
  - Only takes a single string as an argument.
  - Lightweight print (small executable footprint).
- printf
  - Normal printf (large executable footprint).
- xil_printf
  - Lightweight printf (small executable footprint).

Questions

1) What is an FPGA?
2) What is an SoC? Why is it different from your desktop computer system?
3) What does soft-core IP mean?
4) What is an MHS, MSS, and UCF file?
5) What does cross-compile mean?
6) Why does it take so long to build the HW portion of your system?
7) How does the desktop computer program the FPGA, how does it monitor the FPGA?