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).

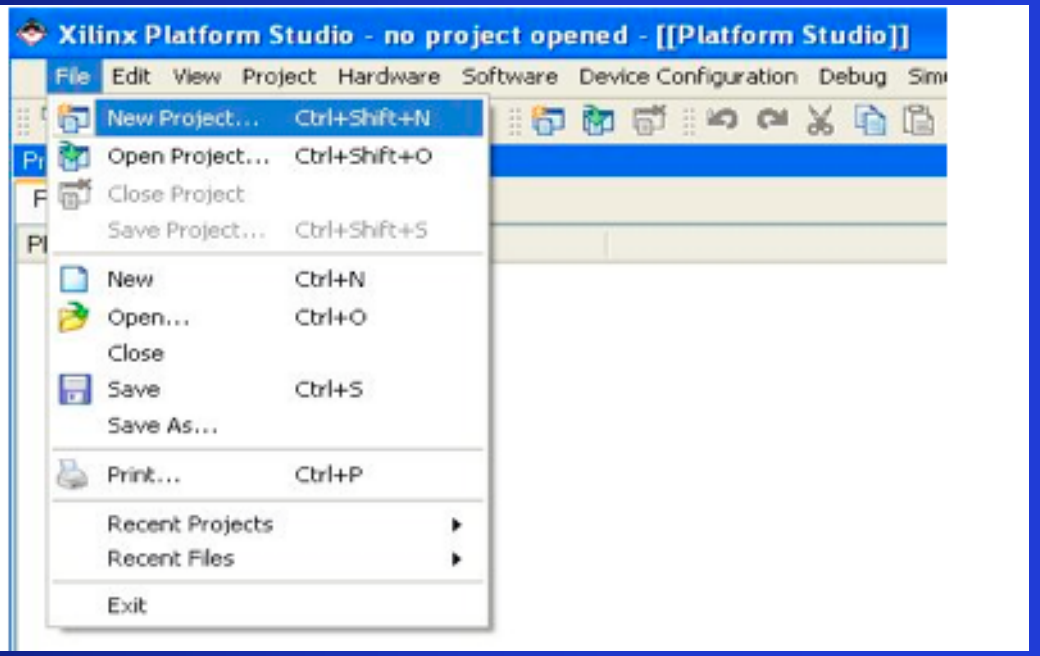
Important EDK Files

- 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.

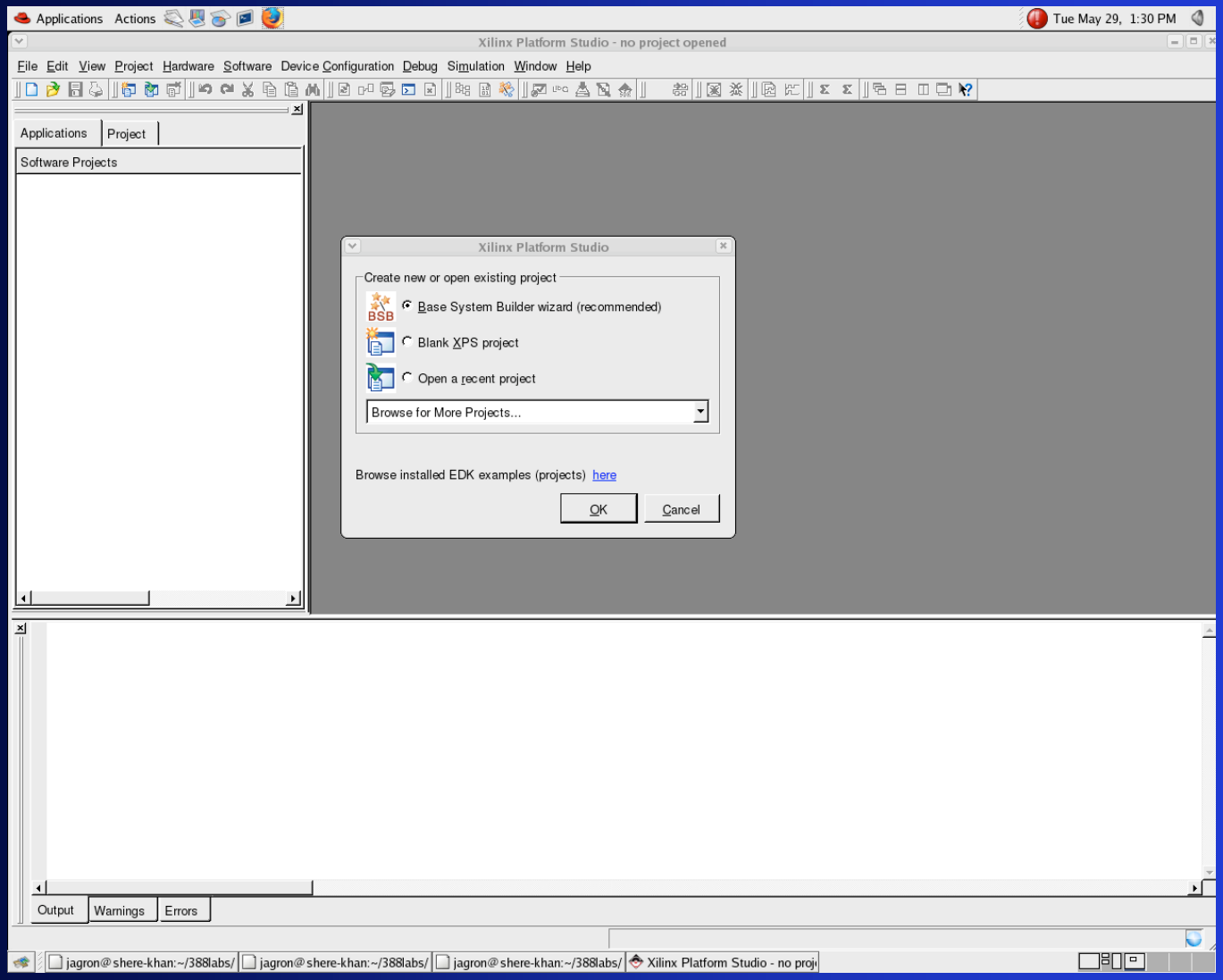
How To Get Started

- 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.

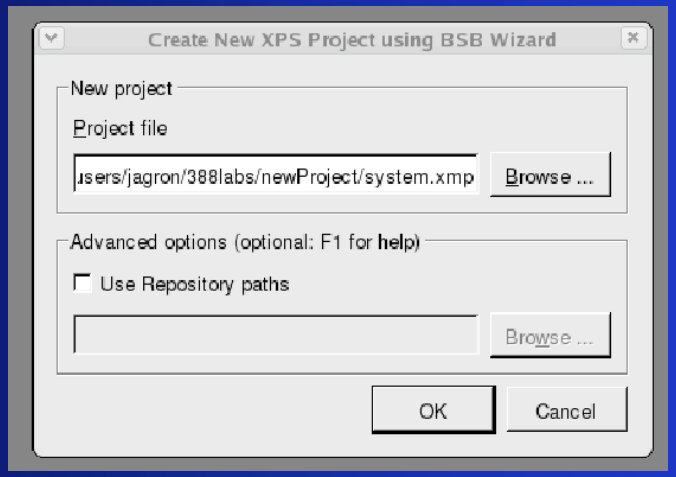
XPS - Getting Started



XPS - New Project Creation



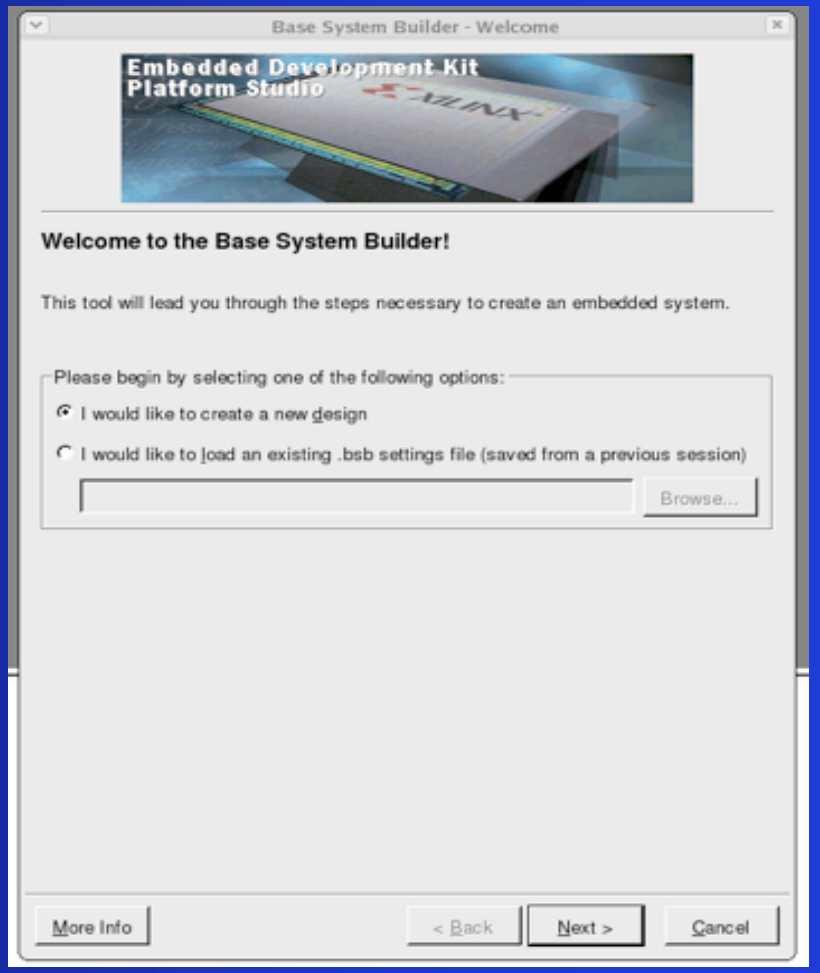
XPS - Creating The Base System



- Now, create a directory for this EDK project.
  - Saved as a .xmp file.
- IMPORTANT NOTE!!!!
  - Make sure that the absolute path contains no spaces!!!!

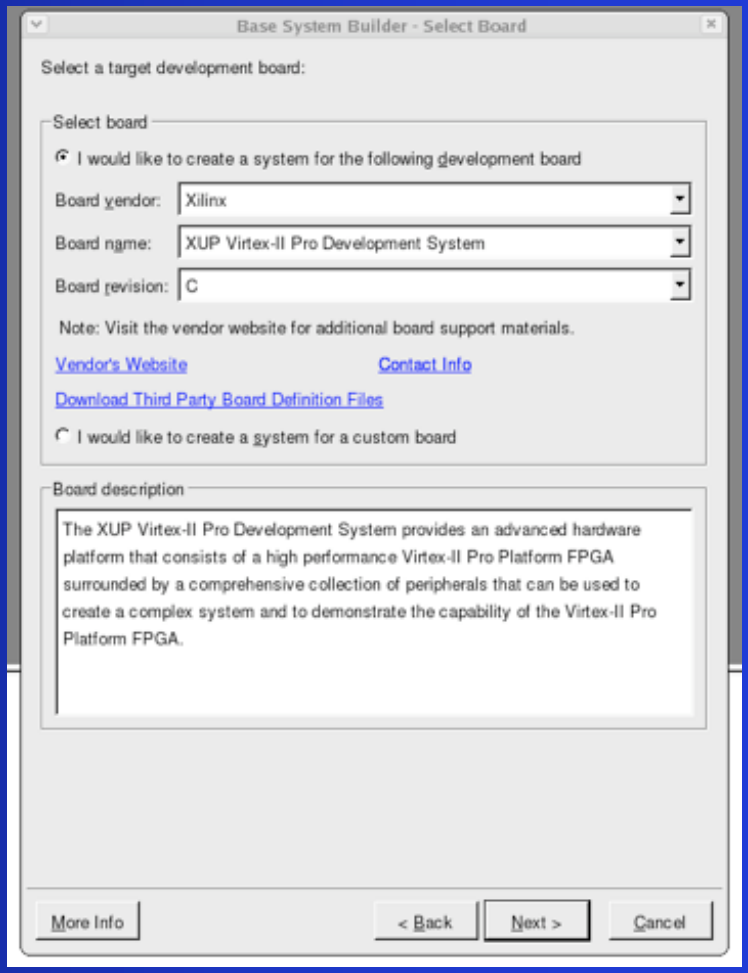
XPS - Base System Builder

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



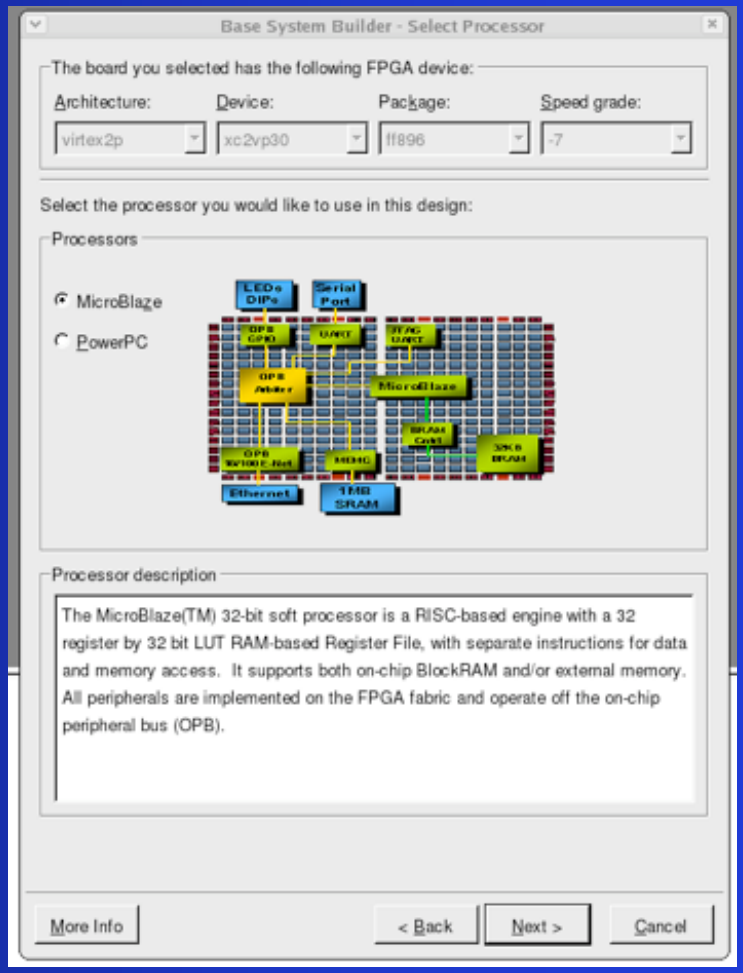
XPS - Board Selection

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



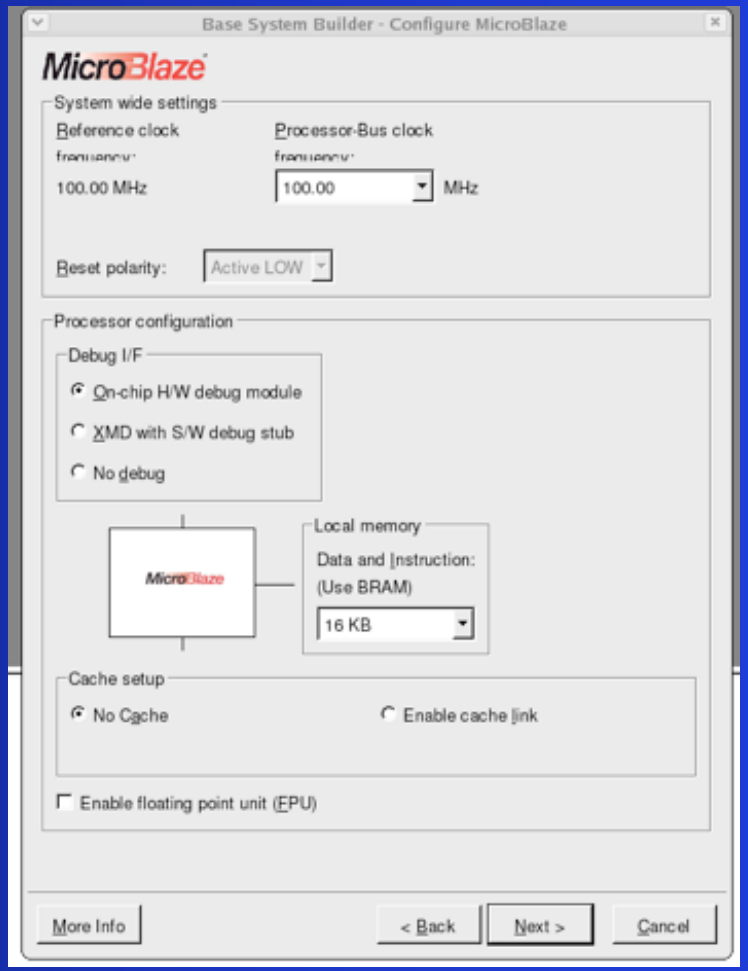
XPS - Processor Selection

- 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.



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.

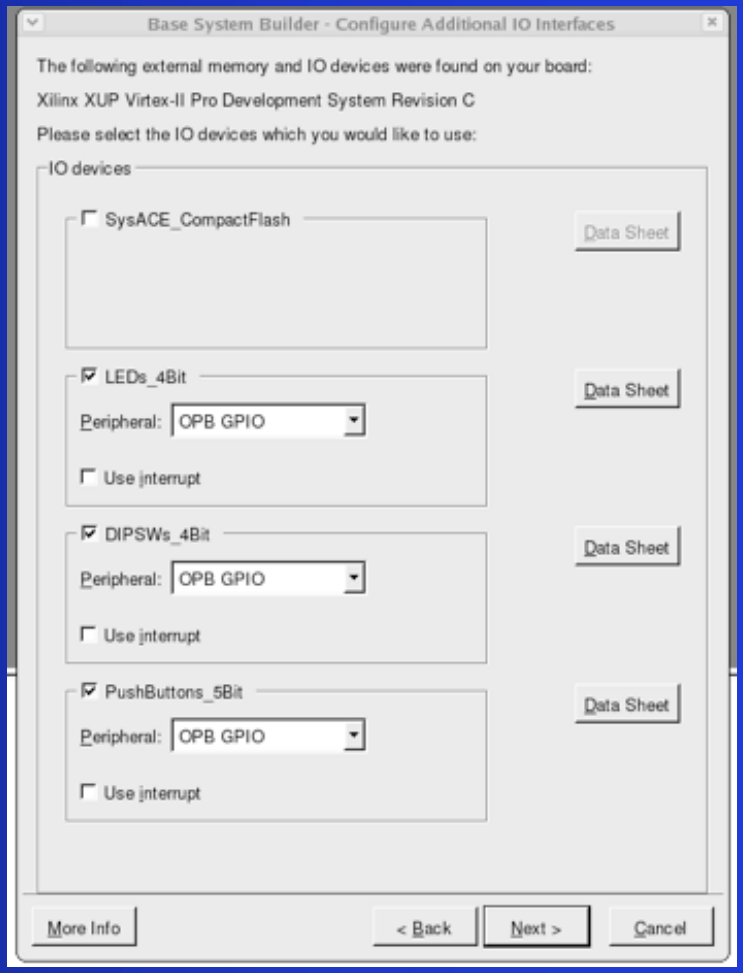


XPS - I/O Interface Configuration

- 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.

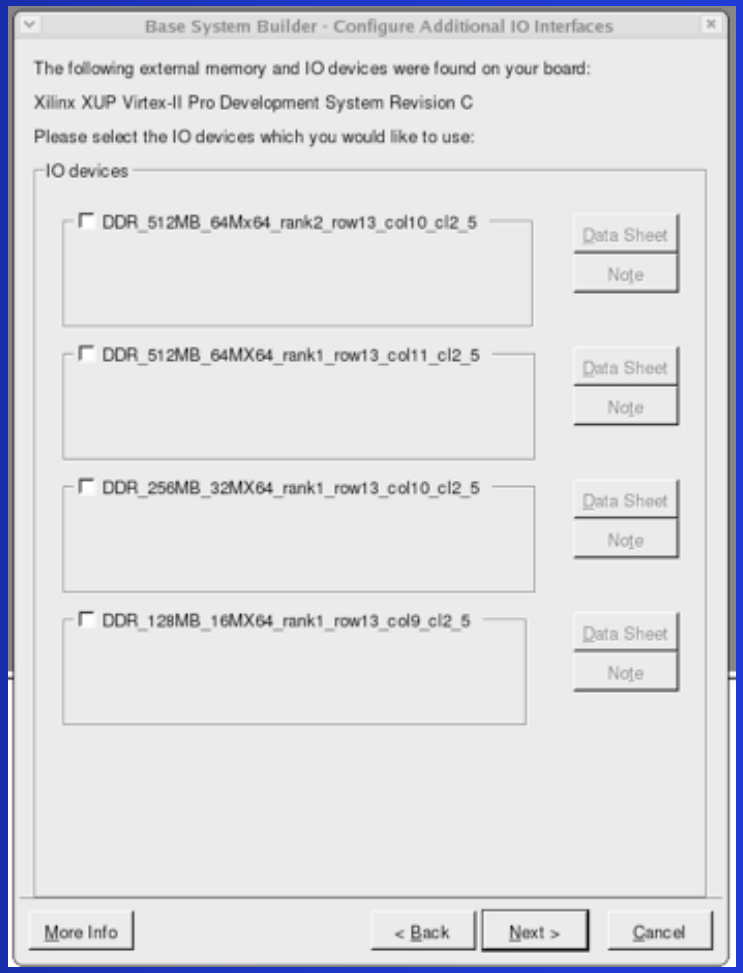
XPS - I/O Interface Configuration

- 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).



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.

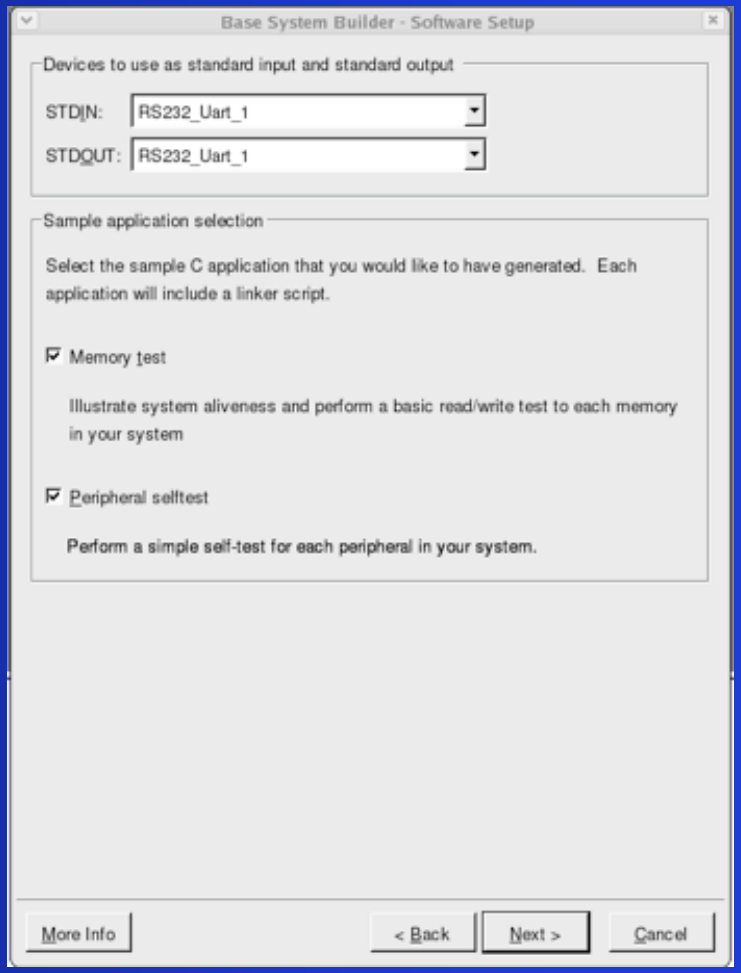


XPS - Add Internal Peripherals

- 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.

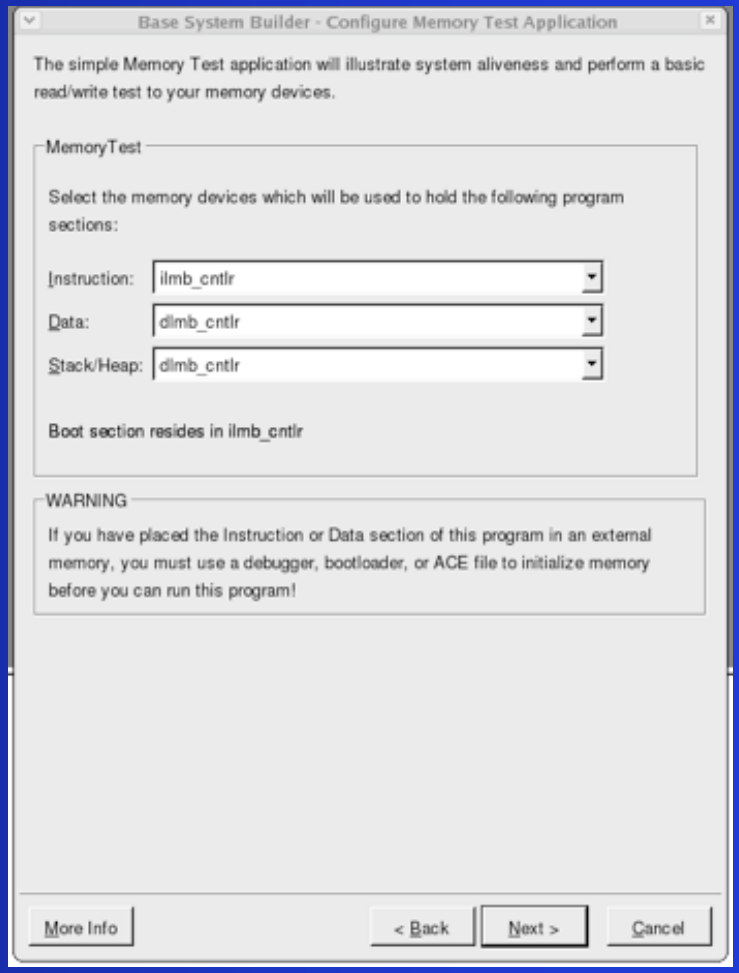
XPS - Software Setup

- 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.



XPS - Application Configuration

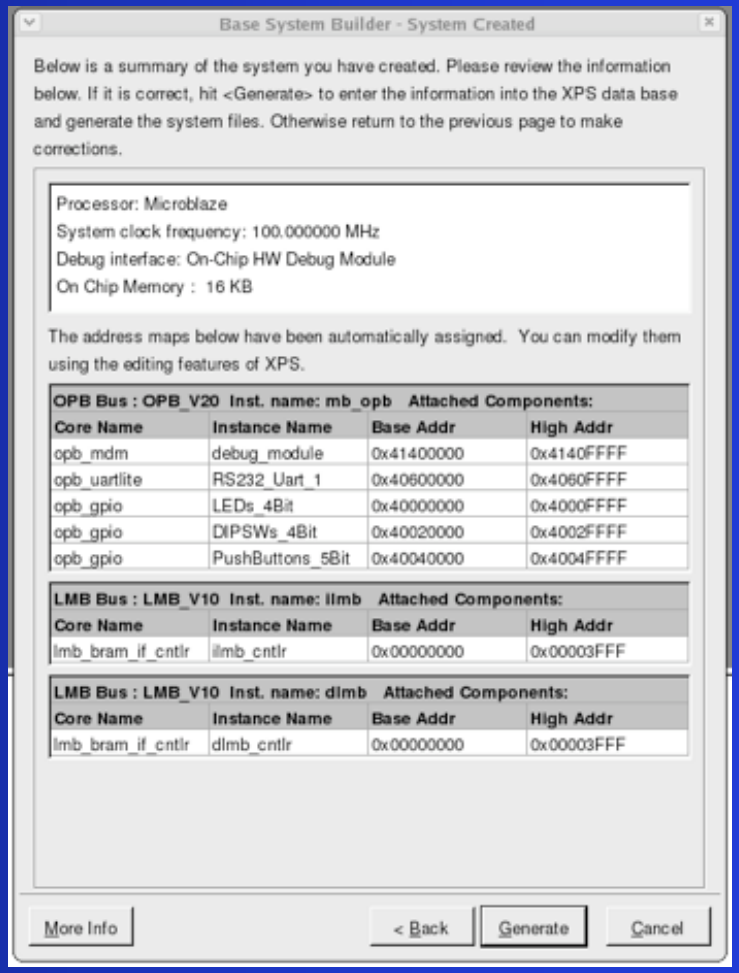
- Configuration of application-related properties of the system.
- Choose where instructions and data are stored in the system.
- In our case…
  - Our system is simple a single memory for instructions and a single memory for data.
  - Use the defaults.



XPS - System Created

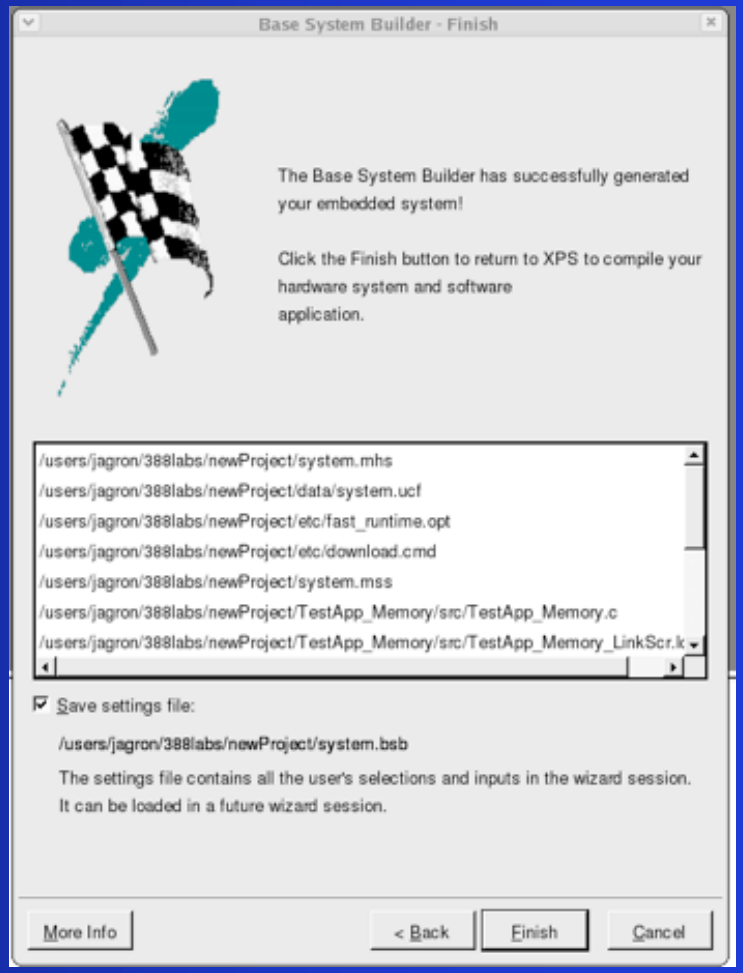
- System configuration is complete.
- Displays all of the included components and their associated address spaces.
- What are *address spaces*?
- Why is knowing the address space of a device useful?

- Click “Generate”…

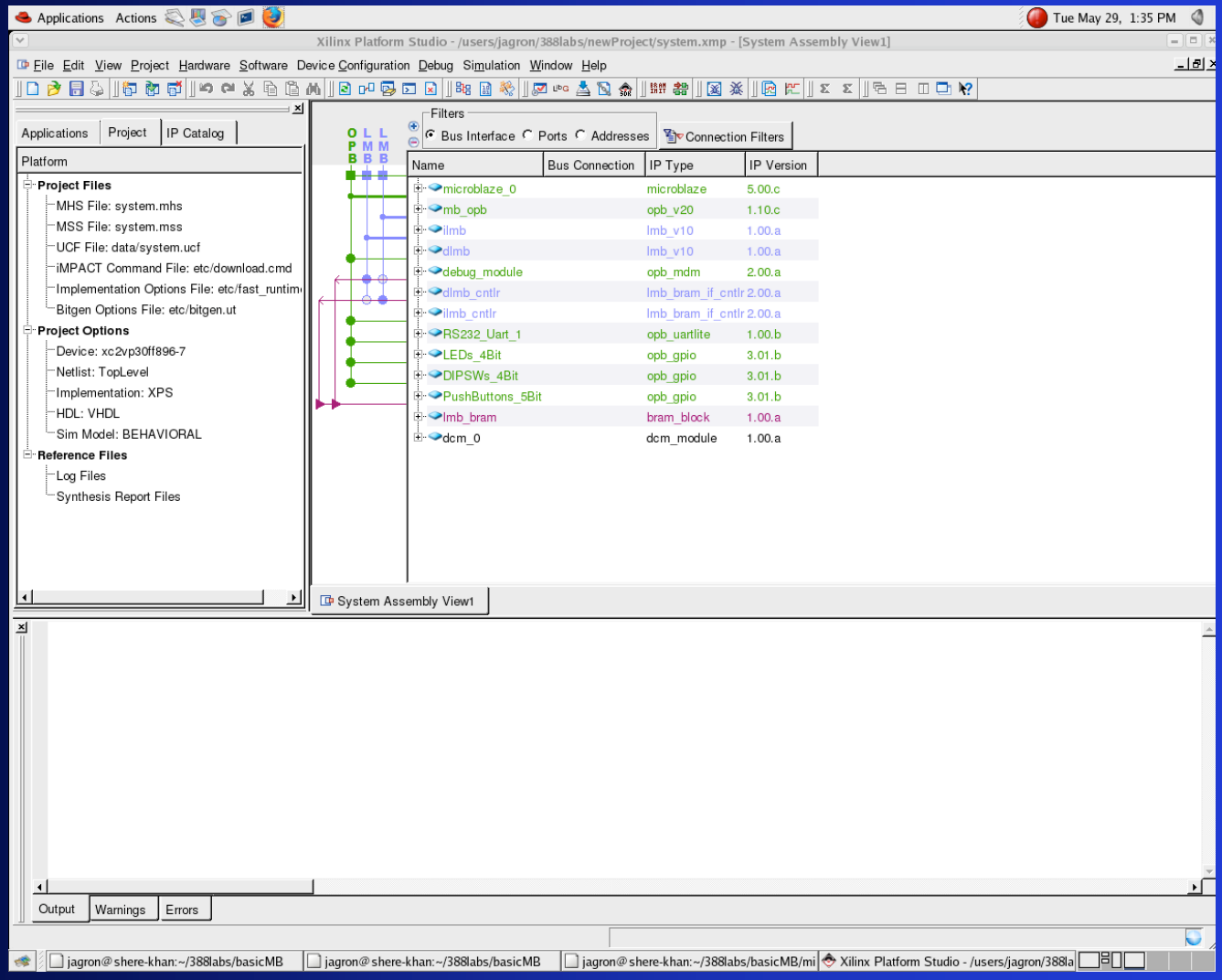


XPS - BSB Complete

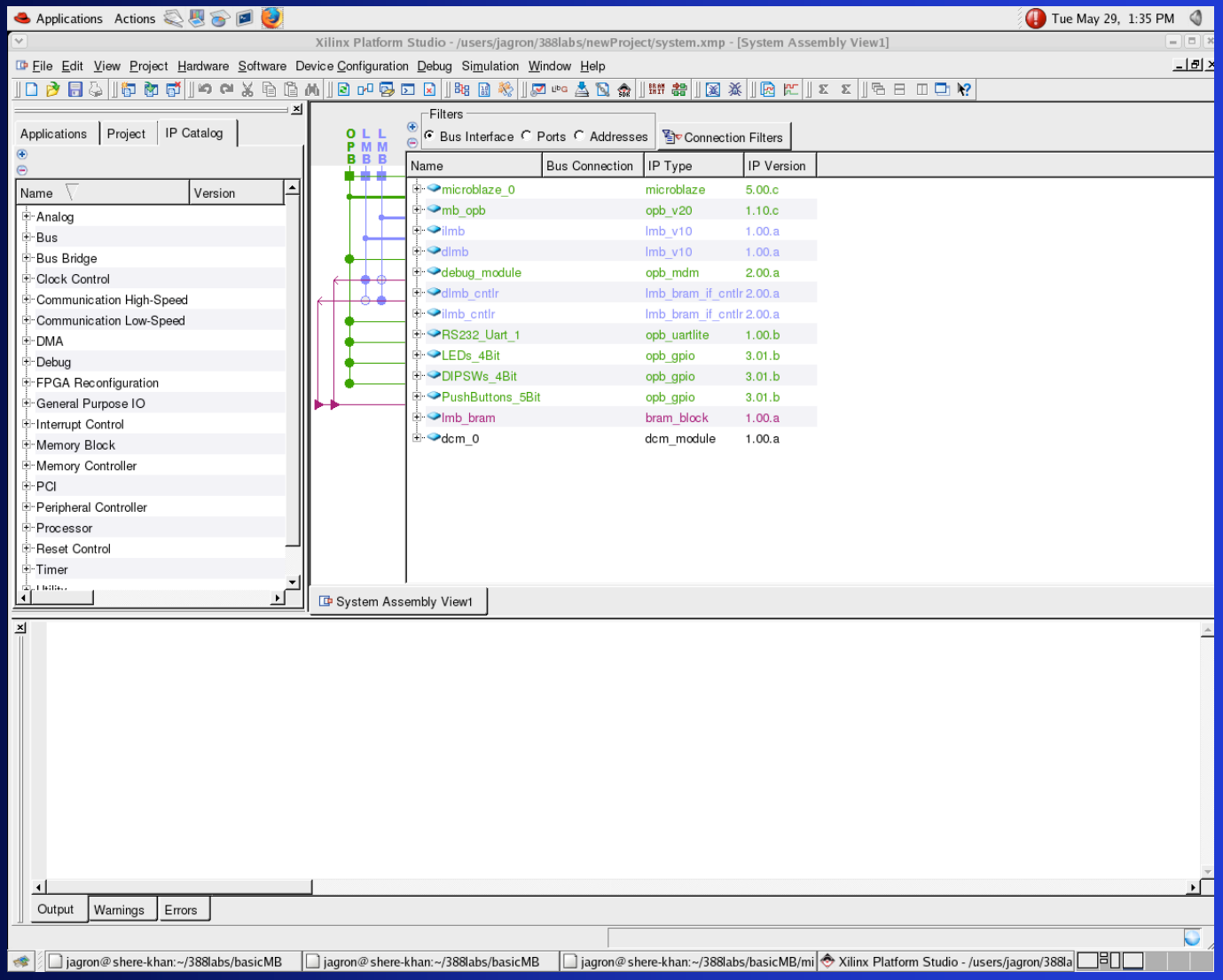
- Congratulations!!!!
  - You have just created a custom SoC!
- Now click on “Finish”, and you can begin…
  - Using the system.
  - Developing custom HW and SW for the system.
- BSB has just generated a .mhs file for your system.
  - A file that lists all components and how they are configured and connected.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



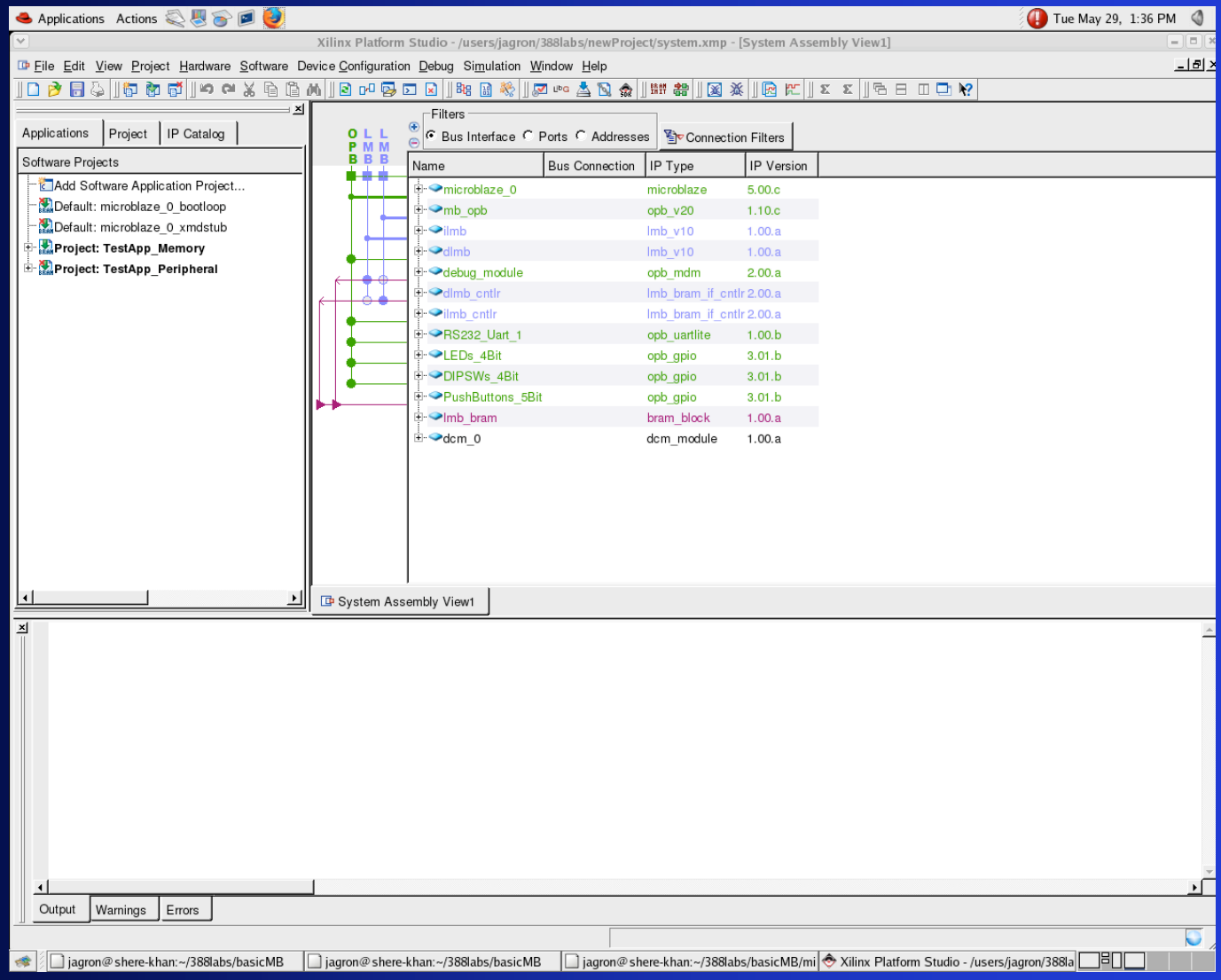
XPS - Project Tab



XPS - IP Catalog Tab



XPS - Applications Tab



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

- 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.

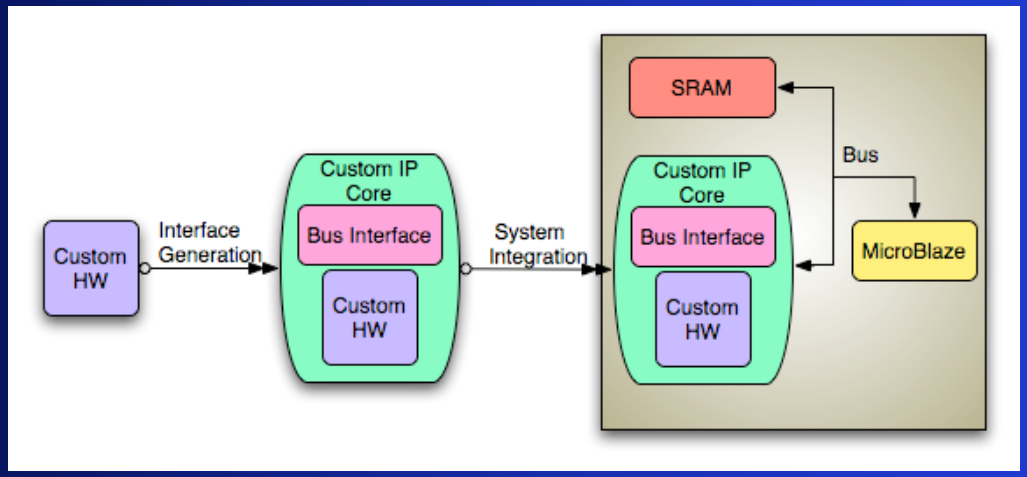
Monitoring Software Execution On The FPGA

- 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 windows.
    - 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.
  - 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>!”
- 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).

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?