XAPP564 (v1.0.2) January 29, 2007

PPC405 Lockstep System on ML310

Author: Harn Hua Ng

Summary

This application note describes the implementation of a processor lockstep system using embedded PowerPC™ 405 (PPC405) processors in Xilinx Virtex™-II Pro FPGAs, along with Xilinx software tools. To verify lockstep functionality, users learn how to build and run the Linux operating system with the MontaVista Linux Preview Kit and also how to probe signals in the lockstep system with Xilinx ChipScope™ Pro tools.

Introduction

A typical processor lockstep system is one in which two processors execute the same instructions simultaneously, based on the same clock. While one processor updates memory and I/O, the other performs instruction or data integrity checks and reacts to mismatches.

This application note describes how to create a processor lockstep system with multi-processor Virtex-II Pro FPGAs and Xilinx Platform Studio (XPS). In this design, an instruction integrity check is done in hardware, through an error checker module that flags a lockstep mismatch by dimming an LED.

To demonstrate processor lockstep execution in a real-life application, this design runs the Linux operating system built with a combination of XPS and MontaVista Linux tools. The design was tested on the Xilinx ML310 Embedded Development Platform.

Contents

- "Lesson I: Create a Lockstep Design in XPS," page 4 shows how to build a lockstep design with a Virtex-II Pro FPGA containing two PPC405 processors.
- "Lesson II: Create a Bootloader Program," page 9 shows how to create a bootloader program to synchronize the processors and start Linux.
- "Lesson III: Configure and Build a Linux Kernel," page 10 shows how to configure and compile Linux for this lockstep system.
- "Lesson IV: Run Hardware and Software Together," page 11 shows how to run lockstep hardware and software on the board.
- "Lesson V: Force a Lockstep Error," page 12 shows how to force a lockstep error.
- "Lesson VI: (Optional) Include ChipScope Pro in the Lockstep System," page 12 shows how to use the ChipScope Pro Analyzer to probe signals and verify lockstep execution.

Included Files

Table 1 describes the files contained in the XPS project located in the xapp564.zip file.

Table 1: **Files in the XPS Project**

File	Description
ml310_lockstep.bit	Lockstep bitstream
ml310_lockstep_chipscope.bit	Chipscope-enabled bitstream
bootloader_lockstep/bootloader.c	Bootloader program source code
ramdisk.image.gz	Linux RAM disk image
zImage.initrd.elf	Linux kernel image
ml310_lockstep.cdc	(Optional) Chipscope Pro Inserter project
ml310_lockstep.cpj	(Optional) Chipscope Pro Analyzer project
ml310_lockstep/	XPS project files directory

Required Hardware/Tools

- Xilinx ML310 board (Rev. D)
- Xilinx ISE 6.3.02i (G.37)
- Xilinx Platform Studio 6.3i (Gmm.10.1)
- MontaVista Linux 3.1 Preview Kit (for Xilinx ML300)
- (Optional) ChipScope Pro 6.3i

System Overview

Hardware Lockstep

To synchronize the operation of the processors, the two PPC405s are combined in a custom hardware module. A block diagram of this dual processor module is shown Figure 1.

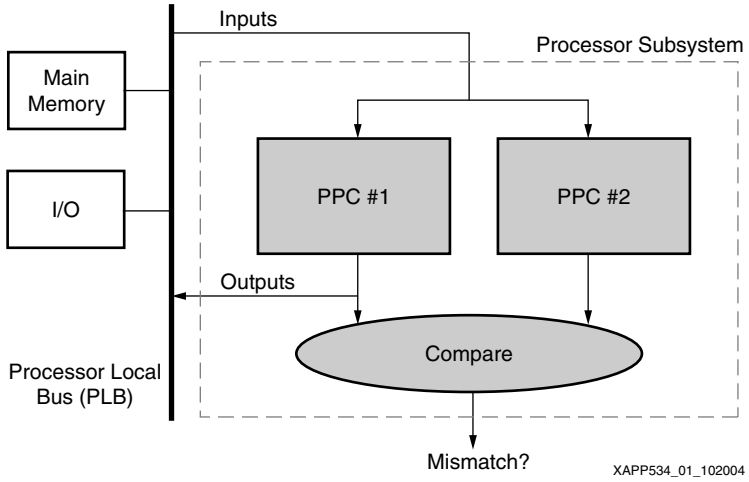

Figure 1: **Dual-Processor Module Block Diagram**

PPC #1 receives the same inputs as PPC #2 and both processors execute the same instructions simultaneously. However, only PPC #1's outputs are delivered to the rest of the system, for example, the memory and the I/O.

Error Checking and Signaling

A logical mismatch of the processors' instruction request signals[1] (PLBICUREQUEST) flags an error signal. The PLBICUREQUEST signal indicates if the processor's ICU is asking for an instruction from a PLB slave device - in this case, PLB DDR memory. If PPC #1's execution is out-of-sync with that of PPC #2, this PLBICUREQUEST signal will be asserted at different times and the resulting mismatch permanently turns off an LED (labeled DBG1) on the ML310 board.

In this application note, only one signal (PLBICUREQUEST) is used to check for a lockstep mismatch in this design. Users are encouraged to use multiple signals to implement a more complicated lockstep system.

Communicating with the Processors

The outside world communicates with each processor via the JTAGPPC[2] module, so in the dual processor module, both processors must be daisy-chained, as shown in Figure 2.

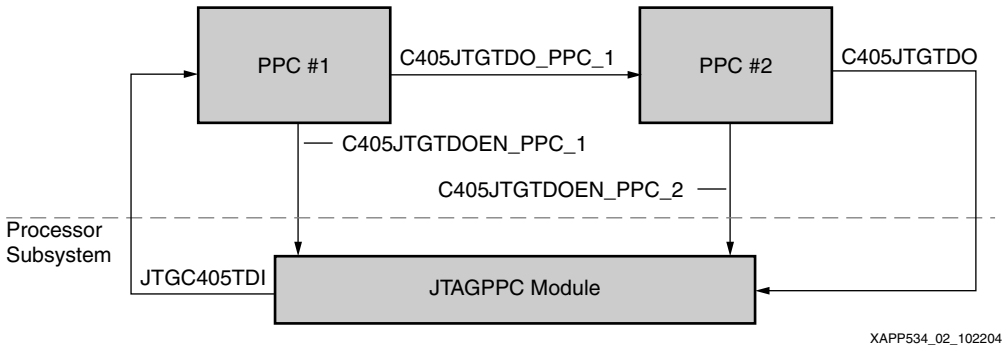

Figure 2: **Daisy-Chained Processors**

Software Lockstep

The Linux operating system is widely used by engineers in industry and is a suitable software application to test the lockstep system. Through running Linux, the lockstep system demonstrates that the two processors maintain synchronization while executing a complex software application. Other types of software programs may also be run on the lockstep system.

JTAG commands are used to start software execution on Virtex-II Pro FPGA PowerPC processors. In a lockstep system, because the processors are daisy-chained, processors along the chain receive JTAG commands at different times, resulting in out-of-sync execution.

A bootloader program, covered in "Lesson II: Create a Bootloader Program," page 9, is needed to reset both processors and to start executing software simultaneously on both processors. The sequence of events is shown in Figure 3, page 4.

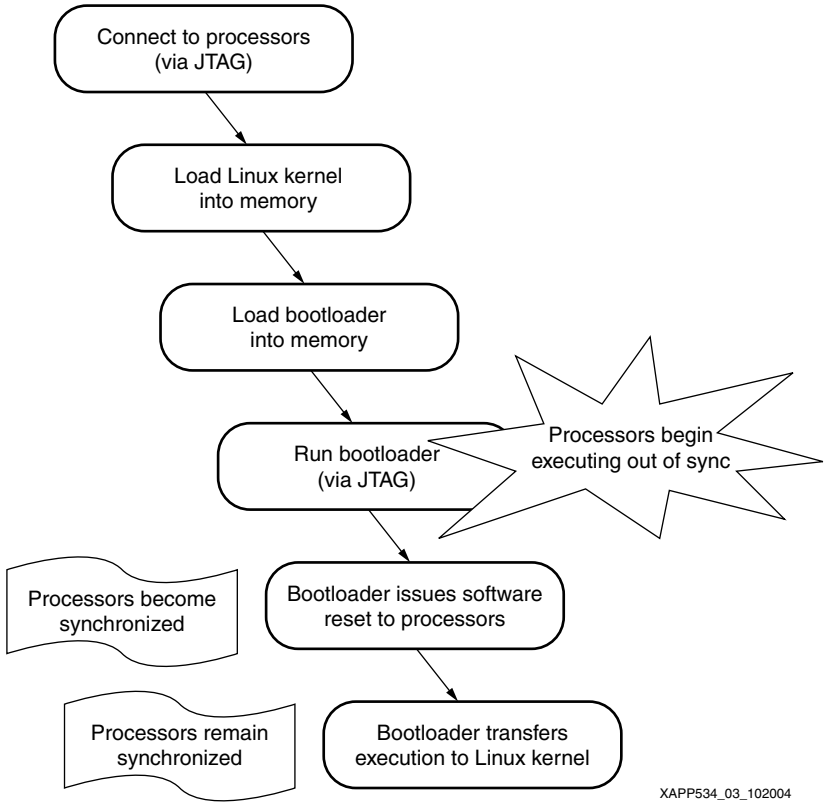

Figure 3: **Bootloader Program Sequence**

Signal Probing

To debug and verify lockstep operation, ChipScope Pro Integrated Logic Analyzer (ILA) cores are added into the system. For example, the PLBICUREQUEST signal from each processor can be viewed in a waveform window. Optionally, the "Step-By-Step Instructions" section in this application note explains how to implement ChipScope cores in this lockstep system.

Step-By-Step Instructions

The following sections contain detailed instructions on creating the lockstep system. A complete design, located in the xapp564.zip file, is provided for your reference.

Files created in this design are organized into the following paths:

```
~/lockstep
          /ml310_lockstep       (XPS project)
          /bootloader_lockstep  (Bootloader software application)
          /linux_lockstep       (Linux operating system)
```

Lesson I: Create a Lockstep Design in XPS

XPS provides a wizard interface called the Base System Builder (BSB) to configure and build a system from scratch. BSB helps in obtaining an initial system that is then modified for lockstep functionality.

Table 2 lists the IP contained in the Lockstep System.

Table 2: **Lockstep System IP**

IP Modules	Descriptions
BRAM memory, PLB_BRAM controller	Manages instruction and data accesses to BRAM memory.
JTAGPPC module	Downloads software into memory blocks and debugs the PowerPC processors.
PLB, OPB buses, PLB2OPB bridge	Handles data transfers between the PLB and OPB system buses.
OPB UART16550	Serial input/output to interact with software.
OPB Interrupt Controller	Handles interrupts from system peripherals, such as the OPB UART16550.
PLB DDR memory, PLB DDR controller	Manages instruction and data accesses to DDR memory.
Processor reset module	Initializes the PowerPC processors and system components to a known state.
Custom dual PPC405 wrapper module	Lockstep dual PowerPC module.
DCM modules	Provides clocks for the PowerPC processors and system components.

1. Create an XPS project
    - Create a ~/lockstep/ml310_lockstep directory
    - Start XPS and select “Base System Builder Wizard” from the initial popup dialog
    - Choose a directory for this project, for example, ~/lockstep/ml310_lockstep, and click **OK**
    - Select **I would like to start a new design** and click **Next**

In the following screens, unless otherwise noted, click **Next** or hit **<Alt+N>** to continue.

2. Choose ML310 as the Target Board
    - Board Vendor: Xilinx
    - Board Name: Virtex-II Pro ML310 Evaluation Platform
    - Board Revision: D
3. Select PowerPC as the processor
4. Select Clocking, PPC405 Processor Engine, and On-Chip Memory options
    - Reference Clock Frequency = Processor Clock Frequency = Bus Clock Frequency = 100 MHz
    - PowerPC 405 Processor Engine: FPGA JTAG
    - Cache: Enabled
5. IO devices
    - RS232 UART: OPB_UART16550, Configure as UART 16550, use interrupt
    - DDR SDRAM: PLB_DDR, use interrupt
    - *Deselect* SPI EEPROM, LEDs_8Bit, LCD_OPTIONAL, pci_arbiter_0, PCI32_BRIDGE, and SysACE CompactFlash
    - IIC Bus: OPB_IIC, use interrupt
    - PLB_BRAM_IF_CNTLR: 16kB

6. Cache setup: Mark all memory regions as cacheable.
7. Software Configuration
   - *Deselect* Generate Sample Application and Linker Script

   Note: Linux, the target application, is compiled separately, so a sample application is not required.
8. Summary screen: address map of the lockstep system.
   - Verify that all the required peripherals are included, and click Generate or **<Alt+G>** to continue.
   - Click "Finish" or hit **<Enter>** to build the system.
9. Save and close the project.

From this point on, manually edit files to create a dual processor wrapper and then customize project files to reflect changes in the processor wrapper.

10. Changes to the processor wrapper comprise of:
    - Instantiate an additional processor core.
    - Add signals to connect JTAG interfaces of both processors.
    - Add lockstep mismatch signal and the comparison logic that generates it.

    To edit the PPC405 wrapper, a local copy of the code is needed.
    - Copy $XILINX_EDK/hw/XilinxProcessorIPLib/pcores/ppc405_v2_00_c

      to ~/lockstep/ml310_lockstep/pcores/
11. Add JTAG and lockstep mismatch signals to the processor wrapper interface.
    - Start a text editor and open ~/lockstep/ml310_lockstep/pcores/ppc405_v2_00_c/hdl/vhdl/ppc405_top.vhd
    - In entity *ppc405_top* port list, replace

      ```
      C405JTGTDOEN : out std_logic;
      ```

      with

      ```
      C405JTGTDOEN_PPC_0 : out std_logic;
      C405JTGTDOEN_PPC_1 : out std_logic;
      ```
    - To the end of the portlist, add

      ```
      lockstep_error : out std_logic;
      ```
12. Declare JTAG connections between the two processors as well as signals used to generate the lockstep mismatch.
    - In the **signal declaration** section, add

      ```
      -- shared signal between two ppc405s
      signal C405JTGTDO_PPC_0 : std_logic;
      -- comparison of signals between the two ppc405s
      signal C405PLBICUREQUEST_PPC_0 : std_logic;
      signal C405PLBICUREQUEST_PPC_1 : std_logic;
      ```
13. Assign output signals for PPC #1's instruction request and for lockstep mismatch detection.

In the **begin** block

```
-- output assignments
C405PLBICUREQUEST <= C405PLBICUREQUEST_PPC_0;
-- flip-flop to store lockstep error signal
   process(PLBCLK, RSTC405RESETCHIP, RSTC405RESETCORE,
           RSTC405RESETSYS, C405PLBICUREQUEST_PPC_0,
           C405PLBICUREQUEST_PPC_1)
    begin
```

```
    if ( (RSTC405RESETCHIP = '1') or
       (RSTC405RESETCORE = '1') or
       (RSTC405RESETSYS  = '1') )
     then
      lockstep_error <= '0';
    elsif (PLBCLK = '1' and PLBCLK'event) then
      if (C405PLBICUREQUEST_PPC_0 /= C405PLBICUREQUEST_PPC_1)
      then
        lockstep_error <= '1';
      end if;
    end if;
  end process;
```

14. Duplicate the module instance **ppc405_i** and rename each instance **ppc405_i_0** and **ppc405_i_1** respectively.
15. In module instance **ppc405_i_0**
    - Replace

      ```
      C405PLBICUREQUEST   => C405PLBICUREQUEST,     -- O
      ```

      with

      ```
      C405PLBICUREQUEST   => C405PLBICUREQUEST_PPC_0, -- O
      ```

    - Replace

      ```
      C405JTGTDO          => C405JTGTDO,              -- O
      C405JTGTDOEN        => C405JTGTDOEN,            -- O
      ```

      with

      ```
      C405JTGTDO          => C405JTGTDO_PPC_0,        -- O
      C405JTGTDOEN        => C405JTGTDOEN_PPC_0,      -- O
      ```

16. In module instance **ppc405_i_1**, assign all outputs to “open” because the system uses only the outputs from the first PPC. Any output port (marked with “-- O”) should be connected to “open”, for example:

    ```
    C405RSTCHIPRESETREQ   => open, -- O
    ```

    - Replace

      ```
      C405PLBICUREQUEST     => C405PLBICUREQUEST, -- O
      ```

      with

      ```
      C405PLBICUREQUEST     => C405PLBICUREQUEST_PPC_1, -- O
      ```

    - Retain

      ```
      C405JTGTDO            => C405JTGTDO, -- O
      ```

    - Replace

      ```
      C405JTGTDOEN          => C405JTGTDOEN, -- O
      ```

      with

      ```
      C405JTGTDOEN          => C405JTGTDOEN_PPC_1, -- O
      ```

    - Replace

      ```
      JTGC405TDI            => JTGC405TDI, -- I
      ```

      with

      ```
      JTGC405TDI            => C405JTGTDO_PPC_0, -- I
      ```

17. Changes to ports must be reflected in the **ppc405_v2_1_0.mpd** file that describes the input and output ports of the **ppc405_v2_00_c** module.
    - Add write permissions to ~/lockstep/ml310_lockstep/pcores/ppc405_v2_00_c/data/ppc405_v2_1_0.mpd
    - Open the above file in a text editor
    - Replace

      ```
      PORT C405JTGTDOEN = C405JTGTDOEN, DIR = OUT, BUS = JTAGPPC
      ```

      with

      ```
      PORT C405JTGTDOEN_PPC_0 = C405JTGTDOEN_PPC_0, DIR = OUT, BUS = JTAGPPC
      PORT C405JTGTDOEN_PPC_1 = C405JTGTDOEN_PPC_1, DIR = OUT, BUS = JTAGPPC
      ```

    - Add

      ```
      PORT lockstep_error = "", DIR = OUT
      ```

18. Now that there are two processors inside a single wrapper, references to the second processor outside the wrapper must be removed.
    - Edit ~/lockstep/ml310_lockstep/system.xmp and remove the following three lines (at the end):

      ```
      Processor: ppc405_1
      BootLoop: 0
      XmdStub: 0
      ```

19. Edit ~/lockstep/ml310_lockstep/system.mhs and change the ports in the JTAGPPC controller and PPC405 blocks. (Cross-check with **system.mhs** in the reference design.)
    - Add signal **lockstep_error** as a debug LED in the global port list:

      ```
      PORT debug_led1 = lockstep_error, DIR = OUTPUT
      ```

    - Remove the **ppc405_1** instance
    - Add to the **ppc405_0** instance:

      ```
      # lockstep changes
      PORT JTGC405TDI = JTGC405TDI0
      PORT C405JTGTDO = C405JTGTDO
      PORT C405JTGTDOEN_PPC_0 = C405JTGTDOEN0
      PORT C405JTGTDOEN_PPC_1 = C405JTGTDOEN1
      PORT lockstep_error = lockstep_error
      ```

    - In the **jtagppc_cntlr** instance, remove the following line:

      ```
      BUS INTERFACE JTAGPPC1= jtagppc_0_1
      ```

      Add the following lines:

      ```
      # lockstep changes
      PORT JTGC405TDI0 = JTGC405TDI0
      PORT JTGC405TDI1 = JTGC405TDI1
      PORT C405JTGTDO1 = C405JTGTDO
      PORT C405JTGTDOEN0 = C405JTGTDOEN0
      PORT C405JTGTDOEN1 = C405JTGTDOEN1
      ```

20. Edit ~/lockstep/ml310_lockstepdata/system.ucf
    - Add pin mappings and IO constraints for the lockstep error LED

      ```
      Net debug_led1 LOC = G13;
      Net debug_led1 IOSTANDARD = PCI33_3;
      Net "debug_led1*" TIG;
      ```

      This LED corresponds to the LED labeled, “DBG1”, on the ML310.
21. Edit **system.mss** and remove the “OS” and “Processor” blocks which contain “ppc405_1”

22. Generate the hardware bitstream
    - Open ~/lockstep/ml310_lockstep/system.xmp in XPS
    - Select **Tools→Update Bitstream**

      An FPGA configuration bitstream will be generated as ~/lockstep/ml310_lockstep/implementation/download.bit.

This concludes the hardware changes. Next, create a bootloader program that resides in the FPGA's built-in block RAM.

Lesson II: Create a Bootloader Program

1. Open **system.xmp** in XPS and click on the Applications tab.
    - Right-click on “Software Projects” and select “Add SW Application Project…”
    - Use “bootloader” as the project name and “ppc405_0” as the processor.
2. In a separate terminal, create a ~/lockstep/bootloader_lockstep directory.
    - Start a text editor and enter the following C code in ~/lockstep/bootloader_lockstep/bootloader.c

```
#include "xpseudo_asm.h"
/*****************************************************************
* Reset processors and load Linux
*****************************************************************/
static int first = 0; /* Keep track of program execution.
                         Initialized to 0 at the beginning,
                         when software reset occurs
                         this variable is set to 1.
                         After both processors come out of
                         reset, they will execute in lockstep.
                         At that point, by examining this variable
                         (the value of which is now 1), the processors
                         know that (a) reset has occurred; (b) they are
                         are now in lockstep; and (c) they are ready to
                         execute a branch to Linux's start address.
                      */
int main()
{
  void (*f)(void); /* function pointer to Linux start address */

  if (first == 0) { /* software reset has not occurred */
    first = 1; /* set flag */
    mtspr(0x3f2, 0x30000000); /* reset processors via DBCR0 reg */
  } else { /* software reset has occurred */
    f = (void *)0x400000; /* assign Linux start address */
    (*f)(); /* start Linux kernel */
  }

  return 1; /* should never reach here unless reset circuitry is
               connected erroneously */
}
```

    - Save and exit.
3. In XPS, add the above source code file to the bootloader software project.
    - Right-click on “Sources” under “Project: bootloader” and select **Add Source…** to add ~/lockstep/bootloader_lockstep/bootloader.c

4. Set software project options for Bootloader project.
   - Right-click on “Compiler Options” and select **Set Compiler Options…**
   - Specify **`0xffffc000`** as the “Program Start Address”
   - Switch to the “Optimization” tag and select “No Optimization”
   - Click **OK**
5. Compile the bootloader.
   - Right-click on **Project: bootloader** and select **Build Project**

   The output ELF file is generated as ~/lockstep/ml310_lockstep/bootloader/executable.elf.

Lesson III: Configure and Build a Linux Kernel

Linux Configuration with MontaVista Linux 3.1 Preview Kit

In this system, Linux is configured to boot from a RAM disk file system that exists solely in volatile memory. This approach simplifies the boot process.

Assuming that MontaVista Linux 3.1 Preview Kit is installed and the environment properly set up, your PATH environment variable *must* contain references to the MontaVista Linux tools, for example:

```
export
PATH=/opt/montavista/previewkit/host/bin:/opt/montavista/previewkit/ppc/4
05/bin:$PATH
```

1. Copy a Linux kernel source tree from your MontaVista Linux 3.1 Preview Kit installation to a local directory, for example, ~/lockstep/linux_lockstep.

   ```
   $ cd ~/lockstep/linux_lockstep
   $ tar cf - -C /opt/montavista/previewkit/lsp/xilinx-ml300-previewkit-
   ppc_405/linux-2.4.20_mvl31 . | tar xf -
   ```
2. Configure the Linux kernel. (This is a recommended setup, feel free to configure the kernel to your specific requirements.)

   ```
   $ cd ~/lockstep/linux_lockstep
   $ make menuconfig
   ```

   A configuration screen comes up with Linux options. Use the cursor keys to navigate, **<Spacebar>** to cycle between options, and **<Enter>** to select.
   - Enable (mark with an *) the following items and disable (clear the checkbox) items where indicated.

     ```
     General Setup
         Disable PC PS/2 style keyboard
         Disable Networking support
         Disable Default bootloader arguments
     Block devices
         Disable Xilinx on-chip System ACE
     Console drivers
         Frame-buffer support
             Disable Support for frame buffer devices (EXPERIMENTAL)
     Character devices
         Disable Virtual terminal
     ```
   - Exit and Save this kernel configuration.

Linux Configuration in XPS

1. Open the lockstep project in XPS and select **Project→Software Platform Settings**
2. In the Kernel and Operating Systems section, choose **linux_mvl31 with OS version 1.00.a**
3. Switch to the Library/OS Parameters tab and enter the following values in the specified fields:

```
MEM_SIZE = 0x8000000
(Linux addresses 128 MB of memory on the ML310)
PLB_CLOCK_FREQ_HZ= 100000000
TARGET_DIR= ~/lockstep/linux_lockstep
(where Linux kernel source tree is located)
connected_periphs= RS232_Uart, OPB_IIC, opb_intc
```

   Click **OK**. If TARGET_DIR refers to a valid location, XPS copies the header and source files to <*TARGET_DIR*>.
4. Click **OK** to save the software settings.
5. Generate the Linux BSP by selecting **Tools→Generate Libraries and BSPs**. Files needed for Linux compilation will be copied to ~/lockstep/linux_lockstep.

Compile the Linux Kernel

After configuring the Linux kernel and generating a corresponding BSP, the final step is to compile the kernel. First, copy a default RAM disk image from the provided reference design to ~/lockstep/linux_lockstep. Linux will use this image as the RAM disk file system.

```
$ mkdir ~/lockstep/reference
$ unzip -d ~/lockstep/reference xapp564.zip
$ cp ~/lockstep/reference/ramdisk.image.gz
~/lockstep/linux_lockstep/arch/ppc/boot/images
$ cd ~/lockstep/linux_lockstep
$ make dep zImage.initrd
```

The resulting kernel image file is ~/lockstep/linux_lockstep/arch/ppc/boot/images/zImage.initrd.elf.

Lesson IV: Run Hardware and Software Together

Program FPGA with Lockstep System

Use Impact 6.3 to program the FPGA via JTAG connections on the ML310 board.

Run Bootloader and Linux Kernel

1. Connect a serial cable from a host computer to the ML310
2. Start a terminal, such as HyperTerminal, and set the serial port to 9600 8N1
3. Program the ML310 board with the hardware bitstream
4. Start a Xilinx Microprocessor Debugger (XMD) session.

```
$ xmd
```

   Connect to the PowerPC

```
XMD % connect ppc hw
```

   Reset the system

```
XMD % rst
```

   Download the Linux kernel

```
XMD % dow ~/lockstep/linux_lockstep/arch/ppc/boot/images/zImage.initrd.elf
```

   Download the bootloader program

```
XMD % dow ~/lockstep/ml310_lockstep/bootloader/executable.elf
```

Run the software in lockstep.

```
XMD % run
XMD % exit
```

At this point, Linux should boot on the terminal screen and the DBG1 LED should be lit. To log on, use “root” as the user name; a password is not required.

Lesson V: Force a Lockstep Error

To force a lockstep error, start an XMD session and connect to either processor. The “connect ppc hw” directive stops one of the processors, effectively breaking the lockstep execution.

1. Start an XMD session

   ```
   $ xmd
   ```

2. Connect to the PowerPC

   ```
   XMD % connect ppc hw
   ```

At this point, the DBG1 LED on the ML310 will turn off, signaling a lockstep mismatch.

Lesson VI: (Optional) Include ChipScope Pro in the Lockstep System

Signals can be monitored using Xilinx ChipScope Pro software. For information on how to use the ChipScope Pro Inserter and Analyzer tools, see the *ChipScope Pro Software and Cores User Guide* (for v6.3i) [Ref 1].

1. Extract a netlist with information needed for ChipScope from the original netlist.

   In ~/lockstep/ml310_lockstep/implementation/:

   ```
   > ngcbuild -i system.ngc system2.ngc
   ```

2. Use the ChipScope Pro Core Inserter with **system2.ngc** as the input file and **system3.ngc** as the output file.
   - Attach the signals of interest to the ChipScope core, for example, lockstep_error and PLBICUREQUEST.
   - Save the Inserter project as ~/lockstep/ml310_lockstep/ml310_lockstep.cdc
3. Replace the original netlist with the output netlist from ChipScope Pro Core Inserter.

   In ~/lockstep/ml310_lockstep/implementation/:

   ```
   > cp system3.ngc system.ngc
   ```

4. Re-implement the hardware design.
   - Open the lockstep project in XPS
   - Choose **Tools→Update Bitstream**

   The output bitstream is ~/lockstep/ml310_lockstep/implementation/download.bit
5. View the lockstep signals.

   After programming the FPGA, use ChipScope Pro Analyzer to import ~/lockstep/ml310_lockstep/ml310_lockstep.cdc and set a trigger, such as on the lockstep_error signal.

Resource Utilization

Compared to a system before lockstep modification, the lockstep modifications use two additional 4-input Look-up Tables (LUTs).

References

1. UG029: *ChipScope Pro Software and Cores User Guide*
2. UG018, *PowerPC 405 Processor Block Reference Guide*
3. UG011, *PowerPC Processor Reference Guide*
4. ML310 Embedded Development Platform website, http://www.xilinx.com/ml310
5. MontaVista Linux 3.1 Preview Kit for ML300, http://www.mvista.com/previewkit

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/27/04	1.0	Initial Xilinx release.
02/04/05	1.0.1	Corrected code in Lesson I, Step 13 on Page 7.
01/29/07	1.0.2	Revised URL link to *PowerPC Processor Reference Guide* (UG011).