Debugging Kernel Boot Problems

- Using Git
- Open Source Linux (OSL)
- QEMU System Model
- Archive
  - Using Git
  - Open Source Linux (OSL)
    - OSL Main Page
    - PowerPC Linux
    - PowerPC Linux On ML510
    - MicroBlaze Linux
    - MicroBlaze Linux - Little Endian
    - Kernel Debug
    - Device Drivers
    - Development Branch
    - FPU on PowerPC 440
    - Expanding A File System
  - Device Tree Generator

Site Utilities

- Home
- What is a Wiki Site?
- How to edit pages?
- How to join this site?
- Recent changes
- List all pages
- Page Tags
- Site members
- Site Manager

Add a new page

New Page

Fold

Table of Contents

Most of this information is based on past problems that were encountered and may be fixed now.

General

- If you are using JTAG pod to load the kernel into memory, after you have started the kernel and see no output, stop the processor using the pod (XMD%stop). Read the processor registers (XMD%rrd) and note the address of the program counter (pc) register. If the address is in virtual memory (0xCxxxxxxx), then from the root of the linux kernel tree, disassemble the vmlinux image using objdump (using the ELDK, >ppc_4xx-objdump -d vmlinux > dis.out) so that the address found in the program counter can be viewed. The file, dis.out, can then be viewed and you can determine what function it may be stuck in.
  - I have seen it stuck in probe_machine when the Xilinx virtex.c file didn't get linked in because the kernel configuration was wrong. In this case the kernel never booted, just the bootstrap loader had any console output.
  - I think (not sure I remember correctly) I have seen it stuck in calibrate_delay when the timebase-frequency was not set in the device tree. In this case the kernel may have console output and show it's stuck there.

Booting from GenAce

*If you need to boot from GenAce, you will have to do this a little differently. You can still use the JTAG tools and XMD to check the memory and registers.

No Console Ouput At All

- You forgot to load a design into your FPGA. Since the PowerPC is a hardcore it can still run software, but has no peripheral attached to it.
- The bootstrap loader does minimal device tree parsing but does use the linux-stdio-path statement to connect the UART to the std I/O. Make sure this line is in the chosen section of the device tree file.
- The version information of the UART in the device tree is not likely to be a problem. NS16550 in the compatible list should work for the Xilinx 16550 UART.
- I don't think the console drivers or console on the command line are needed for the bootstrap loader to have some output, but I'm not 100% sure on that. If in doubt, perform the checks for the console in the next section also.
- If using the custom FDT flow (gen-mhs-devtree), ensure that your Uart is included as the stdout in the dts file. It should look similar to the following:

```
chosen {
bootargs = " <your boot arguments > ";
linux,stdout-path = "/plb@0/serial@83000000";
}
```

- If nothing else works, you may need to dump the __log_buf assuming you are using some kind of JTAG pod to load the kernel into the board memory or using u-boot. Kernel console ouput is buffered in this buffer and then output to the std I/O, but if the console is not working you can still see the output in the memory buffer by using the JTAG pod to dump the memory and convert from hex ASCII to something readable.
  - From the root of the linux kernel, where vmlinux is located, get the virtual address of the variable __log_buf (assuming using the ELDK tools).
  - >ppc_4xx-objdump -t vmlinux | grep __log_buf
  - After the kernel is hung and not outputting anything to the console, use XMD to stop the processor and dump the log buffer from memory.
    - XMD>stop
    - XMD>mrd <virtual address of __log_bug>
    - Then you may need to convert it from hex ASCII to something more readable using a utility, or now there is a TCL file on the front page of the wiki, http://xilinx.wikidot.com, which is linux-syslog.tcl. This TCL will read the __log_buf and print it out nicely.
  - If using u-boot to load the kernel the following instructions should be used.
    - Convert the virtual address to an offset from a base address (virtual address - 0xC0000000). Add that offset to the real address of the kernel (0x00400000) which is the load address of the kernel with the bootstrap loader, the one you would load with XMD. XMD indicates the address the elf file is loaded into or objdump of the elf file will also show it.
    - After the kernel is hung and not outputting anything to the console:
      - Reset the board using a reset button for the CPU, not a power cycle such that u-boot should be running again.
      - From u-boot, dump the log buffer from memory using the following command.
      - uboot>md <real address of the log buffer>

Console Output From The Bootstrap Loader Only

Assuming you see some output from the bootstrap loader, but the kernel has no output, check the following items.

- Make sure the console is configured correctly in the kernel configuration. The UART driver must be selected and the UART must be specified as the console.
- The boot args for the kernel in the device tree, specifically the console baud rate, must agree with the baud rate on the UART itself. If they disagree, the boot strap loader will use the baud of the UART and the kernel will use the baud rate on the console in the boot args. No baud rate on the console defaults to 9600.
- Make sure the console is specified correctly in the kernel command line, console = ttyS0 for UART 16550 and console = ttyUL0 for UartLite.
- Versions of the PLB bus in the device tree file has been a problem in the past. The kernel checks the version of the PLB bus that is specified in the device tree file against a list contained in arch/powerpc/platforms/40x/virtex.c or arch/powerpc/platforms/44x/virtex.c depending on the processor. We added simple-bus to try to avoid the large number of versions in the future.
- With the simple-bus compatible PLB in the device tree, there was a bug that prevented the console from working. This was fixed in the Git tree and you should check to make sure you have the commit that fixed it.

commit 05ac320d0ef49b334ccc40de17cfc1ad8b309783
Author: John Linn <john.linn@xilinx.com>
Date: Fri Jun 27 09:23:00 2008 -0700
Xilinx: Don't use PPC_UDBG_16550 with Virtex platforms.

Devices Not Found During Boot

- Each device in the device tree specifies a version that the driver must be compatible with. If there is a new version of an IP core and the driver has not been updated, the kernel will not bind to that driver. For example, compatible = "xlnx,xps-ll-temac-1.01.a"; is in the device tree.

page revision: 3, last edited: 28 Oct 2011, 09:42 GMT (171 days ago)

Edit Tags History Files Print Site tools + Options