arch/ppc, arch/powerpc and Device Trees - A Walk Through A Port

Embedded Linux Conference – Europe
2nd November 2007

Hugh Blemings

Introduction

- Who am I ?
  - Working on Free Software since mid '90s
  - My profession since 1997 :)
    - With OzLabs at IBM Linux Technology Centre since 2001
  - Mostly work on PowerPC “stuff”, particularly Linux
  - Based in Canberra, Australia where we have kangaroos, unlike Austria
- What is this talk about ?
  - A Newbie's observations on porting Linux to a new hardware platform
    - I'm *not* an expert!
      - But am fortunate to work with a number of them
  - A brief introduction to device trees
- Questions welcome at any time

Linux and PowerPC

- First port of Linux to 32 bit PowerPC c. 1992
  - /arch/ppc
  - Apple PowerMacs, IBM RS/6000 & Embedded boards
- Linux ported to 64 bit PowerPC c. 2001
  - /arch/ppc64
  - IBM pSeries, Apple G5 PowerMacs & Embedded boards
- c. 2006 decision to consolidate 32 & 64 bit to single arch tree
  - /arch/powerpc
  - Some low level code unique to 32 or 64 bit implementation
  - Can build 32 bit kernel for 64 bit CPU
- 64 bit kernels handle 32 bit userspace with no overhead
  - don't need 64 bit cron
  - might need 64 bit emacs :)

What Are Device Trees ?

- Provide a (mostly!) consistent way of representing hardware properties to OS at boot
- Origins in Open Firmware used on PowerMacs, RS/6000 and SPARCmachines
- Typically passed from firmware to bootloader as binary blob of data
- Supported by LinuxPPC from early days
- Work by David Gibson et. al. to create a Device Tree compiler and associated tools
  - Allows a device tree to be created from a high level text representation
  - Library allows boot loader, kernel or even firmware code to get properties from tree in consistent manner
  - http://dtc.ozlabs.org
- More on Device Trees later

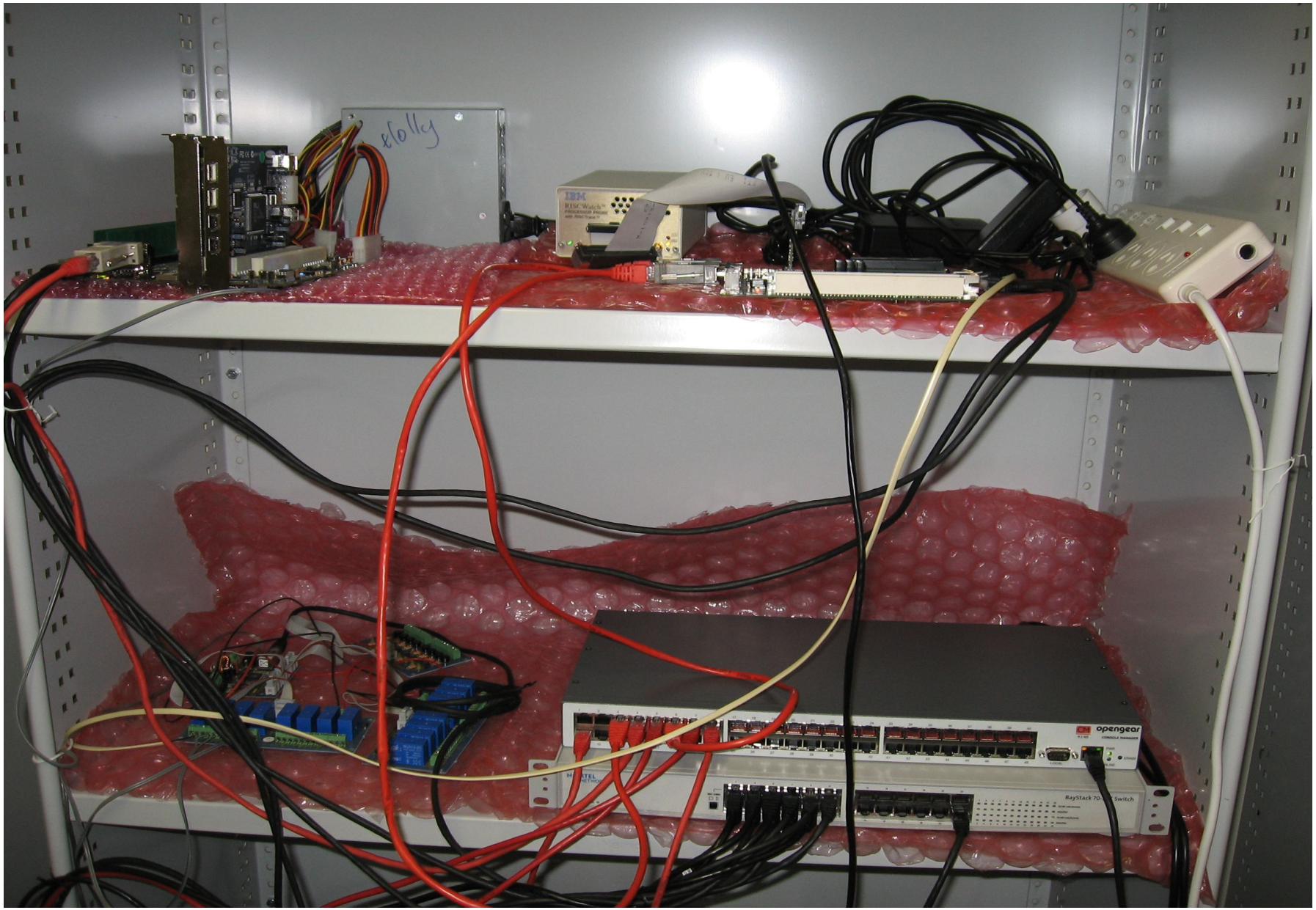
The Task At Hand

- Simple, really: Port the board support for the AMCC440GX Taishan Board from arch/ppc to arch/powerpc
- Needed to be done, but not time critical - ideal for me :)
- Had very experienced colleagues to draw on as well as existing arch/ppc port
  - Any of whom could have done the port in a few days

First Steps

- Taishan board set up with
  - Serial port server
  - Remote reset circuit
  - In circuit debugger

First Steps (continued)



First Steps (continued)

- Confirmed that would boot off the onboard flash image
- Grabbed the AMCC tree from the board support CD and built using my toolchain
  - `hugh@fandango2 $ arch=PPC make taishan_defconfig`
  - `hugh@fandango2 $ arch=PPC make -j`
- Made the resulting kernel available over tftp
- Board booted fine using NFS root
  - NFS roots useful to have lying around!
- At this point know build and boot environment ok

Getting the kernel boot loader working

- Used existing code in arch/powerpc/boot for ``Bamboo'' platform
- Added/modified until built cleanly, then tweak from there
- Following files were modified/added
  - arch/powerpc/configs/taishan_defconfig
  - arch/powerpc/boot/Makefile
  - arch/powerpc/boot/dts/taishan.dts
  - arch/powerpc/boot/dcr.h
  - arch/powerpc/boot/44x.h
  - arch/powerpc/boot/cuboot-taishan.c
  - arch/powerpc/boot/taishan.c
- Pretty quickly got boot messages working up to point where kernel called

Getting the kernel running

- Kernel will re-initialise serial ports
  - Can lead you astray if divisors wrong or other registers incorrect
- Following files were added/modified
  - arch/powerpc/Kconfig.debug
  - arch/powerpc/platforms/44x/Kconfig
  - arch/powerpc/platforms/44x/Makefile
  - arch/powerpc/platforms/44x/taishan.c
- Don't forget `debug` on kernel command line...

More on Device Trees – Typical CPU entry

```
cpus {
    PowerPC,440GX@0 {
        device\_type = "cpu";
        clock-frequency = <2FAF0800>; // 800MHz
        timebase-frequency = <0>; // Filled in by boot code
        i-cache-line-size = <32>;
        d-cache-line-size = <32>;
        i-cache-size = <8000>; /* 32 kB */
        d-cache-size = <8000>; /* 32 kB */
        dcr-controller;
        dcr-access-method = "native";
    };
};
```

More on Device Trees – On Chip Bus & UART

```
POB0: opb {
      UART0: serial@40000200 {
          device_type = "serial";
          compatible = "ns16550";
          reg = <40000200 8>;
          clock-frequency = <A8C000>;
          current-speed = <115200>;
          interrupt-parent = <&UIC0>;
          interrupts = <0 4>;
      };
      [Other devices...]
};
```

More on Device Trees – Accessing properties

```
/* From arch/powerpc/boot/devtree.c */

void dt_fixup_cpu_clocks(u32 cpu, u32 tb, u32 bus)
{
        void *devp = NULL;

        /* ... */

        while ((devp = find_node_by_devtype(devp, "cpu"))) {
                setprop_val(devp, "clock-frequency", cpu);

                setprop_val(devp, "timebase-frequency", tb);

                if (bus > 0)
                        setprop_val(devp, "bus-frequency", bus);
        }

        /* ... */
}
```

A little bit about handling source code

- Git usage very basic - mostly to pull current trees
- Quilt is wonderful
  - Modest learning curve time well spent
  - Tutorial would be a whole session in itself
- Keep work as patches
  - Makes tracking mainline much easier

Random Datapoints

- A neat conversion utility
  - iprint
- Object code in separate tree to kernel source
  - make O=../some-other-dir
- Device numbering fun
  - Linux thinks: eth0 and eth1
  - Hardware thinks: EMAC2 and EMAC3
  - PCB is labelled: Ethernet I and Ethernet II

Conclusions ?

- Code currently boots all the way up to a prompt
  - Debugging EMAC ethernet driver continues
- Remaining work includes
  - Incorporating driver for on board FLASH
  - Bootime selection between SRAM and L2 mode
  - Modifying ibm_new_emac to use phylib
- Create proof of concept device tree for an ARM CPU ?
- Macro support for Device Tree Compiler

Close

- Thankyou to aforementioned experienced colleagues...
  - benh, paulus and dgibson
- ... and to the audience for your time