Recent Plan 9 Work at Bell Labs

Geoff Collyer
geoff@plan9.bell-labs.com

Fifth International Workshop on Plan 9
Seattle, WA, USA

October 11-13, 2010

Introduction

Bell Labs recently ported Plan 9 to new systems with non-PC architectures. This is a status report on those ports and a summary of what helped, what didn’t and what could.

Most code just recompiled and worked. We had to create new kernels and in some cases new bootstraps.

The two main classes of target systems are PowerPC 4XX and recent ARMs (926 and Cortex-A8).

Outline

Non-PC Ports: Common Problems and Techniques

Memory Barriers and Caches

- use write-back caches where possible (not USB structures).

- need to use *barrier* instructions in some places (DMA, locks) to force previous instructions to execute and to force writes to memory:

- instruction barrier
- data barrier
- flush L1 cache
- flush L2 cache

PC largely conceals this with cache-coherent memory; PowerPC and ARM expose it.

- were conservative with barriers, but have been trimming them back now that the ports work.

Kernel Memory Mapping, Debugging, ARM Initialisation

- chose kernel virtual base address to allow trivial *k(un)map* and identity-mapping of I/O devices.
- possible because systems had at most 512MB of memory

- used normal serial port and LEDs on Virtex and IGEP
- other ARMs have combined JTAG and serial ports with special cables with USB connector at one end
- *usb/serial* can drive these, so can MS Windows
- all ports enable hardware watchdog timers to regain control (via reset) if kernel goes off the deep end

- use U-boot to load kernel and *plan9.ini*, which kernel parses.
- U-boot initialises lots of hardware; sometimes we rely on that because it’s too much work

Floating-Point Emulation, Debugging

- Virtex and Marvell Kirkwood have no FPUs, so kernel emulates them when they trap.
- FP used in *awk*, etc.
- ARM ports also emulate *ldrex*, *strex* and local *cas* instructions.
- used *paranoia* to catch buggy FP emulation
- wrote new program to verify *vlong* arithmetic; found bug in *qc* (now fixed)

PowerPC Ports: Blue Gene

- all PowerPC ports are CPU server kernels

- Blue Gene port for DoE project HARE; members are Bell Labs, IBM Research, Sandia National Labs and Vitanuova.
- looking for OS lighter than Linux, but not primitive like MPI, for machines with thousands or millions of cores

- networks were built specially by IBM and are unusual
- original port for BG/L with 64K PowerPC 440 cores
- current port for BG/P with 160K PowerPC 450 cores
- future work likely to move to BG/Q with at least 750K PowerPC cores

- paper by Charles Forsyth on aggressive 9P caching for mostly read-only files

- BG/P port source (*9k* kernel) available via

```
   hg clone http://bitbucket.org/ericvh/hare
   git clone http://git.anl-external.org/plan9.git
```

Virtex 4 and 5 Ports, Bootstrapping

- Xilinx Virtex 4 FX, 5 FXT evaluation boards: FPGA-based with 300MHz PowerPC 405 and 400MHz PowerPC 440
- have L1 but not L2 caches
- ported Plan 9 to demonstrate feasibility of a device for embedded systems

- CAD tool creates FPGA bit-stream image, includes initial SRAM contents
- stripped-down *9load* goes into initial SRAM contents, loads kernel via BOOTP and TFTP
- needs manual intervention to boot this way, so implemented `/dev/reboot` and `#ec` early; lets us boot new kernel remotely
- 2nd 440 CPU, if any, is put to sleep due to lack of L1 cache coherency

Virtex Performance, Availability

- few TLB entries in MMU, so now round power segments to 1MB multiples; lets us use larger pages if we want, and take fewer page faults and TLB reloads
- Ethernet is clunky, uses DMA engines and hardware FIFOs; every step initiated by processor, so little asynchrony
- our benchmark on diskless machines:

```
   cd /sys/src/9/kw
   { mk clean; mk; mk clean; time mk } >/dev/null
```

- Virtex 5 over Gb Ethernet results:

```
   44.32u 35.60s 99.80r     mk # virtex 5 gb
```

- small L1 caches and no L2 cache likely at fault

- future Virtex FPGA designs use ARM processors; mail me if you really want these ports

ARM Ports: Beagleboard

- all CPU server kernel so far; work in progress on video for terminals

- Beagleboard is a dog: no Ethernet, can’t even PXE boot over USB Ethernet
- OMAP35 system-on-chip (or sack-of-components), 500MHz ARM v7-a architecture Cortex-A8 CPU
- have OMAP35 video working (courtesy Per Odlund), more useful on Gumstix Overo
- OMAP35 USB buggy and not working yet, can’t really use Plan 9 on Beagleboard yet
- Cortex-A8 has new FPU; we don’t yet save/restore its registers but *5c* doesn’t generate the right op-codes yet

ARM Ports: IGEP, Gumstix Overo

- IGEPv2 is Beagle plus 100Mb/s SMSC 9221 Ethernet, etc.
- no flash access yet; not well documented
- when HDMI/DVI cable is inserted, Ethernet doesn’t work; may not be an ideal terminal

- Gumstix Overo uses very similar OMAP3503 SoC; just worked first time
- uses SMSC 9221 Ethernet, but video works at the same time
- we use FIFOs but not DMA for Ethernet; benchmark results:

```
24.58u 30.07s 70.65r     mk # igep 100mb
28.88u 38.38s 81.67r     mk # gumstix overo 100mb
```

ARM Ports: Sheeva/Guruplug, OpenRD & Performance

- based on Marvell’s Kirkwood SoC (88F6281)
- 1.2GHz ARM v5/6 architecture ARM 926EJ-S CPU
- Kirkwood less tedious to initialise than OMAP35
- Guruplug has two Gb Ethernets, runs hot but hardware fix is promised

- started with Inferno Kirkwood port
- have access to flash and crypto accelerator working
- Marvell 1116 Gb Ethernet DMAs in and out of buffer rings, so port is zippy and quite usable:

```
   16.66u 8.39s 31.44r      mk # guruplug gb
```

The Legendary AMD64 PC Port

- based on *9k* kernel; minimal CPU server for early DoE work, but work changed direction
- little recent work

- does not run 386 binaries
- limited set of drivers: better Ethernet controllers, UARTS
- not working yet: audio, video, USB, floppies, disks, PCMCIA
- video will be painful: either have to run VESA BIOS in 16-bit real mode or emulate it

- some legacy junk has been left behind deliberately
- also requires modified *9load*, *6[cal]*, *libmach*; these exist

Performance Summary

- only Virtex 5 lacks an L2 cache
- ‘issue’ column is max. instructions issued per clock cycle
- machines above line have buffer-ring Ethernet controllers

times	name	MHz	Ether	issue
44.32u 35.60s 99.80r	virtex 5	400	1Gb	2
28.88u 38.38s 81.67r	gumstix overo	600	100Mb	2
24.58u 30.07s 70.65r	igep	720	100Mb	2
20.78u 11.42s 49.48r	586 pc	500	100Mb	1
16.66u 8.39s 31.44r	guruplug	1,200	1Gb	1
2.95u 2.34s 8.31r	amd k8 pc	2,200	1Gb	1
2.26u 2.46s 4.01r	quad xeon pc	2,500	1Gb	2

Table 1: Performance Summary

- sub-GHz dual-issue machines are slowest
- slowest machines have clumsy DMA/FIFO/Ethernet combos; fastest have Ethernet controllers that use buffer rings, initiate DMA autonomously

Hardware Documentation

- Virtex docs okay; IBM parts better than Xilinx ones

- ARM docs are pretty sad all around
- need 3 or 4 manuals per SoC port: CPU arch. from ARM, CPU man. from ARM (maybe another from SoC vendor), SoC manual, board manual.
- they all point to each other, so you flip back and forth
- ARM arch. man.s are improving, V7 is actually pretty good
- CPU and Soc manuals are long but fuzzy, incomplete, buggy
- OMAP35 manual is 1 PDF of 3,500 pages; good PDF viewer is vital

Poor SoC Design & Documentation

- SoC manuals are really bad
- SoCs look like they were slapped together from existing device designs; hardly a system
- manuals reflect lack of cohesion: devices often described in isolation with little guidance about how or why to use them
- assumes background in hardware design; wrong for system programmers

- dubious SoC features: internal firewalls, unenabled clocks
- require much tedious initialisation just to make hardware get out of the way
- U-boot helps by doing some of the initialisation (but it’s still basically Linux)

Marvell & Board Documentation

- Marvell’s are low quality and almost all are proprietary
- even with NDA, it’s hard to get the ones you need
- Marvell believes Linux drivers are adequate hardware documentation. BZZZT! Wrong, but thanks for playing.

- Beagleboard docs are pretty thorough, IGEPv2 less so.
- Global Technologies documented Sheevaplug pretty well, others not as well

Lessons Learned & Recommendations

- for kernel debugging, connecting JTAG interfaces via USB to *db* or *acid* would help when CPU just goes away
- U-boot could help by printing processor state when it started
- dead simple *iprint* that always works would be a big help
- small addition to `portclock.c` catches infinite loops due to incorrect clock primitives
- one test added to `xalloc.c` catches cycles in the `Hole` list

- somebody should build a cheap yet sane and well-documented general-purpose ARM system
- hardware cache coherency requires less software effort, which improves correctness and performance, especially in multi-core systems
- design of Ethernet controllers can limit their performance and thus responsiveness of Plan 9

Recommendations (cont'd)

- future Ethernet controllers should use buffer rings and initiate DMA by themselves
- Ethernet via USB is disgusting and USB 2 has only ¼ the bandwidth of Gb Ethernet (480 Mb/s vs 2×1Gb)
- Gb Ethernet is cheap enough now that it should be standard

- hardware should come out of reset in a sane and usable state without further diddling of PHYs, clocks, or other nonsense
- hardware documentation should be aimed at system programmers, among others, and be public, complete, clear and concise. Ill-defined bits (GPIO, multi-purpose pins, etc.) need to be documented in great detail at CPU, SoC and board levels.

Acknowledgements

Salva Peiró and Mechiel Lukkien ported native Inferno to the Kirkwood SoC.

Rae McLellan helped me decrypt voluminous yet often unhelpful hardware documentation. He and Claudia Collyer provided useful comments on drafts of this paper.

These ports were made feasible by the existing Plan 9 PowerPC and ARM compiler suites, and made easier by existing Plan 9 kernels for other PowerPC and ARM systems.