Inferno for the Sheevaplug

Mechiel Lukkien

mechiel@xs4all.nl

ABSTRACT

The Sheevaplug is a ''development kit'' based on the *Marvell Kirkwood* system-on-chip. This chip has an ARM processor and controllers for USB 2.0, gigabit ethernet and SDIO, among others. *Inferno-kirkwood** is a port of Inferno to the Sheevaplug, and hopefully in the future to other devices based on this chip.

Introduction

Inferno-kirkwood is a port of Inferno to the Sheevaplug, a ''development kit'' based on the *Marvell Kirkwood* system-on-chip. Specifications of the Sheevaplug:

- ARM processor, 1.2 Ghz, with 16KB instruction and 16KB data caches, and 256KB L2 cache.
- 512MB DDR2 RAM, 512MB NAND flash
- serial console and JTAG interface, over USB
- single 1Gbps ethernet port
- single USB 2.0 port
- SDIO slot, supporting up to SDHC cards
- RTC, GPIO, hardware crypto support (for DES, AES, MD5, SHA1), and hardware XOR/DMA copy support.

The *kirkwood* chip has a second gigabit ethernet controller, a SATA controller, a PCI Express interface, and controllers for SPI and I2S. However, these require connectors or connector pins that are not present on the Sheevaplug. It is unfortunate that the Sheevaplug has no SATA connectors, storage must be delivered through USB. Other devices (e.g. *OpenRD*) with those connectors have become available recently.

The Sheevaplug uses just below 3 watts when idle. It comes with Linux that runs from flash, configures its network with DHCP and runs an ssh server with a default root password: getting started with the device is trivial.

The *Inferno-kirkwood* project was started after the Sheevaplug was mentioned on the Inferno mailing list. The Sheevaplug is reasonably cheap, has enough hardware on it to provide useful network services, and its hardware is documented. It looked like a nice device to get experience writing device drivers and other low-level code. Without specifications it would be highly unlikely I would be able to run Inferno on these devices. It did turn out that some documents referenced from the main specification document† were not publicly available, but that has not hampered development yet.

---

* *Inferno-kirkwood*, `http://code.google.com/p/inferno-kirkwood/`

† *Kirkwood function specification*, `http://www.marvell.com/files/products/embedded_processors/kirkwood/FS_88F6180_9x_6281_OpenSource.pdf`

The initial code to get a kernel booted, was written by me. Soon Salva Peiró got interested and started developing. Before his Sheevaplug arrived he developed remotely, connecting to a machine that had console access to a Sheevaplug. One of the first things he implemented was rebooting with `^t^tr`. Development has recently shifted to other projects, but is expected to shift back soon.

Progress

A kernel can already be booted, with a working serial console, real-time clock and ethernet controller. Some devices are partially supported, some do not have any support at all.

The UART, for serial console, was the first device to be used. We probably do not yet set all parameters for the serial console properly, instead relying on the boot loader *u-boot* to initialize them.

Ethernet has better support, but not all hardware features are supported. For example use of multiple queues for different types of network traffic each with its own packet memory, or TCP/UDP checksum offloading. Perhaps these will not be implemented in the future either though. Interrupt coalescing has already been implemented. Before the processors *dcache* can be turned on, the ethernet driver must be modified to flush the cache of the descriptors for packet memory.

The real time clock is read at start up and can be set as well. The RTC hardware also has an alarm, but we do not support it.

The NAND flash is detected but cannot yet be accessed. Once it is supported, Inferno can be booted from flash.

SD controller support is not complete either, but data can be read from SD cards. A FAT32 partition has already been mounted, but the driver has stability problems (kernel crashes). Writing to SD cards has not been tested but is not very different from reading.

A driver for the cryptography hardware has been written, but is not currently enabled. It is not clear if all hardware support will be used in the future. The DES library used in some parts of Inferno has a library interface incompatible with the hardware's interface to DES, so the changes required may be too intrusive and the gain too small. The hardware supports DMA for cryptography operations, but that has some caveats and is not the easiest way to use the crypto hardware.

The XOR controller copies memory using DMA and can optionally XOR data sources. This is aimed at RAID implementations, but perhaps plain DMA memory copies can be used by Inferno to lower CPU load.

GPIO signals have been set, but there is no generic driver that gives access to GPIO functionality yet.

Finally, we have a driver for the *efuse*: small memory that can be written once and read often.

History

Initial development went surprisingly quickly, considering how little experience with low-level programming I had. I started with Inferno on OpenBSD, but any other Unix that Inferno runs on would have worked just as well. The first goal was to create an Inferno kernel and convince the Sheevaplug to load it. The Sheevaplug comes with the *u-boot* boot loader, which supports BOOTP/DHCP and can fetch a kernel over tftp. The kernel has to be in *u-boot*'s *uimage* format, which turned out to be easy to create. Before my Sheevaplug arrived, I had a *mkuimage* program that would take an Inferno kernel and add a *uimage* header. Code from another Inferno ARM port was taken as a starting point. That gave me skeleton code that could be compiled into a kernel. I could turn that into an image that the Sheevaplug was willing to load and start executing

code from. Of course, the first kernel did nothing useful. Initially, I was not even sure which starting point addresses I had to put in the *uimage* header and kernel image, so whether *u-boot* was jumping to the right instructions. The best way to show a sign of life seemed to be to write a character to the serial console. I now usually connect my Sheevaplug through a power meter: power consumption shows me whether a broken kernel is hanging, spinning or causes too many interrupts. After some trial and error, trying a few kernel starting addresses and various UART configurations, the first character appeared on the console! That was encouraging. The existing Inferno ARM code was extremely helpful for getting to that point. After that, support for more controllers has gradually been added.

Developing

Currently, development proceeds as follows: The standard Inferno build process is used to create a kernel, in `os/kirkwood/`, which is then wrapped in a *uimage* header. The resulting image is copied to a tftp server (the `mkfile` has a target to copy it to `/n/tftp`). The Sheevaplug is rebooted, and the DHCP+tftp boot method fetches the new kernel and starts it. *U-boot* can also boot from flash, and from the SD card, and with the latest experimental version even from a USB mass storage device. Eventually we want to boot from flash, and probably also from SD card.

Future

Obviously there is still a lot to do. The current *inferno-kirkwood* code is not yet ready for normal use. This is exemplified by the lack of NAND flash support that would allow writing to and booting and running from flash. Beside the obvious need for making the code more stable, faster (enabling the *dcache*) and finishing existing drivers such as for SD cards, new drivers need to be written. Apparently the USB controller is a standard EHCI controller, so hopefully it will not be too hard to port Plan 9's USB EHCI driver. The SATA controller does not need support yet because the Sheevaplug does not have connectors for it. Other devices using the *kirkwood* chip have become available now, and they do have ESATA ports and the second gigabit ethernet port. They also have more USB ports and one device has a PCI Express-connected video card. Support could be extended to these devices, but it is not currently planned.

When enough hardware is supported, the Sheevaplug can provide network services. Inferno does not have programs for all common network services. Some of those I have started on, e.g. a DHCP server (only a BOOTP server exists in Inferno), a simple anonymous FTP server, an NFS server, etc. Plans for other network services exist too, e.g. an ssh 2 server. The goal is to replace an existing OpenBSD server with this more power-efficient Sheevaplug running Inferno, providing similar network services. Somewhat unfortunately (but unavoidably), implementing those has kept me from improving hardware support.