FreeBSD/arm on the Atmel AT91RM9200

Warner Losh
Timing Solutions, Inc
http://people.freebsd.org/~imp/bsdcan2006.html
BSDcan 2006
May 12, 2006

Experiences with FreeBSD/arm, the Atmel SoC and the embedded environment.

TIMING SOLUTIONS™

Outline

- Overview
- Background
- What is the Atmel AT91RM9200?
- How we use the AT91RM9200
- Why FreeBSD/arm?
- Adding SoC support to FreeBSD/arm
- Adapting to the Embedded World
- Experiences from this project

Acronyms In This Talk

- ACPI: Advanced Configuration and Power Interface
- ARM: Advanced RISC Machine
- BIOS: Basic I/O System
- CF: CompactFlash
- FPGA: Field Programmable Gate Array
- I2C: Inter-Integrated Circuit
- ISA: Industry Standard Architecture
- IRIG: Inter Range Instrumentation Group
- PITA: Pain in the Ass
- RISC: Reduced Instruction Set Computer
- RTOS: Real Time Operating System
- SBC: Single Board Computer
- SoC: System on a Chip
- SPI: Serial Peripheral Interface
- SSC: Serial Synchronous Control
- UART: Universal Asynchronous Receiver/Transmitter

Embedded vs Server

- Embedded
  - 8-64MB RAM
  - No Hard Disk
  - 2-16MB FLASH
  - NO BIOS
  - NO ACPI
  - Serial Bus
  - Limited Graphics
  - ARM, MIPS, SH4, PowerPC, etc
  - Each board different
- Desktop/Server
  - 128MB-32GB RAM
  - 60GB-1TB Hard Disk
  - NO FLASH
  - Extensive BIOS
  - ACPI
  - 32-bit or 64-bit Bus
  - Full Graphics
  - IA32 or AMD64
  - Most systems the same

Timing Solution's Hardware

- Existing systems use ia32 ISA SBC
  - No hot swap
  - 4U rack mount
  - FreeBSD/i386
- New system uses ARM SoC
  - Hot swap
  - 1U rack mount
- Systems read GPS or IRIG time code
- Systems distribute timing signals
  - PPS, 5MHz, 10MHz
  - IRIG, Optical Time Code

Atmel AT91RM9200

- ARM920T based SoC made by Atmel
- ARM Core standard
- ARM bus standard
- System peripherals unique
- Pins multiplexed
- Ethernet, USB, MMC/SD, I2C, SSC, UART, CF, SmartMedia, Flash and SDRAM i/f
- 16KB SRAM, 128KB ROM

Using the AT91RM9200

- Embedded system
- Recovers Time and creates Timing Signals
- ARM controls custom hardware
  - ARM does higher level tasks and coordination
  - Custom hardware for timing functions
- SNMP and Web management
- Reduced Cost

Alternatives to FreeBSD/arm

- OpenBSD/zaurus
- NetBSD/evbarm
- ARM Linux
- FreeRTOS
- eCos
- VxWorks

Why FreeBSD/arm?

- FreeBSD/i386 in other products
- FreeBSD/arm support newer than NetBSD
- FreeBSD/arm stable on real hardware
- In-house FreeBSD experience
- Extensive build system based on FreeBSD

Porting Checklist

- Determine scope of support
- Create a new directory under src/sys/arm
- Create boot loader, if necessary
- Create foo_machdep.c with initarm()
- Write device drivers for on chip devices
- Test Test Test

Embedding FreeBSD

- Size matters
- Cross Building and Debugging
- Reproduction of small footprint
- Upgrade path
- Making due with less
- More tools to make images needed
- Cross environment challenging

Experiences

- Ollivier Houchard is a huge asset.
- Mainstream drivers relatively easy
- Oddball drivers harder (no infrastructure)
- Boot loader unforeseen time sink
- Little hardware differences are a PITA
- “Cost Savings” in hardware are a PITA
- FreeBSD/arm easy to port
- Cross compile support good in FreeBSD

A Call to ARMs

- More SoC and board support
- More developers involved
- Vendor Relations
- More build options
- Images for “Cool” devices
- Better packaging support (FreeSBIE?)
- Better flash support
  - Common flash hardware interface
  - Flash layering (meta data per block)
- Embedded interfaces: I2C, SPI, MMC
- Documentation

Questions

Warner Losh
imp@freebsd.org
Timing Solutions, Inc
http://people.freebsd.org/~imp/bsdcan2006.html