

FreeBSD on high performance multi-core embedded PowerPC systems

Rafał Jaworowski

raj@semihalf.com, raj@FreeBSD.org

AsiaBSDCon 2009, Tokyo

Presentation outline

- Introduction
- PowerPC architecture background
- Existing FreeBSD/powerpc support
- MPC8572 port details
 - Overall scope
 - Multi-core support
 - Integrated peripherals
 - Current state summary (and TODOs)

Introduction

- **Definitions**
 - FreeBSD
 - Embedded system
 - PowerPC
 - Instruction-set architecture definition
 - Derived from POWER (RS/6000)
- **Focus on low level design of
FreeBSD/powerpc on MPC8572 (dual-core)**

PowerPC basics

- **Apple-IBM-Motorola (AIM)**
- **Now maintained by Power.org**
 - Power Architecture (note lower case!)
 - Covers all variations (POWER, PowerPC, Cell etc.)
- **Multiple vendors**
 - AMCC, Freescale, IBM, Xilinx
- **Widespread**
 - Embedded systems, supercomputers, game consoles

More about PowerPC

- **Highlights**

- RISC-like (load-store)
- Superscalar
- 32- and 64-bit

- **Book-E**

- More recent PowerPC variation
- Embedded applications profile
- Binary compatible with AIM (user instruction set level)

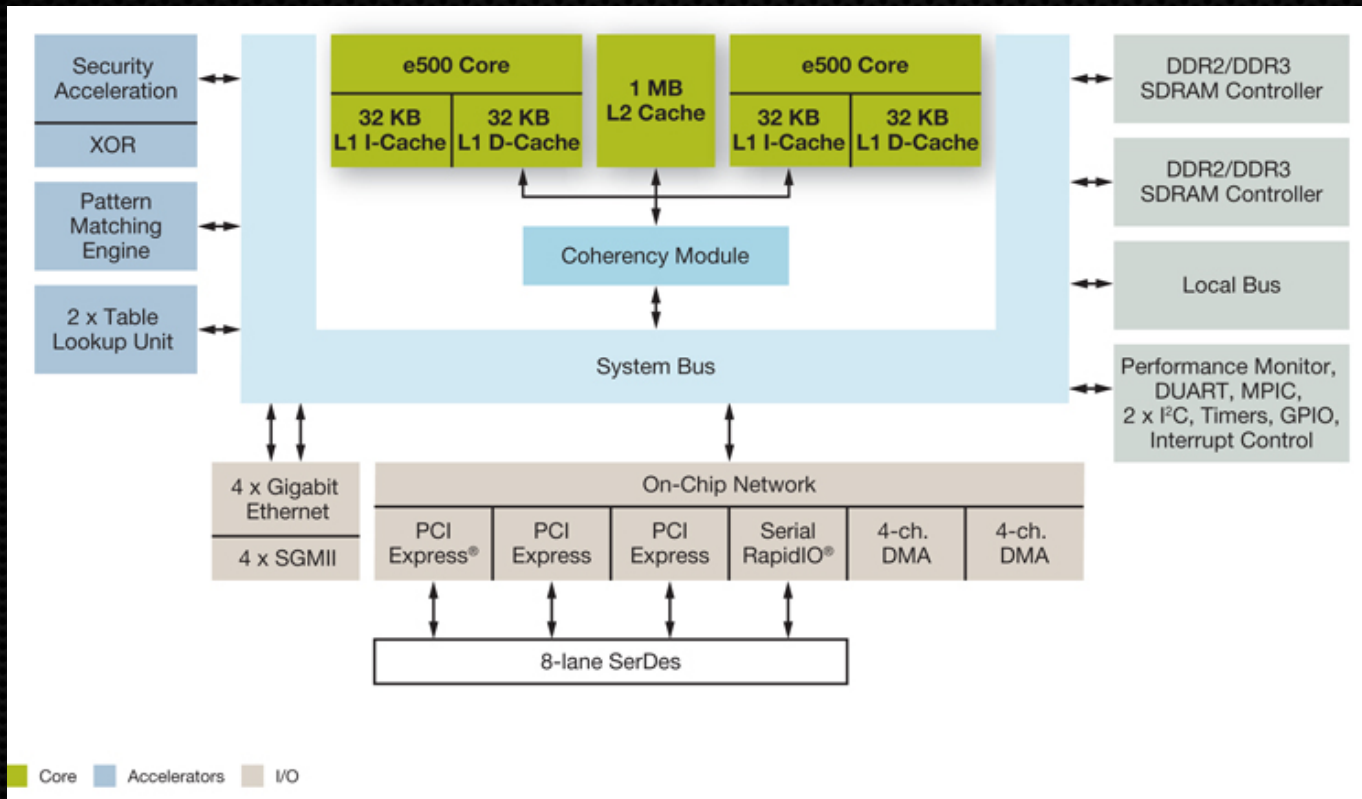
Book-E highlights

- **Flexible approach to memory management**
 - No more segmented mode, no more block translations
 - Page-based, multiple variable-sized pages
 - Pure Translation Lookaside Buffer (TLB) approach
- **Exceptions model updated**
 - New exceptions classes introduced
 - Dedicated machine instructions for handling
- **Some implementation details not imposed**

Freescal MPC8572 system

- **Based on E500 CPU core**
 - Book-E compliant core implemented by Freescal Semiconductor, Inc.
 - Dual-core
- **System-on-chip (SOC)**
 - Numerous supporting devices besides CPU cores
 - Many peripherals integrated on the same chip
 - PowerQUICC III family

MPC8572E System-on-chip



* Diagram source: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC8572E

FreeBSD/powerpc E500 port

- **MPC85xx with single-core E500 CPU**
 - Already in the FreeBSD repository
 - Support for MPC8533, MPC8541, MPC8548, MPC8555
- **Basis for the MPC8572 work**
 - Build environment
 - Bootloader support, kernel bootstrap (locore)
 - Low-level MMU layer (pmap)
 - On-chip peripherals hierarchy, selected drivers (Ethernet)

First steps of the MPC8572 port

- **Baseline code**

- FreeBSD 8-CURRENT (around March 2008)
- Rebase early, rebase often

- **Build environment**

- In-tree toolchain (gcc 4.2.1, binutils 2.15)
- Traditional PowerPC Application Binary Interface (ABI)
- PowerPC Embedded ABI (EABI) not used

FreeBSD/MPC8572 next steps

■ Bootstrap

- U-Boot firmware
- FreeBSD *loader(8)* running on top of U-Boot

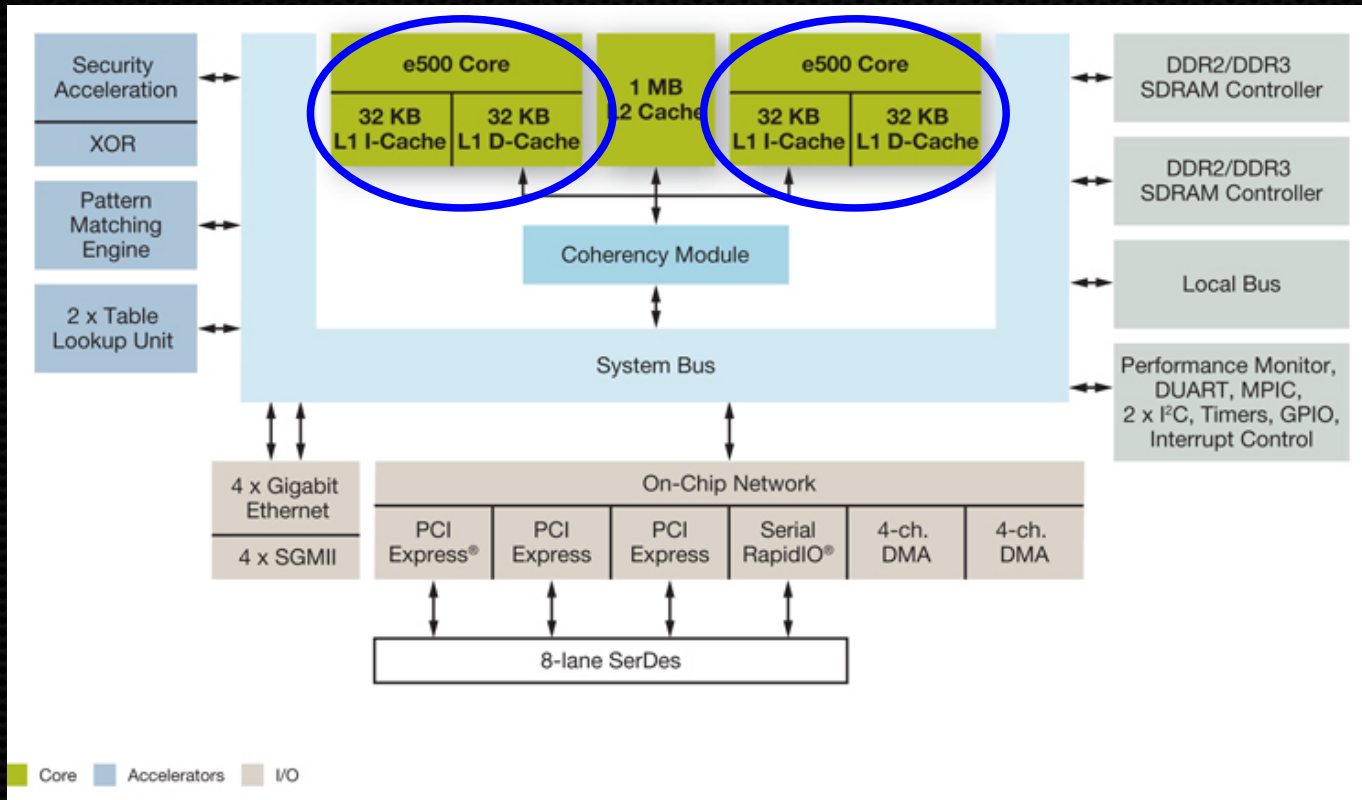
■ Minimal kernel operation

- Early E500 initialization
- Exceptions and interrupts
- Local bus operations: *bus_space(9)*
- DMA operations: *bus_dma(9)*
- *newbus* devices hierarchy

Multi-core operation bring-up

- **Multiprocessor architecture**
 - Symmetric vs. Asymmetric approach (SMP,AMP)
 - Bootstrap Processor (BSP)
 - Application Processor(s) (AP)
- **MPC8572**
 - Dual-core E500
 - Core0 (BSP), core1 (AP)
 - Core complex (CPU, MMU, L1 cache, other resources)

2x E500 core complex



MPC8572 system initialization

- **First instruction fetched from a preconfigured location**
 - Different on-reset behavior (no *reset vector* as in AIM)
- **Various options for bootstrap code storage**
 - FLASH, PCI-Express, I²C
- **Bootstrap sequence**
 - Initial and foremost responsibility of the firmware code
 - Core0 executing code, core1 inactive

The way of the bootstrap processor

- **Assumptions about the bootloader**
 - Memory starts at address 0
 - Kernel loaded at 16-MByte boundary
- **FreeBSD/MPC8572 kernel initialization outline**
 - Enable machine-specific features in CPU (hardware-implementation dependent: HID registers)
 - Initialize MMU, set up stack, initialize exceptions vector offsets
 - Jump to `e500_init()`, jump to `mi_startup()`

Book-E initialization specifics

- **MMU always on**
 - Valid TLB translations *always* required to fetch instructions or load/store data
- **Be careful during preliminary MMU clean-up**
 - Invalidate translations left by firmware
 - Kernel code being executed (including the clean-up routine) and data being accessed have to be TLB-translated *all the time!*
 - Flipping address spaces technique

BSP after machine-dependent init

- **Critical areas covered by TLB translations**
 - Kernel text, data (debug symbols), internal structures
 - SOC registers (on-chip peripherals control and status registers)
 - All other TLB resources cleared
- **Decrementer configured**
 - Time counting, DELAY() available
- **L1 and L2 caches enabled**

MPC8572 multi-core basics

- **One or more APs**
 - MPC8572: one BSP + one AP
- **CPU *holdoff* mode**
 - Prevents CPU from getting out of reset condition
 - Configurable, sampled at system reset
 - U-Boot runs on BSP (core0), leaving AP (core1) inactive
- **Boot *page* translation**

Boot page translation

- Required for fetching the 1st instruction after reset
 - E500 fetches and executes the instruction from the last word of the 32-bit address space:
 - *Effective* address `0xFFFF_FFFC`
- The default boot page translation
 - Covers the last 4-KByte page in the address space:
 - `0xFFFF_F000-0xFFFF_FFFF`
 - I:I translation (EA == PA)



■ Awakening the AP (done by the BSP)

- Adjust the boot page translation to point to AP initial code
- Let the AP run
- Note: only one boot page translation in the system (shared by all cores)

More on the AP start-up

- **Secondary processor initialization sequence**
 - Enable machine-specific features in CPU (HID registers)
 - Initialize MMU, set up stack, initialize exceptions vector offsets
 - Assign per-CPU resources and structures
 - Finalize MMU setup: `pmap_bootstrap_ap()`
 - Machine-specific SMP init `cpudep_ap_bootstrap()`
 - Call `machdep_ap_bootstrap()`, machine-independent SMP init

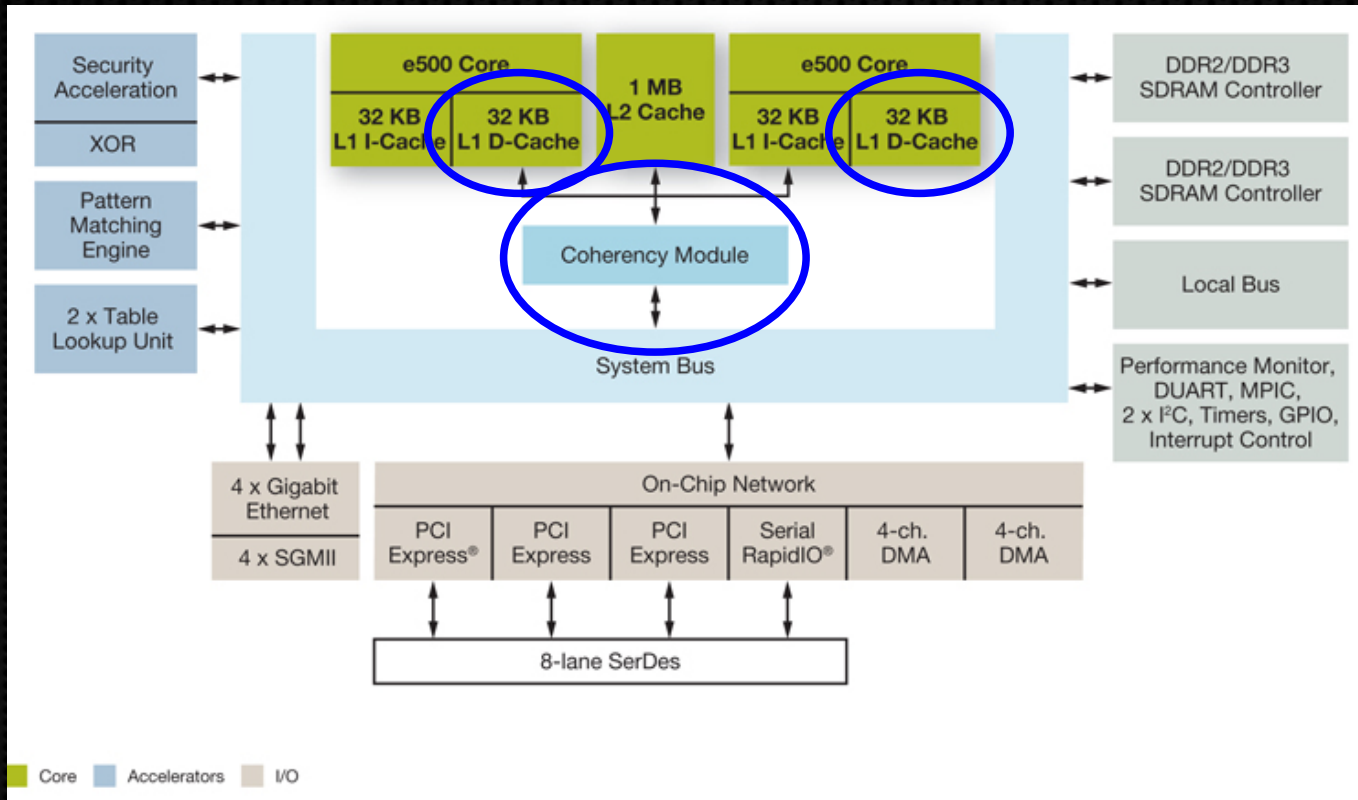
AP going „on-line”

- **TLB state in-sync with the BSP**
 - Translations for kernel and SOC integrated peripherals
- **Final steps of the AP**
 - Busy-wait for the **green light** from the BSP
 - Initialize *decrementer* and *time base* registers with BSP-provided values
 - Enable external interrupts
 - Start accepting scheduled work

E500 assistance for multiprocessing

- **Atomic operations**
 - `lwarx` / `stwcx` instructions
- **Hardware-enforced data coherence**
 - E500 Coherency Module (ECM)
 - L1, L2 cache snooping on the Core Complex Bus (CCB)
 - Other bus masters (DMA entities) hint cache logic about modifications of possibly cached locations
 - M-bit (memory coherency) among TLB page attributes
 - Invalidation (TLB, D-cache) instructions broadcast

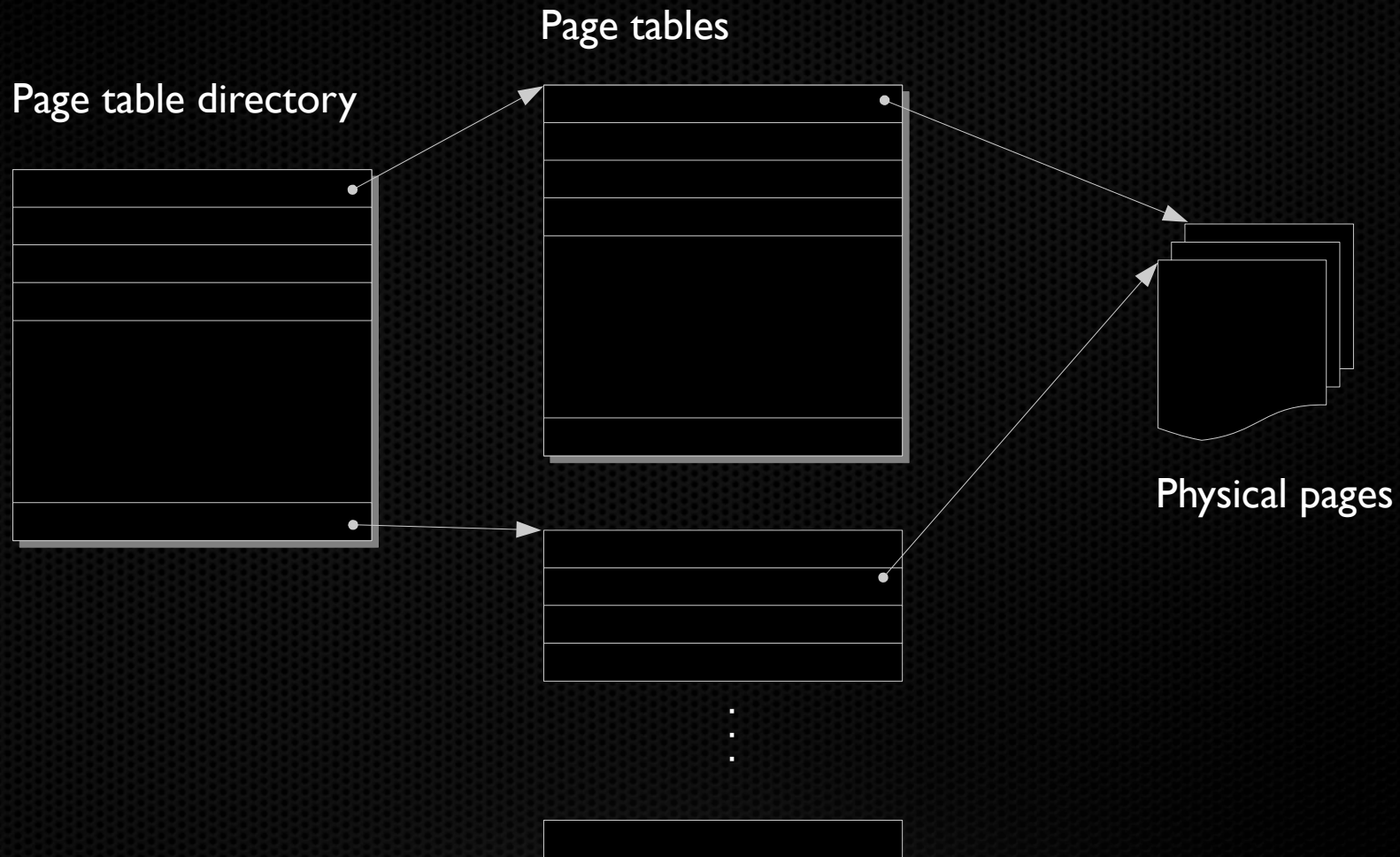
MPC8572 data coherency



Memory management

- E500-dedicated *pmap* module
- MMU hardware summary
 - Two MMU sub-units (L1 and L2); L1 handled entirely by hardware, only L2 managed by software
 - L2 unit consists of two separate TLBs
 - TLB0, set-associative, fixed 4-KByte page size, 256/512 entries; *dynamic* translations
 - TLB1, fully-associative, pages of variable size (4-KByte – 1-GByte, or 4-KByte – 4-GByte); *permanent* translations

Forward page table



E500 *pmap* challenges

- **Parallel and nested TLB miss exceptions and page faults**
 - Deadlock avoidance
- **TLB invalidations synchronization accross CPUs**
 - Only one system-wide TLB invalidation allowed at a time
- **MP-safe page tables contents update**
 - Dedicated TLB miss handling spin lock, other optimizations

MPC8572 integrated peripherals

- **Enhanced Three-Speed Ethernet Controller (eTSEC)**
 - Advanced features: polling, interrupts coalescing, VLAN tagging, h/w checksum calculation, jumbo frames
- **Pattern Matching Engine (PME)**
- **Security Engine (SEC)**
 - 3DES, AES, DES
 - MD5, SHA1, SHA256, SHA384, SHA512
 - *crypto(9)* compliant

More MPC8572 integrated peripherals

- Host/PCI-Express bridge
- Integrated DMA engine
 - General purpose DMA units
- I²C controller
- TODO
 - Table Lookup Unit (TLU)
 - eTSEC IEEE 1588 Precision Time Protocol
 - SEC support for RC4 and built-in RNG

Acknowledgements

- Alan L. Cox (The FreeBSD Project)
- Mark J. Douglas (Freescale)
- Marcel Moolenaar (The FreeBSD Project)
- Grzegorz Bernacki, Rafał Czubak, Michał Hajduk, Jan Sięka, Piotr Zięcik (all Semihalf)

Questions, please?

FreeBSD on high performance multi-core embedded PowerPC systems

Rafał Jaworowski

raj@semihalf.com, raj@FreeBSD.org

AsiaBSDCon 2009, Tokyo