November, 2010

Multicore Software Technology Roadmap

Jacques Landry
This presentation will review the multicore software strategy for the Freescale Networking and Multimedia processor roadmap

This will give you an overview of the key enablement plans for high-end and low-end processor families

The multicore software strategy in this presentation focuses on the software components that will be used by application software to drive application-level performance and capability
After completing this session you will be able to:

• Describe how the Freescale multicore software architecture enables application developers to achieve performance on Freescale multicore devices

• Name the components that make up the Freescale multicore software architecture

• Understand the roadmap for the major multicore software architecture components
Overview of the application space for networking
Multicore software programming models
High-performance multicore software architecture
Low-end multicore software architecture
Full system instrumentation for multicore
Multicore software roadmap
Networking Segments – User, Access, Core

User

Access

Core

Mobile

Business

Home

Note: Simplified diagram, not all connections/applications shown
Key Technology Trends in Networking Segments

- **Flattening of the network** – Consolidation of services into fewer devices, increased sustained bandwidth and processing rates, secure and trusted processing requirements

- **Moore’s law challenged** – Higher performance with power constraint. Advent of multicore and application-specific acceleration. Required software support.

  - Frequency scaling of CPU cores no longer valid due to power constraints

  - **Multicore processors** viewed as the most viable approach to achieve required performance gains within power budgets
Hardware Multicore Implementations

**Single Core with Hardware Accelerators**

- CPU
- Shared Bus
- Bridge
- I/O
- I/O
- Accel

- Sequential operations that cannot be multi-threaded
- Hardware acceleration provides more power/performance efficiency than software

**Homogeneous Multi/Many Core**

- With or without accelerators
- Shared or distributed memory

- CPU
- CPU
- CPU
- CPU

- Accel
- I/O
- I/O
- Accel

- Easier programming environment
- Easier migration of legacy code
- Lack of specialized hardware for differing tasks

**Heterogeneous Multi/Many Core**

- CPU
- GPU
- DSP
- CPU

- Accel
- I/O
- I/O
- FPU

- Specialized hardware for different tasks
- Most power/performance efficient
- Software complexity and portability

---

Increasing Software Complexity

---

Freescale, the Freescale logo, AltisVec, C-5, CodeTEST, CodeWarrior, ColdFire, C-Ware, mobileGT, PowerQUICC, StarCore, and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeKit, BeeStack, CoreNet, the Energy Efficient Solutions logos, Flexis, MXC, Platform in a Package, Processor Expert, QorIQ, QUICC Engine, SMARTMOS, TurboLink and VortiQa are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2010 Freescale Semiconductor, Inc.
Varied Multicore Programming Models Required

Symmetric Multiprocessing
**Single OS on all cores**
Applications can run on any core
- Common implementation in desktops

Asymmetric Multiprocessing
**Many instances of the same OS on cores**
- Common implementation in servers
  Goal: consolidate servers, increase utilization

Asymmetric Multiprocessing
**Many different OSs on dedicated cores**
- Common implementation in embedded markets
Closing the Gap with Multicore Solutions

Freescale closes “The Gap” with “Balanced Architecture”

- Smart multicore devices
- Targeted application acceleration
- Aggressive process technology
- Extensive multicore optimized software and ecosystem support

30W Ceiling

<table>
<thead>
<tr>
<th>Performance / Watt</th>
<th>1xCPU</th>
<th>NxCPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>~50x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Host Processors

PowerQUICC Processors

QorIQ Multicore Platforms

NxCPU + Acceleration
Multicore DSP / Processor Architectures

- Multiple SC3850 DSP cores
- Multi accelerator platform engine for baseband (MAPLE-B)
- High-speed, high-bandwidth CLASS / Tri-level cache hierarchy CoreNet on-chip fabric
- Dual RISC core QUICC Engine subsystem

- Multiple e500 superscalar processor cores
- Hardware virtualization support
- On-demand application acceleration
- Industry-leading performance
Networking Software Strategy

► Invest in Optimizing Freescale Platforms

• Own / control silicon-optimized software IP across all of our hardware devices and platforms
  ▪ Multi-core, PPC, DSP, accelerator

• Standalone base tools and run-time technologies

• Built around standard platforms

• Available throughout the ecosystem

► Partner for Vertical Solutions

• Complete solutions in select application spaces – VortiQa networking apps

• Leverage partners (ENEA, Mentor Graphics, Greenhills, QNX and others) elsewhere
Multicore Software Solution Model

► MC Applications:
  • VortiQa security applications
  • SMP And AMP models
  • Component model for scalability

► Com Stacks/APIs:
  • Silicon-optimized
  • Open and scalable

► Linux:
  • Control plane processing
  • SMP support

► Light Weight Executive (LWE):
  • Data path acceleration library
  • Run to completion

► BSP:
  • Silicon-optimized
  • Full featured
  • Open source

► HyperVisor:
  • Security and separation
  • Messaging among cores
  • System-level event handling
  • Debug support
DSP Application Frameworks
- Reusable system frameworks
- Robust media delivery platform
- Well-defined APIs

Components:
- Optimized for high channel densities
- WiMax, LTE L1
- Support of numerous multimedia codecs
- Video - H264, SVC
- Voice - G.729AB, G.726, G.723.1, iLBC

DSP RTOS:
- Lightweight and optimized
- Multicore enabled/support
- Real-time support

Drivers:
- Silicon-optimized
- Well-defined APIs
High End Multicore Software Architecture
QorIQ P4 Multicore Hardware Architecture
Multicore Software Architecture Models
Multicore Approaches

► Unsupervised AMP
  • Support for allowing heterogeneous operating environments to execute concurrently on different cores in an SoC
  • Cooperative approaches to sharing the resources of the device

► Supervised AMP
  • Software hypervisor to provide a virtualized environment in which guests can run
  • Spatial and (optional) temporal partitioning
  • Hypervisor manages global resources (e.g. interrupt controller) and provides services

► Lightweight Executive API
  • Provide model for programming (run-to-completion) simple AMP applications for the data path
  • Virtualized guest of the hypervisor
  • Migration to an encapsulated environment in user space of a host OS (Linux)
Unsupervised AMP
Unsupervised AMP

► General Approach
  • Manage the partitioning of the system so that multiple OSes can boot safely
  • Partition the peripherals so that they are segregated into different resource pools (each managed by a different OS)
  • Provide interpartition communication for applications and operating environments
  • Paravirtualize I/O requests so that they can be delegated to the operating environment that “owns” the device
  • Create standards for interoperability for heterogeneous OSes (boot, IPC)

► Freescale solution
  • Linux AMP environment that supports multicore booting of the Linux OS and allows a guest to be booted on secondary cores

► Ecosystem support
  • Wide support within the ecosystem for enablement of AMP solutions
  • Adoption of standards is growing (e.g. ePAPR)
  • Typically support vendor RTOS and Linux
Lightweight Executive
QorIQ P4080/4040 Multicore Programming Paradigm

- **Support a variety of customer use cases**
  - Multiple operating systems utilized across cores on a single device
  - Proprietary, third-party and open source multicore operating systems
  - Symmetric Multi-Processing (SMP) and Asymmetric Multi-Processing (AMP), often running concurrently
  - Often bare-metal, or engineered light-OS, used on forwarding/data plane cores

- **Freescale has developed a reference development platform**
  - Freescale embedded reference Hypervisor
  - Freescale boot standards, including u-boot
  - Leverage open boot protocol and API standards (e.g. Power.org™)
  - Freescale Light Weight Executive (LWE) for run-to-completion data plane processing
  - Demonstrate performance and provide reference example for customers
Light Weight Executive Concept

- Set of C-libraries needed to support data plane applications (C++ planned)

- Run-to-completion software model
  - Processes do not pre-empt each other. The process must run to completion before other processes get a chance to run, as scheduled by the QMan (= implicit work scheduler)
  - IRQs are supported, software responsibility to postpone actual processing using SWI or implement proper protection/sharing mechanism

- Device trees for LWE configuration

- Runs in supervisor state

- Dependency on Hypervisor

- Hypercalls used to access Hypervisor functionality

**Ingress Channel**

- FQ
- FQ
- FQ
- FQ

**Egress Channel**

- FQ
- FQ
- FQ
- FQ

*Function*
Lightweight Executive (LWE)

- Support *Run-to-completion* software model in Linux user space
  - LWE (data plane) tasks run as user space applications
  - No dependencies outside of Linux kernel (i.e. Hypervisor optional)
- Flexible association between software portals and LWE tasks
  - Core/thread affine for maximum performance or “floating” for scalability
- Leverage Linux facilities and standard capabilities whenever possible
  - Runtime support
  - Isolation of cores running realtime tasks
  - Manage realtime constraints

Linux

- Standardize on distribution mechanism
  - Freescale delivers optimized BSPs; partners deliver value-added distributions
  - Single commercial/non-commercial distribution of Linux for Freescale
- Freescale focus on driving Linux features that enhance system performance, enforce partitioning and satisfy realtime constraints
  - Realtime scheduling and synchronization
  - Optimization of Linux networking stack and drivers for DPAA
  - Improved latency and non-blocking operation: tickless, TLB miss reduction, etc.
  - Enforce partitioning between LWE encapsulation environments (virtualization)
Supervised AMP (Hypervisor)
Freescale’s Embedded Hypervisor

► A small hypervisor for embedded systems based on Power Architecture technology (architecture version 2.06)

► Initial version focuses on static partitioning
  • CPUs, memory and I/O devices can be divided into logical partitions
  • Partitions are isolated from one another
  • Configuration is fixed until a reconfigure and system reboot
  • Not addressing problem of multiple operating systems on 1 CPU

► Uses the Embedded Hypervisor feature in the QorIQ/e500mc which makes virtualization efficient

► Uses a combination of full-virtualization and para-virtualization which provides good performance and minimal changes to guest operating systems
**Hypervisor Contrasts**

**Freescale Hypervisor Implementation**

- **Requirement**: supervised AMP -- isolation, performance
- **Implications**: No more than one OS per core, OS has direct control of high-speed peripherals

**Traditional Hypervisor Implementation**

- **Requirement**: high level of virtualization -- solves problem of under-utilized CPUs, plus isolation
- **Implications**: more than one OS per core, complexity, performance implications

**QorIQ™ P4080 hypervisor hardware assists in meeting both requirement sets**
Hypervisor Architecture Overview

- Hypervisor partitions system “spatially” into separate domains
- Guests run in separate partitions
- Separation of domains enforced by virtualization capabilities of e500-mc core and P4080 SoC
Hypervisor Features

Operating System sees a virtual core plus Hypervisor services

- Virtual CPU (like e500mc minus hypervisor features)
- Services via hypercall
  - Interrupt controller
  - IOMMU
  - Inter-partition doorbells
  - Partition management
  - Byte-channels
  - Power management
  - Error management
  - HA failover
- Debug stub interface for debugging guest operating systems

---

[Image of a diagram showing the components of a virtual core and hypervisor services, including:
- Virtual CPU (e500vcpu)
- Services
  - Boot services (ePAPR)
  - Emulation (privileged instructions)
  - Hypercalls
  - Device tree
- Hypervisor
  - Partition Mgmt
  - Error Mgmt
  - Doorbells
- System hardware
  - UART
- Device tree
- Direct I/O]
Partition Management

Capabilities

- Copy data to/from another partition’s memory (e.g. loading OS images)
- Starting, rebooting other partitions
- Notifications—watchdog expiration, guest requests reboot, state change
- Linux `partman` command implements basic partition management features

Hypervisor

Multicore System Hardware

- Shared Cache
- Interrupt Controller
- CPU
- Memory
- I/O
- Partition Management

Linux®

RTOS

Legacy OS
Error Management

- Each partition has a guest event queue for partition-specific errors.

- A global error queue is owned by a partition designated to be an “error manager.”

- The guests implement policies specific to their needs.
Debugging

- Debug of guest operating systems is supported using hypervisor-resident debug agents.
- Transport over multiplexed serial interface.
- Code Warrior and GDB supported.
- Plug-in architecture for creating stubs.

Host

GDB

MUX server

System Hardware

CPU

Memory

UART

MUX

Stub

Partition

OS

Debugging

GDB remote serial protocol

Host

GDB

MUX server

System Hardware

CPU

Memory

UART

MUX

Stub

Partition

OS
Byte-channel—
a hypercall-based character I/O channel

Flexible endpoint configuration

- A physical UART on the QorIQ P4080
- Another byte-channel endpoint
- A byte-channel to UART multiplexer
- A hypervisor debug stub
- The hypervisor console
Hypervisor Contrasts

Freescale Hypervisor Implementation

- **Requirement**: supervised AMP -- isolation, performance
- **Implications**: No more than one OS per core, OS has direct control of high-speed peripherals

Traditional Hypervisor Implementation

- **Requirement**: high level of virtualization -- solves problem of under-utilized CPUs, plus isolation
- **Implications**: more than one OS per core, complexity, performance implications

**QorIQ™ P4080 hypervisor hardware assists in meeting both requirement sets**
A push towards consolidation has led the need to run multiple operating systems on a single processor.

A hypervisor:
- Establishes a “virtual machine” environment in which OSes run
- Enforces security
- Provides services

Varying level of OS modifications may be needed
Freescale – Virtualization Strategy

- Standards
- KVM
- Topaz
- 3rd Party ISVs
 Standards

► power.org ePAPR
(embedded Power Architecture Platform Requirements)
  • 1.0 complete in 8/2008
  • Resource discovery (device tree)
  • Multi-CPU boot
  • Updated version, including virtualization extensions targeted for Q4 2010

► power.org Embedded Virtualization Committee
  • Virtual CPU standard– the behavior of instructions and registers in a virtual machine
  • Paravirtualization & standard hcalls
Partitioning and Virtualization

Partitioning

- Hardware consolidation
- Partitioned/dedicated resources, minimal sharing.
- Dedicated CPUs, I/O devices

Virtualization

- N virtual machines
- Shared resources
- Virtual I/O
- Highly virtualized environment

Topaz

KVM
KVM (kernel virtual machine)

- Linux kernel is the hypervisor: KVM kernel module + Qemu
- Qemu provides virtual I/O services
- Allows fully virtualized platforms—can run many more virtual machines than physical resources available
- Extensive virtual I/O support
- Established open source community
- Targets: e500v2, e500mc
- GNU Public License
Freescale Embedded Hypervisor (Topaz)

- An embedded hypervisor designed for Power architecture from the ground up
- Requires CPU with Power ISA 2.06 virtualization extensions
- Partitioning focus—secure partitioning of the hardware resources of an SOC and board
- No scheduler
- Hypervisor is minimally intrusive
- A moderate set of services—interrupt controller, inter-partition interrupts, byte-channels, power management, active/standby/failover, error management
- Targets: e500mc
- BSD License
Virtual Machines Under KVM and Topaz

Architectural goal is to provide compatible virtual machines environments under both KVM and Topaz

- No guest OS modifications required
- Compliant with base ePAPR (device trees)
- ISA 2.0.6.1 virtual CPU
- ePAPR para-virtualization extensions
Virtualization Roadmap

► Q4 2010
  • KVM (e500v2): performance evaluation
  • KVM (e500mc): initial port

► Q1 2011
  • KVM (e500v2/e500mc): release with basic feature set, minimal direct assignment of memory/devices
  • Topaz: SDK 2.3

► Q2 2011
  • KVM: direct device assignment, hugetlbfs, performance, extended vcpu features, MMU performance improvements
  • e500mc/P4080 KVM: virtual network I/O via P4080 datapath,
  • Topaz: performance, 64-bit, targeted feature development

► 2H 2011 (and beyond)
  • KVM: SMP, error management, failover, power management, P4080 portal context switching
  • e500mc KVM: 64-bit support
  • Topaz: Processor roadmap support, targeted feature development
Multicore Software Development Kit (SDK)
P4080 SDK Architecture Today

**Linux User Space**
- PME Compiler
- FM Config Script
- Other Linux Apps
- Partition Manager
- LWE/Apps Image
- LWE CP Apps

**LWE Applications**
- IPFwd
- Pktwire
- PME
- IPSec
- Bridging
- QM tester
- Crypto
- FM Tester

**Linux Kernel Space**
- PME Driver
- BM Driver
- Legacy Drivers
- QM Driver
- FM Config Driver
- SEC Driver
- Ethernet Driver

**LWE**
- Mem Mgt
- BM Driver
- PME Driver
- Atomic Calls
- SEC Driver
- Statistics
- Timer
- Inter Process Communication

**Hypervisor**
- Virtual CPU
- Interrupt controller
- Error Mgmt
- Boot services
- IPI
- U-Boot
- GNU Tools
- Secure Boot
- Integration / Packaging
- Power Mgmt
- Partition Mgmt
- IOMMU
- Byte-channels
- Guest debugging
Linux User Space QorIQ DPAA Architecture Example

USDPAAP Application Can Use 1 to 8 Cores
Each thread has a dedicated portal and is affined to a core,
1 thread per core

Core 0 has portal for kernel use and standard Linux networking

7 cores are isolated but 1 can run an USDPAAP thread as well as other processes.
Future High-end Multicore Software Architecture

Linux User Space
- PME Tools
- DPAA Tools
- Std commands/libs
- pthreads
- stats/state access
- perfmon control
- FM Enhanced CfgDriver
- System Configuration and Control

Linux Kernel
- FM Basic Cfg Driver
- DPAA Ethernet Driver
- perfmon
- QM Driver
- BM Driver
- SEC Driver
- PME Driver
- scheduler control
- hugetlbfs
- UIO Drivers for LWE
- Legacy Drivers

Hypervisor
- Virtual CPU
- Interrupt controller
- Error Mgmt
- Boot services
- IPI
- IOMMU
- Byte-channels
- Guest debugging
- Power Mgmt
- Partition Mgmt
- U-Boot
- GNU Tools
- Secure Boot
- MG System Builder

LWE
- Mem Mgt
- SEC Driver
- Initialization
- BM Driver
- PME Driver
- Statistics
- QM Driver
- Atomic Calls
- Timer
Low End Multicore Software Architecture
All flows are created equal …

… But some flows can be put on a fast-track.

Store flows requiring simple, deterministic processing in a cache.

Recognize cached flows and process such packets in a separate highly optimized context – Fast-Path.
Application Specific Fast-Path

► Advantage:
  • 2x to 5x advantage over standard Linux
  • 2-core scaling of > 1.8x
  • Leverages hardware acceleration features effectively
  • Compatible with hardware fast-path IP (QE)

► Current scope (under definition):
  • Feature: IPv4, NAPT, firewall, IPSec
  • Platforms: P1020, P1021, P2020, P1022, P1010
  • Single fast-path for both Linux BSP and VortiQa

► Roll-out strategy
  • Q3-10: Demo release
  • Q4-10: Reference solution
  • Q1-Q4-11: IPv6, QoS, GTP
Full System Instrumentation
NSD Software and Enablement Technologies

- Advanced software development tools
- Full application visibility/control

- Silicon-optimized software components
- Scalable robust software architectures

- Compiler friendly cores
- Advanced debug IP
Proﬁling, Debug, and Instrumentation Libraries

- Instant access to debug hardware resources (no exposure to the specifics of hardware debug complex)
- Re-use your debug and proﬁle solutions (seamless porting on QorIQ processors)

Access hardware debug resources from target

- Applications running on target may conﬁgure and control hardware debug resources using API
- At least two scenarios may be implemented
  - User applications may be instrumented to directly call API to get access to hardware resources
  - Operating system may implement its debug services (e.g. proﬁle, events, trace) by calling API

Access hardware debug resources from host

- Traditional debug and proﬁle tools running on host may use the library to
  - Apply conﬁgurations on target
  - Control the selected resources
  - Retrieve proﬁle/trace data from target
- The classic Debug Agent required by the host tools only needs to call API to get access to the required debug services
Multicore Software Roadmap
## Multicore Software Roadmap

### 2010

<table>
<thead>
<tr>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sept</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SDK Beta 2.0**
- P4080 silicon support
- DS Expedition board
- Linux + LWE guests

**Beta 2.1.1 SDK**
- Full testing on initial Rev 2 silicon samples
- Linux Ethernet jumbo frame support
- Critical bug fixes

**Hypervisor**
- Error management services
- High availability/Fault tolerant HV infrastructure – notifications and shared device management
- 64-bit support

**SDK 2.2:**
- 36 bit addressing
- 10 GB XAUI dual mode support
- Offline port support enabled in FMD
- Congestion group demo
- 32 bit support for P5020
- 32 bit support for P3041

**Distributions:**
- P4080DS SDK 2.2 (Open Source)

---

**Distribution Tools**
- System Builder

**Linux**
- DPAA to user space
- 64-bit kernel (P5020)
- Rapid IO driver
- HugeTLBfs

**Multicore**
- Fast Path
- Driver optimization

**SDK QorIQ**
- OpenEmbedded support
- UserSpace DPAA (Alpha)
- Huge TLB fs support
- Initial 64 bit kernel for P5020
- RAID for P5020
- SATA for P5020/3041

---

Freescale, the Freescale logo, Altiview, CodeTEST, CodeWarrior, ColdFire, C-Ware, mobileGT, PowerQUICC, StarCore, and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeKit, BeeStack, CoreNet, the Energy Efficient Solutions logo, Flexis, MXC, Platform in a Package, Processor Expert, QorIQ, QUICC Engine, SMARTMOS, TurboLink and VortiQa are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2010 Freescale Semiconductor, Inc.
Freescale is focused on developing high-performance, full enablement multicore software components

Freescale’s multicore software strategy supports various customer application programming models

Freescale’s multicore software strategy supports both high performance and low cost multicore devices