June 25, 2007

Power Architecture™ Primer
Deep Dive into the ISA

Gary Whisenhunt
Power Architecture Principal Architect
Contents

► Power Architecture™ Technology and History
  • About Power Architecture technology
  • ISA history / future directions
► Power ISA Overview
  • Format of architecture
  • Categories
  • Key architectural elements
  • User instruction set architecture
► Freescale Implementations of Power ISA™
  • Freescale Embedded Implementation Standards (EIS)
  • Embedded instruction set features
  • New Embedded Features in Power ISA™ 2.03
  • SIMD numeric acceleration
  • Variable Length Encoding
► Freescale Cores
  • Core Families
  • Licensable Cores
  • System-on-a-Chip
  • Roadmap
Power Architecture™ Technology

Consistency of Architecture
User code compatible
Extensible for market specific enhancements

Performance
Roadmap
MHz and multi-core

Product Range
Microcontroller to Enterprise
Many variants from Freescale alone

Breadth of Capabilities
Single chip MCUs to highly integrated SoCs to high-performance CPUs

Breadth of Applications
Computing, Networking, Gaming, Telecom, Automotive, Storage, Printing & Imaging, Military Industrial Automation

Customer Endorsement
Large installed base

3rd Party Support
Largest selection of third party support

Freescale/IBM
AMCC/Xilinx/PA Semi
Strong competition to ensure leading edge products
Power Architecture™ Technology History/Future

Freescale/IBM
PowerPC™
Book E 1.9
APUs

IBM
PowerPC™
2.02 + Proposals
Book 1
Book 2
Book 3

Merged ISA from Power.org PAAC
Power ISA™ Version 2.03
Common User Mode (VLE, 32b, 64b)
Separate Privilege Mode (server, embedded)

Power ISA™ 2.04/2.05
Power ISA™ 2.06

2006 2007 2008+
Power Architecture™ Technology History

► The Power ISA 2.03 version is referred to as the *merged* architecture because it brings together:
  • The base functionality of PowerPC™ 1.10
  • Embedded features defined by PowerPC Book E and Freescale APUs
  • The PowerPC 2.02 architecture defined by IBM. Aspects of the Power ISA server category are not discussed in detail here.

► Power ISA is defined through proposals from the Power Architecture Advisory Council (PAAC) within the Power.org consortium

► Power ISA version 2.03 was published in September 2006

► Power ISA version 2.04 was published in May 2007
  • Download at: www.power.org/resources/downloads/PowerISA_Public.pdf
Power Architecture™ Technology Future

- Versions 2.04/2.05 contain mostly server oriented enhancements
- Version 2.06 is slated to bring partitioning and virtualization technology to the embedded environment.
  - Hypervisor/partitioning/virtualization available to embedded implementations
  - Timeframe is sometime in 2008
Power ISA Overview
<table>
<thead>
<tr>
<th>Books and Contents</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory reference, flow control, Integer, Floating Point, Numeric acceleration, Application-level programming model</td>
<td>Application-level programming model, generally compiler generated</td>
</tr>
<tr>
<td>Time, synchronization, cache management, and storage features, such as byte ordering</td>
<td>Library-level, non privileged programming model, generally assembler generated</td>
</tr>
<tr>
<td>Exceptions, interrupts, memory management, debug facilities, and special control instructions and resources</td>
<td>Supervisor/Hypervisor-level instructions and resources</td>
</tr>
</tbody>
</table>
Power ISA Modularity/Scalability – Categories

- Power ISA architecture is divided into categories.
- Every component of the architecture is defined as part of a category.
- Processors implement a set of these categories.
- Classes of processors are required to implement certain categories.
  - For example, server class processors implement categories Server, Base, Floating Point, 64-bit, etc.
- All processors implement the Base category

**Base**
Most of Book I and Book II, except numeric acceleration and some specialized instructions

**Server**
Generally Book III-S

**Embedded**
Generally Book III-E

**Problem specific categories:**
Floating Point, Vector, Signal Processing Engine (SPE), Cache Locking, …
Power ISA Modularity/Scalability – Categories

► Categories allow implementations to be tailored for application domains spanning a broad spectrum
  • Addresses the disparate needs of server and embedded devices
  • Supports niche-specific extensions
    ▪ Categories extend the PowerPC™ Book E concept of auxiliary processing units (APUs) to include all features in the architecture

► The Base category (as well as others) preserve binary compatibility, enabling easier software migration across generations of implementations

► Some categories are hierarchical
  • Dependent categories cannot be implemented without the category on which it depends—e.g., the Floating Point.Record category is dependent on the Floating Point category.
Power ISA Modularity/Scalability – Categories

- Partial list of categories for Power ISA:
  - Base
  - Server, Embedded
  - Floating Point, Floating Point Record
  - Vector
  - Signal Processing Engine
    - SPE.Embedded Float Scalar Single
    - SPE.Embedded Float Scalar Double
    - SPE.Embedded Float Vector
  - 64-bit, Variable Length Encoding
  - Embedded Cache Locking, Embedded.Enhanced Debug
  - Stream, Trace, Alternate Time Base, Move Assist
  - Embedded.External PID, External Proxy, Embedded.Processor Control
  - Embedded.MMU Type FSL, Wait
Auxiliary Processing Units (APUs)
- AltiVec™ technology (128-bit wide SIMD tailored for multimedia)
- SPE (64-bit wide SIMD tailored for signal processing)
- Embedded Floating Point (saturating, low cost floating point operations)
- Isel (conditional register move reducing branch mispredicts)
Freescale APUs to Categories (Book II)

► APUs

- Alternate Time Base (timer facility similar to time base, usually incrementing at core frequency)
APUs
- Cache Line Locking (lock lines in cache for enhanced performance)
- Enhanced Debug (additional interrupt level for debug)
- Performance Monitor (record performance related data)
- Machine Check (additional interrupt level for machine check)
Freescale APUs to Categories (Book VLE)

APUs
- Variable Length Encoding (denser encoding of some Power ISA™ instructions, useful for applications that require better code density)
Power ISA Key Architectural Elements

► Power ISA is a RISC load/store architecture
► Multiple register sets accentuate performance
  • 32 General Purpose Registers (GPRs) for integer ops (32-bit or 64-bit)
  • 32 Floating Point Registers (FPRs) for floating point ops (64-bit)
  • 32 Vector Registers (VRs) for vector ops (128-bit)
  • 8 Condition Register Fields (CRs) for comparison and flow control (4-bit)
► Special Registers (SPRs)
  • Counter Register (CTR) provides efficient loop control
  • Link Register (LR) provides efficient subroutine linkage
  • Time Base (TBU, TBL) and Alternate Time Base (ATBU, ATBL)
  • Accumulator (ACC) provides accumulation operations
  • Status registers (XER, FPSCR, VSCR, SPEFSCR) for providing control
    and status for various computational operations
► Most instructions are triadic, with 2 source operands and one
  destination allowing for better register allocation by compilers
### User Mode Registers

#### SPE Category

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GPR0 hi</td>
</tr>
<tr>
<td>31</td>
<td>GPR 0</td>
</tr>
<tr>
<td>32</td>
<td>GPR0 hi</td>
</tr>
<tr>
<td>63</td>
<td>GPR 0</td>
</tr>
<tr>
<td>63</td>
<td>GPR 1</td>
</tr>
<tr>
<td>63</td>
<td>GPR 31</td>
</tr>
<tr>
<td>0</td>
<td>ACC</td>
</tr>
<tr>
<td>32</td>
<td>SPEFSCR</td>
</tr>
</tbody>
</table>

#### Floating Point Category

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FPR0</td>
</tr>
<tr>
<td>63</td>
<td>FPR 0</td>
</tr>
<tr>
<td>63</td>
<td>FPR1</td>
</tr>
<tr>
<td>63</td>
<td>FPR 1</td>
</tr>
<tr>
<td>0</td>
<td>FPR31</td>
</tr>
<tr>
<td>32</td>
<td>FPR31</td>
</tr>
<tr>
<td>63</td>
<td>FPSCR</td>
</tr>
</tbody>
</table>

#### Vector Category

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>VR0</td>
</tr>
<tr>
<td>127</td>
<td>VR 0</td>
</tr>
<tr>
<td>0</td>
<td>VR1</td>
</tr>
<tr>
<td>127</td>
<td>VR 1</td>
</tr>
<tr>
<td>127</td>
<td>VR 31</td>
</tr>
<tr>
<td>127</td>
<td>VR 31</td>
</tr>
<tr>
<td>63</td>
<td>VRSAVE</td>
</tr>
</tbody>
</table>

#### Base Category

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GPR 0</td>
</tr>
<tr>
<td>63</td>
<td>GPR 0</td>
</tr>
<tr>
<td>0</td>
<td>GPR 1</td>
</tr>
<tr>
<td>63</td>
<td>GPR 1</td>
</tr>
<tr>
<td>0</td>
<td>GPR 31</td>
</tr>
<tr>
<td>127</td>
<td>GPR 31</td>
</tr>
<tr>
<td>63</td>
<td>CR</td>
</tr>
<tr>
<td>63</td>
<td>CTR</td>
</tr>
<tr>
<td>0</td>
<td>LR</td>
</tr>
<tr>
<td>63</td>
<td>LR</td>
</tr>
<tr>
<td>0</td>
<td>XER</td>
</tr>
<tr>
<td>63</td>
<td>XER</td>
</tr>
<tr>
<td>0</td>
<td>TBL</td>
</tr>
<tr>
<td>31</td>
<td>TBL</td>
</tr>
<tr>
<td>0</td>
<td>TBU</td>
</tr>
<tr>
<td>31</td>
<td>TBU</td>
</tr>
</tbody>
</table>

#### Embedded Performance Monitor Category

13 PMRs

#### Alternate Time Base Category

ATB (ATBU)
Power ISA Key Architectural Elements

► Uniform-length instructions with the exception of the VLE category
  • VLE provides variable length instructions for applications with extreme code density requirements (for example, cost sensitive devices with embedded flash such as automotive engine control MCUs)
► A precise exception model
► Single and double precision floating point IEEE-754 with additional multiply-add instructions
► Vector SIMD operations on integer and floating point data types providing operations on up to 16 elements in a single instruction
Power ISA Key Architectural Elements

- Support for Harvard instruction and data caches as well as unified caches
- Cache operations for block zeroing as well as cache streaming hints
- Memory operations are strictly load/store
- Support for both big- and little-endian addressing, with separate categories for moded (server) and per-page (embedded) endianness
- Support for 64-bit addressing and computation (64-bit category)
Power ISA Key Architectural Elements

► Flexible memory model allowing weakly-ordered memory accesses for enhanced performance
  • Loads and stores may be performed out of order
  • From the initiating processor’s perspective, loads and stores occur in program order
  • Memory barrier instructions \([\text{sync (msync), eieio, mbar}]\) are provided to enforce ordering across multiple processors and system devices -- for example:

\[
\begin{align*}
stw & \quad r4,0,r5 \\
sync & \\
lwz & \quad r6,0,r7
\end{align*}
\]

// ensures stw & lwz occur in-order

• Usually only required for multiprocessing code and device drivers
Power ISA Key Architectural Elements

► Privilege modes
  • User, Supervisor, Hypervisor (category Server)

► Hypervisor provides virtualization capabilities through partitioning:
  • Processors and hardware threads
  • Memory
  • Interrupts
  • Devices

User privilege

Supervisor privilege

Hypervisor privilege (category Server)

Hypervisor

partition
Minor Modifications from Book E

Some minor changes from Book E are in Power ISA:

► The Book E–defined \textit{msync} instruction has reverted to being the Synchronize instruction \textit{sync} defined by the PowerPC™ Architecture. The \textit{msync} simplified mnemonic is defined for Book E compatibility.
  • \textit{msync} and \textit{sync} are the same binary representation and processors defined under Book E will execute \textit{sync} correctly.

► The names of some architecturally defined fields and descriptions of instructions have changed to make the merged architecture more consistent.
  • The architectural semantics from these still remain the same. For example, the \textit{dcbt} instruction now describes the first field as TH instead of CT.
Freescale Implementations of Power ISA
Freescale Embedded Implementation Standards (EIS)

► The Freescale Embedded Implementation Standards (EIS):
  • Define a specific architectural behavior in places where the architecture may define things generally
  • Define several implementation dependent features that are consistent across Freescale processors
► Much of the Freescale EIS defined under Book E has been merged into Power ISA, but the EIS still contains Freescale-specific features, such as L1 and L2 cache management functionality and HID registers, because they do not make sense to architect globally
► The EIS will still be used to provide similar Freescale-specific extensions to Power ISA.
► Implementations may have their own extensions that are not part of either Power ISA or the EIS, almost exclusively in supervisor mode.
Freescale EIS and Categories

- EIS cores include the following categories:
  - Base
  - Embedded
  - Embedded.MMU Type FSL
  - Embedded.Enhanced Debug
  - Embedded.Cache Locking
  - Embedded.Cache Specification
  - Embedded.Little Endian

- Future EIS core designs will also include:
  - Alternate Time Base
  - Embedded.External PID
  - Embedded.Processor Control
  - External Proxy
  - Wait
  - Embedded.Performance Monitor
Freescale EIS and Categories

Depending on target market segment, EIS cores may optionally include any of these categories:

- Signal Processing Engine
- SPE.Embedded Float * (single, double, or vector)
- Floating Point, Floating Point.Record
- Vector
- Variable Length Encoding
Freescale’s Implementations

► All Freescale Power Architecture™ cores are 32-bit integer user-mode binary compatible in the Base category
  • Load/store RISC with 32 GPRs, CTR, LR, 8-field CR,…
  • SMP behaviors architected from the start
► Power ISA extends basics for embedded
  • Simplified MMU (software page table friendly, multiple page sizes)
  • Cache control (line locking)
  • Domain-specific acceleration
    ▪ Vector: graphics and performance-oriented 128-bit SIMD
    ▪ SPE: area-efficient 2-way DSP-oriented SIMD
  • Domain-specific acceleration. More coming in future releases
User Accessible Registers

- General-use registers are user registers accessed either as source/destination operands, explicit instructions (e.g., mtcr), or as side effects of another operation (an integer overflow would set XER[OV])
- 32 general-purpose registers (GPRs) – 32 or 64 bits (32-bit embedded SPE categories use 64-bit GPRs)
- Former PowerPC™ floating point registers now part of Floating Point category
  - 32 64-bit floating point registers FPRs
  - Floating Point Status and Control Register (FPSCR)
- 32 128-bit vector registers (VRs), defined by Vector category
- Vector Control and Status Register (VSCR)
- Link Register (LR)
- Counter Register (CTR)
- Condition Register (CR) (eight 4-bit fields)
- Integer Exception Register (XER)
- SPE Floating Point Status and Control Register (SPEFSCR) – part of the Signal Processing Engine category
Supervisor and General Control Registers

► Processor is generally controlled through the Machine State Register (MSR)
  • Problem state (privilege states)
  • Interrupt enables
  • Instruction enables (Floating Point, SPE, Vector)
  • Address space (used to easily separate user and supervisor address spaces)
  • Performance Monitor control

► Processor Identification Register (PIR) – used to identify a processor in a multiprocessing or multithreaded environment
  • PIR will be writeable in future Freescale cores

► Processor Version Register (PVR) – identifies the type and revision of the processor
Exceptions and Interrupts

► Interrupts
  • Action where processor saves current context and begins execution at predetermined interrupt handler

► Exceptions
  • Events which cause the processor to take an interrupt
    ▪ Synchronous
    ▪ Asynchronous

► Exception syndromes
  • The ‘syndrome’ is the reason for the exception
    ▪ Reported in the Exception Syndrome Register (ESR)
Exceptions and Interrupts

► Reset vector
  • PowerPC™ 1.10 cores: real mode – offset 0x0 0100
  • Power ISA embedded category: virtual mode with hardware initialized TLB entry

► Interrupts
  • PowerPC 1.10 cores: fixed vector offsets
  • Power ISA embedded category: vector is a concatenation of vector prefix/vector offset
  • Power ISA Book III-E defines additional interrupts not in PowerPC 1.10: critical interrupt, fixed-interval timer, DTLB/ITLB errors
Exception and Interrupt Registers

► Intermits
  - Interrupt save and restore registers (SRR0/1, CSRR0/1, DSRR0/1, MCSRR0/1) for interrupt state saving and return
    - 4 levels of interrupts are provided
      - Machine Check (generally recoverable if software can fix cause)
      - Debug
      - Critical
      - Non Critical
    - Most interrupts are non-critical
  - Interrupt Vector Prefix Register (IVPR)
  - Interrupt Vector Offset Registers (IVORs)
    - Define where specific interrupts vector to
  - Exception Syndrome Register (ESR) – contains status about type of exception
Several different timer facilities

- Decrementer with auto-reload capability
- Time base running at a selectable fraction of core or system clock or from an external source
- Fixed interval timer driven from selectable bit transition in the Time Base
- Watchdog timer (generates a critical interrupt or performs a reset)
- Alternate time base is an additional incrementing time base, usually counting at core frequency

Timers may be frozen on debug event
Embedded Debug Capabilities

- Debug interrupt is a critical interrupt as in embedded category (Adds the rfc1 instruction)
- Embedded. Enhanced Debug category defines the debug interrupt as a separate interrupt type (Adds the rfd1 instruction)
  - Future Freescale cores will support this
- Supports debug with hardware assist
  - Breakpoint registers, for data accesses and instructions
  - Instruction complete, interrupt taken, and branch taken events facilitate tracing
- Debug is controlled by enabling debug events in the Debug Control Registers (DBCR0/1). When a debug event occurs it is posted in the Debug Status Register (DBSR), and a debug interrupt occurs.
- Freescale processors also contain an external debug mode with more capabilities allowing cores to be controlled externally
Debug Registers

- Debug Status Register (DBSR) – status for debug events
- Debug Control Registers (DBCRn) – control for debug events
- Instruction Address Compare Registers (IACn) – for setting breakpoints on instruction execution
- Data Address Compare Registers (DACn) – for setting breakpoints on data accesses
Memory Management Functions

► Address Translation
  • Implements virtual memory
  • Each process can have its own unique address space ($2^{32}$ for 32-bit implementations or $2^{64}$ for 64-bit implementations)
  • Dynamic management of memory

► Protection and memory control
  • Protection distinguishes between supervisor and user accesses
  • Read, write, execute permissions
  • Page attributes
    ▪ Cache inhibited, write-through, guarded (speculative access protection)
    ▪ Page size
    ▪ Page types (normal, VLE, endianess)
## Evolution of the PowerPC™ to Power ISA Embedded MMU

<table>
<thead>
<tr>
<th>PowerPC 1.10 MMU</th>
<th>Power ISA 2.03 Embedded Book III-E MMU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Segmented virtual address space</strong> 16 segment registers.</td>
<td><strong>Unsegmented virtual address space</strong> No segment registers</td>
</tr>
<tr>
<td><strong>Hardware managed TLB</strong> using hashed reverse page tables</td>
<td><strong>Software managed TLB</strong> H/W assist for TLB replacement No required page table format</td>
</tr>
<tr>
<td><strong>Fixed 4KB sized pages</strong> Variable sized translation via BATs</td>
<td><strong>Fixed and variable size pages</strong> supported</td>
</tr>
<tr>
<td>Separate instruction and data side TLB</td>
<td>Unified instruction and data TLB</td>
</tr>
<tr>
<td><strong>Real mode</strong> (translation off) Virtual mode (translation on)</td>
<td><strong>No real mode</strong> (real mode can be emulated)</td>
</tr>
</tbody>
</table>
Embedded Address Translation

- **As** [PID] Effective Page Address Offset

  virtual address

  TLB -multiple entry – (real page number [RPN])

  Real Page Number Offset

  64-bit real address

**MSR_{DS}** for data accesses
**MSR_{IS}** for instruction access
64-bit effective address
Memory Management Registers

Memory Management

- MMU Configuration Register (MMUCFG) – describes capabilities of MMU such as number of TLB arrays, real address bits, etc.
- MMU Control and Status Register (MMUCSR0) – basic MMU controls such as TLB invalidate-all and TLB page size
- TLB Configuration Registers (TLBnCFG) – describes capabilities of TLB entries such as min and max page size, invalidation protect, number of entries, associatively
- MMU Assist Registers (MASn) – used by software to read and write TLB entries
- Process ID Registers (PIDn) – used by software as an address space identifier to match TLB entries
External PID Load/Store

- Load/store access to external address space and contexts
  - New feature in Power ISA 2.03. Category: Embedded.External PID
  - Allows loads and stores to be performed in another context from the current executing context
  - The load context can be different than the store context

- New supervisor instructions to perform external loads, stores and cache management operations
  - The EPLC register defines the load context
  - The EPSC register defines the store context
  - The external context includes the PID value, the AS (MSR[DS]) value and the privilege (MSR[PR])
  - Indexed forms only
  - Instruction forms for doing Floating Point, Vector and SPE load/store

- All instructions are translated in the current context only
Normal Address Translation

**Effective Page Address**

- MSR$_{DS}$ for data accesses
- MSR$_{IS}$ for instruction access

**Real Page Number**

- Multiple entry – (real page number [RPN])

- 64-bit real address

**TLB**

- 64-bit effective address

- Virtual address

**AS**

- PID

- Offset
External PID Load/Store Address Translation

AS  PID  Effective Page Address  Offset

virtual address

TLB
-multiple entry –
(real page number [RPN])

Real Page Number  Offset

64-bit real address

EPLC_{DS} for load accesses
EPSC_{DS} for store access

EPLC_{PID} for load accesses
EPSC_{PID} for store access

64-bit effective address
External PID Load/Store Uses

► Useful for kernel `copyin()` and `copyout()` functions
  • True address space separation with high performance
► Kernel address space can be completely independent of user process address spaces
  • Allows user process address spaces to be $2^{64}$ (64-bit implementation) or $2^{32}$ (32-bit implementation)
► Easily validate user addresses passed to the kernel
► Provide a fast method of copying data from one context to another
  • Setup EPLC to be the copy-from context
  • Setup EPSC to be the copy-to context
Interprocessor Messaging

- Topology independent lightweight messages
  - New feature in Power ISA 2.03. Category: Processor Control
  - Works for virtual processors (threads) as well
  - Multi-core feature
- Sends a “doorbell” interrupt to another [virtual] processor(s)
  - `msgsnd` and `msgclr` instructions
Msgsnd & Msgclr Instructions

► The `msgsnd` instruction broadcasts a message to all devices in the coherence domain
  • All [virtual] processors receiving a message filter it based on:
    ▪ Value of the PIR (Processor Identification Register) or
    ▪ Always accept if the message was tagged as a broadcast
  • There are 2 message types:
    ▪ Doorbell (causes a non-critical interrupt – SRR0/1 if accepted)
    ▪ Doorbell Critical (causes a critical interrupt – CSRR0/1 if accepted)

► Software can build higher order interprocessor messaging with `msgsnd` as a notification mechanism

► Software can write the PIR to any value to identify a particular [virtual] processor

► The `msgclr` instruction clears any message accepted by the [virtual] processor that have not yet taken the associated interrupt
  • `msgclr` only clears messages that are pending on the [virtual] processor that executes the `msgclr` instruction
Wait Instruction

- Wait for an interrupt
  - New feature in Power ISA 2.03. Category: Wait
- The *wait* instruction waits until an interrupt occurs
  - Instruction fetching is stopped
  - Any prefetched instructions are discarded
  - When an interrupt occurs, the instruction is complete and the save/restore registers point to the instruction following the *wait*
  - *wait* is a user level instruction
- Executing *wait* is a power saving feature and is generally equivalent to “Doze”
- An idle process might be implemented as:

```
loop: wait
   bu    loop
```
External PID Load and Store

► Operating systems need access to all virtual address spaces, including address spaces from user level processes
SIMD Numeric Acceleration

SIMD (Single Instruction Multiple Data) instructions perform a single operation on multiple data elements (e):

![Diagram of SIMD operation]

Input Register

Output Register
Power ISA SIMD Numeric Acceleration

► Vector (AltiVec™) A 128-bit wide SIMD attacks parallel data-oriented compute application
  • 32, 128 bit vector registers
  • 16 x 8, 8 x 16, 4 x 32 bit integer operations per clock
  • 4 x 32 bit IEEE floating point operations per clock
  • Powerful ‘permute’ unit (splits, shifts, rotates)

► Signal Processing Engine (SPE), area-efficient 64-bit SIMD
  • 2-wide operations on 16- and 32-bit integers and fractions
  • Embedded FP subcategories: Single, Double, and Vector (2 Single)
  • Integer/fraction ops include saturating arithmetic
  • Extends 32 bit GPRs to 64 bits
  • Attacks DSP applications
    ▪ Automotive powertrain: knock detect, signal conditioning, combustion modeling; uses FIR’s, FFTs, Kalman filters
    ▪ VoIP: convolutions, correlation, FIRs, excitation functions
AltiVec™ Technology (category: Vector)

- Concurrency with integer and floating-point units
  - Separate, dedicated 32 128-bit vector registers
    - Larger namespace = reduced register pressure/spillage
    - Longer vector length = more data-level parallelism
    - Separate registers can all be accessed by execution units in parallel
    - Deep register file allows more sophisticated software optimizations

- No penalty for mingling integer, floating point and vector operations
128-bit Vector Architecture

► Offers SIMD parallel execution
  • 16-way parallelism for 8-bit signed and unsigned integers and characters
  • 8-way parallelism for 16-bit signed and unsigned integers
  • 4-way parallelism for 32-bit signed and unsigned integers and IEEE floating point numbers
► Permute unit – full crossbar switch reorders 16 bytes of data from any two source registers to a destination register in a single cycle
► 4 parallel functional units
► Available in e600
Signal Processing Engine (SPE)

► Embedded-friendly, SIMD integer & fractional vector instructions
  • Can implement at lower cost than AltiVec™
► 64-bit, two-element operands using extended GPRs (regardless of 32- or 64-bit implementation)
► Defines own double-word load/store instructions (also used with Embedded Float Double and Embedded Vector floating point)
► Defines SPE/floating-point status/control register (SPEFSCR) and accumulator (ACC)
  • ACC used for loop accumulations allowing back to back accumulation without regard to latency
► Available in the e200/e500
GPRs are 64 bits, even though e500/e200 are 32-bit implementations

- Non category SPE instructions modify only the lower 32 bits leaving the upper 32 bits unchanged
Embedded Floating Point

► Single-precision scalar and vector
► Double-precision scalar
► No FPRs: DP and vector SP use 64-bit extended GPRs. Scalar SP uses lower word only in 32-bit implementations (as do non-SPE instructions)
► Computation is saturating unless exceptions are enabled
► IEEE compliance requires software handlers for exceptions to handle boundary conditions
► Found in the e200/e500
Increased Code Density

► Cost is a key factor in some embedded applications
  • On chip memory required to store software code is inherently system cost
  • Traditionally, RISC architectures have poorer code density than CISC architectures

► Book VLE (Variable Length Encoding) provides significant improvement in code density
  • Re-encoding of Power ISA instructions from fixed 32-bit instructions into a mixture of 16- and 32-bit instructions
  • 30% or greater code footprint reduction
  • e200 family cores will have best-in-class code density
VLE Technology Overview

► VLE is a instruction re-encoding into 16-bit and 32-bit instructions, which may be freely intermixed
  • 16-bit instructions use one of the registers as both a source and destination
  • VLE and non VLE instructions are marked by page – no modes
  • VLE pages may be mixed with ordinary pages
  • VLE and base category code is fully inter-callable with ordinary code
  • VLE is defined as a separate Book in the Power ISA architecture

► VLE instructions are 16-bit aligned

► Primary opcode 31 and opcode 4 have the same encodings as in non VLE instructions
Freescale Cores
Freescale Power Architecture™ Solutions

► Microcontrollers
  • For automotive, consumer, industrial applications and multi-core products
  • e200 and e300 cores

► PowerQUICC™ I, II and III Families
  • SOHO, enterprise, wired and wireless infrastructure + storage, industrial, print/imaging and multi-core products
  • e300 and e500 cores

► High-Performance Processors
  • Networking + home media, commercial robotics, blade servers, printers, gaming and multi-core products
  • e600 core
### Freescale Core Overview

<table>
<thead>
<tr>
<th>Core Type</th>
<th>ISA / MHz</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>e600 Core</strong></td>
<td>PowerPC ISA 600 MHz – 1.8 GHz</td>
<td><strong>e600 Platforms</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPC86xx host processors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPC74xx host processors</td>
</tr>
<tr>
<td><strong>e500 Core</strong></td>
<td>Power ISA 533 MHz – 1.5 GHz</td>
<td><strong>e500 Platforms</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PowerQUICC III MPC85xx</td>
</tr>
<tr>
<td><strong>e500-mc Core</strong></td>
<td>Power ISA 533 MHz – 1.5 GHz</td>
<td><strong>e500-mc Platforms</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEW Multi-core Platform</td>
</tr>
<tr>
<td><strong>e300 Core</strong></td>
<td>PowerPC ISA 266 MHz – 667 MHz</td>
<td><strong>e300 Platforms</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PowerQUICC II Pro MPC83xx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PowerQUICC II MPC82xx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPC52xx microcontrollers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPC51xx microcontrollers</td>
</tr>
<tr>
<td><strong>e200 Core</strong></td>
<td>Power ISA 80 MHz – 475 MHz</td>
<td><strong>e200 Platforms</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPC55xx auto microcontrollers</td>
</tr>
<tr>
<td><strong>z0</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>z1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>z3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>z6</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview
- 32 bit Power ISA Architecture Embedded Category Processor Core
- VLE
- SPE
- Low power
- Fully synthesizable
- Unified 32KB 8-way set-associative cache
- 32-entry unified MMU
- AMBA AHB Bus interface

Features
- Single-issue, in-order, 7-stage pipeline
- Most instructions Single-cycle Execution
  - Integer Multiply 3 clocks, fully pipelined
  - Integer Divide 6-16 clocks, unpipelined
  - Floating Multiply 3 clocks pipelined
  - 3-cycle loads
  - 1-3-cycle branches
Overview
- 32 bit base Power Architecture™ Processor Core based on the PowerPC™ 1.10 Architecture
- Refinement of original 603 for Apple laptops

Features
- Dual issue, 4-stage pipeline, OOO execution
- L1 Cache - 32KB I, 32KB D 8 way set-associative, Parity
- 32/64b Processor Local Bus
- Most instructions Single-cycle Execution

Characteristics
- Technology: Retargetable 130nm, 90nm
- Frequency: 533MHz in 90nm bulk process
Overview:
• 32 bit Power ISA Architecture Embedded Category Processor Core
• Optimized for Embedded Applications
  ▪ Multi-core capable
  ▪ SoC interface

Features:
• 2-way superscalar, 7-stage pipeline design, OOO Execution
• 32K I-cache and 32k D-cache
• Low-latency, low-overhead, deterministic interrupt context switches
• Current Core-Connect-Bus interface provides multiple data busses (tagged, OOO, split transaction) targeted for low latency on-chip interface to platform-IP
• Future CoreNet interface for enhanced multicore scalability

Characteristics
• Technology: 90nm SOI
• Frequency: 1333MHz
e600 Core Overview

► Overview:
  • 32 bit Power Architecture™ Processor Core based on the PowerPC™ Architecture
  • Gigahertz+ Superscalar High Performance
  • High performance embedded computing for Aerospace & Defense, Industrial, Networking and Telecom
  • Multi-core capable

► Features:
  • 3-way + branch, superscalar, 7-stage pipeline design, OOO Execution
  • 11 execution units:
    3 simple fixed-point units, Complex fixed-point unit, Floating-point unit, 4 AltiVec™ units, Branch execution unit, Load/store unit
  • 32K I-cache and 32k D-cache + backside L2
  • Low-latency, low-overhead, deterministic interrupt context switches
  • Core-Connect-Bus interface provides multiple data busses targeted for low latency on-chip interface to platform-IP

► Characteristics
  • Technology: 90nmSOI
  • Frequency: 833MHz - 1.7Ghz
Through IPextreme, Freescale is licensing e200 Power Architecture™ core IP to the general marketplace

- Announced in April 2007

4 Versions of the e200 core will be available:
- All cores are fully synthesizable

<table>
<thead>
<tr>
<th>e200z0</th>
<th>e200z1</th>
<th>e200z3</th>
<th>e200z6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Power ISA v2.03 (VLE only)</td>
<td>Power ISA v2.03</td>
<td>Power ISA v2.03</td>
</tr>
<tr>
<td>Frequency</td>
<td>150 MHz</td>
<td>150 MHz</td>
<td>150 MHz</td>
</tr>
<tr>
<td>Pipeline Depth</td>
<td>4-stage</td>
<td>4-stage</td>
<td>4-stage</td>
</tr>
<tr>
<td>Debug Unit</td>
<td>Nexus 2+</td>
<td>Nexus 1</td>
<td>Nexus 3</td>
</tr>
<tr>
<td>Bus Interface</td>
<td>AMBA 2.0v6</td>
<td>AMBA 2.0v6</td>
<td>AMBA 2.0v6</td>
</tr>
<tr>
<td>MMU</td>
<td>8-entry MMU</td>
<td>16-entry MMU</td>
<td>32-entry MMU</td>
</tr>
<tr>
<td>SIMD</td>
<td>FPU</td>
<td>SPE</td>
<td></td>
</tr>
<tr>
<td>L1 Cache</td>
<td>Up to 32K</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Core frequency based on 90nm process technology estimates
Scalable Performance, Connectivity & Integration

Meeting a wide spectrum of processing and I/O requirements

Scalable Performance.

- e200
- e300
- e500
- e600

Scalable Connectivity.

- Network Acceleration
  - Host Peripheral Bus
  - General Peripherals
  - Protocols
  - Security

- System Fabric
- Storage
- Networking

Customer IP

- PCI
- PCI-X
- PCMCIA
- 60x, MPX
- Local Bus
- RapidIO
- SERDES
- USB 2.0
- USB 1.1
- I²C, DUART
- UTOPIA
- GMII, MII, RGMII, TBI
- XAUI

Customer Logic Integration

- PLLs/DLLs
- D/A, A/D
- Clock Data Recovery
- TIA, LDD
- DDR SDRAM
- DDR II SDRAM
- SDRAM
- Flash, EPROM
- SRAM, DRAM

Integration Structures

- Copper Line
- Magenta/SkyBlue Line
- OCeaN Integrated Fabric
- Coherent System Bus

System Integration

- Bus Controllers
- Timers
- DMA
- Arbiters
- Bridges
- Interrupt Controllers
- GPIO

Mixed Signal

- Interconnect Network Acceleration

Memory Controllers

- System Integration Structures
- Integration Structures
- System Integration Structures
- Scalable Integration Structures

Customer IP

- Ethernet (10/100/1000)
- Serial (Tx/Rx)
- HDLC
- T1/E1/T3/E3 TDM
- ATM (AAL0,1,2,5,IMA)
- IP ML/MC-PPP PPPmux
- BPON

Encryption
- Authentication
- Public Key
- RNG

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2007.
MPC5510 Automotive SoC

Key Features

► e200z1 and optional e200z0 Power Architecture cores
► 0.13u technology
► High performance through parallelism in the architecture
► Hardware and software compatibility to maximize reuse
► Flexible flash partitioning removes the need for external EEPROM chip
► Nexus2+ debug capability, saving development time
PowerQUICC III MPC8572E Networking SoC

Key Features

- e500 dual Power Architecture core (up to 1.5GHz)
- Application Acceleration
  - Table lookup
  - Security (SEC 3.0)
  - Pattern matching (RegEx) & deflate
Conclusions

► One architecture scaling from very low to very high
  • Supports converging technologies
  • Enables collaboration across industries
  • Opens doors for innovation

► PAAC (within Power.org) manages ISA openly
  • Members consist of IBM and Freescale
  • Building new levels of extensibility and compatibility throughout the microprocessor developer community

► Freescale continues Power Architecture™ leadership with participation in Power.org and strong product roadmap with multi-core platforms