IBM zNext –
The 3rd Generation High Frequency Microprocessor Chip

Chung-Lung (Kevin) Shum
Senior Technical Staff Member, System z Processor Development, Systems & Technology Group, IBM Corp.
zNext PU Chip Overview

- IBM mainframe microprocessor chip for the next generation of System z servers

- 32nm SOI technology
  - 597 mm² (23.7mm x 25.2mm)
  - 15 layers of metal, 7.68 miles of wire
  - 2.75 Billion transistors
  - I/Os: 10000+ Power, 1071 Signal
    - SMP connections to external Hub chip (SC)
    - I/O Bus Controller (GX)
    - Memory Controller (MC) with prefetching

- Processor Core Features
  - 2\textsuperscript{nd} Generation out-of-order design
  - Speed & feed improvements
  - Microarchitecture innovations
  - Architecture extensions for software exploitations, e.g.,
    - Hardware Transactional Memory
    - Runtime instrumentation

- Chip Features (vs. z196)
  - 6 new cores per chip (vs. 4)
  - Core-Dedicated (vs. shared) Co-Processors
  - 48 MB EDRAM on-chip shared L3 (2x)
Speed: Higher Operating Frequency

- **z900** – Full 64-bit z/Architecture
- **z990** – Superscalar CISC pipeline
- **z9** – System level scaling
- **z10** – Deep Pipeline, Arch. extensions
- **z196** – Out-Of-Order (OOO), Additional Architectural Extensions
- **zNext** – OOO+, Architectural Extensions, Enablement for new Software Paradigms
Speed: Higher Operating Frequency

- **Technology + Design Optimizations**
  - Provides higher performance & capacity
  - Maintains unmatched reliability
  - Supports Peak workload 24x7
  - At similar power constraints

- **z900** – Full 64-bit z/Architecture
- **z990** – Superscalar CISC pipeline
- **z9** – System level scaling
- **z10** – Deep Pipeline, Arch. extensions
- **z196** – Out-Of-Order (OOO), Additional Architectural Extensions
- **zNext** – OOO+, Architectural Extensions, Enablement for new Software Paradigms
Feed Improvements: Maximizing Out-of-Order Window

- Improved dispatch grouping efficiencies
  - More instructions per group
  - Reduced cracked instructions overhead
  - Increased branches per group to 2
  - Added Instruction Queue (InsnQ) for re-clumping

*clumps – parcel of instructions delivered from instruction fetching*
Feed Improvements: Maximizing Out-of-Order Window

- Increased out-of-order resources
  - More out-of-order groups
    - Multi-grouped instructions speculative completion
    - Increased Global Completion Table (GCT) entries to 30x3 (+25%)
    - Increased usable physical GR entries to 80 (+25%)
    - Increased physical FR entries to 64 (+33%)
Feed Improvements: Maximizing Out-of-Order Window

- Increased execution bandwidth
  - More instructions issued per cycle
  - Added Virtual Branch Queue (VBQ) for relative branch queuing
  - Added Virtual Branch Unit (VBU) for relative branch execution
  - Increased effective issue queue size to 32x2 (+60%)
  - Increased issue bandwidth per cycle to 7 (+40%)
z196 Pipeline (recap)

Diagram based on Brian Curran’s HOTCHIP 22 presentation

© 2012 IBM Corporation
Feed Improvements: Accelerating Specific Functions

- Short-circuit executions
  - Common idioms executed during dispatch
  - e.g. initializing a GR with zeros

- D-Cache (SRAM design) with banking support
  - 32 banks for concurrent 2 read and 1 write operations
  - Faster cache writes reduce future load-use delays

- Dedicated Fixed-point divide engine resulting in 25-65% faster operations

- Millicode (Vertical Microcode) operations
  - Selective hardware execution
    - Translate, Translate and Test, Store Clock
  - Shorter startup latency
    - Move Character variations, Co-Processor operations
  - Hardware assists for prefetching (target cache level & coherency state)
    - Move Character Long variations
  - Dedicated hardware for Unicode conversion (UTF8 <> UTF16)
Micro-Architecture Innovations: Branch Prediction

- Branch Prediction is essential in improving performance
  - 2nd level BTB (BTB2) for capacity (more than 3x)
  - Fast re-Indexing Table (FIT) for latency (up to 33% reduction)

### Micro-Architecture Innovations: Branch Prediction

<table>
<thead>
<tr>
<th>Micro-Architecture Innovation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTBP</td>
<td>Branch Target Pre-buffer 0.5th level Branch IA and target predictor look-up in parallel to BTB1, upon usage, transfer to BTB1</td>
</tr>
<tr>
<td>BTB1</td>
<td>L1 Branch Target Buffer 1st level Branch IA and target predictor</td>
</tr>
<tr>
<td>BTB2</td>
<td>L2 Branch Target Buffer 2nd level Branch IA and target history buffer</td>
</tr>
<tr>
<td>FIT</td>
<td>Fast re-Indexing Table Index histories for recently used BTB1 entries predictions throughput: every 2 cycles (vs. &gt;= 3 cycles) power down some structures when in use</td>
</tr>
<tr>
<td>BHT</td>
<td>Branch History Table Direction predictor (2-bit)</td>
</tr>
<tr>
<td>SBHT</td>
<td>Speculative BHT &amp; PHT Speculative direction prediction updates at resolution time prior to completion</td>
</tr>
<tr>
<td>PHT</td>
<td>Pattern History Table          Pattern based tagged direction prediction</td>
</tr>
<tr>
<td>CTB</td>
<td>Changing Target Buffer         Pattern based target prediction</td>
</tr>
<tr>
<td>SGO</td>
<td>Surprise Guess Override        2nd level direction 1-bit predictor</td>
</tr>
</tbody>
</table>

Diagram & Table based on Eric Schwarz's 2011 VAIL presentation
Micro-Architecture Innovations: Cache Subsystem

- Split Level 2 Cache (instead of unified)
  - 1M-byte Instruction, 1M-byte Data
  - Inclusive of instruction-L1 (64 Kbyte) and data-L1 (96 Kbyte)
  - Bigger aggregate L2 with shorter latency

- Integrated data-L2 directory
  - data-L2 directory is merged into data-L1 directory
  - Logically indexed like in data-L1 directory
  - L2 Hit / miss knowledge at L1 miss time
  ➔ L1 miss, L2 hit latency reduced by up to 45%

- Store “Gathering” Cache
  - circular queue of 64 entries of half-lines (128 bytes)
  - merges stores to same half-line post L1 updates
  - reduces pipeline usage for stores in L2 and L3
  - Hardware Transactions storage updates
  ➔ Store traffic to L3 typically reduced by ~50%

Modeling Data provided by Jim Mitchell @ IBM Poughkeepsie
Targeted Architectural Extensions

- 2 Gigabyte Page support

- Decimal Floating Point Extension
  - Instructions to convert numeric data between 2 formats:
    - zoned fixed-point decimal, and
    - decimal floating point

- Instruction Processing Directives
  - Branch preload instructions
    Specifies the address of a branch instruction and its target to be installed into branch prediction tables (through BTBP)
  - Data access intent instruction
    Specifies what operands of the next instruction may be further accessed for
    - e.g. getting a cache line exclusive on a load for future store
    - e.g. keeping access-once line at current Least-Recently-Used (LRU) position
Architectural Differentiation Extension: Transactional Execution

- **General Purpose Multiprocessor Support**
  - Instructions specifying start, end, and abort of a transaction
  - Pending storage updates are “shielded” from other processors until transaction completes
  - Implemented at heart of CPU (core+L1) for performance
  - Heavy focus on support for software usage and debug
  - “Constrained Transaction” with hardware auto-retries for code simplification

- **Prototype benchmark with HTM**
  - Showed ~2x improvements and better scalability (slope)

Prototype Data provided by Jerry Zheng, Marcel Mitran @ IBM Toronto
Summary: zNext will…..

- Be used in a new family of IBM System z mainframe servers

- Sustain IBM’s mainframe leadership in computing capacity and performance without sacrificing any reliability, with
  - Up to 6 active cores per chip
  - 48M-byte shared on-chip L3 cache
  - uniquely designed low-latency private L2 cache
  - >24K target and >32k direction branch histories
  - Numerous micro-architectural enhancements*

- Provide architecture extensions*, and be the 1\textsuperscript{st} general purpose microprocessor to support
  - hardware transactional memory
  - software self-directed run-time profiling

- Be amongst the fastest microprocessors @ 5.5 GHz
  - joining z196 @ continuous clock-speed of >5 GHz

* Not all features and extensions described in this presentation
AMD RADEON™ HD 7970 WITH GRAPHICS CORE NEXT (GCN) ARCHITECTURE

Mike Mantor, AMD Senior Fellow
michael.mantor@amd.com
August 28, 2012
**AMD RADEON™ HD 7970 ARCHITECTURE**

- **Graphic Core Next (GCN)**
  - 4.3 billion 28nm transistors
AMD RADEON™ HD 7970 ARCHITECTURE

Graphic Core Next (GCN)

- Advanced Power Management
  - Fine grain clock\clock tree gating
  - Power Tune – Dynamic V/F Scaling with power containment
  - Zero Core Power – Power Gating
AMD RADEON™ HD 7970 ARCHITECTURE

Graphic Core Next (GCN)

- 32 Compute Units (CU)
- Non VLIW ISA
- Distributed Control Flow
- 32/64b IEEE-2008 FP
- Integer, Logic & Video Ops
- 4 Texture Units per CU
**AMD RADEON™ HD 7970 ARCHITECTURE**

**Graphic Core Next (GCN)**
- 384-bit GDDR5 - 264GB/Sec
- Unified R/W Cache Hierarchy
  - 768KB R/W L2 Cache
  - 16KB R/W L1 Per CU
  - 16KB Instruction Cache(I$)/4CU
  - 32KB Scalar Data Cache(K$)/4CU
AMD RADEON™ HD 7970 ARCHITECTURE

Graphic Core Next (GCN)
- PCI Express® Gen 3.0 x16
Graphic Core Next (GCN)

- Global Data Share – 64 kb Shared Memory with global synchronization resources (Barriers, Append, ordered append and named semaphores resources)
Graphic Core Next (GCN)

- Dual Geometry Engines
- Dual Rasterizers
- 8 Render Back-ends
  - 32 Pixel Color Raster Operation Pipelines (ROPs)
  - 128 Depth Test (Z)/stencil Ops
  - Color Cache (C$)
  - Depth Cache (Z$)
**GCN | HotChips 2012**

**Graphic Core Next (GCN)**

- Dual Asynchronous Compute Engines (ACE) and Dual DMA
- Compute ECC protection (DRAM & SRAM (Registers, Shared Memories, L1 & L2 Caches))
- GPU support for Compute APIs OpenCL™ 1.2, DirectCompute, C++ AMP
Multi-Media and Display System
Universal Video Decoder (UVD)
Fixed Function with codecs for:
- H.264
- VC-1
- MPEG-2 (SD & HD)
- MVC (Blu-ray HD)
- DivX®
- WMV MFT
- WMV native
Multi-Media and Display System

Video Codec Engine (Fixed Function)
- Multi-stream hardware H.264 HD Encoder
- Power efficient & faster than real-time 1080p @60fps
- Two encode modes: full fixed & hybrid (with GPU compute)
Multi-Media and Display System
CrossFire™ Compositor
- Controller for Multi-GPU Solutions
- Dual, triple or quad-GPU scaling
**GCN DISCRETE GPU FAMILY**

**“Tahiti” HD 79XX**
- 4.3b transistors
- 352 SQMM
- 925e Mhz, 3.78 Tflop
- 32 CU / 32 Pix / 2 Tri
- 384b, 5.5gbps, 264 GB/S

**“Pitcairn” HD 78XX**
- 2.8b transistors
- 212 SQMM
- 1e Ghz, 2.56 Tflop
- 20 CU / 32 Pix / 2 Tri
- 256b, 4.8gbps, 154 GB/S

**“Verde” HD 77XX**
- 1.5b transistors
- 123 SQMM
- 1e Ghz, 1.28 Tflop
- 10 CU / 16 Pix / 1 Tri
- 128b, 4.5gbps, 72 GB/S
GCN COMPUTE UNIT

- Basic GPU building block of unified shader system
  - New instruction set architecture
    - Non-VLIW
    - Vector unit + scalar co-processor
    - Distributed programmable scheduler
    - Unstructured flow control, function calls, recursion, Exception Support
    - Un-Typed, Typed, and Image Memory operations
- Each compute unit can execute instructions from multiple kernels simultaneously
- Designed for programming simplicity, high utilization, high throughput, with multi-tasking
GCN COMPUTE UNIT (CU) ARCHITECTURE

Input Data: PC/State/Vector Register/Scalar Register

LOCAL DATA SHARED MEMORY ARCHITECTURE

- 64 kbyte, 32 bank Shared Memory
  - Direct mode – Interpolation @ rate or 1 broadcast read 32/16/8 bit
  - Index Mode – 64 dwords per 2 clks - Service 2 waves per 4 clks

- Advantages
  - Low Latency and Bandwidth amplifier for lower power
  - Software managed cache
  - Software consistency/coherency - thread group via Hardware barrier
Previous AMD GPUs used VLIW (Very Long Instruction Word) architecture
– Combines instructions into a 4-wide VLIW that gets executed on a SIMD

Shader Instructions

\[
\begin{align*}
    a &= b + c; \\
    b &= c + d; \\
    c &= d + e; \\
    d &= e + f;
\end{align*}
\]
Previous AMD GPUs used VLIW (Very Long Instruction Word) architecture
– Combines instructions into a 4-wide VLIW that gets executed on a SIMD

PREVIOUS VLIW SHADER ARCHITECTURE

Shader Instructions

With Dependencies

a = b + c;
b = a + d;
c = b + e;
d = c + f;

VLIW Instruction

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>b + c</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>b + c</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>b + c</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>b + c</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>b + c</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
</tbody>
</table>

Thread 0
Thread 1
Thread 2
Thread n
Previous AMD GPUs used VLIW (Very Long Instruction Word) architecture – Combines instructions into a 4-wide VLIW that gets executed on a SIMD.
Previous AMD GPUs used VLIW (Very Long Instruction Word) architecture:
- Combines instructions into a 4-wide VLIW that gets executed on a SIMD

Shader Instructions

With Dependencies

\[
\begin{align*}
  a &= b + c; \\
  b &= a + d; \\
  c &= b + e; \\
  d &= c + f;
\end{align*}
\]

VLIW Instruction

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b + e)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>(b + e)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>(b + e)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>(b + e)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
</tbody>
</table>

Thread 0
Thread 1
Thread 2
Thread n
**PREVIOUS VLIW SHADER ARCHITECTURE**

- Previous AMD GPUs used VLIW (Very Long Instruction Word) architecture
  - Combines instructions into a 4-wide VLIW that gets executed on a SIMD

Shader Instructions  
With Dependencies

\[
\begin{align*}
  a &= b + c; \\
  b &= a + d; \\
  c &= b + e; \\
  d &= c + f;
\end{align*}
\]

<table>
<thead>
<tr>
<th>Shader Instructions</th>
<th>VLIW Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a = b + c;)</td>
<td>(c + f)</td>
</tr>
<tr>
<td>(b = a + d;)</td>
<td>idle</td>
</tr>
<tr>
<td>(c = b + e;)</td>
<td>idle</td>
</tr>
<tr>
<td>(d = c + f;)</td>
<td>idle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c + f)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>(c + f)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>(c + f)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
<tr>
<td>(c + f)</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
</tr>
</tbody>
</table>

Thread 0
Thread 1
Thread 2
Thread n
NEW NON-VLIW SHADER ARCHITECTURE

- SIMD architecture without VLIW instructions
  - No need to combine instructions, since multiple threads can run in parallel

Shader Instructions

With or without Dependencies

\[
\begin{align*}
  a &= b + c; \\
  b &= a + d; \\
  c &= b + e; \\
  d &= c + f;
\end{align*}
\]

\[
\begin{align*}
  S_0 & : b + c \\
  S_1 & : a + d \\
  S_2 & : b + e \\
  S_n & : c + f
\end{align*}
\]

ALUs

No idle ALUs due to no dependencies!
**IS VLIW A GOOD LONG TERM SOLUTION?**

**VLIW4 SIMD**
- 64 Single Precision multiply-add
- 1 VLIW Instruction × 4 ALU ops → dependency limited
- Compiler manages register port conflicts
- Specialized, complex compiler scheduling
- Difficult assembly creation, analysis, and debug
- Complicated tool chain support
- Careful optimization required for peak performance

**GCN Quad SIMD**
- 64 Single Precision multiply-add
- 4 SIMDs × 1 ALU op → occupancy limited
- No register port conflicts
- Standardized compiler scheduling & optimizations
- Simplified assembly creation, analysis, and debug
- Simplified tool chain development and support
- Stable and predictable performance

VLIW packing sometimes requires domain transformation to achieve good utilization.
float fn0(float a, float b) {
    if (a > b) {
        return ((a - b) * a);
    } else {
        return ((b - a) * b);
    }
}

// Registers r0 contains “a”, r1 contains “b”
// Value is returned in r2

v_cmp_gt_f32 r0, r1 // a > b, establish VCC
s_mov_b64 s0, exec // Save current exec mask
s_and_b64 exec, vcc, exec // Do “if”
v_sub_f32 r2, r0, r1 // result = a - b
v_mul_f32 r2, r2, r0 // result = result
s_andn2_b64 exec, s0, exec // Do “else” (s0 & !exec)
v_sub_f32 r2, r1, r0 // result = b - a
v_mul_f32 r2, r2, r1 // result = result * b
s_mov_b64 exec, s0 // Restore exec mask

- Generally straightforward to generate and understand ISA
- VCC - Vector condition code
- EXEC – Execution mask
- Multi-threaded enables full vector unit utilization
GCN SCALAR/VECTOR COMPUTE UNIT

- Simpler ISA compared to previous generation
  - No clauses and latency for transitions
  - No VLIW packing required
  - Control flow directly programmed (Exec mask control)
  - Complex Control Flow Supported (Example: non uniform Branch into loop)

- Scalar engine
  - Lower latency for distributed sequencer versus previous centralized
  - Reduces performance in previously clause bound cases
  - Reduces power handling of control flow Ops as control is closer

- Advanced language feature support
  - Exception support
  - Function calls
  - Recursion

- Enhanced extended ALU operations
  - Media ops
  - Integer ops
  - Integer atomic operations
  - Floating point atomics (min, max, cmpxchg)

- Enhanced debug support
  - HW functionality to improve debug support
**R/W CACHE HIERARCHY**

16KB instruction cache (I$) + 32 KB scalar data cache (K$) shared per 4 CUs with L2 backing

Each CU has 256kb registers and 64kb local data share

L1 read/write 16kb write through caches 64 Bytes / CU / clock

L2 read/write cache partitions (64kb/128kb) write back caches 64 Bytes / partition / clock

Global data share facilitates synchronization between CUs
GPU MEMORY MODEL

- Relaxed memory model
  - All work-items within same work groups see same L1 cache
  - Work-items of different work groups may use different L1 caches
  - All work-items and command streams use the same L2 cache
  - Command stream packets & Shader Instruction control data visibility
  - Sufficient primitives in the GPU hardware to implement C++ 11 memory model

- GPU Coherency
  - Acquire/Release semantics control data visibility across the machine (Compiler controlled bit on load/store instructions)
  - L2 coherent ➔ all CUs & CP can have the same view of memory

- Remote Global atomics
  - Performed in L2 cache
  - Full set of integer ops and float max, min, cmp_swap
AMD GCN CU ARCHITECTURE SUMMARY

- Heavily multi-threaded CU architected for throughput
  - Efficiently balanced for graphics and general compute
  - Simplified coding for performance, debug and analysis
  - Simplified machine view for tool chain development
  - Low latency flexible control flow operations
  - Read/Write Cache Hierarchy improves I/O characteristics
  - Flexible vector load, store, and remote atomic operations
  - Load acquire/store release consistency controls
float fn0(float a, float b) {
    if (a > b)
        return ((a - b) * a);
    else
        return ((b - a) * b);
}

// Registers r0 contains “a”, r1 contains “b”
// Value is returned in r2

v_cmp_gt_f32 r0, r1 // a > b, establish VCC
s_mov_b64 s0, exec // Save current exec mask
s_and_b64 exec, vcc, exec // Do “if”

s_cbranch_vccz label0 // Branch if all lanes fail
v_sub_f32 r2, r0, r1 // result = a - b
v_mul_f32 r2, r2, r0 // result = result * a

label0:

s_andn2_b64 exec, s0, exec // Do “else”(s0 & !exec)

s_cbranch_execz label1 // Branch if all lanes fail
v_sub_f32 r2, r1, r0 // result = b - a
v_mul_f32 r2, r2, r1 // result = result * b

label1:

s_mov_b64 exec, s0 // Restore exec mask

Optional:
Use based on the number of instruction in conditional section.
- Executed in branch unit

- Generally straightforward to generate and understand ISA
- Instructions types interleave within program

- Throughput optimized for vector instructions
- Optional scalar instructions jump fully predicated groups of instructions
X-Gene™: 64-bit ARM CPU and SoC

Paramesh Gopi
Gaurav Singh
Greg Favor

8.29.2012
Cloud Computing Technology Trends

- **High Density Servers → “Sea of CPUs”**
  - Smaller & power-efficient CPUs; Beefier memory & IO subsystems
  - Distributed Fabric → networking & storage IO sharing & virtualization

- **Server-on-Chip (SoC) Approach**
  - Integrated NIC and IO chipset
  - CPU/ GPU combination for HPC applications

- **Active Power Management**
  - Firmware based optimization based on user Workload (Power is measured through TDP)
  - Maximize performance while managing TDP

- **Server Standardization**
  - Service provider specified
  - ODM designed & manufactured
  - Open Source/ non-commercial SW base
  - Open Stack, Open Compute
Cloud Servers - Typical Form Factors

<table>
<thead>
<tr>
<th>Public Cloud</th>
<th>Building Out vs. Scaling Out</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applications</strong></td>
<td>TODAY: 2 RU</td>
</tr>
<tr>
<td>• Scale Out Services → Hosted Mail, Search, Social, Cloud Hosting</td>
<td>2 Nodes per Rack Unit</td>
</tr>
<tr>
<td></td>
<td>• 2 Sockets @ 95W each</td>
</tr>
<tr>
<td></td>
<td>• Shared Chassis, Power Supply &amp; Cooling</td>
</tr>
<tr>
<td></td>
<td>• Google, FB, Amazon Custom Datacenters</td>
</tr>
<tr>
<td><strong>Platforms</strong></td>
<td>3 RU</td>
</tr>
<tr>
<td>• Dell PowerEdge C, HP ProLiant Microserver, DCS custom</td>
<td>8/12 Nodes in 3RU</td>
</tr>
<tr>
<td></td>
<td>• Single Socket @ 45W</td>
</tr>
<tr>
<td></td>
<td>• Shared Chassis, Power Supply, &amp; Cooling</td>
</tr>
<tr>
<td></td>
<td>• Dell PowerEdge 5220, Supermicro MicroCloud</td>
</tr>
<tr>
<td><strong>Typical Specifications</strong></td>
<td>TOMORROW: 10 RU</td>
</tr>
<tr>
<td>• 1/2 Socket 2/4 core 2.8GHz, 80W</td>
<td>• 256/ 512 Nodes in 10RU</td>
</tr>
<tr>
<td>• 280 SpecIntRate</td>
<td>• Single Socket @ 10-20W</td>
</tr>
<tr>
<td>• System Power &lt;500W; Cost &lt;$2K</td>
<td>• Shared IO Resources</td>
</tr>
<tr>
<td></td>
<td>• Integrated ToR Switch</td>
</tr>
<tr>
<td></td>
<td>• SeaMicro SM10000, HP Redstone</td>
</tr>
</tbody>
</table>
Integration
- Cores + memory + networking + I/O
- Lower latency, better QoS
  - Multiple Priorities
  - B/W guarantees

Efficient Out-of Order Cores
- Break tradeoff between wimpy and brawny cores
- Energy efficiency at good performance (ARM-based processors are well suited here)

Virtualization Support
- Improve utilization without hurting performance

Highly Integrated Server on Chip

Efficient Low Latency Interconnect

ARMv8 (Oban): Fully Backwards Compatible
New 64b ISA + Current 32b ISA

New: ARMv8

- 64b (General) and 128b (FP,SIMD) registers
- SP, PC no longer general purpose registers
- Uniform load/store addressing modes
- Larger data and instr. offset ranges
- Simplified load/store multiple instructions

- Reduced conditional instructions
- 32 128b FP/SIMD architecture registers
- No SIMD on general purpose registers
- New instructions for debug, TLB, barriers
- New Crypto acceleration instructions

ARMv8

- New High Performance 64bit ISA + compatibility with existing 32bit ISA
- Full CPU, IO, Interrupt, Timer Virtualization
- Enhanced 128b SIMD operations
- High performance Floating-Point operations including FMADD
- Standard Performance Monitoring, Instr. Trace and Debug Architecture
X-Gene™ CPU Design Goals

- **High-Performance Low-Power Microarchitecture**
  - Design point targets balance between performance, power, and size
  - Maximum “bang for the buck”

- **Low Power Microarchitecture Features**
  - Sophisticated branch prediction, Caches, Unified register renaming
  - Minimal instruction replay cases
  - Separate smaller schedulers per pipe
  - Full set of power management features

- **Good Single-Thread Performance, but also Efficiently Scalable to Many Cores**
  - Scalable CPU and interconnect architecture 2-128 cores
  - High bandwidth, low latency switch fabric > 1Tbps
  - High-performance distributed hardware cache coherency

- **Technology Portability**
  - Fully synthesizable RTL
  - Semi-custom cell-based design methodology
  - Small targeted set of custom macros (plus clock distribution cells/macros)
X-Gene™ Processor Module

Processor Module

- 2 cores + shared L2 cache
- 4 wide out-of-order superscalar microarchitecture
- Integer, scalar, HP/SP/DP FPU and 128b SIMD engine
- Hardware virtualization support
- Hardware tablewalk and nested page tables
- Full set of static and dynamic power management features
  - Fine grain/macro clock gating, DVFS
  - C0, C1, C3, C4, C6 states

Cache Hierarchy

- Separate L1I and L1D caches
- Shared L2 cache among 2 CPUs
- Last-level globally shared L3 Cache
- Advanced hardware prefetch in L1 and L2
- L2 inclusive of L1 write-thru data caches

RAS

- ECC and Parity protection of all Caches, Tags, TLBs
- Data poisoning and error isolation
X-Gene™ Instruction Fetch

I-Cache & Fetch Unit
- Fetch multiple instructions /cycle
- Instruction pre-decode bits stored with each cache-line
- Single cycle scan to pick next predicted taken branch
- 2-level branch prediction
- Branch Target Buffer
- Conditional, call/return branch predictors
- History based indirect branch predictors
- First level fully-associative L1 TLB
X-Gene™ Instructions Decoding/Grouping

Decoding/Grouping
- Quad instruction grouping
- On the fly “CISC” instruction to RISC OP mapping
- Full renaming of registers
- Dispatch into execution schedulers
- No dispatch constraints on instruction grouping
X-Gene™ Reorder Buffer, Dispatch and Control

Pipeline Control
- Branch checkpoint buffer
- Re-order buffer
- Unified register file
X-Gene™ Integer, Branch, Load and Store Units

Integer and Load/Store Units
- Separate branch pipe
- Out of order schedulers
- Two integer ops /cycle
- Fully pipelined execution units
- Separate load and store pipes
- Memory disambiguation
**Floating Point & SIMD Unit**

- Separate FP/SIMD renamer
- Out of order scheduler
- Full frequency scalar FPU
- Full frequency int / FP SIMD unit
- Fully pipelined operations
- FP / SIMD Load and FP / SIMD Store and Reg Op per cycle
Data Cache

- First level fully-associative TLB
- Write-through to L2 with write-combining
- Store to load forwarding
- Banked data arrays for performance and low power
Memory Management Unit

- Set-associative second-level TLB
- Supports all architecture page sizes
- Nested page-table walker
X-Gene™ – CPU Memory Subsystem

High-performance Symmetric Multi-core Design
- Modular architecture
- Three level cache hierarchy
- Globally shared L3 Cache

Coherent Network
- Runs at full CPU frequency
- <15ns latency, ~200GB/s B/W
- Over 400 transactions in flight
- Central snoop controller and ordering point
- Decoupled frequency and power domains
- Support global cache and TLB inv operations

Bridges
- Memory Bridges to DRAM interfaces
- IO Bridge for SOC connectivity
X-Gene™ Server on Chip

64bit ARM Server Class CPU → Multi-core for Distributed Computing
Increased Memory Capacity and 10G I/O Integration
Integrated Peripherals and L2 Switching
Workload Specific Specific Acceleration

Available: 2H’12
**Right Sizing + Connected On-Chip Fabric**

**Customizable Blade Design**
- Configurable swappable blades within 1 sled
- Networking/Compute/Storage shared over common bed of CPUs → Saves Power & Cost

**Overall System Optimization**
- Integrated NIC and IO chipset
- Load Balancing Across multiple blades to Optimize System Balance
- Shared Resources for System Management, Power and Cooling

**System Capabilities**
- 1000s of CPU cores in 10RU
- 100s of CPU cores per blade
- 100s of Gbps of network bandwidth
- 10s of Tbps of interconnect fabric bandwidth
X-Gene™

Worlds First True Low Power Server-on-Chip™

A Perfect Storm Cometh …
SPARC T5: 16-core CMT Processor with Glueless 1-Hop Scaling to 8-Sockets

Sebastian Turullols and Ram Sivaramakrishnan
Hardware Directors, Microelectronics
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Outline

• Design Objectives
• SPARC T5 Processor Overview
• Core S3
• Cache Hierarchy Components
• Internode Coherency for 8-Socket Scaling
• Power Management Advances
• PCI-Express Gen3 I/O Subsystem
• Summary
SPARC T5 Design Objectives

• Multiply performance
• Achieve highly efficient 8-socket glueless 1-hop scalability
• Optimize for Oracle workloads and Engineered Systems
• Maximize power efficiency
• Provide Enterprise Class RAS
T5 Processor Overview

- 16 S3 cores @ 3.6GHz
- 8MB shared L3 Cache
- 8 DDR3 BL8 Schedulers providing 80 GB/s BW
- 8-way 1-hop glueless scalability
- Integrated 2x8 PCIe Gen 3
- Advanced Power Management with DVFS
S3 Core Recap

- 28nm port from 40nm T4
- Out-of-order, dual-issue
- High frequency achieved with 3.6GHz 16 stage integer pipe
- Dynamically threaded, one to eight strands
- Accelerates 16 encryption algorithms and random number generation
SPARC T5 Leads in On-Chip Encryption Acceleration

- Built in, zero-overhead crypto
- Works with Solaris ZFS file system for faster file system encryption
- Provides secure consolidation with dynamic VM migration

<table>
<thead>
<tr>
<th>On-Chip Accelerators</th>
<th>SPARC T5</th>
<th>IBM Power7</th>
<th>Intel Westmere/Sandybridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric /Public Key Encryption</td>
<td>RSA, DH, DSA, ECC</td>
<td>none</td>
<td>RSA, ECC</td>
</tr>
<tr>
<td>Symmetric Key / Bulk Encryption</td>
<td>AES, DES, 3DES, Camellia, Kasumi</td>
<td>none</td>
<td>AES</td>
</tr>
<tr>
<td>Message Digest / Hash Functions</td>
<td>CRC32c, MD5, Sha-1, SHA-224, SHA-256, SHA-384, SHA-512</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>Random Number Generation</td>
<td>Supported</td>
<td>none</td>
<td>Supported</td>
</tr>
</tbody>
</table>
Outline

• Design Objectives
• SPARC T5 Processor Overview
• Core S3
• Cache Hierarchy Components
• Internode Coherency for 8-Socket Scaling
• Power Management Advances
• PCI-Express Gen3 I/O Subsystem
• Summary
Core Caches

• 16 KB 4-way set associative L1 Instruction cache
• 16 KB 4-way set associative write through L1 Data cache
• 128 KB 8-way set associative, unified, inclusive L2 cache
L2-L3 Interconnect

• 8x9 Crossbar Switch connects the 16 cores to
  – 8 address interleaved address banks and
  – an I/O bridge

• The L3-L2 direction contains a control and data network.
  – control network provides a heads up for dependent instruction
    wake-up
  – Data network is used to return line fill data and send L3-L2 snoops

• Crossbar network has a bisection BW of 1 TBps, 2x T4
L3 Cache Overview

- 8MB (1MB per bank) 16-way set associative inclusive cache
- MOESI Coherence States per 64B cache line
- Tag results are used to
  - qualify data array access
  - align dependent issue with data return or conditionally flush thread
- Precise tag “reverse” directory keeps L2s coherent
L3 Cache Overview (continued)

• Speeds up IO by allocating DMA buffers in the cache
  – Enhances clustered application performance

• Acceleration of contended locks
  – L3 forms a chain of same address requests
  – Processes them atomically on receiving an exclusive copy

• Supports coherent flushing and retirement of cache lines to avoid persistent errors
Outline

• Design Objectives
• SPARC T5 Processor Overview
• Core S3
• Cache Hierarchy Components
• Internode Coherency for 8-Socket Scaling
• Power Management Advances
• PCI-Express Gen3 I/O Subsystem
• Summary
Internode Coherency Overview

- Glueless 1-hop scaling to eight sockets
- A precise directory tracks all L3s in the system
  - striped across all processors
  - stored in on-chip SRAMs
  - flexible for different socket counts
- Higher BW efficiency than snoop-based protocols enables better scaling
  - 50% more effective bandwidth than comparable snoopy implementation
Internode Coherency Fabric

- Each link is 14 lanes wide and runs up to 15Gbps per lane
- Directly connected links minimize latency
- Trunked links achieve more bandwidth in smaller configurations
- Supports single lane failover
Internode Performance Optimizations

- Speculative memory reads prior to cache line serialization in the directory
- Cache-to-cache line transfers between nodes
- Dynamic congestion avoidance routes inter-node data around congested links
1. A Requester issues a **NodeRequest** to the Directory and a **SpeculativeRead** to the Memory Home

2. After a Directory lookup, either a **HomeRead** or a **SourceData** request is generated

3. **Data** is returned from the Memory Home or C2C Slave

**31% Latency Savings**
Dynamic Congestion Avoidance
T5-8 Bandwidth

DDR3-1066 1+TB/sec

Coherency Bisection Bandwidth 840 GB/sec

PCI Gen3 Bandwidth 256 GB/sec
Multiprocessor Performance

T5 8-Socket Scaling

- CPU
- Java
- OLTP

Socket Count

Scaling
Outline

• Design Objectives
• SPARC T5 Processor Overview
• Core S3
• Cache Hierarchy Components
• Internode Coherency for 8-Socket Scaling
• Power Management Advances
• PCI-Express Gen3 I/O Subsystem
• Summary
Power Management Advances

- Hardware saves power below 100% utilization with:
  - Chip wide DVFS
  - Per core pair cycle skipping
  - SerDes power scaling
  - DIMM off-lining w/ Dynamic Reconfiguration
  - DRAM PPSE and PPFE support
  - PCI Express Power Management
  - Clock Gating
- When peak performance is demanded
  - Power Management Controller achieves maximum frequency within customer imposed power and thermal limits
Power Management Controller: Elastic Savings

- Hardware saves power below 100% utilization
  - Chip wide DVFS
  - Per core pair cycle skipping
- Software monitors frequency needs of all cores
  - Puts chip at DVFS point satisfying all cores requirements
  - Puts core pairs at lowest cycle skip ratio satisfying 2 cores in the pair

![Power vs Frequency with DVFS](chart.png)

\[ f(x) = x^{2.82} \]
Coherency Link Power Savings

- Link scaling (4,3,2,1 dynamically as needed)
  - Hardware monitors link utilization
  - Software sets entry exit policy (thresholds and dwell times)

![Diagram showing link scaling from 4 to 2 links with power savings indicated as 25W Savings]
Memory Link Power Savings

• Two-levels of memory link standby
  – L0s: Power savings with fast wake up
    • Light sleep for N frames, then wake up and listen for data
  – L1: Much more power savings with longer wake up
    • Completely power off both tx and rx except for PLL
    • Used for unallocated memory regions
Peak Performance Thermal Management

- 4 thermal diodes per chip
  - centered in core quads
- If any $T >$ high-water mark
  - Drop Freq, $V$
- If all $T <$ low-water mark
  - Raise Freq, $V$

![Diagram showing thermal management components and connections](attachment:diagram.png)
Peak Performance Current Management

• Drop F,V if any current > high-water mark
• Raise F,V if any current < low-water mark
• Controls currents for CPU VDD plus motherboard and DIMMs
Outline

• Design Objectives
• SPARC T5 Processor Overview
• Core S3
• Cache Hierarchy Components
• Internode Coherency for 8-Socket Scaling
• Power Management Advances
• PCI-Express Gen3 I/O Subsystem
• Summary
T5 PCIe Subsystem

- Dual x8 PCI Express Gen 3 ports provide 32 GB/s peak b/w
- Supports Atomic Fetch-and-Add, Unconditional-Swap and Compare-and-Swap operations
- Accelerates virtualized I/O with Oracle Solaris VMs
  - 128k virtual function address spaces ensure direct SR-IOV access for all logical domains
  - 64-bit DVMA space reduces IO mapping overhead, improving network performance
  - Guarantees fault and performance isolation among guest OS instances
- Supports PCI Express Power Management
## T5 PCIe Progression

<table>
<thead>
<tr>
<th></th>
<th>T4</th>
<th>T5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PCI Express revision</strong></td>
<td>Gen 2 (dual x8 ports)</td>
<td>Gen 3 (dual x8 ports)</td>
</tr>
<tr>
<td><strong>Throughput full duplex</strong></td>
<td>16 GBs</td>
<td>32 GBs</td>
</tr>
<tr>
<td><strong>Data Management Unit</strong></td>
<td>Single shared unit for both x8 PCIe ports</td>
<td>Two independent units one for each x8 PCIe port</td>
</tr>
<tr>
<td><strong>Physical Address Support</strong></td>
<td>44 bit</td>
<td>48 bit</td>
</tr>
<tr>
<td><strong>Transaction Id Identification on MSI and MSI-X</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>PCIe Atomic Transactions</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>TLP Processing Hints</strong></td>
<td>No</td>
<td>Yes, directs data to L3 cache</td>
</tr>
<tr>
<td><strong>PCIe 2.0 compliance (ECN “Internal Error Reporting”)</strong></td>
<td>Signaled via MSI interrupt</td>
<td>Signaled via PCIe message</td>
</tr>
</tbody>
</table>
Outline

• Design Objectives
• SPARC T5 Processor Overview
• Core S3
• Cache Hierarchy Components
• Internode Coherency for 8-Socket Scaling
• Power Management Advances
• PCI-Express Gen3 I/O Subsystem
• Summary
SPARC T5 Summary

- Processor provides
  - Leadership throughput and per-thread performance
  - The industry’s best on-chip encryption acceleration
  - Advanced power management
  - Highly-efficient one hop glueless scalability to 8 sockets
  - Enterprise-class general purpose computing and RAS

- SPARC T5 is the world's best processor for running Oracle software
  - Oracle Database, Fusion Applications, Fusion Middleware
Design Objectives Achieved

- Oracle workloads
- Engineered Systems
- Extends
  - on-chip crypto acceleration
  - RAS
- Scales to 8 sockets using directory
- Minimizes latency
- Avoids congestion
- Maximize bandwidth

- Double cores and cache
- Balance single thread and throughput
- Dynamically thread

- Maximizes peak performance
- Manages thermal and current loads
- Scales elastically

Optimize Systems
Multiply Performance
Scale Efficiently
Advance Power Management

SPARC T5

Optimize Systems
Multiply Performance
Scale Efficiently
Advance Power Management

SPARC T5
Q&A
Hardware and Software

Engineered to Work Together