T4: A Highly Threaded Server-on-a-Chip with Native Support for Heterogeneous Computing

Robert Golla – Senior Hardware Architect
Paul Jordan – Senior Principal Hardware Engineer
Oracle
Agenda

• T4 Overview
• S3 Core
  • Overview
  • Block Diagram
  • Pipeline
• T4 Performance
• Dynamic Threading
• Crypto
  • Overview
  • Block Diagram
  • Performance
• Summary
T4 Design Objectives

- Optimize Software and Hardware for Oracle Workloads and Engineered Systems
  - Extend SPARC ISA
- Performance
  - Much better singlethread performance vs T3
  - Double T3’s per thread throughput performance
  - Enhance overall crypto performance vs T3
- Compatibility
  - Maintain SPARC V9 and CMT model compatibility
  - Maintain current T3 system scalability
- Reliability
  - Extend T3’s RAS capabilities
T4 Chip Overview

- 8 SPARC S3 cores
  - 8 threads each
- Shared 4 MB L3
  - 8-banks
  - 16-way associative
- Two dual-channel DDR3-1066 memory controllers
- Two PCI-Express x8 2.0 ports
- Two 10G Ethernet ports
- TSMC
  - 40 nm
  - ~855 million transistors
S3 Core Overview

- Out-of-order
- Dual-issue
- Dynamically threaded
- Balanced pipeline design
  - Singlethread performance
    - Estimate ~5X S2’s SPECint2006* performance
    - Estimate ~7X S2’s SPECfp2006* performance
  - Throughput performance
    - ~2X S2’s per thread throughput performance
- High frequency, deep pipeline
  - 16 stage integer pipe
  - 3+ GHz

*SPEC, SPECfp, SPECint are registered trademarks of the Standard Performance Evaluation Corporation. Estimates as of 8/18/2011, see www.spec.org for more about SPEC.
S3 Core Overview

- Extensive branch prediction
  - Perceptron direction predictor
  - Return Stack to predict return addresses
  - Far and Indirect target predictors
  - BTC to reduce taken branch penalty

- Hardware / Software optimizations for Oracle applications
  - User level crypto instructions
  - PAUSE instruction
  - Fused compare-branch instruction

- HW prefetchers
  - Instruction cache sequential line prefetcher
  - Data cache stride based prefetcher

- 16 KB, 4-way, L1 instruction and data cache
- 128 KB, 8-way, unified private L2 cache
S3 Core Block Diagram

Thread Select

Branch Predictor

L1 I-Cache
16 KB
(4-way)

Way Select

4-instrs (16B)

Per Thread Instruction Buffers 1..8

Thread Select

8 threads

Instruction Decode

2-instrs

Rename

2-instrs

40-entry
Unified Pick Queue

2-instrs

Register Files
IRF/FRF/WRF

Commit with 128-entry ROB

Perf. Monitor

Private 128 KB L2
(8-way)

I-TLB
64-entries

MMU

D-TLB
128-entries

L1 D-Cache
16 KB
(4-way)

ALU0

ALU1

BRU

FGU

CRYPTO

Slot 0

Slot 1
S3 Core Pipeline

16 Stage Integer Pipeline

F1 F2 F3 S D1 D2 R1 R2 R3 P I1 I2 I3 I4 E W

3-cycle Crypto Pipeline

X1 X2 X3

5-cycle Load-Use

C1 C2 C3 B W

11-cycle Floating-Point Pipeline

FX1 FX2 FX3 FX4 FX5 FX6 FX7 FX8 FX9 FX10 FB FW

Pipeline Key

<table>
<thead>
<tr>
<th>F</th>
<th>Fetch</th>
<th>R</th>
<th>Rename</th>
<th>E</th>
<th>Execute</th>
<th>B</th>
<th>Bypass</th>
<th>FX</th>
<th>Floating-point execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Select</td>
<td>P</td>
<td>Pick</td>
<td>W</td>
<td>Write-back</td>
<td>X</td>
<td>Crypto execute</td>
<td>FB</td>
<td>Floating-point bypass</td>
</tr>
<tr>
<td>D</td>
<td>Decode</td>
<td>I</td>
<td>Issue</td>
<td>C</td>
<td>Data-cache</td>
<td></td>
<td></td>
<td>FW</td>
<td>Floating-point writeback</td>
</tr>
</tbody>
</table>

8/19/2011
Threaded S3 Core Pipeline View

- **Before Pick**
  - Only 1 thread per pipe stage

- **Pick to Commit**
  - Multiple threads per pipe stage

- **Commit**
  - Only 1 thread per pipe stage
T4 Relative Performance

Relative Performance T4 vs T1-T3*

- **T1**: 8 S1 cores, 32 threads
- **T2**: 8 S2 cores, 64 threads
- **T3**: 16 S2 cores, 128 threads
- **T4**: 8 S3 cores, 64 threads

*All performance estimates are relative to T1 performance.*
T4 Relative Performance

Estimated SPECint2006* Relative Performance T4/T3

Estimated SPECfp2006* Relative Performance T4/T3

*SPEC, SPECfp, SPECint are registered trademarks of the Standard Performance Evaluation Corporation. Estimates as of 8/18/2011, see www.spec.org for more about SPEC.
Dynamic Threading

- Many of the resources on S3 are shared between threads
  - Load-buffers, store-buffers, pick-queue, working-register-file, reorder-buffer, etc.

- Thread sharing of resources
  - Static resource allocation
    - Not optimal for heterogeneous workloads
  - Dynamic resource allocation
    - Better for heterogeneous workloads
    - Improves overall application scaling
    - Resources are dynamically configured between threads each cycle
      - No synchronization required
Dynamic Threading – Thread Hogs

• Thread hog definition
  • A thread which fails to release its shared resources in a timely fashion

• Thread hog mitigation using watermarks
  • High and low watermarks defined for each shared resource
    • High watermark reached by allocation
    • Low watermark reached by deallocation
  • Upon reaching high watermark, thread resource allocation stalls
  • Thread resource allocation remains stalled until low watermark is reached
Dynamic Threading – Thread Hogs

- Thread hog mitigation using rate of deallocation
  - Pick Queue (PQ) mitigation technique
    - PQ is most critical shared resource on S3
  - Threads are expected to deallocate PQ entries in a timely fashion
    - If not, thread is considered a PQ thread hog
- If thread is a PQ thread hog
  - PQ resource available to thread is reduced and made available to other threads
  - If hogging behavior continues, PQ resource is further reduced and made available to other threads
  - If thread deallocates PQ entries in a timely fashion, PQ resource is increased
Dynamic Threading – Thread Hogs

• Thread hog mitigation using flushes
  • Flush on L3 miss
    • When a load misses the L3, flush the thread
    • Flushing releases any allocated shared resources for that thread
  • Load/Store timeout
    • Some events are not covered by other thread hog mitigation policies
      • Load that RAWs to a previous store that misses the L3
      • Flush any load/store that is the oldest and does not commit for N cycles
  • Flush after IO
    • Flush thread after an IO access reaches Commit
Crypto Overview

- Crypto programmed via user instructions
- Instructions are either “in-pipe” or “out-of-pipe”
  - “in-pipe” set supports 3 cycle internal latency
    - AES, DES, Kasumi, Camellia, CRC32c
  - “out-of-pipe” set has long latency
    - MD5, SHA-1, SHA-256, SHA-512, MPMUL, MONTMUL, MONTSQR
- MPMUL, MONTMUL, MONTSQR have separate state machine
  - Stall Pick Queue to inject crypto multiplies
  - Fairness heuristic between crypto and non-crypto threads
Crypto Unit Block Diagram

Crypto unit shares the operand and result buses with the FGU.
Crypto Relative Performance

Relative Core Performance T4/T3

- AES-128 CBC Encrypt
- AES-128 CBC Decrypt
- AES-192 CBC Encrypt
- AES-192 CBC Decrypt
- AES-256 CBC Encrypt
- AES-256 CBC Decrypt
- CRC32c
- RSA-512 sign
- RSA-1024 sign
- RSA-2048 sign
- MD5
- SHA-1
- SHA-256
- SHA-512
Summary

• Next-generation processor for Oracle servers
  • Significantly improved per thread throughput performance
  • Much better singlethread performance

• Dynamic threading
  • Better for heterogeneous workloads
  • Improves overall application scaling

• Excellent overall crypto performance
  • Enables transparent encryption across Oracle software stack
Glossary

- SPC – SPARC core
- CCX – crossbar
- BTC – branch target cache
- IRF – integer register file
- WRF – working register file
- FRF – floating-point register file
- FGU – floating-point / graphics unit
- IB – instruction buffer