Post-Silicon Verification Methodology on Sun's UltraSPARC T2 Processor

HLDVT 2007

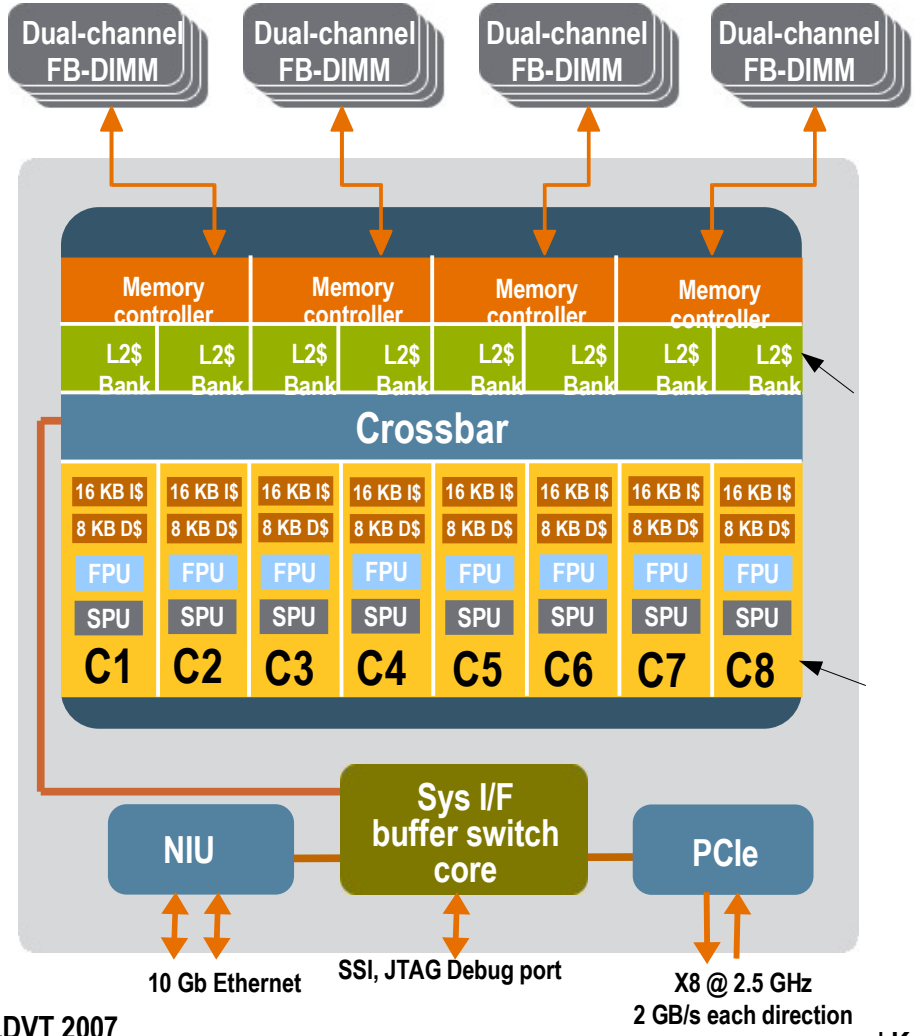
Jai Kumar, Verification Technologist
Catherine Ahlschlager, Manager, Formal Verification
Peter Isberg, Manager, TestGeneration Technologies

Sun Microsystems Inc.
jai.kumar@sun.com
http://sun.com

Outline

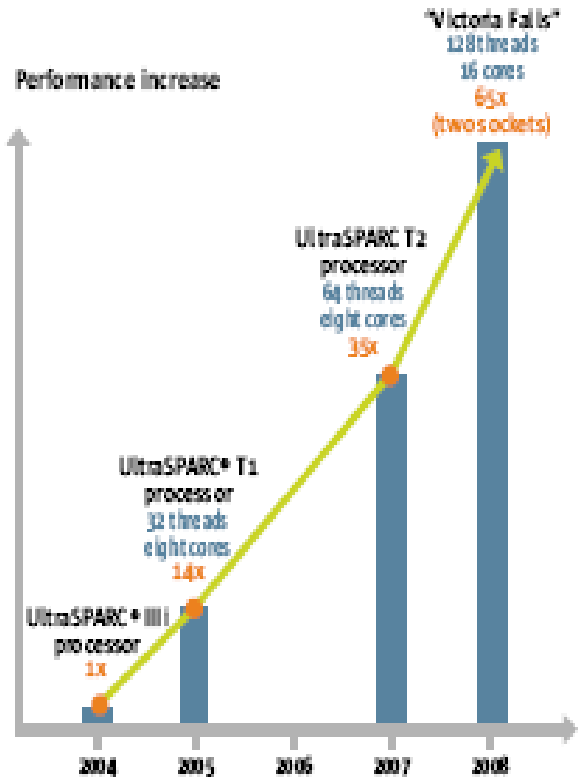
- Introduction
- Pre-Silicon Verification
- On-Chip Debug Features
- Post-Si Validation
- Summary

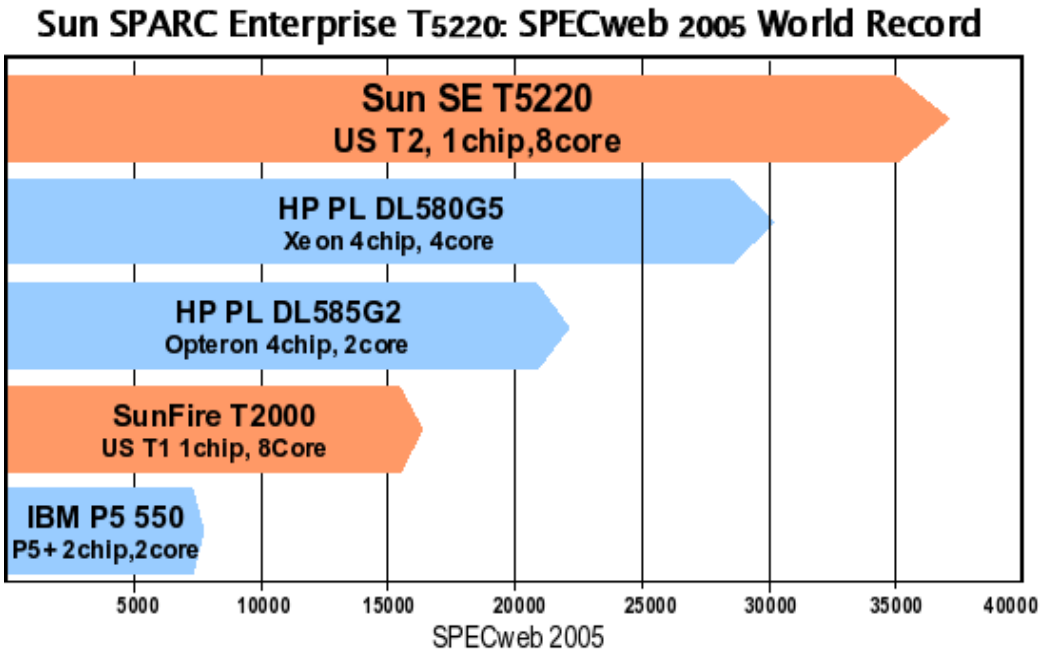
UltraSPARC T2: True Server On a Chip



- 2x+ performance over UltraSPARC T1, within the same power envelope
- Up to 8 cores @1.4GHz
- 2x the threads
  - > Up to 64 threads per CPU
- 2x the memory
  - > Up to 128GB memory
  - > Up to 16 full buffered Dimms
  - > 2.5x memory BW = 60+GB/S
- 8x FPUs, 1 fully pipelined floating point unit/core
- 4MB L2$ (8 banks) 16 way set
- Security co-processor per core
  - > DES, 3DES, AES, RC4, SHA1, SHA256, MD5, RSA to 2096 key, ECC
- Powers SunFire T5120, T5220, T6320 Servers

T2 Processor & System Relative Performance





SPEC, SPECweb reg tm of Standard Performance Evaluation Corporation. Results from www.spec.org as of October 26th 2007. Sun SPARC Enterprise T5220 (8 cores, 1 chip) 37,001 SPECweb2005. HP DL580G5 (16 cores, 4 chips) 30,261 SPECweb2005. HP DL585G2(8 cores, 4 chips) 22,254 SPECweb2005, Sun Fire T2000 (8 cores, 1 chip) 16407 SPECweb2005, IBM p5 550 (4 cores, 2 chips) 7881 SPECweb2005.

World's First Open Source Microprocessor

solaris™ OpenSPARC™

OpenSPARC.net

- Governed by GPL(2)
- OpenSPARC T1
  - > Complete chip architecture
  - > RTL Model – various core/thread comb.
  - > Verification Suite
  - > SW Stack – Hypervisor, OBP, Solaris
- OpenSPARC T2
  - > Programmer's Reference Manual
  - > Micro Architectural Specs

Linux

Verification Methodology

- Goal: Catch bugs early, efficiently
- Emphasis on solid Pre-Si Verif – minimize post-si issues
  - > Methodology
  - > Technology (Simulation, Formal, Emulation)
- DFx (Availability, Reliability, Serviceability, Test, Verification)
- Post-Si Verification – focussed on productization
  - > Bringup readiness – tools, environment
  - > Test Generators
  - > SW Stack (Reset, Hypervisor, OBP, Solaris, Drivers)
  - > Bug Reproducibility & Bug-fix validation

Cost of Bugs – simple analysis

- Showstopper bug early in design phase
  - Cost: almost negligible
- Showstopper bug close to tapeout
  - Cost: schedule impact
- Showstopper bug after tapeout
  - Cost: Few Silicon respins, schedule impact
- Showstopper bug close at Revenue Release
  - Loss of Revenue at $1+ Million per day
  - Reduce competitive edge
- Showstopper bug 1year after Revenue Release
  - Cost of recall at $150Million
  - Damages corporate reputation

Source: N.Winkworth & B.Blohm

Pre-Si Verification

- Philosophy: Multi-layered verification approach at right level of abstraction
- Strategy:
  - Architectural: ISS supports black-box testing
  - Simulation (RTL)
    - Env.: Stand Alone Testing, System-level testing, Gate-level FC, Assertions, Coverage
    - Tests: Directed, Random, Cycle-by-cycle checking w/ ISS, self checking
  - Formal
    - Property Checking, Clock-Domain Checking, Equivalency
  - HW Acceleration/Emulation
    - System-level testing, self-checking long randoms, sw-stack

DFx

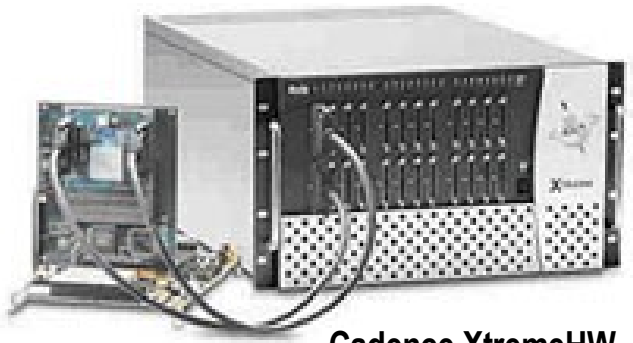
- Availability, Manufacturability, Reliability, Repeatability, Serviceability, Testability, Verification = silicon area trade-off
- Debug port
  - > used for observing L2 operations
  - > Analyze coverage of interesting cases
- Observability features
  - > Scandump – used for deadlock/hang debug
  - > Shadow scan – used for debug startvation/forward progress
- JTAG Enabled Debug Interface – JEDI
  - > Read, Write, Control of Core, EFU, LBIST, MBIST, MCU, PIU, CLK, Array, CSR
- Load N' Go (L2$ resident diag) – ITC paper

On-Chip Debug Features

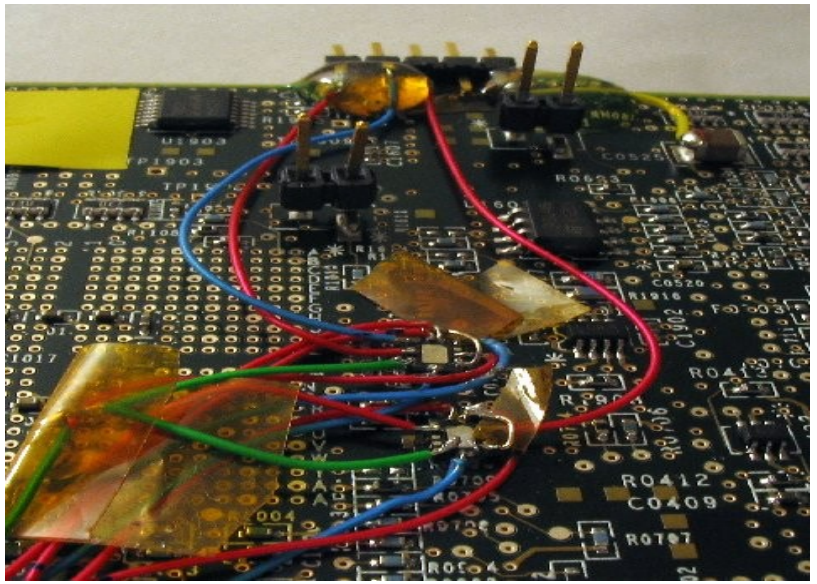
- On-Chip triggers (stop &scan)
- Debug Ports – transaction oriented
- Error Injection & Logging
- JTAG Read Port
- Full Scan Dump
- Memory dumps (SRAM, FIFOs)
- DTM mode (Deterministic Test Mode – put on tester & keep serdes cycle accurate)

Bringup Tasks accomplished pre-TO

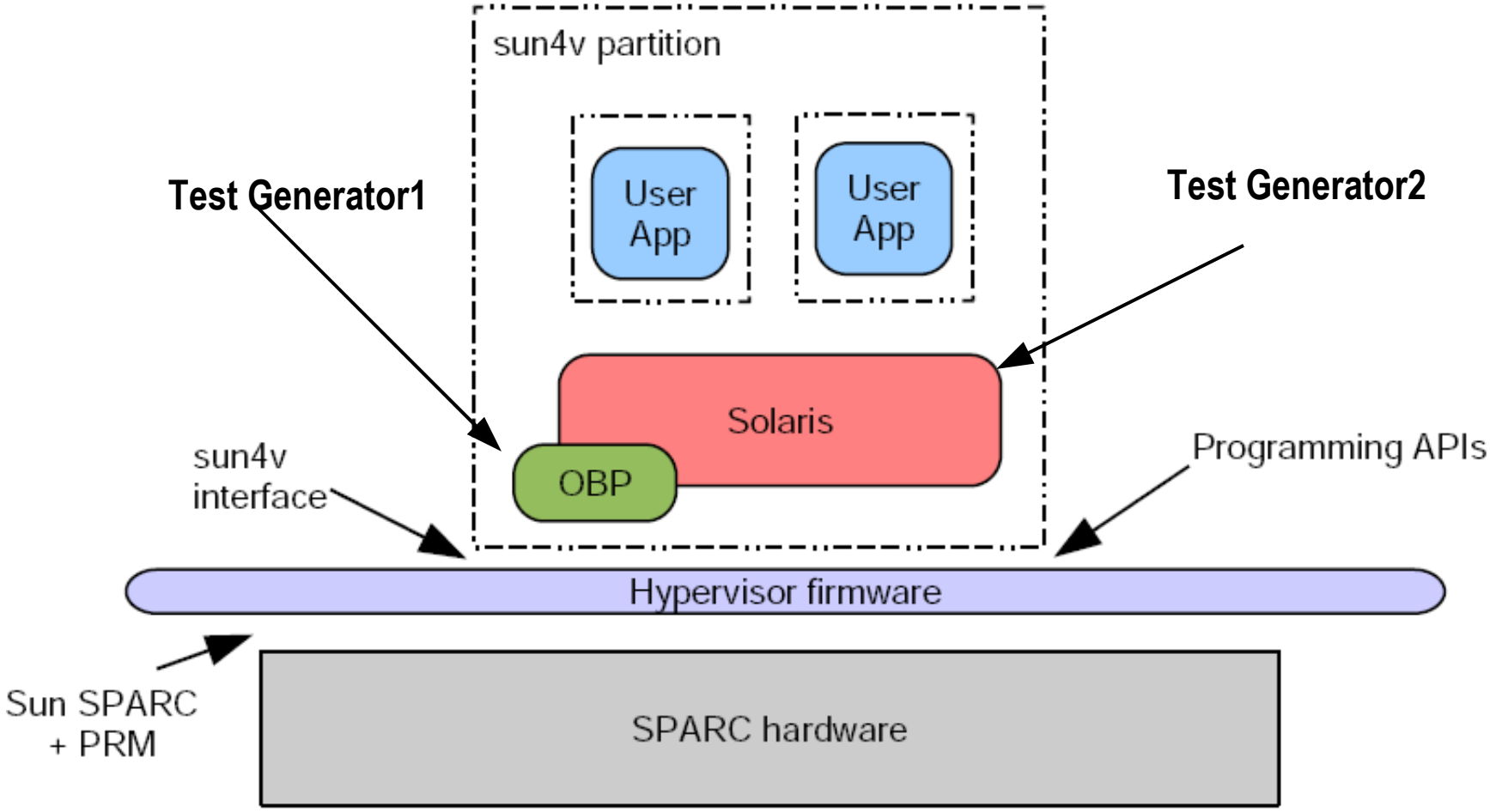
- Emulation enables design prototype - “virtual silicon”
  - > Early System integration
  - > SW Development & Debug
  - > Virtual Silicon Bringup
  - > System Readiness
    - > Motherboard, PCIE, Ethernet In-Circuit Emulation
  - > Test Generator Readiness
  - > Debug Tool Readiness
  - > System Debug training vehicle



Cadence XtremeHW



System-level Validation Approaches



Silicon Validation

- Test Generators:
  - > Boot-able system-level random test generator, allows feature interaction testing
    - Self-checking tests, user control via knobs, built-in CPU debug, repeatability, RAS testing (facilitated by on-chip error injectors)
    - IO Test - self-checking randomized IO device program for PCIe
      - Works with our device emulators (PCIe EP) – same RUST code in Emulation & real System
  - > TSO Tool – tests memory consistency – paper in ISCA
  - > System level random test generators
  - > System level stress test generator, focused crypto, IO, interrupt tests
- SW stack – Hypervisor, OBP, Solaris
- Sun Validation Test Suite – suite of directed tests, manufacturing tests, field failures

Silicon Debug Methodology

- Bug Identification
  - Reduce MTTF (down to few minutes)
- Bug Root Cause Analysis
  - Architecture review, Diagnostic team
  - Add more random generator tunables; debug features
  - Repeatability enhancements
  - Formal aided analysis to validate theories
- Bug Reproduction
  - LeCroy PCIe protocol analyzer, LA trigger on FBDIMM
  - CPU re-configuration
  - HW Emulator duplication

Bug-fix validation

- Bug fix creation
- Root out related bugs
- Simulation – replicate failure with directed test
- Emulation – enhance random tools to address coverage hole, re-run on fixed RTL, billions of cycles quickly
- Formal – validate the potential fixes and their side effects

Summary

- Due diligence in pre-si verification to minimize post-si issues
- Rapid prototyping using HW Emulation is key to enabling virtual silicon bring up before design tape out
- Debugging in post-si is neither easy nor quick
  - > Spend time leveraging, developing & testing tools before you need them: LA, Debug Ports, emulation, formal, debug
  - > Build for repeatability in Silicon & Test Generators
  - > System debug expertise
- Acknowledgements: Dave Wilkins & Niagara2 Verification Team