TSOtool: A Program for Verifying Memory Systems Using the Memory Consistency Model

Sudheendra Hangal[†], Durgam Vahia[‡], Chaiyasit Manovit[‡],
Joseph Lu[‡] and Sridhar Narayanan[‡]
tsotool@sun.com

ISCA-2004





[†]Sun Microsystems India Pvt. Ltd., Bangalore (India)
[‡]Sun Microsystems, Sunnyvale, CA (USA)

Goal:
Find the hard memory-related bugs
in Sun's multiprocessor systems

Motivation

- Many bugs in MP memory subsystem
  - Hard to find, hard to debug
  - Require silicon revs; impact time-to-market; cost $$
- Almost all Sun systems are MP (on-a-chip)
- Throughput computing: SMT, CMP, CMT
  - Now it gets interesting

Why is MP Verification Hard ?

- Many elements in memory hierarchy
  - Asynchronous load-store units, L1,L2,L3 caches, prefetchers, bus protocols, system interconnects, memory controllers, DRAM technologies, etc.
- Many optimizations
  - For performance & scalability
- Many derivative processors & systems
  - Each makes changes to the memory system

Prior Methodology

- Use a reference model to check design
  - Can't fully check pseudo-random MP programs with data races; only for obvious problems
  - Creating 100% cycle-accurate reference is hard
- “False Sharing”
  - Within a $-line, CPUs write non-overlapping bytes
- Specific MP idioms
  - “Litmus tests”, MP code for mutexes, locks, etc.

TSOtool Methodology

- Create a short, pseudo-random program with intense sharing
  - Hopefully, hit corner cases faster
- Analyze correctness of observed architectural results w.r.t. memory model
  - Microarchitecture agnostic
- No dependence on simulation observations
  - Faster simulation (big win for h/w accelerators)
  - Use observability if available

Memory Consistency Model

- Contract between programmer ↔ system

  A formal specification of how memory appears to behave to a programmer. Examples:
  SC, PC, TSO, PSO, RMO, RC, etc.

  Challenge to correctly implement for architects, designers, system programmers, ...

- All Sun systems support TSO

Ref: Adve and Gharachorloo's tutorial
http://citeseer.nj.nec.com/adve95shared.html

TSO Specification

Loosely speaking:

Instructions appear to execute in program order
EXCEPT S->L order relaxed;
loads may “overtake” prior stores on same CPU

Store is visible on same processor before others
(Allows store buffer bypass)

Notation

2 orders: ';' (program order) & '≤' (global order)
4 operations: L for loads, S for stores [L;S] for swaps

L_X^i	Load to address X on processor i
S_Y^j	Store to address Y on processor j
$[L_Z^k ; S_Z^k]$	Atomic swap to address Z on processor k
M	Memory barrier

TSO Formal Specification

LoadOp $L^i_X ; Op^i_Y \Rightarrow L^i_X \leq Op^i_Y$

StoreStore $S^i_X ; S^i_Y \Rightarrow S^i_X \leq S^i_Y$

Order $(S^i_X \leq S^j_Y) \vee (S^j_Y \leq S^i_X)$

Termination $S^i_X \wedge (L^j_X ;) \infty \Rightarrow \exists L^j_X \in (L^j_X ;) \infty \; such\, that \; S^i_X \leq L^j_X$

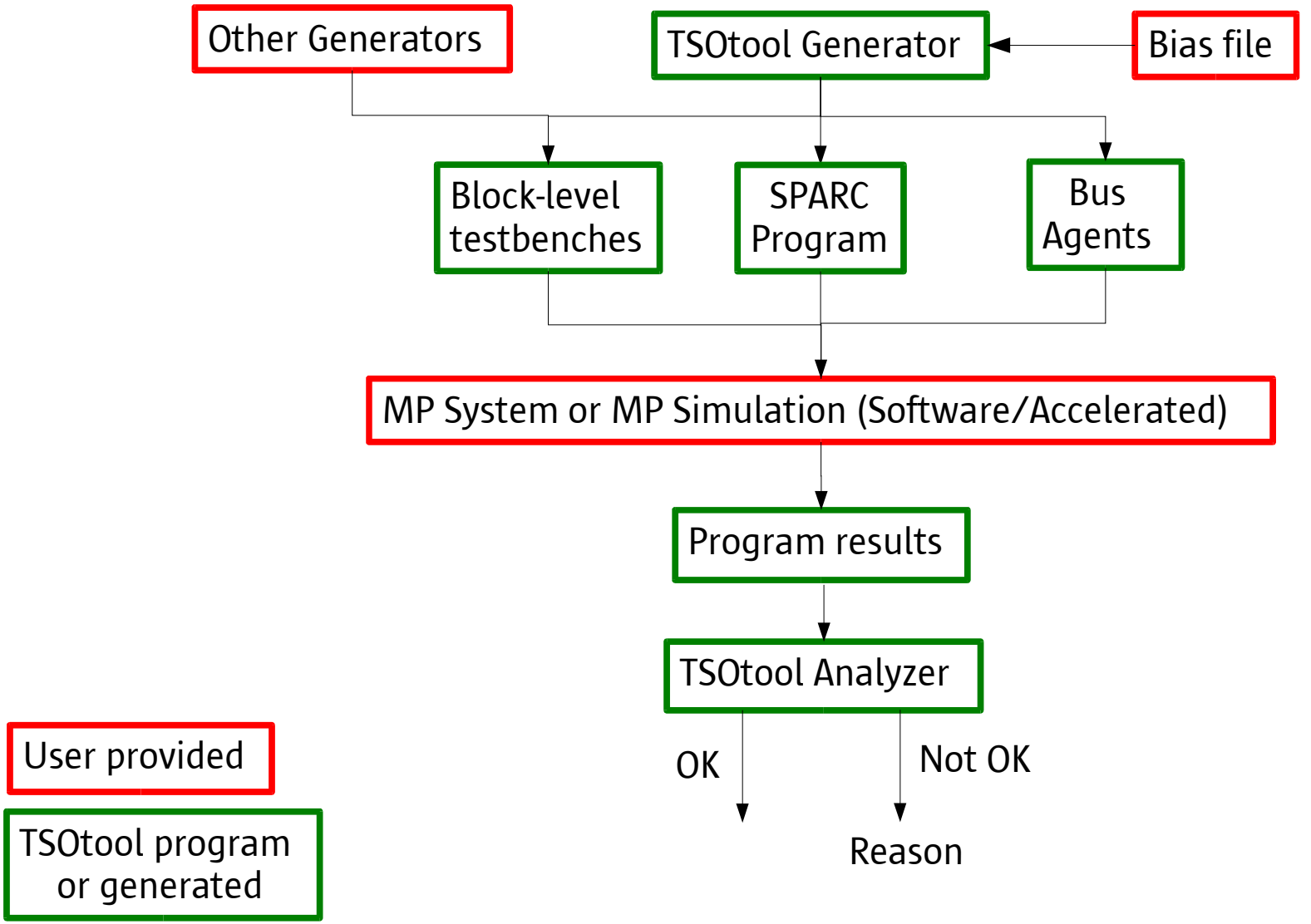
Atomicity $[L^i_X ; S^i_X] \Rightarrow (L^i_X \leq S^i_X) \wedge (\forall S^j_Y : S^j_Y \leq L^i_X \vee S^i_X \leq S^j_Y)$

Value $Val[L^i_X] = Val[MAX \leq \{(S^k_X | S^k_X \leq L^i_X) \cup (S^i_X | S^i_X ; L^i_X)\}]$

Membar $Op_1 ; M ; Op_2 \Rightarrow Op_1 \leq Op_2$

Ref: Formal specifications of Memory models, P. Sindhu et al (Xerox PARC)

TSOtool Usage



TSOtool Test Generation

- Real-world instruction set (SPARC)
  - All operations related to memory
    LD,DWLD,QWLD,BLD,AQLD,PREFETCH
    ST,DWST,QWST,BST,BSTC
    SWAP,CAS,CASX, MEMBAR, FLUSH
    Branches, ASI, Replacements, Interrupts, Macros, …
- All stores write a unique value
  - "read-mapping": Map(L) = S iff Val[L] = Val[S]

TSOtool Analysis

- Is result compatible with TSO axioms?
- Represent loads/stores as nodes in a graph
  Add edges to create constraints on ≤
  If graph has a cycle, ≤ is not a valid order
  If graph has no cycle, ≤ is an order compatible with program results and axioms
- No false failures
  - But not guaranteed to catch a true failure

Analysis Algorithm

- Add Edges

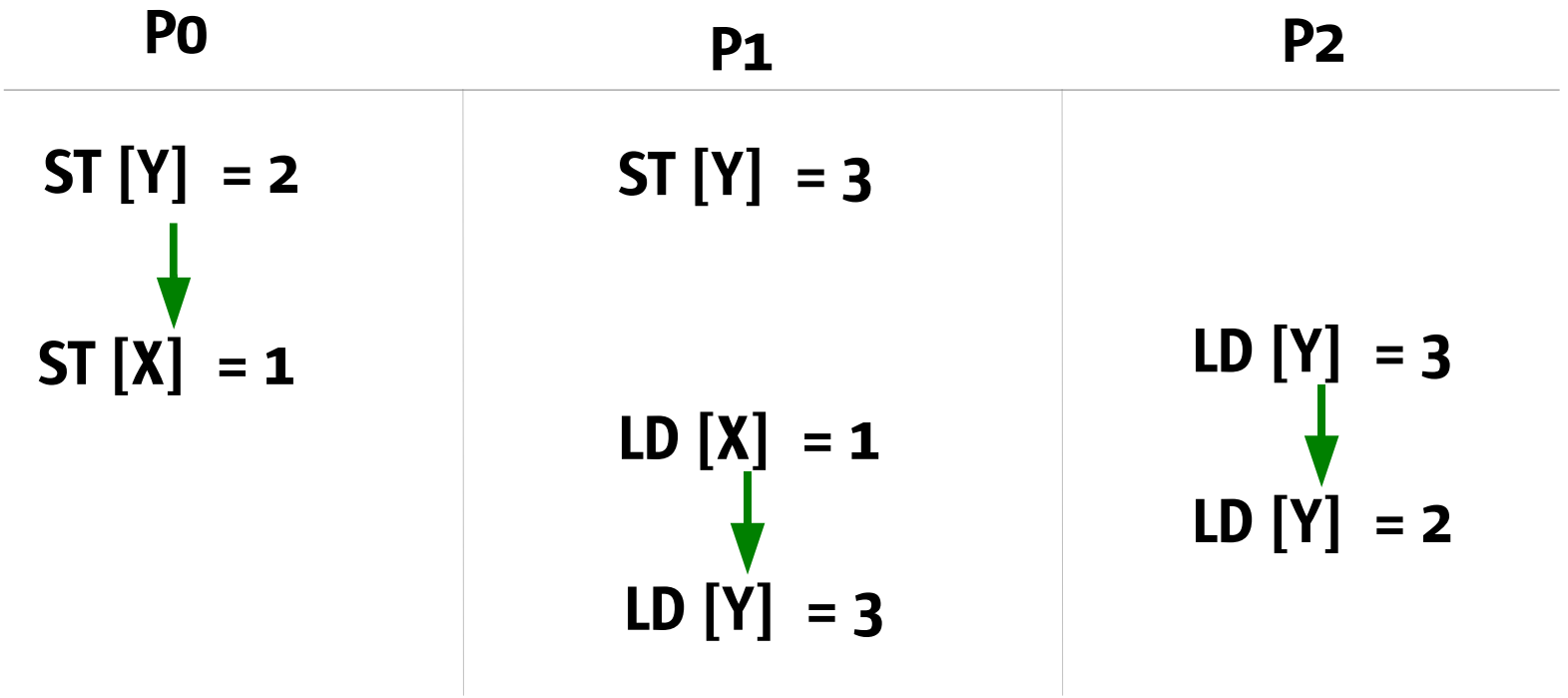
$L ; Op \Rightarrow L \leq Op,\ S ; S' \Rightarrow S \leq S'$	(LdOp and StoreStore Axioms)
$S ; M ; L \Rightarrow S \leq L$	(Membar Axiom)
$Op \leq S \wedge [L;S] \Rightarrow Op \leq L$	(Atomicity Axiom)
$L \leq Op \wedge [L;S] \Rightarrow S \leq Op$	(Atomicity Axiom)
all Ops to the same address:	
$Val[L] = Val[S] \Rightarrow \neg S;L \Rightarrow S \leq L$	(Value Axiom)
$Val[L] = Val[S] \wedge S';L \Rightarrow S' \leq S$	(Value Axiom)
$Val[L] = Val[S] \wedge S' \leq L \Rightarrow S' \leq S$	(Value Axiom)
$Val[L] = Val[S] \wedge S \leq S' \Rightarrow L \leq S'$	(Value Axiom)

- Last 2 steps
  - ≤ used on L.H.S. (but we're deriving ≤)
  - So, iterate over these steps till fixed point

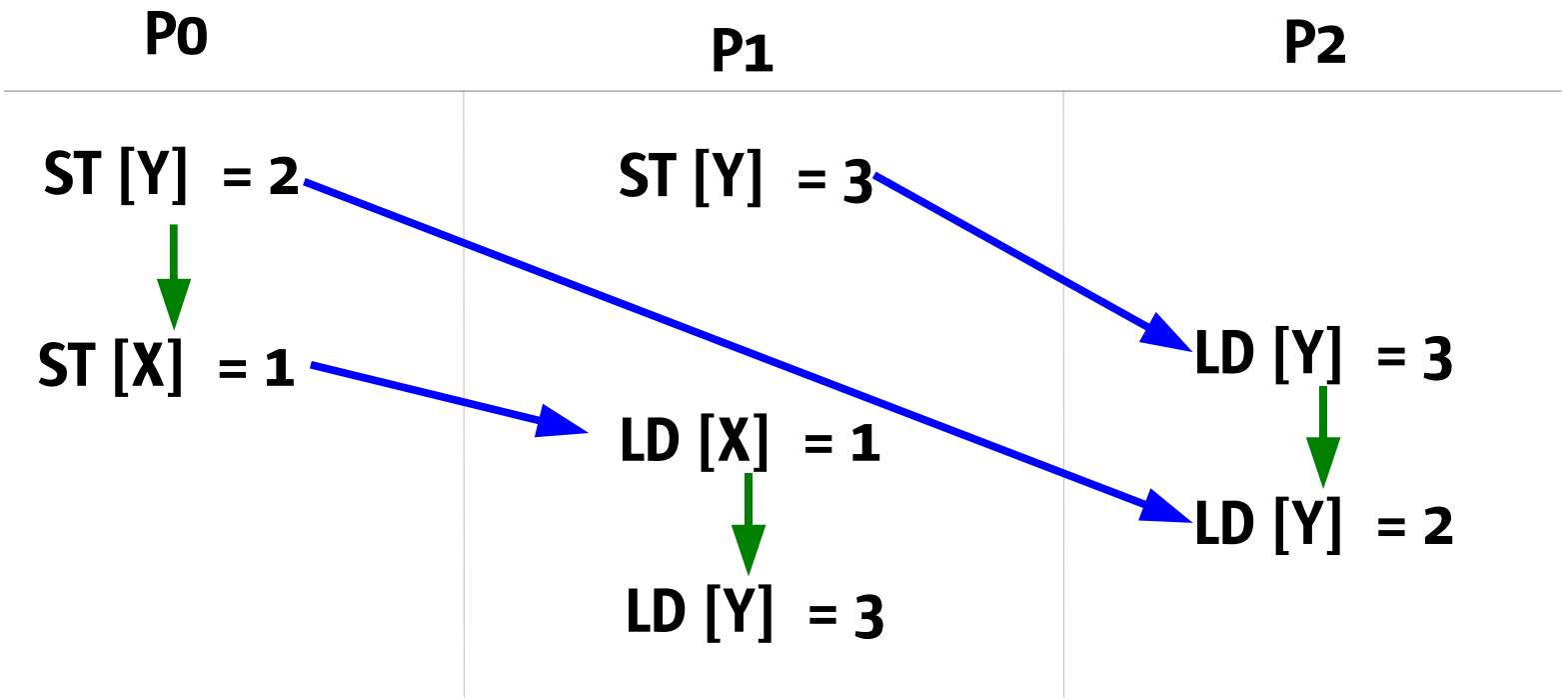
Analysis Example (Cycle)

P0	P1	P2
ST [Y] = 2	ST [Y] = 3	
ST [X] = 1		LD [Y] = 3
	LD [X] = 1	LD [Y] = 2
	LD [Y] = 3	

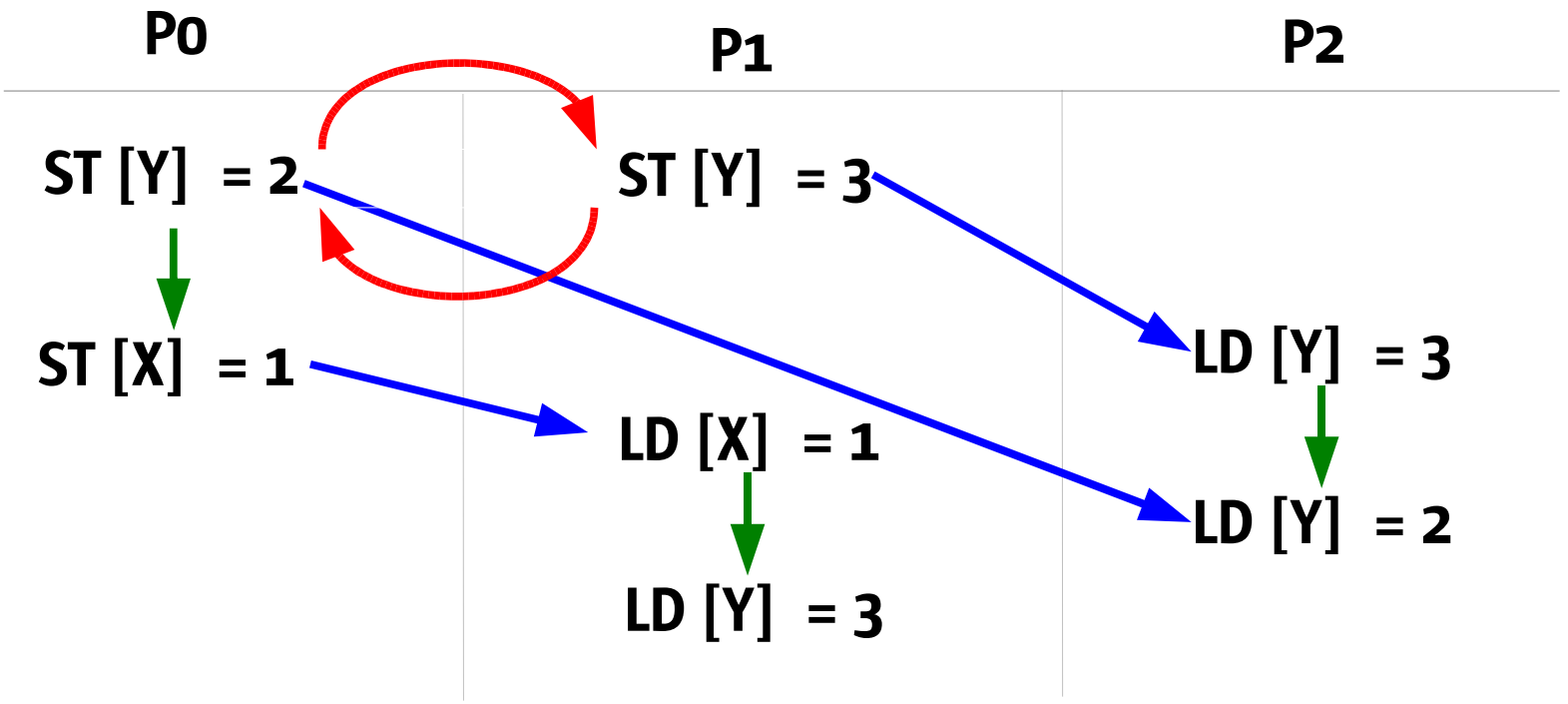
Analysis Example (Cycle)



Analysis Example (Cycle)



Analysis Example (Cycle)



Bugs Found

- Run on all recent Sun CPUs and systems
- Many problems caught early

  (Data corruption, atomicity violations, invalid instruction reordering)

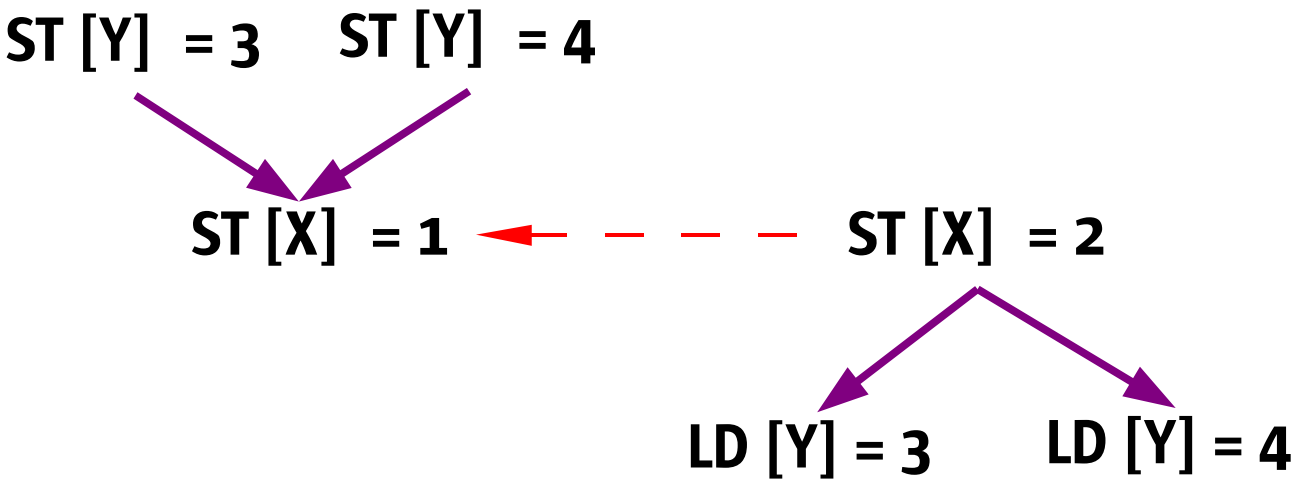
  Lost tag write to Write cache
  Lock for atomic instruction released too early
  Prefetch cache missed an invalidate
  Ordering between cacheable and non-cacheable queues
  DRAM controller corrupted speculative load request
  Software emulation routines

Analysis Complexity

- Complexity results for Verifying SC [GK94]
  - n is # memory operations
  - NP-Complete for unlimited # of processors
    (even if read-mapping is known)
  - n^k for k processors
- TSO version also NP-Complete
- TSOtool analysis is polynomial time
  - See paper for details

A Missed Relation

- Write Order axiom may not be satisfied



Summary

- TSOtool finds hard bugs in real designs
  - Incomplete algorithm is useful
  - Allows checking of pseudo-random MP with races
- Exposes failures with shorter tests
- Microarchitecture independent
  - Useful pre-Si, post-Si

Related Work

Complexity Proofs
- [Gibbons, Korach] Complexity of verifying SC (VSC) and flavors
- [Cantin] Verifying Memory Coherence (VMC) and flavors

Testing Approaches
- Industrial Design Groups: Sun, Intel, IBM, HP, etc.
- [Collier] Litmus tests (ARCHTEST)

Protocol Verification
- [Sorin et al] Lamport clocks for protocol verification

Formal Methods
- [Nalumasu et al] Test model checking
- [Park, Dill] Executable specification

Constraint Graphs
- [Landin et al] Access Graphs
- [Cain et al] Constraint graph analysis

Questions ?

tsotool@sun.com

Backup slides

TSOtool Test Generation (2)

- Real-world environments
  - MP systems (with OS), simulation models (full-chip, block-level, high-level/RTL/gate, accelerated)
- Every load value is saved
  - In program (some perturbation)
  - Directly from simulation if possible
- Users control bias
  - Typically few shared addresses (1-1000)
  - Typically few instructions (100-100K)