



OpenSPARC™

OpenSPARC Simulators

Stephen Henry

Sr. Staff Engineer

Architecture Technology Group

Microelectronics

Sun Microsystems



Agenda

- OpenSPARC simulators overview
- RTL co-simulation
- Full-system simulation
- CPU & device model interface
- User interface
- Trace
- Disk image
- Software development tool

OpenSPARC Arch Tools Download

- SAM: instruction-accurate SPARC full-system simulator
- SAS/NAS: instruction-accurate SPARC arch. simulator
- Binary images for simulation: Solaris 10, Hypervisor, OBP, etc
- Legion: SPARC full-system simulator for firmware and software development
- Hypervisor source code
- Documentation

OpenSPARC Arch Tools Download

- OpenSPARC-T1
 - > <http://www.opensparc.net/opensparc-t1/index.html>
 - > OpenSPARCT1_Arch.1.5.tar.bz2
- OpenSPARC-T2
 - > <http://www.opensparc.net/opensparc-t2/index.html>
 - > OpenSPARCT2_Arch.1.1.tar.bz2

What is SAM

- SPARC Architecture Model
 - > SPARC architecture golden reference model
 - > SPARC full-system simulator
- Functional simulator, no timing, no cache
- Usage
 - > RTL verification
 - > Solaris boot disk validation
 - > Device model development
 - > Software development
 - > Benchmark tracing

Accurate CPU Simulation

- SAM models the functions of SPARC processors
 - > SPARC instruction set
 - > Memory management, TLB
 - > ASI translation
 - > Privileged and hyperprivileged protection
 - > Accurate trap priority
 - > IEEE floating-point instructions behavior and exceptions
- Used to verify RTL design via co-simulation

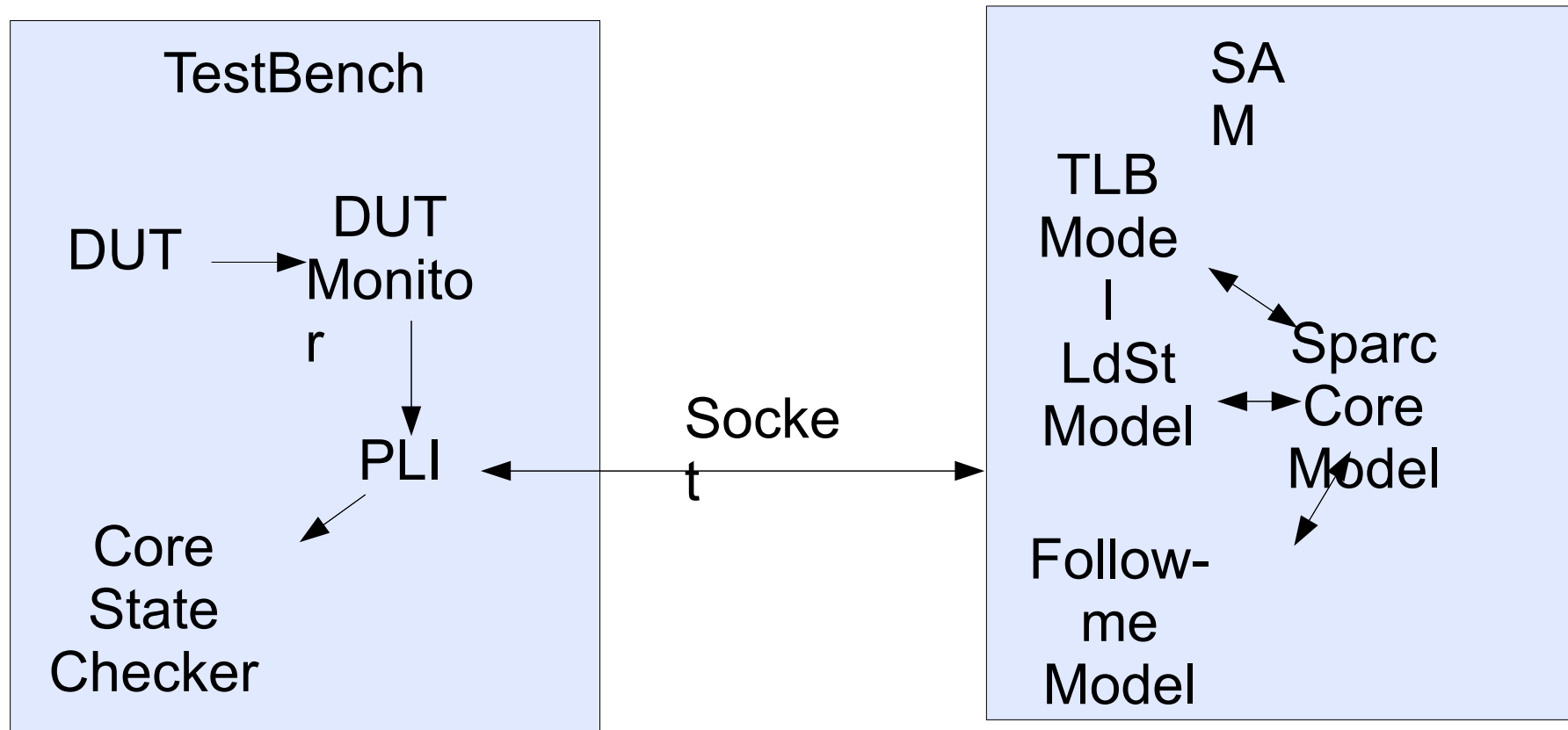
Functional Simulator

- No timing information
- Behavior will not match RTL exactly because of timing issues
 - > TLB replacement
 - > Deferred and disrupting traps
 - > Memory barrier
- No cache

RTL Co-simulation

- Verify RTL by concurrently running the same instructions in RTL and SAM, and cross-checking architectural state and memory instr-by-instr
- RTL overrides SAM's behavior to correct timing mismatches
 - > TLB replacement
 - > Deferred and disrupting traps
 - > Memory barrier
 - > Execution latency, e.g., crypto operations
- Co-simulation is run continuously through revenue release
 - > High level of fidelity to hardware
 - > Direct diags and random diags, comprehensive coverage

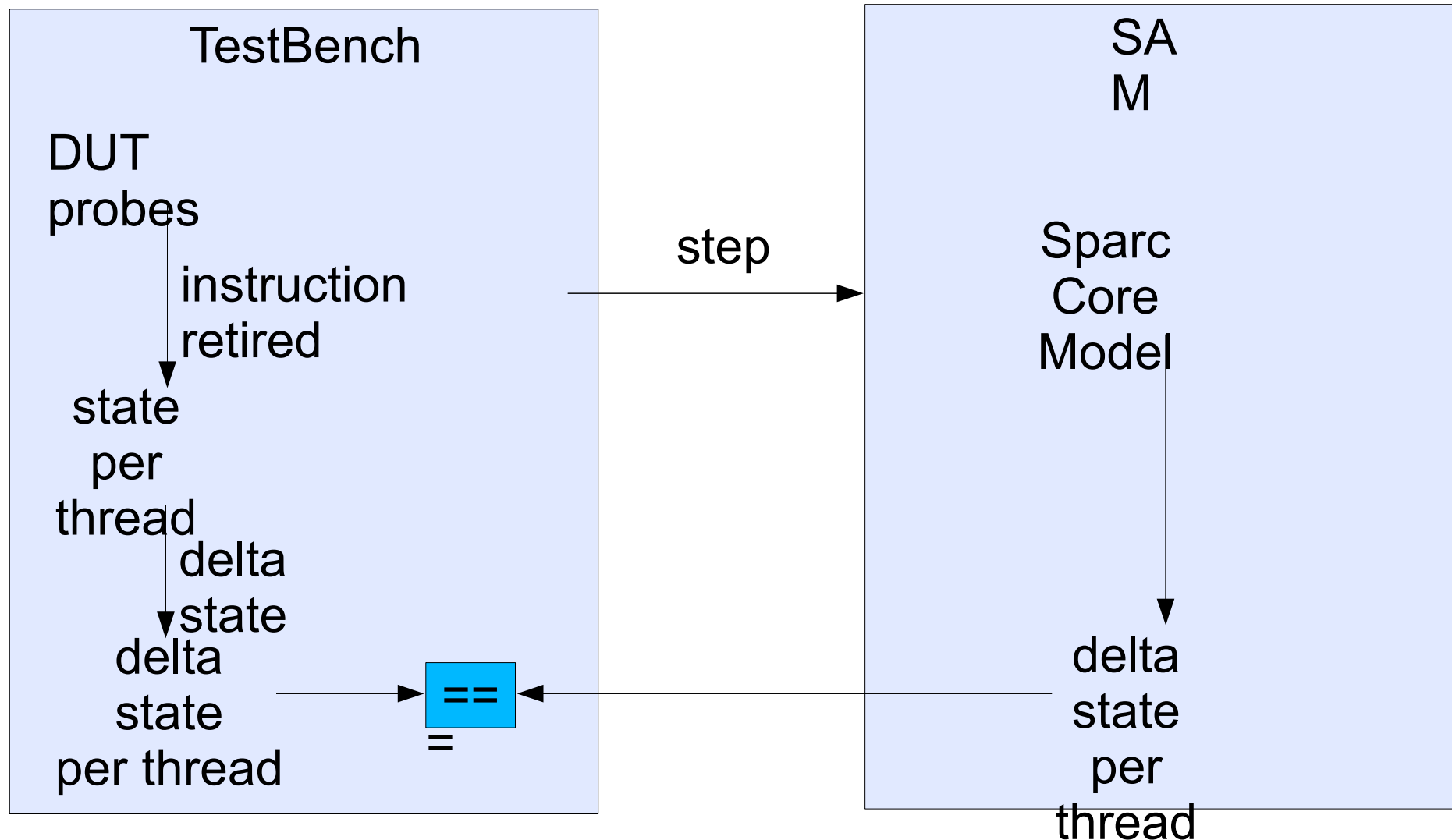
RTL Verification Co-simulation



* DUT: device under test, i.e.,

RTL

RTL Verification State Checking



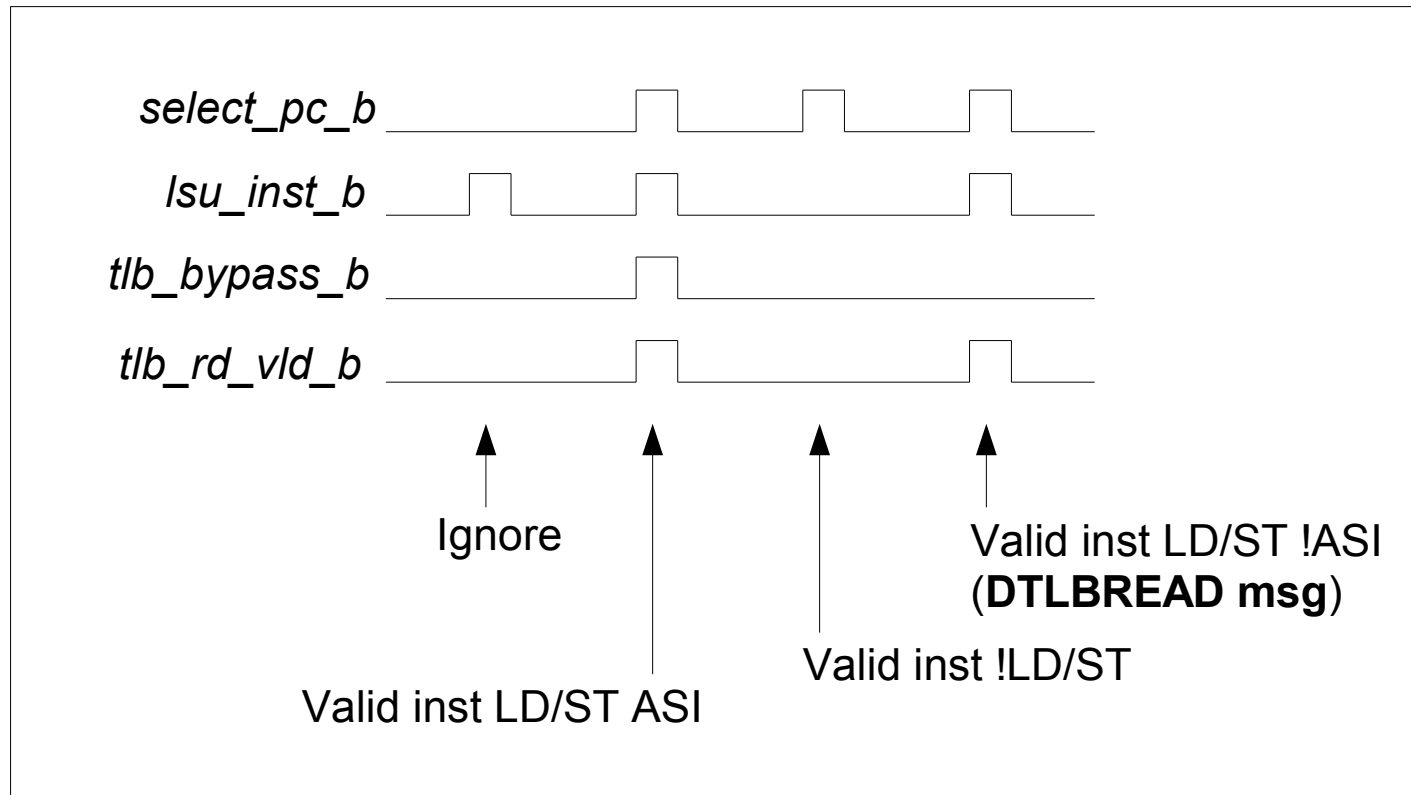
Interrupt-sync Model & Follow-me

- Handle precise traps independently
- Sync up disrupting events
 - > Resets
 - > Interrupts
 - > Error conditions
- Update architecture state
- Post proper trap at the right time
- Architecture state follow-me
 - > ASI, ASR, CMP
 - > I/O CSR

TLB-sync Model

- TLB state out of sync factors
 - > Multi-threading
 - > Post translation instruction buffer
 - > Pick/trap re-ordering
- Hardware table walk de-coupling
- Sample commands
 - > i/d-tlb-read
 - > i/d-tlb-write
 - > i/d-hwtw

DTLBREAD



DTLBREAD

- Bench has state machine to watch the following DUT signals to determine when to send *DTLBREAD* msg:
 - > *select_pc_b*: Instruction in B stage of pipeline. Already includes flush signals.
 - > *lsu_inst_b*: Instruction in B stage is a LD/ST. Must qualify this with *select_pc_b*
 - > *tlb_bypass_b*: Asserted when the instruction did not do DTLB lookup. ASI or Hypervisor mode.
 - > *tlb_rd_vld_b*: Required to qualify *tlb_bypass_b* to make sure that tlb is enabled.

TLB-sync Example

Without TLB-sync

RTL

T0 ifetch itlb#1 instr1 ts1

T1 replace itlb#1 with itlb#1-new ts2 --->

T1 ifetch itlb#1-new instr2 ts3

T1 retire instr2

T0 retire instr1

NAS

T1 replace itlb#1 with itlb#1-new

---> T1 exec instr2, with itlb#1-new

---> T0 exec **instr?**, with itlb#1-new

TLB-sync Example ...

Without TLB-sync

RTL

T0 ifetch itlb#1 instr1 ts1

T1 replace itlb#1 with itlb#1-new ts2 --->

T1 ifetch itlb#1-new instr2 ts3

T1 retire instr2

T0 retire instr1

NAS

T1 replace itlb#1 with itlb#1-new

---> T1 exec instr2, with itlb#1-new

---> T0 exec **instr?**, with itlb#1-new

With TLB-sync

RTL

T0 ifetch itlb#1 instr1 ts1

T1 replace itlb#1 with itlb#1-new ts2 --->

T1 ifetch itlb#1-new instr2 ts3

T1 retire instr2

T0 retire instr1

NAS

---> **T0 itlb-read ts1**

---> **T1 itlb-write itlb#1-new ts2**

---> T1 replace itlb#1 with itlb#1-new

---> **T1 itlb-read ts3**

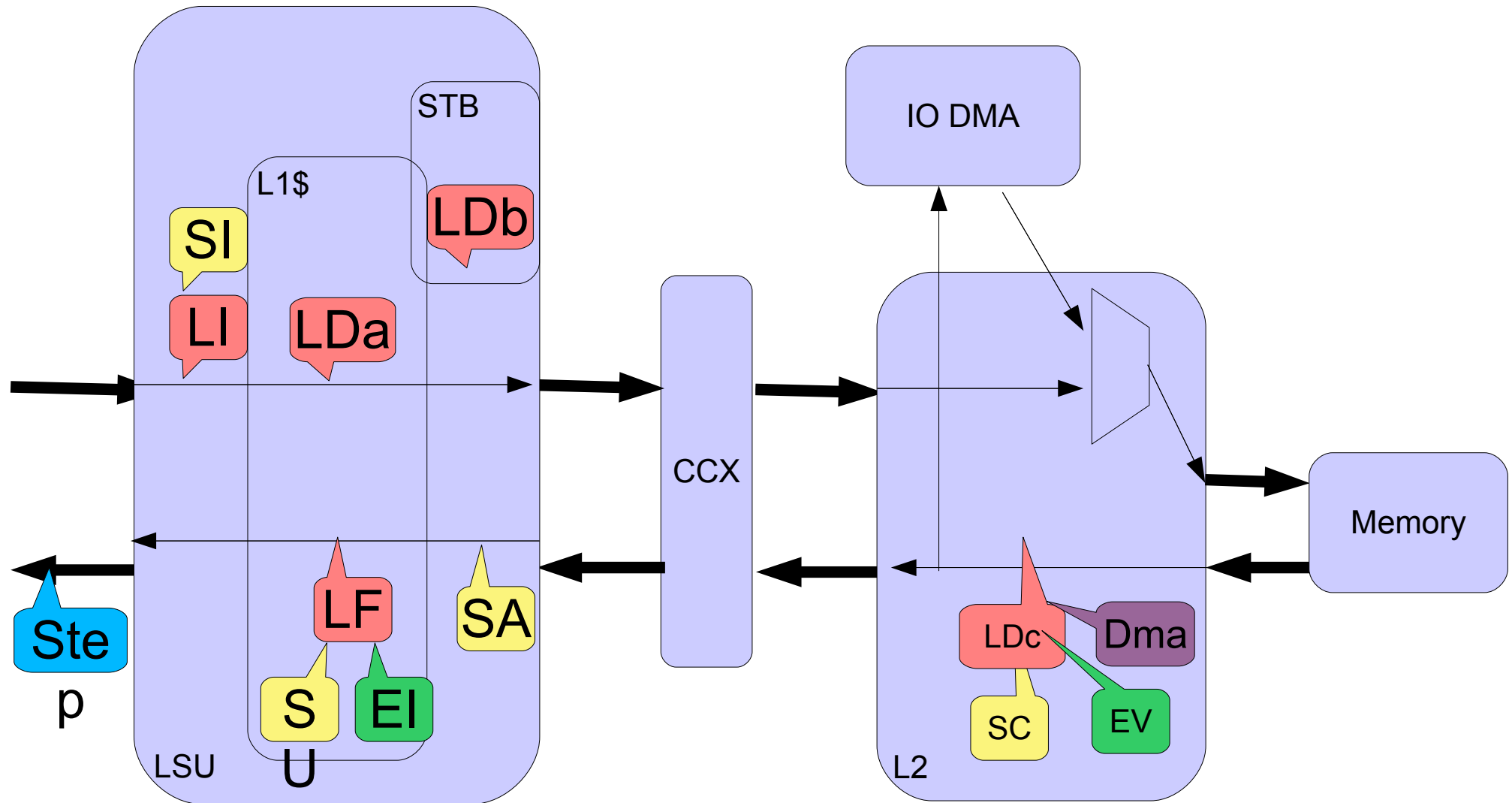
---> T1 exec instr2, with itlb#1-new

---> T0 exec instr1, with itlb#1

LdSt-sync Model

- Mimic store buffer, dCache, cache coherence, without a real cache model
- Maintain proper load & store order
- Sample commands
 - > load-issue, load-data, load-fill
 - > store-issue, store-commit, store-update, store-invalid
 - > evict, evict-invalid

RTL Bench Probe Points



Load Messages

- LoadIssue (LI)
 - > When the Load is issued to LSU. For RAW case.
- LoadData (LDa,LDb,LDc)
 - > When the Load gets its data
 - > LDa - Load Hit
 - > LDb - Load RAW Store
 - > LDc - Load Miss
- LoadFill (LF) (optional)
 - > When the Load Data is seen in the L1\$
- SSTEP

Store Messages

- StoreIssue (SI)
 - > When a Store is issued to LSU. For RAW case.
- SSTEP
 - > Step for a Store happens before it commits to L2\$.
- StoreCommit (SC)
 - > When a Store is in L2\$ and visible to all Load Misses.
- StoreAck (SA)
 - > When a Store is Acked, removed from STB
- StoreInvalidate/StoreUpdate (SU) (Optional)
 - > When Store data or invalidate is seen in the L1\$.

Eviction Messages

- Eviction (EV)
 - > When a line is evicted from L2\$
- EvictInvalidate (EI)
 - > When Eviction invalidation is seen in L1\$

LdSt-sync Example

Without LdSt-sync

RTL

T0 store addr1 data1

T8 load addr1 l1\$-hit (data0) --->

C1 store-invalid addr1

T8 load addr1 l1\$-miss (data1) --->

NAS

T0 step instr store addr1 data1

T8 step instr load addr1 (data0)

T8 step instr load addr1 (data1)

LdSt-sync Example ...

Without LdSt-sync

RTL

T0 store addr1 data1

T8 load addr1 l1\$-hit (data0)

C1 store-invalid addr1

T8 load addr1 l1\$-miss (data1)

NAS

T0 exec instr store addr1 data1

T8 exec instr load addr1 (data0)

T8 exec instr load addr1 (data1)

With LdSt-sync

RTL

T0 store addr1 data1

T8 load addr1 l1\$-hit (data0)

C1 store-invalid addr1

T8 load addr1 l1\$-miss (data1)

NAS

T0 exec instr store addr1 data1

T0 store-commit addr1

T8 load-data addr1 l1\$-hit

T8 exec instr load addr1 (data0)

C1 store-invalid addr1

T8 load-data addr1 l1\$-miss

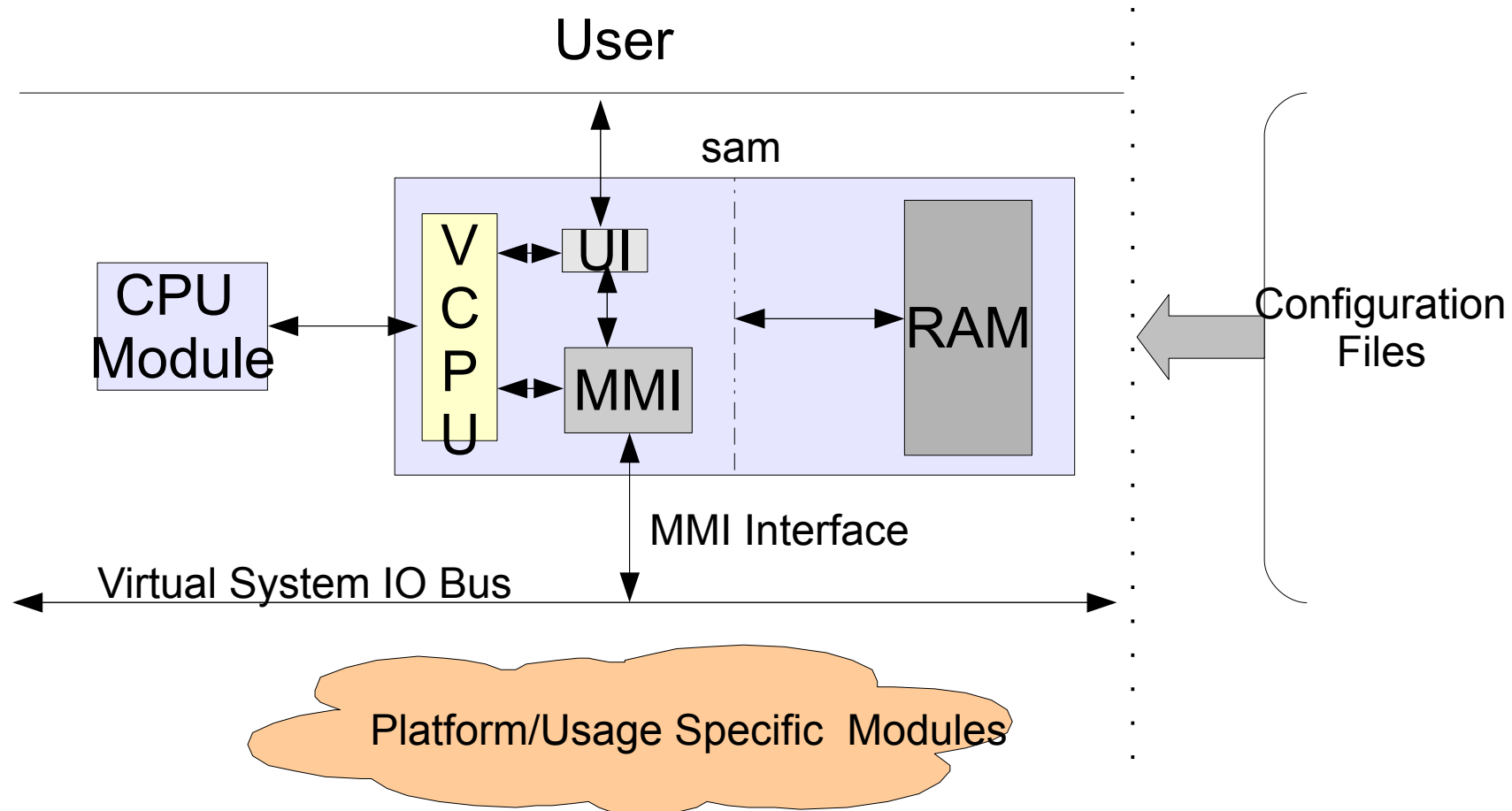
T8 load-fill addr1

T8 exec instr load addr1 (data1)

Full-system Simulation

- System framework with cpus and devices
- Use the same cpu module as in RTL verification, same high degree of fidelity
- Collection of dynamically loadable device modules
- Runtime configuration
- Checkpoint and restart
- Benchmark tracing
- Availability
 - > Full-system simulation available on SPARC platform
 - > RTL co-simulation available on SPARC and x86 platforms

Full-system Simulation



VCPU

- Virtual CPU Model Interface (VCPU)
 - > Stepping CPU's by instruction count
 - > Setting and taking breakpoints
 - > Sending IO Interrupts
 - > Loads/Stores to Registers and Memory
 - > UI access to Registers
 - > Time management
- Connects various cpu models to SAM

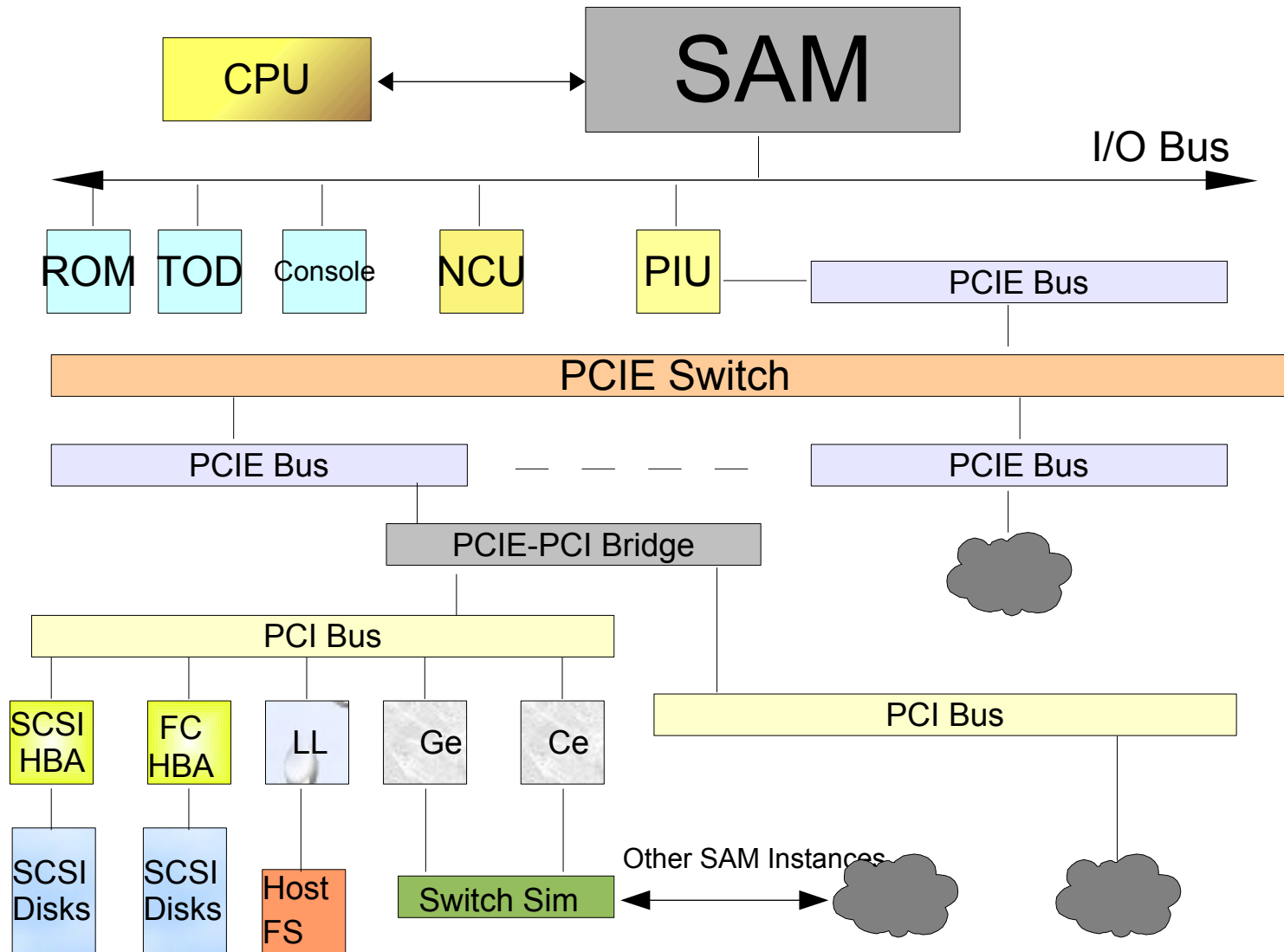
MMI – Modular device Model Interface

- Standard API for loadable device Models
 - > Read/Write Access to Physical Memory
 - > Registration of Address Ranges in I/O Space
 - > Interrupt Delivery and Timing
 - > User Interface Command Registration
 - > Device Creation and Destruction
- Supports Exchange of Device Models

SAM UI

- Rich set of basic intuitive commands
 - > run, stepi, stop – start/stop simulation
 - > read-reg, write-reg, %*reg-name* – register access
 - > break, probe – set breakpoint, execution probe
 - > dump – creates checkpoint (not available for all configs)
- Simple, consistent syntax
- Extensible: MMI modules and python scripts
 - > MMI modules can add their commands
 - > Commands may call python scripts

Sample System Config



Configuration files

- Specify the simulator configuration parameters, e.g.
 - > Simulated RAM size and MIPS, number of CPUs
- Device/CPU models
- Device tree specification
 - > Somewhat mimics Solaris device tree
 - > SAM module not always a Solaris visible device
- Configuration parameters for each loaded module
- Miscellaneous
 - > Platform/environment specific parameters
 - > Setting debug levels

Sample Configuration

- `conf ramsize 64M`
- `sysconf cpu name=cpu0 cpu-type=SUNW,UltraSPARC-T1`
- `sysconf cpu name=cpu1 cpu-type=SUNW,UltraSPARC-T1`
- `sysconf dumbserial serial1 type=GUEST startpa=0x9f10000000
endpa=0x9f1000004f`
- `load bin disk.s10hw2 0x1f40000000`
- `load bin 1up-hv.bin 0x1f12080000`
- `load bin 1up-md.bin 0x1f12000000`
- `load bin reset.bin 0xfff0000000`
- `sysconf ll ll0 bus=pcia dev=1 fun=0`
- `setreg pc 0xfff0000020`
- `setreg npc 0xfff0000024`

Vtrace – Tracing Interface

- Allows tracers to dynamically link with SAM
- Notifies tracers of simulation events
 - > Instruction execution
 - > DMA activity
 - > Traps, interrupts, TLB update etc
- Broad Application
 - > SAM RST tracer – instruction and event traces
 - > Trace data is compressed on-the-fly
 - > Trace data as input for performance models (pipeline, cache, etc)
 - > Architectural analyzer – chip verification coverage

Trace Generation in SAM

- stop
- mod load analyzer rstracer.so
- rstrace -o output-file -n #instr
- stepi N
- rstrace off
- mod unload analyzer

Trace Analysis

- trconv, rst trace data processing tool
- Process data by
 - > cpu#
 - > record range
 - > PC/EA range
 - > summary
- Use together with rstunzip to process trace data

Sample Trace Records

- `rstunzip trace_file | trconv -c`

Counted: 20451321 records
9999998 instruction recs
1 header recs
1 traceinfo recs
1071 tlb recs
...
2288 trap recs
...
8687727 regval recs
...

Sample Trace Records ...

- `rstunzip trace_file | trconv -i`

RST trace format (stdin)

=====

Rec #	Type	User/ Priv	PC	Disassembly	Branch Taken	EA
49	instr	: cpuid=0 p	[0x0000000001063254]	or %o2, 0x80, %o1		
67	instr	: cpuid=0 p	[0x0000000001063258]	stb %o1, [%o0 + 0x4b]		[0x0000070002503e4b]
68	instr	: cpuid=0 p	[0x000000000106325c]	call 0x1069764	T	[0x0000000001069764]
71	instr	: cpuid=0 p	[0x0000000001063260]	ldx [%fp + 0x7f7], %o0		[0x000000000180b7b8]
73	instr	: cpuid=0 p	[0x0000000001069764]	jmp %o7 + 8, %g0	T	[0x0000000001063264]

Local Loopback File System (llfs)

- A special module to connect simulator with outside world
- Within simulator, /ll/root represents the root of the real file system of the underlying machine.
- For example /ll/root/home/johndoe in simulator is /home/johndoe at outside world
- Write to a file under /ll/root have immediate impact on the real file system

Modify Disk Image

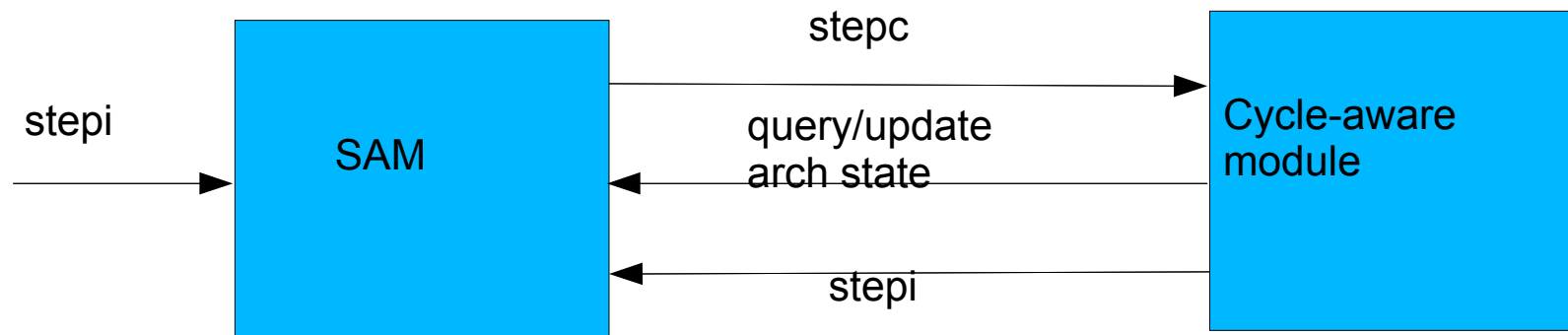
- Use lofiadm to add/remove/modify files in disk image
 - > with root access
 - > lofiadm -a /absolute-path/disk_image
 - > (assume /dev/lofi/1 is the returned value)
 - > mount /dev/lofi/1 /mnt
 - > cd /mnt
 - > add/remove/modify files in /mnt
 - > umount /mnt
 - > lofiadm -d /dev/lofi/1

Save Modified Disk Image

- During initialization
 - > UI(load): loading <disk1> memory image
 - > loading disk1, base addr 0x1f4000000, size 0x200000
- Sync up file system first
 - > sync
 - > halt
 - syncing file system... done
 - Program terminated
- stop
- memdump new_disk 0x1f4000000 0x200000
- use lofiadm/mount to examine files

Correlate With Cycle-Aware Module

- Use SAM's loadable module mechanism
- Create callback functions between modules
- Cycle-aware module maintains cycle-related state



Software Development Tool

Legion

- Full-system simulator for firmware and software development
- Implement enough architecture state to boot up Solaris
- Share the same disk image and binary files (Hypervisor/OBP/reset/etc) with SAM
- Startup script `run_legion.sh`
- Available configuration: 1-thread, 2-thread, 32-thread, 64-thread (T2)
 - > `run_legion.sh 1 / 2 / 32 / 64 [options]`

Legion Runtime Options

- Available options
 - > -debug debug_bits
 - > -t #physical_cpu
 - > -h
- Debug options
 - > 0x2: PC & instruction
 - > 0x8000: hypervisor calls
 - > 0x20000: exceptions, XIR
 - > 0x100000: TLB miss
 - > 0x400000: trap, TSTATE
 - > etc

Legion Runtime Display

- Use ~ (tilde) on 'guest console' window to dump out architecture state
 - > ~z: exit simulation
 - > ~i: dump I-TLB content
 - > ~d: dump D-TLB content
 - > ~b: toggle debug enable bits
 - > etc



OpenSPARC™

OpenSPARC Simulators

Stephen Henry

Sr. Staff Engineer

Architecture Technology Group

Microelectronics

Sun Microsystems