RST Specification v0.1.1 #5

Craig Anderson

May 7, 2004

The purpose of this document is to specify the correct format of non-value RST files as generated by the `blaze` simulator and related trace tools. While the `rstf.h` file describes the syntax of the individual RST records, this document specifies correct *sequences* of records, and additional information not included in the `rstf.h` file.

Because `shade` does not simulate system level events, traces generated by `shade` will not contain some records required in this specification. Shade traces should conform to this specification to the extent possible.

1. General
   (a) All fields in RST structures named "notused" or "reserved" or variants thereof are reserved for future use and have undefined contents. Any software which consumes RST traces shall not contain references to such fields. All software that produces traces shall set such fields to `0`. Setting unused fields to 0 may help trace compression.
2. Beginning of the Trace
   (a) The first record in a trace shall be a `RSTHEADER_T` record. It shall contain the major and minor version of the rst header file which the trace corresponds to. It also will have `header_str` which will start with `RSTF_MAGIC`.
      i. No `RSTHEADER_T` record will appear after the first record in the file.
   (b) The second record shall be of type `TRACEINFO_T`, with a subtype of `RST_TI_INFO_1`.
   (c) *(New Requirement)* The third record shall be of type `TRACEINFO_T`, with a subtype of `RST_TI_INFO_2`.

```
typedef struct {
       uint8_t   rtype;
       uint8_t   rtype2;
       uint8_t   max_cpu_id;
       uint8_t   total_num_cpus;
       uint8_t   major_rst_trace_version;
```

```
        uint8_t   minor_rst_trace_version;
        uint8_t   sub_minor_rst_trace_version;
        uint8_t   reserved;
        uint64_t  date_created;
        uint64_t  date_modified;
} rstf_traceinfo_3T
```

The fields are as follows:

- max_cpu_id: The maximum value of the `cpuid` field in `INSTR_T` records.
- total_num_cpus: The total number of unique `cpuid` numbers in `INSTR_T`. This must be less than or equal to `max_cpu_id`.
- Major, minor, sub-minor rst trace version: The version of the RST specification with which the trace is compliant.
- date_created: The date (as given by *ctime(1)*) that the trace file was first created.
- date_modified: The date (as given by *ctime(1)*) that the trace file was last modified. If the trace file has not been modified, filtered, or otherwise changed since it was generated, then the date modified should be the same as the date created.

(d) A trace shall specify the MMU type in string record(s). The string shall be of the form:
mmutype=*MMUTYPE* iTLB=*spec1*, dTLB=*spec2*
. and *spec1* and *spec2* are of the form:
(integer)(/ integer)*.
or, in case of unified TLB's:
mmutype=*MMUTYPE* uTLB=*spec1*. Note that some configurations may not be describable using this notation. Some trace generators (notably Shade) model only user-level code and are exempt from this requirement. Some example lines are:

- mmutype=cheetahplus iTLB=16/128, dTLB=16/512/512
- mmutype=spitfire iTLB=2048, dTLB=2048

(e) *(New)* A trace shall specify the SPARC implementation type in string record(s). The string shall be of the form:
cputype=*IMPLEMENTATION impl_type*. Some sample implementation types are: Some example lines are:

- cputype=spitfire
- cputype=blackbird
- cputype=cheetah
- cputype=cheetahplus
- cputype=millenium
- cputype=niagra

- cputype=rock

(f) A trace shall provide the initial values of:

- Trap level (tl). If the trap level is not zero, then the current trap type, TPC, and TNPC shall be specified.
- The processor state register (pstate)
- The address space identifier (ASI) register
- The number of register windows. Currently, this is done in a string "nwins=*integer*".

(g) Before the first `INSTR_T` record, there shall be `TLB_T` records sufficient to completely specify the initial contents of all TLBs in the system. *Warm-up records can be substituted for TLB records.*

(h) There shall be an initial `PAVADIFF_T` record.

3. Mid-trace requirement

(a) All `STRCONT_T` records must be followed by either another `STRCONT_T` record or a `STRDESC_T` record.

(b) Trace generators must output records for annulled instructions – they may not be omitted. Annulled records have `cpuid`, `pc_va`, and `instr` valid. `Ea_valid` is 0.

(c) A `PAVADIFF_T` record shall be emitted when:

- Immediately before (i.e. the previous record) an instruction record in which the PC maps to a virtual page different than the previous instruction record from the corresponding cpu, and the difference between the PA and VA differs from the previous PA/VA difference.
- Immediately before an instruction record which is a load/store/atomic instruction and the target of the instruction is on a different virtual page than the previous load/store/atomic instruction from the corresponding processor and the difference between the PA and VA differs from the previous PA/VA difference. The ea_valid bit must be 1.
- *(New?)* When the i or d context changes. The ea_valid bit must be 1 when a d context changes.
- If a `PAVADIFF_T` record is emitted immediately preceding an instruction for which ea_valid is zero, the ea_valid bit in the `PAVADIFF_T` record must also be 0.

(d) Redundant `PAVADIFF_T` records are allowed.

(e) The trace may not contain any records not specified by the version of the rstf.h file specified in the trace's header.

(f) `TRAPEXIT_T` records are recommended but not required in a trace.

(g) If the AM bit in the `pstate` register is one, all virtual addresses are limited to 32 bits. The `ea_va` the `INSTR_T` record must contain a 32-bit value; namely the upper 32 bits of the 64-bit `ea_va` be zero.

4. Trap Behavior

(a) Every instruction causing a synchronous trap shall contain `tr = 1`.

(b) Updates of the TL register by `wrpr %tl` instructions should NOT trigger a trap entrance or trap exit sequence of records.

(c) When a system performs a trap in hardware, the contents of the corresponding trace are currently unspecified.

(d) Trap entrance
The trap record that is part of the start of a trap entrance has the following fields and values:

is_async Has a value of 1 if the trap is asynchronous.

tl The trap level BEFORE the trap is taken.

cpuid A value valid for this trace (see above).

ttype A valid trap type value.

pstate The pstate value BEFORE the trap is taken.

syscall If trap was caused by a system call, this contains a valid system call number. Otherwise, it is 0.

pc The PC value BEFORE the trap.

npc The NPC value BEFORE the trap.

- Asynchronous traps
  When an asynchronous trap occurs, the following records are emitted in the specified order:
  i. A `TRAP_T` record, with `async == 1`.
  ii. A `PAVADIFF_T` record (if needed).
  iii. An instruction record corresponding to the first instruction of the asynchronous trap handler. The `tr` bit is set to `1`.
- Synchronous traps, except for ITLB traps
  When a synchronous trap occurs, the following records are emitted in the specified order:
  i. A `TRAP_T` record, with `async == 0`.
  ii. An instruction record corresponding to the instruction which caused the trap. The `tr` bit is set to `1`.
- ITLB traps
  i. A `TRAP_T` record, with `async == 0`.
  ii. A `PAVADIFF_T` record.
  iii. An instruction record corresponding to the first instruction in the ITLB trap handler. The `tr` bit is set to `1`.

Note that no instruction record is emitted for the instruction fetch which caused the trap.

(e) The instruction record which caused the DTLB miss has the trap bit (`tr`) set.

(f) Trap exit

- Trap exit is only triggered by done or retry instructions.
- Done and retry instruction records must be preceded by a `PREG` record. That record contains the following information:
  - `traplevel`: The trap level *before* the retry/done instruction.
  - `pstate`: The pstate register *after* the retry/done instruction.
  - `traptype` Shall be the trap type that the processor is exiting.
  - `asiReg, cpuid` Must be valid.
- *optional* Done and retry instruction records should be followed by a `REGVAL_T` record that contains the values of TL, TT, TPC, and TNPC that are effective AFTER the done/retry completes. followed by a `REGVAL_T`

5. Multiprocessor trace specifications

   (a) The interleaving of records from different CPUs in a trace is not defined. However, some versions of the rstzip program cannot handle large numbers of contiguous records from the same CPU. Version three of the rstzip program does not have this limitation.

6. Multifile traces If a trace is written to more than one file in a serial manner, the concatentation of the files must yield a valid trace with no dropped records "between" the files.

7. Record specifications
   As described earlier, the `rstf.h` file specifies the format of individual records while this document describes higher level syntax and sequencing of records. However, there are some aspects of the individual records that are not specified in the `rstf.h` that nonetheless need to be defined. In the following section, we elaborate on the definitions of a few of the RST records. If the comments in the `rstf.h` file and this file differ, then one or both of the files needs to be corrected.

   (a) `INSTR_T` record specification:

   - `ea_valid:` For instructions which conditionally or unconditionally load or store a value to memory, or a control transfer instruction, `ea_valid` is 1 if and only if the `ea_va` field in the record is valid. This must be true even if the `an` field is 1. Instructions that cause synchronous traps (other than ITLB miss traps) have `ea_valid` equal to 1. For all other instructions, `ea_valid` is 0. `EA_VALID` must be 1 for all branches and all non-asi load/stores/atomics that have both `an` and `tr` equal to 0. If

the instruction record incurs a data MMU miss, the `EA_VALID` shall be set to 0.
*(New)*Information about the EA that caused the miss, or a non-translateable address (eg internal ASI) will be supplied using a SPECIALVAL or REGVAL record that occurs BEFORE the instruction record but AFTER the trap record corresponding to the trapping instruction. Note: it is not clear why we have `ea_valid` be one for control transfer instructions.

- `tr`: Broadly, the value is 1 if a trap occurred:
  - In the middle of this instruction (e.g. a load that takes DTLB trap), or
  - At the end of this instruction (e.g. TRAP instruction), or
  - This is the first instruction of a trap handler (e.g. ITLB trap handler)

  Note that an instruction record with `tr == 1` does not always mean that the instruction was not executed to completion.
- `bt`: For control transfer instructions, `bt` is 1 if the control transfer occurred, and 0 if it did not. Control transfer instructions include all conditional and non-conditional branches, trap instructions, and the `done` and `retry` instructions. In addtion, all instructions which cause a synchronous trap to occur (except for ITLB misses) also have `bt` equal to one. For all other instructions, `bt` is 0.
- `cpuid`: has a value between *0* and *MAX_INSTR_CPUID -1*, inclusive.
  Note: This field is only 6 bits wide, which is likely to be insufficient in the future. Solutions may include using the two unused bits in the record (which are not contiguous with the cpuid field or each other) or the ihash field.
- `ihash`: The content of the `ihash` field is not defined in the rstf.h file. However, existing software assumes that the field contains the decoded instruction as returned by a call to `spix_sparc_iop()`.
- `instr`: The instruction in 32 bit machine code form.
- `pc_va`: The virtual address of the PC corresponding to the current instruction
- `ea_va`: For load/store/atomic instructions, the virtual address of the target of the i/o instruction. For control transfer instructions, the value of NPC, which is either PC+8 or the branch target. For all other instructions, the value is 0.

(b) `PREG_T` record specification:

- `PREG_T` records contain the *current* values of the specified registers.
- The above does not apply on trap entrance and trap exits.

(c) `REGVAL_T` record specification:

- All values specified by a `REGVAL_T` record are the values after all actions corresponding to records before the `REGVAL_T` record have completed, and before any record coming after the `REGVAL_T` has taken effect. [XXX This is not true in current traces - see `REGVAL_T` records that occur before a `done` instruction].

Here are two tables detailing the records seen on trap entry and exit.

Table 1: Trap Entry Sequence

Record Type/Fields	Asynchonous Trap	Synchronous Trap	ITLB Miss
TRAP_T (first record)	TL: Previous Trap Level TT: This trap type PSTATE: state BEFORE trap PC= NPC=	Same	Same
INSTR_T (second record)			
Trap Bit	TR=1	TR=1	TR=1
Contains	First instruction of trap handler	Trap causing instruction	First instruction of handler
Instruction	IS executed	NOT executed	IS executed

Table 2: Trap Exit Sequence

Record Type/Fields	Asynchonous Trap	Synchronous Trap	ITLB Miss
REGVAL_T (first record)	%g7 = value AFTER done/retry %sp = ???	Same	Same
PREG_T (second record)	TL:trap level BEFORE done/retry TT: trap type BEFORE done/retry PSTATE: state AFTER done/retry MMU context: current	Same	Same
INSTR_T (third record)	Done/Retry Instruction	Same	Same