UltraSPARC-IV+ Errata

Part No. 820-4011-10 *12/12/07*

Sun Microsystems, Inc.
www.sun.com

Submit comments about this document at: `http://www.sun.com/hwdocs/feedback`

Please Recycle

Adobe PostScript

UltraSPARC-IV+ Errata - 12/12/07
Part No. 820-4011-10

1. *Errata Table*

Table 1: UltraSPARC-IV+ Errata Table

Errata Number	Version 2.1	Version 2.2	Version 2.4	See ...
1	✔	Not applicable	Not applicable	page 4
2	✔	✔	✔	page 9

2. Detailed Errata Summary

Erratum #1: A subset of L2 and L3 Cache ECC errors are reported incorrectly if the transaction detecting the error is processed in a narrow window of time along with an unrelated noncacheable transaction.

Applicability:

UltraSPARC-IV+ Version 2.1.

Description:

A class of L2 and L3 Cache ECC errors will not be reported correctly when detected in a sensitive timing window if a noncacheable transaction (e.g., load, store, and instruction fetch) is pending for processing. The noncacheable transaction in question may be pending from either core.

The error types affected are noted below, referenced by the Asynchronous Fault Status Register (AFSR) and Asynchronous Fault Status Extension Register (AFSR_EXT) registers. Those not explicitly called out are unaffected.

Impact:

There are two subsets of this behavior. In both cases, the AFSR, AFSR_EXT, and Asynchronous Fault Address Register (AFAR) registers will not contain any information about the error. Additionally, in the case that an error has already been captured in the AFSR or AFAR registers, this behavior will prevent the secondary AFSR, AFSR_EXT, and AFSR registers from recording the first error.

In the case of errors that result in precise traps, these traps will be taken and software will have to handle this behavior in the absence of additional information.

For other error responses, not only are the AFSR and AFAR registers not recorded, but the error response themselves do not occur (i.e., they are silent). This includes a subset of errors that are reported in the following ways:

- ERROR pin assertion
- Disrupting trap
  In most cases of this type, although the error response will not occur, there is no data corruption. Unfortunately, there are a small number of cases that will result in possible data corruption.

- Deferred trap

Both Correctable (CE or single-bit) and Uncorrectable (UE or double-bit or more) L2 and L3 Cache errors types are affected.

Additional Information:

All of the CE types are corrected by hardware before the data is delivered to the requester and therefore, will not result in system problems as good data is always used. L2 Cache Tag CEs are treated somewhat differently as the hardware also rewrites the L2 Cache Tag with the corrected data and ECC. As noted in the table below, L3 Cache Tag errors, either CE or UE, are not affected.

Of the UE types, L2 cache data UE for a copyout (CPU) and L2 cache data UE error for a writeback (WDU) errors do not result in data corruption, as the data that is copied or written back, respectively, will be marked as having had an uncorrectable error, either by keeping the problematic data and ECC combination intact or by “poisoning” the data by forcing it to be transported with a UE.

In addition, nearly all of the UE cases encompassed in the L2 cache data UE for a store, block load, or prefetch queue operation (EDU) and L3 cache data UE for a store, block load, or prefetch queue operation (L3_EDU) do not result in data corruption, except in the case of block loads. This is because the 8 floating point register file entries used as the destination of the block load will be updated silently with incorrect data. Software prefetches or store misses, which also are reported as EDU and L3_EDU, if they encounter a UE, will not result in data corruption. In the first case, the prefetch data is simply dropped and never used. In the second case, the data is “poisoned” so that an error will be seen the next time it is accessed.

Lastly, and most significantly, the L2 cache tag UE (TUE) and L2 cache tag UE from a foreign Fireplane device (TUE_SH) error types are affected. As these errors would normally result in the assertion of the ERROR pin as they may result in a loss of system cache coherency, there are the most catastrophic. Fortunately, only the smaller L2 Cache Tag suffers any exposure to this corner case -- the much bigger L3 Cache Tag is not affected.

The tables below indicate the AFSR and AFSR_EXT error types that are affected. Bits that specifically called out represent error types that are not affected.

Table 2: Asynchronous Fault Status Register (AFSR) Bits Affected

AFSR Bit #	Error Type	Error Indication	Description
62	TUE_SH	None	TUE_SH is the L2 cache tag UE from a foreign Fireplane device. This error means there was no disrupting trap and no ERROR pin pulled. Coherency was lost.
57	THCE	None	THCE is the L2 cache tag CE. This error means the hardware-corrected L2 tag was supplied and corrected within the L2 Cache tag array. No data corruption occurred.
55	TUE	None	TUE is the L2 cache tag UE. This error means there was no disrupting trap and no ERROR pin pulled. Coherency was lost.
42	UCC	Precise Trap	UCC is the L2 cache CE for an instruction fetch, load-like, or atomic instruction. This error means the fast_ECC_error trap signalled, although both the AFSR and AFSR are empty.
41	UCU	Precise Trap	UCU is the L2 cache UE for an instruction fetch, load-like, or atomic instruction. This error means the fast_ECC_error trap signalled, although both the AFSR and AFSR are empty.
40	CPC	None	CPC is the L2 cache data CE for copyout. This error means hardware-corrected data was supplied. No data corruption occurred.
39	CPU	None	CPU is the L2 cache data UE for copyout. This error means the data poisoned so that the error will be visible on a subsequent access. No data corruption occurred.
38	WDC	None	WDC is the L2 cache data CE for writeback. This error means hardware-corrected data was supplied. No data corruption occurred.

Table 2: Asynchronous Fault Status Register (AFSR) Bits Affected (Continued)

AFSR Bit #	Error Type	Error Indication	Description
37	WDU	None	WDU is the L2 cache data UE error for a writeback. This error means the data poisoned so that the error will be visible on a subsequent access. No data corruption occurred.
36	EDC	None	EDC is the L2 cache data CE for a store, block load, or prefetch queue operation. This error means hardware-corrected data was supplied. No data corruption occurred.
35	EDU	None	EDU is the L2 cache data UE for a store, block load, or prefetch queue operation. This error means the following: • Block load: Bad data was written into the floating point register file. Data corruption occurred. • Software prefetch: Bad data was dropped. No data corruption occurred. • Store miss/atomic: The data was poisoned so that the error will be visible on a subsequent accept. No data corruption occurred.

Table 3: Asynchronous Fault Status Extension Register (AFSR_EXT) Bits Affected

AFSR_EXT Bit #	Error Type	Error Indication	Description
5	L3_UCC	Disrupting Trap	L3_UCC is the L3 cache CE for an instruction fetch, load-like, or atomic instruction. This error means the ECC_error trap signalled, although both the AFSR and AFSR are empty.
4	L3_UCU	Disrupting Trap	L3_UCU is the L3 cache UE for an instruction fetch, load-like, or atomic instruction. This error means the ECC_error trap signalled, although both the AFSR and AFSR are empty.
7	L3_EDC	None	L3_EDC is the L3 cache data CE for a store, block load, or prefetch queue operation. This error means hardware-corrected data was supplied. No data corruption occurred.
6	L3_EDU	None	L3_EDU is the L3 cache data UE for a store, block load, or prefetch queue operation. This error means the following: • Block load: Bad data was written into the floating point register file. Data corruption occurred. • Software prefetch: Bad data was dropped. No data corruption occurred. • Store miss/atomic: The data was poisoned so that the error will be visible on a subsequent accept. No data corruption occurred.

Notes:

For all cases where an L3 Cache access resulted in a UE and data is moved from L3 to L2, the data and ECC pair that caused the UE will be written directly into the L2 Cache so that a subsequent access will detect the error.

The following AFSR_EXT bits are not affected:

- L3 cache tag CE (L3_THCE)
- L3 cache tag UE from a foreign Fireplane device (L3_TUE_SH)

- L3 cache tag UE (L3_TUE)
- L3 cache data CE for a copyout (L3_CPC)
- L3 cache data UE for a copyout (L3_CPU)
- L3 cache data CE for a writeback (L3_WDC)
- L3 cache data UE error for a writeback (L3_WDU)

The reason is that the hardware issue responsible for this behavior is confined to a specific functional unit that reports a subset of AFSR-logged errors. The errors listed above are reported from a different functional unit that is not affected.

Workaround:

None

Status:

This bug has been fixed in Version 2.2 of the silicon.

Erratum #2: The branch prediction on instruction refetch can change, causing instructions to be skipped, or IERR to occur with ITQ_IQFIRST_ERR asserted.

Applicability:

UltraSPARC-IV+ Versions 2.1, 2.2, and 2.4.

Description:

The instruction fetch logic can be confused by a changed prediction for a branch in an interrelated series of branches, resulting in a fetch group of 1-4 instructions being left out of the fetch stream.

Impact:

This can appear as missing instructions in the execution stream or as corruption of the IQ-first pointer that will assert internal processor error (IERR) condition ITQ_IQFIRST_ERR. The missed instructions have no particular pattern and may even be from a branch-free linear instruction sequence.

Additional Information:

Branches read their prediction from the Branch Prediction Array (BPA). The location used to read the BPA is computed using a combination of the PC of the branch and the branch history of previous branches. In gshare mode, the BPA read address is calculated as:

```
bpa_radd =    {3'b000,pc[12:2]}  ^  {history[11:0],2'b00}
```

In PC indexing mode, the BPA read address is simply pc[12:2].

The history is the taken or not taken record of the previous twelve conditional branches, where taken=1, not taken=0, and each new conditional branch shifts the previous history left and is recorded as history[0].

During instruction cache fills, the data on the fill bus may include conditional branches. While these branches are not fetched, they trigger a shift of the branch history register. This shift is repaired, but in the failing case, the BPA is accessed before the repair occurs. This results in the BPA read occurring from an incorrect location, while the correct (repaired) history is used for the update to the BPA when the branch resolves.

The following conditions are required for this bug to occur:

1. Two control transfer instructions (hereafter A and B).
2. CTI A is a conditional branch, which must be predicted weakly, but correctly.
3. The lower bits of the PC of CTI A (pc[3:2]) must be the same as pc[3:2] of CTI B. This means the two are a multiple of four instructions apart.
4. CTI A must store a different branch history for updating the BPA than was used to read the prediction from the BPA (as described above).
5. CTI B is a conditional branch in the first fetch group of a line which misses the I Cache but hits the L2 Cache. When the instructions return from L2 Cache and are forwarded to the pipeline, one of these conditions must be true, which will cause the fetch group to be rejected and refetched:
   - The first instruction in the fetch group is annulled (most common).
   - The instruction queue is full.
   - The CTI tracking queue is full.
6. CTI A must resolve during the refetch of CTI B. The branch prediction array index written by this resolve must match the index used by CTI B, and the prediction in this index must be changed by the write (taken to not taken, or not taken to taken). This causes the refetched instance of CTI B to read a different prediction than the first time this CTI was fetched, which confuses the fetch logic.

If condition four is not met (CTI A reads and writes different BPA indexes), then the prediction for CTI B can not change during the refetch. (If the prediction written were different than the prediction read, CTI A would mispredict and flush CTI B from the fetch pipeline.)

Workaround:

Setting the branch prediction mode to PC indexing (DCR.BPM==10_2) prevents the problem from occurring.

Status:

This bug has been fixed as a firmware workaround.