Atmel ATF697FF

Rad-Hard 32-bit SPARC v8 Processor

ERRATA SHEET

Active Errata List

1. Odd-numbered FPU register dependency not properly checked in some double-precision FPU operations
2. Anomaly in instruction cache controller when using cache freeze. (IPN#303)
3. Anomaly when using software traps (IPN #384)
4. Anomaly when using PCI with double bit errors detected inside PCI data packet.
5. FPU exception origin (Aexc and cexc bit fields) are not updated when FPU trap is activated.

Table 1. Errata History

Product Release	Errata List
All ATF697FF part-numbers	1, 2, 3,4,5

Errata Description

1. Odd-numbered FPU register dependency not properly checked in some double-precision FPU operations

Data dependency is not properly checked between a load singleword floating-point instruction (`LDF`) involving an odd-numbered floating-point register as a destination of the load and an immediately following double-precision floating-point instruction (`FADDd`, `FSUBd`, `FMULd`, `FDIVd` or `FSQRTd`) that satisfies all of the following conditions:

- the odd-numbered floating-point register is used as (part of) a source operand
- the destination floating-point register is also a source operand
- in an `FSUBd` or `FDIVd`, the two source operands are different registers

In this case, the final result of the double-precision floating-point instruction will be wrong.

Other double-precision floating-point instructions (`FCMPd`, `FCMPEd`, `FdTOi` and `FdTOs`) are not affected by this issue and will operate as expected.

The error case appears when any of the six following sequences of instructions is present (`n` in [0:31], `x` and `y` as different even numbers in [0:30]):

Case 1:

```
LD [%rn], %fx+1
FPOPd(1) %fx, %fy, %fx
```

Case 2:

```
LD [%rn], %fx+1
FPOPd(1) %fy, %fx, %fx
```

Case 3:

```
LD [%rn], %fx+1
FPOPd(1) %fx, %fy, %fy
```

Case 4:

```
LD [%rn], %fx+1
FPOPd(1) %fy, %fx, %fy
```

Case 5:

```
LD [%rn], %fx+1
FPOPd(2) %fx, %fx, %fx
```

Case 6:

```
LD [%rn], %fx+1
SQRTd %fx, %fx
```

Notes: *1.* *FPOPd* is one of FADDd, FSUBd, FMULd or FDIVd.

2. *FPOPd* is one of FADDd or FMULd (FSUBd and FDIVd operate as expected).

Workarounds

If direct control over assembly language is possible, simply insert a NOP before the double-precision floating-point instruction (case 1 to 6):

LD [%r_n], %f_{x+1}
NOP
FPOPd <same registers set as described above>

If direct control over assembly language is not possible (high-level programming language such as C), checking the SPARC binary code against any of the six above mentioned faulty sequences of instructions shall be done using the code-checker program provided by Atmel (*search for* doc7787 on Atmel web site).

Although there is a very low likelihood of occurrence with high-level programming languages, customers facing this problem should contact the SPARC hotline (sparc-applab.hotline@nto.atmel.com).

2. Anomaly in instruction cache controller when using cache freeze. (IPN#303)

If the instruction cache is frozen and the data streaming option (CCR.ib) is enabled, wrong instructions coming from the cache instruction line can be transmitted to the processor, due to unexpected valid bit activation behaviour. As a result, the processor executes wrong or invalid instructions.

The error appears in the following condition:

- A cache-line is allocated in normal (non-frozen) mode, but only in its upper part, it will be partially filled (until the end of the cache-line), i.e. not all valid bits are set.

- In frozen mode, a second access is made to the same cache-line, but this time to its middle part, to non-valid words in this line, hence generating a 'line-hit but word-miss'. Even though the cache is frozen (no new allocates allowed), missing data is fetched from memory and additional valid bits are set, but the cache-line is still not fully populated since the lower part has not yet been accessed. Immediately after this access, the processor takes a hit to a cache-line in another 8kB segment, which is allocated in another cache set, and which is fully populated (all valid bits are set).

  ==> As a result here, valid bits from the new cache-line are copied erroneously into the tag of access

- A third access is made to the partially-filled cache-line, again to non-valid words. Since the valid bits are now set, no miss is generated, and whatever contents is in these cache locations (reset value or instructions from a previous use of that cache-line) is forwarded to the processor. As a result, the processor executes wrong or invalid instructions.

Workarounds

Do not use the cache freezing at all.

Disable the cache streaming when the cache is frozen by clearing the bit 16 of Cache Control Register (CCR)

3. Anomaly when using software traps (IPN #384)

A pending interrupt trap (tt = 0x1n- where n in [0:F]) may be cleared from the pending register or the force register (without being handled) when a Ticc (Ticc 0x1n) instruction is called.
A pending interrupt trap (tt = 0x5n- where n in [0:F]) may be cleared from the pending register or the force register (without being handled) when a Ticc (Ticc 0x5n) instruction is called.

On servicing software traps 0x1n & 0x5n (resp. SPARC synchronous traps 0x9n & 0xDn, n in [0:F]), the processor will erroneously behave as servicing the matching interrupt 0xn (SPARC asynchronous trap 0x1n) while it should not.

Consequently, the processor will perform the following erroneous actions as soon as trap 0x9n or 0xDn is entered (n in [0:F]):

- Discard any pending interrupt 0xn by clearing the matching bit "n" in the ITP register (Interrupt pending Register, address 0x80000094)
- Discard any forced interrupt 0xn by clearing the matching bit "n" in the ITF register (Interrupt Force Register, address 0x80000098)
- Freeze the instruction cache and/or the data cache (CCR.ics = "01" and/or CCR.dcs = "01") if the corresponding cache is enabled (CCR.ics = "11" and/or CCR.dcs = "11") and the corresponding freeze on-interrupt feature is activated (CCR.if = "1" and/or CCR.df = "1")

Root cause

The RTL code responsible for acknowledging SPARC interrupts (asynchronous traps) only checks bits [5:4] of the trap type (TBR.tt[5:4]) while it should be checking bits [7:4] of the trap type (TBR.tt[7:4]).

Consequently, that code is activated for 4 sets of trap types where ("xxxx" a binary representation of "n"):

- TBR.tt = "0001xxxx"
  (asynchronous trap 0x1n, aknowledges interrupt 0xn),
- TBR.tt = "0101xxxx"
  (synchronous trap 0x5n, never activated),
- TBR.tt = "1001xxxx"
  (synchronous trap 0x9n, aknowledges software trap 0x1n, triggered by instruction "ticc 0x1n")
- TBR.tt = "1101xxxx"
  (synchronous trap 0xDn, aknowledges software trap 0x5n, triggered by instruction "ticc 0x5n")

Workaround

Avoid using software traps 0x1n & 0x5n (resp. SPARC synchronous traps 0x9n & 0xDn, n in [0:F]).
No Ticc instruction shall use trap numbers which result in a tt value of 0x9n or 0xDn. So if existing software is using such values, it shall be modified.

4. Anomaly when using PCI with double bit errors detected inside PCI data packet.

Issue description:
When double bit error is detected by EDAC inside a PCI data packet to be transferred on the PCI network, only internal error is reported to the device on the PCI side and no errors are reported on the PCI bus. However the processor is aware that an issue occurs (trap 0x01 is detected).

Consequently, the others devices on the PCI network are not aware that a double EDAC error has been detected by one of the devices.

device	Description	Trap (0x1) detected	PCI target Internal error	PCI initiator Internal error	Error deteceted	PERR SERR Target abort Master abort detected
Test case 1/ PCI_DATA_READ HOST initiator – SATELLITE target –DMA mode						
	The whole data packet is transferred but the corrupted data is replaced by the following one. Hence, if the corrupted data is not the last of the dma transfer, the next data is repeated twice.					
HOST				/	N	N
SATELLITE		Y	Y			
Test case 2/ PCI_DATA_WRITE HOST target – SATELLITE Initiator –DMA mode						
	The whole data packet is transferred but the corrupted data is replaced by the following one. Hence, if the corrupted data is not the last of the dma transfer, the next data is repeated twice.					
HOST					N	N
SATELLITE		Y	/	Y		
Test case 3/ PCI_DATA_WRITE HOST initiator – SATELLITE target –DMA mode						
	The whole data packet is transferred but the corrupted data is replaced by the following one. Hence, if the corrupted data is not the last of the dma transfer, the next data is repeated twice.					
HOST		Y	/	Y		
SATELLITE					N	N
Test case 4/ PCI_DATA_READ HOST target – SATELLITE initiator –DMA mode						
	The whole data packet is transferred but the corrupted data is replaced by the following one. Hence, if the corrupted data is not the last of the dma transfer, the next data is repeated twice.					
HOST		Y	Y			
SATELLITE				/	N	N

device	Description	Trap (0x1) detected	PCI target Internal error	PCI initiator Internal error	Error deteceted	PERR SERR Target abort Master abort detected
Test case 5/ PCI_DATA_READ HOST initiator – SATELLITE target –MM mode						
	The whole data packet is transferred but the corrupted data is replaced by the following one. Hence, if the corrupted data is not the last of the dma transfer, the next data is repeated twice.					
HOST				/	N	N
SATELLITE		Y	Y			
Test case 6/ PCI_DATA_WRITE HOST target – SATELLITE Initiator – MM mode						
	The data packet is not completely transferred on the PCI bus. Data before the faulty data are correctly sended whereas the the faulty data and the next one are not sended anymore.					
HOST					N	N
SATELLITE		Y	/	N		
Test case 7/ PCI_DATA_WRITE HOST initiator – SATELLITE target – MM mode						
	The data packet is not completely transferred on the PCI bus. Data before the faulty data are correctly sended whereas the the faulty data and the next one are not sended anymore.					
HOST		Y	/	N		
SATELLITE					N	N
Test case 8/ PCI_DATA_READ HOST target – SATELLITE initiator – MM mode						
	The whole data packet is transferred but the corrupted data is replaced by the following one. Hence, if the corrupted data is not the last of the dma transfer, the next data is repeated twice.					
HOST		Y	Y			
SATELLITE				/	N	N

Root cause:
Double bit error detection by EDAC on data to be transferred by PCI is not addressed in the vhdl code: HRESP_ERROR is not addressed on the PCI transfer. (`pci_tar.vhd - line 652 to 656`)

“ To HRESP_ERROR, a 1 cycle HTRANS_IDLE response must be given,then the burst continues without repeating the old transaction. The data transfer to/from the PCI core is not suppressed, means that the corresponding trcv data is discarded, or invalid data is written to txmt, to ensure the correct order of subsequent data.”

Workaround: no workaround

5. FPU exception origin (Aexc and cexc bit fields) are not updated when FPU trap is activated

Issue description:

When the FPU trap enable mask is activated, (FSR.tem bit field) and when the corresponding IEEE_754_exception floating point trap occurs, the exception type bits fields (FSR.aexc and FSR.cexc) are not updated in returns. Hence, the origin of the trap is not known.

On the contrary, when the FPU trap enable mask is deactivated, (FSR.tem bit field), the exception type bits fields (FSR.aexc and FSR.cexc) are updated in returns when errors are reported.

Root cause:

The rtl code responsible for the update of FSR.aexc and FSR.cexc bit fields is only accessible when the FSR.tem bit field is not activated whereas it should be accessible in both cases (FSR.tem activated or not).

Workaround:

No workaround

Enabling Unlimited Possibilities™

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki
Shinagawa-ku, Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370