MPC8533E Chip Errata

This document details all known silicon errata for this PowerQUICC III device. The following table provides a revision history for this document.

Table 1. Revision History

Revision	Date	Substantive Changes
6	03/2011	Modified PCIe 3, PCIe 4, and PCIe 5 Modified Impact for SEC-A001
5	12/2010	Added CPU-A005, eTSEC56, eTSEC57, eTSEC-A001, eTSEC-A004, and SEC-A001 Added silicon version 2.1 information Modified eTSEC-A002 description and workaround Modified CPU 2 impact Removed LBIU 2 Updated disposition in PCIe4
4	05/2010	Added eTSEC-A002, PCIe-A001 Updated disposition in PCIe4 Removed silicon version 2.0
3	11/2009	Added CPU-A001 Updated eTSEC32 There was no MPC8533 Rev 2 chip errata.
1	05/2009	Updated disposition in PCIe4.
0	02/2009	Initial release.

The following table provides a cross-reference to match the revision code in the processor version register to the revision level marked on the device.

Table 2. Revision Level to Part Marking Cross-Reference

Device Number	Device Revision	e500 v2 Core Revision	Processor Version Register Value	System Version Register Value	Note
MPC8533E	1.0	2.1	0x8021_0021	0x803C_0010	With Security
MPC8533	1.0	2.1	0x8021_0021	0x8034_0010	Without Security
MPC8533E	1.1/1.1.1	2.2	0x8021_0022	0x803C_0011	With Security
MPC8533	1.1/1.1.1	2.2	0x8021_0022	0x8034_0011	Without Security
MPC8533E	2.1	2.3	0x8021_0023	0x803C_0021	With Security
MPC8533	2.1	2.3	0x8021_0023	0x8034_0021	Without Security

The following table summarizes all known errata and lists the corresponding silicon revision level to which they apply. A 'Yes' entry indicates the erratum applies to a particular revision level, while a '—' entry means it does not apply.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1/1.1.1	2.1
CPU					
CPU 1	A core hang possible while executing a msync or mbar 0 instruction and a snoopable transaction from an I/O master tagged to make quick forward progress is present.	Fixed in Rev 1.1	Yes	—	—
CPU 2	"mbar MO = 1" instruction fails to order caching-inhibited guarded loads and stores	No plans to fix	Yes	Yes	Yes
CPU 3	Branch Fails to Decrement CTR in Presence of D-cache Parity Error	No plans to fix	Yes	Yes	Yes
CPU 4	Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing **icbtls**	No plans to fix	Yes	Yes	Yes
CPU 5	Single-precision floating-point zero value may have the wrong sign	No plans to fix	Yes	Yes	Yes
CPU 6	A performance monitor time base transition event will not freeze the performance monitor counters and may not cause an exception	No plans to fix	Yes	Yes	Yes
CPU-A001	Flash-Invalidation of TLB1 with "mtspr MMUCSR0" may fail	No plans to fix	Yes	Yes	Yes
A-001428 (CPU-A005)	Enabling IEEE 754 exceptions can cause errors	Fixed in Rev 2.1	Yes	Yes	—
eTSEC					
eTSEC 1	Frame is dropped with collision and HALFDUP[Excess Defer] = 0	No plans to fix	Yes	Yes	Yes
eTSEC 2	WWR bit Anomaly	No plans to fix	Yes	Yes	Yes
eTSEC 3	Parsing of tunneled IP packets not supported	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1/1.1	2.1
eTSEC 4	Magic packet does not wake device from a SLEEP state	No plans to fix	Yes	Yes	Yes
eTSEC 5	RSTAT[RXF0] set regardless of destination ring if WWR=0	Fixed in Rev 1.1	Yes	—	—
eTSEC 6	Error in arbitrary extraction offset	No plans to fix	Yes	Yes	Yes
eTSEC 7	Parser results may be lost if TCP/UDP checksum checking is enabled	No plans to fix	Yes	Yes	Yes
eTSEC 8	Missing basic integrity check for parsing Tunneled IP packets	No plans to fix	Yes	Yes	Yes
eTSEC 9	Tx IP and TCP/UDP Checksum Generation not supported for some Tx FCB offsets	Fixed in Rev 1.1	Yes	—	—
eTSEC 10	Fetches with errors not flagged, may cause livelock or false halt	No plans to fix	Yes	Yes	Yes
eTSEC 11	Transmit jumbo frames greater than 2400 bytes may cause lost data, loss of BD synchronization, or false underrun error	No plan to fix	Yes	Yes	Yes
eTSEC 12	Parsing of MPLS label stack or non-IPv4/IPv6 label not supported	No plans to fix	Yes	Yes	Yes
eTSEC 13	Parser continues parsing L4 fields when RCTRL[PRSDEP] set for L2 and L3 fields only	No plans to fix	Yes	Yes	Yes
eTSEC 14	Back-to-back IPv6 routing headers not supported by parser	No plans to fix	Yes	Yes	Yes
eTSEC 15	RxBD[TR] not asserted during truncation when last 4 bytes match CRC	No plans to fix	Yes	Yes	Yes
eTSEC 16	RQUEUE[EXn] bits have no effect	No plans to fix	Yes	Yes	Yes
eTSEC 17	eTSEC Parser does not properly parse L3 fields	No plans to fix	Yes	Yes	Yes
eTSEC 18	Arbitrary extraction on short frames uses data from previous frame	No plans to fix	Yes	Yes	Yes
eTSEC 19	Rx TCP/UDP checksum checking may be incorrect while operating at low frequencies in FIFO mode	No plans to fix	Yes	Yes	Yes
eTSEC 20	Some combinations of Tx packets may trigger a false Data Parity Error (DPE)	No plans to fix	Yes	Yes	Yes
eTSEC 21	eTSEC half duplex receiver packet corruption	No plans to fix	Yes	Yes	Yes
eTSEC 22	eTSEC Data Parity Error (DPE) does not abort transmit frames	No plans to fix	Yes	Yes	Yes
eTSEC 23	May drop Rx packets in non-FIFO modes with lossless flow control enabled	No plans to fix	Yes	Yes	Yes
eTSEC 24	Compound filer rules do not roll back the mask	No plans to fix	Yes	Yes	Yes
eTSEC 25	Filer does not support matching against broadcast address flag PID1[EBC]	No plans to fix	Yes	Yes	Yes
eTSEC 26	eTSEC does not support parsing of LLC/SNAP/VLAN packets	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1/1.1	2.1
eTSEC 27	eTSEC receivers may not be properly initialized	Fixed in Rev 1.1	Yes	—	—
eTSEC 28	Incomplete frame with error causes false CR error on next frame	No plans to fix	Yes	Yes	Yes
eTSEC 29	Parser does not check VER/TYPE of PPPoE packets	No plans to fix	Yes	Yes	Yes
eTSEC 30	Back-to-back Rx frames may lose parser results of second frame	No plan to fix	Yes	Yes	Yes
eTSEC 31	No parser error for packets containing invalid IPv6 routing header packet	No plans to fix	Yes	Yes	Yes
eTSEC 32	Generation of Ethernet pause frames may cause Tx lockup and false BD close	Fixed in Rev. 1.1	Yes	—	—
eTSEC 33	Tx errors truncate packets without error in 8-bit Encoded FIFO mode	No plans to fix	Yes	Yes	Yes
eTSEC 34	RMCA, RBCA counters do not correctly count valid VLAN tagged frames	No plans to fix	Yes	Yes	Yes
eTSEC 35	eTSEC filer reports incorrect Ether-types with certain MPLS frames	No plans to fix	Yes	Yes	Yes
eTSEC 36	Wrong source ID reported	No plans to fix	Yes	Yes	Yes
eTSEC 37	Transmission of truncated frames may cause hang or lost data	No plans to fix	Yes	Yes	Yes
eTSEC 38	False TCP/UDP and IP checksum error in FIFO mode without CRC appending	No plans to fix	Yes	Yes	Yes
eTSEC 39	Arbitrary Extraction cannot extract last data bytes of frame	No plans to fix	Yes	Yes	Yes
eTSEC 40	Frames greater than 9600 bytes with TOE = 1 will hang controller	No plans to fix	Yes	Yes	Yes
eTSEC 41	Setting RCTRL[LFC] = 0 may not immediately disable LFC	No plans to fix	Yes	Yes	Yes
eTSEC 42	VLAN Insertion corrupts frame if user-defined Tx preamble enabled	No plans to fix	Yes	Yes	Yes
eTSEC 43	False parity error at Tx startup	No plans to fix	Yes	Yes	Yes
eTSEC 44	User-defined Tx preamble incompatible with Tx Checksum	No plans to fix	Yes	Yes	Yes
eTSEC 45	False TCP/UDP checksum error for some values of pseudo header Source Address	No plans to fix	Yes	Yes	Yes
eTSEC 46	Rx packet padding limitations at low clock ratios	No plans to fix	Yes	Yes	Yes
eTSEC 47	Transmit fails to utilize 100% of line bandwidth	No plans to fix	Yes	Yes	Yes
eTSEC 48	Unexpected babbling receive error in FIFO modes	No plans to fix	Yes	Yes	Yes
eTSEC 49	ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software	No plans to fix	Yes	Yes	Yes
eTSEC 50	Half-duplex collision on FCS of Short Frame may cause Tx lockup	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1/1.1	2.1
eTSEC 51	Magic Packet Sequence Embedded in Partial Sequence Not Recognized	No plans to fix	Yes	Yes	Yes
eTSEC 52	MAC: Malformed Magic Packet Triggers Magic Packet Exit	No plans to fix	Yes	Yes	Yes
eTSEC 53	Receive pause frame with PTV = 0 does not resume transmission	No plans to fix	Yes	Yes	Yes
eTSEC 54	Rx may hang if RxFIFO overflows	No plans to fix	Yes	Yes	Yes
eTSEC 56	Excess delays when transmitting TOE=1 large frames	No plans to fix	Yes	Yes	Yes
eTSEC 57	Controller may not be able to transmit pause frame during pause state	No plans to fix	Yes	Yes	Yes
eTSEC-A001	MAC: Pause time may be shorter than specified if transmit in progress	No plans to fix	Yes	Yes	Yes
eTSEC-A002	Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame	No plans to fix	Yes	Yes	Yes
eTSEC-A004	User-defined preamble not supported at low clock ratios	No plans to fix	Yes	Yes	Yes
SEC					
SEC 1	Scatter/Gather length error also raises boundary error.	No plans to fix	Yes	Yes	Yes
SEC 2	AES-CTR mode data size error	No plans to fix	Yes	Yes	Yes
SEC 3	Single descriptor SRTP error	No plans to fix	Yes	Yes	Yes
SEC 4	AES-CCM ICV checking write-back error	No plans to fix	Yes	Yes	Yes
SEC 5	TLS_SSL_Block_Inbound HMAC Error	No plans to fix	Yes	Yes	Yes
SEC 6	Kasumi hardware ICV checking does not work	No plans to fix	Yes	Yes	Yes
SEC-A001	Channel Hang with Zero Length Data	No plans to fix	Yes	Yes	Yes
LBIU					
LBIU 1	UPM does not have indication of completion of a RUN PATTERN special operation	No plans to fix	Yes	Yes	Yes
LBIU 3	Some of local bus events are not counted correctly in the performance monitor	No plans to fix	Yes	Yes	Yes
DMA					
DMA 1	$\overline{\text{DMA\_DACK}}$ bus timing violation when operating in external DMA master mode	No plans to fix	Yes	Yes	Yes
DMA 2	Transfer error reported for wrong channel	No plans to fix	Yes	Yes	Yes
I2C					
I2C 1	I2C boot sequencer cannot issue snoopable writes	No plans to fix	Yes	Yes	Yes
I2C 2	Enabling I^2C could cause I^2C bus freeze when other I^2C devices communicate	No plans to fix	Yes	Yes	Yes
DUART					

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1/1.1	2.1
DUART 1	BREAK detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes	Yes
DDR					
DDR 1	The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values	No plans to fix	Yes	Yes	Yes
DDR 2	On-die termination at the DDR IOs has been measured 75 Ω too high	Fixed in Rev 1.1	Yes	—	—
DDR 3	DDR IOs default receiver biasing is not optimal	Fixed in Rev 1.1	Yes	—	—
DDR 4	On-die termination at the DDR IOs does not meet spec.	No plans to fix	—	Yes	Yes
DDR 5	MCKE signal may not function correctly at assertion of $\overline{\text{HRESET}}$	No plans to fix	Yes	Yes	Yes
DDR 6	Memory contents may not be retained during $\overline{\text{HRESET}}$ sequence	No plans to fix	Yes	Yes	Yes
PCI					
PCI 1	Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang	No plans to fix	Yes	Yes	Yes
PCI 2	Master-Abort issued incorrectly for outbound DAC with subtractive decode	No plans to fix	Yes	Yes	Yes
PEX					
PCIe 1	PCI Express 2 cannot operate in x1, x2, or x4 mode	Fixed in Rev 1.1	Yes	—	—
PCIe 2	Completion Timeout error disable corrupts CRS threshold error data	No plans to fix	Yes	Yes	Yes
PCIe 3	PCI Express LTSSM may fail to properly train with a link partner following $\overline{\text{HRESET}}$	No plans to fix	Yes	Yes	Yes
PCIe 4	No mechanism for recovery from hang after access to down link	Partial fix in Rev 2.1	Yes	Yes	Yes
PCIe 5	Reads to PCI Express CCSRs or local config space temporarily return all Fs	No plans to fix	Yes	Yes	Yes
PCIe-A001	PCI Express Hot Reset event may cause data corruption	No plans to fix	Yes	Yes	Yes
PIC					
PIC 1	PCI Express MSI other than interrupt 0 not supported via hardware	No plans to fix	Yes	Yes	Yes
JTAG					
JTAG 1	TMS requires hold time beyond the fall of TCK	Fixed in Rev 1.1	Yes	—	—
JTAG 2	Boundary scan test on SerDes transmitter pins needs special requirement incompliant to IEEE 1149.1 specification	No plans to fix	Yes	Yes	Yes
GEN					

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1/1.1	2.1
GEN 1	Some pins do not meet 500V CDM ESD criteria	No plans to fix	Yes	Yes	Yes

CPU 1: A core hang possible while executing a msync or mbar 0 instruction and a snoopable transaction from an I/O master tagged to make quick forward progress is present.

Description: Transactions from I/O masters (eTSEC, Security, DMA, PCI, PCI-X, SRIO, PCI-Exp, HT) are sometimes elevated in priority in the Coherency Module. If a **msync** or **mbar** 0 instruction is being executed inside of the core while one of these high priority transactions from I/O is “snooped” to the core's cache, a hang can result.

Impact: The core hangs and all I/O traffic going through the Coherency Module comes to a halt. The application must be using msync or mbar 0 instructions inside of the core for this problem to appear. Note also that ATMU and or setting of the buffer descriptors must be setup to permit I/O masters to perform coherent transfers for such a core hang to occur.

Note that peer-to-peer type traffic across the OCN complex continues without lockup.

Workaround: Setting the Coherency Module debug register at CCSR offset 0x0_1010, bit 15 to a logic 1 prevents this type of lockup. This disables the state machine inside of the Coherency Module that elevates these transactions from I/O masters to highest priority. The result is that all transactions that the core initiates are always permitted to compete on an equal basis on the Core Connect Bus (CCB) so that the core will not hang.

Fix plan: Fixed in Rev 1.1

CPU 2: "mbar MO = 1" instruction fails to order caching-inhibited guarded loads and stores

Description: This errata describes a failure of the e500 **mbar** instruction when the MO field is one. In particular, the "**mbar** MO = 1" instruction fails to act as a barrier which cannot be bypassed by caching-inhibited loads.

Assume the following instruction sequence:

- **stw** caching-inhibited, guarded address A
- **mbar** MO = 1
- **lwz** caching-inhibited, guarded address B

The “**mbar** MO = 1” instruction is intended to be a barrier which prevents the **lwz** from executing before the **stw** has been performed. However, the e500 does not behave as intended, and allows the **lwz** to be executed before the **stw** has been performed.

Impact: This errata is most likely to affect device drivers that depend on “**mbar** MO = 1” to ensure that the effects of caching-inhibited stores are seen by a device before a subsequent caching-inhibited guarded load is executed.

The “**mbar** MO = 1” instruction is intended to order:

- Cacheable stores
- Cache-inhibited loads
- Cache-inhibited stores

The only case where the instruction may not behave correctly is where cache-inhibited loads may erroneously bypass cache-inhibited stores.

Workaround: Use "**mbar** MO = 0" to ensure that caching-inhibited guarded loads do not bypass the memory barrier.

Fix plan: No plans to fix

CPU 3: Branch Fails to Decrement CTR in Presence of D-cache Parity Error

Description: There is a potential confluence of events that results in a failure to decrement the CTR when the branch instruction is executed. The necessary conditions are as follows:

- The branch is mispredicted, which can occur if branch prediction is either of the following:
  - Enabled
  - Disabled, and the branch is actually taken
- There is a load instruction in the branch's mispredicted path, and that load hits in the L1 D-Cache, and the cache line has a parity error.
- The parity error is detected within a narrow timing window of when the branch is deallocated.

Under these conditions, the branch redirects the instruction stream to the correct branch target, but the CTR is not decremented.

The speculative load that hits the parity error is flushed when the branch is resolved. D-Cache parity errors cause precise machine check interrupts. Therefore, because the load instruction is flushed, no machine check occurs.

After the mispredicted branch is resolved, the core begins executing from the corrected path. However, if an interrupt occurs during the execution of one of the first few instructions in the corrected path, there is a possibility that SRR0 points to an instruction in the mispredicted path.

Impact: This errata may cause undetected, erroneous program behavior in the presence of L1 D-Cache parity errors.

Workaround: None

Fix plan: No plans to fix

CPU 4: Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing icbtls

Description: When executing the **icbtls**, if the instruction fetch unit fetches a line that results in an ITLB Miss interrupt, the value loaded into SRR0 will be incorrect.

Impact: This failure could cause the handler to return (**rfi**) to the wrong address.

This errata does not apply to **icbtls** instructions if CT ! = 0.

Workaround: This problem can be avoided by executing **isync** after **icbtls**. The program must ensure that no ITLB Miss can occur between the execution of the **icbtls** and the execution of the **isync**. This implies that the **isync** should reside on the same page as the **icbtls**.

A sequence of **icbtls** instructions can be followed by a single **isync** to avoid this problem. No ITLB Miss exceptions should be allowed between the execution of the first **icbtls** in this sequence and the **isync** at the end of the sequence.

There is no need to avoid asynchronous interrupts between the execution of **icbtls** and the following **isync**. The occurrence of the interrupt has the same effect as the **isync** and prevents the problem.

Fix plan: No plans to fix

CPU 5: Single-precision floating-point zero value may have the wrong sign

Description: When performing single-precision floating-point operations that produce a result of zero, the sign of the zero value may be incorrect.

Impact: Single-precision floating-point operations that result in zero may not be compatible with IEEE Std 754™.

Workaround: None

Fix plan: No plans to fix

CPU 6: A performance monitor time base transition event will not freeze the performance monitor counters and may not cause an exception

Description: The performance monitor counters will not freeze when a time base transition occurs even though PMGC0[TBEE] and PMGC0[FCECE] are enabled. Also, a performance monitor exception may not happen when a time base transition occurs even though PMGC0[TBEE] and PMGC0[PMIE] are enabled.

Impact: Performance monitor information about processor activity cannot be gathered during a precise interval based on the time base timer.

Workaround: The counters freeze when the msb = 1 in PMCx, PMLCax[CE] = 1, and PMGC0[FCECE] is enabled. Exceptions occur when the msb = 1 in PMCx, PMCLCax[CE] = 1, and PMGC0[PMIE] is enabled. Therefore, one of the performance monitor counters can be set to count a specific number of processor cycles that corresponds to the time interval desired.

After calculating the specific number of processor cycles required, program PMCx to 0x8000_0000 minus this number. This causes PMCx to overflow after the desired amount of cycles.

Fix plan: No plans to fix

CPU-A001: Flash-Invalidation of TLB1 with "mtspr MMUCSR0" may fail

Description: The e500v1 and v2 specification says that writing a 1 to MMUCSR0[TLB1_FI] will flash-invalidate all TLB1 entries that are not marked with "IPROT". However, this flash-invalidate mechanism may fail if the execution of the **mtspr** loosely coincides with the execution of a **tlbivax** that invalidates entries in the TLB1 (i.e., a **tlbivax** with bits 60–61 of the effective address set to binary 10). This **tlbivax** may be executed either on the same processor or on a different processor than the "**mtmsr MMUCSR0**" instruction.

Impact: Failure to invalidate all of the intended lines can result in operating system failure. In most operating systems, invalidation of TLB entries is restricted to a couple of places in the OS, and they are usually performed under controlled circumstances. Therefore, this bug may not impact a particular OS.

Workaround: Use one of the following options:

- Always invalidate the entire TLB1 with **tlbivax** with bits 60–61 = 11 instead of "**mtspr MMUCSR0**".
- Ensure that **tlbivax** and "**mtspr MMUCSR0**" cannot be executed simultaneously by using mutual exclusion locks around any code that invalidates TLB1.
- In a single-core environment, it is sufficient to ensure that **tlbivax** does not closely follow the **mtspr** that sets MMUCSR0[TLB1_FI] = 1 without an intervening **msync**.

Fix plan: No plans to fix

A-001428(CPU-A005): Enabling IEEE 754 exceptions can cause errors

Description: This issue can occur if a single-precision floating-point, double-precision floating-point, or vector floating-point instruction on a mispredicted branch path signals one of the floating-point data interrupts enabled by the SPEFSCR (FINVE, FDBZE, FUNFE or FOVFE bits). This interrupt must be recorded in a one-cycle window when the misprediction is resolved.

If this extremely rare event should occur, the result could be that the SPE Data Exception from the mispredicted path may be reported erroneously if a single-precision floating-point, double-precision floating-point, or vector floating-point instruction is the second instruction on the correct branch path.

It is only possible for this erratum to occur if any of the SPEFSCR exception enable bits (FINVE, FDBZE, FUNFE or FOVFE) are set to one.

Impact: A correctly executing floating point instruction that is the second instruction on the correct path may take an unexpected data exception. This is caused by an un-related floating point instruction that has been cancelled on the mis-predicted path.

Workaround: Use one of the following options:

- Ensure that the floating-point data exceptions are disabled by clearing the SPEFSCR exception enable bits (FINVE, FDBZE, FUNFE or FOVFE).
- Have the exception handler make the hardware re-execute the instruction, if a floating point instruction causes an unexpected data exception. If the exception was a result of this erratum, there will be no exception on re-execution. Freescale will make this modification to the exception handler we provide to the open source community.

Fix plan: Fixed in Rev 2.1

eTSEC 1: Frame is dropped with collision and HALFDUP[Excess Defer] = 0

Description: eTSEC drops excessively deferred frames without reporting error status when HALFDUP[Excess Defer] = 0. This erratum affects 10/100 Half Duplex modes only.

Impact: The eTSEC does not correctly abort frames that are excessively deferred. Instead it closes the BD as if the frame is transmitted successfully. This results in the frame being dropped (because it is never transmitted) without any error status being reported to software.

Workaround: Do not change HALFDUP[Excess Defer] from its default of 1.Thus eTSEC always tries to transmit frames regardless of the length of time the transmitter defers to carrier.

Fix plan: No plans to fix

eTSEC 2: WWR bit Anomaly

Description: DMACTRL[WWR] is intended to delay setting of IEVENT bits TXB, TXF, XFUN, LC, CRL, RXB, RXF until the system acknowledges that the buffer descriptor write data is actually in memory (L2 cache or DDR SDRAM), and not in flight in the system. There are certain cases when there are multiple outstanding BD updates, particularly in high latency memory scenarios, where an IEVENT can be lost when using DMACTRL[WWR] = 1.

Impact: If DMACTRL[WWR] = 1, then there is on occasion a missed IEVENT, or possibly an incorrect IEVENT assertion. This means that the interrupt could be missed altogether (BD still correctly updated in memory), or the IEVENT could be incorrect. In the case of it being incorrect, the IEVENT would not correspond to the BD at the head of the list, but would correspond to the BD second or third in the list.

Workaround: Set DMACTRL[WWR] = 0. The effect of setting DMACTRL[WWR] = 0 is that the interrupt may arrive at the processor before the update to the BD for the received packet that caused the interrupt has been completed in memory. This may or may not have any impact on the system depending on how packets are processed.

If the CPU reads the BD immediately after the interrupt, then in heavily congested systems it is possible that the CPU completes a read of the BD before the BD is closed by the eTSEC so that the BD's Empty bit is still set. In this case, software can either exit the packet processing routine and service the packet upon receiving the next interrupt, or it can schedule another interrupt to process the packet later.

Use of Rx interrupt coalescing of even a few packets reduce the chance of the CPU reading a BD whose update is still in flight to virtually zero, though it is still possible if multiple receive rings are in use.

Fix plan: No plans to fix

eTSEC 3: Parsing of tunneled IP packets not supported

Description: Encapsulation of IP in IP in either TCP or UDP packets is not supported by eTSEC parser. This applies to both IPv4 and IPv6.

A tunneled IP packet is an IP/TCP or IP/UDP packet and one of the following:

1. IPv4 header with a value of either 4 or 41 in the Protocol field, indicating that the next header is either another IPv4 header or IPv6 header, respectively
2. IPv6 header with a value of either 4 or 41 in the Next Header field, indicating that the next header is either a IPv4 header or another IPv6 header, respectively

Impact: Validly encapsulated tunneled IP packets may cause a false parser error or false TCP/UDP checksum error.

Malformed tunneled packets may be received without a parser error.

Tunneled packets with an actual TCP/UDP checksum error may fail to report a checksum error.

Workaround: If L3 or L4 parsing is enabled and tunneled packets are expected, software must examine each packet header to see if it is a tunneled IP packet. If the packet is a tunneled IP packet, software should ignore any parser or checksum error.

Fix plan: No plans to fix

eTSEC 4: Magic packet does not wake device from a SLEEP state

Description: There is a problem waking the device from a SLEEP state with a magic packet. When a magic packet is received on an eTSEC which is configured to come out of a low-power state (DOZE, NAP, SLEEP), the device is supposed to generate an interrupt to the interrupt controller which in turn gets the chip out of the low power state. Current versions of the device perform the correct action for DOZE and NAP, but not for SLEEP.

Impact: Magic packet cannot be used to get the device out of a SLEEP state.

Workaround: There is no work around for this erratum. Use DOZE or NAP low-power states for applications that use the magic packet to get the device out of a low-power state.

Fix plan: No plans to fix

eTSEC 5: RSTAT[RXF0] set regardless of destination ring if WWR=0

Description: If WWR=0, RSTAT[RXF0] may be set when a receive frame event occurs, even if the event actually occurs on a different RxBD ring.

Impact: Software cannot rely on RSTAT[RXF0] to indicate that a ring-0 receive-frame event occurred, or that receive-frame events on other RxBD rings will set the correct RSTAT[RXFn] bit.

Workaround: When RSTAT[RXF0] is set, software should check all active rings for the updated RxBD. If RSTAT[RXF1:RXF7] is set, only the corresponding ring needs to be checked.

See also eTSEC 2 (WWR Bit Anomaly) for a description of other software requirements when WWR=0.

Fix plan: Fixed in Rev 1.1

eTSEC 6: Error in arbitrary extraction offset

Description: The byte offset for the arbitrary extraction filer feature is shifted such that the wrong bytes are extracted in some cases and some byte offsets cannot be extracted. The problem only applies to L2 extraction.

Impact: The following bytes cannot be extracted:

- With no VLAN/MPLS/SNAP/PPOE tag: Packet bytes 20-21 and 54-55 cannot be extracted.
- With 1 tag: Packet bytes 24-25 and 58-59 cannot be extracted.
- With 2 tags: Packet bytes 28-29 and 62-63 cannot be extracted.

Note that PPOE and SNAP count as two tags each.

L2 extraction of bytes other than the above requires software assistance as described in the workaround.

Workaround: With one tag (VLAN/MPLS/PPOE):

- BxFFSET=0-7 extract preamble bytes 0-7.
- BxFFSET=8-27 extract bytes 0-19 of packet. Byte 0 is the first byte of the DA.
- BxFFSET=28-33 extract bytes 18-23 of packet.
- Beginning at offset 34, the pattern is criss-crossed within a 4-byte granularity and is repeated after every 4 bytes. For example:
  - BxFFSET=34 extract byte 28 of packet.
  - BxFFSET=35 extract byte 29 of packet.
  - BxFFSET=36 extract byte 26 of packet.
  - BxFFSET=37 extract byte 27 of packet.
  - BxFFSET=38 extract byte 32 of packet.
  - BxFFSET=39 extract byte 33 of packet.
  - BxFFSET=40 extract byte 30 of packet.
  - BxFFSET=41 extract byte 31 of packet.

With 2 tags (VLAN/MPLS/PPOE):

- BxFFSET=0-7 extract preamble bytes 0-7.
- BxFFSET=8-31 extract bytes 0-23 of packet. Byte 0 is the first byte of the DA.
- BxFFSET=32-37 extract bytes 18-27 of packet.
- Beginning at offset 38, the pattern is criss-crossed within a 4-byte granularity and is repeated after every 4 bytes. For example:
  - BxFFSET=38 extract byte 32 of packet.
  - BxFFSET=39 extract byte 33 of packet.
  - BxFFSET=40 extract byte 30 of packet.
  - BxFFSET=41 extract byte 31 of packet.
  - BxFFSET=42 extract byte 36 of packet.
  - BxFFSET=43 extract byte 37 of packet.
  - BxFFSET=44 extract byte 34 of packet.
  - BxFFSET=45 extract byte 35 of packet.

Fix plan: No plans to fix

eTSEC 7: Parser results may be lost if TCP/UDP checksum checking is enabled

Description: When the parser is enabled and RCTRL[TUCSEN]=1, if the first RxBD data arrives from memory the same cycle that parsing of the packet completes, all the fields of the RxFCB except the receive queue index will be written with zeroes instead of the parser results.

Impact: When a single-cycle collision of first RxBD prefetch and parsing complete occurs, the parser results other than receive queue index are lost and the VLN, IP, IP6, TUP, CIP, CTU, EIP, ETU, PERR, PRO, VLCTL bits of the RxFCB are set to all zeroes.

If VLAN extraction is enabled (RCTRL[VLEX]), the VLAN ID is lost.

Workaround: Option 1: Disable TCP/UDP checksum checking by setting RCTRL[TUCSEN]=0.

Option 2: Disable VLAN extraction by setting RCTRL[VLEX] and check the contents of the RxFCB. If the contents are zero, replicate the parser algorithm in software to determine the correct parser results.

Fix plan: No plans to fix

eTSEC 8: Missing basic integrity check for parsing Tunneled IP packets

Description: eTSEC verifies basic integrity in outer IP headers, but not inner ones. It is good practice to verify packet header integrity on fields that are defined to contain certain values.

Two basic integrity checks involve the following:

- Ensuring that the version field of the IP header is a 4 or a 6, and corresponds to the previous headers encoding. Else, this is a parse error. For non-tunneled packets eTSEC perform this functionality properly.
- In the case of IPv4, the minimum header length allowed by the standard is 20 bytes, encoded as 0x5 in the header length field. Values less that 5 should be considered parse error. For non-tunneled packets eTSEC perform this functionality properly.

Impact: eTSEC will parse and file these irregular packets as valid encodings.

Workaround: If L3 or L4 parsing is enabled and tunneled packets are expected, software should perform basic checking on receive packets to see if they are tunneled IP packets. If so, the software should perform these checks.

Fix plan: No plans to fix

eTSEC 9: Tx IP and TCP/UDP Checksum Generation not supported for some Tx FCB offsets

Description: If the Tx FCB (Frame Control Block) 32-byte offset is 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E or 0x1F, IP and TCP/UDP header checksum generation do not function properly. The checksum value may be inserted in the wrong location or not inserted at all.

Impact: IP and TCP/UDP header checksum generation is not supported in LINUX and other systems in which headers are prepended to pre-aligned packet data, or where the alignment of the Tx FCB cannot be controlled.

This behavior applies to pseudo-header checksum insertion as well as checksum generation.

Workaround: Align Tx FCB to a 16 or 32-byte boundary.

If the alignment of TxFCB is not controllable, set TCTRL[TUCSEN]=0 and TCTRL[IPCSEN]=0 to disable IP and TCP/UDP header checksum generation.

Fix plan: Fixed in Rev 1.1

eTSEC 10: Fetches with errors not flagged, may cause livelock or false halt

Description: The error management for address (for example, unmapped address) and data (for example, multi-bit ECC) errors in the Ethernet controller does not properly handle all scenarios. The behavior is as follows:

Scenario 1

- First TxBD fetch for queue 0 with polling enabled (DMACTRL[WOP] = 0), or,
- Any fetch if EDIS[EBERRDIS] = 1
    - The Ethernet controller keeps refetching the same address, resulting in a livelock of the transmit or receive state machines. If the source of the error (for example, address mapping unit in the case or unmapped address, DDR controller in the case of ECC on memory data) has interrupts enabled for the error condition, the interrupt handler can resolve the error (by for example, mapping the unmapped address or writing the memory location with good ECC). The controller resumes normal function when it receives the fetch data without an error.
    - The controller can also recover from the livelock condition by toggling MACCFG1[TX_EN] for Tx livelock or MACCFG1[RX_EN] for Rx livelock.

Scenario 2

- First TxBD fetch for queue 0 with polling disabled
    - The Ethernet controller halts all Tx queues (TSTAT[THLTn] = 1, n = 0–7), but does not set IEVENT[EBERR]. Software can determine that the halt was due to an error rather than processing complete by examining the rings for ready TxBDs. EDIS[EBERRDIS] must be 0.

Scenario 3

- Address error on Tx data fetch
    - The Ethernet controller does not detect errors on Tx data fetches. IEVENT[EBERR] is not set and the queues are not halted. For address errors, no data is returned to eTSEC and the controller hangs. The error may still be detected at the platform level, via an interrupt from the source of the error (for example, ECM/MCM address mapping error).

Scenario 4

- Data error on Tx data fetch
    - The Ethernet controller does not detect errors on Tx data fetches. IEVENT[EBERR] is not set for an address or data error on Tx data fetches, and the queues are not halted. The error is handled at the platform level, via an interrupt from the source of the error (for example, DDRC multibit ECC error or address mapping error)

Some fetch errors are handled correctly. The correct behavior is as follows:

- Non-first TxBD fetch for queue 0, or,
- TxBD fetch for queues 1–7
    - The Ethernet controller sets IEVENT[EBERR] and halts all Tx queues (TSTAT[THLTn] = 1, n = 0 – 7]. This is the correct operation for Tx fetch error conditions. EDIS[EBERRDIS] must be 0.

- RxBD fetch
  - The Ethernet controller sets IEVENT[EBERR] and halts the queue with the error (RSTAT[QHLTn] = 1). This is the correct operation for Rx fetch error conditions. EDIS[EBERRDIS] must be 0.

Impact: The Ethernet controller may stop transmitting packets without setting IEVENT[EBERR] if a buffer descriptor or data fetch has an uncorrectable error.

The transmit scheduler may halt queues without setting IEVENT[EBERR] if a buffer descriptor fetch has an uncorrectable error.

The controller does not detect errors on Tx data fetches and transmits corrupted data without an error indicator.

Workaround: All scenarios:

1. Make sure all eTSEC BD and data addresses map to valid regions of memory.
2. Ensure EDIS[EBERRDIS] = 0.

For error scenario 2, as well as the correct operation scenarios 1 and 2:

Have software periodically check the state of the Tx queue halt bits. If a queue is halted, but still contains ready BDs, resolve any pending address or data error conditions before restarting the queues.

For error scenarios 1 and 3:

- Option 1 for scenario 1:

  Disable polling (set DMACTRL[WOP]=1). Queue 0 error management will then follow scenario 2.

- Option 2 for scenario 1:

  Enable all error interrupt enables for address and data errors in regions of memory used by TxBDs. Ensure error interrupt handlers resolve each error condition (unmapped address, uncorrectable ECC error, etc.) so the controller would eventually receive data without error and continue.

- Option 3 for scenario 1, or recovery from scenario 3:

  If error interrupt handlers cannot resolve address or data errors without changing Tx state (e.g. BD address), execute a Tx reset to recover from Tx livelock condition.

- **Option 1 for Tx queue 0:**

  Disable polling (set DMACTRL[WOP] = 1). Queue 0 error management then follows queue 1–7 error management (queue halt without error indicator, but no livelock).

- **Option 2 for Tx queue 0:**

  Enable all error interrupt enables for address and data errors in regions of memory used by TxBDs. Ensure error interrupt handlers resolve each error condition (unmapped address, uncorrectable ECC error, etc.) so the controller would eventually receive data without error and continue.

- **Option 3 for Tx queue 0:**

  If error interrupt handlers cannot resolve address or data errors without changing Tx state (for example, BD address), execute a Tx reset to recover from Tx livelock condition.

  The Tx reset sequence is:

  a. Set DMACTRL[GTS]

b. Poll IEVENT[GTSC] until set or 10,000 byte times elapse
c. Clear MACCFG1[Tx_Flow]
d. Wait 256 TX_CLK cycles
e. Clear MACCFG1[Tx_EN]
f. Wait 3 TX clocks
g. Set MACCFG1[Reset Tx MC] and MACCFG1[Reset Tx fun]
h. Wait 3 TX clocks
i. Clear MACCFG1[Reset Tx MC] and MACCFG1[Reset Tx fun]
j. Set TBPTRn to next available BD in the TX ring.
k. Set MACCFG1[Tx EN] and, if desired, MACCFG1[Tx Flow]
l. Clear DMACTRL[GTSC]

For error scenario 4:

Data errors can be flagged by platform for additional software processing, but no workaround exists to prevent the transmission of corrupted data.

Fix plan: No plans to fix

eTSEC 11: Transmit jumbo frames greater than 2400 bytes may cause lost data, loss of BD synchronization, or false underrun error

Description: If the transmit processes a combination of up to four active frames which together exceed 9600 bytes, the Tx FIFO may overflow. When the TxFIFO overflows, one of several error conditions may occur. The scenarios below are representative, and may occur singly or in combination:

Scenario 1 (Lost data): The eTSEC overwrites part of a frame that has already started transmitting. The controller terminates the transmitting frame early without signaling an error condition or aborting the frame with bad CRC. In this scenario, the frame being loaded into the TxFIFO has TOE=1. [original eTSEC-55]

Scenario 2 (Lost BD synchronization): The eTSEC overwrites part of a frame that has already started transmitting. The controller transmits parts of two frames as a single frame with good CRC. Only the first frame's BD is closed. As each subsequent frame is transmitted, the BD of the previous frame is closed. The controller never recovers synchronization of BD to transmitted frame. This can occur with TOE=1 or TOE=0.

Scenario 3 (False underrun error): The eTSEC overwrites part of a frame that has already started transmitting. The controller terminates the transmitting frame with invalid CRC and halts (TSTAT[THLTn]=1). In addition, a transmit underrun error is falsely reported (IEVENT[XFUN]=1 and TxBD[UN]=1). This can occur with TOE=1 or TOE=0.

Impact: Combinations of frames that include jumbo frames greater than 2400 bytes may cause lost data, lost frames or false underrun indication in systems where the transmit throughput can fall behind the memory fetch throughput. This can occur with a fast memory subsystem, a slow interface, or collisions on the interface.

Workaround: Option 1: Limit jumbo frames to 2400 bytes maximum size on transmit.

Option 2: If using jumbo frames larger than 2400 bytes, limit the active TxBDs so no combination of up to four frames exceeds 9600 bytes.

Fix plan: No plan to fix

eTSEC 12: Parsing of MPLS label stack or non-IPv4/IPv6 label not supported

Description: The parser does not continue parsing beyond multi-label stack, or MPLS frame with a label other than IPv4 or IPv6. The RxFCB is written as 0x0000_00ff_0000_0000 (no layer 3 header recognized).

Impact: The eTSEC cannot parse beyond an MPLS stack of greater than depth 1. It also cannot parse beyond an MPLS header with label other than IPv4 or IPv6.

Workaround: Limit MPLS Ether-type packets to MPLS label stack depth = 1 with IPv4 or IPv6 label.

Fix plan: No plans to fix

eTSEC 13: Parser continues parsing L4 fields when RCTRL[PRSDEP] set for L2 and L3 fields only

Description: RCTRL[PRSDEP] has encodings for the following:

- Disabling the parser (b00)
- Enabling parsing for L2 fields only (b01)
- Enabling parsing for L2 and L3 fields only (b10)
- Enabling parsing for L2, L3 and L4 fields (b11)

In the case of setting RCTRL[PRSDEP] = b10, the eTSEC does not stop its parsing activity after the L3 fields have been identified. Instead, it continues to parse the L4 fields, attempting to identify any supported L4 protocols and updating the RxFCB and filer PID = 1 fields TCP and UDP.

Impact: L4 protocols are parsed and status bits are set when the eTSEC is programmed to not include L4 parsing.

Workaround: Knowing this behavior, the user can simply ignore the information associated with L4 protocols. In the case of filer PID = 1, the user must mask bits associated with TCP and UDP.

Fix plan: No plans to fix

eTSEC 14: Back-to-back IPv6 routing headers not supported by parser

Description: Upon encountering back to back IPv6 routing extension headers following an IPv6 header, the eTSEC stops further parsing, and place 0xFF in the RxFCB[PRO], and RQFPR,pid = 0xB[L4P]. If there are 2 or more IPv6 routing extension headers, and there is either an IPv6 hop-by-hop extension header (see reference to RFC2460 below) or an IPv6 destination options header or both between the routing headers, then the eTSEC continues to parse the sequence.

According to RFC2460 (current version of IPv6 specification), “Each extension header should occur at most once, except for the Destination Options header which should occur at most twice (once before a Routing header and once before the upper-layer header).” However, it goes on further to state, “IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the Hop-by-Hop Options header which is restricted to appear immediately after an IPv6 header only.”

Impact: Upon encountering a packet with 2 or more IPv6 routing extension headers that are back to back, the eTSEC parser indicates RxFCB[IP] = 1, RxFCB[IP6] = 1, and RxFCB[PRO] = 0xFF. All layer 4 related information is zero (TUP, CIP, CTU, ETU), even if there is a recognizable L4 protocol field following the extension headers.

Workaround: Software must parse the L4 information out of packets that indicate PRO = 0xFF.

Fix plan: No plans to fix

eTSEC 15: RxBD[TR] not asserted during truncation when last 4 bytes match CRC

Description: The eTSEC truncates any receive frame larger than MAXFRM, unless Huge Frame Enable is set (MACCFG2[Huge Frame] = 1). The proper behavior for the controller is to set the RxBD[TR] bit (and RxBD[LG], if RxBD[L] = 1) for any truncated frame. If the last 4 data bytes received before truncation happens to match the running CRC, then RxBD[TR] (and RxBD[LG]) is not set even though the frame has been truncated.

Impact: If the 4 data bytes just before MAXFRM bytes into the frame match the running CRC for the frame, the packet is silently truncated (no error indication via RxBD[TR]).

Workaround: None

Fix plan: No plans to fix

eTSEC 16: RQUEUE[EXn] bits have no effect

Description: The RQUEUE[EXn] bits are intended to allow ring-by-ring control of stashing (allocate into L2 cache) function. The EXn bits have no effect in the eTSEC IP, so stashing is completely controlled by the settings of the DMA ATTR register.

Impact: Stashing cannot be disabled per ring. Stashing is a performance hint to the L2 cache to allocate data expected to be accessed by the CPU. If stashing is unsuccessful or disabled, the data is fetched from main memory.

Workaround: None

Fix plan: No plans to fix

eTSEC 17: eTSEC Parser does not properly parse L3 fields

Description: The eTSEC parser does not properly process tunneled IP frames, resulting in loss of parser synchronization.

Impact: Tunneled IP frames received on the eTSEC Ethernet MAC and FIFO interfaces cannot be properly parsed and filed into receive queues.

Workaround: Do not enable parser recognition for L3 field, PRSDEP = 00 or 01 in Receive Control Register (RCTRL). Parsing and filling on L2 fields continues to be supported.

Fix plan: No plans to fix

eTSEC 18: Arbitrary extraction on short frames uses data from previous frame

Description: If the Ethernet controller receives a frame which is smaller than one of the defined offsets for arbitrary extraction (RBIFX), it should set the corresponding byte of the ARB property sent to the filer to 0. Instead it returns the corresponding byte extracted from the previous frame.

Impact: If the Ethernet controller receives a frame which is smaller than one of the defined offsets for arbitrary extraction (RBIFX), it should set the corresponding byte of the ARB property sent to the filer to 0. Instead, it returns the corresponding byte extracted from the previous frame.

Workaround: Option 1: Use only BnCTL=01 (extract from offset of DA-8 bytes). For valid Ethernet frames (minimum length 64 bytes), the 6-bit offset cannot go beyond the end of the frame.

Option 2: Do not use the ARB filer property to reject frames if the controller may receive frames shorter than the location of any arbitrary extraction byte offset. Software must handle short frames which may be filed in the wrong queue

Fix plan: No plans to fix

eTSEC 19: Rx TCP/UDP checksum checking may be incorrect while operating at low frequencies in FIFO mode

Description: In FIFO mode operation, a portion of the Rx TCP/UDP checksum checking state machine assumes that the controller has two cycles to respond to certain events on the Rx interface. If the controller runs slower than twice the Rx clock frequency, it may be unable to respond to an event and could either report a false Rx TCP/UDP checksum error (RxFCB[ETU] = 1) for a good packet, or fail to report a true TCP/UDP checksum error (RxFCB[ETU] = 0).

The false or missing checksum errors only occur on frames of size 4n + 1 byte (8-bit FIFO) or 4n + 2 bytes (16-bit FIFO), and the last byte(s) of data are to be included in the checksum calculation (note that frames truncated due to size>MAXFRM may not report a checksum error).

Impact: Rx TCP/UDP checksum checking is not supported for low clock ratios in FIFO modes (both GMII-style and encoded). Truncated frames may fail to report a checksum error.

Workaround: Option 1: Turn off Rx checksum checking by setting RCTRL[TUCSEN] = 0.

Option 2: Ensure that the minimum platform frequency to eTSEC Rx_CLK ratio with Rx checksum checking enabled in FIFO mode is greater than or equal to 4:1 (for example, 500 MHz platform frequency with 125 MHz Rx_CLK).

Option 3: Make sure that packets that are being checksummed have at least 4 bytes of data at the end of the packet that is not to be included in the checksum (such as a CRC)

Fix plan: No plans to fix

eTSEC 20: Some combinations of Tx packets may trigger a false Data Parity Error (DPE)

Description: Some combinations of Tx packets may incorrectly generate parity when loading the Tx FIFO. When the data is unloaded from the FIFO to be transmitted, if parity detection is enabled (EDIS[DPEDIS] = 0), a false parity error is flagged (IEVENT[DPE]).

The packet combinations that trigger the error are as follows:

1. An initial frame (X) which is not a multiple of 8 bytes in size, and
2. A subsequent shorter frame (Y) which is not a multiple of 8 bytes in size, and
3. A specific relationship between TxFIFO write pointer used for frame (Y) relative to frame (X) just prior to EOF (which cannot be controlled by the user), and
4. A data dependency between frames (X) and (Y) - on average only 50% of the scenarios matching (1)–(3) result in a false DPE being reported.

Impact: Systems which transmit packet combinations that trigger this false parity error may see some performance degradation due to false parity error interrupt service processing. No actual data corruption occurs as a result of the false parity error.

Workaround: Although it is possible to prevent false parity error indications by disabling SRAM parity detection (accomplished by setting EDIS[DPEDIS] = 1), this can have the undesirable effect of masking indications of real parity errors. Unless the specific application is experiencing a significant number of false parity errors that are resulting in unsatisfactory performance degradation, it is recommended that SRAM parity detection remain enabled. Please refer to eTSEC 22 for more information.

Fix plan: No plans to fix

eTSEC 21: eTSEC half duplex receiver packet corruption

Description: When eTSEC is configured to run in half-duplex mode, it relies on the PHY to isolate its receiver from receive data during packet transmission. The PHY indicates contention on the wire via the COL signal in xMII, which forces the eTSEC into the standard backoff algorithm. If the eTSEC does in fact get receive data (RX_DV = 1) while it is transmitting (TX_EN = 1), it receives the packet data corrupted and inflected from its original size on its way to memory.

This issue likely arises from the transmit data being wrapped at the PHY and send to the eTSEC receiver. This issue impacts running internal and external loopback while the eTSEC is configured for half-duplex operation.

Impact: Receive data corruption occurs during simultaneous packet transmission and reception in half-duplex. Additionally, the corruption ends up with various receive MIB counters counting invalid errors depending upon the original packet size and the type of corruption (RFCS, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR). Also, the receive byte counters indicate packets larger than those transmitted were received.

Workaround: The eTSEC can be configured through the filer to discard such packets that are wrapped externally in this manner through the use of MAC addresses match and drop. The receive MIB counters still count the packets as errored when they were in fact not errored.

If loopback mode is desired, the eTSEC must be configured for full-duplex operation.

Fix plan: No plans to fix

eTSEC 22: eTSEC Data Parity Error (DPE) does not abort transmit frames

Description: eTSEC supports parity protection on its TX FIFO to protect against memory errors of two types: soft errors and hard errors. Soft errors can be caused by a RAM bit cell temporarily losing its value due to an event such as an alpha particles or neutron impact, but it holds the next write. Hard errors can be caused by a RAM bit cell no longer reliably holding a 0 and/or 1 written to it.

In either case, the parity error indicates that either at least one bit in a 16-bit data word is incorrect, or that the parity bit itself is incorrect. The correct behavior is to make sure that the peer on the other end of the link or connection is notified of this data corruption. This can by done by making sure that FCS at the end of the frame is incorrect, asserting TX_ER, or, in 16-bit FIFO encoded mode, sending an error code group. The controller does assert a local error event (IEVENT[DPE]), but does not give the link peer any error indication.

Impact: If a parity error occurs on a data bit, the corrupted packet is transmitted masked as a good packet with good FCS.

Higher layer protocols that have additional error checking such as TCP/UDP checksums may detect the corrupted data, depending on the location of the bad bit.

Workaround: None

Fix plan: No plans to fix

eTSEC 23: May drop Rx packets in non-FIFO modes with lossless flow control enabled

Description: Lossless Flow Control (enabled by setting RCTRL[LFC] = 1, is intended to ensure zero packet loss of receive packets due to lack of RxBDs. This is done by applying back pressure when the number of free RxBDs drops below a critical threshold set by software. In FIFO modes (both GMII-style and encoded), the back pressure is applied by asserting receive flow control (CRS pin) until the number of RxBDs exceeds the critical threshold.

In non-FIFO modes, the back pressure is applied by transmitting a pause control frame with the pause time as in PTV[PT]. If the number of free RxBDs is still below the critical threshold when half the pause time has expired, the controller sends another pause frame and resets the counter.

If another pause frame is transmitted during the critical window when the 1/2 PTV pause counter expires, either due to software setting TCTRL[TFC_PAUSE] or through basic flow control when the data in the RxFIFO exceeds its threshold, then the 1/2 PTV pause counter may stop counting and the state machine may cease generating automated pause extensions. If the pause time associated with the last pause control frame expires with the number of free BDs still below the critical threshold, then frames may be dropped with the IEVENT[BSY] bit set indicating frame loss due to lack of buffer descriptors.

Only the transmission of another pause frame due to TCTRL[TFC_PAUSE] or data in RxFIFO exceeding threshold restarts the 1/2 PTV counter and state machine for lossless flow control.

Impact: The Ethernet controller may run out of RxBDs and drop receive packets even if lossless flow control is enabled, in a MAC mode (GMII, RGMII, SGMII, MII, RMII, TBI, RTBI).

Workaround:

- **Option 1:**

  Enable interrupts on transmit of pause frames (IMASK[TXCEN] = 1). For every interrupt due to pause frame transmission (IEVENT[TXC] = 1), enable a counter such as a cycle count in the Performance Monitor block which would overflow and generate an interrupt after 2/3 of PVT time. If the counter is already enabled, reset the count to 2/3 of PTV time. In the overflow event interrupt handler, check the current number of free receive buffer descriptors versus the threshold. If the number of free BDs are below the threshold, manually transmit a pause frame by setting TCTRL[TFC_PAUSE] = 1. This also restarts the lossless flow control state machine. In either case (free BDs above or below threshold), disable the counter until the next transmit control frame event.

- **Option 2:**

  Enable interrupts on dropped packets due to lack of RxBDs (EDIS[BSYDIS] = 0, IMASK[BSYEN] = 1). On detecting a busy dropped packet event (IEVENT[BSY] = 1), manually transmit a pause frame by setting TCTRL[TFC_PAUSE] to restart the lossless flow control state machine.

NOTE

The probability of packet loss in MAC modes can be reduced by increasing the pause time value in PTV[PT], or increasing the critical RxBD threshold in RQPRMn[PBTHR], or both.

Fix plan: No plans to fix

eTSEC 24: Compound filer rules do not roll back the mask

Description: The eTSEC filer has associated with it a mask value that is used when rules are comparing fields of the packet against properties in the RQFPR table. The mask sets “don’t cares” in the comparison. When building a compound rule through the use of the AND bit either in or outside of a cluster guard rule (CLE = 1) you can set masks as appropriate for the subsequent rule by setting CMP = 00/01, PID = 0, and RQPROP = “desired mask.” If however the chained rule fails for any single rule, the mask should revert back to what it was prior to entering the rule chain. The erratum is that the mask does not roll back and the resulting mask can be unknown.

Impact: Some rules may falsely match or not match causing the filing of a frame to the wrong queue or incorrectly rejecting the frame in the case of an assumption of the mask being a certain value.

Workaround: When using a compound rule that consists of SETMASK rules, the user must put another SETMASK rule after the last rule in the chain that resets the mask to the value it was prior to entering the chain. The following table shows a compound rule example for work arounds.

Table 4. Compound Rule Example for Work Arounds

Table Entry	RQCTRL CLE	REJ	AND	Q	CMP	PID	RQPROP	Comment
0	0	0	1	0	0	7	0x0000_0800	—
1	0	0	1	0	0	0	0xFFFF_0000	Setmask
2	0	0	1	0	0	12	0xC054_1200	—
3	0	0	1	0	0	0	0xFF00_0000	Setmask
4	0	0	0	5	0	13	0xC055_0000	—
5	0	0	0	0	0	0	0xFFFF_FFFF	Work around

Fix plan: No plans to fix

eTSEC 25: Filer does not support matching against broadcast address flag PID1[EBC]

Description: The controller clears its copy of the Ethernet broadcast address before extracting filer properties, so the filer cannot correctly match based on broadcast address (PID1[EBC]). The frame itself is not affected.

Impact: If broadcast address matching is enabled, frames may be incorrectly filed or rejected.

Workaround: Mask off matching on broadcast address flag (PID1[EBC] = 1) by clearing the bit 16 of the mask register. If the rule needs to be able to distinguish broadcast addresses as defined by IEEE Std 802.3™-2005 is all 1's in the destination address field, then use a filer rule with PID3 and PID4 (destination MAC address) to match on broadcast Ethernet frames.

Fix plan: No plans to fix

eTSEC 26: eTSEC does not support parsing of LLC/SNAP/VLAN packets

Description: eTSEC supports parsing of LLC/SNAP headers following the Ethernet 802.3 length field interpretation. It also supports 802.1p VLAN tags. However, it does not support the encapsulation defined as Ethernet length, followed by LLC/SNAP, followed by VLAN. The parser prematurely completes “normally” upon encountering the VLAN Ether-type at the end of the LLC/SNAP encoding. The erratum is that no layer 3, layer 4, or VLAN information is submitted to the filer or reported in the RxFCB.

Impact: This unique packet type is not parsed beyond layer 2, including any VLAN processing, because the parser terminated before the VLAN tag was found.

Workaround: Software running on the host has to parse these packets because they indicate no parsing functions performed by eTSEC.

Fix plan: No plans to fix

eTSEC 27: eTSEC receivers may not be properly initialized

Description: When MACCFG1[Rx_EN] is enabled during system boot as part of the eTSEC port initialization sequence, the eTSEC Rx logic may not be properly initialized.

Impact: When the eTSEC Rx logic is not properly initialized, packet data may not be correctly received. When this occurs, the first packet received (and subsequent packets) will be affected. Failure modes can include eTSEC Rx logic “deadlock”.

Workaround: During system boot, prior to setting MACCFG1[Rx_EN] in each eTSEC, perform the following:

1. Configure the eTSEC for the expected operation (for example, if use of MAC address filtering is intended, then enable address filtering; if use of the filer is intended, then enable the filer), but do not set MACCFG1[Tx_EN] and MACCFG1[Rx_EN].
2. Set MACCFG1[Loop Back]
3. Set MACCFG1[Tx_EN] and MACCFG1[Rx_EN]
4. Transmit two (2) packets and compare the received packets to the transmitted packets. If packets are not received within 10 ms, toggle MACCFG1[Rx_EN] and repeat step (4).
5. If the comparison is not equal, toggle MACCFG1[Rx_EN] and repeat step (4) to re-initialize the eTSEC Rx logic
6. Clear MACCFG1[Loop Back] for the eTSEC.

NOTE

While in Loop Back mode, the controller must also be in Full-Duplex mode.

Fix plan: Fixed in Rev 1.1

eTSEC 28: Incomplete frame with error causes false CR error on next frame

Description: If a receive error (RX_ER = 1) occurs on an incomplete Ethernet frame which is truncated before start of frame, the error indicator persists and is reported on the following frame by setting RxBD[CR] = 1.

The incomplete frames that can trigger the error are:

1. A junk frame (random data with arbitrary # of beats and no start-of-frame found)
2. A frame with preamble, start-of-frame and with no data after the start-of-frame
3. A frame with preamble-only (no start-of-frame)

Incomplete frames can occur due to collisions in half-duplex mode, or during recovery after a link down condition.

Impact: An incomplete frame with error causes a false CR (code group or CRC) error on the next frame.

Workaround: To work around this problem when the link is down, typically the PHY generates a link down interrupt. When this link down interrupt is detected, software should

1. Perform a soft_reset (MACCFG1[Soft_Reset] = 1) or a receiver reset (MACCFG1[Reset Rx Func] = 1)
2. Keep the MAC in reset until the link is up again.
3. Discard any frames received during link down.
4. Re-enable the receiver once the link is up.

Fix plan: No plans to fix

eTSEC 29: Parser does not check VER/TYPE of PPPoE packets

Description: The Ethernet controller supports PPPoE VER/TYPE = 1 packets. For PPPoE packets with VER/TYPE not equal to 1, the controller should stop Ethernet parsing and treat it as an unrecognized PPPoE packet. Instead, the controller does not check the VER/TYPE field, and assumes it is 1.

Impact: PPPoE packets with VER/TYPE that are not type 1 are parsed as if they are type 1 PPPoE.

Workaround: None

Fix plan: No plans to fix

eTSEC 30: Back-to-back Rx frames may lose parser results of second frame

Description: In some circumstances, the parser results for a frame may be calculated when the controller is still processing the previous frame. If this scenario occurs, the parser results for the second frame are discarded, and the preloaded all zeroes value of RxFCB is returned instead. The circumstances are related to format of the two frames, but are not controllable by the receiver or the user.

Impact: Under some conditions that are not controllable by the receiver or user, parser results for the second of a back-to-back frame may be lost. If VLAN extraction is enabled, the VLAN ID is lost.

Workaround: Option 1: Disable all parsing functions (RCTRL[PRSDEP] = 00.

Option 2: Because this erratum impacts the RxFCB only, the filer still gets the correct information and be able to file to the appropriate queue or reject. If software now finds RxFCB all zeros, it must assume that it encountered this error. The only other time the RxFCB is all zeros is when the eTSEC didn't find any L2–L4 information that is recognized.

Fix plan: No plan to fix

eTSEC 31: No parser error for packets containing invalid IPv6 routing header packet

Description: If a packet with an IPv6 routing header has the “segments left” field greater than the actual number of Destination Addresses (DA) contained in the routing header, then this is an invalid packet. The erratum is that the eTSEC uses the last destination address from the routing header as part of the pseudo-header calculation for the L4 checksum Because this is really an invalid packet, the checksum should not be checked and a parse error should be flagged.

Impact: An IPv6 routing header with segments left greater than number of DAs is not flagged as an invalid packet.

Workaround: Consistency checks for IPv6 routing header packets must be performed in software, or skipped.

Fix plan: No plans to fix

eTSEC 32: Generation of Ethernet pause frames may cause Tx lockup and false BD close

Description: The process of generating a PAUSE control frame may cause the controller to stop transmitting (Tx lockup). After the controller enters a Tx lockup state, only a reset of the eTSEC and associated software allows it to start transmitting frames again.Pause flow control frame generation can be triggered by reaching the Rx FIFO threshold (basic flow control: MACCFG1[Tx_Flow]), running out of Rx buffer descriptors (lossless flow control: RCTRL[LFC]), or direct software control (TCTRL[TFC_PAUSE]).

Impact: Transmit flow control, lossless flow control, and software generation of pause flow control frames may cause the Ethernet controller to stop transmitting and require a reset of the controller.

Workaround: Disable transmit flow control by setting MACCFG1[Tx_flow] = 0.

Fix plan: Fixed in Rev. 1.1

eTSEC 33: Tx errors truncate packets without error in 8-bit Encoded FIFO mode

Description: In 8-bit encoded FIFO mode, there is no error code group or error signal to indicate a Tx error. If FIFOCFG[CRCAPP] = 1, a Tx error should corrupt the appended CRC to indicate the error. The documentation of 8-bit encoded FIFO mode states that Tx FIFO under-runs cause the controller to transmit a stream of invalid bytes. Instead, the Tx simply truncates the frame without appending the CRC at all.

Impact: Transmit under-run, bus error (invalid address or data error on Tx BD or data fetch) and Tx FIFO data parity errors (see also eTSEC 22 concerning data parity errors) cause truncated frames without CRC corruption or transmission of invalid bytes.

Workaround: Enable CRC append on Tx by setting FIFOCFG[CRCAPP] = 1 and CRC checking on Rx by setting FIFOCFG[CRCCHK] = 1. The truncation of the packet at the error presents a smaller than 1 in 4 billion chance that the last four bytes of the packet match the running CRC32 calculation, ensuring that the packet is still seen as an error.

Fix plan: No plans to fix

eTSEC 34: RMCA, RBCA counters do not correctly count valid VLAN tagged frames

Description: According to the reference manual, RMCA increments for each multicast frame with valid CRC and a length between 64 and 1518 (non-VLAN tagged frames) or 1522 (single VLAN tagged frames) excluding broadcast frames. RBCA is the same definition except it counts broadcast frames and not multicast frames. The erratum is that for a valid VLAN tagged frame greater than 1518 the eTSEC does not increment these registers.

Impact: RBCA and RMCA do not increment for validly VLAN tagged Ethernet frames greater than 1518.

Workaround: There is currently no work around for counting these packets other than software running on the core.

Fix plan: No plans to fix

eTSEC 35: eTSEC filer reports incorrect Ether-types with certain MPLS frames

Description: The eTSEC filer gets a property under PID = 7 called ETY. This usually corresponds to the last Ether-type that was encountered in a packet as it was parsed. In the case that there is an MPLS label in the packet, then the Ether-type is incorrectly returned as the last 2 bytes of the MPLS label. The last 2 bytes correspond to the LSB 4 bits of the label, the EXP field, the S field, and the TTL field. The eTSEC does not know of any header types that follow other than IPv4 or IPv6 through the use of reserved label values “IPv4 Explicit Null” and “IPv6 Explicit Null.” Hence the Ether-type is always left as the last 2 bytes of the MPLS label.

Impact: MPLS tagged packets report the incorrect Ether-type (8847 for MPLS unicast or 8848 for MPLS multicast).

Workaround: Use arbitrary extraction bytes to compare to the actual Ether-type if a filer rule is intending to file based on an MPLS label existence.

Fix plan: No plans to fix

eTSEC 36: Wrong source ID reported

Description: The gasket that ties eTSEC3 to the copper line has a bug in it and it uses the wrong source ID on the copperline bus that identifies that transaction as being from eTSEC2 instead for from 3.

Functionally it is not a problem because it is also expecting the wrong source ID.

Impact: Any status register that uses the source ID to report where a transaction came from incorrectly reports the transaction as coming from eTSEC 2 instead of 3.

Workaround: Any time a status registers says it comes from eTSEC 2, it really came from eTSEC 3. When programming the trace buffer, use eTSEC2 code instead of eTSEC3.

Fix plan: No plan to fix

eTSEC 37: Transmission of truncated frames may cause hang or lost data

Description: If all three of the following conditions are concurrently met the controller may hang, or drop some bytes from the second frame without any error indication:

1. The Ethernet controller truncates a transmitted frame which is larger than MAXFRM
2. The following frame has TOE = 1
3. The two frames together are large enough to fill the 10-Kbyte Tx FIFO without truncation

Refer also to eTSEC 15.

Impact: Truncating frames larger than MAXFRM may cause a transmit hang or lost data if combined with TOE = 1 frames.

Workaround:
- Option 1: Disable truncation by setting MACCFG2[Huge Frame] = 1.
- Option 2: Turn off TCP/IP offload enable by setting TxBD[TOE] = 0.

Fix plan: No plans to fix

eTSEC 38: False TCP/UDP and IP checksum error in FIFO mode without CRC appending

Description: Running the Ethernet controller in 8- or 16-bit FIFO mode (GMII-style or encoded) with CRC appending and checking disabled (FIFOCFG[CRCAPP] = 0 for Tx and FIFOCFG[CRCCHK] = 0 for Rx) may cause the IP or TCP/UDP checksum parsing to skip the last two bytes of the frame. This results in a false IP or TCP/UDP header checksum error because the parser may not read the data in the frame properly.

Impact: IP or TCP/UDP checksum checking, enabled through RCTRL[IPCSEN] and RCTRL[TUCSEN], may generate false errors reported in the RxFCB in FIFO modes when a CRC is not appended to the frame.

Workaround: Set FIFOCFG[CRCCHK] = 1 to enable CRC checking on Rx and enable CRC append on the corresponding Tx on the link (by setting the equivalent of FIFOCFG[CRCAPP] = 1). Having CRC in the frame ensures that the last data beat is properly aligned for IP or TCP/UDP checksum checking.

Fix plan: No plans to fix

eTSEC 39: Arbitrary Extraction cannot extract last data bytes of frame

Description: If the arbitrary extraction offset defined in the RBIFX register points to data in the last beat of a frame, the associated ARB property sent to the filer may be zero instead of the data at the designated offset, depending on packet type and length.

The following packet and extraction types are affected:

- L2, L3 or L4 extraction of packets with frame length 4n or 4n + 3
- L4 extraction of TCP/UDP packets with IP total length 4n + 1, 4n + 2, or 4n + 3.

Impact: The following conditions apply to any type of frame and L2, L3 or L4 extraction:

- For frame length of 4n, the last 2 bytes of the frame are not extractable. This applies to L2, L3 or L4 extraction in MAC or FIFO modes.
- For frame length of 4n + 3, the last 1 byte of the frame is not extractable. This applies to L2, L3 or L4 extraction in MAC or FIFO modes.

The following conditions apply to L4 extraction from a packet with TCP/UDP data (when RCTRL[PRSDEP] = 11, RCTRL[TUCSEN] = 1):

- For IP total length of 4n + 1, the L4 byte offsets 4n + m - <IP header length> are not extractable, for m = 1, 2, or 3.
- For IP total length of 4n + 2, the L4 byte offsets 4n + m - <IP header length> are not extractable, for m = 2 or 3.
- For IP total length of 4n + 3, the L4 byte offset 4n + 3 - <IP header length> is not extractable

Workaround: None

Fix plan: No plans to fix

eTSEC 40: Frames greater than 9600 bytes with TOE = 1 will hang controller

Description: The eTSEC supports frames up to 9600 bytes (huge or jumbo frame). If a frame has TOE = 1, it must be no more than 9600 bytes to fit entirely into the Tx FIFO. If the frame is larger, the controller hangs, because it must have the last byte of data in the FIFO to calculate the checksum and allow the frame to start transmission.

Impact: A frame larger than 9600 bytes with TOE = 1 hangs the Ethernet controller.

Workaround: For frames larger than 9600 bytes, set TxBD[TOE] = 0. For frames with TxBD[TOE] = 1, ensure Tx frame length <= 9600 bytes.

Fix plan: No plans to fix

eTSEC 41: Setting RCTRL[LFC] = 0 may not immediately disable LFC

Description: Lossless flow control is controlled by RCTRL[LFC]. Setting RCTRL[LFC] = 0 should immediately disable the lossless flow control state machine and stop the sending of pause frames based on number of free RxBDs. The controller instead waits until the state machine is idle before disabling it. If the state machine has been triggered by the number of free RxBDs falling below the threshold, the controller continues sending pause frame extensions until the number of free RxBDs exceeds the threshold.

Impact: Generation of pause frames due to lack of free RxBDs may continue for a time after setting RCTRL[LFC] = 0.

Workaround: When disabling LFC, first set RCTRL[LFC] = 0, then poll the number of free RxBDs until it exceeds the threshold. Once the number of free RxBDs exceeds the threshold, the configuration for LFC may be safely modified.

Fix plan: No plans to fix

eTSEC 42: VLAN Insertion corrupts frame if user-defined Tx preamble enabled

Description: When TCTRL[VLINS] = 1, the VLAN is supposed to be inserted into the Tx frame 12 bytes after start of DA (after DA and SA). If user-defined Tx preamble is enabled (MACCFG2[PreAmTxEn] = 1), the VLAN ID is inserted 12 bytes after the start of the preamble (4 bytes after start of DA), thus overwriting part of DA and SA.

Impact: If VLAN insertion is enabled with user-defined Tx preamble, the VLAN ID corrupts the Tx frame destination and source addresses.

Workaround: Use one of the following workarounds:

- Disable user-defined Tx preamble by setting MACCFG2[PreAmTxEn] = 0.
- Disable VLAN insertion by setting TCTRL[VLINS] = 0.

Fix plan: No plans to fix

eTSEC 43: False parity error at Tx startup

Description: The 10 KB TxFIFO comes out of reset in an unitialized state. Each FIFO entry is initialized as Tx frame data is written to it. Under certain internal resource contention conditions, the controller may read uninitialized data and falsely signal a parity error in IEVENT.

Impact: If parity errors are enabled before the first 10KB of Tx frame data is written to the TxFIFO, pause frames or Tx frames may trigger a false data parity error event.

Workaround: Disable parity error detection by setting EDIS[DPEDIS]=1 until at least 10 KB of Tx data has been transmitted.

Fix plan: No plans to fix

eTSEC 44: User-defined Tx preamble incompatible with Tx Checksum

Description: If user-defined Tx preamble is enabled (by setting MACCFG2[PreAmTxEn]=1), an extra 8 bytes of data is added to the frame in the Tx data FIFO. IP and TCP/UDP checksum generation do not take these extra bytes into account and write to the wrong locations in the frame.

Impact: Enabling both user-defined Tx preamble and IP or TCP/UDP checksum causes corruption of part of the corresponding header.

Workaround: Use one of the following workarounds:

- Disable user-defined Tx preamble by setting MACCFG2[PreAmTxEn] = 0.
- Disable IP and TCP/UDP checksum generation by setting TCTRL[IPCSEN]=0 and TCTRL[TUCSEN] = 0.

Fix plan: No plans to fix

eTSEC 45: False TCP/UDP checksum error for some values of pseudo header Source Address

Description: The Ethernet controller calculates the pseudo header checksum by first calculating the checksum for the individual fields of the pseudo header, then merging the checksums and carry bits. If the checksum for the Source Address (SA) field of the pseudo header is 0x1_0000 (16-bit checksum=0 with carry out=1), the carry bit is not included in the combined checksum, resulting in a false checksum error (RxFCB[ETU]=1). A pseudo header SA checksum of 0x1_0000 is only possible for IPv6frames, not IPv4.

Impact: False ETU indication when check sum for pseudo header SA is 0x1_0000 for IPv6 frames.

Workaround: If RxFCB[CTU]=1, RxFCB[ETU]=1 and RxFCB[IP6]=1, calculate the checksum for the SA field from the pseudo header. If this checksum equals 0x1_0000, then proceed to calculate the entire TCP checksum to be sure the checksum error is valid. If the SA checksum is not 0x1_0000, then the ETU is a valid checksum error indication.

Fix plan: No plans to fix

eTSEC 46: Rx packet padding limitations at low clock ratios

Description: There are two mechanisms that cause extra bytes to be inserted in front of the data in a received frame:

1. RCTRL[PAL] - packet alignment padding. A programmable mechanism for padding a frame with zeroes to achieve a particular alignment of data.
2. MACCFG2[PreamRxEn] – enables inserting the 8-byte preamble in front of the Rx frame data within the data buffer. These bytes are not accounted for in the value of RCTRL[PAL] setting.

At low clock ratios (less than 2:1 eTSEC system clock to TSECn_TX_CLK), it is possible to overflow the receive buffer where the extra bytes are inserted before data is written to the 2 KB Rx FIFO. When this Rx buffer overflow occurs, the current Rx frame will be dropped and the subsequent frame may be passed to memory without the expected padding bytes inserted.

Impact: If the eTSEC is running at less than 2:1 eTSEC system clock to TSECn_TX_CLK ratio, the controller cannot support inserting 24 or more total bytes (from padding, and the preamble) in front of the frame data

Workaround: Limit total receive packet byte insertion via RCTRL[PAL] and Rx preamble enable to less than 24 bytes total when running at less than 2:1 eTSEC system clock:TSECn_TX_CLK ratio.

Fix plan: No plans to fix

eTSEC 47: Transmit fails to utilize 100% of line bandwidth

Description: The minimum interpacket gap (IPG) between back-to-back frames is 96 bit times. To ensure 100% utilization of an interface, the maximum gap between back-to-back streaming frames should also be 96 bit times (12 cycles). The Tx portion of the Ethernet controller may fail to meet that requirement, depending on mode, clock ratio, and internal resource contention.

- For single-queue operation, IPG varies between 12–20 cycles.
- For multiple queue operation with fixed priority scheduling, IPG for back-to-back frames from different queues varies between 70–140 cycles for a 2:1 eTSEC system clock:TSECn_TX_CLK ratio and twice as many cycles for a 1:1 ratio.
- For multiple queue operation with round-robin scheduling, IPG for back-to-back frames from different queues is on the order of 20–40 cycles longer than multiple queue operation with fixed priority.

In all cases, the impact of longer IPG is greater for smaller frames. With multiple queue operation, small frames may also increase the gap, as the buffer descriptor (BD) prefetching may fall behind the data rate, especially at lower clock ratios.

Impact: Tx bandwidth cannot achieve 100% line rate, especially for multiple queue operation or relatively small frames.

Workaround: The following options maximize the bandwidth utilized by the Ethernet controller.

- If multiple Tx queue operation is not required, use single Tx queue operation (thus eliminating the extra gap caused by switching queues) and use frames larger than 64 bytes (thus reducing the IPG as a portion of total bandwidth).
- If multiple Tx queue operation is required, use priority arbitration by setting TCTRL[TXSCHED]=2'b01 and maximize the number of BDs enabled per ring to minimize switching between rings. Also, minimize use of small frames, thus reducing IPG as a portion of total bandwidth.

Fix plan: No plans to fix

eTSEC 48: Unexpected babbling receive error in FIFO modes

Description: In MAC modes, a babbling receive error occurs if MACCFG2[Huge Frame]=1 and a receive frame exceeds MAXFRM. There is no MAXFRM in FIFO modes, so there should be no babbling receive errors. The Ethernet controller does not qualify the babbling receive error with interface mode, so a babbling receive error will be reported if a receive frame on a FIFO interface exceeds the value of MAXFRM.

Impact: The controller may erroneously report babbling receive errors in FIFO mode.

Workaround: In FIFO modes, disable interrupts for babbling receive errors by setting IMASK[BREN]=0, and ignore any setting of IEVENT[BABR].

Fix plan: No plans to fix

eTSEC 49: ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software

Description: The MIB function of the Ethernet controller has a feature to automatically zero out the registers when reading them if ECNTRL[AUTOZ] = 1. If the register read occurs in the same cycle as a hardware update of the register, then the register clear will not occur. Any software periodically reading MIB registers would expect to read A the first time, B the second, and C the third, with each value representing only the events that occurred in the interval between reads. If the first read collides with a hardware update, the second read would return A + B instead of B.

Hardware updates for MIB registers occur once per frame. For streaming 64-byte frames, the update would be every 84 Rx or Tx clocks (8 bytes of preamble, 64 bytes of data and 12 cycles of IPG).

Impact: Software polling of MIB counters with ECNTRL[AUTOZ] = 1 will over an extended period read a larger number of events than actually seen by the controller.

Workaround: Disable automatic clearing of the MIB counters by writing ECNTRL[AUTOZ] = 0. Software routines which periodically read MIB counters and accumulate the results should accumulate only when an MIB counter overflows, as in the description that follows: Assuming a 32-bit MIB counter (MIB_VALUE), a 64-bit accumulator consisting of two 32-bit registers (ACCUM_HI, ACCUM_LO), and a Carry Out bit (ACCUM_LO_CO), change the 64-bit accumulator update as follows:

Previous accumulate method (with ECNTRL[AUTOZ] = 1):

```
// Accumulate the MIB_VALUE into the lower half of the accumulator
 {ACCUM_LO_CO,ACCUM_LO} = {1'b0,ACCUM_LO} + {1'b0,MIB_VALUE};
// Accumulate the Carry Out from the step above, as well as the MIB register
OVFRFLW, which is detected through the CARn register.
{ACCUM_HI_CO,ACCUM_HI} = {1'b0,ACCUM_HI} + ACCUM_LO_CO + OVRFLW;
```

New accumulate method (with ECNTRL[AUTOZ]=0):

```
// Read instead of accumulate since we are not clearing MIB_VALUE
 ACCUM_LO = MIB_VALUE;
// Accumulate the MIB register OVRFLW, which is detected through the CARn
register
 {ACCUM_HI_CO,ACCUM_HI} = {1'b0,ACCUM_HI} + OVRFLW;
```

Fix plan: No plans to fix

eTSEC 50: Half-duplex collision on FCS of Short Frame may cause Tx lockup

Description: In half-duplex mode, if a collision occurs in the FCS bytes of a short (fewer than 64 bytes) frame, then the Ethernet MAC may lock up and stop transmitting data or control frames. Only a reset of the controller can restore proper operation once it is locked up.

Impact: A collision on hardware-generated FCS bytes of a short frame in half-duplex mode may cause a Tx lockup.

Workaround: **Option 1:**

Set MACCFG2[PAD/CRC] = 1, which pads all short Tx frames to 64 bytes.

Option 2:

Use software-generated CRC (MACCFG2[PAD/CRC] = 0, MACCFG2[CRC EN] = 0 and TxBD[TC] = 0)

Fix plan: No plans to fix

The reference manual will be updated to require padding of all short Tx frames when in half-duplex mode (MACCFG2[Full Duplex] = 0).

eTSEC 51: Magic Packet Sequence Embedded in Partial Sequence Not Recognized

Description: The Ethernet MAC should recognize Magic Packet sequences as follows:

Any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of data payload. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFs, followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses.

If a complete Magic Packet sequence (including 6 bytes of 0xFF) immediately follows a partial Magic Packet sequence, however, the complete sequence will not be recognized and the MAC will not exit Magic Packet mode.

The following are example partial sequences followed by the start of a complete sequence for station address 01_02_03_04_05_06:

- FF_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

  Seventh byte of 0xFF does not match next expected byte of Magic Packet Sequence (01). Pattern search restarts looking for 6 bytes of FF at byte 01.

- FF_FF_FF_FF_FF_FF_01_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

  First FF byte following 01 does not match Magic Packet sequence.

  Pattern search restarts looking for 6 bytes of FF at second byte of FF following 01.

The following is an example partial sequence followed by the start of a complete sequence that is erroneously not recognized for station address 01_02_03_04_FF_06:

- FF_FF_FF_FF_FF_FF_01_02_03_04_FF_FF_FF_FF_FF_FF_01_<complete sequence>

  11th byte (0xFF) is seen as the 11 byte of the partial pattern and is not recognized as the start of a complete sequence.

  Pattern search restarts looking for 6 bytes of 0xFF at 12th byte, but sees only 5.

Impact: The Ethernet controller will not exit Magic Packet mode if the Magic Packet sequence is placed immediately after other frame data which partially matches the Magic Packet Sequence.

Workaround: Place 1 byte of data that is not 0xFF and does not match any bytes of DA before the start of the Magic Packet sequence in the frame.

Because the Magic Packet sequence pattern search starts at the 3rd byte after DA, the Magic Packet Sequence can be placed at the start of the data payload as long as the second byte of the length/type field follows the above rule.

Fix plan: No plans to fix

eTSEC 52: MAC: Malformed Magic Packet Triggers Magic Packet Exit

Description: The Ethernet MAC should recognize Magic Packet sequences as follows:

Any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of data payload. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFs, followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses.

Once the Ethernet MAC has recognized a valid DA for one frame, it continues searching for valid 102-byte Magic Packet sequences through multiple frames without checking for valid DA on each frame. The only events that cause the MAC to go back to check for valid DA before checking for a Magic Packet sequence on new frames are:

1. A frame containing a recognized full Magic Packet sequence (with valid or invalid FCS)
2. Software disable of Magic Packet mode (MACCFG2[MPEN]=0)
3. MAC soft reset (MACCFG1[Soft_Reset]=1)

Impact: The Ethernet controller may exit Magic Packet mode if it receives a frame with DA not matching station address, or invalid unicast or broadcast address, but a valid Magic Packet sequence for the device.

Workaround: None

Fix plan: No plans to fix

eTSEC 53: Receive pause frame with PTV = 0 does not resume transmission

Description: The Ethernet controller supports receive flow control using pause frames. If a pause frame is received, the controller sets a pause time counter to the control frame's pause time value, and stops transmitting frames as long as the counter is non-zero. The counter decrements once for every 512 bit-times. If a pause frame is received while the transmitter is still in pause state, the control frame's pause time value replaces the current value of the pause time counter, with the special case that if the pause control frame's pause time value is 0, the transmitter should exit pause state immediately. The controller does use the frame's pause time value to set the current pause time counter, but it then decrements the pause time counter before performing the compare to zero. As a result an XON (pause frame with PTV = 0), actually causes the transmitter to continue in pause state for 65,535 pause quanta, or 33,553,920 bit times.

Impact: A received pause frame with PTV = 0 causes the transmitter to pause for 65,535 pause_quanta. The expected behavior is for the controller to continue, or resume, transmission immediately. Note that the Ethernet controller always uses the value of the PTV register when generating pause frames. It never automatically generates a pause frame with pause time value of 0 when the receiver recovers from being above the RxFIFO threshold or below the free RxBDs threshold.

Workaround: To force an exit of pause state, use a pause frame with PTV value of 1 instead of 0.

Fix plan: No plans to fix

eTSEC 54: Rx may hang if RxFIFO overflows

Description: If the memory subsystem is unable to keep up with incoming traffic, the Rx FIFO may fill up and overflow. If the RxFIFO fills up, the controller should gracefully drop packets. Instead, under certain conditions on the interface between the controller and the memory subsystem, the Rx will lock up and stop receiving without any error indication.

Impact: For low ratios from platform to Rx_clk and slow memory systems, the Rx FIFO may overflow and hang the Rx controller.

Workaround: To reduce the probability of an RxFIFO overflow, enable flow control by setting MACCFG1[Tx Flow] = 1.

Statistical lockup detection and recovery:

Lockup detection:

1. Enable debug mode in the controller by writing 0x00E00C00 to offset 0x000 (TSEC_ID1).
2. Periodically poll the state of the Ethernet controller by reading RPKT, RSTAT, and the register at offset 0xD1C. If RPKT has changed, the RSTAT[QHLTn] bits are clear, and the value of register offset 0xD1C has not changed, wait X*16 bit times, where X is the largest frame expected to be received on this interface, then read the value of register offset 0xD1C again. If it still has not changed, and RPKT has changed again, then the Rx controller may be locked up. If promiscuous mode is disabled (RCTRL[PROM] = 0), or if the controller is likely to receive and discard fragmentary packets (both of which may cause RPKT to increment for packets which are discarded before the RxFIFO) additional iterations may be required to reduce the probability of a false lockup detect. There is no guaranteed algorithm to detect Rx lockup with zero false positives.

Lockup recovery:

1. Perform a graceful receive stop by setting DMACTL[GRS] = 1, and wait to ensure any outstanding prefetches are cleared. The wait time is system and memory dependent, but a reasonable worst-case time is the receive time for a 9.6 KB frame at 10/100/1000 Mbps). Note that if the Rx is truly locked up, IEVENT[GRSC] will never be set. The graceful receive stop also ensures that data and state are not corrupted during a soft reset if the lockup detection falsely detects a lockup due to rejected packets.
2. Toggle MACCFG1[Rx En] (set to 0, then set to 1).
3. Clear the graceful receive stop by setting DMACTL[GRS] = 0.

Fix plan: No plans to fix

eTSEC 56: Excess delays when transmitting TOE=1 large frames

Description: The Ethernet controller supports generation of TCP or IP checksum in frames of all sizes. If TxBD[TOE]=1 and TCTRL[TUCSEN]=1 or TCTRL[IPCSEN]=1, the controller holds the frame in the TxFIFO while it fetches the data necessary to calculate the enabled checksum(s). Because the checksums are inserted near the beginning of the frame, transmission cannot start on a TOE=1 frame until the checksum calculation and insertion are complete.

For TOE=1 huge or jumbo frames, the data required to generate the checksum may exceed the 2500-byte threshold beyond which the controller constrains itself to one memory fetch every 256 eTSEC system clocks. This throttling threshold is supposed to trigger only when the controller has sufficient data to keep transmit active for the duration of the memory fetches. The state machine handling this threshold, however, fails to take large TOE frames into account. As a result, TOE=1 frames larger than 2500 bytes often see excess delays before start of transmission.

Impact: TOE=1 frames larger than 2500 bytes may see excess delays before start of transmission.

Workaround: Limit TOE=1 frames to less than 2500 bytes to avoid excess delays due to memory throttling.

When using packets larger than 2700 bytes, it is recommended to turn TOE off.

Fix plan: No plans to fix

eTSEC 57: Controller may not be able to transmit pause frame during pause state

Description: When the Ethernet controller pauses transmit of normal frames after receiving a pause control frame with PTV!=0, it should still be able to transmit pause control frames. The Ethernet controller, however, does not check whether the MAC is paused before initiating a start-of-frame request to the MAC. Once it has initiated a start-of-frame request, the Ethernet controller cannot initiate a pause control frame request until the normal frame completes transmission. Since the MAC will not transmit the normal frame until the pause time expires, this means the controller may be unable to transmit a pause frame while it is in pause state if there is a normal frame ready to transmit.

Impact: The Ethernet controller may be unable to transmit a pause frame during pause state if a normal frame is ready to transmit.

This applies to pause frame generation as a result of RxFIFO over threshold (ordinary flow control), free BDs below threshold (lossless flow control), or software-generated pause frame (TCTRL[TFC_PAUSE]).

Workaround: None

Fix plan: No plans to fix

eTSEC-A001: MAC: Pause time may be shorter than specified if transmit in progress

Description: When the Ethernet controller receives a pause frame with PTV!=0, and MACCFG1[Rx Flow]=1, it completes transmitting any current frame in progress, then should pause for PTV*512 bit times. The MAC, however, does not take the full transmission time of the current frame into account when calculating the Tx pause time, and may pause for 1-2 pause quanta (512-1024 bit times) less than the PTV value.

Impact: The eTSEC transmitter may pause transmission for up to 1024 bit times less than requested in a receive pause frame. If the PTV value does not contain at least 2 pause quanta worth of margin, this may lead to receive buffer overflows in the link partner.

Since the transmit pause does not take effect until after the current frame completes transmitting, the link partner's pause frame generator must already include the maximum frame size as margin when calculating the pause time value to use to prevent overflow of the receiver's buffers.

Workaround: Add 2 pause quanta to the pause time value used when generating pause frames to prevent receive buffer overflow.

Fix plan: No plans to fix

eTSEC-A002: Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame

Description: Ethernet standards define the minimum frame size as 64 bytes. The eTSEC controller also supports receiving short frames less than 64B, and can accept frames more than 16B and less than 64B if RCTRL[RSF] = 1. Frames shorter than 17 bytes are supposed to be silently dropped with no side-effects. There are, however, two scenarios in which receiving frames <= 2B cause erroneous behavior in the controller.

In the first scenario, if the last frame (such as an illegal runt packet or a packet with RX_ER asserted) received prior to asserting graceful receive stop (DMACTRL[GRS]=1) is <= 2 bytes, then the controller will fail to signal graceful receive stop complete (IEVENT[GRSC]) even though the GRS has successfully executed and the receive logic is completely idle. Any subsequent receive frame which is larger than 2 bytes will reset the state so the graceful stop can complete. An Rx reset will also reset the state.

In the second scenario, the parser and filer are enabled (RCTRL[PRSDEP] = 01,10,11). If a 1 or 1.5B frame is received, the controller will carry over some state from that frame to the next, causing the next frame to be parsed incorrectly. This in turn may cause incorrect parser results in RxFCB and incorrect filing (accept versus reject, or accept to wrong queue) for that following frame. The parser state recovers itself after receiving any frame >= 2B in length.

Impact: If software initiates a graceful receive stop after a 1- or 2-byte frame is received, the stop may not complete until another frame has been received.

A frame following a 1 or 1.5B frame may be parsed and filed incorrectly.

Workaround: For GRS scenario:

After asserting graceful receive stop (DMACTRL[GRS] = 1), initiate a timeout counter. The wait time is system and memory dependent, but a reasonable worst-case time is the receive time for a 9.6 Kbyte frame at 10/100/1000 Mbps. If IEVENT[GRSC] is still not set after the timeout, read the eTSEC register at offset 0xD1C. If bits 7-14 are the same as bits 23-30, the eTSEC Rx is assumed to be idle and the Rx can be safely reset. If the register fields are not equal, wait for another timeout period and check again.

MAX Rx reset procedure:

1) Clear MACCFG[RX_EN].

2) Wait three Rx clocks.

3) Set MACCFG2[RX_EN].

For silicon version 1.0 use this step 3:

3) Execute workaround for the following eTSEC erratum: eTSEC 27: eTSEC receivers may not be properly initialized.

Fix plan: No plans to fix

eTSEC-A004: User-defined preamble not supported at low clock ratios

Description: The Ethernet controller is not designed to inject user-defined preambles into Tx frames at eTSEC system clock to Tx clock ratios of less than 1.8:1. If the controller is run with a slower eTSEC system clock, the eTSEC may hang, underrun, or corrupt the transmitting frame.

Impact: User-defined Tx preamble may cause hang, underrun, or corrupted frames.

Workaround: Disable user-defined preamble Tx injection by setting MACCFG2[PreAmTxEn]=0.

Fix plan: No plans to fix

SEC 1: Scatter/Gather length error also raises boundary error.

Description: There are two main ways of sending data to Security Engine (SEC) 2.1. The first is non-scattered. The second method is using the scatter/gather feature. In this case, there is a link table that contains the starting location and length of each part of the data. The data can be broken up into many different sections, each a continuation of the last section. The Security Engine has an overall length of the entire field, and individual lengths for each section. It follows the link table, gathering (for reading) or scattering (for writing) the data until it has read/written the entire data field.

If the link table has been incorrectly created, it is possible that a length in the table is too long or short for an individual field. For instance, in SRTP, the Extend field is followed by the main data field. The link table must have a separate link entry for each field. It is not allowed for the length of the Extend field to include any of the main data field. Conversely, it is not allowed to start the data for the main data field in the Extend link entry. Either case should cause a boundary error and a length error. However, in the case where the length of the Extend field is too short, and the data field length is correct, one would expect only a length error, and not a boundary error.

Due to the errata, Scatter/Gather length error also raises a unexpected false boundary error.

Impact: None

Workaround: None required.

Fix plan: No plans to fix

SEC 2: AES-CTR mode data size error

Description: The SEC 2.1 supports acceleration of AES Counter mode, an underlying algorithm in Secure Realtime Transport Protocol (SRTP), and optionally, IPSec. The SEC is designed to accelerate AES-CTR alone (using descriptor type 0001_0) or in parallel with an HMAC-SHA-1 using a special SRTP descriptor type 0010_1. SRTP uses AES-CTR with HMAC-SHA-1. Although AES in counter mode (AES-CTR) is meant to act as a stream cipher, the AESU considers any input data size that is not an even multiple of 16 bytes to be an error.

Impact: None

Workaround: Use one of the following options:

- The AESU Data Size error can be disabled via the AESU Interrupt Control Register to prevent a nuisance interrupt.
- The input data length (in the descriptor) can be rounded up to the nearest 16B. Set the data-in length (in the descriptor) to include X bytes of data beyond the payload. Set the data-out length to only output the relevant payload (don't need to output the padding). SEC reads from memory are not destructive, so the extra bytes included in the AES-CTR operation can be whatever bytes are contiguously trailing the payload.

Fix plan: No plans to fix

SEC 3: Single descriptor SRTP error

Description: The SRTP protocol specifies the use of AES-CTR with HMAC-SHA-1. The SEC 2.1 is designed to accelerate AES-CTR in parallel with HMAC-SHA-1 using a special SRTP descriptor type 0010_1. Single descriptor SRTP does not work for data sizes that are not even multiples of 16 bytes.

Impact: When operating on input data which is N*16B, the AESU and MDEU can each read in the portion of the datastream relevant to their respective operations. When the input data is not N*16B, the AESU data size workaround described in SEC5 (rounding up to the next 16B) does not work because these excess bytes are ‘snooped’ by the MDEU and corrupt the HMAC-SHA-1 operation.

Workaround: Because the SRTP protocol does not require the payload to be N*16B, most packets will likely be of a data length that hits this erratum. The workaround is to use two descriptors to perform SRTP.

Outbound: The first descriptor is type 0001_0 "common non-snoop" set for an AES-CTR operation using either of the workarounds described in SEC5. The second descriptor is type 0001_0 “common non-snoop” set for an HMAC-SHA-1 of the headers + unpadded encrypted payload + MKI (when present).

Inbound: Same descriptors as outbound, but in reverse order.

To minimize the performance impact of this workaround, these two descriptors should be created simultaneously and launched back-to-back. By configuring the SEC Crypto-channel to perform Done notification on selected descriptors, the first descriptor should be set to not generate a Done interrupt, while the second descriptor (which completes the SRTP operation) should be set to generate the Done interrupt. All the parameters required to build both descriptors are available at the start of the request to the SEC device driver, so there is no reason to wait for the first descriptor to complete before building and launching the second.

By running these descriptors back to back with no interrupt in between, the data operated upon by the second descriptor will likely be fetched from L2, thereby reducing the memory bus utilization associated with the second descriptor.

Fix plan: No plans to fix

SEC 4: AES-CCM ICV checking write-back error

Description: Most security protocols add an authentication tag to the frame or packet which provides the receiver with proof that the frame/packet has not been modified in transit. These authentication tags are also known as an HMACs or ICVs. The SEC is capable of generating these authentication tags for out-bound operations, and re-calculating the tag and comparing it to the received tag for in-bound operations. When performing in-bound authentication tag comparison, the SEC can notify software of a miscompare (which indicates an intentional modification of a frame/packet) by generating a crypto-channel error interrupt, or by writing a tag miscompare bit to the original descriptor header in memory. For authentication tags (ICVs) created by the AESU when in CCM mode, hardware can successfully perform the ICV comparison, but signaling of an ICV miscompare can only be performed via a channel error interrupt. When attempting to use the writeback method when an ICV miscompare is detected, all subsequent descriptors will be flagged as having ICV miscompares, whether they do or not.

Impact: A full SEC reset is required to clear the ICV miscompare writeback logic.

Workaround: For AES-CCM mode only, do not use the header writeback method for signaling authentication tag miscompare. Use the channel interrupt method. Alternately, perform ICV checking in software, as is required in all SEC cores 2.0 and below.

Fix plan: No plans to fix

SEC 5: TLS_SSL_Block_Inbound HMAC Error

Description: The SSL/TLS protocol allows for the use of several different cipher suites to encrypt and integrity check payload data. The encryption cipher can be a stream cipher like RC-4, or a block cipher like 3DES or AES.

The SSL/TLS processing steps for out-bound operations with a block cipher are as follows:

1. Calculate HMAC over the SSL record header and payload.
2. Add HMAC to the end of the payload.
3. Calculate the number of bytes of payload and HMAC, plus a Pad Length byte, and add padding bytes to the record until the total number of bytes (payload||HMAC||padding||Pad Length) is an integral number of cipher blocks.
4. Encrypt the record, starting at the first byte of payload, ending at the Pad Length (Pad Length is included in the encryption).
5. Change the Record Header Length field to match the length of payload||HMAC||padding|| Pad Length

This order of operations is opposite to most other security protocols, but does not present any special challenges to single pass hardware accelerators.

For inbound, the order of operations is almost the opposite:

1. Decrypt the record, starting at the first byte of payload, ending at the Pad Length (Pad Length is included in the decryption).
2. Calculate the number of bytes of payload by subtracting the length of the HMAC, padding, and pad length from the Length field received with the record header.
3. Change the length field to match the payload length only. Remove the padding and pad length bytes.
4. Calculate HMAC over the SSL record header and decrypted payload.
5. Compare the received HMAC with the calculated HMAC, and if different, drop the record.

This order of operations presents a significant challenge to single pass hardware. It isn't possible to change the length field in the record header until the Pad Length byte has been decrypted, and if the length field isn't changed prior to inbound integrity checking, the calculated HMAC are wrong. The SEC implementation did not comprehend that the Length field would not be modified by software prior to launching the descriptor, as is done for out-bound processing.

Impact: TLS_SSL_Block_Inbound operations produce incorrect HMAC values.

Workaround:

- Option 1: For TLS_SSL_Outbound operations, use the SEC's single pass descriptor type 1000_1 for maximum performance. For TLS_SSL_Inbound, use two descriptors:
  - First descriptor: Use type 0001_0 'common non-snoop', with header set for the appropriate block cipher algorithm, to decrypt the payload through Pad Length. Upon completion of this descriptor, software changes the value of the Length field in the record header and launches the second descriptor.
  - Second Descriptor: Type 0001_0 'common non-snoop' set for the appropriate HMAC algorithm (for example, HMAC-SHA-1). The pointers in this descriptor tell the SEC to calculate an HMAC over the SSL Record Header||Payload. The SEC can be set to automatically compare the generated HMAC against the received HMAC, or the descriptor can write the generated HMAC to memory for a comparison by software.

- Option 2: For TLS_SSL_Outbound operations, use the SEC's single pass descriptor type 1000_1 for maximum performance. For TLS_SSL_Inbound, use a two-descriptor method that minimizes the total amount of data the SEC must read to generate a decrypted and authenticated record:
  - First descriptor: Use type 0001_0 'common non-snoop', with header set for the appropriate block cipher algorithm. Assuming CBC mode, decrypt the final block of the record, using as the IV the second-to-last block of the record. Do not write the decrypted data back to the memory location of the record. The purpose of decrypting the final block is to recover the Pad Length field, which is the last byte of the output generated by this descriptor. After you have the Pad Length, discard the decrypted data. Upon completion of this descriptor, software subtracts the length of the HMAC (a known constant value), padding, and pad length from the Length field, updates the Length field in the record header and launches the second descriptor.
  - Second Descriptor: Type 1000_1 TLS_SSL_Block. Because the Length field is set properly, the single pass TLS_SSL_Inbound descriptor properly decrypts and authenticates the record.

Fix plan: No plans to fix

SEC 6: Kasumi hardware ICV checking does not work

Description: SEC's execution units (EU) that generate integrity check values (ICVs) are also capable of comparing a received ICV with a calculated ICV. In the case of the KEU (Kasumi) F9 function, the SEC is not able to properly compare a received F9 MAC (another acronym for an ICV) with the calculated MAC.

Impact: The Kasumi algorithm produces a 128-bit integrity check value, which is shortened in the 3G protocol to a 64-bits message authentication code (MAC). To verify a received MAC, the user copies the received MAC to the KEU's IV data, which the SEC fetches (along with other parameters) in order to generate the calculated MAC. The KEU is supposed to compare the upper 64 bits of the calculated MAC with the 64 bits received MAC; however, it attempts to compare the full 128 bits and consequently always reports an ICV comparison failure, for example, integrity check comparison result (ICCR) returns binary "10."

Workaround: Users are not required to use the hardware ICV comparison feature. Rather than copying the 64 bits received MAC to the KEU's IV data, the user can merely have the SEC output the 128-bit MAC generated by the KEU, and software can perform the comparison of the upper 64 bits. Performing the MAC comparison in software takes only a few more CPU cycles than copying the received MAC to the IV data and reading the SEC's comparison result from the descriptor header.

Fix plan: No plans to fix

SEC-A001: Channel Hang with Zero Length Data

Description: Many algorithms have a minimum data size or block size on which they must operate. The SEC EUs detect when the input data size is not a legal value and signal this as an error. For most EUs, a zero byte length data input should be considered illegal, however the EUs do not properly notify the crypto-channel using the CHA of a data size error. Instead, the EUs wait forever for a valid data length, leading to an apparent channel hang condition.

Impact: When EUs detect illegal input data size, EUs wait forever for a valid data length and leading to an apparent channel hang condition.

Workaround: Option 1: Ensure that software does not create SEC descriptors to encrypt or decrypt zero length data.

Option 2: Use the SEC crypto-channel watchdog timer to detect hung channels. This is accomplished by enabling each channel's watchdog timer via the Channel Configuration Register CCRx[WGN]. This is a one time configuration. The timer stops when the channel completes a descriptor and restarts at zero each time the channel fetches a new descriptor. The default setting for the watchdog timer is 2^27 SEC clock cycles. If the timer expires, the channel will generate an interrupt and the Channel Status Register will show the cause as a Watchdog Timeout [WDT]. The channel hang is cleared by resetting the channel via CCR[RST]. The offending EU is reset automatically by the channel.

Fix plan: No plans to fix

LBIU 1: UPM does not have indication of completion of a RUN PATTERN special operation

Description: A UPM special operation is initiated by writing to MxMR[OP] and then triggering the special operation by performing a dummy access to the bank.

The UPM is expected to have an indication of when a special operation is completed. The UPM will see MxMR[MAD] increment when a write to or read from UPM array special operation completes. However, the UPM does not have any status indication of completion of the Run Pattern special operation.

The following scenario could be affected by this erratum:

- A UPM Run Pattern special operation is initiated and a second UPM command is issued before the Run Pattern is completed.

If the above scenario occurs the programmed mode registers could be altered according to the second operation and cause the current/first operation to encounter errors due to mode changes in the middle of the operation. Note that if a second command issued is a Run Pattern operation and it does not change the mode registers, the first operation should not encounter errors.

The behavior of the LBIU is unpredictable if the Run Pattern special operation mode is altered between initiation of the operation and the relevant memory controller completing the operation.

Impact: Because of this erratum, when a UPM Run Pattern special operation is to be followed by any other UPM command for which MxMR needs to be changed, the run pattern operation may not be handled properly. Software does not have any means to confirm when the current Run Pattern special operation has completed so that register programming for the next operation can be done safely.

Workaround: None

Fix plan: No plans to fix

LBIU 3: Some of local bus events are not counted correctly in the performance monitor

Description: Followings events are not counted correctly:

- Bank 1-8 hits(chip-select)
- Request granted to ECM port
- Cycles atomic reservation for ECM port is enabled

Impact: Those local bus events can not be used in the performance monitor.

Workaround: None

Fix plan: No plans to fix

DMA 1: $\overline{\text{DMA\_DACK}}$ bus timing violation when operating in external DMA master mode

Description: The specification requires the external DMA master signal $\overline{\text{DMA\_DACK}}$ to be held for at least three system clocks. The DMA may violate this requirement, depending on the internal latency of a transaction. The DMA asserts $\overline{\text{DMA\_DACK}}$ based on an internal 'write done' signal, which varies depending on the write target and whether it is the last write of a transaction.

In most cases, the ' write done' signal asserts too quickly, causing the $\overline{\text{DMA_DACK}}$n to be held for too short a time.

Impact: $\overline{\text{DMA\_DACK}}$ does not meet minimum hold specification of 3 SYSCLK cycles.

Workaround: None. In the external DMA master mode, the $\overline{\text{DMA_DACK}}$n output is held for at least 16 CCB clock cycles.

Fix plan: No plans to fix

DMA 2: Transfer error reported for wrong channel

Description: The DMA controller has resources that are shared between all channels. Each channel is given a time period in the shared resources corresponding to the value of the bandwidth control specified in MR[BWC]. The last write transaction corresponding to the transfer of a block (specified by the byte count register in the channel) is a write that requires a response from the target port (WRFTP). This type of write is referred to here as an WRFTP. While a channel that sent its last write data is waiting for the write response, another channel is allowed to start using the shared resources.

When the WRFTP gets an error response, it is the channel that is active in the shared resources that will get the transfer error bit set, not the channel that is waiting for the response of the WRFTP transaction. Sources of error responses for an WRFTP are:

- The write gets a translation error from an outbound ATMU translation window at the target port (Serial RapidIO, PCI Express).
- The WRFTP translates to a non-posted write on PCI Express, and the non-posted write receives an error response from the attached device (WRFTP translation is controlled by ATMU configuration).
- The WRFTP translates to an NWRITE_R on Serial RapidIO and the write receives an error response from the attached device (WRFTP translation is controlled by ATMU configuration).

Note that an error response on a DMA read will set the transfer error bit in the correct channel. This problem is limited to getting an error on WRFTP response.

Impact: The wrong channel could have Transfer Error set. The actual failing channel will complete normally, when data may not actually have been written to the destination successfully. When the Transfer error bit is set for one channel, software will have to assume that it could be have been caused by any other channel.

Workaround: A few work arounds have been identified with varying performance and software impact:

- Do not configure the ATMUs as described above, or
- When a channel completes a block transfer or descriptor chain, check that no other channel has its transfer error set, or
- Use one channel at a time. Not very practical since it reduces the DMA to a one channel DMA.

Fix plan: No plans to fix

I2C 1: I2C boot sequencer cannot issue snoopable writes

Description: The I2C boot sequencer cannot issue snoopable writes. Boot sequencer transactions are therefore not presented on the CCB.

Impact: There is no possible way to use the I2C boot sequencer to initialize regions of the L2 configured as SRAM. Note that only being able to issue Non-snoopable transactions causes no cache coherency issues since the core is not permitted to perform its initial boot vector fetch until after the boot sequencer completes its initialization process.

Workaround: None

Fix plan: No plans to fix

I2C 2: Enabling I²C could cause I²C bus freeze when other I²C devices communicate

Description: When the I²C controller is enabled by software, if the signal SCL is high, the signal SDA is low, and the I²C address matches the data pattern on the SDA bus right after enabling, an ACK is issued on the bus. The ACK is issued because the I²C controller detects a START condition due to the nature of the SCL and SDA signals at the point of enablement. When this occurs, it may cause the I²C bus to freeze. However, it happens very rarely due to the need for two conditions to occur at the same time.

Impact: Enabling the I²C controller may cause the I²C bus to freeze while other I²C devices communicate on the bus.

Workaround: Use one of the following workarounds:

- Enable the I²C controller before starting any I²C communications on the bus. This is the preferred solution.
- If the I²C controller is configured as a slave, implement the following steps:
  a. Software enables the device by setting I2CnCR[MEN] = 1 and starts a timer.
  b. Delay for 4 I²C bus clocks.
  c. Check Bus Busy bit (I2CnSR[MBB])

```
if MBB == 0
                jump to Step f; (Good condition. Go to Normal
operation)
                else
                Disable Device (I2CnCR[MEN] = 0)
```

  d. Reconfigure all I²C registers if necessary.
  e. Go back to Step a.
  f. Normal operation.

Fix plan: No plans to fix

DUART 1: BREAK detection triggered multiple times for a single break assertion

Description: A UART break signal is defined as a logic zero being present on the UART data pin for a time longer than (START bit + Data bits + Parity bit + Stop bits). The break signal persists until the data signal rises to a logic one.

A received break is detected by reading the ULSR and checking for BI = 1. This read to ULSR clears the BI bit. After the break is detected, the normal handling of the break condition is to read the URBR to clear the ULSR[DR] bit. The expected behavior is that the ULSR[BI] and ULSR[DR] bits do not get set again for the duration of the break signal assertion. However, the ULSR[BI] and ULSR[DR] bits continue to get set each character period after they are cleared. This continues for the entire duration of the break signal.

At the end of the break signal, a random character may be falsely detected and received in the URBR, with the ULSR[DR] being set.

Impact: The ULSR[BI] and ULSR[DR] bits get set multiple times, approximately once every character period, for a single break signal. A random character may be mistakenly received at the end of the break.

Workaround: The break is first detected when ULSR is read and ULSR[BI]=1. To prevent the problem from occurring, perform the following sequence when a break is detected:

1. Read URBR, which returns a value of zero, and clears the ULSR[DR] bit
2. Delay at least 1 character period
3. Read URBR again, which return a value of zero, and clears the ULSR[DR] bit

ULSR[BI] remains asserted for the duration of the break. The UART block does not trigger any additional interrupts for the duration of the break.

This workaround requires that the break signal be at least 2 character-lengths in duration.

This work around applies to both polling and interrupt-driven implementations.

Fix plan: No plans to fix

DDR 1: The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values

Description: When the DDR controller executes the automatic CAS-to-preamble logic, it will be possible that the controller will calibrate to an erroneous value. In addition, there will not be an error reported in the ERR_DETECT register, and there is no valid way to tell if the calibrated value is valid.

Impact: The automatic CAS-to-preamble function of the DDR controller can fail, which could lead to data corruption on the DDR interface if this feature is enabled.

Workaround: The automatic CAS-to-preamble feature should not be enabled. Instead the CAS-to-preamble should be calculated (using examples shown in application note AN2583 from Freescale).

Fix plan: No plans to fix

DDR 2: On-die termination at the DDR IOs has been measured 75 Ω too high

Description: The DDR IOs provide termination options of 75 Ω and 150 Ω. Silicon measurements show about 150 Ω and 225 Ω, respectively.

Impact: The termination at the DDR IOs will be inaccurate. By setting the 75 Ω option, one can still get termination of about 150 Ω. However, there is no way to get 75 Ω.

Workaround: When trying to obtain 150 Ω termination, the 75 Ω termination option can be set by clearing DDRCDR[ODT]. However, there is no workaround to obtain 75 Ω termination. Note that this issue has been present on all previous revisions of the device. Customers who designed using simulation results should not have to alter impedance settings via software in fixed silicon, as the lines will simply be better balanced using fixed silicon.

Fix plan: Fixed in Rev 1.1

DDR 3: DDR IOs default receiver biasing is not optimal

Description: The DDR IO receiver biasing is controlled through settings in an engineering use only register. The current default settings are not optimal. In order to run the DDR interface reliably, the default setting needs to be overridden.

Impact: The DDR interface may fail if the default receiver biasing value is not overridden.

Workaround: Write register at offset 0xE_0F48 with a value of 0x9000_0000 before enabling DDR controller. This will set the receiver to an acceptable bias point, but it is not the ideal biasing value.

Fix plan: Fixed in Rev 1.1

DDR 4: On-die termination at the DDR IOs does not meet spec.

Description: The DDR IOs provide termination options of 75 Ω and 150 Ω . In rev 1.1, silicon measurements show the impedance is about 22 Ω too high for both the 75-Ω and 150-Ω ODT options

Impact: For rev 1.1 the ODT was improved to be closer to spec. However, silicon measurements show about 92 Ω and 172 Ω for ODT options of 75 Ω and 150 Ω, respectively.

Workaround: The IBIS models provided for rev 1.1 reflect silicon, not the spec. Customers who designed using simulation results should not have to alter impedance settings through software because the lines are better balanced in rev. 1.1 silicon.

Fix plan: No plans to fix

DDR 5: MCKE signal may not function correctly at assertion of HRESET

Description: During the assertion of HRESET (excluding the initial power-on-reset) the device may erroneously drive the state of MCKE to the incorrect level or release it to high impedance after removing the clocks from the DRAM. This could place the DRAMs into an undefined state causing future operations to fail. The primary fail mechanism is for the device to incorrectly train its I/O receivers during DDR initialization.

There are power on reset configuration signals sampled during HRESET that do not quickly achieve correct values using their internal pull-ups. As a result, the device may be temporarily placed into a test mode.

The DDR controller should drive the MCKE[0:3] pins active low throughout and after HRESET assertion. However, when the DDR controller enters a test mode, the DDR MCKE driver is released to high impedance or driven high, preventing the MCKE pins from being driven low immediately after HRESET assertion.

Impact: The DRAMs may erroneously enter an undefined state preventing the completion of read operations during DRAM initialization sequence. This may result in an auto calibration error (ERR_DETECT[ACE]) or improper training during the initialization sequence. A failure to train properly may result in corrupted data transfers to and from DDR.

Workaround: There are several possible workarounds. Depending on the application, select one of the following options:

Option 1

After negation of HRESET perform an alternative DDR controller initialization sequence for each utilized controller. This clears the DRAM state machines and allows them to operate properly. Before this sequence is implemented do not enable any DDR LAWBAR entries. Details of alternative sequence are as follows:

NOTE the following DEBUG registers:

- DEBUG_2 offset is CCSRBAR + DDR_OFFSET + 0xf04
- DEBUG_3 offset is CCSRBAR + DDR_OFFSET + 0xf08

1. Configure DDR registers as is done in normal DDR configuration. Do not set DDR_SDRAM_CFG[MEM_EN].
2. Set reserved bit EEBACR[3] at offset 0x1000.
3. Before DDR_SDRAM_CFG[MEM_EN] is set, write DDR_SDRAM_CFG_2[D_INIT].
4. Before DDR_SDRAM_CFG[MEM_EN] is set, write D3[21] to disable data training.
5. Wait 200 µs (as described in the section “DDR SDRAM Initialization Sequence,” in the applicable device reference manual)
6. Set DDR_SDRAM_CFG[MEM_EN].
7. Poll DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware.
8. Clear D3[21] to re-enable training.
9. Set D2[21] to force the data training to run.
10. Poll on D2[21] until it is cleared by hardware.

After this step there are two options that can be followed if ECC is enabled before continuing on to step 11 . If DDR ECC is not utilized enable the DDR LAWBARs and continue to step 11 . Sub-Option 1 requires a calculated delay. Sub-Option 2 does not require the delay, but it is not supported for applications with DDR interleaving enabled.

Sub-Option 1

a. Wait calculated delay

   Required delay for 64-bit DDR2 can be calculated as follows:

   Delay = 400 ms/Gbytes × max memory size

   For 32-bit data buses, multiply this number by 2.

   Example: assume 64-bit DDR2, memory size = 1 Gbyte

   Delay = 400ms/Gbytes × 1 Gbyte = 400 ms

b. Set DDR_SDRAM_CFG_2[D_INIT]
c. Poll on DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware, then the system can proceed.
d. Enable any DDR LAWBAR entries and proceed to step 11 .

Sub-Option 2

a. Enable any DDR LAWBAR entries.
b. Set ERR_DISABLE[MBED] and ERR_DISABLE[SBED] to disable SBE and MBE detection.
c. Complete a 32-byte non-snoopable DMA transaction with the source and destination address equal to the DDR initialization address which is either the starting address of CS0_BNDS by default or programmed in DDR_INIT_ADDR.
d. After the DMA transaction has completed clear ERR_DISABLE[MBED] and ERR_DISABLE[SBED] to enable SBE and MBE detection as desired for specific applications.

11. Clear reserved bit EEBACR[3] at offset 0x1000.

Work Around Option 2

Use an active component (for example, CPLD) to drive MCKE signals to the DRAMs. Inputs to this logic should include MCKE and $\overline{\text{HRESET\_REQ}}$, both from the device. When $\overline{\text{HRESET\_REQ}}$ asserts, the MCKE signal to the DRAMs should be driven low by the active component. When $\overline{\text{HRESET\_REQ}}$ is negated, the MCKE value driven by the CPLD should match the value driven by the device. The JEDEC defined t_{Delay} parameter between the MCKE and MCK/$\overline{\text{MCK}}$ signals must also be controlled by this workaround. In addition, note that MCKE must still meet all JEDEC-defined ADDR/CMD setup/hold requirements when using the external component to help drive MCKE.

Option 3

Power cycle the DRAM during $\overline{\text{HRESET}}$ assertions.

Fix plan: No plans to fix

DDR 6: Memory contents may not be retained during $\overline{\text{HRESET}}$ sequence

Description: It may be desirable for customers to have the DDR controller enter self refresh mode and $\overline{\text{HRESET}}$ the part shortly after, while retaining contents of memory. However, it is possible that CKE will not be driven active low during $\overline{\text{HRESET}}$, bringing the DRAM out of self refresh.

There are pin-sampled signals that may erroneously place the DDR into a test mode if they are not set to a valid state when $\overline{\text{HRESET}}$ is asserted. The DDR controller should drive the MCKE[0:3] pins active low throughout and after $\overline{\text{HRESET}}$. However, when this test mode is entered, the DDR drivers will be inactive, and the MCKE[0:3] pins will be released to high impedance. This test mode will be entered if a value of 0xf is presented on LA[28:31] during $\overline{\text{HRESET}}$, or if a value of 0x0 is presented on MSRCID[2] during $\overline{\text{HRESET}}$. The MCKE[0:3] pins will remain released to high impedance until LA[28:31] and MSRCID[2] are set correctly for pin sampling.

Impact: The DRAMs may erroneously exit self refresh mode if MCKE is released to high impedance and transitions above the minimum AC switching voltage level.

Workaround: Depending upon the board application, it may be possible to apply a weak pull-down or use active components during $\overline{\text{HRESET}}$ to ensure that MCKE[0:3] will remain low throughout $\overline{\text{HRESET}}$.

Fix plan: No plans to fix

PCI 1: Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang

Description: As a master, the PCI IP block can combine a memory write to the last PCI double word (4 bytes) of a cacheline with a 4 byte memory write to the first PCI double word of the subsequent cacheline.

This only occurs if the second memory write arrives to the PCI IP block before the deassertion of $\overline{\text{FRAME}}$ for the first write transaction. If the writes are combined, the PCI IP block masters a single memory-write transaction on the PCI bus. If for this transaction, the PCI target asserts $\overline{\text{STOP}}$ during the last data beat of the transaction ($\overline{\text{FRAME}}$ is deasserted, but $\overline{\text{TRDY}}$ and $\overline{\text{IRDY}}$are asserted), the transaction completes correctly. A subsequent write transaction other than an 8-byte write transaction causes a hang on the bus. Two different hang conditions can occur:

- If the target disconnects with data on the first beat of this last write transaction, the PCI IP block deasserts $\overline{\text{IRDY}}$ on the same cycle as it deasserts $\overline{\text{FRAME}}$ (PCI protocol violation), and no more transactions will be mastered by the PCI IP block.
- If the target does not disconnect with data on the first beat of this last write transaction, $\overline{\text{IRDY}}$ will be deasserted after the first beat is transferred and will not be asserted anymore after that, causing a hang.

Impact: This affects 32-bit PCI target devices that blindly assert $\overline{\text{STOP}}$ on memory-write transactions, without detecting that the data beat being transferred is the last data beat of the transaction. It can cause a hang.

If the PCI transaction is a one data beat transaction and the target asserts $\overline{\text{STOP}}$ during the transfer of that beat, there is no impact.

Workaround: Hardware workaround:

Ensure that the PCI target device does not assert $\overline{\text{STOP}}$ during the last beat of a PCI memory write transaction that is greater than one data beat and crosses a cacheline boundary. It could assert $\overline{\text{STOP}}$ during the last data beat of the 32-byte cacheline or not assert $\overline{\text{STOP}}$ at all.

Software workarounds:

Set bit 10, the master disabling streaming (MDS) bit, of the PCI bus function register (address 0x44) to prevent the combining of discrete outbound PCI writes or the recombining of writes that cross the 32-byte cacheline boundary into a burst.

Set the PCI latency timer register (offset 0x0D) to zero. A value of zero is the reset value for this register, so keeping this register unmodified after reset prevents the PCI IP block from ever combining writes. It is not necessary to set the PCI latency timer register to zero if the MDS bit is set.

Fix plan: No plans to fix

PCI 2: Master-Abort issued incorrectly for outbound DAC with subtractive decode

Description: If an outbound command is issued by a PCI host and no response to the transaction occurs through the subtractive decoding cycle, then per the PCI specification a Master-Abort command should be issued by the host on the cycle subsequent to the subtractive decoding cycle. The device does not conform to the PCI standard for DAC transactions, and instead issues the Master-Abort one cycle earlier.

Impact: This device is not fully compliant with PCI rev2.2 specification regarding Master-Abort for DEVSEL not asserted, for DAC transactions. However, in most cases the subtractive decoding is used in error condition (when no device answers).

Workaround: Do not use subtractive decoding for DAC transactions other than error conditions.

Fix plan: No plans to fix

PCIe 1: PCI Express 2 cannot operate in x1, x2, or x4 mode

Description: A timing issue in PCI Express 2 prevents x1, x2, or x4 operation. The PCI Express 2 clocks arrive late to the protocol converter and hold times were violated on all four lanes.

Impact: PCI Express 2 (address offset 0x9000) can not operate in x1, x2, or x4 configuration.

Affected pins are: SD1_RX[4:7], /SD1_RX[4:7], SD1_TX[4:7], /SD1_TX[4:7]

Workaround: No workaround exists to allow PCI Express 2 to operate in x1, x2, or x4 configuration. An alternative is to use PCI Express 1.

Fix plan: Fixed in Rev 1.1

PCIe 2: Completion Timeout error disable corrupts CRS threshold error data

Description: Several attributes of enabled error conditions are written to the PEX Error Capture Status register (offset 0xE20) when an error condition is detected. If PEX_ERR_DISR[PCTD]=1 (disable detection of Completion Timeout threshold errors), then the global source ID attribute (PEX_ERR_CAP_R2[19:24]) for CRS Threshold errors will be incorrect.

Impact: An incorrect global source ID is captured in PEX Error Capture Status register for CRS Threshold errors if Completion Timeout errors are disabled.

Workaround: Enable Completion Timeout and CRS threshold error detection by keeping the default setting of PEX_ERR_DISR[PCTD]=0 and PEX_ERR_DISR[CRSTD] = 0.

Fix plan: No plans to fix

PCIe 3: PCI Express LTSSM may fail to properly train with a link partner following $\overline{\text{HRESET}}$

Description: Following $\overline{\text{HRESET}}$, the PCI Express controller's internal link training and status state machine (LTSSM) may fail to properly detect a receiver as defined in the PCI Express Base Specification. This can be determined by reading the LTSSM state status register (offset 0x404) in the PCI Express extended configuration space.

When this failure occurs, the status code may be either 0 or 1h, which indicates that it is still in a detect state. If the link has properly trained, the status code reads 0x16h.

Note that the LTSSM status code resides at the least significant byte (little endian) of the 32-bit LTSSM register. Software has to ensure that the checking of the LTSSM status code is performed on the whole byte value.

Impact: Following $\overline{\text{HRESET}}$, (or a hot swap and/or dynamic power up/down on the link partner) the PCI Express controller may fail to properly train with an active link partner, causing the PCI Express controller to hang.

Workaround: If the link partner is not recognized, the PCI Express controller may be reset by performing the following procedure for both root complex and endpoint applications, depending on the silicon revision.:

Rev 1.1 only

1. OR the values at CCSRBAR offset 0x0_AF00, 0x0_9F00, and 0x0_BF00 with 0x0800_0000.
2. Wait 1 ms.
3. AND the values at CCSRBAR offset 0x0_AF00, 0x0_9F00, and 0x0_BF00 with 0xF7FF_FFFF.
4. Poll LTSSM register until it returns 0x16h to indicate that the link is up.

Rev 1.1 or rev. 2.1

1. Read the LTSSM state status register (offset 0x404) in each PCI Express controller's extended configuration space. If the least significant byte return value (little endian) of the read to the LTSSM status code equals 0x01, apply the following workaround in the same order listed.
2. Save the values at CCSRBAR offset 0x0_AF00, 0x0_9F00, and 0x0_BF00 to temporary locations.
3. OR the temporary values with 0x0800_0000 and write back to CCSRBAR offset 0x0_AF00, 0x0_9F00, and 0x0_BF00.
4. Wait 1 ms.
5. AND the temporary values with 0xF7FF_FFFF and write back to CCSRBAR offsets 0x0_AF00, 0x0_9F00, and 0x0_BF00.
6. Poll PEX_IP_BLK_REV1 until it returns a non-F value.
7. Poll LTSSM register until it returns 0x16h to indicate that the link is up.

Note that after reset or when recovering from a link-down condition, external transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX_LTSSM_STAT) to check the status of link training before issuing external requests.

Fix plan: No plans to fix

PCIe 4: No mechanism for recovery from hang after access to down link

Description: When a link goes down, the PCI Express controller clears all outstanding transactions with an error indication and sends a link-down exception to the interrupt controller if PEX_PME_MES_DISR[LDDD] = 0.

If new transactions (with a detailed transaction type described in the impact section below) targeting external PCI Express space are sent to the controller after the link-down event but before the link comes back up, the following occurs, depending on the silicon revision:

Rev 1.1: New transactions are accepted by the controller and wait for the link to come back up before starting any timeout counters (for example, completion timeout). There is no mechanism to cancel the new transactions except a device HRESET.

Rev 2.1: Until the link is up again, new non-posted transactions via the ATMU are cancelled with an error status signaled back to the core, and new posted transactions are silently discarded.

Impact: New transactions targeting the external memory or configuration space of the PCI Express controller while its link is down behave in the following manner, depending on the silicon revision:

Rev 1.1: Are stalled until the link recovers. The outstanding transactions may cause a core or other masters (for example, DMA) to hang or a core watchdog timeout. This includes transactions issued using both the ATMU and PEX_CONFIG_ADDR/PEX_CONFIG_DATA mechanisms.

Rev 2.1: Are either cancelled (non-posted transactions) or discarded (posted transactions), if issued via the ATMU. Non-posted transactions issued via the ATMU requiring completion result in an instruction stall with a machine-check interrupt generated to the core to allow further processing by the handler, depending on application requirement. The exception is that non-posted transactions issued using the PEX_CONFIG_ADDR/PEX_CONFIG_DATA mechanism are cancelled and do not result in an instruction stall or machine check.

Workaround: Ensure that link-down exceptions are enabled (PEX_PME_MES_DISR [LDDD] = 0 and PEX_PME_MES_IER[LDDIE] = 1) to trigger an interrupt when the Link Down Detected (LDD) event occurs. The LDD interrupt handler must then disable further access to the PCI Express interface for the duration of the link-down.

Note that the PCI Express controller can be accessed via normal reads and writes (LAW/ ATMU target) or by reads or writes to the PEX_CONFIG_ADDR/PEX_CONFIG_DATA in the PCI Express controller memory-mapped register space.

Rev 1.1: Both types of access must be prevented. This, however, may not completely eliminate the problem, because there could be accesses to the PCI Express interface between the time the link goes down and the time the interrupt handler disables access to the interface. In this situation, device HRESET is required.

Rev 2.1: As with previous revisions of silicon, a link-down event causes a Link Down Detected interrupt if enabled, followed by clean-up of the outstanding transactions. It is advisable that software disable further access to that PCI-Express link until the PEX_LTSSM_STAT returns a value of 0x16 that indicates the link is fully up.

In contrast, in rev 2.1, every new non-posted access via the ATMU to a PCI-Express link that is down causes the associated instruction to stall. However, interrupts can still be processed while this instruction is pending. Setting HID1[RXFE] allows a machine check to be generated for each non-posted transaction via the ATMU that is performed to the PCI-Express interface

while the link is down. When a machine check is detected, software should check the machine check syndrome register (MCSR) and the machine check address register (MCAR). If an access to a PCI-Express link that is down has occurred, the MCSR indicates BUS_RBERR, and the address in the MCAR falls within the PCI-Express region. The machine check interrupt handler should then prevent further accesses to the PCI-Express link until it has successfully trained again, as indicated by a value of 0x16 in the PEX_LTSSM_STAT register.

Note that this procedure requires that MSR[ME] and HID1[RXFE] are set.

Fix plan: Partial fix in Rev 2.1

PCIe 5: Reads to PCI Express CCSRs or local config space temporarily return all Fs

Description: The PCI Express Configuration, Control and Status Registers (CCSRs) refer to the complete memory-mapped register space (offset 0x000 to 0xFFF) starting at either 0xA000, 0x9000, or 0xB000 of the PCI Express controller affected by the link down event.

When a link goes down, the PCI Express controller clears all outstanding transactions, and if PEX_PME_MES_DISR[LDDD] = 0 and PEX_PME_MES_IER[LDDIE] = 1, it sends a link-down exception to the interrupt controller.

Due to this erratum, reads to local PCI Express configuration and memory-mapped CCSR space are cancelled and return all Fs in the following amount of time:

- **Rev 1.1 only:** For the duration of the link-down clean-up period only, which could be a short period of time on some occasions. See the Rev 1.1 impact section below for more detailed information regarding the duration of the link-down clean-up.
- **Rev 2.1 only:** For the entire duration of the link-down period until the link returns to fully-up status.

During this time, writes to internal CCSR space or local PCI Express configuration registers are handled normally, though the results of the write are not verifiable.

If the controller is configured as an endpoint and receives a hot reset request, it also brings the link down and follows the same clean-up procedures as in an externally detected link-down. Note that reads to PCI Express memory-mapped registers or configuration registers also return all Fs for the duration of the hot reset event.

Impact: **Rev 1.1 only:** Reads to the PCI Express CCSRs or local configuration space return all Fs during link-down clean-up or a hot reset event.

The duration of the link-down clean-up varies depending on the number and type of pending transactions. Clean-up for pending outbound transactions takes just a few cycles, regardless of the source. Pending inbound transactions wait for all responses from targets (data response for reads or successful arbitration to the target queue for writes) before completing clean-up.

Once all pending transactions have been cleared, new PCI Express CCSR accesses and local configurations return to normal operation.

Note that because of PCIe 4 erratum, any accesses to external PCI Express space (including both configure and memory) are queued up for transmission until the link comes back up. Because the configuration access state machine is serialized, a configuration read to an off-chip register prevents access to local CCSRs or local configuration operations while the link remains down.

Rev 2.1 only: Reads to the PCI Express CCSRs or local configuration space return all Fs during the entire period of the link-down or hot reset event.

Workaround: **Both rev 1.1 and rev 2.1:**

Poll the PEX_IP_BLK_REV1 register until it returns a non-F value.

Rev 2.1 only:

Once the link is confirmed as being down, register accesses can be enabled by writing 0x80C0_0000 to the affected PCI Express controller's debug mode register at its memory-mapped offset 0xF00 followed by polling the PEX_IP_BLK_REV1 register until it returns non-

Fs to confirm that the reset of pending transactions is complete (which could be a short period of time on some occasions, depending on the nature of the pending transactions as described in the rev 1.1 silicon impact section).

Once software has completed any needed register accesses, it should return to normal function by writing 0x8040_0000 to the affected PCI Express controller's debug register at its memory-mapped offset 0xF00. It should be noted that if the link is still down at this time, the reads to local PCI Express configuration and memory-mapped CCSR space still return all Fs values until the link is up.

Be aware that while the PCI Express controller is in the debug mode that enables normal register accesses, new reads or writes to PCI Express are not canceled. Software must ensure that no new accesses are sent to PCI Express while this debug mode is enabled.

Fix plan: No plans to fix

PCIe-A001: PCI Express Hot Reset event may cause data corruption

Description: When the PCI Express controller is configured in EP Mode, if the controller detects an in-band Hot Reset event from its upstream device (either RC or Switch) before it finishes processing an inbound memory write TLP, the following may occur.

- TLP received right before the Hot Reset event may be discarded
- Data corruption may occur on the first inbound memory transaction received after the Hot Reset event.

Depending on the type of the first inbound memory transaction received after Hot Reset, data corruption may occur as below:

- If it is a memory write, the transaction may finish with data corruption at the target.
- If it is a memory read, the transaction may be decoded incorrectly and the return data might be incorrect.

Impact: This only affects devices with PCI Express controller configured in EP Mode.

An inbound memory write TLP received by the PCI Express controller in EP Mode may be discarded, if a Hot Reset event is detected while the controller is still in the middle of moving the payload data of this memory write TLP from its receiver buffer toward the packet destination. As a consequence, data corruption may also occur on the first inbound memory transaction received right after this “inbound memory write followed immediately by a Hot Reset event” sequence.

Note that after the data corruption occurs, the system will return to normal operating condition.

If the first inbound transaction received is a configuration cycle, after the above mentioned “inbound memory write followed immediately by Hot Reset event” sequence, the configuration cycle will finish normally, with no error. Since a Hot Reset event resets all the configuration space registers in any PCI Express EP controller, per PCI Express base specification requirements, the upstream RC must re-configure all these registers after the Hot Reset event. Therefore, with the normal PCI Express programming model, there are always configuration cycles before the upstream device can send memory transactions to downstream EP controllers, which means the data corruption scenario after the Hot Reset event might not happen for most applications. However, the Memory Write TLP received immediately before the Hot Reset event might still be discarded. End product designers must check against their application programming model and determine the actual impact and appropriate workaround adoption.

The above described error will not occur in any of the following conditions:

- If the last inbound memory transaction received immediately before the Hot Reset event is a memory read, or,
- If the system (PCI Express controller) is idle in its inbound path when the Hot Reset event occurs.

Workaround: When possible, before issuing the Hot Reset, the upstream RC or Switch should quiesce the system first to ensure no inbound traffic is flowing into the Freescale PCI Express EP controller. This prevents the above mentioned “inbound memory write followed immediately by Hot Reset event” sequence from occurring. The exact requirement and action required to quiesce the system is dependent on system, application and software used. For example, some requirements may include, but not be limited to, using a software semaphore at the RC system side to stop the new memory requests targeting the downstream Freescale PCI Express EP controller from the RC system’s software API layer or DMA controller.

Once such “quiescing system” actions have been finished, the RC system can send a configuration write cycle to clear the Memory Space bit in the Freescale PCI Express EP controller’s Command Register, followed by a several micro-second delay to allow all previously received inbound memory writes to propagate through the EP controller. A Hot Reset command can then be applied.

If an “inbound memory write followed immediately by Hot Reset event” sequence cannot be avoided, do the following. Right after the Hot Reset event, the upstream RC can issue a dummy memory write followed by another write or read to the read-only PEX_IP_BLK_REV1 memory-mapped register in the downstream Freescale device with PCI Express controller configured in EP Mode. The RC can then re-transfer the last memory write TLP that occurred right before the Hot Reset event and resume other normal traffic.

Fix plan: No plans to fix

PIC 1: PCI Express MSI other than interrupt 0 not supported via hardware

Description: PCI Express MSI memory write is defined as a 32-bit write to the address location in the Message Address Register of the MSI Capability Register. The (little-endian) data format of the write is 31:16 - all zeroes, 15:0 from the Message Data Register of the MSI Capability Register, with lower bits modified as necessary to indicate the particular message.

The MPIC implements MSI via the Message Shared Interrupt Index Register (MSIIR), which is big-endian with two fields: Shared Interrupt Register Select (SRS - bits 0:2) and Interrupt Bit Select (IBS - bits 3:7). However, due to this erratum, the MPIC incorrectly implements the MSI logic at MSIIR [24:31] instead of MSIIR [0:7].

The PCI Express Root Complex logic swaps the bytes of inbound writes to big-endian memory space, so the msi_data[31:24] is written to MSIIR[24:31], msi_data[23:16] to MSIIR[16:23], msi_data[15:8] to MSIIR[8:15], and msi_data[7:0] to MSIIR[0:7]. Since msi_data[31:16] are defined as all zeroes, MSIIR[SRS] and MSIIR[IBS] are always set to zero on an MSI write, and all standard MSIs trigger interrupt 0 (MSIR0[SH0]=1, MSISR[S0]=1). The contents of msi_data[15:0] are lost.

Impact: The only MSI that can be triggered through the standard PCI Express hardware mechanism and configuration is interrupt 0.

Workaround: **Option 1:** If no other devices or mechanisms write to the 1 MB memory-mapped register region defined by CCSRBAR through PCI Express, use an inbound ATMU window with the attributes set as follows:

1. Set PEXIWBAR0 to some memory region unused by PCI Express
2. Enable an inbound ATMU (example using window 1):
    a. PEXIWBAR1 = previous value of PEXIWBAR0
    b. PEXIWTAR1[8:31] = CCSRBAR[8:23]||8'b0
    c. PEXIWAR1[0:31] = 0xC0F4_4013 (Enable, No prefetch, Local Memory No snoop, 1 MB window)

Option 2: Use a software or other programmable mechanism to generate the 32-bit MSI memory write with MSI[31:24] set to the value of MSI[7:0]. Duplicating the MSI data in the LSB and MSB allows the workaround to be compatible with future versions of the IP which correct the definition and implementation of MSIIR.

Fix plan: No plans to fix

JTAG 1: TMS requires hold time beyond the fall of TCK

Description: If the SAMPLE/PRELOAD or EXTEST instruction is the current instruction in the JTAG TAP, and the JTAG state machine is moving into the UPDATE-DR state, TMS must be held past the falling edge of TCK.

Impact: This requirement violates the IEEE 1149.1 spec which only requires TMS to be valid at the rise of TCK.

The boundary scan update register will not update, and interconnect testing will fail.

This affects only the SAMPLE/PRELOAD and EXTEST JTAG instructions because these are the only ones required to update the boundary register.

Workaround: **Option 1:** Hold the TMS signal 2ns past the falling edge of TCK.

Option 2: If the SAMPLE/PRELOAD or EXTEST instruction is the current instruction in the JTAG TAP, and the JTAG state machine is in the UPDATE-DR state, move to the SELECT-DR state instead of to RUN-TEST-IDLE.

Moving from UPDATE-DR to RUN-TEST-IDLE requires TMS to change from a '1' to a '0'. In this case, care must be taken to ensure that TMS is held at '1' for 2 ns past the falling edge of TCK before changing to the '0' value.

If the JTAG tool has the option to move to the SELECT-DR state instead of the RUN-TEST-IDLE state from the UPDATEDR state, then the value on the TMS pin does not change, because TMS will be a '1' moving into the UPDATE-DR state and a '1' moving into the SELECT-DR state. Keeping TMS at a '1' value will satisfy the hold time requirement. Then, go through the IR route to go back to the RUN-TEST-IDLE state.

Fix plan: Fixed in Rev 1.1

JTAG 2: Boundary scan test on SerDes transmitter pins needs special requirement incompliant to IEEE 1149.1 specification

Description: The IEEE 1149.1 EXTEST or CLAMP specification requires to drive all output/bidirectional pins to a logical 1 at the same time. However, if this requirement is followed when executing the 1149.1 EXTEST or CLAMP command to drive SerDes transmitter pins, incorrect data will be present on SerDes pins during the boundary scan test.

Impact: A user will have to scan in all zeroes to the boundary cells associated with non-SerDes pins when testing the SerDes pins. There is no requirement in regards to testing all other pins.

Workaround: When executing the 1149.1 EXTEST or CLAMP command to drive SerDes transmitter pins, the user must scan in logic 0 to the boundary cells associated with non-SerDes pins, or configure all non-SerDes pins as inputs.

Fix plan: No plans to fix

GEN 1: Some pins do not meet 500V CDM ESD criteria

Description: CDM (Charged Device Model) ESD testing has shown that SerDes pins only pass the 250-V CDM and do not meet the 500-V CDM ESD criteria.

Impact: None

Workaround: Ensure equipment used in handling of the device is properly grounded. Ensure parts are handled in an environment that is compliant with current ESD standards.

Fix plan: No plans to fix

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: MPC8533ECE
Rev. 6
03/2011