80960Cx Processor

Specification Update

October 1998

Notice: The 80960Cx processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Order Number: 272875-004

Contents

Revision History

Date of Revision	Version	Description
10/05/98	004	Corrected erroneous Status marking (e.g., Fix, NoFix) for errata items #17, #18, #28, #36, #40, #42
04/10/98	003	Added Errata #45.
06/30/97	002	Added Errata #44.
07/01/96	001	This is the new Specification Update document. It contains all identified errata published prior to this date.

Preface

As of July, 1996, Intel's Computing Enhancement Group has consolidated available historical device and documentation errata into this new document type called the Specification Update. We have endeavored to include all documented errata in the consolidation process, however, we make no representations or warranties concerning the completeness of the Specification Update.

This document is an update to the specifications contained in the Affected Documents/Related Documents table below. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Information types defined in Nomenclature are consolidated into the specification update and are no longer published in other documents.

This document may also contain information that was not previously published.

Affected Documents/Related Documents

Title	Order #
80960CA 32-Bit High-Performance Embedded Processor datasheet	270727
80960CF-40, -33, -25, -16 32-Bit High-Performance Superscalar Embedded Microprocessor datasheet	272886
i960® Cx Microprocessor User's Manual	270710
i960® Cx Microprocessor User's Manual - Instruction Set Quick Reference	272220

Nomenclature

Errata are design defects or errors. These may cause the 80960CA/CF's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

Note: Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

Summary Table of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the 80960CA/CF product. Intel may fix some of the errata in a future stepping of the component, and account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Errata exists in the stepping indicated. Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Page

(Page):	Page location of item in this document.

Status

Doc:	Document change or update will be implemented.
Fix:	This erratum is intended to be fixed in a future step of the component.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
Eval:	Plans to fix this erratum are under evaluation.

Row

|	Change bar to left of table row indicates this erratum is either new or modified from the previous version of the document.

Errata (Sheet 1 of 3)

New No.	Prev No.	Steppings						Pg	Status	ERRATA
		80960CA			80960CF					
		C2	C3	D	B	C	E			
1	A2	X	X					12	Fixed	A2 Bus Backoff and READY#
2	A5A	X	X					12	Fixed	A5A DACK Timing Vs Ready Wait States
3	A5B	X	X					12	Fixed	A5B DACK Timing with Multiple Accesses Per Bus Request
4	A18	X	X					12	Fixed	A18 DMA SSDEM Mode Packing and EOP
5	A19	X	X					13	Fixed	A19 Suspend DMA, EOP and SDMA
6	A20A	X	X					13	Fixed	A20 A EOP, SDMA and Multi-Channel DMA Operations
7	A20B	X	X					14	Fixed	A20 B EOP, SDMA and Multi-Channel DMA Operations
8	A21	X	X					14	Fixed	A21 Disabling Branch Lookahead Causes DMA problems
9	A22	X	X					14	Fixed	A22 NMI is Level-Triggered after RESET
10	A25	X	X		X			14	Fixed	A25 Interrupts During SYSCTL Instruction to Load and Lock the Instruction Cache
11	A26	X	X		X			15	Fixed	A26 32-32 Bit Transfers in Source-Synchronized Demand Mode with Unaligned Destination Address
12	A27				X			15	Fixed	A27 Terminal Count (TC) pins are non-functional when using any channel in source synchronized demand mode or block mode
13	A28	X	X		X			15	Fixed	A28 DMA and Instruction Scheduler Interaction
14	A29	X	X		X			17	Fixed	A29 Destination Synchronized 128- to 128-Bit Quad-Word Transfer Mode
15	A30				X			17	Fixed	A30 Disabling the Instruction Cache
16	A31				X			18	Fixed	A31 Interrupt Can Cause User Process Stall when DMA Active Under Certain Complex Conditions
17	A32			X				18	NoFix	A32 Register Cache
18	B1	X	X					18	Fixed	B1 Cache Functionality/Re-fetching Cached Instructions
19	B3	X	X					19	Fixed	B3 COBR Branch Trace
20	B5	X	X					19	Fixed	B5 EOP and DREQ Deassertion
21	B6	X	X					19	Fixed	B6 DREQ Sampling

Errata (Sheet 2 of 3)

New No.	Prev No.	Steppings						Pg	Status	ERRATA
		80960CA			80960CF					
		C2	C3	D	B	C	E			
22	B8	X	X					19	Fixed	B8 SYSCTL followed by RET May Incorrectly Fault
23	B10	X	X					20	Fixed	B10 Terminal Count Operation
24	B11A	X	X					20	Fixed	B11A Testif and Faultif Cause Incorrect Branch Trace
25	B11B	X	X					20	Fixed	B11B Branch Tracing Misses Branch Instruction
26	B12	X	X					21	Fixed	B12 Disabling Interrupts with sysctl is non-atomic
27	B13	X	X		X			21	Fixed	B13 Unaligned DMA Transfer Modes When Using Incrementing Source and Destination Address
28	B14				X	X	X	21	NoFix	B14 Data Cache Global Disable Bit is 0 After Device Reset
29	C1	X	X	X	X	X	X	21	NoFix	C1 Instruction Cache
30	C2	X	X		X	X	X	21	NoFix	C2 Register Cache
31	C3	X	X	X	X	X	X	22	NoFix	C3 Data Address Breakpoint Fault on a CALLX
32	C4	X	X	X	X	X	X	22	NoFix	C4 Data Address Breakpoints on Stacks and Tables
33	C5	X	X	X	X	X	X	22	NoFix	C5 Pipelined Region Limitation
34	C6	X	X	X	X	X	X	22	NoFix	C6 EOP and Buffer Complete Interrupt
35	C7	X	X	X	X	X	X	22	NoFix	C7 MULI Fault Return
36	C8	X	X		X			23	Fixed	C8 (B7) Fault Handler Executes before Interrupt Handler
37	C9	X	X	X	X	X	X	23	NoFix	C9 (A17) Extra RIP Read
38	C10	X	X	X	X	X	X	23	NoFix	C10 (B9) Disabling and Enabling Interrupts via Modifying the Interrupt Mask Register (sf1)
39	C10b	X	X	X	X	X	X	23	NoFix	C10b Microcoded Instructions Can Be Corrupted after Writing to Interrupt Mask Register (sf1)
40	C11	X	X		X			24	Fixed	C11 NMI during Built-In-Self-Test (BIST)
41	C12	X	X		X			24	Fixed	C12 BTERM# Functionality
42	C13	X	X	X		X	X	25	NoFix	C13 Modifying the Previous Frame Pointer (PFP) before Returning

Errata (Sheet 3 of 3)

New No.	Prev No.	Steppings						Pg	Status	ERRATA
		80960CA			80960CF					
		C2	C3	D	B	C	E			
43				X		X	X	26	NoFix	Erroneous TC Can Be Signaled When Using Multiple DMA Channels
44		X	X	X	X	X	X	26	NoFix	Supervisor Access Pin Operation During System-Supervisor Fault Handling
45		X	X	X	X	X	X	27	NoFix	Power Supply Sequence Can Damage Internal Diodes

Specification Changes

No.	Steppings			Page	Status	SPECIFICATION CHANGES
						None for this revision of this specification update.

Specification Clarifications

No.	Steppings			Page	Status	SPECIFICATION CLARIFICATIONS
						None for this revision of this specification update.

Documentation Changes

No.	Document Revision	Page	DOCUMENTATION CHANGES
1	270710-003	30	Chapter 4 - Instruction Set Summary
2	270710-003	30	Chapter 9 - Instruction Set Reference
3	270710-003	30	Chapter 11 - External Bus Description
4	270710-003	30	Chapter 12 - Interrupt Controller
5	270710-003	30	Chapter 12 - Page 12-22, Table 12-2
6	270710-003	31	Chapter 13 - DMA Controller
7	270710-003	31	Appendix A - Instruction Execution and Performance Optimization
8	270710-003	31	Appendix A - Page A-42, Section A.2.6.10
9	270710-003	31	Appendix B - Bus Interface Examples
10	270710-003	31	Appendix F - Register and Data Structures
11	272220-002	31	divi Instruction Issue changed in Quick Reference
12	272220-002	31	modac Instruction Issue changed in Quick Reference

Identification Information

Markings

80960CA/CF processors may be identified electrically according to device type and stepping. Refer to the data sheet for instructions on how to obtain the identifier number.

The various 80960CA steppings are identified by a topside mark as indicated below.

	C-2 Stepping	C-3 Stepping	D-Stepping
KU 80960CA -25	SV914	SW033	D2
KU 80960CA -16	SV913	SW032	D2
A 80960CA-33	SV908	SW031	D2
A 80960CA-25	SV907	SW030	D2
A 80960CA-16	SV906	SW029	D2
TA 80960CA-16		SW147	VA80960CA-16[1]

NOTE:

1. TA80960CA16,S W147 will no longer be offered. It is replaced by VA80960CA16, which has a temperature range of -40℃ to +125℃, and manufactured on the D-stepping. In addition, a 25 MHz extended temperature part will be offered as VA80960CA25.

Errata

1. A2 Bus Backoff and READY#

Problem: If either RDY# or BTERM# is asserted between the deassertion of BOFF# and the completion of the ADS# strobe, then the regenerated access will be lost or corrupted.

Implication: Improper operation and/or data corruption may result.

Workaround: The work around is to ensure that RDY or BTERM# is not asserted from the time BOFF# is asserted until the time that ADS# is regenerated.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

2. A5A DACK Timing Vs Ready Wait States

Problem: The deassertion of DACK can be triggered by conditions other than the end of a Ready access. Because DACK is used as a chip select, simple connection of DACK to system DACK does not work for Ready controlled regions.

Implication: Additional glue logic is required.

Workaround: External logic is necessary for proper operation. This logic should latch DACK during the address cycle.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

3. A5B DACK Timing with Multiple Accesses Per Bus Request

Problem: DACK deasserts early on multiple bus accesses per bus request during the following conditions. If the DMA transfer type and external bus width are unequal so that the Bus Controller is required to issue multiple bus accesses per request, then DACK deasserts on the rising edge of BLAST of the first access and stays deasserted for the duration of the request. For example, a DMA transfer type of 128 by 128 requires the Bus Controller to issue quad stores on a non-burst 32-bit external bus. Therefore DACK asserts with the first ADS and deasserts on the rising edge of BLAST of the first access.

Implication: Improper DMA transfers may occur.

Workaround: Match the DMA mode transfer width with the external bus width. The correct functioning of DACK is as follows: DACK should assert at the falling edge of ADS and remain asserted throughout the entire bus request. DACK should deassert at the rising edge of BLAST of the last access.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

4. A18 DMA SSDEM Mode Packing and EOP

Problem: Asserting EOP during a Source Synchronized Demand Mode DMA can corrupt the last word or byte of the transfer. The following table shows the packing modes and how the SAR (Source Address Register) and DAR (Destination Address Register) are aligned in order to manifest this condition. The EOP failure point denotes where a failure will occur with respect to EOP and individual transfers.

Packing type	SAR	DAR	EOP Fail Point
16-32	Aligned	Aligned	All Xfers
8-16	-	Aligned	1st Xfer Only
16-16	Unaligned[1]	Aligned	1st Xfer Only
32-32	Aligned	Unaligned	1st Xfer Only

NOTE:
1. Block Mode

Implication: Data corruption may occur.

Workaround: There are three workarounds. The first one is to not use EOP but rather an interrupt to signal an asynchronous termination of a transfer. Second, if it is known that the above condition is going to happen, then throw away the last word or byte of the transfer and adjust the SAR and the BCR (Byte Count Register) accordingly. Third, for those packing modes where EOP can only cause a failure during the first transfer, a dummy transfer can be inserted as the first transfer and no data will be corrupted.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

5. A19 Suspend DMA, EOP and SDMA

Problem: An unrecoverable error can occur during the time that an interrupt is being posted if the Suspend DMA function is enabled (ICON bit 15) and either an EOP or SDMA event occurs. The following conditions must exist for the error to occur:

1. Suspend DMA function is enabled by setting ICON Register bit 15.
2. Two or more DMA channels are operating at the same time.
3. An interrupt is actually being posted.

4a. An EOP is received or,

4b. An SDMA is executed.

Implication: Unrecoverable error state may occur.

Workaround: There are several workarounds. The Suspend DMA function should be enabled only if it can be guaranteed that an EOP or an SDMA will not occur. An SDMA can occur by first disabling the Suspend DMA function by resetting ICON Register bit 15. EOP, however, due to its asynchronous nature, cannot be used with the Suspend DMA feature enabled. Instead, a separate external hardware interrupt should be used to signal the end of a channel. An interrupt handler can then disable the channel by clearing the enable bit in the DMAC (DMA Command Register sf2) and wait until the active bit in the DMAC is cleared.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

6. A20 AEOP, SDMA and Multi-Channel DMA Operations

Problem: When a Setup DMA (SDMA) instruction is issued, transfer requests on all four channels are held off until SDMA completion. It is possible to enter a state in which a DMA Load request has completed but not the corresponding Store. Once the setup is done, the pending Store request will be serviced and completed. However, during the time that the SDMA instruction is executing, an EOP on the same channel may be accepted by the DMA controller. The EOP will be serviced BEFORE the DMA cycle has completed. The Store access will never be executed.

Implication: The transfer may be terminated prematurely and the data that is expected on the last Load may be lost.

Workaround: The workaround for this anomaly is to make sure that an EOP does not occur during an SDMA instruction. An alternate method of setting up a DMA channel is to use the Chaining mode with wait. However, the user cannot change the DMA mode this way.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

7. A20 BEOP, SDMA and Multi-Channel DMA Operations

Problem: If during the time that an SDMA is issued for a channel that was the last active channel, an EOP occurs on another channel, then the Byte Count, Source Address, and Next Pointer Address pertaining to the SDMA may be corrupted.

Implication: Data corruption may occur.

Workaround: The workaround for this anomaly is to make sure that an EOP does not occur during a SDMA instruction.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

8. A21 Disabling Branch Lookahead Causes DMA problems

Problem: When using multiple DMA channels, bit 21 of the word at PRCB offset 0x20 must be zero (0). Otherwise, DMA data corruption occurs.

Implication: When using older debug monitors, DMA corruption can occur.

Workaround: Make sure bit 21 of the PRCB offset 0x20 (instruction cache configuration word) is zero (0). This disables the branch lookahead logic, so some decrease in performance may be observed.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

9. A22 NMI is Level-Triggered after RESET

Problem: NMI (Non-Maskable Interrupt) normally only occurs after a high-to-low (1-to-0) transition on the NMI pin. After RESET, however, NMI is level-triggered and a low (0) on the NMI pin will generate a Non-Maskable Interrupt.

Implication: Spurious NMI could occur.

Workaround: Make sure the NMI pin is driven high during the RESET sequence. After RESET is deasserted, (driven high), the user must drive NMI high within 10 clocks. Otherwise, a Non-Maskable Interrupt will occur.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

10. A25 Interrupts During SYSCTL Instruction to Load and Lock the Instruction Cache

Problem: Any hardware interrupts (maskable or unmaskable) which occur during the execution of a **SYSCTL** instruction to load and lock the instruction cache causes the processor to malfunction. **SYSCTL** instructions have the configure cache message and either the load and lock 1 Kbyte or load and lock 512 bytes cache mode configuration. The processor actually enters the interrupt handler, but enters it in an unrecoverable state. The only way to completely recover from this malfunction is to reset the device.

Implication: Processor can enter an unrecoverable error state.

Workaround: The workaround is to disable hardware interrupts while executing the **SYSCTL** instruction to load and lock the instruction cache. The user must also guarantee that his system never generates an NMI during the **SYSCTL** instruction.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

11. A26 32-32 Bit Transfers in Source-Synchronized Demand Mode with Unaligned Destination Address

Problem: The DMA controller is optimized to perform unaligned 32-32 bit transfers. However, when in source-synchronized demand mode, the DMA controller requires an extra DREQ# to complete the transfer when the destination address is unaligned.

For example, assume an aligned source address and a destination address which is unaligned by one byte (has an address of xxxxxxxx1) and a byte count of 16. The DMA controller should transfer 3 bytes for the first DREQ#, 4 bytes for the two middle DREQ#s, and 5 bytes for the last DREQ#. However, only 4 bytes are transferred for the 4th DREQ# and requires a 5th DREQ# to transfer the last byte.

All other source synchronized transfer modes with an unaligned destination address will work correctly (i.e., 16-16, 8-8, etc.).

Implication: Improper unaligned DMA behavior may result.

Workaround: The workaround is to always ensure that your destination address is aligned on a 4 byte boundary, or have your hardware generate an extra DREQ#.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

12. A27 Terminal Count (TC) pins are non-functional when using any channel in source synchronized demand mode or block mode

Problem: When any EOP/TC# pin is configured as an output (terminal count function), it may not ever be asserted if any of the four DMA channels is configured for either source synchronized demand mode, or block mode. The TC pin works correctly when all DMA channels are programmed only for destination synchronized demand mode or fly-by modes. This errata does not affect the operation of the EOP/TC# pin when programmed for the EOP function.

The TC pin will not be asserted when any unsynchronized DMA store is present on the external bus and the DMA controller issues the last load (the load in which TC should be asserted) of a source synchronized demand mode or block mode transfer. Note that even though TC is not asserted, the DACK# pin is asserted correctly for the load. An unsynchronized DMA store is defined to be a DMA store in which the DACK# pin is not asserted, as in source synchronized demand mode and block mode transfers.

Implication: For application purposes, this renders the TC pins non-functional when using these modes.

Workaround: None

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

13. A28 DMA and Instruction Scheduler Interaction

Problem: The DMA controller is implemented primarily in microcode. DMA operations are executed in microcode while providing core bandwidth for the user's program. Core resources are shared by implementing a separate hardware process for each DMA channel and one for the user code. This hardware mechanism allows the core to switch processes on clock boundaries to service either a DMA process or the user process. See Chapter 13 in the *i960® Cx Microprocessor User's Manual.*

Under very specific circumstances, a DMA process can cause a conditional branch instruction to branch along the incorrect path. The branch instruction unconditionally follows the path dictated by the branch prediction bit. The circumstances are:

1. The instruction scheduler must attempt to issue three instructions in parallel (a Reg-Mem-conditional branch triplet). For the device to execute three instructions in parallel, they must be in this order. For example:

```
CMPI    r4,r5;reg-side must affect condition codes

ST      r6,(r7)

BGE.f   some_where;must be conditional branch
              ;which relies on condition codes
```

2. The reg-side instruction must begin on an odd word boundary.
3. The mem-side must be scoreboarded; i.e., the ST instruction cannot execute because of a resource limitation. The conditions in which the mem-side is scoreboarded are covered in the *i960® Cx Microprocessor User's Manual*.
4. A DMA event must occur while the mem-side is scoreboarded. A DMA event is defined to be the assertion of DREQ# on an active DMA channel and causing a process switch to a DMA task.
5. The branch prediction bit for the branch instruction must be incorrect for the result of the reg-side comparison.
6. Branch Lookahead must be enabled.

The instructions affected are listed below. Any combination of these instructions in the reg-mem-ctrl ordering are affected.

REG-side	Mem side	Conditional
ADDC	LDA offset	BE, BNE, BL, BLE
CMPI,CMP0	LDA (reg)	BG, BGE, BO, BNO
CMPDECI,CMPDECO	LDA offset(reg)	
CMPINCI,CMPINCO		
CONCMPI,CONCMPO	LD/ST offset	
CHKBIT	LD/ST (reg)	
SCANBYTE		
SUBC		

The LD/ST instructions include the byte, short, word, triple, and quad versions. The chances of this anomaly occurring in an application are exacerbated when the processor is in NIF mode, because the mem-side is always scoreboarded until the reg-side compare instruction is finished.

Implication: Incorrect program operation may occur.

Workaround: For the 80960CA A through C-3 stepping, errata A21 prevents simply disabling Branch Lookahead for applications which have more than one DMA channel active at a time. Multichannel DMA applications will have to prevent this code sequence in their application to workaround this errata for the 80960CA. Intel will also provide switches for the development tools (C compilers) which will prevent this sequence of instructions from taking place, as well as a method for screening existing object code for the instruction triplet. Please contact your local Intel support person for information on obtaining these tools.

Code workaround - Because it is very difficult to predict when the mem-side will be scoreboarded, code workarounds should remove all cases of the code triplet. This guarantees that the errata will not occur in an application.

The simplest workaround is to insert a nop (mov g0,g0) between the reg-side and mem-side instructions. This prevents the triplet from being issued in parallel. The Intel tools will provide this kind of workaround as an assembler patch. An alternate workaround is to reorder the instructions so the triplet is not issued in parallel. A third workaround is to combine the separate reg-side and conditional branch instructions into a single compare and branch (cobr) instruction.

```
CMPI    r4,r5;insert a nop
MOV     g0,g0
ST      r6,(r7)
BGE.f   some_where

ST      r6,(r7);reorder the instructions
CMPI    r4,r5
BGE.f   some_where

ST      r6,(r7);Use a cobr instruction
CMPIBGE.f r4,r5,some_where #.
```

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

14. A29 Destination Synchronized 128- to 128-Bit Quad-Word Transfer Mode

Problem: The destination synchronized multi-cycle quad-word transfer mode will execute the final transfer (last sixteen bytes) immediately without waiting for the final DREQ#. A 32 byte transfer, for example, will only require a single DREQ# to transfer both quad words. The single DREQ# also causes two DACKS# to be asserted. When the DMA is programmed to transfer a single quad word (16 bytes), DMA will work correctly and require a single DREQ#. This erratum does not affect the fly-by, block, or source synchronized 128- to 128-bit quad word transfer modes.

Implication: Synchronous DMA operation may occur.

Workaround: Do not use destination synchronized multi-cycle quad-word DMA.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

15. A30 Disabling the Instruction Cache

Problem: The instruction cache cannot be disabled while the DMA is active. Prefetch algorithms operate differently with the instruction cache disabled. An interrupt, DMA event, and instruction prefetch all occurring around the same time can infinitely scoreboard the prefetch, causing user code to stop executing at the prefetch. This erratum cannot happen while the instruction cache is enabled.

Implication: No stepping of the 80960CA is effected by this errata.

Workaround: The only way to recover from this condition is to reset the device.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

16. A31 Interrupt Can Cause User Process Stall when DMA Active Under Certain Complex Conditions

Problem: An interrupt request can cause the user program to stop execution when several specific events occur in sequence under strict timing requirements. The full explanation of this condition can be found in the paper titled “80960CF User Process Stall Bug” available through Intel application's FaxBack system; document number 2050. The FaxBack telephone number is 1-800-628-2283 or 916-356-3105 from outside the U.S. and Canada. The conditions required to cause the stall are:

1. A REG-format instruction followed by an “invalid” word in the cache. An “invalid” word is a word that was not written into the cache due to a change of program flow. This REG-format instruction must be on an even word (address ending in 0x0 or 0x8).
2. An instruction fetch issued in parallel with the execution of this REG-format instruction.
3. MEM side of CPU scoreboarded (e.g., due to bus queues full) causing fetch to be canceled.
4. DMA process switch occurring at least 3 clocks after the fetch, and MEM-side must still be scoreboarded.
5. Interrupt request occurs while MEM-side still scoreboarded.

This is most likely to occur under heavy bus traffic with 0 data to data wait states, heavy DMA activity, and heavy interrupt activity.

Implication: Only the 80960CF B step is affected.

Workaround: DMA activity will continue until the DMA transfer is “done” (e.g., null chaining pointer, zero byte count, EOP). To recover from this condition, reset the device.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

17. A32 Register Cache

Problem: Programming a register cache size of zero (0) causes the processor to generate an operation fault while performing a return. During proper operation, a value of zero (0) should yield one (1) set of register cache.

Implication: Improper program operation may occur.

Workaround: Program the register cache size with a value of one (1). A value of zero (0) or one (1) yields one (1) set of register cache.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

18. B1 Cache Functionality/Re-fetching Cached Instructions

Problem: If loops exceed the cache size (256 instructions), even by one instruction, then all instructions in that loop will be fetched again. This can be eliminated by setting bit 31 of the PRCB offset 0x20 to a one. However, setting this bit may impact performance. As a rule of thumb, it would be best to operate with bit 31 of the PRCB offset 0x20 cleared unless this cache re-fetching is clearly affecting the performance of the design.

Further analysis of how the 80960CA fetches instructions shows that other conditions exist where instructions are fetched from memory instead of being executed from the cache. This is a result of a change that improves the performance of the prefetch mechanism. Instructions are fetched into an instruction queue before they are executed. If an instruction is executed, then it is loaded into the cache. If an instruction is not executed because of a branch or a call being taken, then that instruction may not be loaded into the cache. This may leave instructions in the cache that are invalid, although the tag for the current cache line is valid. Extra fetches occur because of how the prefetch mechanism handles an invalid instruction when no cache miss occurs.

This erratum may affect performance as much if not greater than the setting of bit 31 mentioned above. Bit 31 enables or disables the original slow prefetching algorithm used in previous steppings.

Be aware that even with the above problem corrected, there is the possibility of seeing instructions being fetched that would appear to have been fetched previously. This extra fetching occurs because the cache is only guaranteed to be loaded if an instruction is executed, not just fetched.

Implication: Performance may be negatively impacted.

Workaround: Setting bit 31 is considered the best workaround.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

19. B3 COBR Branch Trace

Problem: A Branch Trace may be incorrectly reported on a cobr (Compare and Branch) instruction that is scoreboarded by a previous instruction. This occurs only when trace faults are enabled. Because the compare is scoreboarded, the branch never takes place; however, the trace fault is still reported.

Implication: Debugger may report branch trace when one did not occur.

Workaround: Do not order a cobr instruction just after instructions such as multiply where one could possibly be using resources needed by the cobr instruction.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

20. B5 EOP and DREQ Deassertion

Problem: A DMA cycle may terminate prematurely if the DMA controller accepts a DMA request and EOP occurs while DREQ is still asserted. Data may be buffered internally but never stored.

Implication: Data corruption may occur.

Workaround: Do not assert EOP while DREQ is asserted.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

21. B6 DREQ Sampling

Problem: In some DMA modes, the DREQ signal is sampled when the internal wait-state generator has timed out, regardless of the state of the external READY signal. This could lead to extra DMA cycles being generated erroneously by the incorrect sampling of DREQ during the time that READY is deasserted by an external memory system. Sampling of DREQ does not depend on the external system trying to complete a slow memory cycle; i.e., DREQ sampling does not depend on external READY.

Implication: Extra DMA transfers may occur.

Workaround: Make sure DREQ is de-asserted before wait-state timer expires.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

22. B8 SYSCTL followed by RET May Incorrectly Fault

Problem: A **sysctl** instruction followed by a **ret** instruction with no instructions in between may cause the false generation of an OPERATION.UNIMPLEMENTED fault.

Implication: User software operation may be negatively impacted.

Workaround: Insert an instruction other than **ret** between sysctl and Ret instruction.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

23. B10 Terminal Count Operation

Problem: On the 80960CA only (80960CF not affected): on steppings previous to the D-step the Terminal Count (TC) pin timing does not become active at the specified time. TC should have the same timing as DACK#, but does not. The following describes the condition:

Source Synchronized Demand Mode (SSDEM): The TC bit will not become active until all stores related to the last access are issued. For most aligned transfers in which there is no unpacking, the number of clock cycles between the deassertion of the last DACK and the active edge of TC will be 20. This assumes the 4:1 DMA cycle mode.

Block Mode: Same as SSDM with above.

Destination Synchronized Demand Mode (DSDEM): The last store is synchronized with the external bus. When the DACK deasserts, there are approximately 17 cycles to the active edge of TC.

The TC pin externally indicates when the DMA has fully completed its transfer and is no longer active. The TC bit in the DMA control register is set in parallel with pin activation.

Current Operation: On the D step, which is not affected by this erratum, TC is active during the last DACK# of that channel and has the same timing as DACK#. For Source Synchronized transfers, TC is active during the last load. For Destination Synchronized transfers, TC is active during the last store.

If the last load/store bus request is executed as multiple bus accesses (such as a quad store to a non burst bus or unequal bus widths), then DACK# and TC are active for the entire bus request (multiple accesses) with only one negative edge and one positive edge. An exception is if Nxda wait states are used. In this case, TC is only active for the first access. Note that TC is not a one clock cycle pulse but a pulse as long as the DACK.

Implication: Performance may be negatively impacted.

Workaround: Rely on TC only after waiting the appropriate amount of time.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

24. B11A Testif and Faultif Cause Incorrect Branch Trace

Problem: The testif and faultif instructions may cause branch traces.

Implication: Debugger may report branch trace when one did not occur.

Workaround: None.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

25. B11B Branch Tracing Misses Branch Instruction

Problem: When modpc is used to enable tracing and the Branch Tracing bit is set in the Process Control Register, a branch in parallel with the modpc instruction is not traced. If tracing is already enabled and modtc is used to turn on branch tracing, then the same thing happens with a branch in parallel with the modtc.

Implication: Debugger may miss a branch breakpoint.

Workaround: Separate the branch instruction from modtc or modpc by at least one instruction.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

26. B12 Disabling Interrupts with sysctl is non-atomic

Problem: When the sysctl instruction is used to disable interrupts, there is a two-clock period window of time where interrupts can be received and posted.

Implication: User software must take this period into account.

Workaround: None.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

27. B13 Unaligned DMA Transfer Modes When Using Incrementing Source and Destination Address

Problem: Existing DMA functionality limits the alignment address of the synchronizing device to the DMA transfer width; transfer width is equal to the synchronized side of the DMA transfer. For example, for a 32-32 source synchronized transfer, the source address must be aligned to a word boundary (synchronizing side), while the destination can be aligned to a byte boundary. For a 16-32 destination synchronized transfer, the destination must be aligned to a word boundary, and the source address can be aligned to a byte boundary.

Byte count must also be aligned to the transfer width boundary or evenly divisible by the transfer.

Implication: Incorrect DMA operation will result.

Workaround: Use only aligned source addresses.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

28. B14 Data Cache Global Disable Bit is 0 After Device Reset

Problem: The data cache comes up globally enabled after device reset.

Implication: Data cache operation may not function as expected.

Workaround: Applications that need the data cache disabled after a device reset should disable the data cache by executing a setbit 30,sf2,sf2 before referencing any data that resides in a cacheable region.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

29. C1 Instruction Cache

Problem: When the instruction cache is disabled, two cache lines (16 words) of the cache remain enabled. These two lines are not part of the 1024 byte cache.

Implication: Instructions of an extremely tight loop may be cached even when cache is disabled.

Workaround: None.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

30. C2 Register Cache

Problem: Local Register Cache Size - programming a register cache size of 0 causes 15 sets to be allocated. During proper operation, the register cache should be disabled by programming 0 frames.

Implication: User software must take this restriction into account.

Workaround: Use data cache enable bit to disable data cache.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

31. C3 Data Address Breakpoint Fault on a CALLX

Problem: When a Data Address breakpoint fault occurs on a CALLX, or any call with a frame flush, the return IP (RIP) reported will be that of the call. Per specification, the RIP should point to the first instruction of the called procedure.

Implication: Debugger must provide additional logic.

Workaround: The trace fault handler must detect this condition and adjust the RIP before returning.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

32. C4 Data Address Breakpoints on Stacks and Tables

Problem: If a data address breakpoint occurs on a memory access associated with the processor's interrupt or fault context switches or execution of a CALLS instruction, then the fault may not be signaled. If it is signaled, then the associated fault record may be incorrect and the Trace Controls register (TC) may be corrupted. During proper operation, the data address breakpoint fault would be signaled after completion of all operations associated with these microcoded sequences.

Implication: This may result in improper debugger behavior.

Workaround: Data address breakpoints should not be set on the system procedure table, fault table, interrupt table, or stack locations which will contain interrupt records or fault records.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

33. C5 Pipelined Region Limitation

Problem: Each pipelined region which has burst enabled must have Ready Control disabled in that region. During proper operation, the ready pins are ignored during reads in a pipelined region but can be used in a write to a pipelined region.

Implication: User software must take this restriction into account.

Workaround: Disable READY for pipelined-burst memory regions.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

34. C6 EOP and Buffer Complete Interrupt

Problem: There is no way to distinguish between an interrupt caused by an EOP and the interrupt that occurs at the end of a buffer in a DMA transfer when using source AND destination chaining.

Implication: User is not provided enough information about the interrupt.

Workaround: To distinguish between the two interrupts, connect EOP to an interrupt pin.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

35. C7 MULI Fault Return

Problem: A situation can occur where a MULI instruction generates a fault and the next instruction after the MULI is a complex memory instruction which is scoreboarded from some previous instruction. When the fault handler microcode executes, it may not return a correct RIP and, therefore, the complex memory instruction may never complete. All that is guaranteed is that the fault handler returns the proper IP for the MULI instruction.

Implication: Users must be aware of this fault handler restriction.

Workaround: Fault handler for MULI should explicitly adjust RIP.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

36. C8 (B7) Fault Handler Executes before Interrupt Handler

Problem: The *i960® Cx Microprocessor User's Manual* states that if an interrupt occurs at the same time that an instruction generates a fault, the interrupt will be serviced first, then the fault handler executes. Currently, the handler is executed first, then the interrupt service routine is executed.

Implication: This could result in abnormally long interrupt latencies and negatively impact performance.

Workaround: None.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

37. C9 (A17) Extra RIP Read

Problem: The return instruction (RET) does not perform as documented in the *i960® Cx Microprocessor User's Manual*. At the end of RET execution, an extra RIP (Return Instruction Pointer) read from the previous Stack Frame occurs. This may cause problems if the current PFP (Previous Frame Pointer) is not valid, such as during cold Reset. If the memory region that the PFP points to has Ready enabled and no memory exists in that region, then the processor hangs.

Implication: User software must take this into account.

Workaround: Make sure that the PFP is valid.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

38. C10 (B9) Disabling and Enabling Interrupts via Modifying the Interrupt Mask Register (sf1)

Problem: One way to enable and disable interrupts is to mask interrupts by controlling certain bits of the Interrupt Mask Register (IMSK or sf1). Setting appropriate bits of sf1 to zero masks individual interrupts at the input pins. This occurs as soon as the IMSK register is modified by an instruction such as setbit or move. However, interrupts that occur at or just before the IMSK register is modified can still be serviced for up to eight clock cycles after the execution of the instruction that modifies the IMSK register.

Implication: User software must take this into consideration.

Workaround: To ensure that critical code is not interrupted, insert eight no-op (mov g0, g0) instructions immediately after the instruction modifying the IMSK register. Conversely, after unmasking interrupts, it takes four clock cycles after modifying the IMSK register to generate interrupts.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

39. C10b Microcoded Instructions Can Be Corrupted after Writing to Interrupt Mask Register (sf1)

Problem: One way to mask or disable interrupts is to clear certain bits in the Interrupt Mask Register (IMSK or sf1). If a hardware interrupt occurs during the same clock in which it was masked by a write to IMSK, then any microcoded instruction executed within eight clock cycles following the write to IMSK will be corrupted.

Implication: User software must take this into consideration.

Workaround: Make sure that no microcoded instructions are executed in eight clocks following any write to the IMSK register. The microcoded instructions are covered in Appendix A of the user's manual. It is probably best to simply insert 8 no-op (mov g0,g0) instructions after writing to the IMSK register, as this is also the workaround for errata C10.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

40. C11 NMI during Built-In-Self-Test (BIST)

Problem: If an NMI occurs during BIST, then the processor hangs and does not recover until the processor is reset.

Implication: Improper processor state may result.

Workaround: Make sure NMI cannot occur during processor self test.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

41. C12 BTERM# Functionality

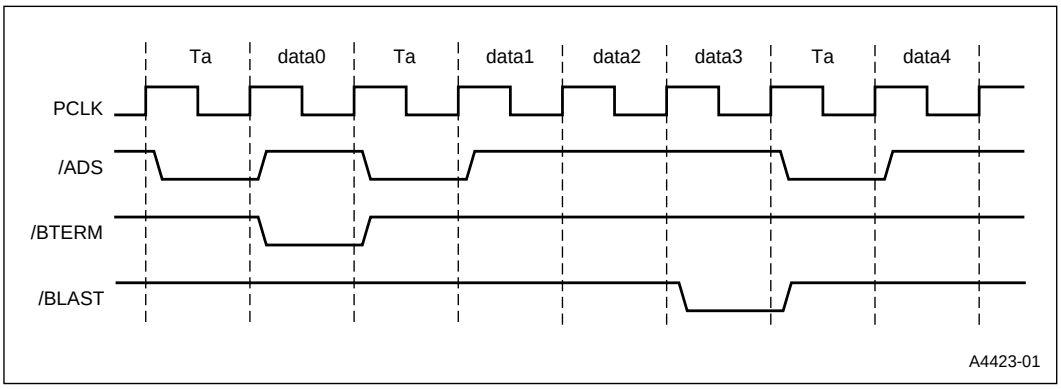
Problem: This applies to 8-bit and 16-bit mode. This is best described by example:

```
lda 0x0, r3
ldq (r3), r4
```

Correct Operation

If BTERM# is asserted during data(0), then ADS# will be asserted again for data(1) and the access will end by asserting BLAST# while reading data(3).

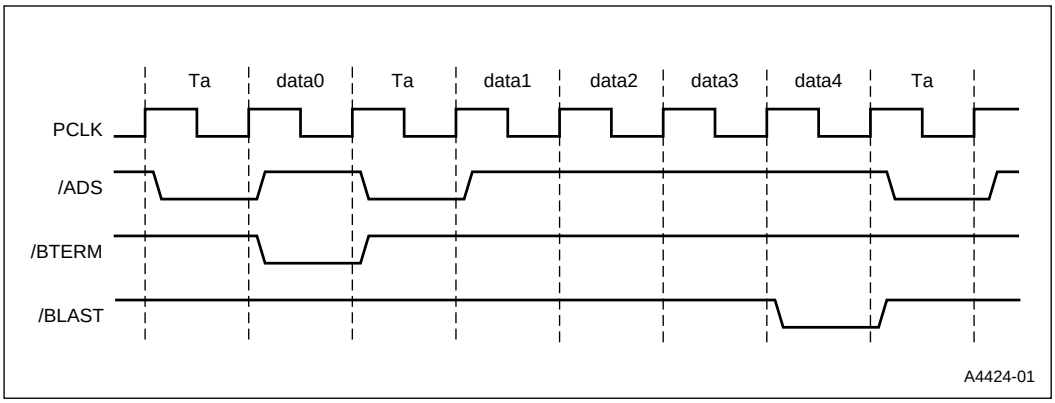
Figure 1. Timings for Correct Operation



Incorrect Operation

The access spans over data3. For example, BLAST# is asserted while data4 is read.

Figure 2. Timings for Incorrect Operation



Implication: Memory and I/O systems must accept these forms of access when using BTERM.

Workaround: None.

Status: Fixed. Refer to Summary Table of Changes to determine the affected stepping(s).

42. C13 Modifying the Previous Frame Pointer (PFP) before Returning

Problem: If a return is performed immediately after the PFP's three least significant bits (Encoding of Return Status Field) is modified, then the return instruction may operate improperly. This can occur because the return instruction may not see the new value in the PFP. This is most likely to occur when instructions are cached. See Example 1.

Example 1:

```
or g1, 7, pfp/* modify return status in PFP */
ret
```

Implication: User software must take this into consideration.

Workaround: PFP modification must occur at least three clock cycles before the return instruction is executed. One solution is to place three no-op (mov g0, g0) instructions between the instruction that modifies the PFP and the return instruction. See Example 2.

Example 2:

```
or g1, 7, pfp/* modify return status in PFP */
mov g0, g0/* no-op */
mov g0, g0/* no-op */
mov g0, g0/* no-op */
ret
```

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

43. Erroneous TC Can Be Signaled When Using Multiple DMA Channels

Problem: A terminal count on one DMA channel causes the assertion of TC on another DMA channel under certain specific conditions. This only occurs when one or more channels are programmed as "source synchronized". The problem is due to the lack of synchronization (by the CPU microcode) of the store operation when performing "Source Synchronized" DMA transfers. When an end of buffer condition occurs on a source-synchronized transfer, the DMA microcode asserts an internal signal used by the bus controller to cause the assertion of TC on the next DMA transfer (DACK asserted). The bus controller resets this handshaking bit after the access is completed on the bus. A DREQ for another channel which occurs near the last access of a source-synchronized transfer causes internal microcode process switching between the load and store of the source-synchronized transfer. When this occurs, the bus controller incorrectly fails to reset this internal TC request bit. This causes the next DMA transfer to incorrectly include an active TC.

Workaround: Any of the following four workarounds may be used:

1. When using multiple DMA channels, do not use any source-synchronized transfers. Fly-by and destination-synchronized transfers do not exhibit this behavior.
2. Ensure that when performing source-synchronized transfers that no other DMA activity can occur on other channels.
3. Do not use the TC output signal, use interrupts instead. Program the transfer count to perform all but one transfer using the DMA. Enable interrupt on buffer complete bits. The interrupt handler should complete the access using an address similar to the DMA device, but change one address bit so it can be used as a pseudo TC. For example, if address bit 22 is not decoded in the system, then it can be used as follows:

   0x00001000 Regular access to address 0x1000 (DMA access)

   0x00401000 Last access to address 0x1000 (From Interrupt handler)
4. Use a fixed high-priority DMA dummy channel (requires an unused DMA channel). Gate the TC pin of each source-synchronized DMA channel with an "and" gate to the DREQ pin of the highest priority channel which is programmed to perform a dummy fly-by transfer from memory. This will cause the erroneous TC to occur on an unused channel.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

44. Supervisor Access Pin Operation During System-Supervisor Fault Handling

Problem: A system-supervisor fault may not assert the supervisor access (SUP#) pin while accessing the supervisor stack.

When the processor is in user mode and a fault occurs that invokes a system-supervisor fault handler, the processor switches to the supervisor stack as usual. Then the processor allocates a new frame on the supervisor stack as usual. However, the SUP# pin remains inactive until the first instruction of the fault handler executes.

Implication: The SUP# pin behavior is unreliable when system-supervisor fault handlers are invoked during user mode. Memory controllers that qualify supervisor stack accesses with the SUP# pin can prevent valid accesses under these conditions.

Workaround:

1. Do not design the external memory controller to qualify supervisor memory accesses with the SUP# pin if system-supervisor fault handlers can be invoked during user mode operation.
2. Conversely, if the local memory controller qualifies supervisor memory accesses with the SUP# pin, do not use either (a) user mode or (b) system-supervisor fault handlers.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

45. Power Supply Sequence Can Damage Internal Diodes

Problem: If the voltage on the VCCPLL power supply pin exceeds the V_{CC} pin voltage by 0.5 V at any time, including the power up and power down sequences, excessive currents can permanently damage on-chip electrostatic discharge (ESD) protection diodes. The damage can accumulate over multiple episodes.

Pragmatically, this problem only occurs when the VCCPLL and V_{CC} pins are driven by separate power supplies or voltage regulators. Applications that use one power supply for VCCPLL and VCC are not typically at risk. Verify that your application does not allow the VCCPLL voltage to exceed V_{CC} by 0.5 V.

ESD diodes connect the VCCPLL circuitry to V_{CC}. Normally, those diodes are unbiased or reverse biased, so no current flows. In the event of a positive electrostatic pulse on VCCPLL, the diodes protect the phase-locked loop circuitry by shunting the excess charge to V_{CC}. However, when power supplies forward bias these diodes for any length of time, the current flow can damage or destroy the diodes.

The VCCPLL low-pass filter recommended in the datasheet does not promote this problem.

The VCC5 power supply pin is also susceptible to excessive current damage, but is adequately protected by the 100 Ω series resistor recommended in the datasheet. See the 80960CA/CF datasheets for more details.

Implication: Diode damage can manifest itself as any of the following:

- Resistive short circuits between the VCCPLL and V_{CC} pins.
- Compromised ESD protection on the VCCPLL pin.
- Unspecified functional or parametric failures resulting from damage to the circuitry near the diodes.

Workaround: Use one common power supply or regulator for the VCCPLL and V_{CC} pins. Otherwise, ensure that the V_{CC} pins power up before VCCPLL and power down after VCCPLL. Or, limit the VCCPLL diode current flow by providing at least 100 Ω of resistance in series with the VCCPLL pin.

Status: NoFix. Refer to Summary Table of Changes to determine the affected stepping(s).

Specification Changes

None for this revision of the specification update.

Specification Clarifications

None for this revision of this specification update.

Documentation Changes

1. Chapter 4 - Instruction Set Summary

Issue: Page 4-22, Table 4-4 For the CF, Mode 100 was incorrectly shown as locking 4 Kbytes; it should show 2 Kbytes.

In the third paragraph below the table, the second sentence should be replaced with the following bullets that more accurately describe the load and lock mechanism:

Modes 100 2 and 110 2 select cache load-and-lock options:

- On the CA: mode 100_2 loads and locks the full 1-Kbyte cache, 110_2 loads and locks half the cache.
- On the CF: either mode (100_2 and 110_2) loads and locks half the cache.

This errata also occurs in Table 12-2 on page 12-22.

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

2. Chapter 9 - Instruction Set Reference

Issue: On page 9-78, in the sysctl description, the last paragraph on the page is incorrect and should be removed. It reads:

"When executing a sysctl instruction to load and lock either half or all of the cache, it is necessary to provide a cache load address. The last two bits of the cache load address must be 10base2 for the cache locking mechanism to work properly."

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

3. Chapter 11 - External Bus Description

Issue: In Figure 11-2 (pg 11-8), the WAIT# signal is incorrectly shown as transitioning; it should show that the signal is asserted high throughout.

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

4. Chapter 12 - Interrupt Controller

Issue: Page 12-11 Figure 12-6. Vector Cache Enable bits (ICON.vce) are incorrectly defined.

Bit 0 is defined as debounce; it should be defined as Fetch From External Memory.

Bit 1 is defined as Fast; is should be defined as Fetch From Internal RAM.

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

5. Chapter 12 - Page 12-22, Table 12-2

Issue: For the CF, Mode 100 is incorrectly shown as locking 4 Kbytes; it should show 2 Kbytes. This errata also occurs on page 4-22.

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

6. Chapter 13 - DMA Controller

Issue: Page 13-22, Figure 13-9. DMA Command Register bits 30 (Data Cache Global Disable) and 31 (Data Cache Invalidate) were not defined in Figure 13-9 or in the text that follows the figure. The User's Manual PDF file available on Intel's website now includes these definitions.

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

7. Appendix A - Instruction Execution and Performance Optimization

Issue: Page A-29, Table A-11. Mnemonic "bbe" should be changed to "be."

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

8. Appendix A - Page A-42, Section A.2.6.10

Issue: modpc definition should be changed to say, "requires 25 clocks."

modac definition should be changed to say, "requires 11 clocks."

See also the document change for the Instruction Set Quick Reference (Documentation Item #11).

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

9. Appendix B - Bus Interface Examples

Issue: On page B-5, Fig B-3, the ADS# signal incorrectly shows a deassertion in the 6th cycle and the 3rd deassertion in the 11th cycle. It should show NO deassertion in the 6th cycle and the last deassertion in the 10th cycle. (2nd deassertion should be removed; 3rd deassertion should be shifted left 1 cycle).

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

10. Appendix F - Register and Data Structures

Issue: Appendix F is a compilation of all registers and data structures; therefore, this appendix has the same errata indicated in Figure 13-9 and Figure 12-6.

Affected Docs: *i960® Cx Microprocessor User's Manual*, 270710-003

11. divi Instruction Issue changed in Quick Reference

Issue: Page 9: divi Instruction Issue should be changed to 13, emul Result Latency should be changed to 2,3,5,6.

Affected Docs: *i960® Cx Microprocessor User's Guide Instruction Set Quick Reference*, 272220-002

12. modac Instruction Issue changed in Quick Reference

Issue: Page 11: modac Instruction Issue should be changed to 11; Result Latency should be changed to 11. modpc Instruction Issue should be changed to 25; Result Latency should be changed to 25.

Affected Docs: *i960® Cx Microprocessor User's Guide Instruction Set Quick Reference*, 272220-002