IA-32 Intel® Architecture Software Developer's Manual

Documentation Changes

March 2004

Document Number: 252046-008

Contents

Revision History

Version	Description	Date
-001	Initial Release	November 2002
-002	Added 1-10 Documentation Changes. Removed old Documentation Changes items that already have been incorporated in the published Software Developer's manual	December 2002
-003	Added 9 -17 Documentation Changes Removed Documenation Change #6 - References to bits Gen and Len Deleted Removed Documenation Change #4 - VIF Information Added to CLI Discussion	February 2003
-004	Removed Documentation changes 1-17 Added Documentation changes 1-24	June 2003
-005	Removed Documentation Changes 1-24 Added Documentation Changes 1-15	September 2003
-006	Added Documentation Changes 16- 34	November 2003
-007	Updated Documentation changes 14, 16, 17, and 28. Added Documentation Changes 35-45.	January 2004
-008	Removed Documentation Changes 1-45 Added Documentation Changes 1-5	March 2004

Preface

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

Affected Documents/Related Documents

Document Title	Document Number
IA-32 Intel® Architecture Software Developer's Manual: Volume 1, Basic Architecture	245470-011
IA-32 Intel® Architecture Software Developer's Manual: Volume 2, Instruction Set Reference	245471-011
IA-32 Intel® Architecture Software Developer's Manual: Volume 3, System Programming Guide	245472-011

Nomenclature

Documentation Changes include errors or omissions from the current published specifications. These changes will be incorporated in the next release of the Software Development Maunal.

Summary Table of Changes

The following table indicates documentation changes which apply to the IA-32 Intel Architecture. This table uses the following notations:

Codes Used in Summary Table

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

Summary Table of Documentation Changes

Number	DOCUMENTATION CHANGES
1.	CPUID Sector Size, Cache Line Size Expressions Have Been Updated
2.	Text Added to LDDQU Description
3.	Exception Lists Corrected in ANDNPD/ANDNPS
4.	Description Section of PSADBW Instruction Has Been Corrected
5.	Reference to Wrong Step Corrected
6.	APIC Chapter Updated

Documentation Changes

1. CPUID Sector Size, Cache Line Size Expressions Have Been Updated

IA-32 Intel Architecture Software Developer's Manual, Volume 2A, Chapter 3, CPUID-CPU Identification section, Table 3-13 has been corrected. A number of table cells have been updated to correct inconsistent expressions. An error in 78H has also been corrected. The impacted table cells are reproduced below (not all text in the table has been reproduced).

22H	3rd-level cache: 512 KB, 4-way set associative, 64-byte line size, 2 lines per sector
23H	3rd-level cache: 1 MB, 8-way set associative, 64-byte line size, 2 lines per sector
25H	3rd-level cache: 2 MB, 8-way set associative, 64-byte line size, 2 lines per sector
29H	3rd-level cache: 4 MB, 8-way set associative, 64-byte line size, 2 lines per sector

. . .   .

78H	2nd-level cache: 1 MB, **4-way set associative**, 64-byte line size
79H	2nd-level cache: 128 KB, 8-way set associative, 64-byte line size, 2 lines per sector
7AH	2nd-level cache: 256 KB, 8-way set associative, 64-byte line size, 2 lines per sector
7BH	2nd-level cache: 512 KB, 8-way set associative, 64-byte line size, 2 lines per sector
7CH	2nd-level cache: 1 MB, 8-way set associative, 64-byte line size, 2 lines per sector

2. Text Added to LDDQU Description

IA-32 Intel Architecture Software Developer's Manual, Volume 2B, Chapter 3, LDDQU: Load Unaligned Integer 128 bits section: text has been added. The area impacted is reprinted below. See the change bar for specific lines.

. . .   .

Implementation Notes

- If the source is aligned to a 16-byte boundary, based on the implementation, the 16 bytes may be loaded more than once. For that reason, the usage of LDDQU should be avoided when using uncached or write-combining (WC) memory regions. For uncached or WC memory regions, keep using MOVDQU.
- This instruction is a replacement for MOVDQU (load) in situations where cache line splits significantly affect performance. It should not be used in situations where store-load forwarding is performance critical. If performance of store-load forwarding is critical to the application, use MOVDQA store-load pairs when data is 128-bit aligned or MOVDQU store-load pairs when data is 128-bit unaligned.

- **If the memory address is not aligned on 16-byte boundary, some implementations may load up to 32 bytes and return 16 bytes in the destination. Some processor implementations may issue multiple loads to access the appropriate 16 bytes. Developers of multi-threaded or multi-processor software should be aware that on these processors the loads will be performed in a non-atomic way.**

. . .   .

3. Exception Lists Corrected in ANDNPD/ANDNPS

IA-32 Intel Architecture Software Developer's Manual, Volume 2B, Chapter 3 - ANDNPD and ANDNPS sections: exception lists have been updated. The impacted area for each section is reprinted below.

ANDNPD—Bitwise Logical AND NOT of Packed Double-Precision Floating-Point Values

. . .   .

Protected Mode Exceptions

#GP(0)	For an illegal memory operand effective address in the CS, DS, ES, FS or GS segments.
	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
#SS(0)	For an illegal address in the SS segment.
#PF(fault-code)	For a page fault.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.
	If CPUID feature flag SSE2 is 0.

Real-Address Mode Exceptions

#GP(0)	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
	If any part of the operand lies outside the effective address space from 0 to FFFFH.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.
	If CPUID feature flag SSE2 is 0.

Virtual-8086 Mode Exceptions

Same exceptions as in Real Address Mode

#PF(fault-code)	For a page fault.

… … …

ANDNPS—Bitwise Logical AND NOT of Packed Single-Precision-Floating-Point Value

… … …

Protected Mode Exceptions

#GP(0)	For an illegal memory operand effective address in the CS, DS, ES, FS or GS segments.
	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
#SS(0)	For an illegal address in the SS segment.
#PF(fault-code)	For a page fault.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.
	If CPUID feature flag SSE is 0.

Real-Address Mode Exceptions

#GP(0)	If memory operand is not aligned on a 16-byte boundary, regardless of segment.
	If any part of the operand lies outside the effective address space from 0 to FFFFH.
#NM	If TS in CR0 is set.
#UD	If EM in CR0 is set.
	If OSFXSR in CR4 is 0.
	If CPUID feature flag SSE is 0.

Virtual-8086 Mode Exceptions

Same exceptions as in Real Address Mode

#PF(fault-code)	For a page fault.

4. Description Section of PSADBW Instruction Has Been Corrected

IA-32 Intel Architecture Software Developer's Manual, Volume 2B, Chapter 4, the PSADBW-Compute Sum of Absolute Differences section: an error has been corrected in the Description paragraph. The impacted area is reproduced below (not all text in the section has been reproduced). The location of the change is indicated by the change bar.

Description

Computes the absolute value of the difference of 8 unsigned byte integers from the source operand (**second operand**) and from the destination operand (**first operand**). These 8 differences are then summed to produce an unsigned word integer result that is stored in the destination operand. The source operand can be an MMX technology register or a 64-bit memory location or it can be an XMM register or a 128-bit memory location. The destination operand can be an MMX technology register or an XMM register. Figure 4-5 shows the operation of the PSADBW instruction when using 64-bit operands.

5. Reference to Wrong Step Corrected

IA-32 Intel Architecture Software Developer's Manual, Volume 3, Chapter 7, section 7.5.4.1: contains an incorrect reference in a numbered list. This has been corrected. The impacted step is reproduced below with a change bar to indicate the location of the error (not all steps have been reproduced).

....

15. Broadcasts an INIT-SIPI-SIPI IPI sequence to the APs to wake them up and initialize them:

```
MOV ESI, ICR_LOW; load address of ICR low dword into ESI
MOV EAX, 000C4500H; load ICR encoding for broadcast INIT IPI
                ; to all APs into EAX
MOV [ESI], EAX  ; broadcast INIT IPI to all APs
; 10-millisecond delay loop
MOV EAX, 000C46XXH; load ICR encoding for broadcast SIPI IP
               ; to all APs into EAX, where xx is the
               ; vector computed in step 10.
MOV [ESI], EAX  ; broadcast SIPI IPI to all APs
; 200-microsecond delay loop
MOV [ESI], EAX  ; broadcast second SIPI IPI to all APs
; 200-microsecond delay loop
```

6. APIC Chapter Updated

IA-32 Intel Architecture Software Developer's Manual, Volume 3, Chapter 8 has been updated. Information has been added to multiple sections; this information indicates the model-specific nature of some features. The new information is reprinted below (with enough of surrounding text to indicate the new text's location; not all text in the chapter is reproduced). See the change bars to locate the updated lines.

8.4.6. Local APIC ID

At power up, system hardware assigns a unique APIC ID to each local APIC on the system bus (for Pentium 4 and Intel Xeon processors) or on the APIC bus (for P6 family and Pentium processors). The hardware assigned APIC ID is based on system topology and includes encoding for socket position and cluster information (see Figure 7-2).

In MP systems, the local APIC ID is also used as a processor ID by the BIOS and the operating system. However, the ability of software to modify the APIC ID is processor model specific. Because of this, operating system software should avoid writing to the local APIC ID register.

The processor receives the hardware assigned APIC ID by sampling pins A11# and A12# and pins BR0# through BR3# (for the Pentium 4, Intel Xeon, and P6 family processors) and pins BE0# through BE3# (for the Pentium processor). The APIC ID latched from these pins is stored in the APIC ID field of the local APIC ID register (see Figure 8-6), and is used as the initial APIC ID for the processor. It is also the value returned to the EBX register, when the CPUID instruction is executed with a source operand value of 1 in the EAX register.

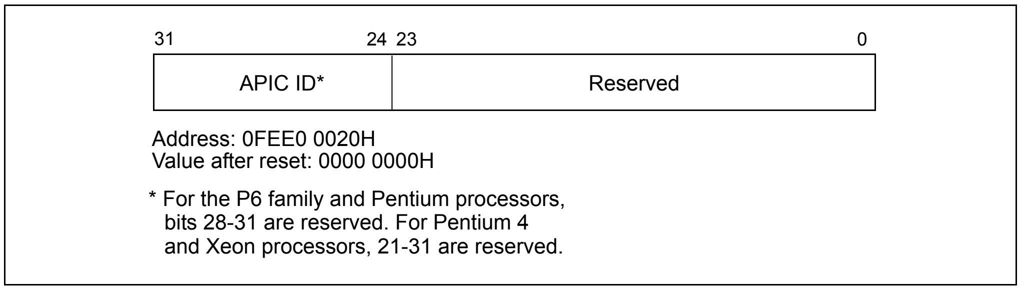
31
24 23
0
APIC ID*
Reserved
Address: 0FEE0 0020H
Value after reset: 0000 0000H
* For the P6 family and Pentium processors, bits 28-31 are reserved. For Pentium 4 and Xeon processors, 21-31 are reserved.

Figure 8-6. Local APIC ID Register

For the P6 family and Pentium processors, the local APIC ID field in the local APIC ID register is 4 bits, and encodings 0H through EH can be used to uniquely identify 15 different processors connected to the APIC bus. For the Pentium 4 and Intel Xeon processors, the xAPIC specification extends the local APIC ID field to 8 bits which can be used to identify up to 255 processors in the system.

Following power up or a hardware reset, software (typically the BIOS software) can modify the APIC ID field in the local APIC ID register for each processor in the system. When changing APIC IDs, software must insure that each APIC ID for each local APIC is unique throughout the system.

... ... omitted text....

8.6. ISSUING INTERPROCESSOR INTERRUPTS

The following sections describe the local APIC facilities that are provided for issuing interprocessor interrupts (IPIs) from software. The primary local APIC facility for issuing IPIs is the interrupt command register (ICR). The ICR can be used for the following functions:

- To send an interrupt to another processor.
- To allow a processor to forward an interrupt that it received but did not service to another processor for servicing.
- To direct the processor to interrupt itself (perform a self interrupt).
- To deliver special IPIs, such as the start-up IPI (SIPI) message, to other processors.

Interrupts generated with this facility are delivered to the other processors in the system through the system bus (for Pentium 4 and Intel Xeon processors) or the APIC bus (for P6 family and Pentium processors). **The ability for a processor to send a lowest priority IPI is model specific and should be avoided by BIOS and operating system software.**

8.6.1. Interrupt Command Register (ICR)

The interrupt command register (ICR) is a 64-bit local APIC register (see Figure 8-12) that allows software running on the processor to specify and send interprocessor interrupts (IPIs) to other IA-32 processors in the system.

To send an IPI, software must set up the ICR to indicate the type of IPI message to be sent and the destination processor or processors. (All fields of the ICR are read-write by software with the exception of the delivery status field, which is read-only.) The act of writing to the low doubleword of the ICR causes the IPI to be sent.

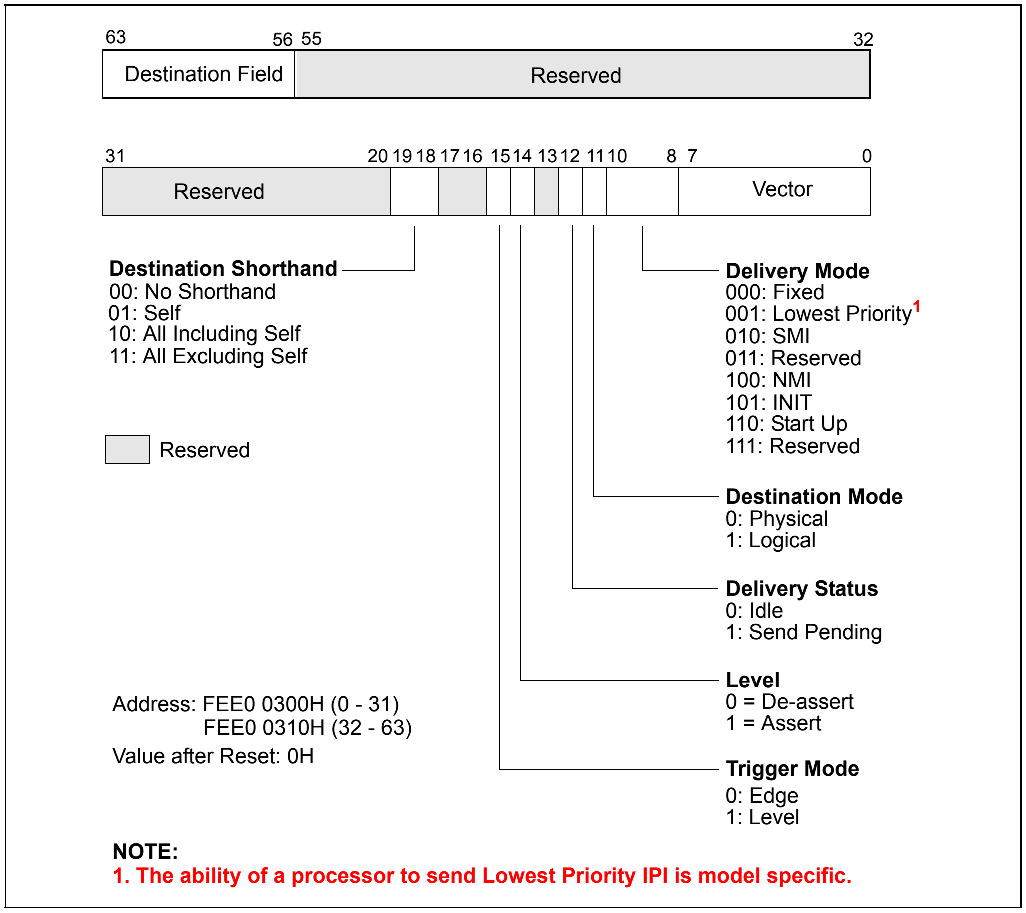

Figure 8-12. Interrupt Command Register (ICR)

The ICR consists of the following fields.

Vector The vector number of the interrupt being sent.

Delivery Mode Specifies the type of IPI to be sent. This field is also know as the IPI message type field.

000 (Fixed) Delivers the interrupt specified in the vector field to the target processor or processors.

001 (Lowest Priority)
Same as fixed mode, except that the interrupt is delivered to the processor executing at the lowest priority among the set of processors specified in the destination field. **The ability for a processor to send a lowest priority IPI is model specific and should be avoided by BIOS and operating system software.**

010 (SMI) Delivers an SMI interrupt to the target processor or processors. The vector field must be programmed to 00H for future compatibility.

011 (Reserved)

100 (NMI) Delivers an NMI interrupt to the target processor or processors. The vector information is ignored.

101 (INIT) Delivers an INIT request to the target processor or processors, which causes them to perform an INIT. As a result of this IPI message, all the target processors perform an INIT. The vector field must be programmed to 00H for future compatibility.

101 (INIT Level De-assert)
(Not supported in the Pentium 4 and Intel Xeon processors.) Sends a synchronization message to all the local APICs in the system to set their arbitration IDs (stored in their Arb ID registers) to the values of their APIC IDs (see Section 8.7., "System and APIC Bus Arbitration"). For this delivery mode, the level flag must be set to 0 and trigger mode flag to 1. This IPI is sent to all processors, regardless of the value in the destination field or the destination shorthand field; however, software should specify the "all including self" shorthand.

110 (Start-Up) Sends a special "start-up" IPI (called a SIPI) to the target processor or processors. The vector typically points to a start-up routine that is part of the BIOS boot-strap code (see Section 7.5., "Multiple-Processor (MP) Initialization"). Note that IPIs sent with this delivery mode are not automatically retried if the source APIC is unable to deliver it. It is up to the software to determine if the SIPI was not successfully delivered and to reissue the SIPI if necessary.

Destination Mode Selects either physical (0) or logical (1) destination mode (see Section 8.6.2., *Determining IPI Destination*).

Delivery Status (Read Only)
Indicates the IPI delivery status, as follows:

0 (Idle) There is currently no IPI activity for this local APIC, or the previous IPI sent from this local APIC was delivered and accepted by the target processor or processors.

1 (Send Pending)
Indicates that the last IPI sent from this local APIC has not yet been accepted by the target processor or processors.

Level For the INIT level de-assert delivery mode this flag must be set to 0; for all other delivery modes it must be set to 1. (This flag has no meaning in Pentium 4 and Intel Xeon processors, and will always be issued as a 1.)

Trigger Mode Selects the trigger mode when using the INIT level de-assert delivery mode: edge (0) or level (1). It is ignored for all other delivery modes. (This flag has no meaning in Pentium 4 and Intel Xeon processors, and will always be issued as a 0.)

Destination Shorthand Indicates whether a shorthand notation is used to specify the destination of the interrupt and, if so, which shorthand is used. Destination shorthands are used in place of the 8-bit destination field, and can be sent by software using a single write to the low doubleword of the ICR. Shorthands are defined for the following cases: software self interrupt, IPIs to all processors in the system including the sender, IPIs to all processors in the system excluding the sender.

00: (No Shorthand)
The destination is specified in the destination field.

01: (Self) The issuing APIC is the one and only destination of the IPI. This destination shorthand allows software to interrupt the processor on which it is executing. An APIC implementation is free to deliver the self-interrupt message internally or to issue the message to the bus and "snoop" it as with any other IPI message.

10: (All Including Self)
The IPI is sent to all processors in the system including the processor sending the IPI. The APIC will broadcast an IPI message with the destination field set to FH for Pentium and P6 family processors and to FFH for Pentium 4 and Intel Xeon processors.

11: (All Excluding Self)
The IPI is sent to all processors in a system with the exception of the processor sending the IPI. The APIC broadcasts a message with the physical destination mode and destination field set to 0xFH for Pentium and P6 family processors and to 0xFFH for Pentium 4 and Intel Xeon processors. **Support for this destination shorthand in conjunction with the lowest-priority delivery mode is model specific. For Pentium 4 and Intel Xeon processors, when this shorthand is used together with lowest priority delivery mode, the IPI may be redirected back to the issuing processor.**

Destination Specifies the target processor or processors. This field is only used when the destination shorthand field is set to 00B. If the destination mode is set to physical, then bits 56 through 59 contain the APIC ID of the target processor for Pentium and P6 family processors and bits 56 through 63 contain the APIC ID of the target processor the for Pentium 4 and Intel Xeon processors. If the destination mode is set to logical, the interpretation of the 8-bit destination field depends on the settings of the DFR and LDR registers of the local APICs in all the processors in the system (see Section 8.6.2., *Determining IPI Destination*).

Note that not all the combinations of options for the ICR are valid. Table 8-2 shows the valid combinations for the fields in the ICR for the Pentium 4 and Intel Xeon processors; Table 8-3 shows the valid combinations for the fields in the ICR for the P6 family processors.

Table 8-2. Valid Combinations for the Pentium 4 and Intel Xeon Processors' Local xAPIC Interrupt Command Register

Destination Shorthand	Valid/ Invalid	Trigger Mode	Delivery Mode	Destination Mode
No Shorthand	Valid	Edge	**All Modes[1]**	Physical or Logical
No Shorthand	Invalid[2]	Level	All Modes	Physical or Logical
Self	Valid	Edge	Fixed	X[3]
Self	Invalid[2]	Level	Fixed	X
Self	Invalid	X	Lowest Priority, NMI, INIT, SMI, Start-Up	X
All Including Self	Valid	Edge	Fixed	X
All Including Self	Invalid[2]	Level	Fixed	X
All Including Self	Invalid	X	Lowest Priority, NMI, INIT, SMI, Start-Up	X
All Excluding Self	Valid	Edge	**Fixed, Lowest Priority[1,4], NMI, INIT, SMI, Start-Up**	X
All Excluding Self	Invalid[2]	Level	FIxed, Lowest Priority[4], NMI, INIT, SMI, Start-Up	X

NOTES:
1. **The ability of a processor to send a lowest priority IPI is model specific**.
2. For these interrupts, if the trigger mode bit is 1 (Level), the local xAPIC will override the bit setting and issue the interrupt as an edge triggered interrupt.
3. X—don't care.
4. When using the "lowest priority" delivery mode and the "all excluding self" destination, the IPI can be redirected back to the issuing APIC, which is essentially the same as the "all including self" destination mode.

Table 8-3. Valid Combinations for the P6 Family Processors Local APIC Interrupt Command Register

Destination Shorthand	Valid/ Invalid	Trigger Mode	Delivery Mode	Destination Mode
No Shorthand	Valid	Edge	**All Modes[1]**	Physical or Logical
No Shorthand	Valid[2]	Level	**Fixed, Lowest Priority[1], NMI**	Physical or Logical
No Shorthand	Valid[3]	Level	INIT	Physical or Logical
Self	Valid	Edge	Fixed	X[4]
Self	1	Level	Fixed	X
Self	Invalid[5]	X	Lowest Priority, NMI, INIT, SMI, Start-Up	X
All including Self	Valid	Edge	Fixed	X
All including Self	Valid[2]	Level	Fixed	X
All including Self	Invalid[5]	X	Lowest Priority, NMI, INIT, SMI, Start-Up	X
All excluding Self	Valid	Edge	**All Modes[1]**	X
All excluding Self	Valid[2]	Level	**Fixed, Lowest Priority[1], NMI**	X
All excluding Self	Invalid[5]	Level	SMI, Start-Up	X
All excluding Self	Valid[3]	Level	INIT	X
X	Invalid[5]	Level	SMI, Start-Up	X

NOTES:
1. **The ability of a processor to send a lowest priority IPI is model specific.**
2. Treated as edge triggered if level bit is set to 1, otherwise ignored.

3. Treated as edge triggered when Level bit is set to 1; treated as "INIT Level Deassert" message when level bit is set to 0 (deassert). Only INIT level deassert messages are allowed to have the level bit set to 0. For all other messages the level bit must be set to 1.
4. X—Don't care.
5. The behavior of the APIC is undefined.

8.6.2. Determining IPI Destination

The destination of an IPI can be one, all, or a subset (group) of the processors on the system bus. The sender of the IPI specifies the destination of an IPI with the following APIC registers and fields within the registers:

- The ICR register—The following fields in the ICR register are used to specify the destination of an IPI:
  - Destination Mode—selects one of two destination modes (physical or logical).
  - Destination field—In physical destination mode, used to specify the APIC ID of the destination processor; in logical destination mode, used to specify a message destination address (MDA) that can be used to select specific processors in clusters.
  - Destination Shorthand—A quick method of specifying all processors, all excluding self, or self as the destination.
  - **Delivery mode, Lowest Priority—Architecturally specifies that a lowest-priority arbitration mechanism be used to select a destination processor from a specified group of processors. The ability of a processor to send a lowest priority IPI is model specific and should be avoided by BIOS and operating system software.**
- Local destination register (LDR)—Used in conjunction with the logical destination mode and MDAs to select the destination processors.
- Destination format register (DFR)—Used in conjunction with the logical destination mode and MDAs to select the destination processors.

How the ICR, LDR, and DFR are used to select an IPI destination depends on the destination mode used: physical, logical, broadcast/self, or lowest-priority delivery mode. These destination modes are described in the following sections.

8.6.2.1. PHYSICAL DESTINATION MODE

In physical destination mode, the destination processor is specified by its local APIC ID (see Section 8.4.6., *Local APIC ID*). For Pentium 4 and Intel Xeon processors, either a single destination (local APIC IDs 00H through FEH) or a broadcast to all APICs (the APIC ID is FFH) may be specified in physical destination mode.

A broadcast IPI (bits 28-31 of the MDA are 1's) or I/O subsystem initiated interrupt with lowest priority delivery mode is not supported in physical destination mode and must not be configured by software. Also, for any non-broadcast IPI or I/O subsystem initiated interrupt with lowest priority delivery mode, software must ensure that APICs defined in the interrupt address are present and enabled to receive interrupts.

For the P6 family and Pentium processors, a single destination is specified in physical destination mode with a local APIC ID of 0H through 0EH, allowing up to 15 local APICs to be addressed on the APIC bus. A broadcast to all local APICs is specified with 0FH.

NOTE

The actual number of local APICs that can be addressed on the system bus may be restricted by hardware.

8.6.2.2. LOGICAL DESTINATION MODE

In logical destination mode, IPI destination is specified using an 8-bit message destination address (MDA), which is entered in the destination field of the ICR. Upon receiving an IPI message that was sent using logical destination mode, a local APIC compares the MDA in the message with the values in its LDR and DFR to determine if it should accept and handle the IPI. **For both configurations of logical destination mode, when combined with lowest priority delivery mode, software is responsible for ensuring that all of the local APICs included in or addressed by the IPI or I/O subsystem interrupt are present and enabled to receive the interrupt.**

Figure 13 shows the layout of the logical destination register (LDR). The 8-bit logical APIC ID field in this register is used to create an identifier that can be compared with the MDA.

NOTE

The logical APIC ID should not be confused with the local APIC ID that is contained in the local APIC ID register.

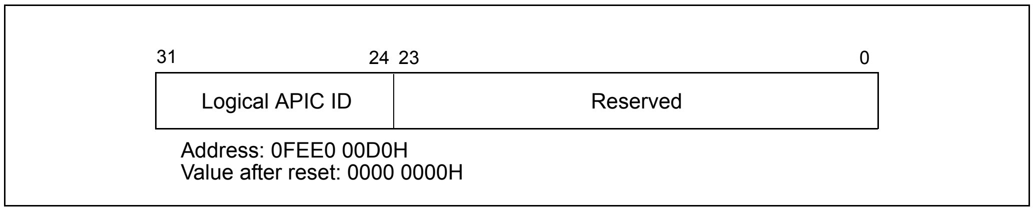

Figure 8-13. Logical Destination Register (LDR)

Figure 14 shows the layout of the destination format register (DFR). The 4-bit model field in this register selects one of two models (flat or cluster) that can be used to interpret the MDA when using logical destination mode.

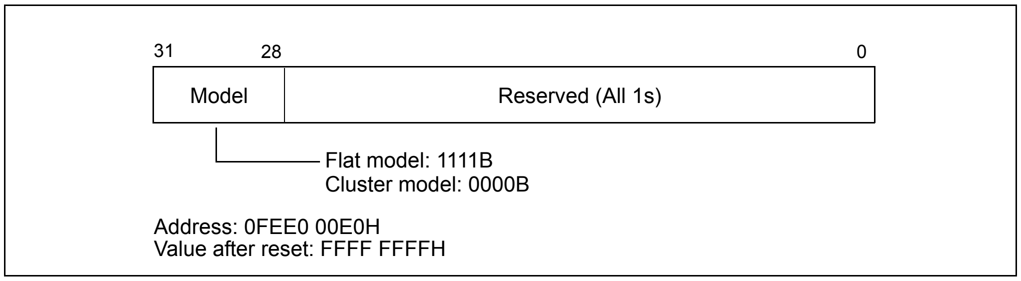

Figure 8-14. Destination Format Register (DFR)

The interpretation of MDA for the two models is described in the following paragraphs.

Flat Model. This model is selected by programming DFR bits 28 through 31 to 1111. Here, a unique logical APIC ID can be established for up to 8 local APICs by setting a different bit in the logical APIC ID field of the LDR for each local APIC. An group of local APICs can then be selected by setting one or more bits in the MDA.

Each local APIC performs a bit-wise AND of the MDA and its logical APIC ID. If a true condition is detected, the local APIC accepts the IPI message. A broadcast to all APICs is achieved by setting the MDA to all 1s.

Cluster Model. This model is selected by programming DFR bits 28 through 31 to 0000. This model supports two basic destination schemes: flat cluster and hierarchical cluster.

The flat cluster destination model is only supported for P6 family and Pentium processors. Using this model, all APICs are assumed to be connected through the APIC bus. Bits 28 through 31 of the MDA contains the encoded address of the destination cluster, and bits 24 through 27 identify up to four local APICs within the cluster (each bit is assigned to one local APIC in the cluster, as in the flat connection model). To identify one or more local APICs, bits 28 through 31 of the MDA are compared with bits 28 through 31 of the LDR to determine if a local APIC is part of the cluster. Bits 24 through 27 of the MDA are compared with Bits 24 through 27 of the LDR to identify a local APICs within the cluster.

Sets of processors within a cluster can be specified by writing the target cluster address in bits 28 through 31 of the MDA and setting selected bits in bits 24 through 27 of the MDA, corresponding to the chosen members of the cluster. In this mode, 15 clusters (with cluster addresses of 0 through 14) each having 4 local APICs can be specified in the message. For the P6 and Pentium processor's local APICs, however, the APIC arbitration ID supports only 15 APIC agents, and hence the total number of processors and their local APICs supported in this mode is limited to 15. Broadcast to all local APICs is achieved by setting all destination bits to one. This guarantees a match on all clusters, and selects all APICs in each cluster. **A broadcast IPI or I/O subsystem broadcast interrupt with lowest priority delivery mode is not supported in cluster mode and must not be configured by software.**

The hierarchical cluster destination model can be used with Pentium 4, Intel Xeon, P6 family, or Pentium processors. With this model, a hierarchical network can be created by connecting different flat clusters via independent system or APIC buses. This scheme requires a cluster manager within each cluster, which is responsible for handling message passing between system or APIC buses. One cluster contains up to 4 agents. Thus 15 cluster managers, each with 4 agents, can form a network of up to 60 APIC agents. Note that hierarchical APIC networks requires a special cluster manager device, which is not part of the local or the I/O APIC units.

... ... omitted text....

8.11.1. Message Address Register Format

The format of the Message Address Register (lower 32-bits) is shown in Figure .

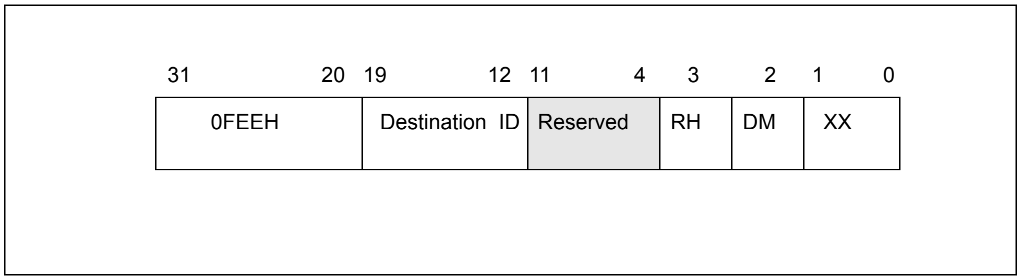

Figure 8-23. Layout of the MSI Message Address Register

Fields in the Message Address Register are as follows:

1. Bits 31-20: These bits contain a fixed value for interrupt messages (0FEEH). This value locates interrupts at the 1MB area with a base address of 4G – 18M. All accesses to this region are directed as interrupt messages. Care must to be taken to ensure that no other device claims the region as I/O space.
2. Destination ID: This field contains an 8-bit destination ID. It identifies the message's target processor(s). The destination ID corresponds to bits 63:56 of the I/O APIC Redirection Table Entry if the IOAPIC is used to dispatch the interrupt to the processor(s).
3. Redirection Hint Indication (RH): This bit indicates whether the message should be directed to the processor with the lowest interrupt priority among processors that can receive the interrupt.
    - **When RH is 0, the interrupt is directed to the processor listed in the Destination ID field.**
    - **When RH is 1 and the physical destination mode is used, the Destination ID field must not be set to 0xFF; it must point to a processor that is present and enabled to receive the interrupt.**
    - **When RH is 1 and the logical destination mode is active in a system using a flat addressing model, the Destination ID field must be set so that bits set to 1 identify processors that are present and enabled to receive the interrupt.**
    - **If RH is set to 1 and the logical destination mode is active in a system using cluster addressing model, then Destination ID field must not be set to 0xFF; the processors identified with this field must be present and enabled to receive the interrupt.**
4. Destination Mode (DM): This bit indicates whether the Destination ID field should be interpreted as logical or physical APIC ID for delivery of the lowest priority interrupt. If RH is 1 and DM is 0, the Destination ID field is in physical destination mode and only the processor in the system that has the matching APIC ID is considered for delivery of that interrupt (this means no re-direction). If RH is 1 and DM is 1, the Destination ID Field is interpreted as in logical destination mode and the redirection is limited to only those processors that are part of the logical group of processors based on the processor's logical APIC ID and the Destination ID field in the message. The logical group of processors consists of those identified by matching the 8-bit Destination ID with the logical destination identified by the Destination Format Register and the Logical Destination Register in each local APIC. The details are similar to those described in Section 8.6.2., *Determining IPI Destination*. If RH is 0, then the DM bit is ignored and the message is sent ahead independent of whether the physical or logical destination mode is used.