

Sable CPU Module Specification

This document describes the functional and physical characteristics of the Sable System CPU module. The Sable System is an implementation of the Digital Alpha AXP Architecture and uses the (DECchip 21064 and DECchip 21064-A275) processor chips.

Revision/Update Information: Revision 1.1—5-August-1994

Alpha and VAX Servers Group (AVS)
Digital Equipment Corporation, Maynard, Massachusetts



Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software, if any, described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Copyright © 1990–1994 by Digital Equipment Corporation.

All Rights Reserved.
Printed in U.S.A.

The following are trademarks of Digital Equipment Corporation: AlphaServer, DEC LANcontroller, OpenVMS, StorageWorks, VAX, and the DIGITAL logo.

MEMORY CHANNEL is a trademark of Encore Computer Corporation.

FCC NOTICE: The equipment described in this manual generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference, in which case the user at his own expense may be required to take measures to correct the interference.

CONTENTS

Preface	v
Chapter 1 CPU MODULE COMPONENTS AND FEATURES	1
1.1 The 21064 CPU Chip	4
1.2 The Alpha AXP Architecture	5
Chapter 2 CPU MODULE VARIATIONS	7
2.1 Form Factors	7
Chapter 3 FUNCTIONS LOCATED ON THE DECCHIP 21064	9
3.1 21064 Chip Features	9
3.1.1 I-box Internal Processor Registers	11
3.1.2 Branch Prediction Logic	11
3.1.3 Instruction Translation Buffers (ITBs)	12
3.1.4 Interrupt Logic	13
3.1.5 Performance Counters	14
3.1.6 Translation Buffer Tag Register (TB_TAG)	16
3.1.7 Instruction Translation Buffer Page Table Entry Register (ITB_PTE)	18
3.1.8 Instruction Cache Control and Status Register (ICCSR)	19
3.1.9 Instruction Translation BUffer Page Table Entry Temporary Register(ITB_PTE_TEMP)	21
3.1.10 Exception Address Register (EXC_ADDR)	21
3.1.11 Serial Line Clear Register (SL_CLR)	23
3.1.12 Serial Line Receive Register (SL_RCV)	23
3.1.13 Instruction Translation Buffer Zap Register (ITBZAP)	24
3.1.14 Instruction Translation Buffer ASM Register (ITBASM)	24
3.1.15 Instruction Translation Buffer IS Register (ITBIS)	24
3.1.16 Processor Status (PS)	25
3.1.17 Exception Summary Register (EXC_SUM)	26
3.1.18 Privileged Architecture Library Base Register (PAL_BASE)	28
3.1.19 Hardware Interrupt Request Register (HIRR)	29
3.1.20 Software Interrupt Request Register (SIRR)	31
3.1.21 Asynchronous Trap Request Register (ASTRR)	32
3.1.22 Hardware Interrupt Enable Register (HIER)	33
3.1.23 Software Interrupt Enable Register (SIER)	34
3.1.24 Asynchronous System Trap Enable Register (ASTER)	35
3.1.25 Serial Line Transmit (SL_XMIT)	36
3.2 Ebox	37
3.3 Abox	37

3.3.1 Abox IPRs	37
3.3.2 Translation Buffer Control Register (TB_CTL)	38
3.3.3 Data Translation Buffer Page Table Entry Register (DTB_PTE)	39
3.3.4 Data Translation Buffer Page Table Entry Temporary Register (DTB_PTE_TEMP)	40
3.3.5 Memory Management Control and Status Register (MM_CSR)	41
3.3.6 Virtual Address Register (VA)	41
3.3.7 Data Translation Buffer Zap Register (DTBZAP)	41
3.3.8 Data Translation Buffer ASM Register (DTBASM)	42
3.3.9 Data Translation Buffer Invalidate Signal Register (DTBIS)	42
3.3.10 Flush Instruction Cache Register (FLUSH_IC)	42
3.3.11 Flush Instruction Cache ASM Register (FLUSH_IC_ASM)	42
3.3.12 A-box Control Register (ABOX_CTL)	42
3.3.13 Alternative Processor Mode Register (ALT_MODE)	46
3.3.14 Cycle Counter Register (CC)	46
3.3.15 Cycle Counter Control Register (CC_CTL)	46
3.3.16 Bus Interface Unit Control Register (BIU_CTL)	46
3.3.17 Data Translation Buffer (DTB)	53
3.3.18 Bus Interface Unit (BIU)	54
3.3.19 Load Silos	54
3.3.20 Write Buffer	55
3.4 Fbox	56
3.5 21064 IEEE Floating Point Conformance	57
3.6 Cache Organization	59
3.6.1 Data Cache (Dcache)	59
3.6.2 Instruction Cache (Icache)	59
3.7 Pipeline Organization	59
3.7.1 Static and Dynamic Stages	61
3.7.2 Aborts	61
3.7.3 Non-Issue Conditions	63
3.8 Scheduling and Issuing Rules	63
3.8.1 Instruction Class Definition	63
3.8.2 Producer-Consumer Latency	65
3.8.3 Producer-Producer Latency	67
3.8.4 Instruction Issue Rules	67
3.8.5 Dual Issue Table	68
3.9 PALcode	69
3.9.1 Required PALcode Instructions	70
3.9.2 Architecturally Reserved PALcode Instructions	70
3.9.3 PAL_TEMP's	70
3.9.4 Data Cache Status Register (C_STAT)	70
3.9.5 DECchip 21064-A275 Data Cache Status Register (C_STAT)	71

3.9.6 Data Cache Address Register (DC_ADDR)	73
3.9.7 Bus Interface Unit Status Register (BIU_STAT)	74
3.9.8 Bus Interface Unit Address Register (BIU_ADDR)	75
3.9.9 Fill Address Register (FILL_ADDR)	77
3.9.10 Fill Syndrome Register (FILL_SYNDROME)	78
3.9.11 Backup Cache Tag Register (BC_TAG)	79
3.9.12 EDC Error Correction	81

Chapter 4 FUNCTIONS LOCATED ELSEWHERE ON THE CPU

MODULE	83
4.1 Back-up Cache (B-Cache)	83
4.1.1 Control Store	83
4.1.2 Tag Store	84
4.1.3 Data Store	84
4.1.4 B-Cache Control Register Definitions	85
4.1.4.1 CSR Space	85
4.1.4.2 B-Cache Control Register - CSR0	86
4.1.4.3 B-Cache Correctable Error Register - CSR1	92
4.1.4.4 B-Cache Correctable Error Address Register - CSR2	95
4.1.4.5 B-Cache Uncorrectable Error Register - CSR3	97
4.1.4.6 B-Cache Uncorrectable Error Address Register - CSR4	100
4.1.5 Back-up Cache Cycle Time	101
4.1.5.1 The 21064 Cycles	102
4.1.5.2 System-bus Cycles	102
4.2 Cache Block Merge Buffer	102
4.3 Duplicate Primary Data Cache Tag Store	103
4.3.1 Duplicate Tag Error Register - CSR5	105
4.4 Lack of Duplicate Primary Instruction Cache Tag Store	106
4.5 Lack of Cache Block Prefetch	107
4.6 Data Integrity	107
4.7 System-bus Interface	107
4.7.1 System-bus Control Register - CSR6	108
4.7.2 System-bus Error Register - CSR7	111
4.7.3 System-bus Error Address Low Register - CSR8	117
4.7.4 System-bus Error Address High Register - CSR9	118
4.8 Multiprocessor Configuration CSR Definitions	121
4.8.1 Processor Mailbox Register - CSR10	121
4.8.2 Interprocessor Interrupt Request Register - CSR11	122
4.9 System Interrupt Clear Register - CSR12	123
4.10 Address Lock Register - CSR13	125

4.11 Miss Address Register - CSR14	127
4.11.1 C4 Revision Register - CSR15	128
4.12 Interval Timer	130
4.13 D-bus	131
4.14 System-bus Arbiter	133
4.15 System-bus CRESET L Generation	136
4.15.1 Sable CPU Non-Volatile EEPROM	137
Chapter 5 CPU MODULE TRANSACTIONS	143
5.1 Processor Transactions	143
5.1.1 21064 Processor TRANSACTIONS	146
5.1.1.1 FAST EXTERNAL CACHE READ HIT	146
5.1.1.2 FAST EXTERNAL CACHE WRITE HIT	147
5.1.1.3 READ_BLOCK TRANSACTION	148
5.1.1.4 WRITE_BLOCK	150
5.1.1.5 LDxL TRANSACTION	152
5.1.1.6 StxC TRANSACTION	152
5.1.1.7 BARRIER TRANSACTION	153
5.1.1.8 FETCH TRANSACTION	153
5.1.1.9 FETCHM TRANSACTION	154
5.1.2 Cacheable vs Non-Cacheable vs Allocate-Invalid	155
5.2 System-bus Transactions	156
5.2.1 CPU as Commander	157
5.2.2 CPU as Bystander	157
5.2.3 CPU as Responder	157
5.3 Control Flow of CPU Module Transactions	158
5.3.1 Processor Initiated	158
5.3.2 System-bus Initiated	162
Chapter 6 CACHE INVALIDATE MANAGEMENT	165
6.1 Processor Caused Invalidates	165
6.2 C-bus Caused Invalidates	165
Chapter 7 EXCEPTIONS AND INTERRUPTS	167
7.1 Processor Generated	167

7.1.1 Exception Handling	168
7.1.1.1 PAL Priority Level	169
7.1.1.2 PALcode 0020 Entry Characteristics	169
7.1.1.3 PAL Routine Behavior	169
7.1.1.3.1 B-Cache Tag Parity Error	169
7.1.1.3.2 B-Cache Tag Control Parity Error	170
7.1.1.3.3 B-Cache Data Single Bit EDC Error	170
7.1.1.3.4 B-Cache Data Uncorrectable EDC Error	170
7.1.1.3.5 B-Cache Data Single Bit EDC Error	171
7.1.1.3.6 B-Cache Data Uncorrectable EDC Error	171
7.1.1.3.7 21064 Data bus Single Bit EDC Error	171
7.1.1.3.8 21064 Data bus Uncorrectable EDC Error	171
7.1.1.3.9 B-Cache Tag or Tag Control Parity Error	172
7.1.1.3.10 Cobra-bus Parity Error	172
7.1.1.3.11 Invalid Cobra-bus Address	172
7.1.1.3.12 Other CPU Errors	173
7.1.1.3.13 Main Memory Uncorrectable EDC Errors	173
7.2 Non-processor Generated	173
7.2.1 Interrupt Handling	173
7.2.1.1 PAL Priority Level	174
7.2.1.2 PALcode 00E0 Entry Characteristics	174
7.2.1.3 Hardware 0 - Hardware Error	175
7.2.1.4 Hardware 1 - Local I/O	177
7.2.1.5 Hardware 3 - Interprocessor	177
7.2.1.6 Hardware 4 - Interval Timer	177
7.2.1.7 Hardware 5 - System Events	177
7.2.1.8 Software X	178
7.2.1.9 Serial Line	178
7.2.1.10 Performance Counter X	178
7.2.1.11 Asynchronous System Trap	178
Chapter 8 FAULT MANAGEMENT/ERROR RECOVERY	179
8.1 Processor Errors	179
8.2 B-Cache Errors	179
8.2.1 Tag and Tag Control Store Parity Errors	179
8.2.2 Data Store EDC Errors	181
8.2.2.1 Correctable	181
8.2.2.2 Uncorrectable	183
8.3 Duplicate P-Cache Tag Store Parity Errors	186
8.4 System-bus Errors	186
8.4.1 C/A Parity Error	186
8.4.2 Data Parity Error	187

8.4.3 Invalid Address - Bus Time-out	187
8.5 I/O Subsystem Errors	188
8.6 C_ERR L Assertion	188
Chapter 9 CPU POWERUP AND INITIALIZATION	189
9.1 Processor Initialization	189
9.1.1 Internal Processor Registers	189
9.1.2 Internal JSR stack	189
9.2 B-Cache Initialization	189
9.2.1 LDQ Data Format - BCC ENABLE B-CACHE INIT Set	190
9.3 Duplicate Tag Store Initialization	191
9.4 System-bus Interface Initialization	191
9.5 CPU clocks and reset	191
9.6 Power-up Sequence	191
9.7 Powering Up with Bad Main Memory	192
Chapter 10 OVERVIEW OF THE CPU TESTABILITY FEATURES	193
10.1 Bus Verification	193
10.1.1 Address	193
10.1.2 Data	193
10.2 C4 EDC Generators	193
10.3 C4 EDC Checkers	194
10.4 DECchip 21064 EDC Checkers	194
10.5 DECchip 21064 EDC Generators	194
10.6 C4 Cobra-bus Probe Predicted Tag Parity Generator	194
10.7 B-Cache Data Store Verification	194
10.8 B-Cache Tag/Control Store Verification	194
10.9 Performance Counters	195
10.10 Brain Dead Module Errors	195
Chapter 11 PHYSICAL AND ELECTRICAL CHARACTERISTICS	197
11.1 CPU Module Physical Specification	197
11.2 SABLE CPU CONNECTOR PINNING	197
11.3 CPU Module Max DC Power Requirement	203

11.4 Environmental Specifications - Class B modified	203
11.4.1 Temperature	203
11.4.1.1 Storage	203
11.4.1.2 Operating	203
11.4.2 Relative Humidity	203
11.4.2.1 Storage	204
11.4.2.2 Operating	204
11.4.3 Altitude	204
11.4.3.1 Storage	204
11.4.3.2 Operating Altitude	204
11.4.4 Airflow	204
11.4.5 Contamination	204
11.4.6 Mean Time Between Failure (MTBF) Rate	204
11.4.7 Electrical Characteristics	204
11.4.7.1 AC References	204
11.4.8 Clock Description	204
11.4.9 AC and DC Characteristics	208
 Appendix A ALPHA ARCHITECTURE OPTIONS SUPPORTED	 211
 Appendix B SABLE CPU MODULE REGISTER REFERENCE GUIDE	 213
 Appendix C COBRA SPECIFIC (PRIVILEGED ARCHITECTURE LIBRARY CODE) PALCODE	 223
C.1 Introduction	223
C.2 PAL Environment	223
C.3 Special PAL Instructions	224
C.3.1 HW_MFPR and HW_MTPR	225
C.3.2 HW_LD and HW_ST	227
C.3.3 HW_REI	228
C.4 PAL Entry Points	228
C.5 General PALmode Restrictions	231
C.5.1 21064 PAL Restrictions	231
C.5.2 21064 Specific PALmode Restrictions	234
C.6 Powerup	235
C.7 TB Miss Flows	237
C.7.1 ITB Miss	237
C.7.2 DTB Miss	238
C.8 Error Flows	239

C.8.1 21064 Error Flows	239
C.8.1.1 EDC Error Handling	239
C.8.1.1.1 Single Bit I-stream EDC error	239
C.8.1.1.2 Single Bit D-stream EDC error	240
C.8.1.1.3 Double Bit I-stream EDC error	240
C.8.1.1.4 Double Bit D-stream EDC error	240
C.8.1.2 BIU: tag address parity error	241
C.8.1.3 BIU: tag control parity error	241
C.8.1.4 BIU: system transaction terminated with CACK_HERR	241
C.8.2 Response to Multiple Errors	241

Appendix D REVISION HISTORY	243
--	------------

INDEX

EXAMPLES

1 Update vs Invalidate Algorithm	104
--	-----

FIGURES

1 Sable CPU Module Block Diagram	2
2 C4 Chip Block Diagram	3
3 21064 CPU Chip Block Diagram	4
4 CPU Form Factors	8
5 TB_TAG_FIG	17
6 ITB_PTE	18
7 ICCSR	19
8 ITB_PTE_TEMP	21
9 EXC_ADDR	22
10 SL_CLR	23
11 SL_RCV	24
12 PS	25
13 EXC_SUM	26
14 PAL_BASE	28
15 HIRR	29
16 SIRR	31
17 ASTRR	32
18 HIER	33
19 SIER	34
20 ASTER	35
21 SL_XMIT	36
22 TB_CTL_FIG	38
23 DTB_PTE	39
24 DTB_PTE_TEMP	40

25	MM_CSR	41
26	DECchip 21064 ABOX_CTL	43
27	DECchip 21064-A275 ABOX_CTL	44
28	ALT_MODE	46
29	BIU Control Register (BIU_CTL)	47
30	DECchip 21064-A275 BIU Control Register (EV45_BIU_CTL)	50
31	Integer Operate Pipeline	60
32	Memory Reference Pipeline	60
33	Floating Point Operate Pipeline	61
34	Producer-Consumer Latency Matrix	66
35	DECchip 21064 C_STAT	71
36	DECchip 21064-A275 C_STAT	71
37	BIU_STAT	74
38	Fill Syndrome	78
39	BC_TAG	80
40	Back-up Cache Entry	83
41	B-Cache Control Register (BCC)	87
42	B-Cache Correctable Error Register (BCCE)	92
43	B-Cache Correctable Error Address Register (BCCEA)	95
44	B-Cache Uncorrectable Error Register (BCUE)	97
45	B-Cache Uncorrectable Error Address Register (BCUEA)	100
46	Duplicate Tag Error Register (DTER)	105
47	System-bus Control Register (CBCTL)	108
48	System-bus Error Register (CBE)	112
49	System-bus Error Address Low Register (CBEAL)	117
50	System-bus Error Address High Register (CBEAH)	118
51	Processor Mailbox Register (PMBX)	121
52	Interprocessor Interrupt Request Register (IPIR)	122
53	System Interrupt Clear Register (SIC)	123
54	Address Lock Register (ADLK)	125
55	Miss Address Register Low (MADRL)	127
56	C4 Revision Register (CRR)	128
57	The 21064 Serial Load Data Format	131
58	Granting Order	135
59	FAST EXTERNAL CACHE READ HIT	147
60	FAST EXTERNAL CACHE WRITE HIT	148
61	READ BLOCK TRANSACTION	149
62	WRITE BLOCK	151
63	BARRIER TRANSACTION	153
64	FETCH TRANSACTION	154
65	Address Space Map	156
66	LDQ Data Format (LDQ_DF)	190
67	System backplane clocks	206
68	module clocks	207
69	Module Layup (reference)	210

70	B-Cache Control Register (BCC-CSR0, offset = 0000)	214
71	B-Cache Correctable Error (BCCE-CSR1, offset = 0020)	215
72	B-Cache Correctable Error Address (BCCEA-CSR2, offset = 0040)	215
73	B-Cache Uncorrectable Error (BCUE-CSR3, offset = 0060)	216
74	B-Cache Uncorrectable Error Address (BCUEA-CSR4, offset = 0080)	216
75	Duplicate Tag Error Register (DTER-CSR5, offset = 00A0)	217
76	Cobra-bus Control Register (CBCTL-CSR6, offset = 00C0)	218
77	Cobra-bus Error Register (CBE-CSR7, offset = 00E0)	219
78	Cobra-bus Error Address Low (CBEAL-CSR8, offset = 0100)	220
79	Cobra-bus Error Address High (CBEAH-CSR9, offset = 0120)	220
80	Processor Mailbox (PMBX-CSR10, offset = 0140)	220
81	Interprocessor Interrupt Request (IPIR-CSR11, offset = 0160)	221
82	System Interrupt Clear Register (SIC-CSR12, offset = 0180)	221
83	Address Lock Register (ADLK-CSR13, offset = 01A0)	221
84	Miss Address Register Low (MADRL-CSR14, offset = 01C0)	222
85	C4 Revision Register (CRR-CSR15, offset = 01E0)	222

TABLES

1	CPU Modules Variations	7
2	Latencies for Single and Double Precision Divide Instructions	10
3	Performance Counter 0 Input Selection	15
4	Performance Counter 1 Input Selection	16
5	ICCSR	19
6	BHE,BPE Branch Prediction Selection	20
7	SL_CLR	23
8	EXC_SUM	27
9	HIRR	29
10	MM_CSR	41
11	Abox Control Register	45
12	ALT Mode	46
13	BIU Control Register Description	47
14	DECchip 21064-A275 BIU Control Register Description	50
15	BC_SIZE	53
16	BC_PA_DIS	53
17	BIU_CTL Initialization Values	53
18	Producer-Consumer Classes	64
19	Opcode Summary (with Instruction Issue Bus)	69
20	Required PALcode Instructions	70
21	Instructions Specific to the 21064	70
22	DECchip 21064 Cache Status Register	71
23	DECchip 21064-A275 Cache Status Register	72
24	BIU STAT	75
25	Syndromes for Single-Bit Errors	78
26	84

27	Base Addresses for CSRs	85
28	B-Cache Control Register Description	88
29	B-Cache Correctable Error Register Description	92
30	B-Cache Correctable Error Address Register Description	95
31	B-Cache Uncorrectable Error Register Description	98
32	B-Cache Uncorrectable Error Address Register Description	100
33	The 21064 B-Cache cycle times	102
34	System-bus B-Cache Access Time	102
35	Duplicate Tag Error Register Description	105
36	Data Integrity Reference	107
37	System-bus Control Register Description	109
38	System-bus Error Register Description	113
39	System-bus Error Address Low Register Description	117
40	System-bus Error Address High Register Description	118
41	Processor Mailbox Register Description	121
42	Interprocessor Interrupt Request Register Description	122
43	System Interrupt Clear Register Description	123
44	Address Lock Register Description	125
45	Miss Address Register Low Description	127
46	C4 Revision Register Description	128
47	CSR Space	130
48	Interval Timer Interrupt Generation	130
49	D-bus Micro Port Mapping	132
50	D-bus Microcontroller Clock Frequency	133
51	D-bus Microcontroller System Control Bus Address	133
52	Sable Arbitration Latency	136
53	CPU I ² C Bus EEPROM Field Definitions	137
54	BIU_CTL Field Description	138
56	BIU_CTL Initialization Values	140
57	CPU EEPROM Defaults	141
58	141
59	Cycle Request	144
60	Cycle Acknowledgment Types	145
61	Read Data Acknowledgment Types	145
62	Processor Initiated Transactions	146
63	System-bus Initiated Transactions	157
64	Processor Initiated Transactions - Control Flow	158
65	System-bus Initiated Transactions - Control Flow	162
66	Invalidate Management - C-bus Caused	165
67	General Exception Isolation Matrix	167
68	Machine Check Isolation Matrix	168
69	Exception Priority/PAL Offset/SCB Offset/IPL	169
70	Hardware Interrupt Configuration	173
71	Interrupt Priority/SCB Offset/IPL	174
72	Tag/Tag Control Error Severity Matrix	179

73	B-Cache Tag Control or Tag Store Errors	181
74	B-Cache Data Correctable Errors	183
75	Uncorrectable Data Store Error Severity Matrix	184
76	B-Cache Uncorrectable Errors	185
77	C/A Parity Errors	187
78	Data Parity Errors	187
79	LDQ Data Format Description	191
80	Kernel Survivability Matrix	195
81	MCA240 J5	197
82	MCA44 J1	200
83	MCA44 J3	201
84	CPU Power Requirements	203
85	System bus AC and DC Characteristics	208
86	HW_MFPR and HW_MTPR Format Description	225
87	IPR Access	225
88	HW_LD and HW_ST Format Description	227
89	The HW_REI Format Description	228
90	PAL Entry Points	229
91	D-stream Error PAL Entry Points	231
92	IPR Reset State	235
93	Revision History	243

Preface

Scope and Organization of this Specification

This specification describes the operation of the Sable CPU. It contains an overview of the system, the CPU module interfaces to the memory and I/O modules and descriptions of transactions, addressing, signals, interface registers, and electrical specifications. In addition, the specification contains discussions of error handling, interrupts, and initialization as well as a description of the **PALcode** required to provide ALPHA SRM conformance.

Related Documents

The following documents are related to or were used in the preparation of this document:

- 21064 Processor Specification
- Alpha System Reference Manual Version 5.0
- Cobra System Bus Specification
- Sable System Spec
- Sable I/O module Specification
- Sable Motherboard Specification
- Sable Memory Specification
- Sable PFMS

Terminology and Conventions

Ranges and Extents

Ranges are specified by a pair of numbers separated by a “..” and are inclusive, for example, a range of integers 0..4 includes the integers 0,1,2,3,and 4.

Extents are specified by a pair of numbers in angle brackets separated by a colon and are inclusive, for example, bits <7:3> specify an extent of bits including bits 7, 6, 5, 4, and 3.

Conventions

Physical addresses are expressed as nine hexadecimal digits punctuated with periods every four digits for readability. (for example, 2.0800.0040(hex))

All numbers are decimal unless otherwise indicated. Where there is ambiguity, numbers other than decimal are indicated with the name of the base following the number in parentheses, for example, FF (hex).

Register bits that must both be set are identified as the logical **and** of these bits (for example, <49&3>). Bits where either bit could be set are identified as the logical **or** of these bits (for example, <49 | 3>. A range of bits that is either all set or any set is specified with "&:" or "|:".

Shaded boxes found in register definitions indicate bit locations that are read and should be written as zero's.

Terminology

B-Cache	The Back-up Cache memory subsystem including its control machines.
BIU	Bus Interface Unit.
BYSTANDER	A Cobra-bus node which is not addressed by a current Cobra-bus commander transaction address.
CACHE LINE	A 32 byte contiguous segment of cache memory, which starts at an aligned hexaword address.
CACHE BLOCK	Synonymous with CACHE LINE.
C4	The gate array(s) that contain the Cobra-bus interface, and B-Cache control machines on the CPU module. Each CPU module will have two C4 chips which are quadword slices of the Cobra-bus interface.
Cobra-bus	The Cobra System bus, which implements a snooping protocol, refer to the Cobra System Bus Specification for further details.
CLEAN	With reference to a cache block in the cache of a Cobra-bus node, the cache block is valid but has not been written so it is exactly the same as the contents in the system memory.
COMMAND	A field of the Cobra-bus address and command cycle (cycle 1) which encodes the transaction type.
COMMANDER	A Cobra-bus node which participates in arbitration and initiates transactions.
CONDITIONAL INVALIDATION	Invalidation of a cached location based upon a set of conditions which are the state of other caches, or the source of the information causing the invalidate.
CYCLE	One clock interval.
D-bus	The bus over which the serial ROM data flows after power-up or system restart.
DIRTY	With reference to a cache block in the cache of a Cobra-bus node, the cache block is valid and has been written so it differs from the copy in system memory.
E-bus	The 21064 Processor Pin bus.
EV4	Pseudonym for the DECchip 21064 processor chip sometimes used in this Document
EV45	Pseudonym for the DECchip 21064-A275 processor chip sometimes used in this document.
FEPR0M	The Flash Erasable Programmable Read Only Memory subsystem is found on the CPU module on the A-bus.
HIR	Hardware Interrupt Request, a bit field located in the Hardware Interrupt Request Register of the 21064 processor.
HIT	Indicates that a copy of a desired memory location is in a cache.
IPR	Internal Processor Register

Serial Control Bus	Based on the Signetics I ² C; this bus is used on the Cobra System for intercommunication of system components as well as error reporting. Refer to the Cobra System Overview for further details.
Masked Write	A write cycle that only updates a subset of nominal data block. (for example, A single longword update to a cache block)
Memory Like	<ul style="list-style-type: none"> • Each page frame in the region either exists in its entirety or does not exist in its entirety; there are no holes within a page frame. • All locations are read/write. • A write to a location followed by a read from that location returns precisely the bits written; all bits act as memory. • A write to one location does not change any other location. • Reads have no side effects. • Longword access granularity is provided. • Instruction-fetch is supported. • Load-locked and store-conditional are supported.
MISS	Indicates that a copy of a desired memory location is not in a cache.
Non-memory Like	Non-memory like regions may have arbitrary behavior. There may be un-implemented locations or bits anywhere; some locations or bits may be read-only and others write-only; address ranges may overlap, such that a write to one location changes the bits read from a different location; reads may have side effects; longword granularity need not be supported; instruction-fetch need not be supported; and load-locked and store-conditional need not be supported.
P-Cache	The Primary Cache memory subsystem inside the 21064 processor including its control machines. The Primary Cache is the cache that is the fastest and closest to the processor.
PROBE	The act of using the current operation address (Cobra-bus or processor) to perform a cache line lookup to determine if the line is valid and/or must be invalidated, updated, or returned.
PROCESS FATAL ERROR	An error that is only fatal to the process(es) whose context the error occurred in.
Read-Merge	The term Read-Merge indicates that an item is read from a responder/bystander, and new data is then added to the returned read data. This occurs when a masked write cycle is requested by the processor, or when unmasked cycles occur and the CPU is configured to allocate on full block write misses.
RESPONDER	A Cobra-bus node which accepts or supplies data in response to an address and command from a Cobra-bus commander.
SHARED	With reference to a cache block in the cache of a Cobra-bus node, the cache block is valid and it is valid in at least one other cache of another Cobra-bus node.
SNOOP	For a cached node, the act of monitoring Cobra-bus transactions to determine whether the node has a copy of a cache line.
Snooping Protocol	A cache coherence protocol whereby all nodes on a common system bus monitor all bus activity. This allows a node to keep it's copy of a particular datum up to date, and/or supply data to the bus when it has the newest copy. Refer to the Cobra System Bus Specification and Chapter 5 for further details.
SYSTEM FATAL ERROR	An error that is fatal to system as the error occurred in the context of a system process, or if the context of an error can not be determined.
TRANSACTION	A sequence of cycles which comprise a complete bus operation.

Unmasked Write	A write cycle that updates all locations of a nominal data block. (for example, A hexaword update to a cache block)
VICTIM	With reference to a cache block in the cache of a Cobra-bus node, the cache block is valid but is about to be replaced due to a cache block resource conflict.
Victim Processing	The process of replacing a VICTIM in the cache.

CHAPTER 1

CPU MODULE COMPONENTS AND FEATURES

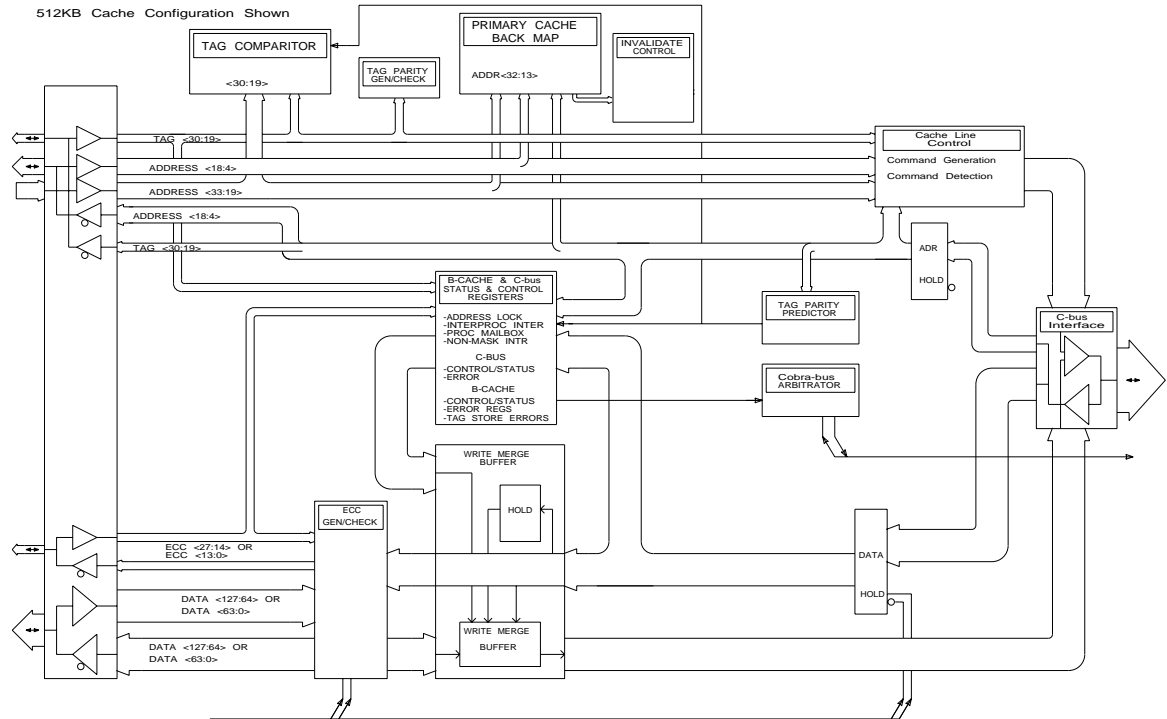
The Sable CPU module consists of 11 subsystems listed below:

- DECchip 21064 processor
- Back-up Cache
- Write Merge Buffer
- Duplicate Tag Store
- System-bus interface
- Clock Generator/Distributor
- Arbiter
- Reset Generator
- D-bus serial ROM/Micro
- Clock/Power detect circuitry
- Address Lock to support the Load Lock/Store Conditional construct

Figure 1 shows a high-level block diagram of the CPU module.

Many of the CPU module's subsystems are incorporated into a single ASIC chip, the C4 chip, developed in the AVS Engineering Group. These subsystems include the Write Merge Buffer, Duplicate Tag Store, System-bus interface, and arbitrator, the Address Lock, and the control machines for B-Cache management and System-bus Snooping. Figure 2 shows a block diagram of the C4 chip.

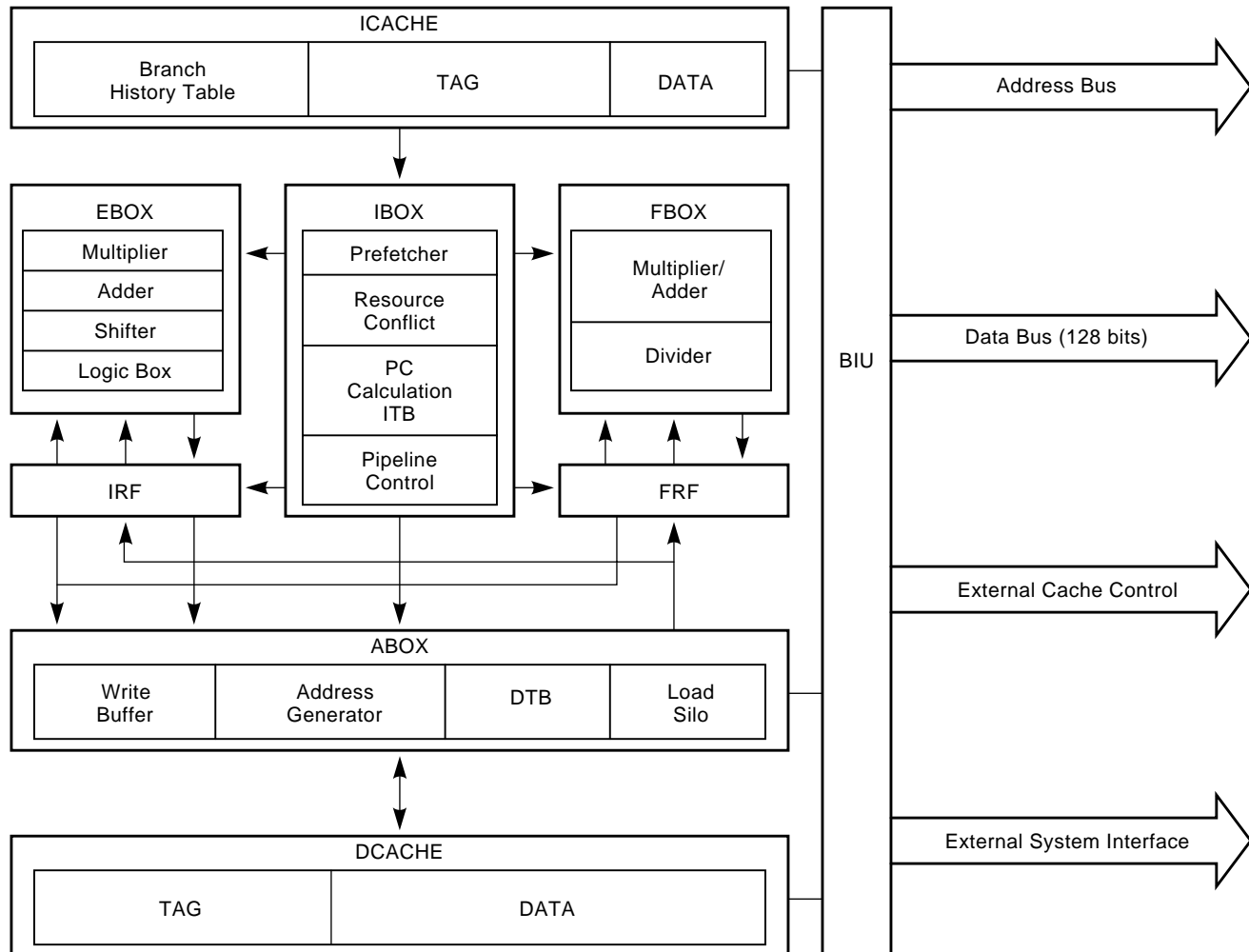
Figure 2: C4 Chip Block Diagram



1.1 The 21064 CPU Chip

The 21064 CPU chip is a 431-pin PGA CMOS-4 chip that contains approximately 2.3 million transistors. The chip is a super-scalar, super-pipelined implementation of the Digital AXP Architecture. The DECchip 21064-A275 has basically the same pinning, but is a CMOS-5 chip containing a larger Icache and Dcache. Figure 3 shows a block diagram of the CPU chip and Section 1.2 provides a brief description of the architecture.

Figure 3: 21064 CPU Chip Block Diagram



LJ-02107-T10

4 CPU Module Components and Features

1.2 The Alpha AXP Architecture

The Alpha AXP architecture is a 64-bit load/store RISC architecture designed with particular emphasis on speed, multiple instruction issue, multiple processors, and software migration from VAX/VMS and MIPS/ULTRIX.

All registers are 64 bits in length and all operations are performed between 64-bit registers. All instructions are 32 bits in length. Memory operations are either loads or stores. All data manipulation is done between registers.

The Alpha AXP architecture supports the following data types:

- 8-, 16-, 32- and 64-bit integers
- IEEE 32-bit and 64-bit floating point formats
- VAX computer 32-bit and 64-bit floating point formats

In the Alpha AXP architecture, instructions interact with each other only by one instruction writing to a register or memory location and another instruction reading from that register or memory location. This use of resources makes it easy to build implementations that issue multiple cycles every CPU cycle.

The 21064 uses a set of subroutines, called privileged architecture library code (PAL-code), that is specific to a particular Alpha AXP operating system implementation and hardware platform. These subroutines provide operating system primitives for context switching, interrupts, exceptions, and memory management. These subroutines can be invoked by hardware or CALL_PAL instructions. CALL_PAL instructions use the function field of the instruction to vector to a specified subroutine. PALcode is written in standard machine code with some implementation-specific extensions to provide direct access to low-level hardware functions. PALcode supports optimizations for multiple operating systems, flexible memory management implementations, and multi-instruction atomic sequences.

The Alpha AXP architecture does byte shifting and masking with normal 64-bit register-to-register instructions; it does not include single byte load/store instructions. The software implementor must determine the precision of arithmetic traps.

For a complete introduction to the Alpha AXP architecture, see the *Alpha Architecture Handbook*.

CHAPTER 2

CPU MODULE VARIATIONS

This specification describes CPU modules that are based on the DECchip 21064 and DECchip 21064-A275 CPU chips. The module differ with respect to on-chip cache size, backup cache size, operational speed, number of system CPU's supported and form factor. This chapter attempts to define the physical and functional design differences between the Alpha-based processor modules. Table 1 lists the different CPU modules.

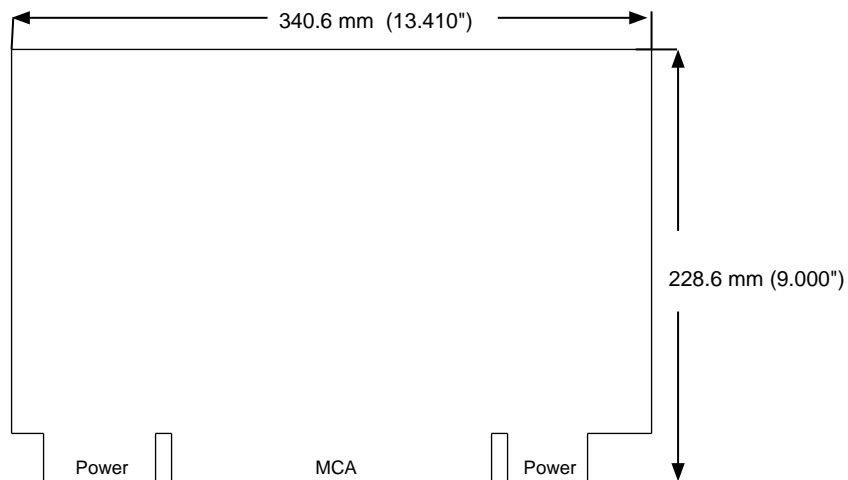
Table 1: CPU Modules Variations

Option Number	CPU/ I/D cache Chip-Size	Frequency	Cycle Time	Bcache Size	Cache Access	CPUs Supported
B2020-AA	DECchip 21064 8K/8K	190 MHZ	5.26 ns	1 MByte	26.3 ns	4
B2024-AA	DECchip 21064-A275 16K/16K	275 MHZ	3.64 ns	4 MByte	25.45 ns	4

2.1 Form Factors

The processor modules measure approximately 340.6 mm x 228.6 mm and use hi-density connectors. The Figure 4. form factor for the CPU module is shown in

Figure 4: CPU Form Factors



CPU_CARD.DOC
PS @ 100%

CHAPTER 3

FUNCTIONS LOCATED ON THE DECCHIP 21064

3.1 21064 Chip Features

The DECchip 21064 microprocessor is the first in a family of chips to implement the Alpha AXP architecture. The DECchip 21064 is a .75 micron CMOS (.5 micron for the DECchip 21064-A275) based super-scalar super-pipelined processor using dual instruction issue. The 21064 chip is packaged in a 431-pin (24x24, 100-mil pin pitch) PGA package.

The 21064 and associated PALcode implement IEEE single and double precision, VAX F_floating and G_floating datatypes and supports longword (32-bit) and quad-word (64-bit) integers. Byte (8-bit) and word (16-bit) support is provided by byte manipulation instructions. Limited hardware support is provided for the VAX D_floating datatype.

(G_floating and S_floating)

The architecturally optional instructions RCC is also implemented.

Other 21064 features include:

- Peak instruction execution of 380 million operations per second at a 190-MHz clock rate (275 MHz for DECchip 21064-A275)
- Flexible external interface supporting a complete range of system sizes and performance levels while maintaining peak CPU execution speed
- Selectable data bus width and speed: 64-bit or 128-bit width, 75 MHz to 18.75 MHz bus speed
- An internal clock generator providing a high-speed chip clock and a pair of programmable system clocks (CPU/2 to CPU/8)
- Support for external secondary cache including programmable cache size and speed
- An on-chip write buffer with four 32-byte entries, with “byte merging” capability.
- An on-chip pipelined floating point unit capable of executing both DEC and IEEE floating point instructions. The floating point unit can accept a new instruction every cycle, except for divide instructions. The operate-to-operate latency for all instructions other than divide is six CPU cycles. The latencies for single and double precision divide instructions are shown in Table 2.

Table 2: Latencies for Single and Double Precision Divide Instructions

	DECchip 21064	DECchip 21064-A275
single	34	19
double	64	29

- An on-chip 8K byte for DECchip 21064 (or 16K byte two-way set associative DECchip 21064-A275) direct mapped, write through physical data cache with a block size of 32 bytes.
- An on-chip 8K byte for DECchip 21064 and 16K byte for DECchip 21064-A275 direct mapped, read-only physical (managed as a virtual) instruction cache with a block size of 32 bytes.
- An on-chip demand paged memory management unit which, in conjunction with properly written PALcode, fully implements the Alpha memory management architecture. The translation buffer can be used with alternative PALcode to implement a different page table structure. The memory mangament unit contains the following:
 - A 12-entry I-stream translation buffer with 8 entries for 8K byte pages and 4 entries for 4 MB pages
 - A 32-entry D-stream translation buffer with each entry able to map a single 8K, 64K, 512K, or 4 MB page
- A demand page memory management unit
- On-chip parity and ECC support
- 3.3-volt Power supply with interface to 5-volt logic
- An on-chip 8-entry I-stream Translation Buffer for mapping 8KB physical pages and a 4-entry I-stream Translation Buffer for mapping groups of up to 512 contiguous 8KB pages. A 32-entry D-stream Translation Buffer for mapping 8KB physical pages, and a 4-entry data stream translation buffer for mapping aligned groups of 512 contiguous 8KB pages.
- The 21064 will implement a dynamic branch prediction algorithm using a 2KB x 1 bit branch history table. An internal control register bit will be used to select between these two algorithms.
- The Alpha AXP Architecture includes scaled add instructions that improve the performance of address calculations for longword and quadword length array elements. The 21064's integer execution unit will be changed to support these instructions.
- Low average cycles per instruction (CPI). The 21064 chip can issue two Alpha instructions in a single cycle, thereby minimizing the aver CPI. Branch history tables are also used to minimize the branch latency, further reducing the average CPI.
- A single-entry stream buffer will be used to prefetch 32-byte instruction cache blocks.

- A bus interface unit, that contains logic that directly controls one bank of external cache RAMs, and processes “easy” cycles without any CPU module action.
- A serial ROM interface for booting and diagnostics for chip and module level test support
- A clock generator.

The 21064 CPU chip consists of three independent execution units: integer execution unit (E-box), floating point unit (F-box), and the address generation, memory management, write buffer and bus interface unit (A-box). Each unit can accept at most one instruction per cycle, however if code is properly scheduled, this CPU chip can issue two instructions to two independent units in a single cycle. A fourth box, the (I-box), is the central control unit. It issues instructions, maintains the pipeline, and performs all of the PC calculations.

3.1.1 I-box Internal Processor Registers

The primary function of the I-box is to issue instructions to the E-box, A-box, and F-box. The I-box decodes two instructions in parallel and checks that the required resources are available for both instructions. Section 3.1.6 through Section 3.1.25 describe the I-box internal processor registers. In order to provide those instructions, the Ibox contains:

- The prefetcher
- PC pipeline
- ITB
- Abort logic
- Register conflict or dirty logic
- Exception logic

The Ibox decodes two instructions in parallel and checks that the required resources are available for both instructions.

If resources are available then both instructions are issued. See Section 3.8.5 for details on instructions that can be dual issued. The Ibox does *not* issue instructions out of order; if the resources are available for the second instruction, but not for the first instruction, then the Ibox issues neither. The resources for the first instruction must be available before the Ibox issues any instructions. If the Ibox issues only the first of a pair of instructions, the Ibox does not advance another instruction to attempt dual issue again. Dual issue is only attempted on aligned quadword pairs.

3.1.2 Branch Prediction Logic

The Ibox contains the branch prediction logic. The 21064 offers a choice of branch prediction strategies selectable through the ICCSR IPR. The Icache records the outcome of branch instructions in a single history bit provided for each instruction location in the Icache. This information can be used as the prediction for the next execution of

the branch instruction. The prediction for the first execution of a branch instruction is based on the sign of the displacement field within the branch instruction itself.

- If the sign bit is negative, the instruction prefetcher predicts the conditional branches to be taken.
- If the sign is positive, the instruction prefetcher predicts the conditional branches not to be taken.
- Alternatively, if the history table is disabled, branches can be predicted based on the sign of the displacement field at all times.

The 21064 provides a four-entry subroutine return stack that is controlled by the hint bits in the BSR, HW_REI, and jump to subroutine instructions (JMP, JSR, RET, or JSR_COROUTINE). The chip also provides a means of disabling all branch prediction hardware.

3.1.3 Instruction Translation Buffers (ITBs)

The Ibox contains two ITBs.

- An eight-entry, fully associative translation buffer that caches recently used instruction-stream page table entries for 8K byte pages
- A four-entry, fully associative translation buffer that supports the largest granularity hint option (512 * 8K byte pages) as described in the *Alpha Architecture Handbook*.

Both translation buffers use a not-last-used replacement algorithm. They are hereafter referred to as the small-page and large-page ITBs, respectively.

In addition, the ITB includes support for an extension called the super page, which can be enabled by the MAP bit in the ICCSR IPR. Super page mappings provide one-to-one virtual PC [33:13] to physical PC [33:13] translation when virtual address bits [42:41] = 2. When translating through the super page, the PTE[ASM] bit used in the Icache is always set. Access to the super page mapping is only allowed while executing in kernel mode.

PALcode fills and maintains the ITBs. The operating system, through PALcode, is responsible for ensuring that virtual addresses can only be mapped through a single ITB entry (in the large page, small page, or super page) at the same time.

The Ibox presents the 43-bit virtual program counter (VPC) to the ITB each cycle while not executing in PALmode. If the PTE associated with the VPC is cached in the ITB then the Ibox uses the PFN and protection bits for the page that contains the VPC to complete the address translation and access checks.

The 21064 ITB supports a single address space number (ASN) via the PTE[ASM] bit. Each PTE entry in the ITB contains an address space match (ASM) bit. Writes to the ITBASM IPR invalidate all entries that do not have their ASM bit set. This provides a simple method of preserving entries that map operating system regions while invalidating all others.

3.1.4 Interrupt Logic

The 21064 chip supports three sources of interrupts.

- Hardware

There are six level-sensitive hardware interrupts sourced by pins.

- Software

There are fifteen software interrupts sourced by an on-chip IPR (SIRR).

- Asynchronous system trap (AST)

There are four AST interrupts sourced by a second internal IPR (ASTRR).

All interrupts are independently maskable by on-chip enable registers to support a software controlled mechanism for prioritization. In addition, AST interrupts are qualified by the current processor mode and the current state of SIER [2].

By providing distinct enable bits for each independent interrupt source, a software controlled interrupt priority scheme can be implemented by PALcode or the operating system with maximum flexibility.

For example, the 21064 can support a six-level interrupt priority scheme through the six hardware interrupt request pins. This is done by defining a distinct state of the hardware interrupt enable register (HIER) for each interrupt priority level (IPL). The state of the HIER determines the current interrupt priority. The lowest interrupt priority level is produced by enabling all six interrupts, for example bits <6:1>. The next is produced by enabling bits <6:2> and so on, to the highest interrupt priority level that is produced by enabling only bit 6, and disabling bits <5:1>. When all interrupt enable bits are cleared, the processor can not be interrupted from the hardware interrupt request register (HIRR). Each state, (<6:1>, <6:2>, <6:3>, <6:4>, <6:5>, <6>) represents an individual IPL. If these states are the only states allowed in the HIER, a six-level hardware interrupt priority scheme can be controlled entirely by PAL software.

The scheme is extensible to provide multiple interrupt sources at the same interrupt priority level by grouping enable bits. Groups of enable bits must be set and cleared together to support multiple interrupts of equal priority level. This method reduces the total available number of distinct levels.

Because enable bits are provided for all hardware, software, and AST interrupt requests, a priority scheme can span all sources of processor interrupts. The only exception to this rule is the following restriction on AST interrupt requests:

Four AST interrupts are provided, one for each processor mode. AST interrupt requests are qualified such that AST requests corresponding to a given mode are blocked whenever the processor is in a higher mode regardless of the state of the AST interrupt enable register. In addition, all AST interrupt requests are qualified in the 21064 with SIER[2].

When the processor receives an interrupt request and that request is enabled, hardware reports or delivers an interrupt to the exception logic if the processor is not currently executing PALcode. Before vectoring to the interrupt service PAL dispatch address, the pipeline is completely drained and all outstanding data cache fills are completed. The restart address is saved in the Exception Address IPR (EXC_ADDR) and the processor enters PALmode. The cause of the interrupt may be determined by examining the state of the interrupt request registers.

NOTE

Hardware interrupt requests are level-sensitive and, therefore, may be removed before an interrupt is serviced. If they are removed before the interrupt request register is read, the register will return a zero value.

3.1.5 Performance Counters

The 21064 chip contains a performance recording feature. The implementation of this feature provides a mechanism to count various hardware events and cause an interrupt upon counter overflow. Interrupts are triggered six cycles after the event, and therefore, the exception program counter may not reflect the exact instruction causing counter overflow. Two counters are provided to allow accurate comparison of two variables under a potentially non-repeatable experimental condition. Counter inputs include:

- Issues
- Non-Issues
- Total cycles
- Pipe dry
- Pipe freeze
- Mispredicts and cache misses
- Counts for various instruction classifications

In addition, the 21064 provides one chip pin input to each counter to measure external events at a rate determined by the selected system clock speed.

They are reset to zero on power-up, but are otherwise never cleared. They are intended as a means of counting events over a long period of time relative to the event frequency and therefore provide no means of extracting intermediate counter values. Because the counters continuously accumulate selected events despite interrupts being enabled, the first interrupt after selecting a new counter input has an error bound as large as the selected overflow range.

In addition, some inputs may over count events occurring simultaneously with D-stream errors which abort the actual event very late in the pipeline. For example, when counting load instructions, attempts to execute a load resulting in a DTB miss exception will increment the performance counter after the first aborted execution attempt and again after the TB fill routine when the load instruction reissues and completes.

Performance counter interrupts are reported six cycles after the event that caused the counter to overflow. Additional delay may occur before an interrupt is serviced if the processor is executing PALcode which always disables interrupts. In either case, events occurring during the interval between counter overflow and interrupt service are counted toward the next interrupt. Only in the case of a complete counter wraparound while interrupts are disabled will an interrupt be missed.

The six cycles before an interrupt is triggered implies that a maximum of 12 instructions may have completed before the start of the interrupt service routine. When counting I-Cache misses, no intervening instructions can complete and the exception PC contains the address of the last I-Cache miss. Branch "mispredictions" allow a maximum of only 2 instructions to complete before start of the interrupt service routine.

Table 3: Performance Counter 0 Input Selection

MUX0[3:0]	Input	Comment
000X	Total Issues / 2	Counts total issues divided by 2, e.g dual issue increments count by 1
001X	Pipeline Dry	Counts cycles where nothing issued due to lack of valid I-stream data. Causes include I-Cache fill, misprediction, branch delay slots and pipeline drain for exception
010X	Load Instructions	Counts all Load instructions
011X	Pipeline Frozen	Counts cycles where nothing issued due to resource conflict.
100X	Branch Instructions	Counts all Branch instructions, conditional, unconditional, any JSR, HW_REI
1011	PALmode	Counts cycles while executing in PALmode
1010	Total cycles	Counts total cycles
110X	Total Non-issues / 2	Counts total non_issues divided by 2, e.g no issue increments count by 1
111X	PERF_CNT_H	Counts external event supplied by pin at selected system clock cycle interval

Table 4: Performance Counter 1 Input Selection

MUX1[2:0]	Input	Comment
000	D-Cache miss	Counts total D-Cache misses
001	I-Cache miss	Counts total I-Cache misses
010	Dual issues	Counts cycles of Dual issue
011	Branch Mispredicts	Counts both conditional branch mispredictions and JSR or HW_REI mispredictions. Conditional branch mispredictions cost 4 cycles and others cost 5 cycles of dry pipeline delay.
100	FP Instructions	Counts total floating point operate instructions, i.e no FP branch, load, store
101	Integer Operate	Counts integer operate instructions including LDA, LDAH with destination other than R31
110	Store Instructions	Counts total store instructions
111	PERF_CNT_H	Counts external event supplied by pin at selected system clock cycle interval

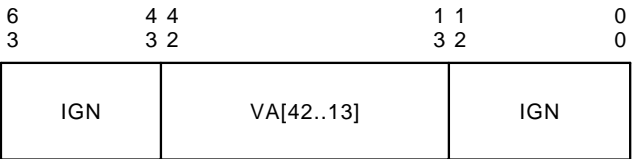
3.1.6 Translation Buffer Tag Register (TB_TAG)

The TB_TAG register is a write-only register that holds the tag for the next TB update operation in either the ITB or DTB. To ensure the integrity of the TB, the tag is actually written to a temporary register and not transferred to the ITB or DTB until the ITB_PTE or DTB_PTE register is written. The entry to be written is chosen at the time of the ITB_PTE or DTB_PTE write operation by a not-last-used algorithm implemented in hardware.

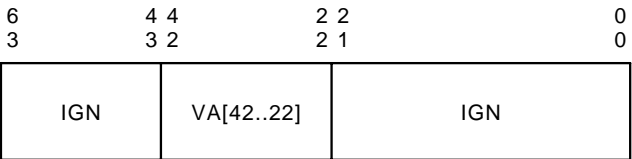
Writing the ITB_TAG register is only performed while in PALmode regardless of the state of the HWE bit in the ICCSR IPR.

Figure 5: TB_TAG_FIG

Small Page Format:



GH = 11(bin) Format (DTB only):



TB_TAG

3.1.7 Instruction Translation Buffer Page Table Entry Register (ITB_PTE)

The ITB PTE register is a read/write register representing the twelve ITB page table entries. The first eight provide small page (8K byte) translations while the remaining four provide large page (4M byte) translations. The entry to be written is chosen by a not-last-used algorithm implemented in hardware. Writes to the ITB_PTE use the memory format bit positions as described in the *Alpha Architecture Handbook* with the exception that some fields are ignored.

To ensure the integrity of the ITB, the ITB's tag array is updated simultaneously from the internal tag register when the ITB_PTE register is written. Reads of the ITB_PTE require two instructions. First, a read from the ITB_PTE sends the PTE data to the ITB_PTE_TEMP register, then a second instruction reading from the ITB_PTE_TEMP register returns the PTE entry to the register file. Reading or writing the ITB_PTE register increments the TB entry pointer corresponding to the large/small page selection indicated by the TB_CTL, which allows reading the entire set of ITB_PTE register entries.

Reading and writing the ITB_PTE register is only performed while in PALmode regardless of the state of the HWE bit in the ICCSR IPR.

Figure 6: ITB_PTE

Write Format:

6	5 5	3 3	1 1 1 0 0 0 0 0 0 0 0 0 0 0 0						
3	3 2	2 1	2 1 0 9 8 7 6 5 4 3 2 1 0						
IGN	PFN[33..13]	IGN	U R E	S R E	E R E	K R E	IGN	A S M	IGN

Read Format:

6	3 3	1 1 1 1 0 0 0 0 0 0 0 0 0 0 0					
3	4 3	3 2 1 0 9 8 7 6 5 4 3 2 1 0					
RAZ	A S M	PFN[33..13]	U R E	S R E	E R E	K R E	RAZ

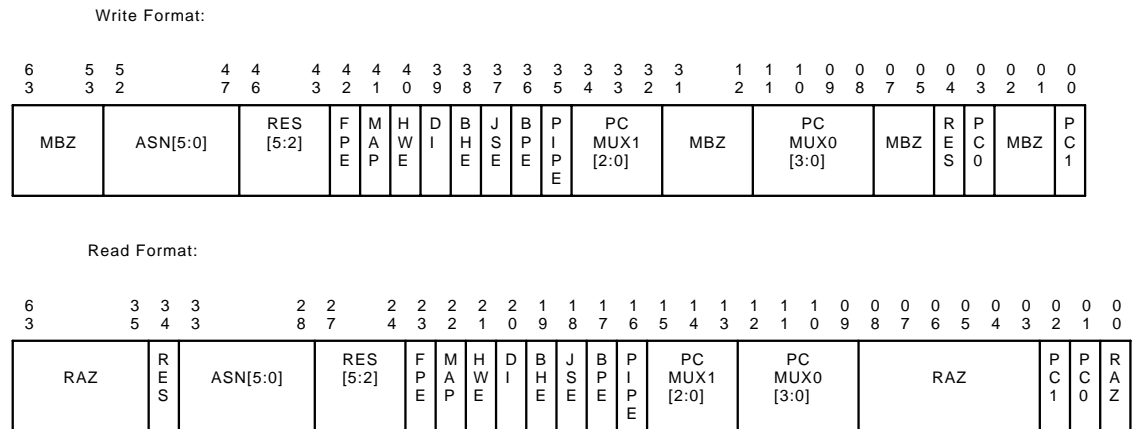
ITB_PTE

3.1.8 Instruction Cache Control and Status Register (ICCSR)

The ICCSR register contains various Ibox hardware enables. The only architecturally defined bit in this register is the FPE, floating point enable, which enables floating point instruction execution. When clear, all floating point instructions generate FEN exceptions. Most of this register is cleared by hardware at reset. Fields which are not cleared at reset include ASN, PC0, and PC1.

The HWE bit allows the special PAL instructions to execute in kernel model. This bit is intended for diagnostics or operating system alternative PAL routines only. It does not allow access to the ITB registers while not running in PALmode. Therefore, some PALcode flows may require the PALmode environment to execute properly (e.g. ITB fill).

Figure 7: ICCSR



ICCSR

Table 5: ICCSR

Field	Type	Description
FPE	RW,0	If set, floating point instructions can be issued. If clear, floating point instructions cause FEN exceptions.
HWE	RW,0	If set allows the five PALRES instructions to be issued in kernel mode. If cleared, attempted execution of PALRES instructions while not in PALmode result in OPCDEC exceptions.
MAP	RW,0	If set allows super page I-stream memory mapping of VPC<33:13> directly to Physical PC<33:13> essentially bypassing ITB for VPC addresses containing VPC<42:41>=2. Super page mapping is allowed in Kernel mode only. The ASM bit is always set. If clear, superpage mapping is disabled.

Table 5 (Cont.): ICCSR

Field	Type	Description
DI	RW,0	If set enables dual issue. If cleared, instructions can only single issue.
BHE	RW,0	Used in conjunction with BPE. See table Table 6 for programming information.
JSE	RW,0	If set enables the JSR stack to push return addresses. If cleared, JSR stack is disabled.
BPE	RW,0	Used in conjunction with BHE. See table Table 6 for programming information.
PIPE	RW,0	If clear causes all hardware interlocked instructions to drain the machine and waits for the write buffer to empty before issuing the next instruction. Examples of instructions that do not cause the pipe to drain include HW_MTPR, HW_REI, conditional branches, and instructions that have a destination register of R31. If set, pipeline proceeds normally.
PCMUX1	RW,0	See table Table 4 for programming information. Performance counters are present only in the 21064.
PCMUX0	RW,0	See table Table 3 for programming information. Performance counters are present only in the 21064.
PC1	RW	If clear enables performance counter 1 interrupt request after 2^{12} events counted. If set enables performance counter 1 interrupt request after 2^8 events counted.
PC0	RW	If clear enables performance counter 0 interrupt request after 2^{16} events counted. If set enables performance counter 0 interrupt request after 2^{12} events counted.
ASN	RW	The Address Space Number field is used in conjunction with the I-Cache in the 21064 to further qualify cache entries and avoid some cache flushes. The ASN is written to the I-Cache during fill operations and compared with the I-stream data on fetch operations. Mismatches invalidate the fetch without affecting the I-Cache. This function is only present in the 21064.
RES	RW,0	The RES state bits are reserved by Digital and should not be used by software.
ICCSR<45:44>	RW,0	When set the performance counters will be enabled and will increment. During normal operation these bits should be written with 0 to disable the performance counters.

NOTE

Use of HW_MTPR instruction to update the EXC_ADDR IPR while in native mode is restricted to values with bit<0> equal to 0. The combination of native mode and EXC_ADDR<0> equal to one causes UNDEFINED behavior.

Table 6: BHE,BPE Branch Prediction Selection

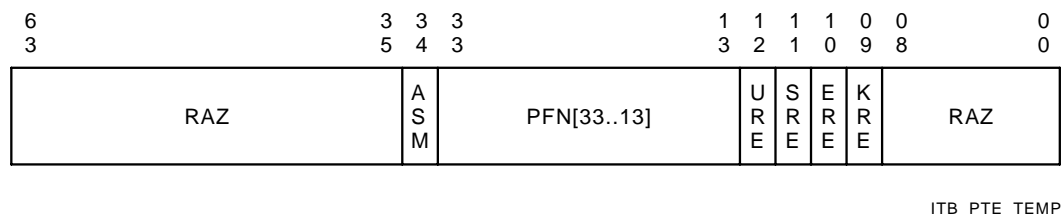
BPE	BHE	Prediction
0	X	Not Taken
1	0	Sign of Displacement
1	1	Branch History Table

3.1.9 Instruction Translation BUffer Page Table Entry Temporary Register(ITB_PTE_TEMP)

The ITB_PTE_TEMP register is a read-only holding register for ITB_PTE read data. Reads of the ITB_PTE require two instructions to return the data to the register file. The first reads the ITB_PTE register to the ITB_PTE_TEMP register. The second returns the ITB_PTE_TEMP register to the integer register file. The ITB_PTE_TEMP register is updated on all ITB accesses, both read and write. A read of the ITB_PTE to the ITB_PTE_TEMP should be followed closely by a read of the ITB_PTE_TEMP to the register file.

Reading the ITB_PTE_TEMP register is only performed while in PALmode regardless of the state of the HWE bit in the ICCSR IPR.

Figure 8: ITB_PTE_TEMP



3.1.10 Exception Address Register (EXC_ADDR)

The EXC_ADDR register is a read/write register used to restart the machine after exceptions or interrupts. The EXC_ADDR register can be read and written by software by way of the HW_MTPR instruction as well as being written directly by hardware.

The HW_REI instruction executes a jump to the address contained in the EXC_ADDR register. The EXC_ADDR register is written by hardware after an exception to provide a return address for PALcode.

The instruction pointed to by the EXC_ADDR register did not complete execution. Since the PC is longword aligned, the lsb of the EXC_ADDR register is used to indicate PALmode to the hardware. When the lsb is clear, the HW_REI instruction executes a jump to native (non-PAL) mode, enabling address translation.

CALL_PAL exceptions load the EXC_ADDR with the PC of the instruction following the CALL_PAL. This function allows CALL_PAL service routines to return without needing to increment the value in the EXC_ADDR register.

This feature, however, requires careful treatment in PALcode. Arithmetic traps and machine check exceptions can pre-empt CALL_PAL exceptions resulting in an incorrect value being saved in the EXC_ADDR register. In the cases of an arithmetic trap or a machine check exception, and only in these cases, EXC_ADDR<1> takes on special meaning. PALcode servicing these two exceptions should interpret a zero in EXC_ADDR<1> as indicating that the PC in EXC_ADDR<63:2> is too large by a value of 4 bytes and subtract 4 before executing a HW_REI from this address. PALcode should interpret a one in EXC_ADDR<1> as indicating that the PC in EXC_

ADDR<63:2> is correct and clear the value of EXC_ADDR<1>. All other PALcode entry points except reset can expect EXC_ADDR<1> to be zero.

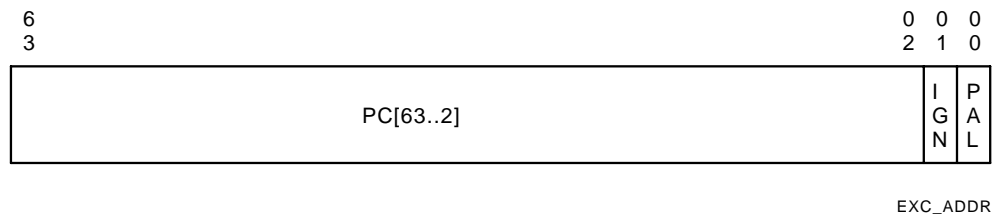
This logic allows the following code sequence to conditionally subtract 4 from the address in the EXC_ADDR register without use of an additional register. This code sequence should be present in arithmetic trap and machine check flows only.

```
HW_MFPR Rx, EXC_ADDR ; read EXC_ADDR into GPR
SUBQ Rx, #2, Rx ; subtract 2 causing borrow if [1]=0
BIC Rx, #2, Rx ; clear [1]
HW_MTPR Rx, EXC_ADDR ; write back to EXC_ADDR
```

Note that bit 1 is undefined when the EXC_ADDR is read. The actual hardware ignores this bit, however PALcode must explicitly clear this bit before it pushes the exception address on the stack.

IPR Format:

Figure 9: EXC_ADDR



3.1.11 Serial Line Clear Register (SL_CLR)

This write-only register clears the serial line interrupt request, the performance counter interrupt request and the CRD interrupt request. Therefore, the write of any data to the SL_CLR register will clear the remaining serial line interrupt request. The 21064 requires that the indicated bit be written with a zero to clear the selected interrupt source.

Figure 10: SL_CLR

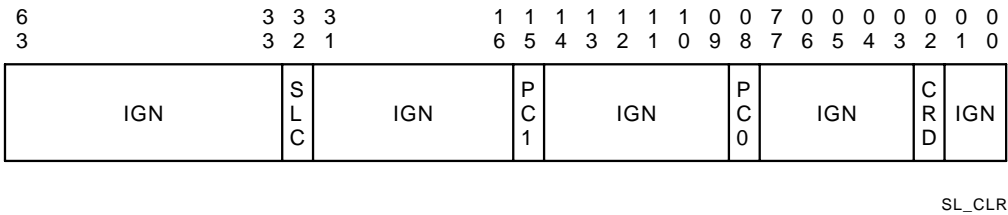
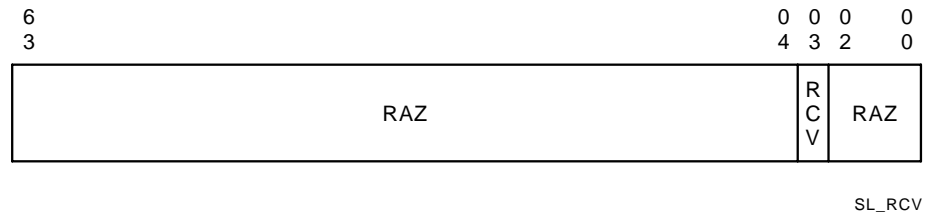


Table 7: SL_CLR		
Field	Type	Description
CRD	W0C	Clears the correctable read error interrupt request.
PC1	W0C	Clears the performance counter 1 interrupt request.
PC0	W0C	Clears the performance counter 0 interrupt request.
SLC	W0C	Clears the serial line interrupt request.

3.1.12 Serial Line Receive Register (SL_RCV)

The serial line receive register contains a single read-only bit used with the interrupt control registers and the sRomD_h and sRomClk_h pins to provide an on-chip serial line function. The RCV bit is functionally connected to the sRomD_h pin after the I-Cache is loaded from the external serial ROM. Reading the RCV bit can be used to receive external data one bit at a time under a software timing loop. A serial line interrupt is requested on detection of any transition on the receive line which sets the SL_REQ bit in the HIRR. Using a software timing loop, the RCV bit can be read to receive data one bit at a time. The serial line interrupt can be disabled by clearing the HIER register SL_ENA bit. (See figure Figure 11)

Figure 11: SL_RCV



3.1.13 Instruction Translation Buffer Zap Register (ITBZAP)

A write of any value to this IPR invalidates all twelve ITB entries. It also resets the NLU pointer to its initial state. The ITBZAP register should only be written in PALmode.

3.1.14 Instruction Translation Buffer ASM Register (ITBASM)

A write of any value to this IPR invalidates all ITB entries in which the ASM bit is equal to zero. The ITBASM register should only be written in PALmode.

3.1.15 Instruction Translation Buffer IS Register (ITBIS)

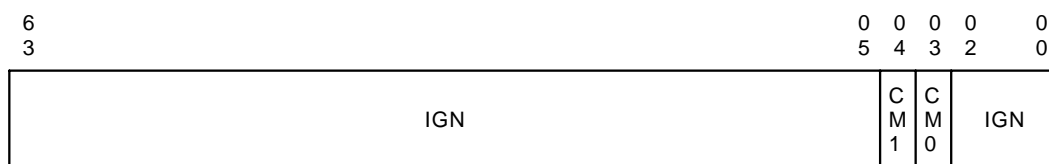
A write of any value to this IPR invalidates all twelve ITB entries. It also resets both the NLU pointer to their initial state. The ITBIS register should only be written in PALmode.

3.1.16 Processor Status (PS)

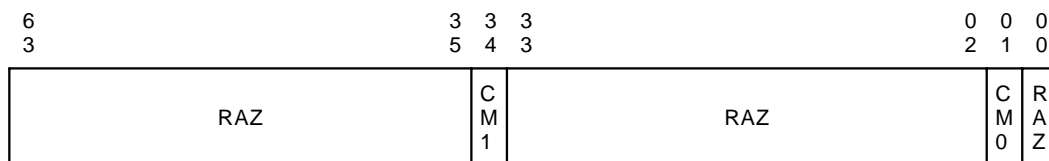
The processor status register is a read/write register containing only the current mode bits of the architecturally defined PS.

Figure 12: PS

Write Format:



Read Format:



PS

3.1.17 Exception Summary Register (EXC_SUM)

The exception summary register records the various types of arithmetic traps that have occurred since the last time the EXC_SUM was written (cleared). When the result of an arithmetic operation produces an arithmetic trap, the corresponding EXC_SUM bit is set.

In addition, the register containing the result of that operation is recorded in the exception register write mask IPR, as a single bit in a 64-bit field specifying registers F31-F0 and I31-I0. This IPR is visible only through the EXC_SUM register. The EXC_SUM register provides a one-bit window to the exception register write mask. Each read to the EXC_SUM shifts one bit in order F31-F0 then I31-I0. The read also clears the corresponding bit. Therefore, the EXC_SUM must be read 64 times to extract the complete mask and clear the entire register.

Any write to EXC_SUM clears bits [8..2] and does not affect the write mask.

The write mask register bit clears three cycles after a read. Therefore, code intended to read the register must allow at least three cycles between reads to allow the clear and shift operation to complete in order to ensure reading successive bits.

Figure 13: EXC_SUM

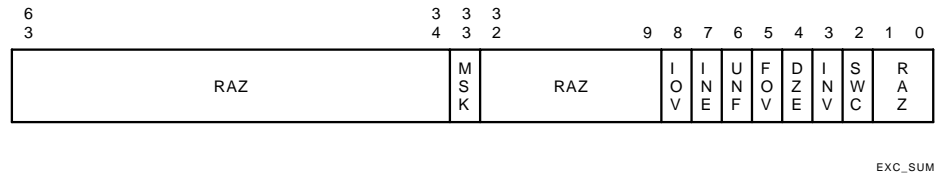


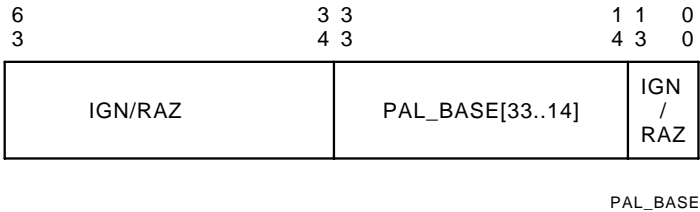
Table 8: EXC_SUM

Field	Description	Type
33	MSK - Exception Register Write Mask IPR Window.	RC
8	IOV - Indicates Fbox Convert to Integer Overflow or Integer Arithmetic Overflow	WA
7	INE - Indicates Floating Inexact Error	WA
6	UNF - Indicates Floating Point Underflow	WA
5	FOV - Indicates Floating Point Overflow	WA
4	DZE - Indicates Divide by Zero	WA
3	INV - Indicates Invalid Operation	WA
2	SWC - Indicates Software Completion possible. The bit is set after a floating point instruction containing the /S modifier completes with an arithmetic trap and all previous floating point instructions that trapped since the last MTPR EXC_SUM also contained the /S modifier. The SWC bit is cleared whenever a floating point instruction without the /S modifier completes with an arithmetic trap. The bit remains cleared regardless of additional arithmetic traps until the register is written via an MTPR instruction. The bit is always cleared upon any MTPR write to the EXC_SUM register.	WA

3.1.18 Privileged Architecture Library Base Register (PAL_BASE)

The PAL base register is a read/write register containing the base address for PAL-code. This register is cleared by hardware at reset.

Figure 14: PAL_BASE



3.1.19 Hardware Interrupt Request Register (HIRR)

The Hardware Interrupt Request Register is a read-only register providing a record of all currently outstanding interrupt requests and summary bits at the time of the read. For each bit of the HIRR [5:0] there is a corresponding bit of the HIER (Hardware Interrupt Enable Register) that must be set to request an interrupt. In addition to returning the status of the hardware interrupt requests, a read of the HIRR returns the state of the software interrupt and AST requests. Note that a read of the HIRR may return a value of zero if the hardware interrupt was released before the read (passive release). The register guarantees that the HWR bit reflects the status as shown by the HIRR bits. All interrupt requests are blocked while executing in PALmode.

Figure 15: HIRR

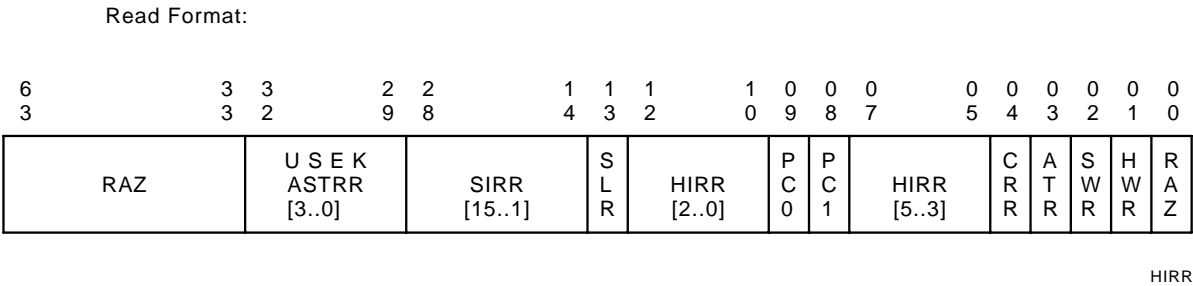


Table 9: HIRR		
Field	Description	Type
63:33	Read as Zero	
32:29	ASTRR[3..0] - Corresponds to AST request three thru zero (USEK).	RO
28:14	SIRR[15..1] - Corresponds to software interrupt request 15 thru 1.	RO
13	SLR - Serial line interrupt request. Also see SL_RCV, SL_XMIT, and SL_CLR.	RO
12:10 and 7:5	HIRR[5..0] - Corresponds to pins Irq_h[5:0]. 5 - System Event Interrupt/Node Halt 4 - Interval Timer Interrupt 3 - Interprocessor Interrupt 2 - XIO Interrupt 1 - Local I/O Interrupt 0 - Hardware Error Interrupt	RO
9	PC0 - Performance counter 0 interrupt request. Performance counters are only present in the 21064.	RO

Table 9 (Cont.): HIRR

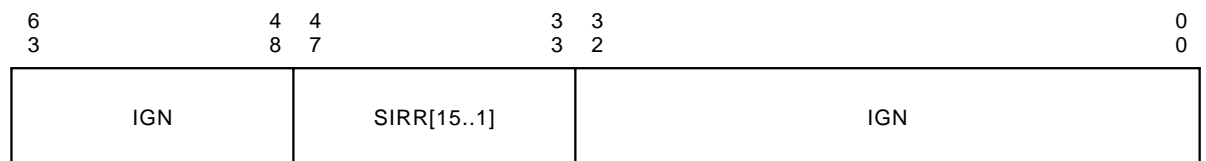
Field	Description	Type
8	PC1 - Performance counter 1 interrupt request. Performance counters are only present in the 21064.	RO
4	CRR - CRD correctable read error interrupt request. This interrupt is cleared by way of the SL_CLR register.	RO
3	ATR - Is set if any AST request and corresponding enable is set. This bit also requires that the processor mode be equal to or higher than the request mode. SIER[2] must be set to allow AST interrupt requests.	RO
2	SWR - Is set if any software interrupt request and corresponding enable is set	RO
1	HWR - Is set if any hardware interrupt request and corresponding enable is set	RO
0	Read as Zero	RO

3.1.20 Software Interrupt Request Register (SIRR)

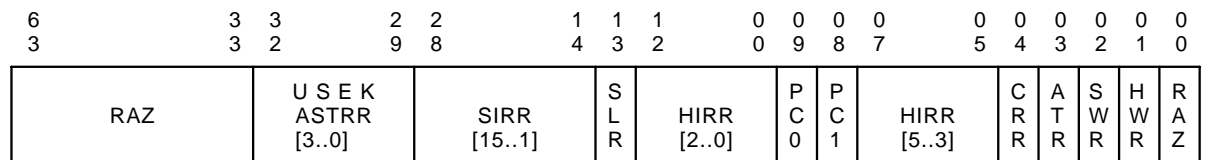
The Software Interrupt Request Register is a read/write register used to control software interrupt requests. For each bit of the SIRR there is a corresponding bit of the SIER (Software Interrupt Enable Register) that must be set to request an interrupt. Reads of the SIRR return the complete set of interrupt request registers and summary bits, see the HIRR Table 9 for details. All interrupt requests are blocked while executing in PALmode.

Figure 16: SIRR

Write Format:



Read Format:

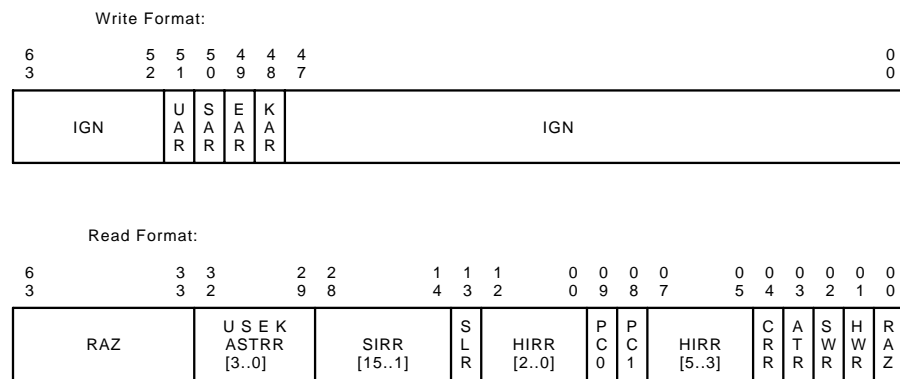


SIRR

3.1.21 Asynchronous Trap Request Register (ASTRR)

The Asynchronous Trap Request Register is a read/write register. It contains bits to request AST interrupts in each of the processor modes. In order to generate an AST interrupt, the corresponding enable bit in the ASTER must be set and the processor must be in the selected processor mode or higher privilege as described by the current value of the PS CM bits. In addition, AST interrupts are only enabled in the 21064 if the SIER[2] is set. This provides a mechanism to lock out AST requests over certain IPL levels. All interrupt requests are blocked while executing in PALmode. Reads of the ASTRR return the complete set of interrupt request registers and summary bits, see the HIRR Table 9 for details.

Figure 17: ASTRR

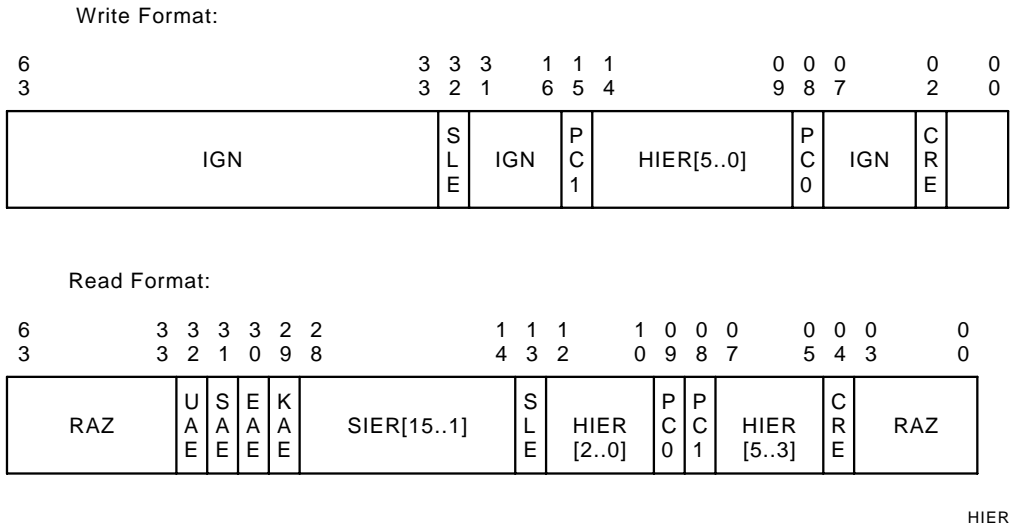


ASTRR

3.1.22 Hardware Interrupt Enable Register (HIER)

The Hardware Interrupt Enable Register is a read/write register. It is used to enable corresponding bits of the HIRR requesting interrupt. The PC0, PC1, SLE and CRE bits of this register enable the performance counters, serial line and correctable read interrupts. There is a one-to-one correspondence between the interrupt requests and enable bits, as with the reads of the interrupt request IPRs, reads of the HIER return the complete set of interrupt enable registers, see the HIRR Table 9 for details.

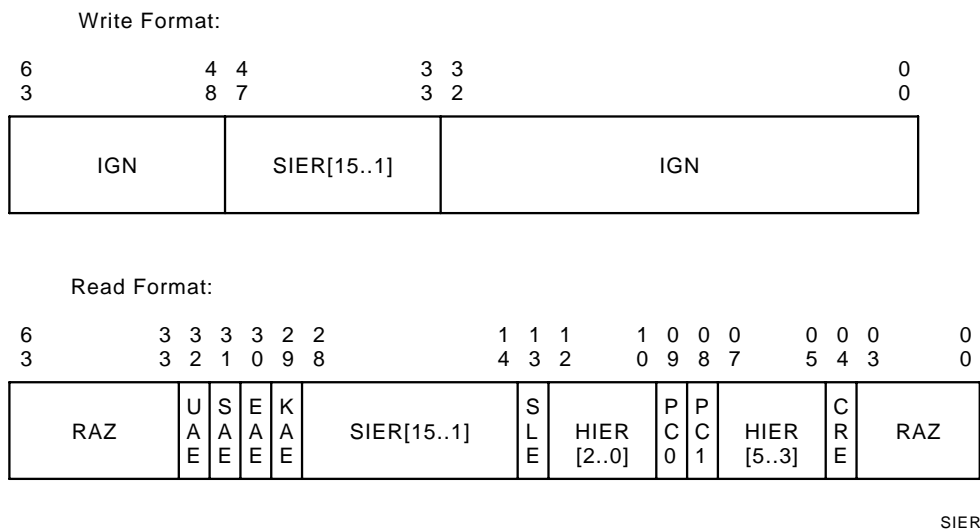
Figure 18: HIER



3.1.23 Software Interrupt Enable Register (SIER)

The Software Interrupt Enable Register is a read/write register. It is used to enable corresponding bits of the SIRR requesting interrupts. There is a one-to-one correspondence between the interrupt requests and enable bits, as with the reads of the interrupt request IPRs, reads of the SIER return the complete set of interrupt enable registers, see the HIRR Table 9 for details.

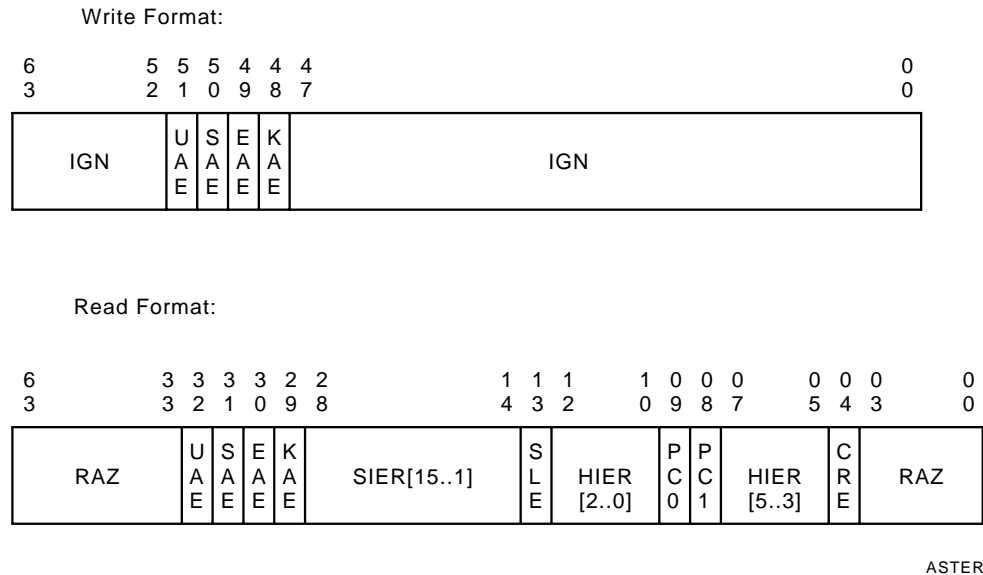
Figure 19: SIER



3.1.24 Asynchronous System Trap Enable Register (ASTER)

The AST Interrupt Enable Register is a read/write register. It is used to enable corresponding bits of the ASTRR requesting interrupts. There is a one-to-one correspondence between the interrupt requests and enable bits, as with the reads of the interrupt request IPRs, reads of the ASTER return the complete set of interrupt enable registers, see the HIRR Table 9 for details.

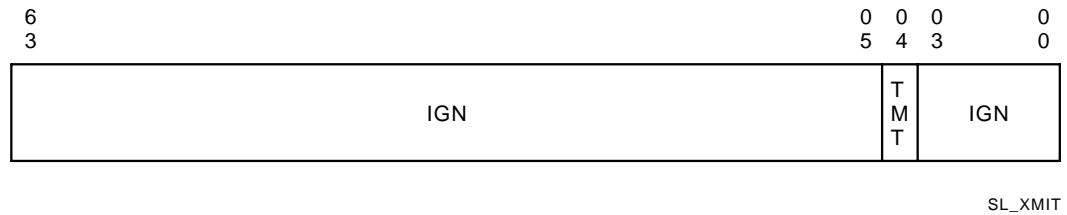
Figure 20: ASTER



3.1.25 Serial Line Transmit (SL_XMIT)

The serial line transmit register contains a single write-only bit used with the interrupt control registers and the sRomD_h and sRomClk_h pins to provide an on-chip serial line function. The TMT bit is functionally connected to the sRomClk_h pin after the I-Cache is loaded from the external serial ROM. Writing the TMT bit can be used to transmit data off chip one bit at a time under a software timing loop.

Figure 21: SL_XMIT



3.2 Ebox

The Ebox contains the 64-bit integer execution datapath.

- Adder
- Logic box
- Barrel shifter
- Byte zapper
- Bypassers
- Integer multiplier

The integer multiplier retires four bits per cycle. The Ebox also contains the 32-entry 64-bit integer register file. The register file has four read ports and two write ports that allow reading operands from and writing operands (results) to both the integer execution datapath and the Abox.

3.3 Abox

The A-box in the 21064 contains six major sections:

- Address translation datapath
- Load silo
- Write buffer
- Dcache interface
- Internal processor registers (IPRs)
- External bus interface unit (BIU)

The address translation datapath has a displacement adder that generates the effective virtual address for load and store instructions, and a translation buffer(S) that generates the corresponding physical address.

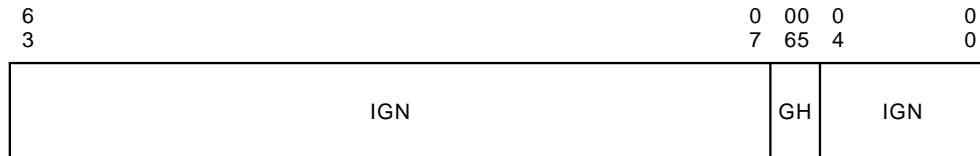
3.3.1 Abox IPRs

Section 3.3.2 through Section 3.3.16 describe the registers contained within the 21064 processor's A-box unit.

3.3.2 Translation Buffer Control Register (TB_CTL)

The granularity hint (GH) field selects between the 21064 TB page mapping sizes. the 21064 provides two sizes in the ITB and all four sizes in the DTB. When only two sizes are provided, the large-page-select (GH=11(bin)) field selects the largest mapping size (512*8Kbytes) and all other values select the smallest (8Kbyte) size. The GH field affects both reads and writes to the ITB and DTB in the 21064. The TB_CTL register itself is write only.

Figure 22: TB_CTL_FIG



3.3.3 Data Translation Buffer Page Table Entry Register (DTB_PTE)

The DTB_PTE register is a read/write register representing the 32-entry small-page and 4-entry large-page DTB page table entries. The entry to be written is chosen by a not-last-used algorithm implemented in hardware. Writes to the DTB_PTE use the memory format bit positions as described in the *Alpha Architecture Handbook* with the exception that some fields are ignored. In particular the valid bit is not represented in hardware.

To ensure the integrity of the DTBs, the DTB's tag array is updated simultaneously from the internal tag register when the DTB_PTE register is written. Reads of the DTB_PTE require two instructions. First, a read from the DTB_PTE sends the PTE data to the DTB_PTE_TEMP register, then a second instruction reading from the DTB_PTE_TEMP register returns the PTE entry to the register file. Reading or writing the DTB_PTE register increments the TB entry pointer of the DTB indicated by the DTB_CTL IPR which allows reading the entire set of DTB PTE entries.

Figure 23: DTB_PTE

Small Page Format:

[illegible]

Large Page Format:

6	5	5			4	4			1	1	1	1	1	1	1	0	0	00	0	0	0	0	0	0
3	3	2			1	0			6	5	4	3	2	1	0	9	8	75	4	3	2	1	0	0
IGN	PFN[33..22]				IGN				U W E	S W E	E W E	K W E	U R E	S R E	E R E	K R E	I G N	A S M	I G N	F O W	F O R	I G N		

DTB_PTE

3.3.4 Data Translation Buffer Page Table Entry Temporary Register (DTB_PTE_TEMP)

The DTB_PTE_TEMP register is a read-only holding register for DTB_PTE read data. Reads of the DTB_PTE require two instructions to return the data to the register file. The first reads the DTB_PTE register to the DTB_PTE_TEMP register. The second returns the DTB_PTE_TEMP register to the integer register file.

Figure 24: DTB_PTE_TEMP

Small Page Format:

6	3	3	3	1	1	1	1	0	0	0	0	0	0	0	00
3	5	4	3	3	2	1	0	9	8	7	6	5	4	3	20
RAZ		AS M	PFN[33..13]		U R E	S R E	E R E	K R E	U W E	S W E	E W E	K W E	F O W	F O R	R A Z

Large Page Format:

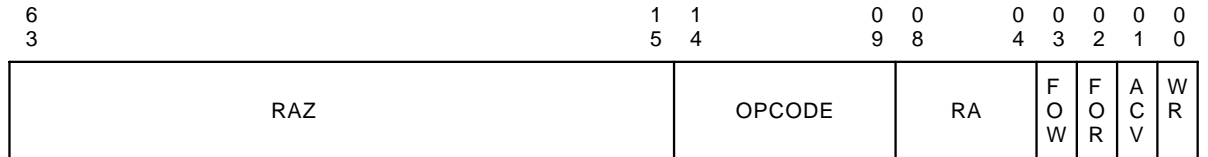
6	3	3	3	2	2	1	1	1	1	0	0	0	0	0	0	00
3	5	4	3	2	2	3	2	1	0	9	8	7	6	5	4	20
RAZ		AS M	PFN[33..22]		I G N	U R E	S R E	E R E	K R E	U W E	S W E	E W E	K W E	F O W	F O R	R A Z

DTB_PTE_TEMP

3.3.5 Memory Management Control and Status Register (MM_CSR)

When D-stream faults occur the information about the fault is latched and saved in the MM_CSR register. The VA and MM_CSR registers are locked against further updates until software reads the VA register. PALcode must explicitly unlock this register whenever its entry point was higher in priority than a DTB miss. MM_CSR bits are only modified by hardware when the register is not locked and a memory management error or a DTB miss occurs. The MM_CSR is unlocked after reset.

Figure 25: MM_CSR



MM_CSR

Table 10: MM_CSR

Field	Description	Type
63:15	Read as Zero	
14:9	OPCODE - Opcode field of the faulting instruction.	RO
8:4	RA - Ra field of the faulting instruction.	RO
3	FOW - Set if reference was a write and the PTE's FOW bit was set.	RO
2	FOR - Set if reference was a read and the PTE's FOR bit was set.	RO
1	ACV - Set if reference caused an access violation.	RO
0	WR - Set if reference which caused error was a write.	RO

3.3.6 Virtual Address Register (VA)

When D-stream faults or DTB misses occur the effective virtual address associated with the fault or miss is latched in the read-only VA register. The VA and MM_CSR registers are locked against further updates until software reads the VA register. The VA IPR is unlocked after reset. PALcode must explicitly unlock this register whenever its entry point was higher in priority than a DTB miss.

3.3.7 Data Translation Buffer Zap Register (DTBZAP)

A write of any value to this IPR invalidates all 32 small-page and four large-page DTB entries. It also resets the NLU pointer to its initial state. The DTBZAP is a pseudo-register.

3.3.8 Data Translation Buffer ASM Register (DTBASM)

A write of any value to this IPR invalidates all 32 small-page and 4 large-page DTB entries in which the ASM bit is equal to zero. The DTBASM is a pseudo-register.

3.3.9 Data Translation Buffer Invalidate Signal Register (DTBIS)

Any write to this pseudo-register will invalidate the DTB entry which maps the virtual address held in the integer register. The integer register is identified by the Rb field of the HW_MTPR instruction, used to perform the write.

3.3.10 Flush Instruction Cache Register (FLUSH_IC)

A write of any value to this pseudo-register flushes the entire instruction cache.

3.3.11 Flush Instruction Cache ASM Register (FLUSH_IC_ASM)

Any write to this pseudo-IPR invalidates all I-Cache blocks in which the ASM bit is clear.

3.3.12 A-box Control Register (ABOX_CTL)

The Abox control register differs between the DECchip 21064 Figure 26 and the DECchip 21064-A275 (Figure 27. The Abox for DECchip 21064-A275 uses bits <12> through <15>, where the Abox for the DECchip 21064 keeps these as undefined.

Figure 26: DECchip 21064 ABOX_CTL

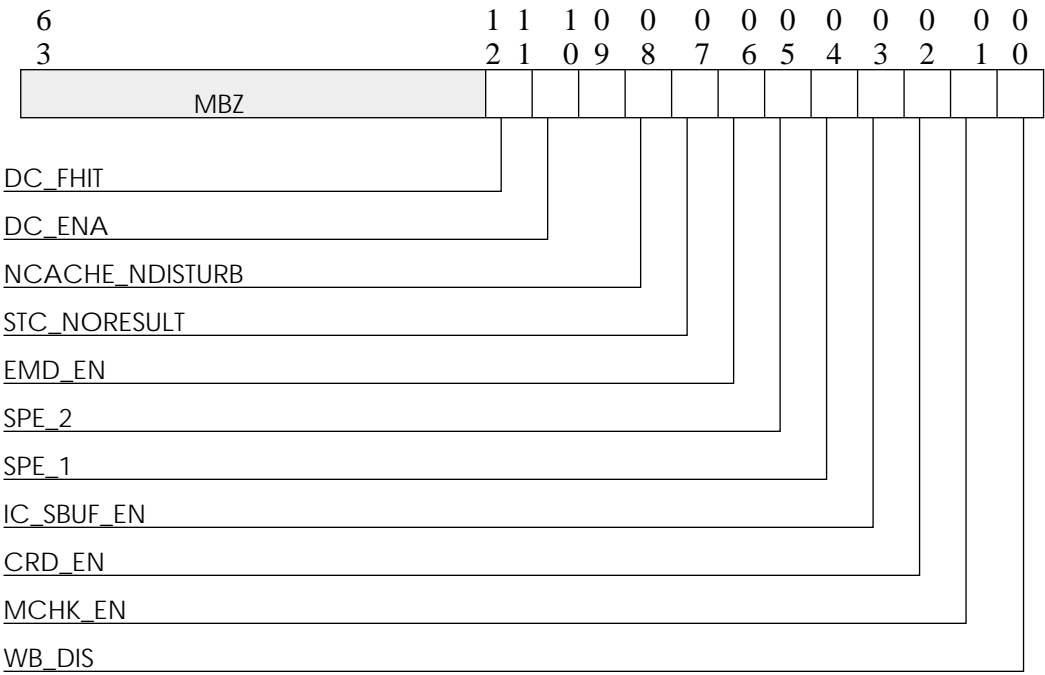


Figure 27: DECchip 21064-A275 ABOX_CTL

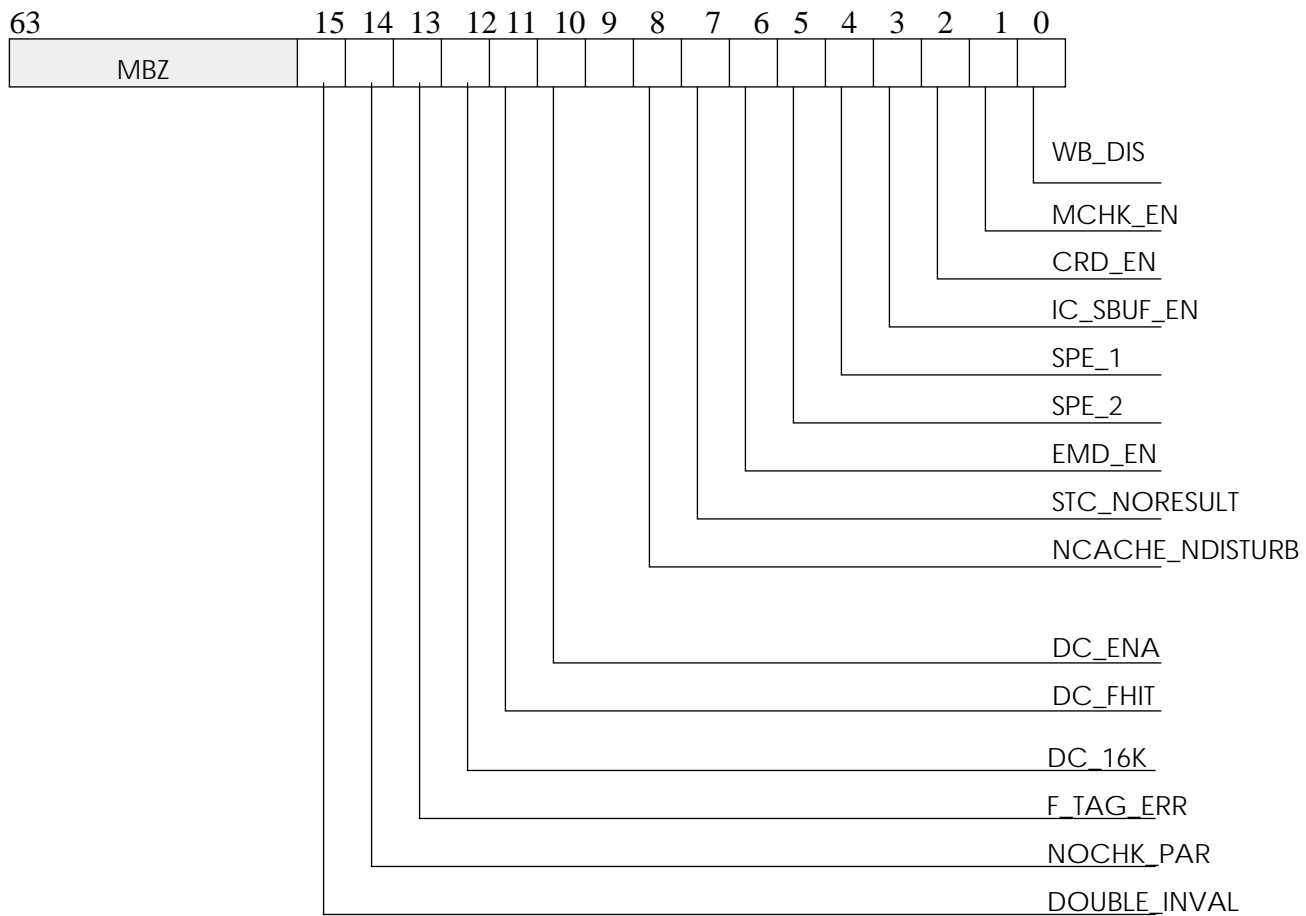
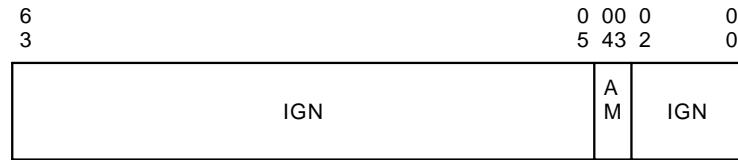


Table 11: Abox Control Register**DECchip 21064-A275 uses bits 12-15**

Field	Description
15	DOUBLE_INVAL - When set, dInvReq_h<0> assertions invalidate both DCache blocks addressed by iAdr_h<12:5>. Cleared by reset.
14	NOCHK_PAR - Set to disable checking of primary cache parity. Cleared by reset.
13	F_TAG_ERR - Set to generate bad primary cache tag parity on fill. Cleared by reset.
12	DC_16K - Set to select 16 KB Dcache, clear to select 8 KB Dcache. Cleared by reset.
10	DC_EN (WO,0) D-Cache enable. When clear, this bit disables and flushes the D-Cache. When set, this bit enables the D-Cache.
11	DC_FHIT (WO,0) D-Cache force hit. When set, this bit forces all D-stream references to hit in the D-Cache. This bit takes precedence over DC_EN, for example, when DC_FHIT is set and DC_EN is clear all D-stream references hit in the D-Cache.
8	NCACHE_NDISTURB (WO,0) When set, any reference to non-cacheable memory space will not cause an arbitrary invalidation of a primary D-Cache location. This should be set.
7	STC_NORESULT (WO,0) Must be cleared
6	EMD_EN (WO,0) Limited hardware support is provided for big endian data formats by way of bit <6> of the ABOX_CTL register. When set, this bit, inverts physical address bit <2> for all D-stream references. It is intended that chip endian mode be selected during initialization PALcode only.
5	SPE_2 (WO,0) This bit, when set, enables one-to-one super page mapping of D-stream virtual addresses with VA<33:13>, if virtual address bits VA<42:41> = 2. Virtual address bits VA<40:34> are ignored in this translation. Access is only allowed in kernel mode.
4	SPE_1 (WO,0) This bit, when set, enables one-to-one super page mapping of D-stream virtual addresses with VA<42:30> = 1FFE(Hex) to physical addresses with PA<33:30> = 0(Hex). Access is only allowed in kernel mode.
3	IC_SBUF_EN (WO,0) I-Cache stream buffer enable. When set, this bit enables operation of a single entry I-Cache stream buffer.
2	CRD_EN - the 21064 only (WO,0) Corrected read data interrupt enable. When this bit is set the Abox generates an interrupt request whenever a pin bus transaction is terminated with a cAck_h code of SOFT_ERROR.
1	MCHK_EN (WO,0) Machine Check Enable. When this bit is set the Abox generates a machine check when errors (which are not correctable by the hardware) are encountered. When this bit is cleared, uncorrectable errors do not cause a machine check, but the BIU_STAT, DC_STAT, BIU_ADDR, FILL_ADDR and DC_ADDR registers are updated and locked when the errors occur.
0	WB_DIS (WO,0) Write Buffer unload Disable. When set, this bit prevents the write buffer from sending write data to the BIU. It should be set for diagnostics only.

3.3.13 Alternative Processor Mode Register (ALT_MODE)

ALT_MODE is a write-only IPR. The AM field specifies the alternate processor mode used by HW_LD and HW_ST instructions which have their ALT bit (bit 14) set.

Figure 28: ALT_MODE**Table 12: ALT Mode**

ALT_MODE[4..3]	Mode
0 0	Kernel
0 1	Executive
1 0	Supervisor
1 1	User

3.3.14 Cycle Counter Register (CC)

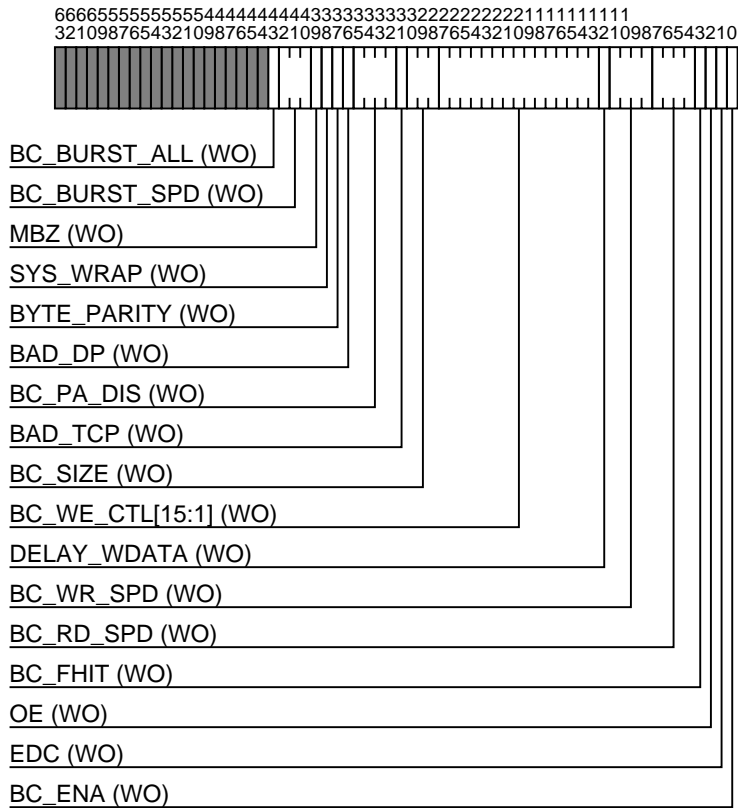
The 21064 supports a cycle counter as described in the *Alpha Architecture Handbook*. This counter, when enabled, increments once each CPU cycle. HW_MTPR Rn,CC writes CC[63..32] with the value held in Rn[63..32], and CC[31..0] are not changed. This register is read by the RPCC instruction defined by the Alpha AXP Architecture.

3.3.15 Cycle Counter Control Register (CC_CTL)

HW_MTPR Rn,CC_CTL writes CC[31..0] with the value held in Rn[31..0], and CC[63..32] are not changed. CC[3..0] must be written with zero. If Rn[32] is set then the counter is enabled, otherwise the counter is disabled. CC_CTL is a write-only IPR.

3.3.16 Bus Interface Unit Control Register (BIU_CTL)

The following figures and tables show the bus interface unit control register format and field descriptions.

Figure 29: BIU Control Register (BIU_CTL)**Table 13: BIU Control Register Description**

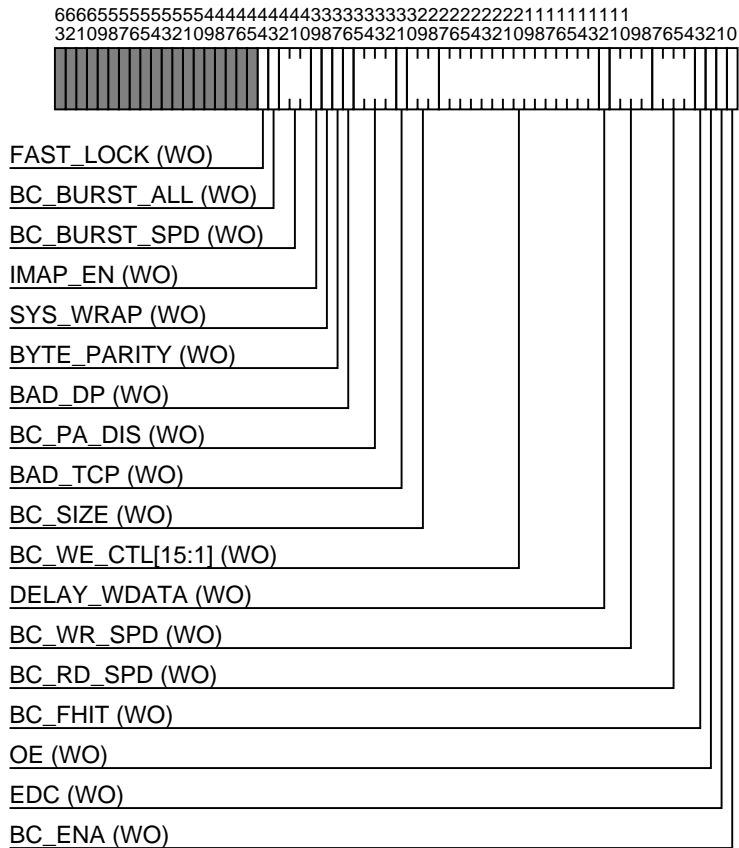
Field	Description
43	BC_BURST_ALL [<i>write-only</i>]
42:40	BC_BURST_SPD [<i>write-only</i>]
39	MBZ [<i>write-only</i>]
38	SYS_WRAP [<i>write-only</i>]
37	BYTE_PARITY [<i>write-only</i>] If set when BIU_CTL<ECC> is cleared, external byte parity id selected, When BIU_CTL<ECC> us set, this bit is ignored, BYTE_ARITY is cleared by reset.
36	BAD_DP [<i>write-only</i>] Bad Data Parity - When set, BAD_DP causes the 21064 CPU to invert the value placed on bits <0>, <7>, <14> and <21> of the check_h <27:0> field during off-chip writes. This produces bad parity when the 21064 CPU is in parity mode, and bad check bit codes when the CPU is in EDC mode.
35:32	BC_PA_DIS [<i>write-only</i>]

Table 13 (Cont.): BIU Control Register Description

Field	Description
	<p>Backup cache physical address disable - This 4-bit field may be used to prevent the CPU chip from using the external cache to service reads and writes based on the quadrant of physical address space which they reference. Table 16 shows the correspondence between this bit field and the physical address space.</p> <p>When a read or write reference is presented to the bus interface unit (BIU), the values of BC_PA_DIS, BC_ENA, and physical address bits <33:32> together determine whether or not to try using the external cache to satisfy the reference. If the external cache is not to be used for a given reference, the bus interface unit does not probe the tag store and makes the appropriate system request immediately. The value of BC_PA_DIS has no impact on which portions of the physical address space may be cached in the primary caches. System components control this through the RDACK field of the pin bus. BC_PA_DIS is not initialized by a reset.</p>
31	<p>BAD_TCP [<i>write-only</i>]</p> <p>BAD Tag Control Parity - When set, BAD_TCP causes the 21064 CPU to write bad parity into the tag control RAM whenever it does a fast external RAM write.</p>
30:28	<p>BC_SIZE [<i>write-only</i>]</p> <p>Backup Cache Size - This field is used to indicate the size of the external cache. BC_SIZE is not initialized by a reset and must be explicitly written before enabling the backup cache. (See Table 15 for the encodings.)</p>
27:13	<p>BC_WE_CTL[15:1] [<i>write-only</i>]</p> <p>Backup Cache Write Enable Control. This field controls the timing of the write enable and chip enable pins during writes into the data and tag control RAMs. It consists of 15 bits, where each bit determines the value placed on the write enable and chip enable pins during a given CPU cycle of the RAM write access.</p> <p>When a given bit of BC_WE_CTL is set, the write enable and chip enable pins are asserted during the corresponding CPU cycle of the RAM access. BC_WE_CTL<0> (bit 13 in BIU_CTL) corresponds to the second cycle of the write access, BC_WE_CTL<1> (bit 14 in BIU_CTL) to the third CPU cycle, and so on. The write enable pins are never asserted in the first CPU cycle of a RAM write access.</p> <p>Unused bits in the BC_WE_CTL field must be written with 0s.</p> <p>BC_WE_CTL is not initialized on reset and must be explicitly written before enabling the external backup cache.</p>
12	<p>DELAY_WDATA [<i>write-only</i>]</p> <p>DELAY_DATA</p>
11:8	<p>BC_WR_SPD [<i>write-only</i>]</p> <p>Backup cache write speed. This field indicates to the bus interface unit the write cycle time of the RAMs used to implement the off-chip external cache, (Backup cache on the CPU module), measured in CPU cycles. It should be written with a value equal to one less the write cycle time of the external cache RAMs.</p> <p>Access times for writes must be in the range 16..2 CPU cycles, which means the values for the BC_WR_SPD field are in the range of 15..1.</p> <p>BC_WR_SPD is not initialized on reset and must be explicitly written before enabling the external cache.</p>
7:4	<p>BC_RD_SPD [<i>write-only</i>]</p>

Table 13 (Cont.): BIU Control Register Description

Field	Description
	<p>Backup (external) cache read speed. This field indicates to the bus interface unit the read access time of the RAMs used to implement the off-chip external cache, measured in CPU cycles. This field should be written with a value equal to one less the read access time of the external cache RAMs.</p> <p>Access times for reads must be in the range 16..3 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..2.</p> <p>BC_RD_SPD are not initialized on reset and must be explicitly written before enabling the external cache.</p>
3	<p>BC_FHIT [<i>write-only</i>]</p> <p>Backup cache force hit. (This cache is external to the 21064 chip.) When this bit and BC_EN are set, all pin bus READ_BLOCK and WRITE_BLOCK transactions are forced to hit in the backup cache. Tag and tag control parity are ignored when the BIU operates in this mode. BC_EN takes precedence over BC_FHIT. When BC_EN is clear and BC_FHIT is set, no tag probes occur and external requests are directed to the CREQ_H pins.</p> <p>Note that the BC_PA_DIS field takes precedence over the BC_FHIT bit.</p>
2	<p>OE [<i>write-only</i>]</p> <p>Output enable - When this bit is set, the 21064 CHP chip does not assert its chip enable pins during RAM write cycles, thus enabling these pins to be connected to the output enable pins of the cache RAMs.</p>
1	<p>EDC [<i>write-only</i>]</p> <p>Error detection and correction. When this bit is set, the 21064 CPU chip generates/expects EDC on the CHECK_H pins. When this bit is clear the CPU chip generates/expects parity on four of the CHECK_H pins.</p>
0	<p>BC_ENA [<i>write-only</i>]</p> <p>External cache enable. When clear, this bit disables the external cache. When the external cache is disabled, the BIU does not probe the external cache tag store for read and write references; it initiates a request on CREQ_H immediately.</p>

Figure 30: DECchip 21064-A275 BIU Control Register (EV45_BIU_CTL)**Table 14: DECchip 21064-A275 BIU Control Register Description**

Field	Description
44	FAST_LOCK [<i>write-only</i>] When set, fast_lock mode operation is selected. This mode can be used only when BIU_CTL<2> is also set, indicating that OE-mode Bcache RAMs are used. Cleared by reset.
43	BC_BURST_ALL [<i>write-only</i>]
42:40	BC_BURST_SPD [<i>write-only</i>]
39	IMAP_EN [<i>write-only</i>] Set to allow dMapWe_h<1:0> to assert for I-stream backup cache reads. Cleared by reset.
38	SYS_WRAP [<i>write-only</i>]
37	BYTE_PARITY [<i>write-only</i>] If set when BIU_CTL<ECC> is cleared, external byte parity id selected, When BIU_CTL<ECC> is set, this bit is ignored, BYTE_ARITY is cleared by reset.
36	BAD_DP [<i>write-only</i>]

Table 14 (Cont.): DECchip 21064-A275 BIU Control Register Description

Field	Description
	Bad Data Parity - When set, BAD_DP causes the 21064 CPU to invert the value placed on bits <0>,<7>,<14> and <21> of the check_h<27:0> field during off-chip writes. This produces bad parity when the 21064 CPU is in parity mode, and bad check bit codes when the CPU is in EDC mode.
35:32	<p>BC_PA_DIS [<i>write-only</i>]</p> <p>Backup cache physical address disable - This 4-bit field may be used to prevent the CPU chip from using the external cache to service reads and writes based on the quadrant of physical address space which they reference. Table 16 shows the correspondence between this bit field and the physical address space.</p> <p>When a read or write reference is presented to the bus interface unit (BIU), the values of BC_PA_DIS, BC_ENA, and physical address bits <33:32> together determine whether or not to try using the external cache to satisfy the reference. If the external cache is not to be used for a given reference, the bus interface unit does not probe the tag store and makes the appropriate system request immediately.</p> <p>The value of BC_PA_DIS has no impact on which portions of the physical address space may be cached in the primary caches. System components control this through the RDACK field of the pin bus.</p> <p>BC_PA_DIS is not initialized by a reset.</p>
31	<p>BAD_TCP [<i>write-only</i>]</p> <p>BAD Tag Control Parity - When set, BAD_TCP causes the 21064 CPU to write bad parity into the tag control RAM whenever it does a fast external RAM write.</p>
30:28	<p>BC_SIZE [<i>write-only</i>]</p> <p>Backup Cache Size - This field is used to indicate the size of the external cache. BC_SIZE is not initialized by a reset and must be explicitly written before enabling the backup cache. (See Table 15 for the encodings.)</p>
27:13	<p>BC_WE_CTL[15:1] [<i>write-only</i>]</p> <p>Backup Cache Write Enable Control. This field controls the timing of the write enable and chip enable pins during writes into the data and tag control RAMs. It consists of 15 bits, where each bit determines the value placed on the write enable and chip enable pins during a given CPU cycle of the RAM write access.</p> <p>When a given bit of BC_WE_CTL is set, the write enable and chip enable pins are asserted during the corresponding CPU cycle of the RAM access. BC_WE_CTL<0> (bit 13 in BIU_CTL) corresponds to the second cycle of the write access, BC_WE_CTL<1> (bit 14 in BIU_CTL) to the third CPU cycle, and so on. The write enable pins are never asserted in the first CPU cycle of a RAM write access.</p> <p>Unused bits in the BC_WE_CTL field must be written with 0s.</p> <p>BC_WE_CTL is not initialized on reset and must be explicitly written before enabling the external backup cache.</p>
12	<p>DELAY_WDATA [<i>write-only</i>]</p> <p>DELAY_DATA</p>
11:8	BC_WR_SPD [<i>write-only</i>]

Table 14 (Cont.): DECchip 21064-A275 BIU Control Register Description

Field	Description
	<p>Backup cache write speed. This field indicates to the bus interface unit the write cycle time of the RAMs used to implement the off-chip external cache, (Backup cache on the CPU module), measured in CPU cycles. It should be written with a value equal to one less the write cycle time of the external cache RAMs.</p> <p>Access times for writes must be in the range 16..2 CPU cycles, which means the values for the BC_WR_SPD field are in the range of 15..1.</p> <p>BC_WR_SPD is not initialized on reset and must be explicitly written before enabling the external cache.</p>
7:4	<p>BC_RD_SPD [<i>write-only</i>]</p> <p>Backup (external) cache read speed. This field indicates to the bus interface unit the read access time of the RAMs used to implement the off-chip external cache, measured in CPU cycles. This field should be written with a value equal to one less the read access time of the external cache RAMs.</p> <p>Access times for reads must be in the range 16..3 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..2.</p> <p>BC_RD_SPD are not initialized on reset and must be explicitly written before enabling the external cache.</p>
3	<p>BC_FHIT [<i>write-only</i>]</p> <p>Backup cache force hit. (This cache is external to the 21064 chip.) When this bit and BC_EN are set, all pin bus READ_BLOCK and WRITE_BLOCK transactions are forced to hit in the backup cache. Tag and tag control parity are ignored when the BIU operates in this mode. BC_EN takes precedence over BC_FHIT. When BC_EN is clear and BC_FHIT is set, no tag probes occur and external requests are directed to the CREQ_H pins.</p> <p>Note that the BC_PA_DIS field takes precedence over the BC_FHIT bit.</p>
2	<p>OE [<i>write-only</i>]</p> <p>Output enable - When this bit is set, the 21064 CHP chip does not assert its chip enable pins during RAM write cycles, thus enabling these pins to be connected to the output enable pins of the cache RAMs.</p>
1	<p>EDC [<i>write-only</i>]</p> <p>Error detection and correction. When this bit is set, the 21064 CPU chip generates/expects EDC on the CHECK_H pins. When this bit is clear the CPU chip generates/expects parity on four of the CHECK_H pins.</p>
0	<p>BC_ENA [<i>write-only</i>]</p> <p>External cache enable. When clear, this bit disables the external cache. When the external cache is disabled, the BIU does not probe the external cache tag store for read and write references; it initiates a request on CREQ_H immediately.</p>

Table 15: BC_SIZE

BC_SIZE	Size
0 0 0	128 Kbytes
0 0 1	256 Kbytes
0 1 0	RESERVED
0 1 1	1 Mbytes
1 0 0	2 Mbytes
1 0 1	4 Mbytes
1 1 0	8 Mbytes

Table 16: BC_PA_DIS

BIU_CTL bits	Physical Address
[32]	PA[33..32] = 0
[33]	PA[33..32] = 1
[34]	PA[33..32] = 2
[35]	PA[33..32] = 3

Table 17: BIU_CTL Initialization Values

Module Description	Init Value	Note
B2020-AA	E30006447	1MB 5/5 tick cache, 2 cycle write strobe
B2024-BA	E5000E567	4MB 6/7 tick cache, 3 cycle write strobe

3.3.17 Data Translation Buffer (DTB)

The 21064 contains a 32-entry, fully associative, translation buffer (DTB) that caches recently used data-stream page table entries and supports all four variants of the granularity hint option, as described in the *Alpha Architecture Handbook*.

The 21064 provides an extension referred to as the super page, which can be enabled using ABOX_CTL [5:4]. Super page mappings provide virtual to physical address translation for two regions of the virtual address space. The first enables super page mapping when virtual address bits [42:41] = 2. In this mode, the entire physical address space is mapped multiple times to one quadrant of the virtual address space defined by VA [42:41] = 2. The second super page mode maps a 30-bit region of the total physical address space defined by PA [33:30] = 0 into a single corresponding region of virtual space defined by VA [42:30] = 1FFE(Hex). Super page translation is only allowed in kernel mode. The operating system via PALcode is responsible for ensuring that translation buffer entries, including super page regions, do not map overlapping virtual address regions at the same time.

The 21064 DTB supports a single address space number (ASN) with the PTE[ASM] bit. Each PTE entry in the DTB contains an address space match (ASM) bit. Writes to the DTBASM IPR invalidate all entries that do not have their ASM bit set. This provides a simple method of preserving entries that map operating system regions while invalidating all others.

For load and store instructions, the effective 43-bit virtual address is presented to the DTBs. If the PTE of the supplied virtual address is cached in the DTB, the PFN and protection bits for the page that contains the address are used by the Abox to complete the address translation and access checks.

The DTB is filled and maintained by PALcode. Note that the DTB can be filled in kernel mode by setting the HWE bit in the ICCSR IPR.

3.3.18 Bus Interface Unit (BIU)

The BIU controls the interface to the 21064 pin bus. The BIU responds to three classes of CPU-generated requests:

- Dcache fills
- Icache fills
- Write buffer-sourced commands

The BIU resolves simultaneous internal requests using a fixed priority scheme in which Dcache fill requests are given highest priority, followed by Icache fill requests. Write buffer requests have the lowest priority.

The BIU contains logic to directly access an external cache to service internal cache fill requests and writes from the write buffer. The BIU services reads and writes that do not hit in the external cache with help from external logic.

Internal data transfers between the CPU and the BIU are made through a 64-bit bidirectional bus. Since the internal cache fill block size is 32 bytes, cache fill operations result in four data transfers across this bus from the BIU to the appropriate cache. Also, because each write buffer entry is 32 bytes wide, write transactions may result in four data transfers from the write buffer to the BIU.

3.3.19 Load Silos

The Abox contains a memory reference pipeline that can accept a new load or store instruction every cycle until a Dcache fill is required. Since the Dcache lines are only allocated on load misses, the Abox can accept a new instruction every cycle until a load miss occurs. When a load miss occurs the Ibox stops issuing all instructions that use the load port of the register file or are otherwise handled by the Abox. This includes LDx, STx, HW_MTPR, HW_MFPR, FETCH, FETCH_M, RPCC, RS, RC, and MB. It also includes all memory format branch instructions, JMP, JSR, JSR_COROUTINE, and RET.

However, a JSR with a destination of R31 may be issued.

Because the result of each Dcache lookup is known late in the pipeline (stage [6]) and instructions are issued in pipe stage [3], there can be two instructions in the Abox pipeline behind a load instruction that misses the Dcache. These two instructions are handled as follows:

- Loads that hit the Dcache are allowed to complete - hit under miss.
- Load misses are placed in a silo and replayed in order after the first load miss completes.
- Store instructions are presented to the Dcache at their normal time with respect to the pipeline. They are siloed and presented to the write buffer in order with respect to load misses.

In order to improve performance, the Ibox is allowed to restart the execution of Abox directed instructions before the last pending Dcache fill is complete. Dcache fill transactions result in four data transfers from the BIU to the Dcache. These transfers can each be separated by one or more cycles depending on the characteristics of the external cache and memory subsystems. The BIU attempts to send the quadword of the fill block that the CPU originally requested in the first of these four transfers (it is always able to accomplish this for reads that hit in the external cache). Therefore, the pending load instruction that requested the Dcache fill can complete before the Dcache fill finishes. Dcache fill data accumulates one quadword at a time into a "pending fill" latch, rather than being written into the cache array as it is received from the BIU. When the load miss silo is empty and the requested quadword for the last outstanding load miss is received, the Ibox resumes execution of Abox-directed instructions despite the still-pending Dcache fill. When the entire cache line has been received from the BIU, it is written into the Dcache data array whenever the array is not busy with a load or a store.

3.3.20 Write Buffer

The Abox contains a write buffer for two purposes.

- To minimize the number of CPU stall cycles by providing a high bandwidth (but finite) resource for receiving store data
This is required since the 21064 can generate store data at the peak rate of one quadword every CPU cycle, which is greater than the rate at which the external cache subsystem can accept the data.
- To attempt to aggregate-store data into aligned 32-byte cache blocks to maximize the rate at which data may be written from the 21064 into the external cache

The write-merging operation of the write buffer may result in the order of off-chip writes being different from the order in which their corresponding store instructions were executed. Further, the write buffer may collapse multiple stores to the same location into a single off-chip write transaction. Software that requires strict write ordering, or that multiple stores to the same location result in multiple off-chip write sequences, must insert a memory barrier instruction between the store instructions of interest.

In addition to store instructions, MB, STQ/C, STL/C, FETCH, and FETCH_M instructions are also written into the write buffer and sent off-chip. Unlike stores, however, these write buffer-directed instructions are never merged into a write buffer entry with other instructions.

The write buffer has four entries; each has storage for up to 32 bytes. The buffer has a "head" pointer and "tail" pointer. The buffer puts new commands into empty tail entries and takes commands out of nonempty head entries. The head pointer increments when an entry is unloaded to the BIU, and the tail pointer increments when new data is put into the tail entry. The head and tail pointers only point to the same entry when the buffer has zero or four nonempty entries.

Suppose for a moment that no writes ever merge with existing nonempty entries. In this case the ordering of writes with respect to other writes will be maintained. The write buffer never reorders writes except to merge them into nonempty entries. Once a write merges into a nonempty slot, its "programmed" order is lost with respect to both writes in the same slot and writes in other slots.

The write buffer attempts to send its head entry off-chip by requesting the BIU when one of the following conditions is met:

- The write buffer contains at least two valid entries.
- The write buffer contains one valid entry and at least 256 CPU cycles have elapsed since the execution of the last write buffer-directed instruction.
- The write buffer contains an MB instruction.
- The write buffer contains an STQ/C or STL/C instruction.
- A load miss is pending to an address currently valid in the write buffer that requires the write buffer to be flushed. The write buffer is completely flushed regardless of which entry matches the address.

3.4 Fbox

The Fbox is on-chip, pipelined, and capable of executing both Digital and IEEE floating point instructions. IEEE floating point datatypes S_floating and T_floating are supported with all rounding modes except round to \pm infinity, which can be provided in software. Digital floating point datatypes F_floating and G_floating are fully supported with limited support for D_floating format.

The Fbox contains:

- A 32-entry, 64-bit floating point register file.
- A user-accessible control register, FPCR, containing:
 - Round mode controls
 - Exception flag information

The Fbox can accept an instruction every cycle, with the exception of floating point divide instructions. The latency for data dependent, non-divide instructions is six cycles. For detailed information on instruction timing, refer to Section 3.7.

For divide instructions, the Fbox does not compute the inexact flag. Consequently, the INE exception flag in the FPCR register is never set for IEEE floating-point divide using the inexact enable (/I) qualifier. In order to deliver IEEE conforming exception behavior to the user, 21064 FPU hardware always traps on DIVS/SI and DIVT/SI instructions. The intent is for the arithmetic exception handler in either PALcode or the operating system to identify the source of the trap, compute the inexact flag, and deliver the appropriate exception to the user. The exception associated with DIV/SI and DIVT/SI is imprecise. Software must follow the rules specified by the Alpha AXP architecture associated with the software completion modifier in order to ensure that the trap handler can deliver correct behavior to the user.

For IEEE compliance issues, see Section 3.5 and the *Alpha Architecture Handbook*.

3.5 21064 IEEE Floating Point Conformance

The 21064 supports the IEEE floating-point operations as defined by the Alpha AXP architecture. Support for a complete implementation of the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Standard 754-1985) is provided by a combination of hardware and software as described in the *Alpha Architecture Handbook*.

Additional information that provides guidelines for writing code supporting precise exception handling (necessary for complete conformance to the Standard) is in the *Alpha Architecture Handbook*.

Information specific to the 21064 follows:

- When operating without the /Underflow qualifier, the 21064 replaces underflow results with exact zero even if the correct result would have been negative zero as defined in the IEEE Standard. This is an Alpha AXP architecture value-added behavior for improved performance over either hardware or software denormal handling. When strict IEEE compliance is required, the /Underflow modifier is necessary and the software must provide the correct result (including negative zero).
- The 21064 supports Infinity operands only when used in the CMPT instruction. Other instructions using Infinity operands cause Invalid Operation (INV) arithmetic traps.
- NaN, Denormal, or Infinity (except when used in CMPT) input operands produce Invalid Operation (INV) arithmetic traps when used with arithmetic operation instructions. CPYSE/CPYSN, FCMOV instructions, and MF_FPCR/MT_FPCR are not arithmetic operations, and will pass NaN, Denormal, and Infinity values without initiating arithmetic traps. Input operand traps take precedence over arithmetic result traps.
- The 21064 will not produce a NaN, Denormal, or Infinity result.
- The 21064 supports IEEE normal and chopped rounding modes in hardware. Instructions designating plus infinity and minus infinity rounding modes cause precise exceptions to the OPCDEC PAL entry point. This implies that the EXC_

ADDR register will be loaded with the address of the faulting instruction and all following instructions will be aborted.

- The DIVS and DIVT with /Inexact modifier instructions report an Inexact (INE) arithmetic trap on all results of operations that do not involve NaN, Infinity, or Denormal input operands. Operations using NaN, Infinity, and Denormal input operands generate Invalid Operation (INV) arithmetic traps.
- Floating-point exceptions generated by the 21064 are recorded in two places.
 - The FPCR register, as defined in the Alpha AXP architecture and accessible by the MT/MF_FPCR instructions, records the occurrence of all exceptions that are detected (except SWC), whether or not the corresponding trap is enabled (through the instruction modifiers). This register can only be cleared through an explicit clear command (MT_FPCR) so that the exception information it records is a summary of all exceptions that have occurred since the last clear.
 - In addition, if an exception is detected and the corresponding trap is enabled, the 21064 will record the condition in the EXC_SUM register and initiate an arithmetic trap. As a special case, in order to support Inexact exception behavior with the DIVS/I and DIVT/I instructions, the FPCR will not record an Inexact exception, although the 21064 will always set the INE bit in the EXC_SUM register during these instructions. This behavior allows software emulation of the division instruction with accurate reporting of potential Inexact exceptions.

3.6 Cache Organization

The 21064 includes two on-chip caches, a data cache (Dcache) and an instruction cache (Icache). All memory cells in both Dcache and Icache are fully static six transistor CMOS structures.

3.6.1 Data Cache (Dcache)

The DECchip DECchip 21064 Dcache contains an 8 KB (16 KB for the DECchip 21064-A275) data cache. It is a write-through, direct mapped, read allocate physical cache and has 32-byte blocks. System components can keep the Dcache coherent with memory by using the invalidate bus described in the pin bus section of this specification.

3.6.2 Instruction Cache (Icache)

The DECchip 21064 Icache is an 8-KB (16-KB for the DECchip 21064-A275), physical direct-mapped cache. Icache blocks, or lines, contain 32-bytes of instruction stream data with associated tag, as well as a six-bit ASN field, a one-bit ASM field, and an eight-bit branch history field per block. It does not contain hardware for maintaining coherency with memory and is unaffected by the invalidate bus.

The chip also contains a single-entry Icache stream buffer that, together with its supporting logic, reduces the performance penalty due to Icache misses incurred during in-line instruction processing. Stream buffer prefetch requests never cross physical page boundaries, but instead wrap around to the first block of the current page.

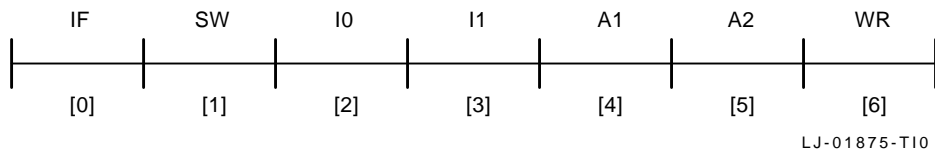
3.7 Pipeline Organization

The 21064 has a seven-stage pipeline for integer operate and memory reference instructions. Floating point operate instructions progress through a ten stage pipeline. The Ibox maintains state for all pipeline stages to track outstanding register writes, and determine Icache hit/miss.

The following figures show the integer operate, memory reference, and the floating point operate pipelines for the Ibox, Ebox, Abox, and Fbox. The first four cycles are executed in the Ibox and the last stages are box specific. There are bypassers in all of the boxes that allow the results of one instruction to be used as operands of a following instruction without having to be written to the register file. Section 3.8 describes the pipeline scheduling rules.

Figure 31 shows the Integer Operate Pipeline.

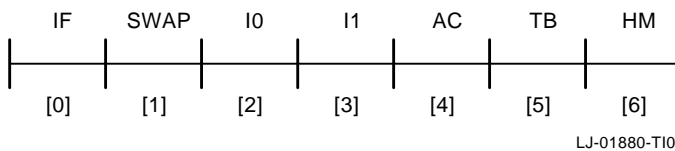
Figure 31: Integer Operate Pipeline



- IF: Instruction Fetch
- SW: Swap Dual Issue Instruction / Branch Prediction
- I0: Decode
- I1: Register file(s) access / Issue check
- A1: Computation cycle 1 / Ibox computes new PC
- A2: Computation cycle 2 / ITB look-up
- WR: Integer register file write / Icache Hit/Miss

Figure 32 shows the Memory Reference Pipeline.

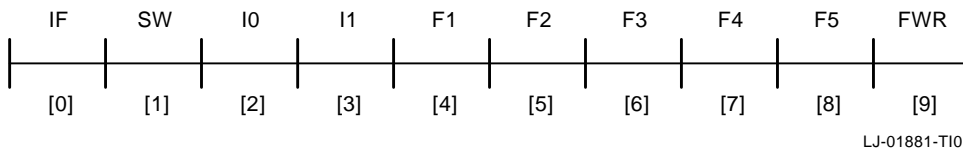
Figure 32: Memory Reference Pipeline



- AC: Abox calculates the effective D-stream address
- TB: DTB look-up
- HM: Dcache Hit/Miss and load data register file write

Figure 33 shows the Floating Point Operate Pipeline.

Figure 33: Floating Point Operate Pipeline



- F1-F5: Floating point calculate pipeline
- FWR: Floating point register file write

3.7.1 Static and Dynamic Stages

The 21064 integer pipeline divides instruction processing into four static and three dynamic stages of execution. The 21064 floating point pipeline maintains the first four static stages and adds six dynamic stages of execution. The first four stages consist of:

- Instruction fetch
- Swap
- Decode
- Issue logic

These stages are static because instructions can remain valid in the same pipeline stage for multiple cycles while waiting for a resource, or stalling for other reasons.

Dynamic stages always advance state and are unaffected by any stall in the pipeline. (Pipeline stalls are also referred to as pipeline freezes.) A pipeline freeze may occur while zero instructions issue, or while one instruction of a pair issues and the second is held at the issue stage. A pipeline freeze implies that a valid instruction or instructions are presented to be issued but cannot proceed.

Upon satisfying all issue requirements, instructions are allowed to continue through any pipeline toward completion. Instructions cannot be held in a given pipe stage after they are issued. It is up to the issue stage to ensure that all resource conflicts are resolved before an instruction is allowed to continue. The only means of stopping instructions after the issue stage is a chip-internal abort condition.

3.7.2 Aborts

Aborts can result from a number of causes. In general, they are grouped into two classes:

- Exceptions (including interrupts)
- Non-exceptions

There is one basic difference between the two classes: exceptions require that the pipeline be drained of all outstanding instructions before restarting the pipeline at a redirected address. In both exceptions and non-exceptions, the pipeline must be flushed of all instructions that were fetched after the instruction that caused the abort condition. This includes stopping one instruction of a dual-issued pair in the case of an abort condition on the first instruction of the pair.

The non-exception case, however, does not need to drain the pipeline of all outstanding instructions ahead of the aborting instruction. The pipeline can be immediately restarted at a redirected address. Examples of non-exception abort conditions are branch mispredictions, subroutine call/return mispredictions, and instruction cache misses. Data cache misses do not produce abort conditions but can cause pipeline freezes.

If an exception occurs, the processor aborts all instructions issued after the excepting instruction as described. Due to the nature of some error conditions, this can occur as late as the write cycle. Next, the address of the excepting instruction is latched in the EXC_ADDR IPR. When the pipeline is fully drained, the processor begins instruction execution at the address given by the PALcode dispatch. The pipeline is drained when:

- All outstanding writes to both the integer and floating point register file have completed and arithmetic traps have been reported.
- All outstanding instructions have successfully completed memory management and access protection traps.

3.7.3 Non-Issue Conditions

There are two basic reasons for non-issue conditions.

- A pipeline freeze when a valid instruction or pair of instructions are prepared to issue but cannot due to a resource conflict

This type of non-issue cycle can be minimized through code scheduling.

- Pipeline bubbles when there is no valid instruction in the pipeline to issue

Pipeline bubbles exist due to abort conditions as described in Section 3.7.2. In addition, a single pipeline bubble is produced whenever a branch-type instruction is predicted to be taken, including subroutine calls and returns. Pipeline bubbles are reduced directly by the hardware through bubble squashing, but can also be effectively minimized through careful coding practices. Bubble squashing involves the ability of the first four pipeline stages to advance whenever a bubble is detected in the pipeline stage immediately ahead of it while the pipeline is otherwise frozen.

3.8 Scheduling and Issuing Rules

Scheduling and issuing rules are covered in the sections that follow.

3.8.1 Instruction Class Definition

The scheduling and dual issue rules covered in this section are only performance related. There are no functional dependencies related to scheduling or dual issuing. The scheduling and issuing rules are defined in terms of instruction classes. Table 18 specifies all of the instruction classes and the box that executes the particular class.

Table 18: Producer-Consumer Classes

Class Name	Box	Instruction List
LD	Abox	All loads, (HW_MFPR, RPCC, RS, RC, STC producers only), (FETCH consumer only)
ST	Abox	All stores, HW_MTPR
IBR	Ebox	Integer conditional branches
FBR	Fbox	Floating point conditional branches
JSR	Ebox	Jump to subroutine instructions JMP, JSR, RET, or JSR_COROUTINE, (BSR, BR producer only)
IADDLOG	Ebox	ADDL ADDL/V ADDQ ADDQ/V SUBL SUBL/V SUBQ SUBQ/V S4ADDL S4ADDQ S8ADDL S8ADDQ S4SUBL S4SUBQ S8SUBL S8SUBQ LDA LDAH AND BIS XOR BIC ORNOT EQV
SHIFTCM	Ebox	SLL SRL SRA EXTQL EXTLL EXTWL EXTBL EXTQH EXTLH EXTWH MSKQL MSKLL MSKWL MSKBL MSKQH MSKLH MSKWH INSQL INSL INSWL INSB INSQL INSLH INSWH ZAP ZAPNOT CMOVEQ CMOVNE CMOVL CMOVLB CMOVLBS CMOVLBC
ICMP	Ebox	CMPEQ CMPLT CMPL CMPULT CMPULE CMPBGE
IMULL	Ebox	MULL MULL/V
IMULQ	Ebox	MULQ MULQ/V UMULH
FPOP	Fbox	Floating point operates except divide
FDIV	Fbox	Floating point divide

3.8.2 Producer-Consumer Latency

The 21064 enforces the following issue rules regarding producer-consumer latencies.

The scheduling rules are described as a producer-consumer matrix, shown in Figure 34. Each row and column in the matrix is a class of Alpha AXP instructions. A “1” in the Producer-Consumer Latency Matrix indicates one cycle of latency. A one cycle latency means that if instruction B uses the results of instruction A, then instruction B can be issued *one* cycle after instruction A is issued.

When determining latency for a given instruction sequence, first identify the classes of each instruction. The following example lists the classes in the comment field:

```
ADDQ    R1, R2, R3      ! IADDLOG class
SRA     R3, R4, R5      ! SHIFT class
SUBQ    R5, R6, R7      ! IADDLOG class
STQ     R7, D(R10)      ! ST class
```

The SRA instruction consumes the result (R3) produced by the ADDQ instruction. The latency associated with an iadd-shift producer-consumer pair as specified by the matrix is one. That means that if the ADDQ was issued in cycle “n”, the SRA could be issued in cycle “n+1.”

The SUBQ instruction consumes the result (R5) produced by the SRA instruction. The latency associated with a shift-iadd producer-consumer pair, as specified by the matrix, is two. That means that if the SRA was issued in cycle “n”, the SUBQ could be issued in cycle “n+2.” The Ibox injects one nop cycle in the pipeline for this case.

The final case has the STQ instruction consuming the result (R7) produced by the SUBQ instruction. The latency associated with an iadd-st producer-consumer pair, when the result of the iadd is the store data, is zero. This means that the SUBQ and STQ instruction pair can be dual-issued.

Figure 34: Producer-Consumer Latency Matrix

<div> <div>Producer</div> <div>Consumer</div> </div>	LD (1)	JSR	IADDLOG	SHIFTCM	ICMP	IMULL (3)	IMULQ (3)	FPOP	FDIV F/S (4)	F
LD	3	3	2	2	2	21	23	X	X	
ST (2)	3	3	2/0	2/0	2/0	21/20	23/22	A/4	A/32	
IBR	3	3	1	2	1	21	23	X	X	
JSR	3	3	2	2	2	21	23	X	X	
IADDLOG	3	3	1	2	2	21	23	X	X	
SHIFTCM	3	3	1	2	2	21	23	X	X	
ICMP	3	3	1	2	2	21	23	X	X	
IMUL	3	3	1	2	2	21/19	23/21	X	X	
FBR	3	X	X	X	X	X	X	6	34	
FPOP	3	X	X	X	X	X	X	6	34	
FDIV	3	X	X	X	X	X	X	6	34/30	6

LJ-C

Notes:

- X indicates an impossible state, or a state not encountered under normal circumstances. For example, a floating point branch would not follow an integer compare.
- For loads, Dcache hit is assumed. The latency for a Dcache miss is dependent on the system configuration.
- For some producer classes, two latencies, X/Y, are given with the ST consumer class. The X represents the latency for base address of store and the Y represents the latency for store data. Floating point results cannot be used as the base address for load or store operations.
- For IMUL followed by IMUL, two latencies are given. The first represents the latency with data dependency; in other words, the second IMUL uses the result from the first. The second is the multiply latency without data dependencies.

- For FDIV followed by FDIV, there are two latencies given. The first represents the latency with data dependency; the second FDIV uses the result from the first. The second is the division latency without data dependencies.

3.8.3 Producer-Producer Latency

Producer-producer latency, also known as write-after write-conflicts, are restricted only by the register write order. For most instructions, this is dictated by issue order; however, IMUL, FDIV, and LD instructions may require more time than other instructions to complete and, therefore, must stall following instructions that write the same destination register to preserve write ordering. In general, only cases involving an intervening producer-consumer conflict are of interest. They can occur commonly in a dual issue situation when a register is reused. In these cases, producer-consumer latencies are equal to or greater than the required producer-producer latency as determined by write ordering and therefore dictate the overall latency. An example of this case is shown in the code:

```
LDQ R2,D(R0) ; R2 destination
ADDQ R2,R3,R4 ; wr-rd conflict stalls execution waiting for R2
LDQ R2,D(R1) ; wr-wr conflict may dual issue when addq issues
```

3.8.4 Instruction Issue Rules

The following conditions prevent instruction issue:

- No instruction can be issued until all of its source and destination registers are clean; in other words, all outstanding writes to the destination register are guaranteed to complete in issue order and there are no outstanding writes to the source registers or those writes can be bypassed.
- No LD, ST, FETCH, MB, RPCC, RS, RC, TRAPB, HW_MXPR, or BSR, BR, JSR (with destination other than R31) can be issued after an MB instruction until the MB has been acknowledged on the external pin bus.
- No IMUL instructions can be issued if the integer multiplier is busy.
- No SHIFT, IADDLOG, ICMP or ICMOV instruction can be issued exactly three cycles before an integer multiplication completes.
- No integer or floating point conditional branch instruction can be issued in the cycle immediately following a JSR, JMP, RET, JSR_COROUTINE, or HW_REI instruction.
- No TRAPB instruction can be issued as the second instruction of a dual issue pair.
- No LD instructions can be issued in the two cycles immediately following an STC.
- No LD, ST, FETCH, MB, RPCC, RS, RC, TRAPB, HW_MXPR or BSR, BR, JSR (with destination other than R31) instruction can be issued when the Abox is busy due to a load miss or write buffer overflow. For more information see Section 3.3.19.
- No FDIV instruction can be issued if the floating pointer divider is busy.

- No floating point operate instruction can be issued exactly five or exactly six cycles before a floating point divide completes.

3.8.5 Dual Issue Table

Table 19 can be used to determine instruction pairs that can issue in a single cycle. Instructions are dispatched using two internal data paths or buses. For more information about instructions and their opcodes and definitions, refer to the *Alpha Architecture Handbook*.

The buses are referred to in Table 19 as IB0, IB1, and IBx.

Any instruction identified with IB0 in the table can be issued in the same cycle as any instruction identified with IB1. An instruction that is identified as IBx may be issued with either IB0 or IB1.

Dual issue is attempted if the input operands are available as defined by the Producer-Consumer Latency Matrix (Figure 34) and the following requirements are met:

- Two instructions must be contained within an aligned quadword.
- The instructions must not both be in the group labeled as IB0.
- The instructions must not both be in the group labeled as IB1.
- No more than one of JSR, integer conditional branch, BSR, HW_REI, BR, or floating-point branch can be issued in the same cycle.
- No more than one of load, store, HW_MTPR, HW_MFPR, MISC, TRAPB, HW_REI, BSR, BR, OR JSR can be issued in the same cycle.

NOTE

Producer-Consumer latencies of zero indicate that dependent operations between these two instruction classes can dual issue. For example, ADDQ R1, R2, R3 STQ R3, D(R4).

Table 19: Opcode Summary (with Instruction Issue Bus)

	00	08	10	18	20	28	30	38
0/8	PAL*	LDA	INTA*	MISC*	LDF	LDL	BR	BLBC
	IB1	IB0	IB0	IB1	IBx	IBx	IB1	IB1
1/9	Res	LDAH	INTL*	HW_MFPR	LDG	LDQ	FBEQ	BEQ
	IB1	IB0	IB0	IB1	IBx	IBx	IB0	IB1
2/A	Res	Res	INTS*	JSR*	LDS	LDL_L	FBLT	BLT
	IB1	IB1	IB0	IB1	IBx	IBx	IB0	IB1
3/B	Res	LDQ_U	INTM*	HW_LD	LDT	LDQ_L	FBLE	BLE
	IB1	IBx	IB0	IB1	IBx	IBx	IB0	IB1
4/C	Res	Res	Res	Res	STF	STL	BSR	BLBS
	IB1	IB1	IB1	IB1	IB0	IB1	IB1	IB1
5/D	Res	Res	FLTV*	HW_MTPR	STG	STQ	FBNE	BNE
	IB1	IB1	IB1	IB1	IB0	IB1	IB0	IB1
6/E	Res	Res	FLTI*	HW_REI	STS	STL_C	FBGE	BGE
	IB1	IB1	IB1	IB1	IB0	IB1	IB0	IB1
7/F	Res	STQ_U	FLTL*	HW_ST	STT	STQ_C	FBGT	BGT
	IB1	IB1	IB1	IB1	IB0	IB1	IB0	IB1

Key to Opcode Summary

FLTI*—IEEE floating-point instruction opcodes
 FLTL*—Floating-point operate instruction opcodes
 FLTV*—VAX floating-point instruction opcodes
 INTA*—Integer arithmetic instruction opcodes
 INTL*—Integer logical instruction opcodes
 INTM*—Integer multiply instruction opcodes
 INTS*—Integer subtract instruction opcodes
 JSR*—Jump instruction opcodes
 MISC*—Miscellaneous instruction opcodes
 PAL*—PALcode instruction (CALL_PAL) opcodes
 Res—Reserved for Digital

3.9 PALcode

In a family of machines, both users and operating system implementors require functions to be implemented consistently. When functions conform to a common interface, the code that uses those functions can be used on several different implementations without modification.

Five opcodes are provided by the Alpha AXP architecture as implementation-specific privileged instructions. These instructions are defined independently for each Alpha AXP hardware implementation to provide PALcode software routines with access to specific hardware state and functions.

3.9.1 Required PALcode Instructions

The PALcode instructions listed in Table 20 are described in the *Alpha Architecture Handbook*.

Table 20: Required PALcode Instructions

Mnemonic	Type	Operation
HALT	Privileged	Halt processor
IMB	Unprivileged	I-stream memory barrier

3.9.2 Architecturally Reserved PALcode Instructions

The hardware-specific instructions shown in Table 21 are executed in the PALcode environment. They produce OPCDEC exceptions if executed while not in the PALcode environment. These instructions are mapped using the architecturally reserved opcodes (PAL19, PAL1B, PAL1D, PAL1E, PAL1F). They can only be used while executing chip-specific PALcode.

Table 21: Instructions Specific to the 21064

Mnemonic	Type	Operation
HW_MTPR	PALmode, Privileged	Move data to processor register
HW_MFPR	PALmode, Privileged	Move data from processor register
HW_LD	PALmode, Privileged	Load data from memory
HW_ST	PALmode, Privileged	Store data in memory
HW_REI	PALmode, Privileged	Return from PALmode exception

Programming note: PALcode uses the HW_LD and HW_ST instructions to access memory outside of the realm of normal Alpha AXP memory management.

3.9.3 PAL_TEMP

The 21064 contains 32 registers that provide temporary storage for PALcode. These registers are accessible through HW_MXPR instructions.

3.9.4 Data Cache Status Register (C_STAT)

The DC_STAT is a read-only IPR and is for diagnostic use only.

In order to use this register, software must first execute the following instruction before executing the load or store, whose D-cache probe result is recorded in DC_HIT.

HW_MTPR R31, 4B (hex)

Figure 35: DECchip 21064 C_STAT

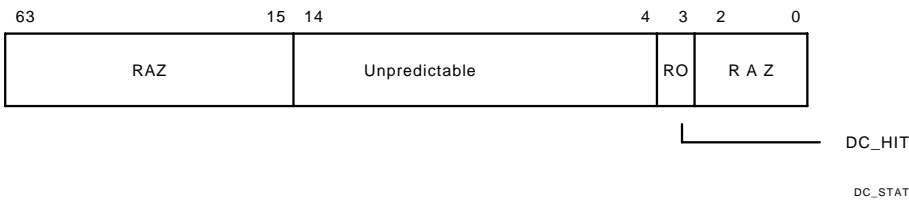


Table 22: DECchip 21064 Cache Status Register

Field	Description	Type
3	DC_HIT - This bit indicates whether the last load or store instruction processed by the Abox hit, (DC_HIT set) or missed, (DC_HIT clear) the D-Cache. Loads that miss the D-Cache may be completed without requiring external reads. e.g. pending fill or pending store hits.	RO

3.9.5 DECchip 21064-A275 Data Cache Status Register (C_STAT)

Figure 36: DECchip 21064-A275 C_STAT

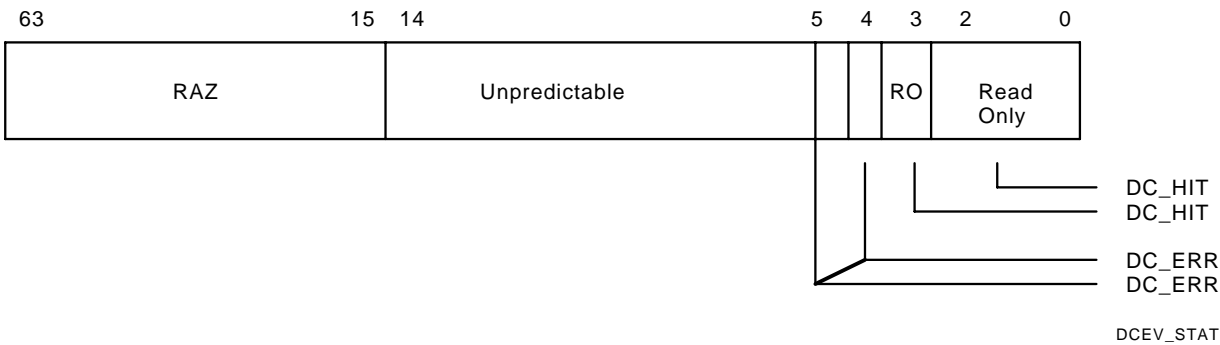


Table 23: DECchip 21064-A275 Cache Status Register

Field	Description
5	IC_ERR - Set by ICache parity error. Cleared by read of C_STAT register.
4	DC_ERR - Set by Dcache parity error. Cleared by read of C_STAT register.
3	DC_HIT - Same as existing hit bit in DECchip 21064.
2:0	DC_HIT (RO) This bit indicates whether the last load or store instruction processed by the Abox hit, (DC_HIT set) or missed, (DC_HIT clear) the D-Cache. Loads that miss the D-Cache may be completed without requiring external reads. e.g. pending fill or pending store hits.

3.9.6 Data Cache Address Register (DC_ADDR)

In the 21064, this is a pseudo-register used for unlocking DC_STAT.

In the 21064, DC_STAT and DC_ADDR are unlocked when DC_ADDR is read.

3.9.7 Bus Interface Unit Status Register (BIU_STAT)

BIU_STAT is a read-only IPR.

When one of BIU_HERR, BIU_SERR, BC_TPERR or BC_TCPERR is set, BIU_STAT[6..0] are locked against further updates, and the address associated with the error is latched and locked in the BIU_ADDR register. BIU_STAT[6..0] and BIU_ADDR are also spuriously locked when a parity error or an uncorrectable EDC error occurs during a primary cache fill operation. BIU_STAT[7..0] and BIU_ADDR are unlocked when the BIU_ADDR register is read.

When FILL_EDC or BIU_DPERR is set, BIU_STAT[13..8] are locked against further updates, and the address associated with the error is latched and locked in the FILL_ADDR register. BIU_STAT[14..8] and FILL_ADDR are unlocked when the FILL_ADDR register is read.

This register is not unlocked or cleared by reset and needs to be explicitly cleared by PALcode.

Figure 37: BIU_STAT

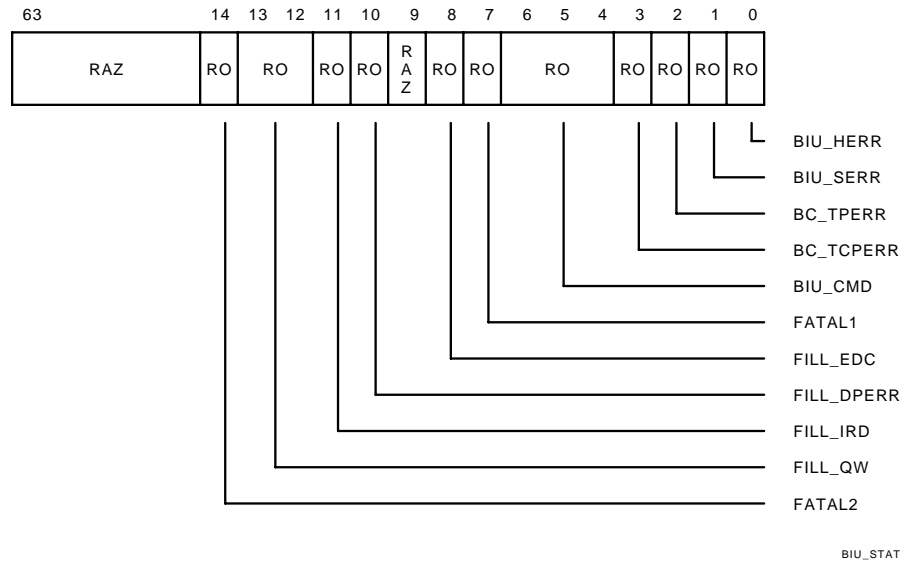


Table 24: BIU STAT

Field	Type	Description
BIU_HERR	RO	This bit, when set, indicates that an external cycle was terminated with the cAck_h pins indicating HARD_ERROR.
BIU_SERR	RO	This bit, when set, indicates that an external cycle was terminated with the cAck_h pins indicating SOFT_ERROR. NOTE: This should never occur on a Cobra System.
BC_TPERR	RO	This bit, when set, indicates that a external cache tag probe encountered bad parity in the tag address RAM.
BC_TCPERR	RO	This bit, when set, indicates that a external cache tag probe encountered bad parity in the tag control RAM.
BIU_CMD	RO	This field latches the cycle type on the cReq_h pins when a BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR error occurs.
FATAL1	RO	This bit, when set, indicates that an external cycle was terminated with the cAck_h pins indicating HARD_ERROR or that a an external cache tag probe encountered bad parity in the tag address RAM or the tag control RAM while BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR was already set.
FILL_EDC	RO	EDC error. This bit, when set, indicates that primary cache fill data received from outside the CPU chip contained an EDC error.
FILL_DPERR	RO	Fill Parity Error. This bit when set, indicates that the BIU received data with a parity error from outside the CPU chip while performing either a D-Cache or I-Cache fill. FILL_DPERR is only meaningful when the CPU chip is in parity mode, as opposed to EDC mode.
FILL_IRD	RO	This bit is only meaningful when either FILL_EDC or FILL_DPERR is set. FILL_IRD is set to indicate that the error which caused FILL_EDC or FILL_DPERR to set occurred during an I-Cache fill and clear to indicate that the error occurred during a D-Cache fill.
FILL_QW	RO	This field is only meaningful when either FILL_EDC or FILL_DPERR is set. FILL_QW identifies the quadword within the hexaword primary cache fill block which caused the error. It can be used together with FILL_ADDR[33..5] to get the complete physical address of the bad quadword.
FATAL2	RO	This bit, when set, indicates that a primary cache fill operation resulted in either an uncorrectable EDC error or in a parity error while FILL_EDC or FILL_DPERR was already set.

3.9.8 Bus Interface Unit Address Register (BIU_ADDR)

This read-only register contains the physical address associated with errors reported by BIU_STAT[7..0]. Its contents are meaningful only when one of BIU_HERR, BIU_SERR, BC_TPERR, or BC_TCPERR are set. Reads of BIU_ADDR unlock both BIU_ADDR and BIU_STAT[7..0].

In the 21064, BIU_ADDR[33..5] contain the values of adr_h[33..5] associated with the pin bus transaction which resulted in the error indicated in BIU_STAT[7..0].

In the 21064, if the BIU_CMD field of the BIU_STAT register indicates that the transaction which received the error was READ_BLOCK or load-locked, then BIU_ADDR[4..2] are UNPREDICTABLE. If the BIU_CMD field of the BIU_STAT register encodes any pin bus command other than READ_BLOCK or load-locked, then BIU_ADDR[4..2] will contain zeros. BIU_ADDR[63..34] and BIU_ADDR[1..0] always read as zero.

3.9.9 Fill Address Register (FILL_ADDR)

This read-only register contains the physical address associated with errors reported by BIU_STAT[14..8]. Its contents are meaningful only when FILL_EDC or FILL_DPERR is set. Reads of FILL_ADDR unlock FILL_ADDR, BIU_STAT[14..8] and FILL_SYNDROME.

In the 21064, FILL_ADDR[33..5] identify the 32-byte cache block which the CPU was attempting to read when the error occurred.

In the 21064, if the FILL_IRD bit of the BIU_STAT register is clear, indicating that the error occurred during a D-stream cache fill, then FILL_ADDR[4..2] contain bits [4..2] of the physical address generated by the load instruction which triggered the cache fill. If FILL_IRD is set, then FILL_ADDR[4..2] are UNPREDICTABLE. FILL_ADDR[63..34] and FILL_ADDR[1..0] read as zero.

3.9.10 Fill Syndrome Register (FILL_SYNDROME)

The FILL_SYNDROME register is a 14-bit read-only register.

If the chip is in EDC mode and an EDC error is recognized during a primary cache fill operation, the syndrome bits associated with the bad quadword are locked in the FILL_SYNDROME register. FILL_SYNDROME[6..0] contain the syndrome associated with the lower longword of the quadword, and FILL_SYNDROME[13..7] contain the syndrome associated with the upper longword of the quadword. A syndrome value of zero means that no errors were found in the associated longword. See Table 25 for a list of syndromes associated with correctable single-bit errors. The FILL_SYNDROME register is unlocked when the FILL_ADDR register is read.

If the chip is in parity mode and a parity error is recognized during a primary cache fill operation, the FILL_SYNDROME register indicates which of the longword in the quadword got bad parity. FILL_SYNDROME[0] is set to indicate that the low longword was corrupted, and FILL_SYNDROME[7] is set to indicate that the high longword was corrupted. FILL_SYNDROME[13..8] and [6..1] are RAZ in parity mode.

Figure 38: Fill Syndrome

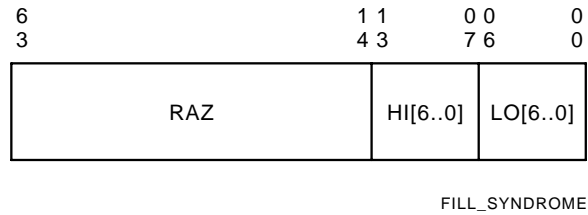


Table 25: Syndromes for Single-Bit Errors

Data Bit	Syndrome(Hex)	Check Bit	Syndrome(Hex)
00	4F	00	01
01	4A	01	02
02	52	02	04
03	54	03	08
04	57	04	10
05	58	05	20
06	5B	06	40
07	5D		
08	23		
09	25		
10	26		
11	29		

Table 25 (Cont.): Syndromes for Single-Bit Errors

Data Bit	Syndrome(Hex)	Check Bit	Syndrome(Hex)
12	2A		
13	2C		
14	31		
15	34		
16	0E		
17	0B		
18	13		
19	15		
20	16		
21	19		
22	1A		
23	1C		
24	62		
25	64		
26	67		
27	68		
28	6B		
29	6D		
30	70		
31	75		

NOTE

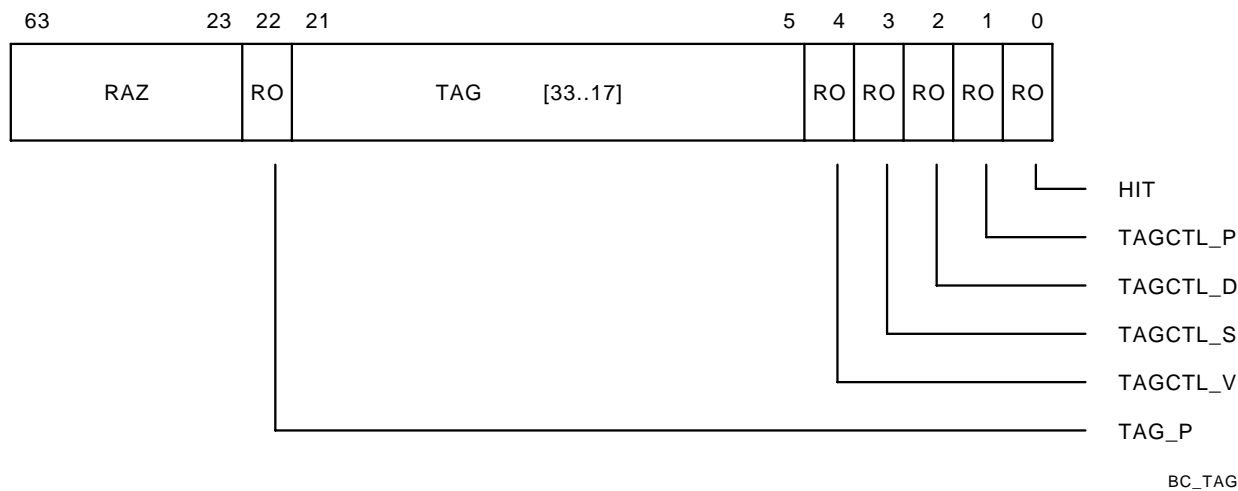
A syndrome of 1F is invalid and indicates that a bad EDC code was written intentionally into the cache. This is done when bad data is provided by the System-bus to the C4 interface. A syndrome of 0 indicates that no EDC error was detected.

3.9.11 Backup Cache Tag Register (BC_TAG)

BC_TAG is a read-only IPR. Unless locked, the BC_TAG register is loaded with the results of every backup cache tag probe. When a tag or tag control parity error or primary fill data error (parity or EDC) occurs this register is locked against further updates. Software may read the LSB of this register by using the HW_MFPR instruction. Each time an HW_MFPR from BC_TAG completes the contents of BC_TAG are shifted one bit position to the right, so that the entire register may be read using a sequence of HW_MFPRs. Software may unlock the BC_TAG register using a HW_MTPR to BC_TAG.

Successive HW_MFPRs from the BC_TAG register must be separated by at least one null cycle.

Figure 39: BC_TAG



Unused tag bits in the TAG field of this register are always clear, based on the size of the external cache as determined by the BC_SIZE field of the BIU_CTL register.

3.9.12 EDC Error Correction

When in EDC mode the 21064 generates longword EDC on writes, and checks EDC on reads.

When an EDC error is recognized during a D-Cache fill, the BIU places the affected fill block into the D-Cache unchanged, validates the block, and posts a machine check. The load instruction which triggered the D-Cache fill is completed by writing the requested longword(s) into the register file. The longword(s) read by the load instruction may or not have been the cause of the error, but a machine check is posted either way.

The Ibox will react to the machine check by aborting instruction execution before any instruction issued subsequent to the load could overwrite the register containing the load data, and vectoring to the PAL code machine check handler. Sufficient state is retained in various status registers (see Section C.8) for PAL code to determine whether the error affects the longword(s) read by the load instruction, and whether the error is correctable. In any event, PAL code must explicitly flush the D-Cache.

If the longword containing the error was written into the register file, report an uncorrectable hardware error to the operating system. Independent of whether the failing longword was read by the load instruction, PAL may scrub memory by explicitly reading the longword with the physical/lock variant of the HW_LD instruction, flipping the necessary bit, and writing the longword with the physical/conditional variant of the HW_ST instruction. Note that when PAL rereads the affected longword the hardware may report no errors, indicating that the longword has been overwritten.

When an EDC error occurs during an I-Cache fill the BIU places the affected fill block into the I-Cache unchanged, validates the block and posts a machine check. The Ibox will vector to the PAL code machine check handler before it executes any of the instructions in the bad block. PAL code may then flush the I-Cache and scrub memory as described above.

CHAPTER 4

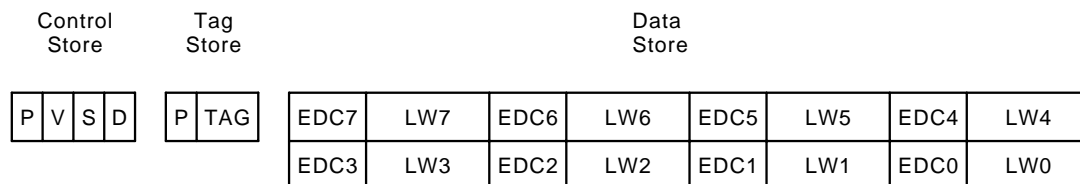
FUNCTIONS LOCATED ELSEWHERE ON THE CPU MODULE

4.1 Back-up Cache (B-Cache)

The B-Cache is a 1 Mbyte direct mapped physical write back cache. A 4 Mbyte cache may be used in future Sable systems. It has a fixed 32-byte block size and supports the System-bus snooping protocol to allow for a multi-processor implementation.

Each cache block entry is made up of three functionally identifiable storage element arrays. The control store, which is parity protected, contains the binary flags which indicate whether a particular cache block entry valid, dirty, and/or shared. The tag store, which is also parity protected, contains the high order address bits of the data currently stored in that particular cache block entry. The data store, which is EDC protected, contains the actual 32 bytes of “cached” data.

Figure 40: Back-up Cache Entry



For a complete description of the B-Cache allocation policy refer to *CPU Module Transactions* Chapter 5.

4.1.1 Control Store

The Control Store of the B-Cache is where the binary flags indicating the status of the cache block are found. These flags are defined in Table 26. To learn how to initialize any of these bit flags onpower-up refer to Section 9.2.

Table 26:

Flag	Description
VALID	When set indicates that the data found in the other bits of the control store, the tag store, and the data store contain valid information. The VALID bit is set only by the B-Cache controller when a cache block is filled with new data. The VALID bit is cleared only by the B-Cache controller when a System-bus write requires a cache block invalidation.
DIRTY	When set (and VALID bit set) indicates that the data store contains an updated copy of a main memory location. This cache data must be written back to main memory when the cache location is victimized. At most, in a Cobra system, there will be only one copy of any given memory location marked DIRTY. The DIRTY bit is set by the processor performing a Fast B-Cache write hit cycle, or by the B-Cache assisting the processor perform a STxC cycle. When the VALID bit is cleared the B-Cache controller will guarantee that the DIRTY bit is also cleared.
SHARED	When set (and VALID bit set) indicates that the data store contains a copy of a main memory location that another System-bus node also has a copy of. Writes to this location must “write-through” the cache so that a coherent view of memory is maintained. The SHARED bit is set by the B-Cache controller when the System-bus CSHARED_L signal is asserted during cache block allocation or when a System-bus read hits a location found in the B-Cache (CSHARED_L must be pulled). The SHARED bit is cleared by the B-Cache controller when it performs a System-bus WRITE and the CSHARED_L signal isn't asserted. When the VALID bit is cleared the B-Cache controller guarantees that the DIRTY bit is also cleared.
PARITY	Contains EVEN parity over the contents of the Control Store. The PARITY is valid even if the VALID bit is not set. Parity is checked by the processor during every B-Cache probe cycle, and by the B-Cache controller during every System-bus initiated probe cycle. For details regarding parity error detection refer to Chapter 8.

4.1.2 Tag Store

The Tag Store contains the high order address bits <30:20> (<30:22> for a 4 MB cache) of the memory location that currently resides in the cache entry. There is a single parity bit that provides EVEN parity over the complete Tag Store. Once power-up initialization has occurred, the parity bit will contain valid parity regardless of the value of the Control Stores VALID bit.

Parity is checked by the processor during every B-Cache probe cycle, and by the B-Cache controller during every System-bus initiated probe cycle. For details regarding parity error detection refer to Chapter 8.

4.1.3 Data Store

The Data Store contains the actual data of the memory location that is “cached”. Every cache entry is made up of 8 longwords each longword protected by 7 bits of EDC. Physically the cache is only 4 longwords wide, so a cache block consists of 2 consecutive addresses aligned on a 32 byte block boundary. After initialization, the EDC bits are valid regardless of the validity of the Control Store's VALID bit. Error

checking is performed whenever the processor hits the cache on a read and error checking and possibly correcting is performed whenever a System-bus read probe hits dirty, the victimization of a dirty location, or a masked processor write to a shared location. For details regarding EDC error detection refer to Chapter 8.

4.1.4 B-Cache Control Register Definitions

The B-Cache subsystem is manipulated at separate times by two different controllers. The first is the processor which will probe the tag field, and if hit, will read or write data into the B-Cache. The behavior of the processor relative to the B-Cache is controlled or monitored by the processors BIU_STAT, BIU_ADDR, FILL_ADDR, BIU_CTL, AND BC_TAG internal processor registers. For a description of these registers refer to the Section 3.9.3.

The second is the B-Cache controller that processes all miss traffic and System-bus probe activity. The behavior of the B-Cache controller is determined by the settings of the B-Cache CSR's defined below.

4.1.4.1 CSR Space

There exists primary and secondary CSR space. Primary CSR space is used by Sable systems, secondary CSR space is used by Cobra systems.

Sable CPUs recognize only primary CSR space. What follows is a list of base addresses for the CPUs.

Table 27: Base Addresses for CSRs

CPU	Base Address
CPU ₀	3.8000.0000
CPU ₁	3.8100.0000
CPU ₂	3.8200.0000
CPU ₃	3.8300.0000

4.1.4.2 B-Cache Control Register - CSR0

CSR offset = 00₁₆

The B-Cache Control Register contains all of the control fields to enable/disable various functions of the B-Cache. This register is provided primarily for the use of diagnostic self test code running at system initialization time.

NOTE: The CPU shall ensure that the state of control flags in the low longword are consistent with control flags in the upper longword by performing quadword writes.

Figure 41: B-Cache Control Register (BCC)

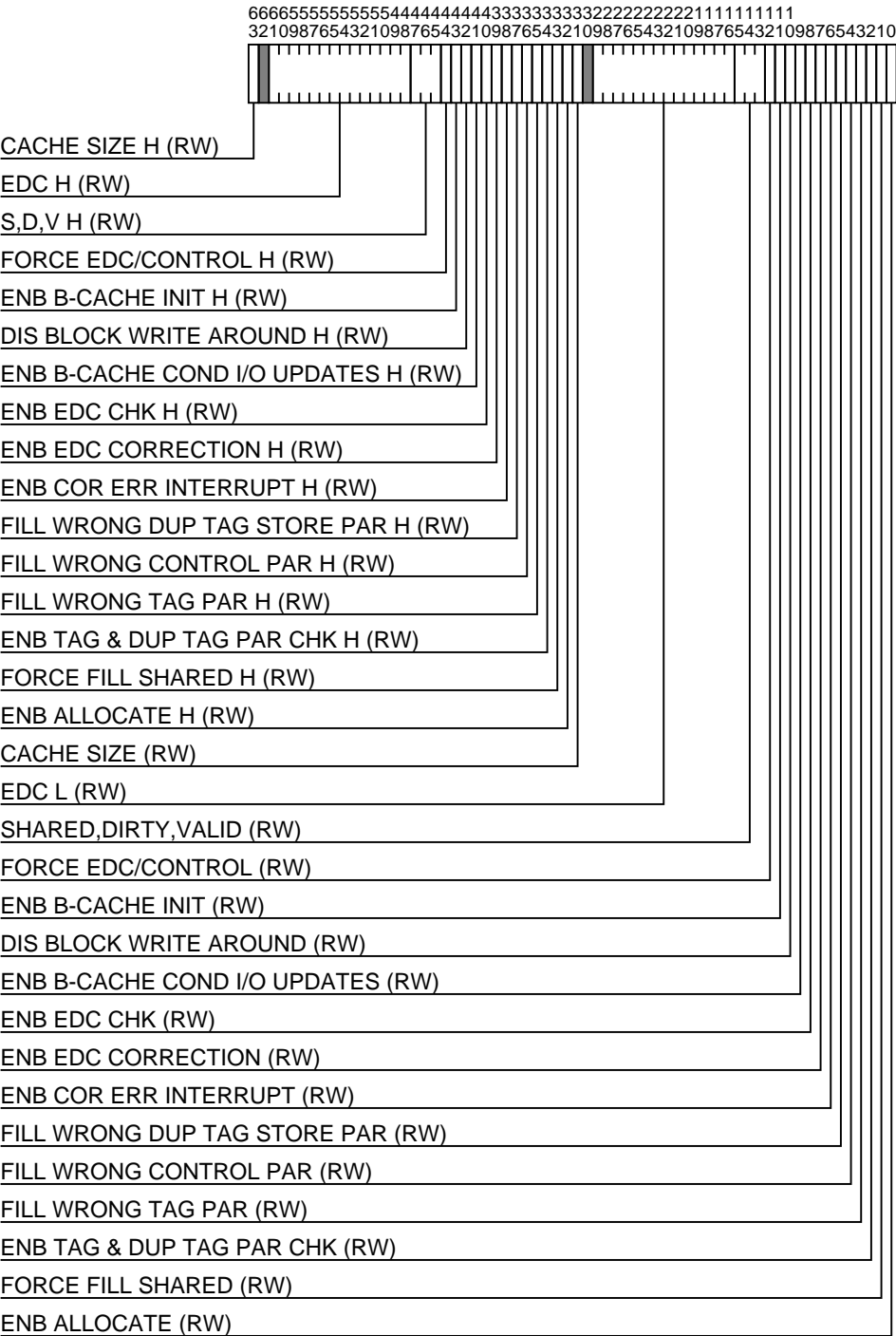


Table 28: B-Cache Control Register Description

Field	Description
63	CACHE SIZE H [<i>read/write</i>] Cleared on power-up, These bits must be correctly set before the B-Cache is accessed, and match bits <31:30>. <ul style="list-style-type: none"> • 1 - 4MB Cache • 0 - 1MB Cache The value of this field controls not only which tag signals are used in determining the proper parity value for the tag, but also in determining whether address bits 21 and 20 should be driven by the Cobra Bus interface ASICs during access to the B-Cache.
61:48	EDC H [<i>read/write</i>] Cleared on power-up, When the FORCE EDC/CONTROL bit 12 is set during B-Cache initialization, the values specified into this register are written forced on the EDC field of longwords 3 and 1 of any filled B-Cache locations. Refer to Section 9.2 for further details.
47:45	S,D,V H [<i>read/write</i>] Cleared on power-up, These bits must always match bits <15:13> in this register.
44	FORCE EDC/CONTROL H [<i>read/write</i>] Cleared on power-up, This bit must always match bit <12> in this register.
43	ENB B-CACHE INIT H [<i>read/write</i>] Cleared on power-up, This bit must always match bit <11> in this register.
42	DIS BLOCK WRITE AROUND H [<i>read/write</i>] Cleared on power-up, This bit must always match bit <10> in this register.
41	ENB B-CACHE COND I/O UPDATES H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <9> in this register.
40	ENB EDC CHK H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <8> in this register.
39	ENB EDC CORRECTION H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <7> in this register.
38	ENB COR ERR INTERRUPT H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <6> in this register.
37	FILL WRONG DUP TAG STORE PAR H [<i>read/write</i>] Cleared on power-up, This bit must always match bit <5> in this register.
36	FILL WRONG CONTROL PAR H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <4> in this register.
35	FILL WRONG TAG PAR H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <3> in this register.
34	ENB TAG & DUP TAG PAR CHK H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <2> in this register.
33	FORCE FILL SHARED H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <1> in this register.
32	ENB ALLOCATE H [<i>read/write</i>] Cleared on power-up. This bit must always match bit <0> in this register.
31	CACHE SIZE [<i>read/write</i>]

Table 28 (Cont.): B-Cache Control Register Description

Field	Description
	<p>Cleared on power-up, These bits must be correctly set before the B-Cache is accessed.</p> <ul style="list-style-type: none"> • 1 - 4MB Cache • 0 - 1MB Cache <p>The value of this field controls not only which tag signals are used in determining the proper parity value for the tag, but also in determining whether address bits 21 and 20 should be driven by the Cobra Bus interface ASICs during access to the B-Cache.</p>
29:16	<p>EDC L [<i>read/write</i>]</p> <p>Cleared on power-up, When the FORCE EDC/CONTROL bit 12 is set during B-Cache initialization, the values specified into this register are written forced on the EDC field of longwords 2 and 0 of any filled B-Cache locations. Refer to Section 9.2 for further details.</p>
15:13	<p>SHARED,DIRTY,VALID [<i>read/write</i>]</p> <p>Cleared on power-up, When the FORCE EDC/CONTROL bit is set during B-Cache initialization, the values specified in here in SHARED,DIRTY, and VALID will be filled into the control field of the B-Cache location referenced by a <i>READ_BLOCK</i>. Refer to Section 9.2 for further details. When FORCE EDC/CONTROL is cleared these bits have no effect.</p>
12	<p>FORCE EDC/CONTROL [<i>read/write</i>]</p> <p>Cleared on power-up. Attention to the state of bits <8,7,5:2> should be given when setting this bit, to avoid machine check responses. When set, for every processor <i>READ_BLOCK</i> cycle to a cacheable location the appropriate B-Cache location (indicated by address bits <19:5>) is updated as follows:</p> <ol style="list-style-type: none"> 1. Tag Probe at <i>READ_BLOCK</i> address is forced clean, to avoid victims. 2. The value of address bits <30:19> with its associated parity is filled into the B-Cache tag store. 3. The value of the control bits specified in this register's SHARED,DIRTY, and VALID and their associated parity are filled into the B-Cache control store, and the EDC bits specified here in EDC H and L are filled into the EDC field of the data store. 4. The B-Cache is updated with the data returned on the System-bus or the CSR data if the ENB B-CACHE INIT bit <11> is set. This data is also returned to the processor to satisfy the <i>READ_BLOCK</i> request. <p>This bit is NOT self clearing. Refer to Section 9.2 for further details.</p>
11	<p>ENB B-CACHE INIT [<i>read/write</i>]</p>

Table 28 (Cont.): B-Cache Control Register Description

Field	Description
	<p>Cleared on power-up. When set, READ transactions to address space 1 0000 0000 - 1 7FFF FFFF are forced to return data from the corresponding CPU's CSR8-CBEAL register. When set, memory WRITE allocate transactions are not supported, however CSR write transactions are supported. This causes the transactions to the System-bus reserved memory address space to supply a known pattern, which is the transaction address as it appears on the System-bus, to fill into the B-Cache. For a CPU commander, the CSR data will be filled to the B-Cache tag and data entry indexed by the transaction physical address as follows :</p> <p><u>INDEX TAG FILLED WITH ADDRESS BITS</u></p> <p><19:5> <30:20> for 1MB cache <21:5> <30:22> for 4MB cache</p> <p>Also, when this bit is set, the tag probe at the transaction address is forced clean to avoid victims, and if bits <32> and <0> ENB ALLOCATE are also set the B-Cache tag control bits are filled with the value specified in bits <15:13> of this register correct parity unless parity is forced bad, regardless of the state of bit <12> and the returned data is written into the B-Cache data store. Refer to Section 9.2 for the actual returned data format.</p>
10	<p>DIS BLOCK WRITE AROUND [read/write]</p> <p>Cleared on power-up. When clear only masked processor writes result in allocation, full block - unmasked writes will write around the B-Cache and not result in allocation. When set processor writes result in allocation regardless of the block mask.</p>
9	<p>ENB B-CACHE COND I/O UPDATES [read/write]</p> <p>Cleared on power-up. When set, Conditional updates of the B-Cache will occur due to System-bus writes when the I/O module is the commander. A System-bus write will cause an update if both the contents of the duplicate tag store and the B-Cache indicate a hit. When clear, Unconditional updates of the B-Cache will occur when a System-bus write hits a VALID B-Cache location when the I/O module is the System-bus commander. Refer to Chapter 6 and Section 4.3 for further details about conditional invalidation and updating.</p>
8	<p>ENB EDC CHK [read/write]</p> <p>Cleared on power-up. When set, the EDC of the B-CACHE Data Store and processor written data will be checked. EDC will be checked by the B-Cache controller only when a System-bus READ or EXCHANGE hits Dirty, a masked write to a shared location occurs, or during a victim write. When clear, checking for single and multiple bit errors is disabled, but EDC generation still occurs.</p>
7	<p>ENB EDC CORRECTION [read/write]</p> <p>Cleared on power-up to disable EDC correction of data during any transaction, even correctable single bit errors are reported as uncorrectable. When set, enables EDC correction of processor write data, victim data, and dirty read hit data.</p>
6	<p>ENB COR ERR INTERRUPT [read/write]</p> <p>Cleared on power-up. When set, EDC correctable errors detected during any transaction that is logged by CSR1 will result in the assertion of the HARDWARE ERROR INTERRUPT. When clear the interrupt is not sent but error information is captured in CSR1.</p>
5	<p>FILL WRONG DUP TAG STORE PAR [read/write]</p> <p>Cleared on power-up. When set, the wrong parity value will be written in the Primary D-Cache duplicate tag store parity location when a primary D-Cache location is updated during a processor primary D-cache allocation. This bit is NOT self clearing. When this bit is set, the ENB TAG & DUP TAG PAR CHK bits 34 and 2 should be cleared; otherwise a HARDWARE ERROR interrupt will be signaled every time the duplicate tag store is written.</p>

Table 28 (Cont.): B-Cache Control Register Description

Field	Description
4	FILL WRONG CONTROL PAR [<i>read/write</i>] Cleared on power-up. When set, wrong parity is forced onto the B-Cache tag control store during the next B-Cache allocation/updates (System-bus read from this processor or write accept). This bit is self clearing.
3	FILL WRONG TAG PAR [<i>read/write</i>] Cleared on power-up. When set, wrong parity is forced onto the B-Cache tag address store during the next B-Cache allocation/update (System-bus read from this processor or write accept). This bit is self clearing.
2	ENB TAG & DUP TAG PAR CHK [<i>read/write</i>] Cleared on power-up. When set, enables parity checking of B-Cache tag address, tag control, and duplicate tag memory contents whenever the B-Cache controller accesses the B-Cache and/or the duplicate tag store of the primary cache. When clear, checking of tag address, tag control, and duplicate tag address parity is ignored.
1	FORCE FILL SHARED [<i>read/write</i>] Cleared on power-up. When set, forces the setting of the SHARED bit in the Tag Control Store when a new entry is allocated in the B-Cache.
0	ENB ALLOCATE [<i>read/write</i>] Cleared on power-up. When set, enables the filling of the B-Cache when cacheable cycles occur. When clear, B-Cache allocation is disabled, but victimization of dirty cache lines still occurs. This means that the cache line associated with the read/write request is written to memory if dirty and then marked invalid; or if not dirty then no change in the cache line state occurs. In either case the processor is informed not to cache the data in either internal cache. The B-Cache still provides data to the System-bus when read cycles probe dirty, System-bus write accepting continues. Invalidation of the primary cache due to Cobra bus writes still occurs. Because coherence is maintained, flushing the primary and backup caches is not necessary before re-enabling them. This bit can be cleared directly by writing or by any B-Cache error reported by CSR3 bits 1 or 3. This bit will not be set-able if CSR3 bits 1 or 3 are set.

4.1.4.3 B-Cache Correctable Error Register - CSR1

CSR offset = 20₁₆

The B-Cache Correctable Error Register latches the state of the B-Cache tag and control stores when a correctable EDC error (during the data portion of the cycle) is detected. The contents of B-Cache Correctable Error Address Register are not updated while error flags are set, lost error flags do not inhibit error logging.

These errors are only detected as a result of a processor masked write hit to a shared location, victimization of a cache location, or a System-bus READ or EXCHANGE to a dirty location.

Figure 42: B-Cache Correctable Error Register (BCCE)

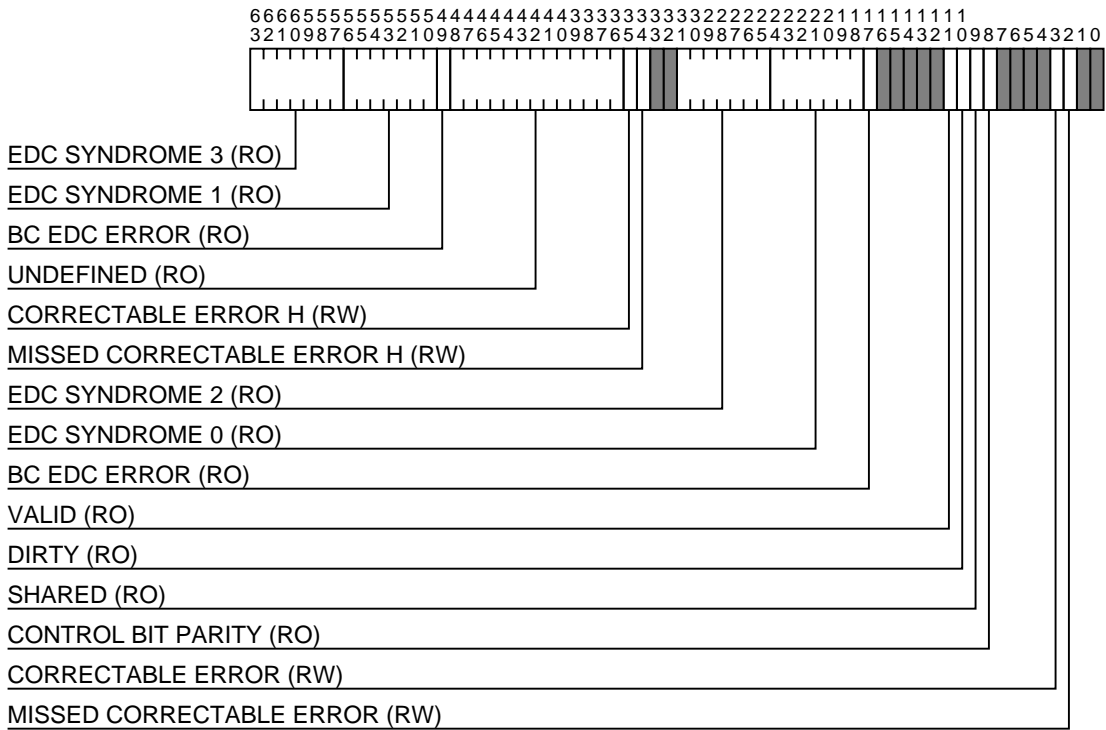


Table 29: B-Cache Correctable Error Register Description

Field	Description
63:57	EDC SYNDROME 3 [<i>read-only</i>] Undefined on power-up. EDC SYNDROME 3 is Valid when a correctable error has occurred. This register is updated when the BCCE ERROR H bit is clear. The syndrome contained in this register is relevant to longword 3 of data. Refer to Table 25 for the single bit error syndrome list.
56:50	EDC SYNDROME 1 [<i>read-only</i>]

Table 29 (Cont.): B-Cache Correctable Error Register Description

Field	Description
	Undefined on power-up. EDC SYNDROME 1 is Valid when a correctable error has occurred. This register is updated when the BCCE ERROR H bit is clear. The syndrome contained in this register is relevant to longword 1 of data. Refer to Table 25 for the single bit error syndrome list.
49	BC EDC ERROR [<i>read-only</i>] Undefined on power-up. Valid when a correctable error has occurred to indicate that the CPU or the B-Cache data was the cause of the error. When set the B-Cache was the cause of the data error. This register is updated when the BCCE ERROR H bit is clear.
48:36	UNDEFINED [<i>read-only</i>] Undefined
35	CORRECTABLE ERROR H [<i>read/write</i>] Cleared on power-up. This must be cleared when CORRECTABLE ERROR is cleared. This bit will not be set if a single bit error occurs when EDC correction is disabled. (The UNCORRECTABLE ERROR bit will be set in CSR3.) Write "1" to clear.
34	MISSED CORRECTABLE ERROR H [<i>read/write</i>] Cleared on power-up. This must be cleared when MISSED CORRECTABLE ERROR is cleared. Write "1" to clear.
31:25	EDC SYNDROME 2 [<i>read-only</i>] Undefined on power-up. Valid when a correctable error has occurred. This register is updated when the BCCE ERROR bit is clear. The syndrome contained in this register is relevant to longword 2 of data. Refer to Table 25 for the single bit error syndrome list.
24:18	EDC SYNDROME 0 [<i>read-only</i>] Undefined on power-up. Valid when a correctable error has occurred. This register is updated when the BCCE ERROR bit is clear. The syndrome contained in this register is relevant to longword 0 of data. Refer to Table 25 for the single bit error syndrome list.
17	BC EDC ERROR [<i>read-only</i>] Undefined on power-up. Valid when an correctable error has occurred to indicate that the CPU or the B-Cache data was the cause of the error. When set the B-Cache was the cause of the data error. This register is updated when the BCCE ERROR bit is clear.
11	VALID [<i>read-only</i>] Cleared on power-up. Contains the value of the VALID bit for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear.
10	DIRTY [<i>read-only</i>] Cleared on power-up. Contains the value of the DIRTY bit for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear.
9	SHARED [<i>read-only</i>] Cleared on power-up. Contains the value of the SHARED bit for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear.
8	CONTROL BIT PARITY [<i>read-only</i>]

Table 29 (Cont.): B-Cache Correctable Error Register Description

Field	Description
	Cleared on power-up. Contains the value of Control Bit Parity for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear.
3	CORRECTABLE ERROR [<i>read/write</i>] Cleared on power-up, Set when an Correctable EDC error occurs in the B-Cache which causes the contents of this registers (non-error bits) and the contents of the B-Cache Correctable Error Address Register to freeze. This bit will not be set if a single bit error occurs when EDC correction is disabled. (The UNCORRECTABLE ERROR bit will be set in CSR3.) Write "1" to clear.
2	MISSED CORRECTABLE ERROR [<i>read/write</i>] Cleared on power-up, Set when a Correctable EDC error occurs in the B-Cache and CORRECTABLE ERROR bit is set from a previous error. A single cache block will only log one error even if both octawords of the cache block hexaword are bad. Thus an error in a single cache block will not cause the MISSED CORRECTABLE ERROR bit to be set. Write "1" to clear.

4.1.4.4 B-Cache Correctable Error Address Register - CSR2

CSR offset = 40₁₆

When a B-Cache correctable EDC error has been detected, the B-Cache Correctable Error Address Register contains the index of the B-Cache location that contains the error.

Figure 43: B-Cache Correctable Error Address Register (BCCEA)

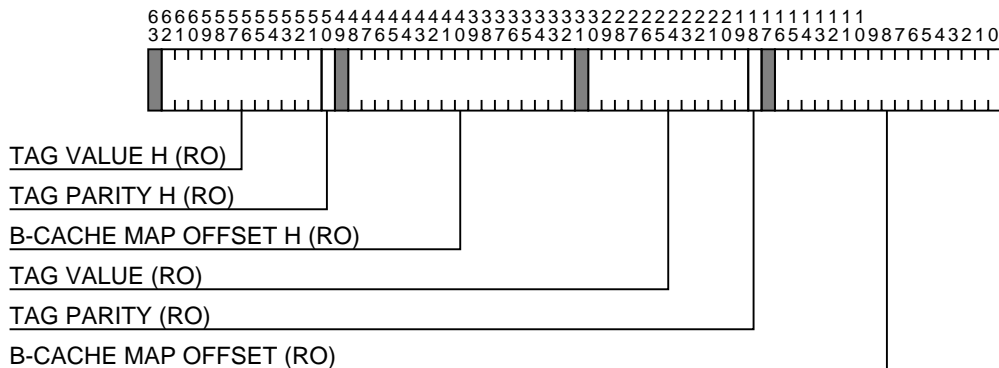


Table 30: B-Cache Correctable Error Address Register Description

Field	Description
62:51	TAG VALUE H [<i>read-only</i>] Cleared on power-up. Contains the value of the Tag for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear. If LW 3 or LW 1 has a correctable EDC error this field provides an address pointer into the B-Cache.
50	TAG PARITY H [<i>read-only</i>] Cleared on power-up. Contains the value of Tag Store Parity for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear.
48:32	B-CACHE MAP OFFSET H [<i>read-only</i>] Cleared on power-up. Contains the last B-Cache MAP index. If a correctable EDC error has been detected, then the contents of this register are frozen until the BCCE ERROR bit in the BCCE Register has been cleared. If LW 3 or LW 1 has a correctable EDC error this field provides an address pointer into the B-Cache.
30:19	TAG VALUE [<i>read-only</i>] Cleared on power-up. Contains the value of the Tag for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear. If LW 2 or LW 0 has a correctable EDC error this field provides an address pointer into the B-Cache.
18	TAG PARITY [<i>read-only</i>] Cleared on power-up. Contains the value of Tag Store Parity for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCCE ERROR bit is clear.

Table 30 (Cont.): B-Cache Correctable Error Address Register Description

Field	Description
16:0	B-CACHE MAP OFFSET [<i>read-only</i>] Cleared on power-up. Contains the last B-Cache MAP index. If a correctable EDC error has been detected, then the contents of this register are frozen until the BCCE ERROR bit in the BCCE Register has been cleared. If LW 2 or LW 0 has a correctable EDC error this field provides an address pointer into the B-Cache.

4.1.4.5 B-Cache Uncorrectable Error Register - CSR3

CSR offset = 60₁₆

The B-Cache Uncorrectable Error Register latches the state of the B-Cache tag and control stores when a parity error or an EDC uncorrectable error (during the data portion of the cycle) is detected. The contents of B-Cache Uncorrectable Error Address Register are not updated while error flags are set, the lost error flag does not inhibit error logging.

These errors are only detected by the B-Cache controller as a result of a processor masked write hit to a shared location, a LDxL or STxC, victimization of a cache location, or a System-bus READ, WRITE, or EXCHANGE transaction. (EDC errors are not detected by the Cobra Bus interface ASICs on LDxL, and System-bus WRITE transactions as a bystander or responder.)

Figure 44: B-Cache Uncorrectable Error Register (BCUE)

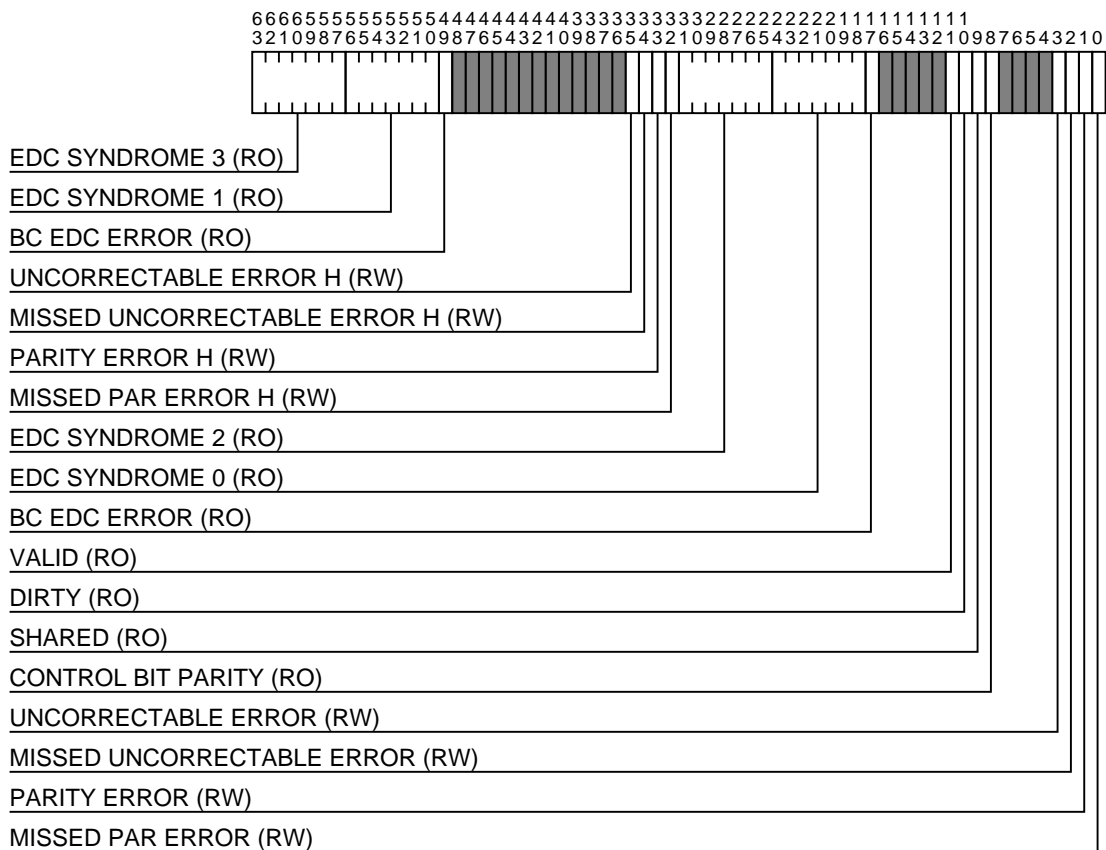


Table 31: B-Cache Uncorrectable Error Register Description

Field	Description
63:57	EDC SYNDROME 3 [<i>read-only</i>] Undefined on power-up. EDC SYNDROME 3 is Valid when a uncorrectable error has occurred. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR H bits are clear. The syndrome contained in this register is relevant to longword 3 of data.
56:50	EDC SYNDROME 1 [<i>read-only</i>] Undefined on power-up. EDC SYNDROME 1 is Valid when a uncorrectable error has occurred. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR H bits are clear. The syndrome contained in this register is relevant to longword 1 of data.
49	BC EDC ERROR [<i>read-only</i>] Undefined on power-up. Valid when a uncorrectable error has occurred to indicate that the CPU or the B-Cache data was the cause of the error. When set the B-Cache was the cause of the data error. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR H bits are clear.
35	UNCORRECTABLE ERROR H [<i>read/write</i>] Cleared on power-up, This must be cleared when UNCORRECTABLE ERROR is cleared. Write "1" to clear.
34	MISSED UNCORRECTABLE ERROR H [<i>read/write</i>] Cleared on power-up, This must be cleared when MISSED UNCORRECTABLE ERROR is cleared. Write "1" to clear.
33	PARITY ERROR H [<i>read/write</i>] Cleared on power-up. This must be cleared when PARITY ERROR is cleared. Write "1" to clear.
32	MISSED PAR ERROR H [<i>read/write</i>] Cleared on power-up. When set indicates that a parity error occurred in the tag or tag control store while the PARITY ERROR or the UNCORRECTABLE ERROR bit was set. Write "1" to clear.
31:25	EDC SYNDROME 2 [<i>read-only</i>] Undefined on power-up. Valid when a uncorrectable error has occurred. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear. The syndrome contained in this register is relevant to longword 2 of data.
24:18	EDC SYNDROME 0 [<i>read-only</i>] Undefined on power-up. Valid when a uncorrectable error has occurred. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear. The syndrome contained in this register is relevant to longword 0 of data.
17	BC EDC ERROR [<i>read-only</i>] Undefined on power-up. Valid when a uncorrectable error has occurred to indicate that the CPU or the B-Cache data was the cause of the error. When set the B-Cache was the cause of the data error. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.
11	VALID [<i>read-only</i>] Cleared on power-up. Contains the value of the VALID bit for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.
10	DIRTY [<i>read-only</i>]

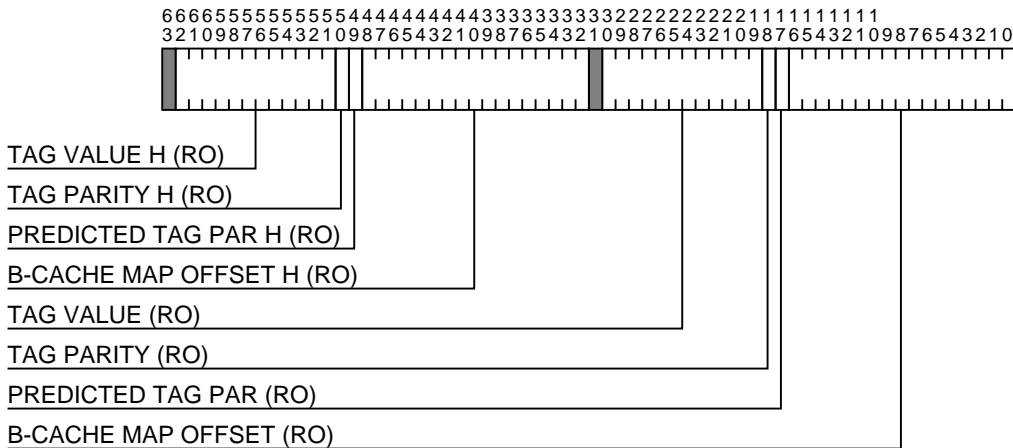
Table 31 (Cont.): B-Cache Uncorrectable Error Register Description

Field	Description
	Cleared on power-up. Contains the value of the DIRTY bit for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.
9	SHARED [<i>read-only</i>] Cleared on power-up. Contains the value of the SHARED bit for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.
8	CONTROL BIT PARITY [<i>read-only</i>] Cleared on power-up. Contains the value of Control Bit Parity for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.
3	UNCORRECTABLE ERROR [<i>read/write</i>] Cleared on power-up, Set when an Uncorrectable EDC error occurs in the B-Cache which causes the contents of this registers (non-error bits) and the contents of the B-Cache Uncorrectable Error Address Register to freeze. Write "1" to clear. When this bit is set, CSR0 bit 0 (ENB ALLOCATE) is cleared, to stop allocation.
2	MISSED UNCORRECTABLE ERROR [<i>read/write</i>] Cleared on power-up, Set when an Uncorrectable EDC error occurs in the B-Cache and UNCORRECTABLE ERROR or PAR ERROR bits are set from a previous error. A single cache block will only log one error even if both octawords of the cache block hexaword are bad. Thus an error in a single cache block will not cause the MISSED UNCORRECTABLE ERROR bit to be set. Write "1" to clear.
1	PARITY ERROR [<i>read/write</i>] Cleared on power-up. When set indicates that the BCUE register contains information about a B-Cache Tag Error or Control Store Parity Error. Setting causes the contents of this registers (non-error bits) and the contents of the B-Cache Uncorrectable Error Address Register to freeze. Write "1" to clear. When this bit is set, CSR0 bit 0 is cleared, to stop allocation and to disable parity checking on the B-Cache Tag address and Control storage elements during local CPU initiated reads and writes.
0	MISSED PAR ERROR [<i>read/write</i>] Cleared on power-up. When set indicates that a parity error occurred in the tag or tag control store while the PARITY ERROR or the UNCORRECTABLE ERROR bit was set. Write "1" to clear.

4.1.4.6 B-Cache Uncorrectable Error Address Register - CSR4**CSR offset = 80₁₆**

When a B-Cache tag store or control store parity error, or an uncorrectable EDC error has been detected, the B-Cache Uncorrectable Error Address Register contains the index of the B-Cache location that contains the error.

The B-Cache Uncorrectable Error Address Register for CPU₀ is located at address 2.0000.0080 (hex), and for CPU₁ at address 2.0800.0080 (hex).

Figure 45: B-Cache Uncorrectable Error Address Register (BCUEA)**Table 32: B-Cache Uncorrectable Error Address Register Description**

Field	Description
62:51	TAG VALUE H [<i>read-only</i>] Cleared on power-up. Contains the value of the Tag for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear. If LW 3 or LW 1 has an uncorrectable EDC error this field provides an address pointer into the B-Cache.
50	TAG PARITY H [<i>read-only</i>] Cleared on power-up. Contains the value of Tag Store Parity for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.
49	PREDICTED TAG PAR H [<i>read-only</i>]

Table 32 (Cont.): B-Cache Uncorrectable Error Address Register Description

Field	Description
	<p>Cleared on power-up. Contains the value of the last System-bus predicted tag parity. The value of this bit will be frozen if a parity error occurs in the tag, control store, or an EDC error is detected, on a victim write, a masked write to a shared location, a LDxL or STxC, victimization of a cache location, or a System-bus READ, WRITE or EXCHANGE. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.</p> <p>This has been added to provide to power-up diagnostics visibility to an internal parity tree. The value of this location is valid if the register is frozen and the cycle causing the error was a System-bus read or write or if the register is not frozen and the register is read (the EVEN parity of the address bits <33:20> during the CSR read access will be found in this location.) This bit should be disregarded during normal system operation. Refer to Chapter 10 for further details.</p>
48:32	<p>B-CACHE MAP OFFSET H [<i>read-only</i>]</p> <p>Cleared on power-up. Contains the last B-Cache MAP index (address bits <19:5> for 1MB backup cache, address bits 21:5 for 4MB backup cache) If a parity error occurs in the tag, or control bits, or and EDC error has been detected, then the contents of this register are frozen until the BCUE UNCORRECTABLE and PARITY ERROR bits are clear. If LW 3 or LW 1 has an uncorrectable EDC error this field provides an address pointer into the B-Cache.</p>
30:19	<p>TAG VALUE [<i>read-only</i>]</p> <p>Cleared on power-up. Contains the value of the Tag for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear. If LW 2 or LW 0 has an uncorrectable EDC error this field provides an address pointer into the B-Cache.</p>
18	<p>TAG PARITY [<i>read-only</i>]</p> <p>Cleared on power-up. Contains the value of Tag Store Parity for the last B-Cache location accessed by the B-Cache controller. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.</p>
17	<p>PREDICTED TAG PAR [<i>read-only</i>]</p> <p>Cleared on power-up. Contains the value of the last System-bus predicted tag parity. The value of this bit will be frozen if a parity error is detected in the tag, tag control store, or an EDC error is detected, when a System-bus System-bus READ, WRITE or EXCHANGE probe occurs. This register is updated when the BCUE UNCORRECTABLE and PARITY ERROR bits are clear.</p> <p>This has been added to provide to power-up diagnostics visibility to an internal parity tree. The value of this location is valid if the register is frozen and the cycle causing the error was a System-bus read or write (the EVEN parity of the address bits <30:20> for 1MB cache) This bit should be disregarded during normal system operation. Refer to Chapter 10 for further details.</p>
16:0	<p>B-CACHE MAP OFFSET [<i>read-only</i>]</p> <p>Cleared on power-up. Contains the last B-Cache MAP index. If a parity error occurs in the tag, or control bits, or and EDC error has been detected, then the contents of this register are frozen until the BCUE UNCORRECTABLE and PARITY ERROR bits are clear. If LW 2 or LW 0 has an uncorrectable EDC error this field provides an address pointer into the B-Cache.</p>

4.1.5 Back-up Cache Cycle Time

4.1.5.1 The 21064 Cycles

The cycle time of the cache is between 20-32 ns which gives you the following overall the 21064 cycle times:

Table 33: The 21064 B-Cache cycle times

The 21064 Transaction	Processor	Speed (ns)	Size (Bytes)	Eff. Band (MB/sec)
Fast Read Hit	B2020-AA	52.6	32	608
Fast Write Hit	B2020-AA	73.64	32	435
Fast Read Hit	B2024-AA	50.9	32	629
Fast Write Hit	B2024-AA	61.8	32	518

4.1.5.2 System-bus Cycles

The B-Cache is single ported, and as such the System-bus and the processor must contend for its ownership. The arbitration algorithm provides top priority to the System-bus. Thus whenever a System-bus transaction is requested by a commander, every B-Cache on the System-bus (maximum of 4 in a Sable quad-processor system) will be unavailable to its processor for the number of System-bus cycles shown in the Table 34. The System-bus guarantees that a processor will not be held off its own cache for any more than two consecutive System-bus WRITE cycles.

Table 34: System-bus B-Cache Access Time

System-bus Transaction	Access Time (System-bus cycles)†
Read Hit Dirty	6
Read Hit Clean	5
Read Miss	3
Write Hit	5
Write Miss	3
Exchange Hit Dirty	6
Exchange Hit Clean	5
Exchange Miss	3

†The cycle time of the System-bus is 24 or 25 ns.

4.2 Cache Block Merge Buffer

The Cache Block Merge Buffer is a CPU specific buffer used to prevent word tearing caused by simultaneous writes to different words in the same cache block. It is also used to form complete cache blocks of masked writes when the B-Cache is disabled.

The merge buffer consists of a 32 byte data buffer, a valid bit associated with each longword in the buffer, and a complete address tag.

When a masked write miss occurs, the data longwords indicated by the 21064 longword mask are written into the Cache Block Merge Buffer and the associated valid bits are set. This miss is handled like other misses in that a System-bus Read is performed to allocate the cache entry. As the read data is returned off the System-bus it is merged with the data already in the merge buffer and then written into the B-Cache data store.

If an intervening System-bus write occurs to that same cache entry (by a different System-bus commander), the valid longwords in the merge buffer are merged with the new data provided by that System-bus write cycle, and subsequent System-bus write (if that datum is now SHARED) provides the newly merged data to the System-bus.

4.3 Duplicate Primary Data Cache Tag Store

Every CPU module provides a mechanism to perform “conditional invalidation” of the cache memory subsystems. This is accomplished through the use of a copy of the Primary D-Cache tag store. As updates to cacheable locations occur, invalidation of the internal data cache is 100% accurate. The duplicate tag store also provides a means to “selectively” accept updates to the B-Cache. This allows optimal system performance for accesses to truly shared locations; however protects the integrity of the write-back system by invalidating “shared cold” locations caused by phenomena such as process migration.

The general Update vs Invalidate algorithm is as follows when a System-bus write cycle occurs:

Example 1: Update vs Invalidate Algorithm

```
IF (Waiting in ARB for System-bus)
{
  IF (HIT B-Cache)
  {
    update_B-Cache_entry();
  }

  IF (HIT primary data cache)
  {
    invalidate_primary_data_cache_entry();
  }
}
ELSE
{
  IF (System-bus commander is a CPU OR ENB B-CACHE COND I/O UPDATES (BCC Reg)
  {
    IF (HIT primary data cache)
    {
      invalidate_primary_data_cache_entry();

      IF (HIT B-Cache)
      {
        update_B-Cache_entry();
      }
    }
  }
  ELSE
  {
    IF (HIT B-Cache)
    {
      invalidate_B-Cache_entry();
    }
  }
}
ELSE
{
  IF (HIT primary data cache)
  {
    invalidate_primary_data_cache_entry();
  }

  IF (HIT B-Cache)
  {
    update_B-Cache_entry();
  }
}
}
```

Two independent duplicate tag stores are used for DECchip 21064-A275 CPU modules to duplicate each DCache in the DECchip 21064-A275. The algorithm in Example 1 is used for each duplicate tag store.

4.3.1 Duplicate Tag Error Register - CSR5

CSR offset = A0₁₆

The Duplicate Tag Error Register is updated after each access to the duplicate tag store RAM in both slices. After a parity error is detected, the contents of the register are not updated while the error flag is set, the lost error flag does not inhibit error logging. The missed error bit in this register when set, suppresses further error interrupts from duplicate tag parity errors. When these parity errors are detected, invalidation of the primary and/or secondary caches result. Diagnostic Hint - filling the duplicate tag ram with bad parity causes the coherence policy to degenerate from update to invalidate.

NOTE

The system software shall ensure that the state of error flags in the low long-word are consistent with control flags in the upper longword by performing quadword writes to clear flags.

Figure 46: Duplicate Tag Error Register (DTER)

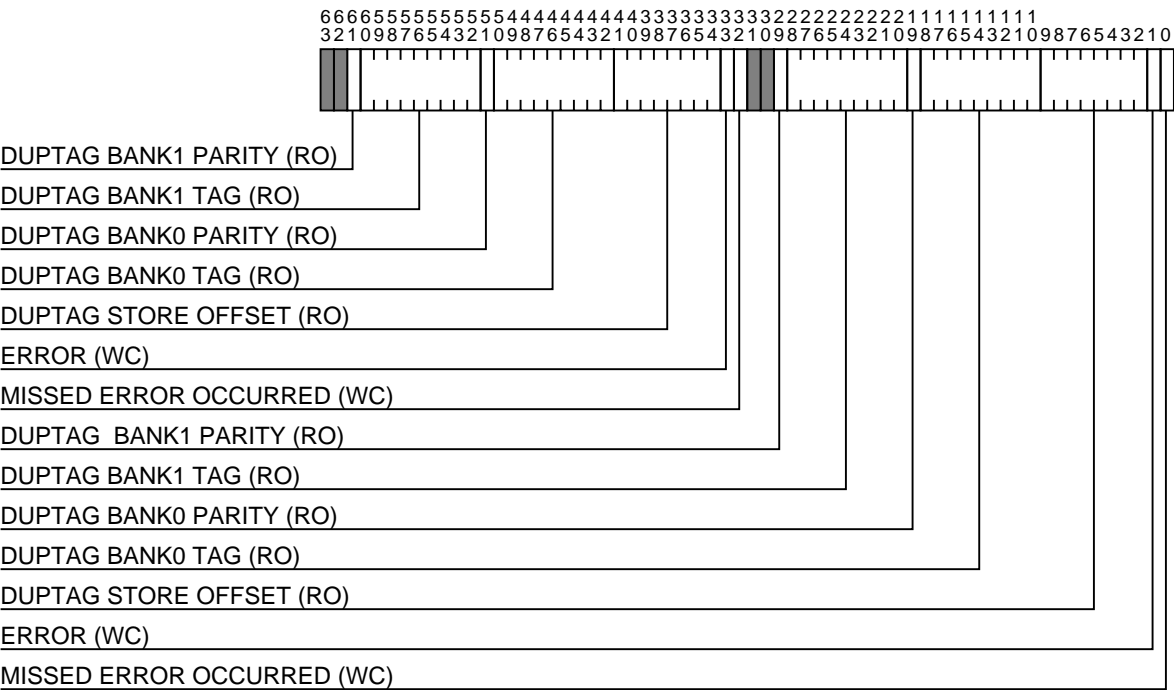


Table 35: Duplicate Tag Error Register Description	
Field	Description
61	DUPTAG BANK1 PARITY [read-only]

Table 35 (Cont.): Duplicate Tag Error Register Description

Field	Description
	Undefined on power-up. Contains the last P-Cache duplicate tag. If a parity error occurs in the P-Cache duplicate tag store, then the contents of this register will be frozen until the ERROR bits have been cleared. This field is only updated when memory space references occur while the register is not frozen.
60:52	DUPTAG BANK1 TAG [<i>read-only</i>]
51	DUPTAG BANK0 PARITY [<i>read-only</i>]
50:42	DUPTAG BANK0 TAG [<i>read-only</i>]
41:34	DUPTAG STORE OFFSET [<i>read-only</i>]
	Undefined on power-up. Contains the last Duplicate Tag Store Offset. If a parity error occurs in the duplicate tag store, then the DUP TAG STORE OFFSET will remain for that cycle until the ERROR bits have been cleared.
	This field is only updated when memory space references occur while the register is not frozen.
33	ERROR [<i>read/write 1 to clear</i>]
	Cleared on power-up. Set when, a Primary D-Cache duplicate tag store parity error has been detected to hold the contents of this register until this flag is cleared.
32	MISSED ERROR OCCURRED [<i>read/write 1 to clear</i>]
	This bit is set when a Primary D-Cache duplicate tag store parity error bit was set and another parity error is detected. When set, logging in this CSR is inhibited and further interrupts from this error are suppressed. This bit is cleared on power-up.
29	DUPTAG BANK1 PARITY [<i>read-only</i>]
28:20	DUPTAG BANK1 TAG [<i>read-only</i>]
19	DUPTAG BANK0 PARITY [<i>read-only</i>]
18:10	DUPTAG BANK0 TAG [<i>read-only</i>]
9:2	DUPTAG STORE OFFSET [<i>read-only</i>]
	Undefined on power-up. Contains the last Duplicate Tag Store Offset. If a parity error occurs in the duplicate tag store, then the DUP TAG STORE OFFSET will remain for that cycle until the ERROR bits have been cleared.
	This field is only updated when memory space references occur while the register is not frozen.
1	ERROR [<i>read/write 1 to clear</i>]
	This bit is set when a PRIMARY D-CACHE DUPLICATE TAG STORE PARITY ERROR is detected. The contents of this register are held until this error flag is cleared. Cleared on power-up.
0	MISSED ERROR OCCURRED [<i>read/write 1 to clear</i>]
	Cleared on power-up. Set when, a Primary D-Cache duplicate tag store parity error bit was set and another one is detected. When set, logging in this CSR is inhibited and further interrupts from this error are suppressed.

4.4 Lack of Duplicate Primary Instruction Cache Tag Store

Because the primary instruction cache is a virtual cache, all cache coherence is managed by the system software. Thus no hardware invalidates of the instruction cache will be performed. Also because very little writing to shared I-stream locations occur, the benefit of “selective” updating is diminished. As such there is **no** duplicate primary instruction cache tag store on the Cobra CPU module.

4.5 Lack of Cache Block Prefetch

Both *FETCH* and *FETCHM* instructions will be handled in the same manor. When either instruction is issued, the processor will be immediately released to continue executing.

4.6 Data Integrity

The CPU module provides error detection mechanisms for its major data storage elements and buses. Table 36 provides an overview of the error detection mechanisms provided.

Table 36: Data Integrity Reference

Element	Protection Method	Reference
Primary I-Cache Tag Store	Parity	Section C.8
Primary I-Cache Data Store	LW Parity	Section C.8
Primary D-Cache Tag Store	Parity	Section C.8
Primary D-Cache Data Store	LW Parity	Section C.8
B-Cache Control Store	Parity	Section 4.1.1,Section 8.2.1
B-Cache Tag Store	Parity	Section 4.1.2,Section 8.3
B-Cache Data Store	EDC†	Section 4.1.3,Section 8.2.2
Duplicate Tag Store	Parity	Section 4.3,Section 8.3
System-bus	LW Parity	Cobra System Bus Specification,Section 8.4
Processor Data Bus	EDC‡	Section 4.1.3,Section 8.2.2

†Single bit correction, double bit detection

‡Single bit correction, double bit detection, Depending on the source of the data, this could include errors that occur in the B-Cache data store as well as those occurring on the bus.

4.7 System-bus Interface

The System-bus interface is the CPU module's "window to the world". All standard data processed by a processor is obtained over the System-bus from either a Memory Module, the I/O Module, or another CPU Module. Refer to the system bus specification. All System-bus registers are visible to all System-bus commanders.

4.7.1 System-bus Control Register - CSR6

CSR offset = C0₁₆

The System-bus Control Register provides a means of controlling the System-bus arbitration and interface signals for diagnostic purposes, and contains the information required to support the WHOAMI requirement specified in the *Alpha Architecture Handbook*.

This register's arbitration control bits allows a System-bus commander to selectively disallow other commanders ownership of the System-bus and should only be used during system initialization; however it may also be useful to guard the System-bus from a broken CPU module. As the register can only be accessed via the System-bus, a commander will not be allowed to mask itself off as it would never be able to re-enable access to the System-bus.

NOTE

The CPU shall ensure that the state of control flags in the low longword are consistent with control flags in the upper longword by performing quadword writes.

Figure 47: System-bus Control Register (CBCTL)

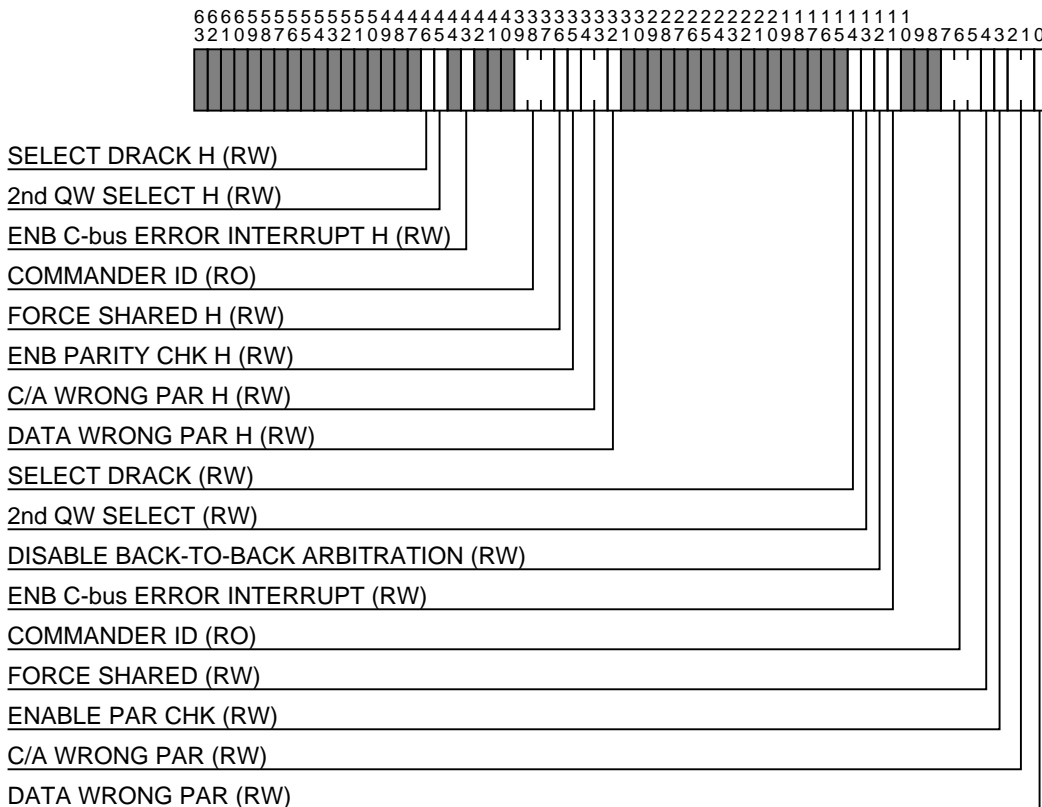


Table 37: System-bus Control Register Description

Field	Description
46	SELECT DRACK H [<i>read/write</i>] Cleared on power-up. This bit is reserved for timing configuration selection of the 21064 response strobe DRACK<1>. When set the DRACK<1> strobe timing is shifted by six nanoseconds. The state of this bit must match bit <14> in this register.
45	2nd QW SELECT H [<i>read/write</i>] Cleared on power-up. This bit is reserved for timing configuration selection of the second quad-word of returned data to the 21064. When set the data mux control strobe is shifted by six nanoseconds. The state of this bit must match bit <13> in this register.
43	ENB C-bus ERROR INTERRUPT H [<i>read/write</i>] Cleared on power-up. The state of this bit must match bit <11> in this register.
39:37	COMMANDER ID [<i>read-only</i>] Identifies the CPU as being CPU ₀ , CPU ₁ , CPU ₂ , or CPU ₃ based on the System-bus commander ID. Writes have no effect. Regardless of being CPU ₀ or CPU ₁ performing a read of this CSR, the contents of this field are returned with the CPU ID of the CPU commanding the CSR read. <div style="margin-left: 40px;"> 011 CPU₂ 010 CPU₁ 001 CPU₀ 000 CPU₃ </div>
36	FORCE SHARED H [<i>read/write</i>] Cleared on power-up. When set asserts CSHARED_L on all System-bus transactions.
35	ENB PARITY CHK H [<i>read/write</i>] Cleared on power-up. When set, longword parity checking on the System-bus for both the C/A and data portion of cycle (if responder) is enabled for C/A longwords 3 and 2 and data longwords 7,5,3 and 1.
34:33	C/A WRONG PAR H [<i>read/write</i>] Cleared on power-up. When set forces wrong parity on Longwords 3 and 1 respectively during the C/A portion of the next C/A cycle from this node to the System-bus. Once this cycle has occurred the C/A WRONG PAR bits are automatically cleared.
32	DATA WRONG PAR H [<i>read/write</i>] Cleared on power-up. When set forces wrong parity on longwords 7,5,3 and 1 during both data portions of the next System-bus cycle to which this node responds. Once this cycle has occurred the DATA WRONG PAR bit is automatically cleared. This should not be set when any of the C/A WRONG PAR bits are set as a responder is not able to log both wrong C/A parity and wrong data parity in the same System-bus transaction.
14	SELECT DRACK [<i>read/write</i>] Cleared on power-up. This bit is reserved for timing configuration selection of the 21064 response strobe DRACK<1>. When set the DRACK<1> strobe timing is shifted by six nanoseconds.
13	2nd QW SELECT [<i>read/write</i>] Cleared on power-up. This bit is reserved for timing configuration selection of the second quad-word of returned data to the 21064. When set the data mux control strobe is shifted by six nanoseconds.
12	DISABLE BACK-TO-BACK ARBITRATION [<i>read/write</i>] FOR DIAGNOSTICS ONLY. Cleared on power-up. This bit is reserved for diagnostic testing, when set the behavior of the C-bus arbiter is modified. This bit is used by CPU ₀ <i>only</i> .

Table 37 (Cont.): System-bus Control Register Description

Field	Description
11	ENB C-bus ERROR INTERRUPT [<i>read/write</i>] Cleared on power-up to disable the System-bus C_ERR_L interrupt signal from being driven due to errors encountered by this node. This bit does not disable reception of this interrupt signal.
7:5	COMMANDER ID [<i>read-only</i>] Identifies the CPU as being CPU ₀ , CPU ₁ , CPU ₂ , or CPU ₃ based on the System-bus commander ID. Writes have no effect. Regardless of being CPU ₀ or CPU ₁ performing a read of this CSR, the contents of this field are returned with the CPU ID of the CPU commanding the CSR read. 011 CPU ₂ 010 CPU ₁ 001 CPU ₀ 000 CPU ₃
4	FORCE SHARED [<i>read/write</i>] Cleared on power-up. When set asserts CSHARED_L on all System-bus transactions.
3	ENABLE PAR CHK [<i>read/write</i>] Cleared on power-up. When set, longword parity checking on the System-bus for both the C/A and data portion of cycle (if responder) is enabled for C/A longwords 1 and 0 and data longwords 2 and 0.
2:1	C/A WRONG PAR [<i>read/write</i>] Cleared on power-up. When set forces wrong parity on Longwords 2 and 0 respectively during the C/A portion of the next C/A cycle from this node to the System-bus. Once this cycle has occurred the C/A WRONG PAR bits are automatically cleared.
0	DATA WRONG PAR [<i>read/write</i>] Cleared on power-up. When set forces wrong parity on longwords 6,4,2 and 0 during both data portions of the next System-bus cycle to which this node responds. Once this cycle has occurred the DATA WRONG PAR bit is automatically cleared. This should not be set when any of the C/A WRONG PAR bits are set as a responder is not able to log both wrong C/A parity and wrong data parity in the same System-bus transaction.

4.7.2 System-bus Error Register - CSR7

CSR offset = E0₁₆

The System-bus Error Register is updated every System-bus cycle. Whenever a System-bus error is detected, the contents of this register are frozen until the ERROR bits are cleared. The contents of the register are not updated while the error flags are set, the lost error flags do not inhibit error logging.

NOTE

Even though the CBE register will be updated with the latest cycle on the System-bus, the contents of this register in the non-error case will always be the cycle that was issued to actually read the CBE register. This is because the register is accessed via a System-bus cycle.

If a tag control or tag store parity error is detected then the C/A NOT ACKED and/or the WRITE DATA NOT ACKED bits are set.

Figure 48: System-bus Error Register (CBE)

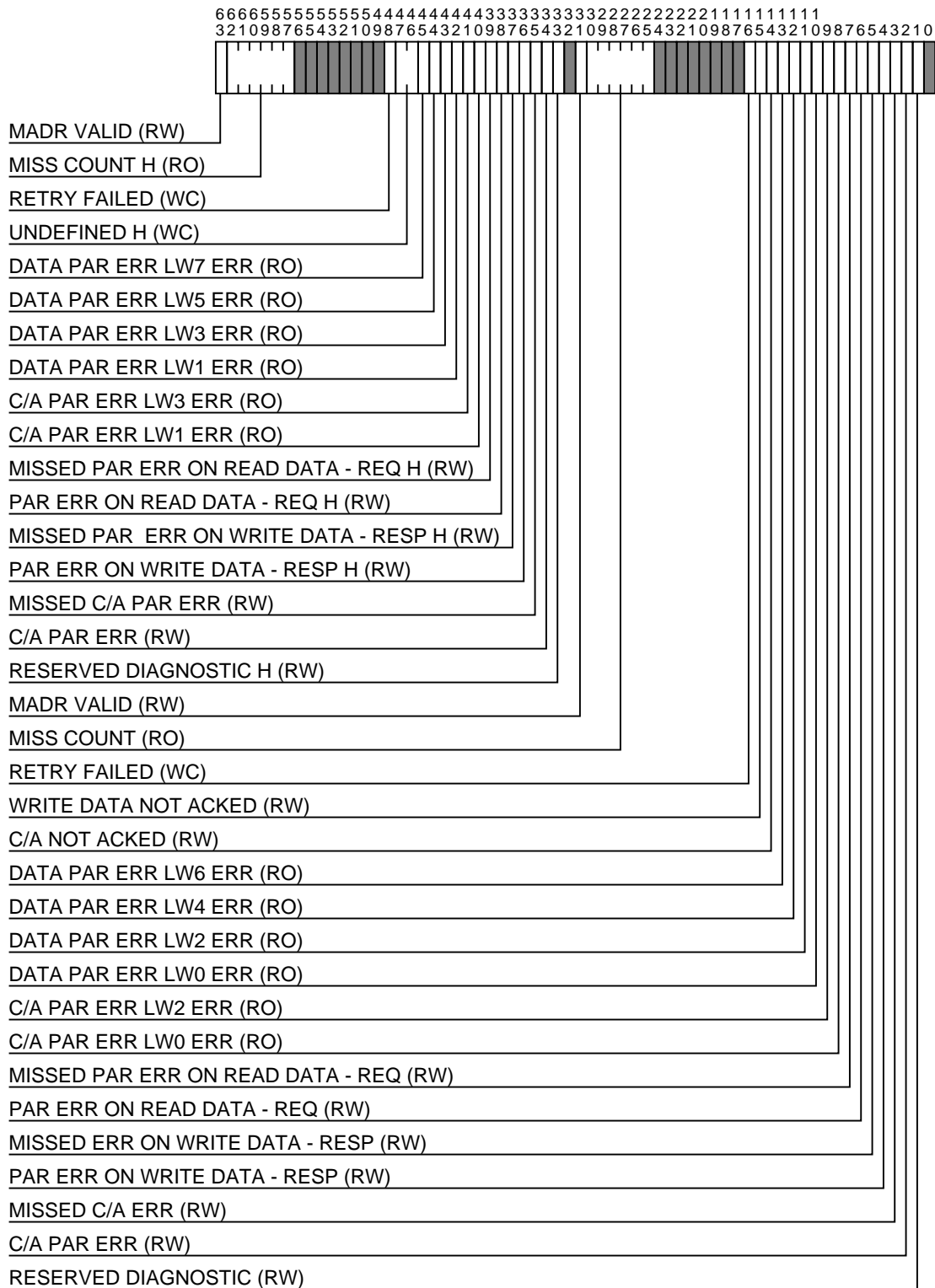


Table 38: System-bus Error Register Description

Field	Description
63	MADR VALID [<i>read/write</i>] Cleared on power-up. Has same function as bit 31, but indicates another valid sample in MADRL <63:32> that occurs 62 miss counts offset from the sample in the other half of these registers. Effectively providing two samples every 64 misses. A copy of this bit is readable from bit 0 of the MADRL register. Write "1" to clear.
62:57	MISS COUNT H [<i>read-only</i>] Set to 000000 on power-up. The six bit miss address counter is readable in this field.
48	RETRY FAILED [<i>read/write 1 to clear</i>] RETRY FAILED: (RW) Cleared on Power-up. Reading a "1" indicates that a CBUS retry was seen when not expected. This indicates that a CBUS retry signal was seen on the CBUS, but RETRY is not enabled in CSR15. When this occurs, the CPU command that was retried will acknowledge to EV with a CACK of HARD_ERR, and this bit will be set. The setting of this bit will NOT cause a CBUS C_ERRL or CPU interrupt to occur other than the HARD_ERR machine check to the CPU that failed.
47:46	UNDEFINED H [<i>read/write 1 to clear</i>] Cleared on power-up. Undefined usage.
45	DATA PAR ERR LW7 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a "1" indicates an error on data longword 7 System-bus bits <127:96>, during the second data cycle portion of a System-bus transaction. DATA PAR ERR LW7 ERR will remain valid until the Data ERR bits (38,36) are cleared.
44	DATA PAR ERR LW5 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a "1" indicates an error on data longword 5 System-bus bits <95:64>, during the second data cycle portion of a System-bus transaction. DATA PAR ERR LW5 ERR will remain valid until the Data ERR bits (<38> and <36>) are cleared.
43	DATA PAR ERR LW3 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a "1" indicates an error on data longword 3 System-bus bits <127:96>, during the first data cycle portion of a System-bus transaction. DATA PAR ERR LW3 ERR will remain valid until the Data ERR bits (38,36) are cleared.
42	DATA PAR ERR LW1 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a "1" indicates an error on data longword 1 System-bus bits <95:64>, during the first data cycle portion of a System-bus transaction. DATA PAR ERR LW1 ERR will remain valid until the Data ERR bits (38,36) are cleared.
41	C/A PAR ERR LW3 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a "1" indicates an error on command/address longword 3 System-bus bits <127:96>, during the C/A cycle portion of a System-bus transaction. C/A PAR ERR LW3 ERR will remain valid until the Data ERR bits (34) is cleared.
40	C/A PAR ERR LW1 ERR [<i>read-only</i>]

Table 38 (Cont.): System-bus Error Register Description

Field	Description
	Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a "1" indicates an error on command/address longword 1 System-bus bits <95:32>, during the C/A cycle portion of a System-bus transaction. C/A PAR ERR LW1 ERR will remain valid until the Data ERR bits (34) is cleared.
39	MISSED PAR ERR ON READ DATA - REQ H [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on returned read data longwords 3 or 1 as a commander and the error command and address could not be saved in the CBEAH register. Write 1 to clear.
38	PAR ERR ON READ DATA - REQ H [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on returned read data longwords 3 or 1 as a commander. Write "1" to clear.
37	MISSED PAR ERR ON WRITE DATA - RESP H [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on longwords 3 or 1 as a responder or bystander the error command and address could not be saved in the CBEAH register. Write "1" to clear.
36	PAR ERR ON WRITE DATA - RESP H [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on odd longwords 3 or 1 as a responder or a bystander in the case of accepting write data to update the B-Cache. Write "1" to clear. Only detects parity error on longwords 3 and 1.
35	MISSED C/A PAR ERR [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on the even C/A longwords (3,1) and the error command and address could not be saved in the CBEAH register. Write "1" to clear.
34	C/A PAR ERR [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on the even C/A longwords (3,1) regardless of the address or which node was the commander. A CPU node checks its own parity as a commander. Write "1" to clear.
33	RESERVED DIAGNOSTIC H [<i>read/write</i>] Cleared on power-up. Set as a result of an error event being communicated between the even and odd interface chips. This bit is for chip debug and should not be used by system software. The address is not held in CSR 8 and 9 when this bit is set. Write "1" to clear.
31	MADR VALID [<i>read/write</i>] Cleared on power-up. This bit will be set when the miss counter overflows and holds sampled miss address into the MADRL register. The miss address is sampled every 64th B-Cache miss by a free running miss transaction counter. When this bit is set MADRL <31:0> have valid miss contents. This bit being set inhibits capturing a new sample, but does not inhibit the counter from incrementing. A copy of this bit is readable from bit 0 of the MADRL register. Write "1" to clear.
30:25	MISS COUNT [<i>read-only</i>] Set to 111110 on power-up. The six bit miss address counter is readable in this field.
16	RETRY FAILED [<i>read/write 1 to clear</i>] RETRY FAILED: (RW) Cleared on Power-up. Reading a "1" indicates that a CBUS retry was seen when not expected. This indicates that a CBUS retry signal was seen on the CBUS, but RETRY is not enabled in CSR15. When this occurs, the CPU command that was retried will acknowledge to EV with a CACK of HARD_ERR, and this bit will be set. The setting of this bit will NOT cause a CBUS C_ERRL or CPU interrupt to occur other than the HARD_ERR machine check to the CPU that failed.

Table 38 (Cont.): System-bus Error Register Description

Field	Description
15	WRITE DATA NOT ACKED [<i>read/write</i>] Cleared on power-up. This bit will be set if either octaword portion of a System-bus write cycle generated by this commander was not acknowledged. The error address is logged in CBEAL and CBEAH, hence this error may cause subsequent lost errors. Write “1” to clear. This error could be flagged if a double bit error occurs in the tag store and the exchange address of the dirty victim places its address outside the physical address space of the currently configured system.
14	C/A NOT ACKED [<i>read/write</i>] Cleared on power-up. This bit will be set if the C/A portion of a System-bus cycle generated by this commander was not acknowledged. The error address is logged in fields associated with the lower 32 bits of the CBEAL and CBEAH. The upper 32 bits of this registers have no meaning when this error has been detected. This error may cause subsequent lost errors. Write “1” to clear.
13	DATA PAR ERR LW6 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a “1” indicates an error on data longword 6 System-bus bits <63:32>, during the second data cycle portion of a System-bus transaction. DATA PAR ERR LW6 ERR will remain valid until the Data ERR bits (6,4) are cleared.
12	DATA PAR ERR LW4 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a “1” indicates an error on data longword 4 System-bus bits <31:0>, during the second data cycle portion of a System-bus transaction. DATA PAR ERR LW4 ERR will remain valid until the Data ERR bits (6,4) are cleared.
11	DATA PAR ERR LW2 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a “1” indicates an error on data longword 2 System-bus bits <63:32>, during the first data cycle portion of a System-bus transaction. DATA PAR ERR LW2 ERR will remain valid until the Data ERR bits (6,4) are cleared.
10	DATA PAR ERR LW0 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a “1” indicates an error on data longword 0 System-bus bits <31:0>, during the first data cycle portion of a System-bus transaction. DATA PAR ERR LW0 ERR will remain valid until the Data ERR bits (6,4) are cleared.
9	C/A PAR ERR LW2 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a “1” indicates an error on command/address longword 2 System-bus bits <63:32>, during the C/A cycle portion of a System-bus transaction. C/A PAR ERR LW2 ERR will remain valid until the Data ERR bits (2) is cleared.
8	C/A PAR ERR LW0 ERR [<i>read-only</i>] Cleared on power-up. This bit is valid if a parity error is detected by this module, reading a “1” indicates an error on command/address longword 0 System-bus bits <31:0>, during the C/A cycle portion of a System-bus transaction. C/A PAR ERR LW0 ERR will remain valid until the Data ERR bits (2) is cleared.
7	MISSED PAR ERR ON READ DATA - REQ [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on returned read data longwords 2 or 0 as a commander and the error command and address could not be saved in the CBEAL register. Write “1” to clear.

Table 38 (Cont.): System-bus Error Register Description

Field	Description
6	PAR ERR ON READ DATA - REQ [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on returned read data longwords 2 or 0 as a commander. Write "1" to clear.
5	MISSED ERR ON WRITE DATA - RESP [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on longwords 2 or 0 as a responder or bystander or a write data cycle from this commander was not ack'ed and the error command and address could not be saved in the CBEAL register. Write "1" to clear.
4	PAR ERR ON WRITE DATA - RESP [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on even longwords 2 or 0 as a responder or a bystander in the case of accepting write data to update the B-Cache. Write "1" to clear. Only detects parity error on longwords 2 and 0.
3	MISSED C/A ERR [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on the even C/A longwords (2,0) or the C/A cycle was not ack'ed from this commander and the error command and address could not be saved in the CBEAL register. Write "1" to clear.
2	C/A PAR ERR [<i>read/write</i>] Cleared on power-up. Set when a parity error was detected on the even C/A longwords (2,0) regardless of the address or which node was the commander. A CPU node checks its own parity as a commander. Write "1" to clear.
1	RESERVED DIAGNOSTIC [<i>read/write</i>] Cleared on power-up. Set as a result of an error event being communicated between the even and odd interface chips. This bit is for chip debug and should not be used by system software. The address is not held in CSR 8 and 9 when this bit is set. Write "1" to clear.

4.7.3 System-bus Error Address Low Register - CSR8

CSR offset = 100₁₆

The System-bus Error Address Low Register is updated by this nodes commander transactions or by System-bus errors and contains the actual data found on the System-bus <63:0> during the latest C/A cycle. Whenever a System-bus error is detected and logged in the CBE register, the contents of this register are frozen until all of the error indications, not the missed error indications in the CBE Register are cleared.

For System-bus command/address cycles that are not acknowledged, the failing address is latched only in the error logging bits <31:0>. Bits <63:34> do not contain valid information when this type of error is logged.

Figure 49: System-bus Error Address Low Register (CBEAL)

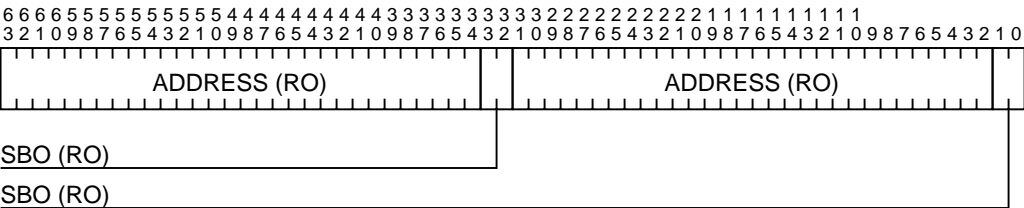


Table 39: System-bus Error Address Low Register Description

Field	Description
63:34	ADDRESS [<i>read-only</i>] Undefined on power-up. Address field <33:4>, from CAD<63:34>.
33:32	SBO [<i>read-only</i>] These bits should be ones.
31:2	ADDRESS [<i>read-only</i>] Undefined on power-up. Address field <33:4>, from CAD<31:2>.
1:0	SBO [<i>read-only</i>] These bits should be ones.

4.7.4 System-bus Error Address High Register - CSR9

CSR offset = 120₁₆

The System-bus Error Address High Register is updated by this nodes commander transactions or by System-bus errors and contains the actual data found on the System-bus <127:64> during the latest C/A cycle. Whenever a System-bus error is detected and logged in the CBE register, the contents of this register are frozen until all of the errors, not missed error, indications in the CBE Register are cleared.

For System-bus command/address cycles that are not acknowledged, the failing address is latched only in the error logging bits <31:0>. Bits <63:34> do not contain valid information when this type of error is logged.

Figure 50: System-bus Error Address High Register (CBEAH)

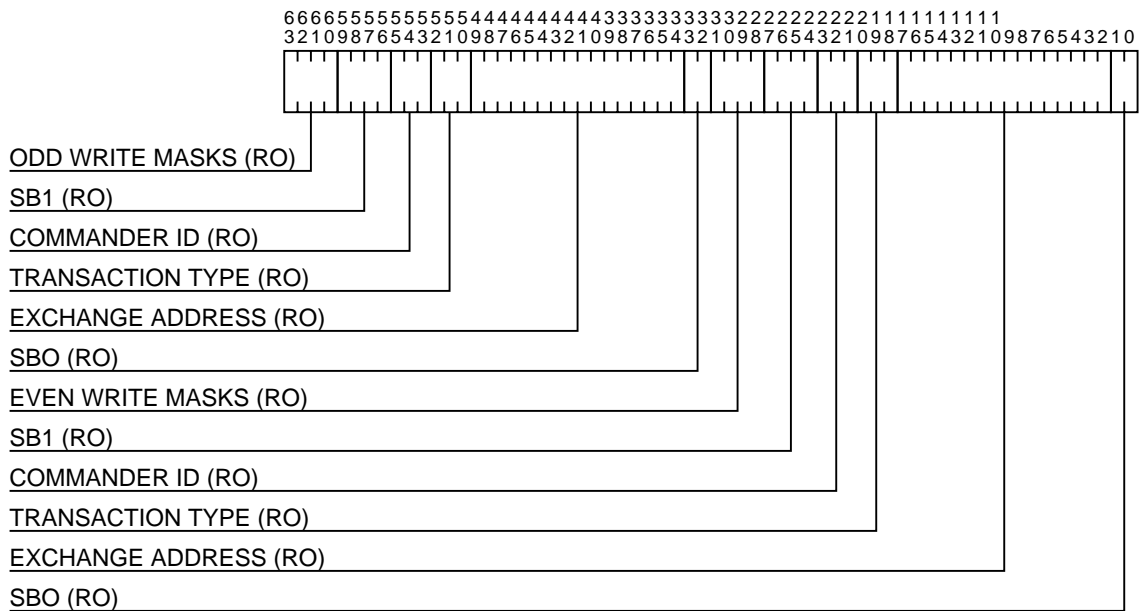


Table 40: System-bus Error Address High Register Description

Field	Description
63:60	ODD WRITE MASKS [<i>read-only</i>]

Table 40 (Cont.): System-bus Error Address High Register Description

Field	Description									
	<p>Odd Write Masks: (RO) This field contains the odd longword write MASKS bits used in the EV transaction that caused this CBUS cycle. Specifically:</p> <ul style="list-style-type: none">• CBEAH<63> = EV Write Mask bit 7• CBEAH<62> = EV Write Mask bit 5• CBEAH<61> = EV Write Mask bit 3• CBEAH<60> = EV Write Mask bit 1 <p>NOTE: The CPU uses the MASKS bits to communicate different data during READ cycles, and this data is related over these CBUS signals as well.</p>									
59:56	<p>SB1 [<i>read-only</i>]</p> <p>Undefined on power-up. These bits should be ones.</p>									
55:53	<p>COMMANDER ID [<i>read-only</i>]</p> <p>Undefined on power-up. Contains the commander identification code. The field is encoded as:</p> <table><tr><th>CID Module</th></tr><tr><td>000 - CPU₃</td></tr><tr><td>001 - CPU₀</td></tr><tr><td>010 - CPU₁</td></tr><tr><td>011 - CPU₂</td></tr><tr><td>100 - I/O 0 (Standard I/O)</td></tr><tr><td>101 - I/O 1 (expansion I/O)</td></tr><tr><td>110 - RESERVED I/O</td></tr><tr><td>111 - RESERVED I/O</td></tr></table>	CID Module	000 - CPU ₃	001 - CPU ₀	010 - CPU ₁	011 - CPU ₂	100 - I/O 0 (Standard I/O)	101 - I/O 1 (expansion I/O)	110 - RESERVED I/O	111 - RESERVED I/O
CID Module										
000 - CPU ₃										
001 - CPU ₀										
010 - CPU ₁										
011 - CPU ₂										
100 - I/O 0 (Standard I/O)										
101 - I/O 1 (expansion I/O)										
110 - RESERVED I/O										
111 - RESERVED I/O										
52:50	<p>TRANSACTION TYPE [<i>read-only</i>]</p> <p>Undefined on power-up. The commander transaction request. The field is encoded as:</p> <table><tr><td>000 - READ</td></tr><tr><td>001 - RESERVED</td></tr><tr><td>010 - EXCHANGE</td></tr><tr><td>011 - RESERVED</td></tr><tr><td>100 - WRITE</td></tr><tr><td>101 - RESERVED</td></tr><tr><td>110 - RESERVED</td></tr><tr><td>111 - NUT</td></tr></table>	000 - READ	001 - RESERVED	010 - EXCHANGE	011 - RESERVED	100 - WRITE	101 - RESERVED	110 - RESERVED	111 - NUT	
000 - READ										
001 - RESERVED										
010 - EXCHANGE										
011 - RESERVED										
100 - WRITE										
101 - RESERVED										
110 - RESERVED										
111 - NUT										
49:34	<p>EXCHANGE ADDRESS [<i>read-only</i>]</p> <p>Undefined on power-up. Contains the tag, of the victimized cache location. Refer to the Cobra System Bus Specification for further details.</p>									
33:32	<p>SBO [<i>read-only</i>]</p> <p>These bits should be ones.</p>									
31:28	<p>EVEN WRITE MASKS [<i>read-only</i>]</p> <p>Even Write Masks: (RO) This field contains the even longword write MASKS bits used in the EV transaction that caused this CBUS cycle. Specifically:</p> <ul style="list-style-type: none">• CBEAH<31> = EV Write Mask bit 6• CBEAH<30> = EV Write Mask bit 4• CBEAH<29> = EV Write Mask bit 2• CBEAH<28> = EV Write Mask bit 0									

Table 40 (Cont.): System-bus Error Address High Register Description

Field	Description
27:24	SB1 [<i>read-only</i>] These bits should be ones.
23:21	COMMANDER ID [<i>read-only</i>] Undefined on power-up. Contains the commander identification code. The field is encoded as: 000 - CPU ₃ 001 - CPU ₀ 010 - CPU ₁ 011 - CPU ₂ 100 - I/O 0 (Standard I/O) 101 - I/O 1 (expansion I/O) 110 - RESERVED I/O 111 - RESERVED I/O
20:18	TRANSACTION TYPE [<i>read-only</i>] Undefined on power-up. The commander transaction request. The field is encoded as: 000 - READ 001 - RESERVED 010 - EXCHANGE 011 - RESERVED 100 - WRITE 101 - RESERVED 110 - RESERVED 111 - NUT
17:2	EXCHANGE ADDRESS [<i>read-only</i>] Undefined on power-up. Contains the tag, of the victimized cache location. Refer to the Cobra System Bus Specification for further details.
1:0	SBO [<i>read-only</i>] These bits should be ones.

4.8 Multiprocessor Configuration CSR Definitions

4.8.1 Processor Mailbox Register - CSR10

CSR offset = 140₁₆

The Processor Mailbox Register allows any System-bus commander to communicate with any other System-bus commander (implementing the Processor Mailbox Register) without the need for any memory subsystem. This is primarily intended to be used during the initialization of the system and system exception processing.

Figure 51: Processor Mailbox Register (PMBX)

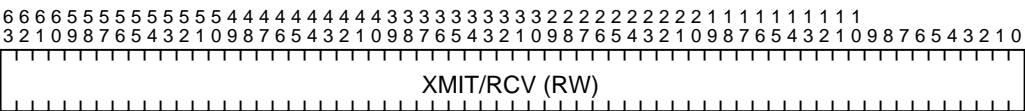


Table 41: Processor Mailbox Register Description

Field	Description
63:0	XMIT/RCV [<i>read/write</i>] Cleared on power-up. The protocol used in communication between the two processors is completely under software control.

4.8.2 Interprocessor Interrupt Request Register - CSR11

CSR offset = 160₁₆

Each CPU will have an Interprocessor Interrupt Request Register to support the interprocessor interrupt IPR specified in the *Alpha Architecture Handbook*. As these are System-bus visible, any System-bus commander including the owner can post an interprocessor interrupt to any CPU. The interrupt is driven into the processor signal identified internally as HIRR(3). In addition, a high IPL interrupt to each local processor can be requested for halting a node by writing to the REQUEST_NODE_HALT field in this register. Refer to the 21064 Processor Specification and Chapter 7.

Figure 52: Interprocessor Interrupt Request Register (IPIR)

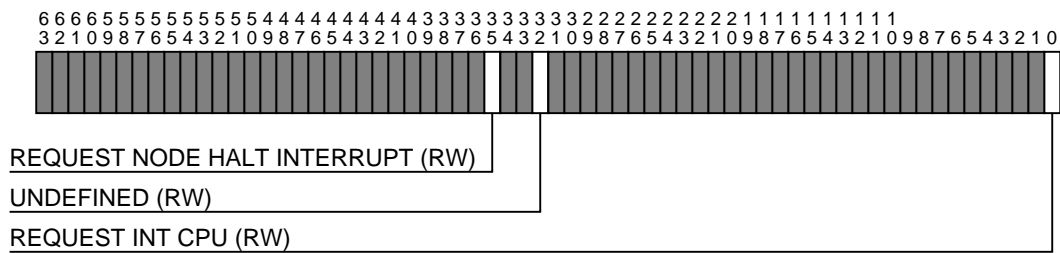


Table 42: Interprocessor Interrupt Request Register Description

Field	Description
35	REQUEST NODE HALT INTERRUPT [read/write] Cleared on power-up. Writing a "1" to this bit will cause a hardware interrupt similar to sys-event, except for a single CPU only. Reading this register returns the state of the requested node halt interrupt signal. Cleared by writing a "1" to the NODE HALT INT CLEAR bit in CSR12 (bit <35>).
32	UNDEFINED [read/write] Undefined
0	REQUEST INT CPU [read/write] Cleared on power-up. Writing a "1" will cause a hardware interrupt to be posted to the CPU defined by the address of the CSR. Reading this register returns the state of the interrupt request signal. Cleared by writing a '0'.

4.9 System Interrupt Clear Register - CSR12

CSR offset = 180₁₆

The System Interrupt Clear Register provides a path for the CPU to clear the edge triggered interrupts from the System-bus C_ERR_L signal, the SYS_EVENT_L signal, and the interval timer interrupt, CINT_TIM signal. The C_ERR_L and SYS_EVENT_L signals are broadcast to both CPU modules. The interval timer clock is received from the System-bus and used to generate a Interval Timer Interrupt to each processor. The Interval Timer Interrupt is local to each CPU so that this interrupt will occur 90 degrees out of phase with the other processor node. The system generic SYSTEM EVENT INTERRUPT is generated by an I/O halt request, a operator control panel halt request, or an enclosure event, or a power supply event. The generic transaction error signal C_ERR_L is generated by soft or hard errors related to data transactions.

Node halt interrupt is generated by writing to the REQUEST_NODE_HALT_INT bit (<33>) in CSR11 (Interprocessor Interrupt Register). When asserted, this interrupt is driven to the local CPU via the SYS_EVENT_L signal. Software must read this register to determine whether a Cbus sys event or node halt was the initiator of the interrupt.)

Figure 53: System Interrupt Clear Register (SIC)

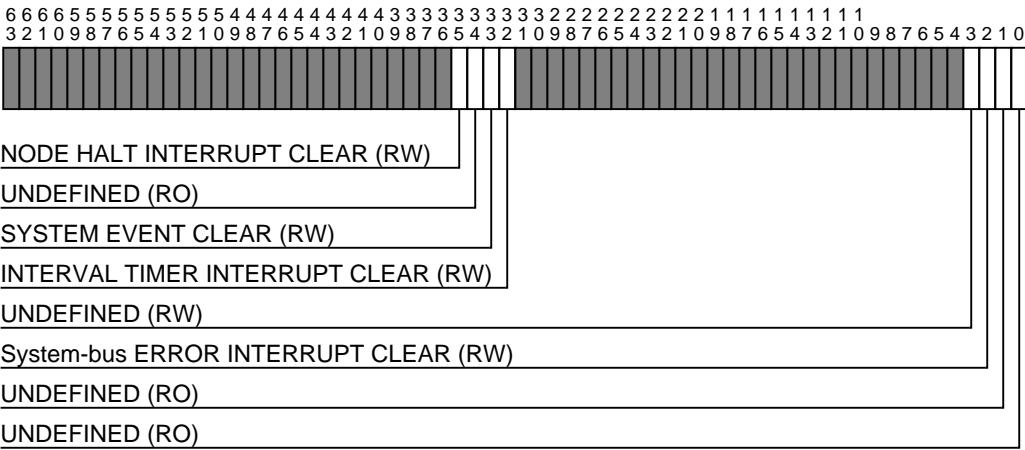


Table 43: System Interrupt Clear Register Description

Field	Description
35	NODE HALT INTERRUPT CLEAR [read/write] Cleared on power-up. When read, this bit reeturns the state of the interrupt signal to the procesor. A write to this register with a "1" in this bit position will clesar the latched node halt interrupt bit and the sys_event signal driven to the local CPU.

Table 43 (Cont.): System Interrupt Clear Register Description

Field	Description
34	UNDEFINED [<i>read-only</i>] Undefined
33	SYSTEM EVENT CLEAR [<i>read/write</i>] Cleared on power-up. A read of this register returns the state of the request signal to the processor. A write to this register with a “1” in this bit position will clear the latched CSYS_EVENT_L interrupt signal driven to the local CPU. This interrupt can be masked in the 21064 chip.
32	INTERVAL TIMER INTERRUPT CLEAR [<i>read/write</i>] Cleared on power-up. A read of this register returns the state of the interrupt signal to the processor. A write to this register with a “1” in this bit position will clear the latched CINT_TIM interrupt signal driven to the local CPU. This interrupt can be masked in the 21064 chip.
3	UNDEFINED [<i>read/write</i>]
2	System-bus ERROR INTERRUPT CLEAR [<i>read/write</i>] Cleared on power-up. A read of this register returns the state of the interrupt signal to the processor. A STQ to this register with a “1” in this bit position will clear the latched C_ERR_L interrupt signal driven to the local CPU. This interrupt can be masked in the 21064 chip. Interrupts generated by errors detected by this node may be disabled via bit <43,11> in CSR6.
1	UNDEFINED [<i>read-only</i>] Undefined
0	UNDEFINED [<i>read-only</i>] Undefined

4.10 Address Lock Register - CSR13

CSR offset = 1A0₁₆

The Address Lock Register is required by the *Alpha Architecture Handbook* to support the LDxL and STxC instructions. This is supported on Cobra in “memory like” regions only (Address <33> is “0”). This register will latch the address and set the Lock Address Valid bit when a LDxL instruction to memory address space is executed. The Lock Address Valid bit will be cleared when a STxC to any location or a System-bus write to the locked location occurs even if the write is from this node, or by explicitly clearing it by writing bit <0>. If the Lock Address Valid bit is set, and a LDxL to a different location occurs, the contents of the Lock Address field will be updated with the new address.

The resolution of the address lock in the system is a single aligned 32 byte block.

NOTE

The system software shall ensure that the state of the lock VALID flag in the low longword is consistent with the lock flag in the upper longword by performing quadword writes.

The lock flag must always be cleared before returning from PALmode.

Figure 54: Address Lock Register (ADLK)

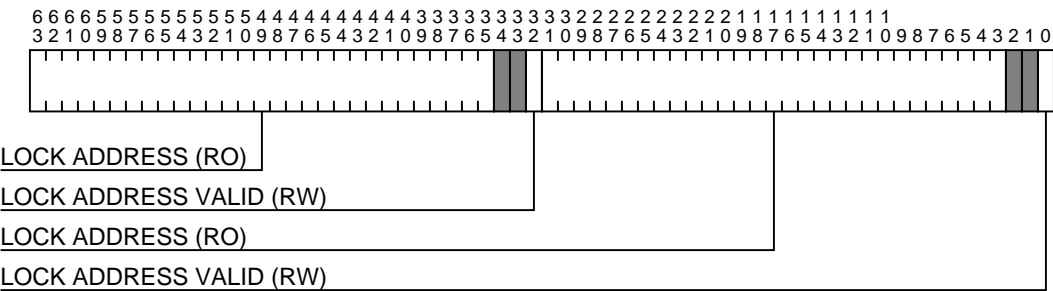


Table 44: Address Lock Register Description

Field	Description
63:35	LOCK ADDRESS [<i>read-only</i>] Set on power-up. A LDxL instruction to memory address space from this nodes processor will cause the contents of this register to be updated with the lock address. The contents are invalid when the LOCK ADDRESS VALID bit is clear. Bit 63 will always read as zero.
32	LOCK ADDRESS VALID [<i>read/write</i>] When set this bit indicates that the LOCK ADDRESS field is valid and a STxC will succeed. This bit is cleared on power-up, by a write from the System-bus to the locked location, by a STxC from the processor, or by writing a “1” to this bit.
31:3	LOCK ADDRESS [<i>read-only</i>]

Table 44 (Cont.): Address Lock Register Description

Field	Description
	Set on power-up. A LDxL instruction to memory address space from this nodes processor will cause the contents of this register to be updated with the lock address. The contents are invalid when the LOCK ADDRESS VALID bit is clear. Bit 31 will always read as zero.
0	LOCK ADDRESS VALID [<i>read/write</i>] When set this bit indicates that the LOCK ADDRESS field is valid and a STxC will succeed. This bit is cleared on power-up, by a write from the System-bus to the locked location, by a STxC from the processor to memory address space, or by writing a "1" to this bit.

4.11 Miss Address Register - CSR14

CSR offset = 1C0₁₆

The Miss Address Register captures the System-bus read or exchange address on every miss with address bit <33> not set, and holds the 64th sample until a sample valid flag is cleared. The read or exchange may have resulted from a 21064 read or write transaction. The latching strobe is skewed by 62 counts between the low and high longwords of CSR14, or 2 counts from high to low longwords.

The counters used for sampling are free running and no reset of the counter occurs when the MADR VALID flag in CSR7 is cleared. This means that the next sample frozen after clearing this flag could occur between 1 and 64 transactions later.

Figure 55: Miss Address Register Low (MADRL)

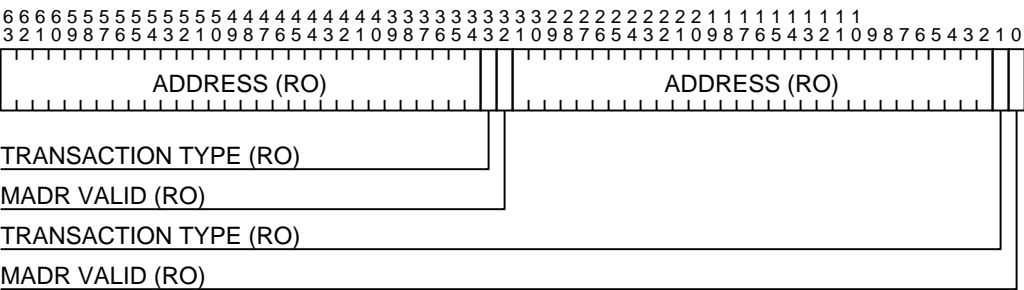
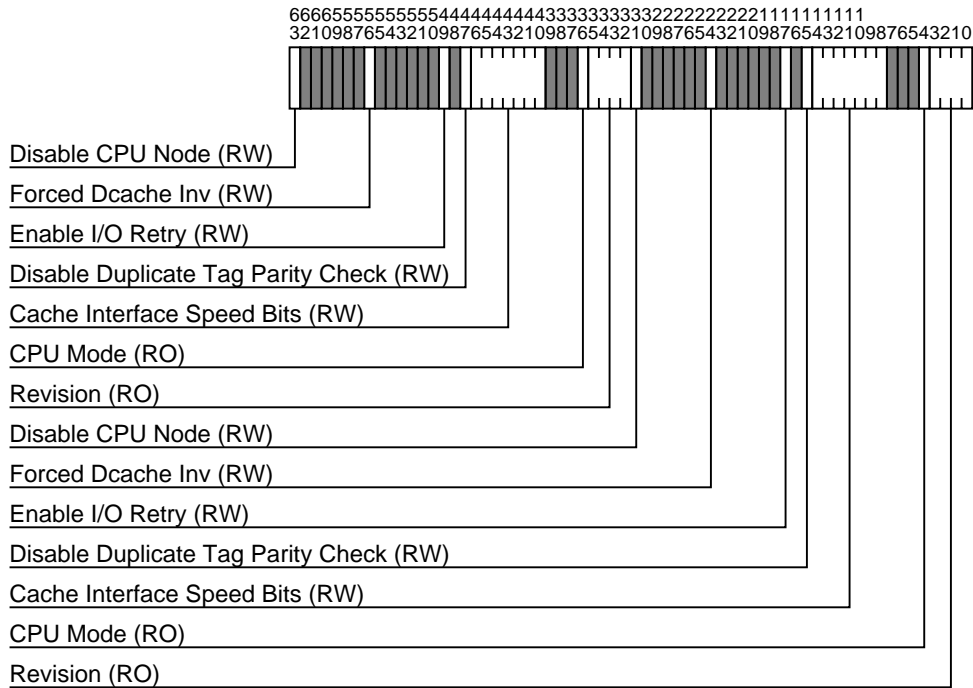


Table 45: Miss Address Register Low Description

Field	Description
63:34	ADDRESS [<i>read-only</i>] Undefined on power-up. Address field <33:4>, from CAD<63:34>.
33	TRANSACTION TYPE [<i>read-only</i>] Undefined on power-up. The commander transaction type, set indicates read cleared indicates exchange.
32	MADR VALID [<i>read-only</i>] Clear on power-up. When set this bit indicates that this sample is valid. This bit is a read-only copy of csr7 bit 63.
31:2	ADDRESS [<i>read-only</i>] Undefined on power-up. Address field <33:4>, from CAD<31:2>.
1	TRANSACTION TYPE [<i>read-only</i>] Undefined on power-up. The commander transaction type, set indicates read cleared indicates exchange.
0	MADR VALID [<i>read-only</i>] Cleared on power-up. When set this bit indicates that this sample is valid. This bit is a read-only copy of csr7 bit 31.

4.11.1 C4 Revision Register - CSR15**CSR offset = 1E0₁₆****Figure 56: C4 Revision Register (CRR)****Table 46: C4 Revision Register Description**

Field	Description
63	Disable CPU Node [read/write] The state of this bit must match bit <31>.
56	Forced Dcache Inv [read/write] The state of this bit must match bit <24>.
49	Enable I/O Retry [read/write] The state of this bit must match bit <17>.
47	Disable Duplicate Tag Parity Check [read/write] Cleared on power-up. When set, disables parity checking logic for the duplicate tag in the bus interface ASICs. When clear, duplicate tag parity checking may be enabled or disabled as programmed by CSR0 bit 34. This bit only controls the ODD slice ASIC.
46:40	Cache Interface Speed Bits [read/write] The state of these bits must match the bits in field <14:8>.
36	CPU Mode [read-only]

Table 46 (Cont.): C4 Revision Register Description

Field	Description
35:32	Revision [<i>read-only</i>] The state of these bits must match bits <3:0>
31	Disable CPU Node [<i>read/write</i>] Cleared on power-up. When set, this bit disables a CPU from probing the backup cache and participating in system bus transactions. The state of this bit must match bit <63>.
24	Forced Dcache Inv [<i>read/write</i>] Cleared on power-up. When set forces all CBUS writes to invalidate the processors d-cache regardless of whether it hits in the duplicate tag. This is a feature for diagnostics. The state of this bit must match bit <56>
17	Enable I/O Retry [<i>read/write</i>] Cleared on power-up. When this bit is set, read and writes to I/O space will be re-tried if signaled by the I/O module. If clear, the retry signaling mechanism is treated as an error and logged in CSR-7 (CBE). The state of this bit must match bit<49>
15	Disable Duplicate Tag Parity Check [<i>read/write</i>] Cleared on power-up. When set, disables parity checking logic for the duplicate tag in the interface ASICS. When clear, duplicate tag parity checking may be enabled or disabled as programmed by CSR0 bit 2. This bit only controls the EVEN slice asic.
14:8	Cache Interface Speed Bits [<i>read/write</i>] Cleared on power-up. When set, each bit corresponds to a control signal which can be slightly shifted in time, thereby allowing greater flexibility in operating speed combinations between processor and system bus. The state of this bit must match bit <46:40>
4	CPU Mode [<i>read-only</i>] Indicates the pin bus mode for which the ASIC has autoconfigured. This is a read-only field. <ul style="list-style-type: none"> • 0 - DECchip 21064 Mode. • 1 - DECchip 21064-A275 Mode.
3:0	Revision [<i>read-only</i>] Indicates the revision level of the bus interface (C ³ or C4) device. <ul style="list-style-type: none"> • 1000₂ - rev C4 (This is the first revision to support 4-CPU's.) • 1001₂ - rev C4-b. • 1010₂ - rev C4-c. (support for expansion I/O.)

Table 47: CSR Space

CSR	Mnemonic	CPU ₀	CPU ₁	CPU ₂	CPU ₃
0	BCC	3.8000.0000	3.8100.0000	3.8200.0000	3.8300.0000
1	BCCE	3.8000.0020	3.8100.0020	3.8200.0020	3.8300.0020
2	BCCEA	3.8000.0040	3.8100.0040	3.8200.0040	3.8300.0040
3	BCUE	3.8000.0060	3.8100.0060	3.8200.0060	3.8300.0060
4	BCUEA	3.8000.0080	3.8100.0080	3.8200.0080	3.8300.0080
5	DTER	3.8000.00A0	3.8100.00A0	3.8200.00A0	3.8300.00A0
6	CBCTL	3.8000.00C0	3.8100.00C0	3.8200.00C0	3.8300.00C0
7	CBE	3.8000.00E0	3.8100.00E0	3.8200.00E0	3.8300.00E0
8	CBEAL	3.8000.0100	3.8100.0100	3.8200.0100	3.8300.0100
9	CBEAH	3.8000.0120	3.8100.0120	3.8200.0120	3.8300.0120
10	PMBX	3.8000.0140	3.8100.0140	3.8200.0140	3.8300.0140
11	IPIR	3.8000.0160	3.8100.0160	3.8200.0160	3.8300.0160
12	SIC	3.8000.0180	3.8100.0180	3.8200.0180	3.8300.0180
13	ADLK	3.8000.01A0	3.8100.01A0	3.8200.01A0	3.8300.01A0
14	MADRL	3.8000.01C0	3.8100.01C0	3.8200.01C0	3.8300.01C0
15	CRR	3.8000.01E0	3.8100.01E0	3.8200.01E0	3.8300.01E0

4.12 Interval Timer

The Sable CPU module receives the signal CINT_TIM H off the system bus and generates an interval timer interrupt to the processor. The interface gate arrays require that the source of the CINT_TIM H have a duty cycle of approximately 50%, and that the frequency of CINT_TIM H be twice that of the required interval timer interrupt passed to the processor.

Interval timer interrupts are intentionally staggered in a multiprocessing system so that the different processors see the interval timer interrupt at different times. The following table shows the order that interrupts are delivered to the different CPU modules:

Table 48: Interval Timer Interrupt Generation

CINT_TIM H	Interrupted CPU
rising edge	CPU 0
falling edge	CPU 1
rising edge	CPU 2
falling edge	CPU 3
rising edge	CPU 0
falling edge	CPU 1

Table 48 (Cont.): Interval Timer Interrupt Generation

CINT_TIM H	Interrupted CPU
etc...	

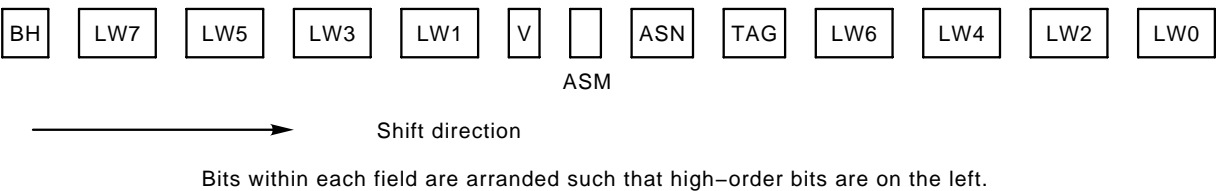
Note that for an interval timer frequency of 976.5625 microseconds, the source should actually be programmed for twice that frequency.

4.13 D-bus

A serial ROM of size 16KB is located on the CPU module and is used to supply the processor with its power-up code. This ROM pattern is loaded into 8K of the 21064 processor's internal ICache under the control of the processor after power-up reset **only**.

The data loaded consists of each cache blocks' tag, ASN, ASM, valid, branch history, and 8 longwords of data. This data is loaded in sequential order starting with block zero and ending with block 255. The order in which bits within each block are serially loaded is shown in Figure 57.

Figure 57: The 21064 Serial Load Data Format



The table also shows the values which must be loaded into each cache block's tag and control bits.

An extension serial ROM of up to 16KB will be automatically multiplexed onto the D-bus after the initial 16KB is read into the processor. The software system is able to load this ROM into the data cache of the processor (or the B-Cache to execute) it by alternately toggling the TMT bit in the SL_XMIT register of the processor, and reading the RCV bit of the processors SL_RCV register. There is no need to read the unused portion of the initial serial ROM under program control. The extension serial ROM can be read immediately. The first extension ROM data bit is present in the SL_RCV register 500ns after the completion of the initial serial ROM load and then 500ns after every subsequent 0 to 1 transition of the TMT bit in the SL_XMIT register. The TMT bit should not be toggled from 1 to 0 nor 0 to 1 with a delay smaller than 500ns between transitions. This ROM can only be read once, after a hard system reset.

As initialization proceeds the microcontroller (87C652) monitors the progress of the 21064 processor as it reads each serial ROM. This is done by observing the state of the serial ROM read status signals. When all ROMs have been read, or an error has been detected, the micro has the ability to communicate with the 21064 over the 21064 Serial Line Interface.

To monitor the bit by bit progress of the 21064 reading the serial ROM's, the D-bus micro can drive a 1 on the 21064/Serial Line Output and enable communication with the 21064 by clearing the "Disable Communications with the 21064" bit. By monitoring the 21064/Serial Line Input the D-bus micro can detect each clock sent to the serial ROM devices.

Refer to Table 49 for mapping of D-bus microcontroller ports to module functions.

Table 49: D-bus Micro Port Mapping

Port	Port Type	Init State	Functional Description
P3.5	I	—	Interval Timer Interrupt Request
P3.4	I	—	Unused Timer
P3.3	I	—	Interval Timer Interrupt Request
P3.2	I	—	Serial ROM Read Interrupt; when set EV I-cache initial serial ROM load not complete
P3.1	O	1	21064/Serial Line Output - only active when P2.7 is cleared
P3.0	I	—	21064/Serial Line Input - only active when P2.7 is cleared
P2.7	O	1	Disable Communication with the 21064
P2.6	I	—	Unused; pulled high ¹
P2.5	I	—	CPU_ID1
P2.4	I	—	CPU_ID0; 00 = 0; 01 = 1; 10 =2; 11=3
P2.3	I	—	Serial ROM 2 Read Status; when set supplemental serial ROM 2 not READ; as the processor will never completely read this ROM this be will never be cleared.
P2.2	I	—	Serial ROM 1 Read Status; when set the 21064 I-cache initial serial ROM load not completed.
P2.1	I	—	Unused; tied to GND ¹
P2.0	I	—	The 21064 Processor Resting due to 3.3V under voltage or Oscillator not running.
P1.7	B	—	Serial Command Bus Data Line
P1.6	B	—	Serial Command Bus Clock Line

¹These values could be sampled by the micro to distinguish from a group-0 CPU board. The values specified here are for are for non-group0 CPU modules.

The D-bus microcontroller's input oscillator is:

Table 50: D-bus Microcontroller Clock Frequency

System-bus OSC (MHz)	System-bus Period (ns)	Micro OSC (MHz)
333†	24.0	10.41667
320	25.0	10.00000
303	26.4	9.46970

†This is the shipping frequency for Sable systems.

Table 51: D-bus Microcontroller System Control Bus Address

Transaction	CPU ₀ Address	CPU ₁ Address	CPU ₂ Address	CPU ₃ Address
Read	B1 (HEX)	B3 (HEX)	BB (HEX)	BD (HEX)
Write	B0 (HEX)	B2 (HEX)	BA (HEX)	BC (HEX)

4.14 System-bus Arbiter

The System-bus arbiter supports six arbitrated nodes, (four CPU modules, and two I/O modules). The arbiter is located on the CPU₀ module. Arbitration is a prioritized Round-Robin. I/O is the highest priority, and both CPU's a lesser but equal priority. This provides the arbitration characteristic of CPU₀, I/O₀, CPU₁, I/O₁, CPU₃, I/O₀, etc. If only one CPU is in the system, the arbitration algorithm becomes a standard Round-Robin. These arbitration characteristics occur only when the System-bus is saturated.

When the System-bus is saturated with traffic, in order to avoid starvation of the 21064 from its B-Cache, a single dead System-bus cycle (nom. 24 ns) will be inserted every third transaction.

Arbitration cycles occur in parallel with data transfer cycles to minimize arbitration overhead. During the normal arbitration mode of operation, the arbiter grants the bus to the Cobra-bus commanders as per the following algorithm.

1. The arbitration is round-robin among the processor type commander nodes. Among the processor nodes the priority is determined in the higher order of processor-node-id since the last selection of the processor node. The FAST supports upto four processor nodes with processor-node-id of CPU₀, CPU₁, CPU₂, and CPU₃.
2. The arbitration is round-robin among the IO type commander nodes. Among the IO nodes the priority is determined in the higher number of IO-NODE-ID since the last selection of the IO node. The FAST supports up to two IO nodes with IO-NODE-ID of IO_{0(T2)} and IO_{1(XIO)}.
3. The arbitration is alternating between the processor type commander nodes and IO type commander nodes.

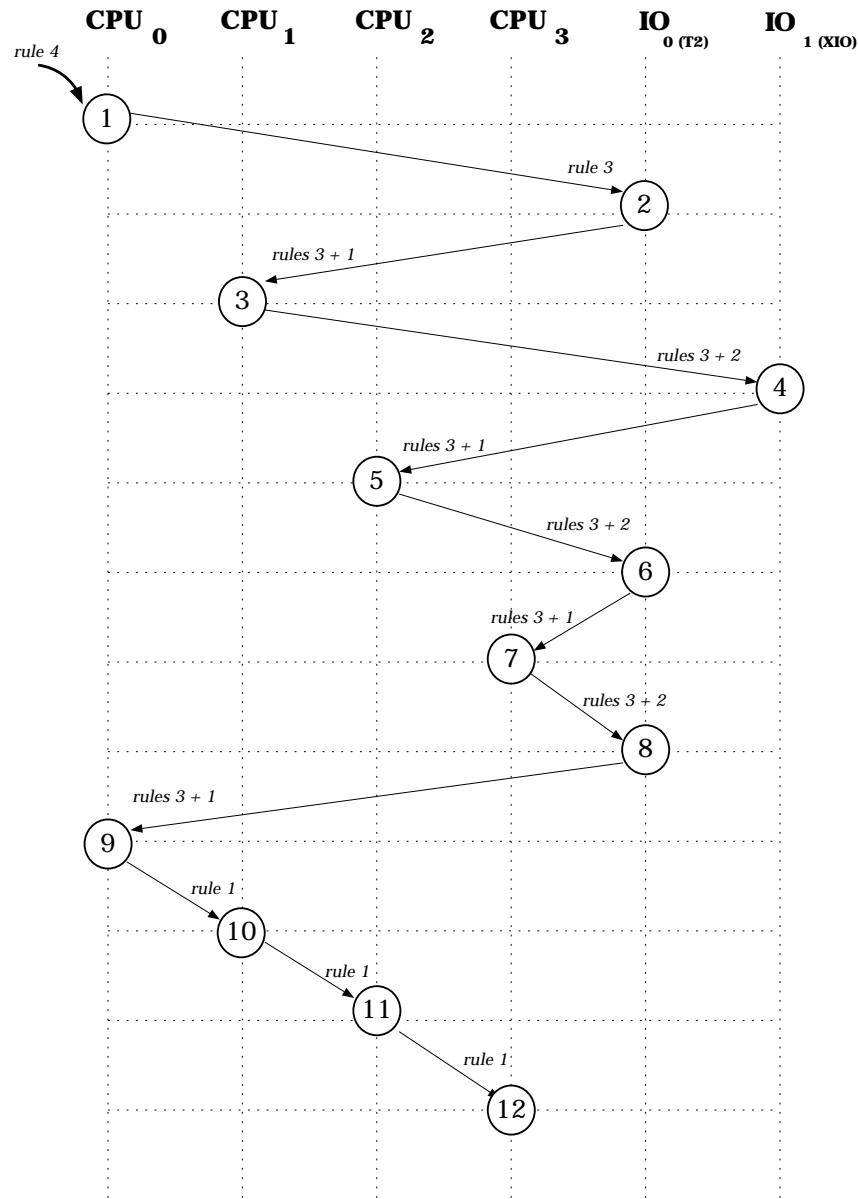
4. The processor type commander requests have biased after the idle Cobra-bus.

For example,

If CPU₀, CPU₁, CPU₂, CPU₃, IO_{0(T2)}, and IO_{1(XIO)} are all requesting the Cobra-bus simultaneously after the system reset event and each node needs the Cobra-bus for two transaction cycles, then the granting of the bus will follow the order listed below. Note that CPU₀ commander node has won the first Cobra-bus mastership because the processor type commander requests have a bias in arbitration. An illustration of the granting order is shown in Figure 58.

1. CPU₀ : algorithm rule-4
2. IO_{0(T2)} : algorithm rule-3
3. CPU₁ : algorithm rule-3, rule-1
4. IO_{1(XIO)} : algorithm rule-3, rule-2
5. CPU₂ : algorithm rule-3, rule-1
6. IO_{0(T2)} : algorithm rule-3, rule-2
7. CPU₃ : algorithm rule-3, rule-1
8. IO_{1(XIO)} : algorithm rule-3, rule-2
9. CPU₀ : algorithm rule-3, rule-1
10. CPU₁ : algorithm rule-1
11. CPU₂ : algorithm rule-1
12. CPU₃ : algorithm rule-1

Figure 58: Granting Order



lyuryan.cpubgranting_order.doc
PS @ 70%

There are special arbitor modes of operation where the bus priority algorithm is modified. In particular, a CPU or I/O mode can guarantee exclusive ownership of the system bus by keeping its appropriate request line asserted. Normal arbitration protocol requires that the request signal deassert for at least one cycle before reasserting if exclusive bus ownership is not required. This deassertion and reassertion can occur during an existing cycle so that back-to-back arbitration is still granted to the requesting node if there are no other requesters for the system bus.

Both CPU and I/O modes may be allowed this mode of operation, however DECchip 21064 and DECchip 21064-A275 implementations will have this feature disabled for CPU requests. Only I/O will be allowed to enter the Exclusive access mode.

During normal operation for Sable DECchip 21064 and DECchip 21064-A275 systems, the Sable bus arbitor will monitor bus activity and insert a single idle cycle after the fifth consecutive back-to-back transaction. This feature guarantees that a CPU is never locked out of its own backup cache and is guaranteed to make progress.

There is also a debug arbitration feature available by writing to bit <12> in the System Bus Control Register (CSR6-CBCTL) which forces the injection of two idle bus cycles between every bus transaction.

Table 52: Sable Arbitration Latency

Latency From EV_Req<2:0> to Cobra Bus cycle 0	
Node	Latency (Bus Cycles)
CPU ₀	3-4
CPU ₁	4-5
CPU ₂	4-5
CPU ₃	4-5
Latency From I/O req to Cobra Bus cycle 0	
Node	Latency (Bus Cycles)
IO ₀ (T2)	3
IO ₁ (XIO)	3

4.15 System-bus CRESET L Generation

System-bus CRESET L is generated on Cobra CPU₀ module. It is asserted when either async_reset from the Standard I/O module is asserted, the System-bus or the 21064 oscillator on CPU₀ stop, or the 3.3V supply on CPU₀ goes under voltage.

The clock detect circuits on the CPU module will guarantee that the oscillator generating the clocks for the module have been running for at least 20ms. This is to allow enough time to for the oscillators to stabilize.

System-bus CRESET L deassertion is synchronous with the System-bus PHI1 H clock. Async_reset from the power subsystem must remain asserted for at least 40ms.

4.15.1 Sable CPU Non-Volatile EEPROM

The Non-Volatile EEPROM is a 256 x 8 byte Non-Volatile RAM that resides on the System Control BUS (I²C). This RAM is used to store the CPU modules serial number, as well as maintaining error logging information. The last 10 bytes of this EEPROM are used to contain the following configuration information:

Table 53: CPU I²C Bus EEPROM Field Definitions

Address	Bits	Definition																									
^xF5	6-0	Cobra-bus speed bits corresponding to CRR<14:8> and CRR<46:40> (Cache Interface speed bits).																									
	7	Duplicate Tag Parity Disable bit, corresponding to CRR<15> and CRR<47>																									
.																											
^xF6-^xFD		Bus Interface Unit Control Register. All 8 bytes of the IPR can be specified by using these bytes in the EEPROM. The EEPROM bytes correspond to the following BIU_CTL bits. These 8 bytes determine the power up state of the cache control register. The definition for all of the BIU_CTL bits follows.																									
^xF6	7-0	BIU_CTL<63:56>																									
^xF7	7-0	BIU_CTL<55:48>																									
^xF8	7-0	BIU_CTL<47:40>																									
^xF9	7-0	BIU_CTL<39:32>																									
^xFA	7-0	BIU_CTL<31:24>																									
^xFB	7-0	BIU_CTL<23:16>																									
^xFC	7-0	BIU_CTL<15:8>																									
^xFD	7-0	BIU_CTL<7:0>																									
^xFE	7-5	Module type field. Must be programmed with the correct value depending on whether the module is a CPU an I/O module or a memory module.																									
<table> <tr> <th>Bit(s)</th><th>Value</th><th>Definition</th></tr> <tr> <td rowspan="4">7 6 5</td><td>0 0 0</td><td>Memory</td></tr> <tr> <td>0 0 1</td><td>CPU *</td></tr> <tr> <td>0 1 0</td><td>Standard I/O</td></tr> <tr> <td>0 1 1</td><td>Expansion I/O</td></tr> <tr> <td>4-2</td><td>x x x</td><td>Unused</td></tr> <tr> <td rowspan="2">1</td><td>0</td><td>Do not run tests on power-up/reset</td></tr> <tr> <td>1</td><td>Run tests on power-up/reset</td></tr> <tr> <td rowspan="2">0</td><td>0</td><td>Do not Auto-unload on power-up/reset</td></tr> <tr> <td>1</td><td>Auto-unload on power-up/reset</td></tr> </table>			Bit(s)	Value	Definition	7 6 5	0 0 0	Memory	0 0 1	CPU *	0 1 0	Standard I/O	0 1 1	Expansion I/O	4-2	x x x	Unused	1	0	Do not run tests on power-up/reset	1	Run tests on power-up/reset	0	0	Do not Auto-unload on power-up/reset	1	Auto-unload on power-up/reset
Bit(s)	Value	Definition																									
7 6 5	0 0 0	Memory																									
	0 0 1	CPU *																									
	0 1 0	Standard I/O																									
	0 1 1	Expansion I/O																									
4-2	x x x	Unused																									
1	0	Do not run tests on power-up/reset																									
	1	Run tests on power-up/reset																									
0	0	Do not Auto-unload on power-up/reset																									
	1	Auto-unload on power-up/reset																									
^xFF	0-7	Checksum																									

NOTE

The control flags in location ^xFE only have meaning if is a Serial Console is attached to the IIC bus on the system. If no Serial Console is present these flags are treated as set i.e. always run tests and auto-unload on power-up/reset.

Table 54: BIU_CTL Field Description

Field	Description										
63:44	MBZ [write-only]										
43	BC_BURST_ALL [write-only]										
42:40	BC_BURST_SPD [write-only]										
39	MBZ [write-only]										
38	SYS_WRAP [write-only]										
37	BYTE_PARITY [write-only]										
36	BAD_DP [write-only] Bad Data Parity - When set, BAD_DP causes the 21064 CPU to invert the value placed on bits <0>, <7>,<14> and <21> of the check_h<27:0> field during off-chip writes. This produces bad parity when the 21064 CPU is in parity mode, and bad check bit codes when the CPU is in EDC mode.										
35:32	BC_PA_DIS [write-only] Backup cache physical address disable - This 4-bit field may be used to prevent the CPU chip from using the external cache to service reads and writes based on the quadrant of physical address space which they reference. The table below shows the correspondence between this bit field and the physical address space. When a read or write reference is presented to the bus interface unit (BIU), the values of BC_PA_DIS, BC_ENA, and physical address bits <33:32> together determine whether or not to try using the external cache to satisfy the reference. If the external cache is not to be used for a given reference, the bus interface unit does not probe the tag store and makes the appropriate system request immediately. The value of BC_PA_DIS has no impact on which portions of the physical address space may be cached in the primary caches. System components control this through the DRACK field of the pin bus. BC_PA_DIS is not initialized by a reset.										
<table> <tr> <th>BIU_CTL bit</th><th>Physical Address</th></tr> <tr> <td>[32]</td><td>PA[33..32] = 0</td></tr> <tr> <td>[33]</td><td>PA[33..32] = 1</td></tr> <tr> <td>[34]</td><td>PA[33..32] = 2</td></tr> <tr> <td>[35]</td><td>PA[33..32] = 3</td></tr> </table>		BIU_CTL bit	Physical Address	[32]	PA[33..32] = 0	[33]	PA[33..32] = 1	[34]	PA[33..32] = 2	[35]	PA[33..32] = 3
BIU_CTL bit	Physical Address										
[32]	PA[33..32] = 0										
[33]	PA[33..32] = 1										
[34]	PA[33..32] = 2										
[35]	PA[33..32] = 3										
31	BAD_TCP [write-only] BAD Tag Control Parity - When set, BAD_TCP causes the 21064 CPU to write bad parity into the tag control RAM whenever it does a fast external RAM write.										

Table 54 (Cont.): BIU_CTL Field Description

Field	Description																
30:28	<p>BC_SIZE [write-only]</p> <p>Backup Cache Size - This field is used to indicate the size of the external cache. BC_SIZE is not initialized by a reset and must be explicitly written before enabling the backup cache.</p> <table> <tr> <th>BC_SIZE</th><th>Size</th></tr> <tr> <td>0 0 0</td><td>128 Kbytes</td></tr> <tr> <td>0 0 1</td><td>256 Kbytes</td></tr> <tr> <td>0 1 0</td><td>RESERVED</td></tr> <tr> <td>0 1 1</td><td>1 Mbytes</td></tr> <tr> <td>1 0 0</td><td>2 Mbytes</td></tr> <tr> <td>1 0 1</td><td>4 Mbytes</td></tr> <tr> <td>1 1 0</td><td>8 Mbytes</td></tr> </table>	BC_SIZE	Size	0 0 0	128 Kbytes	0 0 1	256 Kbytes	0 1 0	RESERVED	0 1 1	1 Mbytes	1 0 0	2 Mbytes	1 0 1	4 Mbytes	1 1 0	8 Mbytes
BC_SIZE	Size																
0 0 0	128 Kbytes																
0 0 1	256 Kbytes																
0 1 0	RESERVED																
0 1 1	1 Mbytes																
1 0 0	2 Mbytes																
1 0 1	4 Mbytes																
1 1 0	8 Mbytes																
27:13	<p>BC_WE_CTL[15:1] [write-only]</p> <p>Backup Cache Write Enable Control. This field controls the timing of the write enable and chip enable pins during writes into the data and tag control RAMs. It consists of 15 bits, where each bit determines the value placed on the write enable and chip enable pins during a given CPU cycle of the RAM write access.</p> <p>When a given bit of BC_WE_CTL is set, the write enable and chip enable pins are asserted during the corresponding CPU cycle of the RAM access. BC_WE_CTL<0> (bit 13 in BIU_CTL) corresponds to the second cycle of the write access, BC_WE_CTL<1> (bit 14 in BIU_CTL) to the third CPU cycle, and so on. The write enable pins are never asserted in the first CPU cycle of a RAM write access.</p> <p>Unused bits in the BC_WE_CTL field must be written with 0s.</p> <p>BC_WE_CTL is not initialized on reset and must be explicitly written before enabling the external backup cache.</p>																
12	<p>DELAY_WDATA [write-only]</p> <p>DELAY_DATA</p>																
11:8	<p>BC_WR_SPD [write-only]</p> <p>Backup cache write speed. This field indicates to the bus interface unit the write cycle time of the RAMs used to implement the off-chip external cache, (Backup cache on the CPU module), measured in CPU cycles. It should be written with a value equal to one less the write cycle time of the external cache RAMs.</p> <p>Access times for writes must be in the range 16..2 CPU cycles, which means the values for the BC_WR_SPD field are in the range of 15..1.</p> <p>BC_WR_SPD is not initialized on reset and must be explicitly written before enabling the external cache.</p>																

Table 54 (Cont.): BIU_CTL Field Description

Field	Description
7:4	<p>BC_RD_SPD [write-only]</p> <p>Backup (external) cache read speed. This field indicates to the bus interface unit the read access time of the RAMs used to implement the off-chip external cache, measured in CPU cycles. This field should be written with a value equal to one less the read access time of the external cache RAMs.</p> <p>Access times for reads must be in the range 16..3 CPU cycles, which means the values for the BC_RD_SPD field are in the range of 15..2.</p> <p>BC_RD_SPD are not initialized on reset and must be explicitly written before enabling the external cache.</p>
3	<p>BC_FHIT [write-only]</p> <p>Backup cache force hit. (This cache is external to the 21064 chip.) When this bit and BC_EN are set, all pin bus READ_BLOCK and WRITE_BLOCK transactions are forced to hit in the backup cache. Tag and tag control parity are ignored when the BIU operates in this mode. BC_EN takes precedence over BC_FHIT. When BC_EN is clear and BC_FHIT is set, no tag probes occur and external requests are directed to the CREQ_H pins.</p> <p>Note that the BC_PA_DIS field takes precedence over the BC_FHIT bit.</p>
2	<p>OE [write-only]</p> <p>Output enable - When this bit is set, the 21064 CHP chip does not assert its chip enable pins during RAM write cycles, thus enabling these pins to be connected to the output enable pins of the cache RAMs.</p>
1	<p>EDC [write-only]</p> <p>Error detection and correction. When this bit is set, the 21064 CPU chip generates/expects EDC on the CHECK_H pins. When this bit is clear the CPU chip generates/expects parity on four of the CHECK_H pins.</p>
0	<p>BC_ENA [write-only]</p> <p>External cache enable. When clear, this bit disables the external cache. When the external cache is disabled, the BIU does not probe the external cache tag store for read and write references; it initiates a request on CREQ_H immediately.</p>

Table 56: BIU_CTL Initialization Values

Module Description	Init Vaule	Note
Sable B2020-AA	E30006447	1MB 5 tick cache, 2 cycle write strobe
Sable B2024-AA	E5000E567	4MB 6/7 tick cache, 3 cycle write strobe

If an error occurs (either non-existent or bad checksum) when reading the CPU's IIC bus EEPROM the following defaults for will be used.

Table 57: CPU EEPROM Defaults

Location	Value	Definition
^xF5	^x05	speed bits for 5.26ns Ev and 24ns C-bus
^xF6	^x00	mbz
^xF7	^x00	mbz
^xF8	^x00	burst mode clear, burst speed clear
^xF9	^x0E	sys_wrap, byte_parity, bad_dp clear. PA_DIS = 1110
^xFA	^x30	bad_tcp clear, cache size = 1Meg
^xFB	^x00	upper portion of we_ctl
^xFC	^x64	we_ctl, delay_write_data clear, wr_spd = 5 ticks
^xFD	^x47	rd_spd = 5 ticks, oe, ecc, bc_ena all set
^xFE	^x23	auto-unload, run tests on power-up both set, module type is CPU

The RAM address on the system control bus is shown below:

Table 58:

Transaction	ADDRESS			
	CPU ₀	CPU ₁	CPU ₂	CPU ₃
WRITE	A8	AA	AE	A2
READ	A9	AB	AF	A3

CHAPTER 5

CPU MODULE TRANSACTIONS

The CPU module has two transaction initiators, the processor, and the System-bus interface controller.

The processor initiates a transaction as a result of a program operation that requires access to memory or I/O devices. When a local resource can not service the processor request (as a result of a cache miss or I/O space access) a request is made to the System-bus interface controller to obtain the information from the System-bus.

System-bus transactions occur as a result of System-bus resource requests from any of the possible System-bus Commanders. Transactions are monitored by every member of the bus for the following reasons:

- Any module on the System-bus could be a responder to a specific System-bus transaction.
- The System-bus implements a “snooping” protocol to guarantee cache coherence.

Refer to the Cobra-bus2 spec for details.

5.1 Processor Transactions

The 21064 Processor requests an external cycle when it determines that the cycle it wants to perform requires module level action.

The cycle types are as follows:

BARRIER

A BARRIER cycle is generated by the MB instruction. As no external write buffer exists between the processor and an error detection point in the system the Cobra CPU module will just acknowledge the cycle.

FETCH and FETCHM

The FETCH and FETCHM cycles are generated by the FETCH and FETCHM instructions respectively. The B-Cache controller will acknowledge the instruction and no other action will take place.

READ_BLOCK

The READ_BLOCK cycle is generated on read misses. The B-Cache controller reads the addressed block from memory and supplies it, 128 bits at a time, to the 21064 via the data bus. The B-Cache location that missed will be victimized and then updated with the new cache entry.

WRITE_BLOCK

The WRITE_BLOCK cycle is generated on write misses, and on writes to shared blocks. The B-Cache controller pulls the write data, 128 bits at a time, from the 21064 via the data bus, and writes the valid longwords to memory.

LDxL

The LDxL cycle is generated by the LDLL and LDQL instructions. The cycle works just like a READ_BLOCK, although the B-Cache is not probed by the processor. The B-Cache controller performs the B-Cache probe and if the reference is to cacheable address space, the address is latched into the Address Lock Register.

STxC

The STxC cycle is generated by the STLC and STQC instructions. The cycle works just like a WRITE_BLOCK, although the B-Cache is not probed by the processor. The B-Cache controller performs the B-Cache probe and then the cycle is acknowledged with the completion status. The following tables show the encodings of the cycle request and acknowledge signals between DECchip 21064 and the interface ASICs.

Table 59: Cycle Request

cReq_h[2]	cReq_h[1]	Creq_h[0]	Type
L	L	L	IDLE
L	L	H	BARRIER
L	H	L	FETCH
L	H	H	FETCH_M
H	L	L	READ_BLOCK
H	L	H	WRITE_BLOCK
H	H	L	LDL_L/LDQ_L
H	H	H	STL_C/STQ_C

Table 60: Cycle Acknowledgment Types

cAck_h[2]	cAck_h[1]	cAck_h[0]	Type
L	L	L	IDLE
L	L	H	HARD_ERROR
L	H	L	SOFT_ERROR†
L	H	H	STL_C_FAIL/STQ_C_FAIL
H	L	L	OK

†This type not used in Sable systems

Table 61: Read Data Acknowledgment Types

dRAck_h[2]	dRAck_h[1]	dRAck_h[0]	Type
L	L	L	IDLE
H	L	L	OK_NCACHE_NCHK†
H	L	H	OK_NCACHE
H	H	L	OK_NCHK†
H	H	H	OK

†This type not used in Sable systems

Table 62: Processor Initiated Transactions

TRANSACTION	Activity
P-Cache Read	Not Visible outside Processor
P-Cache Write	B-Cache Written if Hit Write Block Generated if Miss
P-Cache Masked Write	B-Cache Written if Hit Write Block Generated if Miss
Fast B-Cache Read Hit	B-Cache data read
Fast B-Cache Write Hit	Data written to B-Cache data store, Dirty bit set
Fast B-Cache Masked Write Hit	Data written to B-Cache data store, Dirty bit set
Read Block ¹	System-bus read or exchange cycle generated
Write Block ²	System-bus read or exchange and possibly write or nut cycles generated
LDxL - Load Lock	System-bus read, exchange, or nut cycle generated, IF (cacheable address space reference) THEN address latched and Lock bit set IF (Non-cacheable address space) THEN no change to address lock or lock bit.
STxC - Store Conditional	System-bus read, exchange and possibly nut or write cycles generated, IF (cacheable address space reference) Lock bit cleared (it was set) store completes IF (Non-cacheable address space) THEN no change to lock bit, store failed if responder asserts UC_ERR L during cycle.
Barrier ³	All data buffers flushed to system coherence point. Acknowledge request, no other module level activity.
FETCH/FETCHM ⁴	Acknowledge Request, no other module level activity.

¹Generated as a result of a Fast B-Cache Read Miss.²Generated as a result of a Fast B-Cache Write Miss.³Generated as a result of the execution of a Memory Barrier Instruction.⁴Generated as a result of the execution of a FETCH or FETCHM Instruction.

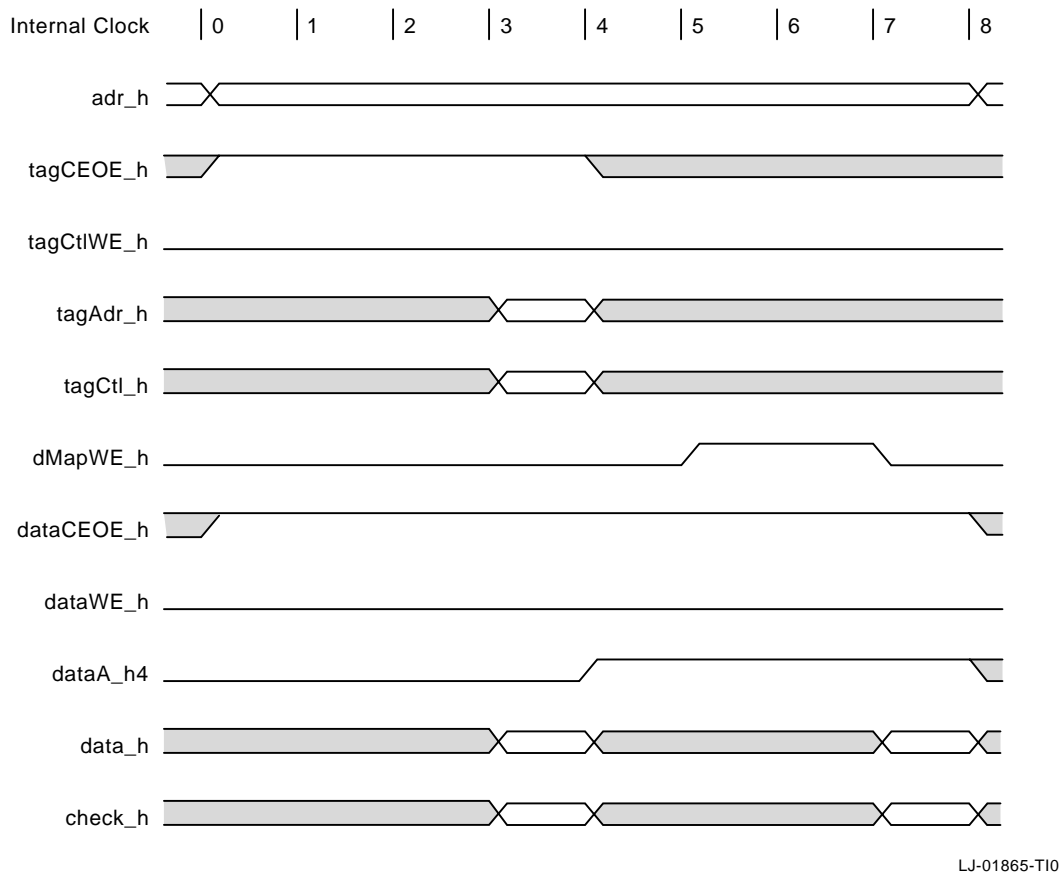
5.1.1 21064 Processor TRANSACTIONS

5.1.1.1 FAST EXTERNAL CACHE READ HIT

A fast external cache read consists of a probe read (overlapped with the first data read), followed by the second data read if the probe hits. The following diagram illustrates the Cobra CPU fast external cache read which selects 4 CPU cycle reads (BC_RD_SPD = 3), 4 CPU cycle writes (BC_WR_SPD = 3), chip enable control OE = L).

If the probe misses then the cycle aborts at the end of clock cycle 3. If the probe hits and the miss address had bit 4 set then the two data reads would have been swapped (dataA_h[4] would have been true in cycles 0, 1, 2, 3, and would have been false in cycles 4, 5, 6, 7).

Figure 59: FAST EXTERNAL CACHE READ HIT



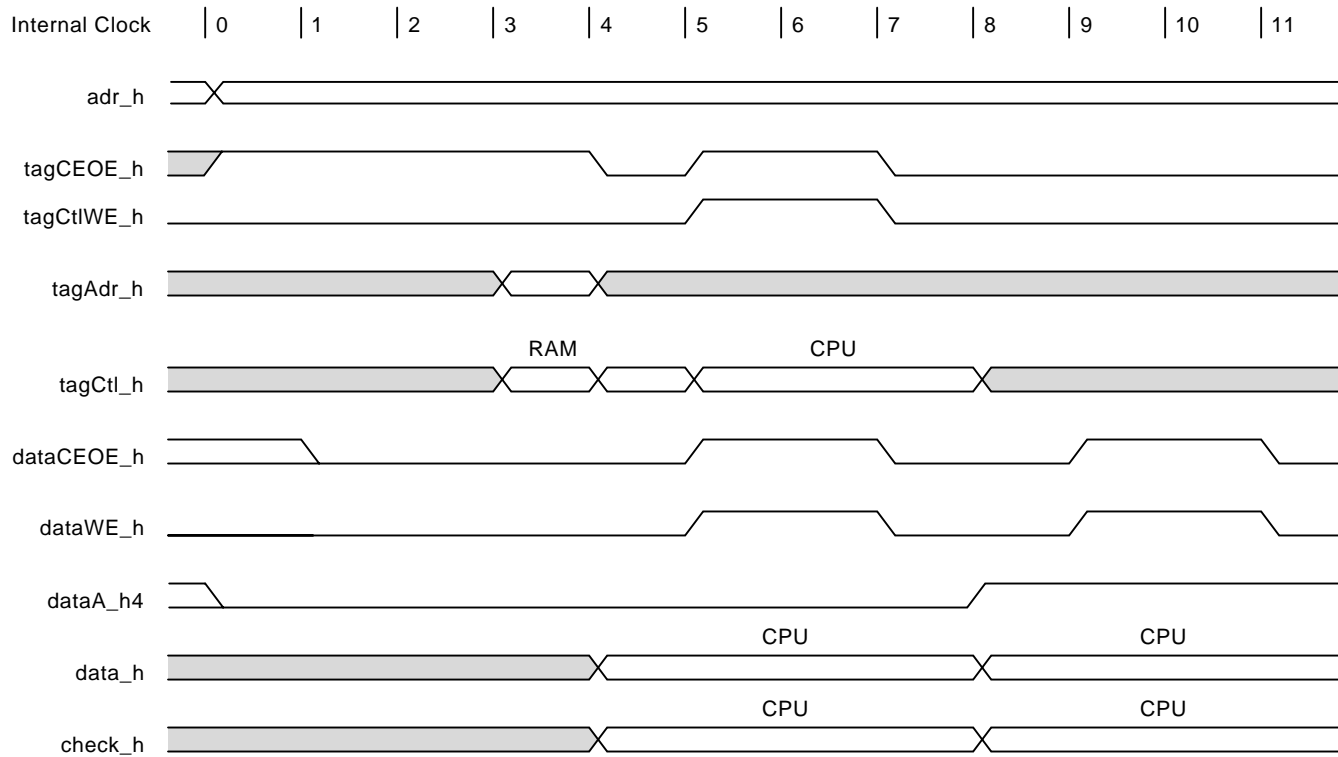
5.1.1.2 FAST EXTERNAL CACHE WRITE HIT

A fast external cache write consists of a probe read, followed by 1 or 2 data writes. The following diagram shows that the Cobra CPU external cache transaction is using 4 CPU cycle reads (BC_RD_SPD = 3), 4 CPU cycle writes (BC_WR_SPD = 3), chip enable control (OE = L), and a 2 cycle write pulse centered in the 4 cycle write (BC_WE_CTL[15..1] = LLLLLLLLLLLLLLHH).

Note that EVx drives the tagCtl_h pins one CPU cycle later than it drives the data_h and check_h pins relative to the start of the write cycle. This is because, unlike data_h and check_h, the tagCtl_h field must be read during the tag probe which precedes the write cycle. Because EVx can switch its pins to a low impedance state much more

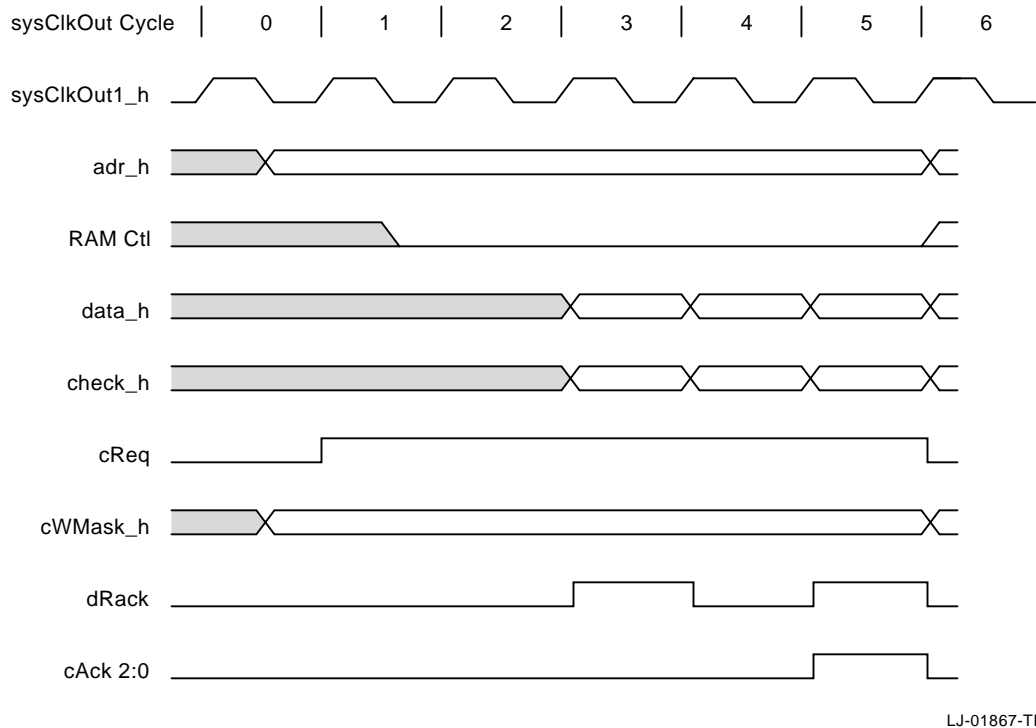
quickly than most RAMs can switch their pins to a high impedance state, EVx waits one CPU cycle before driving the tagCtl_h pins in order to minimize tristate driver overlap. If the probe misses then the cycle aborts at the end of clock cycle 3.

Figure 60: FAST EXTERNAL CACHE WRITE HIT



5.1.1.3 READ_BLOCK TRANSACTION

A READ_BLOCK transaction appears at the external interface on external cache read misses, either because it really was a miss, or because the external cache has not been enabled.

Figure 61: READ BLOCK TRANSACTION

1. The `cReq_h` pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The `adr_h` pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
2. The READ_BLOCK transaction begins. The 21064 has already placed the address of the block containing the miss on `adr_h`. The 21064 places the quadword-within-block and the I/D indication on `cWMask_h`. The 21064 places a READ_BLOCK command code on `cReq_h`. The 21064 will clear the RAM control pins (`dataA_h[4..3]`, `dataCEOE_h[3..0]` and `tagCEOE_h`) no later than one CPU cycle after the system clock edge at which the transaction begins.
3. The external logic obtains the first 16 bytes of data. Although a single stall cycle has been shown here, there could be no stall cycles, or many stall cycles.
4. The external logic has the first 16 bytes of data. It places it on the `data_h` and `check_h` busses. It asserts `dRack_h` to tell the 21064 that the data and check bit busses are valid. The 21064 detects `dRack_h` at the end of this cycle, and reads in the first 16 bytes of data at the same time.

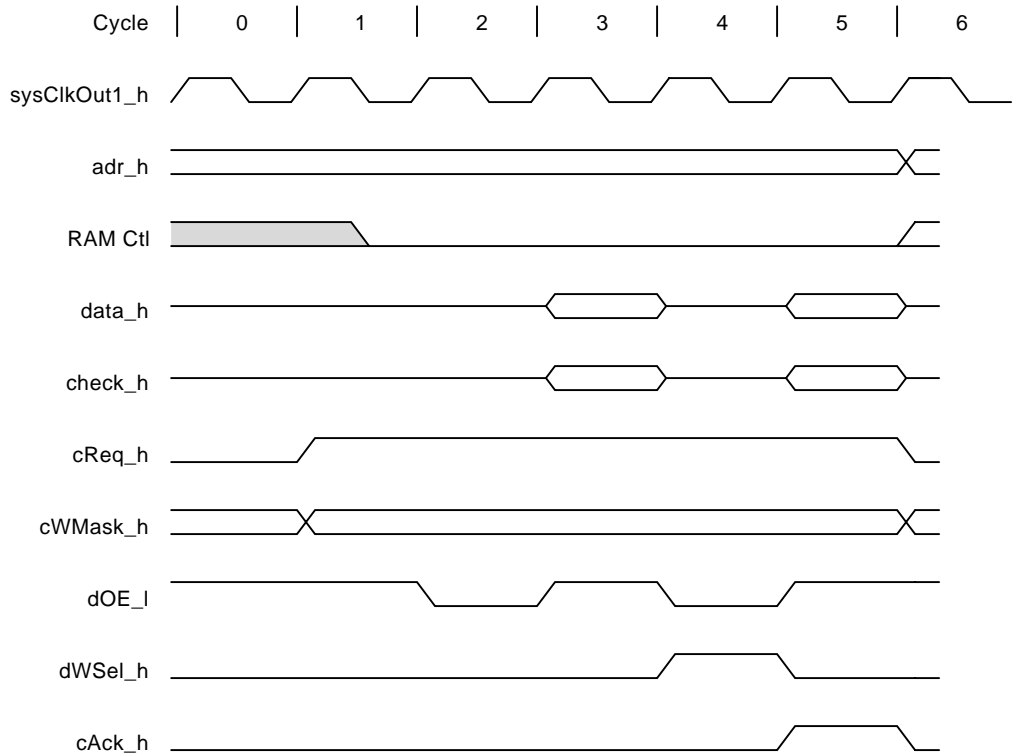
5. The external logic obtains the second 16 bytes of data. Although a single stall cycle has been shown here, there could be no stall cycles, or many stall cycles.
6. The external logic has the second 16 bytes of data. It places it on the data_h and check_h busses. It asserts dRAck_h to tell the 21064 that the data and check bit busses are valid. The 21064 detects dRAck_h at the end of this cycle, and reads in the second 16 bytes of data at the same time. In addition, the external logic places an acknowledge code on cAck_h to tell the 21064 that the READ_BLOCK cycle is completed. The 21064 detects the acknowledge at the end of this cycle, and may change the address.
7. Everything is idle. The 21064 could start a new external cache cycle at this time.

Because external logic owns the RAMs by virtue of the 21064 having deasserted its RAM control signals at the beginning of the transaction, external logic may cache the data by asserting its write pulses on the external cache during cycles 3 and 5.

The 21064 performs EDC checking on the data supplied to it via the data and check busses if so requested by the acknowledge code. It is not necessary to place data into the external cache to get checking.

5.1.1.4 WRITE_BLOCK

A WRITE_BLOCK transaction appears at the external interface on external cache write misses (either because it really was a miss, or because the external cache has not been enabled), or on external cache write hits to shared blocks.

Figure 62: WRITE BLOCK

LJ-01868-T10

1. The `cReq_h` pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The `adr_h` pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
2. The `WRITE_BLOCK` cycle begins. The 21064 has already placed the address of the block on `adr_h`. The 21064 places the longword valid masks on `cWMask_h` and a `WRITE_BLOCK` command code on `cReq_h`. The 21064 will clear `dataA_h[4..3]` and `tagCEOE_h` no later than one CPU cycle after the system clock edge at which the transaction begins. The 21064 clears `dataCEOE_h[3..0]` at least one CPU cycle before the system clock edge at which the transaction begins.
3. The external logic detects the command, and asserts `dOE_l` to tell the 21064 to drive the first 16 bytes of the block onto the data bus.
4. The 21064 drives the first 16 bytes of write data onto the `data_h` and `check_h` busses, and the external logic writes it into the destination. Although a single stall cycle has been shown here, there could be no stall cycles, or many stall cycles.

5. The external logic asserts `dOE_l` and `dWSel_h` to tell the 21064 to drive the second 16 bytes of data onto the data bus.
6. The 21064 drives the second 16 bytes of write data onto the `data_h` and `check_h` busses, and the external logic writes it into the destination. Although a single stall cycle has been shown here, there could be no stall cycles, or many stall cycles. In addition, the external logic places an acknowledge code on `cAck_h` to tell the 21064 that the `WRITE_BLOCK` cycle is completed. The 21064 detects the acknowledge at the end of this cycle, and change the address and command to the next values.
7. Everything is idle. Because external logic owns the RAMs by virtue of the 21064 having deasserted its RAM control signals at the beginning of the transaction, external logic may cache the data by asserting its write pulses on the external cache during cycles 3 and 5.

The 21064 performs EDC generation on data it drives onto the data bus. Cobra's 21064 processor interface performs EDC checking and correction on this data.

Although in the above diagram external logic cycles through both 128-bit chunks of potential write data, this need not always be the case. External logic must pull from the 21064 chip only those 128-bit chunks of data which contain valid longwords as specified by the `cWMask_h` signals. The only requirement is that if both halves are pulled from the 21064, then the lower half must be pulled before the upper half.

5.1.1.5 LDxL TRANSACTION

An LDxL transaction appears at the external interface as a result of an LDQL or LDLL instruction being executed. The external cache is not probed. With the exception of the command code output on the `cReq` pins, the LDxL transaction is exactly the same as a `READ_BLOCK` transaction. See section Refer to Section 5.1.1.3.

5.1.1.6 STxC TRANSACTION

An STxC transaction appears at the external interface as a result of STLC and STQC instructions. The external cache is not probed.

The STxC transaction is the same as the `WRITE_BLOCK` transaction, with the following exceptions:

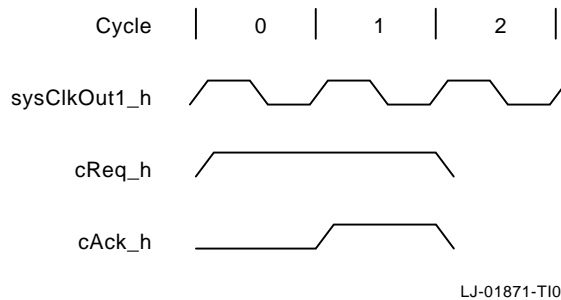
1. The code placed on the `cReq` pins is different.
2. The `cWMask` field will never validate more than a single longword or quadword of data.
3. External logic has the option of making the transaction fail by using the `cAck` code of `STxC_FAIL`. It may do so without asserting either `dOE_l` or `dWSel_h`.

Refer to Section 5.1.1.4.

5.1.1.7 BARRIER TRANSACTION

The BARRIER transaction appears on the external interface as a result of an MB instruction. The acknowledgment of the BARRIER transaction tells the 21064 that all invalidates have been supplied to it, and that any external write buffers have been pushed out to the coherence point. Any errors detected during these operations can be reported to the 21064 when the BARRIER transaction is acknowledged. The Cobra CPU will immediately acknowledge this transaction because it does not buffer block_write transactions.

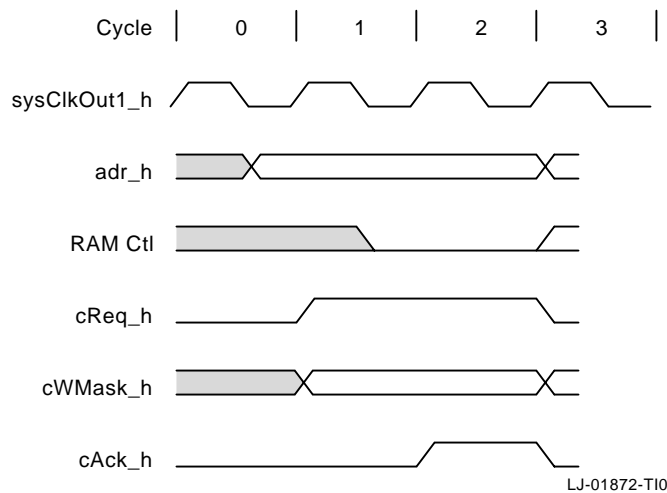
Figure 63: BARRIER TRANSACTION



1. The BARRIER transaction begins. The 21064 places the command code for BARRIER onto the `cReq_h` outputs.
2. The external logic notices the BARRIER command, and because it has completed processing the command (it isn't going to do anything), it places an acknowledge code on the `cAck_h` inputs.
3. The 21064 detects the acknowledge on `cAck_h`, and removes the command. The external logic removes the acknowledge code from `cAck_h`. The cycle is finished.

5.1.1.8 FETCH TRANSACTION

A FETCH transaction appears on the external interface as a result of a FETCH instruction. The transaction supplies an address to the external logic, which the system chooses to ignore and responds with an immediate acknowledge.

Figure 64: FETCH TRANSACTION

1. The cReq_h pins are always idle in the system clock cycle immediately before the beginning of an external transaction. The adr_h pins always change to their final value (with respect to a particular external transaction) at least one CPU cycle before the start of the transaction.
2. The FETCH transaction begins. The 21064 has already placed the effective address of the FETCH on the address outputs. The 21064 places the command code for FETCH on the cReq_h outputs. The 21064 will clear the RAM control pins (dataA_h[4..3], dataCEOE_h[3..0] and tagCEOE_h) no later than one CPU cycle after the system clock edge at which the transaction begins.
3. The external logic notices the FETCH command, and because it has completed processing the command (it isn't going to do anything), it places an acknowledge code on the cAck_h inputs.
4. The 21064 detects the acknowledge on cAck_h, and removes the address and the command. The external logic removes the acknowledge code from cAck_h. The cycle is finished.

5.1.1.9 FETCHM TRANSACTION

A FETCHM transaction appears on the external interface as a result of a FETCHM instruction. The transaction supplies an address to the external logic, which Cobra chooses to ignore and responds with an immediate acknowledge. With the exception of the command code placed on cReq_h, the FETCHM transaction is the same as the FETCH transaction. Refer to Section 5.1.1.8.

Refer to the 21064 Processor Specification for further details.

5.1.2 Cacheable vs Non-Cacheable vs Allocate-Invalid

Cacheable

Only “memory like” locations will be cached. These are defined as those locations with address bits 32 and 33 equal to “0”. These locations will be placed in the B-Cache when it is enabled as a side effect of the processor issuing a READ_BLOCK. They will also be placed in the P-Cache as the read data will be acknowledged with OK.

Non-Cacheable

“Non-memory like” locations will **not** be cached. These are defined as those locations with address bits 32 or 33 equal to “1”. These locations will not be placed in the B-Cache, and the read data will be acknowledged with OK_NCACHE. Writes to Non-Cacheable space are restricted to aligned quadword accesses only. The quadword write data is presented to the system-bus in the proper quadword associated with the address of the access. The data presented in the other quadwords of the cache block (during that system-bus cycle) is undefined; however the correct system-bus parity will be driven. No read merge will occur. All I/O locations are Non-Cacheable locations.

Allocate-Invalid

When the ENB B-CACHE INIT bit in the BCC register is clear, “Memory like” locations with address bit 33 equal to “0” and address bit 32 equal to “1”, and address bit 31 equal to “0”, will be treated as Cacheable locations to be allocated in the B-Cache as invalid. Read references to locations in this address space are intercepted by the C4 and cause either an EXCHANGE, or a READ request on the System-bus (depending on the state of the DIRTY bit of the cache block in question). Address bit 32 will be masked off before the address is presented on the System-bus to provide a legal Cobra System-bus address.

When the read data is returned, the cache block in the B-Cache will be marked invalid.

When the ENB B-CACHE INIT bit in the BCC register is set, “Memory like” locations with address bit 33 equal to “0” and address bit 32 equal to “1”, and address bit 31 equal to “0”, will be treated as Cacheable locations to be allocated in the B-Cache. Read references to locations in this address space are intercepted by C4 and cause the data found in the local CPU modules CSR8 (CBEAL) to be returned as the read data and subsequently filled into the B-Cache. Address bit 32 is passed through the C4 and presented to the System-bus. The C4 acknowledges the transaction itself, and the other CPU module will not probe its B-Cache. Refer to the definition of the B-Cache Control and Status Register, and Section 9.2 for further details.

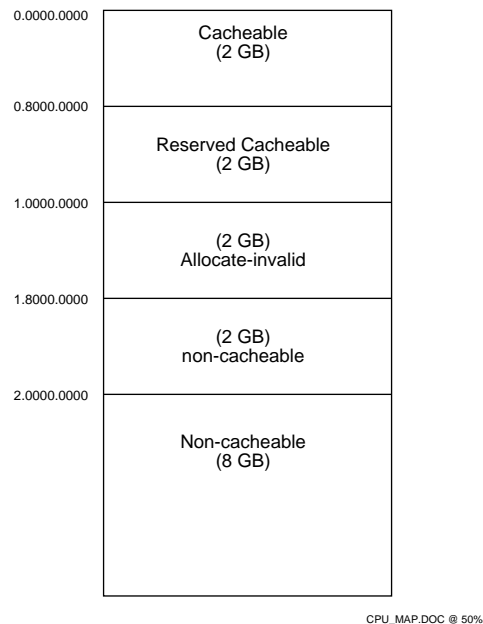
This feature can be used to just flush dirty entries from the cache, and/or to also guarantee that all cache entries are marked invalid. Simple flushing of dirty entries can be accomplished by setting the FORCE_SHARED bit in the CBCTL register, and then walking the Allocate-Invalid address space to flush the cache (a process at low

IPL). When the full cache address space (nominally 1-4MB) has been accessed, the cache is guaranteed to be clean; however not necessarily invalid. If the routine that flushes the cache operates at an IPL higher than any other processes can execute, then when complete, the cache will be clean and marked completely invalid.

NOTE

As the bus interface re-maps all references through Allocate-Invalid space by clearing address bit 32 and presenting the resulting address to the memory subsystem. Only privileged processes should be allowed to map Allocate-Invalid space into their virtual address space.

Figure 65: Address Space Map



5.2 System-bus Transactions

There are four System-bus transaction types. These are READ, WRITE, EXCHANGE, and NUT. READ, and WRITE transactions consist of a Command/Address cycle followed by two data cycles. The EXCHANGE transaction, which is used for dirty victim processing, consists of a Command/Address cycle followed by four data cycles, two write and two read. The NUT transaction consists of a Command/Address cycle only.

Refer to the Cbus Version 2 specification for a more detailed description of the System-bus.

As the System-bus is based on a “snooping” protocol, every System-bus transaction is monitored by all System-bus participants. The memory modules can only act as responders or bystanders, and as such only respond to requests from bus commanders. The Processor and I/O subsystems can act as either commanders, responders, or bystanders.

Table 63: System-bus Initiated Transactions

Transaction	Abbrv	System-bus cycles	Activity
Write	WR	6	B-Cache probe;
		9	IF HIT update/invalidate IF update pull System-bus SHARED, cache block SHARED = prev SHARED
Read	RD	7	B-Cache probe; Provide data if HIT dirty
Exchange	XD	7	B-Cache probe (READ); Provide data if HIT dirty
No-op Transaction	NUT	7	No operation

Refer to Table 65 for a detailed description of the control flows for System-bus initiated cycles.

5.2.1 CPU as Commander

When one of the CPU module’s processors requests data that isn’t resident on the CPU module, a request is passed to the System-bus interface controller, and a System-bus transaction is initiated.

5.2.2 CPU as Bystander

The CPU module is a Bystander when some other System-bus commander has requested data from a resource on the System-bus that doesn’t reside exclusively on the CPU module.

As a Bystander, each B-Cache controller may have to update or invalidate a stale datum, or provide dirty data to the System-bus thereby pre-empting a memory module from returning stale data (for that transaction only).

5.2.3 CPU as Responder

The CPU module is a Responder when some other System-bus commander has requested data from a System-bus visible CPU module register.

5.3 Control Flow of CPU Module Transactions

5.3.1 Processor Initiated

The CPU module B-Cache controller provides B-Cache control under the following circumstances. A B-Cache miss occurs, a LDxC, STxC, FETCH/FETCHM, or a MB is executed, or when a write to a shared cache block is detected.

The behavior of the B-Cache controller is based on the current processor cycle type, the state of the B-Cache control bits, and the state of the System-bus. The following table illustrates the control flows for these cycles.

Table 64: Processor Initiated Transactions - Control Flow

Cycle	Datum/Cache		Activity
	Status	Size	
Read HIT†	X‡	X	<ul style="list-style-type: none"> Block read - Processor managed³² Update duplicate tag store
Write HIT	NOT Shared†	X	<ul style="list-style-type: none"> Block written (DIRTY = TRUE) - Processor managed³
	Shared	32 bytes	<ul style="list-style-type: none"> Request System-bus³⁴⁵ System-bus WRITE cycle¹⁵ Update Cache block (DIRTY = FALSE, SHARED = System-bus shared during write) Relinquish System-bus ACK write

¹Refer to Section 4.2 for details.

²Processor Checks Data EDC.

³Processor Checks Tag Store and Control Store Parity.

⁴B-Cache Controller Checks Tag Store and Control Store Parity.

⁵B-Cache Controller Checks Data EDC.

†“Fast Cache” cycles, external logic does not intervene

‡X - don't care

Table 64 (Cont.): Processor Initiated Transactions - Control Flow

Cycle	Datum/Cache		Activity
	Status	Size	
	Shared	< 32 bytes	<ul style="list-style-type: none"> Request System-bus³⁴⁵ System-bus WRITE cycle¹⁵ with merge Update Cache block (DIRTY = FALSE, SHARED = System-bus shared during write) Relinquish System-bus ACK write
Read MISS	No Victim / Clean Victim	X	<ul style="list-style-type: none"> Request System-bus³⁴ System-bus READ cycle⁶ Cache block allocated² (SHARED = System-bus share during read, DIRTY = FALSE) Relinquish System-bus ACK read
	Dirty Victim	X	<ul style="list-style-type: none"> Request System-bus³⁴ System-bus EXCHANGE cycle⁵⁶ Cache block allocated² (SHARED = System-bus share during read, DIRTY = FALSE) Relinquish System-bus ACK read
	Enable Allocate Disabled (BCC bit 0 clear)	X	<ul style="list-style-type: none"> Request System-bus If B-Cache probe dirty then System-bus EXCHANGE cycle⁶ If B-Cache probe not dirty then System-bus READ cycle⁶ Update Cache block (VALID = NOT VALID) Relinquish System-bus ACK Read no cache

¹Refer to Section 4.2 for details.²Processor Checks Data EDC.³Processor Checks Tag Store and Control Store Parity.⁴B-Cache Controller Checks Tag Store and Control Store Parity.⁵B-Cache Controller Checks Data EDC.⁶System-bus interface Controller Checks System-bus Parity.

Table 64 (Cont.): Processor Initiated Transactions - Control Flow

Cycle	Datum/Cache		Activity
	Status	Size	
Write MISS	No Victim / Clean Victim	< 32 bytes	"Multiple-Arb Operation" <ul style="list-style-type: none"> Request System-bus³⁴ System-bus READ cycle⁶ Cache block allocated (SHARED = System-bus share during read, DIRTY = FALSE) Continue to request System-bus Write data merged with cache block (SHARED = SHARED, DIRTY = FALSE) System-bus WRITE cycle¹⁵ Update B-Cache Relinquish System-bus ACK write
	Dirty Victim	< 32 bytes	"Multiple-Arb Operation" <ul style="list-style-type: none"> Request System-bus³⁴ System-bus EXCHANGE cycle⁵⁶ Cache block allocated (SHARED = System-bus share during read, DIRTY = FALSE) Continue to request System-bus Write data merged with cache block (SHARED = SHARED, DIRTY = FALSE) System-bus WRITE cycle¹⁵ Update B-Cache Relinquish System-bus ACK write
	Enable Allocate Disabled (BCC bit 0 clear)	< 32 bytes	"Multiple-Arb Operation" <ul style="list-style-type: none"> Request System-bus If B-Cache probe dirty then System-bus EXCHANGE cycle⁶ If B-Cache probe not dirty then System-bus READ cycle⁶ Update Cache block (VALID = NOT VALID) Continue to request System-bus System-bus WRITE cycle from merge buffer Relinquish System-bus ACK write

¹Refer to Section 4.2 for details.³Processor Checks Tag Store and Control Store Parity.⁴B-Cache Controller Checks Tag Store and Control Store Parity.⁵B-Cache Controller Checks Data EDC.⁶System-bus interface Controller Checks System-bus Parity.

Table 64 (Cont.): Processor Initiated Transactions - Control Flow

Cycle	Datum/Cache		Activity
	Status	Size	
	X	32 bytes	<ul style="list-style-type: none"> Request System-bus³⁴ System-bus WRITE cycle⁵ Relinquish System-bus ACK write
LOAD LOCK	X	X	<ul style="list-style-type: none"> Request System-bus Probe B-Cache⁴ Same as generic read except if reference to cacheable address space then lock bit set after arbitrating for the System-bus; otherwise no change to lock bit.⁶ (Lock bit VALID = TRUE, Lock Address = access address) (IF (B-Cache hit) THEN change System-bus read to NUT.)² Relinquish System-bus ACK read
STORE COND - Cacheable	X	X	<ul style="list-style-type: none"> Request System-bus Probe B-Cache⁴ Same as generic write to except fails IF (Lock bit VALID = FALSE). (IF (write failed due to lock bit not set) THEN change System-bus write to NUT.) Relinquish System-bus ACK write (conditionally)

²Processor Checks Data EDC.³Processor Checks Tag Store and Control Store Parity.⁴B-Cache Controller Checks Tag Store and Control Store Parity.⁵B-Cache Controller Checks Data EDC.⁶System-bus interface Controller Checks System-bus Parity.

Table 64 (Cont.): Processor Initiated Transactions - Control Flow

Cycle	Datum/Cache		Activity
	Status	Size	
STORE COND. I/O	X	X	<ul style="list-style-type: none"> Request System-bus If the CUCERR_L signal is asserted by the responder module during the first cycle 4 of a System-bus transaction when the 21064 is performing a Store Conditional to non-cacheable address space, then the write will not complete, and the Store conditional failed indicators will be signaled to the processor. Relinquish System-bus ACK write (conditionally)

Processor initiated transactions to non-cacheable address space may be retried by the system bus (CERR_L asserted during cycle 3). When this occurs, the C4 will initiate a retry of the transaction (READ or WRITE) without DECchip 21064 intervention required. Multiple retries are allowed and a transaction will either be retried, complete successfully, or not be acknowledge (error condition). Retry is NOT supported to cacheable address space.

5.3.2 System-bus Initiated

The CPU module System-bus interface controller provides B-Cache control when the the System-bus is active.

The behavior of the System-bus interface controller is determined by the System-bus transaction, and the state of the B-Cache control bits. Table 65 illustrates the control flows for these cycles. The activity column shows the activity that occurs after the processor has relinquished its ownership of the B-Cache and the B-Cache TAG probe results are available.

Table 65: System-bus Initiated Transactions - Control Flow

Cycle	Probe Result	
	Status	Activity
Read HIT B-Cache	Dirty	Assert System-bus DIRTY and SHARED set SHARED bit in Tag Control Store if not already set, cache block remains DIRTY. Provide Data to System-bus ¹²

¹B-Cache Controller Checks Tag Store and Control Store Parity.

²B-Cache Controller Checks Data EDC.

Table 65 (Cont.): System-bus Initiated Transactions - Control Flow

Cycle	Probe Result	
	Status	Activity
Read HIT Duplicate Tag Store, or Lock Address (if valid)	Clean	Assert System-bus SHARED ¹ . Set SHARED bit in Tag Control Store if not already set.
	Clean	Assert System-bus SHARED.
Write HIT	X‡	IF (location found in P-Cache or Commander ID is I/O module or in ARB) THEN Update B-Cache (Clean, shared = no change) and assert System-bus SHARED ELSE Invalidate B-Cache location (clean, not shared) ¹³⁴ . IF (HIT address lock register) THEN clear address lock. or IF (I/O conditional update set) THEN IF (location found in P-Cache) THEN Update B-Cache (Clean, shared = no change) and assert System-bus SHARED ELSE Invalidate B-Cache location (clean, not shared) ¹³⁴ . IF (HIT address lock register) THEN clear address lock.
Exchange HIT	Dirty	Assert System-bus DIRTY and SHARED. Address Lock not changed. Provide Read data to System-bus ¹²³ .
	Clean	Assert System-bus SHARED ¹³ . Address Lock not changed.
Read MISS	NA†	NOP ¹³ . IF (HIT Address Lock register) THEN assert System-bus SHARED.
Write MISS	NA	NOP ¹³⁴ IF (WRITE address hits the address lock) THEN clear address lock.
Exchange MISS	NA	NOP ¹³ . IF (READ address HIT Address Lock register) THEN assert System-bus SHARED. IF (EXCHANGE write address hits the address lock) THEN clear address lock.
NUT	X	NOP ³

¹B-Cache Controller Checks Tag Store and Control Store Parity.²B-Cache Controller Checks Data EDC.³System-bus interface Controller Checks System-bus Parity.⁴System-bus interface Controller Checks Duplicate Tag Store Parity.

†Not Applicable

‡don't care

NOTE

The B-Cache TAG probe during an exchange cycle is for the Read data ONLY.

CHAPTER 6

CACHE INVALIDATE MANAGEMENT

As memory locations in the system are updated, it may become necessary to invalidate cached copies of these locations to maintain cache coherence. The reasons for invalidation can be divided into two major categories. Those due to processor, and those due to C-bus activity.

The Primary I-Stream Cache is a virtual cache and as such all invalidate activity will be completely under software control. Refer to the EV4 Chip Specification for further details.

6.1 Processor Caused Invalidates

The Primary I-stream cache is a virtual cache and its coherence is managed by the system software.

The Primary D-stream cache is a physical cache and is managed to be a subset of the B-Cache. The B-Cache controller may victimize a location in the B-Cache as a result of an I-stream read miss, or a masked write read-merge operation. Thus to maintain the subset rule, the primary D-Cache is invalidated whenever an I-Stream (read allocate) or read-merge victim address hits the duplicate tag store.

System software should disable the Primary D-stream cache while flushing the B-Cache and when flushing is complete, the Primary I-stream cache should also be flushed. This is done to guarantee that the Primary caches remain a strict subset of the B-Cache.

6.2 C-bus Caused Invalidates

Table 66 indicates the cache invalidate policies for C-bus generated cycles.

Table 66: Invalidate Management - C-bus Caused		
Cycle	B-Cache	P-Cache
Read	NOP	NOP

Table 66 (Cont.): Invalidate Management - C-bus Caused

Cycle	B-Cache	P-Cache
Write	If (found in B-Cache and (location NOT found in P-Cache and Commander ID is not I/O subsystem)) THEN Invalidate B-Cache location. †	IF (found in Primary D-cache) THEN invalidate D-cache.
	IF (conditional invalidates for I/O enabled) THEN IF (found in B-Cache and location NOT found in P-Cache) THEN Invalidate B-Cache location.†	IF (found in Primary D-cache) THEN invalidate D-cache.
	IF (ENB COND INVALIDATE cleared) THEN Invalidate B-Cache location.†	IF (found in Primary D-cache) THEN invalidate D-cache.
Exchange	NOP	NOP

†Refer to Section 4.3 which discusses the P-Cache duplicate tag store.

CHAPTER 7

EXCEPTIONS AND INTERRUPTS

When an interrupt or exception occurs the processor drains the pipeline, loads the PC into the EXC_ADDR IPR, and then dispatches to one of the PAL exception routines. If multiple exceptions occur, the 21064 processor dispatches to the highest priority PAL entry point. Table 90 prioritizes the entry points from highest to lowest priority. Refer to Appendix C for further details.

7.1 Processor Generated

General Exceptions

General Exceptions caused by user malfeasance (ie. arithmetic traps or attempted illegal opcode execution), or by normal system operation (for example, translation buffer miss). The list of General Exceptions is as follows:

Table 67: General Exception Isolation Matrix

PAL entry	Cause	Cause Isolation
External Signal RESET L asserted	RESET	NR†
ARITH	Arithmetic Exception (divide by zero etc.)	EXC_SUM IPR
DTB_MISS	Data Translation Buffer Miss	NR†
UNALIGN	D-Stream Unaligned Reference	NR†
DTB_FAULT	Remaining D-Stream Memory Management Errors.	NR†
ITB_MISS	Instruction Translation Buffer Miss	NR†
ITB_ACV	I-Stream Access Violation	NR†
CALLPAL	CALLPAL instruction executed	Entry based on EXC_ADDR->instruction[7..0]
OPDEC	Attempted execution of a reserved or privileged opcode	NR† EXC_ADDR points to instruction
FEN	Floating Point Operation Attempted with Floating Point Unit disabled, Under/Overflows, Inexact Errors, Div Zero, Invalid Ops	IPR EXC_SUM

†Isolation Not Required as PALcode entry identifies cause.

Machine Check Exceptions

Machine Check Exceptions are a special case of exception that are caused by errors in the hardware system. They all dispatch to the general MCHK entry. Refer to Table 68 for the machine check isolation matrix.

Table 68: Machine Check Isolation Matrix

PAL entry	Cause	Cause Isolation
MCHK	BIU detects B-Cache tag store parity error	BIU_STAT IPR
MCHK	BIU detects B-Cache tag control store parity error	BIU_STAT IPR
MCHK	BIU detects B-Cache EDC data store error	BIU_STAT IPR
MCHK	System external transaction terminated with CACK_HERR (Hard Error)	BIU_STAT IPR

NOTE

When a system error occurs all caches in the system should be examined to make sure that a location has not been purposely marked with a “bad” EDC code.

For a more complete description of the processors' hardware error flows refer to Section C.8.

7.1.1 Exception Handling

Most Exceptions have unique entries through which control flows. This allows easy identification, and fast dispatch to the appropriate system code. Exceptions that fall into this category are Reset, Arithmetic, DTB Miss, Unaligned Reference, Data Access Fault, ITB Miss, ITB Access Violation, Reserved Opcode Fault, and Floating Point Operation exceptions. They can occur relatively frequently as part of normal system operation.

The class of exceptions that occur as a result of hardware system errors are called Machine Checks. These result when an uncorrectable system error is detected during the processing of a data request.

Generally exceptions are handled as follows by PALcode. First the PALcode determines the cause of the exception. If possible it corrects the problem and returns the system to normal operation. If a problem is not correctable, or error logging is required, then control is passed through the SCB to the appropriate Exception handler.

7.1.1.1 PAL Priority Level

Below is the prioritized list of the exceptions that can occur on a Sable System. This list goes from the highest to the lowest priority.

Table 69: Exception Priority/PAL Offset/SCB Offset/IPL

Priority	Description	PAL Offset (h)	SCB Offset (h)	IPL (h)
1	HALT	0000	NA	NA
2	Machine Check	0020	0660	31
3		0020	0670	31
4	DTB Miss -PAL	09E0	NA	NA
5	DTB Miss -NATIVE	08E0	NA	NA
6	ITB Miss	03E0	NA	NA
7	ITB Access Violate	07E0	0080-00C0	X
8	Data Access Fault	01E0	0080-00C0	X
9	Unaligned Data	11E0	0300-03F0	X
10	Arithmetic -NFPU	17E0	0010	X
11	Arithmetic -ARTH	17E0	0200	X
12	Reserve Opcode Fault	13E0	0420	X

7.1.1.2 PALcode 0020 Entry Characteristics

Exceptions occur as a direct† result of the detection of errors during the execution of the current instruction. The PALcode found at the PALentry 0020(h) must sift through the system error information and determine the severity of the error. In some cases the PALcode at this entry may correct the error and allow the machine to continue execution without any higher level software intervention.

7.1.1.3 PAL Routine Behavior

7.1.1.3.1 B-Cache Tag Parity Error

† Except during “disconnected” write operations which occur as a result of masked write operations causing two consecutive Cobra-bus transactions. If an error is detected between the completion of the first Cobra-bus transaction, and the second (due to an unrelated intervening Cobra-bus transaction) a machine check will occur and the information relating to the actual error is logged in the C4-CSR's. Software can determine when this occurs by comparing the address in the EV-EXC_ADDR register to the address locked in the C4-CSR's.

Current CPU B-Cache Tag Parity Error Processor Detected

Restricted to reads only, PAL must scrub the parity error from the Tag store using the standard B-Cache initialization procedure, and if the Dirty bit on the cache block in question is set, a SYSTEM FATAL Error should be signaled to the system software. Otherwise only error logging is required.

Refer to Chapter 8, Section 8.2.1 for further details.

7.1.1.3.2 B-Cache Tag Control Parity Error

Current CPU B-Cache Tag Control Parity Error Processor Detected

Restricted to reads only, PAL must scrub the parity error from the Tag Control store using the standard B-Cache initialization procedure. This error is FATAL to the context the cached location is referenced in.

Refer to Chapter 8, Section 8.2.1 for further details.

7.1.1.3.3 B-Cache Data Single Bit EDC Error

Current CPU B-Cache EDC Error Processor Detected

Restricted to reads only, PAL must scrub the EDC error from the Data store using the 21064 Processor B-Cache force Hit mode. Then the instruction that encountered the error should be restarted. A CRD interrupt should be dispatched.

There is no way to determine if the failure was due to a B-Cache SRAM fault or a fault on the bus between the C4 and the 21064 processor chip. If the location found in the B-Cache does not have the same address as the address found in the EV-FILL_ADDR register then the error occurred on the data bus between the 21064 and C4. Generally this type of error is indicative of a hard fault; however if a simple test of the interface passes then the error may be recoverable after scrubbing affected locations and restarting the failing instruction.

Refer to Chapter 8, Section 8.2.2 for further details.

7.1.1.3.4 B-Cache Data Uncorrectable EDC Error

Current CPU B-Cache EDC Error Processor Detected

Restricted to reads only, PAL must scrub the EDC error from the Data store using the 21064 B-Cache force Hit mode. If the Dirty bit on the cache block is set, the error is FATAL to the context the cached location is referenced in. Otherwise the error should be scrubbed from the cache and the failing instruction restarted.

If the location found in the B-Cache does not have the same address as the address found in the EV-FILL_ADDR register then the error occurred on the data bus between the 21064 and C4. Generally this type of error is indicative of a hard fault; however if a simple test of the interface passes then the error may be recoverable after scrubbing affected locations and restarting the failing instruction.

NOTE

If the EDC error has a syndrome of 1F (which indicates an uncorrectable error), this means the data store of the cache was written with an intentionally bad EDC pattern as a result of the CUCERR L signal being asserted during a B-Cache read fill or write update.

Refer to Chapter 8, Section 8.2.2 for further details.

7.1.1.3.5 B-Cache Data Single Bit EDC Error

Current CPU B-Cache EDC Error C4 Detected

Refer to Section 7.1.1.3.3.

7.1.1.3.6 B-Cache Data Uncorrectable EDC Error

Current CPU B-Cache EDC Error C4 Detected

Refer to Section 7.1.1.3.4.

7.1.1.3.7 21064 Data bus Single Bit EDC Error

EDC error occurred on the bus as the data was being driven from the 21064 to the C4. Generally this type of error is indicative of a hard fault; however if a simple test of the interface passes then the error may be recoverable after scrubbing affected locations and restarting the failing instruction.

Refer to Section 7.1.1.3.3.

7.1.1.3.8 21064 Data bus Uncorrectable EDC Error

EDC error occurred on the bus as the data was being driven from the 21064 to the C4. Generally this type of error is indicative of a hard fault; however if a simple test of the interface passes then the error may be recoverable after scrubbing affected locations and restarting the failing instruction.

Refer to Section 7.1.1.3.4.

7.1.1.3.9 B-Cache Tag or Tag Control Parity Error

Current CPU B-Cache Parity Error C4 Detected

First the actual cause of the error should be determined. This is done by calculating the expected Tag Control and Tag parity based on the data latched in the B-Cache Uncorrectable Error Register, and the B-Cache Uncorrectable Error Address Register. The result will indicate whether the error was a Tag Control Store Error, a Tag Store error, or both.

Tag Control Store Errors

Restricted to reads only, PAL must scrub the parity error from the Tag Control store using the standard B-Cache initialization procedure. This error is FATAL to the context the cached location is referenced in.

Tag Store Errors

Restricted to reads only, PAL must scrub the parity error from the Tag store using the standard B-Cache initialization procedure, and if the Dirty bit on the cache block in question was set, a SYSTEM FATAL Error should be signaled to the system software. Otherwise only error logging is required.

Refer to Chapter 8, Section 8.2.1, for further details.

If a tag parity error occurs where an even number of tag bits change state, then during vicimization of a dirty cache block, it is possible to generate a WRITE DATA not ACK'ed error (causing a machine check). When this occurs a SYSTEM FATAL Error should be signaled to the system software. Refer to Section 8.4.3 for further details.

7.1.1.3.10 Cobra-bus Parity Error

Based on logged information and simple R/W to devices on Cobra-bus determine nature of failure. Disable failing module if possible and restart execution otherwise system FATAL.

Refer to Table 77 for further details.

7.1.1.3.11 Invalid Cobra-bus Address

This error occurs when a process has mapped physical addresses in the system that don't exist, or if an error on an address bus or a "double" bit error has occurred in a B-Cache Tag Store. PAL should verify that the error was a legitimately invalid address as a result of incorrect mapping and pass control to the access violation exception handler. If the address logged is a legitimate physical address, and no Cobra-bus parity errors have been logged, then it is SYSTEM FATAL. (Unprotected address bus or double bit parity protected address bus error)

A C/A Parity Error may result in the C/A not ack'ed bit being set in one of the commander CSR's. This is due to the fact that responders will not ack a C/A that has bad parity.

Refer to Table 77 for further details.

7.1.1.3.12 Other CPU Errors

When a Machine Check is initiated as a result of errors on the other CPU module, that module must correct the error(s). B-Cache locations can only be scrubbed using standard B-Cache initialization mechanisms, and these require residence on the CPU local to the failing B-Cache. The other CPU will be notified in the case of these errors automatically via the HARDWARE ERROR INTERRUPT. Refer to Section 7.2.1 for further details.

7.1.1.3.13 Main Memory Uncorrectable EDC Errors

Main Memory Uncorrectable EDC errors are detected only when a Cobra-bus read is responded to by a memory module. The severity of this error depends on the context of the processes that reference it. In USER space then PROCESS FATAL, in SYSTEM space then SYSTEM FATAL.

7.2 Non-processor Generated

Hardware Interrupts are caused by hardware activity that requests the attention of the processor. Every processor in the Sable System will be configured as follows:

Table 70: Hardware Interrupt Configuration

HIR†	Vector	Description
5	none	System Event Interrupt, Node Halt Interrupt
4	none	Interval Timer Interrupt - 976.5625 microseconds
3	none	Inter-processor Interrupt
2	Expansion I/O Interrupt Register‡	Expansion I/O Interrupt
1	T2 Interrupt Register‡	Standard I/O Interrupt
0	none	Hardware Error

†Hardware Interrupt Request - found in the 21064 Processor HIRR IPR

‡Both the Standard I/O Interrupt Registers and Expansion I/O Interrupt Register are accessed over the Cobra-bus and reside off the CPU module. Refer to the Cobra I/O Module Specification for further details.

7.2.1 Interrupt Handling

All system interrupts are funneled through the “Interrupt” PAL entry defined as the 21064 IPR PAL_BASE + “00E0(h)”. From there the appropriate IPL is set and the system is interrogated to determine the cause of the interrupt. When the cause has been determined, the associated SCB offset is added to the SCB base address, and control is passed to that Interrupt Service Routine.

7.2.1.1 PAL Priority Level

Below is the prioritized list of the interrupts that can occur on a Sable System. This list goes from the highest to the lowest priority interrupts.

Table 71: Interrupt Priority/SCB Offset/IPL

PAL Priority	Description	SCB Offset (h)	SRM IPL (h)	PAL IPL (d)
1	Hardware 5 -HLT	0000	31	20
2	Hardware 5 -PWF	0640	30	20
3	Hardware 0 -HRD	0660-0670	31	20
4	Hardware 0 -CRD	0620-0630	20	20
5	Hardware 4	0600	22	20
6	Hardware 3	0610	22	20
7	Hardware 1	800-1FF0	14	20
8	Hardware 2	800-1FF0	14	20
9	Serial Line	800-1FF0	14	20
10	Performance Counter 0	0650	14	20
11	Performance Counter 1	0650	14	20
12	Software 1-15	0500-05F0	1-0F	20
13	Asynchronous System Trap	0240-0270	2	20

7.2.1.2 PALcode 00E0 Entry Characteristics

Various system status and error conditions are reported using one of the many interrupts that all cause entry into the PAL interrupt entry point. Due to the nature of B-Cache errors, the PALcode executed upon entry is restricted to “Read Only” behavior. The following procedure should be followed. Once it has been determined that a B-Cache error has not occurred, the restrictions are lifted.

Whenever a B-Cache error occurs the state of the B-Cache is effectively frozen; however memory system coherence is still maintained. This is done by having cache allocation suspended whenever an error is detected. Refer to Section 4.1.4.2. B-Cache probing by the processor should also be disabled by PALcode by writing a “0” to the BC_ENA bit of the BIU_CTL register.

NOTE

Reading of modifiable memory like data areas should be avoided as these locations could be dirty in the disabled cache and thus produce an incoherent access.

7.2.1.3 Hardware 0 - Hardware Error

Interrupts generating a Hardware Interrupt 0 are caused by the detection of hardware errors on the CPU, I/O, Memory modules, and/or the Cobra-bus. These errors consist of RAM array correctable and uncorrectable errors as well as bus transport and protocol errors. Correctable error interrupts are individually maskable at each modules detection point. Before servicing the Hardware Error Interrupt it should be cleared in the local System Interrupt Clear register.

B-Cache Tag Parity or Uncorrectable EDC Error

A given CPU module must scrub its own B-Cache Tag Parity and Uncorrectable EDC errors. If the error causing this interrupt is a result of this module initiating a transaction, then it also causes a machine check exception and the processing of the error is left up to the machine check handler. If however this node was not the transaction initiator, (Cobra-bus probe) then the interrupt should initiate the scrubbing/logging process.

Parity errors in the Tag or Tag Control Stores of a B-Cache can only be scrubbed using the FORCE EDC/CONTROL and SDV bits of the BCC CSR.

When scrubbing dirty entries from the B-Cache using the Allocate Invalid Address Space, if a Tag, or Tag Control parity error is detected, the expected location will not be scrubbed. If an Uncorrectable data error is encountered, the data will be written to the Cobra-bus coincident with the assertion of the CUCERR L signal. This causes the location to be written into Main Memory with a bad EDC code. The Tag Control Store of the B-Cache location in question will not be updated.

B-Cache Single Bit EDC Error

A given CPU module must scrub its own B-Cache Single bit EDC errors. If the error causing this interrupt is a result of this module initiating a transaction, and if EDC correction is disabled in the B-Cache Control Register, it will also cause a machine check exception and the processing of the error is left up to the machine check handler. If however this node was not the transaction initiator, (Cobra-bus probe) and EDC correction was disabled, then the interrupt should initiate the scrubbing/logging process.

When scrubbing dirty entries from the B-Cache using the Allocate Invalid Address Space, if a Tag, or Tag Control parity error is detected, the expected location will not be scrubbed. If an Uncorrectable data error is encountered, the data will be written to the Cobra-bus coincident with the assertion of the CUCERR L signal. This causes the location to be written into Main Memory with a bad EDC code. The Tag Control Store of the B-Cache location in question will not be updated.

When EDC correction is enabled, no machine checks occur for this error, so the interrupt handler is responsible for scrubbing/logging errors that occur in its own cache.

As compared with hardware error correction, this approach is vulnerable to single-bit errors which may occur during I-stream reads of the PAL code machine check handler, to single-bit errors which occur in multiple quadwords of a cache fill block, and to single-bit errors which occur as a result of multiple silo'ed load misses.

Cobra-bus Parity Error

Cobra-bus Parity Errors indicate that a node on the Cobra-bus has a bad driver or receiver, or a problem exists in the physical interconnect. It is unlikely that this is a correctable error so the handler should attempt to discover which node is bad, and if possible disable it so it won't interfere with normal bus operation.

If no Uncorrectable error can be detected, then retrying the operation causing the error may help to isolate it.

Generally when an uncorrectable error of this type is detected it is system fatal.

A C/A Parity Error may result in the C/A not ack'ed bit being set in one of the commander CSR's. This is due to the fact that responders will not ack a C/A that has bad parity. Refer to Section 7.1.1.3.10 for further details.

Invalid Cobra-bus Address Bystander

When an Invalid Cobra-bus Address is broadcast by a CPU node, and not ack'ed, the Cobra-bus C_ERR L signal is asserted. The CPU that initiated the transaction will machine check and receive a Hardware Error Interrupt; the bystander CPU will only receive the Interrupt. The transaction initiator should handle the logging/error recovery for this error. This results in an access violation. Refer to Section 7.1.1.3.11 for further details.

Invalid Cobra-bus Address I/O Commander

When an Invalid Cobra-bus Address is broadcast by the I/O node, and not ack'ed, the Cobra-bus C_ERR L signal is asserted. Both CPU's will receive a Hardware Error Interrupt; and one should be designated to handle the logging/error recovery.

Information found in the IO error register will indicate whether this error occurred as a result of mailbox operations or DMA accesses and the logging and recovery can be handled differently for each.

Memory Uncorrectable EDC Error I/O Commander

When an Uncorrectable EDC Error is detected while the I/O module is the Cobra-bus Commander, the I/O module asserts the C_ERR L signal which causes the CPU modules to receive a Hardware Error Interrupt. When this occurs, one CPU should be designated to handle the logging/error recovery. This may or may not be SYSTEM FATAL.

Memory Correctable EDC Error

When a Correctable EDC Error is detected the Memory module will assert the C_ERR L signal if the Enable CRD reporting bit is set in Memory module CSR6. This in turn causes the CPU module(s) to receive a Hardware Error Interrupt. When this occurs, one CPU should be designated to handle the scrubbing/logging activity.

Duplicate Tag Store Parity Error

Duplicate tag store parity errors are treated in hardware as uncorrectable errors; however the system will always recover from a parity error of this sort without any loss of data and/or memory coherence.

7.2.1.4 Hardware 1 - Local I/O

Hardware 1 Interrupts are generated as a result of “Local I/O” device interrupt requests. They occur as a result of normal device operation, and when device errors are detected. For a complete description of the unique device interrupt request characteristics, refer to the component specifications for the device in question.

When a Local I/O Interrupt is posted, the PALcode entered will read the Local Interrupt Register located on the I/O module. This register contains a bit field indicating which local I/O device posted the interrupt. (Several device interrupts could be pending.) PALcode dispatches to the appropriate interrupt service routine based on a priority scheme resident in the PALcode environment.

Local I/O Interrupts only are restricted to being serviced by a dedicated processor (identified as the Primary processor). The Primary processor designation can be made at system power-up, or moved from one processor to the other over time. This requires that at any time, the processor not designated the Primary must mask off the Local I/O Interrupt request line bit 1 in the 21064 Processor HIER Internal Processor Register.

7.2.1.5 Hardware 3 - Interprocessor

Hardware 3 Interrupts are requested when a CPU requires the attention of itself or another CPU.

The Interprocessor Interrupt Request Register is used to post an interrupt request to a specific processor. Note that, like software interrupts, no indication is given as to whether there is already an interprocessor interrupt pending when one is requested. Therefore, the interprocessor interrupt service routine must not assume there is a one-to-one correspondence between interrupts requested and interrupts generated. Refer to the ALPHA System Reference Manual for the proper usage and definition of Interprocessor interrupts.

7.2.1.6 Hardware 4 - Interval Timer

The Interval Timer interrupt occurs at a regular interval which allows the processor to effectively schedule processing time to each process requiring attention.

The interval timer interrupt will regularly interrupt the processor every 976.5625 microseconds. The PALcode that handles this interrupt must update its copy of the absolute time, copy it to register R4, clear the interrupt in the local System Interrupt Clear Register, and then pass control to the interval timer interrupt routine.

7.2.1.7 Hardware 5 - System Events

System Events such as power system status changes, halt requests from the Operator Control Panel, or Node Halt, or an SGEN on the I/O module are signaled via the Hardware 5 Interrupt.

Actual Power system status, and Operator Control Panel halt request status must be requested from each subsystem which is accomplished over the System Serial Control bus.

There is no “passive release” mechanism associated with this interrupt so a software must be provide a means to guarantee only one processor services the interrupt.

The pending interrupt must be cleared by explicitly writing “1” to the SYSTEM EVENT or NODE HALT clear bits in the System Interrupt Clear Register - CSR12. Regardless of which processor services this interrupt, this bit must be cleared in all CPU's registers.

7.2.1.8 Software X

Software interrupts provide a mechanism to allow a process to force the flow of control back into the system domain. Their use is determined by the software environment.

7.2.1.9 Serial Line

The Serial Line Interrupt occurs when a change in state on the 21064 serial data receive line changes. Refer to Section 4.13 for further details around the use of the 21064 serial line.

This interrupt should be masked off under normal operation.

7.2.1.10 Performance Counter X

The Performance Counter Interrupts after a specified number of “events” have been counted. The use of these counters is TBD.

7.2.1.11 Asynchronous System Trap

Asynchronous System Traps are a means of notifying a process of events that are not synchronized with its execution, but which must be dealt with in the context of the process.

CHAPTER 8

FAULT MANAGEMENT/ERROR RECOVERY

8.1 Processor Errors

The general error handling mechanisms provide an interrupt to the processor for Uncorrectable and Correctable errors. Uncorrectable errors detected on data accesses force machine check exceptions to occur as the data is being returned to the processor. Correctable errors and latent hardware errors are signaled using the HARDWARE ERROR INTERRUPT.

Refer to Section 8.2, Section 8.3, and Section 8.4 for the complete Cobra CPU module error matrix, and Chapter 7 for the error parse trees.

8.2 B-Cache Errors

8.2.1 Tag and Tag Control Store Parity Errors

B-Cache Tag and Tag Control Parity Errors are only detected when the Tag and Tag Control Stores of the B-Cache are read (probed). The error handling mechanism used when an error is encounter is based not only on whether the B-Cache Controller or the 21064 detected the error, but also what cycle was being performed.

Parity Errors in the Tag and Tag Control Stores may or may not be fatal errors to the system, or for that matter to non-system processes. Refer to Table 72 for the complete error severity matrix.

Table 72: Tag/Tag Control Error Severity Matrix

Error	Address Context	Dirty	Severity
Tag Parity	Unknown	Yes	System Fatal
	Unknown	No	Scrub location - recoverable
Tag Control Parity	User†	Unknown	Process Fatal - Scrub location

†Virtual Page mapping could have been changed between the time the error occurred and the time it was examined, therefore a deterministic address context can not be established. In lieu of a deterministic means for identifying address context, system software should consider any Tag Control Parity Error, System Fatal.

Table 72 (Cont.): Tag/Tag Control Error Severity Matrix

Error	Address Context	Dirty	Severity
	System	Unknown	System Fatal

The 21064 processor probes the B-Cache Tag and Tag Control Stores whenever a Load, Load Lock, Store, or Store Conditional cycle is requested to a cacheable location. If the 21064 processor discovers a parity error during this probe the following sequence of events will occur:

1. the transaction will be forced to miss the external cache
2. BC_TAG (processor register) will hold the results of the external cache tag probe
3. The Machine Check PALcode is Invoked
4. BIU_STAT: BIU_TCPERR (if Tag Control Parity Error) and/or BIU_TPERR (if Tag Parity Error) are Set
5. BIU_ADDR holds the physical address of the probe where the parity error was detected

The B-Cache Controller detects Tag and Tag Control Parity Errors under several different circumstances. These include the origin of the cycle as well as the cycle type. Table 73 shows the complete error handling matrix for B-Cache Tag and Tag Control Store Parity errors.

A cycle request can originate from either the System-bus or the 21064

When a Tag or Tag Control Store Parity Error is detected during a B-Cache probe due to the System-bus initiated transaction, the probe results in a “Fail” condition. When this occurs the B-Cache Controller will log the error in the BCUE register, the corresponding address in the BCUEA register, and will assert the C_ERR L signal during the next t0 or idle System-bus cycle. Refer to the Cobra System Bus Specification for the details of the C_ERR L signal.

When a Tag Control or Tag Store parity error is detected on a System-bus read probe, the CPU detecting the error will assert the CUCERR L signal coincident with the data being returned from main memory on the System-bus.

The B-Cache Controller will **not**:

- Accept data from the System-bus if the System-bus transaction is a write
- Supply data to the System-bus even if the address probes HIT, DIRTY
- Modify the contents of the B-Cache, Tag, or Tag Control Stores

Once this error has been detected, and the BCUE error bits set, the allocation of the B-Cache is disabled and will remain disabled until all of the BCUE error bits have been cleared, and the BCC ENB_ALLOCATE bit set.

When a Tag or Tag Control Store Parity Error is detected during a transaction requested by the 21064, such as Read Block, Load Lock, Write Block, or Store Conditional the resultant System-bus transaction is converted to a NUT transaction. The BCUE and BCUEA registers log the error, and allocation in the B-Cache is disabled.

Data is neither read from the System-bus (for read type transactions) nor data written on the System-bus (for write type transactions). The state of the B-Cache remains unchanged.

The cycle requested by the 21064 is acknowledged with the **HARD_ERROR** response which invokes the Machine Check PALcode, and the **C_ERR L** signal is asserted during the next t0 or idle System-bus cycle.

Table 73: B-Cache Tag Control or Tag Store Errors

Transaction Causing Probe	Error Recovery Activity
System-bus address Probe	<p>In B-Cache Uncorrectable Error (BCUE) PAR ERROR bit set, In B-Cache Uncorrectable Error Address (BCUEA) Register System-bus address is latched, The allocation of the cache is disabled. Subsequent address probe related errors are only logged as missed errors until the BCUE error bits have been cleared. If the System-bus address probe was due to a read or exchange cycle then the CUCERR_L is asserted as the read data is returned to the transaction commander. This is done even if the module discovering the error was a bystander in the System-bus transaction.</p>
DECchip 21064 Read Block, write block,load lock, store conditional	<p>Force NUT transaction on the System-bus, assert System-bus C_ERR L signal, Log the error in BCUE and BCUEA registers and freeze them, Return cycle acknowledgment to the 21064 of HARD_ERROR, Allocation of the cache is disabled. Subsequent address probe related errors are only logged as missed errors until the BCUE error bits have been cleared.</p>
C/A NOT ACKED bit will be set in CBE Register and the WRITE DATA NOT ACKED bit may be set indicating an extraneous error.	

8.2.2 Data Store EDC Errors

8.2.2.1 Correctable

Correctable Data Store Errors can only be detected when either the B-Cache Controller or the 21064 processor read the B-Cache Data Store. Or when an EDC error occurs on the bus between the C³ chip and the 21064 processor. The mechanism used to report the error is different depending on which subsystem detected it. All Correctable errors are fully recoverable. Those detected by the B-Cache Controller will be corrected “on the fly” and the normal flow of the transaction requested will complete. Those detected by the 21064 processor will cause a machine check to allow software to correct the corrupted data.

The 21064 will detect Correctable EDC errors only when executing a Load, or Load Lock instruction. When the error is detected the following sequence of events will occur:

1. Data put into I or D-Cache (appropriately) unchanged, block gets validated.
2. Machine check
3. BIU_STAT: FILL_EDC set, and FILL_IRD set for I-Stream reference cleared for D-Stream. (FILL_SEQ set if multiple errors occur.)
4. FILL_ADDR[33:5] & BIU_STAT[FILL_QW] gives bad quadword's address
5. If D-Stream FILL_ADDR[4..2] contain PA bits [4..2] of location which the failing load instruction attempted to read.
6. FILL_SYNDROME contains syndrome bits associated with the failing quadword.
7. BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
8. If I-Stream DC_STAT locked - contents are UNPREDICTABLE, If D-Stream DC_STAT locked, RA identifies register which holds the bad data, LW, LOCK, INT, VAX_FP identify type of load instruction
9. BC_TAG holds the results of the external cache tag probe if the external cache was enabled for this transaction.

If the physical address of the location with the Correctable error is not resident in the B-Cache then the error was a bus error and the transaction should just be replayed, otherwise the B-Cache location should be scrubbed and then the transaction replayed.

The B-Cache controller could detect the error under several different circumstances, and the way it is handled depends on how it was discovered. Table 74 shows the complete error handling matrix for B-Cache Data Correctable errors detected by the B-Cache controller. Correctable errors are detected/reported when a System-bus Reads HIT Dirty Locations, a B-Cache entry is victimized, or a masked write to a shared location.

All Correctable errors detected by the B-Cache Controller are corrected to allow the normal operation to proceed as if no error occurred. If the physical address of the location with the Correctable error is not resident in the B-Cache then the error was bus error and no scrubbing is required, otherwise the B-Cache location should be scrubbed.

Table 74: B-Cache Data Correctable Errors

Transaction Causing Probe	Access Reason	Error Recovery Activity
System-bus READ (Dirty) or Exchange (Read Dirty)	System-bus READ/ EXCHANGE Dirty	The B-Cache Correctable Error (BCCE) and B-Cache Correctable Error Address Registers (BCCEA) are frozen and the CORRECTABLE ERROR bit set, if the BCC ENB B-CACHE COR ERR INTERRUPT bit is set, the System-bus C_ERR L signal is asserted.
System-bus Write	—	—
DECchip 21064 Read Block	Victim Processing	The BCCE and BCCEA registers are frozen and the CORRECTABLE ERROR bit set if the BCC ENB B-CACHE COR ERR INTERRUPT bit is set, the System-bus C_ERR L signal is asserted.
DECchip 21064 Write Block	Victim Processing / Shared Masked Write / Unmasked Write read allocate	The BCCE and BCCEA registers are frozen and the CORRECTABLE ERROR bit set if the BCC ENB B-CACHE COR ERR INTERRUPT bit is set, the System-bus C_ERR L signal is asserted.
DECchip 21064 Load Lock	Victim Processing	The BCCE and BCCEA registers are frozen and the CORRECTABLE ERROR bit set, if the BCC ENB B-CACHE COR ERR INTERRUPT bit is set, the System-bus C_ERR L signal is asserted.
DECchip 21064 Store Conditional	Victim Processing / Shared Masked Write	The BCCE and BCCEA registers are frozen and the CORRECTABLE ERROR bit set if the BCC ENB B-CACHE COR ERR INTERRUPT bit is set, the System-bus C_ERR L signal is asserted.

8.2.2.2 Uncorrectable

Uncorrectable Data Store Errors can only be detected when either the B-Cache Controller or the 21064 processor read the B-Cache Data Store. Or when an EDC error occurs on the bus between the C³ chip and the 21064 processor. The mechanism used to report the error is different depending on which subsystem detected it.

The severity of the error to the systems integrity depends on the address space the error occurred in as well as the state of the location in the B-Cache.

Table 75: Uncorrectable Data Store Error Severity Matrix

Address Context	Dirty	Severity
User	Yes	Process Fatal if corrupted data used - Scrub
	No	Scrub location - recoverable
System	Yes	System Fatal if corrupted data used
	No	Scrub location - recoverable

The 21064 will detect Uncorrectable EDC errors only when executing a Load, or Load Lock instruction. When the error is detected the following sequence of events will occur:

1. Data put into I or D-Cache (appropriately) unchanged, block gets validated.
2. Machine check
3. BIU_STAT: FILL_EDC set, and FILL_IRD set for I-Stream reference cleared for D-Stream. (FILL_SEQ set if multiple errors occur.)
4. FILL_ADDR[33:5] & BIU_STAT[FILL_QW] gives bad quadword's address
5. If D-Stream FILL_ADDR[4..2] contain PA bits [4..2] of location which the failing load instruction attempted to read.
6. FILL_SYNDROME contains syndrome bits associated with the failing quadword.
7. BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
8. If I-Stream DC_STAT locked - contents are UNPREDICTABLE, If D-Stream DC_STAT locked, RA identifies register which holds the bad data, LW, LOCK, INT, VAX_FP identify type of load instruction
9. BC_TAG holds the results of the external cache tag probe if the external cache was enabled for this transaction.

When detected by the 21064 (as part of the machine check handler), if the physical address of the location with the Uncorrectable error is not resident in the B-Cache then the error was a bus error and should be re-tried.

The B-Cache controller could detect the error under several different circumstances, and the way it is handled depends on how it was discovered. Table 76 shows the complete error handling matrix for B-Cache Data Uncorrectable errors detected by the B-Cache controller. Uncorrectable errors are detected/reported when a System-bus Reads HIT Dirty Locations, a B-Cache entry is victimized, or a masked write to a shared location occurs.

When detected by the B-Cache Controller, if the physical address of the location with the Uncorrectable error is not resident in the B-Cache then the error was a bus error and severity of the error is the same as if a DIRTY B-Cache location was found with an Uncorrectable Error - this location should be flushed to main memory where it will be written with BAD EDC.

Table 76: B-Cache Uncorrectable Errors

Transaction Causing Probe	Access Reason	Error Recovery Activity
System-bus READ (Dirty) or Exchange (Read Dirty)	System-bus READ / EXCHANGE Dirty	Assert System-bus CUCERR L with the data returned to the System-bus. The B-Cache Uncorrectable Error (BCUE) and B-Cache Uncorrectable Error Address Registers (BCUEA) are frozen and the UNCORRECTABLE ERROR bit set.
System-bus Write Probe	If HIT System-bus write data has either bad parity or has CUCERR L	The B-Cache is written with BAD EDC.
DECchip 21064 Read Block	Victim Processing	The System-bus C_ERR L signal is asserted, the victim written to memory coincident with the assertion of CUCERR L and the cycle is acknowledged to the processor with HARD_ERROR. The B-Cache is not updated, and B-Cache allocation is disabled. The BCUE and BCUEA registers are frozen and the UNCORRECTABLE ERROR bit set.
	System-bus returned data with bad System-bus parity or CUCERR L asserted with read data	The System-bus C_ERR L signal is asserted, bad "second" read data from the System-bus is returned to the processor, bad "second" data from the System-bus is written with "bad EDC" in the B-Cache, The BCUE and BCUEA registers are frozen and the UNCORRECTABLE ERROR bit set, processor is acknowledged with HARD_ERROR.
DECchip 21064 Write Block	Victim Processing / Shared Masked Write	The System-bus C_ERR L signal is asserted, the victim / shared write is written to memory coincident with the assertion of CUCERR L, and the B-Cache is updated with the new data. In the case of a shared masked write, the B-Cache longwords with failing EDC that are not updated by the processor write, will not be re-written. The BCUE and BCUEA registers are frozen and the UNCORRECTABLE ERROR bit set.

Table 76 (Cont.): B-Cache Uncorrectable Errors

Transaction Causing Probe	Access Reason	Error Recovery Activity
DECchip 21064 Load Lock	Victim Processing	<p>The System-bus C_ERR L signal is asserted, the victim is written to memory coincident with the assertion of CUCERR L, and the read data is returned from the System-bus to the processor.</p> <p>The BCUE and BCUEA registers are frozen and the UNCORRECTABLE ERROR bit set.</p>
DECchip 21064 Store Conditional	Victim Processing	<p>The System-bus C_ERR L signal is asserted, the victim is written to memory coincident with the assertion of CUCERR L, and the B-Cache is updated with the new data.</p> <p>The BCUE and BCUEA registers are frozen and the UNCORRECTABLE ERROR bit set.</p> <p>If store fails then no data cycles will occur; however C_ERR L will still be asserted.</p>

8.3 Duplicate P-Cache Tag Store Parity Errors

Duplicate P-Cache Tag Store Parity Errors can only be detected by the B-Cache controller when a System-bus write occurs.

When an error is detected, the Cobra System-bus C_ERR L signal is asserted, and the B-Cache Duplicate Tag Store Error Register (DTSER) frozen, and the Error bit set. Detection of the parity error will force the invalidation of the associated Primary D-Cache location, and if HIT, the invalidation of the associated B-Cache location regardless of System-bus transaction commander ID.

Thus single bit errors in the Duplicate Tag Store are not system fatal; however a HARDWARE ERROR INTERRUPT will occur every time one is encountered.

8.4 System-bus Errors

System-bus errors may not be reportable as the error handling routines are most likely located in main memory. Only when the System-bus is still operable, and main memory has not been corrupted will System-bus errors be reportable.

8.4.1 C/A Parity Error

When a System-bus node detects a C/A parity error, it will log the error, signal the system by asserting the C_ERR L signal and ignore the rest of the System-bus transaction.

Table 77: C/A Parity Errors

Commander	Responder	Bystander
C/A cycle not acknowledged Reply with a HARD_ERROR failure status to the processor Freeze System-bus Error Address Registers, set the appropriate bits in the System-bus Error Register, and assert the System-bus C_ERR L signal	N/A (No responder) D-Cache must be flushed as duplicate tag store is incoherent.	Freeze System-bus Error Address Registers, set the appropriate bits in the System-bus Error Register, and assert the System-bus C_ERR L signal

8.4.2 Data Parity Error

A System-bus data parity error can be detected under two circumstances as a responder and as a bystander.

A responder should not “ACK” the data transfer which will indicate to the commander that the transaction failed due to a parity error. The commander will assert the C_ERR L signal to notify the system of the error.

A bystander will check System-bus data parity when a System-bus write probe indicates a B-Cache HIT, and the System-bus write data will be used to update the B-Cache. If this “bystander” detects bad parity, the cache location in question will be marked invalid and the C_ERR L signal will be asserted to notify the system of the error.

The corrupted read data will be placed in the B-Cache with “correct” the EDC code.

Table 78: Data Parity Errors

Commander	Responder	Bystander
Read data - respond with HARD_ERROR to the processor	Write data - don't ACK the data Freeze System-bus Error Address Registers, set the appropriate bits in the System-bus Error Register	If accepting data on a System-bus write update then mark the B-Cache location invalid Freeze System-bus Error Address Registers, set the appropriate bits in the System-bus Error Register, and assert the System-bus C_ERR L signal.
Write data - not ACK'ed by responder, respond with HARD_ERROR to the processor, assert C_ERR L		

8.4.3 Invalid Address - Bus Time-out

An invalid address is identified as an address that is not acknowledged by any responder. Refer to the Cobra System Bus Specification.

When a commander detects an Invalid address on the System-bus, it will freeze the System-bus Error Address Registers, set the appropriate bits in the System-bus Error Register, assert the System-bus C_ERR L signal, and responds to the processor with HARD_ERROR which initiates a machine check.

An invalid address is detected when a C/A is not acknowledged and no System-bus node indicates that a parity error has occurred or when a dirty victim with a bad tag value (possible double bit error) is written to memory with an EXCHANGE address that is outside currently configured memory space. This has occurred when the WRITE DATA not ACK'ed bit is asserted, and the EXHANGE field logged in the CBEAH register indicates an invalid address.

8.5 I/O Subsystem Errors

All errors occurring in the I/O subsystem will be signaled using the CIRQ_{<0>} and CIRQ_{<1>} interrupts and all System-bus errors detected by the I/O subsystem are signaled using the C_ERR L signal.

8.6 C_ERR L Assertion

When the B-Cache controller detects the assertion of the C_ERR L System-bus signal it asserts the CPU module HARDWARE ERROR INTERRUPT.

CHAPTER 9

CPU POWERUP AND INITIALIZATION

This chapter describes the behavior of the CPU module when System-bus RESET_L deasserts. The initial states of the processor and module registers are described.

9.1 Processor Initialization

9.1.1 Internal Processor Registers

Refer to Section C.6 for a description of the processors internal registers powerup state.

9.1.2 Internal JSR stack

Refer to Section C.6 for a discussion regarding the initialization of the jsr stack via PALcode.

9.2 B-Cache Initialization

B-Cache Initialization is performed by the DECchip 21064 processor. This is accomplished as follows:

1. Write to CSR15 to configure desired values for speed bits and enable CBUS retry.
2. Set Cache Size set to the appropriate value in BCC
3. Clear BC_EN in the EV BIU_CTL Register
4. Clear Fill Wrong Parity and disable tag and tag control store parity checking by clearing the ENB TAG & DUP TAG PAR CHK bits in the BCC register.
5. Set EDC H, EDC L, and SHARED, DIRTY, VALID (BCC) to desired values.
6. Set ENABLE B-CACHE INIT (H) in the BCC Register (<43>,<11>)
7. Set ENABLE ALLOCATE (H) in the BCC Register (<32>,<0>)
8. Perform LDQ from Bcache Init space starting at address 1.0000.0000 address range and continue up in 32 byte increments until the address range equal to the size of the cache in the system has been exhausted.

The B-Cache Control Store will then contain the values for Shared, Dirty, and Valid provided in the BCC register before initialization. The B-Cache Tag Store will contain a Tag equivalent to the high order address bits specified during the 9th step above. Each quadword in the B-Cache Data Store of a particular cache block will contain data identical to the data returned during the read (format indicated in Section 9.2.1.

Once the B-Cache RAMS have been initialized the B-Cache Control/Status Register should be set to 0000.01C5.0000.01C5 for a 1MB backup cache module, or 8000.01C5.8000.01C5 for a 4MB backup cache module. This configures the module as follows:

4-meg CACHE SIZE	set/cleared	BCC<63,31>
FORCE EDC/CONTROL	Cleared	BCC<44,12>
ENB B-CACHE INIT	Cleared	BCC<43,11>
ENB B-CACHE COND I/O UPDATES	Cleared	BCC<41,9>
ENB B-CACHE EDC CHK	Set	BCC<40,8>
ENB B-CACHE CORRECTION	Set	BCC<39,7>
ENB B-CACHE COR ERR INTERRUPT	Set	BCC<38,6>
FILL WRONG DUP TAG STORE PAR	Cleared	BCC<37,5>
FILL WRONG CONTROL PAR	Cleared	BCC<36,4>
FILL WRONG TAG PAR	Cleared	BCC<35,3>
ENB TAG and DUP TAG PAR CHK	Set	BCC<34,2>
FORCE FILL SHARED	Cleared	BCC<33,1>
ENABLE ALLOCATE	Set	BCC<32,0>

All error bits in the B-Cache Correctable, Uncorrectable, and Duplicate Tag Store Error Registers should be cleared.

9.2.1 LDQ Data Format - BCC ENABLE B-CACHE INIT Set

Figure 66: LDQ Data Format (LDQ_DF)

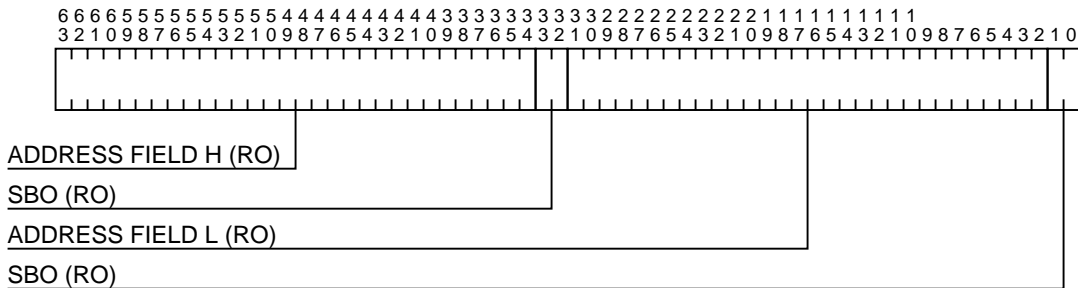


Table 79: LDQ Data Format Description

Field	Description
63:34	ADDRESS FIELD H [<i>read-only</i>] Contains the C/A address field for this cycle. Which is equal to the address of the data reference of the LDQ.
33:32	SBO [<i>read-only</i>] These bits should be zero.
31:2	ADDRESS FIELD L [<i>read-only</i>] Contains the C/A address field for this cycle. Which is equal to the address of the data reference of the LDQ.
1:0	SBO [<i>read-only</i>] These bits should be zero.

9.3 Duplicate Tag Store Initialization

The Duplicate Tag Store will be reset to all zero's and correct parity on powerup. No user intervention is required.

9.4 System-bus Interface Initialization

1. The System-bus Control Register should be initialized so that System-bus parity checking is enabled and all other write-able bits cleared. As well as setting the SELECT DRACK and 2nd QW SELECT bits to the appropriate values.
2. All error bits in the System-bus Error Register should be cleared.
3. Retry should be enabled and speed-bits set to the appropriate values.

9.5 CPU clocks and reset

System-bus reset will assert asynchronously with respect to the System-bus clocks, and will deassert synchronously with PHI1. The CPU module has been designed to expect the clocks to be free running during reset. Refer to Cobra System Bus Specification and Section 4.15 for further details.

9.6 Power-up Sequence

The system power supply is required to bring 3.3V up before 5V. If 3V goes below regulation, the DCOK signal to EV will be deasserted.

9.7 Powering Up with Bad Main Memory

As the processors B-Cache update policy is designed to accept the data of an I/O module write (when the location is in the B-Cache). It is possible to simultaneously load the B-Cache with the console image as it is being writing to a broken memory module.

This is done by initializing the B-Cache Tag Store with the tag values of the memory region being written, and the Tag Control Store Valid, Not Shared, Not Dirty. Also by creating a mailbox structure in the B-Cache and initializing its Tag Control Store to Valid, Not Shared, Dirty the processor will respond with the mailbox data when the I/O module requests it. The only function required by the memory module is to acknowledge the System-bus transactions as they occur, which means it should be initialized as though it was functioning normally.

The console image can then be started as if it was located on a memory module, and just moved into the B-Cache via the normal cache allocation processes handled in hardware.

CHAPTER 10

OVERVIEW OF THE CPU TESTABILITY FEATURES

This chapter contains an overview of the hooks added to the CPU module to provide an easier means to verify/debug them. The intended audience is diagnostic engineers; however anyone involved in the manufacture/debug of the module may find it useful.

10.1 Bus Verification

10.1.1 Address

As part of the B-Cache Init Mode (BCC register) the address bus is looped back through the C4 onto the address bus. Thus the addresses presented to the C4 will be returned as read data to the processor. This enables the serial ROM firmware to verify the integrity of the address bus.

It should be noted that whenever a C4 CSR is accessed the data is broadcast on the Cobra-bus, therefore if there is a failure on the Cobra-bus these cycles could become corrupted.

10.1.2 Data

By reading and writing arbitrary patterns to the mailbox register it is possible to verify the integrity of the data bus all the way to the C4.

It should be noted that whenever a C4 CSR is accessed the data is broadcast on the Cobra-bus, therefore if there is a failure on the Cobra-bus these cycles could become corrupted.

10.2 C4 EDC Generators

The EDC Generator in the C4 can be tested quite simply by reading and writing arbitrary patterns to the Processor Mailbox register. When this or any CSR is read (locally CPU CSR's only) the data returned is not only presented on the low order quadword of the cache block, but also every other quadword. This way all EDC generators can be verified by reading both low and high order quadwords of a given CPU module local CSR.

10.3 C4 EDC Checkers

The EDC Checkers/correctors in the C4 can be verified by writing various data patterns in the B-Cache with correct and incorrect EDC values. (Using FORCE EDC/CONTROL BCC functionality) Then forcing them to be writing to memory, using allocation invalid mode or some other means to victimize dirty cache blocks.

10.4 DECchip 21064 EDC Checkers

The EDC Checkers in DECchip 21064 can be verified by writing various data patterns in the B-Cache with correct and incorrect EDC values, (Using FORCE EDC/CONTROL BCC functionality) and then reading them in force hit mode.

10.5 DECchip 21064 EDC Generators

The EDC Generators in DECchip 21064 can be verified by enabling EDC mode and writing various data patterns that are known to miss the B-Cache.

10.6 C4 Cobra-bus Probe Predicted Tag Parity Generator

This parity generator calculates the parity of the normal address portion presented on the Cobra-bus across the field of bits relating to the address bits stored in the modules tag store. By causing various Cobra-bus cycles to occur at various addresses, resulting in errors, this parity tree can be completely verified.

10.7 B-Cache Data Store Verification

The B-Cache Data Store can be verified by setting the DECchip 21064 processor in force hit mode on the B-Cache (EV-BIU_CTL). This enables the user to access the B-Cache as if it was just a physically contiguous section of memory starting at 0 and ending at the limits of its size. (It is also ghosted on up through the complete DECchip 21064 address space.) Arbitrary patterns can be read and written.

When testing the EDC RAMs in the B-Cache Data Store, it is advised to use the B-Cache Init Mode with the FORCE EDC/CONTROL functionality. This allows the user to simply write arbitrary patterns in the EDC RAMs without regard to the actual EDC code.

10.8 B-Cache Tag/Control Store Verification

The B-Cache Tag and Tag Control Stores can be verified by writing different patterns into each Tag/Tag Control Store location. This is accomplished using the B-Cache Init Mode. (Refer to the BCC register for details.) When this mode is enabled, by prudently choosing the addresses read, the user can completely test the tag store. Using the FORCE EDC/CONTROL functions, all the Tag Control Store locations can also be tested.

10.9 Performance Counters

There are two performance counters on the DECchip 21064 processor. They can be configured to count various processor internal events. Externally these have been connected to pulldown resistors on the CPU module.

10.10 Brain Dead Module Errors

When the reset signal going to the DECchip 21064 processor deasserts, the processor begins reading serial ROM 1. This continues until the complete ROM had been read which causes the Serial ROM 1 Read indicator of the D-bus microcontroller to be asserted. As there is a fixed time between when reset deasserts by adding a time out between D-bus micro powerup and Serial ROM 1 Read, it is possible to signal to the Operator Control Panel via the Serial Control Bus that the DECchip 21064 processor is dead. Depending on the firmware in the serial ROMs the code running in the D-bus micro may watch for one or all serial ROM read signals to be asserted before indicating CPU module failure.

Table 80: Kernel Survivability Matrix

Node	Failing Element				Survivability
	Cobra-bus OSC	DECchip 21064 OSC	DECchip 21064†	3.3V	
CPU	X				D-bus micro clock also dead so no ability to signal failure.
CPU		X			D-bus micro can detect this as Clock/Voltage Detect is clear, serial ROM read signals not asserted before time out period, and DECchip 21064 processor Re-setting asserted. When occurs signal OCP and log in EEPROM.
CPU			X		D-bus micro can detect this as serial ROM read signals not asserted by time out of TBDms. Can communicate with OCP and log in CPU module EEPROM.
CPU				X	Possible to detect if problem exists in CPU ₁ 3.3V detect circuit. If 3.3V actually failed, as all 3.3V comes from the backplane CPU ₀ would detect failure and CRESET L would assert taking D-bus micro reset with it. CRESET(⇒ D-bus micro reset). Send to OCP and log in EEPROM.

†DECchip 21064 unable to finish reading serial ROM(s)

Once the Serial ROMs have been read, the D-bus micro will enter into dialogue with the DECchip 21064 processor over the D-bus. Thus if the DECchip 21064 processor is unable to make it to the FEPRoms on the I/O module it is able to log the errors and communicate them to the OCP.

CHAPTER 11

PHYSICAL AND ELECTRICAL CHARACTERISTICS

11.1 CPU Module Physical Specification

There are three attached components to the Sable CPU module which should be noted.

Two ejectors are attached in two corners to facilitate insertion and removal of the modules.

A stiffener is attached along the edge with the ejectors to reduce bowing and add rigidity during insertion and removal

There may be an air flow accelerator attached to the module to direct air flow to the more critical components.

11.2 SABLE CPU CONNECTOR PINNING

Table 81: MCA240 J5

SIDE 1		SIDE 2	
1	CAD<127>H	121	
2	CAD<124>H	122	CAD<126>H
3	GND	123	GND
4	CAD<125>H	124	CAD<120>H
5	CAD<122>H	125	CAD<116>H
6	CAD<123>H	126	CAD<118>H
7	GND	127	GND
8	CAD<121>H	128	CAD<113>H
9	CAD<119>H	129	CAD<111>H
10	CAD<110>H	130	CAD<115>H
11	GND	131	GND
12	CAD<117>H	132	CAD<109>H
13	CAD<108>H	133	CAD<114>H

Table 81 (Cont.): MCA240 J5

SIDE 1		SIDE 2	
14	CAD<112>H	134	CAD<107>H
15	GND	135	GND
16	CAD<105>H	136	CAD<106>H
17	CAD<104>H	137	CAD<103>H
18	CAD<100>H	138	CAD<102>H
19	GND	139	GND
20	CAD<101>H	140	CAD<98>H
21	CAD<99>H	141	CPARITY<3>L
22	CAD<61>H	142	CAD<97>H
23	GND	143	GND
24	CAD<57>H	144	CAD<63>H
25	CAD<48>H	145	CAD<59>H
26	CAD<55>H	146	CAD<62>H
27	GND	147	GND
28	CSHARED L	148	CSTALL1 L
29	GND	149	GND
30	CAD<38>H	150	CAD<58>H
31	CAD<96>H	151	CAD<41>H
32	CAD<44>H	152	CAD<47>H
33	GND	153	GND
34	CAD<49>H	154	CAD<53>H
35	CAD<51>H	155	CAD<56>H
36	CAD<60>H	156	CAD<54>H
37	CAD<36>H	157	CAD<40>H
38	GND	158	GND
39	CAD<37>H	159	CAD<43>H
40	CPARITY<1>L	160	CAD<50>H
41	CSYS_EVENT L	161	CAD<52>H
42	GND	162	GND
43	CAD<34>H	163	CAD<42>H
44	CAD<39>H	164	CAD<46>H
45	CAD<32>H	165	CAD<45>H
46	CAD<35>H	166	CAD<33>H
47	GND	167	GND
48	CUCERR L	168	C_ERR L
49	BCREQ L	169	4BCREQ1 L
50	GND	170	GND
51	CAD<95>H	171	CAD<94>H
52	CAD<92>H	172	CAD<88>H
53	CAD<93>H	173	CAD<84>H

Table 81 (Cont.): MCA240 J5

SIDE 1		SIDE 2	
54	GND	174	GND
55	CAD<90>H	175	CAD<86>H
56	CAD<91>H	176	CAD<81>H
57	CAD<89>H	177	CAD<79>H
58	GND	178	GND
59	CAD<87>H	179	CAD<83>H
60	CAD<78>H	180	CAD<77>H
61	CRESET L	181	CAD<74>H
62	GND	182	GND
63	CAD<82>H	183	CAD<85>H
64	CAD<76>H	184	CAD<75>H
65	CAD<73>H	185	CAD<71>H
66	GND	186	GND
67	CAD<80>H	187	CPARITY<2>L
68	CAD<68>H	188	CAD<66>H
69	CAD<72>H	189	CAD<70>H
70	GND	190	GND
71	CPUG L(<i>old</i>)	191	CA L(<i>old</i>)
72	RCPUG L	192	CA L
73	CXACK L	193	CSTALL0 L
74	GND	194	GND
75	DCA L	195	DSTALL0 L
76	CAD<67>H	196	CAD<65>H
77	CDIRTY L	197	CAD<69>H
78	GND	198	GND
79	CAD<29>H	199	CAD<31>H
80	CAD<27>H	200	CAD<30>H
81	CAD<64>H	201	CAD<25>H
82	GND	202	GND
83	CAD<23>H	203	CAD<26>H
84	CAD<28>H	204	CAD<16>H
85	CAD<21>H	205	CAD<24>H
86	GND	206	GND
87	CAD<15>H	207	CAD<22>H
88	CAD<19>H	208	CAD<17>H
89	CAD<9>H	209	CAD<12>H
90	GND	210	GND
91	CAD<18>H	211	CAD<20>H
92	CAD<6>H	212	CAD<11>H
93	CAD<5>H	213	CPARITY<0>L

Table 81 (Cont.): MCA240 J5

SIDE 1		SIDE 2	
94	GND	214	GND
95	CAD<8>H	215	CAD<4>H
96	CAD<10>H	216	CAD<14>H
97	CAD<7>H	217	CAD<2>H
98	CAD<3>H	218	CAD<13>H
99	GND	219	GND
100	CAD<0>H	220	CAD<1>H
101	CPUREQ L(<i>old</i>)	221	IOG L
102	XIOREQ L	222	IOREQ L
103	GND	223	GND
104	XIOG L	224	CINT_TIM H
105	ID2 H	225	CIRQ<1>L
106	ID1 H	226	CIRQ<0>L
107	IO0 H	227	ASYNC_RESET L
108	GND	228	GND
109	CPUREQ3 L	229	ABCREQ1 L
110	CPUREQ2 L	230	ABCREQ1 L
111	GND	231	GND
112	CPUG3 L	232	ABCREQ0 L
113	CPUG2 L	233	ABCREQ0 L
114	GND	234	GND
115	CPUG1 L	235	CPUREQ1 L
116	CPUG0 L	236	CPUREQ0 L
117	GND	237	GND
118	PHI1 L	238	PHI3 L
119	GND	239	GND
120	PHI1 H	240	PHI3 H

Table 82: MCA44 J1

Side 1		Side 2	
1	VCC	23	VCC
2	VCC	24	VCC
3	VCC	25	VCC
4	GND	26	GND
5	GND	27	GND
6	GND	28	GND
7	VCC	29	VCC
8	VCC	30	VCC

Table 82 (Cont.): MCA44 J1

Side 1		Side 2	
9	VCC	31	VCC
10	GND	32	GND
11	GND	33	GND
12	GND	34	GND
13	VCC	35	VCC
14	VCC	36	VCC
15	VCC	37	VCC
16	GND	38	GND
17	GND	39	GND
18	GND	40	GND
19	VCC	41	VCC
20	VCC	42	VCC
21	-12VDC (not used on B2020)	43	SDA H
22	+12VDC (not used on B2020)	44	SCL H

NOTE**VCC_3V pins not used on B2024.****Table 83: MCA44 J3**

Side 1		Side 2	
1	VCC_3V H	23	VCC_3V H
2	VCC_3V H	24	VCC_3V H
3	VCC_3V H	25	VCC_3V H
4	GND	26	GND
5	GND	27	GND
6	GND	28	GND
7	VCC_3V H	29	VCC_3V H
8	VCC_3V H	30	VCC_3V H
9	VCC_3V H	31	VCC_3V H
10	GND	32	GND
11	GND	33	GND
12	GND	34	GND
13	VCC_3V H	35	VCC_3V H
14	VCC_3V H	36	VCC_3V H
15	VCC_3V H	37	VCC_3V H
16	GND	38	GND
17	GND	39	GND

Table 83 (Cont.): MCA44 J3

Side 1		Side 2	
18	GND	40	GND
19	VCC_3V H	41	VCC_3V H
20	VCC_3V H	42	VCC_3V H
21	VCC_3V H	43	VCC_3V H
22	GND	44	GND

11.3 CPU Module Max DC Power Requirement

The power requirements shown in Table 84 assume a 24ns System bus cycle.

Table 84: CPU Power Requirements

Configuration	EV Clock Speed (ns)	3.3V \pm 5% DC Max (amps)	5V \pm 5% DC Max (amps)
B2020-AA	5.26	14.5	13.0
B2024-AA	3.64	0.0	25.0

11.4 Environmental Specifications - Class B modified

11.4.1 Temperature

11.4.1.1 Storage

Range -40 degrees Celsius to +66 degrees Celsius.

Before operating a module which is at a temperature beyond the operating range, that module must first be brought to an environment within the operating range, and then allowed to stabilize for a reasonable length of time. (Five or more minutes, depending on air circulation.)

11.4.1.2 Operating

Range 5 degrees Celsius to 35 degrees Celsius environment with 10 degrees Celsius rise from inlet to outlet air.

The enclosure is Class B rated with a 10 to 40 degrees Celsius environment and 10 degrees Celsius rise from inlet to outlet with a 20-80% noncondensing relative humidity.

De-rate the maximum operating temperature by 1.82 degrees Celsius for each 1000 meters of altitude above sea level. Reference: Standard Atmosphere, Standard Gravity.

11.4.2 Relative Humidity

11.4.2.1 Storage

10% to 95%

11.4.2.2 Operating

10% to 95% non-condensing at 304 meters of altitude.

11.4.3 Altitude

Reference: Standard Atmosphere, Standard Gravity.

11.4.3.1 Storage

The module is not mechanically or electrically damaged at altitudes up to 4.9 Km

11.4.3.2 Operating Altitude

The module can be operated up to 2.4 Km.

11.4.4 Airflow

Airflow must be at least 91.44 linear meters/minute.

11.4.5 Contamination

The module must be stored and used in a non-caustic environment.

11.4.6 Mean Time Between Failure (MTBF) Rate

The MTBF for the CPU module is TBD hours.

11.4.7 Electrical Characteristics

This module connects to the System bus backplane to an MCA 240-pin connector, two 299-pin PGA gate arrays, and one 44-pin PLCC arbitration gate array. System bus protocol signals are TTL levels, and backplane clocks are PECL levels.

11.4.7.1 AC References

System bus signals are driven relative to a TPhi1 or TPhi1_L rising edge. Receivers must latch these signals using a TPhi1 or TPhi1_L rising edge, respectively.

11.4.8 Clock Description

The module receives two differential clock signal pairs and AC couples them to two PECL to CMOS level converter chips (DC277BA DEC PN 21-36105-02). The four clock signals are routed on layer 4 etch from the connector pins and are 50mm using a 50 ohm etch impedance. Each of the four PECL clock signals connects to a 50 ohm termination (Thevenin - 82 ohms and 120 ohms - 5%) and two 680pF capacitors. etch from the connector pins and are 50mm using a 48 ohm etch impedance (12/20 lines).

Each of the four PECL clock signals connects to a 48 ohm termination (Thevenin - 82 ohms and 120 ohms - 5%) and two 680pF capacitors.

Figure 67: System backplane clocks

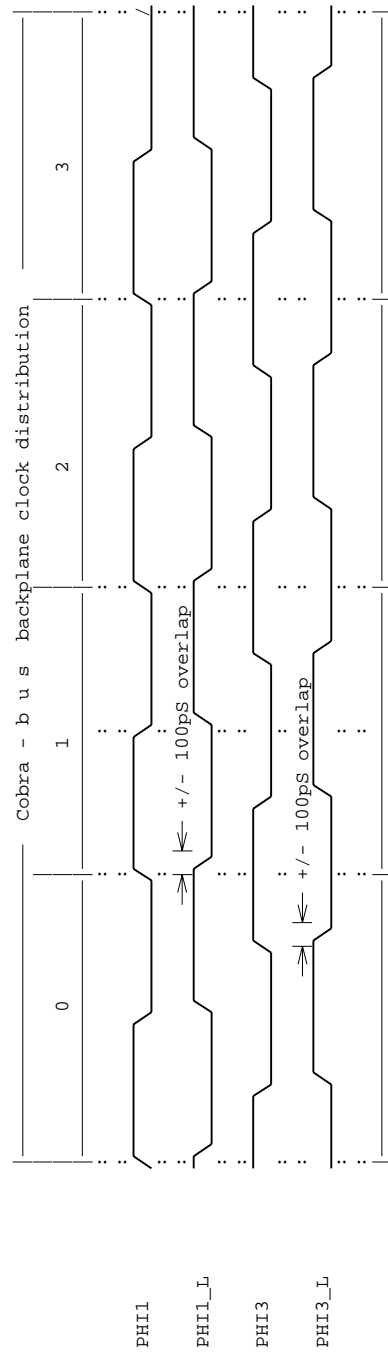
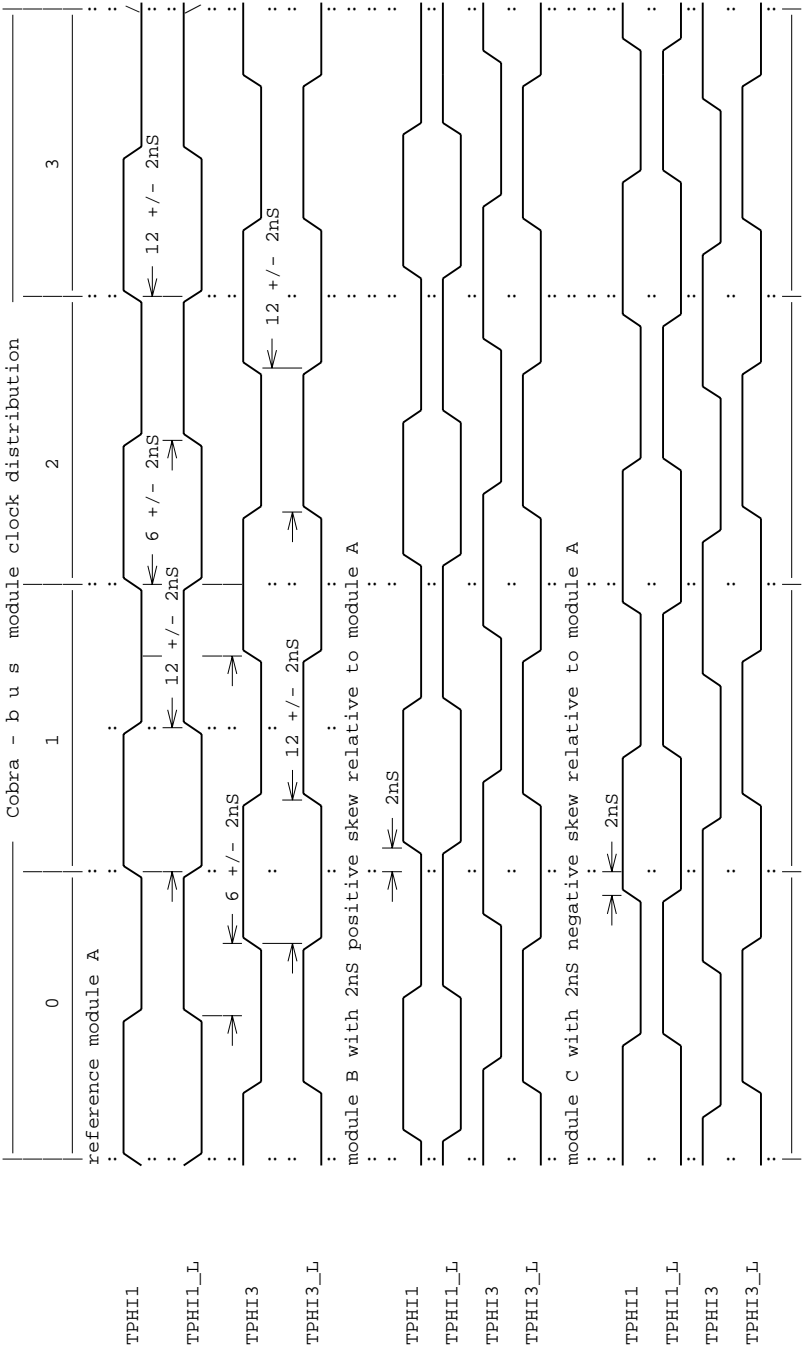


Figure 68: module clocks



11.4.9 AC and DC Characteristics

Table 85: System bus AC and DC Characteristics

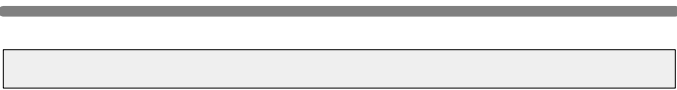

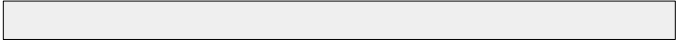

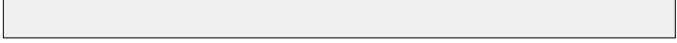



Parameter	Description	Min	Max	Units	Notes
DC Characteristics					
Voh	Output high voltage @Ioh = 8ma	2.4	-	V	
Vol	Output low level @Iol= -8ma	-	.4	V	
Vohck	Output high voltage @Ioh = -8ma	4.5	-	V	
Volck	Output low voltage @Iol = 8ma	-	.5	V	
Volod	Output low voltage @Iol = 48ma	-	.4	V	
Ioh	Output high current @Voh = min	8	-	ma	
Iol	Output low current @Vol = max	8	-	ma	
Iohck	Output high current @Voh= max	8	-	ma	
Iolck	Output low current @Vol= max	8	-	ma	
Iolod	Output low current @Vol= max	48	-	ma	
Ioz	Tri-state leakage current Vpin= 0 to 4.74v	-100	+100	ua	
Vih	Input high voltage	2	-	V	
Vil	Input low voltage	-	.8	V	
Vihck	Input high clock voltage	70%-Vdd	-	V	
Vilck	Input low clock voltage	-	30%-Vdd	V	

Table 85 (Cont.): System bus AC and DC Characteristics

Parameter	Description	Min	Max	Units	Notes
AC Characteristics					
Tcp	Clock Period	24	40	ns	
Tcw	Tphi Clock high or low width @Tcmin	12.0	20.0	ns	
Tmskh	module-to-module TPHI1 to TPHI1 skew	± 0	± 1.0	ns	
Tmskl	module-to-module TPHI1_L to TPHI1_L skew	± 0	± 1.0	ns	
Tmsk	module-to-module TPHI1 to TPHI1_L skew	± 0	± 1.0	ns	
Taskh	ASIC-to-ASIC (same module) TPHI1 to TPHI1 skew	± 0	± 0.5	ns	
Taskl	ASIC-to-ASIC (same module) TPHI1_L to TPHI1_L skew	± 0	± 0.5	ns	
Tcko	TPHI1 to System bus BD8TRP output high or low	3.0	6.0	ns	
Tsu	System bus input setup time high or low to TPHI	2.0	-	ns	
Thld	TPHI-to-System bus input hold high or low	1.0	-	ns	
Thz	TPHI-to-System bus output off time	3.5	6	ns	
Tpup	TPHI1_L rising to System bus OD output @Voh = 2.4V, rising not critical	4.0	21	ns	
Tpuc	TPHI1_L rising to System bus rising edge critical OD output @Voh = 2.4v	4.0	18	ns	
Cp	BD8TRP or BD4TRP Signal pin capacitance in 299 PGA	2	5	pF	
Cpod	48ma open drain Signal pin capacitance in 299 PGA	3	8	pF	

The module is an 8-layer module with a signal Z0 of 68 ± 7 ohms with a minimum width 5-mil line and minimum 9-mil spaces and a clock Z0 of $48 +6/-5$ ohms on layer four only, with a minimum width 11-mil line and minimum 20-mil spaces. Via diameter is typically 15 mils, but 10 mil vias are used also. Figure 69 shows the module layer construction.

Figure 69: Module Layup (reference)

L1		Pads (1/2 oz (.0014 min finished)) . 005 + .003/-0.001
L2		68 (6.0 MIL FLW) ohm signal (1 oz) .008 + .003/-0.002
L3		Gnd (1 oz) .009 + .003/-0.002
L4		68 (50 mil FLW) 48 (11 mil FLW) ohm signal/clock .010 + .003/-0.003
L5		68 (50 mil FLW) ohm signal/clock (1 oz) .009 + .003/-0.002
L6		3.3V and 5.0V (1 oz) .008 + .003/-0.002
L7		68 (6.0 MIL FLW) Ohm signal (1 oz) .005 + .003/-0.001
L8		Pads (1/2 oz (.0014 min finished))

[yuryan.cpu]layup_decw.doc

APPENDIX A

ALPHA ARCHITECTURE OPTIONS SUPPORTED

- Cycle counter - RCC
- 8 KB virtual page size
- 4 GB physical memory address space reserved of which the Sable CPU will support a maximum of 2GB of physical memory. (Cacheable)
- 10 GB physical I/O address space reserved (non-Cacheable)
- 976.5625 microseconds interval-timer frequency

APPENDIX B

SABLE CPU MODULE REGISTER REFERENCE GUIDE

Figure 70: B-Cache Control Register (BCC-CSR0, offset = 0000)

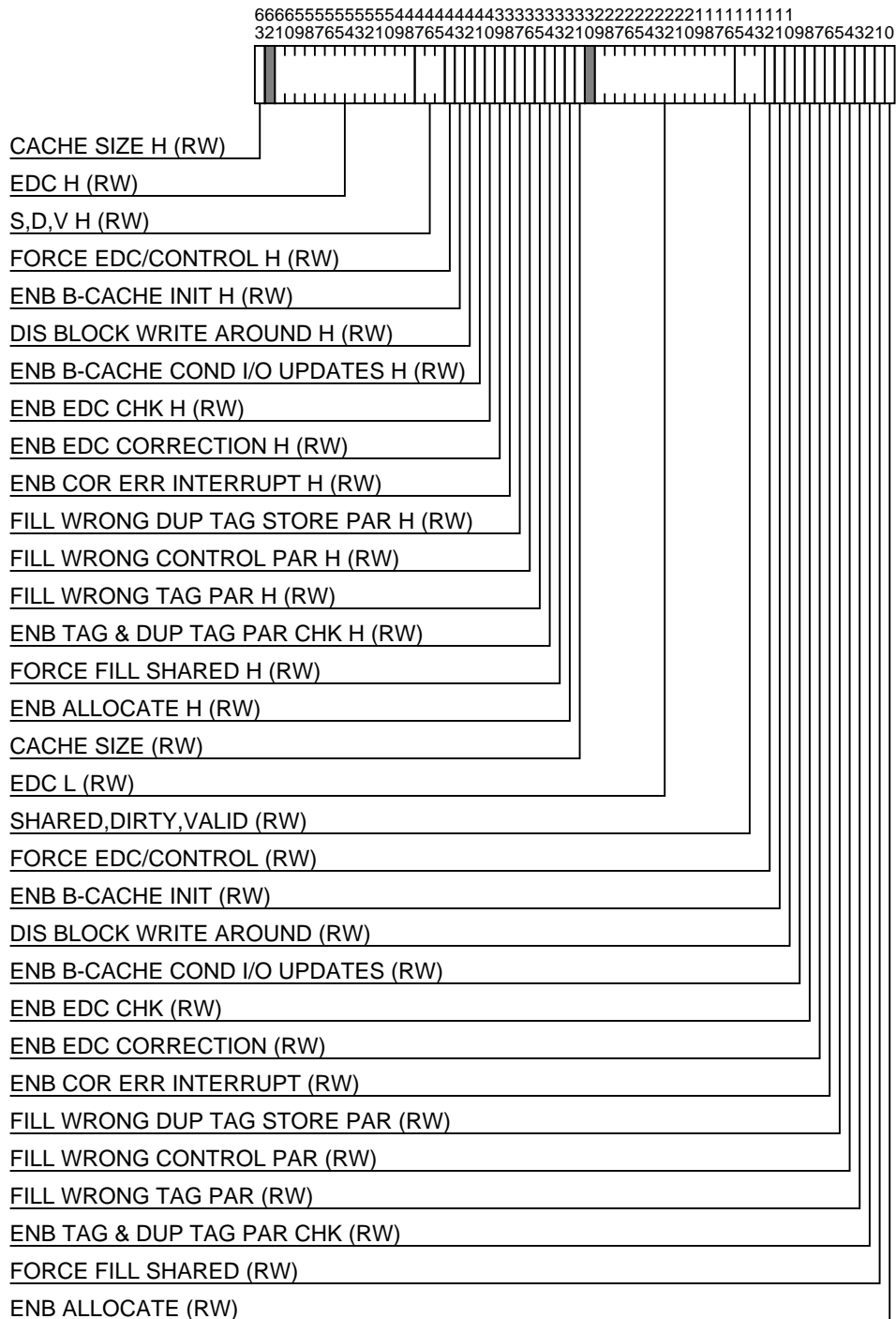


Figure 71: B-Cache Correctable Error (BCCE-CSR1, offset = 0020)

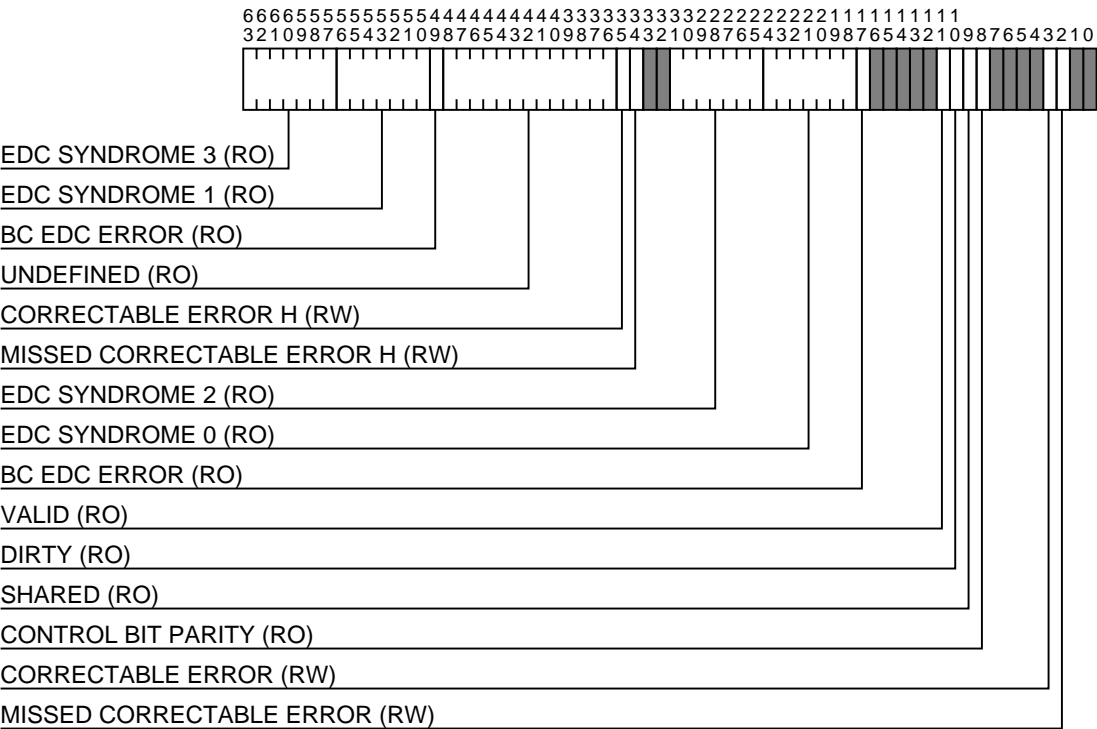


Figure 72: B-Cache Correctable Error Address (BCCEA-CSR2, offset = 0040)

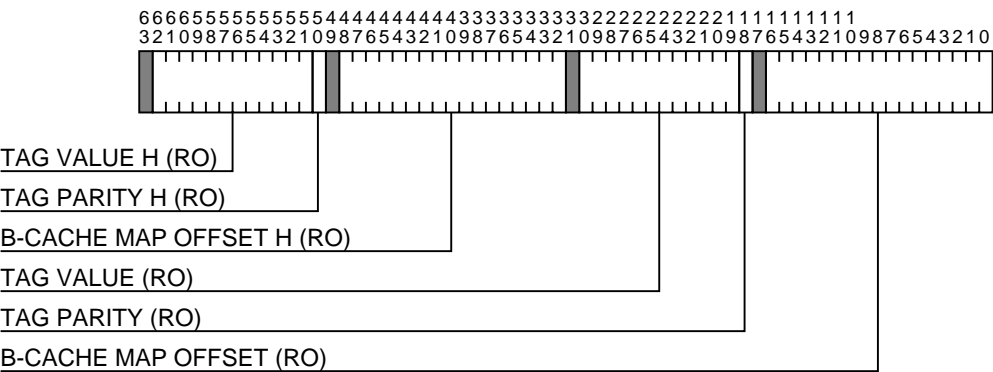


Figure 73: B-Cache Uncorrectable Error (BCUE-CSR3, offset = 0060)

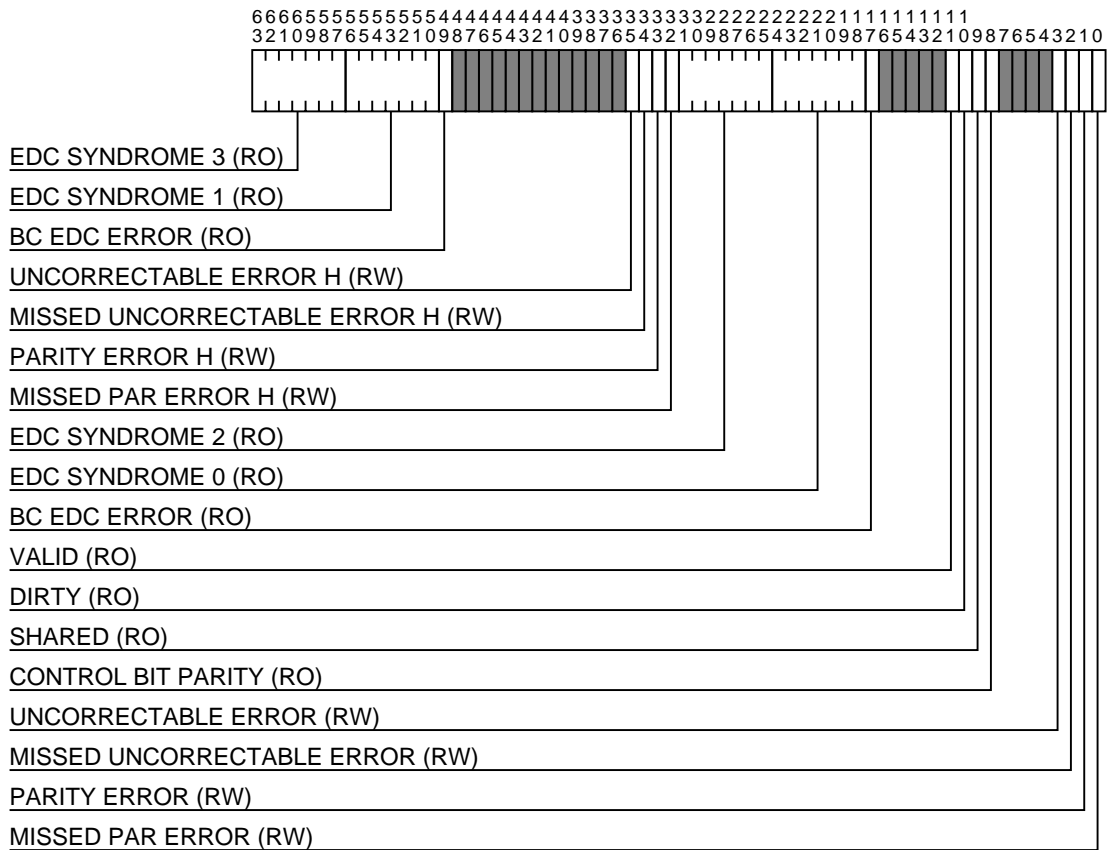


Figure 74: B-Cache Uncorrectable Error Address (BCUEA-CSR4, offset = 0080)

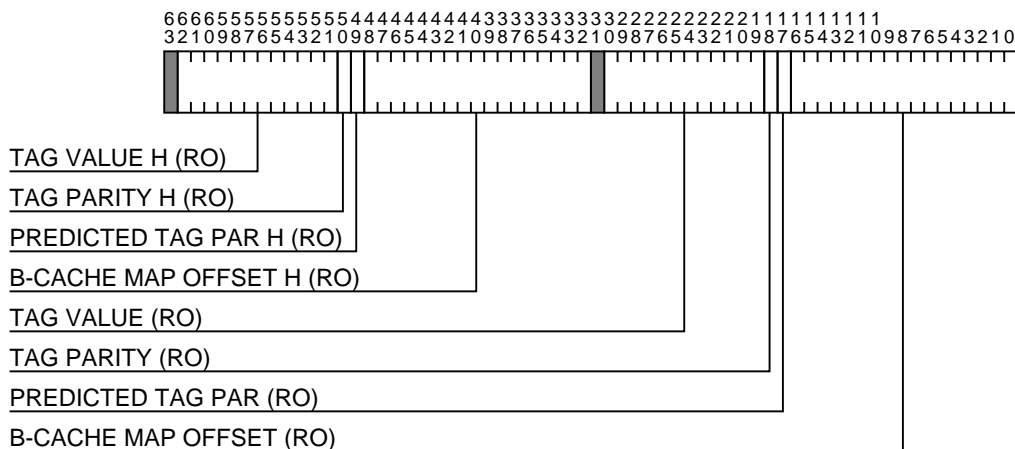


Figure 75: Duplicate Tag Error Register (DTER-CSR5, offset = 00A0)

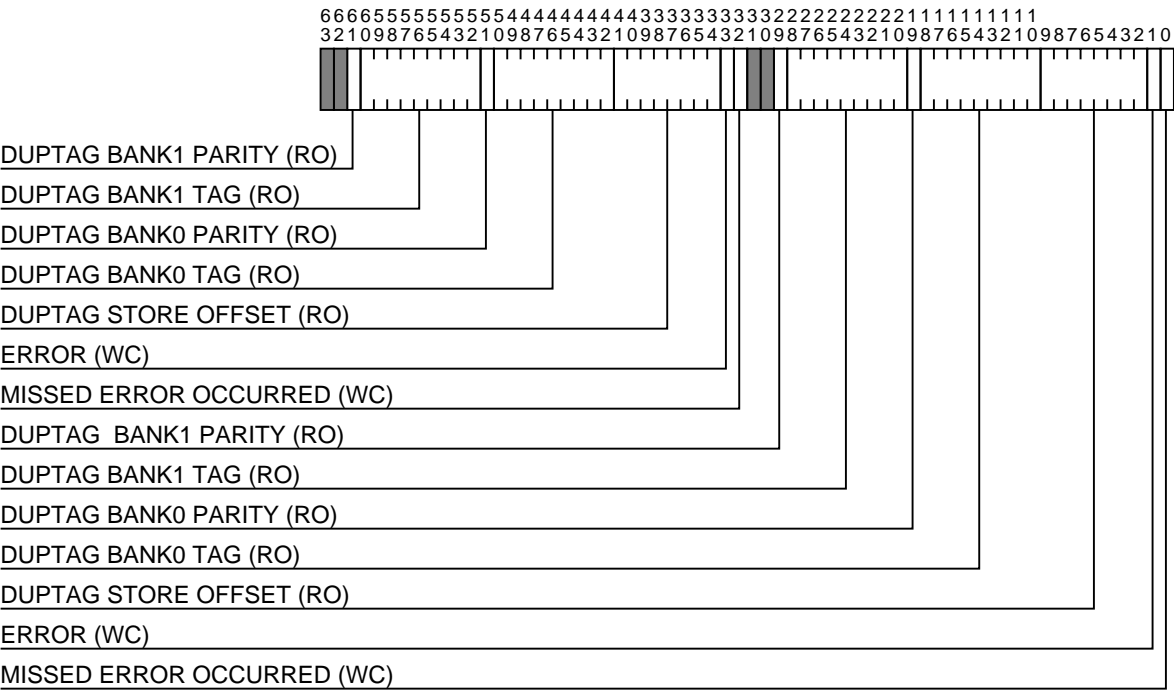


Figure 76: Cobra-bus Control Register (CBCTL-CSR6, offset = 00C0)

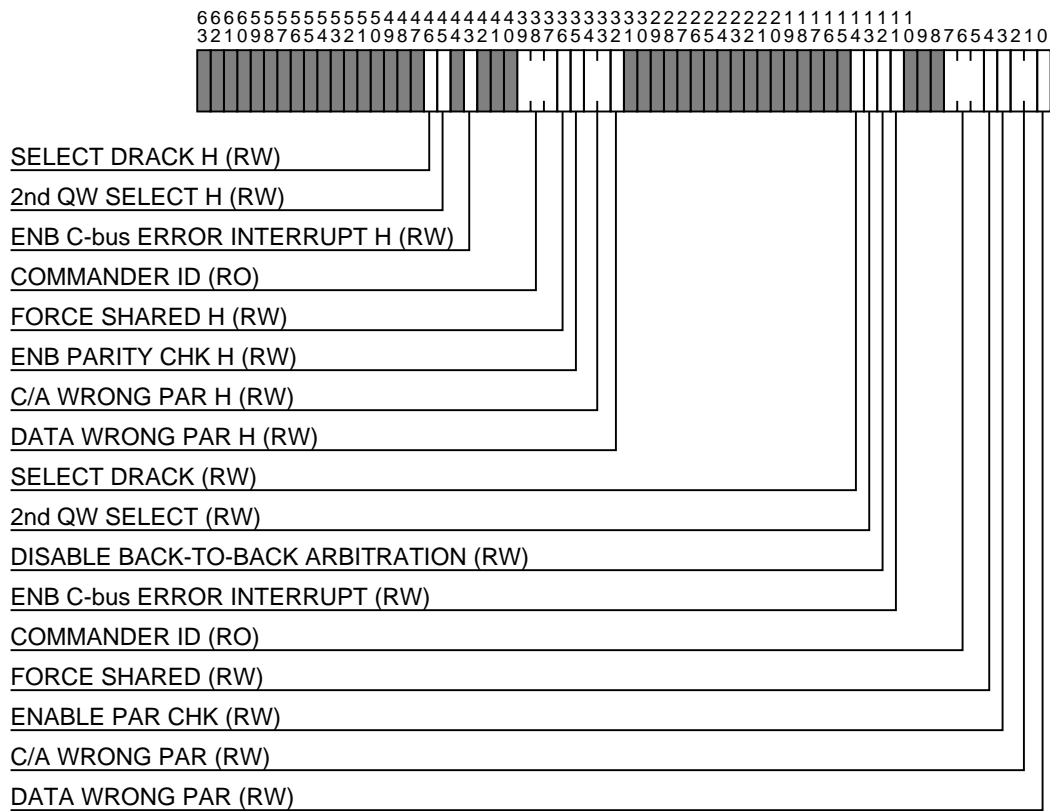


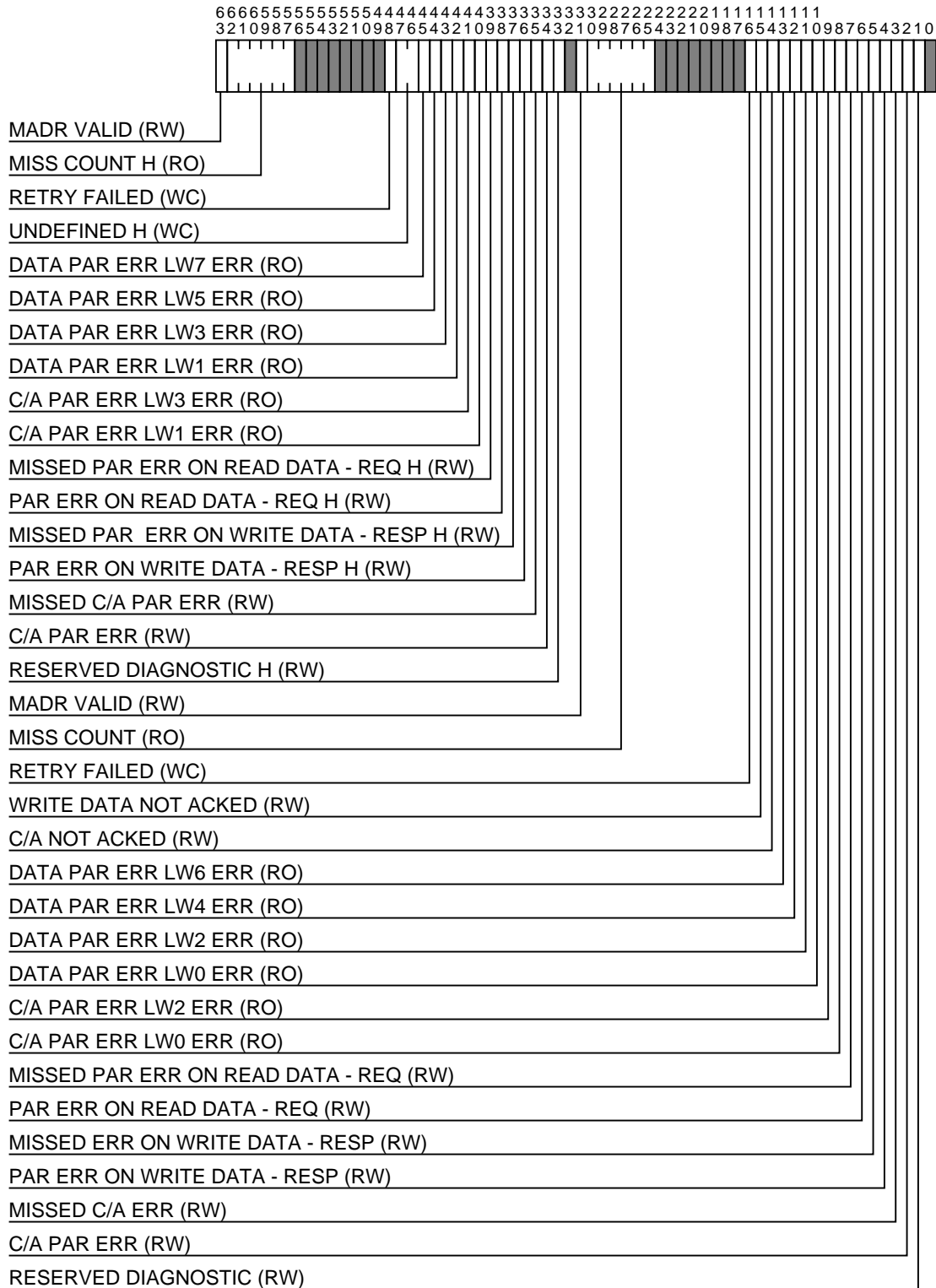
Figure 77: Cobra-bus Error Register (CBE-CSR7, offset = 00E0)

Figure 78: Cobra-bus Error Address Low (CBEAL-CSR8, offset = 0100)

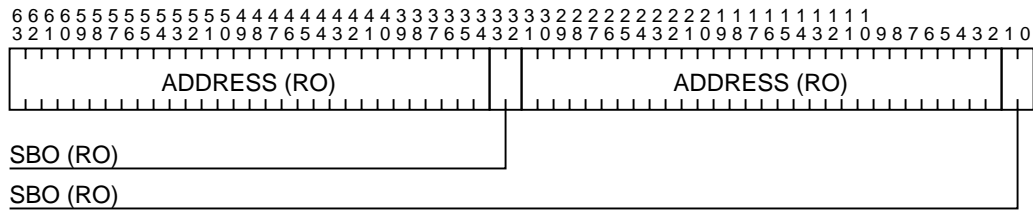


Figure 79: Cobra-bus Error Address High (CBEAH-CSR9, offset = 0120)

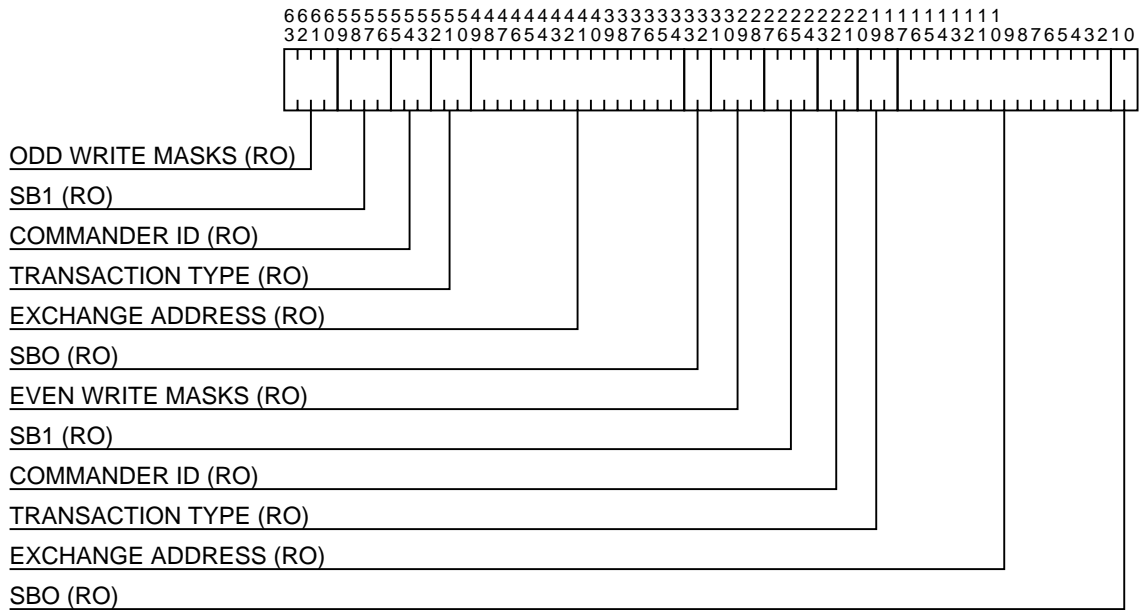


Figure 80: Processor Mailbox (PMBX-CSR10, offset = 0140)

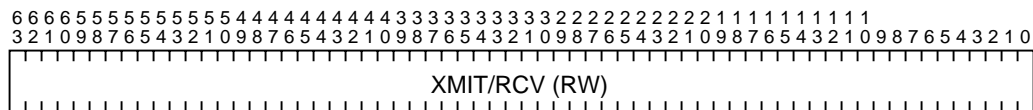


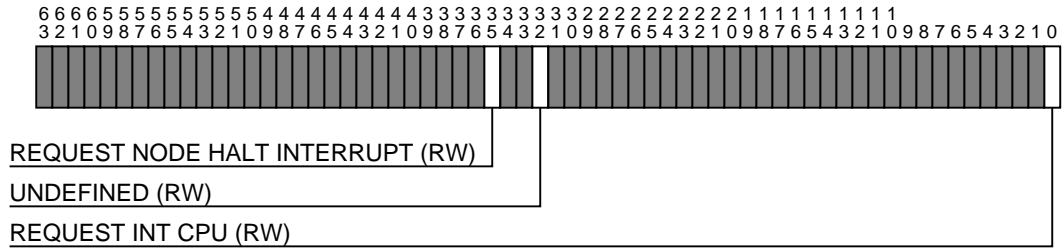
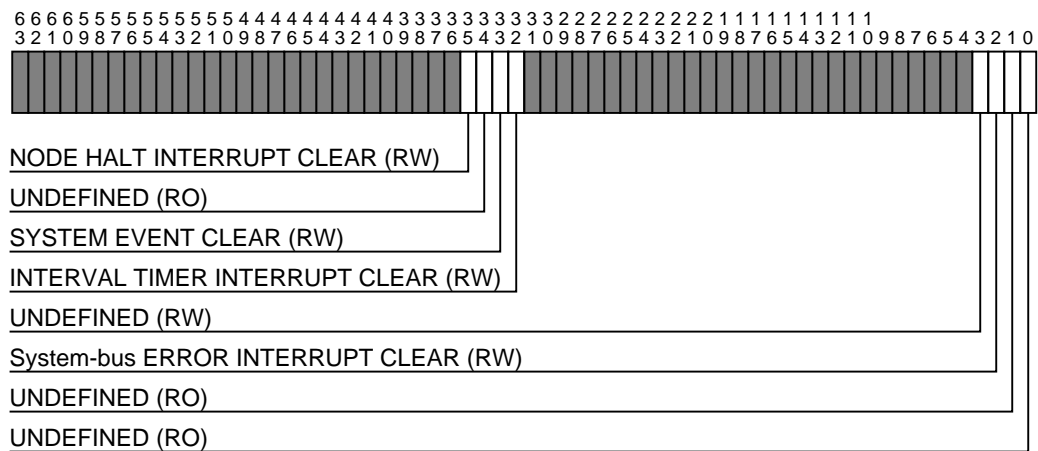
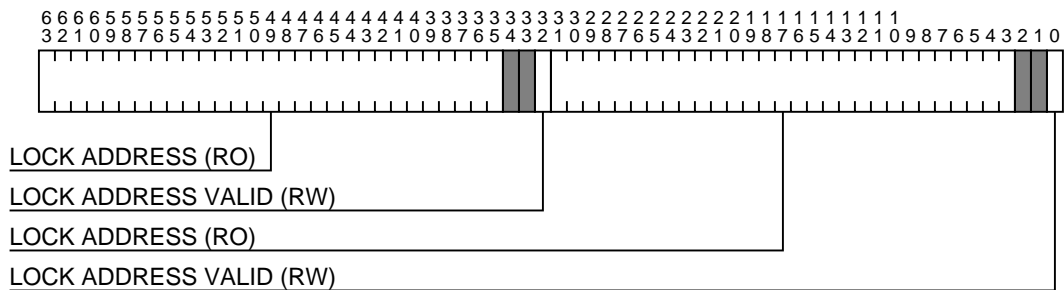
Figure 81: Interprocessor Interrupt Request (IPIR-CSR11, offset = 0160)**Figure 82: System Interrupt Clear Register (SIC-CSR12, offset = 0180)****Figure 83: Address Lock Register (ADLK-CSR13, offset = 01A0)**

Figure 84: Miss Address Register Low (MADRL-CSR14, offset = 01C0)

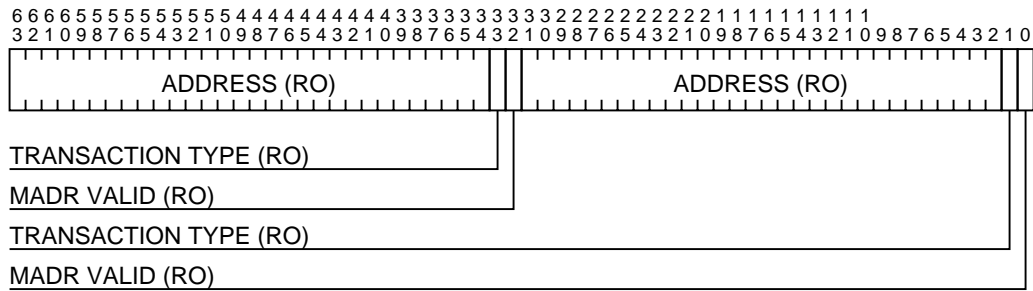
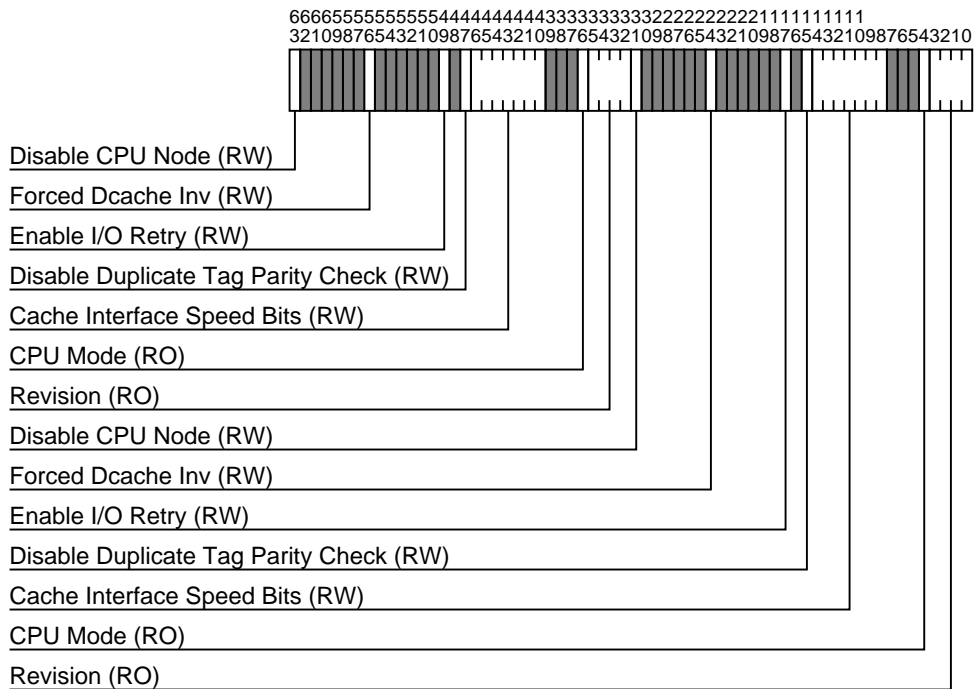


Figure 85: C4 Revision Register (CRR-CSR15, offset = 01E0)



APPENDIX C

COBRA SPECIFIC (PRIVILEGED ARCHITECTURE LIBRARY CODE) PALCODE

C.1 Introduction

In a family of machines both users and operating system implementers require functions to be implemented consistently. When functions are implemented to a common interface, the code that uses those functions can be used on several different implementations without modification.

These functions range from the binary encoding of the instructions and data, to the exception mechanisms and synchronization primitives. Some of these functions can be cost effectively implemented in hardware, but several are impractical to implement directly in hardware. These functions include low-level hardware support functions such as translation buffer fill routines, interrupt acknowledge, and exception dispatch. Also included is support for privileged and atomic operations that require long instruction sequences such as Return from Exception or Interrupt (REI).

In the VAX architecture, these functions are generally provided by microcode. In the 21064, there is no microcode. However an architected interface to these functions that will be consistent with other members of ALPHA family of machines is still required. The Privileged Architecture Library Code (PALcode) is used to implement these functions without resorting to a microcoded machine. The 21064 hardware development group will provide and maintain a version of the PALcode for the 21064. Module development groups will have to provide and maintain module specific modifications to the PALcode.

C.2 PAL Environment

PALcode runs in an environment with privileges enabled, instruction stream mapping disabled, and interrupts disabled. The enabling of privileges allows all functions of the machine to be controlled. Disabling of instruction stream mapping allows PALcode to be used to support the memory management functions (e.g., translation buffer miss routines can not be run via mapped memory). PALcode can perform both virtual and physical data stream references. The disabling of interrupts allows the system to provide multi-instruction sequences as atomic operations. The PALcode environment in the 21064 also includes 32 PAL temp registers which are accessible only by PAL reserved move to/from processor register instructions.

C.3 Special PAL Instructions

PALcode uses the ALPHA instruction set for most of its operations. The 21064 maps the architecturally reserved PALcode opcodes (PALRES0 - PALRES4) to a special load and store (HW_LD, HW_ST), a move to and move from processor register (HW_MTPR, HW_MFPR), and a return from **PALmode** exception (HW_REI). These instructions produce a Reserved Opcode fault if executed while not in the PALcode environment unless the HWE bit of the ICCSR IPR is set, in which case these instructions can be executed in kernel mode.

Register checking and bypassing logic is provided for PALcode instructions as it is for non-PALcode instructions when using general purpose registers. Explicit software timing is required for accessing the hardware specific IPRs and the PAL_TEMP. These constraints are described in the **PALmode** restriction and IPR sections.

C.3.1 HW_MFPR and HW_MTPR

The internal processor register specified by the PAL, ABX, IBX, and index field is written/read with the data from the specified integer register. Processor registers may have side effects that happen as the result of writing/reading them. Coding restrictions are associated with accessing various registers. Separate bits are used to access Abox IPRs, Ibox IPRs, and PAL_TEMP, therefore it is possible for an MTPR instructions to write multiple registers in parallel if they both have the same index.

The HW_MFPR and HW_MTPR instructions have the following format:

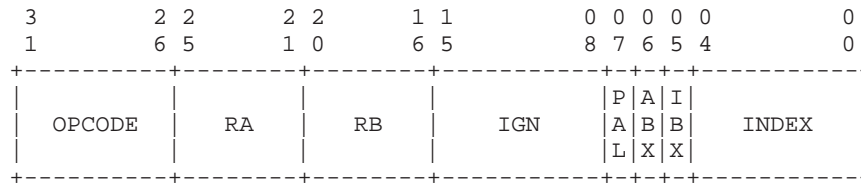


Table 86: HW_MFPR and HW_MTPR Format Description

Field	Description
OPCODE	Is either 25 (HW_MFPR) or 29 (HW_MTPR).
RA/RB	Contain the source,HW_MTPR or destination,HW_MFPR, register number. The RA and RB fields must always be identical.
PAL	If set this HW_MFPR or HW_MTPR instruction is referencing a PAL temporary register, PAL_TEMP.
ABX	If set this HW_MFPR or HW_MTPR instruction is referencing a register in the Abox.
IBX	If set this HW_MFPR or HW_MTPR instruction is referencing a register in the Ibox.
INDEX	Specifies hardware specific register as shown in Table 87

The following table indicates how the PAL, ABX, IBX, and INDEX fields are set to access the internal processor registers. Setting the PAL, ABX, and IBX fields to zero generates a NOP.

Table 87: IPR Access

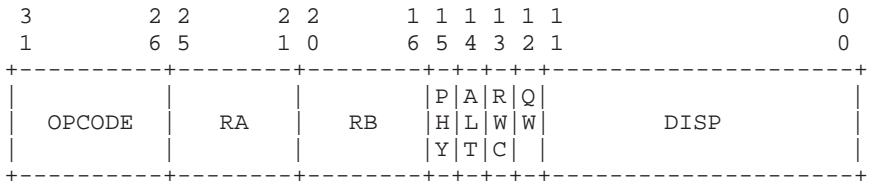
Mnemonic	PAL	ABX	IBX	INDEX	Access	Comments
TB_TAG	x	x	1	0	W	PAL mode only
ITB_PTE	x	x	1	1	R/W	PAL mode only
ICCSR	x	x	1	2	R/W	
ITB_PTE_TEMP	x	x	1	3	R	PAL mode only

Table 87 (Cont.): IPR Access

Mnemonic	PAL	ABX	IBX	INDEX	Access	Comments
EXC_ADDR	x	x	1	4	R/W	
SL_RCV	x	x	1	5	R	
ITBZAP	x	x	1	6	W	PAL mode only
ITBASM	x	x	1	7	W	PAL mode only
ITBIS	x	x	1	8	W	PAL mode only
PS	x	x	1	9	R/W	
EXC_SUM	x	x	1	10	R/W	
PAL_BASE	x	x	1	11	R/W	
HIRR	x	x	1	12	R	
SIRR	x	x	1	13	R/W	
ASTRR	x	x	1	14	R/W	
HIER	x	x	1	16	R/W	
SIER	x	x	1	17	R/W	
ASTER	x	x	1	18	R/W	
SL_CLR	x	x	1	19	W	
SL_XMIT	x	x	1	22	W	
DTB_CTL	x	1	x	0	W	
DTB_PTE	x	1	x	2	R/W	
DTB_PTE_TEMP	x	1	x	3	R	
MMCSR	x	1	x	4	R	
VA	x	1	x	5	R	
DTBZAP	x	1	x	6	W	
DTASM	x	1	x	7	W	
DTBIS	x	1	x	8	W	
BIU_ADDR	x	1	x	9	R	
BIU_STAT	x	1	x	10	R	
DC_ADDR	x	1	x	11	R	
DC_STAT	x	1	x	12	R	
FILL_ADDR	x	1	x	13	R	
ABOX_CTL	x	1	x	14	W	
ALT_MODE	x	1	x	15	W	
CC	x	1	x	16	W	
CC_CTL	x	1	x	17	W	
BIU_CTL	x	1	x	18	W	
FILL_SYNDROME	x	1	x	19	R	
BC_TAG	x	1	x	20	R	
FLUSH_IC	x	1	x	21	W	
FLUSH_IC_ASM	x	1	x	23	W	21064 Only
PAL_TEMP[31..0]	1	x	x	31-00	R/W	

C.3.2 HW_LD and HW_ST

PALcode uses the HW_LD and HW_ST instructions to access memory outside of the realm of normal ALPHA memory management. The HW_LD and HW_ST instructions have the following format:



The effective address of these instructions is calculated as follows:

```
addr <- (SEXT(DISP) + RB) AND NOT (QW | 11(bin))
```

Table 88: HW_LD and HW_ST Format Description	
Field	Description
OPCODE	Is either 27 (HW_LD) or 31 (HW_ST).
RA/RB	Contain register numbers, interpreted in the normal fashion for loads and stores.
PHY	If clear the effective address of the HW_LD or HW_ST is a virtual address. If set then the effective address of the HW_LD or HW_ST is a physical address.
ALT	For virtual-mode HW_LD and HW_ST instructions this bit selects the processor mode bits which are used for memory management checks. If ALT is clear the current mode bits of the PS register are used, while if ALT is set the mode bits in the ALT_MODE IPR are used. In the 21064 physical-mode load-lock and store-conditional variants of the HW_LD and HW_ST instructions may be created by setting both the PHY and ALT bits.
RWC	The RWC (read with write check) bit, if set, enables both read and write access checks on virtual HW_LD instructions.
QW	The quadword bit specifies the data length. If it is set then the length is quadword. If it is clear then the length is longword.
DISP	The DISP field holds a 12-bit signed byte displacement.

Note that PALcode at PAL entry points of higher priority than DTBMISS must unlock possible MMCSR IPR and VA IPR locks.

Table 90: PAL Entry Points

Entry Name	Time	Offset(Hex)	Cause
RESET	anytime	0000	
MCHK	pipe_stage[7]	0020	Uncorrected hardware error.
ARITH	anytime	0060	Arithmetic exception.
INTERRUPT	anytime	00E0	Includes corrected hardware error.
D-stream errors	pipe_stage[6]	01E0, 08E0, 09E0, 11E0	See Table 91.
ITB_MISS	pipe_stage[5]	03E0	ITB miss.
ITB_ACV	pipe_stage[5]	07E0	I-stream access violation.
CALLPAL	pipe_stage[5]	2000,40,80 thru 3FC0	128 locations based on instruction[7,5:0]. If bit[7] equals zero and CM does not equal kernel mode then an OPDEC exception occurs.
OPCDEC	pipe_stage[5]	13E0	Reserved or privileged opcode.
FEN	pipe_stage[5]	17E0	FP op attempted with : FP instructions disabled via ICCSR FPE bit FP IEEE round to +/- infinity FP IEEE with datatype field other than S,T,QW

PALcode functions are implemented via the CALL_PAL instruction. CALL_PAL instructions cause exceptions in the hardware. As with all exceptions, the EXC_ADDR register is loaded by hardware with a possible return address. CALL_PAL exceptions do not load the EXC_ADDR register with the address of the CALL_PAL instruction. Rather, they load the EXC_ADDR register with the address of the instruction following the CALL_PAL. For this reason, the 21064 PALcode supporting the desired PAL mode functions need not increment the EXC_ADDR register before executing a HW_REI instruction to return to native mode. This feature requires special handling in the arithmetic trap and machine check PALcode flows for the 21064.

To improve the speed of execution, a limited number of CALL_PAL instructions are directly supported in hardware with dispatches to specific address offsets. The 21064 provides the first 64 privileged and 64 unprivileged CALL_PAL instructions with larger code regions of 64 bytes. This produces hardware PAL entry points as described below.

```
Privileged CALL_PAL Instructions [00000000:0000003F]
Offset(HEX) = 2000 + ([5:0] shift left 6)
Unprivileged CALL_PAL Instructions [00000080:000000BF]
Offset(HEX) = 3000 + ([5:0] shift left 6)
```

The 21064 CALL_PAL instructions that do not fall within the ranges [00000000:0000003F] and [00000000:000000BF] result in an OPCDEC exception. In addition, CALL_PAL instructions that fall within the range [00000000:0000003F] while the 21064 is not executing in kernel mode will result in an OPCDEC exception.

The PAL entry points assigned to D-stream errors require a bit more explanation. The hardware recognizes four classes of D-stream memory management errors: bad virtual address (improper sign extension), DTB miss, alignment error and everything else (ACV, FOR, FOW). These errors get mapped into four PAL entry points: UNALIGN, DTB_MISS PAL mode, DTB_MISS Native mode and D_FAULT. Table 90 lists the priority of these entry points as a group with respect to each of the other entry points. Since a particular D-stream memory reference may generate errors which fall into more than one of the four error classes which the hardware recognizes, we also must define the priority of each of the D-stream PAL entry points with respect to the others in the D-stream PAL entry group. Table 91 gives this priority. The PAL entry point 8E0 for Native mode DTB_MISS is only available in the 21064. EV3 provides only one DTB_MISS PAL entry point at address offset 9E0.

Table 91: D-stream Error PAL Entry Points

BAD_VA	DTB_MISS	UNALIGN	PAL	Other	Offset(Hex)
1	x	0	x	x	01E0 D_FAULT
1	x	1	x	x	11E0 UNALIGN
0	1	x	0	x	08E0 DTB_MISS Native
0	1	x	1	x	09E0 DTB_MISS PAL
0	0	1	x	x	11E0 UNALIGN
0	0	0	x	1	01E0 D_FAULT

C.5 General PALmode Restrictions

Many of the restrictions involve waiting 'n' cycles before using the results of PAL instructions. Inserting 'n' instructions between the two time-sensitive instructions is the typical method of waiting for 'n' cycles. Because the 21064 can dual issue instructions it is possible to write code that requires $2*n+1$ instructions to wait 'n' cycles. Due to the resource requirements of individual instructions, and the 21064 hardware design, multiple copies of the same instruction can not be dual issued. This fact is used in some of the code examples below.

C.5.1 21064 PAL Restrictions

1. As a general rule, HW_MTPR instructions require at least 4 cycles to update the selected IPR. Therefore, at least three cycles of delay must be inserted before using the result of the register update.

Note that only the write followed by read operation requires this software timing. Multiple reads, multiple writes, or read followed by write will pipeline properly and do not require software timing except for accesses of the TB registers.

These cycles can be guaranteed by either including 7 instructions which do not use the IPR in transition or proving through the dual issue rules and/or state of the machine, that at least 3 cycles of delay will occur. As a special case, multiple copies of a HW_MTPR instruction, used as a NOP instruction, can be used to pad cycles after the original HW_MTPR. Since multiple copies of the same instruction will never dual issue, the maximum number of instructions necessary to insure at least 3 cycles of delay is 3.

An example of this is :

```

HW_MTPR Rx, HIER      ; Write to HIER
HW_MFPR R31, 0        ; NOP mxpr instruction
HW_MFPR R31, 0        ; NOP mxpr instruction
HW_MFPR R31, 0        ; NOP mxpr instruction
HW_MFPR Ry, HIER      ; Read from HIER

```

The HW_REI instruction uses the ITB if the EXC_ADDR register contains a non PAL mode VPC, $VPC<0> = 0$. By the rule above, this implies that at least 3 cycles of delay must be included after writing the ITB before executing a HW_REI instruction to exit PAL mode.

Exceptions:

- HW_MFPR instructions reading any PAL_TEMP register can never occur exactly two cycles after a HW_MTPR instruction writing any PAL_TEMP register. The simple solution results in code of the form:

```
HW_MTPR Rx, PAL_R0      ; Write PAL temp 0
HW_MFPR R31, 0          ; NOP mxpr instruction
HW_MFPR R31, 0          ; NOP mxpr instruction
HW_MFPR Ry, PAL_R0      ; Read PAL temp 0
```

The above code guarantees 3 cycles of delay after the write before the read. It is also possible to make use of the cycle immediately following a HW_MTPR to execute a HW_MFPR instruction to the same (accomplishing a swap) or a different PAL_TEMP register. The swap operation only occurs if the HW_MFPR instruction immediately follows the HW_MTPR. This timing requires great care and knowledge of the pipeline to insure that the second instruction does not stall for one or more cycles. Use of the slot to accomplish a read from a different PAL_TEMP register requires that the second instruction not stall for exactly one cycle. This is much easier to insure. A HW_MFPR instruction may stall for a single cycle as a result of a write after write conflict.

- The EXC_ADDR register may be read by a HW_REI instruction only 2 cycles after the HW_MTPR. This is equivalent to one intervening cycle of delay. This translates to code of the form:

```
HW_MTPR Rx, EXC_ADDR    ; Write EXC_ADDR
HW_MFPR R31, 0          ; NOP cannot dual issue with either
HW_REI                  ; Return
```

2. An MTPR operation to the DTBIS register cannot be bypassed into. In other words, all data being moved to the DTBIS register must be sourced directly from the register file. One way to insure this is to provide at least 3 cycles of delay before using the result of any integer operation (except MUL) as the source of an MTPR DTBIS. Do not use a MUL as the source of DTBIS data. Sample code for this operation is :

```
ADDQ R1,R2,R3           ; source for DTBIS address
ADDQ R31,R31,R31        ; cannot dual issue with above, 1st cycle of delay
ADDQ R31,R31,R31        ; 2nd cycle of delay
ADDQ R31,R31,R31        ; 3rd cycle of delay
ADDQ R31,R31,R31        ; may dual issue with below, else 4th cycle of delay
HW_MTPR R3,DTBIS        ; R3 must be in register file, no bypass possible
```

3. When loading the CC register, bits <3:0> must be loaded with zero. Loading non-zero values in these bits may cause the count to be inaccurate.
4. An MTPR DTBIS cannot be combined with an MTPR ITBIS instruction. The hardware will not clear the ITB if both the Ibox and Abox IPRs are simultaneously selected. Instead, two instructions are needed to clear each TB individually. Code example:

```
HW_MTPR Rx,ITBIS
HW_MTPR Ry,DTBIS
```

5. An MXPR ITB_TAG, ITB_PTE, ITB_PTE_TEMP cannot follow a HW_REI that remains in PAL mode. (Address bit<0> of the EXC_ADDR is set) This rule implies that it is not a good idea to ever allow exceptions while updating the ITB. If an exception interrupts flow of the ITB miss routine and attempts to REI back, and the return address begins with a HW_MxPR instruction to an ITB register, and the REI is predicted correctly to avoid any delay between the two instructions, then the ITB register will not be written. Code example:

```
HW_REI                ; return from interrupt
HW_MTPR R1,ITB_TAG    ; attempts to execute very next cycle, instr ignored
```

6. The ITB_TAG, ITB_PTE and ITB_PTE_TEMP registers can only be accessed in PAL mode. If the instructions HW_MTPR or HW_MFPR to/from the above registers are attempted while not in PAL mode by setting the HWE (hardware enable) bit of the ICCSR, the instructions will be ignored.
7. Machine check exceptions taken while in PAL mode may load the EXC_ADDR register with a restart address one instruction earlier than the proper restart address. Some HW_MxPR instructions may have already completed execution even though the restart address indicates the HW_MxPR as the return instruction. Re-execution of some HW_MxPR instructions can alter machine state. (e.g. TB pointers, EXC_ADDR register mask)

The mechanism used to stop instruction flow during machine check exceptions causes the machine check exception to appear as a D-stream fault on the following instruction in the hardware pipeline. In the event that the following instruction is a HW_MxPR, a D-stream fault will not abort execution in all cases. Although the EXC_ADDR will be loaded with the address of the HW_MxPR instruction as if it were aborted, a HW_REI to this restart address will incorrectly re-execute this instruction.

Machine check service routines should check for MXPR instructions at the return address before continuing.

8. When writing the PAL_BASE register, exceptions may not occur. An exception occurring simultaneously with a write to the PAL BASE may leave the register in a metastable state. All asynchronous exceptions but reset can be avoided under the following conditions:

```
PAL mode ..... blocks all interrupts
machine checks disabled ..... blocks I/O error exceptions
                               (via ABOX_CTL reg or MB isolation)
Not under trap shadow ..... avoids arithmetic traps
```

The trap shadow is defined as :

```
less than 3 cycles after a non-mul integer operate that may overflow
less than 22 cycles after a MULL/V instruction
less than 24 cycles after a MULQ/V instruction
less than 6 cycles after a non-div fp operation that may cause a trap
less than 34 cycles after a DIVF or DIVS that may cause a trap
less than 63 cycles after a DIVG or DIVT that may cause a trap
```

9. The sequence HW_MTPR PTE, MTPR TAG is NOT allowed. At least two null cycles must occur between HW_MTPR PTE and HW_MTPR TAG.

10. The AMCHK exception service routine must check the EXC_SUM register for simultaneous arithmetic errors. Arithmetic traps will not trigger exceptions a second time after returning from exception service for the machine check.
11. Three cycles of delay must be inserted between HW_MFPR DTB_PTE and HW_MFPR DTB_PTE_TEMP. Code example:

```
HW_MFPR Rx,DTB_PTE      ; reads DTB_PTE into DTB_PTE_TEMP register
HW_MFPR R31,0           ; 1st cycle of delay
HW_MFPR R31,0           ; 2nd cycle of delay
HW_MFPR Ry,DTB_PTE_TEMP ; read DTB_PTE_TEMP into register file Ry
```
12. Three cycles of delay must be inserted between HW_MFPR IPTE and HW_MFPR ITB_PTE_TEMP. Code example:

```
HW_MFPR Rx,DTB_PTE      ; reads DTB_PTE into DTB_PTE_TEMP register
HW_MFPR R31,0           ; 1st cycle of delay
HW_MFPR R31,0           ; 2nd cycle of delay
HW_MFPR Ry,DTB_PTE_TEMP ; read DTB_PTE_TEMP into register file Ry
```
13. The content of the destination register for HW_MFPR Rx,DTB_PTE or HW_MFPR Rx,ITB_PTE is UNPREDICTABLE.
14. Two HW_MFPR DTB_PTE instructions cannot be issued in consecutive cycles. This implies that more than one instruction may be necessary between the HW_MFPR instructions if dual issue is possible. Similar restrictions apply to the ITB_PTE register.
15. Reading the EXC_SUM and BC_TAG registers require special timing. Refer to Section 3.1.17 and Section 3.9.11 for specific information.
16. DMM errors occurring one cycle before HW_MxPR instructions to the IPTE will NOT stop the TB pointer from incrementing to the next TB entry even though the mxpr instruction will be aborted by the DMM error. This restriction only affects performance and not functionality.
17. PALcode that writes multiple ITB entries must write the entry that maps the address contained in the EXC_ADDR register last.

C.5.2 21064 Specific PALmode Restrictions

1. HW_STC instructions cannot be followed, for two cycles, by any load instruction that may miss in the Dcache.
2. Updates to the ASN field of the ICCSR IPR require at least 10 cycles of delay before entering native mode that may reference the ASN during Icache access. If the ASN field is updated in Kernel mode via the HWE bit of the ICCSR IPR, it is sufficient that all I-stream references during this time be made to pages with the ASM bit set to avoid use of the ASN.
3. At least one cycle of delay must occur after a HW_MTPR TB_CTL before either a HW_MTPR ITB_PTE or a HW_MFPR ITB_PTE to allow set up of the ITB large page/small page decode.

4. The first cycle (the first one or two instructions) at all PALcode entry points may not execute a conditional branch or any other instruction that uses the jsr stack hardware. This includes instructions JSR, JMP, RET, COROUTINE, BSR, HW_REI and all Bxx opcodes except BR, which is allowed.
5. The following table indicates the number of cycles required after a HW_MTPR instruction before a subsequent HW_REI instruction for the specified IPRs. These cycles can be insured by inserting one HW_MFPR R31,0 instruction or other appropriate instruction(s) for each cycle of delay required after the HW_MTPR.

IPR	Cycles between HW_MTPR and HW_REI
xTBIS,ASM,ZAP	0
xIER	2
xIRR	2
ICCSR[FPE]	3
ICCSR[ASN]	5
FLUSH_IC[ASM]	6

C.6 Powerup

The table below lists the state of all the IPRs immediately following reset. The table also specifies which IPRs need to be initialized by power-up PALcode.

Table 92: IPR Reset State

IPR	Reset State	Comments
ITB_TAG	undefined	
ITB_PTE	undefined	
ICCSR	cleared except ASN, PC0, PC1	Floating point disabled, single issue mode, VAX mode enabled, ASN = 0, jsr predictions disabled, branch predictions disabled, branch history table disabled, performance counters reset to zero, Perf Cnt0(16b) : Total Issues/2, Perf Cnt1(12b) : Dcache Misses
ITB_PTE_TEMP	undefined	
EXC_ADDR	undefined	
SL_RCV	undefined	
ITBZAP	n/a	PALcode must do a itbzap on reset.
ITBASM	n/a	
ITBIS	n/a	
PS	undefined	PALcode must set processor status.
EXC_SUM	undefined	PALcode must clear exception summary and exception register write mask by doing 64 reads.
PAL_BASE	cleared	Cleared on reset.

Table 92 (Cont.): IPR Reset State

IPR	Reset State	Comments
HIRR	n/a	
SIRR	undefined	PALcode must initialize.
ASTRR	undefined	PALcode must initialize.
HIER	undefined	PALcode must initialize.
SIER	undefined	PALcode must initialize.
ASTER	undefined	PALcode must initialize.
SL_XMIT	undefined	PALcode must initialize. Appears on external pin.
DTB_CTL	undefined	PALcode must select between SP/LP dtb prior to any TB fill.
DTB_PTE	undefined	
DTB_PTE_TEMP	undefined	
MMCSR	undefined	Unlocked on reset.
VA	undefined	Unlocked on reset.
DTBZAP	n/a	PALcode must do a dtbzap on reset.
DTBASM	n/a	
DTBIS	n/a	
BIU_ADDR	undefined	Potentially locked.
BIU_STAT	undefined	Potentially locked.
SL_CLR	undefined	PALcode must initialize.
DC_ADDR	undefined	Potentially locked.
DC_STAT	undefined	Potentially locked.
FILL_ADDR	undefined	Potentially locked.
ABOX_CTL	see comments	[11..0] <- ^x0100 Write buffer enabled, machine checks disabled, correctable read interrupts disabled, lcache stream buffer disabled, Dcache disabled, forced hit mode off.
ALT_MODE	undefined	
CC	undefined	Cycle counter is disabled on reset.
CC_CTL	undefined	
BIU_CTL	see comments	Bcache disabled, parity mode undefined, chip enable asserts during RAM write cycles, Bcache forced-hit mode disabled. BC_PA_DIS field cleared. BAD_TCP cleared. BAD_DP undefined. Note: The Bcache parameters BC RAM read speed, BC RAM write speed, BC write enable control, and BC size are all undetermined on reset and must be initialized before enabling the Bcache.
FILL_SYNDROME	undefined	Potentially locked.
BC_TAG	undefined	Potentially locked.
PAL_TEMP[31..0]	undefined	

PALcode should execute four jsr call instructions to initialize the jsr stack. This is necessary to insure deterministic behavior for testers. The following code will initialize the stack once the ICCSR [JSE] bit is set.

```

stk_1:   BSR    r1,stk_1      ; push RET PC
          BSR    r2,stk_2      ; push RET PC
stk_2:   BSR    r3,stk_3      ; push RET PC
stk_3:   BSR    r4,stk_4      ; push RET PC
stk_4:

```

C.7 TB Miss Flows

This section describes hardware specific details to aid the PALcode programmer in writing ITB and DTB fill routines. These flows were included to highlight trade-offs and restrictions between PAL and hardware. The PALcode source that is released with the 21064 should be consulted before any new flows are written. A working knowledge of the ALPHA memory management architecture is assumed.

C.7.1 ITB Miss

When the Ibox encounters an ITB miss it latches the VPC of the target instruction-stream reference in the EXC_ADDR IPR, flushes the pipeline of any instructions following the instruction which caused the ITB miss, waits for any other instructions which may be in progress to complete, enters **PALmode**, and jumps to the ITB miss PAL entry point. The recommended PALcode sequence for translating the address and filling the ITB is described below.

1. Create some scratch area in the integer register file by writing the contents of a few integer registers to the PAL_TEMP register file.
2. Read the target virtual address from the EXC_ADDR IPR.
3. Fetch the PTE (this may take multiple reads) using a physical-mode HW_LD instruction. If this PTE's valid bit is clear report TNV or ACV as appropriate.
4. Since the ALPHA SRM states that translation buffers may not contain invalid PTEs, the PTE's valid bit must be explicitly checked by PALcode. Further, since the ITB's PTE RAM does not hold the FOE bit, the PALcode must also explicitly check this condition. If the PTE's valid bit is set and FOE bit is clear, PALcode may fill an ITB entry.
5. Write the original virtual address to the TB_TAG register using HW_MTPR. This writes the TAG into a temp register and not the actual tag field in the ITB.
6. Write the PTE to the ITB_PTE register using HW_MTPR. This HW_MTPR causes both the TAG and PTE fields in the ITB to be written. Note it is not necessary to delay issuing the HW_MTPR to the ITB_PTE after the MTPR to the ITB_TAG is issued.
7. Restore the contents of any modified integer registers to their original state using the HW_MFPR instruction.
8. Restart the instruction stream using the HW_REI instruction.

C.7.2 DTB Miss

When the Abox encounters a DTB miss it latches the referenced virtual address in the VA IPR and other information about the reference in the MMCSR IPR, and locks these registers against further modifications. The Ibox latches the PC of the instruction which generated the reference in the EXC_ADDR register, drains the machine as described above for ITB misses, and jumps to the DTB miss PALcode entry point. Unlike ITB misses, DTB misses may occur while the CPU is executing in **PALmode**. The recommended PALcode sequence for translating the address and filling the DTB is described below.

1. Create some scratch area in the integer register file by writing the contents of a few integer registers to the PAL_TEMP register file.
2. Read the requested virtual address from the VA IPR. Although the act of reading this register unlocks the VA and MMCSR registers, the MMCSR register only updates when D-stream memory management errors occur. It therefore will retain information about the instruction which generated this DTB miss. This may be useful later.
3. Fetch the PTE (may require multiple reads). If the valid bit of the PTE is clear, a TNV or ACV must be reported unless the instruction which caused the DTB miss was FETCH or FETCH/M. This can be checked via the opcode field of the MMCSR register. If the value in this field is 18 (hex), then a FETCH or FETCH/M instruction caused this DTB miss, and as mandated by the ALPHA SRM, the subsequent TNV or ACV should NOT be reported. Therefore PALcode should read the value in EXC_ADDR, increment it by four, write this value back to EXC_ADDR, and do a HW_REI.
4. Write the register which holds the contents of the PTE to the DTB_CTL IPR. This has the effect of selecting either the small or large page DTB for subsequent DTB fill operations, based on the value contained in the granularity hint field of the PTE.
5. Write the original virtual address to the TB_TAG register. This writes the TAG into a temp register and not the actual tag field in the DTB
6. Write the PTE to the DTB_PTE register. This HW_MTPR causes both the TAG and PTE fields in the DTB to be written. Note it is not necessary to delay issuing the HW_MTPR to the DTB_PTE after the MTPR to the DTB_TAG is issued.
7. Restore the contents of any modified integer registers.
8. Restart the instruction stream using the HW_REI instruction.

C.8 Error Flows

The following sections give a summary of the hardware flows for various error conditions recognized by the 21064. The 21064 recognizes hardware errors associated with off-chip interactions. These errors may or may not be corrected by hardware.

The 21064 reports corrected hardware errors through the maskable corrected-read interrupt. The ABOX_CTL[CRD_EN] bit controls whether the error detection logic generates an interrupt request for corrected errors. The HIER[CRE] bit is used to mask pending corrected-read interrupt requests. Corrected-read interrupts are masked when the CPU is in **PALmode**. The 21064 reports uncorrected hardware errors by generating a machine check trap to **PALmode**. The ABOX_CTL[MCHK_EN] bit control whether machine checks are generated by uncorrectable hardware errors. Machine checks (when enabled) are the highest priority exception in the system other than reset. They can occur while the CPU is operating in **PALmode**.

The 21064 recognized hardware errors occur during interactions between the 21064 BIU and off-chip hardware. These errors fall into three categories:

- Errors recognized by system components while processing 21064 generated requests and communicated to the 21064 pin bus command acknowledge signals (cAck_h [2:0]).
 - Correctable hardware errors communicated with a cAck_h code of HARD_ERROR.
- External cache tag probe errors recognized by the 21064 during 21064 controlled access of the B-Cache.
 - Tag address parity errors
 - Tag control parity errors
- Primary cache fill data errors recognized by the 21064. These errors could occur during 21064 controlled reads of the external cache, or during external read transactions between 21064 and system components.
 - EDC errors (EDC mode)

C.8.1 21064 Error Flows

C.8.1.1 EDC Error Handling

The 21064 corrects single-bit EDC errors in hardware. Only double-bit errors generate machine checks.

C.8.1.1.1 Single Bit I-stream EDC error

- Corrected data put into Icache, block gets validated
- CRD interrupt posted if enabled by ABOX_CTL[CRD_EN]
- BIU_STAT: FILL_EDC, FILL_IRD set, FILL_CRD set
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_SYNDROME contains syndrome bits associated with failing quadword

- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

C.8.1.1.2 Single Bit D-stream EDC error

- Corrected data put into Dcache, block gets validated
- CRD interrupt posted if enabled by ABOX_CTL[CRD_EN]
- BIU_STAT: FILL_EDC set, FILL_IRD clear, FILL_CRD set
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_ADDR[4..2] contain PA bits [4..2] of location which the failing load instruction attempted to read
- FILL_SYNDROME contains syndrome bits associated with failing quadword
- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

C.8.1.1.3 Double Bit I-stream EDC error

- Corrupted data put into Icache unchanged, block gets validated
- machine check if enabled by ABOX_CTL[MCHK_EN]
- BIU_STAT: FILL_EDC, FILL_IRD set, FILL_CRD clear
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_SYNDROME contains syndrome bits associated with failing quadword
- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE
- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

C.8.1.1.4 Double Bit D-stream EDC error

- Corrupted data put into Dcache unchanged, block gets validated
- machine check if enabled by ABOX_CTL[MCHK_EN]
- BIU_STAT: FILL_EDC set, FILL_IRD clear, FILL_CRD clear
- FILL_ADDR[33..5] & BIU_STAT[FILL_QW] give bad QW's address
- FILL_ADDR[4..2] contain PA bits [4..2] of location which the failing load instruction attempted to read
- FILL_SYNDROME contains syndrome bits associated with failing quadword
- BIU_ADDR, BIU_STAT[6..0] locked - contents are UNPREDICTABLE

- BC_TAG holds results of external cache tag probe if external cache was enabled for this transaction

C.8.1.2 BIU: tag address parity error

- recognized at end of tag probe sequence
- lookup uses predicted parity so transaction misses the external cache
- BC_TAG holds results of external cache tag probe
- machine check if enabled by ABOX_CTL[MCHK_EN]
- BIU_STAT: BC_TPERR set
- BIU_ADDR holds address

C.8.1.3 BIU: tag control parity error

- recognized at end of tag probe sequence
- transaction forced to miss external cache
- BC_TAG holds results of external cache tag probe
- machine check if enabled by ABOX_CTL[MCHK_EN]
- BIU_STAT: BC_TCPERR set
- BIU_ADDR holds address

C.8.1.4 BIU: system transaction terminated with CACK_HERR

- machine check if enabled by ABOX_CTL[MCHK_EN]
- BIU_STAT: BIU_HERR set, BIU_CMD holds cReq_h[2..0]
- BIU_ADDR holds address

C.8.2 Response to Multiple Errors

This section describes the 21064's response to multiple hardware errors, that is, to errors which occur after an initial error and before execution of the PALcode exception handler associated with that initial error.

The 21064 error reporting hardware consists of two sets of independent error reporting registers.

- BIU_STAT[7..0] and BIU_ADDR contain information about the following hardware errors:
 - Correctable or uncorrectable errors reported with CAck_h[2..0] by system components
 - Tag probe parity errors in the tag address or tag control fields
- BIU_STAT[14..8], FILL_ADDR and FILL_SYNDROME contain error information about data fill errors.

The BC_TAG register contains information which can relate to any of the error conditions listed above.

Each of the above two sets of error registers can contain information about either corrected or uncorrected hardware errors. When a hardware error occurs information about that error is loaded into the appropriate set of error registers and those registers are locked against further updates until PALcode explicitly unlocks them. If a second error occurs between the time that an initial error occurs and the time that software unlocks the associated error reporting registers, information about the second is lost.

When the 21064 recognizes the second error it still posts the required corrected read interrupt or machine check, however it does not overwrite information previously locked in an error reporting register. If the second hardware error is not correctable and the error reporting register normally associated with this second error is already locked, the 21064 will set a bit to indicate that information about an uncorrectable hardware error was lost. Each set of error reporting registers has a bit to report these fatal errors.

For example, BIU_STAT[FATAL1] is set by hardware to indicate that a tag probe parity error or HARD_ERROR terminated external transaction occurred while BIU_STAT[6..0], BIU_ADDR and BC_TAG were already locked due to some previous error. Similarly, BIU_STAT[FATAL2] is set by hardware to indicate that a primary cache fill received either a parity or single or double bit EDC error while BIU_STAT[13..8], FILL_ADDR, FILL_SYNDROME, and BC_TAG were already locked.

APPENDIX D

REVISION HISTORY

Table 93: Revision History

Revision	Date	Author	Description of change
0.1	1-April-93	Kurt Thaller	Sable Version 1
0.3	11-OCT-1993	Kurt Thaller	Fully updated Sable Version 3
1.0	16-FEB-1994	Kurt Thaller	Cleanup and release
1.1	5-AUG-1994	Kurt Thaller	Add B2024 (EV45) detail

INDEX

21064
IEEE floating point conformance, 57
PALcode instructions, 70

A

Aborts, 61
Abox, 37
data translation buffer, 53
load silos, 54
write buffer, 55
Abox IPRs, 37
ABOX_CTL, 42
Algorithms
Update vs Invalidate, 104
Alpha AXP
PALcode instructions, 69
ALT_MODE, 46
ASTER, 35
ASTRR, 32

B

Back-up Cache (B-Cache), 83
Back-up Cache Cycle Time, 101
BARRIER cycle, 143
BARRIER transaction, 153
B-Cache
see Back-up Cache
behavior on tag control parity error, 170
behavior on tag parity error, 169
defined, vi
B-Cache Control Register - CSR0, 86
B-Cache Control Register Definitions, 85
B-Cache Correctable Error Address Register -
CSR2, 95
B-Cache Correctable Error Register - CSR1,
92
B-Cache Initialization, 189
B-Cache Uncorrectable Error Address Register
- CSR4, 100
B-Cache Uncorrectable Error Register - CSR3,
97
BC_TAG, 79

BIU, vi
BIU_ADDR, 75
BIU_STAT, 74
Block diagrams
C4 chip, 3
CPU chip, 4
CPU module, 1
Branch prediction logic, 11
Bus interface unit, 54
BYSTANDER
defined, vi

C

C³ Revision Register - CSR15, 128
C4
defined, vi
C4block diagram, 3
C4 subsystems, 3
Cache
cycle time, 102
invalidate management, 165
System-bus B-Cache Access Time, 102
CACHE BLOCK
defined, vi
Cache Block Merge Buffer, 102
Cache Block Prefetch
lack of, 107
Cache hit
defined, vi
CACHE LINE
defined, vi
Cache miss
defined, vii
Cache organization, 59
CC, 46
CC_CTL, 46
CLEAN
defined, vi
Cobra-bus
defined, vi
COMMAND
defined, vi

- COMMANDER
 - defined, vi
- CONDITIONAL INVALIDATION
 - defined, vi
- Control Store, 83
- Conventions, v
- CPU chip block diagram, 4
- CPU module block diagram, 1
- CPU module transactions, 143
- CPU Module Transactions
 - Control Flow of, 158
 - processor initiated, 158
 - System-bus initiated, 162
- CPU subsystems, 1
- CPU System-bus Register Definitions, 107
- CRESET L, 136
- Cycle
 - defined, vi
- 21064 cycle types, 143
- Cycle types
 - LDQL, 144
 - LDxL, 144
 - STQC, 144
 - STxC, 144
- Cycle types
 - 21064, 143
 - BARRIER, 143
 - FETCH, 143
 - READ_BLOCK, 144
 - WRITE_BLOCK, 144

D

- Data cache, 59
- Data Integrity, 107
- Data Store, 84
- Data translation buffer, 53
- D-bus, 131
- D-BUS
 - defined, vi
- Dcache, 59
- DC_ADDR, 73
- DC_STAT, 70
- DIRTY
 - defined, vi
- DTBASM, 42
- DTBIS, 42
- DTBZAP, 41
- DTB_PTE, 39
- DTB_PTE_TEMP, 40
- Dual issue rules, 68
- Duplicate Primary Data Cache Tag Store, 103

- Duplicate Primary Instruction Cache Tag Store
 - lack of, 106
- Duplicate Tag Error Register - CSR5, 105
- Duplicate Tag Store Initialization, 191

E

- Ebox, 37
- E-BUS
 - defined, vi
- EDC Error Correction, 81
- Errors
 - SYSTEM FATAL, vii
- Exception handling, 168
- Exceptions
 - see interrupts and exceptions
 - PALcode 0020 entry characteristics, 169
 - PAL priority level, 169
 - PAL routine behavior, 169
- EXC_ADDR, 21
- EXC_SUM, 26
- Extents, v

F

- FAST EXTERNAL CACHE WRITE HIT, 147
- Fault management, 179
- Fbox, 56
- FEPROM
 - defined, vi
- FETCH cycle, 143
- FETCHM transaction, 154
- FETCH transaction, 153
- FILL_ADDR, 77
- FILL_SYNDROME, 78
- Floating point operate pipeline, 61
- FLUSH_IC, 42
- FLUSH_IC_ASM, 42

H

- Hardware Interrupt Request
 - defined, vi
- HIER, 33
- HIRR, 29

I

- Ibox
 - branch prediction logic, 11
 - instruction translation buffers, 12
 - super page, 12
 - virtual program counter, 12
- I-box internal processor registers, 11

- Icache, 59
- ICCSR, 19
- IEEE Floating point conformance, 57
- Initialization, 189
 - and reset, 191
 - and the B-Cache, 189
 - and the CPU clocks, 191
 - <of the System-bus interface, 191
- Initialization of the duplicate tag store, 191
- Instruction cache, 59
- Instruction class definition, 63
- Instruction issue rules, 67
- Instruction translation buffers, 12
- Integer operate pipeline, 60
 - static and dynamic stages, 61
- internal processor register format, 22
- Internal Processor Registers
 - and initialization, 189
- Interprocessor Interrupt Request Register - CSR11, 122
- Interrupt logic, 13
- Interrupts and Exceptions, 167
 - B-Cache behavior on data single bit EDC error, 170
 - B-Cache tag control parity error, 170
 - B-Cache tag parity error, 169
 - machine check, 168
 - processor generated, 167
- Interrupts and Exceptions B-Cache data
 - uncorrectable EDC error, 170
- Invalidate Management
 - C-bus caused, 165
- IPR
 - defined, vi
- ITBASM, 24
- ITBIS, 24
- ITBZAP, 24
- ITB_PTE, 18
- ITB_PTE_TEMP, 21

L

- LDQL cycle, 144
- LDxL cycle, 144
- LDxL transaction, 152
- Load silos, 54

M

- Machine check, 168
- Masked write
 - defined, vii
- Memory
 - bad on power-up, 192

- Memory like
 - defined, vii
- Memory reference pipeline, 60
- Miss Address Register - CSR14, 127
- MM_CSR, 41
- Multi-Processor Configuration CSR
 - Definitions, 121

N

- Non-issue conditions, 63
- Non-memory like
 - defined, vii
- Numbering conventions, v

P

- PALcode instructions, 69, 70
- PAL_BASE, 28
- PAL_TEMP, 70
- P-Cache
 - see also primary cache
 - defined, vii
- Performance counters, 14
- Physical address conventions, v
- Pipeline
 - aborts, 61
 - non-issue conditions, 63
 - organization, 59
- Power-up
 - with bad main memory, 192
- Power-up sequence, 191
- Probe
 - defined, vii
- PROCESS FATAL ERROR
 - defined, vii
- Processor Caused Invalidates, 165
- Processor Initiated Transactions, 145
- Processor Mailbox Register - CSR10, 121
- Processor status register, 25
- 21064 Processor Transactions
 - Allocate-Invalid, 155
 - BARRIER, 153
 - cacheable, 155
 - Cacheable vs Non-Cacheable vs Allocate-Invalid, 155
 - FAST EXTERNAL CACHE READ HIT, 146
 - FAST EXTERNAL CACHE WRITE HIT, 147
 - FETCH, 153
 - FETCHM, 154
 - LDxL, 152
 - non-cacheable, 155
 - READ BLOCK, 148
 - STxC, 152
 - WRITE_BLOCK, 150

- Producer-consumer classes, 64
- Producer-consumer latency, 65
 - matrix, 65
- Producer-producer latency, 67
- Protocols
 - snooping, vii
- PS
 - See processor status register

R

- Ranges, v
- READ BLOCK, 148
- Read-Merge
 - defined, vii
- READ_BLOCK cycle, 144
- Register
 - Data Translation Buffer Page Table Entry
 - Temporary, 40
- Register bit conventions, vi
- Registers
 - Abox control, 42
 - alternative processor mode, 46
 - Asynchronous System Trap Enable, 35
 - Asynchronous Trap Request Register (ASTRR), 32
 - Backup Cache, 79
 - B-Cache Correctable Error Address (CSR2), 95
 - B-Cache Uncorrectable Error Address (CSR4), 100
 - B-Cache Uncorrectable Error Register (CSR3), 97
 - Bus Interface Unit Address, 75
 - Bus Interface Unit Status, 74
 - Cycle Counter, 46
 - Cycle Counter Control, 46
 - Data Cache Address, 73
 - Data Cache Status, 70
 - Data Translation Buffer ASM, 42
 - Data Translation Buffer Invalidate Signal, 42
 - Data Translation Buffer Page Table Entry, 39
 - Data Translation Buffer Zap, 41
 - duplicate tag error (CSR5), 105
 - Exception Address, 21
 - Exception Summary, 26
 - Fill Address, 77
 - Fill Syndrome, 78
 - Flush Instruction Cache, 42
 - Flush Instruction Cache ASM, 42
 - Hardware Interrupt Enable, 33
 - Hardware Interrupt Request, 29
 - Instruction Cache Control and Status, 19

Registers (cont'd)

- Instruction Translation Buffer ASM, 24
- Instruction Translation Buffer IS, 24
- Instruction Translation Buffer Page Table Entry, 18
- Instruction Translation Buffer Page Table Entry Temporary, 21
- Instruction Translation Buffer Zap, 24
- Interprocessor Interrupt Request (CSR11), 122
- Memory Management Control and Status, 41
- Miss Address (CSR14), 127
- Multi-Processor Configuration CSR, 121
- Privileged Architecture Library Base, 28
- Processor Mailbox (CSR10), 121
- processor status, 25
- Revision (CSR15), 128
- Serial Line Clear, 23
- Serial Line Receive, 23
- Serial Line Transmit, 36
- Software Interrupt Enable, 34
- Software Interrupt Request, 31
- System-bus, 107
- System-bus Control (CSR6), 108
- System-bus Error (CSR7), 111
- System-bus Error Address High (CSR9), 118
- System-bus Error Address Low (CSR8), 117
- System Interrupt Clear (CSR12), 123
- Translation Buffer Control, 38
- Translation Buffer Tag, 16
- Virtual Address, 41
- Relation Documents, v
- Responder
 - defined, vii

S

- Scheduling and issuing rules, 63
- Serial control bus
 - defined, vii
- SHARED
 - defined, vii
- SIER, 34
- SIRR, 31
- SL_CLR, 23
- SL_RCV, 23
- SL_XMIT, 36
- Snoop
 - defined, vii
- Snooping protocol
 - defined, vii
- Specification Organization, v

- STQC cycle, 144
- STxC cycle, 144
- STxC TRANSACTION, 152
- Super page, 12
- System-bus
 - arbiter, 133
- System-bus Arbiter, 133
- System-bus Control Register - CSR6, 108
- System-bus CRESET L Generation, 136
- System-bus Cycles, 102
- System-bus Error Address High Register - CSR9, 118
- System-bus Error Address Low Register - CSR8, 117
- System-bus Error Register - CSR7, 111
- System-bus Interface, 107
- System-bus Interface Initialization, 191
- System-bus transactions, 143
 - reasons for monitoring, 143
- System-bus Transactions, 156
 - CPU as bystander, 157
 - CPU as commander, 157
- System-bus Transactions CPU as responder, 157
- SYSTEM FATAL ERROR
 - defined, vii
- System Interrupt Clear Register - CSR12, 123

T

- Tag Store, 84
- TB_CTL, 38
- TB_TAG, 16
- Terminology, vi
- The 21064 Cycles, 102
- TRANSACTION
 - defined, vii
- Transactions
 - CPU module, 143
 - processor initiated, 145
 - System-bus, 143

U

- Unmasked Write
 - defined, viii

V

- VA, 41
- Victim
 - defined, viii
- Victim processing
 - defined, viii

- Virtual program counter, 12

W

- Write buffer, 55
- WRITE_BLOCK cycle, 144
- WRITE_BLOCK transaction, 150