

IEEE Standard for Futurebus+ — Logical Protocol Specification

Sponsor
**Bus Architecture Standards Committee
of the
IEEE Computer Society**

Approved September 26, 1991
IEEE Standards Board

Abstract: IEEE Std 896.1-1991 provides a set of tools with which to implement a Futurebus+ architecture with performance and cost scalability over time, for multiple generations of single- and multiple-bus multiprocessor systems. Although this specification is principally intended for 64-bit address and data operation, a fully compatible 32-bit subset is provided, along with compatible extensions to support 128- and 256-bit data highways. Allocation of bus bandwidth to competing modules is provided by either a fast centralized arbiter, or a fully distributed, one or two pass, parallel contention arbiter. Bus allocation rules are provided to suit the needs of both real-time (priority based) and fairness (equal opportunity access based) configurations. Transmission of data over the multiplexed address/data highway is governed by one of two intercompatible transmission methods: (1) a technology-independent, compelled-protocol, supporting broadcast, broadcast, and transfer intervention (the minimum requirement for all Futurebus+ systems), and (2) a configurable transfer-rate, source-synchronized protocol supporting only block transfers and source-synchronized broadcast for systems requiring the highest possible performance. Futurebus+ takes its name from its goal of being capable of the highest possible transfer rate consistent with the technology available at the time modules are designed, while ensuring compatibility with all modules designed to this standard both before and after. The plus sign (+) refers to the extensible nature of the specification, and the hooks provided to allow further evolution to meet unanticipated needs of specific application architectures. It is intended that this standard be used as a key component of an approved IEEE Futurebus+ profile. An IEEE Recommended Practices document, P896.3, is under development for specific Futurebus+ applications.

Keywords: bus architecture, Futurebus+, logical protocol, multiprocessor systems

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

© 1992 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published in 1992
Printed in the United States of America

ISBN 1-55937-168-4

*No part of this publication may be reproduced in any form,
in an electronic retrieval system or otherwise,
without the prior written permission of the publisher.*

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Foreword

(This foreword is not a part of IEEE Std 896.1-1991, IEEE Standard for Futurebus+ — Logical Protocol Specification.)

The Futurebus+ Working Group has been exceptionally well-endowed with outstanding experts in the field of multiprocessor systems architecture and bus design. It seems that the principal goals of “getting it right” in the first phase of the development of this standard (IEEE Std 896.1-1987), followed by “getting it finished” in the second phase (Futurebus+), have proved to be powerful magnets to computer scientists and engineers in the beginning of the development of this standard, as well as to companies wishing to build products in the commercial, industrial, and military worlds towards the end of the development phase leading up to the approval of this standard.

Although this specification has been developed over many years by volunteers too numerous to mention, special acknowledgement goes to several individuals for their contribution.

This standard would never have been possible without the visionary outlook of Lyman Hevle, Executive Director of VITA; Shlomo Pri-Tal, Chair of the VITA Technical Committee; LCDR Harrison Beasley, Chair of the U.S. Navy NGCR Backplane Working Group; Paul Cook, Chair of the P1496 Rugged Bus Working Group and Vice Chair of the Futurebus+ Working Group; John Hyde, Chair of the P1296.1 Committee and representative of the Multibus Manufacturers Group; and Wayne Fischer, Chair of the P1014.1 Working Group.

Many individuals have excelled in their work on the Futurebus+ Working Group, some by the pure genius of their ideas, some by their tenacity and outstanding hard work. Paul Sweazey's work laid the foundation for the Cache Coherence model we have come to know as MOESI. This work unified the academic and commercial communities by providing a common way of looking at all protocols used to maintain consistency amongst multiple caches. The specifications for the cache protocols were brilliantly described by Bob Greiner, who extended the Futurebus+ cache model and proved that it could be made to work across multiple bus repeaters, giving the Futurebus+ a truly hierarchical consistency model, scalable over many processors on many interconnected bus segments. John Theus is the principal architect of the Futurebus+ parallel protocols. The packet protocols were developed from an initial proposal by John Theus for a separate source-supplied clock; this was refined to include an embedded clock scheme by Paul Sweazey, and Emil Hahn was instrumental in the final evolution to the current clockless packet protocol. Dr. Matthew Taub, inventor of the parallel contention arbiter, not only provided the Working Group with many innovations in the arbitration scheme, but kept the group “honest” in many other areas, too. Paul Dixon of Mizar took over leadership of the Arbitration Task Group, due to his intimate knowledge of the arbitration scheme. Paul's dedication and creativity helped to extend the facilities of the arbiter and to considerably improve its potential performance. Paul was also the principal inventor of the flow control method used in the packet transfer mode. R.V. (Balu) Balakrishnan, was instrumental in the development of Backplane Transceiver Logic on which the electrical specifications of some of the Futurebus+ profiles are based. Ed Jacques is acknowledged for his hard work and leadership skills in compiling the Message Passing Protocols from the difficult and sometimes conflicting requirements. Worthy of mention for their hard work and dedication are the remaining Task Group Coordinators who delivered their sections of the document despite the aggressive schedules that were set: Mike Foster, Coordinator of the Bus Management Section; and Jay Cantrell, Coordinator for the Cache Coherence Task Group.

Perhaps the biggest debt is owed to Joe George of Nanotek, Inc., whose energy and dedication to the editing and overall consistency of this document were singularly responsible for this specification being completed to the level of maturity and quality that now exists.

Joe George was promoted to Vice Chair of the P896.1 Working Group in recognition of this dedication and hard work.

Special thanks is extended to Professor Mary Vernon, of the University of Wisconsin, who turned down the potential financial rewards of her patents on the round-robin arbitration scheme, in order to allow the Futurebus+ specification to remain a truly open standard. The University of Wisconsin's cooperation is acknowledged in allowing anyone using these protocols in conformant implementations of the Futurebus+ standard to do so without payment of royalties or other legal impediments.

This document is one of several IEEE Standards developed under the Futurebus+ Steering Committee. The members are as follows: Paul Borrill, Futurebus+ Steering Committee Chair; Paul Cook, Futurebus+ Steering Committee Vice Chair; Lyman Hevle, Executive Director of VITA; Anatol Kaganovich, Treasurer of the Futurebus+ Working Group; Mark Karan, Secretary of the Working Group, who supported the administrative burden of this activity.

Also acknowledged are the Chairs of the related IEEE Standards and projects who also influenced the Futurebus+ significantly: Dave Gustavson, Chair of the P1596 (Scalable Coherent Interface) Working Group; Dave James, Chair of the P1212 CSR Architecture Working Group; Mike Teener, Chair of the P1394 Serial Bus; Hans Karlsson, Chair of IEEE Std 1301-1991 (Metric Mechanical); Eike Walz, Chair of IEEE Std 1101.1-1991 (Core Mechanical Specifications); Mike Humphrey, Chair of P1101.2 (Core Mechanical Specifications with 2 mm Connector for Convection-Cooled Equipment); Kim Clohessy, Chair of P1101.3 (Core Mechanical Specifications with 2 mm Connector for Conduction-Cooled Equipment); and Dale Younge, Editor of P1156 (Environmental Conditions).

The Futurebus+ (896) logo was drawn by Jan Borrill from an idea by Hugh Field-Richards. Todd Gustavson was responsible for the specification of the logo in PostScript, so that it may be accurately reproduced here.



The following is a list of those who were members of the IEEE Futurebus+ Working Group at the time this standard was approved:

Paul L. Borrill, *Chair*

Barbara Aichinger
Duane Anderson
Martin Blake
David Brewer
Jim Brown
Jay Cantrell
Paul Cook
Dante Del-Corso
Ian Dobson
Chris Eck
Joseph George
Robert Greiner
David Hartig
Billy Ho
Ed Jacques
Anatol Kaganovich
Ralph Lachenmaier
Dick Lawrence
Kent Leung
Klaus Müller
Ronald Niederhagen
Clarence Peckham
Mike Raynham
Ali Sarabi
Craig Scott
Dan Sieworek
Fahad Tabrizzi
Judy Teske
Mike Thompson
Harvey Walthersdorf
Mike Wenzel
Mark Williams
Dale Younge

Ray Alderman
Harrison Beasley
Richard Boberg
David Brash
Marc Briel
Mark Bunker
Stephen Cecil
Ernie Crocker
Steve Dieas
Emer Dooley
Bill Evertz
Wayne Fischer
Larry Gilbert
David Gustavson
David Hawley
Mike Humphrey
David James
Hans Karlsson
Subasis Laha
Joel Liblove
Chris Nichols
Mira Pauker
Shlomo Pri-Tal
Jack Regula
James Scaminaci
Don Senzig
Mike Snodgrass
Matthew Taub
Morton Thayer
Nigel Topham
Eike Waltz
Mike Wiles
David Wright

Hamid Amirazzi
Janos Biri
Andy Bonafini
David Brearly
Charles Brill
John Campbell
Kim Clohessy
Jon Crowell
Paul Dixon
Sam Duncan
Mike Foster
Jim Goodman
Emil Hahn
Lym Hevle
John Hyde
Greg Jewell
David Kemp
Cees Lambretche
Jim Leahy
Mike Lazar
Thanos Mentzelopoulos
Jim Nicholson
Chet Pawlowski
Surinder Rai
Bill Ruszczyk
Dennis Schmitz
Lui Sha
Michael Sweeney
Mike Teener
John Theus
Mary Vernon
Randy Weber
John Wise



The following persons were on the balloting committee that approved this document for submission to the IEEE Standards Board:

William B. Adams	William Groseclose	Mira Pauker
Sid Ahuja	David B. Gustavson	Donald Pavlovich
Mohammad Al-Malki	Thomas W. Harkaway	James M. Pexa
John Allen	David Hawley	Arthur V. Pohm
Richard P. Ames	Herbert Hecht	Bruno R. Preiss
Duane L. Anderson	Rick Henderson	Shlomo Pri-tal
Jack Arabian	Frank Hom	Greg Prom
R. V. Balakrishnan	Scott Hopkinson	Richard Rawson
David M. Barnum	Zoltan R. Hunor	Michael Raynham
Harrison A. Beasley	Peter J. Iliev	Ed Rodriguez
Janos Biri	Bob Jacobsen	Tom Sakoda
Kyle M. Black	Edgar Jacques	Debabrata Sarma
John Black	David V. James	Carl Schmiedekamp
William P. Blase	Kenneth Jansen	Norman Schneidewind
Jack L. Blevins	Jack R. Johnson	Eugene C. Schramm
David Brearley	Anatol Kaganovich	David Seraphin
Charles Brill	Hans Karlsson	Philip Shutt
Lyle Burnett	David Keeney	Michael R. Sitzer
Luis-Felipe Cabrera	Willis K. King	Michael Smolin
Clyde Camp	Hubert Kirrman	Benjamin Stoppe, Jr.
Donald Chi	Ernst H. Kristiansen	Paul Sweazey
Kim Clohessy	Thomas M. Kurihara	Daniel Tabak
David Cohen	Tuvia Lamdan	Darius Tanksalvala
Paul D. Cook	Glen Langdon	Daniel Tarrant
Robert Crowder	Thomas Leonard	Michael Teener
Jonathan C. Crowell	Per Lindman	Michael G. Thompson
Philip D'Angelo	William Lindow	Carsten Thomsen
Ana Maria Dealvare	Rollins Linser	Joseph P. Trainor
Stephen Deiss	Wayne M. Loucks	Robert Tripi
Dante Del Corso	Anthony G. Lubowe	Joseph G. Tront
Su Dongzhuang	Andy J. Luque	Robert J. Voight
Mike Dorsett	Roy Maurer	Eike Waltz
Samuel H. Duncan	William McDonald	David R. Weller
Sourav Dutta	Darrell B. McIndoe	Walter L. Whipple
Jeffrey S. Ebeling	Bruce Millard	Thomas Wicklund
William P. Evertz	Lee Minsuk	Hans A. Wiggers
Harry D. Feit	James M. Moidel	Mark Williams
Wayne Fischer	James Moloney	John S. Willy
Gordon Force	J.D. Nicoud	Andrew Wilson
Andrew Fraser	Tadahiko Nishimukai	John Wise
Joseph D. George	Duane J. Northcutt	Joel Witt
Andy Glew	Gregory C. Novak	David L. Wright
Patrick Gonia	Michael Orlovsky	Qiufeng Wu
Willard Graves	Jame R. Otto	Oren Yuen
	Dick Palmer	

When the IEEE Standards Board approved this standard on September 26, 1991, it had the following membership:

Donald C. Loughry, Vice Chair

Marco W. Migliaro, Chair

Andrew G. Salem, Secretary

Dennis Bodson	Thomas L. Hannan	John E. May, Jr.
Paul L. Borrill	Donald N. Heirman	Lawrence V. McCall
Clyde Camp	Kenneth D. Hendrix	Donald T. Michael*
James M. Daly	John W. Horch	Stig L. Nilsson
Donald C. Fleckenstein	Ben C. Johnson	John L. Rankine
Jay Forster*	Ivor N. Knight	Ronald H. Reimer
David F. Franklin	Joseph L. Koepfinger*	Gary S. Robinson
Ingrid Fromm	Irving Kolodny	Terrance R. Whittemore
	Michael A. Lawler	

*Member Emeritus

Paula M. Kelty
IEEE Standards Project Editor

Contents

SECTION	PAGE
1. Introduction.....	19
1.1 Scope.....	19
1.2 References.....	20
2. Definitions and Structure.....	21
2.1 Special Word Usage.....	21
2.2 Definitions.....	21
2.3 Signal Conventions.....	26
2.4 Document Structure	26
2.5 Futurebus+ Logo	27
2.6 Bus Line Description.....	27
2.6.1 Information Lines.....	27
2.6.1.1 AD[63..0]* Address/Data/Lane Deselect/Request Address/Return Address.....	27
2.6.1.2 D[255..64]* Data Lines.....	28
2.6.1.3 CM[7..0]* Command.....	28
2.6.1.4 CP* Command Parity.....	28
2.6.1.5 ST[7..0]* Status	28
2.6.1.6 CA[2..0]* Capability.....	29
2.6.1.7 BP[31..0]* Bus Parity.....	29
2.6.1.8 TG[7..0]* Tag.....	29
2.6.1.9 TP* Tag Parity.....	29
2.6.2 Synchronization Lines	29
2.6.2.1 AS* Address Synchronization.....	29
2.6.2.2 AK* Address Acknowledge.....	29
2.6.2.3 AI* Address Acknowledge Inverse	29
2.6.2.4 DS* Data Synchronization.....	29
2.6.2.5 DK* Data Acknowledge	29
2.6.2.6 DI* Data Acknowledge Inverse.....	29
2.6.2.7 ET* End of Tenure.....	29
2.6.3 Distributed Arbitration and Arbitrated Message Lines.....	29
2.6.3.1 AB[7..0]* Arbitration Bus.....	29
2.6.3.2 ABP* Arbitration Parity.....	29
2.6.3.3 AP*, AQ*, AR* Control Acquisition Synchronization	30
2.6.3.4 AC[1..0]* Arbitration Condition	30
2.6.4 RE* Reset/Bus Initialize	30
2.6.5 Central Arbiter.....	30
2.6.5.1 PE* Preemption.....	30
2.6.5.2 GR* Grant.....	30
2.6.5.3 RQ[1..0]* Request.....	30
2.6.6 GA[4..0]* Geographical Address	30
2.7 Attribute Cross Reference.....	30
2.8 Implementation Mnemonics.....	38
3. Bus Signaling Environment.....	39
3.1 Description.....	39
3.1.1 Incident Wave Switching.....	39
3.1.2 Skew.....	39
3.2 Specification	40
3.2.1 Skew.....	40
3.2.2 Glitch Filters.....	40

SECTION	PAGE
4. Centralized Arbitration	40
4.1 Description	40
4.1.1 Bus Lines Used for Centralized Arbitration.....	41
4.1.1.1 RQ0* Request 0.....	41
4.1.1.2 RQ1* Request 1.....	41
4.1.1.3 GR* Grant.....	41
4.1.1.4 PE* Preempt.....	41
4.1.2 Centralized Arbitration Operation	41
4.1.3 Central Arbiter Description	42
4.2 Specification	42
4.2.1 Bus Arbitration Attributes	42
4.2.2 Bus Arbitration Signals	43
4.2.2.1 RQ0*	43
4.2.2.2 RQ1*	43
4.2.2.3 GR*	43
4.2.2.4 PE*	43
5. Distributed Arbitration and Arbitrated Messages	43
5.1 Description	43
5.1.1 Arbitrated Messages—Central Arbiter	43
5.1.1.1 General Arbitrated Messages.....	44
5.1.1.2 Central Arbitrated Messages.....	44
5.1.2 Arbitration and Messages—Distributed Arbiter.....	44
5.1.2.1 Distributed Arbitration Messages	45
5.1.2.2 Distributed Arbitration Requests	45
5.1.2.2.1 Priority Field	45
5.1.2.2.2 Round-Robin Field.....	46
5.1.2.2.3 Geographical Address Field	46
5.1.2.3 Depositing the Master Elect	46
5.1.2.4 Transaction Preemption	46
5.1.2.5 Parking.....	47
5.1.3 Bus Lines	47
5.1.3.1 AP*, AQ*, AR* Control Acquisition Synchronization	47
5.1.3.2 AC[1..0]* Arbitration Condition	47
5.1.3.3 AB[7..0]* Arbitration Bus.....	47
5.1.3.4 ABP* Arbitration Parity.....	47
5.1.4 Arbitration Competition Logic.....	47
5.1.5 Arbitration Competition Settling Time	48
5.1.6 Arbitration States.....	49
5.1.7 Arbitration Phases	50
5.1.7.1 Phase 0: Idle Phase	51
5.1.7.2 Phase 1: Decision Phase	51
5.1.7.3 Phase 2: Competition Phase	52
5.1.7.4 Phase 3: Error Check.....	52
5.1.7.5 Phase 4: Master Release	52
5.1.7.6 Phase 5: Tenure/Message Transfer	53
5.1.8 Arbitration Examples.....	54
5.1.8.1 Single-Pass Competition (Central or Distributed).....	54
5.1.8.2 Single-Pass Deposed (Distributed) Competition	55
5.1.8.3 Distributed Arbitration Message	56
5.1.8.4 Two-Pass Competition (Central or Distributed)	57
5.1.8.5 Single-Pass Competition with Error	58
5.2 Specification	59

SECTION	PAGE
5.2.1	Arbitrated Message Attributes—Central Arbiter 59
5.2.2	Arbitrated Message Attributes—Distributed Arbiter 59
5.2.3	Arbitration Attributes—Distributed Arbiter 60
5.2.4	General Arbitration and Message Attributes 61
5.2.5	Arbitration Timing Attributes 62
5.2.6	Arbitration Error Attributes 63
5.2.7	Signal Definition 63
5.2.7.1	Arbitration Synchronization 63
5.2.7.1.1	AP* 63
5.2.7.1.2	AQ* 64
5.2.7.1.3	AR* 64
5.2.7.2	Arbitration Condition 64
5.2.7.2.1	AC0* 64
5.2.7.2.2	AC1* 64
5.2.7.3	Arbitration Bus 64
5.2.7.3.1	AB7* 64
5.2.7.3.2	AB6* 64
5.2.7.3.3	AB5* 64
5.2.7.3.4	AB4* 64
5.2.7.3.5	AB3* 64
5.2.7.3.6	AB2* 64
5.2.7.3.7	AB1* 64
5.2.7.3.8	AB0* 64
5.2.7.3.9	ABP* 64
5.2.8	Protocol Definition—Distributed Arbitration and Messages 64
5.2.8.1	Phase 0—Idle 64
5.2.8.2	Phase 1—Decision 65
5.2.8.3	Phase 2—Competition 65
5.2.8.4	Phase 3—Error Check 65
5.2.8.5	Phase 4—Release Mastership 66
5.2.8.6	Phase 5—Transfer Tenure 66
5.2.9	Protocol Definition—Central Arbiter Messages 66
5.2.9.1	Phase 0—Idle 66
5.2.9.2	Phase 1—Decision 67
5.2.9.3	Phase 2—Competition 67
5.2.9.4	Phase 3—Error Check 67
5.2.9.5	Phase 4—Release Mastership 67
5.2.9.6	Phase 5—Transfer Tenure 67
6.	Parallel Protocol 69
6.1	Description 69
6.1.1	Mastership 69
6.1.2	Transactions 69
6.1.3	Bus Transaction Phases 69
6.1.4	Data Transfer Protocols 70
6.1.5	Broadcast Handshake 70
6.1.6	Intervention 71
6.1.7	Cache Line Status 71
6.1.8	Split Transactions 71
6.1.9	Locked Operations 71
6.1.10	Lock Commands 72
6.1.10.1	Mask and Swap Lock Command 72
6.1.10.2	Fetch and Add Lock Command 73

SECTION

PAGE

6.1.10.3	Compare and Swap Lock Command.....	74
6.1.11	Busy.....	76
6.1.12	Wait.....	76
6.1.13	Extended Bus Width.....	76
6.1.14	Extended Address.....	76
6.1.15	Module Capabilities.....	77
6.1.16	Transactions.....	77
6.1.16.1	Read Unlocked.....	77
6.1.16.2	Write Unlocked.....	77
6.1.16.3	Address Only Unlocked.....	77
6.1.16.4	Read Locked.....	78
6.1.16.5	Write Locked.....	78
6.1.16.6	Address Only Locked.....	78
6.1.16.7	Read Partial.....	78
6.1.16.8	Write Partial.....	78
6.1.16.9	Read Partial Locked.....	78
6.1.16.10	Write Partial Locked.....	78
6.1.16.11	Write Response.....	78
6.1.16.12	Read Response.....	78
6.1.16.13	Write No Acknowledge.....	79
6.1.16.14	Read Invalid.....	79
6.1.16.15	Write Invalid.....	79
6.1.16.16	Copyback.....	79
6.1.16.17	Read Shared.....	79
6.1.16.18	Read Modified.....	79
6.1.16.19	Invalidate.....	79
6.1.16.20	Shared Response.....	79
6.1.16.21	Modified Response.....	80
6.1.17	Bus Signal Descriptions.....	80
6.1.17.1	AS* Address Sync.....	80
6.1.17.2	AK* Address Acknowledge.....	80
6.1.17.3	AI* Address Acknowledge Inverse.....	80
6.1.17.4	DS* Data Sync.....	80
6.1.17.5	DK* Data Acknowledge.....	80
6.1.17.6	DI* Data Acknowledge Inverse.....	81
6.1.17.7	CM[7..0]* Command Field.....	81
6.1.17.7.1	CM7* Command 7.....	81
6.1.17.7.2	CM6* Command 6.....	82
6.1.17.7.3	CM5* Command 5.....	82
6.1.17.7.4	CM4* Command 4.....	83
6.1.17.7.5	CM3* Command 3.....	83
6.1.17.7.6	CM2* Command 2.....	84
6.1.17.7.7	CM1* Command 1.....	84
6.1.17.7.8	CM0* Command 0.....	84
6.1.17.8	CP* Command Parity.....	84
6.1.17.9	ST[7..0]* Status.....	84
6.1.17.9.1	ST7* Status 7.....	85
6.1.17.9.2	ST6* Status 6.....	85
6.1.17.9.3	ST5* Status 5.....	85
6.1.17.9.4	ST4* Status 4.....	85
6.1.17.9.5	ST3* Status 3.....	85
6.1.17.9.6	ST2* Status 2.....	86



SECTION	PAGE
6.1.17.9.7	ST1* Status 1 86
6.1.17.9.8	ST0* Status 0 86
6.1.17.10	CA[2..0]* Capability.....86
6.1.17.10.1	CA2* Capability 2.....87
6.1.17.10.2	CA1* Capability 1.....87
6.1.17.10.3	CA0* Capability 0.....87
6.1.17.11	AD[31..0]* Address/Data/Lane Deselect/Return Address .87
6.1.17.12	AD[63..32]* Address/Data.....88
6.1.17.13	D[255..64]* Data.....88
6.1.17.14	BP[31..0]* Byte Parity.....88
6.1.17.15	TG[7..0]* Tag.....89
6.1.17.16	TP* Tag Parity.....89
6.1.17.17	ET* End of Tenure.....89
6.1.18	Bus Beats.....89
6.1.18.1	Single Slave Bus Beat.....89
6.1.18.2	Multiple Slave Bus Beat.....90
6.1.19	Transaction Examples 91
6.1.19.1	Address Only Transaction.....91
6.1.19.2	Compelled Read.....93
6.1.19.3	Compelled Read Broadcast/Broadcast.....94
6.1.19.4	Compelled Read Intervention.....95
6.1.19.5	Compelled Read Partial.....96
6.1.19.6	Compelled Write.....97
6.1.19.7	Compelled Write Broadcast.....98
6.1.19.8	Compelled Write Partial.....99
6.1.19.9	Transaction Error.....100
6.1.19.10	Packet Data Phase.....101
6.1.19.11	Single Packet Read.....102
6.1.19.12	Single Packet Read Broadcast.....103
6.1.19.13	Single Packet Read Intervention.....104
6.1.19.14	Single Packet Write.....105
6.1.19.15	Multiple Packet Read Intervention.....106
6.1.19.16	Multiple Packet Read Intervention Queued.....107
6.2	Specification 108
6.2.1	Attribute Definition 108
6.2.1.1	Transfer Protocol Attributes 108
6.2.1.2	Split Transaction Attributes.....108
6.2.1.3	Address/Data Width Attributes.....109
6.2.1.4	Transaction Attributes.....109
6.2.1.5	Locked Operations/Attributes.....113
6.2.1.6	Data Transfer Length.....114
6.2.1.7	Transaction Status.....115
6.2.1.7.1	Master Transaction Status.....115
6.2.1.7.2	Module Transaction Status.....116
6.2.1.8	Transaction Timing Attributes 118
6.2.2	Signal Definition.....121
6.2.2.1	Synchronization Signals.....124
6.2.2.1.1	AS* (Address Sync).....124
6.2.2.1.2	AK* (Address Acknowledge).....124
6.2.2.1.3	AI* (Address Acknowledge Inverse) 124
6.2.2.1.4	DS* (Data Sync) 124
6.2.2.1.5	DK* (Data Acknowledge) 124

SECTION		PAGE
	6.2.2.1.6 DI* (Data Acknowledge Inverse).....	122
6.2.2.2	CM[7..0]* (Command).....	122
	6.2.2.2.1 CM0* (Command 0).....	122
	6.2.2.2.2 CM1* (Command 1).....	122
	6.2.2.2.3 CM2* (Command 2).....	122
	6.2.2.2.4 CM3* (Command 3).....	122
	6.2.2.2.5 CM4* (Command 4).....	122
	6.2.2.2.6 CM5* (Command 5).....	122
	6.2.2.2.7 CM6* (Command 6).....	122
	6.2.2.2.8 CM7* (Command 7).....	122
	6.2.2.2.9 CP* (Command Parity).....	122
6.2.2.3	CA[2..0]* (Capability).....	122
	6.2.2.3.1 CA0* (TS*).....	122
	6.2.2.3.2 CA1* (CO*).....	122
	6.2.2.3.3 CA2* (SR*).....	123
6.2.2.4	ST[7..0]* (Status).....	123
	6.2.2.4.1 ST0* (TE*).....	123
	6.2.2.4.2 ST1* (BS*/ED*).....	123
	6.2.2.4.3 ST2* (SL*).....	123
	6.2.2.4.4 ST3* (BC*).....	123
	6.2.2.4.5 ST4* (TF*).....	123
	6.2.2.4.6 ST5* (IV*).....	123
	6.2.2.4.7 ST6* (BE*).....	123
	6.2.2.4.8 ST7* (WT*).....	124
6.2.2.5	Information Fields.....	124
	6.2.2.5.1 AD[63..0]* (Address/Data).....	124
	6.2.2.5.2 D[255.64]* (Data).....	127
	6.2.2.5.3 BP[31..0]* (Bus Parity).....	127
	6.2.2.5.4 TG[7..0]* (Tag).....	129
	6.2.2.5.5 TP* (Tag Parity).....	129
	6.2.2.5.6 ET*.....	129
6.2.3	Protocol Definition.....	129
	6.2.3.1 Master Connection Phase.....	129
	6.2.3.2 Slave Connection Phase.....	130
	6.2.3.3 Master Compelled Data Phase—First Odd Beat.....	131
	6.2.3.4 Slave Compelled Data Phase—First Odd Beat.....	131
	6.2.3.5 Master Compelled Data Phase—First Even Beat.....	132
	6.2.3.6 Slave Compelled Data Phase—First Even Beat.....	132
	6.2.3.7 Master Compelled Data Phase—Second Odd Beat.....	133
	6.2.3.8 Slave Compelled Data Phase—Second Odd Beat.....	133
	6.2.3.9 Master Compelled Data Phase—Third and Subsequent Odd Beats.....	134
	6.2.3.10 Slave Compelled Data Phase—Third and Subsequent Odd Beats.....	134
	6.2.3.11 Master Compelled Data Phase—Second and Subsequent Even Beats.....	134
	6.2.3.12 Slave Compelled Data Phase—Second and Subsequent Even Beats.....	135
	6.2.3.13 Master Packet Data Phase—Single Packet Mode.....	135
	6.2.3.14 Slave Packet Data Phase—Single Packet Mode.....	136
	6.2.3.15 Master Packet Data Phase—Multiple Packet Mode— Odd Packet Queueing.....	136

SECTION	PAGE
6.2.3.16	Slave Packet Data Phase—Multiple Packet Mode— Odd Packet Queueing..... 136
6.2.3.17	Master Packet Data Phase—Multiple Packet Mode— Even Packet Queueing..... 136
6.2.3.18	Slave Packet Data Phase—Multiple Packet Mode— Even Packet Queueing..... 137
6.2.3.19	Master Packet Data Phase—Multiple Packet Mode— Packet Transmission..... 137
6.2.3.20	Slave Packet Data Phase—Multiple Packet Mode— Packet Transmission..... 137
6.2.3.21	Packet Data Protocol 138
6.2.3.22	Master Disconnection Phase..... 138
6.2.3.23	Slave Disconnection Phase..... 139
6.2.3.24	Transaction Timeout Recovery 139
7.	Bus/System Management..... 140
7.1	Description..... 140
7.1.1	Bus Control..... 140
7.1.1.1	Power-up..... 140
7.1.1.2	System Reset 140
7.1.1.3	Bus Initialize..... 141
7.1.1.4	Live Insertion..... 141
7.1.1.5	Live Withdrawal..... 141
7.1.2	Futurebus+ Control and Status Registers 142
7.1.2.1	Module Capability Registers 142
7.1.2.1.1	Module Logical Capability Register..... 142
7.1.2.1.2	Competition Internal Delay Register 144
7.1.2.1.3	Message Frame Size Register..... 144
7.1.2.1.4	Packet Speed Register..... 144
7.1.2.2	Module Control Registers..... 144
7.1.2.2.1	Logical Common Control Register 144
7.1.2.2.2	Logical Module Control Register..... 145
7.1.2.2.3	Bus Propagation Delay Register..... 146
7.1.2.2.4	Competition Settling Time Register..... 146
7.1.2.2.5	Transaction Timeout Register 146
7.1.2.2.6	Message Passing Select Mask 147
7.2	Specification 147
7.2.1	Bus Control Attributes..... 147
7.2.2	RE* Reset Signal 148
7.2.3	Protocol Definition 148
7.2.3.1	Power up 148
7.2.3.2	System Reset 148
7.2.3.3	Bus Initialize..... 148
7.2.3.4	Live Insertion..... 148
7.2.3.5	Live Withdrawal..... 149
7.2.4	Futurebus+ Control and Status Registers 149
7.2.4.1	Module Capability Registers 149
7.2.4.1.1	Module Logical Capability Register..... 149
7.2.4.1.2	Competition Internal Delay Register 151
7.2.4.1.3	Message Frame Size Register..... 151
7.2.4.1.4	Packet Speed Register..... 151
7.2.4.2	Module Control Registers..... 151
7.2.4.2.1	Logical Common Control Register..... 151

SECTION	PAGE
7.2.4.2.2	Logical Module Control Register..... 153
7.2.4.2.3	Bus Propagation Delay Register..... 153
7.2.4.2.4	Competition Settling Time Register..... 153
7.2.4.2.5	Transaction Timeout Register 154
7.2.4.2.6	Message Passing Select Mask 154
8. Cache Coherence	155
8.1 Description.....	155
8.1.1 Cache Attributes.....	156
8.1.2 Bus Snooping.....	156
8.1.3 Cache Coherence Using Connected Transactions.....	157
8.1.3.1 State Changes	157
8.1.3.2 Read Miss Process.....	158
8.1.3.3 Write Miss Process.....	159
8.1.3.4 Copyback Process.....	159
8.1.3.5 I/O Controller Optimization.....	159
8.1.3.6 Examples.....	160
8.1.3.6.1 One Processor Reads, Then Writes, a Location.....	161
8.1.3.6.2 All Processors Simultaneously Read and Write a Location.....	161
8.1.3.6.3 Cache Resident Fetch and Add.....	162
8.1.4 Cache Coherence Using Split Transactions on a Single Bus Segment.....	163
8.1.4.1 Split Transactions.....	163
8.1.4.1.1 Shared Response.....	163
8.1.4.1.2 Modified Response	163
8.1.4.2 Single Outstanding Transaction Per Cache Line.....	164
8.1.4.3 Requester and Responder Attributes	164
8.1.4.4 I/O Controller Optimization.....	164
8.1.4.5 Examples.....	164
8.1.4.5.1 One Processor Reads, Then Writes, a Location.....	164
8.1.4.5.2 Three Processors Simultaneously Read and Write a Location.....	165
8.1.5 Using Split Transactions to Delay Invalidation Completions.....	166
8.1.5.1 Cache Agents.....	167
8.1.5.2 Memory Agents.....	167
8.1.5.3 Split Transactions.....	168
8.1.5.3.1 Read Modified	168
8.1.5.3.2 Invalidate	168
8.1.5.3.3 Write Invalid.....	169
8.1.5.3.4 Modified Response	169
8.1.5.4 Requester and Responder Attributes	169
8.1.5.5 Multiple Outstanding Transactions per Cache Line.....	169
8.1.5.6 Examples.....	170
8.1.5.6.1 One Processor Reads and Writes a Location..	171
8.1.5.6.2 Four Processors Simultaneously Read and Write a Location.....	173
8.1.6 Summary of Cache Coherence Commands and Status.....	178
8.1.7 Illegal Attribute Combinations	178
8.1.7.1 Shared Unmodified	178
8.1.7.2 Exclusive Unmodified.....	178
8.1.7.3 Exclusive Modified.....	178

SECTION	PAGE
8.1.7.4	Memory Line Modified..... 178
8.2	Specification 178
8.2.1	Module Attributes..... 178
8.2.2	Status Attributes..... 178
8.2.3	Cache Module Attributes Per Cache Line 181
8.2.4	Requester Attributes Per Cache Line 182
8.2.5	Responder Attributes Per Cache Line 182
8.2.6	Protocol Definition..... 183
8.2.6.1	Cache Line Size..... 183
8.2.6.2	Cache Line Transfers..... 183
9.	Message Passing..... 184
9.1	Description 184
9.1.1	Frame Level..... 185
9.1.1.1	Frame Structure 185
9.1.1.2	Mailbox Structure..... 187
9.1.1.2.1	Request Mailbox..... 187
9.1.1.2.2	Response Mailbox 187
9.1.1.2.3	Broadcast Transactions..... 188
9.1.1.3	Frame Level Protocol 189
9.1.1.4	Message Priority 190
9.1.1.5	Frame Level Example 190
9.1.2	Message Level 191
9.1.2.1	Message Level Frame Structure..... 191
9.1.2.1.1	Frame Type 191
9.1.2.1.2	Frame Size 193
9.1.2.1.3	Message Identification 193
9.1.2.1.4	Extended Header 193
9.1.2.1.5	Message Size 193
9.1.2.1.6	Frame Interval 193
9.1.2.1.7	Sequence Number 193
9.1.2.1.8	Exception Type..... 193
9.1.2.1.9	Unacknowledged Event Frame..... 193
9.1.2.1.10	Acknowledged Event Frame..... 194
9.1.2.1.11	Event Positive Acknowledge Frame..... 194
9.1.2.1.12	Event Negative Acknowledge Frame..... 194
9.1.2.1.13	Multiple Frame Message Request Frame 194
9.1.2.1.14	Multiple Frame Message Positive Response Frame..... 194
9.1.2.1.15	Multiple Frame Message Negative Response Frame..... 194
9.1.2.1.16	Multiple Frame Message Ordered Data Frame..... 194
9.1.2.1.17	Multiple Frame Message Sequenced Data Frame..... 195
9.1.2.1.18	Multiple Frame Message Last Ordered Frame..... 195
9.1.2.1.19	Multiple Frame Message Last Sequenced Frame..... 195
9.1.2.1.20	Multiple Frame Message Positive 195
9.1.2.1.21	Multiple Frame Message Negative Acknowledge Frame..... 195
9.1.2.2	Message Protocols..... 196

SECTION

PAGE

	9.1.2.2.1	Unacknowledged Single Frame Protocol	196
	9.1.2.2.2	Acknowledged Single Frame Protocol.....	196
	9.1.2.2.3	Ordered Multiple Frame Protocol	196
	9.1.2.2.4	Sequenced Multiple Frame Protocol.....	197
	9.1.2.3	Request/Response Timeout.....	198
	9.1.2.4	Multiple Frame Message Timeout	198
	9.1.2.5	Frame Ordering.....	198
	9.1.2.6	Bus Tenure Time.....	199
9.2	Specification		199
9.2.1	Attribute Definition		199
9.2.2	Frame Format Specification		200
	9.2.2.1	Frame Format—Non-Futurebus+ Message Level.....	200
	9.2.2.2	Frame Format—Unacknowledged Event Frame	200
	9.2.2.3	Frame Format—Acknowledged Event Frame.....	201
	9.2.2.4	Frame Format—Event Positive Acknowledge Frame....	201
	9.2.2.5	Frame Format—Event Negative Acknowledge Frame....	201
	9.2.2.6	Frame Format—Multiple Frame Message Request Frame	202
	9.2.2.7	Frame Format—Multiple Frame Message Positive Response Frame	202
	9.2.2.8	Frame Format—Multiple Frame Message Negative Response Frame	203
	9.2.2.9	Frame Format—Multiple Frame Message Ordered Data Frame.....	203
	9.2.2.10	Frame Format—Multiple Frame Message Sequenced Data Frame.....	203
	9.2.2.11	Frame Format—Multiple Frame Message Last Ordered Frame	204
	9.2.2.12	Frame Format—Multiple Frame Message Last Sequenced Frame.....	204
	9.2.2.13	Frame Format—Multiple Frame Message Positive Acknowledge Frame.....	205
	9.2.2.14	Frame Format—Multiple Frame Message Negative Acknowledge Frame.....	205
9.2.3	Message Size		205
9.2.4	Frame Interval		205
9.2.5	Sequence Number		205
9.2.6	Exception Type Field.....		206
9.2.7	Protocol Definition		206
	9.2.7.1	Master Unacknowledged Single Frame Message Protocol	206
	9.2.7.2	Slave Unacknowledged Single Frame Message Protocol	206
	9.2.7.3	Master Acknowledged Single Frame Message Protocol...	206
	9.2.7.4	Slave Acknowledged Single Frame Message Protocol ...	206
	9.2.7.5	Master Ordered Multiple Frame Message Protocol	206
	9.2.7.6	Master Sequenced Multiple Frame Message Protocol.....	207
	9.2.7.7	Slave Multiple Frame Message Protocol.....	207

FIGURES

Fig 1	Interfaces in a Family of Typical Futurebus+ Systems.....	20
Fig 2	Signal Conventions.....	26
Fig 3	IEEE 896 Logo.....	27
Fig 4	Two-Competitor Centralized Arbitration	42
Fig 5(a)	Serial Implementation of Parallel Contention Logic.....	48



Documents

Document Sharing Hub

FIGURES	PAGE
5(b) Parallel Implementation of Parallel Contention Logic.....	48
Fig 6 Arbitration Competition Example.....	49
Fig 7 Control Acquisition States	50
Fig 8 Control Acquisition Sequence.....	51
Fig 9 Single-Pass Competition.....	54
Fig 10 Single-Pass Deposed Competition.....	55
Fig 11 Arbitration Message.....	56
Fig 12 Two-Pass Competition.....	57
Fig 13 Single-Pass Competition with Error.....	58
Fig 14 Mask and Swap Lock Command.....	73
Fig 15 Fetch and Add Lock Command.....	74
Fig 16 Compare and Swap Lock Command	75
Fig 17 Single Slave Bus Beat.....	90
Fig 18 Multiple Slave Bus Beat	91
Fig 19 Address Only Transaction	92
Fig 20 Compelled Read Transaction	93
Fig 21 Compelled Read Transaction with Broadcast	94
Fig 22 Compelled Read Transaction with Intervention.....	95
Fig 23 Compelled Read Partial Transaction	96
Fig 24 Compelled Write Transaction	97
Fig 25 Compelled Write Transaction with Broadcast.....	98
Fig 26 Compelled Write Partial Transaction	99
Fig 27 Transaction Error Timing.....	100
Fig 28 Packet Data Phase Timing.....	101
Fig 29 Single Packet Read Transaction	102
Fig 30 Single Packet Read Transaction with Broadcast	103
Fig 31 Single Packet Read Transaction with Intervention.....	104
Fig 32 Single Packet Write Transaction	105
Fig 33 Multiple Packet Read Transaction with Intervention.....	106
Fig 34 Multiple Packet Read Transaction with Intervention and Queueing.....	107
Fig 35 Power-up Sequence.....	140
Fig 36 System Reset Sequence	141
Fig 37 Module Logical Capability Register.....	143
Fig 38 Competition Internal Delay Register.....	144
Fig 39 Message Frame Size Register	144
Fig 40 Packet Speed Register	144
Fig 41 Logical Common Control Register.....	145
Fig 42 Logical Module Control Register.....	146
Fig 43 Bus Propagation Delay Register.....	146
Fig 44 Competition Settling Time Register.....	146
Fig 45 Transaction Time-out Register.....	146
Fig 46 Message Passing Select Mask.....	147
Fig 47 Shared Memory Architecture.....	155
Fig 48 Invalid State	157
Fig 49 Shared Unmodified State.....	157
Fig 50 Exclusive Unmodified State	158
Fig 51 Exclusive Modified State.....	158
Fig 52 Connected Cache Coherence Model	160
Fig 53 Split Response Architecture.....	167
Fig 54 Multiple Bus Segment Split Transaction Example Model	171
Fig 55 Futurebus+ Message Passing.....	184
Fig 56 Frame Notation	185

FIGURES	PAGE
Fig 57 64-Byte Frame Format.....	186
Fig 58 Message-Passing Mailboxes.....	188
Fig 59 Broadcast Mailbox Select Attribute.....	189
Fig 60 Linked List Message Passer.....	190
Fig 61 Futurebus+ Message Level Frame Structure.....	192
Fig 62 Unacknowledged Single Frame Message.....	196
Fig 63 Acknowledged Single Frame Message	196
Fig 64 Ordered Multiple Frame Message	197
Fig 65 Sequenced Multiple Frame Message	198

TABLES	
Table 1 List of Bus Lines.....	28
Table 2 General Arbitrated Message Fields	44
Table 3 Central Arbitrated Message Fields	44
Table 4 Distributed Arbitration Message Fields.....	45
Table 5 Distributed Arbitration Request Fields.....	45
Table 6 Command Field Encoding.....	81
Table 7 Data Transfer Length.....	82
Table 8 Data Path Width.....	82
Table 9 Transaction Command Encoding.....	83
Table 10 Locked Command Encoding	84
Table 11 Status Encoding.....	85
Table 12 Capability Encoding.....	86
Table 13 Event and RE* Relationships	140



IEEE Standard for Futurebus+ — Logical Protocol Specification

1. Introduction

1.1 Scope. This IEEE standard specifies the logical (relative timing and behavioral protocol) layer for a set of signal lines that constitute a multiple segment bus architecture, and for the interfacing of modules connected to a bus segment. This standard is intended to be used as a component within a profile (a collection of related specifications that must be used together by a product in order to claim conformance to a standard) to build systems with higher levels of compatibility.

Futurebus+ provides the means for the transfer of binary information between boards over one or more logical buses. Boards may contain any combination of one or more processors and local resources such as cache, memory, peripheral and communication controllers, etc. Figure 1 shows a block diagram of a typical application of Futurebus+.

Protocols are specified for the allocation of bus time to modules that need to conduct transactions with other modules over the bus. However, the standard does not mandate the priority rules for modules to use when competing for use of the bus. These are considered the privilege and responsibility of the system integrator. The standard includes a complete set of signaling rules to be followed by all modules in both the distributed and centralized control acquisition processes leading to bus mastership (Sections 4 and 5). The standard also gives a comprehensive set of signaling rules for all modules participating in a bus transaction (Section 6).

Most of the transfer protocols in this standard are *compelled*; that is, they are governed by a pure cause-and-effect relationship. This is what gives this standard its technology-independent nature. The compelled signaling provides a designer with a logical simplicity for what takes place in the protocols. As a result, there will be maximum compatibility between products designed to this standard throughout its operational lifetime.

With any bus, there is the dilemma of how much the standard should specify. There must be a balance between ensuring that all boards designed by a variety of manufacturers can operate together, while not restricting the users of the bus to any preconceived system design. Although the scope of this document has been restricted to exclude many of the system requirements associated with bus-based computer systems, these are being addressed in companion standards.

The common control and register interface to this series of standards for the Futurebus+, and to other proposed IEEE standards (in particular, IEEE P1596 [16]¹, IEEE P1014.1 [5], and IEEE P1296.1 [12]), is embodied in the unified CSR architecture standard, IEEE Std 1212-1991 [10], along with a unified DMA architecture for moving data around a system without the need to pass through a processor (P1212.1 [11]).

This set of protocols has been designed to be as close to technology-independent as possible while maintaining a very high level of efficiency and performance. The bus signals may be implemented using any technology (TTL, Backplane Transceiver Logic, ECL, CMOS, GaAs, etc.) so long as the Futurebus+ signaling conditions are met (incident wave switching on the transmission-line signaling environment, along with the constraints on skew, crosstalk, and transmission reliability). However, in the interest of maximum compatibility between product families, implementations are expected to be associated with one or more IEEE Futurebus+ profiles, which specify the physical layer and set of transactions to suit a particular family of applications.

¹ The numbers in brackets correspond to those of the references in 1.2.

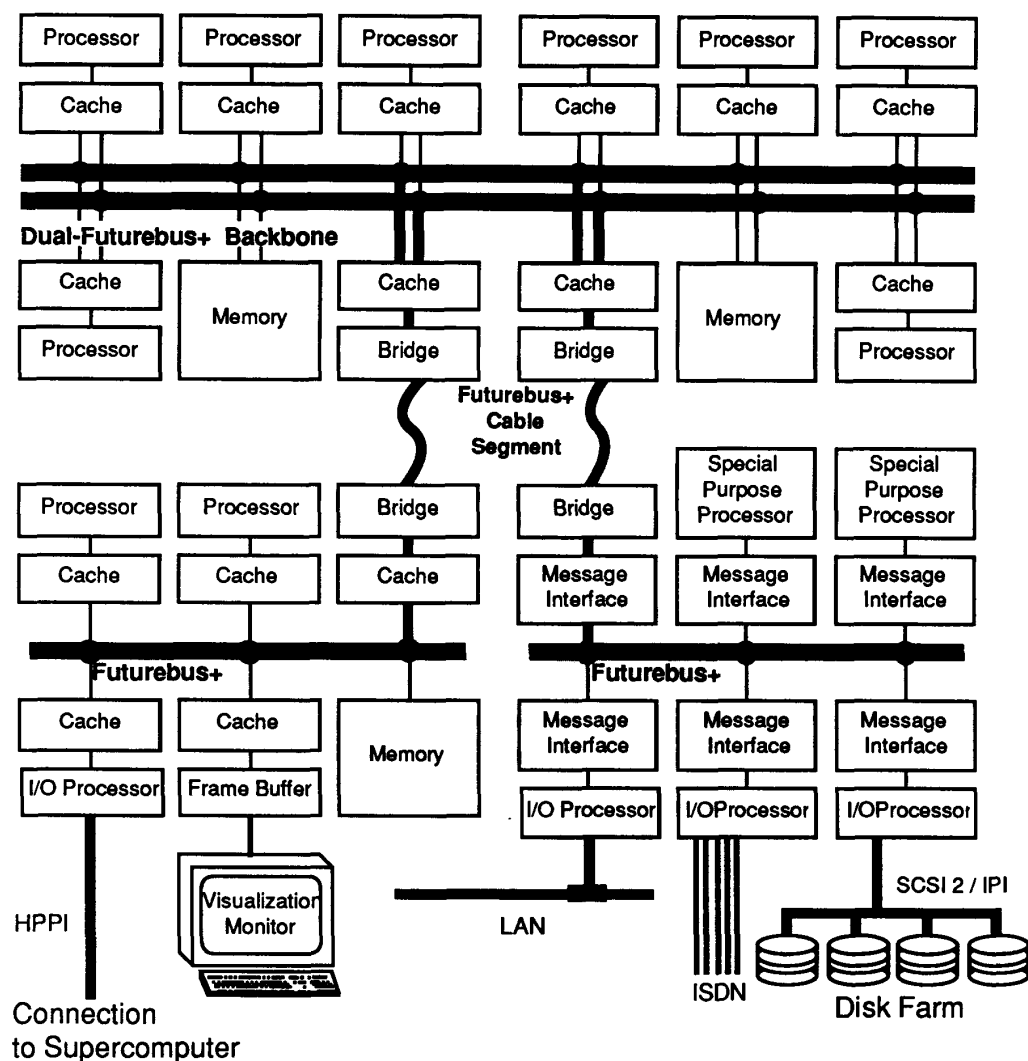


Fig 1
Interfaces in a Family of Typical Futurebus+ Systems

1.2 References. The following publications should, when applicable, be used in conjunction with this standard:

- [1] IEEE Std 896.2-1991, IEEE Standard for Futurebus+ — Physical Layer and Profile Specifications.²
- [2] P896.3, Futurebus+ Recommended Practices (Draft No. 4, Jan. 7, 1992).³
- [3] P896.4, Conformance Test Requirements for Futurebus+ (Draft No. 1.1, July 1991).

² IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.

³ Numbers with a "P" prefix are IEEE-authorized standards projects. For information about obtaining drafts, contact the IEEE Computer Society, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903.

- [4] IEEE Std 960-1989, IEEE Standard FASTBUS Modular High-Speed Data Acquisitions and Control Systems.
- [5] P1014.1, VME Bus, Futurebus+ Extended Architecture (VFE) (Draft No. 8, Jan. 7, 1992).
- [6] IEEE Std 1101.1-1991, IEEE Standard for Mechanical Core Specifications for Microcomputers Using IEC 603-2 Connectors.
- [7] P1101.3, Mechanical Core Specification for Microcomputer—2 mm Connector, Conduction-Cooled Form Factor (Draft No. 4.0, Oct. 7, 1991).
- [8] P1156.1, Environmental Specifications for Computer Modules (Draft No. 3.2, Nov. 19, 1991).
- [9] IEEE Std 1194.1-1991, IEEE Standard for Electrical Characteristics of Backplane Transceiver Logic (BTL) Interface Circuits. (ANSI)
- [10] IEEE Std 1212-1991, IEEE Standard Control and Status Register (CSR) Architecture for Microcomputer Buses.
- [11] P1212.1, Configuration Status Register for Direct Memory Access (CSR/DMA) Framework (Draft 3.32, Nov. 12, 1991).
- [12] IEEE Std 1301-1991, IEEE Standard for a Metric Equipment Practice for Microcomputers—Coordination Document.
- [13] IEEE Std 1301.1-1991, IEEE Standard for a Metric Equipment Practice for Microcomputers—Convection-Cooled with 2 mm Connectors.
- [14] P1394, High Performance Serial Bus (Draft No. 4.3, March 1992).
- [15] P1596, Scalable Coherent Interface (SCI) (Draft No. 2.0, Nov. 18, 1991).
- [16] Kernighan, Brian W. and Dennis M. Ritchie, *The C Programming Language*, 2nd Ed. Englewood Cliffs, NJ: Prentice-Hall, 1988.

2. Definitions and Structure

2.1 Special Word Usage

may. A key word indicating flexibility of choice with no implied preference.

shall. A key word indicating a mandatory requirement. Designers must implement all such mandatory requirements to ensure interoperability of IEEE 896.1 conformant products and claim conformance to this standard.

should. A key word indicating flexibility of choice with a strongly preferred implementation. The phrase *it is recommended* is used interchangeably with the key word *should*.

2.2 Definitions

activate. (A) The action of applying a set of signals to a group of bus lines. (B) The state of a group of bus lines when they carry signals.

address-only transaction. A bus transaction that does not include a data phase. The only information transferred is contained within the connection phase and, in some cases, the disconnection phase.

arbitration. The process of selecting the next bus master.

arbitrated message. A number broadcast on the arbitrated message bus lines to all modules on the bus.

assert. (A) The action of applying a logic one signal to a bus line. (B) The state of a bus line when the signal it carries represents a logic one.

*** (asterisk).** When appended to a signal's name, the suffix "*" indicates that the logic one state of the signal is such that it will override the logic zero state applied by any other module on that line.

beat. An event that begins with the transition on a synchronization line by the master, followed by the release of an acknowledge line by one or more slaves. Command and data information may be transferred from the master to one or more slaves in the first half of the beat. During the second half of the beat the slaves may transfer capability, status, and data information back to the master.

block copy. A block copy operation is characterized by a long series of read or write transactions to sequential memory locations.

bus bridge. A bus bridge is an interconnect between two or more buses that provides signal and protocol translation from one bus to another. The buses may adhere to different bus standards for mechanical, electrical, and logical operation (such as a bus bridge from Futurebus+ to VMEbus or to Multibus II).

bus line. The medium for the transmission of signals. Since Futurebus+ requires drivers with wire-OR capability, a bus line may be driven by several modules simultaneously. Therefore, the signal carried by the bus line is the combination of signals applied to that line from each module.

bus tenure. The duration of a master's control of the bus; i.e., the time during which a module has the right to initiate and execute bus transactions.

bus transaction. An event initiated with a connection phase and terminated with a disconnection phase. Data may or may not be transferred during a bus transaction. *See: transaction.*

busy. If a slave is unable to accept a bus transaction from a master, it may issue a busy status to the master of the transaction. The master must relinquish the bus and may reacquire the bus and retry the transaction after a suitable time interval.

byte. A set of eight adjacent binary digits.

byte lane. A data path formed by eight data lines and one parity line and used to carry a single byte between system modules.

cache coherence. A system of caches is said to be coherent with respect to a cache line if each cache and main memory in the coherence domain observes all modifications of that same cache line. A modification is said to be observed by a cache when any subsequent read would return the newly written value.

cache memory. A buffer memory inserted between one or more processors and the bus, used to hold currently active copies of blocks from main memory. Cache memories exploit spatial locality by what is brought into a cache. Temporal locality is exploited by the strategy employed for determining what is removed from the cache.

coherence domain. A coherence domain is a region in a multiple-cache system, inside of which, cache consistency measures are enforced. In a system that contains bus bridges, a coherence domain may or may not be extensible beyond the local bus through a bus bridge to remote buses.

coherence line. A data block for which cache consistency attributes are maintained.

compelled data transfer protocol. A technology-independent transfer mechanism in which the slave is compelled to provide a response before the master proceeds to the next transfer.

competitor. A module actively participating in the current control acquisition cycle of the arbitration process.

connected transaction. A transaction in which both the request and response are performed within the same bus transaction.

connection phase. A beat that begins with the assertion of the address synchronization line followed by the release of an address acknowledge line. It is used to broadcast the address and command information. Modules determine whether they wish to take part in the transaction based on this information.

control acquisition. The total of all bus activity associated with acquiring exclusive control of the bus.

copyback cache. A cache memory scheme with the attribute that data written from the processor is normally written to the cache rather than the main memory. Modified data in the cache is written to the main memory to avoid loss of the data when a cache line flush or replacement occurs.

CSR. Control and status register.

CSRA. Control and status register architecture. (See IEEE Std 1212-1991 [10]).

data phase. A period within a transaction used to transfer data.

deadlock. Deadlock occurs when modules are awaiting actions that can only be performed by those waiting, and those waiting cannot perform the actions.

disconnection phase. A period within a transaction used to return the bus signals to their quiescent state. In addition, this phase might be used to transfer additional information required to perform or abort the requested operation.

doublet. A set of two adjacent bytes.

entrant. A live inserted module in the process of aligning itself with the arbitration protocol.

forward progress. A module that is not blocked from performing the tasks necessary to achieve its goal is said to be making forward progress. Forward progress is guaranteed only in the absence of deadlock or starvation.

geographical address. A unique identifier assigned to each physical module slot on the bus and assumed by any module connected to that slot.

global identification. A unique identifier assigned to each physical module slot in a system. This identifier would typically include both a bus identifier and a slot identifier. IEEE Std 1212-1991 [10] specifies the format for such a global identifier.

intervening slave. The participating slave that, although not the repository of last resort of the requested data, finds it necessary to prevent the repository of last resort from providing the requested data. Having done so, the intervening slave provides the data instead.

livelock. A metastable situation in which some modules acquire and release resources in such a way that none of them makes progress.

locking. A facility whereby a module is requested to guarantee exclusive access to addressed data, blocking other modules from accessing that data. This allows indivisible operations to be performed on addressed resources.

master. A module that has acquired control of the bus through the control acquisition procedure.

master elect. A module that has won the most recent arbitration competition.

module. A collection of circuitry designed to perform specific functions that include an interface to Futurebus+.

monarch processor. The monarch processor, or simply the monarch, is the processor selected to manage the configuration and initialization of all modules on one logical bus. An emperor processor is the monarch processor selected to direct the configuration and initialization of an entire system with multiple interconnected logical buses.

octlet. A set of eight adjacent bytes.

packet data transfer protocol. A very fast but technology-dependent non-compelled transfer mechanism that uses a compelled protocol over the entire packet to provide flow control.

parallel contention arbitration. A process whereby modules assert their unique arbitration number on a parallel bus and release signals according to an algorithm such that after a period of time the winner's number appears on the bus.

participating slave. A slave involved in a transaction as a selected slave, an intervening slave, a broadcast slave, or a slave involved in multiple packet mode.

preemption. Preemption occurs when the current bus master relinquishes the bus because another module has requested it. In some systems, any module may cause preemption; in some systems, only a module with a higher priority request may cause preemption.

quadlet. A set of four adjacent bytes.

release. (A) The action of applying a logic zero signal to a bus line. (B) The state of a bus line when the signal it carries represents a logic zero.

repository of last resort. In a hierarchical memory (or cache-based) environment, a storage location that "owns" the only, or last remaining, copy of sharable data. It may be a unique source, an ultimate destination, or simply a "safe" repository of data that may not be invalidated, unless action is taken to preserve a copy of that data at some higher level in the memory (or cache) hierarchy. In a cache-only Futurebus+ system (e.g., one where even the main DRAM storage is also designed as a hardware cache), the repository of last resort begins life as the binding of an address to a physical location in one of the caches, along with the creation of the data by initialization, a copy from some higher level in the memory hierarchy, or by its arrival from some I/O device. This data may migrate around the system, and be owned by different caches at different times, provided no less than one copy of that data is maintained somewhere. A repository of last resort may end its life by an explicit instruction to "destroy" the data by migration to a higher level in the memory (or cache hierarchy), or by transfer of ownership through some I/O device to another system, storage device, or display.

request. A command, generated by a requester, to initiate an action on a responder. For a processor-to-memory read transaction, for example, the request transfers the memory address and command from the processor to memory. In the case of a split transaction, the request would be a separate bus transaction. In the case of a connected transaction, the request would be the connection phase of a bus transaction.

requester. A module that initiates a transaction by sending a request (containing address, command, and sometimes data).

responder. A module that completes a transaction by sending a response (containing the completion status and sometimes data).

response. A reply, generated by a responder, to complete a transaction initiated by a requester. For a processor-to-memory read transaction, for example, the response returns the data and status from the memory to the processor. In the case of a split transaction, the re-

sponse would be a separate bus transaction. In the case of a connected transaction, the response would be the data and disconnection phases of a bus transaction.

round robin. A bus allocation rule where, after a module acquires and uses the bus, it will not be granted use of the bus again until all other modules currently requesting the bus at the same priority level have had control of the bus.

selected slave. A slave is selected during the connection phase by the master when it recognizes its address on the bus lines.

shared memory. The address space in the system accessible to all caching modules.

signal names. Where a group of bus lines are represented by the same letters, the lines within the group are numbered; e.g., **AD0***, **AD1***, **AD2***, etc. In order to represent a group of lines or signals in more convenient form, notation such as **AD[63..0]*** is used. Also, in these examples, the notation **AD[]*** is used to refer to all of the lines within the group.

skew. The difference between the propagation delays of two or more signals on any bus line.

slave. A module that can be addressed and is capable of participating in bus transactions.

snarf. A module is said to snarf a transaction if it takes a copy of data passing by on the bus, even though it did not request it.

snoop. A module is said to snoop a transaction if it is not the master that originated the transaction or the repository of last resort for the data, but it still monitors the transaction. Cache memories snoop transactions to maintain coherence.

spatial locale. The tendency for a program to reference closely related clusters of memory addresses over short time intervals.

split transaction. A system transaction in which the request is transmitted in one bus transaction and the response is transmitted in a separate subsequent bus transaction.

starvation. A system condition that occurs when one or more modules performs no useful work for an indefinite period of time due to lack of access to the bus or other system resources.

strong sequential consistency. A system exhibits strong sequential consistency if each participating cache in the system observes all modifications to lines within itself in the same order as all other participating caches in the system. *See also: weak sequential consistency.*

Futurebus+ allows modules to issue transactions that dynamically choose between following a weakly consistent behavior model (which implies greater concurrency and hence higher performance) and strongly consistent behavior models (which may be necessary to ensure correct operation of an algorithm written by a programmer not cognizant of the concurrency possible in different parts of a system).

system transaction. A complete operation, such as a memory read or write, as viewed from the initiating unit. A system transaction can be translated into one or more bus transactions by the Futurebus+ interface to complete the operation.

temporal locality. The tendency for a program to reference the same memory locations over short time intervals.

transaction. An event initiated with a connection phase and terminated with a disconnection phase. Data may or may not be transferred during a transaction. Often used instead of the more precise phrase "bus transaction" for the sake of brevity. *See: bus transaction; system transaction.*

unselected slave. A slave that does not recognize its address on the bus lines during the connection phase of a bus transaction.

weak sequential consistency. A system exhibits weak sequential consistency if references to global synchronizing variables exhibit strong sequential consistency, and if no reference to a synchronizing variable is issued by any processor until all previous modifications to global data have been observed by all caches, and if no reference to global data is issued by any processor until all previous modifications to synchronizing variables have been observed by all caches. *See also:* **strong sequential consistency.**

2.3 Signal Conventions. Figure 2 shows an example of the signal conventions used in this document. The example is shown with open collector technology (although this specification is independent of this logic family). The signal that a specific module applies to the input of its driver is designated by a lowercase label; e.g., **ai**. The signal that a specific module applies to a bus line is designated by a lowercase label with an asterisk; e.g., **ai***. The signal that appears on a bus line as the result of the combined signals applied to it by all modules is designated by an uppercase label with an asterisk; e.g., **AI***.

When appended to a signal name, the suffix “f” (filtered) refers to the bus line signal after it has passed through a receiver and a wire-OR glitch filter (integrator). For example, **AIf** refers to the signal on the **AI*** line, after it has passed through an inverting receiver to become **AI** and a wire-OR glitch filter to become **AIf**.

In a label with an appended asterisk (e.g., **WR***), a “0” would mean the signal was released. A “1” would mean an asserted signal.

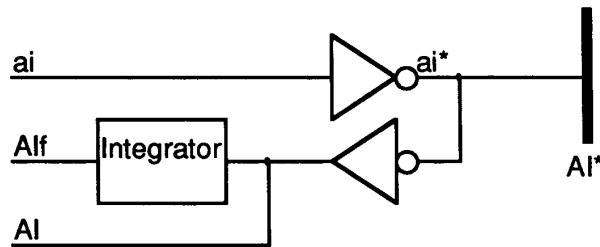


Fig 2
Signal Conventions

2.4 Document Structure. Each of Sections 3 through 9 is divided into two subsections: one containing descriptions and one containing specifications. The numbers of the description subsections begin with the section number followed by “.1”. The numbers of the specification subsections begin with the section number followed by “.2”. For example, the description subsection of Section 3 is numbered 3.1, and the specification subsection is numbered 3.2.

The description subsections are intended to help designers understand the operation of the bus and are *not intended to imply any requirements on equipment* built to this standard. The specification subsections contain *all the requirements* with which equipment designed to this standard shall comply.

The specification subsections are written using attributes (including CSR bits), states, and equations. This format, although somewhat difficult to read, allows the specification to be precise. Designers may wish to read all of the description subsections prior to reading the specification subsections.

Attributes are identified by uppercase with the underscore character separating words. For example, **COMPELLED_STATUS**, **MASTER**, and **MASTER_ELECT** are attributes. The symbols used in equations in order of precedence are as follows:

- ~ Negation
- & Logical AND
- | Logical OR

For example, the specification "A COMPETITOR shall assert **ab6*** when **cn6** & COMPETE & (**cn7** | ~**AB7***)" would mean: A module with the COMPETITOR attribute set shall assert the signal **ab6*** when the **cn6** signal is set and the COMPETE attribute is set and either **cn7** is set or **AB7*** is released.

Attributes are logical entities that do not necessarily correspond to circuitry on a module. A machine's internal state variables may or may not conform to the attributes described in this specification. Any machine whose external behavior is equivalent to the specified machine conforms to this specification.

Although this specification indicates, where appropriate, possible areas where a designer may encounter metastability (the misbehavior of digital logic at the boundary between two synchronization domains), no guarantee is given that those warnings cover all possible causes of metastability; avoiding metastability must ultimately be the responsibility of the designer, since different application architectures require different synchronization boundaries.

2.5 Futurebus+ Logo. The IEEE 896 logo shown in Fig 3 is copyrighted by the IEEE Computer Society and may be used only on products that fully conform to all mandatory requirements of an IEEE-approved 896 profile document, or adhere to either this document (IEEE Std 896.1-1991) or to IEEE Std 896.1-1987.



Fig 3
IEEE 896 Logo

2.6 Bus Line Description. Table 1 shows the list of bus lines, while 2.7.1.1–2.7.1.5 give a brief description of their function. The signals that use these lines are more fully described in later sections.

2.6.1 Information Lines. The following bus lines are used during the execution of bus transactions.

2.6.1.1 AD[63..0]* Address/Data/Lane Deselect/Request Address/Return Address. These multiplexed bidirectional address/data lines carry the address during an address transfer. The address is always driven by the master and received by one or more slaves. During the data transfer these lines carry data. Data is driven by the master during write transactions, and by the selected or intervening slave during read transactions. During the first data beat of partial transfers **AD[31..0]*** carry lane deselect signals. During cached split transactions **AD[31..0]*** carry the requesting module's address and priority and the responding module's status. During non-cached split transactions **AD[31..0]*** carry the requesting module's global identification and transaction identification.

Table 1
List of Bus Lines

Data Path Width		32-bit	64-bit	128-bit	256-bit
INFORMATION LINES					
AD[63..0]*	Address/Data	32	64	64	64
D[255..64]*	Data	-	-	64	192
BP[31.0]*	Bus Parity	4	8	16	32
TG[7..0]*	TaG	8	8	8	8
TP*	Tag Parity	1	1	1	1
CM[7..0]*	CoMmand	8	8	8	8
CP*	Command Parity	1	1	1	1
ST[7..0]*	STatus	8	8	8	8
CA[2..0]*	CApability	3	3	3	3
SYNCHRONIZATION LINES					
AS*,AK*,AI*	Address handshake	3	3	3	3
DS*,DK*,DI*	Data handshake	3	3	3	3
ET*	End of Tenure	1	1	1	1
ARBITRATED MESSAGE LINES					
AB[7..0]*	Arbitrated Message Bus	8	8	8	8
ABP*	Arbitrated Message Bus Parity	1	1	1	1
AP*,AQ*,AR*	Arbitrated Message Synch.	3	3	3	3
AC[1..0]*	Arbitrated Message Condition	2	2	2	2
RESET/BUS INITIALIZE LINE					
RE*	REset	1	1	1	1
CENTRAL ARBITRATION					
RQ[1..0]*	ReQuest	2	2	2	2
GR*	GRant	1	1	1	1
PE*	PrEmption	1	1	1	1
GA[4..0]*	Geographical Address	5	5	5	5
Total active signals		96	132	204	348

NOTE: For the number of 0V logic lines, serial bus lines, and other profile-related information, please refer to IEEE Std 896.2-1991 [1].

2.6.1.2 D[255..64]* Data Lines. These bidirectional data lines carry the data during a data transfer. Data is driven by the master during write transactions and the response portion of split read transactions. Data is driven by the selected or intervening slave during read transactions.

2.6.1.3 CM[7..0]* Command. A set of lines that carry command information from the master to one or more slaves for the current transaction.

2.6.1.4 CP* Command Parity. The line that carries the parity for the command lines.

2.6.1.5 ST[7..0]* Status. A set of lines that are activated by slaves to respond to the master's command. ST[7..0]* provide information both about the status of slaves and their disposition in regards to the requested transfer.

2.6.1.6 CA[2..0]* Capability. A set of lines that a module activates to declare its ability to accommodate major modes of bus transactions. CA[2..0]* provide the basic mechanism that allows modules with various transaction capabilities to coexist in the system; e.g., a module that can respond to packet data transfers and a module that can only execute compelled data transfers.

2.6.1.7 BP[31..0]* Bus Parity. These lines carry parity for the address/data lines. There is one parity line for every 8 address/data lines.

2.6.1.8 TG[7..0]* Tag. These bidirectional lines carry additional information related to the address/data lines. This standard does not prescribe the use of TG[7..0]*; the system designer may use them to fit the specific requirements of his application. The number of tag lines active is specified in a control and status register.

2.6.1.9 TP* Tag Parity. The line that carries the parity code for TG[7..0]*.

2.6.2 Synchronization Lines. The synchronization lines coordinate the exchange of the address, command, capability, status, and data during a transaction.

2.6.2.1 AS* Address Synchronization. A synchronization signal that is asserted during the connection phase by the master to inform the system's slaves that the address and command information are valid. AS* is released during the disconnection phase to indicate that the disconnection data and command information are valid.

2.6.2.2 AK* Address Acknowledge. An acknowledge signal released by all modules during the disconnection phase to indicate to the master that their status information is valid.

2.6.2.3 AI* Address Acknowledge Inverse. An acknowledge signal released by all modules during the connection phase to indicate to the master that their status and capability information is valid.

2.6.2.4 DS* Data Synchronization. A synchronization signal asserted or released by the master during the data transfer phase to indicate to the participating slaves that command information and write data are valid or that the master is ready to receive read data.

2.6.2.5 DK* Data Acknowledge. An acknowledge signal released by the participating slaves during the even beats or even packets of the data phase to indicate to the master that status information and read data are valid or write data has been received.

2.6.2.6 DI* Data Acknowledge Inverse. An acknowledge signal released by the participating slaves during odd beats or odd packets of the data phase to indicate to the master that status information and read data are valid or write data has been received.

2.6.2.7 ET* End of Tenure. This signal marks the end of tenure of the current master. The release of this signal notifies the master elect that it is the new master.

2.6.3 Distributed Arbitration and Arbitrated Message Lines

2.6.3.1 AB[7..0]* Arbitration Bus. A set of lines that carry a number that signifies the precedence of competitors during the arbitrated message process and the distributed arbitration process.

2.6.3.2 ABP* Arbitration Parity. ABP* carries the parity for AB[7..0]*.

2.6.3.3 AP*, AQ*, AR* Control Acquisition Synchronization. Handshake lines that perform a cyclic handshake sequence that controls the sequencing of the arbitrated message process and the distributed arbitration process.

2.6.3.4 AC[1..0]* Arbitration Condition. Lines that control the arbitrated message process and the distributed arbitration process.

2.6.4 RE* Reset/Bus Initialize. A line asserted by a module to initialize the bus interface logic of all modules to a predefined state, and provide a reset signal for use on the boards. The duration of the asserted state on **RE*** is used to decide whether the signal represents live insertion, bus initialize, or system reset.

2.6.5 Central Arbiter. In systems implementing a central arbiter, these signals are used for bus master selection as described in the following subsections.

2.6.5.1 PE* Preemption. The central arbiter asserts this line to indicate to the current master that a preemptive condition exists and the master should relinquish bus mastership.

2.6.5.2 GR* Grant. This line is asserted by a central arbiter to grant bus mastership. There is a separate grant line from the central arbiter to each module on the bus.

2.6.5.3 RQ[1..0]* Request. One or both of these lines are asserted by modules to request bus mastership. There are two separate request lines from each module on the bus to the central arbiter.

2.6.6 GA[4..0]* Geographical Address. A set of static signals from the backplane that encode the slot number of the module.

2.7 Attribute Cross Reference. Each attribute lists the sections that refer to it. A section in **bold** includes the definition of the attribute.

32_BIT_ADDRESS_CAPABLE 7.2.4.1.1

32_BIT_CAPABLE 6.2.1.3

32_BIT_DATA_CAPABLE 7.2.4.1.1

64_BIT_ADDR 6.2.1.3

64_BIT_DATA 6.2.1.3, 6.2.2.5.1, 6.2.2.5.3

64BIT_ADDRESS_CAPABLE 6.2.1.3, 7.2.4.1.1

64BIT_DATA_CAPABLE 7.2.4.1.1

128_BIT_DATA 6.2.1.3, 6.2.2.5.1, 6.2.2.5.2, 6.2.2.5.3

128BIT_DATA_CAPABLE 7.2.4.1.1

256_BIT_DATA 6.2.1.3, 6.2.2.5.1, 6.2.2.5.2, 6.2.2.5.3

256BIT_DATA_CAPABLE 7.2.4.1.1

ADDR_DECODED 6.2.1.7.2, 6.2.1.8, 6.2.2.1.3, 6.2.3.2, 8.2.2, 9.2.1

ADDR_WIDTH 6.2.1.3, 6.2.2.2.8, 6.2.2.5.1, 6.2.2.5.3

ADDRESS_ONLY 6.2.1.4, 8.2.3

ADDRESS_ONLY_LOCKED 6.2.1.4, 6.2.1.5, 6.2.1.8, 6.2.2.5.1

ADDRESS_ONLY_UNLOCKED 6.2.1.4, 6.2.1.8, 6.2.2.5.1
ADDRESS_PARITY_ERROR 6.2.1.7.2, 6.2.1.8
ADDRESS_TAG_PARITY_ERROR 6.2.1.7.2
ARBITRATED_MESSAGE_REQUEST 5.2.1, 5.2.4, 5.2.7.1.1, 5.2.9.1, 5.2.9.2, 5.2.9.6,
7.2.4.2.1
ARBITRATED_MESSAGE_SENT 5.2.1, 5.2.9.1, 5.2.9.2, 5.2.9.6
ARBITRATION_COMPARISON_ERROR 5.2.6
ARBITRATION_ERROR 5.2.4, 5.2.6, 5.2.7.2.1, 5.2.8.1, 5.2.8.3–5.2.8.5, 5.2.9.1, 5.2.9.3–
5.2.9.5
ARBITRATION_OWNER 5.2.3, 5.2.4, 5.2.7.1.3, 5.2.8.1, 5.2.8.2, 5.2.8.5, 5.2.8.6
ARBITRATION_PARITY_ERROR 5.2.6, 5.2.8.1, 5.2.8.4, 5.2.9.1, 5.2.9.4
ARBITRATION_TIMEOUT 5.2.6, 5.2.7.1.2, 5.2.7.1.3, 5.2.8.1, 5.2.8.3, 5.2.8.5, 5.2.9.1,
5.2.9.3, 5.2.9.5
ASSERT_CS 6.2.2.4.5, 8.2.2
ATTRIBUTES_QUEUED 8.2.2
BEAT_ERROR 6.2.1.7.2, 6.2.2.4.7
BEAT_ERROR_STATUS 6.2.1.7.1, 6.2.1.8
BROADCAST_CALL_ADDRESS 6.2.1.7.2, 6.2.2.4.4
BROADCAST 6.2.1.7.2, 6.2.1.8, 6.2.2.1.5, 6.2.1.8, 6.2.2.4.4, 6.2.3.4, 6.2.3.6, 6.2.3.8,
6.2.3.10, 6.2.3.12, 6.2.3.14, 9.2.1
BROADCAST_SELECT_ATTRIBUTE 9.2.1
BROADCAST_STATUS 6.2.1.7.1, 6.2.1.8, 6.2.2.1.4, 6.2.2.1.5, 6.2.2.1.6, 6.2.2.5.1–
6.2.2.5.3, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 6.2.3.22
BUS_HOLD 5.2.1–5.2.3, 6.2.1.8, 6.2.2.1.1, 7.2.1
BUS_IDLE 7.2.1
BUS_IDLE_1US 5.2.7.1.3, 6.2.2.1.3, 7.2.1, 7.2.2, 7.2.3.4
BUS_INIT 7.2.1, 7.2.3.3, 7.2.4.2.1–7.2.4.2.6
BUSY 6.2.1.7.2, 6.2.2.4.2, 8.2.2
BUSY_STATUS 5.2.3, 6.2.1.7.1, 6.2.1.8
CACHE 8.2.1, 8.2.3
CACHE_AGENT 8.2.1, 8.2.2, 8.2.3, 8.2.5
CACHE_CAPABLE 7.2.4.1.1
CACHE_CMD 6.2.1.4, 6.2.2.4.5
CACHE_OP_CMPLT 8.2.2
CACHED 8.2.2, 8.2.3–8.2.5
CANCEL 5.2.4, 5.2.7.2.2, 5.2.8.1, 5.2.8.4, 5.2.8.5, 5.2.9.1, 5.2.9.4
CAPABILITY_ERROR 6.2.1.7.2

CENTRAL_ARBITER 4.2.2.1, 4.2.2.2, 5.2.1–5.2.3, 5.2.7.1.1, 5.2.7.1.3, **7.2.4.2.1**

CENTRAL_MESSAGE_REQUEST 5.2.1, 5.2.4, 5.2.7.1.1, 5.2.9.1, 5.2.9.2, 5.2.9.6, 7.2.4.2.1

CENTRAL_MESSAGE_SENT 5.2.1, 5.2.9.1, 5.2.9.2, 5.2.9.6

CENTRALIZED_MASTER 4.2.1, 4.2.2.3, 6.2.1.1

COMPARE_SWAP 6.2.1.5, 6.2.1.8, 6.2.3.3–6.2.3.8, 7.2.4.1.1

COMPARE_SWAP_CAPABLE 7.2.4.1.1

COMPELLED 6.2.1.1, 6.2.2.3.2

COMPELLED_STATUS 6.2.1.1, 6.2.1.6, 6.2.1.7.2, 6.2.1.8, 6.2.2.2.1, 6.2.3.1, 6.2.3.2, 6.2.3.22, 6.2.3.23

COMPETE 5.2.5, 5.2.7.3.1–5.2.7.3.9, 5.2.8.2, 5.2.8.5, 5.2.9.2, 5.2.9.5

COMPETITOR 5.2.4, 5.2.5, 5.2.6, 5.2.7.3.1–5.2.7.3.9, 5.2.8.2–5.2.8.6, 5.2.9.2–5.2.9.6

CONNECTION_COMMAND_PARITY_ERROR 6.2.1.7.2, 6.2.1.8

CONNECTION_INFO_CAPTURED 6.2.1.7.1, 6.2.1.8, 6.2.2.1.1

CONNECTION_PHASE 6.2.1.1–6.2.1.4, 6.2.1.7.2, 6.2.1.8, 6.2.2.1.1, 6.2.2.1.6, 6.2.2.2.1–6.2.2.2.8, 6.2.2.3.1, 6.2.2.5.1, 6.2.2.5.3

CONNECTION_STATUS 6.2.1.1, 6.2.1.2, 6.2.1.8, 6.2.2.4.1, 6.2.3.2

COPY_BACK 6.2.1.4, 6.2.1.8, 6.2.2.5.1, 6.2.3.1, 8.2.2–8.2.4

DATA_COMMAND_PARITY_ERROR 6.2.1.7.2

DATA_INFO_CAPTURED 6.2.1.1, 6.2.1.4, 6.2.1.7.1, 6.2.1.8, 6.2.2.1.1, 6.2.2.1.4–6.2.2.1.6, 6.2.3.3, 6.2.3.5, 6.2.3.7, 6.2.3.9, 6.2.3.11, 6.2.3.13, 6.2.3.15, 6.2.3.17

DATA_LENGTH_0 6.2.1.6, 6.2.2.2.6

DATA_LENGTH_1 6.2.1.6, 6.2.2.2.7

DATA_LENGTH_2 6.2.1.6, 6.2.2.2.8

DATA_PARITY_ERROR 6.2.1.7.2

DATA_PHASE 6.2.1.1, 6.2.1.4–6.2.1.6, 6.2.1.7.2, 6.2.1.8, 6.2.2.1.4, 6.2.2.2.1–6.2.2.2.3, 6.2.2.2.5–6.2.2.2.8, 6.2.2.5.1–6.2.2.5.3, 6.2.3.2, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 6.2.3.16, 6.2.3.18

DATA_TAG_PARITY_ERROR 6.2.1.7.2

DATA_WIDTH_0 6.2.1.3, 6.2.1.8, 6.2.2.2.6, 6.2.2.5.1, 6.2.2.5.2, 6.2.2.5.3

DATA_WIDTH_1 6.2.1.3, 6.2.1.8, 6.2.2.2.7, 6.2.2.5.1, 6.2.2.5.2, 6.2.2.5.3

DEPOSITION 5.2.3, 5.2.4, 5.2.8.1, 5.2.8.4

DISC_INFO_CAPTURED 6.2.1.8, 6.2.2.1.2, 8.2.2

DISCONNECTION_COMMAND_PARITY_ERROR 6.2.1.7.2

DISCONNECTION_DATA_PARITY_ERROR 6.2.1.7.2

DISCONNECTION_END 6.2.1.7.2, 6.2.1.8, 6.2.2.3.1–6.2.2.3.3, 6.2.2.4.2–6.2.2.4.8, 6.2.3.23, 7.2.1

DISCONNECTION_PHASE 6.2.1.4–6.2.1.6, 6.2.1.8, 6.2.2.2.1, 6.2.2.2.2, 6.2.2.2.5–
6.2.2.2.8, 6.2.2.5.1, 6.2.2.5.3, 6.2.3.2, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12,
6.2.3.16, 6.2.3.18, 6.2.3.22

DISCONNECTION_TAG_PARITY_ERROR 6.2.1.7.2

DISTRIBUTED_ARBITER_REQUEST 5.2.2, 5.2.3, 5.2.4, 5.2.7.1.1, 5.2.8.1, 5.2.8.2,
5.2.8.6, 7.2.4.2.2

DISTRIBUTED_MASTER 5.2.3, 5.2.8.1, 5.2.8.6, 6.2.1.1

DISTRIBUTED_MESSAGE 5.2.2, 5.2.3, 5.2.4, 5.2.8.1, 5.2.8.4

DISTRIBUTED_MESSAGE_ENABLE 5.2.1, 5.2.2, 7.2.4.2.1

DISTRIBUTED_MESSAGE_REQUEST 5.2.2, 5.2.3, 5.2.4, 5.2.7.1.1, 5.2.8.1, 5.2.8.2,
5.2.8.6, 7.2.4.2.1

DISTRIBUTED_MESSAGE_SENT 5.2.2, 5.2.8.1, 5.2.8.2, 5.2.8.6

END_OF_DATA 6.2.1.7.1, 6.2.1.7.2, 6.2.2.4.2

END_OF_DATA_STATUS 6.2.1.7.1, 6.2.1.8

ENTRANT 7.2.1, 7.2.2.

EXCLUSIVE_MODIFIED 8.2.2, 8.2.3

EXCLUSIVE_MODIFIED_STATUS 8.2.2, 8.2.5

EXCLUSIVE_UNMODIFIED 8.2.2, 8.2.3

EXCLUSIVE_UNMODIFIED_STATUS 8.2.2

FETCH_ADD_CAPABLE 7.2.4.1.1

FETCH_ADD_BIG 6.2.1.5, 6.2.1.8, 6.2.3.3–6.2.3.6, 7.2.4.1.1

FETCH_ADD_LITTLE 6.2.1.5, 6.2.1.8, 6.2.3.3–6.2.3.6, 7.2.4.1.1

HIGH_SPEED_ENABLE 6.2.1.1, 7.2.4.2.2

ILLEGAL_OPERATION_ERROR 6.2.1.7.2

INIT 4.2.1, 4.2.2.1, 4.2.2.2, 5.2.1–5.2.5, 5.2.7.1.1–5.2.7.1.3, 6.2.1.1–6.2.1.6, 6.2.1.7.1,
6.2.1.7.2, 6.2.1.8, 6.2.2.1.1–6.2.2.1.6, 6.2.2.2.1–6.2.2.2.8, 6.2.2.3.1–6.2.2.3.3,
6.2.2.4.1–6.2.2.4.8, 6.2.2.5.1–6.2.2.5.3, 6.2.2.5.6, 7.2.1, 9.2.1

INTERVENTION 6.2.1.7.2, 6.2.2.4.6, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 6.2.3.14,
6.2.3.16, 6.2.3.18, 8.2.2

INTERVENTION_STATUS 6.2.1.7.1, 6.2.2.2.5

INVALID 8.2.2, 8.2.3

INVALID_STATUS 8.2.2, 8.2.3, 8.2.5

INVALIDATE 6.2.1.4, 6.2.1.8, 6.2.2.5.1, 8.2.2–8.2.5

INVALIDATE_COLLISION 8.2.5

KEEP_COPY 8.2.2, 8.2.3

LAST_TRANSACTION_FLAG 6.2.1.4, 8.2.2

LIVE_INSERTION 7.2.1

LIVE_WITHDRAWAL_COMMAND 7.2.1, 7.2.3.5

LOCK_CMD_0 6.2.1.5, 6.2.1.6, 6.2.2.2.1
LOCK_CMD_1 6.2.1.5, 6.2.1.6, 6.2.2.2.2
LOCK_CMD_2 6.2.1.5, 6.2.1.6, 6.2.2.2.3
LOCK_HOLD 6.2.1.2, 6.2.1.5, 6.2.2.2.1, 6.2.3.23
LOCKED 6.2.1.1, 6.2.1.5, 6.2.3.2
LOW_SPEED 6.2.1.1, 6.2.2.3.1
LOW_SPEED_STATUS 6.2.1.1
MAILBOX_INITIALIZATION 9.2.1
MAILBOX_SELECTED 6.2.1.7.2, 6.2.2.4.5, 9.2.1
MASK_SWAP_CAPABLE 7.2.4.1.1
MASK_SWAP 6.2.1.5, 6.2.3.3–6.2.3.8, 7.2.4.1.1
MASTER 5.2.3, 6.2.1.1, 6.2.1.2–6.2.1.6, 6.2.1.7.1, 6.2.1.7.2, 6.2.1.8, 6.2.2.1.1, 6.2.2.1.4,
6.2.2.2.1–6.2.2.2.9, 6.2.2.5.1–6.2.2.5.3, 6.2.2.5.6, 6.2.3.1, 6.2.3.3, 6.2.3.5, 6.2.3.7,
6.2.3.9, 6.2.3.11, 6.2.3.13, 6.2.3.15, 6.2.3.17, 6.2.3.19, 6.2.3.22, 6.2.3.24, 7.2.4.1.1,
8.2.2–8.2.5, 9.2.1
MASTER_CAPABILITY_ERROR 6.2.1.7.1, 6.2.1.8
MASTER_CAPABLE 7.2.4.1.1
MASTER_ENABLE 4.2.2.1, 4.2.2.2, 5.2.3, 7.2.4.2.2
MEMORY 8.2.1
MEMORY_AGENT 8.2.1, 8.2.2, 8.2.4, 8.2.5
MESSAGE_BUSY 6.2.1.7.2, 9.2.1
MESSAGE_CAPABLE 7.2.4.1.1
MESSAGE_CONFLICT 6.2.2.4.5, 9.2.1
MESSAGE_PASSING_SELECT_MASK 9.2.1
MODIFIED_RESPONSE 6.2.1.4, 6.2.1.7.2, 6.2.2.5.1, 6.2.3.1, 8.2.2–8.2.5
MORE 6.2.1.4, 6.2.2.2.2
MULTIPLE_PACKET_CAPABLE 7.2.4.1.1
MULTIPLE_PACKET_MODE 6.2.1.1, 6.2.1.4, 6.2.1.7.1, 6.2.1.7.2, 6.2.1.8, 6.2.3.1, 6.2.3.2,
8.2.2
MULTIPLE_PACKET_MODE_ENABLE 6.2.1.1, 7.2.4.2.1
NO_ARBITRATION_OWNER 5.2.3, 5.2.7.1.3, 5.2.8.5, 5.2.8.6
PACKET_CAPABLE 6.2.1.1, 7.2.4.1.1
PACKET_ENABLED 6.2.1.1
PACKET_LENGTH_2_CAPABLE 7.2.4.1.1
PACKET_LENGTH_2_ENABLE 6.2.1.1, 7.2.4.2.1
PACKET_LENGTH_4_CAPABLE 7.2.4.1.1
PACKET_LENGTH_4_ENABLE 6.2.1.1, 7.2.4.2.1

PACKET_LENGTH_8_CAPABLE 7.2.4.1.1
PACKET_LENGTH_8_ENABLE 6.2.1.1, 7.2.4.2.1
PACKET_LENGTH_16_CAPABLE 7.2.4.1.1
PACKET_LENGTH_16_ENABLE 6.2.1.1, 7.2.4.2.1
PACKET_LENGTH_32_CAPABLE 7.2.4.1.1
PACKET_LENGTH_32_ENABLE 6.2.1.1, 7.2.4.2.1
PACKET_LENGTH_64_CAPABLE 7.2.4.1.1
PACKET_LENGTH_64_ENABLE 6.2.1.1, 7.2.4.2.1
PACKET_NAK 6.2.1.7.2, 6.2.2.4.2
PACKET_REQUEST 6.2.1.1, 6.2.1.7.2, 6.2.2.2.1
PARITY_REPORT_ENABLE 5.2.6, 6.2.1.7.2, 7.2.4.2.2
PARTIAL 6.2.1.4, 6.2.1.7.2, 6.2.1.8, 6.2.2.5.1–6.2.2.5.3, 6.2.3.3–6.2.3.8
PARTICIPATING 6.2.1.1–6.2.1.6, 6.2.1.7.2, 6.2.1.8, 6.2.2.1.5, 6.2.2.1.6, 6.2.2.5.1–
6.2.2.5.3, 6.2.3.2, 8.2.2
PHASE_0 5.2.1–5.2.4, 5.2.5, 5.2.6, 5.2.7.1.1, 5.2.7.2.1, 5.2.7.2.2
PHASE_1 5.2.1–5.2.4, 5.2.5, 5.2.7.1.1, 5.2.7.1.3
PHASE_2 5.2.5, 5.2.6, 5.2.7.1.2, 5.2.8.3, 5.2.9.3
PHASE_3 5.2.2–5.2.4, 5.2.5, 5.2.6, 5.2.7.1.1, 5.2.7.1.2, 5.2.7.2.1
PHASE_4 5.2.4, 5.2.5, 5.2.6, 5.2.7.1.3, 5.2.8.5, 5.2.9.5
PHASE_5 5.2.1–5.2.4, 5.2.5, 5.2.7.1.2, 5.2.7.1.3, 6.2.1.5, 7.2.1
POWER_UP 4.2.2.4, 5.2.7.1.3, 6.2.2.1.3, 7.2.1, 7.2.3.1, 7.2.4.2.1–7.2.4.2.6
POWER_UP_TIME 7.2.1, 7.2.3.2
PREVENT_EU 8.2.2
QUEUED_TRANSACTION_FLAG 8.2.2, 8.2.3
READ_INVALID 6.2.1.4, 6.2.1.8, 6.2.3.1, 8.2.2–8.2.5
READ_LOCKED 6.2.1.4, 6.2.1.5, 6.2.1.8, 6.2.3.1
READ_MODIFIED 6.2.1.4, 6.2.1.8, 6.2.3.1, 8.2.2–8.2.5
READ_PARTIAL 6.2.1.4, 6.2.1.8, 6.2.3.1
READ_PARTIAL_LOCKED 6.2.1.4, 6.2.1.5, 6.2.1.5, 6.2.3.1
READ_RESPONSE 6.2.1.3, 6.2.1.4, 6.2.2.5.1, 6.2.2.5.3, 6.2.3.1
READ_SHARED 6.2.1.4, 6.2.1.8, 6.2.3.1, 8.2.2–8.2.5
READ_UNLOCKED 6.2.1.4, 6.2.1.8, 6.2.2.2.2, 6.2.3.1
READY_FOR_WITHDRAWAL 6.2.3.2, 7.2.1, 7.2.3.5
REQUEST_MAILBOX 9.2.1
REQUEST_MAILBOX_BUSY 9.2.1
REQUEST_MAILBOX_CONFLICT 9.2.1

REQUEST_MAILBOX_SELECTED 9.2.1
REQUESTER 6.2.1.7.2, 8.2.2, 8.2.4
REQUESTER_EXCLUSIVE 8.2.2, 8.2.3, 8.2.4
REQUESTER_EXCLUSIVE_STATUS 8.2.2
REQUESTER_SHARED 8.2.2, 8.2.3, 8.2.4
REQUESTER_SHARED_STATUS 8.2.2
REQUESTER_STATUS 8.2.2
REQUESTER_WAITING 8.2.2, 8.2.4
REQUESTER_WAITING_STATUS 8.2.2
REQUESTER_WRITE_INVALID 8.2.2, 8.2.4
REQUESTER_WRITE_INVALID_STATUS 8.2.2
RESET_COMPLETE 7.2.1, 7.2.2
RESET_EVENT 7.2.1
RESOURCE_LOCK 6.2.1.5, 6.2.3.2, 6.2.3.23
RESPONDER 8.2.2, 8.2.5
RESPONDER_EXCLUSIVE 8.2.2, 8.2.5
RESPONDER_EXCLUSIVE_STATUS 8.2.2
RESPONDER_INVALIDATE 8.2.2, 8.2.5
RESPONDER_INVALIDATE_STATUS 8.2.2, 8.2.5
RESPONDER_SHARED 8.2.2, 8.2.5
RESPONDER_SHARED_STATUS 8.2.2
RESPONDER_STATUS 8.2.2
RESPONDER_WRITE_INVALID 8.2.2, 8.2.5
RESPONDER_WRITE_INVALID_STATUS 8.2.2
RESPONSE_MAILBOX 9.2.1
RESPONSE_MAILBOX_BUSY 9.2.1
RESPONSE_MAILBOX_CONFLICT 9.2.1
RESPONSE_MAILBOX_SELECTED 9.2.1
ROUND_ROBIN 5.2.3, 5.2.4, 5.2.8.2, 5.2.8.6
RQ 5.2.1, 5.2.9.2
SECOND_PASS 5.2.1–5.2.3, 5.2.4, 5.2.8.1, 5.2.8.2, 5.2.8.4, 5.2.8.6, 5.2.9.2, 5.2.9.6
SECOND_PASS_REQUIRED 5.2.1–5.2.3, 5.2.4, 5.2.8.1, 5.2.8.4, 5.2.8.6, 5.2.9.1, 5.2.9.4,
5.2.9.6
SELECTED 6.2.1.5, 6.2.1.7.2, 6.2.2.4.3, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 6.2.3.14,
6.2.3.16, 6.2.3.18
SELECTED_STATUS 6.2.1.7.1

SETTLED 5.2.5, 5.2.8.1, 5.2.8.3, 5.2.9.1, 5.2.9.3
SHARED_RESPONSE 6.2.1.4, 6.2.1.7.2, 6.2.2.5.1, 6.2.3.1, 8.2.2–8.2.5
SHARED_UNMODIFIED 8.2.2, 8.2.3
SHARED_UNMODIFIED_STATUS 8.2.2, 8.2.3
SLAVE_WRITE 6.2.1.4, 6.2.1.8, 6.2.2.1.5, 6.2.2.1.6, 6.2.2.5.1–6.2.2.5.3, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 6.2.3.14, 6.2.3.16, 6.2.3.18
SNARF_DATA 6.2.2.4.4, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 8.2.2, 8.2.3
SPLIT 6.2.1.2, 6.2.2.3.3, 8.2.3, 8.2.5
SPLIT_ACCEPTOR 7.2.4.1.1, 7.2.4.2.1, 8.2.1
SPLIT_CACHED 6.2.1.2, 8.2.2
SPLIT_ENABLE 6.2.1.2, 7.2.4.2.1, 8.2.1
SPLIT_INITIATOR 6.2.1.2, 7.2.4.1.1
SPLIT_STATUS 6.2.1.2, 6.2.1.6, 6.2.1.8, 6.2.2.2.2, 6.2.2.5.1, 6.2.3.3, 6.2.3.22, 6.2.3.23, 8.2.3–8.2.5
SYS_RESET 4.2.2.4, 7.2.1, 7.2.2, 7.2.3.1–7.2.3.4, 7.2.4.2.1–7.2.4.2.6, 8.2.3–8.2.5
TAG_CAPABLE 7.2.4.1.1
TAG_ENABLE 6.2.2.5.5, 7.2.4.2.2
TRANS_CMD_0 6.2.1.4, 6.2.2.2.1
TRANS_CMD_1 6.2.1.4, 6.2.2.2.2
TRANS_CMD_2 6.2.1.4, 6.2.2.2.3
TRANS_CMD_3 6.2.1.4, 6.2.2.2.4
TRANSACTION_BEGIN 5.2.4, 5.2.8.5, 6.2.1.8, 6.2.2.1.1, 6.2.3.1
TRANSACTION_COMPLETE 5.2.7.1.3, 5.2.8.5, 6.2.1.1–6.2.1.6, 6.2.1.7.1, 6.2.1.7.2, 6.2.1.8, 6.2.2.1.1, 6.2.2.1.3, 6.2.3.22, 6.2.3.23, 8.2.2, 9.2.1
TRANSACTION_ERROR 6.2.1.7.2, 6.2.1.8, 6.2.2.4.1, 8.2.2
TRANSACTION_ERROR_STATUS 6.2.1.7.2
TRANSACTION_FLAG_STATUS 6.2.1.7.1, 6.2.2.2.2, 8.2.3
TRANSACTION_TIMEOUT 6.2.1.8, 6.2.3.1, 6.2.3.3, 6.2.3.5, 6.2.3.7, 6.2.3.9, 6.2.3.11, 6.2.3.13, 6.2.3.15, 6.2.3.17, 6.2.3.22, 6.2.3.24, 7.2.4.2.5
TRANSFER_1 6.2.1.6, 6.2.1.8
TRANSFER_2 6.2.1.6, 6.2.1.8, 7.2.4.1.1
TRANSFER_4 6.2.1.6, 6.2.1.8, 7.2.4.1.1
TRANSFER_8 6.2.1.6, 6.2.1.8, 7.2.4.1.1
TRANSFER_16 6.2.1.6, 6.2.1.8, 7.2.4.1.1
TRANSFER_32 6.2.1.6, 6.2.1.8, 7.2.4.1.1
TRANSFER_64 6.2.1.6, 6.2.1.8, 7.2.4.1.1

TRANSFER_COUNT_ZERO 6.2.1.1, **6.2.1.8**, 6.2.2.1.1, 6.2.2.1.4, 6.2.3.1, 6.2.3.3, 6.2.3.5,
6.2.3.7, 6.2.3.9, 6.2.3.11, 6.2.3.15, 6.2.3.17

TRANSFER_UNRESTRICTED **6.2.1.6**

TWO_PASS **5.2.3**, 5.2.4, 5.2.8.2

UNIQUENESS_GREATER **5.2.3**, 5.2.8.1, 5.2.8.4

WAIT **6.2.1.7.2**, 6.2.2.4.8

WAIT_CACHED 6.2.1.7.2, **8.2.2**

WAIT_STATUS 5.2.3, **6.2.1.7.1**, 6.2.1.8, 8.2.3, 8.2.4

WINNER 5.2.1–5.2.4, **5.2.5**, 5.2.7.1.2, 5.2.7.1.3, 5.2.8.1, 5.2.8.3, 5.2.8.5, 5.2.8.6, 5.2.9.1,
5.2.9.3, 5.2.9.6

WRITE_CMD **6.2.1.4**, 6.2.1.6, 6.2.1.7.2, 6.2.2.2.5, 6.2.2.5.1, 6.2.2.5.2, 6.2.2.5.3, 6.2.3.3,
6.2.3.5, 6.2.3.7, 6.2.3.9, 6.2.3.11, 6.2.3.13, 6.2.3.15, 6.2.3.17, 6.2.3.22, 6.2.3.23

WRITE_INVALID **6.2.1.4**, 6.2.2.5.1, 6.2.3.1, 8.2.2–8.2.5

WRITE_LOCKED **6.2.1.4**, 6.2.1.5, 6.2.2.5.1, 6.2.3.1

WRITE_NO_ACKNOWLEDGE **6.2.1.4**, 6.2.2.5.1, 6.2.3.1, 9.2.1, 9.2.2.1–9.2.2.14

WRITE_PARTIAL **6.2.1.4**, 6.2.2.5.1, 6.2.3.1

WRITE_PARTIAL_LOCKED **6.2.1.4**, 6.2.1.5, 6.2.2.5.1, 6.2.3.1

WRITE_RESPONSE 6.2.1.3, **6.2.1.4**, 6.2.1.8, 6.2.2.5.1, 6.2.2.5.3

WRITE_STATUS **6.2.1.4**, 6.2.1.7.2, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 6.2.3.14,
6.2.3.16, 6.2.3.18

WRITE_UNLOCKED **6.2.1.4**, 6.2.2.2.2, 6.2.2.5.1, 6.2.3.1, 9.2.1, 9.2.2.1–9.2.2.14

2.8 Implementation Mnemonics. Data sheets for products based on this standard shall use the following mnemonics to indicate capabilities:

A32	32-bit addressing implemented.
A64	64-bit addressing implemented.
CACHE	The cache coherence protocol is implemented.
CARB	A central arbiter interface is implemented.
DARB	A distributed arbiter interface is implemented.
D32	Data path width of 32 bits.
D64	Data path width of 64 bits.
D128	Data path width of 128 bits.
D256	Data path width of 256 bits.
PKTn	Able to use packet transfer mode at a clock rate of n MHz. If more than one frequency is supported, multiple PKTn mnemonics shall be listed.
MSG	Message passing protocol is implemented.
MSTR	Capable of becoming bus master.
MSLOCK	Able to service mask and swap locked commands.

CSLOCK	Able to service compare and swap locked commands.
FABLOCK	Able to service fetch and add big endian locked commands.
FALLOCK	Able to service fetch and add little endian locked commands.
SPLT	Able to service split transactions.
SPLTING	Able to service and initiate split transactions.
TAGn	Tag lines are implemented. The number of tag lines implemented is n.

A module able to become bus master, with a 64-bit data path width, 32-bit address capability, cache coherence mechanism, and able to service split transactions, would be A32/CACHE/D64/MSTR/SPLT.

3. Bus Signaling Environment

3.1 Description. This section describes and specifies the signaling environment with which Futurebus+ implementations must comply. These specifications are *independent* of the signaling technology used: Futurebus+ may be implemented with any technology (TTL, Backplane Transceiver Logic, ECL, CMOS, or GaAs), provided the condition of incident wave switching and the specified constraints on skew are met. In order to ensure compatibility between products within a set described by a profile, particular logic families may be associated with specific profiles in IEEE P896.2 [1].

3.1.1 Incident Wave Switching. Incident wave switching is a requirement for all Futurebus+ systems. The requirement is as follows: The parameters of current, voltage, and characteristic impedance, under all conditions of module loading, shall always enable the first voltage step launched by a driver down the transmission line to exceed the threshold required to trigger correct reception by the receiver, on both positive and negative transitions, and shall guarantee that subsequent reflections do not re-cross the receiver thresholds. Typically, this is related to the ratio of the current capable of being driven by a device, to the capacitance it presents to a bus line when the driver is disabled.

3.1.2 Skew. Skew is the variance in the propagation time of two or more signals that may travel over different paths, but share a common source and destination. Two signal transitions can never be guaranteed to be exactly simultaneous. Therefore, the bus protocol must always allow for a skew between them.

A critical skew in a Futurebus+ system is measured between information signals and the synchronization signal that marks their validity on the bus. The sender module's designer must ensure that transmitted information signals arrive at the connection to the bus before the validating synchronization signal occurs. This skew requirement must take into account any skews introduced by the bus transmitters themselves, and the signaling medium.

In addition, the designer of the signaling medium must guarantee that the skew introduced by capacitive loading does not delay the information signals more than the synchronization signals. A module that is logically completely passive is still electrically active with regard to bus skew. The length of the PCB trace stubs, through-plated holes, and the transceivers themselves contribute to the loading.

The designer of receiving modules must also delay the synchronization signal to account for any skew in the bus receivers as the information is received. This will ensure that the cause-and-effect relationship between data is preserved as the signals are provided to the first synchronization barrier within the receiving module.

3.2 Specification

3.2.1 Skew. The bus shall exhibit a skew such that the propagation delays of the signal lines to the left of the relational operator in relationships 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, and 3.8 are greater than or equal to the propagation delays of the signal lines to the right of the relational operator in these relationships:

$$(AS^*, DS^*) \geq (AD[]^*, D[]^*, BP[]^*, TG[]^*, TP^*, CM[]^*, CP^*) \quad (3.1)$$

$$AK^* \geq ST[]^* \quad (3.2)$$

$$AI^* \geq CA[]^*, ST[]^* \quad (3.3)$$

$$(DK^*, DI^*) \geq (AD[]^*, D[]^*, BP[]^*, TG[]^*, TP^*, ST[]^*) \quad (3.4)$$

$$(AP^*, AQ^*, AR^*) \geq (AB[]^*, AC[]^*, ABP^*) \quad (3.5)$$

$$DS^* \geq DK^* \quad (3.6)$$

$$AS^* \geq (AK^* AI^*) \quad (3.7)$$

$$GR^* \geq PE^* \quad (3.8)$$

All modules shall restrict the capacitive loading skew such that the relative loading on the signal lines exhibited by the module meets the relationships 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, and 3.8 above.

3.2.2 Glitch Filters. Glitch filters shall be provided if necessary to filter out a fundamental transmission line effect called the wire-OR glitch. It is anticipated that they will be needed for AK^* , AI^* , DK^* , DI^* , AP^* , AQ^* , AR^* , and RE^* . This specification is written assuming they are needed. The document specifying the electrical environment will specify whether they are needed for a particular signaling environment. For example, IEEE Std 896.2-1991 [1] mandates the use of glitch filters in systems implemented using the signaling environment of Bus Transceiver Logic.

4. Centralized Arbitration

4.1 Description. When a module needs to send data to or obtain data from another module, it must first gain tenure of the parallel bus. Since two or more modules may seek to gain tenure of the bus at the same time, an arbitration process is used to restrict tenure of the bus to one module at a time.

The central arbiter scheme involves a starred request/grant signal set from each requesting module to a central arbiter facility, which resolves simultaneous requests and serializes grants to the requesting modules.

This section describes a simple centralized arbitration protocol that modules may use to gain tenure of the bus. It allows high-speed operation and low request-to-grant latency. A distributed arbitration protocol is specified in Section 5.

4.1.1 Bus Lines Used for Centralized Arbitration

4.1.1.1 RQ0* Request 0. A module asserts **rq0*** to request bus mastership at one of its two priorities. Modules maintain **rq0*** asserted until the last transaction of their bus tenure.

Modules may assert **rq0*** even if they are asserting **rq1***. This possibly allows an increase of their request priority. A module maintain both **rq0*** and **rq1*** asserted until it becomes bus master and is in its last transaction on the parallel bus, or until it receives **GR*** and the internal request has been cancelled. Modules release **rq0*** and **rq1*** simultaneously.

4.1.1.2 RQ1* Request 1. A module asserts **rq1*** to request bus mastership at one of its two priorities. Modules maintain **rq1*** asserted until the last transaction of their bus tenure.

Modules may assert **rq1*** even if they are asserting **rq0***. This possibly allows an increase of their request priority. A module maintains both **rq0*** and **rq1*** asserted until it becomes bus master and is in its last transaction on the parallel bus, or until it receives **GR*** and the internal request has been cancelled. Modules release **rq0*** and **rq1*** simultaneously.

Modules should not perform low-priority transactions during a tenure that resulted from a high-priority request. Modules should release the bus and request it again at the lower priority to avoid priority inversion.

4.1.1.3 GR* Grant. The central arbiter asserts **gr*** in response to the assertion of **RQ0*** or **RQ1*** to indicate to a module that it is the master elect. The module may then begin parallel bus transactions when it detects that **ET*** is released. The central arbiter maintains the assertion of **gr*** until both **RQ0*** and **RQ1*** from the master have been released.

4.1.1.4 PE* Preempt. The central arbiter asserts **pe*** to indicate to the master that a preemptive condition exists. The central arbiter releases **pe*** when it releases **gr***. The central arbiter resolves any metastability problems on the central arbiter occurring if **PE*** is asserted near the time that the master releases its request.

Preemptive conditions would be defined by the central arbiter design. The central arbiter might assert **PE*** if other modules are requesting at equal or higher priority. A different central arbiter design could assert **PE*** if any other module was requesting the bus.

The central arbiter asserts **pe*** during power-up and system reset to indicate to monarch-capable modules that there is a central arbiter in the system. Modules that are not monarch-capable are commanded to use the central arbiter through the **CENTRAL_ARBITER** bit in the Logical Common Control register (7.1.2.2.1).

4.1.2 Centralized Arbitration Operation. The centralized arbitration competition shown in Fig 4 proceeds as follows:

- (1) Modules A and B each assert their arbitration request. For this example, Module A's request is a higher priority than Module B's request.
- (2) The central arbiter asserts **gr*** to Module A.
- (3) The current master, Module A, asserts **et*** after beginning a transaction.
- (4) During its last transaction, Module A finishes its need for bus tenure and releases its request.
- (5) Upon detecting that Module A has released its request, the central arbiter releases the grant to Module A and asserts the grant to Module B. Module B, at that point, becomes the master elect.
- (6) The master, Module A, after releasing the parallel bus signals, releases **et*** to indicate to the master elect that it is the new master.
- (7) After beginning its transaction the current master, Module B, asserts **et***.

- (8) During its last transaction, Module B finishes its need for bus tenure and releases its request.
- (9) Upon detecting that Module B has released its request, the central arbiter releases the grant to Module B. Since no requests are pending, the central arbiter issues no grant.
- (10) When the current master, Module B, has released the parallel bus signals, it releases et^* .

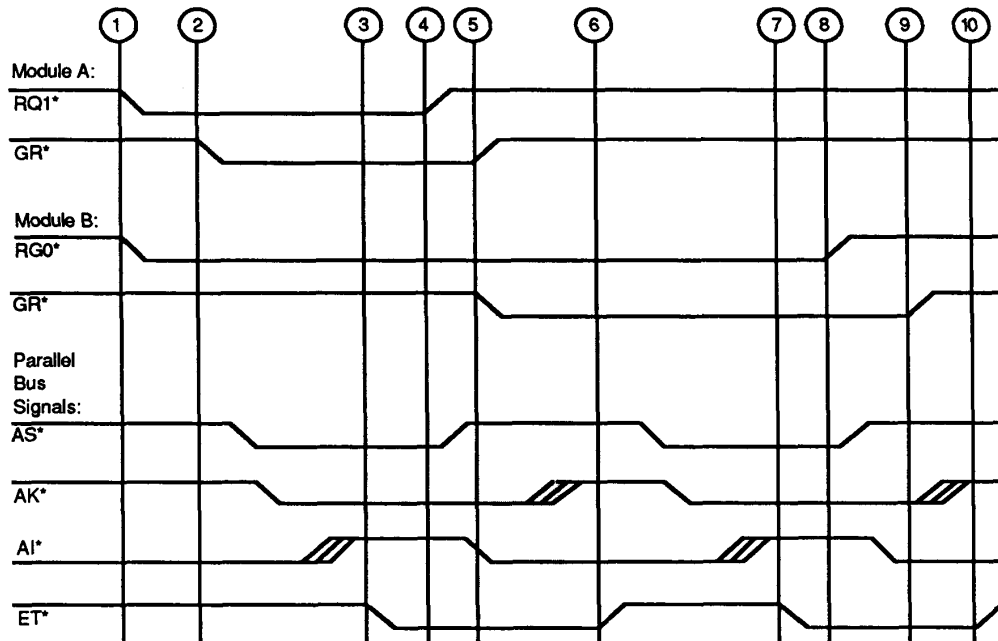


Fig 4
Two-Competitor Centralized Arbitration

4.1.3 Central Arbiter Description. The central arbiter has few mandatory requirements. Since many different types of arbiters are possible that will allow interoperability, most requirements are the responsibility and privilege of the system integrator to specify.

Any number of bus allocation schemes may be implemented. Some possibilities may be:

- A simple, single-priority, first-come-first-served arbiter.
- A two-priority arbiter with fairness allocation within each priority level.
- A more complex scheme where each request is assigned one of 256 priorities. Priorities would be changed by modules sending messages to the central arbiter using the arbitrated message facilities.

4.2 Specification

4.2.1 Bus Arbitration Attributes

CENTRALIZED_MASTER. A module shall set `CENTRALIZED_MASTER` when $\sim INIT$ & $(rq0^* \mid rq1^*)$ & GR^* & $\sim ET^*$ and maintain it set until $INIT \mid \sim rq0^* \& \sim rq1^* \& \sim et^*$.

4.2.2 Bus Arbitration Signals

4.2.2.1 RQ0*. A module may assert $rq0^*$ if $\sim INIT$ & $MASTER_ENABLE$ & $CENTRAL_ARBITER$ & $\sim GR^*$ to request mastership of the bus and shall maintain it asserted until $INIT \mid GR^*$ & et^* during the master's last transaction of this tenure $\mid GR^*$ and the module chooses not to perform any transactions during this tenure.

4.2.2.2 RQ1*. A module may assert $rq1^*$ if $\sim INIT$ & $MASTER_ENABLE$ & $CENTRAL_ARBITER$ & $\sim GR^*$ to request mastership of the bus and shall maintain it asserted until $INIT \mid GR^*$ & et^* during the master's last transaction of this tenure $\mid GR^*$ and the module chooses not to perform any transactions during this tenure.

4.2.2.3 GR*. The central arbiter asserts gr^* to indicate to a module that it is the master elect. The central arbiter shall only assert gr^* to one module at a time. The central arbiter shall release gr^* to the $CENTRALIZED_MASTER$ when it detects $\sim RQ1^*$ & $\sim RQ0^*$.

4.2.2.4 PE*. The central arbiter shall assert pe^* if it detects a preemptive condition, and shall maintain it asserted until it releases gr^* . The central arbiter shall release pe^* prior to asserting gr^* to another module. If the central arbiter needs to assert pe^* for consecutive tenures, it may choose to ignore the previous requirement and leave pe^* asserted from the assertion of one gr^* to the next.

The central arbiter shall assert pe^* if $POWER_UP \mid SYS_RESET$ and shall maintain it asserted for a minimum of 1 μs after it detects REF at logic zero.

NOTE: The detection of PE^* asserted by the central arbiter is subject to metastability. The pulse width of PE^* may be shorter than that required by a module. Modules must allow for these conditions and operate correctly.

5. Distributed Arbitration and Arbitrated Messages

5.1 Description. When a module needs to send data to or obtain data from another module, it must first gain tenure of the parallel bus. Since two or more modules may seek to gain tenure of the bus at the same time, an arbitration process is used to restrict tenure of the bus to one module at a time. Futurebus+ provides the user with the choice of two methods to select the next bus master.

The central arbitration protocol is described in Section 4. The distributed arbitration protocol is described in this section. In either case, the distributed protocol is used to broadcast arbitrated messages to one or more modules, or to the central arbiter.

Since the arbitration process operates in parallel with, and independent of, the data transfer process, the arbitration competition for control of the bus or message transfer will normally take place concurrently with transactions on the parallel data bus.

5.1.1 Arbitrated Messages—Central Arbiter. When the central arbiter is used, the arbitration process is used solely for the transmission of arbitrated messages. Two types of messages are provided, general arbitrated messages and central arbiter messages. General messages are usually directed at other modules on the bus, while central messages are used to control module request priorities in the central arbiter.

Some examples of uses for this messaging facility include sending power fail and other asynchronous broadcast interrupts, asynchronous targeted interrupts, software synchronization barriers, retry events, and bidding to supply services.

5.1.1.1 General Arbitrated Messages. Modules use a message number to resolve an arbitration competition. When two or more modules arbitrate to send a message, the winner is the one whose message number is greatest. Losing modules initiate a new arbitration competition to try again. The mechanism used is called parallel contention arbitration, a process whereby modules assert their message numbers on the arbitration bus and release signals according to an algorithm which, after a period of time, will ensure that only the winner's number remains on the arbitration bus.

General arbitrated messages are only 7 bits, so a unique number can be selected in a single competition. Each cycle consists of a competition using an 8-bit competition number, **cn[7..0]**. A module wishing to deliver a general arbitrated message uses the competition number shown in Table 2. The most significant bit is a 1, which ensures that general arbitrated messages have higher priority than central messages.

Table 2
General Arbitrated Message Fields

General Arbitrated Messages (Central Arbiter)								
Bit	cn7	cn6	cn5	cn4	cn3	cn2	cn1	cn0
Pass 1	1	AM6	AM5	AM4	AM3	AM2	AM1	AM0

5.1.1.2 Central Arbitrated Messages. Central arbitrated messages are 14 bits, and are intended primarily to send priority information to the central arbiter. Central arbiter messages require two competition cycles in order to allow up to 256 priority levels to be used in a system. New message requests cannot be initiated between the first and second pass of a two-pass message transmission. Although there can be more than one winner after the first pass, the second pass will choose a unique message.

A central arbiter message consists of three fields, as shown in Table 3. The priority field, **PR[7..0]**, selects the priority level of the module's request line. This priority level is applied to the request line corresponding to the request select field, **RQ**. If **RQ** is zero, the **RQ0*** line is selected. If **RQ** is one, the **RQ1*** line is selected. The geographical address field, **GA[4..0]**, identifies the module requesting the updated request level. In general, a module will control its own priority level, but it is possible for one module to take responsibility for another's priorities. The most significant bit of the competition number determines whether the message is in the first or last pass, and ensures that central arbitrated messages have lower priority than general ones.

Table 3
Central Arbitrated Message Fields

Central Arbitrated Message (Central Arbiter)								
Bit	cn7	cn6	cn5	cn4	cn3	cn2	cn1	cn0
Pass 1	0	PR7	PR6	PR5	PR4	PR3	PR2	PR1
Pass 2	1	PR0	RQ	GA4	GA3	GA2	GA1	GA0

5.1.2 Arbitration and Messages—Distributed Arbiter. Systems with no central arbiter use a distributed arbitration protocol. In this case, the arbitration process is used for both bus requests and arbitrated messages. The distributed arbitration requests allow modules to obtain bus tenure. The distributed arbitration messages are similar to the general arbitrated messages above, directed at other modules on the bus.

5.1.2.1 Distributed Arbitration Messages. Distributed arbitration messages are 7 bits consisting of bits **AM[6..0]**. However, they require a two-pass competition because a first pass of all 1's is needed to distinguish them from distributed arbitration requests, and to ensure that those requests do not interfere with the transmission of emergency information. A module wishing to deliver a distributed arbitration message uses the competition number shown in Table 4. During the second pass, the most significant bit is set to 1 for consistency with the general arbitrated message format.

Table 4
Distributed Arbitration Message Fields

Distributed Arbitration Messages (Distributed Arbiter)								
Bit	cn7	cn6	cn5	cn4	cn3	cn2	cn1	cn0
Pass 1	1	1	1	1	1	1	1	1
Pass 2	1	AM6	AM5	AM4	AM3	AM2	AM1	AM0

5.1.2.2 Distributed Arbitration Requests. The value of the distributed arbitration request numbers used by modules in a competition determine the number of competition passes. Some numbers require two passes of the control acquisition cycle, while others require only a single pass. Each pass of the control acquisition cycle consists of an arbitration competition using the 8-bit competition number, **cn[7..0]**. More than one module may have the highest number after the first pass, but there is only a single winner at the end of the second pass. This is guaranteed because the competition numbers used in the second pass are unique. Arbitration numbers requiring only a single-control acquisition cycle may be freely and dynamically mixed with those requiring two-control acquisition cycles.

An arbitration number consists of three fields, as shown in Table 5: the priority field, the round-robin field, and the geographical address field. A module's arbitration number maps onto one or two competition numbers. This mapping is purely a performance enhancement and does not imply that two separate classes of arbitration numbers exist. The bit **cn7** is used to indicate whether the module requires another arbitration pass, so that a single-pass arbitration number defeats a two-pass arbitration number if both occur in the same control acquisition cycle.

Table 5
Distributed Arbitration Request Fields

Distributed Arbitration Requests (Distributed Arbiter)								
Bit	cn7	cn6	cn5	cn4	cn3	cn2	cn1	cn0
Single Pass								
Pass 1	1	PR0	RR	GA4	GA3	GA2	GA1	GA0
Two Pass								
Pass 1	0	PR7	PR6	PR5	PR4	PR3	PR2	PR1
Pass 2	1	PR0	RR	GA4	GA3	GA2	GA1	GA0

5.1.2.2.1 Priority Field. The 8-bit priority field, **PR[7..0]**, selects one of 256 priority levels for the request. A module's priority class can be set by the system or the module. The highest two priority levels, where all but **PR0** are 1's, use the single pass competition cycle.

This is possible because when PR[7..1] are all 1's the requester would always win a first pass, so it can be skipped for this case. Therefore, the priorities 254 and 255 have almost twice the request-to-grant performance of the other priorities, so they should be used in systems that only use one or two priority levels.

5.1.2.2.2 Round-Robin Field. The single-bit round-robin field, **RR**, in conjunction with the geographical address field, determines the position of a module in a round-robin request queue. The round-robin protocol ensures fair and equitable allocation of bus tenure to competing modules by allowing all modules a turn at bus tenure, in a sequence defined by the magnitude of the unique field assigned to that module.

Modules using the round-robin protocol monitor arbitration competitions. The value of a module's round-robin bit is adjusted each time a transfer of tenure to another module in its priority class is observed, whether or not it competes in that arbitration competition. The round-robin bit is set each time tenure of the bus is transferred to a module in the same priority class, but with a larger unique field value. Whenever tenure of the bus is transferred to a module in its priority class with a lesser unique field value, the round-robin bit is cleared.

Once a module has acquired tenure of the bus and cleared its round-robin bit, it is defeated in any subsequent arbitration competition by all modules competing with any geographical address field value including those that are lower. When all outstanding requests from modules with a lower value of geographical address field have been satisfied, it has a chance to regain bus tenure.

If a module using the round-robin protocol relinquishes its bus tenure to a module of higher priority class before its own transactions are complete, it may retain the value "1" for its round-robin bit, ensuring that it regains tenure of the bus as soon as it becomes available to a module of its priority class.

5.1.2.2.3 Geographical Address Field. The geographical address field is a five-bit field, **GA**[4..0], which guarantees the uniqueness of the arbitration number. If a module is competing for the bus, this number must correspond to the geographical address of the module. The value "11111" is not allowed, due to a conflict with the CSR broadcast addressing scheme.

5.1.2.3 Deposing the Master Elect. In distributed arbitration, the module with the highest arbitration number at the end of the competition must wait until the master has finished performing bus transactions before it can become master. While waiting in this state, a module is referred to as the *master elect*.

There may be times when a module has urgent need of the bus after a master elect has been chosen, but before the master elect has become master. If that module has a higher arbitration number than the master elect, it may initiate a new competition to establish a new master elect. This process is referred to as *deposing the master elect*.

Deposing the master elect allows a high priority module to acquire tenure of the bus with minimum latency (although with some sacrifice to the overall performance of the system, since it forces a new arbitration competition).

5.1.2.4 Transaction Preemption. Where a priority-based scheme is in operation, a module that is the initiator of a long block transaction or a series of related transactions that could take place within a single tenure of the bus may, in the presence of a master elect from a higher priority class, choose to voluntarily suspend its current transaction. This allows a scheme of preemptive bus scheduling to be implemented. The master obtains the information it needs on the new master elect by monitoring the identity and priority of arbitration winners while it simultaneously conducts its long transaction or series of transactions on the parallel bus.

If the master observes that a module of greater priority has become master elect, it should relinquish tenure of the bus to the higher priority module as soon as is practical.

5.1.2.5 Parking. If there has been no intervening bus request, a bus master retains control of the data bus even after it has finished its transactions. This state is known as parking. If the master has a new request while it still owns the data bus, it can begin a new data bus transaction immediately, without initiating an arbitration cycle. This has the potential to greatly reduce average bus latency in some types of systems.

5.1.3 Bus Lines

5.1.3.1 AP*, AQ*, AR* Control Acquisition Synchronization. These lines keep modules synchronized throughout the control acquisition process. The lines operate in a six-state sequence referred to as phase 0 through phase 5.

5.1.3.2 AC[1..0]* Arbitration Condition. During phase 3 through phase 5, modules assert **AC0*** to signal an arbitration error.

In distributed arbitration, during phase 3 through phase 5, modules assert **AC1*** to inhibit the transfer of tenure at the completion of a control acquisition cycle. In a central arbiter system, modules assert **AC1*** to inhibit the transmission of a partial central arbitrated message which occurs during the first pass of the two-pass competition. **AC1*** is always asserted along with **AC0*** if an error occurs. If **AC1*** is asserted alone during phase 3 through phase 5, transfer of tenure is inhibited for reasons other than the detection of errors.

5.1.3.3 AB[7..0]* Arbitration Bus. Competing modules apply their competition numbers, **cn[]**, to **ab[]*** during phase 1. However, if any signal **cnx** is a logic zero, and the signal on the corresponding bus line **ABx*** is a logic one, then for as long as this condition persists, all bits, **ab[x-1..0]***, of lower significance are released. This continues through phase 3.

5.1.3.4 ABP* Arbitration Parity. If the number of ones in a competitors competition number is even, the competitor must assert **abp*** during phase 1. However, if any of the bits, **ab[]***, are released as described in 5.1.3.3, **abp*** shall also be released. This continues through phase 3.

5.1.4 Arbitration Competition Logic. As shown in the sample circuits in Figs 5(a) and 5(b), if a module's competition number has a bit position of 0 and another module has a 1 in that position, the module with the 0 releases all **ab[]*** lines of lower significance. Eventually the **AB[]*** lines carry the number of the module with the highest arbitration number.

In some cases, modules release **ab[]*** lines only to assert them at a later time. Figure 6 illustrates an arbitration competition where three modules simultaneously begin competition using the numbers "00001010", "00001001", and "00000100", respectively. **AB[7..4]*** are not shown on the figure since they remain released throughout the sequence.

The arbitration competition in Fig 6 proceeds as follows:

- (1) The three modules assert their competition numbers on the arbitration bus.
- (2) After a delay due to bus propagation, the bus lines become the wire-OR of the individual module lines causing **AB[3..0]*** to assert.
- (3) Module one, upon detecting **AB2*** asserted, releases **ab1***. Module two, upon detecting **AB2*** and **AB1*** asserted, releases **ab0***. Module three, upon detecting **AB3*** asserted, releases **ab2***.
- (4) The bus reflects the release of the bus lines.
- (5) Module 1, upon detecting **AB2*** released, asserts **ab1***. Module 2, upon detecting **AB2*** and **AB1*** released, asserts **ab0***.

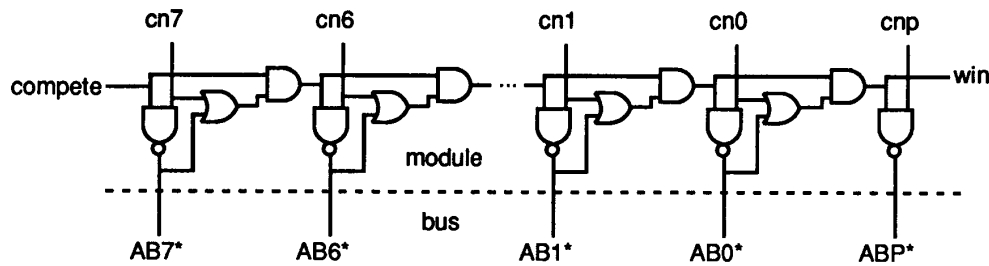


Fig 5 (a)
Serial Implementation of Parallel Contention Logic

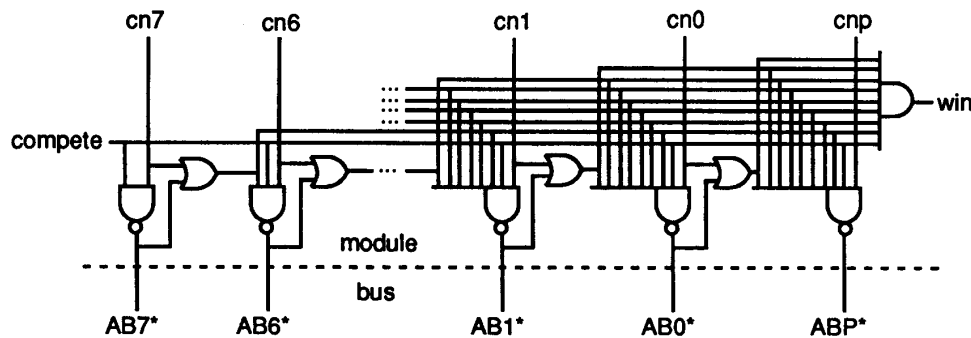


Fig 5 (b)
Parallel Implementation of Parallel Contention Logic

NOTE: These circuits should not be considered a required implementation.

- (6) The bus reflects the assertion of the bus lines.
- (7) Module 2 detects **AB1*** asserted and releases **AB0***.
- (8) The bus reflects the release of the bus line.

As shown in Fig 6, the winning module (Module 1) releases **ab1*** and then later reasserts it. Module 1 also must wait for Module 2 to release **ab0*** before it knows it is the winner. The time required for the winning arbitration number to settle on the bus is referred to as the *arbitration competition settling time*.

5.1.5 Arbitration Competition Settling Time. The arbitration settling time is a function of the length of the backplane, the module's competition number, and the worst-case transceiver and logic delays of both the internal arbitration logic of the module and that of its competitors. The theoretical calculation of the worst-case arbitration settling time can be extremely complex, and is beyond the scope of this standard. The worst possible settling time for any competition number, however, can be given by the following equation:

$$t_a = 10 * t_{pd} + 5 * t_{ext} + 4 * t_{int} + t_{win}$$

The worst-case delay can be considerably lower for most competition numbers, so if the performance of the distributed arbiter is of concern, it is worth trying to optimize this equation for each module's requests and messages.

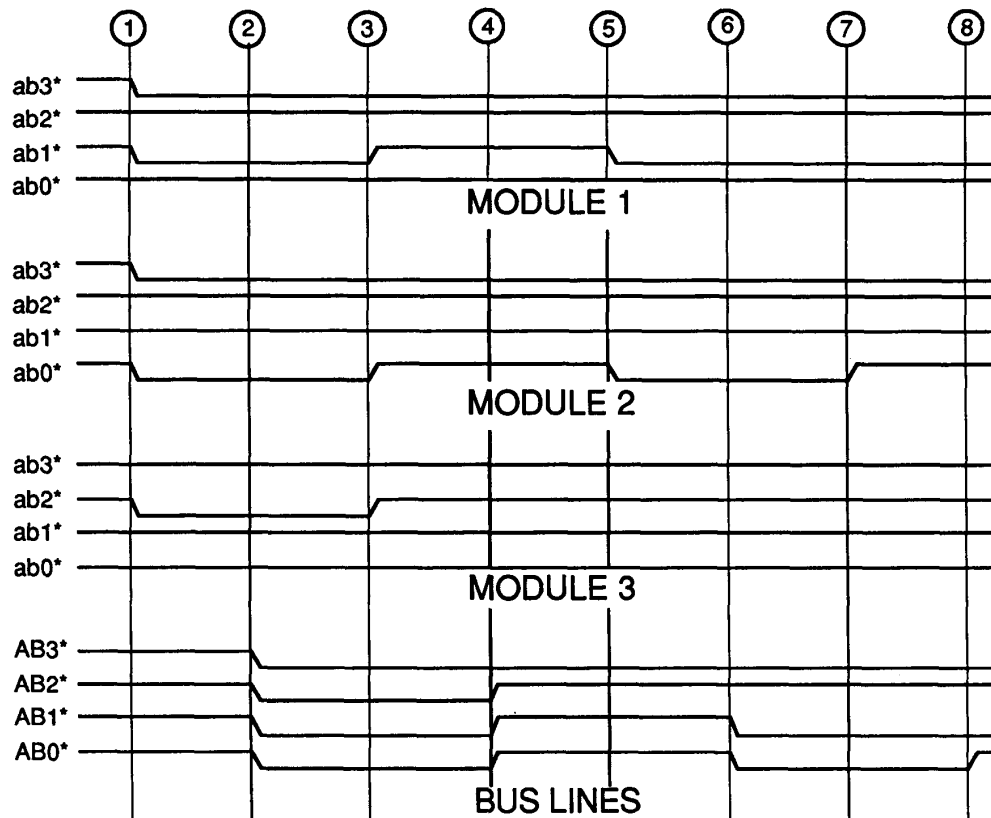


Fig 6
Arbitration Competition Example

The symbol t_{pd} is used to represent the maximum end-to-end signal propagation delay along the transmission lines which constitute the logical bus. The value for t_{pd} is in the Bus Propagation Delay register described in 7.1, and is a function of the maximum length of the transmission line in the application system.

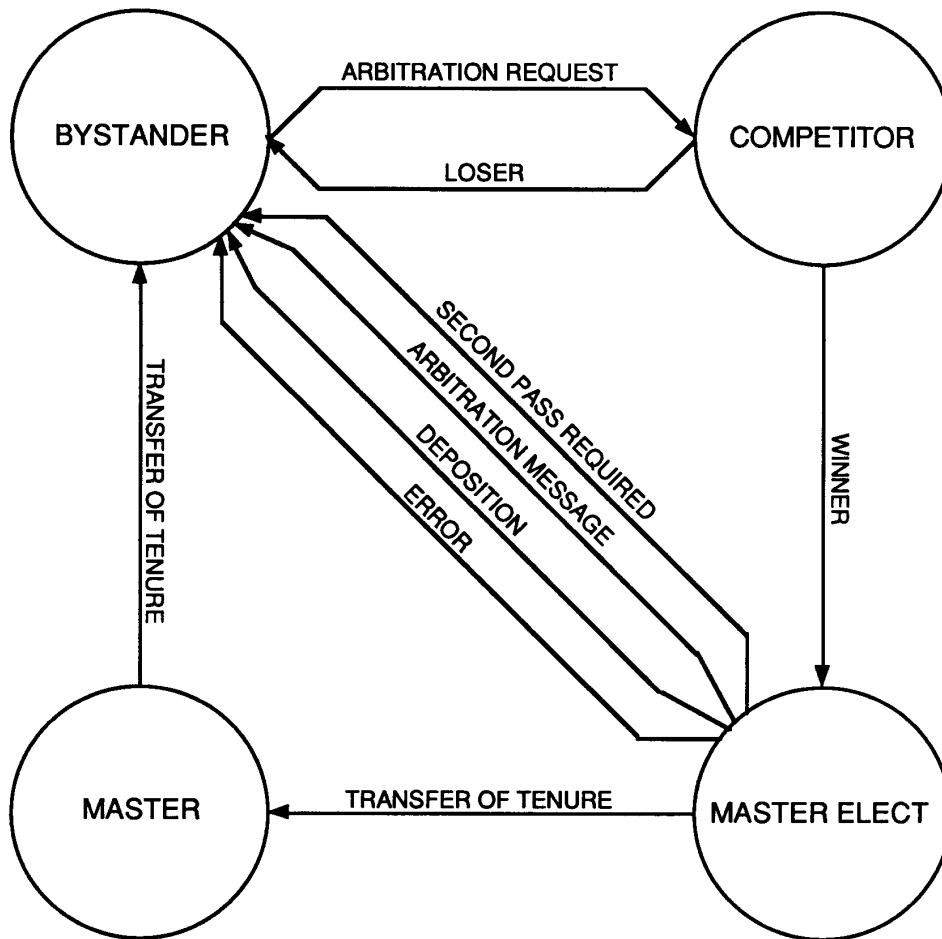
The symbol t_{int} is used to represent the delay between a change on any arbitration line, ABx^* , on the input to a module and the corresponding change on that module's adjacent arbitration output, $ab[x-1]^*$, or abp^* , as its competition number is applied to or withdrawn from those bits.

The symbol t_{ext} is used to represent the delay between a change on any arbitration line, ABx^* , on the input of any other competing module and the corresponding changes on that module's arbitration outputs, $ab[x-1]^*$, or abp^* , as its competition number is applied to or withdrawn from those bits.

The symbol t_{win} is used to represent the maximum delay between the time when the arbitration bus, $AB[7..0]^*$, becomes stable and equivalent to $cn[7..0]$ on the input to a module and the indication within that module that it is the winner.

5.1.6 Arbitration States. The potential states that may be entered by a module involved in the arbitration process is illustrated in Fig 7, along with the events that may result in a status change from each state. The figure shows examples of legal state transitions that may occur as a consequence of a control acquisition cycle.

Transitions from one state to another occur only when certain conditions are met.



NOTE: This figure is meant to provide a high level overview, not a precise state diagram of the control acquisition process.

Fig 7
Control Acquisition States

5.1.7 Arbitration Phases. Six phases of the three synchronization signals, AP^* , AQ^* , and AR^* , are used during the control acquisition cycle. Normally, each phase is completed as quickly as propagation delays and metastability resolution times allow. If a two-pass arbitration competition is required, the entire control acquisition cycle is repeated.

The signals AP^* , AQ^* , and AR^* are synchronizing signals representing the wire-OR of the module signals ap^* , aq^* , and ar^* , respectively. For any of them to be released on the bus, each module must individually release its signal. The release of a signal is a global synchronization point for each module. No module may proceed to the next phase of the synchronization arbitration cycle until all modules have released the appropriate arbitration line. Similarly, the assertion of one of these signals indicates that a new phase is starting, and all modules, even if they are not participating in the current arbitration competition, must remain synchronized by asserting their own signal in response.

Figure 8 shows the sequence of phases and line state changes.