


ANSI/VITA 5.1-1999

Approved as an American National Standard by 

American National Standard for RACEway Interlink

Secretariat
VMEbus International Trade Association

Approved August 31, 1999
American National Standards Institute, Inc.



VMEbus INTERNATIONAL TRADE ASSOCIATION

7825 E. Gelding Drive, Suite 104, Scottsdale, AZ 85260 USA

PH: 480-951-8866, FAX: 480-951-0720

E-mail: info@vita.com, URL: <http://www.vita.com>

ANSI/VITA 5.1-1999

American National Standard
for RACEway Interlink

Secretariat
VMEbus International Trade Association

Approved August 31, 1999
American National Standards Institute, Inc.

Abstract

This standard provides a specification of the data link protocol and physical interface of a high performance extension to the VMEbus standard. This extension consists of high bandwidth, low latency interconnects across a VMEbus computer chassis backplane using the P2 connector. Bi-directional connectivity between boards in a VMEbus chassis is achieved through the use of a network of crossbar switches with point-to-point interconnects.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether they have approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

Published by

**VMEbus International Trade Association
7825 E. Gelding Drive, Suite 104, Scottsdale, AZ 85260**

Copyright © 1999
by VITA, the VMEbus International Trade Association
All rights reserved.

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

Printed in the United States of America

ISBN 1-885731-22-1

Table of Contents

Table of Contents	i
List of Tables	iv
List of Figures	v
Foreword	vii
Introduction	1
1 Scope and purpose	5
1.1 Scope	5
1.2 Purpose	5
1.3 Normative references	5
2 Definitions	7
2.1 Key words	7
2.2 General	7
2.3 Protocol	7
2.4 Physical	9
3 Overview	11
3.1 Characteristics	11
3.1.1 VMEbus compatibility.....	11
3.1.2 Interconnectivity	11
3.1.3 Transaction size.....	11
3.1.4 Resource Contention.....	11
3.1.5 Physical.....	11
3.2 Fundamental concepts	11
3.2.1 Crossbar network.....	11
3.2.2 Routing.....	11
3.2.3 Handshaking	13
3.2.4 Configuration.....	14
3.2.5 Latency.....	14
3.2.6 Priorities and Kill requests.....	14
3.2.7 Adaptive routing	14
3.2.8 Locked operations.....	14
3.2.9 Split read transfers.....	14
3.2.10 Split write transfers.....	15

4	A Data Link Layer specification.....	17
4.1	Transaction format	17
4.2	Extended route/address format	18
4.3	Routing code/Broadcast Mode	19
4.4	Broadcast acceptance code	20
4.5	Routing priority code	21
4.6	Transfer width and alignment specification	23
4.7	Address	24
4.8	Reserved field	24
4.9	Reads	25
4.10	Locked transfers	25
5	Physical layer specification.....	27
5.1	Interface	27
5.2	Connections	27
5.3	P2 Pinout	28
5.4	Master interface signals to the RACEway crossbar network	29
5.5	Slave interface signals to the RACEway crossbar network	30
5.6	Signal characteristics	31
5.7	Signal notation	31
5.8	Master/slave relationship	32
5.9	Timing	33
5.9.1	Clocks and control phases.....	33
5.9.2	Route phase: REQO, REQI, ROUTE, SHIFTED ROUTE.....	35
5.9.3	Address phase: CHANGE TO ADDRESS.....	35
5.9.4	Data phase: DSEN.....	37
5.9.5	Data phase: ASTROBE and DSTROBE.....	39
5.9.6	Write transaction.....	41
5.9.7	Broadcast write transactions.....	43
5.9.8	Data phase during Reads: ASTROBE, DSTROBE and READ READY	44
5.9.9	Data phase during Reads: RDCON* and ERR.....	45
5.9.10	Read transaction.....	46
5.9.11	Read transaction.....	47
5.9.12	Read-Modify-Write transaction	48
5.9.13	Read-Modify-Write transaction	49
5.9.14	Route phase: Kill Request	50
5.9.15	Split Read	51

5.9.16 Split Write.....	52
5.9.17 Route Phase: REQO, REQI - Extended Route Option.....	53
6 Mechanical specification.....	55

ANNEX

A Raceway Interlink Signal Characteristics (Informative)	57
A.1 Signal timing notation	58

List of Tables

Table 1 -Transaction format.....	17
Table 2 -Route word format	17
Table 3 -Address word format.....	17
Table 4 -Route/address space trade-offs.....	17
Table 5 -Extended transaction format.....	18
Table 6 -Route word format	18
Table 7 -Optional additional extended route word format.....	18
Table 8 -Address word format.....	18
Table 9 -Possible extended route/address space trade-offs.....	18
Table 10 -Route word format - Route code.....	19
Table 11 -Routing codes.....	19
Table 12 -Route word format - Broadcast accept code.....	20
Table 13 -Route word format - Split mode.....	20
Table 14 -Route word format - Routing priority code.....	21
Table 15 -Transaction priority code.....	21
Table 16 -Address word format - Transfer width/alignment.....	23
Table 17 -Data width and alignment code.....	23
Table 18 -Route word format - High order address bits.....	24
Table 19 -Address word format - Address bits.....	24
Table 20 -Address word format - Reserved field.....	24
Table 21 -Address word format - Read Flag.....	25
Table 22 -Address word format - Lock Flag.....	25
Table 23 -P2 pin assignments.....	28
Table 24 -Control signals sent by the master.....	29
Table 25 -Control signals received by the master.....	29
Table 26 -Control signals sent by the slave.....	30
Table 27 -Control signals received by the slave	30

List of Figures

Figure 1 - Module backplane location	1
Figure 2 - Four slot RACEway Interlink.....	2
Figure 3 - Slots 1-3, 2-4, and external ports simultaneously transferring data.....	2
Figure 4 - Possible 8-slot RACEway Interlink topology (Clos network).....	3
Figure 5 - Possible 8-slot RACEway Interlink topology (Torus (combination mesh and ring)).....	3
Figure 6 - Possible 16-slot RACEway Interlink topology (Fat tree).....	3
Figure 7 - Possible 20-slot RACEway Interlink topology (Fat tree).....	4
Figure 8 - Routing example.....	12
Figure 9 - P2 Interface Connections.....	27
Figure 10 - Timing notation	31
Figure 11 - Master/slave relationship	32
Figure 12 - Clocks and control phases.....	34
Figure 13 - REQO, Route, Shifted Route (as seen at master side initiator).....	36
Figure 14 - REQL, Route, Shifted Route (as seen at slave responder).....	36
Figure 15 - Change To Address (as seen at slave side).....	36
Figure 16 - DSEN (as seen at slave side).....	38
Figure 17 - ASTROBE and DSTROBE (as seen at master side).....	40
Figure 18 - Write transaction.....	42
Figure 19 - ASTROBE, DSTROBE And READ READY (as seen at slave side).....	44
Figure 20 - RDCON* and ERR (as seen at slave side).....	45
Figure 21 - Read transaction (as seen at slave side).....	47
Figure 22 - Read-Modify-Write transaction (as seen at slave side)	49
Figure 23 - KILL (as seen at master).....	50
Figure 24 - Split Read (as seen at slave side)	51
Figure 25 - Split Write (as seen at slave side).....	52
Figure 26 - Route, Extended Route, Shifted Route, Shifted Extended Route (as seen at master side).....	53

Foreword

(This foreword is not part of the standard.)

Technology improvements in processing, data acquisition, and display capabilities create opportunities for new high performance applications. In order to provide platforms for these new applications, the VME community requires migration paths offering higher bandwidth while preserving investments in existing chassis and boards. RACEway Interlink is a VMEbus enhancement that can deliver up to 3.2 Gbytes/sec of scalable bandwidth over rows A, C, D and Z of the P2 connector in a standard VMEbus chassis.

In addition to increased bandwidth, RACEway Interlink offers: 1) low latency, deterministic transactions, 2) concurrent point-to-point transactions for multiple simultaneous transfers, and 3) scalable bandwidth so total bandwidth increases as more slots are added. Many new applications, especially those with multiple processors and multiple real-time I/O interconnects, require these capabilities.

RACEway Interlink Highlights

In addition to being backward-compatible with VMEbus backplanes, RACEway Interlink has the following characteristics:

- Full interconnectivity: configurations are scalable from two to the size of the VMEbus backplane, with any slot capable of reads or writes to any other slot.
- High bandwidth: a single RACEway connection is capable of 160Mbytes/s peak and 150Mbytes/s sustained.
- Scalable bandwidth: a four-slot configuration supports up to four simultaneous full-speed transfers (640Mbytes/s); twenty slots can support up to twenty (3200Mbytes/s).
- Low latency: processors and devices in a twenty slot configuration experience a write latency of approximately five hundred nanoseconds.
- Deterministic: firmware or software can control contention for the predictable delivery of real-time data (e.g. A/D interfaces).
- Broadcast and multicast are supported.
- Block or non-block: while optimized for block transfers, non-block transfers are also supported.
- Low-overhead protocol: the RACEway Interlink protocol does not require a control microprocessor.
- No VMEbus interaction: operation of RACEway Interlink does not require any VMEbus signals, resources, or additional slots. VMEbus operations can take place simultaneously with RACEway Interlink transfers.
- Board compatibility: existing VME boards, which do not use rows A and C of the P2 connector, can coexist with RACEway Interlink-compatible boards.
- Protocol compatibility: RACEway Interlink is able to interface to existing P2 protocols, for example VSB.

RACEway Interlink uses pipelined circuit switching rather than a bus architecture. The master of a transaction establishes a route through one or more crossbar switches to the slave. The slave acknowledges that it is connected, and the master begins the transaction. This is like wormhole routing, except the master waits for acknowledgment from the slave before starting the transaction. Read and locked read-modify-write transactions (as well as writes) are possible with pipelined circuit switching. This enables a system to use RACEway Interlink with either a shared memory programming model or a message passing model.

Acknowledgments

RACEway Interlink was originally conceived by Robert Frisch, Barry Isenstein, Jim Kulp, and Robert Siegel at Mercury Computer Systems. A team led by Robert Frisch and John Nitzsche, consisting of Thomas Baillio, Robert Blau, Marc Burke, John Freeburn, Earl Gooch, Neil Johnson, Christopher Kolb, William McCaffrey, and Tom Morin, developed RACEway at Mercury Computer Systems. Robert Blau wrote the first version of this spec with technical and editorial assistance from Thomas Baillio, Christopher Kolb, James LaLone, and Gregory Rocco. Dick Somes at Digital Equipment Corporation provided many valuable comments during the review period.

This standard was processed and approved for submittal to ANSI by the VITA Standards Organization Technical Committee. Committee approval of this standard does not necessarily imply that all committee members voted for its approval. At the time it approved this standard, the VSO Technical Committee had the following members:

Robert Downing, Chair
John Rynearson, Secretariat

<i>Organization Represented</i>	<i>Name of Representative</i>
AMP Incorporated	Elwood Parsons Mike Galloway
Amphenol Corp	Ron Hudson
Ariel Corp.	Bob Dillon
Berg Electronics, Inc.	Jim Koser
CETIA	Steve Paavola
CSPI	Paul Vena
Cypress Semiconductor	Dave Horton
DIALOGIC CORPORATION	Lou Francz
DY 4 Systems, Inc.	Clarence Peckham
Dawn VME Products, Inc.	Tad Kubic
Digital Equipment Corporation	Dick Somes
EPT	Bruce Sarnowitz Klaus Heimann
ERNI Components	Michael Savage
Electronic Solutions	Frank Hom
Force Computers	Wayne Fischer
Force Computers GmbH	Istvan Vadasz Juergen Baumann
Harting Elektronik Inc.	David Robak
Heurikon Corporation	Dennis Terry
Hughes Aircraft Company	Douglas Endo
Hybricon Corporation	Michael Munroe
IBM Microelectronics	Stephen Howland
MITRE Corp.	Robert Dougherty
Mercury Computer Systems, Inc.	Bob Blau
Micro Memory, Inc.	Mike Hasenfratz
Motorola, Computer Group	Jerry Gipper Chau Pham
NAWC	Don Lane Mary Mosier
National Semiconductor	Rich Hovey
Natural Microsystems	Michael Motta
Naval Surface Warfare Center	Gerry Thomas Jerry Braun Larry Thompson
Newbridge Microsystems	David Lisk
Northern Telecom Canada Ltd.	Paul Ramsden
OTI	Kim Clohessy
PEP Modular Computers	Ray Alderman
Parallelogram Magazine	Craig Lund
Primary Rate Inc.	Bob Grochmal
Rittal Corp	Eike Waltz
Sky Computers	Doug Williams
SGS-Thomson Microelectronics	Steve Christian
Schroff, Inc.	Mike Thompson
Univ. Of ILL.	Robert Downing
VERO Electronics Ltd.	Martin Blake
VITA	John Rynearson
Vector Electronic Co.	Walt Waldron
Vista Controls	Mike MacPherson Gorky Chin
Winchester Electronics	Scott Annis

The RACEway Task Group, which worked on this standard had the following members:

Tony Lavelly, Draft Editor

Ray Alderman
Sam Han
Michael Hasenfratz
David Horton
Robert McKee
Michael Munroe
Richard O'Connor
Chau Pham

Consensus for this standard was achieved by use of the Canvass Method.

The following organizations, recognized as having an interest in the standardization of high performance extensions to the VMEbus were contacted prior to the approval of this revision of the standard. Inclusion in this list does not necessarily imply that the organization concurred with the submittal of the proposed standard to ANSI.

Adaptive Optics Associates
Adaptive Technology, Inc.
AnTeL, Inc.
Bolt Beranek & Newman Inc.
Coastal Systems Station
Cypress Semiconductor
Digital Equipment Corp.
E-Systems
Electrospace Systems Inc.
ERIM
ESL
Heurikon Corporation
HRB Systems
Hughes Aircraft Company
Innovative Configuration Inc.
Iotek Inc.
Lawrence Livermore Nat'l. Lab
Lockheed Fort Worth Company
Lockheed Missles & Space
Lockheed Sanders
Loral Electronic Systems
Martin Marietta Laboratories
Mercury Computer Systems
Micro Memory, Inc.
MIT Lincoln Laboratory
MITRE Corporation
Monowave Corporation
PEP Modular Computers

Differences between ANSI/VITA 5-1994 and ANSI/VITA 5.1-1999

The main intent of the changes is to allow for the use of the new five row connector as specified in ANSI/VITA 1.1-1997, VME64 Extensions with RACEway. In addition, we have taken the opportunity to improve technical clarity, fix typos and make grammatical changes throughout.

Some specific areas of change which designers may want to read are indicated below.

Section 3.1.1 Add rows D and Z of P2.

Section 3.1.2 Add Connectivity comments.

Section 3.2.10 Added Split Write.

Section 4.4 Added Single Modedescription.

Section 5.3 Added rows D and Z.

Section 5.4, Section 5.5, Corrected descriptions.

Section 5.7 Added explanatory paragraph about timing information.

Section 5.9.2 through Section 5.9.6 Changed wording.

Section 5.9.3 sixth paragraph changed to read "fifth clock edge"

Section 5.9.16 Added Split Write.

Section 5.9.17 Renumbered.

Section 6 Added Five Row connector

Annex A Added V_{IH} and V_{IL} specifications for REQ1.

Figures 16, 18, 19, 21, 22, 24 and 25: Corrected the relationship shown between Change to Address and Address.

RACEway Interlink - Data Link and Physical Layer Specification

Introduction

The development of RACEway Interlink was motivated by the desire to use point-to-point interconnects to provide high bandwidth communication while still being compatible with off-the-shelf VMEbus chassis. RACEway Interlink is targeted at multiprocessor and high-speed I/O applications. I/O devices such as A/D, D/A, frame stores, graphic displays, LAN/WAN connections, and storage devices need the high bandwidth, low latency access to all processors and other I/O devices provided by RACEway Interlink.

RACEway Interlink is implemented using crossbar switches as building blocks. A RACEway crossbar connects gluelessly (without additional circuitry) to other RACEway crossbars to form an interconnection network. The first implementation of a RACEway crossbar is a single six-port crossbar chip that operates synchronously at 40MHz. Each port of the RACEway crossbar consists of 32 bits for multiplexed address and data, and eight control pins. A single RACEway crossbar is non-blocking and supports up to three simultaneous transfers. Since each crossbar represents 480Mbytes/s of aggregate bandwidth, application bandwidth scales up quickly as more crossbars are added.

The RACEway Interlink module is a pluggable active backplane on the P2 connectors (see Figure 1). Systems physically scale by either plugging boards into open RACEway slots or adding incremental removable backplane modules for additional RACEway slots.

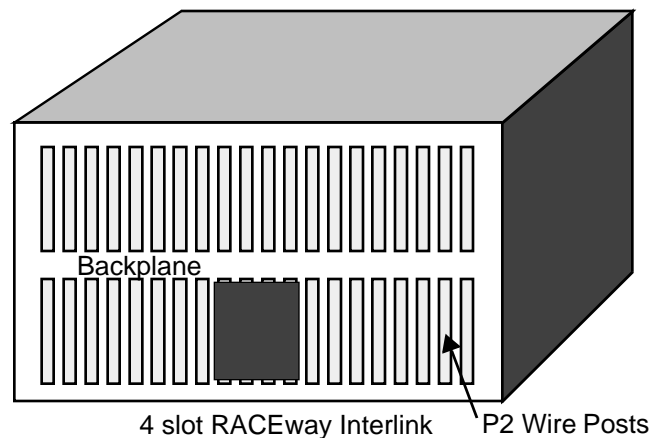


Figure 1 - Module backplane location

A four slot RACEway Interlink is illustrated in Figure 2. Each slot interfaces to the RACEway Crossbar via rows A and C of the P2 connector. Each connection on a RACEway Interlink is an independent point-to-point connection, so three paths can transfer data simultaneously (see Figure 3.)

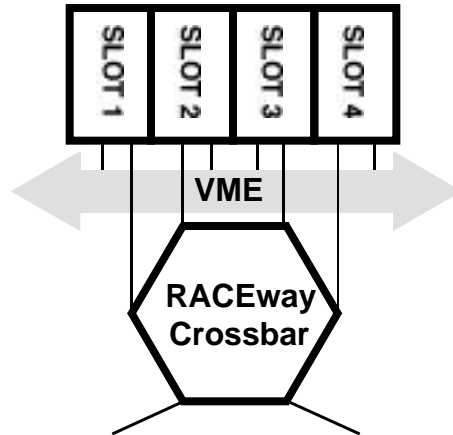


Figure 2 - Four slot RACEway Interlink

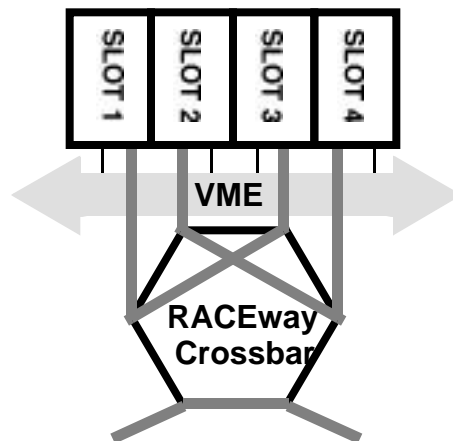


Figure 3 - Slots 1-3, 2-4, and external ports simultaneously transferring data

RACEway crossbars may be put together in many different topologies to provide different levels of connectivity and performance. Possible eight, sixteen and twenty slot topologies are shown in Figures 5 through 6. This standard does not specify a single topology for RACEway Interlink modules. RACEway crossbars may be put together on a RACEway Interlink module using ring, mesh, or tree topologies, for example.

It is possible to provide a bridge between RACEway Interlink and other standards, such as VSB (reference [1]) or PCI (reference [2]).

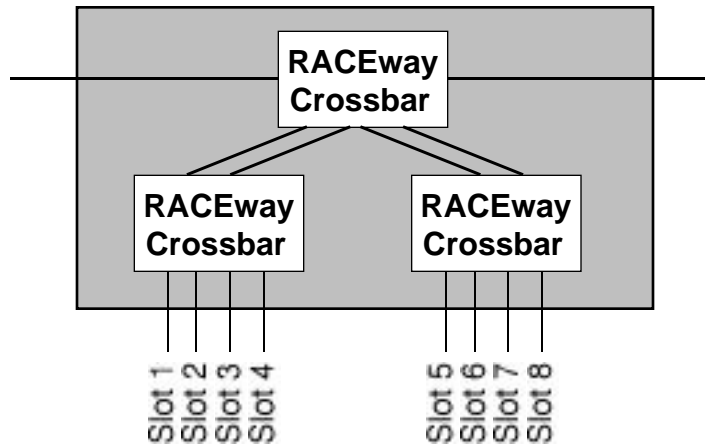


Figure 4 - Possible 8-slot RACEway Interlink topology (Clos network)

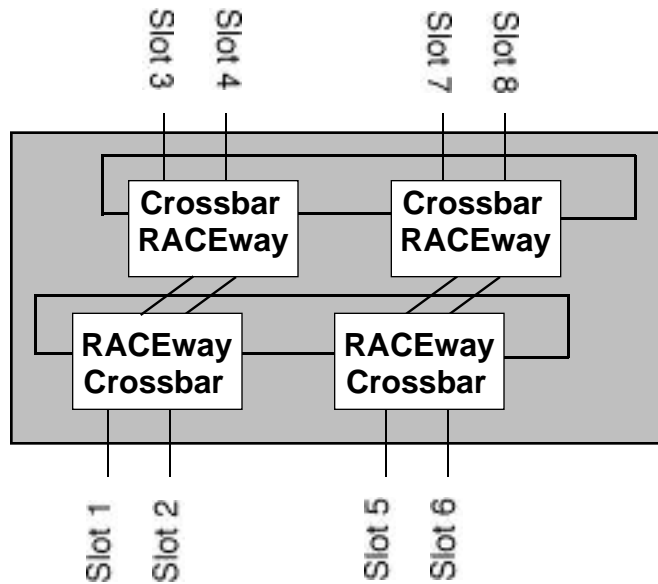


Figure 5 - Possible 8-slot RACEway Interlink topology (Torus (combination mesh and ring))

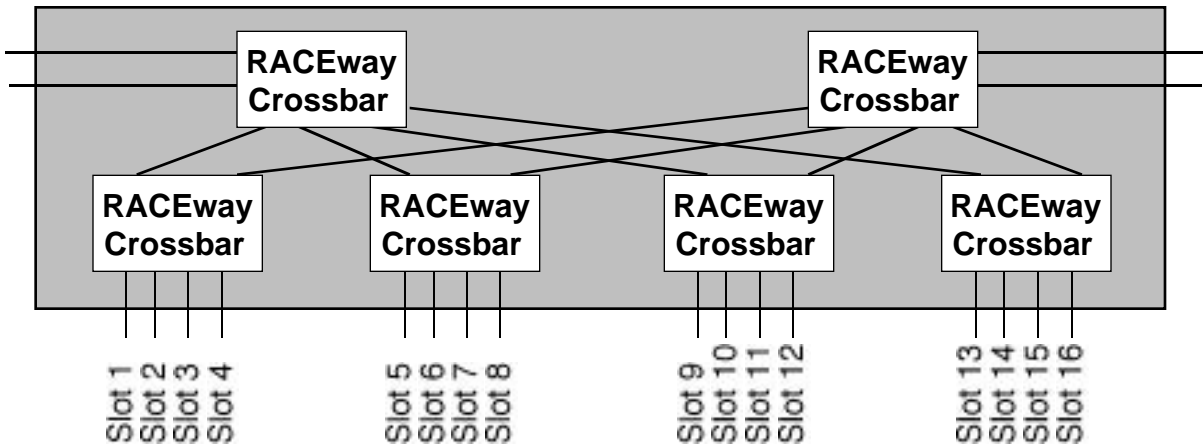


Figure 6 - Possible 16-slot RACEway Interlink topology (Fat tree)

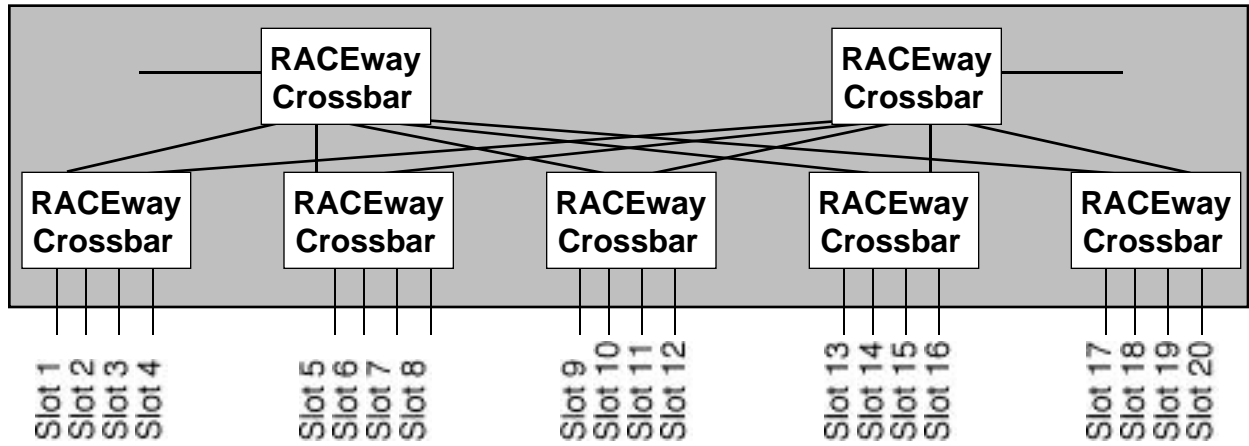


Figure 7 - Possible 20-slot RACEway Interlink topology (Fat tree)

1 Scope and purpose

1.1 Scope

This standard describes a high performance extension to the VMEbus standards ([3], [4]). RACEway Interlink is a backplane communications standard which is compatible with VMEbus, VME64, and most of its extensions.

The Data Link Layer and Physical Layer of RACEway Interlink are specified in this standard. A RACEway Interlink master attaches a header to a transaction to control the routing and addressing of the transaction. The Data Link Layer section specifies the format and contents of this header. A RACEway Interlink compliant board communicates with a RACEway Interlink module through the P2 connector on the backplane. The Physical Layer section specifies the signalling, electrical, and mechanical aspects of this connection.

Higher level protocols may be encapsulated and carried over RACEway Interlink. These are beyond the scope of this standard.

1.2 Purpose

It is the purpose of this standard to fully specify RACEway Interlink so that VMEbus boards and RACEway Interlink modules designed according to this specification will be able to inter-operate at the physical and signalling level.

1.3 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

[1] ANSI/IEEE STD 1096-1988, IEEE Standard for Multiplexed High-Performance Bus Structure: VSB, July, 1989 (expired)

[2] PCI Local Bus Specification, Revision 2.0, April, 1993

[3] ANSI/IEEE STD 1014-1987, VMEbus Specification (expired)

[4] ANSI/VITA 1-1994, VME64 Specification, Rev 1.11, February, 1995

[5] ISO 7498, Open Systems Interconnection Reference Model, October, 1984

[6] ANSI/IEEE STD 1101-1987, IEEE Standard for Mechanical Core Specifications for Microcomputers (expired)

2 Definitions

There are many terms and definitions which have special meaning to VMEbus and RACEway Interlink systems. The following are definitions of terms used in this specification:

2.1 Key words

May: A key word indicating an option with no specific preference.

Shall: A key word indicating a requirement. Requirements must be met to conform to the standard.

Should: A key word indicating an option with a preferred or recommended implementation.

2.2 General

Data Link Layer: The OSI reference model [5] layer specifying the low level logical protocol.

Physical Layer: The OSI reference model [5] layer specifying electrical and mechanical connections.

RACEway Interlink: A high bandwidth, low latency, multistage switching technology. Each stage is a non-blocking, multiport crossbar switch. Raceway Interlink provides high performance by interconnecting the slots using point-to-point connections instead of a bus. The crossbar data path is bidirectional; a slot can either read from or write to another slot. Raceway Interlink maintains inter-operability with existing VMEbus chassis and boards by using a pluggable active backplane on the VMEbus P2 connectors.

2.3 Protocol

Arbitration:

Central arbitration: A scheme where masters send a request to a central resource asking permission to proceed. The central resource (arbiter) directs traffic, coordinating all of the transactions. It prevents collisions and ensures that higher priority transactions get serviced first. It also ensures that no deadlock or starvation situations are created where transactions wait forever to proceed. Central arbitration doesn't scale well, since its complexity grows rapidly with the number of masters it must service.

Distributed arbitration: A method where transactions arbitrate for the needed crossbar resources at the location of those crossbar resources. Each crossbar performs its own arbitration, preventing collisions and ensuring that higher priority transactions get serviced first. The algorithms used must also prevent deadlock and starvation situations. Distributed arbitration scales well, since adding more crossbars adds more arbitration resources at the same time.

Circuit switching: RACEway Interlink uses a pipelined circuit switch fabric rather than a bus architecture. The master of a transaction establishes a route through one or more crossbar switches to the slave. The slave acknowledges that it is connected, and the master begins the transaction. A circuit switch fabric performs better than a packet switch fabric when the number of crossbar hops is small compared to the size of the data transactions. Read and locked read-modify-write transactions (as well as writes) are possible with a pipelined circuit switch fabric. This enables a system to use RACEway Interlink with either a shared memory programming model or a message passing model.

RACEway Interlink uses virtual channels, where different transactions share the same communication resources transparently to the software. In addition, the use of pre-emptive arbitration (where higher priority transactions automatically suspend lower priority transactions) combines real-time performance with the efficiency of a pipelined circuit switch fabric.

Clos switch: This is a very efficient crossbar topology developed by Dr. Charles Clos of AT&T. Figure 5 shows slots which are interconnected using a three stage Clos switch topology.¹

Crossbar hop: A transaction goes through several crossbar switches on the way to its destination. Each crossbar traversed is called one crossbar hop. For instance, a transaction going through three crossbars takes three crossbar hops.

Compelled data transfer protocol: A fully interlocked protocol which only transfers data with permission from both the master and slave.

Kill request: A signal sent to a master requesting that it stop its transfer as soon as possible. The Kill request is initiated by a crossbar which has a port which is needed by a higher priority transaction. The master stops its transfer, freeing up the crossbar ports it was using for use by other transactions. The master restarts its transfer at the address where it stopped. This continuation of the transfer proceeds until it reaches the resources now being used by the higher priority transaction. It waits at that point for the higher priority transaction to complete, and then regains the needed crossbar port(s) and continues.

Master/slave relationship: A master node initiates a transaction to which one or more slave nodes respond.

Mapping registers: A set of registers containing route and address information. Typically several high order bits of a system address are used to select mapping registers containing the route and the high order bits of the address to be used in the header of a transaction. The method of address mapping is implementation specific, and is beyond the scope of this specification.

Node: An endpoint to a transaction. A node can be either a master or a slave. Typically a node is a processor or an I/O device.

Programming models:

Message passing programming model: A programming technique where a process in one node communicates with a process in another node by writing messages back and forth. The messages consist of a header specifying which node the message is intended for, along with control information, data or a combination of the two. Programs typically use subroutine calls to send the messages to a buffer allocated by the message handling services on the remote node. Reads are performed by sending a message asking the other process to send the data back. Transactions over local area networks typically use message passing.

Shared memory programming model: A programming model where a process in the master node communicates with a process in the slave node by writing or reading directly to or from memory on the slave node. A program typically has part of its address space dedicated to the other node, and reads or writes directly to addresses in that space. The hardware translates those addresses into reads or writes to corresponding memory locations in the other node. The hardware typically uses mapping registers to translate an address into a route to the other node and an address within the other node.

Routing types:

Adaptive routing: A method of routing a transaction from the master node to the slave node with the ability to dynamically adapt and route around used crossbar resources.

Broadcast routing: Writing from a master node to all of the other nodes.

Multicast routing: Writing from a master node to a subset of the other nodes.

Single port routing, single port mode: Routing a transaction from a master node to a single slave node. The transaction only uses a single exit port on each of the crossbars it goes through.

Split read: A method of performing a read where the slave tells the master that the read will be completed later. The master suspends its transaction, freeing up the crossbar ports it was using. When the slave is ready, it writes the data back to the master as a separate transaction.

1. Clos, Dr. Charles, "A study of non-blocking switching networks", Bell Syst. Tech. J., vol. 32, pp.406-424, March 1953.

Virtual channel: Software uses virtual channels to communicate with other nodes. These channels use crossbar resources which appear to be dedicated to them, but are actually shared with other virtual channels. The crossbar resources implement a fairness algorithm to ensure that equal priority virtual channels share the resources equally. In addition, virtual channels can be given higher priority to allow them to be completed more quickly by pre-empting lower priority virtual channels.

2.4 Physical

Backplane adapter: A module which is added onto a backplane to improve its connectivity characteristics. In a VMEbus chassis this plugs onto the DIN connector posts which extend through to the back of the backplane. User defined signals on the DIN connector posts are used to connect to the boards plugged into the chassis. The backplane adapter may consist of several parts, one or more active logic boards which may plug into interposer boards.

Interposer: A part of the backplane adapter which plugs onto the DIN connector posts which extend through to the back of the backplane. It is used to interpose between the high insertion force DIN connector posts and the low insertion force connectors of the active logic part of the backplane adapter.

RACEway Interlink module: The backplane adapter which holds the RACEway Interlink logic.

3 Overview

3.1 Characteristics

3.1.1 VMEbus compatibility

RACEway Interlink is compatible with VMEbus and most VME extensions since it uses the user defined pins on rows A, C, D and Z of the P2 connector. It coexists with VMEbus operations, requiring no VMEbus signals or extra slots. It also coexists with existing VME boards which do not use rows A and C (or D or Z) of P2. RACEway Interlink also uses the power and ground pins of row B of the P2 connector. The signal assignments for row B of the P2 connector are specified in Chapter 7 of the VME64 specification [4].

3.1.2 Interconnectivity

RACEway Interlink provides full interconnectivity, being scalable from two slots up to the size of a VMEbus chassis, with any slot able to read or write to any other slot. Bandwidth scales with the number of slots, with four slots supporting up to four simultaneous full speed transfers and twenty slots supporting up to twenty simultaneous full speed transfers. The data link and physical layers have very low overhead, allowing very fast switching for low latency. Broadcast support is also provided, where one slot is able to write to all other slots or to only a selectable subset of slots.

3.1.3 Transaction size

RACEway Interlink is optimized for block transfers of eight byte data words, and also provides for single transfers of one, two or four byte units of data. It uses a compelled data transfer protocol, and is able to burst blocks of data by pipelining the control signals. Transactions have a maximum block size of 2048 bytes.

3.1.4 Resource Contention

RACEway Interlink transactions have either default or assigned priorities to control contention for routing resources. The hardware automatically suspends and resumes lower priority transactions to allow for the predictable delivery of real-time data. Atomic operations are also supported. RACEway configurations with redundant routing resources may have limited adaptive routing capability to automatically use an alternate route if the primary route is busy. Split read support is also provided to read from long latency devices without tying up routing resources. Atomic operation is limited during split read operations.

3.1.5 Physical

RACEway Interlink uses point-to-point, impedance matched, terminated signals to provide robust operation with excellent noise immunity. The Interlink module may fit over 4 slot wide interposers (similar to VSB connectors) for ease of insertion. This works with 6U and 9U boards in 6U and 9U chassis.

3.2 Fundamental concepts

3.2.1 Crossbar network

The RACEway Interlink architecture uses an innovative and powerful crossbar switch chip as its building block. The RACEway crossbar chips are able to gluelessly combine into a crossbar network which is much more scalable and efficient than a centralized crossbar switch. Transactions find their way through the crossbar network based on routing information contained in a transaction header. This is somewhat analogous to the way a telephone call is made, with the routing information equivalent to a telephone number.

3.2.2 Routing

A user defined RACEway master interface adds a header to the beginning of each transaction. The header contains routing information which lists a series of directions to the crossbar chips through which the connection is to be made. Typically, a system address is translated using mapping registers into a pre-calculated route and address which are put into the transaction header. The route is used to connect to the

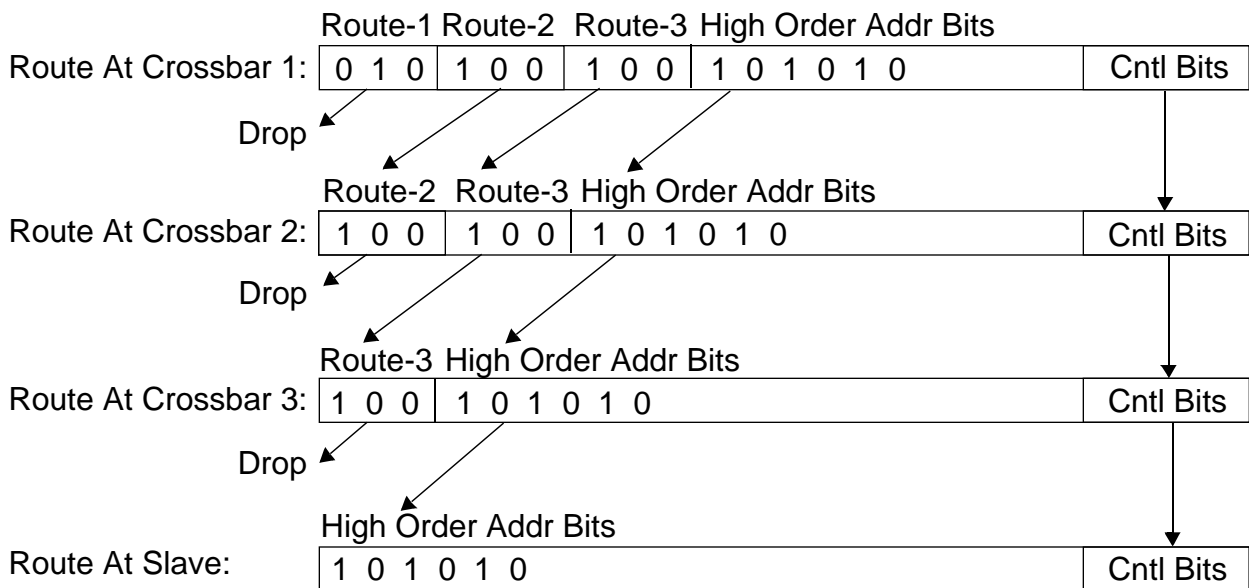
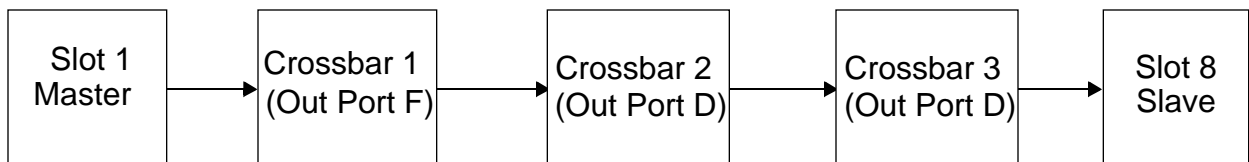
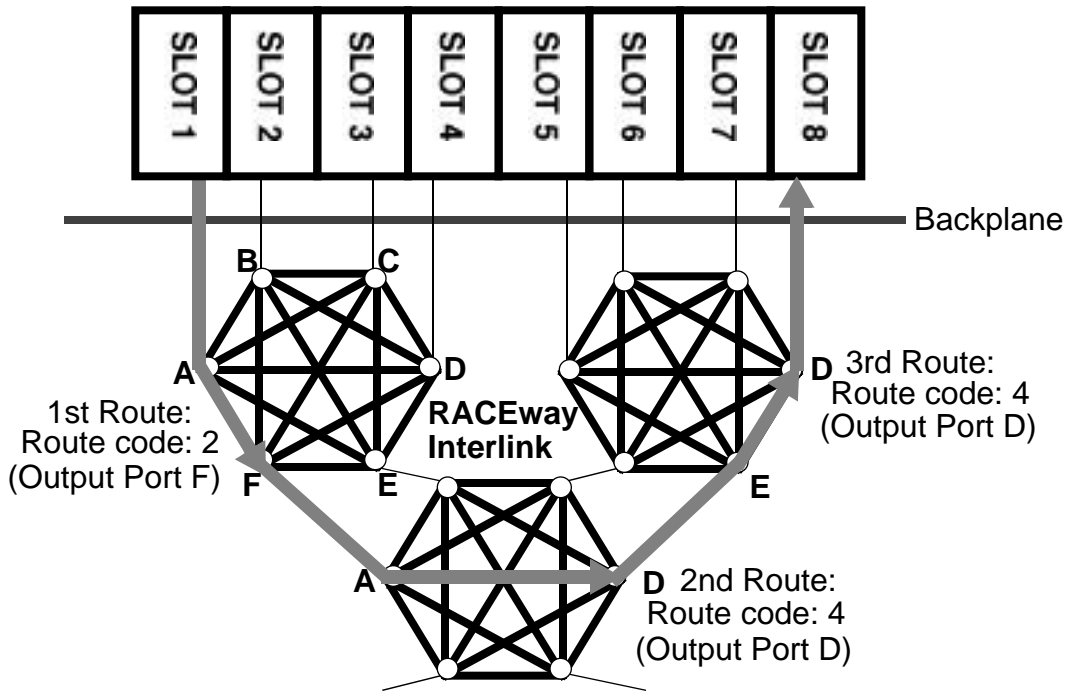


Figure 8 - Routing example

destination(s.) The address is used at the destination(s.) A RACEway interface capable of being a master of a transaction should be able to be configured to support all systems in which it may be used (similar to a master supporting geographic addressing.) This is usually provided by using address mapping registers which are able to be configured by software.

As the route traverses each crossbar, the crossbar logic shifts the route information to expose the directions to be used in navigating through the next crossbar. Figure 8 shows an example of this. Note that the bits used to backfill the shifted route are undefined, and depend on the implementation. Note further that since each crossbar shifts off the used route bits, future crossbar implementations may use four or more bits of route information and then shift them off. Thus future implementations are not limited to three bit route fields or six port crossbars. These future crossbar implementations can be made backward compatible with master interfaces which have route fields which are able to be configured by software. The slave interfaces do not see the Route information, and so could also be compatible with these future crossbar implementations. Note that inter-operability between current and future crossbar implementations would be constrained.

The circuit through the crossbar network is held open for the transaction behind the header. When the route and address reach the slave node, the slave acknowledges the connection, and a circuit through the crossbar network is established. The master node is then able to write or read data through the established circuit.

At the destination, the remaining bits of the Route/Address field of the routing information may be used by the slave as high order address bits. These bits are left justified in the route word due to the way the routes get shifted left at every crossbar.

In order to prevent starvation or deadlock situations, the master shall terminate the transaction when it is complete, it reaches a 2KByte address boundary, or it is killed. When the transaction is terminated, all of the crossbar resources in the path are released for other transactions. In the case of reaching a 2KByte address boundary or being killed, the master then shall start a new transaction using the next sequential address.

3.2.3 Handshaking

After the route and address reach the destination, the slave responds with a signal to inform the master to start writing or reading data. The master is able to read or write from one byte up to 2048 bytes per transaction, with a maximum of 4 bytes transferred every cycle.

RACEway Interlink uses a compelled protocol, so it is possible to throttle back transactions to only transfer data as it becomes available or when it can be accepted. Thus the slave is able to control the data flow with handshaking signals. A slave should be able to receive write data or source read data at 80 MBytes per second or greater. Likewise, the master does not have to write or read data every cycle, but is able to control the data flow with handshaking signals. A master which is unable to complete a transaction within twenty-five microseconds should suspend its transaction at twenty-five microseconds or less so that system resources are fairly utilized, and slave nodes are not needlessly locked up. Thus a master should either be able to maintain a throughput of 80 MBytes per second, or should transfer less than 2048 bytes per transaction.

The control (and data) signals have weak pull-ups on them, so unconnected crossbar ports appear to return the proper handshaking signals. This allows broadcast writes to complete when unused ports are included in the broadcast code. It also allows reads to unused ports to return an error (the ERR signal is active high) so that the configuration software may probe to see which ports are connected. A side effect of this, however, is that faulty writes to unused ports complete without returning an error.

3.2.4 Configuration

Nodes on a RACEway network are typically configured using route tables. Initialization software running on a host can use these to set up mapping registers in each master node. These mapping registers allow the hardware to translate address ranges into Route words and Address words. Future extensions are planned to incorporate CSR registers for automatic configuration.

3.2.5 Latency

The start up latency of a transaction is implementation specific. Typically, routing of a path may take $3 \times X$ cycles of latency through X crossbars. The acknowledgment back may take X cycles. It may also take a few cycles at the master and slave to respond to the signals. The setup of the connection through the crossbar network is therefore typically $4X+4$ cycles for a route through X crossbars. At 40 MHz this is 400 nanoseconds (ns) for a twenty slot configuration which uses three crossbar hops to reach the furthest slot.

Once the route is established, data traversing the connection may take X cycles of latency through X crossbars. After that, four bytes can be transferred every cycle. Thus, a typical minimum write latency for a slave to get the first eight bytes of data is $5 \times X + 6$ cycles. This works out to 525ns for a twenty slot configuration which uses three crossbar hops to reach the furthest slot.

For efficiency, a master should transfer blocks of data whenever possible. This amortizes the routing time overhead across the number of items transferred. This allows a master to acquire a path, use it intensively for a short while, and then release the path for use by other devices.

3.2.6 Priorities and Kill requests

Even after paths are established through a crossbar, a high priority message can successfully acquire a port presently in use by a lower priority message. The sender of the lower priority message is killed (transparently to software running on the sender); the high priority message is routed and sent; and then the lower priority sender's path is automatically re-established, and transmission resumes (again, transparently to software running on the sender).

3.2.7 Adaptive routing

Special route codes are provided to allow a crossbar to dynamically find an open path through the crossbar network. This is a limited (constrained) capability targeted at larger crossbar networks which have redundant routing resources. See Section 4.3 for more information.

3.2.8 Locked operations

The lock capability is essential in certain multicomputer synchronization operations. Locked ("atomic") accesses are needed to implement resource sharing and processor synchronization primitives, such as semaphores. The Lock Flag informs the slave that the transaction is an atomic read-modify-write operation.

3.2.9 Split read transfers

A RACEway network may include slave devices such as serial links, which have a high data-access latency, or which transfer data slowly. The RACEway crossbar architecture provides a special split read capability to reduce the crossbar network traffic of such slow devices. The term "split read" comes from the fact that a slave splits the read into a request and a reply.

The split read capability allows a slave device to request a Return Route from the master. This suspends the master and then releases the crossbar connection between the master and the slave. When the slave is ready to respond, it sends the block of data back to the master using this Return Route. The number of outstanding split read requests is dependent on the implementation of the end points. Typically, end point implementations only allow one outstanding split read at a time.

Split read capable slaves handle locked accesses by performing a test-and-set operation. This approach allows the atomic access to be concluded without waiting for the read data to return. This avoids tying up the crossbar path from the slave back to the master during the (possibly long) time from the master's read request to the slave's read data return.

3.2.10 Split write transfers

A master performing a RACEway write may need acknowledgement from the destination that the operation completed. For instance, this may be used to maintain strong ordering of transactions. A Split Write transaction returns a completion status to the master from the slave after the write completes. The term "split write" comes from the fact that a slave splits the operation into a write and a completion notification.

See Section 5.9.16 for more detail on Split Write transfers.

4 A Data Link Layer specification

4.1 Transaction format

Tables 1 through 3 show the route, address, and data word formats. Table 4 gives the trade-off between address bits and route bits.

With the maximum number of route bits being used, there are a total of 28 address bits (address word bits 30:3 (Tables 3 and 17)). Bits 27:3 address an 8 byte quantity and 30:28 can be used to select bytes within that 8 byte quantity (see Section 4.6.)

In order to obtain the maximum number of address bits, six additional high order address bits may be placed in the route/address field of the route word. This reduces the route field to 21 bits in length. These high order address bits get shifted with the route bits and appear at the slave as the left-most bits of the route/address field.

The master has the responsibility to properly set up the route and address fields to support the number of crossbar hops and the number of address bits for transactions with a given slave (see Section 4.7.)

A master shall kill and re-arbitrate transactions at 2K byte address boundaries, so there are no more than 256 eight byte data words per transaction. This helps to ensure an equal sharing of the routing resources by equal priority transactions.

Table 1 - Transaction format

Start of transaction:	ROUTE [31:0]
	ADDRESS[31:0]
	DATA 0[63:32]
	DATA 0[31:0]
	...
	DATA n[63:32]
End of transaction:	DATA n[31:0]

Table 2 - Route word format

31	5	4	3	2	1	0
Routes/ High Order Address Field		Broadcast Accept Code	Routing Priority	Broadcast/ Single Mode		

Table 3 - Address word format

31	28	27	3	2	1	0
Width/Alignment	Address		Reserved	Read Flag	Lock Flag	

Table 4 - Route/address space trade-offs

Base format	Route Bits	Address Space
Maximum Address space	21	2^{34} bytes
Maximum Number of Route Bits:	27	2^{28} bytes

4.2 Extended route/address format

The purpose of this section is to reserve a header word for future extensions (see Section 5.9.17.) Tables 5 through 8 show the route, extended route, address, and data word formats. The intent is that the master can put additional route bits or high-order address bits in the extended route word to perform transactions with a slave which need more than 27 bits of route and/or 34 bits of address.

The extended route word shall use cycles not needed with the normal route and address format. These cycles are reserved (unused) at the slave interface to maintain their ability to inter-operate with this future addressing extension. Note that there will be constraints on this inter-operability.

Section 5.9.17 defines these (currently unused) cycles when the extended route/address word shall be driven by the master and received by the slave.

The extended route/address field shall be shifted into the low order bits of the route/address field as the route bits get shifted off at each crossbar, extending the route and address fields.

Table 9 gives a possible trade-off between address bits and route bits using the extended route word.

Table 5 - Extended transaction format

Start of transaction:	ROUTE
	Extended ROUTE (Optional)
	ADDRESS
	DATA 0[63:32]
	DATA 0[31:0]
	...
	DATA n[63:32]
End of transaction:	DATA n[31:0]

Table 6 - Route word format

31	5	4	3	2	1	0
Routes/ High Order Address Field		Broadcast Accept Code	Routing Priority	Broadcast/ Single Mode		

Table 7 - Optional additional extended route word format

31	0
Reserved	

Table 8 - Address word format

31	28	27	3	2	1	0
Width/Alignment		Address		Reserved	Read Flag	Lock Flag

Table 9 - Possible extended route/address space trade-offs

Using Optional Route/Address	Route Bits	Address Space
Maximum Address space	21	2^{64} bytes
Maximum Number of Route Bits	57	2^{28} bytes

4.3 Routing code/Broadcast Mode

Table 10 gives the route code format within the route word. Broadcast operations are performed when the Broadcast Flag (bit [0]) = 1. Single port routing is performed when the Broadcast Flag = 0.

Table 10 - Route word format - Route code

31	28	25	22	19	16	13	10	7	4	3	2	1	0
Route 0	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Route 7	Route 8	Broadcast Accept Code	Routing Priority	Broadcast / Single Mode		
(or up to 6 addr bits)													

The crossbar routing codes specify the exit port(s) for each crossbar in the path. The codes are defined in Table 11. If the route word contains high order address bits then broadcast operations are only valid for slave destinations the same number of crossbar hops from the master.

Table 11 - Routing codes

Code	Single mode exit port	Broadcast mode exit port
7	A	B, C, D if entered at A, else A [*]
6	B	A, C, D if entered at B, else B [*]
5	C	A, B, D if entered at C, else C [*]
4	D	A, B, C if entered at D, else D [*]
3	E	E
2	F	F
1	E first, adaptive route [‡]	A, B, C, D, E [†]
0	F first, adaptive route [‡]	A, B, C, D, F [†]

* If a broadcast message enters port A, B, C, or D and the single-port routing code matches the port ID, then the message exits through all other valid ports except E and F. Otherwise, the message exits through the port defined by the single-port routing code. For example, if a message enters at port A with code 7, it exits through ports B, C, and D. If a message enters port B with code 7, it exits through port A.

If a broadcast message enters ports E or F, it will exit a single port (A, B, C, or D) in response to routing codes 7, 6, 5 or 4, respectively. For example, if a message enters port E with code 7, it exits through port A.

† For codes 1 and 0, broadcasts will exit all ports in the list except the port the message entered. A broadcast message entering either port E or F, going to A, B, C, and D, shall use codes 1 or 0, respectively. For example, a broadcast entering port F uses a route code of 0 to go to ports A, B, C, and D. A broadcast entering port F uses a route code of 1 to go to ports A, B, C, D and E, and a broadcast entering port A uses a route code of 1 to go to ports B, C, D and E.

‡ The adaptive route option only works with symmetric crossbar networks where the rest of the route fields will be valid no matter which of the two ports is taken. Adaptive routing is available when a crossbar is used in single port mode. In adaptive routing mode, the routing logic attempts to route through the assigned port (E or F). If that port is busy, the routing logic attempts to route through the other port. This process toggles between the two ports until arbitration succeeds.

4.4 Broadcast acceptance code

The route word of each broadcast transaction contains the broadcast acceptance code in bits [4:3], as shown in Table 12. Slaves may accept the arriving broadcast if the acceptance code matches the slave's acceptance key. A master may use this mechanism to select different sub-populations of slaves for multicast operation.

The slave shall respond to the broadcast transaction whether or not the acceptance code matches the key.

The two bit broadcast acceptance code gives up to four slave populations for multicast operation. Multicast operation may also be performed by using the broadcast route codes to route a broadcast to a subset of the nodes. A combination of the two methods may also be used.

Table 12 - Route word format - Broadcast accept code

31	28	25	22	19	16	13	10	7	4	3	2	1	0
Route 0	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Route 7	Route 8	Broadcast Accept Code		Routing Priority		Broadcast/ Single Mode = 1
(or up to 6 addr bits)													

When not in Broadcast mode, bit [4] is reserved and shall be set to 0. Bit [3] is used as the SPLIT mode flag. A master that supports Split Read and Write operations shall set bit [3] of the Route word to 1 when bit [0] = 0. See Section 3.2.9 and Section 3.2.10 for more information on Split transaction capabilities. See Section 5.9.15 and Section 5.9.16 for more detail on Split transactions.

Table 13 - Route word format - Split mode

31	28	25	22	19	16	13	10	7	4	3	2	1	0
Route 0	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Route 7	Route 8	Rsvd =0	Split Mode	Routing Priority		Broadcast/ Single Mode = 0
(or up to 6 addr bits)													

4.5 Routing priority code

The route word of each transaction contains the routing priority in bits [2:1], as shown in Table 14. Table 15 defines those priorities. Higher priority messages take precedence over lower priority messages. The lower priority transaction will automatically be killed, the higher priority transaction will take place, and then the lower priority transaction will automatically start back up where it left off.

Table 14 - Route word format - Routing priority code

31	28	25	22	19	16	13	10	7	4	3	2	1	0
Route 0	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Route 7	Route 8	Broadcast Accept Code	Routing Priority		Broadcast/ Single Mode	
(or up to 6 addr bits)													

Table 15 - Transaction priority code

Priority Code		Priority Level
Bit 2	Bit 1	
0	0	0 (lowest)
0	1	1
1	0	2 (highest)
1	1	3 (reserved)

Priority operation:

RACEway uses distributed arbitration instead of centralized arbitration. Each crossbar uses transaction priorities to resolve contention and schedule transactions. This distributed arbitration ensures deadlock and starvation free performance, while providing real-time responsiveness to transactions which need it.

When a RACEway crossbar detects that a higher priority transaction wants a port already in use by a lower priority transaction, it sends a Kill request to the master of the lower priority transaction.

The master receiving the Kill request shall complete any outstanding operations and terminate its current transaction. For example, it shall wait for any read data already requested to return before terminating its transaction.

The master shall then continue operation by starting a new transaction using the next sequential address from where it was suspended. This new transaction shall be initiated at least four cycles after terminating the previous transaction.

This new transaction will propagate forward until blocked by a higher priority transaction. It then waits at that crossbar for the higher priority transaction to terminate. When the higher priority transaction terminates, then the lower priority transaction continues traversing the crossbar network.

Transaction priority at a RACEway crossbar is established in two ways. The route priority is designated in the transaction header by the two bit priority field in the route word. The port priority is determined by which port the transaction arrives at the crossbar. Crossbar ports are designated alphabetically, where port A has the lowest priority, and the highest alphabetical port has the highest priority. (This is port F in a six port implementation.)

A transaction shall be killed when it uses a port needed by a higher route priority transaction.

A transaction may be killed in the following cases for efficient arbitration:

1. it is blocked waiting for another port, so a new transaction at the same route priority (coming in on any other port of the same crossbar) that needs the blocked transaction's input port takes precedence.
2. it uses a port needed by a transaction entering the crossbar at either of the two highest priority input ports (ports E and F in a six port implementation) with the same route priority.
3. it is routed out either of the two highest priority ports (E or F in a six port implementation) and hasn't received an acknowledge back from the slave indicating that the circuit is connected (so it probably is blocked), and a new transaction at the same route priority wants its input port.

There is also a default priority for tie breaking when requests come into a crossbar on the same cycle and at the same route priority. The higher port priority wins (i.e. port D will win over port C).

When two or more transactions of equal route priority (and not coming in either of the two highest priority ports) are blocked waiting for the same exit port, then the port which was waiting longest shall win. The losing transaction shall continue to wait for its turn. This ensures a round-robin fairness among transactions of equal route priority.

Additional priority rules may be implemented to provide efficient arbitration.

4.6 Transfer width and alignment specification

The width/alignment bits use the address word bits [31:28] as shown in Table 16. Table 17 defines the function of the width/alignment bits.

The data path on each port of the RACEway crossbar chip is conceptually 8 bytes wide. The crossbar data path is physically 4 bytes wide, with crossbar logic automatically transferring 8 byte quantities over two consecutive cycles. This double cycling is transparent to software running on the master.

A RACEway slot may use the crossbar to access data that is 1 byte, 2 bytes, 4 bytes or 8 bytes wide. In addition, most alignments of 1-byte, 2-byte and 4-byte data are supported. The data-format information resides in the address word. Block mode operation may only be supported for 8 byte words.

Table 16 - Address word format - Transfer width/alignment

31	28	27	3	2	1	0
Width/Alignment		Address		Reserved	Read Flag	Lock Flag

Table 17 - Data width and alignment code

Width (Bytes)	Alignment	Byte Enables								Width/Align Bits			
		^{63:56} B7	B6	B5	B4	B3	B2	B1	^{7:0} B0	31	30	29	28
1	B7	1								0	0	0	0
1	B6		1							0	0	0	1
1	B5			1						0	0	1	0
1	B4				1					0	0	1	1
1	B3					1				0	1	0	0
1	B2						1			0	1	0	1
1	B1							1		0	1	1	0
1	B0								1	0	1	1	1
2	B7:B6	1	1							1	0	0	0
2	B5:B4			1	1					1	0	1	0
2	B3:B2					1	1			1	1	0	0
2	B1:B0							1	1	1	1	1	0
4	B7:B4	1	1	1	1					1	0	0	1
4	B3:B0					1	1	1	1	1	1	0	1
8	B7:B0	1	1	1	1	1	1	1	1	1	0	1	1

4.7 Address

Six high order address bits may be located within the route word as shown in Table 18. The exact location within bits [31:5] where these address bits start out depends on how many crossbars the packet is going to traverse. The remaining address bits are in the address word at bits [27:3] and [31:28], as shown in Table 19.

RACEway route/address header words contain up to a 31-bit address, which (together with the width and alignment information) provides a 34-bit address to be used by the slave device. This gives a master access to up to 16 GBytes of slave address space.

The master shall put the high-order address bits in the route/address field for transactions with slaves which use more than 28 bits of addressing.

The master shall place the address bits left justified next to the last route bits which are needed to traverse the crossbars between the master and the slave. Figure 8 shows an example of this.

A slave which uses more than 28 bits of addressing shall be able to get its high order address bits from bits 31:26 of the route/address field (see Section 5.9.2 for a description of when the route word is valid at the slave.)

Use of the high order address bits in the ROUTE word shall be constrained to ensure that an equal number of hops is taken between the master and slave(s) when using broadcast or adaptive routing.

Table 18 - Route word format - High order address bits

31	5	4	3	2	1	0
Routes/ High Order Address Field		Broadcast Accept Code	Routing Priority	Broadcast/ Single Mode		

Table 19 - Address word format - Address bits

31	28	27	3	2	1	0
Width/Alignment		Address		Reserved	Read Flag	Lock Flag

4.8 Reserved field

Table 21 gives the location of the Reserved field within the address word. Bit [2] is reserved and shall be set to 0.

Table 20 - Address word format - Reserved field

31	28	27	3	2	1	0
Width/Alignment		Address		Reserved	Read Flag	Lock Flag

4.9 Reads

The Read Flag uses address word bit [1] as shown in Table 21.

Reads, read-modified-writes, and split read operations shall set the Read Flag to 1.

Writes and broadcast operations shall set the Read Flag to 0.

Table 21 - Address word format - Read Flag

31	28	27	3	2	1	0
Width/Alignment		Address		Reserved	Read Flag	Lock Flag

4.10 Locked transfers

The Lock Flag uses address word [0], as shown in Table 22.

Locked operations shall be performed when the Lock Flag = 0. A master shall perform a locked read-modify-write operations for one word of 8 or less bytes.

A crossbar may try to suspend a locked operation, but the master shall ignore the Kill request until the locked transaction is complete.

Table 22 - Address word format - Lock Flag

31	28	27	3	2	1	0
Width/Alignment		Address		Reserved	Read Flag	Lock Flag

5 Physical layer specification

5.1 Interface

This section contains a detailed description of how the physical interface layer RACEway Interlink operates.

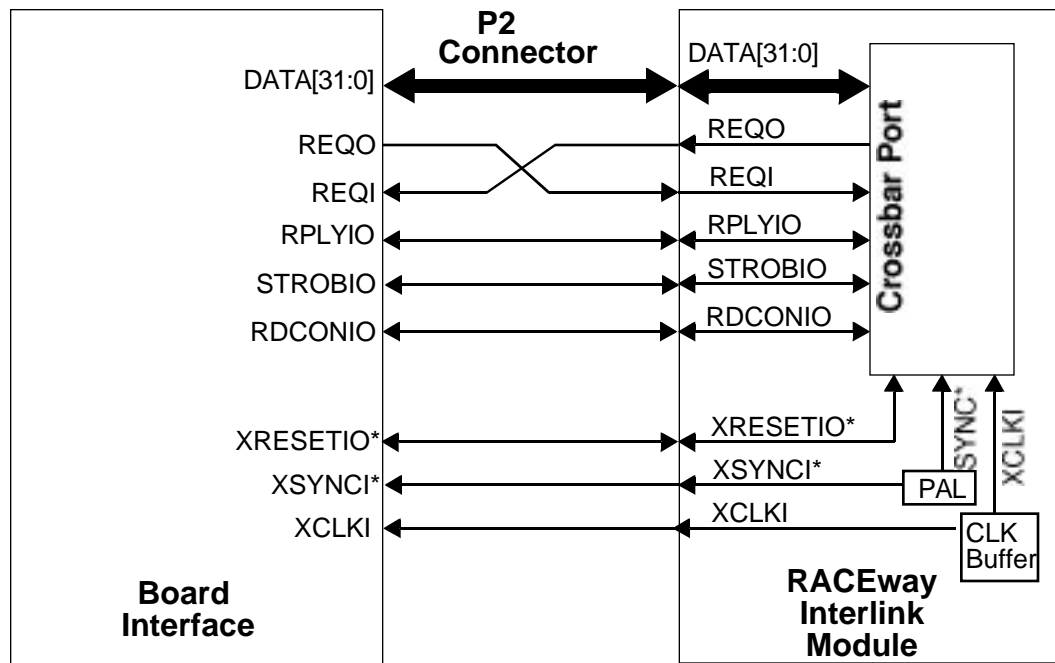
A RACEway board has a connector and signalling protocol which permit it to interface with one of the ports of a RACEway Crossbar chip on a RACEway Interlink module via the P2 connector of the VMEbus. (The RACEway Crossbar chip is the building block which interconnects (gluelessly) to create the crossbar network.) Each interface has 32 data lines and 8 control lines. Several different controls are multiplexed over the control lines. These control the establishment of a route and the transfer of data along the route. The interface between RACEway Crossbar chips is the same as the interface between a Crossbar chip and a RACEway board.

While the signals are all synchronous, the higher level transaction protocol is asynchronous. Master and slave ports interact through handshaking signals for compelled data transfers.

The following pages will describe what the control signals are and how they are used for write and read transactions over the crossbar network.

5.2 Connections

Figure 9 shows how a board connects to the RACEway Interlink module through the P2 connector.



* Indicates an active low signal

Figure 9 - P2 Interface Connections

5.3 P2 Pinout

RACEway Interlink uses the user-defined pins of the P2 connector, along with the power and ground pins in row B. Rows A and C are used for one Interlink connection; rows Z and D of the 160 pin connector are used for the second Interlink connection. Table 23 shows the P2 pin assignments. XB1_ indicates a signal connected to the first Interlink's crossbar; XB2_ indicates a connection to the second Interlink's crossbar.

Table 23 - P2 pin assignments

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
z1	XB2_RPLYIO	A1	XCLKI	B1	+5 Volts	C1	XRESETIO*	D1	XB2_IO00
z2	GND	A2	GND	B2	GND	C2	Reserved	D2	GND
z3	XB2_REQI	A3	XB1_IO09	B3		C3	XSYNCI*	D3	XB2_IO01
z4	GND	A4	XB1_IO08	B4		C4	GND	D4	XB2_IO02
z5	XB2_REQO	A5	GND	B5		C5	XB1_IO07	D5	GND
z6	GND	A6	XB1_IO06	B6		C6	GND	D6	XB2_IO03
z7	XB2_RDCON	A7	GND	B7		C7	XB1_IO11	D7	XB2_IO04
z8	GND	A8	XB1_IO10	B8		C8	GND	D8	XB2_IO05
z9	XB2_STRBIO	A9	XB1_IO04	B9		C9	XB1_STRBIO	D9	GND
z10	GND	A10	GND	B10		C10	XB1_RPLYIO	D10	XB2_IO06
z11	XB2_IO21	A11	XB1_IO05	B11		C11	GND	D11	XB2_IO07
z12	GND	A12	XB1_IO03	B12	GND	C12	XB1_REQI	D12	GND
z13	XB2_IO22	A13	GND	B13	+5 Volts	C13	XB1_REQO	D13	XB2_IO08
z14	GND	A14	XB1_RDCON	B14		C14	GND	D14	XB2_IO09
z15	XB2_IO23	A15	Reserved	B15		C15	XB1_IO02	D15	XB2_IO10
z16	GND	A16	GND	B16		C16	XB1_IO01	D16	GND
z17	XB2_IO24	A17	XB1_IO00	B17		C17	GND	D17	XB2_IO11
z18	GND	A18	XB1_IO15	B18		C18	XB1_IO12	D18	XB2_IO12
z19	XB2_IO25	A19	GND	B19		C19	XB1_IO25	D19	GND
z20	GND	A20	XB1_IO24	B20		C20	GND	D20	XB2_IO13
z21	XB2_IO26	A21	XB1_IO31	B21		C21	XB1_IO29	D21	XB2_IO14
z22	GND	A22	GND	B22	GND	C22	XB1_IO30	D22	GND
z23	XB2_IO27	A23	XB1_IO28	B23		C23	GND	D23	XB2_IO15
z24	GND	A24	XB1_IO27	B24		C24	XB1_IO26	D24	XB2_IO16
z25	XB2_IO28	A25	GND	B25		C25	XB1_IO23	D25	GND
z26	GND	A26	XB1_IO22	B26		C26	GND	D26	XB2_IO17
z27	XB2_IO29	A27	XB1_IO20	B27		C27	XB1_IO19	D27	XB2_IO18
z28	GND	A28	GND	B28		C28	XB1_IO21	D28	GND
z29	XB2_IO30	A29	XB1_IO18	B29		C29	GND	D29	XB2_IO19
z30	GND	A30	XB1_IO17	B30		C30	XB1_IO16	D30	XB2_IO20
z31	XB2_IO31	A31	GND	B31	GND	C31	XB1_IO14	D31	
z32	GND	A32	XB1_IO13	B32	+5 Volts	C32	GND	D32	
*	*	* The signal assignments for row B are specified in Chapter 7 of the VME64 specification [4].							

5.4 Master interface signals to the RACEway crossbar network

Tables 24 and 25 give the RACEway control signals sent and received by the master.

Table 24 - Control signals sent by the master

Symbol	Pin	Name and Function
ASTROBE	†STRBIO	AStrobe is sent by the master during 1 (after asserting Request Out) to indicate that a transfer is occurring.
DATA[31:0]	†IO[31:0]	DATA[31:0] carries the Route, Address, and Write Data between the master and the slave.
DSTROBE	STRBIO	Data Strobe is sent by the master during 0 (after receiving DSEN) to indicate that Data is being driven on DATA[31:0] for a Write or to request Data from the slave during a Read.
REQO	†REQO	Request Out is asserted by a master during 0 to get control of DATA[31:0], it then stays asserted as long as the master is in control. It shall only be asserted if Request In was inactive coincident with the previous 1 control phase.
RESET	XRESETIO*	RESET is asserted to clear and initialize the RACEway crossbar network. It is an open collector signal.

† is used in place of XB1_ or XB2_, as appropriate.

Table 25 - Control signals received by the master

Symbol	Pin	Name and Function
0	XCLKI, XSYNCI*	Control Phase 0 is a time period synchronized by the XSYNCI* control.
1	XCLKI, XSYNCI*	Control Phase 1 is the opposite sense of 0.
CHANGE TO ADDRESS	†RPLYIO	Change to Address is received coincident with 0 (after raising Request Out) to indicate that the slave is ready to receive the Address.
DATA[31:0]	†IO[31:0]	DATA[31:0] carries Read Data between the slave and the master.
DSEN	†RPLYIO	Data Strobe Enable is received coincident with 1 (after raising Request Out) to indicate that the slave is ready to receive Data. One DSEN pulse is sent for every 64 bits of Data the slave node is able to receive.
ERR	†RDCON	Error is received coincident with 1 for a Read operation after the last READ READY to indicate that a Read error occurred.
KILL	†REQI	KILL is received by a master coincident with 1, while it is asserting Request Out. The master shall then stop its transaction and drop Request Out.
RDCON*	†RDCON	RDCON* is received coincident with 0 to tell the master to get off DATA[31:0] for 2 cycles.
READ READY	†RPLYIO	Read Ready is received coincident with 0 for a Read operation to indicate that the slave is driving Data on DATA[31:0].
SPLIT	†RPLYIO	Split is received coincident with 0 (after Change To Address and before DSEN) to indicate that the slave is splitting the Read and is ready to receive the Return Route.

5.5 Slave interface signals to the RACEway crossbar network

Tables 26 and 27 give the RACEway control signals sent and received by the slave.

Table 26 - Control signals sent by the slave

Symbol	Pin	Name and Function
CHANGE TO ADDRESS	†RPLYIO	Change to Address is sent by the slave coincident with 1 (after receiving Request In) to indicate that it is ready to receive the Address.
DATA[31:0]	†IO[31:0]	DATA[31:0] carries Read Data between the slave and the master.
DSEN	†RPLYIO	Data Strobe ENable is sent by the slave coincident with 0 (after receiving Request In) to indicate that it is ready to receive Data. One DSEN pulse is sent for every 64 bits of Data the slave node is able to receive.
ERR	†RDCON	Error is sent by the slave coincident with 0 for a Read operation after the last DSEN to indicate that a Read error occurred.
RDCON*	†RDCON	RDCON* is sent by the slave coincident with 1 (after receiving an Address with the Read bit on) to tell the master to get off DATA[31:0] for 2 cycles.
READ READY	†RPLYIO	Read Ready is sent by the slave coincident with 1 for a Read operation to indicate that it is driving Data on DATA[31:0].
SPLIT	†RPLYIO	Split is sent coincident with 1 (after sending Change To Address and before sending DSEN) to indicate that the slave is splitting the Read and is ready to receive the Return Route.

† is used in place of either XB1_ or XB2_, as appropriate.

Table 27 - Control signals received by the slave

Symbol	Pin	Name and Function
0	XCLKI, XSYNCI*	Control Phase 0 is a time period synchronized by the XSYNCI* control.
1	XCLKI, XSYNCI*	Control Phase 1 is the opposite sense of 0.
ASTROBE	†STRBIO	AStrobe is received by the slave coincident with 0 (after receiving the Route) to indicate that a transfer is occurring.
DATA[31:0]	†IO[31:0]	DATA[31:0] carries Write Data between the master and the slave.
DSTROBE	†STRBIO	Data Strobe is received by the slave coincident with 1 (after sending DSEN) to indicate that valid data is coming on DATA[31:0] for a Write. The number of DSTROBEs received will not exceed the number of DSEN pulses sent. For a Read operation it is received to request 64 bits of data.
REQI	†REQI	Request In is received by a slave coincident with 1 when the master is taking control of DATA[31:0].
RESET	XRESETIO*	RESET is received to clear and initialize the internal logic of the slave.

5.6 Signal characteristics

Annex A details the RACEway Crossbar signal characteristics.

RACEway Interfaces shall inter-operate with TTL logic levels. (Note that the requirements for V_{IH} for the data and control signals are slightly more restricted than the normal TTL V_{IH} . However, the input buffers are very high impedance, so the V_{IH} is specified with current < 1 mA.) (The clock operates at TTL logic levels.)

5.7 Signal notation

Figure 10 shows the timing notation used in the following sections.

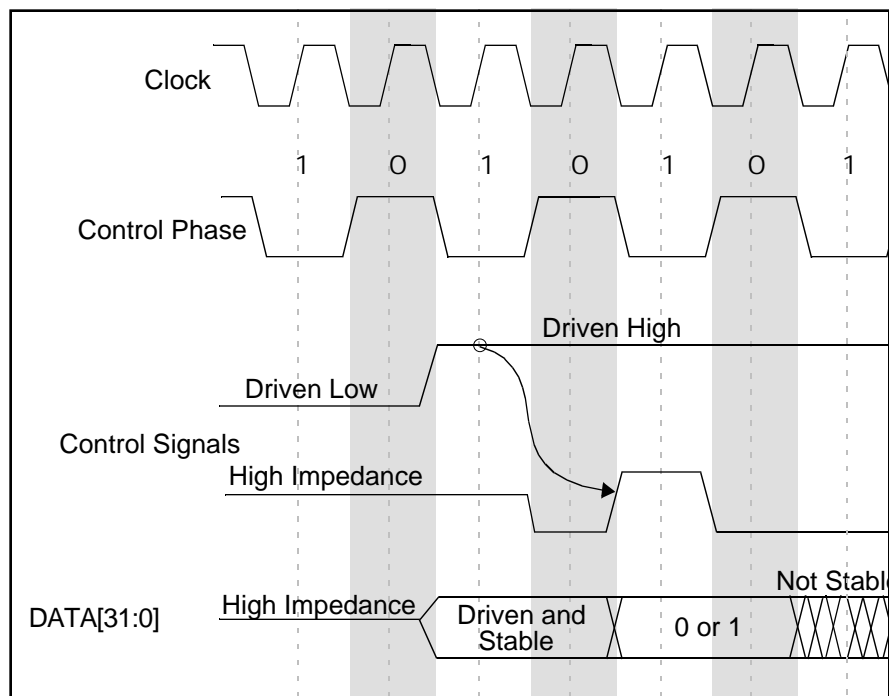


Figure 10 - Timing notation

Note that signals may be described in one of two ways, depending on the reference point. The timing diagrams should be viewed as the authority; textual comments may either refer to a signal which is received or asserted, which does not imply received by a clocked register, or to a signal which is sampled or clocked, which does imply the use of a register or equivalent device. The diagrams should remove any confusion as to which is implied.

5.8 Master/slave relationship

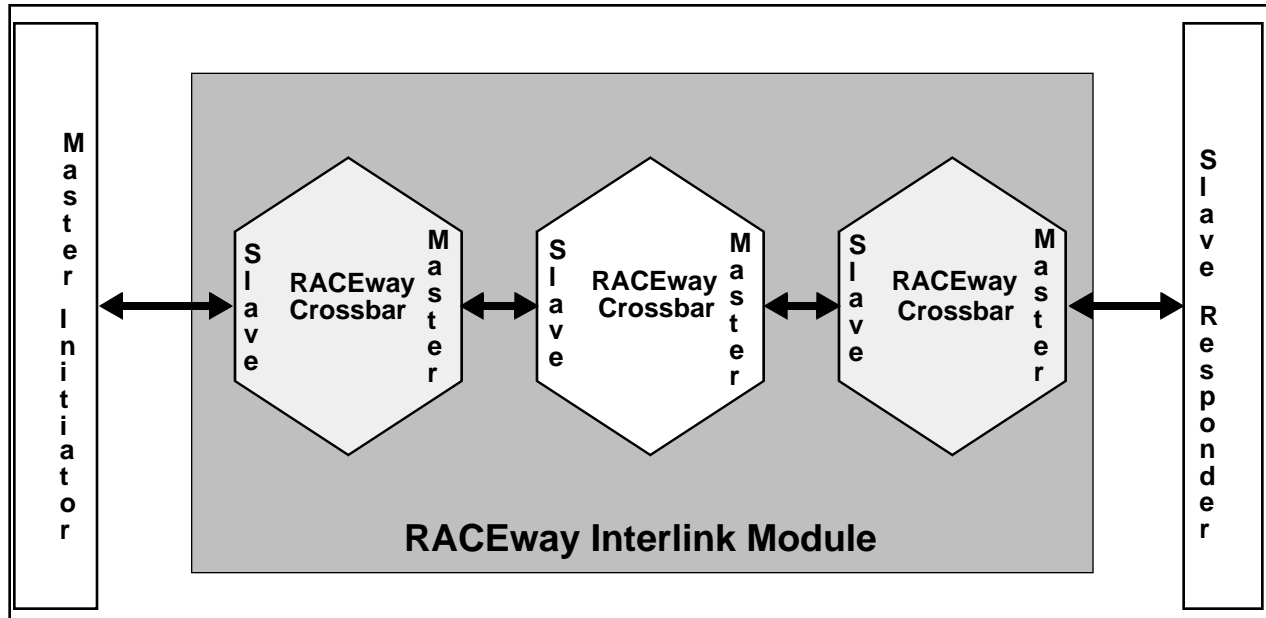


Figure 11 - Master/slave relationship

RACEway interfaces behave identically, whether they are between a crossbar and an endpoint or between crossbars. A Master Initiator starts a transaction, with the connected crossbar acting like a slave. The master - slave relationship is repeated through the crossbar network to the Slave Responder(s). The signals generally start at the endpoints and are relayed through the crossbar network using the same timing relationships as between the initiator and the first crossbar. Figure 11 shows this relationship.

Thus, almost all of the master signals are initiated by the Master Initiator, and propagated through the crossbar network. Almost all of the slave responses come from the Slave Responder(s), and are relayed back to the Master Initiator through the crossbars. Three exceptions are:

1. Change To Address signal, which comes from each slave side along the route.
2. Kill Request, which is also generated by crossbars.
3. Shifted Route, which is generated by each master along the way. In the case of the RACEway crossbars the Shifted Route is a shifted version of the Shifted Route which this same crossbar just received from the previous master up the chain.

The crossbars distribute the signals to all of the specified ports for broadcasts. The crossbars also collect and coordinate the return responses for broadcasts.

The Master/slave relationship exists for the transaction life, and may be reversed for the next transaction. Controls are driven in one direction for the duration of the transaction, with DATA[31:0] able to be turned around during a transaction for Read and Read-Modify-Write operations.

The master shall drive all its data control signals on the same cycle that it starts a transaction, and tri-state all of its data and controls on the same cycle it completes a transaction.

The slave shall drive all of its control signals one cycle after becoming a slave, and shall tri-state its controls one cycle after the transaction is complete.

5.9 Timing

5.9.1 Clocks and control phases

Events in a RACEway crossbar network are synchronized by a global 40 MHz clock, and by a control phase signal derived from the 40 MHz clock. A clock line (XCLKI) and a phase sync line (XSYNCI*) are distributed from the RACEway Interlink to each slot.

During phase 0 (0), the control phase is sampled high; during phase 1 (1), the control phase is low (see Figure 12). Each phase of the control phase signal lasts for one cycle and is sampled on the rising edge of the clock.

DATA[31:0] is clocked on the rising edge of the data clock (XCLKI). Data is always transferred over DATA[31:0] in 64 bit quantities, with bits [63:32] sent during one control phase and bits [31:00] sent during the following control phase. (Note that the width and alignment bits of the address field shall be used to select valid bytes of the 8 byte quantity.)

Each signal wire is able to carry two different control signals in the same direction, with the control phase used to differentiate between the two. Thus, control signals may have different meanings depending on whether they occur coincident with 0 or 1.

Two out of phase synchronization signals are distributed such that a board and the crossbar port it is connected to on the Interlink module see opposite control phases. A board sees 0 at the same time the connected Interlink port sees 1. Likewise, a board sees 1 at the same time the connected Interlink port sees 0. The synchronization signal supplied to the board by the RACEway Interlink interface is called XSYNCI*.

Thus, controls which are initiated during one phase are always received during the opposite phase. For instance, REQO is asserted coincident with 0, and is simultaneously received (at the REQI input) coincident with 1 on the other side of the connection.

This is an important feature because it prevents control collisions between the board and the Interlink module. For instance, requests to get control of DATA[31:0] (REQO) only occur coincident with 0. Since the board sees 0 at a different time than the Interlink, it is impossible for both to initiate a REQO and drive the Route information onto DATA[31:0] at the same time. Thus arbitration for control of the interface between a board and the Interlink becomes much easier because there are no ties and no collisions.

A crossbar shall receive the control phase opposite to all crossbars or slots to which it is connected. This is a minor restriction, but some topologies, such as crossbars connected in a triangle, are not allowed.

The controls (and data) are pipelined through one or more Crossbar chips on the RACEway interlink module. The timing relations are repeated at each stage of the pipeline, traversing up to 9 crossbar chips.

Note that while RACEway uses a global clock, it is not necessary to control clock skew across the entire system. The skew only needs to be controlled between adjacent crossbar chips. Aligning the clock propagation topology to the crossbar interconnect topology simplifies the clock distribution requirements.

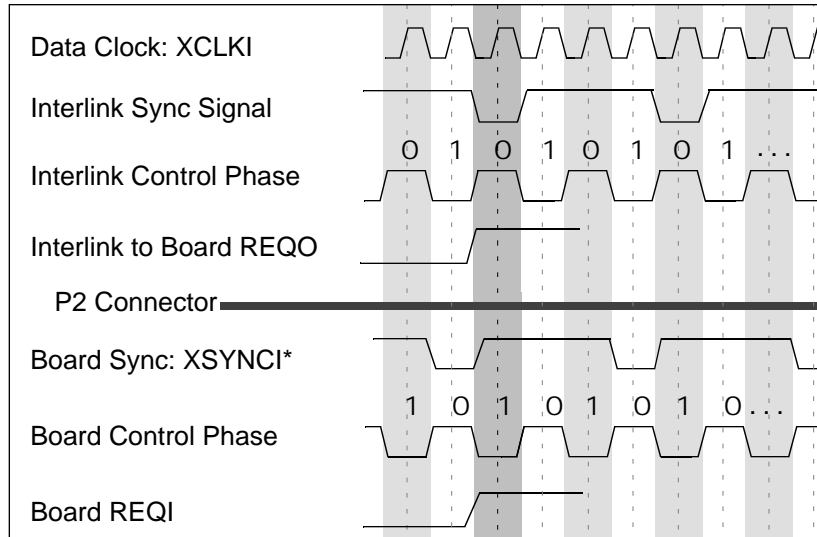


Figure 12 - Clocks and control phases

5.9.2 Route phase: REQ0, REQ1, ROUTE, SHIFTED ROUTE

A master initiates a transaction by starting the Route Phase, where a path is established between the Master Initiator and the Slave Responder(s).

A master shall initiate a transaction by asserting REQ0 coincident with 0 while driving the ROUTE word onto DATA[31:0] (see Figure 13). REQ0 is received at the REQ1 pin coincident with 1 on the slave side of the interface (see Figure 14). Note that enabling and driving the data lines coincident with REQ0 may be difficult to do in FPGA technology, but it is possible.

REQ0 shall only be asserted if REQ1 was inactive during the previous control phase.

REQ0 (REQ1 at the slave) may deassert after two, four, six, or more cycles. Slave interfaces shall test REQ1 for this condition every other cycle to recognize when the transaction has been terminated.

REQ0 (REQ1 at the slave) may be deasserted before a DSTROBE (see Section 5.9.5) is asserted, resulting in a transaction where no data is transferred. This may occur if the transaction is killed, or if the master has a higherpriority task, such as refresh.

However, REQ0 shall remain asserted until any data transfers in progress complete. Thus, DSTROBE shall be inactive and all of the Read Data requested shall be received before a master deasserts REQ0.

REQ0 shall only deassert coincident with 0.

Note that while the entire ROUTE word is driven onto the data bus, only the current route field and the priority and broadcast bits of the ROUTE word need to be used by the receiving crossbar chip to select and arbitrate for the output port of that crossbar to route the transaction through.

The SHIFTED ROUTE word consists of the entire route/address field of the ROUTE word shifted to the left three bits (for example, bits [28:5] are shifted to locations [31:8].) The new contents of bits [7:5] are undefined and may change value. The contents of bits [4:0] of the ROUTE word are used unchanged as bits [4:0] of the SHIFTED ROUTE word.

Two cycles after raising REQ0, the master shall drive the SHIFTED ROUTE on DATA[31:0].

The master shall drive the SHIFTED ROUTE on DATA[31:0] for an even number of cycles (at least two cycles) until the Address phase starts (see Section 5.9.3).

Note that the SHIFTED ROUTE is passed through by the receiving crossbar chip (at least three cycles after it received REQ1) as the ROUTE word for the next Crossbar chip.

A Slave Responder may sample the 6 high order bits [31:26] of the ROUTE as high order address bits only coincident with 1.

The Slave Responder should not use the SHIFTED ROUTE.

5.9.3 Address phase: CHANGE TO ADDRESS

CHANGE TO ADDRESS and the ADDRESS word proceed through the crossbars behind the ROUTE and SHIFTED ROUTE words. A crossbar will signal CHANGE TO ADDRESS to indicate to the master side that the transaction has successfully traversed that crossbar. Transactions held pending waiting for an output port shall not return CHANGE TO ADDRESS, and the master side shall continue to drive the SHIFTED ROUTE on the data bus until receiving CHANGE TO ADDRESS.

The slave side of the interface shall drive REPLYIO low (from its inactive high-impedance state) coincident with 0 one clock edge after sampling REQ1 active.

The slave side shall assert CHANGE TO ADDRESS coincident with 1 on the REPLYIO line when it is ready to receive the ADDRESS (see Figure 15).

The master side of the interface shall drive the ADDRESS word onto DATA[31:0] coincident with 0 either two or four cycles after receiving CHANGE TO ADDRESS.

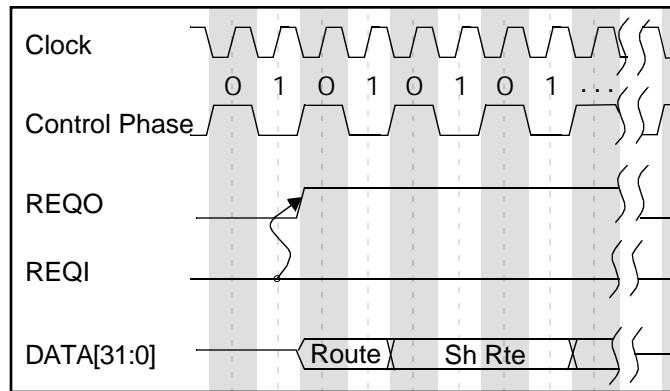


Figure 13 - REQO, Route, Shifted Route (as seen at master side initiator)

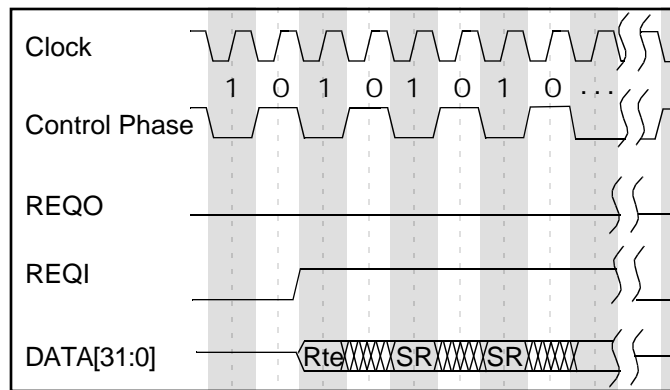


Figure 14 - REQI, Route, Shifted Route (as seen at slave responder)

The master side shall stop driving the ADDRESS one cycle after asserting the first DSTROBE (see Section 5.9.5).

The slave side of the interface may sample a valid ADDRESS beginning with the fifth clock edge after asserting CHANGE TO ADDRESS, up to the same cycle DSTROBE is received (see Section 5.9.5). This freedom does not apply to broadcast transactions, where the address is only guaranteed valid on the same cycle DSTROBE is received.

The slave should sample the Address coincident with 1.

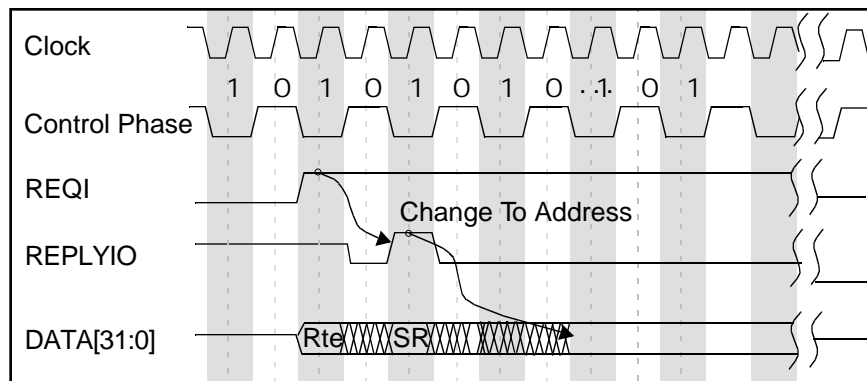


Figure 15 - Change To Address (as seen at slave side)

5.9.4 Data phase: DSEN

DSEN is used by the Slave Responder to give the Master Initiator permission to transfer or ask for data.

The slave shall assert DSEN on the REPLYIO line coincident with 0 when it is able to receive a data word (8 bytes) during a write, or when it has a data word available during a Read (see Figure 16). The master shall receive DSENs coincident with 1.

A slave may assert DSEN as early as three cycles after asserting CHANGE TO ADDRESS. At this point the slave doesn't know whether the operation is a read or a write.

If the slave needs to know whether the operation is a read or write before responding with DSEN then it shall detect that a transaction is a broadcast or wait until the Address is valid (see Section 5.9.3).

The soonest a slave performing a Split Read (see Section 5.9.15) shall be able to start asserting DSEN is seven cycles after asserting CHANGE TO ADDRESS.

The number of data words the master shall write or read shall never exceed the number of DSENs the slave sends.

The slave may throttle the master by only sending DSENs when it is able to source (for a read) or sink (for a write) data.

The slave shall not stop sending DSENs altogether, but shall continue sending DSENs (even if at a reduced rate) throughout the transaction to prevent the master from hanging waiting to transfer data.

A slave able to sink or source a continuous stream of data should generate DSENs every 0. The master may or may not respond to every DSEN sent.

The slave shall be able to handle the number of DSTROBES (see Section 5.9.5) that come from the master. For a Write, the slave shall be able to sink enough data to account for pipelined DSENs in the crossbar network. The fact that the DSENs get pipelined through the crossbar network on the way to the master means that there is a time lag between when the slave sends a DSEN and when the corresponding DSTROBE arrives (where each DSTROBE is for a read or write of an 8 byte word.) The way a slave handles this control delay (which may be caused by as many as 19 crossbar hops) is implementation specific.

The slave may control the arrival rate of DSTROBES by:

- sending DSENs at a slow rate.
- waiting a set time between groups of DSENs.
- providing buffering to compensate for the time lag (which provides improved write performance.)
- some combination of these techniques.

For a Read, the slave shall be able to count DSTROBES if it can not source the data quickly enough to account for pipelined DSENs in the crossbar network. This means that for performance the slave may send DSENs before the slave has the Read data available. However, then the slave must use a counter to keep track of how much data the master requests as a result of the DSENs.

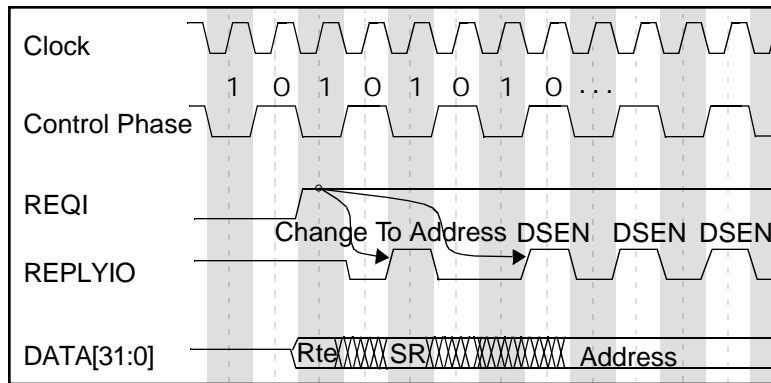


Figure 16 - DSEN (as seen at slave side)

5.9.5 Data phase: ASTROBE and DSTROBE

The master asserts ASTROBE as an indicator to the slave that a transaction is still in progress.

The master shall assert ASTROBE coincident with τ_1 starting one or three cycles after asserting REQO.

ASTROBE shall be asserted every τ_1 while a transaction is in progress (see Figure 17).

ASTROBE shall be asserted at least once after the last DSTROBE of a Read operation.

When the transaction is complete, the master shall deassert ASTROBE at least one cycle before REQO deasserts. Note that some existing implementations do not deassert ASTROBE before REQO deasserts; it is recommended that, while Masters comply, Slaves should not depend on this behavior to detect end of transfer. Note further that, to ensure compatibility with future designs, it is recommended that designers use REQI and ASTROBE to detect the end of a transfer.

The slave may use the deassertion of ASTROBE during a Read to detect when the master is finished reading. Once ASTROBE is deasserted, it shall not be reasserted.

ASTROBE is not actually used to strobe the Address. See Section 5.9.3 for information as to when the Address should be sampled from DATA[31:0].

The master asserts DSTROBE to tell the slave that Write data is coming, or to request Read data.

The master may assert DSTROBE coincident with τ_0 in response to receiving DSEN during τ_1 one cycle earlier.

DSTROBE is used during Writes to indicate that [D63:D32] of a word will be driven on DATA[31:0] on the next cycle (τ_1), and [D32:D00] of the word will be driven on DATA[31:0] during the cycle following (τ_0).

DSTROBE is used during Reads to request that the slave respond with data.

The number of DSTROBEs sent may be less than or equal to, but shall never exceed the number of DSENs received by the master.

The master may hold off or throttle the rate at which the slave responds with Read Data by throttling DSTROBE.

During Reads, the slave shall respond with data for each DSTROBE received. The slave may still throttle returning the data using READ READYs (see Section 5.9.8).

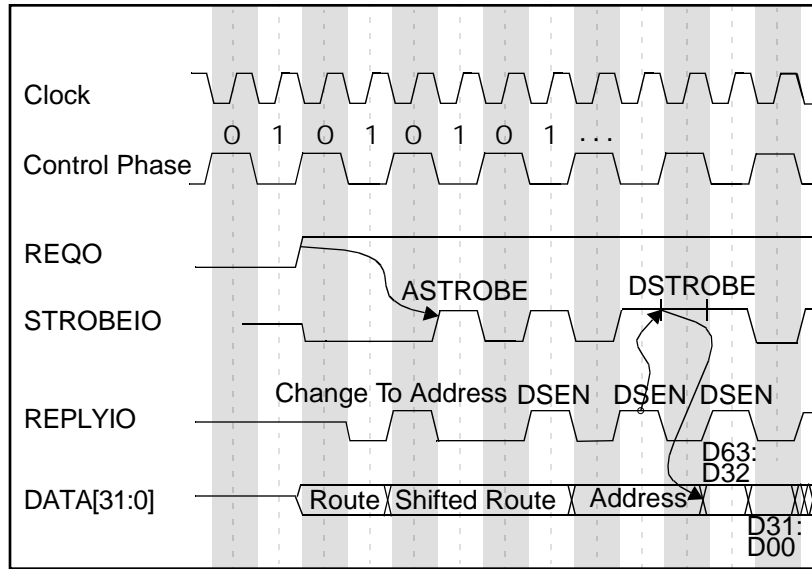


Figure 17 - ASTROBE and DSTROBE (as seen at master side)

5.9.6 Write transaction

The following sequence describes a typical write transaction from a master node to a slave node over several crossbars (see Figure 18).

The master node places the ROUTE word on DATA[31:0] while raising REQO. Coincident with the following 0 cycle, the master node places the SHIFTED ROUTE word on DATA[31:0].

The ROUTE and SHIFTED ROUTE words propagate along with REQO to the destination slave node. If the route is blocked by a higher priority transaction along the path, then the transfer remains pending as far as it got until the route becomes free.

At each crossbar along the way the most significant bits of the route word are used to determine which port of the crossbar will be selected for output. The SHIFTED ROUTE received by the crossbar will become the ROUTE word transmitted to the next crossbar (or the Slave Responder). The SHIFTED ROUTE received by a crossbar is further shifted to become the SHIFTED ROUTE output for the next crossbar (or the slave responder).

Each successfully traversed crossbar (and the slave node) responds with a CHANGE TO ADDRESS telling the sender it is ready for the Address. A blocked crossbar doesn't respond with a CHANGE TO ADDRESS, which holds up the transaction at that point. The ADDRESS word follows the ROUTE words through the crossbar network. For writes, the Read bit in the ADDRESS word is off.

When the REQO reaches the destination slave node (as REQI), a circuit through the crossbar network is established. The slave node then responds with DSEN, which propagates back to the master through the intervening crossbars. The slave should pump out DSENs every cycle if it is able to receive that many data words. Otherwise the slave may throttle back the Write transaction by only sending a DSEN when it can accept data. Note that the crossbar network can become a pipeline with several DSENs in the pipe before the master responds with data.

When the master receives a DSEN it asserts DSTROBE and then puts data on DATA[31:0] the next cycle. The data is sent in two 32 bit quantities on consecutive 40MHz clock cycles. The data propagates through the crossbar network to the destination slave node. If the DSENs are received every other clock cycle then data is written every clock without any idle cycles.

When the master is done writing, it stops asserting DSTROBE, ASTROBE and sending data. Any DSENs still coming from the slave are ignored. The master then drops REQO. The dropped REQO propagates through the crossbar network behind the data, releasing the path through the crossbars behind the last data word.

Data transfers are automatically terminated (and then re-established) at 2KByte address boundaries by the master.

A higher priority transfer may come through the crossbar network and suspend another data transfer. The suspension is done gracefully, with the suspension request in the form of Kill request propagating from the point of suspension back to the master. When the master receives a Kill request, it stops writing data and drops its REQO. This then propagates to the slave, releasing the crossbar port needed for the higher priority transaction. The original master attempts to reestablish its connection (after at least 3 cycles,) which is then reconnected up to the point where it was broken. As soon as the higher priority transaction is completed, the original connection is reestablished and the transfer continues where it left off. In order to do this, the master keeps a copy of the original route and uses an address counter to keep track of where the data transfer was suspended.

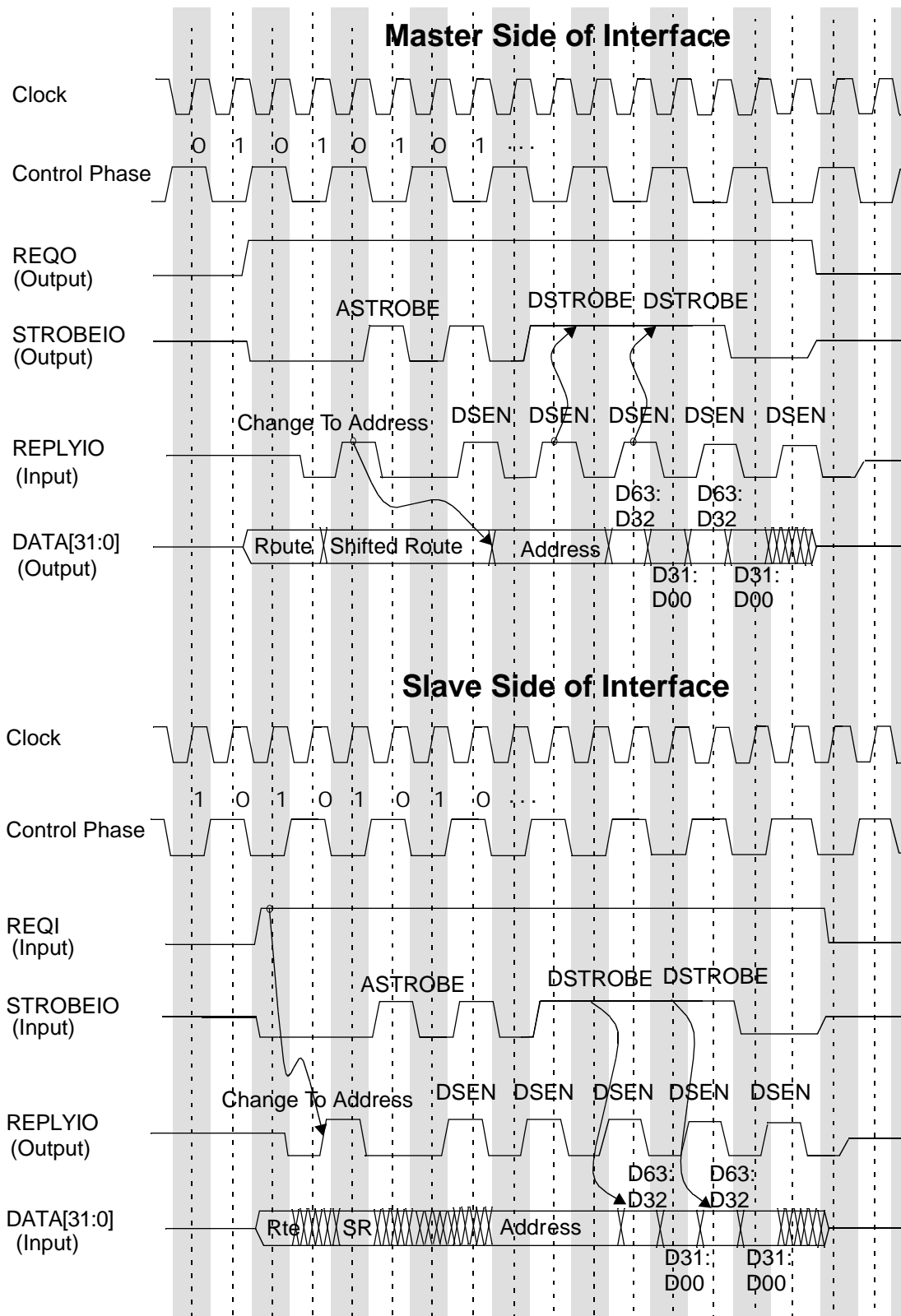


Figure 18 - Write transaction

5.9.7 Broadcast write transactions

Broadcast Write Transactions are identical to normal Write Transactions, with the only difference being that the Crossbar chips on the RACEway Interlink modules shall gather all of the responses (such as DSENs) from all of the ports selected before returning them to the master. This means that a crossbar waits until all of the ports have returned a DSEN before returning a corresponding DSEN. Thus, the broadcast does not proceed until all of the connections to the Slave Responders have been made.

Slaves shall respond to Broadcasts the same as with normal Write transactions, even if the BROADCAST ACCEPT CODE doesn't match.

Nodes being Broadcast to shall have compatible characteristics so that the transactions proceed the same way for all of the paths used with a single Broadcast. This limitation means that nodes being broadcast to may receive data at the full transfer rate for the duration of the transfer (after returning the first DSEN.) Nodes unable to sink this continuous stream of data must behave as if the BROADCAST ACCEPT CODE does not match (see Section 4.4.) These nodes can still be written to individually.

Some topologies may have constraints on which nodes can be reached at the same time using a single Broadcast.

Broadcast Writes to unoccupied slots are allowed because the RPLYIO and RDCONIO control lines are terminated with a weak pull-up to a logic high inside the crossbars. (Note that this is not shown in the timing diagrams to clearly identify the high impedance states.) A Broadcast Write transaction to an unoccupied port will therefore see all of the correct responses when the crossbar samples the RPLYIO line. This allows the Broadcast transaction to complete since it is not kept waiting for the unused port to respond.

5.9.8 Data phase during Reads: ASTROBE, DSTROBE and READ READY

During a Read, the slave node responds after REQI with DSEN, which propagates back to the master through the intervening crossbars.

The master shall send a DSTROBE after receiving a DSEN, indicating that it wants to read a word of data (64 bits) from the slave.

The master should assert only one ASTROBE after its last DSTROBE.

When the slave is ready after sampling DSTROBE, it shall assert READ READY coincident with 1 (see Figure 19.)

The slave side shall drive data on DATA[31:0] 2 and 3 clock cycles after asserting READ READY.

A slave may implement prefetching, and return more data than was requested with DSTROBES. If the master is unable to sink this additional data then it shall not insert any idle cycles between DSTROBES, but shall stop sending DSTROBE and then ASTROBE. It will then need to start a new transaction to read more data from the slave. Slaves which perform destructive reads, such as from a FIFO, do not implement prefetching. Slaves are not required to prefetch across a 2K boundary (although they may do so).

Masters may implement prefetching. If a slave does not wish to support this (eg. for register reads), it must still return as many RDYs as DSTROBES. A Master may stop sending DSTROBES if (DSTROBE count >> RDY count) in order to allow the slave to catch up.

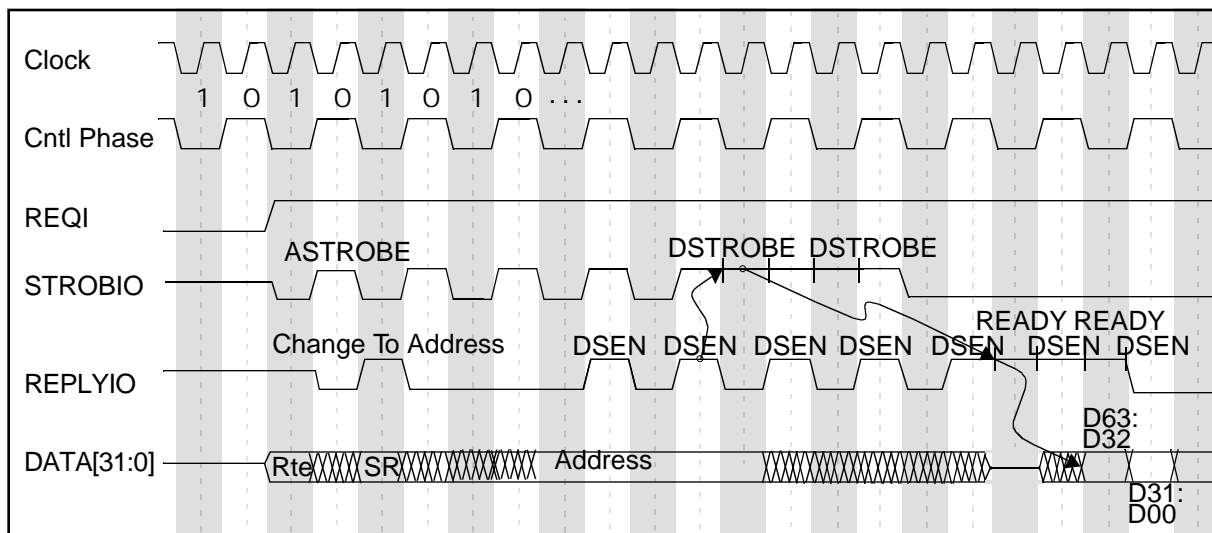


Figure 19 - ASTROBE, DSTROBE And READ READY (as seen at slave side)

5.9.9 Data phase during Reads: RDCON* and ERR

READ CONTROL (RDCON*) shall be used to tri-state the master side's DATA[31:0] drivers to always allow one idle cycle to turn around DATA[31:0].

RDCON* shall be asserted at least 2 clock cycles before, and on the same cycle as a READ READY (see Figure 20.)

The slave side shall tri-state its DATA[31:0] drivers 3 clock cycles after deasserting RDCON* (at least 5 clock cycles after asserting the last READ READY).

The master side shall tri-state its DATA[31:0] drivers 2 cycles after seeing RDCON*, or earlier, to allow at least one idle cycle to turn around DATA[31:0].

The master side may drive DATA[31:0] 4 cycles after seeing RDCON* deasserted.

During Reads, the master side may tri-state its DATA[31:0] drivers after asserting the first DSTROBE.

If an error occurred during the Read transaction, the slave shall assert ERR during 0, three cycles following the last READ READY it sends. Otherwise ERR shall be deasserted.

ERR shall be asserted on a block basis rather than for each data word. Typical designs will assert RDCON (ERR) in 0 coincident with the D64 which had the error. The RDCON signal will continue to be set in every subsequent 0. This is necessary to support error signaling when a Slave is prefetching read data.

Reading from an unconnected slot will result in the Read completing with ERR asserted (due to an internal pull-up resistor on RDCONIO at the crossbar). This may be used as a probe for configuration information.

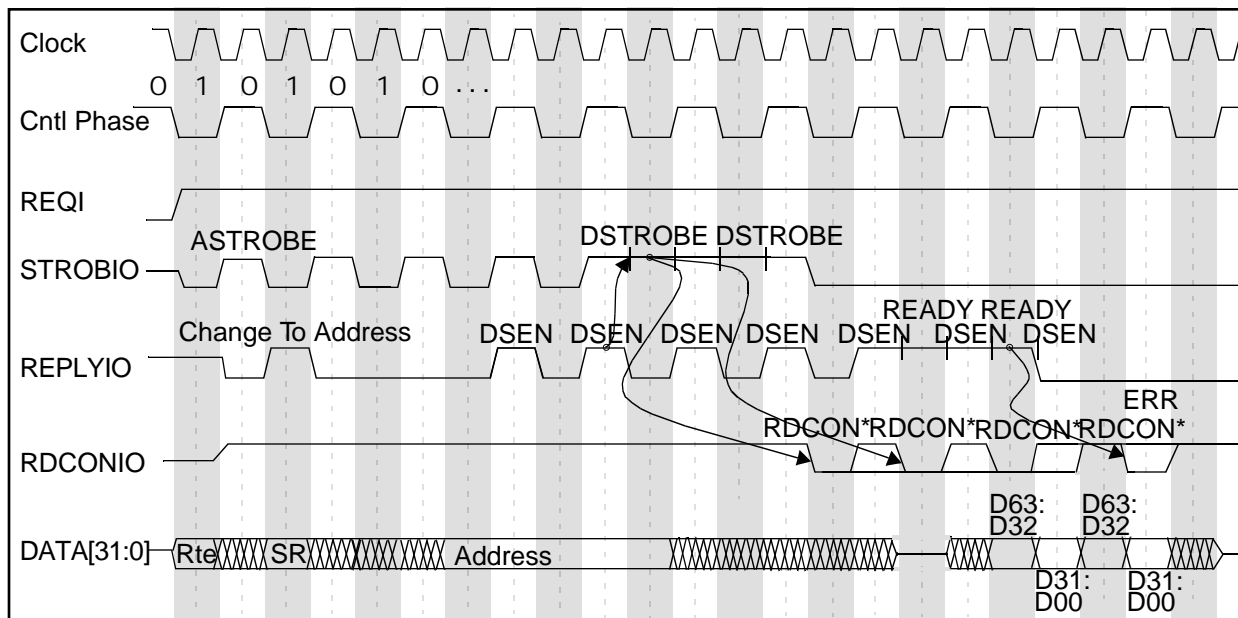


Figure 20 - RDCON* and ERR (as seen at slave side)

5.9.10 Read transaction

The following sequence describes a typical read transaction from a master node to a slave node over several crossbars (see Figure 21.)

The master node places the ROUTE word on DATA[31:0] while raising REQO. Coincident with the following 0 cycle, the master node places the SHIFTED ROUTE word on DATA[31:0].

The ROUTE and SHIFTED ROUTE words propagate along with REQO to the destination slave node. If the route is blocked by a higher priority transaction along the path, then the transfer remains pending as far as it got until the route becomes free.

Each successfully traversed crossbar (and the slave node) responds with a CHANGE TO ADDRESS telling the sender it is ready for the Address. A blocked crossbar doesn't respond with a CHANGE TO ADDRESS, which holds up the transaction at that point. The ADDRESS word follows the ROUTE words through the crossbar network. For reads, the Read bit in the ADDRESS word is on.

When the REQO reaches the destination slave node (as REQI), a circuit through the crossbar network is established. The slave node then responds with DSEN, which propagates back to the master through the intervening crossbars. The slave should generate a DSEN for every cycle if it is able to source that many data words. Otherwise the slave may throttle back the Read transaction by only sending a DSEN when it can source data. Note that the crossbar network can become a pipeline with several DSENs in the pipe before the master responds with a DSTROBE.

When the master receives a DSEN it asserts DSTROBE to request a word (8 bytes) of Read data. The master may ignore DSENs until it is ready to receive the data.

If the DSTROBEs are sent every other 40MHz cycle then data is returned every 40MHz cycle without any idle cycles. Some slaves may perform read-prefetching, and stream data back before the master requests it. If the master is unable to accept this data stream, then it may deassert ASTROBE immediately after the last DSTROBE (where it would have inserted an idle cycle.) It shall not request any more data after ASTROBEs are deasserted.

The slave asserts RDCON* to ensure that the master stops driving the data bus. This allows at least one turn-around cycle before the slave drives the data back. The slave also asserts READ READY, followed by the data two and three cycles later.

The slave is able to notify the master that the read data contained an error by asserting ERR on the last cycle of the last data word. See Section 5.9.9.

When the master is done reading, it stops asserting DSTROBE, and then ASTROBE. Any DSENs still coming from the slave are ignored. The master drops REQO only after receiving the last data it requested. The dropped REQO propagates through the crossbar network, releasing the path through the crossbars.

Data transfers are automatically terminated (and then re-established) at 2KByte address boundaries by the master.

A higher priority transfer may come through the crossbar network and suspend another data transfer. The suspension is done gracefully, with the suspension request in the form of Kill request propagating from the point of suspension back to the master. When the master receives a Kill request, it stops requesting data, waits for the already requested data to return, and drops its REQO. This then propagates to the slave, releasing the crossbar port needed for the higher priority transaction. The original master attempts to re-establish its connection (after at least 3 cycles,) which is reconnected up to the point where it was broken. As soon as the higher priority transaction is completed, the original connection is reestablished and the transfer continues where it left off. In order to do this, the master keeps a copy of the original route and uses an address counter to keep track of where the data transfer was suspended.

Some high performance slaves may implement aggressive pre-fetching, returning data as soon as DSEN is sent. This may be before the first DSTROBE is received.

5.9.11 Read transaction

Figure 21 shows a typical read transaction from a master node to a slave node over one crossbar.

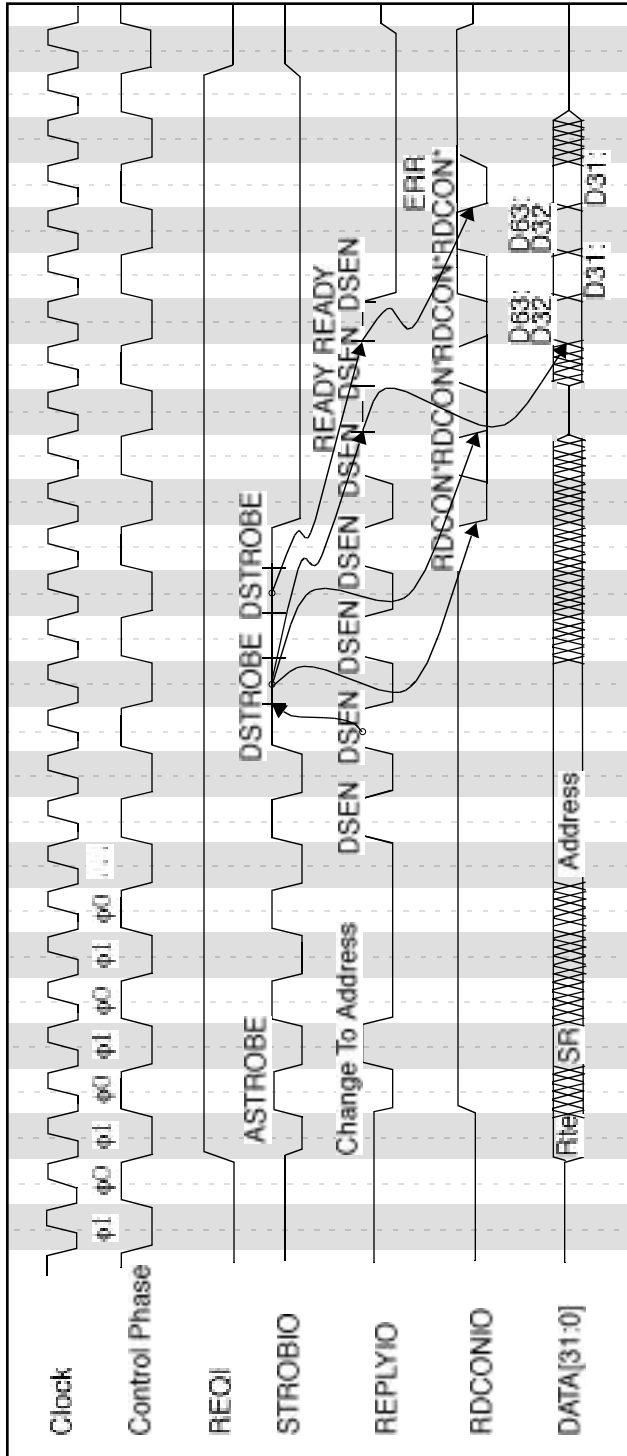


Figure 21 - Read transaction (as seen at slave side)

5.9.12 Read-Modify-Write transaction

A Read-Modify-Write transaction consists of a read, followed by turning the data bus around once more and then a write (see Figure 21.) Only one 64 bit word is transferred in both directions.

The slave is informed that it is a read-modify-write operation by the Lock bit of the Address word.

RDCON* is used by the slave to control turning the data bus around.

5.9.13 Read-Modify-Write transaction

Figure 21 shows a typical read-modify-write transaction from a master node to a slave node.

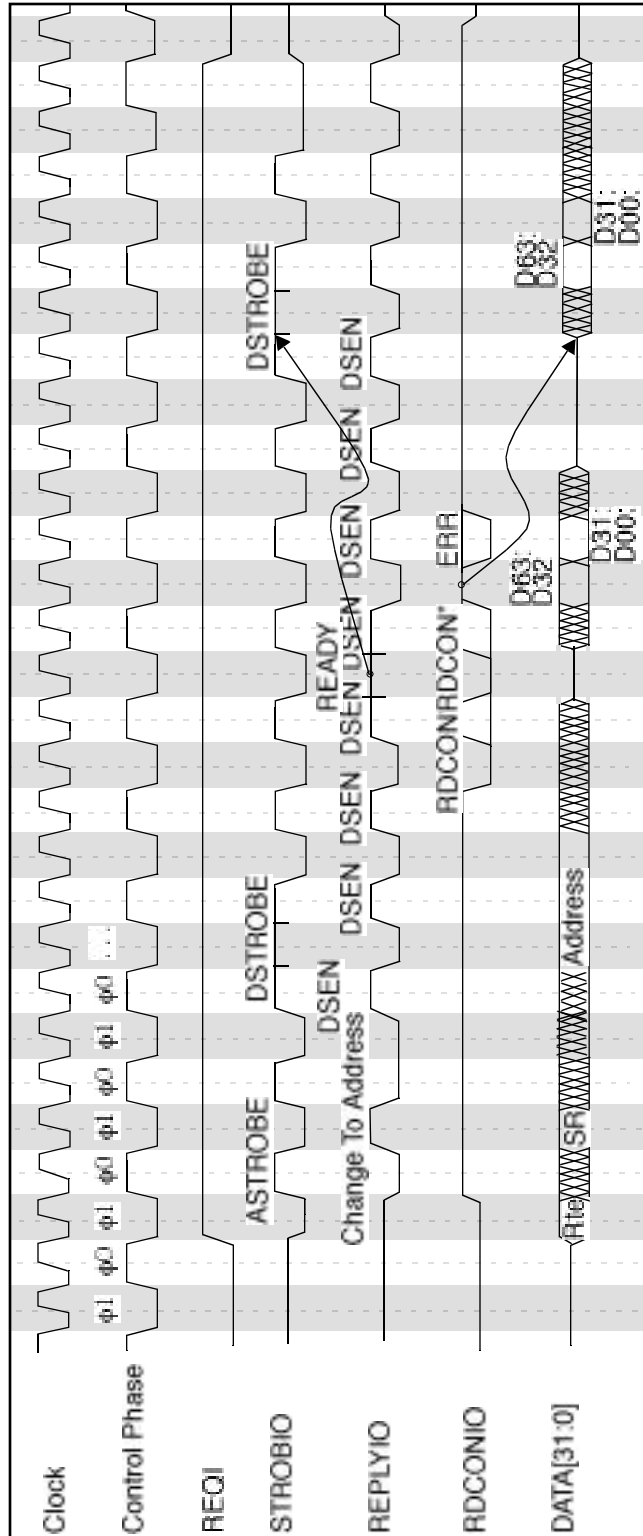


Figure 22 - Read-Modify-Write transaction (as seen at slave side)

5.9.14 Route phase: Kill Request

A crossbar shall assert REQO while REQI is active to request a KILL (see Figure 23). The KILL propagates to the master, where the master shall gracefully give up control of the established Route.

In order for a higher priority transaction to reverse the role between a master and a slave, the slave shall deassert KILL after the old master drops its REQO. After waiting two cycles the new master shall then assert REQO (REQI to the old master.) This must be done to end the KILL and start the new transaction. Figure 23 shows the signals from the old master's (new slave's) point of view.

The old master shall attempt to continue its original transaction where it left off when it is no longer a slave, or at least 4 cycles after deasserting REQO. If the new master (the one that generated the kill) does not assert REQO two cycles after deasserting it, the original master can again attempt to become master.

Slave devices shall only generate KILLS when they are able to guarantee initiating a transaction with a higher priority than the incoming transaction. This will prevent route thrashing, where two transactions spend all of their time killing each other rather than transferring data.

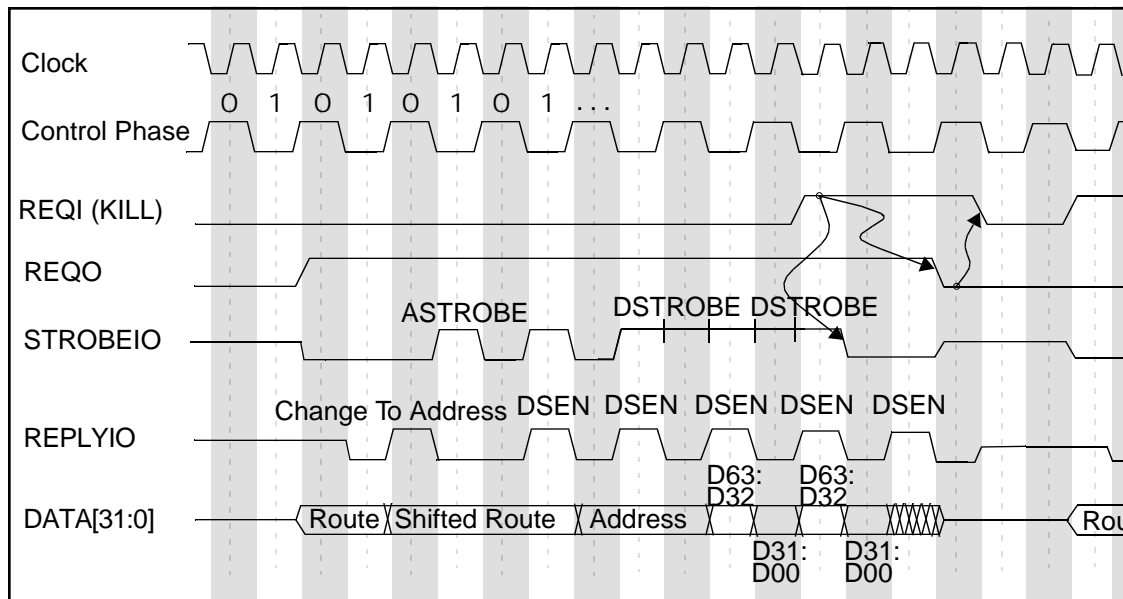


Figure 23 - KILL (as seen at master)

5.9.15 Split Read

A slave may assert SPLIT READ after getting the ADDRESS but before the first DSEN to inform the master that the read is being split. The master shall respond with the Return Route information (see Figure 24). See Section 3.2.9 for more information on Split Read capabilities.

In typical endpoint implementations, when the slave is ready, it shall write (2 minimum, 512 maximum - must not cross a 2 Kbyte boundary) words using the Return Route, with an ADDRESS of BFFF FFF9 hex. This is interpreted as the Split Read Return by the master.

This Write transaction performed for the Split Read Return is identical to a typical Write shown in Figure 18.

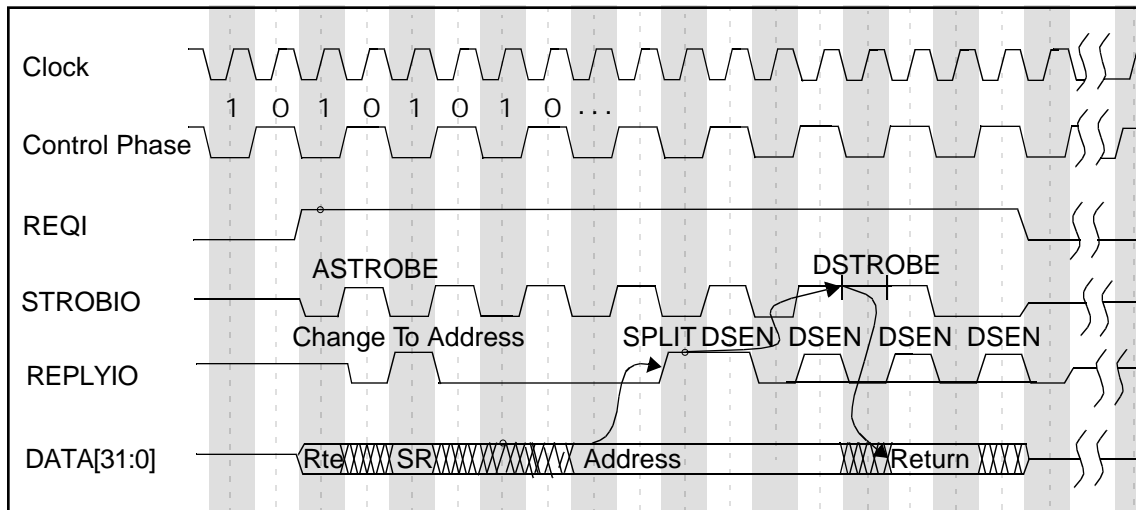


Figure 24 - Split Read (as seen at slave side)

5.9.16 Split Write

A slave may assert SPLIT WRITE after getting the ADDRESS but before the first DSEN to inform the master that the write is being split. The master shall respond with the Return Route information (see Figure 24) before the data.

Masters and slaves are not required to support split write capability. A master shall only set bit [3] while bit [0] is reset in the Route (and Shifted Route) if it supports split writes. A slave shall only split a write transaction if it supports SPLIT WRITES and bit [3]=1 and bit[0]=0 of the Route word (or shifted Route). See Sec-

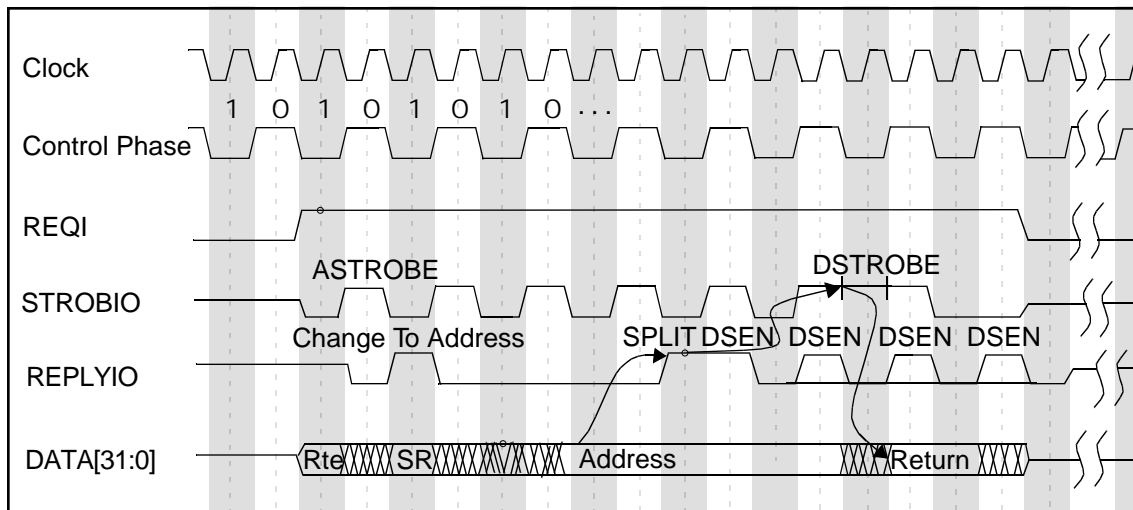


Figure 25 - Split Write (as seen at slave side)

tion 3.2.10 for more information on Split Write capabilities.

When the slave has completed its split write operation, it shall write a completion status word using the return Route with an ADDRESS of BFFF FFF9 hex. This is interpreted as the Split Write Return by the master

The Write transaction performed for the Split Write Return is identical to a typical Write shown in Figure 18.

5.9.17 Route Phase: REQO, REQI - Extended Route Option

An optional EXTENDED ROUTE word (32 bits) may be driven onto DATA[31:0] on the cycle following the ROUTE word.

The master shall then alternate driving the SHIFTED ROUTE and SHIFTED EXTENDED ROUTE words onto DATA[31:0] until the Address Phase (see Figure 26.) The format of the EXTENDED ROUTE word is reserved for future specification (see Section 4.2.)

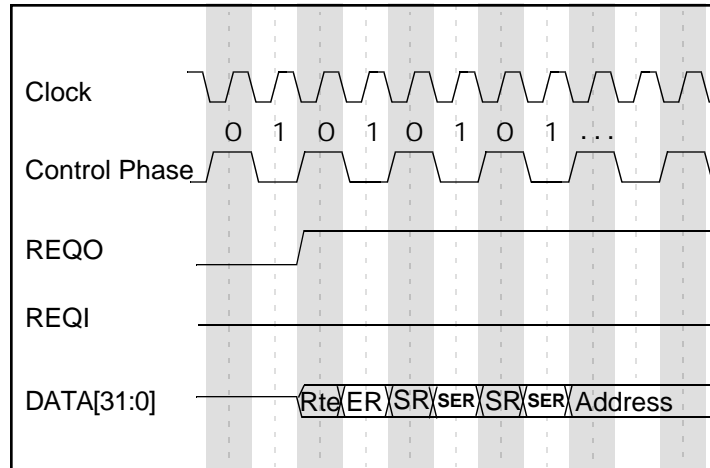


Figure 26 - Route, Extended Route, Shifted Route, Shifted Extended Route (as seen at master side)

6 Mechanical specification

Backplanes conforming to the mechanical specification given in Chapter 7 of the VME64 specification [4] and IEEE 1101 mechanical specification [6], which are equipped with wire posts on rows A, C, and the power and ground pins on row B of the P2 connector are capable of supporting RACEway Interlink compatible products.

Low insertion force may be achieved by using a two-level system. The first level is a four-slot “connector only” interposer board, which converts the high insertion force P2 connectors to low insertion force connectors. This board is plugged into the P2 wire posts. The second level contains the RACEway Crossbar(s) and support logic. It mates to the low insertion force connectors on the interposer board.

Interposer boards with five row DIN connectors may be used for compatibility with chassis which have 160 pin connectors with posts on all five rows.

Multiport Interlink Modules shall use interposers which have five row DIN connectors; these MLKs are compatible with backplanes meeting the VME64X specifications which are equipped with the wire-wrap tails on rows A, C, D and Z, and the power and ground pins of Row B of the P2 connector.

Annex A Raceway Interlink Signal Characteristics (Informative)

Tables A.1 through A.3 detail the RACEway Crossbar signal characteristics:

Worst case commercial conditions: +70°C junction temperature and $V_{CC} = 4.75V$.

Typical power per crossbar ASIC: 1.25W

Table A.1 - Driver characteristics

Driver Spec		Min.	Typ.	Max.
Z_0	@ 2.5 V	50	70	90
V_{OH}	@ -2mA	3.0 V	4.5 V	
V_{OL}	@ 4mA		.2 V	.4 V

Table A.2 - Receiver characteristics

Receiver Spec		Min.	Max.
I_{il}			.2 mA
C_{in}			10 pF
V_{IL}			1.4 V
V_{IH}		2.1 V	
V_{IH} (Clock)		2.0 V	
V_{IH} (REQI)		.7* V_{CC}	
V_{IL} (REQI)			.3* V_{CC}

Table A.3 - Signal timing

AC Spec	Min.	Typ.	Max.
Clock T_{CYCLE}	24 ns	25ns	26 ns
Clock (T_{Clhi})	10 ns	12.5ns	15 ns
Clock (T_{Cllo})	10 ns	12.5ns	15 ns
Control Setup (T_{CS})*	7.5 ns		
Control Hold (T_{CH})*	2 ns		
Control Clk to Out ($T_{CCO}=T_{CZhl}$)	3 ns	6 ns	10.5 ns
Data Setup (T_{DS})*	5 ns		
Data Hold (T_{DH})*	2 ns		
Data Clk to Out ($T_{DCO}=T_{DZhl}$)	3 ns	6 ns	10 ns
* Setup and Hold times include ± 1 ns skew at the clock input pins between the chips at the endpoints of a RACEway Interlink connection.			

A.1 Signal timing notation

Figure A.1 shows the signal timing specified in Table A.3.

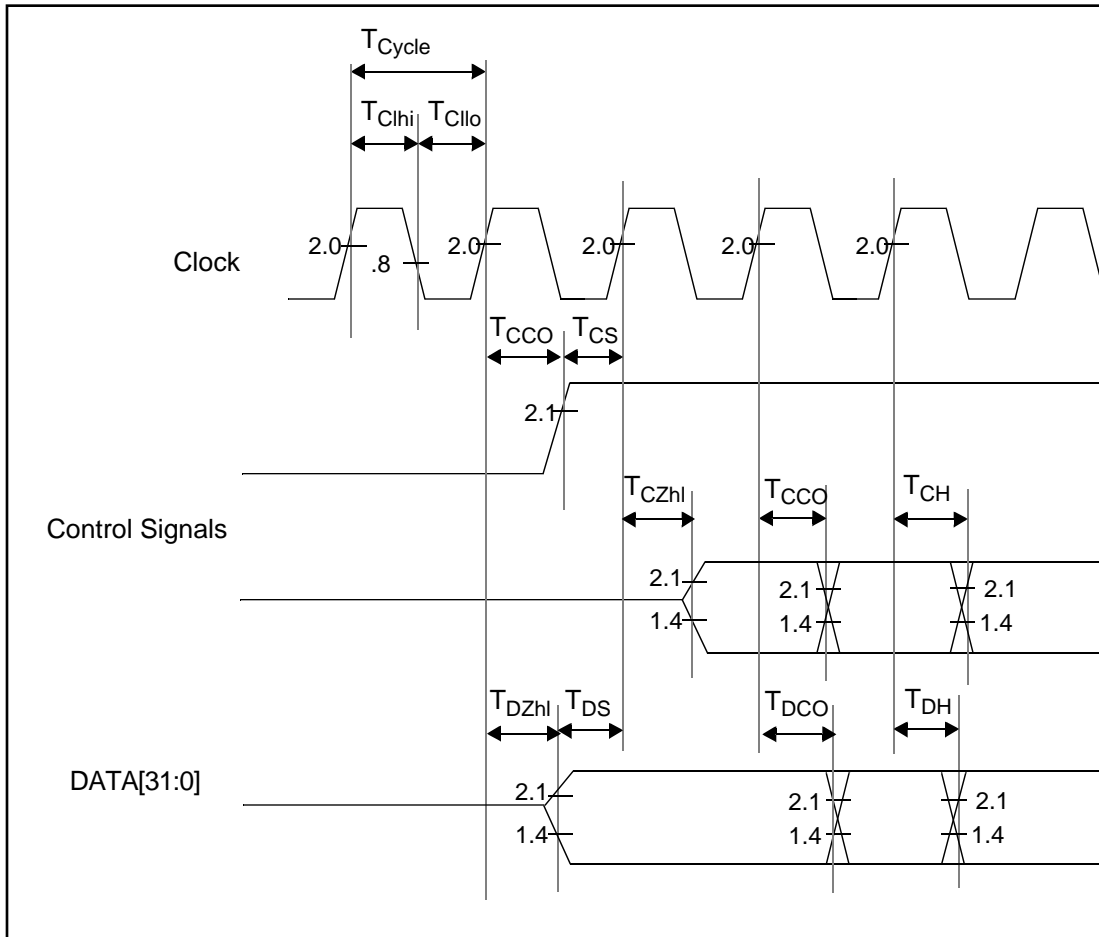


Figure A.1 - Signal timing notation