VESA®

2150 North First Street, Suite 440
San Jose, CA. 95131-2029

Phone: (408) 435-0333
FAX: (408) 435-8225

VESA VL-Bus Standard

Version: 2.0
Revision Date: November 1, 1993

Purpose

To standardize the hardware interface of peripherals to a high-speed bus. This document describes a uniform interface, architecture, timing, electrical and physical specification which allows users to interchange VL-Bus based products from various manufacturers.

Summary

This standard provides common timings, electrical and physical characteristics for the hardware designer. The VL-Bus is transparent to software; all applications should run on VL-Bus systems without modification.

The VL-Bus architecture is based on the x86 CPU local bus. Additional hardware functions have been added to the base architecture to accommodate VL-Bus master devices and interaction with the system I/O bus.

Timing specifications allow all VL-Bus devices to operate at maximum CPU bandwidth. Loading parameters define limits of VL-Bus operations without taxing the output drivers of the host processor. Physical dimensions allow motherboard and add-in board manufacturers a common set of standards to ensure physically compatible products.

Intellectual Property

Trademarks

VESA, VL-Bus	Video Electronics Standards Association
IBM, PS/2, Micro Channel Architecture	IBM Corporation
EISA	Compaq Computer Corporation
Tri-state	National Semiconductor

All other trademarks mentioned in this document are property of their respective owners.

Patents

This and other VESA documents are adopted by the Video Electronics Standards Association as standards without regard as to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Revision History

2.0	November 1, 1993
1.0	Initial release [August 28, 1992]

Table of Contents

1. Introduction

1.1. Background of the VL-Bus™ Standard

Connecting devices to a CPU local bus can dramatically increase the speed of I/O-bound peripherals with only a slight increase in cost over traditional systems. This price/performance point has created a vast market potential for local bus products. The main barrier to this market has been the lack of an accepted standard for local bus peripherals. Many motherboard and chipset manufacturers developed their own local bus implementations, but they are incompatible with each other. The Video Electronics Standards Association (VESA) VL-Bus™ specification was created to end this confusion. The committee in charge of creating the standard consists of many companies representing different segments of the industry such as manufacturers of core logic chipsets, video chips, motherboards, and other peripheral manufacturers.

1.2. Summary of VL-Bus Features

Types of CPU's Supported

The VL-Bus architecture was designed to be easily adapted from existing x86 designs. This allowed for a lower cost solution on the most commonly available platform. Support for non x86 CPU can be easily accomplished by abstracting the VL-Bus signals through a bridge chip. Thus the same peripherals can work on x86 and non x86 platforms.

Number of VL-Bus Slots

A VL-Bus design can consist of zero or more VL-Bus slots. The maximum number of slots is determined by the buffering and loading employed on the motherboard, and also by the frequency of operation. A slotless VL-Bus device would physically reside directly on the motherboard. Loading requirements allow some VL-Bus implementations to connect directly to the CPU bus without buffering address, data, and control signals. Optionally, the VL-Bus may buffer address, data, and control signals to meet the loading requirements of a multi-slot system. Bridge logic may also be employed to support additional slots.

Maximum Number of VL-Bus Masters

A maximum of three bus masters are supported on a VL-Bus subsystem in addition to the local bus controller. The maximum number of bus masters on the system I/O bus is specified by that system I/O bus. By incorporating multiple VL-Bus subsystems, more than three bus masters can be supported on the same system board.

VL-Bus Connector Type and Location

The VL-Bus connector type is a standard 16-bit Micro Channel type connector. The VL-Bus connector is located inline with a system I/O bus connector. This layout allows full use of all system I/O bus slots if a VL-Bus board is not occupying a slot. This arrangement also allows the VL-Bus add-in board access to all system I/O bus features. A board may, for example, have a VL-Bus based video controller and a system I/O bus based parallel port on the same board. Or if the VL-Bus board wants access to a feature such as the system I/O bus REFRESH signal, it is free to use that feature on the system I/O bus. Use of any signals on the inline system I/O bus connector is optional, as the VL-Bus connector has all signals needed to fully support a VL-Bus device.

Host CPU

Design is optimized for single-host CPU systems. The VL-Bus supports 386SX, 386DX, and 486-type host CPUs. Other types of host CPUs can be used if that CPU's native control signals are converted into one of the supported CPU types. Multiple-host systems should appear as single-host systems to the VL-Bus controller.

Types of VL-Bus Devices

The VL-Bus' main objective is to support high speed video controllers. Other peripherals, such as hard disk controllers, LAN adapters, etc., that can benefit from a high-speed interface may also use the VL-Bus.

Interface Speeds

The VL-Bus operates up to 66 MHz. Electrical characteristics of the physical VL-Bus connector limit the speed of a VL-Bus device operating across the connector (i.e., an add-in board) to 50 MHz. The VL-Bus clock operates at the same frequency and in phase with the CPU clock. CPUs using double-speed clocks (386-type CPUs, for example) must divide down the CPU clock before driving the VL-Bus clock. Systems that dynamically switch CPU speeds (such as portables) are supported. The system may also stop the CPU clock entirely, provided that no DMA activity occurs during the time the CPU clock is stopped.

Wait States

The VL-Bus allows for a minimum number of 1 wait states for reads and 0 or 1 wait states for write (depending on the motherboard). The VL-Bus target may add additional wait states beyond these minimums by simply delaying the assertion of the end-of-cycle signal. Bursting may be done at 3-1-1-1 for reads and 2-1-1-1 or 3-1-1-1 (depending on the motherboard) for writes. The motherboard may do 0 wait state cycles at any time.

Interface Speed Dependency

Interface protocol depends on CPU speed, but protocol selection and switching is invisible to all add-in boards, all software, and end users. The BIOS may optionally have knowledge of the presence of the VL-Bus and manipulate registers accordingly, but the VL-Bus always remains totally transparent to all application software.

Future Compatibility

If an older technology VL-Bus video board is placed into a newer technology VL-Bus backplane, the computer will continue to operate. A newer technology VL-Bus board should operate in an older VL-Bus backplane transparently.

Write-Back Cache Support

The VL-Bus supports a write back cache in the CPU by allowing the system board to maintain cache coherency by backing off the VL-Bus master when the CPU contains modified data. This function also allows a greater variety of second level cache implementations.

16-Bit Support

Optimum data bus width is 32 bits. VL-Bus target bus sizing to 16 bits is supported. A 16-bit CPU, such as the 386SX, is also supported. 16-bit peripherals are responsible for steering data to the CPU on the proper byte lanes when using the 16-bit feature on the VL-Bus.

64-Bit and Wider Support

The VL-Bus allows for expansion to a 64-bit bus. A 64-bit VL-Bus board will operate in a 32-bit VL-Bus slot as a 32-bit device. A 32-bit VL-Bus board will operate in a 64-bit slot as a 32-bit device. Physical space has also been reserved for future enhancements.

3.3 Volt Signaling

The VL-Bus Specificiation does not explicitly define 3.3 volt signaling, but the associated bus protocol can be used in a tightly controlled enfironment (i.e. the system designer is responsible for insuring that all setup and hold conditions are meet on the individual devices used in the particular architectue). No 3.3 volt expansion slots are defined.

Identifying VL-Bus Cycles

A VL-Bus slave must always identify its own cycles. If an intelligent VL-Bus controller knows (possibly via a BIOS setup routine) that a cycle will be claimed by a VL-Bus target, it may identify the cycle as claimed by a VL-Bus device before actually receiving the signal from that device.

DMA

Any memory mapped VL-Bus target may be a DMA target to the motherboard DMA or system I/O bus master. DMA initiators are not supported on the VL-Bus. For a VL-Bus device to initiate a memory transfer, it must contain full bus master capabilities.

Bus Masters

Bus mastering on both VL-Bus or system I/O bus are fully supported. An active bus master and the target can both sit on the VL-Bus and transfer information to each other.

Software Compatibility

The VL-Bus is totally transparent to any BIOS or application software. Configuration of the interface is controlled entirely by hardware. Configuration of the VL-Bus device itself is accomplished in the same manner as if that device was a system I/O bus device.

1.3. Terms and Definitions

VL-Bus

VESA local bus. An architectural, timing, electrical and physical interface that allows high-speed peripheral devices to interface, either directly or indirectly, to the local bus of a host CPU. Cached systems can be interfaced either directly or indirectly to the host CPU through the cache controller. The specification of the VL-Bus is the subject of this document.

Local Bus

The set of address, data, and control signals that are directly connected to the host CPU. For simple systems, the local bus and the VL-Bus may actually be the same bus. For more complex systems, the VL-Bus may be a buffered version of the local bus. For cached systems, the VL-Bus may interface indirectly to the host CPU through a cache controller. Hosts other than 386- or 486-class CPUs may require control translation circuitry between the host CPU local bus and the VL-Bus.

Host CPU

The device that owns the VL-Bus when no other VL-Bus master is requesting the VL-Bus. The host CPU may talk directly to any VL-Bus target without arbitrating for bus control if no other LBM is active.

VL-Bus Device

Any device other than the host CPU that is connected to the VL-Bus is considered a VL-Bus device. A VL-Bus device can be a VL-Bus controller, VL-Bus master, or VL-Bus target. Any 386- or 486-type local bus device that adheres to the VL-Bus specification can be considered a VL-Bus device.

LBC

VL-Bus local bus controller. The LBC controls accesses on the VL-Bus and resolves bus master arbitration between the CPU and VL-Bus masters. The LBC physically resides on the motherboard and normally is integrated inside the system chipset. The VL-Bus controller logically sits between the system DRAM controller and the system I/O bus controller. For purposes of this document the host CPU is considered part of the local bus controller.

LBM

VL-Bus local bus master. A device connected to the VL-Bus that has the capability of initiating transfers on the bus. A LBM must arbitrate with the VL-Bus controller (and hence the host CPU) for bus ownership.

LBT

VL-Bus local bus target. A device connected to the VL-Bus that answers requests for transfers initiated elsewhere in the system is considered a LBT.

System I/O Bus

The current public add-in buses available for microcomputers. Industry Standard Architecture (ISA), Expanded Industry Standard Architecture (EISA), and Micro Channel Architecture buses are all examples of system I/O bus architectures.

System I/O Bus Master

Any system I/O bus device with bus master capability on the system I/O bus. System I/O bus masters may target any VL-Bus device.

DMA Target

A memory device that a DMA INITIATOR or a bus master is addressing for a memory data transfer. The DMA target is not involved in initiating the transfer but merely answering the memory transfer request of the initiator. Any memory mapped VL-Bus device can be a DMA target.

DMA Initiator

Any device that uses the motherboard DMA facilities to initiate a data transfer between itself and a memory DMA target. The standard ISA bus floppy disk controller is an example of a DMA initiator. DMA initiators are not supported on the VL-Bus. For a VL-Bus device to initiate DMA transfers, it must contain full VL-Bus master capabilities.

1.4. Maximum Number of VL-Bus Devices and Slots

The maximum number of VL-Bus and devices is determined by the loading. There is a capacitance budget of 125pF for address, data, some control signals, and 100pF for the remaining signals for systems running at 40MHz and below. There is also a capacitance budget of 75pF for all signals for systems running at above 40MHz. Each adapter card is budgeted 25pF. The loading on the system board must be added in before determining the allowable number of slots. VL-Bus peripherals on the motherboard must be counted also, at their actual values.

Frequency of VL-Bus Connectors	System Board Load (ADRn, BEn#, DATn, ADS#, W/R, D/C, M/IO)	System Board Load (all other signals)	Maximum number of slots
Up to 40MHz	Up to 50pF	Up to 25pF	3
Up to 40MHz	Up to 75pF	Up to 50pF	2
Up to 50MHz	Up to 25pF	Up to 25pF	2
Up to 50MHz	Up to 50pF	Up to 50pF	1

Table 1.1. CPU Speed vs. Recommended Maximum Slots.

Additional slots may be implemented by isolating multiple VL-Buses through the use of bridge logic.

1.5. CPU Speeds Supported

The VL-Bus can support any CPU clock frequency. In a system designed to support higher CPU speeds than the VL-Bus connector allows (see Table 1.2 below), the signals in the VL-Bus connector must be divided to achieve a legal frequency for the connector.

VL-Bus Frequency	Theoretical Bandwidth (32 Bit)	Theoretical Bandwidth (64 Bit)
25 MHz	80 MB/s	133 MB/s
33 MHz	107 MB/s	178 MB/s
40 MHz	128 MB/s	213 MB/s
50 MHz	160 MB/s	267 MB/s

Table 1.2. Example Connector Speeds and Bandwidth

1.6. Block Diagram

Figure 1.1 illustrates the logical flow of information from the host CPU to peripheral devices. If one module accepts the current transfer as its own, control of that transfer is not passed to the next module. Note that this diagram depicts logical flow only; physical and electrical connections may differ from the logical flow. Although address and data buses may be shared between several modules, a lower priority module does not claim ownership if a higher priority module claims ownership.

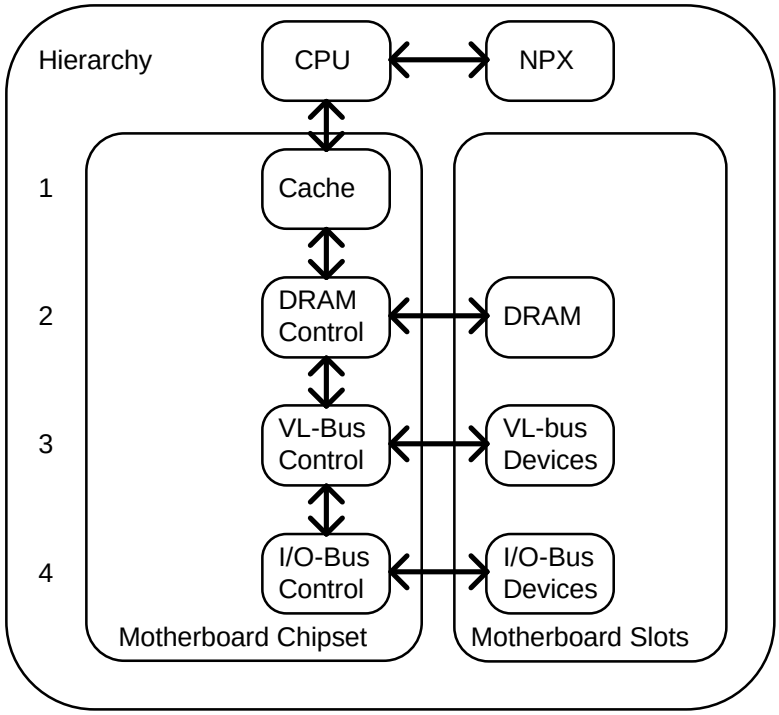


Figure 1.1. VL-Bus Interface Block Diagram.

1.7. Difference Between the VL-Bus and 486 Local Bus

The VL-Bus specification is modeled after the 486-type CPU. There are, however, some differences worth noting. These deviations were made to either make the specification more robust, to also accommodate a 386-type CPU, or to streamline the operation of VL-Bus targets.

1.7.1. 16-Bit Mode

BS16# vs. LBS16#

386 and 486 type processors handle the second 16-bit transfer generated by BS16# differently. Since the VL-Bus supports both types of processors, a superset of both schemes must be used for 16-bit VL-Bus targets using LBS16# on the VL-Bus. Active VL-Bus masters should follow the 486 scheme when splitting a cycle when the target VL-Bus device has asserted LBS16#. See the 16-bit section for further details.

1.7.2. Local Bus Arbitration Methods

HOLD, HLDA, BREQ, BOFF# vs. LREQ# and LGNT#

The VL-Bus utilizes a simple Request/Grant type handshake to arbitrate the ownership of the VL-Bus. A VL-Bus master requesting the bus asserts LREQ#. The VL-Bus controller asserts LGNT# to grant the requesting master bus ownership. The 486 uses the signals HOLD, HLDA, BREQ, and BOFF# for bus arbitration.

1.7.3. System Reset

RESET#

The RESET# pin defined on the VL-Bus is an active low signal that goes active when the entire system is being reset. It normally goes active only at power up or the pressing of a system reset button. It does not carry any phase information as the 386 (and sometimes 486) reset signal does. RESET# should be used for initializing state machines to a known state only.

1.7.4. Cycle Termination

RDY# vs. LRDY# and RDYRTN#

A VL-Bus target can terminate a transfer by asserting LRDY#. The termination is acknowledged by the VL-Bus controller by asserting RDYRTN#. RDYRTN# is typically the same signal as the 486 RDY#. VL-Bus targets are required to keep read data valid until RDYRTN# is sampled active to guarantee that the current master has read the data. The main purpose of this is DMA reads, where RDYRTN# is delayed until the DMA initiator has read the data. When generating BRDY# instead of LRDY#, a target should not wait for RDYRTN#. However, there are times when a VL-Target must generate LRDY# instead of BRDY#. See section 3.6 for details. A cycle started with ADS# might not be terminated with either RDYRTN# or BRDY# (specifically in the case of a CPU with a write back cache. See sections 2.1.5 and 9.2.9 for details).

2. VL-Bus Signals

2.1. VL-Bus Connector Signals From the System Logic

2.1.1. Power, Ground, and Reserved Pins

Every VL-Bus Device must connect to all power and ground pins. Power must be drawn equally from all power pins. The adapter board may optionally connect to the compatibility bus power and ground pins also. To ensure future compatibility, reserved pins must not be connected to any signal on either the VL-Bus controller (i.e., the motherboard) or any VL-Bus add-in boards.

2.1.2. RESET# - System Reset

This low-asserted signal is a master reset that is asserted after system power up and before any valid CPU cycles take place. RESET# is always driven by the system logic or the VL-Bus controller to all VL-Bus masters and targets. RESET# places all devices at a known state before execution begins. Unlike the 386 CPU reset, there is no guaranteed relationship between the rising or falling edges of RESET# and the phase of LCLK. While RESET# is active, VL devices should float all signals, with the possible exception of LDEV# and LREQ#, which may be driven inactive. LDEV# may be driven from the address decode during RESET# and need not be specifically driven high with RESET#. RESET# may be asynchronous to LCLK. Devices should float their buses within 16 LCLKs of RESET# going active. VL-Bus cards should use RESET# from the VL-Bus rather than RSTDRV from the ISA bus (in EISA systems, the ISA RSTDRV signal may go active at times other than system reset.).

2.1.3. LCLK - Local CPU Clock

The LCLK VL-Bus clock signal is a 1x clock that follows the same phase as a 486-type CPU. LCLK is always driven by the system logic or the VL-Bus controller to all VL-Bus masters and targets. Its frequency range is up to 66 MHz, but is limited to 50MHz if an expansion slot is provided (see Table 1.2). LCLK is allowed to dynamically change frequencies if the system logic supports a dynamically changing clock. In the case of a 386 or other CPUs that use a clock running at 2x, the main clock must be divided down to a 1x clock. The rising edge of the clock signifies the change of CPU states.

2.1.4. RDYRTN# - Ready Return

RDYRTN# establishes a handshake so the VL-Bus target knows when the cycle has ended. Usually RDYRTN# is equivalent to the RDY# signal that is tied directly to the CPU or cache controller. RDYRTN# is always driven by the VL-Bus controller to all VL-Bus masters and targets. For LCLK speeds up to 33 MHz, RDYRTN# is typically asserted in the same LCLK cycle as LRDY# is asserted. At higher LCLK speeds RDYRTN# may trail LRDY# by one LCLK cycle due to signal resynchronization. During DMA or system I/O bus master signals, the VL-Bus controller asserts RDYRTN# for one LCLK cycle when the DMA or system I/O bus master command ends. VL-Bus Masters should treat this signal exactly as the 486 treats RDY#. It will generally not be asserted following BRDY#.

2.1.5. WBACK# - Write Back

WBACK# is an output of the VL-Bus controller used to maintain cache coherency in systems that have a cache structure that requires this function. An example of this is a system with a CPU containing a write back cache. The VL-Bus controller may assert WBACK# at any time after an ADS# is issued and before or coincident with the first ready (either RDYRTN# or BRDY#) of that access. When an active VL-Master samples WBACK# asserted, it must

immediately abort the bus cycle and float all address, data, and control signals that it drives as master. When WBACK# is sampled inactive, the VL-Master restarts the bus cycle with a new ADS#. If a ready was returned at the same time as WBACK# was sampled active, the ready (as well as the data on a read) should be ignored. WBACK# may be generated on either a read or a write and is synchronous to LCLK. It should always be sampled on a rising clock edge, and not be used asynchronously.

2.1.6. ID<4..0> - Identifier pins

The identifier pins allow any VL-Bus target to identify the type and speed of the host CPU. The ID pins typically are static levels and do not change values while the system is in operation; however, the ID pins are defined as valid only during power on reset (i.e., while RESET# is asserted) and should be latched at the trailing edge of RESET#.

High Speed Write	ID2
VL Targets may do 0 wait state write	1
VL Targets must do minimum 1 wait state write	0

CPU Speed	ID3
<= 33MHz	1
>33MHz	0

Selection	ID4	ID1	ID0	References
RESERVED	X	0	0	
386: No bursting	0	0	1	See noteA,B
486: May support bursting (1.0 motherboard)	0	1	0	See note A,C
486: Supports burst protocol for reads	1	1	0	See note A,D
486: Supports burst protocol for reads and writes	1	1	1	See note A,E

TABLE 2.1. Local Bus ID Selection

Note A:

- The CPU Type Identifier bits describe the host CPU type. "386" describes a 386SX, 386DX or compatible processor. "486" describes a 486SX, 486DX, 486DX2 or compatible processor. Other CPUs indicate the CPU protocol which is emulated by the bridge, and usually indicate a 486 CPU type.

Note B:

- Motherboard: Uses this setting for 386SX and DX CPUs Must drive BLAST# low at all time when it owns the bus. This will cause the targets to drive LRDY#.
- Targets: Must drive LRDY# to terminate transfers (this is a fallout of BLAST# always being low).
- Masters: Must not attempt to burst. Should drive BLAST# low at all times.

Note C:

- Motherboard: 1.0 motherboards use this setting for all types of 486 CPU and bridges from other CPUs.
- Targets: May burst.
- Masters: Bursting not recommended, unless it is known that the motherboard supports it.

Note D:

- Motherboard: Uses this setting for all types of 486 CPUs and bridges from other CPUs. The motherboard must use this setting if it cannot accept multi-dword burst writes (I.E. it drives BRDY# for write cycles and assumes a single transfer, as the 486 will provide).
- Targets: May burst reads or writes.
- Masters: May burst for read cycles. Burst write attempts are not advised since the motherboard may not support them (the motherboard may return a single BRDY# on a burst write attempt).

Note E:

- Motherboard: Uses this setting for all types of 486 CPUs and bridges from other CPUs. This setting may be used if the motherboard can accept multi-dword burst write attempts. Note that the motherboard does not necessarily have to burst writes; but if it doesn't, it must drive RDYRTN# when BLAST# is high. The motherboard must also be 64 bit aware to use this setting. This means that it floats ID bits 0, 1, and 4 following RESET to allow these pins to be used for 64 bit transfers. The motherboard need not actually do 64 bit transfers.
- Targets: May burst reads or writes. Must be tolerant of multi dword burst writes (need not burst writes, but must support the protocol properly by driving LRDY# when BLAST# is high if it does not wish to burst.)
- Masters: May burst reads or writes.

The High Speed Write Identifier bit is set if the VL-Bus controller is capable of handling a high speed zero wait state write transfers. Many motherboards with cache drive the CPU RDY# and BRDY# signal during the first T2 state. Since the motherboard cache and the VL-Bus controller cannot simultaneously drive CPU RDY#, the VL-Bus controller must wait until the second T2 when the cache is no longer driving CPU RDY# and BRDY#. If a cache does not drive CPU RDY# or BRDY# during the first T2 state, the High Speed Write bit is set because the VL-Bus controller may complete a write transfer with zero wait states. If the VL-Bus target cannot complete a write in zero wait states it may ignore this bit and default to a minimum of one wait states. Read transfers are unaffected by the High Speed Write configuration bit setting. A VL-Master must be capable of accepting a zero wait state write at any time, since the motherboard may produce zero wait state writes regardless of the ID2 setting.

The CPU Speed Identifier describes the maximum clock speed of the CPU. In the case of systems that can dynamically change the CPU speed (portables, for example), the speed is defined as the maximum speed of the CPU. A system with a clock speed of 33.3 MHz is considered 33 MHz.

2.2. VL-Bus Connector Signals From the CPU

2.2.1. ADR<31..02> - Address Bus

The address bus furnishes the physical memory or I/O port addresses to the VL-Bus target. On the motherboard, ADR<31..02> can either directly connect to the CPU address bus or be tri-state buffered. ADR<31..02> is driven by the CPU for all CPU initiated transfers. During system I/O bus master or DMA cycles the VL-Bus controller drives system I/O bus addresses back to ADR<31..02>. During VL-Bus master cycles the VL-Bus device acting as bus master drives the address bus. If no VL-Bus target claims the transfer, the VL-Bus controller drives the VL-Bus master address to the system I/O bus. VL-Bus targets should decode all address bits through A31 for memory cycles and through A15 for I/O cycles. It should be noted when selecting memory addresses for VL-Targets, that many motherboards do not decode some of the upper address bits for local memory. Main memory may wrap at a 128M or even 64M boundary. Special cycles which may be seen as proper memory decodes by VL-Bus targets (such as SMM access to the video buffer area), should be disguised by setting high an upper address bit which is unused by the system logic, or by using an address which will not be decoded by a VL Target.

2.2.2. DAT<31..00> - Data Bus

This is a bi-directional data path between VL-Bus devices and the CPU. On the motherboard DAT<31..00> can either be directly connected to the CPU data bus or buffered. During read transfers the active VL-Bus target drives data onto DAT<31..00>. If the read is initiated from a system I/O bus master or motherboard DMA, the data is driven onto the system I/O bus data bus by the VL-Bus controller. During write transfers the CPU, DMA initiator, or bus master drives data onto the data bus. Byte enables BE<3..0># (see below) determine which byte lane(s) of DAT<31..00> are valid. VL-Targets should not drive the data bus during the first T2 of read cycles. Secondary caches will often drive the data during this time even if the address is non-cacheable.

2.2.3. BE<3..0># - Byte Enables

The byte enables indicate which byte lanes of the 32-bit data bus are involved with the current VL-Bus transfer. On the motherboard, BE<3..0># can either be directly connected to CPU BE<3..0># or tri-state buffered. A 386SX class host CPU must translate A1, BLE#(A0) and BHE# the proper byte enable format before driving BE<3..0># to the VL-Bus targets. BE<3..0># is driven by the CPU for all CPU initiated transfers. During system I/O bus master or DMA cycles the VL-Bus controller drives BE<3..0># according to the values of system I/O bus address lines 0, 1 and SBHE#. During VL-Bus master transfers the VL-Bus device acting as bus master drives BE<3..0># to indicate which byte lanes contain valid data.

2.2.4. M/IO# - Memory or I/O Status

This CPU output indicates the type of access currently executing on the VL-Bus. On the motherboard, M/IO# can either be directly connected to the CPU M/IO# or tri-state buffered. A memory cycle is indicated by M/IO# high, while an I/O cycle is indicated by M/IO# low. M/IO# is driven by the CPU for all CPU initiated transfers. During system I/O bus master or DMA cycles the VL-Bus controller drives M/IO# according to the values of system I/O bus signals MEMR#, MEMW#, IOR#, and IOW#. During VL-Bus master transfers the VL-Bus device acting as bus master drives M/IO#.

2.2.5. W/R# - Write or Read Status

This CPU output indicates the type of access currently executing on the VL-Bus. On the motherboard, W/R# can either be directly connected to the CPU W/R# or tri-state buffered. A write access is indicated by W/R# high, while a read access is indicated by W/R# low. W/R# is driven by the CPU for all CPU-initiated transfers. During system I/O bus master or DMA cycles the VL-Bus controller drives W/R# according to the values of system I/O bus signals MEMR#, MEMW#, IOR#, and IOW#. During VL-Bus transfers the VL-Bus device acting as bus master drives W/R#.

2.2.6. D/C# - Data or Code Status

The data/code status signal indicates whether the current cycle is transferring data or code. D/C# is useful to devices employing separate data and code caches. On the motherboard D/C# can either be directly connected to the CPU D/C# or tri-state buffered. During VL-Bus master transfers the VL-Bus device acting as bus master drives D/C#. If the VL-Bus master does not differentiate between data and code, it must drive this signal to the default data state (high) whenever it owns the VL-Bus. VL-Bus targets need not decode D/C# for I/O accesses if they do not contain I/O ports in the 0000-001F range or memory accesses if they do not contain memory in the 00000000-0000001F range. No VL-Bus targets are expected to contain these addresses, since they are a standard part of the system board. The CPU will produce addresses in these areas for interrupt acknowledge cycles and special cycles (halt, shutdown, etc.).

M/IO#	D/C#	W/R#	Function	VL-Target may respond	VL-Master may generate
0	0	0	Interrupt Acknowledge	No *	No
0	0	1	Halt/Special (486)	No *	No
0	1	0	I/O read	Yes	Yes
0	1	1	I/O write	Yes	Yes
1	0	0	Code read	Yes	Yes
1	0	1	Halt/Shutdown (386)	No *	No
1	1	0	Memory read	Yes	Yes
1	1	1	Memory write	Yes	Yes

* D/C# decoding is usually not required. See the D/C# pin description for details.

TABLE 2.2. VL-Target/Master

2.2.7. BLAST# - Burst Last

The BLAST# signal indicates that the next time BRDY# is asserted the burst cycle will complete. On the motherboard BLAST# can either be directly connected to the CPU BLAST# or tri-state buffered. 386SX and DX class host CPUs must drive this signal low whenever the host controls the VL-Bus. During VL-Bus master transfers the VL-Bus device acting as bus master drives BLAST#. A VL-Bus master that does not support burst transfers must drive this signal low whenever it controls the VL-Bus. A VL-Target must use LRDY# to terminate a cycle when the DMA controller has the bus (see section 3.6 for details on determining when the DMA controller owns the bus).

2.2.8. ADS# - Address Data Strobe

ADS# indicates the start of the VL-Bus cycle. It is always strobed low for one clock period (386 pipelined timing is not supported where ADS# remains low for multiple T states). The address and status lines will be stable on the T state boundary where ADS# is sampled low,

having become valid during the previous cycle (the same T state as ADS# is driven low), or in some cases, having already been valid. ADS# is driven by the current bus owner, which may be the CPU, VL-Master, or the VL-Bus controller (for DMA or system I/O bus masters).

2.3. VL-Bus Connector Signals From the VL-Bus Controller

2.3.1. LEADS# - Local External Address Data Strobe

The VL-Bus controller or active VL-Bus master asserts this signal whenever an address is present on the VL-Bus that performs a CPU cache invalidation cycle. A VL-Bus master must drive this signal while it owns the bus. LEADS# is not active for CPU cycles. LEADS# is driven by local bus masters at the same time ADS# is driven for both memory read and write cycles. It must not be driven low for I/O cycles since this could cause write backs from the CPU during I/O cycles.

2.3.2. LGNT<x># - Local Bus Grant

LGNT<x># is used in conjunction with LREQ<x># to establish a VL-Bus bus arbitration protocol. When the VL-Bus device asserts LREQ<x>#, the VL-Bus controller responds by asserting LGNT<x># for that slot. The active VL-Bus master then has control of the VL-Bus and may own the bus until it no longer needs the bus or until the VL-Bus controller removes LGNT<x># to preempt the active VL-Bus master. There is one pair of LREQ# and LGNT# signals per slot.

2.4. VL-Bus Connector Signals From the VL-Bus Target

2.4.1. LDEV<x># - Local device

LDEV# is an output of a VL-Bus target indicating that the current address on the bus is addressing the target. LDEV# is an asynchronous decode of the address, M/IO#, and in a few cases W/R#. It should not be latched by ADS# or the clock. The target has 20nS following any address, M/IO#, or W/R# (where W/R# is part of the decode) change to produce a valid LDEV# (either high or low). When M/IO# is high, the target should decode from A31 down through as many address lines necessary to uniquely decode the memory space(s). When M/IO# is low, the target should decode from A15 down through as many address lines necessary to uniquely decode the I/O space(s). A target may decode several memory and/or IO spaces which are not necessarily contiguous. There is a unique LDEV# signal for each slot or device, which should be driven with a totem pole driver by the target at all times (LDEV# must be actively driven high to avoid rise time problems). W/R# is generally NOT used in the LDEV# decode. It must be used in a few cases, such as the VGA palette, where reads should generate LDEV# but writes should not.

The VL-Bus controller combines the LDEV# from all devices together, usually with an AND gate (a logical OR function). When the CPU or a VL-Master owns the bus, the VL-Bus controller samples LDEV# to determine whether to send the bus cycle to the system I/O bus (the default target if no other device claims it). It usually does this at the end of the first or second T2, but the only requirement is that it allow the VL-Targets 20ns to perform their decode. The VL-Bus controller must take into account any address buffering and/or LDEV# combining logic. During DMA or system I/O bus Master accesses the VL-Bus controller samples LDEV# to determine whether the access should be sent to the VL-Bus. The VL-Bus controller may ignore LDEV# when there is a cache hit or local memory controller hit. The system board contains an 8K to 20K pull up resistor on each LDEV# signal to keep it high when no device is present.

LDEV# must not be driven low during interrupt acknowledge cycles and special cycles. For the purpose of this spec., these cycles are limited to the address range of 0000-001F for I/O and 00000000-0000001F for memory. Targets may ignore D/C# in their decoding if they will not respond to addresses in these range.

2.4.2. LRDY# - Local Ready

LRDY# begins the handshake that terminates the current active bus cycle when the target is not bursting. LRDY# is shared among all VL-Bus devices. The active LBT drives this LRDY# only during the time of the cycle that it has claimed as its own. While the VL-Bus is inactive, LRDY# is held high by a 8k to 20k pull-up resistor located on the motherboard. LRDY# must be asserted low for one LCLK period, but may optionally remain low until RDYRTN# is sampled low. LRDY# is driven high before being released. LRDY# is synchronized to LCLK so appropriate setup and hold times to LCLK must be satisfied. In most cases, LRDY# should not be driven during the first T2 period because the system cache controller may be driving it. However, if the High Speed Write configuration bit is set the LBT may start driving LRDY# during the first T2 state of a write cycle.

2.4.3. LBS16# - Local Bus Size 16

The local bus size 16 signal forces the CPU or LBM to run multiple 16-bit transfers to a VL-Bus target that cannot accept 32 bits of data in a single clock cycle. LBS16# is shared among all VL-Bus targets and must only be asserted by the active VL-Bus target. LBS16# must be asserted one clock cycle before LRDY# and must be held until RDYRTN# is sampled as active. LBS16# then is driven high before being released. While the VL-Bus is inactive, LBS16# is held high by a 8k to 20k pull-up resistor located on the motherboard.

2.4.4. BRDY# - Burst Ready

BRDY# terminates the current active burst cycle. BRDY# is asserted low for one LCLK period at the end of each burst transfer. It must be synchronized to LCLK, so appropriate setup and hold times to LCLK must be satisfied. If LRDY# is asserted at the same time as BRDY#, BRDY# is ignored and RDYRTN# will be generated. If a VL-Bus target does not support burst cycles, it may leave this signal unconnected. If a VL-Master receives BRDY# and RDYRTN# at the same time, it should ignore BRDY#.

Tri-state control of BRDY# follows the same rules as LRDY#. BRDY# is shared among all VL-Bus devices; therefore, the active VL-Bus target drives BRDY# only during a burst transfer that it has claimed as its own. BRDY# must not be driven while ADS# is asserted. The active LBT must drive BRDY# high before tri-stating the signal. In most cases, BRDY# should not be driven during the first T2 period, because the system cache controller may be driving it. However, if the High Speed Write configuration bit is set, the LBT may start driving BRDY# during the first T2 state of a write cycle. While no burst cycle is active on the VL-Bus, BRDY# is held high by a 8k to 20k pull-up resistor located on the motherboard.

2.4.5. LREQ<x># - Local Request

LREQ<x>#, used in conjunction with LGNT<x>#, is used by a VL-Bus device to gain control of the VL-Bus and become an active LBM. There is one pair of LREQ<x># and LGNT<x># signals per slot. LBTs that are never a bus master should leave this signal not connected. The motherboard pulls this signal high using a 8k to 20k pull-up resistor. When the VL-Bus device asserts LREQ<x>#, the VL-Bus controller responds by asserting LGNT<x># for

that slot. The VL-Bus device then has control of the VL-Bus and may hold the bus until the VL-Bus controller removes LGNT<x>#.

2.4.6. IRQ9 - Interrupt Request Line 9

IRQ9 line is a high-asserted, level-triggered interrupt that is electrically connected to IRQ9 on the ISA bus. IRQ9 is present on the VL-Bus connector for "stand-alone" VL-Bus device boards that do not have any system I/O bus signals available. Normally, the VL-Bus device should connect to the interrupt lines on the system I/O bus. For Micro Channel systems, IRQ9 is connected to Micro Channel -IRQ 09 through an inverter on the motherboard.

2.5. VL-Bus Slot Pinout

Pinout shown is a top-view. The "A" side of the connector is the add-in board component side. The "B" side of the connector is the add-in board solder side. The 64 bit columns show the differences from the 32 bit signals during a 64 bit transfer.

B Side			A Side	
64 bit	**32 bit**	**Pin #**	**32 bit**	**64 Bit**
	DAT00	01	DAT01	
	DAT02	02	DAT03	
	DAT04	03	GND	
	DAT06	04	DAT05	
	DAT08	05	DAT07	
	GND	06	DAT09	
	DAT10	07	DAT11	
	DAT12	08	DAT13	
	VCC	09	DAT15	
	DAT14	10	GND	
	DAT16	11	DAT17	
	DAT18	12	VCC	
	DAT20	13	DAT19	
	GND	14	DAT21	
	DAT22	15	DAT23	
	DAT24	16	DAT25	
	DAT26	17	GND	
	DAT28	18	DAT27	
	DAT30	19	DAT29	
	VCC	20	DAT31	
DAT63	ADR31	21	ADR30	DAT62
	GND	22	ADR28	DAT60
DAT61	ADR29	23	ADR26	DAT58
DAT59	ADR27	24	GND	
DAT57	ADR25	25	ADR24	DAT56
DAT55	ADR23	26	ADR22	DAT54
DAT53	ADR21	27	VCC	
DAT51	ADR19	28	ADR20	DAT52
	GND	29	ADR18	DAT50
DAT49	ADR17	30	ADR16	DAT48
DAT47	ADR15	31	ADR14	DAT46
	VCC	32	ADR12	DAT44
DAT45	ADR13	33	ADR10	DAT42
DAT43	ADR11	34	ADR08	DAT40
DAT41	ADR09	35	GND	
DAT39	ADR07	36	ADR06	DAT38
DAT37	ADR05	37	ADR04	DAT36
	GND	38	WBACK#	
DAT35	ADR03	39	BE0#	BE4#
DAT34	ADR02	40	VCC	
LBS64#	NC	41	BE1#	BE5#
	RESET#	42	BE2#	BE6#
	D/C#	43	GND	
DAT33	M/IO#	44	BE3#	BE7#
DAT32	W/R#	45	ADS#	
		KEY		
		KEY		
	RDYRTN#	48	LRDY#	
	GND	49	LDEV<X>#	
	IRQ9	50	LREQ<X>#	
	BRDY#	51	GND	
	BLAST#	52	LGNT<X>#	
	ID0	53	VCC	
	ID1	54	ID2	
	GND	55	ID3	
	LCLK	56	ID4	ACK64#
	VCC	57	NC	
	LBS16#	58	LEADS#	

Figure 2.1. VL-Bus Slot Pinouts

2.6. Connector Pin Count

Table 2.3 is the complete list of signals on the VL-Bus connector. The signal directions of the VL local bus controller (LBC) columns are relative to the motherboard. The signal directions of the VL local bus master (LBM) and VL local bus target (LBT) columns are relative to those devices.

Pin Name	No.	LBC[1]	LBC[2]	LBM[3]	LBT
ACK64#	1	I	O	I	O
ADR<31..2>	30	O	I	O	I
ADS#	1	O	I	O	I
BE<3..0>#	4	O	I	O	I
BLAST#	1	O	I	O	I
BRDY#	1	I	O	I	O
D/C#	1	O	I	O	I
DAT<31..0>	32	I/O	I/O	I/O	I/O
ID<4..0>	5	O	O	I	I
IRQ9	1	I	I	O	O
LBS16#	1	I	O	I	O
LBS64#	1	O	I	O	I
LCLK	1	O	O	I	I
LDEV<x>#[4]	1	I	N/A	N/A	O
LEADS#	1	O	I	O	I
LGNT<x>#[4]	1	O	O	I	N/A
LRDY#	1	I	I	N/A	O
LREQ<x>#[4]	1	I	I	O	O
M/IO#	1	O	I	O	I
RDYRTN#	1	O	O	I	I
RESET#	1	O	O	I	I
W/R#	1	O	I	O	I
WBACK#	1	O	O	I	N/A
(power)	8				
(ground)	14				
(reserved)	1				
Total per slot	112				

Table 2.3. VL-Bus Slot Pin List.

1 Direction of signals for the LBC if the host CPU or a system I/O bus master is active.
2 Direction of signals for the LBC if an LBM is active.
3 Direction of signals for any LBM while it owns the VL-Bus.
4 One unique signal per slot or device.

2.7. (E)ISA Signals Used by a Motherboard VL-Bus Controller

Since the VL-Bus connector is physically inline with the ISA or EISA connector, all (E)ISA signals are available to the VL-Bus board. While use of (E)ISA signals are not required for VL-Bus devices, this is an area that a manufacturer may use for product differentiation.

Since the VL-Bus controller must echo back (E)ISA bus master and DMA transfers addressed to VL-Bus targets, the VL-Bus controller *must* have ISA-bus signals available to it (i.e., it must see MEMR#, MEMW#, etc., to track a DMA or ISA bus master command). This is not a limitation since generally the VL-Bus controller physically resides on the motherboard. The following signals are the minimum signals a VL-Bus controller must have available to operate in an EISA or ISA environment.

2.7.1. MEMR#, MEMW#, IOR#, IOW# - ISA Command Strobes

These command strobe inputs, along with AEN and REFRESH, tell the VL-Bus controller what type of DMA transfer is taking place. M/IO# is encoded as high if MEMR# or MEMW# is the active command strobe during DMA/bus master cycles. W/R# is encoded as high if MEMW# or IOW# is the active command strobe during DMA/bus master cycles.

2.7.2. AEN - Address Enable

This input indicates to the VL-Bus controller when a DMA cycle is in progress. When AEN is asserted and REFRESH is not asserted, the memory address is driven back to the VL-Bus address bus.

2.7.3. REFRESH# - Refresh Status

This input indicates that the current cycle is a refresh cycle. All DMA cycles that are refresh cycles are ignored by the VL-Bus controller.

2.7.4. IOCHRDY - I/O Channel Ready

This signal generates additional wait states during an IO bus master or DMA transfer. The VL-Bus controller pulls IOCHRDY low immediately after recognizing that the DMA target is a VL-Bus target. IOCHRDY is released by the VL-Bus controller when the VL-Bus target asserts LRDY#.

2.7.5. SA<16..0>, LA<23..17>, SBHE# - ISA Address Bus

The system addresses are driven back to the VL-Bus address bus during DMA and ISA bus master cycles.

2.7.6. SD<15..0> - ISA Data Bus

This bi-directional bus provides the data path between VL-Bus devices and the system I/O bus devices. During DMA memory write operations the VL-Bus controller drives the VL-Bus data bus with the SD bus information. During DMA memory reads the VL-Bus controller drives SD with VL-Bus data bus information.

2.7.7. MEMCS16# - Memory Chip Select 16-Bit

This signal is output by the VL-Bus controller to signify that the VL-Bus target can accept 16-bit memory transfers. It is always asserted for DMA and bus master transfers involving a VL-Bus target.

3. VL-Bus Transfers

3.1. General Overview of a VL-Bus Transfer

A VL-Bus CPU transfer begins when valid address and status information is placed on ADR<31..02>, M/IO#, W/R#, D/C# and BE<3..0>#. ADS# is strobed to begin the transfer. The VL-Bus target has 20ns to recognize the address to which it should respond and assert LDEV#. Depending on the host CPU speed and VL-Bus controller design, LDEV# is generally sampled at either the end of the first T2 or second T2 for CPU or Local Bus Master initiated transfer. If LDEV# is asserted, the system I/O bus controller does not start a system I/O bus cycle.

The VL-Bus target begins to drive LRDY# (asserted or negated, depending on whether additional wait states are needed) after ADS# is negated (rules governing exactly when the VL-Bus target may start to drive LRDY# are covered in more detail in section 3.3). When the VL-Bus target has completed the transfer, it asserts LRDY# for one LCLK cycle and then negates LRDY# for one-half LCLK cycle prior to the VL-Bus target releasing LRDY#. When the VL-Bus controller detects LRDY# asserted it may immediately assert RDYRTN# or it may resynchronize LRDY# and assert RDYRTN# on the next LCLK cycle. If the current transfer is a read, the VL-Bus target must hold the read data on the data bus until the LCLK which RDYRTN# is sampled asserted.

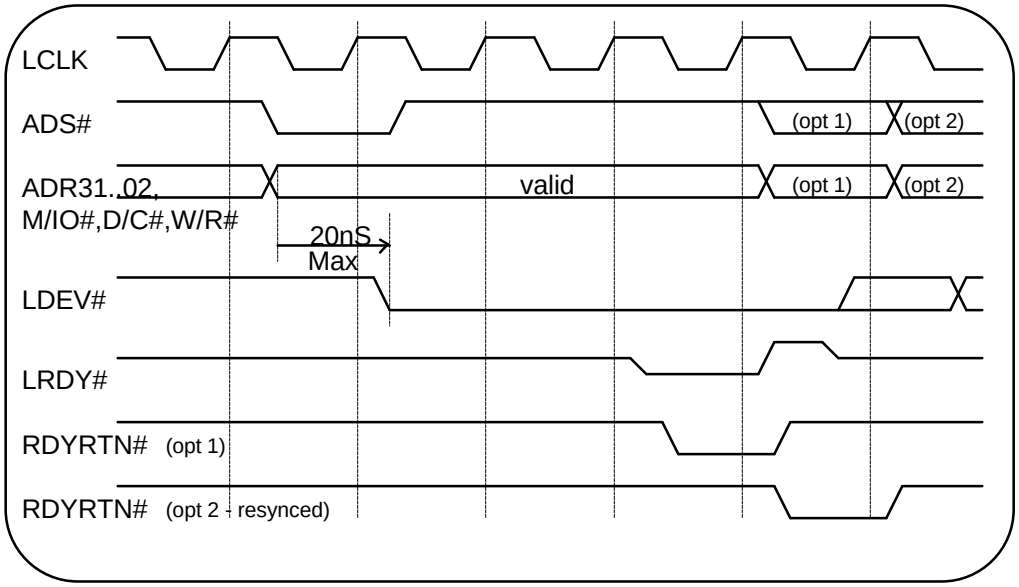


Figure 3.1. Simplified Timing Diagram.

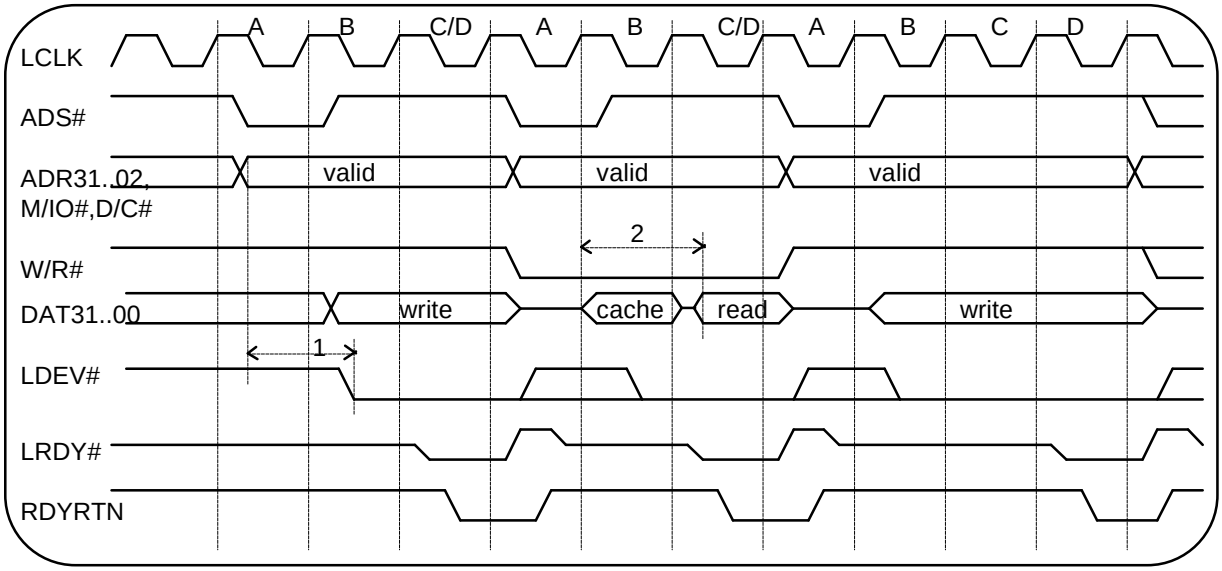
RDYRTN# is equivalent to the 486 CPU RDY#. Slower systems (i.e., 33 MHz or less) do not require LRDY# to be resynchronized. Faster systems may resynchronize LRDY# before asserting RDYRTN#, causing CPU RDY# to appear one clock cycle after LRDY#. During a DMA or system I/O bus master read cycle the end of the cycle is asynchronous to the CPU clock; therefore RDYRTN# (synchronized to LCLK) may appear any number of clock cycles after LRDY#. The VL-Bus target must always watch RDYRTN# during read operations, which it terminates with LRDY# and hold data on the data bus until RDYRTN# is sampled asserted.

DMA transfers and system I/O bus master transfers are both handled equivalently by the VL-Bus controller. Up to three bus masters may reside on the VL-Bus. A handshake with LREQ<x># and LGNT<x># is used for arbitrating for VL-Bus control.

3.2. Detailed Description of a CPU Transfer

During a host CPU read the VL-Bus target must not start driving the data bus until after the end of the first T2 period. This data bus drive restriction is necessary because a main memory cache may be driving the data bus during the first T2 period. Read transfers are not effected by the High Speed Write configuration bit. When LRDY# is asserted by the VL-Bus target, the VL-Bus controller has the option of either immediately asserting RDYRTN# or resynchronizing the signal and asserting RDYRTN# on the next LCLK cycle. To ensure that the CPU has received requested data, the VL-Bus target must hold any read data on the data bus until RDYRTN# is sampled asserted.

During host CPU writes, the VL-Bus target may start driving LRDY# and BRDY# at the LCLK cycle after ADS# (i.e., the first T2 period) if the High Speed Write configuration bit is set. If the configuration bit is a negated, the VL-Bus target must not start driving LRDY# until the second LCLK cycle after ADS# (second T2 period). The High Speed Write bit also indicates when the data bus may be first sampled. If the bit is high, the VL-Bus target may sample (or latch) DAT<31..00> during the first T2 state. If the bit is low, the VL-Bus target must wait until the second T2 to sample DAT<31..00>. Supporting a High Speed Write transfer is optional for both the VL-Bus controller and VL-Bus target. All signal relationships to LCLK are shown in section 5.



No.	Description	Min	Max	Units
1	Address to LDEV# asserted	0	20	nsec
2	Data bus Driven after ADS# negated (clock edge)	1	-	LCLK

Figure 3.2. CPU to VL-Bus Timing.

A. ADS# is asserted while the address and status signals are stabilizing. In this phase the address and status lines are changing; therefore, LDEV# will be unstable. Since LRDY# may still be driven by the previous LBT during the first half of this phase, the new LBT must not drive LRDY# during this phase.

B. ADS# is negated. LDEV# may not yet be stable. The active LBT may begin driving LRDY# during either this or the next phase, depending on the value of the High Speed Write configuration bit.

C. On the rising edge of LCLK during this phase, LDEV# is sampled. The active LBT must start driving LRDY# by this phase. If additional wait states are required they are added after phase C but before phase D.

D. LRDY# is asserted by the VL-Bus target for one LCLK cycle. LRDY# may be echoed back directly to the VL-Bus controller or resynchronized and echoed back as RDYRTN# on the next LCLK cycle. When the active LBT receives RDYRTN#, it stops driving the data bus (in the case of a read). LRDY# is negated on the next one half LCLK cycle before being released.

3.3. Simulated Open Collector Outputs

LRDY#, BRDY#, and LBS16# are simulated open collector signals from the VL-Bus target. These signals are shared among all the VL-Bus targets. The following rules allow these shared signals to coexist without driver contention.

3.3.1. Rules

1. The simulated open collector outputs must not be driven by any VL-Bus target that is not the active target. If no activity is occurring on the VL-Bus, these signals are held in the high state by 8k to 20k pull-ups on the motherboard.

2. The VL-Bus target must not start driving the simulated open collector output signal during T1 states. The reason for this restriction is that if the previous transfer was for another VL-Bus target, the line will still be driven negated by that device during the T1 state.

3. If the High Speed Write configuration bit is set, the VL-Bus target may start driving LRDY# and BRDY# at the end of the first T2 state. If the High Speed Write configuration bit is a 0, the VL-Bus target must wait until the end of the second T2 state to start driving the LRDY#. If LBS16# is used, the VL-Bus target must start driving this simulated open collector output at the end of the first T2 state.

4. Throughout the remainder of the transfer, the active VL-Bus target must drive these signals. When the signals are asserted, the active VL-Bus target drives the appropriate signal low.

5. Before returning the signal back to tri-state, the VL-Bus target must drive the signal high. The target has one full clock cycle to bring the signal to a valid high level and float it. The valid high level must be met within the maximum "Valid delay" time for that signal. The target may accomplish this by driving the signal high during the first half of the cycle, and then floating it during the second half. The

designer should note the allowable minimum and maximum clock duty cycle when using this approach and assure that the high level will be met and the signal will be floated in time.

3.3.2. Examples

Figure 3.3 show LRDY# in different situations with respect to LCLK and ADS#. The highlighted areas of LRDY# show the stages LRDY# is driven. Notice that the signal is driven high one half LCLK cycle into the next transfer. Each example situation is labeled and described in more detail. These are example situations only.

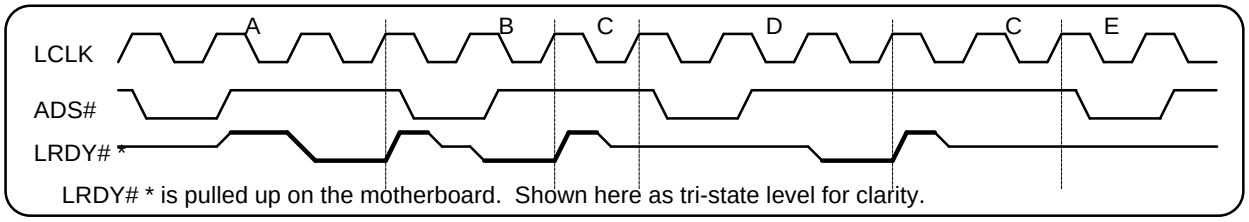


Figure 3.3. Example of a Simulated Open Collector Control.

A. This is an example of a one wait state write transfer with the High Speed Write configuration bit set. While ADS# is asserted, the intended VL-Bus target must not drive LRDY#. After ADS# is negated, the intended VL-Bus target starts driving LRDY#. Since an additional wait state is needed, the VL-Bus target drives LRDY# high. After driving LRDY# low, the active VL-Bus target must drive LRDY# back high for one half LCLK cycle before tri-stating this signal.

B. This is an example of a High Speed Write zero-wait-state transfer. The active VL-Bus target immediately drives LRDY# low after ADS# (i.e., during the first T2 cycle). LRDY# is being driven back high by the VL-Bus target active in the previous transfer during the first half of the first LCLK cycle.

C. Idle state. In these examples LRDY# is being driven back high by the VL-Bus target active in the previous transfer during the first half of the first LCLK cycle.

D. This example shows a standard one wait state read (or write). LRDY# is not driven until the second T2 cycle. The active VL-Bus target then drives LRDY# low (i.e., during the second T2 cycle).

E. Transfer not intended for any VL-Bus target. LRDY# remains tri-stated. A pull-up on the motherboard holds this line high.

3.4. 16-bit Operations on the VL-Bus

3.4.1. Encoding BE<3..0># for Host 386SX-Class Systems.

Although the VL-Bus is defined as a 32-bit host system, a 16-bit 386SX-class CPU can be used as a host CPU of the VL-Bus interface. Since the VL-Bus does not have A1, BLE#, or BHE# required for 386SX operation, this information must be encoded onto BE<3..0>#. The following equations show how to encode BE<3..0># using BLE#, BHE#, and A1. These equations are expressed as logical levels, not actual levels.

- BE0 = BLE * !A1
- BE1 = BHE * !A1
- BE2 = BLE * A1
- BE3 = BHE * A1

3.4.2. Use of LBS16#

VL-Bus targets with a 16-bit data bus can connect to the VL-Bus interface using the LBS16# signal. LBS16# informs the CPU or active LBM that the target can transfer only up to 16 bits of data and another transfer may be required if data is still remaining.

Current				Next with LBS16#			
BE3#	BE2#	BE1#	BE0#	BE3#	BE2#	BE1#	BE0#
1	1	1	0	N/A[B]			
1	1	0	0	N/A[B]			
1	0[A]	0	0	1	0	1	1
0[A]	0[A]	0	0	0	0	1	1
1	1	0	1	N/A[B]			
1	0[A]	0	1	1	0	1	1
0[A]	0[A]	0	1	0	0	1	1
1	0	1	1	N/A[B]			
0	0	1	1	N/A[B]			
0	1	1	1	N/A[B]			

Table 3.1. Byte Enables with LBS16#.

[A] Byte enable ignored on the first transfer. It is repeated on the second transfer.
[B] A second transfer is not required. LBS16#, if asserted, is ignored.

3.4.3. 16-Bit Word Steering

Since the VL-Bus supports both 386-class and 486-class host CPUs, the 16-bit word steering via LBS16# must be compatible with both class of CPUs. The main difference between the two CPU classes is shown in the table below. The first transfer of both CPUs is the same. The 16-bit device would operate on the low order bits only and assert LBS16# to force a second transfer. The second transfer, however, differs in that the 386 duplicates the high order data word to the low order word, while on the 486 the data remains only on the high order word. The 386 duplication feature cannot be used on VL-Bus boards since all VL-Bus add-in boards must work in both 386 and 486 environments. If the VL-Bus target always receives write data on the proper byte lane, it can stay compatible with both 386 and 486 word (and byte) steering.

	First Transfer				Second Transfer			
CPU	byte3	byte2	byte1	byte 0	byte 3	byte 2	byte 1	byte 0
386	31..24	23..16	***15..8***	***7..0***	***31..24***	***23..16***	31..24	23..16
486	31..24	23..16	***15..8***	***7..0***	***31..24***	***23..16***		

Table 3.2. 32-Bit Write Transfer Sequences Using LBS16#.

If the VL-Bus target is performing a 16-bit read using LBS16#, it must broadcast the data onto both the high order and low order words to satisfy both the 386 and 486 word steering requirements. The 386 reads both high order and low order data off the low order word only, while the 486 reads all data from the proper byte lanes. For any CPU or LBM reads from a VL-Bus target, the VL-Bus target must drive all 32 bits of the data bus, whether all bytes of data are defined or not.

	First Transfer				Second Transfer			
CPU	byte3	byte2	byte1	byte 0	byte 3	byte 2	byte 1	byte 0
386			***15..8***	***7..0***	31..24*	23..16*	***31..24***	***23..16***
486			***15..8***	***7..0***	***31..24***	***23..16***	31..24*	23..16*

* The VL-Bus target must duplicate data to satisfy both 386 and 486 requirements.

Table 3.3. 32-Bit Read Transfer Sequences Using LBS16#.

A VL-Bus master board should emulate the 486 byte and word steering when initiating a second transfer due to LBS16# being asserted by a target VL-Bus device.

The enable control for the 16-bit VL-Bus target buffers are listed below. It is assumed that LBS16# is used to split a transfer occurring on both the upper and lower word. These equations are expressed as logical levels, not actual levels.

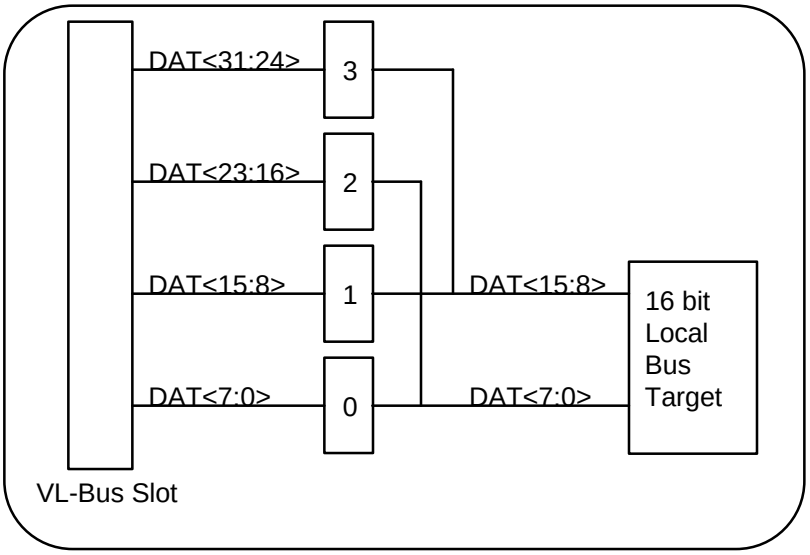


Figure 3.4. Buffer Control for 16-Bit Devices.

- Buffer 3 = Read + Write * (BE3 * !BE1)
- Buffer 2 = Read + Write * (BE2 * !BE0)
- Buffer 1 = Read + Write * BE1
- Buffer 0 = Read + Write * BE0

3.4.4. Decoding A1, BLE, and BHE# for 16-Bit VL-Bus Devices

Some 16-bit VL-Bus devices may require BLE#, BHE#, and A1. These signals may be decoded from BE<3..0>#. The following equations can be implemented within a PLD on the LBT or internal to a VL-Bus device chip. These equations are expressed in logical levels, not actual levels.

- BLE = BE0 + !BE1 * BE2 + !BE1 * BE3
- BHE = BE1 + BE3
- A1 = BE0 + BE1

3.4.5. Burst Ordering Using LBS16#

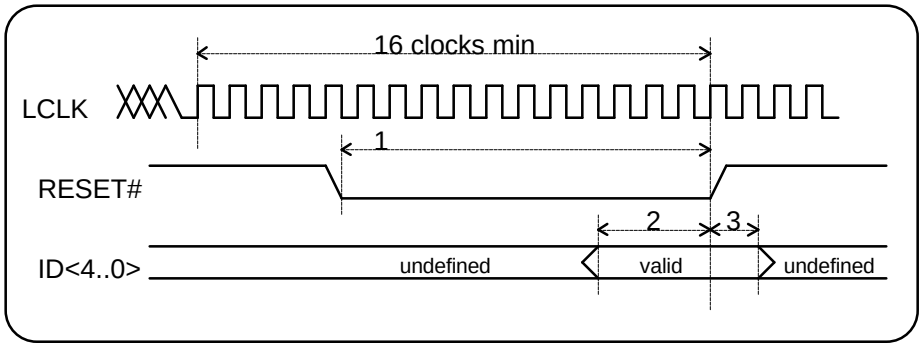
The burst address ordering when LBS16# is asserted follows the same sequence as the 32-bit burst order. Each 32-bit transfer, however, is split into two 16-bit transfers with the second transfer address incremented by two bytes. The table below is a listing of possible burst address sequences when LBS16# is asserted. The value of the first address in the burst sequence identifies which sequence is used.

1st Adr	2nd Adr	3rd Adr	4th Adr	5th Adr	6th Adr	7th Adr	8th Adr
0	2	4	6	8	A	C	E
4	6	0	2	C	E	8	A
8	A	C	E	0	2	4	6
C	E	8	A	4	6	0	2

Table 3.4. 16-Bit Burst Address Ordering Using LBS16#.

3.5. Activity during RESET#

LCLK is not guaranteed stable throughout the entire duration of RESET#. However, there are at least 16 stable clock cycles while RESET# is still asserted immediately prior to the trailing edge of RESET# (See Figure 3.5). Table 3.5 shows the permissible signal states during RESET# after VCC and LCLK have reached their normal operating parameters.



No.	Description	Min.	Max.	Units
1	RESET# pulse width	1.5	-	usec
2	ID setup to RESET#	200	-	nsec
3	ID hold from RESET#	10	-	ns

Figure 3.5. RESET# Timing

Signal	Motherboard	Target	Master
BE<3..0>#, ADR<31..2>	May drive	(Never drives)	Floated
DAT<31..0>	May drive	Floated	Floated
LCLK, RESET#	Drives	(Never drives)	(Never drives)
LGNT#, WBACK#	May drive	(Never drives)	(Never drives)
LREQ#	(Pull-up only)	(Never drives)	May drive high or float
LEADS#	May drive	(Never drives)	Floated
ADS#, D/C#, W/R#, M/IO#	May drive	(Never drives)	Floated
LDEV#	(Pull-up only)	May drive* or float	(Never drives)
LBS16#	May drive	Floated	(Never drives)
RDYRTN#	May drive	(Never drives)	(Never drives)
BLAST#	May drive	(Never drives)	Floated
LRDY#, BRDY#	May drive	Floated	(Never drives)
ID<4..0>	Drives	(Never drives)	(Never drives)

* If LDEV# is driven, it must not be driven low except by decoding a valid address.

May drive	Optionally driven by the motherboard (often by the CPU)
Drives	Must be driven by the motherboard during reset (specifically when RESET# is going high.)
(Pull-up only)	Refers to a signal that is only driven by the motherboard with a pull-up resistor
(Never drives)	Refers to a signal that is never driven by the respective device. A signal may be listed as "never drives" for a master that is driven at times by the same card as a target and vice-versa.
Floated	Refers to a signal that the respective device sometimes drives, which must be floated when RESET# is low.

Table 3.5. Signal State during RESET#

3.6. Burst Transfers

3.6.1. Overview

486 style bursting is supported on the VL-Bus. Bursting is optional for both VL-Master and VL-Targets. A burst cycles begins the same way as a non-burst cycle. The address and control signals are generated and an ADS# pulse is issued. BLAST# will be undefined during T1, but will become valid at the end of the first T2. When the CPU owns the bus BLAST# may change from high to low during the second T2. This is due to the KEN decode on the motherboard, which cannot occur by the end of T1 (a motherboard must drive KEN# low during T1 in case there is a secondary cache hit, then take it high during the first T2 if it is to be a non-cache cycle).

If the target is capable of bursting and BLAST# is high at the end of the first T1, it may drive BRDY#. The target may change the data on the T state boundary where BRDY# is sampled. It should not wait for a RDYRTN# (a RDYRTN# will probably not be generated). The target should sample BLAST# on the clock edge where BRDY# is sampled low to determine if the bus cycle terminates. If BLAST# is low, the BRDY# will terminate the burst. The bus owner will drive new values for A2 and A3 following each BRDY#. The VL-Target may wait for the new values, or predict the values internally for faster bursts. The 486 algorithm is used to predict the address. If LBS16# is asserted during a burst cycle, BLAST# should be qualified with BE<3..0># to determint the last transfer of a burst cycle.

The ID bits provide information, particularly to the VL-Master, as to the capabilities of the motherboard. ID4=1 indicates that the motherboard supports the 2.0 spec. bursting protocol. ID0 indicates whether the motherboard implements the burst protocol for write cycles also.

ID4	ID1	ID0	CPU	VL-Master Bursting
0	1	0	486	May support bursting (1.0 motherboard)
1	1	0	486	Support burst protocol for reads
1	1	1	486	Support burst protocol for reads and writes

Table 3.6. VL-Master Bursting

3.6.2. Burst Protocol Responsibilities - Master

VL-Bus Masters which do not wish to burst, must drive BLAST# low when they own the bus. They must look at both RDYRTN# and BRDY# and terminate the cycle when either is low.

VL-Bus Masters must be able to accept 0 wait state read and writes. VL-Bus Targets will never generate a 0 wait state read, but the system board may.

If ID4 is 0, it is suggested that the VL-Master not attempt to burst since the motherboard implementation may vary.

If ID4 is 1, the VL-Master can be assured that the system board supports the bursting protocol on reads. It may not necessarily burst or may burst on selected addresses (to main memory but not to the ISA bus, for instance) but it will support the protocol. Burst writes should not be attempted, since the motherboard may ignore BLAST# on writes. Many 486 cache systems drive BRDY# for write cycles.

If ID4 and ID0 are both 1s, the VL-Master can be assured that the system board supports the bursting protocol on both reads and writes. It may not necessarily burst, or may burst selected addresses.

Basic Burst Protocol for the VL-Master:

BRDY# and RDYRTN# are both sampled, as the 486 samples them.

BLAST# is driven low for a single transfer or for final transfer of a burst.

When BLAST# is low, either BRDY# and RDYRTN# will terminate the bus cycle.

When BLAST# is high, BRDY# will continue the burst, while RDYRTN# will terminate it.

Burst reads may be of 1 (LBS16# low), 2, or 4 dwords.

Burst writes must be 4 dwords (16 bytes) and all bytes must be written. If a block of data is to be written that does not start on a 16 byte boundary, non-burst cycle should be done until a 16 byte boundary is reached. 1 dword burst writes can also occur using LBS16#.

Bursts by a VL-Master must always increment the address, although all targets that burst must follow the 486 burst algorithm. Therefore, the only bursts that are allowed from a VL-Master are as follows:

Starting Address	Burst Length	Burst Sequence	Burst Sequence w/BS16 low
xxx0	2 dwords	xxx0, xxx4	0, 2, 4, 6
xxx0	4 dwords	xxx0, xxx4, xxx8, xxxC	0, 2, 4, 6, 8, A, C, E
xxx8	2 dwords	xxx8, xxxC	8, A, C, E

Table 3.7. Burst Read Protocol For VL-Master

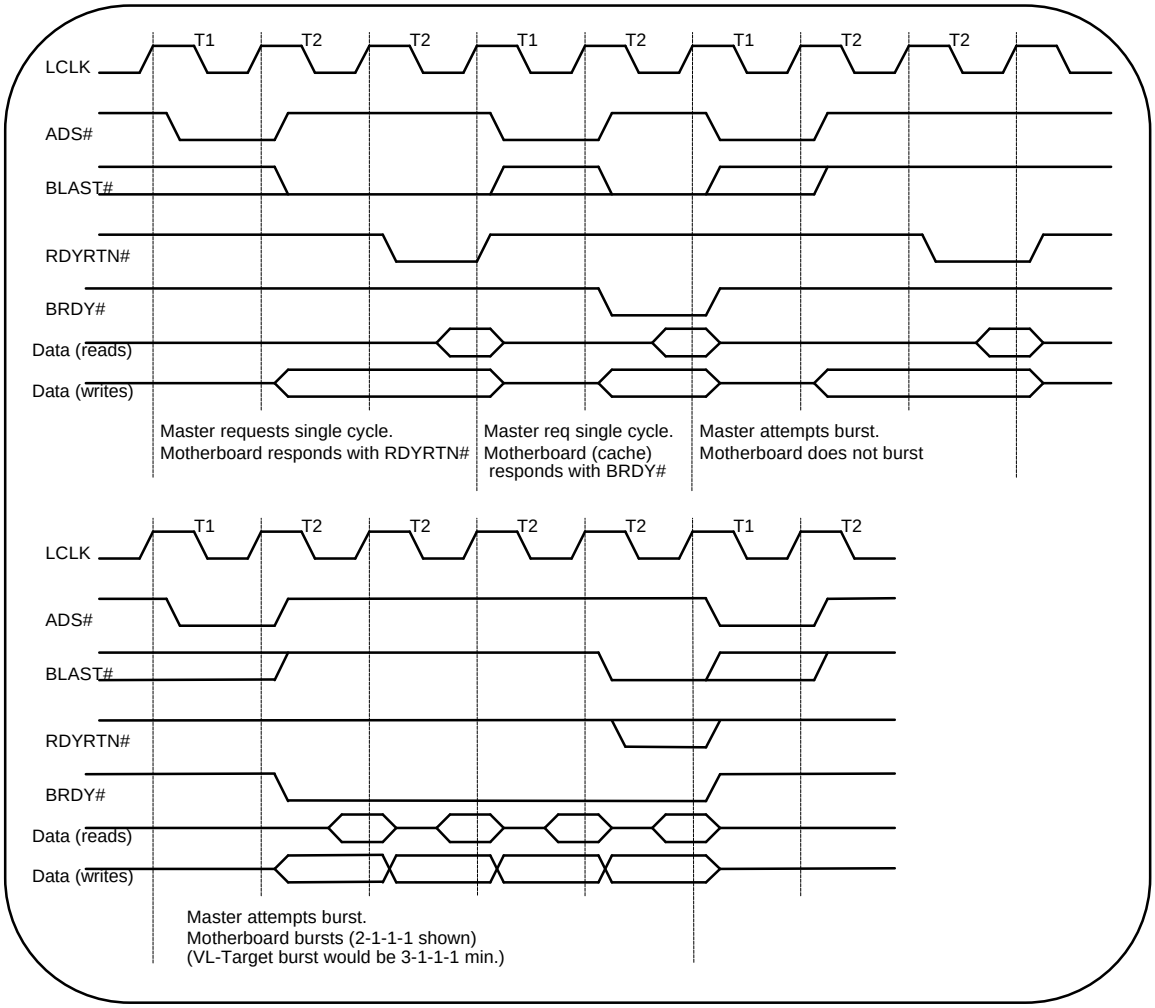


Figure 3.6. Bursting From VL -Master Point Of View

Same timing for reads and writes

3.6.3. Burst Protocol Responsibilities - Targets

VL-Bus Targets which do not wish to burst may drive LRDY# to terminate all bus cycles. On read cycles, they should continue to drive valid data until the end of the T state where RDYRTN# is sampled active. The remainder of the items below may be ignored if LRDY# is always driven.

VL-Bus Targets may attempt to burst read or write cycles in any system, with limitations listed below. The ID bits need not be checked, except for the speed optimization listed below.

If BLAST# is low at the end of the first T2, LRDY# must be used to terminate the bus cycle (there is an exception to this rule below). BRDY# may also be driven, but LRDY# will take precedence. This is to allow DMA cycles, which require the VL-Target to hold the data until RDYRTN# is generated, to work properly. Since BLAST# must be sampled at the end of the first T2, the minimum burst is a 3-1-1-1. This does not impact the speed of read cycles since they are a minimum of 1 wait state any way. It does impact the speed of write bursts, however, since BRDY# may not be returned for the first transfer in 0 wait states. If ID0, ID1, and ID4 are all 1s (indicating that write burst are supported by the motherboard), BRDY# may be returned in 0 wait states. This is an optional optimization allowing a 2-1-1-1 write burst.

The exception to the LRDY# rule is as follows: If BLAST# is high at any time during the bus cycle (including the end of T1) the target is free to use BRDY#. This is a safe assumption since the system board will always drive BLAST# low for the entire cycle for DMA accesses. This exception allows the target to burst in a few more cases, specifically if it pulls BS16# low, which will cause BLAST# to go high later in the cycle when the CPU has the bus.

If BLAST# is high at the end of the first T2, the VL-Bus target may treat the cycle as a burst or non-burst, following the protocol of the 486. The VL-Bus target need not end the burst with an LRDY#. When generating BRDY#, the target should not wait for a RDYRTN#.

A VL-Bus target should NOT assume that BLAST# will be high on a write cycle, and drive BRDY#, ignoring BLAST#, and expecting the bus cycle to terminate without bursting.

A VL-Bus target that bursts should be able to start a burst write on any address, should be able to burst 2 or 4 dwords, and should monitor the byte enables on all transfer of the burst. This will allow the target to work if some future CPU decides to do bursts that are not necessarily 16 byte linear.

A VL-Bus target that bursts should follow the 486 burst algorithm if it updates the address internally. If the target pulls BS16# low, it should follow the 486 burst algorithm in this case also. VL-Masters will only generate a subset of the 486 cases, but the CPU will of course, generate all cases.

3.6.4. Burst Protocol Responsibilities - System Board

A system board may set ID4 to a 1 if it conforms to the 2.0 spec. As far as bursting is concerned, this means that the 486 burst protocol is followed for read cycles. The motherboard does not necessarily have to burst, but it should honor the exact burst protocol (driving RDYRTN# if not bursting, driving BRDY# to burst).

A system board may drive ID1 and ID0 both high if it follows the burst protocol for both read and write cycles. It does not necessarily have to burst write cycles, but if it is NOT bursting, it should assert RDYRTN# when BLAST# is high, and not BRDY#.

VL-Masters are treated identically to a 486 CPU. Optimization may be implemented, such as disabling secondary cache line fills when a VL-Master owns the bus.

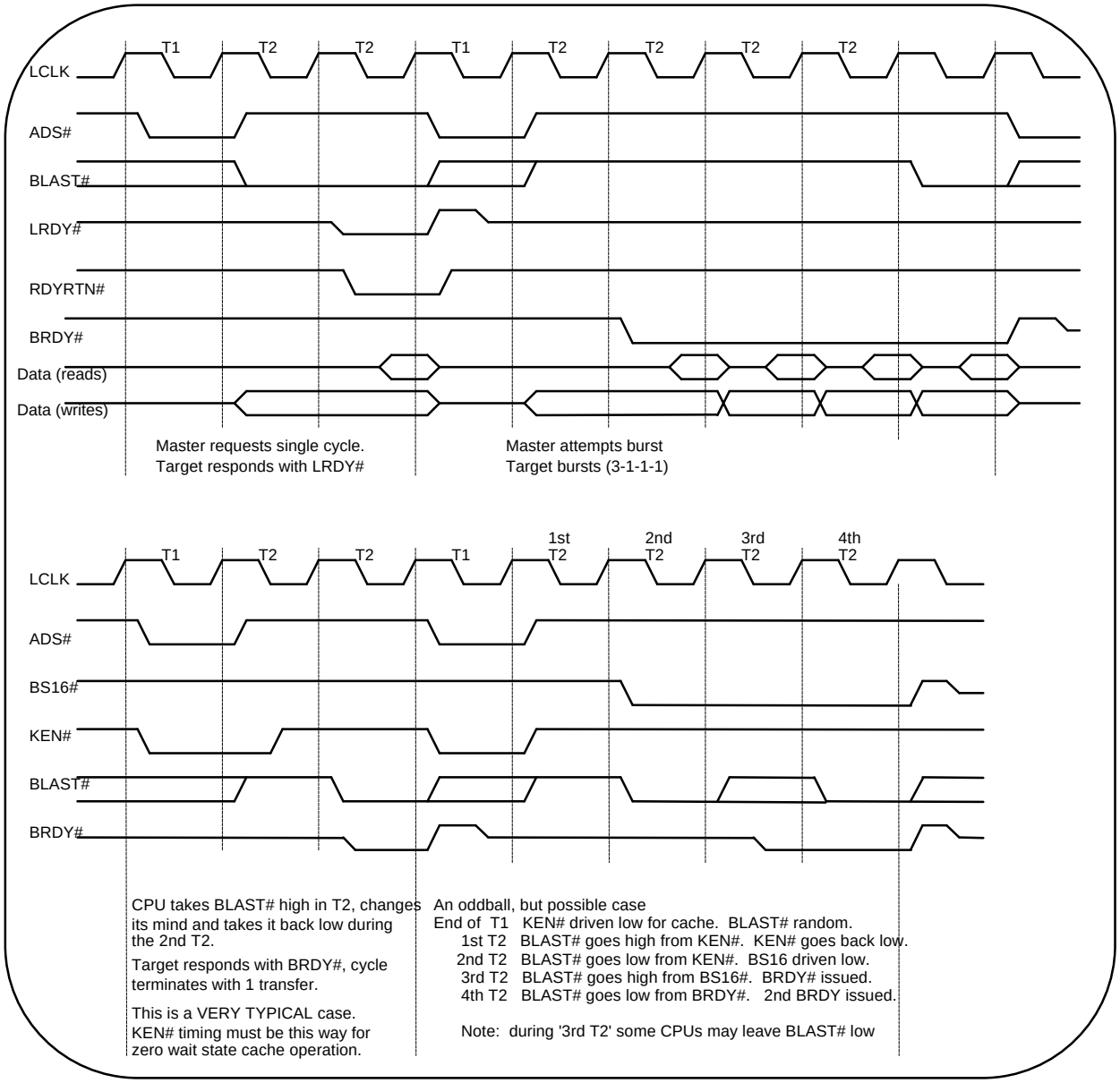


Figure 3.7. Bursting from VL-Target Point of View

3.7. Cache Support

The VL-Bus supports both write through and write back, for both the L1 (CPU) and L2 (motherboard) caches. Versions 1.0 of the spec. described the write through cache support (which would also work with most implementations of L2 write back cache). It left the definition of the WBACK# signal for the future. WBACK# is now fully defined.

3.7.1. Version 1.0 Write Through Cache Support

In version 1 of the spec. VL-Bus masters were instructed to drive LEADS# on all accesses to invalidate the CPU cache. Several things are implied by the wording, but not specifically stated. These are:

- LEADS# should be generated for read cycles. This is not required for the 486 write through cache, but is required for CPUs with write back caches.
- LEADS# should be generated for I/O cycles. It is important that this **NOT** happen for CPUs with write back cache.
- LEADS# is driven at the same time as ADS#

See the rules for VL-Bus Masters below for when to drive LEADS#.

3.7.2. Write back cache

The purpose of the WBACK# signal is to support systems with write back caches in the CPU. All VL-Masters must honor the WBACK# signal or they will not work in many systems where the CPU contains a write back cache.

A CPU with a write back cache must be snooped on each memory read and write cycle by a VL-Master, if there is some chance that it may contain modified data. The snoop need not occur if the memory address is not cacheable, or it is write through cacheable (although the CPU cache must still be invalidated on memory write cycles). The motherboard may be able to make this determination.

When a snoop occurs, the CPU provides a signal indicating whether it contains modified data or not. If it does contain modified data, the only recourse is to allow the CPU to write the entire dirty cache line back to main memory. CPUs will not respond as targets and provide the requested data or receive the write data. In order to allow the CPU to write the modified line back to main memory, the VL-Master must be backed off of the bus. This is the function of the WBACK# signal.

The "snoop complete" function is contained entirely on the motherboard, since VL-Bus targets are not write back cacheable.

The Basic VL-Bus Write-Back Algorithm is:

The VL-Bus Master initiates a memory cycle.
An inquiry cycle is initiated to the CPU cache (no RDYRTN# or BRDY# is returned yet).
If the inquiry cycle hits on a modified or 'dirty' cache line:

- The Local Master is removed as bus owner.
- The CPU is given temporary bus ownership.

- The CPU writes the data back to the L2 cache and/or DRAMs.
- The CPU is removed as bus owner.
- Bus ownership is returned to the Local Master.
- The Local Master reissues the bus cycle with a new ADS#.

The L2 cache or main memory responds to the bus cycle.

3.7.2.1. Rules for Each Device:

VL-Bus Master:

VL-Masters must drive LEADS# low at the same time as ADS# for both memory reads and memory writes. It must not drive it for I/O cycles.

When WBACK# is sampled low, the local master must immediately remove itself from the bus, floating the address and control buses. This must be done on the clock edge where WBACK# is sampled.

The buses must remain floated until WBACK# is sampled high.

The bus cycle that was in progress when WBACK# went low must be reinitiated when WBACK# goes back high.

The master must NOT take LREQ# high while the write back is in progress. The backed off bus cycle, the write back, and the reinitiated bus cycle should be considered an atomic operation. LGNT# could be high or go high during this process as part of the standard preemption process. The master does not surrender its claim to the bus until after the reinitiated bus cycle has completed.

WBACK# is synchronous to LCLK, and it will only go low after the ADS# and with or before the first READY (either RDYRTN# or BRDY#)

VL-Bus Targets:

WBACK# will not go low while accessing a VL-Target.

A VL-Target must be able to tolerate seeing an ADS on the bus without a READY being generated. However, this type of cycle will not be directed at the target.

Local Bus Controller (system board)

WBACK# should only be generated after an ADS# and with or before the first ready (either RDYRTN# or BRDY#). The L2 cache controller and DRAM controller may have to be held off until the CPU cache is snooped. On a read cycle the DRAM or cache controller may drive the data bus until the WBACK# signal goes active.

The LGNT# to the current master is unaffected by the write back. It will normally remain low, but if it was high as part of the preemption process, it should remain high. The LGNT<x># may also go from low to high during a write back to begin the preemption process if another master or the system board DMA controller requests the bus. This will not effect the write back process. The master will not surrender its claim on the bus by dropping its LREQ# until the write back is completed and it has finished its reinitiated bus cycle.

WBACK# should only be generated for system board memory accesses. The HITM# signal from the CPU may be used directly in many cases. Caching only main memory in write back mode is sufficient for guaranteeing that WBACK# will not be generated for other cycles.

The system board is responsible for maintaining the cache coherency and must do whatever is necessary in the architecture implemented. This generally includes generating the

WBACK# signal (which may be HITM# from the CPU in many cases) and taking the CPU out of hold to allow the write back to occur.

Local Bus Controller System Enhancements (optional)

The local bus controller need not wait for the snoop result from the CPU if it can accurately predict the answer. For instance, if a previous snoop indicated that the data is not in the cache (or invalidated it) and another access is made to the same cache line, the snoop result may be accurately predicted. In this case the L2 cache and/or DRAM controller need not wait for the snoop result before completing the access.

Other Possible Optimizations:

Making areas of memory write-through instead of write back (specific buffer areas).

Guaranteeing the L2 cache is inclusive of the entire L1 cache. In this case a secondary cache miss assures that the line is not dirty in the CPU. This is complicated in many cases, requiring at least as many sets in the L2 cache as the L1 cache, and knowing the CPU replacement set number on each line fill.

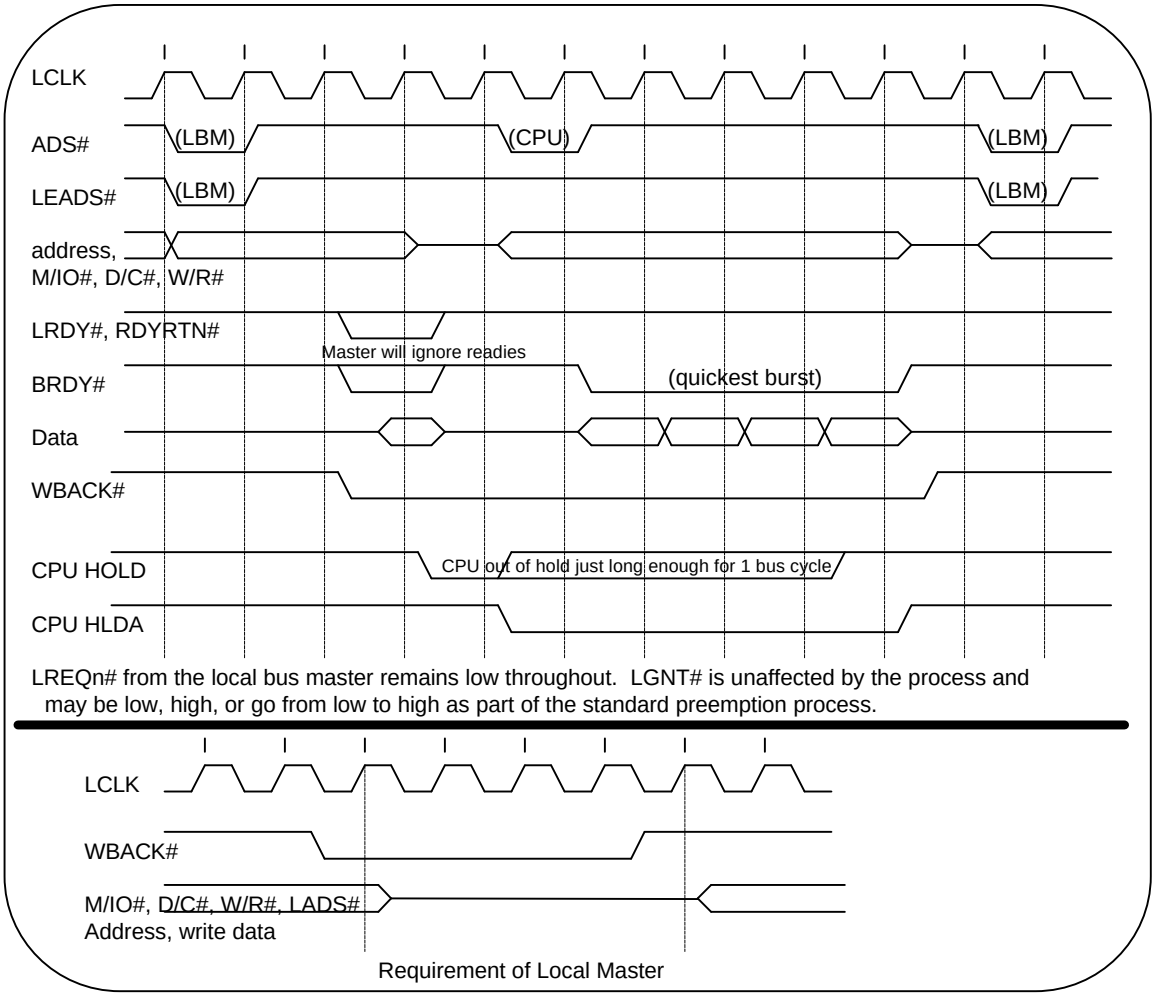


Figure 3.8. Back off Cycle of Local Bus Master

3.8. Shadowing VL-Bus Writes to the System I/O Bus

Some system I/O bus add-in boards may need to monitor I/O port write activity to other add-in boards. The most popular example of this type of activity is a multimedia board that mixes its own graphics on top of the standard display graphics. A board such as this must monitor activity to the palette tables so it may maintain a current copy in its registers.

Shadowing can be accomplished in two ways. For both these schemes to work properly, the VL-Bus controller MUST allow all VL-Bus target devices to "see" all transfers occurring on the system I/O bus. In other words, the addresses and data being presented to the system I/O bus must also be presented to the VL-Bus. This is not a serious limitation, since a typical motherboard design does this anyway.

The primary method of shadowing a VL-Bus I/O port write to the system I/O bus allows the VL-Bus target to decide about which addresses should be shadowed. When an I/O write transfer is presented to the VL-Bus that addresses the VL-Bus target, the target does not assert LDEV# but completes the I/O port write anyway. The VL-Bus target also does not assert LRDY#. Since the VL-Bus controller does not receive LDEV# asserted, it passes the transfer along to the system I/O bus. The system I/O bus controller is then responsible for ending the cycle in the normal system I/O bus fashion. In order to facilitate VGA palette snooping by devices on the ISA bus, LDEV# must not be generated by any VL-Bus device for I/O write cycles to the palette addresses (0x3C6-0x3C9). If a VL-Bus VGA contains a palette mapped onto the VL-Bus, it should generate LDEV# for reads from the palette addresses and respond with data.

There are a few limitations to shadowing by the VL-Bus target:

- The VL-Bus target cannot add any extra wait states beyond what the system I/O bus normally adds.
- All transfers to the system I/O bus must also be seen by the VL-Bus targets, the transfers cannot be hidden from the VL-Bus once the system determines that the transfer is not for the VL-Bus.
- Since the VL-Bus target never claims ownership of the transfer, LBS16# is not available. Therefore, the VL-Bus target must be prepared to accept a 32-bit I/O port write transfer to any register that it allows shadowing.

An optional method allows the VL-Bus controller to decide which addresses should be shadowed. This method allows the system to be modified after the fact to account for future unforeseen incompatibilities. The VL-Bus target responds to the I/O port write as it normally would, asserting LDEV# and, at the appropriate time, asserting LRDY#. The VL-Bus controller, however, also passes the transfer to the system I/O bus. The system must then monitor the end of the transfer of both the VL-Bus controller and the system I/O bus controller. The transfer must not end until both controllers have ended their respective transfers. This method is optional and specific implementation is left to the discretion of the LBC designer. The recommended way is to control the range of shadowed I/O ports via a set of registers. These registers can be controlled by the system BIOS and set by the user in the setup configuration menu. Whatever method is chosen, the system must never require application software to have knowledge of the presence of the VL-Bus.

4. Bus Masters

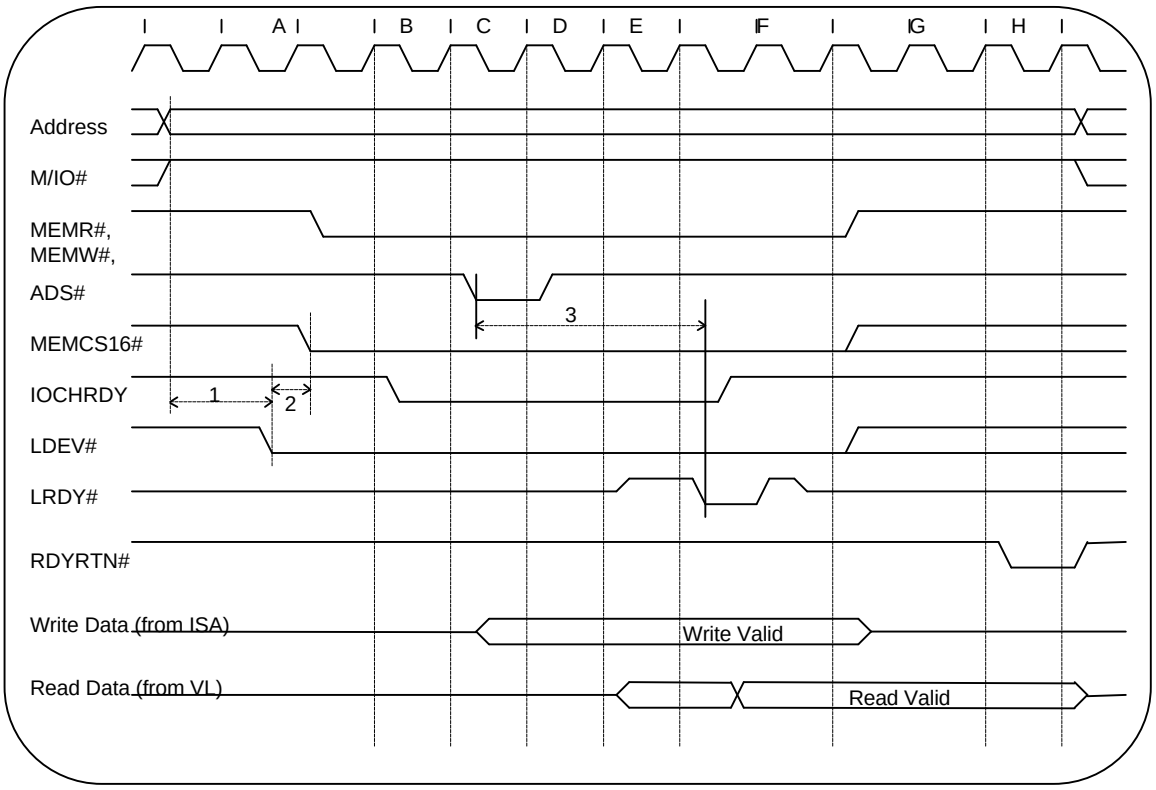
4.1. DMA and System I/O Bus Master Cycles (ISA)

4.1.1. DMA and System I/O Bus Master Memory cycles

The VL-Bus controller drives the ISA address to the VL-Bus for all DMA and system I/O bus master cycles. IOCHRDY is pulled negated for additional wait states as soon as both LDEV# is asserted by a VL-Bus target and the ISA-bus command strobe is asserted. Following the active ISA command signal, the VL-Bus controller drives ADS# asserted for one LCLK period.

When the VL-Bus target completes the command, it asserts LRDY# for one LCLK period. On the LCLK edge qualified by LRDY#, the VL-Bus controller releases IOCHRDY. The DMA controller or bus master can then terminate the pending ISA cycle. After the VL-Bus controller detects the end of the ISA command, it drives RDYRTN# low for one LCLK cycle. If the transfer is a read, the DMA or bus master has received the requested data. The VL-Bus target then stops driving the data on the data bus when detecting RDYRTN# asserted. If the VL-Bus controller latches in the data after receiving LRDY#, it may issue RDYRTN# even before the end of the DMA or bus master transfer since, the target no longer needs to drive the data bus for the remainder of the transfer.

M/IO# and W/R# signals are generated from the ISA-bus MEMR# and MEMW# signals. Since MEMR# and MEMW# appear late in the DMA or bus master cycle, M/IO# and W/R# should both be high early in the cycle, as if the cycle is a memory write cycle. Since the decode circuit for LDEV# uses M/IO#, LDEV# may not be stable until a maximum of 20ns after M/IO#, even though the system address lines have long been stable. The VL-Bus controller assumes that the current cycle is a memory write cycle and asserts MEMCS16# asynchronously with LDEV# low on the system I/O bus. If this assumption turns out to be incorrect (i.e., the cycle is actually an I/O transfer), asserting MEMCS16# does no harm. All DMA or system I/O bus master transfers are always assumed to be data transfers, therefore D/C# must always be pulled high by the VL-Bus Controller during any DMA or system I/O bus master transfers.



No.	Description	Min.	Max.	Units
1	Address valid to LDEV# asserted	0	20	nsec
2	LDEV# asserted to MEMCS16# asserted	0	25	nsec
3	ADS# asserted to LRDY# asserted	30	5000	nsec

Figure 4.1. DMA and System I/O Bus Master Memory Timing.

A. Address becomes valid. LDEV# is asserted because M/IO# is assumed to be high (memory). In this case the M/IO# assumption is correct. MEMR# (or MEMW#) is then driven asserted. Since MEMCS16# is asserted, the transfer can be 16 bits wide. If MEMCS16# is not asserted by the time the command strobe is asserted, the cycle falls back to an 8-bit transfer. During this period, the memory command strobe becomes asserted.

B. In phase B, the VL-Bus controller has detected the memory command strobe asserted. Since the command strobe is asynchronous to LCLK, one clock cycle is needed to synchronize the memory command strobe.

C. After the strobe is synchronized, the VL-Bus controller issues an ADS# pulse for one LCLK cycle.

D. If LDEV# is sampled negated, no VL-Bus target is claiming the cycle and no further action is required from the VL-Bus controller except to generate RDYRTN# at the end of the transfer if it generated ADS#. If LDEV# is sampled asserted, a VL-Bus target has claimed the current cycle. Since IOCHRDY has already been driven low, the VL-Bus controller must drive the data bus in the correct direction and wait for LRDY#. During this phase LRDY# is driven (either asserted or negated) by the active target.

E. When the VL-Bus target has completed the requested transfer, it pulls LRDY# low. If the current cycle is an memory read, the data must be on the data bus when LRDY# is asserted. If the current cycle is a memory write, the write must have been completed when LRDY# is asserted.

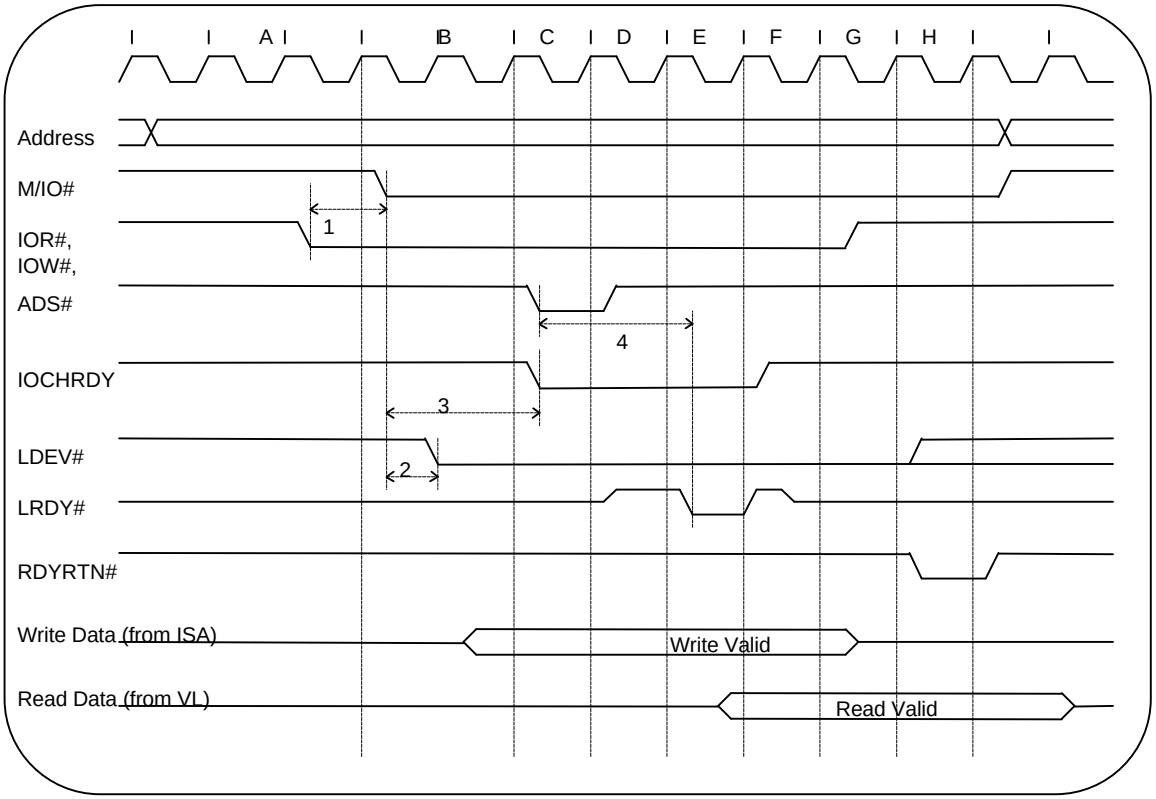
F. The VL-Bus controller detects LRDY# asserted and releases IOCHRDY on the ISA bus.

G. Some unspecified amount of time after IOCHRDY is asserted, the ISA master releases the memory command strobe.

H. After the memory command strobe is negated the VL-Bus controller asserts RDYRTN# for one LCLK cycle. RDYRTN# signals the LBT that the transfer is over. If the transfer is a read, the LBT may stop driving the data bus.

4.1.2. System I/O Bus Master I/O Cycle

I/O cycles from an ISA bus master are similar to the memory cycle described above. The principle difference is that since the M/IO# signal is decoded late, the transfer is not be recognized as an I/O cycle until after the ISA-bus command starts. Since this is too late to assert IOCS16#, all I/O bus masters involving a VL-Bus target are 8-bit transfers. IOCHRDY is also asserted later for I/O cycles, due to M/IO# being late. The IOCHRDY delay should not pose a serious problem, since 8-bit cycles tend to default to a much longer cycle then a 16-bit cycle, hence a relaxed IOCHRDY timing.



No.	Description	Min.	Max.	Units
1	ISA Command to M/IO# low	0	25	ns
2	M/IO# asserted to LDEV# asserted	0	20	ns
3	ISA Command to IOCHRDY low	0	65	ns
4	ADS# asserted to LRDY# asserted	30	5000	ns

Figure 4.2. System I/O Bus Master I/O Timing

A. Address becomes valid. LDEV# is not asserted because M/IO# is assumed to be high (memory). In this case M/IO# is assumed incorrect. IOR# (or IOW#) is then driven asserted. Since IOCS16# is not asserted, the transfer is eight bits. During this period, the I/O command strobe becomes asserted.

B. In phase B, the VL-Bus controller has detected the I/O command strobe asserted. Since the command strobe is asynchronous to LCLK, one extra LCLK cycle is needed to synchronize the I/O command strobe to the VL-Bus.

C. After the strobe is synchronized, the VL-Bus controller issues an ADS# pulse for one LCLK cycle.

D. If LDEV# is sampled negated, no VL-Bus target is claiming the cycle and no further action is required from the VL-Bus controller except to assert RDYRTN# at the end of the transfer if it generated ADS#. . If LDEV# is sampled asserted, a VL-Bus target has claimed the current cycle. Since IOCHRDY has already been driven low, the VL-Bus controller must drive the data bus in the correct direction and wait for LRDY#. During this phase LRDY# is asserted (either asserted or negated) by the active target.

E. When the VL-Bus target has completed the requested transfer it pulls LRDY# low. If the current cycle is an I/O read, the data must be on the data bus when LRDY# is asserted. If the current cycle is an I/O write, the write must have been completed when LRDY# is asserted.

F. The VL-Bus controller detects LRDY# asserted and releases IOCHRDY on the ISA bus.

G. Some unspecified amount of time after IOCHRDY is asserted, the ISA master releases the I/O command strobe.

H. After the I/O command strobe is negated the VL-Bus controller asserts RDYRTN# for one LCLK cycle to signal the LBT that the transfer is over and, if the cycle was a read, the LBT may stop driving the data bus.

4.2. Local Bus Masters (LBMs)

4.2.1. Signals Driven by an Active LBM.

The active LBM takes the responsibility for driving the following list of signals while it is granted the VL-Bus:

- LREQ<x># - must remain asserted while the active LBM controls the VL-Bus.
- ADS# - same timing as a 486.
- ADR<31..02>, BE<3..0>#, M/IO#, W/R# - same timing as a 486.
- LEADS# - same timing as ADS# when the active LBM controls the VL-Bus.
- D/C# - if LBM does not support this feature, it must be driven high.
- BLAST# - if LBM does not support this feature, it must be driven low.

4.2.2. Arbitration Protocol

The VL-Bus can support up to three local bus bus-masters (LBMs). Control is arbitrated by the VL-Bus controller by a pair of Request/Grant signals. Each slot has its own signal pair. When LREQ<x># is asserted by a potential LBM, the VL-Bus controller replies with LGNT<x>#. After the requesting VL-Bus device receives LGNT<x>#, that device becomes the active LBM and has control of the VL-Bus. After the active LBM has completed its transfers, it returns control by deasserting LREQ<x>#. The VL-Bus controller acknowledges this by removing LGNT<x>#. LREQ<x># should be deasserted on the clock when the final ready (RDYRTN# or BRDY#) is received for the final bus cycle.

If a higher-priority device wants control of the VL-Bus, the VL-Bus controller may preempt the active LBM by removing LGNT<x># from that slot. The active LBM must then relinquish the bus as soon as practical by removing LREQ<x>#. The active LBM should release the bus at the end of the current cycle or, in the case of locked cycles, after the last unlocked cycle. The LBM relinquishing the bus may, after a minimum two LCLK cycles, reassert LREQ<x># to regain control of the VL-Bus.

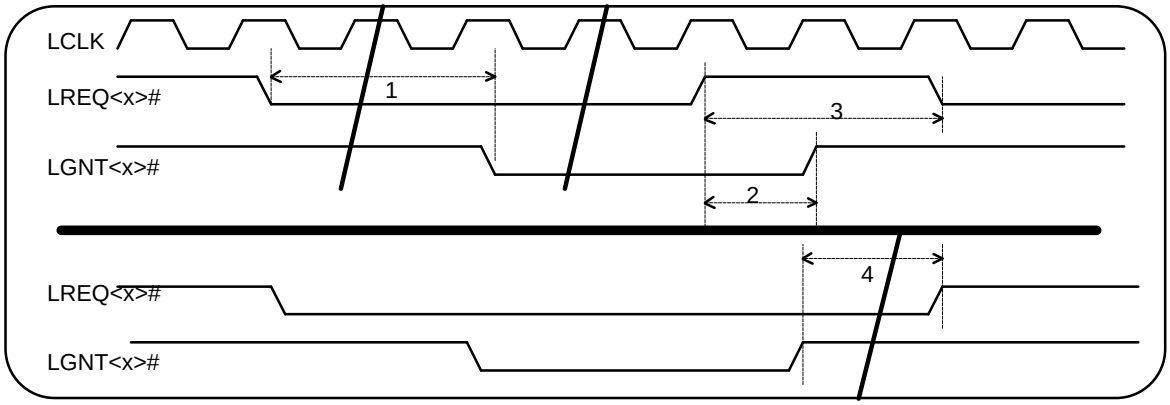
4.2.3. Arbitration Timing

Figure 4.3 shows the timing a VL-Bus Master and VL-Bus controller must meet. The preempt acknowledge allows for up to 5 usec delay; the active VL-Bus master should attempt to complete the current transfer (or set of transfers if locked cycles) and then relinquish control of the VL-Bus back to the VL-Bus controller. A bus master that does not support preempting may request control of the VL-Bus but must always release the bus within 5 usec. The normal maximum latency between a VL-Master asserting LREQ# and the system controller granting the bus by asserting LGNT# is 20uS. There will be times where this latency is exceeded, specifically if an ISA master holds the bus for long periods of times; many wait states are added to ISA bus accesses by the CPU, etc. 20uS is a good number to be used in determining FIFO size, but a design should be aware that it could be exceeded. The card could presumably do a retry in these cases. The system controller should be designed such that a VL-Master is served within 20uS the vast majority of the time, as far as it has control.

The VL master may start driving the address and control busses on the clock edge where it samples LGNT<x># low. It must float the address and control busses on the clock edge where it drives LREQ<x># high. This applies for normal and preempted terminations. In Figure 4.4 the top diagram shows the VL master giving up the bus by taking LREQ<x># high, while the

bottom diagram shows the Local Bus Controller preemping the master. A third case is possible, where the master coincidentally gives up the bus on the same clock as the local bus controller begins preempting it. The timing would be identical to the top diagram except LGNT<x># would go high one clock earlier.

The VL Master should not take LREQ# high until the final ready (RDYRTN# or BRDY#) of the last bus cycle is received as shown in Figure 4.4.



No.	Description	Min.	Max.	Units
1	LREQ# asserted to LGNT# asserted	1		LCLK
2	LREQ# negated to LGNT# negated	1	2	LCLK
3	LREQ# negated to LREQ# asserted	2	-	LCLK
4	LGNT# negated to LREQ# negated	-	5	usec

Figure 4.3. LBM Arbitration Timing.

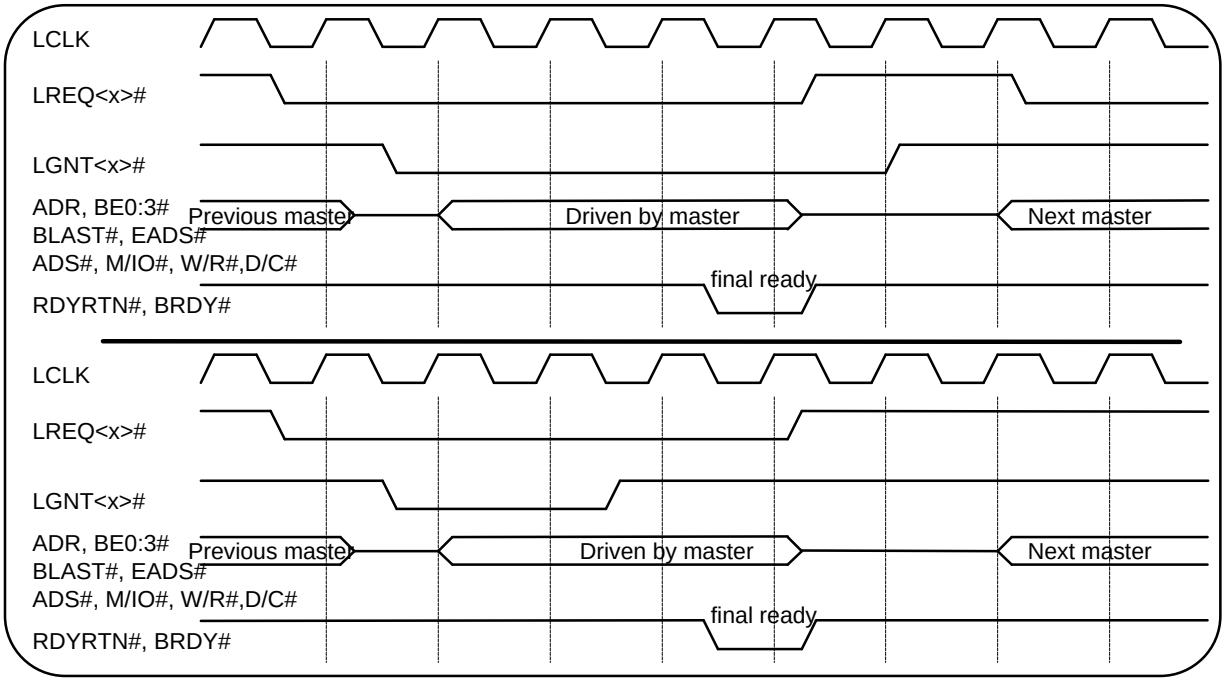


Figure 4.4. LBM Bus driving Protocol.

5. Timing

5.1. LCLK Skew

Controlling clock skew between components is critical for a reliable system. The clock skew between components, including motherboard components and VL connectors should be within the following limits:

Max LCLK	Max skew
<= 40 MHz	2ns
>= 50 MHz	1ns

Table 5.1. CPU Clock to LCLK Skew.

5.2. Other LCLK Timing Parameters

LCLK must have a high time of no less then 40 percent and no greater then 60 percent of the total clock period. Low time must be no less then 40 percent and no greater then 60 percent of the total clock period. The clock enters the high state at 2.0 volts and enters the low state at 0.8 volts. Maximum rise time (0.8 to 2.0 volts) and fall time (2.0 to 0.8 v) is 2ns.

5.3. Signal Timing to LCLK

The VL-Bus AC timing specification is divided into 33MHz, 40MHz, and 50MHz. Maximum loading (CL_{MAX}) of each signal is defined, but minimum loading of each signal varies from system to system. Each VL-Bus card is allowed 25pF on each signal.

The following timing is relative to each VL-Bus connector, and has been divided between the driver of the signal and the receiver of the signal. The "valid delay" and "float delay" timings refer to the driver, and the "setup time" and "hold time" refer to the receiver. The "valid delay, minimum" implies the hold time at the driver from the previous cycle. The margin between the "valid delay, maximum" of the driver and the "setup time" of the receiver allows for clock skew between the VL-Bus connectors.

Add-in cards must meet the Min and Max delay times when driving a signal, and they are guaranteed the minimum setup and hold times when receiving a signal. All timings are with respect to the clock at the specific add in card connector. Add in cards must assume that any load up to the maximum capacitance for each signal may be present, since they must work in all types of systems. It should not be assumed that all signals are loaded equally.

Motherboards, when driving a signal, are responsible for providing setup and hold times at each connector, with respect to the LCLK at that connector. Motherboards are not required to meet specified delay times. This is because the motherboard has control of such things as the CPU type, the clock skew, trace length, loading, etc. This leeway allows motherboard caches, whose timing is critical, and some CPU generated signals to meet the less restrictive setup and hold timing at the VL-Bus connector. Motherboards must be capable of driving their own load plus up to 25pF per VL-Bus card. The VL-Bus sockets are considered part of the motherboard load. When receiving a signal from a VL-Bus slot, motherboards may assume the signal is valid at the VL-Bus connector at the maximum delay time for that signal with respect to the LCLK of

that connector. Loading and timing requirements must be met with any valid upgrade combinations installed in the motherboard, such as memory, coprocessor, or CPU upgrade.

LCLK minimum high and low times are 40% of the cycle time. All signal timing is referenced at 1.5V. Valid delay is for tri-state signals; this defines the time from LCLK to the proper logic level when driving from a high impedance or a previous logic level. Active delay is for totem pole outputs; this defines the time from LCLK to the new logic level. Float delay also refers to tri-state signals; this defines the time from LCLK to the tri-state level. Refer to the timing signal figures for further reference.

LDEV# is defined to be valid 20nS after the address, M/IO#, and in some cases D/C# become valid. Since it is monitored only by the VL-Bus controller, setup and hold times are not specified and it is left to the systems designer to determine when LDEV# may be sampled.

SIGNAL NAME	T#	PARAMETERS	DRV(O) min	DRV(O) max	RCV(I) min	RCV(I) max	CL Max	NOTES
LCLK	t1	Period	30	30			50pf	0.1% stability
	t2	High time	12					At 2.0v
	t3	Low time	12					At 0.8v
	t4	Rise time		3				
	t5	Fall time		3				
ADR2-ADR31	t6	Valid delay	3	18			125pf	
M/IO#, W/R#,	t7	Float delay		20				
BE<X>#, D/C#	t8	Setup time			7			
	t9	Hold time			2			
ADS#,	t10	Valid delay	3	18			125pf	
LBS64#,	t11	Float delay		20				
ACK64#	t12	Setup time			7			
	t13	Hold time			2			
BLAST#	t14	Valid delay	3	18			100pf	
	t15	Float delay		20				
	t16	Setup time			7			
	t17	Hold time			2			
DAT0-DAT31	t18	Valid delay	3	18			125pf	
	t19	Float delay		20				
	t20	Setup time (read)			5			
	t21	Setup time (write)			7			
	t22	Hold time			2			
LEADS#	t23	Valid delay	3	18			100pf	
	t24	Float delay		20				
	t25	Setup time			7			
	t26	Hold time			2			
LBS16#	t27	Valid delay	3	18			100pf	BS16# must be driven
	t28	Float delay		30				high before floating.
	t29	Setup time			7			
	t30	Hold time			2			
LGNT<x>#	t32	Setup time			7		50pf	LGNT# 50pf
WBACK#	t33	Hold time			2		100pf	WBACK# 100pf
LREQ<x>#	t34	Active delay	3	18			50pf	
	t35	Setup time			7			
	t36	Hold time			2			
BRDY#	t37	Valid delay	3	18			100pf	BRDY# must be
	t38	Float delay		30				driven high before
	t39	Setup time			5			floating.
	t40	Hold time			2			
LRDY#	t41	Valid delay	3	14			100pf	LRDY# must be
	t42	Float delay		30				driven high before
	t43	Setup time			5			floating.
	t44	Hold time			2			
LRDY#	t45	Valid delay	3	17			175pf	LRDY# must be driven
	t46	Float delay		30				high before floating. If
	t43	Setup time			5			LRDY# and RDYRTN# are
	t44	Hold time			2			tied together, LRDY# is 175pf
RDYRTN#	t47	Setup time			5		100pF	
	t48	Hold time			2			
LDEV#	t49	Active delay	3	20			50	From address change

Table 5.2. 33 MHz VL-Bus A.C. Characteristics

SIGNAL NAME	T#	PARAMETERS	DRV(O) min	DRV(O) max	RCV(I) min	RCV(I) max	CL Max	NOTES
LCLK	t1	Period	25	25			50pf	0.1% stability
	t2	High time	10					At 2.0v
	t3	Low time	10					At 0.8v
	t4	Rise time		3				
	t5	Fall time		3				
ADR2-ADR31	t6	Valid delay	3	18			125pf	
M/IO#, W/R#,	t7	Float delay		19				
BE<X>#, D/C#	t8	Setup time			5			
	t9	Hold time			2			
ADS#	t10	Valid delay	3	18			125pf	
LBS64#,	t11	Float delay		19				
ACK64#	t12	Setup time			5			
	t13	Hold time			2			
BLAST#	t14	Valid delay	3	18			100pf	
	t15	Float delay		19				
	t16	Setup time			5			
	t17	Hold time			2			
DAT0-DAT31	t18	Valid delay	3	18			125pf	
	t19	Float delay		19				
	t20	Setup time			5			
	t22	Hold time			2			
LEADS#	t23	Valid delay	3	18			100pf	
	t24	Float delay		19				
	t25	Setup time			5			
	t26	Hold time			2			
LBS16#	t27	Valid delay	3	17			100pf	BS16# must be driven
	t28	Float delay		25				high before floating.
	t29	Setup time			5			
	t30	Hold time			2			
LGNT<x>#	t32	Setup time			5		50pf	LGNT# 50pf
WBACK#	t33	Hold time			2		100pf	WBACK# 100pf
LREQ<x>#	t34	Active delay	3	17			50pf	
	t35	Setup time			5			
	t36	Hold time			2			
BRDY#	t37	Valid delay	3	17			100pf	BRDY# must be
	t38	Float delay		25				driven high before
	t39	Setup time			5			floating.
	t40	Hold time			2			
LRDY#	t41	Valid delay	3	14			100pf	LRDY# must be
	t42	Float delay		25				driven high before
	t43	Setup time			5			floating.
	t44	Hold time			2			
LRDY#	t45	Valid delay	3	17			175pf	LRDY# must be driven
	t46	Float delay		25				high before floating. If
	t43	Setup time			5			LRDY# and RDYRTN# are
	t44	Hold time			2			tied together, LRDY# is 175pf
RDYRTN#	t47	Setup time			5		100pF	
	t48	Hold time			2			
LDEV#	t49	Active delay	3	20			50	From address change

Table 5.3. 40 MHz VL-Bus A.C. Characteristics

SIGNAL NAME	T#	PARAMETERS	DRV(O) min	DRV(O) max	RCV(I) min	RCV(I) max	CL Max	NOTES
LCLK	t1	Period	20	20			50pf	0.1% stability
	t2	High time	8					At 2.0v
	t3	Low time	8					At 0.8v
	t4	Rise time		2				
	t5	Fall time		2				
ADR2-ADR31	t6	Valid delay	3	14			75pf	
M/IO#, W/R#,	t7	Float delay		18				
BE<X>#, D/C#	t8	Setup time			5			
	t9	Hold time			2			
ADS#	t10	Valid delay	3	14			75pf	
LBS64#,	t11	Float delay		18				
ACK64#	t12	Setup time			4			
	t13	Hold time			2			
BLAST#	t14	Valid delay	3	14			75pf	
	t15	Float delay		18				
	t16	Setup time			4			
	t17	Hold time			2			
DAT0-DAT31	t18	Valid delay	3	14			75pf	
	t19	Float delay		18				
	t20	Setup time			4			
	t22	Hold time			2			
LEADS#	t23	Valid delay	3	14			75pf	
	t24	Float delay		18				
	t25	Setup time			4			
	t26	Hold time			2			
LBS16#	t27	Valid delay	3	14			75pf	BS16# must be driven
	t28	Float delay		23				high before floating.
	t29	Setup time			4			
	t30	Hold time			2			
LGNT<x>#	t32	Setup time			4		50pf	LGNT# 50pf
WBACK#	t33	Hold time			2		75pf	WBACK# 75pf
LREQ<x>#	t34	Active delay	3	14			50pf	
	t35	Setup time			4			
	t36	Hold time			2			
BRDY#	t37	Valid delay	3	13			75pf	BRDY# must be
	t38	Float delay		23				driven high before
	t39	Setup time			4			floating.
	t40	Hold time			2			
LRDY#	t41	Valid delay	3	13			75pf	LRDY# must be
	t42	Float delay		23				driven high before
	t43	Setup time			4			floating.
	t44	Hold time			2			
RDYRTN#	t47	Setup time			5		100pF	
	t48	Hold time			2			
LDEV#	t49	Active delay	3	20			50	From address change

Table 5.4. 50 MHz VL-Bus A.C. Characteristics

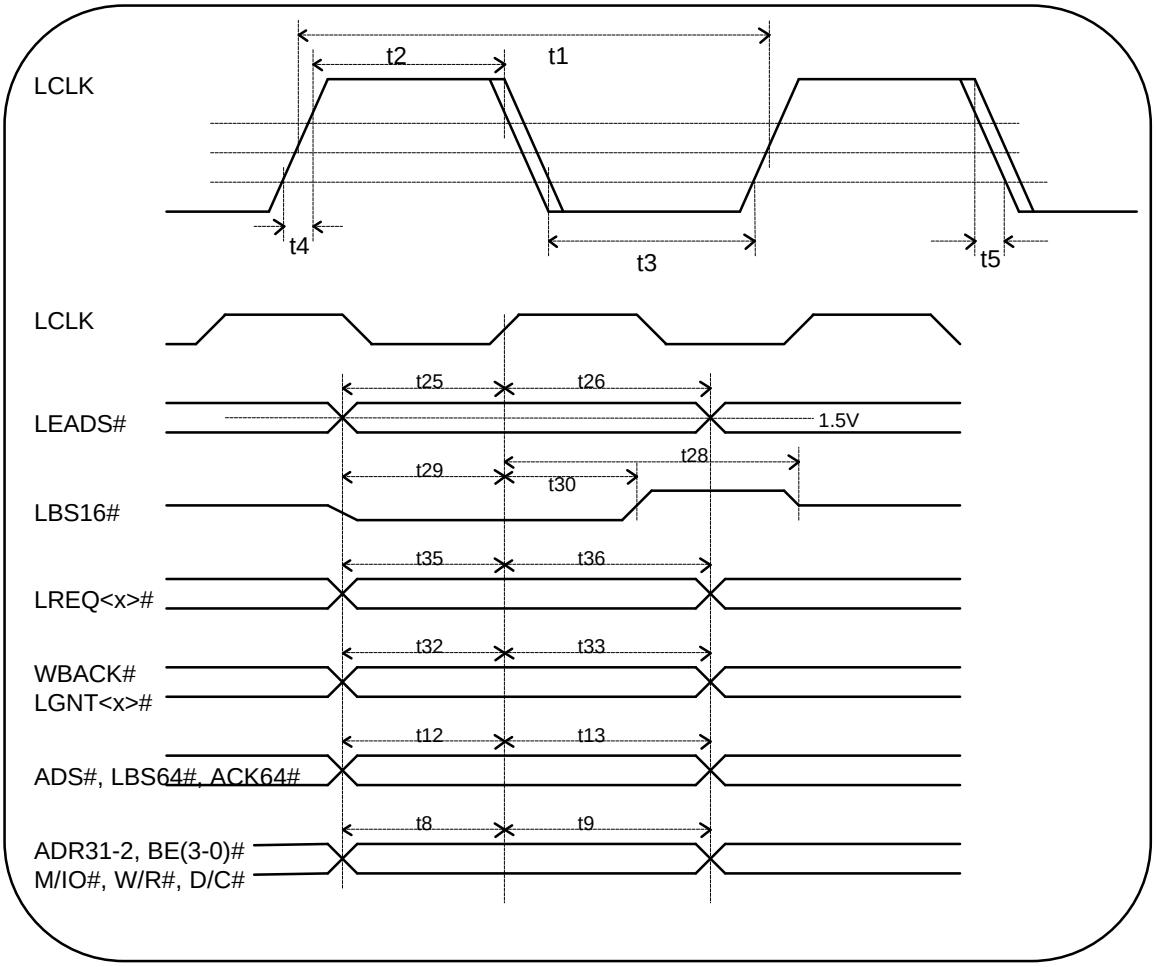


Figure 5.1. Input Setup And Hold Timing

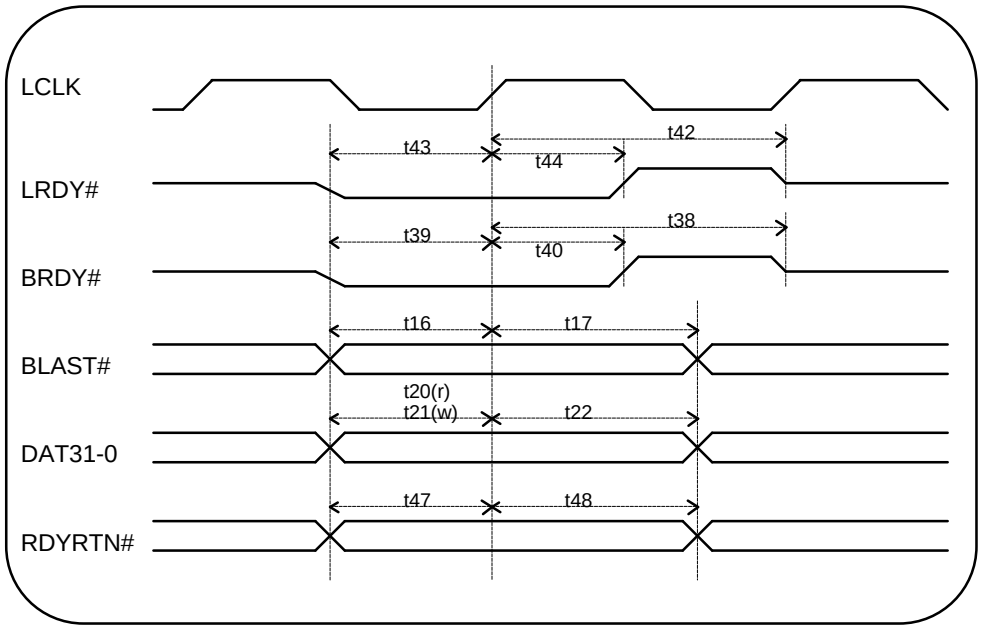


Figure 5.2. Input Setup And Hold Timing

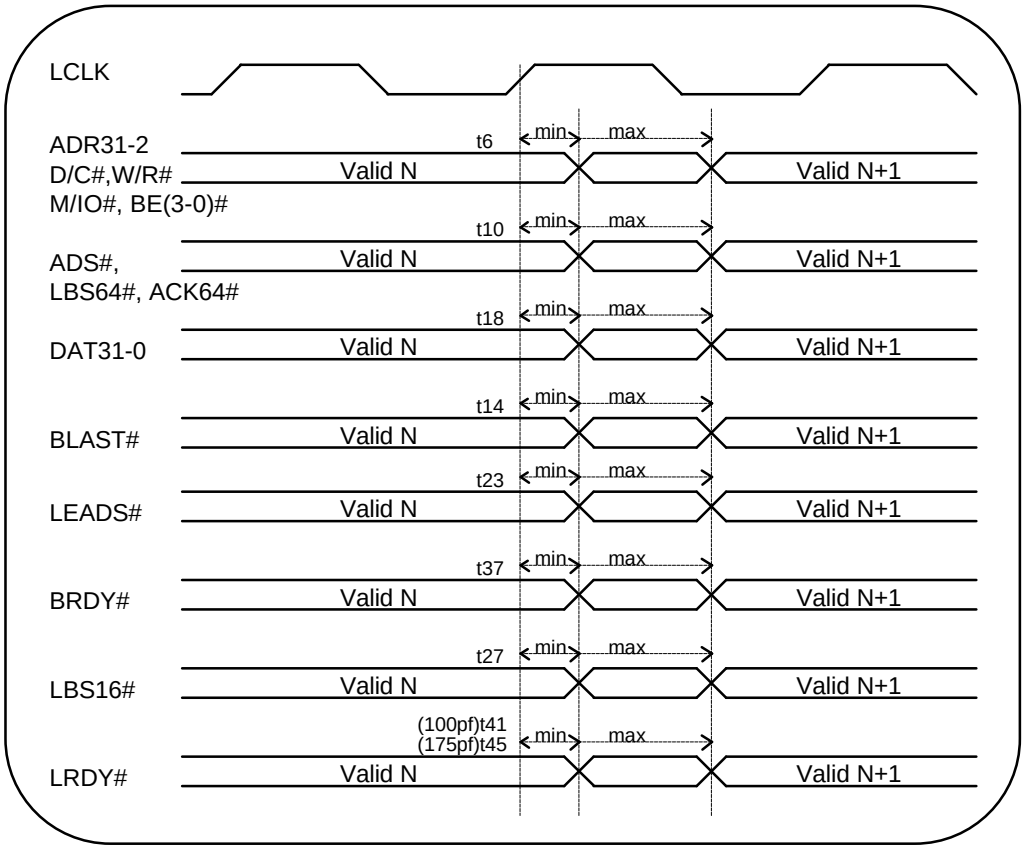


Figure 5.3. Output Valid Delay Timing

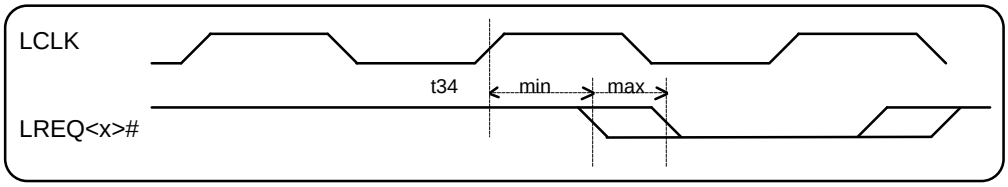


Figure 5.4. Output Active Delay Timing

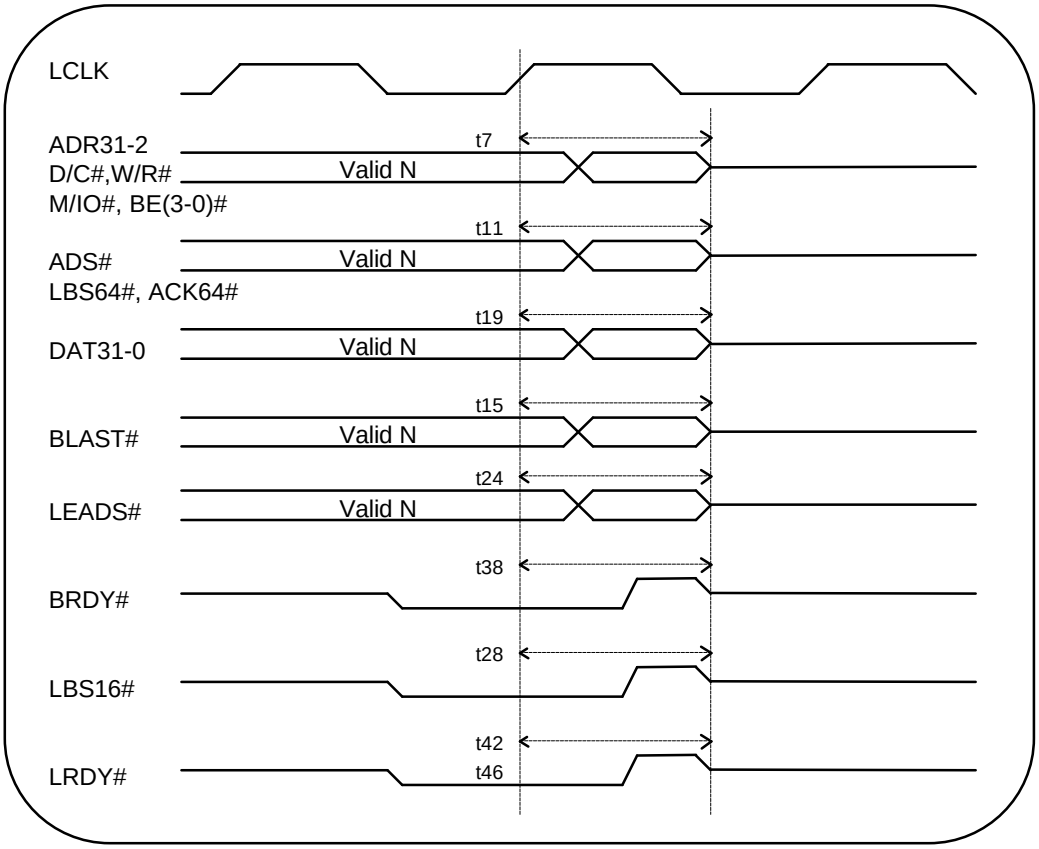


Figure 5.5. Output Float Delay Timing

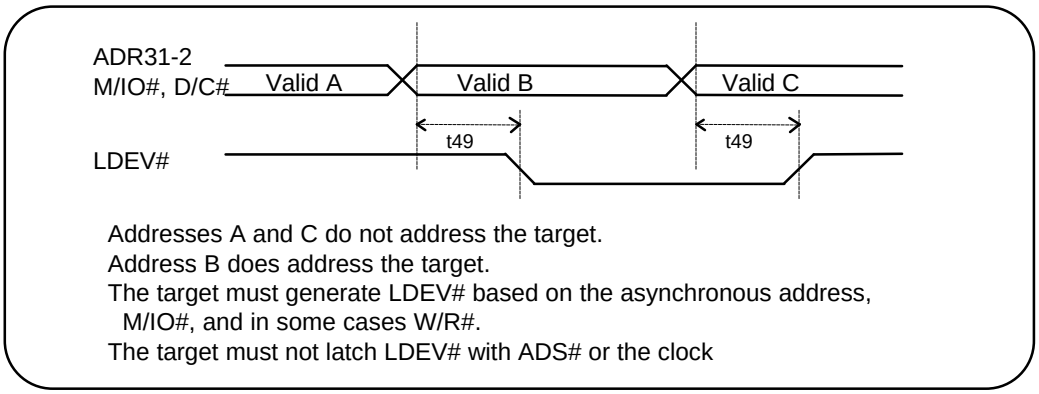


Figure 5.6. LDEV# Timing

6. 64-bit VL-Bus

The 64-bit wide VL-Bus allows systems based on 64 bit processors to effectively double the available local bus bandwidth while still allowing existing 32-bit VL-Bus boards to coexist within the system. The 64-bit VL-Bus uses the existing connector to extend VL-Bus to 64-bits. Only 2 extra signals (LBS64# and ACK64#) are needed to extend the VL-Bus to 64-bits. A multiplexing scheme is employed that will not affect the installed base of 32-bit motherboards and target devices. 32 bit devices should not require the address, control (M/IO# and W/R#), byte enables, and ID signals to be stable during VL bus accesses to other devices.

The minimum wait states for 64 bit cycles are as follows:

Read Cycles: 5 cycles (2 wait states plus 1 turnaround cycle)
Write Cycle: 3 cycles (1 wait state, no turnaround cycle)
Burst Read: 4-1-1-1 plus 1 turnaround cycle at the end.
Burst Write 3-1-1-1 (no turnaround cycle required).

6.1. Additional Signal Definitions for 64-bit Transfers

6.1.1. ID<4,1..0> - Identification Bits

Identification bits 0, 1 and 4 identify the motherboard as 64-bit aware during the reset period. 64-bit aware means that either the motherboard is capable of a 64-bit transfer or it will allow a bus master and target to accomplish a 64-bit transfer. The latter will require the motherboard to tri-state the ID bits when RESET# is not asserted.

All 64-bit aware motherboards must also support VL-Bus 2.0 read and write burst protocol (which does not necessarily mean that they actually burst). When ID bits 0, 1, and 4 are all high during the reset period, the VL-Bus controller supports read and write burst protocol and is 64-bit aware.

ID1	ID0	ID4	Protocol	Bus size awareness
0	0	X	Reserved	reserved
0	1	X	386	16,32
1	0	X	486	16,32
1	1	1	486	16,32,64

Table 6.1. Identification Bits

6.1.2. DAT<63..32> - Data Bus

DAT<63..32> is the high-order double word of the data bus. DAT<63..34> are multiplexed onto ADR<31..2> address bus. DAT<33..32> are multiplexed onto M/IO# and W/R#.

6.1.3. BE<7..4> - Byte Enables

The byte enables indicate which byte lanes of the 64-bit data bus are involved with the current VL-Bus transfer. BE<7..4> are the byte enables controlling DAT<63..32> in the same manner as BE<3..0> control DAT<31..0>. BE<7..4> is muxed onto BE<3..0>. Consecutive bytes must always be transferred both during a burst and non-burst transfer (See the 64 bit master rules for more details).

Byte Enable	Associated Data Bus
BE0#	DAT<07..00>
BE1#	DAT<15..08>
BE2#	DAT<23..16>
BE3#	DAT<31..24>
BE4#	DAT<39..32>
BE5#	DAT<47..40>
BE6#	DAT<55..48>
BE7#	DAT<63..56>

Table 6.2. Byte Enables

6.1.4. LBS64# - Bus Size 64.

LBS64# is asserted with ADS# by a 64-bit VL-Bus controller or active 64-bit VL-Bus master to advise the target that a 64-bit transfer is possible. This is a shared signal that occupies a former unconnected slot pin. LBS64# is active for only 1 clock cycle and has the same timing as ADS#. This assures that LBS64# is driven high by the master before giving up the bus. A 39K pull-up is placed on each 64-bit VL-Bus add-in board to ensure LBS64# will remain high while undriven. A 64-bit bus master should never drive LBS64# if the motherboard is not 64-bit aware. A 32-bit motherboard that is 64-bit aware will tristate its ID signals following reset.

6.1.5. ACK64# - Acknowledge 64.

ACK64# is asserted by the 64-bit target to inform the VL-Bus controller that a 64-bit transfer is in progress. ACK64# is a shared signal is driven by an active 64-bit target device. It is an input for an active 64-bit bus mastering device. ACK64# is driven during either the first T2 or 2nd T2 state, depending on the decode speed of the target. It is a synchronous signal, driven only by the addressed target, and therefore must be driven only after the target's decode is stable. ACK64# is driven low for 1 T state, driven back high for 1 T state, then floated. Neither LRDY# or BRDY# may be driven low before or while ACK64# is low.

ACK64# is multiplexed onto the ID<4> slot pin. ID<4> will be tri-stated on the motherboard following reset. If LBS64# is not asserted by the VL-Bus controller, the target must not drive ACK64#. A 39K pull-up is placed on the 64-bit motherboard and 64-bit bus master device to maintain a high level on the ACK64# signal when the active target is a 32-bit device.

6.2. 64-bit Signal Pin Locations

Table 6.3 shows the pin locations of the 64-bit signal extensions. The 32-bit use column shows the name of the signal when in 32-bit mode.

Signal Name	32 Bit Use	Pin Number	Source
DAT32	W/R#	B45	Target (read) or Controller (write)
DAT33	M/IO#	B44	Target (read) or Controller (write)
DAT<63..34>	ADR<31..2>	(many)	Target (read) or Controller (write)
BE<7..4>#	BE<3..0>#	A-39,41,42,45	Controller
LBS64#	n/c	B-41	Controller
ACK64#	ID<4>	A-56	Target

Table 6.3. 64-bit Pin Locations

6.3. 64-Bit Timing

The 64-bit extension is designed such that existing 32-bit cards will not be affected by the inclusion of the 64-bit extension. Any 64-bit bus masters must check ID<4, 1, 0> during the reset period to identify the motherboard as 64-bit. If the motherboard is not 64-bit aware, a 64-bit transfer must never be attempted. The 64-bit extension is kicked off by the LBS64# / ACK64# handshake.

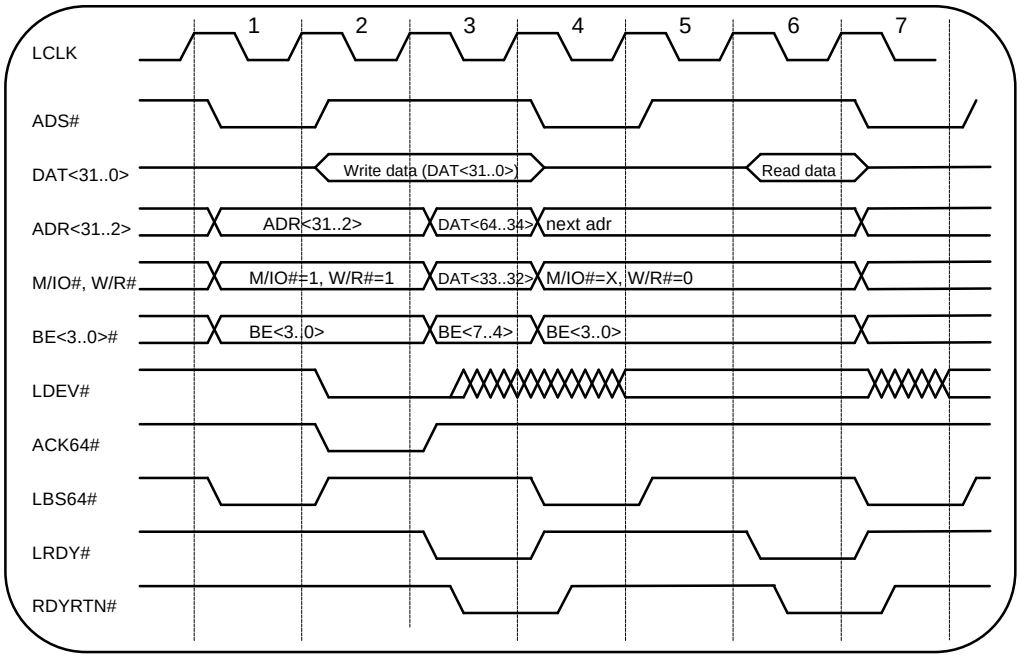


Figure 6.1. 64-Bit write transfer timing.

Figure 6.1 shows the fastest 64 bit write cycle (3 clocks). The following describes each clock cycle.

1 ADS# and LBS64# are driven low together since the master is 64 bit capable and the bus cycle may be done as 64 bit.

2 ACK64# generated by target to convert the cycle to 64 bit. DAT<31..0> are driven with valid write data as in normal 32 bit cycle.

3 ADR<31..2>, M/IO#, and W/R# switch to DAT<64..32> in response to ACK64#. ACK64# goes back high. LRDY# is generated.

4-6 Since this was a write cycle the next cycle may start immediately. The master attempts a 64 bit cycle, the target is 32 bit.

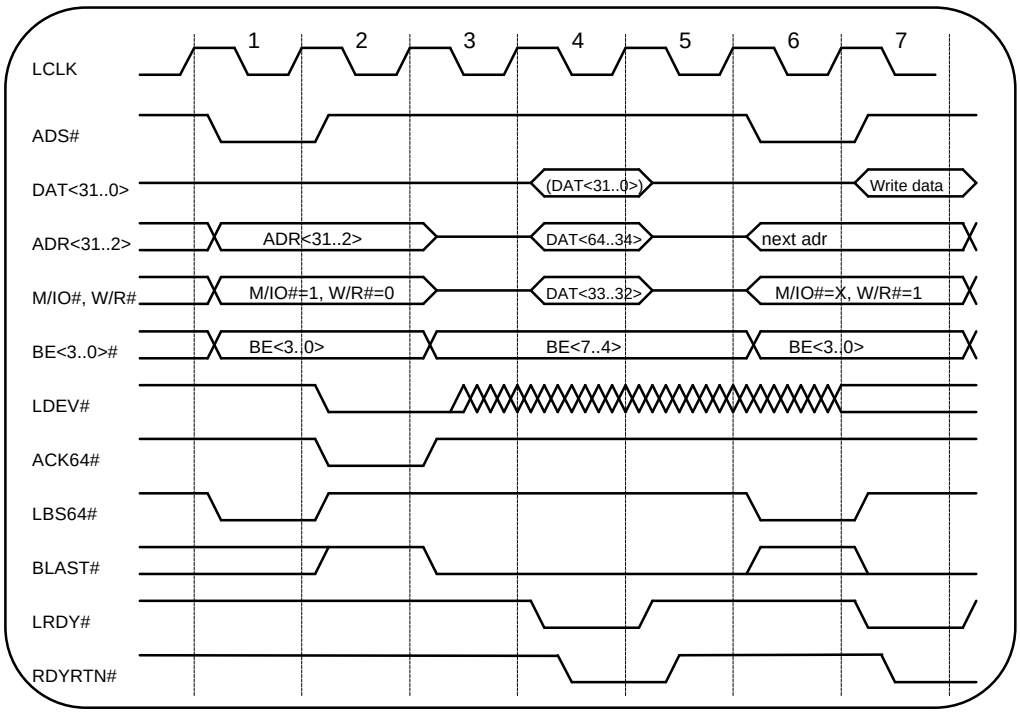


Figure 6.2. 64-Bit read transfer timing.

Figure 6.2 shows the fastest 64 bit read cycle (5 clocks). The following describes each clock cycle.

1 ADS# and LBS64# are driven low together since the master is 64 bit capable and the bus cycle may be done as 64 bit.

2 BLAST# is high since the master would attempt to burst if the slave is 32 bit. ACK64# generated by target to convert the cycle to 64 bit.

3 ADR<31..2>, M/IO#, and W/R# float in response to ACK64#. BLAST# goes low indicating that the master will not burst to the 64 bit target. ACK64# goes back high. LDEV# becomes invalid in response to the address change. It is ignored during the 64 bit transfer. The target may not drive DAT<63..32> during this clock cycle.

4 The target drives the data and drives LRDY# low.

5 Since this is a read cycle, a turnaround cycle is required to avoid a bus contention between the data driven by the target and the next address driven by the master.

6-7 The master attempts a 64 bit cycle, the target is 32 bit. A minimum length 32 bit write is shown.

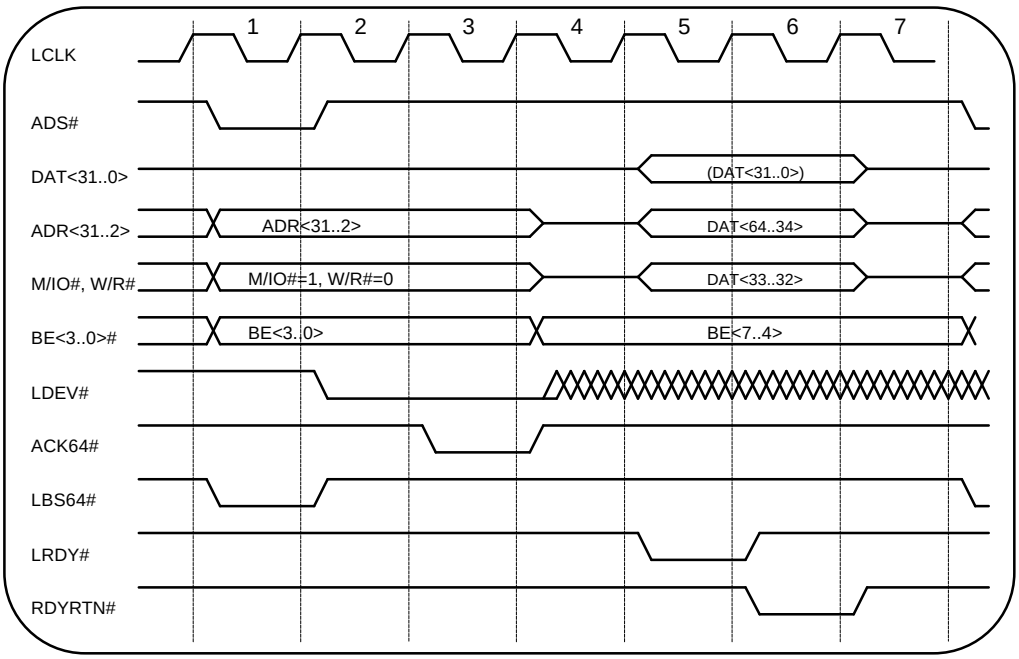


Figure 6.3. Additional Cases.

Figure 6.3 shows two separate variations from the normal cycle. ACK64# is generated one cycle later than in figures 6.1 and 6.2. This may be done if the target requires more decode time. The master does not switch the address, M/IO#, and W/R# until the clock where it receives ACK64# low. At the end of the cycle, RDYRTN# is delayed by one cycle from LRDY#. The target holds the data until the clock where RDYRTN# is sampled active.

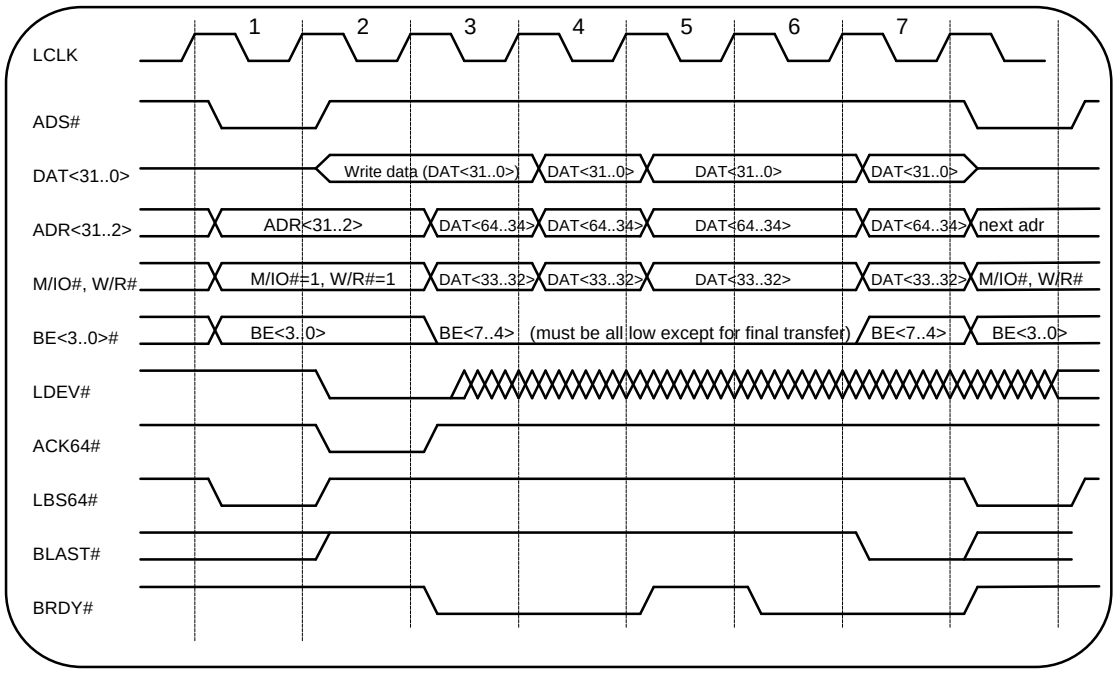


Figure 6.4. 64-Bit burst write.

Figure 6.4 shows a burst write. As in a 32 bit burst, new data is driven following each BRDY#. BE<7..4> are all active on each data transfer with the possible exception to the final one. BE<3..0># are assumed active for all transfers after the first. This fulfills the requirement that

consecutive bytes be written with no holes. Wait states may be added to any transfer. A wait state is shown added to the 3rd data transfer.

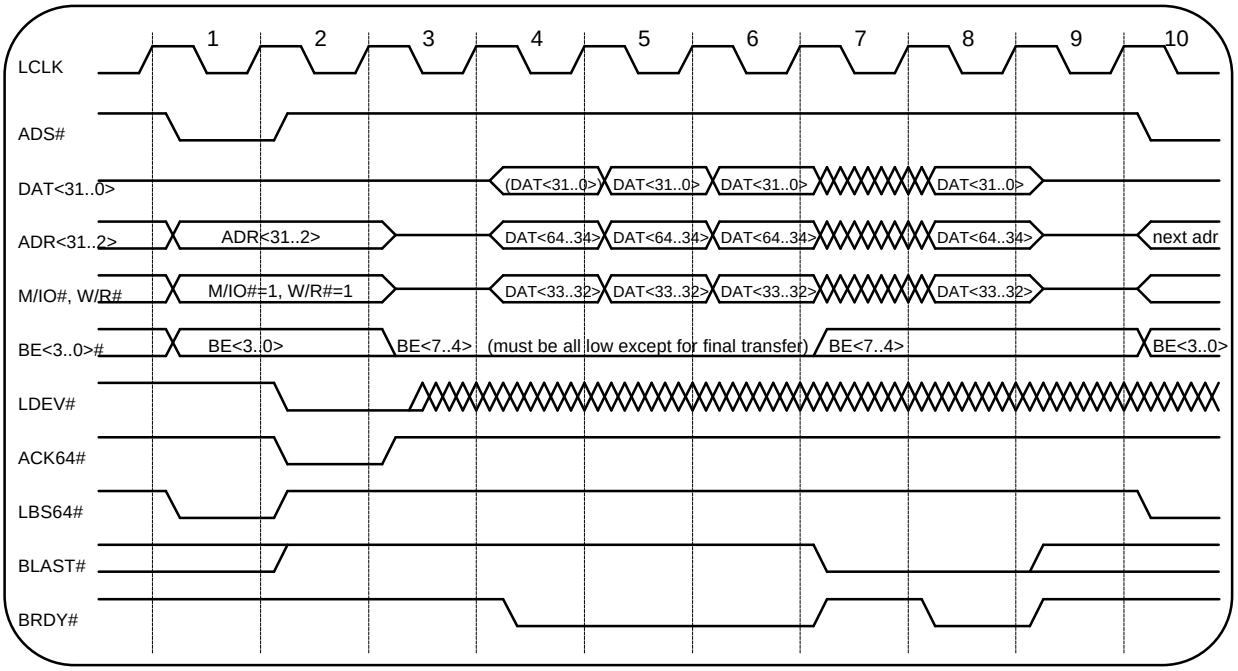


Figure 6.5. 64-Bit burst read.

Figure 6.5 shows a burst read. It is identical to the 64 bit write except that a turnaround cycle is required at the beginning and end of the data transfers. A wait state is shown added to the 4th data transfer.

6.3.1. 64 Bit Master Rules

1 The Master may request a 64 bit transfer by driving LBS64# low with ADS# if all of the following are true:

The transfer will straddle a 32 bit DWORD. This implies that at least BE3# and BE4# are both active.

The cycle is a memory read, code read, or memory write.

2. LBS64# must not be driven unless ID4, ID1, and ID0 were all sampled as 1s when RESET# went inactive.
3. Contiguous bytes must be transferred, and at least 1 byte enable on the 32 bit bus must be active (which implies BE3#). If none of the byte enables on the 32 bit bus would be active, the cycle must be done as a 32 bit transfer, with A2=0, BE<7..4># moved to BE<3..0>, and the data steered to DAT<31..0>. The legal byte enable combinations are shown below.

		Burst Transfer		
	Single Transfer	First Transfer	Middle Transfers	Final Transfer
BE0#	X	X	(0)	(0)
BE1#	X	X	(0)	(0)
BE2#	X	X	(0)	(0)
BE3#	0	0	(0)	(0)
BE4#	0	0	0	X
BE5#	X	0	0	X
BE6#	X	0	0	X
BE7#	X	0	0	X

Table 6.4. Legal byte enable combinations for 64 bit transfers.

0 Byte enable active.
X Either high or low (contiguous bytes must be transferred, however)
(0) Byte enable implied active. BE<3..0># are not available after first transfer of burst.

4. A2 must be 0 for a 64 bit transfer.
5. Bursts start with A3=0 and burst upwards except for the system board, which may do cache line ordering if the CPU requests it.
6. If RDYRTN# or BRDY# is received before ACK64#, the transfer is a normal 32 bit transfer.
7. If RDYRTN# or BRDY# is received after ACK64#, the transfer is 64 bit.
8. ACK64# will never be received on the same clock edge as BRDY# or RDYRTN#.
9. ACK64# should be expected at the end of the first T2 or at the end of the second T2.
10. When LBS64# was driven, the master must switch the address bus, M/IO#, and W/R# over to data on the clock edge when ACK64# is received. For read cycles this means floating them, for writes it means driving valid write data. BLAST# must also be converted from being based on a 32 bit target to being based on a 64 bit target on the clock edge when ACK64# is received.
11. A burst may be 2, 3, or 4 data transfers for either reads or writes.
12. When passing a CPU cache line fill to the VL-Bus, the VL-Bus controller (CPU bridge logic) is responsible for driving the byte enables all active if necessary. This is NOT required for a 32 bit cycle.
13. LBS16# should be ignored by the master during 64 bit transfers.

6.3.2. 64 Bit Target Rules

1. ACK64# must not be driven unless LBS64# was driven low with ADS#.
2. LBS64# must be ignored unless ID4, ID1, and ID0 were all sampled as 1s when RESET# went inactive.
3. ACK64# must be driven synchronously during the first T2 or second T2, depending on the decode time requirements of the target.
4. ACK64# must not be driven after or while driving LRDY# or BRDY# low.
5. Targets must be able to follow any combination of the burst order described below if they burst. A3 and A4 must be generated internally during a burst since they are not available on the bus.
6. A target may respond to any request as a 32 bit target.
7. A 64 bit target must be capable of operating as a 32 bit target.

8. A target should not drive LBS16# low during a 64 bit transfer.

6.3.3. Burst cycles

Burst transfers are two, three, or four 64-bit transfers. Table 6.5 shows the 64-bit burst using cache line fill ordering, and which combinations may be initiated by a VL-Bus master. All combinations must be accepted by a 64 bit target which bursts.

Initiated by	1st Address	2nd Address	3rd Address	4th Address
Any Master	0	8	10h	18h
CPU only	8	0	18h	10h
CPU only	10h	18h	0	8
CPU only	18h	10h	8	0
Any Master	10h	18h	-	-

Table 6.5. 64-bit Cache Line Fill Ordering.

6.3.4. 64 bit master accesses to 32 bit targets.

A 64 bit master begins a bus cycle like a normal 32 bit access except that LBS64# is driven with ADS# when the master attempts a 64 bit transfer. A 32 bit target will ignore LBS64# and respond as if the cycle were initiated by a 32 bit master. The 64 bit master will always drive at least one of the lower byte enables (BE<3..0>#) low, so a 32 bit target will never see a transfer request with no byte enables active.

A bridge from a 64 bit CPU must convert cycles from the CPU as follows:

CPU BE<3..0># all =1, some or all of CPU BE<7..4>#=0.
- Copy the CPU BE<7..4># to the VL Bus BE<3..0>#
- Copy the CPU D<63..32># to the VL bus DAT<31..0>
- VL bus ADR2=1
- Do not drive LBS64# low.

Some or all of CPU BE<3..0># =0, CPU BE<7..4># all =1
- Copy CPU BE<3..0># to VL bus BE<3..0>#
- Copy CPU D<31..0># to the VL Bus DAT<31..0>#
- VL bus ADR2=0
- Do not drive LBS64# low.

Some or all of CPU BE<3..0># =0, some or all of CPU BE<7..4># =0
- Copy CPU BE<3..0># to VL bus BE<3..0>#
- Copy CPU D<31..0># to the VL Bus DAT<31..0>#
- VL bus ADR2=0
- Drive LBS64# low. Monitor ACK64#.

The bridge may convert 64 bit requests into 32 bit bursts. The standard 32 bit burst rules must be adhered to, including a maximum burst length of 16 bytes. The bridge is responsible for translating the data.

If a CPU requests non-contiguous bytes, the bridge must convert them into multiple contiguous byte VL-Bus cycles.

7. DC Characteristics

Functional Operating Range : Vcc = 5V +- 5%; T = 0-60 Degree C.

7.1. Power Consumption

The power available on the VL-Bus V_{CC} lines is +5volts with a tolerance of 5 percent. Each VL-Bus device may draw a maximum of 10 watts (2 amps) of power from the slot. Additional power is available from the in-line system I/O bus slot and is not regulated by this specification. The VL-Bus device should draw power equally from all power and ground lines.

7.2. Signal Voltage Levels

No signal on the VL-Bus can have a steady-state voltage higher than the voltage applied to the V_{CC} power pins. The signal steady-state voltage must not be lower then the ground reference applied to the GND pins. Overshoot must be no more than 1.0 volts over V_{CC} for 5ns. Undershoot must be no more than 1.0 volts under GND for 5ns.

Description	Measured from	Value
VOL	Driver	-0.5 - 0.5 volts
VIL	Receiver	-0.5 - 0.8 volts
VOH	Driver	2.4 - 5.5 volts
VIH	Receiver	2.0 - 5.5 volts

Table 7.1. Signal Voltage Levels.

Support for 3.3 Volt Logic

V_{CC} pins on the VL-Bus are all +5Vdc. Devices needing other voltages should generate them using the +5Vdc supply. The VL-Bus supports 3.3-volt logic levels. Any 3.3-volt devices with inputs connected directly to the VL-Bus slot must have input structures capable of tolerating 5 volt logic levels as described above. Devices operating at 3.3 volts may output directly onto the VL-Bus slot.

7.3. Add-in Board Signal Loading and Routing

Trace length from the VL-Bus connector to the add-in board circuitry is limited to 4 inches. If the trace branches on the add-in board then the total length of all branches is limited to 2 inches. Traces not passing through the VL-Bus connector are not subject to this maximum trace length limitation.

Each add-in board may have a maximum of one F TTL load on each VL-Bus input signal. Traces not passing through the VL-Bus connector are not subject to maximum signal loading limitations.

7.4. Capacitive Loading Requirements

Each signal going across the VL-Bus connector is subject to capacitive load of up to 25 pf per slot. All VL-Bus add-in boards must be capable of driving the max. capacitive loading requirement according to the A.C. timing specification. Maximum slot number is not used as the calculation of VL-Bus capacitance loading.

If actual capacitance loading values cannot be obtained from manufacturer specification sheets use the values in Table 7.2 to estimate total capacitive loading of a VL-Bus add-in board.

Typical Capacitance	Value (pf)
Receiver	4 - 8
Driver	10 - 12
Transceiver	12 - 16
PC trace (per inch)	1.5 - 3
PC board via	0.5

Table 7.2. Typical Capacitance Values.

7.5. Signal Impedance

As a guideline, signal impedance should equal 50 Ohms or less on each trace. The signal impedance can be calculated with the following equation:

$$Z_{signal} = Z_{trace} / ((C_{trace} / C_{component})+1)^{1/2}$$

where

Z_{signal} = signal loaded trace impedance
Z_{trace} = the impedance of the board trace
C_{trace} = the capacitance of the board trace
$C_{component}$ = the load capacitance from components and connectors

7.6. Output Driver Sink Current Requirements

LBC Outputs

Signal	Unbuffered	Buffered	Units
LCLK	8	8	ma
Address and data	4	8	ma
BE<3..0>, M/IO#, W/R#, ADS#, RDYRTN#, D/C#, LEADS#, BLAST#	5	8	ma
LGNT#	4	4	ma
ID<4..0>	8	8	ma

LBM Outputs

Signal		Buffered	Units
Address and data		8	ma
BE<3..0>, M/IO#, W/R#, ADS#, RDYRTN#, D/C#, BLAST#		8	ma
LREQ#		4	ma
IRQ9		8	ma

LBT Outputs

Signal		Buffered	Units
Data bus		8	ma
LDEV#		4	ma
IRQ9		8	ma
LRDY#, BRDY#, LKEN#, LBS16#		8	ma

Table 7.3. Output Drive.

8. Physical Characteristics

8.1. VL-Bus Slot Location

The VL-Bus slot is defined as a 16-bit Micro Channel connector (Burndy part no. CEE2X56S3Z14 or equivalent) inline with the system I/O bus connector. The following diagrams illustrate the approximate location of the VL-Bus connectors for ISA, EISA, and Micro Channel systems. A Micro Channel pin 1 of the VL-Bus connector is closest to the system I/O bus connector. Pin 1 of the VL-Bus slot is 0.5 inch (pin center) from the last pin in the system I/O bus connector (pin center). The VL-Bus connectors may be inline with any of the system I/O bus slots. Specific slot(s) are not specified and are left to the system designer.

The board space under the 64-bit extension area should be kept clear to allow a future 64-bit VL-Bus board with additional fingers in this area to hang freely. The extension area length is 2 inches beyond the end of the 32-bit connector, width is 0.4", equal to the width of the 32-bit connector. Future 64-bit VL-Bus motherboards will use a 32-bit Micro Channel connector. In all VL-Bus systems at least one slot should be kept clear to allow room for a future 64-bit VL-Bus board to be installed.

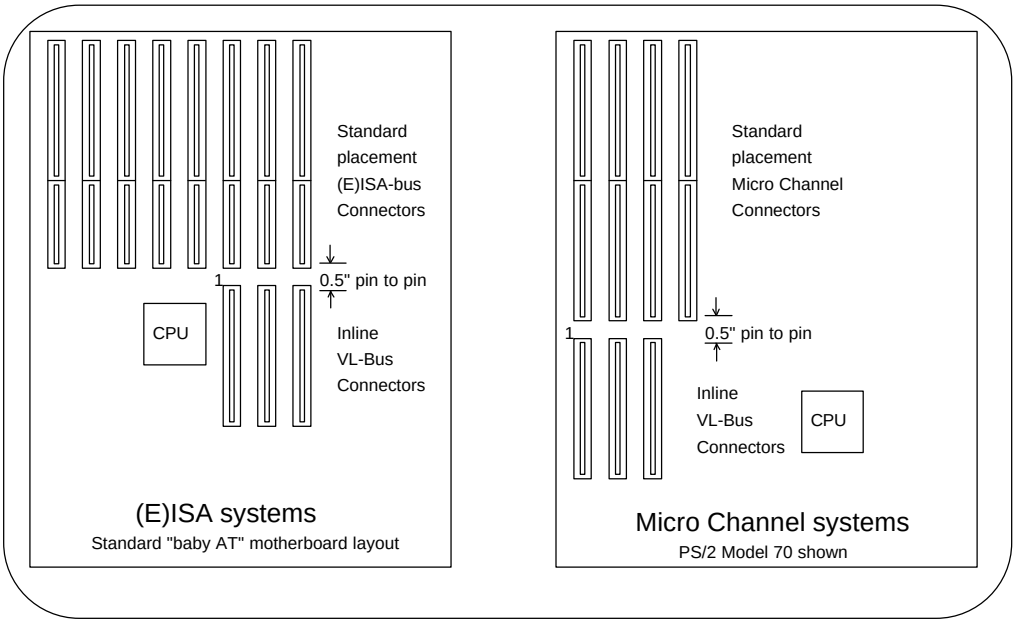


Figure 8.1. Sample Physical Layout Diagram.

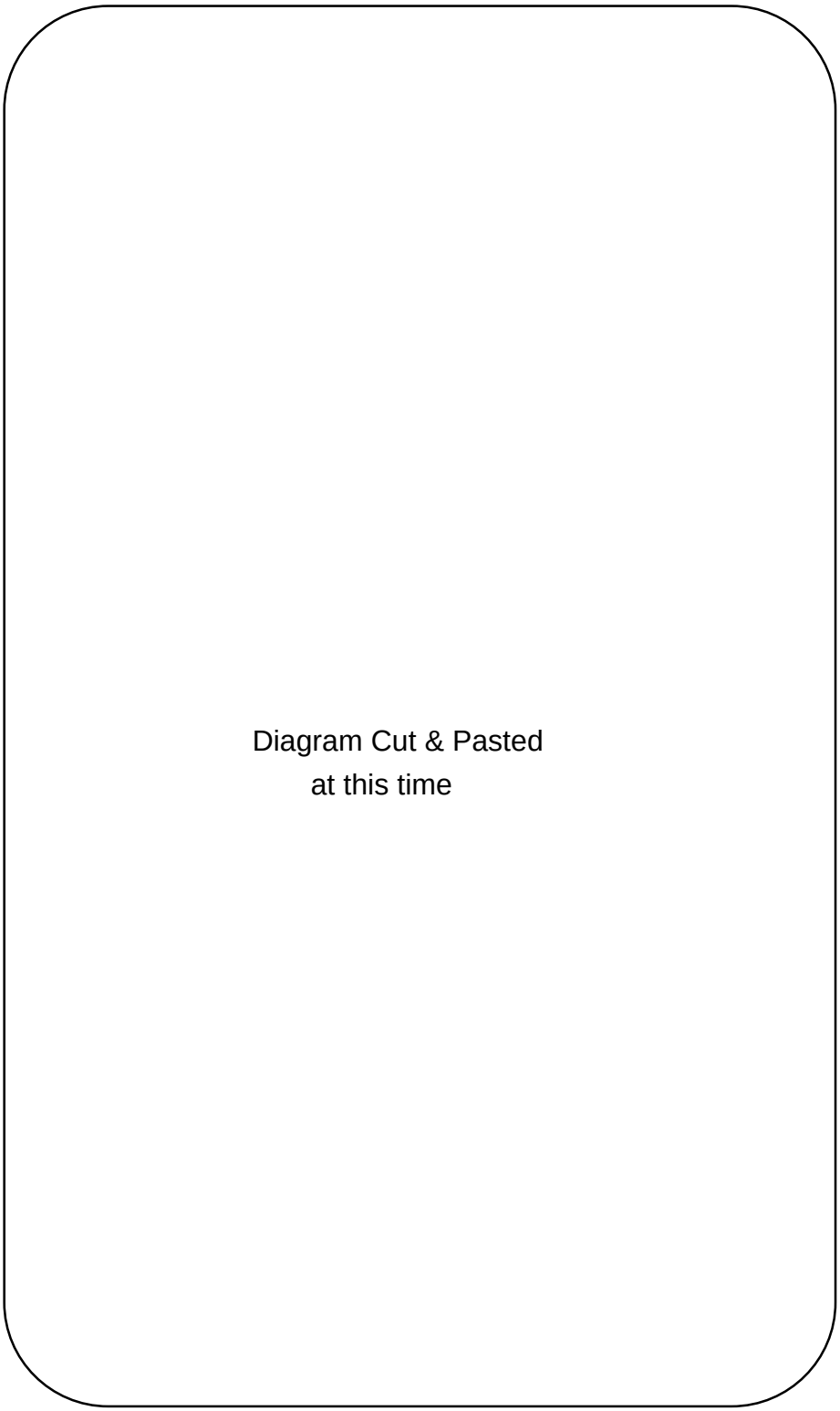


Figure 8.2. 32/64-Bit VL-Bus/ISA PC Board Hole Pattern

8.2. Motherboard Slot Dimensions

Anything on the motherboard inside of the 64-bit extension area should not rise above 0.3 inches below the top of the VL-Bus connector. The extension area length is 2 inches beyond the end of the 32-bit connector, width is 0.4", equal to the width of the 32-bit connector. This restriction allows a future VL-Bus board to fit into a an existing VL-Bus slot without physical obstructions caused by the additional fingers striking obstacles on the motherboard.

8.3. Add-in VL-Bus Board Specifications

8.3.1. ISA or EISA/VL-Bus Add-in Boards

Figure 8.3 represents a 32-bit VL-Bus/ISA board. Figure 8.4 represents a 32-bit VL-Bus/EISA board. The dimensions of both boards are based on the EISA adapter physical specification. Indicated length and height of the boards are maximums only. All specifications on these drawings are given in inches, unless otherwise indicated. B-size masters of these drawings are available in the VESA office. For detailed information on other aspects of the ISA or EISA system, see the EISA Hardware Specification.

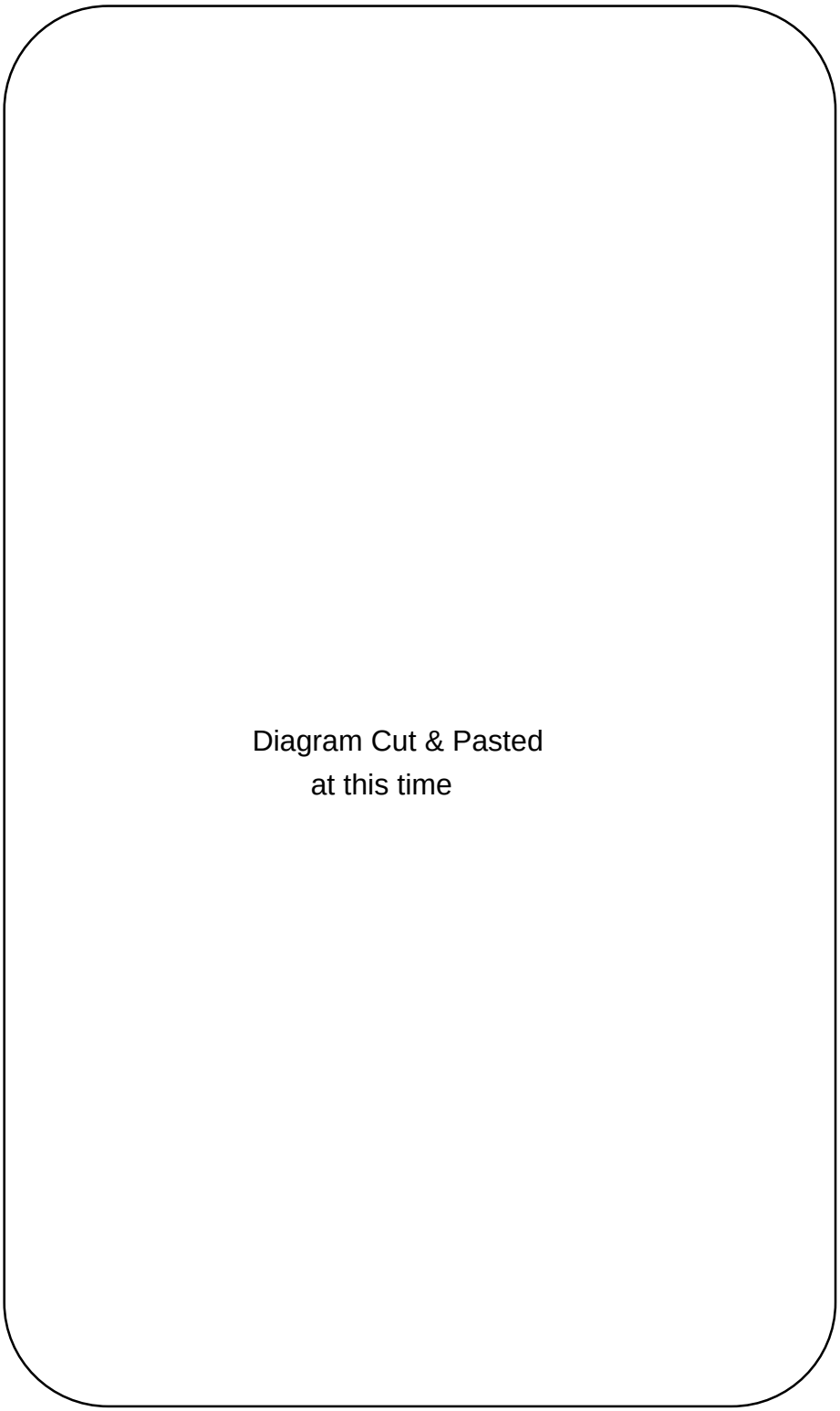


Figure 8.3. VL-Bus/ISA Adapter Card Physical Layout

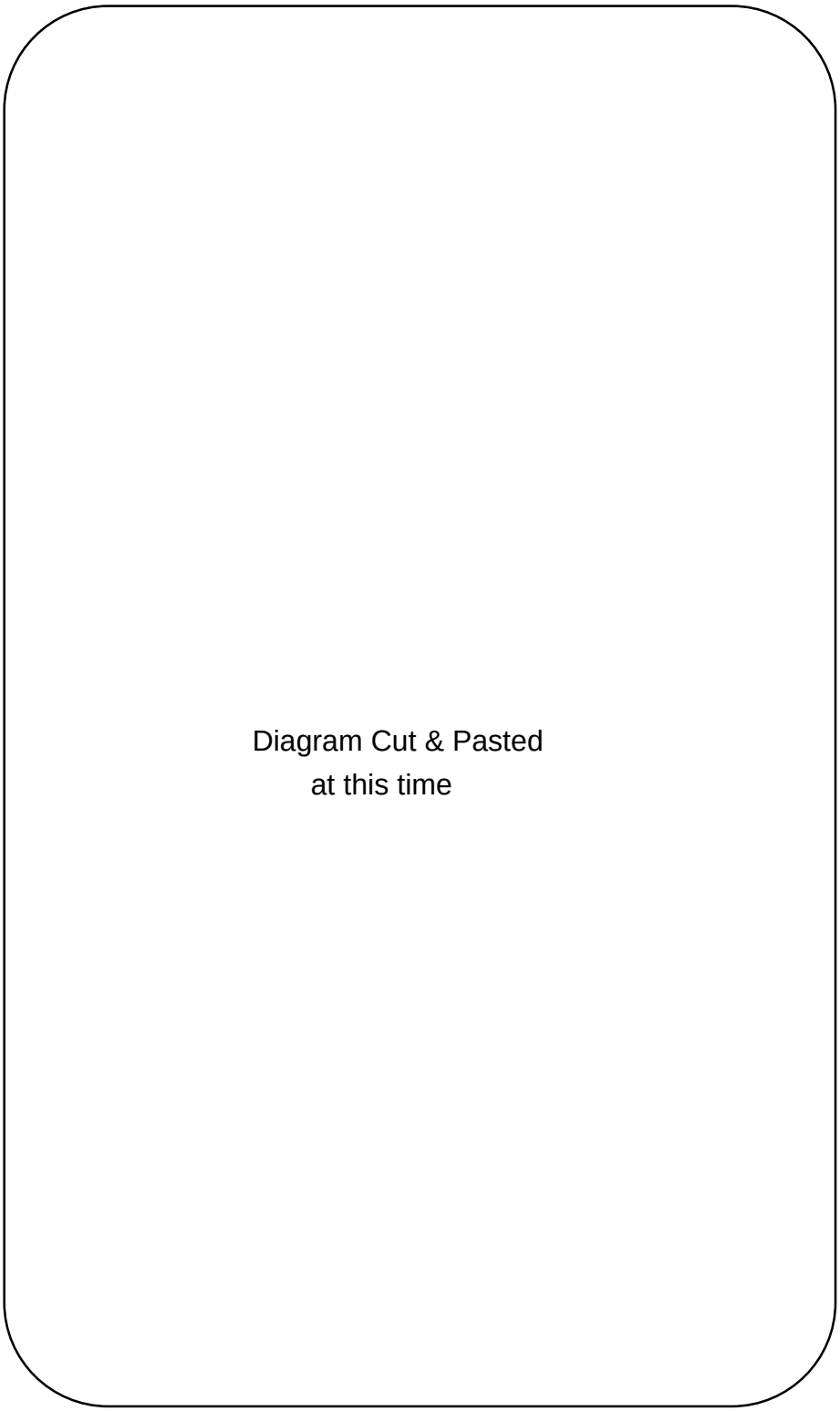


Figure 8.4. VL-Bus/EISA Adapter Card Physical Layout

8.3.2. Micro Channel/VL-Bus Add-in Boards

Figure 8.5 represents a 32-bit or 64-bit Micro Channel type VL-Bus board. The dimensions are based on the Type 3 adapter physical specification. Thickness of the board is 0.063 inch. This mechanical drawing should be used for the relative position of the VL-Bus connector only. For detailed information on the board mechanicals and other aspects of the Micro Channel system, contact the Micro Channel Developers Association (MCDA) at (800) GET-MCDA or (916) 222-2262. The fax number is (916) 222-2528.

Micro Channel adapters should use the software identification and setup proceedures provided on the Micro Channel bus. Technical assistance is available from the MCDA.

The VL-Bus connector occupies space allocated to the Micro Channel 64-bit extension. Adapter cards must adhere to the mechanical specification to ensure that a VL-Bus card cannot mate with a 64 bit extended Micro Channel slot.

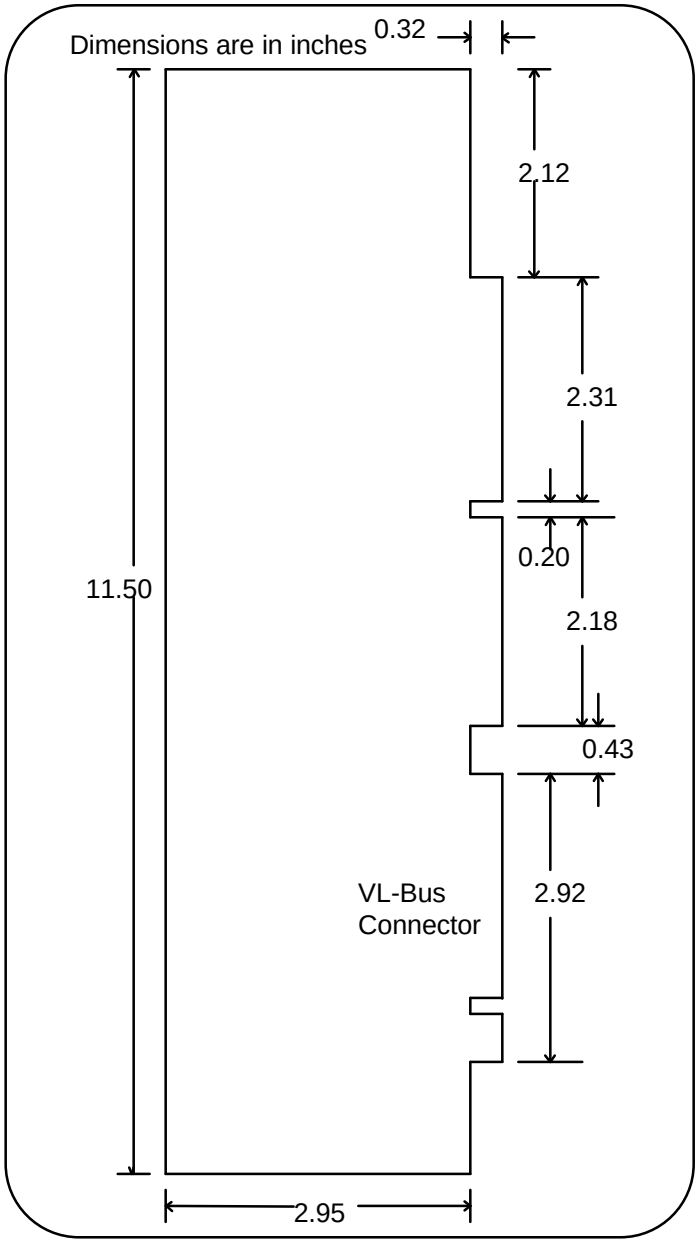


Figure 8.5. VL-Bus/Micro Channel Adapter Card Physical Layout

9. Required Support

This section describes the basics of what motherboards, targets, and masters must do to be VL-Bus Compatible.

9.1. Motherboard Required Support

The following is a basic list of features that a motherboard must adhere to in order to be VL-Bus compliant. It is not intended to be a comprehensive list, but a brief guideline that outlines the major points. The entire VL-Bus specification should be referenced and adhered to for designs.

9.1.1. CPU Access Support

The local bus controller must allow the CPU to access local bus memory and IO by sampling LDEV# after allowing the appropriate decode time from address M/IO # and W/R#. It must float LRDY#, BRDY#, and the data bus (on read cycles) by the end of the first T2 if LDEV# was sampled active. The local bus controller may ignore LDEV# for system board cache hits or local memory cycles.

9.1.2. DMA Access Support

The local bus controller must generate the local bus control signals during DMA or system IO Bus Master accesses to local bus targets. Specifically, the address must be driven onto the local bus and BE<3..0>#, ADS#, M/IO#, D/C#, W/R#, and BLAST# must be generated from the appropriate system IO bus signals. LDEV# must be sampled to determine if the local bus is the target. ADS# is generated if LDEV# is active (or optionally always) and wait states are added to the system IO bus until LRDY# is received. Data must be transferred between the local bus and the system IO bus according to the cycle type and lower address bits (byte enables). On read cycles, the system IO bus data hold time must be met by delaying RDYRTN# or by latching the data. If the motherboard wants to guarantee that LRDY# is returned by the target, BLAST# must be driven low through the entire cycle starting in T1.

9.1.3. 16 -Bit Target Support

The system board will support accesses to either 32-bit or 16-bit local bus targets. Accesses that attempt to cross a 16-bit boundary with one cycle will be split into two accesses when the target drives BS16# low. The system board is responsible for this on CPU, DMA, and System IO bus master accesses. Local Bus Masters are responsible for this when they initiate an access. In most ISA bus systems, this involves merely connecting BS16# from the VL-Bus to the CPU and providing a pull-up resistor.

9.1.4. Local Bus Master Support

The system board will support local bus masters by providing arbitration signals (LREQ# and LGNT#) to allow the local bus master to gain control of the bus. A separate set of LREQ# and LGNT# signals are provided for each VL-Bus slot. The system board will allow the Local Bus Master to access all system resources including main memory, system IO bus memory and IO, and VL-Bus memory and IO. Interrupt acknowledge and "special" cycles need not be supported. The address, byte enables, ADS#, M/IO#, D/C#, W/R#, BLAST#, and LEADS# must be floated by the system board when a VL-Bus master has control of the bus.

9.1.5. Bus Arbitration

An arbitration scheme must be provided which provides a predictable bus latency for local bus masters. The latency should not exceed 20uS the vast majority of the time, as far as the motherboard can control. Sufficiently low bus latency must be provided for DMA and System Bus Masters to allow the particular System Bus devices to function normally. The system board must allow VL-Bus master activity to proceed normally during the preemption period.

9.1.6. Cache Coherency

Cache coherency must be maintained at all times, for all master/target combinations. When a local bus master has control of the bus, LEADS# may be used to invalidate the CPU and/or secondary cache. The system board is responsible for driving KEN# at all times (no longer a VL-Bus signal), and must make VL-Bus targets non-cacheable by default. Caching of specific VL-Bus targets is allowed as an option.

9.1.7. System Level Function

The motherboard is responsible for generating or receiving the following signals for the system:

LCLK	A 1x system with proper skew control.
RESET#	An active low reset that occurs at power on and system reset.
RDYRTN#	Functionally the 486 RDY#. Generated from LRDY# and motherboard RDY# sources.
WBACK#	A "Backoff" type signal optionally used for cache coherency. Must be pulled high if not used.
ID0:4	Motherboard ID bits, indicating CPU type and motherboard capabilities.
IRQ9	Video interrupt; active high.

9.2. VL-Bus Target Device Required Support

The following is a basic list of features that a VL-Target must adhere to in order to be VL-Bus compliant. It is not intended to be a comprehensive list, but a brief guideline that outlines the major points. The entire VL-Bus specification should be referenced and adhered to for designs.

9.2.1. Generate LDEV#

The VL-Bus Target must GENERATE LDEV# BASED ON THE ADDRESS, M/IO# , AND in SOME cases W/R#, DRIVING IT LOW WHEN IT IS ADDRESSED. LDEV # must be stable (high or low) within 20nS of the address, M/IO# and W/R# (when used) being stable. LDEV# must not be latched with ADS# or LCLK.

9.2.2. Responds to All Data Sizes

The VL-Bus target must respond to 8, 16, 24, and 32 bit accesses (all legal combinations of the byte enables). IO targets may limit their allowable access types to those that are legal when it is on the system IO bus. IO targets should take into account 16 or 32 bit local bus accesses that would normally be divided into several cycles by the system IO bus controller.

9.2.3. Data Steering

The VL-Bus Targets should receive the data on writes in the bytes lanes specified by the 486 and not assume the 386 data copying. The VL-Bus target must drive all bytes of data regardless of the data size on all read cycles and provide the data on the byte lanes specified by the 486. If the VL-Bus target drives BS16# low, it should also copy the high word data onto the low word when BE1# and BE0# are inactive during read cycles.

9.2.4. Hold Data Until RDYRTN#

On read cycles which the target terminates the LRDY#, the target must provide valid data at the end of the T-state where LRDY# will be received by the system board, and continue to drive valid data until the end of the T-state where RDYRTN# is sampled low. This may be the same T-state or many T-states later.

9.2.5. Data Bus Driving

On read cycles, the target must not drive the data bus until the second T2 to prevent data contention with a cache on the system board.

9.2.6. LRDY# and BRDY# Driving

LRDY# and BRDY# must not be driven the first T2 to avoid a contention with the system board cache and/or memory controller. An exception to this is on write cycles on system boards that will accept zero wait states writes from VL-Bus targets (this is indicated by ID2). When LRDY# and BRDY# are driven low, they must be driven high for one half clock cycle before floating.

9.2.7. LRDY# and BRDY#

Since the motherboard DMA function may require LRDY# - RDYRTN# protocol to hold the data for the DMA initiator or ISA master, a VL-Bus target must terminate the cycle with LRDY# rather than BRDY# when a DMA transfer might be in progress. The target may us either of the following rules to determine whether it must terminate the cycle with LRDY#:

- Sample BLAST# at the end of the first T2 and drive LRDY# if it is low.
- The target may terminate with BRDY# if BLAST# was high at any time during the cycle, including T1.

The last cycle of a burst does not require an LRDY#.

9.2.8. Palette Snooping

A VL-Target must not drive LDEV# low or return a ready for IO write cycle to the 03C6-03C9 range. If the VGA palette is mapped onto the VL-Bus, the VGA should capture the data, but allow the cycle to proceed onto the system I/O bus. Then VGA should drive LDEV# for palette reads (if the palette is mapped onto the VL-Bus).

9.2.9. Cycle Tracking

VL-Targets which are tracking the VL-Bus state should be aware that RDYRTN# will not necessarily be returned between two ADS#, since another target may use BRDY# exclusively for many cycles. It should also be aware that in systems that use the WBACK# function, neither RDYRTN# or BRDY# may be returned for aborted cycles. This will occur mostly in systems in which the CPU contains a write back cache, but may also appear in other cases.

VL-Targets should also be aware that the ADR<31..2>, M/IO#, and W/R# will change to data during 64 bit transfers. Targets not involved in the transfer should treat this in the same manner as idle cycles are treated. LDEV# should continue to be driven asynchronously from these signals during this time (it will be ignored by the system board), but the target should not drive the data bus or otherwise respond until it sees the next ADS#.

9.2.10. Trace Length Limits

The trace length limits in the VL-Bus Spec. must be adhered to in order to maintain satisfactory operation in a fully loaded system.

9.2.11. Mechanical Specifications

The Mechanical specifications must be adhered to in order to allow the board to be used in all compatible systems. The card edge connector should be beveled on each side of the key pins (positions 46 and 47) to prevent breakage of the key on the motherboard.

9.3. VL-Bus Master Required Support

The following is a basic list of features that a VL-Master must adhere to in order to be VL-Bus compliant. VL-Masters are also targets and must adhere to the target list also. It is not intended to be a comprehensive list, but a brief guideline that outlines the major points. The entire VL-Bus specification should be referenced and adhered to for designs.

9.3.1. Monitor RDYRTN# and BRDY#

All VL-Bus Masters must receive RDYRTN# and BRDY#, even if they do not burst (this is required to maintain a consistent burst protocol on the bus). The VL-Master should handle RDYRTN# and BRDY# in the same way as a 486.

9.3.2. Bursting

A master which would like to burst should monitor ID4 and the CPU type (ID1 and ID0) to determine the bursting level which is supported on the motherboard. VL-Bus masters must drive BLAST# according to the 486 specification if they support bursting. If bursting is not supported, BLAST# must be driven low when the master owns the bus.

9.3.3. Drive LEADS#

VL-Bus Masters must drive LEADS# low with ADS# for all memory cycles which they initiate. LEADS# must not be driven for I/O cycles. The VL-Bus master should assume that the system board may finish the write cycle in 0 wait states.

9.3.4. Support WBACK#

VL-Bus masters must monitor the WBACK# signal, abort the bus cycle, and float the address, control, and data buses when WBACK# is low. The bus cycle should be restarted (with a new ADS#) when WBACK# goes back high.

9.3.5. Honor Preemption Protocol

VL-Bus masters must give up control of the bus within 5uS of being preempted by the Local Bus Controller.