Designer May Step Address, Data, PAR (and PAR64) and IDSEL

The address may be stepped onto the AD bus (including the 64-bit extension) because it is qualified by FRAME#. PAR (and PAR64) may also be stepped because they are guaranteed qualified one clock after the end of the address phase and each data phase. IDSEL can be stepped because it is qualified by the FRAME# signal (refer to the section entitled "Resistively-Coupled IDSEL Is Slow" in the chapter entitled "Configuration Transactions"). Data can be stepped onto the AD bus during each data phase because it is qualified by the assertion of IRDY# (on a write) or TRDY# (on a read).

Table 8-3 defines the relationship of the AD bus, PAR, PAR64, IDSEL and DEVSEL# and the conditions that qualify them as valid.

Table 8-3. Qualification Requirements

Signal(s)	Qualifier
AD bus during address phase	Qualified when FRAME# signal sampled asserted at the end of the address phase.
AD bus during data phase on read	Qualified when TRDY# signal sampled asserted on a rising clock edge during the data phase.
AD bus during data phase on write	Qualified when IRDY# signal sampled asserted on a rising clock edge during the data phase.
PAR and PAR64	Implicitly qualified on rising clock edge after address phase, or by IRDY# and TRDY# (data phase).
IDSEL	Qualified when FRAME# sampled asserted at the end of the address phase and a type zero configuration command is present on the C/BE bus (with AD[1:0] =00b).

Continuous and Discrete Stepping

The initiator (or the target) may use one of two methods to step a valid address or data onto the AD bus, or a valid level onto the PAR and PAR64 signal lines, or IDSEL:

- If the device driving the AD bus and the parity pins or IDSEL, either initiator or target, **uses very weak output drivers**, it may take several clocks for it to drive a valid level onto these bus signals (i.e., the propagation delay may be lengthy because it may take several reflections, with the resultant voltage-doubling effect, before the address (or data) is in the correct state on the bus). This is known as **continuous stepping**. See note in the next section.
- The device driving the AD bus and the parity pins or IDSEL, either initiator or target, may have **strong output drivers** and may drive a subset of them on each of several clock edges until all of them have been driven. This is known as **discrete stepping**.

Disadvantages of Stepping

There are two disadvantages associated with stepping:

- Due to the prolonged period it takes to set up the address or data on the bus, there is a performance penalty associated in any address or data phase where stepping is used.
- In the midst of stepping the address onto the bus, the arbiter may remove the grant from the stepping master. This subject is covered in the next section.

The specification strongly discourages the use of continuous stepping because it results in poor performance and also because it creates violations of the input setup time at all inputs.

Preemption While Stepping in Progress

When the PCI bus arbiter grants the bus to a bus master, the master then waits for bus idle before initiating its transaction. If, during this period of time, the arbiter detects a request from a higher priority master, it can remove the grant from the first master before it begins a transaction (i.e., before it asserts FRAME#).

Assuming that this doesn't occur, the master retains its grant and awaits bus idle. Upon detection of the bus idle state, the master begins to step the address onto the AD bus, but delays the assertion of FRAME# for several clocks until the address is fully driven. During this period of time, the arbiter may still remove the grant from the master. The arbiter hasn't detected FRAME# as-

serted and may therefore assume that the master hasn't yet started a transaction (even though the arbiter can see that the bus is idle). If the arbiter receives a request from a higher-priority master, it may remove the grant from the master that is currently engaged in stepping an address onto the AD bus. In response to the loss of grant, the stepping master must immediately tri-state its output drivers.

It is a rule that the arbiter cannot deassert one master's grant and assert grant to another master during the same clock cell if the bus is idle. The bus may not, in fact, be idle. A master may not have asserted FRAME# yet because it is in the act of stepping the address onto the AD bus.

If the arbiter were to simultaneously remove the stepping master's GNT# and issue GNT# to another master, the following problem would result. On the next rising-edge of the clock, the stepping master detects removal of its GNT# and begins to turn off its address drivers. At the same time, the other master detects its GNT# and bus idle (because the stepping master had not yet asserted FRAME#) and initiates a transaction. This results in a collision on the AD bus.

When the bus appears to be idle, the arbiter must remove the grant from one master, wait one clock cell, and then assert grant to the other master. This provides a one clock cell buffer zone for the stepping master to disconnect completely before the other master detects its grant plus bus idle and starts its transaction.

It is permissible for the arbiter to simultaneously remove one master's grant and assert another's during the same clock cell if the bus isn't idle (i.e., a transaction is in progress). There is no danger of a collision because the master that has just received the grant cannot start driving the bus until the current master idles the bus.

Broken Master

The arbiter may assume that a master is broken if the arbiter has issued GNT# to the master, the bus has been idle for 16 clocks, and the master has not asserted FRAME# to start its transaction. The arbiter is permitted to ignore all further requests from the broken master and may optionally report the failure to the operating system (in a device-specific fashion).

Stepping Example

Figure 8-5 provides an example of an initiator using stepping over a period of three clocks to drive the address onto the AD bus. The initiator can start the transaction on clock three (GNT# sampled asserted and bus idle: FRAME# and IRDY# sampled deasserted). It then begins to drive the address onto the AD bus and the command onto the C/BE bus. During the clock cell four, it continues to drive the address onto the AD bus. During the clock cell five, it finalizes the driving of the address and asserts FRAME#, indicating the presence of the address and command. When the targets sample FRAME# asserted on the rising-edge of clock six (the end of the address phase), they latch the address and command and begin the address decode. Since this is an example of a write transaction, the initiator begins to drive the data onto the AD bus at the start of the data phase (clock six). Once again, it uses stepping, asserting the write data over a period of two clocks. It withholds the assertion of IRDY# until the data has been fully driven.

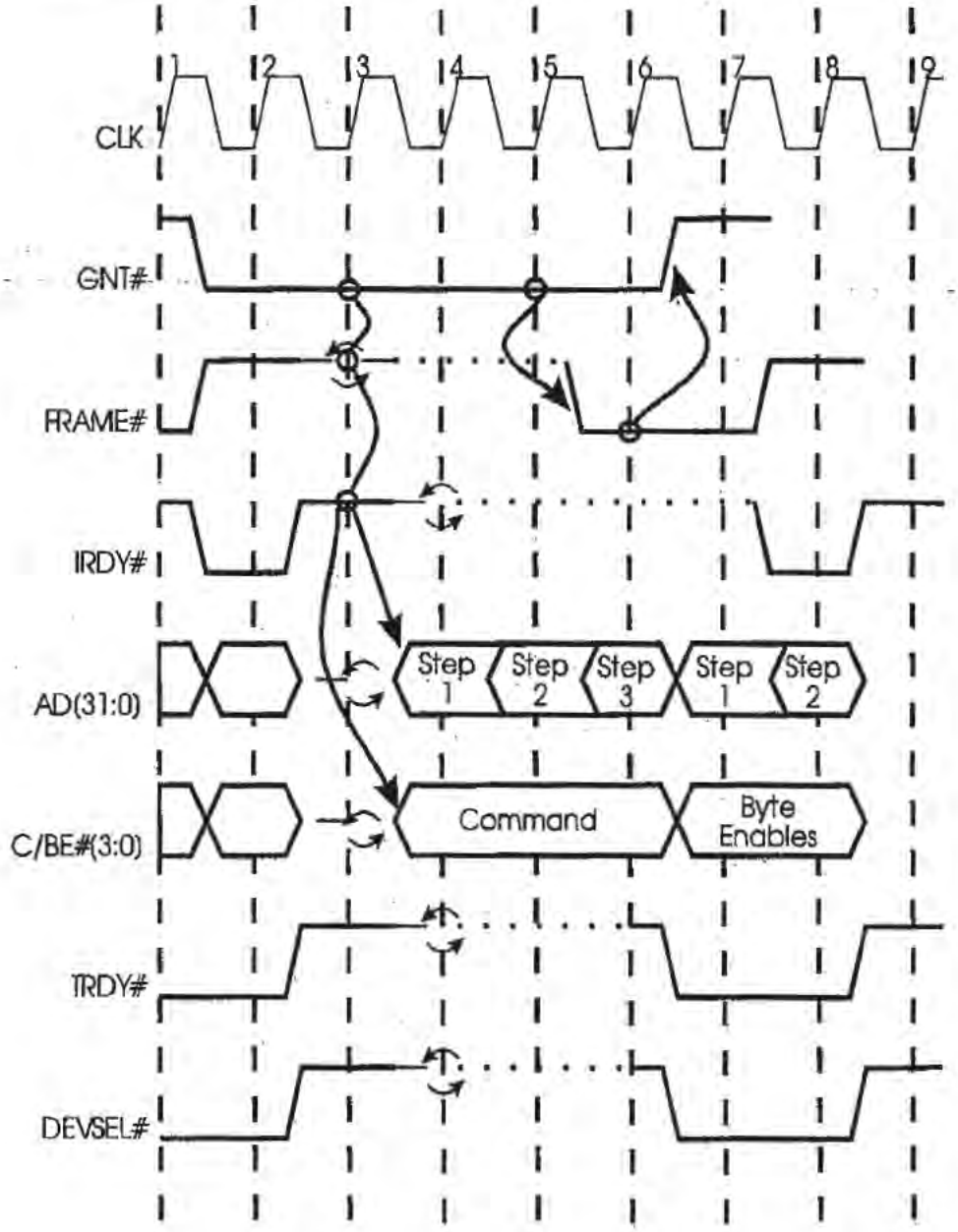

Figure 8-5. Example of Address Stepping

When Not to Use Stepping

Stepping must not be utilized when using 64-bit addressing because targets that respond to 64-bit addressing expect the upper 32 bits of the address to be presented one tick after FRAME# is sampled asserted.

Who Must Support Stepping?

All PCI devices must be able to handle address and data stepping performed by the other party in a transaction. The ability to use stepping, however, is optional.

Response to Illegal Behavior

Upon detection of illegal use of bus protocol, all PCI devices should be designed to gracefully return to the idle state (i.e., cease driving all bus signals) as quickly as possible. The specification is understandably vague on this point. It depends on the nature of the protocol violation as to whether the devices can gracefully return to their idle states and still function properly. As an example, the specification cites the case where the initiator simultaneously deasserts FRAME# and IRDY#. IN this case, when the target detects this illegal end to the transaction, it is suggested that the target deassert all target-related signals and return its state machine to the idle state. In the event that a protocol violation leaves a target device questioning its ability to function correctly in the future, it can respond to all future access attempts with a target abort. If the target thinks that the protocol violation has not impaired its ability to function correctly, it just surrenders all signals, returns to the idle state, and does not indicate any type of error.

Part VII

66MHz PCI Implementation

Chapter 22

Prior To This Chapter

The previous chapter provided a detailed description of issues related to caching from PCI memory targets. This subject was segregated in the latter part of the book because most PCI systems currently on the market do not support cacheable memory on the PCI bus. It injects considerable complexity into system and component design and the rewards may not be justified (due to the resultant degradation in performance).

In This Chapter

This chapter describes the implementation of a 66MHz bus and components.

The Next Chapter

The next chapter provides an overview of the VLSI Technology VL82C59x SuperCore PCI chipset.

Introduction

The revision 2.1 PCI specification defines support for the implementation of buses and components that operate at speeds of up to 66MHz. This chapter covers the issues related to this topic.

66MHz Uses 3.3V Signaling Environment

66MHz components only operate correctly in a 3.3V signaling environment. The 5V environment is not supported. This means that 66MHz add-in cards are keyed to install in 3.3V or universal card connectors and cannot be installed in 5V card connectors.

How Components Indicate 66MHz Support

The 66MHz PCI component or add-in card indicates its support in two fashions: electrically and programmatically.

A 66MHz PCI bus includes a newly-defined signal, M66EN. This signal must be bussed to the M66EN pin on all 66MHz-capable devices embedded on the system board and to a redefined pin (referred to as M66EN) on any 3.3V connectors that reside on the bus. The system board designer must supply a single pullup on this trace. The redefined pin on the 3.3V connector is B49 and is used as a ground pin by 33MHz PCI devices. Unless grounded by a PCI device, the natural state of the M66EN signal is asserted (due to the pullup). 66MHz embedded devices and cards either use M66EN as an input or don't use it at all (this is discussed later in this chapter).

The designer must include a 0.01μF capacitor located within .25" of the M66EN pin on each add-in connector in order to provide an AC return path and to decouple the M66EN signal to ground.

The PCI devices embedded on a 66MHz PCI bus are all 66MHz devices. A card installed in a connector on the bus may be either a 66MHz or a 33MHz card. If the card connector(s) aren't populated, M66EN stays asserted (by virtue of the pullup). If any 33MHz component is installed in a connector, the ground plane on the 33MHz card is connected to the M66EN signal, deasserting it.

How Clock Circuit Sets Its Frequency

The M66EN signal is provided as an input to the PCI clock circuit on the system board. If M66EN is sampled asserted by the clock circuit, it provides a 66MHz PCI clock to all PCI devices. If M66EN is sampled deasserted, the clock circuit supplies a 33MHz PCI clock. It should be fairly obvious that if any 33MHz components are installed on a 66MHz bus, the bus then operates at 33MHz.

Does Clock Have to be 66MHz?

As defined in revision 1.0 and 2.0 of the specification, the PCI bus does not have to be implemented at its top rated speed of 33MHz. Lower speeds are

acceptable. The same is true of the 66MHz PCI bus description found in revision 2.1 of the specification. All 66MHz-rated components are required to support operation from 0 through 66MHz. The system designer may choose, however, to implement a 50MHz PCI bus, a 60MHz PCI bus, etc.

Clock Signal Source and Routing

The specification recommends that the PCI clock be individually-sourced to each PCI component as a point-to-point signal from separate, low-skew clock drivers. This diminishes signal reflection effects and improves signal integrity. In addition, the system board and add-in card designer must adhere to the clock signal maximum trace length defined in revision 2.0 of the specification (2.5").

Stopping Clock and Changing Clock Frequency

As with the 33MHz PCI bus specification, the 66MHz specification states that the clock frequency may be changed at any time as long as the clock edges remain clean and the minimum high and low times are not violated. However, the clock frequency may not be changed except in conjunction with assertion of the PCI RST# signal. As an exception, components designed to be integrated onto the system board may be designed to operate at a fixed frequency (up to 66MHz) and may require that no clock frequency changes occur.

The clock may be stopped, but only in the low state (to conserve power).

How 66MHz Components Determine Bus Speed

When a 66MHz-capable device senses M66EN deasserted (at reset time), this automatically disables the device's ability to perform operations at speeds above 33MHz. If M66EN is sensed asserted, this indicates that no 33MHz devices are installed on the bus and the clock circuit is supplying a high-speed PCI clock.

A 66MHz device uses the M66EN signal in one of two fashions:

- The device is not connected to M66EN at all (because the device has no need to determine the bus speed in order to operate correctly).

- As described above, the device implements M66EN as an input (because the device requires knowledge of the bus speed in order to operate correctly).

System Board with Separate Buses

The system board designer can partition the board into two or more PCI buses. A 66MHz bus can be populated with devices that demand low-latency and high throughput. A separate 33MHz PCI bus is populated only with 33MHz devices.

Maximum Achievable Throughput

The theoretical maximum achievable throughput on a 66MHz PCI bus would be:

- 4 bytes per data phase * 66 million data phases per second = 264MB/second. This would be a 32-bit bus master bursting with a 32-bit target.
- 8 bytes per data phase * 66 million data phases per second = 528MB/second. This would be a 64-bit bus master bursting with a 64-bit target.

Electrical Characteristics

To ensure compatibility when operating in a 33MHz PCI bus environment, 66MHz PCI drivers must meet the same DC characteristics and AC drive points as 33MHz bus drivers. However, 66MHz PCI bus operation requires faster timing parameters and redefined measurement conditions. Because of this, a 66MHz PCI bus may require less loading and shorter trace lengths than the 33MHz PCI bus environment.

Figure 22-1 illustrates the differences in timing between 33 and 66MHz component operation. The chapter entitled "Intro To Reflected-Wave Switching" provides detailed information regarding 33MHz bus timing and the various timing components (e.g., Tval, Tprop, etc.).

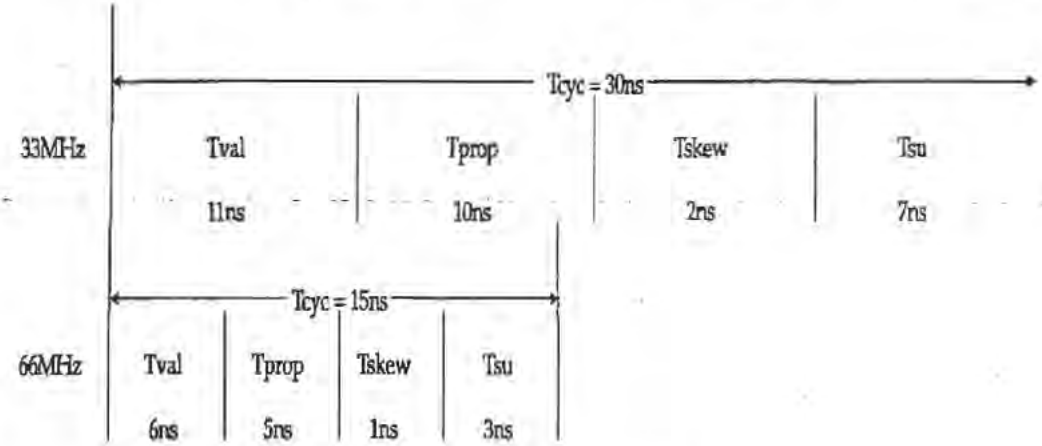

Figure 22-1. 33 versus 66MHz Timing

33MHz drivers are specified by their V/I curves, while 66MHz drivers are specified in terms of their AC and DC drive points, timing parameters, and slew rate. The specification defines the following parameters:

- The minimum AC drive point defines an acceptable first step voltage and must be reached within the maximum Tval time.
- The maximum AC drive point limits the amount of overshoot and undershoot in the system.
- The DC drive point specifies steady-state conditions.
- The minimum slew rate and the timing parameters guarantee 66MHz operation.
- The maximum slew rate minimizes system noise.

66MHz PCI designers must design drivers that launch sufficient energy into a 25Ω transmission line so that correct input levels are guaranteed after the first reflection.

At 66MHz, the clock cycle time is 15ns (vs. 30ns at 33MHz), while the minimum clock high and low times are 6ns each (vs. 11ns at 33MHz). The clock slew rate has a minimum specification of 1.5 and a maximum of 4 volts/ns (same as 33MHz specification). Table 22-1 defines the 66MHz timing parameters and provides a side-by-side comparison with the 33MHz timing parameters. The following exception applies to the 66MHz values in the table: REQ# and GNT# are point-to-point signals and have different setup times than do bussed signals. They have a setup time of 5ns.

Table 22-1. 66MHz Timing Parameters

		66MHz		33MHz	
Symbol	Description	Min	Max	Min	Max
Tval	CLK to signal valid delay, bussed signals	2ns	6ns	2ns	11ns
Tval (ptp)	CLK to signal valid delay, point-to-point signals	2ns	6ns	2ns	12ns
Ton	Float to active delay	2ns		2ns	
Toff	Active to float delay		14ns		28ns
Tsu	Input setup time to CLK, bussed signals	3ns		7ns	
Tsu (ptp)	Input setup time to CLK, point-to-point signals				
Th	Input hold time from CLK	0ns		0ns	
Trst	Reset active time after power stable	1ms		1ms	
Trst-clk	Reset active time after CLK stable	100µs		100µs	
Trst-off	Reset active to output float delay		40ns		40ns
Trrsu	REQ64# to RST# setup time	10Tcyc		10Tcyc	
Trrh	RST# to REQ64# hold time	0ns	50ns	0ns	50ns

When computing the 66MHz bus loading model, a maximum pin capacitance of 10pF must be assumed for add-in boards, whereas the actual pin capacitance may be used for devices embedded on the system board.

Addition to Configuration Status Register

A 66MHz-capable device adds one additional bit to its configuration status register. Bit 5 is defined as the 66MHZ-CAPABLE bit. A 66MHz-capable device hardwires this bit to one. For all 33MHz devices, this bit is reserved and is hardwired to zero. Software can determine the speed capability of a PCI bus by checking the state of this bit in the status register of the bridge to the bus in question (host/PCI or PCI-to-PCI bridge). Software can also check this bit in the status register of each additional device discovered on the bus in question to determine if all of the devices on the bus are 66MHz-capable. If even just one device returns a zero from this bit, the bus runs at 33MHz, not 66MHz. Table 22-2 defines the combinations of bus and device capability that may be detected.

Table 22-2. Combinations of 66MHz-Capable Bit Settings

Bridge's 66MHz-Capable Bit	Device's 66MHz-Capable Bit	Description
0	0	33MHz device located on 33MHz bus. Bus and all devices operate at 33MHz.
0	1	66MHz-capable device located on 33MHz bus. Bus and all devices operate at 33MHz. If the device is an add-in device and is only capable of proper operation when installed on a 66MHz bus, the configuration software may decide to prompt the user to install the card in an add-in connector on a different bus.
1	0	33MHz device located on 66MHz-capable bus. Bus and all devices operate at 33MHz.
1	1	66MHz-capable device located on 66MHz-capable bus. If status check of all other devices on the bus indicates that all of the devices are 66MHz capable, the bus and all devices operate at 66MHz.

Latency Rule

Devices residing on the 66MHz PCI bus are typically low-latency devices. The revision 2.1 specification requires that, on a read transaction, the time from assertion of FRAME# to the completion of the first data phase not exceed 16 PCI clocks. If it will, the target device must issue retry to the master. For multimedia applications, the majority of accesses are writes, not reads. Typically, a target device can accept write data faster than it may be able to supply read data. On a read, the device may need to access a slow medium. The device cannot be permitted to tie up the bus while fetching the requested data.

66MHz Component Recommended Pinout

The revision 2.0 specification suggested a recommended PCI component pinout wherein the signals wrapped around the component in the same order as the pin sequence on the add-in connector. The revision 2.1 specification states that "the designer may modify the suggested pinout...as required" to meet the 66MHz electrical specification.

Adding More Loads and/or Lengthening Bus

Running the PCI bus at 66MHz imposes tighter constraints on trace length and the number of loads the bus supports. The system board designer may choose to run the bus at a lower speed (e.g., 50MHz), thereby permitting longer traces and/or additional loads.

Number of Add-In Connectors

As a general rule, there is only one add-in connector on a 66MHz bus, but the specification does not preclude the inclusion of additional connectors (as long as the electrical integrity of the bus is maintained).

Part VIII

Overview of VLSI Technology VL82C59x SuperCore PCI Chipset

Chapter 23

Prior To This Chapter

The previous chapter described the implementation of a 66MHz bus and components.

In This Chapter

The PCI specification supports many permutations of system and therefore chipset design. This chapter provides an overview of the VL82C59x SuperCore PCI chip set from VLSI Technology. This overview is provided to present an example of PCI chipset implementation. It is not intended to provide a detailed description of the chipset operation. The VLSI component specification should be consulted for that purpose. In addition, it is assumed that the reader already has an understanding of the ISA bus. For detailed information on the ISA bus operation and environment, refer to the Addison-Wesley publication entitled *ISA System Architecture*, also authored by MindShare. The author would like to thank VLSI Technology for providing access to the chipset specification.

Chipset Features

The VLSI VL82C59x chipset provides the core logic necessary to design a Pentium-based system that incorporates both the PCI and ISA buses. It supports all 5V and 3.3V Pentium processors with host bus speeds of up to 66MHz. This includes the P5, P54C, P54CM and P54CT. It also supports dual-P54C processors. The chipset design includes the following features:

- Bridges the host and PCI buses.
- Bridges the PCI and ISA buses.
- Integrated L2 lookaside, direct mapped, write-through cache.
- Integrated system DRAM controller.
- Integrated PCI bus arbiter.

- Provision of posted-memory write buffers in both bridges.
- Supports Pentium processor's pipelined bus cycles.
- Self-configuring system DRAM banks.
- Shadow RAM support
- SMM support.
- Decoupled DRAM refresh.
- Supports synchronized or asynchronous processor and PCI clocks.
- Supports optional posting of I/O writes.
- Optional support for memory prefetching.
- PCI master reads from system DRAM memory can be serviced from processor's L2 cache or system memory.
- PCI master writes to system DRAM memory are absorbed by the bridge's posted-write buffer.
- Supports multiple-data phase PCI burst transactions.

Intro to Chipset Members

Refer to figure 23-1. The VLSI VL82C59x SuperCore PCI chipset consists of the following entities:

- VL82C591 Pentium System Controller. In conjunction with two VL82C592 Data Buffers, the system controller comprises the bridge between the host processor's local bus and the PCI bus.
- VL82C592 Pentium Processor Data Buffer. Taken together, two data buffers provide a triple-ported data bus bridge between the host data bus, system DRAM data bus and the PCI data bus (AD bus).
- VL82C593 PCI/ISA Bridge. The '593 provides the bridge between the ISA and PCI buses. In addition, the '593 incorporates much of the ISA system support logic.

The sections that follow provide additional information about the capabilities of the chipset.

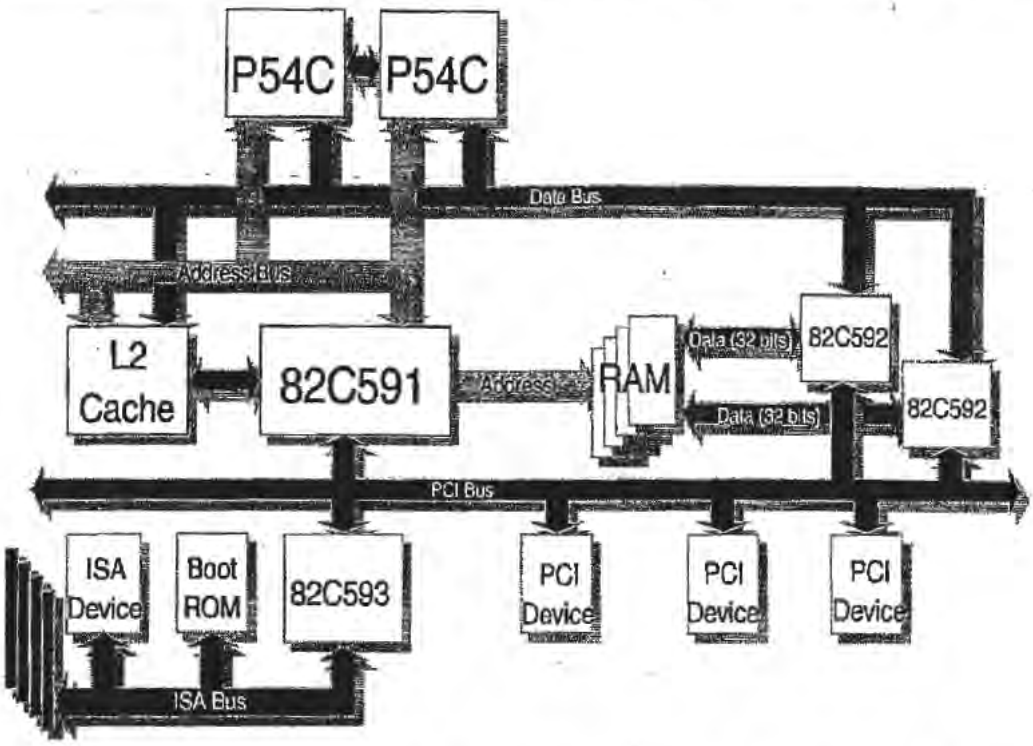

Figure 23-1. System Design Using VLSI VL82C59x SuperCore Chipset

VL82C592 Pentium Processor Data Buffer

As illustrated in figure 23-1, the host/PCI bridging function consists of the VL82C591 Pentium system controller and two VL82C592 data buffers. The two data buffer chips are controlled by the '591. They provide the following basic capabilities:

- On host processor reads from system memory, the '591 reads the requested data from DRAM and instructs the '592 data buffers to pass it to the host data bus.
- On host processor writes to system memory, the '591 instructs the data buffers to accept the write data into the posted-write buffer. This permits the host processor to conclude the memory write quickly. The posted-write buffer then offloads the write data to DRAM memory.

- On PCI-initiated memory reads from system DRAM memory, the '591 reads the requested data from memory and instructs the '592 data buffers to pass it to the requester on the PCI data bus.
- On PCI-initiated memory writes to system DRAM memory, the '591 addresses memory and instructs the '592 data buffers to accept the data presented on the PCI data bus and route it into system DRAM.

A discussion of the data buffer posted write capability can be found later in this chapter.

The following section discusses the functionality of the host/PCI bridge.

'591/'592 Host/PCI Bridge

General

As stated earlier, the host/PCI bridge functionality is provided by the '591 in combination with two '592 data buffers. The bridge performs the following basic functions:

- Services system DRAM memory reads and writes initiated by the host processor.
- Permits host processor(s) L1 cache(s) to snoop system memory accesses initiated by PCI and ISA masters.
- Translates host processor-initiated memory and I/O accesses into PCI memory and I/O accesses.
- Services system memory accesses initiated by PCI and ISA masters.
- Translates specific host processor-initiated I/O operations into PCI configuration read or write operations.
- Translates specific host processor-initiated I/O operations into PCI special cycle transactions.
- Incorporates the PCI and host bus arbiters.
- Translates host processor-initiated interrupt acknowledge bus cycles into PCI interrupt acknowledge transaction.

System DRAM Controller

The controller for system DRAM memory resides within the '591. Each memory bank (up to four) is either 64-bits (without parity) or 72-bits wide (with

parity). Each bank may be up to 256MB in size, yielding a maximum possible memory population of 1GB. In addition, each bank may be populated with 32 or 36-bits memory modules, permitting less-costly memory upgrade. The DRAM configuration registers permit the DRAM controller to work with DRAMs of various speeds and different geometries.

The controller supports two-way interleaved, page-mode memory. One or two pages (one in each bank) can be kept open at a time. For page-mode DRAMs that have a page open timeout of less than 15µs, the controller automatically closes a page that has been open for a period of 10µs. When using DRAMs with a maximum page open timeout in excess of 15µs, the 10µs automatic page close feature may be disabled and the refresh cycles can take care of ensuring that a page does not remain open for an excessive period. Non-page mode DRAM is not supported.

Refresh cycles may be set to occur every 15.625µs, 62.5µs, 125µs or 250µs. DRAM refresh cycles are transparent to the processor. If the processor initiates a DRAM access request simultaneously with a refresh cycle, the processor is stalled (i.e., wait states are inserted in its bus cycle) until the refresh cycle completes.

When a system DRAM parity error is detected, it is reported by the assertion of the PCI SERR# signal (assuming that the SERR# enabled and parity error response bits are set in the bridge's configuration command register). SERR# is typically connected to the '593 which asserts NMI to the host processor when SERR# is asserted. An option permits bad parity to be deliberately written to system DRAM to facilitate test and diagnostics.

Host processor-initiated memory accesses that target locations above the top of installed system DRAM are passed to the PCI bus and are not cached in the L1 and L2 caches. In addition, memory address ranges defined by the bridge's segment attribute and programmed memory region registers are also passed to the PCI bus and are not cached from.

The chipset does not permit the L1 and L2 caches to cache information from memory beyond the host/PCI bridge (i.e., PCI and ISA memory). This being the case, the '591 does not implement the snoop result outputs (SDONE and SBO#).

L2 Cache

The L2 cache controller is embedded within the '591 system controller. It is a direct-mapped, lookaside, buffered write-through cache. The L2 cache only caches information from system DRAM memory, never from PCI or ISA memory. The DRAM controller may be programmed to recognize sub-ranges within the overall memory address range assigned to system DRAM as PCI memory. When the processor initiates a memory transaction targeting an address in any of these programmed sub-ranges, the transaction is passed to the PCI bus and the data is not cached in L1 or L2.

The recommended L2 cache sizes are 256KB, 512KB and 1MB, but the L2 cache may be implemented as any desired size. The limitation is the amount of tag SRAM supplied by the system designer. The tag SRAM (i.e., the cache directory) is external to the '591 and can be of any size. Optionally, the L2 cache may be parity-protected.

The cache controller supports L2 cache line sizes of both 32 and 64 bytes. Additional SRAM is necessary to support the larger line size. When the 64 byte line size is implemented, a processor-initiated read miss in L2 results in the requested 32 byte line being read from DRAM. The line is sent back to the processor and a copy is also stored in the L2 cache. To the L2 cache, this is considered to be half of a line. The cache reads the next 32 bytes from DRAM and establishes it in the L2 as the second half of the 64 byte line.

A write-through cache usually extends the duration of a host processor-initiated memory write until the data has been written through to system memory. In this chipset design, the '591 instructs the '592 data buffers to accept the write data and permits the processor to complete its memory write immediately.

The cache controller supports both asynchronous and synchronous SRAMs. When using asynchronous SRAMs, burst read timing of 3-2-2-2 (three processor bus clocks to transfer the first quadword, and two clocks each for the transfer of each of the other three quadwords in the line) is achievable (at a bus speed of 66MHz). Burst write timing of 4-2-2-2 or 3-2-2-2 is achievable (depending on tag SRAM speed and signal loading). When using synchronous SRAMs, burst reads and write timing of 3-1-1-1 or 2-1-1-1 is achievable (depending on tag SRAM speed and SRAM type). When the processor performs back-to-back burst reads, pipelining reduces access time to 1-1-1-1.

Startup software can determine the following information related to the L2 cache:

- Cache SRAM type (asynchronous, synchronous type one or synchronous type two).
- Cache size.
- Line size.
- Cacheable memory range.
- Wait states imposed by cache SRAM type/speed.

Posted-Write Buffer

General

The posted-write buffer absorbs processor-initiated writes and permits the processor to end the write transaction quickly. The buffer logic then initiates the write to memory (or to PCI). While the buffer is engaged in the write, the processor can start and complete another memory write (assuming the buffer isn't full, it is absorbed by the posted-write buffer as well), a read hit on the L2 cache, or a write to the PCI bus (if the previously posted write was to system memory). The posted-write buffer that absorbs processor writes destined for system memory is eight quadwords deep (a quadword is 64-bits).

The posted-write buffer that absorbs memory writes destined for the PCI (or ISA) bus is one quadword deep. The bridge can only post writes to PCI/ISA memory within regions of memory programmed with the prefetchable attribute (in a '591 device-specific register). Optionally, the '591 can also be programmed to post PCI I/O writes initiated by the host processor. Any time the processor initiates a write to PCI/ISA memory in an area programmed to permit posting, the write is absorbed into the 64-bit PCI posted-write buffer. If the processor should initiate a subsequent memory write within the same quadword, the second write is merged into the bytes already in the buffer. This can result in non-contiguous byte enables asserted during the resulting PCI memory transaction, but this feature can be disabled. When disabled, the '591 uses a byte-reduction algorithm to generate two separate PCI memory writes utilizing only contiguous byte enables. Whenever the '591 has a PCI/ISA memory write posted in the buffer, it arbitrates for PCI bus ownership. When the bus has been acquired, it performs the memory write on the PCI bus. If the processor should initiate another PCI memory write prior to the conclusion of the one already in progress on the PCI bus, the processor is

stalled until the write buffer becomes available at the completion of the current PCI memory write transaction. In addition, bus ownership requests from other PCI bus masters are ignored until the conclusion of the current transaction.

The '591 does not permit a processor-initiated PCI read transaction to be performed on the PCI bus if a processor write to PCI memory is currently-posted in the buffer. The buffer is first flushed to PCI memory before the read is performed on the PCI bus.

The processor initiates burst write operations during the castout of a modified line or a snoop push-back (write-back) operation). The posted-write buffer (located in the data buffers) can accept the burst data at full bus speed (0 wait states).

The write buffer permits posting of memory writes to PCI memory within regions of memory space defined as prefetchable by bridge configuration registers.

A status bit can be checked by software to determine if the write buffer is empty.

Combining Writes Feature

The write buffer supports combining of writes. Assume that the processor performs a memory write to write two bytes into memory locations 00000100h and 00000101h. The processor outputs the following information:

- The quadword-aligned address placed on the host processor address bus is 00000100h.
- Byte enables [1:0] are asserted to indicate that the first two locations in the currently-addressed quadword are being addressed. Byte enables [7:2] are deasserted, indicating that the third through the eight locations in the quadword are not being addressed.
- The two bytes of data destined for memory locations 00000100h and 00000101h are driven onto data paths zero (D[7:0]) and one (D[15:8]).

The posted-write buffer latches the quadword address and the two bytes into the next available quadword location in its FIFO buffer. BRDY# is assert to the processor, permitting it to end the memory write transaction. Now assume that the processor initiates another memory write, this time to memory loca-

tion 00000104h (before the buffer logic has written the previous two bytes into system DRAM memory). Assume that the processor outputs the following information:

- The quadword-aligned address placed on the host processor address bus is 00000100h.
- Byte enable [4] is asserted to indicate that the fifth location in the currently-addressed quadword is being addressed. Byte enables [7:5] and [3:0] are deasserted, indicating that the first through fourth and the sixth through the eight locations in the quadword are not being addressed.
- The byte of data destined for memory location 00000104h is driven onto data path four (D[39:32]).

The buffer recognizes that some portion of quadword 00000100h has already been posted to be written to memory. Instead of using up another quadword-wide buffer location for the new write, it combines the new data being supplied by the processor with the older data in the buffer location. The buffer location now contains three bytes to be written to quadword 00000100h in system DRAM. Although the processor performed two separate memory writes to system memory, the buffer logic only has to perform one write operation when it offloads the data to memory.

Read-Around and Merge Features

If the processor initiates a read from system DRAM while one or more memory write operations reside within the posted-write buffer, the buffer logic performs the read from DRAM before flushing the writes to memory. If the read hits on a posted-write in the buffer, the bytes posted to be written to memory are merged with the data read from memory and the resulting data is supplied back to the processor.

Write Buffer Prioritization

The '591 can be programmed to adjust the priority of posted-write buffer writes to memory relative to memory reads. The following settings are available:

- The write buffer can access memory whenever the DRAM is idle.
- The write buffer can access memory after a minimum of 2, 4, 8, 16, 32 or 64 CPU clocks from the completion of the last DRAM read. The count is restarted at the completion of each read. When any of these settings are selected, the write buffer is permitted to access memory when the proces-

sor generates an access that is not a read (e.g., another write or a PCI transaction).

- Write buffer access to system memory is permitted only when the processor generates a non-system memory read transaction.

Configuration Mechanism

The '59x chipset implements PCI configuration mechanism number one (configuration address port at I/O location 0CF8h and configuration data port at I/O location 0CFCh).

PCI Arbitration

The '591 incorporates the PCI bus arbiter. The arbiter supports the '591, the '593 and up to four additional PCI bus masters. The '591's REQ# and GNT# signals are internally connected to the arbiter. A single signal line is used by the '593 to request and be granted ownership of the PCI bus (refer to the section in this chapter entitled "'593 Characteristics When PCI Master." Optionally, the '593 may use one of the four REQ#/GNT# signal pairs for arbitration.

The '591 never generates fast back-to-back transactions because it doesn't know the address boundaries of different targets.

The priority scheme may be software selected as fixed or rotational. When fixed is selected, the '593's REQGNT# signal has highest priority. This guarantees DMA channels timely access to the bus. Then, in descending order of importance, the priorities of the other masters are master 3, master 2, master 1, master 0 and the processor. The processor has lowest priority. Whenever any of the PCI masters require access to the PCI bus, the '591 asserts HOLD to the processor and takes the bus away from it to grant to the most important PCI master.

When rotational priority is selected, the '593 has highest priority, with priority rotating between bus masters 0 through 3 and the host processor. Refer to figure 23-2.

The arbiter can be programmed to park the bus either on the '591 or on the last master that used the bus. The latter mode can only be selected when rotational priority has been selected.

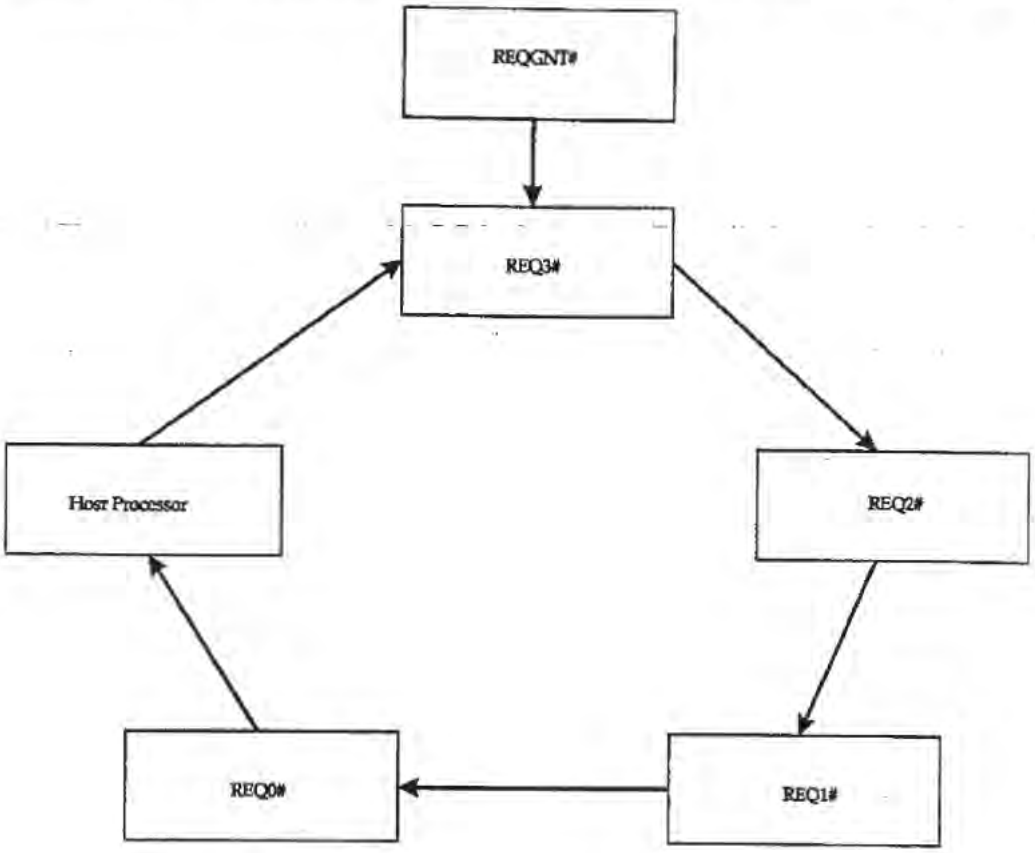

Figure 23-2. Rotational Priority Scheme

Locking

The arbiter in the '591 implements a bus lock. It does not support target locking.

Special Cycle Generation

Software can stimulate the host/PCI bridge to generate a PCI special cycle to PCI bus zero or to any of its subordinate PCI buses using the method defined for configuration mechanism number one; that is, the programmer performs a 32-bit write to the configuration address port specifying the target PCI bus, and sets the target device number, function number and doubleword number

to 1Fh, 7h and 00h, respectively. The programmer then performs a two byte or four byte write to the configuration data port. The bridge performs a special cycle on the target PCI bus, supplying the data written to the configuration data port as the message during the data phase. If the target bus is a subordinate bus, the '591 generates a special cycle request using a type 1 configuration write transaction.

'591 Configuration Registers

Figure 23-3 illustrates the '591's PCI configuration registers. The sections that follow define the manner in which the chipset implements each of these registers. For a description of the '591's device-specific configuration registers, refer to the chipset specification.

Vendor ID Register

The vendor ID for VLSI Technology is 1004h.

Device ID Register

The device ID for the '591 is 0005h.

Command Register

Table 23-1 defines the '591's usage of its command register bits.

Table 23-1. '591 Command Register Bit Assignment

Bit	Description
0	**I/O enable bit.** Hardwired to zero because the '591 doesn't respond to any PCI I/O transactions.
1	**Memory enable bit.** When set to one, PCI bus masters can access system DRAM memory. Reset sets this bit to one.
2	**Master enable bit.** Hardwired to one because the '591 is always enabled to initiate PCI transactions.
3	**Special cycle monitor enable bit.** Hardwired to zero because the '591 does not monitor special cycles generated by other PCI masters.
4	**Memory write and invalidate enable bit.** Hardwired to zero because the '591 never generates the memory write and invalidate command.
5	**VGA color palette snoop enable bit.** Hardwired to zero. Only VGA-compatible devices and PCI-to-PCI bridges are required to implement this bit.
6	**Parity error response bit.** When set to one, the '591 asserts PERR# when a data parity error is detected. Also used to qualify the assertion of SERR# on address phase parity error. Reset clears this bit.
7	**Stepping enable bit.** Hardwired to zero because the '591 never uses address or data stepping.
8	**System error response bit.** When set to one, the '591 is enabled to assert SERR# (if the PARITY ERROR RESPONSE bit is also set to one) when address phase parity error detected or system DRAM parity error. Reset clears this bit.
9	**Fast back-to-back enable bit.** Hardwired to zero because the '591 never performs fast back-to-back transactions.
15:10	**Reserved** and hardwired to zero.

Status Register

Table 23-2 defines the '591's usage of its status register bits.

Table 23-2. '591's Status Register Bit Assignment

Bit	Description
6:0	**Reserved** and hardwired to zero.
7	**Fast back-to-back capable bit.** Hardwired to one, indicating that, when acting as a target, the '591 supports fast back-to-back transactions to different targets.
8	**Signaled parity error bit.** Set to one when the '591, acting as a master, samples PERR# asserted by the target during a write or the '591 asserts PERR# on a read. Reset clears this bit to zero.
10:9	**DEVSEL timing.** Hardwired to 01b, indicating that the '591 has a medium speed PCI address decoder.
11	**Signaled target abort.** Hardwired to zero because the '591 never signals a target abort.
12	**Received target abort.** Set to one when the '591 receives a target abort from a target when acting as master. Reset clears this bit to zero.
13	**Received master abort.** Set to one when the '591 experiences a master abort when acting as master. Reset clears this bit to zero.
14	**Signaled system error.** Set to one by the '591 when it assert SERR#. Reset clears this bit to zero. The SERR# ENABLE and PARITY ERROR RESPONSE bits in the '591's command register must be set to enable the '591 to generate SERR# and set this bit.
15	**Received parity error.** Set to one when the '591 detects an address or data phase parity error. Reset clears this bit to zero.

Revision ID Register

The revision ID register contains 00h in the first release of the '591.

Class Code Register

The class code register contains 060000h. 06h specifies the bridge class. The middle byte, 00h, specifies that the sub-class is host/PCI bridge. The lower byte is always 00h for all revision 2.x-compliant devices.

Cache Line Size Configuration Register

Not implemented. Since the '591 contains the cache controller, it "knows" the system cache line size.

Latency Timer Register

Hardwired with a value of 10h (16d). When acting as a PCI bus master, the '591 never performs bursts of longer than two data phases. The specification states that any device that never performs more than two data phases may hardwire a value into its LT, but the value may not exceed 16d.

Header Type Register

Hardwired with the value 00h. This indicates that the '591 is a single-function device (bit 7 = 0) and that the format of configuration doublewords 4 through 15 adheres to the header type zero definition.

BIST Register

Hardwired to 00h. Bit 7 = 0 indicates that the '591 does not implement a built-in self test.

Base Address Registers

None implemented. The '591 utilizes device-specific registers to set up its system DRAM address decoders. Regarding I/O, the '591 only implements two I/O ports: the configuration address port at I/O address 0CF8h and the configuration data port at I/O address 0CFCh. It has hardwired address decoders for these registers.

Expansion ROM Base Address Register

Not implemented because the '591 does not incorporate a PCI device ROM.

Interrupt Line Register

Not implemented because the '591 does not generate interrupt requests.

Interrupt Pin Register

Hardwired with 00h, indicating that the '591 does not implement a PCI interrupt request output pin.

Min_Gnt Register

The '591 incorporates the PCI bus arbiter and already knows its timeslice and intrinsically knows its own bus acquisition latency requirements.

Max_Lat Register

See Min_Gnt register section (previous section).

Bus Number Register

Hardwired to 00h, indicating that the PCI bus residing directly behind the '591 is PCI bus zero.

Subordinate Bus Number Register

Hardwired to FFh. Any software requests to perform special cycles or configuration reads or writes on buses other than bus zero are therefore passed through the '591 as type one configuration accesses. If the target bus doesn't exist, the type one configuration access will terminate in a master abort.

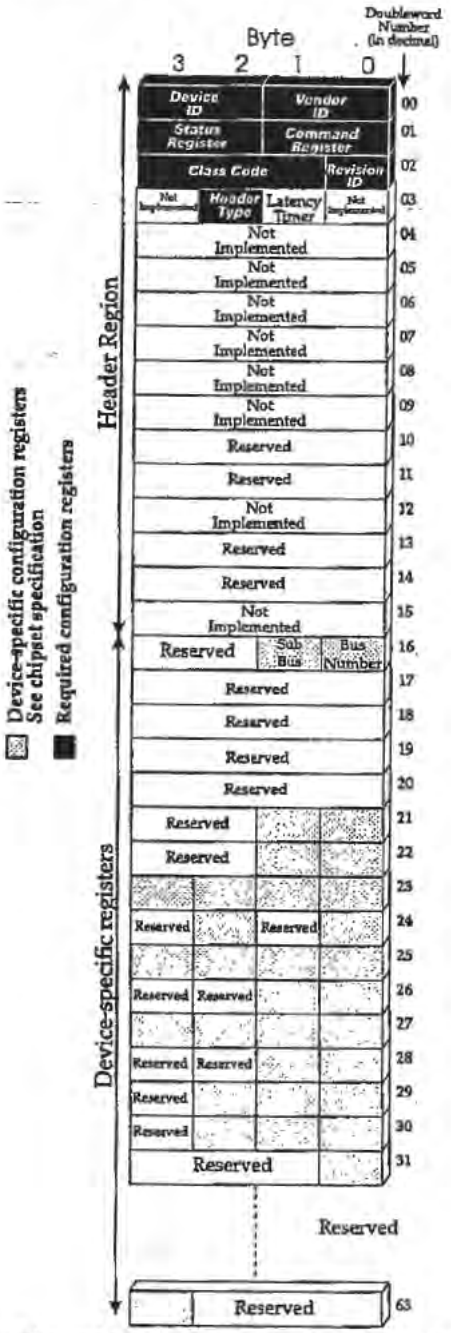

Figure 23-3. '591 PCI Configuration Registers

PCI Device Selection (IDSEL)

When translating accesses to the configuration data port into PCI type zero configuration accesses, the '591 internally decodes the target device number specified in the configuration address port and selects an IDSEL to assert. Rather than implementing an IDSEL output for each physical device position on the PCI bus, the '591 asserts one of the upper AD lines during the address phase of the PCI configuration access. Table 23-3 defines the AD line asserted (set to one) for each device number that may be specified. On the system board, each physical device position resistively-couples one of the upper AD lines to the device's IDSEL input pin.

Table 23-3. Device Number To AD Line Mapping

Device Number Specified (decimal)	AD Line Asserted
0	11 *
1	12
2	13
3	14
4	15
5	16
6	17
7	18
8	19
9	20
10	21
11	22
12	23
13	24
14	25
15	26
16	27
17	28
18	29
19	30
20	31 *
21 - 31	none

* Note: the '591 may be programmed to be either device 0 or device 31.

Handling of Host Processor-Initiated Transactions

Memory Read

When the host processor initiates a memory (code or data) read transaction, one of the following is true:

- The target location is within the range of system DRAM memory.
- The target location falls within the range assigned to system DRAM memory, but is within a sub-range defined as PCI or ISA memory.
- The target location is above the top of installed system memory.

In the first case, the address is considered to be cacheable. The lookaside L2 cache performs a lookup to determine if the requested data is present in the cache. If present, the cache tells the DRAM controller to abort the access to system DRAM and the cache supplies the requested data to the processor. If the requested data isn't currently present in the L2 cache, the DRAM controller proceeds with the memory read to fetch the target line. If the read hits on any posted-writes currently outstanding in the posted-write buffer, the write data is merged with the data read from memory and the requested data is supplied to the processor. The L2 cache also latches a copy of the line of information. If the L2 cache line size is 64 bytes, the cache initiates a second line read from memory to load the second half of its line into the L2 cache.

In the second and third cases, the '591 arbitrates for ownership of the PCI bus and initiates a memory read transaction (note that if the processor already has a PCI memory write posted in the '591, the posted write will be flushed to PCI memory before the PCI memory read is initiated). Because the L2 and L1 caches are not permitted to cache from memory beyond the bridge, the resulting PCI memory read transaction does not fetch an entire line (32 bytes) from memory. Rather, the access consists of one or two data phases (at most, the processor is expecting to get back eight bytes of information). When the PCI memory read transaction is initiated, DEVSEL# is asserted if a memory target on the PCI bus recognizes that it is the target of the transaction. That target then supplies the requested data to the '591 and the '591 supplies it to the processor. If no PCI memory target asserts DEVSEL#, the subtractive decoder built into the '593 eventually asserts DEVSEL# and claims the transaction. The transaction is passed to the ISA bus. If an ISA memory target recognizes the address, that target supplies the data. If the address is not recognized by any ISA memory target, the ISA bus controller in the '593 latches all ones from the

ISA data bus (its quiescent state when not being driven) and that is sent back to the '591 and then to the processor.

Note that the '591 can be programmed to recognize that specific PCI memory regions support prefetching. In this case, when the '591 performs the PCI memory read, it asserts all four byte enables and fetches the entire doubleword being addressed in the data phase (even if the processor had only requested a subset of the doubleword). The requested data is fed back to the processor and the prefetched bytes within the doubleword are stored in the '591's read-ahead buffer. This buffer can hold a quadword of data. If the processor should subsequently request any of the prefetched data, the data is supplied from the read-ahead buffer and the '591 does not perform a PCI memory read transaction.

Memory Write

When the host processor initiates a memory write transaction, there are the same three cases:

- The target location is within the range of system DRAM memory.
- The target location falls within the range assigned to system DRAM memory, but is within a sub-range defined as PCI or ISA memory.
- The target location is above the top of installed system memory.

In the first case, the memory write is absorbed by the posted-write buffer (if the eight quadword FIFO isn't full). The processor can then initiate another transaction immediately. If the buffer is full, the processor's current memory write stalls.

In the second and third case, there are two possible cases:

- Memory write posting is enabled for the addressed area of PCI memory .
- Memory write posting is disabled for that area.

If write posting is enabled and the write buffer is currently-available, the memory write is absorbed by the buffer and the processor is permitted to end its transaction. The '591 then initiates the PCI memory write when it has acquired PCI bus ownership. If write posting is disabled in the target area, the processor is stalled until the PCI memory write has been completed on the PCI bus.

I/O Read

When the processor initiates an I/O read transaction, the target device is one of the following:

- Configuration address port at I/O location 0CF8h.
- Configuration data port at I/O location 0CFCh.
- A PCI or an ISA I/O target device.

In the first two cases, the transaction is not passed through to the PCI bus. These two ports are integrated into the '591. The '591 therefore supplies the requested data directly to the host processor.

In the third case, the '591 stalls the processor until the PCI target (or the '593) supplies the requested data. The data is then routed to the processor, concluding the transaction.

I/O Write

When the processor initiates an I/O write transaction, the target location is one of the following:

- Configuration address port at I/O location 0CF8h.
- Configuration data port at I/O location 0CFCh.
- A PCI or an ISA I/O target,

In the first two cases, the '591 accepts the write data into the target port and the transaction is not passed to the PCI bus.

In the third case, the transaction must be passed to the PCI bus. By default, the '591 does not post I/O writes, but it can be programmed to do so. Assuming I/O write posting is disabled, the processor is stalled until the '591 acquires ownership of the PCI bus and completes the PCI I/O write transaction.

Interrupt Acknowledge

In response to an external interrupt from an 8259A interrupt controller, the processor generates two, back-to-back interrupt acknowledge transactions. The first one is generated to command the interrupt controller to prioritize its pending requests. The processor does not transfer data during this transaction. The processor generates the second interrupt acknowledge to request the

interrupt vector associated with the highest-priority pending request. When the '591 detects the first interrupt acknowledge, it responds with BRDY# to permit the processor to end the transaction. This transaction is not passed through the bridge. When the '591 detects the initiation of the second interrupt acknowledge, it acquires ownership of the PCI bus and performs an interrupt acknowledge transaction. In response, the '593 internally generates two INTA (interrupt acknowledge) pulses to the two 8259A cores that reside inside the '593. During the second INTA, the interrupt controller gates the one byte vector onto PCI data path zero, AD[7:0], and asserts TRDY#. The '591 is already asserting IRDY# so the '591 latches the vector from the AD bus and terminates the transfer. During this period, the '591 has been stalling the processor by keeping BRDY# deasserted until the data is presented on the processor's data bus. The vector is placed on host data path zero, D[7:0], and BRDY# is asserted. The processor latches the vector, concluding the second interrupt acknowledge transaction.

Special Cycle

The host processor is capable of generating the following types of special cycle transactions:

- Shutdown.
- Flush.
- Halt.
- Writeback.
- Flush Acknowledge.
- Branch Trace Message.
- Stop/Grant.

The '591 only passes shutdown, halt and stop/grant through the bridge to the PCI bus. The shutdown special cycle causes the '591 to generate a PCI special cycle transaction with the shutdown message sent during the data phase. The halt special cycle causes the '591 to generate a PCI special cycle transaction with the halt message sent during the data phase. The stop/grant special cycle causes the '591 to generate a PCI special cycle transaction with the halt message sent during the data phase. The stop/grant message is differentiated from a halt by AD4 set to one during the address phase (rather than low for a halt). The '593 is designed to test the state of AD4 during the address phase to determine if the message is a halt or a stop/grant. The other processor-initiated special cycles have the following effects:

- The flush special cycle transaction causes the '591 to invalidate the L2 cache.
- The writeback special cycle transaction has no effect (because the L2 cache is not a writeback cache and therefore does not have any modified lines to be written back to memory).
- The flush acknowledge special cycle transaction is generated by the processor in response to assertion of its FLUSH# input when it has completed writing back all modified lines to memory and has cleared the L1 cache. The author believes (but isn't certain) that the '591 ignores this transaction.
- If enabled to do so, the processor generates the branch trace message special cycle transaction whenever a branch instruction is taken. The '591 ignores this transaction.

Handling of PCI-Initiated Transactions

General

The following sections describe how the '591 responds to transactions initiated on the PCI bus by other masters. It's important to note that the '591 does not support concurrent operation of the host and PCI buses. In other words, when a PCI master other than the '591 has acquired ownership of the PCI bus, it has also acquired ownership of the host bus. The '591 accomplishes this by asserting HOLD to the processor and waiting until HLDA is asserted, indicating that the processor has released ownership of the host bus. The '591's PCI arbiter then grants ownership of both buses to the PCI master. The transaction initiated by the PCI master is passed through to the host bus so that the PCI master can access system DRAM memory (if this is a memory read or write transaction that targets system DRAM). If the transaction is not a memory transaction, or it is a memory transaction, but the target address is not system DRAM memory, then the DRAM controller, L2 cache and the processor (which is not told to snoop) ignore the transaction.

PCI Master Accesses System DRAM

The '591 aliases all three of the PCI memory read transaction types (memory read, memory read line, memory read multiple) to the PCI memory read transaction. If it is a memory transaction and the target address is system DRAM, the following actions are taken:

- The processor's L1 cache is told to snoop the address (via AHOLD and EADS#) and report the snoop result (on HIT# and HITM#).
- The L2 cache performs a lookup.

If the L1 snoop results in a miss or a hit on a clean line (HITM# is not asserted), the PCI master is permitted to continue with the transaction.

1. If a read and the requested data is present in the L2 cache, the L2 cache supplies the data to the PCI master. This is a nice feature. If the PCI master is accessing a data structure that is shared by the host processor, the resulting L2 cache hits can result in remarkable performance for the PCI master.
2. If a read and the requested data is not present in the L2 cache, the DRAM controller accesses system DRAM and the data is supplied to the PCI master from system dram.
3. If a write, the posted-write buffer absorbs the write data from the PCI master. If the L2 and/or L1 caches have a hit, the cache copies of the line are invalidated. If the data isn't resident in either cache, the write has no effect on the caches. The memory write and invalidate command is treated as a memory write.

PCI Master Accesses PCI or ISA Memory

Although the transaction is presented on the host bus, it has no effect on the L1 or L2 caches or on system memory.

PCI Master Accesses Non-Existent Memory

In this case, the '593 asserts DEVSEL# (due to subtractive decode) and passes the transaction to the ISA bus. The '591 passes the transaction to the host bus, but it has no effect (because the target address is not within range of system DRAM memory).

I/O Read or Write Initiated by PCI Master

Although passed to the host bus by the '591, have no effect.

Special Cycle

The '591 ignores special cycle transactions generated by PCI masters.

Type 0 Configuration Read or Write

A PCI master may access the '591's PCI configuration registers by directly generating the standard type 0 configuration read and write transactions. It cannot use the configuration address and data ports to stimulate the '591 to generate configuration transactions because the '591 would then have to use the PCI bus at the same time that the PCI master was using it.

Type 1 Configuration Read or Write

A PCI master may use a type one configuration transaction to access the configuration registers in a device on another PCI bus or to cause the generation of a special cycle on a specified target PCI bus. The '591 is unaffected by these transaction types.

Dual-Address Command (64-bit Addressing)

Because the system DRAM memory resides in the area up to but not above the 1GB address boundary, the detection of a PCI dual-address command has no effect on the '591.

Support for Fast Back-to-Back Transactions

The '591 can act as the target of fast back-to-back transactions, but cannot initiate them when acting as a PCI master.

'593 PCI/ISA Bridge

The '593 provides the following functionality:

- Bridges the PCI and ISA buses.
- Two 8259A interrupt controllers and the APIC I/O module.
- Two 8237A DMA controllers and their associated page registers.
- Real-Time Clock and CMOS RAM function (or, alternately, supports external RTC/CMOS).
- Port B logic.
- Tunable subtractive decoder (can be tuned to assert DEVSEL# in slow time slot).
- Programmable decoders for memory regions that exists on the ISA bus. This permits fast address decode of PCI accesses to the ISA bus. It also

permits memory accesses initiated by ISA bus masters or DMA channels to be contained on the ISA bus and not be passed to the PCI bus.

- Handles shutdown-to-processor INIT conversion. When a PCI special cycle transaction is detected with the shutdown message, the '593 toggles INIT to force a system reboot.
- A20 mask function.
- Processor self-test initiation.
- FERR# to IRQ13 conversion.
- NMI generation. The '593 generates NMI when CHCHK is asserted on the ISA bus, or when SERR# is asserted on the PCI bus.
- Hot reset generation.
- System Management Mode (SMM) interrupt logic.
- Monitors up to 27 different events related to power management.
- Includes watchdog timer for SMM usage.
- Supports software generation of the system management interrupt.
- Includes logic utilized to stop the processor clock.
- Positive decode for system ROM and keyboard controller, permitting fast address decode within these ranges.
- POWERGOOD/Reset logic.
- Speaker timer.
- Support for turbo mode. Can be used to periodically stop the processor's clock to make the processor appear to run slower.
- Integrated X-bus buffers.
- Can increase ISA BUSCLOCK speed up to 16MHZ within specified ISA memory regions.
- Decoupled ISA memory refresh. ISA memory refresh can occur on the ISA bus while PCI transactions are in progress.
- Supports disabling of internal DMA controllers (permitting implementation of an external DMA controller).

'593 Handling of Transactions Initiated by PCI Masters

The '593 responds to PCI transactions as follows:

- When acting as the target of a multiple-data phase PCI transaction, the '593 always disconnects the master upon completion of the first data phase.

- The '593 treats the PCI memory read line and read multiple commands as a memory read.
- The '593 treats the PCI memory write and invalidate command as a memory write.
- The '593 ignores the PCI dual-address command.
- The '593 can respond as the target of fast back-to-back transactions, but cannot generate them.
- The '593 responds to the PCI interrupt acknowledge transaction by claiming the transaction and returning the interrupt vector on the lower data path.
- When the '593 detects a PCI special cycle transaction, it determines if the message delivered during the data phase is a shutdown or a halt (it ignores all other message types). If a shutdown, it issues INIT to the processor to reboot the system. If a halt, it examines the state of AD4 from the transaction's address phase. If AD4 = 0, the processor is halting. The SMM logic can be programmed to take action on this event. If AD4 = 1, the message is really a stop/grant message. This informs the SMM logic that the processor has stopped its clock in response to assertion of STPCLK# by the '593.
- The '593 ignores type one configuration read and writes. A type zero configuration read or write that asserts the '593's IDSEL is permitted to access the '593's PCI configuration registers.
- The '593 handles address and data phase parity errors according to the revision 2.x PCI specification.
- The posted-memory write buffer in the '593 can be enabled/disabled by software. When enabled, the buffer accepts up to 32-bits of write data being presented by a PCI master and then disconnects. It then performs the memory write on the ISA bus. If a PCI master attempts to access the ISA bus while the posted-write is being performed on the ISA bus, the '593 stalls it (by keeping TRDY# deasserted) until the write completes.

Subtractive Decode Capability

The subtractive decoder in the '593 claims any unclaimed PCI memory access (even to addresses above 16MB). This means that areas of memory space above the top of system DRAM memory and above 16MB and not populated by PCI memory devices alias down into ISA's 16MB memory address space.

The subtractive decoder can be tuned to respond with slow DEVSEL#, rather than the normal subtractive DEVSEL# one clock after the slow time slot.

'593 Characteristics When Acting as PCI Master

The '591 incorporates the PCI bus arbiter. When the '593 must pass an ISA bus master or DMA transaction onto the PCI bus, the '593 must issue a request to the '591 and await assertion of its grant. The '59x chipset can handle the arbitration between the '591 and the '593 in one of two ways (software-selectable):

1. The '593 normally drives its REQGNT# output high. When it requires access to the PCI bus, it drives REQGNT# low for one cycle of the processor's bus clock. One clock after that, the '591 takes ownership of the REQGNT# signal and drives it high. The '591 continues to drive REQGNT# high until the arbiter is going to grant the bus to the '593. The '591 then drives REQGNT# low for one cycle of the processor's bus clock. One clock after that, the '593 resumes ownership of the REQGNT# signal and continues to drive it low until it has completed using the PCI bus. When the '593 has concluded using the PCI bus, it drives REQGNT# high and leaves it high until it requires the PCI bus again.
2. Alternately, the '593 and '591 can use a normal PCI REQ#/GNT# signal pair for '593 bus arbitration.

When acting as the PCI master, the '593 cannot generate fast back-to-back transactions.

The '593 recognizes that the arbiter is parking the bus on it when it detects its GNT# asserted and the bus is idle. The '593 then takes ownership of the AD and C/BE buses and drives them low. One clock after driving them low, the '593 drives PAR low.

The '593 only initiates transactions on the PCI bus because:

- A DMA channel is performing a transfer to or from system memory.
- An ISA bus master is performing a system memory or an I/O read or write.

The only types of PCI transactions it generates are therefore memory and I/O read and write transactions.

Interrupt Support

The '593 incorporates two 8259A interrupt controllers and an advanced programmable interrupt controller (APIC) I/O module. All of the system interrupt request signals, IRQ[15:0], are connected in parallel to the pair of 8259A's and to the APIC. This permits the programmer to set up the '593's interrupt logic to handle requests in an 8259A-compatible manner, or, in a multiprocessing environment, to route all interrupts from ISA and PCI to the APIC for delivery to the processors. A description of 8259A and APIC operation is outside the scope of this book. A complete description of the 8259A interrupt controller operation can be found in the Addison-Wesley publication (also authored by MindShare) entitled *ISA System Architecture*. A complete description of the APIC operation can be found in the Addison-Wesley publication (also authored by MindShare) entitled *Pentium Processor System Architecture*.

Eleven of the system IRQ inputs, IRQ[15:14], [12:9], and [7:3] may be individually programmed as either shareable or non-shareable interrupt request lines. Of these eleven, the '593's four PCI interrupt inputs may be routed to any of eight of them (IRQ[15:14], [12:9], [5], and [3]).

DMA Support

The '593 incorporates two 8237A DMA controllers in a master/slave configuration. It also incorporates the page registers necessary to extend the start address registers to a full 32 bits. This enables the DMA controller to transfer data to or from memory anywhere within the 4GB memory address space. It is a constraint, however, that the specified DMA transfer in memory must reside fully with a 64KB-aligned block of memory space (because the 8237A DMA controller cannot issue a carry to the page register when incrementing over a 64KB address boundary).

'593 Configuration Registers

Figure 23-4 illustrates the '593's usage of its PCI configuration space. The sections that follow define the manner in which the chipset implements each of these registers. For a description of the '593's device-specific configuration registers, refer to the chipset specification.

Vendor ID Register

The vendor ID for VLSI Technology is 1004h.

Device ID Register

The device ID for the '593 is 0006h.

Command Register

Table 23-4 defines the '593's usage of its command register bits.

Table 23-4. '593's Command Register Bit Assignment

Bit	Description
0	**I/O enable bit.** When set one, the '593 is enabled to respond to PCI I/O transactions. Reset sets this bit to one.
1	**Memory enable bit.** When set to one, PCI bus masters can access ISA memory. Reset sets this bit to one.
2	**Master enable bit.** When set to one, the '593 is enabled to initiate PCI transactions. Reset sets this bit to one.
3	**Special cycle monitor enable bit.** When set to one, the '593 recognizes special cycles (only shutdown and halt) generated by other PCI masters (usually, the '591). Reset sets this bit to one.
4	**Memory write and invalidate enable bit.** Hardwired to zero because the '593 never generates the memory write and invalidate command.
5	**VGA color palette snoop enable bit.** Hardwired to zero. Only VGA-compatible devices and PCI-to-PCI bridges are required to implement this bit.
6	**Parity error response bit.** When set to one, the '593 asserts PERR# when a data parity error is detected. Also used to qualify the assertion of SERR# on address phase parity error. Reset clears this bit.
7	**Stepping enable bit.** Hardwired to zero because the '593 never uses address or data stepping.
8	**System error response bit.** When set to one, the '593 is enabled to assert SERR# (if the PARITY ERROR RESPONSE bit is also set to one) when address phase parity error detected. Reset clears this bit.
9	**Fast back-to-back enable bit.** Hardwired to zero because the '593 never performs fast back-to-back transactions.
15:10	**Reserved** and hardwired to zero.

Status Register

The '593's configuration status register bit assignment is defined in table 23-5.

Table 23-5. '593's Status Register Bit Assignment

Bit	Description
6:0	**Reserved** and hardwired to zero.
7	**Fast back-to-back capable bit.** Hardwired to one, indicating that, when acting as a target, the '593 supports fast back-to-back transactions to different targets.
8	**Signaled parity error bit.** Set to one when the '593, acting as a master, samples PERR# asserted by the target during a write or the '593 asserts PERR# on a read. Reset clears this bit to zero.
10:9	**DEVSEL timing.** Hardwired to 01b, indicating that the '593 has a medium speed PCI address decoder.
11	**Signaled target abort.** Set to one when the '593 has signaled a target abort to the initiator of a transaction. Reset clears this bit to zero.
12	**Received target abort.** Set to one when the '593 receives a target abort from a target when acting as master. Reset clears this bit to zero.
13	**Received master abort.** Set to one when the '593 experiences a master abort when acting as master. Reset clears this bit to zero.
14	**Signaled system error.** Set to one by the '593 when it assert SERR#. Reset clears this bit to zero. The SERR# ENABLE and PARITY ERROR RESPONSE bits in the '593's command register must be set to enable the '593 to generate SERR# and set this bit.
15	**Received parity error.** Set to one when the '593 detects an address or data phase parity error. Reset clears this bit to zero.

Revision ID Register

The revision ID register contains 00h in the first release of the '593.

Class Code Register

The class code register contains 060100h. 06h specifies the bridge class. The middle byte, 01h, specifies that the sub-class is ISA/PCI bridge. The lower byte is always 00h for all revision 2.x-compliant devices.

Cache Line Size Configuration Register

Not implemented.

Latency Timer Register

Not implemented. When acting as a PCI master, the '593 only performs single data phase transactions initiated by ISA bus masters and DMA channels.

Header Type Register

Hardwired with the value 00h. This indicates that the '593 is a single-function device (bit 7 = 0) and that the format of configuration doublewords four through 15 adheres to the header type zero definition.

BIST Register

Hardwired to 00h. Bit 7 = 0 indicates that the '593 does not implement a built-in self test.

Base Address Registers

None implemented. The '593 utilizes device-specific registers to set up its ISA, ROM and I/O address decoders.

Expansion ROM Base Address Register

Not implemented because the '593 does not incorporate a PCI device ROM.

Interrupt Line Register

Not implemented because the '593 does not generate interrupt requests for itself.

Interrupt Pin Register

Hardwired with 00h, indicating that the '593 does not implement a PCI interrupt request output pin.

Min_Gnt Register

Hardwired to 00h, indicating that the '593 has no specific requirements regarding the timeslice assigned to it. In addition, the '593 does not implement the LT, so the Min_Gnt register is a moot point.

Max_Lat Register

Hardwired to 00h, indicating that the '593 has no specific requirements regarding its arbitration priority level.

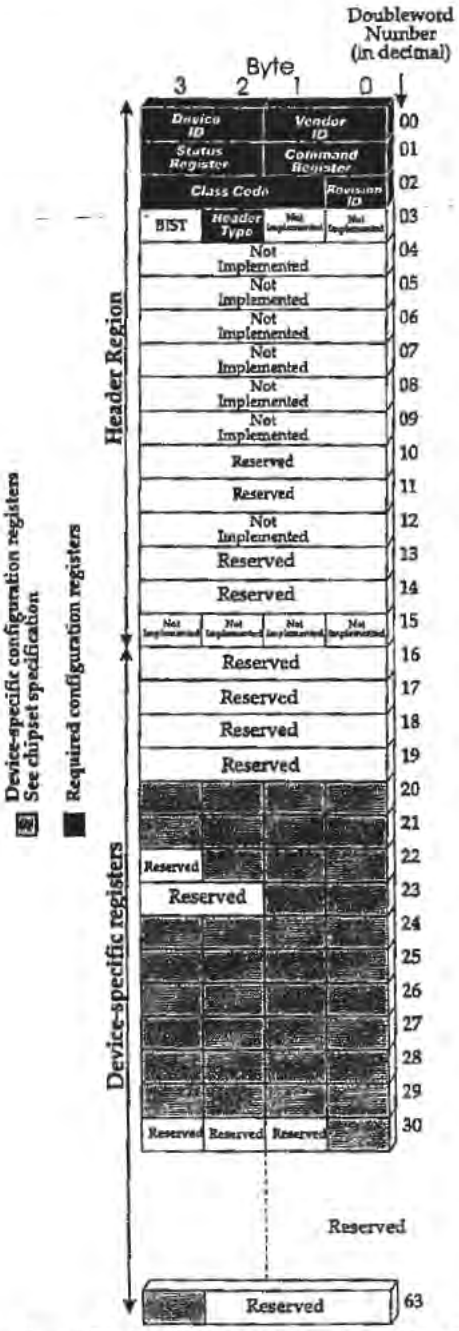

Figure 23-4. VL82C593's Configuration Registers

Access Latency. The amount of time that expires from the moment a bus master requests the use of the PCI bus until it completes the first data transfer of the transaction.

AD Bus. The PCI address/data bus carries address information during the address phase of a transaction and data during each data phase.

Address Ordering. During PCI burst memory transfers, the initiator must indicate whether the addressing sequence will be sequential (also referred to as linear) or will use cacheline wrap ordering of addresses. The initiator uses the state of AD[1:0] to indicate the addressing order. During I/O accesses, there is no explicit or implicit address ordering. It is the responsibility of the programmer to understand the I/O addressing characteristic of the target device.

Address Phase. During the first clock period of a PCI transaction, the initiator outputs the start address and the PCI command. This period is referred to as the address phase of the transaction. When 64-bit addressing is used, there are two address phases.

Agents. Each PCI device, whether a bus master (initiator) or a target is referred to as a PCI agent.

Arbiter. The arbiter is the device that evaluates the pending requests for access to the bus and grants the bus to a bus master based on a system-specific algorithm.

Arbitration Latency. The period of time from the bus master's assertion of REQ# until the bus arbiter asserts the bus master's GNT#. This period is a function of the arbitration algorithm, the master's priority and system utilization.

Atomic Operation. A series of two or more accesses to a device by the same initiator without intervening accesses by other bus masters.

Base Address Registers. Device configuration registers that define the start address, length and type of memory space required by a device. The type of space required will be either memory or I/O. The value written to this register during device configuration will program its memory or I/O address decoder to detect accesses within the indicated range.

BIST. Some integrated devices (such as the i486 microprocessor) implement a built-in self-test that can be invoked by external logic during system start up.

Bridge. The device that provides the bridge between two independent buses. Examples would be the bridge between the host processor bus and the PCI bus, the bridge between the PCI bus and a standard expansion bus (such as the ISA bus) and the bridge between two PCI buses.

Bus Access Latency. Defined as the amount of time that expires from the moment a bus master requests the use of the PCI bus until it completes the first data transfer of the transaction. In other words, it is the sum of arbitration, bus acquisition and target latency.

Bus Acquisition Latency. Defined as the period time from the reception of GNT# by the requesting bus master until the current bus master surrenders the bus and the requesting bus master can initiate its transaction by asserting FRAME#. The duration of this period is a function of how long the current bus master's transaction-in-progress will take to complete.

Bus Concurrency. Separate transfers occurring simultaneously on two or more separate buses. An example would be an EISA bus master transferring data to or from another EISA device while the host processor is transferring data to or from system memory.

Bus Idle State. A transaction is not currently in progress on the bus. On the PCI bus, this state is signalled when FRAME# and IRDY# are both deasserted.

Bus Lock. Gives a bus master sole access to the bus while it performs a series of two or more transfers. This can be implemented on the PCI bus, but the preferred method is resource locking. The EISA bus implements bus locking..

Bus Master. A device capable of initiating a data transfer with another device.

Bus Parking. An arbiter may grant the buses to a bus master when the bus is idle and no bus masters are generating a request for the bus. If the bus master that the bus is parked on subsequently issues a request for the bus, it has immediate access to the bus.

Byte Enable. I486, Pentium™ or PCI Bus control signal that indicates that a particular data path will be used during a transfer. Indirectly, the byte enable

signal also indicates what byte within an addressed doubleword (or quadword, during 64-bit transfers) is being addressed.

Cache. A relatively small amount of high-speed Static RAM (SRAM) that is used to keep copies of information recently read from system DRAM memory. The cache controller maintains a directory that tracks the information currently resident within the cache. If the host processor should request any of the information currently resident in the cache, it will be returned to the processor quickly (due to the fast access time of the SRAM).

Cache Controller. See the definition of **Cache**.

Cache Line Fill. When a processor's internal cache, or its external second level cache has a miss on a read attempt by the processor, it will read a fixed amount (referred to as a line) of information from the external cache or system DRAM memory and record it in the cache. This is referred to as a cache line fill. The size of a line of information is cache controller design dependent.

Cache Line Size. See the definition of **Cache Line Fill.**

CacheLine Wrap Mode. At the start of each data phase of the burst read, the memory target increments the doubleword address in its address counter. When the end of the cache line is encountered and assuming that the transfer did not start at the first doubleword of the cache line, the target wraps to start address of the cacheline and continues incrementing the address in each data phase until the entire cache line has been transferred. If the burst continues past the point where the entire cache line has been transferred, the target starts the transfer of the next cache line at the same address that the transfer of the previous line started at.

CAS Before RAS Refresh, or CBR Refresh. Some DRAMs incorporate their own row counters to be used for DRAM refresh. The external DRAM refresh logic has only to activate the DRAM's CAS line and then its RAS line. The DRAM will automatically increment its internal row counter and refresh (recharge) the next row of storage.

CBR Refresh. See the definition of **CAS Before RAS Refresh.**

Central Resource Functions. Functions that are essential to operation of the PCI bus. Examples would be the PCI bus arbiter and "keeper" pullup resistors

that return PCI control signals to their quiescent state or maintain them at the quiescent state once driven there by a PCI agent.

Claiming the Transaction. An initiator starts a PCI transaction by placing the target device's address on the AD bus and the command on the C/BE bus. All PCI targets latch the address on the next rising-edge of the PCI clock and begin to decode the address to determined if they are being addressed. The target that recognizes the address will "claim" the transaction by asserting DEVSEL#.

Class Code. Identifies the generic function of the device (for example, a display device) and, in some cases, a register-specific programming interface (such as the VGA register set). The upper byte defines a basic class type, the middle byte a sub-class within the basic class, and the lower byte may define the programming interface.

Coherency. If the information resident in a cache accurately reflects the original information in DRAM memory, the cache is said to be coherent or consistent.

Commands. During the address phase of a PCI transaction, the initiator broadcasts a command (such as the memory read command) on the C/BE bus.

Compatibility Hole. The DOS compatibility hole is defined as the memory address range from 80000h - FFFFFh. Depending on the function implemented within any of these memory address ranges, the area of memory will have to be defined in one of the following ways: Read-Only, Write-Only, Read/Writable, Inaccessible.

Concurrent Bus Operation. See the definition of **Bus Concurrency.**

Configuration Access. A PCI transaction to read or write the contents of one of a PCI device's configuration registers.

Configuration Address Space. x86 processors possess the ability to address two distinct address spaces: I/O and memory. The PCI bus uses I/O and memory accesses to access I/O and memory devices, respectively. In addition, a third access type, the configuration access, is used to access the configuration registers that must be implemented in all PCI devices.

Configuration CMOS RAM. The information used to configure devices each time an ISA, EISA or Micro Channel™ machine is powered up is stored in battery backed-up CMOS RAM.

Configuration Header Region. Each functional PCI device possesses a block of two hundred and fifty-six configuration addresses reserved for implementation of its configuration registers. The format, or usage, of the first sixty-four locations is predefined by the PCI specification. This area is referred to as the device's configuration header region.

Consistency. See the definition of **Coherency.**

Data Packets. In order to improve throughput, a PCI bridge may consolidate a series of single memory reads or writes into a single PCI memory burst transfer.

Data Phase. After the address phase of a PCI transaction, a data item will be transferred during each data phase of the transaction. The data is transferred when both TRDY# and IRDY# are sampled asserted.

Deadlock. A deadlock is a condition where two masters each require access to a target simultaneously, but the action taken by each will prevent the desired action of the other.

Direct-Mapped Cache. In a direct-mapped cache, the cache is the same size as a page in memory. When a line of information is fetched from a position within a page of DRAM memory, it is stored in the same relative position within the cache. If the processor should subsequently request a line of information from the same relative position within a different page of memory, the cache controller will fetch the new line from memory and must overwrite the line currently in the cache.

Dirty Line. A write-back cache controller has cached a line of information from memory. The processor subsequently performs a memory write to a location within the cache line. There is a hit on the cache and the cache line is updated with the new information, but the cache controller will not perform a memory write bus cycle to update the line in memory. The line in the cache no longer reflects the line in memory and now has the latest information. The cache line is said to be "dirty" and the memory line is "stale." Dirty is another way of saying "modified."

Disconnect. A very slow access target may force a disconnect between accesses to give other initiators a chance to use the bus. This is known as a disconnect. If the current access is quite long and will consume a lot of bus time, the target signals a disconnect, completes the current data phase, and the initiator terminates the transaction. The initiator then arbitrates for the bus again so that it may re-initiate the transaction, continuing the data transfer at the point of disconnection.

DOS Compatibility Hole. See the definition of **Compatibility Hole.**

DRAM controller. The DRAM controller converts memory read or write bus cycles on the host or PCI bus to the proper sequence of actions on the memory bus to access the target DRAM location.

EISA. Extension to Industry Standard Architecture. The EISA specification was developed to extend the capabilities of the ISA machine architecture to support more advanced features such as bus arbitration and faster types of bus transfers.

Exclusive Access. A series of accesses to a target by one bus master while other masters are prevented from accessing the device.

Expansion ROM. A device ROM related to a PCI function. This ROM typically contains the initialization code and possibly the BIOS and interrupt service routines for its associated device.

Functional PCI Devices. A PCI device that performs a single, self-contained function. Examples would be a video adapter or a serial port. A single, physical PCI component might actually contain from one to eight PCI functional devices.

Hidden bus arbitration. Unlike some arbitration schemes, the PCI scheme allows bus arbitration to take place at the same time that the current PCI bus master is performing a data transfer. No bus time is wasted on a dedicated period of time to perform an arbitration bus cycle. This is referred to as hidden arbitration.

Hidden Refresh. If a DRAM controller uses the memory bus to perform DRAM refresh when the system is currently accessing a device other than DRAM memory, this is referred to as hidden refresh. Refreshing DRAM in this fashion doesn't impact system performance.

Hierarchical PCI Buses. When one PCI bus is subordinate to another PCI bus, they are arranged in a hierarchical order. A PCI-to-PCI bridge would interconnect the two buses.

Hit. Refers to a hit on the cache. The processor initiates a memory read or write and the cache controller has a copy of the target line in its cache.

Host/PCI Bridge. A device that provides the bridge between the host processor's bus and a PCI bus. The bridge provides transaction translation in both directions. It may also provide data buffering and/or a second-level cache for the host processor.

Idle State. See the definition for the **Bus Idle State.**

Incident-Wave Switching. See the definition of **Class I Driver.**

Initiator. When a bus master has arbitrated for and won access to the PCI bus, it becomes the initiator of a transaction. It then starts a transaction, asserting FRAME# and driving the address and command onto the bus.

Interrupt Acknowledge. The host ix86 processor responds to an interrupt request on its INTR input by generating two, back-to-back interrupt acknowledge bus cycles. If the interrupt controller resides on the PCI bus, the host/PCI bridge translates the two cycles into a single PCI interrupt acknowledge bus cycle. In response to an interrupt acknowledge, the interrupt controller must send the interrupt vector corresponding to the highest priority device generating a request back to the processor.

Interrupt Controller. A device requiring service from the host processor generates a request to the interrupt controller. The interrupt controller, in turn, generates a request to the host processor on its INTR signal line. When the host processor responds with an interrupt acknowledge, the interrupt controller prioritizes the pending requests and returns the interrupt vector of the highest priority device to the processor.

Level-Two, or L2, Cache. The host processor's internal cache is frequently referred to as the primary, or level-one cache. An external cache that attempts to service misses on the internal cache is referred to as the level-two cache.

Line. See the definition of **Cache Line Fill.**

Line Buffer. If a device has a buffer that can hold an entire line of information previously fetched from memory, the buffer is frequently referred to as a line buffer.

Linear Addressing. If a PCI master initiates a burst memory transfer and sets AD[1:0] equal to a 00b, this indicates to the addressed target that the memory address should be incremented for each subsequent data phase of the transaction.

Local Bus. Generally, refers to the processor's local bus structure. An example would be the 486's bus structure.

Look-Through Cache Controller. A cache controller that resides between its associated processor and the rest of the world is referred to as a look-through cache controller. Look-through cache controllers are divided into two categories: write-through and write-back.

Master Abort. If an initiator starts a PCI transaction and the transaction isn't claimed by a target (DEVSEL# asserted) within five PCI clock periods, the initiator aborts the transaction with no data transfer.

Master Latency Timer. Each bus master must incorporate a Latency Timer, or LT. The LT benefits both the current bus master and any bus master that may request access to the PCI bus while the current bus master is performing a transaction. The LT ensures that the current bus master will not hog the bus if the PCI bus arbitrator indicates that another PCI master is requesting access to the bus. Looking at it from another point of view, it guarantees the current bus master a minimum amount of time on the bus before it must surrender it to another master.

Master-Initiated Termination. The LT count may have expired and the transaction continued without preemption by the arbitrator (removal of its grant) because no other PCI master required the bus. Another bus master may then request and be granted the bus while the current master still has a transaction in progress. Upon sensing the removal of its grant by the arbitrator, the current bus master must initiate transaction termination at the end of the current data transfer with the target. This is referred to as master-initiated termination.

Memory Read Command. The PCI memory read command should be used when reading less than a cache line from memory.

Memory Read Line Command. This PCI command should be used to fetch one complete cache line from memory.

Memory Read Multiple Command. This command should be used to fetch multiple memory cache lines from memory. When this command is used, the target memory device should fetch the requested cache line from memory. When the requested line has been fetched from memory, the memory controller should start fetching the next line from memory in anticipation of a request from the initiator. The memory controller should continue to prefetch lines from memory as long as the initiator keeps FRAME# asserted.

Memory Write Command. The initiator may use the PCI memory write or the memory write and invalidate command to update data in memory.

Memory Write and Invalidate Command. The memory write and invalidate command is identical to the memory write except that it transfers a complete cache line during the current transaction. The initiator's configuration registers must allow specification of the line size during system configuration. If, when snooping, the cache/bridge's write-back cache detects a memory write and invalidate issued by the initiator and has a snoop hit on a line marked as dirty, the cache can just invalidate the line and doesn't need to perform the flush to memory. This is possible because the initiator is updating the entire memory line and all of the data in the dirty cache line is therefor invalid. This increases performance by eliminating the requirement for the line flush.

Message. An initiator may broadcast a message to one or more PCI devices by using the PCI Special Cycle command. During the address phase, the AD bus is driven to random values and must be ignored. The initiator does, however, use the C/BE lines to broadcast the special cycle command. During the data phase, the initiator broadcasts the message type on AD[15:0] and an optional message-dependent data field over AD[31:16]. The message and data are only valid during the clock after IRDY# is asserted. The data contained in, and the timing of subsequent data phases is message dependent.

Miss. Refers to a miss on the cache. The processor initiates a memory read or write and the cache controller does not have a copy of the target line in its cache.

Multi-Function Devices. A physical PCI component may have one or more independent PCI functions integrated within the package. A component that incorporates more than one function is referred to as a Multi-Function Device.

Non-Cacheable Memory. Memory whose contents can be altered by a local processor without using the bus should be designated as non-cacheable. A cache somewhere else in the system would have no visibility to the change and could end up with stale data and not realize that it no longer accurately reflects the contents of memory.

Packets. See the definition of **Data Packets.**

Page Register. The upper portion of the start memory address for a DMA transfer is written to the respective DMA channel's Page register. The contents of this register is then driven onto the upper part of the address bus during a DMA data transfer.

Parking. See the definition of **Bus Parking.**

PCI. See the definition of **Peripheral Component Interconnect.**

Peer PCI Buses. PCI buses that occupy the same ranking in the PCI bus hierarchy (with respect to the host bus) are referred to as peer PCI buses.

Peripheral Component Interconnect (or PCI). Specification that defines the PCI bus. This bus is intended to define the interconnect and bus transfer protocol between highly-integrated peripheral adapters that reside on a common local bus on the system board (or add-in expansion cards on the PCI bus).

Physical PCI Device. See the definition of **Functional PCI Devices.**

Point-To-Point Signals. Signals that provide a direct interconnect between two PCI agents. An example would be the REQ# and GNT# lines between a PCI bus master and the PCI bus arbiter.

Posted-Write Capability. The ability of a device to "memorize" or post a memory write transaction and signal immediate completion to the bus master. As long as there is room in the posted-write buffer, this permits the bus master to complete memory writes with no wait states. After posting the write

and signalling completion to the bus master, the device will then perform the actual memory write.

Preemption. Preemption occurs when the arbitrator removes the grant from one master and gives it to another.

Prefetching. Fetching a line of information from memory before a bus master requests it. If the master should subsequently request the line, the target device can supply it immediately. This shields the master from the slow access time of the actual target memory.

Primary Bus. The bus closest to the host processor that is connected to one side of an inter-bus bridge.

Reflected-Wave Switching. The output drivers commonly implemented in highly-integrated components fall into the class II category. They take advantage of the reflected-wave switching characteristic common to high-speed transmission lines and printed circuit traces in order to achieve the input logic thresholds. When a class II output driver transitions a signal line from a logic low to a high, the low-to-high transition is rather weak and only achieves half of the voltage change required to cross the logic threshold. This transition wavefront is transmitted along the trace and is seen sequentially by the input of each device connected to the line. When the wavefront gets to the end of the transmission line, it is reflected back along the trace, effectively doubling the voltage change and thereby boosting the voltage change past the logic threshold. As the wave passes each device's input, the new valid logic level is detected.

Reserved Bus Commands. Several of the PCI bus command codes are reserved for future use. Targets should ignore reserved bus commands.

Resource Lock. The PCI bus implements a signal, LOCK#. It allows a master (a processor) to reserve a particular target for its sole use until it completes a series of accesses to the target. The master then indicates that it no longer requires exclusive access to the target. One of the nice features of the PCI bus is that it permits other masters to use the bus to access targets other than the locked target during the period that a master has locked access to a particular target.

Retry. If a target cannot respond to a transaction at the current time, it will signal "retry" to the initiator and terminate the transaction. The initiator will

respond by ending the transaction and then retrying it later. No data transfer takes place during this transaction. An example of the need for a retry would be if the target is currently locked for exclusive access by another initiator.

SCSI. Small Computer System Interface. A bus designed to offload block data transfers from the host processor. The SCSI host bus adapter provides the interface between the host system's bus and the SCSI bus.

Secondary Bus. The bus further from the host processor that is connected to one side of an inter-bus bridge.

Sideband Signals. A sideband signal is defined as a signal that is not part of the PCI bus standard and interconnects two or more PCI agents. This signal only has meaning for the agents it interconnects.

Single-Function Devices. A physical PCI component may have one or more independent PCI functions integrated within the package. A component that incorporates only one function is referred to as a Single-Function Device.

Slave. Another name for the target being addressed during a transaction.

Snooping. When a memory access is performed by an agent other than the cache controller, the cache controller must snoop the transaction to determine if the current master is accessing information that is also resident within the cache. If a snoop hit occurs, the cache controller must take an appropriate action to ensure the continued consistency of its cached information.

Soft-Encoded Messages. The first two message codes, 0000h and 0001h, are defined as SHUTDOWN and HALT. Message code 0002h is reserved for Intel device-specific messages, while codes 0003h - through - FFFFh are reserved. Questions regarding the allocation of the reserved message codes should be forwarded to the PCI SIG. Also see the definition of **Message**.

Special Cycle Command. See the definition of **Message**.

Special Interest Group, or SIG. The PCI SIG manages the specification.

Speedway. Intel performed over 5000 hours of simulations in order to establish the best possible layout of the PCI bus on a high-frequency system board. The result is the Speedway (trademarked by Intel) definition. This

layout may be used for up to ten physical PCI components operating at speeds up to 33MHz.

Stale Information. See the definition of **Dirty Line.**

Standard Form Factor. Also referred to as the long card form factor, the standard form factor defines a PCI expansion board that is designed to fit into existent desktop machines with ISA, EISA or Micro Channel™ card slots.

Streaming Data Procedures. Advanced bus cycle types implemented on the more advanced Micro Channel machines.

Subordinate Bus Number. The subordinate bus number configuration register in a PCI-to-PCI bridge (or a host/PCI bridge) defines the bus number of the highest-numbered PCI bus that exists behind the bridge.

Subtractive Decode. The PCI-to-expansion bus bridge is designed to claim many transactions not claimed by other devices on the PCI bus. If the bridge doesn't see DEVSEL# asserted by a PCI target within four PCI clock periods from the start of a transaction, the bridge may assert DEVSEL# to claim the transaction and then pass the transaction onto the standard expansion bus (such as ISA, EISA or the Micro Channel).

Tag SRAM. The high-speed static RAM used as a directory by a cache controller.

Target. The PCI device that is the target of a PCI transaction initiated by a PCI bus master.

Target Latency. Defined as the period of time until the currently-addressed target is ready to complete the first data phase of the transaction. This period is a function of the access time for the currently-addressed target device.

Target-Abort. If the target detects a fatal error or will never be able to respond to the transaction, it must signal a target-abort (using the STOP# signal). This will cause the initiator to end the transaction with no data transfer and no retry.

Target-Initiated Termination. Under some circumstances, the target may have to end a transfer prematurely. The following are some examples. A very slow access target may force a disconnect between accesses to give other

initiators a chance to use the bus. This is known as a ***disconnect***. If the current access is quite long and will consume a lot of bus time, the target signals a disconnect, completes the current data transfer, and the initiator terminates the transaction. The initiator then arbitrates for the bus again so that it may re-initiate the transaction, continuing the data transfer at the point of disconnection. If a target cannot respond to a transaction at the current time, it will signal ***retry*** to the initiator and terminate the transaction. The initiator will respond by ending the transaction and then retrying it. No data transfer takes place during this transaction. An example of the need for a retry would be if the target is currently locked for exclusive access by another initiator. If the target detects a fatal error or will never be able to respond to the transaction, it may signal a ***target-abort***. This will cause the initiator to end the transaction with no data transfer and no retry.

Turn-Around Cycle. A turn-around cycle is required on all signals that may be driven by more than one PCI bus agent. This period is required to avoid contention when one agent stops driving a signal and another agent begins.

Type One Configuration Access. The type one access is used to configure a device on a lower-level PCI bus (in a system with hierarchical PCI buses).

Type Zero Configuration Access. The type zero access is used to configure a device on the PCI bus the configuration access is run on.

Utility Bus. The utility bus is located on the system board and is a buffered version of the standard expansion bus (ISA, EISA or the Micro Channel). Devices such as the keyboard controller, CMOS RAM, and floppy controller typically reside on the utility bus. This bus is also frequently referred to as the X-bus.

Vendor ID. Every PCI device must have a vendor ID configuration register that identifies the vendor of the device.

VESA. The Video Electronics Standards Association, or VESA, is a consortium of add-in card manufacturers tasked with developing standards for PC device interfacing.

VESA VL Bus. This is the local bus standard developed by the VESA consortium.

Video Electronics Standards Association (VESA). See the definition of **VESA.**

Video Memory. Memory that is dedicated to the storage of the video image to be scanned out to the display device.

VL bus. See the definition of **VESA VL Bus.**

VL Type A Local Bus. This is the direct-connect version of the VESA VL bus. For more information, refer to chapter two.

VL Type B Local Bus. This is the buffered version of the VESA VL bus. For more information, refer to chapter two.

Wait State. A delay of one PCI clock period injected into a PCI data phase because either the initiator (IRDY# deasserted), the target (TRDY# deasserted), or both are not yet ready to complete the data transfer.

Write Miss. The processor initiates a memory write and the cache controller does not have a copy of the target memory location within its cache.

Write-Back Cache. The write-back cache controller is a variant of the look-through cache controller. When the processor initiates a memory write bus cycle, the cache controller determines whether or not is has a copy of the target memory location within its cache. If it does, this is a write hit. The cache controller updates the line of information in its cache, but does not initiate a memory write bus cycle to update the line in DRAM memory. This permits processor-initiated memory writes to complete with no wait states. The cache line is now dirty and the memory line is stale. The cache controller will not flush its dirty lines to memory until later. In the event of a miss, the data is written to memory.

Write-Through Cache. The write-through cache controller is a variant of the look-through cache controller. When the processor initiates a memory write bus cycle, the cache controller determines whether or not is has a copy of the target memory location within its cache. If it does, this is a write hit. The cache controller updates the line of information in its cache, and also writes it through to DRAM memory. This ensures that the cache and DRAM memory are always in sync. In the event of a miss, the data is written to memory.

X-Bus. See the definition of **Utility Bus.**