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 Super-Core 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.

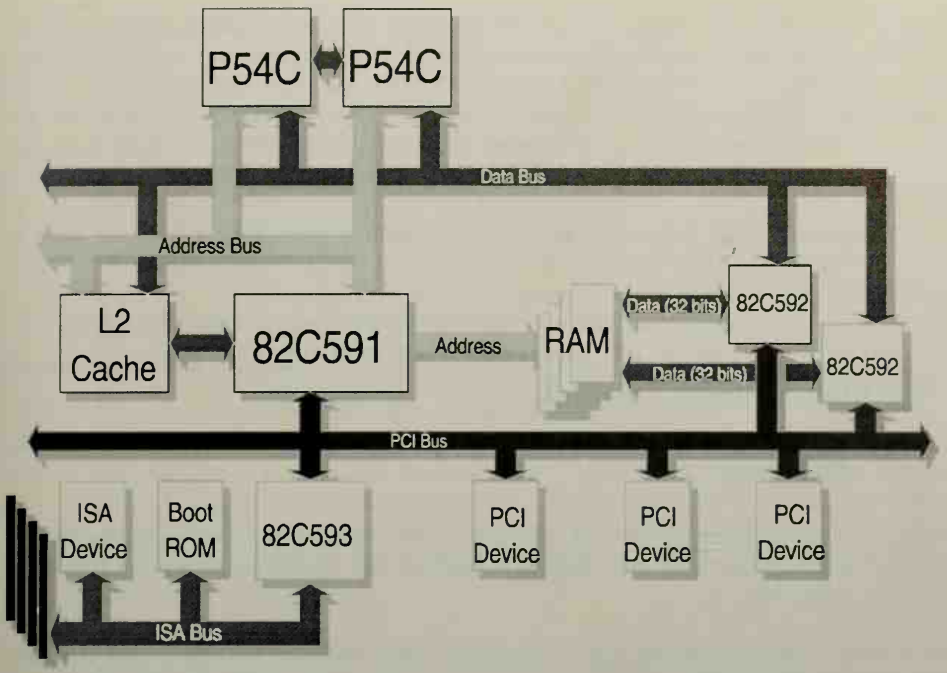

Figure 23-1. System Design Using VLSI VL82C59x SuperCore Chipset

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.

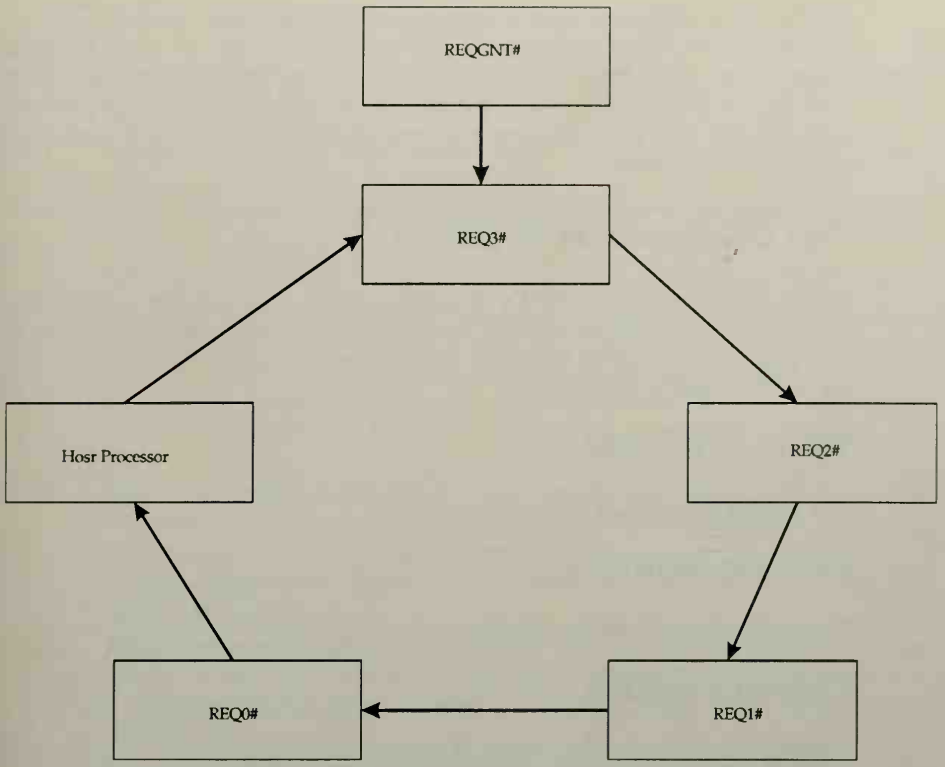

Figure 23-2. Rotational Priority Scheme

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.

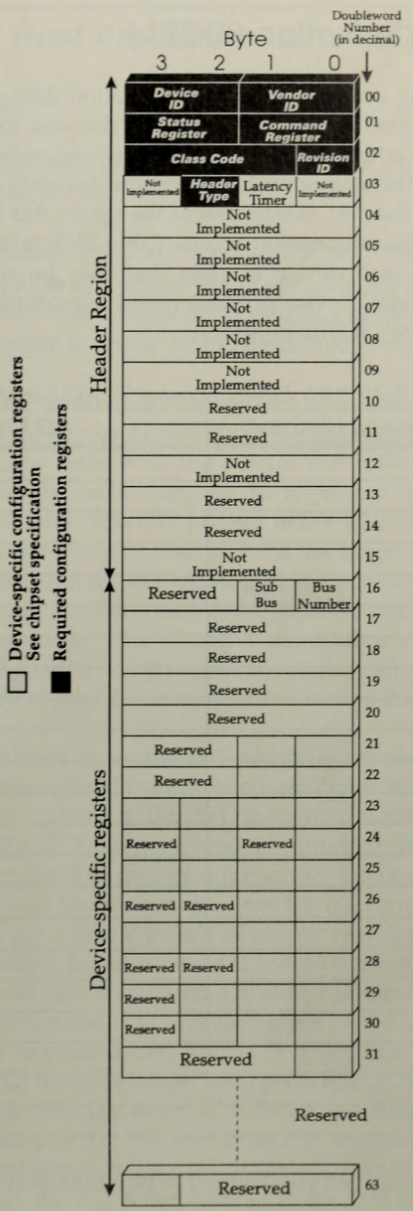

Figure 23-3. '591 PCI Configuration Registers

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.

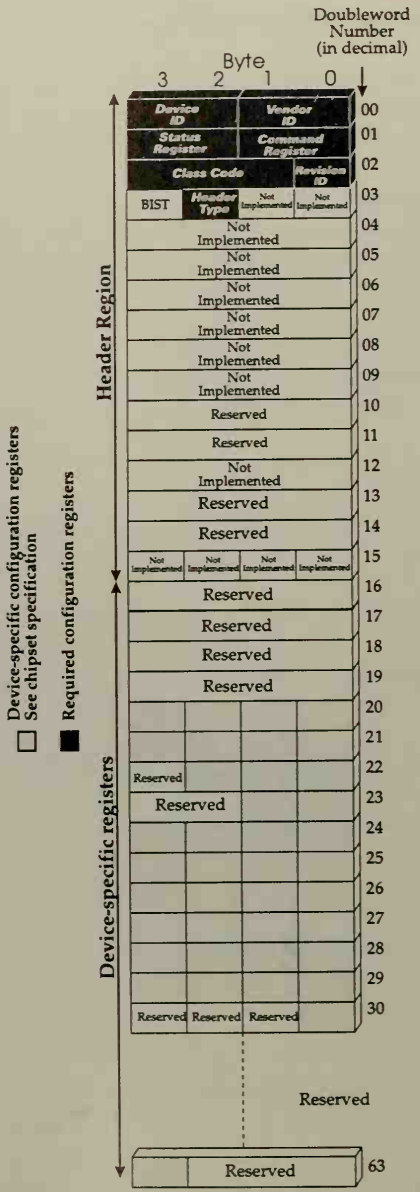

Figure 23-4. VL82C593's Configuration Registers