CPE 186 Computer Hardware Design

The 64-bit PCI Extension

Dr. Pang

64-bit Data Transfers and 64-bit Addressing: Separate Capabilities

- The **PCI** specification provides a mechanism that permits a 64-bit bus master to perform 64-bit data transfers with a 64-bit target. At the beginning of a transaction, the 64-bit bus master automatically senses If the responding target is a 64-bit or a 32-bit device. If it's a 64-bit device, up to eight bytes (a quadword) may be transferred during each data phase. Assuming a series of 0-wait state data phases, throughput of 264Mbytes/second can be achieved at a bus speed of 33MHz (8 bytes/transfer **x** 33 million transfers/second) and 528Mbytes/second at 66MHz. If the responding target is a 32-bit device, the bus master automatically senses this and steers all data to or from the target over the lower four data paths (AD[31:0]).

64-bit Data Transfers and 64-bit Addressing: Separate Capabilities

- The specification also defines 64-bit memory addressing capability. This capability is only used to address memory targets that reside above the 4GB address boundary.
- It is important to note that 64-bit addressing and 64-bit data transfer capability are two features, separate and distinct from each other. A device may support one, the other, both, or neither.

64-Bit Extension Signals

In order to support the 64-bit data transfer capability. the PCI bus implements an additional thirty-nine pins:

- REQ64# is asserted by a 64-bit bus master to indicate that it would like to perform 64-bit data transfers. REQ64# has the same timing and duration as the FRAME# signal.
- ACK64# is asserted by a target in response to REQ64# assertion by the master (if the target supports 64-bit data transfers). ACK64# has the same timing and duration as DEVSEL# (but ACK64# must not be asserted unless REQ64# is asserted by the initiator).
- AD[63:32] comprise the upper four address/data paths.
- C/BE#[7:4] comprise the upper four command/byte enable signals.
- PAR64 is the parity bit that provides even parity for the upper four AD paths and the upper four C/BE signal lines.

64-bit Cards in 32-bit Add-in Connectors

- A 64-bit card installed in a 32-bit expansion slot automatically only uses the lower half of the bus to perform transfers. This is true because the system board designer connects the REQ64# output pin and the ACK64# input pin on the connector to individual pullups on the system board and to nothing else.

64-bit Cards in 32-bit Add-in Connectors

- **When** a 64-bit bus master is installed in a 32-bit card slot and it initiates a transaction, its assertion of REQ64# is not visible to any of the targets. In addition, its ACK64# input is always sampled deasserted (because it's pulled up on the system board). This forces the bus master to use only the lower part of the bus during the transfer. Furthermore, if the target addressed in the transaction is a 64-bit target, it samples REQ64# deasserted (because it's pulled up on the system board), forcing it to only utilize the lower half of the bus during the transaction and to disable its ACK64# output.

Pullups Prevent 64-bit Extension from Floating When Not in Use

- If the 64-bit extension signals (AD[63:32]. C/BEX[7:4] and PAR64) are permitted to float when not in use, the CMOS input buffers on the card will oscillate and draw excessive current. In order to prevent the extension from floating when not in use, the system board designer is required to include pullup resistors on the extension signals to keep them from floating. Because these pullups are guaranteed to keep the extension from floating when not in use, 64-bit devices that are embedded on the system board and 64-bit cards installed in 64-bit PCI add-in connectors don't need to take any special action to keep the extension from floating when they are not using it.

The 64-bit extension is not in use under the following circumstances

1. The PCI bus is idle.

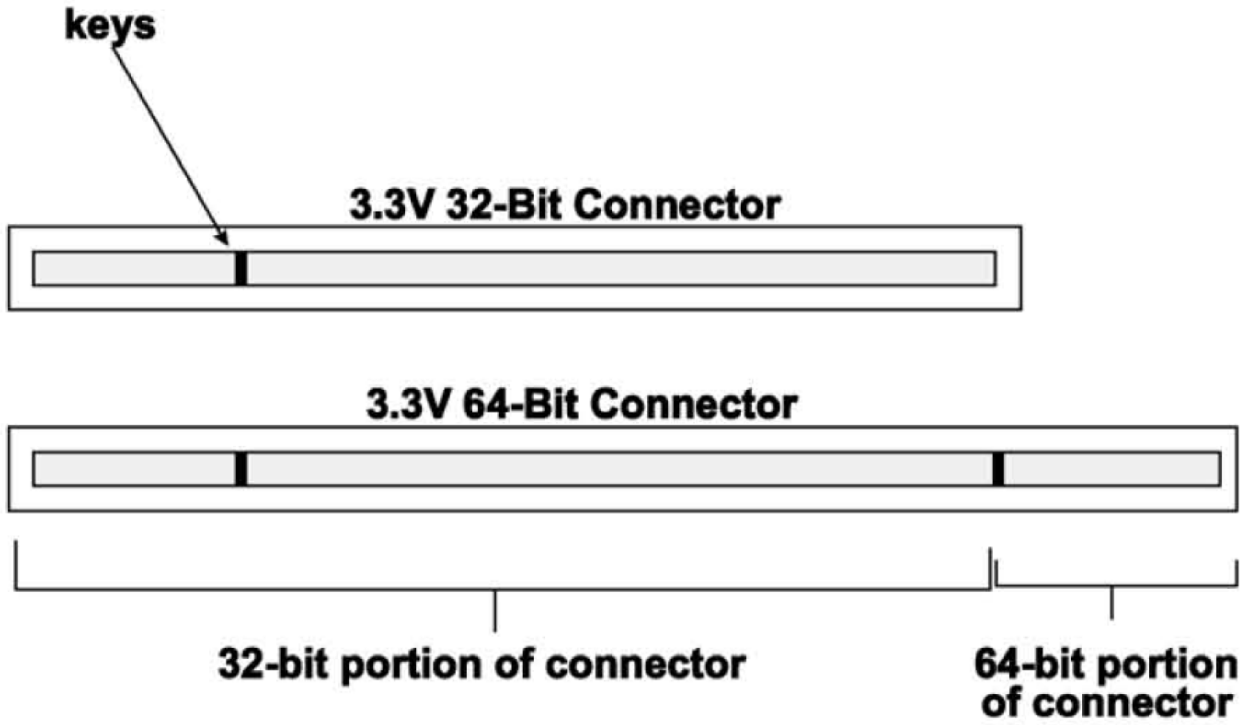
2. A 32-bit bus master is performing a transaction with a 32-bit target.

3. A 32-bit bus master is performing a transaction with a 64-bit target. Upon detecting REQ64# deasserted at the start of the transaction, the target will not use the upper half of the bus.

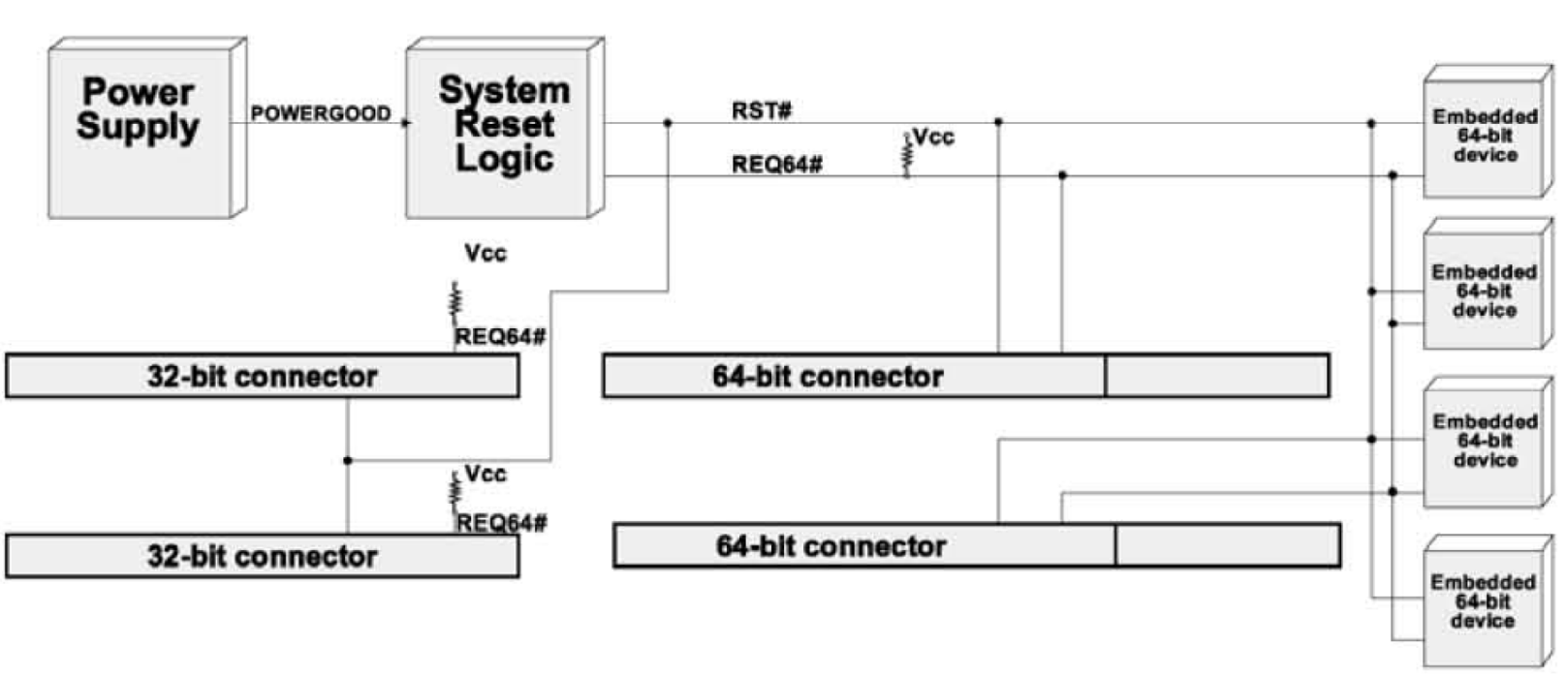
4. A 64-bit bus master addresses a target to perform 32-bit data transfers (REQ64# deasserted) and the target resides below the 4GB address boundary (the upper half of the bus is not used during the address phase and is also not used in the data phases). Whether the target is a 32-bit or a 64-bit target, the upper half of the bus isn't used during the data phases (because REQ64# is deasserted).

5. A 64-bit bus master attempts a 64-bit data transfer (REQ64# asserted) with a 32-bit memory target that resides below the 4GB boundary. In this case, the initiator only uses the lower half of the bus during the address phase (because it's only generating a 32-bit address). When it discovers that the currently-addressed target is a 32-bit target (ACK64# not asserted when DEVSEL**#** asserted), the initiator ceases to use the upper half of the bus during the data phases.

64- and 32- Bit Connectors



REQ64# Signal Routing



64-bit Data Transfer Capability

- The agreement to perform 64-bit transfers is established by a handshake between the initiator and the target.
- When the initiator supports 64-bit transfers and wishes to perform 64-bit transfers, it asserts REQ64# along with FRAME# during the address phase. If the currently-addressed target supports 64-bit data transfers, it replies with ACK64#.
- If either the master or the target, or both, do not support 64-bit data transfers, 32-bit data transfers are used instead.

64-bit Data Transfer Capability

During 64-bit transfers, all transfer timing during data phases is identical to that used during 32-bit data transfers. One to eight bytes may be transferred between the initiator and the target during each data phase and all combinations of byte enables are valid (including none asserted: an example is provided later in this chapter). The setting on the byte enable lines may be changed with each data phase. The following sections provide examples of:

- 64-bit initiator performing a transfer with a 64-bit target.
- 64-bit initiator performing a transfer with a 32-bit target.
- 32-bit initiator performing a transfer with a 64-bit target.

Only Memory Commands May Use 64-bit Transfers

- Only memory commands may utilize 64-bit data transfer capability. The specification provides the following arguments for not implementing support for 64- bit data transfers for the other types of commands:
- During the special cycle transaction, no target responds with DEVSEL#. ACK64#, therefore, is also not asserted.
- Configuration transactions do not require the level of throughput achievable with 64-bit data transfers and therefore do not justify the added complexity and cost necessary to support 64-bit data transfer capability.
- As with configuration transactions, IO transactions do not require a high level of throughput and therefore do not justify the added complexity and cost necessary to support 64-bit data transfer capability.
- By definition, the Interrupt Acknowledge command only performs a single data phase consisting of a one, two, three or four byte transfer.

Start Address Quadword-Aligned

- When a bus master starts a transfer and asserts REQ64#, the start address it issues is quadword-, not dword-, aligned (e.g.. address 00000100h, 00000108h, 00000110h, etc.). This means that AD[2] must be set to zero**.** AD[1:0] still convey the addressing sequence (for more information see "Memory Addressing”).

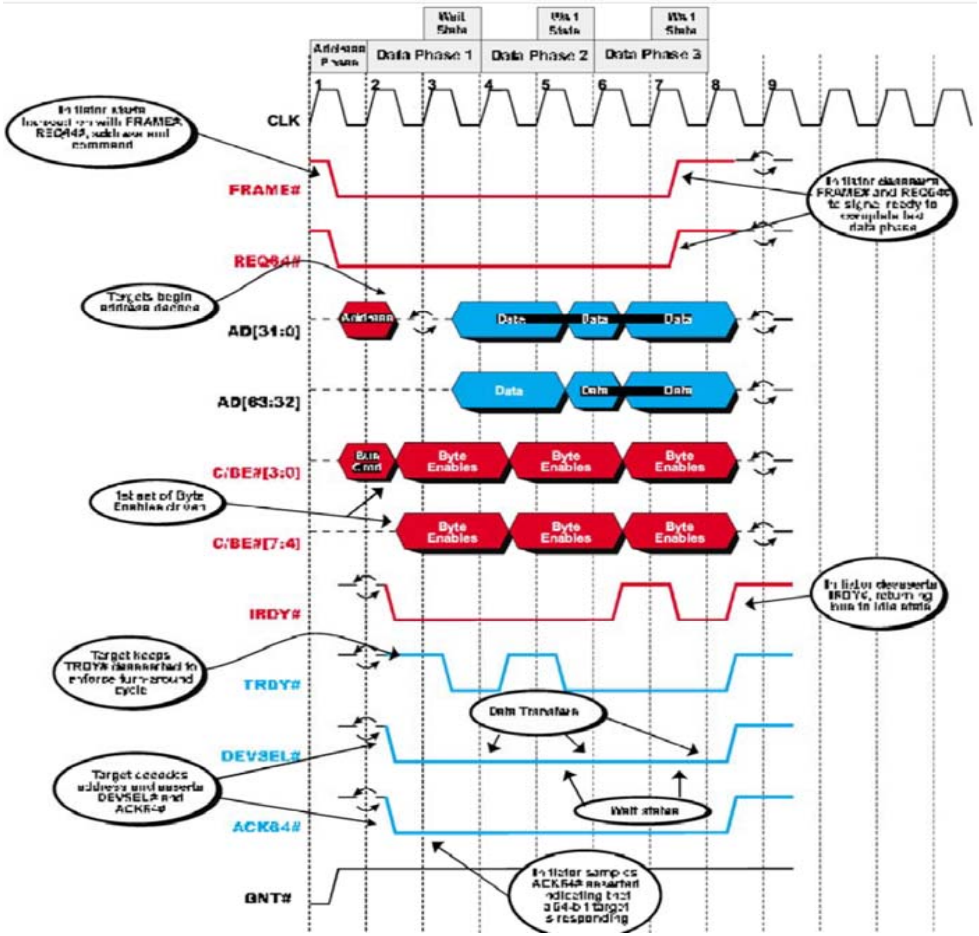
64-bit Target's Interpretation of Address

- Assuming that the target supports 64-bit data transfers (it asserts ACK64#), the target latches the start quadword-aligned address into its address counter (if It supports burst mode). If the addressing sequence indicated by the bus master on AD[1:0] is sequential (Linear) addressing, the target increments the address in its address counter by eight at the completion of each data phase to point to the next sequential quadword.
- If Cache Line Wrap addressing is indicated, it will increment through the cache line quadword-by-quadword and then wrap to the start of the line when it hits the end of the line. The target samples the eight byte enables. C/BE#[7:0], during each data phase to determine which of the eight bytes within the currently-addressed quadword is to be transferred (and therefore which of the eight data paths are to be used).

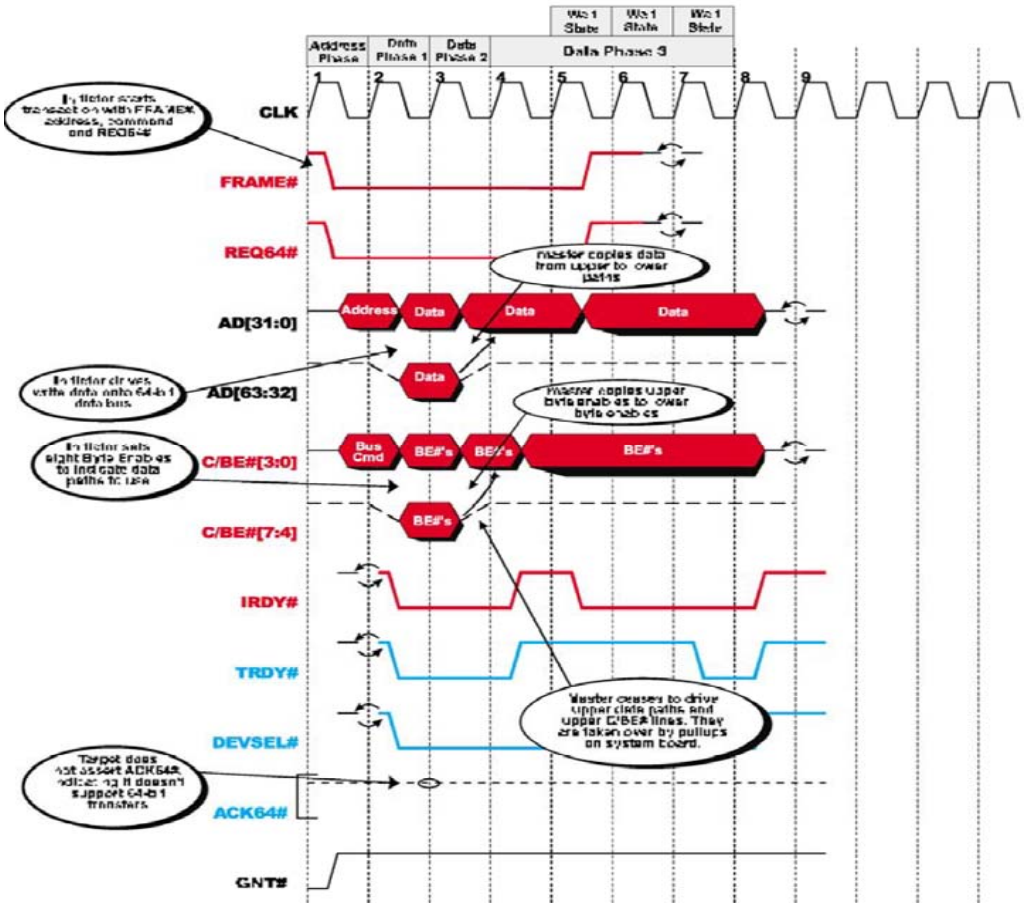
32-bit Target's Interpretation of Address

- If the target that responds to the transaction is a 32-bit target **(**ACK64# not asserted), it treats the start address as a dword-aligned address and latches it into its address counter. If the addressing sequence indicated by the bus master on AD[1:0] is sequential (Linear) addressing, the target increments the address in its address counter by four at the completion of each data phase to point to the next sequential dword. If Cache Line Wrap addressing is indicated, it will increment through the cache line dword-by-dword and then wrap to the start of the line when it hits the end of the line. The target samples the four lower byte enables, C/BE#[3:0], during each data phase to determine which of the four bytes within the currently-addressed dword is to be transferred and which of the four data paths (on the AD[31:0] portion of the bus) are to be used.

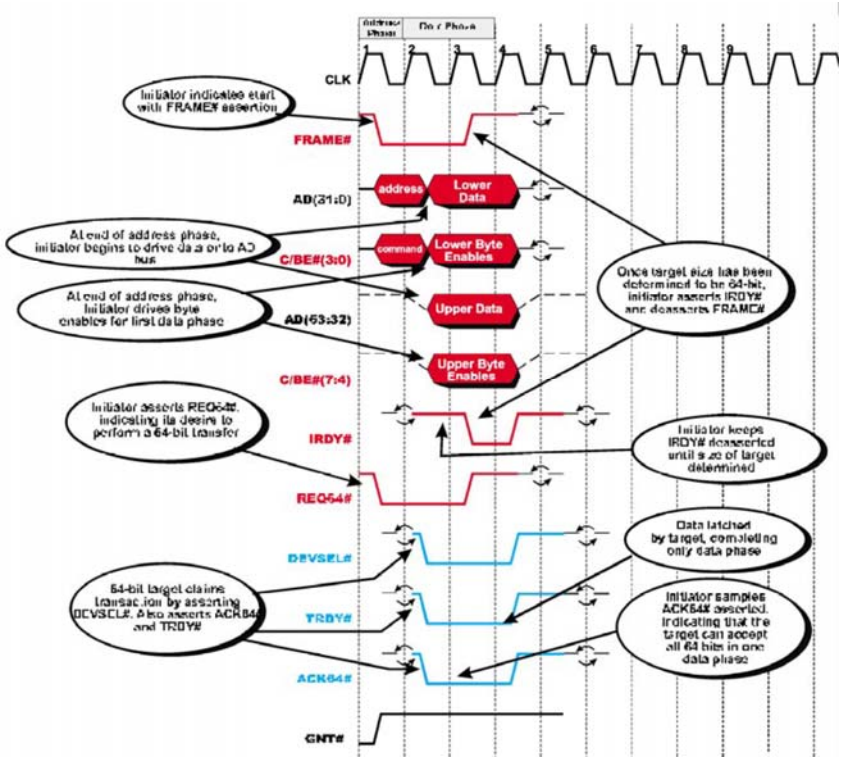
Transfer Between a 64-bit Initiator and 64-bit Target



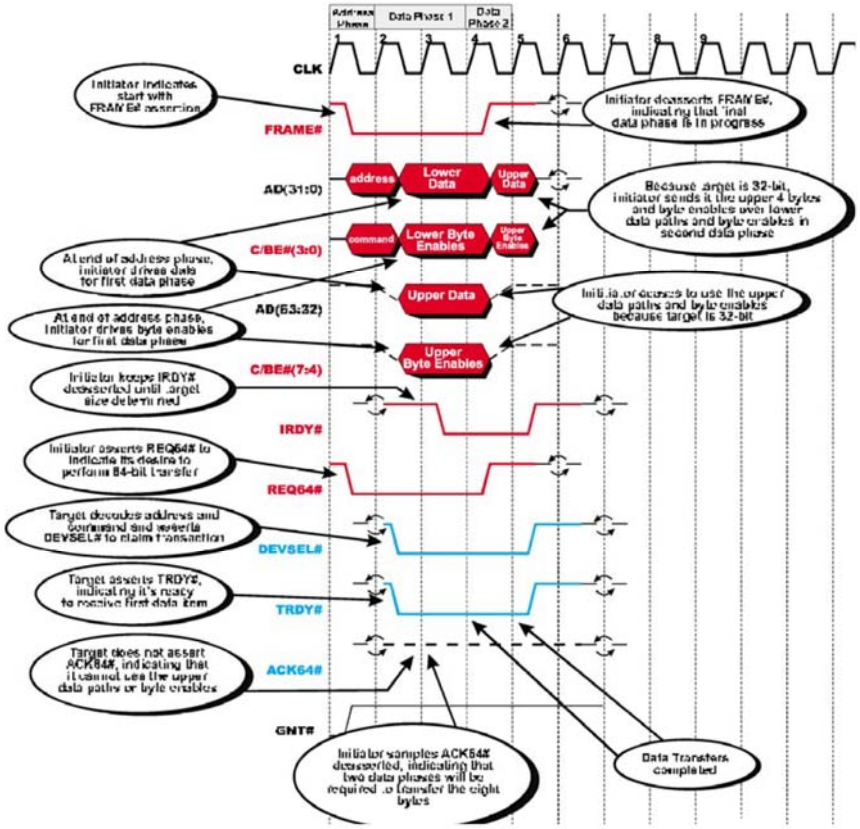
Transfer Between a 64-bit Initiator and a 32-bit Target



Single Data Phase 64-bit Transfer with a 64-bit Target



Dual-Data Phase 64-bit Transfer With a 32-bit Target



Disconnect on Initial Data Phase

- Assume that a master starts a multiple-data phase read transaction and asserts REQ64#. The quadword-aligned start address is 00000100 h and all eight byte enables are asserted during the first data phase. Also assume that the target is a 32-bit target and is not capable of handling a multiple-data phase transaction. This has the following effects:
- 1. The target does not assert **ACK64#**.
- 2. The target only transfers locations 00000100h through 00000103h during the first data phase and terminates the transaction by asserting **STOP#** along with **TRDY#.**
- 3. The master accepts the four bytes and ends the transaction.
- 4. The master re-arbitrates for the bus and then initiates the read again. It asserts **FRAME#** and **REQ64#** again. This time. the quadword-aligned start address is once again 00000100h, but only the upper four byte enables are asserted during the first data phase (because 00000100 through 00000103h have already been transferred).
- 5. The 32-bit target doesn't assert ACK64# and doesn't transfer any data (because the lower four byte enables are deasserted) and terminates the transaction with a Disconnect With Data Transfer **TRDY#** and **STOP#** asserted).
- 6. Recognizing that the upper four bytes in the quadword still haven't been transferred, the master re-arbitrates for the bus and tries it again. It will never be successful in transferring the upper four bytes.

The solution to this dilemma is as follows: when the master was disconnected after the first dword transfer, it should restart the Transaction as a 32-bit transfer (i.e.. do not assert REQ64#, and output the dword-aligned address of the next four bytes to be transferred: 00000104h).

64-bit Addressing

- **Used to Address Memory Above 4GB**

Bus masters are only permitted to use 64-bit addressing when communicating with memory that resides above the 4GB address boundary. Standard 32-bit addressing (in other words, a single address phase) must be used if the start address resides below this boundary (i.e.. the upper 32 bits of the address are all zero).

64-bit Addressing

- **Introduction**

Using the basic command set, the **PCI** address bus, AD[31:0], permits the initiator to address devices that reside within the first 4GB of address space (using a single address phase; any command that uses a single address phase is referred to as a **Single Address Command**, or **SAC**).

Without the addition of any signals, the **PCI** specification also provides support for addressing memory devices that reside above the 4GB boundary The **Dual Address Cycle**, or **DAC**, is used by an initiator to inform the community of targets that it is broadcasting a 64-blt memory address in two, back-to-back address phases. 64-bit addressing capability is not restricted to 64-bit initiators. Initiators fall into two categories:

- Those that are capable of generating only 32-bit addresses over AD[31:0] using a single address phase.
- Those that are capable of generating 32- and 64-bit addresses.

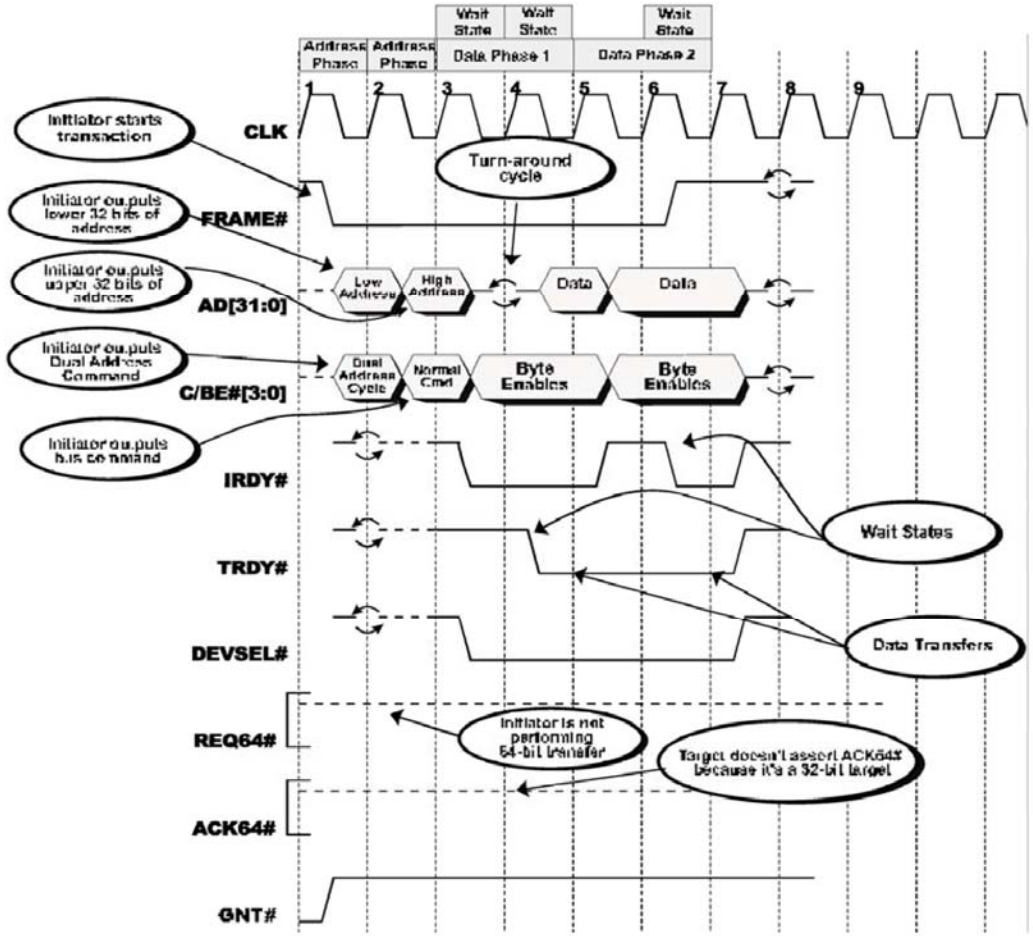
The sections that follow discuss the methods used by both 32-bit and 64-bit initiators in presenting a 64-bit address to the community of targets. Targets fall into two categories:

- Those that recognize the 64-bit addressing protocol (i.e., they have memory that resides above the 4GB address boundary).
- Those that only recognize the 32-bit addressing protocol (i.e.. they do not have any memory that resides above the 4GB address boundary).

64-bit Addressing Protocol

- **64-bit Addressing by 32-bit Initiator**
- Figure on next slide illustrates a 32-bit initiator performing a burst memory read access from above the 4GB boundary. The target in this example is a 32-bit target, although a 64-bit target would respond exactly the same way to a 64-bit address generated by a 32-bit initiator (because it would not see REQ64# asserted).

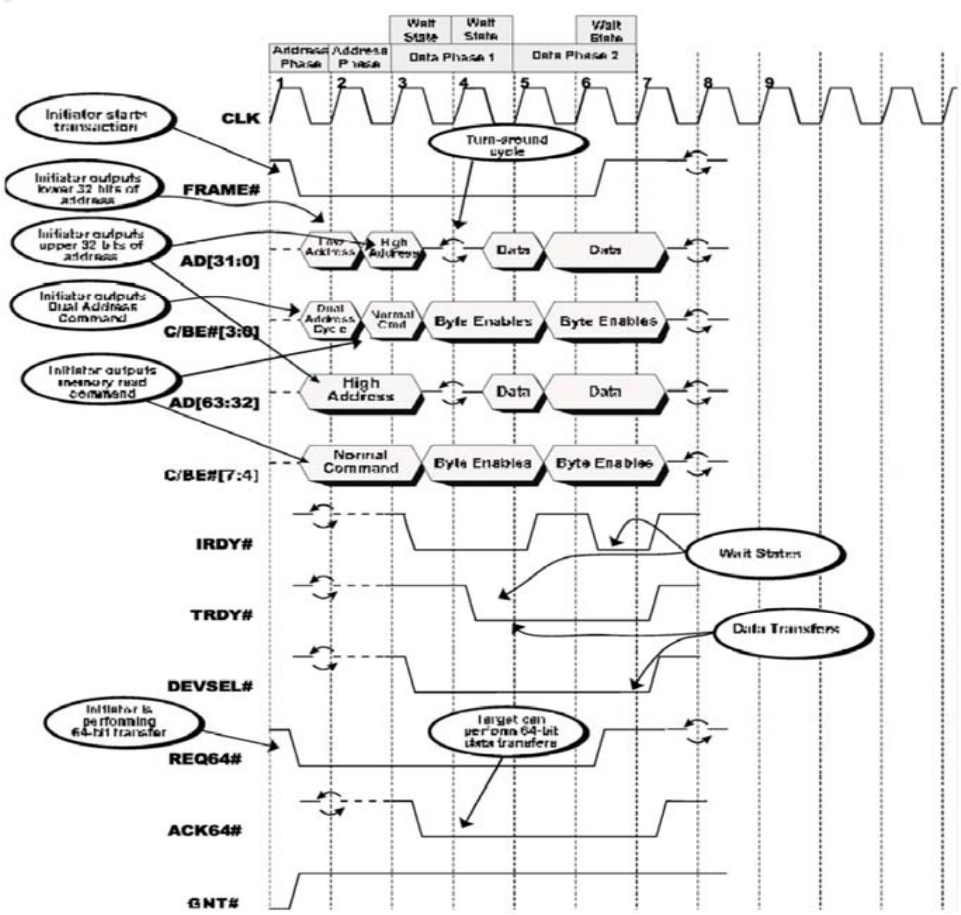
32-bit Initiator Reading From Address at or Above 4GB



64 bit Addressing by 64 bit Initiator

- Figure on next slide illustrates a 64-bit initiator performing a burst memory read access from an address above the 4GB boundary. In addition to using 64-bit addressing, the initiator asserts REQ64# to indicate that it wishes to perform 64 bit data transfers. The target of this transaction is a 64-bit target.
- If the target were a 32-bit device, it would not assert ACK64#, thereby indicating that it could not handle 64-bit data transfers. The initiator would then copy the byte enables from the upper lines, C/BE#[7:4], to the lower lines C/BE#[3:0], because the target would only have access to AD[31:0] and C/BE#[3:0].

64-bit Initiator Reading From Address Above 4GB With 64-Bit Data Transfers



32-bit Initiator Addressing Above 4GB

- An initiator that is only connected to AD[31:0] can communicate with memory above 4GB in two ways:
- The 64-bit memory **target** *can* alias all or some of its memory above 4GB into an address range in the lower 4GB. In other words, the configuration programmer could set up two address decoders (Base Address registers) for the same block of memory: one that responds to addresses above 4GB and one that recognizes addresses below 4GB. The master can then use SAC (single address phase command) memory commands to transfer data into/out of the memory above 4GB (using the window below 4GB). The master can use the DAC (Dual Address Cycle) to communicate directly with the memory above 4GB.
- The specification also contains the following statement: "Another alternative is for the master to support only 32-bit addressing and the device driver moves the data from 32-bit address space to the 64-bit address space." The 32-bit bus master transfers a data block into a memory buffer below the 4GB address boundary and generates an interrupt to inform its driver that the data is present in memory. The device driver then moves the data to memory above the 4GB address boundary.

Subtractive Decode Timing Affected

- If an expansion bus bridge located on the PCI bus employs subtractive decode (in other words, the ISA bridge), the expansion bus on the other side of the bridge either supports 64-bit addressing or it doesn't. As two examples, the ISA bus does not support 64-bit addressing, while the 64-bit Micro Channel does (but it's typically not a subtractive bridge).
- An ISA bridge that incorporates subtractive decode capability (virtually **all** implementations do) ignores any PCI transaction that uses 64-bit addressing (because there are no memory targets on the ISA bus that reside above 4GB).

Subtractive Decode Timing Affected

- A bridge to an expansion bus that supports 64-bit addressing and that incorporates a subtractive decoder operates in the following manner:
  - starting one clock after the end of the second address phase, it samples DEVSEL# on three successive rising edges of the clock to determine that the transaction isn't claimed by any PCI memory targets.
  - If none claim it (by asserting DEVSEL#), the bridge's subtractive decoder asserts DEVSEL# during the next clock to claim the transaction. It then initiates the memory transaction on the expansion bus to make it visible to all memory targets that reside above the 4GB address boundary on the expansion bus.

Subtractive Decode Timing Affected

During a single address phase transaction. starting one clock after the end of the one and only address phase, a subtractive decoder samples DEVSEL# on three successive rising-edges of the clock to determine that the transaction isn't claimed by any **PCI** memory targets.

Master Abort Timing Affected

- Normally, a bus master samples DEVSEL# at the end of the second. third, fourth and fifth clocks of the transaction and then Master Aborts if DEVSEL# was not sampled asserted. When the master is using 64-bit addressing, it samples DEVSEL# at the end of the third, fourth, fifth and sixth clocks of the transaction and then aborts if DEVSEL# was not sampled asserted.

Address Stepping

- A bus master that uses stepping to gradually place the full address on the bus keeps FRAME# deasserted until the full address is present and stable on the bus. When the targets sample FRAME# asserted, the address is decoded.
- The specification says that stepping cannot be used for 64-bit addressing. Although the master could keep FRAME# deasserted for several clocks while it gradually built the lower 32 bits of the address on AD[31:0]. 32-bit memory targets then expect the upper 32 bits of the address to be present and stable on AD[31:0] on the next rising-edge of the clock. The upper 32 bits of the address therefore cannot be stepped onto AD[31:0]. It must be presented in one clock.

FRAME# Timing in Single Data Phase Transaction

- If the initiator were using 64-bit addressing to perform a single data phase transaction. FRAME# is asserted at the start of the first address phase and must be kept asserted until **IRDY#** is asserted in the data phase. It cannot be removed during the second address phase.

64-bit Parity

- **Address Phase Parity**
- **PAR64 Not Used for Single Address Phase**

When the initiator is not using 64-bit addressing, there is a single address phase and a 32-bit address is generated by the initiator. In this case. address phase parity is supplied solely over the PAR signal. The 64-bit extension, consisting of AD[63:32], PAR64, and C/BE#[7:4], is not in use during the address phase of the transaction. Please note that the 2.2 spec describes the use of the upper part of the bus during single address phase transactions differently than the 2.1 spec and these differences can lead to confusion. Be sure and read both descriptions carefully.

- **PAR64 Not Used for Dual-Address Phases by 32-bit Master**

When a 32-bit bus master is performing 64-bit addressing, it is only using AD[31:0] and C/BE#[3:0]. The 64-bit extension is not in use. The PAR64 signal is therefore not used to supply parity on the upper half of the bus.

- **PAR64 Used for DAC by Master When Requesting 64 bit Transfers**

When a 64-bit bus master is performing 64-bit addressing and has asserted REQ64#, it is required to drive the upper 32 bits of the address onto AD[63:32] and the memory command onto C/BE#[7:4] during the first address phase. It is also required to supply even parity for **this** information on the PAR64 signal on the next rising-edge of the clock. During the second address phase. the initiator is required to continue driving the same information on AD[63:32] and C/BE#[7:4]. The initiator must once again supply even parity on PAR64 on the rising-edge of the clock that follows completion of the second address phase. A 64-bit target may be designed to latch the entire address and command and begin the decode at the end of either the first or the second address phase. The target then latches the state of PAR64 on the rising-edge of the clock that follows.

Data Phase Parity

- Parity works the same for 32- and 64-bit transfers.
- In a 64-bit implementation, an additional parity signal PAR64 is added. It's timing and function are identical to that of the PAR signal. PAR64 must be implemented by all 64-bit agents.

Data Phase Parity

- PAR64 must be set to the appropriate state (to force even parity) one clock after the completion of each address phase and one clock after the data is presented **(IRDY#** asserted on a write, or **TRDY#** asserted on a read) in each data phase.

Usage of PAR64 is qualified by the assertion of **REQ64#** (indicating that the initiator wants to perform 64-bit data transfers) and **ACK64#** (the target supports 64-bit data transfers). If either REQ64# or ACK64t is deasserted. only the lower half of the bus is in use and PAR64 is not used.