CPE 186 Computer Hardware Design

Error Detection and Handling

Dr. Pang

Error Detection and Handling

- The PCI bus architecture provides two error reporting mechanisms: one for reporting data parity errors and the other for reporting more serious system errors.

Introduction to PCI Parity

- The PCI bus is parity-protected during both the address and data phases of a transaction. A single parity bit, PAR, protects AD[31:0] and C/BE#[3:0]. If a 64-bit data transfer is in progress, an additional parity bit, PAR64, protects AD[63:32] and C/BE#[7:4]. 64-bit parity has the same timing as 32-bit parity.

Introduction to PCI Parity

The PCI device driving the AD bus during the address phase or any data phase of a transaction must always drive a full 32-bit pattern onto the AD bus (because parity is always based on the full content of the AD bus and the C/BE bus).

Introduction to PCI Parity

- The PCI device driving the AD bus during the address phase or any data phase of a transaction is responsible for calculating and supplying the parity bit for the phase.
- The parity bit must be driven one clock after the address or data is first driven onto the bus (when TIDY# is asserted during a read data phase to indicate the presence of the read data on the bus, or when IRDY# is asserted during a write data phase to indicate the presence of the write data on the bus) and must continue to be driven until one clock after the data phase completes.
- Even parity is used (i.e.. there must be an even number of ones in the overall 37-bit pattern consisting of AD[31:0], C/BE#[3:0] and PAR).

Introduction to PCI Parity

During the clock cycle immediately following the conclusion of the address phase or any data phase of a transaction, the PCI agent receiving the address or data computes expected parity based on the information latched from AD[31:0] and C/BE#[3:0]. The agent supplying the parity bit must present it:

- on the rising-edge of the clock that immediately follows the conclusion of the address phase.
- one clock after presentation of data (IRDY# assertion on a write or TRDY# assertion on a read) during a data phase.

Introduction to PCI Parity

The device(s) receiving the address or data expect the parity to be present and stable at that point.

- The computed parity bit is then compared to the parity bit actually received on PAR to determine if address or data corruption has occurred.
- If the parity is correct. no action is taken. If the parity is incorrect, the error must be reported.

Introduction to PCI Parity

During a read transaction where the initiator is inserting wait states, the initiator can optionally be designed with a parity checker that:

- samples the target's data (when TRDY# Is sampled asserted),
- calculates expected parity,
- samples actual parity (on the clock edge after TRDY# sampled asserted)
- and asserts PERR# (if the parity is incorrect).

In this case, the initiator must keep PERR# asserted until one dock after the completion of the data phase.

Introduction to PCI Parity

The same is true in a write transaction. If the target is inserting wait states by delaying assertion of TRDY#, the target's parity checker can optionally be designed to:

- sample the initiator's data (when IRDY# Is sampled asserted),
- calculate expected parity,
- sample actual parity (on the clock edge after lRDY# sampled asserted)
- and assert PERR# (if the parity is incorrect).

In this case, the target must keep PERR# asserted until two clocks after the completion of the data phase.

PERR# Signal

PERR# is a sustained tri-state signal used to signal the detection of a parity error related to a data phase. There is one exception to this rule: a parity error detected on a data phase during a Special Cycle is reported using SERR# rather than PERR#.

PERR# is implemented as an output on targets and as an input/output on masters. Although PERR# is bussed to all PCI devices. it is guaranteed to be driven by only one device at a time (the initiator on a read, or the target on a write.)

Data Parity

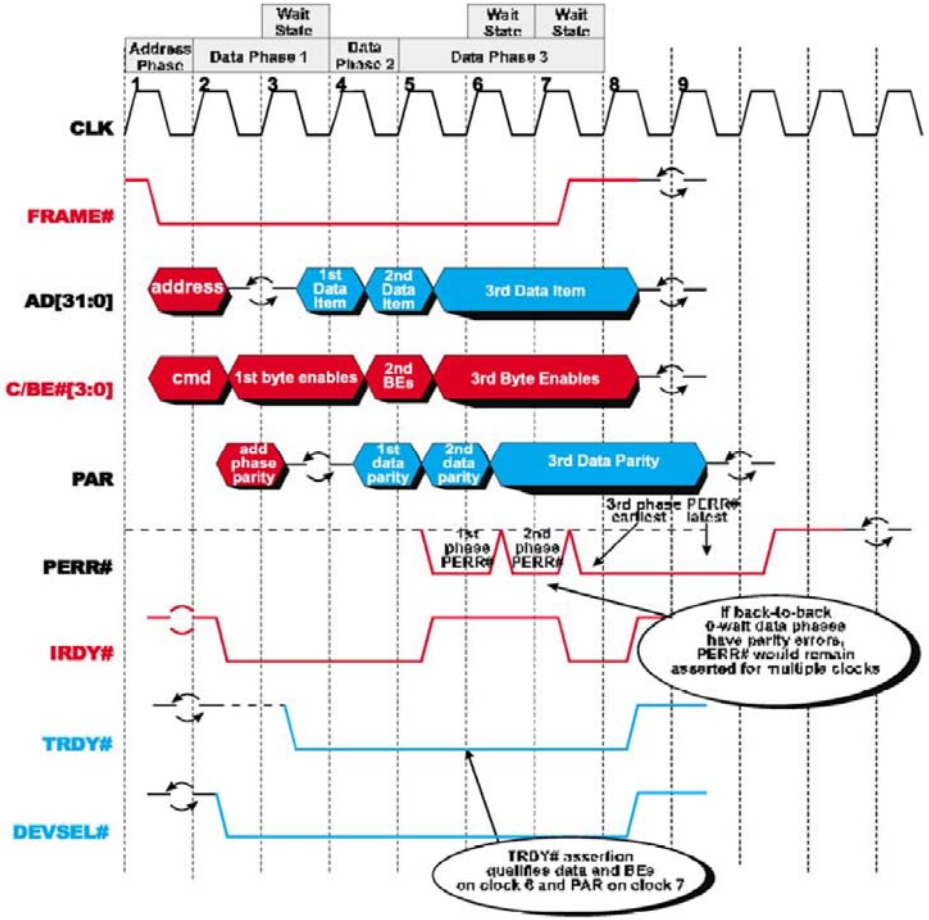
- **Data Parity Generation and Checking on Read**
- **Introduction**

During each data phase of a read transaction, the target drives data onto the AD bus. It is therefore the target's responsibility to supply correct parity to the initiator on the PAR signal starting one clock after the assertion of TRDY#.

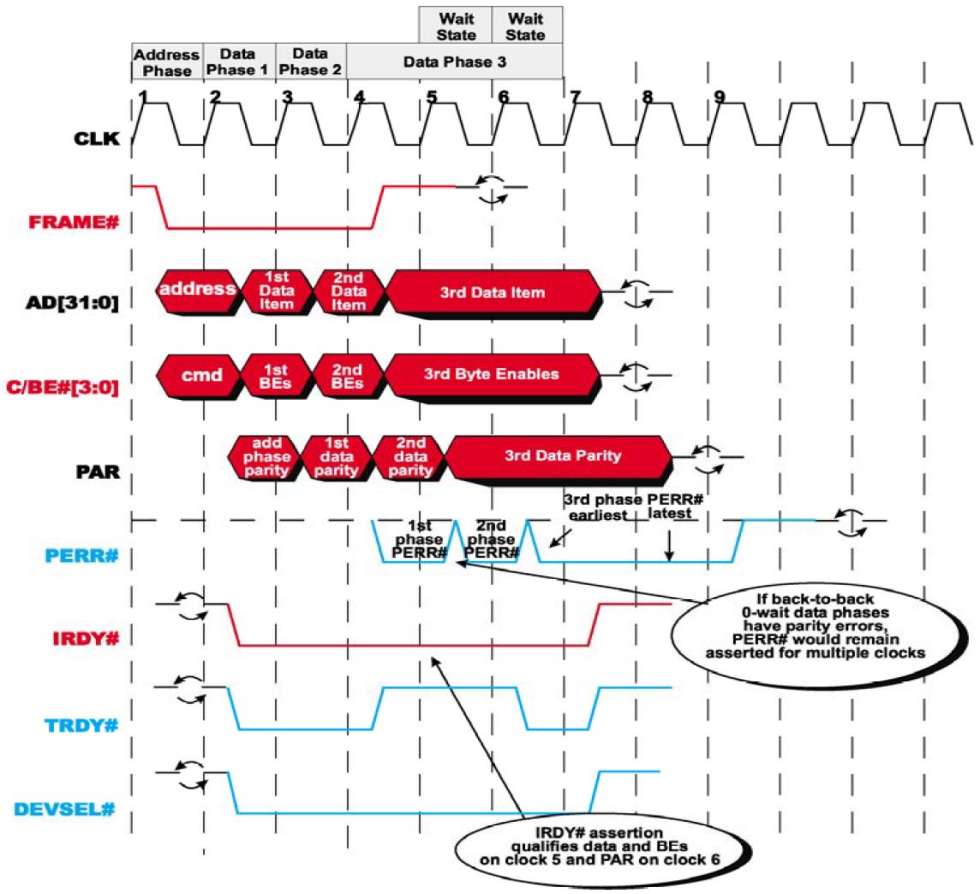
At the conclusion of each data phase. it is the initiator's responsibility to latch the contents of AD[31:0] and C/BE#[3:0] and to calculate the expected parity during the clock cycle immediately following the conclusion of the data phase. The initiator then latches the parity bit supplied by the target from the PAR signal on the next rising-edge of the clock and compares computed vs. actual parity. If a miscompare occurs, the initiator then asserts PERR# during the next clock (if it's enabled to do so by a one in the Parity Error Response bit in its Command register). The assertion of PERR# lags the conclusion of each data phase by two PCI clock cycles.

The platform design (in other words, the chipset) may or may not include logic that monitors PERR# during a read and takes some system-specific action.

Parity on Read Transaction



Parity on Write Transaction



Data Parity Reporting

- Upon detection of a data phase parity error, the device that checked the parity is responsible for asserting the Detected Parity Error bit in its PCI configuration Status register. It also asserts PERR# if the Parity Error Response bit in its PCI configuration Command register is set to one. Only two categories of devices are excluded from the requirement to implement the PERR# signal and the Parity Error Response bit.
- If a data phase parity error is detected, PERR# must be asserted (at the latest) in the second clock after completion of the data phase (i.e.. one clock after PAR is latched). Once PERR# is asserted, it must not be deasserted until during the third clock after the data phase completes. Figure 13-1 on page 204 and Figure 13-2 on page 208 both illustrate examples where the devices receiving the data checked parity and asserted PERR# before the completion of the data phase.

Master Can Choose Not to Assert PERR#

- The master of a transaction that has a data phase parity error could choose to assert PERR#. Rather, it could choose to attempt error recovery on its own or to invoke its driver (by generating an interrupt) so that it may attempt error recovery.

Parity Error During Read

- During a read transaction, the target sources the data and the parity. The initiator receives the data and parity and checks the parity for correctness. If the data is incorrect, the initiator must set the Detected Parity Error bit in its PCI configuration Status register (irrespective of the state of its Parity Error Response bit). Assuming that the initiator's Parity Error Response bit is set to one, the initiator asserts PERR# in the second clock following completion of the data phase and sets the Master Data Parity Error bit in its configuration Status register.

Parity Error During Read

- Whether or not the bus master continues the transaction or terminates it is master design-dependent. The specification recommends that the transaction be continued to completion. In addition to the assertion of PERR#, the bus master is required to report the parity error to the system software. The specification recommends utilization of an interrupt or the setting of a bit in a device-specific status register that is polled by the device driver. Alternately, the designer can assert SERR#, but this approach should not be used lightly. It will more than likely result in a system shutdown.

Important Note Regarding Chipsets That Monitor PERR#

- In many platform designs (i.e.. chipset designs), the chip set logic converts any assertion of PERR# by anyone into SERR#. This means that if either the master (on a read) or the target (on a write) asserts PERR#, the chipset may very well either assert SERR# or just take the same action that it normally does when it detects SERR# asserted. Typically, this results in the generation of a fatal interrupt to the processor (such as NMI or Machine Check).

Address Parity Generation/Checking

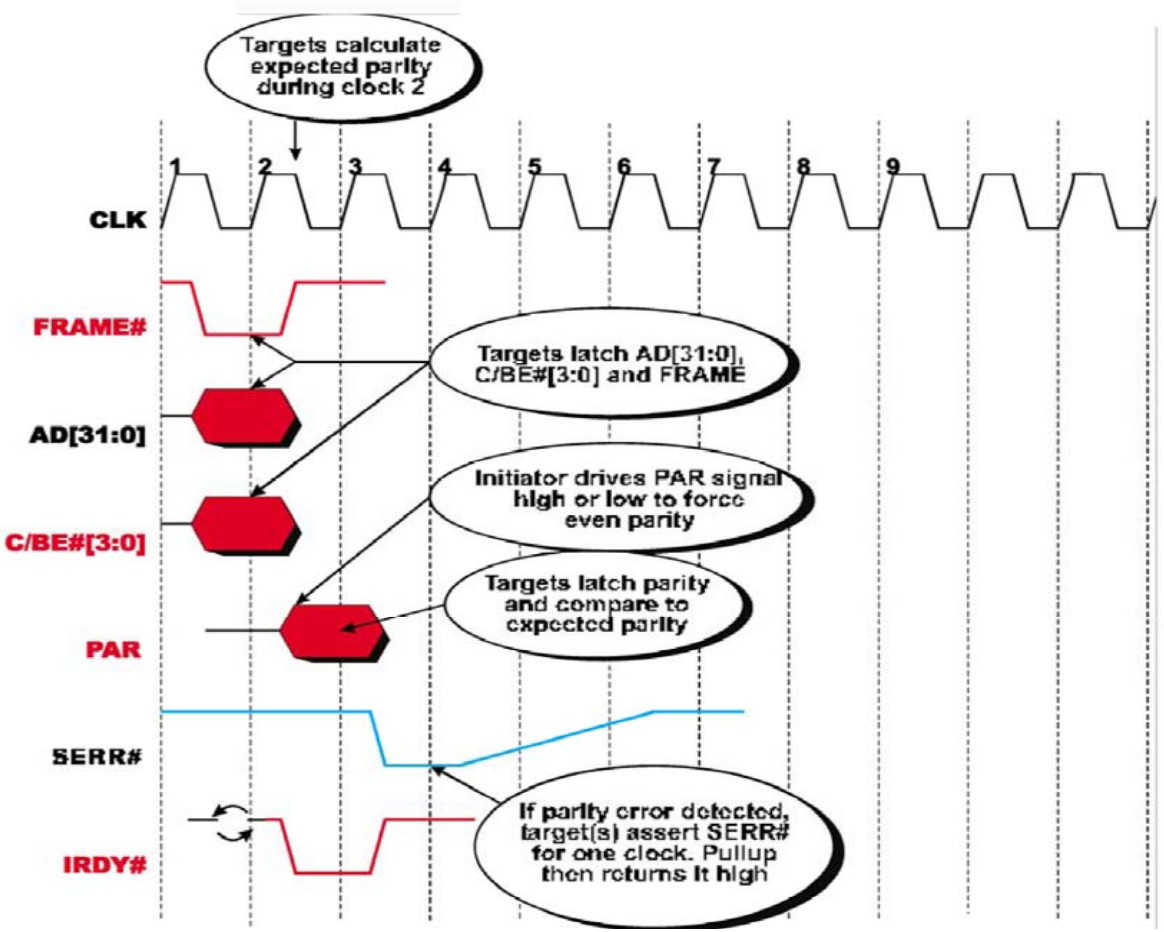


Figure 13-3: PCI Device's Configuration Command Register

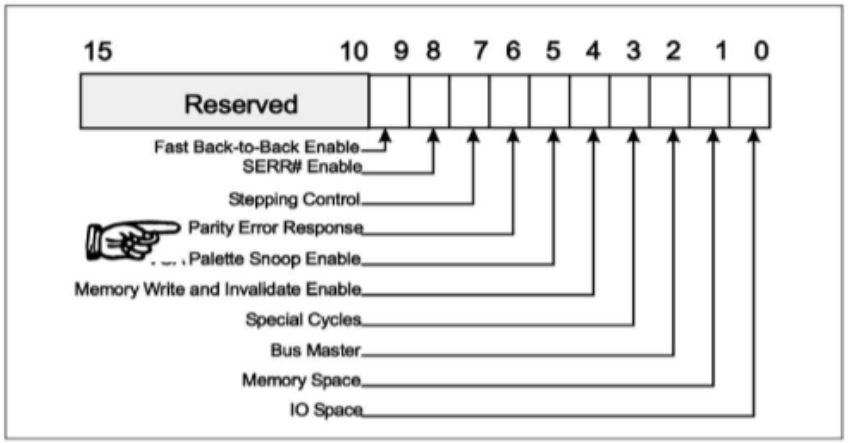
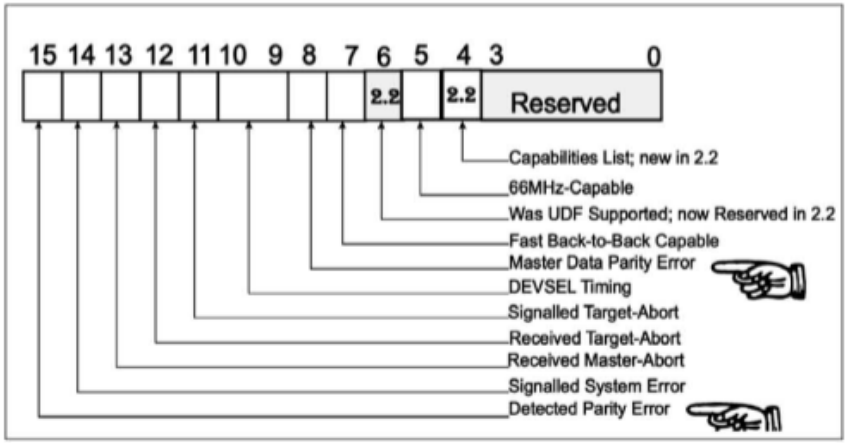


Figure 13-4: PCI Device's Configuration Status Register



Data Parity Error Recovery

The PCI specification permits recovery from data phase parity errors but does not require it.

Devices Excluded from PERR# Requirement

- The PCI specification excludes two types of devices from the requirement to implement the PERR# signal and the Parity Error Response bit in the configuration Command register. The following two sections describe these device categories.
- In a PC, the chipset is embedded on the system board and typically consists of the host/PCI bridge and the PCl/expansion bus bridge. These two entities can be designed without the PERR# pin.

SERR# Signal

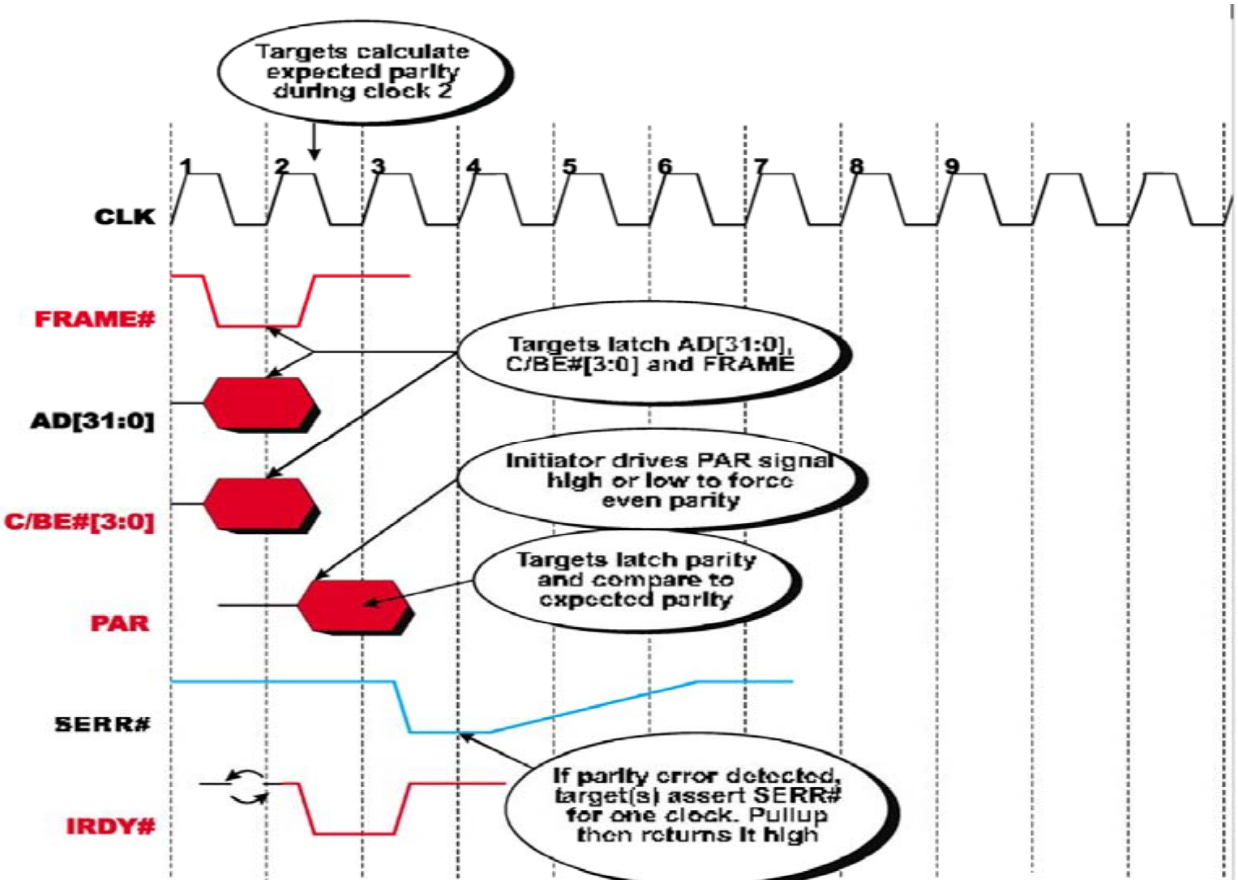
- SERR# is a required output from all PCI devices and is an input to the platform support logic (i.e.. the chipset). A PCI device is not permitted to assert SERR# unless the SERR# Enable bit in its configuration Command register is set to one. SERR# is implemented as an open-drain, shared signal because multiple PCI devices may assert SERR# simultaneously The system designer must provide a pull-up resistor on the SERR# signal line.
- When asserted, SERR# is asserted starting on the rising-edge of the PCI clock and is asserted for one clock and then tri-stated. It may be asserted at any time (i.e. its assertion is not tied to any type or phase of a PCI transaction). The specification suggests that SERR# should be asserted as quickly as possible (within two clocks of detecting an error condition is recommended).

SERR# Signal

The SERR# signal is used to signal the following types of conditions:

- parity error on the address phase of a transaction
- parity error on the data phase of a Special Cycle transaction.
- error when attempting a memory write to deliver a message signaled interrupt
- serious problems other than parity detected by a PCI device.
- critical system failures detected by system board logic.

Address Parity Generation/Checking

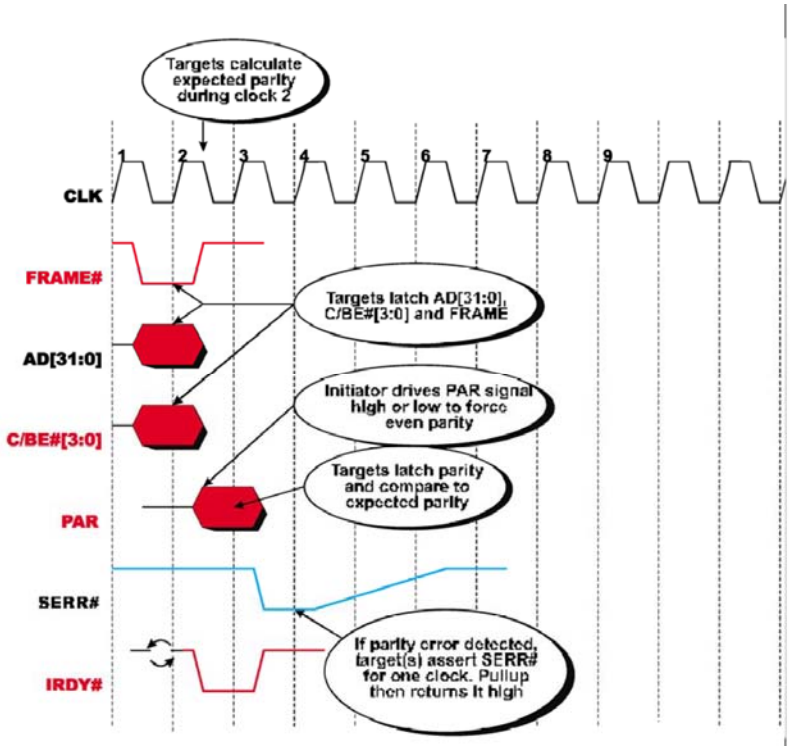


SERR# Signal

- In addition to the assertion of SERR#, the target device that is apparently addressed in a corrupted address phase can react in one of the following ways:
- METHOD1 . assert DEVSEL# and complete the transaction normally.
- METHOD 2. assert DEVSEL# and terminate the transaction with a Target Abort.
- METHOD 3. not assert DEVSEL# and let the master time out and execute a Master Abort.

- The target is not permitted to terminate the transaction with a Retry or a Disconnect.

Address Parity Generation/Checking



Address Phase Parity Error Reporting

- Each target that detects an address phase parity miscompare must assert SERR# for one *clock* (if the device's SERR# Enable bit is set to one in its configuration Command register).
- Any devices that assert SERR# must also set the Signaled System Error bit in their configuration Status registers to one.
- The device's Detected Parity Error bit must also be set to one.

Figure 13-3: PCI Device's Configuration Command Register

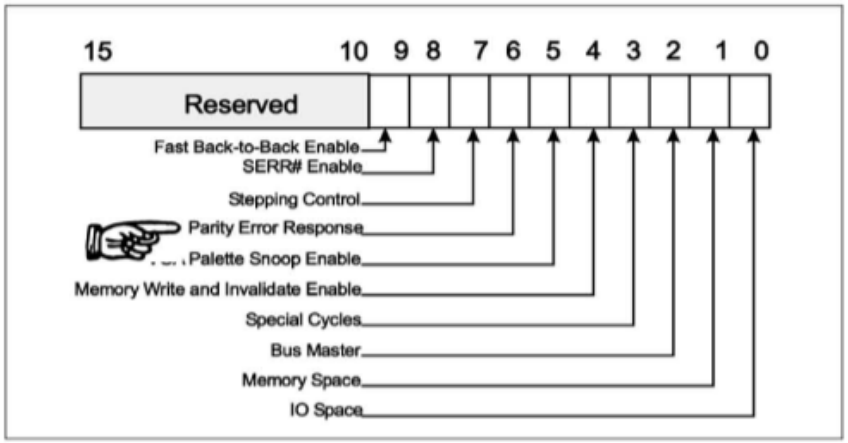
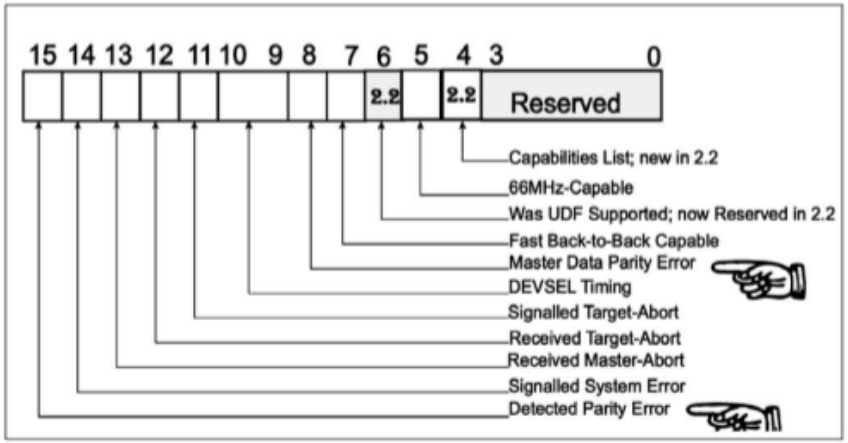


Figure 13-4: PCI Device's Configuration Status Register



System Errors

In all cases of SERR# assertion, a device is not enabled to assert SERR# unless the SERR# Enable bit in its configuration Command register is set to one.

The various causes of SERR# assertion are shown below:

- **Address Phase Parity Error**
- **Data Parity Error During Special Cycle**
- **Master of MSI (message signaled interrupt) Receives an Error**

System Errors

- **Target Abort Detection**

A master that doesn't have a mechanism (such as an interrupt line) to report receipt of a Target Abort to system software may assert SERR# to alert system software. The master must also set the Received Target Abort bit in its Status register.

- **Other Possible Causes of System Error**

When any PCI device suffers a serious failure that impairs its ability to operate correctly, it may assert SERR# (if its SERR# Enable bit is set to one) to inform system software of the failure. It must also set the Signaled System Error status bit in its configuration Status register.