R4000 Data Integrity

Prepared by: Michael Ngo

MIPS Technologies, Inc.
A Technology Company of Silicon Graphics, Inc.
2011 N. Shoreline Blvd.
P.O. Box 7311
Mountain View, CA 94039-7311
Phone: (415) 960-1980

R4000 Data Integrity

What is R4000 data integrity?

- Detection of data corrupted due to soft errors caused by system noise, power surges, and alpha particles

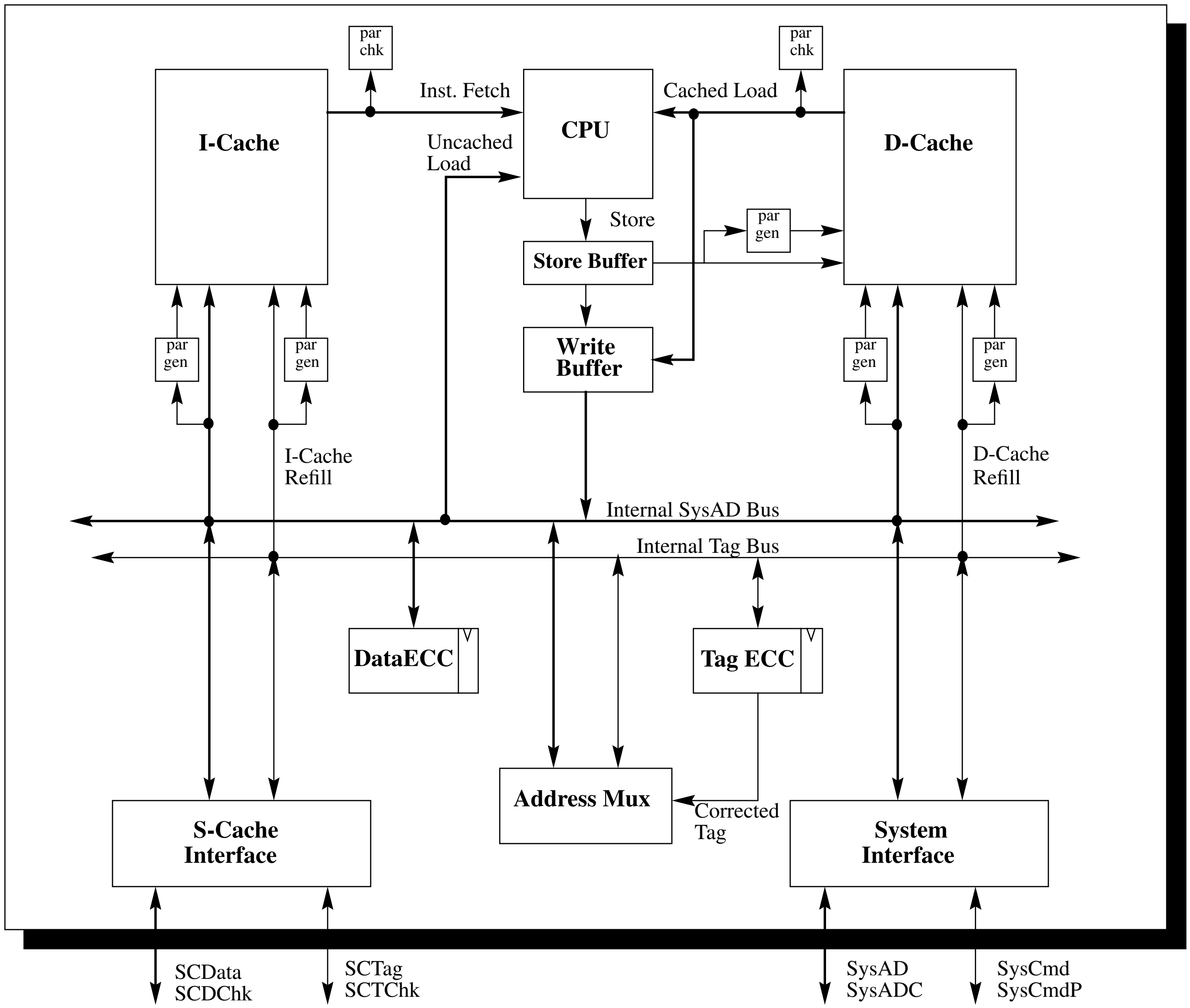
Why worry about data integrity?

- Large and highly integrated on-chip static RAM
- Increased system reliability
- Fault tolerant system design

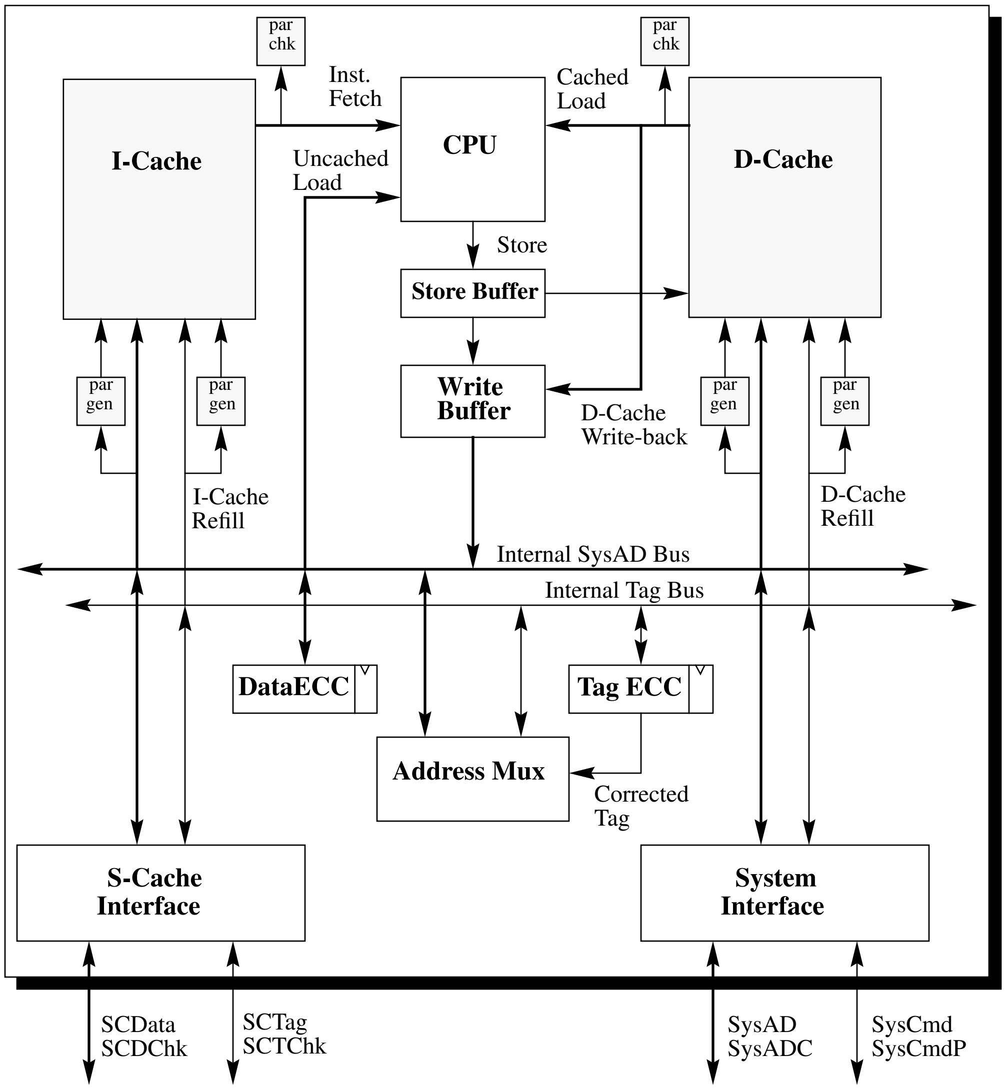
Type of error checking in R4000:

- Parity checking on the primary instruction and data caches
- ECC checking on the secondary cache
- Parity or ECC checking on the system interface bus
- Master-checker

Internal Cache Memory and Datapath

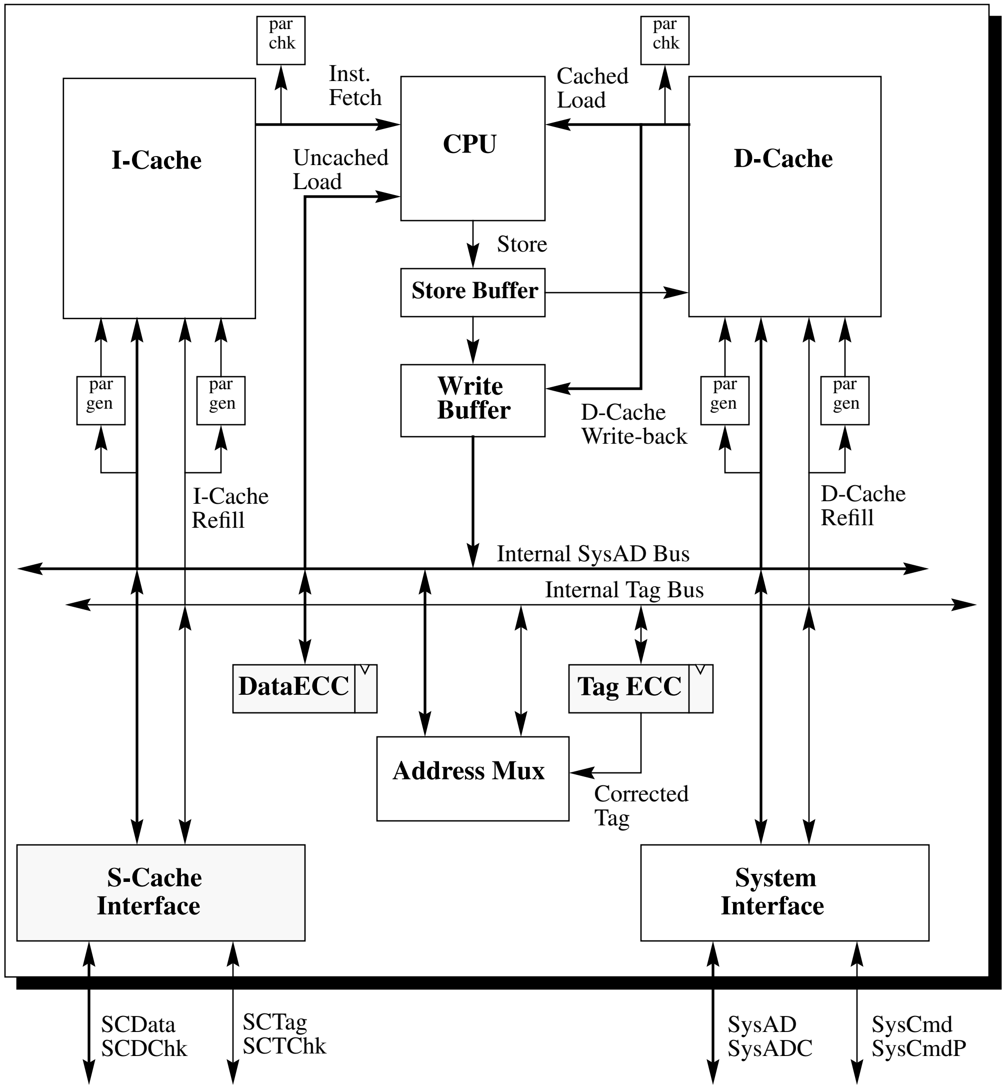


Error Checking in Primary Instruction/Data Caches



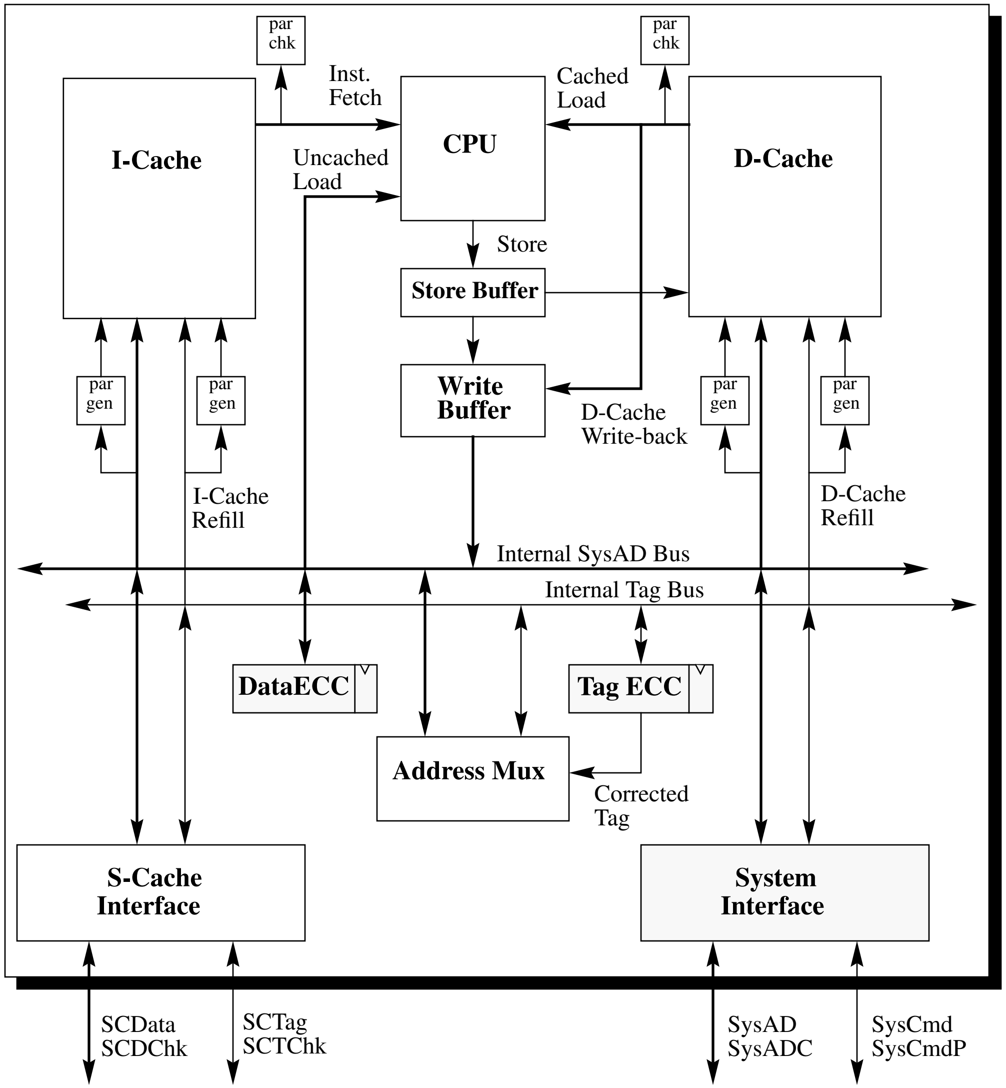
- Byte parity protected on Data Array
- Word parity protected on Tag Array
- Parity is generated on primary cache write
- Parity checking is performed on every primary cache access (instruction fetch, data fetch, or cache write-back)

Error Checking in Secondary Cache



- ECC protected on Data Array
- ECC protected on Tag Array
- ECC check bits for the Data and Tag portions are generated on secondary cache update
- ECC checking is performed on every secondary cache access (primary instruction/data cache refill or secondary cache write-back)

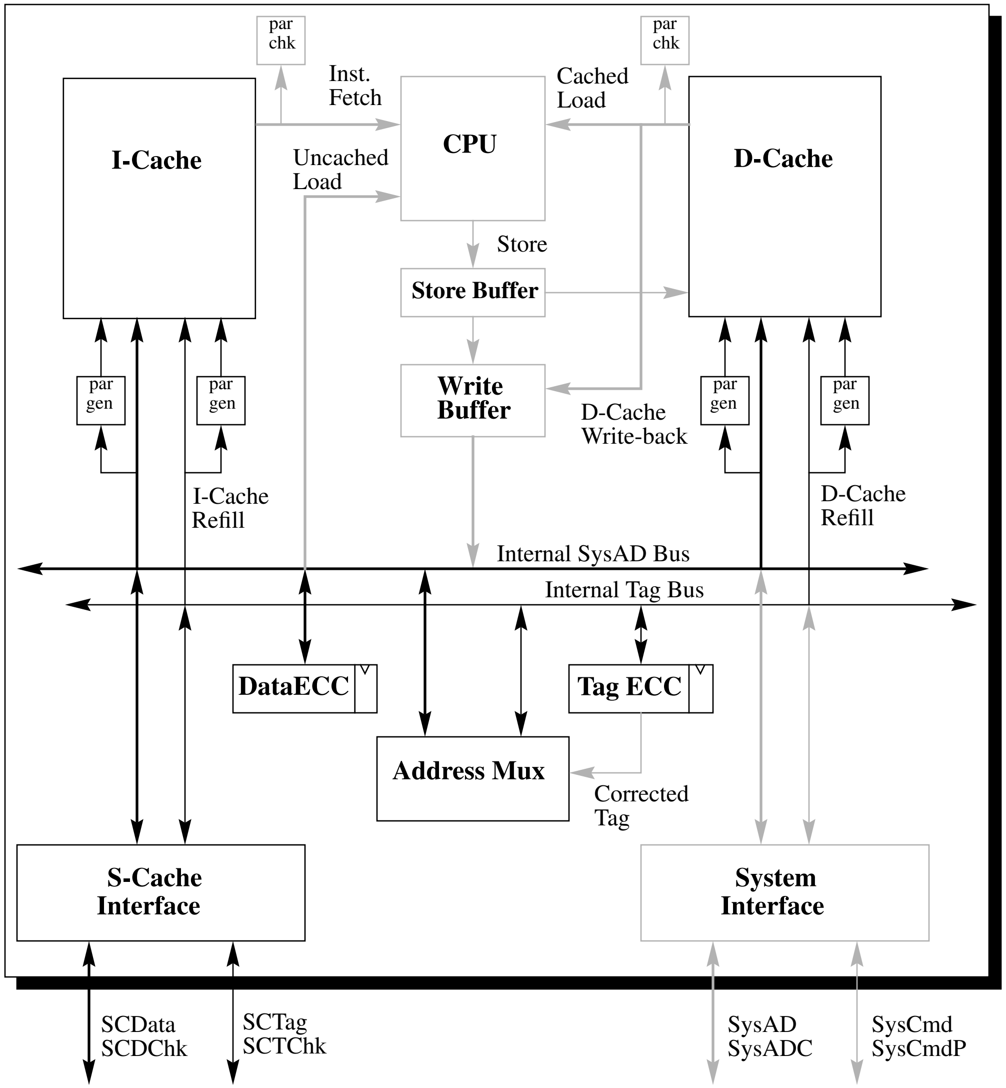
Error Checking on System Interface



- **Parity or ECC protected on SysAD bus**[1]
- **Parity protected on SysCmd bus**
- **For out-going transactions:**
  - Parity or ECC check bits are generated for the address and parity is generated for the command
  - Data's parity or ECC check bits come from primary cache, secondary cache or generated for the processor store data[2]
- **For in-coming transactions:**
  - Address and command portions are not checked
  - Parity or ECC checking is performed for in-coming data[3]

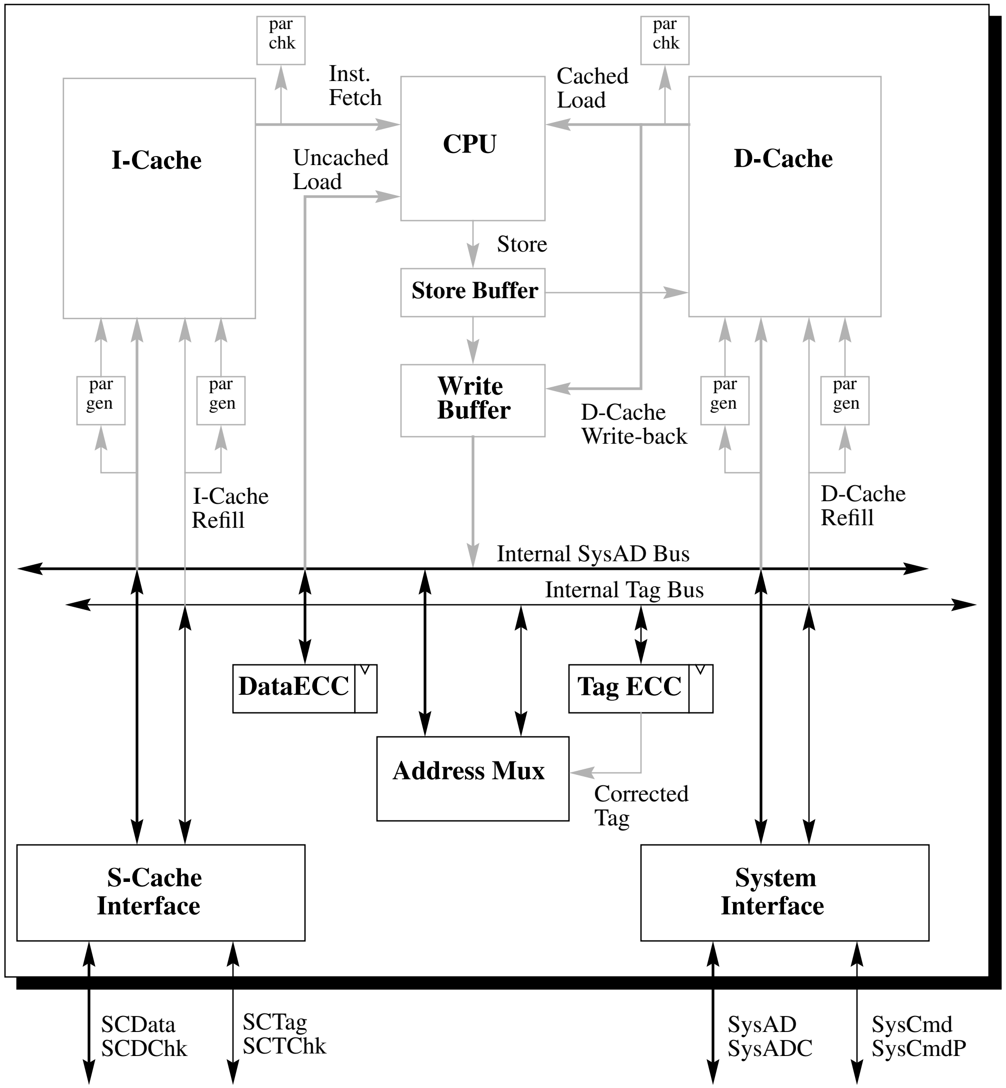
1. Programmable via boot-mode
2. The appropriate type of check bits will be generated if they do not match with the selected data protection scheme for the system interface
3. in-coming data will not be checked for external invalidate or update request

Primary Instruction/Data Cache Refill



- ECC checking is performed on the secondary cache Data and Tag
- Primary cache Tag and Tag's parity are generated for the line being refilled
- Data byte parity bits are generated for the line being refilled in the primary cache
- The processor will take a Cache Error Exception when an error is detected in the secondary cache Data or Tag portion

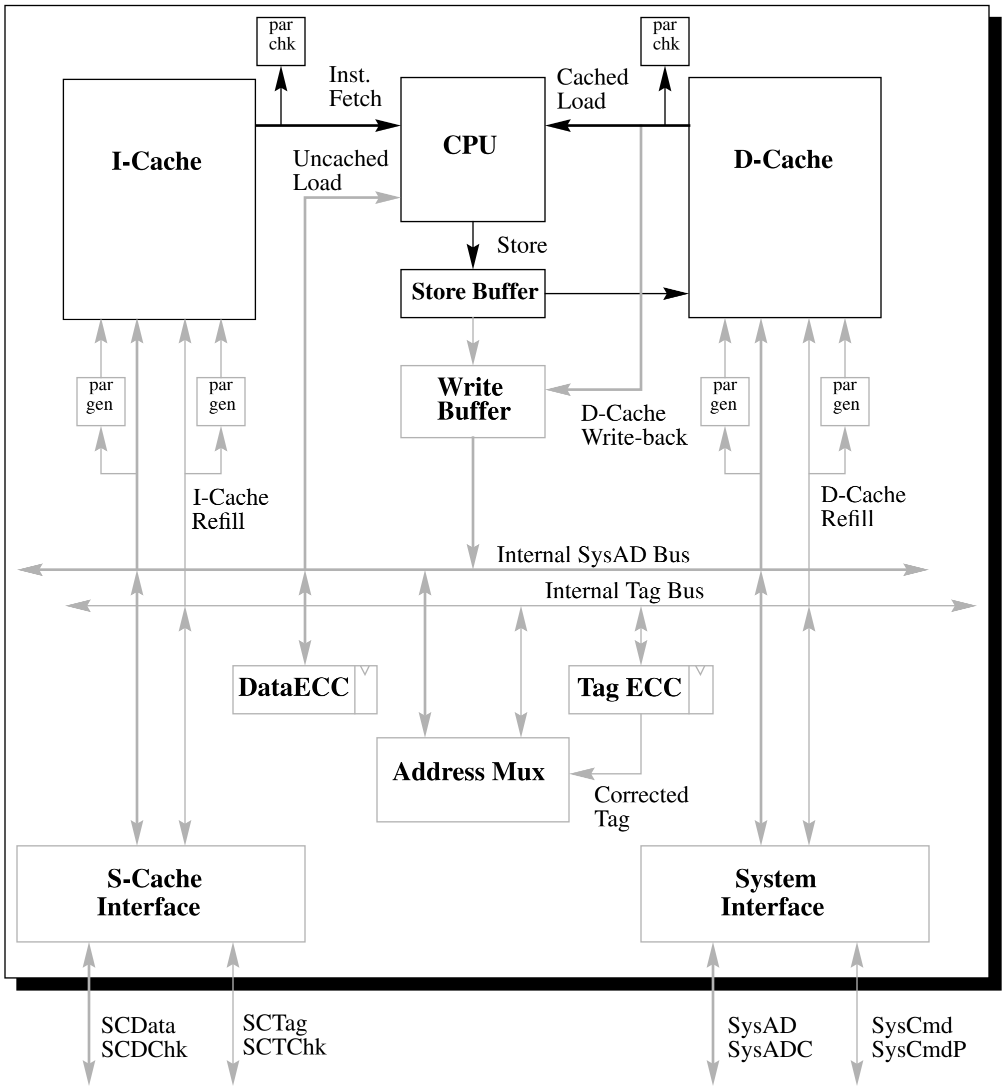
Secondary Cache Refill



- **Performs parity or ECC checking on data coming from the system interface**[1]
- **Secondary cache Tag and Tag's ECC check bits are generated for the line being refilled**
- **Data ECC check bits are generated for the line being updated in the secondary cache**[2]
- **The processor will take a Cache Error Exception when an error is detected on the in-coming data from the system interface bus**

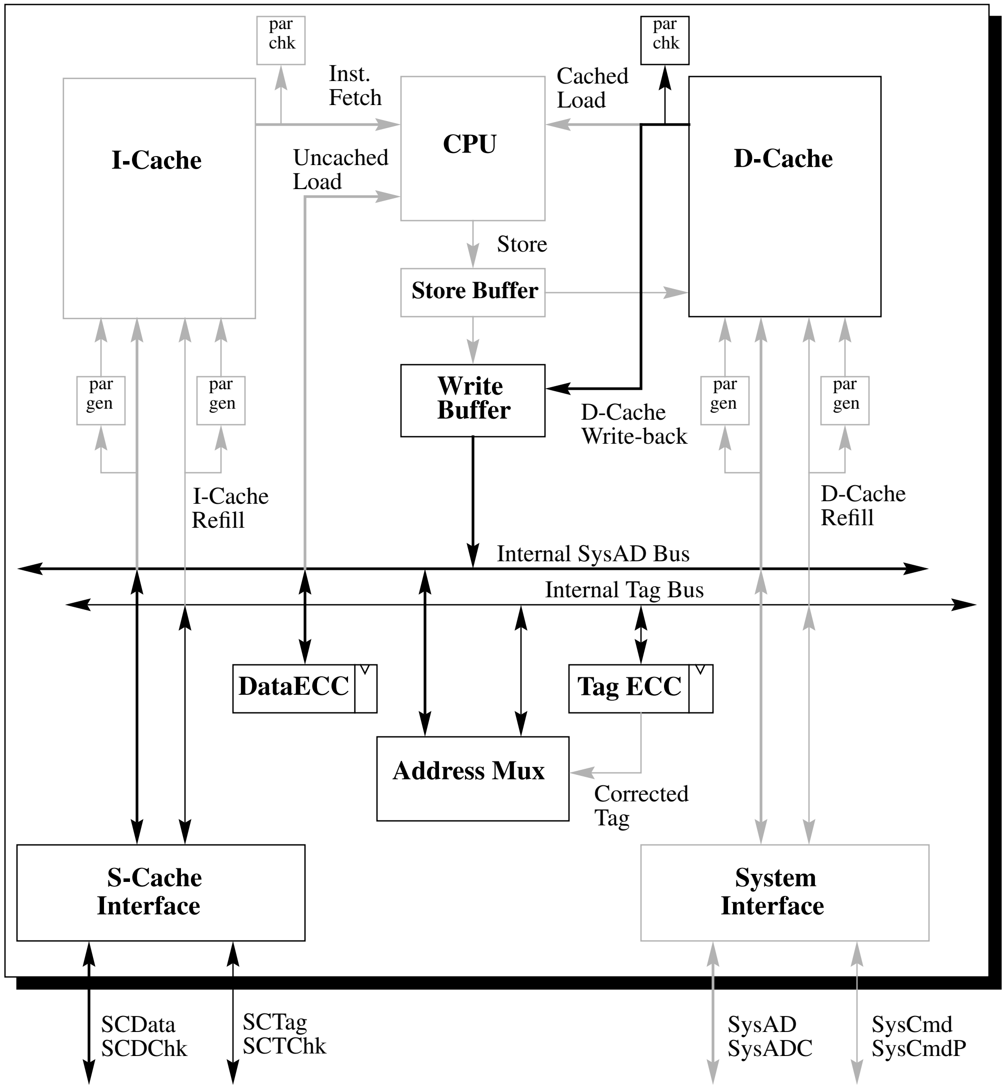
1. Byte parity or ECC checking is performed based on the selected data protection scheme for the system interface
2. If the system interface bus is programmed to use ECC protection scheme then the in-coming check bits will be written to the secondary cache unchanged

Primary Instruction/Data cache Access



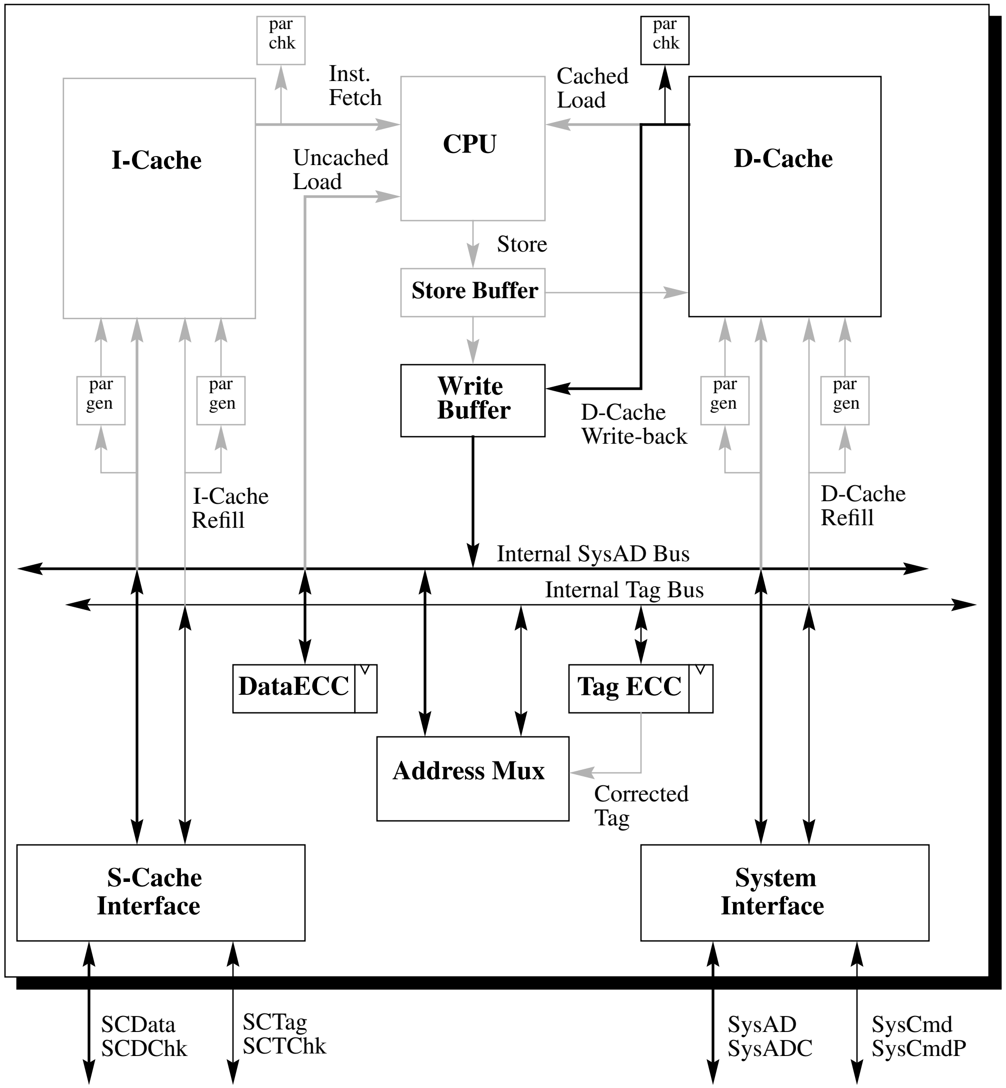
- Parity checking is performed on Data and Tag portions of the selected primary cache line
- The processor will take a Cache Error Exception when a parity error is detected on the primary cache Data or Tag portion

Primary Cache Write-back to Secondary Cache



- Parity checking is performed on Data and Tag portions of the selected primary cache line
- Secondary cache tag's ECC check bits are generated for the line being updated
- Data ECC check bits are generated for the line being updated in the secondary cache
- The processor will take a Cache Error Exception when a parity error is detected on the primary cache Data or Tag portion

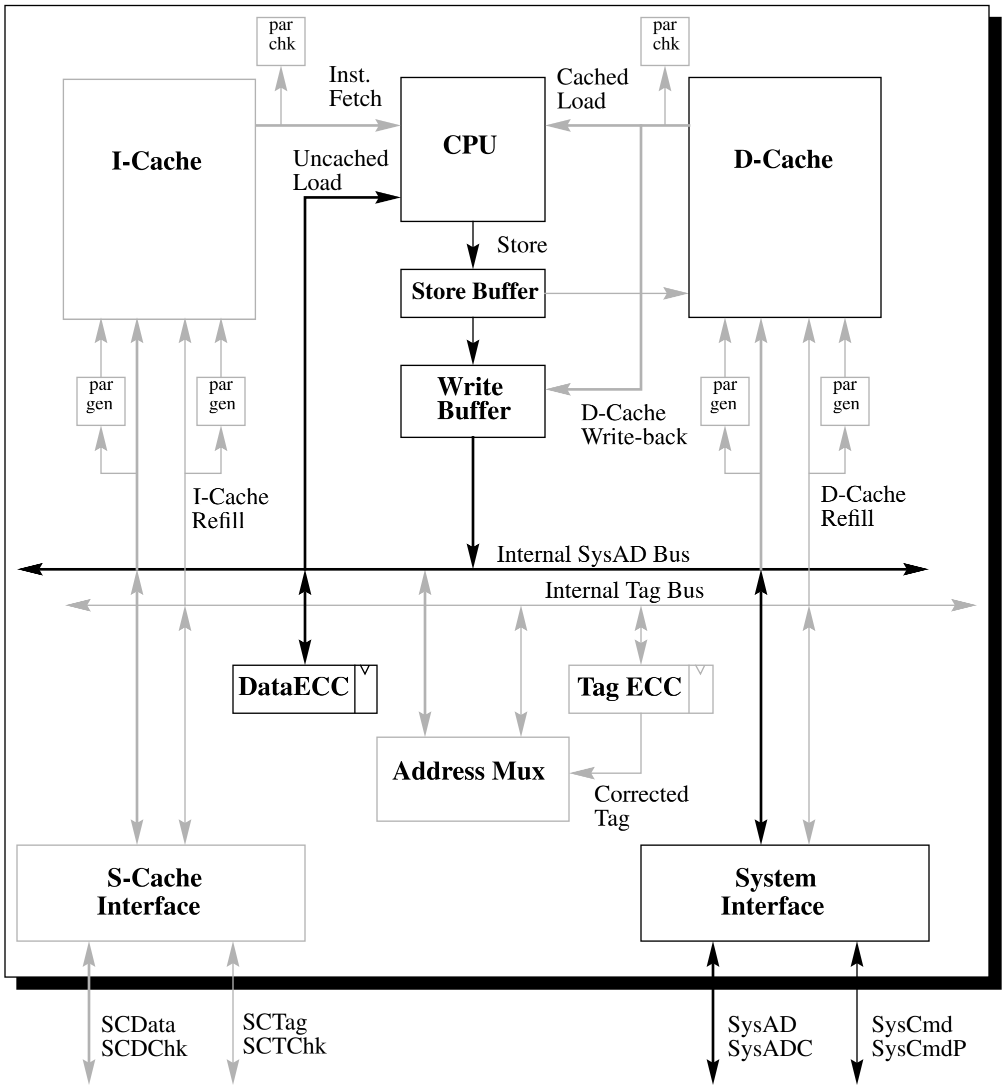
Primary/Secondary Cache Write-back to System Interface



- ❑ Parity checking is performed on Data and Tag portions of the selected primary cache line
- ❑ ECC checking is performed on the Data and Tag portions of the selected secondary cache line
- ❑ Check bits are generated for the data transfer to the system interface[1]
- ❑ The processor will take a Cache Error Exception when a parity error is detected in the primary cache data or tag portions or ECC error is detected in the secondary cache data or tag portions

1. The appropriate type of check bits will be generated if they do not match the selected data protection scheme for the system interface, Otherwise they will be transferred to the system interface unchanged.

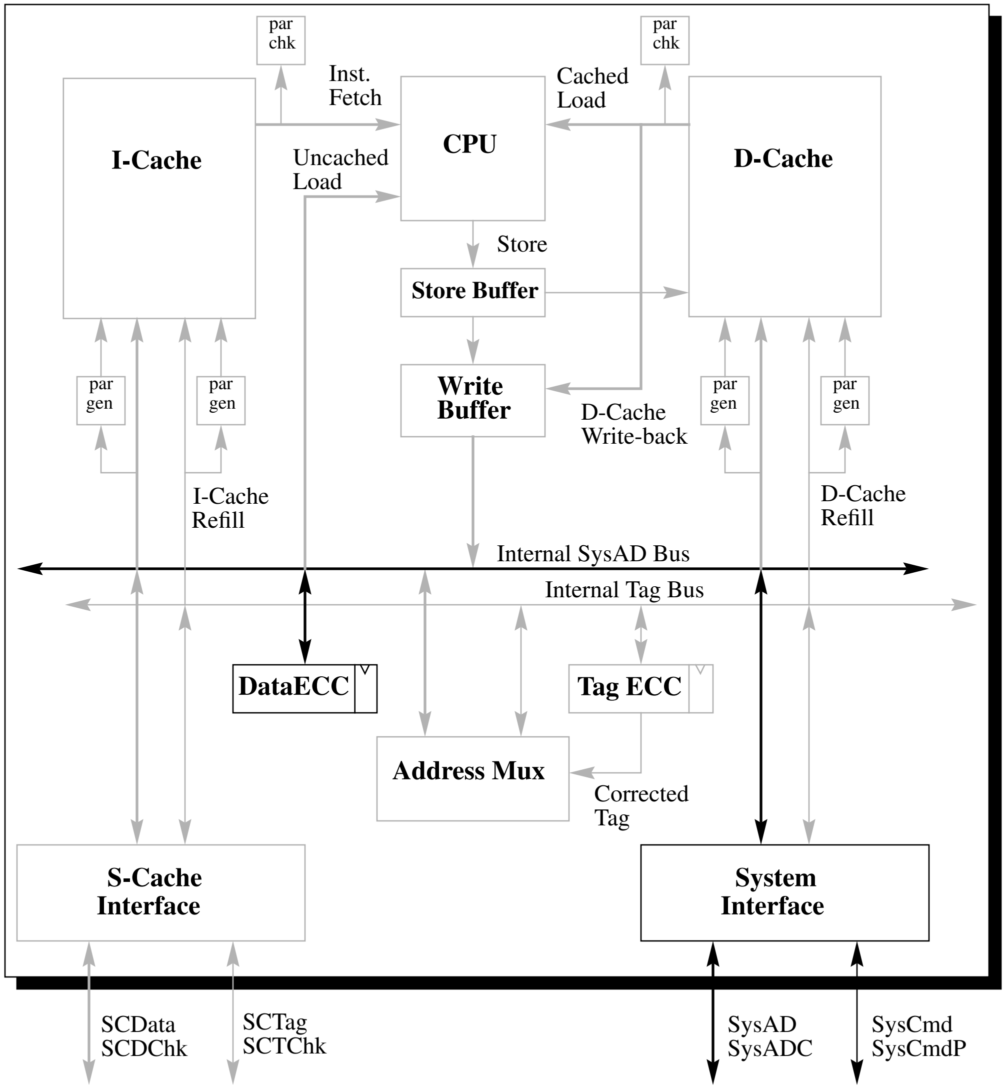
Processor Uncached Load/Store Operations



- Parity or ECC checking is performed on uncached load data coming from the system interface[1]
- For store data, the appropriate type of check bits will be generated before transferred to the system interface bus
- The processor will take a Cache Error Exception when a parity or ECC error is detected on in-coming data

1. Byte parity or ECC checking is performed based on the selected data protection scheme for the system interface

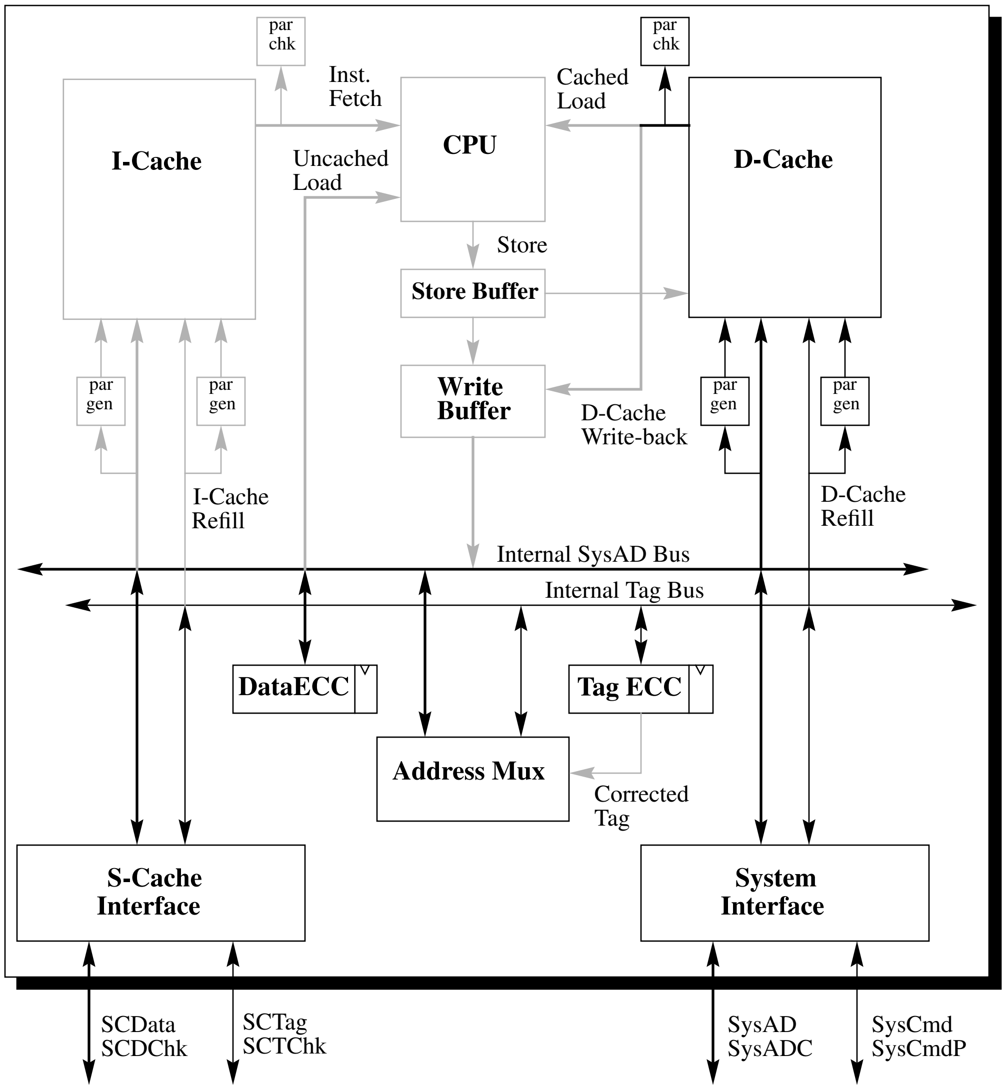
External Read/Write Requests



- **The address and command of the issue cycle coming from the system interface are not checked**
- **Parity or ECC checking is performed on in-coming data from the system interface[1]**
- **Data check bits are generated for out going data before being transferred to the system interface**
- **The processor will take a Cache Error Exception when a parity or ECC error is detected on in-coming data**

1. Byte parity or ECC checking is performed based on the selected data protection scheme for the system interface

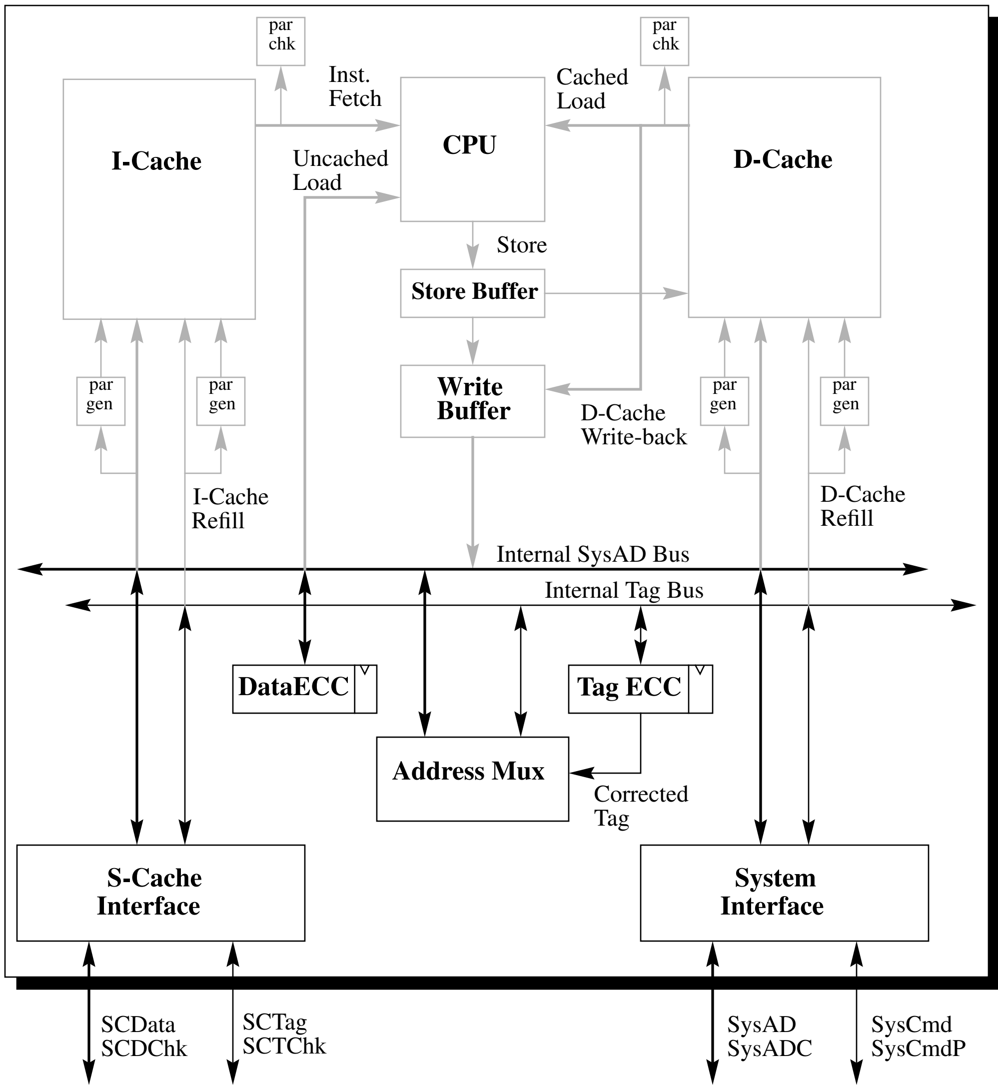
External Update/Invalidate Requests



- Secondary cache data is not checked for invalidate request
- For update request, data ECC checking is performed on the Read-Modify-Write part of the selected secondary cache line
- ECC checking is performed on the tag portion of the selected secondary cache line
- Secondary cache tag and tag's ECC check bits are generated for the Read-Modify-Write part
- Data ECC check bits are generated for the Read-Modify-Write part[1]
- The processor will take a Cache Error Exception when a parity or ECC error is detected in the primary or secondary cache line or on in-coming data

1. If the system interface bus is programmed to use the ECC protection scheme then the in-coming check bits will be written to the secondary cache unchanged

External Intervention/Snoop Requests



- Secondary cache data is not checked for intervention with state returned or for snoop request
- For intervention request with data returned, data ECC checking is performed on data returned from the selected secondary cache line
- ECC checking is performed on the portion of the selected secondary cache line
- ECC checking is performed on Secondary cache tag
- Secondary cache tag is "corrected" if required and tag's ECC check bits are generated for the Read-Modify-Write part
- Data check bits are generated for the part transfer to the system interface[1]
- The processor will take a Cache Error Exception when a parity or ECC error is detected on in-coming data

1. The appropriate type of check bits will be generated if they do not match the selected data protection scheme for the system interface, Otherwise they will be transferred to the system interface unchanged.

Cache Error Recovery

- Secondary cache data ECC does not include a single-bit error corrector
- Secondary cache tag ECC has a single-bit error corrector which is used to correct a tag error during external intervention and snoop requests. The corrected tag is written back to secondary cache only if the tag needs to be modified
- Cache Error Exception is taken when the processor detects any cache parity or ECC error. Error status is saved in CP0 CacheErr register
- Software Cache Error Exception routine can use the information saved in the CacheErr register to perform single-bit error corrections in the secondary cache Data or Tag portion

Master-checker

What is the master-checker?

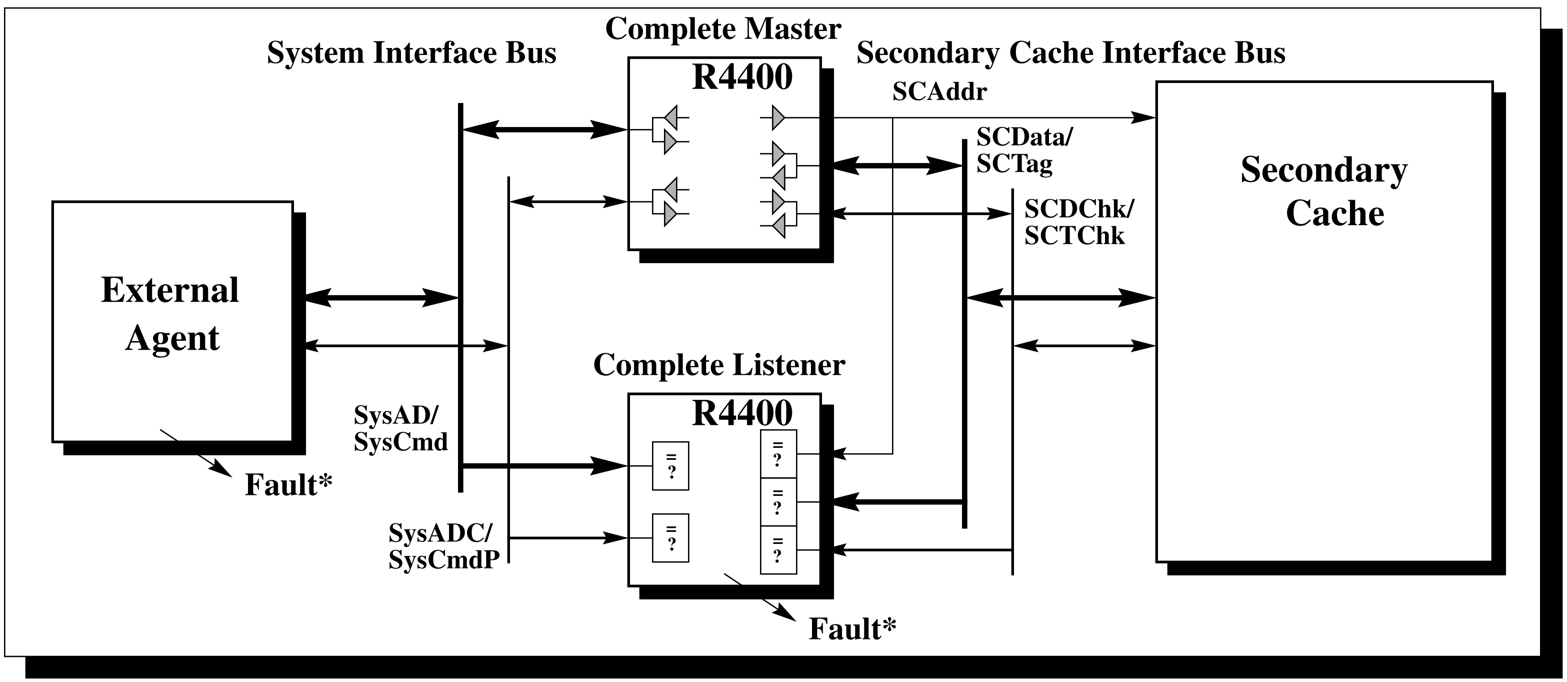
- Based on logic redundancy concept
- Enhanced system reliability for systems designed with R4400 using master-checker operation mode

How does it work?

- Two processors are connected in parallel with one being the bus driver while the other is the bus monitor
- The bus monitor compares the outputs and buses at bus-cycle boundaries, and asserts a “Fault*” signal in the event of a miscomparison
- The key factor in making this work is having both processors synchronized, and executing identical operations on a cycle-by-cycle basis
- There are two possible configurations: Master-Listener and Cross-Coupled configuations

Master-Listener Configuration

- The Master-Listener lock step configuration pairs two R4400 RISC-processors, with one as a Complete Master and the other as a Complete Listener
- The Complete Listener has disabled output drivers
- The two processors operate identically, both receiving the same inputs
- On all output cycles, the Complete Listener compares data on the output and the I/O buses with expected data, and asserts the “Fault*” signal in the event of a miscomparison



Cross-Coupled Checking Configuration

- In the Cross-Coupled Checking configuration, one of the R4400 processors, the SI-Master, drives the system interface bus, while the other R4400 processor, the SC-Master, drives the secondary cache bus
- The SI-Master has disabled output drivers for the secondary cache interface bus while the SC-Master has disabled output drivers for the system interface bus
- The two processors operate identically, both receiving the same input
- On all output cycles, both processors monitor the buses and indicate a miscomparison by asserting the "Fault*" signal