The Itanium™ Processor Features for High Availability and Reliability

Nhon Quach
IA-64 System Architect
Intel Corporation

©2000 Intel Corporation
Outline

- Problem statement
- Fault coverage and error correction
- Error signaling and containment
- Availability features
  - Data poisoning
  - Processor watchdog timer
  - Error logs
  - OS notification on corrected errors
- IA-64 Machine Check Architecture
  - Error handling flow
- Implementation cost and effort
- Summary
What are the Problems?

- **Soft errors**
  - May cause silent data corruption if undetected and loss of machine availability if not corrected efficiently
  - Caused by $\alpha$-particle bombardment and $\gamma$-ray radiation
  - Memory cells are particularly sensitive as they “remember”

- **Hard errors**
  - Field failures caused by long term wear out of metal interconnects or active devices
  - Same detrimental effects as soft errors

- **Traditionally an area that has received relatively little attention**
  - Expect problems to get worse with technology scaling

The Itanium™ Processor is designed up front to handle these problems
Extensive Fault Coverage

- ECC protected
- Parity protected

Core pipeline

L1I (16KB, 4W, RO, ~1.5M xtors)
L1D (16KB, 4W, WT, ~2M xtors)
ITLB1 (64 entries)
DTLB1 (32 entries)

Data
Tag
HW Page Walker
DTLB2 (96 entries)

L2 (96KB, 6W, Unified, ~7M xtors)

L3 (2-4MB, Unified, 294M xtors)

Tag + Data + M-state

All large memory arrays are protected

System bus, connects to chipset

Backside bus

Data Bus
Addr Bus

Bus Queues
Data Poisoning

25.4M xtors in processor
Typical Platform Topology

- **FLASH**
- **PIIX6**
- **Intel Itanium**
- **Intel Itanium**
- **Intel Itanium**
- **Intel Itanium**
- **System Bus**
- **Address Bus**
- **Data Bus**
- **Control**
- **PAL**
- **System Firmware**
- **82460 SAC**
- **82460 SDC**
- **82450 PXB**
- **FLASH**
- **FLASH Ctrlr**
- **PID**

Option 1

Option 2

PCI 32b

Hot Chips, 16 August 2000
## Error Correction in Cache, TLB, and Buses

<table>
<thead>
<tr>
<th>Structures</th>
<th>Error Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLBs</td>
<td>Not correctable</td>
</tr>
<tr>
<td>L1I Cache</td>
<td>Tag/Data parity errors 100% correctable by PAL</td>
</tr>
<tr>
<td>L1D Cache</td>
<td>Tag/Data parity errors correctable by PAL if no architectural states corrupted</td>
</tr>
</tbody>
</table>
| L2 Cache   | - Tag parity errors in clean lines correctable if no architectural states corrupted  
- Tag parity errors in dirty lines are not correctable  
- 1xECC data errors are correctable |
| L3 Cache   | - Tag parity and data 1xECC errors are correctable by ECC (ECC shared between tags and data)  
- 1xECC errors in state bits are correctable by pseudo-ECC |
| Buses      | - Address/command parity errors are not correctable  
- Data 1xECC errors are correctable by ECC |
Error Signaling and Containment

<table>
<thead>
<tr>
<th>Level of containment</th>
<th>Error signaling</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction</td>
<td>No signaling or Local MCA</td>
<td>1xECC data errors or L1I cache errors</td>
</tr>
<tr>
<td>Process</td>
<td>Local MCA</td>
<td>L1D tag or data errors on data loads</td>
</tr>
<tr>
<td>Node</td>
<td>Global MCA</td>
<td>L2 tag errors involving dirty lines on loads and stores</td>
</tr>
<tr>
<td>System</td>
<td>Global Bus Reset</td>
<td>L2 tag errors involving dirty lines on snoops or eviction</td>
</tr>
</tbody>
</table>

- Local MCA - only the detecting processor enters PAL
- Global MCA - all processors on the same system bus enter PAL
- Global Bus Reset - Global MCA + reset of states in all processors in system

Multi-level error signaling improves error containment and system availability
Data Poisoning

- Interleave bits from adjacent lines in the physical layout in all caches
  - Convert multiple-bit errors in a single cache line into single-bit errors in multiple cache lines
- Mark bad data in L2 when 2xECC errors detected from main memory or on system bus
  - Poison at a cache line granularity
  - Does not poison cache lines in L3
    - Does not write back “poisoned” lines on fills
    - Assert a Global Bus Reset on loads
- Follow rules when handling a poisoned cache line
  - A store to a poisoned line is ignored
  - A load causes a local MCA
  - A snoop causes a local MCA in the receiving processor
  - A write-back sends a notification to the OS
Processor Watchdog Timer

- **Detect system hang condition**
  - Use a dedicated “64-bit timer” on the processor
  - Increment at processor clock rate

- **Designed with proliferation in mind**
  - Account for latency variation in all platform configurations
  - Provide plenty of headroom for clock speed upgrade in future proliferation

- **Use existing Interval Time Counter (ITC) and a state machine**
  - Watchdog timer overflows if the state machine indicates that ITC has wrapped around twice

**Watchdog Timer enhances system availability**
Processor Hardware Error Logs

• **Provide error analysis/diagnostic capabilities**
  – Log error type, cause of errors, opcode, and physical address of offending instruction, etc.
  – Facilitate PAL error handling during initial error analysis phase

• **Implemented as Model Specific Registers**
  – Content exported to SAL/OS via defined PAL interfaces
  – OS error recovery based exclusively on platform logs

• **Use a log-first-error strategy**
  – Set an overflow bit on subsequent errors
OS Notification on Corrected Errors

- Allow OS/platform to keep error statistics
  - Make intelligent system maintenance decision (when to replace a processor, etc.)
  - Prevent correctable errors from becoming uncorrectable
  - May implement thresholding

- Sent by both hardware and PAL error handler
  - Use IA64 Corrected Machine Check Interrupt (CMCI)
  - Save OS the burden of periodically polling the error logs
IA-64 MCA Components

IA-64 Operating System

System Abstraction Layer (SAL)

Platform

Error Logs

Processor Abstraction Layer (PAL)

Processor

Hardware logs

Platform MCA events

Processor MCA events

MCA Hand-off

PAL MCA Procedure calls
Machine Check Architecture

- Work well with the service processor error handling model
- Integrate well with platform/OS error handling flow
  - Handoff states architecturally defined
  - Does not restrict platform/OS error handling policy
- Run from architected uncachable space with address translation turned off
  - Does not rely on caches, TLB, and main memory during PAL error handling
  - PAL error handler available immediately after reset

Architected to be MP scalable and extensible
Error Handling Flow

- Global Bus Reset
  - Need HW error containment?
    - Yes
      - Signal local or global MCA
      - Drain all queues and clear MCA pending bit
      - Send non-recoverable error indicator to SAL/OS
    - No
      - HW correctable?
        - Yes
          - Correct error
          - Send CMCI
          - Resume
        - No
          - Non-correctable error?
            - Yes
              - Send CMCI
            - No
              - To next page

HW Action
- PAL Action
- SAL/OS Action
Error Handling Flow (cont.)

- Correct error
- Send CMCI
- Log errors
- PAL correctable errors?
  - Yes: Send non-recoverable error indicator to SAL/OS
  - No: Error handling successful?
    - Yes: Resume
    - No: SAL corrects error. OS terminates process as needed
- Any platform or OS correctable errors?
  - Yes: Send non-recoverable error indicator to SAL/OS
  - No: Log errors
Implementation Cost and Effort

• Considerable die area cost
  – 5% processor die area and 10-15% L3 die area dedicated to ECC, parity, and ECC correction logic
  – Area cost (as well as design and validation complexity) of each protection feature carefully weighed against its benefits

• Intensive project level focus and effort
  – 3-5% of total team effort including definition, microarchitecture, implementation, and validation
  – Extensive customer feedback collected early on
  – System firmware and OS teams engagement important
  – Design, PAL, and validation teams participation crucial
Summary

- **Extensive error detection and correction**
  - Protect caches, TLBs, and backside and system buses

- **Multi-level error signaling and containment**
  - Contain errors at instruction, process, node, and system levels

- **Rich availability features**
  - Data poisoning
  - Watchdog timer
  - Comprehensive error logs
  - OS notification on corrected errors

- **Enhanced IA-64 Machine Check architecture**
  - Reliable PAL error handler immediately available after reset

---

The Itanium™ Processor is designed for mission critical applications
Acknowledgement

- Valentine Anders (HP), Jim Hays (HP), Amy O’Donnell, Susan Mar, Suresh Marisetty, Rad Mahalikudi, Kiran Desai, Gary Hammond, John Crawford, and Mani Ayyar
- The Itanium™ Processor design implementation, PAL, and validation teams