June, 2010

Understanding Interrupt Sources and Error Handling Procedures for the QorIQ P4080 Multicore Processor

FTF-NET-F0314

Ted Peters
Senior Applications Engineer
This session will provide a summarized look at all of the P4080 error interrupt sources and recommendations for error handling procedures at both the e500mc core level and SOC level.

Having the right software in place will allow better debug of errors and possible recovery of errors with minimal system downtime and interference.
Session Objectives

After completing this session you will be able to:

• Determine what error sources are important in a particular system.
• Implement SW routines to identify an error source and preserve relevant registers for analysis.
Agenda

► Intro to the QorIQ™ P4080 processor
► e500mc Machine Check Interrupt Reporting Paths
► e500mc Error Sources Handling and Recovery
► MPIC Error Interrupt Reporting Paths
► MPIC Error Sources Handling and Recovery
► Session Review and Wrap-up
QorIQ™ P4 Series Block Diagram

128KB Backside L2 Cache

Power Architecture™
e500-mc Core

32KB D-Cache
32KB I-Cache

1024KB Frontside L3 Cache

1024KB Frontside L3 Cache

64-bit DDR-2 / 3 Memory Controller

64-bit DDR-2 / 3 Memory Controller

CoreNet™ Coherency Fabric

Coherency Fabric

PA Mgmt

PAMU

PA Mgmt

PAMU

PA Mgmt

PAMU

Tree

Frame Manager

Parse, Classify, Distribute

Buffer

10GE

1GE

1GE

1GE

1GE

1GE

1GE

1GE

PCI Express 2.0/3.0

2x DMA

RapidIO Message Unit (RMU)

PCI Express

SRIO

PCI Express

SRIO

PCI Express

SRIO

Watchpoint

Cross

Trigger

Perf

Monitor

CoreNet

Trace

18-Lane 5 GHz SERDES

Real Time Debug

Peripheral Access Mgmt Unit

Security Monitor

Internal BootROM

Power Mgmt

SD/MMC

SPI

2x DUART

4x I²C

2x USB 2.0/ULPI

Clocks/Reset

GPIO

CCSR

PreBoot Loader

eOpenPIC

Test Port/SAP

eLBC

Security 4.0

Pattern Match Engine 2.0

Queue Mgr.

Buffer Mgr.

Freescale, the Freescale logo, AbiCore, C-5, CodeTEST, CodeWarrior, ColdFire, C-Ware, mobileGT, PowerQUICC, StarCore, and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeKit, BeeStack, CoreNet, the Energy Efficient Solutions logo, Flexis, MXC, Platform in a Package, Processor Expert, QorIQ, QUICC Engine, SMARTMOS, TurboLink and VortiQa are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2010 Freescale Semiconductor, Inc.
Agenda

► Intro to the QorIQ™ P4080 processor
► e500mc Machine Check Interrupt Reporting Paths
► e500mc Error Sources Handling and Recovery
► MPIC Error Interrupt Reporting Paths
► MPIC Error Sources Handling and Recovery
► Session Review and Wrap-up
e500mc Machine Check Errors

Machine Check Syndrome Register (MCSR)

Asynchronous

Non-maskable

Status

Error Report (synchronous/precise)

MCP
ICPERR
DCPERR
L2MMU_HIT
TLBSYNC
BSL2_ERR
NMI
MAV
MEA
IF
LD
ST
LDG

IVPR + IVOR1

Machine Check ISR
Machine Check Exception Types

► Asynchronous – error within e500mc core or \textit{mcp} assertion

► NMI – asserted to an e500mc \texttt{coren\_nmi} via MPIC PNMIR (Processor NMI Register)

► Status – gives validity of address captured in MCAR (Machine Check Address Register)

► Error Report – limit the propagation of bad data (synchronous), error source, if known, is indicated by an asynchronous MCSR bit
The following registers are updated upon taking a machine check:

- **MCSRR0** – EA of instruction executing or about to execute
- **MCSRR1** – Copy of the MSR at time of exception
- **MSR** – All bits are cleared at time of exception
- **MCSR** – Set according to machine check condition
- **MCAR/MCARU** – Updated with address associated with machine check
Machine Check Interrupt Service Routine

List steps for machine check ISR:

1. Save MCSRR0, MCSRR1 and free some scratch registers
2. Check MCSR
   - If ICPERR do this…
   - Else if DCPERR, check write-shadow mode, if enabled do this, else, do this
   - Else if….
3. Recover or reset?
   - If error is recovered, check MCSRR1[RI] to determine of machine check interrupt can recover to previous state
   - Else If not recoverable, reset system
4. Restore state
5. RFMCI
► Intro to the QorIQ™ P4080 processor
► e500mc Machine Check Interrupt Reporting Paths
► e500mc Error Sources Handling and Recovery
► MPIC Error Interrupt Reporting Paths
► MPIC Error Sources Handling and Recovery
► Session Review and Wrap-up
MCSR[MCP]

- Occurs on assertion of the \textit{mcp} input signal to the e500mc core
- If MSR[ME]=1 or MSR[GS]=1, a machine check exception is taken
- The \textit{mcp} input must be enabled via HID0[EMCP]
- Bit must be cleared by software (write 0b1 to clear)

\begin{verbatim}
MCSR

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>61</th>
<th>51</th>
<th>50</th>
<th>49</th>
<th>48</th>
<th>47</th>
<th>46</th>
<th>45</th>
<th>44</th>
<th>43</th>
<th>42</th>
<th>41</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCP</td>
<td>CPERR</td>
<td>DCPERR</td>
<td>L2MMU_HIT</td>
<td>-</td>
<td>NMI</td>
<td>MAV</td>
<td>MEA</td>
<td>-</td>
<td>IF</td>
<td>LD</td>
<td>ST</td>
<td>LDG</td>
<td>-</td>
<td>TLBSYNC</td>
</tr>
</tbody>
</table>
\end{verbatim}
MCSR[ICPERR]

- Occurs on detection of a L1 i-cache parity error
- If MSR[ME]=1 or MSR[GS]=1, a machine check exception is taken
- L1 i-cache must be enabled via L1CSR1[ICE]
- L1 i-cache parity checking must be enabled via L1CSR1[ICECE]
- Bit must be cleared by software (write 0b1 to clear)
- L1 i-cache is invalidated by hardware upon ICPERR, software may simply RFMCI
MCSR[DCPERR]

- Occurs on detection of a L1 d-cache parity error
- If MSR[ME]=1 or MSR[GS]=1, a machine check exception is taken
- L1 d-cache must be enabled via L1CSR0[CE]
- L1 d-cache parity checking must be enabled via L1CSR0[CPE]
- Bit must be cleared by software (write 0b1 to clear)
- If in write-shadow mode (L1CSR2[DCWS]=1), L1 d-cache is invalidated by hardware upon DCPERR, software may simply RFMCI
  - In DCWS mode, all modified data in the L1 cache is written through into the L2 cache.
- NOTE: DCPERR may occur on a load or store that hits in the L1D, or on a L1D cast out (non-write-shadow mode)
MCSR[L2MMU_HIT]

- Occurs on detection of a multiple hit signaled in the level 2 (L2) MMU
- If MSR[ME]=1 or MSR[GS]=1, a machine check exception is taken
- L2MMU multiple-hit detection must be enabled via HID0[EN_L2MMU_MHD]
- Bit must be cleared by software (write 0b1 to clear)
MCSR[TLBSYNC]

- Occurs on detection of simultaneous tlbsync operations
- If MSR[ME]=1 or MSR[GS]=1, a machine check exception is taken
- Bit must be cleared by software (write 0b1 to clear)

- This error typically indicates a system software issue; software is allowing the tlbsync operation without the proper mutual exclusion lock
MCSR[BSL2_ERR]

- Occurs on detection of a backside L2 cache error
- If MSR[ME]=1 or MSR[GS]=1, a machine check exception is taken
- L2 cache must be enabled via L2CSR[L2E]
- Applies L2 cache errors that are enabled and reported via L2ERRDET, L2ERRDIS and L2ERRINTEN registers
- Bit must be cleared by software (write 0b1 to clear)
Backside L2 Cache Errors

- Tag multi-way hit
- Tag parity
- Data multiple-bit ECC
- Data single-bit ECC
  - Corrected by hardware
  - Software may set threshold, ISR may flush data to remove accumulated single-bit errors
- Data parity

- Parity and multi-bit errors are not generally recoverable
  - Unless system guarantees no modified data in L2 (L2IO mode or write-through data space)
- Occurs on assertion of the **core_nmi** input signal to the e500mc core
- The **core_nmi** input is non-maskable by the e500mc core
- The **core_nmi** input is asserted/deasserted to an e500mc via setting/clearing the corresponding core’s bit in the MPIC PNMR (Processor NMI Register)
- MCSR[NMI] bit must be cleared by software (write 0b1 to clear)
Error occurred during an instruction fetch

Bit must be cleared by software (write 0b1 to clear)
Error occurred during a load instruction

Bit must be cleared by software (write 0b1 to clear)
Error occurred during a store instruction

Bit must be cleared by software (write 0b1 to clear)
Guarded load instruction

- Set along with MCSR[LD] if error is on a guarded load
- Set if error occurs is an L2 or CoreNet error (no L1 data cache error)

Bit must be cleared by software (write 0b1 to clear)
MCSR[MAV]

- Indicates whether the address captured in MCAR is valid

- Bit must be cleared by software (write 0b1 to clear)

- Read MCAR prior to clearing MCSR[MAV]

- Clear MCSR[MAV] before restoring MSR[ME]=1
- Indicates whether the address captured in MCAR/MCARU is EA (effective address) or RA (physical/real address)
- Only meaningful if MCSR[MAV] = 1
- Bit must be cleared by software (write 0b1 to clear)

<table>
<thead>
<tr>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>48</th>
<th>49</th>
<th>50</th>
<th>51</th>
<th>61</th>
<th>62</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCP</td>
<td>ICPERR</td>
<td>DCPERR</td>
<td>-</td>
<td>L2MMU_HIT</td>
<td>-</td>
<td>NMI</td>
<td>MAV</td>
<td>MEA</td>
<td>-</td>
<td>IF</td>
<td>LD</td>
<td>ST</td>
<td>LDG</td>
<td>-</td>
<td>TLBSYNC</td>
<td>BSL2_ERR</td>
<td>MCSR</td>
<td></td>
</tr>
</tbody>
</table>
Intro to the QorIQ™ P4080 processor
► e500mc Machine Check Interrupt Reporting Paths
► e500mc Error Sources Handling and Recovery
► MPIC Error Interrupt Reporting Paths
► MPIC Error Sources Handling and Recovery
► Session Review and Wrap-up
SOC Platform Errors

MPIC Error Interrupt Summary Register 0 (EISR0)

May select one destination

- mcp
- int
- cint

- IVPR + IVOR1
- IVPR + IVOR4
- IVPR + IVOR0

- Machine Check ISR
- External Input ISR
- Critical Input ISR
Error Interrupt Registers

The following registers are updated upon taking a machine check:

- **MCSRR0** – EA of instruction executing or about to execute
- **MCSRR1** – Copy of the MSR at time of exception
- **MSR** – All bits are cleared at time of exception
- **MCSR** – Set according to machine check condition (MCSR[MCP=1])
- **MCAR/MCARU** – Updated with address associated with machine check

The following MPIC registers are required for the Error Interrupt:

- **IIVPR0** – Internal interrupt 0 vector/priority register
- **IIDR0** – Internal interrupt 0 destination register
- **IILR0** – Internal interrupt 0 level register
- **MSCR1** – Machine check summary register 1
- **EISR0** – Error interrupt summary register 0
- **EIMR0** – Error interrupt mask register 0
Error Interrupt Service Routine

List steps for OR’ed interrupt directed at mcp:

1. Save MCSRR0, MCSRR1 and free some scratch registers
2. Check MCSR
   1. If MCSR[MCP=1] continue and check MPIC MCSR1
   2. Else continue with standard MCheck ISR
3. Check MPIC MCSR1
   If MCSR1[0]=1, mcp assertion is due to internal interrupt source 0 (OR’ed error interrupt).
   Else, mcp assertion is due to another MPIC interrupt source in MCSRn
4. Check MPIC EISR0 to determine OR’ed error interrupt source
   If FMAN do this…
   Else if BMAN do this…
   Else if….
5. Recover or reset?
   If error is recovered, check MCSRR1[RI] to determine of machine check interrupt can recover to previous state
   Else If not recoverable, reset system
6. Restore state
7. RFMCI
Agenda

► Intro to the QorIQ™ P4080 processor
► e500mc Machine Check Interrupt Reporting Paths
► e500mc Error Sources Handling and Recovery
► MPIC Error Interrupt Reporting Paths
► MPIC Error Sources Handling and Recovery
► Session Review and Wrap-up
EISR0[FMAN\textit{n}]

- Occurs on detection of a Frame Manager error
- Applies to FMAN errors that are reported in the FM\_EPI (FMAN Error Pending Interrupt register)
- Errors reported in FM\_EPI must be enabled to be reported as an interrupt through various FMAN Error Enable registers, for example: FMFP\_EE, FMBM\_IER, FMQM\_EIEN, etc.
- The error report bits in the various detect registers must be cleared by software (write 0b1 to clear)
EISR0[FMANn] Example

FMAN0 Error
  → MPIC core 0 mcp assertion
  → Core 0 MCheck ISR w/ MCSR[MCP]=1
  → MPIC ISR w/ MCSR1[0]=1 (read only)
    And EISR0[0]=1 (read only)
  → FMAN ISR checks FM_EPI (read only)
    If FM_EPI[0]=1
      → FMAN FPM ISR
      → Read FPM_EE[0:2]
      → Satisfy error, clear error bit (write 1 to clear)
    If FM_EPI[8]=1
      → FMAN BMI ISR
      → Read FMBM_IEVR[0:2]
      → Satisfy error, clear error bit (write 1 to clear)
    If FM_EPI[n]=1
      → FMAN ? ISR
      → Check status register
  → Return to MPIC ISR, check for other MCSRn bits = 1
  → Return to MCheck ISR clear MCP bit (write 1 to clear), recover
- Occurs on detection of a DDR controller error
- Applies to DDR errors that are enabled and reported via ERR_DETECT, ERR_DISABLE and ERR_INT_EN registers
- ERR_DETECT bits must be cleared by software (write 0b1 to clear)
DDR Memory Controller Errors

► Address parity error
► Automatic calibration error
► Corrupted data error
► Data multiple-bit ECC
► Data single-bit ECC
  • Corrected by hardware
  • Software may set threshold, ISR may read data and write data back to remove accumulated single-bit errors
► Memory select error
- Occurs on detection of a CPC (CoreNet Platform Cache) error
- Applies to CPC errors that are enabled and reported via CPCERRDET, CPCERRDIS and CPCERRINTEN registers
- CPCERRDET bits must be cleared by software (write 0b1 to clear)

**NOTE:** The CPC will attempt to correct single bit errors in the CPC array
CoreNet Platform Cache Errors

► Tag multi-way hit error
► Tag or status multi-bit ECC error
► Tag or status single-bit ECC error
► Data multiple-bit ECC
► Data single-bit ECC
  • Corrected by hardware
  • Software may set threshold, ISR may read data and write data back to remove accumulated single-bit errors
Occurs when a multi-bit ECC error is detected on various internal RAM arrays in the P4080 SOC

By default, internal RAM multi-bit ECC errors cause the assertion of RESET_REQ_B

Detected error will set RSTRQSR1[MBEE_RR]

If RSTRQMR1[MBEE_EN]=0 (this is the default value), the error is reported via the MPIC OR'ed interrupt instead of RESET_REQ_B

If RSTRQMR1[MBEE_EN]=1, the error will assert RESET_REQ_B

Error is not recoverable

MBEE_RR bit must be cleared by software (write 0b1 to clear)
Agenda

► Intro to the QorIQ™ P4080 processor
► e500mc Machine Check Interrupt Reporting Paths
► e500mc Error Sources Handling and Recovery
► MPIC Error Interrupt Reporting Paths
► MPIC Error Sources Handling and Recovery
► Session Review and Wrap-up
Session Summary

► e500mc Error interrupts are serviced via the Machine Check interrupt.

► SOC/Platform error interrupts are serviced from MPIC via the e500mc core machine check, critical input or external input interrupts.

► Detected errors may be recoverable, depending on the system.
More Information

► AN4038 - QorIQ P4080 Interrupt Sources and Error Handling

► E500CORERM – e500mc Core Reference Manual

► P4080RM - P4080 QorIQ Integrated Multicore Communication Processor Family Reference Manual

Note: Currently, these documents must be obtained from a Freescale sales representative.
Session Wrap-up

By now, you should be able to:

► Effectively describe, at a high level, all error interrupt sources and how it applies to a particular system.

► Apply the knowledge gained in this presentation to implement or refine interrupt service routines.
Question and Answer
Backup Slides
Thank you for your response.

I informed customer that basically we do not open checkstop function to general user.

On the other hand, I'd like to confirm how we handle the SW for error report machine sequence?
For e500v2 core, if machine check happens and entered into checkstop, HW reset is the only way to recover a system.
However, in P4080, there is no check-stop and it seems that if error report machine check error happens, SW must handle the sequence to be back to correct operation.

I do not understand we should treat the error report as same as other int/cint and P4080 accept to return to normal operation even if fatal data error happened.
So could you please explain the sequence to recover from machine check in case of P4080?
I basically understand the error report interrupt is better to treat like that,
-While error report machine check occurred by fatal data error (multibit ECC error, bus error etc), the e500mc core which is interrupted by machine check error will gather error syndrome and then the core should perform stopping other 7 cores via MPIC.

Regards,
Kosuke Sogawa

---

Thank you for previous reply. It was helpful for customer.
And I got response from customer for further question.

-Do we plan to open the specification of checkstop module (including CKSTP_OUT pin) to general user?
-If not, do we have alternate functionality which can stop all the P4080 operation stop like a traditional checkstop?

I'll confirm the background of their requirement but please advice whether we can support traditional checkstop.

Regards,
K.Sogawa

---

Any update on this issue?

---

Hotline,
Customer asked us to explain about checkstop in P4080.

Could you please advice below questions?

1) Seeing e500mcRM and P4080RM, Checkstop state is removed from e500mc's machine check error and, as instead, error report machine checks are implemented while MSR[ME]=0.

However the error is asynchronous so it seems hard to stop all the HW timely. I (also my customer, NEC) could not understand enough for this change. Could you explain why is the asynchronous checkstop removed and changed to synchronous error? Or do we have any other error or way to stop HW like checkstop?(RCPM can realize it?)

2) In the P4080RM, I found Checkstop out pin but I do not understand what caused to assert the checkstop out pin. Seeing page 35-6, there is a note that "Collect device hardware assert errors from IPs." What the hardware assert from IPs means? Are there any list of HW assert errors?

3) In the chapter fro RCPM unit, there is a description about DACRD which seems to generate check stop but I could not find any register details for DACRD in the P4080RM. Where is the description for DACRD? Not yet on the current document?

Regards,
Kosuke Sogawa

Correct, the e500mc will not checkstop when a machine check condition occurs while MSR[ME] =0.

For a machine check error report condition, if the MCSR shows both an error report exception type and an async exception type, SW should determine if the error is recoverable.

For instance, if a LD error report + BSL2_ERR async is detected, the LD error is possible due a L2 cache error whose recoverability depends on the L2ERRDET register. An ICPERR would be recoverable as the data in invalidated by hardware, same for DCPERR in write-shadow mode.

If only a an error report exception type is given, the reason for the error is not captured by the MCSR, and the system would expected the error to be reported via the MPIC. For instance a LD error report due to a PCI error. The system would expect the PCI error to be reported and handle via the MPIC causing an assertion of the mcp_ input to the core, if correctable, the instruction the signaled the LD error report can be returned to and execute without error.

== 12/9/2009 10:45:31 AM (TP) ==

There are no plans currently. If a strong use case can be given, I can definitely present it to make a case for putting the feature in public documentation.

== 11/30/2009 2:02:18 PM (TP) ==

1) There are now multiple types of machine check errors, asynchronous, error report (synchronous), and NMI. The errors are type are specified in the e500mcRM rev E table 2-7 MCSR.

Async errors are taken only if MSR[ME] or MSR[GS] are set. When an async error bit is set, it remains set and the interrupt will be taken when if MSR[ME] or MSR[GS] are set, SW must clear these bits.

Error Report errors are not gated by MSR[ME] or MSR[GS].

NMI errors are not gated by MSR[ME] or MSR[GS]. NMI errors are generated by asserting the e500mc NMI input via the MPIC PNMR.

2) Checkstop out can be asserted due to conditions set in CKSTPCR. This register is not in public documentation at this time.

3) The DACRDn, as well as DACREn and DACRRn registers are not in public documentation at this time. These registers along with CKSTPCR will determine what events can cause the CKSTP_OUT_N pin to be asserted.

The checkstop out functionality is currently a expert mode feature for tool vendors to implement.
I have questions about the error handling of non-posted transactions in multi-core environment.

1. If an error occurred at PCIe with a read request from e500mc, the error would be reported to the one of eight cores, which was configured in MPIC?

2. If an error occurred with a request from Core0 and the MPIC was configured for the PCIe error to be routed to Core1, how the request in core0 completion queue would be treated? If Core0 can detect it as an error, what kinds of error?

Core1 would get OR'ed interrupt (mcp_, cint_, int_ depending on MPIC config) from MPIC. Core1 would need to behave as described in #1 of my last response.

It is up to SW on how a system design will handle this type of error.

One solution could be that in the MCP ISR, core0 poll a memory location for some amount of time expecting another core to handle the PCI error via the OR'ed MPIC interrupt, that core would then instruction core0 (possibly by updating the polled memory location with an action code) to perform some action to recover (ie. reset, kill process, skip the instruction, etc.) If core0 does not get an action code in x amount of time, it can assume the core that was to handle the PCI error is down.

The SW solution really depends on how the system intends to recover from such an error.

1) The PCI error would be signaled via the MPIC internal OR'ed interrupt source 0. This source can be configure to interrupt any core, via mcp_, cint_, int_, etc. via the IIDR0 and IILR0 registers. It is recommended the OR'ed interrupt be directed to a core (typically core 0) mcp_ input. The MCP handler would see MCSR[MPIC]=1, then check the MPIC MCSRR registers for the mcp_ assertion source. If MCSRR[0] = 1 the mcp_ assertion source is the internal interrupt source 0 (the OR'ed interrupt source). Then the OR'ed interrupt handler would check the EISR0 register to determine the OR'ed interrupt source, then investigate that particular IP block(s) source's error registers (example would be the PEX error status registers).

2) I'm working on getting this answered.
EISR0[FMANn]
EISR0[SRIO]
EISR0[Internal RAM MBECC]