DOT/FAA/AR-10/21

Air Traffic Organization
NextGen & Operations Planning
Office of Research and
Technology Development
Washington, DC 20591

Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 4 Report

September 2010

Final Report

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AR-10/21		

4. Title and Subtitle	5. Report Date
MICROPROCESSOR EVALUATIONS FOR SAFETY-CRITICAL, REAL-TIME APPLICATIONS: AUTHORITY FOR EXPENDITURE NO. 43 PHASE 4 REPORT	September 2010
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Rabi N. Mahapatra*, Jason Lee*, Nikhil Gupta*, and Bob Manners**	TAMU-CS-AVSI-72005

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
* Aerospace Vehicle Systems Institute Texas Engineering Experiment Station Texas A&M University Department of Computer Science College Station, TX 77843-3141 **Lumark Technologies, Inc. 4904 Tydfil Court, Suite 100 Fairfax, VA 22030	11. Contract or Grant No. DTFACT-03-Y-90018

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
U.S. Department of Transportation Federal Aviation Administration Air Traffic Organization NextGen & Operations Planning Office of Research and Technology Development Washington, DC 20591	Phase 4 Final Report August 2007 – December 2008
	14. Sponsoring Agency Code AIR-120

15. Supplementary Notes

The Federal Aviation Administration Airport and Aircraft Safety R&D Division COTR was Charles Kilgore.

16. Abstract

Research into the characteristics of complex microprocessors and systems-on-a-chip revealed increasing difficulties in design assurance of airborne systems containing these commercial off-the-shelf devices. This was found to require system-level methods, including monitoring, fault or anomaly detection, and corrective actions based on resilient system architectures and system controls. This approach is accumulatively identified as safety nets. Phase 4 initiated the evaluation of simulation and test tools to aid in the safety net approach. It was determined that safety nets must be involved at multiple levels with aircraft systems. Phase 5 will further research the device characteristics and the safety net approach. A Federal Aviation Administration Handbook for the selection and verification of systems using these devices was planned in Phase 4 and will be implemented with the results from Phase 5.

17. Key Words	18. Distribution Statement
Microprocessor, System-on-a-chip, Avionics, Certification, Safety net	This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161. This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	69	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

A	Address bits
A_STRM_CNT	Stream count
A_STRM_DIS	ECM streaming disable
ABMDIS	ARM port interface boot mode disable
ACS	Address to chip-select setup
Addr	Address
AEH	Airborne electronic hardware
AFE	Authorization for expenditure
AHD	Address hold disable
AL	Address length
AM	Address mask
AM	Address multiplex size
AMX	Address multiplexing
API	ARM port interface
APIBN	ARM Port Interface Boot Normal
ARM	Advanced RISC Machine [also the company ARM Holdings plc (programmable logic controller)]
AS	Address/data byte 0-3
AS16	Address Shift for 16-bit port size
ATMU	Address translation and mapping unit
ATOM	Atomic operation
AVSI	Aerospace Vehicle Systems Institute
BA	Base address
BANK	Bank
BASE CNT	Base Count
BC	Byte Count
BCTLC	Defines the use of LBCTL
BCTLD	Buffer control disable
BI	Burst inhibit
BIT	Built-in test
BLK	Flash block address
BMD	Bus monitor disable
BMI	Bus monitor interrupt enable
BMT	Bus monitor timing
BMTPS	Bus monitor timer prescale
BOOT	Flash auto-boot load mode
BR	Base register
CA	Column address
CCB	Core complex bus
CCD	FCM command completion checking disable
CCI	FCM command completion interrupt enable
CEH	Complex electronic hardware
cfg rom loc	Configure Boot ROM interface location

CHT	Command hold time
CI	Count inhibit
CI0	Critical Interrupt bit(s) for Core 0
CI1	Critical Interrupt bit(s) for Core 1
CI	Column index
Cint	Critical interrupt
CLKDIV	System clock divider
CLKR	Clock ratio
CM	Command
CMD	General-purpose FCM flash command
CNT	Count
CNTL_TIMER0	Control timer 0
CODEC	Coder decoder
Config	Configuration
Core	Microprocessor core
CORE_STRM_DIS	e500 core streaming disable
COTS	Commercial-off-the-shelf
CPU	Central processing unit
CPU0_EN	CPU0 port enable
CPU0_ PRI	Priority level of the e500 core 0 (CPU) port
CPU1_EN	CPU1 port enable
CPU1_PRI	Priority level of the e500 core 1 (CPU) port
CPU_RD_HI_DIS	DDR targets read queue assigned to the e500 core (CPU) ports
csb_clk	replaced by ccb_clk in a recent revision of reference A-1
CSD	Chip select error checking disable
CSCT	Chip select to command time
CSI	Chip select error checking interrupt enable
CSNT	Chip select negation time
CST	Command setup time
CTPR	Current task priority register
CTO	Completion timeout
CTRL	Controller
CW0	Wait for LFRB to return high or time-out, then issue command from FCR(CMD0)
CW1	Wait for LFRB to return high or time-out, then issue command from FCR(CMD1)
CWTO	Command wait time-out
D	Data
DDR	Double data rate
DECC	Error checking bits
DEFLATE	Unidentified software application
DMA	Direct memory access
DRAM	Dynamic random access memory
DROM	Data read only memory
DS	Disable timer period
DSP	Digital signal processor

DSP_REG	Digital signal processor register
DSPSS	Digital signal processor subsystem
DUART	Dual universal asynchronous receiver transmitter
E	Enable bit(s)
E2PROM	Electrically erasable programmable read only memory
EAD	External address latch delay
EADC	External address delay cycles of LCLK
ECC	Error correction codes
ECCM	ECC mode
ECM	e500 coherency module
EEBACR	ECM CCB address configuration register
EEBPCR	ECM CCB port configuration register
EEDR	ECM error detect register
EEER	ECM error enable register
EEPROM	Electrically erasable programmable read only memory
EHTR	Extended hold time on read accesses
eLBC	Enhanced local bus controller
EOI	End of interrupt register
EP	External signal
EPAR	Determines odd or even parity
EPIC	Embedded programmable interrupt controller
EPROM	Erasable programmable read-only memory
FAA	Federal Aviation Administration
FBAR	Flash block address register
FBCR	Flash byte count register
FCM	Flash control machine
FCR	Flash command register
FCTD	FCM command time-out disable
FCTI	FCM command time-out interrupt enable
FIR	Flash instruction register
FMEA	Failure Mode and Effect Analysis
FMR	Flash mode register
FPAR	Flash page address register
G0CL	General line 0 control
GCR	Global configuration register
GPCM	General-purpose chip-select machine
GTBCR	Global timer base count register
GTCCR	Global timer current count register
GTVPR	Global timer vector/priority register
HID1	Hardware implementation-dependent register 1
HOM	Host-only mode
Hreset	Hard reset
I/F	Interface
I/O	Input/output
I^2C	Inter-integrated circuit
IBS	Interrupt bit select

ID	Identification
int	Interrupt
IP	Intellectual property
IPIDR	Interprocessor interrupt dispatch register
IPIVPR	Interprocessor interrupt vector/priority register
IRDA	Infrared data association
IRQ	Interrupt request
ISR	Interrupt status register
JTAG	Joint test access group
L1	Level 1 Cache
L2	Level 2 Cache
LA	Local bus address
LAD	Multiplexed address/data bus A-2 See A.6.12
LAE	Local access error
LAEE	Local access error enable
LALE	External address latch enable
LBC	Local bus controller
LBCR	Local bus controller register
LBCTL	Local bus control
LBIUCM	Local bus clock mode
LBS	Local byte select
LCLK	Local bus clock
LCRR	Clock ratio register
LCS	Local bus chip select
LDIS	Local bus enabled/disabled
LDP	Local bus data parity
LFALE	Local bus flash address latch enable
LFCLE	Local bus flash command latch enable
LFRB	Local bus flash read write busy
LFRE	Local bus flash read enable
LFWE	Local bus flash write enable
LFWP	Local bus flash write protect
LGPL	Local bus GP line
LGTA	General-purpose chip select machine terminate access
LOAD_TIM	Load timer
LOE	Output enable
LPBSE	Local bus parity byte select enable
LSOR	Special operation initiation register
LSYNC_OUT	Local bus phase locked loop synchronization out
LSYNC_IN	Local bus phase locked loop synchronization in
LTEAR	Transfer error address register
LTEATR	Transfer error attributes register
LTEDR	Transfer error check disable register
LTEIR	Transfer error interrupt enable register
LTESR	Transfer error status register
LUPWAIT	Local bus UPM wait

LURT	UPM refresh timer
LWE	Local bus write enable
M	Mode
MAD	Machine address
MAR	Memory address register
MAMR	First UPM mode register
MBMR	Second UPM mode register
McBSP	Multichannel buffered serial port
MCMR	Third UPM mode register
MCP	Machine check processor
MCU	Microcontroller unit
MDR	UPM/FCM data register
MDVAL	Memory data valid
MER	Message enable register
MRTPR	Memory refresh timer prescaler register
MS	Main/spare region locator
MSEL	Machine select
MSG	Message
MSGR	Message register
MSIIR	Shared message signaled interrupt index register
MSIR	Shared message interrupt register
MSK	Mask
MSR	Message status register
MSRCID	Memory debug source port ID
MULT_ERR	Multiple error
Mux	Multiplexer
MxMR	Mode X mode register
NAND	NOT AND (a type of logic operator)
NOP	No operation
NOR	NOT OR (a type of locic operator)
OCeaN	On-chip network
OP	Command opcode
OP bits 30, 31	Flash operation
OR	Option register
OVLY	Overlay
P0	Processor bit for Core 0
P1	Processor bit for Core 1
PA	Page address
PARD	Parity and ECC error checking disabled
PARI	Parity and ECC error checking interrupt enable
PBYP	PLL Bypass
PCI	Peripheral component interconnect
PGS	Page size, buffer size, and block size
PI	Page index
PIC	Programmable interrupt controller (also known as OpenPIC)
PIR	Processor core initialization register

PLL	Phase-locked loop
PMMR	Performance monitor mask register
PRIORITY	Priority
Prog	Program
PS	Port size
PTP	Refresh timers prescaler
R&D	Research and development
R/W	Read/write
RapidIO	an architecture for high-performance packet-switched interconnect
RAM	Random access memory
RAW	Read after write
RAWA	Read after write atomic error checking disable
RB	Read FBCR bytes of data from flash device into current FCM RAM buffer
RBW	Wait for LFRB to return high or time-out, then read FBCR bytes of data from flash device into current FCM RAM buffer
RCWHR	Reset configuration word high register
RCWL	Reset configuration word low
Reg's	Registers
Rev	Revision
RFEN	Refresh enable
RFXE	Read fault exception enable
RISC	Reduced instruction set computer
RLF	Read loop field
ROMLOC	Boot ROM interface location
ROVR	Roll-over
RS	Read one byte (8b port) of data from flash device into next AS field of MSR
RSW	Wait for LFRB to return high or time-out, then read 1 byte (8b port) of data from flash device into next AS field of MDR
RST	Read setup time
RST	Reset
RTC	Real time clock
RTCA	Radio Technical Commission for Aeronautics
RTM	Real-time mode
RTOS	Real-time operating system
S	Status bit(s)
SAM	Shared-access mode
SDRAM	Synchronous dynamic random-access memory
SCY	Cycle length
SerDes	Serializer deserializer
SETA	External address termination
SEU	Single-event upset
SGMII	Serial gigabit media independent interface
Simics	A full-system simulator from Virtutech Inc.
SoC	System-on-a-chip
SPI	Serial peripheral interface

SRAM	Static random access memory
SRESET	Soft reset
SRIO	Serial RapidIO
SRS	Shared interrupt register select
SVR	Spurious vector register
TASKP	Task priority
TCR	Timer control register
TDMI	A version of ARM7 processor
TLF	Refresh loop field
TLU	Table lookup unit
TRLX	Timing relaxed
UA	User-defined address
UART	Universal asynchronous receiver transmitter
UPM	User-programmable machine
UWPL	LUPWAIT polarity active low
V	Valid bit
VECTOR	Vector
WAR	Write after read
WARA	Write after read atomic error checking disable
WB	Write FBCR bytes of data from current FCM buffer to flash device
WCET	Worst-case execution time(s)
WLF	Write loop field
WP	Write protect
WPD	Write protect error checking disable
WPI	Write protect error checking interrupt enable
WS	Write one byte (8b port) of data from next AS field of MDR to flash device
XACS	Extra address to chip-select setup
XIO	External input/output
XOR	Exclusive OR (a type of logic operator)

EXECUTIVE SUMMARY

The Aerospace Vehicle Systems Institute Microprocessor Evaluation Project Phase 4, described in this report, was an industry-driven research collaboration that focused on identifying unique opportunities and risks of using system-on-a-chip (SoC) devices for safety-critical aerospace applications.

The emergence of microprocessors and SoCs as essential design components to meet performance requirements in target systems and the relative lack of complete design information available on these devices, presents safety issues to system designers and regulators. These issues are further heightened by the increased complexity in microarchitectural features and on-chip functionality, making safety assessment and, consequently, aircraft certification more difficult and expensive.

To determine the appropriate characteristics of microprocessors and SoCs in safety-critical aerospace applications, common and typical risks inherent to their design must be known. This research analyzed recent commercial off-the-shelf devices using publicly available specification documents. Given resource and time constraints, two SoC designs that best represented typical devices were selected for a detailed analysis. This research and in-depth safety analysis of the Freescale MPC8572 and Texas Instruments TMS320VC5470 revealed three safety issues:

- The effect of shared resources on timing
- Risks associated with programmable register status and configuration
- Visibility and debug limitations

Additional evaluation of devices used by the project participants did not expand these safety issues, indicating that they are generally common across current SoC designs. This project further studied four metrics based on current regulatory guidance to determine the ease of compliance for SoCs to meet suitability, testability, stability, and availability requirements.

This research revealed that there is no consistent method to evaluate the safety of systems using these devices at the component or subsystem level and that a system-level analysis is required to mitigate this issue at the system level. Current evaluation guidance under RTCA/DO-254 must be supplemented with a system-level approach. Because complete knowledge of the evaluated devices cannot be acquired, it must be assumed that they will malfunction and/or fail. This led to the development of a safety net approach to assure safety at the system level.

This research defines safety net as the ability to demonstrate and protect against unintended/misleading device behavior at a level above the microprocessor/SoC through an appropriate combination of board and system-level architecture. The safety net approach is grounded in the premise that devices will fail and malfunction. Several system-level techniques were evaluated during the research, including:

- External monitoring of safety-related behavior
- Redundancy

- External watchdogs
- Architectures that allows run-time correction

Phase 4 was the first half of a planned 2-year process. In the second half (Phase 5), follow-on research is planned to develop and validate the safety net approach, to inform systems designers and certifiers, and to provide guidance for the selection and evaluation of microprocessors and SoCs. These results will be documented in a Federal Aviation Administration Handbook.

1. INTRODUCTION.

This report documents the research and results of the Aerospace Vehicle Systems Institute (AVSI) authority for expenditure (AFE) 43 Supplement 3 (Phase 4) Microprocessor Evaluations Project. This project researches methods to evaluate microprocessors for safety-critical aerospace applications.

The emergence of microprocessors and systems-on-a-chip (SoC) as essential design components to meet performance requirements in target systems and the relative lack of complete design information available on these devices, presents safety issues to system designers and regulators. These issues are further heightened by increasing complexity in microarchitectural features and on-chip functionality, making safety assessment more difficult.

To determine appropriate characteristics of microprocessors and SoCs in safety-critical aerospace applications, common and typical risks inherent to their design must be known. Phase 4 research analyzed two SoCs designs: the Freescale MPC8572 and Texas Instruments (TI) TMS320VC5470, and revealed three common safety issues:

- The effect of shared resources on timing
- Risks associated with programmable register status and configuration
- Visibility and debug limitations

Neither RTCA/DO-178B [1] nor RTCA/DO-254 [2] specifically documents how complex, modern microprocessors and SoCs should be assured. Phase 4 research revealed that there is no consistent method to evaluate the safety of systems using these devices at the component or subsystem level and that a system-level analysis is required to mitigate this issue at the system level. Current evaluation guidance under DO-254 must be supplemented with a system-level approach. Because complete knowledge of the evaluated devices cannot be acquired, it must be assumed that they will malfunction and/or fail. This led to the development of a safety net approach to assure safety at the system level. Section 5 of this report defines the safety net approach that will be developed and validated in Phase 5.

This project investigates assessment criteria and safety concerns for microprocessor applications. The expected outcomes are to

- develop methods and procedures to permit the safe, economical qualification of microprocessor applications with complex, nondeterministic architectures.

- provide guidelines for selecting microprocessors to be used in safety-critical aerospace applications.

- provide input to the Federal Aviation Administration (FAA) for regulations and policy development regarding the design and test of commercial off-the-shelf (COTS) microprocessor components.

The team of industry and Government pooling their resources as well as their respective goals not only expands feasible research and development (R&D) considerations, but provides real-world guidance across a much broader span of responsibilities and with tighter focus on realistic long-term success. As the R&D partnership is mirrored in the requirements of the resultant systems and products to meet regulatory requirements, it provides a laboratory to identify, refine, and meld common requirements and to smooth the entry of new technologies into the future aerospace infrastructure. Government participation ameliorates the perception of providing information to competitors and ensures that contributors have their issues considered by members with public responsibilities; industry participation ensures that the project goals are practical and worthwhile to commercial concerns.

AVSI is currently performing and considering additional research into a broader range of programmable devices under the categories of airborne electronic hardware (AEH) and complex electronic hardware (CEH) in safety-critical airborne applications. There may also be extensions of the research in future AVSI projects being planned now.

1.1 BACKGROUND.

Phase 1 of this project reviewed the available literature and surveyed microprocessor users to identify the issues and potential solutions associated with the use of microprocessors in regulated safety-critical applications; addressed the suitability of DO-254; listed possible evaluation criteria and categories; discussed tentative classification of some of these criteria in safety-critical levels; described the potential issues with obsolescence management; presented feature modeling as a potential approach to identify risks; presented testing and validation aspects, as well as safety concerns related to these aspects; and outlined risk-related issues that arise in an SoC that integrates microprocessor intellectual property (IP) cores with peripheral components.

Phase 2 developed three major project objectives and found an approach to work toward the solution of the issues and achievement of these objectives. These objectives included formulating a process for assessing safety of emerging microprocessor features and SoCs, developing tools and guidelines to aid in the design and safety assessment processes, and preparing a draft Microprocessor Selection and Evaluation Handbook. This draft Handbook is considered only an initial version of this Handbook that identifies preliminary evaluation criteria and safety concerns for microprocessors. The Handbook will be completed in Phase 5.

Phase 3 validated the potential approaches identified in Phases 1 and 2. The Sun® Microsystems OpenSPARC T1 SoC, an open-source SoC, was used as a validation and verification test-bench for these approaches.

Phase 4, described in this report, was an industry-driven research collaboration that focused on identifying unique opportunities and risks of using COTS SoC devices for safety-critical aerospace applications.

As a summary of major Phase 1, 2, and 3 findings, the following concerns were identified with respect to COTS microprocessors and SoCs:

- In Phase 1, unpredictable worst-case execution times (WCET) due to the unpredictable nature of microprocessor features were found.

- In Phase 2, safety issues presented by emerging microprocessor features and increased levels of feature integration into SoCs were revealed.

- In Phase 3, the lack of accurate documentation for COTS devices was identified as a major roadblock to modeling features and devices for safety analysis.

In addition, in Phase 2, approval frameworks for microprocessors and SoCs were identified to provide standard processes that could be implemented by aerospace industries. The research staff applied the proposed frameworks on candidate microprocessors (Freescale MPC7447 and Freescale MPC8540), but due to time and information constraints, were unable to completely apply the steps.

As a continuation and completion of Phases 1 and 2, additional phases were planned to validate the approach begun in Phases 1 and 2 and to continue the development of processes, services, and prototype tool development. These results will be included in the Microprocessor Selection and Evaluation Handbook in Phase 5 to facilitate application to actual, safety-critical applications. The proposed AFE43 Phase 5 activities will be a direct continuation of the project activities and will realize the value of the completed research. In addition, Phase 5 will also consider the feasibility of more complete future solutions for safety assurance of systems employing future microprocessors and SoCs.

1.2 OBJECTIVES.

The following objectives were identified for Phase 4 of this research project:

- Facilitate the safe, economical qualification of microprocessor applications with complex, nondeterministic architectures.

- Attempt to integrate regulatory and development/integration requirements to identify and develop common techniques and tools that facilitate the smooth transition from initiation to certification for aerospace vehicles.

- Determine methods to predict, evaluate, and tune microprocessor-embedded system performance and safety.

- Provide guidance to select and evaluate microprocessors for safety-critical aerospace applications.

- Provide recommendations to the FAA for regulations and policy development regarding system design of safety-critical systems using emerging COTS microprocessor components.

1.3 PROJECT SCOPE.

The Microprocessor Evaluations Project identified areas that were acknowledged but not addressed through Phase 4 due to cost and schedule constraints, including, but not limited to

- identification of additional critical features.
- identification of any other fault effects that are associated with emerging features.
- implementation of features that were modeled as tools to gain confidence on the modeling and verification approach.
- exhaustive analysis for risk evaluation of specified features, and the evaluation of a tool-based approach for estimating risks in a complete system.
- risk analysis specific to SoCs.

1.4 PROJECT LIMITATIONS.

The areas associated with safety assurance of the evolving microprocessors are growing and changing faster than the research associated with their solution. Continuous development of methods, tools, and processes may have to be implemented to keep up with the changes. Consequently, the Handbook to be generated by this project is a document that requires on-going support from the FAA and industry members to remain current.

The use of COTS microprocessors and SoC in safety-critical applications extends beyond aerospace applications (e.g., nuclear applications, medical systems, pharmaceutical research and implementation, automotive and marine applications, etc.). Currently, there is little communication between these varying industry domains to share experience and technical knowledge to mitigate safety issues.

Once solutions are established, centralized services by a trusted source need to be provided. These services, provided across a wide range of industrial domains, must include continued development to match on-going evolution of complex electronic hardware and software; maintenance of models, tools, and processes; library services; and associated regulatory policy, procedures, and tools.

The size of the AFE43 problem set suggests additional research based on (1) the replanning of project activities and priorities based on R&D results and (2) the probable expansion of the using communities as the project results become refined into effective solutions. The need for solutions to the issues is vital to industrial, national, and global economies. The development of consortia, involving manufacturing, system developers, integrators, maintainers, and users, may be the only feasible, long-term approach. AVSI is currently performing and considering

additional research into a broader range of programmable devices under the categories of AEH and CEH in safety-critical airborne applications. There may also be extensions of the research in future AVSI projects currently in the planning stages.

2. THE COTS MICROPROCESSOR AND SoC RISK ANALYSIS.

2.1 OVERVIEW.

Using COTS microprocessors and SoC in aerospace applications must account for common and typical risks inherent to the complexity of their design. This research analyzed COTS devices using publicly available specification documents. Given resource and time constraints, two SoC designs that best represented typical devices were selected for a detailed analysis.

2.2 ANALYSIS METHODOLOGY.

Each device was evaluated with the following considerations:

- Configuration of registers using a memory map
- Associated registers, which are vulnerable to producing nondeterministic situations
- Effects of unintentional register reconfiguration
- Potential nondeterministic interactions with other blocks
- Effects of unintentional "wake-up" of that block if the block is unused

2.3 THE COTS DEVICE CANDIDATES.

The following devices were selected on the basis of complexity, manufacturer history, and system composition. Additionally, the selection of device analysis candidates attempted to capture the diversity of available devices on the market. Table 1 lists the COTS devices under consideration for evaluation. The devices marked with an asterisk were selected for an in-depth safety evaluation within this phase of research.

Table 1. The COTS Device Analysis Candidates

Manufacturer	Product	Core Architecture
Freescale	MPC8540	1 PowerPC e500 Core
Freescale	MPC8572*	2 PowerPC e500 Cores
Freescale	MPC5567	1 PowerPC e200 Core
AMCC	460GT	1 PowerPC 440 Core
PA Semi	PA6T-1682M	1 PowerPC PA6T Core
PMC-Sierra	MSP8520	1 MIPS E9000 Core
PMC-Sierra	MSP8120	1 MIPS 34K Core
ST	STR912FAW44	1 ARM9 Core
TI	TMS320VC5470*	1 ARM7TDMI Core

Table 1. The COTS Device Analysis Candidates (Continued)

Manufacturer	Product	Core Architecture
TI	TMS470	1 ARM7TDMI Core
Intel	T7500	2 Core 2 Cores
Intel	U7500	2 Core 2 Cores
Intel	Pentium M	1 Core

AMCC = Applied Microcircuits Corp. ST = ST Microelectronics
PA Semi = Power Architecture Semi TI = Texas Instruments

2.4 THE FREESCALE MPC8572.

The Freescale MPC8572 is a dual-core SoC that represents a typical COTS device used in safety-critical aerospace applications. Figure 1 depicts the MPC5872 configuration.

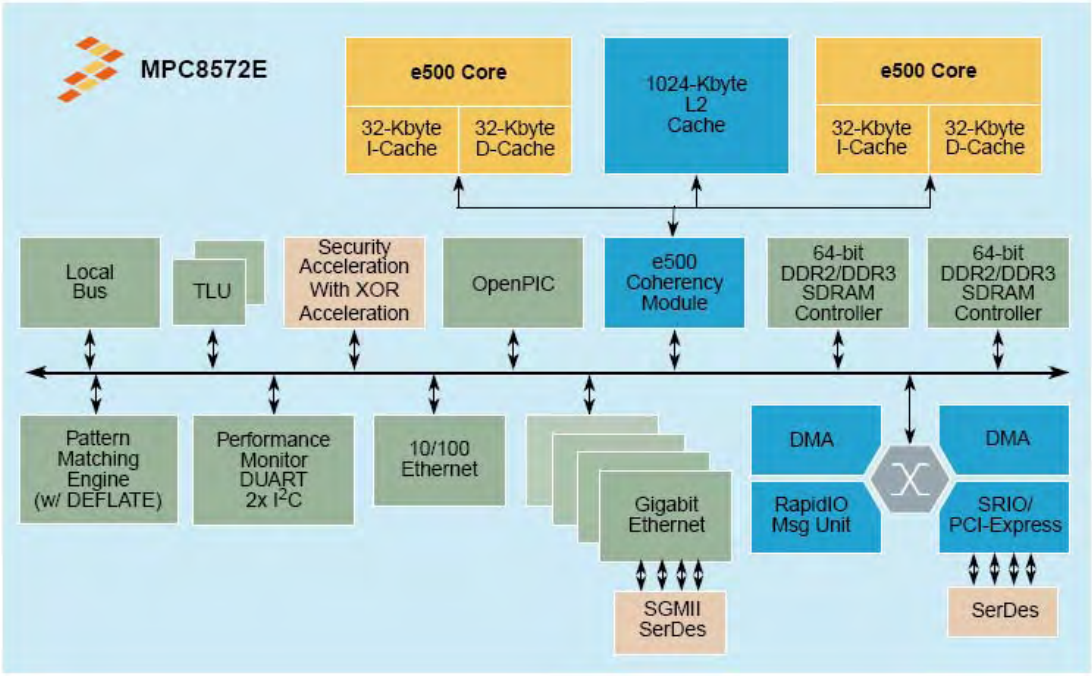


Figure 1. Freescale MPC8572 Block Diagram [3]

Using the reference documentation available from the manufacturer, the following 15 on-chip cores were identified in the MPC8572. Each core type was evaluated for potential safety hazards based on software and hardware reconfigurability, testability, and expected failure effect criticality. Please refer to appendix A for a register-level analysis of the MPC8572 components.

- Two e500 processor cores
- e500 coherency module (ECM)
- On-chip network (OCeaN)

- Direct memory access (DMA) controller
- RapidIO interface controller
- Peripheral component interconnect (PCI)-express controller
- Embedded programmable interrupt controller (EPIC)
- Inter-integrated circuit (I^2C)
- Double data rate (DDR) memory controller
- n-chip cache
- Ethernet controllers
- Dual universal asynchronous receiver transmitter (DUART)
- Local bus controller (LBC)
- Joint test access group (JTAG) boundary scan

2.4.1 Processing Core and Peripheral Evaluation.

The MPC8572 contains two e500 processor cores that may be independently controlled through direct on-chip peripherals—the ECM and EPIC. Together, these components form the central application logic control and execution of the entire device. The following issues were identified during this phase of research.

- The processing cores are vulnerable to inappropriate software reconfigurability through configuration registers. For example, setting a single bit within the EPIC configuration space may reset or disable processing cores at runtime. Refer to appendix A for register details.

- Interprocessor communication is controlled by the ECM and EPIC through interrupts. Configuration registers of the ECM and EPIC must be verified for correctness before and during operation.

- The EPIC provides an interrupt interface between the processing cores and external interface cores. The proper configuration of the EPIC must be verified for correctness before and during operation to ensure that interrupts from external interfaces are not erroneously masked.

2.4.2 Interface Core Evaluation.

The DUART, Ethernet, PCI-Express, LBC, JTAG, I^2C, Memory, and RapidIO interface controllers offer the same class of functionality to the system—off-chip communication. Therefore, common issues and evaluation techniques can be applied to each of these cores. The following issues have been identified during this phase of research.

- Data packets should include cyclic redundancy check codes to validate data correctness.

- Critical configuration registers associated with each used interface core should be identified and monitored. Refer to appendix A for examples of critical configuration registers.

- If a function has been disabled, the system designers must verify it remains disabled. Inadvertent re-enabling may cause spurious activity within the device.

- A method for coping with corrupt or lost data at the system level must be identified.

2.4.3 The PCI-Express Controller Evaluation.

In addition to the concerns common to all external interface cores, the PCI-Express protocol presents further safety concerns. Specifically, PCI-Express devices initiate communication through a link-training process, which autonomously negotiates link bandwidth and frequency. It is possible that communicating devices may negotiate a level of link performance that system designers do not anticipate, leading to potential safety hazards. Although this negotiation process is mandatory, it is possible to monitor link properties through status registers. Inappropriate link negotiations can therefore be detected and reset.

2.4.4 The DDR Controller and Memory Evaluation.

SoC microprocessors include controllers to support DDR dynamic random access memory (RAM) use as main memory. DDR dynamic RAM and its successors have become the standard high-volume main memory for most large computers. This memory is now migrating into embedded and avionics designs. There are safety concerns using SoC DDR controllers and DDR dynamic RAM in avionics related to determinism, WCET, and several other issues.

DDR memory requires periodic refreshing—memory bits are stored as capacitive charges that slowly leak. The SoC DDR controllers manage this refresh and read/write access complexity, allowing SoC devices to simply read and write locations without concern for DDR-related overhead. The memories have a protocol for accessing a location that involves opening a row, reading or writing a location, and closing a row. Tables 2 and 3 list the safety benefits and concerns of DDR memory, respectively. Overall, the proper use of DDR memory is not a safety issue for aerospace applications.

Table 2. The DDR Memory and Controller Safety Benefits

Feature	Benefit
Built-in Error Correction Circuitry (ECC)	Single-event effect immunity, minimum speed penalty. Single-bite error correction, double-bit detection
ECC built-in test (BIT)	Inject errors for BIT purposes
64-bit-wide data typical, sometimes dual controllers	Higher bandwidth
Optimization for cache line transfers	Higher bandwidth

Table 3. The DDR Memory and Controller Safety Concerns

Property	Issue
DDR rows must be open to access locations.	WCET is affected, limited number of rows can be open at one time. Opening a row is a separate command to the memories. Determinism is affected as prediction of execution time is difficult.
DDR is dynamic memory and requires periodic refresh.	Affects determinism, as the DDR controller will periodically use the DDR buses to do a refresh cycle. Loss of refresh makes memory contents unreliable, for example, during a hard reset.
DDR contents are not static.	A typical method of having SoC DDRs maintain contents through a "Reset" is to use a soft reset instead of a hard reset. Soft resets are exceptions that do not completely reset the hardware. They rely on software to reconfigure the hardware. This can lead to lockup states that do not recover on a reset.
ECC does not scrub memory.	Any error that occurs in memory, single- or multiple-bit, will remain until software rewrites the location.
ECC initialization	Error correction bits are random on power up, and full RAM initialization is required on any hard reset prior to access.
ECC exception	If exceptions are used to manage single- or multiple-bit errors, the exception routine will affect WCET. If errors are frequent, due to components such as a faulty data line, the exceptions can severely affect execution time.
DDR is typically available as commercial-grade parts.	These are not specified over military specification temperature and need up-screening and derating of DDR controller timing parameters to ensure proper operation.
DDR is a commodity product.	A product may become obsolete while still in use.
DDR controller configuration	Many registers control timing, ECC, memory sizes, transfer sizes, etc. All are subject to single-event upset (SEU), malfunctioning software, incorrect programming, and insufficient margin.
Cache interaction	For optimal throughput, the entire used DDR address range would be made write-back cacheable, with DDR programmed to read and write entire cache lines. As cache line misses or write-backs occur, cache lines are moved to and from DDR. It is hard to predict WCET and other determinism effects.
Highly optimized around moving cache line-sized objects	If used for subcache line-sized reads and writes, memory access becomes very inefficient. This is intended to be used with cache enabled.
Writes smaller than a cache line	If a cache is set to write through, or is turned off for certain areas, the subcache line writes will be sent to DDR. Typical Freescale DDR controllers always write entire cache lines and use mask bits to the memories to block certain byte lanes or beats of the write. This reduces throughput and affects WCET.

Table 3. The DDR Memory and Controller Safety Concerns (Continued)

Property	Issue
Reads smaller than a cache line	Typical Freescale DDR controller always reads entire cache lines even if only one byte is requested. This leads to inefficient use of resources.
Writes smaller than a cache line with ECC enabled	With ECC enabled for subcache line writes, the controller will perform read/modify/write cycles, where read is a full cache line and write uses masks to block writes to unwanted portions.
Cycle time prediction	It is difficult to exactly determine read or write latency to any given location of memory. WCET and determinism are negatively affected. Causes include open row status, cache line misses and flushes, refresh cycles, and bus-shared resources.
DDR access from other SoC resources	Other devices in the SoC may access DDR resources. This competes with processor and cache for resources and affects determinism and WCET.
DDR bus to external memories is complex.	Layout is critical, and proper termination of signals is required. Signal integrity analysis and control of board impedances is important.
Freescale errata	On the MPC8540, there is DDR errata. Freescale does not always fix errata, and workarounds may be difficult.
ECC double-bit error response	The DDR controller responds to uncorrectable errors with a programmable processor exception. It is up to the software to respond appropriately. This method is inferior to an automatic hardware response.
Source synchronous data movement	The DDR or controller sources data with strobe signals used to determine where to capture data on the receiving end. This is more complex than past methods and is open to timing problems.
Debug	Use of logic analyzers with DDR interfaces is extremely difficult. Some issues include capturing data correctly, mixing command along with actual data being moved, and following code flow when only cache line reads and writes are observable.
DDR termination	DDR buses must be terminated, which increases power consumption.
Cache line data placement	Placement of data and code in the address space can have a large impact on throughput. For example, if data variables needed for a routine are mapped to addresses in different cache lines, WCET and determinism are affected.

2.5 THE TEXAS INSTRUMENTS TMS320VC5470.

The TI TMS320VC5470 is a dual central processing unit (CPU) processor integrating a digital signal processor (DSP) and a reduced instruction set computer (RISC) microcontroller unit (MCU). It represents a typical COTS SoC that would be used in safety-critical aerospace applications. Figure 2 shows the TMS320VC5470 functional block diagram.

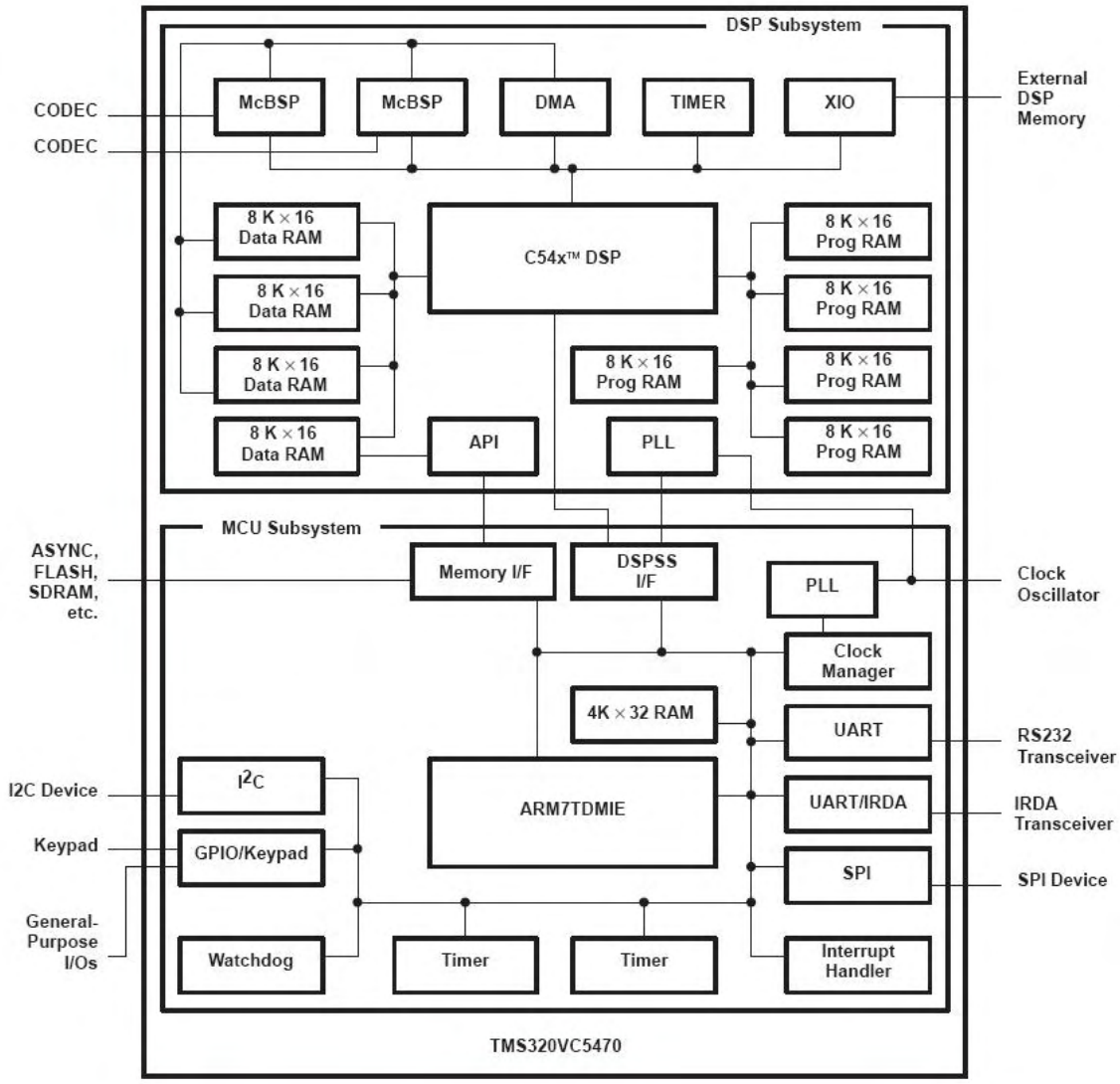


Figure 2. Texas Instruments TMS320VC5470 Functional Block Diagram [4]

The TI TMS320VC5470 is divided into the DSP and the MCU subsystems. The DSP subsystem contains the following on-chip components [4].

- TMS320C54x DSP Core
- On-chip static random access memory (SRAM)
- External DSP memory interface

- Advanced RISC Machine (ARM) port interface (API) for providing access to the MCU and the DSP's SRAM
- Multichannel buffered serial ports for the DSP core
- DMA controller
- External memory interface

The MCU subsystem includes the following seven components.

- ARM7TDMI CPU Core
- Memory interface
- General purpose input/output (I/O)
- Timers
- Interrupt handler
- Clock generator and phase-locked loop (PLL)
- Universal asynchronous receiver transmitter (UART)-based serial and infrared data association (IrDA) ports

2.5.1 The ARM Port Interface.

The TMS320VC5470 uses an API for information exchange between the MCU and the DSP units. The API is implemented as 8K X 16-bit, dual-access RAM and can operate in two modes: shared-access mode (SAM) and host-only mode (HOM). In the SAM, both the DSP and the MCU have access to the API memory, while in HOM, only the MCU can access the API memory. In SAM, if both the DSP and the MCU try to access the API memory simultaneously, the MCU is given priority and the DSP access is delayed by one cycle. The effect of this behavior is that accesses to the API memory from the DSP might take more than one cycle when the API is operating in SAM. This aspect must be taken into account by the designer while using the API.

2.5.2 The DSP Memory Map.

The DSP core features a configurable memory map. The memory map can be controlled using the following configuration registers:

- "The OVLY bit in the program mode status register can be used to control whether the lower address range of the DSP program space is mapped to on-chip RAM or external memory. Similarly, the DROM bit in the program mode status register controls whether the higher address range of the DSP data space is mapped to on-chip RAM or external memory." [4]

- "ABMDIS bit in the bank-switching control register forces the DSP out of API boot mode and disables the memory mapping used for the API boot mode." [4]

- "DSP_APIBN bit in the DSP_REG register controls the instruction fetch address of the DSP after a reset." [4]

Changes in these bits will alter the memory map. The application running on the DSP core will crash if the memory map changes unexpectedly. Hence, these registers are critical and must be protected.

2.5.3 The MCU TIMER0.

The MCU core provides three timers. TIMER0 is configured as a watchdog timer by default, which resets the MCU and all subsystem peripherals after reaching the value 0. The user program is expected to write periodically into the LOAD_TIM register before the counter reaches 0 to prevent a reset. On power up, the TIMER0 is automatically configured as a watchdog timer. Although this watchdog feature may be disabled, changing the WATCHDOG bit in the CNTL_TIMER0 register puts it back into the watchdog mode. Hence, the CNTL_TIMER0 register is critical.

2.5.4 Interprocessor Communication.

Communication between the DSP and the MCU cores is achieved by interrupt signals [5]. Further, the DSP subsystem includes a six-channel DMA controller for independent transfers. The DMA controller is able to interrupt the DSP core based on the status of the transfer. In a safety-critical system, proper attention must be paid to these features as they can hamper the system capability to meet task deadlines.

2.5.5 Power Management.

Both the MCU and the DSP cores feature multiple power-down modes with varying levels of power savings. Resuming from these modes requires additional time, depending on the level of power savings. The system designer must be able to choose the appropriate power management level for the system design to ensure timely execution. The timing properties of a resume from each of the power-down modes are not documented in the specification. This makes the power management feature incompatible for use in a safety-critical system.

2.5.6 Status Registers.

To determine the internal state of the processor, users need visibility into status registers of the DSP and the MCU cores. The DSP core has 27 memory-mapped CPU registers, including two status registers that are mapped in the data memory space. The DSP core also features a processor mode status register. These registers enable visibility into the internal state of the cores.

2.5.7 The ARM7TDMI Debugging Features.

The ARM7TDMI-based MCU core has been enhanced by Texas Instruments with additional debugging features for tools compatible with the ARM7TDMIE core. These features provide the ability to set software and hardware breakpoints during code execution. The debugging features include single-processor (MCU only) and multiprocessor (DSP and MCU) debug capabilities. The debug features can be used in both 16- and 32-bit modes.

3. THE SoC RISK CLASSES.

3.1 OVERVIEW.

After reviewing the devices discussed in section 2, broad observations were made regarding the technical risks of using COTS SoCs in safety-critical aerospace applications. The development of these general risk classes was one of the major goals of Phase 4. SoCs present additional risks over standalone microprocessors due to the greater levels of functional integration into a single package. These risk classes address both the operational and developmental risks of this greater integration.

3.2 THE EFFECT OF SHARED RESOURCES ON TIMING.

The effect of shared resources on systems is a known problem. SoC designs amplify this problem beyond what has been common for discrete parts in that control of, and visibility into, the interactions of the shared resources and sharing agents is often not well documented, e.g., bus arbitration protocols.

One approach is to follow these steps:

1. Identify the resources that might have contention, e.g., buses, devices (RAM, Flash, PCI bridges, etc.), and architectural/instruction set architecture-implied components, such as cache.

2. Identify the resource consumers, e.g., CPU and other bus masters, such as DMA controllers.

3. Characterize the contention to the degree required by the system. Some general guidance is that the shorter the time period of interest to the application, the more detail is required of the characterization. This is because timing variation on small scales can be orders of magnitude in difference (e.g., a single RAM access without contention might be 10s or 100s of nanoseconds, but could be several orders of magnitude larger with bus contention, dynamic RAM, refresh, and other interactions).

   Characterizing the duration and frequency of these interactions is often very difficult, if not impossible, on short time scales. On longer time scales, it may be possible because, for example, RAM refresh might be known to happen relatively infrequently. If the response time for the system requires time scales similar to the potential delays imposed by these effects, then they must be very well understood, perhaps even modeling queue

depths, arbitration schemes, maximum "hold-off" times, etc. In effect, any device with sufficiently complex behavior that is not understood or modeled well enough may need to be treated as a pseudorandom response time device and the randomness bounded or otherwise accounted for in some manner.

4. Resource usage and contention need to be evaluated on both "normal" and "fault" conditions, for the effect of those conditions on the SoC, and the ability of the SoC to affect the system to which it belongs.

3.3 REGISTER STATUS AND CONFIGURATION.

COTS microprocessors and SoCs are highly configurable devices. Ensuring that the hardware and software controls for device configuration are operating as expected is a necessary part of demonstrating that the device will operate correctly. Phase 4 of this project has included an analysis of design trends in COTS device configurability. The high level of component integration offered by SoC designs adds complexity to configurability issues. For example, the Freescale MPC8572 analyzed in Phase 4 of this project includes the ability to disable and enable individual processing cores, interface units, interrupts, and memory subsystems through hardware or software controls.

The following abstract approach can be tailored to specific applications:

- Assume that a valid combination of I/O has been selected for normal operation of the application.

- Document the chosen configuration and the implications of that configuration, such as:

  - Review the protection capabilities for the Control/Configuration Memory Space.

  - Review excessive (extra) capabilities of the device that are not being used. Include these devices as disabled within the configuration parameters.

  - Document the initial (preconfiguration) and program default configuration for the entire Control/Configuration Memory Space.

  - Review the power-up process for the selected device.

- Verify the order of the voltage rails for the device and how it aligns with other devices.

- Verify the preconfigured values of the external pins.

  - Assure that these pins cannot damage other components.

  - Verify that the external components are protected (if necessary) during configuration.

- After initial configuration of the device, validate the configuration.
- Unexpected configuration changes
    - Perform a failure mode and effect analysis (FMEA) for every register.
        - Determine the impact of an undesired change to the register (independent of the cause)
    - Based on FMEA, classify registers as:
        - Safe: No protection necessary (no impact).
        - Noncritical: Do not need to be directly monitored (covered by overall safety net).
        - Critical: Require direct explanation of how these registers are covered by the safety net, as defined in section 5 of this report.
- Verification deliverables
    - Device Configuration Documentation.
    - Analysis of Preconfiguration Device State.
    - Analysis of device during configuration.
    - Register FMEA and Classification (includes validation of classification).
        - Includes every device configuration register.
    - Validation of safety net (ties into FMEA) (Discusses how the safety net covers the configuration items that may contribute to a safety hazard.)

3.4 VISIBILITY AND DEBUG.

As COTS device manufacturers provide greater visibility and debugging capabilities for a given microprocessor and SoC, less risk exists for unintended and unexpected behavior to occur. The ability and depth of visibility into a microprocessor and SoC were examined to the extent possible in this research. The research concluded that the existing tools and techniques continue to be less effective due to increasing complexity and advancements in technology. This report summarizes the limitations and consideration for evaluating the level of visibility and debug capability identified by the research.

Limitations are based on

- limited number of pins.
- specifications do not reveal complete device operation.
- accessibility of internal components of complex SoCs.

- on-ground debugging hardware/software during integration and maintenance.
- software diagnostic via the Ethernet port (from Flash).
- hardware debug via JTAG port (each device is different).
- performance monitoring.
- operating system (polling).

Each SoC must be evaluated with related applications and associated tools, such as:

- Virtutech Simics® (the version used not timing accurate).
- CodeWarrior®.
- Instruction set simulator.
- VxWorks Real-Time Operating System (RTOS).
- Green Hills Software, Inc. Integrity® RTOS.
- Hardware probes.
- Bound with test interface.
- Software development and debug tools.

As technology advances, the likelihood of discovering all behavioral traits of a microprocessor and SoC in its intended application is unlikely through a design assurance process in accordance with DO-178 and DO-254. The ability to deterministically demonstrate through a comprehensive level of physical tests and structured approaches at the device level alone is unlikely. This research assumes that the device will fail and/or malfunction and suggests the use of a safety net to mitigate safety hazards. Safety is assured through the robustness of the architecture and less dependent, or less demonstrated, at the device level as currently provided for under DO-178 and DO-254.

4. COMPLIANCE ISSUES.

4.1 OVERVIEW.

In addition to the technical issues associated with COTS microprocessors and SoCs, regulatory compliance issues add developmental and operational risks. Current regulatory guidance has not kept pace with the rapid evolution of COTS devices. As Phase 1 of this research has concluded, DO-254 is not sufficient guidance for assuring the safety of COTS microprocessors and SoCs and is inappropriately focused on caching and WCET as a pointed solution.

Discussions held during Phase 4 concluded that, due to rapid technology advances, guidance adopted by aviation authorities should not be prescriptive in nature when addressing COTS device safety issues. However, the following COTS device aspects should be part of the safety evaluation process.

4.2 AVAILABILITY.

In many cases, product errata release dates, monitoring of publicly available device data, and market usage can signal potential obsolescence. The average lifespan of many devices is understood to be 18 months. Availability is commonly discussed in the context of product life and obsolescence.

4.3 SUITABILITY.

The focus of suitability is the evaluation of product features and their use in the targeted application. The following sections attempt to identify areas as appropriate where suitability risks may occur for safe use.

4.3.1 Trustworthiness of Manufacturer and Device.

Trustworthiness of the manufacturer is an attempt to evaluate the manufacturer's susceptibility of excluding change information from their errata data and the susceptibility of the manufacturer to move the product line from one fabrication location to another.

4.3.2 Predictable Operation.

Predictable operation is the initial assessment of the published design features to gain a comprehensive understanding of the product, their alignment to the intended application, and an assessment of the manufacturer's notification of changes that have occurred, such as a recorded change description in the errata data. This research has demonstrated that device manufacturers intentionally limit the amount and details associated with design and usage changes reflected in the errata data. In some cases, changes that could cause potential malfunctions are considered minor by the device manufacturers and therefore are not reported.

4.3.3 Tool Availability and Suitability.

Tool availability focuses on the timing of a new product introduced to the market and the actual maturity of the device. In many cases, the market sector becomes the beta tester of new products and tool availability may lag. Such is the case of Simics.

Tool suitability focuses on the comparison of how the tool is expected to be used as specified by the OEM and how the application is designed to use the tool. The difference may present a risk to discovering anomalous behavior.

4.3.4 Product Service History.

Product service history needs to be reviewed to ensure its applicability to the intended application.

The following aspects of intended use and functionality within a product service history must be considered:

- Functional overview
    - A COTS device can be viewed as a composition of many functions. A proper COTS device selection process seeks to provide a "best fit" between the application requirements and the functional characteristics of the candidate devices.
- Used functions
    - The identified used functions should be well understood to the extent possible for discovering potential anomalous behavior that may occur in the intended application.
- Unused functions
    - Any unused functionality within a COTS device must be proven disabled so that no unintended functional interference occurs. Standard approaches to disabling unused features include physical disconnection from the system and power disconnection. However, COTS devices may contain multiple unused features integrated on-chip that can only be disabled through software controls. This presents a new challenge to system designers in ensuring that unused functions are disabled and that they remain disabled and do not have an adverse affect on the system.

4.4 STABILITY.

The term stability is used in the context of stable, predictable operation where suitability is an evaluation of product feature to the intended application as discussed above. Product stability is viewed as a means of evaluating risks of unexpected behavior induced by the surrounding architecture; environmental influences, such as SEU rates, high-intensity radiated field, and lightning; and malfunctions inherent to the design. Stability risks can be a result of device die shrinks, timing issues due to internal device substrate noise, internal device energy coupling, and interface to intended circuitry. Along with availability and stability, the publicly available product errata data is the primary input to the evaluation process. Periodically monitoring the errata data for revisions and changes against the targeted application is necessary.

4.5 TESTABILITY.

Inherently, COTS are equivalent to a "black box" where the design details, design methodology, and verification methodology are not directly controlled by the integrator. The integrator attempts to use the product as-purchased and the design knowledge is limited to the I/O of the

device. For this reason, the demonstration of safe use is limited to the ability to effectively discover and test the product for safe behavior at the device I/O level. Testability risks and their mitigation are associated with

- sufficient available data to understand the intended functional behavior. This includes the designer's level of understanding of the intended architectural application measured against the behavior as provided by the microprocessor and SoC.

- sufficient test approach for the intended application and whether it is tested as intended by the manufacturer. The risk of testing the microprocessor and SoC differently than the manufacturer intended is that it may produce a false sense of security that the microprocessor and SoC operate as the design is documented. Application tests should augment and not circumvent the manufacturer test methodology. Evaluating the testing approach in the intended application against the method expected by the manufacturer helps to identify potential false assurance.

- confidence tests to discover hidden latent behavior. Confidence tests inherently contain a level of engineering judgment. The amount of time and resources to extensively verify safe use through physical tests would likely exceed the time available for effectively bringing the product to the market. Therefore, confidence testing is the process of evaluating all engineering (hardware and software), field, and user experiences to define the best possible tests that demonstrate no latent behaviors. This becomes more complicated when working with an architecture where behavior is individually controlled and integrated between hardware and software.

5. SAFETY NET METHODOLOGY.

5.1 OVERVIEW.

Because COTS devices must be treated as black boxes, current COTS evaluation guidance under DO-254 must be supplemented with a system-level approach. Because complete knowledge of the COTS device cannot be acquired, it must be assumed that the COTS device will malfunction and/or fail.

This research defines safety net, also referred to as architectural mitigation, as the ability to demonstrate and protect against unintended/misleading device behavior at a level above the microprocessor or SoC through an appropriate combination of board- and system-level architecture. Safety net is grounded in the premise that devices will fail and malfunction. Several techniques were evaluated during the research where concentrated effort was focused above the microprocessor/SoC level. This included, but was not limited to

- external monitoring of safety-related behavior.
- redundancy.
- external watchdogs.
- architectures that allow run-time correction.

Architecture is defined as a design that leverages all levels from the device to the aircraft and system interface for detecting, reporting, protecting against, and correcting unintended behavior. This approach is expected to use less effort, depend less on the device, and rely on stronger efforts at the architecture level. Safety net design is becoming a more complex, application-specific art that will be required to detect, resolve, and validate component failure in a run-time environment to required levels of availability and safety.

The next proposed phase of this project will investigate specific methods of constructing safety nets and determining adequate levels of protection for specific applications.

5.2 SYSTEM-LEVEL DESIGN ASSURANCE.

The term safety net has been selected to emphasize the need for a broader approach than is currently used today to demonstrate the safe use of microprocessors and SoCs. Today, microprocessor and SoC safety is addressed through specific domains largely independent and less coordinated to other domains. Currently, the software community has been the prominent domain for demonstrating safe use of COTS devices. Recently, the AEH domain assisted and made an attempt to support safe use through a COTS implementation and accountability approach. This approach considers safety issues similar to the topics in section 4. Even with common (standardized) industry-wide approaches, each solution needs to be tailored to the unique application. This study cautions against viewing safety as a cookie cutter approach. Safety nets (solutions) must provide the ability to protect against unintended/misleading behavior at a higher level through a combination of techniques.

A safety net solution needs to coordinate multiple domains, including system design architecture, safety engineering, software, AEH, reliability, and manufacturing. This study recognizes that in most cases, these domains are involved with a project. However, they may not be brought together to produce a safe, integrated solution specific to microprocessors and SoCs. This study and the use of the term safety net recognizes that a single domain solution is impractical and microprocessors and SoCs will malfunction in their life cycle.

The task of ascertaining the safety considerations of SoCs is further complicated by the variety in designs. Unlike COTS microprocessors, where the majority of features of interest are similar across microprocessors, SoC components tend to vary based on the product selected, further complicating safety analysis. Not only are the safety concerns due to the individual IP cores an issue, but interaction among them presents a verification challenge to system designers. Certain systems using SoCs may not require a particular on-chip core and would require the disabling of that core for safety reasons. These issues need to be considered when formulating a safety assessment process for SoCs.

5.3 VIRTUTECH SIMICS.

Virtutech has provided the AFE43 Project with a copy of Simics, models of the devices under test, including the MPC8572DS, and associated software tools. The AFE43 Project will evaluate Simics for use in hardware and software evaluation in Phase 5 in coordination with Virtutech support.

Virtutech Simics [6] is a full-system simulator capable of running both high-performance (functionally accurate) and slower (microarchitecturally accurate) simulations of popular COTS microprocessors and SoCs within a board-level environment [5].

6. ADDITIONAL APPROACHES.

6.1 OVERVIEW.

This section contains descriptions of additional architectural approaches, beyond the proposed safety net methodology, that were identified during this phase of research. Specifically, standard techniques to disable unused features and lock-step architectures were identified as partial, successful solutions to mitigating COTS microprocessor risks. However, the use of COTS SoC in safety-critical aerospace applications presents new risk classes, as described in section 3, which may reduce the effectiveness of these techniques in future designs. These additional approaches are presented in this section as a record of alternative techniques that were not investigated in detail during this phase of research.

6.2 FEATURE DETERMINATION AND DISABLING.

6.2.1 Determination.

The selection and use of features (also known as functions) heavily depends on the ability to demonstrate the behavior is fully predictable. The selection, use of, and disabling of functions is an architectural approach for controlling configuration (to include one or more functions interconnected within the device) and behavior at the device level. The hardware aspects of a silicon device itself are considered fully deterministic. Logic gates, logic cells, and electrical properties have the ability to be physically measured and tested. An architectural approach to safe use could be to construct functions with the ability to be physically tested.

For example, the Ethernet has two safety concerns that could be considered nondeterministic components. Nondeterminism is defined as the loss of ability to fully predict and demonstrate the features behavior under all foreseeable (known) conditions. Software throughput and resultant functional behaviors are examples of nondeterministic behaviors. Software concerns centers around timing and jitter. Timing is defined as the amount of elapsed time the software is expecting the task to take within the defined boundaries and can be nondeterministic. Hardware timing is the physical (actual) time it takes for the transistor circuit to operate per the device manufacturer, based on the specific device technology. Proper Ethernet operation is deterministic if the functional behavior can be tested and/or monitored repeatedly. Therefore, an example of externally (outside the specific device) monitoring the registers for correct data or monitoring a specific bit for unintended activation could be an architectural approach for meeting safety concerns and determinism.

6.2.2 Disabling.

SoC devices currently available include a variety of built-in functionality that can be enabled or disabled (and configured) to provide the subset of functions required for the system. It is unlikely that any specific application will require the use of all available functionality. Functions

may be explicitly disabled by the system designer or implicitly disabled (i.e., functions may be mutually exclusive based on the SoC design).

When configuring a device for use in a specific application, the safety net should include an assessment of the disabled features and their potential impact on safety. The items identified during the research to consider are:

- The mechanism(s) for disabling features.
- The effect on the hardware interface for a disabled function.
- Failure modes that might affect the device configuration.
- System effect if a disabled feature becomes enabled.
- System effect if the application attempts to access a disabled function.
- The potential effect of disabling a feature on shared resources (e.g., memory, other features, device pinouts).

6.3 LOCK-STEP ARCHITECTURES.

Lock-step involves two (or more) processors that run on the same time base and at any point during execution (in the absence of failures) will execute the exact same instruction/memory cycle in both processors. This means there is no nondeterminism in the execution of instructions or in the response to external interrupts. For instance, if a timer interrupt occurs for each processor within the same system clock, both processors will stop execution on the same instruction cycle, even if this execution is happening in L1 cache. This implies deterministic design within the processor that prohibits any asynchronous design in the external interrupt interface circuitry or in the core to external memory bus interfaces.

Frame-lock does the comparison at the I/O boundary, but in this case, this is the only place it can be done since the processors are not necessarily synchronous at an instruction level. This means frame-lock designs must wait for an application's execution frame to complete to be sure that both processors have executed all instructions and that the I/O in both processors should be the same. At the end of the application's execution frame, the data from both processors is then moved out into the system and compared.

Lock-step and frame-lock architectures are both good for detecting random hardware failures, including errors from SEU. They are cost effective since they can eliminate some complex software to detect hardware failures.

Additionally, using dissimilar processors and a voting scheme may be used to detect latent processor design errors or common mode defects. An example of a processor design error is a floating point unit that produces the wrong result from a mathematical operation. Lock-step and frame-lock architectures using identical processors cannot detect latent design errors or common mode defects since any erroneous results will occur identically in both processors.

7. CONCLUSIONS.

Phase 4 of this project identified the growing significance of system-on-a-chip (SoC) designs within commercial off-the-shelf (COTS) manufacturer device offerings. SoC designs present new challenges to ensuring the safety of devices within safety-critical aerospace applications. The high level of integration present in SoC devices reduces the observability and controllability of system components. Furthermore, standard approaches to disable unused features at the board level have become obsolete.

To better understand the technical and compliance issues surrounding the use of COTS SoCs, two commercially relevant devices were selected for detailed analysis with regard to safety: the Freescale MPC8572, a dual-core SoC, and the TI TMS320VC5470, a processor-DSP unit pair SoC. Numerous technical and compliance-related safety issues have been identified through evaluation of device characteristics documented in specification documents and errata notices.

This research has identified three fundamental safety issues with respect to the use of COTS SoC in aerospace applications:

- The effect of shared resources on timing
- Register status and configuration
- Visibility and debug

Beyond the technical aspects of COTS SoC safety, several compliance issues also exist:

- Availability
- Suitability
- Stability
- Testability

The use of a system-level safety net has been proposed to protect the system from unspecified or erroneous behavior occurring at the COTS device level. A safety net must assume that the COTS device will malfunction at some point within its operational lifetime, and the safety net must not allow all reasonable sources of error within the COTS device to propagate to the system level.

Phase 5 of this research project will study the three technical safety issues identified in this phase. Additionally, mitigation techniques and potential safety net architectures will also be studied.

8. REFERENCES.

1. RTCA/DO-178B, “Software Considerations in Airborne Systems and Equipment Certification,” 01 December 1992.

2. RTCA/DO-254, “Design Assurance Guidance for Airborne Electronic Hardware,” April 2000.

3. Freescale Semiconductor, MPC8572E PowerQuiCC III Integrated Host Processor Family Reference Manual, Rev. 2, May 2008.

4. Texas Instruments, TMS320VC5470 Fixed-Point Digital Signal Processor Rev. B, 09 December 2002.

5. Texas Instruments, TMS320VC5471/TMS320VC5470 Inter-Processor Communication, 10 June 2002.

6. Virtutech, http://www.virtutech.com, date last visited 11 April 2009.

APPENDIX A—FREESCALE MPC8572 BLOCK-LEVEL ANALYSIS

A.1 OVERVIEW.

This appendix serves as an example of a contemporary commercial off-the-shelf (COTS) system-on-a-chip (SoC) architecture, which includes a high level of hardware and software configurability. Registers that allow the reconfiguration of significant system behaviors are included in this analysis. Additionally, device subsystems, such as interfaces, memory subsystems, and interrupts, can be dynamically enabled and disabled through these registers. An analysis of feature control is also included.

A.2 INTRODUCTION.

The MPC8572E PowerQuiCC III Integrated Host Processor Family Reference Manual, Rev. 2, denoted as Reference for the rest of this appendix, is used as the primary source of information for this evaluation [A-1]. Each block will be studied with the following objectives:

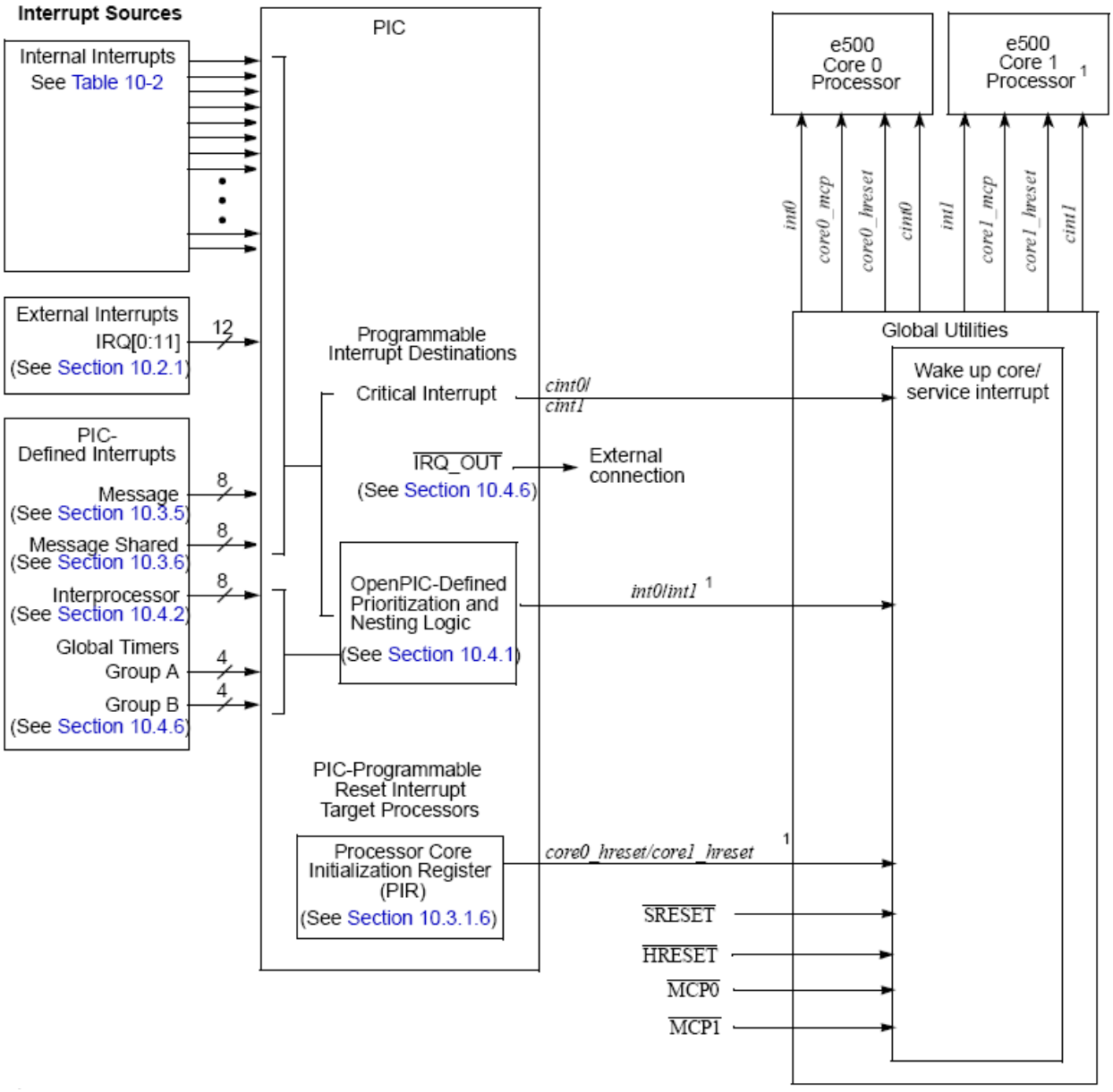
- Determine configuration of registers using memory map.
- List associated registers that are vulnerable to producing nondeterministic situations.
- Describe effects of unintentional register reconfiguration.
- Describe potential nondeterministic interactions with other blocks.
- If block is unused, describe effects of unintentional "wake-up" of that block.

A.3 PROGRAMMABLE INTERRUPT CONTROLLER (OpenPIC).

Chapter 10 of reference A-1 describes the structure and behavior of the OpenPIC block. The introduction provides the following overview:

> "The PIC conforms to the OpenPIC architecture. The interrupt controller provides multiprocessor interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to a CPU for servicing." [A-1]

Figure A-1 from reference A-1 is also included to illustrate the structure of the OpenPIC block and its relation to other blocks during normal interaction. Although not shown in figure A-1, it is important to note that at this time, it is understood that the OpenPIC block accesses the e500 cores through the e500 Coherency Module block. This understanding is based on the chip architecture figures in reference A-1 that shows the e500 Coherency Module as the interface between the cores and the peripheral blocks of the SoC.



[1]Not supported in single-processor implementations.
IRQ = Interrupt request
PIC = Programmable Interrupt Controller (also known as OpenPIC)
PIR = Processor-core initialization register
cint = Critical interrupt
int = Interrupt
SRESET = Soft reset
HRESET = Hard reset
MCP = Machine check processor
core = Microprocessor core

Figure A-1. Interrupt Sources Block Diagram [A-1]

A.4 OpenPIC INTERACTIONS.

A.4.1 OpenPIC—e500 CORE INTERACTION.

The OpenPIC block is responsible for sending interrupts (int and cint), reset (core_hreset), and machine check (core_mcp) signals to each e500 core. Additionally, the OpenPIC block can be used as an intermediary for interprocessor communication, and it is responsible for core initialization [A-1]. If the OpenPIC block produces erroneous behavior, the most apparent effects on the e500 core are the following:

- Unintentional reset of the core
- Unintentional interrupt of the core
- Erroneous interprocessor communication
- Erroneous core initialization

A.5 OpenPIC REGISTERS.

Currently, this document focuses on the OpenPIC block registers that contain read/write (R/W) fields. Read-only registers are assumed to not cause any unintentional or nondeterministic behaviors within the SoC.

A.5.1 PROCESSOR CORE INITIALIZATION REGISTER.

The Processor Core Initialization Register (PIR) (offset 0x1090) allows software to reset the e500 cores [A-1]. Because the MPC8572 contains only two e500 cores, only two bits of the PIR are used.

- Bit 30—Reset Processor 1
- Bit 31—Reset Processor 0

Erroneous writing to this register can cause one or both cores to be reset unintentionally.

A.5.2 GLOBAL CONFIGURATION REGISTER.

The Global Configuration Register (GCR) (offset 0x1020) allows software to reset the PIC and contains only two controllable fields [A-1]:

- Reset (RST), (bit 0)—"Cleared automatically when the reset sequence is complete" [A-1]

- Mode (M), (bit 2)—The Mode register sets the OpenPIC to Pass-through Mode or Mixed Mode

Erroneous writing to this register can cause the OpenPIC to be reset, or the interrupt mode of the OpenPIC to be changed. This, in turn, can cause the cores to be interrupted in a nondeterministic manner.

A.5.3 INTERPROCESSOR INTERRUPT VECTOR/PRIORITY REGISTERS.

The Interprocessor Interrupt Vector/Priority Registers (IPIVPR0-3) (offsets 0x10A0, 0x10B0, 0x10C0, and 0x10D0) "contain the interprocessor interrupt priority and vector for the four interprocessor interrupt channels." [A-1] These registers contain three controllable fields:

- Mask (MSK), (bit 0)—Disables (masks) interrupts on this channel if set to 1.
- Priority (PRIORITY), (bits 12-15)—Sets priority of this interrupt from 0 (disabled) to 15 (highest priority).
- Vector (VECTOR), (bits 16-31)—Contains the return value of the interrupt.

The Reference notes that the PRIORITY and VECTOR fields should not be modified while the Activity field is active [A-1].

Erroneous writing to this register can lead to nondeterministic interprocessor interrupts, especially if the fields are modified while an interrupt is in progress. Additionally, interrupts can be erroneously masked if the MSK bit is enabled.

A.5.4 SPURIOUS VECTOR REGISTER.

The Spurious Vector Register (SVR) (offset 0x10E0) contains the 16-bit return value of a spurious interrupt [A-1]. This return value is the only modifiable field of the SVR.

A.5.5 GLOBAL TIMER BASE COUNT REGISTERS.

There are eight Global Timer Base Count Registers (GTBCR): four registers for each timer, per two timer groups [A-1]. These registers control the frequency of interrupts per clock cycle. There are two controllable fields in the GTBCR:

- Count Inhibit (CI), (bit 0)—If set to 1, counting is inhibited for this timer.
- Base Count (BASE CNT), (bits 1-31)—This initializes the counting value in the corresponding Global Timer Current Count Register (GTCCR) register, which begins decrementing from this value.

Erroneous writing to this register could modify the frequency of interrupts, or disable interrupting on this timer altogether. This may lead to nondeterministic behavior of the system.

A.5.6 GLOBAL TIMER VECTOR/PRIORITY REGISTERS.

The Global Timer Vector/Priority Registers (GTVPR) are identical in structure and function to the IPIVPRs with the exception that these registers store return values corresponding to timer interrupt channels.

Erroneous writing to this register can lead to nondeterministic timer interrupts, especially if the fields are modified while an interrupt is in progress. Additionally, interrupts can be erroneously masked if the MSK bit is enabled.

A.5.7 TIMER CONTROL REGISTERS.

There are two Timer Control Registers (TCR): one per timer group. These registers control clock frequency, roll-over, and cascading behavior for each of the timers in the TCR's associated timer group [A-1]. There are four controllable fields within each TCR:

- Roll-Over (ROVR), (bits 5-7)—These determine whether the timers within the timer group roll-over to values set in the GTBCR or to all ones. Additionally, these can set cascading roll-over behavior, as shown in Tables 10-19 of reference A-1.

- Real-time mode (RTM), (bit 15)—This sets the clock source for the associated OpenPIC timer group. Either the e500 coherency module (ECM) or the Real Time Clock (RTC) is used.

- Clock ratio (CLKR), (bits 22-23)—If the ECM clock is used, this sets the clock divider used for the timers. The clock is divided by 8, 16, 32, or 64.

A.5.8 PERFORMANCE MONITOR MASK REGISTERS.

There are 12 Performance Monitor Mask Registers (PMMR): four sets of three 32-bit registers [A-1]. Each set forms a 96-bit mask vector that is initialized to all ones. Only one bit in each 96-bit vector may be set to zero—setting more than one bit to zero leads to nondeterministic behavior. These vectors control which interrupt source (interprocessor, timer, message, share message, external, internal) generates a performance monitor event.

Erroneous writing to these registers may result in nondeterministic behavior of the OpenPIC block or the performance monitor.

A.5.9 MESSAGE REGISTERS.

Eight Message Registers (MSGR) (offsets 0x1400, 0x1410, …, and 0x1470) store the 32-bit messages used in messaging interrupts [A-1]. Writing to MSGR signals messaging interrupts based on the settings of the other messaging interrupt registers. Reading the MSGR clears the messaging interrupt. The entire MSGR is a single field: the 32-bit message (MSG).

A.5.10 MESSAGE ENABLE REGISTERS.

The Message Enable Registers (MER) (offsets 0x1500 and 0x2500) control which of the eight messaging interrupts are enabled [A-1]. The Reference notes that each MER should be set to 0x0000_000F (all messaging interrupts enabled) at reset and left unchanged during normal operation [A-1]. Each MER contains four controllable fields:

- Enable 3/7 (E3/7), (bit 28)—Enable messaging interrupt 3 or 7 (depending on register offset).
- Enable 2/6 (E2/6), (bit 28)—Enable messaging interrupt 2 or 6 (depending on register offset).
- Enable 1/5 (E1/5), (bit 28)—Enable messaging interrupt 1 or 5 (depending on register offset).
- Enable 0/4 (E0/7), (bit 28)—Enable messaging interrupt 0 or 4 (depending on register offset).

Erroneous writing to the MER may result in inappropriate messaging interrupt enabling or masking, or may result in lost messaging interrupts if the MER is reconfigured during interrupt.

A.5.11 MESSAGE STATUS REGISTERS.

The Message Status Registers (MSR) (offsets 0x1510 and 0x2510) contain status bits for each messaging interrupt. Each status bit is enabled during an active associated messaging interrupt, and writing a 1 to a status bit clears the status bit and the associated messaging interrupt. Each MSR contains four controllable fields:

- Status 3/7 (S3/7), (bit 28)—Reports status of interrupt 3 or 7 (depending on register offset).
- Status 2/6 (S2/6), (bit 28)—Reports status of interrupt 2 or 6 (depending on register offset).
- Status 1/5 (S1/5), (bit 28)—Reports status of interrupt 1 or 5 (depending on register offset).
- Status 0/4 (S0/7), (bit 28)—Reports status of interrupt 0 or 4 (depending on register offset).

Erroneous writing to the MSR can lead to lost or unhandled messaging interrupts.

A.5.12 SHARED MESSAGE SIGNALED INTERRUPT INDEX REGISTER.

The Shared Message Signaled Interrupt Index Register (MSIIR) (offset 0x1740) controls shared message behavior by setting which Shared Message Interrupt Register (MSIR) and which interrupt field is used for each shared messaging interrupt [A-1]. The MSIIR has two controllable fields:

- Shared interrupt register select (SRS), (bits 0-2)—Selects the MSIR to be written (0 to 7).
- Interrupt bit select (IBS), (bits 3-7)—Selects which bit in the MSIR will be set (0 to 31).

A.5.13 SHARED MESSAGE/INTERNAL/EXTERNAL/MESSAGING VECTOR/PRIORITY REGISTERS.

These registers are identical in structure and function to the IPIVPR registers with the exception that these registers store return values corresponding to shared message, internal, external, or messaging interrupt channels [A-1].

Erroneous writing to these registers can lead to nondeterministic timer interrupts, especially if the fields are modified while an interrupt is in progress. Additionally, interrupts can be erroneously masked if the MSK bit is enabled.

A.5.14 SHARED MESSAGE/INTERNAL/EXTERNAL/MESSAGING INTERRUPT DESTINATION REGISTERS.

Each destination register contains destination fields for shared message, internal, external, or messaging interrupts. According to the Reference, setting more than one destination bit in a destination register may lead to undefined or nondeterministic behavior [A-1]. Each destination register contains five controllable fields:

- External signal (EP), (bit 0)—Enabling this bit forwards the interrupt to IRQ_OUT for external servicing.
- Critical interrupt 0 (CI0), (bit 1)—Enabling this bit forwards the interrupt to core 0 along its critical interrupt signal (cint0).
- Critical interrupt 1 (CI1), (bit 2)—Enabling this bit forwards the interrupt to core 1 along its critical interrupt signal (cint1).
- Processor core 1 (P1), (bit 30)—Enabling this bit forwards the interrupt to core 1 using the normal interrupt signal.
- Processor core 0 (P0), (bit 31)—Enabling this bit forwards the interrupt to core 0 using the normal interrupt signal.

A.5.15 INTERPROCESSOR INTERRUPT DISPATCH REGISTERS.

The four Interprocessor Interrupt Dispatch Registers (IPIDR0-3) (offsets 0x0040, 0x0050, 0x0060 and 0x0070) control which cores are interrupted for each interprocessor interrupt channel [A-1]. Neither one, nor both cores may be interrupted per channel. There are two controllable fields within each IPIDR:

- Processor core 1 (P1), (bit 30)—If set to 1, processor core 1 receives the interrupt.
- Processor core 0 (P0), (bit 31)—If set to 1, processor core 0 receives the interrupt.

Erroneous writing to these registers results in improper forwarding of interrupts to the processor cores. This can lead to erroneous or nondeterministic system behavior.

A.5.16 PROCESSOR CORE CURRENT TASK PRIORITY REGISTERS.

The Processor Core Current Task Priority Registers (CTPR0-1) (offsets 0x0080 and 0x1080) stores the value of the current task's priority running on the associated core [A-1]. It is the responsibility of the software running on each core to write each task's priority to this register. The OpenPIC block compares the value stored in the CTPR to the priority of each incoming interrupt; an interrupt is asserted to a core when its priority is higher than the task's priority and the interrupt status register (ISR). There is only one controllable field within each CTPR: Task priority (TASKP, bits 28-31).

Erroneously writing to these registers results in inappropriately asserted interrupts. This may lead to nondeterministic execution times of tasks running on each core.

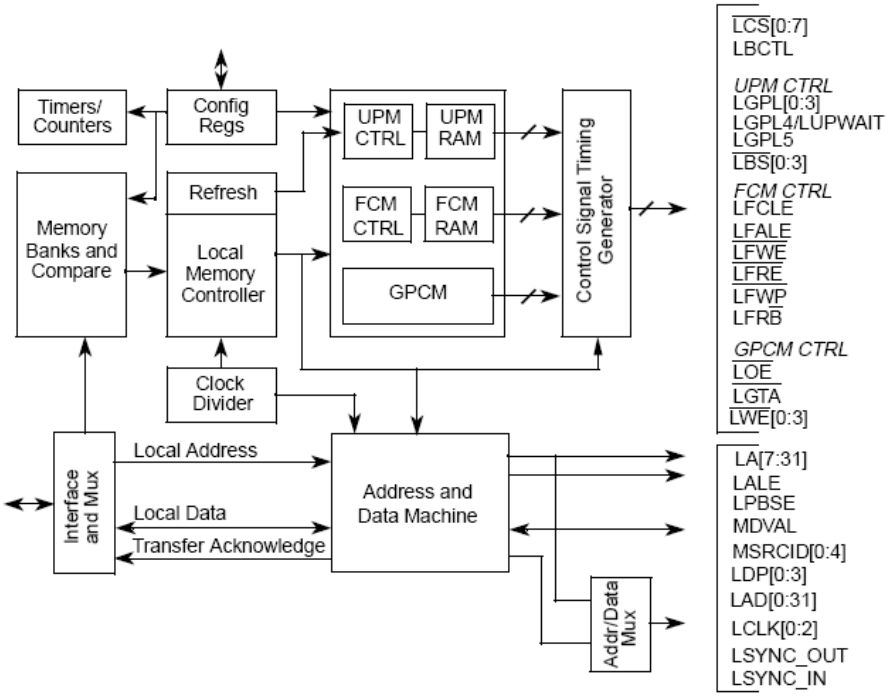
A.5.17 PROCESSOR CORE END OF INTERRUPT REGISTERS.

The Processor Core End of Interrupt Registers (EOI0-1) (offsets 0x00B0 and 0x10B0) allows the associated core to signal its completion in handling an interrupt [A-1]. The OpenPIC block assumes that the highest-priority interrupt is completed each time the core writes to the EOI register. There is only one controllable field within the EOI register: EOI CODE (bits 28-31), which signals EOI upon writing. The contents of EOI CODE are assumed to be zero and are ignored.

Erroneous writing to this register by the core results in incomplete interrupts being signaled as completed. This can lead to nondeterministic behaviors in the entire system.

A.6 ENHANCED LOCAL BUS CONTROLLER.

Chapter 14 of reference A-1 (page 805) describes the structure and behavior of the Enhanced Local Bus Controller (eLBC) block, as shown in figure A-2. The eLBC contains three separate machine types: one general-purpose chip-select machine (GPCM), one NAND flash control machine (FCM), and three user-programmable machines (UPM).



LCS = Local bus chip select
LBCTL = Local bus control
LGPL = Local bus GP line
LBS = Local byte select
LFCLE = Local bus flash command latch enable
LFALE = Local bus flash address latch enable
LFWE = Local bus flash write enable
LFRE = Local bus flash read enable
LFWP = Local bus flash write protect
LFRB = Local bus flash read-write busy
LOE = Output enable
LGTA = General purpose chip-select machine terminate added
LWE = Local bus write enable

LA = Local bus address
LALE = Local bus controller external address latch signal
LPBSE = Local bus parity byte select enable
MDVAL = Memory data valid
MSRCID = Memory debug source port IO
LDP = Local bus data parity
LAO = Multiplexed address/data bus A-2, see A.6.12
LCLK = Local bus clock
LSTNC = Local bus phase locked loop synchronization
LUPWAIT = Local bus UPM wait
GP04 = General-purpose chip-select machine
CTRL = Controller

Figure A-2. Enhanced Local Bus Controller Block Diagram [A-1]

Reference A-1 (page 806) describes the structure and function of the eLBC as follows:

> "The main component of the eLBC is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a GPCM, a FCM, and up to three UPMs. As such, it supports a minimal glue logic interface to SRAM, EPROM, NOR flash EEPROM, NAND flash EEPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. The external address latch signal (LALE) allows multiplexing of addresses with data signals to reduce the device pin count."

The eLBC also includes a number of data-checking and protection features, such as data parity generation and checking, write protection, and a bus monitor, to ensure that each bus cycle is terminated within a user-specified period.

A.6.1 THE eLBC INTERACTIONS.

The interactions of the eLBC are particularly important because a variety of external memory devices can interface with the MPC8572 through the eLBC. The eLBC can be used for booting the MPC8572, so any reset characteristics may be affected by eLBC behavior.

A.6.2 THE eLBC REGISTERS.

Currently, this document focuses on registers of the eLBC block that contain R/W fields. Read-only registers are assumed to not cause any unintentional or nondeterministic behaviors within the SoC.

A.6.3 BASE REGISTERS.

The Base Registers (BR) (bits 0-7) (offsets 0x5000, 0x5008, 0x5010, …, and 0x5038) maps local bus addresses to one of the three machines (UPM, FCM, and GPCM) of the eLBC. The default setting is to set BR0, and to clear BR1-7. The References states:

> "BR0 has its valid bit (V) set for RCWHR [ROMLOC] = LBC. Thus bank 0 is valid with port size (PS) configured from cfg_rom_loc[0:3] at power-on reset. M = 0 for MSEL of GPCM, 11 for MSEL of FCM at boot. All other base registers have all bits cleared to zero during reset." [A-1]

The BR contains the following controllable fields:

- Base address (BA), (bits 0-16)—Used for address comparison of bus transactions to determine if a bank is being accessed.
- Port size (PS), (bits 19-20)—Specifies the associated port as 8, 16, or 32 bits.
- Error checking (DECC), (bits 21-22)—Specifies the level of error checking enabled for a particular bank. Error checking can be disabled, parity checks can be used for the GPCM and UPM, and error correction codes (ECC) can be enabled for the FCM.
- Write protect (WP), (bit 23)—Can disable write accesses to a particular bank.
- Machine select (MSEL), (bits 24-26)—Specifies the type of machine (GPCM, FCM, or UPM) used for a particular bank.
- Atomic operation (ATOM), (bits 28-29)—Allows read-after-write (RAW) or write-after-read (WAR) atomic transactions between the bus master and the memory controller.
- Valid (V), (bit 31)—If set to 1, indicates to the system that the associated BR/option register (OR) (described next) pair are valid.

A.6.4 OPTION REGISTERS.

The OR (bits 0-7) (offsets 0x5004, 0x500c, 0x5014, …, and 0x503c) define the memory bank size for each attached external device and other attributes described below. Depending on the machine type selected for a particular bank (via the MSEL field of the associated BR), the OR fields are interpreted differently for that machine type (GPCM, FCM, and UPM) [A-1].

A.6.4.1 The GPCM Mode.

The OR contains the following controllable fields if the associated BR[MSEL] = GPCM mode.

- Address mask (AM), (bits 0-16)—Specifies the memory bank size for the particular bank (from 4 GB to 32 KB).

- Buffer control disable (BCTLD), (bit 19)—Disables the local bus control (LBCTL) signal during memory bank accesses.

- Chip select negation time (CSNT), (bit 20)—For GPCM-controlled banks, can negate LCSn and LWE signals normally, or a quarter-bus clock cycle early to accommodate for slow peripherals.

- Address to chip-select setup (ACS), (bits 21-22)—Controls the setup time delay of the LCSn signal relative to changes in the address signal. This can be triggered simultaneously, a quarter-bus cycle later, or a half-bus cycle later.

- Extra address to chip-select setup (XACS), (bit 23)—Allows for an extended delay between LCSn assertion and address changes allowed by the ACS.

- Cycle length (SCY), (bits 24-27)—Determines the number of wait states inserted into a bus cycle to control cycle length. Can insert 0 to 15 wait states per cycle.

- External address termination (SETA), (bit 28)—Allows accesses to be terminated internally or externally through the LGTA pin.

- Timing relaxed (TRLX), (bit 29)—If enabled, accommodates slow peripherals by adding a cycle between address and control signals, doubles the wait states set by the SCY, extends hold times on read accesses, and setting the LCSn and LWE signals earlier.

- Extended hold time on read accesses (EHTR), (bit 30)—Allows 1, 4, or 8 idle clock cycles to be inserted on read accesses.

- External address latch delay (EAD), (bit 31)—Allows extra bus cycles during latched external address accesses.

A.6.4.2 The FCM Mode.

The OR contains the following controllable fields if the associated BR[MSEL] = FCM mode.

- Address mask (AM), (bits 0-16)—Specifies the memory bank size for the particular bank (from 4GB to 32KB).

- Buffer control disable (BCTLD), (bit 19)—Disables the LBCTL signal during memory bank accesses.

- Chip select negation time (CSNT), (bit 20)—For GPCM-controlled banks, can negate LCSn and LWE signals normally, or a quarter-bus clock cycle early to accommodate for slow peripherals.

- Page size, buffer size, block size (PGS), (bit 21)—Supports two NAND flash EEPROM page, buffer, and block size configurations.

- Chip select to command time (CSCT), (bit 22)—Determines the amount of advance notice given to slow memories for LCSn assertion prior to any other bus activity. This can assert LCSn 1, 2, 4, or 8 cycles before any other command.

- Command setup time (CST), (bit 23)—Specifies the memory bank size for the particular bank (from 4GB to 32KB).

- Command hold time (CHT), (bit 24)—Determines the amount of time LFWE is asserted before any command, address, or data changes on the bus. This can assert LFWE 0.5, 1, 1.5, or 2 clock cycles before these other commands.

- Cycle length (SCY) in bus clocks, (bits 25-27)—Determines the number of wait states inserted into a bus cycle to control cycle length. This can insert 0 to 7 wait states per cycle.

- Read setup time (RST), (bit 28)—Determines the amount of time LFRE is asserted before any wait states. This can assert LFRE 0.5, 0.75, or 1 clock cycle before any wait states.

- TRLX, (bit 29)—If enabled, accommodates slow peripherals by adding a cycle between address and control signals, doubles the wait states set by the SCY, extends hold times on read accesses, and sets the LCSn and LWE signals earlier.

- Extended hold time on read accesses (EHTR), (bit 30)—Allows 1, 4, or 8 idle clock cycles to be inserted on read accesses.

A.6.4.2 The UPM Mode.

The OR contains the following controllable fields if the associated BR[MSEL] = UPM mode.

- AM (bits 0-16)—Specifies the memory bank size for the particular bank (from 4GB to 32KB).
- BCTLD (bit 19)—Disables the LBCTL signal during memory bank accesses.
- CSNT (bit 20)—For GPCM-controlled banks, can negate LCSn and LWE signals normally, or a quarter-bus clock cycle early to accommodate for slow peripherals.
- Burst inhibit (BI), (bit 23)—Specifies whether the particular bank can support burst memory accesses.
- TRLX, (bit 29)—If enabled, accommodates slow peripherals by adding a cycle between address and control signals, doubles the wait states set by the SCY, extends hold times on read accesses, and setting the LCSn and LWE signals earlier.
- EHTR (bit 30)—Allows 1, 4, or 8 idle clock cycles to be inserted on read accesses.
- EAD (bit 31)—Allows extra bus cycles during latched external address accesses.

Erroneous writing to the BR/OR pairs could disable any functionally correct interactions with external memory devices since they require specific transaction protocols to be maintained.

A.6.5 THE UPM MEMORY ADDRESS REGISTER.

The UPM Memory Address Register (MAR) (offset 0x0_5068) stores the address that will be sent to the external address bus signals during UPM operation.

The MAR contains only one controllable field:

- Address (A), (bits 0-31)—"Address that can be output to the address signals under control of the AMX bits in the UPM RAM word" [A-1]

Erroneous writing to the MAR can destroy data integrity of external devices since all transactions are based on the address stored in the MAR.

A.6.6 THE UPM MODE REGISTERS.

There are three UPM Mode Registers (MAMR offset 0x0_5070, MBMR offset 0x0_5074, and MCMR offset 0x0_5078) that control the three UPMs of the eLBC [A-1].

The Mode X mode register (MxMR) contains the following controllable registers:

- Refresh enable (RFEN), (bit 1)—Specifies whether the device controlled by the associated UPM requires refreshing.

- Command opcode (OP), (bits 2-3)—Determines what type of operation is executed when a memory access hits the associated bank.

  - "Normal operation" [A-1]

  - Write to UPM array—Write the contents of the UPM/FCM Data Register (MDR) into the location pointed to by the machine address (MAD). MAD is then incremented.

  - Read from UPM array—Read the contents of the location pointed to by the MAD into the MDR. MAD is then incremented.

  - Run pattern—Run the pattern written in the random access memory (RAM) array. The pattern starts at the location pointed to by the MAD until the last bit is set in the RAM word.

- Local bus UPM wait (LUPWAIT) polarity active low (UWPL), (bit 4)—"Sets the polarity of the LUPWAIT pin when in UPM mode" [A-1]. This can be set to either active high or active low.

- Address multiplex size (AM), (bits 5-7)—Determines the format of the address of the current cycle is output on the address pins.

- Disable timer period (DS), (bits 8-9)—Guarantees a minimum time between accesses to the same bank controlled by the associated UPM. The bus clock cycle can be disabled for 1, 2, 3, or 4 clock cycles.

- General line 0 control (G0CL), (bits 10-12)—Determines which address line is output to the local bus GP line (LGPL), 0 pin when the associated UPM is in control.

- LGPL4 output line disable (bit 13)—Determines how the LGPL4/LUPWAIT pin is controlled by the UPM array.

- Read loop field (RLF), (bits 14-17)—Determines how many times a loop defined by the associated UPM will be executed for burst or single read patterns. Can be run 1 to 16 times.

- Write loop field (WLF), (bits 18-21)—Determines how many times a loop defined by the associated UPM will be executed for burst or single write patterns. Can be run 1 to 16 times.

- Refresh loop field (TLF), (bits 22-25)—Determines how many times a loop defined by the associated UPM will be executed for refresh patterns. Can be run 1 to 16 times.

- Machine address (MAD), (bits 26-31)—RAM address pointed for the command to be executed. Incremented when the UPM is accessed and the OP is write or read.

Erroneous writing to the MxMR can affect the behaviors of their associated UPM. Because the MxMR controls external signal behavior, refresh timing, and special operations, it is likely that external devices would become nonfunctional if the MxMRs are faulty.

A.6.7 MEMORY REFRESH TIMER PRESCALER REGISTER.

The Memory Refresh Timer Prescaler Register (MRTPR) (offset 0x0_5084) is a configurable clock divider used by the UPM [A-1].

The MRTPR contains only one controllable field:

- Refresh timers prescaler (PTP), (bits 0-7)—Determines the system clock divider setting for the refresh timers. The maximum divider is 256.

Erroneous writing to the MRTPRs affect refresh timing of external devices, which may affect data integrity of those devices.

A.6.8 THE UPM/FCM DATA REGISTER.

The UPM/FCM Data Register (MDR) (offset 0x0_5088) contains the data written to or read from the RAM array for UPM read/write commands, or data to/from an external EEPROM device for FCM read/write commands [A-1]. The configurable fields of the MDR depend on whether it is used in UPM mode or FCM mode.

A.6.8.1 The UPM Mode.

The MDR contains only one controllable field during UPM mode:

- Data (D), (bits 0-31)—Data to be read/written during UPM operations.

A.6.8.2 The FCM Mode.

The MDR contains four controllable fields during FCM mode:

- Address/Data Byte 3 (AS3), (bits 0-7)—The fourth byte of address sent by a custom address write operation or the fourth byte of data read from a read status operation.

- Address/Data Byte 2 (AS2), (bits 8-15)—The third byte of address sent by a custom address write operation or the third byte of data read from a read status operation.

- Address/Data Byte 1 (AS1), (bits 16-23)—The second byte of address sent by a custom address write operation or the second byte of data read from a read status operation.

- Address/Data Byte 0 (AS0), (bits 24-31)—The first byte of address sent by a custom address write operation or the first byte of data read from a read status operation.

Erroneous writing to the MDR will affect data integrity between the external devices and the MPC8572.

A.6.9 SPECIAL OPERATION INITIATION REGISTER.

The Special Operation Initiation Register (LSOR) (offset 0x0_5090) allows software to trigger a special operation (defined by the MxMR[OP]) of a specified bank.

The LSOR contains only one controllable field:

- Bank on which a special operation is initiated (BANK), (bits 29-31)—Determines which bank (0 to 7) will initiated a special operation. The associated BRn[V] and OP fields must be set to activate a special operation.

Erroneous writing to the LSOR could inappropriately trigger a special operation to be executed on an external memory device. This can affect the data integrity of the external device.

A.6.10 THE UPM REFRESH TIMER.

The UPM Refresh Timer (LURT) (offset 0x050A0) controls the refresh timer frequency for any refresh-enabled (MxMR[RFEN] = 1) banks using a UPM.

The LURT contains only one controllable field:

- UPM refresh timer period (LURT), (bits 0-7)—With the MRTPR, sets the refresh timer period.

Erroneous writing to the LURT can affect the refresh characteristics of the eLBC, causing errors or disabling any interactions with an external memory device.

A.6.11 ERROR AND EVENT MANAGEMENT REGISTERS.

The five error and event management registers control the error-checking capabilities of the eLBC and report the cause and source of errors to software [A-1]. Three of the registers (transfer error status register (LTESR), LTEATR, and transfer error address register (LTEAR)) are write-1-clear registers, and are therefore not configurable. However, they will be briefly described in this section. The transfer error check disable register (LTEDR) and transfer error interrupt enable register (LTEIR) are the only configurable error and event management registers, and its fields will be described in more detail.

The LTESR (offset 0x0_50B0) reports which errors have occurred in the eLBC, such as time-outs or parity/ECC errors. This register is a write-1-clear register [A-1].

The LTEDR (offset 0x0_50B4) is a configurable register of the error and event management registers since it controls the amount of error reporting by the eLBC [A-1].

The LTEDR contains the following controllable fields:

- Bus monitor disable (BMD), (bit 0)—Enables or disables the bus monitor.
- FCM command time-out disable (FCTD), (bit 1)—Enables or disables FCM command time-out.
- Parity and ECC error checking disabled (PARD), (bit 2)—Enables or disables parity and ECC error checking.
- Write protect error checking disable (WPD), (bit 5)—Enables or disables write protect error checking.
- Write after read atomic error checking disable (WARA), (bit 8)—Enables or disables WARA error checking.
- Read after write atomic error checking disable (RAWA), (bit 9)—Enables or disables RAWA error checking.
- Chip select error checking disable (CSD), (bit 12)—Enables or disables chip select error checking.
- FCM command completion checking disable (CCD), (bit 31)—Enables or disables command completion checking.

The LTEIR (offset 0x0_50B8) enables or disables error reporting through the eLBC internal interrupt mechanism [A-1].

The LTEIR contains the following controllable fields:

- Bus monitor interrupt enable (BMI), (bit 0)—Enables or disables the bus monitor.
- FCM command time-out interrupt enable (FCTI), (bit 1)—Enables or disables FCM command time-out.
- Parity and ECC error-checking interrupt enable (PARI), (bit 2)—Enables or disables parity and ECC error checking.
- Write protect error-checking interrupt enable (WPI), (bit 5)—Enables or disables write protect error checking.

- Write after read atomic error checking interrupt enable (WARA), (bit 8)—Enables or disables WARA error checking.

- Read after write atomic error checking interrupt enable (RAWA), (bit 9)—Enables or disables RAWA error checking.

- Chip select error checking interrupt enable (CSI), (bit 12)—Enables or disables chip select error checking.

The LTEATR (offset 0x0_50BC) captures the attributes of an error that has occurred in the eLBC. These attributes include whether the transaction was a read or write transaction, the source identification (ID) of the transaction, parity error, and controller bank [A-1].

The LTEAR (offset 0x0_50C0) stores the address of the transaction, which resulted in an eLBC error. This register is not supported for FCM-controlled transactions [A-1].

If error reporting is required for a particular application, erroneous writing to the LTEIR or LTEDR could disable or enable error-reporting capabilities of the eLBC. If error reporting is not required by the application, errors occurring in these registers will have no affect on system correctness.

A.6.12 LOCAL BUS CONTROLLER REGISTER.

The local bus controller register (LBCR) contains the following controllable fields [A-1]:

- LDIS (bit 0)—Local bus is enabled/disabled.

  - Erroneous writing to this field can cause local bus enabled or disabled unintentionally.

- BCTLC (bits 8-9)—Defines the use of signal data buffer control (LBCTL) for GPCM or UPM accesses.

  - Erroneous writing to this field can change the mode of LBCTL for GPCM or UPM. This may lead to nondeterministic data communication on the local bus.

- AHD (bit 10)—Address hold disable. Remove part of hold time for signal multiplexed address/data bus (LAD) with respect to signal external address latch enable (LALE) to lengthen the LALE pulse [A-1].

  - Erroneous writing to this field results in timing error on the local bus inappropriately. This may lead to nondeterministic execution times.

- AS16 (bit 13)—Address shift for 16-bit port size.

    - Erroneous writing to this field can cause inappropriate address value. This may lead to nondeterministic behavior of the memory devices and the peripherals on the local bus.

- LPBSE (bit 14)—"Enables parity byte select on LGTA/LFRB/LGPL4/LUPWAIT /LPBSE signal." [A-1]

- EPAR (bit 15)—"Determines odd or even parity. Writing GPCM or UPM-controlled memory with EPAR = 1 and reading the memory with EPAR = 0 generate parity errors for testing." [A-1]

    - Erroneous writing to two fields (LPBSE and EPAR) results in testing error caused by wrong parity.

- BMT (bits 16-23)—"Bus monitor timing. Defines the bus monitor time-out period." [A-1]

- BMTPS (bit 28-31)—"Bus monitor timer prescale. Defines the multiplier, PS, to scale LBCR[BMT] for determining bus time-outs." [A-1]

    - A bus monitor is provided to ensure that each bus cycle is terminated within a reasonable (user defined) period. When a transaction starts, the bus monitor counts down from the time-out value (LBCR[BMT] × LBCR[BMTPS]) until a data beat is acknowledged on the bus. It is very important to ensure that the value of LBCR[BMT] is not set too low; otherwise spurious bus time-outs may occur during normal operation—resulting in incomplete data transfers. Therefore, erroneous writing to two fields (BMT and BMTPS) causes nondeterministic behavior on the entire system.

A.6.13 CLOCK RATIO REGISTER.

The Clock Ratio Register (LCRR) "sets the system clock to eLBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals." [A-1]

The LCRR contains the following controllable fields:

- PBYP (bit 0)—PLL bypass. "This bit should be set when using low bus clock frequencies." [A-1] (66 MHz or lower)

- EADC (bits 14-15)—"External address delay cycles of LCLK. Defines the number of cycles for the assertion of LALE." [A-1]

- CLKDIV (bits 27-31)—"System clock divider. Sets the frequency ratio between the system clock and the memory bus clock. The system clock can be equal to csb_clk or twice csb_clk (if RCWL[LBIUCM] is set." [A-1]

Erroneous writing to this register results in a timing error on the local bus inappropriately. This may lead to nondeterministic execution times or incorrect behavior of the entire system.

A.6.14 FLASH MODE REGISTER.

The local bus Flash Mode Register (FMR) controls global operation of the FCM [A-1]. The FMR contains the following controllable fields:

- CWTO (bits 16-19)—"Command wait time-out. For FCM commands that wait on LFRB being sampled high (CW0–CW3), FCM pauses execution of the instruction sequence until either LFRB is sampled high, or a timer controlled by CTO expires, whichever occurs first." [A-1]

  - Erroneous writing to this field results in timing error on the local bus inappropriately. This may lead to nondeterministic execution times.

- BOOT (bit 20)—"Flash auto-boot load mode. During system boot from NAND flash E2PROM, this bit remains set to alter the use of the FCM buffer RAM. Software should clear BOOT once FCM is to be restored to normal operation. Setting BOOT without auto-boot in progress only alters the mapping of the buffer RAM." [A-1]

  - Erroneous writing to this field can cause unintentional boot mode of FCM. This may lead to nondeterministic behavior of FCM.

- ECCM (bit 23)—"ECC mode. When hardware checking and/or generation of error correcting codes (ECC) is enabled (that is, when BRn[DECC] is 01 or 10, and full page transfers are specified with FBCR[BC] = 0), ECCM sets the ECC block size and position of the ECC code word(s) in the NAND flash spare region for both checking and generation functions." [A-1]

  - Erroneous writing to this fields result in unintentional error-correcting mode of FCM.

- AL (bits 26, 27)—"Address length. AL sets the number of address bytes issued during page address (PA) operations." [A-1]

  - Erroneous writing to this field can cause incorrect page address operation. Therefore, it may cause nondeterministic FCM behavior.

- OP (bits 30, 31)—"Flash operation. For OP not equal to 00, a special operation is triggered on the next write to LSOR or dummy access to a bank controlled by FCM. Once a special operation has commenced, OP is automatically reset to 00 by FCM. Individual blocks may be temporarily unlocked for erase and reprogramming operations." [A-1]

  - Erroneous writing to this field can cause unintentional flash operation. It may cause incorrect FCM behavior.

A.6.15 FLASH INSTRUCTION REGISTER.

Reference A-1 describes the Flash Instruction Register (FIR) as follows:

> "The local bus flash instruction register (FIR) holds a sequence of up to eight instructions for issue by the FCM. Setting FMR[OP] non-zero and writing LSOR or accessing a bank controlled by FCM causes FCM to read FIR 4 bits at a time, starting at bit 0 and continuing with adjacent 4-bit opcodes, until only NOP opcodes remain. The programmed instruction sequence of OP0, OP1, ..., OP7 is performed on the activated bank, using the data buffer addressed by FPAR. If LTEIR[CCI] = 1 and LTEDR[CCD] = 0, eLBC will generate an interrupt once the entire sequence has completed, and software should examine LTEATR and clear its V bit. Software must not alter the contents of the addressed FCM buffer, FIR, MDR, FCR, FBAR, FPAR, or FBCR while an operation is in progress—or eLBC will behave unpredictably—but software can freely modify the contents of any currently unused FCM RAM buffer in preparation for the next operation."

The FIR contains the following controllable fields:

- OP0 (bits 0-3)—"FCM operation codes. OP0 is executed first, followed by OP1, through to OP7." [A-1]
- OP1 (bits 4-7)
  - "0000 NOP: No-operation and end of operation sequence." [A-1]
  - "0001 CA: Issue current column address as set in FPAR, with length set by ORx[PGS]." [A-1]
- OP2 (bits 8-11)
  - "0010 PA: Issue current block+page address as set in FBAR and FPAR, with length set by FMR[AL]." [A-1]
  - "0011 UA: Issue user-defined address byte from next AS field in MDR." [A-1]
- OP3 (bits 12-15)—0100 CM0: "Issue command from FCR[CMD0]." [A-1]
- OP4 (bits 16-19)
  - "0101 CM1: Issue command from FCR[CMD1]." [A-1]
  - "0110 CM2: Issue command from FCR[CMD2]." [A-1]

- OP5 (bits 20-23)
  - "0111 CM3: Issue command from FCR[CMD3]." [A-1]
  - "1000 WB: Write FBCR bytes of data from current FCM buffer to flash device." [A-1]
- OP6 (bits 24-27)—"1001 WS: Write one byte (8b port) of data from next AS field of MDR to flash device." [A-1]
- OP7 (bits 28-31)
  - "1010 RB: Read FBCR bytes of data from flash device into current FCM RAM buffer." [A-1]
  - "1011 RS: Read one byte (8b port) of data from flash device into next AS field of MDR." [A-1]
  - "1100 CW0: Wait for LFRB to return high or time-out, then issue command from FCR[CMD0]." [A-1]
  - "1101 CW1: Wait for LFRB to return high or time-out, then issue command from FCR[CMD1]." [A-1]
  - "1110 RBW: Wait for LFRB to return high or time-out, then read FBCR bytes of data from flash device into current FCM RAM buffer." [A-1]
  - "1111 RSW: Wait for LFRB to return high or time-out, then read one byte (8b port) of data from flash device into next AS field of MDR." [A-1]

Erroneous writing to this register results in inappropriate timing and correctness of the FCM operation. This may lead to nondeterministic FCM operation.

A.6.16 FLASH COMMAND REGISTER.

Reference A-1 describes the Flash Command Register (FCR) as follows:

> "The local bus flash command register (FCR) holds up to four NAND flash EEPROM command bytes that may be referenced by opcodes in FIR during FCM operation. The values of the commands should follow the manufacturer's datasheet for the relevant NAND flash device."

The FCR contains the following controllable fields:

- CMD0 (bits 0-3)—"General purpose FCM flash command byte 0. Opcodes in FIR that issue command index 0 write CMD0 to the NAND flash command/data bus." [A-1]

- CMD1 (bits 4-7)—"General purpose FCM flash command byte 1. Opcodes in FIR that issue command index 1 write CMD1 to the NAND flash command/data bus." [A-1]

- CMD2 (bits 8-11)—"General purpose FCM flash command byte 2. Opcodes in FIR that issue command index 2 write CMD2 to the NAND flash command/data bus." [A-1]

- CMD3 (bits 12-15)—"General purpose FCM flash command byte 3. Opcodes in FIR that issue command index 3 write CMD3 to the NAND flash command/data bus." [A-1]

Erroneous writing to this register results in incorrect value of FCM flash command byte. It means that inappropriate flash command byte may be written to command/data/bus, which could lead to nondeterministic FCM operation.

A.6.17 FLASH BLOCK ADDRESS REGISTER.

The local bus Flash Block Address Register (FBAR) "locates the NAND flash block index for the page currently accessed." [A-1]

The FBAR contains the following controllable fields:

- BLK (bits 8-31)—"Flash block address. The size of the NAND flash, as configured in ORn[PGS] and FMR[AL], determines the number of bits of BLK that are issued to the E2PROM during block address phases." [A-1]

A.6.18 FLASH PAGE ADDRESS REGISTER.

The local bus Flash Page Address Register (FPAR) "locates the current NAND flash page in both the external NAND flash device and FCM buffer RAM." [A-1]

The FPAR contains the following controllable fields:

- PI (bits 17-21)—"Page index. PI indexes the page in NAND flash E2PROM at the current block defined by FBAR, and locates the corresponding transfer buffer in the FCM buffer RAM." [A-1]

- MS (bit 22)—"Main/spare region locator. In the case that FBCR[BC] = 0, MS is treated as 0." [A-1]

- CI (bits 23-31)—"Column index. CI indexes the first byte to transfer to/from the main or spare region of the NAND flash E2PROM and corresponding transfer buffer." [A-1]

Erroneous writing to these registers (FBAR and FPAR) can cause in unintentional operation of NAND flash. If some FCM operations get the address of the NAND flash block currently accessed from these registers, erroneous register values can indicate incorrect address values, which could lead to nondeterministic behavior on FCM.

A.6.19 FLASH BYTE COUNT REGISTER.

The local bus Flash Byte Count Register (FBCR) "defines the size of FCM block transfers for reads and writes to the NAND flash EEPROM." [A-1]

The FBCR contains the following controllable fields:

- BC (bits 20-31)—"Byte count determines how many bytes are transferred by the FCM during data read (RB) or data write (WB) opcodes. The first byte accessed in the NAND flash E2PROM is located by the FPAR register, and successive bytes are transferred until either BC bytes have been counted, or the end of the spare region of the currently addressed flash page has been reached." [A-1]

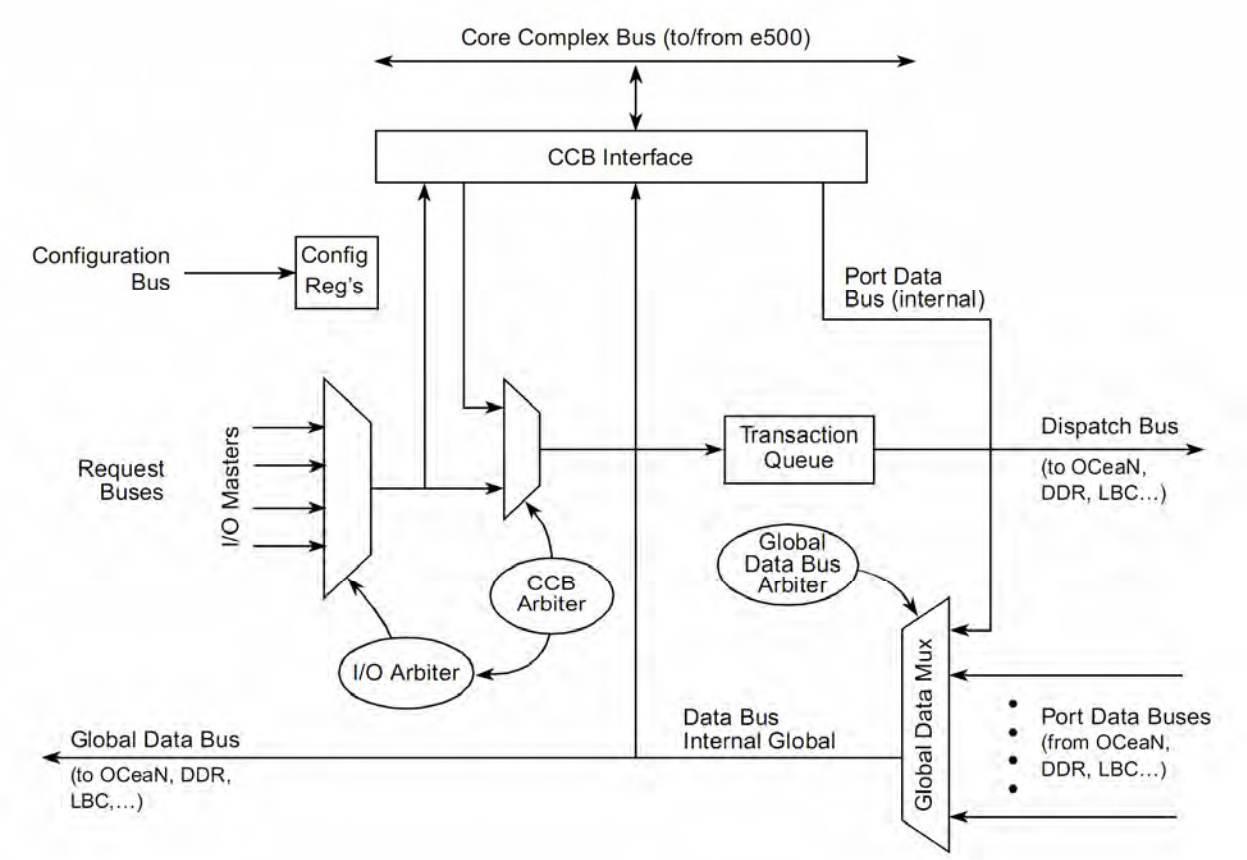
Erroneous writing to this register results in the incorrect size of FCM block read/write transfers to the NAND flash E2PROM. It means that inappropriate successive bytes can be transferred, which could lead to nondeterministic behavior on FCM.

A.7 THE e500 COHERENCY MODULE.

Chapter 8 of reference A-1 (page 419) describes the structure and behavior of the e500 coherency module (ECM) block. The introduction provides the following overview:

> "The ECM routes transactions initiated by the e500 cores to the appropriate target interface on the device. In a manner analogous to a bridging router in a local area network, the ECM forwards I/O-initiated transactions that are tagged with the global attribute onto the core complex bus (CCB). This allows on-chip caches to snoop these transactions as if they were locally initiated and to take actions to maintain coherency across cacheable memory."

A block diagram (from reference A-1) of the ECM block and its relation to other blocks during normal interaction is shown in figure A-3.



CCB = Core complex bus
Reg's = Registers
OCeaN = On-chip network
I/O = Input/output
DDR = Double data rate
LBC = Local bus controller

Figure A-3. The ECM Block Diagram [A-1]

A.7.1 INTERACTIONS OF THE ECM.

The ECM includes these distinctive features [A-1]:

- Support for two e500 cores and an L2/SRAM on the CCB, including a CCB arbiter.
- According to reference A-1, the ECM interacts with the two e500 cores, the L2 cache, the CCB, and the CCB arbiter. It also provides the e500 cores access to various on-chip I/O targets. These targets include the memory controllers, the local bus, PCIe and Ethernet controllers, and the configuration register access block.
- L2/SRAM and two 128-bit write data buses—one from each of the e500 cores.
- Four connection points for I/O initiating (mastering into the device) interfaces. The ECM supports five connection points for I/O targets. The double data rate (DDR) memory controllers, local bus, on-chip network (OCeaN) targets, and configuration register access block all have a target port connection to the ECM.

- Split transaction support—separate address and data tenures allow for pipelining of transactions and out-of-order data tenures between initiators and targets.

- Proper ordering of I/O-initiated transactions.

- Speculative read bus for low-latency dispatch of reads to the DDR controllers.

- Low-latency paths for returning read data from DDR to the e500 cores.

- Error registers trap transactions with invalid addresses. Errors can be programmed to generate interrupts to the e500 core.

If the ECM block produces erroneous behavior, the most apparent effects on the e500 core are the following:

- Unintentional streaming between the core and CCB
- Unintentional core arbitration
- Erroneous transaction assignment of the core

A.7.2 THE ECM REGISTERS.

This document focuses on registers of the ECM block that contain R/W fields. Read-only registers are assumed to not cause any unintentional or nondeterministic behaviors within the SoC.

A.7.3 THE ECM CORE COMPUTER BUS ADDRESS CONFIGURATION REGISTER.

The ECM CCB Address Configuration Register (EEBACR) "controls arbitration and streaming policies for the CCB." [A-1]

The EEBACR contains the following controllable fields:

- A_STRM_DIS (bit 28)—"Controls whether the ECM allows any streaming to occur." [A-1]

- CORE_STRM_DIS (bit 29)—"With A_STRM_DIS, controls whether the e500 cores can stream commands onto the CCB. A_STRM_DIS and CORE_STRM_DIS must both be cleared for the e500 cores to be enabled to stream their address tenures that they master." [A-1]

- A_STRM_CNT (bit 30, 31)—"Stream count. Specifies the maximum number of transactions that any master can stream (issue sequentially without preemption) on the CCB following an initial transaction." [A-1]

Erroneous writing to this register can cause streaming onto the CCB unintentionally. This can unexpectedly cause the cores to be enabled to stream their address tenures in a nondeterministic manner.

A.7.4 THE ECM CCB PORT CONFIGURATION REGISTER.

The ECM CCB Port Configuration Register (EEBPCR) contains the following controllable fields [A-1]:

- CPU1_EN (bit 6)—"CPU1 port enable. Controls boot holdoff mode when the device is an agent of an external host or when the other core is used to perform initialization prior to allowing this core to boot." [A-1]
  - "0 Boot holdoff mode. CPU1 arbitration is disabled on the CCB and no bus grants are issued." [A-1]
  - "1 CPU1 is enabled and receives bus grants in response to bus requests for the boot vector." [A-1]
- CPU0_EN (bit 7)—"CPU0 port enable. Controls boot holdoff mode when the device is an agent of an external host or when the other core is used to perform initialization prior to allowing this core to boot." [A-1]
  - "0 Boot holdoff mode. CPU0 arbitration is disabled on the CCB and no bus grants are issued." [A-1]
  - "1 CPU0 is enabled and receives bus grants in response to bus requests for the boot vector." [A-1]
- CPU1_PRI (bits 26, 27)—"Identifies the priority level of the e500 core 1 (CPU) port. This priority level is used to determine whether a particular port's bus request can cause the CCB Arbiter to terminate another port's streaming of address tenures." [A-1]
- CPU_RD_HI_DIS (bit 29)—"Identifies which read queue of DDR targets is assigned to the e500 core (CPU) ports' read transactions (in understressed system)." [A-1]
- CPU0_PRI (bits 30, 31)—"Specifies the priority level of the e500 core 0 (CPU) port. This priority level is used to determine whether a particular port's bus request can cause the CCB arbiter to terminate another port's streaming of address tenures." [A-1]

Erroneous writing to CPU1_EN or CPU0_EN can cause the cores arbitration to be enabled or disabled on the CCB unintentionally. In the case of CPU1_PRI or CPU0_PRI, erroneous writing can suddenly terminate another central processing unit (CPU) port's streaming of address tenures by changing the priority level. In addition, erroneous writing to CPU_RD_HI_DIS can change the read transaction assignment from DDR targets to the cores unintentionally. Therefore, erroneous writing to this register can cause nondeterministic behavior of the system.

A.7.5 THE ECM ERROR DETECT REGISTER.

The ECM Error Detect Register (EEDR) contains the following controllable fields [A-1]:

- MULT_ERR (bit 0)—"Multiple error. Indicates the occurrence of multiple errors of the same type. Write 1 to clear." [A-1]

- LAE (bit 31)—Local access error. Write 1 to clear. Two cases can generate LAEs:

  - "Transaction does not map to any target. In this case the ECM injects read responses (with the corrupt attribute set) and write data is dropped. Note that a read that attempts to access an unmapped target causes the assertion of core_fault_in, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, EEER[LAEE] must be set to ensure that an interrupt is generated." [A-1]

  - "Source and target IDs indicate that an OCN port initiated a transaction that targets an OCN port. This loopback behavior can result from programming errors where inbound ATMU window targets are inconsistent with targets configured in the local access windows for a given address range. For this type of LAE, the dispatch (to OCN target in this case) is not screened off; the LAE error is reported, but the transaction is still sent to its OCN target." [A-1]

Erroneous writing to this register can show unintentional error status.

A.7.6 THE ECM ERROR ENABLE REGISTER.

The ECM Error Enable Register (EEER) "enables the reporting of error conditions to the e500 core through the internal int interrupt signal." [A-1]

- LAEE (bit 31)—"Local access error enable. Note that a read that attempts to access an unmapped target causes the assertion of core_fault_in, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If HID1[RFXE] is zero and this error occurs, LAEE must be set to ensure that an interrupt is generated." [A-1]

Erroneous writing to this register can generate a machine check interrupt. This, in turn, can cause the cores to be interrupted in a nondeterministic manner.

A.8 REFERENCE.

A-1. Freescale Semiconductor, MPC8572E PowerQuiCC III Integrated Host Processor Family Reference Manual, Rev. 2, May 2008.