

Standard Operating Procedure

Document Number: GMAX-SYS-001

Standard Operating Procedure (SOP): Gigamax Cache Coherence Protocol Simulation

Revision: 1.0

Last Updated: [DATE]

Contents

1. Introduction
 - 1.1 Purpose
 - 1.2 System Overview
 - 1.3 Regulatory Compliance
2. System Specifications
 - 2.1 Coherence State Parameters
 - 2.2 Component Architecture
3. Operational Protocols
 - 3.1 Initialization and Startup Logic
 - 3.2 Master-Slave Command Arbitration
 - 3.3 Cache and Bus Interaction Rules
4. Emergency Operations
 - 4.1 Abort Conditions
 - 4.2 Recovery and Retry Mechanism
5. Maintenance Requirements
 - 5.1 State and Variable Verification
 - 5.2 Fault Reinitialization Procedure
6. Quality Assurance
 - 6.1 Exclusive Access Validation
 - 6.2 Read/Write Reachability Verification

- 7. Security Protocols
 - 7.1 Command Isolation per Processor
 - 7.2 Shared State Integrity
 - 8. Environmental Considerations
 - 8.1 Load Distribution Management
 - 8.2 Multi-Core Synchronization Timing
 - 9. Training Requirements
 - 9.1 Protocol Flow Understanding
 - 9.2 Cache State Debugging Techniques
 - 10. Document Control
 - 10.1 Revision History
 - 10.2 Authorization
 - 11. Process Flows and State Transitions
 - 11.1 Command Evaluation Logic
 - 11.2 Cache State Transition Conditions
 - 11.3 Bus Response Handling
-

1. Introduction

1.1 Purpose

- Simulate and validate the Gigamax multiprocessor cache coherence protocol.
- Prevent simultaneous write access to shared memory across processors.

1.2 System Overview

- A memory system with 3 processors (`p0` , `p1` , `p2`) and a central memory module.
- Uses command arbitration and coherence states to regulate access.
- Readable and writable status defined per cache state and bus wait logic.

1.3 Regulatory Compliance

- Aligned with MESI-like coherence model behavior for academic simulation.
- Designed to support verification under AG EF and AG properties.

2. System Specifications

2.1 Coherence State Parameters

- Cache States:
 - `invalid` : Not valid; must re-fetch data.
 - `shared` : Read-only and valid.
 - `owned` : Read/write access is permitted.
- Bus States:
 - `waiting` : Stall condition pending reply.
 - `abort` : Protocol-level reset due to error or deadlock.

2.2 Component Architecture

- Each processor module includes:
 - A local cache
 - A command generator
 - Bus arbitration logic
- Shared command variable `CMD` is evaluated across modules.

3. Operational Protocols

3.1 Initialization and Startup Logic

- All cache and bus states are initialized to `invalid` or `FALSE` .

- CMD defaults to `idle` until a master processor issues a command.

3.2 Master-Slave Command Arbitration

- Processor priority:
 - Only one processor may issue a command (`master = TRUE`).
 - Other processors remain idle or await bus status.
- Arbitration proceeds in sequence: `p0`, then `p1`, then `p2`, then `memory`.

3.3 Cache and Bus Interaction Rules

- A processor with `invalid` cache may issue `read-shared` or `read-owned`.
 - Owned cache blocks respond with `write-invalid`, `write-resp-shared`, or `write-resp-invalid` based on snoop status.
 - Memory responds with `response` only when not busy.
-

4. Emergency Operations

4.1 Abort Conditions

- Abort triggers when:
 - CMD is a read and reply is stalled or delayed (`REPLY-WAITING`).
 - A reply stall exists anywhere in the system.

4.2 Recovery and Retry Mechanism

- Upon `abort`, all CMD and state variables retain their values.
 - Processor attempts to reissue request based on state and CMD availability.
-

5. Maintenance Requirements

5.1 State and Variable Verification

- Periodic checks:
 - CMD validity and transition correctness
 - Cache state integrity under arbitration

5.2 Fault Reinitialization Procedure

- If deadlock or multi-master conflict is detected:
 - Reset all processors to `idle`
 - Reassign bus master starting with `p0`

6. Quality Assurance

6.1 Exclusive Access Validation

- Ensure `AG !(p0.writable & p1.writable)` holds at all times.
- No two processors can have `writable = TRUE` simultaneously.

6.2 Read/Write Reachability Verification

- Confirm `AG EF p0.readable` and `AG EF p0.writable`.
- Ensure any processor can eventually gain read/write access.

7. Security Protocols

7.1 Command Isolation per Processor

- Each processor's `cmd` must be isolated from others unless arbitration grants access.

7.2 Shared State Integrity

- Snoop and state transitions must occur without race conditions.
- No unauthorized updates to CMD outside of `master = TRUE` context.

8. Environmental Considerations

8.1 Load Distribution Management

- Monitor command patterns for overuse of specific cores.
- Rotate master assignment more frequently under high contention.

8.2 Multi-Core Synchronization Timing

- Account for clock skews in `reply-stall` and `waiting` evaluations.
- Synchronize memory and cache command phases.

9. Training Requirements

9.1 Protocol Flow Understanding

- Engineers must understand:
 - State transitions of caches
 - Arbitration flow logic
 - Read/write permission computation

9.2 Cache State Debugging Techniques

- Use logging tools to trace CMD and state over time.
- Isolate transitions leading to `abort` for root cause analysis.

10. Document Control

10.1 Revision History

- Rev 1.0 – Initial SOP derived from Gigamax formal specification (gigamax.txt)

10.2 Authorization

- Approved by: Multi-Core Simulation Lead
 - Reviewed by: Hardware Verification Committee
-

11. Process Flows and State Transitions

11.1 Command Evaluation Logic

- CMD transitions from `idle` to active states only when processor is master.
- Transition order prioritizes idle processors first.

11.2 Cache State Transition Conditions

- From `invalid` → `shared/owned` on reads.
- From `owned` → `shared/invalid` based on snoop result.
- `owned` → `invalid` on `write-invalid` or invalidation commands.

11.3 Bus Response Handling

- Memory responds with `response` only when not busy.
- Processors use `reply-stall`, `reply-waiting`, and `reply-owned` to determine aborts or continuations.