

Standard Operating Procedure

Document Number: PCI-SYS-001

Standard Operating Procedure (SOP): PCI Bus Protocol

Arbitration and Master Control

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 Arbitration Latency
 - 2.2 Bus Signal Definitions
 - 2.3 Equipment Configuration
3. Operational Protocols
 - 3.1 Standard Operating Procedures
 - 3.1.1 Arbitration Bank Logic
 - 3.1.2 Master Bus Access Flow
 - 3.2 Performance Optimization
 - 3.2.1 Latency Measurement
 - 3.2.2 Grant and Request Behavior
4. Emergency Operations
 - 4.1 Abort Scenarios
 - 4.2 System Recovery Behavior
5. Maintenance Requirements
 - 5.1 Preventive Maintenance Schedule
 - 5.2 Predictive Signal Conflict Handling

- 6. Quality Assurance
 - 6.1 Arbitration Fairness Tests
 - 6.2 Transaction Consistency Verification
 - 7. Security Protocols
 - 7.1 Control Logic Isolation
 - 7.2 Bus Access Integrity
 - 8. Environmental Considerations
 - 8.1 Bus Load Regulation
 - 8.2 Line Timing and Electrical Margin
 - 9. Training Requirements
 - 9.1 Personnel Competencies
 - 9.2 Simulation Exercises
 - 10. Document Control
 - 10.1 Revision History
 - 10.2 Authorization
 - 11. Process Flows and State Transitions
 - 11.1 Arbitration Decision Tree
 - 11.2 Master State Machines
 - 11.3 Abort Logic and Re-entry Control
-

1. Introduction

1.1 Purpose

To ensure reliable, fair, and low-latency bus access among PCI components through structured arbitration and master transaction control.

1.2 System Overview

- The PCI protocol manages shared access to a bus between multiple requesters (masters).

- Arbitration logic selects which requester gains access using Fixed Priority (FP) or Round Robin (RR) policies.
- Modules include banked arbitration units and masters that issue memory or I/O transactions.

1.3 Regulatory Compliance

- Complies with PCI Local Bus Specification standards.
 - Incorporates fairness constraints to avoid starvation.
-

2. System Specifications

2.1 Arbitration Latency

- **REQ** → **GNT**: Arbitration latency (time between request and grant)
- **GNT** → **FRAME**: Bus acquisition latency
- **FRAME** → **TRDY**: Target latency

2.2 Bus Signal Definitions

- `frame` : Starts transaction
- `irdy` : Master ready
- `trdy` : Target ready
- `ad` : Address/data lines
- `c_bd` : Command and byte enable
- `req/gnt` : Request and grant handshake

2.3 Equipment Configuration

- Hierarchical arbiter with multiple banks (2in, 3in)
 - Master modules with control FSMs and transaction buffers
 - Passive master-null slots emulate inactive devices
-

3. Operational Protocols

3.1 Standard Operating Procedures

3.1.1 Arbitration Bank Logic

- Bank modules decide which request to grant based on:
 - Current `last` grant
 - Policy (FP or RR)
 - Presence of valid requests
- Bank2 aggregates sub-bank outputs to determine global grant

3.1.2 Master Bus Access Flow

- Request (`req`) is raised until grant (`gnt`) is received
- Upon grant:
 - Master enters `address` phase (assert `frame`)
 - Proceeds to `data` phase, sending/receiving lines
 - Ends when internal counter `_count = 0`

3.2 Performance Optimization

3.2.1 Latency Measurement

- Use signals like `start_transaction` , `end_transaction` , and grant timing
- Evaluate:
 - Arbitration latency
 - Bus acquisition time
 - Target readiness delay

3.2.2 Grant and Request Behavior

- Prevent starvation via Round Robin arbitration
 - Master retains `req` until transaction begins or aborted
-

4. Emergency Operations

4.1 Abort Scenarios

- If `abort_random` is active during `b_frame`, trigger abort
- `abort_count` tracks consecutive abort events (max 3)

4.2 System Recovery Behavior

- Reset transaction parameters on abort
 - Restart arbitration and request issuance as needed
-

5. Maintenance Requirements

5.1 Preventive Maintenance Schedule

- Validate grant fairness via test scenarios
- Reset internal states for `last`, `count`, and `state` on each boot

5.2 Predictive Signal Conflict Handling

- Monitor overlap between `irdy` and `trdy`
 - Detect turnaround cycles using timing offsets
-

6. Quality Assurance

6.1 Arbitration Fairness Tests

- Check starvation prevention using `AF_grant = X` for each requester

6.2 Transaction Consistency Verification

- Ensure `start_transaction` always leads to `end_transaction`

- Validate transitions: idle → address → data → idle
-

7. Security Protocols

7.1 Control Logic Isolation

- Policy registers (FP/RR) remain static post-init
- Isolation between arbitration levels via grant signals

7.2 Bus Access Integrity

- Ensure requests reflect true master intent
 - Avoid false deassertions due to premature `last` updates
-

8. Environmental Considerations

8.1 Bus Load Regulation

- Limit simultaneous request assertions
- Space out transactions to minimize contention

8.2 Line Timing and Electrical Margin

- Insert turnaround cycles on read/write transitions
 - Ensure signal integrity through alignment of `irdy` / `trdy`
-

9. Training Requirements

9.1 Personnel Competencies

- Understanding of PCI protocol and arbitration logic
- Familiarity with state machine transitions and transaction timing

9.2 Simulation Exercises

- Execute trace-based simulations with variable load
- Validate recovery from forced aborts and re-entry into arbitration

10. Document Control

10.1 Revision History

- Rev 1.0 – Initial SOP generated from formal model specification (pci.txt)

10.2 Authorization

- Approved by: Lead Hardware Architect
- Reviewed by: Systems Verification Team

11. Process Flows and State Transitions

11.1 Arbitration Decision Tree

- Evaluate request presence and policy
- Grant priority based on last-used and fairness mode

11.2 Master State Machines

- `idle` → wait for grant
- `address` → assert `frame`, prepare `c_bd`
- `data` → count data cycles using `_count`, assert `irdy`
- Return to `idle` upon transaction end

11.3 Abort Logic and Re-entry Control

- Abort asserted if `abort_random` & active frame

- Recover by resetting counters and restarting req