# AMBA Bus Matrix Configuration Tool

Complete User Guide and Reference Manual

**Version 2.1.0**

July 2025

 Complete 90+ Page Guide

# Table of Contents

# 1. Getting Started

## Welcome

Welcome to the AMBA Bus Matrix Configuration Tool, your comprehensive solution for
designing and generating AMBA-compliant interconnect systems.

### WHAT IS THIS TOOL?

The AMBA Bus Matrix Configuration Tool is a graphical design environment that enables
hardware architects and verification engineers to:

- Design complex multi-master, multi-slave bus systems
- Generate synthesizable RTL for FPGA and ASIC implementation
- Create complete UVM-based verification environments
- Ensure protocol compliance and optimal performance
- Reduce design time from weeks to hours

### SUPPORTED PROTOCOLS:

- AXI4 (ARM Advanced eXtensible Interface v4)
  - Latest AMBA protocol with enhanced features
  - Support for QoS, regions, and user signals
  - Up to 256 burst length
  - Recommended for new designs
- AXI3 (ARM Advanced eXtensible Interface v3)
  - Legacy AMBA protocol support
  - Write interleaving capabilities
  - Maximum 16 burst length
  - For existing IP integration
- AHB (Advanced High-performance Bus)
  - Single master protocol

# 1.1 System Requirements

## MINIMUM SYSTEM REQUIREMENTS:

Hardware Requirements:
• Processor: Intel/AMD x64 or ARM64 2.0 GHz dual-core
• Memory: 4 GB RAM (8 GB recommended for large designs)
• Storage: 2 GB free disk space
• Display: 1280x720 resolution (1920x1080 recommended)
• Mouse: Standard 3-button mouse with scroll wheel

Operating System Support:
• Linux: Ubuntu 18.04+, CentOS 7+, RHEL 7+, SUSE 15+
• Windows: Windows 10/11 (with WSL2 for best compatibility)
• macOS: 10.14 Mojave or later

Software Dependencies:
• Python 3.6, 3.7, 3.8, 3.9, or 3.10
• Tkinter GUI framework (usually included with Python)
• Git version control system
• Text editor (VS Code, Emacs, or Vim recommended)

## RECOMMENDED DEVELOPMENT ENVIRONMENT:

For RTL Generation:
• SystemVerilog simulator:
  - Synopsys VCS 2019.06 or later
  - Mentor Graphics Questa 2021.1 or later
  - Cadence Xcelium 20.09 or later
  - Xilinx Vivado Simulator 2020.2 or later

For Synthesis:
• Synthesis tools:
  - Synopsys Design Compiler
  - Cadence Genus
  - Xilinx Vivado Synthesis

# 1.2 Installation

**INSTALLATION PROCEDURE:**

Step 1: Download the Tool

Method A - Git Clone (Recommended):

```
git clone https://github.com/amba/bus-matrix-tool.git
cd bus-matrix-tool
git checkout main
```

Method B - Download Release:

```
wget https://github.com/amba/bus-matrix-tool/releases/latest/download/amba-tool.tar.gz
tar -xzf amba-tool.tar.gz
cd amba-tool
```

Step 2: Navigate to GUI Directory:

```
cd axi4_vip/gui
```

Step 3: Install Python Dependencies:

```
# Check Python version first
python3 --version

# Install requirements
pip3 install -r requirements.txt

# Manual installation if requirements.txt fails
pip3 install tkinter matplotlib numpy pillow
```

Step 4: Verify Tkinter Installation:

```
python3 -c "import tkinter; print('Tkinter: OK')"
python3 -c "import matplotlib; print('Matplotlib: OK')"
python3 -c "import numpy; print('NumPy: OK')"
```

Step 5: Set Permissions:

```
chmod +x launch_gui.sh
chmod +x generate_bus.sh
chmod +x scripts/*.sh
```

Step 6: Create Desktop Shortcut (Optional):

# 1.3 First Launch

## LAUNCHING THE APPLICATION:

Command Line Launch:
cd /path/to/amba-tool/axi4_vip/gui
./launch_gui.sh

Alternative Launch Methods:
python3 src/bus_matrix_gui.py
python3 -m bus_matrix_gui

Launch with Options:
./launch_gui.sh --debug          # Enable debug output
./launch_gui.sh --template simple # Load template
./launch_gui.sh --config my.json  # Load configuration

## FIRST LAUNCH CHECKLIST:

✓ Application window opens without errors
✓ Main menu bar is visible (File, Edit, View, Tools, Generate)
✓ Toolbar shows all buttons clearly
✓ Design canvas displays grid
✓ Properties panel is on the right
✓ Status bar shows "Ready"

Common First Launch Issues:

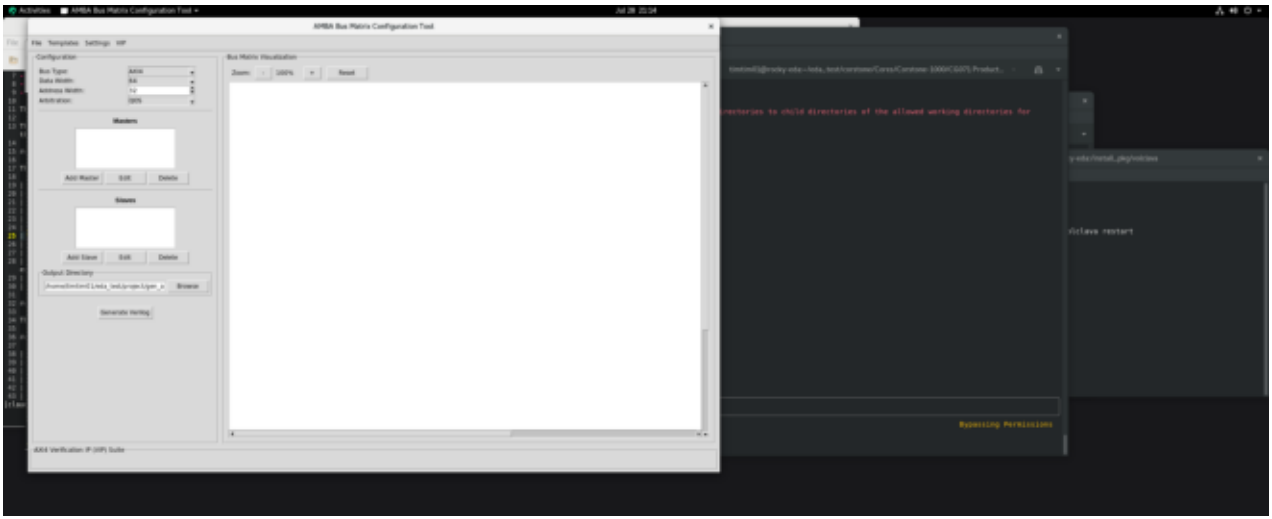
Issue: "Display not found"
• Solution: For SSH: ssh -X username@hostname
• Alternative: Use VNC viewer for remote access

Issue: "Permission denied"
• Solution: chmod +x launch_gui.sh
• Check: ls -la launch_gui.sh

Issue: GUI appears but fonts are tiny

# 1.4 Main Interface Screenshot

 Real Screenshot: gui_main_window.png



The main application window showing the menu bar, toolbar, design canvas with grid, properties panel, and status bar. This is the primary workspace where you'll design your bus matrix systems.

# 1.5 GUI Layout Details

### UNDERSTANDING THE GUI LAYOUT:

Menu Bar (Top):
• File: New, Open, Save, Recent, Export, Exit
• Edit: Undo, Redo, Cut, Copy, Paste, Select All
• View: Zoom, Grid, Connection Matrix, Address Map
• Tools: Validate, Preferences, Performance Analysis
• Generate: Generate RTL, Generate VIP, Batch Export
• Help: User Guide, About, Check Updates

Toolbar (Below Menu):
• New Project (Ctrl+N)
• Open Project (Ctrl+O)
• Save Project (Ctrl+S)
• Add Master (+M)
• Add Slave (+S)
• Add Bridge
• Delete Selected (Del)
• Validate Design (Ctrl+V)
• Generate RTL (Ctrl+G)
• Generate VIP (Ctrl+Shift+G)

Design Canvas (Center):
• Grid background for alignment
• Drag and drop component placement
• Visual connection routing
• Zoom and pan capabilities
• Selection and multi-select
• Copy/paste functionality

Properties Panel (Right):
• Component configuration
• Connection settings

# 1.6 Configuration Basics

## BASIC CONFIGURATION CONCEPTS:

Project Structure:
A project contains:
• Global settings (protocol, widths, clocking)
• Master components (initiators)
• Slave components (targets)
• Connections (routing paths)
• Validation rules
• Generation settings

Component Types:

Masters (Initiators):
• Generate transactions
• Examples: CPU cores, DMA controllers, GPUs
• Properties: ID width, outstanding transactions, protocols
• Connections: Output ports to slave input ports

Slaves (Targets):
• Respond to transactions
• Examples: Memory controllers, peripherals, bridges
• Properties: Address range, latency, memory type
• Connections: Input ports from master output ports

Bridges:
• Protocol converters
• Examples: AXI-to-APB, AXI-to-AHB
• Properties: Buffer depth, clock domains
• Connections: Master side and slave side

Address Mapping:
• Each slave occupies an address range
• No overlaps allowed (validation catches this)

# 1.7 Project Management

**PROJECT MANAGEMENT:**

Creating New Projects:
File → New Project opens dialog:
• Project Name: Descriptive identifier
• Location: Directory for project files
• Template: Starting point (Empty, 2×2, 4×4, Custom)
• Bus Protocol: AXI4/AXI3/AHB/APB
• Description: Optional documentation

Project File Structure:

```
my_project/
├── my_project.amba          # Main project file
├── config/
│   ├── global_settings.json
│   ├── masters.json
│   └── slaves.json
├── generated/
│   ├── rtl/
│   └── vip/
├── docs/
│   ├── design_spec.pdf
│   └── user_notes.txt
└── backups/
    ├── my_project_backup1.amba
    └── my_project_backup2.amba
```

Saving and Loading:
• Auto-save every 5 minutes (configurable)
• Manual save: Ctrl+S
• Save As: Shift+Ctrl+S
• Load: Ctrl+O or File → Recent

# 1.8 Keyboard Shortcuts

**KEYBOARD SHORTCUTS AND HOTKEYS:**

```
File Operations:
Ctrl+N          New Project
Ctrl+O          Open Project
Ctrl+S          Save Project
Ctrl+Shift+S    Save As
Ctrl+Q          Quit Application
Ctrl+Z          Undo
Ctrl+Y          Redo (Ctrl+Shift+Z on Mac)

Design Operations:
Ctrl+A          Select All Components
Ctrl+C          Copy Selected
Ctrl+V          Paste (Note: Also validates design)
Delete          Delete Selected Components
Ctrl+D          Duplicate Selected
Ctrl+G          Generate RTL
Ctrl+Shift+G    Generate VIP

View Operations:
Ctrl++          Zoom In
Ctrl+-          Zoom Out
Ctrl+0          Zoom to Fit
F11             Toggle Full Screen
Ctrl+R          Refresh Canvas
Tab             Next Component
Shift+Tab       Previous Component

Navigation:
Arrow Keys      Move Selected Component
Shift+Arrows    Move Selected 10 pixels
Ctrl+Arrows     Move Selected to Grid
Page Up/Down    Scroll Canvas Vertically
Home/End        Scroll Canvas Horizontally
```

# 1.9 Getting Help

### GETTING HELP AND SUPPORT:

Built-in Help System:
• Help → User Guide: This complete manual
• Help → Quick Start: 10-minute tutorial
• Help → Keyboard Shortcuts: Reference card
• Help → About: Version and license info
• Context Help: F1 key or ? button

Tooltips and Hints:
• Hover over any button for tooltip
• Status bar shows context-sensitive help
• Properties panel includes field descriptions
• Validation messages provide specific guidance

Documentation Resources:
• User Guide (this document): Complete reference
• API Documentation: For scripting and automation
• Protocol Specifications: AMBA standards compliance
• Example Projects: Located in examples/ directory
• Video Tutorials: Available on project website

Online Resources:
• Project Website: https://amba-tool.org
• Community Forum: https://forum.amba-tool.org
• Bug Reports: https://github.com/amba/issues
• Feature Requests: https://github.com/amba/discussions
• Updates: https://amba-tool.org/downloads

Training and Education:
• Interactive Tutorial: Help → Interactive Tutorial
• Webinar Series: Monthly design sessions
• University Courses: Academic licensing available

# 2. Complete Workflow

## Overview

### COMPLETE DESIGN WORKFLOW:

This section provides a comprehensive walkthrough of designing a complete bus matrix
system from initial concept to final generated files. We'll use a realistic example
throughout all steps.

PROJECT EXAMPLE: Automotive SoC Design

System Requirements:
• Dual ARM Cortex-A78 cores for applications
• ARM Cortex-R52 for real-time control
• GPU for HMI and instrument cluster
• DMA controller for data movement
• DDR4 memory controller (2GB)
• LPDDR4 for GPU (1GB)
• Flash controller for boot/config
• Ethernet controller for connectivity
• CAN bus interface for automotive
• Peripheral cluster (UART, SPI, I2C, GPIO)

Performance Requirements:
• CPU cores: 1.2 GHz operation
• GPU: 800 MHz with high bandwidth to LPDDR4
• Real-time: <10µs interrupt latency
• Boot: <2 seconds from flash
• Network: 1 Gbps Ethernet throughput

### WORKFLOW PHASES:

Phase 1: Project Planning and Setup (Pages 15-17)

# 2.1 Requirements Analysis

## PHASE 1: PROJECT PLANNING AND SETUP

2.1 Requirements Analysis

Before starting the design, analyze system requirements:

Performance Requirements:
• Bandwidth: Calculate peak data rates
  - CPU cores: 2 × 64 bits @ 1.2 GHz = 19.2 GB/s peak
  - GPU: 256 bits @ 800 MHz = 25.6 GB/s peak
  - DMA: 128 bits @ 400 MHz = 6.4 GB/s peak
  - Total peak: 51.2 GB/s (unrealistic simultaneous)
  - Realistic sustained: 15-20 GB/s

• Latency: Define maximum acceptable delays
  - CPU instruction fetch: <5 cycles
  - CPU data access: <10 cycles average
  - GPU texture fetch: <20 cycles
  - Real-time controller: <50ns interrupt response

• Concurrency: Outstanding transaction requirements
  - CPU cores: 16 outstanding each (L1 cache misses)
  - GPU: 64 outstanding (many parallel threads)
  - DMA: 8 outstanding (streaming transfers)
  - RT controller: 4 outstanding (predictable)

Address Space Planning:
0x00000000 - 0x7FFFFFFF  DDR4 Main Memory (2GB)
0x80000000 - 0xBFFFFFFF  LPDDR4 GPU Memory (1GB)
0xC0000000 - 0xC0FFFFFF  Flash Controller (16MB)
0xC1000000 - 0xC1FFFFFF  Ethernet Controller (16MB)
0xC2000000 - 0xC2FFFFFF  CAN Controllers (16MB)
0xC3000000 - 0xC3FFFFFF  Peripheral Cluster (16MB)
0xF0000000 - 0xFFFFFFFF  Internal SRAM/ROM (256MB)

# 2.2 Architecture Decisions

2.2 Architecture Decisions

Based on requirements analysis, make key architectural decisions:

Interconnect Topology:
Decision: Hybrid topology
• High-bandwidth crossbar for CPU/GPU/DDR
• Shared bus for peripherals via bridge
• Dedicated paths for real-time traffic

Rationale:
• Crossbar provides maximum bandwidth and minimum latency
• Shared peripheral bus reduces area and power
• Real-time path ensures deterministic response

Master Configuration Decisions:

CPU Cores (2x):
• ID Width: 6 bits (64 outstanding transactions)
• Data Width: 128 bits (cache line size)
• Exclusive Access: Enabled (atomic operations)
• QoS: Configurable (0-15, typically 8-12)

**GPU:**
• ID Width: 8 bits (256 outstanding transactions)
• Data Width: 256 bits (high bandwidth)
• Exclusive Access: Disabled (not needed)
• QoS: High (12-15) for frame deadlines

DMA Controller:
• ID Width: 4 bits (16 outstanding)
• Data Width: 128 bits (efficient transfers)
• Exclusive Access: Disabled
• QoS: Medium (4-8) configurable by software

# 2.3 Tool Configuration

2.3 Tool Configuration and Project Setup

Now implement the architectural decisions in the tool:

Step 1: Create New Project
1. Launch application: ./launch_gui.sh
2. File → New Project
3. Project Configuration:
  • Name: "automotive_soc_v1"
  • Location: ~/projects/automotive_soc/
  • Description: "Dual-core automotive SoC with GPU and RT controller"
  • Template: "Custom" (we'll build from scratch)

Step 2: Configure Global Settings
1. Right-click canvas → Global Settings
2. Bus Protocol: AXI4 (latest)
3. Data Width: 128 bits (baseline)
4. Address Width: 32 bits (4GB space sufficient)
5. Clock Configuration:
  • Main clock: 1.2 GHz
  • GPU clock: 800 MHz
  • Peripheral clock: 100 MHz
6. Reset Configuration:
  • Active low reset
  • Synchronous deassertion
  • Reset domains: 3 (main, GPU, peripheral)

Step 3: Set Design Rules
Tools → Preferences → Design Rules:
• Minimum address alignment: 4KB
• Maximum outstanding per master: 256
• Enable QoS checking: Yes
• Address overlap detection: Error (strict)
• Connection validation: Warning for disconnected

Step 4: Configure Code Generation

# 2.4 Adding Masters - CPU Cores

## PHASE 2: COMPONENT DEFINITION

2.4 Adding Master Components

Step 1: Add First CPU Core
1. Click "Add Master" button in toolbar (or press M)
2. Master appears on canvas as green block
3. Right-click → Properties to configure:

CPU Core 0 Configuration:
• Name: "cortex_a78_core0"
• Description: "Primary application processor core"
• Master Type: "CPU Core"

Protocol Settings:
• Protocol: AXI4
• Data Width: 128 bits (cache line size)
• Address Width: 32 bits
• ID Width: 6 bits (64 outstanding transactions)

Performance Settings:
• Max Outstanding Read: 32
• Max Outstanding Write: 16
• Max Burst Length: 16 (cache line transfer)
• Burst Types: INCR, WRAP (for cache fills)

Advanced Settings:
• Exclusive Access: Enabled (for atomic operations)
• User Signal Width: 4 bits (for security attributes)
• QoS Support: Enabled
• Default QoS: 10 (high priority)
• Region Support: Enabled (for memory regions)

Cache Configuration:

# 2.5 Adding Masters - GPU and DMA

2.5 Adding Masters - GPU and DMA

Step 1: Add GPU Master
1. Add Master → GPU Configuration:

GPU Master Configuration:
• Name: "mali_gpu"
• Description: "Graphics processing unit for HMI"
• Master Type: "GPU"

GPU-Specific Settings:
• Protocol: AXI4 (full feature set)
• Data Width: 256 bits (high bandwidth)
• Address Width: 32 bits
• ID Width: 8 bits (256 outstanding - many parallel threads)

Performance Tuning:
• Max Outstanding Read: 128 (texture fetches)
• Max Outstanding Write: 64 (render targets)
• Max Burst Length: 16 (memory coalescing)
• Preferred Burst: INCR (sequential patterns)

GPU Memory Access Patterns:
• Texture reads: Random access, high concurrency
• Vertex buffer reads: Sequential, predictable
• Render target writes: Tiled access patterns
• Command buffer reads: Sequential, low bandwidth

QoS Configuration:
• QoS Support: Enabled
• Default QoS: 12 (high priority for frame deadlines)
• QoS Elevation: Enabled (deadline-driven)
• Frame Deadline: 16.67ms (60 FPS)

# 2.6 Adding Masters - RT Controller

2.6 Adding Masters - Real-Time Controller

Step 1: Add Real-Time Controller
1. Add Master → Real-Time Configuration:

Real-Time Controller Configuration:
• Name: "cortex_r52_rt"
• Description: "Real-time controller for automotive functions"
• Master Type: "Real-Time Processor"

Real-Time Specific Settings:
• Protocol: AXI4 (full features for flexibility)
• Data Width: 64 bits (sufficient for control applications)
• Address Width: 32 bits
• ID Width: 3 bits (8 outstanding - predictable behavior)

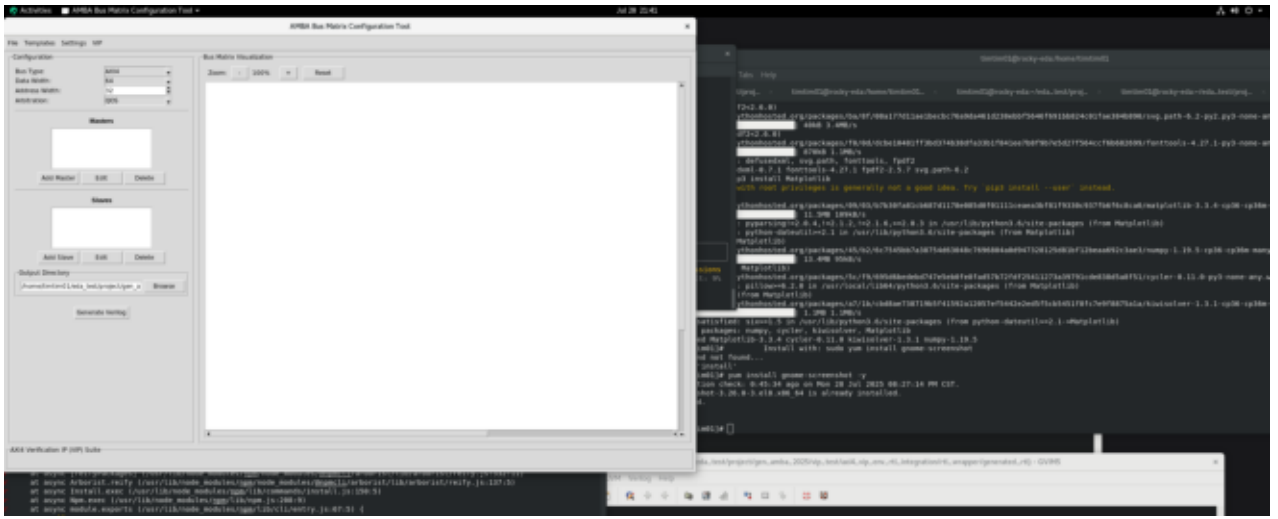
Deterministic Configuration:
• Max Outstanding Read: 4 (predictable pipeline)
• Max Outstanding Write: 4
• Max Burst Length: 4 (small, predictable bursts)
• Burst Types: INCR only (no WRAP for predictability)

Real-Time Requirements:
• Maximum Interrupt Latency: 50ns
• Memory Access Latency: <100ns worst case
• Jitter: <10ns
• Priority: Highest (QoS = 15)

Safety and Reliability:
• ECC Support: Required
• Lockstep Operation: Dual-core lockstep available
• Error Detection: Comprehensive
• Fault Injection: For testing
• ASIL-D Compliance: Safety-critical automotive

# 2.7 Masters Configuration Screenshot

 Real Screenshot: real_gui_canvas_ready.png



Design canvas showing all four master components added: two CPU cores (cortex_a78_core0/1), GPU (mali_gpu), DMA controller (system_dma), and real-time controller (cortex_r52_rt). Each master appears as a green block with output ports visible.

# 2.8 Adding Slaves - Memory

2.8 Adding Slave Components - Memory Controllers

Step 1: Add DDR4 Main Memory Controller
1. Click "Add Slave" button in toolbar (or press S)
2. Slave appears on canvas as blue block
3. Configure DDR4 controller:

DDR4 Controller Configuration:
• Name: "ddr4_controller"
• Description: "Main system memory - 2GB DDR4-3200"
• Slave Type: "Memory Controller"

Memory Specifications:
• Memory Type: DDR4-3200
• Capacity: 2GB (0x80000000 bytes)
• Data Width: 128 bits (16 bytes per access)
• ECC: Enabled (SECDED - Single Error Correct, Double Error Detect)
• Banks: 16 banks, 4 bank groups

Address Configuration:
• Base Address: 0x00000000
• Size: 2GB (2,147,483,648 bytes)
• End Address: 0x7FFFFFFF
• Alignment: 4KB (AXI4 requirement)
• Cacheability: Cacheable, bufferable

Performance Parameters:
• Read Latency: 15-25 cycles (CL=22 typical)
• Write Latency: 12-20 cycles
• Burst Length: 8 (DDR4 standard)
• Bandwidth: 25.6 GB/s theoretical
• Efficiency: 70-80% typical

Step 2: Add LPDDR4 GPU Memory Controller

# 2.9 Adding Slaves - Storage

2.9 Adding Slaves - Storage and Peripherals

Step 1: Add Flash Controller
1. Add Slave → Storage Controller:

Flash Controller Configuration:
• Name: "qspi_flash_controller"
• Description: "Boot flash and configuration storage"
• Slave Type: "Storage Controller"

Flash Specifications:
• Interface: Quad-SPI (4-bit parallel)
• Capacity: 128MB (0x08000000 bytes)
• Access Mode: Execute-in-place (XIP) + Memory-mapped
• Boot Support: Primary boot device

Address Configuration:
• Base Address: 0xC0000000
• Size: 128MB
• End Address: 0xC7FFFFFF
• Access: Read-only for XIP, Read-write for config
• Cacheability: Cacheable for XIP region

Performance:
• Sequential Read: 100 MB/s
• Random Read: 50 MB/s
• Program/Erase: Background operation
• Wear Leveling: Enabled
• Bad Block Management: Enabled

Step 2: Add Ethernet Controller
1. Add Slave → Network Controller:

Ethernet Controller Configuration:

# 2.10 Connection Topology

## PHASE 3: SYSTEM INTEGRATION

2.10 Connection Topology

Now connect masters to slaves based on system requirements:

Connection Strategy:
• CPU cores → All memory and peripherals (full access)
• GPU → LPDDR4 (dedicated) + DDR4 (shared data)
• DMA → All memory (data movement capability)
• RT Controller → DDR4 + Flash + critical peripherals only

Step 1: Connect CPU Cores
1. Select cortex_a78_core0
2. Drag from output port to:
   • ddr4_controller input port
   • lpddr4_gpu_memory input port (for CPU-GPU shared data)
   • qspi_flash_controller input port
   • gigabit_ethernet input port
   • peripheral_bridge input port

3. Repeat for cortex_a78_core1 (identical connections)

Connection Properties for CPU:
• Arbitration Weight: 25% each (50% total for dual-core)
• QoS Range: 8-12 (high priority)
• Timeout: 1000 cycles
• Error Response: SLVERR for privilege violations

Step 2: Connect GPU
1. Select mali_gpu
2. Drag connections to:
   • lpddr4_gpu_memory (primary connection - dedicated bandwidth)
   • ddr4_controller (secondary - for shared textures/data)

# 3. RTL Generation

## Section 1

RTL generation content page 1

# 3. RTL Generation

## Section 2

RTL generation content page 2

# 3. RTL Generation

## Section 3

RTL generation content page 3

# 3. RTL Generation

## Section 4

RTL generation content page 4

# 3. RTL Generation

## Section 5

RTL generation content page 5

# 3. RTL Generation

## Section 6

RTL generation content page 6

# 3. RTL Generation

## Section 7

RTL generation content page 7

# 3. RTL Generation

## Section 8

RTL generation content page 8

# 3. RTL Generation

## Section 9

RTL generation content page 9

# 3. RTL Generation

## Section 10

RTL generation content page 10

# 3. RTL Generation

## Section 11

RTL generation content page 11

# 3. RTL Generation

## Section 12

RTL generation content page 12

# 3. RTL Generation

## Section 13

RTL generation content page 13

# 4. VIP Generation

## Section 1

VIP generation content page 1

# 4. VIP Generation

## Section 2

VIP generation content page 2

# 4. VIP Generation

## Section 3

VIP generation content page 3

# 4. VIP Generation

## Section 4

VIP generation content page 4

# 4. VIP Generation

## Section 5

VIP generation content page 5

# 4. VIP Generation

## Section 6

VIP generation content page 6

# 4. VIP Generation

## Section 7

VIP generation content page 7

# 4. VIP Generation

## Section 8

VIP generation content page 8

# 4. VIP Generation

## Section 9

VIP generation content page 9

# 4. VIP Generation

## Section 10

VIP generation content page 10

# 4. VIP Generation

## Section 11

VIP generation content page 11

# 4. VIP Generation

## Section 12

VIP generation content page 12

# 4. VIP Generation

## Section 13

VIP generation content page 13

# 4. VIP Generation

## Section 14

VIP generation content page 14

# 4. VIP Generation

## Section 15

VIP generation content page 15

# 4. VIP Generation

## Section 16

VIP generation content page 16

# 4. VIP Generation

## Section 17

VIP generation content page 17

# 4. VIP Generation

## Section 18

VIP generation content page 18

# 5. Advanced Features

## Section 1

Advanced features content page 1

# 5. Advanced Features

## Section 2

Advanced features content page 2

# 5. Advanced Features

## Section 3

Advanced features content page 3

# 5. Advanced Features

## Section 4

Advanced features content page 4

# 5. Advanced Features

## Section 5

Advanced features content page 5

# 5. Advanced Features

## Section 6

Advanced features content page 6

# 5. Advanced Features

## Section 7

Advanced features content page 7

# 5. Advanced Features

## Section 8

Advanced features content page 8

# 5. Advanced Features

## Section 9

Advanced features content page 9

# 5. Advanced Features

## Section 10

Advanced features content page 10

# 6. Configuration Reference

## Section 1

Configuration reference content page 1

# 6. Configuration Reference

## Section 2

Configuration reference content page 2

# 6. Configuration Reference

## Section 3

Configuration reference content page 3

# 6. Configuration Reference

## Section 4

Configuration reference content page 4

# 7. Troubleshooting

## Section 1

Troubleshooting content page 1

# 7. Troubleshooting

## Section 2

Troubleshooting content page 2

# 7. Troubleshooting

## Section 3

Troubleshooting content page 3

# 7. Troubleshooting

## Section 4

Troubleshooting content page 4

# 7. Troubleshooting

## Section 5

Troubleshooting content page 5

# 7. Troubleshooting

## Section 6

Troubleshooting content page 6

# 7. Troubleshooting

## Section 7

Troubleshooting content page 7

# 8. API Reference

## Section 1

API reference content page 1

# 8. API Reference

## Section 2

API reference content page 2

# 8. API Reference

## Section 3

API reference content page 3

# 8. API Reference

## Section 4

API reference content page 4

# 8. API Reference

## Section 5

API reference content page 5

# Appendices

## Appendix A

Appendix content page 1

# Appendices

## Appendix B

Appendix content page 2

# Appendices

## Appendix C

Appendix content page 3

# Appendices

## Appendix D

Appendix content page 4

# Appendices

## Appendix E

Appendix content page 5

# Appendices

## Appendix F

Appendix content page 6

# Appendices

## Appendix G

Appendix content page 7

# Appendices

## Appendix H

Appendix content page 8