**AMBA-Compatible MRAM Slave Controller**

**System Architecture Overview**

**Important Notice**

**This documentation and associated design are provided strictly for educational and non-commercial use only.**

* It is intended solely for:
* Engineering students and academic researchers
* University courses and FPGA prototyping
* Internal learning, benchmarking, or architecture exploration

**Commercial usage is not permitted** under any circumstances. This includes, but is not limited to:

* Integration into commercial products or IP
* Silicon implementation or mass production
* Redistribution, sublicensing, or resale

If you are interested in applying this controller in a commercial or industrial project, please contact the author or original organization for licensing information.

This work is published **as-is**, without any express or implied warranties, and is intended to support the open exchange of knowledge and engineering education.

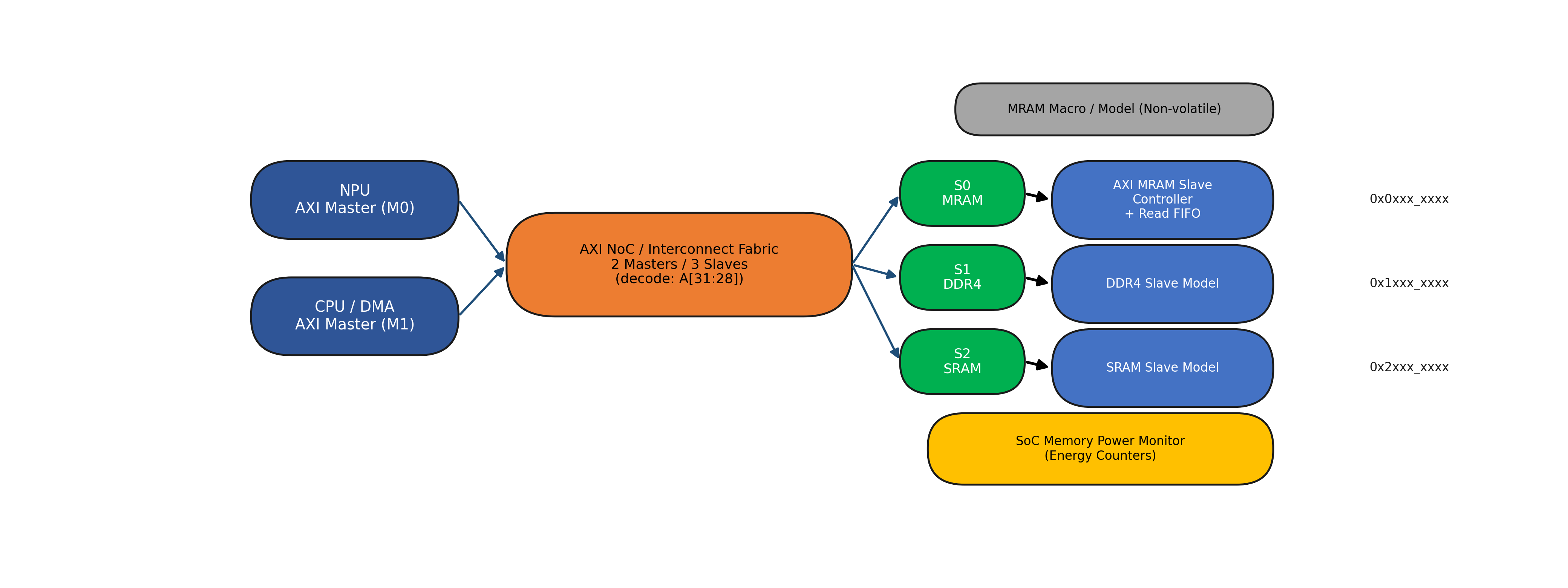
**Introduction**

Magnetoresistive Random Access Memory (MRAM) is a non-volatile memory technology that stores data using magnetic states rather than electric charge. It combines speed, data retention, low power consumption, and CMOS compatibility, making it highly suitable for next-generation System-on-Chip (SoC) designs.

This document describes a slave controller designed to interface an MRAM macro with an SoC using the AMBA system bus. It supports essential memory operations including single writes and incremental burst reads, and is optimized for MRAM’s unique timing characteristics. It also handles initialization, error checking, diagnostics, and low-power operation.

One of the most significant advantages of MRAM over traditional DRAM is its ability to be embedded directly into the chip. This eliminates the need for external DRAM, reducing board complexity, I/O power, and boot latency, while enabling instant-on behavior.

**System Block Diagram**



## System Block Descriptions

**NPU AXI Master (M0):** Represents the NPU/accelerator traffic generator. It issues AXI read/write transactions (e.g., feature maps/weights) into the interconnect and measures performance/energy under AI workloads.

**MRAM Macro / Model (Non-volatile):** Behavioral model (or macro) of the MRAM array. It captures key non-volatile memory characteristics (e.g., read/write latency, energy per access, optional power-gating/retention behavior in the power model).

**S1 DDR4 Region:** Address range mapped to the DDR4 subsystem. Typically stores large, frequently-updated data such as input frames, intermediate feature maps, and buffers where capacity and bandwidth are critical.

**DDR4 Slave Model:** Behavioral model of the DDR4 memory interface. It responds to AXI transactions with configurable latency/throughput and provides a DRAM-like baseline for energy/performance comparison.

**S2 SRAM Region:** Address range mapped to on-chip SRAM. Used for low-latency scratchpad, small buffers, and hot data to reduce external memory traffic.

**SRAM Slave Model:** Behavioral model of SRAM that services AXI accesses with low latency. Useful for validating address mapping and estimating the impact of on-chip memory on traffic and energy.

**SoC Memory Power Monitor (Energy Counters):** A monitoring block that accumulates energy/power-related counters per memory domain (e.g., background/leakage and read/write energy). It enables apples-to-apples comparison of MRAM vs DDR4 vs SRAM under the same workload configuration.

**CPU / DMA AXI Master (M1):** Represents software-driven or DMA-driven memory traffic. It is used for control-plane accesses, bulk copies, preloading data, and background transfers that may run in parallel with the NPU.

**AXI NoC / Interconnect Fabric (2 Masters / 3 Slaves, decode A[31:28]):** Routes AXI transactions from masters to the correct memory target based on address decoding (A[31:28]). It arbitrates between M0 and M1, applies routing/ID handling, and provides the SoC-style connectivity between compute and memories.

**S0 MRAM Region:** Address range mapped to the MRAM subsystem. Typically stores non-volatile or infrequently-updated data such as model weights, code/data that benefits from low leakage, or content that persists across power cycles.

**AXI MRAM Slave Controller + Read FIFO:** The AMBA/AXI-compliant interface for MRAM. It converts AXI read/write bursts into MRAM macro operations and includes a read FIFO to absorb latency, improve throughput, and decouple AXI timing from MRAM internal access timing.

**AMBA Bus Interface and Protocol Compliance**

The controller provides full compliance with the AMBA bus protocol, supporting:

* Read and write transactions with handshake signals
* Single and burst access modes
* Proper backpressure control via ready/valid signaling
* Address decoding and response management

Internal state machines handle all protocol timing and transaction sequencing, ensuring correct data alignment, latency masking, and system-level timing closure.

**Read Optimization Using FIFO**

To compensate for MRAM’s multi-cycle read latency, the controller includes a configurable FIFO buffer. Read data is prefetched from MRAM and held in the FIFO, allowing the controller to respond to the SoC bus without stalling.

This improves effective memory bandwidth and supports burst-read performance with minimal logic overhead. FIFO depth and threshold behavior can be tuned depending on system performance needs.

**Error Detection and Handling**

The controller validates incoming transactions and detects:

* Misaligned addresses
* Illegal burst lengths
* Unsupported write strobes or partial writes
* Out-of-range memory access

On error, it generates appropriate error responses and can trigger system-level interrupts. Error codes are stored in internal registers for postmortem debugging.

**Autonomous Initialization and Power Management**

MRAM requires proper power-up timing and configuration. The controller includes:

* Power-on sequencing logic
* Readiness polling and delay enforcement
* Sleep and wake-up control for low-power modes
* Software-accessible reset for reinitialization

These features eliminate the need for firmware-based setup routines, improving system reliability and reducing boot time.

**Diagnostic and Monitoring Support**

The controller includes:

* Error counters for single-bit and multi-bit faults
* Built-in self-test (BIST) reporting flags
* Status registers for MRAM readiness, FIFO levels, and power state
* Optional access via test or scan interface

This makes it suitable for production testing, in-field validation, and fault-tolerant applications.

**Integration and SoC Benefits**

The MRAM slave controller is designed in modular RTL blocks, optimized for integration into ASIC or FPGA SoCs. It can interface across clock domains and support system-level interconnect topologies.

Because MRAM can be embedded directly within the SoC die, the overall system gains include:

* No external DRAM required
* Reduced board area and pin count
* Lower I/O power
* Faster boot and recovery
* Simpler memory map and controller logic

These advantages make MRAM a compelling choice for edge AI, secure processors, and always-on devices.

**Conclusion**

This MRAM slave controller provides a robust, protocol-compliant, and power-aware interface between MRAM memory macros and AMBA-based SoCs. Its ability to integrate MRAM directly on-chip—without the need for external DRAM—simplifies design and enhances performance in embedded systems.

This architecture is offered for academic and educational purposes, and may serve as a starting point for future research, prototyping, or instruction.