AXI + DMA Subsystem Basics

1️⃣ What is a DMA?

A DMA (Direct Memory Access) is a small hardware block inside a chip that copies data from one memory location to another without CPU involvement.

Think of it as a robot helper:

- CPU says: “Copy data from A to B, length = X”
- DMA says: “Okay boss, I’ll do it automatically.”
- CPU can go do other work.

2️⃣ Why do we need a DMA?

Without DMA:

- CPU must read each byte from memory and write it somewhere else.
- Slow and wastes CPU cycles.

With DMA:

- Copy happens in hardware.
- CPU becomes free.
- Faster throughput (uses AXI bursts).

DMA is used in:

- SoCs
- GPUs
- Networking
- Audio/video data movement
- Embedded controllers

3️⃣ Where does DMA sit inside a SoC?

CPU <----> AXI Interconnect <----> Memories / Peripherals

|

--> DMA Engine

- CPU programs the DMA.
- DMA becomes an AXI Master and directly accesses memory.
- Memory or peripherals become AXI Slaves.

4️⃣ What is a Memory-Copy DMA?

A simple DMA that only does:
src → dst copy.

The CPU gives DMA a structure:

src address → where to read from

dst address → where to write to

length → how many bytes to copy

control flags → start, interrupt enable, etc.

This is called a Descriptor.

5️⃣ AXI Basics Required Before DMA

AXI Channels:

- Write: AW, W, B
- Read: AR, R

VALID/READY handshake:

A transfer happens only when:

VALID=1 and READY=1

Bursts:

AXI supports multi-beat transfers → improves DMA performance.

WSTRB (write strobe):

Used during unaligned or partial first/last beats.

6️⃣ DMA Workflow (Simple)

CPU → writes descriptor into DMA registers

DMA → issues AXI Read from source

DMA → receives RDATA

DMA → issues AXI Write to destination

DMA → finishes and raises interrupt

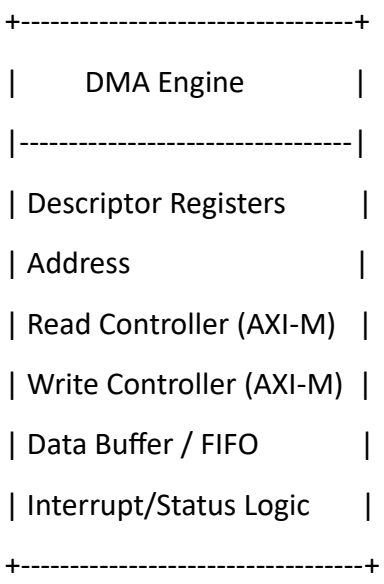
ASCII diagram:

CPU (descriptor) → DMA → AXI Read → Memory (src)

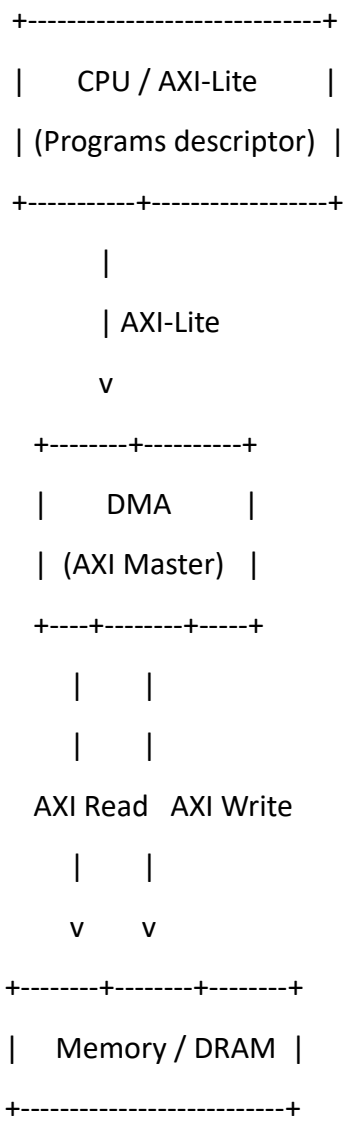
|

└→ AXI Write → Memory (dst)

7️⃣ DMA Internal Architecture (Simple Block Diagram)



8 AXI + DMA Subsystem Block Diagram



9️⃣ Key Concepts Needed Before Verification

DMA concepts:

- Descriptors (src, dst, length, flags)
- Bursts for faster transfers
- Alignment & WSTRB handling
- Backpressure (READY de-assert)
- Data ordering (read → write)
- Outstanding transactions
- Error handling (SLVERR)
- Interrupt generation
- Descriptor chaining (advanced)

🔟 What to Verify in DMA Subsystem

Core goals (must-have):

- Correct copy (src == dst)
- Descriptor behavior
- AXI protocol correctness
- Alignment handling via strobes
- Backpressure handling
- Correct interrupts/status
- Scoreboard comparison

Advanced:

- Random stress
- Error injection
- Descriptor chaining
- Performance
- Coverage

1️⃣1️⃣ Example High-level Test Flow

1. CPU writes:
    - src = 0x1000
    - dst = 0x2000
    - length = 64
    - start = 1
2. DMA reads memory from 0x1000
3. DMA writes data to 0x2000
4. DMA sets completion flag
5. Scoreboard verifies correctness

1️⃣2️⃣ Checklist Before Starting Verification

- AXI-lite interface ready for descriptor programming
- AXI master interface for read/write
- Memory model supporting bursts/backpressure/errors
- UVM components planned:
    - drivers/monitors
    - scoreboard
    - coverage
- Smoke → directed → random → stress tests ready

Final Summary

A DMA subsystem is nothing but a hardware module that automatically copies a block of data using AXI Read + AXI Write. You verify that data is copied correctly under all protocol, alignment, backpressure, error, and timing conditions.

This document gives the full beginner-level explanation, just like APB and AXI4-Lite docs.