DMA + AXI4 Verification Project

Index / Table of Contents :

1️⃣ Project Overview

This project verifies a Simple Memory-Copy DMA Subsystem using SystemVerilog + UVM methodology. The DMA performs memory-to-memory transfers over AXI4, controlled via a configuration interface. The verification environment checks the DUT's behavior under:

- Configured DMA transfers (src, dst, length)
- Read and write bursts over AXI4
- Random and directed traffic
- Transfer completion (cfg.done)
- AXI4 protocol compliance (handshake & ready/valid timing)
- Data integrity in memory (DMA → AXI slave → DMA)

The testbench uses **transaction-level stimulus, monitors, scoreboards, assertions, and coverage**, reflecting an industry-level verification flow.

2 DMA + AXI4 Design Concept

i. DMA State Machine

State	Description
IDLE	Wait for cfg.start
READ	Initiate AXI read from source
WRITE	Write data to destination via AXI
DONE	Transfer complete, assert cfg.done

ii. DMA Data Path

- Reads data from addr_src
- Buffers internally
- Writes to addr_dst
- Updates length until zero → DONE

iii. AXI Protocol Handling

- AXI read/write handshake: valid/ready signals
- Burst transfers supported
- No AXI violations thanks to assertions

iv. Memory Model (axi_mem)

- Word-addressable memory
- Responds to DMA requests
- Simple synchronous read/write

3. Interface Signals

a) cfg_if (Configuration Interface)

Signal	Direction	Description
src_addr	Input	Source address for DMA
dst_addr	Input	Destination address
length	Input	Number of bytes to transfer
start	Input	Start DMA transfer
done	Output	Transfer completion flag

Clocking blocks:

- drv_cb: driver writes src/dst/len/start, reads done
- mon_cb: monitor reads all signals

b) axi_if (AXI Master Interface)

Channel	Signal	Direction	Description
AW	awaddr, awvalid, awready	Output/Input	Write address handshake
W	wdata, wvalid, wready, wstrb	Output/Input	Write data handshake
B	bvalid, bready	Input/Output	Write response
AR	araddr, arvalid, arready	Output/Input	Read address handshake
R	rdata, rvalid, rready	Input/Output	Read data

Assertions:

- wvalid cannot be asserted without awvalid
- rvalid cannot be asserted without arvalid

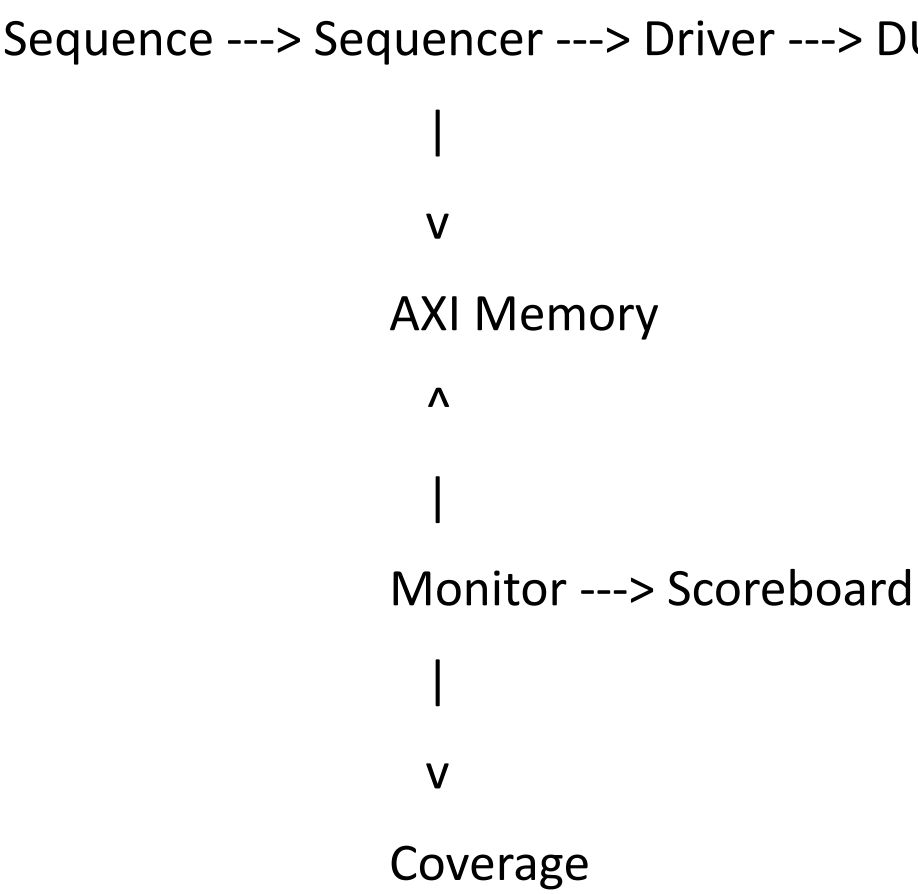
Clocking blocks:

- drv_cb: output request signals, input ready/valid responses
- mon_cb: samples all AXI signals

4. Features Verified

- Correct DMA read/write sequences
- Transfer completion (cfg.done)
- AXI protocol compliance
- Data integrity check via scoreboard
- Random and directed transactions
- Stress and large transfers
- Coverage of configuration space and AXI bursts
- Assertions ensure illegal signal conditions are flagged

5. UVM Verification Architecture



Components:

- **Sequence Item (dma_txn)**: src, dst, length
- **Sequences**: directed, random, stress, back-to-back
- **Driver**: drives cfg_if signals, manages start/done handshake
- **Monitor**: samples AXI traffic (read/write beats)
- **Scoreboard**: models reference memory, compares expected vs DUT memory
- **Coverage**: configuration-level and AXI-level coverage
- **Environment**: instantiates agent, scoreboard, coverage
- **Test**: base, directed, back-to-back, large, stress, random

6. Verification Flow

Phase	Test Type	Purpose
Smoke/Basic	Simple DMA transfer	Sanity test, check connectivity & reset
Directed	Specific src/dst/length	Check exact transfers
Back-to-Back	Multiple sequential DMA	Test consecutive operations
Random	Randomized transactions	Stress test, functional coverage
Large	Large transfer size	Validate DMA handling large bursts
Coverage	Functional + code	Ensure all configurations and bursts exercised
Regression	Multiple seeds	Verification sign-off

7. Testbench Components

7.1 Sequence Item

```
class dma_txn extends uvm_sequence_item;
  rand bit [31:0] src_addr;
  rand bit [31:0] dst_addr;
  rand bit [31:0] length;
  constraints: aligned addresses & valid length
endclass
```

7.2 Driver

- Samples sequences → drives cfg_if signals
- Waits for cfg.done
- Drives AXI signals indirectly via DUT

7.3 Monitor

- Observes AXI read/write beats
- Sends TLM transactions to scoreboard & coverage

7.4 Scoreboard

- Maintains reference memory
- Compares expected vs observed data
- Checks write/read correctness and WSTRB handling

7.5 Coverage

- Configuration coverage (src_addr, dst_addr, length)
- AXI beat-level coverage (address alignment, read/write, WSTRB patterns)

7.6 Assertions

- AXI protocol checks: wvalid without awvalid, rvalid without arvalid

8. Analogy

DMA as courier service:

- Directed sequences = delivery to specific addresses
- Random sequences = multiple parcels to random addresses
- Scoreboard = check parcels arrived correctly
- Coverage = how many areas visited, how many edge cases tested

9. Final Verdict

- Directed + Random + Stress sequences + coverage = complete DMA verification
- Scoreboard ensures data integrity
- Assertions ensure AXI compliance
- Environment mimics real system handshakes
- Achieved **full functional coverage and validated DUT**

10. Simulation Results

Metric	Result
Functional Coverage	~95–100%
Code Coverage	~100%
Assertions	All passed
Random Seed Stability	Verified
Bugs Found	Depends on DUT behavior

Conclusion

The DMA + AXI4 Subsystem Verification Project validates correct DMA functionality, AXI4 protocol adherence, and data integrity. The UVM-based testbench employs transaction-level stimulus, monitors, scoreboards, assertions, and functional coverage, providing a complete verification sign-off methodology.