AXI4 Lite Basics

1️⃣ What is AXI4-Lite?

AXI4-Lite is a simplified version of the AXI4 protocol.

- Purpose: Single-beat register read/write (no bursts, no IDs).
- Use case: Control and status registers of peripherals (UART, GPIO, timers, etc.).
- AXI4-Lite allows independent address and data channels and robust VALID/READY handshake for backpressure handling.

Analogy:

- AXI4-Lite = a quiet neighborhood road: simple, predictable, one car (transaction) at a time.
- AXI4 full = freeway: supports multiple cars (transactions), out-of-order, bursts, pipelining.
- APB = slow peripheral road: very simple, one transaction at a time, slower, used for low-speed peripherals.

2️⃣ Why AXI4-Lite Exists?

Full AXI4 is overkill for simple peripherals:

- AXI4 supports bursts, multiple IDs, out-of-order, pipelining → complex.
- Peripherals like UART, timers, GPIO, or registers don't need this complexity.

AXI4-Lite solves this by:

- Single-beat transfers
- No transaction IDs → deterministic responses
- Simpler interface → easier to implement and verify

💡 Think: CPU writing to a UART control register does not need multiple outstanding transactions.

3️⃣ Roles in AXI4-Lite

- Master: Initiates transactions (CPU, DMA controller, testbench driver).
- Slave: Responds to master requests and holds registers (peripherals: UART, SPI, GPIO, DMA control, Ethernet MAC, PCIe config registers).
- IP / Blocks:
  - IP here = individual hardware blocks like UART, GPIO, DMA controller, AXI4-Lite Slave module, etc.
  - AXI4-Lite is the bus protocol, connecting the master to these peripheral IPs.

4️⃣ AXI Channels (Simplified)

AXI4-Lite has **5 logical channels** (simplified from AXI4):

1. **AW (Address Write)** – Master → Slave
   - Signals: AWADDR, AWVALID, AWREADY
   - "I want to write to this address."
2. **W (Write Data)** – Master → Slave
   - Signals: WDATA, WSTRB, WVALID, WREADY
   - "Here's the data to write."
3. **B (Write Response)** – Slave → Master
   - Signals: BRESP, BVALID, BREADY
   - "Write done — OKAY or SLVERR."
4. **AR (Address Read)** – Master → Slave
   - Signals: ARADDR, ARVALID, ARREADY
   - "I want to read from this address."
5. **R (Read Data)** – Slave → Master
   - Signals: RDATA, RRESP, RVALID, RREADY
   - "Here's the data and status."

💡 **Important:** Address and data channels are **independent** → Master can provide address before or after data.

5 VALID / READY Handshake

- Master asserts *_VALID when it has address/data ready.
- Slave asserts *_READY when it can accept address/data.
- Transfer occurs only when VALID && READY are both high.

Backpressure example:

- If slave is busy, READY=0 → master must wait.
- Ensures protocol works with variable latency.

6 Write Transaction Sequence

1. Master asserts AWADDR + AWVALID.
2. Slave asserts AWREADY when it can accept address.
3. Master asserts WDATA + WSTRB + WVALID.
4. Slave asserts WREADY when it can accept data.
5. Slave writes to register, asserts BVALID + BRESP.
6. Master asserts BREADY to consume response.

Note: WSTRB = byte-enable (partial writes).

7 Read Transaction Sequence

1. Master asserts ARADDR + ARVALID.
2. Slave asserts ARREADY when it can accept address.
3. Slave prepares data, asserts RVALID + RDATA + RRESP.
4. Master asserts RREADY to consume data.

Only single-beat read per AXI4-Lite.

8 AXI4-Lite Rules & Restrictions

- No bursts → single transaction per access.
- No transaction IDs → responses are in order.
- Simple responses → OKAY or SLVERR.
- Usually one outstanding transaction → deterministic, easy verification.

9️⃣ Comparison: AXI4-Lite vs APB vs AXI4 Full

Feature	APB	AXI4-Lite	AXI4 Full
Bursts	❌ No	❌ No	✓ Yes (1–256 beats)
Out-of-order	❌ No	❌ No	✓ Yes
Pipelining	❌ No	✓ Yes	✓✓ Deep
Clocking	Slow	High-speed	High-speed
Typical use	UART, GPIO, Timer	Control/status registers of IP	DDR, caches, DMA
Parallel read/write	❌ No	✓ Yes	✓✓ Full parallel

Notes:

- AXI4-Lite is used for fast register-mapped peripherals, not high-speed data streams.
- APB is used for slow, low-cost peripherals.
- AXI4 Full is used for high-performance, bursty data transfer (e.g., DDR, DMA).

1️⃣0️⃣ Key Concepts for Verification

- Registers / IPs: Slave IPs hold configuration/control/status registers.
- Independent address/data channels: Address can arrive before or after data.
- Simultaneous read/write: AXI4-Lite allows a read and a write at the same time.
- Backpressure: Slave can deassert READY → master waits.
- Byte enables: Ensure slave supports partial writes if WSTRB < full width.
- Reset behavior: All channels idle, registers cleared.
- Ordering: Since no IDs, responses must correspond exactly to request order.

1️⃣1️⃣ Where AXI4-Lite is Used

- CPU writes configuration registers of peripherals: UART baud rate, GPIO direction, timers.
- Simple MMIO accesses in SoC.
- Often bridges to APB for low-speed peripherals.
- Typical AXI4-Lite peripherals: UART, SPI, GPIO, timers, interrupt controllers, PWM controllers.

1️⃣2️⃣ Verification Mindset

Before writing UVM sequences:

- Basic functionality: reads/writes succeed, data matches registers.
- Byte enables: partial writes update selected bytes.
- Backpressure: master & slave handle READY low.
- Error handling: slave generates SLVERR for invalid addresses.
- Ordering: single-beat transfers in order.
- Reset behavior: interface & slave return to known state.

1️⃣3️⃣ Practical Checklist Before Coding

- Data width (e.g., 32-bit)
- Byte lanes (WSTRB width = DATA_WIDTH/8)
- Address alignment (word-aligned?)
- Response handling (OKAY/SLVERR)
- Clock/reset conventions
- Interface modports & clocking blocks (for UVM driver & monitor)

Summary:

AXI4-Lite is ideal for verifying simple memory-mapped peripherals.

- Use single-beat transactions.
- Use VALID/READY handshakes for timing control.
- Focus on register correctness, byte enables, backpressure, and response validation.