IJETIR

ISSN (Online) 2583-0554

International Journal of Emerging Technologies and Innovative Research (IJETIR)

Volume 6, Issue 8, August 2026

Impact Factor: 5.731

Design and Transaction-Level Verification of an AMBA AXI4-Lite/ACE Interconnect Architecture for Scalable SoC Integration

Mogili Ravi[1] and Boppani Om Chandra Shanthi Swaroop Kumar[2]

Assistant Professor, ECE Department[1]

PG Scholar, ECE Department[2]

Annamacharya Institute of Technology & Sciences-Hyderabad, Telangana, India.

mogili.rv@gmail.com

Abstract: *Modern System-on-Chip (SoC) platforms integrate processors, hardware accelerators, DMA engines, shared caches, memory controllers, and a large number of memory-mapped peripherals, making the on-chip interconnect a critical architectural component. This paper presents a transaction-oriented interconnect framework that combines AXI4-Lite for lightweight control and register access with ACE/ACE-Lite coherency concepts for shared-memory communication. The architecture is organized around master and slave interfaces, memory-mapped address decoding, multi-master arbitration, independent read/write channel control, a snoop control unit, a coherency manager, and a protocol/response handler. A transaction-level verification environment comprising a generator, driver, monitor, protocol checker, scoreboard, and coverage collector is defined to validate legal and illegal accesses, simultaneous requests, back-pressure, response errors, and coherency-oriented scenarios. The proposed methodology supports modular design reuse and early validation before RTL refinement. The supplied source material does not contain completed RTL simulation waveforms, FPGA synthesis statistics, or measured latency/throughput values; therefore, this manuscript reports the architecture, verification criteria, and design-level evaluation without inventing quantitative hardware results.*

Keywords: AMBA, AXI4-Lite, ACE, ACE-Lite, cache coherency, transaction-level modeling, SoC interconnect, address decoder, arbitration, verification.

I. INTRODUCTION

The rapid growth of heterogeneous SoC design has shifted the performance bottleneck from individual processing blocks toward the communication fabric that connects them. A modern embedded or application processor can contain multiple CPU cores, accelerators, DMA engines, memory controllers, and diverse peripheral controllers. These components differ widely in bandwidth, latency, coherency, and control requirements. The interconnect must therefore provide reliable transaction routing, arbitration, synchronization, and error reporting while remaining scalable and reusable [1] – [5].

AXI4-Lite is well suited to register-oriented communication because it removes burst-transfer complexity and uses five independent channels for write address, write data, write response, read address, and read data. Control/status registers of GPIO, UART, SPI/I2C, timers, and configuration blocks can therefore be accessed using simple memory-mapped transactions. However, AXI4-Lite alone does not provide cache-coherency support for shared-memory systems containing multiple processors or accelerators [7] – [12].

ACE extends AXI with coherency mechanisms such as snoop-oriented communication and cache-maintenance behavior, while ACE-Lite provides a lighter mechanism for master's that require coherent visibility but do not maintain a fully coherent local cache.[14]. In the proposed framework, AXI4-Lite is used for low-bandwidth control paths, whereas ACE/ACE-Lite concepts are applied to coherent memory-oriented paths. The objective is not to make AXI4-Lite itself coherent, but to organize both communication classes within one scalable SoC interconnect architecture.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

A second motivation is verification complexity. As the number of masters and slaves grows, the number of possible interactions increases rapidly. Transaction-level modeling abstracts a bus exchange into meaningful operations such as read, write, response, and coherency request, allowing architecture and routing behavior to be examined before every signal-level timing detail is fixed. This paper therefore emphasizes both the interconnect architecture and a reusable transaction-based verification strategy [15].

II. BACKGROUND AND RELATED WORK

2.1 AMBA AXI4-Lite communication

AXI4-Lite uses independent read and write paths with valid-ready handshaking. The sender asserts VALID when information is available, while the receiver asserts READY when it can accept the transfer. A transfer is completed only when both are asserted in the same cycle:

$$Transfer = VALID \wedge READY \quad (1)$$

This rule decouples the timing of masters and slaves and permits controlled back-pressure. In a write operation, the write address and write data are accepted before a write response is returned. In a read operation, the read address is accepted before read data and a response are returned.

2.2 ACE/ACE-Lite coherency

In a shared-memory SoC, different processing agents may hold copies of the same memory block. If one agent modifies a cached copy while another reads an older copy, system behavior becomes inconsistent. The architecture therefore includes a snoop control unit and a coherency manager to coordinate visibility between CPU caches, shared cache, DMA access, and main memory. Full ACE behavior is associated with coherent cacheable masters, whereas ACE-Lite behavior is associated with non-cacheable or I/O-oriented masters that still require coherent memory visibility.

2.3 Transaction-level modeling and verification

Transaction-level modeling represents a communication event at a higher abstraction level than RTL signal toggles. A generic transaction can be expressed as a tuple containing address, data, control type, and response:

$$T = \{A, D, C, R\} \quad (2)$$

This abstraction supports rapid generation of legal and illegal traffic, conflict conditions, and response scenarios. The final architecture can then be refined into synthesizable Verilog or SystemVerilog modules after the intended transaction behavior has been established.

III. EXISTING INTERCONNECT LIMITATION

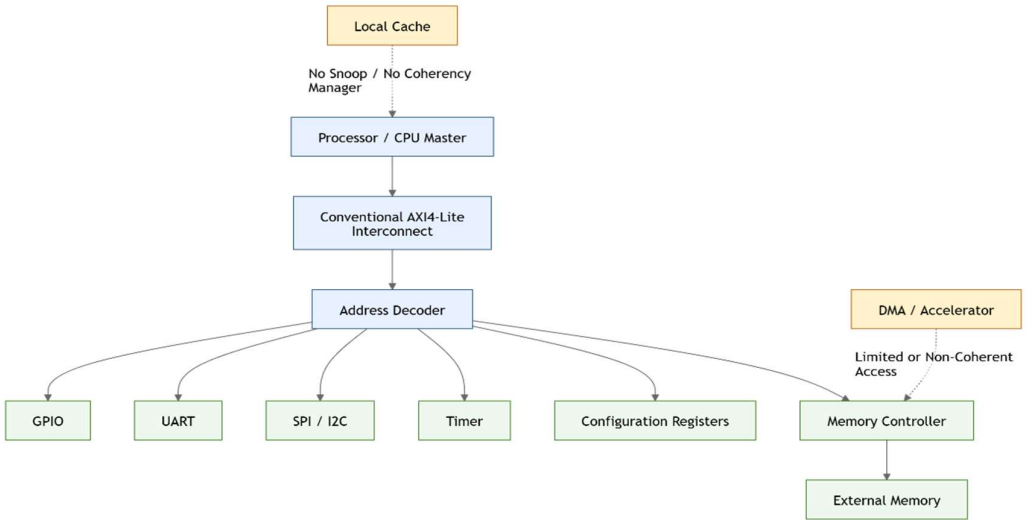

Fig. 1. Existing AXI4-Lite-based SoC interconnect without ACE/ACE-Lite coherency support, reproduced from the supplied design document.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

A conventional AXI4-Lite-centered SoC can connect a processor to memory-mapped peripherals through address decoding and simple read/write transactions. Such an arrangement is appropriate for control traffic but provides limited support for multi-master arbitration, shared-memory coherency, systematic error handling, and transaction-oriented verification. When DMA engines or accelerators share buffers with cached processors, software may have to manage cache synchronization explicitly.

IV. PROPOSED AXI4-LITE/ACE INTERCONNECT ARCHITECTURE

Figure 2 shows the proposed architecture. The processing and coherent-agent layer contains CPU cores with private caches, a GPU/accelerator, and a DMA/I/O master. The central interconnect contains six major functional blocks: address decoder, arbitration logic, read/write channel controller, snoop control unit, coherency manager, and protocol/response handler. AXI4-Lite slave interfaces connect the peripheral-control layer, while the coherent side connects shared cache and the main-memory subsystem.

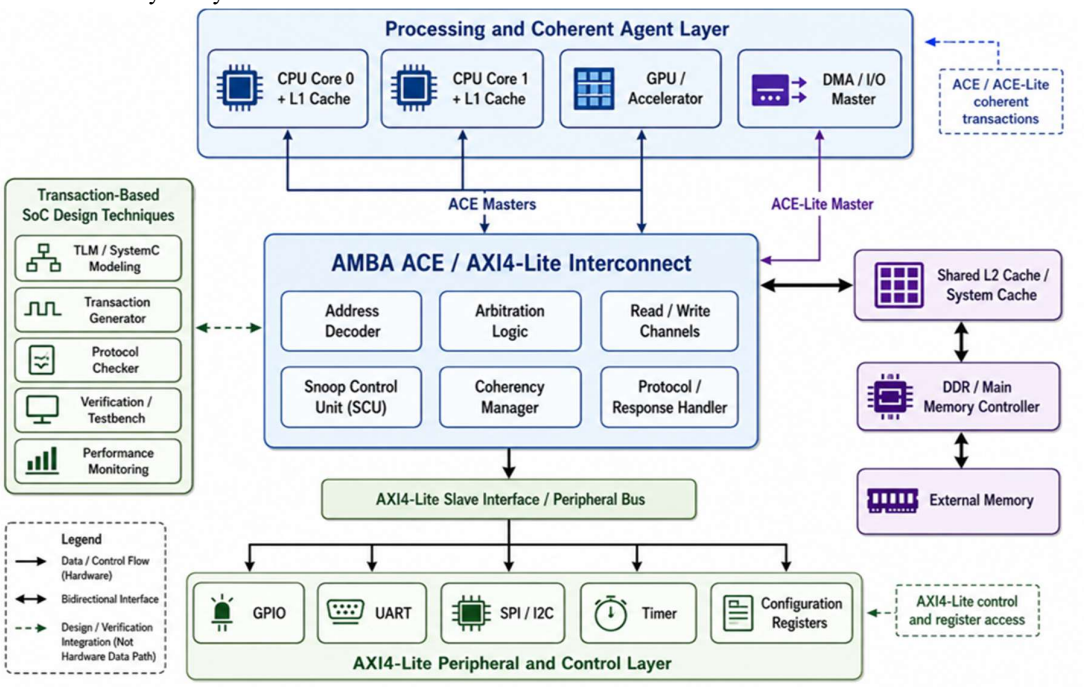

Fig. 2. Proposed transaction-based AMBA ACE/AXI4-Lite interconnect architecture for scalable SoC integration.

4.1 Master and slave interfaces

Master interfaces originate address, data, and control requests. Candidate masters include CPU cores, DMA engines, accelerators, and testbench traffic sources. AXI4-Lite peripheral slaves respond to memory-mapped single-beat transactions. The slave interface accepts a decoded request, updates a register or control location for a write, and returns data for a read.

4.2 Address decoder and memory map

The address decoder routes each transaction according to a predefined memory map. For slave i with base address B_i and upper address U_i, the select condition can be represented as:

$$S_i = 1, if\ B_i \le A \le U_i;\ otherwise\ S_i = 0 \qquad (3)$$

Address range	Target slave
0x0000_0000 – 0x0000_0FFF	GPIO
0x0000_1000 – 0x0000_1FFF	UART

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

0x0000_2000 – 0x0000_2FFF	SPI/I2C
0x0000_3000 – 0x0000_3FFF	Timer
0x0000_4000 – 0x0000_4FFF	Configuration Registers
0x1000_0000 – 0x1FFF_FFFF	Memory Controller

Table 1. Memory-mapped address allocation used by the proposed methodology.

If no address range matches, the response handler returns a decode-error response. This prevents undefined access and gives software an explicit indication that the target address is invalid.

4.3 Arbitration logic

In a multi-master SoC, simultaneous requests are expected. The arbitration unit selects one requester for a shared resource and prevents transaction collision. The source methodology proposes round-robin arbitration as a fair default because the priority rotates after each completed grant. A priority-based policy may instead be used when real-time masters require bounded or preferred access.

4.4 Independent read/write channel control

The read and write controllers manage the independent AXI4-Lite channels and preserve channel-level handshaking. The interconnect may accept address or data only when the corresponding READY signal is asserted. Back-pressure is therefore naturally represented without data loss. Figure 3 summarizes the transaction sequence.

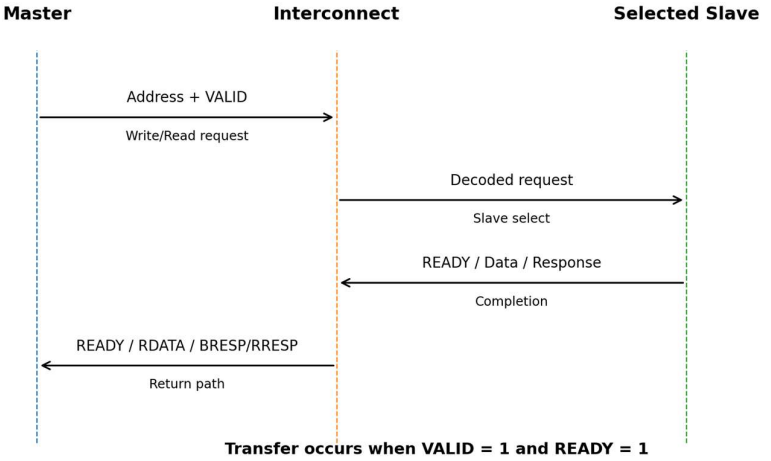

Fig. 3. Simplified transaction sequence for master-interconnect-slave communication.

4.5 Coherency manager and snoop control

The coherency manager observes transactions directed to shared memory and coordinates data visibility across coherent agents. The snoop control unit checks whether another cache may contain the requested line and can trigger abstract invalidate, update, or forwarding actions according to the coherency model. The source methodology intentionally keeps these blocks at a transaction-oriented level so that coherency scenarios can be validated before detailed cache-state RTL is implemented.

4.6 Protocol and response handler

Every transaction must terminate with a well-defined response. A successful access returns a normal completion response; an unmapped address generates a decode error; and a slave that cannot complete the requested operation can generate a slave error. Explicit error signaling improves fault isolation and prevents a master from treating an incomplete operation as successful.

V. TRANSACTION-LEVEL VERIFICATION METHODOLOGY

The verification environment is organized around a transaction generator, driver, monitor, protocol checker, scoreboard, and coverage collector. The generator creates legal and illegal scenarios; the driver converts them into interface activity; the monitor reconstructs observed transactions; the checker evaluates protocol rules; and the scoreboard compares expected and actual behavior. Coverage identifies which classes of transactions and corner cases have been exercised.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

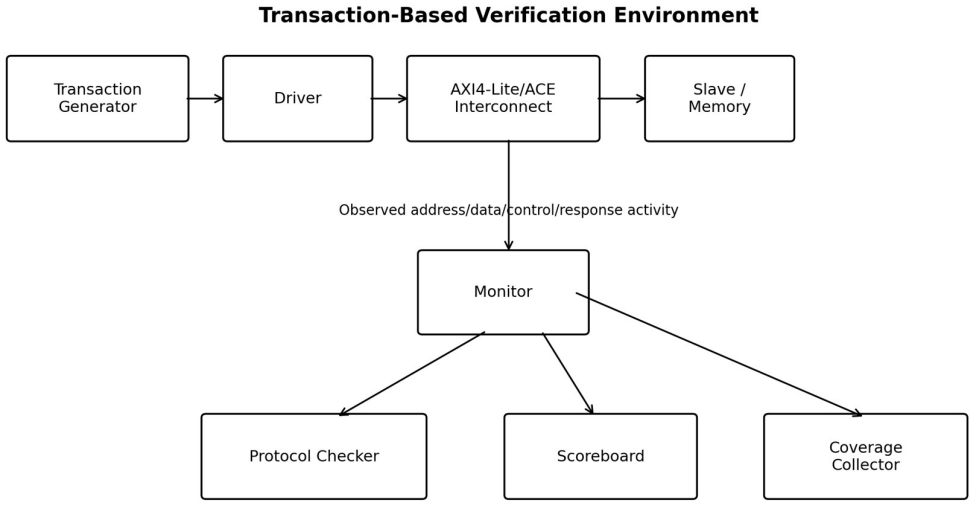

Fig. 4. Transaction-based verification environment for the proposed interconnect.

5.1 Verification scenarios

Test scenario	Stimulus	Expected interconnect behavior	Primary check
Valid AXI4-Lite write	Mapped address + write data	Route to selected slave and return successful write response	Decoder, write channels, response
Valid AXI4-Lite read	Mapped read address	Return selected slave data and successful read response	Read channels, routing
Invalid address	Address outside all ranges	No slave selected; return decode error	Decoder/response handler
Simultaneous masters	Two or more active requests	Grant one requester according to arbitration policy	Mutual exclusion/fairness
Slave busy/back-pressure	Slave delays READY	VALID/data remain stable until acceptance	Handshake compliance
Slave error	Selected slave rejects operation	Return slave-error response	Error propagation
Coherent memory access	CPU/DMA/accelerator targets shared buffer	Invoke coherency/snoop path before completion	Coherency handling
Repeated conflict	Continuous requests from multiple masters	No starvation under round-robin policy	Arbitration fairness

Table 2. Core transaction-level verification matrix.

5.2 Protocol properties to check

The protocol checker should verify that transfers occur only under VALID/READY agreement, that a pending transaction is not silently discarded, that response channels correspond to accepted requests, that no two masters simultaneously own the same shared target, and that error responses are generated for invalid or rejected accesses. For coherent accesses, the checker should also confirm that the transaction is routed through the coherency-management path rather than directly bypassing consistency control.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

5.3 Scoreboard and coverage

The scoreboard maintains a reference model of the memory map and expected slave behavior. For every generated transaction, it predicts the selected slave, expected data or error response, and expected grant ownership. Functional coverage can classify address regions, read/write types, arbitration combinations, response types, back-pressure duration, and coherent versus non-coherent accesses.

VI. PERFORMANCE METRICS AND DESIGN EVALUATION

The source methodology defines transaction latency, throughput, arbitration delay, response time, and resource utilization as the principal evaluation metrics. Transaction latency is measured from request initiation to response completion:

$$L_{tx} = t_{response} - t_{request} \quad (4)$$

Throughput can be expressed as the number of completed transactions during an observation interval:

$$Throughput = \frac{N_{completed}}{T_{observation}} \quad (5)$$

Arbitration delay is the waiting interval between a master's request and its grant. These metrics are appropriate for comparing arbitration policies, peripheral response behavior, and coherent versus non-coherent paths after an RTL implementation is available.

6.1 Results supported by the supplied source

The supplied document contains the full architecture, address map, verification plan, advantages, applications, and future work, but its Results chapter does not contain completed simulation waveforms or quantitative implementation measurements. It also states that a complete Verilog/SystemVerilog RTL implementation, simulator-based verification, FPGA synthesis, and measurement of timing, power, and resource utilization are future work. Consequently, numerical latency, throughput, LUT count, flip-flop count, maximum frequency, and power values are not reported here.

6.2 Design-level evaluation

Design concern	Architectural mechanism	Expected validation evidence
Peripheral control complexity	AXI4-Lite control path	Correct single-beat register read/write behavior
Multi-master contention	Round-robin/priority arbitration	Exclusive grants and starvation-free service
Invalid access	Address decoder + response handler	Decode error for unmapped addresses
Shared-memory consistency	Coherency manager + SCU	Coherency path invoked for shared data
Heterogeneous timing	VALID/READY back-pressure	Stable pending data until handshake
Verification scalability	Transaction generator/checker/scoreboard	Reusable test scenarios and automated checking
Design reuse	Modular decoder/arbiter/controllers	Blocks can be parameterized for different SoCs

Table 3. Design-level evaluation derived from the proposed architecture.

At the architectural level, the hybrid organization balances protocol complexity by applying AXI4-Lite only to low-bandwidth control traffic and reserving coherency support for shared-memory agents. This selective deployment can reduce unnecessary control-path hardware while still providing a structured route for coherent transactions. The modular decomposition also supports independent verification of routing, arbitration, channel control, coherency, and response handling.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

VII. RESULTS AND DISCUSSION

7.1 Functional Simulation and Transaction Verification

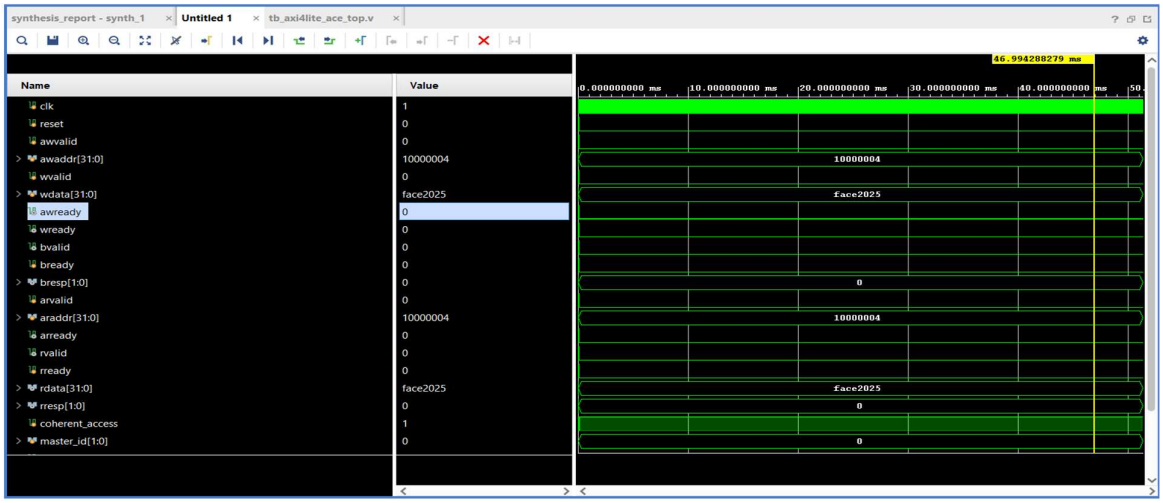

Fig 5. AXI4-Lite ACE transaction-level simulation waveform.

The simulation waveform demonstrates a representative AXI4-Lite ACE-oriented read/write test scenario. At the cursor position near 46.994 ms, the design is out of reset (reset = 0) and the coherent-access control is asserted (coherent_access = 1). The write address and read address are both observed as 0x10000004. The write-data value is 0xFACE2025, and the read-data bus shows the same value, 0xFACE2025. This equality between the applied write data and returned read data is consistent with correct storage/routing behavior for the exercised address path.

7.2 Synthesized RTL Architecture

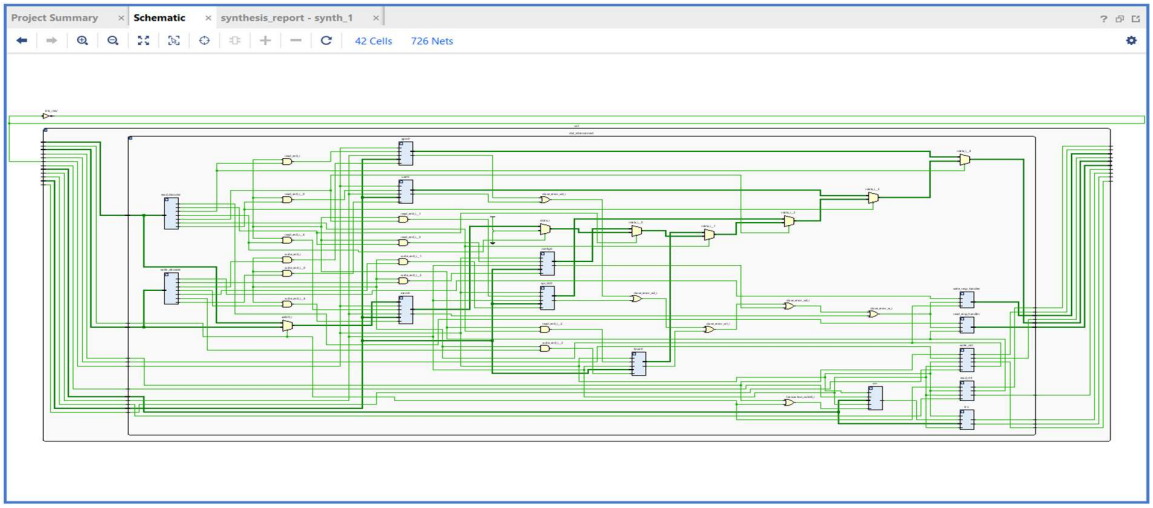

Fig 6. Synthesized RTL schematic of the proposed AXI4-Lite ACE interconnect.

The synthesized schematic confirms that the high-level interconnect description has been translated into an interconnected RTL network. The Vivado schematic reports 42 cells and 726 nets. The visible logic contains decoder and routing structures, peripheral-oriented blocks, control logic, multiplexing, and response paths. This is consistent with the project methodology, in which the address decoder, read/write transaction control, slave interfaces, response handler, and coherency-related control are implemented as modular blocks and then integrated into a common interconnect.

7.3 Synthesis Result

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

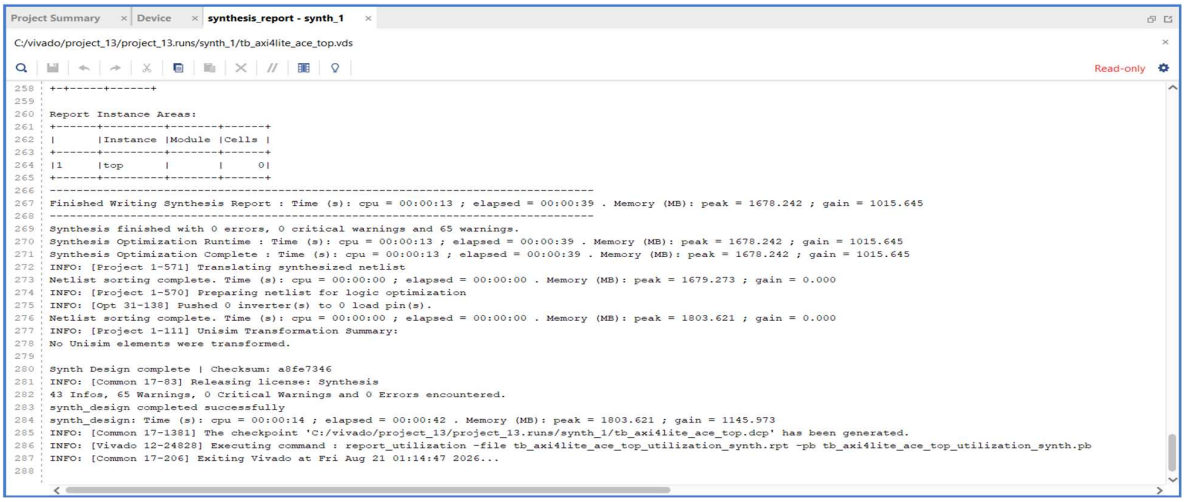

Fig 7. Vivado synthesis completion report.

Vivado completes synthesis successfully. The report explicitly shows 0 errors and 0 critical warnings, demonstrating that the RTL is syntactically valid and synthesizable for the selected target device. A total of 65 warnings are reported. These warnings do not prevent synthesis, but they should be reviewed individually before final hardware deployment because some warnings may relate to unconnected signals, optimization, testbench-derived hierarchy, or constraints.

7.4 Power Analysis

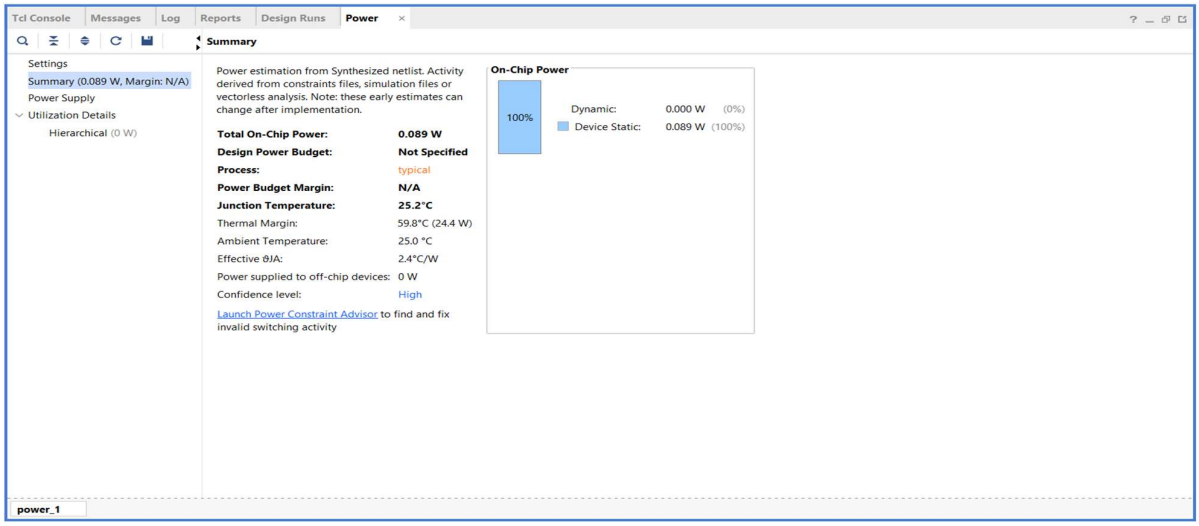

Fig 8. Vivado on-chip power estimation report.

The Vivado power estimator reports a total on-chip power of 0.089 W. The complete reported power is device-static power (0.089 W, 100%), while dynamic power is shown as 0.000 W. The junction temperature is estimated at 25.2 °C for an ambient temperature of 25.0 °C. The report also shows a thermal margin of 59.8 °C (24.4 W), an effective junction-to-ambient thermal resistance of 2.4 °C/W, and a high confidence level for the available estimate.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

7.5 FPGA Device View

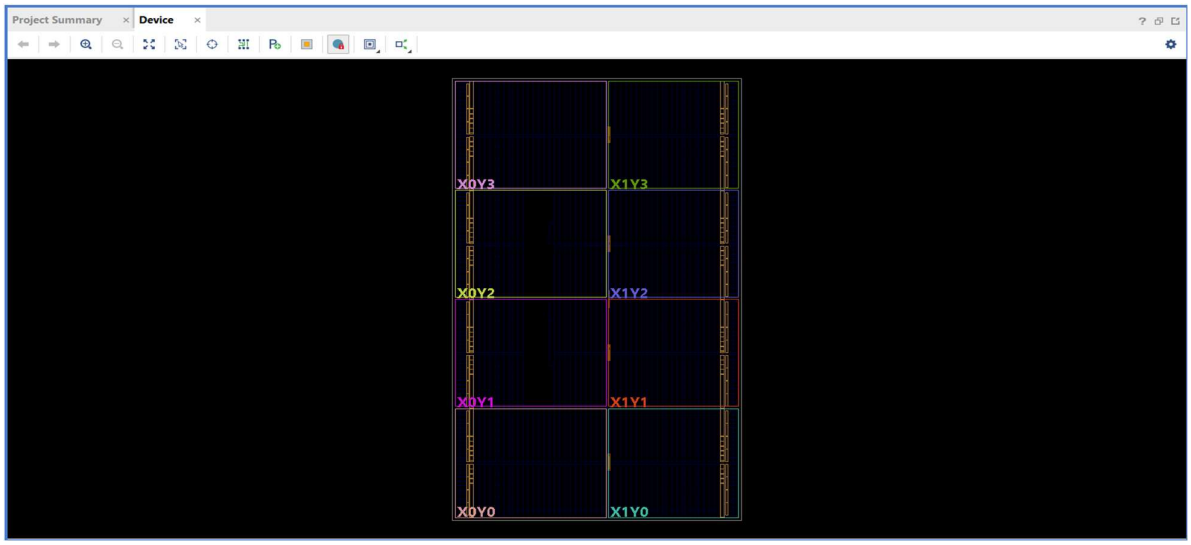

Fig 9. Target FPGA device view used for implementation analysis.

The FPGA device view shows the selected device divided into its clock/placement regions (for example, X0Y0 through X1Y3). This confirms that the design is being evaluated within a concrete FPGA target environment rather than only as an abstract RTL model. The screenshot itself does not display numerical LUT, flip-flop, BRAM, DSP, or I/O utilization values, so exact percentage utilization cannot be claimed from this image alone. Nevertheless, the device-level view complements the synthesis and power outputs by showing that the project has progressed into the FPGA-oriented design flow.

VIII. ADVANTAGES, APPLICATIONS, AND LIMITATIONS

8.1 Advantages

The architecture supports scalable integration of multiple masters and slaves, provides memory-mapped peripheral access, prevents request collisions through arbitration, and improves correctness of shared-memory communication through coherency-oriented handling. Transaction-level verification reduces dependence on manual waveform inspection and supports reusable automated checking. Because the decoder, arbiter, channel controllers, response handler, and verification components are modular, the methodology can be adapted to different SoC sizes.

8.2 Application domains

The proposed framework is applicable to multicore processors, AI and machine-learning accelerators, FPGA-based SoC prototypes, embedded control systems, IoT platforms, DMA-based data-movement systems, image and signal-processing SoCs, automotive controllers, robotics, industrial automation, and high-performance embedded processors.

8.3 Limitations

The present work is an architecture and transaction-level verification methodology rather than a completed silicon or FPGA benchmark. Detailed ACE cache-state implementation, snoop filtering, full AXI4 burst support, clock-domain crossing, quality-of-service behavior, formal property sets, post-synthesis timing, area, and power measurements are outside the evidence supplied in the source document. These items should be addressed before making implementation-level performance claims.

IX. FUTURE WORK

Future development should refine the transaction model into synthesizable Verilog or SystemVerilog and verify it using tools such as ModelSim, Vivado Simulator, or QuestaSim. The design can then be synthesized on an FPGA to obtain resource utilization, critical-path timing, maximum frequency, and power estimates. A UVM environment can extend the existing generator-monitor-scoreboard structure with reusable sequencers, agents, assertions, and functional coverage.

The interconnect can also be extended to support full AXI4 burst transactions for high-bandwidth data paths, while retaining AXI4-Lite for control access. The coherency subsystem can be improved through cache-state tracking and snoop filtering, and power/area can be optimized through clock gating, power gating, and reduced switching. At larger scale, the same transaction concepts can be migrated toward a Network-on-Chip topology with multiple AXI/ACE-compatible endpoints.

X. CONCLUSION

This paper presented a structured AMBA AXI4-Lite/ACE interconnect architecture for transaction-based SoC integration. The design separates lightweight memory-mapped control communication from coherency-oriented shared-memory communication while unifying both within a modular interconnect framework. Address decoding, arbitration, independent read/write control, snoop support, coherency management, and explicit response handling address the principal communication challenges of heterogeneous SoCs.

A transaction-level verification methodology was defined using generators, drivers, monitors, protocol checkers, scoreboards, and coverage. The proposed verification matrix covers successful reads and writes, invalid addresses, simultaneous master requests, slave back-pressure, response errors, and coherency-oriented accesses. Because the supplied project source does not contain quantitative RTL or FPGA results, this manuscript intentionally limits its conclusions to architecture and verification methodology. The design nevertheless provides a clear foundation for future RTL implementation, UVM verification, FPGA synthesis, and performance characterization.

REFERENCES

[1] Arm Limited, AMBA AXI and ACE Protocol Specification, ARM IHI 0022H, Cambridge, U.K., 2017.
[2] Arm Limited, Introduction to AMBA AXI4, Document 102202_0100_01, Cambridge, U.K., 2020.
[3] A. Stevens, “Introduction to AMBA 4 ACE and big.LITTLE Processing Technology,” Arm White Paper, 2011.
[4] K. Keshamoni, J. MD, and D. S. Reddy, “Deep Learning-Based Spectrum Sensing for Cognitive Radio Communication Networks,” *Res Militaris*, vol. 13, no. 1, 2023.
[5] A. Kriouile and W. Serwe, “Using a Formal Model to Improve Verification of a Cache-Coherent System-on-Chip,” INRIA/CADP, 2015.
[6] Accellera Systems Initiative, SystemC Transaction-Level Modeling Overview, 2024.
[7] S. Pasricha, N. Dutt, and M. Ben-Romdhane, “Extending the Transaction Level Modeling Approach for Fast Communication Architecture Exploration,” in Proc. 41st DAC, 2004, pp. 113–118.
[8] S. K. Swain and K. Mahapatra, “Modeling Method to Develop an AMBA AXI4 Bus Interconnect: A Survey,” IJERT, vol. 4, no. 4, pp. 899–903, 2015.
[9] S. S. Bhople and P. N. Chatur, “TLM Based AMBA AXI4 Protocol Implementation Using SystemC,” IJISET, vol. 2, no. 5, pp. 1061–1066, 2015.
[10] Javeed MD, Nagaraju R, Chandrasekaran R, et al. Brain tumor segmentation and classification with hybrid clustering, probabilistic neural networks. Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology. 2023;45(4):6485-6500. doi:10.3233/JIFS-232493, Google Scholar
[11] P. V. S. Ram and K. S. Rao, “Design of AMBA AXI4-Lite for Effective Read/Write Transactions with a Customized Memory,” International Journal of Engineering Research and Applications, vol. 12, no. 8, pp. 45–51, 2022.
[12] S. R. Gade and P. S. Patil, “Design and Verification Environment for AMBA AXI Protocol for SoC Integration,” International Journal of Engineering Research and Applications, vol. 4, no. 5, pp. 152–156, 2014.
[13]. Richitha and J. MD, “Fault space transformation: A generic approach for fault-tolerant parallel filters,” *Int. J. Innovation Eng. Res. Manag.*, vol. 11, Special Issue 07, Paper ID-IJIERM-XI-VII, pp. 30–36, Jul. 2024. Google Scholar
[14] C. Kumar and M. Thottappan, “A Design of AMBA AXI4-Lite ACE Interconnect Protocol for Transaction-Based SoC Design Techniques Integration,” International Journal of Engineering and Computer Science, vol. 6, no. 6, pp. 21870–21874, 2017.

[15] K. Lahiri, A. Raghunathan, and S. Dey, “Design Space Exploration for Optimizing On-Chip Communication Architectures,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 6, pp. 952–961, 2004.