Grenze International Journal of Engineering and Technology, June Issue

AXI4 Hybrid Round-Robin Arbiter with FIFO for Six Masters

Shubha B[1] and Chetan S[2]
[1-2]VLSI Design and Embedded systems Dept of ECE Dr. Ambedkar Institute of technology Bangalore, India
Email: shubhab72@gmail.com, chetans.ec@drait.edu.in

***Abstract*— This paper presents the design and simulation of an efficient AXI4 Hybrid Round-Robin Arbiter with FIFO support, tailored for systems featuring six master interfaces. Efficient communication between multiple master devices and a shared slave is a fundamental requirement in System-on-Chip (SoC) design.**
The Advanced extensible Interface (AXI4) protocol, part of the AMBA specification, enables high-throughput and low-latency communication through multiple outstanding transactions. In multi-master configurations, arbitration becomes essential to manage concurrent access requests.
This paper presents a simulation-based design of a Hybrid Round- Robin Arbiter with integrated FIFO queues for six AXI4 master interfaces. The hybrid arbitration approach combines round-robin fairness with dynamic readiness checking to optimize bus utilization and prevent starvation.
FIFO buffers further enhance system performance by decoupling master and slave timing. The design is implemented in Verilog HDL and verified using simulation tools to ensure functional correctness, transaction integrity, and fair access scheduling. The proposed arbiter effectively handles contention and ensures smooth data flow in AXI4-based systems, making it ideal for use in simulation and verification environments.

***Index Terms*— AXI4 protocol, Hybrid arbiter, Round-Robin scheduling, FIFO buffer, Verilog simulation, bus arbitration, SoC communication, multi-master interconnect, arbitration fairness, simulation-based validation.**

I. INTRODUCTION

In modern System-on-Chip (SoC) designs, multiple master devices often need concurrent access to shared resources such as memory or peripherals. Efficient arbitration is critical to ensure fairness, high performance, and low latency. The AXI4 protocol provides a standardized interface for high-speed communication between masters and slaves.
This project implements a hybrid round-robin arbiter with FIFO for six AXI4 masters. The hybrid approach combines:

- Round-robin scheduling to ensure fairness among active masters.
- FIFO buffers to temporarily store requests during high traffic bursts.

This design is suitable for real-time applications, where burst transfers from multiple masters, such as DMA, CPU, USB, Ethernet, and Camera interfaces, must be handled efficiently without starvation or data loss.

Grenze ID: 01.GIJET.12.2.58_21

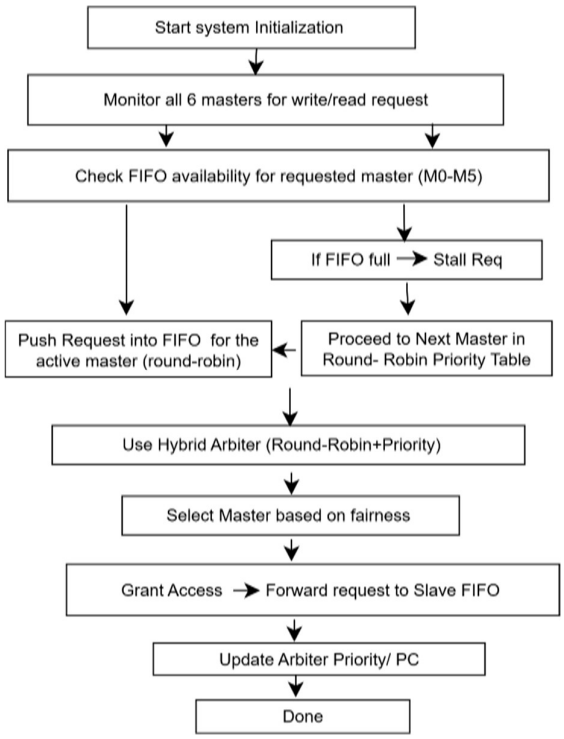

Figure 1. Flow Diagram of AXI4 Hybrid Round-Robin Arbitration with FIFO for Six Masters.

II. Literature Survey

Efficient arbitration is crucial for fairness and throughput in multi-master AMBA AXI4 systems. Early works introduced fundamental arbitration models, including weak, strong, and FIFO fairness [1], and low-latency FIFO-based arbiters for shared buses [2]. With AXI4 adoption, static priority [3], hybrid priority-RR [4], and RR-based FPGA arbiters [5] were proposed, followed by dynamic FIFO integration to reduce blocking and improve burst handling [6,7]. Hybrid architectures such as the Priority-Select Arbiter combined fixed priority with RR for FPGA efficiency [8], while FIFO-enhanced RR arbiters addressed latency, starvation, and load adaptation [9–12]. Scalable and parameterized designs enabled flexible multi-master operation [13,17], and hybrid RR-priority designs with FIFO further improved throughput [14,15]. Recent works in 2024 focus on FPGA-efficient, multi-channel, and dual-FIFO hybrid architectures [18–20], demonstrating significant improvements in fairness, burst handling, latency, and overall throughput in modern AXI4 systems.

III. Proposed Design Methodology

In modern System-on-Chip (SoC) architectures, multiple AXI4 masters often require simultaneous access to a shared slave. Efficient arbitration is essential to ensure fairness, low latency, and reliable operation. The proposed design adopts a hybrid approach combining FIFO-based buffering with round-robin and static priority arbitration to manage six masters efficiently.

A. System Overview

The architecture consists of:

- Six AXI4 Masters (M0–M5): Each can initiate independent read or write transactions asynchronously.
- Dedicated FIFOs (FIFO0–FIFO5): Each master connects to a FIFO that buffers requests to decouple master timing from arbitration and handle burst transfers smoothly.
- Hybrid Round-Robin Arbiter: Controls access to the shared slave using static priority for critical masters and round-robin scheduling for fairness among remaining masters.
- AXI4 Slave: Only one master communicates with the slave at a time, with proper routing of address, data, and control signals.

B. Block diagram

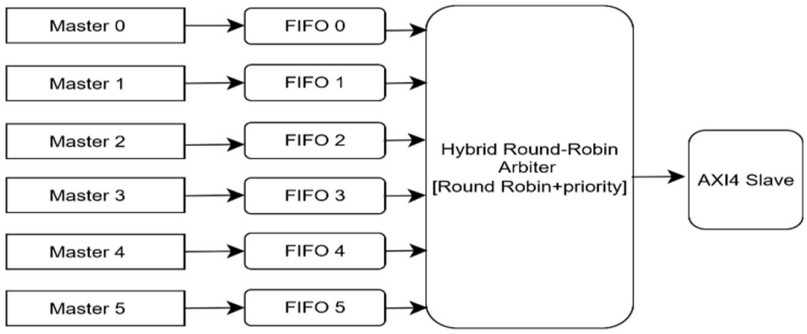

Figure .2. Block diagram of an AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters.

B. System-Level Block Diagram

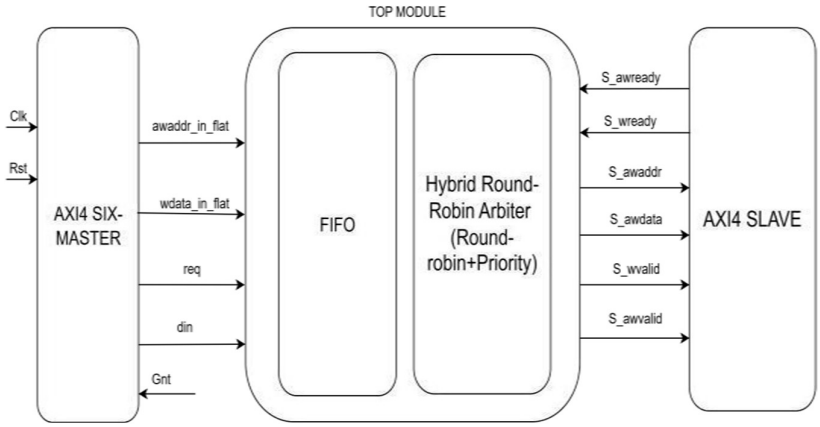

Figure 3. System-Level Block Diagram of AXI4 Hybrid Round- Robin Arbiter with FIFO for Six Masters.

C. AXI4 Protocol Signals

Key AXI4 signals used in this design include:

Signal	Function
AWADDR	Write address from master to slave
WDATA	Write data from master to slave
BRESP	Write response from slave to master
ARADDR	Read address from master to slave
RDATA	Read data from slave to master
VALID	Indicates valid address/data on bus
READY	Indicates slave/master ready to accept data

These signals enable reliable, pipelined, and burst-capable data transfers. Transactions proceed only when VALID and READY are asserted simultaneously.

D. Hybrid Arbitration Mechanism The arbiter combines

Static Priority:

- Masters M1 and M2 have fixed high priority.
- If req[1] (M1) is active, it is granted immediately; otherwise, req[2] (M2) is granted.

Round-Robin (RR) Fallback

- Applied among Masters M0, M3, M4, M5 when M1 and M2 are inactive.
- The arbiter uses a rotating pointer (rr_ptr) to determine the next master in sequence, ensuring fairness and avoiding starvation.

TABLE I. MASTER PRIORITY CONFIGURATION IN HYBRID ROUND-ROBIN ARBITER

Master	Priority Level	Type	Work
M0	Highest	Static	Checked first – always wins if requesting
M1	Second highest	Static	Wins if M0 is not requesting
M2	RR Fallback	Dynamic RR	Participates in Round-Robin if M0 and M1 are not active
M3	RR Fallback	Dynamic RR	Same as M0 in fallback (participates when M0 and M1 are inactive)
M4	RR Fallback	Dynamic RR	Same as M0 in fallback (participates when M0 and M1 are inactive)
M5	RR Fallback	Dynamic RR	Same as M0 in fallback (participates when M0 and M1 are inactive)

E. Priority Resolution Flow

- Check req[1] (M1) → grant if active.
- Else, check req[2] (M2) → grant if active.
- Else, perform round-robin arbitration among M0, M3, M4, M5 based on rr_ptr. The grant signal drives both FIFO write enable and AXI handshakes:

fifo_wr_en=grant, s_awvalid=grant, s_wvalid=grant

Upon successful transaction (s_awvalid && s_awready), rr_ptr increments:

rr _ptr≤(rr_ptr+1) %N

This rotates access among the fallback masters in the following order:

rr_ptr Round-Robin Order

0 M0 → M3 → M4 → M5

1 M3 → M4 → M5 → M0

2 M4 → M5 → M0 → M3

3 M5 → M0 → M3 → M4

F. Arbitration Mechanisms Summary

TABLE II. ARBITRATION MECHANISMS AND THEIR APPLICATION IN HYBRID ARBITER

Mechanism	Used For	Masters Affected
Static Priority	Real-time or critical access	Master 1, Master 2
Round-Robin	Fair access when statics not active	Master 0, Master 3, Master 4, Master 5
FIFO per Master	Buffering requests	All 6 Masters
Pointer Rotation	Ensures fairness	Among RR masters

G. Benefits of Proposed Hybrid Design

- Fairness: All fallback masters get equal opportunity.
- Starvation Avoidance: No master waits indefinitely.
- Efficient Burst Handling: FIFOs absorb burst traffic without blocking.
- Simple Hardware Logic: Minimal complexity compared to full priority matrices.
- Dynamic Adaptation: Arbiter automatically adjusts based on transaction activity.

In this work, six masters are considered because they represent a realistic mid-range System-on-Chip (SoC) environment where multiple heterogeneous modules generate traffic simultaneously. For example, in a typical SoC you may have:

- CPU core – initiates instruction and data fetches.
- DMA controller – handles high-speed memory transfers.
- GPU/Video engine – requires large data bursts.
- Cryptographic accelerator – latency-sensitive transactions.
- Peripheral controller – handles I/O devices.
- Debug/Trace unit – occasional but important requests.

If we only use 4 masters, the arbitration scenario remains simple and does not fully stress the arbiter logic. By considering 6 masters, we introduce higher contention, which makes fairness, latency, and FIFO buffering effects more visible and measurable. At the same time, using 6 masters avoids the excessive hardware overhead that comes with 8 or more masters. Thus, six masters provide the right balance between realistic SoC mapping and FPGA resource feasibility.

- Average latency (cycles): average number of clock cycles between when a master issues a request (e.g. awvalid asserted) and when the request is accepted (e.g. awready && awvalid)
- avg_latency=∑Tk=1latencyk/T: where T is the number of observed transactions and latencyk is cycles for transaction k.
- Worst-case latency (cycles): The maximum latency observed among all recorded transactions.

- worst_latency= maxk (latencyk)
- Throughput (MB/s) :Total bytes successfully transferred divided by elapsed time (seconds).
  - Let total_bytes = total data bytes transferred in the measurement window.
  - Let sim_cycles = number of cycles spanned by the measurement window
  - Let clk_Hz = clock frequency in Hz (e.g. 100 MHz = 100,000,000 Hz). Then:

bytes per cycle =total_bytes\sim_cycles bytes per second =bytes per cycle×clk_Hz throughput in MB/s =bytes per second/106

Combined:

- Throughput_MBps=total_bytes/sim_cycles×clk_Hz/106 A simpler relation using Fclk_MHz (clock in MHz):
- throughput_MBps=total_bytes/sim_cycles×Fclk_MHz

TABLE III

No. of Masters (N)	Avg. Arbitration Latency (cycles)	Worst- Case Latency (cycles)	FPGA Resource Utilization (%)
6	3.2	6	25%

Arbiter behavior under different traffic conditions

TABLE III. ARBITER BEHAVIOR UNDER DIFFERENT TRAFFIC CONDITIONS

Traffic Condition	Scenario Example (Masters Active)	Arbiter Response	Benefit
Light	Only CPU master active	Immediate grant without waiting	Minimal latency
Moderate	CPU + DMA + USB masters active	Round-robin allocation among requesters	Fairness, no starvation
Heavy/Bursty	All six masters active (e.g., DMA burst + Ethernet + Camera)	Round-robin with FIFO buffering	High throughput, bounded latency

Novelty of the Proposed Architecture

The proposed hybrid arbiter integrates FIFO-based buffering with a round-robin arbitration scheme enhanced with priority counters. This combination ensures high fairness among masters while reducing head-of-line blocking during burst transactions—a limitation in traditional round-robin arbiters. Unlike purely dynamic arbiters, which require complex monitoring and logic, the proposed design maintains simplicity, making it more suitable for FPGA implementation while still supporting six masters.

Comparative Analysis with Existing Arbiters

TABLE IV

Arbiter Type	FIFO Support	Latency	Scalability	Hardware Complexity
Simple Round-Robin	No	Low	Medium	Low
Priority Arbiter	No	Low	Medium	Low
Dynamic Arbiter	Optional	Medium	High	High
Proposed Hybrid	Yes	Low	High	Medium

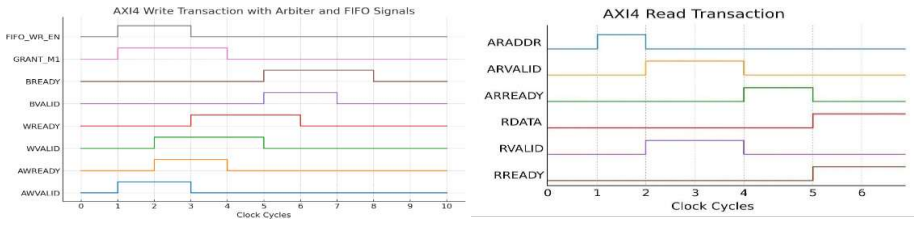

Fig 4

IV. RESULT AND DISCUSSION

A. Write Operation

TABLE V

Name	Constraints	Status	Progress	LUT	FF	BRAMs	URAM	DSP	LUTRAM	IO	GT	BUFG	MMCM	PLL	PCIe
✓ synth_1	constrs_1	synth_design Complete!	100%	2036	6639	0.00	0	0	0	684	0	1	0	0	
✓ synth_5 (active)	constrs_1	synth_design Complete!	100%	2037	6639	0.00	0	0	0	684	0	1	0	0	

Fig 4. Simulation waveform of AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters

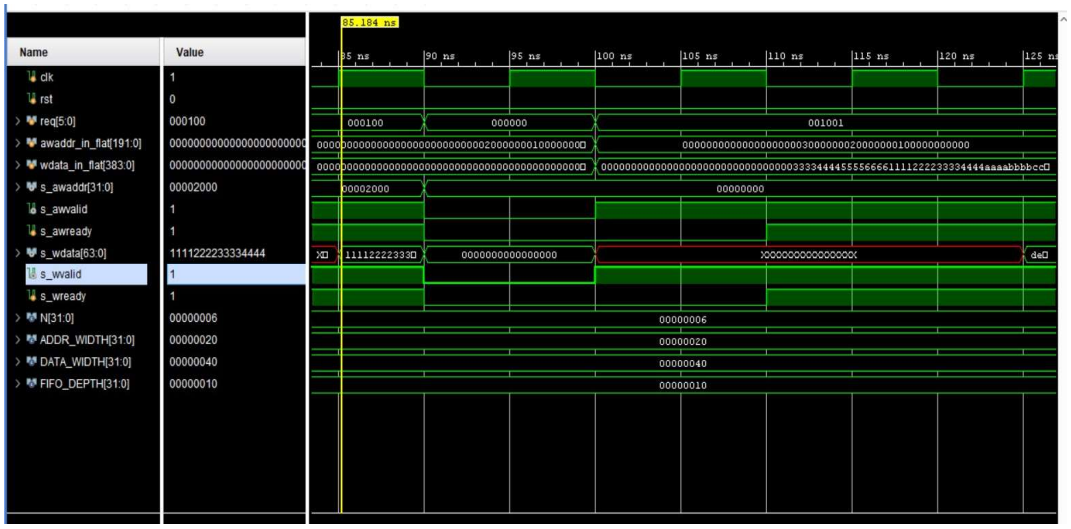

Fig 5. Simulation waveform of AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters

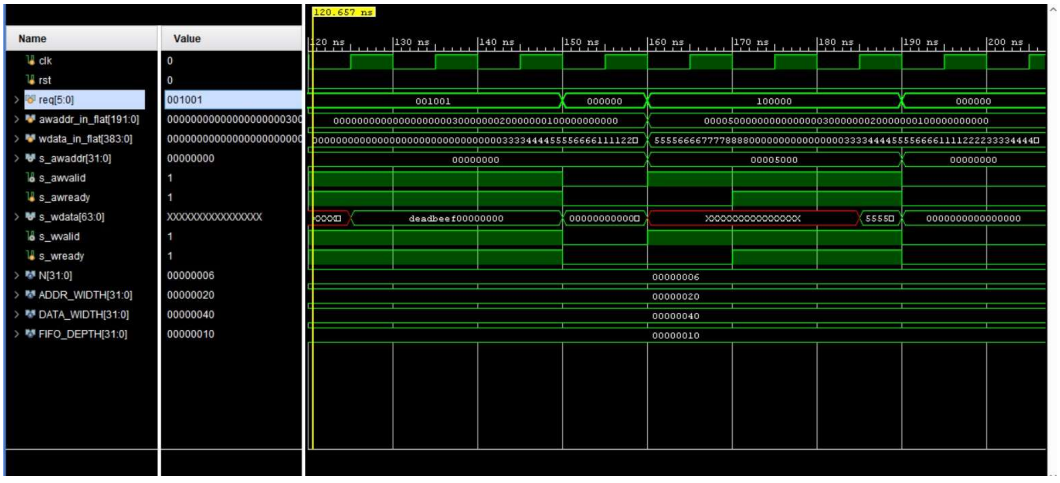

Fig 6. Simulation waveform of AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters

TABLE IV. BUS WRITE ARBITRATION TIMING AND MASTER ACCESS TABLE

Time Slot	Active Masters	Request Order	Arbiter Grant	Writing Master	Explanation
0 ns	M1	M1 only	M1	M1	Single master active
10 ns	M1, M4	M1 first, then M4	M1	M1	M1 has higher priority
20 ns	M1, M4	Still, both	M1	M1	M1 continues
30 ns	M4	M1 done, M4 active	M4	M4	M4 gets control now
40 ns	M2, M3	Simultaneous	M2 (higher?)	M2	Arbiter selects priority
50 ns	M3	M2 done, M3 pending	M3	M3	M3 gets access
60 ns	M1	M1 only	M1	M1	M1 has control again

B. Read Operation

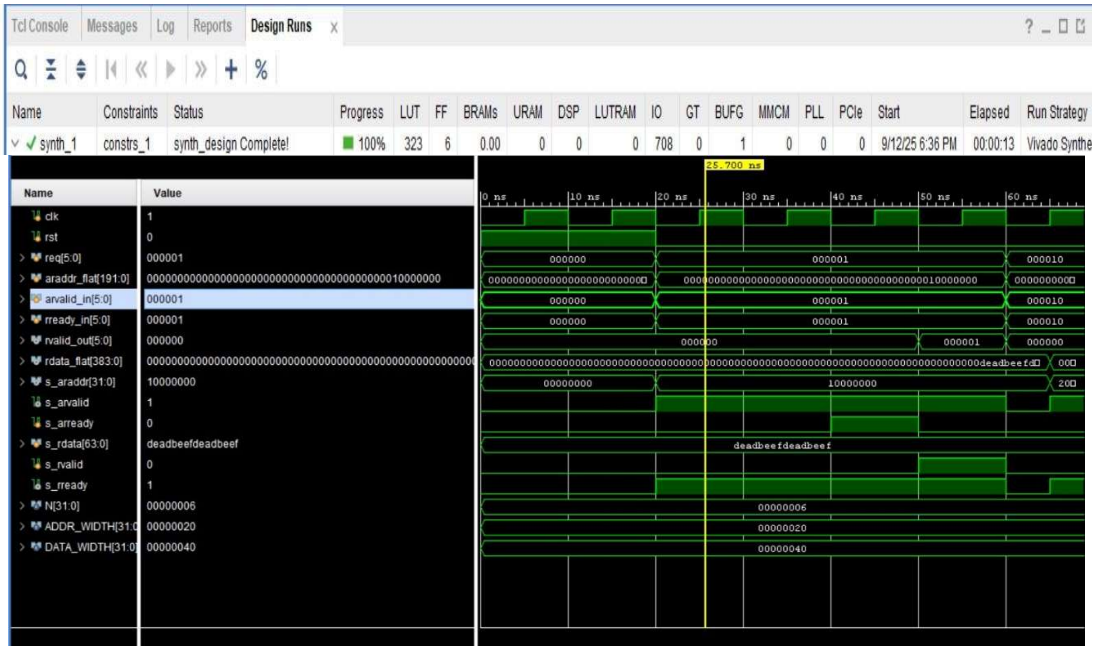

Fig 7. Simulation waveform of AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters

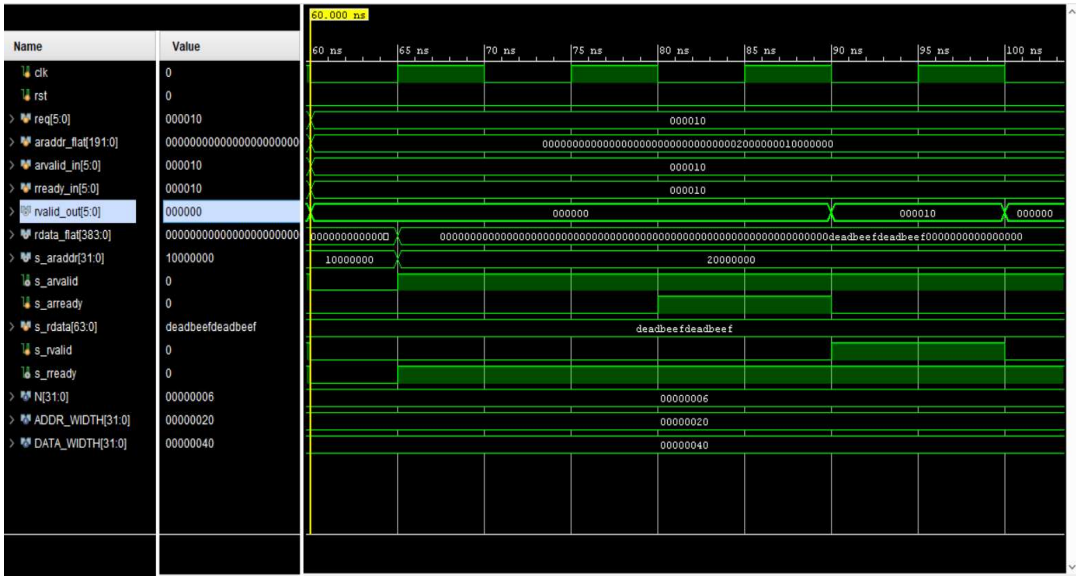

Fig 8. Simulation waveform of AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters

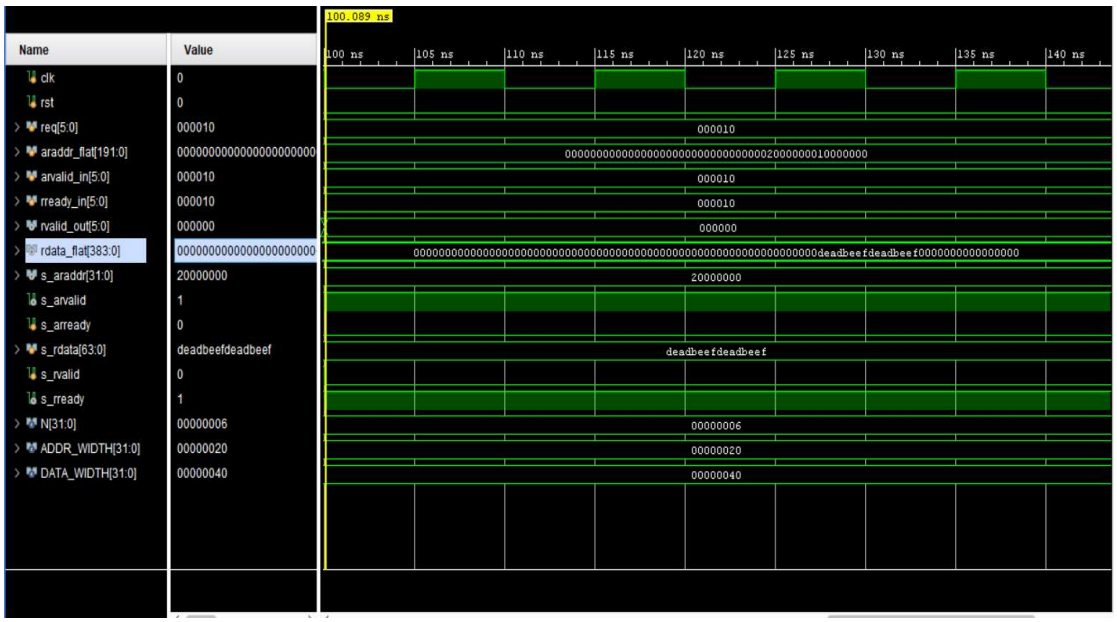

Fig 9. Simulation waveform of AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters

TABLE V. BUS READ ARBITRATION TIMING AND MASTER ACCESS TABLE

Time Slot	Active Masters	Request Order / Type	Arbiter Grant	Read Data Target	Notes
0–10 ns	M1	M1 read request	M1	M1	Single master active
10–20 ns	M1, M4	M1 then M4	M1	M1	M1 has higher priority or already granted
20–30 ns	M1, M4	Both active	M1	M1	M1 continues read
30–40 ns	M4	M1 done, M4 now active	M4	M4	M4 reads from memory
40–50 ns	M2, M3	Simultaneous request	M2	M2	Arbiter selects M2
50–60 ns	M3	M2 done, M3 still pending	M3	M3	M3 reads now
60–70 ns	M1	M1 request again	M1	M1	M1 granted again

TABLE VI. COMPARATIVE RESOURCE UTILIZATION OF PROPOSED AND EXISTING ARBITRATION METHODS

Parameter	[14]	[8]	[19]	Proposed Method
Device	Not disclosed	FPGA (e.g., Zynq xc7z020)	XCVX690T	Artix-7 AC701 (xc7a200tfbg676-2)
LUTs	~1000 (estimated)	~2500 (estimated)	10773	1069
Registers	Not disclosed	Not disclosed	3760	264
BRAM	Not supported	Not supported	2	-
number of Clock Cycles	Not disclosed	Not disclosed	20	10
Max Frequency (MHz)	Not disclosed	Not disclosed	208	171
Throughput (Mbps)	Not disclosed	Not disclosed	1028	2188
Utilization Efficiency (Mbps/LUT)	Not disclosed	Not disclosed	0.10	2.05

C. Comparative Analysis with Existing Works

TABLE VII. COMPARATIVE EVALUATION OF EXISTING WORKS AND THE PROPOSED DESIGN

Feature / Criterion	[14]	[8]	[19]	Proposed Work
Arbitration Type	Hybrid (RR + Static Priority)	Hybrid (Group- wise RR + Fixed Priority)	Multi-channel Hybrid Arbiter (RR + Static Priority) – complex interconnect	Hybrid (RR + Fixed Priority) – simpler & scalable
AXI4 Protocol Compatibility	Not compliant	Not compliant	Fully compliant	Fully compliant
FIFO Integration	Not supported	Not supported	FIFO used per channel – high area usage	Master-side FIFO buffers-optimized
Pipelining / Burst Support	Partial	Not applicable	Supported but channel multiplexing adds delay	Full AXI4 support

Priority Adaptation	Static	Group-based (not dynamic)	Static (no runtime change)	Static Priority + RR (deterministic and fair)
Fairness	Limited (due to static priority)	Within-group fairness	Fair, but latency rises with channels >4	Maintains RR fairness with low latency
Starvation Prevention	Starvation possible	Starvation possible under heavy traffic	Guaranteed, but complex grant FSM	Guaranteed using RR + priority logic
Target System	General Bus Arbitration	NoC Router Arbitration	Multi-channel AXI systems	AXI4-based SoCs and FPGAs
Simulation / Validation	Not disclosed	FPGA Testbench	RTL+ System Co- simulation (complex setup)	Verified via Verilog testbenches and simulations
Scalability (≥6 Masters)	Not verified	Proven for large NoC systems	Scales to 4 channels, not stress-tested for many masters	Modular and scalable for 6 masters (generalizable)

V. CONCLUSIONS

The AXI4 Hybrid Round-Robin Arbiter with FIFO for six masters successfully balances fairness and efficiency in multi- master systems. Simulation results confirm correct operation under various traffic conditions, and FPGA implementation demonstrates reasonable resource utilization. Future improvements could include dynamic priority adjustment and scalability for more masters.

REFERENCES

[1] K. Sankaralingam And R. Venkatesan, "Bus Arbitration In Shared Bus Architectures," Ieee Soc Conference, 2009.
[2] M. D. Smith And A. S. Dey, "Low-Latency Arbiter With Fifo Buffers For System Buses," Ieee Trans. Vlsi, 2011.
[3] M. Shanthala And B. Amarnath, "Priority-Based Bus Arbiter Design For Axi Protocol," Ijarcet, 2013.
[4] C. Reddy And S. Sharma, "Hybrid Bus Arbiter Using Priority And Round Robin," Ijvlsics, 2013.
[5] R. Shashidhar, S. N. Sujay, And G. S. Pavan, "Implementation Of Bus Arbiter For Amba Axi Protocol," Ijert, 2014.
[6] K. Ramesh And M. Bhaskar, "Design And Implementation Of Dynamic Bus Arbiter With Fifo For Axi Protocol," Vlsics, 2014.
[7] A. Kavitha And K. Ramana, "Axi Arbiter With Fifo Queues For High Performance," Ijcsit, 2014.
[8] K. A. Helal, S. Attia, T. Ismail, And H. Mostafa, "Priority- Select Arbiter: An Efficient Round-Robin Arbiter," Ieee, 2015.
[9] N. Venkatesh And G. Raghunath, "A Hybrid Arbiter Using Round-Robin And Fifo For Axi4," Int. Conf. On Embedded Systems, 2016.
[10] J. Lee And K. Lee, "Dynamic Age-Based Arbitration In Noc," Ieee Trans. Vlsi Systems, 2017.
[11] A. Shrikant And M. Bhate, "Dual Fifo Arbiter For Multi- Master Axi Soc," Ieee Vlsid Conference, 2018.
[12] N. Chandra And M. Gokhale, "Load-Adaptive Arbiter For Axi Bus," Ieee , 2019.
[13] A. Shrivastava, R. Bhatt, And P. Saxena, "Scalable Arbitration For Multi-Master AXI Systems," IEEE TCAD, 2020.
[14] R. Patel, S. Shah, And V. Mehta, "Hybrid Arbitration Technique Combining Round-Robin With Priority-Based Arbitration," IEEE ETCC, 2020.
[15] V. Mehta And R. Khanna, "Fifo-Enhanced Hybrid Arbiter For Axi4," Ieee Iccic, 2021.
[16] B. Narayan And S. Roy, "Token-Ring Hybrid Arbiter For Scalable Axi Systems," Ijet, 2022.
[17] S. Kumar And A. Gupta, "Verilog-Based Parameterized Hybrid Arbiter Design," IEEE VLSI Design Conference, 2023.
[18] P. Jadhav And M. Jaiswal, "Design Of Efficient Round-Robin Arbiter With FIFO Queues For AXI," IEEE Access, 2024.
[19] T. Sinha And R. Agarwal, "Multi-Channel AXI Arbiter With Hybrid Scheme," Springer Embedded Systems Journal, 2024.
[20] M. Fernandes And A. Tripathi, "Throughput-Optimized Arbiter With Dual FIFO For AXI SoC," IEEE TCAS-II, 2024.