Design and Verification of AXI4-Lite Protocol using SystemVerilog

Sai Charan Teja Balne, saibalne@pdx.edu
Sai Tarun Mokkala, saitarun@pdx.edu
Akshay Kumar Medishetti , amedishe@pdx.edu
Monesh Kareti, monesh@pdx.edu

ECE 571
Introduction to SystemVerilog for design and Verification
Venkatesh Patil

Electrical and Computer Engineering Department
Maseeh College of Engineering and Computer Science

Portland State UNIVERSITY

Contributions

Sai Charan Teja Balne :

- Verification of the Write Transaction.
- Implementation of Random constraints for the testbench.
- Report Work.

Akshay Kumar Medishetti :

- Design and implementation of the AXI4-Lite Write Channels.
- FSM implementation.

Sai Tarun Mokkala :

- Design and implementation of the AXI4-Lite Read Channels.
- Collected References.

Monesh Kareti

- Verification of the Read transaction and implemented the testbench connectivity. Controlled use of interface and clocking blocks.
- PPT Presentation

Contents

Introduction

- **Definition**
  A streamlined subset of the AMBA AXI4 protocol
- **Application**
  Ideal choice for implementing Control and Status Registers (CSRs) in peripheral IP cores
- **Purpose**
  Tailored for low-throughput, memory-mapped register access
- **Key Constraint**
  All transactions are restricted to a burst length of one, simplifying the interface

Background: Advantages of AXI4-Lite

- **Decoupled Channels**

Read and write channels are separate, allowing for concurrent operations and improved bus utilization and throughput.

- **Separate Address/Data Phases**

The address and data phases for write transactions are decoupled, offering greater flexibility and the potential to hide latency.

- **Scalability and Integration**

As part of the AXI family, it integrates seamlessly with high-performance AXI interconnects, simplifying system integration.

Architecture Overview:

The Five Channels:

The AXI4-Lite protocol is based on five independent channels, each utilizing a two-way VALID/READY handshake for reliable data transfer.

Write Channels (Master to Slave)

Write Address (AW): AWVALID, AWADDR, AWREADY

Write Data (W): WVALID, WDATA, WSTRB, WREADY

Read Channels (Master to Slave)

Read Address (AR): ARVALID, ARADDR, ARREADY

Response Channels (Slave to Master)

Write Response (B): BVALID, BRESP, BREADY

Read Data (R): RVALID, RDATA, RRESP, RREADY

Implementation Details: Parameters and Memory

Configurable Parameters

Parameter	Description	Default Value
ADDRESS_WIDTH	Defines the width of the address bus	32
DATA_WIDTH	Defines the width of the data bus (32 or 64)	32
STRB_WIDTH	Number of byte-lanes (DATA_WIDTH / 8)	4

Internal Memory Structure

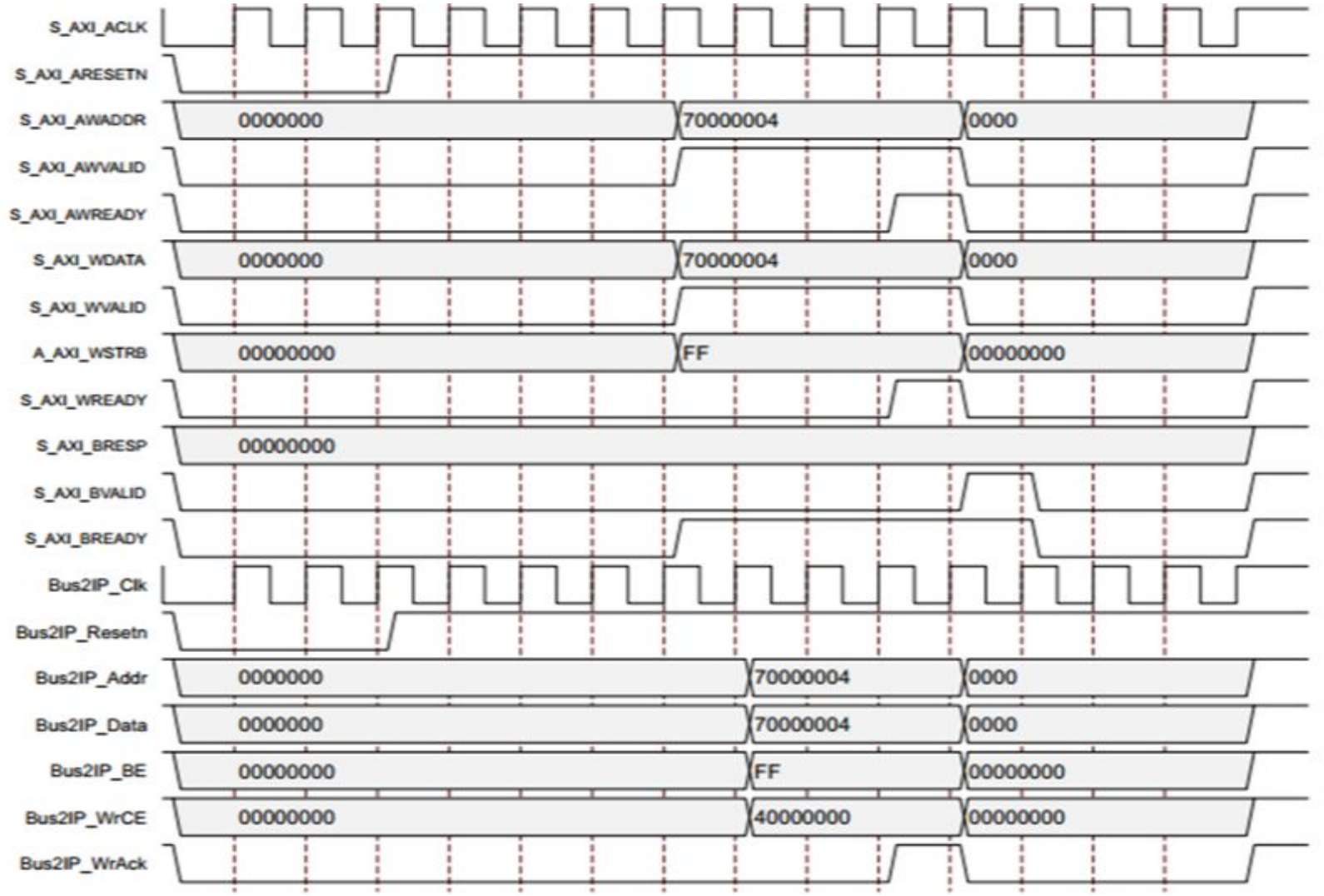
A simple register array serves as the memory space for the Control and status register(CSR'S):

logic [DATA_WIDTH-1:0] mem[ADDRESS_WIDTH-1:0]

Byte-Enable Support

The WSTRB signal enables partial register updates, allowing the master to write only specific bytes within a register for flexible data manipulation.

AXI4-Lite Write Transaction



Write Channel State Machine (FSM)

Purpose

Manages synchronization of independent AW and W channels to ensure write operations occur only after both address and data are received.

States

- NONE (Idle)
- GOT_W (Data)
- GOT_AW (Addr)
- GOT_BOTH (Ready)

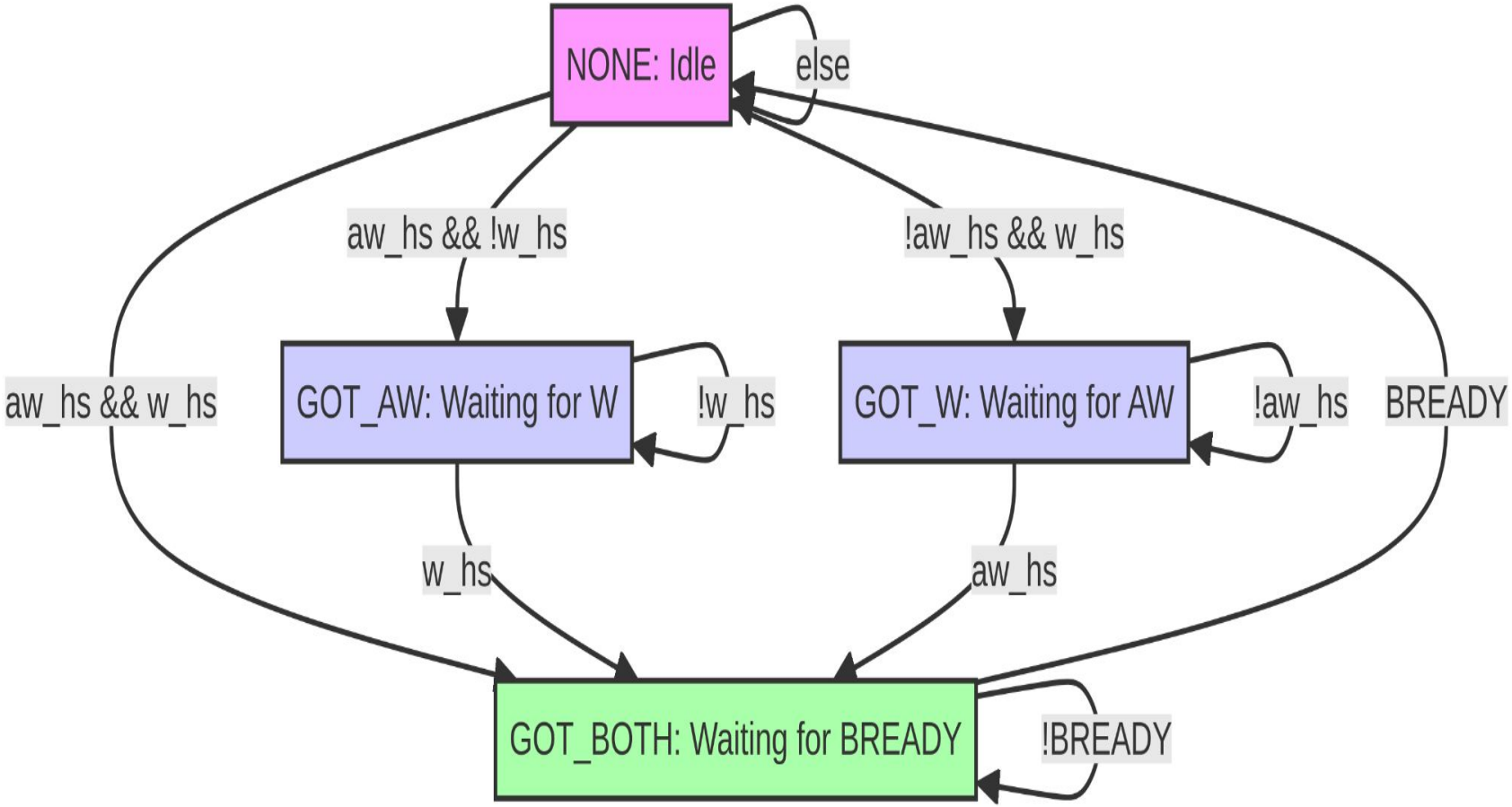
Transition Logic

Governed by handshakes: aw_hs (AWVALID & AWREADY) and w_hs (WVALID & WREADY).

Action

Memory update and BVALID assertion occur when the FSM transitions into the GOT_BOTH state.

Write Channel State Machine (FSM)



Read Channel Logic

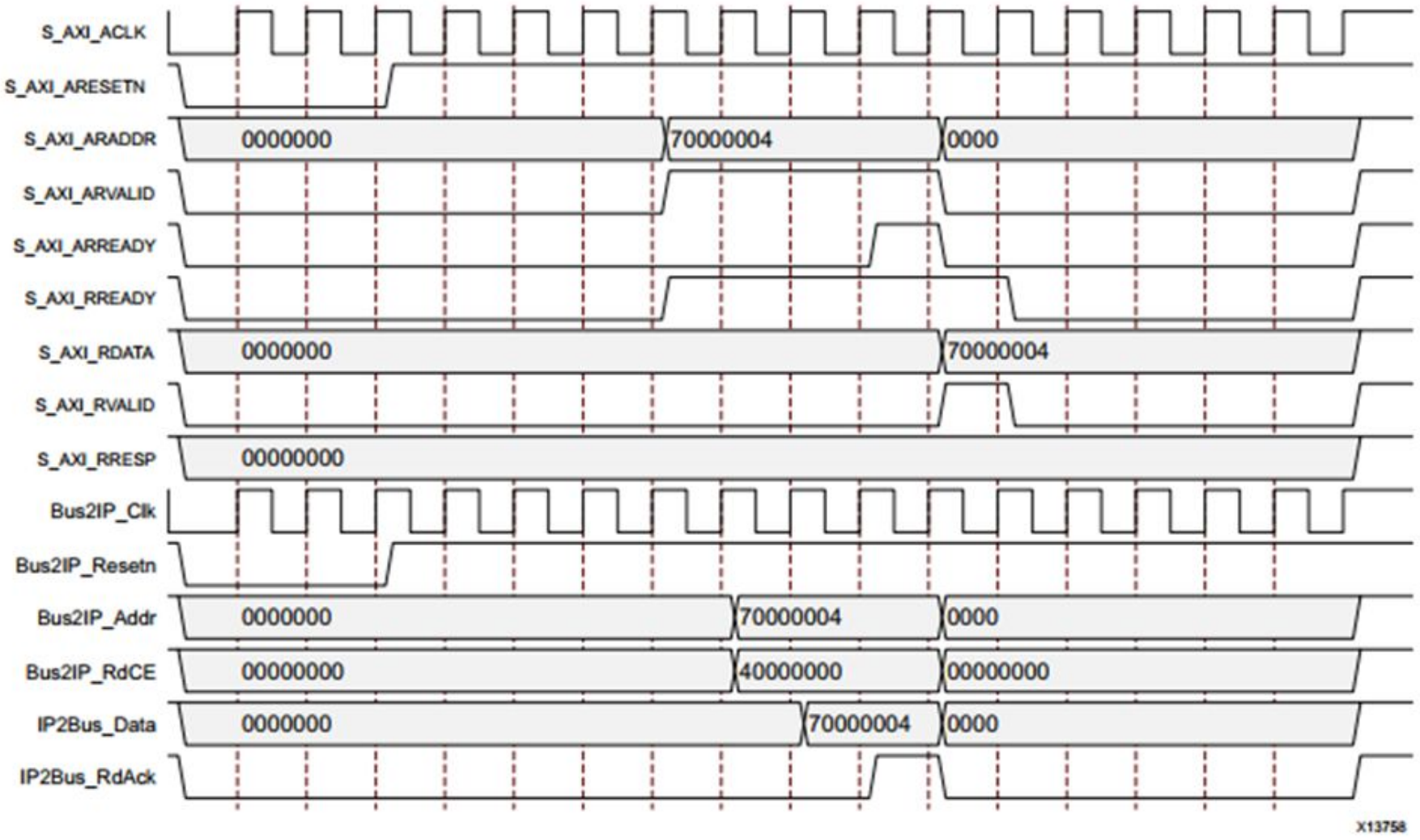
Initiation
The read transaction begins when the master asserts ARVALID.

Slave Action
The slave latches the address (latched_r_addr) and prepares the data for the Read Data (R) channel.

Standard Read
If no concurrent write is pending, the data is read directly from the internal memory array: RDATA ≤ mem[latched_r_addr].

Error Handling
The slave performs address decoding. If the address is out of the NUM_REGS range, a SLVERR (Slave Error) response is asserted on the R channel.

AXI4-Lite Read Transaction



Read-After-Write Forwarding Mechanism

Critical Requirement

Ensures data consistency when a read targets a register currently being written to.

Mechanism

Bypasses the single-cycle delay of the memory update by forwarding the pending write data.

Condition for Forwarding

- Write FSM is in the GOT_BOTH state
- Latched read address matches latched write address

Data Construction

- Bytes with WSTRB asserted: forwarded from WDATA
- Bytes with WSTRB de-asserted: read from memory

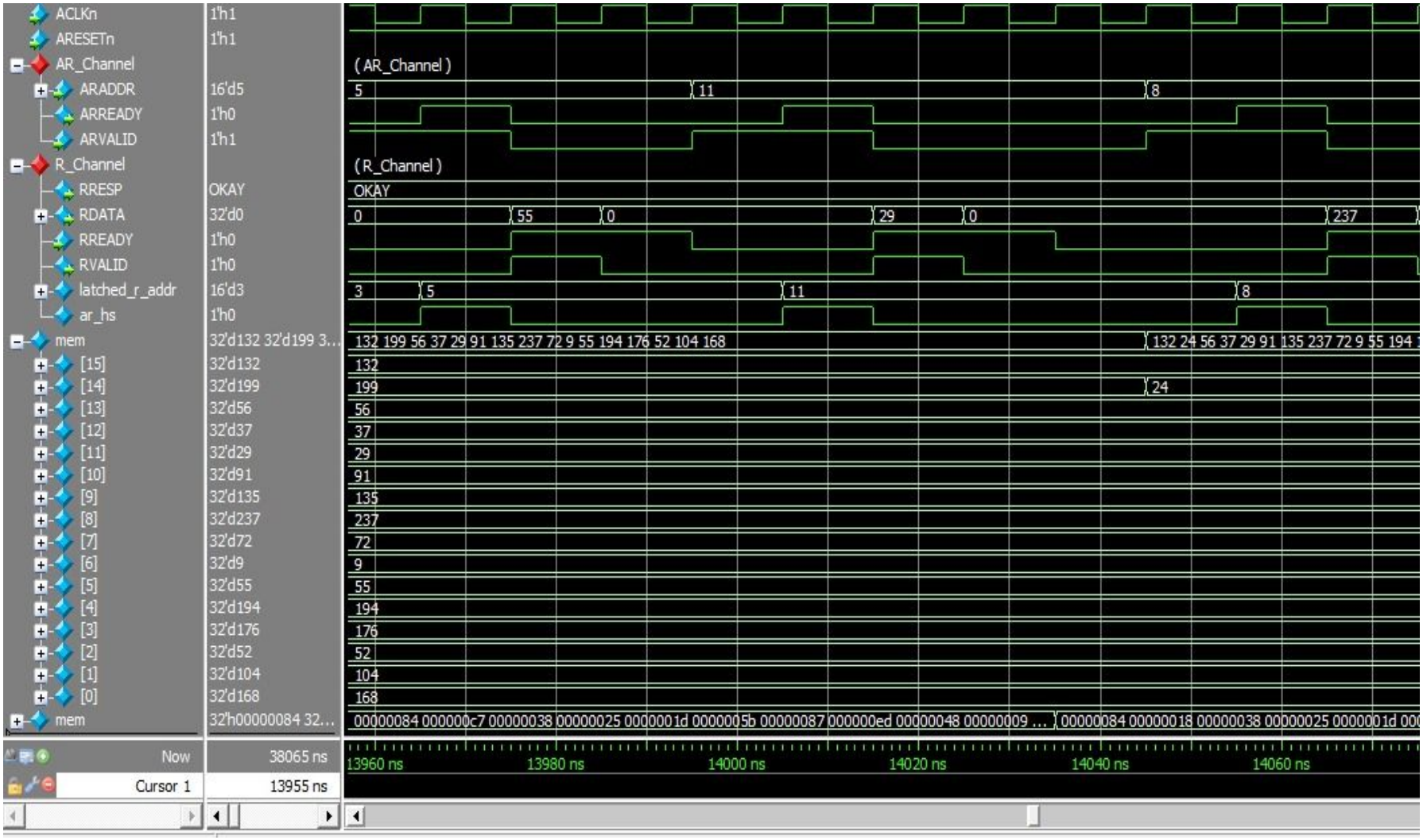
Simulation Results

Transcript

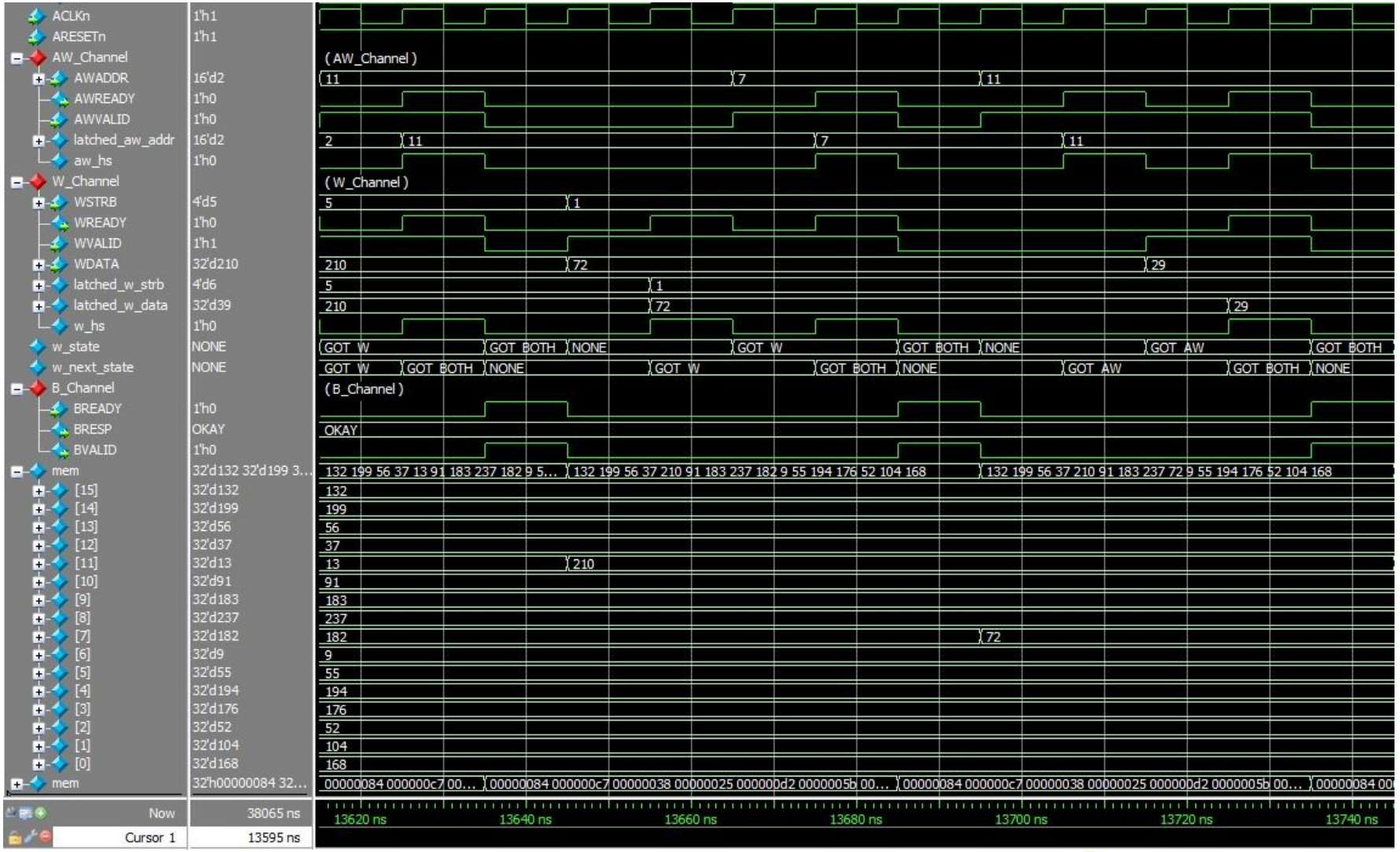
```
# The value of mem = 226 222 125 157 201 61 148 150 59 0 195 234 243 211 64 207, rdata - 222
# Comparison of packet 998, finished at time : 30425
# Driving of packet 998 in AR Channel finished at time : 30435
# Driving of packet 998 finished at time : 30435
# Stimulus generation of packet 999 started at time : 30435
# Operation: OP_RW ,wr_operation : W_AW, Write Signals : AW_ADDR : 10, W_DATA : 51, W_STRB: 7 ,  READ Signals : AR_ADDR : 5
# Stimulus generation of packet 999 finished at time : 30435
# Driving of packet 999 started at time : 30435
# Driving of 999 in AR Channel started at time : 30435
# Comparison of packet 999, started at time : 30465
# The value of mem = 226 222 125 157 201 51 148 150 59 0 195 234 243 211 64 207, rdata - 195
# Comparison of packet 999, finished at time : 30465
# Driving of packet 999 in AR Channel finished at time : 30475
# Driving of packet 999 finished at time : 30475
# Stimulus generation of packet 1000 started at time : 30475
# Operation: OP_RD ,wr_operation : AW_W, Write Signals : AW_ADDR : 14, W_DATA : 231, W_STRB: 1 ,  READ Signals : AR_ADDR : 1
# Stimulus generation of packet 1000 finished at time : 30475
# Driving of packet 1000 started at time : 30475
# Driving of 1000 in AR Channel started at time : 30475
# Comparison of packet 1000, started at time : 30505
# The value of mem = 226 222 125 157 201 51 148 150 59 0 195 234 243 211 64 207, rdata - 64
# Comparison of packet 1000, finished at time : 30505
# Driving of packet 1000 in AR Channel finished at time : 30515
# Driving of packet 1000 finished at time : 30515
# Total Number of Transactions : 1000
# Total Write Transactions : 512
# Total Read Transactions : 528
# Total Read Hits : 528
# Total Read valid Rate : 100.000000
# ** Note: $finish    : axi_tb.sv(249)
#    Time: 30615 ns  Iteration: 2  Instance: /axi_top/axi_tb_inst
# 1
# Break in Module axi_tb at axi_tb.sv line 249

VSIM 3>
```

Read Transaction



Write Transaction



Conclusion

Achievement

Successfully implemented a highly compliant AXI4-Lite slave module in SystemVerilog with robust protocol handshake logic.

Impact

The design guarantees data consistency and flexibility, making it a solid foundation for integrating peripheral IP cores into AXI-based systems.

Key Features

Parameterized architecture, dedicated FSM for synchronization, byte-enable support, and critical read-after-write forwarding mechanism.

Future Work

Integration into larger SoC environments, performance benchmarking, and extension to support additional AXI4 features.

This implementation serves as a robust and reusable foundation for memory-mapped register access in complex digital hardware designs, ensuring reliable operation and seamless integration within modern SoC architectures.

References

- Arm. (2020). AMBA AXI and ACE Protocol Specification.
- Xilinx. (2020). AXI Reference Guide.