ICHIP -2025

Direct Memory Access Controller

TEAM _BHARVI

Ayush Yadav

Shivansh Mishra

Design Choices

1. **State Machine-Based Control:** Two separate state machines manage **read and write** operations to ensure data integrity and proper handshaking with the AXI bus.
2. **FIFO for Intermediate Storage:** A synchronous FIFO buffers the data between AXI read and write transactions to decouple the two operations.
3. **AXI-Lite Protocol:** The design uses AXI-Lite handshaking for reliable data transfers, ensuring compatibility with standard AXI interfaces**.**
4. **Configurable Transfer Length:** The length of the data transfer is controlled via an input signal (length), making the module flexible.
5. **Trigger-Based Execution:** The DMA transaction begins upon receiving a trigger signal, allowing external control of transfers.
6. **Synchronous Reset: T**he module supports a synchronous reset mechanism to ensure a clean start for the read and write operations.
7. **Byte-Address Calculation**: Since AXI expects byte addresses, word addresses must be converted before sending to the slave memory.

Aligned DMA Transfer

01
FIFO Usage

02
Handshaking Methodology

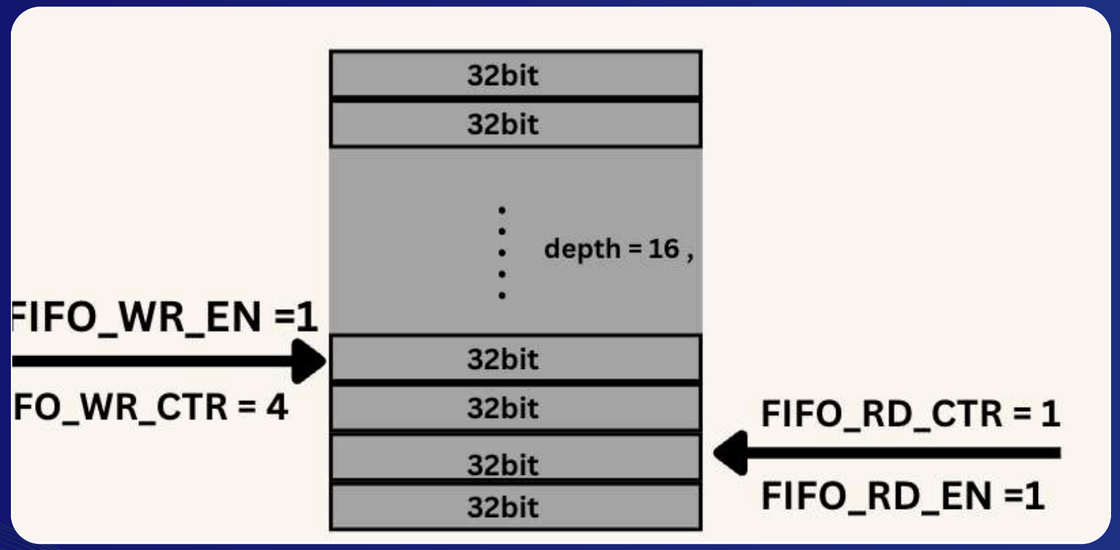
03
Read State Machine

04
Write State Machine

05
Simulation

FIFO USAGE

A 16-depth synchronous FIFO is used to buffer data between read and write transactions. The FIFO allows read and write operations to proceed independently, improving efficiency. The FIFO maintains



01

FIFO_WR_ENABLE
Signals when data should be written into the FIFO

02

FIFO_RD_EN
Signals when data should be read from the FIFO

03

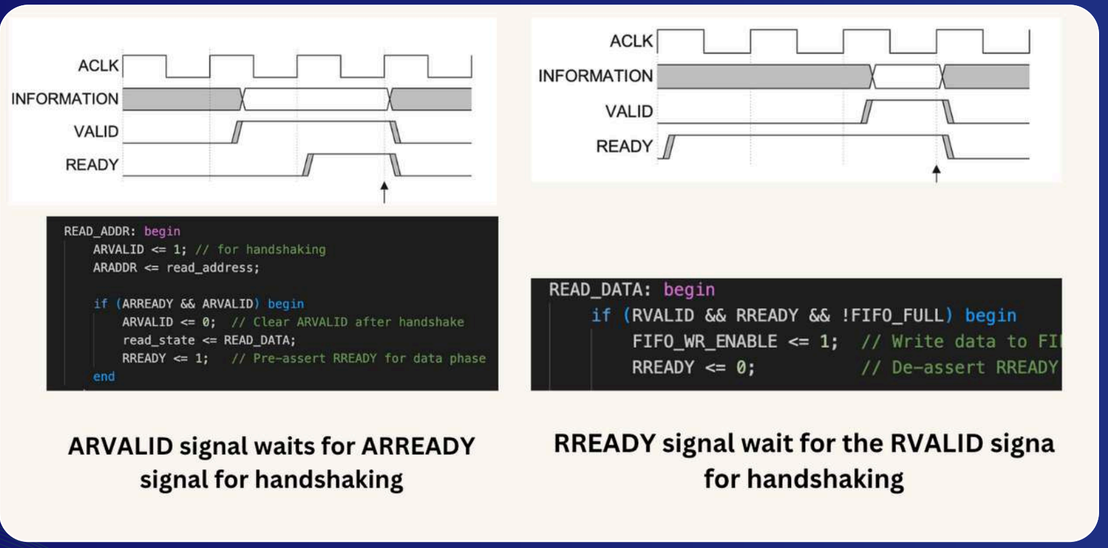
FIFO_EMPTY & FIFO_FULL
Flags indicating FIFO status to prevent overflow or underflow.

04

FIFO_WR_PTR & FIFO_RD_PTR
Pointers for write and read operations to manage FIFO entries.

AXI-Lite Handshaking Methodology

This handshaking ensures that data transfers occur correctly without collisions or data loss



ARVALID signal waits for ARREADY signal for handshaking

RREADY signal wait for the RVALID signa for handshaking

01

ARADDR, ARVALID, ARREADY
The module asserts ARVALID with the read address

02

RDATA, RVALID, RREADY
The slave asserts RVALID when data is ready.

03

AWADDR, AWVALID, AWREADY
The module asserts AWVALID with the write address

04

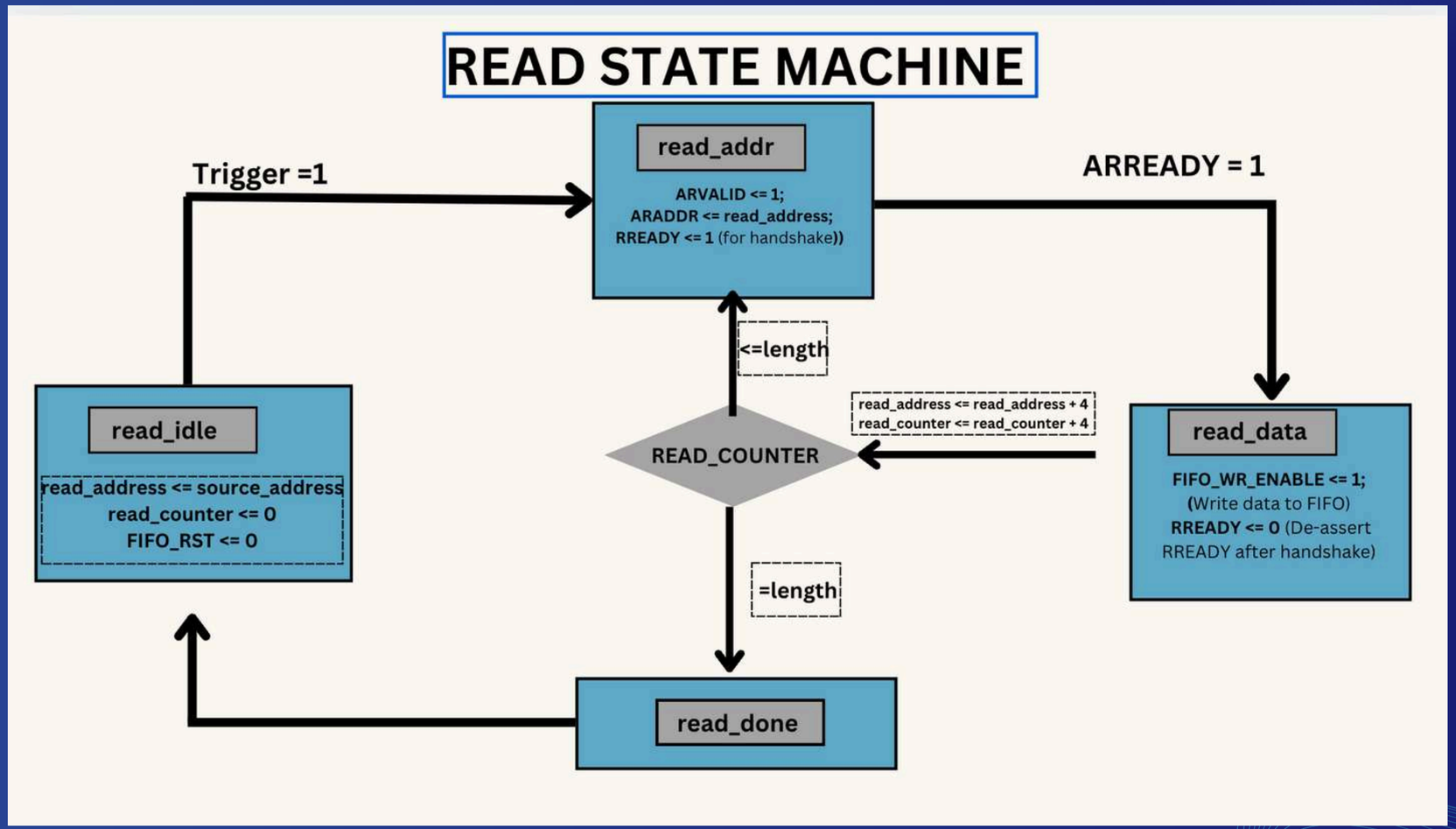
WDATA, WVALID, WREADY
The module asserts WVALID with data (WDATA) read from the FIFO

05

BVALID, BREADY, BRESP
The slave asserts BVALID with a response.

Read State Machine

The read state machine controls fetching data from the source address via AXI.



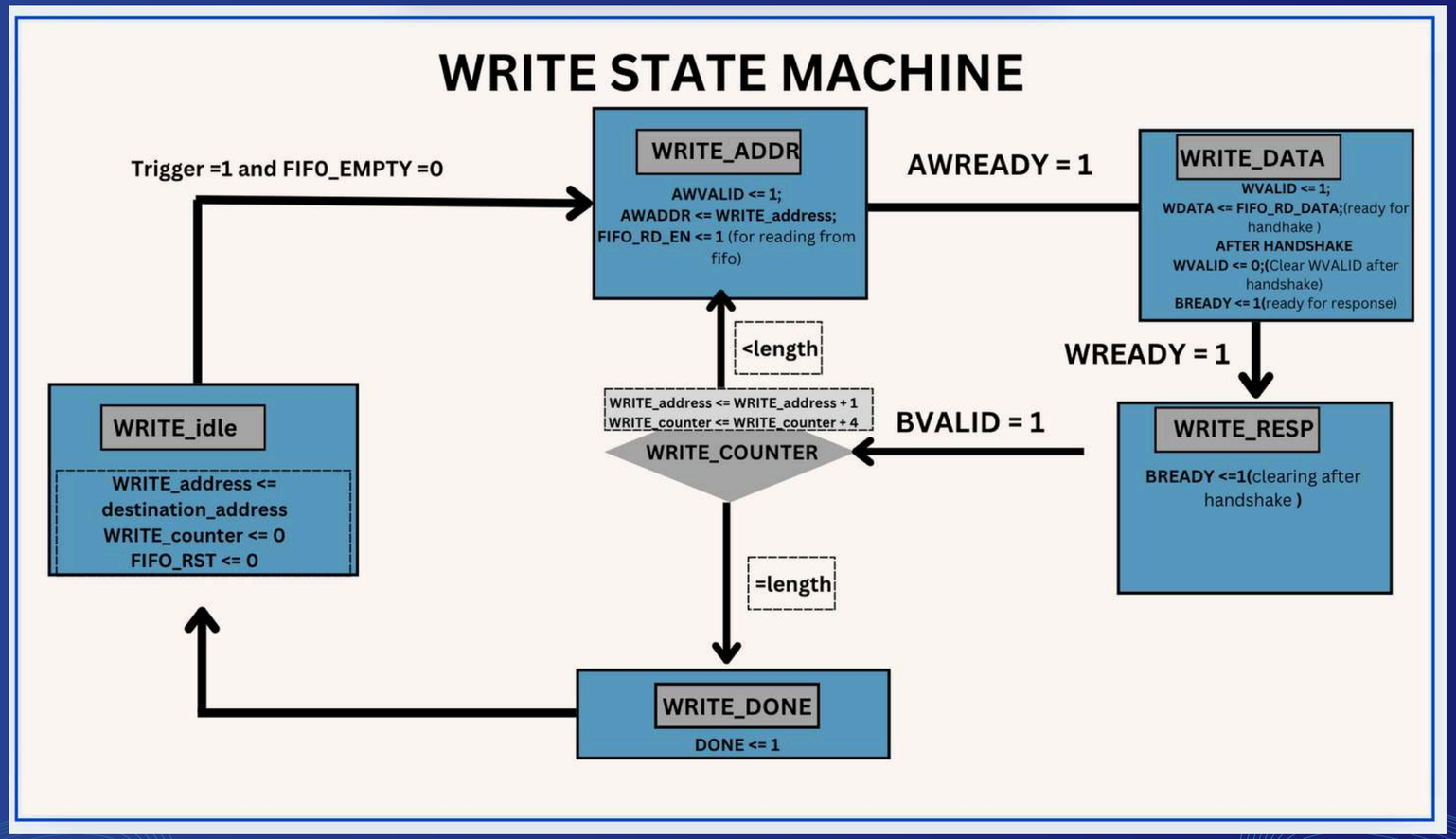
Read State Machine

The read state machine controls fetching data from the source address via AXI.

State	Description
READ_IDLE	Waits for the trigger signal to start the transaction.
READ_ADDR	Sends the read address to the AXI bus and waits for address acceptance.
READ_DATA	Waits for the data to be available from the AXI bus and writes it to the FIFO.
READ_DONE	Completes the read operation and transitions to READ_IDLE.

Write State Machine

The write state machine manages storing data into the destination address via AXI.

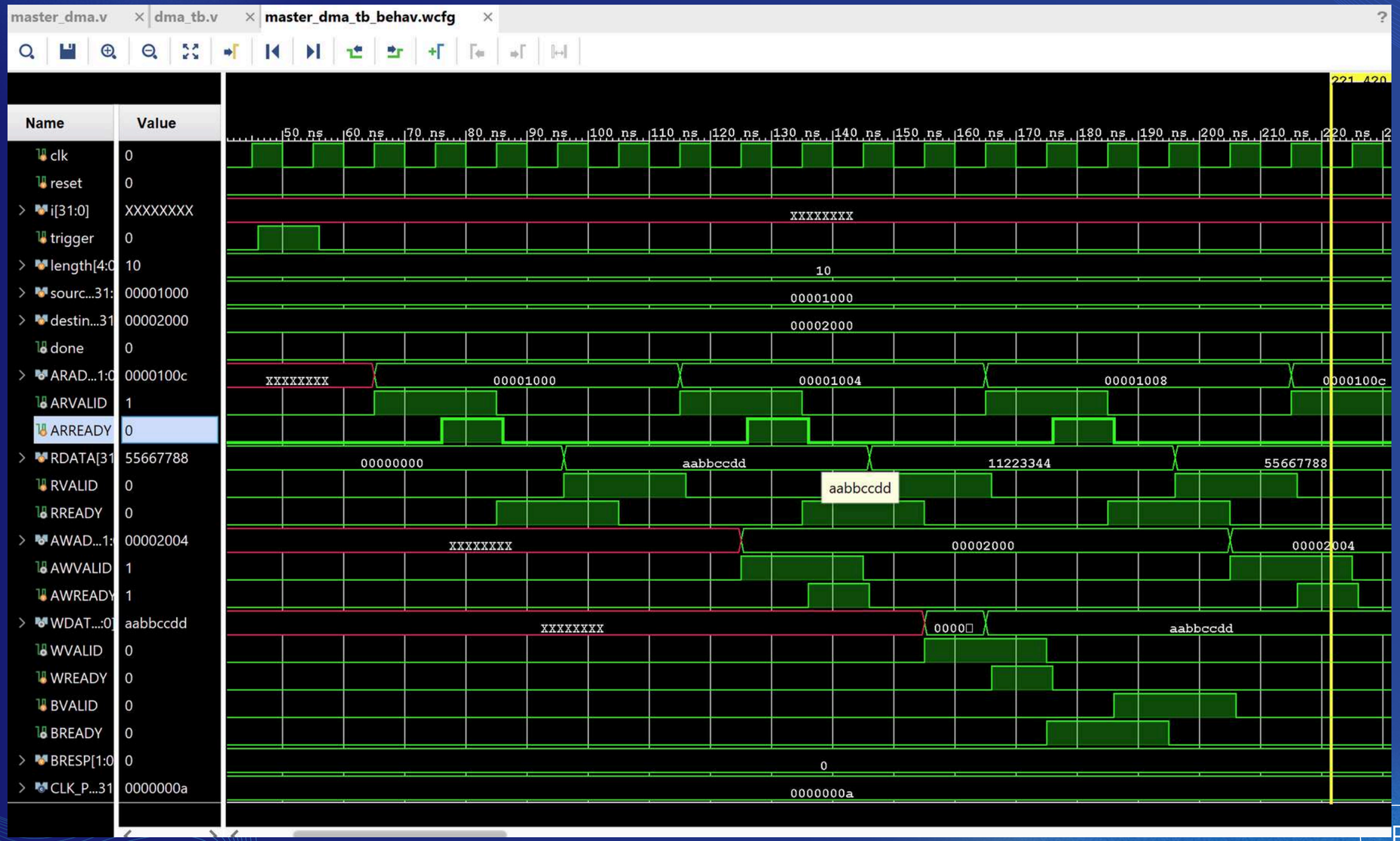


Write State Machine

The write state machine manages storing data into the destination address via AXI.

State	Description
WRITE_IDLE	Waits for data in the FIFO and a trigger signal to start the transaction.
WRITE_ADDR	Sends the write address to the AXI bus and waits for address acceptance.
WRITE_DATA	Reads data from the FIFO and writes it to the AXI bus.
WRITE_RESP	Waits for write response from the AXI bus.
WRITE_DONE	Completes the write operation and transitions to WRITE_IDLE.

SIMULATION FOR ALIGNED TRANSFER



UnAligned DMA Transfer- Core Features

Feature	Description
Unaligned Reads	Reads from addresses like 0x1002 by merging words (e.g., 0x3344AABB).
Unaligned Writes	Writes to addresses like 0x2003 using WSTRB masking (e.g., 4'b0001).
AXI-Lite Compliance	Implements handshaking (ARVALID/ARREADY, WVALID/WREADY).
FIFO Buffering	Decouples read/write operations for parallel transfers.

Design Breakdown

Helper Functions

- **align_to_word(addr)**: Strips lower 2 bits of addr to align to 4-byte boundaries.
  - Example: 0x1003 → 0x1000
- **get_offset(addr)**: Returns addr[1:0] to compute byte offset (0–3).
- **packed_word()**: Merges partial bytes into a 32-bit word based on offset and remaining bytes.

STANDARD FIFO

- **16-entry buffer** to decouple read/write operations , so that reading and writing canhappen at different instants.
- **Synchronous read/write** with full/empty flags.

Design Breakdown

FOR UNALIGNED

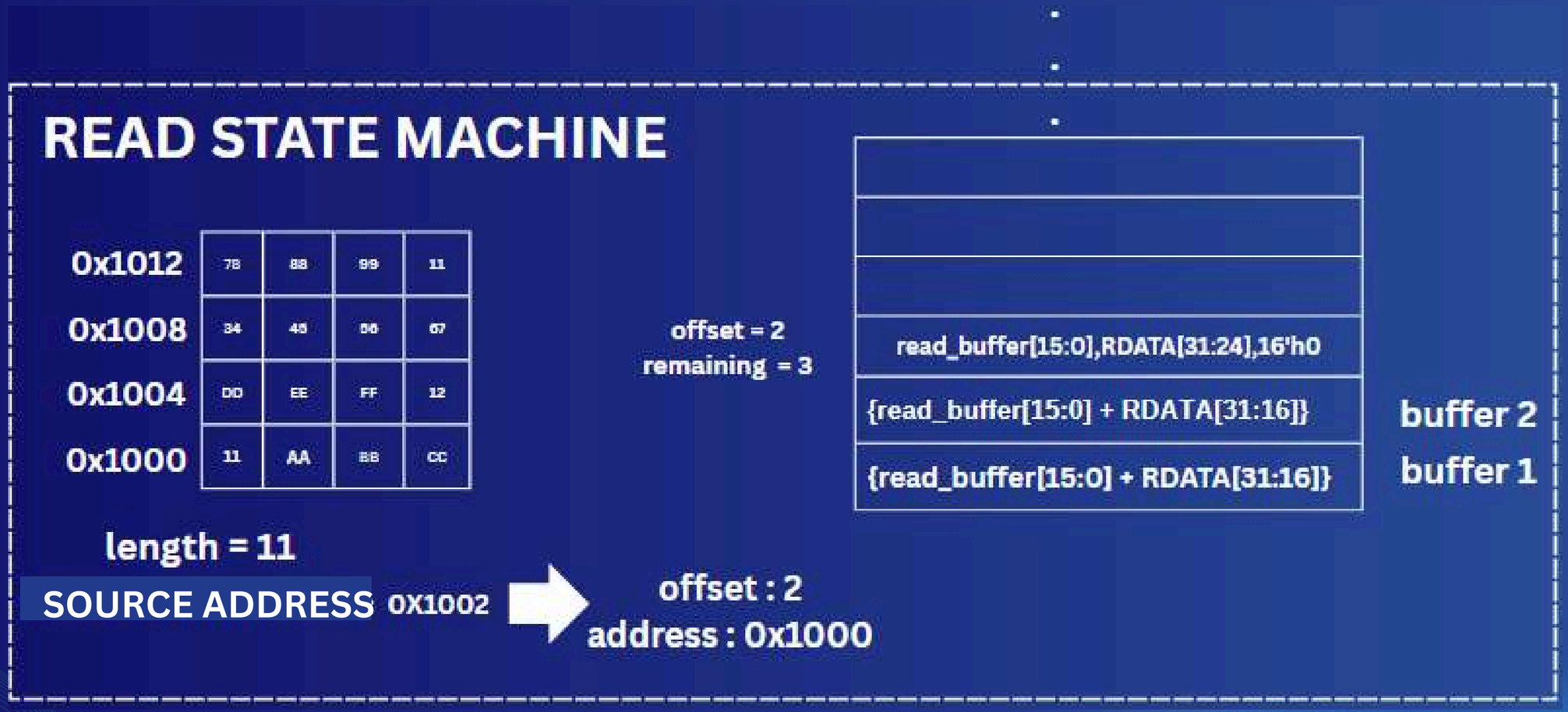
Read FSM

Logic:

- Reads aligned words (e.g., 0x1000 for src_addr=0x1002).
- Combines partial data using src_offset (see table below).
- Pushes aligned 32-bit words to FIFO.

src_offset	Data Merging Example
2'b01	{prev_word[23:0], new_word[31:24]}
2'b10	{prev_word[15:0], new_word[31:16]}

READ_PACK

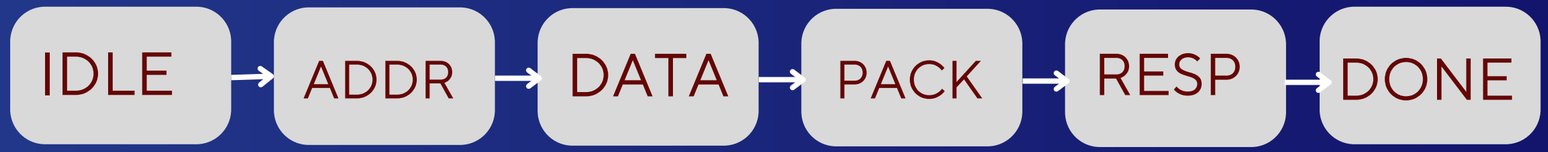

READ STATE MACHINE
0x1012
0x1008
0x1004
0x1000
78 88 99 11
34 45 56 67
DD EE FF 12
11 AA BB CC
length = 11
SOURCE ADDRESS 0X1002
offset : 2
address : 0x1000
offset = 2
remaining = 3
read_buffer[15:0],RDATA[31:24],16'h0
{read_buffer[15:0] + RDATA[31:16]}
{read_buffer[15:0] + RDATA[31:16]}
buffer 2
buffer 1

STATE MACHINES

WRITE FSM

- **Logic**:
  - Writes to aligned addresses but shifts data using dst_offset.
  - Generates WSTRB dynamically for partial writes.

FOR UNALIGNED

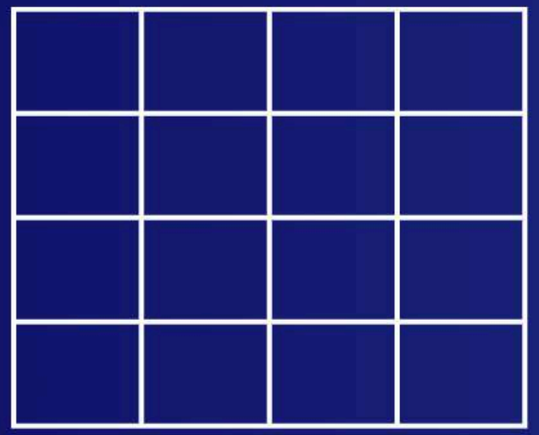


dst_offset	Data Shifting & WSTRB Example
2'b01	WDATA = {8'h0, FIFO_data[31:8]}; WSTRB=4'b0111
2'b11	WDATA = {24'h0, FIFO_data[31:24]}; WSTRB=4'b0001

UNALIGNED WRITE STATE MACHINE

LAST WORD

REM (REMANING) = **TOTAL - WRITTEN**



INTERMEDIATE WORDS

FIFO

DST OFFSET

01: BUFFER <= FIFO_RD_DATA[7:0]

10: BUFFER <= FIFO_RD_DATA[15:0]

11: BUFFER <= FIFO_RD_DATA[23:0]

FIRST WORD

REM	WDATA	WSTRB
1	[BUFFER[OFFSET] , 24'h0]	1000
2	[BUFFER[OFFSET] ,FIFO_RD_DATA[REM-Offset], 16'h0]	1100
3	[BUFFER[OFFSET] ,FIFO_RD_DATA[REM-Offset], 24'h0]	1110

OFFSET	WDATA	WSTRB
0	**FIFO_RD_DATA**	**1111**
1	**{write_buffer[7:0], FIFO_RD_DATA[31:8]}**	**1111**
3	**{write_buffer[15:0], FIFO_RD_DATA[31:16]}**	**1111**
3	**{write_buffer[23:0], FIFO_RD_DATA[31:24]}**	**1111**

OFFSET	WDATA	WSTRB
1	**{8'h0, FIFO_RD_DATA[31:8]}**	0111
2	**{16'h0, FIFO_RD_DATA[31:16]}**	0011
3	**{8'h0, FIFO_RD_DATA[31:24]}**	0001

Thank You!