Memory Controller for DDR Memories

Project Guide: Prof. Mahesh A

Pranav M – PES2UG23EC076
Lalith – PES2UG23EC074
Harshini – PES2UG23EC058

Introduction and Problem Statement

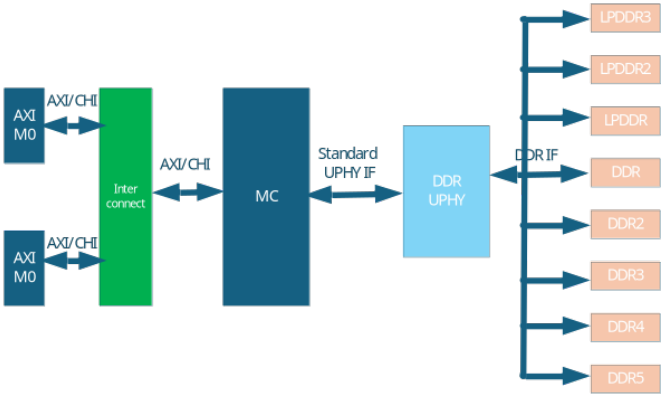
What is a Memory Controller?

- Bridge between processor/SoC and DRAM
- Manages read/write transactions, timing, refresh
- Critical for system performance and power

Our Goal:

Design and implement an RTL-level memory controller targeting DDR memories.

System Block Diagram



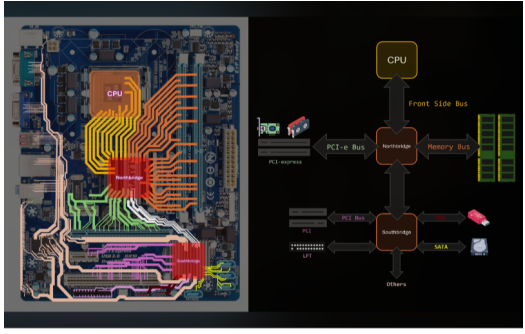
Evolution of Memory Controller Placement

Northbridge-based Controller

- Controller in external chipset
- Higher latency due to front-side bus

Integrated Memory Controller

- Controller moves onto CPU/SoC die
- Simpler DRAM design, lower cost
- Advanced scheduling, prefetching, power management



DRAM Controller Functions

Correct Operation

- Ensure refresh and timing constraints
- Translate requests to DRAM commands

Performance

- Schedule requests for high throughput
- Reordering and row-buffer management

Power Management

- Manage DRAM power modes and thermals

Why Complex Controllers?

- Reduces DRAM die area and DIMM cost
- Centralizes protocol changes
- Improves scalability across generations
- Better performance-per-watt

Challenge – Many DRAM Timing Constraints

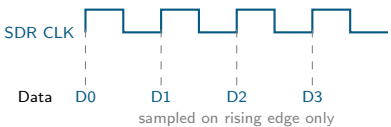
DRAM has 50+ timing constraints that must be obeyed. Key examples:

Constraint	Description
tWTR	Min cycles before READ after WRITE
tRC	Min cycles between consecutive ACTIVATEs to same bank
tRRD	Min cycles between ACTIVATEs to different banks
tFAW	Max 4 ACTIVATEs in a rolling time window

Controller must track:

- Channels, banks, ranks
- Data bus, address bus
- Row buffers
- Refresh intervals
- Power states

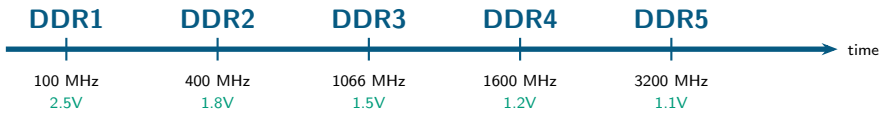
SDR vs DDR – What Changed?



Key Differences

	SDR	DDR
Transfer	Rising edge	Both edges
Bandwidth	1×	2×
Voltage	3.3V	2.5V / lower
Prefetch	1n	2n

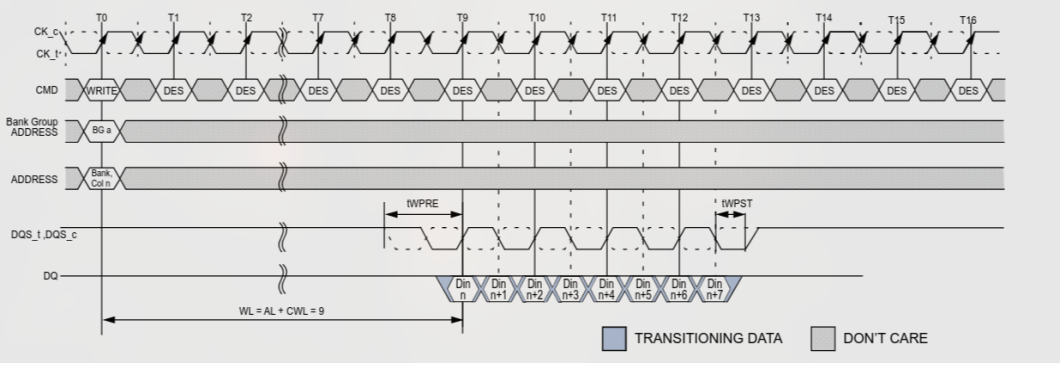
DDR Generations – Evolution



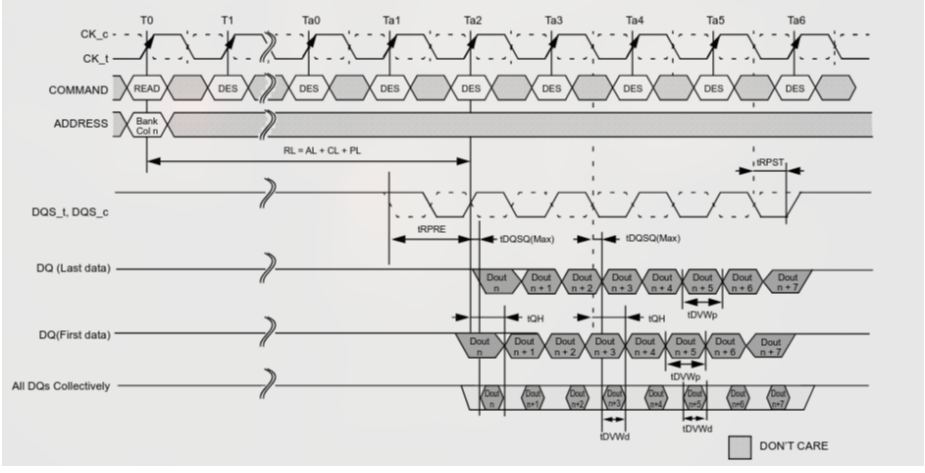
Each generation brought:

- Higher clock frequencies and bandwidth
- Lower operating voltage (power savings)
- Improved prefetch depth (2n $\rightarrow$ 4n $\rightarrow$ 8n $\rightarrow$ 16n)
- Key challenges as speeds increase: signal integrity, timing closure, power delivery, and clock domain crossing

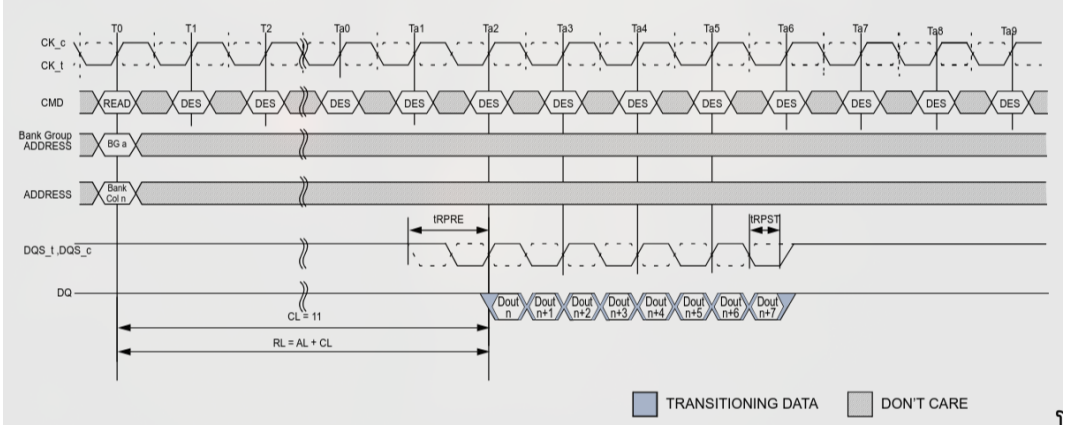
DDR Timing – Write



DDR Timing – Read



DDR Timing – Read Burst



LPDDR – Low Power DDR

Requirements

- High bandwidth
- Low latency
- Power efficient

Advanced Power Management

- Partial Array Self Refresh (PASR)
- Temperature Compensated Self Refresh (TCSR)
- Deep Power-Down (DPD)

LPDDR vs DDR

	DDR5	LPDDR5
Voltage	1.1V	1.05V
Form factor	DIMM	PoP/BGA
Power modes	Basic	PASR/TCSR/DPD

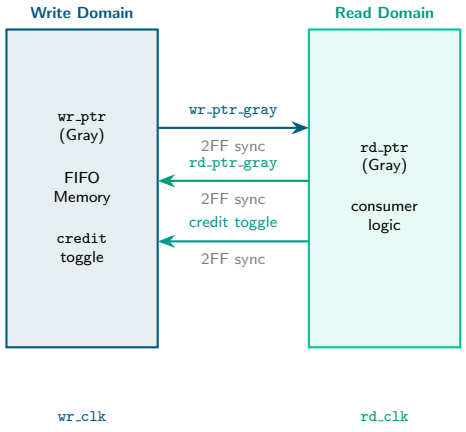
Asynchronous FIFO – Functionality

Clock Domain Crossing (CDC)

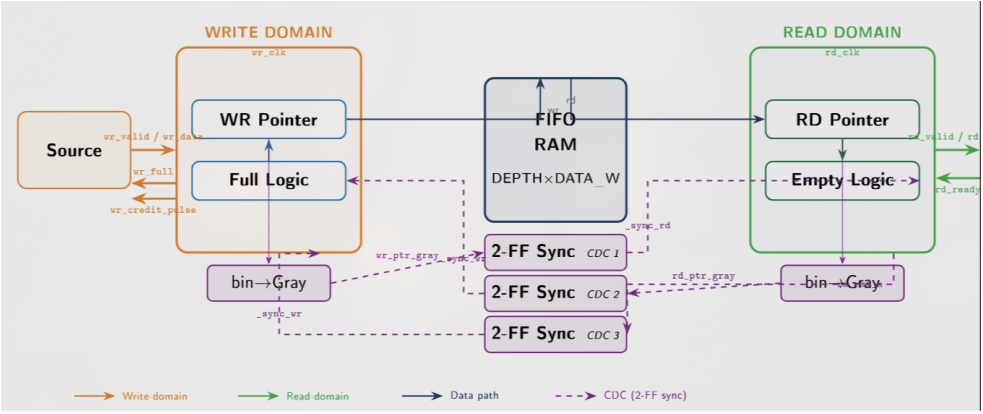
- AXI clock $\neq$ PHY/memory clock
- Direct signal passing causes metastability
- Async FIFO safely bridges the two domains

Our Implementation

- Parameterizable depth and width
- Gray-coded pointers for safe CDC
- 2-FF synchronizer (`cdc_sync`) module
- Credit-based flow control



Asynchronous FIFO – Block Diagram



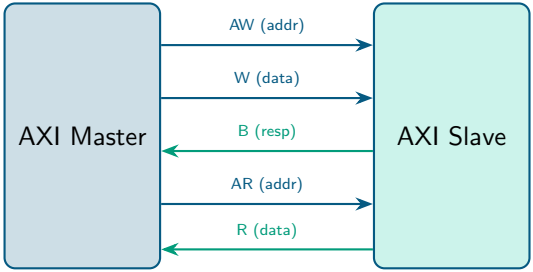
AXI Interface – Overview

Why AXI?

- Industry standard (ARM AMBA)
- Separate read and write channels
- Burst transfers, out-of-order support
- Decouples master from slave

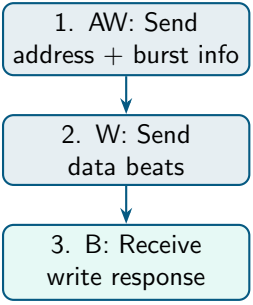
5 Channels:

- AW – Write Address
- W – Write Data
- B – Write Response
- AR – Read Address
- R – Read Data

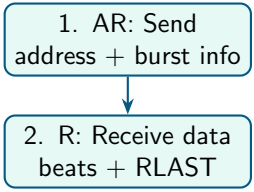


AXI – Write and Read Transaction Flow

Write Transaction

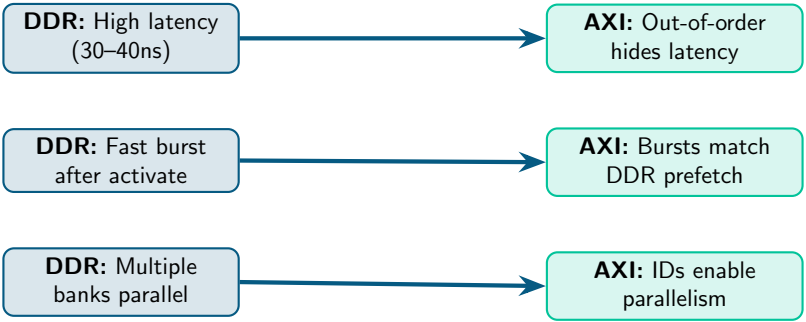


Read Transaction



Handshake: Every channel uses `VALID/READY` handshake – transfer occurs only when both are high.

Why AXI is Perfect for DDR Memory Controllers

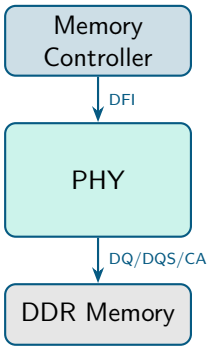


Perfect match between protocol and memory!

PHY Layer – Physical Interface

Role of PHY

- Bridge between the memory controller and the memory
- Temporarily stores instructions and data in buffers
- Performs Write Leveling, DQS Gate Training and Vref Training



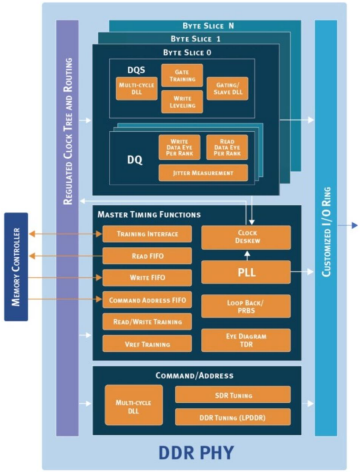
DFI Interface

What is DFI?

- Universal language for PHY interoperability
- Decouples memory controller from PHY implementation

Three Pillars of DFI 5.0/5.1

- PHY independent boot sequence
- Expanding frequency range support
- New PHY-to-Controller Interface interaction



Next Steps


Async FIFO RTL
Done ✓
Looking into
DDR Memory
Done ✓
Finalize
Microarchitecture
1–2 weeks