地平线
Horizon Robotics

# An introduction to SDRAM and memory controller

芯片平台-ASIC

邢志国

2019-01

# 目录 CONTENTS

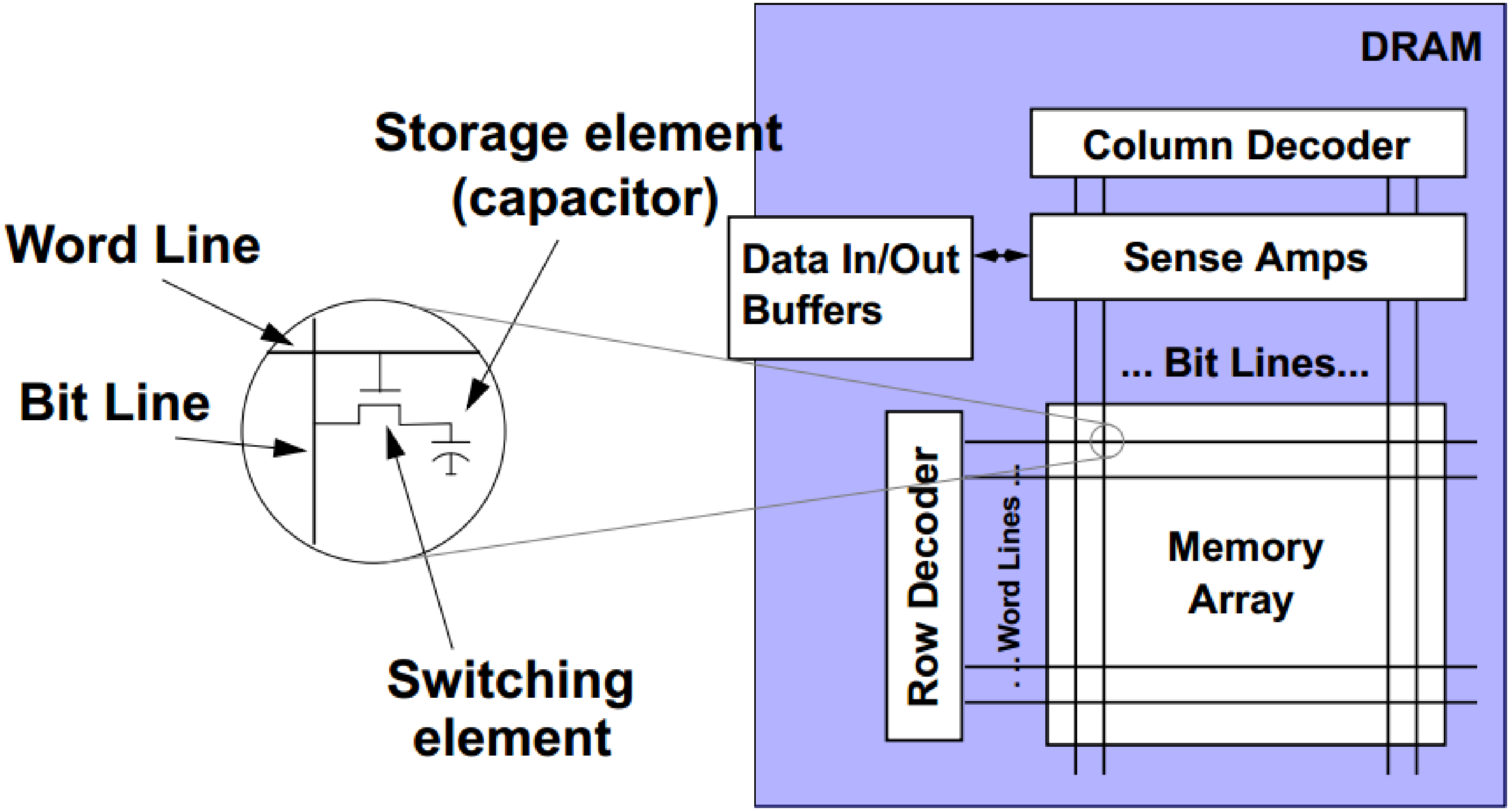
1. DDR SDRAM Brief Introduction
2. DDR SDRAM Efficiency
3. LPDDR3 VS LPDDR4
4. DDR Memory Controller

# Part 1

## DDR SDRAM Brief Introduction

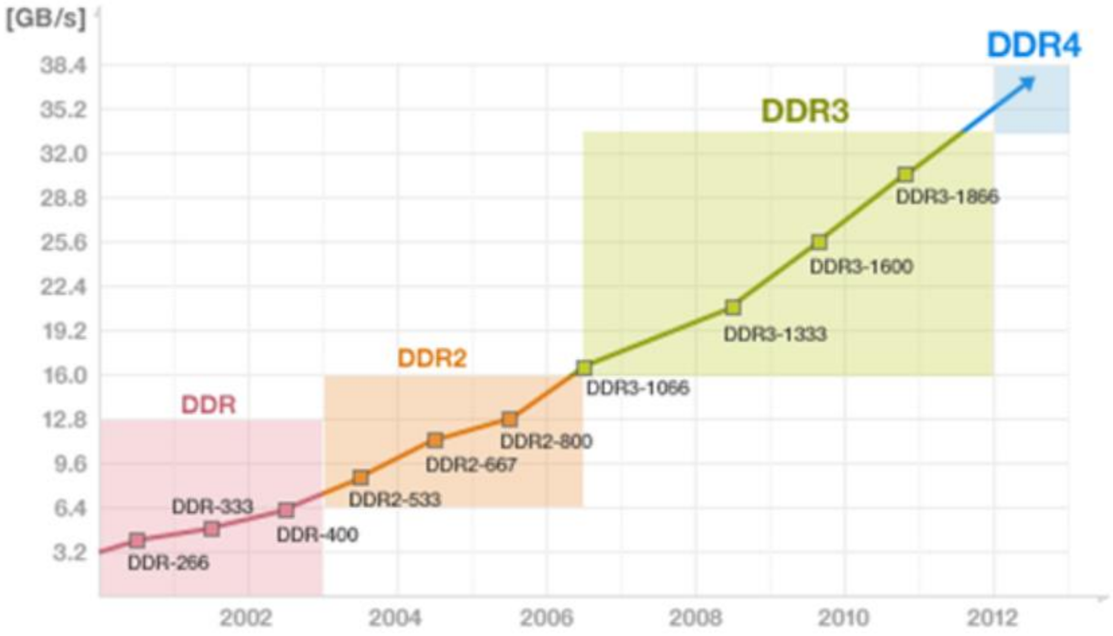
## Dynamic RAM (DRAM)

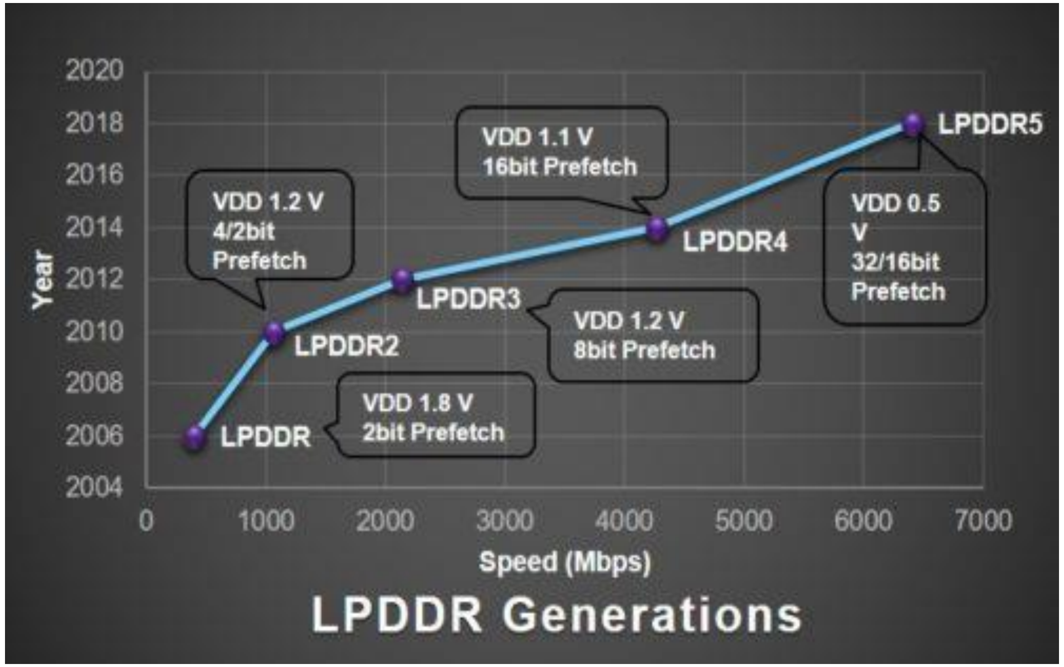
DRAM was patented in 1968 by **Robert Dennard** at IBM



# Evolution of the DRAM Design

**SDRAM (single Data rate ):** A clock signal was added making the design synchronous
**DDR (Double Data Rate) :** The data bus transfers data on both rising and falling edge of the clock

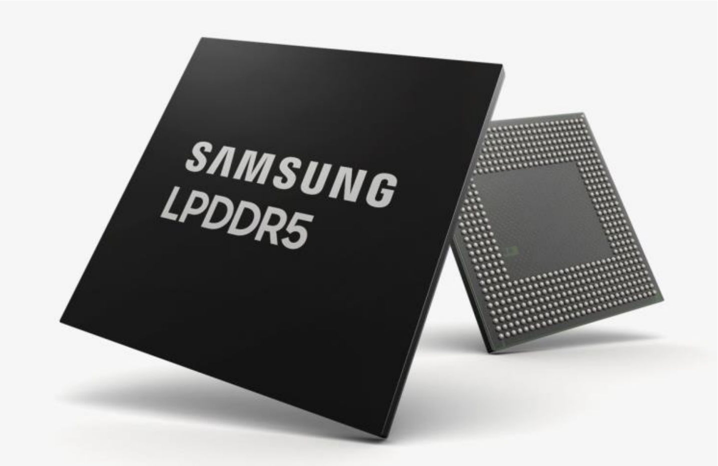




# DDR SDRAM

## DDR5:

- DDR4 successor JEDEC standard for Server/Enterprise markets
- Targeting data rates up to 6400Mbps
- DDR5 samples projected for H1/2019



## LPDDR5:

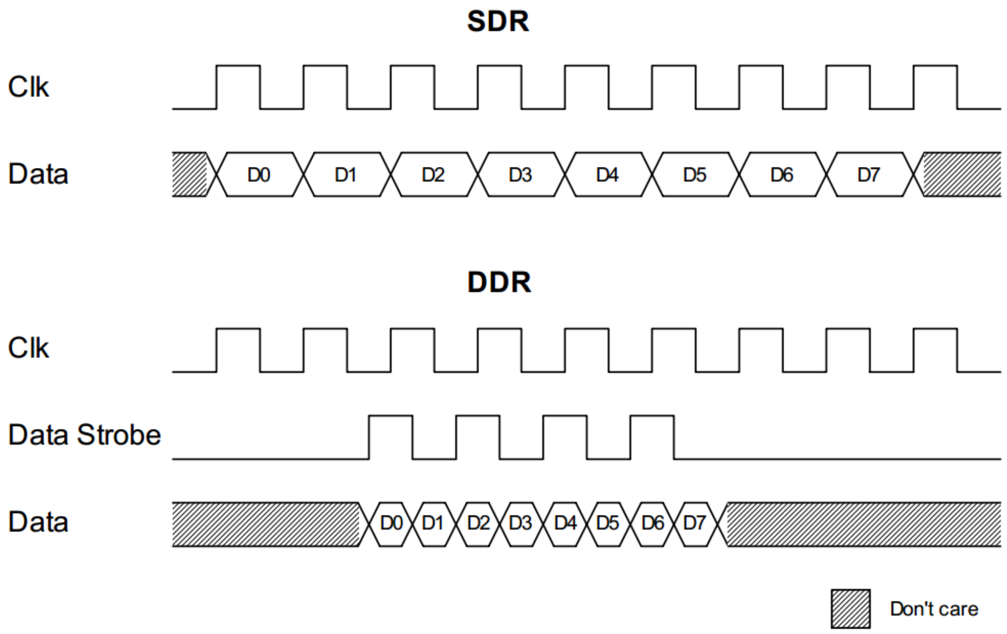
- LPDDR4/4X successor JEDEC standard for Mobile/ Automotive markets
- Targeting data rates up to 6400Mbps
- LPDDR5 samples projected for mid 2019

# DDR SDRAM

地平线 Horizon Robotics

## DDR(Double Data Rate)

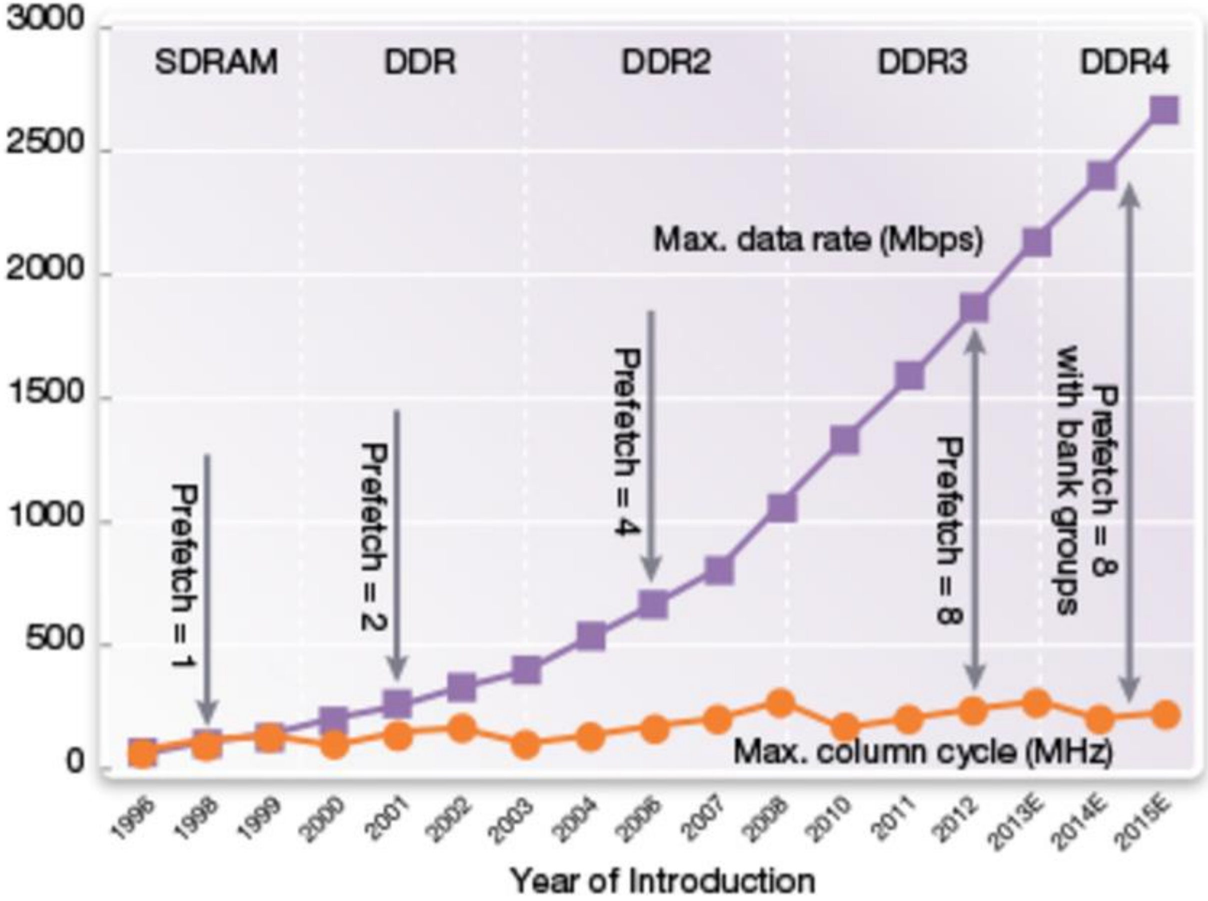
- Double-data-rate architecture; two data transfers per clock cycle
- **Differential clock inputs**
- Bidirectional, data strobe (DQS) is transmitted/received with data, to be used in capturing data at the receiver
- DQS is **edge-aligned** with data **for READs; center- aligned** with data **for WRITEs**
- Read and write accesses to the DDR SDRAM are **burst oriented.**



# DDR SDRAM

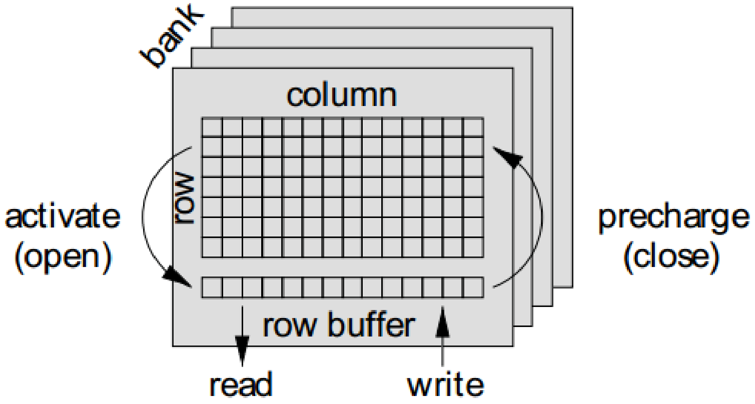
**Prefetch :**

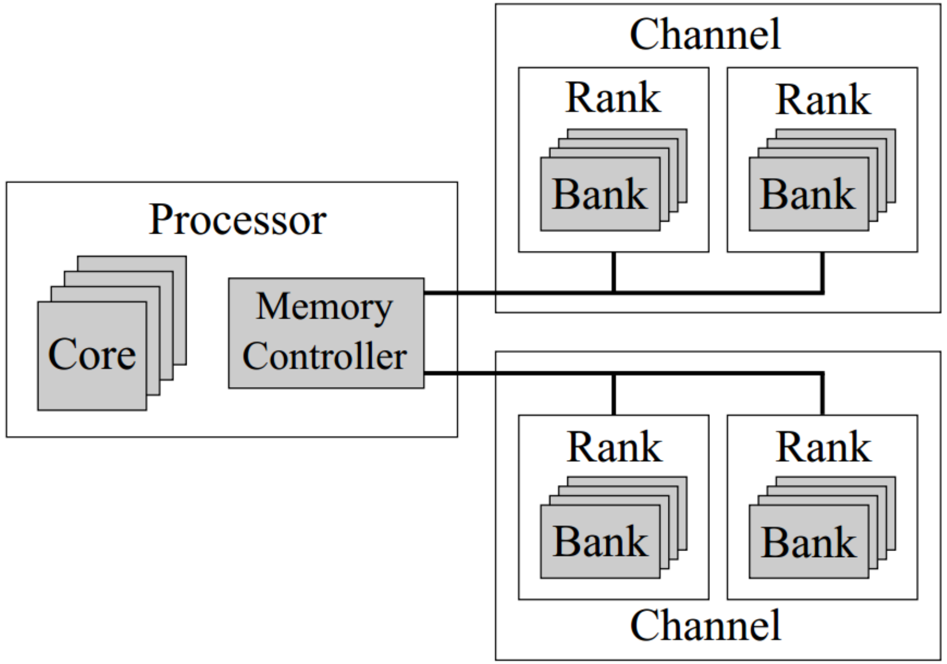
Internal data bus width = Prefetch value * external bus width
DDR : 2n
DDR2/LPDDR2： 4n
DDR3/LPDDR3: 8n
DDR4: 8n
LPDDR4/LPDDR4x: 16n



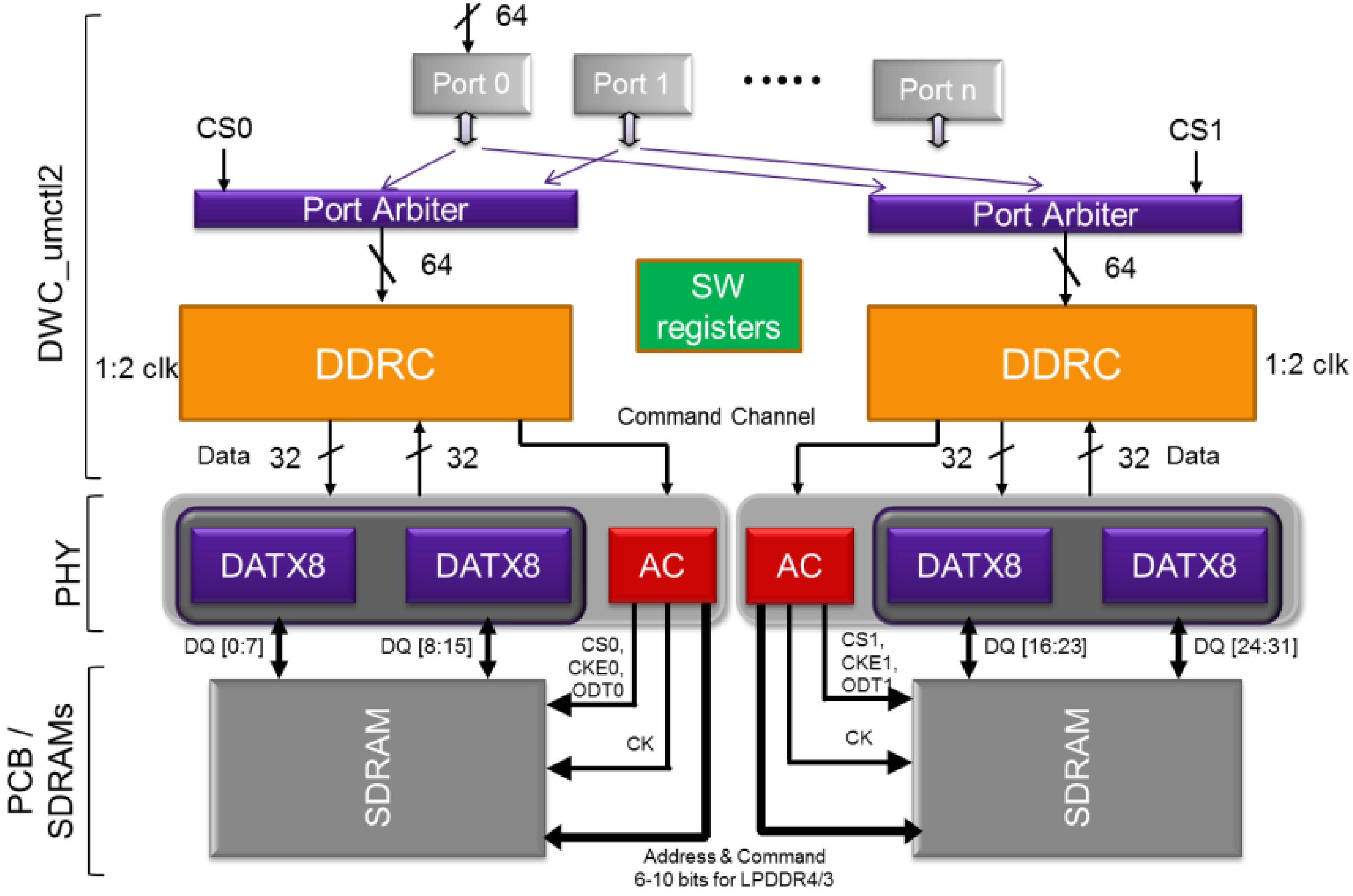
# SDRAM Architecture Basic Conception

**Row (page)**
**Column**
**Bank :** a bank consists of multiple rows and columns
**Rank:** a set of DRAM chips connected to the same chip select
**Channel：** Bus --> Controller-->PHY-->SDRAM
**Burst Length (BL):** determined by prefetch architecture





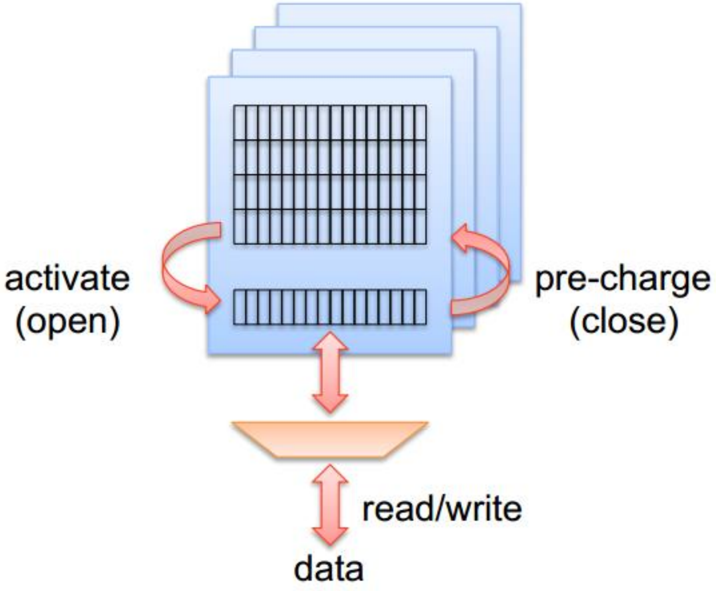
## Example： Dual-Channel



**Channel 0** **Channel 1**

## SDRAM Command Summary

**Write Command (WR) :** Initiate a write burst to an active row
**Read Command (RD) :** Initiate a read burst to an active row
**Mode Register Read (MRR):**
**Mode Register Write (MRW):**
**Activate (ACT) :** Activate a row in a particular bank
**Precharge (PRE):** Close a row in a particular bank independently or rows of all banks simultaneously
**Refresh (REF) :** Start a refresh operation
**NOP/DES :** No Operation / Ignores all inputs



# Refresh Operation

### Refresh :

Refresh is the periodic command required by DRAM to maintain the charge on memory cells in the DRAM device.

The exact refresh interval time depends on the DRAM type (e.g., DDR or LPDDR) and the operating temperature.

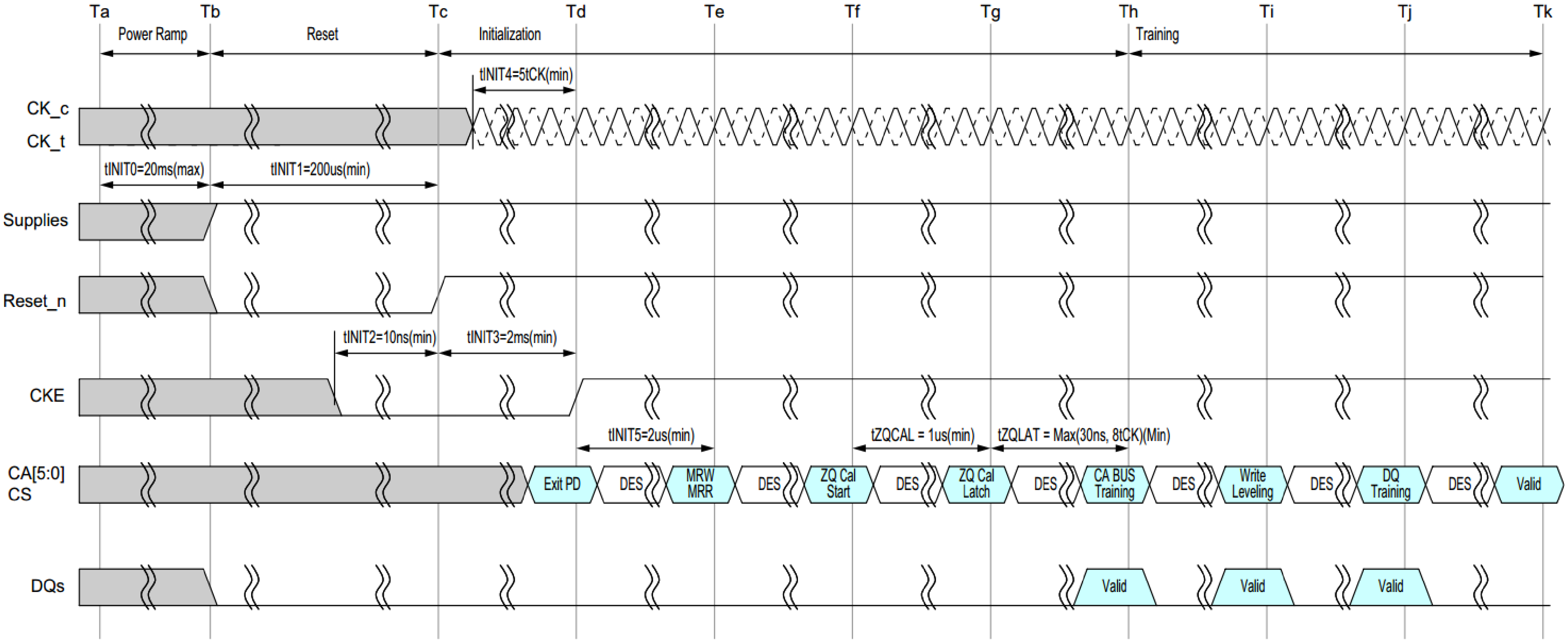
### Auto-refresh:

Auto refresh is generally a term associated with automatic generation of refresh commands with the memory controller;

### Self-refresh:

The DRAM supports a mode where refresh is maintained internally by the memory device. Self refresh is a low power mode and generally entered when memory is IDLE. No read or write transactions can occur when memory is in self refresh.

## DDR SDRAM initialization (Example： LPDDR4）



# DDR SDRAM

## Memory Training

**Command /Address Training (CA Training) :** Centers the DRAM clock in the center of the DDR CA data eye. LPDDR3/LPDDR4 Only

**Write leveling :** align DQS rising edge with the MEMCLK rising edge

**Read Gate Training (aka RxEn training) :** determine the round-trip latency

**Read DQ deskew training :** compensates for delay differences among the DQ lanes, primarily caused by board routing and DRAM DQ output skew

**Data Eye Training (Read Leveling):** center the read DQS timing in the read DQ passing region.

**VREF Training :**Defines the reference voltage range. VREF training is only applicable for DDR4/LPDDR4 memory systems

地平线
Horizon Robotics

# Part 2

## DDR SDRAM Efficiency

## Memory efficiency

Memory efficiency is the fraction between the amount of clock cycles when data is transferred and the total amount of clock cycles

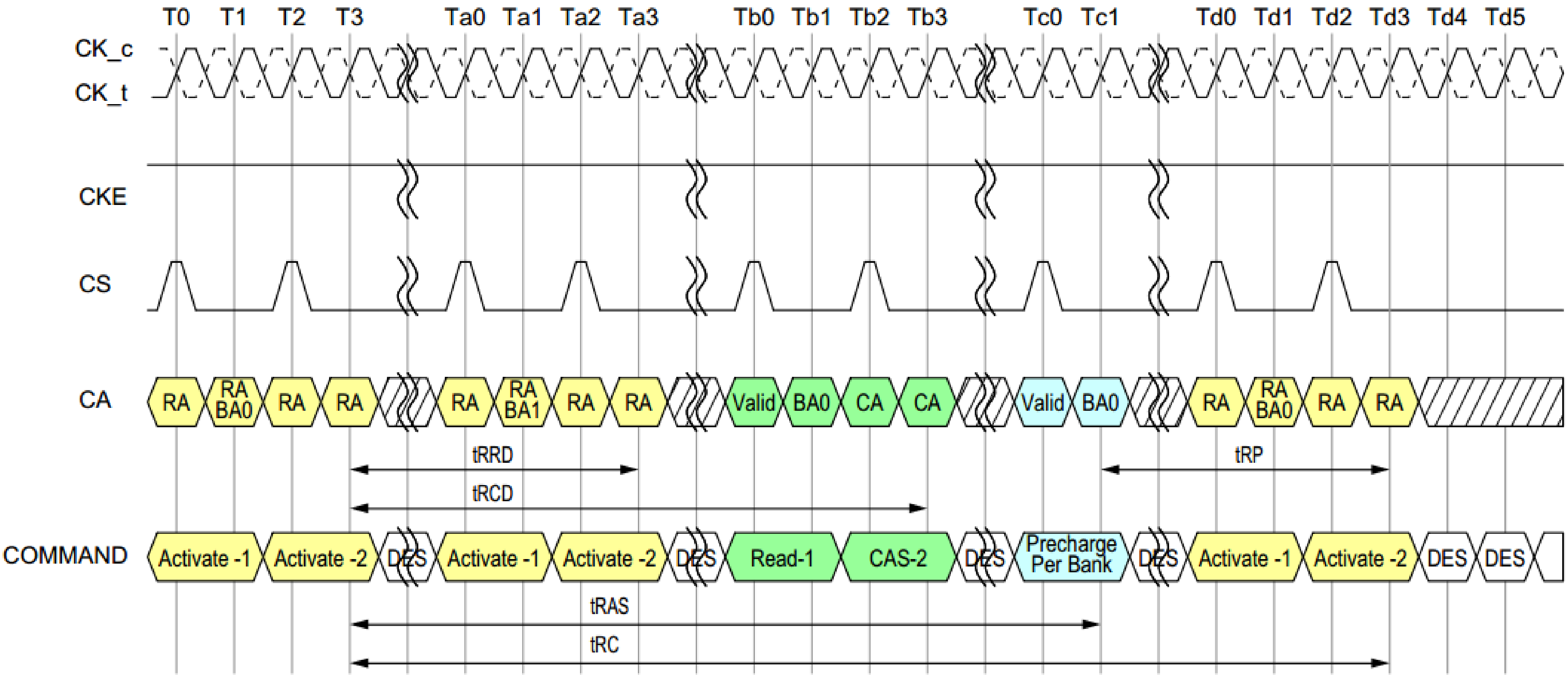
- Defines the exchange rate between peak bandwidth and actual bandwidth
- peak bandwidth = memory clock freq * 2 * (memory data bus width/8)

Four categories of memory efficiency for SDRAM:

- Bank conflict efficiency
- Refresh efficiency
- Data efficiency
- Read/write efficiency

## Bank conflict efficiency

Bank conflicts : two consecutive bursts to different rows in the same bank.



## Timing Constraints (LPDDR4 3200Mbps)

| Parameter | Definition | Cycles(tCK) |
|---|---|---|
| tRRD | Active bank-A to active bank-B | 16 |
| tRCD | RAS-to-CAS delay | 29 |
| tRP | Row precharge time | 29(per bank) |
| tRAS | Row active time | 68 |
| tRC | ACTIVATE-to-ACTIVATE command period (same bank) | 97(per bank) |

## Refresh efficiency

### Refresh requirement:

- A bank must be in the idle state before it is refreshed
- No data can be transfered when the memory is being refreshed.

### Degrade performance:

Any accesses to a DRAM bank that is refreshing must wait for the refresh latency *tRFC*,

**All bank refresh :** operates at the rank level, refreshes a number of rows in all banks of a rank concurrently.

**Per bank refresh :** enhanced mode  for LPDDR, This enables a bank to be accessed while another in the same rank is being refreshed,

# DDR SDRAM

## Data efficiency

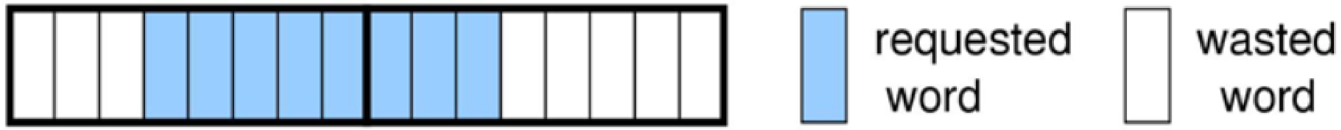
**A memory burst cannot access segments of the minimum burst length**

Minimum access granularity

- Burst length 16， words is 64 B with 32-bit memory

**If data is poorly aligned an extra segment has to be transferred**

Cycles are lost when transferring unrequested data



## Read / write efficiency

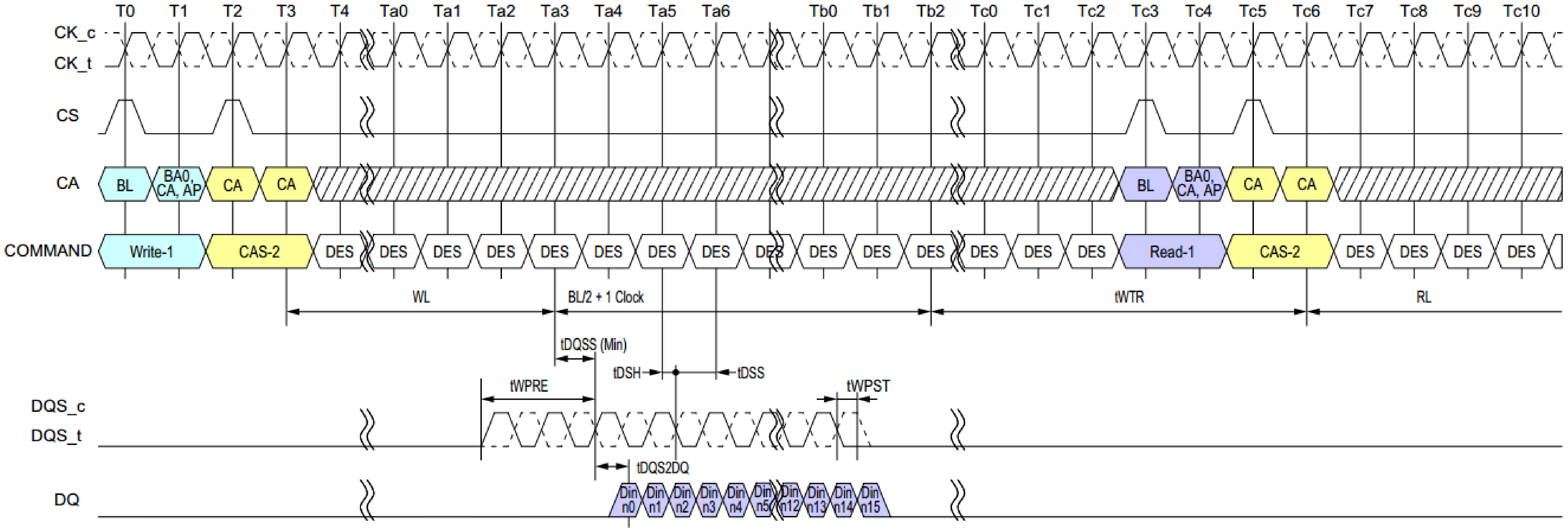
### The data bus of an SDRAM is bi-directional

- ◆ Cycles are lost when switching direction of the data bus
- ◆ Extra NOPs must be inserted between read and write commands

### Read/write efficiency depends on traffic

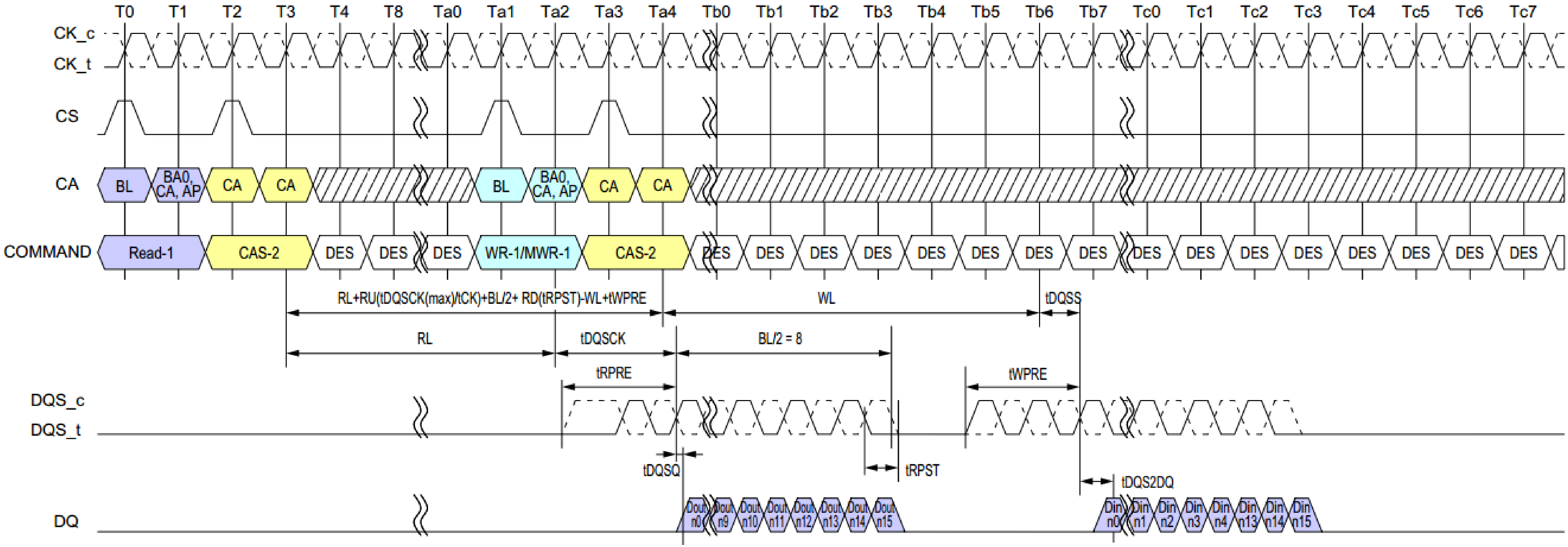
- ◆ Determined by frequency of read/write switches
- ◆ Switching too often has a significant impact on memory efficiency

## Read / write efficiency



**Burst Write Followed by Burst Read**

## Read / write efficiency

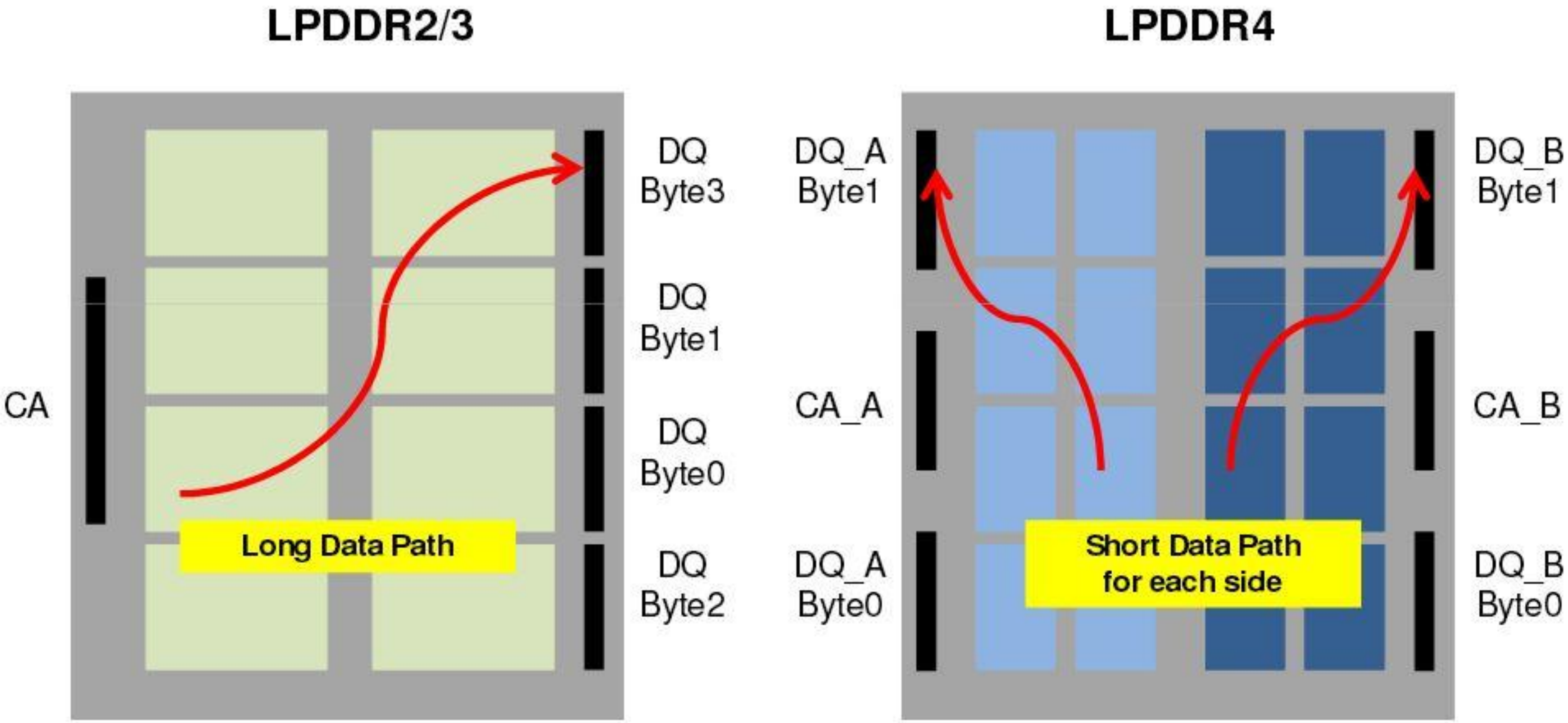


**Burst Read Followed by Burst Write**

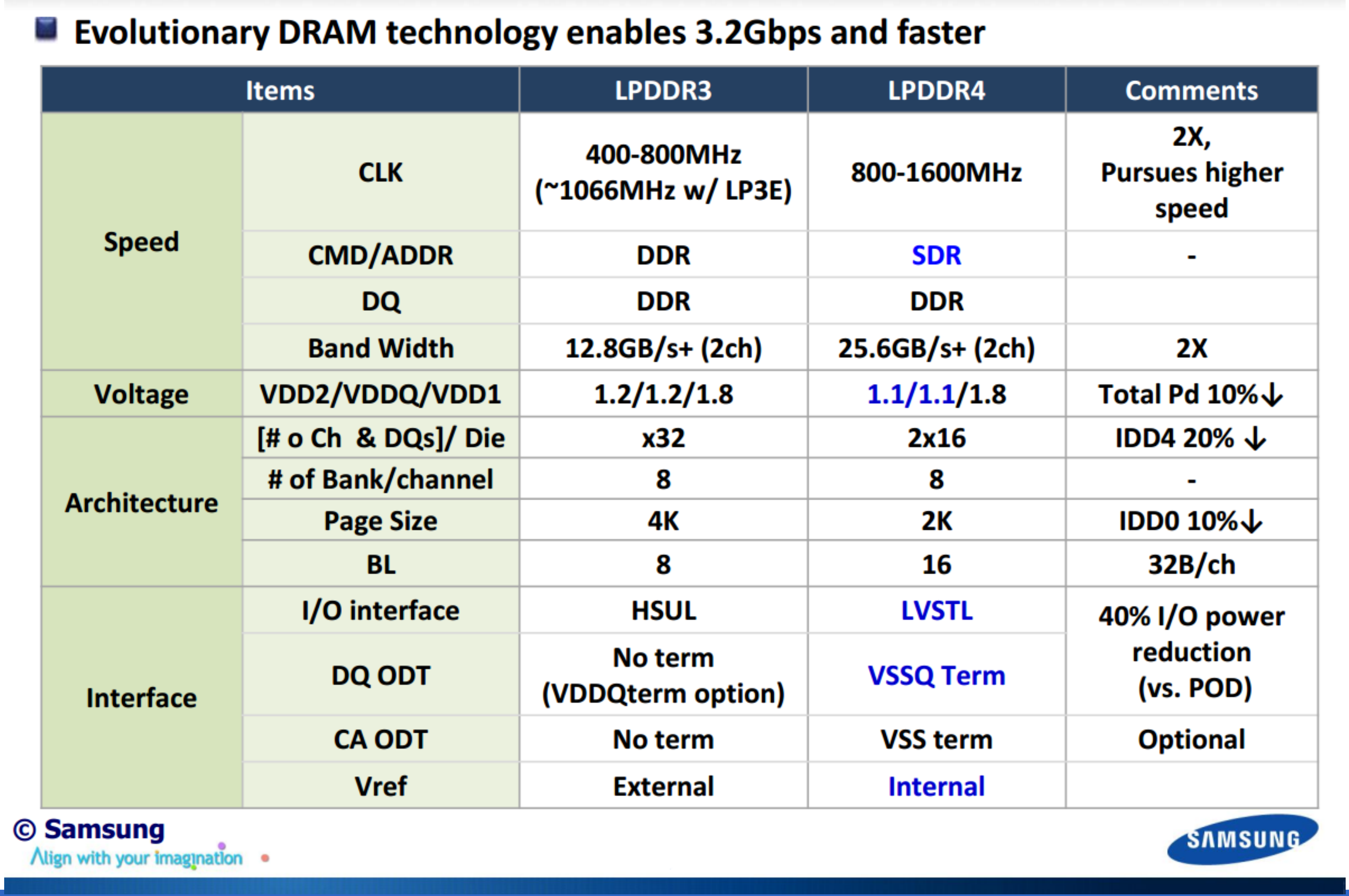
# Part 3

## LPDDR3 VS LPDDR4

## LPDDR3 VS LPDDR4



## LPDDR3 VS LPDDR4



- Evolutionary DRAM technology enables 3.2Gbps and faster

| Items | | LPDDR3 | LPDDR4 | Comments |
|---|---|---|---|---|
| Speed | CLK | 400-800MHz<br>(~1066MHz w/ LP3E) | 800-1600MHz | 2X,<br>Pursues higher speed |
| | CMD/ADDR | DDR | SDR | - |
| | DQ | DDR | DDR | |
| | Band Width | 12.8GB/s+ (2ch) | 25.6GB/s+ (2ch) | 2X |
| Voltage | VDD2/VDDQ/VDD1 | 1.2/1.2/1.8 | 1.1/1.1/1.8 | Total Pd 10%↓ |
| Architecture | [# o Ch & DQs]/ Die | x32 | 2x16 | IDD4 20% ↓ |
| | # of Bank/channel | 8 | 8 | - |
| | Page Size | 4K | 2K | IDD0 10%↓ |
| | BL | 8 | 16 | 32B/ch |
| Interface | I/O interface | HSUL | LVSTL | 40% I/O power reduction<br>(vs. POD) |
| | DQ ODT | No term<br>(VDDQterm option) | VSSQ Term | |
| | CA ODT | No term | VSS term | Optional |
| | Vref | External | Internal | |

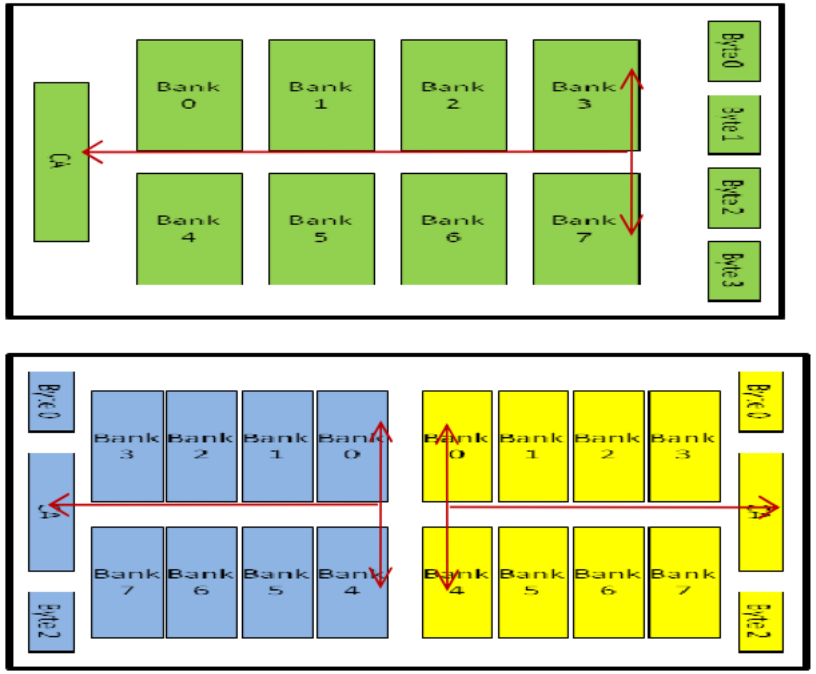
© Samsung
Align with your imagination

SAMSUNG

# DDR SDRAM

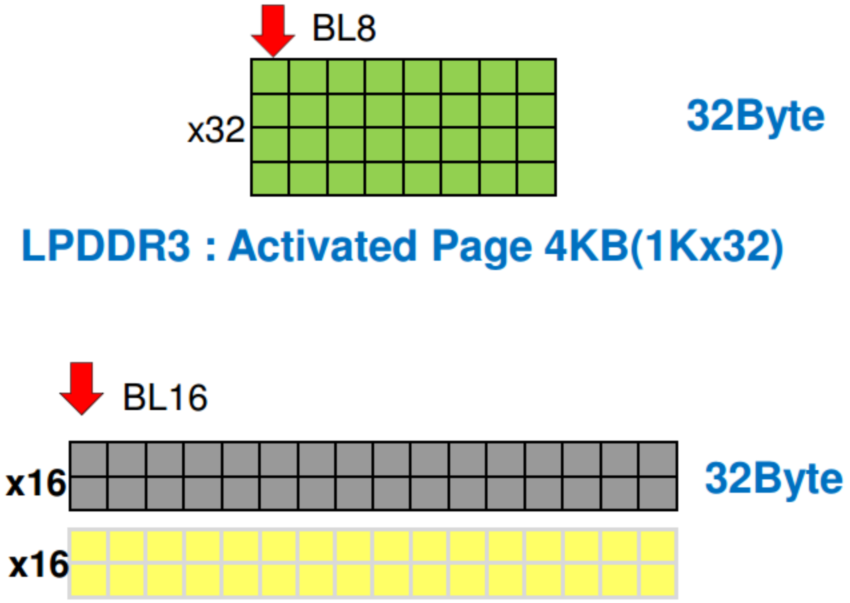
## LPDDR3 VS LPDDR4

### Low Power 32-bit Architecture



~20% IDD4 reduction

### Access Granularity & Page-size/Ch



LPDDR3 : Activated Page 4KB(1Kx32)

LPDDR4 : Activated Page 2KB(1Kx16)

~10% IDD0 reduction

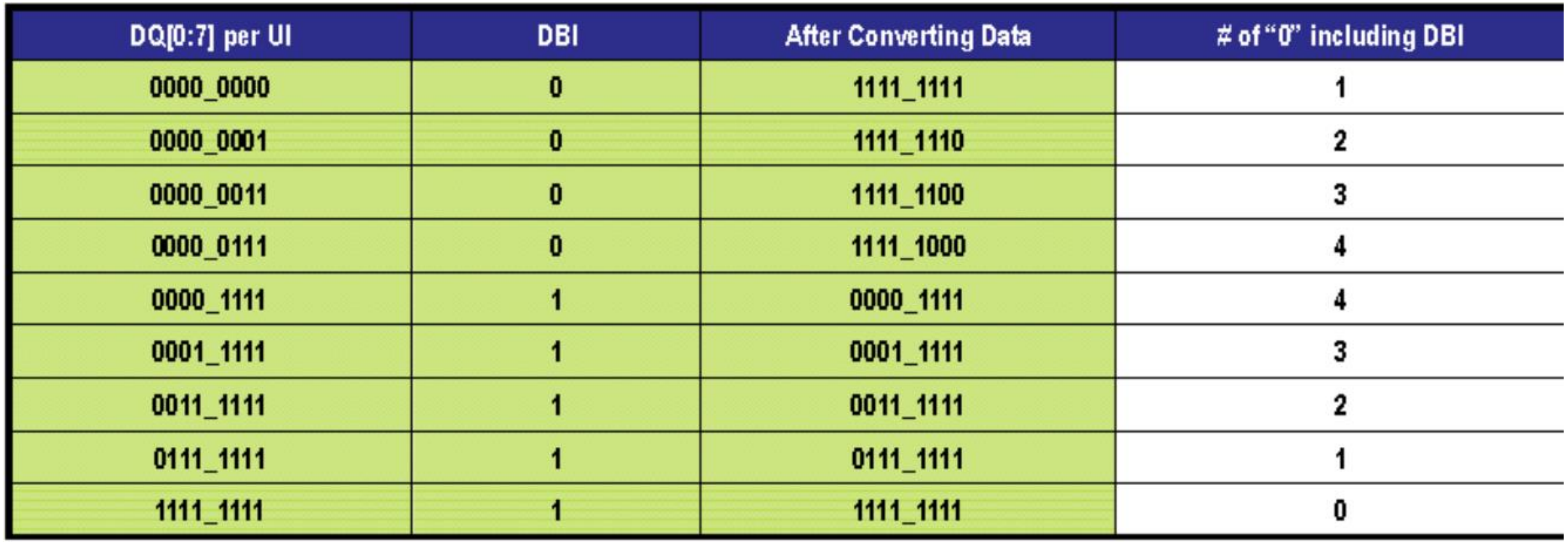
## LPDDR3 VS LPDDR4

**Data Bus Inversion (DBI):** Data Bus inversion is a power saving DDR4/LPDDR4 specific feature which aims at minimizing:

DDR4: Logic 0
LPDDR4: Logic 1
Simultaneous switching outputs

DBI removes a significant amount of SSO noise from data transactions.

**Example：** DDR4

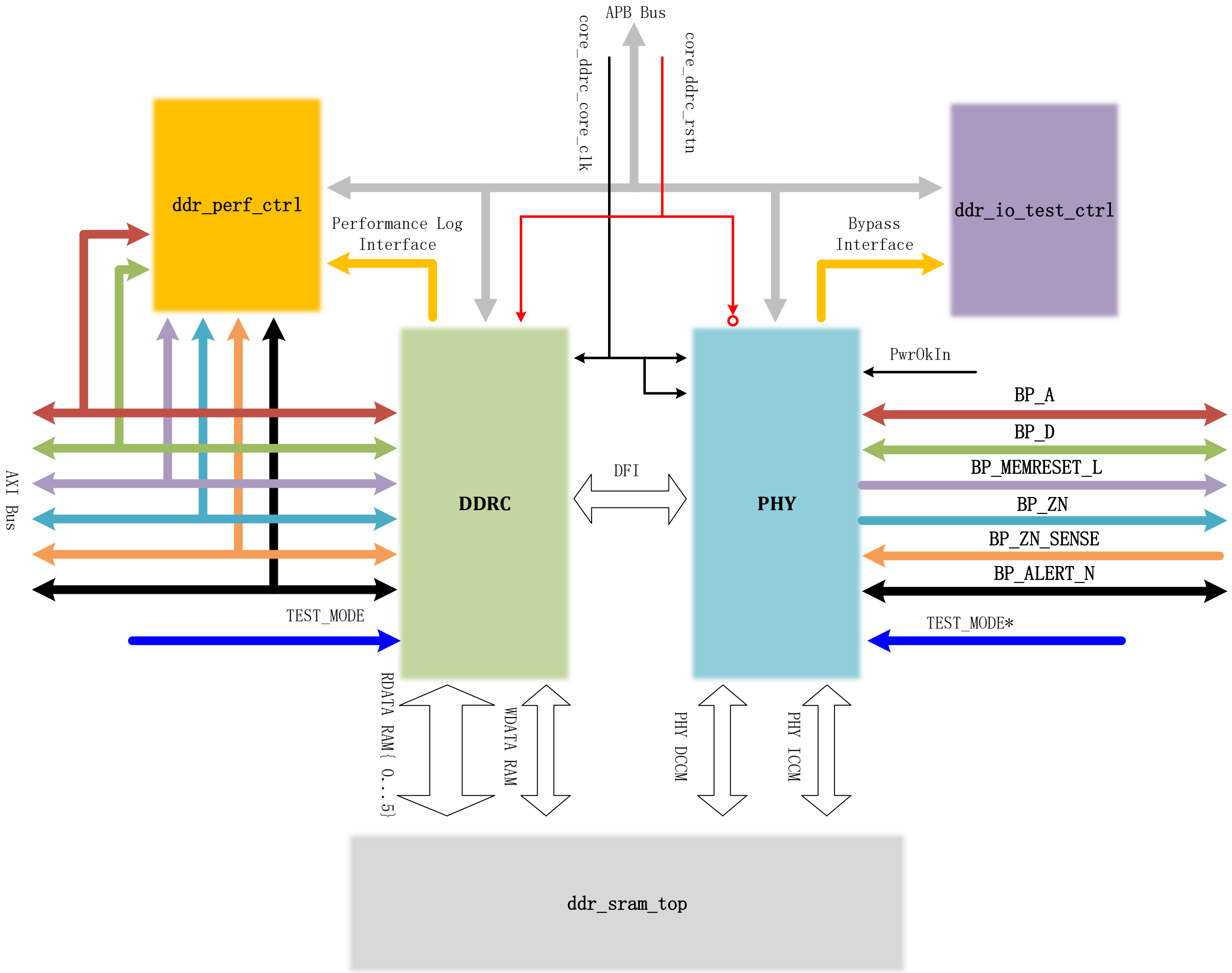


| DQ[0:7] per UI | DBI | After Converting Data | # of "0" including DBI |
|---|---|---|---|
| 0000_0000 | 0 | 1111_1111 | 1 |
| 0000_0001 | 0 | 1111_1110 | 2 |
| 0000_0011 | 0 | 1111_1100 | 3 |
| 0000_0111 | 0 | 1111_1000 | 4 |
| 0000_1111 | 1 | 0000_1111 | 4 |
| 0001_1111 | 1 | 0001_1111 | 3 |
| 0011_1111 | 1 | 0011_1111 | 2 |
| 0111_1111 | 1 | 0111_1111 | 1 |
| 1111_1111 | 1 | 1111_1111 | 0 |

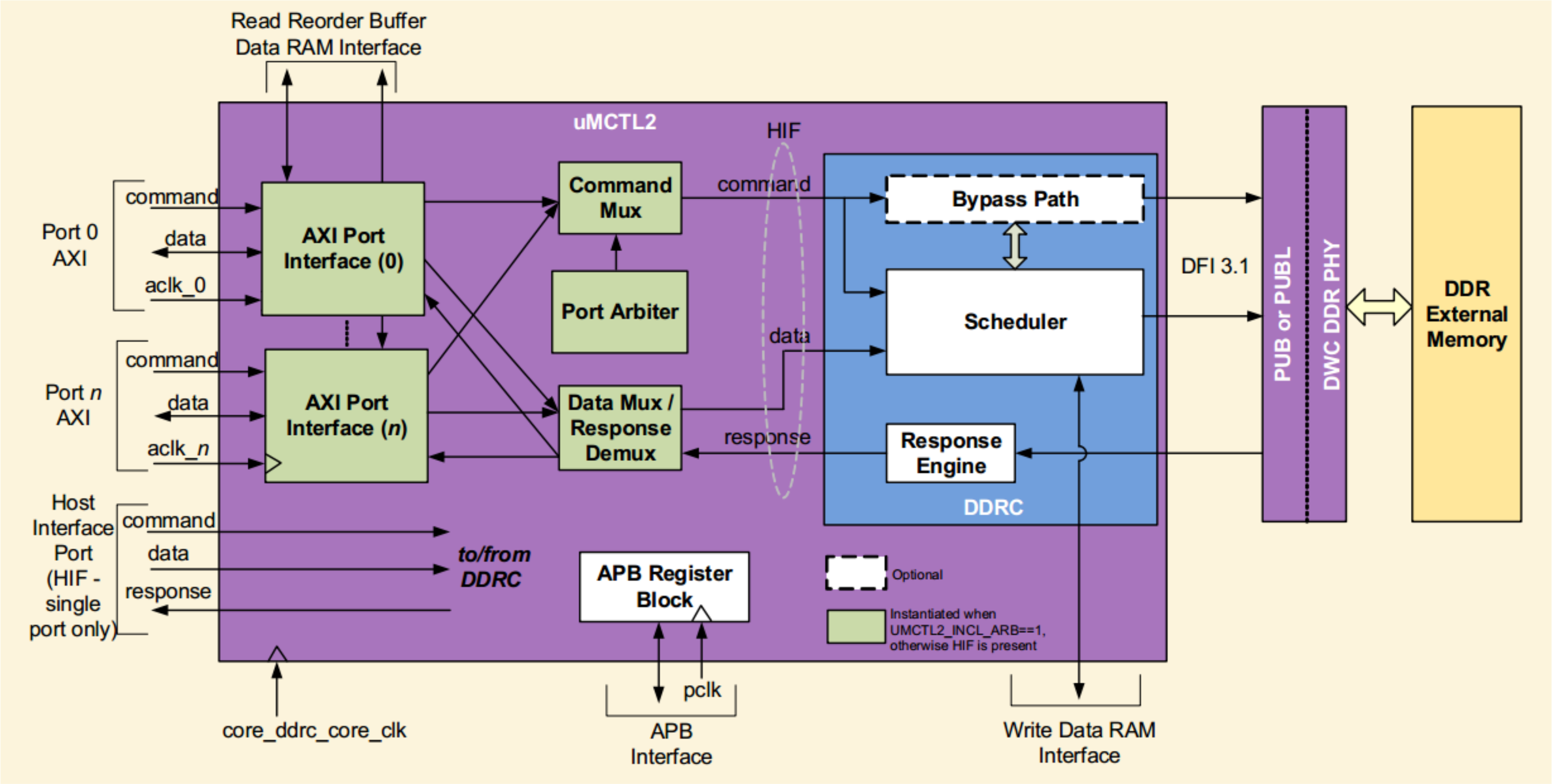
# Part 4

## DDR Memory Controller

## X2 DDR System-Level Block Diagram



## DDR Memory Controller Overview



## Major Features

- Multiple memory support for DDR4/3/2 and LPDDR4/4X/3/2
- The uMCTL2 is fully compatible with DFI 4.0 Addendum Version 2 to DFI 3.1
- Scalable 1:1/1:2 frequency ratio architecture.
- Up to 16 host ports using AMBA AXI/AHB
- APB interface for the uMCTL2 software accessible registers
- Read and write buffers in fully associative CAMs, configurable in powers of two, from 16 up to 64 reads and 64 writes
- Hardware configurable and software programmable Quality of Service (QoS) support
- Support 1, 2, or 4 memory ranks
- Support self-refresh entry and exit
- Support for dynamically changing clock frequency while in self-refresh
- Flexible address mapper logic to allow application specific mapping of row, column, bank, and rank bits

# DDR Controller

## DDR Memory Controller Performance

- ◆ For maximum SDRAM efficiency, commands are executed out-of-order
- ◆ CAM-based scheduling and reordering can reorder up to 64 read and 64 write commands
- ◆ Read Reorder Buffer ensures coherent and correctly in-order read responses on each AXI ID
- ◆ Low latency – 6 clock cycle minimum, 8 clock cycle nominal latency
- ◆ QoS features improve system-level performance
- ◆ Optional: Strictly in-order controller (uPCLT2) for external schedulers / NoCs

# DDR Controller

## Port Arbiter

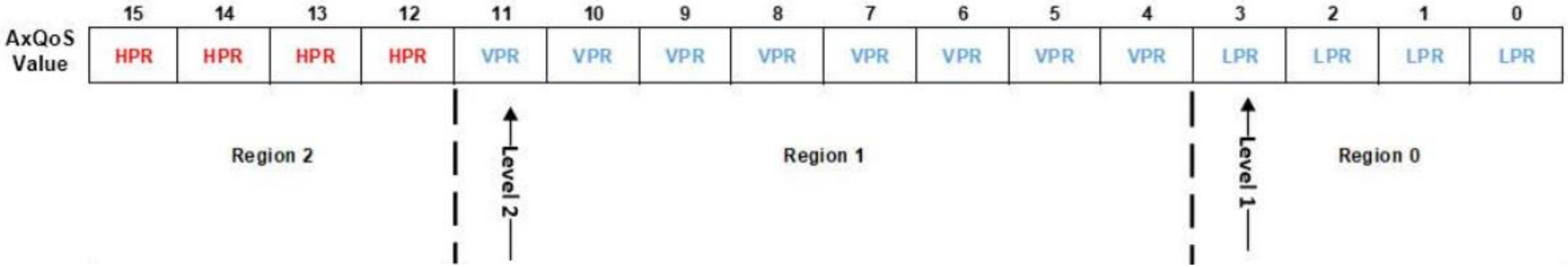
### QOS overview

**Read Classes**

Low Priority Read (LPR): (best effort, no guarantees)
Variable Priority Read (VPR): (maximum latency bound)
High Priority Read (HPR): (minimum latency)

**Write Classes**

Normal Priority Write (NPW): (best effort, no guarantees)
Variable Priority Write (VPW): (maximum latency bound)



|  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AxQoS Value | HPR | HPR | HPR | HPR | VPR | VPR | VPR | VPR | VPR | VPR | VPR | VPR | LPR | LPR | LPR | LPR |

## Port Arbiter

- Read/Write Arbitration
- Port Timeout
- DDRC Read Priorities (HPR/LPR/VPR) and Write Priorities (NPW/VPW) for Ports
- Port Command Priority
- Round-Robin Arbitration

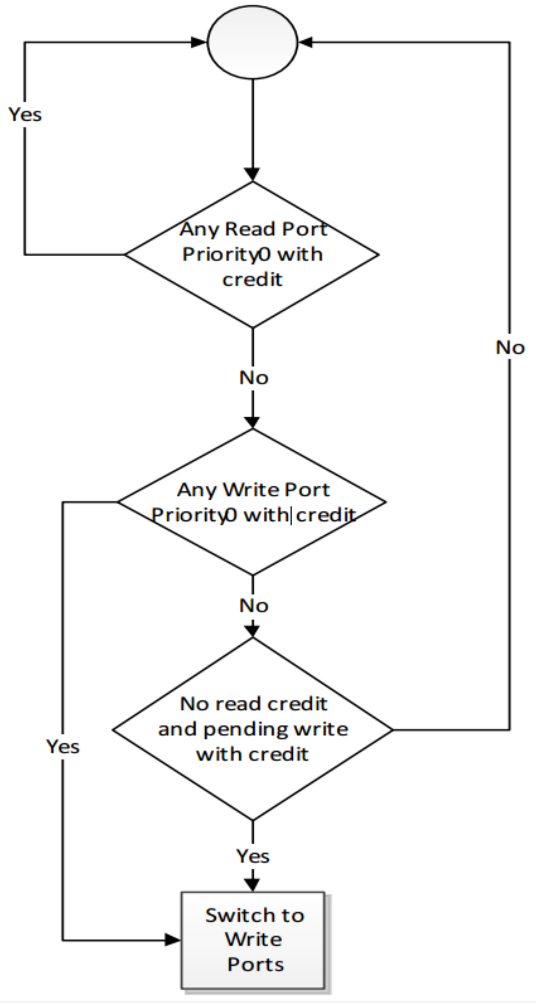
# Read/Write Arbitration

First level of arbitration , based on:

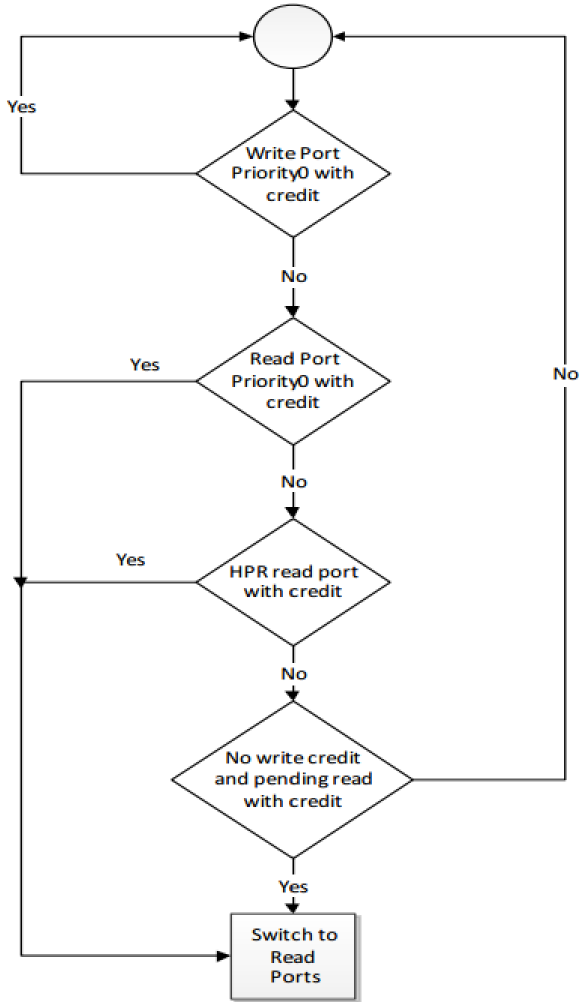
- Read (LPR-VPR, HPR) and write credits
- Timeout/Expired-VPR/Expired-VPW (Priority0)

If DUAL-HIF enabled , Read/Write Arbitration is not performed by the PA

Executing Read Ports


Yes
Any Read Port Priority0 with credit
No
Any Write Port Priority0 with credit
No
No read credit and pending write with credit
No
Yes
Yes
Switch to Write Ports

Executing Write Ports


Yes
Write Port Priority0 with credit
No
Yes
Read Port Priority0 with credit
No
No
Yes
HPR read port with credit
No
No write credit and pending read with credit
Yes
Switch to Read Ports

## Port Timeout

- Aging counters ( per port, per direction (read/write) )
- when a port aging counter becomes 0 and the port becomes the highest priority requester (Priority0)
- The aging feature and the timeout can be enabled by registers

## DDRC Read Priorities and Write Priorities

- ◆ initial priority : HPR > (VPR=LPR) , NPW =VPW
- ◆ VPR (not expired) = LPR
- ◆ VPR (expired) > HPR or writes.
- ◆ VPW (not expired) = NPW
- ◆ VPW (expired) > NPW or HPR.

## Port Command Priority

### External Port Priorities (UMCTL2_EXT_PORTPRIO = 1)

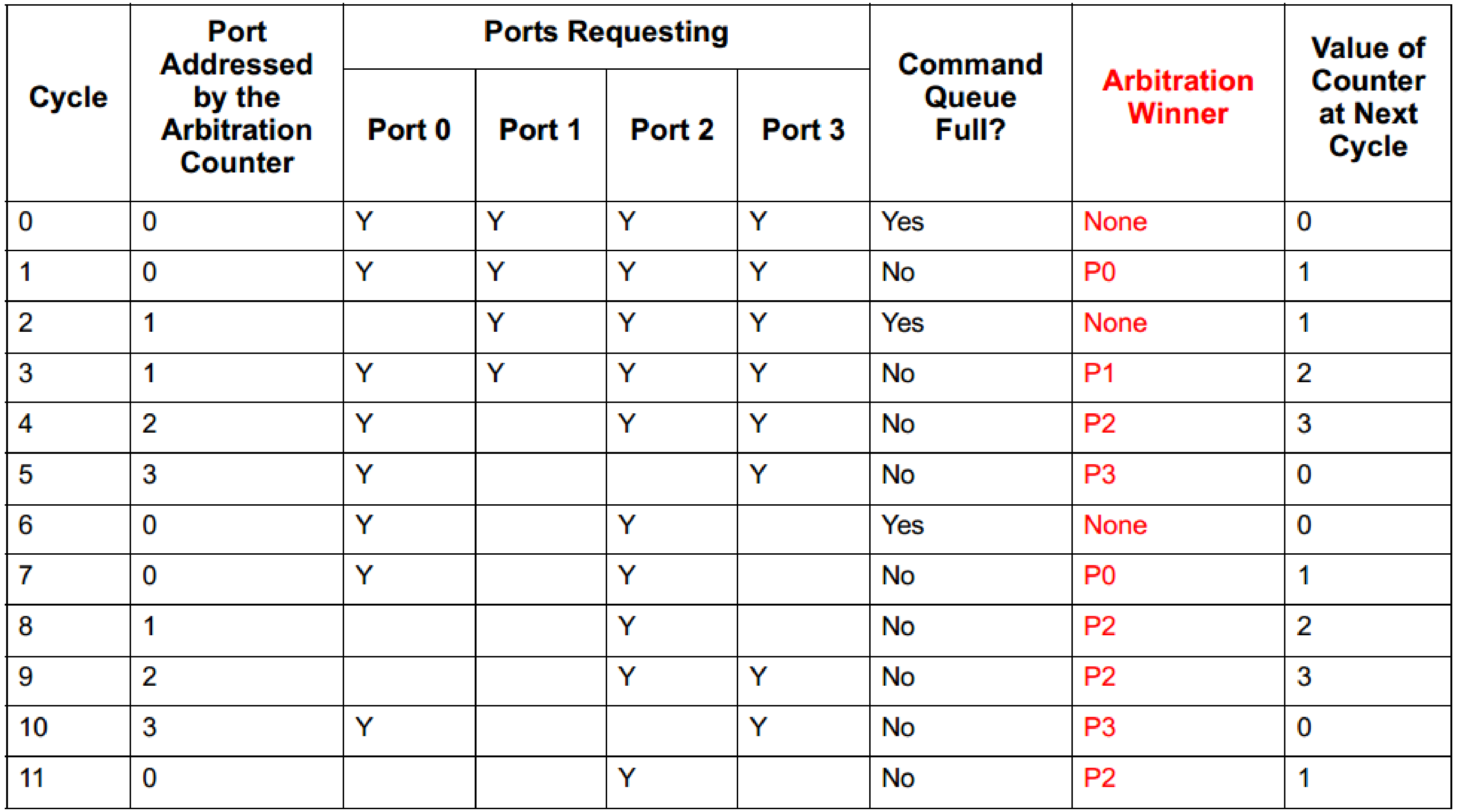
- The priorities are per-command and can dynamically change for a given port based on AXI AxQoS signals (arqos/awqos).
- There are 16-level priorities set by 4-bit wide arqos/awqos signals where the higher binary value represents a higher priority.

### Internal Port Priorities (UMCTL2_EXT_PORTPRIO = 0)

- Another way of setting port priorities is through port aging counters.
- The upper-most 5 bits of the counter determine the port priority out of 32 levels

## Round-Robin Arbitration

round-robin : the final arbitration stage.



| Cycle | Port Addressed by the Arbitration Counter | Ports Requesting | | | | Command Queue Full? | Arbitration Winner | Value of Counter at Next Cycle |
|---|---|---|---|---|---|---|---|---|
| | | Port 0 | Port 1 | Port 2 | Port 3 | | | |
| 0 | 0 | Y | Y | Y | Y | Yes | None | 0 |
| 1 | 0 | Y | Y | Y | Y | No | P0 | 1 |
| 2 | 1 | | Y | Y | Y | Yes | None | 1 |
| 3 | 1 | Y | Y | Y | Y | No | P1 | 2 |
| 4 | 2 | Y | | Y | Y | No | P2 | 3 |
| 5 | 3 | Y | | | Y | No | P3 | 0 |
| 6 | 0 | Y | | Y | | Yes | None | 0 |
| 7 | 0 | Y | | Y | | No | P0 | 1 |
| 8 | 1 | | | Y | | No | P2 | 2 |
| 9 | 2 | | | Y | Y | No | P2 | 3 |
| 10 | 3 | Y | | | Y | No | P3 | 0 |
| 11 | 0 | | | Y | | No | P2 | 1 |

## Transaction Service Control (DDRC Part)

- Transaction Stores
- Read/Write Turn-around
- Address Collision Handling

# Transaction Stores

## Read Transaction Stores

SCHED.lpr_num_entries register splits the Read CAM in the controller into LPR and HPR sections. The LPR and VPR commands are sent to the LPR section of the CAM, and the HPR commands are sent to the HPR section.

## Write Transaction Stores

The NPW and VPW commands are sent to the Write CAM. VPW commands do not have reserved space in the Write CAM.

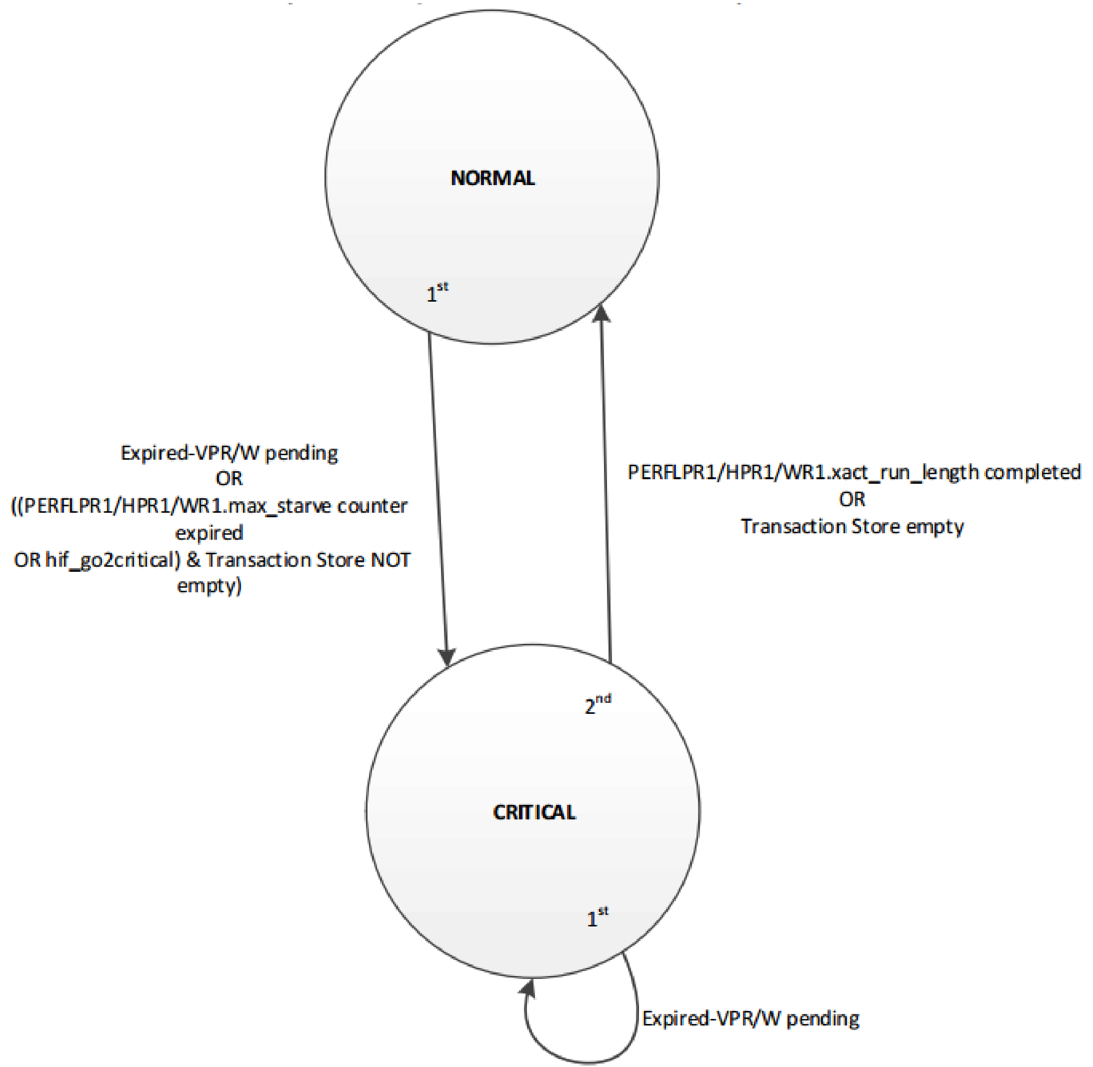
## Transaction Store State Transitions

Each class of transaction store (LPR, HPR, or WR) can be in any one of the following two states:

- Normal: It is the state where the transaction store starts.
- Critical: It indicates that the transaction store must be prioritized for service.

The transaction stores move between these two states under the control of *_max_starve and *_xact_run_length registers.

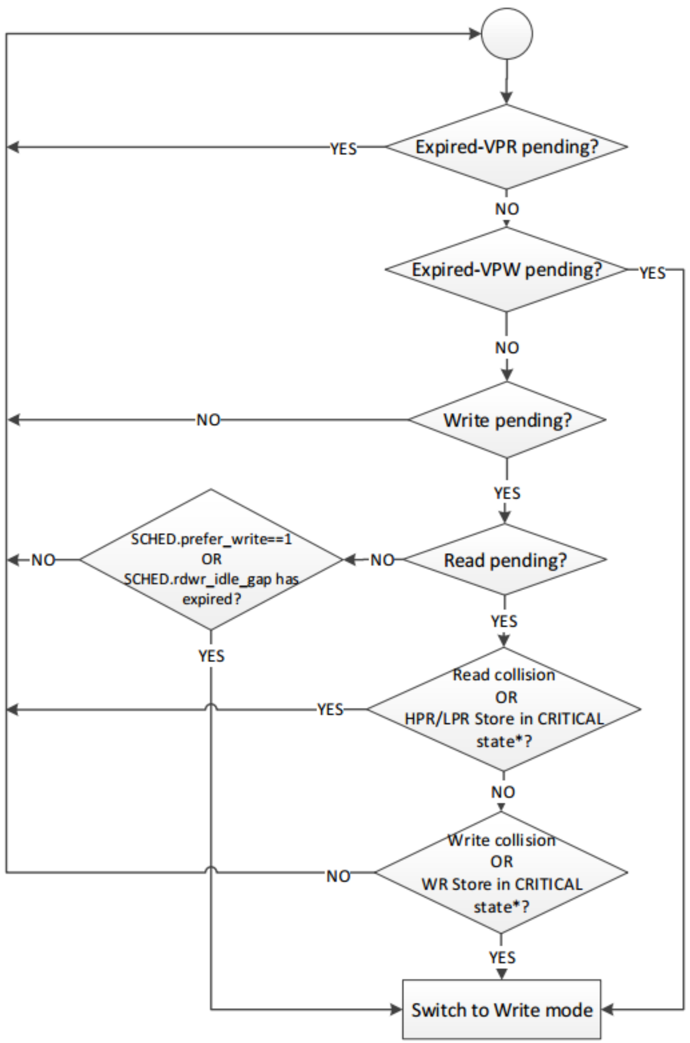
# Transaction Store FSM (One FSM per store - WR, LPR, HPR)



NORMAL
1st
Expired-VPR/W pending
OR
((PERFLPR1/HPR1/WR1.max_starve counter
expired
OR hif_go2critical) & Transaction Store NOT
empty)
PERFLPR1/HPR1/WR1.xact_run_length completed
OR
Transaction Store empty
2nd
CRITICAL
1st
Expired-VPR/W pending

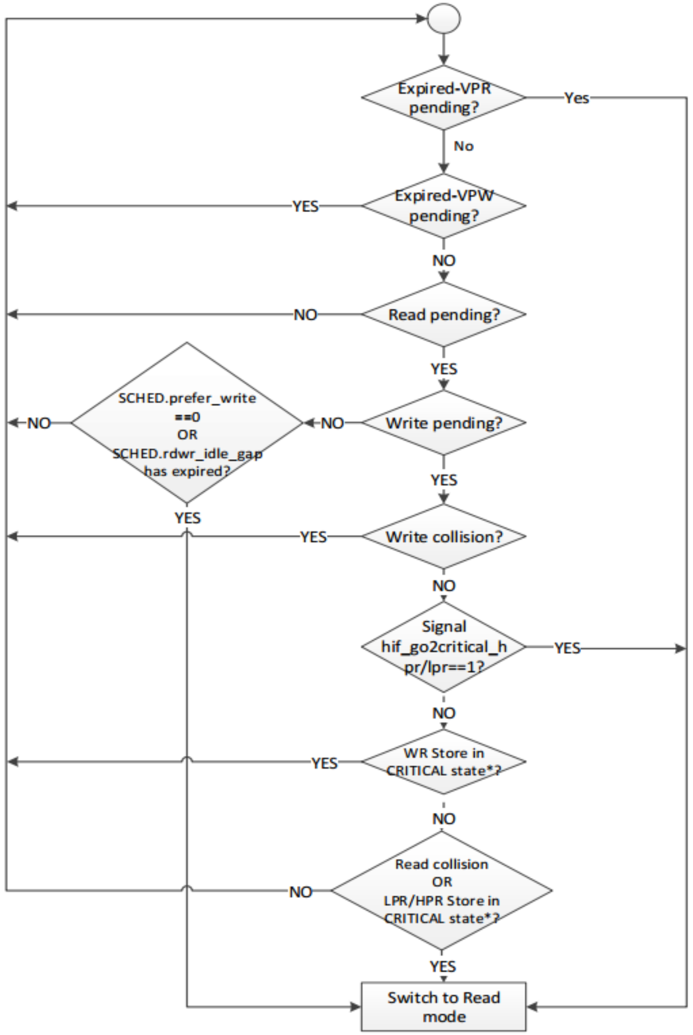
## Read/Write Turn-around

Executing Reads



* CRITICAL state – see Transaction Store FSM diagram

Executing Writes



* CRITICAL state – see Transaction Store FSM diagram

## Address Collision Handling

**New read colliding with queued read:** This collision causes no problems. The two reads can end up being executed out-of-order.

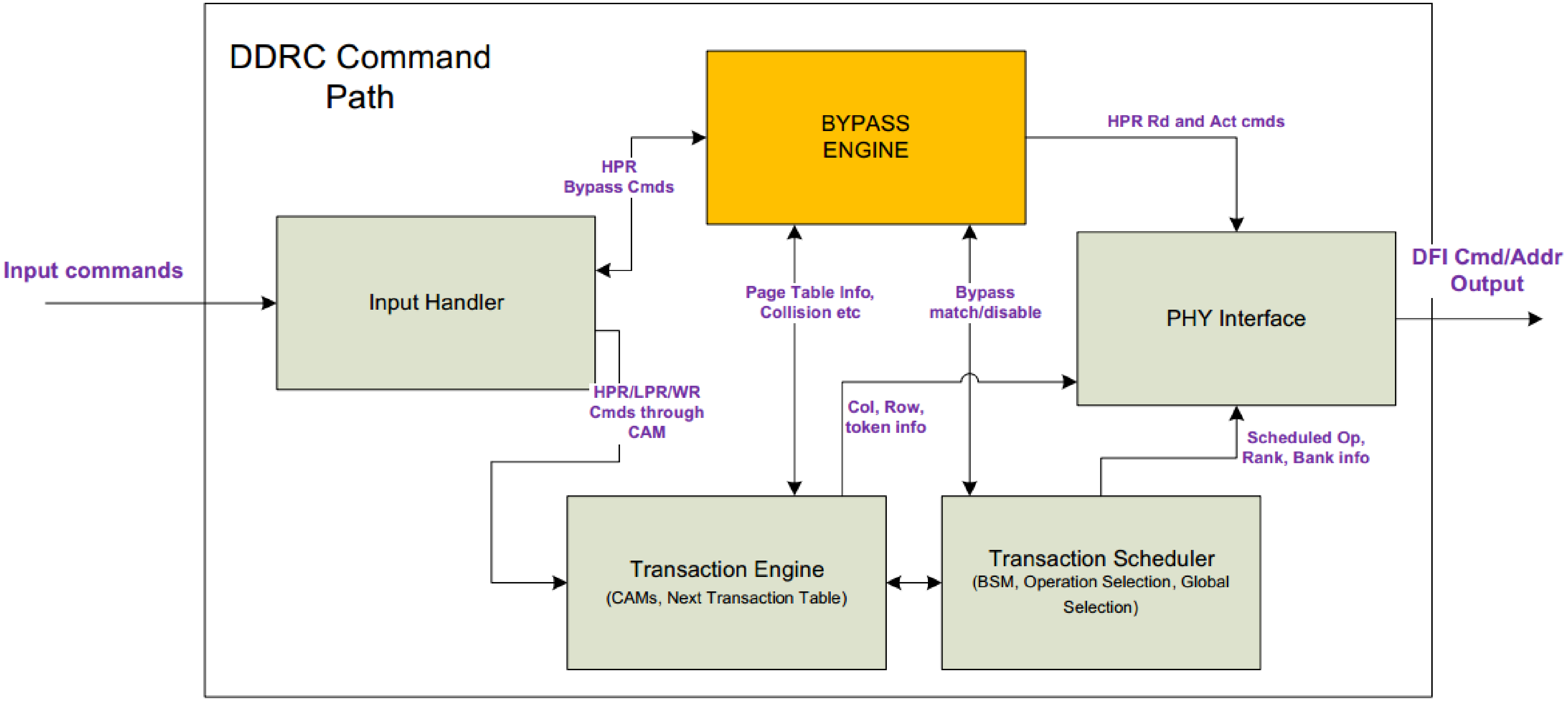
**New write colliding with queued write:** If write combine is enabled, the DDRC overwrites the data for the old write with that from the new write and only performs one write transaction (write combine).

**New read (or write) colliding with queued write (or read) respectively:** In this case, the DDRC performs the following sequence:

a. Holds the new transactions in a temporary buffer.
b. Applies flow control back to the SoC core to prevent more transactions from arriving.
c. Flushes the internal queue holding the colliding transaction until that transaction is serviced.
d. Accepts the new transaction and removes flow control.

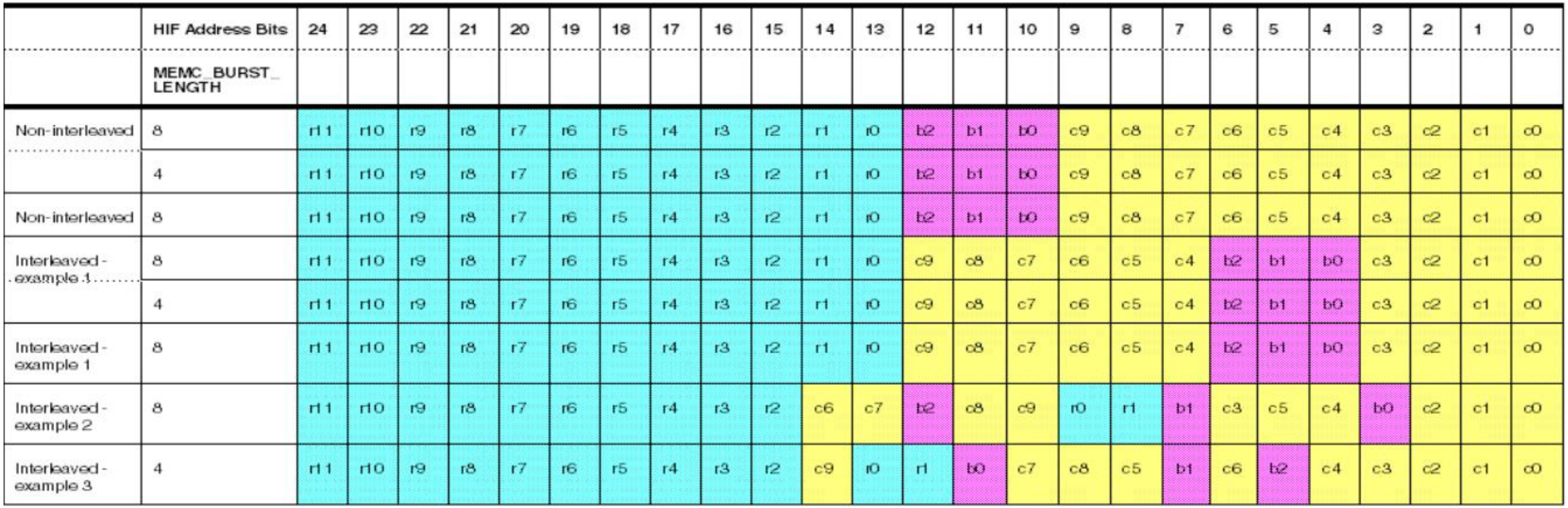
## Bypass Operation

The bypass engine in the uMCTL2 handles high-priority, low-latency read requests.



## Address Mapper

Flexible address mapper logic to allow application specific mapping of row, column, bank, and rank bits



| | HIF Address Bits | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MEMC_BURST_LENGTH | | | | | | | | | | | | | | | | | | | | | | | | | |
| Non-interleaved | 8 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 | b2 | b1 | b0 | c9 | c8 | c7 | c6 | c5 | c4 | c3 | c2 | c1 | c0 |
| | 4 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 | b2 | b1 | b0 | c9 | c8 | c7 | c6 | c5 | c4 | c3 | c2 | c1 | c0 |
| Non-interleaved | 8 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 | b2 | b1 | b0 | c9 | c8 | c7 | c6 | c5 | c4 | c3 | c2 | c1 | c0 |
| Interleaved - example 1 | 8 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 | c9 | c8 | c7 | c6 | c5 | c4 | b2 | b1 | b0 | c3 | c2 | c1 | c0 |
| | 4 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 | c9 | c8 | c7 | c6 | c5 | c4 | b2 | b1 | b0 | c3 | c2 | c1 | c0 |
| Interleaved - example 1 | 8 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | r1 | r0 | c9 | c8 | c7 | c6 | c5 | c4 | b2 | b1 | b0 | c3 | c2 | c1 | c0 |
| Interleaved - example 2 | 8 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | c6 | c7 | b2 | c8 | c9 | r0 | r1 | b1 | c3 | c5 | c4 | b0 | c2 | c1 | c0 |
| Interleaved - example 3 | 4 | r11 | r10 | r9 | r8 | r7 | r6 | r5 | r4 | r3 | r2 | c9 | r0 | r1 | b0 | c7 | c8 | c5 | b1 | c6 | b2 | c4 | c3 | c2 | c1 | c0 |

# THANKS!