DRAM controllers surveys

Created	@December 21, 2024 7:49 PM
Class	

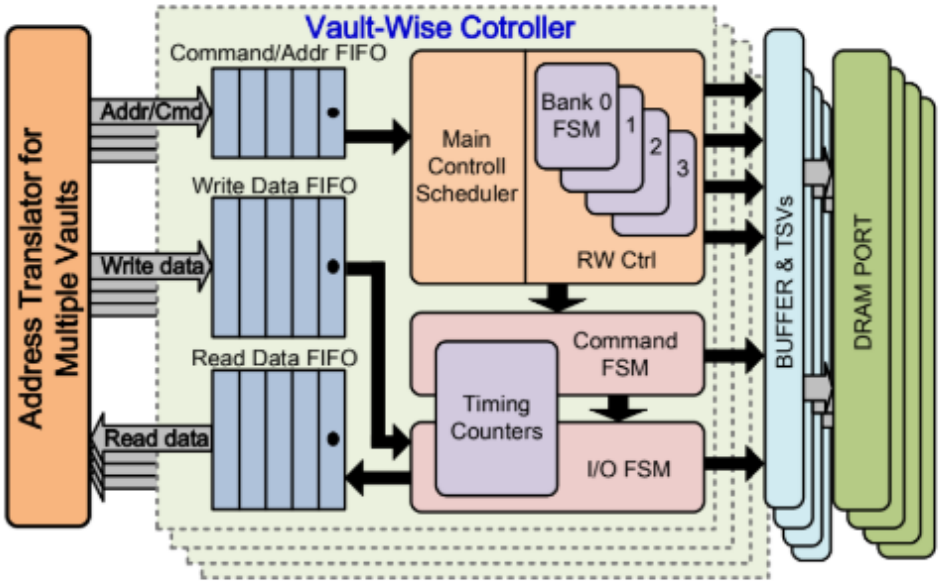


圖 24：分散式記憶體管理單元(memory management unit, MMU)。

- Seniors slice controller with main control scheduler performing refresh rescheduling.
- Timing counters obeying the timing constraints of specifications
- Command scheduling and bank status should be recorded
- Different power mode is supported, low power mode and high performance mode and IDLE mode for making DRAM entering the self-refresh mode, the control option must be included within the DRAM controller interfaces
- Thermal aware refresh control to adaptively change the refresh according to the temperature of the DRAM

CACTI Single Bank timing constraints

Size: 1Gb

Bank Size: 256Mb ➡ **1Gb**

TSV/Bank: 1024 ➡ **TSV/4Bank(1 vault): 1024**

Number of stacks: 4 **(burst length = 4 for each vault)**

Bank/layer: 1

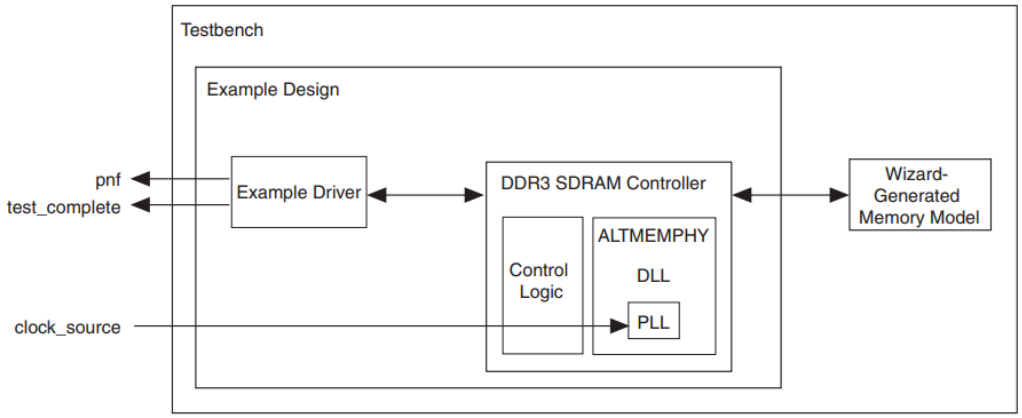
Page Size: 2KB

Fine-grained rank level

- Split out 4 × 1024 bits data at one access, this is the goal

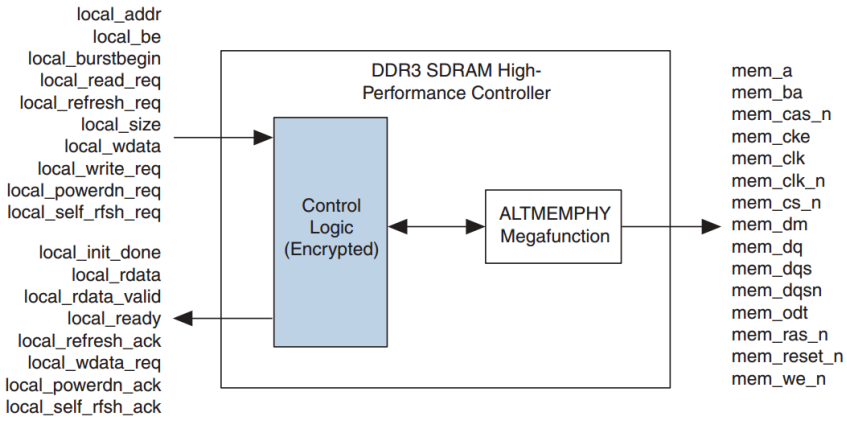
DDR3 SDRAM HP Controller

Figure 4–4. Testbench & Example Design



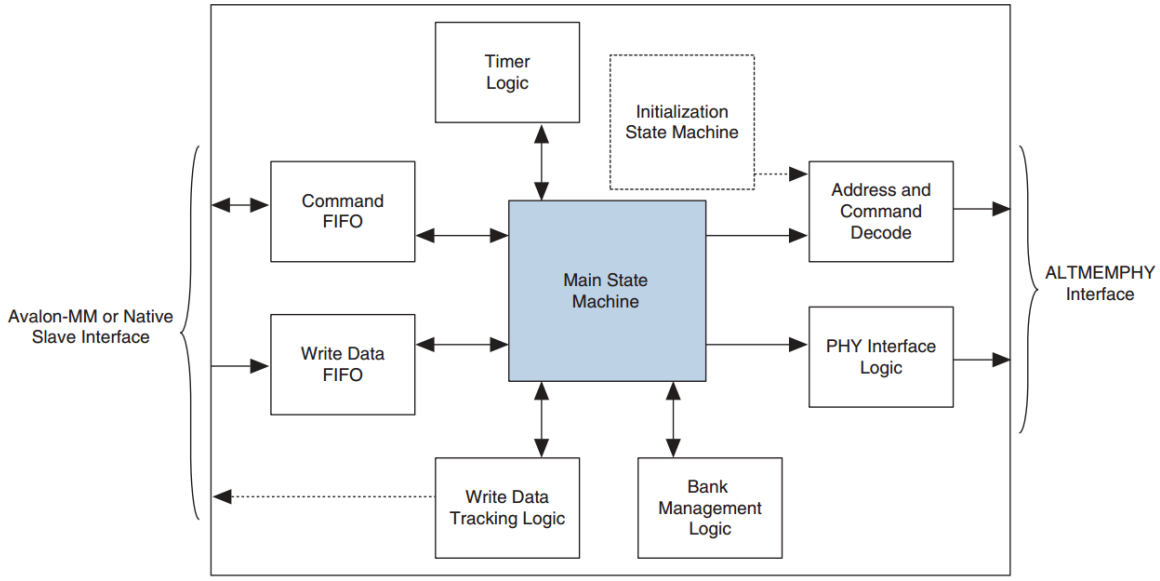
- The driver architecture and the usage of its controller

Figure 4–1. DDR3 SDRAM High-Performance Controller Block Diagram



- Interface blocks of DDR3 high performance controller of Altera Corporation

Figure 4–2. DDR3 SDRAM High-Performance Controller Architecture Block Diagram



- Refer to HPC memory controller for more details about what each block is doing

Testing

Table 4–4. Test Status[] Bit Mapping

Bit	Test
0	Sequential address test
1	Incomplete write test
2	Data mask pin test
3	Address pin test
4	Power-down test
5	Self-refresh test
6	Auto precharge test

- Important stuff for the test cases to verify that the design is working properly! Built up a test framework to perform such a testing.

It performs the following tests and loops back the tests indefinitely:

- Sequential addressing writes and reads

  The state machine writes pseudo-random data generated by a linear feedback shift register (LFSR) to a set of incrementing row, bank, and column addresses. The state machine then resets the LFSR, reads back the same set of addresses, and compares the data it receives against the expected data. You can adjust the length and pattern of the bursts that are written by changing the `MAX_ROW`, `MAX_BANK`, and `MAX_COL` constants in the example driver source code, and the entire memory space can be tested by adjusting these values. You can skip this test by setting the `test_seq_addr_on` signal to logic zero.

- Incomplete write operation

  The state machine issues a series of write requests that are less than the maximum burst size supported by your controller variation. The addresses are then read back to ensure that the controller has issued the correct signals to the memory. This test is only applicable in full-rate mode or in DDR3 half-rate mode, when the local burst size is two. You can skip this test by setting the `test_incomplete_writes_on` signal to logic zero.

- Byte enable/data mask pin operation

  The state machine issues two sets of write commands, the first of which clears a range of addresses. The second set of write commands has only one byte enable bit asserted. The state machine then issues a read request to the same addresses and the data is verified. This test checks if the data mask pins are operating correctly. You can skip this test by setting the `test_dm_pin_on` signal to logic zero.

- Address pin operation

  The example driver generates a series of write and read requests starting with an all-zeros pattern, a walking-one pattern, a walking-zero pattern, and ending with an all-zeros pattern. This test checks to make sure that all the individual address bits are operating correctly. You can skip this test by setting the `test_addr_pin_on` signal to logic zero.

- Low-power mode operation

  The example driver requests the controller to place the memory into power-down and self-refresh states, and hold it in those states for the amount of time specified by the `COUNTER_VALUE` signal. You can vary this value to adjust the duration the memory is kept in the low-power states. This test is only available if your controller variation enables the low-power mode option.

- DRAM controller test cases to ensure the correct operations of the design, testing progress is measured by coverage and test conditions

Xilinx DDR SDRAM Controller

DDR SDRAM 接口的实现

这一部分介绍了 DDR SDRAM (DDR SDRAM controller) 控制器和接口的参数、接口框图（图 6）和控制器状态机（图 7）。

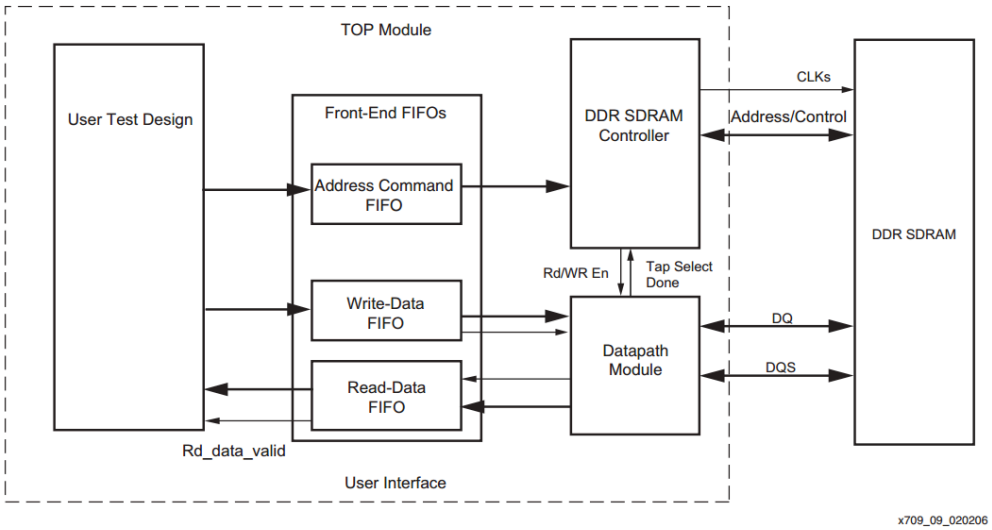
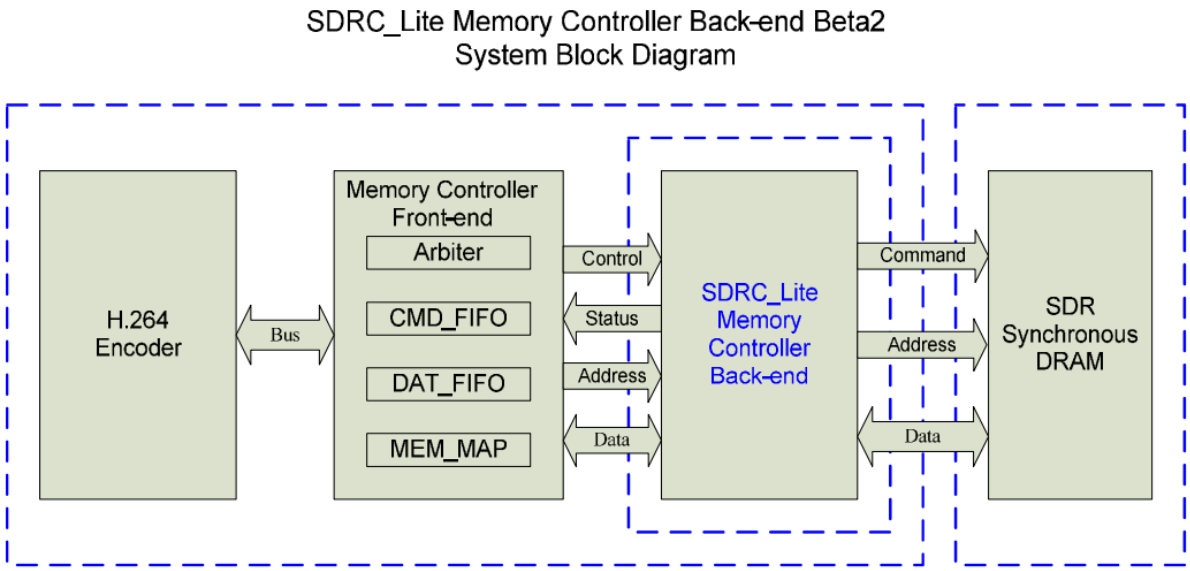


图 6: **控制器设计框图**

用户接口 (User Interface)

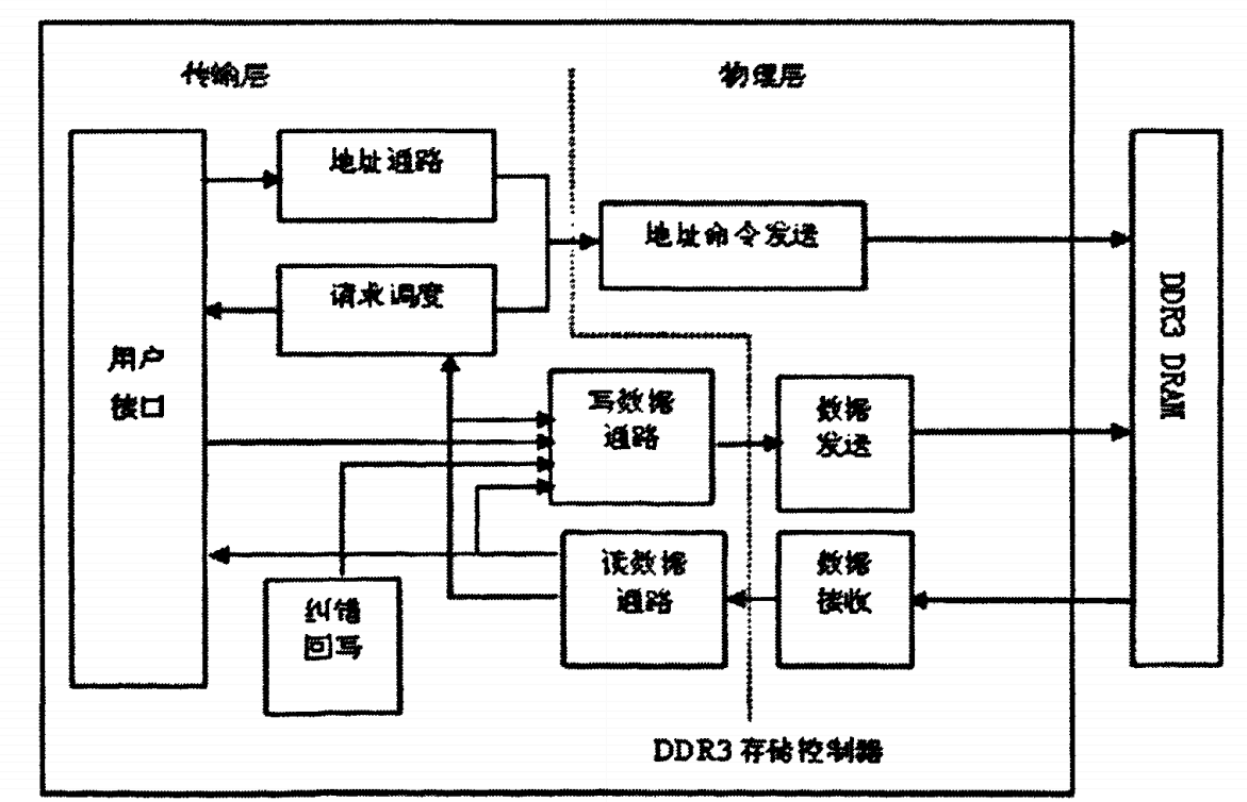
后端用户接口由三个 FIFO 构成，即地址命令 FIFO (Address Command FIFO)、写数据 FIFO (Write–Data FIFO) 和读数据 FIFO (Read–Data FIFO)。前两个 FIFO 通过用户后端模块访问，而读数据 FIFO 则通过数据通路模块 (Datapath Module) 访问，以存储采集的读数据。

SRC_LITE



- The defined interface for accessing the frontend accelerator,

DDR Controller



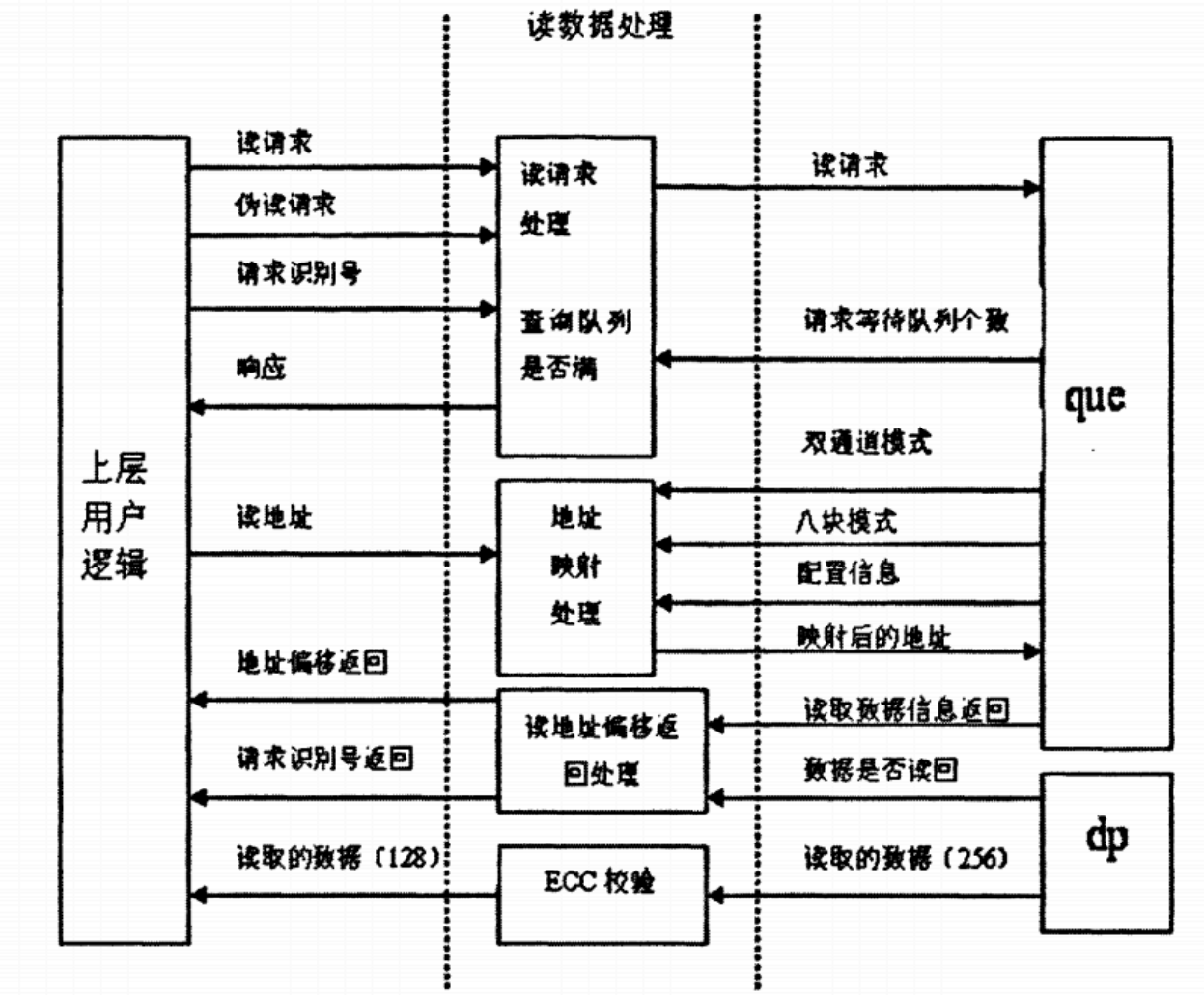


图 4.7　UIB 读请求处理

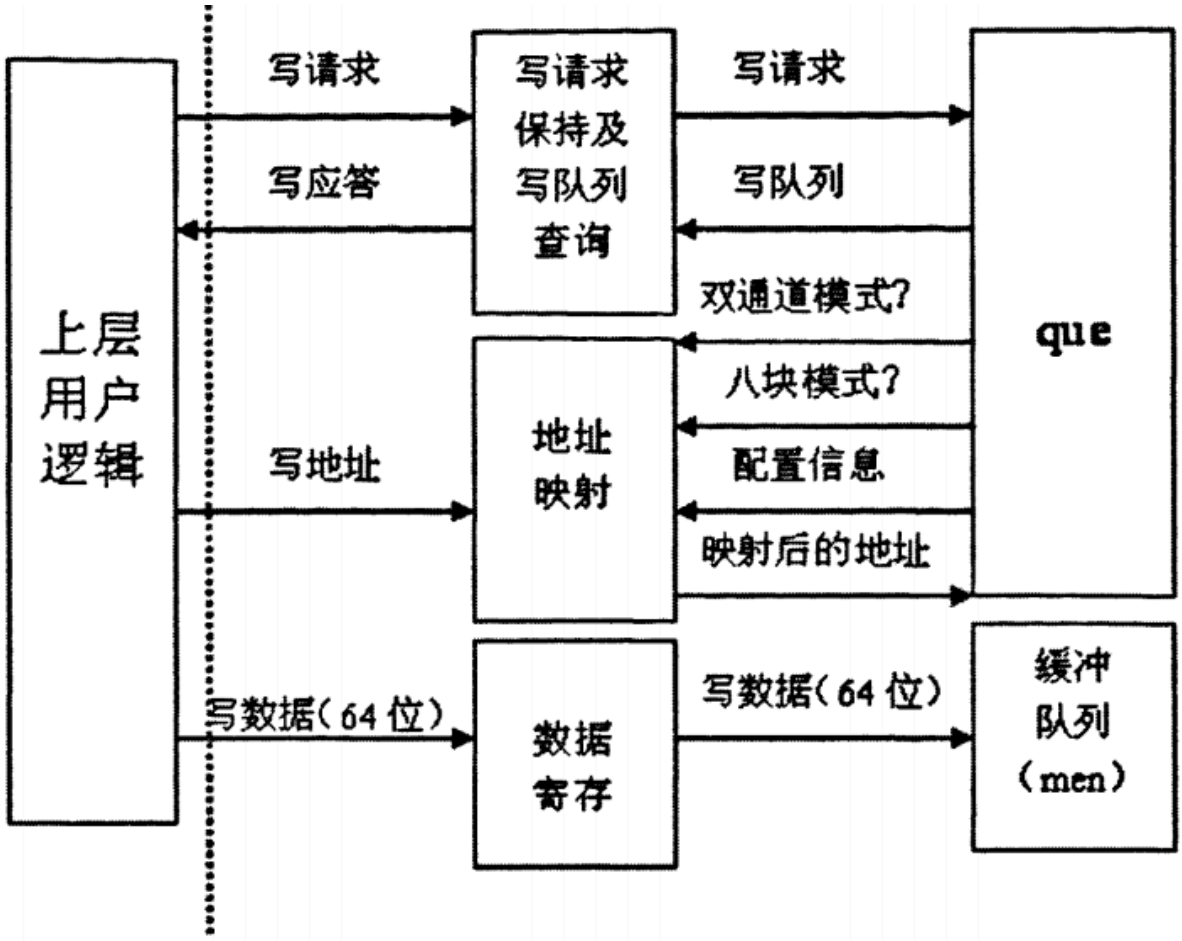


图 4.9　写请求处理处理模块

4.3 请求调度模块 ARB

请求调度模块完成各种存储器请求的仲裁和调度。存储控制器中需要进行存储器访问的请求有：上层用户逻辑发来的存储器读写请求、纠错回写模块发出的存储器读写请求、存储器的定时刷新请求等。请求调度模块的核心就是各种请求的仲裁，仲裁器按照一定的优先级仲裁算法将这些访存请求发送给物理层。如何选择仲裁算法是本小节研究的主要内容。

为了合理高效地控制和管理系统中命令的传输，必须使用一个特别的优先级仲裁算法，以便在多个访存请求同时提出仲裁请求时，能够依据某仲裁算法判决出哪个请求可优先访问存储器。仲裁算法必须确保任何时刻只有一条命令发送给物理层。DDR3存储控制器的仲裁器是基于访问的，也就是说，一个请求模块必须

- An arbitrator is needed to issue different requests to the backend of memory controller.

2、请求类型仲裁

在 DDR3 控制器中，对于不同类型的访存请求，仲裁的常规仲裁优先级如下：

●CAS 请求：最高优先级，保证请求尽快完成；

●纠错回写 RAS 请求；

●与读队列请求有地址相关性的写操作 RAS 请求；

●读队列中的读 RAS 请求，或者当饥饿计数器（starvation counter）达到

第 36 页

国防科学技术大学研究生院工程硕士学位论文

极限时写队列中的写 RAS 请求；

●写队列中的写 RAS 请求，或者当饥饿计数器（starvation counter）达到极限时读队列中读 RAS 请求；

●新进入的读 RAS 请求。

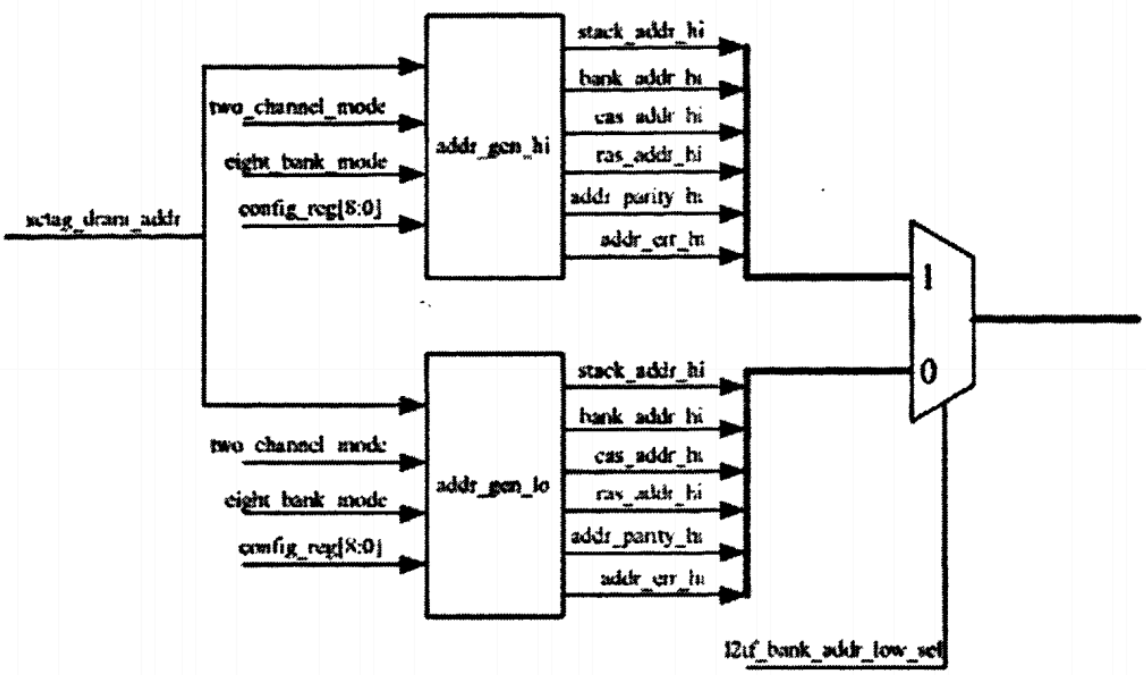


图 4.15　物理地址映射单元

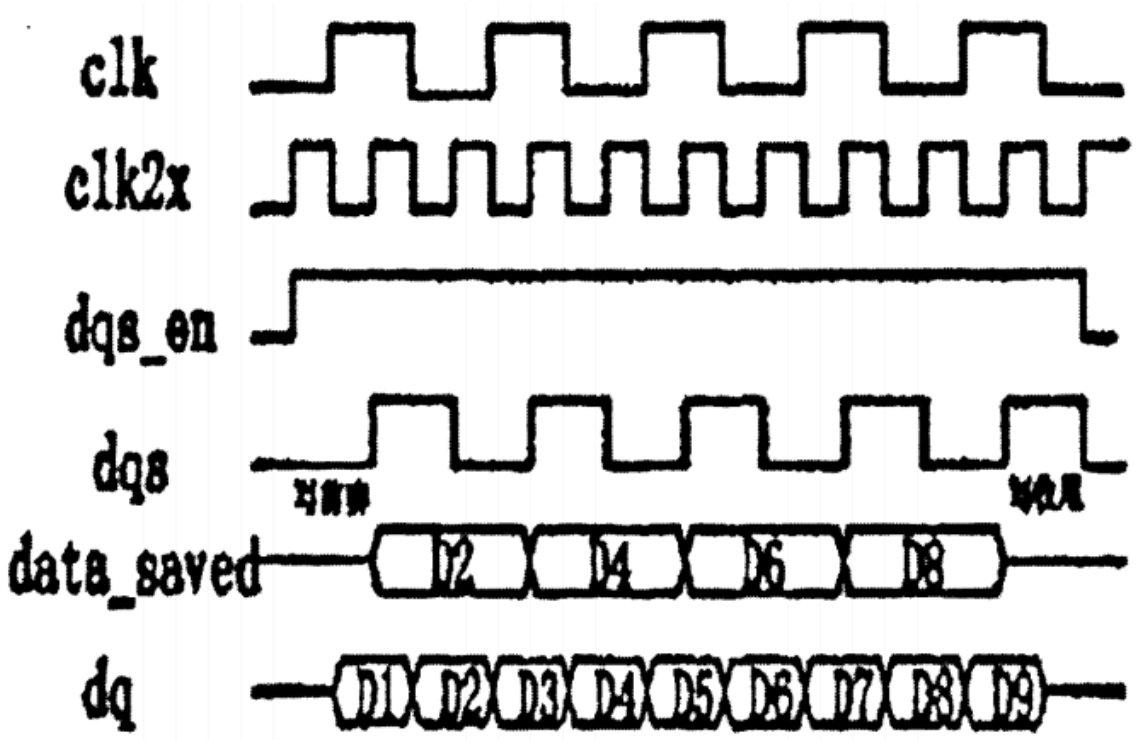


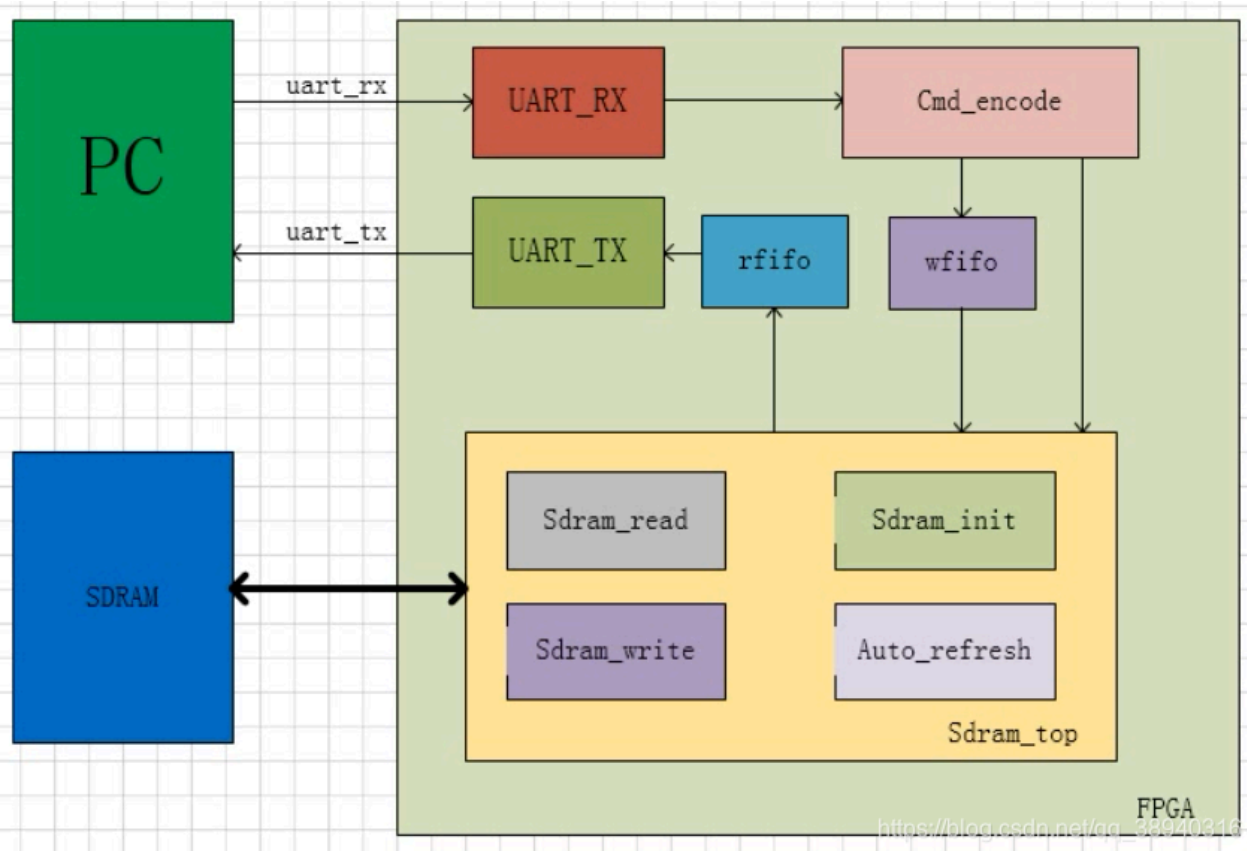
图 4.20　写结构时序图

5.3.3 数据接收

为了准确接收从 DRAM 存储器返回的数据，在 DDR3 DRAM 控制器的请求调度模

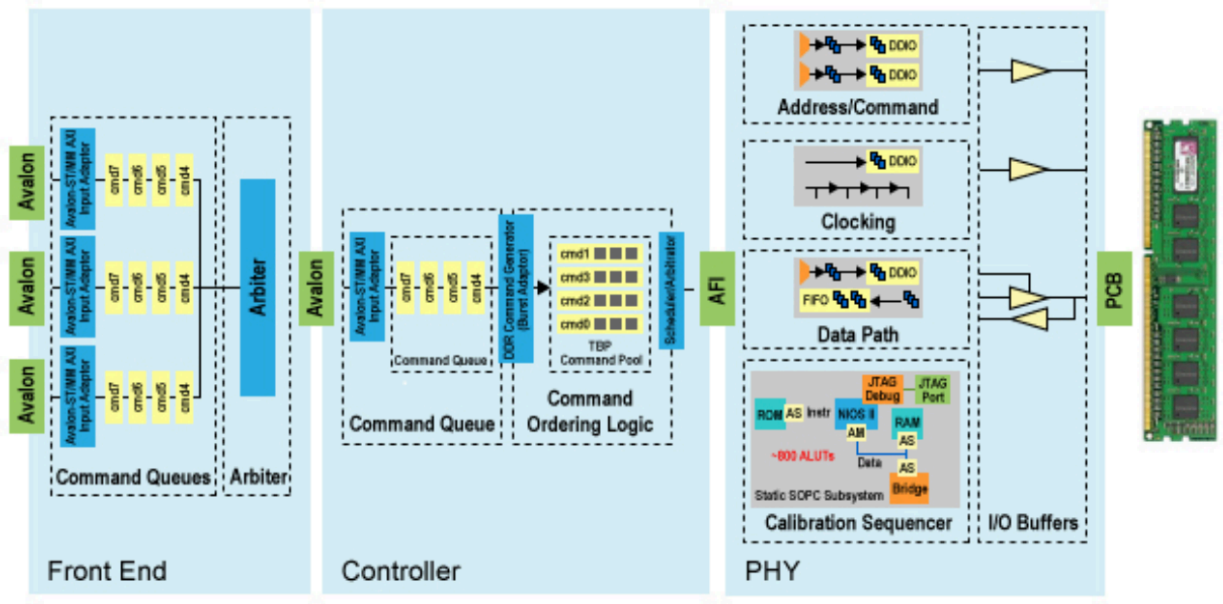
- Remember that within the physical layer, the capturing of the data in the edge is an important task. Dual edge problem might occurs~

SDRAM 那檔事



- Note arbitration is needed in the SDRAM top for the RD,WRITE & AUTO Refresh block
- Quickly build and test the simple DRAM controller to quickly grasp what should be done and what should not be using Assertions, systemVerilog interfaces

- Learn how to use systemVerilog language features and assertions also how DDR3 memory works in the process



- Notice that it is partitioned into multiple parts, frontend, controller and the back ends

Altera HPC controller

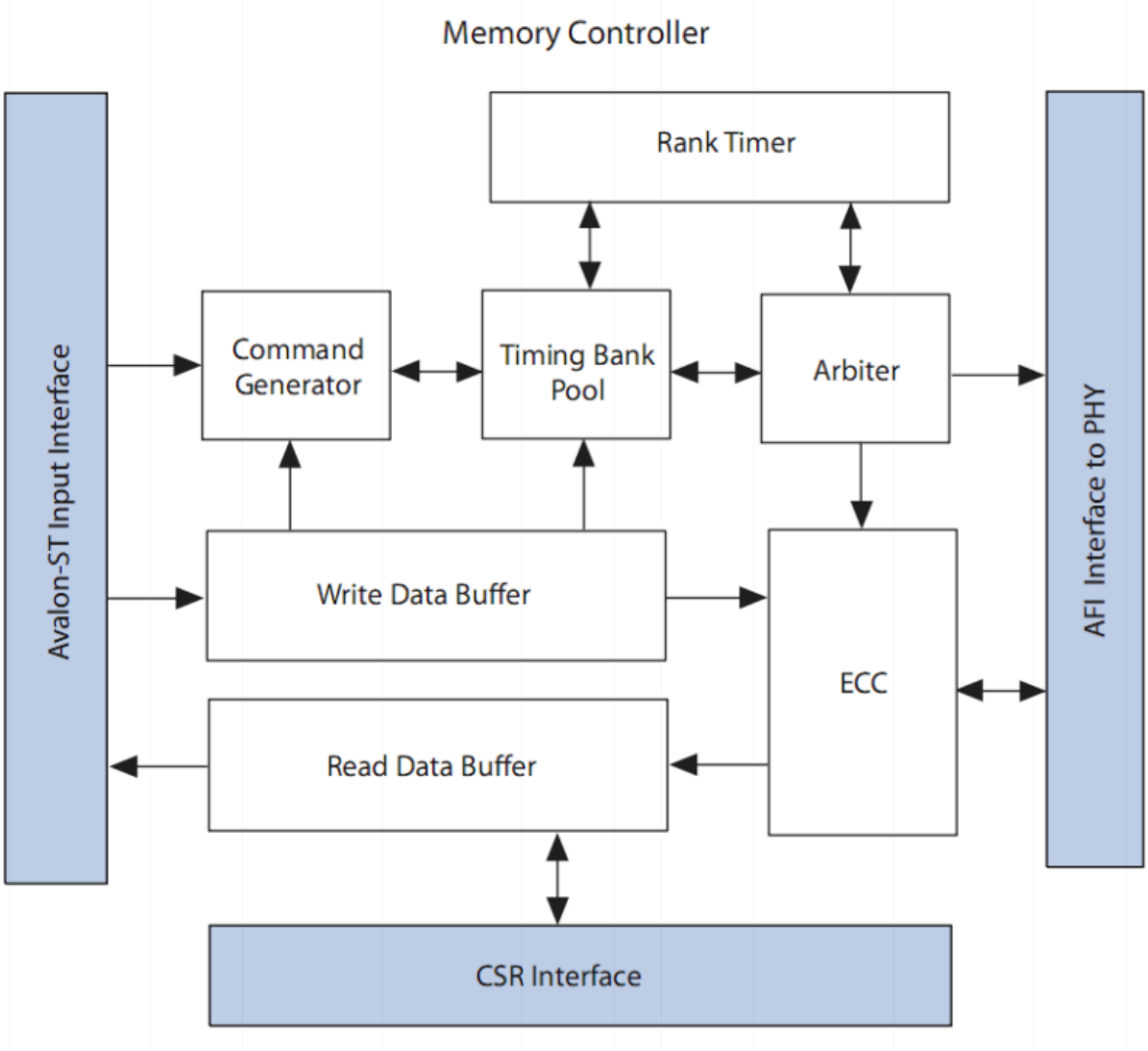


图3.10 Altera高性能内存控制器顶层结构[13]

Backend Open page consideration

第四章 DDR3 SDRAM控制器后端设计

本次毕业设计主要期望完成一个适用于实验室视频编解码系统的DDR3 SDRAM存储器控制器设计。考虑到视频编解码系统段读写请求具有较强的规律性，地址局部性特征明显，较为适合使用基于open page policy的存储器控制器设计。但同时基于open page policy的存储器控制器设计较为艰难，故先行完成了一个基于close page policy的轻量级存储器控制器，进而完成基于open page policy的存储器控制器设计。

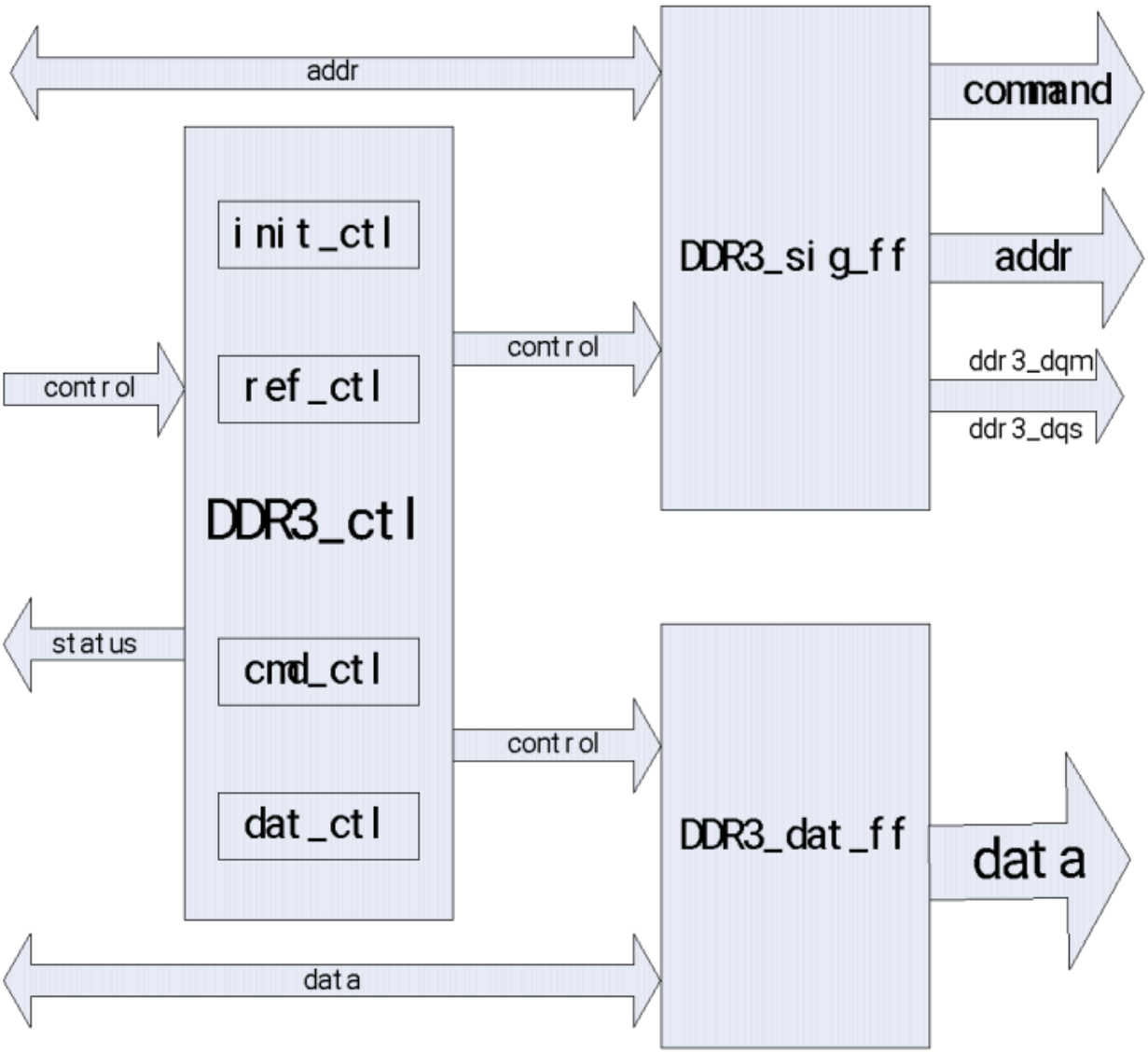


图4.2 基于close page policy策略控制器后端模块结构

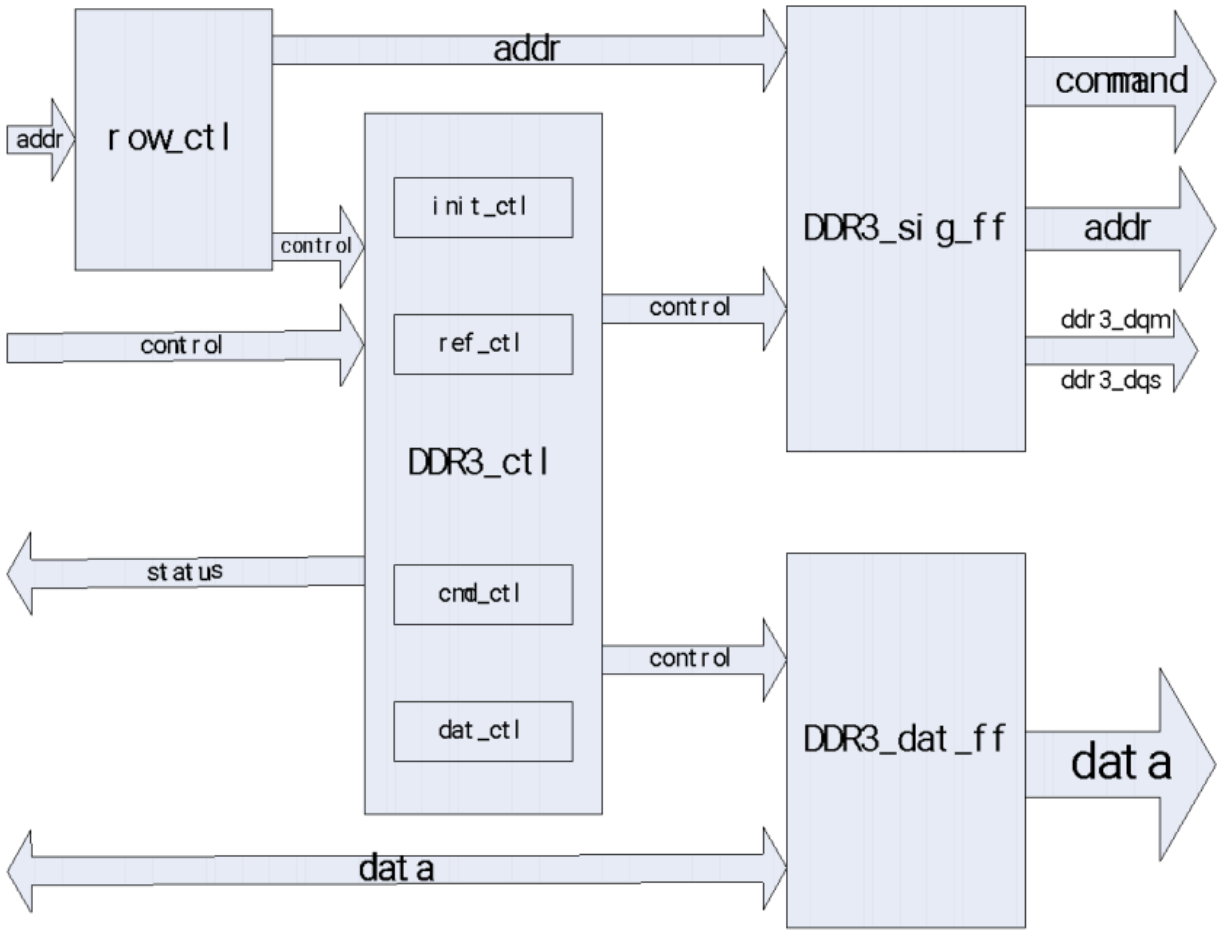


图4.4 基于open page policy策略控制器后端模块结构

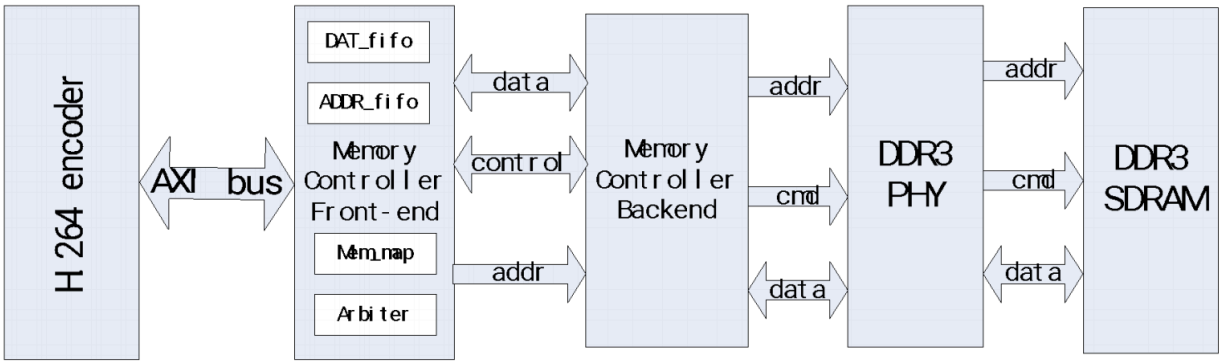


图5.1 控制器系统应用框图

图5.1 控制器系统应用框图

控制器前端实际上是相对于控制器后端提出的一个概念，一般而言控制器后端是通用的，在很多场合由于接口不匹配等原因还需要加上控制器前端来将控制器后端集成到大的系统中。在真正完成的成熟设计中经常会把前端和后端结合在一起，这样可以大大减少冗余逻辑并提高性能。

- Notice in usual cases, before completing a design, we would actually merge the frontend and the back end part for improved performance.
- But since the purpose is to design and ready-to-modify DRAM controller, seperation of part should be a lot more important!!!!

在本次设计中，并没有采用数据FIFO转换时钟域的方式，而是选择将数据写入双端口SRAM中，这样的设计有以下特点：

1、双端口SRAM两端输入频率可以不一致，通过双端口SRAM可以顺利完成时钟域的转换。

2、读写双端口SRAM需要地址，这也就意味着设计中不需要引入地址FIFO，直接使用SRAM的地址即可。

3、双端口SRAM可以作为DDR3　SDRAM的一个高速缓存，正常情况下系统只需要读写双端口SRAM就可以顺利完成数据通信，这将大大降低系统的读写时间开销。

4、每一笔数据不直接与DDR3　SDRAM打交道，而是在更新数据时大批写入DDR3 SDRAM，双端口SRAM实际上部分承担了判决器的功能。

5、引入双端口SRAM可以大大减轻存储器控制器后端的压力，只有在双端口SRAM需要更新数据的时候才会访问DDR3 SDRAM，且这种数据读写非常规律，一个非常简单的存储器控制器后端就可以高效率的完成这样的数据读写，同时根据数据读写的特征可以有针对性的设计控制器后端。

- The problem of CDC should be considered in future extensible design. However, synchronous design is also available.

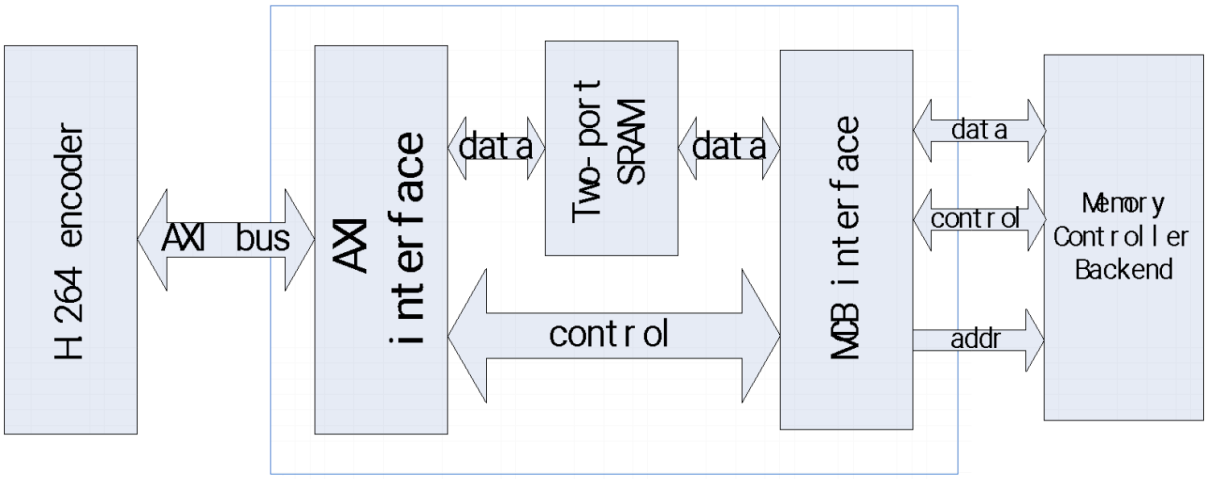


图5.2 控制器前端模块结构图

- The frontend of the memory controller to enable the connection and conversion of memory controller signals to interconnection compatible signals

Benini 3D Controller

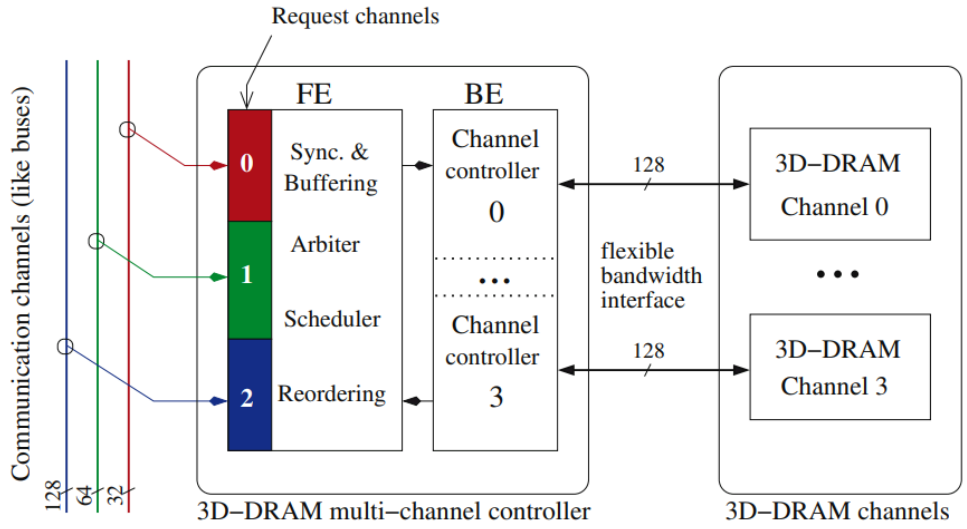


Fig. 1. 3D-DRAM subsystem - functional overview (including front-end (FE) and back-end (BE) of the controller, the request channels (RC) and the channel controllers (CC))

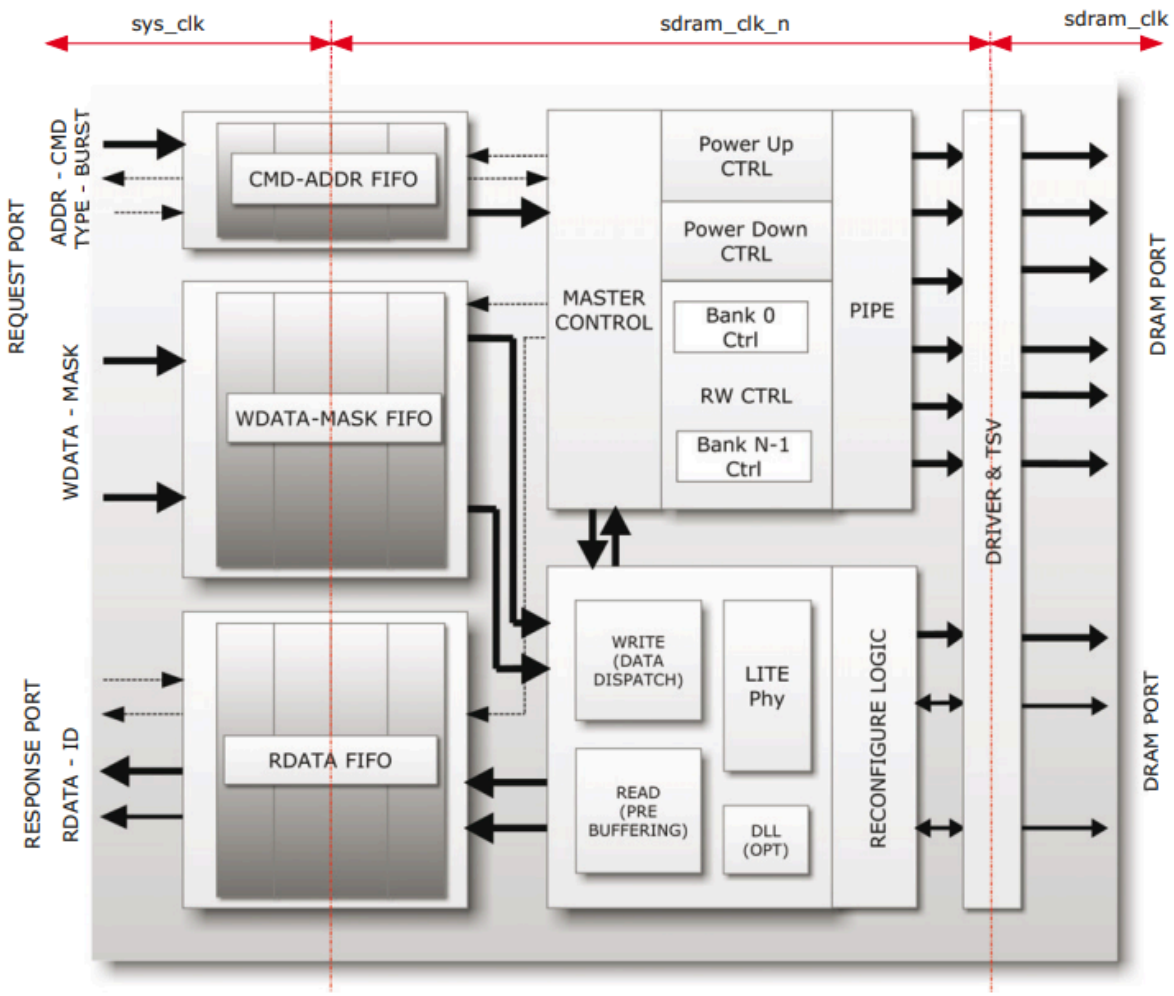
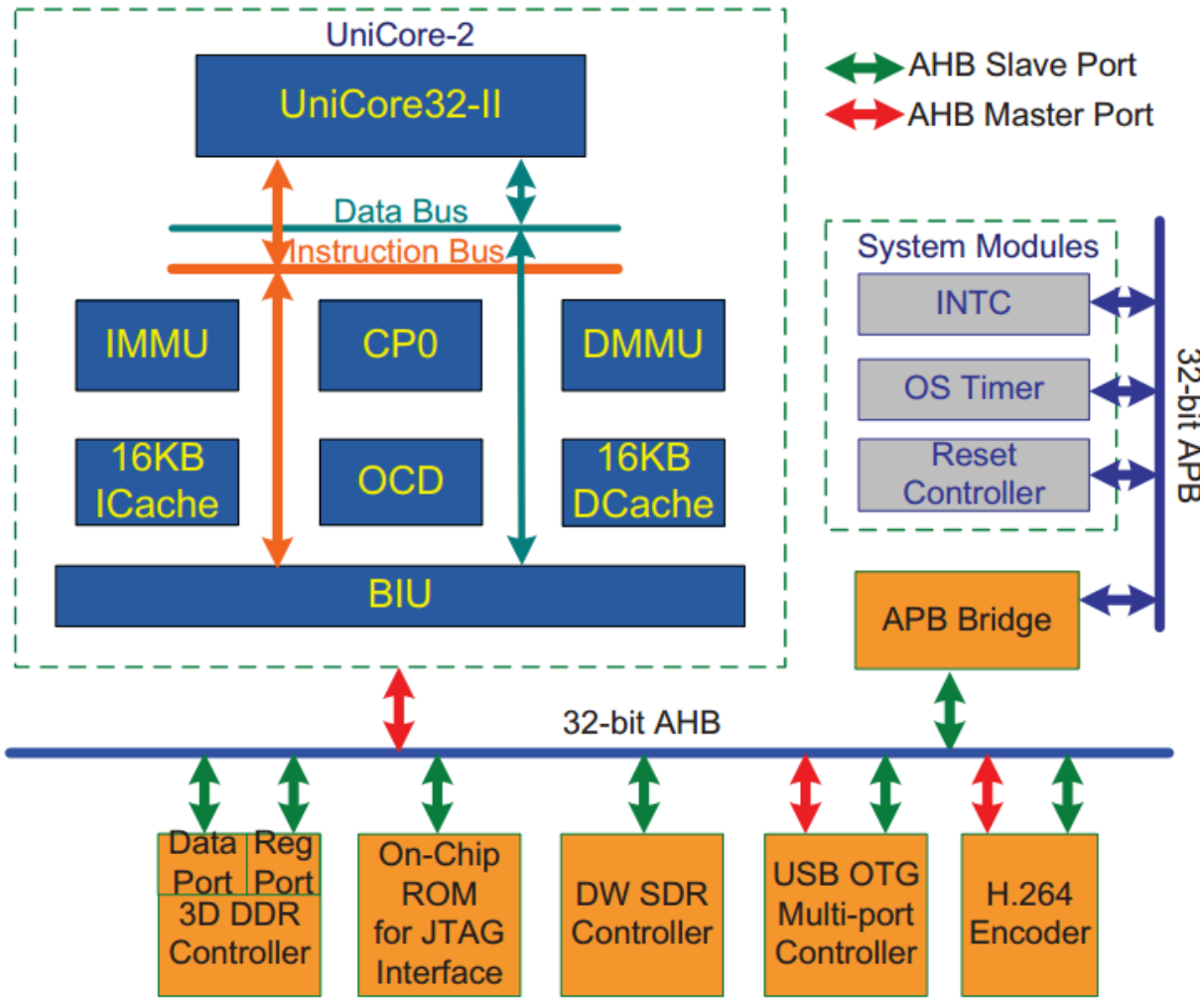


Fig. 2. 3D-DRAM channel controller (CC)

YongLong's 3D Controller


UniCore-2
UniCore32-II
Data Bus
Instruction Bus
IMMU
CP0
DMMU
16KB ICache
OCD
16KB DCache
BIU
AHB Slave Port
AHB Master Port
System Modules
INTC
OS Timer
Reset Controller
32-bit APB
APB Bridge
32-bit AHB
Data Port
Reg Port
3D DDR Controller
On-Chip ROM for JTAG Interface
DW SDR Controller
USB OTG Multi-port Controller
H.264 Encoder

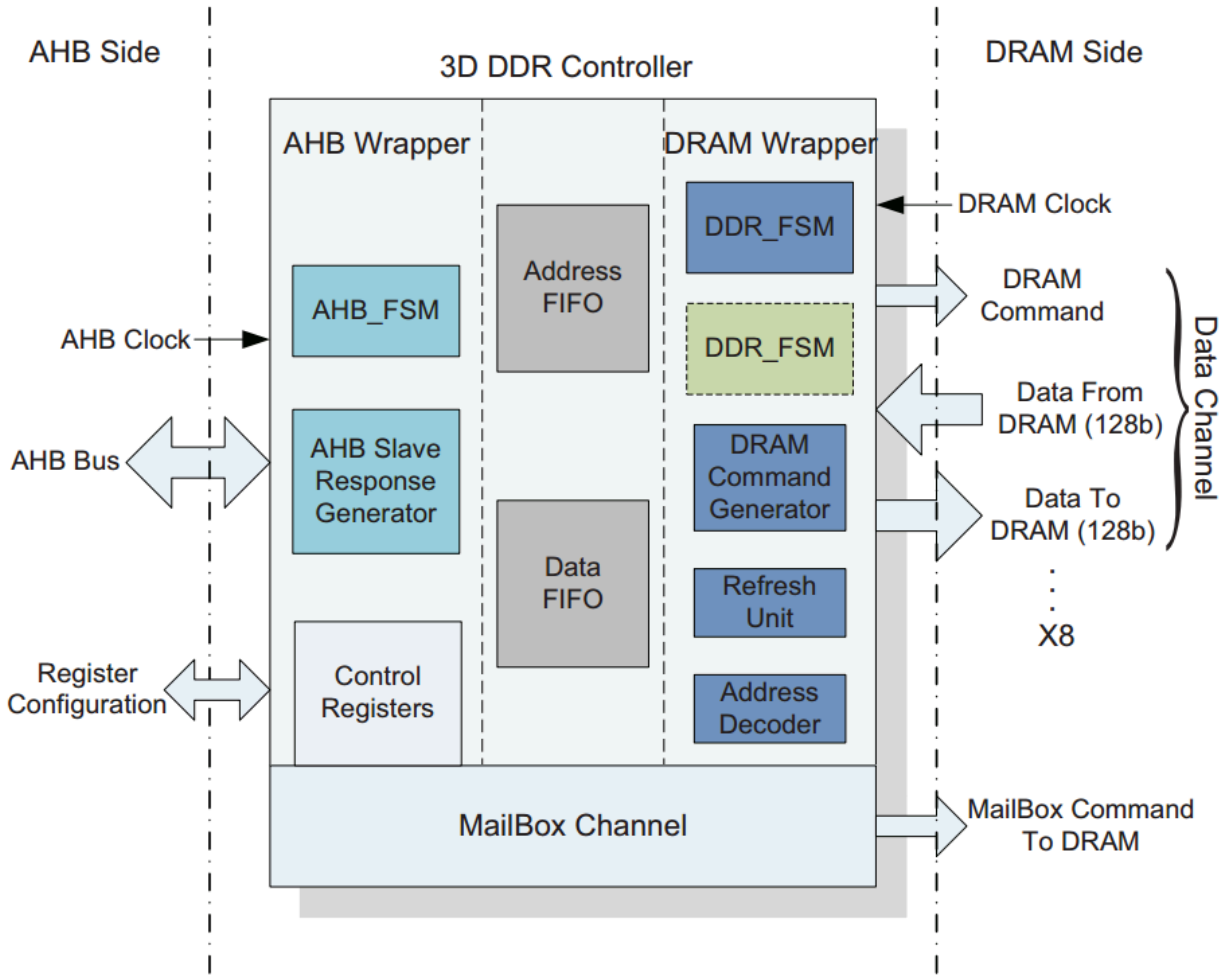


Fig. 2. Block view of 3D DRAM controller

Hwang's Controller

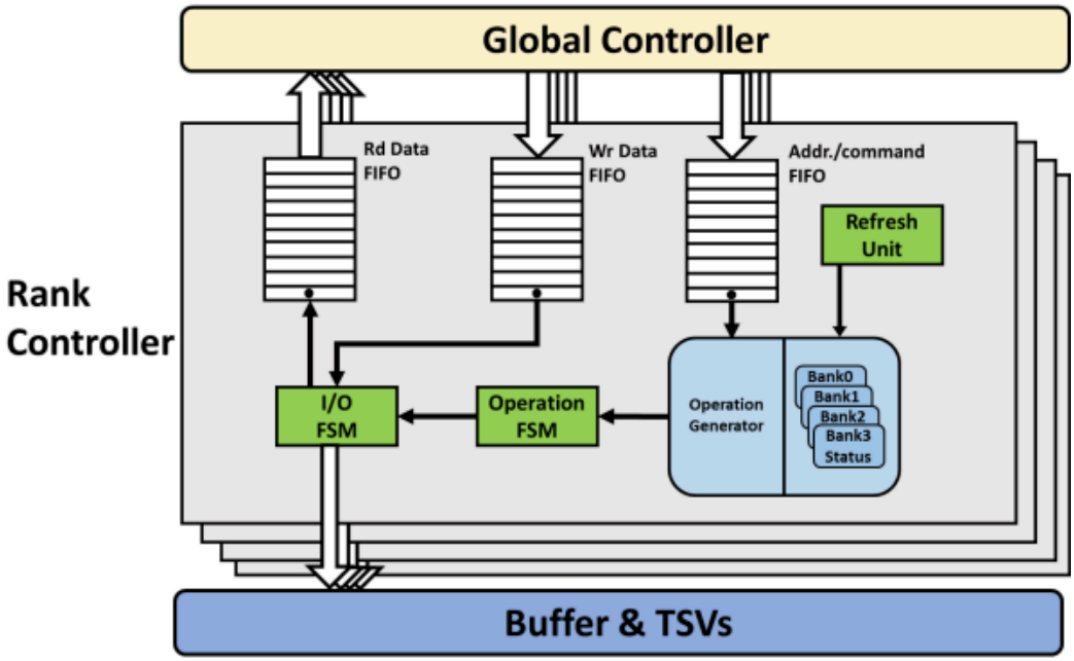


Fig. 4. Architecture of Rank Controller

- Connection to buffer’s and TSVs for DRAM physical connection.

Questions

- What is the interconnecton width and datapath for your interconnections?
- What is the size of IO port for each of your PE? What is the datapath for your RISC-V cores?
- What is the interconnection size needed to connect all your SRAM, PEs and RISC-V core?
- Which kind of interconnection architecture do you plan to use? AXI? Wishbone?Avalon Bus?
- How is the MMIO table, and what is the address space for DRAM?
- Please give me the traces of your cores accessing the DRAM in the LD,ST formats for my performance analysis.