DDR 使用说明书

版本号 DDR_v1.1

2015 级袁亚鹏

注：如有错误，请随时与笔者联系！

一、DDR_v1.1[1]概述

DDR 接口簇按照一定的寻址规律对 DDR 进行读写访问，主要用于按照配置信息指定的方式从 DDR 中读取数据或向 DDR 写入数据。相比于 DDR_v1.0，DDR_v1.1 不控制任务切换[2]！

DDR 中的数据包括以下三类：

1） 配置信息类[3]

对各个簇进行配置所需的信息（只读），它其实是 MC 指令信息的一部分！

2） 指令信息类

系统中有两种独特的数据（对于 DDR 只读）：

A) MC 指令信息
B) COP 运算类指令信息

3） 运算类数据

在任务中参与运算的数据，按照因果关系分为三种：

A) 源数据
B) 中间数据
C) 结果数据

这三种数据对 DDR 来说是没有区别的，但仍有必要清晰地建立这种概念！

表 1 DDR_v1.1 功能模式

<table>
<tr><td colspan="2" rowspan="2">功能模式</td><td colspan="2">三方节点</td><td colspan="2">目的节点</td><td rowspan="2">三方节点和目的节点是否成对</td><td rowspan="2">备注</td></tr>
<tr><td>使用</td><td>数量</td><td>使用</td><td>数量</td></tr>
<tr><td colspan="2">三方</td><td>√</td><td>1</td><td>√</td><td>1</td><td>√</td><td rowspan="3">一个地址通道只能选择一种模式</td></tr>
<tr><td colspan="2">广播</td><td>×</td><td>——</td><td>√</td><td>全部</td><td>——</td></tr>
<tr><td colspan="2">普通</td><td>×</td><td>——</td><td>√</td><td>1</td><td>——</td></tr>
<tr><td rowspan="2">功模式组合</td><td>支持</td><td colspan="5">主要功能：
1、 三方 —— 支持三方功能、不支持广播功能
2、 广播 —— 支持广播功能、不支持三方功能
3、 普通 —— 不支持三方功能、不支持广播功能

辅助功能：
1、 循环功能
2、 一级矩阵行/列优先
3、 读/写</td><td>当三方开启时，循环功能、行列优先功能无效。</td></tr>
<tr><td>不支持</td><td colspan="5">三方广播</td><td></td></tr>
</table>

注：√表示使用　　×表示不使用

[1] 与 DDR_V1.0 相比，DDR_V1.1 的配置信息发生了很大变化！

[2] 在 DDR_v1.1 版本中，任务切换是由主控制器控制的，而非 DDR 簇。

[3] 这里的配置信息包括系统中所有需要配置的簇的配置信息，而非单指 DDR 簇的配置信息！

二、DDR 结构

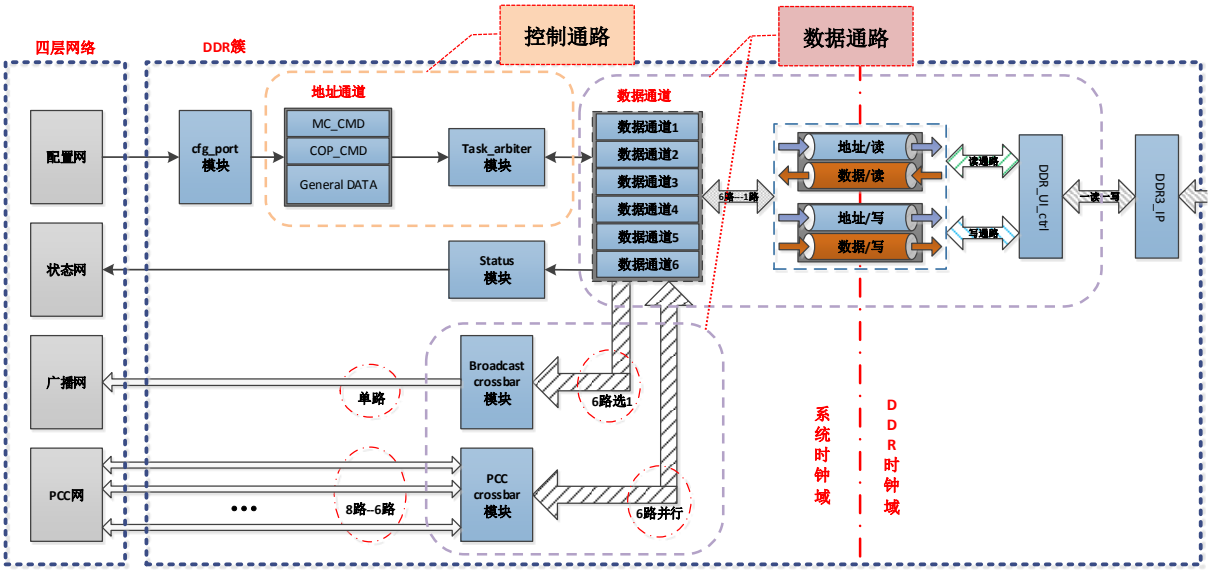


图 1　DDR 簇内部结构示意图

注：附录 1 是图 1 的扩展，它指明了 DDR 在整个系统中所处的位置！

表 2 给出了 DDR 结构中各功能模块的简要功能。

图 2 展示了控制通路的局部结构。

表 2　DDR 模块功能说明

Module	Function	Attention
<Cfg_port>	【功能 1】接收并解析来自配置网的配置信息(CFG info)，对 DDR 通用地址通道(general address channel, ach); 【功能 2】接收并解析来自配置网的数据请求信息(REQ info)，产生请求信息送往 ach;	内含子模块<cfg_pos_ok_v2>, 该子模块仅在<cfg_port>实现【功能 1】时起作用。
<MC_CMD>	系**地址通道**，专为主控制器(Main controller, MC)取指令设定，不接受配置信息的配置。	
<COP_CMD>	系**地址通道**，专为 COP 取指令设定，不接受配置信息的配置。	
<General_DATA>	系**地址通道**，内含 32 个 general ach,专为取数据设定，必须接受配置信息的配置。	
<Task_arbiter>	接收来自地址通道的数据请求，为其请求分配数据通道后将请求的详细信息送至数据通道。	
通道 1~6 <ddr_ch_fsm>	系**数据通道**，共六个 dch，是设计中处理请求的真正场所，也是数据层网络、状态层网络与 DDR Wrapper 发生信息交互的起点！ 【功能 1】在读请求起始产生写应答请求至状态模块，后经状态网至 MC,MC 将写应答请求经配置网广播给写请求簇 【功能 2】产生地址通道释放请求至状态模块，后经状态网发送至 MC 【功能 3】产生读写请求地址送至后续模块<rd_ddr_arbiter>和<wr_ddr_arbiter> 【功能 4】读请求下将数据打包成 PCC 格式或者广播网格式	

	【功能 5】写请求下解析 PCC 包中数据，送至内部数据 FIFO	
<status_port>	接收 dch 发出的状态请求，包括写应答请求和通道释放请求，打包成状态网格式并上传！	
<broadcast_crossbar>	控制 6 个 dch 与广播层网络的数据交互，实现 6 路转 1 路！	同一时刻只能连接至多 1 路 dch,且只会在读请求时发生！
<PCC_crossbar>	控制 6 个 dch 与 PCC 网络的 8 个端口的数据交互！ 【功能 1】读请求下在空闲 op 通道中查询并分配最优通道，实现 6 路 dch 转 8 路 op 【功能 2】写请求下实现 8 路 ip 转 6 路 dch	实际上就是一个大的 crossbar
<rd_ddr_arbiter>	以时间轮片的方式轮流接收 6 路 dch 的读请求地址，将地址送至读地址 FIFO，并将抽取到的读数据正确送至对应的 dch	两个 FIFO 跨时钟域
<wr_ddr_arbiter>	以时间轮片的方式轮流抽取 6 路 dch 的写请求地址和写数据，同步送至写地址 FIFO 和写数据 FIFO	两个 FIFO 跨时钟域
<ddr_UI_ctrl>	操作 DDR-ip controller 用户接口，以时间轮片的方式在读写 DDR 操作之间切换，既是写请求下写数据的终点，又是读请求下读数据的起点	

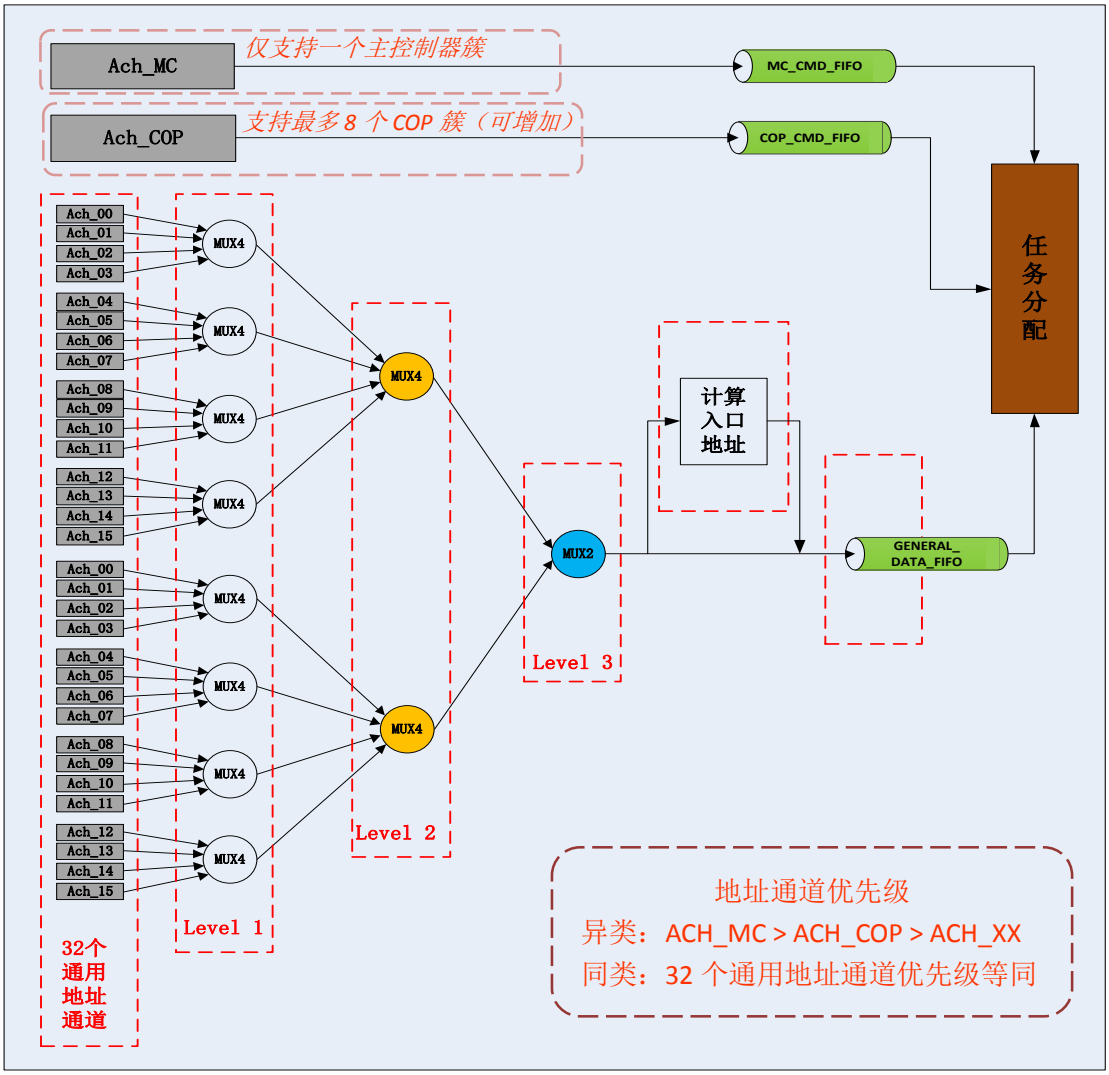


图 2　控制通路局部结构示意图

三、DDR 配置信息

一段 DDR 配置信息流(config_stream)包括 16 条配置字，每条配置字 32-bits。DDR 中包含 32 个通用地址通道(为方便起见，表示成 ach)，每一段 config_stream 只对一个 ach 配置，若想在某次任务(task)中使用多个 ach，则必须多次下发 config_stream。

表 3 给出了 DDR 配置字的组织格式。

表 4 给出了 DDR 配置字的具体含义。

表 3　DDR 配置字组织格式

CFG_0					
31	30	29	28	27	26
third_req	stream	Broadcast	Row/Column	Read/Write	D2D_FLAG
25:23	22:20	19:17	16:8	7:5	4:0
reserved	ch_cluster_num	reserved	Vr_id	VR_FR	ddr_channel_id

CFG_1			
31:24	23:16	15:8	7:0
ch_data_pos3_1st	ch_data_pos2_1st	ch_data_pos1_1st	ch_data_pos0_1st

CFG_2			
31:24	23:16	15:8	7:0
ch_data_pos7_1st	ch_data_pos6_1st	ch_data_pos5_1st	ch_data_pos4_1st

CFG_3			
31:24	23:16	15:8	7:0
ch_addr_pos3_1st	ch_addr_pos2_1st	ch_addr_pos1_1st	ch_addr_pos0_1st

CFG_4			
31:24	23:16	15:8	7:0
ch_addr_pos7_1st	ch_addr_pos6_1st	ch_addr_pos5_1st	ch_addr_pos4_1st

CFG_5	
31:30	29:0
Reserved	ch_start_addr

CFG_6		
31:30	29:16	15:0
Reserved	ch_column_num_1st	ch_row_num_1st

CFG_7		
31:30	29:16	15:0
Reserved	ch_column_num_2nd	ch_row_num_2nd

CFG_8		
31:24	23:21	20:18
Reserved	ch_ data_pos7_2nd	ch_data_pos6_2nd

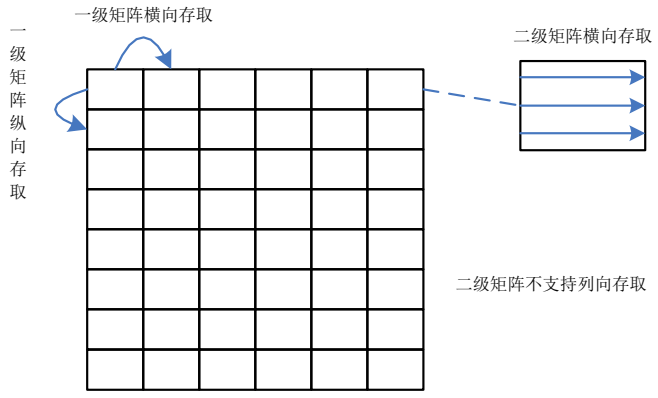
17:15	14:12	11:9
ch_data_pos5_2nd	ch_data_pos4_2nd	ch_data_pos3_2nd
8:6	5:3	2:0
ch_data_pos2_2nd	ch_data_pos1_2nd	ch_data_pos0_2nd
CFG_9		
31:24	23:21	20:18
Reserved	ch_addr_pos7_2nd	ch_addr_pos6_2nd
17:15	14:12	11:9
ch_addr_pos5_2nd	ch_addr_pos4_2nd	ch_addr_pos3_2nd
8:6	5:3	2:0
ch_addr_pos2_2nd	ch_addr_pos1_2nd	ch_addr_pos0_2nd
CFG_10		
31:30	29:0	
Reserved	ch_circu_times	
CFG_11		
31:30	29:0	
Reserved	ch_col_addr_burst	
CFG_12		
31:30	29:0	
Reserved	ch_row_addr_burst	
CFG_13		
31:30	29:0	
Reserved	ch_2nd_addr_burst	
CFG_14		
31:30	29:0	
Reserved	ch_access_length	
CFG_15		
31: 0		
Reserved		

表 4　DDR 配置字含义说明

配置项	位宽	含义	值
third_req	1	是否三方。　0－否　1－是	
stream	1	是否流请求。　0－否　1－是	
Broadcast	1	是否广播。　0－否　1－是	
Row/Column	1	行/列优先。　0－列　1－行	
Read/Write	1	读/写模式。　0－读　1－写	
D2D_FLAG	1	数据从 DDR 到 DDR 使能信号	

ch_cluster_num	3	通道簇数目（0~7）	真实值 减一
Vr_id	9	主控制器虚拟寄存器编号	
ddr_channel_id	5	地址通道编号（0~31）	
ch_data_posx_1st	8	目标节点的一级坐标	
ch_data_posx_2nd	3	目标节点的二级坐标	
ch_addr_posx_1st	8	三方节点的一级坐标	
ch_addr_posx_2nd	3	三方节点的二级坐标	
ch_start_addr	30	起始地址（DDR 的绝对地址）	真实值
ch_column_num_1st	14	一级矩阵列数	真实值 减一
ch_row_num_1st	16	一级矩阵行数	真实值 减一
ch_column_num_2nd	14	二级矩阵列数	真实值/burst_length -1 *现有系统中 burst_length 为 8*
ch_row_num_2nd	16	二级矩阵行数	真实值 减一
ch_circu_times	30	矩阵循环次数（0 表示无循环）	真实值 减一
ch_col_addr_burst	30	一级矩阵列加一地址跳变数目	真实值 二级矩阵列数（真实值）
ch_row_addr_burst	30	一级矩阵行加一地址跳变数目	真实值 下面三个参数的乘积： 参数 1：二级矩阵列数（真实值） 参数 2：一级矩阵列数（真实值） 参数 3：二级矩阵行数（真实值）
ch_2nd_addr_burst	30	二级矩阵行加一地址跳变数目	真实值 下面两个参数的乘积： 参数 1：二级矩阵列数（真实值） 参数 2：一级矩阵列数（真实值）
ch_access_length	30	二级矩阵大小	真实值 假若一次性访问 1K 个 64-bits 数据，则设置为 1024

一级矩阵和二级矩阵的访存模式：



自问自答

问：数据为什么组织成二维矩阵形式呢？

答：在现有系统中，数据最大的特点是粗粒度，即运算簇的对象是“大批量的数据集合”。为了“迎合”粗粒度的需求，“将数据组织成二维矩阵形式”这种概念应运而生，因为它非常有利于数据的管理！

放弃二级矩阵列向存取的原因：

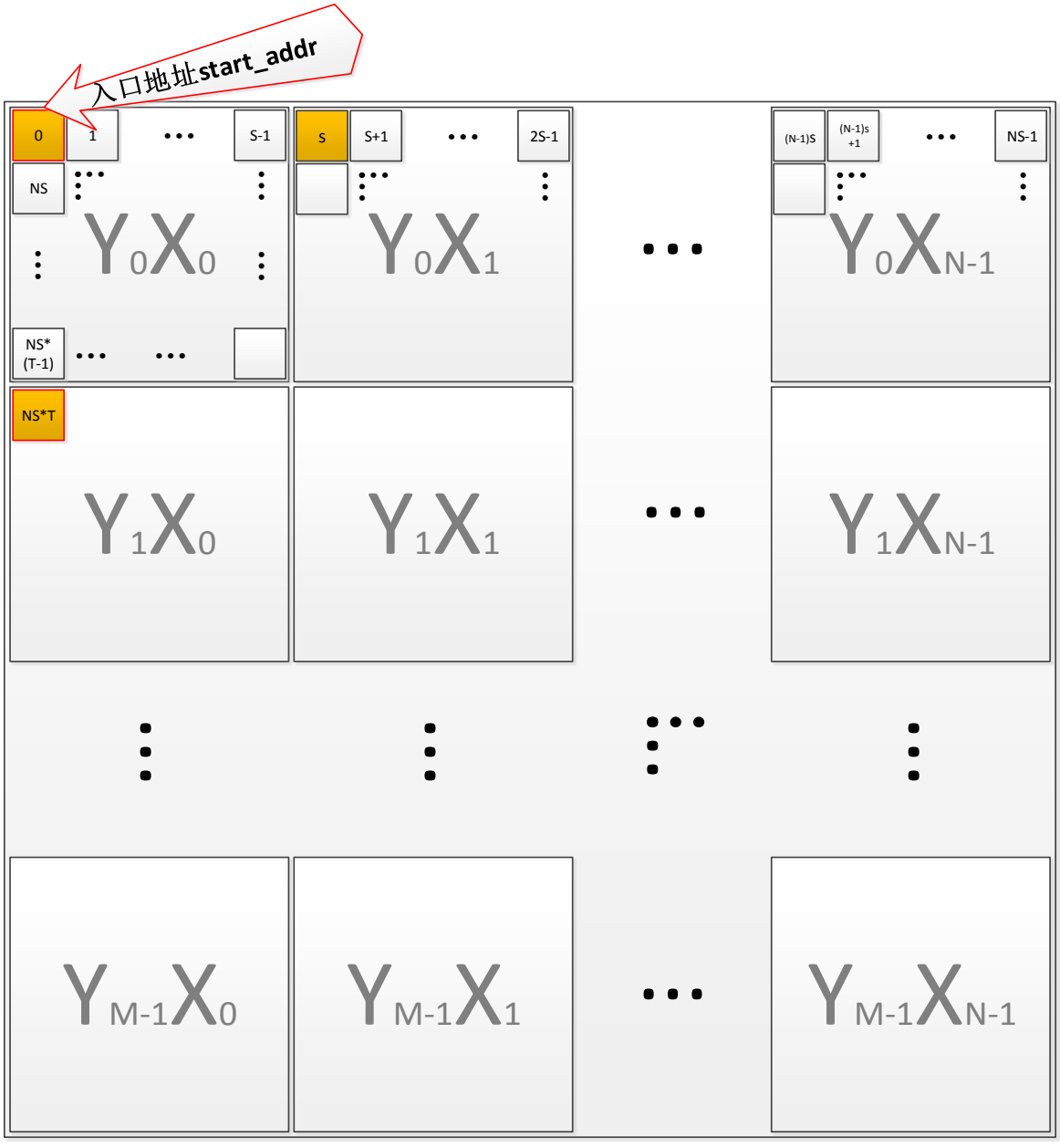
列向存取只会在 DDR_SDRAM 配置为 BURST_LENGTH=1 时（即 DDR_SDRAM 不使用突发模式）起作用。但很不幸的是，这要付出巨大代价：

一方面不使用突发模式时，要想访问 8 个数据，则必须送出 8 次地址，发出 8 次读 DDR_SDRAM 指令（突发模式下访问一个连续的 8 个地址空间数据，只需发布一次读指令，附带首地址），会使得 DDR 的访存带宽大为降低（理论上，突发：不突发 = 4:1 写操作时 / 8:1 读操作时[4]）；

另一方面，列向存取会使得 DDR_SDRAM 频繁地“关闭当前行—激活新行”，大部分时间都花费在“不必要的动作”上，性能大打折扣！

虽然二级矩阵列向存取对矩阵转置类型任务有莫大的吸引力，但显而易见的是，以如此大的代价换取在整个任务中明不明显的好处，得不偿失，实不可取！现有系统中为了挖掘 DDR_SDRAM 带宽，已经默认的将 DDR 中的突发长度设定为 8，并且将来的系统即使更新换代，也不会脱离这一点！

DDR 配置信息中部分参数的定义：



[4] 该数据有待验证！

一级矩阵：M行N列
二级矩阵：T行S列
二级矩阵块个数： TS
一级矩阵块$Y_{i-1}X_{j-1}$：一级矩阵表格的第i行j列, 其入口地址为：
(一级矩阵入口地址) + (i-1)*(一级矩阵行加一地址跳变距离) + (j-1)*(一级矩阵列加一地址跳变距离)

注意图中的三个橙色小块，为了说明的方便，三个小块一次命名为P1/P2/P3,相对于一级矩阵入口地址的偏移量依次为0、S、NS*T，那么有如下定义：
一级矩阵入口地址： P1块在内存空间中的真实地址,图中红色箭头指向的真实地址！
一级矩阵列加一地址跳变距离： P1到P2的距离，即：S-0 = S
一级矩阵行加一地址跳变距离： P1到P3的距离，即：NS*T-0 = NS*T
二级矩阵行加一地址跳变距离： P1到P1块下面的那个小块的距离，即NS-0 = NS

注：“二维矩阵”和“二级矩阵”不是一个概念，“二维矩阵”是“一级矩阵”和“二级矩阵”的嵌套组合，每一个“一级矩阵元素”（一级矩阵块）都对应一个“二级矩阵”。

四、请求信息（REQ）

在现有系统中，一段请求信息流由 3 条请求信息（REQ_0/REQ_1/REQ_2）组成，其中 REQ_0 通用，REQ_1 复用（三种封装格式，其解析方式由 REQ_1[MSB]及 REQ_0 共同决定），REQ_2 通用（Reserved）。

表 5　配置网络请求信息格式

REQ_0 通用			
31	30:22		
End_flag	reserved		
21:20	19	18:16	15:8
Src_type	reserved	Src_pos_2nd	Src_pos_1st
7:0			
Dst_pos_1st			
REQ_1（1）用于 COP 三方取数据			
31	30	29:16	15:0
1'b0	Last_trans	ddr_1st_col_pos	ddr_1st_row_pos
REQ_1（2）用于 MC/COP 取指令			
31	30:20	19:0	
1'b1	reserved	MC_addr/COP_addr	
REQ_1（3）用于普通数据请求			
31	30:0		
1'b0	reserved		
REQ_2			
31:0			
reserved			

注 1：REQ_1 的最高位（MSB）用于区分当前请求是指令类请求还是数据类请求。

注 2：对于 Reserved 信号，一般都默认为低电平（default: GND）。

表 6　REQ_INFO 中的某些关键信号说明

信号	说明	
End_flag	对于 DDR 来说暂时没什么用！ 1'b0　not end 1'b1　end	
Src_type	A) 对于主控制器来说是区分顶层/底层指令的唯一标识 2'b01 src_A　//top 256 2'b10 src_B　//btm 512 B) 对于目的节点来说是区分源数据类型的唯一标识 C) 三方请求中，源类型标识由目的节点请求决定，三方节点仅提供访存块坐标	
Src_pos_2nd	请求节点在网络中的二级坐标	
Src_pos_1st	请求节点在网络中的一级坐标	
Dst_pos_1st	请求包的“目的地”在网络中的一级坐标	
Last_trans	三方请求是否是最后一次请求，高有效！	仅使用在三方请求中 *（暂且只有 COP 充当三方节点）*
ddr_1st_col_pos	三方请求的访存块在一级矩阵表格中的坐标	
ddr_1st_row_pos		

五、DDR 与四层网络接口

DDR 簇与四层网络（配置层网络、状态层网络、PCC 层网络、广播层网络）相连的示意图（参见附录 1）

5.1 配置网络接口

配置网络用于下发配置信息和访存 DDR 请求，配置网络上的信息由主控制器发出。DDR 簇中 cfg_decoder 模块负责接收来自配置网的信息，并按照对应方式进行解析。

1） 判断当前信息是配置信息还是请求信息？

2） 如果是请求信息，则继续判断是数据请求信息还是指令请求信息？

配置信息和请求信息已经叙述过，这里不再阐述，只附上链接！

链接 1：配置信息说明

链接 2：请求信息说明

附录 1：DDR 簇结构及其在系统中的位置

5.2 状态网络接口

状态网络接口是 DDR 向外界传递自身状态信息的唯一途径！

附录 1：DDR 簇结构及其在系统中的位置

在两种情况下，DDR 会上传状态信息：

1） 写反馈

DDR 在响应运算簇 A 发起的写请求时，会返回给簇 A 一个写反馈，然后簇 A 接收到该写反馈信息时，开始通过 PCC 向 DDR 发送数据！

表 7 给出写反馈信息格式

表 8 给出写反馈信息中的某些关键信号的说明

2） 地址通道释放

一旦某个通用地址通道（general address channel, ach）关闭，在该通道对应的最后一次访存操作完成后，DDR 会上传地址通道释放信息，并被主控制器接收！

表 9 给出地址通道释放信息格式

表 10 给出地址通道释放信息中的某些关键信号的说明

注：STATUS_0 最高位[31]是区分写反馈/地址通道释放的唯一标志（1'b0:写反馈/1'b1:地址通道释放）

表 7　DDR 返回给写请求簇的状态信息格式（写反馈）

STATUS_0						
31	30:22	21:20	19	18:16	15:8	7:0
1'b0	reserved	Src_type	1'b0	offset_pos	ddrport_pos	Status_dst_pos
STATUS_1						
31:8				7:0		
reserved				ddr_pcc_port_pos		
STATUS_2						
31:0						
reserved						

表 8　写反馈信息中的某些关键信号说明

信号	说明
Src_type	写请求簇附带的源类型标志
offset_pos	写请求簇附带的二级坐标
Status_dst_pos	写请求簇附带的一级坐标
ddrport_pos	DDR 在配置网中的一级坐标
ddr_pcc_port_pos	DDR 为写请求簇分配的 PCC 端口

表 9　返回给主控制器的状态信息格式（地址通道释放）

STATUS_0				
31	30:17	16:8	7:5	4:0
1'b1	reserved	VR_ID	VR_FR	ddr_channel_ID
STATUS_1				
31:0				
reserved				
STATUS_2				
31:0				
reserved				

表 10　地址通道释放信息中的某些关键信号说明

信号	说明
VR_ID	虚拟寄存器编号（由主控制器决定，DDR 只负责转发）
Cluster_type	功能单元类型（现有 6 种） 编号 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 类型 | VR | RCU | FFT | COP | FR | ETH | Reserved 注：功能单元类型不由 DDR 决定，DDR 只负责转发！
ddr_channel_ID	DDR 地址通道编号（0~31）

问：为什么需要写反馈？

答：回答这个问题，需要先弄清楚写操作过程中发生了什么！

典型写操作流程

step1:簇 A 发起写请求

作　用：告诉 DDR 簇“簇 A 已经准备好，随时可以向 DDR 簇发送数据”。

目的地：DDR 簇

路　径：簇 A→状态层网络→主控制器→配制层网络→DDR 簇

step2:DDR 响应上述写请求，发送写反馈

作　用：告诉簇 A“DDR 簇已经准备好，随时可以接收来自簇 A 的数据”；另外，为簇 A 分配一个 PCC 接口(8×8 尺寸 PCC 有 8 个 port)比如 PA，簇 A 后续发送数据时会将数据通过 PA 口传给 DDR 簇！

目的地：簇 A

路　径：DDR 簇→状态层网络→主控制器→配制层网络→簇 A

step3:簇 A 检测到写反馈，开始打通一条 PCC 链路

路　径：簇 A→（相关 PCC_NODE..→）PA→DDR 簇

####注：至此，进入 PCC 协议，此处不再赘述！####

从典型写操作流程示意图中可以明确看出，写反馈的存在有两方面的意义！

意义一：告诉写请求簇一个信息“DDR 已经准备好，随时可以接收写请求簇的数据”。

意义二：告诉写请求簇一个信息“发送数据时要从哪一个 PCC 口进入 DDR”。

问：为什么需要**地址通道释放**？

答：用来告诉主控制器一个信息“DDR 簇中哪些 ach 处于关闭状态”，主控制器得知 DDR 中的资源哪些被释放，然后会释放对应的寄存器资源、任务资源！

5.3 PCC 网络接口

表 11　PCC 网络协议

起始包	**65:64**	**63:18**	**17:16**	**15:8**	**7:0**
	2'b10	46'b0	Src_type	Local_pos	Status_dst_pos
配置包	**65:64**	**63**	**62:0**		
	2'b01	1'b0	reserved		
数据包	**65:64**	**63:0**			
	2'b00	DATA			
结束包	**65:64**	**63**	**62:0**		
	2'b01	1'b1	reserved		

5.4 广播网络接口

表 12　广播网协议

起始包 1	**65:64**	**63:56**	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
	2'b01	dest_7	dest_6	dest_5	dest_4	dest_3	dest_2	dest_1	dest_0
起始包 2	**65:64**	**15:14**	**13:12**	**11:10**	**9:8**	**7:6**	**5:4**	**3:2**	**1:0**
	2'b10	type_7	type_6	type_5	type_4	type_3	type_2	type_1	type_0
数据包	**65:64**	**63:0**							
	2'b00	data							
结束包	**65:54**	**63:0**							
	2'b11	64'b0							

注 1：dest 表示在一次广播任务中目的节点坐标，type 表示对应坐标的数据类型为源 A/源 B。

注 2：起始包 2 中为了保持与起始包 1 对应，[63:16]位没有显示，这 48 位保留不使用，默认为零。

附录

附录 1

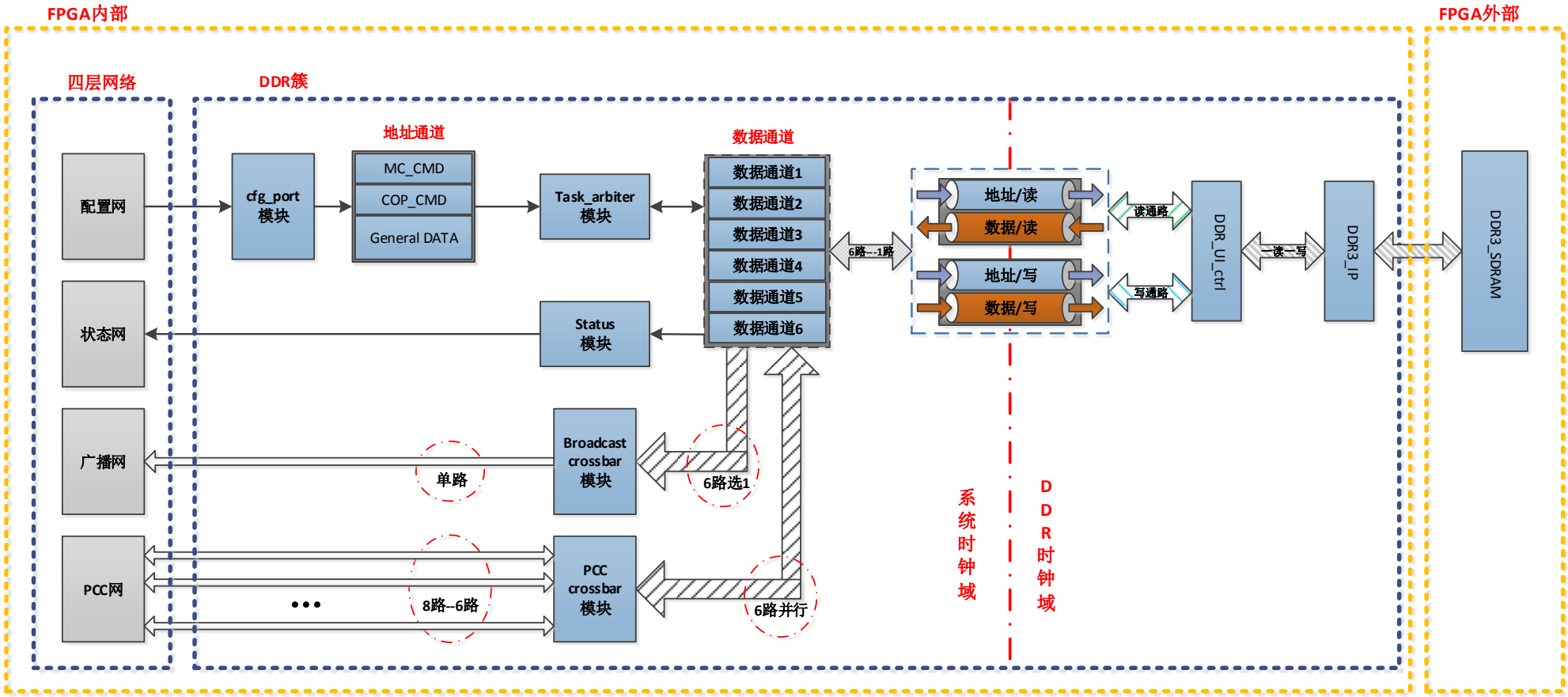


图 DDR簇结构图及其在系统中的位置