**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]](#footnote-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]](#footnote-2)！

DDR中的数据包括以下三类：

1. **配置信息类[[3]](#footnote-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！

1. **指令信息类**

系统中有两种独特的数据（对于DDR只读）：

1. MC指令信息
2. 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：

1. 源数据
2. 中间数据
3. 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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 功能模式 | | 三方节点 | | 目的节点 | | 三方节点和目的节点是否成对 | 备注 |
| 使用 | 数量 | 使用 | 数量 |
| 三方 | | **√** | 1 | **√** | 1 | **√** | 一个地址通道只能选择一种模式 |
| 广播 | | **╳** | —— | **√** | 全部 | —— |
| 普通 | | **╳** | —— | **√** | 1 | —— |
| 功  模  式  组  合 | 支持 | 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 | | | | | 当三方开启时，循环功能、行列优先功能无效。 |
| 不支持 | 三方广播 | | | | |  |

*注：√表示使用 ×表示不使用*

## 二、DDR结构

**控制通路**

**数据通路**



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 |  |



*仅支持一个主控制器簇*

*支持最多8个COP簇（可增加）*

地址通道优先级

异类：ACH\_MC > ACH\_COP > ACH\_XX

同类：32个通用地址通道优先级等同

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：**

**放弃二级矩阵列向存取的原因：**

**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！

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]](#footnote-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：**





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 | 1. 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   1. 对于目的节点来说是区分源数据类型的唯一标识 2. 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](#_三、DDR配置信息)

链接2：[请求信息说明](#_四、请求信息（REQ）)

附录1：[DDR簇结构及其在系统中的位置](#_附录1)

### 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](#_附录1)

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

1. 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



1. 与DDR\_V1.0相比，DDR\_V1.1的配置信息发生了很大变化！ [↑](#footnote-ref-1)
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。 [↑](#footnote-ref-2)
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！ [↑](#footnote-ref-3)
4. 该数据有待验证！ [↑](#footnote-ref-4)