Memory Controller Design Spec

Writer: Kaiqiang Huang

2022.10

contents

[1. Overview 3](#_Toc142857312)

[1.1 Function description 3](#_Toc142857313)

[1.2 Feature list 3](#_Toc142857314)

[1.3 Block diagram 3](#_Toc142857315)

[1.4 Interface description 4](#_Toc142857316)

[2. axi\_slave module 5](#_Toc142857317)

[2.1 Function description 5](#_Toc142857318)

[2.2 Feature list 5](#_Toc142857319)

[2.3 Block diagram 5](#_Toc142857320)

[2.4 Interface description 6](#_Toc142857321)

[2.5 Primary sub module 7](#_Toc142857322)

[2.5.1 w\_channel 7](#_Toc142857323)

[2.5.2 r\_channel 9](#_Toc142857324)

[2.5.3 arbiter 12](#_Toc142857325)

[3. array\_ctrl module 15](#_Toc142857326)

[3.1 Function description 15](#_Toc142857327)

[3.2 Feature list 15](#_Toc142857328)

[3.3 Block diagram 16](#_Toc142857329)

[3.4 Interface description 16](#_Toc142857330)

[3.5 Primary sub module 17](#_Toc142857331)

[3.5.1 array\_state\_ctrl 17](#_Toc142857332)

[3.5.2 array\_write 19](#_Toc142857333)

[3.5.3 array\_read 21](#_Toc142857334)

[3.5.4 array\_refresh 23](#_Toc142857335)

[3.5.5 array\_mux 24](#_Toc142857336)

# Overview

## Function description

memory controller实现了对array read/write/refresh的控制，完成了从axi bus到array interface之间的转换，即将axi协议转换成ddr协议。

## Feature list

* 支持对array接口的时序可配置。
* 支持对array的刷新周期可配置。
* 支持axi bus对array进行跨行访问。
* array的工作频率为200MHz。

## Block diagram



mc\_top包含三个子模块，具体如下：

* axi\_slv：主要实现了axi\_bus到internal frame接口的转换。
* array\_ctrl：主要实现了对array的read/write/refresh控制，以及internal frame接口到array\_interface的转换。
* mc\_apb\_cfg：主要实现了对mc功能寄存器的配置。

## Interface description

|  |  |  |  |
| --- | --- | --- | --- |
| signal name | width | direction | description |
| global signal | | | |
| clk | 1 | input | system clk, 400MHz |
| rst\_n | 1 | input | system reset |
| axi bus | | | |
| axi\_s\_awvalid | 1 | input | axi aw channel valid |
| axi\_s\_awready | 1 | output | axi aw channel ready |
| axi\_s\_awlen | 8 | input | axi aw channel len |
| axi\_s\_awaddr | 25 | input | axi aw channel address |
| axi\_s\_wvalid | 1 | input | axi w channel valid |
| axi\_s\_wready | 1 | output | axi w channel ready |
| axi\_s\_wlast | 1 | input | axi w channel last |
| axi\_s\_wdata | 256 | input | axi w channel data |
| axi\_s\_arvalid | 1 | input | axi ar channel valid |
| axi\_s\_arready | 1 | output | axi ar channel valid |
| axi\_s\_ar\_len | 8 | input | axi ar channel len |
| axi\_s\_ar\_addr | 25 | input | axi ar channel address |
| axi\_s\_rvalid | 1 | output | axi r channel valid |
| axi\_s\_rlast | 1 | output | axi r channel last |
| axi\_s\_rdata | 256 | output | axi r channel data |
| apb bus | | | |
| apb\_pclk | 1 | input | apb clock, 50MHz |
| apb\_prst\_n | 1 | input | apb reset |
| apb\_psel | 1 | input | apb select |
| apb\_pwrite | 1 | input | apb read/write indication |
| apb\_penable | 1 | input | apb enable |
| apb\_paddr | 8 | input | apb address |
| apb\_pwdata | 32 | input | apb write data |
| apb\_pready | 1 | output | apb ready |
| apb\_prdata | 32 | output | apb read data |
| array interface | | | |
| array\_cs\_n | 1 | output | array chip select, low active |
| array\_raddr | 16 | output | array row address |
| array\_caddr\_vld\_wr | 1 | output | array column address valid for write |
| array\_caddr\_wr | 6 | output | array column address for write |
| array\_caddr\_vld\_rd | 1 | output | array column address valid for read |
| array\_caddr\_rd | 6 | output | array column address for read |
| array\_wdata\_vld | 1 | output | array write data valid |
| array\_wdata | 64 | output | array write data |
| array\_rdata\_vld | 1 | input | array read data indication |
| array\_rdata | 64 | input | array read data valid |

# axi\_slave module

## Function description

axi\_slv模块实现了axi bus向internal frame接口之间的转换，通过internal frame接口与array\_ctrl模块进行通信。

## Feature list

* 支持axi bus对array进行跨行访问。
* 支持axi读、写优先级的配置。

## Block diagram



axi\_slave模块包含三个子模块：

* w\_channel：将aw channel中的地址、控制信息与w channel中的数据信息封装成w\_frame发送到arbiter模块。
* r\_channel：将ar channel中的地址、控制信息封装成r\_frame发送到arbiter模块，并接收从array中读回的数据array\_rdata和控制信号array\_rdata\_valid到r channel。
* arbiter：对w\_channel模块与r\_channel模块发出的w/r\_frame进行仲裁，并将仲裁所得的frame发送到array\_ctrl模块。

## Interface description

|  |  |  |  |
| --- | --- | --- | --- |
| signal name | width | direction | description |
| global | | | |
| clk | 1 | input | system clk, 400MHz |
| rst\_n | 1 | input | system reset |
| axi\_s\_aw | | | |
| axi\_s\_awvalid | 1 | input | axi aw channel valid |
| axi\_s\_awready | 1 | output | axi aw channel ready |
| axi\_s\_awlen | 8 | input | axi aw channel len |
| axi\_s\_awaddr | 25 | input | axi aw channel address |
| axi\_s\_w | | | |
| axi\_s\_wvalid | 1 | input | axi w channel valid |
| axi\_s\_wready | 1 | output | axi w channel ready |
| axi\_s\_wlast | 1 | input | axi w channel last |
| axi\_s\_wdata | 256 | input | axi w channel data |
| axi\_s\_ar | | | |
| axi\_s\_arvalid | 1 | input | axi ar channel valid |
| axi\_s\_arready | 1 | output | axi ar channel valid |
| axi\_s\_arlen | 8 | input | axi ar channel len |
| axi\_s\_araddr | 25 | input | axi ar channel address |
| axi\_s\_r | | | |
| axi\_s\_rvalid | 1 | output | axi r channel valid |
| axi\_s\_rlast | 1 | output | axi r channel last |
| axi\_s\_rdata | 256 | output | axi r channel data |
| apb\_cfg | | | |
| mc\_en | 1 | input | apb对axi\_slave的配置完成指示信号，单bit信号打两拍实现跨时钟域同步 |
| axi\_rw\_prio | 2 | input | axi read/write priority  00：read priority  01：write priority  10：round robin  11：reserved |
| internal\_frame | | | |
| axi2array\_frame\_valid | 1 | output | indicate frame valid |
| axi2array\_frame\_ready | 1 | input | indicates array\_ctrl module ready to receive frame |
| axi2array\_frame\_data | 89 | output | [5:0] column address  [21:6] row address  [85:22] write data  [86] read/write flag, 0: read, 1: write  [87] sof, start of frame  [88] eof, end of frame |
| array\_r | | | |
| array\_rdata\_vld | 1 | input | array read data valid |
| array\_rdata | 64 | input | array read data |

## Primary sub module

### w\_channel

#### Function description

将aw channel中的地址、控制信息与w channel中的数据信息封装成w\_frame发送到arbiter模块。

#### block diagram



#### Interface description

|  |  |  |  |
| --- | --- | --- | --- |
| signal name | width | direction | description |
| global | | | |
| clk | 1 | input | system clk, 400MHz |
| rst\_n | 1 | input | system reset |
| axi\_s\_aw | | | |
| axi\_s\_awvalid | 1 | input | axi aw channel valid |
| axi\_s\_awready | 1 | output | axi aw channel ready |
| axi\_s\_awlen | 8 | input | axi aw channel len |
| axi\_s\_awaddr | 25 | input | axi aw channel address |
| axi\_s\_w | | | |
| axi\_s\_wvalid | 1 | input | axi w channel valid |
| axi\_s\_wready | 1 | output | axi w channel ready |
| axi\_s\_wlast | 1 | input | axi w channel last |
| axi\_s\_wdata | 256 | input | axi w channel data |
| w\_frame | | | |
| axi2arb\_wframe\_valid | 1 | output | indicate frame valid |
| axi2arb\_wframe\_ready | 1 | input | indicates array\_ctrl module ready to receive frame |
| axi2arb\_wframe\_data | 97 | output | [5:0] column address  [21:6] row address  [85:22] write data  [86] read/write flag, 0: read, 1: write  [87] sof, start of frame  [88] eof, end of frame  [96:89] awlen |

#### FSM description



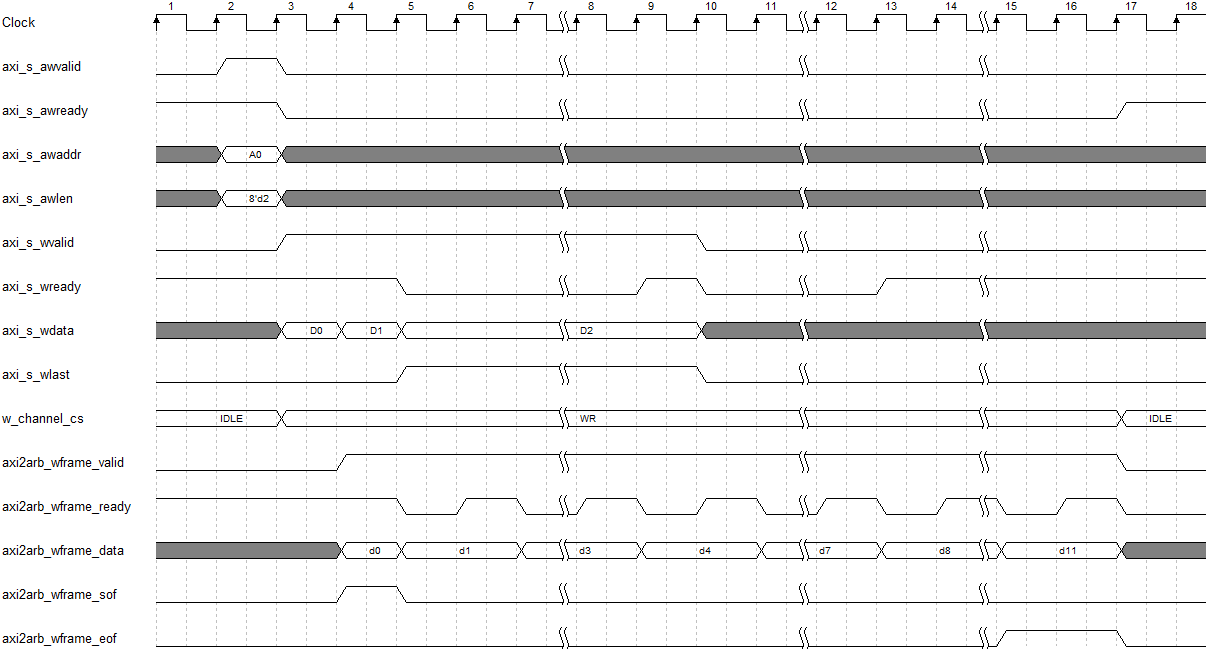
**IDLE：**

1. FSM默认处于IDLE状态，IDLE状态下axi\_s\_awready为1。
2. 当接收到aw通道的axi\_s\_awvalid后（将axi\_s\_awvalid视为写请求），wr\_start == 1， wr\_start = axi\_s\_awvalid && axi\_s\_awready，即握手成功，跳转到WR状态接收写地址axi\_s\_awaddr（首地址）和写数据axi\_s\_awdata。
3. 未接收到aw通道的axi\_s\_awvalid时，wr\_start == 0，保持在IDLE状态。

**WR：**

1. WR状态下axi\_s\_awready为0，反压上级模块对aw通道的操作。
2. 接收写地址axi\_s\_awaddr（首地址）并将其保存到22-bit寄存器awaddr\_reg中。
3. 接收axi\_s\_awlen并保存到8-bit寄存器awlen\_reg中。
4. 接收写数据axi\_s\_wdata并将其保存到256to64\_fifo中。通过256to64\_fifo将一个256-bit的写数据拆分成4个64-bit的写数据。
   1. FIFO位宽为256-bit，深度为2。
   2. FIFO满时axi\_s\_wready为0，反压上级模块。
   3. FIFO非空时，axi2arb\_wframe\_valid为1，发送w\_frame，即axi2arb\_wframe\_valid = ((fsm\_cs == WR) && ~fifo\_empty)。
   4. FIFO的读使能rd\_en == (axi2arb\_wframe\_ready & ~fifo\_empty)，FIFO的写使能wr\_en == (axi\_s\_wvalid & axi\_s\_wready)。
5. 当w\_channel与下级模块arbiter握手成功时，通过“首地址 + 计数器wframe\_cnt”，自增出当前写数据对应的写地址，并将该写地址装进w\_frame中。
6. 通过axi\_s\_awlen确定出该次burst封装成w\_frame后的sof和eof，再根据地址确定出该次burst对array的访问中需要换行的位置，并通过sof和eof标识出，对frame进行切分，得到包含跨行信息的w\_frame。sof和eof的产生逻辑如下：
   1. 进入WR状态时，计数器wframe\_cnt为0，第一个w\_frame准备好，产生第一个sof；
   2. 当首地址的列地址 + wframe\_cnt == 63时，说明即将对array进行跨行访问，产生eof；
   3. 当首地址的行地址变化时，说明对array的访问进入新的一行，产生sof；
   4. 当wframe\_cnt == (axi\_s\_awlen + 1)\*4 – 1时，说明最后一个wframe准备好，产生最后一个eof。
7. 当最后一个写数据封装成wframe且与下级模块arbiter握手成功后，wr\_finish == 1，wr\_finish = (wframe\_cnt == (axi\_s\_awlen + 1)\*4 – 1)) && (axi2array\_wframe\_valid && array2axi\_wframe\_ready)，跳回IDLE状态等待下次burst请求。

#### Timing



### r\_channel

#### Function description

将ar channel中的地址、控制信息封装成r\_frame发送到arbiter模块，并接收从array中读回的数据array\_rdata和控制信号array\_rdata\_valid到r channel。

#### block diagram



#### Interface description

|  |  |  |  |
| --- | --- | --- | --- |
| signal name | width | direction | description |
| global | | | |
| clk | 1 | input | system clk, 400MHz |
| rst\_n | 1 | input | system reset |
| axi\_s\_ar | | | |
| axi\_s\_arvalid | 1 | input | axi aw channel valid |
| axi\_s\_arready | 1 | output | axi aw channel ready |
| axi\_s\_arlen | 8 | input | axi aw channel len |
| axi\_s\_araddr | 25 | input | axi aw channel address |
| axi\_s\_r | | | |
| axi\_s\_rvalid | 1 | output | axi w channel valid |
| axi\_s\_rlast | 1 | output | axi w channel indication of the last rdata |
| axi\_s\_rdata | 256 | output | axi w channel the last rdata |
| r\_frame | | | |
| axi2arb\_rframe\_valid | 1 | output | indicate frame valid |
| axi2arb\_rframe\_ready | 1 | input | indicates array\_ctrl module ready to receive frame |
| axi2arb\_rframe\_data | 97 | output | [5:0] column address  [21:6] row address  [85:22] write data  [86] read/write flag, 0: read, 1: write  [87] sof, start of frame  [88] eof, end of frame  [96:89] arlen |
| array\_r | | | |
| array\_rdata\_valid | 1 | input | array read data valid |
| array\_rdata | 64 | input | array read data |

#### FSM description



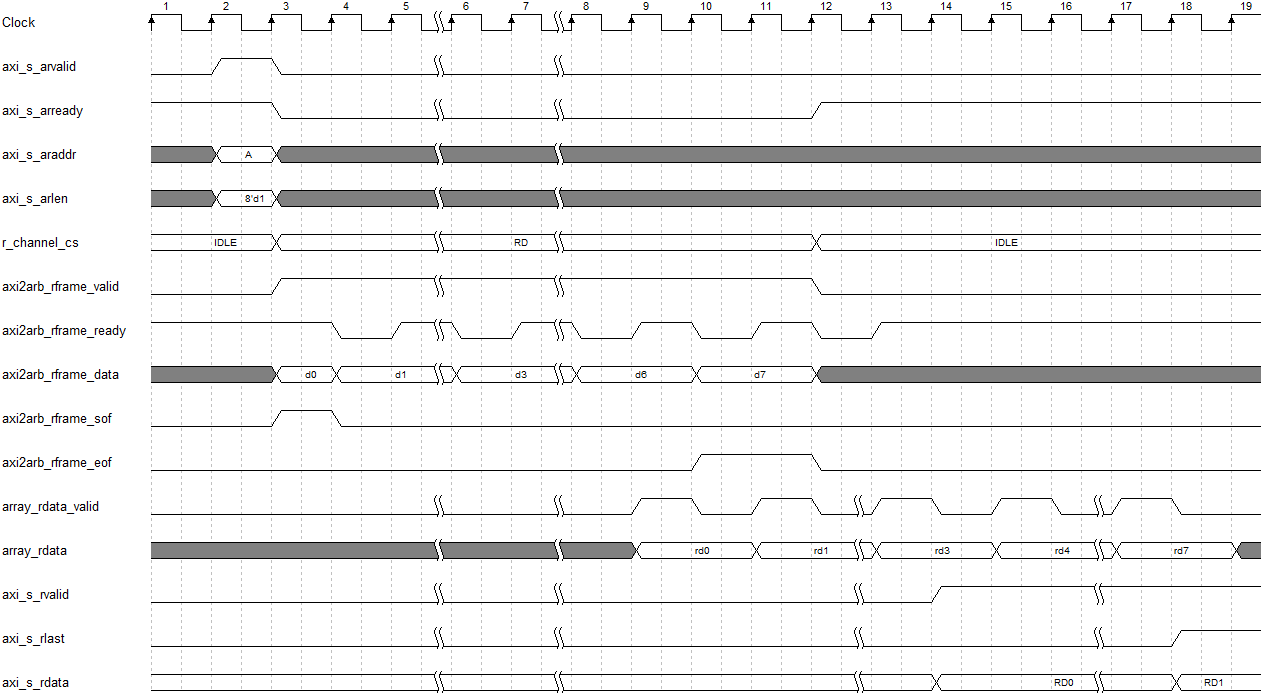
**IDLE：**

1. FSM默认处于IDLE转态，IDLE状态下axi\_s\_arready为1。
2. 当接收到ar通道的axi\_s\_arvalid后（将axi\_s\_arvalid视为读请求），rd\_start == 1， rd\_start = axi\_s\_arvalid && axi\_s\_arready，即握手成功，跳转到RD状态接收读地址axi\_s\_araddr（首地址）。
3. 未接收到aw通道的axi\_s\_arvalid，rd\_start == 0，保持在IDLE状态。

**RD：**

1. RD状态下axi\_s\_arready为0，反压上级模块对r通道的操作。
2. 接收读地址axi\_s\_araddr（首地址）并将其保存到22-bit寄存器araddr\_reg中。
3. 接收axi\_s\_arlen并保存到awlen\_fifo中。
   1. 需要将axi\_s\_arlen存到arlen\_fifo中，通过axi\_s\_arlen计算产生axi\_s\_rlast，这样才能知道从array读回的多少个array\_rdata是属于同一次burst的，完成一次读burst后，将对应的axi\_s\_arlen从arlen\_fifo中弹出。且等待array\_rdata的返回是需要时间的，若此时有新的burst到来，则其对应的axi\_s\_arlen也将被保存到FIFO中。
   2. FIFO位宽为8-bit，深度为2。
   3. FIFO满时axi\_s\_arready为0，也会反压上级模块，即axi\_s\_arready = ((FSM\_cs==IDLE) && ~arlen\_fifo\_full)。
   4. FIFO的读使能，计数器 == (axi\_s\_arlen + 1)\*4 – 1)时，读FIFO，引起读指针变化。
4. 当r\_channel与下级模块arbiter握手成功时，通过“首地址 + 计数器rframe\_cnt”，自增出当前rframe对应的读地址，并封装进rframe中。64-bit的data为reserved。
5. 通过64to256\_reg将4个64-bit的读数据转换为1个256-bit的读数据axi\_s\_rdata，并发送到axi总线上。
   1. array返回的读数据，4个64-bit组成1个256-bit，需要4个时钟周期接收，则在第4个时钟周期接收到后，将拼好的256-bit数据发送出去，因为r\_channel是没有ready的，所以数据不会被反压。若r\_channel是有ready的，在array\_read模块中，每次握手成功意味着要给array发送一个读cmd，需要在array\_read模块中有一个位宽为1-bit的同步FIFO，每次握手成功时，FIFO的指针加1，每次有数据读回时，FIFO的指针减1。array返回到array\_ctrl模块中异步FIFO的读数据，会经过tRD的延迟，为3个时钟周期左右，之前发出的读地址在这段时间按内会依次返回读数据，故该FIFO是用来监控是否还有数据在路上没返回的。FIFO满时，反压上级模块，不再接收rframe，不再继续发送读cmd，故同时也需要看64转256FIFO是否满，只有在上述两个FIFO都没满的时候才能接收rframe发读cmd。
6. 当r通道最后一个rframe封装完成，且与下级模块arbiter握手成功时，rd\_finish == 1，rd\_finish=(rframe\_cnt==(len+1)\*4 – 1) && (axi2arb\_rframe\_valid & axi2arb\_rframe\_ready)，跳转回IDLE状态，等待下次burst请求。

#### Timing



### arbiter

#### Function description

对w\_channel模块与r\_channel模块发出的w/r\_frame进行仲裁，并将仲裁所得的frame发送到array\_ctrl模块。

#### Interface description

|  |  |  |  |
| --- | --- | --- | --- |
| signal name | width | direction | description |
| global | | | |
| clk | 1 | input | system clk, 400MHz |
| rst\_n | 1 | input | system reset |
| w\_frame | | | |
| axi2arb\_wframe\_valid | 1 | input | indicate frame valid |
| axi2arb\_wframe\_ready | 1 | output | indicates array\_ctrl module ready to receive frame |
| axi2arb\_wframe\_data | 97 | input | [5:0] column address  [21:6] row address  [85:22] write data  [86] read/write flag, 0: read, 1: write  [87] sof, start of frame  [88] eof, end of frame  [96:89] awlen |
| r\_frame | | | |
| axi2arb\_rframe\_valid | 1 | input | indicate frame valid |
| axi2arb\_rframe\_ready | 1 | output | indicates array\_ctrl module ready to receive frame |
| axi2arb\_rframe\_data | 97 | input | [5:0] column address  [21:6] row address  [85:22] write data  [86] read/write flag, 0: read, 1: write  [87] sof, start of frame  [88] eof, end of frame  [96:89] arlen |
| internal\_frame | | | |
| axi2array\_frame\_valid | 1 | output | indicate frame valid |
| axi2array\_frame\_ready | 1 | input | indicates array\_ctrl module ready to receive frame |
| axi2array\_frame\_data | 89 | output | [15:0] row address  [21:16] column address  [85:22] write data  [86] read/write flag, 0: read, 1: write  [87] sof, start of frame  [88] eof, end of frame |
| axi\_rw\_prio | | | |
| axi\_rw\_prio | 2 | input | axi bus read and write priority |

#### FSM description



**IDLE：**

1. FSM默认处于IDLE转态，IDLE状态下axi2arb\_wframe\_ready与axi2arb\_rframe\_ready为0，反压上级模块，根据状态跳转来选择哪个ready可被置1。即IDLE状态下是要做仲裁的，是不能接收数据的，故ready为0，需要根据状态机来分发ready。
2. 当接收到axi2arb\_wframe\_valid或axi2arb\_rframe\_valid后（分别视为wframe请求和rframe请求），arb\_start == 1，arb\_start == axi2arb\_wframe\_valid || axi2arb\_rframe\_valid，根据请求进行状态跳转：
   1. axi2arb\_wframe\_valid == 1，axi2arb\_rframe\_valid == 0，跳转到WR状态。
   2. axi2arb\_wframe\_valid == 0，axi2arb\_rframe\_valid == 1，跳转到RD状态。
   3. axi2arb\_wframe\_valid == 1，axi2arb\_rframe\_valid == 1，根据当前的读写优先级仲裁出将要跳转到的状态。
3. 未接收到axi2arb\_wframe\_valid或axi2arb\_rframe\_valid，arb\_start == 0，保持在IDLE状态。

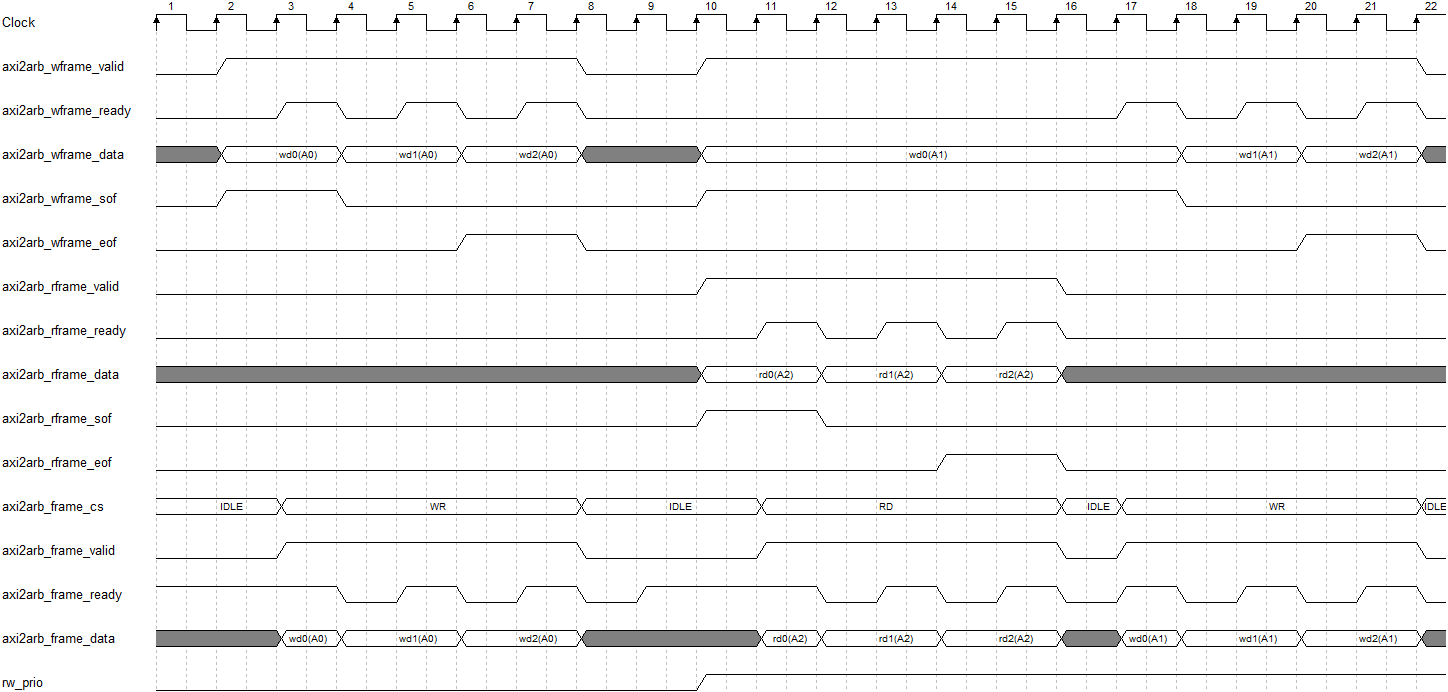
**WR：**

1. 当一次burst中的最后一个wframe与下级模块握手成功时，wr\_finish == 1，wr\_finish = axi2array\_wframe\_valid && axi2array\_wframe\_ready && w\_eof && (wframe\_cnt == (axi\_s\_awlen+1)\*4-1)，跳转回IDLE状态。否则wr\_finish == 0，保持在WR状态。
2. wlen信息要随录到arbiter中，通过wframe\_cnt计算发送的wframe个数，计数到wlen时，说明一次burst的wframe已经发送完，发送到下级模块的wframe信号中不再包含wlen。
3. 处于WR状态时，axi2arb\_frame\_ready穿透给axi2arb\_wframe\_ready，即axi2arb\_wframe\_ready = (arb\_cs == WR) && axi2array\_frame\_ready

**RD：**

1. 当一次burst中的最后一个rframe与下级模块握手成功时，rd\_finish == 1，rd\_finish = axi2array\_rframe\_valid && axi2array\_rframe\_ready && r\_eof && (rframe\_cnt == (axi\_s\_awlen+1)\*4-1)，跳转回IDLE状态。否则rd\_finish == 0，保持在RD状态。
2. rlen信息要随录到arbiter中，通过rframe\_cnt计算发送的rframe个数，计数到rlen时，说明一次burst的rframe已经发送完，发送到下级模块的wframe信号中不再包含wlen。
3. 处于RD状态时，axi2arb\_frame\_ready穿透给axi2arb\_rframe\_ready，即axi2arb\_rframe\_ready = (arb\_cs == RD) && axi2array\_frame\_ready。

#### Timing



# array\_ctrl module

## Function description

array\_ctrl模块实现了对array的read/write/refresh操作，其接口完成了从internal frame到array interface的转换。

## Feature list

* 支持array接口时序可配置。
* 支持array刷新周期可配置。

## Block diagram



## Interface description

|  |  |  |  |
| --- | --- | --- | --- |
| signal name | width | direction | description |
| global | | | |
| clk | 1 | input | system clk, 400MHz |
| rst\_n | 1 | input | system reset |
| internal\_frame | | | |
| axi2array\_frame\_valid | 1 | input | indicate frame valid |
| axi2array\_frame\_ready | 1 | output | indicates array\_ctrl module ready to receive frame |
| axi2array\_frame\_data | 89 | input | [5:0] column address  [21:6] row address  [85:22] write data  [86] read/write flag, 0: read, 1: write  [87] sof, start of frame  [88] eof, end of frame |
| array\_interface | | | |
| array\_cs\_n | 1 | output | array chip select, low active |
| array\_raddr | 16 | output | array row address |
| array\_caddr\_vld\_wr | 1 | output | array column address valid for write |
| array\_caddr\_wr | 6 | output | array column address for write |
| array\_wdata\_vld | 1 | output | array write data vld |
| array\_wdata | 64 | output | array write data |
| array\_caddr\_vld\_rd | 1 | output | array column address valid for read |
| array\_caddr\_rd | 6 | output | array column address for read |
| array\_r | | | |
| array\_rdata\_vld | 1 | input | array read data valid |
| array\_rdata | 64 | input | array read data |
| sync\_array\_r | | | |
| sync\_array\_rdata\_vld | 1 | output | array read data valid |
| sync\_array\_rdata | 64 | output | array read data |
| apb\_cfg | | | |
| mc\_en | 1 | input | mc enable |
| array\_tRAS | 8 | input | the minimum duration of array\_cs\_n being in a low-level signal |
| array\_tRP | 8 | input | precharge time |
| array\_tRC | 8 | input | the time between two falling edges of array\_cs\_n |
| array\_tRCD\_WR | 8 | input | write RCD time |
| array\_Trcd\_RD | 8 | input | read RCD time |
| array\_tWR | 8 | input | write recover time |
| array\_tRTP | 8 | input | read to precharge time |
| array\_rf\_period\_sel | 1 | input | array refresh period select:  0: array\_rf\_period\_0  1: array\_rf\_period\_1 |
| array\_rf\_period\_0 | 25 | input | array refresh period 0(60ms) |
| array\_rf\_period\_1 | 25 | input | array refresh period 1(50ms) |

## Primary sub module

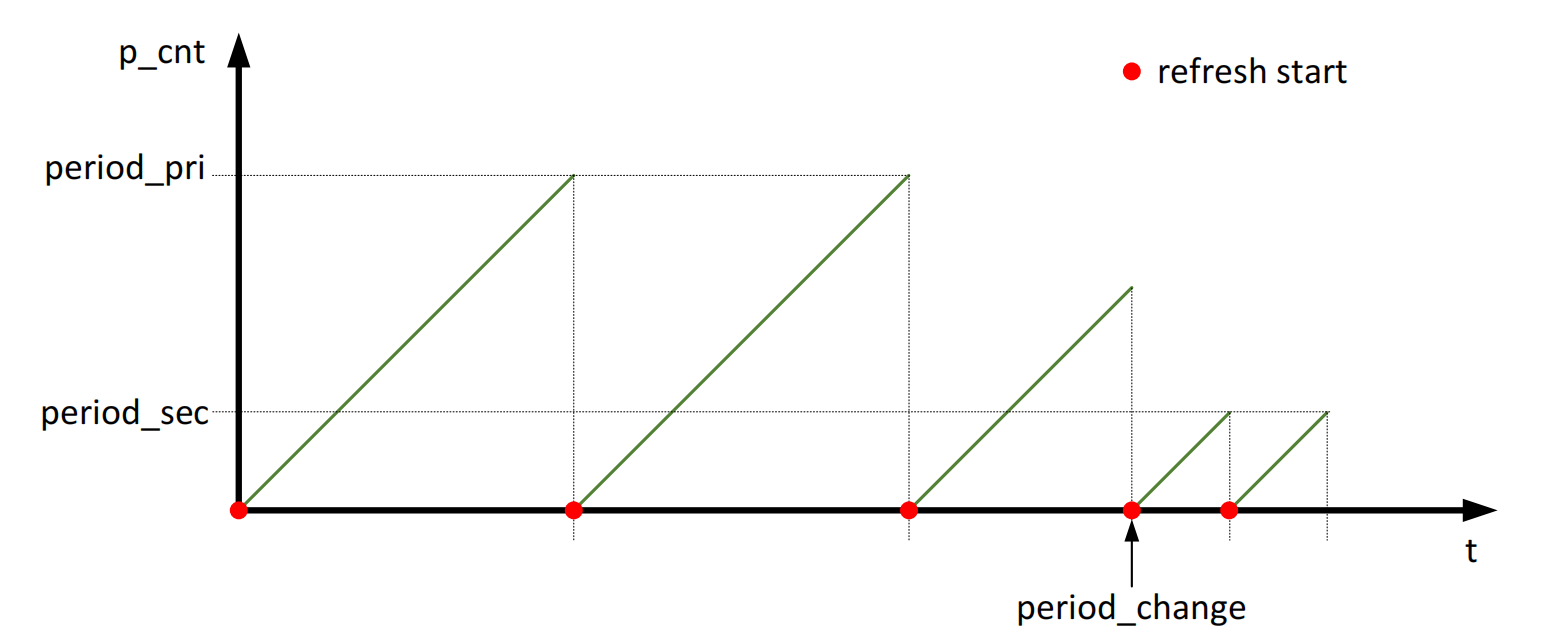
### array\_state\_ctrl

#### Function description

根据axi\_slv模块发送的frame中的读写标志位来控制array\_ctrl模块的RD、WR状态跳转，将frame分发到对应的模块中。

内置刷新周期计数器，控制RF状态的跳转，内置的刷新周期计数器计数到目标值时，计数器清零，并发起刷新请求，若此时未处于IDLE状态，则将该请求保存下来，等到FSM跳回IDLE状态时，立即执行刷新请求，故刷新周期的设置是有一定余量的。

刷新周期可配置，可在mc\_apb\_cfg模块中通过准静态信号同步的方式修改刷新周期。



根据FSM的当前状态，向array\_mux模块发送sel信号。

#### FSM description



**IDLE：**FSM默认处于IDLE状态，ready为0。当array\_ctrl\_start == 1时，根据请求跳转到对应的RF/WR/RD状态，故需要占用IDLE状态的一个时钟周期。array\_ctrl\_start == (rf\_start || wr\_start || rd\_start)。当array\_ctrl\_start == 0时，保持在IDLE状态。

其中，各个请求的产生条件如下所示：

1. rf\_start == (trp\_cnt >= array\_rf\_period)，即当刷新周期计数器大于当前所选的刷新周期时，刷新请求到来，且优先级最高。
2. wr\_start == (axi2array\_frame\_valid && axi2array\_frame\_data[86] && [87])，其中[86]为读写flag指示，[87]为sof。
3. rd\_start == (axi2array\_frame\_valid && !axi2array\_frame\_data[86] && [87])，其中[86]为读写flag指示，[87]为sof。

**RF：**对array发起刷新操作，并将刷新计数器清零。每刷新完一行，次数计数器加1，直至加到总行数，执行完最后一次刷新时，即rf\_finish == 1时，跳回IDLE状态。

**WR：**对array发起写操作，precharge中的最后一拍，即wr\_finish == 1时，跳回IDLE状态。

**RD：**对array发起读操作，precharge中的最后一拍，即rd\_finish == 1时，跳回IDLE状态。

### array\_write

#### Function description

将array\_state\_ctrl模块分发给array\_write模块的wframe转换成array\_interface，控制array的写操作。

#### FSM description



**IDLE：**

FSM默认处于IDLE状态，ready为1。当wr\_start == 1时，即array\_write模块接收到有效写数据时，选中array\_write的ready穿透给上级模块，握手成功，跳转到TSADDR状态。当wr\_start == 0时，保持在IDLE状态。

**TSADDR：**

1. 将frame中的行地址放到总线上，等array\_cs\_n的下降沿时刻，对行地址进行采样。
2. 内置计数器赋初值trcd\_wr - 1并开始倒数（在TRCD\_WR的前一个状态对计数器赋初值）。
3. tras计数器赋初值tras – 1并开始计数。
4. 一个时钟周期后，无条件跳转到TRCD\_WR状态。
5. 连续写操作时，TRP最后一拍将替代TSADDR时间。

**TRCD\_WR：**

1. 进入该状态，立即拉低array\_cs\_n，对array\_raddr进行采样。
2. 当内置计数器计数到0时，说明已满足trcd\_wr时间，将列地址放到总线上，并将array\_caddr\_vld\_wr第一次置1，根据frame的eof和sof判断状态跳转。内置计数器未计数到0时，trcd\_wr\_finish == 0，保持在TRCD\_WR状态。
   1. eof和sof同时为1时，wr\_single == 1，说明只有一个frame，跳转到WR\_LAST状态。
   2. eof和sof不同时为1时，wr\_single == 0，说明不止一个frame，跳转到WR状态。

**WR：**

1. 因为array是根据array\_caddr\_vld\_wr的下降沿采列地址的，所以array\_caddr\_vld\_wr为0时，说明当前的列地址已被array采走，此时可将ready置1（ready == (fsm\_cs == IDLE) || (fsm\_cs==WR && ~array\_caddr\_vld)）。
2. 与上级模块握手更新写数据，写数据更新完成后，将array\_caddr\_vld\_wr置1，表示数据有效，一个时钟周期后，array\_caddr\_vld\_wr置0，重复上述操作。
3. 当接收到的frame中的eof为1时，即更新最后一个写数据时，wr\_pre\_finish == 1，跳转到WR\_LAST状态。wr\_pre\_finish == 0时，保持在WR状态。

**WR\_LAST：**

1. 向array发送最后一个写数据。
2. 内置计数器赋初值twr – 1并开始倒数（在TWR的前一个状态对计数器赋初值）。
3. 一个时钟周期后，无条件跳转到TWR状态。

**TWR：**

当内置计数器和tras计数器都计数到0时，说明已满足twr和tras时间，将array\_cs\_n置1，关闭当前行，twr\_finish == 1，跳转到PRE\_TRP状态。计数未到0时，twr\_finish == 0，保持在TWR状态。

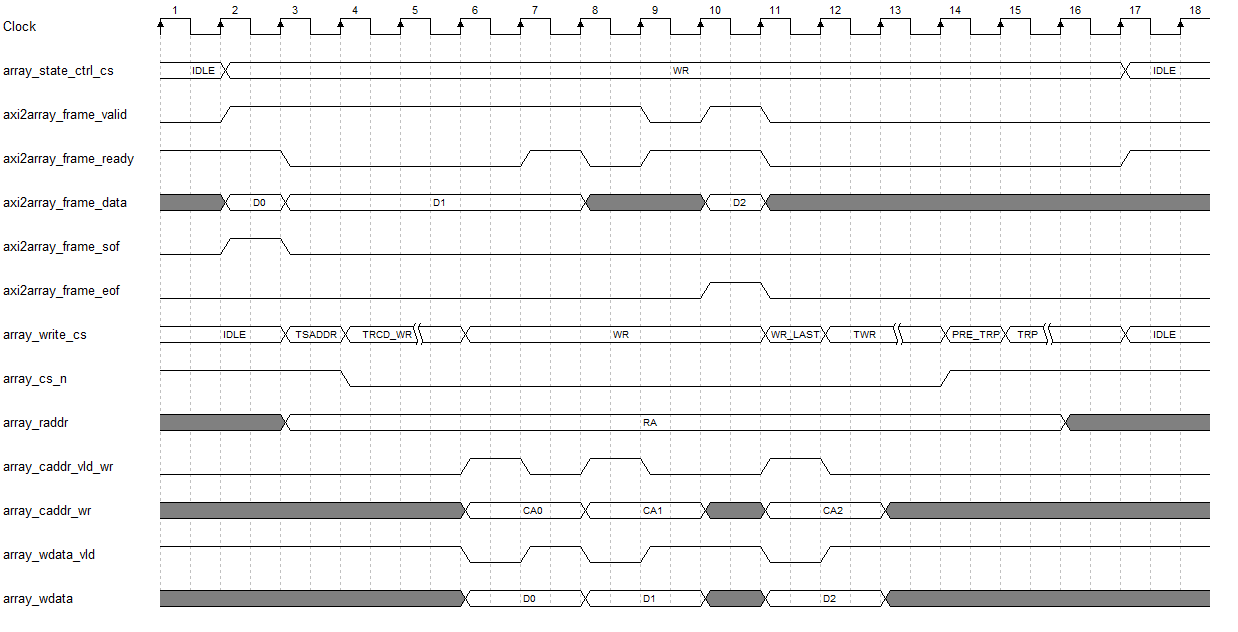
**PRE\_TRP：**

1. 内置计数器赋初值trp – 2并开始倒数（在TRP的前一个状态对计数器赋初值），因为当前状态占用了trp的一个时钟周期，所以是减2。
2. 一个时钟周期后，无条件跳转到TRP状态。

**TRP：**

当内置计数器计数到0时，说明已满足trp时间，trp\_finish == 1，在TRP的最后一拍跳回IDLE状态。计数未到0时，trp\_finish == 0，保持在TRP状态。

#### Timing



### array\_read

#### Function description

将array\_state\_ctrl模块分发给array\_read模块的rframe转换成array\_interface，控制array的读操作。

异步FIFO：

1. FIFO的非空信号即为输出数据的valid信号。

#### FSM description



**IDLE：**

FSM默认处于IDLE状态，ready为1。当rd\_start == 1时，即array\_read模块接收到有效读数据时，选中array\_read的ready穿透给上级模块，握手成功，跳转到TSADDR状态。当rd\_start == 0时，保持在IDLE状态。

**TSADDR：**

1. 将frame中的行地址放到总线上，等array\_cs\_n的下降沿时刻，对行地址进行采样。
2. 内置计数器赋初值trcd\_rd - 1并开始倒数（在TRCD\_RD的前一个状态对计数器赋初值）。
3. tras计数器赋初值tras – 1并开始计数。
4. 一个时钟周期后，无条件跳转到TRCD\_RD状态。
5. 连续读操作时，TRP最后一拍将替代TSADDR时间。

**TRCD\_RD：**

1. 进入该状态，立即拉低array\_cs\_n，对array\_raddr进行采样。
2. 当内置计数器计数到0时，说明已满足trcd\_rd时间，将列地址放到总线上，并将array\_caddr\_vld\_wr第一次置1，根据frame的个数判断状态跳转。计数未到0时，trcd\_rd\_finish == 0，保持在TRCD\_RD状态。
   1. eof和sof同时为1时，rd\_single == 1，说明只有一个frame，跳转到RD\_LAST状态。
   2. eof和sof不同时为1时，rd\_single == 0，说明不止一个frame，跳转到RD状态。

**RD：**

1. 因为array是根据array\_caddr\_vld\_rd的下降沿采列地址的，所以array\_caddr\_vld\_rd为0时，说明当前的列地址已被array采走，此时可将ready置1（ready == (fsm\_cs == IDLE) || (fsm\_cs==RD && ~array\_caddr\_vld)）。
2. 与上级模块握手更新写数据，读数据更新完成后，将array\_caddr\_vld\_rd置1，表示数据有效，一个时钟周期后，array\_caddr\_vld\_rd置0，重复上述操作。
3. 当接收到的frame中的eof为1时，即更新最后一个读数据时，rd\_pre\_finish == 1，跳转到RD\_LAST状态。rd\_pre\_finish == 0时，保持在RD状态。

**RD\_LAST：**

1. 向array发送最后一个读数据。
2. 内置计数器赋初值trtp– 1并开始倒数（在TRTP的前一个状态对计数器赋初值）。
3. 一个时钟周期后，无条件跳转到TRTP状态。

**TRTP：**

当内置计数器和tras计数器都计数到0时，说明已满足trtp和tras时间，将array\_cs\_n置1，关闭当前行，trtp\_finish == 1，跳转到PRE\_TRP状态。计数未到0时，trtp\_finish == 0，保持在TRTP状态。

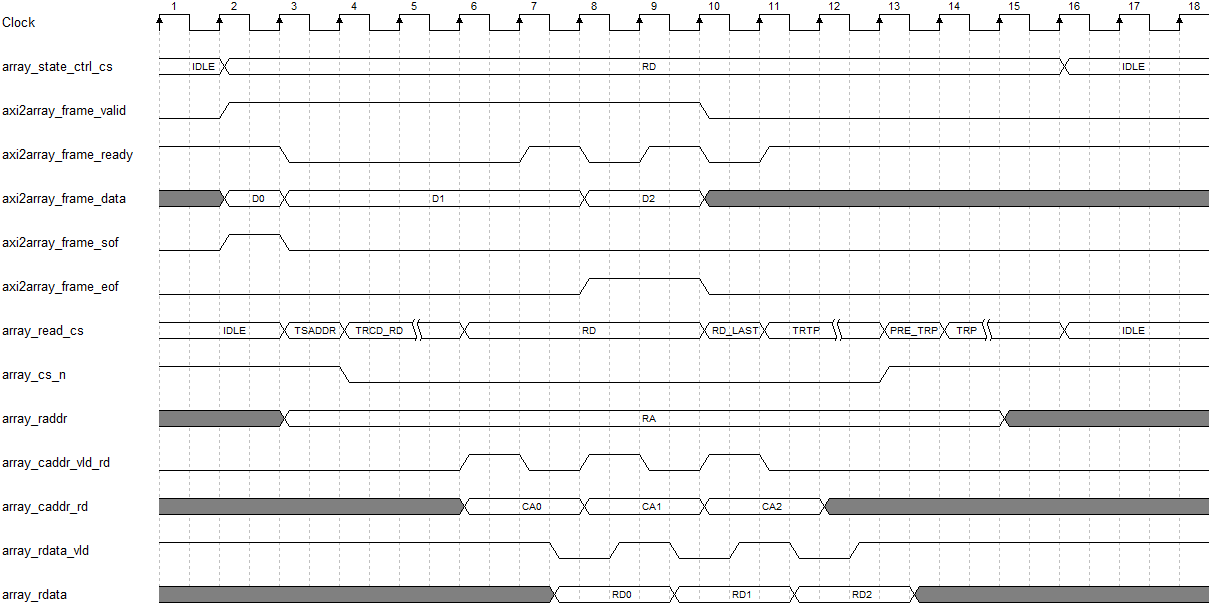
**PRE\_TRP：**

1. 内置计数器赋初值trp – 2并开始倒数（在TRP的前一个状态对计数器赋初值），因为当前状态占用了trp的一个时钟周期，所以是减2。
2. 一个时钟周期后，无条件跳转到TRP状态。

**TRP：**

当内置计数器计数到0时，说明已满足trp时间，trp\_finish == 1，在TRP的最后一拍跳回IDLE状态。计数未到0时，trp\_finish == 0，保持在TRP状态。

#### Timing



### array\_refresh

#### Function description

array\_ctrl中内置的刷新计数器计数到目标值时，发起刷新请求，控制array的刷新操作，其中array的刷新周期可配置。

若在执行读或写操作时，刷新请求到来了，则需要将该刷新请求存起来，等到读或写操作结束跳回IDLE状态后（这里处理完一帧后就跳回IDLE状态），立即处理保存下来的刷新请求，刷新请求的优先级最高。

#### FSM description



**IDLE：**

FSM默认处于IDLE状态，当array\_refresh模块接收到刷新请求时，rf\_start == 1，跳转到TSADDR状态。rf\_start == 0时，保持在IDLE状态。

**TSADDR：**

1. 将需要刷线的行地址放到总线上，等array\_cs\_n的下降沿时刻，对行地址进行采样。
2. tras计数器赋初值tras – 1并开始计数。
3. 一个时钟周期后，无条件跳转到TRAS状态。
4. 连续写操作时，TRP最后一拍将替代TSADDR时间。

**TRAS：**

tras计数器都计数到0时，说明已满足tras时间，将array\_cs\_n置1，关闭当前行，tras\_finish == 1，跳转到PRE\_TRP状态。计数未到0时，tras\_finish == 0，保持在TRAS状态。

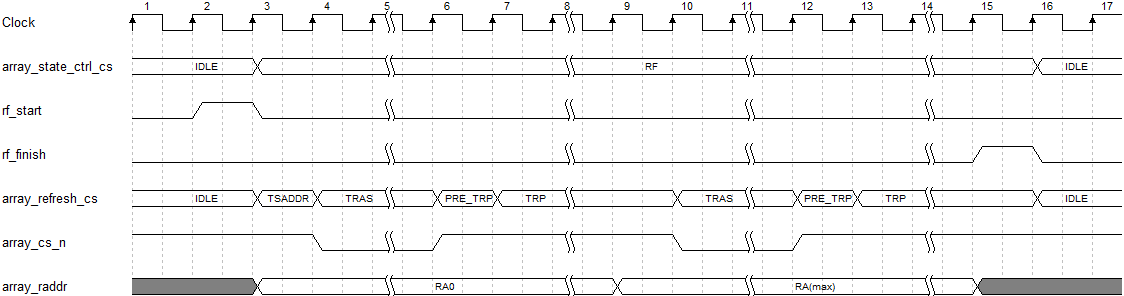
**PRE\_TRP：**

1. 内置计数器赋初值trp – 2并开始倒数（在TRP的前一个状态对计数器赋初值），因为当前状态占用了trp的一个时钟周期，所以是减2。
2. 一个时钟周期后，无条件跳转到TRP状态。

**TRP：**

当内置计数器计数到0时，说明已满足trp时间，trp\_finish == 1，在TRP的最后一拍跳回IDLE状态。计数未到0时，trp\_finish == 0，保持在TRP状态。

#### Timing



### array\_mux

array\_state\_ctrl模块的状态信号就是mux的选择信号。