Memory Controller Design Spec

Writer: Kaiqiang Huang

2022.10

contents

1. Overview

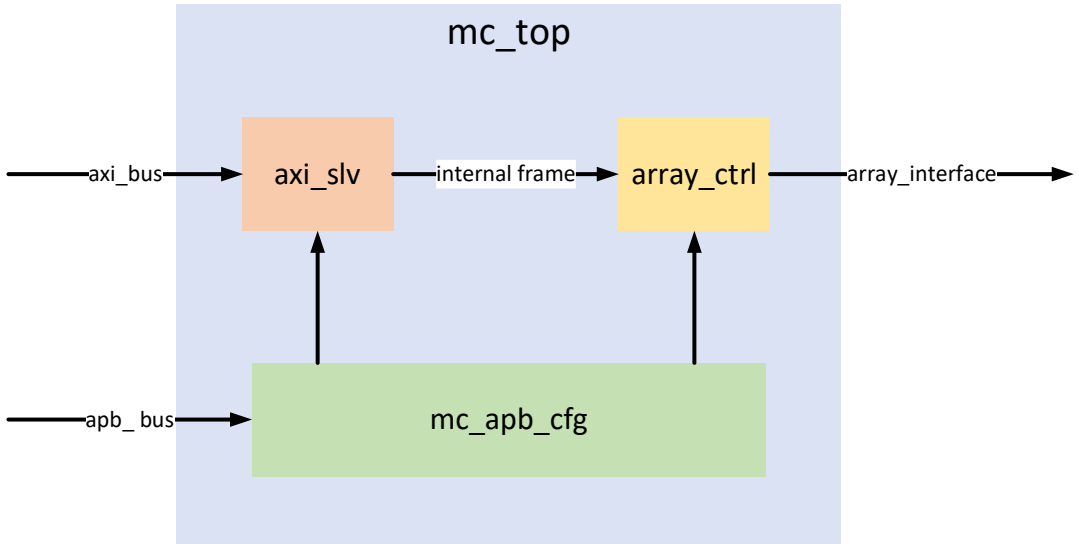
1.1 Function description

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。

1.2 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。

1.3 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。

1.4 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

2.axi_slave module

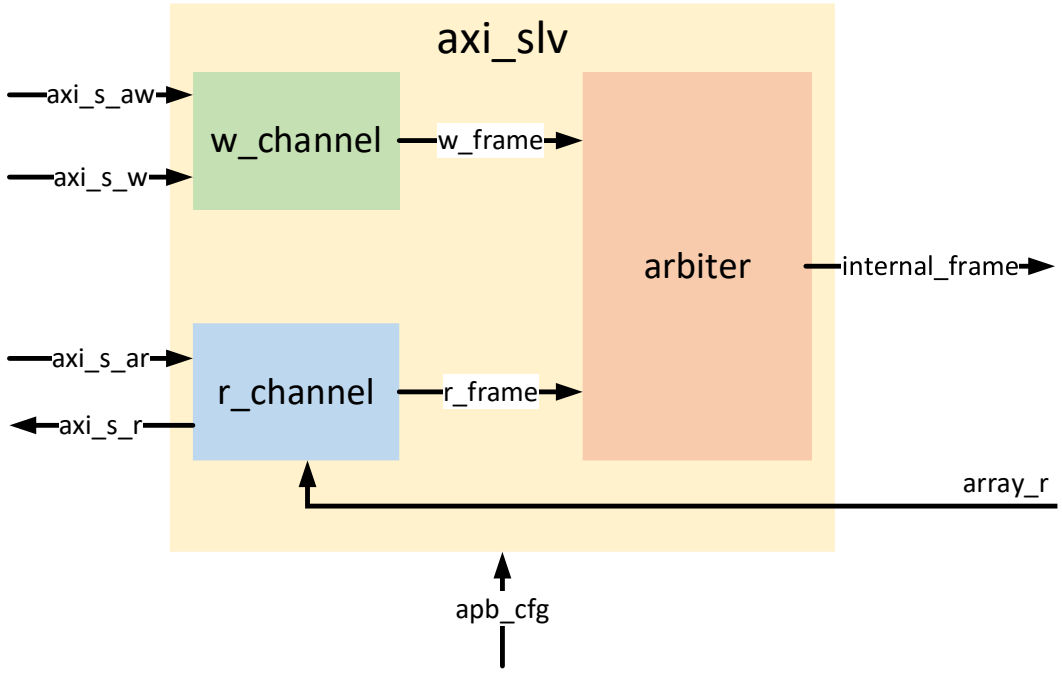
2.1 Function description

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。

2.2 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。

2.3 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。

2.4 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

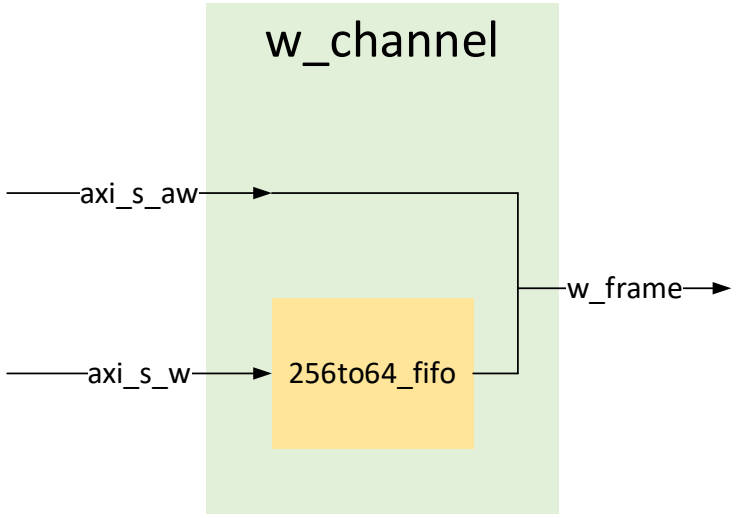
2.5 Primary sub module

2.5.1 w_channel

2.5.1.1 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。

2.5.1.2 block diagram

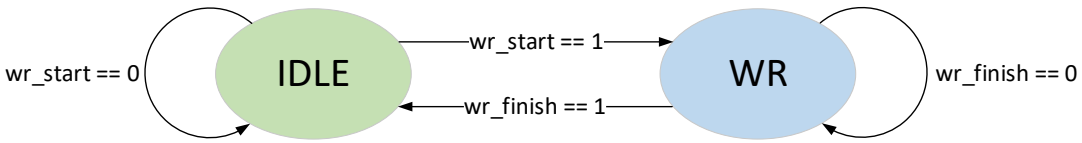


2.5.1.3 Interface description

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

2.5.1.4 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：

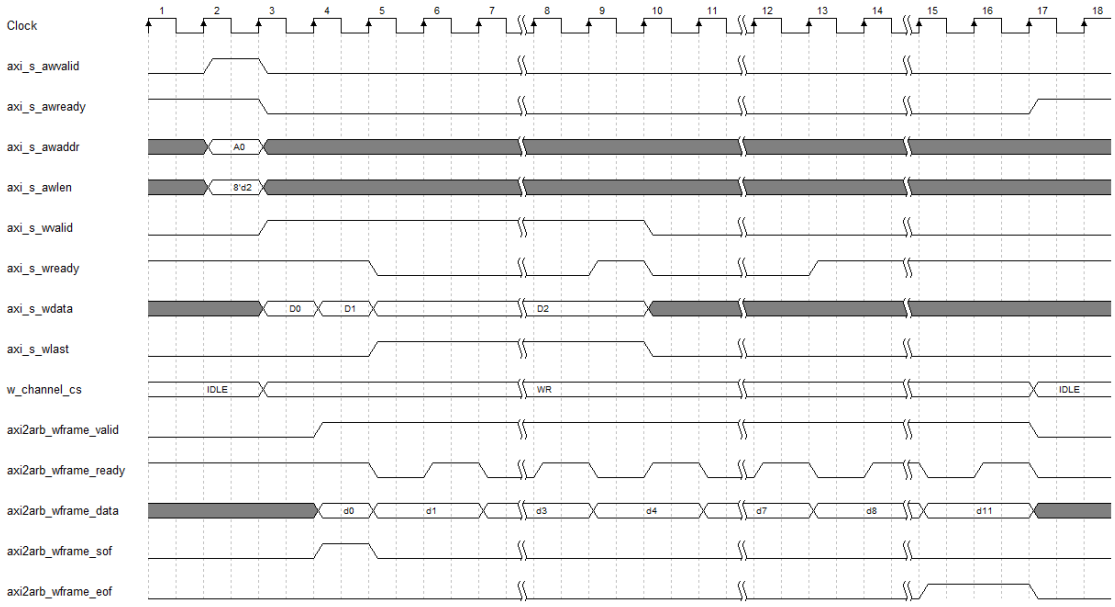
1. WR 状态下 axi_s_awready 为 0，反压上级模块对 aw 通道的操作。
2. 接收写地址 axi_s_awaddr（首地址）并将其保存到 22-bit 寄存器 awaddr_reg 中。
3. 接收 axi_s_awlen 并保存到 8-bit 寄存器 awlen_reg 中。
4. 接收写数据 axi_s_wdata 并将其保存到 256to64_fifo 中。通过 256to64_fifo 将一个 256-bit 的写数据拆分成 4 个 64-bit 的写数据。
   a) FIFO 位宽为 256-bit，深度为 2。
   b) FIFO 满时 axi_s_wready 为 0，反压上级模块。
   c) FIFO 非空时，axi2arb_wframe_valid 为 1，发送 w_frame，即 axi2arb_wframe_valid

= ((fsm_cs == WR) && ~fifo_empty)。

d) 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：
   a) 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；
   b) 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；
   c) 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；
   d) 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。

2.5.1.5 Timing



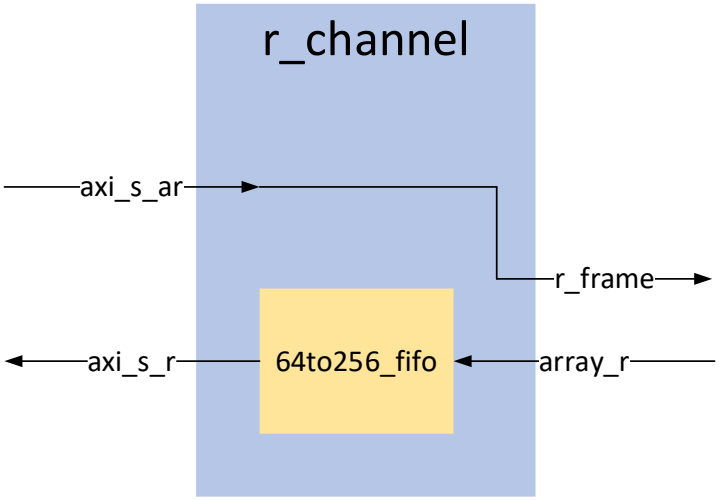
2.5.2 r_channel

2.5.2.1 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。

2.5.2.2 block diagram

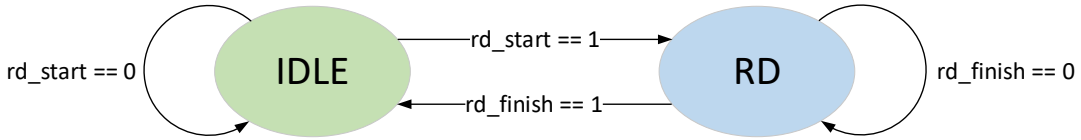


2.5.2.3 Interface description

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

2.5.2.4 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。

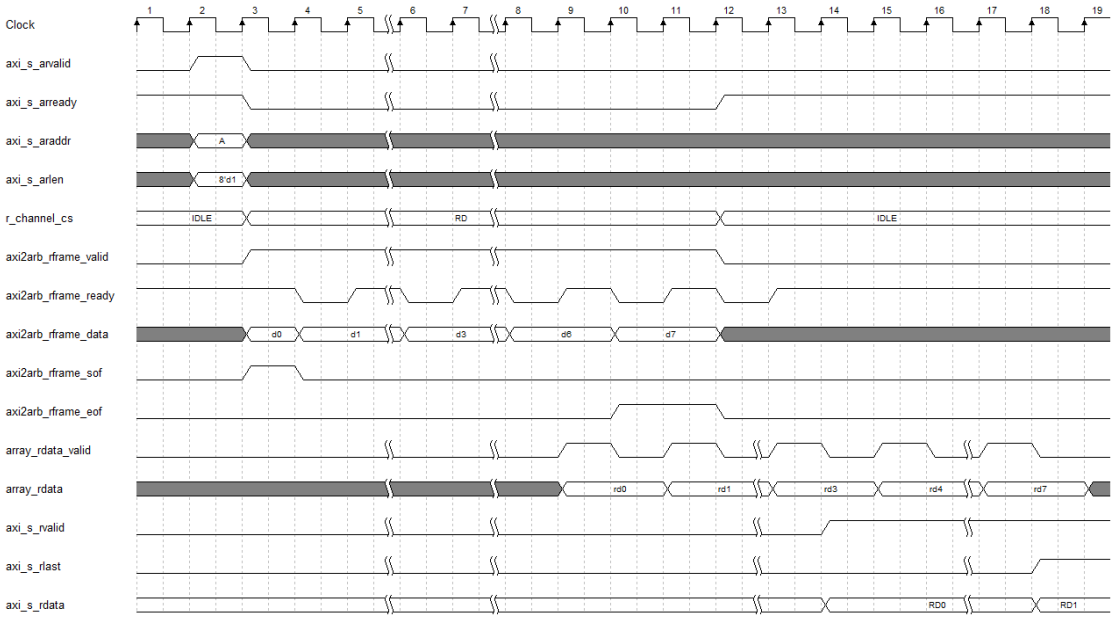
RD：

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。
   a) 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。
   b) FIFO 位宽为 8-bit，深度为 2。
   c) 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)。
   d) 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。
   a) 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。

2.5.2.5 Timing



2.5.3 arbiter

2.5.3.1 Function description

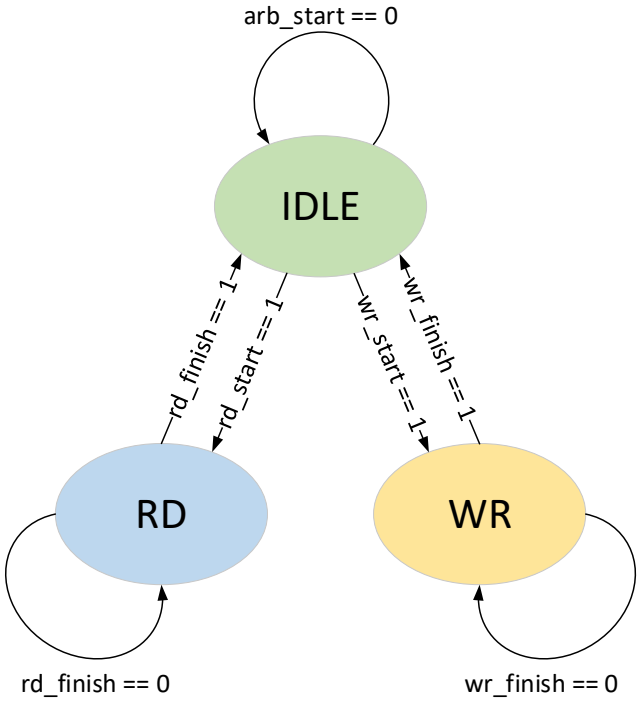
对 w_channel 模块与 r_channel 模块发出的 w/r_frame 进行仲裁，并将仲裁所得的 frame 发送到 array_ctrl 模块。

2.5.3.2 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

2.5.3.3 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：
   a) axi2arb_wframe_valid == 1，axi2arb_rframe_valid == 0，跳转到 WR 状态。
   b) axi2arb_wframe_valid == 0，axi2arb_rframe_valid == 1，跳转到 RD 状态。
   c) 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。

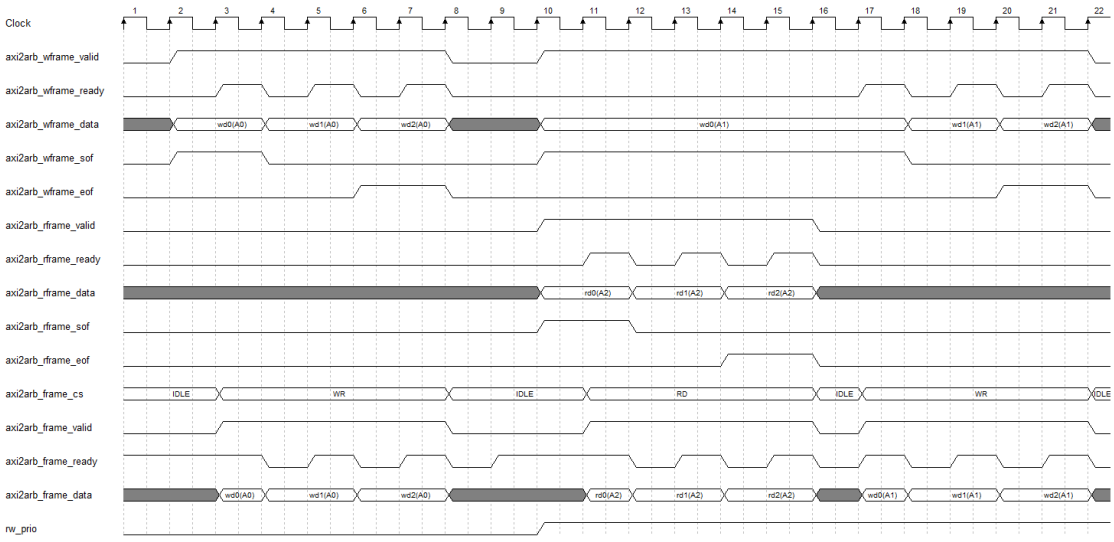
WR：

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。

2.5.3.4 Timing



3. array_ctrl module

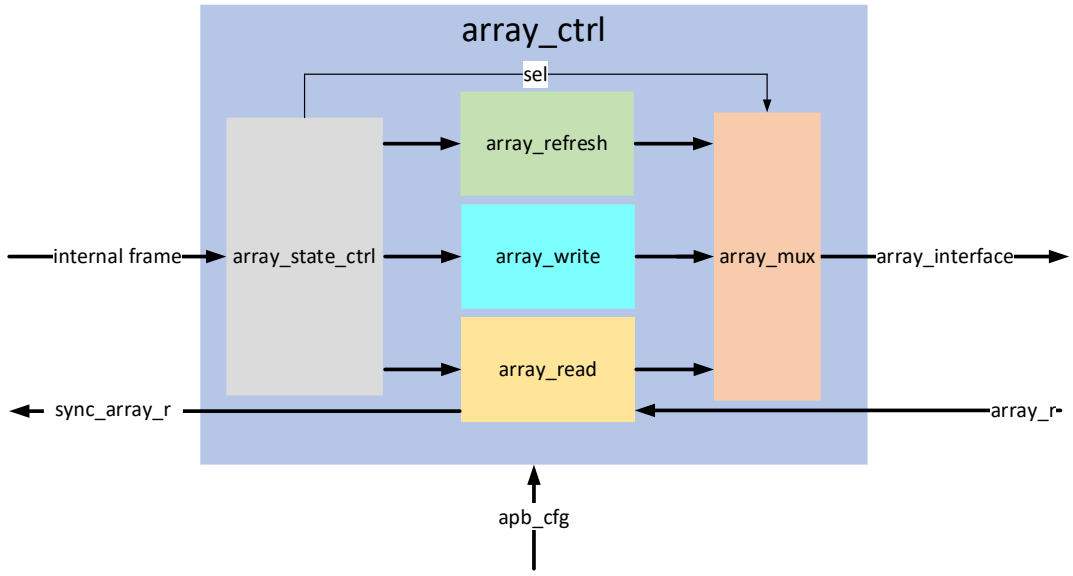
3.1 Function description

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。

3.2 Feature list

- 支持 array 接口时序可配置。
- 支持 array 刷新周期可配置。

3.3 Block diagram



3.4 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)

3.5 Primary sub module

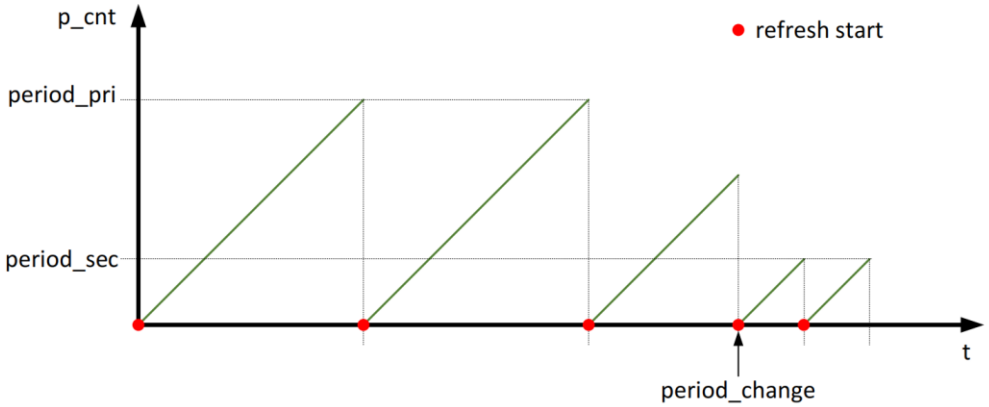
3.5.1 array_state_ctrl

3.5.1.1 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。

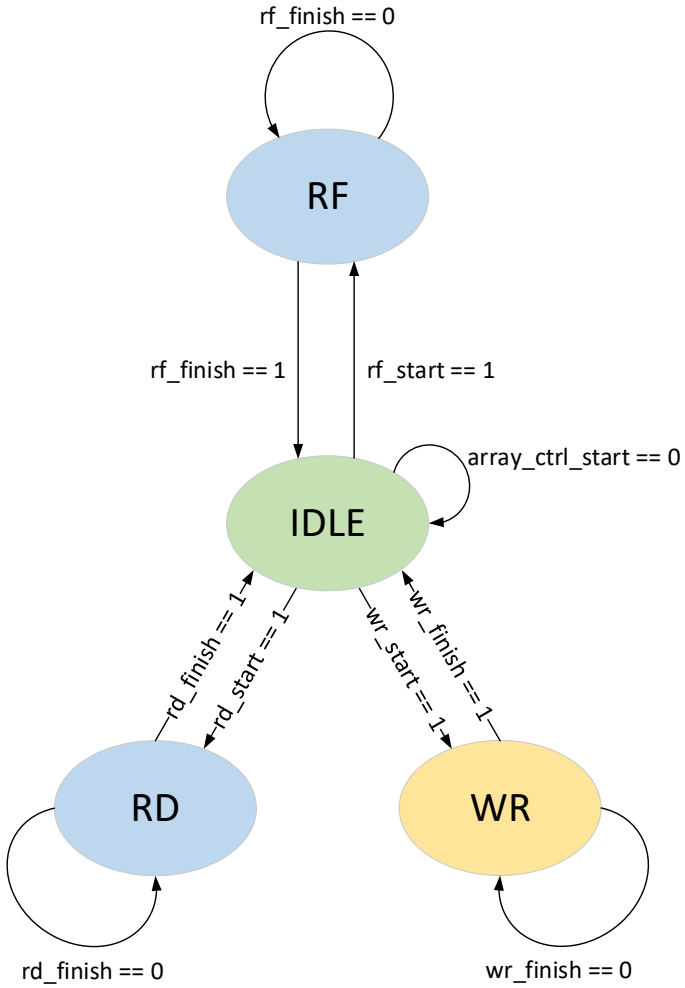
内置刷新周期计数器，控制 RF 状态的跳转，内置的刷新周期计数器计数到目标值时，计数器清零，并发起刷新请求，若此时未处于 IDLE 状态，则将该请求保存下来，等到 FSM 跳回 IDLE 状态时，立即执行刷新请求，故刷新周期的设置是有一定余量的。

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。

3.5.1.2 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。

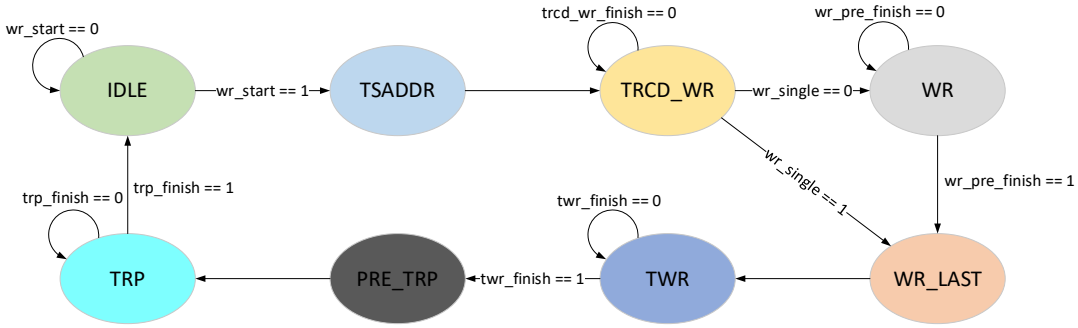
RD：对 array 发起读操作，precharge 中的最后一拍，即 rd_finish == 1 时，跳回 IDLE 状态。

3.5.2 array_write

3.5.2.1 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。

3.5.2.2 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。
   a) 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。
   b) 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。

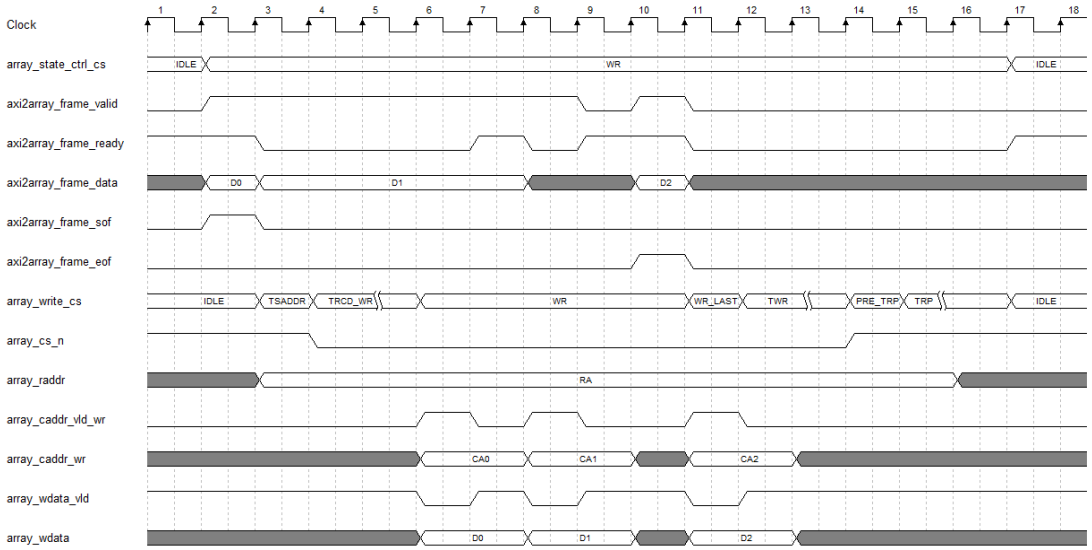
PRE_TRP：

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。

3.5.2.3 Timing



3.5.3 array_read

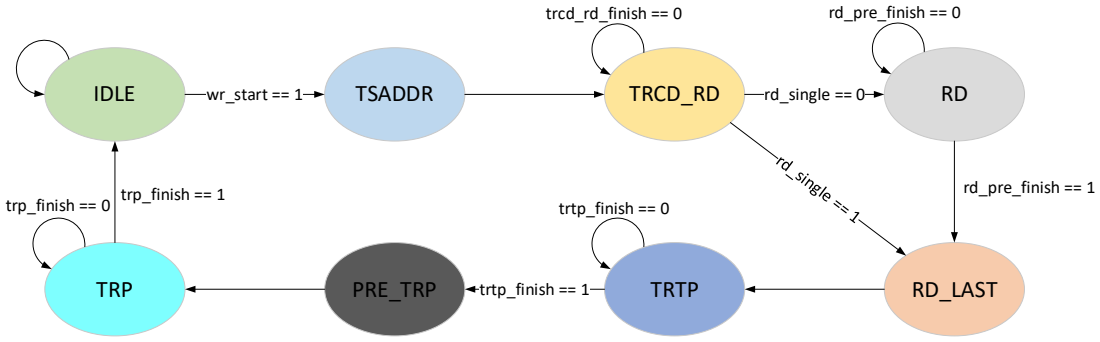
3.5.3.1 Function description

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。

3.5.3.2 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。
   a) 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。
   b) 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。

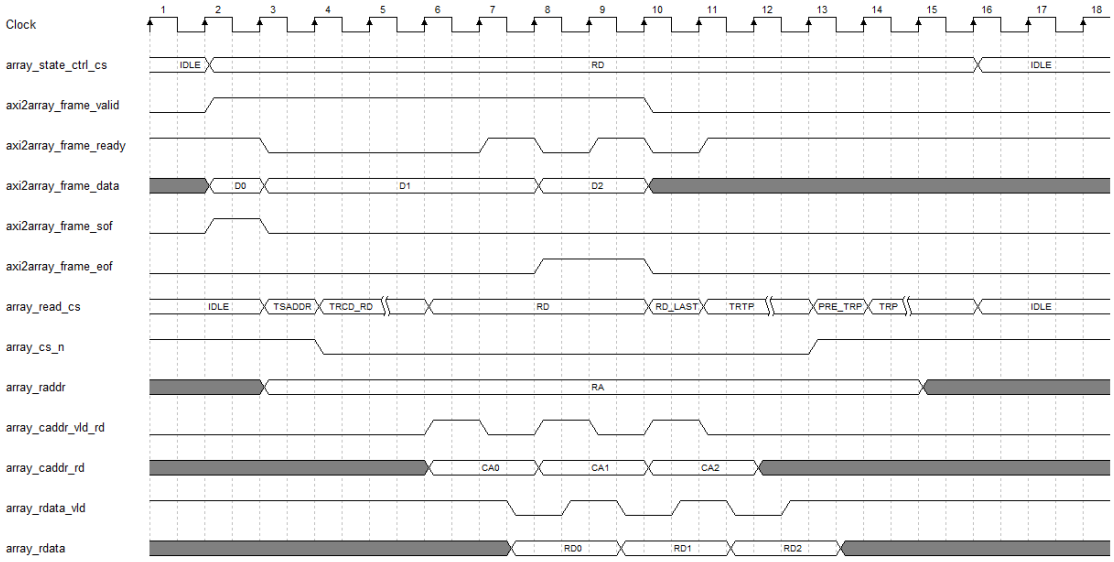
PRE_TRP：

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。

3.5.3.3 Timing



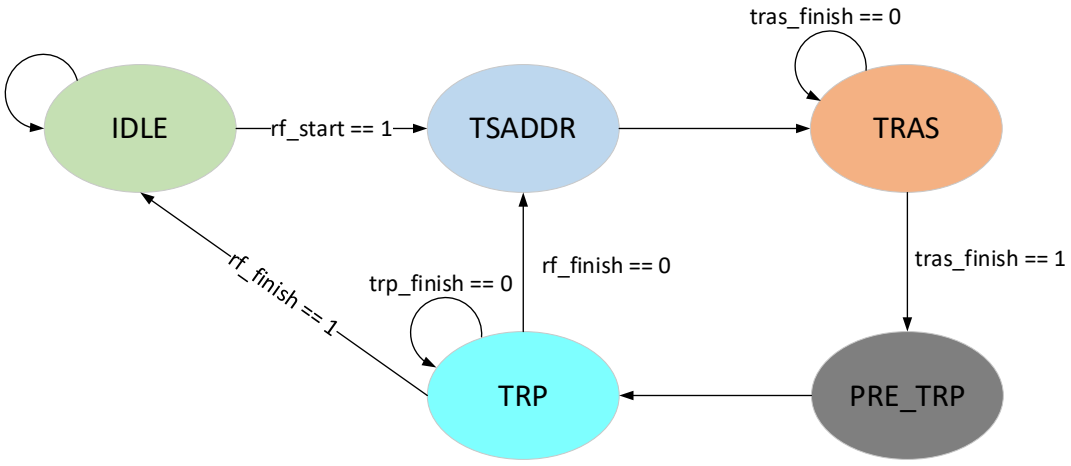
3.5.4 array_refresh

3.5.4.1 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。

3.5.4.2 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。

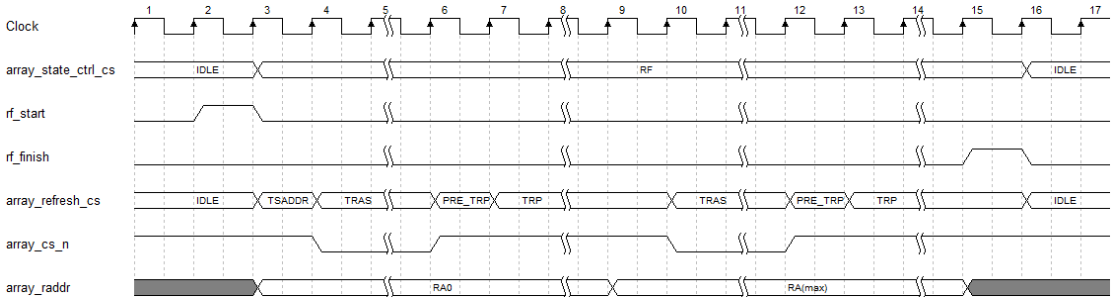
PRE_TRP：

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。

3.5.4.3 Timing



3.5.5 array_mux

array_state_ctrl 模块的状态信号就是 mux 的选择信号。