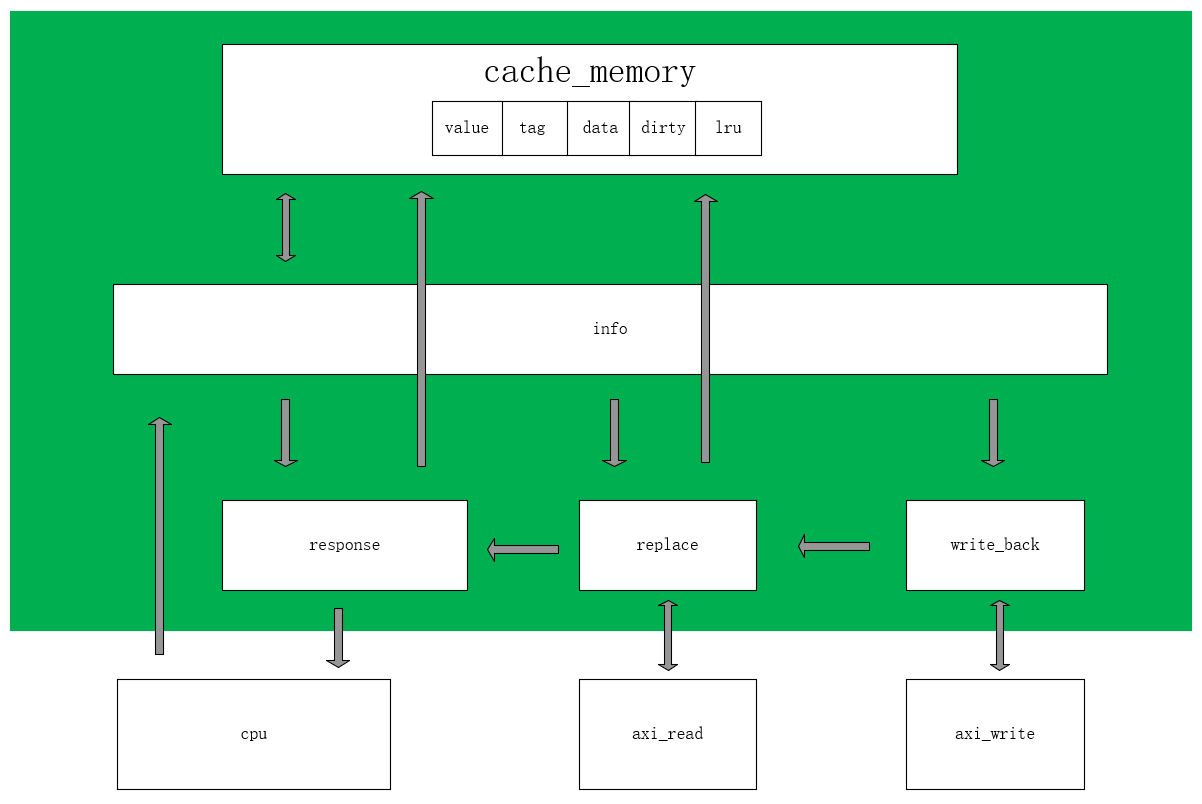
Cache模块逻辑设计方案

逻辑负责人：吴伟

# 1、Cache模块整体架构

## 1.1 Cache模块逻辑框图



图中，绿色部分为cache的组件，由cache\_memory、info、response、replace、write\_back这五个构件组成，其中，cache\_memory中存储的信息是value，tag，data，dirty，lru。value用于表示该cache line中的数据是否有效，dirty表示该cache line中的数据是否为脏。lru用于记录最近访问的组，tag用于记录data对应的tag信息。info用于读取cache信息，response用于向CPU反馈结果，write\_back用于向memory写回脏数据，replace用于替换未命行的数据。

## 1.2 cache的基本信息

cache中的可存储的数据大小：4KB（tag不计算在内）。

cache line ：64 Bytes

tag位宽：53 bits

二路组相连

替换算法：LRU（最近最少使用）

## 1.3 cache工作行为描述：

（1）当CPU发起的读请求命中时，cache会直接返回对应的数据，并更新lru的值。

（2）当CPU发起的写请求命中时，cache会直接返回写完成信号，同时将对应的dirty位置为有效，并更新lru的值。

（3）当CPU发起的读写请求没有命中时，cache会将最近最少使用的那一路数据进行替换。替换的过程分为三步。① 若待替换的数据为脏，则先将待替换的数据写回内存中，同时将对应的dirty位清零；若待替换的数据不为脏，则本步骤不进行任何操作。② 从内存中读出新的数据并将其存储到cache中的对应位置，并将对应的value置为有效。③ 根据CPU发起的是读请求还是写请求，分别重复（1）或者（2）的行为。

# 2、模块介绍

## 2.1 info模块：

info模块用于接受CPU发起的访存请求，然后将cache\_memroy中对应index的信息全部读出，并将信息进行处理，传递给response模块，replace模块以及write\_back模块。

info模块传递的信息主要包含以下内容：

* info\_hit : CPU的访存请求是否命中
* info\_hit\_way : 若命中，则命中的是哪一路
* info\_hit\_data: 若命中，则命中那一路的数据
* info\_miss : CPU的访存请求是否没有命中
* info\_rplc\_way : 若没有命中，则需要替换的是哪一路
* info\_rplc\_data : 若没有命中，则需要替换的那一路的数据
* info\_rplc\_dirty : 若没有命中，则需要替换的那一路的数据是否为脏
* info\_rplc\_dirty : 若没有命中，则需要替换的那一路的tag

info模块引脚介绍：

|  |  |  |  |
| --- | --- | --- | --- |
| I/O | Width | Name | Discription |
| input | 1 | core\_cen | CPU发起读写请求 |
| input | 5 | core\_index | CPU传递的读写地址中的index部分 |
| input | 53 | core\_tag | CPU传递的读写地址中的tag部分 |
| input | 1 | core\_rsp | CPU读写响应信号 |
| output | 1 | info\_rsp | info模块输出的响应信号，表示该模块工作已完成 |
| output | 1 | info\_arb | info模块参与对cache\_memroy模块读写仲裁的请求信号 |
| input | 1 | info\_hit | info模块传递的是否命中信息 |
| input | 1 | info\_hit\_way | 命中的是哪一路 |
| input | 512 | info\_hit\_data | 命中的那一路的数据 |
| input | 1 | info\_miss | info模块传递的是否命中信息 |
| input | 1 | info\_rplc\_way | 待替换的路 |
| input | 1 | info\_rplc\_dirty | 待替换的路的数据是否为脏 |
| input | 512 | info\_rplc\_data | 待替换的路的数据 |
| input | 53 | info\_rplc\_tag | 待替换的路的tag信息 |
|  |  |  |  |
| input | 1 | info\_tag\_cen0 | way0的tag片选信号 |
| input | 1 | info\_tag\_wen0 | way0的tag读写信号，高电平为写 |
| input | 5 | info\_tag\_addr0 | way0的tag地址信号 |
| output | 53 | info\_tag\_rdata0 | way0的tag读数据信号 |
| input | 1 | info\_tag\_cen1 | way1的tag片选信号 |
| input | 1 | info\_tag\_wen1 | way1的tag读写信号，高电平为写 |
| input | 5 | info\_tag\_addr1 | way1的tag地址信号 |
| output | 53 | info\_tag\_rdata1 | way1的tag读数据信号 |
| input | 1 | info\_data\_cen0 | way0的data片选信号 |
| input | 1 | info\_data\_wen0 | way0的data读写信号，高电平为写 |
| input | 5 | info\_data\_addr0 | way0的data地址信号 |
| output | 512 | info\_data\_rdata0 | way0的data读数据信号 |
| input | 1 | info\_data\_cen1 | way1的data片选信号 |
| input | 1 | info\_data\_wen1 | way1的data读写信号，高电平为写 |
| input | 5 | info\_data\_addr1 | way1的data地址信号 |
| output | 512 | info\_data\_rdata1 | way1的data读数据信号 |
| input | 5 | info\_dirty\_raddr0 | way0的dirty读地址信号 |
| output | 1 | info\_dirty\_rdata0 | way0的dirty读数据信号 |
| input | 5 | info\_dirty\_raddr1 | way1的dirty读地址信号 |
| output | 1 | info\_dirty\_rdata1 | way1的dirty读数据信号 |
| input | 5 | info\_value\_raddr0 | way0的value读地址信号 |
| output | 1 | info\_value\_rdata0 | way0的value读数据信号 |
| input | 5 | info\_value\_raddr1 | way1的value读地址信号 |
| output | 1 | info\_value\_rdata1 | way1的value读数据信号 |
| input | 5 | info\_lru\_raddr | lru读地址信号 |
| output | 1 | info\_lru\_rdata | lru读数据信号 |

## 2.2 response模块：

response模块用于向CPU反馈信息。

response模块会在以下两种情况下启动：

* CPU的访存请求发生命中。
* replace模块工作完成。

response模块的工作流程如下所示：

①若core\_wen为低，则reponse模块会将core\_rsp拉高一个时钟周期，并且从512位的info\_hit\_data中选出core\_addr对应的64位数据。

若core\_wen为高，则response模块会将core\_rsp拉高一个时钟周期，并且向cache\_memory模块发送写数据请求

②response模块会在将core\_rsp拉高的同一个时钟周期更新lru的数据。

response模块引脚介绍

|  |  |  |  |
| --- | --- | --- | --- |
| I/O | Width | Name | Discription |
| input | 5 | core\_index | CPU传递的地址中的index部分 |
| input | 3 | core\_offset | CPU传递的地址中的offset部分，用于从512位数中选中一个64位的数据 |
| input | 1 | core\_wen | CPU传递的写使能信号 |
| input | 8 | core\_wstrb | CPU传递的写数据字节选通信号 |
| input | 64 | core\_wdata | CPU传递的写数据信号 |
| input | 1 | info\_rsp | info模块传递的信息有效信号 |
| input | 1 | info\_hit | info模块传递的是否命中信息 |
| input | 1 | info\_hit\_way | 命中的是哪一路 |
| input | 512 | info\_hit\_data | 命中的那一路的数据 |
| input | 1 | rplc\_rsp | replace模块响应信号，表示rplc\_rsp\_data有效 |
| input | 512 | rplc\_rsp\_data | replace模块返回的新数据 |
| input | 1 | info\_rplc\_way | info模块的指示信息，表示替换的是哪一路 |
| output | 1 | rsp\_arb | response模块参与对cache\_memroy模块读写仲裁的请求信号 |
|  |  |  |  |
| input | 5 | rsp\_dirty\_waddr0 | way0的dirty写地址信号 |
| input | 1 | rsp\_dirty\_wdata0 | way0的dirty写数据信号 |
| input | 1 | rsp\_dirty\_wen0 | way0的dirty写使能信号 |
| input | 5 | rsp\_dirty\_waddr1 | way1的dirty写地址信号 |
| input | 1 | rsp\_dirty\_wdata1 | way1的dirty写数据信号 |
| input | 1 | rsp\_dirty\_wen1 | way1的dirty写使能信号 |
|  |  |  |  |
| input | 5 | rsp\_lru\_waddr | lru写地址信号 |
| input | 1 | rsp\_lru\_wdata | lru写数据信号 |
| input | 1 | rsp\_lru\_wen | lru写使能信号 |
|  |  |  |  |
| input | 1 | rsp\_data\_cen0 | way0的data片选信号 |
| input | 1 | rsp\_data\_wen0 | way0的data读写信号，高电平为写 |
| input | 5 | rsp\_data\_addr0 | way0的data地址信号 |
| input | 512 | rsp\_data\_wdata0 | way0的data写数据信号 |
| input | 64 | rsp\_data\_wstrb0 | way0的data写数据字节选通信号 |
| input | 1 | rsp\_data\_cen1 | way1的data片选信号 |
| input | 1 | rsp\_data\_wen1 | way1的data读写信号，高电平为写 |
| input | 5 | rsp\_data\_addr1 | way1的data地址信号 |
| input | 512 | rsp\_data\_wdata1 | way1的data写数据信号 |
| input | 64 | rsp\_data\_wstrb1 | way1的data写数据字节选通信号 |

## 2.3 write\_back模块

write\_back模块用于将脏数据写回memory中。

当CPU的读写请求没有命中，并且待替换的行的数据为脏时，write\_back模块开始工作。

write\_back模块的工作流程如下所示：

①会向axi总线发起写数据请求，并将info传递的脏数据通过总线逐个写入memory中。

②等待接收总线的写反馈信号，代表写memory操作完成。

③（将数据的脏位清零，并且）发送write\_back模块工作完成信号。

write back 模块引脚介绍

|  |  |  |  |
| --- | --- | --- | --- |
| I/O | Width | Name | Discription |
| input | 1 | info\_miss | info模块传递的是否命中信息 |
| input | 1 | info\_rsp | info模块传递的信息有效信号 |
| input | 1 | info\_rplc\_dirty | 待替换的路的数据是否为脏 |
| input | 512 | info\_rplc\_data | 待替换的路的数据 |
| input | 53 | info\_rplc\_tag | 待替换的路的tag信息 |
| input | 5 | core\_index | CPU传递的地址中的index部分 |
| input | 1 | wb\_rsp | write back模块响应信号，表示模块工作结束 |
| input/output | - | wb\_aw\_channel | AXI write address channel通道信号，此为一簇信号，不一一列出。 |
| input/output | - | wb\_w\_channel | AXI write data channel通道信号，此为一簇信号，不一一列出。 |
| input/output | - | wb\_b\_channel | AXI write response channel通道信号，此为一簇信号，不一一列出。 |

## 2.4 replace模块

replace模块用于替换cache中的数据。

replace模块会在以下两种情况下启动：

* write\_back模块工作完成
* CPU的读写请求没有命中，并且待替换的cache\_line中的数据不为脏。

replace模块的工作流程如下

① 通过axi总线向memory发起读数据请求，burst长度为8，burst size为64。

② 等待接收axi总线上的返回的数据，直到axi\_rlast信号为高为止。

③ 将接收到的8个数据写入待替换的cache\_line中，然后修改待替换行的tag信息以及value信息，将对应的dirty值清零，发送替换完成信号。

模块引脚介绍：

|  |  |  |  |
| --- | --- | --- | --- |
| I/O | Width | Name | Discription |
| input | 1 | info\_miss | info模块传递的是否命中信息 |
| input | 1 | info\_rplc\_way | 待替换的路 |
| input | 1 | info\_rplc\_dirty | 待替换的路的数据是否为脏 |
| input | 1 | info\_rsp | info模块传递的信息有效信号 |
| input | 1 | wb\_rsp | write\_back模块响应信号 |
| output | 1 | rplc\_arb | replace模块参与对cache\_memory的读写仲裁请求信号 |
| input | 5 | core\_index | CPU传递的地址中的index部分 |
| input | 53 | core\_tag | CPU传递的地址中的tag部分 |
| output/input | - | rplc\_ar\_channel | AXI read address channel通道信号，此为一簇信号，不一一列出。 |
| input/output | - | rplc\_r\_channel | AXI read data channel通道信号，此为一簇信号，不一一列出。 |
| output | 512 | rplc\_rsp\_data | replace模块数据响应信号，数据是从memory中读出的新数据 |
| output | 1 | rplc\_rsp | replace模块响应信号 |
|  |  |  |  |
| output | 1 | rplc\_data\_cen0 | way0的data片选信号 |
| output | 1 | rplc\_data\_wen0 | way0的data读写信号，高电平为写 |
| output | 5 | rplc\_data\_addr0 | way0的data地址信号 |
| output | 512 | rplc\_data\_wdata0 | way0的data写数据信号 |
| output | 64 | rplc\_data\_wstrb0 | way0的data写数据字节选通信号 |
| output | 1 | rplc\_data\_cen1 | way1的data片选信号 |
| output | 1 | rplc\_data\_wen1 | way1的data读写信号，高电平为写 |
| output | 5 | rplc\_data\_addr1 | way1的data地址信号 |
| output | 512 | rplc\_data\_wdata1 | way1的data写数据信号 |
| output | 64 | rplc\_data\_wstrb1 | way1的data写数据字节选通信号 |
|  |  |  |  |
| output | 1 | rplc\_tag\_cen0 | way0的tag片选信号 |
| output | 1 | rplc\_tag\_wen0 | way0的tag读写信号，高电平为写 |
| output | 5 | rplc\_tag\_addr0 | way0的tag地址信号 |
| output | 53 | rplc\_tag\_wtag0 | way0的tag写数据信号 |
| output | 1 | rplc\_tag\_cen1 | way1的tag片选信号 |
| output | 1 | rplc\_tag\_wen1 | way1的tag读写信号，高电平为写 |
| output | 5 | rplc\_tag\_addr1 | way1的tag地址信号 |
| output | 53 | rplc\_tag\_wtag1 | way1的tag写数据信号 |
|  |  |  |  |
| output | 1 | rplc\_dirty\_wen0 | way0的dirty写使能信号 |
| output | 5 | rplc\_dirty\_waddr0 | way0的dirty写地址信号 |
| output | 1 | rplc\_dirty\_wdirty0 | way0的dirty写数据信号 |
| output | 1 | rplc\_dirty\_wen1 | way1的dirty写使能信号 |
| output | 5 | rplc\_dirty\_waddr1 | way1的dirty写地址信号 |
| output | 1 | rplc\_dirty\_wdirty1 | way1的dirty写数据信号 |
|  |  |  |  |
| output | 1 | rplc\_value\_wen0 | way0的value写使能信号 |
| output | 5 | rplc\_value\_waddr0 | way0的value写地址信号 |
| output | 1 | rplc\_value\_wvalue0 | way0的value写数据信号 |
| output | 1 | rplc\_value\_wen1 | way1的value写使能信号 |
| output | 5 | rplc\_value\_waddr1 | way1的value写地址信号 |
| output | 1 | rplc\_value\_wvalue1 | way1的value写数据信号 |

## 2.5 cache\_memory模块

cache\_memory模块用于存储cache中的数据信息。数据信息包括以下内容：

* data：将最近访问的数据存储到cache中，用SRAM存储。
* tag：用于记录该data对应的tag，用SRAM存储。
* value：用于记录该数据是否有效，用寄存器存储。
* dirty：用于记录该数据是否被改写过，改写过则该数据为脏，用寄存器存储。
* lru：用于记录最近最少访问的组，用寄存器存储

模块引脚介绍：

|  |  |  |  |
| --- | --- | --- | --- |
| I/O | Width | name | Discription |
| input | 1 | sram\_tag\_cen0 | way0的tag片选信号 |
| input | 1 | sram\_tag\_wen0 | way0的tag读写信号，高电平为写 |
| input | 5 | sram\_tag\_addr0 | way0的tag地址信号 |
| input | 53 | sram\_tag\_wdata0 | way0的tag写数据信号 |
| output | 53 | sram\_tag\_rdata0 | way0的tag读数据信号 |
| input | 1 | sram\_tag\_cen1 | way1的tag片选信号 |
| input | 1 | sram\_tag\_wen1 | way1的tag读写信号，高电平为写 |
| input | 5 | sram\_tag\_addr1 | way1的tag地址信号 |
| input | 53 | sram\_tag\_wdata1 | way1的tag写数据信号 |
| output | 53 | sram\_tag\_rdata1 | way1的tag读数据信号 |
|  |  |  |  |
| input | 1 | sram\_data\_cen0 | way0的data片选信号 |
| input | 1 | sram\_data\_wen0 | way0的data读写信号，高电平为写 |
| input | 5 | sram\_data\_addr0 | way0的data地址信号 |
| input | 512 | sram\_data\_wdata0 | way0的data写数据信号 |
| input | 64 | sram\_data\_wstrb0 | way0的data写数据字节选通信号 |
| output | 512 | sram\_data\_rdata0 | way0的data读数据信号 |
| input | 1 | sram\_data\_cen1 | way1的data片选信号 |
| input | 1 | sram\_data\_wen1 | way1的data读写信号，高电平为写 |
| input | 5 | sram\_data\_addr1 | way1的data地址信号 |
| input | 512 | sram\_data\_wdata1 | way1的data写数据信号 |
| input | 64 | sram\_data\_wstrb1 | way1的data写数据字节选通信号 |
| output | 512 | sram\_data\_rdata1 | way1的data读数据信号 |
|  |  |  |  |
| input | 5 | reg\_dirty\_raddr0 | way0的dirty读地址信号 |
| output | 1 | reg\_dirty\_rdata0 | way0的dirty读数据信号 |
| input | 5 | reg\_dirty\_waddr0 | way0的dirty写地址信号 |
| input | 1 | reg\_dirty\_wdata0 | way0的dirty写数据信号 |
| input | 1 | reg\_dirty\_wen0 | way0的dirty写使能信号 |
| input | 5 | reg\_dirty\_raddr1 | way1的dirty读地址信号 |
| output | 1 | reg\_dirty\_rdata1 | way1的dirty读数据信号 |
| input | 5 | reg\_dirty\_waddr1 | way1的dirty写地址信号 |
| input | 1 | reg\_dirty\_wdata1 | way1的dirty写数据信号 |
| input | 1 | reg\_dirty\_wen1 | way1的dirty写使能信号 |
|  |  |  |  |
| input | 5 | reg\_value\_raddr0 | way0的value读地址信号 |
| output | 1 | reg\_value\_rdata0 | way0的value读数据信号 |
| input | 5 | reg\_value\_waddr0 | way0的value写地址信号 |
| input | 1 | reg\_value\_wdata0 | way0的value写数据信号 |
| input | 1 | reg\_value\_wen0 | way0的value写使能信号 |
| input | 5 | reg\_value\_raddr1 | way1的value读地址信号 |
| output | 1 | reg\_value\_rdata1 | way1的value读数据信号 |
| input | 5 | reg\_value\_waddr1 | way1的value写地址信号 |
| input | 1 | reg\_value\_wdata1 | way1的value写数据信号 |
| input | 1 | reg\_value\_wen1 | way1的value写使能信号 |
|  |  |  |  |
| input | 5 | reg\_lru\_raddr | lru读地址信号 |
| output | 1 | reg\_lru\_rdata | lru读数据信号 |
| input | 5 | reg\_lru\_waddr | lru写地址信号 |
| input | 1 | reg\_lru\_wdata | lru写数据信号 |
| input | 1 | reg\_lru\_wen | lru写使能信号 |

# 3、与CPU的交互

core\_cen为高，core\_wen为低时，CPU发出读内存请求，读地址为core\_addr，读的数据经过core\_rdata信号返回，并且将core\_rsp作为读数据有效信号。

core\_cen为高，core\_wen为高时，CPU发出写内存请求，写地址为core\_addr，写数据为core\_wdata，写选通信号为core\_wstrb。core\_wstrb为字节选通信号，每一个位代表core\_wdata对应的字节是否有效。写数据操作完成之后，core\_rsp信号必须拉高一个周期。cache与CPU交互的时序图见5.5节。

关于地址对齐的说明：

cache不会对非对齐地址进行处理，而是简单地忽略core\_addr的低三位。因此要求CPU必须进行地址对齐处理，即core\_addr必须是8字节地址对齐的。对于lb、lh、lw、lbu、lhu、lwu操作，统一读取64位的数据，然后由CPU对读取的数据进行处理。对于sb、sh、sw操作，core\_addr必须是8字节地址对齐的，用core\_wstrb标识需要写入哪些字节位的数据。

# 4、模块中的AXI接口介绍

## 4.1 replace模块中的AXI接口

rplc\_arid ：若为I-Cache，则恒为1。若为D-Cache，则恒为0

rplc\_araddr ：读地址，64字节地址对齐。

rplc\_arlen ：burst长度，恒为7，表示长度为8。

rplc\_arsize ：每次传输数据的大小，恒为3‘b011，表示64位

rplc\_arburst ：burst模式选择，恒为2‘b01, 表示INCR模式

rplc\_arlock ：恒为0，表示normal

rplc\_arcache ：恒为0

rplc\_arport ：恒为0

rplc\_arqos ：恒为0

rplc\_arregion ：恒为0

rplc\_arvalid ：当araddr有效时拉高，直到rplc\_arready为低时拉低。

rplc\_arready ：从模块就绪信号。

rplc\_rid : 返回的ID信号，若为I-Cache，则应为1。若为D-Cache，则应为0

rplc\_rdata : 接收数据信号

rplc\_rresp : 指示读请求是否发生错误。为0表示没有错误。本模块没有处理错误的能力。

rplc\_rlast : 指示burst中的最后一个数据。

rplc\_rvalid : 数据有效信号。

rplc\_rready : 就绪信号，恒为1。

以上引脚中，除了恒定的信号外，均采用寄存器输出。当replace模块启动之后（关于replace模块会在什么情况下启动，请见2.4节），会通过寄存器输出64字节对齐的rplc\_araddr地址信号，并将rplc\_arvalid拉高，当检测到rplc\_arready为高之后，再将rplc\_arvalid拉低。然后等待read data channel 上返回的数据。当rplc\_rlast为高，并且rplc\_rvalid有效时，代表数据接收完成。数据接收完成后会发送replace完成信号，并且将接收到的数据写入存储数据的SRAM中。

## 4.2 write\_back模块中的AXI接口

wb\_awid ：为D-Cache，恒为0。（I-Cache不需要写回模块）

wb\_awaddr ：写地址，64字节地址对齐。

wb\_awlen ：burst长度，恒为7，表示长度为8。

wb\_awsize ：每次传输数据的大小，恒为3‘b011，表示64位

wb\_awburst ：burst模式选择，恒为2‘b01, 表示INCR模式

wb\_awlock ：恒为0，表示normal

wb\_awcache ：恒为0

wb\_awport ：恒为0

wb\_awqos ：恒为0

wb\_awregion ：恒为0

wb\_awvalid ：当araddr有效时拉高，直到rplc\_arready为低时拉低。

wb\_awready ：从模块就绪信号。

wb\_wid ：为D-Cache，恒为0。（I-Cache不需要写回模块）

wb\_wdata ：写数据，64位

wb\_wstrb ：写选通信号

wb\_wlast ：指示burst传输的最后一个数据

wb\_wvalid ：写数据有效信号

wb\_wready ：从设备就绪信号

wb\_bid ：为D-Cache，恒为0。（I-Cache不需要写回模块）

wb\_bresp ：写响应信号，为0表示没有错误。本模块没有处理错误的能力。

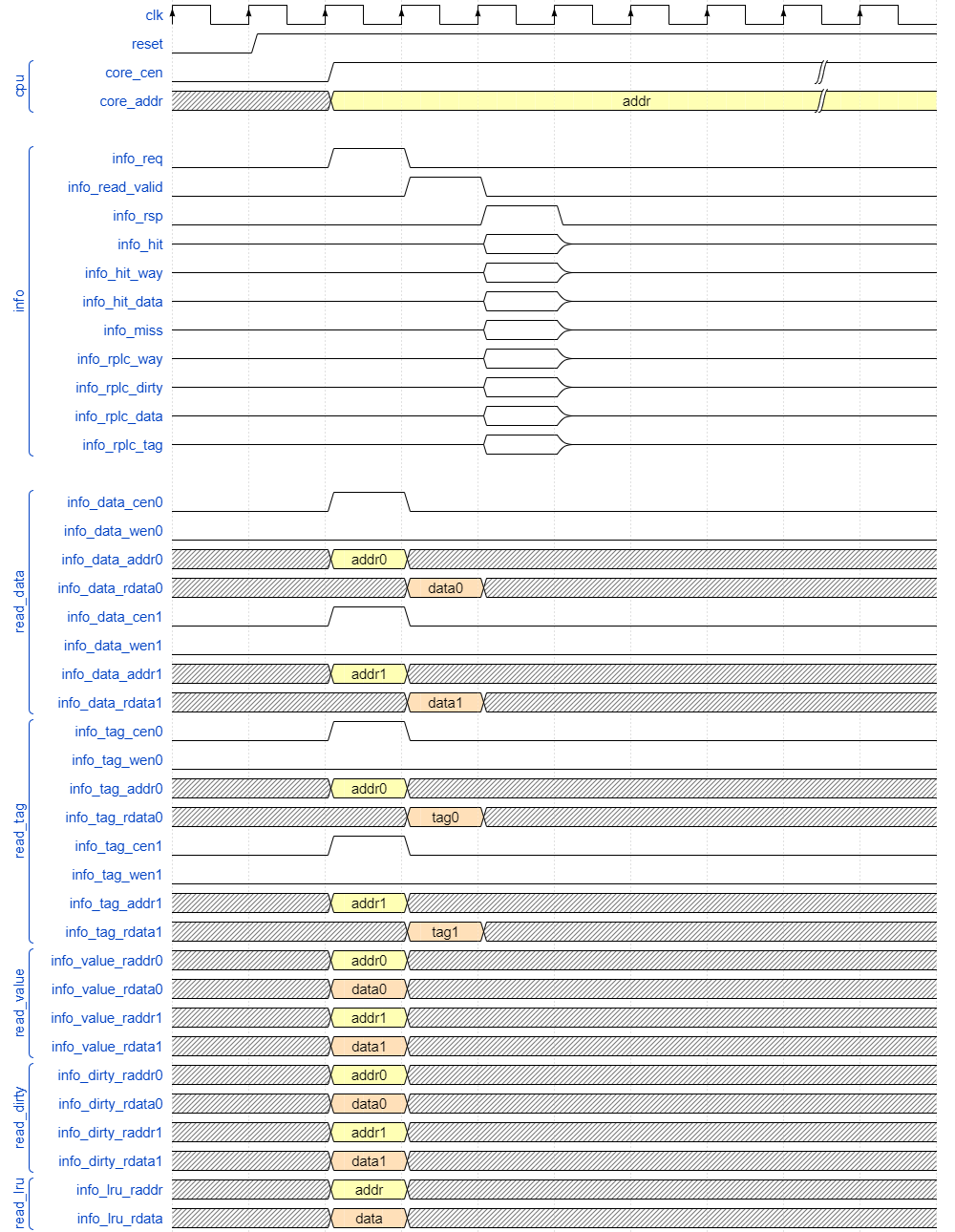
wb\_bvalid ：写响应有效信号

wb\_bready ：写响应接收就绪信号，恒为1

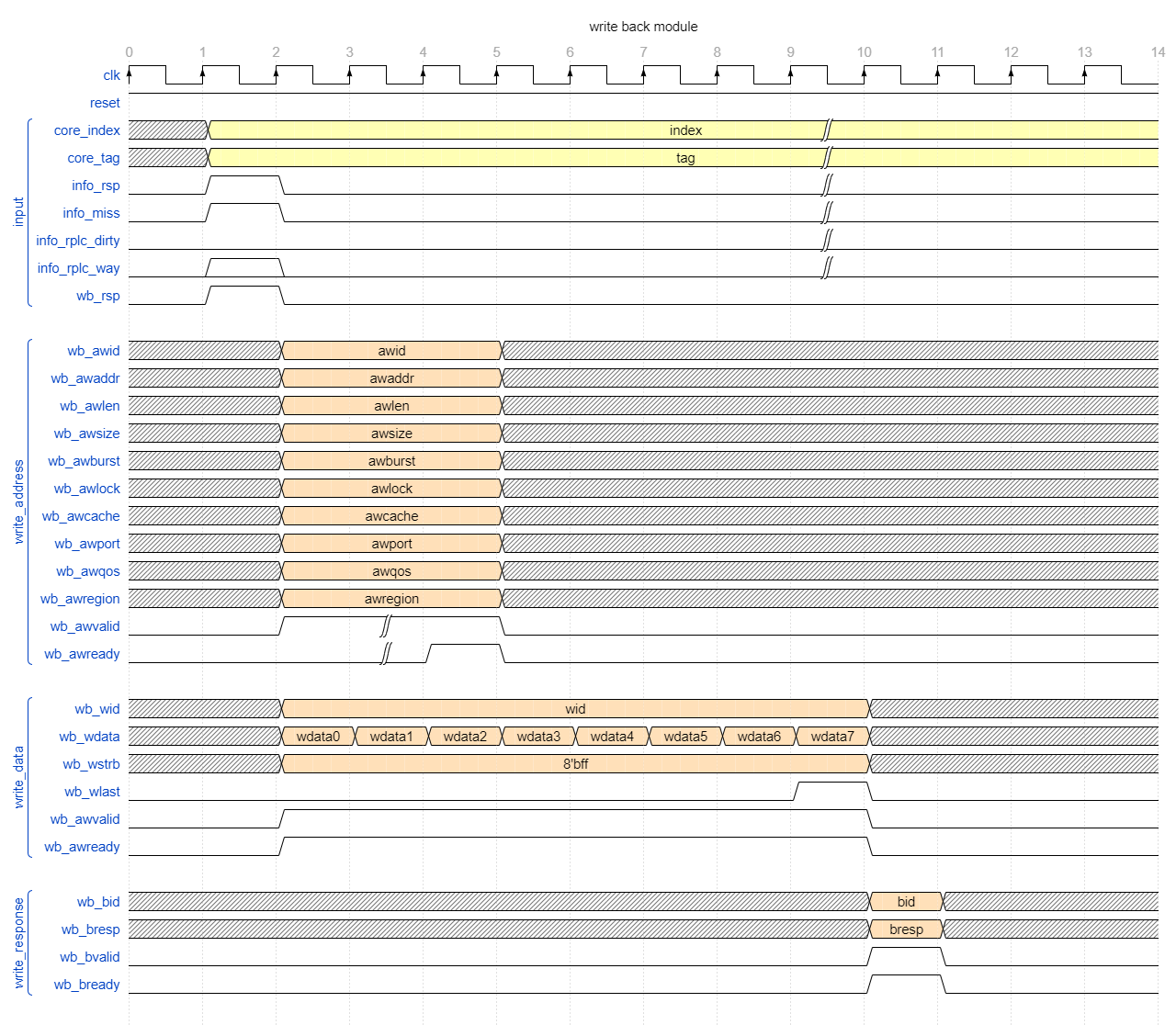
以上引脚中，除了恒定的信号外，均采用寄存器输出。当write\_back模块启动之后（关于write\_back模块会在什么情况下启动，请见2.3节），会通过寄存器输出64字节对齐的wb\_awaddr地址信号，并将wb\_awvalid拉高，当检测到wb\_awready为高之后，再将wb\_awvalid拉低。并且将需要写回memory的8个64位数据通过write data channel逐个发送出去，当发送到第八个数据时，将wb\_wlast拉高。写请求和写数据均发送完成之后，等待write response channel 返回的写响应信号。接收到写响应信号之后，发送write\_back模块完成信号。

# 5、Cache模块的时序

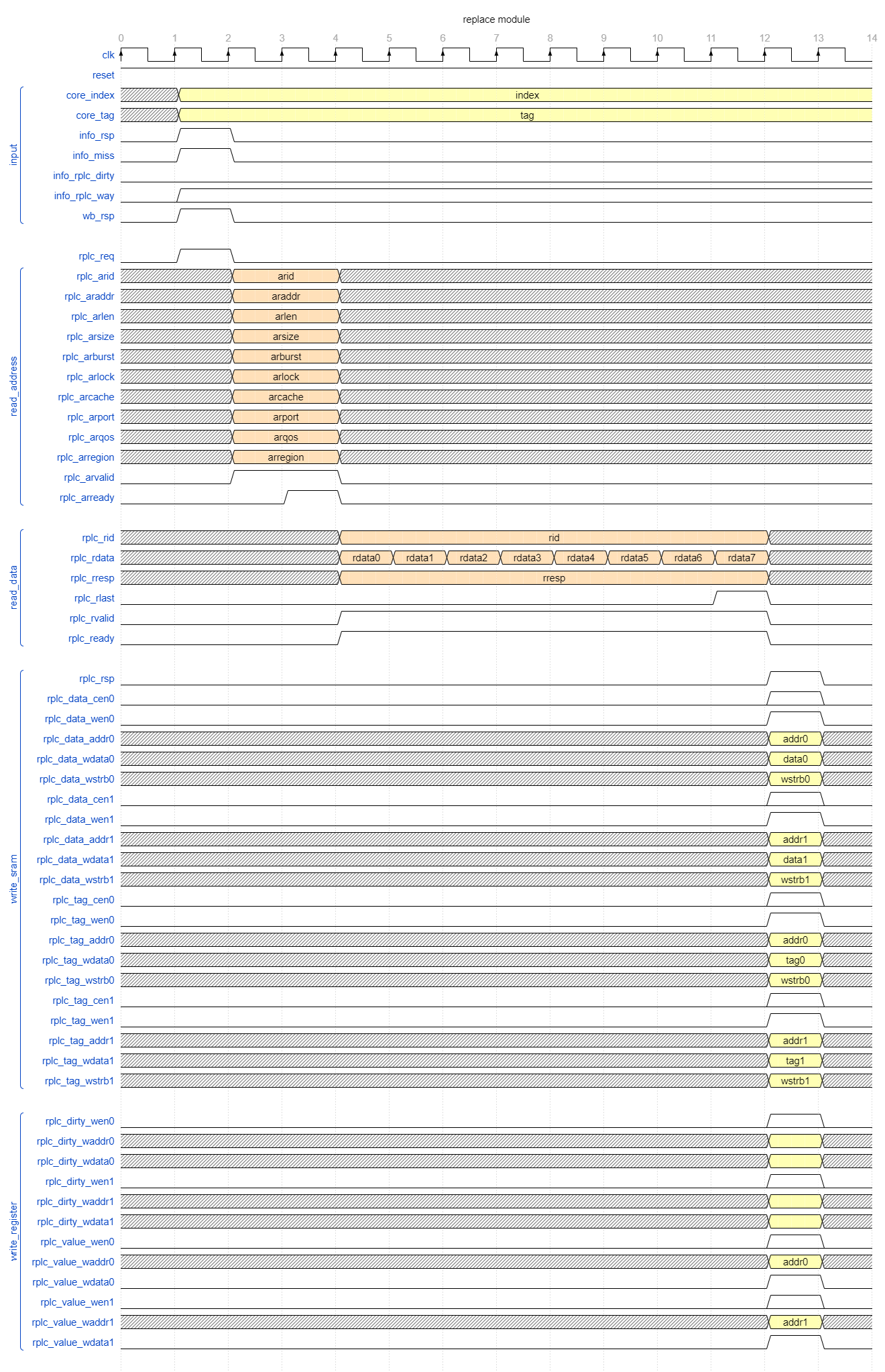
## 5.1 info获取信息的时序



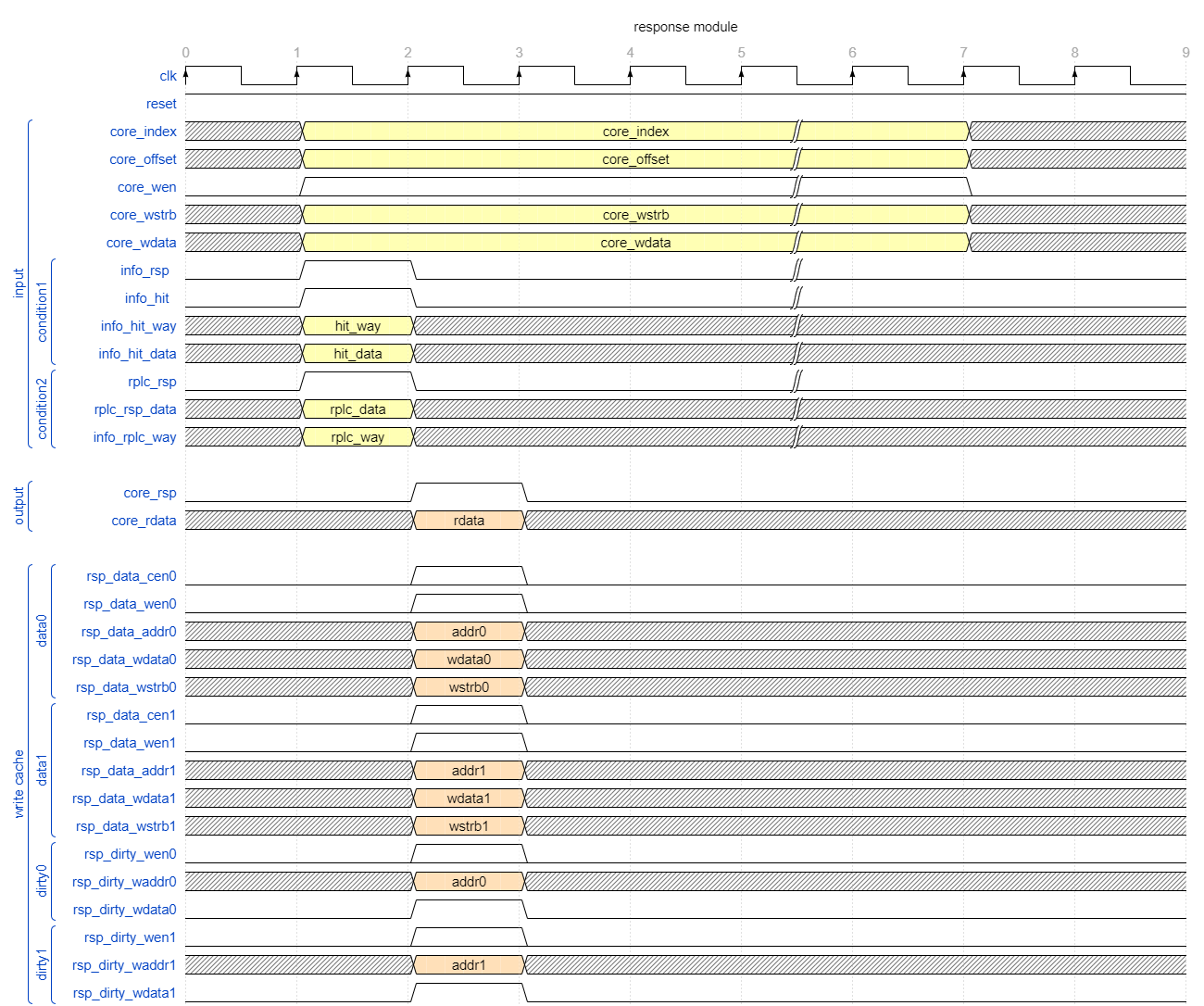
## 5.2 write\_back写回脏数据的时序



## 5.3 replace读取新数据的时序

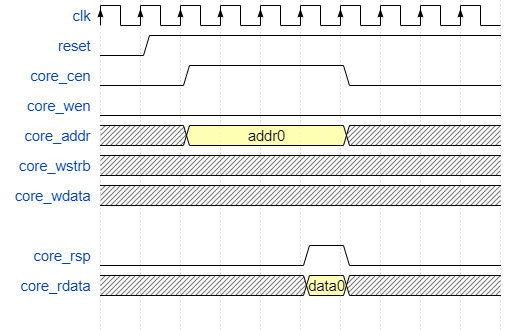


## 5.4 response模块时序

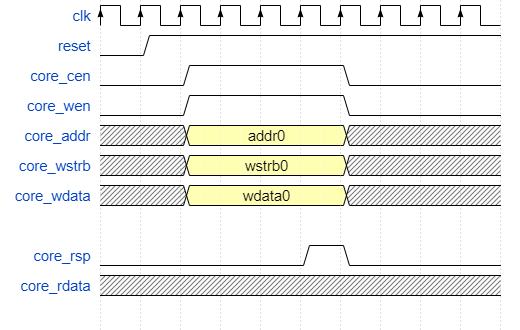


## 5.5 cache模块与CPU交互的时序

当CPU发送读数据请求，并且命中了cache中的数据时，时序如下所示。若发生了读命中，则cache会在三个时钟周期之后返回core\_addr地址对应的数据，并且core\_rsp拉高一个时钟周期，表示读数据完成。



当CPU发送写数据请求，并且命中了cache中的数据时，时序图如下所示。若发生了命中写，则会固定三个时钟周期之后返回写数据完成信号。



当CPU的读写请求未命中时，时序图与上图类似，区别在于无法在三个时钟周期后进行响应，响应时间无法确定。

# 6、cache模块资源分布

以下是使用vivado综合生成的资源分布评估报告。

# 