# 存储体系设计

## 一.数据正确性

1.使用hanming编码,拓宽有效数据值之间的hanming距离.
对数据进行SEC需要要求任意两个有效值之间hanming距离>=3
对数据进行DED需要要求任意两个有效值之间hanming距离>=4

2.通过对有效数据插入控制位去进行冗余编码,在满足hanming距离要求时,调整bit序列使控制位存在于2的整幂次位上
利于使用2的整幂次位对对应有效数值进行二分法锁定

3.为了避免SEC控制位异或逻辑门延迟与DED控制位异或逻辑门延迟的不均衡,所以采用Hasio编码进行优化
即Odd-weight-column SEC-DED,通过逻辑变换,将DED控制位中不参与运算的异或逻辑去掉,使各控制位的产生异或逻辑门相对均衡,便于硬件实现

### hanming

| | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | d0 | d1 | d2 | d3 | d4 | d5 | d6 | d7 | d8 | d9 | d10 | p0 | p1 | p2 | p3 | p4 | SUM |
| R0 | 1 | 1 | | 1 | 1 | | 1 | | 1 | | 1 | 1 | | | | | 8 |
| R1 | 1 | | 1 | 1 | | 1 | 1 | | | 1 | 1 | | 1 | | | | 8 |
| R2 | | 1 | 1 | 1 | | | | 1 | 1 | 1 | 1 | | | 1 | | | 8 |
| R3 | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | 1 | | 8 |
| R4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 16 |

### Hasio

| | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | d0 | d1 | d2 | d3 | d4 | d5 | d6 | d7 | d8 | d9 | d10 | p0 | p1 | p2 | p3 | p4 | SUM |
| R0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | 1 | 1 | | | | | 8 |
| R1 | 1 | 1 | 1 | | | | 1 | | | | 1 | | 1 | | | | 6 |
| R2 | 1 | | | 1 | 1 | | 1 | 1 | | 1 | 1 | | | 1 | | | 8 |
| R3 | | 1 | | 1 | | 1 | 1 | | 1 | 1 | 1 | | | | 1 | | 8 |
| R4 | | | 1 | | 1 | 1 | | 1 | 1 | 1 | 1 | | | | | 1 | 8 |

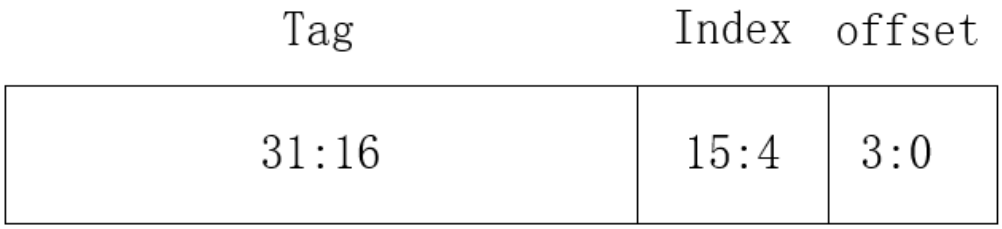
## 二.Cache配置

### I-Cache

- 容量64KB
- cache line大小16B(128bits,四字)
- 4K项
- 直接映射
- 预译码(00非分支,10无条件跳转,11有条件分支)
- 分为TagRAM和DataRAM以及eccEAM

### I-Cache/Fetch Interface

| 信号名 | 作用 | in/out |
| --- | --- | --- |
| PC[31:0] | 取指地址 | in |
| enable | 取指使能信号 | in |
| data[127:0] | 指令(最多四条有效) | out |
| ready | I-Cache是否准备好指令 | out |
| pre_decode[7:0] | 4指令(00非分支,10无条件,11有条件分支) | out |
| valid[3:0] | 对应指令是否有效(非对齐访问) | out |
| memory_error | 无法恢复的异常信号(DED) | out |




## D-Cache

- 容量64KB
- cache line大小8B(64bit,两字)
- 4K项
- 两路组相联
- 替换策略:PLRU
- 支持数据非对齐访问
- 分为TagRAM和DataRAM以及eccEAM

### D-Cache/LSU Interface

| 信号名 | 作用 | in/out |
| --- | --- | --- |
| address[31:0] | 数据地址 | in |
| byte_en[3:0] | 字节使能(字,半字,字节) | in |
| enable | 数据操作使能信号 | in |
| W/R | 读写信号 | in |
| SB_en | 并行访问SB和D-Cache之后,SB命中通知D-Cache不需要工作了 | in |
| data[31:0] | 数据值 | inout |
| ready | D-Cache是否准备好 | out |
| memory_error | 无法恢复的异常信号(DED) | out |

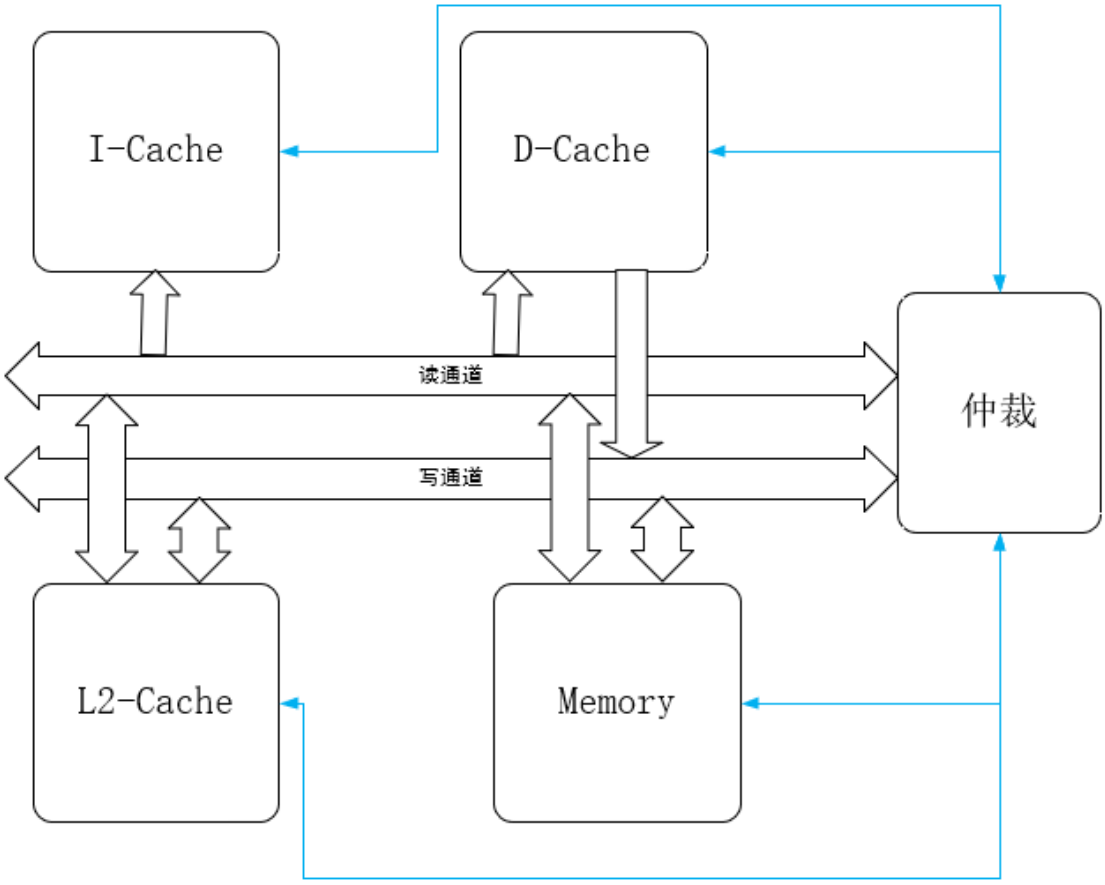
## L2-Cache

- 容量1MB
- cache line大小16B(128bits,四字)
- 16K项
- 四路组相联
- 替换策略:PLRU
- 写分配策略
- 分为TagRAM和DataRAM以及eccEAM

# 三.AXI总线

## 总述

1.基于AXI1.0总线协议的双通道总线,通过设计统一的接口管理实现内部共享总线传输
2.分离的地址/控制和数据相位使得地址请求和数据传输可以流水化并行执行
3.分离的读写数据通道,便于充分发挥数据传输并行性
4.采用基于突发的传输,master只提供起始地址,slave依据控制信号实现连续地址数据搬移,易于实现启动初始化操作
5.支持传输请求的非抢占式优先级仲裁
6.通过双向握手(valid/ready)保证数据传输完整性

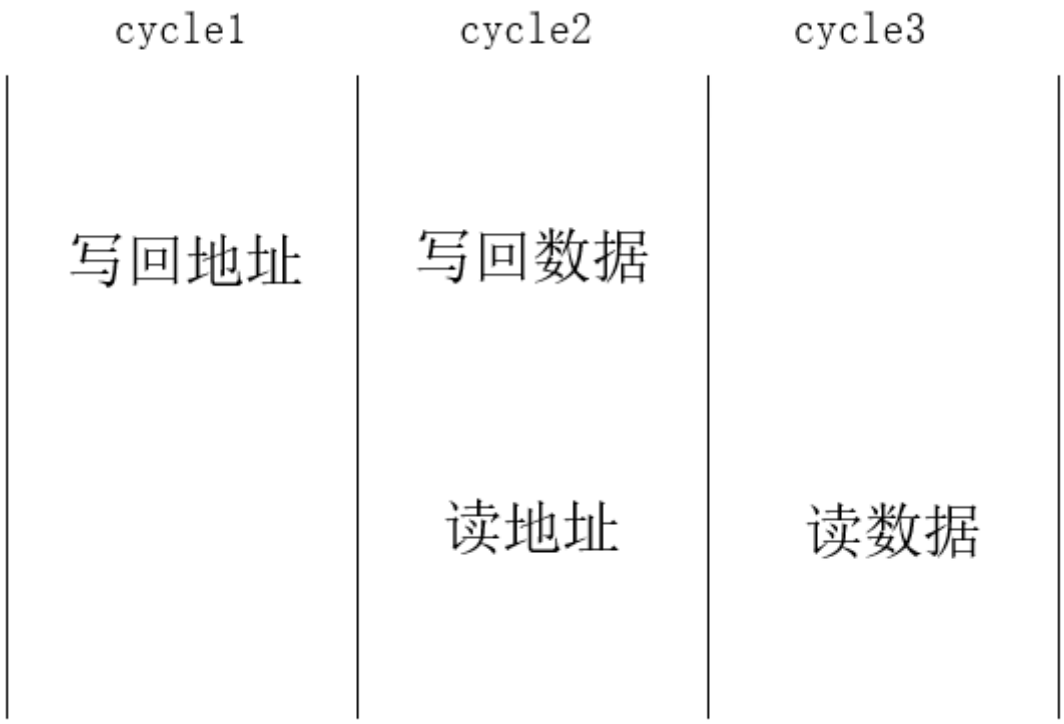



## 总线交互

主从关系:
I-Cache发起:L2-Cache响应
D-Cache发起:L2-Cache响应
L2-Cache发起:Memory响应

### 思路

- 因为I-Cache不允许修改,所以I-cache只会发起读申请
- 因为读写通道分离且流水化地址和数据传输,可以加速替换写回逻辑(D-Cache/L2-Cache)




- 因为实际运行中L1-Cache并非每周期都会有miss,所以对于总线的争用处于可容忍水平

### 优先级问题

| 优先关系 | 冲突通道 | 原因 |
| --- | --- | --- |
| D-Cache > I-Cache | 读通道 | 因为对于D-Cache代表的load指令会影响相关指令的唤醒<br>又因为对于I-Cache代表的取值因为I-buffer的存在减弱了影响 |
| L2-Cache > L1-Cache | 读/写通道 | 因为对于L2-Cache的请求只会因为L1-Cache |

# 四.总线信号

## 全局

| 信号 | 来源 | 描述 |
| --- | --- | --- |
| CLK | 时钟源 | 全局的时钟信号,同步电路(上升沿采样) |
| RESET | Reset源 | 全局复位信号,拉低有效 |

## 写地址通道信号

| 信号 | 来源 | 描述 |
| --- | --- | --- |
| AWID[1:0] | AWID[1]来自L2-Cache<br>AWID[0]来自D-Cache | 写地址发起者ID |
| AWADDR[31:0] | master | 写地址,在突发传输中为起始地址 |
| AWLEN[3:0] | master | 突发长度,传输次数 |
| AWSIZE | master | 突发大小,对应一次传输位宽<br>0传64bits(D-Cache)<br>1传128bits(L2-Cache) |
| AWVALID | master | 写地址信息是否有效,申请启动写入 |
| AWREADY | slave | 写地址是否准备好,是否可以接受写入 |

## 写数据通道信号

| 信号 | 来源 | 描述 |
| --- | --- | --- |
| WDATA[127:0] | master | 总线宽128bits |
| WLAST | master | 最后一次写,表示突发传输的结束 |
| WVALID | master | 写有效信号,表示此时数据有效 |
| WREADY | slave | 写准备好信号,表示本周期可以接收数据 |

## 写响应通道信号

| 信号 | 来源 | 描述 |
| --- | --- | --- |
| BRESP | slave | 写响应:OKAY,DECERR |
| BVALID | slave | 写响应有效 |
| BREADY | master | 响应准备好 |

## 读地址通道信号

| 信号 | 来源 | 描述 |
| --- | --- | --- |
| ARID[2:0] | ARID[2]:来源于L2-Cache<br>ARID[1]:来源于D-Cache<br>ARID[0]:来源于I-Cache | 读地址发起者ID |
| ARADDR[31:0] | master | 读地址,一次突发传输的首地址 |
| ARLEN[3:0] | master | 突发长度 |
| ARSIZE | master | 突发大小 |
| ARVALID | master | 读地址有效 |
| ARREADY | slave | 读地址准备好 |

### 读数据通道信号

| 信号 | 来源 | 描述 |
| --- | --- | --- |
| RDATA[127:0] | slave | 总线宽128bits |
| RRESP | slave | 读响应:OKAY,DECERR |
| RLAST | slave | 最后一次读,表示突发传输的结束 |
| RVALID | slave | 读有效信号,表示此时数据有效 |
| RREADY | master | 读准备好信号,表示本周期可以接收数据 |

## 五.情况处理

### 情况一

```
情况一:I-Cache/D-Cache hit则正常响应,不使用总线(0+1个cycle)
```

### 情况二

```
情况二:I-Cache miss,L2-Cache hit(D-Cache miss,L2-Cache hit且不用写回)(3+1个cycle)
    读地址通道:
        cycle1  :   I-Cache miss
        cycle2  :   向仲裁发出读申请并产生相关控制信号
                    仲裁尝试拉高读通道master:I-Cache,slave:L2-Cache
                    L2-Cache hit不回应则仲裁选定
        cycle3  :   L2-Cache产生RREADY信号开始读地址
    读数据通道:
        cycle3  :   仲裁确定后L2-Cache送出数据和产生Rvaild/Rlast信号
```

### 情况三

```
情况三:D-Cache miss,L2-Cache hit(需要写回)(4+1个cycle)
    写地址通道:
        cycle2  :   向仲裁发出写申请并产生相应控制信号
                    仲裁尝试拉高写通道master:D-Cache,slave:L2-Cache
                    L2-Cache hit不回应则仲裁确定
        cycle3  :   L2-Cache产生WREADY信号开始写
    写数据通道:
        cycle3  :   仲裁确定后D-Cache发送数据和产生相应信号
    写响应通道:
        cycle3  :   产生响应信号
    读地址通道:
        cycle3  :   向仲裁发出读申请并产生相关控制信号
                    仲裁尝试拉高读通道master:D-Cache,slave:L2-Cache
                    L2-Cache hit不回应则仲裁选定
        cycle4  :   仲裁确定后L2-Cache产生RREADY信号开始读
    读数据通道:
        cycle4  :   仲裁确定后L2-Cache送出数据和产生Rvaild/Rlast信号
```

### 情况四

```
情况四:D-Cache miss,L2-Cache miss(不需要写回)(I-Cache miss,L2-Cache miss同理)(5+1个cycle)
    读地址通道:
        cycle2  :   向仲裁发出读申请并产生相关控制信号
                    仲裁尝试拉高读通道master:D-Cache,slave:L2-Cache
                    L2-Cache miss并向仲裁发出读申请并产生相关控制信号
                    仲裁拉低D-Cache,拉高master:L2-Cache,slave:memory
        cycle3  :   仲裁确定后memory产生RREADY信号开始读
        cycle4  :   向仲裁发出读申请并产生相应控制信号
                    仲裁尝试拉高写通道master:D-Cache,slave:L2-Cache
                    L2-Cache hit不回应则仲裁确定
        cycle5  :   仲裁确定后L2-Cache产生RREADY信号开始读
    读数据通道:
        cycle3  :   仲裁确定后memory送出数据和产生Rvalid/Rlast信号
        cycle5  :   仲裁确定后L2-Cache送出数据和产生Rvaild/Rlast信号
```

## 其他情况

```
还有其他请况以及多情况并发
例如:
    D-Cache miss,L2-Cache miss(需写回,且写回位置miss)
    D-Cache miss,L2-Cache miss(需写回,且读位置miss)
    D-Cache miss,L2-Cache miss(需写回,且两个位置都miss)
    等...

基本原则就是一个周期的仲裁周期,根据定义的优先关系去指导仲裁电路仲裁
各部件内部冲突由自己本身内部信号生成部分处理串并行关系
```

# 进一步拓展(时间充裕时)

- 初始化模式(reset时)

  因为cache体系中没有初始化机制,所以对于取值和读数据都会有一个启动的过程,在这个过程中每次读cache都会miss,即强制不命中,所以在后期时间充裕的情况下可以加入一个reset后初始化cache的机制,通过监听AXI总线的突发传输机制实现整块数据搬移,对cache进行初始化操作,尽可能缓解cache的强制不命中操作.

- I-Cache预取

  通过AXI突发传输机制可以实现对于下一条指令块的预期而不会造成过多周期损耗,
  且在I-Cache中单独设置一个预取buffer,预取指令块先存于预取buffer,避免造成I-Cache污染,
  当之后在预取buffer命中时再搬移到I-Cache中.

- 更复杂的Cache替换策略

  ```
  例如:
      Next-Line Prefetching
      Fetch-Directed Instruction Prefetching,FDIP
      ...
  ```

- .....

# 解释

- 因为对于本存储系统而言在发出申请的时候地址空间也就准备好了,所以请求的地址和空间有效的信号会同时产生
- 因为我们设定I-Cache和D-Cache需要的数据是不重叠的即数据和程序分区,也不支持自修改程序,所以I-Cache和D-Cache之间的读写不会产生冲突,所以设计L2-Cache双端口(一个读一个写)不存在同时针对同一个位置的冲突,不需要考虑是否先写后读问题
- 因为对于正常的交互而言仅需要一个cycle就完成了传输,所以仲裁选定后相关信号都可以产生了
- 仲裁器严格按照申请优先级去拉高或拉低对应master和slave部件,各信号要严格符合AXI1.0协议的接口定义