

任务

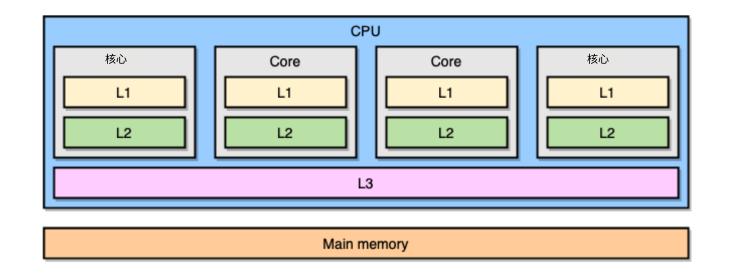
- •缓存行设计
- •验证缓存行
- •观察仿真波形

Chenlu Miao 11/2022

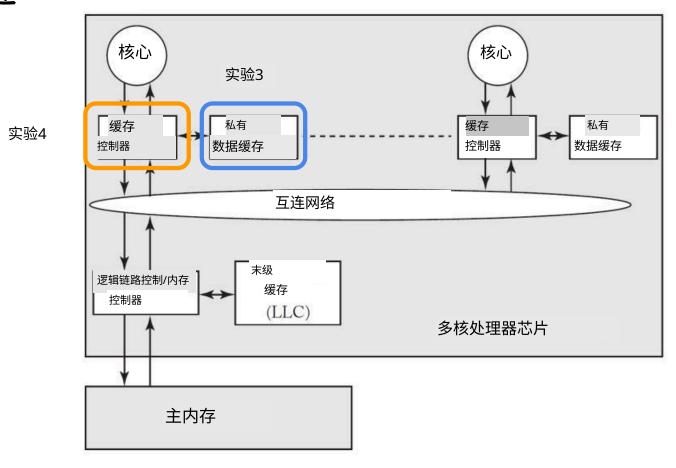
概述

- •缓存概述
- •配置
- •地址解码
- •缓存存储
- •访问缓存
- •模拟

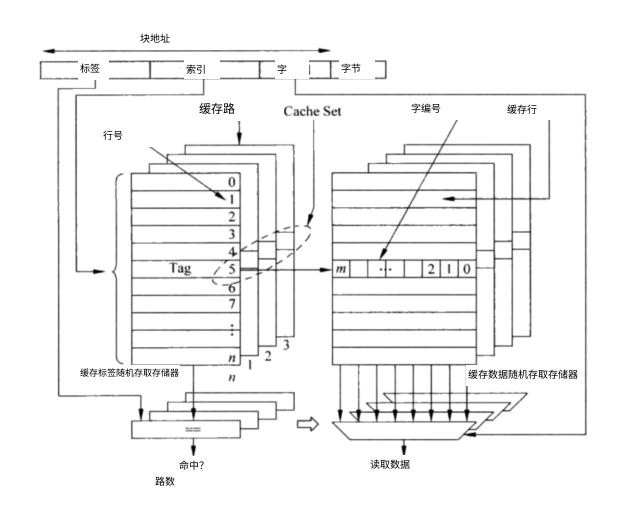
缓存概述



缓存概述



缓存概述



配置

- •块大小
- •块组织方式
 - •直接映射
 - •全相联
 - •组相联
- •块替换策略
 - O FIFO O LRU
 - •随机
- •写策略
 - •写回 •直写

配置

- •64条缓存行
 - •缓存行大小 =16 字节 =4字 =4*32 位
- •2路组相联
- •替换策略:最近最少使用(LRU)
- •写策略
 - •写回
 - •写分配

地址解码

•偏移量

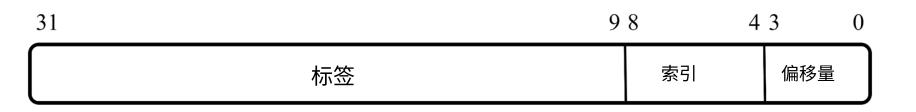
•log2(缓存行大小) = 4

位 - 索引

•

 $\log 2 \, (\# \text{cache line}) - \log 2 \, (\text{cache assoc}) = 5$

位 - 标签 - 32 - 偏移量 - 索引 = 23 位



缓存存储

•标签 -有效

•脏 - 最 近最少使 用 - 数据



缓存存储

•标签 -

有效

•脏位 - LRU

•数据

1位 1位 1位 4个单词 reg [ELEMENT_NUM - 1:0] 内部最近使用标志 = 0;

标签

数据

reg [ELEMENT_NUM-1:0] 内部有效标志 = 0; reg [ELEMENT_NUM-1:0] 内部脏标志 = 0;

有效

LRU

reg [TAG_BITS - 1:0] 内部标签 [0:ELEMENT_NUM - 1];

脏数据

寄存器 [31:0] 内部数据 [0:元素数量*元素字 - 1];

32 组;64 条缓存行

组i路0:i imes2+0 o内部最近使用 $[\{i,1'b0\}]$

设置 i 方式 $1:i \times 2+1 \rightarrow$ 内部最近 $[\{i,1'b1\}]$

设置 i 方式 j 单词 k:i imes 8+j imes 4+k
ightarrow 内部数据 $[\{{
m i},{
m j},{
m k}\}]$

访问缓存

模块 缓存(

输入线 时钟, // 时钟

输入线 复位, // 复位

输入线 [地址位 - 1:0] 地址, // 地址

输入线 load,// 读取刷新最近位

输入线存储,// 将有效位设为1并将脏位重置为0

输入线编辑,// 将脏位设为1

输入线无效,// 将有效位重置为0

输入线 [~2~:~0~] u_b_h_w, // s 选择有符号或无符号 ${\mathcal E}$ 数据宽度LB、LH、LW、LBU、LHU

输入线 [31:0] din,// 写入的数据

输出寄存器命中=0,//是否命中

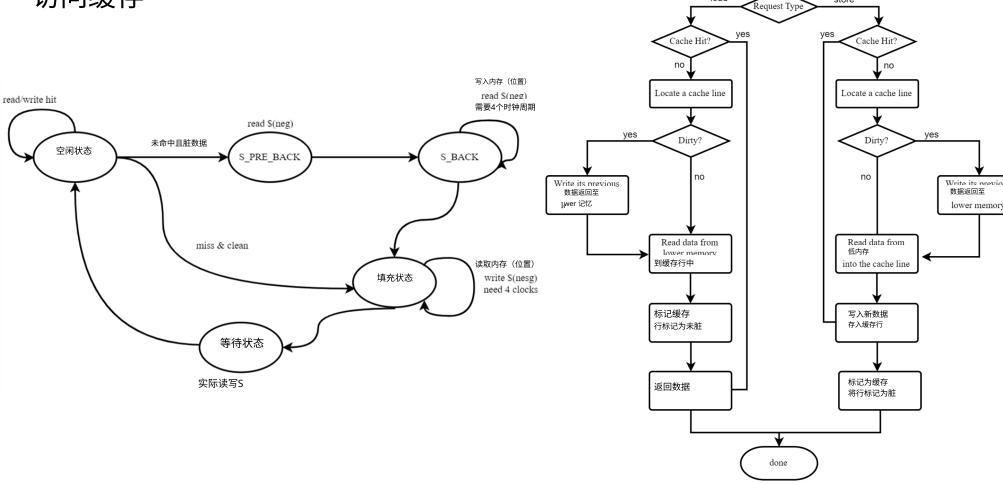
输出寄存器 [31:0] dout = 0, // 读出的数据

输出寄存器 valid = 0, // 有效位

输出寄存器 dirty = 0, // 脏位

输出寄存器 [TAG_BITS-1:0] tag = 0 // 标签位

访问缓存



Write its previou data back to

从.....读取数据

进入缓存行

Mark the cache

访问缓存

赋值 addr_tag = ... 赋值 addr_index = ...

山直 dddi_iiidex - ...

赋值 valid1 = inner_valid[addr_element1]; 赋值 valid2 = ...

。自 Valiuz - ...

赋值 tag1 = inner_tag[addr_element1]; 赋值 tag2 = ...

赋值 hit1 = valid1 & (tag1 == 地址标签);

赋值 hit2 = ..._______

... 命中 <=...

始终 @ (时钟信号的上升沿) 开始

结束

Locate a cache li

从.....读取数据

into the cache li

写入新数据

Write its previou data back to

访问缓存

始终(在时钟信号上升沿)开 始…如果(处于编辑状态)开

始如果(命中条件1满足)开始

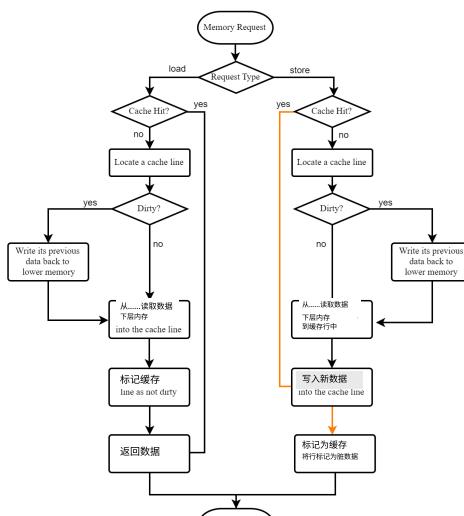
内部数据[地址字1] <= ...内部脏数据标志 [地址元素1] <= 1'b1;内部最近使用标志[地址元素1] <= 1'b1;内部最近使用标志[地址元素2] <= 1'b0;

结束

否则如果(命中条件2

满足)…结束

end



定位一个缓存行

从.....读取数据

lower memory into the cache line

Mark the cache 将该行标记为未修改

返回数据

data back to lower memory 定位一个缓存

lower memory into the cache line

Write new data

写入其先前内容

data back to lower memory

访问缓存

始终 @ (时钟信号上升沿) 开始

如果 (加载) 开始

如果 (命中1) 开始 数据输出 <=...

> 内部最近项[地址元素1] <= 1'b1; 内部最近项[地址元素2] <= 1'b0;

结束

否则如果(命中2)...

^{结束} 否则...

否贝 ...

结束

访问缓存

赋值 recent1 = inner_recent[addr_element1]; 赋值

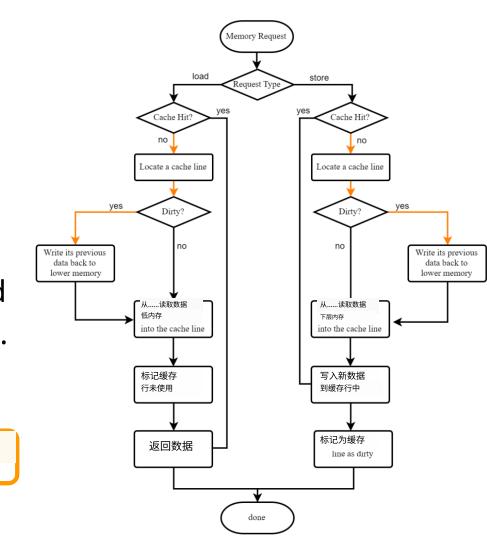
recent2 = ... 赋值 dirty1 = inner_dirty[addr_element1]; 赋值 dirty2 = ... 始终在时钟上升沿开始 valid <= recent1 ? valid2 : valid1; dirty <= ... tag <= ... hit <= ... 如果 (load) 开始 ...

否则 dout <= inner_data[recent1?

addr_word2 : addr_word1]:

结束

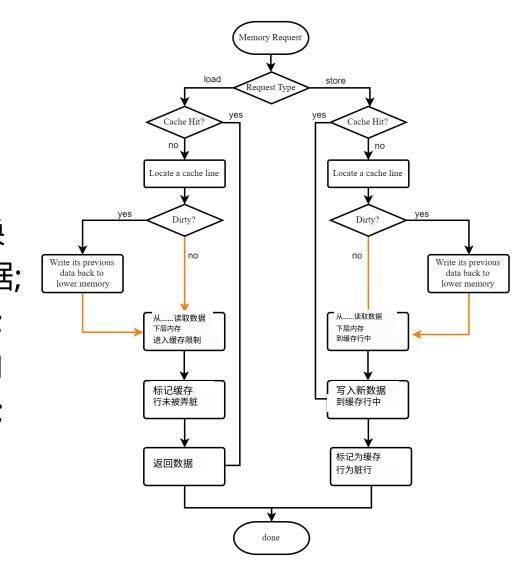
用于回写



访问缓存

始终 @ (时钟上升沿) 开始...如果 (存储) 开始

如果 (最近使用1) 开始 // 替换 2内部数据[地址字2] <= 输入数据; 内部有效位[地址元素2] <= 1'b1; 内部脏位[地址元素2] <= 1'b0;内部标签[地址元素2] <= 地址标签; 结束 否则 开始…结束结束…



仿真

// 初始化
32'd10: 开始
 加载 <= 0;
 存储 <= 1;
 编辑 <= 0;

数据输入(din)小于等于32位十六进制数1111111;
 地址(addr)小于等于32位十六进制数00000004;
end

32位十进制数11: 开始
 地址(addr)小于等于32位十六进制数0000000C;
结束

32位十进制数12: 开始
 地址 <= 32位十六进制数00000010;
结束

地址 <= 32位十六进制数0000014;

// 读取未命中 32'd14: 开始 加载 <= 1;

存储 <= 0; 编辑 <= 0;

u_b_h_w 小于等于 3 位二进制数 010; din 小于等于 0;

addr 小于等于 32 位十六进制数 00000020;

// 读命中

32 位十进制数 15: 开始

u_b_h_w 小于等于 3 位二进制数 010; 地址小于等于 32 位十六进制数 00000010;

结束

仿真

end

// 读取组0的第0行,组 最近的位 32位值18: 开始 加载 <= 1; 存储 <= 0; 编辑 <= 0; u_b_h_w <= 3位二进制010; 数据输入小于等于0; 地址小于等于32位十六进制数00000004; 结束 // 存储到组0的行1 由于行0最近使用 32位十进制数19: 开始 加载 ≤ 0 ; 存储 <= 1; 编辑 <= 0;

u_b_h_w <= 3位二进制数010; 数据输入 <= 32位十六进制数33333333; 地址 <= 32位十六进制数00000204;

结束

仿真

// 编辑组0的第1行,设置脏位并 更新最近使用位 32'd20: 开始 加载 <= 0;存储 <= 0;编辑 <= 1; u_b_h_w <= 3'b010;数 据输入 <= 32'h4444444;地址 <= 32'h00000204;结束//读取 组0的第0行,设置

最近使用位 32'd21:开始加 载 <= 1;存储 <= 0;编辑 <= 0;

32位十进制数13: 开始

结束

u_b_h_w <= 3'b010; 数据输入 <= 0; 地址 <= 32'h00000004; 结束 // 读取未命中,标签不匹配。 输出标签(第1行的),有效 并且脏位等于1 32位十进制数22: 开始 加载 <= 1; 存储 <= 0; 编辑 <= 0;

> u_b_h_w <= 3位二进制数010; ^{数据输入 (din) 小于等于32位十六进制数0;}

地址(addr)小于等于32位十六进制数00000404;

// 自动替换第0组的第1行 32位十进制数23: 开始 加载 <= 0; 存储 <= 1; 编辑 <= 0;

> u_b_h_w <= 3位二进制数010; 数据输入 <= 32位十六进制数5555555; 地址 <= 32位十六进制数00000404;

结束

模拟



仿真

名称	值				-															
量 (3)	1																			
是 182	0																			
琴计数器[31:0]	0000000a	00000013 00000014		_	00000015		00000016		00000017		00000018		0019	0000001a	集	钟; 专心	0000001c 00000			
量和效	0																			
U商店	0																			
是0厘米	0																			
L匹配该	0																			
) 琴 u_b_h_w{2:0}	0	2																		
甲 300/310	00000000	000000	00000004 0000				0000	00000004			00404			0000000						
, \mathbb{D} dim $[3,1,0]$	00000000	000000	00	333333333 直觉		ī觉		0101016	0 60		55555555						00000000			
十分钟后	0									ı										
W 有效	0			\lnot	$\neg \vdash$		_									\top				
所说的	0				$\neg \vdash$									1						
学校在校	000000	000000									0000	01	1 000002							
解 道格31.01	0000000	000 / 11111111 / 00000000 / 33333333 / 11111111								1111	$\overline{}$	44444	444444 DISDIGIGIG							
甲 单词1 [310]	00000000		实用性														00000000			
甲病房2[31,0]	00000000	0.05500000 333333							444	44 \$5555		5555					布拉兹穆恩			
,即 标签122.0	000000											000000								
琴 标签2[220]	000000	000000					000	000001			<u> </u>					01	1000{Z			
最近	0															\top				
16个最近的2	0																			
1å hit1	0															Ь				
¼ hit2	0															0				
无效	0																			
1个有效的2	0																			
像脏东西1	0																			
16个脏东西2	0																			

参考文献

- •超标量处理器设计
- •现代处理器设计