

# 嵌入式 GPU 中二级高速缓存的设计与实现

杜慧敏, 杨超群, 季凯柏

(西安邮电大学 电子工程学院, 陕西 西安 710121)

**摘 要:** 针对嵌入式 GPU 与主存之间进行数据交互时出现速度不匹配的问题, 设计了一种适用于嵌入式 GPU 的二级高速缓存 Cache 控制器. 二级 Cache 控制器采用四路组相联的映射结构, 使用伪最近最少使用 (Pseudo\_LRU) 替换算法, 可以管理 16~512 kB 的二级高速缓存. 实验结果表明, 当选取 Cache 大小为 128 kB 时, Cache 的命中率达到 71.12%.

**关键词:** 嵌入式 GPU; 高速缓存控制器; Rseudo\_LRU 算法

**中图分类号:** TP399

**文献标识码:** A

**文章编号:** 1000-7180(2018)02-0094-06

## Design and Implementation of Embedded GPU Cache Controller

DU Hui-min, YANG Chao-qun, JI Kai-bo

(School of Electornic Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China)

**Abstract:** A suitable for embedded GPU cache controller is designed for the speed mismatch problems data interaction between main memory and embedded GPU. The cache controller adopts a four-way set associative mapping structure, uses pseudo-Least Recently Used replacement algorithm. The size of CacheSRAM can be configured, which configuration range is 32~512 kB. The experimental results show that when the CacheSRAM capacity is 128 kB, hit rate of the graphics application's Cache can reach to 71.12%, the cache controller can further enhance the overall performance of embedded GPU.

**Key words:** embedded GPU; cache controller; Rseudo\_LRU algorithm

### 1 引言

随着便携设备迅速的发展和用户对高质量图形渲染的需求, 用于绘制图形的嵌入式 GPU 得到了蓬勃的发展<sup>[1]</sup>. 嵌入式系统如智能手机终端都包含 3D 游戏和绚丽的用户界面, 嵌入式 GPU 已经成为便携设备的嵌入式系统中不可或缺的一部分<sup>[2]</sup>.

嵌入式 GPU 相较于嵌入式 CPU 拥有更多的运算单元, 需要存储通路提供更大的带宽, 这使得嵌入式 GPU 的整体性能更加依赖于主存. 由于主存速度的提高始终无法赶上嵌入式 GPU 计算能力的发展, 在 GPU 和主存之间的高速缓冲控制器已成为解决图形处理器与存储器系统之间速度差距的有效方法<sup>[3]</sup>.

本文主要研究适应于嵌入式 GPU 的高速 Cache 设计与实现, 提出一种 Cache 大小可以配置的控制器的设计方案. 本文所设计的 Cache 控制器采用一个四路组相联的映射结构, 使用伪最近最少使用替换算法, 可以管理 16~512 kB 的 Cache 存储块, 在 Xilinx 公司的 ZC706 开发板上测试, Cache 的命中率达到 71.12%.

### 2 嵌入式 GPU 框架总体结构

嵌入式 GPU 主要由几何处理器、像素处理器、二级高速缓存和内存管理单元组成, 嵌入式 GPU 的系统结构如图 1 所示. 几何处理器的任务是产生图元列表. 像素处理单元的任务是将图元列表变换形成最终在显示器上显示的图像. 存储管理单元的

任务是将地址翻译和内存保护,几何处理器和像素处理器对内存的访问都要经过内存管理单元。

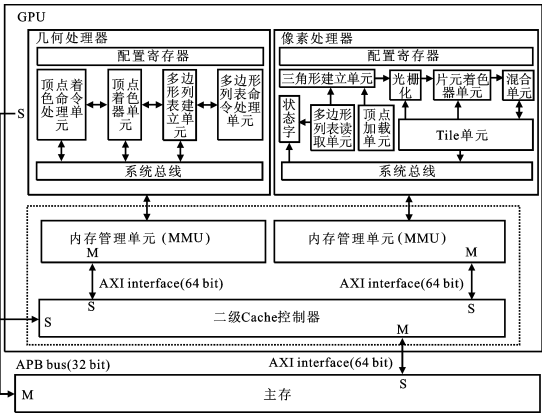


图 1 嵌入式 GPU 的系统结构

在主存和嵌入式 GPU 之间增加了合适的二级高速缓存控制器,可以使 GPU 不用直接访问主存,而是与高速缓存控制器交换数据。这样就避免了速度不匹配问题,提升了图形渲染速度,从而充分发挥嵌入式 GPU 的整体性能,使用户有更加优良的视觉体验。

3 Cache 的主要参数设计

3.1 Cache 的映射方式

Cache 的映射方式主要有直接映射、组相联映射和全相联三种方式。为了研究三种不同映射方式对系统性能的影响,配置 Cache 的容量大小为 4 kB,测试了相联度对系统性能的影响<sup>[4]</sup>,结果如图 2 所示。

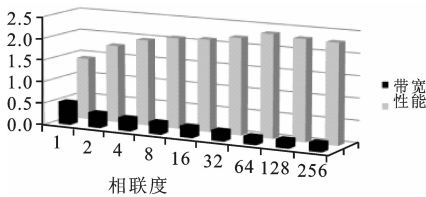


图 2 相联度对性能及带宽需求的影响

由图 2 可知,相联度越高,系统的性能也越高,但 Cache 的设计也就越复杂,从而导致面积和功耗的增加<sup>[5]</sup>。

直接映射是最简单的 Cache 映射方式,但其性能不高;全相联映射 Cache 在小容量时的性能较好,但该映射方式产生的功耗较大;通过对嵌入式 GPU 性能、设计复杂度和功耗的综合考虑,本文设计的 Cache 采用组间直接相联,组内全相联的四路组相联映射方式。

3.2 Cache 的读写策略

当系统需要访问的数据或指令在 Cache 中命中或缺失时,需要一种读写策略来完成 Cache 的读写操作。Cache 的读策略主要有读旁路策略和读贯穿策略两种方式。Cache 的写策略对 Cache 组织有较大影响,因此 Cache 的写策略需要考虑写命中和写缺失时的策略。对于写命中,有写穿透和写回两种策略。对于写缺失,有写分配和写不分配两种策略<sup>[6]</sup>。

Cache 对几何和像素处理器发送的读操作采用读旁路和读贯穿两种不同的读策略进行处理。几何处理器处理的顶点数据因为在几何处理器处理后直接进入图形管线下一级处理,因此在读 Cache 不命中的时候,可以旁路 Cache,将顶点数据送入到顶点处理器进行处理。对于像素处理器发送到 Cache 的读操作,采用读贯穿策略。

对于嵌入式 GPU 中几何和像素处理器发送到 Cache 进行写操作,为了保持 Cache 和内存中数据的一致性,当写 Cache 命中时采用写穿透策略,处理器将数据同时写入 Cache 块和内存中;当写 Cache 缺失时,采用写不分配策略,处理器直接将数据写入内存<sup>[7]</sup>。

3.3 Cache 的替换算法

嵌入式 GPU 从 Cache 中读取数据时,如果需要的数据不在 Cache 中存储,则需要将所需的数据存储到 Cache 中,但 Cache 中的空间已经完全被占满时,这时就产生了替换策略的问题<sup>[7]</sup>。替换算法的合理选择对整个 Cache 的性能起到至关重要的作用。

Cache 中常用的替换算法有先进先出(FIFO)算法、最近最少使用(LRU)算法以及随机(Rand)算法。Cache 容量比较小时 LRU 最好,FIFO 替换算法要比 Rand 算法性能略好;Cache 容量较大时 LRU 跟随机替换算法几乎没有差别。总体上讲,Cache 容量越大,相联路数越多,Rand 算法跟 LRU 算法的性能差不多<sup>[8]</sup>。

近似的 LRU 算法主要有改进伪最近最少使用(MPLRU)和伪最近最少使用(PLRU)算法<sup>[9]</sup>。在 Cache 设计中,采用四路组相联的映射方式中,MPLRU 算法比 PLRU 算法 Cache 缺失次数下降 8.52%<sup>[10]</sup>,但增加面积产生了额外的功耗。由于功耗是衡量嵌入式 GPU 性能的重要指标,在综合考虑功耗和性能的前提下,本文设计的 Cache 采用 PLRU 算法。

3.4 Cache 的存储结构

由于 Cache 控制器管理的 CacheSRAM 大小可配置,因此需考虑 Cache 的存储结构参数与主存大小2W KB、Cache 大小2X KB、Cache 块大小2Z字节和相联度为2Y的关系. 访问 Cache 的 Tag 标签、Index 索引以及 TagSRAM 地址位宽的计算如式(1)~(3)所示,其中 W,X,Y 和 Z 的取值为正整数.

Tag = log<sub>2</sub>( $\frac{\text{主存大小} * \text{相联度}}{\text{Cache 的大小}}$ ) =  
log<sub>2</sub>( $\frac{2^W * 2^Y}{2^X}$ ) = W + Y - X (1)

Tagaddr = log<sub>2</sub>( $\frac{\text{Cache 的大小}}{\text{Cache 块的大小} \times \text{相联度}}$ ) =  
log<sub>2</sub>( $\frac{2^X}{2^Z \times 2^Y}$ ) = X - Z - Y (2)

Index = log<sub>2</sub>( $\frac{\text{Cache 的大小}}{\text{Cache 块的大小}}$ ) =  
log<sub>2</sub>( $\frac{2^X}{2^Z}$ ) = X - Z (3)

Cache 存储器主要包括 TagSRAM(标识存储器)和 CacheSRAM(Cache 数据存储器). CacheSRAM 采用四路组相联的映射结构,Cache 块为25字节. 根据 W=32,Y=2,Z=5 化简式(1)、(2)和(3)得到计算 Cache 存储器的相关参数表达式:

Tag = 34 - X  
Tagaddr = X - 7  
Index = X - 5 (4)

3.4.1 TagSRAM 存储结构设计

每个存储单元包含四个 Cache 块的标识 Tag (访问 Cache 地址的高位)、错误标志位(err)、有效状态位(valid)和该组 Cache 的替换状态(lru). TagSRAM 每个单元的存储结构如图 3 所示.

| Tag3          |      |      | Tag2          |        |        | Tag1          |        |       | Tag0          |  |
|---------------|------|------|---------------|--------|--------|---------------|--------|-------|---------------|--|
| addr3[32:X-2] |      |      | addr2[32:X-2] |        |        | addr1[32:X-2] |        |       | addr0[32:X-2] |  |
| err3          | err2 | err1 | err0          | valid3 | valid2 | valid1        | valid0 | lru   |               |  |
| [14]          | [13] | [12] | [11]          | [10:9] | [8:7]  | [6:5]         | [4:3]  | [2:0] |               |  |

图 3 TagSRAM 每个存储单元的结构

由图 3 和式(4)的 Tag 表达式可知,TagRAM 每个单元需要存储的数据宽度为:

TagWIDTH=(4 \* Tag)+15=  
4 \* (32-X)+15=151-4 \* X (5)

根据式(4)和式(5)的 Tagaddr 和 TagWIDTH 的表达式可知,TagSRAM 的总体存储结构如图 4 所示.

| 数据宽度(TagWIDTH) | 地址宽度(Tagaddr) |
|----------------|---------------|
| [151-4*X:0]    | 0             |
| [151-4*X:0]    | 1             |
| .....          | .....         |
| [151-4*X:0]    |               |

图 4 TagSRAM 的总体存储结构

3.4.2 CacheSRAM 存储结构设计

CacheSRAM 的存储结构主要由 Cache 块大小和式(2)的 Index 参数决定. CacheSRAM 的每个存储单元就是一个 Cache 块,Cache 块大小为 32 字节. 每个 Cache 块包含 8 个小 Cache 块,每个小 Cache 块为 4 个字节且每个小 Cache 块都有读写使能信号 we. 根据读写使能信号 we,可以实现对每个小 Cache 块的读写操作,提高 CacheSRAM 的利用率. CacheSRAM 的总体结构如图 5 所示.

| we[7]  | we[6]  | we[5]  | we[4]  | we[3]  | we[2]  | we[1]  | we[0]  | 写使能<br>(Cacheaddr) |
|--------|--------|--------|--------|--------|--------|--------|--------|--------------------|
| [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | 0                  |
| [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | 1                  |
| .....  | .....  | .....  | .....  | .....  | .....  | .....  | .....  | .....              |
| [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | [31:0] | 2 <sup>(n-5)</sup> |

图 5 CacheSRAM 的总体存储结构

由图 4 和图 5 的 TagSRAM 总体存储结构和 CacheSRAM 总体存储结构可知,为了实现 32~512 kB 的 CacheSRAM 配置,只需要修改 X 值,X 取值范围为 15~19 之间的正整数.

4 Cache 控制器电路设计

Cache 控制器是一个四路组相联的映射结构,使用伪最近最少使用(pseudo\_LRU)替换算法,可以管理 16~512 kB 的二级高速缓存 RAM. Cache 控制器使用 AXI 总线与图形处理器的其他模块互连,AXI 从端口传输的数据宽度为 64 位,AXI 主端口传输的数据宽度可配置为 64 位或 128 位. Cache 控制器的总体硬件架构如图 6 所示.

Cache 控制器主要包含仲裁器模块、标志访问模块、CacheSRAM 模块、Cache 标识模块、重排队列模块、访问路由模块、Cache 行匹配模块、写缓存模块和读缓存模块. 图中虚线表示各个模块之间的控制通路,实线表示数据通路.

4.1 仲裁器模块设计

当内存有多个读写请求进入 Cache 控制器时,需要设计一个仲裁器来确保进入 Cache 的访问请求能够按设计的优先级进行处理,择出当前需要处理

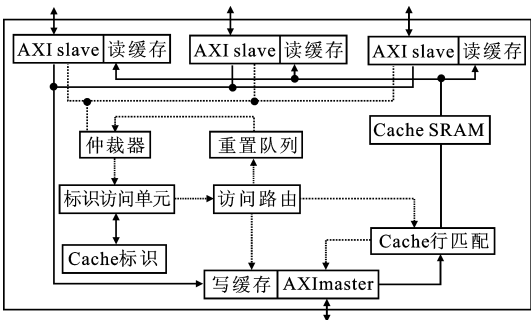


图 6 高速缓存控制器的总体硬件架构图

的请求。

仲裁器模块主要由 AXI 总线的读写地址的仲裁( *axi\_arb* )、循环调度( *MUX1* )模块、时间标识产生( *Timestamp* )模块和选择( *MUX2* )模块组成。具体的电路设计结构如图 7 所示。

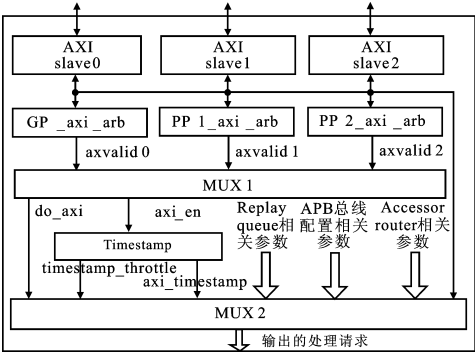


图 7 仲裁器模块的设计框图

读写地址仲裁( *axi\_arb* )模块,根据几何处理器和两个像素处理器对应的 AXI 总线输入的读写地址等相关信息进行仲裁,选择出当前每组 AXI 总线上有效的读写请求地址产生相应的读写信号。

循环调度( *MUX1* )模块,根据读写地址仲裁模块输出的读写信号,对几何和像素处理器的 AXI 从端口输入请求进行选择。复位完成后,首先优先处理 AXI 从端口 slave2 的请求。然后开始依次处理几何、像素 1 和像素 2 处理器的读写请求,直到所有 AXI 从端口的请求全部处理完毕。

时间标识模块是二级 Cache 控制器中管理活锁( *Live-Lock* )风险的一种安全机制。如果某个请求访问 Cache 发生缺失,这个请求操作长时间无法完成,这时会阻塞 Cache 行匹配模块的其他请求,直到该 Cache 缺失请求操作完成。

根据优先级选择( *MUX2* )模块首先处理访问路由模块输入进来的请求;其次是处理 Cache 行匹配模块输入的请求;最后是交替处理 AXI 总线从端口

与重排队列模块输入的请求。

4.2 标识访问模块

标识访问模块采用一个五级流水线结构。该五级流水线包含:读取 *Tag\_ram*、产生反馈信号、处理非缓存和缓存的读缺失操作、输出处理后的结果和数据缓存。该模块主要根据仲裁器模块输入的不同请求,完成标识( *Tag* )的匹配和 *Tag\_ram* 中标识的更新。

流水线的第一阶段主要是根据读地址与读使能信号,将 *Tag\_ram* 中对应的 *tagword* 读取出来。由于 cache 采用四路组相联的映射结构, *Tag\_ram* 中每个 *tagword* 包含四个 cache 块的标识 *tag*、错误标志位、有效状态位和该组 cache 的替换状态。每个 *tagword* 的具体结构如图 8 所示。

|         |         |         |         |      |      |      |      |        |        |        |        |       |
|---------|---------|---------|---------|------|------|------|------|--------|--------|--------|--------|-------|
| addr3   | addr2   | addr1   | addr0   | err3 | err2 | err1 | err0 | valid3 | valid2 | valid1 | valid0 | lru   |
| [90:72] | [71:53] | [52:34] | [33:15] | 14   | 13   | 12   | 11   | [10:9] | [8:7]  | [6:5]  | [4:3]  | [2:0] |

图 8 tagword 的结构

流水线的第二阶段主要完成输入请求信号的缓存和产生反馈信号。为了避免多次间隔很近的请求,访问 *Tag\_ram* 中的同一位置,造成从 *Tag\_ram* 中读取到旧的 *tagword*。该阶段会根据当前输入请求的地址 *addr1* 与前三个时钟周期 *arbiter* 模块输入的请求地址 *add2*, *addr3* 和 *addr4* 进行比较产生反馈( *feedback* )信号。

流水线的第三阶段主要完成对 *Tag\_ram* 中数据的更新和替换。当 cache 读缺失时,cache 的替换机制采用 PLRU 算法。首先,将输入的 *Tag* 信号与 *Tag\_ram* 读出的 *tagword* 的 *addr0*, *addr1*, *addr2* 和 *addr3* 块标识依次匹配,产生 *hit* 信号;然后,根据 *hit* 和 *tagword* 中各个块对应的 *valid* 和 *error* 信号确定被匹配的 cache 块;最后 *choose* 模块根据输入的访问类型( *access\_type* )和需要更新的 cache 块的标号( *line\_to\_validate* )来完成更新输入的 *tagword*,输出更新后的 *tagword\_out* 和匹配命中的相关信号。

流水线的第四阶段主要将 *tagword\_out* 和匹配命中的访问信息输出到访问路由模块,同时将 *tagword\_out* 写到 *Tag\_ram*。流水线的第五阶段主要完成 *tagword* 的缓冲,用于产生流水线第二阶段的反馈信号。

4.3 访问路由模块

访问路由模块根据不同的指示信号将接收内存请求发送到不同的模块,其中包括重置队列模块、

Cahce 行匹配模块、Cache SRAM 存储、写缓存模块和读缓存模块。

访问路由模块主要由 Eight-Buffer 模块和 Element queue 模块组成。Eight\_Buffer 模块包含 8 个元素，用于检测数据冲突。Element\_queue 模块主要用于匹配 Arbiter 模块与 Access\_router 模块之间的处理速度，该模块的设计结构如图 9 所示。

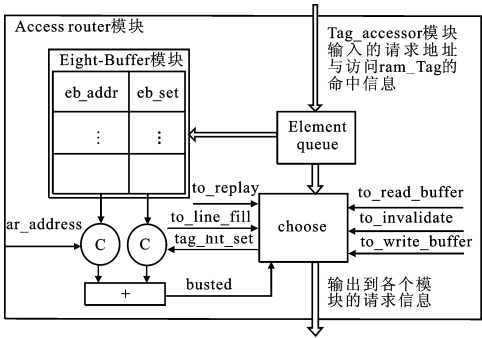


图 9 访问路由模块设计结构

访问路由模块根据 Tag\_accessor 模块输入的请求地址和匹配 cache 块的相关信息产生不同的判决条件。这些判决条件将不同的请求发送到不同的模块进行处理，具体的条件有：发生 cache 读缺失，发生 cache 读缺失且需要的数据在总线传输时发生错误，发生 cache 写缺失和几何和像素处理器的读请求在 Tag\_ram 匹配命中。

4.4 cache 行匹配模块

Cache 行匹配模块主要缓存读缺失的地址和需要替换的 Cache 块号，采用先进先出的原则对读缺失地址进行处理，根据缺失地址从内存读取回缺失的数据，并存储到相应的 Cache 块。该模块的设计结构如图 10 所示。

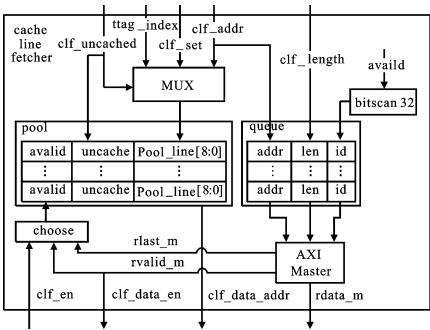


图 10 cache 行匹配模块设计结构

由图 10 可知，当发生 cache 读缺失时，pool 模块用于存储访问路由模块输入需要替换的 Tag\_ram 的地址 (clif\_addr[21:6]) 和 cache 块号 (clif\_

set)，由于 Cache 控制器最多能支持 32 个 AXI 从端的未处理请求，pool 模块可以缓存 32 个 cache 读缺失请求。queue 模块用于存储读缺失请求的地址 (clif\_addr) 和需要从内存读取回字节的长度 (clif\_length)，该模块包含 8 个元素。

pool 模块存储替换的 cache 块有效 (即 avalid) 时，读取 queue 模块存储的读缺失地址，发送缺失地址到 AXI 总线上，从内存中读回缺失的数据，数据写入到 CacheSRAM 中，数据传输完成后，pool 模块对应的数据标记为无效。读缺失的地址 (clif\_data\_addr) 发送到 Access\_router 模块进行再次匹配。

4.5 重置队列模块

重置队列模块主要完成对 cache 读缺失请求和坏命中请求的缓存。重置队列模块包含一个队列，该队列包含 32 个元素，结构如图 11 所示。

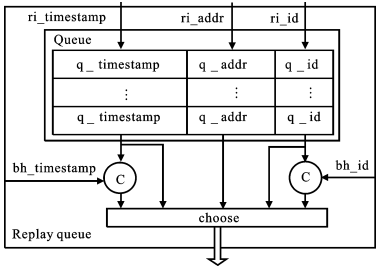


图 11 重置队列模块设计结构

当 ri\_enable 信号有效时，访问路由模块选择输出读缺失或坏命中的请求，存储到重置队列模块的队列里。对于路由访问模块处理的请求，要先与重置队列模块队列中的所有元素进行比较。若匹配命中，表示重置队列模块当前处理的请求无效，输出坏命中信号，坏命中是指 cache 中的数据有效，但总线有错误。若匹配缺失，表示重置队列模块当前处理的请求有效。缺失数据写入到 CacheSRAM 中时，队列中的元素会按先进先出的原则进入仲裁模块，进行匹配处理。

4.6 读缓存模块与写缓存模块

读缓存模块完成的主要功能是根据读命中请求的地址从内存读取数据传递给处理器。写缓存模块是根据写地址将数据经过 AXI 总线写入到内存中。

5 系统级验证

本文设计的高速缓存模在 Xilinx 公司开发的 ZC706 开发板上进行系统级验证，用 OpenGL ES 开发的场景对 Cache 的进行了测试。

通过配置不同 CacheSRAM 的大小和对不同复

杂度场景的绘制来测试 Cache 的命中率,结果如图 12 所示.

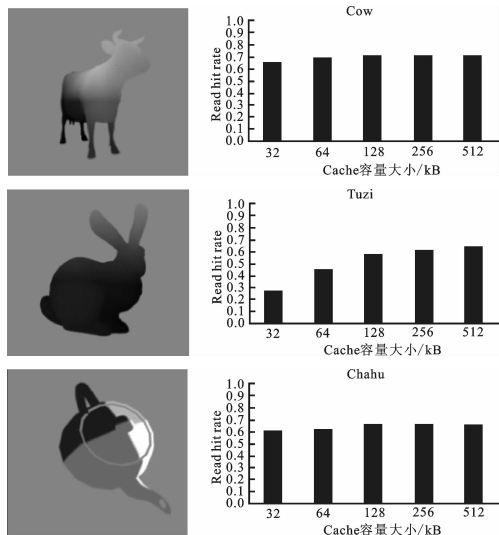


图 12 不同复杂度场景命中率

茶壶模型由 1 024 个小三角形构成,当 CacheSRAM 的容量为 128 kB 时,Cache 的命中率达到最高,当 CacheSRAM 容量大于 128 kB 时,由于图形应用程序需要绘制的顶点数据量小,使得 CacheSRAM 的容量过大,造成命中时间过长,从而 Cache 的命中率有所下降.牛模型采用 5 804 个三角形,兔子模型采用 69 451 个三角形,CacheSRAM 的容量从 128 kB 变化到 512 kB 时,Cache 的命中率虽然有所提高,但 Cache 的命中率提高不到 2%.

## 6 结束语

本文针对图形处理数据的特点,设计并实现了一种适合嵌入式 GPU 二级 Cache 结构,实现主存和 GPU 的数据交换.测试结果表明所设计 Cache 能在合适的参数配置下,命中率为 71.2%.

由于整个嵌入式 GPU 的数据通路较为复杂和时间精力等因素的限制,整个设计可以在以下几个方面进行优化来提高 Cache 的命中率.

(1)优化硬件电路:深入了解图形处理器与存储系统之间的数据交互和协同过程,优化高速缓存控制器中各个模块的具体硬件实现.

(2)编写复杂的图形应用程序:本文采用简单场景的图形应用程序,结果具有局限性.后期编写复杂场景的图形应用程序,进一步完善对 Cache 性能测试.

## 参考文献:

- [1] 史鸿声.可信嵌入式 3D 图形系统关键技术的研究[D].合肥:中国科学技术大学,2008.
- [2] 雷宇.移动设备上基于图像的图形绘制系统若干关键技术的研究[D].杭州:浙江大学,2005.
- [3] 张绪冰.基于高性能 VIM 体系结构的嵌入式存储系统的研究与实现[D].长沙:国防科学技术大学,2004.
- [4] Steve Furber. ARM SoC 体系结构[M].北京:北京航空航天大学出版社,2002.
- [5] 王思瑶,樊晓桢,肖楠.一种微处理器二级 Cache 的优化设计[J].科学技术与工程,2008,7(9):2356-2359.
- [6] 温泉.16 位高性能 CPU 的 Cache 设计[D].中国地质大学(北京),2014.
- [7] 隋秀峰,吴俊敏,陈国良.同时多线程处理器上的 Cache 性能分析与优化[J].小型微型计算机系统,2009,30(1):159-163.
- [8] J Hennessy, DPatterson. ComputerArchitecture: A Quantitative Approach [M]. 3rd ed. SanFrancisco: Morgan Kaufmann Publishers, 2003.
- [9] Jeong J, Dubios M. Cost-Sensitive Cache Replacement Algorithms[C]//The Ninth International Symposium on High-Performance Computer Architecture (HPCA-9'03). New Orleans, USA,2002.
- [10] Ghasemzadeh H, Mazrouee S S, Kakoe M R. Modified Pseudo LRU Replacement Algorithm[C]// IEEE International Symposium and Workshop on Engineering of Computer Based Systems, China, IEEE Xplore, 2006:6.

## 作者简介:

杜慧敏 女,(1966-),博士.研究方向为集成电路设计、计算机体系结构. E-mail:228660529@qq.com.

杨超群 男,(1991-),硕士.研究方向为计算机体系结构.

季凯柏 男,(1990-),硕士.研究方向为计算机体系结构.