

一种适用于嵌入式图形处理器的多端口纹理 Cache 的设计

许 强, 陈 杰, 刘 建, 王 云, 胡哲琨

(中国科学院 微电子研究所, 北京 100029)

摘 要: 为了提高嵌入式图形处理器的纹理单元效率, 提出了一种多端口纹理高速缓存(Texture Cache)结构. 该结构采用了基于块的光栅化和块交错的纹理内存组织, 能够充分发掘数据间相关性, 提高了 Cache 命中率; 此外该结构采用 Cache 预取技术, 有效隐藏了访存延迟; 为了进一步提高数据吞吐率, 设计了 4 个读端口, 可支持并行读取 4 个纹素. 仿真结果表明, 设计的 Cache 可达到 92% 左右命中率, 访存性能可达到零延迟内存系统的 90%, 数据吞吐率是单端口 Cache 的 3 ~ 4 倍.

关键词: 图形处理器; 光栅化; 纹理高速缓存; 分层细化; 预取; 多端口

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2013)11-0027-04

The Design of Multiported Texture Cache for Embedded Graphics Processing Unit

XU Qiang, CHEN Jie, LIU Jian, WANG Yun, HU Zhe-kun

(Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China)

Abstract: In order to improve GPU's texture unit efficiency, a multiported texture cache architecture is proposed. The architecture employs a tiling rasterization order and a block interleaving memory organization, which can fully exploit the data locality and improve cache hit rate. In addition, the architecture employs cache prefetching technology, which can hide memory latency. In order to further improve data throughput, four read ports supporting 4 parallel reading are designed. Simulation results show that the hit rate of the proposed cache is about 92% and can attain 90% of the performance of a zero latency memory system. The data throughput is about 3~4 times of the single ported cache.

Key words: Graphics Processing Unit; rasterization; texture cache; mipmap; prefetching; multiported

1 引言

图形处理器(Graphics Processing Unit, GPU)被广泛应用于图形处理任务的硬件加速. 随着平板电脑和手机的普及, 人们对移动领域 3D 图形的需求越来越大, 对图形质量的要求也越来越高. 由于实时性的 3D 显示系统需要很大的计算量与带宽, 所以移动平台的 GPU 在过去 10 年发展并不算快.

纹理映射单元在 GPU 流水线中起着关键性的作用, 纹理映射是一个非线性的过程, 纹理映射到屏幕空间会发生变形, 出现放大和缩小现象, 为了解决

这种现象, 需要进行纹理滤波. 纹理滤波分为各向同性滤波和各向异性滤波, 嵌入式领域一般选择实现 OpenGL ES^[1] 嵌入式 3D 图形标准建议的 Mipmap^[2] 三线性滤波. 然而三线性滤波产生一个像素需要访问 8 个纹素(Texel), 带宽需求较大, 综合考虑渲染质量和带宽因素, 在质量损失不大的情况下可以使用比较通用的双线性滤波^[3].

纹理访存是决定 GPU 像素产生率的关键, 也是纹理映射单元和整个 GPU 性能的瓶颈, 纹理 Cache 可以缓解带宽问题. 纹理 Cache 不同于 CPU 的 Cache: 首先, 纹理 Cache 一般为只读 Cache; 其

次,纹理 Cache 同整个 GPU 结构结合十分紧密, Hakura 等人指出,纹理 Cache 的命中率同纹理的内存组织、光栅化方式和 Cache 结构紧密相关^[4];另外,为了达到实时渲染的目的,纹理 Cache 最好能实现多端口,由于双线性纹理滤波单元需求 4 个纹素,如果不实现 4 路并行读取,则流水线必须停顿下来等待 4 个纹素分别读回.本文分析了双线性滤波所需纹素的位置关系,并实现了 4 纹素并行读取,减少了 GPU 流水线的停顿.

2 Tile-based 图形处理架构

Tile-based 架构图形处理器是 Imagination 公司 PowerVR 系列 GPU 所采用的结构,被广泛应用于 iPhone、iPad 等消费电子中.该架构的 GPU 将屏幕切割为相对较小的 Tile,再分别对每个 Tile 进行渲染.相比传统的直接渲染,Tile-based 架构能大量减少甚至消除深度缓存的读写,配合延迟渲染技术能减少无效的纹理操作.同时,基于 Tile 的光栅化方式可以提高纹理 Cache 的效率.图 1 所示为本文 Tile-based 结构 GPU 软件模型的整体结构.

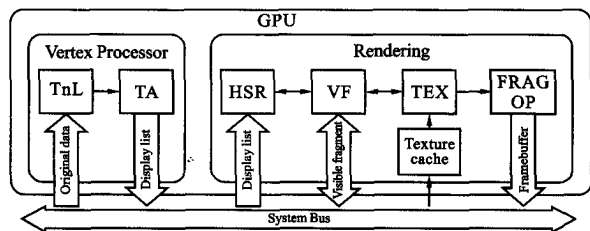


图 1 Tile-based GPU 模型

整个 GPU 可分为顶点处理和三角形渲染两个部分.输入场景由顶点、法向量和纹理坐标组成,经过顶点变换和光照(TnL)、三角形建立和显示列表的生成(TA)、光栅化和隐藏面消除(HSR)等操作之后,送入可显示缓存(VF).纹理模块从 VF 中取出可被显示的片段(Fragment),通过插值计算其纹理坐标,通过纹理 Cache 和纹理内存交互取出纹素之后再进行逐像素处理,最后存入帧缓存.文献^[4]中指出 8×8 的光栅化可以使 Cache 性能最优,但是从整个 GPU 效率考虑,本文采用 Antochi^[5]建议的 32×32 大小的 Tile.

3 纹理滤波

纹理映射最基本的形式是将二维图像投影到屏幕的三角形上面,在 GPU 中,该过程是逆向进行的,即对每个屏幕上的像素点,计算其在二维纹理图像中对应的坐标.屏幕中的像素点对应到纹理图像

的时候,并不一定刚好对应一个纹理图像中的点,而是对应于纹理图像中的一个区域,这时候就需要对该区域的纹理图像进行加权,即滤波.图 2(a)中表示了纹理映射过程.

图 2(b)中表示的是 Mipmap 图像金字塔, Mipmap 是 Lance Williams 提出的一种纹理滤波技术,也是目前几乎所有的图形硬件均支持的滤波方式.其本质思想是通过下采样构建一个纹理图像金字塔,金字塔中纹理图像大小必须满足 $2^m \times 2^n$,通过计算屏幕像素点在纹理图像中轨迹足迹半径来决定选择纹理图像细化层次(Level-of-Detail, LOD).光栅化阶段得到纹理坐标 (s, t) 和细化层次 d 后,分别在邻近 d 的两层图像中取 4 个邻近 (s, t) 坐标的采样点进行双线性滤波,对上下两层纹理图像的双线性滤波结果进行加权,这就是 Mipmap 三线性滤波.而双线性滤波是三线性滤波的简化,即只取层次与 d 最接近的纹理图像进行滤波.

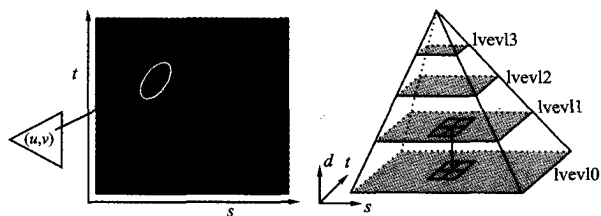


图 2 纹理映射和 Mipmap 图像金字塔

双线性滤波虽然不是最理想的滤波方式,但是实现简单,成本低,存储纹理金字塔所需的额外空间不超过原始纹理图像的 $1/3$.

4 多端口纹理 Cache 设计

4.1 纹理 Cache 整体架构

图 3 是本文纹理 Cache 系统整体结构图.该结构主要包含纹理地址产生模块(tc_in)、地址分配模块(router)、纹理预取和纹理滤波(filter)等模块.

4.2 模块描述

4.2.1 纹理地址产生模块

纹理地址产生模块主要负责计算双线性滤波所需的四个纹素在内存中的地址.并将其发送给地址分配模块.

为了降低纹理对方向的依赖性,提高命中率和减少冲突,纹理图像一般都按块(Tile or Block)的方式存储.图 4 是本文采用的纹理存储示意图,方格表示纹素在纹理图像中的位置,数字代表纹素在内存中的存储位置.本文结构最大支持 1024×1024 分辨率的纹理图像, (s, t) 坐标最大 10 比特,每个纹素

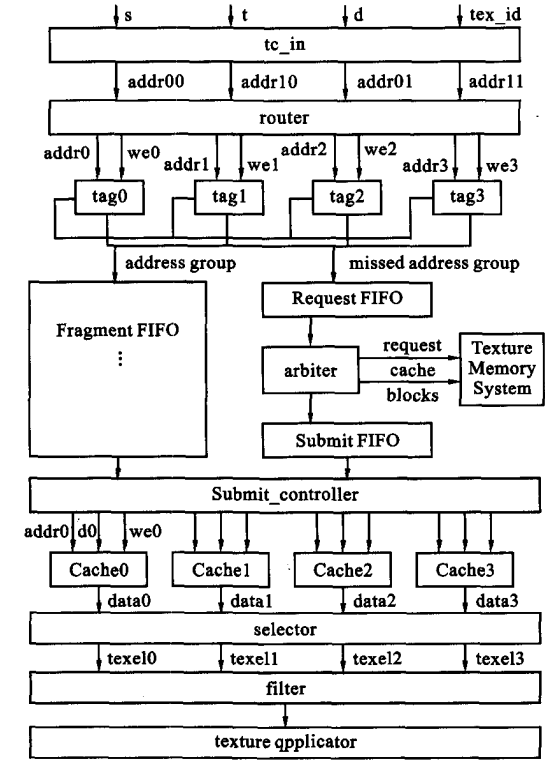


图 3 纹理 Cache 结构

32 比特,按 RGBA8888 格式存储。若纹理图像长宽相等,(s,t)坐标对应的纹素在内存中的地址按式(1)的形式产生:

$$\text{Addr0} = (s_9\ t_9\ s_8\ t_8\ s_7\ t_7\ s_6\ t_6\ s_5\ t_5\ s_4\ t_4\ s_3\ t_3\ s_2\ t_2\ s_1\ t_1\ s_0\ t_0)$$
 (1)

若纹理图像长宽不同,比如大小为 16×64,即 s 和 t 分别用 4 比特和 6 比特表示,那么相应的内存地址为:

$$\text{Addr0} = (t_5\ t_4\ s_3\ t_3\ s_2\ t_2\ s_1\ t_1\ s_0\ t_0)$$
 (2)

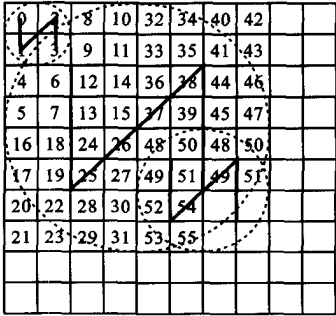


图 4 z 字形纹理映射

从图中可以看出,纹素之间是交织的,2ⁿ×2ⁿ(0≤n≤10)块之间也是交织的. 为了支持多纹理和 Mipmap 纹理金字塔的存储,纹理内存的地址位宽为 25 比特,纹理 ID 和 Mipmap 层次决定了每个纹理图像的起始地址(Texture Base Address),假设为 Addr1,用 Address 作为纹素(s,t)的内存地址,

则:

$$\text{Address} = \text{Addr0} + \text{Addr1}$$
 (3)

4.2.2 纹理地址分配模块

该模块负责对上一模块产生的 4 个地址进行分配. 为了 4 路实现并行读取,将 128 行的 Cache 分成了 4 个 32 行的单端口 Cache. 图 5 显示了 4 个纹素的位置关系. 其中,数字表示的是纹理地址的第 6 位和第 5 位的组合. 0、1、2、3 号 Cache 分别存储图中数字为 0、1、2、3 的 4×4 纹素块. 则滤波所需的 4 个纹素有 3 种情况:

- (1)分布在一个单端口 Cache 的一行.
- (2)分布在两个单端口 Cache 中,每两个纹素分布在一个单端口 Cache 的同一行.
- (3)分布在四个单端口 Cache 中.

可以看出,任意 4 个纹素不可能分布到同一个 Cache 的不同行,因此只需分别取出 4 个单端口 Cache 中的某一行,再利用纹理地址低 6 位选取 4 个纹素便可以实现并行读取.

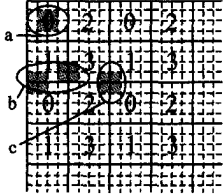


图 5 滤波所需 4 纹素位置

4.2.3 纹理预取结构

图 6 所示为本文所采用的 Cache 预取结构的示意图.

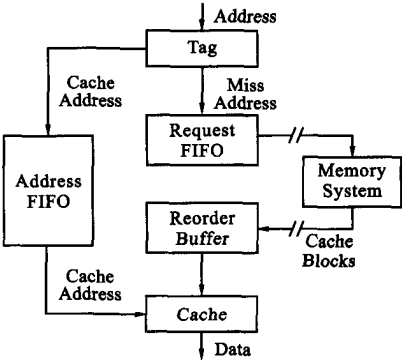


图 6 Cache 预取结构

该结构具有以下特点:第一,Tag 比较和 Cache 访问是分开的;第二,每次 Tag 比较,新的地址都将改写 Tag 的内容,因此,后到的地址请求看到的永远是 Cache 最新的内容. Cache 比较后将地址请求写入 Address FIFO,如果未命中,则将地址请求写入 Request FIFO;第三,针对乱序返回结果的内存

系统,需要一个重排序缓存(Reorder Buffer)对从内存取回来的 Cache 行重新排序,如果所使用的内存系统返回数据和内存请求发出的先后顺序一致,那么 Reorder Buffer 可以用 FIFO 代替,本文内存建模较简单,所以可使用 Submit FIFO 代替 Reorder Buffer;第四,Fragment FIFO 提交的必要条件是,其请求的 Cache 行已全部写回.为保证正确性,若 Address FIFO 头部地址命中,则可以直接提交.若未命中,必须等待相应的 Reorder Buffer 提交后,Address FIFO 才可以提交,取出所需数据,同理 Reorder Buffer 也必须等待 Address FIFO 相应地址请求到达头部时才能提交.该提交控制逻辑由图 3 中 submit_controller 模块完成.

流水线停顿的条件是:Request FIFO 满并且有 Request FIFO 的写请求,或者 Fragment FIFO 满.因此在该 Cache 系统中,3 个 FIFO 的深度是影响性能的关键. Matthew Eldridge 等人在文献[6]中指出,Fragment FIFO 深度与不同内存系统的延迟特性有关,Request FIFO 深度与命中率相关,命中率越高 Request FIFO 深度可以越小,而 Submit FIFO 深度一般和 Request FIFO 一样.本文对内存系统建立了简单的模型,根据文献[6]的经验模型,Fragment FIFO 深度为 128,Request FIFO 和 Submit FIFO 的深度为 16.

4.2.4 纹理滤波模块

纹理滤波模块负责对 Cache 中取出的四个纹素进行双线性滤波,权值是纹理坐标转换时产生的小数值,输出数据为最终所求的像素值,再由逐像素处理模块进行处理后存入帧缓存.

4.3 纹理 Cache 参数

纹理 Cache 的设计需要考虑以下几个参数:Cache 大小,Cache 块(行)大小,相联方式.提高命中率最直接的方法就是增加 Cache 容量,但这样做将会使 Cache 命中时间延长,开销增大^[7].文献[3, 8]中指出,8 kB 大小的纹理 Cache 已经能达到很高的命中率,因此本文采用 8 kB 大小的 Cache;文献[4]中指出,Cache 行大小为 64 字节时,性能可以达到最优,本文每个纹素 32 比特,则一个 Cache 行存储 16 个纹素,即一个 4×4 纹素块.可以计算出,本文 Cache 一共有 128 行;经过大量实验发现,按块交织的方式存储纹理时,直接相联的冲突几率几乎和组相联方式一样,因此采取直接相联的 Cache 映射方式.

5 实验结果

测试平台采用 SystemC 搭建, SystemC 是建立在 C++ 基础上的系统级设计语言,它将并发、时钟、端口等概念引入 C++,使其可以对硬件进行建模描述.由于 SystemC 本质上是在 C++ 的基础上添加的一个扩展类,所以仿真速度非常高,更加适用于系统级的建模^[9].

测试平台包括三个部分:时钟精确的纹理 Cache 模型;支持 OpenGL ES 图形标准的 Tile-based 架构 GPU 流水线的软件模型;测试场景.测试场景由顶点、法向量和纹理坐标组成, GPU 软件模块对测试场景进行顶点变换和光栅化,得到纹理坐标后,再由本文纹理 Cache 模块渲染.图 7 是测试场景渲染效果图.

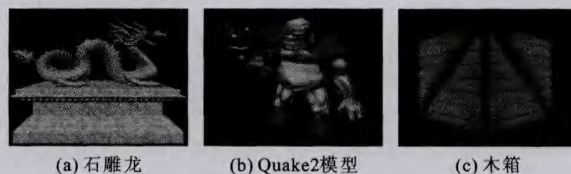


图 7 场景渲染效果图

表 1 是对不同测试场景所得到的结果.从实验结果可以看出,本文纹理 Cache 结构可以达到 92% 左右的平均命中率;像素产生率可达零延迟内存系统的 90% 左右;相对于单端口理想的 0.25 pixel/clock 的像素生成率,本文结构可提升 3 ~ 4 倍.相对文献[3]中的 0.1 pixel/clock 的平均像素产生率,提高了 8 倍左右,而代价是 Cache 预取结构中 FIFO 所消耗的资源.

表 1 龙、木箱和 Quake 模型渲染结果

测试场景	三角形数	平均像素产生率/ (pixel/clock)	命中率/%
石雕龙	214 176	0.89	91.32
木箱	12	0.95	94.73
Quake2 模型	109	0.87	90.30

6 结束语

本文提出了一种多端口基于块存储的图形处理器的纹理 Cache 结构,相对于通过复制数据来实现多端口的结构,提高了 Cache 利用率,消除了 Cache 同步问题,逻辑更简单.通过纹理 Cache 预取策略,降低了缺失代价,减少了流水线停顿.另外,本文提

(下转第 34 页)

区域半径最初呈现出上升状态,达到最大值之后随之减小,直至趋于平稳状态.而图 5 中有效转发能效比也与鱼群搜索区域半径呈现出类似情况.分析其原因,当搜索区域半径从较小开始增大时,包括的转发节点数目增多,从而使得有效转发距离和有效转发能效比增加;当达到一定程度时,搜索区域半径进一步地增加,使转发节点的内聚相关程度降低,反而造成有效转发距减少,从而影响有效转发能效比.

5 结束语

针对网络的有效转发能效比,本文基于概率神经网络和鱼群算法提出了一种新的计算方法 EE-PF.该方法首先利用概率神经网络对转发节点进行聚类,其次根据鱼群的四种行为来刻画网络的有效转发距离和转发能耗,从而获得其有效转发能效比.通过仿真实验对比分析了 EEPF 算法与 EQGOR 算法的性能,结果发现 EEPF 算法具有较好的适应性.在后续研究中,可考虑结合粒子算法和退火算法进行刻画网络的有效转发能效比,以此形成比较完善的评价体系.

参考文献:

- [1] Dai Zhicheng, Li Zhi, Wang Bingwen, et al. An energy aware cluster based routing protocol for wireless

sensor and actor network[J]. Information Technology, 2009, 8(7): 1044-1048.

- [2] 林蔚, 祝启龙. 无线传感器网络节能型数据融合算法[J]. 哈尔滨工程大学学报, 2011, 32(10): 1386-1390.
- [3] 任敬安, 涂亚庆, 张敏, 等. 基于蚁群优化的 Ad Hoc 网络生存时间和其他网络性能平衡路由协议[J]. 计算机工程与科学, 2011, 33(10): 15-24.
- [4] 梁华为, 陈万明, 李帅, 等. 基于蚁群优化的无线传感器网络能量均衡路由算法[J]. 模式识别与人工智能, 2007, 20(2): 275-280.
- [5] 孟凡治, 王换招, 何晖. 基于联合感知模型的无线传感器网络连通性覆盖协议[J]. 电子学报, 2011, 39(4): 772-779.
- [6] 岳林, 易本顺, 肖进胜. 能量平衡与 QoS 保障的无线传感器网络机会路由[J]. 湖南大学学报: 自然科学版, 2011, 38(11): 82-87.
- [7] Xu Bo, Cheng Xianyi. Semantic region classification based on improved probabilistic neural network[J]. Computer Engineering and Applications, 2010, 46(24): 182-184.
- [8] 王兴伟, 秦培玉, 黄敏. 基于人工鱼群的 ABC 支持型 QoS 单播路由机制[J]. 计算机学报, 2010, 33(4): 718-725.

作者简介:

盛光磊 男, (1981-), 硕士, 讲师. 研究方向为网络安全.
李文泽 男, (1979-), 硕士, 讲师. 研究方向为网络优化.

(上接第 30 页)

出的结构可以很方便地与纹理压缩、多级 Cache 等技术相结合,进一步节省带宽需求.测试结果表明,本文提出的纹理 Cache 结构具有良好的性能.

参考文献:

- [1] Khronos Group. Open GL ES common profile specification [S]. USA: Khronos Group, 2008.
- [2] Williams L. Pyramidal parametrics [J]. Computer Graphics, 1983, 17(3): 1-11.
- [3] 程龙, 郭立, 史鸿声, 等. 一种纹理映射算法的 FPGA 实现 [J]. 小型微型计算机系统, 2009, 30(9): 1855-1859.
- [4] Hakura Z, Gupta A. The design and analysis of a cache architecture for texture mapping[C]//Proceedings of the 24th International Symposium on Computer Architecture, Denver, Colorado, 1997: 108-119.
- [5] Ben Juurlink, Iosif Antochi, Dan Crisu, et al. GRAAL: a framework for low-power 3D graphics accelerators [J]. IEEE computer graphics and applications, 2008, 28(4): 63-73.

- [6] Homan Igehy, Matthew Eldridge, Kekoa Proudfoot. Prefetching in a texture cache architecture[C]//SIGGRAPH/Eurographics Workshop on Graphics Hardware, Orlando, 1998: 133-142.
- [7] John L. Hennessy, David A Patterson. 计算机系统结构—量化研究方法[M]. 4 版. 白跃彬, 译. 北京: 电子工业出版社, 2007.
- [8] Michael Cox, Narendra Bhandari, Michael Shantz. Multi-level texture caching for 3D graphics Hardware[C]//Proceedings of the 25th International Symposium on Computer Architecture, Barcelona, 1998: 86-97.
- [9] 奚杰, 陈杰, 朱钥. 利用 SystemC 实现多核系统的快速建模 [J]. 微电子学与计算机, 2010, 27(7): 214-217.

作者简介:

许强 男, (1988-), 硕士研究生. 研究方向为集成电路设计.