

引用格式:郝武,杜慧敏,张丽果,等.嵌入式 GPU 中 U 型存储布局 tile 缓存的设计与实现 [J]. 微电子学与计算机,2019,36(3):91-95. [HAO Wu,DU Hui-min,ZHANG Li-guo, et al. Design and implementation of embedded GPU tile buffer with U-Order layout [J]. Microelectronics & Computer,2019,36(3):91-95.]

# 嵌入式 GPU 中 U 型存储布局 tile 缓存的设计与实现

郝 武,杜慧敏,张丽果,黄世远

(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 tile 缓存在线性布局和 Z 型布局写回时由于地址跨度大而导致 cache 频繁冲突缺失的问题,设计了一种支持多级 U 型存储布局的 tile 缓存,使像素数据写回的地址连续,减少 cache 的冲突缺失,提高 cache 命中率. 实验结果表明,当配置不同尺寸的 tile 缓存时,U 型布局相对于线性布局 cache 命中率提高 4%~13%,相对于 Z 型布局 cache 命中率提高 1%~9%.

**关键词:** 嵌入式 GPU;tile 缓存;U 型布局

**中图分类号:** TP399

**文献标识码:** A

**文章编号:** 1000-7180(2019)03-0091-05

## Design and implementation of embedded GPU tile buffer with U-Order layout

HAO Wu,DU Hui-min,ZHANG Li-guo,HUANG Shi-yuan

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

**Abstract:** Aiming at solving the problem that cache frequent conflict misses in linear layout and Z-layout write back of embedded GPU tile buffer caused by the large address spans, a tile buffer supporting multi-level U-layout is designd,which makes the address continuous when the pixel data is written back, reduces the cache conflict misses and improves the cache hit rate. The experimental results show that when configuring tile buffers of different sizes, the U-layout improves the hit rate by 4%~13% compared to the linear layout cache, and increases the hit rate by 1%~9% compared to the Z-layout.

**Key words:** embedded GPU;tile buffer;U-order layout

## 1 引言

随着半导体技术和计算机技术的发展,嵌入式 GPU 已经成为高性能移动设备中不可缺少的一部分<sup>[1]</sup>,以满足用户对 3D 游戏和高分辨率的需求. 嵌入式 GPU 大部分采用 tile-based 架构<sup>[2]</sup>,将屏幕分为多个 tile,GPU 每次处理一个 tile 的数据<sup>[3]</sup>,以降低功耗,提高性能.

为了提高数据处理速度,嵌入式 GPU 中采用多核技术<sup>[4]</sup>与 SIMD 扩展技术<sup>[5]</sup>提高计算能力,但

是嵌入式 GPU 中 tile 缓存的存储布局(Memory Layout)<sup>[6]</sup>采用行或列的线性布局或者 Z 型存储布局,不能有效的利用 tile 缓存中数据的二维空间局部性<sup>[7]</sup>,在写回数据时,数据地址之间会有较大的跨度,因此经常发生 cache 冲突缺失,降低 cache 命中率.

## 2 嵌入式 GPU 结构

嵌入式 GPU 由像素处理器、几何处理、cache 和内存管理单元组成. 其系统结构如图 1 所示,几何

收稿日期:2018-07-04; 修回日期:2018-08-10

基金项目: 陕西省重点研发计划(2017ZDXM-GY-005);西安邮电大学创新基金(102-602080009)

处理器用于处理顶点数据,像素处理器主要进行多边形建立,光栅化,以及片段着色操作,产生在显示器上显示的图像. 存储管理单元进行地址翻译和内存保护.

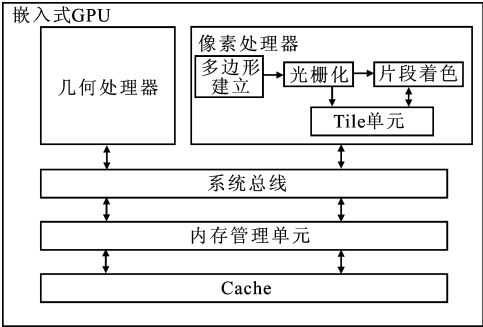


图 1 嵌入式 GPU 结构框图

Tile 单元位于像素处理器处理流程的最后阶段,用于存储一个 tile 的像素数据并将数据通过系统总线和 cache 传输到主存中的帧缓存中. Tile 单元中采用多级 U 型存储布局可以利用图形的二维局部性,提高 cache 的命中率.

3 tile 缓存布局

3.1 线性布局

线性布局也称为行或列布局. 如图 2 所示,图中尺寸  $8\times 8$  的 tile 缓存为行顺序布局,每个像素的数据以行为顺序进行排列,网格中数字即为写回顺序. 因为这种线性扫描顺序无法利用 tile 内数据的空间局部性,导致数据的写回效率较差. 以行顺序布局为例, GPU 绘制的图形多为按块分布,则在进行数据写回时列方向的访问可能由于其较大的地址跨度而导致 cache 频繁的产生冲突缺失.

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|---|----|----|----|----|----|----|----|----|
| 0 | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| 1 | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| 2 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 3 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 4 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 5 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 6 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 7 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

图 2 线性布局示意图

3.2 Z 型布局

如图 3 所示为  $8\times 8$  尺寸的 Z 型布局 tile 缓存. 每  $2\times 2$  的像素块呈 Z 字型,这种布局方式中,图像每两行内所有像素的地址连续,但是 tile 内还是会出现地址的不连续情况<sup>[8]</sup>,图 3 中 15 与 16、31 与

32、47 与 48 之间的地址是不连续的,故在写回时仍会产生冲突缺失. Tile 尺寸越小产生冲突缺失的次数越多.

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|---|----|----|----|----|----|----|----|----|
| 0 | 0  | 1  | 4  | 5  | 8  | 9  | 12 | 13 |
| 1 | 2  | 3  | 6  | 7  | 10 | 11 | 14 | 15 |
| 2 | 16 | 17 | 20 | 21 | 24 | 25 | 28 | 29 |
| 3 | 18 | 19 | 22 | 23 | 26 | 27 | 30 | 31 |
| 4 | 32 | 33 | 36 | 37 | 40 | 41 | 44 | 45 |
| 5 | 34 | 35 | 38 | 39 | 42 | 43 | 46 | 47 |
| 6 | 48 | 49 | 52 | 53 | 56 | 57 | 60 | 61 |
| 7 | 50 | 51 | 54 | 55 | 58 | 59 | 62 | 63 |

图 3 Z 型布局示意图

3.3 多级 U 型布局

为了考虑 tile 缓存内数据的局部性,并减少冲突缺失,本文提出多级 U 型 tile 布局,使帧缓存中所有像素地址变成连续的. 如图 4 所示,以  $8\times 8$  像素尺寸的 tile 为例,整个 tile 第一级为  $8\times 8$  的图块,在第二级将 tile 划分为 4 个尺寸为  $4\times 4$  的图块,它们以 U 型排列,在第三级将第二级的图块划分为 4 个  $2\times 2$  的子图块,第四级为一个像素. 在 tile 尺寸  $16\times 16$  时和  $4\times 4$  时也是按照此方法进行划分,直到最后一级为一个像素.

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|---|----|----|----|----|----|----|----|----|
| 0 | 0  | 1  | 4  | 5  | 16 | 17 | 20 | 21 |
| 1 | 3  | 2  | 7  | 6  | 19 | 18 | 23 | 22 |
| 2 | 12 | 13 | 8  | 9  | 28 | 29 | 24 | 25 |
| 3 | 15 | 14 | 11 | 10 | 31 | 30 | 27 | 26 |
| 4 | 48 | 49 | 52 | 53 | 32 | 33 | 36 | 37 |
| 5 | 51 | 50 | 55 | 54 | 35 | 34 | 39 | 38 |
| 6 | 60 | 61 | 56 | 57 | 44 | 45 | 40 | 41 |
| 7 | 63 | 62 | 59 | 58 | 47 | 46 | 43 | 42 |

图 4 tile 尺寸  $8\times 8$  的 U 型布局示意图

采用多级 U 性布局,一帧图像内所有像素写回的地址都是连续的. 减少了线性布局和 Z 型布局中列方向上的地址跳跃,在数据写回时地址连续,一定程度上减少 cache 的冲突缺失. 用多级 U 型顺序布局不需要程序员进行额外操作,只需要编译器支持这种存储布局,但是这种布局的地址计算需要相应的硬件支持.

4 tile 地址计算

采用 tile-based 架构时, GPU 每次处理一个 tile 的数据,所以每次写回一个 tile 的数据. Tile 的地址分布如图 5 所示, GPU 会依据指令给出帧缓存的起

始地址,首先确定当前处理的 tile 在帧缓存中的序号,计算当前 tile 的起始地址,然后进行 tile 内像素地址迭代即可按顺序写回该 tile 的数据。



图 5 Tile 地址分布示意图

4.1 线性顺序布局 tile 地址计算

4.1.1 线性顺序布局 tile 起始地址计算

Tile 内采用线性顺序布局时,帧缓存顺序也采用线性顺序布局,如图 6 所示,每个方格代表一个 tile,所有 tile 按照线性排列。

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|---|----|----|----|----|----|----|----|----|
| 0 | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| 1 | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| 2 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 3 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 4 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 5 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 6 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 7 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

图 6 帧缓存中 tile 布局示意图

GPU 依据指令提供当前 tile 的位置信息,如图 6 所示,阴影方格为当前 tile 在整个帧缓存中的位置,其坐标为(3,2)。

首先计算对应的行初始地址,以图 4 为例就是求出第 0 行与第 1 行总的长度:

Addr 行=tileY×tilesca

linelength

×Rowspertile

(1)

式中,tileY 为当前 tile 的 Y 坐标;tilesca  
linelength 为帧缓存一行的数据长度;Rowspertile 为每个 tile 的行数。其次计算列初始地址,以图 4 为例,求出 16,17,18 三个 tile 的总数据长度。

Addr 列=tileX×Pixel

size

×Colspertile

(2)

式中,tileX 为当前 tile 的 X 坐标;Pixel  
size 为一个像素的数据长度;Colspertile 为每个 tile 的列数。最后将行地址、列地址与帧缓存起始地址相加得到当前 tile 的内存地址 Resultaddr。

Resultaddr=startaddr+addr 行+addr 列

(3)

4.1.2 线性顺序布局 tile 内像素地址计算

计算出 tile 初始地址后,依据 tile 初始地址可对整个 tile 的像素地址进行计算。像素地址更新算法如图 7 所示。

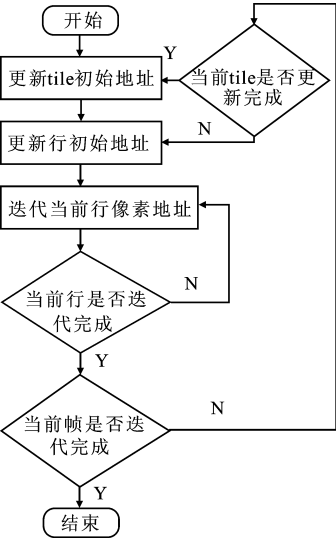


图 7 线性布局像素地址计算算法流程图

(1)依据行顺序计算每个像素的地址,依据像素尺寸对地址进行累加。

(2)当计算完 tile 当前行最后一个像素时,需要更新行初始地址,切换到帧缓存中的下一行。

(3)重复(1)、(2),直至计算完当前 tile 的最后一个的像素地址。

(4)切换到下一个 tile 的初始地址。

通过这样就可以对整个帧缓存的地址进行迭代计算。针对不同 tile 尺寸,所设置的迭代次数不同。

4.2 U 型顺序布局 tile 地址计算

4.2.1 U 型顺序布局 tile 起始地址计算

采用 U 型存储布局时,Tile 在帧缓存中的布局与像素在 tile 中的布局相同。如图 8 所示,都以 U 型顺序来进行布局。

以图 8 为例,说明 U 型存储布局中 tile 地址的计算:

(1)得到 tile 的序号 Umap ,U 型布局中所有 tile 的序号都是固定的,故以当前 tile 的坐标 tileX, tileY 为索引,得到 tile 对应的序号,如图 8 中当前 tile 的序号为 9。

(2)计算整个 tile 的地址跨度,为像素尺寸 pixelsize 与 tile 尺寸 tilesize 的乘积。Umap 与 tile 地址跨度的乘积与帧缓存初始地址求和即可求出当前

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|---|----|----|----|----|----|----|----|----|
| 0 | 0  | 1  | 4  | 5  | 16 | 17 | 20 | 21 |
| 1 | 3  | 2  | 7  | 6  | 19 | 18 | 23 | 22 |
| 2 | 12 | 13 | 8  | 9  | 28 | 29 | 24 | 25 |
| 3 | 15 | 14 | 11 | 10 | 31 | 30 | 27 | 26 |
| 4 | 48 | 49 | 52 | 53 | 32 | 33 | 36 | 37 |
| 5 | 51 | 50 | 55 | 54 | 35 | 34 | 38 | 39 |
| 6 | 60 | 61 | 56 | 57 | 44 | 45 | 40 | 41 |
| 7 | 63 | 62 | 58 | 59 | 47 | 46 | 43 | 42 |

图 8 帧缓存 U 型布局示意图

tile 的初始地址, tile 初始地址计算如下:

$$\begin{aligned} \text{resultaddr} = & \text{startaddr} + \text{Umap} \times \text{pixelsize} \\ & \times \text{Tilesize} \end{aligned} \quad (4)$$

4.2.2 U 型顺序布局 tile 内像素地址计算

U 型存储一个 tile 内像素地址连续排列, 所以  
一个 tile 内的地址可连续迭代计算, 计算流程图如  
图 9 所示.

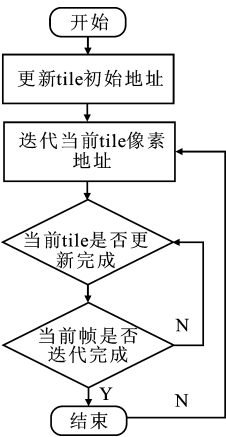


图 9 U 型像素地址迭代流程图

当前 tile 像素地址迭代完成时, 更新 tile 初始  
地址, 迭代下一个 tile 的像素地址, 直至整个帧缓存  
像素地址迭代完成.

5 地址计算电路设计

地址计算电路分为 tile 起始地址计算模块和像  
素地址迭代模块, 其中 tile 起始地址计算模块产生  
当前 tile 的初始地址, 像素地址迭代模块产生每个  
像素对应内存中的地址.

5.1 tile 起始地址电路设计

本文采用三级流水计算 tile 起始地址, 为减小  
电路面积, 降低设计复杂度将线性地址计算与 U 型地  
址计算电路组合在一起, 共用计算单元, 并采用对  
数运算与移位器结合的方式解决地址计算中的部分  
乘法问题, 电路框图如图 10 所示.

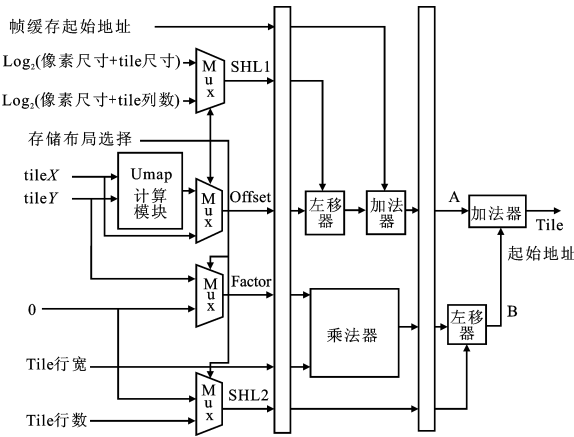


图 10 tile 起始地址计算电路

tileX, tileY 为当前 tile 在帧缓存中的坐标,  
SHL1 控制第二级流水线中左移器左移位数, offset  
为被左移的数. SHL2 控制第三级流水线中左移器  
左移位数, 乘法器结果为被左移的数.

(1)第一级流水线: 根据存储布局选择进行哪种  
地址计算并依据数据布局类型选择数据, 若为线性  
计算, 则选择 SHL1 为  $\log_2(\text{像素尺寸} + \text{tile 列数})$ ,  
offset 为 tileX, Factor 为 0, SHL2 为 0; 若为 U 型布  
局, 则选择 SH1 为  $\log_2(\text{像素尺寸} + \text{tile 尺寸})$ , off-  
set 为 Umap, Factor 为 tile 行宽.

(2)第二级流水线: 若为线性布局, 使用对数及  
左移器计算列地址, 对 offset 进行左移, 左移的位  
数为 SHL1, 将列地址与帧缓存起始地址相加得到 A,  
乘法器输出为 tileY  $\times$  tileX 行宽的结果; 若为 U 型  
布局, 则左移器中进行的操作即为式 3 中后半部分  
乘法操作. 加法器中进行的操作就得到了 tile 的  
起始地址. 乘法器结果为 0.

(3)在第三级流水线中, 若为线性计算, 则左  
移器中计算行地址 B, 然后将行地址与第二级得到  
的 A 相加得到 tile 的起始地址; 若为 U 型布局, 则  
左移器结果 B 为 0, 加法器结果为 tile 的起始  
地址.

5.2 像素地址迭代电路设计

为节省电路资源, 本文同样将两种地址迭代算  
法组合成为一个电路进行设计. 电路结构如图 11 所  
示, 其中 CurrentScanlineAddr 为当前行初始地址,  
TilestartAddr 为当前 tile 初始地址, pixelsize 为像  
素尺寸, Scanlinelength 为当前帧每行地址跨度. 电  
路功能依据布局不同执行不同功能.

在线性布局模式下, 当切换 tile 后, 选择 tile 初  
始地址为当前地址, 使用下一行的初始地址更新当

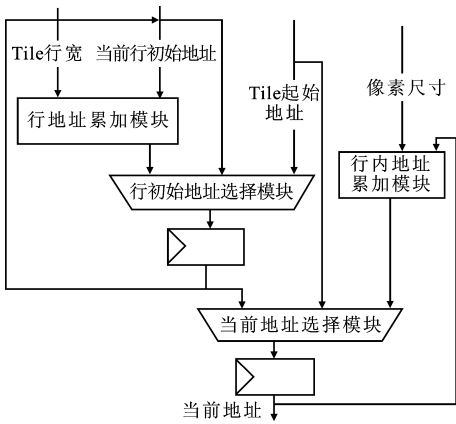


图 11 像素地址迭代电路结构图

前行初始地址,随后进行 tile 行内迭代,当前地址每次加像素尺寸进行更新,行内迭代完成后,选择当前行地址为当前地址。

在 U 型布局模式下,当切换 tile 后,与线性布局模式相同,然后进行像素地址迭代,U 型布局像素地址迭代次数为 tile 尺寸,即一个 tile 有多少个像素就进行多少次迭代。将整个 tile 迭代完成后将当前地址更新为下一个 tile 的初始地址,当前 tile 像素地址迭代完成。

6 算法验证

本文对三种存储布局下 cache 的命中率进行测试。配置不同的 tile 的尺寸大小,对线性存储布局、Z 型存储布局和多级 U 型存储布局进行像素写回测试,并记录 cache 的命中率。

由图 12 可以得出,在不同尺寸的 tile 下,使用多级 U 型存储布局相对于线性布局 cache 命中率提高 4%~13%,相对于 Z 型布局 cache 命中率提高 1%~9%。

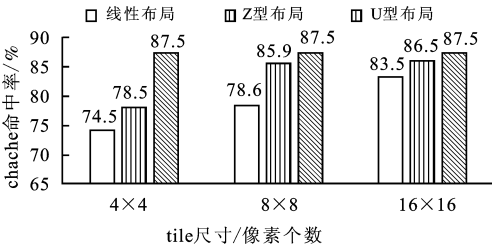


图 12 三种存储布局 cache 命中率测试

7 结束语

本文针对 tile 缓存数据二维局部性提出并实现了多级 U 型存储布局和寻址电路,利用 tile 缓存数据二维局部性,提高 cache 的命中率. 测试结果表明,使用 U 型布局可将 cache 命中率提高. 说明应用多级 U 型存储布局可以提高数据写回效率。

参考文献:

[1] 焦继业. 低功耗高性能移动图形顶点处理器设计关键技术研究[D]. 西安:西安电子科技大学,2013.

[2] 朱玥. Tile-Based 图形处理方法及高质量图形算法设计[D]. 合肥:中国科学技术大学,2012.

[3] 王佳. Tlie-based 嵌入式图形处理器的研究与设计[D]. 济南:山东大学,2014.

[4] 陈刚,关楠,吕鸣松,等. 实时多核嵌入式系统研究综述[J]. 软件学报,2018, 29(7):1-26.

[5] 常轶松. 面积带宽优化的嵌入式 GPU 可编程着色器体系结构研究[D]. 天津:天津大学,2013.

[6] BETHEL E W, CAMP D, DONOFRIO D, et al. Improving performance of structured-memory, data-intensive applications on multi-core platforms via a space-filling curve memory layout[C]// IEEE International Parallel and Distributed Processing Symposium Workshop. IEEE Computer Society. Salvador, Brazil, 2015:565-574.

[7] LI N, FANG Y J. Memory layout method for image processing embedded system[C]// ICEMI'2005 国际电子测量与仪器学术会议. 中国,北京,2005.

[8] WANG B K, FUKAZAWA Y, KONDO T, et al. A cache memory with unit tile and line accessibility[C]// IEEE/ACIS, International Conference on Computer and Information Science. Nagoya, Japan IEEE, 2016: 1-6.

作者简介:

郝武 男,(1992-),硕士. 研究方向为计算机体系结构.  
E-mail:476666755@qq. com.

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.

张丽果 女,(1979-),博士. 研究方向为集成电路设计.

黄世远 男,(1992-),硕士. 研究方向为计算机体系结构.