嵌入式多媒体应用中的片上存储器分配

温淑鸿[1,2], 崔慧娟[1], 唐 昆[1]
(1. 清华大学电子工程系, 北京 100084; 2. 中国传媒大学信息工程学院, 北京 100024)

摘 要: 为了提高嵌入式多媒体应用的运行速度并降低功耗, 本文提出一种高效利用片上存储器的方法. 将数据矩阵划分成合理大小的子块, 分阶段地将数据子块转移到片上, 并尽可能重复利用已经转移到片上的数据, 以便有效地减少片外存储器与片上存储器之间的数据转移. 通过对汇编语言中存储器阵操作数适当分配, 避免读写数据延迟. 根据汇编语言代码写出不产生流水线停滞的各个矩阵操作数的存储器位置限制条件, 根据限制条件, 本文提出求解矩阵分配的方法.

关键词: 存储器; 片上存储器 存储器分配; 数据再利用
中图分类号: TP343 文献标识码: A 文章编号: 0372-2112 (2005) 11-1937-04

Scratchpad Memory Assignment in Embedded Multimedia Application

WEN Shu-hong[1,2], CUI Hui-juan[1], TANG Kun[1]
(1. *Depart. of Electronic Engineering, Tsinghua University, Beijing* 100084, *China*;
2. *Information Engineering School, Communication University of China, Beijing* 100084, *China*)

Abstract: To improve embedded multimedia application performance and reduce power consumption, an efficient method of exploiting scratchpad memory is proposed. Big data array are divided in small tiles and copied to on-chip memory by stages. Data in scratchpad memory should be reused as much as possible to reduce data transfer between off-chip memory and scratchpad memory. Memory access delay can be avoided by proper assignment of memory operand in assemble instructions. Some operand memory constraint conditions can be listed out to minimize pipeline stall. A method of array assignment is proposed in these conditions.

Key words: memory; scratchpad memory; memory assignment; data reuse

1 引言

随着 CPU 速度的迅速提高, CPU 与片外存储器的速度差异越来越大, 匹配 CPU 与外部存储器的方法通常是采用 Cache 或者片上存储器. 微处理器中片上存储器结构通常包含指令 Cache, 数据 Cache 或者片上存储器. 对于嵌入式设备上的数据密集的应用, 数据 Cache 与片上存储器相比存在以下缺陷: (1) 片上存储器是固定的单周期访问, 可在设计时而不是运行时研究数据访问模式; 而 Cache 还要考虑击不中的情况, 因而有可变的数据访问时间, 执行时间的预测更加困难. (2) 使用 Cache 执行时间的不可预测性影响编译器的优化; (3) 细颗粒的 Cache 对于图像编码等的规则数据访问并不合适, 因而使用 Cache 对于嵌入式设备可能不是最优的[1]. 文[2] 指出, 对于大多数应用, 使用片上存储器比使用数据 Cache 能量平均大约节省 40%, 芯片面积与时间的乘积仅为 Cache 的 46%. 因而对于嵌入式多媒体处理器, 片上 RAM 作为数据 Cache 的替代, 功耗更低. 片上存储器的有效使用对于提高嵌入式应用的速度, 降低功耗具有重要的意义.

文[3,4] 讨论了在同时具有数据 Cache 和片上 SRAM 的处理器上标量和矩阵变量的存储器分配方法. 文[5~7] 以摩托罗拉公司的 DSP56000 为平台, 文[8] 以 AMS Gepard DSP 为平台, 分别讨论了如何把数据分配到 X/Y 数据存储器块, 以便最大限度地利用数据移动的并行性. DSP56000 片上 X、Y 数据存储器都是单端口的, 并且容量较小. 与 DSP56000 不同, TI 公司的 TMS320C55x 具有更多的数据总线, 片上 RAM 容量更大, 且分块多, 具有访问能力更强的 DARAM.

TMS320C55x 具有极低的功耗(0.05mW/MIPS), 非常适合手持设备, 现在已经集成至 TI 公司专门针对 3G 手机的高性能多媒体处理器上. C55x 片上除了 24K 字节的指令 Cache 外, 还有 64K 字节的双端口存储器(DARAM), 96K 字节的单端口存储器(SARAM). DARAM 和 SARAM 总共 160K 字节, 分成 20 个块, 每个块 8K 字节. 本文以 C55x 上的视频编码器为例, 讨论片上存储器的有效使用.

2 数据的片外、片上动态分配

因为片上存储器比片外存储器具有更高的数据访问能力

收稿日期: 2004-09-30; 修回日期: 2005-06-06
基金项目: 国家自然科学基金(No. 60272020)

和更小的访问功耗,所以尽可能分配数据到片上存储器,未能分配到片上的数据可在 CPU 处理前转移到片上,已经转移到片上的数据,应尽可能在片上保存直到其生命期结束,以便尽可能减少数据从片外存储器到片上存储器的数据转移. 在视频编码等应用中,标量、常数相对于矩阵而言,通常数量较少,可以分配到片上;若分配到片外,在运算时直接存取片外数据,CPU 流水线将会停滞. 直接存储器存取(DMA)可以在存储器之间,存储器与外设之间转移数据,除了 DMA 通道参数初始化以外,DMA 转移数据和 CPU 处理数据可以并行进行. 设置 DMA 通道参数需要一定的时间,采用 DMA 来转移单个变量或常数的开销可能比直接存取更大,因此 DMA 适合转移具有较多数据的矩阵,并不适合片外标量的转移. 包含大量元素的矩阵可以分配到片外,处理前使用 DMA 转移到片上存储器.

局部变量由编译器分配到软件栈上,C55x 具有两个软件栈:数据栈和系统栈. C55x 的栈有三种工作模式,可设置成双 16 比特快返回模式,以减少栈所占的存储器空间,并提高其运行速度. 数据栈和系统栈在函数调用及返回时同时访问,可将这两个栈分配到 DARAM 块或者不同的 SARAM 块内.

本文中数据存储器的分配,强调从实际多媒体应用处理的基本数据块出发,分析简单直观. 多媒体算法总是将原始输入数据分成一定大小的块进行处理,并产生对应该输入的最后输出. 如果片上没有足够的存储器,大量的输入数据和最后结果仅能可存储在片外. 对于元素较多的矩阵,可以根据算法特征将矩阵分成若干数据子块,如 H.263 编码器中的宏块,搜索窗等,或者单纯根据可得到的片上存储器数量分成适当大小的子块逐个运算,然后分析数据子块的生命期和使用频率. 我们定义数据子块的生命期为首次使用到最后一次使用之间的间隔,而通常变量的生命期为定义到最后使用之间的间隔,例如定义整型数组 int MB[384],用来存储待编码宏块的数据,图像的某个宏块的数据在该宏块编码结束后,该宏块数据的生命期也就结束,然后该数组用来存储下一宏块的数据,因而变量的生命期远比存储在该变量中的某一具体数据生命期要长. 若数据子块具有不相交的生命期,则可以共享相同的片上存储器. 很多数据子块在运算中多次使用,可在首次运算前转移到片上,并尽可能保存到生命期结束,即直到这些数据不再使用为止,因而这些数据仅需要一次转移. 将程序执行时间看成是由很多连续的时间间隔组成的,若在下个时间间隔内需要转移新的数据到片上供 CPU 处理,而片上又没有足够的存储器存储这些数据,这时将随后需要连续频繁使用的数据保留到片上;对于随后较少使用的数据,若片外存储器还保存有该数据的备份,这些数据可直接覆盖,等到下次使用时再从片外存储器拷贝到片上;否则,在覆盖前将数据转移到片外. 在片上分配一定的缓冲区,用来存储需要再次使用的数据,可有效地减少片外存储器的访问. 对于中间结果,尽量在使用前分阶段计算,使用后释放,以缩减存储中间结果的存储器需求. 通过数据的这种动态分配,既可以减少或避免访问片外慢速存储器所引起的指令延迟,又可以减少片外到片上的数据转移.

在 H.263 视频编码器中,编码是按宏块顺序进行的,INTRA 宏块编码仅需要当前的编码宏块数据,INTER 宏块编码还需要以当前宏块为中心的重建图像搜索窗. 因此根据算法特征将整帧输入图像划分成宏块,某个宏块数据在编码前转移到片上,这一宏块编码结束后就不再使用,这部分片上存储器就可释放,用来存储下一宏块数据. 若在编码的同时采用 DMA 转移下一个宏块,这需要在片上分配两个宏块的存储器空间,用来存储编码的原始图像.

在进行 INTER 帧编码时,运动搜索需要使用前一帧的重建图像作为参考,设搜索范围为[-16,+16],编码该宏块需要搜索参考图像中以编码宏块位置为中心的 9 个宏块,即前一帧中宏块(x, y)的的重建图像直到编码$(x+1, y+1)$宏块后生命期才结束. 以 CIF 分辨率为例,不可能把一帧图像的所有重建宏块保存到生命期结束,因而部分重建图像必需暂时存储在片外,若在编码$(x-1, y-1)$前将重建宏块(x, y)拷贝到片上并一直保存到编码$(x+1, y+1)$宏块结束,只需要在片上分配将近 3 个 GOB 的空间用来存储参考图像,就可以保证每个宏块的重建图像数据只需要一次片外到片上的转移.

半像素内插结果,用于在整像素运动搜索后作为半像素搜索的参考,因而可在整像素搜索后、半像素搜索前,围绕整像素运动矢量,对整像素运动矢量对应的匹配宏块进行内插,这样就没有必要在编码 INTER 帧前将整帧图像进行内插,可显著减少存储内插结果的存储器数量,从而分配在片上.

3 片上数据的存储器分配

TMS320C55x 除了读指令的地址数据总线外,还有三条用于从存储器读操作数的地址数据总线,两条写操作数到存储器的地址数据总线. CPU 在一个周期内可完成多个操作数的读写,由于每个 DARAM 块或 SARAM 块有限的访问能力,这些操作数位于适当的 DARAM 或 SARAM 块内,才能在单周期内完成多个数据的读入或者数据的同时读写,而不产生延迟.

3.1 指令代码的分配

应用程序的指令代码可以存储在片外存储器,通过指令 Cache 进行访问,可以减少 CPU 读指令代码与 CPU 读写片上存储器内数据的冲突,同时将空余更多的片上存储器空间用于数据分配. 若存储程序代码和数据所需的存储器总和少于片上存储器容量,将代码分配到片外存储器的性能与代码数据全部分配到片上存储器相比,性能降低大约 10%. 因此当代码和数据总和小于片上存储器容量时,应该全部分配到片上存储器. 通常程序代码仅供 CPU 读取、并不修改,而数据经常需要同时读写,因而应尽量将代码存储在 SARAM 内,以便将访问能力更强的 DARAM 用来存储数据. 在单个 CPU 周期内,SARAM 仅有一次访问能力,同时读取指令和数据必然产生延迟,为了保证读取数据时不产生延迟,数据不能与访问这些数据的代码存储在同一 SARAM 块内. 也就是说,当程序代码大小不是刚好整数个块大小时,可通过调整代码或者数据的存储器分配,以免 CPU 读代码与读写数据产生冲突.

3.2 数据分配

前面已经讨论过变量和常数的分配,这里主要讨论耗时

较多的矩阵运算. 通常可以用 C 语言或者汇编语言编写应用程序, C 语言编译后可产生汇编代码. 在汇编语言的代码中, 找到处理矩阵操作数的指令, 依次列举这些指令不产生延迟的矩阵分配限制, 并求解满足这些限制条件的片上存储器分配. 下面列出了 C55x 中一些常见的存储器操作数访问形式:

(1) Xmem read ‖ Ymem read.

Xmem write ‖ Ymem write.

Xmem read ‖ Ymem write.

为了不产生延迟, 要求 Xmem 和 Ymem 位于 DARAM 块内或者不同的块内.

(2) Lmem1 read ‖ Lmem2 write.

为了不产生延迟, 要求 Lmem1 和 Lmem2 位于 DARAM 块内或者不同的块内.

(3) Xmem read ‖ Cmem read.

例如汇编指令: MACMR Xmem, Cmem, ACx, 为了不产生延迟, Xmem, Cmem 不在同一块内, 这包括不在同一 SARAM 块内, 也不在同一 DARAM 内.

(4) Xmem read ‖ Ymem read ‖ Cmem

例如汇编指令: MPY Xmem, Cmem, AC0: : MPY Ymem, Cmem, AC1 以及 FIRSADD Xmem, Ymem, Cmem, ACx, ACy 都要求 Xmem 和 Ymem 位于 DARAM 块内或者不同的 SARAM 块内, 并且 Xmem, Cmem 不在同一块内.

上述指令不产生延迟的约束条件可分成两类基本约束条件: (1) 两变量位于 DARAM 块内或者两变量位于不同的块内, 记为条件 A, 这是由 SARAM 块或者 DARAM 块访问能力产生的限制; (2) 两变量位于不同的块内, 记为条件 B, 这是由于 CPU 总线的特殊结构产生的限制. 其中条件 A 中的两变量可在同一 DARAM 块内, 或者不同的 SARAM 块内, 或者一个变量在 DARAM 内, 另一个在 SARAM 内. 条件 B 指的是两变量在不同的 DARAM 块内, 或者在不同的 SARAM 块内, 或者一个变量在 DARAM 块内, 另一个在 SARAM 块内. 条件 A 可看成是两种条件的逻辑或关系.

$A = B$ or C.

其中条件 C 定义为两变量都位于 DARAM 块内. 循环中的操作数一般表现为矩阵的一个元素, 在一个应用程序中, 通常有多个矩阵, 矩阵中的元素应同时满足多个上述基本条件. 当矩阵较多, 限制条件复杂时, 可以使用计算机求解数据存储器分配, 以满足矩阵访问不产生延迟的条件. 在这里, 我们只需要求出满足条件的一个解, 并不需要求出所有可能的解, 因而对求解问题做一定的简化.

设 x, y 分别是矩阵 X, Y 的某一个元素, X, Y 位于不同的块内是 x, y 位于不同的块内的充分条件, 同样 X, Y 都位于 DARAM 内或者不同的块内是 x, y 都位于 DARAM 内或者不同的块内的充分条件, 例如 X 位于 DARAM 块, Y 矩阵部分位于与 X 相同的 DARAM 内, 其余位于 SARAM 内, 也能使 x, y 满足条件 A.

例如: N 个矩阵需要同时满足 N_1 个 A 类条件和 N_2 个 B 类条件. 从每个 A 类条件中任选一个条件 (B 或者 C), 最多有 $2N_1$ 个组合, 每种组合与 N_2 个 b 类条件联立求解, 其中某些组合可能没有解, 任意一个解都能满足不产生延迟的条件. 这时任何一种组合中可能包含 $M(0 < = M < = N_1)$ 个 C 类条件, 其余的为 B 类条件.

C 类条件是两个矩阵必需在 DARAM 块, 将需要满足 C 类条件的所有矩阵存储器的大小相加, 相同的矩阵不重复累加, 结果为需要分配到 DARAM 的矩阵总数量, 当结果超过可得到的片上 DARAM 数量时, 这种条件组合下就没有解.

每个 B 类条件要求某两个矩阵必需在不同的块内, 由于存在多个 B 类条件, 事实上可能要求多个矩阵相互不在同一个块内, 例如要求矩阵 A_1 和 A_2 不在同一块内, 矩阵 A_3 和 A_1 不在同一块内, 矩阵 A_3 和 A_2 不在同一块内, 这实际上是要求 A_1, A_2, A_3 相互不在同一块内. 若有一组矩阵, 其中任何两个矩阵都必需分配在不同的存储器块内, 称为 B 类约束矩阵组. 若不存在一个矩阵, 要求与某个 B 类约束矩阵组中的所有矩阵都存在 B 类约束关系, 称这个组为最大 B 类约束矩阵组. 最大 B 类约束条件矩阵组中的矩阵数目就是分配这些矩阵所需的最少的存储器块数.

下面给出了以某个 B 类约束条件中的两矩阵为基础, 求解包含这两个矩阵的最大 B 类约束矩阵组的步骤.

(1) 取出其中一个 B 类约束条件, 不妨设为 $S_2 = (A_1, A_2)$, 初始化其标志为 1.

(2) 求出包含 (A_1, A_2) 所有可能的三矩阵组如 (A_1, A_2, A_3), (A_1, A_2, A_4), (A_1, A_2, A_5) 等, 由所有的三矩阵组构成一个集合, 记为 S_3, 并初始化 S_3 中的各个元素标志为 1. 若 S_3 为空集, 即没有包含 (A_1, A_2) 更大的 B 类约束组, 则停止以该条件为基础的继续搜索; 若 S_3 中仅仅包含一个元素, 这时这个三矩阵组为包含 (A_1, A_2) 最大 B 类约束矩阵组, 停止以该三矩阵组为基础的继续搜索. 只要 S_3 不为空集, 更新原两矩阵组标志为 0. 求包含 (A_1, A_2) 的三矩阵组的过程, 只需要检查出现次数不小于 2 的那些矩阵, 若检查 A_i, 只需判断是否存在限制 (A_i, A_1) 及 (A_i, A_2)

(3) 分别以 S_3 集合中的各个三矩阵组为基础, 检测是否存在包含此三矩阵的四矩阵组, 并初始化检测到的四矩阵组标志为 1, 由这些四矩阵组构成 S_4. 若检查到包含此三矩阵的四矩阵组, 将原来的三矩阵组标志更新为 0; 若 S_4 中仅仅包含一个元素, 停止以该四矩阵组为基础的继续搜索. 搜索四矩阵组的过程, 也可简化为: 简单检查 S_3 集合中的三矩阵组能否两两合并, 并初始化合并后的四矩阵组标志为 1. 若某两个矩阵组能够合并, 更新它们的标志为 0. 例如检查 (A_1, A_2, A_3) 和 (A_1, A_2, A_4) 能否合并, 只需检查是否存在限制条件 (A_3, A_4); 检查 (A_1, A_2, A_3) 和 (A_1, A_2, A_5) 能否合并, 只需检查是否存在限制条件 (A_3, A_5).

(4) 继续由四矩阵组搜索五矩阵组, 五矩阵组到六矩阵组…直到矩阵组的集合为空集或仅有一个元素, 停止搜索.

(5) 上述各矩阵组中标志为 0 已经被更大的矩阵组取代, 标志为 1 的矩阵组表示它为包含该矩阵组中各矩阵的最大矩阵了, 因此标志为 1 的矩阵组就是最大 B 类约束矩阵组.

分别以每个 B 类约束条件为基础, 搜索包含这两个矩阵

的最大 B 类约束矩阵组;由所有的最大 B 类约束矩阵组构成一个集合 S,删除 S 中相同的元素,比较各个最大 B 类约束矩阵组中的矩阵数量,包含矩阵数量最多的 B 类约束组中的矩阵数量就是分配这些矩阵所需要的最少片上存储器块数.首先把矩阵数最多的最大组中的各个矩阵分配到不同的存储器块中,然后按照 B 类约束矩阵组中矩阵数从多到少的顺序分配这个组中尚未分配的矩阵,对于具有相同矩阵数的组,先分配未分配矩阵较少的 B 类约束矩阵组中的矩阵.若 B 类约束的矩阵同时存在 C 类限制,则分配到 DARAM 上,否则优先分配到 SARAM 上;若 SARAM 上没有足够的空间,再分配到 DARAM 上.最后在 DARAM 上分配 C 类约束条件中的尚未分配的矩阵.

4 总结

上述数据存储器的分配方法只考虑了 TMS320C55x 中数据分配的主要方面,还有一些因素文中尚未涉及,如长整型数据的分配就必需考虑数据存储器地址的对齐问题,这时数据分配的求解变得更加复杂.可以将矩阵短整型的个数规定为偶数,以简化对齐问题,所以上述求解方法仍具有普遍的实用意义.

参考文献:

[1] Mahmut Kandemir, Ismail Kadayif, Ugur Sezer. Exploiting scratch-pad memory using Presburger formulas[A]. Proceedings of the 14th international symposium on Systems synthesis (ISSS' 01)[C]. New York, NY, USA: ACM press, 2001. 7 - 12.

[2] Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M. Balakrishnan, Peter Marwedel. Scratchpad memory: a design alternative for cache on-chip memory in embedded systems[A]. 10th International Workshop on Hardware/Software Co-Design (CODES' 02)[C]. New York, NY, USA: ACM press, 2002. 73- 78.

[3] Preeti Ranjan Panda, Nikil D Dutt, Alexandru nicolau. On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2000, 05(03): 682- 704.

[4] P R Panda, N D Dutt, A Nicolau. Efficient utilization of scratch-pad memory in embedded processor applications [A]. Proc European Design and Test Conference, Paris, March 1997[C]. Washington, DC, USA: IEEE Computer Society, 1997. 7- 11.

[5] Jeonghun Cho, Yunheung Paek, David Whalley. Fast Memory Bank Assignment for Fixed-Point Digital Signal Processors [J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2004, 09(1): 52- 74.

[6] Mazen A R Saghir, Paul Chow, Corinna G Lee. Exploiting dual data-memory banks in digital signal processors[J]. ACM SIGOPS Operating Systems Review, 1996, 30(05): 234 - 243.

[7] Ashok Sudarsanam, Sharad Malik. Simultaneous reference allocation in code generation for dual data memory bank ASIPs [J]. ACM Transactions on Design Automation of Electronic Systems, 2000, 05(2): 242- 264.

[8] Ruiner Leupers, Daniel Kotte. Variable partitioning for dual memory bank DSPs[A]. Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing [C]. Washington, DC, USA: IEEE Computer Society, 2001. 1121- 1124.

作者简介:

温淑鸿 男,1969 年生于湖北钟祥,清华大学电子工程系博士研究生,中国传媒大学信息工程学院教师,主要研究方向为图像编码,嵌入式系统,嵌入式多媒体处理器结构.E-mail: wenshuhong@sohu.com.

崔慧娟 女,1945 年 11 月出生,清华大学电子工程系,教授,主要专业及研究方向:语音编码,图像编码,多媒体终端等.
E-mail: cuihj@ee.tsinghua.edu.cn.

唐 昆 男,1945 年 10 月生,教授,博士生导师,主要研究方向为语音编码,多媒体终端,多媒体通信等.
E-mail: tangkun@tsinghua.edu.cn.