

基于低功耗 ASIP 的循环缓存的设计

胡 锦¹, 李新泽^{1,2}, 黑 勇², 于增辉², 陈黎明²

(1 湖南大学 物理与微电子科学学院, 湖南 长沙 410082; 2 中国科学院 微电子研究所, 北京 100029)

摘 要: 针对 ASIP 处理器的低功耗设计要求, 提出了多段式的循环缓存结构. 该结构与原有的循环缓存结构相比, 提高了缓存存储器的利用率. 本设计通过减少对主存储器的读操作和缓存存储器的写操作的方式来降低程序存储器的功耗. 在 SMIC 的 0.13 μm 工艺条件下, 将该结构应用于助听器处理器中, 并进行功耗验证. 分析表明, 该方法以较小的面积开销, 最高可存储器的功耗降低大约 50%, 有效的降低程序存储器的功耗.

关键词: 存储器; 低功耗; 循环缓存; 助听器处理器

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2011)05-0189-05

A Design of the Loop Buffering for the Low Power ASIP

HU Jin¹, LI Xin-ze^{1,2}, HEI Yong², YU Zeng-hui², CHEN Li-ming²

(1 College of Physics and Microelectronics, Hunan University, Changsha 410082, China;

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

Abstract: In this paper, a multi-segments loop buffering is proposed and it can be used in the design of the ultra low power application specific instruction set processor (ASIP). It is more efficient on the utilization of the loop buffer SRAM compared to the conventional loop buffer techniques. By decreasing the read operations of main program SRAM and the write operations of loop buffer SRAM, the scheme can greatly reduce the power consumption of the program memory. Based on SMIC 0.13 μm technology, the new loop buffering is validated in a hearing aid processor. The analysis result shows that the power consumption of the program memory can be reduced by more than 50% with only a little compromise of the area.

Key words: memory; lower power; loop buffering; hearing aids

1 引言

随着电子技术的发展,越来越多的微电子芯片应用于医疗设备中. 手持设备的发展,使得低功耗技术成为了现代微电子设计的研究热点. 在处理器的设计中,存储器的功耗占处理器总功耗绝大部分^[1],因此在不降低性能、不增大面积的基础上,寻求合理的设计方法,来降低处理器中存储器的功耗,成为了设计低功耗处理器的一个重点. 专用指令集处理器 ASIP(Application Specific Instruction Set Processor)是一种在应用于特定功能时,具有专用的指令集的一种处理器. 该处理器在功耗,速度,面积等方

面比通用处理器具有更加优越的性能^[2]. 文中基于低功耗助听器 ASIP 设计,通过分析不同大小的存储器模块之间的功耗差别,设计出多段式的循环缓存电路. 该结构是在原有的循环缓存电路的基础上,对结构进行了改进. 提高了缓存存储器的利用率,通过减少对主存储器读操作次数和缓存存储器的写操作次数,实现降低程序存储器的功耗.

2 循环缓存电路原理和结构

2.1 程序存储器低功耗设计对比

不论 ASIC 还是专用处理器,存储器功耗往往占据系统功耗的最大的部分. 因此目前关于功耗优

化的研究有很多是针对程序存储器的, 其中一种方式是通过将存储器分块, 减小读写操作时处于活动状态的存储器的规模来降低存储器的功耗^[3]. 这种方法的缺点是如果程序存储器中指令代码很多, 程序存储器的块数和规模就会增大, 这样会造成芯片面积的增大, 进而静态功耗增大. 另外一种方法是通过在主存储器和处理器取指令模块之间添加一级 cache 结构, 这种方式的优点是减少了对主存储器的读写次数, 降低了动态功耗, 提高了读写速度^[4]. 但是由于 cache 结构容量有限, 当出现命中率过低时, 起不到降低功耗的作用. 为了提高命中率, 往往会因为添加复杂的电路而增加较多的面积和静态功耗.

在处理器功耗的相关研究中可以发现, 由于程序中存在大量的循环操作, 80% 的运算功能是由 20% 的程序代码执行的. 而对存储器进行读取时, 每一条指令所耗费的动态功耗是一样的. 针对以上事实, Lea Hwang Lee 基于通用 DSP 处理器提出了循环缓存结构^[5]. 循环缓存结构是一种对 cache 结构进行改进的存储器优化电路, 该结构主要针对程序中大量的短循环操作, 通过存储短循环操作指令, 在读取短循环操作指令时, 关闭主存储器, 读取缓存存储器, 来实现对程序存储器系统的功耗优化. 在数字信号处理中, 许多算法包含了较多的循环操作. 因此通过在循环缓存中存储循环操作指令, 实现在不额外增大过多的面积和静态功耗的情况下, 有效的降低读程序存储器的功耗.

2.2 循环缓存原理

循环缓存结构降低功耗的原理主要是利用不同大小的存储器的读取功耗不同, 通过关闭大程序存储器读取小程序存储器的方式来实现程序存储器的低功耗设计. 例如在 $0.13\mu\text{m}$ 工艺条件下, 基于 ARM Artisan 的存储器自动生成工具得到的存储器 IP 核中, 在典型工作情况下 (1.2V 、 25°C), 容量为 64×32 的存储器单次读取的功耗电流为 $9.23\mu\text{A}$, 容量为 1024×32 的存储器单次读取的功耗电流为 $44.5\mu\text{A}$. 因此可以看出, 在设计中利用小存储器代替大存储器进行读取可以有效的降低整个系统的功耗.

图 1 中所示是循环缓存与主程序存储器结构. LB 代表整个循环缓存电路模块, 包括循环存储器 (Lb_memory) 和循环缓存控制模块 (LC). 程序存储器输出的指令代码进入 LB 模块内部, 通过 LC 控制, 识别出 repeat 循环指令并将随后的循环体代码

存储在 lb_memory 中. 系统在第一次进入循环操作时, 对 program memory 输出的指令代码进行读取, 将输出的指令分别输入到 MUX 中和 LB 模块中, 当对循环体进行第二次操作时, LC 会关断 program memory, 指令会从 LB 中直接输出, 并且 LC 控制 MUX, 选择从 LB 中输出的指令. 当地址信号超出了循环体的地址范围时, LC 模块会关闭 lb_memory, 开启 program_memory. MUX 模块会选择 program_memory 输出的指令代码并将其传输到 FETCH 模块.

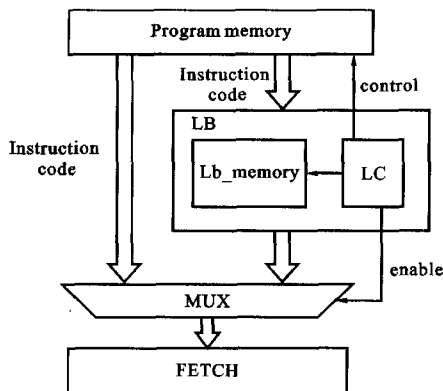


图 1 循环缓存结构示意图

2.3 循环缓存设计方法对比

目前在循环缓存结构的研究中, 主要集中于两大类的方法, 第一类方法是从电路结构入手, 通过在硬件电路设计中添加相关的计数器和寄存器, 根据指令集中的特定指令代码来实现对循环体的识别和存储^[6-8]. 另外一类方法是从处理器的编译器入手, 通过分析相关算法, 来合理地规划循环控制标签的插入位置, 从算法实现的角度利用缓存结构实现功耗优化^[9]. 以上两种都是基于在通用 DSP 处理器中设计应用的, 而在低功耗 ASIP 设计中, 由于处理器规模的缩小和对低功耗要求的提高, 利用设计编译器来规划插入循环控制标签的方式已不能达到极低功耗的要求. 所以针对 ASIP 设计来说, 需要特定的操作运算, 通过设计硬件电路来实现对 ASIP 缓存结构的更加进一步的优化.

3 低功耗 ASIP 中循环缓存设计

3.1 循环指令格式

在 VLSI 设计中, ASIP 作为应用于某一特殊方面的处理芯片, 其在特定应用场合有着特定的性能要求. 在低功耗 ASIP 设计中, 许多传统的通用 DSP 处理器的低功耗设计方法并不适用, 由于采用了专用指令集的设计, 在设计循环缓存结构时, 为了实现

零开销循环指令,就需要对 ASIP 针对指令集进行硬件电路设计,避免插入为循环指令设计的特定标签指令。

图2所示是 repeat 指令的指令代码格式,opcode 代表了循环指令的指令识别码,register number 代表寄存器编号,在该寄存器中存储了循环体循环的次数,repeat inst number 代表循环体中循环指令的个数。在硬件电路设计中,循环缓存结构通过对 repeat 指令进行指令代码的部分提取来实现对循环结构的控制。循环缓存结构中的 LC 模块的启动是根据对 repeat 指令的 opcode 的识别,当识别到 opcode 为 repeat 指令时,LC 中状态机进入工作状态。缓存结构中记录 repeat inst number 数据和 repeat 指令的地址指针。通过记录指令数和地址指针,来实现对缓存存储器的开启和关断操作。

opcode	Register number	Repeat inst number
--------	-----------------	--------------------

图2 循环指令代码格式

3.2 多段式循环缓存结构设计

在传统的缓存设计中,大都是一次仅仅存储一个循环体,在下次执行相同循环体操作时,需要重新向缓存中写入循环体,这样造成了额外的功耗浪费。并且在缓存存储器中,仅仅存储了一个循环体,会造成对存储器利用率不够高。文中提出的多段式循环缓存设计可以提高存储器的利用率,可以提高读取缓存中存储的循环体的命中率,能够在传统结构的基础上更进一步的降低功耗。

多段式循环缓存结构实际上是将缓存存储器分成多段,在每一段中分别存储一个循环体,配合 LB 模块进行控制操作,由于采用多段存储,避免了临近相同循环的重复写操作。利用该方法可以提高缓存存储器的利用率,减少对缓存存储器的写操作,降低整体程序存储器系统的功耗。

通过对比深度为 32、64、128 的缓存存储器可知,深度为 32 的缓存存储器中容量过小,命中率过低,从功耗的角度衡量,深度为 64 的存储器单次读取的功耗相对于深度为 32 的存储器增加了 6%,而深度为 128 的存储器单次读取的功耗比深度为 32 的读取功耗增加了 20%,因此综合考虑,选取深度为 64 的程序存储器作为缓存存储器。

通过分析本设计的相关算法的指令,可以发现,循环操作中最大的循环体为 31 条指令,因此在设计缓存存储器时,就需要考虑如何更加合理的划分存储器中各段的空间大小。由于包含多于 16 条指令的

循环体所占比例很小,大部分循环体的指令数都在 7 条到 14 条之间,因此如果在存储器中开辟的缓存空间均为 32 时,依然会造成存储空间的浪费和读取存储器命中率过低。因此,综合考虑,文中采用将缓存存储器按照 16:16:32 的空间分布划分为三个存储段的结构,这种设计在提高存储器利用率和命中率上比较合理。

图3所示是本设计提出的多段结构的循环缓存电路结构图。由于缓存存储器分成了三个存储段,在 LB 模块中,增加了三个寄存器分别为 pc_store1, pc_store2, pc_store3。这三个寄存器分别记录了最近三次存储的循环体对应的 repeat 指令的地址。当下一次 LB 遇到 repeat 指令时,LB 会将 repeat 的地址和三个寄存器中的地址进行对比,如果和其中一个寄存器的地址相同,表示在缓存存储器中已经记录了该循环的循环体代码,可以直接关断主存储器,并从 LB_memory 读取指令。由于循环体存储到缓存存储器中的位置与三个寄存器存在一定的映射关系,因此在读取某段已缓存的循环体代码时,缓存存储器的读取地址需要加上一个地址偏移量。在本设计中当 pc_store1 命中,则地址偏移量为 0,当 pc_store2 命中,则地址偏移量为 16,如果 pc_store3 命中,则地址偏移量为 32。

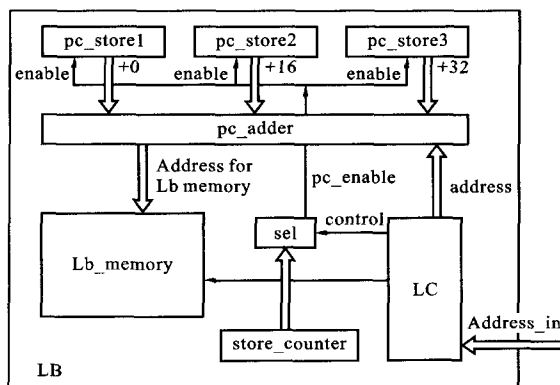


图3 三段结构循环缓存电路结构图

在遇到循环指令操作时,如果已存指令没有命中,则需要对新的循环缓存进行存储。若新循环体包含的指令多于 16 条,直接将 repeat 指令地址存储在 pc_store3 寄存器中,并以 32 为起始地址在缓存中存储该循环体。如果循环体小于 16 条指令,则根据计数器 store_counter 输出的存储序号,来判断应该写入的存储器段,并进行存储,同时将相应的地址写入相应的地址寄存器中。

图4所示是多段式循环缓存控制模块的状态转换流程。在状态机设计中,加入了 LB_WAIT 状态,

目的是实现对地址信号的映射和对已存循环体的命中判断. 当输入的 repeat 地址和 pc_store 相同时, 直接进入 LB_LOAD 状态, 进行读取操作.

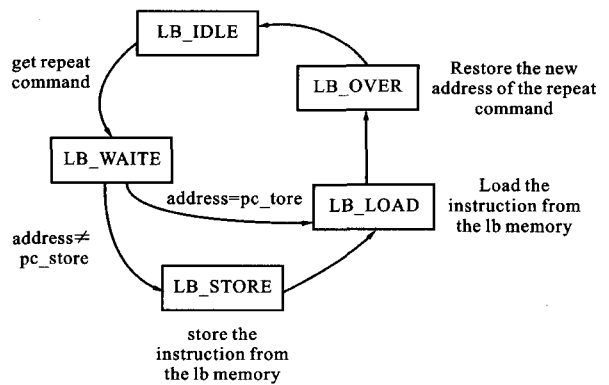


图 4 三段式循环暂存状态流程图

当 repeat 地址和 pc_store 中的地址不相同时, 表示没有命中, 进入 LB_STORE 状态, 进行循环体指令的存储, 然后再进入 LB_LOAD 状态, 进行指令读取. 当完成读取操作后, 在 LB_OVER 状态中, 对 pc_store 存储的地址进行更新.

3.3 数据分析

文中所采用的主程序存储器和缓存存储器是基于 ARM Artisan 的存储器自动生成工具 Artisan Generator 生成的. 分别为单端口的 pm1024×32 存储器和 pm64×32 存储器. 用硬件描述语言 Verilog 设计缓存结构电路, 然后用 Synopsys 公司的 Design Compiler、VCS、PrimePower 等工具完成对循环缓存结构的综合、仿真与门级网表的功耗分析.

表 1 中给出了对采用传统循环缓存结构和三段缓存结构的面积对比和功耗的比较, 分析是基于 SMIC 的 0.13μm 工艺, 工作电压为 1.2V, 温度为 25℃, 工作频率为 10MHz.

表 1 三种存储结构面积对比和功耗对比结果

存储结构	面积/μm ²	功耗/mW
原始存储结构	151144.6250	0.2024
基本缓存结构	175347.1250	0.1064
多段缓存结构	180612.4375	0.0870

通过表 1 可以看出, 采用缓存结构后, 程序存储器部分的面积增大了大约 16%, 而功耗降低了大约 50%. 而采用了多段缓存结构后, 在原有的缓存结构的基础上, 芯片面积增大了大约 3%, 而功耗却比采用基本缓存结构降低 20% 左右. 可以看出使用缓存结构可以有效的降低程序存储器的功耗, 而多段式的缓存结构, 相较基本缓存结构而言, 功耗优化效果更佳.

4 循环暂存结构在低功耗 ASIP 处理器中的应用验证

本缓存结构主要是应用于极低功耗处理器中, 该处理器是针对医用助听器所设计的.

医用数字助听器主要包括三个功能: 多通道频率补偿, 自适应消噪, 自适应反馈消除. 文中设计的数字助听器, 在多通道频率补偿中, 采用加权叠接相加 (Weigthed—OverLap Add, WOLA) 算法实现信号的多频率通道分离, 采用宽动态压缩 (WDRC) 算法对语音信号进行频率补偿, 其中在 WOLA 运算中, 包含了 64 点的 FFT 运算和 IFFT 运算.

由图 5 可知, 多通道频率补偿 (包含 WOLA 和 WDRC) 是数字助听器的核心, 而噪声消除是在存在噪音的情况下实现语音增强. 在实际的指令操作中, 可以设置选择使用噪声消除或者关闭噪声消除.

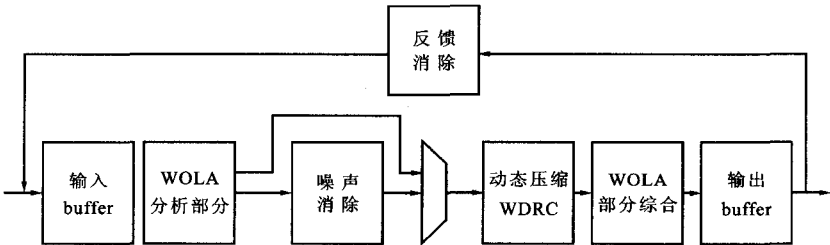


图 5 助听器处理算法结构示意图

表 2 中表示了助听器语音处理各算法的运行周期数及其比例关系, 可以看到在以上操作中 WOLA 算法和 NOISE_REDUCE 算法占据了极大的比例. 其中 WOLA_A 和 WOLA_S 算法中最主要的运算是 FFT 运算和 IFFT 运算, FFT 和 IFFT 运算的运

行周期为 6042, 占总的运算指令周期的 34.8%.

在 FFT/IFFT 中包含了大量的 repeat 指令操作, 循环体的指令有 16 条, 根据 64 点 FFT 的 6 级运算, 分别执行次数为 32×1、16×2、8×4、4×8、2×16、1×32, 即在循环体中, 有 16 条操作指令, 每个

循环要执行 32 次,并且由于 FFT 和 IFFT 运算是 6 级的,因此相邻的循环操作是相同的,执行 6 次. 在 FFT 和 IFFT 操作中,循环操作指令有很大的重复性,因此在实际使用三段式进行功耗优化时,既可以降低主程序存储器的功耗,也可以降低缓存存储器的功耗.

表 2 助听器语音处理各算法运行周期比例

操作	运行周期	百分比(%)
IN	32	0.2
OUT	64	0.4
WOLA_A	3597	20.7
NOISE_REDUCE	7948	45.7
WDRC	1397	8.1
WOLA_S	4365	25.1

基于 SMIC 的 0.13 μ m 工艺进行功耗分析,结果如表 3 所示. 当处理器仅仅进行 WOLA 运算时,采用多段式缓存结构将使程序存储器的功耗降低大约 60%. 而在加入了 WDRC 运算后,多段缓存结构可以降低大约 53%的功耗. 在整个助听器处理器加入消噪处理后,利用多段式缓存结构可以降低 45%的功耗. 无论哪种运算,功耗优化效果均比采用传统的循环缓存结构要好. 另外,在对电路进行综合后得出,利用多段式的缓存结构,增加的面积占整个处理器的 3%,而由表 3 可知采用多段式缓存优化后功耗减少了 0.16mW,占总芯片功耗的 20%,功耗优化特性很可观.

表 3 运行各算法时,程序存储器的功耗比较

算法选择	单程序 存储器 /mW	基本循 环缓存 /mW	多段式 循环缓存 /mW
WOLA	0.2024	0.1064	0.0870
WOLA+WDRC	0.2303	0.1285	0.1091
WOLA+WDRC+ NOISE_RED_UCTION	0.3406	0.2088	0.1895

5 结束语

文中依据极低功耗 ASIP 设计的性能要求,对已有循环缓存结构硬件电路提出了改进,设计出了多段式的循环缓存结构. 该结构提高了缓存存储器的利用率,并且减少了对主程序存储器的读操作,对程序存储器功耗进行了更进一步的优化. 在 SMIC

的 0.13 μ m 工艺条件下,该结构应用于医用极低功耗处理器中. 在芯片面积仅仅增加 3%的前提下,将程序存储器的功耗降低了 45%,整体芯片功耗降低了 20%,充分证明了该种缓存结构对处理器功耗的降低有非常好的效果.

参考文献:

[1] Karandikar A, Parhi K K. Low power SRAM design using hierarchical divided bit—line approach[C]// Computer Design: VLSI in Computers and Processors, Austin, TX: SJR, 1998: 82—88.

[2] Liu D. ASIP (application specific instruction—set processors) design[C]// IEEE 8th International Conference on Changsha, Hunan: IEEE, 2009: 16—16.

[3] Mai Songping, Zhang Chun, Zhao Yinxin, et al. An application—specific memory partitioning method for low power[C]// Proceedings of the 7th International Conference on ASIC, Guilin, China: 2007: 221—224.

[4] Gordon Ross A, Cotterell S, Vahid F. Exploiting fixed programs in embedded systems: a loop cache example [J]. IEEE Computer Architecture Letters, 2002, 1(1): 2—2.

[5] Lea Hwang Lee, Moyer B, Arends J. Instruction fetch energy reduction using loop caches for embedded applications with small tight loops[C]// Proceedings of the International symposium on Low Power Electronics and Design, San Diego, CA: ACM, 1999: 267—269.

[6] Jayapala M, Barat F, Vander Aa T. Clustered loop buffer organization for low energy VLIW embedded processors[J]. Computers, 2005, 54(6): 672—683.

[7] 扬名, 于立新. 基于标志压缩的低功耗指令 cache 设计[J]. 微电子学与计算机, 2008(5): 91—94.

[8] 刘宗林, 马卓, 鲁建壮, 等. 一种并行指令 cache 的设计与实现[J]. 微电子学与计算机, 2007(12): 147—149.

[9] Sias J W, Hunter H C, Hwu W. Enhancing loop buffering of media and telecommunications applications using low—overhead predication [C]// Proceedings of the 34th Annual International Symposium on Microarchitecture, Washington, DC: IEEE, 2001: 262—273.

作者简介:

胡 锦 男, (1964—), 硕士, 教授. 研究方向为电子系统与专用集成电路.

李新泽 男, (1985—), 硕士研究生. 研究方向为低功耗处理器与低功耗医疗电子芯片设计.

黑 勇 男, (1974—), 研究员, 博士生导师. 研究方向为可重构数字信号处理器、低功耗数字基带芯片的设计等.