

基于 TTA 架构对 H.264 帧内预测算法 4×4 模式优化设计

赵丽丽, 谢 憬, 毛志刚

(上海交通大学 微电子学院, 上海 200240)

摘 要: 描述了一种基于传输触发结构(Transport Triggered Architecture, TTA)的专用处理器(Application Specific Instruction Set Processor, ASIP), 用于实现 H.264 帧内预测算法 Intra 4×4 预测模式. 仔细分析了 Intra 4×4 算法的特性和应用通用 DSP 时的缺陷, 介绍了在 MOVE Framework 开发环境下实现 Intra 4×4 算法的专用处理器设计流程及方法. 性能分析显示, 与通用 DSP 相比, 基于 TTA 技术的 ASIP 具有资源消耗少, 处理效率高的特点, 符合嵌入式视频处理系统的要求.

关键词: 传输触发结构; 专用处理器; H.264; 帧内预测算法

中图分类号: TN919.81

文献标识码: A

文章编号: 1000-7180(2010)10-0045-05

Optimization of Intra 4×4 Prediction of H.264 Based on TTA Architecture

ZHAO Li-li, XIE Jing, MAO Zhi-gang

(School of Microelectronics, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: This paper describes an Application Specific Instruction Set Processor (ASIP) based on the Transport Triggered Architecture (TTA) which realized the algorithm of Intra 4×4 mode of H.264. The paper takes an insight into the algorithm and provides the reader with the method and flow of building the processor within MOVE framework. Compared with the general purpose DSP, the ASIP based on TTA with only a few resources can achieves much higher performance.

Key words: TTA; ASIP; H.264; Intra prediction

1 引言

H.264 标准采用了视频编解码理论的最新研究成果, 帧内预测编码算法就是其中之一. 和以往标准(如 H.263+、MPEG-4)采用变换域的预测不同, H.264 标准充分利用图像的空间相关性, 利用帧内宏块周围的像素值进行空域预测, 减少了空域和频域的冗余度而获得更高的压缩效率^[1]. 但算法的复杂度大大增加. 一些文献针对宏块内部各种特征提出了不同的帧内预测模式的快速选择算法, 以减少宏块的率失真计算次数, 在不影响编码效率的前提下提高了算法的速度. 在模式预测方面, 通过改变模式预测中的参考像素, 简化预测模式的计算方式, 更改预测模式的顺序等方法, 进一步降低了预测算法

的复杂度^[2].

上述各类优化方法均是从算法层面上考虑, 根据图像某些特征对算法流程进行简化而避免遍历所有的可能预测模式, 以此实现算法复杂度的降低. 但是这些算法中, 宏块内像素值的预测, 都是无法减免的. 因此, 本文以单个宏块内像素为研究对象, 分析预测算法的特性, 提出一种基于传输触发结构(Transport Triggered Architecture, TTA)的专用处理器. 基于 TTA 技术的专用处理器可根据目标应用进行裁剪, 结构简单却可达到很高的效率, 是一种针对特定应用的比较理想的处理器设计方案.

2 Intra 4×4 帧内预测模式概述

H.264 帧内预测对于亮度信息的预测有两种类

型,分别是宏块以 4×4 大小和以 16×16 为大小,其中 4×4 块有 9 种预测模式,如图 1 所示,且其中直流(DC)模式根据参考像素的特点又有四种不同情况,而 16×16 有 4 种预测模式.通常, 4×4 亮度块适合于纹理比较丰富的区域,而 16×16 块适合于较为平坦的区域^[3].鉴于在目前的图像,尤其是高清图像处理中多用 4×4 空间预测模式,本文针对此模式进行研究设计.

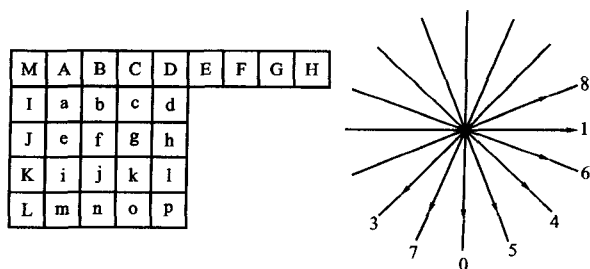


图 1 Intra_4×4 的 9 种预测模式

H.264 的 Intra4×4 模式帧内预测采用子块左边和上边相邻像素构成的 L 型参考像素集进行空间预测,如图 2 所示.其中, $a - p$ 是 4×4 宏块的 16 个组成像素, $A - M$ 是已编码的参考像素,来自于当前被预测块上方及左侧块像素值.在像素预测时用到的算法包括乘法、加法和移位等,以图 2 中第三种模式对角线向左下为例,其各像素求值如下面公式所示:

$$a = (A + 2B + C + 2) \gg 2; \quad (1a)$$

$$b = e = (B + 2C + D + 2) \gg 2; \quad (1b)$$

$$c = f = i = (C + 2D + E + 2) \gg 2; \quad (1c)$$

$$d = g = j = m = (D + 2E + F + 2) \gg 2; \quad (1d)$$

$$h = k = n = (E + 2F + G + 2) \gg 2; \quad (1e)$$

$$l = o = (F + 2G + H + 2) \gg 2; \quad (1f)$$

$$p = (G + 3H) \gg 2; \quad (1g)$$

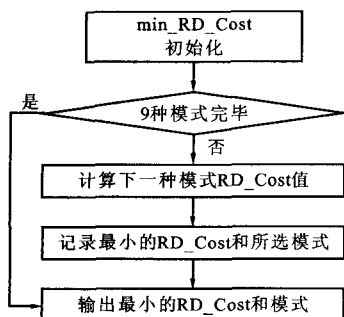


图 2 Intra4×4 模式选择算法

考虑每帧像素大小为 256×256 , 每个像素都要进行预测, 则被预测像素共有 65 536 个. 在 4×4 预测模式下, 每个宏块通常要遍历 9 种不同预测模式

以确定最优编码模式, 相当于是对 589 824 个像素进行求值, 每幅图像又会包含很多帧, 由此可见其运算规模之大. 同时, 从上面公式也可发现, 算法种类较为单一, 因此若是使用通用处理器实现这一算法, 必然会出现处理器大部分功能部件闲置, 而少部分功能部件使用过于频繁而无法高效完成操作的情况. 因此, 在帧内预测算法中, 一种针对这种特性而设计的专用处理器, 不仅能够减少一个宏块计算时所需要的开销, 对提高整个算法的效率也将起到至关重要的作用.

在确定某宏块的帧内预测采用何种方式时, 采用 RDO(rate distortion optimization) 技术, 即分别计算被预测宏块在各种模式下对应的 RD(rate distortion) 代价 (RD_Cost), 选择对应着最小 RD_Cost 的预测模式作为被预测宏块的最佳预测模式. 代价函数求解见式 (2), 其中 SSD(sum of squared difference) 表示当前块与重建块的差值平方和, $\lambda_{\text{mode}} = 0.85 \times 2^{(QP-12)/3}$, (QP 为量化参数), Rate 表示熵编码后的码率.

$$\text{RD_Cost} = \text{SSD} + \lambda_{\text{mode}} \times \text{Rate} \quad (2)$$

Intra_4×4 预测模式选择流程参考图 2, 首先对 RD_Cost 值进行初始化, 然后遍历 9 种预测模式, 选择 RD_Cost 值最小的所在模式为最佳预测模式, 输出此模式及最小的 RD_Cost 值.

3 TTA 架构下设计实现

由 Corporaal 等人提出的传输触发体系结构 TTA, 可被看做是传统 VLIW 体系结构的一个超级, 是 SIMT(Single Instruction Multiple Transport) 类型的体系结构^[4]. 与传统的计算机体系结构, 如 CISC, RISC, VILW, Superscalar 等相对比, 在传输触发架构中规定运算操作作为数据传递的一个副作用, 所有的操作转化为数据的移动.

传输触发架构如图 3 所示, 包含一组执行预定功能的功能单元(Function units), 作为通用寄存器的寄存器单元(Register units)和一些控制逻辑(Control Logic), 以及将这些单元进行通信的互联网络(Interconnection network).

设计者可以根据目标应用改变功能单元所执行功能的种类, 各功能单元的流水深度, 存储器个数及大小, 互联网络的连接方式以及互联网络中总线的数量等等, 以实现不同的应用范围以及复杂度的设计. 所以, 传输触发架构是一种相对于应用可裁剪的理想架构.

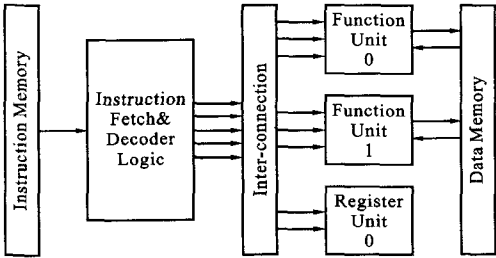


图3 传输触发架构

文中TTA架构的设计开发工具为Move Framework,这是一套半自动生成ASIP的工具集.在MOVE Framework下设计基于TTA的ASIP的基本流程可分为以下几个步骤:

- (1) 对要实现的目标应用进行C/C++描述,这一步骤属于软件层面;
- (2) 在MOVE下对(1)中输入进行编译,利用MOVE Framework中集成的顺序代码仿真器进行仿真,得到顺序代码相关信息,为系统架构设定提供参考依据;
- (3) 编辑系统架构描述(machine description file,M-FILE),这一步中要在硬件资源和性能间进行权衡;
- (4) 利用MOVE Framework中并行代码生成工具scheduler生成并行代码,作为(3)中分析的依据以及(4)中的输入;
- (5) 生成rtl代码,并进行仿真;

3.1 目标应用输入

要生成目标处理器首先需要用C/C++对目标应用算法进行描述,即要对图2所指示的Intra_4×4的流程进行C/C++描述.

设计中代码的编撰主要参考H.264开源代码x264,先将全部代码在VC下进行调试,再将decoder中predict部分功能代码提取出来,并参考相关算法介绍^[5],完成基于RDO技术进行模式选择的Intra_4×4算法.

需要注意的是,在输入的代码中还应包括测试用代码.

3.2 顺序代码生成

MOVE架构中的顺序代码生成器(sequential code generator,SCG)是基于GNU的工具链,包含了编译器、汇编器、链接器等,它可将目标应用程序转化为等价的顺序TTA程序代码.顺序的TTA程序代表了从原有的目标应用到一种理想的目标架构的映射,并且可被后续的仿真器与调度器(Scheduler)利用为处理器的生成提供必要的信息.

SCG提供了单命令行接口,对于C语言的目标应用使用gcc-move命令,并可配合很多选项控制顺序代码的生成.

集合经过SCG编译器编译得到的顺序代码输入到MOVE Framework集成的仿真器中,将得到一份性能文件,其中包括了对所执行指令的数目(Execution count)、性能(Performance for 100 MHz and 1 CPI)、立即数位使用率(Immediate Bits Utilization)、通用寄存器使用率(Average number of Generous Purpose Registers live)、并行度上限(Parallelism upper bounds)、频率最高的20种操作(Operation count top 20)以及20种最常用的数据传输(Data transfer top 20)等信息的描述.

Intra_4×4程序经过顺序代码生成器得到的性能文件部分内容见表1~表4.其中表1中Moves指数据的一次搬移,如在表4中add_r->add_o即是数据从加法器的操作数寄存器搬至加法器结果寄存器.

表1 指令计数

# Moves	8 930
# Cycles	3 765
# Instructions	648

表2 执行效率

Performance for 100 MHz and 1 CPI (w/ squashed operations)	
Total megaops/second	99.044

表3 频率最高的四种运算

add	1 323	ld	583
st	396	gt	302

表4 频率最高的四种数据移动

add_r->add_o	1 093
add_r->ld_t	373
add_r->gt_o	283
shr_r->st_t	276

从上面所列数据可看出,Intra_4×4转化为TTA顺序代码共需要进行8 930次数据移动,需周期3 765个,需指令648条,100MHz下每秒可完成99.044兆条指令,因此Intra_4×4程序需要7μs.程序中使用最多的操作为加法、数据的存储以及比较,因此在设计系统资源时这三种功能单元的数量或互联就需要相对多一些.最后一张表格也显示数

据移动多是在这些运算单元中进行。

3.3 M-FILE 创建

M-FILE 是对目标处理器的架构完整的进行描述的文件,包含功能单元、寄存器组、立即数以及互连网络等信息,编辑 M-FILE 是实现 MOVE 处理器关键的一步.搭建系统时要始终把握一个原则,即在保证功能正确性的前提下,在系统资源占用和性能的最优之间进行平衡。

建立 M-FILE 可以有两种方法:由简入繁和繁而入简.由简入繁这一方法中最初的 M-FILE 内容很少,当不满足要求时逐步增加功能单元和总线的数目以实现功能,这种方法给出的系统资源上是最少的,但其实现过程是根据资源耗尽的错误提示,在系统中添加元件,但由于错误定位不细致,无法判断是哪种资源发生短缺.通常要经过多次尝试.文中采用由繁入简的形式.先给出一个资源富余的系统文件,根据并行代码分析器给出的性能文件对系统资源进行删减.这种方法给出的系统不一定是最小,但是可以借助 MOVE Framework 中集成的工具,提高了工作效率.文中采用由繁入简的设计方法。

M-FILE 可被分为六个部分,分别是总线(Move Busses)、功能单元(Function Units)、功能单元与总线的连接情况(Sockets)、立即数单元(Long Immediate)、通用寄存器(Register Units)以及指令单元(Instruction Units).每个部件的描述都由各自的關鍵字开始,并以花括号界定范围.例如下面一段代码是 M-FILE 中对一个加法功能单元的描述:

```
FunctionUnits
{
    fu2 always,2,{fu2_o},{fu2_t},{fu2_r},{add};
}
```

其实现的功能为: #2 cycles c = a + b,always 关键字表明始终支持流水操作。

参考后面并行代码分析器得到的性能报告,对 M-FILE 进行增减,最终得到的系统架构如下:6 条 32 位总线、3 个通用寄存器、2 个立即数单元以及 7 个功能单元.其中 7 各功能单元包含加法器两个、存取单元两个、移位器、比较器、求绝对值单元各一个。

3.4 并行代码生成

生成并行代码是由 Scheduler 完成的,它也集成在 MOVE Framework 中. Scheduler 读取设计者对目标处理器的描述,将顺序的 TTA 程序转化为并行 TTA 代码.这一转化过程,主要是对互连网络、功能单元及通用寄存器等根据指令进行分配。

Scheduler 也是通过命令行进行操作,且需要设计者提供顺序 TTA 程序、M-FILE 两个文件。

Scheduler 也会生成系统性能的描述文件.较之在顺序代码分析时所产生的性能文件,这一文件中又包含了各个总线、功能单元、寄存器以及立即数等的使用情况.表 5 列出系统中总线的利用率的百分比,B1~B6 代表了六条总线。

表 5 Move busses 利用率

	B1	B2	B3	B4	B5	B6
In	84	74	59	51	35	27
Out	76	74	53	51	31	26

除各单元使用情况外,此性能文件还提供了对代码循环展开、分支预测、重新排序等信息.这是系统设计优化的重要参考。

3.5 Rtl 代码生成

MOVE Framework 中集成的 MOVEgen 可用来生成 VHDL 代码.且分为三个文件,描述总线及互连关系的 interconn. vhd、包含控制信息的 control. vhd 和对算法核心部分进行实现的 movecore. vhd。

结合系统内自带的时钟模块、程序寄存器与数据存储器、仲裁器等硬件模型,可形成一个完整的硬件系统,并可在 Modelsim 进行编译仿真.至此,实现了基于 TTA 技术的 ASIP。

4 性能分析

4.1 峰值性能

在生成 TTA 架构的过程中需要不断对系统资源进行调整,亦即对 M-FILE 进行修改,在性能和资源上进行权衡。

经验证当通用寄存器为 3 个、立即数为 2 个时,本系统可以达到较好的性能.在此基础上,减少通用寄存器(GPR)或立即数单元(I)都将使系统性能下降近 10%.表 6 所示为 6 条总线的系统中减少一个通用寄存器或立即数单元时,总的移动次数(Moves)以及周期数所发生的变化,第三列和第五列指示了增加的百分比。

表 6 性能比较

	Moves Incr		Cycles Incr	
3GPR&&2I	8 389		2 536	
减 1 个 GPR	9 089	8.3	2 722	7.3
减 1 个 I	8 402	0.2	2 706	6.7

总线数量的变化对系统性能影响也很大.图4显示了当总线数量发生变化时,数据移动次数(右)以及周期数(左)随之变化的情况.1、2、3、4种情况分别代表总线数为4、6、7、8.

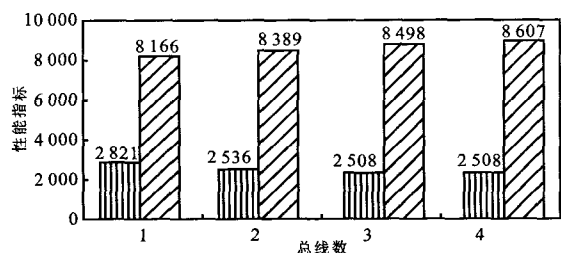


图4 不同总线数目下性能图

从图4可看出,随着总线数的减少,数据移动次数也随之减少,这主要是因为省去了数据在不同总线间的移动.随着总线的增加,算法所需要的周期数呈减少趋势,因为在数据传输中等待几率降低.

功能单元的数目达到某一数目时再增加功能单元的数量也不会带来系统性能的提高,因为数据运算不是均分在各个功能单元中,当一部分功能单元可以满足调度要求时,其余的功能单元将闲置.基于以上各项性能的分析比较得到3.2节中所述系统结构.

4.2 与通用 DSP 性能比较

为考察 TTA 架构下 $\text{Intra}4 \times 4$ 算法实现的效率,以 TI 公司的 C5402 作为参照. C5402 DSP 具有先进的改造型哈佛结构,40 位算术逻辑单元 (ALU),1 个 17×17 乘法器和 1 个 40 位专用加法器.

将 $\text{Intra}4 \times 4$ 算法输入到 CCS 仿真器中,得到运行周期数为 15 469.

根据设定,在 TTA 中每个加法执行需要两个周期,但在流水操作下可以认为,TTA 可以像

C5402 一样,一个周期给出加法的结果.考虑到 TTA 中有两个加法器,因此,执行 $\text{Intra}4 \times 4$ 算法时 TTA 处理器的所需要的周期数为 C5402 的 32.3% ($2\,536 \times 2 / 15\,469 = 0.328$),即效率调高三倍.

5 结束语

基于 TTA 的处理器设计时,可以根据具体的目标应用对系统进行裁剪,避免了系统资源的冗余而节约了面积,且由于 TTA 架构在数据调度上的灵活性,系统有可以实现很高的效率;系统设计的起点为对目标应用的高级语言描述,并可生成硬件描述语言,减少了设计所需时间.鉴于多媒体应用中所表现出的数据量大、操作繁琐等特点,以 TTA 架构进行处理器设计将是一种更为理想的选择.

参考文献:

- [1] 陈光化,翟海华,石旭利,等. H.264 编码器中 $1/4$ 像素精度插值算法的 VLSI 实现[J]. 微电子学与计算机, 2008,25(2):176-180.
- [2] 李霞,黄玄,周莉,等. AVS 帧内预测算法分析及 VLSI 实现[J]. 微电子学与计算机,2009,26(1):29-33.
- [3] 李世平,蒋刚毅. 快速帧内预测模式选择新方法[J]. 电子学报,2006,34(1):141-146.
- [4] Hoogerbrugger J. Code generation for transport triggered architectures[D]. The Netherlands: Delft University of Technology, 1996.
- [5] Bojun Meng, Oscar C Au. Fast intra-prediction mode selection for 4×4 Blocks in H.264[C]//IEEE International Conference on Acoustics, Speech, and Signal Processing. China, Hong Kong, 2003:389-392.

作者简介:

赵丽丽 女,(1984-). 研究方向为系统芯片设计.

谢 憬 硕士研究生. 研究方向为 SoC 项目的开发与研究.

(上接第 44 页)

- [7] 黄岚,王康平,周春光,等. 粒子群优化算法求解旅行商问题[J]. 吉林大学学报:理学版,2003,41(4):477-480.
- [8] 曹平,陈盼,刘世华. 改进的粒子群算法在旅行商问题中的应[J]. 计算机工程,2008,34(11):217-221.

作者简介:

王玉亭 男,(1980-),硕士,讲师. 研究方向为软件工程、智

能优化.

孙 剑 女,(1979-),硕士,讲师. 研究方向为计算机网络、人工智能.

李俊青 男,(1976-),硕士,副教授. 研究方向为 P2P 网络、智能信息处理.

任立群 男,(1975-),工程师. 研究方向为数据库.

潘全科 男,(1971-),博士,教授. 研究方向为优化调度理论与方法、智能优化算法及应用.