

一种高性能 HEVC 整像素运动估计硬件设计

刘凯丽, 陈 成, 李 焱, 陈 松

(中国科学技术大学 电子科学与技术系, 安徽 合肥 230027)

摘 要: 针对整像素运动估计提出了一种适合硬件实现的运动估计算法, 并设计了硬件架构. 通过在不同划分深度的编码单元中复用计算单元, 大大减少了硬件资源. 在 TSMC 90nm 的工艺下, 综合结果表明最高频率可以达到 377 MHz, 在搜索范围为 ± 64 时, 能够达到超高清视频图像 $3\,840 \times 2\,160 @ 60\text{ f/s}$ 的实时处理速度.

关键词: 视频编码; HEVC; 运动估计; 硬件架构设计

中图分类号: TN919. 81

文献标识码: A

文章编号: 1000-7180(2017)10-0081-05

A High Performance Hardware Design for HEVC Integral Motion Estimation

LIU Kai-li, CHEN Cheng, LI Yao, CHEN Song

(Department of Electronic Science and Technology, University of Science and
Technology of China, Hefei 230027, China)

Abstract: In this paper, a motion estimation algorithm suitable for hardware implementation is proposed for integer motion estimation, and the hardware architecture is also presented. By reusing computing unit in the CU depths, the hardware resource is greatly reduced. Synthesized results in the TSMC 90nm show that the frequency of presented architecture can reach 377 MHz and the throughput can achieve $3\,840 \times 2\,160 @ 60\text{ f/s}$ real-time processing in the search range of ± 64 , which meets the requirement of processing HD video images in real time.

Key words: video coding; HEVC; motion estimation; hardware architecture design

1 引 言

运动估计是消除视频时域冗余的重要编码方法, 包括整像素运动估计(Integer Motion Estimation, IME)和分像素运动估计(Fractional Motion Estimation, FME). 官方提供的参考软件 HM 的 IME 搜索算法计算量和复杂度都比较高, 且无规律的数据流不易流水线或并行处理, 因此不适合硬件实现. 在 H. 264 中, 为实现对视频流的实时处理, 前人提出许多用硬件实现的搜索算法^[1-4]. 文献[5]采用全搜索算法, 提出了一种支持可重构搜索区域的层次化数据重用策略的硬件架构, 但其搜索范围最大为 ± 16 . 文献[6]提出了一种并行聚集树搜索算法, 通过合并粗搜索之后的相同运动矢量(Motion Vector, MV)的 PU 再细搜索. 文献[7]提出了一种

CCE(Cost and Coding Efficient)运动估计方法: 为每种 PU 设计各自的计算单元, 串行处理不同种类 PU. 其缺点是相应的电路规模和数据带宽比较大, 因此只支持了 64×64 、 32×32 、 16×16 三种 PU. PU 的划分方式如图 1 所示.

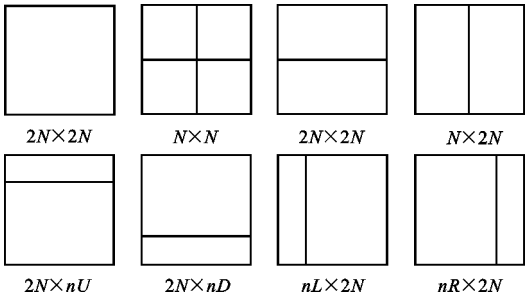


图 1 PU 的划分方式

本文结合前人在 IME 中提出的搜索算法,提出了基于划分深度的先粗搜索再细搜索的搜索算法,通过复用不同划分深度 PU 的计算单元 SAD(Sum of Absolute Difference)来减少硬件资源的消耗.在不同的搜索阶段参考像素数据复用率基本可以达 100%.在±64 的搜索范围内,可以实现 3 840×2 160@60 f/s 的实时处理.

2 整像素运动估计算法

运动估计是在已编码的图像(参考帧)中为当前 PU(当前帧)寻找一个最佳匹配块,该块与当前 PU 相对位置的偏移即为最优 MV. IME 采用率失真代价(RD Cost)来计算匹配代价,其计算公式为

$$RD\ Cost(MV, \lambda_M) = SAD + \lambda_M R(MV) \quad (1)$$

式中,λ_M是拉格朗日乘数因子;SAD 是当前块和参考块的像素绝对值差之和;R(MV)是 MV 的编码比特数.

对 IME 搜索算法,文献[7]先是在±64 范围内以 8 为步长粗搜索,这样可以捕捉无规律和大的运动.对于小的和有规律的运动,在起始点±7 范围内用三步搜索算法搜索,其两次搜索的起始点是相同的.文献[2]提出了一种适用于 H. 264 的 IME 算法,先对 16×16 的宏块以 16:1 下采样成 4×4,再在大范围内只对该 4×4 进行粗搜索,得到最优 MV 和次优 MV,然后对 16×16 的宏块在±3H/±2V 范围内对其分别以最优 MV、次优 MV、预测 MV 为中心点做全搜索,得到不同编码宏块的 41 个最优 MV.本文结合以上两篇文献,提出了一种基于 CU 划分深度的粗搜索加细搜索的 IME 搜索算法. CU 划分深度信息如表 1 所示.

表 1 不同深度下的 PU SIZE

CU depth	PU SIZE
depth=0	64×64, 64×32, 32×64
depth=1	32×32, 32×16, 16×32
depth=2	16×16, 16×8, 8×16
depth=3	8×8, 8×4, 4×8, 4×4

整个运动估计的处理过程分为粗搜索和细搜索两步,图 2 给出了算法流程图,其具体过程如下.

(1)粗搜索:对 2N×2N(N=32,16,8)的 PU 在±64 范围内以步长为 8 全搜索,得到粗搜索最优 MV.

(2)细搜索:

①对当前块和参考块以 2:1 下采样;

②对同一深度的不同 PU 以 2N×2N 的粗搜索最优 MV 为搜索中心点,在±7 范围内全搜索.

③对深度为 3 的 8×8 PU 以深度为 2 的 2N×2N 的粗搜索最优 MV 为搜索中心点,在±7 范围内全搜索;

细搜索时采用全搜索算法是全局最优算法,搜索精度相对较高,而文献[7]中的三步搜索算法是局部最优算法.对大于 8×8 PU 处理时,由于 HM 16.6 对像素以 2:1 下采样,考虑硬件资源的消耗,细搜索时也对像素以 2:1 下采样.

和 HM 16.6 相比,对于 BasketballDrill 视频序列,根据文献[8]在量化参数 QP=22、27、32、37 的情况下,对不同配置,BD-rate 的增加和 BD-PNSR 的下降如表 2 所示.可以发现在 Random Access 下 BD-rate 增加最大为 5.71%,在硬件处理上可以接受的.

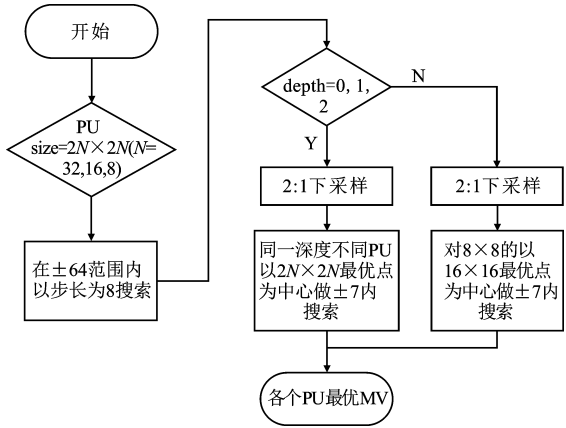


图 2 算法流程图

表 2 本文算法与 HM 算法在编码效率上的差异

配置参数	BD-rate/%	BD-PNSR/dB
Lowdelay	4.46	-0.15
Lowdelay-P	5.05	-0.16
RandomAccess	5.71	-0.18

3 整像素运动估计的硬件架构

3.1 运动估计整体架构设计

整个运动估计的结构框图如图 3 所示,其中 Current RAM 存储 32×32 CU 的像素值,深度为 32,位宽为 256 bit, Cur PU registers 是用来处理当前像素的寄存器. Reference RAM 存储搜索窗口的像素值,由 5 个深度为 192,位宽为 256 bit 的 RAM 组成. SAD 计算模块是一个 32×32 的 PE 阵列. SAD RAM cluster 为 64×64 的 PU 缓存子块的

SAD 值. MVcost 单元计算 $R(MV)$. RDcost 单元计算率失真代价. Comparator 单元比较不同搜索点的率失真代价,得到不同 PU 的最佳的 MV. 图 4 为整个硬件处理过程.

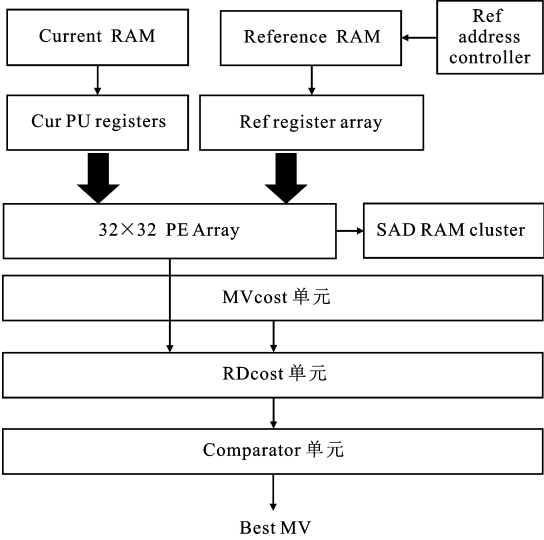


图 3 整体架构设计

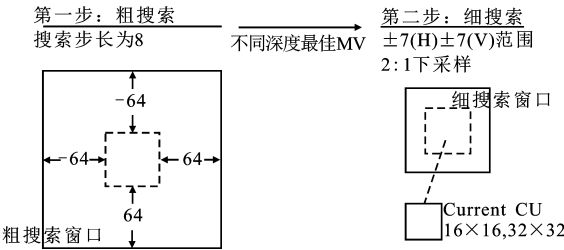


图 4 硬件处理过程

3.2 搜索窗口的存储架构设计

同一个 CTU 内的参考像素之间存在复用情况. 如图 5 所示, 4 个 32×32 CU 的参考像素之间数据复用关系是: CU 2 相对于 CU 0 是向下移动 32 个像素, CU 1 相对于 CU 0 是向右移动 32 个像素. 因此, 搜索窗口存储最终设计成 5 个 192×32 的大小 RAM, 即深度为 192, 位宽为 $32 \times 8 = 256$ bit, 如图 6 所示. 图 5 中 4 个 CU 的处理顺序为 0, 2, 1, 3.

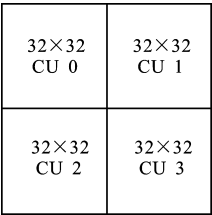


图 5 4 个 CU

在处理 CU 1 时只需更新第一个 RAM 的值, 这样可以降低片上带宽. 在处理整副图像时, 存储第

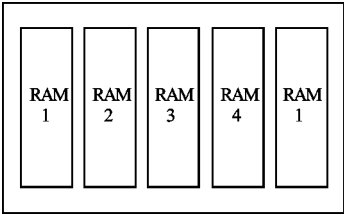


图 6 参考像素 RAM

一个 CTU 的参考像素 RAM 会需比较多的周期, 其他的相对来讲只需更新这 5 个 RAM 中的一个 RAM 的值, 因此参考像素的带宽可以为 256 bit.

3.3 粗搜索架构设计

根据前面的算法描述, 粗搜索是在 ± 64 范围内以步长为 8 进行搜索, 且根据深度信息, 只对深度为 0, 1, 2 根节点处的 PU 进行粗搜索, 即对 64×64 , 32×32 , 16×16 的 PU 进行粗搜索. 因为搜索起始点是相同的, 所以设计中不同 PU 的 SAD 可以合并计算, 即计算一个 32×32 像素块的 SAD 同时可以得到 4 个 16×16 像素块的 SAD.

在以 32×32 像素块为一个 SAD 处理阵列时, 因为步长为 8, 对参考像素来说, 相邻搜索点之间会有很多参考像素是相同的, 如左右两点之间有 24×32 像素块是相同的, 上下两点之间有 32×24 像素块是相同的. 为减少处理的周期数, 需考虑上下左右搜索点参考像素的完全复用, 因此本文设计了一种层次复用的参考像素调度策略, 可以让其参考像素完全复用.

参考像素的调度策略如图 8 所示, 在垂直方向运动矢量 $mv_y = -64$ 时, 取搜索窗口参考像素的前 8 行像素, 可以计算图 7 中第一个 32×8 像素块在 $mv_y = -64$ 时 16 个搜索点的 SAD 值, 但是无法计算图 7 中第 2, 3, 4 的 32×8 像素块在 $mv_y = -64$ 的 SAD 值. 取第二个 8 行参考像素, 可以计算图 7 中第一个 32×8 像素块在 $mv_y = -56$ 的 SAD, 同时可以计算图 7 中第二个 32×8 像素块的 $mv_y = -64$ 的 SAD. 采用这种策略, 每次计算时只需取 8 行参考像素就可完成 SAD 的计算, 并且通过这种调度方式, 在计算时整个参考像素的复用率可以达到 100%.

但这样做的缺点是如果计算 16×16 PU 的 SAD, 则同一搜索点的 SAD 是分层得到的, 所以要缓存上一次的计算结果. 最后采用了 8 个深度为 16 的 FIFO 来缓存之前计算的 16×8 PU 的 SAD 值. 通过 FIFO 的利用, 则可完成第一步粗搜索.

粗搜索的具体设计框架如图 9 所示, 第一个是 8 行参考像素寄存器, 用来缓存 8 行参考像素, 类似

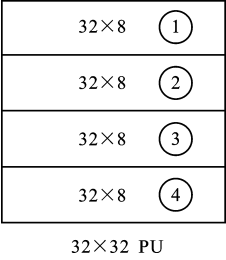


图 7 4 个 32×8 像素块

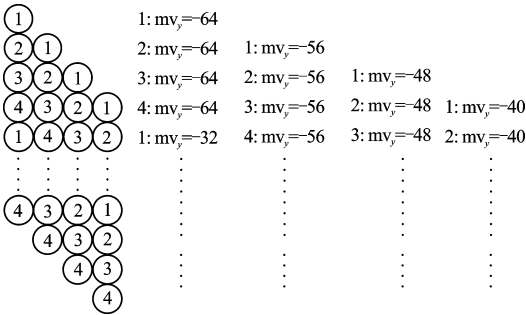


图 8 粗搜索参考像素调度策略

于 line buffer, 每行是 160 个像素, 第二个也是 8 行参考像素寄存器. 具体过程如下:

- (1) 第一个 8 行寄存器缓冲 8 行参考像素.
- (2) 第一个 8 行寄存器缓冲的像素全都转移到第二个 8 行寄存器.
- (3) 每个周期第二个 8 行寄存器向右移 8 个像素, 同时第一个 8 行寄存器缓存下一个 8 行参考像素.

第二个 8 行寄存器向右移 15 次就可计算出同一水平位置 16 个搜索点的 SAD 值. 根据图 8 的参考像素调度策略, 将第二个 8 行参考像素寄存器最右端的 32×8 个像素送入 4 个 SAD 计算单元. 最后, 将 FIFO 处理的数据送入图 3 中最后代价计算模块和比较模块, 则可完成整个的计算过程, 得到粗搜索的最优 MV.

3.4 细搜索处理设计

粗搜索得到了 1 个 32×32 PU 的最优 MV, 4 个 16×16 PU 的最优 MV, 根据前面的算法, 需要以这几个最优 MV 为中心点搜索其±7 范围内的点, 并得到同一深度其他 PU 的最优 MV. 细搜索处理时先对参考像素和当前像素 2:1 下采样. 由于细搜索用全搜索算法, 所以用光栅扫描的方式来处理参考像素, 达到完全复用上下左右搜索点的参考像素的目的. 对参考像素左右方向采用循环右移, 上下方向采用向下移位. 图 10 以 4×4 大小的像素块给出了移位操作过程, (a) 通过循环右移可以得到同一水平方向相邻搜索点的参考像素, 如(b), 处理完同

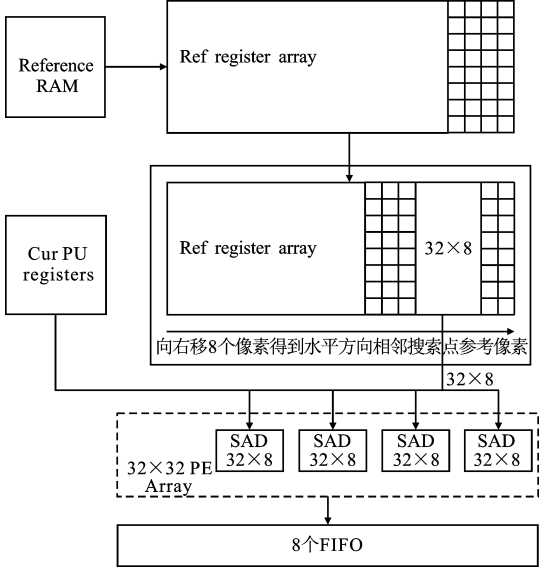


图 9 粗搜索硬件架构

一水平方向搜索点后回到原来位置(c), 然后再向下移位得到垂直方向相邻搜索点参考像素(d).

对 32×32 的 PU 需要 46×46(7+32+7) 参考像素块, 对 16×16 的 PU 需要 30×30(7+16+7) 参考像素块. 所以 32×32 的 PU 需要 46×32 个 8 bit 寄存器, 16×16 的 PU 需要 4 个 30×16 个 8bit 寄存器. 因为有以下采样的处理, 所以 32×32 PU 的参考像素需要 46×16 个 8 bit 寄存器, 16×16 PU 的参考像素需要 30×8 个 8 bit 寄存器, 所需的寄存器数量相对来说比较多. 结合粗搜索的 2 个 8 行寄存器, 考虑到粗搜索和细搜索时参考像素寄存器的复用, 第一个 8 行寄存器没有设计成 line buffer, 这样可以减少寄存器的使用. 在细搜索时寄存器的操作为循环右移和向下移位. 同时由于参考像素 RAM 的设计, 同一行参考像素在一个周期就可得到.

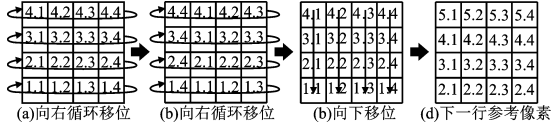


图 10 细搜索时参考像素复用过程

对于深度为 0 的 PU, 用一个深度为 256 的 SAD RAM 来存储 32×32 PU 的粗搜索 SAD 值, 在处理最后一个 32×32 PU 时则可得到 64×64 PU 的 SAD 值, 进一步得到粗搜索的最优 MV. 在细搜索时以 2:1 下采样为 64×32 像素块, 将其分成两个 32×32 像素块来细搜索. 最后通过细搜索则可以得到不同深度不同尺寸 PU 的最优 MV.

4 综合结果

本文提出的整像素运动估计模块在粗搜索时共搜索 256 个点,需 $16 \times 19 = 304$ 个周期,其中 SAD 模块和比较模块是流水线形式的. 第二步细搜索时由于 5 个不同的 MV,所以取参考像素需要花费 $(16 + 8 \times 4) \times 2 = 96$ 个周期, 225 个搜索点需 225 个周期. 考虑 SAD 和比较模块的周期花费,最终一个 32×32 CU 共需 645 个周期. 对于深度为 0 的 PU 的处理共需 524 个周期,所以整个运动估计过程共需 $645 \times 4 + 524 = 3\ 104$ 个周期. 在处理时同时也考虑了不同深度相同 MV 的合并,在细搜索阶段,如果 1 个 32×32 PU 的粗搜索最佳 MV 和 4 个 16×16 PU 的最佳 MV 相同,则可合并计算 SAD 来减少周期数. 在最极端的情况下,不同深度的粗搜索的 MV 都相同,则需 $(304 + 32 + 225 + 20) \times 4 = 2\ 324$ 个周期. 整个模块在 TSMC 90 nm 的工艺库下综合,最高频率为 377 MHz,逻辑门的面积为 645.5 k. 表 3 给出了本文提出的运动估计模块硬件架构与其他论文的比较结果,可以发现本文的搜索范围较大,同时硬件资源(门数)较低.

表 3 结果对比				
比较项	文献[5]	文献[7]	文献[9]	This work
CMOS technology/nm	90	65	65	90
Block Size	4×4- 64×64	16×16 32×32 64×64	4×4- 64×64	8×8- 64×64
Gate Count	860k	1830k	434k	645.5k
RAM/kB	14	208 (ref)	80	337
Search range	±4, ±8, ±16	±64	±27	±64
Modules	IME	IME+FME	IME	IME
Standard	HEVC	HEVC	HEVC	HEVC
Freq/MHz	330	200	720	377
Throughput	3 840×2 160 @30 f/s	3 840×2 160 @30 f/s	1 920×1 080 @30 f/s	3 840×2 160 @60fps

5 结束语

本文提出了一种 HEVC IME 模块的搜索算法和相应的硬件架构. 通过搜索窗口数据复用及不同划分深度 PU 的 SAD 值复用,在 ±64 范围内,支持了从 8×8 到 64×64 PU 的整像素运动估计. 基于上述架构,最终满足了 4 k 视频的实时处理要求.

参考文献:

[1] Sullivan G J, Ohm J R, Han W J, et al. Overview of

the high efficiency video coding (HEVC) standard[J]. IEEE Transactions on circuits and systems for video technology, 2012, 22(12): 1649-1668.

[2] Zhou D, Zhou J, He G, et al. A 1.59 Gpixel/s motion estimation processor with 211 to + 211 search range for UHDTV video encoder[J]. IEEE Journal of Solid-State Circuits, 2014, 49(4): 827-837.

[3] Ndili O, Ogunfunmi T. Algorithm and architecture co-design of hardware-oriented, modified diamond search for fast motion estimation in H. 264/AVC[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2011, 21(9): 1214-1227.

[4] Chen C Y, Chien S Y, Huang Y W, et al. Analysis and architecture design of variable block-size motion estimation for H. 264/AVC[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2006, 53(3): 578-593.

[5] 陈伟伟, 朱惠, 何卫锋, 等. 一种面向 HEVC 运动估计的高性能 VLSI 架构[J]. 微电子学与计算机, 2013, 31(6): 10-14.

[6] Ye X, Ding D, Yu L. A hardware-oriented IME algorithm and its implementation for HEVC[C]// Visual Communications and Image Processing Conference, 2014 IEEE. IEEE, 2014: 205-208.

[7] Sinangil M E, Sze V, Zhou M, et al. Cost and coding efficient motion estimation design considerations for high efficiency video coding (HEVC) standard [J]. IEEE Journal of selected topics in signal processing, 2013, 7(6): 1017-1028.

[8] Bossen F. Common test conditions and software reference configurations [EB/OL]. [2016-12-01]. <http://www.doc88.com/p-041679314712.html>.

[9] Medhat A, Shalaby A, Sayed M S. High-throughput hardware implementation for motion estimation in HEVC encoder[C]// 2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWS-CAS). IEEE, 2015: 1-4.

作者简介:

刘凯丽 女,(1993-),硕士研究生. 研究方向为数字集成电路设计. E-mail:karry@mail.ustc.edu.cn.

陈成 男,(1991-),硕士研究生. 研究方向为数字集成电路设计.

李 垚 男,(1968-),博士,副教授. 研究方向为集成电路建模、集成电路设计.

陈松 男,(1978-),博士,副教授. 研究方向为 VLSI 计算机辅助设计、计算机视觉嵌入式系统和实时多媒体处理系统.