一种 H.264 帧内预测模式判决算法及 VLSI 实现体系

黄 凯, 秦 兴, 严晓浪, 葛海通
(浙江大学超大规模集成电路设计研究所, 浙江杭州 310027)

摘 要: 17 种预测模式和率失真优化模式判决极大的增加了 H.264 帧内编码器硬件设计的复杂度. 目前的模式判决快速算法能大量减少模式判决的复杂度, 但却不易于硬件实现. 本文在 Sobel 边缘检测模式判决算法的基础上, 提出了一种面向 VLSI 实现的模式判决优化算法. 该算法通过修改 16×16 宏块部分像素的 Sobel 边缘检测算子来减少存储器读取次数, 优化预测模式区域的范围来减少硬件设计复杂度, 并采用变换后残差绝对值和(SATD)来简化编码代价判决运算. 实验结果表明, 采用该算法的帧内硬件编码器可以在确保编码质量的前提下, 显著降低硬件实现复杂度和提高编码器效率.

关键词: H.264; 帧内编码; 模式判决; VLSI 实现
中图分类号: TN919.81 文献标识码: A 文章编号: 0372-2112 (2007) 02-0207-05

A H.264 Intra Prediction Mode Decision Algorithm and VLSI Implementation

HUANG Kai, QIN Xing, YAN Xiao-lang, GE Hai-tong
(*Institute of VLSI Design, Zhejiang University, Hangzhou, Zhejiang 310027, China*)

Abstract: The complexity of H.264 hardware intra encoder is remarkable due to 17 prediction modes and Rate-distortion optimization mode decision. Although reduce coding complexity greatly, most of current intra prediction mode decision algorithms are hard to implement in hardware. A new mode decision algorithm for VLSI implementation is proposed based on Sobel edge detection mode decision algorithm. Our algorithm decreases memory bandwidth by modifying Sobel edge detector of part pixels, reduces hardware design complexity by optimizing mode region range, and simplifies cost computations of each prediction mode by using SATD. The results of experiment and hardware design show that our algorithm can greatly reduce the complexity of hardware implementation and significantly improve the efficiency of coder while coding quality is decreased slightly.

Key words: H.264; intra-frame code; mode decision; VLSI implementation

1 引言

H.264/AVC 是一种高性能的视频编码标准, 与 MPEG-4、H.263 和 MPEG-2 相比, H.264/AVC 在码率上相应减少了 39%、49% 和 64%[1,2]. 凭借高效的编码性能, H.264/AVC 已经在高清电视(HDTV)、存储媒体、无线多媒体应用等方面显示出了巨大的应用潜力. 帧内预测是 H.264/AVC 编码标准的一种基本预测方式. 与 JPEG2000 DWT 5/3 编码标准相比, H.264/AVC 帧内编码的 PSNR(Pear Signal-to-Noise Ratio)增加了 0.5-1.0dB[4]. H.264/AVC 帧内预测总共有 17 种预测模式: 9 种 4×4 亮度预测模式、4 种 16×16 亮度预测模式和 4 种 8×8 色度预测模式[3]. 对于每个 16×16 的宏模块(4:2:0 图像), 编码器必须完成 144 次亮度的 4×4 预测模式运算、4 次亮度的 16×16 预测模式运算和 4 次色度的 8×8 预测模式运算, 而每种预测模式背后是大量复杂的率失真优化(RDO, Rate Distortion Optimization)运算. 在实时帧内编码器工作时, 用于预测模式生成和选择的时间占总运算时间的 77%[4]. 因此, 可以通过快速模式判决算法来减少这部分运算量, 从而提高编码器的效率.

2 当前帧内模式判决算法的问题

现有的帧内预测编码模式判决的优化算法主要体现在两个方面: 一个是简化 RDO 算法来减少运算工作量[4,8]; 另一个是通过预处理来减少预测模式的种类[5,6,9]. 简化 RDO 算法一般采用 SATD(Sum of Absolute Transformed Differences)或 SAD(Sum of Absolute Differences)来替换 RDO 中的失真运算. 通过预处理来减少预测模式种类指的是利用局部相关性或其他周围信息来分辨各种预测模式的概率, 从而滤除概率小的预测模式. 基于空间和转换域特征(Joint Spatial and Transform Domain)的快速 H.264 帧内预测模式判决[5]和 Sobel 边缘检测(Edge Detection)快速帧内模式判决[6]是两种典型的部分预测模式搜索算法. Sobel 边缘检测模式判决算法的优化效果非常好(在减少 65% 的运算量时, PSNR 和比特流几乎没有变化)[10]. 下面将详细介绍该算法.

$$\nabla f \approx |G_x| + |G_y| \tag{1}$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2}$$

$$G_x = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3) \tag{3}$$

$$G_y = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7) \tag{4}$$

$$Amp(\boldsymbol{D}_{i,j}) = |Gx_{i,j}| + |Gy_{i,j}| \tag{5}$$

$$Ang(\boldsymbol{D}_{i,j}) = \frac{180^\circ}{\pi} \times \arctan\left(\frac{Gy_{i,j}}{Gx_{i,j}}\right), \quad |Ang(\boldsymbol{D}_{i,j})| < 90^\circ \tag{6}$$

$$Histo(k) = \sum_{(m,n)\in SET(k)} Amp(\boldsymbol{D}_{m,n}) \tag{7}$$

收稿日期: 2006-02-07; 修回日期: 2006-05-10

$$SET(k)\in\{\{(i_0,j_0)\},\{(i_1,j_1)\},\cdots,\{(i_u,j_u)\},\cdots,\{(i_8,j_8)\}\mid Ang(\boldsymbol{D}_{i_u,j_u})\in a_u\} \quad (8)$$

$$G_x=Z_6-Z_5 \quad (9)$$

$$G_y=Z_8-Z_5 \quad (10)$$

$$G_x=Z_5-Z_4 \quad (11)$$

$$G_y=Z_8-Z_5 \quad (12)$$

$$G_x=Z_5-Z_4 \quad (13)$$

$$G_y=Z_5-Z_2 \quad (14)$$

$$G_x=Z_6-Z_5 \quad (15)$$

$$G_y=Z_5-Z_2 \quad (16)$$

边缘检测用于寻找图像中能量强度变化快的地方, 公式(1)和(2)是边缘检测基本公式[7]. ∇f 是当前像素的边缘矢量强度值, 而 $\alpha(x,y)$ 是其边缘矢量方向. G_x 和 G_y 分别是 $f(x,y)$ 对 x 和 y 的一阶导数. Sobel 边缘检测算子(公式(3)和(4))被用来近似像素图 1(a)中 Z_5 的 G_x 和 G_y, 这就是基于 Sobel 算子的边缘检测算法[7].

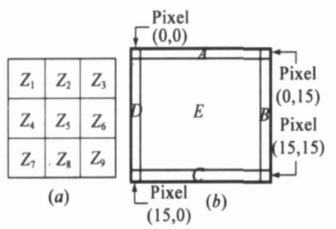

图 1 算法中像素位置示意图

所有帧内预测模式(除 DC 预测模式)具有很强的方向性. 因此文献[6]提出可以将整个 360 度空间分为 8 个区域, 每个区域对应 4 种预测模式. 如公式(5)和(6)所示, 通过基于 Sobel 算子的边缘检测算法得到 4×4 模块中每个像素边缘矢量方向 $Ang(D_{i,j})$ 及其强度值 $Amp(D_{i,j})$. 又如公式(7)和(8)所示, 根据 $Ang(D_{i,j})$ 可以确定该角度处于哪个区域 a_u, 将同一个区域的像素编为一个组($SET(k)$). 然后, 相同组的边缘矢量强度值 $Amp(D_{i,j})$ 求和得到 $Histo(k)$. 最后比较各个组的 $Histo(k)$, 确定哪个区域是最优的模式选择区域. 根据该区域可以得到最优可能较大的 4 种预测模式, 从而滤除其他预测模式.

文献[6]中的算法的优点在于大量减少预测模式的种类, 从而减少模式预测时间, 提高编码器的性能. 但 Sobel 边界检测模式判决算法有三点不利于硬件实现.

(1)增加了约40%的像素读取次数. 根据前面对 Sobel 边界检测模式判决算法的介绍可知: 在处理一个 16×16 的宏块时, 除了读取其本身的 256 个像素外还要获取其周围的 68 个像素, 这将带来额外的存储器读取. 假设数据总线是 32 位, 而每个像素 8 位, 如果不考虑总线和存储器的响应时间(laterncy), 那么需要 64 个总线周期取到 16×16 宏块 256 个像素数据. 但由于宏块左边和右边的 36 个周围像素地址不连续, 所以只能单个读取. 因此需要 44 个时钟周期去取得额外 68 个周围像素.

(2)$Ang(\boldsymbol{D}_{i,j})$ 是 arctan 函数运算的结果, 不易于硬件实现.

(3)由于采用 RDO 来计算每种预测模式的编码代价(cost), 所以导致硬件设计复杂度过高. 因此, Sobel 边缘检测模式判决算法不适合用于硬件实现.

3 基于 VLSI 实现的模式判决算法

3.1 算法介绍

针对上面所提到的三个硬件实现问题, 本文从四个方面对 Sobel 边缘检测模式判决算法进行了相应的改进.

(1)所有 16×16 模块的边缘像素不再采用公式(3)和(4)计算 G_x 和 G_y, 而是如图 1 所示, 在 A, B, C, D 四个不同区域分别采用公式(9)和(10)、(11)和(12)、(13)和(14)、(15)和(16)代替原公式. 这样可以避免读取额外的 68 个相邻像素. 因为区域 E(15×15=225 个像素)在 16×16 宏块的内部, 所以仍采用原公式.

(2)采用正切函数近似值来避免用硬件实现 arctan 函数. 由于 $arctan\left(\frac{G_y}{G_x}\right)$ 函数在(-90°, 90°)之间是随 $\frac{G_y}{G_x}$ 的递增而递增的, 所以对于一定的 $\frac{G_y}{G_x}$ 值, 可以明确知道它处于哪个角度范围. 根据这个角度范围, 就可以知道该像素的属于哪个模式选择区域. 这样, 就将比较角度的工作转为比较正切函数值. 如图 2 所示, 对于 4×4 预测模式(I4MB), 整个 360° 角度空间被分成 8 个区域. 根据图中各个区域(region)的角度, 就可以得到其对应的正切函数值. 但这些值都是较为复杂的小数, 比如 13.3° 的正切函数值约是 0.23639, 在硬件上很不容易实现. 在实验中发现当该数值取 0.25 近似时, 编码的结果几乎不会受任何影响. 因此通过实验, 就可以得到了各个角度相对应的正切函数近似值, 从而各个模式选择区域的范围也就确定了.

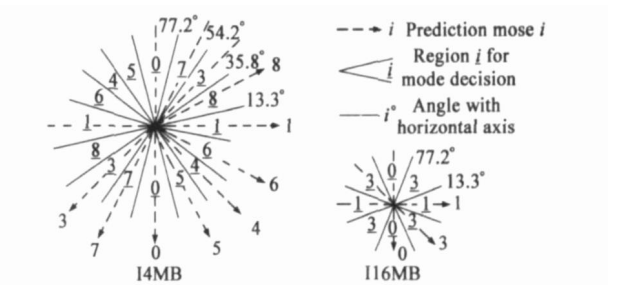

图 2 I4MB 和 I16MB 的预测模式区域

表 1 I4MB 模式区域范围和预测模式的概率排序

Region	Range	Possibility(high to low)
0	(-∞, -4] or (4, +∞)	0 5 7 3 4 1 6 8
1	(-0.25, 0.25]	1 6 8 3 4 0 5 7
3	(0.7, 1.4]	3 7 8 0 1 4 5 6
4	(-1.4, -0.7]	4 5 6 0 1 3 7 8
5	(-4, -1.4]	0 4 5 6 7 1 3 8
6	(-0.7, -0.25]	6 1 4 5 8 0 3 7
7	(1.4, 4]	7 0 3 5 8 1 4 8
8	(0.25, 0.7]	8 1 3 6 7 0 4 5

表 2 I16MB 模式区域范围和预测模式的概率排序

Region	Range	Possibility(high to low)
0	(-∞, -4] or(4, +∞)	0 1 3
1	(-0.25, 0.25]	1 0 3
3	(-4, -0.25] or (0.25, 4]	3 0 1

对于某个模式预测区域, 根据各种预测模式角度离该区

域中心角度的距离可以判断其最优可能性(possibility)大小. 如图 2 所示, 对于区域 1 来说, 预测模式 1 刚好在其区域中心, 所以最优可能性最大. 其次, 预测模式 6 和 8 离区域中心最近, 所以这两种模式的最优可能性应该比除模式 1 外其它预测模式更大. 由统计可知, 一般模式号越小则最优可能性越大[6], 所以模式 6 比模式 8 的最优可能性大. 按照这个规则, 可以得到表 1 中各个区域的预测模式概率顺序. 因为模式 2 (DC 预测)是没有方向性的, 所以无法将其排入预测模式概率顺序中去. 但是考虑到图像的边界宏块或块, 以及各区域初选边缘矢量强度 $Amp(D_{i,j})$ 相等的特殊情况, 本文算法将模式 2 作为一种必需的候选模式.

(3)采用新的 16× 16 预测模式(I16MB)区域范围值来加快模式判决和简化硬件实现. 如图 2 所示, 与文献[6]算法不同, 本文算法将 I16MB 的区域 0 和区域 1 的范围相对减小(与 I4MB 的区域 0 和区域 1 的范围值相同), 而区域 3 的范围相对增大(与 I4MB 区域 3、4、5、6、7、8 总范围相同). 通过上述范围值的修改, 在完成一个 16× 16 模块的 I4MB 模式预测模式区域计算的同时, 也可以得到 I16MB 预测模式区域. I16MB 预测模式概率顺序如表 2 所示. 在硬件实现上, 本文算法可以不用关心 I16MB 的范围比较, 只需有三个寄存器(对应 I16MB 三个预测区域)累加各像素的边缘矢量强度. 在完成 256 个像素的边缘检测算法后, 就可以同时得到 16 个 I4MB 区域和 1 个 I16MB 预测区域.

在本文算法中, 只有最优可能性较大的 4 种 4× 4 预测模式和 2 种 16× 16 预测模式才会被用于帧内模式选择, 这样比全搜索减少 50% 以上的预测模式判决工作量.

(4)本文算法采用了与文献[4]相同的编码代价算法: 失真用基于 DCT 变换的 SATD 表示, 而比特流则用预测模式的比特数代替. 文献[4]的实验证明: 与 RDO 算法相比, 图像质量下降不会超过 0.3dB.

3.2　实验结果和分析

实验工作是基于 JM9.8 参考软件基础上完成的. 本文选择 Mobile、Foreman、Mother 三种不同复杂度的 CIF 格式测试序列. 编码参数为: 采用 CAVLC 熵编码; 没有采用 RDO; 编码帧数为 30; 编码序列是全 I 帧; 量化参数分别是: 22, 28, 32, 38; 通过修改 JM9.8 参考软件分别实现文献[6]中算法和本文算法, 可以得到两个算法相应的编码结果.

3.2.1　不同候选预测模式个数的结果比较

由于文献[6]的算法与本文算法有较多不同, 因此文献[6]中的实验结果并不能直接用于本文算法. 通过实验, 可以得到三种测试序列的 4× 4 预测模式个数与编码性能关系图. 从图 3 中可以看出: 在相同比特流时与全搜索算法相比, 模式个数为 2 的编码图像质量下降了 0.5dB 到 0.8dB, 而模式个数为 4 和模式个数为 6 的编码图像质量与全搜索的结果非常近似. 模式个数为 4 的算法可以比模式个数为 6 的算法减少了三分之一的工作量, 所以 4× 4 预测模式个数为 4 才是最佳选择. 它在显著减少预测模式个数的同时, 几乎不会影响到编码的性能.

3.2.2　与其它模式判决算法比较

为准确比较 JM9.8、文献[6]的算法、本文算法的编码性能, 对于 JM9.8 参考软件(全搜索)和文献[6]的算法(sobel 边缘检测), 本文分别采用了高复杂度的 RDO 和低复杂度的 SATD 来计算编码代价. 表 3 和表 5 是基于 RDO 的实验结果, 而表 4 和表 6 是基于 SATD 的实验结果.

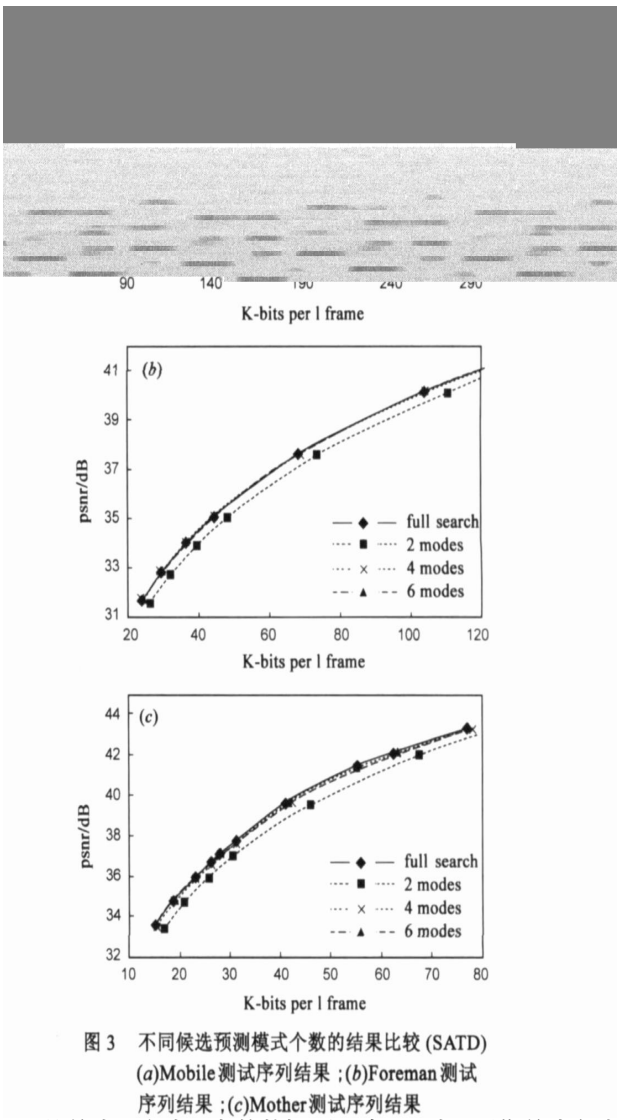

图 3　不同候选预测模式个数的结果比较(SATD)
(a)Mobile 测试序列结果; (b)Foreman 测试序列结果; (c)Mother 测试序列结果

比较表 3 和表 5 中的数据可以发现: 对于图像较为复杂的 Mobile 测试序列, 本文算法与 JM9.8 参考软件 *PSNR* 相差最大(0.2dB~ 0.57dB), 而另外两个图像较为简单的 Foreman 和 Mother 测试序列的 *PSNR* 差值相对较小(0.02dB~ 0.23dB). 从表中还可以发现: *QP* 值越大时, *PSNR* 值相差就越大, 比特流相差也越大.

再比较表 4 和表 6 中的数据可以发现: 不论是对图像复杂度较高的 Foreman 测试序列还是复杂度较低的 Foreman 和 Mother 测试序列, JM9.8 参考软件、本文算法的 *PSNR* 值相差不大(不超过 0.05dB). 而文献[6]的算法与本文算法编码后的 *PSNR* 值几乎相同(最大差值不超过 0.01dB). 文献[6]中算法得到的每帧比特流值最多比 JM9.8 参考软件的每帧比特流值高了 0.4~ 1.9%, 而本文算法得到的每帧比特流值和文献[6]算法结果几乎没有差别.

从上面的分析可以看出: 在编码性能上, 本文算法与基于 SATD 的 JM9.8 和文献[6] 相差不多, 但比基于 RDO 的 JM9.8 和文献[6] 下降了相对较多. 主要原因是采用 SATD 取代 RDO 才导致的性能下降. 但是由于 SATD 的计算量仅是 RDO 的 7%[8], 这可以极大的减少硬件设计复杂度. 实验说明: 本文算法在提高编码器效率的同时, 只会轻微降低图像编码质量.

表 3 JM9.8(RDO)、文献[6]算法(RDO)、本文算法(SATD)的 *PSNR* 值比较(dB)

QP	Foreman			Mobile			Mother		
	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法
22	41.72	41.66	41.53	41.13	40.95	40.60	43.49	43.43	43.26
28	37.66	37.63	37.58	35.67	35.54	35.26	39.65	39.62	39.60
32	35.18	35.15	35.11	32.07	31.95	31.73	37.10	37.07	37.06
38	31.70	31.68	31.68	27.17	27.10	26.97	33.60	33.57	33.57

表 4 JM9.8(SATD)、文献[6]算法(SATD)、本文算法(SATD)的 *PSNR* 值比较(dB)

QP	Foreman			Mobile			Mother		
	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法
22	41.56	41.53	41.53	40.61	40.60	40.60	43.29	43.26	43.26
28	37.60	37.58	37.58	35.28	35.26	35.26	39.62	39.60	39.60
32	35.12	35.12	35.11	31.74	31.73	31.73	37.11	37.07	37.06
38	31.69	31.68	31.68	27.00	26.98	26.97	33.62	33.57	33.57

表 5 JM9.8(RDO)、文献[6]算法(RDO)、本文算法(SATD)的比特流值比较(kbit/f)

QP	Foreman			Mobile			Mother		
	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法
22	129.01	129.37	130.71	333.3	335.76	339.22	75.89	77.03	77.98
28	66.90	67.63	68.88	219.42	221.70	224.33	40.83	41.8	42.76
32	43.49	44.12	45.06	159.11	161.13	163.12	26.96	27.67	28.46
38	23.49	23.93	24.65	90.03	91.87	93.37	14.26	14.74	15.39

表 6 JM9.8(SATD)、文献[6]算法(SATD)、本文算法(SATD)的比特流值比较(kbit/f)

QP	Foreman			Mobile			Mother		
	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法	JM9.8	[6]中算法	本文算法
22	127.3	129.8	130.71	337.29	338.93	339.22	76.81	77.81	77.98
28	68.05	68.7	68.88	222.23	223.75	224.33	41.77	42.70	42.76
32	44.35	44.95	45.06	161.01	162.93	163.12	27.75	28.41	28.46
38	24.17	24.63	24.65	91.55	93.25	93.37	14.99	15.37	15.39

4 VLSI 硬件实现

本文算法的 ASIC 电路可以并行处理 4 个像素的 sobel 算子、边缘检测处理和模式区域处理工作. 如图 4 所示, 电路由 4 像素 G_x/G_y 产生器(4-pixel G_x/G_y)、4×4 缓存(4×4 pipe)、边缘检测处理(EDP)、模式区域边缘检测强度处理和比较、计数器和其他控制逻辑组成. 70 个时钟周期可以得到一个 16×16 宏块的 I4MB 和 I16MB 的最佳预测模式区域.

4.1 4 像素 Gx/Gy 产生器

由公式(3)和(4)可以知道, 每个像素的 sobel 算子 G_x 和 G_y 由周围的 8 个像素决定. 当计算 $Z5$ 的 G_x 时, 必须用到 $Z1$、$Z4$、$Z7$、$Z3$、$Z6$、$Z9$ 六个像素的值, 需要 3 次减法和 2 次加法. 如果每个像素都单独计算 G_x 和 G_y, 那么完成 SDTV 视频编码(720×480, 30fps)的 sobel 算子计算需要约 62208000 次减法(3×2×720×480×30)和 41472000 次加法, 这是非常大的运算量. 但是因为相邻像素的 sobel 算子存在相同运算, 所以可以采用并行来减少运算量. 如图 5 所示, 4 个像素的 G_x 计算只需要 6 次减法和 4 次加法, 这样运算量可以减少一半. 只需 10 个加法器就可以完成 4 个像素的 sobel 算子并行计算.

4.2 边缘检测处理(EDP)

边缘检测处理模块主要根据像素的 Sobel 算子 G_x 和 G_y 来确定边缘检测矢量(***AMP***)和角度的范围(Mode Region). 如图 6 所示, 通过 G_x 和 G_y 的绝对值可以求得表 1 中模式范围边界值所需的 6 种关系比较. 比如判断 G_y; G_x 是否大于 1.4, 只需比较 $|G_y|$ 的 5 倍和 $|G_x|$ 的 7 倍大小即可. 图 6 中的 *sel*0、*sel*1、*sel*2、*sel*3、*sel*4 和 G_x、G_y 的符号值可以确定出该像素边缘检查矢量角度(***Ang***)所在的模式区域.

4.3 4×4 缓存

在实现 4 个像素并行计算 sobel 算子时, 存在一个问题: 无法并行计算同一列的 G_x 和 G_y. 为取得高效的并行性来加快处理速度, 只有采用 4×4 缓存来存储 16 个像素的 G_x 值.

在完成 4×4 块的 4 列像素的 G_x 后, 4×4 块的第一行 G_y 将与缓存中的第一行像素的 G_x 作为 *EDP* 模块的输入, 完成边缘检测处理.

4.4 VLSI 设计电路结果和分析

本文采用 SMIC 的 0.18 工艺完成了电路的物理实现. 在最坏的条件下, 该电路可以工作在 150MH. 电路各个模块的门数如表 7 所示. 从表中可以看出 4×4 缓存(4×4 Pipe) 和模式区域处理(MRP) 两模块共占总门数的一半以上, 而 G_x 和 EDP 模块都相对较小, 这也就说明并行处理导致的面积增加是有限的. 4 个像素并行处理既利于 I4MB 模式处理, 也不会明显增加电路面积.

表 7　各模块门数统计和总面积

Module	Gate Count		
	Comb	Noncomb	Total
4p G_x& G_y	902	0	902
EDP(×4)	192(×4)	0	192(×4)
4×4 Pipe	770	1514	2284
MRP	1815	1388	3203
Comp	1130	0	1130
Others	887	458	1345
Total	6272	3360	9632
Total area	0.32mm×0.32mm		

图 4 中的电路可以作为预处理单元来用于一般的 H.264 帧内编码器. 比如, 它可以用于文献[4] 的硬件视频编码器(总门数是 84985). 由于减少了一半以上的预测模式, 因此该编码器的帧内预测速度可以从原来的每个 16×16 宏块 1300 时钟周期减少到 716 时钟周期. 当芯片工作在 55M 时, 可以完成每秒 21 帧的 720P HDTV(1280×720) 编码和每秒 56 帧的 SDTV (720×480) 编码.

5　结论

本文通过对现有帧内编码预测模式判决算法的分析, 在基于 Sobel 边缘检测模式判决算法[6]的基础上, 提出了面向硬件实现的优化算法. 该算法立足于 VLSI 实现, 不仅减少存储器带宽而且简化了编码器硬件实现. 实验证明, 本文的算法在对硬件设计优化和提高编码速度的同时, 还确保了编码性能和编码器的面积.

参考文献:

[1] Thomas Wiegand, et al. Overview of the H.264/AVC video coding standard[J]. IEEE Transactions on circuits and systems for video technology, 2003, 13(7): 657–673.

[2] Anthony Joch, et al. Performance comparison of video coding standards using lagragian coder control[A]. Proceedings of the 2002 International Conference on Image Processing[C]. New York, USA: ICIP, 2002. 501–504.

[3] Iain E G Richardson. H.264 and MPEG-4 Video Compression Video Coding for Next-Generation Multimedia[M]. New York: John Wiley & Sons, 2003.

[4] Yu-Wen Huang, et al. Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder[J]. IEEE Transactions on circuits and systems for video Technology, 2005, 15(3): 378–401.

[5] Kim Changsung, et al. Fast H.264 intra prediction mode selection using joint spatial and transform domain features[J]. Journal of Visual Communication and Image Representation, 2005, 17(2): 291–310.

[6] Feng Pan, et al. Fast mode decision for intra prediction[A]. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.3)[C]. 7th meeting: Pattaya II, Thailand: JVT-G013, March 2003.

[7] Rafael C Gonzalez, Richard E Woods, Steven L Eddins. Digital Image Processing Using MATLAB[M]. New Jersey: Prentice Hall, 2002.

[8] Hyungjoon Kim, Yucel Altunbasak. Low-complexity macro block mode selection for H.264/avc encoders[A]. Proceedings of the 2004 International Conference on Image Processing [C]. Singapore: ICIP, 2004. 765–768.

[9] Jeyun Lee, Byeungwoo Jeon. Fast mode decision for H.264 [A]. Proceedings of the 2004 IEEE International Conference on Multimedia and Expo[C]. Taipei, Taiwan: ICME, 2004. 1131–1134.

[10] Zhiping Lin. Verification of results for fast mod decision for intra prediction[A]. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q, 6)[C]. 7th Meeting: Pattaya II, Thailand: JVT-G026, March 2003.

作者简介:

黄　凯　男, 1980 年 11 月出生于江西上饶, 现为浙江大学超大规模集成电路设计研究所博士研究生, 研究方向: 视频编解码和集成电路设计. E-mail: huangk@vlsi.zju.edu.cn

秦　兴　男, 现为浙江大学超大规模集成电路设计研究所博士研究生, 研究方向: 多媒体处理算法及集成电路设计.

严晓浪　男, 浙江大学电气工程学院教授, 博士生导师, 研究方向: 集成电路设计和布图技术.