高技术通讯 2012 年 第22 卷 第1 期:106 ~ 111

doi:10.3772/j.issn.1002-0470.2012.01.017

基于叉形编码路径的 JPEG2000 位平面编码器 VLSI 结构的设计与验证①

刘文松②* 朱 恩* 王 健③** 孙 磊* 林 叶*

(*东南大学射频与光电集成电路研究所 南京 210096)

(**中国科学院自动化研究所 北京 100190)

摘 要 研究了 JPEG2000 中位平面编码算法,提出了适用于显著性传播过程和清除过程的叉形编码路径,使得完成一个 $4 \times n$ 条带编码的路径长度缩减为标准的 $(n+1)/(2n)$。基于该编码路径,设计了两窗口流水线硬件编码结构,该结构通过两个编码窗口流水线滑动在平均每个时钟内完成 16 个样本点的位平面编码,关键模块采用组合电路实现,避免了时钟消耗和复杂控制,可在位平面间并行进行。给出了系统的整体 VLSI 架构。FPGA 验证结果表明,系统时钟可综合到203.083MHz,处理 512 × 512 的灰度图像达 276fps,可满足图像实时处理的要求。

关键词 JPEG2000,位平面编码器,叉形编码路径,两窗口流水线

0 引 言

于2000 年提出的 JPEG2000 静态图像压缩标准[1]因采用了离散小波变换(DWT)和最优截断嵌入式块编码(EBCOT)等技术处理图像,相对于 JPEG 而言能在高压缩比情况下依然保持较好图像效果,且同时支持有损压缩和无损压缩、多分辨率渐进式传输和感兴趣区域(ROI)编码等功能,在高端数字图像应用中得到了大力推广。在某些特殊领域,需要捕捉识别高速运动目标,将处理后的图像快速编码,通过无线信道传递到后方指挥中心,形成流畅视频,这就对 JPEG2000 编码速率提出了较高的要求。由于 JPEG2000 中位平面编码器(bit plane coder, BPC)以单比特为单位对所有数据进行操作,因而会影响整个编码系统的性能,如何从算法和电路结构两方面进行优化,以满足图像实时处理的要求,是目前 JPEG2000 研究领域的热点和难点。

JPEG2000 标准中采用的 BPC 算法由 Taubman[2,3]提出。Andra[4]根据标准算法描述的串行编码顺序采用状态机方法实现,没有显著提高编码性能。文献[5]提出了编码过程并行(pass parallel, PP)算法。PP 算法同时执行3 个编码过程对一列样本点操作,同时输出码值和上下文,但在电路实现时显著性状态预测及控制机制过于复杂。文献[6]提出了样本点跳过和列跳过(sample points & columns skipping, SPCS)算法,此算法跳过无上下文输出的样本点以及列,减少了计算次数。但优化程度依赖于图像数据,性能起伏较大。文献[7] 提出了位平面与编码过程并行(bit plane & pass parallel, BPPP)的优化算法,此算法事实上综合运用了位平面并行、编码过程并行和样本点及列跳过 3 种加速方案。由于同时实现了编码过程并行以及样本点跳过和列跳过,其显著性预测及控制操作复杂度大大增加,而各算法单独的缺陷并没有改善,使得整体性能受到约束。但由于实现了编码的位平面间并行,BPPP 算法仍是目前已发表方法中理论编码时间最短的方法。研究基于对 BPC 标准算法结构和内部数据关系的分析,提出了一种可有效降低编码路径长度和易于组合电路实现的叉形编码路径,并利用这种编码路径设计了一种可在位平面间并行编码的两窗口流水线编码硬件结构。本文给出了该结构的设计方法及其性能的实验验证结果。

① 863 计划(2009AA11Z219)资助项目。

② 男,1983 年生,博士生;研究方向:图像编码与集成电路设计;E-mail:xss4@163.com

③ 通讯作者,E-mail: jian.wang@ia.ac.cn

(收稿日期:2010-09-02)

1 位平面编码标准算法

JPEG2000位平面编码标准算法的流程是这样的:首先将量化后的 P 位小波系数组织成为 $N \times N$ 大小的代码块;按比特位顺序将其分割成 P 层 $N \times N$ 的位平面,位平面中的每个比特称为一个样本点;在单个位平面内,样本点每4行划分为一个条带。编码时,编码器自最高位平面向下,逐个位平面逐个条带进行编码。在单个条带内,按照图1所示的串行编码路径,对每个样本点顺序执行3个编码过程,即显著性传播过程(significance propagation pass, SPP)、幅值细化过程(magnitude refinement pass, MRP)和清除过程(cleanup pass, CP)。通过分析样本点及其临域的显著性状态,使用"零编码(zero coding, ZC)"、"符号编码(sign coding, SC)"、"幅值细化编码(magnitude refinement coding, MRC)"和"游程编码(run length coding, RLC)"4种编码原语进行判决并形成上下文。初始时,所有位平面的所有样本点均不显著。

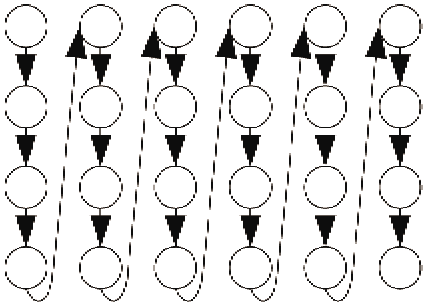

图1 样本点串行编码路径

单条带内样本点编码算法如下:

(1)判断当前样本点是否属于显著性传播过程。若样本点当前不显著,而临域的8个样本点中至少有一个是显著的,则其属于该过程,进行零编码并计算上下文。进而判断其本身是否显著。若是则更新显著性状态,进行符号编码;否则不进行符号编码。如果样本点临域皆不显著或样本点本身已经显著,则其不属于该过程,跳过该点去判断下一个点。对条带内所有样本点执行该操作,然后回到起始点执行(2)。

(2)判断当前样本点是否属于幅值细化过程。若样本点当前显著且未编码,则属于该过程,进行幅值细化编码;否则不属于该过程,跳过该点去判断下一个点。对条带内所有样本点执行该操作,然后回到起始点执行(3)。

(3)执行清除过程。使用游程编码、零编码和符号编码对不属于显著性传播过程和幅值细化过程的所有剩余样本点进行编码。

2 基于叉形编码路径的两窗口流水线结构

2.1 叉形编码路径

标准算法中,在执行显著性传播过程和清除过程时,要判断样本点及其临域的显著性状态,以确定其是否属于当前编码过程,从而判定是否需要更新显著性状态并计算上下文。

设 $v^l_{(m,n)}$ 为第 l 条带中第 m 行 n 列的样本点幅值,$s^l_{(m,n)}$ 为其显著性状态,$s^{*l}_{(m,n)}$ 为其更新后的显著性状态,$c^l_{(m,n)}$ 为临域显著性贡献,$cx^l_{(m,n)}$ 为上下文,$d^l_{(m,n)}$ 为码值。

同一列中4个样本点的临域显著性贡献可用下式表述:

$$\begin{cases} c^l_{(0,n)} = s^{*l-1}_{(3,n-1)} + s^{*l-1}_{(3,n)} + s^{*l-1}_{(3,n+1)} + s^{*l}_{(0,n-1)} \\ \qquad + s^{l}_{(1,n-1)} + s^{*l}_{(1,n-1)} + s^{l}_{(1,n)} + s^{l}_{(1,n+1)} \\ c^l_{(1,n)} = s^{*l}_{(0,n-1)} + s^{*l}_{(0,n)} + s^{l}_{(2,n-1)} + s^{*l}_{(1,n-1)} \\ \qquad + s^{l}_{(1,n-1)} + s^{*l}_{(2,n-1)} + s^{l}_{(2,n)} + s^{l}_{(2,n+1)} \\ c^l_{(2,n)} = s^{*l}_{(1,n-1)} + s^{*l}_{(1,n)} + s^{l}_{(1,n+1)} + s^{*l}_{(2,n-1)} \\ \qquad + s^{l}_{(2,n-1)} + s^{*l}_{(3,n-1)} + s^{l}_{(3,n)} + s^{l}_{(3,n-1)} \\ c^l_{(2,n)} = s^{*l}_{(2,n-1)} + s^{*l}_{(2,n)} + s^{l}_{(2,n+1)} + s^{*l}_{(3,n-1)} \\ \qquad + s^{l}_{(3,n+1)} + s^{l-1}_{(0,n-1)} + s^{l+1}_{(0,n)} + s^{l+1}_{(0,n+1)} \end{cases} \tag{1}$$

式中,当前条带 l 以外的样本点显著性状态取值按照JPEG2000标准中描述的垂直因果模式处理。"+"代表异或运算。

样本点的显著性状态更新公式为

$$s^{*l}_{(m,n)} = s^l_{(m,n)} + c^l_{(m,n)} \cdot v^l_{(m,n)} \tag{2}$$

式中,"·"代表与运算。

根据JPEG2000标准的位平面方法,应顺序地判决第 n 列的 $c^l_{(0,n)}$、$c^l_{(1,n)}$、$c^l_{(2,n)}$ 和 $c^l_{(3,n)}$ 后,才可以判决第 $n+1$ 列的 $c^l_{(0,n+1)}$、$c^l_{(1,n+1)}$、$c^l_{(2,n+1)}$ 和 $c^l_{(3,n+1)}$。但在完成 $c^l_{(0,n)}$ 与 $c^l_{(1,n)}$ 的判决后,$s^l_{(0,n)}$ 与 $s^l_{(1,n)}$ 已通过式(2)更新为 $s^{*l}_{(0,n)}$ 与 $s^{*l}_{(1,n)}$,即 $c^l_{(0,n+1)}$ 的判决条件已满足,可与 $c^l_{(2,n)}$ 的判决并行执行。同理,$c^l_{(3,n)}$ 和 $c^l_{(1,n+1)}$ 也可以并行判决。

通过分析发现,零编码与符号编码的上下文计

算也存在类似规律。零编码上下文计算可用式

$$\begin{cases} cx^{l}_{zc(0,n)} = f_{zc}(s^{*l-1}_{(3,n-1)}, s^{*l-1}_{(3,n)}, s^{*l-1}_{(3,n+1)}, s^{*l}_{(0,n-1)}, \\ \qquad s^{l}_{(0,n+1)}, s^{*l}_{(1,n-1)}, s^{l}_{(1,n)}, s^{l}_{(1,n+1)}) \\ cx^{l}_{zc(1,n)} = f_{zc}(s^{*l}_{(0,n-1)}, s^{*l}_{(0,n)}, s^{l}_{(0,n+1)}, s^{*l}_{(1,n-1)}, \\ \qquad s^{l}_{(1,n+1)}, s^{*l}_{(2,n-1)}, s^{l}_{(2,n)}, s^{l}_{(2,n+1)}) \\ cx^{l}_{zc(2,n)} = f_{zc}(s^{*l}_{(1,n-1)}, s^{*l}_{(1,n)}, s^{l}_{(1,n+1)}, s^{*l}_{(2,n-1)}, \\ \qquad s^{l}_{(2,n+1)}, s^{*l}_{(3,n-1)}, s^{l}_{(3,n)}, s^{l}_{(3,n+1)}) \\ cx^{l}_{zc(3,n)} = f_{zc}(s^{*l}_{(2,n-1)}, s^{*l}_{(2,n)}, s^{l}_{(2,n+1)}, s^{*l}_{(3,n-1)}, \\ \qquad s^{l}_{(3,n+1)}, s^{l+1}_{(0,n-1)}, s^{l+1}_{(0,n)}, s^{l+1}_{(0,n+1)}) \end{cases} \tag{3}$$

表示，符号编码上下文计算可用式

$$\begin{cases} cx^{l}_{sc(0,n)} = f_{sc}(s^{*l-1}_{(3,n)}, s^{*l}_{(0,n-1)}, s^{l}_{(0,n+1)}, s^{l}_{(1,n)}) \\ cx^{l}_{sc(1,n)} = f_{sc}(s^{*l}_{(0,n)}, s^{*l}_{(1,n-1)}, s^{l}_{(1,n+1)}, s^{l}_{(2,n)}) \\ cx^{l}_{sc(2,n)} = f_{sc}(s^{*l}_{(1,n)}, s^{*l}_{(2,n-1)}, s^{l}_{(2,n+1)}, s^{l}_{(3,n)}) \\ cx^{l}_{sc(3,n)} = f_{sc}(s^{*l}_{(2,n)}, s^{*l}_{(3,n-1)}, s^{l}_{(3,n+1)}, s^{l+1}_{(0,n)}) \end{cases} \tag{4}$$

表示。

根据式(3)计算零编码上下文时，在求得 $cx^{l}_{zc(0,n)}$ 与 $cx^{l}_{zc(1,n)}$ 后，同理 $s^{l}_{(0,n)}$ 与 $s^{l}_{(1,n)}$ 也已经通过式(2)更新为 $s^{*l}_{(0,n)}$ 与 $s^{*l}_{(1,n)}$，即进一步计算 $cx^{l}_{zc(0,n+1)}$ 与 $cx^{l}_{zc(2,n)}$ 的条件同时满足。之后的 $cx^{l}_{zc(1,n+1)}$ 和 $cx^{l}_{zc(3,n)}$ 也如此。符号编码上下文计算的情况更为简单一些。根据式(4)，在求得 $cx^{l}_{sc(0,n)}$ 与 $cx^{l}_{sc(1,n)}$ 后，$s^{l}_{(0,n)}$ 已通过式(2)更新为 $s^{*l}_{(0,n)}$，即可以同时计算 $cx^{l}_{sc(0,n+1)}$ 与 $cx^{l}_{sc(2,n)}$，之后的 $cx^{l}_{zc(1,n+1)}$ 和 $cx^{l}_{zc(3,n)}$ 也如此。

考虑到游程编码的上下文较为规律，可基于查表的思想实现，因此结合以上规律，得到如图 2 所示的适用于显著性传播过程和清除过程的一种叉形编码路径，使得完成一个 $4 \times n$ 大小条带编码的路径长度缩减为标准的 $(n+1)/(2n)$，可有效提高编码电路的并行度，且优化效果随着代码块尺寸的变大愈发明显。并根据式(1)、(2)、(3)、(4)易知，可基于组合电路设计实现显著性传播过程和清除过程，以提高编码速度。

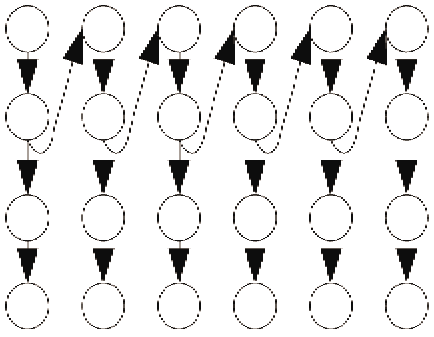

图 2 样本点叉形编码路径

2.2 两窗口流水线结构

基于以上结论，可在单个时钟内完成若干样本点的显著性传播过程和清除过程编码。而幅值细化过程是对显著性传播过程后尚未编码的显著性样本点进行操作，其操作对象及相应的状态位均可以通过显著性传播过程指示，编码结果对清除过程不产生影响，故可与清除过程并行进行，也可基于组合电路实现。

综合考虑编码路径长度对系统时钟的影响以及离散小波变换(DWT)到位平面编码(BPC)的数据结构和接口传输效率问题[8]，提出了一种两窗口流水线结构，如图 3 所示。两窗口分别指实现显著性传播过程的 SPP 窗口和同时实现幅值细化过程与清除过程的 MRP&CP 窗口。该结构将条带内数据细分为 4×4 的样本点块为基本单元进行操作。编码时，在第一个时钟内读入第一个 4×4 样本点块，执行 SPP 窗口，进行显著性传播过程编码；在第二个时钟内读入第二个 4×4 样本点块，执行 SPP 窗口，进行显著性传播过程编码；同时，对第一个 4×4 样本点块并行执行 MRP&CP 窗口，进行幅值细化过程与清除过程编码。通过这种两编码窗口流水线滑动的方式，平均每个时钟可完成 16 个样本点的位平面编码。依次对条带内的 4×4 样本点块重复以上操作来完成整个条带的位平面编码，进而完成整个位平面的。

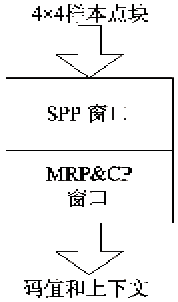

图 3 两窗口流水线结构

2.3 位平面并行编码

文献[7]证明，多个位平面并行编码的条件是预先求得各位平面上样本点的显著性状态、系数符号和细化状态。其中，系数符号可通过小波系数直

接求得。第 k 位平面上第 l 条带的第 m 行 n 列样本点的显著性状态 $s^l_{(m,n,k)}$ 和细化状态 $\eta^l_{(m,n,k)}$，可通过下式求得：

$$s^l_{(m,n,k)} = \sum_{i=k+1}^{MSB} v^l_{(m,n,i)} \tag{5}$$

$$\eta^l_{(m,n,k)} = \begin{cases} 1, & s^l_{(m,n,k)} = 1 \text{ 且 } \sum_{i=k+1}^{MSB} s^l_{(m,n,i)} = 0 \\ 0, & \text{其他} \end{cases} \tag{6}$$

式中，MSB 为为最高位。

因此，若基于式(5)和(6)对量化后的小波系数进行处理，并将各位平面的相关状态位预存在缓存中，则两窗口流水线结构也可以实现在多个位平面间并行执行。

3 位平面编码器 VLSI 结构

3.1 整体结构

单个位平面的编码器硬件结构如图 4 所示，主要包括两个编码窗口、控制模块、时钟模块和缓存。完整的位平面编码器由若干同构的单个位平面编码器并行组成。缓存用来预存各位平面的显著性状态、系数符号和细化状态，以及其他过程变量。量化后的小波系数被划分为多个位平面后，并行流入各个单位平面编码器进行编码，最后将码值和上下文传递给后级的算术编码器。

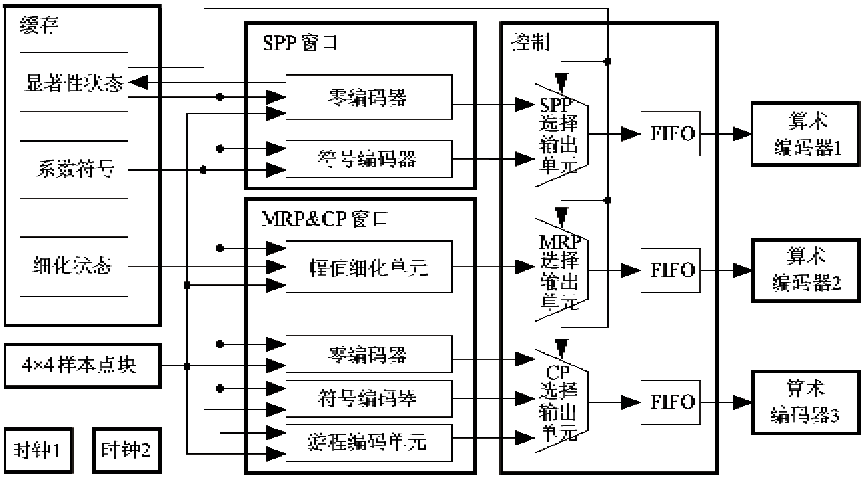

图 4 单个位平面的编码器硬件结构

3.2 SPP 窗口与 MRP&CP 窗口

SPP 窗口主要包含零编码单元和符号编码单元。对于零编码单元而言，其组合电路一次性完成 16 个点的临域显著性贡献判决和显著性状态更新，并将更新后的显著性状态缓存，之后分别计算样本点临域的水平、垂直和斜角显著性贡献度，进而计算出 16 个点的零编码上下文。零编码码值为样本点幅值。对于符号编码单元而言，直接计算 16 个样本点临域的水平和垂直显著性贡献度，从而求出其符号预测值与符号上下文；符号预测值与原符号值异或后得到符号编码码值输出。

MRP&CP 窗口包含幅值细化单元、游程编码单元、零编码单元和符号编码单元。其中，幅值细化单元根据 16 个样本点临域显著性贡献和细化状态计算幅值细化上下文；幅值细化编码码值为样本点幅值。本窗口中零编码单元和符号编码单元的电路结构与 SPP 窗口中的相同。对于游程编码单元，其编码对象是一列样本点，启动条件是该列及临域的共 18 个样本点当前均不显著；启动后，若列中样本点更新为显著，则停止游程编码，输出当前游程编码结果，并对列中剩余样本点进行零编码和符号编码，否则，仅输出游程编码结果而不进行零编码和符号编码。显然，这种编码方式具有很强规律性，可视为游程编码结果和列中样本点零编码和符号编码结果的组合。为提高电路并行度，本结构在对样本列进行游程编码后直接输出给控制模块，由其利用零编码单元和符号编码单元的输出完成最后组合。

SPP 窗口与 MRP&CP 窗口中的所有编码单元都在每个时钟开始时读入相关数据并进行操作，除必要的状态位和过程变量放回缓存外，所有编码结果直接传递给下一级的控制模块，由其进一步处理。

3.3 控制模块及后级接口

控制模块主要包含 SPP 选择输出单元、MRP 选择输出单元和 CP 选择输出单元。这些硬件单元读入前级并行输入的编码结果进行处理,对分属不同过程的样本点编码结果进行选择,缓冲后串行输出给对应的算术编码器。

但存在这样的接口瓶颈:控制模块平均每个时钟输入 16 个样本点的码值和上下文进行处理;而其后级的算术编码器要求串行输入,使得内部的选择输出单元需要按照相应过程内的编码顺序依次输出数据,输入与输出速率相差较大。为此,同时采取 3 种方法进行优化。第一种,对两窗口编码模块与控制模块采用不同的时钟,即两窗口编码模块采用独立时钟,在单个时钟内同时对 16 个样本点进行编码,而后级控制模块采用高速时钟,与算数编码器一致,以平衡前后两级的数据处理能力;第二种,对每个编码过程的选择输出单元连接一个深度先进先出(FIFO)以增强数据吞吐能力;第三种,每个编码过程连接一个算术编码器,并行处理 3 个编码过程的码值和上下文。

4 实验结果

设小波系数为 P 位,代码块大小为 $N \times N$,单个位平面完成上下文输出所消耗最大时钟周期数为 T_{context},则基于本文方法完成一个代码块所需时钟周期数可以表述为

$$T_{\text{Proposed}} = N \times N/16 + 1 + T_{\text{context}} \quad (7)$$

由上文讨论可知,标准算法是完全串行执行的,其完成一个代码块所需时钟周期数为

$$T_{\text{Std}} = N \times N \times ((P-1) \times 3 + 1) \quad (8)$$

由于 BPPP 算法综合实现了 PP 算法与 SPCS 算法,因此不再对后二者逐一比较。BPPP 算法完成一个代码块实际时钟周期数可表述为

$$T_{\text{BPPP}} = N \times N/4 + T_{\text{context}} \quad (9)$$

比较式(7)、(8)和(9),可知本文方法的理论编码时间最少。

使用 Verilog 语言描述实现全部硬件电路,在 ISE 10.1 环境下进行综合,系统频率可达 203.083MHz。基于 Xilinx V2P 平台(xc2vp30-7ff896)进行验证,受限于现场可编程门阵列(FPGA)器件工艺,测试时将频率设定为 100MHz。为方便与文献[7]中测试结果进行比较,选取了 5 幅标准灰度图像,对其进行离散小波变换与量化处理后,将量化结果预存入 FPGA 的片上内存中,然后下载硬件网表,测试其完成一个代码块所需的时钟周期数目。测试结果如表 1 所示。

表 1 完成单个代码块所需时钟周期数目

代码块大小 32×32(8 个位平面)	Lena	Barbara	Boat	Bike	Woman
本文方法	2860	2860	2860	2860	2860
标准算法	22528	22528	22528	22528	22528
BPPP 算法[7]	8565	9693	7712	9572	9787

由表 1 可知,基于本文方法设计的位平面编码器完成同样大小代码块所需时钟周期数最少,且性能稳定;BPPP 算法的性能依赖于图像,其时钟周期消耗最多的约为本文方法的3.4倍,最少的约为2.7倍;标准算法时钟周期消耗最多,约为本文方法的 7.8倍。

5 结 论

本文设计了一种高性能位平面编码器(BPC)以满足 JPEG2000 在图像实时处理应用中的要求。通过分析标准算法及已有文献,提出了一种适用于显著性传播过程和清除过程的叉形编码路径,使得完成一个 $4 \times n$ 大小条带编码的路径长度缩减为标准的 $(n+1)/(2n)$。基于该编码路径设计了一种可在位平面间并行编码的两窗口流水线编码硬件结构,并分析讨论了系统总体架构。FPGA 验证结果表明,系统频率可综合到203.083MHz,处理 512×512 的灰度图像可达到 276fps。可成为现有 JPEG2000 编码芯片中相应部件的升级方案。

参考文献

[1] JPEG2000 Part 1: Final Draft, document ISO/IEC JTCI/SC29/WGI N1855.doc, International Standard (ISO/IEC FDISF DIS15444-1), Aug.2000

[2] Taubman D. High performance scalable image compression with EBCOT. *IEEE Trans on Image Processing*, 2000, 9(7): 1158-1170

[3] Taubman D, Ordentkich E, Weinberger M, et al. Embedded block coding in JPEG 2000, In: Proceedings of the IEEE International Conference on Image Processing, Vancouver, Canada, 2000. 2: 33-36

[4] Andra K, Chakrabarti C, Acharya T. A high - performance JPEG 2000 architecture. *IEEE Trans on Circuits and System for Video Technology*, 2003, 13(3):209-218

[5] Chiang J S, Lin Y S, Hsieh C Y. Efficient pass-parallel architecture for EBCOT in JPEG 2000. In: Proceedings of the IEEE International Symposium on Circuits and Systems, Phoenix-Scottsdale, USA, 2002. 773-776

[6] Lian C J, Chen K F, Chen H H, et al. Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000. *IEEE Trans on Circuits and System for Video Technology*, 2003, 13(3):219-230

[7] Liu K, Li Y S, Wu C K. A high performance EBCOT coding and its VLSI architecture. *Journal of Software*, 2006, 17(7):1553-1560

[8] Amit K G, Saeid N, Taubman D. Efficient interfacing of DWT and EBCOT in JPEG2000. *IEEE Trans on Circuits and System for Video Technology*, 2008, 18(5):687-693

Design and verification of a JPEG2000 VLSI architecture of bit plane coder using 2-branch coding path

Liu Wensong*, Zhu En*, Wang Jian**, Sun Lei*, Lin Ye*

(*Institute of RF- & OE-ICs, Southeast University, Nanjing 210096)

(**Institute of Automation, Chinese Academy of Sciences, Beijing 100190)

Abstract

Based on the study of the bit plane coding in JPEG2000, the 2-branch coding path for significance propagation pass and cleanup pass was proposed with the aim of reducing the coding path length for a $4 \times n$ strip as $(n+1)/(2n)$ of the standard one. Based on the proposed coding path, a hardware structure called 2-window pipeline capable of bit-plane coding 16 bits per cycle on the average by pipeline sliding of 2 coding windows was proposed, whose key module was implemented by combination circuits, avoiding the cycle consumption and the complicated control, while allowing the parallel working between multiple bit planes. The corresponding VLSI architecture of the whole bit plane coder was constructed. The results of the experiment on FPGA shows that the system frequency can be synthesized into 203.083MHz, which can process 276 frames of 512×512 gray image per second, satisfying the requirement of realtime image processing.

Key words: JPEG2000, bit plane coder, 2-branch coding path, 2-window pipeline