多上下文 MQ 编码器优化与 VLSI 实现

邸志雄[1],史江义[1],刘 凯[2],李云松[2],马佩军[1],郝 跃[1]
(1.西安电子科技大学宽带隙半导体技术国家重点学科实验室,陕西西安 710071;
2.西安电子科技大学 ISN 重点实验室,陕西西安 710071)

摘 要: MQ(Multiple Quantization)编码器由于效率低下已经成为 JPEG2000 的性能瓶颈.本文对 MQ 编码算法中的上下文关系进行了提取,对索引表中的启动态和非暂态进行了分离,并提出一种用于预测索引值的方法.同时,对重归一化运算中出现的大概率事件和小概率事件进行分离,使其可并行对 2 个上下文完成编码.依据该算法,本文提出了一种多上下文并行处理的 MQ 编码器 VLSI 结构.实验结果表明,本文提出的 MQ 编码器能够工作在 286.80MHz,吞吐量为 573.60 Msymbols/sec,相比 Dyer 提出的 Brute Force with Modified Byteout 结构,本文的吞吐量提升约 35%,且面积减小 78%.

关键词: MQ 编码器; JPEG2000; 并行
中图分类号: TP402 **文献标识码:** A **文章编号:** 0372-2112 (2013)05-0918-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.05.014

Optimization of Two-Context MQ-Encoder and Implementation of VLSI Architecture

DI Zhi-xiong[1], SHI Jiang-yi[1], LIU Kai[2], LI Yun-song[2], MA Pei-jun[1], HAO Yue[1],
(1. *State Key Discipline Laboratory of Wide Band Gap Semiconductor Technology, Xidian University, Xi'an, Shaanxi* 710071, *China*;
2. *National Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, Shaanxi* 710071, *China*)

Abstract: MQ-encoder is a key bottleneck in the JPEG2000 image compression system. In this paper, the dependence between contexts in the MQ-encoder has been extracted; and an improved mechanism has been proposed to acquire indexes by separating the "start-up" states and remaining states. Besides, a prediction method has been proposed to resolve the stall problem when the current context is the same as the previous context. Also the probabilities of MPS and LPS coding events in the renormalization procedure have been analyzed; and either of the MPS and LPS coding events has been optimized. Based on this improved arithmetic, a VLSI architecture for Two-context MQ-encoder is proposed. Synthesis result shows that the processing speed of the MQ-encoder could reach as high as 286.80MHz with a throughput of 573.60Msymbols/sec. Compared with the "Brute force with modified byteout" architecture, the throughput obtained in this work has been improved by 35% and the area has been reduced by 78%.

Key words: MQ-encoder; JPEG2000; parallel

1 引言

JPEG2000 中的嵌入式块编码算法主要由 Tier-1 和 Tier-2 两部分构成[1,2].其中 Tier-1 采用位平面编码器(Bit Plane Coder, BPC)生成上下文和精度(Context and Decision, CXD),然后将 CXD 再送入 MQ 编码器进行编码.由于 BPC 并行生成多个 CXD,而 MQ 编码器仅能顺序串行处理,因此,MQ 编码器已经成为制约 JPEG2000 图像压缩系统性能的瓶颈.随着高分辨率图像在航空航天以及消费类电子等领域的广泛应用,对图像压缩的实时性也提出了更高的要求.因此,如何实现高吞吐量的 MQ 编码器已经成为当前图像压缩硬件实现领域所面临的关键问题[3,4].

基于现有研究基础,本文首先提取了上下文之间的关系,重点对上下文相等的情况进行优化.同时分离了索引表中的启动态和非暂态,提出一种预测索引值的方法.此外,本文对重归一化算法进行了优化,使其可并行对 2 个上下文完成编码,且更易于被转化为高性能的硬

收稿日期:2012-04-10;修回日期:2013-03-12
基金项目:陕西省自然科学基金(No.2010JM8015)

件结构.本文在 TSMC 0.18μm CMOS ARM 公司 HD 标准单元库的条件下,使用 Synopsys 公司的 Design-Compiler 工具完成了性能评估.结果表明本文提出的 MQ 编码器工作频率为 286.80MHz,吞吐率为 573.60 Msymbols/sec,相比 Brute Force with Modified Byteout 结构,吞吐量提升约 35%,且面积减小 78%.

2 MQ 编码器概述

2.1 MQ 编码器编码算法说明

MQ 编码器用来将输入的上下文 CX 和精度(Decision,D)所组成的序列映射成单个压缩的码流(Compressed Data,CD),如算法 1 所示[5~7].

算法 1　MQ 编码算法

1:initial: $A=0\times8000$, $C=0$, $CT=12$　//MQ 编码器初始化
2: $Qe=Qe(I(CX))$, $A=A-Qe$
3:if $D=\mathrm{MPS}(CX)$　//大概率编码环境
4:　if $A\geq0\times8000$, $C=C+Qe$
5:　else if $A<Qe$, $A=Qe$
6:　　else $C=C+Qe$
7:　　$I(CX)$NMPS($I(CX)$)　//更新索引值
8:　　DO　//重归一化
9:　　$A=2A$, $C=2C$, $CT=CT-1$
10:　　if CT = 0, ByteOut//调用 byteout,输出码流
11:　　while $A<0\times8000$
12:else if $A<Qe$, $C=C+Qe$　//小概率编码环境
13:　else $A=Qe$
14:　MPS(CX) = MPS(CX)$\oplus$SWITCH($I(CX)$)//条件交换
15:　$I(CX)=\mathrm{NLPS}(I(CX))$　//更新索引值
16:　DO,　//重归一化
17:　　$A=2A$, $C=2C$, $CT=CT-1$
18:　　if CT = 0, ByteOut//调用 byteout 输出码流
19:　while $A<0\times8000$

2.2 硬件实现瓶颈分析及现有相关研究

MQ 编码器硬件电路吞吐量之所以难以提升,主要困扰为:若当前输入的上下文与前一个上下文相等,则 MQ 编码器必须完成当前上下文的编码后,才能处理下一个输入的上下文.若在其中插入流水线,则索引表在当前周期无法更新,使得当前输入的 CX 不能被正确编码.该原因导致了 MQ 编码器存在多级且较为复杂的组合逻辑,造成 MQ 编码器的关键路径冗长.

为了提高 MQ 编码器硬件结构的吞吐量,现有文献的实现手段主要集中两个方面:

(1)实现能够并行处理多个上下文的 MQ 编码器,如文献[8~12].文献[8]中两个索引值的更新必须依序完成,编码效率较低.文献[9]逻辑资源不能共享,导致该编码器面积较大.文献[10]使用了大量的 FIFO,以对上下文进行排队,从而提供适合该机制处理的上下文顺序,导致效率较低,且面积较大.文献[11]中多个寄存器 A 只能串行完成重归一化,导致吞吐量较低.文献[12]提出了一种优化的 byte-out 结构,但是对于制约 MQ 编码器性能瓶颈的概率估计值求解部分和重归一化部分并没有进行优化.

(2)提高串行 MQ 编码器的性能,如文献[3,13~16].文献[13]并未对概率估计值求解部分和重归一化部分进行优化.且由于在处理两字节输出时引入了额外的流水级,导致硬件资源开销较大.文献[14]与文献[15]作者相同,所有的算法运算都在第一级流水中完成,包括索引值求解、概率估计值求解和寄存器 A 的重归一化,导致关键路径非常冗长.文献[16]中所有算术运算操作均通过一个复用的 28bit ALU 实现,尽管减小了电路面积,但是却增大了关键路径的延迟.文献[3]中,当连续输入的上下文相等时,该编码器必须暂停流水线,等待索引值的更新完成,降低了编码器的吞吐量.

3 多下上下文并行编码算法优化

3.1 上下文关系提取

定义　给定上下文 CX_k,求解上下文 CX_k 的索引值需已知变量 CX_{k-1}和 index_{k-1},令求解索引值过程中变量之间关系为依赖系数 β,有 $\beta=(CX_{k-1}\neq CX_k)\cdot f(\mathrm{index}_{k-1})$,其中函数 f 表示索引值更新过程.

由定义知,若 $CX_k=CX_{k-1}$,有 $\beta=f(\mathrm{index}_{k-1})$,故 MQ 编码器只能进行串行编码.若 $CX_k\neq CX_{k-1}$,有 $\beta=0$,即 CX_{k-1}在编码完成之后,其索引值是否更新与对 CX_k 的编码过程无关,此时可并行求解 2 个上下文的索引值.因此,若对 CX_k 和 CX_{k-1}之间的关系进行提取,然后对相等的情况进行优化,可使 MQ 编码器在不同的情况下都能实现并行编码.

由表 1 中的数据可知, $CX_k=CX_{k-1}$出现概率约 46.3%.为了并行处理该类型事件,本文提取了四个上下文(CX_k、CX_{k+1}、CX_{k-2}、CX_{k-1})之间的关系.首先,本文检测当前上下文 CX_k 和 CX_{k+1}之间的关系,共有 C_2^2 种,以确认当前待处理的 2 个上下文之间是否有关联.其次,检测当前任一个上下文与前一次所有上下文的关系,以确认对前一次上下文的索引值求解是否影响当前索引值的计算,该关系共有 2×2^2 种.该方法重点优化制约并行化处理最严重的情况,相比文献[9,10],充分利用了 MQ 编码器的算法特点.

表 1 上下文之间各种关系发生概率

图像	分辨率	上下文总数	$CX_k \neq CX_{k-1}$ 发生总数及概率	$CX_k = CX_{k-1}$ 发生总数及概率
Lena	512×512	1831388	1066263(58.2%)	764869(41.8%)
pentagon	1024×1024	8414691	4541745(54.0%)	3871922(46.0%)
airport	1024×1024	8500955	4488418(52.8%)	4011513(47.2%)
bike	512×640	2666095	1419730(53.3%)	1246039(46.7%)
bridge	1024×1024	9158161	4602823(50.3%)	4554314(49.7%)
平均值	- - - -	6114258	3223796(53.7%)	2889731(46.3%)

3.2 索引表优化

根据算法 1 可知,若 $CX_k = CX_{k-1}$,则 index_k 取值可能为:$\text{MPS}(\text{index}_{k-1})$、$\text{LPS}(\text{index}_{k-1})$、$\text{index}_{k-1}$;若 $CX_{k+1} = CX_k = CX_{k-1}$,则 index_{k+1} 取值可能为:index_{k-1},$\text{MPS}(\text{index}_{k-1})$,$\text{LPS}(\text{index}_{k-1})$,$\text{MPS}(\text{MPS}(\text{index}_{k-1}))$,$\text{MPS}(\text{LPS}(\text{index}_{k-1}))$,$\text{LPS}(\text{LPS}(\text{index}_{k-1}))$ 和 $\text{LPS}(\text{MPS}(\text{index}_{k-1}))$.若将以上归纳为函数 $f(\text{index}_{k-1})$,可直接求出 index_k 与 index_{k+1}.

MQ 编码器状态转移表中,状态 index = 0 到状态 index = 13 为启动部分,内容复杂,且没有规律,本文用查找表实现.而 index = 14 到 index = 46 为非暂态部分,其中 MPS 总是引起索引值 index 递增.LPS 引起索引值 index 递减,但是在 LPS 概率约为 1/4 处(index = 21),递减参数由 1 变为 2.故在非暂态状态下,本文将 index_k 与 index_{k+1} 的求解归纳为如下公式,其中 index = 14 时,LPS 没有引起状态索引变化,因此本文将其排除.

$$\text{MPS}(\text{index}_{k-1}) = \begin{cases} \text{index}_{k-1}+1, 15 \leqslant \text{index}_{k-1} \leqslant 44 \\ \text{index}_{k-1}, \text{index}_{k-1} = 45, 46 \end{cases} \quad (1)$$

$$\text{LPS}(\text{index}_{k-1}) = \begin{cases} \text{index}_{k-1}-1, 15 \leqslant \text{index}_{k-1} \leqslant 20 \\ \text{index}_{k-1}-2, 21 \leqslant \text{index}_{k-1} \leqslant 45 \\ \text{index}_{k-1}, \text{index}_{k-1} = 46 \end{cases} \quad (2)$$

$$\text{MPS}(\text{MPS}(\text{index}_{k-1})) = \begin{cases} \text{index}_{k-1}+2, 15 \leqslant \text{index}_{k-1} \leqslant 43 \\ 45, \text{index}_{k-1} = 44 \\ 46, \text{index}_{k-1} = 45, 46 \end{cases} \quad (3)$$

$$\text{MPS}(\text{LPS}(\text{index}_{k-1})) = \begin{cases} \text{index}_{k-1}, 15 \leqslant \text{index}_{k-1} \leqslant 19, \text{index}_{k-1} = 46 \\ \text{index}_{k-1}-1, 20 \leqslant \text{index}_{k-1} \leqslant 44 \\ 43, \text{index}_{k-1} = 45 \end{cases} \quad (4)$$

$$\text{LPS}(\text{MPS}(\text{index}_{k-1})) = \begin{cases} \text{index}_{k-1}, 15 \leqslant \text{index}_{k-1} \leqslant 20, \text{index}_{k-1} = 46 \\ \text{index}_{k-1}-1, 21 \leqslant \text{index}_{k-1} \leqslant 45 \end{cases} \quad (5)$$

$$\text{LPS}(\text{LPS}(\text{index}_{k-1})) = \begin{cases} \text{index}_{k-1}-1, \text{index}_{k-1} = 15 \\ \text{index}_{k-1}-2, 16 \leqslant \text{index}_{k-1} \leqslant 20 \\ \text{index}_{k-1}-3, 21 \leqslant \text{index}_{k-1} \leqslant 22 \\ \text{index}_{k-1}-4, 23 \leqslant \text{index}_{k-1} \leqslant 45 \\ 46, \text{index}_{k-1} = 46 \end{cases} \quad (6)$$

文献[9]、[14]和[16]并未对索引值状态转移表中的启动态和非暂态进行分离,故启动态中复杂而又无规律的数据使得其提出的索引值求解函数中包含多个逻辑分支,且每个分支中均包含大量的算术运算操作符.

3.3 索引值求解优化

若上下文 index_{k-1} 编码后索引值发生更新,则本文认为索引值保持次数为 0;若上下文 index_{k-1} 编码后索引值保持,则认为索引值保持次数为 1.据此可得:若 $CX_{k+1} = CX_k = CX_{k-1}$,则索引值保持次数可能为 0、1、2.此时,由 3.2 节可知,CX_{k+1} 的索引值有 7 种可能.为了获得图像编码过程中索引值是否保持的概率,本文采用了与 3.1 节中相同的 5 幅图片作为实例,统计结果如图 1 所示.

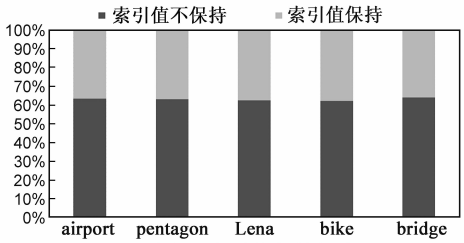

图1 索引值保持概率

由图 1 可知,索引值保持次数非零的概率约为 36.94%.因此,若能提前判断出索引值的保持次数,则可较为简捷地从所有可能索引值中选出正确的索引值.

由 MQ 标准算法可知,若 $CX_k = CX_{k-1}$,则通过式(7)是否成立,即可选出其正确的索引值 index_k.

$$A_{k-2} - Qe_{k-1} \geqslant 0x8000 \quad (7)$$

同样地,根据 $A_{k-1} - Qe_k \geqslant 0x8000$,可得出上下文 CX_k 的编码是否引起了索引值 index_k 的更新.故 CX_k 索引值是否保持的条件可有如下推导:$A_{k-1} - Qe_k = (A_{k-2} - Qe_{k-1}) - Qe_{k-1} = A_{k-2} - 2Qe_{k-1}$.因此,可通过式(8),确定上下文 CX_k 的编码是否引起了索引值 index_k 的更新.

$$A_{k-2} - 2Qe_{k-1} \geqslant 0x8000 \quad (8)$$

综上,若 $CX_{k+1} = CX_k = CX_{k-1}$,本文提出了索引值预测方法,如算法 2 所示.

算法 2　索引值预测方法

If$(A_{k-2} - 2Qe_{k-1} \geq 0x8000)$, $\text{index}_{k+1} = \text{index}_{k-1}$

Else if $((A_{k-2} - Qe_{k-1} \geq 0x8000)\ \&\ (A_{k-2} - 2Qe_{k-1} < 0x8000))$,

$\text{index}_{k+1} = (\text{index}_{k-1})^1$

Else $\text{index}_{k+1} = (\text{index}_{k-1})^2$

算法 2 中,$(\text{index}_{k-1})^1$ 表示 MPS(index_{k-1}),LPS(index_{k-1});$(\text{index}_{k-1})^2$ 表示 MPS(MPS(index_{k-1})),MPS(LPS(index_{k-1})),LPS(LPS(index_{k-1}))和 LPS(MPS(index_{k-1})).根据当前所处的编码环境,即可选出(index_{k-1})^1 和(index_{k-1})^2 正确的值.据此方法,上下文 CX_{k+1}的索引值可与 CX_k 的重归一化并行计算.相比文献[3,8,9,10],可提高 MQ 编码器的并行度.同时,即使连续输入的上下文相等,MQ 编码器也不会出现停顿.

3.4　重归一化优化

由算法 1 可知,MQ 编码算法通过"do-while"函数来完成重归一化,其中迭代次数取决于式(9)或(10)的计算结果 A'_k 的前导零个数(leading zero,LZ).

$$\text{MPS: } A'_k = A_{k-1} - Qe,\ C'_k = C_{k-1} + Qe \tag{9}$$

$$\text{LPS: } A'_k = Qe,\ C'_k = C_{k-1} \tag{10}$$

根据概率估计值表 PET 可知,$\max(Qe) = 0x5601$,又 $A \geqslant 0x8000$,故 $\min(A - Qe) = 0x29FF$,即在公式(10)被执行的情况下,前导零个数 $\max(LZ) = 2$.而 index$\in$[43,44,45]时,$Qe \leqslant 0x0009$ 且 $\min(Qe) = 0x0001$,即 $\max(LZ') = 15$.因此在执行公式(9)时,对当前上下文编码不一定会使计数器自减为 0.而在执行公式(10)时,计数器存在仅通过一次重归一化即可自减为 0 的可能,且有可能生成 1 个甚至 2 个压缩码流 CD.

如图 2 所示,公式(10)被执行的平均概率达 25.1%.因此,本文分别直接提取式(10)和式(9)计算结果 A'_k 的前导零个数 LZ_1 和 LZ'_1,对上下文 CX_{k+1}的重归一化结果进行预测和选择.从而使得上下文 CX_{k+1}的重归一化计算可并行于上下文 CX_k 的重归一化.同时,可直接获取前导零个数,重归一化过程仅通过一次运算即可完成,同文献[14]相比,不再多次迭代.此外,本文将寄存器 C 由 28bit 扩充为 43bit,前导零个数大于 12 时,避免寄存器 C 的有效位被丢弃.根据以上分析,本文对重归一化算法进行了优化,如算法 3 所示.

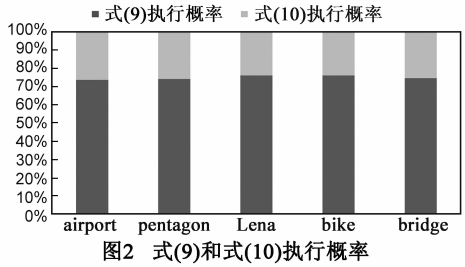

图2　式(9)和式(10)执行概率

算法 3　重归一化算法

初始化　$A \leftarrow 0x8000$, $C \leftarrow 0$

If $(D_k = \text{MPS}[CX_k])$

　$A_k = (A_{k-1} - Qe_k) \ll LZ_1$, $C_k = (C_{k-1} + Qe_k) \ll LZ_1$

　If $(D_{k+1} = \text{MPS}[CX_{k+1}])$

　　$A_{k+1} = ((A_{k-1} - Qe_k) \ll LZ_1 - Qe_{k+1}) \ll LZ_2$,

　　$C_{k+1} = ((C_{k-1} + Qe_k) \ll LZ_1 + Qe_{k+1}) \ll LZ_2$

　Else

　　$A_{k+1} = Qe_{k+1}$, $C_{k+1} = C_k$

Else

　$A_k = Qe_k$, $C_k = C_{k-1}$

　If $(D_{k+1} = \text{MPS}[CX_{k+1}])$

　　$A_{k+1} = (Qe_k \ll LZ'_1 - Qe_{k+1}) \ll LZ'_2$,

　　$C_{k+1} = (C_{k-1} + Qe_{k+1}) \ll LZ'_2$

　Else

　　$A_{k+1} = Qe_{k+1}$, $C_{k+1} = C_k$

采用该优化后的算法,相比文献[11],在一个时钟周期之内可并行完成的 2 个上下文重归一化计算,避免了电路因串行等待造成的效率低下.

4　MQ 编码器 VLSI 结构设计

4.1　四级流水结构

基于以上优化后的算法,本文提出了一种四级流水的多上下文并行 MQ 编码器,如图 3 所示:

第一级流水完成上下文关系提取,索引表更新,可能索引值及可能概率估计值求解.其中上下文关系提取根据 3.1 节的算法,可能索引值查找依据 3.2 节提出的算法.第二级流水采用 3.3 节的算法选出正确的 index_k 与 index_{k+1}和 Qe_k 与 Qe_{k+1};依据 3.4 节提出的算法,对寄存器 A_k 和 A_{k+1}进行重归一化.第三级流水采用 3.4 节的算法,完成寄存器 C_k 和 C_{k+1}的重归一化.第四级流水为码流输出,由于该级的运算不包含加法器、移位器等逻辑较为复杂的运算单元,因此该级依然采用未优化的标准算法实现硬件电路.

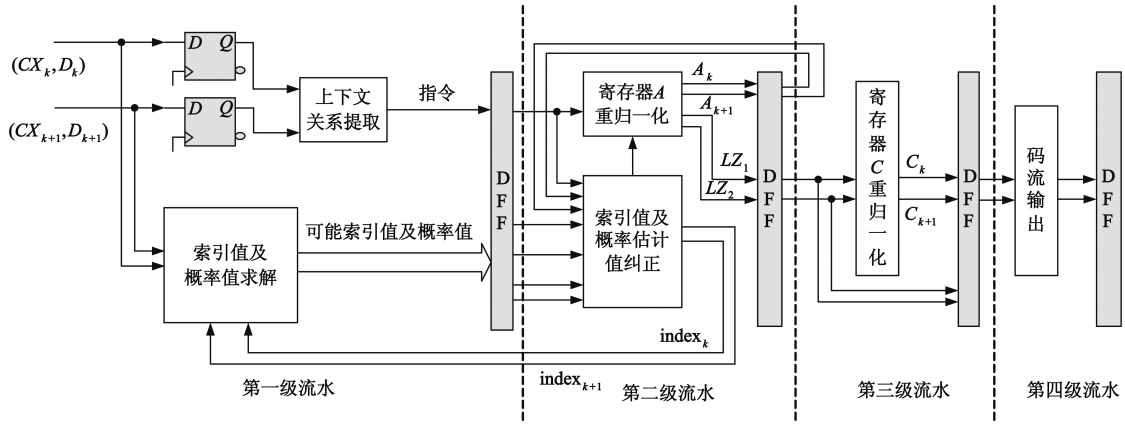

图3 MQ编码器VLSI设计结构

4.2 第一级流水电路结构

如图4所示,本文采用5个比较器和2个DFF提取上下文之间的关系,用于:(1)第一级流水中选择出可能的索引值;(2)寄存到第二级流水,选择出正确的索引值.同时,根据3.2节提出的方法,启动态求解部分使用ROM(规模为15×36)实现查找表的功能;非暂态求解部分通过3.2节提出的函数实现.概率估计值求解部分使用ROM(规模为47×20)实现,其功能为函数 $Qe(I(CX))$、SWITCH($I(CX)$)和 $LZ(Qe)$,其中 $LZ(Qe)$为 Qe 的前导零个数.由于 Qe 最高位始终为 $1'b0$,因此本文将该比特位存放SWITCH($I(CX)$).索引值更新部分使用RAM(规模为19×6)实现,其功能等同于 $I(CX)$.该电路结构的优点在于:(1)简化了多上下文并行索引值和概率估计值求解过程,使得电路资源小于文献[8,9];(2)启动态和非暂态分离,使得各自都采用了更为适合的硬件实现方式,相比文献[9,14]减小了由此带来的传播延迟.

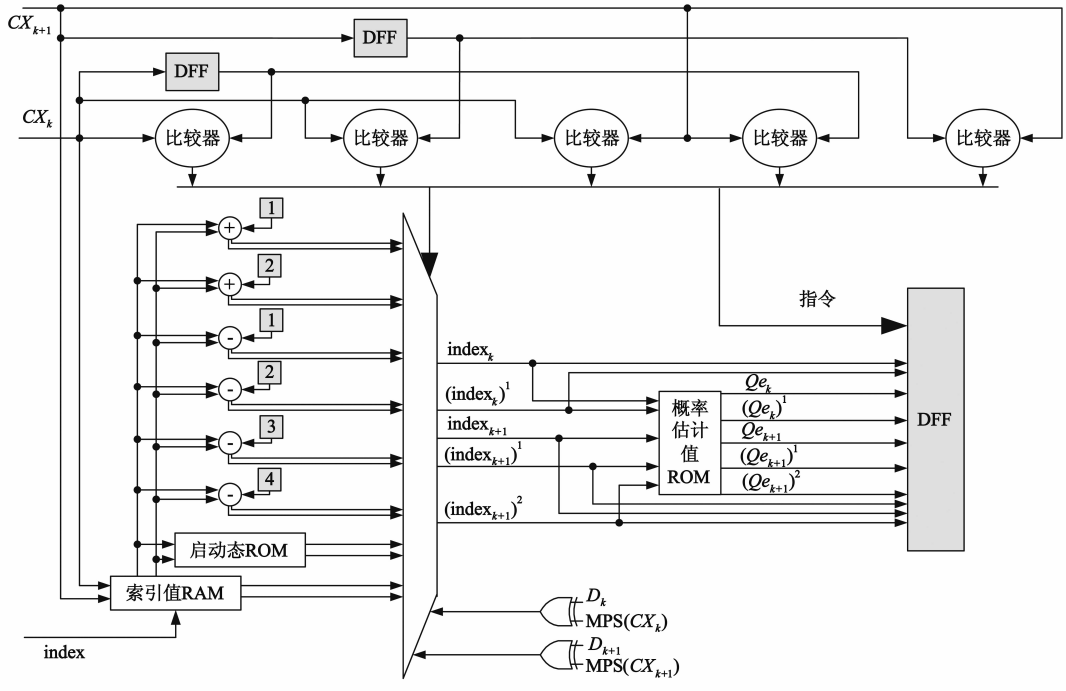

图4 第一级流水电路结构

4.3 第二级流水电路结构

根据3.3节的算法可知,对索引值进行选择须获得前一个周期与其相等的上下文的计算结果 A 和 Qe.为了给当前上下文分配正确的值,本文提出了一种分配结构,如图5所示.

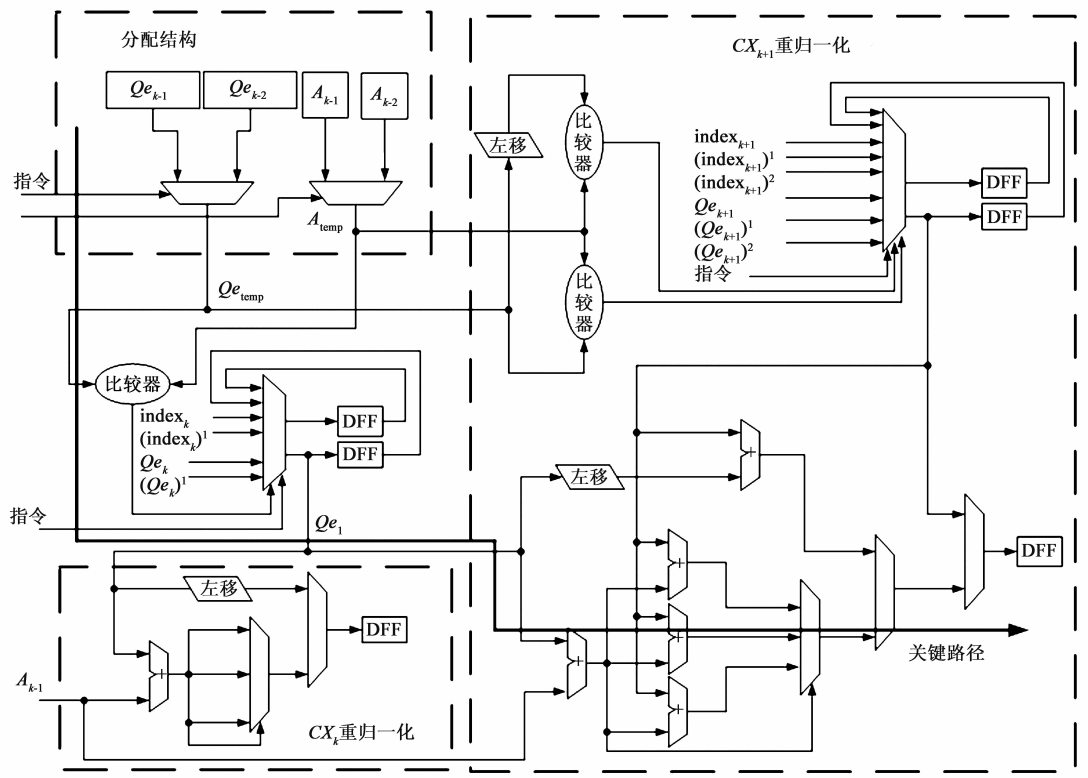

图5　第二级流水电路结构

分配方法为:

(1)若 $CX_k = CX_{k-1}$,则将 A_{k-1}和 Qe_{k-1}分配给上下文 CX_k.此时不论是否存在 $CX_{k-2} = CX_{k-1}$, CX_{k-2}的编码结果都对 CX_k 无影响.若 $CX_k = CX_{k-2}$且 $CX_k = CX_{k-1}$,则将 A_{k-2}和 Qe_{k-2}分配给上下文 CX_k.

(2)若 $CX_{k+1} = CX_k = CX_{k-1}$,则将 A_{k-1}和 Qe_{k-1}分配给上下文 CX_{k+1}.

其他情况类似以上方法.若上下文 CX_k 和 CX_{k+1}与前一周期被编码的上下文 CX_{k-2}和 CX_{k-1}无任何联系,则根据第一级流水中生成的指令直接进行索引值和概率估计值的选取.得到正确的概率估计值之后,依据 3.4 节和 3.5 节提出的重归一化算法,完成寄存器 A 的重归一化操作.

本文所提出 MQ 编码器的关键路径:起点为 Qe_{k-1},终点为 A_{k+1}.与文献[8-11,14,15]相比,本文的关键路径中,标准算法中的前导零检测器、移位器等复杂的逻辑运算单元都被较为简单的逻辑电路所替代.且相比文献[10],无需排队,效率更高.相比文献[8,11],重归一化被并行实现.

5　实验结果分析

本文使用 VerilogHDL 对提出的 MQ 编码器电路结构进行了描述,并且与其他有代表性的算术编码器结构进行了性能对比,如表 2 所示.

表 2　STRATIX 平台性能对比

电路结构	Symbol/Clk	时钟频率(MHz)	吞吐量(Msymbols/sec)	吞吐量归一化	资源使用(LEs)
文献[15]	1	153	153	0.90	279
文献[14]	1	145.9	145.9	0.85	824
文献[16]	1	50.05	57.05	0.34	761
文献[9]	2	48.30	96.60	0.57	12649
Brute force with modified byteout[10]	2	48.85	97.70	0.57	1596
Bruteforce[10]	2	36.90	73.80	0.43	2265
3CXD hypothesis[10]	3	45.53	60.78	0.36	1259
3CXD hypothesis with queue[10]	3	49.50	71.48	0.42	1744
文献[12]	2	58.56	117.12	0.69	1488
文献[7]	2	50.10	100.2	0.59	1596
本文	2	85.40	170.80	1	1554

文献[14,15]将所有复杂的运算都集中在第一级流水中,制约了 MQ 编码器的工作频率,故相比文献[14],本文吞吐量提高了 15%左右;相比文献[15],本文吞吐量提高了 10%.文献[10]提出多种并行处理多个上下文的 MQ 编码器,但是由于其采用了复杂的预测结构,且引入了队列机制对上下文进行排队,使得其面积较大,且吞吐量较低.相比文献[10]吞吐量最高的 Brute

force with modified byteout 结构,本文的吞吐量提高了43%,资源使用率略小于文献[10].文献[9]的索引值求解过程使用了大量的算术运算单元,因此资源使用率较高.相比文献[9],本文吞吐量提高了 43%,资源使用率仅为其 12%.文献[12]由于并未优化 MQ 编码器中瓶颈问题,因此吞吐量为本文的 69%.

本文在 TSMC 0.18μm CMOS ARM 公司 HD 标准单元库的条件下,使用 Synopsys 公司的 Design-Compile 工具完成了对 MQ 编码器的性能评估.如表 3 所示,文献[13]由于引入了额外的流水级,且未对 MQ 编码器的性能瓶颈进行优化,因此其吞吐量为本文的 72%.尽管文献[3]工作频率高于本文,但是由于其在处理连续上下文相同的情况时,流水线需暂停,因此该文献的吞吐量也低于本文.相比文献[10]Dyer 提出的 Brute force with modifid byteout 结构,本文的吞吐量提升约 35%,且面积减小 78%.

表 3 TSMC **0.18**μm 工艺性能对比

电路结构	Symbol/Clk	时钟频率(MHz)	Throughput(Msymbols/sec)	吞吐量归一化	面积(等效门)
文献[13]	1	413	413	0.72	8.8K
Two Symbol[9]	2	220.10	440.20	0.77	24955
文献[17]	1	110	110	0.19	9.8K
文献[3]	1	578	520	0.91	7.8K
Brute force with modifid byteout[10]	2	211.86	423.72	0.74	29608
Bruteforce[10]	2	194.17	388.34	0.50	29877
3CXD hypothesis[10]	3	201.61	268.14	0.48	19257
3CXD hypothesis with queue[10]	3	205.34	301.85	0.53	20862
本文	2	286.80	573.60	1	16614

6 结论

本文提出一种多上下文并行处理的 MQ 编码器 VLSI 结构,其主要优点在于:(1)对上下文之间的关系进行了提取,同时对不同关系下上下文之间的依赖性进行了分析;(2)优化了索引表,将其中的启动态和非暂态分离;(3)提出了一种预测索引值的方法,使得本文在连续上下文相同时也不需要暂停流水线;(4)对重归一化运算中出现的大概率事件和小概率事件进行分离,使得可同时并行处理两个上下位.综合结果表明,本文提出的 MQ 编码器能够工作在 286.80MHz,吞吐率为 573.60 Msymbols/sec,相比 Brute Force with Modified Byteout 结构,本文的吞吐量提升约 35%,且面积减小 78%.

参考文献

[1] 刘雷波,王学进,孟鸿鹰,王志华,陈弘毅,夏宇闻.JPEG2000 小波变换器的 VLSI 结构设计[J].电子学报,2002,30(11):1609 – 1612.
Liu Leibo, Wang Xuejin, Meng Hongying, Wang Zhihua, Chen Hongyi, Xia Yuwen. A VLSI architecture of DWT for JPEG2000[J]. Acta Electronica Sinica, 2002, 30(11): 1609 – 1612. (in Chinese)

[2] 朱悦心,付昀,吴宗泽,郑南宁.基于多级查询表的 JPEG2000 位平面扫描优化方法[J].电子学报,2004,32(5):810 – 813.
Zhu Yuexin, Fu Yun, Wu Zongze, Zheng Nanning. A JPEG2000 optimized bit-plane scanning method based on multilevel query table[J]. Acta Electronica Sinica, 2004, 32(5): 810 – 813. (in Chinese)

[3] 刘文松,朱恩,王健,徐龙涛,林叶.JPEG2000 算术编码器的算法优化和 VLSI 设计[J].电子学报,2011,39(11):2486 – 2491.
Liu Wensong, Zhu En, Wang Jian, Xu Longtao, Lin Ye. Optimization algorithm and VLSI design of the arithmetic coder in JPEG2000[J]. Acta Electronica Sinica, 2011, 39(11): 2486 – 2491. (in Chinese)

[4] M D Adams, F Kossentini. Jasper: a software-based JPEG-2000 codec implementation[A]. Proceedings 2000 International Conference on Image Processing[C]. Canada, 2000.2: 53 – 56.

[5] Tinku Acharya, Ping-Sing Tsai. JPEG2000 Standard for Image Compression Concepts, Algorithms and VLSI Architectures [M]. New Jersey: John Wiley & Sons INC, Publication, 2005: 185 – 196.

[6] David S. Taubman, Michael W. Marcellin. JPEG2000 Image Compression Fundamentals, Standard and Practics[M]. America: Kluwer Academic Publishers, 2001: 56 – 77, 473 – 483.

[7] A Gupta, S Nooshabadi, D Taubman, M Dyer. Realizing low-cost high-throughput general-purpose block encoder for JPEG 2000[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2006, 16(7): 843 – 858.

[8] Nandini Ramesh Kumar, Wei Xiang, Yafeng Wang. Two-symbol FPGA architecture for fast arithmetic encoding in JPEG2000 [J]. Journal of Signal Processing Systems, 2012, 69(2): 213 – 224.

[9] Kai Liu, Yu Zhou, Yun Song Li, Jian Feng Ma. A high performance MQ encoder architecture in JPEG2000[J]. INTEGRATION, the VLSI Journal, 2010, 43: 305 – 317.

[10] M Dyer, D Taubman, S Nooshabadi, A Kumar Gupta. Concurrency techniques for arithmetic coding in JPEG2000[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2006, 53(6): 1203 – 1213.

[11] N Noikaew, O Chitsobhuk. Dual symbol processing for MQ

arithmetic coder in JPEG2000 [A]. 2008 International Congress on Image and Signal Processing(CICP)[C]. China, 2008.1.1521 – 1524.

[12] PENG Zhou, ZHAO Bao-jun. High-throughput hardware architecture of MQ arithmetic coder[A]. 2010 10th International Congress on Image and Signal Processing Proceedings (ICSP 2010)[C]. China, 2010.430 – 433.

[13] Minsoo Rhu, In-Cheol Park. Optimization of arithmetic coding for JPEG2000[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2010, 20(3): 446 – 451.

[14] Kishor Sarawadekar, Swapna Banerjee. VLSI design of memory-efficient, high – speed baseline MQ coder for JPEG2000 [J]. INTEGRATION, the VLSI journal, 2012, 45: 1 – 8.

[15] Sarawadekar Kishor, Banerjee Swapna. Low-cost, high-performance VLSI design of an MQ coder for JPEG 2000[A]. 2010 10th International Congress on Image and Signal Processing Proceedings(ICSP 2010)[C]. China, 2010.397 – 400.

[16] M Dyer, S Nooshabadi, D Taubman. Design and analysis of system on a chip encoder for JPEG 2000[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2009, 19 (2): 215 – 225.

[17] Y Z Zhang, C Xu, W T Wang, L B Chen. Performance analysis and architecture design for parallel EBCOT encoder of JPEG2000[J]. IEEE Transactions Circuits Systems for Video Technology, 2007, 17(10): 1336 – 1347.

作者简介

邸志雄　男, 1984 年 10 月出生于山西忻州. 西安电子科技大学微电子学院博士研究生. 研究方向为 SoC 设计方法学、HDL 源代码质量评估方法学等.

E-mail: zxdi@mail.xidian.edu.cn

史江义　男, 1973 年 9 月出生于陕西韩城, 西安电子科技大学微电子学院副教授. 研究方向为 SoC 设计方法学及物理实现.