

基于 HLS 的 HEIF 图像预测编码模块设计

胡锦^{1†}, 黄明帅¹, 毕晓猛²

(1. 湖南大学物理与微电子科学学院, 湖南 长沙 410082; 2. 中大智能科技股份有限公司, 湖南 长沙 410082)

摘 要:设计使用高层次综合语言 HLS(High Level Synthesize)来实现图像预测帧内编码模块的功能。帧内预测编码是指利用视频空间域的相关性,使用当前图像已编码的像素预测当前像素,然后将预测残差(当前像素真实值与预测值之间的差值)作为后续编码模块的输入,进行下一步编码处理。HLS 实现相比与直接使用 verilog 和其他的语言来实现,HLS 具有开发周期短,可以极大地节省人力成本和时间成本、平台可移植性强以及更便捷的代码迭代等的优势。设计使用 HLS 实现的 HEIF 编码 IP 中图像预测编码模块,主要用于 FPGA 中,优化后的 IP 总占用 10795 个 FIFO 和 30705 个 LUT,并且设计在后续优化代码之后缩短了 1/3 原有编码所需时间,大大提升了编码速率。

关键词:图像帧内预测编码;预测残差;HEIF 编码;HLS

中图分类号:TN46

文献标识码:A

Design of HEIF Image Prediction Coding Module Based on HLS

HU Jin^{1†}, HUANG Mingshuai¹, BI Xiaomeng²

(1. School of Physics & Electronics, Hunan University, Changsha, Hunan 410082, China;

2. Zhongda Intelligent Technology Co., Ltd., Changsha, Hunan 410082, China)

Abstract: The design uses high level synthesize(HLS) language to realize the function of image prediction in-frame coding module. Intra-frame predictive coding refers to using the correlation of the video spatial domain to predict the current pixel using the encoded pixel of the current image, and then using the predicted residual (the difference between the real value of the current pixel and the predicted value) as the input of the subsequent coding module for the next encoding process. Compared with the implementation of verilog and other languages directly, HLS has the advantages of short development cycle, great savings in labor cost and time cost, strong platform portability and more convenient code iteration. The image prediction coding module of HEIF encoded IP implemented by HLS is designed, which is mainly used in FPGA. The optimized IP occupies 10795 FIFO, 30705 and LUT, and the design reduces the time required for original coding by 1/3 after the subsequent optimization of the code, greatly improving the coding rate.

Key words: image intra-frame prediction coding; prediction residual; HEIF encoding; HLS

HEIF 图像编码方法包括:获取 HEIF 图像数据;对图像数据进行计算,获得图像数据分量;对所述图像数据分量进行编程处理,获得所述 HEIF 图像数据的编码数据^[1]。视频序列中的第一帧图像

以及其他 I 帧的图像,在编码时没有可供参考的图像,只能利用自身的局部像素信息来降低空间冗余,该过程即为帧内编码。帧内预测技术是消除视频空间冗余的主要技术之一^[2],因此帧内编码是

HEVC 编码中不可或缺的技术^[3]。HLS 具备很多其他高级语言所不具备的特点,比如任意精度数据库、HLS 串流库以及 HLS 预处理指示等。基于 C/C++ 语言的原生数据类型都位于 8 位边界(8 位、16 位、32 位和 64 位)上。RTL 总线(对应于硬件)支持任意数据长度。使用标准 C/C++ 语言数据类型可能导致硬件实现效率低下。任意精度(AP)数据类型允许代码使用位宽更小的变量,并且可供 C/C++ 语言仿真用于确认功能保持不变或者可接受。较小的位宽所生成的硬件运算符也更小且运行更快;HLS 串流库,串流是一种数据传输形式,其中数据样本从第一个样本开始按顺序发送,串流不需要地址管理。Vitis_HLS 提供了 C++ 模板类“hls::stream<>”用于对串流数据结构进行建模,其相当于无限深度的 FIFO 并且按照顺序对数据进行读写;HLS 预处理指令则是可以将硬件电路的思维实现在高级语言上,比如预处理指令“#pragma HLS dataflow”中 dataflow 编译指示可启用任务级流水打拍,允许函数和循环在其操作过程中重叠,增加 RTL 实现的并发度,并增加设计的整体吞吐量。与文献[4]、文献[5]以及文献[6]中的传统设计研究方法不同,本次设计综合阐述了如何基于 HLS 实现 HEIF 编码 IP 中 8×8 图像预测模块的功能。

1 编码预测算法原理

1.1 算法实现简述

图像的帧内预测分为两大部分,一部分是帧内

预测模式,另一部分是帧内预测过程。其中帧内预测模式又分为亮度预测模式、亮度预测模式编码以及色度预测模式编码;帧内预测过程主要涉及相邻参考像素的获取、参考像素的滤波以及预测值的计算。在本文提到的设计中对进入 IP 的图像首先进行块的划分,当块划分成 8×8 的大小的时候进入 codeLumaQT8_pred 模块,也就是设计的预测模块。codeLumaQT8_pred 主要分为外循环和内循环,其中外循环是为了满足 HEIF IP 的 WPP 功能,内循环则是循环了 (WPP_NUM+3)×64 次,其中的“WPP_NUM”是指对图像进行参考像素滤波、预测、计算残差、输出等四个步骤,“+3”是对这四个步骤进行了流水线的设计,有助于大大提升图像处理速率,“×64”则是因为每个 8×8 的像素块有 64 个像素点需要进行处理,主要的功能实现部分均在内循环。

1.2 codeLumaQT8_pred 内循环详细流程

内循环的处理步骤主要分为参考像素滤波、35 种模式预测、计算残差和输出四个步骤,所以在代码编写的时候使用变量 read_data、read_data1、read_data2、read_data3 是否为 1 来判断数据处理进行到了哪一步。其中初始化的时候 read_data=true;read_data1=read_data2=read_data3=false,因为一开始就需要进行滤波;然后在第一个像素处理循环结束之后再给变量赋值。如上所述内循环总共循环了 (WPP_NUM+3)×64 次,预测模块的关键步骤循环图示如图 1。

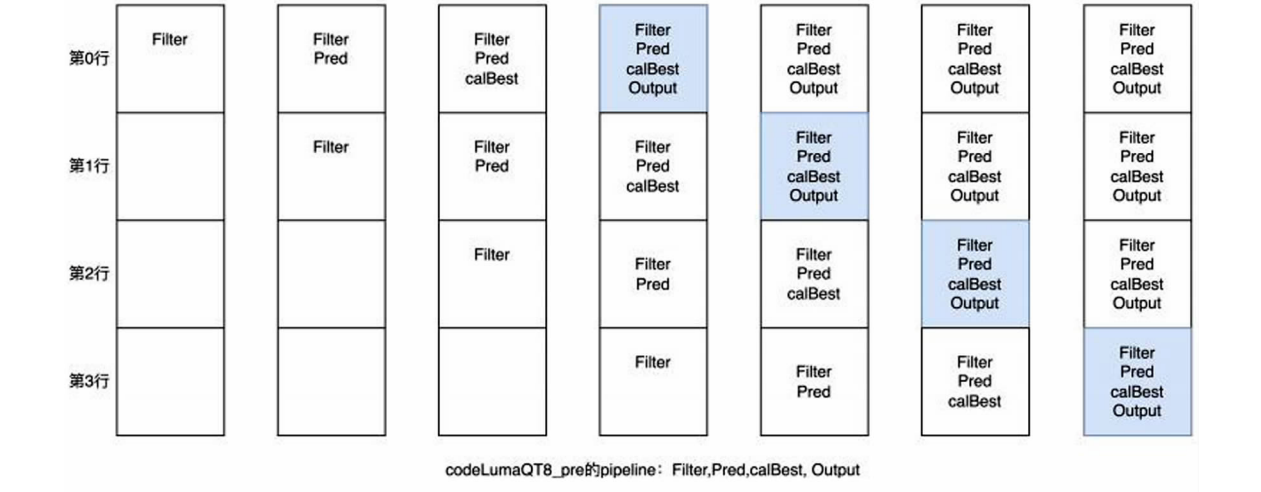


图 1 预测模块关键步骤循环图

下面对内循环的关键步骤进行详细的拆分讲解。

(1)35 个预测模式

根据不同的功能,HEVC 中定义了不同的名称,其中预测单元(Prediccion Unit,PU)^[7]。H. 265/HEVC 亮度分量的帧内预测支持 5 种大小的预测单元 PU:4×4、8×8、16×16、32×32、64×64^[8]。每种 PU 都对应了 35 种预测模式,包括 Planar 模式、DC 模式以及角度模式。所有预测模式使用相同的预测方式。模式 0 为 Planar 模式,模式 1 为 DC 模式,模式 2~34 为角度模式。其中 8×8 亮度块的帧内预测也是如此,第一种是模式 0 (Planar 模式),Planar 模式适用于在水平方向和垂直方向都呈线性变化的图像内容。其计算式如下:

$$P_{x,y}^H = (N - x) \cdot R_{0,y} + x \cdot R_{N+1,0} \tag{1}$$

$$P_{x,y}^V = (N - x) \cdot R_{x,0} + y \cdot R_{0,N+1} \tag{2}$$

$$P_{x,y} = (P_{x,y}^H + P_{x,y}^V + N) \gg (\log_2 N + 1) \tag{3}$$

模式 1 是 DC 模式,DC 模式适用于大面积平坦区域,是使用当前块的预测值由预测模板左侧和上侧(注意不包含左上角、左下方和右上方)参考像素的平均值计算得到。其第一步就是求出该预测单元左侧及上方的像素平均值,如式(4)所示:

$$dcValue = \left(\sum_{x=1}^N R_{x,0} + \sum_{y=1}^N R_{0,y} + N \right) \gg (\log_2 N + 1) \tag{4}$$

模式 2~34 为角度模式,是使用预测模板上侧和左侧的 33 种角度上的参考像素来计算当前块的预测值模式,其中角度模式中的模式 27~34 最为关键,因为它涉及后续参考像素平滑滤波的处理。所有预测模式采用的是相同的预测模板,HEVC 中以逐行扫描的顺序访问编码树单元,在编码树单元内又以深度优先、逐行扫描的顺序访问各个编码单元。因此,除了当前块的左边一列和上边一行像素,HEVC 中当前块的左下角和右上方的像素也可以作为参考像素^[9]。如图 2 所示, $P_{x,y}$ 表示当前块的预测像素, $R_{x,y}$ 表示当前块的参考像素。

(2) 参考像素的平滑滤波

预测值的计算公式是按照模式 27~34 这种方向推导的,在计算其他方向的预测值的时候需要做一些调整。详细的推导过程可以参考 HEVC 的标准文档或者教材。参考像素的强平滑滤波用在 32×32 的亮度块,设计中实现的是 8×8 的亮度块,所以只有 3-tap 的常规平滑滤波。所有尺寸的预测块,DC 预测模式不需要进行平滑滤波。对于 8×8 的块,对 2、18、34 方向进行滤波。

(3) MPM(Most Probable Model)的选择

HEVC 帧内 CU 块划分采用四叉树遍历递归的块划分算法,即从大到小依次划分遍历所有可能的 CU 块类型^[10],所以在一张大图中的每个 8×8 相邻块之间往往存在较强的相关性,因此相邻 PU 的帧内预测模式相同或相似的概率较大。为了去除这部分冗余信息,H. 265 建立了预测模式候选列表 candModelList,表中有 3 个候选预测模式。利用 candModelList 可以降低编码当前 PU 最优模式的码率。同样,设计中根据 cuAboveLumaDir 的模式 (Model1) 和 cuLeftLumaDir 的模式 (Mode2)设置 MPM:如果 Model1 和 Mode2 不同,那么前两个 MPM 是 Model1 和 Mode2,第三个模式从 Planar、DC 和垂直方向中,顺序选择一个和前两个 MPM 不一样的模式;如果 Model1 和 Mode2 相同:如果 Model1 和 Mode2 是非角度模式,那么第一个 MPM 是 Model1,第二和第三个 MPM 按照 Planar、DC 和垂直预测的顺序依次选择;如果 Model1 和 Mode2 是角度预测模式,那么第一个 MPM 是 Model1,第二和第三个 MPM 是与 Model1 相邻的两个角度预测模式。

R _{0,0}	R _{1,0}	R _{2,0}	...	R _{N,0}	R _{N+1,0}	R _{N+2,0}	...	R _{2N,0}
R _{0,1}	P _{1,1}	P _{2,1}	...	P _{N,1}				
R _{0,2}	P _{1,2}	...		...				
...	...							
R _{0,N}	P _{1,N}	...		P _{N,N}				
R _{0,N+1}								
R _{0,N+2}								
...								
R _{0,2N}								

图 2 帧内预测模板

1.3 算法功能优化

算法优化思路:将 16 个 8×8 块 pipeline 起来同时进行处理,方便外部块的处理,可以大大提升编码的速率。主要代码的优化位置在于(1)预测数组空间的大小调整;(2)读写数据标志信号的初始值调整;(3)预测模块内外循环判断条件的调整;(4)参考像素滤波、预测、计算残差、输出等四个步骤的执行判断条件的调整。预测模块优化后关键步骤循环图如图 3 所示。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output			
	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output		
		Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output		Filter	Filter pred	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	
			Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output	Filter	calBest	Filter pred calBest	Filter pred calBest Output	Filter	Filter pred	Filter pred calBest	Filter pred calBest Output

图 3 预测模块优化后关键步骤循环图

1.4 优化时候的问题分析

(1)遇到的问题

在优化的过程中遇到了几个关键的问题：
(1)没有修改代码输出部分代码时，初次优化 pred 只输入了一个 8×8 就不输入数据了，原因是第一组 8×8 的输出没有结束与下一组的 read_data 输入时序相冲突。只能通过缩短输出时间/延长下一组 read_data 输入的时间来解决。如图，可以看到，read_data 只输入了一轮数据(1000)，outputY8 只输出了 14 次，本应该输出 16 次，即没有完整地进行输出。修改之后 pred 模块的 Latency 变成了之前的 2 倍，预测是因为 VCT_GET16 引起的。

(2)解决思路

主要问题在于如何让 codeLumaQT8_pred 模块 outputY8、pred 输出与 read_data 输入时序不冲突，使得图片完整数据能够完整输入。

(3)解决方法

在原始代码的基础之上进行了多次尝试之后最终找到了优化方案。也就是前文中提到的优化思路；(1)预测数组空间的大小调整；(2)读写数据标志信号的初始值调整；(3)预测模块内外循环判断条件的调整；(4)参考像素滤波、预测、计算残差、输出等四个步骤的执行判断条件的调整，具体的代码调整参见代码。

2 验证结果

最终修改完成之后，在赛灵思的开发板 xc7vx485t-ffg1157-1 上进行综合测试，选择的时钟频率为 100 MHz，codeLumaQT8_pred 模块

计算一组 16×8×8 块需要的时间(1227 个时钟周期)相比于优化之前(1840 个时钟周期)减少了 33%的时间，如表 1 所示。表 2 是优化之前该模块处理一组 8×8 数据块需要的时间。表 3 是优化后该模块能达到的时钟频率。表 4 是整个 IP 的资源占用情况。

表 1 优化后处理一组 8×8 数据块需要的时间

Latency(cycles)	Latency(absolute)	Interval(cycles)
max/min	Max/min	Max/min
1227	12.270μs	1227

表 2 优化前处理一组 8×8 数据块需要的时间

Latency(cycles)	Latency(absolute)	Interval(cycles)
max/min	Max/min	Max/min
1840	18.400μs	1840

表 3 优化后工作能达到的时钟频率

Clock	Target	Estimated	Uncertainty
ap_clk	10.00ns	9.373ns	1.25ns

表 4 资源占用率

Name	BRAM	DSP	FF	LUT
Total	75	57	10795	30805
Utilization/%	3	2	1	10