

基于五级流水线的 HEVC DBF 模块硬件架构设计

沈高峰¹, 陈 松¹, 杨 晨¹, 杨灿美²

(1 中国科学技术大学 电子科学与技术系, 安徽 合肥 230027;

2 中国科学技术大学 先进技术研究院, 安徽 合肥 230027)

摘 要: DBF (Deblocking Filter) 是 HEVC 解码器中复杂度较高的一个部分, 它的处理速度直接影响到整个视频解码器的性能. 提出了一种基于五级流水线操作的高性能 DBF 模块硬件架构, 为了消除流水线操作中数据依赖的影响, 选取了 16×8 像素单元作为基本处理单元, 占用很少的内部存储器资源. 硬件架构使用 Verilog 硬件描述语言实现, 并通过 Xilinx-ISE 工具进行了电路综合. 结果表明, 此电路架构能够在 48.1 MHz 的工作频率下, 达到 $4\,096 \times 2\,048$ 分辨率 60 fps 高清视频的实时解码要求.

关键词: 视频编码; HEVC; 环路滤波; 去方块滤波; 硬件架构设计

中图分类号: TN 919. 81

文献标识码: A

文章编号: 1000-7180(2016)10-0001-06

Five-Stage Pipeline Hardware Architecture Design for Deblocking Filter in HEVC

SHEN Gao-feng¹, CHEN Song¹, YANG Chen¹, YANG Can-mei²

(1 Department of Electronic Science and Technology, University of Science and Technology of China, Hefei 230027, China; 2 Institute of Advanced Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: Deblocking Filter (DBF) is one of the components which have high computational complexity in HEVC decoder. Its processing speed directly affects the whole performance of video decoders. This paper proposes a high performance five-stage-pipeline-based hardware Architecture of DBF. In order to eliminate the influence of data dependency in the pipeline operation, a 16×8 pixels unit is chosen as basic processing unit. This unit occupies very little internal storage resources. The proposed hardware Architecture is implemented in Verilog-HDL. The implementation is synthesized by Xilinx ISE tool. The results show that the proposed architecture can meet requirement of real-time decoding $4\,096 \times 2\,048 @ 60\text{fps}$ high definition video under an operating frequency of 48.1 MHz.

Key words: video coding; HEVC; in-loop filtering; deblocking filtering; hardware architecture design

1 引言

HEVC 是国际标准化组织和国际电工技术委员会于 2013 年 4 月份正式发布的最新一代视频编码标准. 在相同视频清晰度的情况下, HEVC 比上一代视频编码标准 H. 264 减少了近 50% 的比特率^[1].

HEVC 标准中采用了与 H. 264 类似的去方块

滤波(DBF)方法^[2]. 为了降低算法复杂度, 新标准中的 DBF 只对 PU 和 TU 中不小于 8×8 模块的边界进行滤波处理. 尽管如此, 由新标准中编码单元尺寸的多变性引起 DBF 复杂的滤波操作仍然是编解码过程中的一个重要部分^[3], 也是影响整个编解码器性能的主要因素之一.

在视频编码技术中, 去方块滤波一直都是一个重要的研究方向. 文献[4]提出了一种以 16×16 CU

为基本处理单元的方法,通过增加数据通路并行处理的方式来提高处理速度.文献[5]提出了一种以包含4条子边界的 8×8 块作为基本处理单元的方法.上述文献的方案中都没有将DBF操作划分成流水线,导致关键路径太长,限制了电路工作频率.

为了解决上述问题,本文提出了一种基于五级流水线的硬件架构方案,合理划分滤波操作,减小了关键路径长度.并以 16×8 单元作为基本处理单元,不仅能满足流水线操作中数据依赖的要求,而且降低了内部存储器资源的消耗.

2 HEVC DBF 算法与分析

在HEVC标准中,DBF滤波算法主要分为两个步骤:滤波决策和滤波操作.

2.1 滤波决策

如图1(a)所示,图片左边为包含 64×64 个像素点的CTU单元,其中最小的虚线块包含 4×4 个像素点,是DBF处理中划分的最小单元. CTU中所有实线反应的是某种情况下的全部滤波边界(包含垂直边界和水平边界两种),DBF模块需要对所有的边界进行去方块滤波的判断和处理.滤波算法处理的最小单元是一个 8×4 的块单元,如图中深色部分所示.在这个处理单元中包含的边界被称为子边界.在图1(b)表示出包含一个子边界的 8×4 像素单元的内部情况.

图1(b)中总共有四行像素(line₀~line₃),每一行包含8个像素点,单元块中间的竖线就可以被看成是一条垂直方向的子边界.子边界两侧各有一个 4×4 单元块,左边的块被称为P块,像素编号为 $p_{00} \sim p_{33}$.右边的块被称为Q块,像素编号为 $q_{00} \sim q_{33}$.DBF实际就是由line₀和line₃两行的像素判断P块和Q块的像素分布情况,并根据判断结果对两个单元块中的部分像素做适当的修改.其中滤波决策分为三个步骤:

第一步,获取当前子边界的边界强度(Boundary Strength, BS);BS值主要是由P和Q块的预测与变换编码相关参数决定,有效取值为0、1和2.

第二步,滤波开关决策;根据子边界两侧的P块和Q块内各像素间的变化程度,判断此区域的图像纹理特性,再由纹理特性确定是否对当前子边界进行滤波处理.计算如下^[6]:

$$dp_0 = |p_{20} - 2 \cdot p_{10} + p_{00}| \quad (1)$$

$$dp_3 = |p_{23} - 2 \cdot p_{13} + p_{03}| \quad (2)$$

$$dq_0 = |q_{20} - 2 \cdot q_{10} + q_{00}| \quad (3)$$

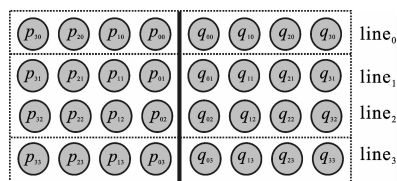
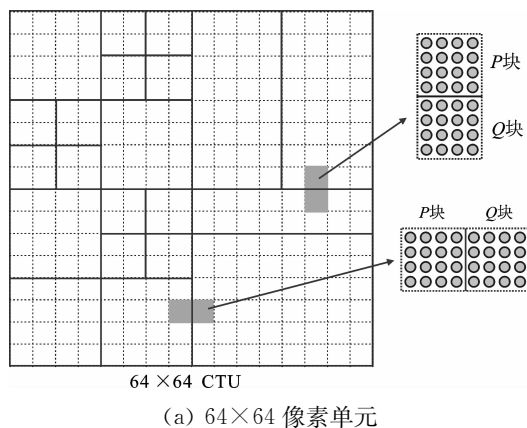


图1 像素单元示意图

$$dq_3 = |q_{23} - 2 \cdot q_{13} + q_{03}| \quad (4)$$

对上述四个公式求和,得到此区域图像纹理度表征变量 d :

$$d = dp_0 + dp_3 + dq_0 + dq_3 \quad (5)$$

如果纹理度 $d < \beta$,那么打开边界的滤波开关(其中阈值 β 是由像素的量化参数QP的值决定的).

第三步,滤波模式选择;滤波模式有两种:强滤波和弱滤波.滤波模式选择的相关公式如下:

$$2 \cdot (dp_0 + dq_0) < (\beta >> 2) \quad (6)$$

$$2 \cdot (dp_3 + dq_3) < (\beta >> 2) \quad (7)$$

$$|p_{30} - p_{00}| + |q_{00} - q_{30}| < (\beta >> 3) \quad (8)$$

$$|p_{33} - p_{03}| + |q_{03} - q_{33}| < (\beta >> 3) \quad (9)$$

$$|p_{00} - q_{00}| < (5 \cdot tc + 1) >> 1 \quad (10)$$

$$|p_{03} - q_{03}| < (5 \cdot tc + 1) >> 1 \quad (11)$$

公式中,阈值 tc 也是由量化参数QP决定的.只有当式(6)~(11)同时都满足的情况下,才会选择强滤波模式,否则为弱滤波模式.

在弱滤波模式下,滤波处理的像素点数量存在不同情况.子边界处的两列像素 $p_{00} \sim p_{03}$ 和 $q_{00} \sim q_{03}$ 肯定会被处理,而另外的两列像素则需要根据当前像素变化情况做更细致的判断.判断公式如下:

$$dp_0 + dp_3 < (\beta + (\beta >> 1)) >> 3 \quad (12)$$

$$dq_0 + dq_3 < (\beta + (\beta >> 1)) >> 3 \quad (13)$$

只有当式(12)和式(13)满足时,列像素 $p_{10} \sim p_{13}$ 和 $q_{10} \sim q_{13}$ 才会被处理.

2.2 滤波操作

滤波操作需要对 8×4 像素单元中的四行像素

(line₀~line₃)进行适度的修正. 根据滤波决策得到的结果,可以将滤波操作分为三种情况:亮度块的强滤波、亮度块的弱滤波和色度块的滤波.

第一种,亮度块的强滤波.

强滤波操作是对子边界两侧的六列像素进行较大幅度的修正. 对于每一行像素的修正操作是相同的,计算公式如下:

$$p_0' = \text{clip}_3(p_0 - 2 \cdot tc, p_0 + 2 \cdot tc, \varphi_0) \quad (14)$$

$$p_1' = \text{clip}_3(p_1 - 2 \cdot tc, p_1 + 2 \cdot tc, \varphi_1) \quad (15)$$

$$p_2' = \text{clip}_3(p_2 - 2 \cdot tc, p_2 + 2 \cdot tc, \varphi_2) \quad (16)$$

$$q_0' = \text{clip}_3(q_0 - 2 \cdot tc, q_0 + 2 \cdot tc, \gamma_0) \quad (17)$$

$$q_1' = \text{clip}_3(q_1 - 2 \cdot tc, q_1 + 2 \cdot tc, \gamma_1) \quad (18)$$

$$q_2' = \text{clip}_3(q_2 - 2 \cdot tc, q_2 + 2 \cdot tc, \gamma_2) \quad (19)$$

其中,

$$\varphi_0 = (p_2 + 2 \cdot p_1 + 2 \cdot p_0 + 2 \cdot q_0 + q_1 + 4) >> 3 \quad (20)$$

$$\varphi_1 = (p_2 + p_1 + p_0 + q_0 + 2) >> 2 \quad (21)$$

$$\varphi_2 = (2 \cdot p_3 + 3 \cdot p_2 + p_1 + p_0 + q_0 + 4) >> 3 \quad (22)$$

$$\gamma_0 = (p_1 + 2 \cdot p_0 + 2 \cdot q_0 + 2 \cdot q_1 + q_2 + 4) >> 3 \quad (23)$$

$$\gamma_1 = (p_0 + q_0 + q_1 + q_2 + 2) >> 2 \quad (24)$$

$$\gamma_2 = (p_0 + q_0 + q_1 + 3 \cdot q_2 + 2 \cdot q_3 + 4) >> 3 \quad (25)$$

式中的函数 $\text{clip}_3(x, y, z)$ 是对 z 进行限幅操作, z 的取值范围为 $[x, y]$. 上述公式是对每一行像素中的六个像素点进行修正,修正后的结果分别为 p_0' 、 p_1' 、 p_2' 、 q_0' 、 q_1' 和 q_2' .

第二种,亮度块的弱滤波.

相对于强滤波来说,弱滤波操作修正的像素范围和强度较小, 8×4 像素单元每一行像素中最多有四个像素点被修正. 每一行像素的修正操作也是相同的,计算公式如下:

$$\Delta = (9 \cdot (q_0 - p_0) - 3 \cdot (q_1 - p_1) + 8) >> 4 \quad (26)$$

如果 $|\Delta| < 10 \cdot tc$, 那么对此行像素执行式 (27) ~ (35) 的修正操作. 否则,该行像素不被修正.

$$\Delta' = \text{clip}_3(-tc, tc, \Delta) \quad (27)$$

$$p_0' = \text{clip}_1(p_0 + \Delta') \quad (28)$$

$$q_0' = \text{clip}_1(q_0 - \Delta') \quad (29)$$

$$\Delta p = (((p_2 + p_0 + 1) >> 1) - p_1 + \Delta) >> 1 \quad (30)$$

$$\Delta p' = \text{clip}_3(-(tc >> 1), (tc >> 1), \Delta p) \quad (31)$$

$$p_1' = \text{clip}_1(p_1 + \Delta p') \quad (32)$$

$$\Delta q = (((q_2 + q_0 + 1) >> 1) - q_1 - \Delta) >> 1 \quad (33)$$

$$\Delta q' = \text{clip}_3(-(tc >> 1), (tc >> 1), \Delta q) \quad (34)$$

$$q_1' = \text{clip}_1(q_1 + \Delta q') \quad (35)$$

经过上述公式修正后的像素分别为 p_0' 、 p_1' 、 q_0' 和 q_1' .

第三种,色度块的滤波.

色度块的滤波只会对子边界相邻的两列像素 $p_{00} \sim p_{03}$ 和 $q_{00} \sim q_{03}$ 进行修正. 每一行像素的修正操作也是相同的,计算公式如下:

$$\Delta = (((q_0 - p_0) << 2) + p_1 - q_1 + 4) >> 3 \quad (36)$$

$$\Delta' = \text{clip}_3(-tc, tc, \Delta) \quad (37)$$

$$p_0' = \text{clip}_1(p_0 + \Delta') \quad (38)$$

$$q_0' = \text{clip}_1(q_0 - \Delta') \quad (39)$$

公式中函数 clip_1 是限幅操作,如果色度的比特深度为 8,那么像素点的色度取值范围为 $[0, 255]$.

2.3 DBF 硬件设计挑战

分析上述 DBF 滤波算法可知,设计 DBF 硬件架构要面对两大挑战:

第一,访问存储器过于频繁;DBF 模块需要从外部存储器输入所有像素数据,经过滤波后,再输出给外部存储器. 如果是对 CTU 边界进行滤波,还需要参考相邻 CTU 中已被修正的数据. 因此,DBF 模块需要频繁访问外部存储器. 为了达到一定的处理速度,必定对存储器带宽要求很高,这会带来功耗和速度的损失.

第二,数据依赖限制;DBF 滤波顺序是先对所有纵向边界进行滤波再对所有横向边界滤波,而且在当前 CTU 进行滤波之前,它的顶端和左侧相邻的两个 CTU 必须已经处理完成. 这些都会导致 DBF 处理过程中的数据依赖,对硬件流水线设计带来困难. 为了提高处理速度,同时减少缓存资源,必须要选择合适的硬件滤波顺序和基本处理单元尺寸.

3 DBF 硬件架构设计

3.1 基本处理单元和滤波顺序

DBF 处理过程中,CTU 边界的滤波必须要参考相邻 CTU 的像素,一般都是采用把当前 CTU 的上端和左侧边界与内部边界一起处理的方法. 为了减少滤波过程中对外部存储器的访问次数,需要在

DBF 模块内部缓存一部分像素数据,等到与此部分像素相关的所有边界都滤波过后,再统一输出给外部存储器。

文献[7]中 DBF 模块的内部存储器缓存 68×64 个亮度像素,总共占用 53.2 k SRAM 资源.文献[4]是以 16×16 CU 作为 DBF 模块处理单元.为了尽量减少模块的存储器资源,同时兼顾数据依赖对模块流水线的影响,本文选择一种包含 16×8 个像素的单元块作为硬件架构的基本处理单元.如图 2 所示,每个处理单元(图中深色部分)包含 8 个 4×4 最小单元.每个 16×8 处理单元中总共包含 8 条子边界($E_0 \sim E_7$),需要 8 个周期处理完成.考虑到算法的边界滤波顺序要求和流水线数据依赖限制,本文采用的边界滤波顺序如图 2 中所示.前 4 个周期按顺序处理四条纵向边界: $E_0 \sim E_3$,后 4 个周期按顺序处理四条横向边界 $E_4 \sim E_7$.这种滤波顺序不仅能够满足标准中算法对滤波顺序的要求,而且能充分发挥流水线的作用,避免产生等待周期。

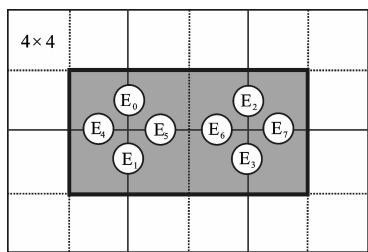


图 2 16×8 处理单元

3.2 硬件架构

本文提出了一种基于五级流水线的 DBF 硬件架构.如图 3 所示,硬件架构中包括滤波单元、有限状态机(Finite State Machine, FSM)控制单元、内部缓存器、流水线寄存器和外部 CTU 存储器.其中,滤波单元主要执行 DBF 滤波算法,内部分成三级流水线: S_1 、 S_2 和 S_3 ;FSM 控制单元主要负责整个模块的调度,根据不同时刻的状态产生各种控制信号;内部缓存器包含两个缓存器,每个缓存器能够存储 4×4 个像素点;流水线寄存器包含三个独立的寄存器,每个寄存器暂存一个 8×4 单元的像素值;外部 CTU 存储器负责把一帧图像转化为包含当前 CTU 和相邻 CTU 的像素数据,每个外部 CTU 存储器能够存储 64×64 个像素点。

由于视频处理数据量较大,为了达到一定的处理吞吐速率,一般视频处理电路中都会设计较多并行单元.为了尽量减少 DBF 算法并行实现所占用的硬件资源,本文使用了许多硬件优化策略.比如,用

加法和移位操作来代替倍数乘法操作,在本文设计的电路中所有的倍数乘法都未使用乘法器.在实现滤波算法操作中还充分使用了数据共享方法,来减少算术运算的硬件资源。

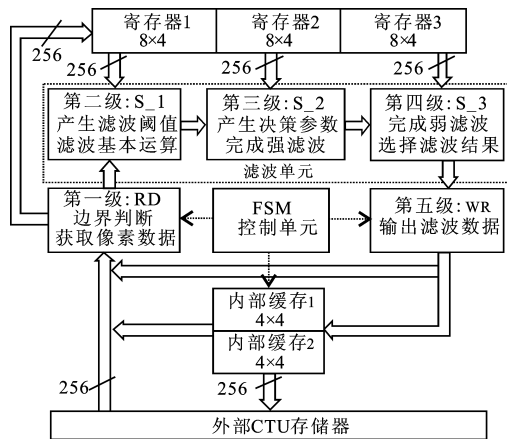


图 3 DBF 模块硬件架构图

3.3 五级流水线设计

为了提高硬件工作频率,本文将每个子边界的滤波处理过程划分为五级流水线: RD 级, S_1 级, S_2 级, S_3 级和 WR 级.每级占用一个工作周期,每个周期从存储器(外部 CTU 存储器或内部存储器)中读入一个 8×4 单元的像素数据,经过流水线处理过后,再存入内部缓存或者外部 CTU 存储器中.流水线结构需要的存储器带宽为 256 比特,流水线每一级具体操作如下。

第一级: RD 级;主要完成对数据的读入操作,产生外部 CTU 存储器控制信号.选择下一个需要处理的 8×4 单元数据来源,有三种可能外部 CTU 数据、内部缓存数据和上一个单元 S_3 级的输出结果。

第二级: S_1 级,算法操作第一级主要负责计算滤波决策中的阈值变量 β 和 t_c ,同时计算出强弱滤波中的中间变量,如式(20)~(26)的结果。

第三级: S_2 级,算法操作第二级主要负责计算滤波决策参数 dE 、 dEp 和 dEq ,完成强滤波操作,并计算出弱滤波的中间变量。

第四级: S_3 级,算法操作第三级主要完成弱滤波操作,并根据滤波决策结果选择像素的最终修正结果.最后把修正后的 8×4 单元数据做纵向或横向转换。

第五级: WR 级;主要完成对最终像素的输出操作.选择内部缓存器或者 S_3 级的结果输出给外部 CTU 缓存器。

图 4 给出了五级流水线操作和数据流通的示意图. 图中上半部分表示每个时钟周期下的流水线操作, 图中给出了一个 16×8 处理单元的完整流水线操作. 在 $T_0 \sim T_7$ 的每个周期中, 滤波单元读入一个 8×4 块像素数据, 经过五级流水线操作后输出. 图中下半部分给出重要数据端口的数据流通示意图, 图中深色部分表示与当前 16×8 处理单元相关的数据, 其它数据为前后相邻单元色处理数据, 空白部分表示没有数据流通. “Input_0”、“Input_1”、“Output_0”和“Output_1”表示与外部 CTU 存储器相连接的模块输入输出端口, 每个周期读入或输出 256 比特像素数据. “Data_in_P”和“Data_in_Q”表示滤波单元中 P 块和 Q 块像素的输入端口, “Trans_P_out”和“Trans_Q_out”表示滤波完成后经过转换的 P 块和 Q 块像素的输出端口.

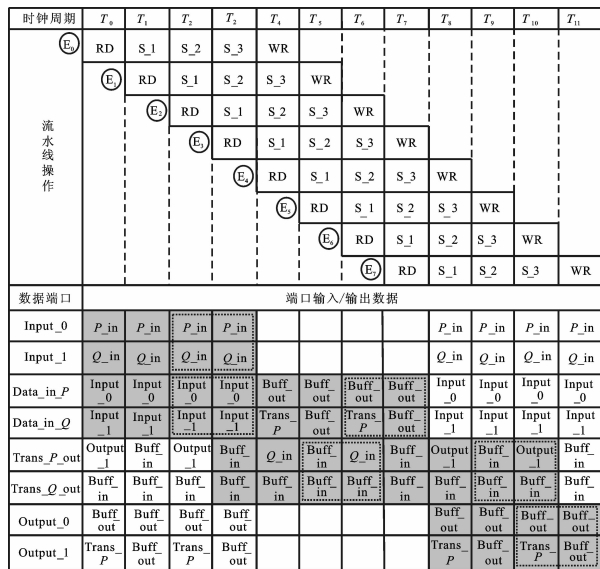


图 4 流水线操作与数据流通

3.4 存储器资源

从图 4 中可以看出, 在本文设计的硬件架构中存储器包括内部缓存器、流水线寄存器和外部 CTU 存储器三大部分. 内部缓存器主要负责各个边界滤波过程中相互依赖数据的暂存和输出数据转换的缓存, 其存储容量为 $2 \times 4 \times 4 \times 8 \text{ bits/pixel} = 256$ 比特. 流水线寄存器包括三级独立的寄存器, 作用是为流水线操作提供参考数据. 在时钟周期上升沿到达时, 每个寄存器的数据向后移动一级. 寄存器总存储容量为 $3 \times 8 \times 4 \times 8 \text{ bits/pixel} = 768$ 比特. 外部 CTU 存储器负责缓存每个 CTU 中所有边界相关的滤波数据, 这些数据来自解码器的帧级存储器, 按照光栅扫描顺序逐个扫描所有的 CTU. 总共的

CTU 存储器容量为 $64 \times 64 \times 8 \text{ bits/pixel} = 32\,768$ 比特.

3.5 状态机设计

为了有效控制架构中每个单元能够处在有序工作状态, 本文设计了一个有限状态机控制器. 如图 5 所示, 状态机包括 14 中不同状态: Idle、Fir_8×8_Skip、Sec_8×8_Skip、Sec_Skip_Wait、Sec_Skip_F4、Sec_Skip_F5 和 Filter_0~Filter_7. 其中, Idle 表示 reset 空闲状态, Fir_8×8_Skip 表示跳过不包含有效边界的第一个 8×8 块, Sec_8×8_Skip 表示跳过不包含有效边界的第二个 8×8 块, Sec_Skip_Wait、Sec_Skip_F4 和 Sec_Skip_F5 表示跳过第二个 8×8 块后完成第一个 8×8 块的后滤波操作, Filter_0~Filter_7 表示按顺序处理 $E_0 \sim E_7$ 边界. 状态机中特别设置的跳过两个 8×8 块的状态, 这种设计可以在较大尺寸 CU 的情况下, 快速跳过不需滤波的边界, 极大地提高滤波处理速度.

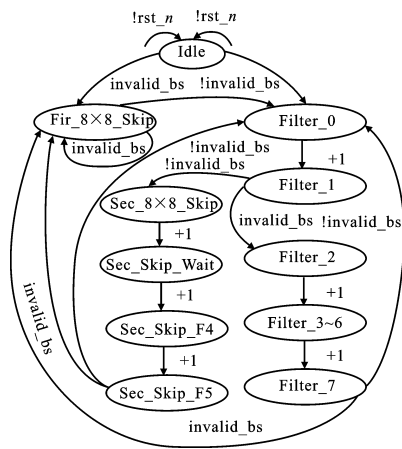


图 5 有限状态机

图 5 中, “rst_n”表示模块复位信号, 低电平有效. “invalid_bs”表示当前 8×8 单元中四条边界的 BS 值都等于零, DBF 模块会跳过此单元, 继续检测下一个单元的 BS 值. “+1”表示下一个时钟上升沿.

4 实验结果及分析

本文提出的 DBF 模块硬件架构使用 Verilog HDL 硬件描述语言实现, 并在 Xilinx ISE 中进行电路功能仿真, 仿真结果表明本文设计的电路实现了标准中 DBF 的算法要求. 在电路综合中选取的 FPGA 型号为 Xilinx Virtex-6 系列的 xc6vcx240t-2ff784 器件, 综合得到的最高工作频率为 213 MHz, 亮度分量的电路总共占用 4 203 个查找表

(Look-up Tables, LUTs)资源和 2 130 个寄存器 (Slice Registers)资源. 电路完成一个 CTU 单元的滤波操作, 需要的周期数为: $8 \times 8 \times 4 + 4 + 1 = 261$ (最坏情况下). 对于采样比为 4:2:0 的 $4\,096 \times 2\,048$ 分辨率视频来说, 本文设计的电路可以在 48.1 MHz 的工作频率下满足 60 fps 的视频解码要求. 表 1 给出了本文提出的电路架构与其它论文的比较结果, 其中等效吞吐率由最高工作频率和每个 CTU 消耗的周期数计算而来.

表 1 DBF 模块电路性能比较

对比项	本文	文献[4]	文献[8]	文献[9]
FPGA 型号	Virtex-6	—	Virtex-5	Virtex-6
SliceLUTs	4 230	5 236	8 479	1 398
SliceRegisters	2 130	1 547	3 457	441
最高工作频率(MHz)	213	108	80	140
CTU 尺寸	64×64	64×64	16×16	—
周期数/CTU	261	7 680	14	1 027
达到 4 096×2 048@60fps	48.1	—	—	124.8
要求的最小频率/MHz	3.35	0.06	1.46	0.56
等效吞吐率/(Gpixels/s)	3.35	0.06	1.46	0.56
支持标准	HEVC	HEVC	H.264/HEVC	HEVC

从表 1 中可以看出, 本文提出的电路架构的最高工作频率值最大, 处理每个 64×64 尺寸的 CTU 所占用的时钟周期数也最少, 等效吞吐率最高, 能够达到高速视频解码要求.

5 结束语

本文提出了一种基于五级流水线的高性能 HEVC DBF 模块的硬件架构. 选用 16×8 个像素点作为 DBF 基本处理单元, 并给出相应的边界滤波顺序. 根据这样的顺序依次对基本处理单元中的 8 条边界进行滤波处理, 不仅能够满足标准对纵向和横向边界滤波顺序的要求, 而且能巧妙的配合五级流水线, 使得整个滤波过程中不会出现数据等待现象. 实验结果显示, 本文提出的硬件架构具有较高的工作频率和数据吞吐率, 能够满足 $4\,096 \times 2\,048 @ 60\text{fps}$ 高清视频的实时解码要求.

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]. Circuits and Systems for Video Technology, IEEE Transactions on, 2012, 22(12): 1649-1668.

[2] Norkin A, Bjontegaard G, Fuldseth A, et al. HEVC

deblocking filter [J]. Circuits and Systems for Video Technology, IEEE Transactions on, 2012, 22(12): 1746-1754.

[3] Vanne J, Viitanen M, Hamalainen T D, et al. Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs [J]. Circuits and Systems for Video Technology, IEEE Transactions on, 2012, 22(12): 1885-1898.

[4] Ozcan E, Adibelli Y, Hamzaoglu I. A high performance deblocking filter hardware for high efficiency video coding [J]. Consumer Electronics, IEEE Transactions on, 2013, 59(3): 714-720.

[5] Jiayi Z H U, Dajiang Z. A high performance HEVC de-blocking filter and SAO architecture for UHDTV decoder [J]. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2013, 96(12): 2612-2622.

[6] ITU. ITU-T Recommendation H.265. High efficiency video coding [S]. Geneva, Switzerland: Telecommunication Standardization Sector, 2013.

[7] Cheng W, Fan Y, Lu Y H, et al. A high-throughput HEVC deblocking filter VLSI architecture for $8k \times 4k$ application [C]//Circuits and Systems (ISCAS), 2015 IEEE International Symposium on. Portugal, Lisbon, IEEE, 2015: 605-608.

[8] Srinivasarao B K N, Chakrabarti I, Ahmad M N. High-speed low-power very-large-scale integration architecture for dual-standard deblocking filter [J]. Circuits, Devices & Systems, IET, 2015, 9(5): 377-383.

[9] Diniz C M, Shafique M, Dalcin F V, et al. A deblocking filter hardware architecture for the high efficiency video coding standard [C]//Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. California, EDA Consortium, 2015: 1509-1514.

作者简介:

沈高峰 男, (1990-), 硕士研究生. 研究方向为数字集成电路设计. E-mail: gfsHEN@mail.ustc.edu.cn.

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.

杨晨 男, (1991-), 硕士研究生. 研究方向为数字集成电路设计.

杨灿美 男, (1965-), 博士, 研究员. 研究方向为通信数字基带系统集成电路设计、低功耗与高密度 SoC 架构设计.