

AVS 解码中反量化的 FPGA 实现

雷海军^{1,3}, 陈战夫², 陈意军³, 利驿飞¹, 李 立³

(1 深圳大学 计算机与软件学院, 广东 深圳 518060; 2 深圳大学 信息工程学院, 广东 深圳 518060;

3 湖南工程学院 电气与信息工程系, 湖南 湘潭 411101)

摘 要: 对 AVS 反量化算法模块进行分析, 结合 AVS 反量化的算法特点, 提出了使用较少存储空间来存储量化表数据的方法. 通过编写 VHDL 以实现优化后的 AVS 解码中反量化的算法, 利用 FPGA 开发工具 ISE7.1 和仿真工具 ModelSim SE 6.0, 完成了 AVS 反量化的 FPGA 设计与实现. 证明了算法实现的正确性.

关键词: FPGA; AVS 标准; 视频解码; 反量化

中图分类号: TN919.8

文献标识码: A

文章编号: 1000-7180(2010)07-0090-04

Implementation with FPGA Based on IQ for AVS Decoding

LEI Hai-jun^{1,3}, CHEN Zhan-fu², CHEN Yi-jun³, LI Yi-fei¹, LI Li³

(1 College of Computer and Software, Shenzhen University, Shenzhen 518060, China;

2 College of Information Engineering, Shenzhen University, Shenzhen 518060, China;

3 Department of Elect. and Information, Hunan Institute of Engineering, Xiangtan 411101, China)

Abstract: Analysing the module of the AVS inverse quantization, a method of using less ROM to save data in Quantization Table is proposed in this paper. Through the preparation of VHDL in order to implement the optimized inverse algorithm for AVS decoding, By means of the FPGA tool ISE7.1 and ModelSim SE 6.0, inverse quantization for AVS decoding based on FPGA can be implemented. The result of simulation is given in this paper which the Validity of the algorithm has been proved.

Key words: FPGA; AVS standard; video decoding; inverse quantization

1 引言

AVS 是中国具备自主知识产权的最新数字音视频编解码标准^[1]. 它包括系统、视频、音频 3 个主要部分和一致性等支撑部分组成. AVS 是以当前国际上最先进的视频编码标准 H. 264 框架为起点, 充分考虑实现复杂度, 自主制定适合既定应用的中国标准. AVS 中的大部分压缩技术经过优化, 计算复杂度、存储器需求和存储带宽较之 H. 264 都有所降低, 因此 AVS 的硬件实现是完全可行的, 并且实现复杂度更低, 硬件规模更小. 其编码效率和 AVC 相当, 是 MPEG-2 两倍以上^[2].

本文就 AVS 反量化模块的 FPGA 实现进行分析, 通过减少量化表的空间冗余度, 实现反量化算法

的设计. 文中采用 Xilinx 公司的 FPGA 开发工具—ISE7.1 进行设计, 从而实现 AVS 反量化模块的设计与综合, 并采用仿真工具 Modelsim6.0 进行仿真, 结果证明此反量化算法的正确性.

2 AVS 量化与反量化

2.1 AVS 量化

在图像或视频编解码器中, 量化包括两个部分: 编码器的正向量化器 FQ 和解码器的反向量化器 IQ^[3]. 在完成变化操作 DCT 变化后, 量化可以用来降低图像数据的精度, 去除不重要的数据值, 如去除接近 0 的变换系数. 编码器中的正向量化器将不重要的系数值映射为 0, 并保留一小部分重要的非零系数值. 量化步长是量化器的一个重要参数, 如果步

收稿日期: 2009-05-23; 修回日期: 2009-07-23

基金项目: 湖南省自然科学基金资助项目(05JJ40094); 广东省自然科学基金资助项目(5010493)

长大,量化值范围就小,压缩效率就高,但反量化值与原始值的近似就差;反之,步长小,反量化值能够更好地近似原始值,但压缩效率降低.

AVS 中采用的量化方法为

$$Z_{ij} = \text{round}(W_{ij} \times \text{PF}/Q_{\text{step}})$$
$$= \text{round}(W_{ij} \times \text{MF}/2^{\text{qbits}})$$
$$\frac{\text{MF}}{2^{\text{qbits}}} = \frac{\text{PF}}{Q_{\text{step}}}$$

式中, W_{ij} 是正向变换后的系数, Q_{step} 是量化步长, 每一个 Q_{step} 对应一个量化参数 QP , Z_{ij} 是量化后的系数, MF 为量化系数, PF 为缩放矩阵. $\text{round}()$ 为四舍五入运算.

2.2 AVS 反量化

AVS 的反量化是根据量化参数 QP 将二维量化系数矩阵 QuantCoeffMatrix 转换为二维变换系数矩阵 CoeffMatrix 的过程^[1]. 其中, 量化系数的取值范围是 $-2^{10} \sim (2^{10} - 1)$.

二维变换系数矩阵 CoeffMatrix 的元素 W_{ij} 由下式得到^[4]:

$$W_{ij} = (\text{QuantCoeffMatrix}[i, j] \times (\text{DequantTable}(QP) + 2^{\text{ShiftTable}(QP)-1}) >> \text{ShiftTable}(QP))$$
$$i, j = 0 \sim 7, QP \text{ 与 DequantTable 和 ShiftTable 的关系见表 1、表 2.}$$

表 1 QP<32 时与 DequantTable 和 ShiftTable 的关系

QP	0	1	2	3
DequantTable(QP)	32768	36061	38968	42495
ShiftTable(QP)	14	14	14	14
QP	4	5	6	7
DequantTable(QP)	46341	50535	55437	60424
ShiftTable(QP)	14	14	14	14
QP	8	9	10	11
DequantTable(QP)	32932	35734	38968	42495
ShiftTable(QP)	13	13	13	13
QP	12	13	14	15
DequantTable(QP)	46177	50535	55109	59933
ShiftTable(QP)	13	13	13	13
QP	16	17	18	19
DequantTable(QP)	65535	35734	38968	42577
ShiftTable(QP)	13	12	12	12
QP	20	21	22	23
DequantTable(QP)	46341	50617	55027	60097
ShiftTable(QP)	12	12	12	12
QP	24	25	26	27
DequantTable(QP)	32809	35734	38968	42454
ShiftTable(QP)	11	11	11	11
QP	28	29	30	31
DequantTable(QP)	46382	50576	55109	60056
ShiftTable(QP)	11	11	11	11

表 2 QP≥32 时与 DequantTable 和 ShiftTable 的关系

QP	32	33	34	35
DequantTable(QP)	65535	35734	38968	42495
ShiftTable(QP)	11	10	10	10
QP	36	37	38	39
DequantTable(QP)	46320	50515	55109	60076
ShiftTable(QP)	10	10	10	10
QP	40	41	42	43
DequantTable(QP)	65535	35744	38968	42495
ShiftTable(QP)	10	9	9	9
QP	44	45	46	47
DequantTable(QP)	46341	50535	55099	60087
ShiftTable(QP)	9	9	9	9
QP	48	49	50	51
DequantTable(QP)	65535	35734	38973	42500
ShiftTable(QP)	9	8	8	8
QP	52	53	54	55
DequantTable(QP)	46341	50535	55109	60097
ShiftTable(QP)	8	8	8	8
QP	56	57	58	59
DequantTable(QP)	32771	35734	38965	42497
ShiftTable(QP)	7	7	7	7
QP	60	61	62	63
DequantTable(QP)	46341	50535	55109	60099
ShiftTable(QP)	7	7	7	7

AVS 的反量化中,亮度量化参数和色度量化参数又有所不同.

亮度量化参数和色度量化参数的取值范围是 0~63. 如果当前块是亮度块,其量化参数 QP 等于 CurrentQP . 如果当前块是色度块,以 CurrentQP 为索引查表 3,得到色度块的 QP .

表 3 色度块的 QP 与 CurrentQP 的映射关系

CurrentQP	< 43	43	44	45	46	47
QP	CurrentQP	42	43	43	44	44
CurrentQP	48	49	50	51	52	53
QP	45	45	46	46	47	47
CurrentQP	54	55	56	57	58	59
QP	48	48	48	49	49	49
CurrentQP	60	61	62	63		
QP	50	50	50	51		

3 AVS 反量化的算法实现

AVS 反量化过程分析: 是二维量化系数阵 QuantCoeff 转化为二维变换系数阵 Coeff 的过程. 首先确定量化参数 QP , AVS 的量化参数取值为 0~63, QP 的值越大,精度越低. 再通过 AVS 中规定的 QP 与 DequantTable , ShiftTable 的关系表来求解二维变换系数阵元素. 量化系数的取值范围为 $-2^{10} \sim (2^{10} - 1)$, 变换矩阵系数的取值范围为 $-2^{11} \sim (2^{11} - 1)$. 所以反量化过程只是乘法、移位、加法的整数

运算,没有除法操作,易于实现.

AVS 反量化的算法实现:

二维量化系数阵 QuantCoeff 的数据存储在存储器中,输入的顺序为 A_{00} 、 A_{01} 、 A_{02} 、……、 A_{nm} ,按此顺序对数据进行反量化,反量化后的数据也存储在存储器中.

DequantTable(QP)和 ShiftTable(QP)的表中数据通过数组存储在存储器中,程序通过查表中的不同 QP 对应的 DequantTable 和 ShiftTable 的值.其中 DequantTable(QP)的值为 16 位二进制数,将其看成无符号数进行乘法运算,而 ShiftTable(QP)的值为 7~14 的整数,要转换成 4 位二进制数进行运算.

因为 ShiftTable(QP)值的取值范围为 7~14,所以 $2^{\text{ShiftTable(QP)}-1}$ 值的取值只有 8 个,分别为 2^6 、 2^7 、 2^8 、 2^9 、 2^{10} 、 2^{11} 、 2^{12} 、 2^{13} ;可以提前计算,其二进制表示为: 2^6 : 00000001000000, 2^7 : 00000010000000, 2^8 : 00000100000000, 2^9 : 00001000000000, 2^{10} : 00010000000000, 2^{11} : 00100000000000, 2^{12} : 01000000000000, 2^{13} : 10000000000000;本设计针对这个特点,把原本需要 64 个存储单元来存储的 $2^{\text{ShiftTable(QP)}-1}$ 值,现在只需要 8 个寄存器单元就可以实现了,从而减少了量化表中数据的空间冗余度,并且省去了 $2^{\text{ShiftTable(QP)}-1}$ 的查表时间.在程序中只需要针对 8 个不同 ShiftTable 的值用选择语句就能轻易实现.

4 AVS 反量化的 FPGA 设计与实现

根据反量化公式:

$$W_{ij} = (\text{QuantCoeffMatrix}[i,j])$$

$$(\text{DequantTable}(\text{QP}) + 2^{\text{ShiftTable}(\text{QP})-1}) >>$$

ShiftTable(QP) $i, j = 0 \sim 7$,

式中, QP 是量化参数,其取值范围是 0~63,用 4 位二进制数表示; QuantCoeffMatrix 是二维量化系数矩阵,其量化系数的取值范围是 $-2^{11} \sim 2^{11} - 1$,用 12 位二进制数表示. DequantTable 与 ShiftTable 是 8 行 8 列的矩阵,也可以看作是含有 64 个元素的数组,它们的最大值分别是 65535 和 14. QP 和 QuantCoeffMatrix 是输入, CoeffMatrix 是输出.

根据反量化公式和条件描绘出程序的流程图见图 1,然后利用 FPGA 开发工具 ISE7.1 编写程序,其端口描述的 VHDL 代码:

```
port( clk: in std_logic;
      QP_Map: in std_logic_vector(5 downto 0);
      Quant_Coeff: in std_logic_vector(10 downto 0);
      CoeffMatrix: out std_logic_vector(13 downto 0));
```

DequantTable(QP), ShiftTable(QP)用数组类型来定义存储的 VHDL 代码:

```
type Table_type1 is array(63 downto 0) of std_logic_vector(15 downto 0);
signal Table_quant: Table_type1 := (
  "1110101011000011", -- 60099
  "1101011101000101", -- 55109
  .....
  "1000000000000000" -- 32768
);
type Table_type2 is array(63 downto 0) of std_logic_vector(3 downto 0);
signal Table_shift: Table_type2 := (
  "0111", -- 7
  "0111", -- 7
  .....
  "1110" -- 14
);
```

QP 作为标号映射到每个 DequantTable(QP)、ShiftTable(QP)值,高地址存储后面的数,低地址存储前面的数.

$2^{\text{ShiftTable(QP)}-1}$ 的实现见以下代码:

```
case conv_integer(ShiftMap) is
  -- ShiftMap 为 ShiftTable(QP)的值
  when 14 => Exp_2_QP <= "10000000000000";
  when 13 => Exp_2_QP <= "01000000000000";
  when 12 => Exp_2_QP <= "00100000000000";
  when 11 => Exp_2_QP <= "00010000000000";
  when 10 => Exp_2_QP <= "00001000000000";
  when 9 => Exp_2_QP <= "00000100000000";
  when 8 => Exp_2_QP <= "00000010000000";
  when 7 => Exp_2_QP <= "00000001000000";
  when others => Exp_2_QP <= "00000000000000";
end case;
```

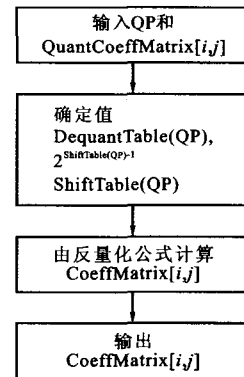


图 1 程序流程图

综合后生成的 RTL 原理图见图 2.

Modelsim6.0 对程序仿真的结果与分析:

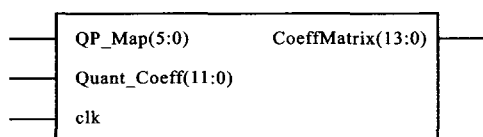


图2 综合后生成的RTL原理图

通过设定不同的量化参数QP值和量化系数quantcoeff值来观察仿真波形,输出波形反映的结果与理论计算结果一致。

如输入的量化参数qp为10,量化系数quantcoeff为16,此时通过查找量化表找到DequantTable(10)=38968,ShiftTable(10)=13.由反量化公式可计算出其反量化输出值:

$$W_{ij} = (\text{quantcoeff}(\text{DequantTable}(\text{QP}) + 2^{\text{ShiftTable}(\text{QP}) - 1}) >> \text{ShiftTable}(\text{QP})) = (16 \times 38968 + 2^{13-1}) >> 13 = 7$$

用二进制表示: $W_{ij} = "1001100"$. QP=10、quantcoeff=16时输出波形见图3。

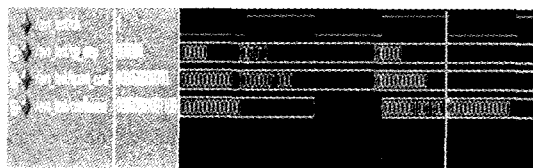


图3 qp=10、quantcoeff=16时输出波形

5 结束语

文中结合AVS的算法特点完成了其反量化的FPGA设计,在此过程中减少了量化表数据的存储空间,在某种程度上减小了算法的复杂度并简化了程序;该算法简单,程序简短,最后仿真的结果表明了该算法的有效性。

参考文献:

- [1] 林佳亮,王小力.基于AVS标准整数反变换的硬件设计研究[J].微电子学与计算机,2009,26(7):134-137.
- [2] 邓琼,杨立志,刘力政,等.一种基于FPGA的AVS编码加速器的设计[J].微电子学与计算机,2008,25(6):97-100.
- [3] 胡倩,张珂,虞露.AVS视频解码器的一种结构设计与硬件实现[J].浙江大学学报:工学版,2006,40(12):2139-2143.
- [4] 王晓慧.基于DSP的AVS反量化反变换的实现[J].Signal Process & System,2005(12):39-41.

作者简介:

雷海军 男,(1968-),博士,副研究员,硕士生导师.研究方向为图像处理、视频编解码、嵌入式系统、并行计算、IC设计等。

(上接第89页)

参考文献:

- [1] Kubota T, Nakatani I, Yoshimitsu T. Path planning for planetary rover based on traversability probability[C]// Proc IEEE Int. Conf. Adv. Robot., San Feliu de Guixols. Spain, Catalonia, 1995: 739-744.
- [2] 于红斌,李孝安.基于栅格法的机器人快速路径规划[J].微电子学与计算机,2005,22(6):98-100.
- [3] Iagnemma K, Genot F, Dubowsky S, et al. Rapid physics-based rough-terrain rover planning with sensor and control uncertainty[C]// 1999 IEEE International Conference on Robotics and Automation. Detroit, MI, 1999: 2286-2291.
- [4] Kamon I, Rivlin E, Rimon E. A new range-sensor based globally convergent navigation algorithm for mobile robots[C]// Proceedings IEEE Conference Robotics Automation. USA: Minneapolis, 1996.
- [5] Stentz A. Optimal and efficient path planning for partially-known environments[C]// Proceedings IEEE Conference Robotics Automation. USA: San Diego, 1994.
- [6] Laubach S L, Burdick J W. An autonomous sensor-based path-planner for planetary microrovers[C]// Proc. IEEE International Conference on Robotics and Automation. Detroit, USA, 1999.
- [7] Ishigami G, Nagatani K, Yoshida K. Path planning for planetary exploration rovers and Its evaluation based on wheel slip dynamics[C]// 2007 IEEE International Conference on Robotics and Automation (ICRA2007). Roma, Italy, 2007:2361-2366.
- [8] 周兰凤,洪炳镕.用基于知识的遗传算法实现移动机器人路径规划[J].电子学报,2006,34(5):911-914.

作者简介:

周兰凤 女,(1966-),博士,副教授.研究方向为智能机器人、自主导航。