一种高效的 H.264 CABAC 解码器的 VLSI 结构

石迎波，李云松，张建龙
(西安电子科技大学 综合业务网理论与关键技术国家重点实验室，陕西 西安 710071)

摘要：提出一种 H.264/AVC 中基于上下文的自适应二进制算术编码(CABAC)解码器的硬件设计方法，在采用并行结构的基础上，给出了一种高效的 VLSI 实现方案. 采用两级有限状态机结构控制宏块解码过程，并通过对残差系数存储器的定时清零解决了数据存储耗时的问题，大大降低了解码控制的复杂度，从而提高解码速度，达到每 1 至 2 个时钟解出 1 比特. 仿真结果表明，该方案能满足 H.264/AVC main profile CIF 30fps 实时解码的要求.

关键词：H.264/AVC；CABAC 解码器；大规模集成电路；有限状态机
中图分类号：TN919.81 **文献标识码**：A **文章编号**：1001-2400(2006)06-0844-05

An efficient VLSI architecture of the CABAC decoder in H.264

SHI Ying-bo, *LI Yun-song*, *ZHANG Jian-long*
(State Key Lab. of Integrated Service Networks, Xidian Univ., Xi'an 710071, China)

Abstract: A hardware implementation of the Context-based Adaptive Binary Arithmetic Coding (CABAC) decoder for H.264/AVC is presented. Based on the full use of the parallel architecture, an efficient solution for VLSI implementation is described. By developing the two-level finite state machines to control the decoding process and adopting the memory clear schedule to solve the problem of coefficients storage being time-consuming, the complexity of CABAC-decoder implementation is reduced, and the speed is increased to generate one bit within one or two cycles. Simulation results testify that our design can meet the needs of decoding the H.264/AVC main profile CIF bit stream at 30fps in real time.

Key Words: H.264/AVC; CABAC-decoder; VLSI; finite state machine

H.264/AVC[1,2]是由 ITU-T 的视频编码专家组(VCEG)和 ISO/IEC 的活动图像专家组(MPEG)联合制定的新视频编码标准，采用两种熵编码方法：基于上下文的可变长编码(CAVLC)和基于上下文的自适应二进制算术编码(CABAC). CABAC 采用高效的算术编码思想，同时充分考虑视频流相关统计特性，对不同的视频流能够自适应地调整，大大提高了编码效率；与 CAVLC 相比，采用 CABAC 编码可使平均比特率下降 9%～14%，而且在低码率情况下其编码效率优势更为明显[3]. 然而，CABAC 要用到大量的码表、上下文模型及中间变量等，比使用 CAVLC 要多大约 10%的运算量[4]，且需要更大的存储空间，这就增加了实现的复杂度.

在对 CABAC 算法分析的基础上，笔者提出一种 CABAC 解码器的 VLSI 实现方法. 该方法充分利用高效的并行结构，对 CABAC 解码器进行优化，并针对 CABAC 顺序解码的特点和对 H.264/AVC 句法表的分析，提出了一种采用两级有限状态机结构控制 CABAC 解码过程的方法. 另外，对解码过程中最为耗时和复杂的宏块残差系数，设计了一种新颖的残差系数存储器定时清零策略，降低了解码控制复杂度和解码所需的时钟周期数.

收稿日期：2005-12-25
基金项目：国家自然科学基金重点项目(60532060)；国家自然科学基金资助(60372043)
作者简介：石迎波(1981-)，男，西安电子科技大学博士研究生.

1　CABAC 解码器

在 H. 264/AVC 中,序列参数集、图像参数集及片头的句法元素使用通用可变长编码(UVLC)方法,其解码相对简单,而片层数据句法元素使用 CABAC 编码方式,其解码要复杂得多. 另外,片层数据码流占据了整个 H. 264 解码器码流的 90%以上,因此将解码器设计成由主控制器和协处理器组成的结构,CABAC 解码器由协处理器完成,而主控制器完成 UVLC 解码,并向 CABAC 解码器提供片层数据解码参数及码流.

CABAC 共建立了 701 个不同的上下文模型,每个模型分别用 pStateIdx 表示概率最小的符号的概率值,valMPS 表示概率最大的符号值,并定义了算术解码区间参数:区间长度 CodIRange 和区间下限 CodIOffset,用于 CABAC 解码.

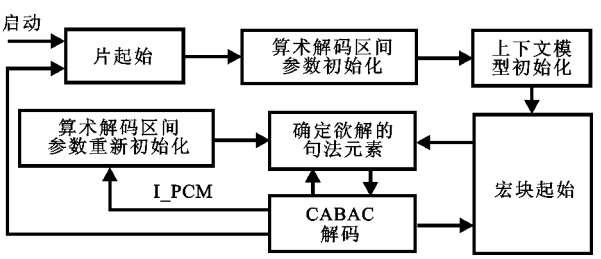

图 1　CABAC 解码流程

CABAC 解码流程如图 1 所示. 在 H. 264/AVC 中,片层是自适应解码的基本单元. 在开始解一个片层数据时,CABAC 首先依据该片的初始量化参数 QP、片类型以及初始化表格选择等句法元素值初始化上下文模型及算术解码区间参数,然后对句法元素进行解码[1]. 需要指出的是,当宏块类型为 I_PCM 时,在其亮度色度值解码完成后需对算术解码区间参数重新初始化[1].

CABAC 采用 3 种解码模式,分别为 DecodeRegular, DecodeBypass 和 DecodeTerminal[1]. 在 H. 264/AVC 中,句法元素 mvd, oeff_abs_level_minus1[i]的后缀及 coeff_sign_flag[i]采用 DecodeBypass 模式进行解码;而对 end_of_slice_flag 和 mb_type 的第一比特位(从零开始)采用 DecodeTerminal 解码;其余的句法元素采用 DecodeRegular 解码模式.

若采用 DecodeRegular 解码,要根据上下文信息确定欲解比特所采用的上下文模型,即其对应的 pStateIdx 和 valMPS. 首先通过查找表(rangeTabLPS)得到当前 CodIRange 所对应的 LPS(概率最小的符号)的区间长度 CodIRangeLPS,进而得到 MPS(概率最大的符号)的区间长度 CodIRangeMPS=CodIRange−CodIRangeLPS. 然后通过比较 CodIOffset 与 CodIRangeMPS 解出比特值为 valMPS 或!valMPS. 在解出比特值后,需对 pStateIdx(查找表 TransIdxMPS 或 TransIdxLPS)和 valMPS 进行更新以及对区间进行重归一化. DecodeTerminal 解码与此类似,是 DecodeRegular 模式的一种特殊情况(上下文模型固定为 276, CodIRangeLPS 固定为 2). 解码区间的重归一化过程,即通过若干次右移操作以保证解码区间长度 CodIRange 不小于 2^8,这是由 CABAC 编码策略决定的,如果区间长度小于 2^8,会因为编码器精度下降而导致编码效率降低.

采用 DecodeBypass 解码模式的比特位 0 或 1 基本等概率出现,不需要确定上下文模型和通过查找表对区间细分,较 DecodeRegular 更为简单.

由上述分析可知,在 CABAC 解码过程中必须首先确定当前欲解的句法元素,对该句法元素的每一比特解码时需要确定其解码模式及上下文模型等信息. 本文中所提出的两级有限状态机结构可以有效地实现上述解码过程中的两种功能.

2　高效 CABAC 解码器的 VLSI 结构

CABAC 解码器硬件实现结构框图如图 2 所示. 主要包括缓冲移位模块、解码模块、状态控制模块和解码输出模块. 缓冲移位模块提供欲解码码流,CABAC 解码模块完成比特解码及相关参数更新,状态控制模块负责一个片层各句法元素的解码流程控制,输出模块对解码结果进行有效的组织和存储.

相邻上块信息 SRAM 用于存储上一行宏块相邻位置的句法元素值(这里只存储那些用于各模型索引增量值计算的句法元素值),左块句法元素值用寄存器 Reg_left 存储. 在每解完一个宏块句法元素值后更新

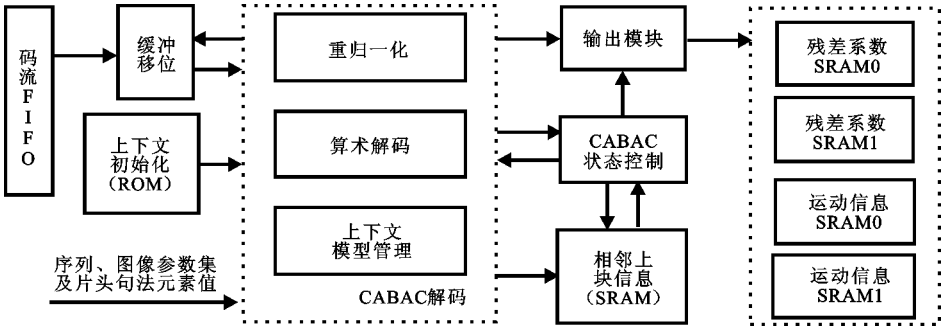

图 2 CABAC 解码器硬件实现结构框图

Reg_left,并更新相邻上块信息 SRAM 中相应位置的句法元素值. 通过即时动态更新,可避免存储两行宏块的句法元素值,进而节省存储空间,而同时又不影响模型索引增量值的计算.

下面分别叙述解码、状态控制和输出模块的 VLSI 结构.

2.1 CABAC 解码

CABAC 解码模块依据状态控制模块的指示进行句法元素比特解码. 依据其功能划分为如图 2 所示的 3 部分:算术解码单元、上下文模型管理单元和重归一化单元. 依次对其进行分析,并给出各单元的实现方法如下.

(1)算术解码实现需要 rangeTabLPS, TransIdxMPS 和 TransIdxLPS 3 个查找表. 通过对 TransIdxMPS 的分析,其除了输入为 62 或 63 时输出不变之外,其余的输出值均为输入值加 1,因此 TransIdxMPS 查找表可省略.

硬件实现中,rangeTabLPS 和 TransIdxLPS 查找表直接固化在芯片中,相当于放在一个片内 ROM 中. 与使用片外 ROM 实现该表的方法相比,本文中方法虽然占用了一些芯片资源,但可大大提高查找效率.

(2)上下文模型管理单元在片层起始时完成各上下文模型的初始化,解码时为算术解码单元提供欲解比特的概率模型信息. 为了提高解码速度,需要在一个时钟周期内完成上下文模型信息的更新,并获得下一个欲解比特需要的上下文模型信息. 该管理单元由 CABAC 状态控制模块进行控制,每个时钟读入算术解码单元更新后的上下文模型值,同时输出解码需要的上下文模型值. 当读入和输出的上下文模型相同时,算术解码单元将直接使用更新后的上下文模型值(即管理单元读入的模型值),这样,可避免一个时钟的等待.

(3)采用 DecodeRegular 和 DecodeTerminal 解出比特值后,均需重归一化. 经分析发现,在重归一化之前,可以根据区间细分之后的 CodIRange 的值和 CodIRange<0X0100 的判断条件确定 CodIRange 需要左移的次数. 确定了左移次数后,CodIRange 的重归一化就可通过一个左移函数一次性完成. 同时,这也是 CodIOffset 的左移次数及读入码流的比特长度,该长度将作为缓冲移位模块的输入值.

上述 3 个单元中,上下文模型管理的操作是受时钟控制的,其余的单元都是组合逻辑.

2.2 CABAC 状态控制

为了实现 CABAC 句法元素 SE 的解码,首先要确定当前欲解的句法元素,这主要是根据 H.264 的片层数据句法(该句法指明在码流中依次出现的句法元素及它们出现的条件)来实现的. 对一个宏块解码时,使用有限状态机控制句法元素间的跳转,如图 3 所示. 当开始宏块解码时,解码器从初始状态出发,根据宏块解码参数及相应的已解出 SE 值,经过一系列的状态转换依次解出宏块所包含的句法元素值,解码完成后重新回到初始状态.

对于特定的句法元素,每解出一个比特后,都需要与该句法元素已解出的比特一起与二进制化后的比特串比较,决定是否解出当前句法元素. 若解出,则进行句法元素的状态跳转. 否则,进行下一个比特解码,这时需要根据该句法元素以及已解码比特确定下一个比特的解码模式(DecodeRegular, DecodeBypass 或 DecodeTerminal)和上下文模型索引(当采用 DecodeRegular 模式时),同时将更新的上下文模型信息传至上下文模型管理单元. 针对各句法元素采用不同的二进制化方法和各比特位解码模式的不同,文中采用相应的比特有限状态机进行比特解码状态的跳转.

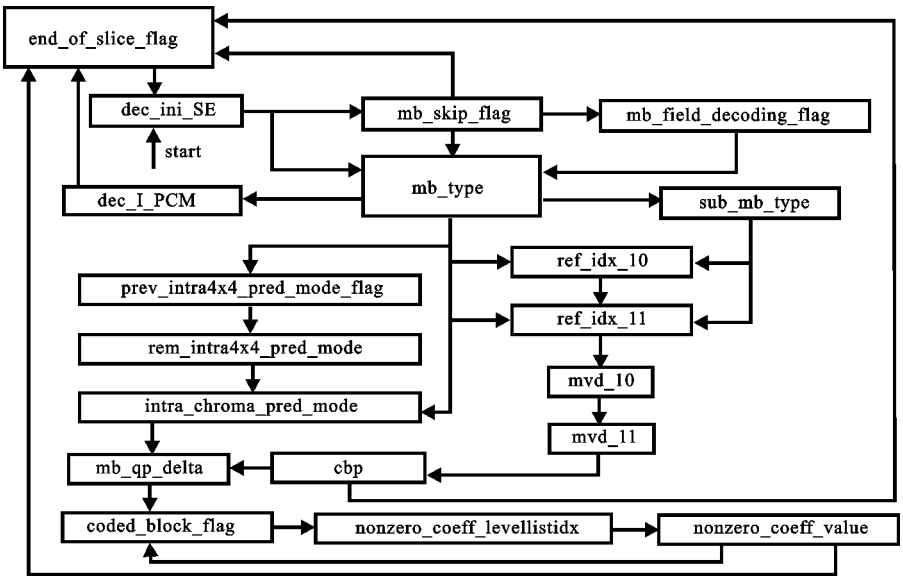

图 3　CABAC SE 解码状态转移图

作为例子,图 4 给出了 mb_type(I 片)的比特解码状态跳转. A(B)中 A 表示当前状态解码比特值,若为 X,则下一状态仅由当前状态决定;否则,下一状态由当前状态及解码比特值共同决定. B 为下一状态的上下文模型索引,dec_0 的模型索引值是根据相邻左块和上块的句法元素值决定的,B 为 276 则意味着采用 DecodeTerminal 解码模式. 可以看出,在状态重新跳转至初始状态时,也就解出了 mb_type 的值.

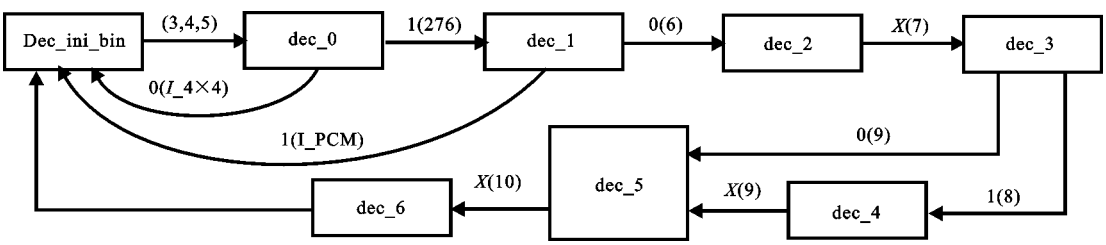

图 4　mb_type(I 片)比特解码状态转移图

SE 解码有限状态机跳转至某特定句法元素时,表明当前宏块存在该句法元素,即启动该 SE 比特解码状态机,开始 CABAC 比特解码. 在解码得到该 SE 值后,则结束该 SE 比特解码;然后根据已解出的句法元素值,SE 解码有限状态机跳转至下一欲解句法元素. 与比特解码状态类似,当 SE 解码状态跳转至初始状态时,意味着完成一个宏块的解码. 在 SE 解码状态跳转时,CABAC 解码模块会有一个时钟周期的空闲,而在 SE 比特解码时,每 1 至 2 个时钟解码得出 1 比特的二进制化码字.

2.3　输出模块

一个帧内 16×16 的编码宏块有 408 个残差系数,其余的有 392 个. 对运动矢量 MVD,最多各有 16 个 MVD_L0 和 MVD_L1. 针对其数据量大的特点,将其存入片内 SRAM 中. 为了降低延时,并和解码器其他模块(如反变换/反量化 IT/IQ、运动补偿等)能协同工作,将残差系数存储分为两个区,SRAM0 和 SRAM1,采用乒乓操作模式. 当输出模块往 SRAM0 中写入当前宏块的残差系数时,IT/IQ 模块从 SRAM1 中读取前一个宏块的残差系数值. 这样采用流水线技术,使得各模块可以同时运行. 运动信息 SRAM 采用与此相同的操作.

在对残差系数解码时,当 coded_block_flag 为 0 时表明当前残差块没有非零系数,否则首先解出残差块的非零系数位置 nonzero_coeff_levellistidx,同时可知其个数,然后按照反扫描顺序依次解出非零残差系数值 nonzero_coeff_value. 输出模块根据解出的非零系数位置,将其值写入 SRAM 中的相应地址,其余位置写入零.

由于非零系数值解码周期数以及两个非零系数间隔数的不确定性,可能导致解码等待. 例如,一个 4×4 块中有两个非零系数,但仍要消耗另外 14 个时钟周期将其余的零值写入 SRAM 中. 两个非零系数的位置及间隔不同,将使得解码控制极为复杂. 一种简单的处理方法是将 408 个地址各设置一个标志位 zero_flag,在将所有非零系数写入后,根据标志位将其余的零值写入 SRAM 的相应地址中. 该方法能使解码持续进行,但宏块解码完成需额外的 408 个时钟周期. 显然不利于实时解码.

因此,考虑到 IT/IQ 模块最多需要 408 个时钟周期就可将残差系数读完,可以在其读完残差系数后将该 SRAM 初始化为 0,使得下一个宏块解码时输出模块往该 SRAM(乒乓操作)中写入残差系数时,只需将那些非零系数值写入相应地址即可. 采用该方法能够显著地降低宏块解码时钟周期,进而提高解码速度. 而且,CABAC 解码模块和输出模块可以实现流水线结构,原来对残差系数的解码和其相应存储地址的计算及写入 SRAM 是在一个时钟内顺序执行的,运算量较大,而现在将其分为两部分,在一个时钟周期内同时执行,具有并行执行的特点,从而提高了效率. 更为重要的是,采用并行结构可使时钟频率提高.

3 试验结果与分析

对上述的 CABAC 解码器用 Verilog HDL 硬件设计语言进行设计,并在 Modelsim 6.0 环境下进行仿真,对 H.264/AVC 测试序列视频流解码测试,并与 JVT 校验模型 JM9.6[5] 的解码结果进行比较,表明该 CABAC 解码器可以满足实时解码的要求,且具有更好的性能.

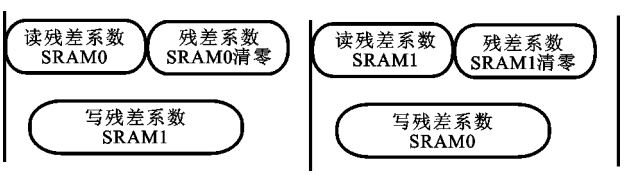

图 5 残差系数 SRAM 操作时序关系

图 5 为 CABAC 输出模块对残差系数 SRAM 乒乓操作的时序关系. 在一个宏块解码周期内,IT/IQ 模块读出 SRAM0 中的上一个宏块的残差系数后,输出模块对其清零. 同时,输出模块将解码模块解出的残差系数写入 SRAM1 中. 这两个操作是并行执行的.

对于上述设计,采用 ALTERA 公司的 QUARTUSⅡ5.0 开发软件,表 1 给出了具体的实现参数.

表 1 FPGA 芯片说明

目标器件	StratixⅡEP2S60F1020C5
编程语言	Verilog HDL
Total ALUTs	11 071
Total Memory bits	22 116
Total registers	3 303
系统时钟/MHz	60.23

表 2 宏块平均解码时钟数比较

平均时钟数	I 宏块	B 宏块	P 宏块
文[4]结构	1 661	328	576
本文中结构	923	200	340

对测试序列视频流解码验证,该设计能在一至两个时钟周期解出 1 比特码字. 一个宏块所需的解码时钟数主要由该宏块的码流比特数决定,宏块解码时钟统计见表 2. 可以看出,采用两级有限状态机结构及输出模块对残差系数 SRAM 定时清零策略使宏块解码速度得到了很大的提高.

4 结 束 语

提出一种高效的 CABAC 解码器硬件实现结构. 在对 H.264/AVC 句法表和 CABAC 算法深入分析的基础上,设计了一种有效的 SE 级和 SE 比特级有限状态机结构完成宏块解码的流程控制方法,并在输出模块对残差系数存储器定时清零,有效地解决了宏块残差系数解码耗时的问题,大大降低解码实现复杂度,继而提高了解码速度,同时对输入码流和上下文的有效管理也显示出较好的效果. 仿真结果表明,该方案能够满足 H.264/AVC main profile CIF 30fps 实时解码系统的要求.

(下转第 891 页)