基于 FPGA 的高速并行 Viterbi 译码器的设计与实现

童　琦，何洪路，吴明森
（中国科学院上海微系统与信息技术研究所，上海 200050）

摘　要：针对 319 卷积编码，提出一种 Viterbi 译码器的 FPGA 实现方案。该方案兼顾了资源消耗和译码效率，通过有效的时钟和存储介质复用，实现了高速并行的译码功能，并利用 Verilog 语言在 Xilinx ISE 6.2 中进行了建模仿真和综合实现。

关键词：Viterbi 译码　路径值　回溯

数字通信中，降低数据传输的误码率及提高通信质量是关键问题。为了避免通信过程中的突发错误，常常引入卷积码。

卷积码的译码方式中，Viterbi 译码是一种最大似然译码算法，与序列译码算法相比具有译码效率高、速度快及译码器实现结构简单的优点，是一种有效的前向纠错方法。

传统的 Viterbi 译码方法，译码需要存储的路径值等信息均随约束长度成指数增长[1]，且受到硬件水平的限制，不可能实现较高码率的译码。随着微电子技术的发展、可编程器件的广泛使用、开发工具的逐渐完善，以及在此基础上算法的研究改进，更高码率的译码实现成为可能。

本文将以（3，1，9）卷积编码为例，结合 Viterbi 译码的内部结构，均衡考虑资源消耗和译码效果，通过有效的时钟和存储介质复用，介绍一种可在 FPGA 上实现的高速并行的 Viterbi 译码实现方法，并利用 Verilog 语言在 Xilinx 公司的 ISE 6.2 上进行了建模仿真。

1 算法结构

1.1 译码器模块结构划分

本设计以（3，1，9）卷积码为例，其编码结构为 C0—$(557)_8$、C1—$(663)_8$、C2—$(711)_8$，共有 256 个状态点，即每输入 3 个 bit，需计算 512 个路径值，每个状态点计算出的两条路径取较小值存储并同时保存路径标识以供回溯解码。

本译码设计假定在解码前已有同步措施，输入信号不需要考虑相位问题，即每组 3 位信号有同步信号配合。

译码器划分为四个模块实现：加比选模块（Computer）、回溯模块（Trace）、存储模块（RAM1+RAM2）和时钟控制模块（CLK）。并行译码器结构如图 1 所示。

Computer 模块完成路径值累积，比较同一路径 1、0 分支累积，并抛弃一半较大路径的工作而将较小值送入 RAM1 存储，同时将路径标识存储到 RAM2 中；Trace 模块在满足条件后从 RAM2 中保存的数据回溯得出译码值。为实现高速并行功能，译码器除了根据算法简化存储值位数和缩小存储空间的同时缩短存储时间，还要对时钟复用，CLK 模块完成译码器内部的时钟管理工作。

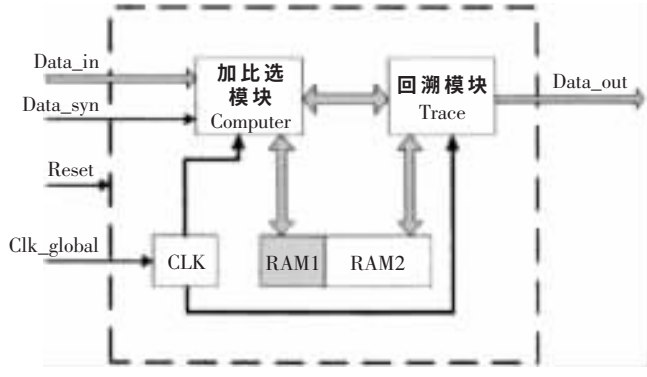

图 1　并行译码器结构图

1.2 关键算法分析

1.2.1 Computer 模块

Computer 的工作是 Viterbi 译码中最基本、最核心的工作，观察状态的变化可以发现它们都以一个蝶形运算为基本单元沿时间轴变化[3]，如图 2 所示。

图 2 中，X 轴为译码时间轴，Y 轴为状态轴。将状态值看成 i 的二进制编码，则从旧的状态 i 或 $i+128$ 到新的状态 $2i$ 和 $2i+1$ 有四条路径，分别有规律地对应 $C_0C_1C_2$、$\overline{C_0C_1C_2}$、$\overline{C_0C_1C_2}$、$C_0C_1C_2$。以此蝶形结构为最基本的运算单元构建 Computer 模块。Computer 主要由四个部分组成：8 种输出状态列表储存、简单状态控制、蝶形运算单元以及路径值存储控制。图 3 为 Computer 模块内部结构。蝶形运算路径量度值计算过程为：

$$m_f(2i)=\begin{cases} m_b(i)+b\_m_0 \\ \quad \text{if } m_b(i)+b\_m_0 \leqslant m_b(i+128)+b\_m_1 \\ m_b(i+128)+b\_m_1 \\ \quad \text{otherwise} \end{cases}$$

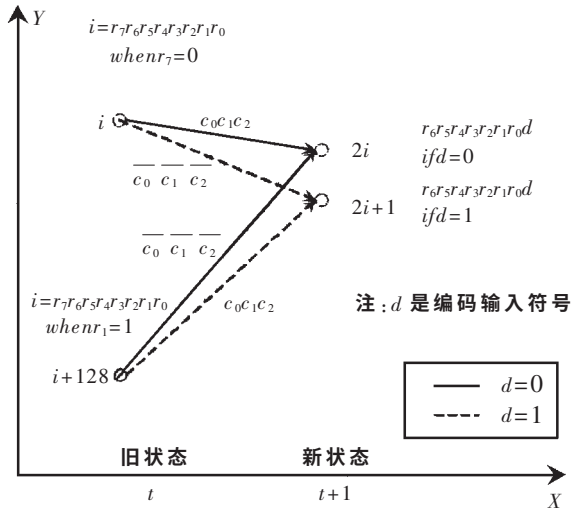

图 2 蝶形运算示意

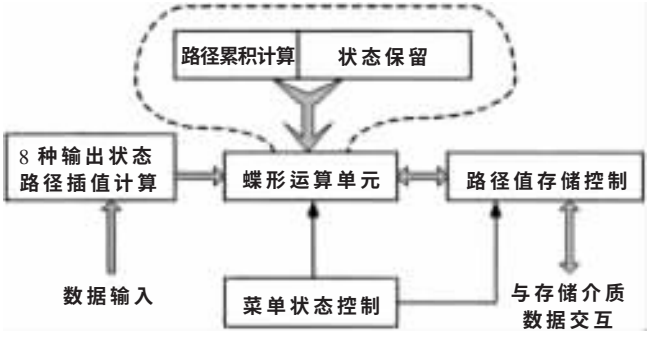

图 3 Computer 模块内部结构

$$m_f(2i+1)=\begin{cases} m_b(i)+b\_m_1 \\ \quad \text{if } m_b(i)+b\_m_1 \leqslant m_b(i+128)+b\_m_0 \\ m_b(i+128)+b\_m_0 \\ \quad \text{otherwise} \end{cases}$$

其中：$m_f(i)$表示新状态 i 的路径度量；$m_b(i)$表示旧状态 i 的路径度量；b_m_0 表示状态 r_7 输出与输入 d 数值相同时；b_m_1 表示状态 r_7 输出与输入 d 数值不相同时。

新的路径量度和旧的路径量度的保存可以通过 RAM1 中两块 ping-pong 结构的内存实现。

译码过程中路径量度的数值累加时，要防止溢出。本设计在每一步运算中将各个状态的路径量度减去前一步所有状态支路量度的最小值。

1.2.2 Trace 模块

从图 2 中可以看出路径信息已经隐含在状态信息中，如新状态 $2i$ 的最低位表明它由老状态 i 或 $i+128$ 经新的输入 0 位路径转移而来。因此知道新状态时只要保存这一个新输入位值为路径转移信息，就可以完成向前一个旧状态的回溯了[3]。图 4 为回溯示意图。

本设计在保存路径标识，即转移信息时，将 RAM2 中的地址与图 2 中状态 i 值对应，根据转移信息就能自动判断出前一时间的转移信息存储地址，从而逐步回溯到前 L 步，作出译码输出判决。回溯公式为：

address<=(path_bit==1)?((address>>1)+128):(address>>1)

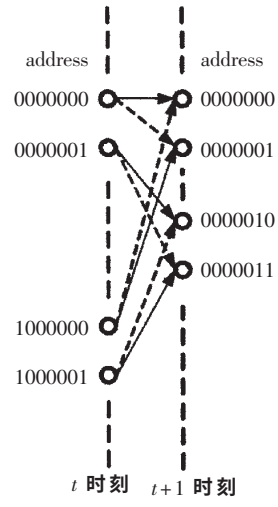

图 4 回溯示意图

本设计的原始数据块长度设定为 $L=192$。当 L 达到约束长度(64)后，每一个译码时钟(每输入 3 个位)都可以译出 1 个解码位，当所有输入位输入完毕后，共可译出 120 位信息位。

由于卷积编码前的数据最后 8 个尾位为 0，因此可以认为 Viterbi 译码器最后的状态归为 0 状态，所以译出 120 位信息位后的回溯应从状态 0 开始，回溯公式同上。不同的是之前的回溯要回溯 64 步才输出一个译码位，而此时每向前回溯一步，都输出 1 位译码位。最后的 64 步回溯输出信息位序号为 184~121，共 64 位信息位，且输出顺序为逆序(先 184 最后 121)。

1.2.3 时钟复用与存储模块

每个译码时钟 F_{vit} 期间内，本设计要完成 512 个度量值计算，即共有 128 个 M 型结构。采用硬件并行与时钟复用相结合方式实现。

硬件实现规定每译码时钟并行 8 个蝶形运算；蝶形运算每 2 个时钟运算一次，因而 8 个蝶形运算在 32 个时钟期间完成 8×16=128 次蝶形运算；剩下 4 个时钟供硬件状态准备和归位使用，并提供同步上的缓冲区时间，因此时钟复用为 36 倍译码时钟。

本设计使用乒乓结构实现存储度量值，因此需要 2 个 RamBank。由于每个蝶形运算中每一个时钟运算 1 次，每次运算输出 2 个度量值，因此每个 RamBank 需要 8×2=16 个数据口。定每个 RamBank 由 16 个 Ram 模块组成，每个 Ram 模块深度为 16，对应于 16 次计算。蝶形运算输出的度量值依次和 16 个 Ram 的数据口对接，每 2 个复用时钟将地址加 1。度量值存储区 RAM1 结构如图 5 所示。

回溯部分使用复用 72 倍的时钟，即 $F_{vit}\times 72$，这样在一个译码时钟之内，就可以完成 64 步回溯，剩余 8 个时钟用于硬件状态准备和归位使用，并提供同步上的缓冲区间。

路径存储既要考虑与度量值计算的接口，又要考虑回溯时的接口，因此采用双口 Ram。度量值计算部分每个复用时钟输出 8 个路径位，每个译码时钟间隔内共有 32 个写操作；回溯部分每个复用时钟只需读出 1 位路径信息。设计使用 32 个 Ram，每个 Ram 为 8bit×64depth，并带有一个片选信号 CS，8 个蝶形单元输出的路径位合并为 8 位总线，同时与 32 个 Ram 的数据口对接；根据 32 个复用时钟的序号控制 32 个Ram 的片选，每次选中

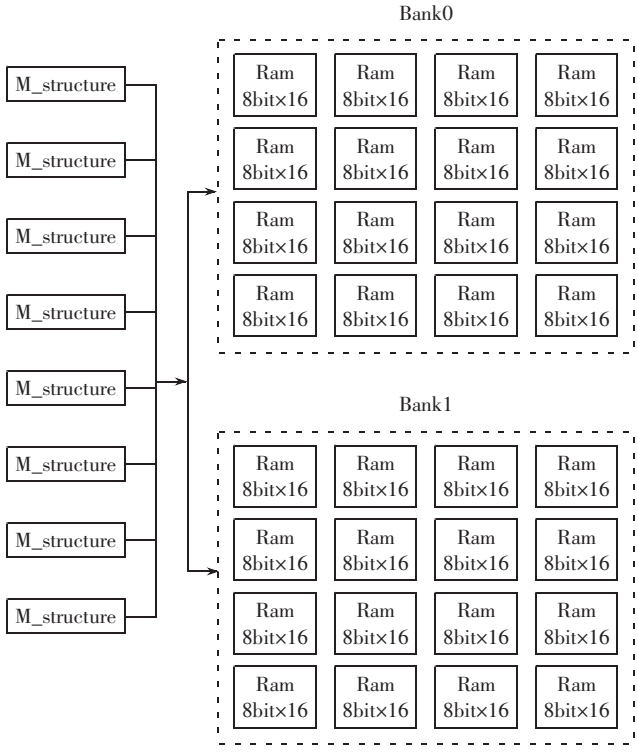

图 5 度量值存储 RAM1 结构示意

1 个；每个译码时钟将所有 Ram 地址加 1，对应于路径深度。路径标识 RAM2 结构如图 6 所示。

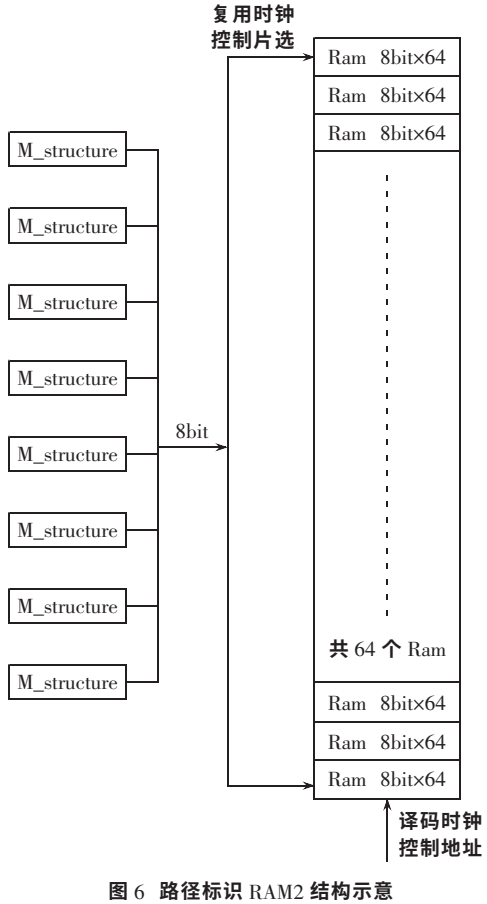

图 6 路径标识 RAM2 结构示意

2 基于 FPGA 的仿真实现

本设计采用 k 级时钟的码率，复用后的时钟不大于 72MHz，RAM 需 4kbit。Xilinx 公司的 VirtexII 系列芯片 XC2V3000 拥有足够的资源可以完成本设计。

本设计运用 Verilog 语言在 Xilinx ISE 6.2 环境下编写并采用 XST 综合实现，测试平台包括了卷积编码模块、本设计的解码模块以及一个简单的控制模块。仿真波形图的结果是卷积编码源码与维特比译码解码一致。

本设计的完整测试程序实际占用资源情况见表 1。以前用传统方法(由于资源问题未能在实际芯片上使用) 实现的资源占用情况如表 2 所示。通过对比可以看出，传统设计为求快速，主要运用芯片内的寄存器完成路径值和回溯标志的存储，资源消耗非常大(Slices 占用达 93%)，能够再用于完成其他基带功能的资源非常有限。而本设计的算法除 RAM 外，资源占用都不大，RAM 也仅占用 66%，剩余 34%可用于完成其他工作。

表 1 本设计测试程序占用资源列表

使用资源	使用数目	资源总数	占用比例(%)
Slices	1 532	14 336	10
Slice Flip Flops	1 252	28 672	4
4 input LUTs	2 836	28 672	9
BRAMs	64	96	66
GCLKs	1	16	6

表 2 传统设计测试程序占用资源列表

使用资源	使用数目	资源总数	占用比例(%)
Slices	1 3347	14 336	93
Slice Flip Flops	7 000	28 672	24
4 input LUTs	21 957	28 672	76
BRAMs	——	96	——
GCLKs	1	16	6

随着微电子技术的发展以及低功耗的需求，将数字系统集成到 ASIC 上势在必行。本文讨论了在 FPGA 上实现高速并行大约束度的 Viterbi 译码器的设计及仿真实现，为卷积编解码的 ASIC 实现提供了一种经实践证明切实可行的方案。

参考文献

[1] 曹志刚，钱亚生.现代通信原理.北京：清华大学出版社，1992.

[2] 胡爱群，旁 康，苏 杰.K=9 卷积码的 Viterbi 译码算法及其FPGA 实现.应用科学学报，1998，(6).

[3] FORNEY G D.The Viterbi algorithm.Proceeding of the IEEE，1973，61(3)：268~278.

[4] Virtex-II Platform FPGA Handbook.Xilinx 公司，2002.

(收稿日期：2006-08-02)