Viterbi 译码器回溯算法实现研究

王建新 于贵智
(南京理工大学电光学院 南京 210094)

摘 要: 该文介绍了两种 Viterbi 译码器回溯译码算法，通过对这两种算法硬件实现结构上的优化，给出了这两种算法的 FPGA 实现方法，比较了两种实现方法的优缺点。最后将其应用在实际的 Viterbi 译码器设计上，验证了算法实现的正确性。

关键词: Viterbi 译码；回溯算法；FPGA
中图分类号: TN911.22 **文献标识码:** A **文章编号:** 1009-5896(2007)02-0278-05

Study on Implementation of Traceback Algorithm in Viterbi Decoders

Wang Jian-xin Yu Gui-zhi
(*School of Electronic Engineering & Optoelectronic Technology*, *NUST*, *Nanjing* 210094, *China*)

Abstract: This paper discusses two traceback algorithms for Viterbi decoder. The realization methods for the traceback algorithms with FPGA are given through optimization for the hardware architecture. The comparison between the two realization methods is given. Finally, the two realization methods are applied to Viterbi decoder, and both simulation and hardware test show that the presented implementation methods are correct.

Key words: Viterbi decoding; Traceback algorithm; FPGA

1 引言

卷积编码是Elias等人在 1955 年提出的一种非常有前途的编码方法，其在通信系统中得到了极为广泛的应用，特别是在卫星通信系统中。其中约束长度K=7，码率为 1/2 和 1/3 的Odenwalder卷积编码已经成为商业卫星通信系统中的标准编码方法。1967 年Viterbi提出了卷积编码的一种概率译码算法——Viterbi算法，它是一种最大似然译码算法[1]。在码的约束度较小时，它比其它概率译码算法效率更高、速度更快，译码器也较简单。因而Viterbi算法提出以来，无论在理论上还是在实践上都得到了极其迅速的发展，并广泛地应用于各种数字通信系统。

对于 Viterbi 译码器的硬件实现，主要在于加比选模块(ACSU)和幸存路径存储模块(SMU)的设计与实现。ACSU 的功能在于计算进入各状态的候选路径的度量值，比较并选出各状态的幸存路径，而后存储各幸存路径度量值并送出状态转移标志矢量给 SMU。SMU 的主要功能是根据 ACSU 送来的状态转移标志矢量进行回溯译码操作，最后送出 Viterbi 译码器的译码结果。本文主要介绍 SMU 的设计与 FPGA 实现。

实现SMU中幸存路径的存储主要有两种方式：第一种是寄存器交换法，这种方式利用寄存器对各状态的幸存路径进行存储，译码时需对各寄存器进行更新、交换、写入操作，这种方式概念简单，译码延时短(可以为回溯深度值M)，但由于其在硬件实现上连线复杂度高，硬件资源耗费大(主要是因为寄存器位数较宽)，因此在实际实现Viterbi译码器时常不被采用。另外一种方式是Rader提出的回溯译码方式[2]，这种方式利用硬件存储器对状态转移标志矢量进行存储，有效地降低了硬件实现上的连线复杂度，因而在当前Viterbi译码器的设计中得到了广泛应用，但是其与寄存器交换法相比有更大的译码延时，一般是寄存器交换法延时的 2～4 倍。

关于不同的回溯译码算法Feygin在其文章中给出了全面具体的归纳与总结，其中提出了 4 种算法[3]。本文将简要介绍其中的两种算法，随后对这两种算法的硬件实现结构给出优化，并介绍以这两种算法为基础的幸存路径存储模块的FPGA实现，最后对这两种算法的实现给出比较分析。

2 回溯译码方式简介

如前所述，SMU是根据从ACSU送来的一系列状态转移标志矢量，搜索出一条译码路径，并以正常的顺序输出译码结果，而实际上搜索的过程是在网格图中回溯各个状态点转移的过程，如图 1 所示[4]。

根据 Viterbi 译码算法，随着译码的进行，经过一定的时间后，各个状态的幸存路径将开始结合，并最终结合于一条幸存路径，图 1 中展示出了 Viterbi 译码算法的这个特性，并根据这个特性，只要经过足够长的时间，根据任一状态向前回溯 M 时刻，会回到相同的状态点，M 为回溯深度，M 的大小将决定 SMU 模块中存储块 RAM 的大小，回溯深度

2005-05-30 收到，2006-03-20 改回

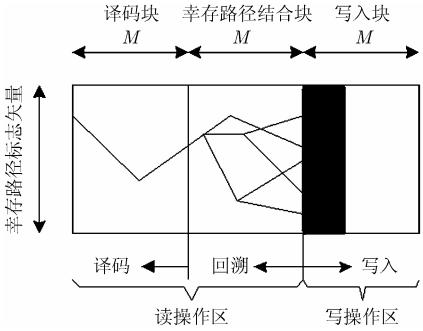

图 1　SMU 回溯网格图

的大小通常选定为约束长度 K 的 5～10 倍。另外，如图 1 所示，将 SMU 的存储器分为 3 个存储块，在不同的时刻分别进行不同的操作，下面介绍这 3 种操作：

(1) 写入新数据操作　随着 ACSU 模块不断产生新的状态转移标志矢量送给 SMU，SMU 首先需要将这些矢量保存，以进行后面的回溯译码操作。

(2) 回溯读操作　当 M(或更多)个状态转移标志矢量写入存储器后，SMU 模块需要启动回溯操作来回溯网格图中起始点的状态。在每次回溯操作中，得到一个位宽为 b(b=K-1)的指针向量，该指针向量指示本次的回溯结果指向状态 2^b，根据这个指针向量和当前的状态转移标志矢量，可以继续进行下一时刻的回溯操作。当进行了 M(或更多)的回溯读操作后，将最后的指针矢量保留，并启动译码读操作。

(3) 译码读操作　从读操作方法上来讲，译码读操作与回溯读操作是相同的，所不同的一点就是译码读操作要产生译码输出，作为 SMU 的输出结果。译码读操作开始的状态点由前面回溯操作所提供，同时随着译码读操作的进行，存储空间也随之释放，以供写入新数据。

3　两种回溯译码算法

3.1　k-pointer Even 算法

k-pointer Even算法需要k个读操作指针，$2k$(偶数)个存储块，每一个存储块深度为 $M/(k-1)$，写指针以从左至右的方向将新数据写入存储块中，而读指针以从右至左的方向将数据读出，并进行相应的回溯和译码操作。在每一时刻，一个写入新数据操作，一个译码读出操作和(k-1)个回溯读操作以并行的方式同时进行。该算法的译码延时为 $2kM/(k-1)$。其算法如图 2 所示，图中给出的是k值为 3 的情形。

3.2　one-pointer 算法

one-pointer算法的特殊之处在于只有一个读指针，在one-pointer算法下需要将读出操作加速，每进行一次写新数据操作应对应于k次读操作，这也就意味着仅仅需要k+1 块存储块，每块存储块的深度仍为 $M/(k-1)$。该算法的译码延时为 $(k+1)M/(k-1)$。其算法如图 3 所示[3]，仍选定k值为 3。

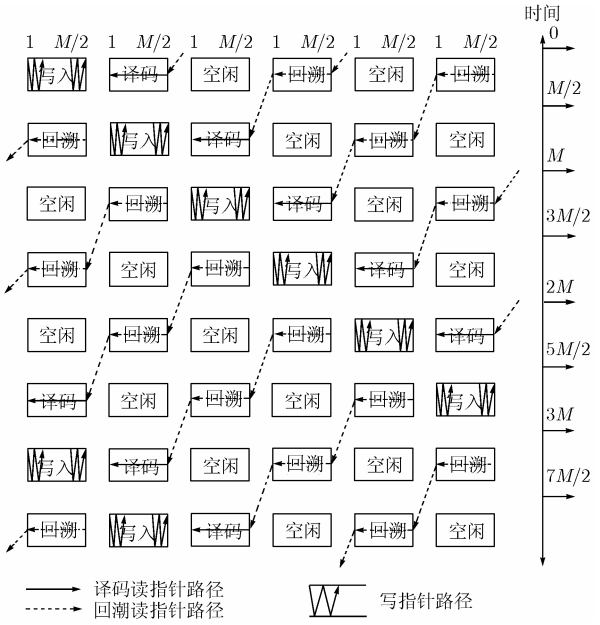

图 2　k-pointer Even 算法示意图(k=3)

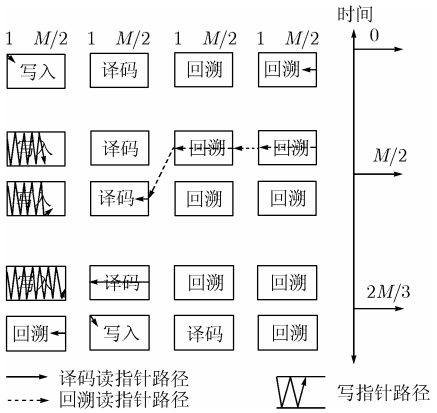

图 3　one-pointer 算法示意图(k=3)

4　两种回溯算法的 FPGA 实现

下面介绍上面两种回溯译码算法的FPGA实现，本文设计针对的是常用于卫星通信系统的(2，1，7)标准的Viterbi译码器。实现的硬件平台为Altera公司的Cyclone系列的FPGA：EP1C20。该器件具有 20 060 个逻辑单元和 294 912bit存储器。Cyclone器件提供的嵌入式存储资源M4K存储块，可以用于不同要求的存储模式，其中包括单端口随机存储器(Single-Port RAM)、简单的双端口随机存储器(Simple Dual-Port RAM)、真正的双端口随机存储器(True Dual-Port RAM)、移位存储器(Shift-Register)、只读存储器(ROM)和先进先出(FIFO)缓冲器。M4K存储块还具有一些先进的特性，例如字节使能操作、奇偶校验位检错和混合端口位宽等。另外，本文实现的软件平台为Altera公司的Quartus II 3.0 软件。

4.1　k-pointer Even 算法的 FPGA 实现

在 k-pointer Even 算法中需要占用 $2k$ 个深度为 $M/(k-1)$ 的存储块，译码延时为 $2kM/(k-1)$。因此对应于

不同的 k 值，译码延时的长短也不相同，k 值取的越大延时就越小，但是应该看到 k 值的增大，会使存储块的个数和读指针的个数线性增加，从而导致读地址的控制变得复杂，而译码延时的减小并不是线性地下降，下降的速度较慢，并且有下限值 $2M$。因此在实现中权衡实现复杂度和译码延时的关系，选定 k 值为 2，这样按前面的介绍，本设计译码器需要存储块个数为 4，译码延时为 $4M$，其中回溯深度 M 选定为 40，因此计算出译码延时为 160 bit。

首先介绍本算法实现的存储器组织方式，如前所述本算法实现需要 4 块位宽为 64 bit 的存储块，每块存储块的深度为 40。由于 Cyclone 器件的 M4K 块仅仅支持最大位宽为 32 的 RAM 实现，因此这种实现方案一个回溯模块共需要耗费 8 块 M4K 块。可以作如下改进，将 M4K 块配置为简单的双端口随机存储器(Simple Dual-Port RAM)的模式，由于这种模式下的存储器，读写口有单独的地址总线，可以进行单独读写寻址操作，因此可以将设计中需要的 4 块存储块两两合并，配置为一块 Simple Dual-Port RAM，这样就仅需要 4 块 M4K 块来实现存储器设计。并且还应保证任意时刻，在一块 Simple Dual-Port RAM 上的操作最多同时只能有一个写操作和一个读操作，显然这个要求是可以做到的，下面给出本设计的存储器组织示意图，如图 4 所示。

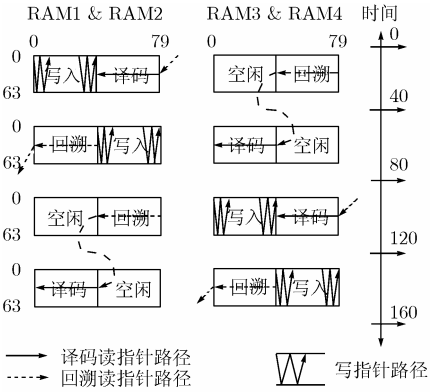

图 4 k-pointer Even 算法 SMU 存储器结构示意图(k=2)

这样设计的存储器组织结构，仅仅需要 4 块 M4K 块，而且可以注意到这种组织方式下，读写地址操作十分简单，因为两块存储器仅需要一套读写地址即可。

下面给出 k-pointer Even 算法的 SMU 模块实现的整体框图，如图 5 所示。

图 5 中的 P_Sign_En 是由 ACSU 模块送入的路径矢量使能信号，它标志送出的 P_Sign[63..0]信号即 64 位的状态转移标志矢量有效。

图 5 中的控制模块(Ctr module)提供回溯模块内部所有的控制信号，七位存储器的读地址 Rd_add[6..0]，写地址 Wr_add[6..0]，存储器写地址使能信号 Wr_En，控制 LIFO 模块的触发信号，控制回溯操作模块和译码操作模块初始化的信号 Ini_syn，还有控制连接器的选择信号 sel。需要说明的是控制模块还产生最后的译码输出使能信号

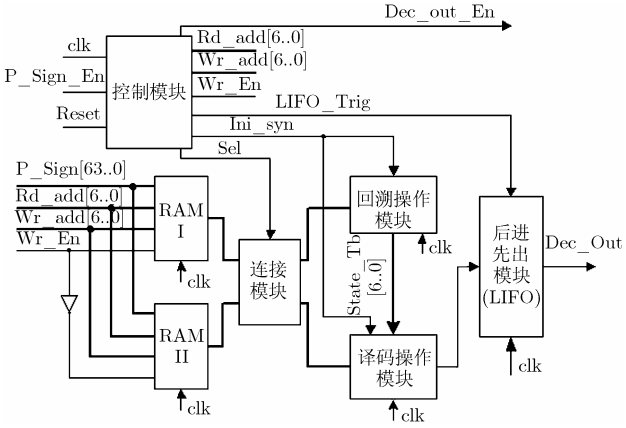

图 5 k-pointer Even 算法 SMU 实现结构示意图

Dec_Out_En，该信号是由 ACSU 模块提供的 P_Sign_En 使能信号直接延时 162 个周期而产生的，其中 160 个周期是前面提到的译码延时，而另加的两个周期延时是由于存储器读出延时一个周期，LIFO 模块中又延时一个周期，因此最后共延迟 162 个周期。对于 162 个周期的长延时，如果用触发器来实现，可想而知共需要 162 级触发器，相当耗费硬件资源，因此本文采用了计数器的方法分别实现上升沿的延时和下降沿的延时。

图 5 中连接模块(Con module)的作用是在 sel 信号的控制下将存储器读出的路径矢量值，送给相应的回溯操作模块和译码操作模块。

图 5 中回溯操作模块(TB module)根据送来的状态转移标志矢量和当前的状态值回溯得出前一状态值，设当前状态矢量为(s5 s4 s3 s2 s1 s0)，则回溯的上一状态为(s4 s3 s2 s1 s0 Sign)，其中的 Sign 是回溯操作模块中 64 选 1 数据选择器的输出结果，选择控制矢量信号为当前的 5 bit 状态矢量，另外回溯模块还在 Ini_syn 信号的控制下将每一阶段的回溯结果送给译码模块。

图 5 中的译码操作模块(Dec module)根据回溯操作模块送出的回溯结果作为其回溯译码操作的初始状态并开始回溯译码操作，其回溯操作与前面介绍的回溯操作模块中的方式相同，只是回溯的同时将 s5 作为译码输出信号送给后面的后进先出(LIFO)模块。

由于，回溯译码操作是以倒位序的顺序将译码结果送出的，因此还需要一个后进先出(LIFO)模块，将译码结果以正常的顺序输出。图 5 中的 LIFO 模块中的主要结构是两个移位寄存器：输入移位寄存器和输出移位寄存器。输入移位寄存器将译码操作模块送出的倒序译码结果移入输入移位寄存器，并在 LIFO_Trig 信号的控制下将移入结果送入输出移位寄存器，而后输出移位寄存器以正常的顺序将译码结果作为最后的 Viterbi 译码器输出结果送出。

为验证该算法实现的正确性，本文利用 Matlab 产生 100 个随机数，并产生测试向量生成 vec 仿真文件，利用 Quartus II 自带的仿真软件进行时序仿真后将仿真结果保存为 tbl 文

件格式，用 Matlab 将结果读回，最终验证了设计的正确性。

4.2 one-pointer 算法的 FPGA 实现

在 one-pointer 算法中需要占用 $k+1$ 个深度为 $M/(k-1)$ 的存储块，译码延时为 $2kM/(k-1)$，k 值取的越大延时就越小，并且有下限值 M，显而易见 one-pointer 算法相比较于 k-pointer Even 算法而言，在存储资源的消耗和译码延时的长度上都具有优势。本文根据实际情况，在设计中选用 k 值为 4，这意味 SMU 每进行一次写数据操作，需要同时完成 4 次读操作，因此需要的读时钟频率为写数据时钟(码率时钟)的 4 倍，可以将读时钟 4 分频得到写数据时钟(码率时钟)。在设计中将回溯深度改为 42，因此共需要 5 块位宽 64、深度 14 的存储块，并且可以算出译码延时为 70 bit。

首先设计 SMU 的存储器组织结构，在 one-pointer 算法中对于所有的存储器，在任意时刻仅有一个读操作和一个写操作在同时进行，因此可以考虑利用一块位宽 64、深度 70 的 Simple Dual-Port RAM 来实现存储器的设计，这样只需要两块 M4K 存储块就可以实现本文的存储器结构设计。下面给出本文设计的存储器组织示意图，如图 6 所示。

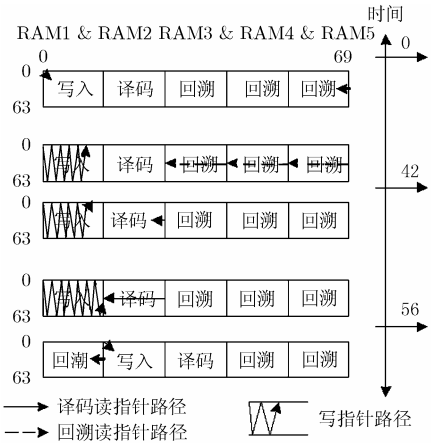

图 6　one-pointer 算法 SMU 存储器结构示意图(k=4)

从图中可以看到，写指针的方向始终为从左向右，地址递增，而读指针的方向始终从右向左，地址递减，因此在地址控制实现上比较简单。

下面给出 one-pointer 算法的 SMU 模块实现的整体框图，如图 7 所示。

图 7 中的控制模块(Ctr module)提供回溯模块内部所有的控制信号，七位存储器的读地址 Rd_add[6..0]，写地址 Wr_add[6..0]，控制 LIFO 模块的触发信号 LIFO_Trig 和使能信号 LIFO_En，控制回溯译码操作模块初始化的信号 Ini_syn，还有最终的译码输出使能信号 Dec_Out_En。需要说明的是 Dec_Out_En 信号是 P_Sign_En 信号延时 71 个写时钟周期得到的，实现方法与 k-pointer Even 算法中相同。

图 7 中回溯译码操作模块(TB&Dec module)将根据存储器中读出的路径矢量同时进行回溯和译码操作，回溯和译码的过程与 k-pointer Even 算法中的相同，只是其送出的译码数据并不都是需要的，因此利用 LIFO_En 信号来标志哪些信号是应该送入 LIFO 的输入移位寄存器的。

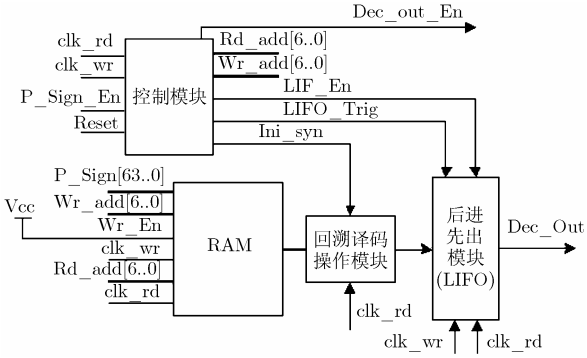

图 7　one-pointer 算法 SMU 实现结构示意图

图 7 中的后进先出(LIFO)模块，同样也是由输入移位寄存器和输出移位寄存器所构成的，只是由于回溯译码送出的数据速度是码率的 4 倍，因此由回溯译码操作模块送出的译码数据是突发性的，则输入移位寄存器的输入时钟是输出移位寄存器时钟的 4 倍，这样才能实现码率匹配，以正常码率的速度将译码数据以正常顺序连续送出。

为验证该算法实现的正确性，本文利用 Matlab 产生 100 个随机数，并产生测试向量生成 vec 仿真文件，利用 Quartus II 自带的仿真软件进行时序仿真后将仿真结果保存为 tbl 文件格式，用 Matlab 将结果读回，最终验证了设计的正确性。

目前已把上述算法成功地应用于研制的数字基带 MODEM 中。

5　结束语

上面介绍了两种常用的回溯译码算法的 FPGA 实现，并将它们应用到了实际的 Viterbi 译码器设计中，在译码器实际的应用中验证了设计的正确性。下面将这两种算法的实现进行比较。在存储资源上，one-pointer 算法消耗的存储器资源要明显少于 k-pointer Even 算法，在上面实际的两种实现方法中看到 one-pointer 算法消耗的存储器资源仅为 k-pointer Even 算法的一半。在译码延时方面，one-pointer 算法的也要明显优于 k-pointer Even 算法，从前面的实现上来看前者的译码延时只有后者的一半，这对于实时性要求较高的场合，这种性质是很具有吸引力的。从硬件实现的复杂度上来看，实现 one-pointer 算法需要一个高速的读时钟，这对于实现高速率的译码器来说是不利的，除此之外在其它的硬件实现上反而要比 k-pointer Even 算法简单。因此，可以得到如下结论：对于两种回溯算法的选择要根据实际情况具体分析，当实现的系统对实时性要求较高，而码率较低，可选择 one-pointer 算法来实现 SMU，例如语音通信系统；而对于码率要求较高，但对系统实时性要求不高的场合，可选用 k-pointer Even 算法来实现 SMU 的设计。

参 考 文 献

[1] Viterbi A J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm [J]. *IEEE Trans. on Information Theory*, 1967, 13(2): 260–269.

[2] Rader C M. Memory management in a Vitrbi decoder [J]. *IEEE Trans. on Communications*, 1981, 29(9): 1399–1401.

[3] Feygin G and Gulak P G. Architectural tradeoffs for survivor sequence memory management in Viterbi decoders [J]. *IEEE Trans. on Communications*, 1993, 41(3): 425–429.

[4] Black P J and Meng T H. A 140-Mb/s, 32-state, radix-4 Viterbi decoder [J]. *IEEE Journal of Solid-State Circuits*, 1992, 27(12): 1877–1885.

[5] Corporation A. Cyclone Device Handbook, Volume 1 [M]. 2003.

王建新： 男，1963 年生，教授，主要研究方向为雷达和通信信号处理、软件无线电技术.

于贵智： 男，1981 年生，硕士，研究方向为纠错码、基带调制解调.