

高性能 LDPC 编码器 IP 核设计与验证

刘艳欢¹, 崔文朋², 张一山¹, 朱曦阳¹, 郑 哲²

(1 清华大学 微电子学研究所, 北京 100084; 2 全球能源互联网研究院 信息与通信研究所, 北京 102211)

摘 要: 为了实现大数据时代以固态硬盘为代表的大容量高吞吐率存储器的纠错编码, 基于数学特性优异的 LDPC 码, 提出了一种半并行的 LDPC 编码器架构. 采用 SIMD 指令的调度控制流方式实现了 RU 编码算法, 完成了码率和码长等参数可动态配置的 LDPC 编码器的电路设计. 通过 Matlab 与 ModelSim 联合仿真, 并在 Xilinx FPGA 平台上验证了编码器的功能正确性. 结果表明: 工作频率为 100 MHz 时, 配置时间少于 7 μ s 且相对于其它编码器结构可以在较少的资源下吞吐率可达 4.82 Gb/s.

关键词: LDPC; 动态配置; 编码器; 固态硬盘

中图分类号: TN432

文献标识码: A

文章编号: 1000-7180(2017)04-0008-05

Design and Verification of High-performance LDPC Encoder IP Core

LIU Yan-huan¹, CUI Wen-peng², ZHANG Yi-shan¹, ZHU Xi-yang¹, ZHENG Zhe²

(1 Institute of Microelectronics, Tsinghua University, Beijing 100084, China; 2 Information and Communication Department, Global Energy Interconnection Research Institute, Beijing 1002211, China)

Abstract: A semi-parallel LDPC encoder architecture was proposed for Error correction coding used in a solid state hard drive (SSD) which is the representative of large capacity and high throughput memory based on the mathematical characteristics of excellent LDPC codes. Then SIMD instruction scheduling control flow approach was applied to achieve the RU coding algorithm; LDPC encoder whose parameters such as bit rate and code length were dynamically configurable was designed. The correctness of the encoder was verified through Xilinx FPGA platform and co-simulation of Matlab and ModelSim; Tests showed that configuration time was less than 7 μ s and the encoder achieved a throughput of 4.82 Gb/s using fewer resources compared with other encoders when the operating frequency is 100 MHz.

Key words: LDPC; dynamic configuration of IP core; encoder; SSD

1 引言

在当今大数据时代, 传统的机械硬盘 (Hard Disk Drive) 已经逐渐不能满足应用需求. 而固态硬盘 (Solid State Drive) 以闪存颗粒 (NAND Flash) 为存储介质, 结合新接口技术 SATA 或 PCI Express, 同时增加通道数量, 其读写速率能达到 10Gb/s, 成为当下的研究热点. 多比特存储技术提高了固态硬盘存储密度并降低成本, 但也使得闪存颗粒存储数

据的原始误码率越来越大, 达到 10^{-3} 量级. 于是在固态硬盘的控制芯片中引入基于 LDPC (LOW-density Parity-check) 码的编解码器来进行数据纠错, LDPC 码凭借其性能好, 纠错能力强以及优异的数学特性, 受到越来越多的关注.

对 RU 算法进行分析, 采用指令集的方式来实现编码运算要完成了可动态配置编码电路设计, 可支持不同码率和码长的编码需求, 而且配置时间很快. 可以更加方便的适用于不同数据系统中; 搭建一

套基于 ModelSim 与 Matlab 的联合编解码仿真验证系统平台,并给出仿真结果,并通过 FPGA 平台来验证其在硬件平台下的功能以及性能^[1-2]。

2 编码算法的选择

2.1 基本原理

LDPC 码作为前向纠错技术中的一种线性分组码,其数学结构清晰。它将原始数据按照 k 比特为一组打包成 $m = [m_{k-1}, \dots, m_1, m_0]$,然后通过方程 $c = mG$ 将其变成 $c = [c_{n-1}, \dots, c_1, c_0]$,其中的 G 对应的 k 维子空间与 H 矩阵对应 $n-k$ 维子空间正交,有 $GH^T = 0$ 。码字 c 属于 G 所在子空间,故有 $Hc^T = 0$,该公式可用于检验接收码字是否正确,故 H 矩阵称为校验矩阵。且 LDPC 码可用 H 矩阵来定义,但 H 矩阵的设计需要考虑码长、行重列重分布、环的分布和码字间距等参数,不在本文论述范围内,因为本文实现的纠错模块可动态支持各种 H 矩阵。

QC-LDPC 码是一种 LDPC 码,它是易于硬件实现的,对应的校验矩阵 H 可表示为式(1)。

$$H = \begin{bmatrix} Q^{a_{1,1}} & Q^{a_{1,2}} & \dots & Q^{a_{1,n}} \\ Q^{a_{2,1}} & Q^{a_{2,2}} & \dots & Q^{a_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ Q^{a_{m,1}} & Q^{a_{m,2}} & \dots & Q^{a_{m,n}} \end{bmatrix} \quad (1)$$

式(1)中, $Q^{a_{i,j}}$ 为一个 $b \times b$ 子矩阵,其可以为一个零矩阵或由 $b \times b$ 单位矩阵经循环移位而成。右上标 $a_{i,j}$ 表示子矩阵中“1”出现在第一排的位置。若 $a_{i,j} = 0$,意味着 $Q^{a_{i,j}}$ 为零矩阵。

2.2 传统编码算法

编码即由原始数据生成对应 LDPC 码字的过程。根据前面介绍,原始数据 m 经过公式 $c = mG$ 变为完整码字 c 。因此,传统的编码算法可视为矩阵乘法,如式(2)所示, G 矩阵前 k 列为单位矩阵。

$$\begin{bmatrix} c_{n-1} \\ c_{n-2} \\ \dots \\ c_0 \end{bmatrix}^T = \begin{bmatrix} m_{k-1} \\ m_{k-2} \\ \dots \\ m_0 \end{bmatrix}^T * \begin{bmatrix} 1 & 0 & \dots & 0 & g_{1k+1} & \dots & g_{1n} \\ 0 & 1 & \dots & 0 & g_{2k+1} & \dots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & g_{nk+1} & \dots & g_{nn} \end{bmatrix}_{k \times n} \quad (2)$$

对于一个 LDPC 码对应的 H 矩阵,其典型大小为 $35\ 840 \times 2\ 048$,则其对应的 G 矩阵大小为 $32\ 768 \times 35\ 840$,除去单位矩阵外还有 $32\ 768 \times 2\ 048$,这不具有稀疏性的部分将需要消耗大量资源,针对此问题 RU 算法被 Richardson 和 Urbanke 提出^[1],此算法直接基于 H 矩阵来进行编码,可充分发挥 H 矩阵的稀疏性优势。

2.3 RU 编码算法

对 H 矩阵进行分析可知,其校验过程本质上是很多个奇偶校验方程组,因此其行与行以及列与列之间的位置交换对校验方程组的结果无影响,本质上对 H 矩阵无改变。RU 算法^[3-4]正是利用这一特点,通过行列交换,把 H 矩阵变为准下三角矩阵,如图 1 所示,同时根据算法将校验矩阵划分为 6 个部分,其中 g 代表 H 矩阵和下三角矩阵所差行数,通常远小于 n 。

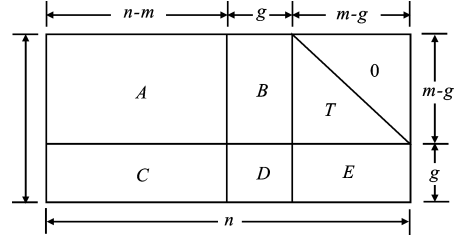


图 1 RU 编码算法对 H 矩阵的划分

设待编码数据为 r , 编码后添加的校验比特 $\{f_1, f_2\}$, 三者一起 $\{r, f_1, f_2\}$ 则形成一个 LDPC 码字。其中 r 的长度 $k = n - m$, f_1 长度为 g , f_2 长度为 $n - g$ 。 f_1 和 f_2 对应的计算如式(3)、(4)所示。

$$f_1^T = \psi^{-1} [ET^{-1}Ar^T + Cr^T] \quad (3)$$

$$f_2^T = T^{-1} [Ar^T + Bf_1^T] \quad (4)$$

$$\psi = ET^{-1}B + D \quad (5)$$

在上面算式中, ψ 矩阵及其他的逆矩阵可以提前计算出来,可看作已知参数。因此可以证明对于子矩阵大小为 b 的 QC-LDPC 码, ψ^{-1} 也由循环子矩阵构成,子矩阵个数为 $(g/b) \times (g/b)$,每个子矩阵由单位矩阵经过循环移位所得。

3 指令集的方式实现 RU 算法

由于在伽罗华二元域中矩阵乘法对应于元素的“与”和“异或”操作, H 矩阵由子矩阵组成,子矩阵是全零矩阵或者循环单位矩阵,所以子矩阵与向量的乘法,要不得到全零向量,要不就是向量的循环移位。因此矩阵乘法转化成细粒度计算为循环移位或按位异或^[5-6]。

本文中 LDPC 编码器的 SIMD 指令分为普通类型以及结束类型指令。结束类型指令标志着一次编码算法指令段的结束,当运行到该指令时,编码器控制模块即产生一个编码结束信号告诉前面模块这帧数据编码完成,并告诉其后面模块可以接收数据了。普通类型指令对应着一个循环单位子矩阵与向量相乘,然后可能与之前的中间结果按位异或。该指

令很灵活,在实现 RU 算法时可分解为一些基本指令的组合.

基于上面分析,本文完成了类 CPU 架构的 LDPC 编码器设计. 一条 LDPC 编码器的 SIMD 指令由不同信息块字段构成,表 1 中列出了各信息段对应的名称及含义. 然后本文完成了一个基于 Matlab 平台的编译器,输入 H 矩阵即可在指定路径生成对应二进制指令段文件.

表 1 LDPC 编码器指令的结构设计

对应比特位置	对应字段名称	位宽	含义
27	insWreg	1	0:结果不写入寄存器; 1:结果写入寄存器
26	insWram	1	0:结果不写入存储块; 1:结果写入存储块
25	insSRC2	1	0:来自寄存器; 1:来自存储块
24	insSRC1	1	0:来自寄存器; 1:来自存储块
23	insNclr	1	是否与中间结果 SRC1 按位异或
22:16	insWaddr	7	存储块 Data-Ram 的写 地址;
15:9	insRaddr	7	存储块 Data-Ram 的读 地址;
8:0	insShift	9	该子矩阵需要循环移位的 次数

4 编码器硬件电路设计

4.1 内核电路结构

在对编码算法进行研究并设计了对应的指令实现方式的基础上,本节对编码器的内部电路结构进行设计. LDPC 编码器需要包含指令存储、指令解析运行、待编码数据存储、中间结果存储、以及运算单元等,这些模块通过相应控制信号和数据信号连成一个整体,如图 2 所示.

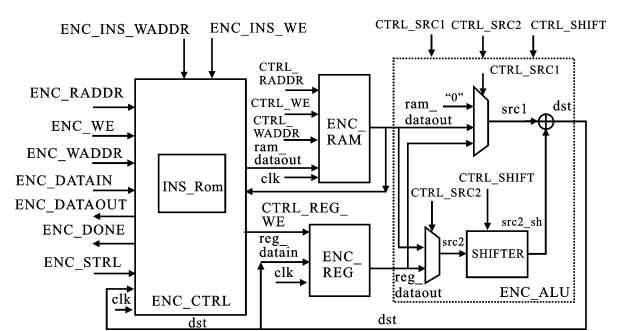


图 2 LDPC 编码器的内核结构

中心控制器(ENC_CTRL)作为整个 LDPC 编码器的核心部分,它不仅负责对外接口控制时序的

实现,还负责编码算法中 SIMD 指令的读取与解析的工作,其内部含有指令存储块(INS-Ram). 数据存储块(ENC_RAM)用于存储待编码原始数据、中间计算结果以及最终编码后的检验比特. 数据寄存器(ENC_REG)内含 512 比特(等于 QC-LDPC 的子矩阵大小)的寄存器,用于在 RU 算法执行中缓存中间数据. 算术逻辑单元(ENC_ALU)用于执行 SIMD 指令中的具体逻辑运算. 此系统在 ENC_CTRL 模块产生的控制流下运行.

在图 2 中“→”表示编码器与外界接口,“→”表示编码器内部信号传递. 上面电路作为编码器内核结构,通过将 ENC_INS_WE 置位来启动配置,待指令文件输入完后将相应使能信号复位,接着 ENC_WE 信号置位,开始待编码数据的输入,然后 ENC_START 信号置位让编码器开始执行编码算法,编码结束后 ENC_DONE 信号置位,接着便可以输出编码后的数据. 上述过程需要外部提供相应的地址和较复杂的控制逻辑.

4.2 可配置电路结构

通过前面的论述可以看出内核结构需要外部提供较复杂的时序控制,而本文设计的 LDPC 编码器主要用于固态硬盘等系统中,要求纠错编码器在外部提供简单的启动信号、地址和数据的情况下就能正常工作. 下面从几个方面在内核结构基础上对编码器进行完善.

(1)增加接口电路设计

编码器内核结构对外部信号的依赖性较强,直接用于各系统中比较麻烦,本节通过添加一个简单接口控制电路来为内核电路提供时序逻辑,如图 3 中的 ENC_CTRL2 模块所示. 对于任意存储器系统,只需外部提供简单的启动配置信号(INS_START)、启动数据载入信号(DATA_START)以及相应的地址数据信号就可以让编码器正常工作. 增加接口电路后该编码器可以很容易实现连续编码多帧数据,而这可以有效满足各种系统中的工作要求.

(2)增加动态配置功能模块

本文中的编码器的设计基于 RU 算法,通过生成与 H 矩阵对应的指令段,然后设计类 CPU 结构来执行指令序列完成 LDPC 编码功能,相应的解码器也必然根据 H 矩阵并采用某种原理来设计. 如图 3 所示,本节通过在内核结构中添加 INS_RAM1 模块以及将原 INS_ROM 改成 INS_RAM2 模块,并对控制模块做相应修改使得可以从外部动态改变编

码器的指令序列,实现可动态配置功能。

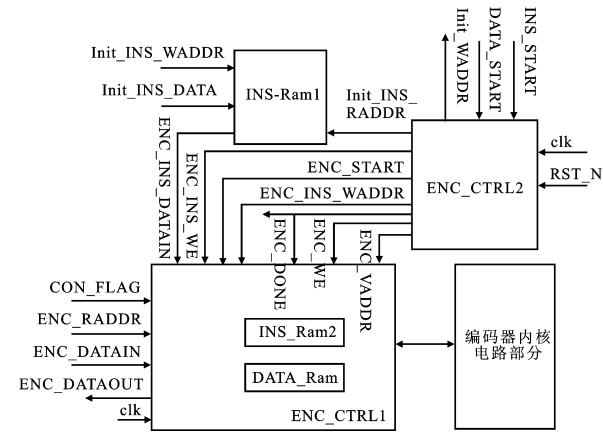


图 3 可动态配置 LDPC 编码器结构

(3)对内核结构的配置时间进行优化

在内核电路结构中,需要先载入指令文件,待其结束后再进行待编码数据的输入.本节对控制逻辑模块进行优化实现了指令文件和待编码数据同时载入的功能,经测试,在工作频率为 100 MHz 时,该 LDPC 编码器的配置时间相比内核结构减少了约 1 μ s,且绝对配置时间少于 7 μ s.因此当该编码器用于需要动态配置系统中时,配置之间的切换将变得很快.

LDPC 编码器的完整结构如图 3 所示.图中的 INS-Ram1 模块用于存储外部输入指令段文件,并在接口控制模块 ENC_CTRL2 的控制下将指令数据载入 INS_Ram2 模块中实现动态配置功能.

5 LDPC 编码器设计验证

5.1 仿真验证

本文设计的 LDPC 编码器作为固态硬盘等存储系统中纠错模块的编码部分,首先采用 Matlab 与 Modelsim 联合仿真.其中 Matlab 主要负责准备待编码数据以及从软件层面实现 RU 编码算法,并对编码结果进行保存.在解码器的仿真上主要实现误比特的模拟添加以及从软件上实现解码算法. Modelsim 则主要从 RTL 级对 LDPC 编解码器的功能进行仿真,并将结果与 Matlab 的结果进行比较以证明电路设计的正确性.验证结构如图 4 所示.由于本文的编码器是整个纠错编解码器的一部分,本文给出整体仿真结构图.

在图 4 中,Matlab 的仿真结果与 Modelsim 的结果通过 Comp 进行对比,如果结果一致,则表明编、解码器的功能正确.

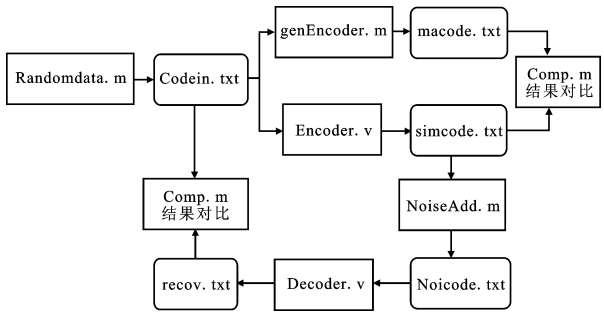


图 4 LDPC 编(解)码器的仿真结构图

在测试中,本文给出一系列随机原始待编码数据,通过 Matlab 平台输出的 macode.txt 与 RTL 级设计编码器给出的 simcode.txt 总是完全一致,而且不同的 H 矩阵对应的 Matlab 编码结果与通过 H 矩阵生成的指令段配置的 RTL 级编码器的编码结果都一致,证明了本文设计的 LDPC 编码器的功能正确且可动态配置.图 5 给出了内核电路和完整编码电路仿真图,可以看出,最终的 LDPC 编码器对外接口简单且可动态配置,不仅可以连续编码多帧数据,而且配置时间也很短,约为 6 μ s.

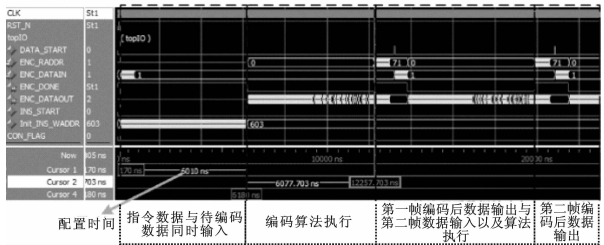


图 5 完整 LDPC 编码器仿真

5.2 FPGA 验证

本文基于 Xilinx 公司的 VC709 开发板,该开发板功能很强大,其中的核心芯片为 XC7VX690T,可以提供最高 693 k 的逻辑资源.并且本文设计了相关电路模块来搭建验证系统,如图 6 所示.

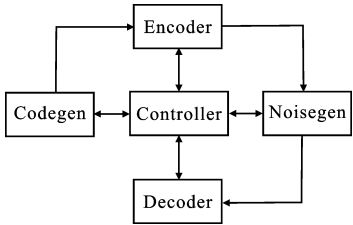


图 6 LDPC 编(解)码器的 FPGA 平台验证框图

在图 6 中,Codegen 模块利用串 M 序列来产生随机待编码数据,然后将数据传到 Encoder 模块,编码之后的数据经过 Noisegen 模块来模拟数据在固

态硬盘中的存储时出现的随机错误,该数据也被送往解码模块 Decoder 中进行解码. controller 模块作为验证系统的核心部分,主要提控制功能、缓存输入输出结果以及对结果进行比较验证编解码器功能的正确性.

本文中统筹设计编码器与解码器联合验证系统,为后期设计带来很大便利,并且在对 LDPC 编码器单独验证时,采用在 Matlab 平台下实现的编码输出数据载入 controller 中的存储部分,然后与 FPGA 平台下的编码输出数据进行对比验证. 结果显示编码器功能正确.

表 2 各种编码器对比

编码器	逻辑单元	频率 /MHz	吞吐量 /(Mbps)	工作平台
本文	862	100	4 822	Xilinx(XC7VX690T)
Xi IP	1 204	106	106	Xilinx(Xc2vpx70)
文献[4]	12 306	151	5 170	Altera(EP1S25F672C6)
文献[6]	3 352	85	1 164	Altera(EP2c70F672C8)

如表 2 给出了本文实现结果与其他方式实现结果的对比. 其中,文献[4]中每个编码步骤都采用专用硬件结构实现,吞吐量最高,但不能动态配置,这限制了其的使用. Xilinx IP 核的编码器指令可在线配置,所消耗的资源也很少,但其采用的是串行编码实现,至少要 10 套电路并行工作才能达到与本文设计相同的吞吐量,消耗的资源与文献[4]接近. 文献[6]中的编码器可在线编程,但是其消耗的逻辑单元数很多,最高工作频率和吞吐量也都比本文中的编码器低,且其主要用于无线通信编码领域. 本文设计的编码器体现了较好的可配置能力,而且用较少的资源实现了不错的性能,实验测得在 100 MHz 的工作频率下吞吐率达到 4. 822 Gb/s.

6 结束语

本文通过 RU 算法与 LDPC 码对应的 H 矩阵相结合,并采用 SIMD 指令集的方式来实现算法. 接着设计了 LDPC 编码器内核电路结构以及对应的接口模块并增加动态可配置功能. 突破了传统固定逻辑电路结构设计,能适用于不同系统,这大大体现

了其的灵活性. 同时将该编码器与其它编码器性能进行比较,凸显出本文设计的编码器的优越性.

参考文献:

[1] Richardson T J, Urbanke R L. Efficient encoding of low-density parity-check codes[J]. Information Theory, IEEE Transactions on, 2001, 47(2): 638-656.

[2] Yoo H, Lee Y, Park I C. 7. 3 Gb/s universal BCH encoder and decoder for SSD controllers[C]//Design Automation Conference (ASP-DAC), 2014. Asia and South Pacific, IEEE, 2014: 37-38.

[3] Sun X R, Zeng Z, Yang Z. A novel low complexity LDPC encoder based on optimized RU algorithm with backtracking[C]//Multimedia Technology (I, CMT), 2010 International Conference on. Ningbo, China, IEEE, 2010: 1-4.

[4] Kopparthi S, Gruenbacher D M . Implementation of a flexible encoder for structured low-density parity-check codes [C] / /Communications, Computers and Signal Processing, IEEE Pacific Rim Conference. Victoria, Canada,2007: 438-441.

[5] Lee D U, Luk W, Wang C, et al. A flexible hardware encoder for low-density parity-check codes[C]// 12th IEEE Symposium on Field-Programmable Custom Computing Machines , FCCM 2004. USA, Napa, 2004;20-23.

[6] 赵明,李亮. 在线可编程 QC-LDPC 码高速编码器结构[J]. 清华大学学报(自然科学版), 2009, 49(7): 1041-1044.

作者简介:

刘艳欢 男,(1992-),硕士研究生. 研究方向为数字集成电路系统设计. E-mail:lyh15@mzil:tsinghua. edu. cn.

崔文朋 男,(1987-),工程师. 研究方向为嵌入式系统软件设计.

张一山 男,(1991-),硕士研究生. 研究方向为数字集成系统设计.

朱曦阳 男,(1992-),硕士研究生. 研究方向为数字集成系统设计.

郑 哲 男,(1986-),工程师. 研究方向为数字集成电路设计、嵌入式系统设计、电力通信.