

西安电子科技大学

硕士学位论文



基于IEEE802.16e标准的
码率兼容QC-LDPC编译码器的FPGA实现

作者姓名 安永宁 导师姓名、职称 张海林教授

一级学科 军队指挥学 二级学科 军事通信学

申请学位类别 工学硕士 提交学位论文日期 2014年11月

西安电子科技大学 学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名： 安永宁 日 期： 2014.12.22

西安电子科技大学 关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，获得学位后结合学位论文研究成果撰写的文章，署名为西安电子科技大学。

保密的学位论文在____年解密后适用本授权书。

本人签名： 安永宁 导师签名： 郭晓林
日 期： 2014.12.22 日 期： 2014.12.22

学校代码 10701
分 类 号 TN91

学 号 1201120662
密 级 公 开

西安电子科技大学

硕士学位论文

基于IEEE802.16e标准的
码率兼容QC-LDPC编译码器的FPGA实现

作者姓名:安永宁

一级学科:军队指挥学

二级学科:军事通信学

学位类别:工学硕士

指导教师姓名、职称:张海林教授

提交日期:2014 年 11 月

FPGA implementation of rate-compatible QC-LDPC Codec based on IEEE802.16e standard

A thesis submitted to
XIDIAN UNIVERSITY
in partial fulfillment of the requirements
for the degree of Master
in Military Command

By
An Yongning
Supervisor: Prof. Zhang Hailin
November 2014

摘要

低密度奇偶校验码(Low-Density Parity-Check Codes, LDPC)作为一种优秀的信道纠错码, 以较低的实现复杂度和逼近香农极限的良好性能成为信道编码领域最令人瞩目的研究热点, 并被广泛应用于包括 IEEE802.16e、IEEE802.11n、DVB-S2 和 CMMB 等各种通信协议标准中。为了支持多种业务需求和提高适应性的要求, 目前多数通信标准都要求 LDPC 码能够同时支持多种码长和码率。

基于实际项目需要, 本文的设计目标是实现能够兼容 1008 码长、1/2 码率和 1024 码长、3/4 码率的 LDPC 编译码器, 并且能够通过利用 IEEE802.16e 协议中的 QC-LDPC 码独特结构设计有效的存储策略, 从而减少硬件实现时的资源占用, 实现与时延等性能之间的平衡。本文主要工作围绕着在 FPGA 上设计实现一组满足上述要求的码率兼容的 LDPC 编译码器展开。

本文首先介绍了 LDPC 码定义和 Tanner 图表示以及 QC-LDPC 码的相关知识和结构特点, 分别具体推导了 LDPC 码的传统编码算法、RU 编码算法和 QC-LDPC 线性编码算法以及 BP 译码算法、最小和算法以及归一化最小和算法等, 并在此基础上对比了各种算法在仿真性能和实现复杂度方面的差异, 确定出最适合硬件实现的 QC-LDPC 线性编码算法和归一化最小和译码算法。

然后, 在对 LDPC 码的译码量化方案进行详细探讨的基础上通过软件仿真, 确定了输入软信息 4bit 量化、中间数据 12bit 有符号整形表示的硬件实现量化方案。结合 IEEE802.16e 标准中 QC-LDPC 码的准循环双对角结构特点, 设计出码率兼容的半并行中间变量数据处理结构和基于位置编址的存储策略。整体采用流水线和乒乓操作的设计方法, 可有效改善编译码器的时延特性。

最后, 重点讨论了码率兼容 QC-LDPC 码的设计和实现方案, 并运用硬件描述语言在 Quartus II 软件平台上完成了能够满足需求的 LDPC 编译码器的设计, 并通过仿真工具完成了编译码器的仿真和验证, 结果表明该硬件结构和存储策略能够达到设计的要求。

关键词: IEEE802.16e, 码率兼容, QC-LDPC, FPGA 实现

论文类型: 应用基础研究类

ABSTRACT

Low-density parity-check (LDPC) codes, as an excellent Channel Error Correction Codes, have become the most notable hotspot and widely used in various communication standards, such as IEEE802.16e, IEEE802.11n, DVB-S2 and CMMB standards. In order to support multiple business needs and improve the adaptability of the requirements, most communication standards require LDPC codes support flexible coding length and rate.

The implementation of codec on which 1008 length 1/2 rate and 1024 length 3/4 rate are compatible very well, is based on the actual needs of the project. Resource consumption will be significantly reduced through effective storage strategy and the unique combination of QC-LDPC codes structure which is recommended by IEEE802.16e standard, which achieves a balance between performance and latency, etc. This paper shows the main work around the implementation on FPGA to meet the requirements of one set of rate-compatible LDPC codec based on IEEE802.16e standard.

This paper briefly introduces the basic theory of LDPC codes and QC-LDPC codes. After derived various algorithms systematically, including traditional encoding algorithm, RU encoding algorithm, QC-LDPC linear encoding algorithm and BP algorithm, min-sum algorithm, normalized min-sum algorithm, etc. This paper compares their performances and implement costs, and then determines the most suitable QC-LDPC encoding algorithm and normalized Min-sum decoding algorithm for hardware implementation.

With the help of simulation software and detailed discussion, this paper determines the FPGA implementation quantization scheme where the soft input information 4bit quantified, intermediate data 12bit signed integer represented. Combined with the quasi-cyclic double diagonal structural characteristics of QC-LDPC in IEEE802.16e standard, rate-compatible semi-parallel data processing hardware structure and storage strategy based on location addressing are designed. The overall design method of using pipelines and ping-pong operations, can effectively improve the delay characteristics of the codec.

Finally, the paper focuses on the implementation of rate-compatible QC-LDPC codec. Using the hardware description language in Quartus II software platform, the paper implements the design of codec. The simulation results indicate that the codec hardware architecture and storage strategy can meet the design demand.

Key Words: IEEE802.16e, rate-compatible, QC-LDPC, FPGA implementation

Type of Dissertation: Applied Basic Research

插图索引

| | |
|--------------------------------------|----|
| 图 2.1 系统码的一种结构形式..... | 8 |
| 图 2.2 一个 (6, 3) 分组码对应的 Tanner 图..... | 9 |
| 图 3.1 近似下三角结构的 LDPC 码校验矩阵 | 16 |
| 图 3.2 下三角结构的 LDPC 码校验矩阵 | 17 |
| 图 3.3 信息传递译码的 Tanner 图..... | 20 |
| 图 3.4 3-SR 节点恢复树结构图 | 24 |
| 图 4.1 接收信号的概率密度曲线..... | 26 |
| 图 4.2 限幅对误码率的影响..... | 27 |
| 图 4.3 量化对误码率的影响..... | 27 |
| 图 4.4 量化对中间变量最大值的影响..... | 28 |
| 图 4.5 迭代次数对中间变量最大值的影响..... | 28 |
| 图 5.1 LDPC 编码器实现框图 | 31 |
| 图 5.2 输入数据移位寄存器工作原理..... | 31 |
| 图 5.3 编码处理模块数据流结构图..... | 32 |
| 图 5.4 循环移位模块 RTL 图 | 33 |
| 图 5.5 编码器状态转换图..... | 33 |
| 图 5.6 LDPC 编码器接口信号时序 | 35 |
| 图 5.7 LDPC 译码器实现框图 | 36 |
| 图 5.8 变量节点更新运算单元 RTL 图 | 36 |
| 图 5.9 校验移位信息生成单元 RTL 图 | 37 |
| 图 5.10 校验存储器地址产生单元 RTL 图 | 37 |
| 图 5.11 校验移位数据存储器 RTL 图 | 37 |
| 图 5.12 移位处理单元 RTL 图 | 37 |
| 图 5.13 译码累加运算器单元 RTL 图 | 38 |
| 图 5.14 译码累加运算底层模块 RTL 图 | 39 |
| 图 5.15 校验节点更新运算单元 RTL 图 | 39 |
| 图 5.16 变量移位信息生成单元 RTL 图 | 39 |
| 图 5.17 变量存储器地址产生单元 RTL 图 | 40 |
| 图 5.18 变量移位数据存储器 RTL 图 | 40 |
| 图 5.19 移位处理单元 RTL 图 | 40 |
| 图 5.20 归一化最小和搜索运算单元 RTL 图 | 40 |

| | |
|-----------------------------------|----|
| 图 5.21 归一化最小和搜索运算底层单元 RTL 图 | 41 |
| 图 5.22 译码器判决处理单元 RTL 图 | 42 |
| 图 5.23 译码判决流程图..... | 42 |
| 图 5.24 判决存储器地址生成单元 RTL 图 | 43 |
| 图 5.25 校验移位处理单元 RTL 图 | 43 |
| 图 5.26 校验累加和运算单元 RTL 图 | 44 |
| 图 5.27 译码器状态转换图..... | 44 |
| 图 5.28 LDPC 译码器接口信号时序 | 46 |
| 图 5.29 译码器编译综合结果图..... | 47 |
| 图 5.30 输出数据移位寄存器工作原理..... | 47 |
| 图 5.31 移位信息生成模块中初始化数据格式..... | 48 |
| 图 5.32 判决控制单元中初始化数据格式..... | 49 |

表格索引

| | |
|-----------------------------|----|
| 表 3.1 P_1^T 的分解计算步骤 | 17 |
| 表 3.2 P_2^T 的分解计算步骤 | 17 |
| 表 5.1 编码器接口使用说明表 | 34 |
| 表 5.2 编码器资源占用 | 35 |
| 表 5.3 译码器接口使用说明表 | 45 |
| 表 5.4 译码器状态标志信号说明表 | 46 |
| 表 5.5 译码器资源占用 | 46 |

符号对照表

| 符号 | 符号名称 |
|------------|--------|
| V | 变量节点集合 |
| C | 校验节点集合 |
| $GF(2)$ | 二元有限域 |
| H_{base} | 基本阵 |
| H | 校验矩阵 |
| G | 生成矩阵 |
| c_i | 校验节点 |
| v_j | 变量节点 |

缩略语对照图

| 缩略语 | 英文全称 | 中文对照 |
|-------|--|-----------|
| DMC | Discret Memoryless Channel | 离散无记忆信道 |
| FEC | Forward Error Control | 前向差错控制 |
| LDPC | Low Density Parity Check Code | 低密度奇偶校验码 |
| RS | Reed-Solomon codes | 里所码 |
| CRC | Cyclic Redundancy Check | 循环冗余校验 |
| TCM | Trellis Coded Modulation | 格形编码调制 |
| LLR | Log Likelihood Ratio | 对数域最大似然概率 |
| ML | Maximum Likelihood | 最大似然概率 |
| SI/SO | Soft In/Soft out | 软输入/软输出 |
| EMD | Extrinsic Message Degree | 外信息度 |
| PEG | Progressive Edge-Growth | 渐近边增长 |
| ACE | Approximate Cycle Extrinsic Message Degree | 近似环外信息度 |
| RA | Repeat Accumulate | 重复累积码 |
| IRA | Irregular RA | 非规则 RA 码 |
| BF | Bit-Flipping | 比特翻转 |
| BP | Belief Passing | 置信传播 |
| WBF | Weighted Bit-Flipping | 加权的比特翻转 |
| MWBF | Modified Weighted Bit-Flipping | 改进的加权比特反转 |
| MLD | Majority-Logic Decoding | 大数逻辑译码 |
| WMLD | Weighted Majority-Logic Decoding | 加权大数逻辑译码 |
| APP | A Posteriori Probability | 后验概率 |
| SPA | Sum-Product Algorithm | 和积算法 |
| MPA | Message Passing Algorithm | 消息传递算法 |
| MSA | Min-Sum Algorithm | 最小和算法 |
| WLAN | Wireless Local Area Networks | 无线局域网络 |
| DSL | Digital Subscriber Line | 数字用户专线 |
| CMMB | China Mobile Multimedia Broadcasting | 中国移动多媒体广播 |
| CSM | Cyclic Shift Matrix | 循环移位矩阵 |
| CSI | Cyclic Shift Identity | 循环移位单位阵 |
| QCI | Quasi Cyclic Identity | 准循环移位单位阵 |

| | | |
|---------|--|------------|
| QCM | Quasi Cyclic Matrix | 准循环移位阵 |
| QC-LDPC | Quasi-Cyclic Low-Density Parity-Check | 准循环 LDPC 码 |
| VLSI | Very Large Scale Integration | 超大规模集成电路 |
| DTMB | Digital Television Terrestrial Multimedia Broadcasting | 数字电视地面广播 |
| ETSI | European Telecommunications Standards Institute | 欧洲电信标准协会 |

目录

| | |
|--------------------------|------|
| 摘要..... | I |
| ABSTRACT | III |
| 插图索引..... | V |
| 表格索引..... | VII |
| 符号对照表..... | IX |
| 缩略语对照图..... | XI |
| 目录..... | XIII |
| 第一章 绪论..... | 1 |
| 1.1 信道编码理论简介 | 1 |
| 1.1.1 信道编码理论的提出..... | 1 |
| 1.1.2 信道编码理论的发展..... | 1 |
| 1.2 LDPC 的再次提出 | 2 |
| 1.3 LDPC 的研究进展 | 3 |
| 1.3.1 LDPC 码的构造方法 | 3 |
| 1.3.2 LDPC 的快速编码 | 4 |
| 1.3.3 LDPC 码的译码 | 5 |
| 1.3.4 LDPC 应用现状 | 5 |
| 1.4 本文主要内容及安排 | 6 |
| 第二章 LDPC 码基础..... | 7 |
| 2.1 线性分组码的基本概念 | 7 |
| 2.1.1 线性分组码的定义..... | 7 |
| 2.1.2 线性分组码的参数指标..... | 7 |
| 2.1.3 线性分组码的表征..... | 7 |
| 2.1.4 系统码..... | 8 |
| 2.2 LDPC 码的基本概念 | 9 |
| 2.2.1 定义..... | 9 |
| 2.2.2 Tanner 图表示..... | 9 |
| 2.2.3 正则 LDPC 码 | 10 |
| 2.3 QC-LDPC 码 | 11 |
| 2.3.1 QC-LDPC 码基本概念..... | 11 |
| 2.3.2 QC-LDPC 的矩阵扩展..... | 12 |

| | |
|------------------------------------|-----------|
| 2.4 IEEE 802.16e 标准中的 LDPC 码 | 12 |
| 第三章 LDPC 算法原理..... | 15 |
| 3.1 LDPC 编码算法..... | 15 |
| 3.1.1 传统编码算法..... | 15 |
| 3.1.2 RU 编码算法 | 15 |
| 3.1.3 QC-LDPC 编码..... | 18 |
| 3.2 LDPC 译码算法 | 19 |
| 3.2.1 硬判决译码算法..... | 19 |
| 3.2.2 BP 算法..... | 19 |
| 3.2.3 最小和算法..... | 22 |
| 3.2.4 归一化最小和算法..... | 23 |
| 3.3 打孔算法 | 23 |
| 第四章 译码器的定点量化方案 | 25 |
| 4.1 定点量化处理需要考虑的原则 | 25 |
| 4.2 量化译码背景 | 25 |
| 4.2.1 量化比特数的选择..... | 25 |
| 4.2.2 量化数据选择..... | 26 |
| 4.3 归一化最小和译码算法的量化处理 | 26 |
| 4.3.1 译码器接收变量的量化处理..... | 26 |
| 4.3.2 内部数据最大值仿真..... | 28 |
| 第五章 LDPC 编译码器的 FPGA 设计..... | 31 |
| 5.1 LDPC 编码器的设计 | 31 |
| 5.1.1 LDPC 编码器设计框图 | 31 |
| 5.1.2 LDPC 编码器顶层接口 | 34 |
| 5.1.3 LDPC 编码器 I/O 时序图 | 35 |
| 5.1.4 LDPC 编码器性能 | 35 |
| 5.2 LDPC 译码器 | 36 |
| 5.2.1 LDPC 译码器设计框图 | 36 |
| 5.2.2 LDPC 译码器顶层接口 | 45 |
| 5.2.3 LDPC 译码器 I/O 接口时序 | 45 |
| 5.2.4 译码器资源占用和时间占用情况..... | 46 |
| 5.3 码率兼容的实现方法 | 47 |
| 第六章 本文总结..... | 51 |
| 附录 A..... | 53 |

| | |
|-----------------------|----|
| 参考文献..... | 55 |
| 致谢..... | 61 |
| 作者简介..... | 63 |
| 1.基本情况 | 63 |
| 2.教育背景 | 63 |
| 3.攻读硕士学位期间的研究成果 | 63 |

第一章 绪论

1.1 信道编码理论简介

1.1.1 信道编码理论的提出

Shannon^[1]于 1948 年首次将信道编码定理提出,奠定了信息论和编码理论的基础。该定理指出,在信息速率小于信道容量的约束条件下,如果要在不牺牲信息传输速率的同时,使得有噪音信道带来的恶化降低到无穷小,可以通过一种编码方式来实现。可见,信道编码是一种有效进行可靠信息传输的方式。

香农定理指出:对于典型的 DMC 信道,存在一个确定的信道容量 C ,当信道的信息传输率 $R < C$ 时,只要码长 n 足够大且采用典型集合译码时,必然有一种相对应的编译码规则,使得接收端的译码错误概率趋于无穷小,相反地,如果 $R > C$,则无论如何使用编码方案都不能实现无差错传输。

1.1.2 信道编码理论的发展

虽然满足信道编码要求的码的“存在性”已被 Shannon 通过信道编码定理证明,并给出了在有噪信道上实现可靠通信的理论极限,但是该证明的给出需要建立在编码随机、码长无穷大并且必须采用典型集合译码三个重要前提,显然这是不可实现的。他并没有提供任何寻找这些码结构的方法,也没有给出构造最佳码和实现最佳译码算法的具体方法。所以,在过去的 60 多年里,从事信道编码方面工作的研究者在不懈的寻找性能尽量逼近香农限,同时编译码实现难度又不太高的信道编码方案。

最先被发现的一种纠错码是利用各种代数和几何方法(例如群、环、域)构造设计出来的线性分组码。历史上第一次实用的前向纠错编码方案是在 1949 年和 1950 年被分别提出的格雷码^[2]和汉明码的^[3]。1954 年, D. Muller 和 I. Reed 分别为通信、数据存储进行差错检测而提出了 Reed-Muller 码^[4],并且实现了该码的第一个译码器^[5]。1957 年, CRC 码的概念第一次被 E. Prange 提出^[6]。1959 年, Bose 和 Chaudhuri 以及 Hocuenghem 分别发现了 BCH 码具有严密的代数结构并且性能在短码和中码情况下接近于理论值^[7]。Reed 和 Solomon 于 1960 年提出了 RS 码的编码方案。

卷积码^[8]是一类区别于分组码的纠错编码,于 1955 年被 P. Elias 提出。通过将编码码元序列寄存的方式,使前后的码元之间产生相关性,即校验码元同时与信息码元都相关。其在码率和实现复杂度与分组码相同的条件下得到的编码增益会更高一些。由于卷积码编译码是连续进行的特点,使得其处理时延更小。然而,

普通的线性分组码和卷积码的性能离香农限依然存在很大差距，译码复杂度也较高。

序列译码算法^[9]于 1961 年被 Wozencraft 和 Reiffen 提出，之后的概率译码算法^[10]是 Fano 在前者的基础上改进得到的。1963 年，充分利用了码字代数结构的门限译码算法^[11]由 Massey 提出。在 1967 年 Viterbi 译码算法^[12]被提出之后，被 Forney 证明是卷积码的最大似然译码算法^[13]。特别值得一提的是，Gallager 也是在这一时间段，第一次将 LDPC 码^[15]提出来。

被 Forney 在 1966 年提出的级联码^[14]是通过若干个短码进行级联而构造出来的长码，可以在不使级联结构过于复杂的情况下，不仅实现复杂度得到降低，而且可以得到逼近极限的性能，另外还可以进一步减小误码率的错误平层。1973 年，Forney 首次以网格表示作为工具来对 Viterbi 译码算法^[16]进行分析阐释。在 Bahl 和 Wolf 等人分别于 1974 和 1978 年首次提出针对线性分组码的网格表示^[17]和构造方法^[18]，Massey 则明确定义了“码网格”的概念^[19]。1988 年，像 RM 码这样的一些分组码具有“简单网格结构”的性质被 Forney 给出了证明^[20]。

Ungerboeck 于 1982 年提出的 TCM^[21]方案，是将卷积码和调制方式结合起来，通过对符号映射空间进行扩展的方法将其运用于分组码的调制技术上，一般是将码字划分成码字长度较小或处理难度较小的网格以获得编码增益的提高。TCM 方案打破了传统的信道编码中要改善系统的可靠性只能展宽传输频谱的禁锢，特别适用于带宽受限的信道传输。1987 年，多维的 TCM 网格编码技术^[22]由 L. Wei 以 TCM 方案为基础而提出来的。

1993 年，法国的 C. Berrou 等人提出了 Turbo 码^[23]的概念，它是利用伪随机交织器和并行级联的结构构造出具有伪随机特性的长码，使用最大后验概率和软输入/软输出的带有交织器的反馈迭代译码进行多次更新迭代来逼近最大似然译码，使其性能接近香农限，以牺牲少量的性能来换取实现复杂度的降低^[24]。

Turbo 码的编码算法简单，译码可以通过最大似然概率算法和对数域最大似然概率算法实现最佳译码。Mackay 和 Neal 在研究 Turbo 码的过程当中，于 1996 年发现之前由 Gallager^[25]提出的 LDPC 码和 Turbo 码有着极其相近的优异性能^[26]。当对 LDPC 码采用 SI/SO 迭代译码时，其性能可逼近香农限。特别是 LDPC 码比 Turbo 码具有良好的内交织特性、更好的误码性能、更低的错误平层和更简单的译码算法等优点，使得 LDPC 码得到信道编码领域的普遍关注，从而成为可靠通信传输中信道编码理论研究的新热点。

1.2 LDPC 的再次提出

首次在 1962 年被提出的低密度校验码（即 LDPC 码）具有极好的汉明距离特

性，以后验概率译码算法在进行多次迭代更新之后得到的误码率曲线随着码长增大呈指数减小。

在论文里，Gallager 系统地论述了 LDPC 码的基本概念，从而为人们对 LDPC 码的研究工作奠定了坚实的理论基础。尽管 LDPC 码表现出了优异的性能，然而，受当时计算机发展水平和人们认识的局限性影响，并且硬件实现较为困难，LDPC 被认为是不实用码，被长期的遗忘了。

Tanner 在 1981 年再次对 LDPC 码^[27]进行系统的研究，他从图的观点基于图模型来引入 Tanner 图，为 LDPC 码提供了一种全新的阐释方式，证明了 Gallager 提出的译码算法与 Tanner 图中的环之间的联系，提出了一种规范的图表示方法，同时还提出了两种消息传递算法：和积算法与最小和算法。

1996 年，在对 LDPC 码进行深刻理解之后，Mackay、Ruby 等对其理论和实际性能进行了详细论述，再次证明 LDPC 码是一种优秀的码型^[28]，对 Gallager 提出的概率迭代译码算法进行推广，并且说明了和积译码算法的具体实现方案，对 LDPC 码的发展起了极大的推进作用。

1997 年 Mackay 和 Luby 在提出并证明了非正则 LDPC 码具有较好的译码性能。Urbank、Richardson 等在总结前者的分析方法后提出了一种被称之为密度进化理论的新式编码算法^{[29][30]}，并且对消息传递译码情况下的容量进行分析，设计出了逼近香农极限的非正则 LDPC 码^[31]，同时指出该码的快速编码方法，降低了编码时由于构造的随机性带来的运算和存储量，在码构造和设计等方面做出了巨大的贡献。

1.3 LDPC 的研究进展

一直以来，学者们对 LDPC 码的研究主要从构造、编码、译码以及应用实现几个方面开展，现从以下几个方面大致概括。

1.3.1 LDPC 码的构造方法

LDPC 码的构造设计都是通过设计码的 H 阵来实现的，主要的方法大致可以分为随机构造和结构化构造两种。

随机构造法是在特定的设计约束条件下，利用计算机进行随机搜索的方法来构造获得 H 矩阵。典型的随机构造包括 Gallager 构造、Mackay 构造、超轻构造、比特填充构造、扩展比特填充构造、基于围长分布的启发式算法、基于外信息度的 EMD 构造、渐近边增长构造和近似环外信息度构造等。随机构造法因为没有太多的限制，所构造的 LDPC 码性能总体较好，但随机码编译复杂度通常较高，并且其构造本身就带有相当大的随机性，不利于硬件编码器的实现。

为解决此随机性带来的问题，学者们提出了相对有章可循，而且可以保证中

长码和短码也具有较好距离特性的结构化构造方法，希望能简化构造便于编码硬件实现。结构化构造主要是以代数学和组合数学为基础，代表性的构造方法包括：有限几何构造法、组合设计构造法、基于 RS 码构造法、分组-移位构造法、 π 旋转构造法和准循环 LDPC 码构造法等。利用此方法进行构造 LDPC 在当前实际系统中得到了广泛应用。

特别需要指出的是，Fossorier 和 Z. Li 等人^[32]提出的具备准循环特性的 LDPC 码。QC-LDPC 码的 H 都是由零矩阵和单位矩阵 I 及其移位矩阵组合而成，各分块矩阵在 H 中的排列方式可能不同，例如分块矩阵的维数在 Tanner 设计的 H 中，要求必须是素数的偶数倍或其本身，而 Fossorier 则要求其是不包括 2 的幂次方在内的所有整数。循环或准循环特性是 QC-LDPC 区别于其它构造法得到的 LDPC 码的主要特点，而正是由于该特性能够利用简单反馈移位寄存器使得线性复杂度编码能够实现，从而获得编码增益的提高^{[33][34]}。

除此之外，人们还提出了一些不仅可以增大参数范围，而且能够在一定条件下提高性能的构造通用技巧。已知的典型通用技巧包括：行列分割^[35]、叠加^[36]、提升、掩蔽^[37]和中国剩余定理^[38]等。

1.3.2 LDPC 的快速编码

MacKay 最先给出 LDPC 码作为一般线性分组码进行处理的编码方案，但是使用高斯消元法得到的 G 通常是不稀疏的，使得编码复杂度为 $O(n^2)$ ，其中码长为 n 。当码长较长时，这种平方关系对 LDPC 码的实际应用具有极大的限制。所以从如何使得 LDPC 更可实现的角度，众多的研究者们开展了对 LDPC 码有效编码问题的研究。

方案路线一：1996 年，Mackay 提出了一种基于下三角形式结构的线性实现复杂度编码方案^[39]，Li Ping 等提出了一类基于半随机矩阵的编码方案^[40]，该方案要求 H 的右边是固定格式的双对角方阵，左边是稀疏的随机矩阵。此类码能够利用串行级联的方式使编码复杂度大大减小，完成线性时间编码。2001-2007 年期间，Y. Kou, Johnson, Z. Li, J. Lu, Freundlich 等分别对具有准循环和非规则等特性的 LDPC 码提出在线性时间内编码的方法^[41,42,43,44]。

方案路线二：利用 LDPC 码的 Tanner 图来进行迭代式编码，其特点是实现比较简单，但并不能确保一定可以成功获得编码码字。

方案路线三：利用 H 的稀疏特性来设计实现具有低复杂度的编码。1999 年，在由 Neal 提出的基于矩阵 RU 分解的编码方案^[45]中虽然减少了乘积项，但其复杂度依然相当高。2001 年，在 Richardson 等人提出的基于近似下三角形式的编码方案^[46]中，编码时仅进行列变换处理 H ，在保持其的稀疏性的同时得到近似下三角矩阵，因而可以既不损失性能又降低了复杂度，这种编码方法总体来说比较高效，

但要进行广泛应用还需要进一步使其复杂度和存储量降低。J. Lu 于 2010 年提出了复杂度恰好为线性^[47]的编码算法。

在上述三种方案路线之外, 1998 年, D. Divsalar 和 H. Jin 等还提出了一类可以使用累加器实现的重复累积码(RA 码)^[48]。Divsalar 等人在 1998 年提出了利用类 Turbo 码家族中的 RA 码实现有效解决 LDPC 码编码复杂度与性能之间的矛盾^[49]。RA 码性能优良, 同时编码复杂度低、结构简单。2001 年, H. Jin 将 RA 码推广得到非规则 RA 码(IRA 码)^[50], 并证明了其优越的性能和较低的编码复杂度^[51]。后来, W. Lin 等人对多进制 IRA 码的编码与构造进行了深入的研究^{[52][53]}。

1.3.3 LDPC 码的译码

在译码方面, 主要的研究方向是尽可能简化 LDPC 码的译码算法使得设计出的译码器具有较小的实现复杂度。LDPC 码的译码算法主要可以分为基于硬判决的比特翻转和基于软判决的置信传播两大类, 通常又被称为 BF 算法和 BP 算法。

目前主要的硬判决译码算法有 BF 算法、加权比特翻转算法(WBF)、改进加权比特反转(MWBF)算法^[54]、RRWBF 算法^[55]、NBP-WBF 译码算法^[56]、LP-WBF 译码算法^[57]以及一步大数逻辑译码算法(MLG), 加权大数逻辑译码算法(WMLG)。这些算法硬件实现简单, 但纠错能力不强。

软译码算法主要有 APP 译码和基于迭代结构的 BP 算法。Gallager 提出的 MPA 算法, 也称 BP 算法, Mackay 在 Gallager 和 Tanner 等人工作的基础上, 提出了性能接近香农限的和积译码算法。之后, 如何通过简化校验消息更新算法来实现和积算法, 进而在性能和复杂度两者之间取得平衡最优化成为 LDPC 译码算法研究的主要方向。Fossorier^[58]和 Eleftherou^[59]分别提出了在概率域和 LLR 域内校验消息更新简化算法: APP-Based 算法和 BP-Based 算法, 其中 BP-Based 算法也称作 MSA 算法^[60]。将软判决消息当作变量消息进行迭代时和积算法被简化为迭代 APP 算法, 这样以牺牲微量性能的代价来换取复杂度的降低。J. Chen 针对 MSA 算法中将双曲正切运算近似为最小和运算造成的性能损失, 提出了两种修正算法——Normalized BP-Based 算法和 Offset BP-Based 算法^[61]。X. Wu 将前者进一步改进成了自适应 Normalized BP-Based 和自适应 Offset BP-Based 算法^[62]。

1.3.4 LDPC 应用现状

工业界陆续有 LDPC 编译码芯片产生, 许多公司都相继推出编译码解决方案^[63]。美国的 Flarion Technology 公司开发了名为 Vector LDPC 的编译码器产品; Comtech AHA 公司推出了一种 LDPC 码编译码器内核, 该内核支持多种编码、调制格式及数据速率, 可自适应变化的信道条件, 其编译码吞吐量约为 30Mbps。另外, Marion 公司推出了基于 ASIC 的 LDPC 解决方案。VOCAL Technologies Ltd 推出了一种旨在节能的下行链路和上行链路分别采用 LDPC 码和 Turbo 码的

LDPC/Turbo 不对称解决方案。据称,该方案可使应用在 WLAN 的移动终端电池寿命增加 3 倍以上^[65]。此外,还有意法半导体等公司目前已经拥有自主知识产权的 IP 核和相应编译码芯片。

一直以来,在国内外通信协议标准制定的过程中,LDPC 码受到了相当大的关注。除了 IEEE802.16e 和 802.11n^[66]等标准将其当作信道编码方案外,由 ETSI(欧洲电信标准协会)于 2004 年发布的第二代欧洲数字电视卫星传输标准 DVB-S2^[67]标准也将 LDPC 码作为信道编码的备选编码方案。此外,LDPC 码目前也成功地被运用于无线通信、数字用户线(DSL)^[68]、图像压缩传输^[69]以及数字图像水印等领域。由于 LDPC 码良好的译码性能和突出的抗突发能力,作为 4G 通信标准中的纠错码技术得到越来越多的应用。DTMB 标准^[70]、CMMB 标准^[71]也都将其作为 FEC 系统中重要的编码方式。

1.4 本文主要内容及安排

目前在实际的通信系统中为了提高系统的可靠性和有效性,通常采用自适应的信道编码和自适应调制相结合的方法,来对抗无线信道变化所带来的不利影响。码率兼容的信道编译码器是实现高速率低复杂度的有效途径。本文主要基于实验室实际项目的需求,研究能够兼容 1008 码长、1/2 码率和 1024 码长、3/4 码率的 LDPC 编译码器的设计实现。为了达到系统要求,如何实现对资源占用和时延的折中,是本文在具体的设计实现过程中需要考虑的重点和难点。围绕如何设计实现满足需求的 LDPC 编译码器这一主题,本文主要内容从接下来的几个章节来开展:

第二章,系统介绍 LDPC 码的相关基础知识,包括线性分组码、LDPC 码和 QC-LDPC 码的相关概念,并结合 IEEE802.16e 标准,说明 QC-LDPC 的扩展构造方法;

第三章,系统总结各种编译码算法,并简要介绍了打孔算法的原理,然后在此基础上分析对比其性能和复杂度;

第四章,运用 Matlab 软件对译码算法进行定点仿真,为硬件实现时设定参数提供了参考和依据;

第五章,结合实际项目需要,针对 IEEE802.16e 协议标准中的一种 QC-LDPC 码结构,设计实现了码率兼容的 FPGA 编译码模块,并结合软件仿真对其功能和性能进行验证。

第六章,总结全文,并对本文中的一些局限性和 LDPC 编译码的实现研究方向进行说明。

第二章 LDPC 码基础

本章首先简要介绍了线性分组码的基础知识，然后给出了 LDPC 码的相关基本概念，包括 LDPC 码的定义和 Tanner 图表示，最后在介绍完 QC-LDPC 码概念的基础上，说明了具有基于 QC-LDPC 的扩展构造方法及其在 IEEE802.16e 协议标准中的应用。

2.1 线性分组码的基本概念

2.1.1 线性分组码的定义

分组码在构造时是将输入的 k 个信息码元分为一组，然后由这组码元按照特定的规律加入冗余码元构成 $n = k + r$ 位一组的码字。如果校验码元是根据信息码元的线性运算产生的，则此类分组码叫做线性分组码。

线性分组码一般用符号 (n, k) 表示，其中 n 表示编码输出的码元长度， k 表示输入的信息码元长度，等于输出码字中信息码元的位数。 $r = (n - k)$ 位码元是在编码过程中人为加入的冗余码元，又称为校验码元或监督码元。

2.1.2 线性分组码的参数指标

码率和最小距离是线性分组码的两个最主要参数指标。

对于 (n, k) 线性分组码，用 $R = k / n$ 表示码字中信息码元所占的比例，称之为码率， R 直观体现了信道利用率，所以也叫传信率。 R 越小，表示冗余度就越大； R 越大，编码的效率也就越高。

码距是指两个码字 x, y 之间对应位上不同码元的位数，用 $d(x, y)$ 表示。以 d_{\min} 表示的最小距离是指码字集合中任意两码字间的最小距离， $d_{\min} = \min\{d(x, y)\}$ 。 d_{\min} 表示码的纠错能力， d_{\min} 越大，则码的抗干扰能力越强。

2.1.3 线性分组码的表征

线性分组码通常由生成矩阵 G 或校验矩阵 H 来表征。

1) 生成矩阵

(n, k) 线性分组码 C 是所有 n 维向量组成的向量空间 C_n 的一个 k 维子空间，于是，可以找到 k 个线性独立的码字对应关系 g_0, g_1, \dots, g_{k-1} ，使得 C_n 中的每个码字 c 都是这 k 个码字的一种线性组合，即 $c \in C_{n,k}$ ， $c = u_0 g_0 + u_1 g_1 + \dots + u_{k-1} g_{k-1}$ ，式中 $u_i = 0$ 或 $1, 0 \leq i \leq k$ 。由这 k 个线性独立的码字可以构造成如式 (2-1) 的 $k \times n$ 矩阵。

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} \quad (2-1)$$

式中 $g_i = (g_{i0}, g_{i1}, \dots, g_{i,n-1})$, $0 \leq i \leq k$ 。如果 $c \in C_{n,k}$ 是待编码的码元，则相应的编码码字可表示为：

$$c = u \cdot G = (u_0, u_1, \dots, u_{k-1}) \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = u_0 g_0 + u_1 g_1 + \dots + u_{k-1} g_{k-1} \quad (2-2)$$

显然 G 的每行生成一个 (n, k) 线性码 C ，所以，称 G 为 C 的生成矩阵。一个 (n, k) 线性码完全由上式中的生成矩阵 G 的 k 个行向量确定。

2) 校验矩阵

对于任意一个由 k 个线性独立的行向量组成的 $k \times n$ 矩阵 G ，都存在由 $(n-k)$ 个线性独立的 n 维向量 $(h_0, h_1, \dots, h_{n-k-1})$ 组成的线性空间 $H_{(n-k) \times n}$ ，使得 G 的行空间中任意向量与 H 的行向量正交，该矩阵被称为该码的校验矩阵。因此可以得到

$$G \cdot H^T = \mathbf{0} \quad (2-3)$$

其中， $\mathbf{0}$ 表示 $k \times (n-k)$ 阶的全 0 矩阵。

对于任意一个码字 c ，有

$$c \cdot H^T = (uG)H^T = u(GH^T) = 0 \quad (2-4)$$

常用 (2-4) 式来判断长度为 n 的向量是否为码集 C 中的合格码字。因此一个 (n, k) 线性码可完全由其校验矩阵确定。

2.1.4 系统码

当一个码字的前 k 位与消息完全相同，则称该类码为系统码。其结构形式如图 2.1 所示。

| | |
|-------|---------|
| k位信息位 | n-k位校验位 |
|-------|---------|

图 2.1 系统码的一种结构形式

由于系统码的码字前 k 位是输入的信息码元，故其生成矩阵 G 左边 k 列肯定能组成单位方阵 I_k ，因此系统码的生成矩阵可用分块矩阵表示为：

$$G = [I_k | P] \quad (2-5)$$

式中， I_k 为 $k \times k$ 阶单位方阵； P 为 $k \times (n-k)$ 阶阵。

相应地，可以通过对校验矩阵 H 各行实现初等变化，并将后 r 列变换为单位子阵，则：

$$H = [Q | I_r] \quad (2-6)$$

由 (2-5) 式中 G 生成的码称为系统码，否则称为非系统码。显然，在系统码的码组 $C = [c_{n-1} \ c_{n-2} \ \dots \ c_0]$ 中，前 k 位 $[c_{n-1} \ c_{n-2} \ \dots \ c_{n-k}] = [u_k \ \dots \ u_1]$ 是信息位，后 $(n-k)$ 位 $[c_{n-k-1} \ c_{n-k-2} \ \dots \ c_0]$ 称为码字的校验位。

由于生成矩阵 G 的每一行都是一个码字，所以 G 的每行都满足 $HC^T = 0^T$ ，则

有： $HG^T = 0^T$ 或 $GH^T = 0$ 。所以，线性分组码的 G 和 H 的行矢量彼此正交。由上述三式，可得

$$GH^T = [I_k | P][Q | I_r]^T = Q^T + P = 0 \quad (2-7)$$

所以，

$$P = Q^T \text{ 或 } P^T = Q \quad (2-8)$$

由此可得 $G = [I_k | P] = [I_k | Q^T]$ 或 $H = [Q | I_r] = [P^T | I_r]$ 。因而线性系统码的 H 和 G 之间可以通过上述表达式进行相互转换。

2.2 LDPC 码的基本概念

2.2.1 定义

LDPC 码，全称是低密度奇偶校验码，作为线性分组码的一员，相对于传统线性分组码，LDPC 的最大特点就在于其校验矩阵 $H = [h_{ij}]$ 是一个稀疏矩阵，即矩阵 H 中绝大多数为“0”，只有少部分为“1”。

2.2.2 Tanner 图表示

Tanner 图能够简单形象地刻画 LDPC 码的编译码特性，显然，LDPC 码的 Tanner 图表示与 H 表示是完全一一对应的^[72]。

Tanner 图包括所有的节点集合和连接它们之间的边，节点集合由两个不相连的变量节点集合 V 和校验节点集合 C 构成，并且任何子集内部各顶点之间都没有边连接。

码字中的码元表现为变量节点，对应 H 的列；校验方程表现为校验节点，对应 H 的行；边是不同集合间节点的连线，对应 H 中的非零元素，表示对应的校验方程中包含的变量节点，当且仅当 $h_{ij} = 1$ 时，相对应的变量节点 v_j 和校验节点 c_i 间有一个边连接。

如图 2.2 是一个 $(6, 3)$ 线性分组码的 Tanner 图，每个 c_i 的度为 4，每个 v_j 的度为 2， H 如式 (2-9) 所示。

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (2-9)$$

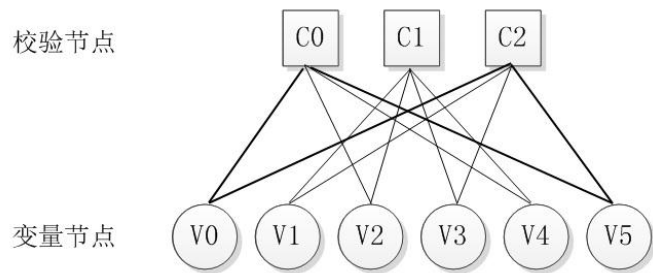


图 2.2 一个 $(6, 3)$ 分组码对应的 Tanner 图

H 的结构特点对码字性能具有决定性的影响，相对应地，在 Tanner 图中体现为环的存在会降低 LDPC 码的译码性能。

环：Tanner 图中，从 A 节点出发，经过不同的连续节点又回到 A 节点所经过的闭合路径，被定义为环 (Cycle)。环所经过的边的数目被称为环长，最小环长被称为围长(girth)。

根据定义，对码长有限的 LDPC 码而言，Tanner 图中通常至少有一个环，最小环长一定不小于 4 且环长一定是偶数。如图 2.2 中的粗线所示，Tanner 图中的最短环长为 4。

环的存在是制约 LDPC 码在 BP 译码算法下达到 MPA 算法的性能的主要原因，并且环长越小这种负面影响就越大，因为环的存在会导致 LDPC 码在迭代译码时，长度为 L 的环的存在决定了某一节点发出的信息在经过 $L/2$ 次迭代后又回到了该节点本身，从而造成自身信息的叠加，导致译码性能的下降。在迭代译码过程中，最小环长趋于无穷大的条件是码字无限长，又由于实际应用中 LDPC 码的码长必然有限，所以 Tanner 图中的环必然存在。通常在构造 LDPC 码时应尽量增加环长，并且避免出现短环来提高码字性能。

2.2.3 正则 LDPC 码

根据 H 中各行和各列中 1 的个数是否相同，可以把 LDPC 码可分为正则和非正则两大类。正则 LDPC 码定义最早是 Gallager 给出的，他指出正则 LDPC 码的校验矩阵 H 需要同时满足下面四个条件：

- 1) 校验矩阵 H 的每列都包含正好 λ 个“1”，即列重为 λ ；
- 2) 校验矩阵 H 的每行都包含正好 ρ 个“1”，即行重为 ρ ；
- 3) 两行或两列中不能有多余一个相同位置上有“1”元素；
- 4) λ 和 ρ 相对于码长和校验矩阵的行数来说很小。

如果 H 具有以上四个性质，则称由该 H 生成的码为正则 LDPC 码，用 (n, λ, ρ) 来表示该码字，其码长为 n ， H 中列重为 λ ，行重为 ρ 。否则，被称为非正则 LDPC 码。

性质 3) 又被称为行列约束，其使得 H 的任意不多于 d_v 列相加都不等于 0 (等价于线性无关)，因此 LDPC 码的最小距离为 $d_v + 1$ 。

对于正则 LDPC (n, λ, ρ) ，由于 d_c 和 d_v 相对于码长和 H 的行数来说很小，LDPC 码的 H 中非零元素占的比例也极小。LDPC 码的密度 r 即是 H 中非零元素所占的比例。

$$r = \frac{d_c}{n} = \frac{d_v}{m} \quad (2-10)$$

校验矩阵 H 是一个 $(n-k) \times n$ 的矩阵， d_c 和 d_v 的关系为

$$d_c = d_v n / (n - k) \quad (2-11)$$

LDPC 码的设计码率为

$$R_c = k / n = 1 - d_v / d_c \quad (2-12)$$

由前述定义知道，非正则 LDPC 码的 H 的各行行重、各列的列重不完全相同，所以不能像正则 LDPC 码那样通过常数 j 、 k 来表示。对于非正则 LDPC 码，通常用序列 $\{\lambda_1, \lambda_2, \dots, \lambda_{dl}\}$ 和 $\{\rho_1, \rho_2, \dots, \rho_{dr}\}$ 来表示其中边的度分布，部分之和等于全部：

$$\begin{cases} \sum_{j=1}^{dl} \lambda_j = 1 \\ \sum_{i=1}^{dr} \rho_i = 1 \end{cases} \quad (2-13)$$

边的度分布序列可以用多项式对来表示如下：

$$\begin{cases} \lambda(x) = \sum_{j=1}^{d_v} \lambda_j x^{j-1} \\ \rho(x) = \sum_{i=1}^{d_c} \rho_i x^{i-1} \end{cases} \quad (2-14)$$

式中， λ_j 和 ρ_i 分别表示度数为 j 变量节点和度数为 i 校验节点的入射边比例，

d_v 和 d_c 分别表示最大列重和最大行重。

这时的设计码率 $R_0(\lambda, \rho) = 1 - \int_0^1 \rho(x) dx / \int_0^1 \lambda(x) dx = 1 - \left(\sum_{i=2}^{d_c} \rho_i / i \right) / \left(\sum_{j=2}^{d_v} \lambda_j / j \right)$ 。

对于正则 LDPC 码

$$\begin{cases} \lambda(x) = x^{d_v-1} \\ \rho(x) = x^{d_c-1} \end{cases} \quad (2-15)$$

因此规则 LDPC 码又可以看作是 LDPC 码的一种特例。

如果用 $\int \lambda$ 和 $\int \rho$ 分别表示 $\int_0^1 \rho(x) dx$ 和 $\int_0^1 \lambda(x) dx$ ，在 LDPC 码 H 满秩条件下，码率为

$$R_c(\lambda, \rho) = 1 - \int \rho / \int \lambda \quad (2-16)$$

2.3 QC-LDPC 码

优秀的 LDPC 码应该在具有较好的译码性能的同时编译码器实现复杂度较低，基于循环移位阵的 QC-LDPC 码正是这样一类优秀的码型。经过优化后的 QC-LDPC 码译码性能卓越，其自身的良好结构使得编译码器的实现复杂度也非常低。

由于 QC-LDPC 码是一类具有准循环码特性的 LDPC 码，所以下面给出准循环码的相关概念和特性。

2.3.1 QC-LDPC 码基本概念

若方阵中除去首行外的各行都可由其上一行经循环右移一位得到，并且首行是尾行经循环右移一位得到的，那么这个方阵称为循环移位矩阵（CSM）。特别地，

若方阵可由单位矩阵经循环右移 n 位后得到，则该矩阵被称为循环移位单位阵（CSI）。

此外，如果将 $n \times (n-1)$ 个相同大小的零矩阵和 n 个大小相同的 CSI 拼接得到方阵，并且其行重、列重都等于 1，那么此方阵被称为准循环移位单位阵（QCI）；

一般地，如果将 $n \times n$ 个大小相同的 CSM 或零矩阵拼接得到一个方阵，那么此方阵被称为准循环移位阵（QCM）。可以看出，QCI 是一种特殊的 QCM，可以证明 QCM 的加、减、乘、求逆运算（如果存在）所得仍是 QCM。

如果一个 LDPC 码的 H 由且仅由大小相同的 CSM 或零阵拼接而成，则此 LDPC 码称为 QC-LDPC 码。QC-LDPC 码的准循环特性利于高效编译码的实现，使得它们广泛的应用于实际系统中，本文所使用的码型就是一类 QC-LDPC 码。

2.3.2 QC-LDPC 的矩阵扩展

QC-LDPC 码的 H 由基本阵 H_{base} 扩展而来， H_{base} 定义了 CSI 在 H 中的位置 (i, j) 和循环移位量的大小 C_{ij} ，如式（2-17）所示。假设校验矩阵 H 的维数为 $(n-k) \times n$ ，基础矩阵的维数为 $(n-k)_z \times n_z$ ，则有 $n-k = (n-k)_z \cdot z$ ， $n = n_z \cdot z$ ， z 是大于 1 的整数。

QC-LDPC 码的 H 扩展生成方法是：将 H_{base} 中的循环移位因子有效（ $C_{ij} \geq 0$ ）位置用 $z \times z$ 的循环移位阵 $P_{i,j} = I(C_{ij})$ 代替，循环移位因子无效（ $C_{ij} = -1$ ）位置用零矩阵 O 代替。

$$H_{base} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & -1 & C_{15} & -1 \\ C_{21} & -1 & C_{23} & C_{24} & -1 & C_{26} \\ -1 & C_{32} & -1 & C_{34} & C_{35} & C_{36} \end{bmatrix} \quad (2-17)$$

2.4 IEEE 802.16e 标准中的 LDPC 码

由 Wimax 论坛推动着的 IEEE802.16 协议标准是面向宽带无线接入网的 MAC 层和物理层的技术协议标准，可同时支持固定和移动宽带无线接入的系统由 IEEE 802.16e 协议作出具体规定，为了满足低误码率数据传输的要求，将 LDPC 码列为其中的一种信道编码方案^[73]。

在 IEEE 802.16e 中给出了六种基本阵 H_{b_m} （见附录 A），分别对应 1/2、2/3 A、2/3B、3/4 A、3/4B 和 5/6 四种码率和不同数据宽度的情况。 H_{b_m} 的每个元素 $c(i, j)$ 对应 $n = 2304$ 码长时相应位置上 CSI 的右移位。 $c(f, i, j)$ 用来决定该码率情况下其它码长时对应的右移位。基阵共有 $n_b = 24$ 列，扩展因子 $z_f = n / 24$ ，其中码长为 n ， $z_0 = 96$ 对应码长 $n = 2304$ 。

对于码率为 1/2，码率为 3/4 的 A、B 码，码率为 2/3 的 B 码和码率为 5/6 的码，相应于扩展因子 z_f 的右移位 $c(f, i, j)$ 的表达式为：

$$c(f, i, j) = \begin{cases} c(i, j), c(i, j) \leq 0 \\ \lfloor c(i, j)z_f / z_0 \rfloor, c(i, j) > 0 \end{cases} \quad (2-18)$$

其中， $\lfloor x \rfloor$ 表示对 x 向下取整。

对于码率为 $2/3$ 的 A 码，对应 z_f 的右移位数 $c(f, i, j)$ 的表达式为

$$c(f, i, j) = \begin{cases} c(i, j), c(i, j) \leq 0 \\ \text{mod}(c(i, j), z_f), c(i, j) > 0 \end{cases} \quad (2-19)$$

IEEE802.16e 协议中的 LDPC 码是由基于基本阵扩展构造出来 H 来对应编码生成的。每个 LDPC 码由 $m \times n$ 的 H 定义，其中 m 表示校验信息码长， n 表示码长。矩阵 H 定义如下：

$$H = \begin{bmatrix} P_{0,0} & P_{0,1} & P_{0,2} & \cdots & P_{0,n_b-2} & P_{0,n_b-1} \\ P_{1,0} & P_{1,1} & P_{1,2} & \cdots & P_{1,n_b-2} & P_{1,n_b-1} \\ P_{2,0} & P_{2,1} & P_{2,2} & \cdots & P_{2,n_b-2} & P_{2,n_b-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ P_{m_b-1,0} & P_{m_b-1,1} & P_{m_b-1,2} & \cdots & P_{m_b-1,n_b-2} & P_{m_b-1,n_b-1} \end{bmatrix} = P^{H_b} \quad (2-20)$$

其中， $P_{i,j}$ 是一个 $z \times z$ 的置换矩阵或零阵， $m = z \times m_b$ ， $n = z \times n_b$ ， $n_b = 24$ ， z 为大于等于零的整数。矩阵 H 由一个大小为 $m_b \times n_b$ 的基本矩阵 H_b 扩展得到，扩展方法如上节所述。 H_b 分为 H_{b1} 和 H_{b2} 两部分， $H_b = [(H_{b1})_{m_b \times k_b} \mid (H_{b2})_{m_b \times m_b}]$ 。其中， H_{b1} 对应于系统比特， H_{b2} 对应于校验比特。 H_{b2} 可以继续分为 h_b 和 H'_{b2} 两部分。其中 h_b 是奇重向量， H'_{b2} 为双对角线结构（在 $i = j$ ， $i = j+1$ 处为 1，其它为 0， i 和 j 表示元素在矩阵中的行和列）。

第三章 LDPC 算法原理

3.1 LDPC 编码算法

LDPC码的基本编码原理和普通的线性分组码完全相同：本质上来源于 $c = u \cdot G$ 或者 $c \cdot H^T = 0$ 。但不同的编码算法在硬件实现时的时延特性和资源占用情况会有明显的差异，本节将分别介绍几种经典的LDPC编码算法。

3.1.1 传统编码算法

同普通的线性分组码一样，LDPC 码的传统编码算法是先把 H 转化为 G ，再按照传统线性分组码的生成方法进行编码：利用信源产生的信息码元和 G 相乘，得到相应的码字， $c = s \cdot G$ ，其中 c 为 LDPC 码编码后的码字， s 为信息码元， G 是大小为 $(k \times n)$ 的生成矩阵。

通常由校验矩阵得到生成矩阵 G 的方法如下：假设一个 $(r \times n)$ 校验矩阵 $H = [C_1, C_2]$ ，其中 C_2 是一个 $r \times r$ 方阵，并且在二元域上是可逆的。对 H 进行高斯消元，然后列置换使 H 转化为 $H_0 = [Q \ I_{r \times r}]$ ，转置 Q 可得到 P ， $P = Q^T$ 。

$$G^T = \begin{bmatrix} I_k \\ C_2^{-1} C_1 \end{bmatrix} \quad (3-1)$$

LDPC 码的 G 通常并不具有低密度特性，因此这种方法编码实现时，通常需要占用大量的存储空间来存储 G 矩阵。

当 H 的秩小于行数 m 时， C_2 不可逆，此时需要先利用高斯消元法去除矩阵中行的线性相关性以使 H 满秩，之后再利用 H 编码，经过处理后的矩阵编码效率高于原矩阵。高斯消元中的求逆过程复杂度是 $O(m^2n)$ 。然而经过高斯消元后的 G 并不具有稀疏性，所以编码矩阵乘法的复杂度是 $O(n^2)$ 。在长码情况下，大量的运算和存储限制了硬件实现的可能，所以这种方法只能用作性能仿真，而基本没有实现意义。

3.1.2 RU 编码算法

RU 编码算法的主旨思路是保持 H 的稀疏性，将 H 转化成近似下三角的形式。RU 算法可以分为预处理、编码和码字交换三个阶段。预处理过程对于一组码字只需要做一次，所以预处理部分不会带来编码器的复杂度增加。由于预处理只涉及初等变换中的行列交换，所以，经过预处理后的矩阵 H^* 仍然是稀疏的。

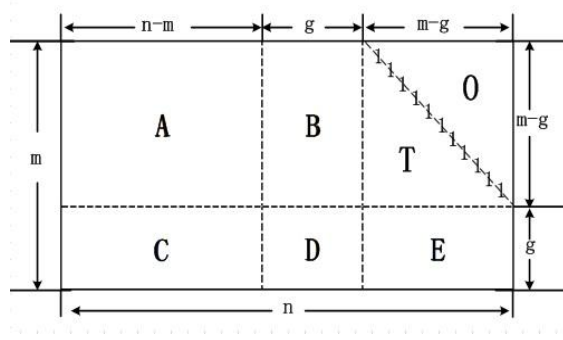


图 3.1 近似下三角结构的 LDPC 码校验矩阵

经过处理后的矩阵 H^* 也是 $m \times n$ 维的，由图 3.1 可知，矩阵 H^* 可以表示成如下形式：

$$H^* = \begin{pmatrix} A & B & T \\ C & D & E \end{pmatrix} \quad (3-2)$$

为了便于推导，将 H^* 记作 H ，事实上如果从校验方程式的角度看，使用 H^* 和 H 的区别仅只在于是否需要对 c 进行一次列交换，由于在 RU 编码的第三阶段（码字交换阶段）会对 c 进行从 H^* 到 H 的逆向列交换过程，所以这两次列交换过程在推导时可认为是完全透明的。

在式 (3-2) 两边同时左乘下三角矩阵 $\begin{pmatrix} I & O \\ -ET^{-1} & I \end{pmatrix}$ ，得到：

$$\begin{pmatrix} I & O \\ -ET^{-1} & I \end{pmatrix} \cdot H = \begin{pmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & O \end{pmatrix} \quad (3-3)$$

由于编码后码字 c 满足 $Hc^T = 0^T$ ，令 $c = (s, p_1, p_2)$ ，其中 s 表示信息位， p_1 和 p_2 联合表示校验位，实际中分别对应矩阵 B 、 D 和 T 、 E 所在的列，则校验位的 p_1 长度为 g ， p_2 长度为 $m-g$ 。由校验方程式，可将式 (3-3) 作如下分解：

$$As^T + Bp_1^T + Tp_2^T = 0 \quad (3-4)$$

$$(-ET^{-1}A + C)s^T + (-ET^{-1}B + D)p_1^T = 0 \quad (3-5)$$

定义矩阵 $\varphi = -ET^{-1}B + D$ ，于是得到校验位 p_1 和 p_2 的表示式：

$$p_1^T = -\varphi^{-1}(-ET^{-1}A + C)s^T \quad (3-6)$$

$$p_2^T = -T^{-1}(As^T + Bp_1^T) \quad (3-7)$$

这样，通过预处理得到的矩阵 φ ，就可以通过信息码元 s 计算出校验码元 p_1 和 p_2 ，从而得到整个码字 c 。

对 RU 算法的复杂度分析如下： p_1^T 的计算可以拆分成几个步骤实现，如表 3.1 所示。其中步骤 1, 3, 4 都是稀疏矩阵与向量相乘，运算复杂度为 $O(n)$ ；步骤 5 是向量加法，复杂度为 $O(n)$ ；步骤 2 利用了 T 是下三角矩阵的特点，根据 $T^{-1}As^T = Y \Leftrightarrow As^T = TY$ 递推计算 Y ，复杂度也为 $O(n)$ ；只有步骤 6 中涉及到矩阵 φ^{-1} 是一个 $g \times g$ 大小的密集矩阵，运算复杂度为 $O(g^2)$ ，因此 p_1^T 的总运算复杂度为 $O(n + g^2)$ 。

表 3.1 p_1^T 的分解计算步骤

| 计算步骤 | 运算复杂度 | 注释 |
|--------------------------------------|----------|--|
| 1. As^T | $O(n)$ | 稀疏矩阵乘以向量 |
| 2. $T^{-1}As^T$ | $O(n)$ | $T^{-1}As^T = Y \Leftrightarrow As^T = TY$ |
| 3. $-ET^{-1}As^T$ | $O(n)$ | 稀疏矩阵乘以向量 |
| 4. Cs^T | $O(n)$ | 稀疏矩阵乘以向量 |
| 5. $-ET^{-1}As^T + Cs^T$ | $O(n)$ | 向量相加 |
| 6. $-\varphi^{-1}(-ET^{-1}A + C)s^T$ | $O(g^2)$ | 稀疏矩阵乘以向量 |

p_2^T 的计算同样可以拆分成几个步骤实现，如表 3.2 所示。 p_2^T 比 p_1^T 的计算简单，步骤 1, 2 都是稀疏矩阵与向量相乘，运算复杂度为 $O(n)$ ；步骤 3 是向量相加，运算复杂度为 $O(n)$ ；步骤 4 也是利用了 T 是下三角矩阵的特点，根据 $-T^{-1}(As^T + Bp_1^T) = Y \Leftrightarrow -(As^T + Bp_1^T) = TY$ 递推计算 Y ，复杂度也为 $O(n)$ 。因此 p_2^T 的总运算复杂度为 $O(n)$ 。

 表 3.2 p_2^T 的分解计算步骤

| 计算步骤 | 运算复杂度 | 注释 |
|-----------------------------|--------|---|
| 1. As^T | $O(n)$ | 稀疏矩阵乘以向量 |
| 2. Bp_1^T | $O(n)$ | 稀疏矩阵乘以向量 |
| 3. $As^T + Bp_1^T$ | $O(n)$ | 向量相加 |
| 4. $-T^{-1}(As^T + Bp_1^T)$ | $O(n)$ | $-T^{-1}(As^T + Bp_1^T) = Y$ $\Leftrightarrow -(As^T + Bp_1^T) = TY$ |

根据上述分析可知，具有近似下三角结构的 LDPC 码的编码复杂度几乎与码长成线性关系，最主要的原因就是 p_1^T 的步骤 2 和 p_2^T 的步骤 4 都利用了 T 的结构特点，将原本复杂度等于 $O(n^2)$ 的矩阵乘法运算简化成复杂度为 $O(n)$ 的递推运算。为了降低复杂度，必须使得 g 的值尽可能小，当 $g=0$ 时，校验矩阵就变成了一个下三角矩阵，如图 3.2 所示。假设信息序列为 s ，长度为 $n-m$ ，校验序列为 p ，长度为 m ，则可以利用式 (3-8) 进行递推编码，显然该 LDPC 码具有线性的编码复杂度。

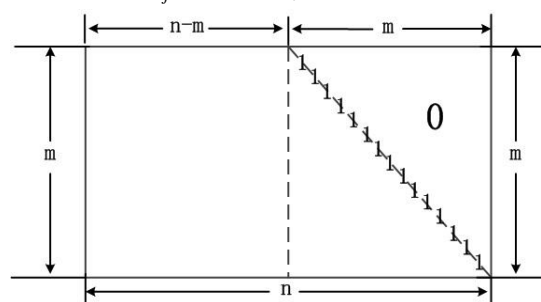
$$p_i = \sum_{j=1}^{n-m} h_{i,j} s_j + \sum_{l=1}^{i-1} h_{i,l+n-m} p_l \quad (3-8)$$


图 3.2 下三角结构的 LDPC 码校验矩阵

3.1.3 QC-LDPC 编码

IEEE802.16e 中定义的 LDPC 属于第 2.3 节所述的 QC-LDPC 码范畴，这类码的 H 完全由循环移位单位阵和零矩阵构成。

假设 H 的基本矩阵大小是 $m \times n$ 的，一般 $m \leq n$ ，循环移位扩展因子为 z 。 H 可以分成 H_s 和 H_p 两部分： $H = [H_s, H_p]$ ，其中 H_s 为 H 的对应信息码元的子矩阵， H_p 记为 H 的对应校验码元的子矩阵，又可以如式 (3-9) 所示将 H_p 分解。

$$H_p = [h_p \ H'_p] = \begin{bmatrix} h_0 & 0 & -1 & \cdots & -1 & -1 \\ h_1 & 0 & 0 & \cdots & -1 & -1 \\ h_2 & -1 & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 & -1 \\ \vdots & -1 & -1 & \cdots & 0 & 0 \\ h_{m-1} & -1 & -1 & \cdots & -1 & 0 \end{bmatrix} \quad (3-9)$$

式中 0 表示大小为 $z \times z$ 的单位阵，-1 表示大小为 $z \times z$ 的零矩阵。 h_p 是由列重为 3 的列组成， $h_p = [1, -1, \dots, -1, 0, -1, \dots, -1, 1]^T$ 这里的 1 表示循环移位因子为 1 大小为 $z \times z$ 的循环移位单位阵。于是， H 就具有如下的形式：

$$H = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,k-1} & 1 & 0 & -1 & \cdots & -1 & -1 \\ h_{1,0} & h_{1,1} & \cdots & h_{1,k-1} & -1 & 0 & 0 & -1 & \cdots & -1 \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \ddots & & \vdots \\ h_{x,0} & h_{x,1} & \cdots & h_{x,k-1} & 0 & -1 & \cdots & 0 & 0 & \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \ddots & & \ddots & \\ h_{m-1,0} & h_{m-1,1} & \cdots & h_{m-1,k-1} & 1 & -1 & -1 & \cdots & -1 & 0 \end{bmatrix} \quad (3-10)$$

其中 x 对应于 h_p 中单位阵所在的基本矩阵的行号。值得注意的是，由式(3-9)所定义的 H 矩阵对应的基矩阵靠右的方阵具有典型的双对角线结构，同时，在作循环移位扩展时，右边方阵循环移位阵中的移位因子几乎都为 0，即单位阵不移位，所以 H 右半边的方阵仍然具备双对角线结构。

QC-LDPC 编码算法具体方法如下：将输入信息比特记为 s ，并且将它分为 $k = n - m$ 组，每组包含 z 比特数据，则 $s = [s_0, s_1, \dots, s_{k-1}]$ 其中 s_i 都是长度为 z 的行向量。所以，编码后的码字就能记作

$$c = [s, p] = [s_0, s_1, \dots, s_{k-1}, p_0, p_1, \dots, p_{m-1}] \quad (3-11)$$

将 $Hc^T = 0$ 按行展开，得到：

$$\begin{cases} \sum_{j=0}^{k-1} h_{0,j} s_j + \Pi_1 p_0 + p_1 = 0 & (\text{第0行}) \\ \sum_{j=0}^{k-1} h_{i,j} s_j + p_i + p_{i+1} = 0 & (i \neq 0, x, m-1) \\ \sum_{j=0}^{k-1} h_{x,j} s_j + p_0 + p_x + p_{x+1} = 0 & (\text{第}x\text{行}) \\ \sum_{j=0}^{k-1} h_{m-1,j} s_j + \Pi_1 p_0 + p_{m-1} = 0 & (\text{第}m-1\text{行}) \end{cases} \quad (3-12)$$

令 $\lambda_i = \sum_{j=0}^{k-1} h_{i,j} s_j$ ，则上式可整理成

$$\begin{cases} \lambda_0 + \Pi_1 p_0 + p_1 = 0 & (\text{第0行}) \\ \lambda_i + p_i + p_{i+1} = 0 & (i \neq 0, x, m-1) \\ \lambda_x + p_0 + p_x + p_{x+1} = 0 & (\text{第}x\text{行}) \\ \lambda_{m-1} + \Pi_1 p_0 + p_{m-1} = 0 & (\text{第}m-1\text{行}) \end{cases} \quad (3-13)$$

其中， $\Pi_1 p_0$ 就是 p_0 循环右移 1 位的结果，将上式所有式子相加，得到：

$$p_0 = \sum_{i=0}^{m-1} \lambda_i \quad (3-14)$$

可见，一旦得到 p_0 ，并通过循环移位得到 $\Pi_1 p_0$ ，进而迭代即可求得 p_i ($1 \leq i \leq m-1$)。

3.2 LDPC 译码算法

LDPC 码的译码算法主要是硬判决和软判决译码两种。在 LDPC 码译码器的实现过程中，通常需要从纠错性能、硬件资源和吞吐率与信息传输速率三个方面考虑。然而它们之间又相互制约，所以在实际应用过程中，需要找到相对理想的折中方案。

3.2.1 硬判决译码算法

硬判决译码是指在译码过程中所有取值和算术运算都是在二元有限域 $GF(2)$ 完成，由于只需要模二和等复杂度较低的运算，所以在 VLSI 技术成熟之前，硬判决译码算法，尤其是比特翻转译码算法和加权比特翻转译码算法以其简单的实现方式获得了较为广泛的应用，当然代价是译码性能比较差。

3.2.2 BP 算法

与硬判决算法中传递硬判决结果 1 或 0 不同的是，软判决译码在校验节点和变量节点之间传递的信息是表示某比特取 1 或 0 概率的实数。通常情况下，软判决译码比硬判决译码具有更好的性能。BP 算法是由 Gallager 在提出 LDPC 时给出的，现在流行的译码算法几乎都是从 BP 算法演化而来的。

1) 概率域上的 BP 译码算法

本文主要讨论二元 LDPC 码的译码，信道模型为 AWGN 信道。在采用 BPSK 调制的情况下，设信号调制后的发送序列为 $\mathbf{x}=[x_1, x_2, \dots, x_N]$ ， $x_i \in \{+1, -1\}$ ，对应的 $a \in \{0, 1\}$ ，经过信道后的接收序列为 $\mathbf{y}=[y_1, y_2, \dots, y_N]$ ，即有 $y_i = x_i + n_i$ ，其中 n_i 是均值为 0、方差为 σ^2 的高斯白噪声序列。

LDPC 码的译码是基于 Tanner 图的迭代概率译码，每次迭代中的变量节点与校验节点沿着 Tanner 图上的边进行概率信息传递。图 3.3 为信息传递译码的 Tanner 图，方形表示校验节点，圆圈表示变量节点。

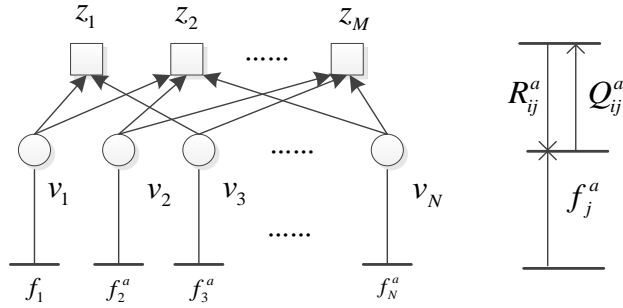


图 3.3 信息传递译码的 Tanner 图

以消息符号的上标表示迭代次数，以下是概率域上的 BP 算法具体推导。

Step1: 初始化。由变量节点传给校验节点软信息为

$$q_{ij}^0(0) = P(0) = \Pr(x_i = 1|y_j) = \frac{1}{1 + e^{-2y_j/\sigma^2}} \quad (3-15)$$

$$q_{ij}^0(1) = P(1) = 1 - q_{ij}^0(0) \quad (3-16)$$

Step2: 迭代

(1) 校验信息更新

对于全部校验节点 j 在第 l 次迭代时，求出第 j 校验节点传递给与其相邻的第 i 变量节点信息

$$r_{ij}^{(l)}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R(j) \setminus i} (1 - 2q_{i'j}^{(l-1)}(1)) \quad (3-17)$$

$$r_{ij}^{(l)}(1) = \frac{1}{2} - \frac{1}{2} \prod_{i' \in R(j) \setminus i} (1 - 2q_{i'j}^{(l-1)}(1)) \quad (3-18)$$

(2) 变量信息更新

对于全部的变量节点 i 以及与其相邻的校验节点 $j \in C_i$ ，计算出第 i 变量节点传递给第 j 校验节点的信息

$$q_{ij}^{(l)}(0) = K_{ij} P_i(0) \prod_{k \in C_i \setminus j} r_{ik}^{(l)}(0) \quad (3-19)$$

$$q_{ij}^{(l)}(1) = K_{ij} P_i(1) \prod_{k \in C_i \setminus j} r_{ik}^{(l)}(1) \quad (3-20)$$

其中 K_{ij} 为校正因子，使得

$$q_{ij}^{(0)}(0) + q_{ij}^{(0)}(1) = 1 \quad (3-21)$$

(3) 译码判决

对所有变量节点求出判决信息

$$q_i(0) = K_i P_i(0) \prod_{j \in C_i} r_{ij}^{(0)}(0) \quad (3-22)$$

$$q_i(1) = K_i P_i(1) \prod_{j \in C_i} r_{ij}^{(0)}(1) \quad (3-23)$$

其中 K_i 称为校正因子，使得 $q_i^{(0)}(0) + q_i^{(0)}(1) = 1$ ，当 $q_i(1) > q_i(0)$ 时， $c' = 1$ ，否则 $c' = 0$ 。

(4) 停止迭代

当 $Hc^T = 0$ 或者迭代次数达到初始设置值时迭代运算停止，否则继续从 (1) 迭代。

2) 对数域上的 BP 译码算法及其简化算法

在概率域上实现 BP 算法时，需要大量的乘除法及指数运算，这些运算除了复杂度高外，在码长较长的情况下在数值计算上会不稳定，影响收敛性，不宜进行量化处理。如果利用对数运算将乘除转换成加减，则可使得算法推导更简洁且利于硬件实现。

为了降低消息传递过程中需要传递两个概率带来的不便，通常用二者之间的某个函数来表示，使得所传递的消息中同时包含这两个概率信息，常用的函数有概率之差、概率之比和对数似然比 (LLR)。LLR 在消去数据归一化因子的同时使得乘除运算简化为加减运算，因而成为广泛选用的概率度量函数。下面分析 LLR-BP 算法的迭代情况，首先定义

初始消息概率的 LLR: $u_j = \log(f_j^0 / f_j^1)$;

变量节点概率的 LLR: $v_{ij} = \log(Q_{ij}^0 / Q_{ij}^1)$;

校验节点概率的 LLR: $u_{ij} = \log(R_{ij}^0 / R_{ij}^1)$;

$$f_j^0 = P(0) = P\{c_i = 0 | y_i\} = \frac{1}{1 + e^{-2y_j/\sigma^2}} \quad (3-24)$$

$$f_j^1 = P(1) = P\{c_i = 1 | y_i\} = \frac{e^{-2y_j/\sigma^2}}{1 + e^{-2y_j/\sigma^2}} \quad (3-25)$$

按照 BP 算法基本公式出发，推导在对数似然比下的初始消息和消息迭代公式：

根据 u_j 定义及 (3-24) 式和 (3-25) 式得到

$$u_j = \log(f_j^0 / f_j^1) = 2y_j / \sigma^2 \quad (3-26)$$

根据 v_{ij} 定义及 (3-22) 式和 (3-23) 式得到

$$v_{ij} = u_j + \sum_{k \in M(j) \setminus i} u_{kj} \quad (11) \quad (3-27)$$

由 (3-18) 式变换:

$$1 - 2r_{ij}(1) = \prod_{i \in R(j) \setminus i} (1 - 2q^{(1-1)}_{ij}(1)) \quad (3-28)$$

利用恒等式

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3-29)$$

$$\tanh\left(\frac{1}{2} \log\left(\frac{p_0}{p_1}\right)\right) = p_0 - p_1 = 1 - 2p_1, (p_0 + p_1 = 1) \quad (3-30)$$

上式结合 v_{ij} 和 u_{ij} 定义可得到下式

$$\tanh\left(\frac{u_{ij}}{2}\right) = \prod_{k \in \mathcal{N}(i) \setminus j} \tanh\left(\frac{v_{ij}}{2}\right) \quad (3-31)$$

整理得到:

$$u_j = 2y_j / \sigma^2 \quad (3-32)$$

$$v_{ij} = u_j + \sum_{k \in M(j) \setminus i} u_{kj} \quad (3-33)$$

$$\tanh\left(\frac{u_{ij}}{2}\right) = \prod_{k \in \mathcal{N}(i) \setminus j} \tanh\left(\frac{v_{kj}}{2}\right) \quad (3-34)$$

对式 (3-33) 中变量信息的更新计算中只包含实现简单的加法运算, 而在式 (3-34) 中校验信息的更新计算中却包含计算相当困难的双曲正切运算和乘法运算, 因而就产生了一些简化校验信息更新计算的简便算法。

首先整理校验消息计算式(3-34)可得

$$u_{ij} = \tanh^{-1} \prod_{k \in \mathcal{N}(i) \setminus j} \tanh\left(\frac{v_{ik}}{2}\right) \quad (3-35)$$

再将 $\tanh(x) = (e^x - 1) / (e^x + 1)$ 代入上式, 并利用 $\tanh(x)$ 函数的奇函数性质, 可得到式(3-35)化简形式

$$u_{ij} = \tanh^{-1} \prod_{k \in \mathcal{N}(i) \setminus j} \text{sign}(v_{ik}) f\left(\sum_{k \in \mathcal{N}(i) \setminus j} f(v_{ik})\right) \quad (3-36)$$

式 (3-36) 是经简化处理后的 LLR-BP 算法的校验信息更新计算公式。 $f(x) = \ln(e^x + 1) / (e^x - 1)$ 是具有自反性的偶函数, $f(f(x)) = x$ 。由式(3-36)可以看出该算法将式(3-35)中校验信息的计算过程分成了绝对值运算和符号运算, 并且将式(3-35)中的非线性乘法运算转化成线性的加法运算, 在有利于对数据的量化处理的同时大大降低复杂度。

3.2.3 最小和算法

在基于似然比的置信传播算法中, 运算 $\ln(e^x + 1) / (e^x - 1)$ 可以用查表的方法来实现。但是由于计算公式中包含两次 $\ln(e^x + 1) / (e^x - 1)$ 运算, 所以在硬件中不能在一个

时钟之内完成校验信息更新运算，并且，查表运算需要占用大量的存储空间，这在硬件实现时显然不是最优化方案。只包含求最小值和求和运算的最小和算法，可以看作是对 BP 算法的一种较好改进算法，能更好的用于硬件实现。

由于式(3-36)中函数 $f(x) = \ln(e^x + 1) / (e^x - 1)$ 的函数值恒为非负数，且它在 $x > 0$ 时具有单调递减性，因而有 $f(f(x) + f(y)) \leq f(\min(x, y))$ 。再根据 $f(x)$ 的自反性，即可得

$$f(f(x) + f(y)) \leq \min(x, y) \quad (3-37)$$

将其代到(3-36)中可得

$$u_{ij} = \prod_{k \in \mathcal{N}(i) \setminus j} \text{sign}(v_{ik}) \min_{k \in \mathcal{N}(i) \setminus j} (v_{ik}) \quad (3-38)$$

式（3-38）是最小和算法中校验信息的更新公式。式中用求最小值代替了 $\ln(e^x + 1) / (e^x - 1)$ ，所以整个过程中只有加法和求最小值运算，复杂度得到极大降低，便于硬件的实现。

3.2.4 归一化最小和算法

最小和算法是对 BP 算法的一种有效改进，它可以有效地降低算法实现复杂度。然而，在推导最小和算法的过程中用到了近似式 $f(f(x) + f(y)) \approx f(\min(x, y))$ ，从而有 $f\left(\sum_{k \in \mathcal{N}(i) \setminus j} f(|v_{ik}|)\right) \leq \min_{k \in \mathcal{N}(i) \setminus j} (v_{ik})$ 。这必然会导致译码性能上的损失。为了抵消引入这种近似所造成的影响，需要对最小和算法作一些修正，归一化最小和算法就是其中的一种修正算法。

归一化最小和算法的基本原理是在最小和算法得到的校验更新信息的基础上乘以一个小于 1 的数来抵消近似运算带来的恶化，式（3-39）为归一化后的校验信息更新公式。

$$u_{ij} = \eta \prod_{k \in \mathcal{N}(i) \setminus j} \text{sign}(v_{ik}) \min_{k \in \mathcal{N}(i) \setminus j} (v_{ik}) \quad (3-39)$$

式中的 η 被称为补偿因子，为了获得最佳的译码性能，补偿因子的取值应该随着信噪比、迭代次数的改变而改变，在应用中 η 通常是通过仿真来确定的固定值。

这种算法仅仅是在最小和算法的运算结果上乘上一个常数 η ，其复杂度与最小和算法相当，而其译码性能却要比最小和算法好得多，因此这种算法适合用于硬件的实现，本文的译码器设计就是采用这种译码算法。

3.3 打孔算法

在实际通信系统中，码率自适应通常是指根据不同的信道状态不断调整纠错码的码率以获得时变信道中最大的吞吐率。打孔是一种实现码率自适应的有效途径。打孔^[74]通常是通过部分校验码元作删余处理来使信道中实际传输的码长变短来使码率提高。由于码率兼容的 LDPC 码仅使用一组编译码器，相应的其硬件

复杂度较低，所以成为优先选用的差错控制实现技术。

实际中，打孔的具体操作是在发送端将那些可以通过 BP 算法恢复出来的校验码元删除，在接收端可以根据剩余码元信息恢复出被打孔的码元。在译码过程中，由于被打孔位置码元没有通过信道传输，译码器将打孔位置的 LLR 消息设为 0。传递非零消息的校验节点称为存活校验节点。当打孔处需要 k 次迭代恢复，则称该变量点为 k -SR 节点。未被打孔的变量点称为 0-SR。图 3.4 给出了 3-SR 节点的恢复树结构。

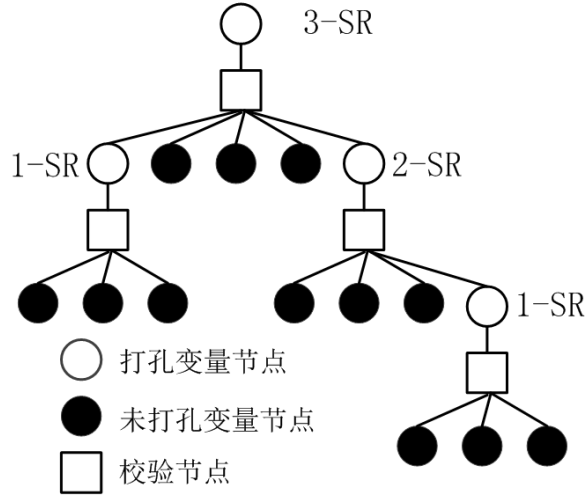


图 3.4 3-SR 节点恢复树结构图

对于相同的码率，不同的打孔方案，所获得的性能也不尽相同。常见的打孔算法有基于密度进化的打孔算法、基于高斯近似的打孔算法以及基于置信传播的打孔算法等，本文中使用的打孔算法是基于置信传播的。

第四章 译码器的定点量化方案

由于通信系统中的噪声值连续，译码信息也是连续的，但译码器则完全由数字系统构成，为了实现其高效性、准确性和可预见性等特点，需要对译码器用到的数据进行量化处理，常见的量化有均匀量化和非均匀量化两种。

本章通过 Matlab 仿真及理论分析研究了量化范围、量化级数、量化方式对译码性能产生的影响，并确定适合硬件实现的接收信息量化方案和译码中间变量的量化方案。

4.1 定点量化处理需要考虑的原则

- 1、从提高系统性能角度考虑，尽可能减少定点量化对性能的恶化；
- 2、从减少资源占用角度考虑，尽可能减少定点量化对资源的消耗。

由于 FPGA 数据处理是按单个比特来进行运算的，在设计时，信号定点量化比特数是工程中必须面对的问题，设计的好坏直接影响着系统性能的优劣。信号量化时的电平数越多，原模拟信号在其对应的定点数字信号中越能得到真实反映，信号的量化信噪比也会提高，但相应的硬件功耗与资源占用也会急剧增加。因此，这两者成为一对不可调和的矛盾，需要根据实际情况进行综合考虑。

4.2 量化译码背景

4.2.1 量化比特数的选择

量化比特数与译码算法精确性和译码算法实现的复杂度紧密相关，具体来说，如果要提高译码算法的精确性，提高数据的精度，显然可以通过增加量化比特数的方法。但是，利用这种方法译码算法实现的资源占用就要加以重视。量化比特数增加过多，硬件电路的资源消耗也会不断增加，译码算法实现的复杂度同样将会极大的增加；反之，如果量化比特数偏少，虽然减少了资源消耗和降低了硬件实现上的复杂度，数据精度却受到了限制，影响了整个译码算法的精确性，而且过少的量化比特数会使信息损失较多，导致译码器的误码率增大。所以，量化比特数的合理选择，是本节需要解决的关键问题。

译码器输入信号的量化和其中间变量的量化是本文中数据量化的两个主要方面。输入信号量化需要确定输入信号的量化区间和量化比特数，在量化过程中要保证的是量化的精度能够足够正确地体现出译码算法性能；而中间变量的量化则需要确定的是中间变量的量化比特数，它是基于输入信号的量化方案来确定算法运算各中间值的定点量化范围，如果中间变量数据溢出会导致整个译码算法效率急剧下降，所以运算过程中需要保证的是中间变量不溢出。

4.2.2 量化数据选择

在进行输入信号量化的方案设计中，必须综合考虑能够减少截位误差的合适的量化区间和能够保证量化精度的足够的量化比特两个方面。中间变量的量化范围也与译码性能有关，足够的量化比特可以降低截位所带来的影响。

4.3 归一化最小和译码算法的量化处理

对于归一化最小和译码算法，需要确定译码器接收变量的动态范围及量化级数、译码中间变量的动态范围和量化级数。

4.3.1 译码器接收变量的量化处理

接收信号的量化性能由量化范围和量化比特数共同决定。对于码率为 R ，信道服从高斯分布的情况下，将每符号传输能量归一化为 1，则加性高斯白噪声的均方根为

$$\sigma = \left(2R \times 10^{\frac{(E_b/N_0)_{dB}}{10}} \right)^{1/2}$$

接收信号关于 $x=0$ 对称，所以能够选择对称的量化范围，记为 $D=[-d, d]$ ，($d>0$)。记接收信号落入量化范围 D 的概率为 p_{in} ，由接收信号是混合高斯分布有：

$$p_{in} = 1 - \phi\left(\frac{d-1}{\sigma}\right) - \phi\left(\frac{d+1}{\sigma}\right), \phi(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$$

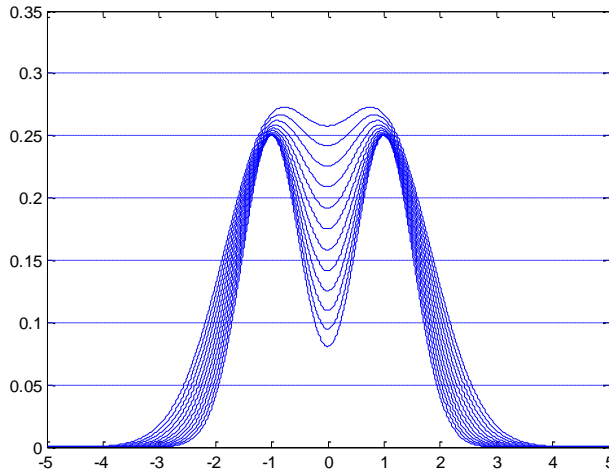


图 4.1 接收信号的概率密度曲线

图 4.1 给出了间隔为 0.2dB，信噪比从 0 到 2.4dB，码率为 0.5 时，接收信号的概率密度曲线。在量化中采用正态分布的“ 3σ 原则”，根据信噪比来确定接收比特的量化范围，从而提高量化精度。

虽然 BPSK 接收信号的大部分都落在 $[-4, 4]$ 之间，但是译码软信息的输入范围会大于此域，为了验证输入软信息范围对译码器的性能影响情况，进行如下仿

真：对同一软解调信息，分成三组，第一组不进行限幅，第二组对软信息数据进行 $[-4,4]$ 限幅，第三组对软信息数据进行 $[-8,8]$ 限幅，再将这三组信息输入同一个译码器进行仿真。

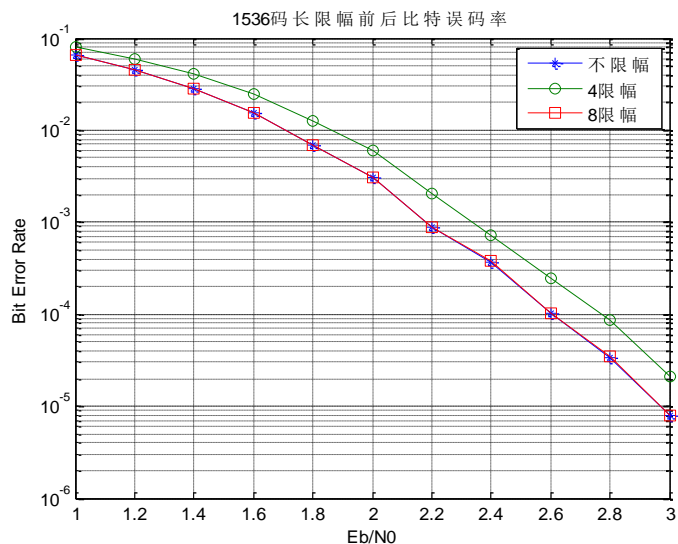


图 4.2 限幅对误码率的影响

如图 4.2 所示，对软信息进行 $[-4,4]$ 限幅后的译码性能有一定的恶化的，对软信息进行 $[-8,8]$ 限幅后的译码性能几乎与不限幅相同。

在此基础上，为了验证输入软信息量化对译码器的性能影响情况，进行如下仿真：对同一软解调信息 $[-8,8]$ 限幅后，分成三组，第一组不进行定点量化，第二组对输入信号 1.0 进行 4 比特定点量化，第三组分别对输入信号 1.0 进行 5 比特定点量化，再将这三组信息输入同一个译码器进行仿真。

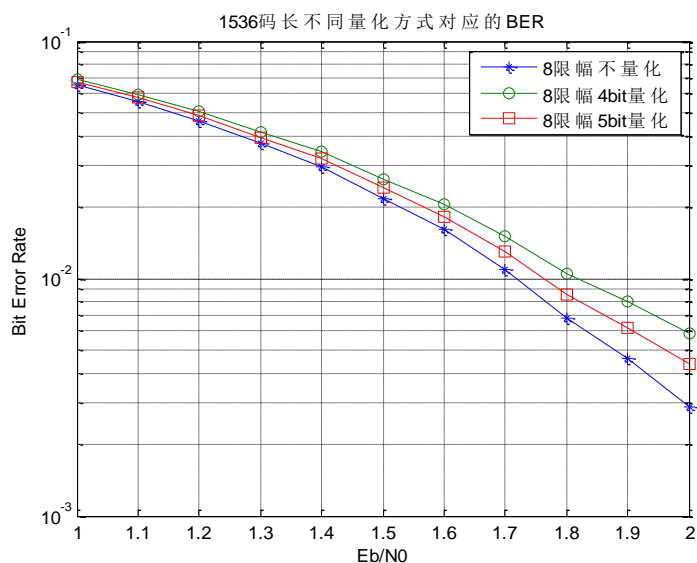


图 4.3 量化对误码率的影响

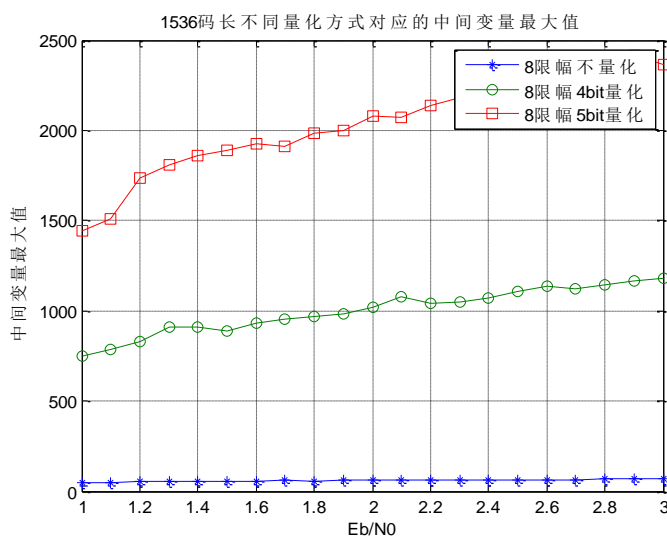


图 4.4 量化对中间变量最大值的影响

通过仿真发现低信噪比时，对软信息进行 4 比特量化和 5 比特量化时，译码性能非常接近连续译码性能，但在高信噪比时与连续译码性能有些差距。综合考虑 4 比特量化和 5 比特量化对中间变量最大值的影响，确定该译码模块输入软信息使用 4 比特量化，同时软信息输入最大值限幅到 $[-8.0, 8.0]$ 之间，这样实际输入软信息为：最小值 $V_{\min} = -8.0 \times 16 = -128$ ；最大值 $V_{\max} = 8.0 \times 16 = 128$ （这里使用 127）。最后得出输入软信息可使用 8 比特数据表示，其中 1.0 需要量化为 4 比特量化。

4.3.2 内部数据最大值仿真

由于 LDPC 译码器内部使用了大量的累加处理，势必会造成内部数据逐渐增大。由本章开始分析可知，如果设置译码器内部数据宽度过大，则会造成 FPGA 内部寄存器资源的巨大浪费，而如果设置内部数据宽度过小，又会面临译码过程中数据迭代溢出而造成译码错误的危险。

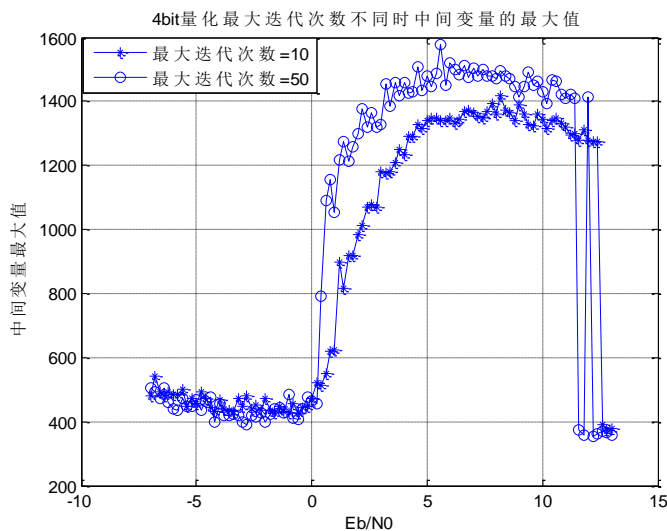


图 4.5 迭代次数对中间变量最大值的影响

为了确定译码器内部使用存储器数据宽度，在第 4.3.1 节译码器接收变量的量化处理的基础上，进行仿真：输入随机接收信号时，将软信息使用 4 比特定点量化表示，多次统计中间变量的最大值，仿真结果如图 4.5 所示。

结论：通过仿真，最终确定输入信号软信息为 8 比特输入，4 比特量化处理，同时内部使用 12 比特有符号整数设计。

第五章 LDPC 编译码器的 FPGA 设计

5.1 LDPC 编码器的设计

5.1.1 LDPC 编码器设计框图

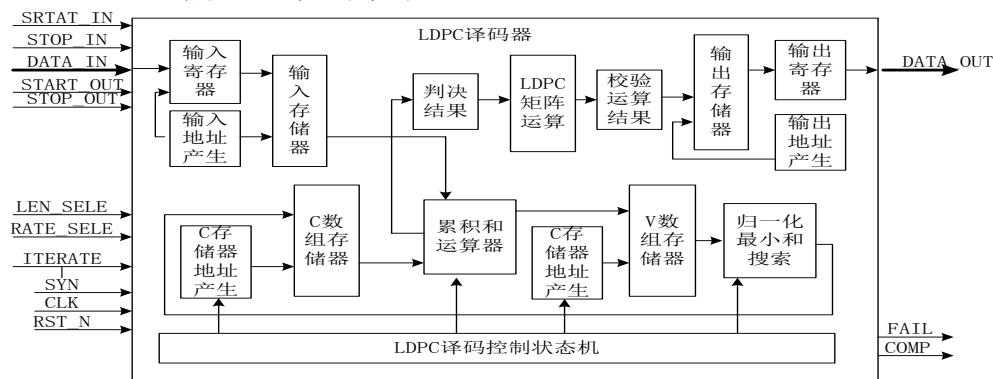


图 5.1 LDPC 编码器实现框图

由图 5.1 可知 LDPC 编码器由输入控制单元、LDPC 编码子矩阵运算单元、输出控制单元和控制状态机四部分构成。

1) 输入控制单元

输入控制单元由输入数据移位寄存器、输入地址产生器和输入存储器三部分组成。输入数据移位寄存器的作用是将输入的 8 比特串行码流进行重新组合，通过 LEN_SELET 信号来确定整合后的输入数据的宽度，使之与 LDPC 所选的子矩阵相匹配，并将组合后的数据存储入输入数据存储器中，工作原理如图 5.2 所示。输入地址发生器生成数据存入输入存储器的地址，同时片选输入存储器中的两个存储块。

当 LEN_SELET=0 时，输入信息长度为 504 时，8 比特输入共需用 63 个时钟来实现。校验矩阵中的子矩阵维数 $b=42$ ，所以需要将 8 比特数据流在输入数据移位寄存器中整合成 42 比特并存入输入存储器。当 LEN_SELET=1 时，输入信息长度为 768 时，8 比特输入共需用 96 个时钟来实现。校验矩阵中的子矩阵维数 $b=64$ ，所以需要将 8 比特数据流在输入数据移位寄存器中整合成 64 比特并存入输入存储器。

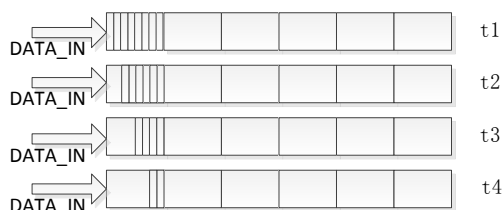


图 5.2 输入数据移位寄存器工作原理

以码长为 1008、1/2 码率的 LDPC 编码输入为例，8 比特数据（DATA_IN）并行输入，移位寄存到 5 个连续字节组成的移位寄存器中，t1 时刻开始依次输入 5 字节数据，取出 42 比特后，t2 时刻开始依次输入 4 字节数据，取出 42 比特后，t3 时刻开始依次输入 4 字节数据，取出 42 比特后，t4 时刻开始依次输入 4 字节数据，取出 42 比特后，再次按照“5 字节-4 字节-4 字节-4 字节”的顺序循环依次输入，以此实现 8 比特向 42 比特的整合转换。

另外，由于本文中 3/4 码率码长为 1024 的 LDPC 是由 1/2 码率 1536 码长打孔而来，校验矩阵中的子矩阵位数 $b = 64$ 为 8 的整数倍，所以输入数据移位寄存器的位宽为 64 比特。考虑到码率兼容及资源共用的问题，输入控制部分中的输入数据移位寄存器的位宽设计为 8 字节，当 LEN_SELET=0 时，使用 8 字节移位寄存器中的低 6 字节。

2) LDPC 编码子矩阵运算单元

LDPC 编码子矩阵运算单元完成信息数据的编码处理，它利用这些子矩阵的结构特点，使用串行的编码方式单周期完成一个子矩阵的乘法运算。这样可以最大程序的降低系统的资源使用量，并最大程序的提高数据的处理速度。

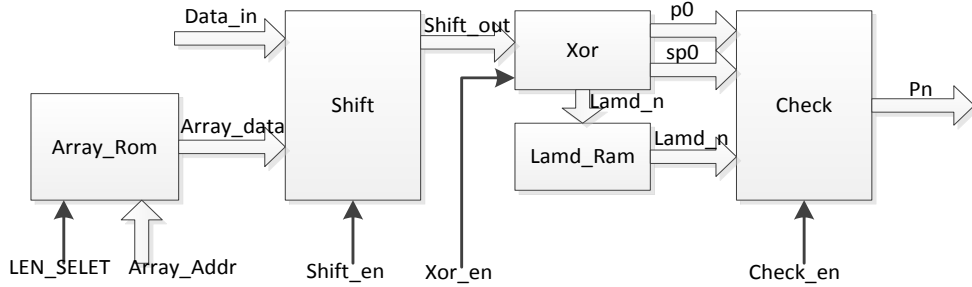


图 5.3 编码处理模块数据流结构图

编码处理单元由矩阵信息存储模块(Array_Rom)、循环移位模块(Shift)、异或累加模块(Xor)、 λ_n 缓冲 Ram(Lamd_Ram)和求校验输出(Check)5 个模块组成。

矩阵信息存储模块（Array_Rom）的输入信号码率选择（LEN_SELET）和校验矩阵地址（Array_Addr）共同决定输出校验矩阵中的校验子矩阵（Array_data）。循环移位模块（Shift）完成输入数据（Data_in）与校验子矩阵数据（Array_data）的乘法运算，RTL 图如图 5.4 所示，推导证明如式（5-1）所示，移位输出结果在异或累加和模块（Xor）中实现异或累加处理，分别得到校验信息中的 p_0 及其循环移位结果 sp_0 和中间变量 $\lambda_n (n=1,...,12)$ 。最后在求校验输出模块（Check）中依次求出除 p_0 之外的校验信息 $p_i (i=1,...,11)$ 。



图 5.4 循环移位模块 RTL 图

循环移位单位阵乘向量运算相当于将被乘向量进行循环移位。如式 (5-1) 所示，大小为 $n \times n$ 的循环移位单位阵的移位因子是 m ，与一个长度为 n 的向量相乘，其结果向量就是将输入向量循环上移 m 位。

$$\begin{bmatrix} 0 & 0 & \cdots & 1 & \cdots & 0 \\ 0 & \ddots & & \ddots & & \vdots \\ \vdots & \ddots & & \ddots & & 1 \\ 1 & \ddots & & \ddots & & \vdots \\ \vdots & \ddots & & \ddots & & 0 \\ 0 & \cdots & 1 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} a_m \\ \vdots \\ a_n \\ a_1 \\ \vdots \\ a_{m-1} \end{bmatrix} \quad (5-1)$$

3) 输出控制单元

输出控制单元由输出数据移位寄存器、输出地址产生器和输出存储器三部分组成。输出地址发生器片选输出存储器中的两个存储块的同时根据 RATE_SELET 信号来确定读出输出存储器中的数据地址，实现打孔功能。输出数据移位寄存器的功能与输入数据移位寄存器功能相反，这里不再赘述，详见第 5.3 节码率兼容的实现方法。

当 RATE_SELET=0 时，不对校验信息打孔，输出码率为 1/2 的码字；当 RATE_SELET=1 时，对校验信息打孔，使之由 1/2 码率打孔成 3/4 码率的码字输出，打孔后仅保留输出第 6、8、10、12 组校验信息。

4) LDPC 编码控制状态机

该状态机完成数据输入/输出接口的管理，同时完成对 LDPC 编码器的运算处理单元的管理，如图 5.3 中的 Shift_en、Xor_en 和 Check_en 都是状态机输出的控制信号。

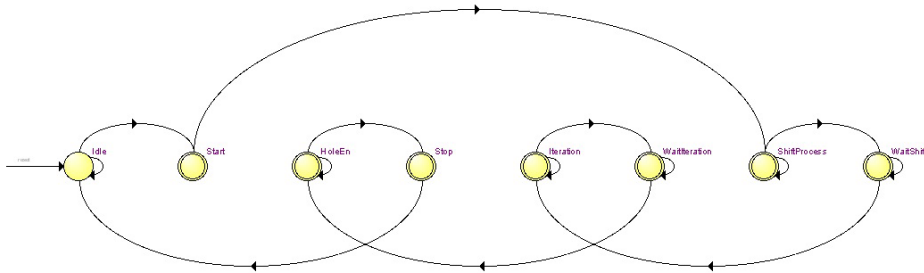


图 5.5 编码器状态转换图

如图 5.5 所示，编码器状态由空闲等待（Idle）开始，由同步信号触发后开始进入编码处理状态；编码处理由移位、移位等待、迭代累加和迭代累加等待四部分构成。当编码处理完成后，经过打孔状态直到停止，完成一次完整的编码周期。其中编码处理中的两次等待是为了防止数据处理时的拥塞，提高系统的稳定性。

5.1.2 LDPC 编码器顶层接口

该模块的接口信号分为控制信号、状态信号、输入信号和输出信号四部分组成，如表 5.1 所示。控制信号有输入时钟信号 `clk`，它为该模块提供系统时钟，`rst_n` 信号则完成该模块内部寄存器的异步复位处理，而 `syn` 信号为一次编码器状态机清零，同时完成 LDPC 编码器子矩阵运算器的启动处理，`len_sele` 信号选择输入数据长度，`rate_sele` 选择信号码率。状态信号为 `comp` 信号，它标识一个 LDPC 输入符号编码运算完成。输入控制部分包括 `start_in`、`stop_in` 和 `data_in[7:0]` 信号，它是通过串行的方式完成编码数据的输入。输出控制部分包括 `start_out`、`stop_out` 和 `data_out[7:0]` 信号，它是通过串行的方式完成编码数据的输出。

表 5.1 编码器接口使用说明表

| 接口名称 | I/O 类型 | I/O 位宽 (bit) | 使用说明 |
|------------------------|--------|--------------|--|
| <code>clk</code> | Input | 1 | 时钟信号 |
| <code>rst_n</code> | Input | 1 | 异步复位信号：调试时用，低电平有效 |
| <code>syn</code> | Input | 1 | 上升沿有效 |
| <code>len_sele</code> | input | 1 | 编码长度选择信号：为 0 时进行输入数据长为 504bit 编码，1 时进行输入数据长为 768bit 编码 |
| <code>rate_sele</code> | Input | 1 | 码率选择：为 0 时选择码率为 1/2，为 1 时选择码率为 3/4。 |
| <code>start_in</code> | input | 1 | 数据输入起始位：syn 信号有效 2 个 clk 之后可以产生该信号，持续一个 clk，高电平有效 |
| <code>stop_in</code> | input | 1 | 数据输入结束位：持续一个 clk，高电平有效 |
| <code>start_out</code> | input | 1 | 数据输出起始位：reset 信号有效 2 个 clk 之后可以产生该信号，持续一个 clk，高电平有效 |
| <code>stop_out</code> | input | 1 | 数据输出结束位：持续一个 clk，高电平有效 |
| <code>data_in</code> | input | 8 | 编码输入数据 |
| <code>data_out</code> | output | 8 | 编码输出数据 |
| <code>comp</code> | output | 1 | 编码状态完成标志：低电平有效 |

5.1.3 LDPC 编码器 I/O 时序图

LDPC 编码器 I/O 时序如图 5.6 所示，在时钟信号的驱动下，每一个 syn 信号将启动一次新的数据编码处理。因此当编码器接收到 syn 信号后，会完成器件内部相关处理寄存器的初始化，同时启动片内输入和输出、处理存储器的翻转，之后启动片内编码处理，同时启动处理上一次编码完成信号输出。而输入信号由外部输入，包括 start_in、stop_in 和 data_in[7:0]信号，但需要在下一次 syn 信号到来前完成输入信号的输入。同时，对于输出信号也相同，在编码器接收到 syn 信号两个时钟周期以后，可给编码器发送 start_out、stop_out 信号，则可以读出编码数据 data_out[7:0]。当新的一次编码完成后 comp 信号变低。

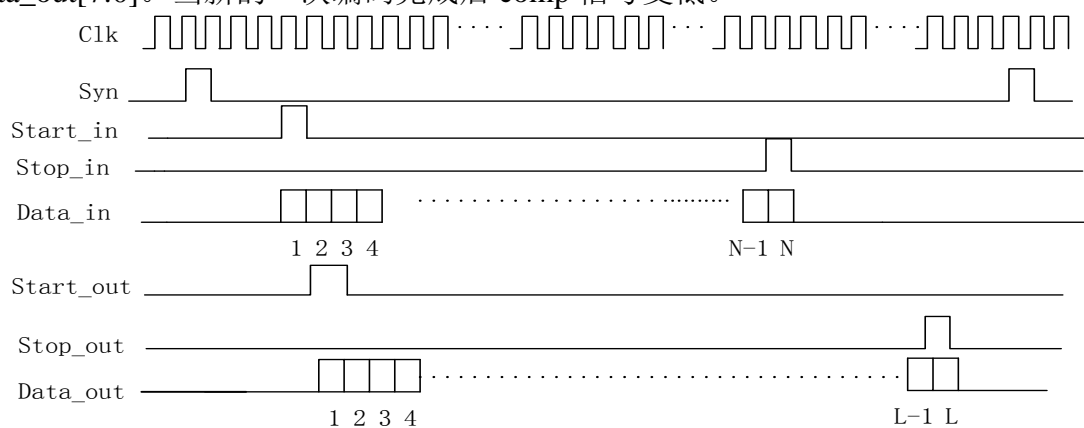


图 5.6 LDPC 编码器接口信号时序

5.1.4 LDPC 编码器性能

对于 LDPC 编码器，其在内部采用流水线操作，单个编码周期内，编码标志信号 comp 在 syn 脉冲上升沿后，经过 186 个时钟周期之后有效。当系统使用时钟 200M 时，完成一次编码时间为 0.94us，当使用时钟为 100M 时，编码时间为 1.86us。

在使用 Altera 公司的器件 EP2AGX125 对该模块进行编译时，模块占用该器件的 ALUT 共计 501 个等效需要占用的系统的寄存器数为 911 个，除此之外还需要占用器件内部的块 RAM 为 92160bit，以及 52 个 memory ALUT。

下表为该模块资源占用情况：

表 5.2 编码器资源占用

| 器件名称 | 占用资源 | 数目 |
|------------------|---------------------|------------|
| Altera EP2AGX125 | Combinational ALUTs | 1051 (702) |
| | Memory ALUTs | 52 |
| | Block memory bits | 11776 |

5.2 LDPC 译码器

5.2.1 LDPC 译码器设计框图

由图 5.7 可知 LDPC 译码器由输入控制、变量节点更新运算、校验节点更新运算、译码判决、输出控制和状态机六部分构成。

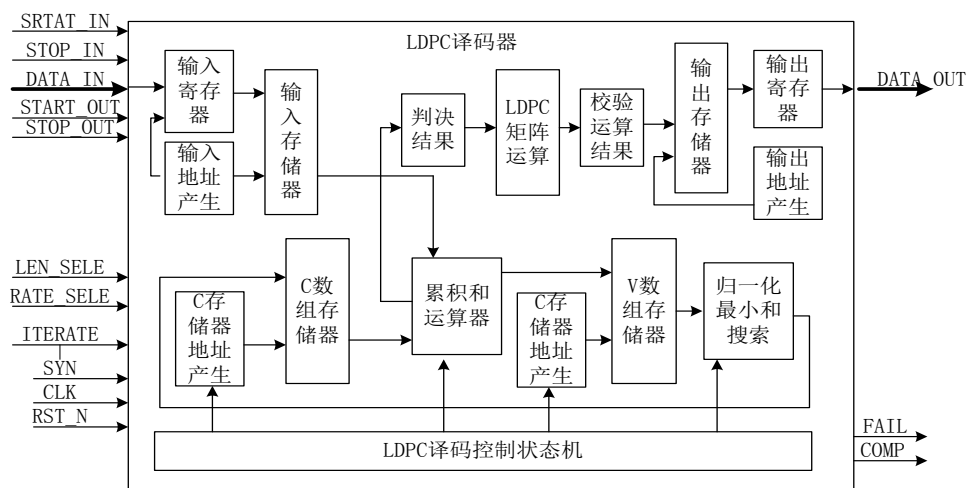


图 5.7 LDPC 译码器实现框图

1) 输入控制单元

译码器中的输入控制部分与编码器类似：第一：由输入数据移位寄存器、输入地址产生器和输入存储器三部分组成；第二：通过 LEN_SELET 信号在输入数据移位寄存器中按照与 LDPC 所选的子矩阵相匹配的格式将输入的 8 比特软信息数据进行重新组合，并将组合后的数据存储在输入数据存储器中，具体分析见第 5.3 节“码率兼容的实现方法”；第三：输入地址发生器生成数据存入输入存储器的地址，同时片选输入存储器中的两个存储块（乒乓操作）。

2) 变量节点更新运算单元

变量节点更新运算单元 HcMatrixAlgorithm 主要分为校验移位信息生成单元（包括校验存储器地址产生、校验移位数据存储器）、移位处理单元和译码累加运算单元三个部分。

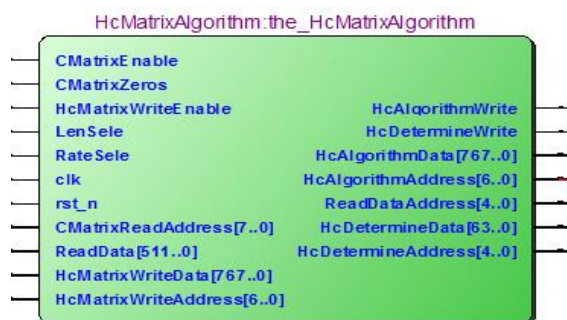


图 5.8 变量节点更新运算单元 RTL 图

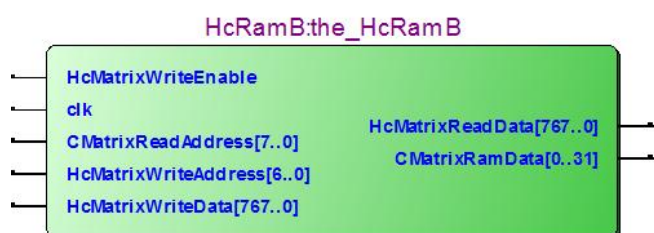


图 5.9 校验移位信息生成单元 RTL 图

校验移位信息生成单元（HcRamB），主要功能是针对不同编址位置上的输入校验信息产生相应的校验信息移位数据，以便于下一步的循环移位处理。该单元中的校验存储器地址产生单元（CAddressRam）是由单端口 ROM 构成，校验移位数据存储器（HcRam）由双端口 RAM 构成。

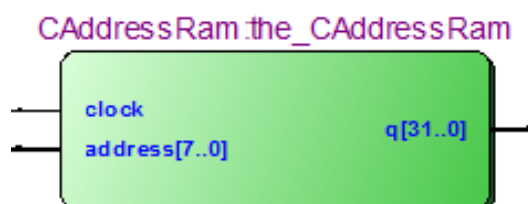


图 5.10 校验存储器地址产生单元 RTL 图

单端口 ROM 中初始化存储数据是对基本矩阵中代表非零矩阵的位置进行横向连续编址，存储数据内容及格式见第 5.3 节“码率兼容的实现方法”。

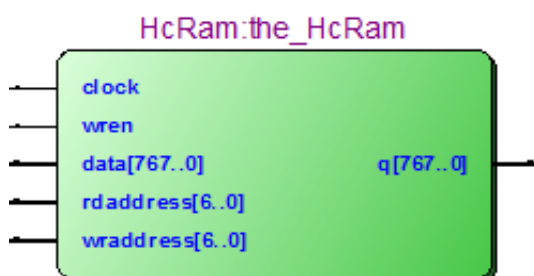


图 5.11 校验移位数据存储器 RTL 图

双端口 Ram 中的写操作相关信号（HcMatrixEnable、HcMatrixWriteAddress、HcMatrixWriteData）均由该模块外部提供，读地址由单端口 Rom 的输出部分数据提供。

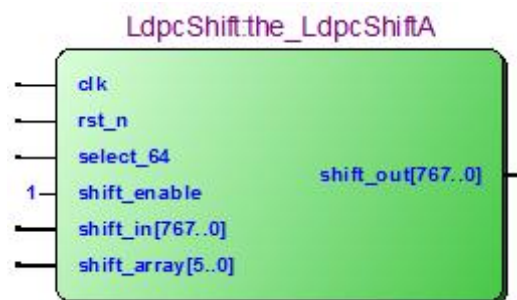


图 5.12 移位处理单元 RTL 图

移位处理单元 LdpcShift 主要功能是实现完成校验数据的移位。这里在使用硬件语言 Verilog 设计时，将位宽和字长均为参数形式，以便于程序的版本更新和移植操作。图 5.12 中所示的 select_64 表示码长选择时的选择信号，当 select_64=1 时，表示此时每行数据由 64 维循环移位单位阵构成；shift_enable 表示移位使能控制信号；shift_in 数据位宽为 $768 = 64 \times 12$ ，表示基本矩阵中的 12 行数据，每行数据由 64 维循环移位单位阵构成；shift_array 表示 shift_in 数据的移位位数，这里数据位宽是 6 比特，最大可以移位 64 位，需要说明的是，这里在设计时同样考虑到程序更新和移植的问题将其设置成参数的形式。当然 shift_out 数据位宽和 shift_in 数据位宽相同，均为 768 比特。

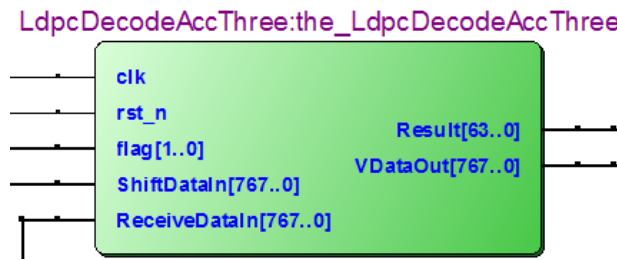


图 5.13 译码累加运算器单元 RTL 图

译码器累加运算单元主要功能是实现校验信息 C 数据的累加处理，同时完成对判决信息数据的生成和变量信息数据的更新输出。它利用校验子矩阵是准循环移位单位阵的结构特点，可在一个时钟周期内完成一次子矩阵数据的累加处理。单时钟周期处理一次子矩阵可以最大程序的降低系统的资源使用量，子矩阵的并行处理可以最大程度的提高数据的处理速度，这里也是部分并行处理的优势。需要指出的是，当 RateSele=1 时，码率选择为 3/4，在第一次累加运算时，对打孔处的校验信息进行补零处理。

由于累加器输入的数据宽度为 768 比特，为了便于编译器的编译操作，本设计采用分层的办法来处理。该模块 LDPCDecodeAccThree 由三层累加器组合而成。最上层 LDPCDecodeAccThree 由四个相同的 LDPCDecodeAccTwo 组成，每个 LDPCDecodeAccTwo 数据宽度为 192 比特，次上层的 LDPCDecodeAccTwo 由四个相同的 LDPCDecodeAccOne 组成，每个 LDPCDecodeAccOne 数据宽度为 48 比特，再下层的 LDPCDecodeAccOne 由四个相同的 LDPCAcc 组成，每个 LDPCAcc 数据宽度为 12 比特。

译码累加运算单元中的移位后数据 ShiftDataIn 来自移位处理单元 LdpcShift 的输出，对应于底层模块 LdpcAcc 中的 shift_data_in；接收数据 ReceiveData 来自顶层模块 HcMatrixAlgorithm 的软信息数据输入，对应于底层模块 LdpcAcc 中的 receive_data_in；判决数据 Result 通过顶层模块 HcMatrixAlgorithm 的 HcDetermineData 送入判决处理单元 LdpcDecodeCheck 进行判决，对应于底层模块

LdpcAcc 中的 result；更新后的变量节点信息数据 VDataOut 通过顶层模块 HcMatrixAlgorithm 的 HcAlgorithmData 送入校验节点更新运算 HvMatrixAlgorithm 进行校验节点信息的更新，对应于底层模块 LdpcAcc 中的 v_data_out。

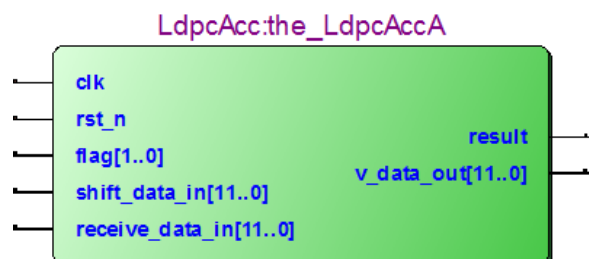


图 5.14 译码累加运算底层模块 RTL 图

需要指出的是，在底层模块 LDPCAcc 中，flag 表示 Rom 中的标志信号，当 flag=0 时，表示该数据为此列的第一个数据，此时的累加应当重新开始，当 flag=2 时，表示该数据为此列的最后一个数据，当 flag=1 时，表示该数据为此列的中间数据。当 flag=0 时，累加器的数据由接收数据 receive_data_in 和 shift_data_in 相加；flag=1/2 时，累加器的数据由上个时钟的寄存数据和新到的 shift_data_in 相加。

3) 校验节点更新运算单元

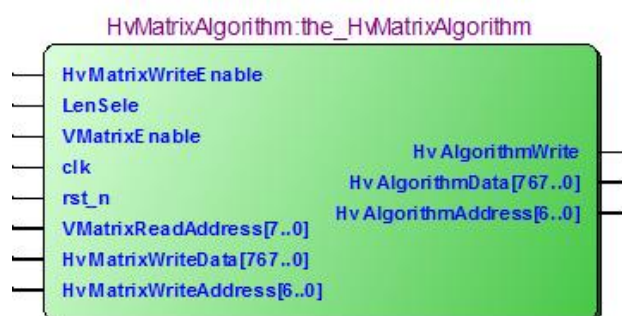


图 5.15 校验节点更新运算单元 RTL 图

校验节点更新运算单元 HvMatrixAlgorithm 主要分为变量移位信息生成单元（包括变量存储器地址产生、变量移位数据存储器）、移位处理单元和归一化最小和运算单元三个部分。

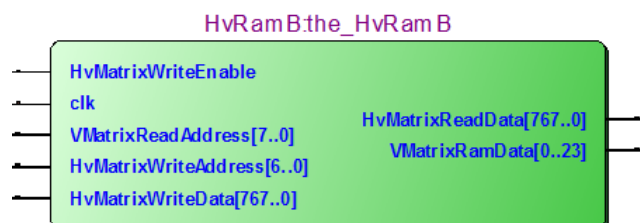


图 5.16 变量移位信息生成单元 RTL 图

变量移位信息生成单元（HvRamB），主要功能是针对不同编址位置上的输入变量信息产生相应的变量信息移位数据，以便于下一步的循环移位处理。该单元

中的变量存储器地址产生单元（VAddressRam）是由单端口 ROM 构成，变量移位数据存储器（HvRam）由双端口 RAM 构成。

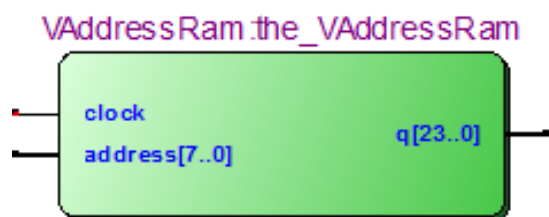


图 5.17 变量存储器地址产生单元 RTL 图

单端口 ROM 中初始化存储数据是对基本矩阵中代表非零矩阵的位置进行纵向连续编址，存储数据格式见第 5.3 节“码率兼容的实现方法”。

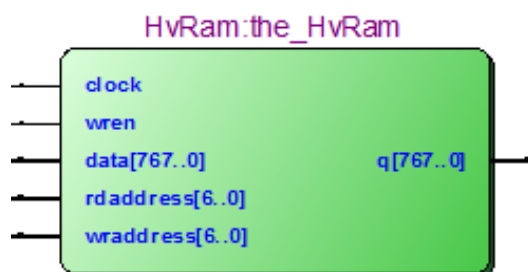


图 5.18 变量移位数据存储器 RTL 图

双端口 Ram 中的写操作相关信号（HvMatrixEnable、HvMatrixWriteAddress、HvMatrixWriteData）均由该模块外部提供，读地址由单端口 Rom 的输出部分数据提供。

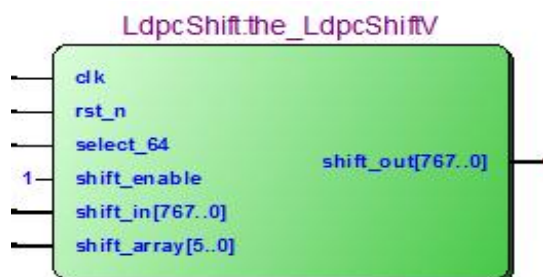


图 5.19 移位处理单元 RTL 图

移位处理单元 LdpcShift 主要功能是实现完成校验数据的移位。这里为模块的复用，所使用的模块和变量信息更新中的移位处理单元相同。

LdpcDecodeComparatorThree the_LdpcDecodeComparatorThree

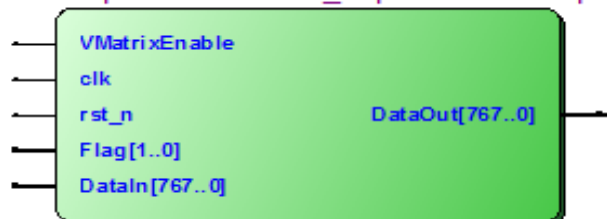


图 5.20 归一化最小和搜索运算单元 RTL 图

译码器归一化最小和搜索运算单元主要功能是实现变量信息 V 数据的归一化最小和搜索处理，以求得校验信息数据的更新输出。与译码器累加运算单元类似的，它利用校验子矩阵是准循环移位单位阵的结构特点，采用部分并行处理的方式，在系统资源使用量和系统时延之间进行了合理的折中，使得系统整体性能最佳。

归一化最小和搜索运算单元的整体设计构架和译码器累加运算单元类似，也是采用模块分层的办法来处理数据，以便于编译器的编译操作。该模块 LDPCDecodeComparatorThree 由三层累加器组合而成。最上层 LDPCDecodeComparatorThree 由四个相同的 LDPCDecodeComparatorTwo 组成，每个 LDPCDecodeComparatorTwo 数据宽度为 192 比特，次上层的 LDPCDecodeComparatorTwo 由四个相同的 LDPCDecodeComparatorOne 组成，每个 LDPCDecodeComparatorOne 数据宽度为 48 比特，再下层的 LDPCDecodeComparatorOne 由四个相同的 LDPCDecodeComparator 组成，每个 LDPCDecodeComparator 数据宽度为 12 比特。

归一化最小和搜索运算单元中的使能控制信号 VMatrixEnable 通过顶层模块 HvMatrixAlgorithm 的 VMatrixEnable 送入来控制该模块的使能状态，高电平有效时，该模块进行归一化最小和的搜索运算，对应于底层模块 LDPCDecodeComparator 中的 VMatrixEnable；标志信号 Flag 是变量移位信息生成单元（HvRamB）输出移位数据的同时输出的，用来指示节点是该行的位置，以利于控制变量信息数据的比较运算和校验更新数据的写操作。输入数据 DataIn 来自移位处理单元 LdpcShift 的输出，对应于底层模块 LDPCDecodeComparator 中的 DataIn；更新后的校验节点信息数据 DataOut 通过顶层模块 HvMatrixAlgorithm 的 HvAlgorithmData 送入变量节点更新运算 HcMatrixAlgorithm 进行变量节点信息数据的更新，对应于底层模块 LDPCDecodeComparator 中的 DataOut。

LdpcDecodeComparator:the_LdpcDecodeComparatorA

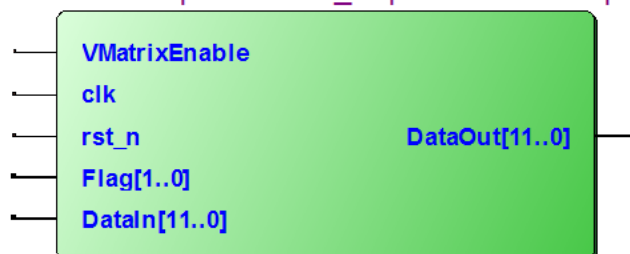


图 5.21 归一化最小和搜索运算底层单元 RTL 图

同样需要指出的是，在底层模块 LDPCDecodeComparator 中，flag 表示 Rom 中的标志信号，当 flag=0 时，表示该数据为此行的第一个数据，此时需要对归一化最小和运算处理单元中的相关数据进行初始化操作，例如比较位置计数器的清

零和最小值寄存器的置数等；flag=1 或 2 时，表示该数据不为此行的第一个数据，此时该数据参与进行相应的符号异或运算、最小值比较运算和次小值比较运算等操作。

4) LDPC 译码器判决处理单元



图 5.22 译码器判决处理单元 RTL 图

译码器判决处理单元 LdpcDecodeCheck 主要分为判决存储器地址生成、判决移位数据存储器、判决移位处理单元和判决累加和运算单元四个部分。

译码器判决处理单元 LdpcDecodeCheck 在 LDPC 译码器校验节点进行更新运算的同时进行。判决处理单元 LdpcDecodeCheck 通过对变量节点更新运算单元 HcMatrixAlgorithm 的判决数据 Result 输出进行判决运算，并在得到一组判决结果后再与校验矩阵相乘运算来判断该码字是否合法。如果该码字是一个合法码字，则通知状态机译码完成，不再进行译码迭代，否则继续迭代。如果迭代次数达到达到输入设定值，而译码校验结果还是一个非法码字，则本组译码失败，程序流程如图 5.23 所示：

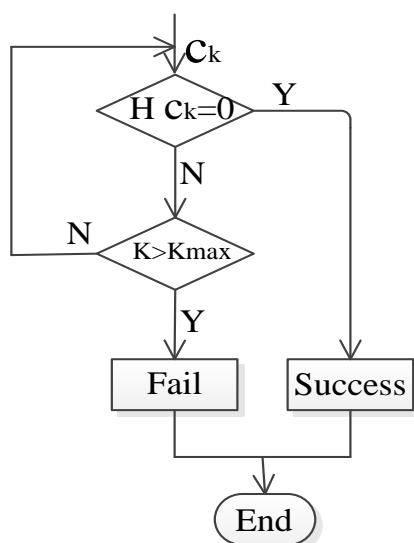


图 5.23 译码判决流程图

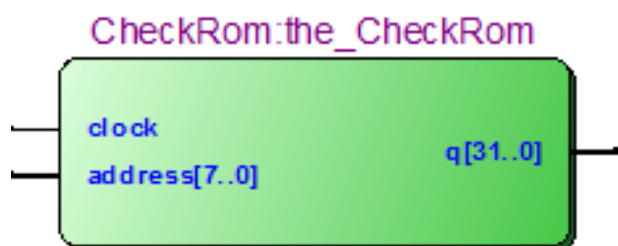


图 5.24 判决存储器地址生成单元 RTL 图

判决存储器地址生成单元 **CheckRom** 主要功能是在控制信号的作用下产生对应不同编址位置上判决信息对应的数据和控制信号，具体包括校验移位数据、校验读地址、译码数据写控制信号、译码数据写地址和译码数据写数据。该模块由单端口 ROM 构成，地址输入端数据由控制信号作用下的累加器和选择器输出提供，输出端数据包含上述中的数据和控制信号。单端口 ROM 中初始化存储数据格式见第 5.3 节“码率兼容的实现方法”。

双端口 Ram 中的写操作相关信号（HcDetermineWrite、HcDetermineAddress、HcDetermineData）均由该模块外部通过顶层模块端口提供。读地址由判决存储器地址生成单元 **CheckRom** 的输出部分数据提供。

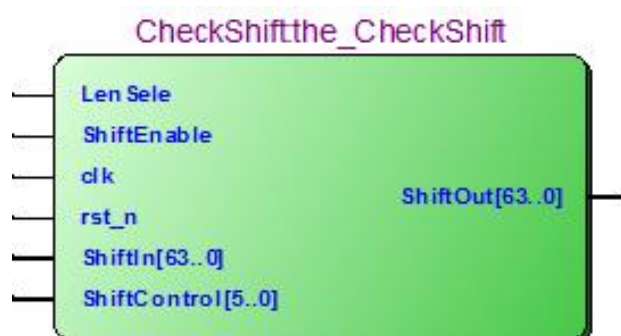


图 5.25 校验移位处理单元 RTL 图

校验移位处理单元 **CheckShift** 主要功能是实现对硬判决数据的移位处理，对应于码字与校验矩阵之间的乘法运算。需要指出的是该模块与变量节点更新运算单元 **HcMatrixAlgorithm** 和校验节点更新运算单元 **HvMatrixAlgorithm** 中所使用的移位处理单元 **LdpcShift** 不同之处在于，该模块的运算输入是硬判决以后的数据，每个比特代表码元中的一个信息位或校验位，而 **LdpcShift** 单元中的运算数据是量化后的软信息，数据宽度要远大于该模块，具体到本文中的量化方案而言，每 8 比特软信息对应于码元中的一个信息位或校验位。

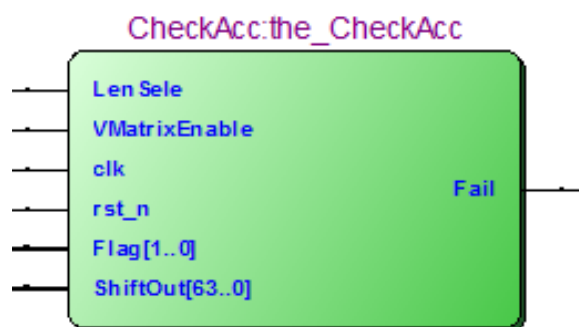


图 5.26 校验累加和运算单元 RTL 图

校验累加和运算单元的主要功能是实现判决后数据与校验矩阵相乘后结果的累加，给出累加数据结果输出，指示当前译码成败状态。

校验累加运算单元中的码长选择信号 `LenSele` 来自顶层，标志信号数据 `Flag` 来自判决存储器地址生成单元 `CheckRom` 的部分输出；移位后数据 `ShiftOut` 来自校验移位处理单元 `CheckShift` 的输出；输出 `Fail` 指示校验结果输出状态，高电平表示当前译码失败。

5) LDPC 译码器输出控制部分

LDPC 译码器输出控制部分与编码器输出控制部分类似，也是由输出数据移位寄存器、输出地址产生器和输出存储器三部分组成，这里不再赘述。

6) LDPC 译码控制状态机

该状态机完成数据输入/输出接口的管理，以及译码运算器各部分的运算和迭代处理管理。

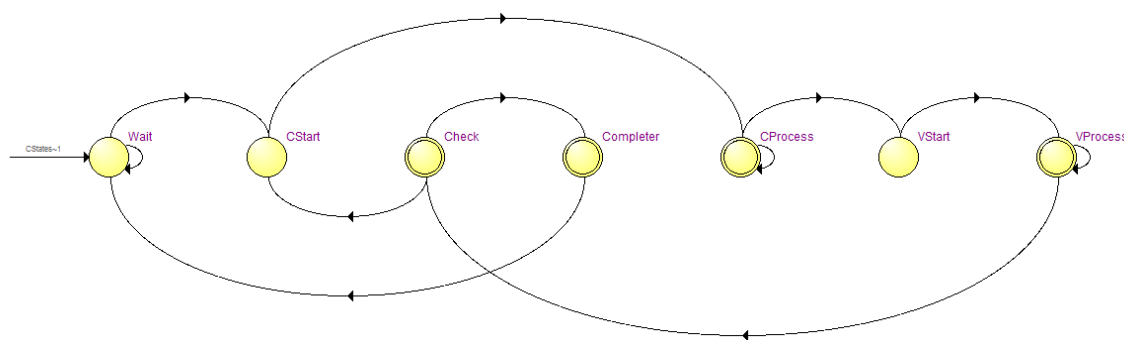


图 5.27 译码器状态转换图

如图 5.27 所示，译码器状态由空闲等待（Wait）开始，由同步信号触发后开始进入译码处理状态；译码处理由校验信息更新开始、校验信息更新处理、变量信息更新开始、变量信息更新处理、判决处理和结束状态等待七部分构成。从校验信息更新开始经过校验信息更新处理、变量信息更新开始和变量信息更新处理重新回到校验信息更新开始为一次完整的迭代，当判决处理结果为译码成功或迭代次数到达限定值后会跳出迭代循环，进入结束状态，完成一次完整的译码操作。

5.2.2 LDPC 译码器顶层接口

LDPC 译码器的接口信号同编码器的接口信号基本相同,可分为控制信号、状态信号、输入信号和输出信号四部分组成,如表 5.3 所示。

表 5.3 译码器接口使用说明表

| 接口名称 | I/O 类型 | I/O 位宽(bit) | 使用说明 |
|-----------|--------|-------------|--|
| clk | input | 1 | 时钟信号 |
| rst_n | input | 1 | 异步复位信号: 调试时用, 低电平有效 |
| syn | input | 1 | 上升沿有效 |
| len_sele | input | 1 | 译码长度选择信号: 为 0 时进行输入数据长为 504bit 编码, 1 时进行输入数据长为 768bit 编码 |
| rate_sele | input | 1 | 码率选择: 为 0 时选择码率为 1/2, 为 1 时选择码率为 3/4。 |
| iterate | input | 6 | 这里设置译码器最大迭代次数。 |
| start_in | input | 1 | 数据输入起始位: syn 信号有效 2 个 clk 之后可以产生该信号, 持续一个 clk, 高电平有效 |
| stop_in | input | 1 | 数据输入结束位: 持续一个 clk, 高电平有效 |
| start_out | input | 1 | 数据输出起始位: reset 信号有效 2 个 clk 之后可以产生该信号, 持续一个 clk, 高电平有效 |
| stop_out | input | 1 | 数据输出结束位: 持续一个 clk, 高电平有效 |
| data_in | input | 8 | 8bit 软信息输入数据 |
| data_out | output | 8 | 8bit 译码结果输出数据 |
| comp | output | 1 | 译码状态完成标志: 低电平有效 |
| fail | output | 1 | 译码失败指示, 如果在译码完成后, 该信号为高, 则表示输入信号译码失败。 |

5.2.3 LDPC 译码器 I/O 接口时序

如图 5.28 所示, 在译码器的接口信号中, 除 Fail 信号外其余信号和编码器的接口信号时序图完全相同。这里需要说明的是, 上图中的 Start_out 和 Stop_out 控制信号不一定要等到当前输入数据译码结束, 因为译码器的结构同编码器整体三级流水线的结构相同, 只要在上个周期中的译码数据处理结束, 即 Comp 信号拉低, 在下次该信号再次拉低前输入读控制信号 (Start_out/Stop_out) 都可以提取出译码的输出单元中寄存的译码数据。Fail 信号在译码完成后, 如果为高说明当前译码失败, 否则表示译码成功。所以译码器的译码状态是由 Comp 和 Fail 两个信号来体现, 如表 5.4 所示。

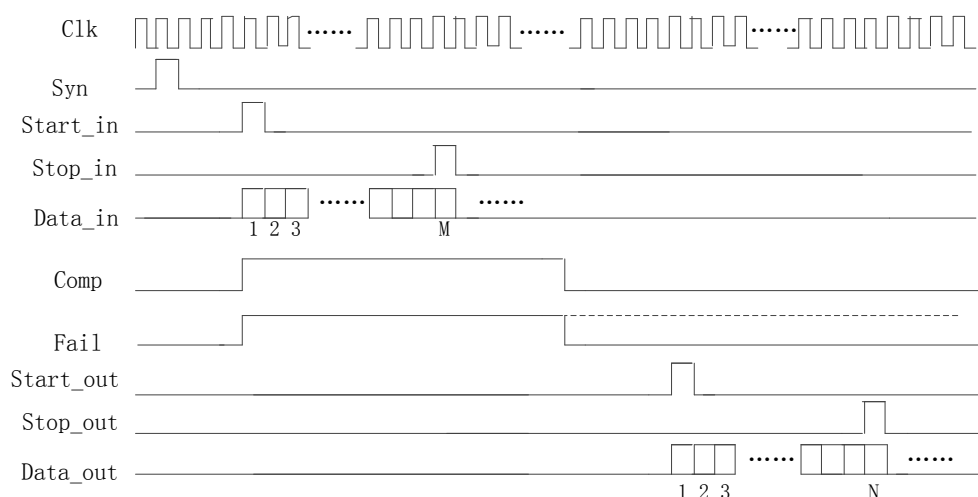


图 5.28 LDPC 译码器接口信号时序

表 5.4 译码器状态标志信号说明表

| 信号状态 | | 译码器状态 |
|------|------|-------|
| Comp | Fail | |
| 1 | * | 正在译码 |
| 0 | 1 | 译码失败 |
| 0 | 0 | 译码成功 |

5.2.4 译码器资源占用和时间占用情况

译码器设计完成后，通过仿真验证其功能完全正确，下面说明该码率译码器完成一次译码所需时间和所需资源的占用情况

当码长为 1008，码率为 1/2 的 LDPC 码，输入软信息共需 1008clk，输出译码结果共需 63clk；当码长为 1024，码率为 3/4 时，输入软信息共需 1024clk，输出译码结果共需 96clk。完成一次迭代（包括校验信息更新和变量信息更新）共需 324clk，一次校验 1clk，最大迭代次数设置为 10，最大迭代时延为 3240clk。所以在系统使用 200M 时钟时，完成一次译码最长时间为 16.2us，在使用 100M 时钟时，译码最长时间为 32.4us。

在使用 Altera 公司的器件 EP2AGX125 时，模块占用该器件的 ALUT 共计 16099 个等效需要占用的系统的寄存器数为 11232 个，除此之外还需要占用器件内部的块 RAM 为 260096bit，以及 20 个 Memory ALUT。表 5.5 为该模块资源占用情况，图 5.29 为译码器编译综合的结果。

表 5.5 译码器资源占用

| 器件名称 | 占用资源 | 数目 |
|------------------|---------------------|---------------|
| Altera EP2AGX125 | Combinational ALUTs | 16099 (11232) |
| | Memory ALUTs | 20 |
| | Block memory bits | 260096 |

| Flow Summary | |
|-----------------------------------|---|
| Flow Status | Successful - Sun Jul 20 16:15:21 2014 |
| Quartus II Version | 10.1 Build 153 11/29/2010 SJ Full Version |
| Revision Name | LdpcDecode |
| Top-level Entity Name | LdpcDecode |
| Family | Arria II GX |
| Device | EP2AGX125DF25C6ES |
| Timing Models | Final |
| Logic utilization | 19 % |
| Combinational ALUTs | 16,099 / 99,280 (16 %) |
| Memory ALUTs | 20 / 49,640 (< 1 %) |
| Dedicated logic registers | 11,232 / 99,280 (11 %) |
| Total registers | 11232 |
| Total pins | 33 / 300 (11 %) |
| Total virtual pins | 0 |
| Total block memory bits | 260,096 / 6,727,680 (4 %) |
| DSP block 18-bit elements | 0 / 576 (0 %) |
| Total GXB Receiver Channel PCS | 0 / 8 (0 %) |
| Total GXB Receiver Channel PMA | 0 / 8 (0 %) |
| Total GXB Transmitter Channel PCS | 0 / 8 (0 %) |
| Total GXB Transmitter Channel PMA | 0 / 8 (0 %) |
| Total PLLs | 0 / 6 (0 %) |
| Total DLLs | 0 / 2 (0 %) |

图 5.29 译码器编译综合结果图

5.3 码率兼容的实现方法

本文设计的编译码器具有码率兼容和码长兼容的突出特点。码率兼容的实现不仅与第 5.1.1 节的第三部分所示的打孔方案有关，还与相关控制数据的初始化格式设计有关。利用这种思路不仅可以减少数据的存储量，更重要的是降低了编译码器的版本更新难度。涉及码率兼容问题的具体数据内容包括以下输入输出单元控制数据、变量信息的更新单元控制数据、变量信息的更新单元控制数据、校验信息的更新单元控制数据、判决单元控制数据五个部分。

1) 输入输出单元控制数据处理

编码器输入端的码率兼容问题在第 5.1.1 节有详尽地描述，输出端处理过程与之对应相反，现分析如下：

当 LEN_SELET=0，对应信息码元长度为 768 时，输出控制单元中的输入 64 比特数据只需要 8 个时钟周期每次取 8 比特数据即可实现；当 LEN_SELET=1 时，对应信息码元长度为 504 时，输出控制单元中的输入 64 比特数据中只有低 42 比特数据有效，此时需要对应于输入控制的方法，将其按 8 比特一组的数据格式取出。

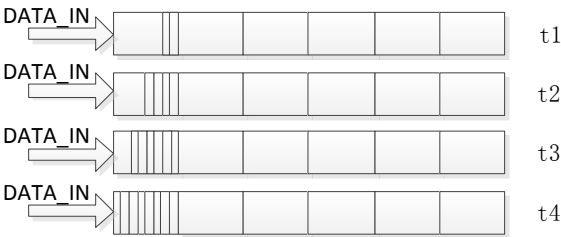


图 5.30 输出数据移位寄存器工作原理

以码长为 1008、1/2 码率的 LDPC 编码输出为例，在输出控制单元中，64 比特输入数据（coderead_data）并行输入，将其中的低 42 比特依次寄存到 6 个连续字节组成的移位寄存器中，具体步骤如下：

t1 时刻将 42 比特数据寄存到移位寄存器后，开始经过 5 个时钟周期依次输出 5 字节数据；t2 时刻再将下一组 42 比特数据紧接着上次剩余的 2 比特寄存后再开始依次输出 5 字节数据；t3 时刻再将下一组 42 比特数据紧接着剩余的 4 比特寄存后再开始依次输出 5 字节数据；t4 时刻再将下一组 42 比特数据紧接着剩余的 6 比特寄存后再开始依次输出 6 字节数据。按照“5 字节-5 字节-5 字节-6 字节”的顺序循环依次输出，以此实现 42 比特数据向 8 比特的整合转换。

译码器的输入端的数据重组处理思路和编码器相同，不同的地方是 8 比特数据并行输入对应一个信息比特或校验比特，所以采用二维数组的方式来存储输入的软信息数据。根据本文中的设计需要，将二维数组设置成 64×8 的格式，既能保证 1024 码长 3/4 码率对应的 64 维子矩阵的使用要求，同时可向下兼容 1008 码长 1/2 码率对应的 42 维子矩阵的使用。译码器的输出端控制和编码器的输出端控制完全相同。

2) 变量信息和校验信息更新控制数据

如上文所述，对于变量信息更新和校验信息更新单元中都会有一个移位信息生成单元模块，该模块是由 Rom 构成，在设计时内部初始化文件中保存有运算相关的数据，以校验信息更新单元中的移位信息生成模块为例，初始化数据格式如下图所示：

| 20 | 19 | 18:13 | 12:7 | 6:0 |
|------|------------|------------|---------|-----|
| flag | shiftdata0 | shiftdata1 | address | |

图 5.31 移位信息生成模块中初始化数据格式

flag:输入读地址 VMatrixReadAddress 所读到的校验节点在该行的对应位置，第一个、最后一个或其他。

地址 address: 输入读地址 VMatrixReadAddress 所读到的校验节点处变量信息对应的编址信息，变量的编址信息是由非零阵节点纵向连续编号。

移位数据 ShiftData: 输入读地址 VMatrixReadAddress 所读到的校验节点处变量信息需要移位的位数；LenSele =0 时，选 64 对应的移位数据，当 Lensele=1 时，选 42 对应的移位数据。

3) 判决单元控制数据

| | | | | |
|-------|---------|---------------|--------------|---------|
| 30:29 | [28 27] | [26:22 21:17] | [16:11 10:5] | 4:0 |
| flag | wr_en | wr_address | shiftcontrol | address |

图 5.32 判决控制单元中初始化数据格式

Flag:指示累加器, Flag=0 时, 从头开始累加, 即为第一个累加数据; 当 Flag=1/2 时, 累加器累加。

当 RateSelect=0 时, 选择 1/2 码率, 表示不打孔时的写地址和写控制信号

LenSelect=0 时, 选择 1008 码长, 相乘移位数据

译码结果写使能/写地址: 当 RateSele=0 时, 表示码率为 1/2, 选择 bit28 数据作为写使能, bit26:22 数据作为写地址; 否则, 表示码率为 3/4, 选择 bit27 数据作为写使能, bit21:17 数据作为写地址;

校验移位数据: 当 LenSele=0 时, 表示选择码长为 1008, 选择 bit16:11 数据作为循环移位数据, 否则, 选择 bit10:5 数据作为循环移位数据。

校验数据读地址: 输出对应位置的校验结果读地址。

当然, 以上论述中的初始化数据及格式为根据项目需要设定而成, 具有一定的局限性, 但这种数据格式的设计和应用方法对于系统的可移植和版本更新具有很大的优势。

第六章 本文总结

本文作者结合工程实践项目，采用理论分析和硬件实现的方法，针对 IEEE802.16e 标准中推荐的一种 QC-LDPC 编译码方案进行设计实现，就到目前为止已经完成的 LDPC 编译码器功能和性能分析来看，已经达到了预期设计的目标。现对本文所作的工作总结如下：

1) 系统总结了 LDPC 码和 QC-LDPC 码的相关知识及特点，简要介绍了 LDPC 码和 QC-LDPC 码的构造方法。

2) 编译码算法方面，首先介绍了传统编码算法、RU 编码算法和 QC-LDPC 码的编码算法，具体推导了其中相关公式，并对其实现复杂度进行比较，给出一种 QC-LDPC 线性编码；然后详细地推导 BP 算法的概率域和对数域形式及其各种简化算法的相关公式，从译码算法的纠错性能和实现复杂度两个方面，进行分析对比之后确定出归一化最小和算法为最适合硬件实现的译码算法；最后简要介绍了打孔算法的原理。

3) 对 LDPC 码的译码量化方案展开了详细探讨，通过运用 Matlab 软件对译码算法进行定点仿真，最后确定出输入软信息和中间存储信息的量化比特宽度和量化范围，为后面的硬件实现提供了重要参考和依据。

4) 根据 IEEE802.16e 标准中 LDPC 码特殊的结构特点，设计实现了一种部分并行的 QC-LDPC 编译码器。该设计充分利用 LDPC 校验矩阵的规律，采用一种恰当的硬件结构和独特的存储器调用策略，在保证高性能和较大吞吐率的情况下，以较少的硬件资源实现了两种码率的复用。运用 verilog HDL 硬件语言在 Quartus II 软件平台上完成了基于 1/2 码率基本阵的兼容 1008 码长、1/2 码率和 1024 码长、3/4 码率的 LDPC 编译码器设计，并通过 Matlab 和 Modelsim 工具完成了编译码器的功能仿真和性能验证，结果表明文中提出的编译码硬件结构能够达到设计的要求。

LDPC 码编译码的涉及范围相当广泛，由于本文作者时间和能力的限制，只掌握了 LDPC 码的一些基础性的知识，本文的研究内容也只涉及了其中个别问题，在以下几个方面需要进行进一步的完善和探索：

1) 本文的设计只针对 AWGN 信道下的 BPSK 调制一种情况，下一步应该进一步完善编译码器的设计，使之能够满足其他调制方式的编译码需求；

2) 本文的设计只针对单一固定的 H 基础矩阵，码率兼容的特性是通过打孔的方式来实现的。由于时间等客观因素的关系，没能针对多种基础矩阵来进行兼容性设计。

3) 编译码器的实现方案, 还需要从以下几个方面进行平衡: 第一, 寻求在保持较低译码复杂度的同时, 性能更加优异的译码算法; 第二, 改善译码量化方案, 减少量化损失; 第三, 进一步改善编译码结构, 在不大幅度提高硬件资源占用量的同时减小处理延迟, 提高吞吐量。

附录 A

IEEE802.16e 中的基本阵 H_{base}

Rate 1/2:

| | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -1 | 94 | 73 | -1 | -1 | -1 | -1 | -1 | 55 | 83 | -1 | -1 | 7 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 27 | -1 | -1 | -1 | 22 | 79 | 9 | -1 | -1 | -1 | 12 | -1 | 0 | 0 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 24 | 22 | 81 | -1 | 33 | -1 | -1 | -1 | 0 | -1 | -1 | 0 | 0 | -1 | -1 | -1 | -1 | -1 |
| 61 | -1 | 47 | -1 | -1 | -1 | -1 | -1 | 65 | 25 | -1 | -1 | -1 | -1 | 0 | 0 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | 39 | -1 | -1 | -1 | 84 | -1 | -1 | 41 | 72 | -1 | -1 | -1 | -1 | 0 | 0 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 46 | 40 | -1 | 82 | -1 | -1 | -1 | 79 | 0 | -1 | -1 | -1 | -1 | 0 | 0 | -1 | -1 |
| -1 | -1 | 95 | 53 | -1 | -1 | -1 | -1 | -1 | 14 | 18 | -1 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | -1 | -1 |
| -1 | 11 | 73 | -1 | -1 | -1 | 2 | -1 | -1 | 47 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | -1 |
| 12 | -1 | -1 | -1 | 83 | 24 | -1 | 43 | -1 | -1 | -1 | 51 | -1 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | -1 |
| -1 | -1 | -1 | -1 | -1 | 94 | -1 | 59 | -1 | -1 | 70 | 72 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 0 | 0 |
| -1 | -1 | 7 | 65 | -1 | -1 | -1 | -1 | 39 | 49 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 0 |
| 43 | -1 | -1 | -1 | -1 | 66 | -1 | 41 | -1 | -1 | -1 | 26 | 7 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 0 |

Rate $2/3$ A code:

| | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 3 | 0 | -1 | -1 | 2 | 0 | -1 | 3 | 7 | -1 | 1 | 1 | -1 | -1 | -1 | 1 | 0 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | 1 | -1 | 36 | -1 | -1 | 34 | 10 | -1 | -1 | 18 | 2 | -1 | 3 | 0 | -1 | 0 | 0 | -1 | -1 | -1 |
| -1 | -1 | 12 | 2 | -1 | 15 | -1 | 40 | -1 | 3 | -1 | 15 | -1 | 2 | 13 | -1 | -1 | 0 | 0 | -1 | -1 | -1 |
| -1 | -1 | 19 | 24 | -1 | 3 | 0 | -1 | 6 | -1 | 17 | -1 | -1 | -1 | 8 | 39 | -1 | -1 | 0 | 0 | -1 | -1 |
| 20 | -1 | 6 | -1 | -1 | 10 | 29 | -1 | -1 | 28 | -1 | 14 | -1 | 38 | -1 | -1 | 0 | -1 | -1 | 0 | 0 | -1 |
| -1 | -1 | 10 | -1 | 28 | 20 | -1 | -1 | 8 | -1 | 36 | -1 | 9 | -1 | 21 | 45 | -1 | -1 | -1 | -1 | 0 | 0 |
| 35 | 25 | -1 | 37 | -1 | 21 | -1 | -1 | 5 | -1 | -1 | 0 | -1 | 4 | 20 | -1 | -1 | -1 | -1 | -1 | 0 | 0 |
| -1 | 6 | 6 | -1 | -1 | -1 | 4 | -1 | 14 | 30 | -1 | 3 | 36 | -1 | 14 | -1 | 1 | -1 | -1 | -1 | -1 | 0 |

Rate 2/3 B code:

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2 | -1 | 19 | -1 | 47 | -1 | 48 | -1 | 36 | -1 | 82 | -1 | 47 | -1 | 15 | -1 | 95 | 0 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 69 | -1 | 88 | -1 | 33 | -1 | 3 | -1 | 16 | -1 | 37 | -1 | 40 | -1 | 48 | -1 | 0 | 0 | -1 | -1 | -1 | -1 | -1 |
| 10 | -1 | 86 | -1 | 62 | -1 | 28 | -1 | 85 | -1 | 16 | -1 | 34 | -1 | 73 | -1 | -1 | -1 | 0 | 0 | -1 | -1 | -1 | -1 |
| -1 | 28 | -1 | 32 | -1 | 81 | -1 | 27 | -1 | 88 | -1 | 5 | -1 | 56 | -1 | 37 | -1 | -1 | 0 | 0 | -1 | -1 | -1 | -1 |
| 23 | -1 | 29 | -1 | 15 | -1 | 30 | -1 | 66 | -1 | 24 | -1 | 50 | -1 | 62 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | -1 | -1 |
| -1 | 30 | -1 | 65 | -1 | 54 | -1 | 14 | -1 | 0 | -1 | 30 | -1 | 74 | -1 | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 0 | -1 |
| 32 | -1 | 0 | -1 | 15 | -1 | 56 | -1 | 85 | -1 | 5 | -1 | 6 | -1 | 52 | -1 | 0 | -1 | -1 | -1 | -1 | -1 | 0 | 0 |
| -1 | 0 | -1 | 47 | -1 | 13 | -1 | 61 | -1 | 84 | -1 | 55 | -1 | 78 | -1 | 41 | 95 | -1 | -1 | -1 | -1 | -1 | -1 | 0 |

Rate 3/4 A code:

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 6 | 38 | 3 | 93 | -1 | -1 | -1 | 30 | 70 | -1 | 86 | -1 | 37 | 38 | 4 | 11 | -1 | 46 | 48 | 0 | -1 | -1 | -1 | -1 |
| 62 | 94 | 19 | 84 | -1 | 92 | 78 | -1 | 15 | -1 | -1 | 92 | -1 | 45 | 24 | 32 | 30 | -1 | -1 | 0 | 0 | -1 | -1 | -1 |
| 71 | -1 | 55 | -1 | 12 | 66 | 45 | 79 | -1 | 78 | -1 | -1 | 10 | -1 | 22 | 55 | 70 | 82 | -1 | -1 | 0 | 0 | -1 | -1 |
| 38 | 61 | -1 | 66 | 9 | 73 | 47 | 64 | -1 | 39 | 61 | 43 | -1 | -1 | -1 | -1 | 95 | 32 | 0 | -1 | -1 | 0 | 0 | -1 |
| -1 | -1 | -1 | -1 | 32 | 52 | 55 | 80 | 95 | 22 | 6 | 51 | 24 | 90 | 44 | 20 | -1 | -1 | -1 | -1 | -1 | -1 | 0 | 0 |
| -1 | 63 | 31 | 88 | 20 | -1 | -1 | -1 | 6 | 40 | 56 | 16 | 71 | 53 | -1 | -1 | 27 | 26 | 48 | -1 | -1 | -1 | -1 | 0 |

Rate 3/4 B code:

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -1 | 81 | -1 | 28 | -1 | -1 | 14 | 25 | 17 | -1 | -1 | 85 | 29 | 52 | 78 | 95 | 22 | 92 | 0 | 0 | -1 | -1 | -1 | -1 |
| 42 | -1 | 14 | 68 | 32 | -1 | -1 | -1 | -1 | 70 | 43 | 11 | 36 | 40 | 33 | 57 | 38 | 24 | -1 | 0 | 0 | -1 | -1 | -1 |
| -1 | -1 | 20 | -1 | -1 | 63 | 39 | -1 | 70 | 67 | -1 | 38 | 4 | 72 | 47 | 29 | 60 | 5 | 80 | -1 | 0 | 0 | -1 | -1 |
| 64 | 2 | -1 | -1 | 63 | -1 | -1 | 3 | 51 | -1 | 81 | 15 | 94 | 9 | 85 | 36 | 14 | 19 | -1 | -1 | 0 | 0 | 0 | -1 |
| -1 | 53 | 60 | 80 | -1 | 26 | 75 | -1 | -1 | -1 | 86 | 77 | 1 | 3 | 72 | 60 | 25 | -1 | -1 | -1 | -1 | 0 | 0 | 0 |
| 77 | -1 | -1 | -1 | 15 | 28 | -1 | 35 | -1 | 72 | 30 | 68 | 85 | 84 | 26 | 64 | 11 | 89 | 0 | -1 | -1 | -1 | -1 | 0 |

Rate 5/6 code:

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 25 | 55 | -1 | 47 | 4 | -1 | 91 | 84 | 8 | 86 | 52 | 82 | 33 | 5 | 0 | 36 | 20 | 4 | 77 | 80 | 0 | -1 | -1 |
| -1 | 6 | -1 | 36 | 40 | 47 | 12 | 79 | 47 | -1 | 41 | 21 | 12 | 71 | 14 | 72 | 0 | 44 | 49 | 0 | 0 | 0 | 0 | -1 |
| 51 | 81 | 83 | 4 | 67 | -1 | 21 | -1 | 31 | 24 | 91 | 61 | 81 | 9 | 86 | 78 | 60 | 88 | 67 | 15 | -1 | -1 | 0 | 0 |
| 68 | -1 | 50 | 15 | -1 | 36 | 13 | 10 | 11 | 20 | 53 | 90 | 29 | 92 | 57 | 30 | 84 | 92 | 11 | 66 | 80 | -1 | -1 | 0 |

参考文献

- [1] C.E.Shannon,“A mathematical theory of communication.”[D] BellSyst. Tech.J., vol.27, pp.379-423,623-656,July-Oct.1948;Reprinted in C.E.Shannon and W.Weaver,The Mathematical Theory of Communication.Urbana,IL:Univ.Illinois Press,1949.
- [2] M. J. E. Golay, Notes on digital coding [J], Proc. IRE, 1949, 37: 657.
- [3] R. W. Hamming, Error detecting and error correcting codes [J], Bell Syst. Tech. J, 1950, 29: 147-150.
- [4] D. E. Muller, “Applications of boolean algebra to switching circuits design and to error detection”[J],IRE Trans. Inform. Theory, 1954, IT-4: 38-49.
- [5] I.S. Reed, “A class of multiple-error-correcting codes and the decoding scheme”[J], IRE Trans. Electron. Comput., 1954, EC-3: 6-12.
- [6] E. Prange, Cyclic error-correcting codes in two symbols, Air Force Cambridge Res. Center, Cambridge[D], MA, Tech. Note AFCRC-TN-57-103, 1957.
- [7] 王新梅,肖国镇.纠错码-原理与方法[M],修订版,西安: 西安电子科技大学出版社, 1991,7: 24
- [8] P. Elias, “Coding for noisy channels”[J], IRE Convention Record, 1955, 3(4): 37-46.
- [9] J. M. Wozencraft and B. Reiffen, Sequential decoding[D], MA: MIT Press, Cambridge, 1961.
- [10] R. M. Fano, “A heuristic discussion of probabilistic decoding”[J], IEEE Trans. Inform. Theory, 1963, IT-9: 64-74.
- [11] J. L. Massey, Threshold decoding[D], MA: MIT Press, Cambridge, 1963.
- [12] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”[J], IEEE Trans. Inform. Theory, 1967, IT-13(4): 260-269.
- [13] G. D. Forney, Review of random tree codes, NASA Ames Res. Ctr., Moffett Field, CA, Contract NAS2-3637, NASA CR73176, Appendix A, Final Rep, 1967.
- [14] G. D. Forney, Concatenated codes. PhD Thesis, MIT, 1966.
- [15] R.G. Gallager, “Low-Density Parity-Check Codes”, IRE Transaction on Information Theory, vol.8, pp.21-28, Jan.1962.
- [16] G. D. Forney, Jr, “The viterbi algorithm”[J], Proc. IEEE, 61: 268-78, 1973.
- [17] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate”[J], IEEE Trans. Inform. Theory, IT-20:284-87,1974.
- [18] J. K. Wolf, “Efficient maximum-likelihood decoding of linear block codes using a trellis” [J], IEEE Trans. Inform. Theory, IT-24:76-80, 1978.
- [19] J. L. Massey, “Foundation and methods of channel encoding”[J], in Proc. Int. Conf. Inform.

Theory and Systems, NTG-Fachberichte, Berlin, 1978.

- [20] G. D. Forney, Jr, "Coset codes II: Binary lattices and related codes"[J], IEEE Trans. Inform. Theory, 34(5): 1152-87, 1988.
- [21] G. Ungerboeck, "Channel coding with multilevel/phase signals" [J], IEEE Trans. Inform. Theory, IT-28:55-67, January 1982.
- [22] L. F. Wei, "Trellis-coded modulation with multidimensional constellations" [J], IEEE Trans. Inform. Theory, 33:483-501, 1987.
- [23] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes" [J], in: Proc. IEEE Int. Conf. on Commun., Geneva, Switzerland, 1993, vol. 2, 1064-1070.
- [24] J. Hagenauer and P. Hoeher. A Viterbi Algorithm with Soft-Decision outputs and Its Applications[J]. Proc. IEEE GLOBECOM '89, Dallas, TX, Nov. 1989: 47(1): 1-7.
- [25] R. G. Gallager. Low-Density Parity-Check Codes. Cambridge[D]. MA: MIT Press, 1963.
- [26] D. J. C. MacKay, R. M. Neal. Near Shannon limit performance of Low-Density Parity-Check Codes [J]. IEE Electronics Letters, 1996, 32(18): 1645-1646.
- [27] R. Michael Tanner. "A recursive approach to low complexity codes"[J], IEEE Transactions on Information Theory, 1981, 27(5): 533-547. Sep. 1981.
- [28] D. J. C. MacKay. "Good Error-Correcting Codes based on Very Sparse Matrices"[J], IEEE Trans. Inform. Theory, vol. 45, pp. 399-431, Mar 1999.
- [29] T. J. Richardson and R. L. Urbanke. "The capacity of low-density parity-check codes under message-passing decoding"[J], IEEE Trans. Inf. Theory, 47(2): 599-618, Feb. 2001.
- [30] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. "Design of capacity-approaching irregular low-density parity-check codes"[J], IEEE Trans. Inf. Theory, 47(2): 619-637, Feb. 2001.
- [31] M. G. Luby and M. Mitzenmacher, "Improved low-density parity-check codes using irregular graphs"[J], IEEE Trans. Inf. Theory, 47(2): 585-598, 2001.
- [32] M. P. C. Fossorier. "Quasi-cyclic low-density parity-check codes from circulant permutation matrices"[J], IEEE Transactions on Information Theory, 2004, 50(8): 1788-1793.
- [33] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-Density Parity-Check Codes Based On Finite Geometries: a rediscovery and new results"[J], IEEE Transactions on Information Theory, Nov. 2001, 47(7), pp. 2711-2736.
- [34] Shu Lin and Daniel J. Costello, Jr. Error Control Coding[M], Second Edition. Prentice Hall, Upper Saddle River, NJ., 2004.
- [35] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-limit quasi-cyclic low-density parity-check codes"[J], IEEE Trans. Commun., 2004, 52(7): 1038-1042.

-
- [36] J. Xu, L. Chen, L.Q. Zeng et al., "Construction of low-density parity-check codes by superposition"[J], IEEE Trans. Commun., 2005, 53(2): 243-251.
 - [37] J. Xu, L. Chen, I. Djurdjevic et al., "Construction of regular and irregular LDPC codes: geometry decomposition and masking"[J], IEEE Trans. Inform. Theory, 2007, 53(1): 121-134.
 - [38] S. Myung and K. Yang, "A combining method of quasi-cyclic LDPC codes by the chinese remainder theorem"[J], IEEE Commun. Lett., Sep. 2005, 9, pp. 823-825.
 - [39] D.J.C. Mackay, S.T. Wilson, and M.C. Davey, Comparison of constructions of irregular Gallager codes[J], IEEE Trans. Commun., 1999, 47(10): 1449-1454.
 - [40] L. Ping, W.K. Leung, and N. Phamdo, "Low density parity check codes with semi-random parity check matrix"[J], IEEE Electronics Letters, Jan. 1999, 35, pp. 38-39.
 - [41] T. Tian, C. Jones, J.D. Villasenor et al., "Selective avoidance of cycles in irregular LDPC code construction"[J], IEEE Trans. Commun., 2004, pp. 1242-1247.
 - [42] T. Mittelholzer, "Efficient encoding and minimum distance bounds of Reed-Solomon-type array codes"[J], ISIT 2002, Lausanne, Switzerland, June 30 - July 5, 2002: 282.
 - [43] S.J. Johnson, S.R. Weller, "A family of irregular LDPC codes with low encoding complexity"[J], IEEE Communications Letters, 2003, 7(2): 79-81.
 - [44] S. Freundlich, D. Burshtein, S. Litsyn, "Approximately lower triangular ensembles of LDPC codes with linear encoding complexity" [J], IEEE Trans. Inform. Theory, 2007, 53(4): 1484-1494.
 - [45] 袁瑞佳. LDPC 码的高效编译码实现技术研究[D]. 西安: 西安电子科技大学, 2012
 - [46] T.J. Richardson and R.L. Urbanke, "Efficient encoding of low-density parity-check codes"[J], IEEE Trans. Inform. Theory, Feb. 2001, 47(2), pp. 638-656.
 - [47] J. Lu and J. Mouru, "Linear time encoding of LDPC codes"[J], IEEE Trans. Inform. Theory, 2010, 56(1) : 90-94.
 - [48] D. Divsalar, H. Jin, and R.J. McEliece, "Coding theorems for turbo-like codes"[J], in Proc. 36th Allerton Conf. on Commun., Control, and Comp., Monticello, IL, USA, 1998, 201-210.
 - [49] Divsalar D, Jin H, McEliece R. Coding theorems for Turbolike codes. Proceedings of the 36th Annual Allerton Conference on Communication Control and Computing. Monticello [J], IL, USA, 1988, 9: 201-210.
 - [50] H. Jin, A. Khandekar, and R.J. McEliece, Irregular repeat-accumulate codes[J], Proc. 2nd Int. Symp. on Turbo Codes, Brest, France, 2000, 9: 1-8.
 - [51] Jin H. Analysis and design of Turbo-like codes[D]. PhD dissertation. California Institute of Technology, 2001.
 - [52] W. Lin, B. Bai, Y. Li, and X. Ma, Design of q -ary irregular repeat accumulate codes [J], IEEE Int.

- Conf. on AINA, Bradford, UK, May. 2009, pp.201-206.
- [53] W.Lin and B.Bai, Efficiently encodable Q-ary LDPC codes based on accumulators, *Advances in Information Sciences and Service Sciences*, vol.3, no.9, pp.9-16, Oct. 2011.
- [54] J.T. Zhang and M.P.C. Fossorier, A modified weighted bit-flipping decoding of low density Parity-check codes [J], *IEEE Commun. Lett.*, 2004, 8(3): 165-167.
- [55] F. Guo and L. Hanzo, Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes [J], *IEE Electronics Letters*, 2004, 40(21): 1356-1358.
- [56] X.F. Wu, C. Ling, M. Jiang et al., New insights into weighted bit-flipping decoding [J], *IEEE Trans. Commun.*, 2009, 57(8): 2177-2180.
- [57] Z.Y. Liu and D.A. Pados, A decoding algorithm for finite-geometry LDPC codes [J], *IEEE Trans. Commun.*, 2009, 53(3): 415-421.
- [58] M.P.C.Fossorier, M. Mihaljevic, and H. Imai, Reduced complexity iterative decoding of low-density parity check codes based on belief propagation[J], *IEEE Trans. Commun.*, 1999, 47(5): 673-680.
- [59] E.Eleftheriou, T.Mittelhozer and A.Dholakia. Reduced-complexity Decoding Algorithm for Low-Density Parity-Check Codes[J]. *IEEE Electronics Letters*, Jan. 2001, (37): 102-104
- [60] M.P.C.Fossorier, "Iterative reliability-based decoding of low-density parity check codes"[J], *IEEE Journal of selected areas in Commun.*, May 2001, 19(5): 908-917.
- [61] J.Chen and M.P.C.Fossorier, Near optimum universal belief propagation based decoding of low-density parity check codes [J], *IEEE Trans. Commun.*, vol.50, 2002, 50(3): 406-414.
- [62] X.F.Wu, Y.Song, M.Jiang, and C.M.Zhao, Adaptive-normalized/offset min-sum algorithm[J], *IEEE Commun. Lett.*, 2010, 14(7): 667-669.
- [63] S.Hemati, A.H.Banihashemi and C.Plett, A 0.18-um CMOS analog min-sum iterative decoder for a(32,8) low-density parity-check code[J]. *IEEE Journal of Solid-State Circuits*, vol. 41, no.11, Nov. 2006.
- [64] 贺鹤云. LDPC 码基础与应用[M]. 北京 人民邮电出版社, 2009.
- [65] 雷菁. 低复杂度 LDPC 码构造及译码研究[D]. 长沙: 国防科学技术大学, 2009.
- [66] IEEE P802. 11 Wireless LANs WWiSE Proposal: High throughput extension to the 802. 11 Standard[S]. August 2004, IEEE 11-04-0886-00-000n.
- [67] Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, New Gathering and other broadband satellite applications[S], Draft EN 302 307 V1.1.1 ed. , ETSI, June 2004.
- [68] Eleftheriou E et al. Low-density parity-codes for DSL transmission. Temporary Document BI-095, ITU-T SS[S], Study Group 15/4, Goa, India, 23-27 Oct, 2000.

- [69] Zang Di, et al. Low-density parity-check codes used for compressed image transmission over noisy channel[S]. from Internet, Apr. 2003.
- [70] 中国数字电视地面广播传输系统标准《数字电视地面广播传输系统帧结构、信道编码和调制》[S].GB20600-2006. 国家标准化管理委员会.
- [71] 中华人民共和国广播电影电视行业标准移动多媒体广播第 1 部分《广播信道帧结构、信道编码与调制》[S].国家广播电影电视总局. 2006.10.24.
- [72] M.G.Luby, M.Mitzenmacher, M.A.Shokrollahi et al.,Analysis of low-density codes and improved designs using irregular graphs[J],In Proc.30th Annu.Symp.Theory of Computing,pp249,1998.
- [73] IEEE Std 802. 16TM Specification[S]. 2009.
- [74] J. Ha, J. Kim, D. Klinc et al., Rate-compatible Punctured Low-density Parity-check Codes with Short Block Lengths [J]. IEEE Trans on Information Theory,2006,52(2):728-738.

致谢

我的硕士研究生生活即将在 2014 年的冬天画上一个句号。在这篇硕士论文完成之际，回顾攻读硕士学位期间的点点滴滴，谨向所有指导、关心和帮助过我的导师、老师、同学、朋友和亲人们表示衷心的感谢！

首先要特别地感谢我的导师张海林教授，很荣幸能够成为张海林教授的一名学生。在此，我将最诚挚的谢意和深深的祝福献给我的导师张海林教授。正是张老师悉心的指导和深切的关怀，给予我不断努力学习动力。张老师平日工作极其繁忙，当项目遇到难题时，他不惜牺牲节假日休息时间，在实验室和我们一起调试板子，一起谈论问题，一起分析数据，给予我极大的帮助和指导，让我在工程技术和学识上都受益匪浅，严谨的治学态度和对科研的执著也一直激励着我。

其次，感谢武德斌老师、任智源老师、杨栋老师等其他老师，你们在生活上和学习上给了我很多的帮助，尤其感谢武德斌老师对我在研究生期间的指导和帮助，特别是武德斌老师丰富的项目经验和对工程实践知识的融会贯通更是让我受益良多。有了你们的帮助，我才能够顺利地毕业。

同时，感谢我的同学，他们是李涛、陈瑞瑞、李爽、张婷婷、王绣琮，我们一起学习、一起成长。感谢我的朋友们还有室友们，是你们让我拥有了深厚的友谊，拥有了对生活的热爱，在我遇到学习和生活中的挫折时，是你们给了我前进的力量。

在此，我要特别要深深地感谢我的父母和亲人，感谢你们一直以来对我默默无闻的支持和关爱，感谢你们为我所付出的所有，使我能够健康快乐地成长并顺利地完成学业。

最后，感谢论文评议组的老师和所有关心、帮助过我的人！真心地祝愿你们健康快乐！

作者简介

1.基本情况

男，河南驻马店人，1988 年 10 月出生，西安电子科技大学通信工程学院军事通信学专业 2012 级硕士研究生。

2.教育背景

2008.09~2012.07 就读于中北大学信息与通信工程学院通信工程专业，获工学学士学位

2012.09~ 西安电子科技大学通信工程学院军事通信学领域硕士研究生

3.攻读硕士学位期间的研究成果

3.1 发表的学术论文

3.2 发明专利和科研情况

[1] 武德斌，安永宁，张海林等.一种基于总线的多位数据相关器设计方法[P].中国，实用新型.201310419479.8,2014

[2] 校级项目.宽度线性数字基带预失真器的实现，2012.至今，整体联调测试，完成软件算法编程和软硬件调试工作；

[3] 校级项目.高速码率兼容的 LDPC 的 FPGA 实现，2013.9-2013.12，已调试验证完成，完成算法验证和硬件实现。



西安电子科技大学

地址：西安市太白南路2号

邮编：710071

网址：www.xidian.edu.cn