研究与开发

一种合并状态度量计算的高效并行 Turbo 码译码器结构设计及 FPGA 实现

张茜[1]，詹明[1]，章坚武[2]，王富龙[1]，冯云开[1]，唐浩[1]
（1. 西南大学，重庆 400715；2. 杭州电子科技大学，浙江 杭州 310018）

摘　要：为满足无线通信中高吞吐、低功耗的要求，并行译码器的结构设计得到了广泛的关注。基于并行 Turbo 码译码算法，研究了前后向度量计算中的对称性，提出了一种基于前后向合并计算的高效并行 Turbo 码译码器结构设计方案，并进行现场可编程门阵列（field-programmable gate array，FPGA）实现。结果表明，与已有的并行 Turbo 码译码器结构相比，本文提出的设计结构使状态度量计算模块的逻辑资源降低 50%左右，动态功耗在 125 MHz 频率下降低 5.26%，同时译码性能与并行算法的译码性能接近。

关键词：状态度量合并计算；Turbo 码；FPGA 实现；并行算法
中图分类号：TN929
文献标志码：A
doi: 10.11959/j.issn.1000－0801.2022023

Design and FPGA implementation of an efficient parallel Turbo decoder for combining state metric calculations

ZHANG Qian[1], ZHAN Ming[1], ZHANG Jianwu[2], WANG Fulong[1], FENG Yunkai[1], TANG Hao[1]
1. Southwest University, Chongqing 400715, China
2. Hangzhou Dianzi University, Hangzhou 310018, China

Abstract: In order to achieve the requirement of high throughput and low-power in wireless communication, a parallel Turbo decoder has attracted extensive attention. By analyzing the calculating of the state metrics, a low-resource parallel Turbo decoder architecture scheme based on merging the forward and backward state metrics calculation modules was proposed, and effectiveness of the new architecture was demonstrated through field-programmable gate array (FPGA) hardware realization. The results show that, compared with the existing parallel Turbo decoder architectures, the proposed design architecture reduces the logic resource of state metrics calculation module about 50%, while the dynamic power dissipation of the decoder architecture is decreased by 5.26% at the frequency of 125 MHz. Meanwhile the decoding algorithm is close to the decoding performance of the parallel algorithm.

Key words: state measure merge calculation, Turbo code, FPGA implementation, parallel algorithm

收稿日期：2021−09−30；修回日期：2022−01−28
通信作者：詹明，zmdjs@swu.edu.cn；章坚武，jwzhang@hdu.edu.cn
基金项目：国家自然科学基金资助项目（No.61671390）
Foundation Item: The National Natural Science Foundation of China (No.61671390)

0 引言

在无线通信系统中，信道编码技术十分重要，主要用于纠正信息传输过程中出现的错误。Turbo码作为先进的迭代信道编码之一，译码性能逼近香农极限[1-2]，所以在各个领域广泛应用，尤其是工业物联网中[3-5]。无线通信系统中因为能量受限、译码算法迭代的依赖性和吞吐量的提高成为瓶颈，同时，工业物联网中 Turbo 码译码器的资源占用亦是待解决问题。因此，一种降低硬件资源且解决迭代译码算法实现高吞吐量 Turbo 码译码器结构成为研究的重点。有关 Turbo 码译码器设计的研究中，译码主要采用最大后验概率（maximum a posteriori probability，MAP）算法[6-7]，通过交换软信息并且迭代译码的方式获得较好的误码率（bit error rate，BER）性能[8]。但是受限于Turbo 码的迭代译码算法 MAP 算法的复杂性，对数域最大后验概率（maximum a posteriori probability in logarithmic domain，Log-MAP）算法成为主要的实现算法。由于 Log-MAP 算法中对数据的依赖性太强，导致算法在处理时需要依次处理每个信息位中的数据，因此，译码速率有所限制[9-10]。

随着电子技术的发展，硬件水平不断提高，并行译码器的高吞吐量结构设计所需要的资源是能够承载的，所以研究人员提出了增强 Turbo 译码并行性的方法，以此提高 Log-MAP 算法译码器的译码速率和吞吐量[11-14]。文献[12]提出使吞吐量更高、译码速率更快、适用性更广的浮点型全并行译码（fully-parallel Turbo decoder，FPTD）算法。因为每次译码迭代时没有数据依赖，所以FPTD 译码算法有利于实现并行处理，但是要达到目前最先进 Turbo 码译码算法的纠错能力，需要 7 倍的迭代次数才能实现[12]。文献[13]将 FPTD 译码算法应用到专用集成电路（application specific integrated circuit，ASIC）中，吞吐量提高到 21.9 Gbit/s，然而随着全并行译码的计算复杂度的攀升，资源占用量变大，大量的比特位数据需要计算。文献[14]设计了一种新的定点型并行译码方案，并将定点型译码器结构在现场可编程门阵列（field-programmable gate array，FPGA）上实现，消息位占用的硬件资源减少了 50%。虽然文献[14]中译码器的吞吐量、译码速率以及复用性都很强，但译码器在硬件上占用的资源和电路面积相对其他译码器结构设计仍然较大[15-16]，本文研究表明并行译码器还能进一步降低硬件资源的使用。

为了使吞吐率、硬件资源使用以及 BER 性能之间能够更好地平衡，本文在文献[14]的基础上，译码算法中采用并行译码策略，提出了一种将并行算法块中的前向状态度量和后向状态度量的递归计算合并的设计方案。结果表明，每个算法块计算时根据需要选择计算前向状态度量或后向状态度量，这两个状态度量值能同时被计算出来。为了保证 BER 性能，使用了一种适合并行译码算法的简化雅可比对数式 $\max^*$ 的计算方法，这使得本文所提出的方案提高硬件资源使用率的同时，BER 性能和文献[14]中的并行算法相近。

1 并行译码器结构基本原理

在并行 Turbo 码编码器中输入长度为 N 的待编码信息序列。设 k 为时序，$u_k \in \{0,1\}$ 为输入信息位，两个校验比特 $x_k^{p_1}$ 、$x_k^{p_2}$ 和系统比特 x_k^s 构成编码序列，然后进行二进制相移键控调制以及通过噪声方差为 σ^2 的加性高斯白噪声（additive white Gaussian noise，AWGN）信道，将解调后的 3 个比特序列提供给译码器。译码器对应接收 3 个比特序列为 $y_k^{p_1}$、$y_k^{p_2}$、y_k^s，其次按照 Log-MAP 译码算法进行计算，计算表达式如式（1）～式（5）所示。

$$\tilde{\gamma}_k\left(s_{k-1},s_k\right)=\frac{L_c}{2}\left(x_k^s\cdot y_k^s+x_k^{p_1}\cdot y_k^{p_1}\right)+\Lambda_{\mathrm{apr},k}\left(u_k\right) \quad (1)$$

$$\tilde{\alpha}_k\left(s_k\right)=\max_{s_{k-1}}{}^*\left[\tilde{\gamma}_k\left(s_{k-1},s_k\right)+\tilde{\alpha}_{k-1}\left(s_{k-1}\right)\right] \quad (2)$$

$$\tilde{\beta}_{k-1}\left(s_{k-1}\right)=\max_{s_k}^{*}\left[\tilde{\gamma}_k\left(s_{k-1},s_k\right)+\tilde{\beta}_k\left(s_k\right)\right] \quad (3)$$

$$\Lambda_{\mathrm{apo},k}\left(u_k\right)= \max_{\left(s_{k-1},s_k,u_k=1\right)}^{*}\left[\tilde{\alpha}_{k-1}\left(s_{k-1}\right)+\tilde{\gamma}_k\left(s_{k-1},s_k\right)+\tilde{\beta}_k\left(s_k\right)\right]- \max_{\left(s_{k-1},s_k,u_k=0\right)}^{*}\left[\tilde{\alpha}_{k-1}\left(s_{k-1}\right)+\tilde{\gamma}_k\left(s_{k-1},s_k\right)+\tilde{\beta}_k\left(s_k\right)\right] \quad (4)$$

$$\Lambda_{\mathrm{ex},k}\left(u_k\right)=\Lambda_{\mathrm{apo},k}\left(u_k\right)-\Lambda_{\mathrm{apr},k}\left(u_k\right)-\Lambda_{\mathrm{in},k}\left(u_k\right) \quad (5)$$

其中，$L_c=2/\sigma^2$ 表示信道值，$\tilde{\gamma}_k(s_{k-1},s_k)$ 表示 k 时刻状态下的分支度量值，$\tilde{\alpha}_k(s_k)$ 表示 k 时刻状态下的前向状态度量值，$\tilde{\beta}_{k-1}(s_{k-1})$ 表示 $k-1$ 时刻状态下的后向状态度量值。$\Lambda_{\mathrm{apo},k}(u_k)$ 表示后验概率似然比（log-likelihood ratio，LLR），变换后的下标“apr”“ex”“in”分别表示先验信息、外信息和输入信息。

在前向递归计算中，使用式（1）中计算出的分支度量 $\tilde{\gamma}_k(s_{k-1},s_k)$ 和输入的前向状态度量 $\tilde{\alpha}_{k-1}(s_{k-1})$，通过式（2）计算前向状态度量 $\tilde{\alpha}_k(s_k)$。在后向递归计算中，同理使用式（1）计算出的分支度量和输入的后向状态度量 $\tilde{\beta}_k(s_k)$，通过式（3）计算后向状态度量 $\tilde{\beta}_{k-1}(s_{k-1})$。其次，通过计算不同的输入信息位的差值计算得到后验概率似然比，即 $\Lambda_{\mathrm{apo},k}(u_k)$，如式（4）所示。最后，通过式（5）得到外信息 $\Lambda_{\mathrm{ex},k}(u_k)$，其中，$\Lambda_{\mathrm{in},k}$ 在输入信息位为 0 时被计算为 $\Lambda_{\mathrm{in},k}=0$，输入信息为 1 时被计算为 $\Lambda_{\mathrm{in},k}=y_k^s\sigma^2/2$。其中，式（2）～式（4）包含的雅可比对数式 $\max^*$ 计算定义如式（6）所示。

$$\max^*(x_1,x_2)=\max(x_1,x_2)+\ln(1+\mathrm{e}^{-|x_1-x_2|}) \quad (6)$$

这一运算在实际运用的时候，算式 $\max^*$ 被近似为最大值或其他优化方式，通过将纠错性能作为代价，减少并行译码的计算复杂度[17-19]。在本文中，为了降低在硬件中实现的复杂度，将式（6）简化为式（7），研究表明，式（7）在译码性能与计算复杂度之间会有更好的取舍[20]。

$$\max^*(x_1,x_2)\approx\max(x_1,x_2)+0.375 \quad (7)$$

基于以上的推导，要想实现并行译码，就需要译码器同时处理每一次的迭代，将这 3 个比特序列位分别作为系统比特位以及校验比特位提供给交织器上、下两层的分量译码器。该译码器共包含 $2N$ 个算法块，上层和下层译码器各包含 N 个算法块，并行译码器结构如图 1 所示，其中，上标“u”表示上层译码器，“1”表示下层译码器，Reg 表示对数据的存储。

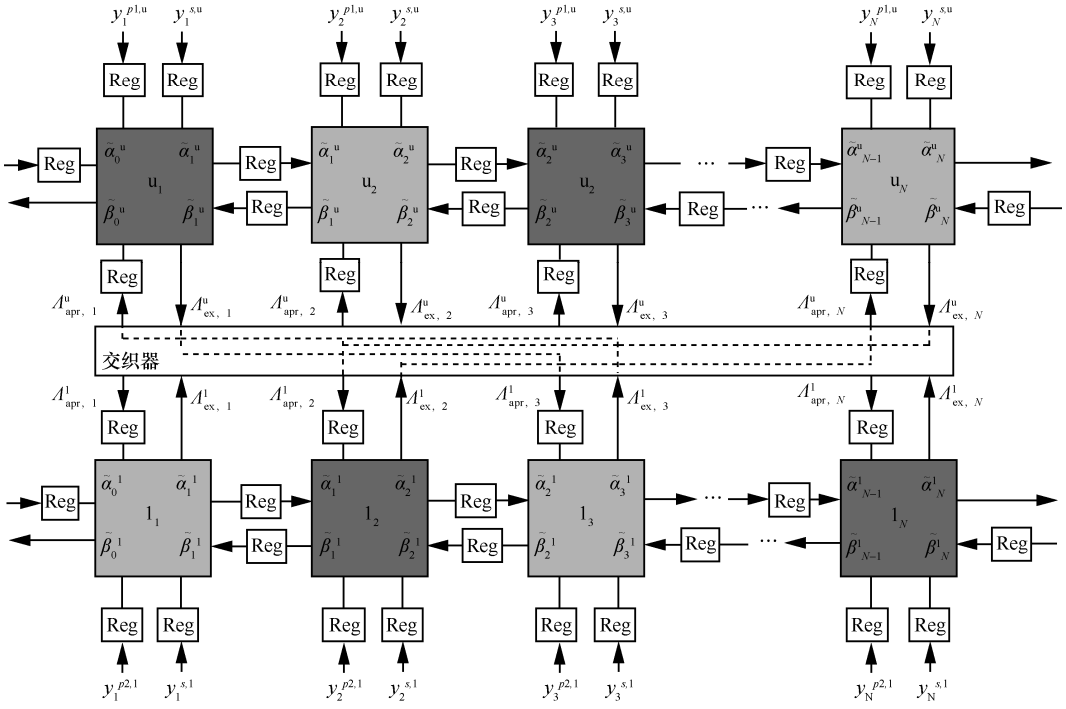

图 1　并行译码器结构

并行译码器中校验比特 $y_k^{p_1,\mathrm{u}}$ 和系统比特 $y_k^{s,\mathrm{u}}$ 被提供给上层译码器的算法块，并且交织器会给算法块提供先验信息 $\Lambda_{\mathrm{apr},k}^{\mathrm{u}}$。同理，下层译码器的算法块也会得到输入的 $y_k^{p_1,1}$、$y_k^{s,1}$ 和 $\Lambda_{\mathrm{apr},k}^{1}$，但 $\Lambda_{\mathrm{apr},k}^{1}$ 是由外信息 $\Lambda_{\mathrm{ex},k}^{\mathrm{u}}$ 的对应比特经过交织器得到的。根据 Turbo 码译码的原理可知，图 1 中的 Turbo 码译码器是迭代操作的，其中，每一次迭代都包含所有算法块，在运算的前半部分迭代中，上层译码器中的奇数算法块和下层译码器中的偶数算法块在第一个时钟周期内同时运行，图 1 中显示为深色阴影部分；后半部分迭代时，上层译码器中的偶数算法块和下层译码器中的奇数算法块在第二个时钟周期中执行，图 1 中显示为浅色阴影部分。

图 1 中虚线表示算法块输出的信息与连接的算法块进行交换，作为下一次解码迭代的先验信息。具体而言，每个分量译码器中算法块输出的前向状态度量和后向状态度量，通过图 1 中的寄存器，作为相邻算法块在下一次迭代过程中的输入。同时，每个分量译码器中算法块输出外信息 $\Lambda_{\mathrm{ex},k}^{\mathrm{u}}$，经过交织器后被用作下层译码器中算法块迭代中先验信息 $\Lambda_{\mathrm{apr},k}^{1}$ 的输入。

2 合并状态度量实现原理

由上文论述和图 1 可知，并行译码算法的计算复杂度主要体现在单个算法块当中，因此，为了减少计算复杂度，进一步减少并行译码器中所占用的资源，在每个算法块中将会对前向状态度量和后向状态度量的递归计算进行合并研究。根据式（2）和式（3）的状态度量计算，可以分析出递归计算与网格图相关。

根据网格图中的 3 个状态 $s_1s_2s_3$ 的取值组合，可以将网格图分为 4 个相同的基本单元，单元 1 为从状态 0 和 1 到状态 0 和 4 的递归，单元 2 为从状态 2 和 3 到状态 1 和 5 的递归，单元 3 为从状态 4 和 5 到状态 2 和 6 的递归，单元 4 为从状态 6 和 7 到状态 3 和 7 的递归。这 4 个基本单元单独计算相应的状态度量，LTE-Advanced 标准下的 Turbo 码网格图分解如图 2 所示。如若依次计算，则前向的递归需要操作 16 次的加减和 $\max^*$ 算式运算。为了减少前向状态度量和后向状态度量分别计算的次数，通过分析前向状态度量和后向状态度量的递推关系，前向状态度量 $\tilde{\alpha}_k(s_k)$ 和后向状态度量 $\tilde{\beta}_{k-1}(s_{k-1})$ 的递推关系如图 3 所示，得出前向状态度量和后向状态度量的递归计算具有相似性，在前向状态度量值递归的过程中，利用 $k-1$ 时刻的度量值 $\tilde{\alpha}_{k-1}(s_0)$ 和 $\tilde{\alpha}_{k-1}(s_1)$ 与分支度量共同递推 k 时刻的状态度量值。后向状态度量值递推过程中，利用 k 时刻的度量值 $\tilde{\beta}_k(s_0)$ 和 $\tilde{\beta}_k(s_4)$ 与分支度量共同递推 $k-1$ 时刻的状态度量值。

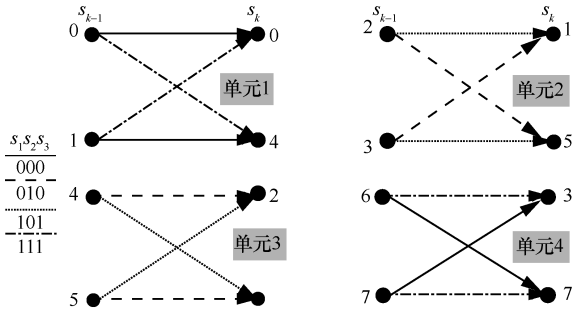

图 2 LTE-Advanced 标准下的 Turbo 码网格图分解

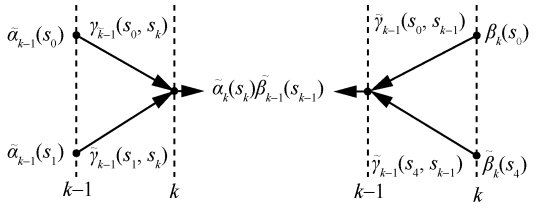

图 3 前向状态度量 $\tilde{\alpha}_k(s_k)$ 和后向状态度量 $\tilde{\beta}_{k-1}(s_{k-1})$ 的递推关系

从图 3 可以得出，后向递推从相反的方向来看与前向递推一致，前向递推对不同状态的度量值进行计算，即 $\tilde{\alpha}_k(s_k)=\max_{s_{k-1}}^{*}[\tilde{\gamma}_k(s_{k-1},s_k)+\tilde{\alpha}_{k-1}(s_{k-1})]$，后向递推按照 $\tilde{\beta}_{k-1}(s_{k-1})=\max_{s_k}^{*}[\tilde{\gamma}_k(s_{k-1},s_k)+\tilde{\beta}_k(s_k)]$ 计算，因此考虑将递归计算合并。

因为网格图的 3 个状态取值分为 4 个基本单元，同时每个基本单元的前向状态度量和后向状

态度量的计算具有对称性，所以在译码该时刻的前向状态度量值的同时可以逆向计算该时刻的后向状态度量值。前后向状态度量合并后的 Turbo 码网格图如图 4 所示，图 4 中黑线代表前向状态度量递归的方向，灰线代表后向状态度量递归的方向。在并行译码的过程中，本文将运用图 4 中合并网格图的方式分为 4 个基本单元计算前向状态度量值和后向状态度量值。并行译码器中的算法块得到上一时刻的前向状态度量值和下一时刻的后向状态度量值，然后将这两个值按照递推的关系在每个算法块中同时进行计算，这样在并行译码时，则每个算法块中的递归计算能进行合并。

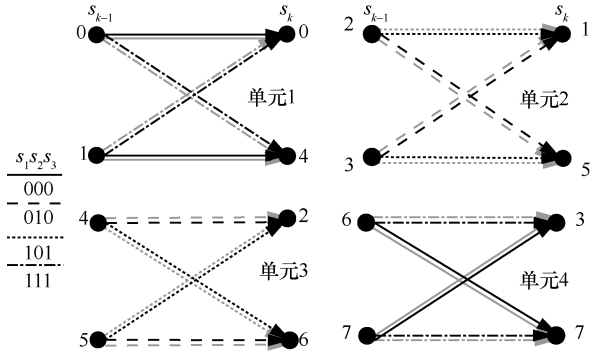

图 4　前后向状态度量合并后的 Turbo 码网格图

因为图 4 中 4 个基本单元的计算方式一致，所以本文针对其中的一个基本单元进行详细分析，基本单元合并计算结构如图 5 所示，其中，黑实线表示前向状态度量的方向，灰虚线表示后向状态度量的方向。

同时在计算时为每个计算单元提供 $k-1$ 时刻的前向度量值和 k 时刻的后向状态度量值，即式（8）中的 x_1、x_2。y_1、y_2 指计算 $k-1$ 时刻的后向状态度量和 k 时刻的前向状态度量。

$$\begin{cases} x_1=\tilde{\alpha}_{k-1}(s_0) \\ x_2=\tilde{\alpha}_{k-1}(s_1) \end{cases} \text{或} \begin{cases} x_1=\tilde{\beta}_k(s_0) \\ x_2=\tilde{\beta}_k(s_4) \end{cases} \\ \begin{cases} y_1=\tilde{\alpha}_k(s_0) \\ y_2=\tilde{\alpha}_k(s_4) \end{cases} \begin{cases} y_1=\tilde{\beta}_{k-1}(s_0) \\ y_2=\tilde{\beta}_{k-1}(s_1) \end{cases} \tag{8}$$

在前向状态度量计算时，图 5 中 m_1、m_2 代表前向状态度量由 $k-1$ 时刻的第 0 状态递推到 k 时刻的分支度量，n_1、n_2 则是由 $k-1$ 时刻的第 1 状态递推到 k 时刻的分支度量，如式（9）所示。

$$\begin{cases} m_1=\tilde{\gamma}_{k-1}(s_0,s_0), n_1=\tilde{\gamma}_{k-1}(s_1,s_0) \\ m_2=\tilde{\gamma}_{k-1}(s_0,s_4), n_2=\tilde{\gamma}_{k-1}(s_1,s_4) \end{cases} \tag{9}$$

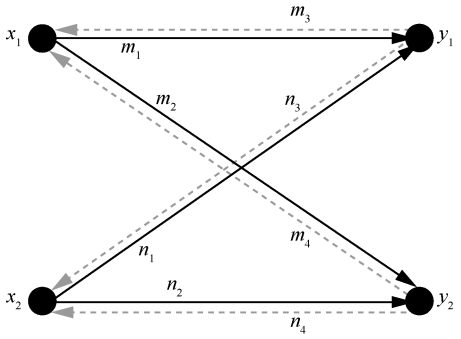

图 5　基本单元合并计算结构

在后向状态度量计算过程中，m_3、m_4 代表后向状态度量由 k 时刻的第 0 状态递推到 $k-1$ 时刻的分支度量，n_3、n_4 则是由 k 时刻的第 4 状态递推到 $k-1$ 时刻的分支度量，如式（10）所示。

$$\begin{cases} m_3=\tilde{\gamma}_k(s_0,s_0), n_3=\tilde{\gamma}_k(s_0,s_1) \\ m_4=\tilde{\gamma}_k(s_4,s_0), n_4=\tilde{\gamma}_k(s_4,s_1) \end{cases} \tag{10}$$

基于式（2）对前向状态度量值的计算，k 时刻的 y_1、y_2 具体计算如式（11）所示。

$$\begin{cases} y_1=\max^*(x_1+m_1,x_2+n_1) \\ y_2=\max^*(x_1+m_2,x_2+n_2) \end{cases} \tag{11}$$

同时基于式（3）的后向状态度量值的计算，$k-1$ 时刻的度量值运用具体计算式如式（12）所示。

$$\begin{cases} x_1=\max^*(y_1+m_3,y_2+n_3) \\ x_2=\max^*(y_1+m_4,y_2+n_4) \end{cases} \tag{12}$$

当合并递归计算完成后，再用于计算后验概率似然比，其次计算外信息位，其中，$\max^*$ 算式的运算均为式（7）的形式。

低资源需求的并行译码器结构设计框图比较如图 6 所示。基于第 1 节以及合并计算的策略，

图 6（b）给出了本文提出的并行译码器结构设计方案中的合并模块，与图 6（a）中并行译码器相比，本文的设计方案节省了大量的逻辑资源，进而影响译码器的整体功耗。该合并计算的结构设计相比已有的结构设计，不同之处在于并行译码器中每个时刻的 8 个前向状态度量和 8 个后向状态度量的计算分为 4 个部分，前向、后向的状态度量以及分支度量合并为一个循环进行计算。

3 高效并行译码器结构设计的 FPGA 实现

基于对上述并行译码算法的分析，对 Turbo 码译码器的整体进行 FPGA 实现，本文方案的并行译码器 FPGA 结构如图 7 所示，译码器模块的两个虚线框表示上层和下层软输入软输出（soft in and soft out，SISO）的译码，中间通过交织器和解交织器实现奇数、偶数索引的衔接。控制模块

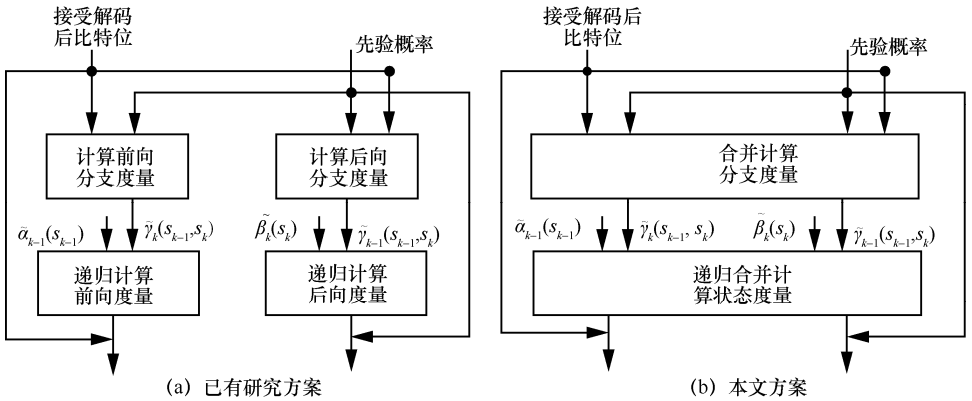

图 6 低资源需求的并行译码器结构设计框图比较

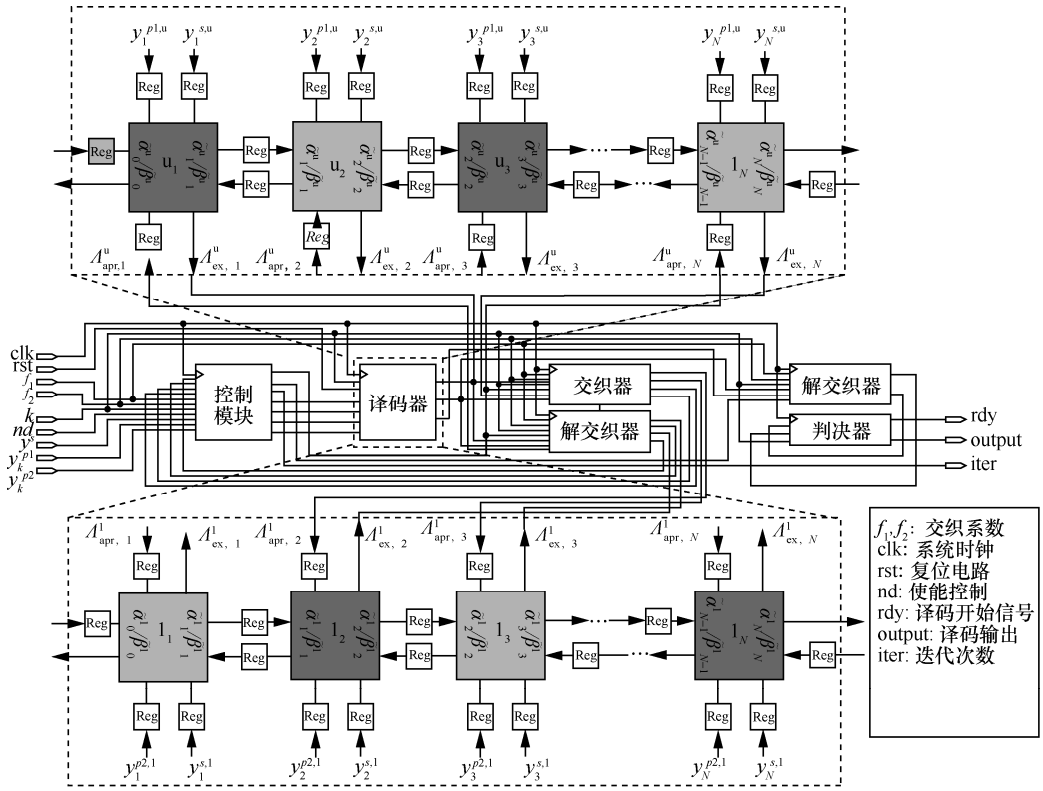

图 7 本文方案的并行译码器 FPGA 结构

对迭代次数、输入输出以及上层下层译码进行控制。图 1 与图 7 的不同之处在于单个算法块中的前后向状态度量计算是否被合并，图 1 算法块内的前向和后向递归计算分为两个模块单独计算，而图 7 的前向后向递归计算合并在一个模块中计算，并分为 4 个相同的单元同步计算。

在并行译码过程中，使用寄存器保存前向状态度量和后向状态度量以及迭代过程中不断更新的先验信息值。并用寄存器保存译码器的输入，即校验比特 $y_k^{p_1,\mathrm{u}}$ 和系统比特 $y_k^{s,\mathrm{u}}$，便于在译码迭代时调用。当前半部分迭代时，上层译码器中的奇数算法块和下层译码器中的偶数算法块同时运行，通过交织后，先验信息传输到后半部分迭代，上层的偶数算法块和下层的奇数算法块开始运行，这样就完成了一次的迭代。基于前文的讨论，为了便于硬件的分析，本文采用了分支度量以及前向后向状态度量的基本计算方式如式(1)～式(3)所示。因为式（2）与式（3）中 max* 算式在硬件实现过程中计算复杂度高，所以本文采用式（7）近似 max* 算式。

针对单个算法块，状态度量合并计算的 FPGA 实现结构如图 8 所示，实现时使用选择和加法操作进行合并。其中，BU1～BU4 代表 4 个基本单元，对应具体结构如左上线框中；S&C 代表选择和控制输出模块，其具体结构对应右上线框中。在单个算法块中递归计算前向状态度量和后向状

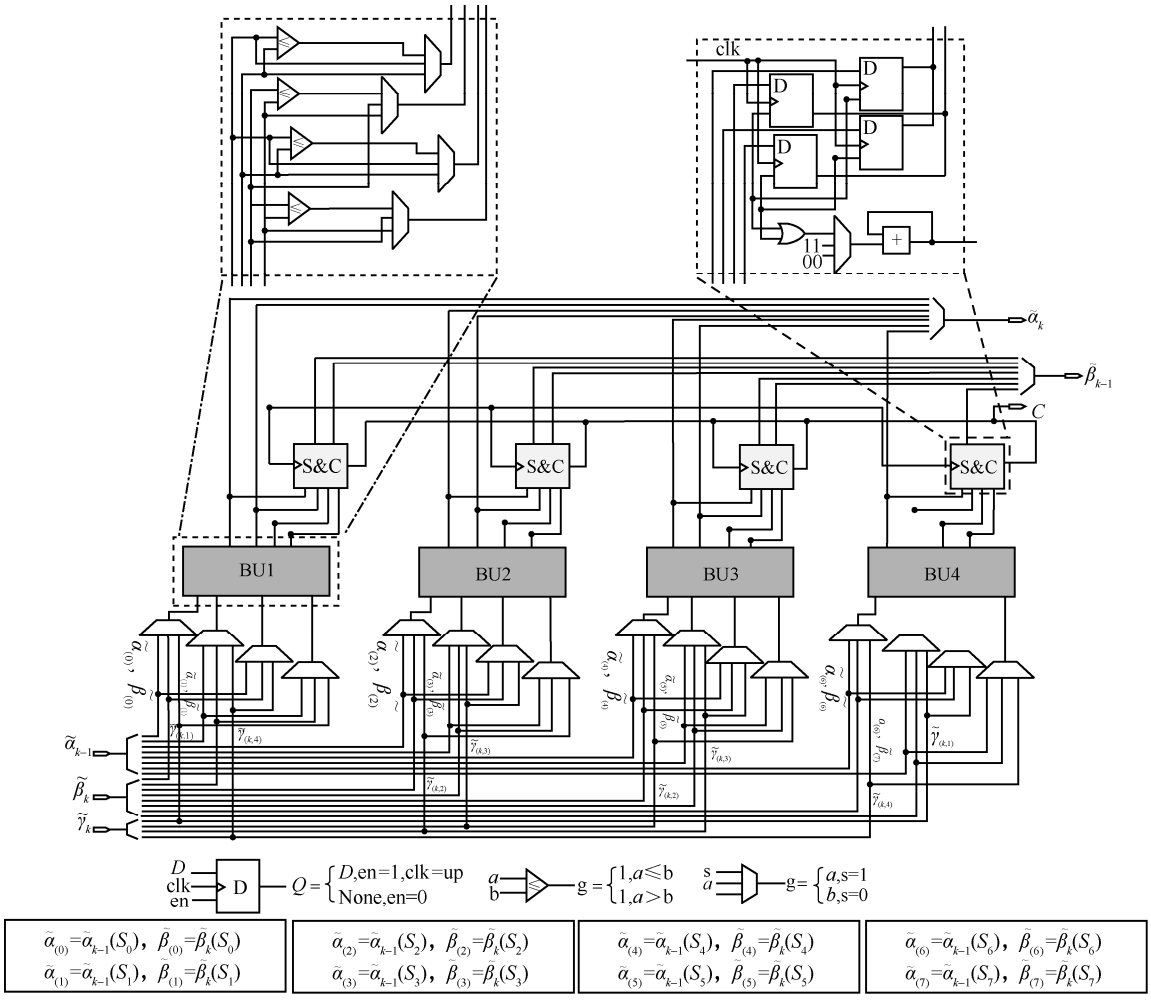

图 8　状态度量合并计算的 FPGA 实现结构

态度量时，通过一个多路选择单元、比较控制单元以及求 $\max^*$ 算式的单元对 $k-1$ 时刻的前向状态度量和 k 时刻的后向状态度量进行选择计算。前向和后向递归计算时共用同一个模块，在输入端通过时钟的上升沿和下降沿控制输入的是前向或后向状态度量值。

同时从寄存器中读出上一个算法块所输出的前向状态度量和后向状态度量。考虑到图 4 的网格图，利用式（1）计算每个时刻的 16 个分支度量，然后输出给状态度量递归模块。在递归模块中，首先将度量寄存器初始化，随着递归过程的持续，度量寄存器通过反馈使用当前计算的状态度量进行更新，然后用于下一轮状态度量递归。随后，本文使用 4 个基本单元 BU1、BU2、BU3 和 BU4 计算 k 时刻的 8 个状态度量。在每个基本单元中，输入前向状态度量、后向状态度量和分支度量通过多路选择器后，利用式（11）和式（12）计算两个状态度量，最终通过 S&C 模块对结果进行选择和控制输出。

4 功耗估算和性能分析

4.1 资源占用比较

为了证明提出的译码体系结构的有效性，本文在 FPGA 上使用硬件描述语言 Verilog HDL 实现了并行 Turbo 码译码器，如图 7 所示。目标设备为 Altera Cyclone IV EP4CE75F23C8 芯片，使用 Quartus II 13.0 软件调试。为了比较的公平性，本文提出的译码结构与基于 Log-MAP 采用式（7）的对数式译码算法的译码结构在实际实现中采用了相同的量化方案。度量值量化方案见表 1，其中，度量值类型中的 q 是量化总比特数，f 是小数位比特数[8]。

表 1 度量值量化方案

度量值类型	$y_k^s, y_k^{p_1}, y_k^{p_2}$	$\tilde{\alpha}_k, \tilde{\beta}_k, \tilde{\gamma}_k$	$\Lambda_{\text{apo},k}$
(q, f)	(5,3)	(10,3)	(11,3)

在 Turbo 码的硬件实现中，资源占用是衡量译码器设计的适用性之一。同时在并行译码器中，单个算法块的资源也是决定整体功耗的关键点[21]。因此，为了更好地分析资源的占用，本文将文献[14]中的并行译码器的单个算法块中状态度量计算模块和本文提出的结构设计方案中的合并计算后的状态度量模块资源使用情况进行对比分析。考虑了两种译码方案 FPGA 实现中主要使用的逻辑单元、寄存器数量以及寄存器容量，状态度量计算模块的资源使用情况比较见表 2。其中，两种译码方案帧长均设为 1 024 bit，同时 $\max^*$ 运算均采用式（7）计算。通过对比资源的使用情况发现，本文提出的结构设计方案能更好地减少计算的复杂度，将算法块的译码器结构逻辑资源占用降低 47.9%，寄存器个数降低 30.8%，寄存器容量减少了 1.8%。因此，本文提出的译码器结构设计方案可以通过节省译码器结构的硬件资源使其利用率提高。

表 2 状态度量计算模块的资源使用情况比较

设计方案	逻辑单元/个	寄存器/个	寄存器容量/bit
并行设计方案	3 044	572	10 240
本文设计的方案	1 583	396	10 060

4.2 功耗分析

本文提出的合并状态度量计算的设计方案最大目的是降低 Turbo 码译码器的整体资源占用和功耗。按照上文中对合并状态度量的并行译码器结构框图的设计，将提出的译码器结构通过 Modelsim 验证，然后采用 PowerPlay Early Power Estimator 测试平台对并行译码器结构中单个算法块的状态度量计算模块功耗进行测试。

通过对并行译码器结构中的单个算法块进行分析，单个算法块的功耗主要是由状态度量的计算所占用的资源决定的，因此在测试时，本文只测试并比较了状态度量计算模块的部分。硬件模块的功耗主要由逻辑计算、RAM 存储、I/O 输入输出，时钟信号和静态功耗组成。静态功耗固定

为 0.127 W。

由于在相同目标器件下，本文将对比各模块中的动态功耗，即逻辑、RAM、I/O 和时钟方面的消耗。因为并行译码器不消耗存储容量，所以 RAM 方面功耗为 0，不同频率下 I/O 和时钟方面的合并状态度量计算模块的功耗对比如图 9 所示。文献[22]表明，逻辑计算的占用对状态度量计算模块的功耗影响较大。因此在功耗测试中，将本文提出的合并状态度量计算模块的逻辑功耗和该模块的功耗分别与已有的并行译码器结构的状态度量计算模块进行了对比，不同频率下与已有并行结构的状态度量计算模块功耗比较如图 10 所示。

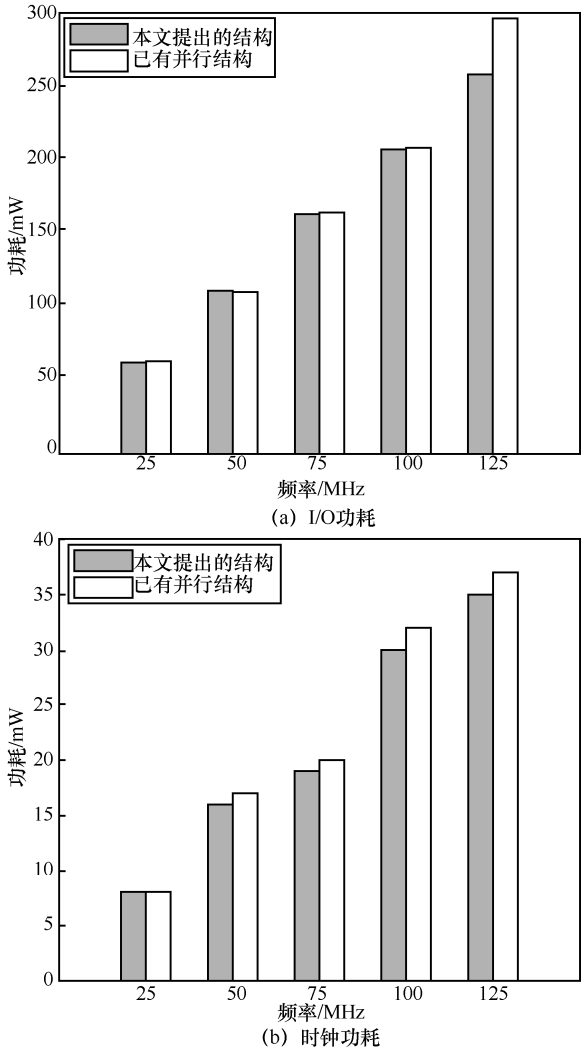

图 9　不同频率下 I/O 和时钟方面的合并状态度量计算模块的功耗对比

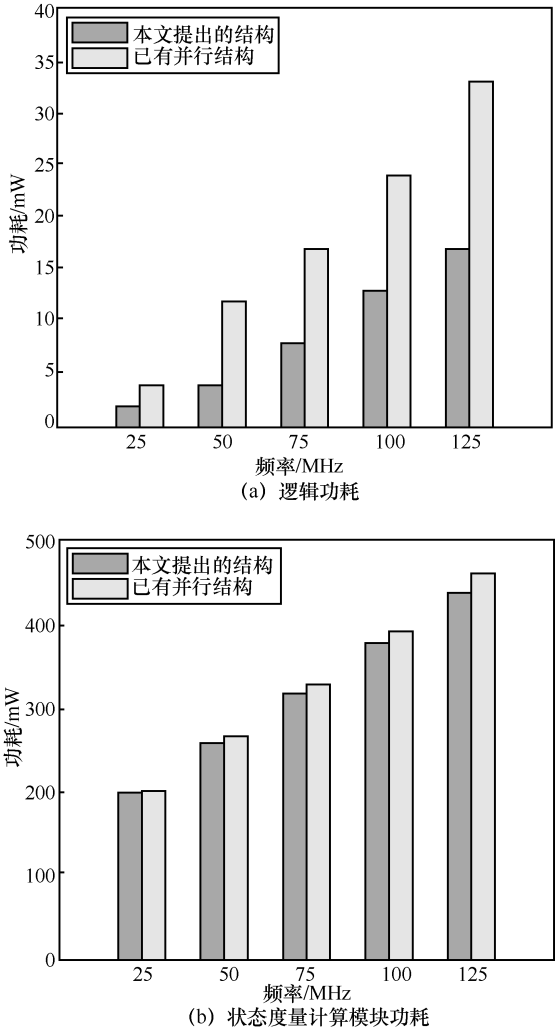

图 10　不同频率下与已有并行结构的状态度量计算模块功耗比较

在不失通用性的前提下，FPGA 实现的时钟频率分别设置为 25 MHz、50 MHz、75 MHz、100 MHz 和 125 MHz 进行对比。前后向的逻辑计算合并后，逻辑计算的功耗在不同频率下最大降低 50%左右；由于逻辑计算的功耗降低从而导致单个算法块中状态度量计算模块的功耗在 125 MHz 下降低 5.26%，同时在 25 MHz、50 MHz、75 MHz、100 MHz、125 MHz 频率约束下状态度量计算模块功耗分别从 201 mW 降到 199 mW、266 mW 降到 258 mW、328 mW 降到 317 mW、391 mW 降到 377 mW、460 mW 降到 437 mW。对于状态度量计算模块来说，功耗测试结果表明逻辑计算功耗随着频率的增

加而不断增大，功耗下降率也随之增加，并且本文设计的译码器结构的状态度量计算模块功耗低于已有的并行译码器结构设计方案中的状态度量计算模块。虽然本文所设计的译码器中单个算法块的资源占用和功耗方面相较有所下降，但是本文所设计的译码器的复用性相较于已有的并行译码器会有所减弱，同时由于芯片使用型号不同，在模拟仿真过程中传输帧的时间以及读写数据的处理时间上会比已有的译码器长一些。

4.3 BER 性能仿真

在仿真过程中，为了验证上述方案的有效性，编码序列按照 LTE-Advanced 标准[5]，构造了码率为 1/3 的 Turbo 编码序列，分别研究基于不同帧长和迭代次数下的 BER 性能。仿真对比了 3 种译码算法，即并行译码算法，并行译码算法采用本文提出的 max* 算式优化以及本文提出的结构设计方案，在帧长为 440 bit 和 1 024 bit 情况下对 BER 性能进行比较，并探索不同的迭代次数对 BER 性能的影响，帧长 440 bit 和 1 024 bit 的不同迭代次数的 BER 性能比较分别如图 11 和图 12 所示。其中的仿真环境设置采用 BPSK 调制方式，以加性高斯白噪声信道作为仿真信道，迭代次数用“*I*”表示。

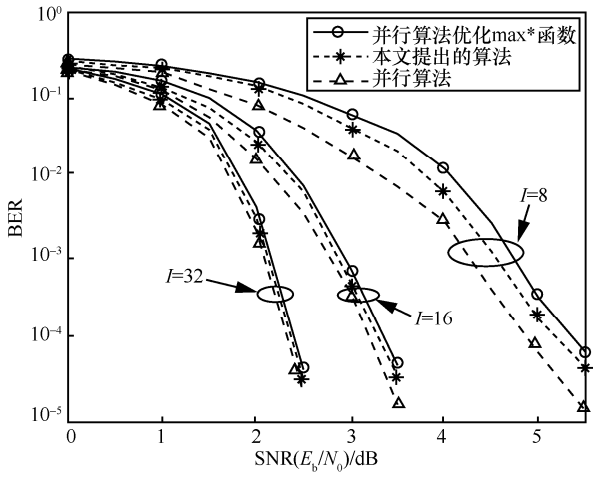

图 11　帧长 440 bit 的不同迭代次数的 BER 性能比较

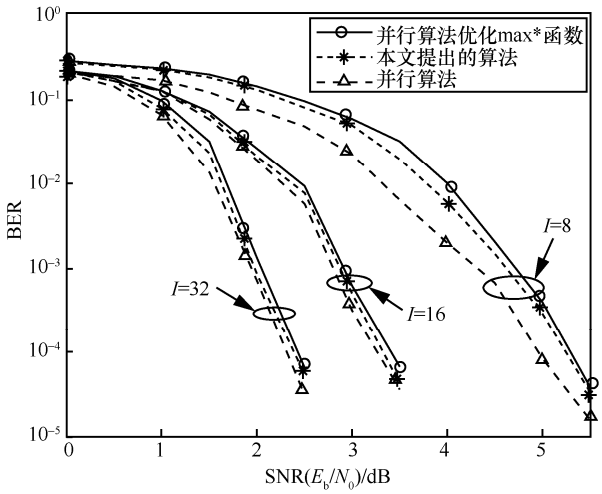

图 12　帧长 1 024 bit 的不同迭代次数的 BER 性能比较

从图 11 可以看出，当帧长为 440 bit、迭代次数为 32 次时，本文提出的译码结构设计算法与并行译码算法的译码性能相近。当 max* 算式优化后应用于并行译码算法时，译码的性能有少量损失，但是简化的并行算法可应用于更多的场景。因此表明该设计方案对 max* 算式优化是有益的。进一步地，随着迭代次数增加，BER 性能逐渐变得更接近并行译码算法。

由图 12 可知，在信噪比相同的情况下，帧长越长译码算法的性能越优。同时在图 11 和图 12 中对同一帧长下不同迭代次数的 BER 性能进行比较，可以证明并行译码能够更好地应用于较长的帧长和迭代次数较多的情况，所以在实际情况下对于帧长和迭代次数的选择要做出合理考虑。因此，本文提出的算法不仅减少了数据传输过程中重复的消耗，还提高了数据传输的可靠性，可应用于高吞吐低功耗无线通信中。

综合以上分析，基于合并状态度量计算的并行译码器结构设计方案，较已有的并行译码器设计方案的结构更为简单，引入的逻辑资源占用更少，硬件的功耗更少。以 BER 性能、计算速率和功率消耗为考查指标，本文基于状态度量合并的译码器结构设计方案，具有明显的总体优势。

5　结束语

在并行译码器的设计过程中，单个算法块的资源占用是决定整个并行译码器功耗的关键点。本文针对并行 Turbo 码译码器，通过对译码网格图的分解，分析前向状态度量和后向状态度量递归计算的方式，研究了优化 $\max^*$ 算式实现方式以及递归计算状态度量的结构，给出了高效并行译码器结构设计，并分析了该结构的 BER 性能以及资源占用。结果表明，本文提出的并行译码器设计结构，对比已有的结构上设计更为简单，虽然在硬件的复用性有所减弱，但是将前向和后向状态度量的递归模块进行合并，可以使该计算模块的资源占用减少 50%左右，同时译码性能较并行算法只有较小的损失。后续考虑在减少资源占用的同时提高复用性，并考虑在不同的通信场合使用。

参考文献：

[1] 田甜, 薛鸿民, 邓志龙. Turbo 码在短波通信系统中的应用研究[J]. 科技资讯, 2020, 18(24): 49-51, 63.
TIAN T, XUE H M, DENG Z L. Application research of turbo code in short wave communication system[J]. Science & Technology Information, 2020, 18(24): 49-51, 63.

[2] 郭臣, 付高原, 杨茂辉. 基于FPGA的Turbo码数据传输系统的实现[J]. 电子测量技术, 2017, 40(10): 221-225.
GUO C, FU G Y, YANG M H. Implementation of Turbo codes data transmission system on the basis of FPGA[J]. Electronic Measurement Technology, 2017, 40(10): 221-225.

[3] FIGUEIREDO F A P, MATHILDE F, CARRILLO D, et al. A framework for the automation of LTE physical layer tests[J]. Wireless Personal Communications, 2018, 102(1): 293-307.

[4] 陈发堂, 刘一帆, 唐成. 一种用于 5G IOT 通信的能量效率方案[J]. 电子技术应用, 2017, 43(11): 2-6, 26.
CHEN F T, LIU Y F, TANG C. An energy-efficient scheme for 5G Internet of Things[J]. Application of Electronic Technique, 2017, 43(11): 2-6, 26.

[5] 3GPP. Multiplexing and channel coding (Release 11): TS 36.212 v11.3.0 [S]. 2013.

[6] 詹明, 文红, 伍军. LTE-Advanced 标准中一种基于反向重算的低存储容量 Turbo 码译码器结构设计[J]. 电子学报, 2017, 45(7): 1584-1592.
ZHAN M, WEN H, WU J. A memory reduced turbo code decoding architecture for LTE-Advanced standard based on reverse recalculation[J]. Acta Electronica Sinica, 2017, 45(7): 1584-1592.

[7] LI Y, XU B J, MA L, et al. High-throughput error correction for continuous-variable quantum key distribution systems based on shuffled iterative decoding[C]//Proceedings of SPIE/COS Photonics Asia. Proc SPIE 11558, Quantum and Nonlinear Optics VII, Online Only. [S.l.:s.n.], 2020, 11558: 77-82.

[8] YOO I, KIM B, PARK I C. Tail-overlapped SISO decoding for high-throughput LTE-advanced turbo decoders[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2014, 61(9): 2711-2720.

[9] LIN J S, SHIEH M D, LIU C Y, et al. Efficient highly-parallel turbo decoder for 3GPP LTE-Advanced[J]. VLSI Design, Automation and Test (VLSI-DAT), 2015: 1-4.

[10] PARVATHY M, GANESAN R. Throughput enhancement of SISO parallel LTE turbo decoders using floating point turbo decoding algorithm[J]. International Journal of Wireless and Mobile Computing, 2018, 15(1): 58.

[11] GONZALEZ-PEREZ L F, YLLESCAS-CALDERON L C, PARRA-MICHEL R. Parallel and configurable turbo decoder implementation for 3GPP-LTE[C]//Proceedings of 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig). Piscataway: IEEE Press, 2013: 1-6.

[12] MAUNDER R G. A fully-parallel turbo decoding algorithm[J]. IEEE Transactions on Communications, 2015, 63(8): 2762-2775.

[13] LI A, XIANG L P, CHEN T H, et al. VLSI implementation of fully parallel LTE turbo decoders[J]. IEEE Access, 2016, 4: 323-346.

[14] LI A, HAILES P, MAUNDER R G, et al. 1.5 Gbit/s FPGA implementation of a fully-parallel turbo decoder designed for mission-critical machine-type communication applications[J]. IEEE Access, 2016(4): 5452-5473.

[15] TRAN M T, GAUTIER M, CASSEAU E. On the FPGA-based implementation of a flexible waveform from a high-level description: application to LTE FFT case study[C]//Proceedings of Cognitive Radio Oriented Wireless Networks. [S.l.:s.n.], 2016.

[16] MURUGAPPA P, BAZIN J N, BAGHDADI A, et al. FPGA prototyping and performance evaluation of multi-standard Turbo/LDPC Encoding and Decoding[C]//Proceedings of 2012 23rd IEEE International Symposium on Rapid System Prototyping. Piscataway: IEEE Press, 2012: 143-148.

[17] 孙增友, 李欢欢, 王蒙, 等. 采用近似 max*运算的 Log-MAP 译码算法[J]. 计算机应用与软件, 2016, 33(3): 255-258.
SUN Z Y, LI H H, WANG M, et al. Log-map decoding algorithm using approximate max*operation[J]. Computer Applications and Software, 2016, 33(3): 255-258.

[18] 付婉, 杨茂辉, 胡明亮, 等. Turbo 码译码算法理论推导及误码性能分析[J]. 电子测量技术, 2018, 41(11): 10-14.
FU W, YANG M H, HU M L, et al. Theoretical derivation and error performance analysis of Turbo decoding algorithm[J]. Electronic Measurement Technology, 2018, 41(11): 10-14.

[19] SYBIS M. Chebyshev inequality based max* approximation for reduced complexity decoding of turbo TCM[C]//Proceedings of 2010 6th International Symposium on Turbo Codes & Iterative Information Processing. Piscataway: IEEE Press, 2010: 265-269.

[20] MISHRA S, SHUKLA H, MADHEKAR S. Implementation of Turbo decoder using MAX-LOGMAP algorithm in VHDL[C]//Proceedings of 2015 Annual IEEE India Conference. Piscataway: IEEE Press, 2015: 1-6.

[21] GAO Z, ZHANG L L, YAN T, et al. Design of SEU-tolerant turbo decoders implemented on SRAM-FPGAs[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28(12): 2563-2572.

[22] ZHAN M, PANG Z B, YU K, et al. Reverse calculation-based low memory turbo decoder for power constrained applications[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(6): 2688-2701.

[作者简介]

张茜（1996− ），女，西南大学硕士生，主要研究方向为信号与信息处理。

詹明（1975− ），男，博士，西南大学学院教授、博士生导师，中国电子学会会员，主要研究方向为信道编码理论与技术、无线传感器网络、超高性能工业无线控制。

章坚武（1961− ），男，博士，杭州电子科技大学教授、博士生导师，中国电子学会高级会员，浙江省通信学会常务理事，主要研究方向为移动通信、多媒体信号处理与人工智能、通信网络与信息安全。

王富龙（1995− ），男，西南大学硕士生，主要研究方向为信号与信息处理。

冯云开（1995− ），男，西南大学硕士生，主要研究方向为信号与信息处理。

唐浩（1996− ），男，西南大学硕士生，主要研究方向为信号与信息处理。