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

詹 明[1],文 红[2],伍 军[3]
(1.西南大学电子信息工程学院,重庆 400715;2.电子科技大学通信抗干扰技术国家级重点实验室,四川成都 611731;3.上海交通大学网络空间安全学院,上海 200240)

摘 要: 在LTE-Advanced标准中,为满足移动环境下的低功耗要求,低存储容量的译码器结构设计引起了广泛关注.本文在分解Turbo码网格图的基础上,研究了前向状态度量的反向重算方法,提出了一种基于反向重算的低存储容量译码器结构设计方案.在Log-MAP算法下研究了一种适合反向重算的修正雅可比对数式实现方法,推导了反向重算的数学表达式,并给出了实现结构.结果表明,所涉及的反向重算译码结构,以很小的冗余计算为代价将存储容量降低了50%,译码性能非常接近Log-MAP算法,在冗余计算复杂度、存储容量和译码性能指标上具有更好的均衡性.

关键词: LTE-Advanced标准;Turbo码;MAP算法;低存储容量译码器结构
中图分类号: TN911.22 **文献标识码:** A **文章编号:** 0372-2112 (2017)07-1584-09
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.07.006

A Memory Reduced Turbo Code Decoding Architecture for LTE-Advanced Standard Based on Reverse Recalculation

ZHAN Ming[1], WEN Hong[2], WU Jun[3]
(1. *College of Electronic and Information Engineering, Chongqing* 400715, *China*;
2. *National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu, Sichuan* 611731, *China*;
3. *School of Cyber Security, Shanghai Jiao Tong University, Shanghai* 200240, *China*)

Abstract: In the LTE-Advanced standards, to satisfy the low-power dissipation requirement in mobile scenarios, a decoder with small memory size has attracted extensive attention. By decomposing the trellis diagram of the adopted turbo code, this paper proposes a memory reduced decoding architecture based on reverse recalculation. A modified Jacobian logarithm is specially investigated for the reverse recalculation, and the reverse recalculation in logarithmic domain and the realization structure are also presented. It shows that at the price of low redundant calculation complexity, the memory size is reduced by 50%, while the decoding performance is very close to that of the Log-MAP algorithm. The proposed decoding scheme is superior to other decoding architectures in terms of dummy computation complexity, memory size and decoding performance.

Key words: LTE-advanced standard; Turbo code; MAP algorithm; memory reduced decoding architecture

1 引言

Turbo码逼近Shannon极限的纠错性能[1],因而在无线通信系统中得到了广泛的应用.目前,Turbo码已为LTE-Advanced标准所采用[2,3],以保证高速、可靠的数据传输.LTE-Advanced标准中定义了Turbo码,而没有定义译码器,因此,Turbo码译码器设计成了业界的热点研究内容.在Turbo码译码器结构设计的研究中,

收稿日期:2015-10-06;修回日期:2016-09-15;责任编辑:李勇锋
基金项目:国家自然科学基金(No.61572114,No.61671390,No.61401273);国家博士后科学基金(No.2015M570776);西南大学基本科研专项资金重点项目(No.XDJK2016B002)

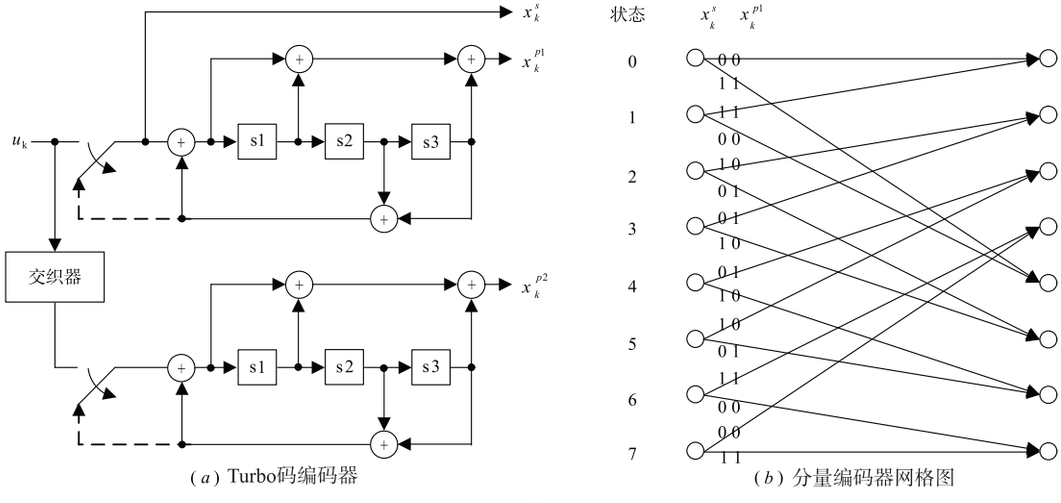
误码率(Bit Error Rate,BER)、吞吐率(Throughput)和功耗(Power Dissipation)是三个同等重要的指标[4~6]. 为获得满意的 BER 性能,Turbo 译码器实现主要采用对数域最大后验概率算法[7](Maximum a Posteriori Probability Algorithm in Logarithmic Domain:Log-MAP). 由于该算法自身的属性,译码器中需要大容量的状态度量缓存(State Metric Cache,SMC). 在采用 Turbo 码的无线接收机中,译码器功耗占整个接收机功耗的一半左右,而译码器功耗的 50% 以上用于对 SMC 的访问操作;进一步地,大容量的 SMC 使得译码器芯片面积较大,增加了芯片的静态漏电流功耗[5,8,9]. 因此,降低 SMC 容量,是低功耗 Turbo 码译码器结构设计研究中的重要内容.

随着微电子技术的发展,深度亚微米技术使得算术运算的功耗远小于对 SMC 访问操作所需的功耗[9]. 以增加冗余计算为代价,代替对 SMC 的访问操作,是一个非常有效的低存储容量译码器结构设计策略. 根据降低存储容量方式的不同,可分为两类:一类是采用反向计算的设计,D S Lee 等提出了状态度量的反向计算试探设计方案[9],只有那些被认为不能反向计算的状态度量值才会存放在 SMC 中,并设置特殊的标志寄存器做识别. 另一类是采用变换法的设计,L C Hung 等提出了基-4 追溯计算法[10],将 8 个状态度量值转换为 6 个差值度量和 2 个比特的符号位,SMC 容量降低了 20%;M Martina 等人研究了 Wash-Hadamard 变换法[11],将状态度量压缩变换为位宽更小的值,SMC 容量降低了约 50%. 然而,反向计算试探设计方案硬件开销大,引入了过多的冗余计算量,在低信噪比(Signal to Noise Ratio,SNR)时尤为明显. 基于 Wash-Hadamard 的变换法虽然降低了更多的 SMC 容量,但本质上属于有损压缩,BER 性能有一定损失,且冗余计算复杂度较高.

为在冗余计算复杂度、SMC 容量、以及 BER 性能之间取得更为理想的均衡性,本文以 LTE-Advanced 标准中的 Turbo 码为研究对象,将其译码网格图做分解,提出了一种前向度量后向重算的译码器结构设计方法. 研究表明,在每个译码时刻的 8 个前向状态度量中,有 4 个不需要存储在 SMC 中. 当需要这 4 个未存储的前向状态度量时,它们可以在后向状态度量递归计算的同时,被重新计算出来. 为保证理想的 BER 性能,提出了一种适合反向重算的雅可比对数式(二变量的 $\max^*$ 函数)简化计算方法,并推导了这种简化 $\max^*$ 函数的反向计算. 以简单的加法、移位和比较操作,给出了 $\max^*$ 函数简化计算和反向计算的实现结构,使得冗余计算复杂度较低,BER 性能与 Log-MAP 算法非常接近.

2　Turbo 码 Log-MAP 译码算法

LTE-Advanced 标准中,Turbo 码编码器如图 1(*a*)所示,其中 s1、s2 和 s3 为移位寄存器状态. 设 k 为时序,$u_k \in \{0,1\}$ 为输入信息位,则系统位 x_k^s、校验位 x_k^{p1} x_k^{p2} 构成码率 1/3 的编码序列[3]. 令 s1s2s3 的十进制数为分量编码器的状态,编码网格图如图 1(*b*)所示.



(*a*) Turbo码编码器　　(*b*) 分量编码器网格图

图1

编码序列 $x_k^s x_k^{p1} x_k^{p2}$ 经二进制相移键控调制后,通过噪声方差为 σ^2 的高斯白噪声信道,则对应的接收机输出软比特序列为 $y_k^s y_k^{p1} y_k^{p2}$. 以 x_k^{p1} 对应的未交织序列分量译码器为例,Log-MAP 译码算法的数学表达式如式(1)表示. 其中 x_k^s、x_k^{p1} 属于 $\{-1,+1\}$,分量编码器状态 s 的下标 $j \in \{0,1,\cdots,7\}$,$s_{j,k}$ 表示第 k 个时序的第 j 个状态,$L_c = 2/\sigma^2$;$\tilde{\gamma}_k^{(z)}(s_{j1,k-1}, s_{j2,k})$ 为分支度量,$\tilde{\alpha}_k(s_{j_2,k})$ 和 $\tilde{\beta}_k(s_{j_1,k})$ 分别为前向和后向状态度量;$\Lambda_{apr,k}^{(z)}(u_k)$、$\Lambda_{apo,k}^{(z)}(u_k)$ 和 $\Lambda_{ex,k}^{(z)}(u_k)$

$$\tilde{\gamma}_k^{(z)}(s_{j_1,k-1},s_{j_2,k})=\frac{L_c}{2}(x_k^s y_k^s+x_k^{p1}y_k^{p1})+\Lambda_{apr,k}^{(z)}(u_k) \quad (1a)$$

$$\tilde{\alpha}_k(s_{j_2,k})=\max_{s_{j_1,k-1}}{}^*[\tilde{\alpha}_{k-1}(s_{j_1,k-1})+\tilde{\gamma}_k^{(z)}(s_{j_1,k-1},s_{j_2,k})] \quad (1b)$$

$$\tilde{\beta}_k(s_{j_1,k})=\max_{s_{j_1,k+1}}{}^*[\tilde{\beta}_{k+1}(s_{j_2,k+1})+\tilde{\gamma}_{k+1}^{(z)}(s_{j_1,k},s_{j_2,k+1})] \quad (1c)$$

$$\Lambda_{apo,k}^{(z)}(u_k)=\max_{(s_{j_1,k-1},s_{j_2,k},u_k=z)}{}^*[\tilde{\alpha}_{k-1}(s_{j_1,k-1})+\tilde{\gamma}_k^{(z)}(s_{j_1,k-1},s_{j_2,k})+\tilde{\beta}_k(s_{j_2,k})]-\max_{(s_{j_1,k-1},s_{j_2,k},u_k=0)}{}^*[\tilde{\alpha}_{k-1}(s_{j_1,k-1})+\tilde{\gamma}_k^{(z)}(s_{j_1,k-1},s_{j_2,k})+\tilde{\beta}_k(s_{j_2,k})] \quad (1d)$$

$$\Lambda_{ex,k}^{(z)}(u_k)=\Lambda_{apo,k}^{(z)}(u_k)-\Lambda_{apr,k}^{(z)}(u_k)-\Lambda_{in,k}^{(z)}(u_k),\quad \begin{cases}\Lambda_{in,k}^{(0)}(u_k)=0\\ \Lambda_{in,k}^{(1)}(u_k)=L_c y_k^s\end{cases} \quad (1e)$$

分别为 $u_k=z, z\in\{0,1\}$ 时的先验概率对数似然比(Log-Likelihood Ratio, LLR),后验概率 LLR 和外信息值. 雅可比对数式 $\max^*(x_1,x_2)$ 函数的定义如式(2)所示[12].

$$\max^*(x_1,x_2)=\ln(\exp(x_1)+\exp(x_2)) \quad (2)$$

3 前向状态度量的反向重算原理与低存储容量译码器结构设计

3.1 基于网格图分解的反向重算

考察图 1(*b*)和式(1*b*)中 $\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{1,3,4,6\}$ 的计算. 从前向的递归可知,$\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{1,3,4,6\}$ 由 $\tilde{\alpha}_{k-1}(s_{j_1,k-1}), j_1\in\{1,3,4,6\}$、$\tilde{\alpha}_{k-1}(s_{j_1,k-1}), j_1\in\{0,2,5,7\}$ 以及对应的分支度量计算得到. 如果从相反的计算方向来看,$\tilde{\alpha}_{k-1}(s_{j_1,k-1}), j_1\in\{0,2,5,7\}$ 可由 $\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{1,3,4,6\}$、$\tilde{\alpha}_{k-1}(s_{j_1,k-1}), j_1\in\{1,3,4,6\}$ 以及相应的分支度量计算得到. 分析 $\tilde{\alpha}_{k-1}(s_{j_1,k-1}), j_1\in\{0,2,5,7\}$ 的反向计算,则可以将 8 个状态分为{0,2,5,7}和{1,3,4,6}两个部分,网格图分解如图 2 所示. 因此,在 $\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{0,1,\cdots 7\}$ 前向递归计算中,可以

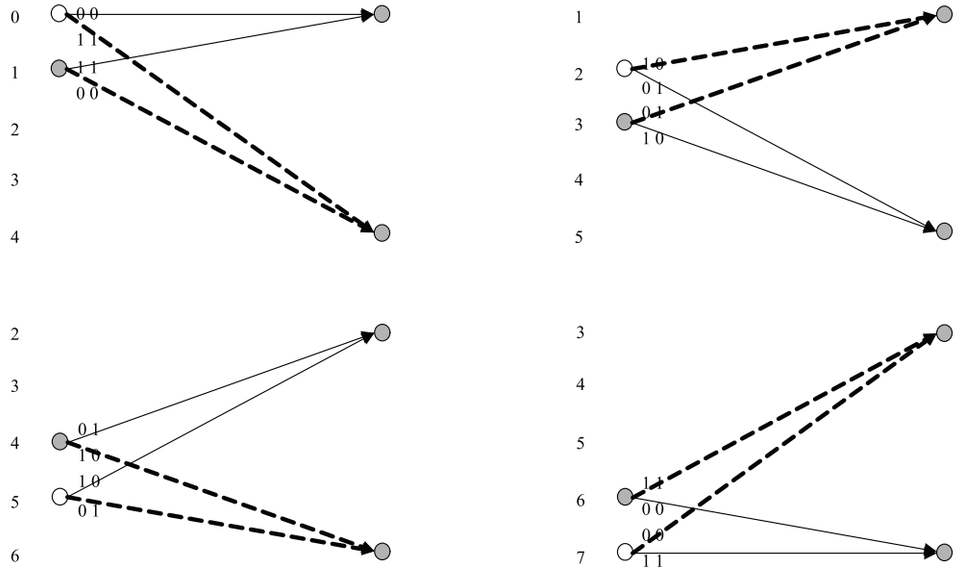


图2 $\tilde{\alpha}_{k-1}(s_{j1,k-1}), j_1\in\{0,2,5,7\}$ 的网格图分解计算

$$\tilde{\alpha}_k(s_{j_2,k})=\max^*[\tilde{\alpha}_{k-1}(s_{m_1,k-1})+\tilde{\gamma}_k^1(s_{m_1,k-1},s_{j_2,k}),\ \tilde{\alpha}_{k-1}(s_{m_2,k-1})+\tilde{\gamma}_k^0(s_{m_2,k-1},s_{j_2,k})] \quad (3)$$

$$\tilde{\alpha}_{k-1}(s_{j_1,k-1})=\ln[\exp(\tilde{\alpha}_k(s_{m_3,k}))-\exp(\tilde{\alpha}_{k-1}(s_{m_4,k-1})+\tilde{\gamma}_k^0(s_{m_4,k-1},s_{4,k}))]-\tilde{\gamma}_k^1(s_{j_1,k-1},s_{m_3,k}) \quad (4)$$

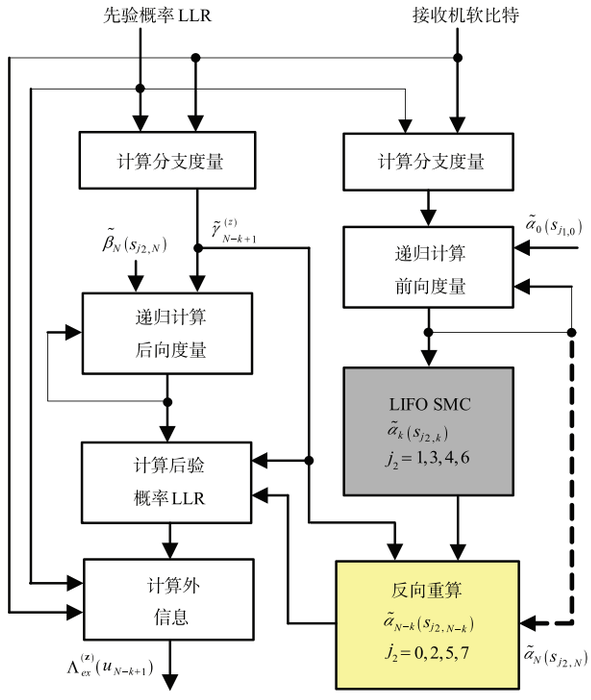
只将 $\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{1,3,4,6\}$ 存储在 SMC 中,式(1*d*)需要 $\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{0,2,5,7\}$ 来计算后验概率 LLR 时,这 4 个值可以利用存储在 SMC 中的 $\tilde{\alpha}_{k+1}(s_{j_1,k+1}), j_1\in\{1,3,4,6\}$、$\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{1,3,4,6\}$ 以及相应的分支度量重新计算出来,而不必存储在 SMC 中.

基于以上分析,可知在式(1*b*)中,$\tilde{\alpha}_k(s_{j_2,k}), j_2\in\{1,3,4,6\}$ 的具体计算可由式(3)代替,其中 $m_1\in\{2,7,0,5\}$, $m_2\in\{3,6,1,4\}$. 再反向重算解得 $\tilde{\alpha}_{k-1}(s_{j1,k-1}), j1\in\{0,2,5,7\}$,如式(4)所示,其中,$m_3\in\{4,1,6,3\}$, $m_4\in\{1,3,4,6\}$.

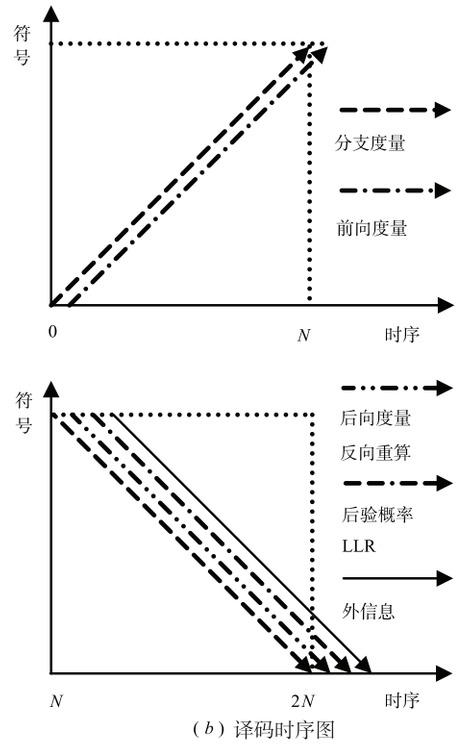
3.2 低存储容量的译码器结构

Turbo 类译码器原理图由交织/解交织器和两个软输入软输出(Soft In and Soft Out, SISO)分量译码器构成[13,14]. 设 N 为译码窗口宽度,$1\leqslant k\leqslant N$,$\tilde{\alpha}_0(s_{j_1,0})$ 和 $\tilde{\beta}_N(s_{j_2,N})$ 分别为前向和后向状态度量的初始值(即当前译码窗口的边界度量值). 基于第 3.1 节的论述,图 3(*a*)给出了一种低存储容量的 SISO 译码器结构设计方案,与经典的译码器相比,本文设计方案有两点不同:(1)后进先出(Last In and First Out:LIFO)的 SMC 中只存储了状态 s 下标 $j_2\in\{1,3,4,6\}$ 的前向状态度量;(2)

在后验概率 LLR 计算模块前,额外插入了一个模块用于反向重算 $j_2 \in \{0,2,5,7\}$ 的前向状态度量. 由于反向重算与后向状态度量递归计算的方向一致,且都需要同一时刻的分支度量 $\tilde{\gamma}^{(z)}_{N-k+1}(s_{j_1,N-k}, s_{j_2,N-k+1})$,因此后向状态度量递归计算模块与反向重算模块可并行工作,译码器工作时序见图 3(*b*)所示.



(*a*) 基于反向重算的SISO分量译码器结构图



(*b*) 译码时序图

图3

4　修正雅克比对数式反向重算实现

4.1　常规的简化 $\max^*$ 函数实现方式反向重算分析

考察第 3.1 节中的反向重算原理,为分析方便,式(3)从左至右依次设各个状态度量值、分支度量值为 y、x_1、c_1、x_2 和 c_2,将式(3)改写成式(5)的形式.

$$y = \max^*(x_1 + c_1, x_2 + c_2) \tag{5}$$

因此,式(4)也可统一表达为式(6)的形式. 反向重算归结为在式(5)的约束条件下,已知 y、x_2、c_2 和 c_1,求解 x_1.

$$x_1 = \ln[\exp(y) - \exp(x_2 + c_2)] - c_1 \tag{6}$$

根据式(1),$\max^*$ 函数是 Log-MAP 算法中最重要的译码操作. 为避免式(2)中复杂的指数和对数计算,雅可比对数式都采取了近似计算的方式. 以一定的误码率损失为代价,换取降低译码复杂度的好处,学者们提出了多种简化的 $\max^*$ 函数实现方法[15~17]. 不同的简化实现方法,式(5)计算出来的值 y 不尽相同;为得到可靠的反向重算结果,式(6)的反向重算实现方式与采用的简化 $\max^*$ 函数是一一对应的. 研究表明,以译码复杂度、占据的芯片面积和 BER 性能为考察指标,文献[12]中提出的 $\max^*$ 函数简化实现方法有明显的优势.

结合式(5),这种简化的 $\max^*$ 函数如式(7)所示.

$$\begin{cases} d_1 = x_1 + c_1 \\ d_2 = x_2 + c_2 \\ y \approx \max\{d_1, 0.75d_1 + 0.25d_2 + 0.5, 0.25d_1 \\ \qquad + 0.75d_2 + 0.5, d_2\} \end{cases} \tag{7}$$

设 d_2 为任意值,以包含 x_1 的变量 d_1 为考察对象,$d_1 - d_2$ 为横坐标,$y - d_2$ 为纵坐标,式(5)的精确计算和式(7)的简化近似计算如图 4 所示. 分析图 4 可知,文献[12]中简化的 $\max^*$ 函数实现,本质为对光滑曲线的四折线逼近. 当 $d_1 - d_2 \geqslant -2$ 时,x_1 可在该区间内求解;而当 $d_1 - d_2 < -2$ 时,d_1 因太小而在 1 区间内被忽略掉,无法求解 x_1. 详细讨论文献[15~17]中的 $\max^*$ 函数实现方式,都存在类似地情况.

4.2　适合反向重算的 $\max^*$ 函数实现

基于第 4.1 节的分析,需构造一个新的雅可比对数式简化计算方法,它能确保任何条件下,d_1 都不会被忽略. 为此,提出了如式(8)所示的 $\max^*$ 函数简化计算方法.

$$y = \begin{cases} \max\{d_1, d_2\} + \ln(1 + e^{-|d_2 - d_1|}), d_2 \geqslant d_1 \\ \min\{d_1, d_2\} + \ln(1 + e^{|d_2 - d_1|}), d_2 < d_1 \end{cases} \tag{8}$$

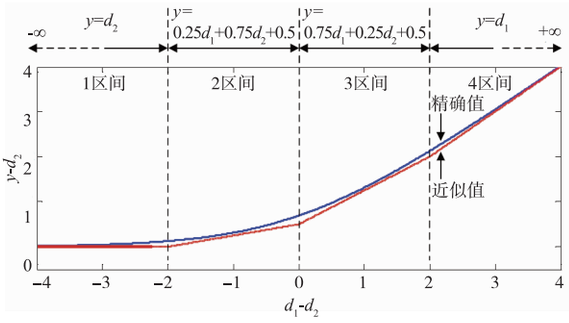


图4 文献[12]中max* 函数简化计算的比较

设 $x=|d_2-d_1|$,则式(8)有两个校正函数:$f_1(x)=\ln(1+e^{-x})$和$f_2(x)=\ln(1+e^{x})$. 为便于低复杂度实现和反向重算,采用折线对校正函数做逼近,如式(9)所示.

$$\begin{cases} f_1(x)\approx\max\{-0.5x+0.6875,-0.125x+0.40625,\\ \qquad -0.0078125x+0.0625\} \\ f_2(x)\approx\max\{0.75x+0.6875,x\} \end{cases} \tag{9}$$

与精确值相比,校正函数的折线近似比较见图 5. 进一步地,由式(9)的近似计算,结合式(7),式(8)可被近似写为式(10).

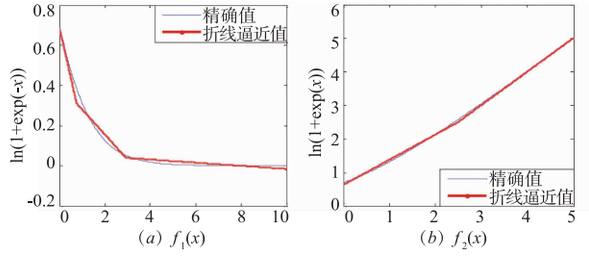


图5 校正函数的折线近似比较

$$y\approx\begin{cases} \max\{d_1,d_2\}+\max\{-0.5x+0.6875,-0.125x \\ \qquad +0.40625,-0.0078125x+0.0625\}\big|_{d_2\geqslant d_1} \\ \min\{d_1,d_2\}+\max\{0.75x+0.6875,x\}\big|_{d_2<d_1} \end{cases} \tag{10}$$

当 $d_1=d_2$ 时,由式(10)表达式解得 $y=d_2+0.6875$. 可知当 $y-d_2\leqslant 0.6875$ 时,有 $d_2\geqslant d_1$,校正函数 $f_1(x)$ 的近似被用于计算 max* 函数;而当 $y-d_2>0.6875$ 时,则有 $d_2<d_1$,校正函数 $f_2(x)$ 的近似被用于计算 max* 函数. 因此,分两种情况推导 x_1 的反向重算过程.

(1)当 $y-d_2\leqslant 0.6875$ 时,由式(10)得到 y 的具体计算式:

$$y=d_2+\max\{-0.5x+0.6875,-0.125x+0.40625,\\ -0.0078125x+0.0625\} \tag{11}$$

注意到此时 $x=d_2-d_1$,代入式(11)中:

①当 $y=d_2-0.5x+0.6875$ 时,解得 $d_1=2y-d_2-1.375$,再代入 $d_1=x_1+c_1$:

$$x_1=d_2+2(y-d_2)-1.375-c_1 \tag{12}$$

②当 $y=d_2-0.125x+0.40625$ 时,同理可得:

$$x_1=d_2+8(y-d_2)-3.25-c_1 \tag{13}$$

③当 $y=d_2-0.0078125x+0.0625$ 时,可得:

$$x_1=d_2+8(y-d_2)-8-c_1 \tag{14}$$

(2)当 $y-d_2>0.6875$ 时,由式(10)得到 y 的具体计算式:

$$y=d_2+\max\{0.75x+0.6875,x\} \tag{15}$$

①当 $y=d_2+0.75x+0.6875$ 时,$x=d_1-d_2$,可得:

$$x_1=d_2+\frac{4(y-d_2)-2.75}{3}-c_1 \tag{16}$$

②当 $y=d_2+x$ 时,可得:

$$x_1=d_2+(y-d_2)-c_1 \tag{17}$$

综合上述两种情况,将式(12)、(13)和(14),以及式(16)、(17)分别写成统一的数学表达式,则反向重算 x_1 的精确表达式如式(18)所示.

$$x_1=\begin{cases} d_2+\min\{2(y-d_2)-1.375,8(y-d_2) \\ \qquad -3.25,128(y-d_2)-8\}\big|_{y-d_2\leqslant 0.6875}-c_1 \\ d_2+\min\left\{\dfrac{4(y-d_2)-2.75}{3},(y-d_2)\right\}\Bigg|_{y-d_2>0.6875}-c_1 \end{cases} \tag{18}$$

4.3 修正 max* 函数及反向重算实现结构

在 Turbo 类译码器的实现中,状态度量采用了(10,3)的二进制量化方案[19]((q,f)中 q 是量化总比特数,f 是小数位比特数),即可保证较好的 BER 性能. 式(11)提出的修正 max* 函数实现方式,其参数经过优化,采用简单的移位、加法和比较操作即可完成. 度量差值 x 的最大右移比特数为 7,小于量化方案中度量值的总比特数. 式(18)能精确的重新计算出 x_1,但式(16)中存在乘 1/3 的操作,为避免复杂的乘法操作,将 1/3 修改为 0.25+0.0625+0.03125(即 0.34375). 仿真表明,替换对 BER 性能的影响很小(请参见 5.3 分析).

根据以上分析,给出了修正 max* 函数实现方式和其反向重算的实现结构图,分别如图 6、图 7 所示.

5 性能分析

5.1 冗余计算的复杂度分析

低存储容量 Turbo 码译码器结构设计研究中[4,9,10,18,19],学者们在状态度量值(前向或后向)递归计算模块后边,插入了一个模块将状态度量值其变换压缩为占据 SMC 容量更小的变换值;而在 SMC 之后,又增加了一个重构解压模块计算原状态度量值. 本文的设计方案中,通过修正状态度量递归计算中必须的

max* 函数实现方式引入反向重算,因而在状态度量递归计算模块之后不需要插入冗余计算,使得译码器结构更为简单,如图 8 所示.

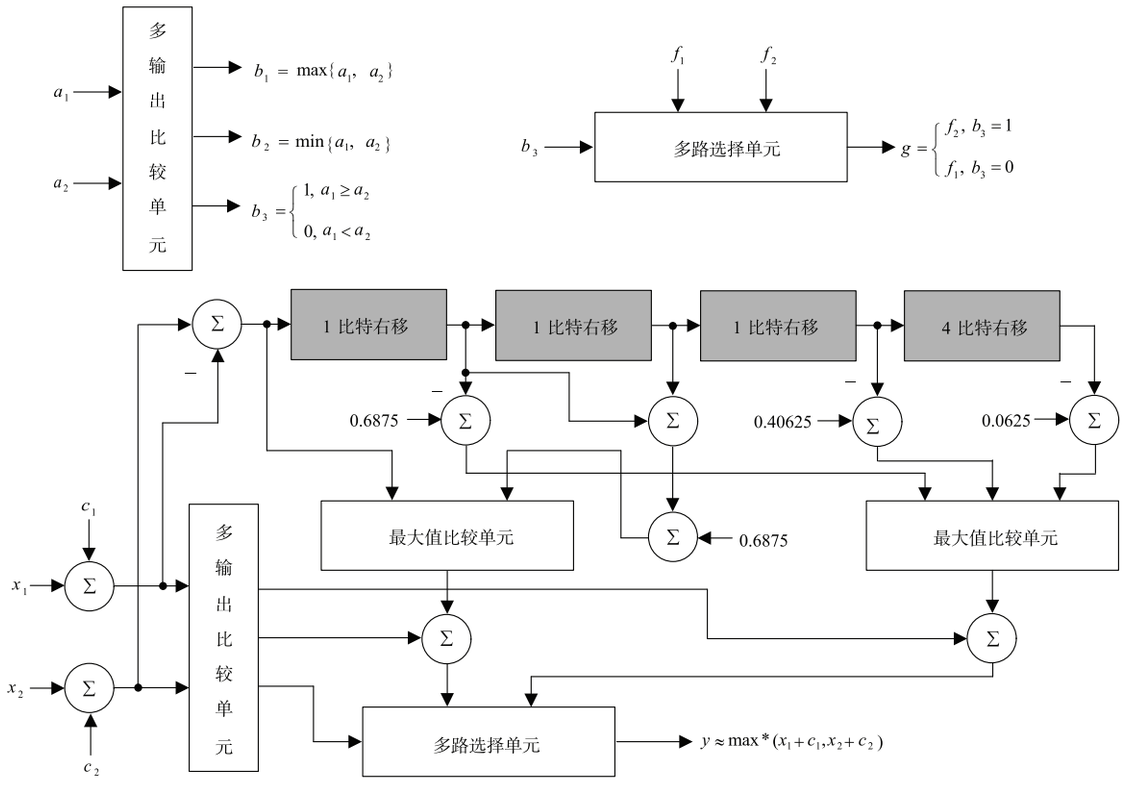


图6　修正 max* 函数的实现结构图

图 8(*b*)中的反向重算模块,采用简单的比较、移位、加法和相应的控制单元即可完成(图 7). 表 1 统计了图 6 修正 max* 函数实现结构和图 7 反向重算实现结构的计算复杂度. 需要指出的是,修正 max* 函数实现是译码算法中必须的操作,因而表 1 中所对应的操作不属于冗余计算复杂度.

本文选择降低 SMC 容量相同的沃尔什-哈达玛变换[11](Walsh-Hadamard Transform, WHT)设计方案为比较对象. 该方案设计引入了 WHT 变换、量化压缩和对应的解压缩、WHT 反变换等复杂操作,其计算复杂度如表 2 所示. 每个译码时刻,表 1 中的反向重算需执行 4 次,而表 2 中的量化压缩、解压缩等操作需执行 8 次. 因此,本设计方案引入了较低的冗余计算复杂度.

表 1　修正 max* 函数与反向重算结构的计算复杂度

操作 / 实现方法	比较 max(*a*,*b*)	1 比特移位(左\右)	加法	多输出比较单元	比较控制单元	多路选择单元
修正 max* 函数实现	3	7	10	1	0	1
反向重算实现	3	12	8	0	1	1

表 2　基于 WHT 变换压缩设计方案引入的冗余计算复杂度

操作 / 插入模块	WHT 变换加法	先导 1 检测器	左/右移位寄存单元	内置编/解码器	内置门电路	多路比较选择单元	其他加法
WHT 量化压缩模块	24	1	1	1	2	1	1
WHT 解量化压缩模块	24	0	1	1	3	1	0

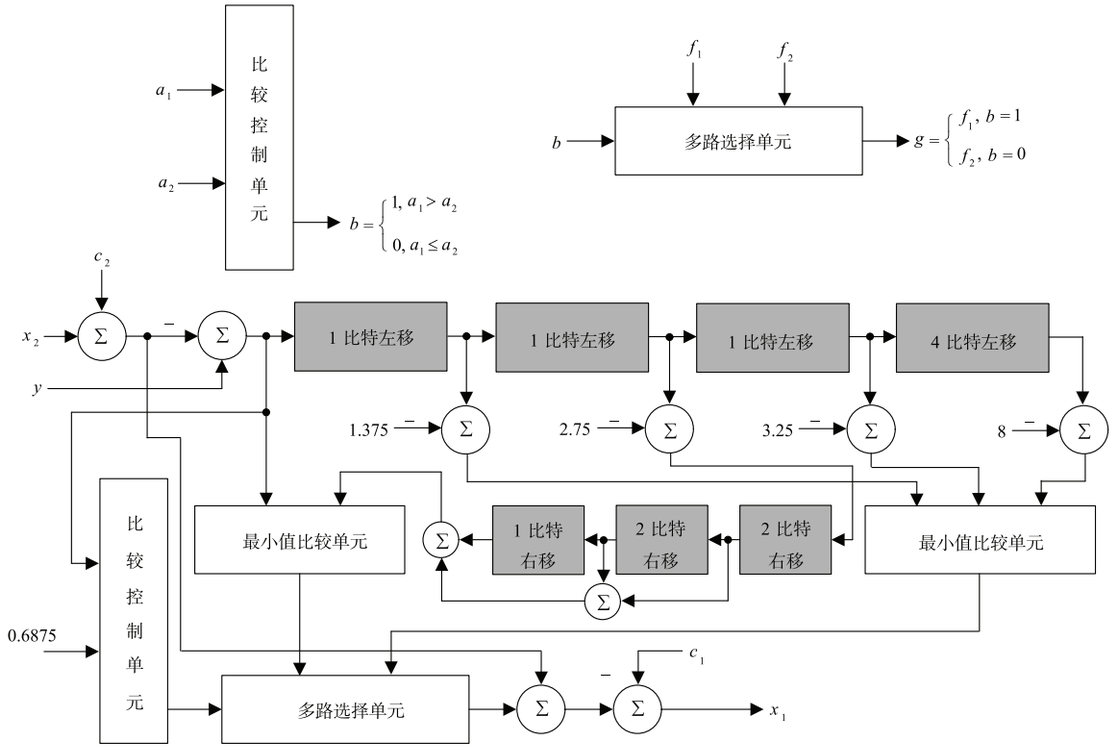


图7 反向重算的实现结构图

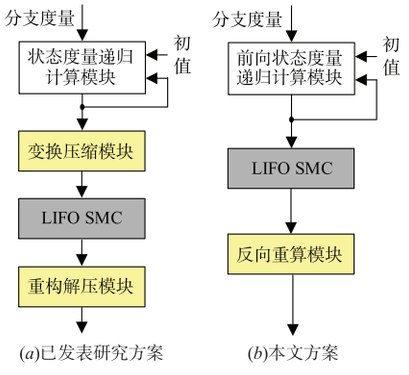


图8 低存储容量译码器结构设计比较框图

5.2 SMC 容量比较

Turbo 译码器的硬件实现中,SMC 容量是决定整体功耗的关键点. 为便于分析,以基-4 追溯计算[10]和基于 WHT 变换的状态度量压缩译码器设计方案为比较对象. 对于 LTE-Advanced 标准中的 Turbo 码,图 1(*b*)表明每个译码时刻有 8 个前(后)向状态度量值. 在基-4 追溯计算设计方案中,将 8 个状态度量值转化为 6 个同位宽的差值度量和 4 个比特的符号位,将 SMC 容量降低了 20%. 基于 WHT 变换的状态压缩设计方案,采用了最大值 Log-MAP(Max-Log-MAP)译码算法,对状态度量值做 WHT 变换后,再做非均匀量化处理,使得变换值位宽为 5 比特,将 SMC 容量降低了 50%. 本文的反向重算译码器结构设计方案,每个译码时刻只需将 4 个状态度量值存储在 SMC 中,且不必做变换压缩等冗余计算,SMC 容量降低了 50%. 比较结果如表 3 所示.

表 3 SISO 译码器的 SMC 容量比较(比特)

SMC 类别 / 设计方案	状态度量(变换度量)	符号比特	合计
经典设计方案	8×10×N	0	80N(100%)
基-4 追溯计算设计方案	6×10×N	4×1×N	64N(80%)
WHT 变换压缩设计方案	8×5×N	0	40N(50%)
本文反向重算设计方案	4×10×N	0	40N(50%)

5.3 BER 性能仿真

根据文献[2]构造码率 1/3 的 LTE-Advanced 标准 Turbo 码,研究不同数据帧长条件下的 BER 性能. 仿真采用了并行窗译码结构,译码窗口宽度 $N=40$,迭代次数为 8,各种度量值的量化方案如表 4[9~11,13,19];为改善 BER 性能,仿真中外信息值 $\Lambda_{ex,k}^{(z)}(u_k)$ 乘上一个度量因子 δ[15],研究了基于 Max-Log-MAP 算法的 WHT 变换压缩结构设计方案,Log-MAP 算法和本文反向重算译码器结构设计方案的 BER 性能. 结果表明,不同帧长条件下,本文所提设计方案 BER 性能非常接近 Log-MAP 算法,且较基于 Max-Log-MAP 算法的 WHT 变换设计方案

提高了约 0.2dB. 图 9 给出了帧长 800 和 1440 两种典型情况的 BER 性能比较.

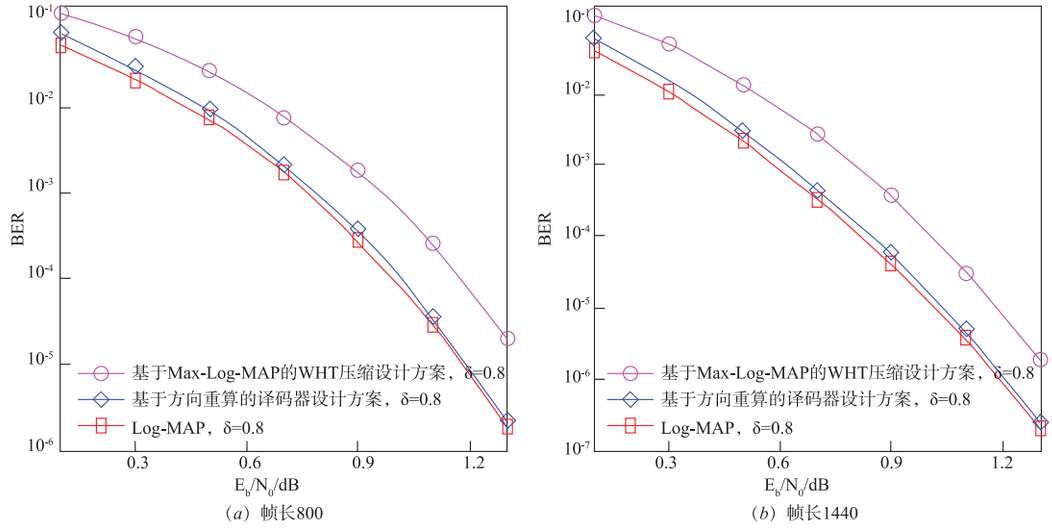


图9　BER性能比较

表 4　采用的度量值量化方案

度量类型	$y_k^s\ y_k^{p1}\ y_k^{p2}$	$\tilde{\gamma}_k^{(z)},\Lambda_{apr,k}^{(z)},\Lambda_{ex,k}^{(z)}$	$\tilde{\alpha}_k,\tilde{\beta}_k$	$\Lambda_{apo,k}^{(z)}$
(q,f)	(5,3)	(9,3)	(10,3)	(11,3)

本文基于反向重算的译码器结构设计方案,采用了逼近误差很小的修正 $\max^*$ 函数实现方法(图 5). 仿真中式(16)中的 1/3 被替换为 0.25 + 0.0625 + 0.03125,引入了约 3% 的误差. 分析式(4)重算可知,$\tilde{\alpha}_{k-1}(s_{j1,k-1})$,$j1 \in \{0,2,5,7\}$ 由 $\tilde{\alpha}_k(s_{m_3,k})$,$m_3 \in \{4,1,6,3\}$、$\tilde{\alpha}_{k-1}(s_{m_4,k-1})$,$m_4 \in \{1,3,4,6\}$ 以及对应的分支度量重算得到,不存在误差递归传播的缺陷;式(18)中存在选择操作,3% 的误差仅在 $0.6875 < y - d_2 < 2.75$ 的特定条件下才被引入,使得本文设计方案具有非常接近 Log-MAP 算法的 BER 性能. 进一步地,随着 SNR 的增加,重算误差对 BER 性能的相对影响随之减小,逐渐变得更接近 Log-MAP 算法.

综合以上分析,基于反向重算的译码器结构设计方案,较已有设计方案的结构更为简单,引入的冗余计算复杂度低. SMC 容量降低效果明显优于基-4 追溯计算的设计方案,与采用 WHT 的状态度量变换压缩设计方案相同,但引入冗余计算复杂度低,在 BER 性能上有约 0.2dB 的优势. 以 BER 性能,冗余计算复杂度和 SMC 容量为考察指标,本文基于反向重算的译码器结构设计方案,具有明显的总体优势.

6　总结

在 Turbo 类译码器的实现中,降低 SMC 容量是减小译码器整体功耗的一个重要策略. 本文针对 LTE-Advanced 标准中的 Turbo 码译码器,在分解其译码网格图的基础上,通过分析现有 $\max^*$ 函数简化实现方式的缺陷,针对性地研究了一种修正 $\max^*$ 函数实现方式和反向重算实现结构,给出了对应的低存储容量译码器结构设计方案,并详细分析了该方案的冗余计算复杂度、SMC 容量以及 BER 性能. 结果表明,本文提出的译码器结构设计方案,较已有报到的低存储容量译码器结构设计方案更为简单,引入的反向重算模块计算复杂度低,SMC 容量减小了 50%,译码性能较最优的 Log-MAP 算法只有很小降低.

参考文献

[1] Berrou C, Glavieux A, Thitimajshima P. Near Shannon limit error-correcting coding and decoding: turbo-codes[A]. International Conference on Communications 1993[C]. Geneva, Switzerland: IEEE Computer Society, 1993. 1064 – 1070.

[2] 3GPP TS 36.212 v11.3.0, 3rd Generation partnership project: Multiplexing and Channel Coding (Release 11)[S].

[3] 3GPP TS 36.212 v9.2.0, 3rd Generation partnership project: Multiplexing and Channel Coding (Release 9)[S].

[4] 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.

[5] Liu H S, Diguet J P, Jego C, et al. Energy efficient turbo decoder with reduced state metric quantization[A]. Workshop on Signal Processing Systems 2007[C]. Shanghai, China: IEEE Computer Society, 2007. 237 – 242.

[6] Lin J S, Shieh M D, Liu C Y, et al. Efficient high-parallel turbo decoder for 3GPP LTE-Advanced[A]. International

Symposium on VLSI Design, Automation and Test 2015 [C]. Hsinchu, Taiwan: IEEE Computer Society, 2015. 1 -4.

[7] Papaharalabos S, Sweeney P, Evans B G. Constant Log-MAP algorithm for duo-binary turbo codes[J]. Electronics Letters, 2006, 42(12): 709 - 710.

[8] Schurgers C, Catthoor F, Engels M. Memory optimization of MAP turbo decoder algorithms[J]. IEEE Transactions on VLSI Systems, 2001, 9(2): 305 - 312.

[9] Lee D S, Park I C. Low-power Log-MAP decoding based on reduced metric memory access[J]. IEEE Transactions on Circuits and Systems-I: Regular Papers, 2006, 53(6): 1244 - 1253.

[10] Lin C H, Chen C Y, Wu A Y. Low-power memory-reduced traceback MAP decoding for double-binary convolutional Turbo decoder[J]. IEEE Transactions on Circuits and Systems-I: Regular Papers, 2009, 56(5): 1005 - 1016.

[11] Martina M, Masera G. State metric compression techniques for turbo decoder architectures[J]. IEEE Transactions on Circuits and Systems, 2011, 58(5): 1119 - 1128.

[12] Papaharalabos S, Mathiopoulos P T, Masera G, et al. On optimal and near-optimal turbo decoding using generalized max* operator[J]. IEEE Communications Letters, 2009, 13(7): 522 - 524.

[13] Lin C H, Wei C C. Efficient window-based stopping technique for double-binary turbo decoding[J]. IEEE Communications Letters, 2013, 17(1): 169 - 172.

[14] Belfanti S, Roth C, Gautschi M, et al. A 1Gbps LTE-Advanced turbo-decoder ASIC in 65nm CMOS[A]. Symposium on VLSI Circuits 2013[C]. Kyoto, Japan: IEEE Computer Society, 2013. 284 - 285.

[15] Vogt J, Finger A. Improving the max-log-MAP turbo decoder[J]. Electronics Letters, 2000, 36(23): 1937 - 1938.

[16] Classon B, Blankenship K, Desai V. Channel coding for 4G systems with adaptive modulation and coding[J]. IEEE Wireless Communications, 2002, 9(4): 8 - 13.

[17] Wang H, Yang H W, D C Yang. Improved Log-MAP decoding algorithm for turbo-like codes[J]. IEEE Communications Letters, 2006, 10(3): 186 - 188.

[18] Zhan M, Wu J, Zhou L, et al. A memory access decreased decoding scheme for double binary convolutional Turbo code[J]. IEICE Transactions on Fundamentals, 2013, E96-A(8): 1812 - 1816.

[19] Lin C H, Chen C Y, A Y Wu. Area-efficient scalable MAP processor design for high-throughput multi-standard convolutional turbo decoding[J]. IEEE Transactions on VLSI Systems, 2011, 19(2): 305 - 318.

作者简介

詹 明 男,1975 生于河南新县,博士、副教授、硕士生导师,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.

文 红(通信作者) 女,1969 年生于成都,博士、教授、博士生导师,目前主要从事无线通信可靠和安全的交叉学科研究,开展 LDPC 码、物理层安全通信、喷泉编码、物理层无条件秘密通信领的研究.
E-mail: sunlike@ uestc. edu. cn

伍 军 男,1979 年生于湖南湘潭,上海交通大学网络空间安全学院副研究员,主要研究领域为:工物联网信息安全、控系统信息安全、下一代互联网安全、云计算技术及其安全、大数据技术及其安全等.