文章编号：2095-4980(2022)05-0431-07

# 优化结构的太赫兹 Turbo 编译码技术

李思凯，李　波

(西安邮电大学 通信与信息工程学院，陕西 西安 710061)

**摘　要：**主要阐述太赫兹(THz)通信系统中的信道编码部分，利用 CPU 多核进行并行计算，实现对 Turbo 码的编译码程序的加速。通过 4 个方面对 Turbo 码的编译码进行优化加速，包括预留内存空间、并行循环以及对编码结构和译码公式的优化，从而实现代码运行时间的缩短。经实验验证，经过对不同码长的数据进行编译码运算，发现在输入码长为 10 000 bit 时，并行计算时间可以缩短 56.6%。

**关键词：**太赫兹；Turbo 码；并行计算；多核加速

**中图分类号：**TN911.22　　**文献标志码：**A　　**doi：**10.11805/TKYDA2021174

# Terahertz Turbo encoding and decoding technology with optimized structure

LI Sikai，LI Bo

(School of Communication and Information Engineering，Xi'an University of Posts & Telecommunications，Xi'an Shaanxi 710061，China)

**Abstract：**In the channel coding of the terahertz communication system, the CPU multi-core is employed to perform parallel computing to achieve acceleration of the compiling code program for Turbo code. The optimized acceleration of Turbo code including four aspects, namely reserved memory space, parallel cycle, the optimization of coding structure and the decoding formula, thereby realizing shortening of code runtime. After experimental verification, the calculation of the compiling code is performed on different code length data. It is found that the parallel calculation can be shortened by 56.6% when the input code length is 10 000 bit.

**Keywords：**terahertz；Turbo code；parallel computing；multi-core acceleration

Turbo 码作为一种结合了交织器和卷积码并采用软输出的方式，其利用随机交织的编码器和迭代的译码器让性能相较于卷积码等其他译码方式有更好的提升，并更接近香农极限。近年来，Turbo 码广泛应用于蜂窝数据移动通信当中[1]，但是 Turbo 码有着较大的计算量和复杂度，在实际应用中会消耗较多的硬件资源，进而增加系统的传输时延并减小吞吐量。Turbo 码的编码原理与卷积码类似，而解码采用最大后验概率(Maximum A Posteriori，MAP)方法[2-3]。在此基础上，文献通过将 MAP 算法的前向和后向变量映射到对数域可以得到 LOG-MAP 算法[4-5]。与此同时，关于 Turbo 码的相关性能已经得到证实，但由于 Turbo 码的译码采用了迭代算法，所以计算量和运行时间依旧不容乐观。关于 Turbo 码的编译码性能，对不同调制阶数、不同码率和不同迭代方式下的误码率进行研究，并证明了 Turbo 码所具有的良好性能[6-7]。

由于太赫兹特殊的传输特性和传输频率为 0.1~10 THz(波长为 3 mm~30 μm)，其同时具有微波和红外的特性，也具有电子和光子的共同性质和优势，因此为电学和光学技术的发展提供了一种可以考虑的方向。相比微波通信有着更广的频率、带宽，更小的波束发散角；相比红外光，有着更好的穿透性，在空间通信中有着更好的应用。因其所蕴含的巨大的带宽和吞吐量，如果引入并行运算将具有更好的前景。同时，2017 年欧盟就正式部署第六代移动通信技术，而其重点正是频率为 0.275 THz 的太赫兹频段，目的正是要将峰值速率提高到 100 Gbps。目前已经可以做到实验室级别的太赫兹频段的信息传输[8-10]。尽管太赫兹有较好的发展前景，但是其较大的自由空间的损耗导致太赫兹通信系统传输的距离难以提升[11]。目前，已经有文章提出 Turbo 乘积码在 100 GHz 太赫兹通信系统上的应用[12]，其通过采用 Turbo 乘积码并改进了前向纠错方法，得到了更好的性能。但是并没有考虑到 Turbo 码在大量的运算中带来的时延问题。同时，也有文章提出在太赫兹通信系统中利用 Turbo 的迭代算法来提

收稿日期：2021-04-17；修回日期：2021-06-25

高性能，足以证明其性能上的优势以及良好的应用前景。

本文提出的并行 Turbo 编译码算法针对上述问题，提高了 Turbo 码编译码的效率，减小了传输时延，并用于太赫兹通信系统，以保证其传输的可靠性。与传统的无线通信系统的信道编码思路类似，由于 Turbo 码在低码率时表现优良，因此可以用于信令信道的传输[13]，Turbo 码在低信噪比的条件下依旧可以表现出接近香农极限的良好性能。就目前的各类信道编码而言，虽然 Turbo 码相较于低密度奇偶校验(low-Density Parity-Check，LDPC)码有着计算量大、复杂度高的缺点，但是其具有良好的抗干扰能力，可以为太赫兹传输中巨大的上下行信息量提供高可靠性，但是 Turbo 码的高复杂度使其在近年的移动通信领域逐渐被 LDPC 码取代。近年来设备的吞吐量尤其是多核运算的能力逐步提升，使得并行优化以及多核计算越来越受人关注，这正好可以弥补 Turbo 码运算量过大的缺陷，并可以运用到太赫兹通信系统的信道编码中。

## 1 Turbo 编码原理

Turbo 码也叫并行级联卷积码(Parallel Concatenated Convolutional Code，PCCC)，其理论性能非常接近香农极限。它的本质是在卷积码的基础上加入随机交织器，这使得编码中加入了随机性，防止出现错误扩散。该结构有两个递归系统卷积码(Recursive Systematic Convolutional code，RSC)分量编码器，其中一个分量编码器通过交织器后进行编码，编码后通过删余(puncture)矩阵来实现不同的码率，最后通过复接器就可生成 Turbo 码[12]。具体结构如图 1 所示。

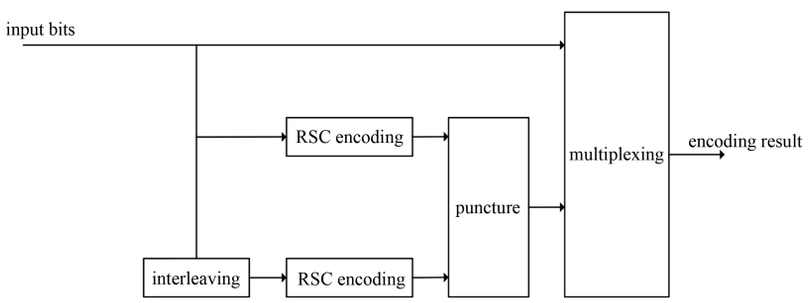


Fig.1 Encoder structure of Turbo code

图 1 Turbo 码编码结构

以(2,1,2)RSC 递归系统卷积码为例，即(7,5)码，此时 RSC1 和 RSC2 的生成矩阵为 $\boldsymbol{G}(D)=\left[1,\frac{1+D^2}{1+D+D^2}\right]$，其中递归体现在输出比特对于输入比特的反馈上。在编码过程中，对于每一个输入比特，输出一个信息比特和一个校验比特，上下两个编码器的唯一区别在于下方的输入比特经过了交织过程，因此两个编码器的输出比特也仅仅改变了序列的顺序，因而递归系统卷积码的实际速率为 1[13]。如上图所示，由于 Turbo 码采用了双 RSC 的编码方案，这使得并行计算成为可能。

### 1.1 交织器

交织器是 Turbo 码的核心，是实现随机编码的关键，交织器的本质其实是一种函数映射，通过将输入的信息比特进行重新排序，可以减小输出校验序列的相关性，当交织器的长度越长，得到的随机性越好，越接近随机序列，因此本编码器的交织器也被称作伪随机交织器。

### 1.2 删余矩阵

删余的过程就是对两个编码器产生的校验矩阵进行轮流选择，让输出的码率从 1/3 变为 1/2。让第 1 个编码器的校验比特乘以 $\boldsymbol{P}_1=[1\ 0]^{\mathrm{T}}$，让第 2 个编码器的校验比特乘以 $\boldsymbol{P}_2=[0\ 1]^{\mathrm{T}}$，即删余矩阵为 $\boldsymbol{P}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$。此时的校验位只有一个。删余过程是可选的，在下面的实验中采用不删余的方案。

## 2 Turbo 译码原理

Turbo 码的译码本质上是一种迭代算法，传统的译码方式采用软输入软输出(Soft Input Soft Output，SISO)的方案，其中 $y^x$ 为接收到比特流中的系统比特，即信息位，$y^{1p}$ 和 $y^{2p}$ 分别是两位校验比特，即校验位，$L^i$ 是内信息，$L_{12}^e$ 和 $L_{21}^e$ 是外信息，用于在 2 个分量译码器之间对信息进行相互传递。

与编码类似，Turbo 码的译码过程通过 2 个分量译码器和交织器组成，其中译码器的交织器结构和编码器一致，但是作用相反。第 1 个分量译码器用于对第 1 个编码器 RSC1 产生的比特进行译码，所产生的外信息经过交织后的 $L_{12}^{e'}$(也被称作先验信息)又作为第 2 个分量译码器的接收信息进行运算，计算出的外信息经过解交织送给第 1 个译码器作为反馈对接收到的比特进行译码，这就是 SISO 形式的 Turbo 译码原理[14]。其具体的译码器结构如图 2 所示。

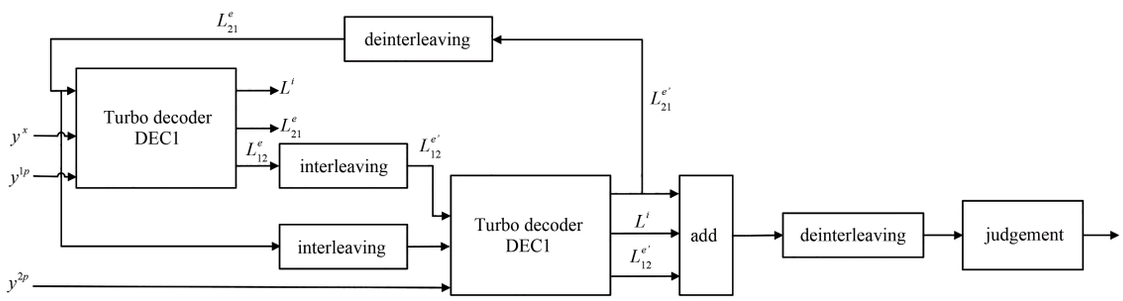


Fig.2 Decoding structure of Turbo code

图 2 Turbo 码译码结构

MAP 算法又称 BCJR 算法，通过计算后验概率来得到结果，由于 MAP 算法有大量的乘法运算，极大地占用资源和计算时间，但又存在大量的重复运算，因此具有优化的必要性和可能性。其本质是一种最优算法，目的是计算后验概率对数似然比：

$$L(\hat{u}_k)=\ln\left(\frac{P(d_k=1|y)}{P(d_k=0|y)}\right) \tag{1}$$

判决的规则为当 $L(d_k)\geqslant 0$ 时，$d_k=1$；当 $L(d_k)\leqslant 0$ 时，$d_k=0$。式中：$d_k$ 是信息比特；$y$ 为接收比特；$k$ 为分组长度。为了计算两个后验概率：$P(d_k=1\,|\,y)$ 和 $P(d_k=0\,|\,y)$ 的值，可以表示为：

$$L(\hat{u}_k)=\ln\left(\frac{\sum\limits_{\substack{(s',s)\\(\hat{u}_k=1)}}P(s_{k-1}=s',s_k=s,y)}{\sum\limits_{\substack{(s',s)\\(\hat{u}_k=0)}}P(s_{k-1}=s',s_k=s,y)}\right) \tag{2}$$

式中分子分母可以化为：

$$P(\hat{u}_k=i\,|\,y)=\sum_{\substack{(s',s)\\(\hat{u}_k=i)}}P(s_{k-1}=s',s_k=s\,|\,y)=\sum_{\substack{(s',s)\\(\hat{u}_k=i)}}P(s_{k-1}=s',s_k=s,y)/P(y) \tag{3}$$

式中分子表示的是从状态 $s_{k-1}$ 转移到 $s_k$，输入比特 $d_k=1$ 的支路和接受序列的联合概率密度之和。在离散无记忆通道的前提下，可以利用马尔可夫性质得到：

$$P(s_{k-1}=s',s_k=s,y)=P(s_{k-1}=s',y_1^{k-1})\cdot P(s_k=s,y_k\,|\,s_{k-1}=s')\cdot P(y_{k+1}^{N}\,|\,s_k=s) \tag{4}$$

式中：$\boldsymbol{\alpha}_k(s_k)$ 是前向状态矩阵；$\boldsymbol{\beta}_k(s_k)$ 是后向状态矩阵；$\gamma_k(s_{k-1},s_k)$ 是状态转移概率。表示为：

$$\begin{cases}\boldsymbol{\alpha}_{k-1}(s')=P(s_{k-1}=s',y_1^{k-1})\\ \gamma_k(s',s)=P(s_k=s,y_k\,|\,s_{k-1}=s')\\ \boldsymbol{\beta}_k(s)=P(y_{k+1}^{N}\,|\,s_k=s)\end{cases} \tag{5}$$

进一步计算可得：

$$\begin{cases}\boldsymbol{\alpha}_k(s)=\sum\limits_{s'}\alpha_{k-1}(s')\cdot\gamma_k(s',s)\\ \boldsymbol{\beta}_k(s)=\sum\limits_{s'}\beta_{k+1}(s')\cdot\gamma_{k+1}(s,s')\\ \gamma_k(s',s)=P(y_k\,|\,c_k)\cdot P(u_k)\end{cases} \tag{6}$$

将式(6)的结果代入式(2)得到联合概率表示为：

$$L(\hat{u}_k)=\ln\left(\frac{\sum\limits_{\substack{(s',s)\\(d_k=1)}}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum\limits_{\substack{(s',s)\\(d_k=0)}}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right) \tag{7}$$

式(3)可以进一步分解得到：

$$L(\hat{u}_k)=L(u_k)+L(y_k^s)+\ln\left(\frac{\sum\limits_{\substack{(s',s)\\(u_k=1)}}\alpha_{k-1}(s')\exp\left(\frac{1}{2}c_k^pL(y_k^p)\right)\beta_k(s)}{\sum\limits_{\substack{(s',s)\\(d_k=0)}}\alpha_{k-1}(s')\exp\left(\frac{1}{2}c_k^pL(y_k^p)\right)\beta_k(s)}\right) \tag{8}$$

第 1 项为来自上一个译码器的先验信息，第 2 项为软输出，第 3 项为外信息。其中第三项将作为反馈提供给下次译码，作为下次译码的先验信息。

## 3 Turbo 码的并行化修改

本并行算法基于 BCJR 算法进行修改优化，主要包括 3 个部分：一是对结构的改进，使用多线程技术对其加速；第二是针对编码部分进行结构并行优化；第三部分是针对译码部分进行并行优化。

### 3.1 整体结构优化

首先对代码的结构优化主要是在两个方面，一个是使用并行循环进行多核计算，另一个是事先预留内存。本次使用的软件为 MATLAB 2020a，硬件为 i9-10980HK。使用并行循环，是指使用 parfor-loop 的形式，使用该循环可以将一个 client 中的多个任务并行分配给各个 worker 中，每个 worker 独立运算返回结果，其本质是通过在并行池中利用多个线程来加快工作的流程。parfor 并行循环的使用步骤可以概括为图 3 所示的流程：

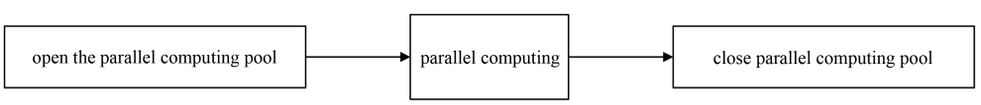


Fig.3 Optimization flow of parallel computing
图 3 并行计算的优化流程

并行计算优化的条件较为特殊，它要求的循环迭代的过程中各个 worker 之间数据不存在相互耦合、相互调用的情况，即要求第 $n$ 次并行循环和第 $m$ 次并行循环($n\neq m$)之间的数据相互独立。其次，parfor 更适用于在代码中简单大量的计算，过于复杂的计算或者计算量过小的计算可能会使计算时间更长。

### 3.2 编码结构优化

其中对 Turbo 编码结构的并行优化主要体现在两个 RSC 编码器处，由于两个 RSC 编码器的输入输出之间并不存在耦合的关系，所以可使用并行加速，让两个编码器同时工作，并将输出结果同时输出。在具体的硬件实现时，需要将第二个 RSC 编码器的交织时间计算，并对第一个 RSC 译码进行相应的延时，将串行编码改为并行编码，将运行时间大大缩短。

### 3.3 译码结构优化

在译码阶段，由于采用 BCJR 算法，前向的计算结果 $\alpha$ 需要作为后向迭代 $\beta$ 的输入，所以在此处不宜使用并行优化，但是在计算 $\gamma$ 分支度量时，定义式如下：

$$\gamma(\sigma_{i-1},\sigma_i)=\frac{P(u_i)}{(\pi N_0)^{n/2}}\exp\left(-\frac{\left\|y_i-c_i\right\|^2}{N_0}\right) \tag{9}$$

式中 $\sigma_{i-1}$ 和 $\sigma_i$ 分别为上一状态和当前状态，上式中可以并行加速的关键就在于 $P(u_i)$，因为针对每次输入的比特需要和多个状态进行计算求出概率，其中分别包括上一状态(0 或 1)，当前状态(0 或 1)，以及交织之后的上一状态(0 或 1)，当前状态(0 或 1)，共计 8 个状态的概率，所以通过使用并行算法同时对这 8 个状态进行计算，同时将这

8 个状态进行并行输出，因其并不存在相互耦合，所以在软件中采用并行加速更加高效，在硬件中同样可以采用流水线形式实现。改进后的 $\gamma$ 分支度量计算公式可以写为：

$$\gamma(\sigma_{i-1},\sigma_i)=\frac{1}{4\pi}\exp(R_0-\sqrt{SNR}*H_{\text{Input}}^2+R_1-\sqrt{SNR}*H_{\text{Parity}}^2) \tag{10}$$

式中：$R_0$ 是没有经过 RSC 编码器的分支；而 $R_1$ 是经过第一个 RSC 编码器的分支；$SNR$ 是信噪比，这里采用的是信号和噪声功率比值；$H_{\text{Input}}$ 表示对于输入 $R_0$ 的当前状态和上一状态的关系；$H_{\text{Parity}}$ 表示的是输入的 $R_1$ 的当前状态和上一状态的关系，包括 0 或 1 两种情况。由于 $H_{\text{Input}}$、$H_{\text{Parity}}$ 这两个变量之间不存在耦合和相互调用，两变量内部也不存在耦合，所以可以采用并行计算的方式。

与此同时，在每次前向和后向迭代的内部也可以使用并行加速，其定义式分别如下：

$$\alpha_i(\sigma_i)=\sum_{\alpha_{i-1}\in\sum}\gamma_i(\sigma_{i-1},\sigma_i)\alpha_{i-1}(\sigma_{i-1}) \tag{11}$$

$$\beta_{i-1}(\sigma_{i-1})=\sum_{\sigma_i\in\sum}\beta_i(\sigma_i)\gamma(\sigma_{i-1},\sigma_i) \tag{12}$$

可以看出前向分支和后向分支同样与上一状态和当前状态有关，一共 4 种情况，所以在内部计算时同样利用并行加速对其进行改进，改进后的前向和后向迭代分支度量为：

$$\alpha_{x,i}=\gamma_x\alpha_{x,i-1}+\gamma_y\alpha_{y,i-1} \tag{13}$$

$$\beta_{x,i}=\gamma_x\beta_{x,i+1}+\gamma_x\beta_{y,i+1} \tag{14}$$

但是前向和后向迭代的外部由于相互耦合并不适合并行计算，对于软件而言不易实现，对于硬件而言则会影响时序。

## 4　仿真结果分析

通过对 Turbo 编译码的修改，得出修改前的 10 次运行时间与修改后的 10 次运行时间如图 4 所示，其中每次运行要求 Turbo 译码过程迭代 5 次：

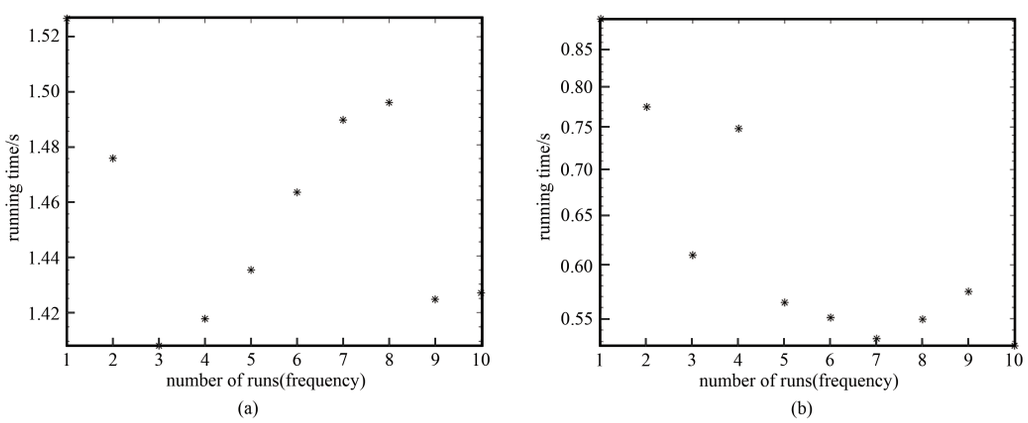


Fig.4 Comparison of the running time before and after optimization using parallel computing
图 4 并行优化前后各自运行 10 次的时间

误码率随着信噪比增加的变化如图 5 所示，可以看出，二者在性能上的差距并不大，并行运算对于编译码结果并不存在损失或者影响，却提高了运行效率。

优化前后运行代码的平均时间如表 1 所示。可以明显看出，修改前的平均运行时间达到了 1.456 2 s，而经过并行优化后，运行时间可以缩短到 0.632 4 s，缩短了约 56.6%。而且并行加速并不会损失精确度，而是提升运行效率，缩短传输时延。

需要注意的是，以上的验证都基于输入码字为 10 000 bit 时的情况，如果改变输入码字为 1 000 bit 时，优化前后的平均运行时间如表 2 所示，在输入码字减小的情况下，优化后的代码平均运行时间反而变长，由此可以进一步佐证，并行优化只能针对长码进行优化，短码结构的运行效率反而会下降。

为了探究不同输入码字长度对优化性能的影响，以及在各个区间并行运算与串行计算的差距，计算了 1 000~

10 000 bit的不同输入码字在并行优化前和并行优化后的时间对比，效果如图6所示。求出并行优化前后的运行时间差，结果如图7所示。

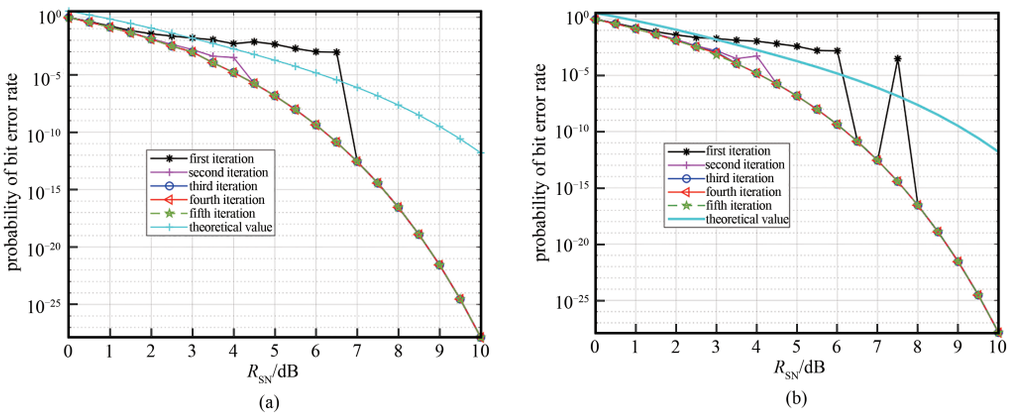


Fig.5 Comparison of bit error rate before and after parallel optimization
图5 并行优化前后误码率的对比

表1 并行优化前后平均运行时间对比

Table1 Comparison of average running time before and after parallel optimization

| average running time of original code/s | average running time after parallel optimization/s |
|---|---|
| 1.456 2 | 0.632 4 |

表2 输入码字为10 000 bit时优化前后平均运行时间对比

Table2 Comparison of average running time before and after optimization in 10 000 bit

| average running time of original code/s | average running time after parallel optimization/s |
|---|---|
| 0.114 1 | 0.213 6 |

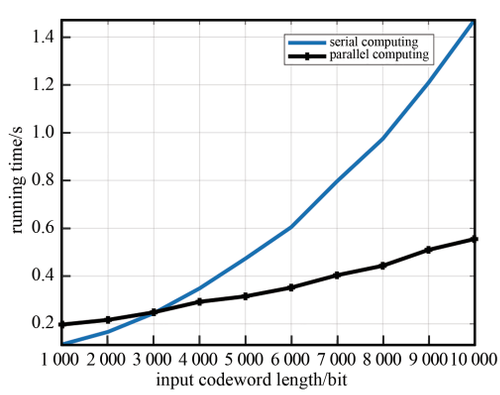


Fig.6 Comparison of running time before and after parallel optimization with different code lengths
图6 不同码长使用并行优化前后时间对比

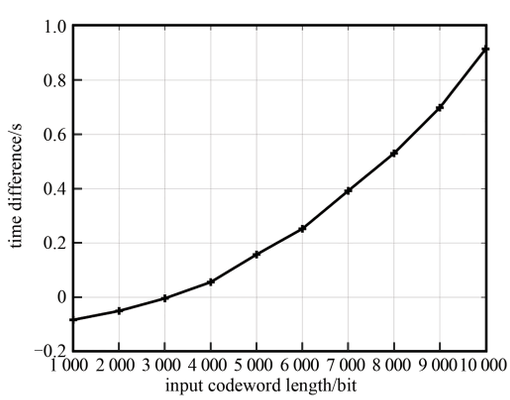


Fig.7 Variation of running time difference before and after parallel optimization with code length
图7 并行优化前后时间差随码长的变化

由此可以得出，随着输入码长的不断增大，运行时间不断增加，但是并行优化后的运行时间增加幅度相比于原始的串行计算而言更低，并行优化在码长大于3 000 bit时，可以得到优于优化前的效果，如果码长小于3 000 bit，此时并行优化后反而会延长运算时间。除此之外，随着输入码字长度的增大，并行优化带来的时间上的缩短呈指数增长。

## 5 结论

本文通过优化Turbo码的基本编译码结构，利用多核进行并行计算，从而对运行时间进行优化。对于并行计算，进行优化后的程序在运行时间上极大缩短，在运行效率上极大提升，尤其是在长码字的情况下提升效果更好。这对于具有极大传输速率和运算量的太赫兹传输环境而言有良好的发展前景。目前，在常见的通信协议中，由于Turbo码在长码以及高码率的情况下复杂度过高，效率太低，因而其多用于控制信道。但是经过并行优化后，Turbo码可以极大提高编译码的运行效率，进而缩短传输时延。

**参考文献：**

[1] YU L,WANG X,LIU J. An improved rate matching algorithm for 3GPP LTE turbo code[C]// 2011 Third International Conference on Communications and Mobile Computing. Qingdao,China:[s.n.], 2011:345-348.

[2] BERROU C,GLAVIEUX A,THITIMAJSHIMA P. Near Shannon limit error-correction coding and decoding: Turbo-codes[C]// IEEE International Conference on Communications. Geneva,Switzerland:IEEE, 1993:1064-1070.

[3] HAGENAUER J,OFFER E,PAPKE L. Iterative decoding of binaryblock and convolutional codes[J]. IEEE Transactions on Information Theory, 1996(42):429-445.

[4] ROBERTSON P,VILLEBRUN E,HOEHER P. A comparison of optimal and sub-optimal decoding algorithms operating in the log domain[C]// IEEE International Conference on Communications. Seattle,WA,USA:IEEE, 1995:1009-1013.

[5] LU X,LYU M,HONG T,et al. Improvement of SINR for MIMO channels in terahertz communication[C]// 2020 International Wireless Communications and Mobile Computing. Limassol,Cyprus:IEEE, 2020:489-493.

[6] SUN Kaiying,YUAN Dongfeng ,ZHOU Xiaotian. Performance analysis of Turbo codes with different interleavers and decoding methods[C]// 2010 IEEE International Conference on Information Theory and Information Security. Beijing,China:IEEE, 2010: 121-124.

[7] HADDAD S,BAGHDADI A,JEZEQUEL M. On the convergence speed of turbo demodulation with turbo decoding[J]. IEEE Transactions on Signal Processing, 2012,60(8):4452-4458.

[8] KOENIG S,LOPEZ-DIAZ D,ANTES J,et al. Wireless sub-THz communication system with high data rate[J]. Nature Photonics, 2013,7(12):977-981.

[9] BOES F,MESSINGER T,ANTES J,et al. Ultra-broadband MMIC-based wireless link at 240 GHz enabled by 64GS/s DAC[C]// 2014 39th International Conference on Infrared,Millimeter,and Terahertz waves(IRMMW-THz). Tucson,AZ,USA:IEEE, 2014: 1-2.

[10] MESSINGER T,KRISHNEGOWDA K,BOES F,et al. Multi-level 20 Gbit/s PSSS transmission using a linearity-limited 240 GHz wireless frontend[C]// IEEE International Conference on Microwaves. Tel Aviv,Israel:IEEE, 2015:1-3.

[11] 赵卓. 太赫兹通信中的高速数字信号处理及其并行算法[J]. 电子技术与软件工程, 2019(11):3. (ZHAO Zhuo. High speed digital signal processing and its parallel algorithm in terahertz communication[J]. Electronic Technology and Software Engineering, 2019(11):3.)

[12] LOPACINSKI L,NOLTE J,BUECHNER S,et al. Improved turbo product coding dedicated for 100 Gbps wireless terahertz communication[C]// 2016 IEEE 27th Annual International Symposium on Personal,Indoor,and Mobile Radio Communications. Valencia,Spain:IEEE, 2016:1-6.

[13] MAO T,WANG Z. Terahertz wireless communications with flexible index modulation aided pilot design[J]. IEEE Journal on Selected Areas in Communications, 2021,39(6):1651-1662.

[14] 李燕斌,杨杨. LTE 链路中 Turbo 编译码原理及 FPGA 实现[J]. 太赫兹科学与电子信息学报, 2015,13(5):48-54. (LI Yanbin, YANG Yang. Principle and FPGA implementation in the LTE link[J]. Journal of Terahertz Science and Electronic Information Technology, 2015,13(5):48-54.)

**作者简介：**

**李思凯**(1997-)，男，广东省深圳市人，在读硕士研究生，主要研究方向为太赫兹通信与信道编码.email: lisikai258@stu.xupt.edu.cn.

**李　波**(1980-)，男，山东省博兴市人，博士，副教授，主要研究方向为无线通信系统设计、卫星载荷系统设计、视频无线传输、FPGA 和 DSP 的软硬件设计.