DOI:10.19651/j.cnki.emt.1802352

ARM 架构下 Turbo 译码时延性能研究

潘 超 陈阳杰 谭国平 李岳衡
(河海大学 计算机与信息学院通信与信息系统研究所 南京 211100)

摘 要：为了适应基于 5G 的物联网技术的快速发展,以 ARM 架构作为物联网终端设备的构想具有重要的实际应用意义。但 ARM 架构能否满足 5G 物联网终端性能要求仍然不明确,特别是终端对于海量数据的处理时延将直接影响其实际可行性。鉴于此,基于开源的空中接口平台(OAI),本文重点研究了采用 ARM 架构的 Turbo 译码时延问题,并与 x86 架构计算机进行对比。测试结果表明,基于 ARM 架构的译码延时要比基于 x86 架构的译码延时长 20%～25%,鉴于两种架构性能上的差异,这一延时差距符合预期。

关键词：Turbo 码;空中接口平台;译码时延;交织器;并行译码 Log-MAP
中图分类号：TN911.22 **文献标识码**：A **国家标准学科分类代码**：510

Research on Turbo decoding delay performance under ARM architecture

Pan Chao Chen Yangjie Tan Guoping Li Yueheng
(Institute of Communication and Information Systems, Hohai University, Nanjing 211100, China)

Abstract: In order to adapt to the rapid development of IoT technologies based 5G, the idea of using the ARM architecture as a terminal device for IoT has important practical significance. However, it is still unclear whether the ARM architecture can meet the 5G IoT terminal performance requirements. In particular, the processing delay of terminal for massive data will directly affect its feasibility.In view of this, based on the open air interface(OAI), this paper focuses on the Turbo decoding delay problem using the ARM architecture and compares it with the x86 architecture computer.The results show that the decoding delay based on the ARM architecture is 20%～25% longer than that of x86 architecture. Given the difference in the performance of the two architectures, this gap is in line with expectations.

Keywords: Turbo code; open airinterface; decoding delay; interleaver; parallel decoding Log-MAP

0 引 言

1993 年,Berrou 等[1]提出了 Turbo 码的并行级联卷积码,由于充分利用了有噪信道编码定理的 3 大条件,其译码性能远远超过了之前提出的所有差错控制码方案[2]。因此,Turbo 码被迅速投入到通信领域的各类应用中,在无线通信系统中,它被应用于信道的估计与译码、多用户检测、Turbo 均衡、正交频分复用(orthogonal frequency division multiplexing,OFDM)以及空时处理等[3];在信息安全领域中,它被应用于数字水印和信息加密。然而,Turbo 码也存在所谓的“错误平层”缺陷,降低“错误平层”是设计人员需要重点考虑的问题[4-5]。

根据香农有噪信道编码定理,满足随机性编译码、编码长度 $L \to \infty$、采用最大似然译码方案这 3 个条件的信道编码方法可以使得信息序列的误码率任意小。对 Turbo 码来说,在其编码码字生成的过程中,交织器有着重要作用,并且对 Turbo 码的性能有着很大的影响。交织器实现了信息序列比特位置的随机置换,使得编码输出比特之间的相关性减小,最后由不同分量编码器生成的短码构造的长码具有了伪随机性[1-2,6]。Turbo 码的译码器具有反馈结构,译码器输出的外信息经过交织/解交织操作后被当做先验信息传递给另一个译码器作为参考信息,从而提高译码准确度,多次迭代以后,每个码元基本上获得了输入信息序列中所有码元的相关信息,也就意味着实现了译码过程的伪随机化[7]。

进阶精简指令集机器(advanced RISC machine,ARM)表示一类 RISC(精简指令集计算机)微处理器,这类处理器采用固定长度的指令格式,使用单周期指令,有很高的指令执行效率。ARM 架构是一个 32 位的 RISC 中央处理器架构[8-9],目前已经被广泛使用于消费电子、无线通信、手持设

收稿日期:2018-11-22

备和工业控制系统等各个领域。当前,在各种 32 位嵌入式处理器中随处可见 ARM 架构的产品,它也被广泛应用在各类电子产品中,从可携式装置(移动电话、多媒体播放器、掌上型电玩)到计算机外围设备(硬盘、桌上型路由器),甚至在导弹的弹载计算机等军用装备中都有它的存在。

1　ARM 架构上实现 Turbo 编译码

OAI 开源代码中的 Turbo 码编解码模块是为 x86 架构所编写的,不能够在 ARM 架构下运行,因此,为了在 ARM 架构下进行译码时延性能的测试,需要重新为其编写可执行的编解码模块。

造成这一障碍的主要原因是使用的指令集不同,在 x86 架构下,Turbo 码编解码模块中使用了大量的 SSE3 和 SSE4 指令集,通过在编程中加入这些指令集操作,可以使程序执行过程中减少了对 CPU 资源的占用,提升了精度和响应速度。与之相对应的,在 ARM 中,NEON 指令集也实现了类似的功能,需要完成两种指令集间的相互转换[10]。

ARM 架构下 Turbo 码编码模块的程序流程如图 1 所示。

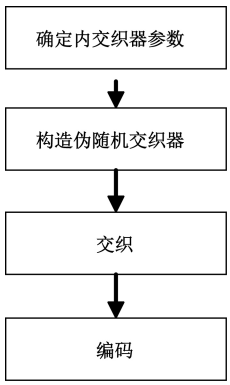

图 1　编码模块流程

其中内交织器参数 f1 和 f2 在 LTE 协议 36.212 中规定,根据输入的交织器长度不同,有 188 种可能的组合[11];然后根据交织器参数构造伪随机交织器,完成交织操作,打乱信息序列的比特顺序;在进行编码操作时,每输入一个字节的信息序列,8 位环绕加法循环执行 3 次,产生 24 个新比特,因此编码的码率为 1/3。当全部输入信息序列完成编码操作后,输出编码结果。

ARM 架构下 Turbo 码译码模块的程序流程如图 2 所示。

译码模块使用 8 路并行 Log-MAP 译码结构[12],在进行译码之前先将接收到的编码信息分割为系统信息、分量编码器 1 经过调制解调和信道传输后的信息,以及分量编码器 2 经过调制解调和信道传输后的信息。然后代入 Log-MAP 函数计算信息比特的对数似然比输出,并在先验信息和系统信息的协助下,进行多次迭代译码,提高译码准确性。

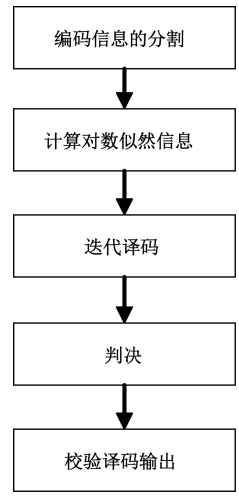

图 2　译码模块流程

2　Turbo 码性能仿真分析

2.1　系统仿真模型

MATLAB 仿真系统如图 3 所示。

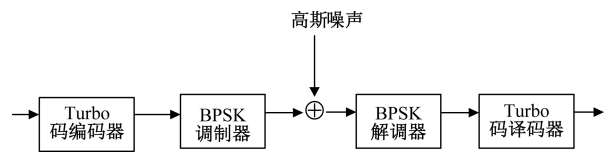

图 3　Turbo 编译码系统仿真模型

参考 LTE 协议中 Turbo 编译码器结构[3],输入到编码器的信息序列为二进制随机序列,使用的交织器为伪随机交织器,两个编码器的结构相同,分量码生成矩阵为(7,5),码率为 1/3,使用 BPSK 调制将其转换为双极性信号,使用的信道模型为普通的高斯信道,使用 Log-MAP 译码算法,最后通过统计译码输出序列中的误比特数来定量分析不同参数取值条件下 Turbo 码的性能。

2.2　仿真结果

1) 迭代译码次数对 Turbo 码性能的影响

在一定范围内,增加迭代译码次数能有效提高 Turbo 码译码性能。图 4～6 所示分别给出当交织器长度为 104、3 008、6 144 时在不同译码迭代次数下的误比特率仿真结果。

从图 4～6 所示曲线可以看出,无论信噪比条件如何,增加迭代次数总能获得一定的译码性能提升,而且随着交织长度的增加,这种性能提升愈加明显。当迭代译码次数较小时,增加迭代次数能够显著地降低 BER,尤其是当迭代次数小于 4 时。随着迭代次数继续增加,BER 曲线开始放缓,特别地,当迭代次数大于 6 以后,较高信噪比下的 BER 曲线已经开始出现平台,这表明通过在不同译码器之间传递先验信息作为译码参考来提升性能的方法已经达到

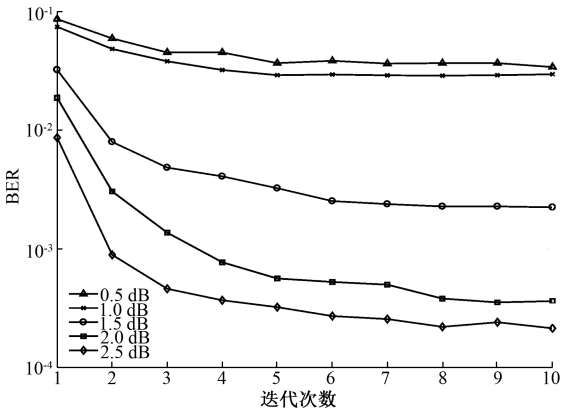

图 4　交织长度为 104 时迭代次数对 Turbo 码性能的影响

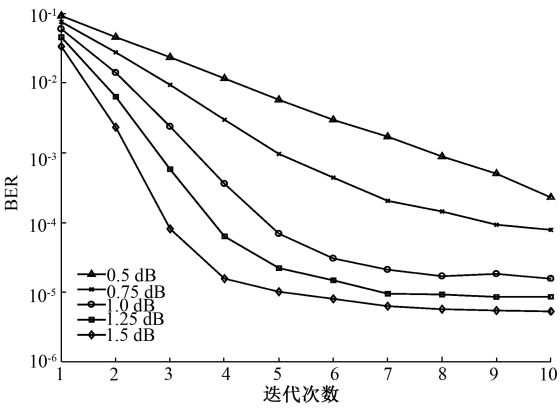

图 5　交织长度为 3 008 时迭代次数对 Turbo 码性能的影响

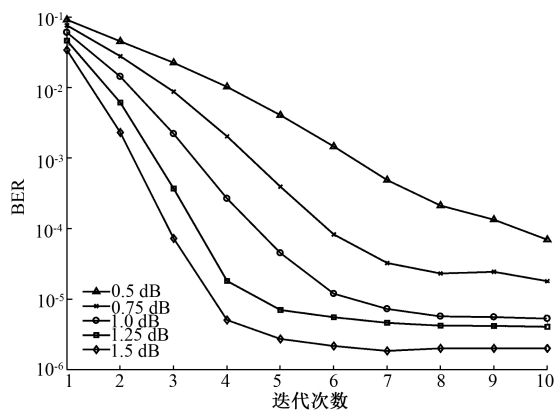

图 6　交织长度为 6 144 时迭代次数对 Turbo 码性能的影响

了极限。当交织长度较小时,误比特率曲线的收敛速度要更快,一般 2～3 次迭代后,误比特率曲线就基本不再下降,而在交织长度为 6 144 bit 时,直到迭代 5 次以后,BER 曲线才开始出现平台。在做 ARM 架构译码时延测试的时候,同样是在高斯信道条件下,考虑的系统性能要求是:在中低信噪比条件下(1.5 dB 左右),误码率在 10^{-5}左右。迭代次数太少,误码率太大达不到要求;迭代次数太多,译码花费时间太多,不满足测试所要求的低时延,综合考虑以上因素和仿真结果设定迭代译码次数为 5。

2) 织器长度对 Turbo 码性能的影响

在 Lte 36.212 协议中,规定了帧长度,最大为 6 144 bit,最小为 40 bit。仿真参数不变,迭代次数使用上述仿真的结论设为 5,交织长度分别取 104、800、1 600、3 008 和 6 144。仿真曲线如图 7 所示。

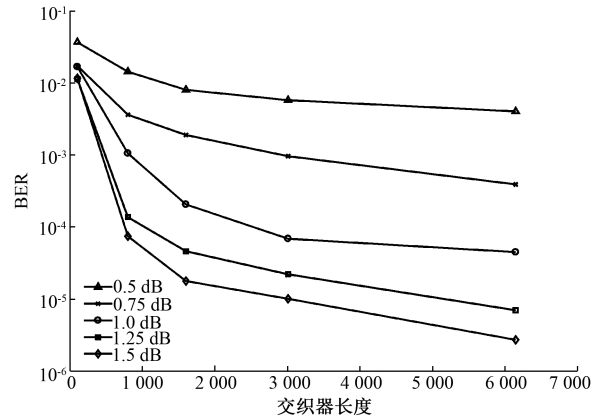

图 7　不同信噪比 Turbo 码的性能随交织长度的变化曲线

从图 7 所示的仿真结果来看,总体趋势上,随着交织长度的增加 Turbo 码的性能是不断提升的。在信噪比较小的情况下,增加交织器长度对性能的提升并不明显;随着信噪比逐渐增加,交织器长度对译码性能的影响开始增大,特别地,当交织器长度小于 2 000 bit 时,BER 随着交织长度的增加迅速下降,之后继续增加交织器长度,BER 曲线则开始趋于平缓,这验证了错误平台的存在。交织器长度设置过小,交织器结构简单,但误码率偏大;交织器长度设置过长,则会造成译码复杂性增加,增加解码时延,因此选择在 1.5 dB 左右误比特率达到 10^{-5}的 3 008 bit 长度的交织器作为参考标准来测试 ARM 架构译码延时性能。

3　Turbo 码译码时延对比

通过之前的 MATLAB 仿真,得到了与 Turbo 码性能相关的两个重要参数的值:交织器长度(3 008 bit)以及迭代译码次数(5 次)。因为需要测试并比较译码时间,所以规定了输入编码器的信息序列的总长度为 10 帧,即 30 080 bit 的随机二进制序列,并且在测试程序中加入了 BPSK 调制模块和高斯噪声模块,采用 Log-MAP 译码算法,进行多次迭代后输出结果。在 ARM 和 PC 上分别测试 20 组随机序列,计算译码时间和误比特率的平均值。

1) 不同输入序列长度下译码时延的比较

分别输入序列为 10、100、1 000 帧的序列进行测试,从测试结果上来看,译码时延基本随着输入信息序列的长度的增加呈现线性增长的趋势。输入序列为 10 帧时,x86 架构下的译码时延 21 ms,而 ARM 架构下为 25.4 ms,增幅

为 21%；输入序列为 100 帧时，x86 架构下的译码时延 206.7 ms，而 ARM 下为 254.2 ms，增幅为 23%；输入序列为 1 000 帧时，x86 架构下的译码时延 2 072.5 ms，而 ARM 下为 2 599.9 ms，增幅为 25%。可以认为，随着输入序列长度的不断增加，译码器的计算量不断加大，对系统的性能要求也不断提高，因此，ARM 架构与 x86 架构之间的性能差异就体现的越来越明显。

2）不同迭代次数下译码时延的比较

分别用迭代次数为 2、5、8 次进行测试，从测试结果上来看，译码时延随迭代次数的增大而增加，但是误比特率在迭代 5 次之后就基本趋于稳定不再减小，继续增加迭代次数不仅不会明显的提高系统性能，反而还会大幅度地增加译码时延，因此在交织器长度为 3 008 bit 的情况下，迭代 5 次为最优选择。从 x86 系统的时延结果来看，每增加3 次迭代次数，译码时间增加 30%左右；与 ARM 架构系统相比，在迭代次数为 2 时，ARM 架构比 x86 架构多花费 19%译码时间；在迭代次数为 5 时，ARM 架构比 x86 架构多花费 21%译码时间；在迭代次数为 8 时，ARM 架构比 x86 架构多花费 22%的译码时间。因此，可以认为，随着迭代次数的增加，译码时间增大，且 ARM 架构与 x86 架构之间的译码时间差距也在缓慢增加。

3）不同交织器长度下译码时延的比较

分别用 104、3 008、6 144 bit 的交织器进行测试，从测试结果上来看，随着交织器长度的增加，译码时延不断增大，误比特率不断减小，基本符合仿真结果所示，误比特率的减小幅度越来越小，最后趋于稳定。在交织器长度为 3 008 bit 时，误比特率已经接近 10^{-5}，继续增大到 6 144 bit 时，交织器长度增加超过一倍，但误比特率仍然位于 10^{-5} 左右，而译码时延却增加了 18%左右，因此，3 008 bit 的交织器长度为较优选择。与 x86 架构系统相比，在交织器长度为 104 bit 时，ARM 架构上的译码时延增幅为 19%；在交织器长度为 3 008 bit 时，ARM 架构上的译码时延增幅为 21%；在交织器长度为 6 144 bit 时，ARM 架构上的译码时延增幅为 24%。可以认为，随着交织器长度的增加，译码器结构更加复杂，译码时间不断增加，且 ARM 架构系统与 x86 架构系统之间的译码时延差距也在增加。

表 1～3 所示分别列出了 3 种不同条件下 ARM 架构和 x86 架构的译码时延对比。

表 1　不同交织长度下两种架构译码时延对比
（迭代 5 次，10 帧）

系统架构	交织长度		
	104 bit	3 008 bit	6 144 bit
x86 架构	12.3 ms	21.0 ms	30.0 ms
ARM 架构	14.7 ms	25.4 ms	37.2 ms
增幅/%	19.5	21.0	24.0

表 2　不同迭代次数下两种架构译码时延对比
（交织长度 3 008 bit，10 帧）

系统架构	迭代次数		
	2 次	5 次	8 次
x86 架构	15.5 ms	21.0 ms	26.8 ms
ARM 架构	18.5 ms	25.4 ms	32.7 ms
增幅/%	19.4	21.0	22.0

表 3　不同序列长度下两种架构译码时延对比
（交织长度 3 008 bit，迭代 5 次）

系统架构	序列长度		
	10 帧	100 帧	1 000 帧
x86 不同架构	21.0 ms	206.7 ms	2 072.5 ms
ARM 架构	25.4 ms	254.2 ms	2 599.9 ms
增幅/%	21.0	23.0	25.4

首先通过横向比较看出，不论是在 ARM 架构还是 x86 架构中，交织长度、迭代次数以及序列长度都会影响译码时延，且都随着这 3 个变量值的增加而增大。对于序列长度来说，译码时延与其成简单的线性关系，序列长度增长 10 倍，译码时间也基本增加 10 倍；对于交织长度和迭代次数来说，译码时延与其并没有明显的数学关系，交织长度和迭代次数增加，译码时间会增加，但 BER 会减小，且减小幅度越来越小，最后趋于平稳；交织长度和迭代次数减少，译码时间会随之减少，但 BER 会快速增加，因此，选择合适的交织长度和迭代次数是十分重要的。其次通过纵向比较得出，ARM 架构的译码时间要普遍长于 x86 架构。交织长度、迭代次数以及序列长度的增加，都会带来译码器结构复杂化或使译码器计算量增加，这些改变会使得 ARM 架构与 x86 架构之间的性能差异越来越明显，体现在数据上就是译码时间的增幅变大。因此，ARM 在 Turbo 译码时延方面的性能与 x86 相比还是有差距的，在所设置的实验条件下，这一差距约为 20%～25%，鉴于两种架构性能上的差异，这一差距符合预期，ARM 架构的单板计算机基本符合要求，但同时还有许多值得优化加速的地方。

4　结　　论

本次实验也存在诸多局限性并且还有许多值得深入测试和改进的地方。使用的 ARM 开发板并不带有 DSP，所得到的译码时延只适用于对比分析，与实际的译码时延不具有可比性；用来测试译码时延的测试程序中，输入的总序列长度较短，传输信道仅仅使用了普通的高斯信道，因此译码模块的计算量并不是很大，ARM 架构单板计算机性能足以满足这一计算要求。而现实环境中存在着各种噪声干扰且实际生活中传输的数据量十分庞大，因此对译码器的处理能力有着很高的要求。虽然在本次对比实验中，两个平台的时延相差并不是很大，但是在复杂的环境下，这一微

弱差距可能被放大至无法接受的程度,因此需要在更加复杂且接近实际应用的场景下进行测试与优化。

参考文献

[1] BERROU C, GLAVIEUX A, THITIMAJSHIMA P. Near Shannon limit error-correcting coding and decoding: Turbo-codes (1) [C]. IEEE International Conference on Communications '93 (ICC '93), 1993: 1064-1070.

[2] BERROU C, GLAVIEUX A. Near optimum error correcting coding and decoding: Turbo-codes[J]. IEEE Transactions on Communications, 1996, 44 (10): 1261-1271.

[3] BURR A G, WHITE G P. Performance of turbo-coded OFDM[C]. IET, 2016:8/1-8/8.

[4] GOFF S L, GLAVIEUX A, BERROU C. Turbo-codes and high spectral efficiency modulation [C]. Communications(ICC '94), 1994:645-649.

[5] GAMAL H E, HAMMONS A R. Analyzing the turbo decoder using the Gaussian approximation[J]. IEEE Transactions on Information Theory, 2001, 47(2): 671-686.

[6] BOHORQUEZ R G, NOUR C A, DOUILLARD C. Improving Turbo codes for 5G with parity puncture-constrained interleavers[C]. International Symposium on Turbo Codes & Iterative Information Processing, IEEE, 2016:151-155.

[7] TONNELLIER T, LEROUX C, GAL B L, et al. Lowering the error floor of Turbo codes with CRC verification[J]. IEEE Wireless Communications Letters, 2016, 5(4):404-407.

[8] 施乐平, 杨征宇, 马宪民, 等. ARM 嵌入式系统综述[J]. 中国测试, 2012,38(S1):14-16.

[9] SEAL D. ARM Architecture Reference Manual[M]. Alpha architecture reference manual, Digital Press, 2000.

[10] 金瑶,陈磊萍. ARM 指令集与 x86 指令集之比较[J]. 成功:教育版, 2007(10):215-215.

[11] SESIA M, TOUFIK M, BAKER M. LTE, the UMTS long term evolution: from theory to practice [M]. Chichester, West Sussex, U. K.; Hoboken, N. J.: Wiley, 2011.

[12] YOSHIKAWA H. On the bit error probability for constant log-MAP decoding of convolutional codes[C]. International Symposium on Information Theory and ITS Applications, IEEE, 2017.

作者简介

潘超,硕士研究生,主要研究方向为 4G/5G 关键技术等。E-mail:949519353@qq.com

陈阳杰,硕士研究生,主要研究方向 4G/5G 关键技术等。

谭国平,博士、教授,主要研究方向为移动自组网、无线多媒体通信等。

李岳衡,博士、教授,主要研究方向为多天线传输理论与技术、通信信号处理及专用电路设计与实现等。