一种高吞吐率的系统Raptor码并行译码方法

任雁鹏[1,2]，管　武[1]，梁利平[1]

(1. 中国科学院微电子研究所　北京 朝阳区　100029；2. 中国科学院大学微电子学院　北京 怀柔区　100049)

【摘要】在系统Raptor码译码中，针对高复杂度的高斯消元运算导致译码延时大、吞吐率低的问题，提出一种低延时高吞吐率的降维并行译码方案。该方案采用仅对少量丢包译码的低复杂度降维运算，替换对全部源数据包译码的高斯消元运算，降低译码延时；并针对降维译码采用全并行的硬件结构实现，提高译码吞吐率。依此方案，在Xilinx FPGA XC7K410T平台上实现系统Raptor译码器。测试结果表明，当网络丢包率在10^{-2}以下时，译码数据吞吐率达到3.5 Gbps，是相同硬件下采用高斯消元译码实现的80倍以上。

关　键　词　数字喷泉码；　降维译码；　并行译码；　系统Raptor码

中图分类号　TN919.3　　**文献标志码**　A　　doi:10.3969/j.issn.1001-0548.2018.06.003

A High Throughput Parallel Decoding Method for Systematic Raptor Codes

REN Yan-peng[1,2], GUAN Wu[1], and LIANG Li-ping[1]

(1. Institute of Microelectronics, Chinese Academy of Sciences　Chaoyang Beijing　100029;

2. School of Microelectronics, University of Chinese Academy of Sciences　Huairou Beijing　100049)

Abstract　The high complexity Gaussian elimination (GE) algorithm of the systematic raptor decoding results in its high latency and low throughputs. A high efficiency parallel dimensionality-reduction decoding scheme is presented in this paper. The proposed scheme uses low complexity dimensionality-reduction algorithm to decode lost packets to replace the GE algorithm which decodes all source packets. Meanwhile, a full parallel structure for the decoding is proposed to implement the dimensionality-reduction-algorithm. At last, the decoder is implemented on Xilinx FPGA XC7K410T. The test results show that the scheme can achieve a 3.5 Gbps throughput within a 10^{-2} packet loss probability, which is 80 times better than that of the GE algorithm.

Key words　digital fountain code; dimensionality-reduction decoding; parallel decoding; systematic raptor codes

数字喷泉码技术以其高性能的纠错能力，得到广泛的青睐。其中的低复杂度码型——系统Raptor码[1-2]，更是广泛应用于无线传输领域[3-5]。系统Raptor码在编码时，对源数据包增加少量冗余包；在译码时，通过接收到的源数据包和冗余包纠正传输过程中的丢包，实现数据恢复。这种以包为单位的纠错方法，具有较高的吞吐率和较低的复杂度，在高速数据传输中优势明显。

随着超高清、3D视频以及海量数据的高速传输，未来无线数据传输吞吐率将达到1 Gbps以上。然而，当前3GPP标准(3rd generation partnership project)中采用的系统Raptor码技术，速率还在100 Mbps量级，已经无法满足高速传输的需求。需要更高速率的系统Raptor码编码译码技术。

目前，许多学者都在进行系统Raptor码的改进和优化。文献[6]发现系统Raptor码运算耗时最大的部分为译码中的高斯消元运算，其占整个译码总时间的91%以上；并提出优化失活译码高斯消元算法，将传统高斯消元算法中矩阵求逆的计算复杂度由$O(L^3)$降低至$O(L)$，其中L为与源数据包数量相关的参数。矩阵递归求逆译码算法[7]和矩阵降维快速译码算法[8]基于系统Raptor码的部分码——系统RaptorQ码，实现了软件优化，但速率较低。文献[9]采用CPU(central processing unit)+GPU(graphics processing unit)的模式进行系统Raptor码并行加速，使无线数据传输吞吐率达到21 Mbps。文献[10-11]

收稿日期：2018－02－02；修回日期：2018－07－12
基金项目：国家自然科学基金面上项目(61471354)
作者简介：任雁鹏(1984－)，男，博士生，主要从事纠错编码、无线通信协议等方面的研究.

则采用FPGA(filed programmable gate array)进行高斯消元运算的硬件加速，进一步提升吞吐率。

上述方案针对高斯消元运算的改进和优化，使译码性能得到了提升。但是，其译码过程中，高斯消元运算需要译码出所有源数据包，译码运算复杂度大、串行依赖度高，无法大幅降低译码延时，提高译码吞吐率。实际上，数据传输中仅存在少量丢包；如果能够只对丢失的少量数据包进行译码，则可大大降低译码复杂度。本文经过对译码过程的分解，提出了一种低复杂度的降维译码算法。该算法仅对丢失的数据包进行译码，并采用全并行的硬件加速结构，实现低延时高吞吐率的系统Raptor码译码器。

1　系统Raptor码编译码原理

系统Raptor码的编译码原理如图1所示。发送端，编码器对源数据分别进行预编码和LT(Luby transform)编码，生成编码数据。编码数据包含所有源数据和少量冗余数据。编码数据经物理信道发送给接收端。物理信道呈删除信道的特征，会有少量编码数据的丢失。接收端，译码器对接收到的数据包进行预译码和LT译码，纠正信道中丢失的数据包，恢复出所有的源数据包[1]。

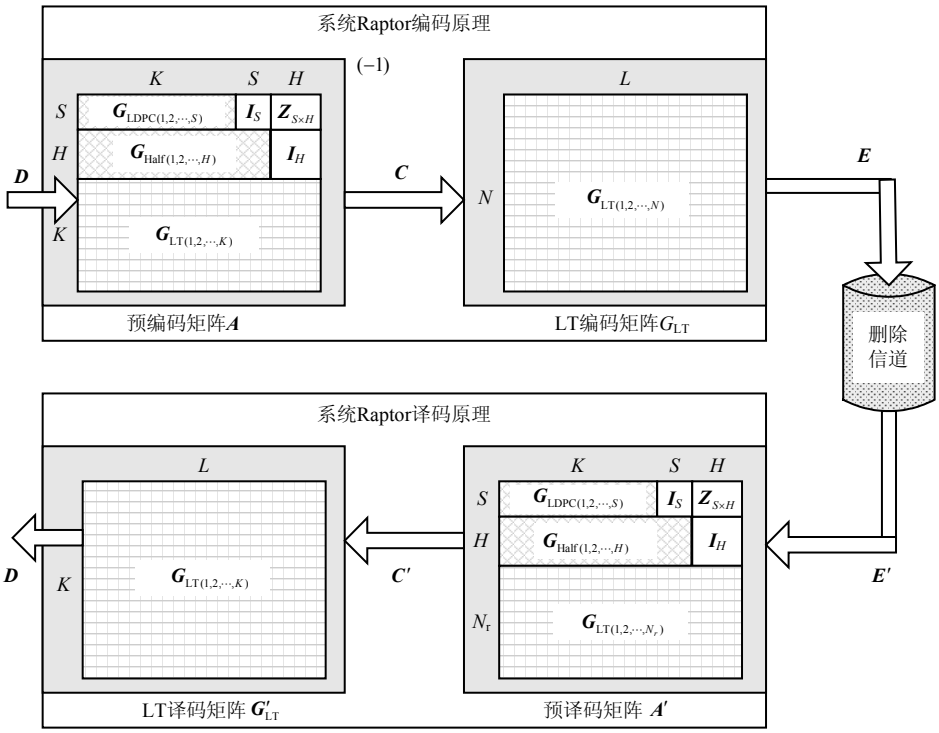

图1　系统Raptor码编译码原理图

1.1　系统Raptor码编码

系统Raptor码编码时，首先，源数据包$\boldsymbol{T}=[t_0, t_1,\cdots, t_{K-1}]$和零矩阵$\boldsymbol{Z}_{1\times(S+H)}$构成预编码输入矢量$\boldsymbol{D}=[\boldsymbol{Z}_{1\times(S+H)}, \boldsymbol{T}]^{\mathrm{T}}$，其中$K$为源数据包的数量，$S$和$H$分别为预编码参数，由$K$计算得出。输入矢量$\boldsymbol{D}$与预编码矩阵的逆矩阵$\boldsymbol{A}^{-1}$进行二进制乘法，生成$L$个预编码码字$\boldsymbol{C}=[c_0, c_1,\cdots, c_{L-1}]^{\mathrm{T}}$，其中$L=K+S+H$：

$$\boldsymbol{C}=\boldsymbol{A}^{-1}\boldsymbol{D} \tag{1}$$

然后，进行LT编码，生成编码码字$\boldsymbol{E}=[e_0, e_1, \cdots, e_{N-1}]^{\mathrm{T}}$：

$$\boldsymbol{E}=\boldsymbol{G}_{\mathrm{LT}}\boldsymbol{C} \tag{2}$$

式中，$\boldsymbol{G}_{\mathrm{LT}}$是一个$N\times L$的矩阵，$N>K$；码字$\boldsymbol{E}$包含$K$个源数据包和$N-K$个冗余包。

1.2　系统Raptor码译码

接收端，首先进行预译码。译码端接收到N_{r}个码字$\boldsymbol{E}'=[e'_0,e'_1,\cdots,e'_{N\mathrm{r}-1}]$ $(K<N_{\mathrm{r}}\leqslant N)$。$\boldsymbol{E}'$和零矩阵$\boldsymbol{Z}_{1\times(S+H)}$构成预译码输入矢量$\boldsymbol{D}'=[\boldsymbol{Z}_{1\times(S+H)}, \boldsymbol{E}']^{\mathrm{T}}$。根据接收码字的ESI(encoding symbol ID，编码符号索引号)，重建与$\boldsymbol{E}'$对应的预译码矩阵$\boldsymbol{A}'_{M\times L}$，其中$M=N_{\mathrm{r}}+S+H$。$\boldsymbol{A}'_{M\times L}$的行数$M$跟列数$L$不相等，无法进行求逆运算。在计算预译码码字$\boldsymbol{C}'=[c'_0, c'_1,\cdots, c'_{L-1}]^{\mathrm{T}}$时，需通过高斯消元法进行运算：

$$\boldsymbol{D}'=\boldsymbol{A}'_{M\times L}\boldsymbol{C}' \tag{3}$$

然后，进行LT译码，得到所有源数据包$\boldsymbol{T}$：

$$\boldsymbol{T}=\boldsymbol{G}'_{\mathrm{LT}}\boldsymbol{C}' \tag{4}$$

式中，$\boldsymbol{G}'_{\mathrm{LT}}$是一个$K\times L$的矩阵，根据已接收冗余包的

ESI构建。

传统系统Raptor码的译码结构如图2所示。接收端首先接收编码码字，并根据ESI构建 $\boldsymbol{A}'_{M\times L}$ 和 $\boldsymbol{G}'_{\mathrm{LT}}$ 。然后执行高斯消元的预译码和LT译码，得到所有源数据包。预译码和LT译码串行执行。

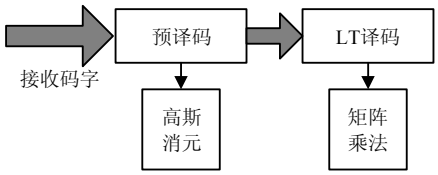

图2 系统Raptor码译码串行执行结构

在译码过程中，高斯消元需经过逐行列的递进处理，串行依赖性高，运算复杂度为$O(L^3)$[12]。即使采用硬件加速，也只能对局部运算并行实现，降低译码延时的效果有限。而且随着K的增大，高斯消元运算的复杂度大幅增长，严重影响译码延时和吞吐率。

2 系统Raptor码降维译码算法

传统系统Raptor码译码延时大，无法满足未来无线传输高吞吐率的需求。在实际中，接收数据仅存在少量丢包。只要译码出丢包数据，即可完成纠错。本节对传统系统Raptor码译码算法进行分解，结合系统Raptor码编码的已知矩阵，提出低复杂度的降维译码方案，降低译码延时。

根据式(1)与式(2)，系统Raptor码的编码可表示为：

$$\boldsymbol{E}=\boldsymbol{G}_{\mathrm{LT}}\boldsymbol{C}=\boldsymbol{G}_{\mathrm{LT}}\boldsymbol{A}^{-1}\boldsymbol{D} \tag{5}$$

经过删除信道丢包后，接收端共收到N_{r}个编码包，其中包含$K_r(K_r\leqslant K)$个源数据包$\boldsymbol{T}_r=[t'_0,\ t'_1,\cdots,t'_{K_r-1}]$和$N_r-K_r$个冗余包$\boldsymbol{R}=[r_0,\ r_1,\cdots,\ r_{N_r-K_r-1}]$。丢失的$K-K_r$个源数据包记为$\boldsymbol{T}_x=[t_{x0},\ t_{x1},\cdots,\ t_{xK-K_r-1}]$。

在式(5)中，当有丢包时，实际接收的数据包构成$\boldsymbol{E}'$。根据实际接收冗余包的ESI重新构建LT矩阵为$\boldsymbol{G}_{\mathrm{LT}}''$。输入矢量$\boldsymbol{D}_{rx}$由零矩阵$\boldsymbol{Z}_{(S+H)\times 1}$，$\boldsymbol{T}_r^{\mathrm{T}}$和$\boldsymbol{T}_x^{\mathrm{T}}$构成，其中$\boldsymbol{T}_r$和$\boldsymbol{T}_x$中的各元素根据ESI置于$\boldsymbol{D}_{rx}$的相应位置。式(5)变换为：

$$\boldsymbol{E}'=\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{C}=\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{A}^{-1}\boldsymbol{D}_{rx} \tag{6}$$

式中，$\boldsymbol{G}''_{\mathrm{LT}}$ 是一个$N_r\times L$的矩阵；$\boldsymbol{D}_{rx}$是一个$L\times 1$的矩阵。

式(6)对输入矢量$\boldsymbol{D}_{rx}$中的源数据包进行分解，拆分成两个输入矢量$\boldsymbol{D}_{r0}$和$\boldsymbol{D}_{0x}$。$\boldsymbol{D}_{r0}$和$\boldsymbol{D}_{0x}$都是$L\times 1$的矩阵。$\boldsymbol{D}_{r0}$包含所有已接收的源数据包，其中丢包的地址用0填充。$\boldsymbol{D}_{0x}$为待求的丢包矩阵，$\boldsymbol{D}_{0x}$中对应已接收数据包的地址都为0。式(6)分解为：

$$\boldsymbol{E}'=\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{A}^{-1}\boldsymbol{D}_{\mathrm{r0}}+\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{A}^{-1}\boldsymbol{D}_{\mathrm{0x}} \tag{7}$$

进一步变换，得出系统Raptor码的降维译码公式为：

$$\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{A}^{-1}\boldsymbol{D}_{0x}=\boldsymbol{E}'-\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{A}^{-1}\boldsymbol{D}_{r0} \tag{8}$$

当网络丢包率较低时，丢包数量远小于源数据包数量K。$\boldsymbol{D}_{0x}$中待求的丢包元素很少，求解$\boldsymbol{D}_{0x}$可实现系统Raptor码的降维运算。另外，在式(8)的运算中，使用预编码逆矩阵$\boldsymbol{A}^{-1}$替代预译码矩阵$\boldsymbol{A}'_{M\times L}$，避免了对$\boldsymbol{A}'_{M\times L}$的高斯消元运算，大幅降低运算复杂度和译码延时。

对式(8)的各项进一步分解：

$$\boldsymbol{A}''\boldsymbol{D}_{0x}=\boldsymbol{E}'-\boldsymbol{Y} \tag{9}$$

其中：

$$\boldsymbol{A}''=\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{A}^{-1} \tag{10}$$

$$\boldsymbol{Y}=\boldsymbol{G}''_{\mathrm{LT}}\boldsymbol{X} \tag{11}$$

$$\boldsymbol{X}=\boldsymbol{A}^{-1}\boldsymbol{D}_{\mathrm{r0}} \tag{12}$$

由式(9)可以看出，根据实际接收的数据$\boldsymbol{E}'$、参数$\boldsymbol{A}''$和$\boldsymbol{Y}$，即可求解出丢失的数据$\boldsymbol{D}_{0x}$。在译码过程中，对于给定的K值，$\boldsymbol{A}^{-1}$为已知矩阵；对于$\boldsymbol{A}''$和$\boldsymbol{Y}$，可根据已接收的源数据包$\boldsymbol{D}_{r0}$和冗余包进行并行运算得到。

经过对传统系统Raptor码算法的分解，仅对丢失的数据包进行译码，实现降维运算。运算中使用已知的预编码矩阵$\boldsymbol{A}^{-1}$替换原始的预译码矩阵$\boldsymbol{A}'_{M\times L}$，避免了复杂的高斯消元运算。实现了低复杂度的降维译码，降低了译码延时。

3 系统Raptor码降维并行译码实现

结合系统Raptor码的降维译码算法，本节给出低延时的降维并行译码结构，如图3所示。译码主要包括3个运算模块：源数据包处理模块、冗余包处理模块和丢包计算模块。源数据包处理模块对式(12)进行运算。根据已接收的源数据包，计算得出矩阵$\boldsymbol{X}$中的元素。冗余包处理模块根据已接收的冗余包构建LT译码矩阵 $\boldsymbol{G}''_{\mathrm{LT}}$ ，并计算式(10)和式(11)中 $\boldsymbol{A}''$ 和$\boldsymbol{Y}$中的元素。然后根据式(9)计算丢包$\boldsymbol{D}_{0x}$，完成整个译码过程。在该结构中，所有接收到的源数据包和冗余包并行处理，降低运算延时，提高译码吞吐率。

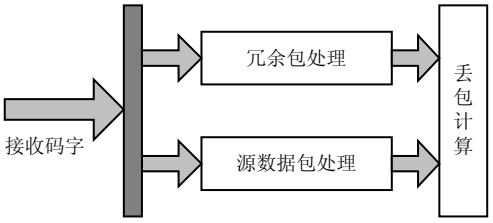

图3 系统Raptor码降维并行译码结构

根据图3的降维并行译码结构，设计系统Raptor码的全并行硬件结构，如图4所示。在系统Raptor码中，对于给定K值，其$\boldsymbol{A}^{-1}$为确定的，可以将$\boldsymbol{A}^{-1}$预先存储在寄存器中。冗余包处理模块处理接收到的冗余包，源数据包处理模块处理接收到的源数据包，最后计算丢包。所有接收的数据包存入寄存器中。下面对各模块的运算做具体说明。

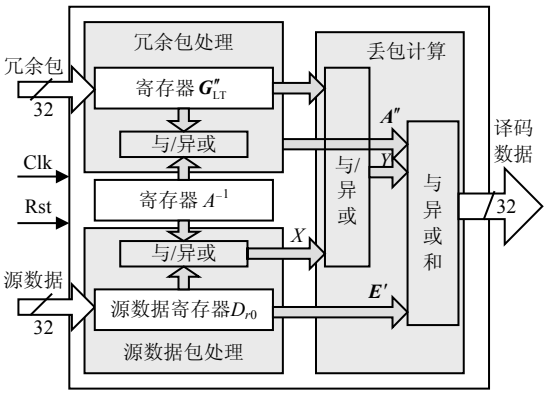

图4　系统Raptor码全并行降维译码硬件结构

3.1　源数据包处理模块

首先将接收数据包根据ESI存储到$\boldsymbol{E}'$的对应位置。发送端发送完一组数据后，接收端根据ESI查询源数据。如果全部接收则不需译码；如果存在丢包，则保存丢包的ESI并等待恢复。

然后利用接收的源数据包计算矩阵$\boldsymbol{X}$。根据接收包的ESI查找$\boldsymbol{A}^{-1}$对应的矩阵行，将$\boldsymbol{A}^{-1}$与接收包进行与和异或运算，得出$\boldsymbol{X}$中该ESI对应的值。所有源数据包的运算并行执行，降低译码延时。

3.2　冗余包处理模块

根据冗余包的ESI查找生成矩阵表$\boldsymbol{G}_{\mathrm{LT}}$，将$\boldsymbol{G}_{\mathrm{LT}}$中ESI对应的行组合在一起重新构建LT译码矩阵$\boldsymbol{G}''_{\mathrm{LT}}$。然后根据式(10)和式(11)计算$\boldsymbol{A}''$和$\boldsymbol{Y}$。$\boldsymbol{G}''_{\mathrm{LT}}$为稀疏矩阵，$\boldsymbol{A}^{-1}$和$\boldsymbol{G}''_{\mathrm{LT}}$矩阵都为0～1矩阵。在硬件实现中，将$\boldsymbol{G}''_{\mathrm{LT}}$矩阵中1对应的地址与$\boldsymbol{A}^{-1}$中的值进行与和异或运算，计算出译码逆矩阵$\boldsymbol{A}''$。同理，$\boldsymbol{G}''_{\mathrm{LT}}$与矩阵$\boldsymbol{X}$进行运算得到矩阵$\boldsymbol{Y}$。

该结构可以对所有冗余包并行处理，不需采用串行处理的模式，这样就避免了类似高斯消元中逐行列串行处理的耗时。

3.3　丢包计算模块

接收端收到N_{r}个接收包时，查询ESI并确定丢包位置，计算丢包。根据式(9)，使用矩阵$\boldsymbol{Y}$、$\boldsymbol{A}''$和$\boldsymbol{E}'$计算丢包矩阵$\boldsymbol{D}_{0x}$。

在系统Raptor码全并行的降维译码硬件结构中，所有已接收的源数据包和冗余包并行运算，源数据包处理模块和冗余包处理模块并行执行，可以在很少的时钟周期内计算出所有丢包，大幅降低了译码延时，提高译码吞吐率。

4　性能仿真及比较

结合本文提出的全并行降维译码方案，在Xilinx Kintex-7系列的410T FPGA芯片实现系统Raptor码译码器。源数据包数量K取216，网络丢包率为10^{-2}量级，单个数据包的位宽W为4 Byte。采用Xilinx软件平台进行硬件综合和功耗估计，结果如表1所示。其中逻辑单元需67 539个，占芯片总资源的26.5%，最高运行时钟为128.2 MHz，功耗为1 155 mW。

表1　系统Raptor码译码器的硬件开销与性能

类型	结果	占比
逻辑单元/个	67 539	26.5%
寄存器/个	11 486	2.26%
块存储器/个	38.5	4.84%
功耗/mW	1 155	N/A
最高时钟/MHz	128.2	N/A

4.1　译码延时比较

在本文提出的降维译码方案中，当K为216时，根据网络丢包率得出编码包数N为230。经仿真，对源数据包和冗余包的并行处理，最多仅需N个时钟周期。对丢包的计算，仅需6个时钟周期。即完成一次系统Raptor码译码共需N+6个时钟周期。

与之相比，如表2所示。文献[11]的译码器采用图2的串行译码结构。源数据包数量K为12，需延时79个时钟周期完成高斯消元运算和LT译码，而且随着K的增大，译码延时大幅增长。本文在K取12时，根据丢包率计算N为14，仅需20个时钟周期可完成译码。而且本文的译码延时随K的变化很小，有效降低了译码延时。

表2　系统Raptor码译码延时比较

源数据包数K/个	文献[11]/个	本文/个
12	79	20
216	N/A	236

4.2　传输吞吐率比较

结合上边的仿真结果，当系统运行时钟f_{Clk}取128.2 MHz时，每秒可以完成的系统Raptor码译码次数S为：

$$S = f_{\mathrm{Clk}}/(N+6)=543\ 220 \tag{13}$$

则数据吞吐率为：

$$\text{吞吐率} = 8\times W\times K\times S = 3.75\ \text{Gbps} \tag{14}$$

式中，W为每个源数据包的字节数。本文采用W为4字节的Raptor码，译码数据吞吐率可达3.75 Gbps。

本文的硬件方案与相关文献的硬件加速方案的性能比较结果如表3所示。其中文献[9]采用3 GHz的i7 CPU与732 MHz的GPU联合平台，文献[10-11]采用FPGA硬件平台，主要对高斯消元法进行硬件加速实现。

表3 系统Raptor码译码性能比较

方案	文献[9]	文献[10]	文献[11]	本文
硬件平台	CPU + GPU	Altera StratixIII	Xilinx Kintex-7	Xilinx Kintex-7
最高时钟/MHz	3 000	100	230	128.2
归一化功耗/mW	N/A	9.88×10^{-4}	3.24×10^{-4}	3.08×10^{-7}
吞吐率/bps	21 M	1.05 M	46.5 M	3.75 G

从表3可以看出，本文的降维译码方案，仅对丢包进行译码，并且采用已知的预编码矩阵，替代需进行高斯消元的预译码矩阵，避免了对所有源数据包译码的高斯消元运算，实现了低复杂度的降维译码。在全并行降维译码的硬件结构中，所有已接收的源数据包和冗余包并行运算，大幅降低了译码延时。在单位时间内完成更多次的译码运算，实现了高吞吐率的译码和传输。

5 结 束 语

本文对系统Raptor码高复杂度的高斯译码算法进行改进，提出了一种低复杂度的降维译码算法。在该算法中，仅对少量丢包进行译码，替代传统译码算法中对全部源数据包译码的高斯消元运算，实现译码维度的大幅降低。并采用全并行的结构实现降维译码算法，最终在Xilinx FPGA XC7K410T上实现了该译码器。测试结果表明，当网络丢包率在10^{-2}以下时，译码器译码延时大幅降低；其译码吞吐率可达到3.5 Gbps，是相同平台下采用高斯消元译码的80倍以上。该方案有效提高了系统Raptor码的译码效率，降低了译码延时，提高了数据传输吞吐率。

参 考 文 献

[1] IETF RFC 5053(2007): Raptor forward error correction scheme for object delivery[S]. California: IETF Proposed Standard, 2007.

[2] IETF RFC 6330(2011): RaptorQ forward error correction scheme for object delivery[S]. California: IETF Proposed Standard, 2011.

[3] RYU E, JAYANT N. Home gateway for three-screen TV using H.264 SVC and raptor FEC[J]. IEEE Transactions on Consumer Electronics, 2011, 57(4): 1652-1660.

[4] 武岩波, 朱敏. 一种用于水声通信的喷泉码最大似然译码方法[J]. 电子与信息学报, 2016, 38(2): 288-293.
WU Yan-bo, ZHU Min. Maximum likelihood decoding of fountain codes in underwater acoustic communication[J]. Journal of Electronics & Information Technology, 2016, 38(2): 288-293.

[5] 朱文杰, 易本顺. 一种改进的喷泉多选择序列峰均比降低算法[J]. 湖南大学学报(自然科学版), 2016, 43(2): 124-129.
ZHU Wen-jie, YI Ben-shun. An improved fountain multi-choice sequence algorithm for peak-to-average power ratio reduction[J]. Journal of Hunan University(Natural Sciences), 2016, 43(2): 124-129.

[6] MLADENOV T, NOOSHABADI S, KIM K. Efficient GF(256) raptor code decoding for multimedia broadcast/multicast services and consumer terminals[J]. IEEE Transactions on Consumer Electronics, 2012, 58(2): 356-363.

[7] LU Yi-pin, LAI I-wei, LEE C, et al. Low-complexity decoding for raptorQ codes using a recursive matrix inversion formula[J]. IEEE Wireless Communications Letters, 2014, 3(2): 217-220.

[8] GUO Xiao, ZHANG Geng-xin, TIAN Chang, et al. Fast decoding for raptorQ codes using matrix dimensionality reduction[J]. Electronics Letters, 2014, 50(16): 1139-1141.

[9] HU Lin-jia, NOOSHABADI S, MLADENOV T. Forward error correction with raptor GF(2) and GF(256) codes on GPU[J]. IEEE Transactions on Consumer Electronics, 2013, 59(1): 273-280.

[10] MLADENOV T, NOOSHABADI S, KIM K. Implementation and evaluation of raptor codes on embedded systems[J]. IEEE Transaction on computers, 2011, 60(12): 1678-1691.

[11] LU Yi-pin, LAN Wei, CHENG Yi-feng, et al. An implementation of a fountain code-based MIMO-OFDM receiver for real-time wireless video streaming[C]//2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). Abu Dhabi: IEEE, 2015: 696-703.

[12] 李越, 张立军, 李明齐, 等. 一种RaptorQ码的低复杂度编码算法[J]. 电视技术, 2017, 41(3): 57-60.
LI Yue, ZHANG Li-jun, LI Ming-qi, et al. Low complexity encoding method for raptorQ code[J]. Video Engineering, 2017, 41(3): 57-60.

编 辑 蒋 晓