循环码编解码器设计与实现

1 实验目的与要求

本次实验将专注于最基本的循环码的相关概念，针对一个$(7,4)$循环码，设计实现其编解码器。本实验最多两人一组。实验大体要求列举如下，详情参考实验材料。

- 阅读理解实验材料，完成3.1的思考题。
- 完成编译码函数的软件设计与实现，并搭建简单的误码率仿真系统。
- 完成编码器Encoder1的硬件设计与实现，并通过VLSI架构优化或者算法变型，设计实现计算延迟更小的编码器Encoder2。
- 完成译码器Decoder1和硬件设计与实现，并利用循环码的数学性质，设计实现不需要查找表的更高效的译码器架构Decoder2。

2 循环码基础

循环码作为信道编码中最基本的一种线性分组码，在各类应用中均有重要作用。在存储系统以及光纤通信系统中，BCH (Bose-Chaudhuri-Hocquenghem)码和RS (Reed-Solomon)码被广泛采用，而BCH码和RS码正是最常见的两类循环码。此外，CRC (Cyclic Redundancy Check)码作为最简单的一类循环码，几乎出现于各类通信协议中。因此，了解掌握循环码的基本概念，是掌握更高级的信道编码方案的基础。

2.1 循环码的定义

对于一个n维向量$\boldsymbol{v}=(v_0,v_1,\cdots,v_{n-1})$，可以将它写成多项式的形式，即$v(X)=v_0+v_1X+v_2X^2+\cdots+v_{n-1}X^{n-1}$。本实验中多项式与向量形式可以互换。在本实验中，所有数均是二元域$\{0,1\}$中的元素，他们之间的**乘法**等价于**与运算**，**加法**等价于**异或运算**。此外，二元域上的运算还满足**结合律**、**交换律**和**分配率**。考虑二元域$\{0,1\}$中的3个元素a,b,c，他们满足

$$\begin{aligned} a+b+c=a+(b+c),\, a\times b\times c=q\times(b\times c);\\ a+b=b+a,\, a\times b=b\times a;\\ a\times(b+c)=&a\times b+a\times c. \end{aligned} \tag{1}$$

对于一个(n,k)线性分组码的码字$v=(v_0,v_1,\cdots,v_{n-1})$，将环右移1位得到

$$v_r=(v_{n-1},v_0,\cdots,v_{n-2}), \tag{2}$$

循环左移1位得到

$$v_l=(v_1,v_2,\cdots,v_{n-1},v_0). \tag{3}$$

显然对一个码字循环左移i位等价于将它循环右移$n-i$位，因此本实验中提到的移位，**均表示循环右移**。下面给出循环码的最基本的定义。

Definition 1. *如果一个(n,k)线性分组码C的每一个码字的循环移位仍然是C中的码字，那么称C是一个循环码(cyclic code)。*

为了更好地表示一个循环码，并且探究其更深层次的数学规律，生成多项式的概念被引出，并且有Theorem 1和Theorem 2。其中Theorem 2告诉了我们一个循环码码字的生成方法：

- 确定(n,k)循环码的生成多项式$g(X)$；
- 将待编码的k个信息比特用多项式的形式表示$m(X)=m_0+m_1X+\cdots+m_{k-1}X^{k-1}$；
- 编码得到码字多项式$v(X)=m(X)g(X)$。

根据以上内容，完成Question 1。

Theorem 1. 一个(n,k)循环码C可以由一个次数为$n-k$的多项式$g(X)$表征，这个多项式称为C的生成多项式，C中任意一个码字多项式$v(X)$，均可以表示成生成多项式$g(X)$和另外一个次数小于等于$k-1$的多项式$a(X)$的乘积：$v(X)=a(X)g(X)$。

Theorem 2. 如果$g(X)$是一个次数为$n-k$的多项式并且整除X^n+1，则$g(X)$生成一个(n,k)循环码。

Question 1. 多项式X^7+1可以被因式分解为$X^7+1=(1+X)(1+X+X^3)(1+X^2+X^3)$，请根据$X^7+1$的因式构造两个$(7,4)$循环码，列出每个码中的所有码字及其对应的信息多项式（即需要列出$\boldsymbol{v}$和对应的$m(X)$），并验证构造出的两个码是否满足循环码的最基本定义。

2.2 循环码的编码

在循环码的定义一节中我们给出了一个编码方法，本节将给出另外一种**系统编码**的编码方法。在很多通信或者存储系统中，希望原本的信息在经过信道编码后，还能**显示地**出现在码字中。例如一个k比特信息向量$\boldsymbol{m}$经过一个(n,k)码信道编码得到码字$\boldsymbol{v}$，码字$\boldsymbol{v}$中的有k个比特与$\boldsymbol{m}$的k个比特分别相等，这样的编码方式称为系统编码。通常，我们会将码字中与信息向量相等的比特放在一起，即$\boldsymbol{v}=(\boldsymbol{m},\boldsymbol{p})$或者$\boldsymbol{v}=(\boldsymbol{p},\boldsymbol{m})$。本实验的系统编码令$\boldsymbol{v}=(\boldsymbol{p},\boldsymbol{m})$，即编码后的最后$k$比特是信息比特，前$n-k$比特是编码生成的校验比特。

信息向量对应的多项式为$m(X)=m_0+m_1X+\cdots+m_{k-1}X^{k-1}$。根据上述系统编码方法，编码后信息位将被放在码字的最后k位，那么对应一个新的向量

$$\boldsymbol{m}^{'}=(\underbrace{0,0,\cdots,0}_{n-k},m_0,m_1,\cdots,m_{k-1}). \tag{4}$$

这个新向量对应的多项式为$m^{'}(X)=X^{n-k}m(X)$。将$m^{'}(X)$除以$g(X)$可以得到

$$m^{'}(X)=X^{n-k}m(X)=a(X)g(X)+b(X), \tag{5}$$

于是

$$X^{n-k}m(X)-b(X)=X^{n-k}m(X)+b(X)=a(X)g(X) \tag{6}$$

（这里需要注意的是二元域的运算中减法就等于加法）。显然$X^{n-k}m(X)+b(X)$就是一个循环码的码字，因此我们得到$b(X)$ 就可以得到一个循环码码字。这里$b(X)$是$m^{'}(X)=X^{n-k}m(X)$除以$g(X)$得到的余项，那么$b(X)$的**次数小于**$n-k$，于是我们有$b(X)=b_0+b_1X+\cdots+b_{n-k-1}X^{n-k-1}$。注意到$\boldsymbol{m}^{'}$的前$n-k$个比特为0，刚好可以将$b(X)$的系数填入这$n-k$个比特。于是编码后的码字表示为

$$\boldsymbol{v}=(b_0,b_1,\cdots,b_{n-k-1},m_0,m_1,\cdots,m_{k-1}). \tag{7}$$

综上，循环码的一种系统编码方法为：

- 用X^{n-k}乘上$m(X)$；
- 用$X^{n-k}m(X)$除以生成多项式$g(X)$，得到余式$b(X)$；
- 编码得到码字多项式$v(X)=b(X)+X^{n-k}m(X)$。

根据以上内容，完成Question 2。

Question 2. 将*Question 1*中构造的两个循环码换成系统编码方法重新构造，仍然列出每个码的所有码字及其对应的信息多项式，并验证构造出的两个码是否满足循环码的最基本定义。

2.3 循环码的生成矩阵

一个(n,k)循环码的生成矩阵是一个行满秩的大小为$k\times n$的矩阵$\boldsymbol{G}$，对于一个信息向量$\boldsymbol{m}$，利用$\boldsymbol{G}$编码得到码字$\boldsymbol{v}=\boldsymbol{mG}$，这可以看作是循环码的又一种编码方法（事实上这种编码方法对所有线性分组码均适用）。如何确定循环码的生成矩阵？事实上，用循环码中任意k个线性无关的码字就可以组成他的一个生成矩阵。例如，我们采用码字多项式$g(X)$，$Xg(X)$，$\cdots$，$X^{k-1}g(X)$对应的码字，就可以组成如下一个有效的生成矩阵：

$$\boldsymbol{G}=\begin{pmatrix} g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} & 0 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & \cdot & 0 \\ \vdots & & & & & & & & & & & & & & \vdots \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} \end{pmatrix}. \tag{8}$$

为得到一个系统形式的生成矩阵，可以将上述生成矩阵进行初等行变换，使得矩阵的后k列组成一个单位矩阵$\boldsymbol{I}$。利用系统形式的生成矩阵乘上信息向量即进行系统编码。

除了初等行变换，也可以利用循环码的性质得到系统形式的生成矩阵。对于一个(n,k)循环码C，考虑k个多项式X^{n-k}，X^{n-k+1}，$\cdots$，X^{n-1}，将这k个多项式分别除以生成多项式$g(X)$，得到$X^{n-k+i}=a_i(X)g(X)+b_i(X)$，其中$i=0,1,2,\cdots,k-1$，$b_i(X)$是对应的余式。显然$X^{n-k+i}+b_i(X)=a_i(X)g(X)$是一个循环码的码字，因此我们得到了$k$个码字。因为$b_i(X)$是除以$g(X)$得到的余项，因此他的次数小于$n-k$，所以对于$i=0,1,2,\cdots,k-1$，$k$个码字多项式$X^{n-k+i}+b_i(X)$是线性无关的。综上，我们得到的这$k$个码字可以组成一个系统形式的生成多项式。

完成Question 3。

Question 3. 写出*Question 1*中生成多项式为$g(X)=1+X+X^3$的一个非系统生成矩阵，再利用上述k个多项式X^{n-k}，X^{n-k+1}，$\cdots$，X^{n-1}求余的方法得到相应的系统形式的生成矩阵。

2.4 循环码的译码

循环码的最基本的译码算法请参考课件，下面我们仅给出简要的描述。在本实验中，**我们也称伴随式为"校正子"**，伴随式多项式称为校正子多项式。对于一个循环码码字，它满足$v(X)=a(X)g(X)$，即可以整除生成多项式。假设因为噪声等原因，码字在传输或者存储过程中引入了错误，则出错的码字记为$\boldsymbol{r}=\boldsymbol{v}+\boldsymbol{e}$，其中$\boldsymbol{e}$ 是一个n 比特长的向量，称为错误图样。假如一个$n=7$的码字的第1和第5个比特发生错误，则其错误图样为$\boldsymbol{e}=(1,0,0,0,1,0,0)$。在错误图样存在的情况下，$r(X)$不一定能整除$g(X)$，则校正子多项式定义为

$$S(X)=r(X)\ mod\ g(X)=e(X)\ mod\ g(X). \tag{9}$$

显然，当$e(X)$的次数小于$n-k$的时候，$r(X)$无法整除$g(X)$，校正子多项式$S(X)\neq 0$，有错误一定被被探测到；当$e(X)$次数大于等于$n-k$并且$e(X)$整除$g(X)$的时候，校正子多项式$S(X)=0$，错误无法被探测。因此我们的译码算法仅仅考虑错误比较少，能够成功纠错的情况。

当确定了一个循环码，那么次数小于$n-k$并且可以被成功纠错的错误图样就随之确定，这些错误图样组成的集合记为$\boldsymbol{E}$。我们可以提前算出校正子多项式$S(X)$与$\boldsymbol{E}$中的错误图样多项式$e(X)$之间的对应关系，在接收到含有错误的码字的时候，计算出校正子多项式$S(X)$，根据对应关系得到错误图样多项式$e(X)$，则原码字恢复为$v(X)=r(X)+e(X)$（二元域上减法等于加法）。

下面给出一个重要的定理，如Theorem 3所示。Theorem 3告诉我们，假设我们把一个含有错误图样（不含错误的码字的错误图样可以看成0）的码字多项式$r(X)$循环移位1位得到一个新的多项式$r^{(1)}(X)$，计算$r^{(1)}(X)$的校正子多项式$S^{(1)}(X)$除了直接用$r^{(1)}(X)$除以$g(X)$，还可以用$r(X)$的校正子多项式$S(X)$先乘上X，再除以$g(X)$。

Theorem 3. $r(X)$是一个(n,k)循环码C的接收码字多项式，即$r(X)=v(X)+e(X)$。记$r^{(1)}(X)$是$r(X)$循环移位1次后的多项式。$r(X)$和$r^{(1)}(X)$的校正子多项式分别记为$S(X)$和$S^{(1)}(X)$，他们满足

$$S^{(1)}(X)=XS(X)\ mod\ g(X), \tag{10}$$

其中$g(X)$是C的生成多项式。

记$r^{(1)}(X)=v^{(1)}(X)+e^{(1)}(X)$，因为$v^{(1)}(X)$是$v(X)$的移位，所以$v^{(1)}(X)$仍是一个循环码码字。$e^{(1)}(X)$是$e(X)$循环移位*1*位。于是对于错误图样多项式，上式变为

$$e^{(1)}(X)\ mod\ g(X)=S^{(1)}(X)=XS(X)\ mod\ g(X)=X(e(X)\ mod\ g(X))\ mod\ g(X). \tag{11}$$

即错误图样多项式$e(X)$循环移位*1*位后的错误图样多项式$e^{(1)}(X)$的校正子多项式$S^{(1)}(X)$，可以由$e(X)$的校正子多项式$S(X)$乘上X再除以$g(X)$得到。

3 实验内容

本次实验包括若干思考题以及代码仿真，**代码仿真部分仅考虑Question 1中多项式**$g(X)=1+X+X^3$**构成的**$(7,4)$**循环码**。

3.1 思考题

- 完成Question 1，Question 2，Question 3。
- 复习课件中二元域多项式除法电路的实现方式。任意考虑一个多项式$g(X)$作为被除数多项式，要求**其次数不小于5**，画出对应的除法电路图。任意考虑一段输入序列，要求**其长度不小于10**，列出该序列进入除法电路过程中每一个始终周期的寄存器的值。
- 用上述除法电路可以实现某个生成多项式为$g(X)$的循环码的校正子计算电路，假设码字已经全部输入到电路中，此时寄存器存放的即是校正子多项式$S(X)$。接下来将输入固定为0，请验证下一个周期结束后寄存器存放的是$XS(X)\ mod\ g(X)$。考虑$g(X)=1+X+X^3$构成的$(7,4)$循环码的任意一个码字序列$\boldsymbol{r}=v(X)+e(X)$（$e(X)\neq 0$）作为除法电路的输入序列。
- 一个二元域上的线性分组码C的**最小距离**d_{min}是指C中任意两个码字之间的最小汉明距离，请结合Question 1，给出$g(X)=1+X+X^3$构成的$(7,4)$循环码的最小距离，并验证其所有**非零**码字中，1的个数**大于等于**最小距离。
- 假设一个线性分组C码的最小距离是3，请思考C是否能利用译码算法纠正任意**大于等于**3个随机错误比特？如果有任意1个或者2个随机错误发生，C是否能检测出错误的存在？是否可以为C设计某种译码算法，当有任意2个错误比特发生时，可以正确纠正错误？假设译码算法的输入是二元域上的码字序列。（提示：最小距离为3表示存在两个码字之间的汉明距离是3，那么对于其中一个码字，改变其相应位置上的3个比特可以变成另一个码字。建议以$g(X)=1+X+X^3$构成的$(7,4)$循环码的码字为例分析说明。）

3.2 代码仿真

接下来将$g(X)=1+X+X^3$构成的$(7,4)$循环码简称为$(7,4)$循环码。本次实现的软件代码可以用任意一种语言（建议MATLAB或者C/C++）实现。

软件：

- **编码算法**：实现$(7,4)$循环码**系统编码**函数，要求函数的输入是待编码信息序列，输出是编码后的码字。严格参考2.2的描述，注意信息位放在码字的后k比特。

- **译码算法：**已知$(7,4)$循环码**可以纠正任意1个随机错误比特**，请给出错误图样与校正子多项式之间的对应关系，并实现$(7,4)$循环码的译码函数，要求函数的输入是含有错误图样的码字，输出是译码后的码字。

- **理论计算：**考虑BSC信道（二进制对称信道），在这个信道中传输的比特发生错误的概率为p（即一个比特有p的概率变成他的补，即0变成1，1变成0），且不同比特之间的传输相互独立。计算$(7,4)$循环码的一个码字在经过一个BSC信道的传输后，发生错误的概率，以及仅发生1个错误比特的概率。考虑$p = 1/2$和$1/4$两种情况。

- **误码率仿真：**随机生成N个信息序列（结合前面理论计算的结果选取合适的N），进行$(7,4)$循环码的系统编码，将这些码字通过比特错误概率为p的BSC信道传输（即用软件代码仿真模拟每个比特发生错误的概率为p），将通过信道的码字输入给译码函数进行译码纠错。考虑$p = 1/2$和$1/4$两种情况，并统计误帧率（Error Fame Rate）和误比特率（Error Bit Rate），与理论结果进行对比。

硬件：

- **编码器及其优化：**画出利用线性反馈移位寄存器实现的$(7,4)$循环码**系统编码**的串行编码器电路（即课件中给出的$m(X)$先乘上X^{n-k}的电路），并编写相应的Verilog代码，**该编码器记为Encoder1**。以减少**延迟**为目标（这里将延迟定义为从输入$m(X)$到生成$v(X)$全部比特所需要的时间），设计新的编码器，画出电路图，并编写相应的Verilog 代码，**该编码器记为Encoder2**。Encoder2可以是采用新的编码方法，也可以是对Encoder1 进行VLSI 优化。

  请结合软件仿真平台（生成少量码字验证Verilog代码正确性，码字数量不少于5个）验证两种编码器的正确性。Vivado综合两种编码器（用各自能达到的最高时钟频率），并比较两者的资源消耗、吞吐率（比特数除以时间）、延迟（时间）。

- **译码器：**按照课件介绍的方法，设计$(7,4)$循环码的译码电路（保存错误图样与校正子多项式之间关系的查找表不需要画出详细电路实现，用框图表示即可），并编写相应的Verilog代码。**该译码器记为Decoder1**。

- **译码器的优化：**根据Theorem 3我们知道，假如在某一时刻，已经计算得到了$r(X)$的校正子多项式$S(X)$，如果想要计算$r^{(1)}(X)$的校正子多项式，可以用$S^{(1)}(X) = XS(X)\ mod\ g(X)$。结合思考题的第3点可知，在$S(X)$已经存放在除法电路的寄存器中的情况下，仅需要一个时钟周期来计算$XS(X)\ mod\ g(X)$。

  前面已经提到，$(7,4)$循环码可以纠正任意1个随机错误比特。现考虑一个码字$v(X)$。假设错误比特发生在最高位，含有错误的码字记为$r_6(X) = v(X) + e_6(X)$，错误图样为$e_6(X) = X^6$，对应的校正子记为$S_6(X)$。假设错误特比发生在第6位，即码字$\boldsymbol{v} = (v_0, v_1, \cdots, v_6)$的第6个比特$v_5$发生了错误，对应的码字记为$r_5(X) = v(X) + e_5(X)$，错误图样为$e_5(X) = X^5$，对应的校正子记为$S_5(X)$。根据Theorem 3，有$XS_5(X)\ mod\ g(X) = S_6(X)$，这是因为$e_6(X)$是$e_5(X)$的一个循环移位。

  根据$(7,4)$循环码的以上特点，可以设计一个**不需要查找表**的译码器，**记为Decoder2**。设计思路为，译码从最高比特到最低比特串行地处理，在码字$r(X)$全部进入除法电路后（得到了校正子多项式$S(X)$）的每个周期，比较$S(X)$是否等于$S_6(X)$。若不相等，将除法电路的输入固定为0，然后除法电路继续工作更新$S(X) = XS(X)\ mod\ g(X)$，同时将$r(X)$右移。根据设计思路，补全Decoder2的实现细节，并画出相应电路图，并编写相应的Verilog代码。

- 请结合软件仿真平台（生成少量码字验证Verilog代码正确性，码字数量不少于5个）验证两种译码器的正确性。Vivado综合两种译码器（用各自能达到的最高时钟频率），并比较两者的资源消耗、吞吐率（比特数除以时间）、延迟（时间）。

3.3 提交要求补充

- 请将实验报告、软件代码、硬件代码打包成一个文件夹压缩后提交。
- 文件夹和压缩包均以以下格式命名：final02_组号。“final02”表示选择的是本实验，组号为两位数字，个位数需要在前面补零，例如组号为3则命名为“final02_03”。
- 文件夹分为3个部分：实验报告（一个PDF文件），软件代码（一个文件夹），硬件代码（一个文件夹）。
- 实验报告须为pdf文件，要包括本实验中要求的各点（思考题、软件设计思路、硬件设计思路、电路图、验证结果图、硬件综合结果、成员分工等等）。报告开头写出成员名称和学号。
- Vivado综合选择V7系列VC709（同Lab1）。
- 涉及数据的地方请附上软件截图，例如Vivado综合结果、软件仿真的BSC信道下的误码率等。
- 软件代码需要给出必要的注释（输入输出的解释，以及必要的逻辑单元的注释）。
- 硬件代码需要包括设计部分（电路实现）和验证部分（testbench），代码给出必要注释（输入输出的解释，以及必要的逻辑单元的注释）。
- **严禁抄袭**。