Binary Polynomial Division

Instructor: James K Beard, PhD

Email:. jkbeard@temple.edu, jkbeard@comcast.net

Web Page: http://temple.jkbeard.com

Table of Contents

1 Principles of Binary Arithmetic

Cyclic codes, finite fields of order 2^k and other signal processing topics in EE521 use binary arithmetic to define operations with signal bits and codeword bits. The base operations are addition, subtraction, multiplication, and division. Arithmetic is done modulo 2, and the results of addition, subtraction, and multiplication simply use ordinary integer arithmetic and take the result modulo 2. The operations are summarized as follows.

- Binary addition of two bits results in the exclusive logical OR of the two bits, usually summarized by the mneumonic XOR. The result of an XOR is a zero if the two bits are the same, and a one if the two bits are different.
- Binary subtraction is the same as binary addition because, modulo 2, -1 is the same as +1.
- Binary multiplication is the same as a logical AND of the two bits. The result of an AND is a one if both the bits are ones; the result is zero if either or both of the input bits is zero.
- Division in binary arithmetic is rarely used, but it is defined. Division of a numerator bit by one results in the numerator bit as the quotient, whether it is zero or one. Division by zero is not defined.

2 Principles of binary polynomials

Vectors of bits such as message words or codewords are represented by binary polynomials in cyclic codes and in finite fields of order 2^k. The coefficients of the polynomials are the bits, and the power of the independent variable is used to denote the position of the bit in the word. The purpose of using the polynomial representation is to define operations on the words using conventional polynomial arithmetic – polynomial addition, subtraction, multiplication and division – except that the coefficients are taken modulo 2 in the completed operations. Note that subtraction of polynomials is the same as addition in binary arithmetic. This can be used to simplify the operations of division of polynomials, as we will show below.

3 Division of Binary Polynomials

3.1 The Fundamental Definition of Division

Division of binary polynomials is based on the fundamental definition of division, which we state here: a number *n* divided by a denominator *d* is characterized by a quotient *q* and a remainder *r* which are related by

$$\frac{n}{d} = q + \frac{r}{d} \tag{3.1}$$

or, multiplying both sides by the denominator *d*,

$$n = d \cdot q + r \tag{3.2}$$

where the remainder *r* is indivisible by the denominator *d*. For the purposes of polynomial division, this means that the order of the remainder polynomial *r* is less than that of the denominator polynomial *d*.

3.2 Division of Polynomials in General

Long division of polynomials uses (3.2) in recursively to reduce the order of the numerator by one. Each such operation produces a "remainder" which is of order less than that of the numerator.

We illustrate this process by writing the numerator and denominator polynomials explicitly,

$$\begin{aligned} n(x) &= n_K \cdot x^K + n_{K-1} \cdot x^{K-1} \ldots n_1 \cdot x + n_0 \\ &= \sum_{i=0}^{K} n_i \cdot x^i \\ d(x) &= d_N \cdot x^N + d_{N-1} \cdot x^{N-1} \ldots d_1 \cdot x + d_0 \\ &= \sum_{i=0}^{N} d_i \cdot x^i \end{aligned} \tag{3.3}$$

If the order *K* of the numerator polynomial is less than the order *N* of the denominator polynomial, then the quotient is zero and the remainder is the numerator polynomial; this

is a trivial case so we assume here that the order of the numerator exceeds that of the denominator.

We can define a step that produces a form similar to that of (3.2) and leaves us with a division operation in which the order of the numerator is reduced. We multiply the denominator by $\left(n_K / d_N\right) \cdot x^{K-N}$ and subtract this product from the numerator:

$$\begin{aligned} n_1(x) &= n(x) - \left(\left(\frac{n_K}{d_N} \right) \cdot x^{K-N} \right) \cdot d(x) \\ &= \left(n_{N-1} - \frac{n_K}{d_N} \cdot d_{N-1} \right) \cdot x^{N-1} + \ldots \end{aligned} \tag{3.4}$$

Now, we note that

$$n(x) = d(x) \cdot \left(\left(\frac{n_K}{d_N} \right) \cdot x^{K-N} \right) + n_1(x) \tag{3.5}$$

which is of the same form as (3.2). We have identified the highest-order term of *q(x)* and reduced the order of the numerator. If the order of $n_1(x)$ is less than that of *d(x)*, we are done, otherwise we repeat the operation with $n_1(x)$ as the numerator.

We pose operation as a recursion. To initialize, we set

$$\begin{aligned} n_0(x) &= n(x) \\ q_0(x) &= 0 \end{aligned} \tag{3.6}$$

Each step reduces the order of the numerator and adds a term to the quotient:

$$\begin{aligned} n_{i+1}(x) &= n_i(x) - \left(\left(\frac{n_{i,K-i}}{d_N} \right) \cdot x^{K-N-i} \right) \cdot d(x) \\ q_{i+1}(x) &= q_i(x) + \left(\frac{n_{i,K-i}}{d_N} \right) \cdot x^{K-N-i} \end{aligned} \tag{3.7}$$

The recursion stops when the order of $n_i(x)$ is less than that of the denominator; the remainder is this polynomial.

3.3 Division of Polynomials in Binary Arithmetic

Since binary arithmetic is far simpler than that for real numbers or ordinary integers, we can simplify the polynomial division process. We recognize that d_N is always 1 and that subtraction is the same as addition. We initialize using (3.6). At each step, we examine the highest order bit of $n_i(x)$, $n_{i,K-i}$, and, if it is 1, we perform the operation

$$\begin{aligned} n_{i+1}(x) &= n_i(x) + x^{K-N-i} \cdot d(x) \\ q_{i+1}(x) &= q_i(x) + x^{K-N-i} \end{aligned} \tag{3.8}$$

If the bit $n_{i,K-i}$ is zero, we do NOT perform (3.8) and simply go to the next i. Note that the polynomial coefficients are taken modulo 2 after addition in (3.8).

3.4 Example Using a Spreadsheet

This recursion is very simple, and can be implemented with any spreadsheet. An Excel implementation, with example, is included below.

Binary Polynomial Division

	x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^10	x^11	x^12	x^13	x^14	x^15	x^16
Denominator polynomial	1	0	1	0	1	0	0	1	0	1	1						
Numerator Polynomial	1	0	1	1	0	1	0	0	1	1	1	0	0	1	0	0	1

Division Process	**Yes/No**																	Highe
Numerator		1	0	1	1	0	1	0	0	1	1	1	0	0	1	0	0	**1**
x^6*d	**1**	0	0	0	0	0	0	1	0	1	0	1	0	0	1	0	1	1
Sum		1	0	1	1	0	1	1	0	0	1	0	0	0	0	0	**1**	
x^5*d	**1**	0	0	0	0	0	1	0	1	0	1	0	0	1	0	1	1	
Sum		1	0	1	1	0	0	1	1	0	0	0	0	1	0	**1**		
x^4*d	**1**	0	0	0	0	1	0	1	0	1	0	0	1	0	1	1		
Sum		1	0	1	1	1	0	0	1	1	0	0	1	1	**1**			
x^3*d	**1**	0	0	0	1	0	1	0	1	0	0	1	0	1	1			
Sum		1	0	1	0	1	1	0	0	1	0	1	1	**0**				
x^2*d	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0				
Sum		1	0	1	0	1	1	0	0	1	0	1	**1**					
x^1*d	**1**	0	1	0	1	0	1	0	0	1	0	1	1					
Sum		1	1	1	1	1	0	0	0	0	0	**0**						
x^0*d	**0**	0	0	0	0	0	0	0	0	0	0	0						
Sum		1	1	1	1	1	0	0	0	0	0							
Quotient		0	1	0	1	1	1	1										
Remainder		1	1	1	1	1	0	0	0	0	0							

Figure 1 Binary Polynomial Division with a Spreadsheet

Note that the columns headers are the powers of x in the polynomials. The original numerator polynomial is bordered by a double line, and the denominator polynomial is bordered by a single line.

In the table of values headed by **Division Process**, we begin with the numerator. This is equivalent to the initialization given as (3.6). We then multiply the denominator by x^{N-K} (in this example, N is 16 and K is 10, so we multiply by x^6. This shifts the bits of the denominator so that the most significant bit falls below the most significant bit of the numerator.

The most significant bit of the numerator now determines whether or not we execute (3.8) at this iteration. This bit is copied to the column labeled **Yes/No** and also to the most significant bit of the Quotient, the x^6 term, near the bottom of the sheet.

At each iteration, the denominator is added to x^{K-N-i} using binary arithmetic. If the most significant bit of the denominator as carried to the **Yes/No** column is zero, as is the case

in two of the rows of the example, the corresponding entries for the bits of the denominator are made zero by multiplying them by this bit.

3.5 Manual Computation

The execution of polynomial division with binary arithmetic is so simple that it can be done by hand quickly, and the process is not particularly error-prone. In manual execution of (3.8), you should follow the general format set in the spreadsheet example:

- Use column headings of x^0, x^1, x^2, etc. and tabulate all your polynomial coefficients under the corresponding column heading. This eliminates confusion as to the bit position in succeeding computations, which is critical to keeping hand computation simple.
- State the numerator and denominator polynomials in terms of the coefficients as bits, using 0 and 1.
- Re-state the denominator polynomial, multiplied by the appropriate power of x and placed in the corresponding columns for each coefficient.
- Use the most significant bit of the numerator at each step to determine the corresponding bit of the quotient, and to determine whether (3.8) is to be executed. Make this bit explicit at the next row, as with the **Yes/No** column in the spreadsheet example, as well as in the quotient. This means that all elements of a particular instance of (3.8) are right on the rows where they are needed to understand and execute each row.

When you are done, you might check your work by executing (3.2) in binary arithmetic.