Practical Number 3

Aim: To multiply two polynomials in GF(2^n) with a given irreducible polynomial.

Theory: An n-bit number satisfying the properties in GF(2^n) can be represented with a polynomial of degree n-1 as

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots\ldots + a_1x^1 + a_0x^0$$

where x^i is called the i^{th} term and a_i is called coefficient of the i^{th} term. The power of x defines the position of the bit in the n-bit word and the coefficient of the term define the value of the bits. Thus the polynomial $x^5 + x^2 + x$ represents the 8-bit word 00100110. Addition and multiplication operation on polynomials involves two operation: operation on the coefficients and the operations on the two polynomials. Since coefficients are made of 0 and 1, they are represented in GF(2) field while the exponent is in GF(2^n)field.

When multiplying two polynomials we multiply the coefficients while the exponents are added. Thus multiplying x^i with x^j results in x^{i+j} , i.e. a term may result with degree more than n-1. Thus, the result needs to be reduced using a modulus. The modulus used here is a prime polynomial i.e. no polynomial in the set can divide this polynomial. Such polynomials are called as irreducible polynomial as they cannot be factored into a polynomial with degree less than n.

The algorithmic implementation of polynomial multiplication uses repeated multiplication of a reduced polynomial by x. Thus, instead of finding the result of (x^2 * P) we find the result of (x * (x * p)). If the highest degree of polynomial P_1 is x^m while multiplying P_1 and P_2, we find the partial result of multiplying x^0, x^1, x^2, …..x^m by p_2. Although we may not require all the terms in actual multiplication, we calculate all the intermediate terms as each intermediate calculation depends on the previous result. This approach has two benefits; first, multiplication of a polynomial by x can be easily achieved by one-bit shifting of the n-bit word and second, the result needs to be reduced only if the power of the previous result is n-1(maximum). This reduction can be achieved by XORing the polynomial to be reduced with the modulus. The partial results of the operation can be implemented as follows:

1. If the most significant bit of the previous result is o, just shift the previous result one bit to the left.
2. If the most significant bit of the previous result is 1,
    - Shift it one bit to the left, and
    - XOR it with the modulus without the most significant bit.

Thus, multiplication of two polynomials in GF(2^n) can be achieved using shift-left and XOR operations.

Example: Let $P_1 = 000100110$ $(x^5 + x^2 + x)$

$P_2 = 10011110$ $(x^7 + x^4 + x^3 + x^2 + x)$

And modulus = 100011011 $(x^8 + x^4 + x^3 + x + x)$

Powers	**Shift-Left Operation**	**Exclusive-OR**
$x^0 \otimes P_2$	----	10011110
$x^1 \otimes P_2$	00111100	00111100 ⊕ 00011011 = 00100111
$x^2 \otimes P_2$	01001110	01001110
$x^3 \otimes P_2$	10011100	10011100
$x^4 \otimes P_2$	00111000	00111000 ⊕ 00011011 = 00100011
$x^5 \otimes P_2$	01000110	01000110

$$P_1 \otimes P_2 = 01000110 \oplus 01001110 \oplus 00100111 = 00101111$$
$$= x^5 + x^3 + x^2 + x + 1$$

Conclusion: Two polynomials represented in GF(2^n) field with a given modulus has been multiplied using an efficient algorithm.