Polynomials in GF(2) and CRCs

GF(2)

A Galois field, denoted as $GF(q)$, is a set of integers and two operations that have certain properties. One of the properties is closure – the result of any operation is an element of the field.

For example, $GF(2)$ includes two integers (0 and 1) and the addition and multiplication operations are defined as modulo 2 addition and multiplication.

Exercise 1: What are the possible results if we used values 0 and 1 but the regular definitions of addition and multiplication? Would this be a field?

Exercise 2: What logic function can be used to implement modulo-2 addition? Modulo-2 subtraction? Modulo-2 multiplication?

Representing Codewords as Polynomials

Thus far we've represented codewords as sequences of bits. We can also represent codewords as polynomials with coefficients from $GF(2)$. For example, the polynomial:

$$1x^3 + 0x^2 + 1x^1 + 1x^0 = x^3 + x^1 + 1$$

can be used to represent the codeword 1011.

Exercise 3: What is the polynomial representation of the codeword 01101?

Polynomials are used to describe codes because many properties of codes can be derived from the mathematical properties of polynomials.

Note that it is the coefficients of the polynomial that are important. The polynomial itself is never evaluated and the variable x that appears in these polynomials is just a dummy variable. These polynomials can thus also be viewed as binary numbers or bit strings where the order of each term indicates the bit position.

Polynomial Arithmetic

We can add, subtract, multiply and divide polynomials.

Exercise 4: What is the result of multiplying $x^2 + 1$ by $x^3 + x$ if the coefficients are regular integers? If the coefficients are values in $GF(2)$?

Digital Implementation of Polynomial Arithmetic

Implementation of arithmetic on polynomials with $GF(2)$ coefficients is simple. Flip-flops, organized as shift registers, hold the coefficient values and XOR and AND gates are used to compute modulo-2 addition and multiplication.

The bits corresponding to the codeword(s) are usually input and output sequentially (bit by bit) into the polynomial arithmetic circuits.

Exercise 5: Draw the schematic of a circuit that sequentially adds two polynomials. A circuit that multiplies the input by x^3. A circuit that multiplies the input by $x^2 + x$.

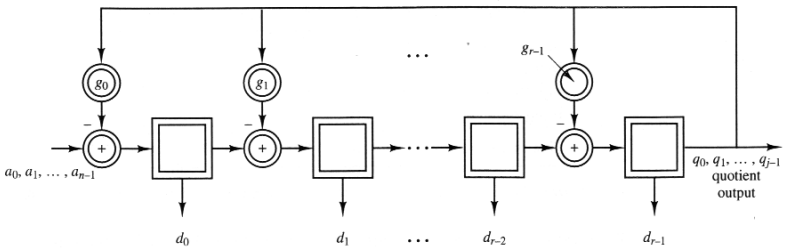
Cyclic Redundancy Checks

A Cyclic Redundancy Check (CRC) code is used to detect errors in codewords. The data to be transmitted, treated as a codeword, is multiplied by x^{n-k} by appending $n - k$ zero bits. This forms the message $M(x)$ which is divided by a *generator polynomial*, $G(x)$. The result is a quotient and a remainder:

$$\frac{M(x)}{G(x)} = Q(x) + R(x)$$

The remainder, $R(x)$ is then subtracted from the message by subsitituting $R(x)$ for the zeros in the least-significant $n - k$ bits of $M(x)$. The transmitted codeword is thus $M(x) - R(x)$. This new codeword will then be divisible by $G(x)$.

The receiver carries out the same polynomial division operation on the combination of the message bits and parity bits. If the remainder is not zero then an error has been detected.



Computing the CRC

Computing the CRC requires polynomial division. The process involves repeated subtraction of the generator polynomial from the message polynomial.

Exercise 6: What is result of dividing $x^3 + x^2$ by $x^3 + x + 1$?

A circuit to perform the division of the polynomials can be implemented using a shift register (SR) that holds the result of the intermediate remainder after each subtraction. The shift register only has to hold $(n - k)$ bits.

The diagram above[1] shows a circuit that performs polynomial division. The squares represent flip-flops in the SR with the most significant bit of the intermediate remainder in the right-most bit. The circles labeled g_i represent either a connection or no connection depending on the coefficient of $G(x)$. The circles with a plus represent modulo-2 addition (or subtraction) implemented using XOR gates. The input labelled a is the message polynomial $M(x)$.

The SR bits are initialized to zero so that the first $n - k$ (data) bits are loaded into the SR unchanged. At each subsequent step in the division the generator polynomial (represented by the presence or absence of the connections labelled g_i) is or isn't subtracted by the xor gates from the intermediate remainder in the SR depending on the value of the most significant bit of the quotient (rightmost bit of the SR). The next input bit is also appended to the intermediate remainder. At the end of the process the shift register holds the final remainder $R(x)$ which is appended to the message as the CRC.

There are alternative CRC generator implemenations that avoid the $n - k$-bit delay associated with initializing the SR.

[1] From *Error Control Systems* by S. B. Wicker.

Checking the CRC

At the receiver the same circuit can be used to divide the received message and the appended remainder polynomial by the generator polynomial. If the remainder is zero then the received polynomial must be a multiple of the generator polynomial. This is always the case when we subtract the remainder $R(x)$ from the message polynomial. Therefore if the remainder in the SR is non-zero then there must have been an error.

CRC Error Detection Performance

The ability of the CRC to detect an error depends on the type of error.

If the received bits are completely random then the probability of not detecting an error is the probability that a random sequence of $n - k$ bits matches the required checksum.

Exercise 7: What is the probability that a randomly-chosen set of $n - k$ parity bits will match the correct parity bits for a given codeword? Assuming random data, what is the undetected error probability for a 16-bit CRC? For a 32-bit CRC?

If the errors happen in a burst then a CRC of length $n - k$ is guaranteed to detect detect all burst of length $n - k$ or less and will detect many longer bursts, assuming the generator polynomial has been properly chosen.

Standard CRC Generator Polynomials

There are several CRC generator polynomials in common use. The most common lengths are 16 and 32 bits since these are multiples of 8 bits. All(?) IEEE 802 standards use the same 32-bit CRC polynomial typically called "CRC-32". The ITU has defined a 16-bit CRC generator polynomial ("CRC-16-CCITT") that is also used in various standards.