RICHMOND School of Arts & Sciences

University of Richmond
UR Scholarship Repository

Math and Computer Science Faculty Publications | Math and Computer Science

10-1998

Finding Cyclic Redundancy Check Polynomials for Multilevel Systems

James A. Davis
University of Richmond, jdavis@richmond.edu

Miranda Mowbray

Simon Crouch

Follow this and additional works at: http://scholarship.richmond.edu/mathcs-faculty-publications

 Part of the Mathematics Commons

Recommended Citation

Davis, James A., Miranda Mowbray, and Simon Crouch. "Finding Cyclic Redundancy Check Polynomials for Multilevel Systems." *IEEE Transactions on Communications* 46, no. 10 (October 1998): 1250-253.

This Article is brought to you for free and open access by the Math and Computer Science at UR Scholarship Repository. It has been accepted for inclusion in Math and Computer Science Faculty Publications by an authorized administrator of UR Scholarship Repository. For more information, please contact scholarshiprepository@richmond.edu.

Finding Cyclic Redundancy Check Polynomials for Multilevel Systems

James A. Davis, Miranda Mowbray, and Simon Crouch

Abstract—**This letter describes a technique for finding cyclic redundancy check polynomials for systems for transmission over symmetric channels which encode information in multiple voltage levels, so that the resulting redundancy check gives good error protection and is efficient to implement. The codes which we construct have a Hamming distance of 3 or 4. We discuss a way to reduce burst error in parallel transmissions and some tricks for efficient implementation of the shift register for these polynomials. We illustrate our techniques by discussing a particular example where the number of levels is 9, but they are applicable in general.**

Index Terms—**Communication systems, cyclic codes, multilevel systems, polynomials.**

I. Introduction

IN AN EFFORT to have high-speed transmission without exceeding emissions regulations, engineers have turned to transmissions which encode information in multiple voltage levels in place of the traditional binary transmission with two voltage levels. In order to ensure reliable transmission, some error detection needs to be included. This letter describes how to choose a cyclic redundancy check polynomial (CRC) in such a system, and we also discuss how to implement such a system, describing a technique which increases protection against burst errors, and efficient implementation of the shift register. We will assume that the transmission channel is the multilevel symmetric channel, so that if the voltage level received is in error, each of the possible wrong levels is equally likely to occur. For this channel, the most suitable distance measure is the Hamming distance.

We will focus our attention on a system with nine levels because of the particular application out of which this arose, but everything that we discuss can be done in a more general setting. We chose to work with cyclic codes over Z_9 because these codes are relatively easy to work with both theoretically as well as practically, but there are other choices. We could have constructed a cyclic code over GF(9), the finite field with nine elements, or we could have constructed a cyclic code over either Z_8 or GF(8) (finite ring or field with eight elements) and then had one signal level which is never used. The latter suggestion came from a referee and was not considered during the original work done on the code. Such a code would be no less efficient than our nine-level coding; but there is a reason for using all nine levels rather than, effectively, just eight, in that with nine levels there are more possible choices for code words which do not encode data, but are special signals used for synchronization information or to signal the end of the packet. The greater choice for nondata codewords allows run lengths to be kept short and allows for more efficient clock recovery. The other alternative mentioned, working over GF(9), does not have any advantages over Z_9 in terms of complexity of algebraic operations, and it seems harder to explain. Thus, we will construct codes over Z_9 in this letter.

We assume that the reader is familiar with CRC's over GF(2): see [1] for background. In Section II, we describe how to choose a CRC over GF(3) with Hamming distance 4—i.e., so that error resulting in a change in up to three symbols in the packet will be detected. In Section III, we show how to use this to construct CRC's over Z_9 with Hamming distance 4. In Section IV, we briefly discuss two implementation issues.

II. Polynomial of GF(3)

We will ultimately want to have error protection for a nine-level system. In order to do that, we first need to construct a system which will work for a three-level system. We will extend this in Section IV, but we state without proof at this stage that the nine-level system will have error detection capabilities at least as good as the three-level system used to generate it. We start with a well-known lemma. (The proofs of all the lemmas are in the Appendix.)

Lemma 1: Suppose that $P(x)$ is a primitive polynomial over GF(3) with degree $d > 1$, and that n is an integer less than $(3^d - 1)/2$. Then the code generated by P truncated at length n has a Hamming distance of at least 3.

In order to increase the Hamming distance of the code we are using to 4 (which would detect any error altering at most three symbols in the packet), we can modify a standard technique from binary transmissions: when working with cyclic codes over GF(2), any code whose generator is divisible by $(x+1)$ will detect any error altering an odd number of symbols in the packet. Thus, we can take a cyclic code over GF(2) whose Hamming distance is 3 and construct a new code whose Hamming distance is 4 by multiplying the generator by $(x + 1)$. This technique does not directly work over GF(3), but the following lemma demonstrates that we can perform a similar trick.

Lemma 2: Suppose that $g(x)$ is a generator of a truncated cyclic code over GF(3) of length n and Hamming distance at least 3. Then the code of length n generated by $(x - 1)g(x)$ has Hamming distance of at least 4 if and only if there are no codewords of the form $\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j}$, $m < n$ in the truncated code generated by $g(x)$.

Paper approved by S. B. Wicker, the Editor for Coding Theory and Techniques of the IEEE Communications Society. Manuscript received March 1, 1997; revised February 16, 1998.

J. A. Davis was with Hewlett-Packard Laboratories, Bristol, BS34 6QZ U.K., on sabbatical from the Department of Mathematics and Computer Science, University of Richmond, Richmond, VA 23173 USA.

M. Mowbray and S. Crouch are with Hewlett-Packard Laboratories, Bristol BS34 6QZ U.K.

Publisher Item Identifier S 0090-6778(98)07771-X.

0090–6778/98$10.00 © 1998 IEEE

We will show later that if $b(x)$ is the generator for a truncated cyclic code over GF(3) $= Z_3$ with Hamming distance h, then the truncated cyclic code over Z_9 whose generator is $b(x)$ [considered as a polynomial over Z_9 rather than GF(3)] will also have Hamming distance h. In our application, we work with packets which have a maximum length of 4500 bytes (the size of Token Ring packets). Each byte is to be encoded as a triple of nine-level symbols. (This coding is not as efficient as sending the bytes directly, but allows for special code words which do not encode data, as explained in the introduction.) Thus, the detection scheme will be a truncated nine-level cyclic code of length at most 13 500. Since the length of the packets that we will be sending is at most 13 500, if we find a primitive polynomial over GF(3) of degree at least 10, then by the first Lemma this will generate a truncated code with Hamming distance 3. Multiplying this primitive polynomial by $(x-1)$ will give a polynomial of degree at least 11, which by the second Lemma will generate a truncated code with Hamming distance 4 as long as we avoid codewords of the specified form. We want to organize the remaining symbols into triples like the rest of the packet, and hence we would like the degree of the primitive polynomial to be congruent to 2 mod 3. We constructed five or six primitive polynomials over GF(3) of degree 11 or 14, but all of these divided a polynomial of the form $\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j}$, $m < n$. There may be a polynomial with degree 11 or 14 which does not divide a polynomial of that form, but we decided to increase the degree to 17. We did find some primitive polynomials of degree 17 which do not divide any polynomial of this form. One example is $f(x) = 1 - x - x^2 - x^3 - x^4 - x^5 - x^6 - x^7 - x^8 - x^9 - x^{10} + x^{11} + x^{12} - x^{13} + x^{17}$.

We now describe the procedure which we followed in order to find this primitive polynomial of degree 17. Once we had found it, we used Mathematica [2] to check whether it divided any polynomial of the form $\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j}$, $m < n$; it does not, but if it had we would have used the same procedure to find another primitive polynomial. The procedure can be used in general to find primitive polynomials of a given degree over GF(3) and is derived from the results in [3].

1) Find an irreducible polynomial of degree 17 over $GF(3)$. The multiplicative group of GF(3^{17}) is cyclic with order $3^{17} - 1 = 2.1871.345\,11$, and so contains an element of order 1871. The minimal polynomial of this element (over GF_3) divides $x^{1871} - 1$ and is irreducible of degree 17. Type the command `Factor[x^(1871) - 1, Modulus →3]` into Mathematica. This gives 110 irreducible factors of degree 17 [and the obvious factor of (x-1)]. Choose one of these; there is no clear reason at this stage to prefer any one of these to any other. The polynomial that we chose was $x^{17} + x^{14} + x^{13} + x^9 + x^7 + x^5 + x^3 + x - 1$.
2) The multiplicative group of GF(3^{17}) = GF(3)$[x]/\langle x^{17} + x^{14} + x^{13} + x^9 + x^7 + x^5 + x^3 + x - 1\rangle$ is cyclic (true of all finite fields) of order $2(1871)(345\,11)$. We need to find a generator of this cyclic group, in other words an element of this order, which we will construct by multiplying together three elements whose orders are 2, 1871, and 34 511, respectively. So we need to find an element of order 34 511. If $(x^2+1)^{2(1871)}$ is not 1 in the field, then it will be an element of order 34 511. (If it is equal to 1, take another irreducible polynomial in place of x^2+1 and try again.) To do this, first use Mathematica to compute the powers of x with the command `Do[g[x_, i_] := PolynomialPowerMod[x, i, x^17 + x^14 +x^13 + x^9 + x^7 + x^5 + x^3+x - 1, 3], i, 1, 5000]` then rewrite $(x^2+1)^{2(1871)}$ as $((x^2+1)^{729})^5((x^2+1)^{81})((x^2+1)^9).((x^2+1)^7) = ((x^{1458}+1)^5(x^{162}+1)(x^{18}+1)(x^2+1)^7)$, exploiting the fact that raising polynomials to powers of 3 is easy mod 3; this can be computed using the Mathematica command `PolynomialMod[PolynomialRemainder[((g[x, 1458] +1)^5) * (g[x, 162] + 1) * (g[x, 18] + 1) * ((x^2 + 1)^7), x^17 +x^14 + x^13 + x^9 + x^7 + x^5 + x^3 + x - 1], 3]` which yields the result $1+x^2+x^4+x^5+x^{11}+x^{13}-x^{14} - x^{15}$. This polynomial has multiplicative order 34 511 in the finite field GF(3)$[x]/\langle x^{17} + x^{14} + x^{13} + x^9 + x^7 + x^5 + x^3 + x - 1\rangle$.
3) The element 2 has order 2 in the field. Multiplying this by x, which has order 1871, and by $1 + x^2 + x^4 +x^5+x^{11}+x^{13}-x^{14}-x^{15}$, we get that $x^{16}+x^{15}-x^{14}-x^{12}-x^6-x^5-x^3-x$ is a primitive element of the field. If we call this element b, then the minimum polynomial for the element is $f(y) = (y-b)(y-b^3)(y-b^9)\cdots(y-b^{3^{16}})$. In Mathematica, call b the function `[x_] := x^16 + x^15 - x^14 - x^12 - x^6 - x^5 - x^3 - x` and $b^{3^{n-1}}$ is the function `gn[x_] := PolynomialMod [PolynomialRemainder[ (g(n - 1)[x])^3, x^17 + x^14 +x^13 + x^9 + x^7 + x^5 + x^3 +x - 1, x], 3]`
4) The Mathematica command `PolynomialMod[Polynomial Remainder[(y - g1[x])(y - g2[x])(y - g3[x])... (y - g17[x]), x^17 + x^14 + x^13 + x^9 + x^7 + x^ 5 + x^3 +x - 1, x], 3]` yields the minimal polynomial for the primitive element that we have constructed and is therefore a primitive irreducible polynomial of degree 17. The output of this command is $f(y) = 1 - y - y^2 - y^3 - y^4 - y^5 - y^6 - y^7 - y^8 - y^9 - y^{10} + y^{11} + y^{12} - y^{13} + y^{17}$

We know by Lemma 2 that the truncated code generated by $f(y)$ has Hamming distance 3. There may well be other primitive polynomials of degree 17 with a greater minimum distance than this one, but we do not have a construction which will automatically produce such polynomials, and indeed the minimum distance of a given primitive polynomial is very difficult to determine. If we can demonstrate that there is a word of weight 4, we are done. Otherwise, we would have to come up with an argument why there cannot be words of weight 4. However, by using Lemma 3, we can construct a truncated code which we know has Hamming distance 4.

When we multiply $f(y)$ by $(y-1)$, we get $g(y) = -1 - y + y^{11} - y^{13} - y^{14} - y^{17} + y^{18}$. The truncated code generated by $g(y)$ has Hamming distance 4, so it will detect any errors which result in changes to at most 3 symbols.

The same method can be used to produce other polynomials of degree 18 over GF(3) which generate a truncated code with Hamming distance 4. Given a number of such polynomials, select the one with the smallest weight, because the smaller the weight the simpler the implementation of the CRC will be.

It is interesting to compare our construction with the ternary codes constructed by Kschischang and Pasupathy in [4]. In that paper there are tables of codes with large minimum distances and lengths of up to 50, whereas our ternary code starts at length $3^{17} - 1$ (about 10^8) and then gets truncated to a length which may be as large as 13 500. Reference [4] also gives some recursive construction methods for generating codes of longer length and large minimum distance. The most prolific construction, the $(u + v + w|2u + v|u)$ construction, produces codes which are not in general cyclic (or constacyclic), which means that it is not possible to encode or decode them with a shift register. The encoding and decoding difficulties for such codes makes them completely impractical for our application. There is, in addition, a more specialized recursive construction [4, Sec. III] which does generate cyclic codes. However, the degree of the codes generated in this construction is not bound, and the degree of a code with length near 13 500 constructed in this way is in general much larger than 17. Using a polynomial of large degree as a CRC is expensive in coding, decoding, and transmission time. In summary, the codes in [4] certainly have larger minimum distances than the ones we construct, but our construction yields polynomials which are more suitable for use as CRC's with large packet lengths.

III. Lifting to a Polynomial over Z_9

So far we have been discussing three-level codes, and we would like to have a nine-level code. We could do the usual coding theory following the same model as above over GF(9) rather than GF(3). We could, alternatively, lift the codes from the previous sections to a code over Z_9. This alternative has the advantage of having a natural connection to the symbols that are going to be transmitted over the wires. We will follow this method of working with a nine-level code.

Recent papers in the literature (see [5]) have described how to generalize binary cyclic codes to codes over Z_4. The general idea is to follow an algorithm to get a polynomial $g(x)$ with coefficients in Z_4 so that the natural homomorphism from $Z_4[x]$ to $Z_2[x]$ maps $g(x)$ to the generator of a binary cyclic code. In [5], $g(x)$ is required to be a "basic primitive polynomial," related to its use in defining Galois Rings. This process can be mimicked to lift polynomials from Z_3 to Z_9, yielding a basic primitive polynomial with coefficients in Z_9 which when reduced modulo 3 is just the original polynomial with coefficients in Z_3. However, we do not really need to have a basic primitive polynomial, only a polynomial that has at least as good error detection as the polynomial over Z_3 from which it is lifted.

Case $m = 3, r = 2$ of the lemma below implies that the truncated code over Z_9 generated by any polynomial over Z_9 has error detection capabilities at least as good as the polynomial over Z_3 whose coefficients are obtained by reducing the coefficients of the original polynomial mod 3.

Therefore, once we have used the technique of Section II to construct a polynomial over GF(3) $= Z_3$ which generates a code truncated at length n with Hamming distance 4, any polynomial over Z_9 which reduces mod 3 to the polynomial we have constructed will also generate a code truncated at length n with Hamming distance 4.

Lemma 3: Suppose $m > 1$ and $P(x)$ is a polynomial over Z with leading coefficient 1. Suppose that $P(x)$ mod (m) generates a truncated cyclic code over Z_m of Hamming distance h. Then for all $r \geq 1$, $P(x)$ mod (m^r) generates a truncated cyclic code (truncated at the same length) over Z_{m^r} of Hamming distance at least h.

IV. Implementation Issues

Two implementation issues are worth briefly mentioning. Burst errors occur when a sequence of consecutive symbols are compromised, typically by some external event. In general, CRC's will protect against bursts whose length is the same as the degree of the CRC because division by the polynomial $g(y)$ will yield a remainder different than the recorded remainder of the message. We can protect against errors of this type when the transmission occurs on multiple wires by using interleaving strategies similar to [6].

A second issue is adding and multiplying modulo 9. The multiplication can be considerably simplified by restricting the coefficients of the CRC polynomial to 0 and ± 1. In addition, we can insist on a clever mapping from binary to nine levels so that multiplication is simply transposition of binary symbols. (Note: in order to do any algebraic manipulations of the symbols, we need to map to binary and use standard gates.) The addition of multiple levels is always complicated by the fact that the only practical gates are binary, but again we can keep this as efficient as possible by a clever mapping from the nine-level symbols to binary.

Appendix
Proofs of Lemmas

Proof of Lemma 1: Let $Q(x)$ be a nonzero polynomial over GF(3), of degree less than n, with fewer than three nonzero coefficients. We want to show that $P(x)$ does not divide $Q(x)$. Clearly, $Q(x)$ must be of the form x^i or $x^i(1 \pm x^j)$ for some i, j. Since $P(x)$ is irreducible of degree greater than 1, $P(x)$ does not divide x, and hence [since the ring of polynomials over GF(3) is a unique factorization domain] it does not divide x^i for any i. So we have ($P(x)$ divides $Q(x)$) $\Rightarrow$ ($Q(x) = x^i(1 \pm x^j)$ for some $i, j < n$ and $P(x)$ divides $(1 \pm x^j)$) $\Rightarrow$ ($P(x)$ divides $(1 \pm x^j)(1 \mp x^j) = x^{2j} - 1$ for some $2j < 2n < 3^d - 1$). But $P(x)$ is primitive of degree d, and so by standard properties of primitive polynomials does not divide $x^k - 1$ for any $k < 3^d - 1$. The result follows. ■

Proof of Lemma 2: Suppose that $b(x)$ is in the truncated cyclic code generated by $(x - 1)g(x)$. It must be of the form $(x - 1)c(x)$, where $c(x)$ is in the truncated cyclic code generated by $g(x)$. Without loss of generality, the codeword $c(x)$ has a nonzero constant term: we can shift it if required. Since $b(x) = (x-1)c(x)$ is in the code generated by $g(x)$, we know that it cannot have fewer than three nonzero coefficients.

The different coefficients could correspond to different voltage levels at the physical layer of the code.

Suppose that it has exactly three nonzero coefficients. Now $b(1) = 0$, so these coefficients must all be equal. It follows that $c(x) = b(x)/(1 - x)$ is of the form $\pm(\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j})$ for some $m < n$. Conversely, if there is any codeword of this form in the truncated cyclic code generated by $g(x)$, multiplying such a codeword by $(x - 1)$ gives a codeword with exactly three nonzero coefficients in the truncated cyclic code generated by $(x - 1)g(x)$. The result follows. ■

One "physical" interpretation of the form of the excluded codewords is that during the transmission of a codeword of this form there are precisely three changes in the voltage level.

Proof of Lemma 3: The proof is by induction on r; case $r = 1$ is trivial. Let $Q(x)$ be a polynomial over Z with degree less than the truncation length, whose coefficients lie between 0 and $m^r - 1$ and with fewer than h nonzero coefficients. Suppose that $P(x)$ divides $Q(x)$ (mod m^r). Then $P(x)$ divides $Q(x)$ (mod m), and since $P(x)$ (mod m) generates a truncated a cyclic code of Hamming distance h it follows that $Q(x)$ is equal to the zero polynomial (mod m); in other words, $Q(x) = m.Q'(x)$ for some $Q'(x)$ whose coefficients lie between 0 and $m^{r-1} - 1$. Since the leading coefficient of $P(x)$ is 1, no prime factor of m divides $P(x)$. It follows that $P(x)$ divides $Q'(x)$ (mod m^{r-1}). But $Q'(x)$ has fewer than h nonzero coefficients, and so by induction on r, $Q'(x)$ must be the zero polynomial, and hence $Q(x)$ is the zero polynomial, which completes the proof. ■

REFERENCES

[1] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* Amsterdam, The Netherlands: North-Holland, 1977.

[2] S. Wolfram, "Mathematica, a system for doing mathematics by computer," in *The Advanced Book Program.* Redwood City, CA: Addison-Wesley, 1988.

[3] R. Lidl and H. Niederreiter, "Finite fields," in *Encyclopedia of Mathematics and its Applications,* Gian-Carlo Rota, Ed. Reading, MA: Addison-Wesley, 1983, vol. 20.

[4] F. R. Kshischang and S. Pasupathy, "Some ternary and quaternary codes and associated sphere packings," *IEEE Trans. Inform. Theory,* vol. 38, no. 2, pp. 227–246, 1992.

[5] A. R. Hammons, Jr., P. V. Kumar, A. R. Calderbank, N. J. A. Sloane, and P. Solé, "The Z_4-linearity of Kerdock, Preparata, Goethals, and related codes," *IEEE Trans. Inform. Theory,* vol. 40, pp. 301–319, 1994.

[6] J. Jedwab and S. E. Crouch, "Method and system for communicating data," U.S. Patent 5 410 309, Apr. 25, 1996.