GF(2)上の多項式を法とする演算の高速化[†]

柏 木 潤*・森 内 勉**

A Fast Method for Arithmetic Modulo a Polynomial Over GF(2)

Hiroshi KASHIWAGI* and Tsutomu MORIUCHI**

A fast method for arithmetic modulo a polynomial over GF(2) is proposed. In this method, any polynomial can be calculated modulo any polynomial over GF(2) at high speed, while no restrictions are imposed on the dividend or the divisor polynomial. The calculation time is shown to be proportional to the logarithm of the order of the dividend polynomial.

The fast method is compared with the conventional division algorithm in terms of the calculation time, so that it becomes clear that the fast method is much faster than the conventional one.

The application to a delayed version of m-sequence of high degree is demonstrated.

ある多項式を法とする演算は、通常は多項式の割算によって行われるが、被除多項式の次数が大きくなると、それに比例して計算時間が増大する。そこでなんらかの高速化の工夫が必要となる。Miller³⁾などは、法とする多項式が3項からなるときにのみ有効な高速演算法を示している。Van Luyk⁴⁾は、別の演算法を示しているが、被除多項式が1項のときに有効な方法である。

本論文では、被除多項式や除多項式になんらの制限はなく、かつ高速に法演算が行える方法を提案し、従来の方法による演算時間と比較した結果、本方法が非常に有効であることを示している。

2. 高速演算法

いま、 $f(x)$ をつぎのような GF(2) 上の n 次多項式とし、 $f(x)$ を法とする演算を考える。

$$f(x) = \sum_{i=0}^n f_i x^{n-i}, \quad f_i \in GF(2), \quad f_0 \neq 0 \quad (1)$$

係数 f_i は、0 または 1 であり、係数同志の演算は、GF(2) の上で行われる。すなわち、乗算および加算は mod 2 で行われる。

$g(x)$ を GF(2) 上の l 次 ($l \geq n$) の多項式とすると、

$$g(x) = Q(x)f(x) + R(x) \quad (2)$$

とかける。ここに $Q(x)$ は商多項式であり、 $R(x)$ は剰余多項式である。 $R(x)$ の次数は n より小さい。 $f(x)$ を法とする演算では、 $R(x)$ を求めることが目的となる。その際、 $Q(x)$ は求める必要はない。

$g(x)$ を $f(x)$ で割って行く通常の割算では、 $l-n$ 次の多項式 $Q(x)$ の各項を次数の高い順に1項ずつ求めてゆくから、 $Q(x)$ の項数は最大 $l-n+1$ 、 $f(x)$ の項数が最大 $n+1$ であるから、最大 $(l-n+1)(n+1)$ 回の x のべきの加算および各加算ごとに x のべきを大きい順に並べる操作が必要になる。減算および配置換えの演算時間は、ほぼ加算の演算時間と同じであるとすると、結局、 $(l-n+1)(n+1)(d_1+2)$ 回の加算が必要

1. ま え が き

M系列は、擬似不規則信号として、計測や制御の分野で広く用いられている。一般に k レベルの M 系列が考えられるが、ふつうに用いられるのは、2-レベルの M 系列である。2-レベルの M 系列を発生するには、二つの要素をもつ有限体、ガロア体 GF(2) のうえの原始多項式と呼ばれる多項式を特性多項式として、シフトレジスタにより容易に発生される¹⁾。

この M 系列を扱うとき、しばしば GF(2) 上の多項式を法とする演算が必要となる。たとえば、ある M 系列よりある時間遅れた M 系列を、シフトレジスタのいくつかの段の出力の mod 2 の和によって得たいとき、特性多項式を法とする演算が必要となる²⁾。このほか、原始多項式を求める場合にも、同様の計算が必要となる。

[†] 第 22 回自動制御連合講演会で一部発表 (昭 54・10)

* 熊本大学工学部 熊本市黒髪 2-39-1

** 八代工業高等専門学校 八代市平山新町 2627

* Faculty of Engineering, Kumamoto University, Kumamoto

** Yatsushiro Technical College, Yatsushiro

(Received May 23, 1981)

(Revised October 29, 1981)

となる. ここに d_1 は剰余多項式の項数の平均である. したがって, 通常の割算で剰余を求める方法の計算時間は, l の増大と共に, ほぼ l に比例して大きくなる.

そこで, (2)式において, $Q(x)$ が再び n 次以上なら, $Q(x)=Q_1(x)f(x)+R_1(x)$ とかけるから, その手順を繰返すと,

$$g(x)=Q_m(x)f^{m+1}(x)+R_m(x)f^m(x) \\ +\cdots+R_1(x)f(x)+R(x) \quad (3)$$

が得られる. $Q_m(x)$ および $R_i(x)$ ($i=1, 2, \dots, m$) は, $n-1$ 次以下の多項式である. もし $f^m(x)$ の形の多項式が簡単に得られるのであれば, $f(x)$ で割ってゆくより, $f^m(x)$ で割ってゆくほうが計算が速いはずである. 任意の m に対しては $f^m(x)$ は簡単には求まらないが, m が 2^k (k : 正整数) の形のときは, つぎのように容易に求まる. すなわち,

$$f^{2^k}(x)=f_0x^{2^kn}+f_1x^{2^k(n-1)}+f_2x^{2^k(n-2)}+\cdots+f_n \quad (4)$$

(4)式は, GF(2) 上の多項式の性質

$$\{f(x)\}^2=f(x^2) \quad (5)$$

を繰返して用いることにより導かれる ($f^{2^k}(x)$ は, $f(x)$ と同じ項数をもつことに注意).

そこで, (3)式をつぎのようにまとめる. すなわち, $l \geq 2^kn$ となる k を選んで,

$$g(x)=P_0(x)f^{2^k}(x)+P_1(x)f^{2^{k-1}}(x) \\ +\cdots+P_k(x)f(x)+R(x) \quad (6)$$

ここに, $P_i(x)$ は, $n \cdot 2^{k-i}-1$ 次以下の多項式であり, 通常最大 $n \cdot 2^{k-i}$ 個の項からなると考えられるが, 実は, この項数は大変少ないことが, つぎのようにしていえる. 簡単のため, $g(x)$ が 1 項のみからなる場合を考える. $g(x)=x^l$ として, つぎつぎに $f^{2^k}(x)$ で割ると, つぎようになる.

$$g(x)=x^l \\ =x^{l-2^kn}f^{2^k}(x)+x^{l-2^kn}(f_1x^{2^k(n-1)} \\ +f_2x^{2^k(n-2)}+\cdots+f_n) \\ =x^{l-2^kn}f^{2^k}(x)+f_1x^{l-2^k}+f_2x^{l-2 \cdot 2^k} \\ +f_3x^{l-3 \cdot 2^k}+\cdots+f_nx^{l-2^kn} \quad (7)$$

ここで

$$f_1x^{l-2^k}=f_1x^{l-2^k-2^kn}f^{2^k}(x) \\ +f_1x^{l-2^k-2^kn}(f_1x^{2^k(n-1)}+f_2x^{2^k(n-2)} \\ +\cdots+f_n)$$

とかけるから, (7)式は,

$$g(x)=x^{l-2^kn}(1+f_1x^{-2^k})f^{2^k}(x) \\ +(f_1+f_2)x^{l-2 \cdot 2^k}+(f_1f_2+f_3)x^{l-3 \cdot 2^k} \\ +(f_1f_3+f_4)x^{l-4 \cdot 2^k}+\cdots+(f_1f_{n-1} \\ +f_n)x^{l-n \cdot 2^k}+f_1f_nx^{l-(n+1)2^k}$$

となる. さらに,

$$(f_1+f_2)x^{l-2 \cdot 2^k} \\ =(f_1+f_2)x^{l-2 \cdot 2^k-2^kn}f^{2^k}(x) \\ +(f_1+f_2)x^{l-2 \cdot 2^k-2^kn}(f_1x^{2^k(n-1)} \\ +f_2x^{2^k(n-2)}+\cdots+f_n)$$

とかけるから,

$$g(x)=x^{l-2^kn}(1+f_1x^{-2^k}+(f_1+f_2)x^{-2 \cdot 2^k})f^{2^k}(x) \\ +(f_1+f_3)x^{l-3 \cdot 2^k}+(f_1f_3+f_4+f_1f_2 \\ +f_2)x^{l-4 \cdot 2^k}+\cdots+(f_1f_{n-1}+f_n \\ +(f_1+f_2)f_{n-2})x^{l-n \cdot 2^k} \\ +(f_1f_n+(f_1+f_2)f_{n-1})x^{l-(n+1)2^k} \\ +(f_1+f_2)f_nx^{l-(n+2) \cdot 2^k}$$

同様にして,

$$g(x)=x^{l-2^kn}(1+f_1x^{-2^k}+(f_1+f_2)x^{-2 \cdot 2^k} \\ +(f_1+f_3)x^{-3 \cdot 2^k})f^{2^k}(x) \\ +(f_4+f_1f_2+f_2+f_1)x^{l-4 \cdot 2^k}+\cdots \quad (8)$$

となる. すなわち,

$$P_0(x)=x^{l-2^kn}(1+f_1x^{-2^k}+(f_1+f_2)x^{-2 \cdot 2^k} \\ +(f_1+f_3)x^{-3 \cdot 2^k}+\cdots) \quad (9)$$

となり, 各項の次数は 2^k ずつ減ってゆく. $P_0(x)$ の次数は, $n \cdot 2^k-1$ 次以下であるから, $P_0(x)$ の項数は最大 n となる.

同様の考察によって, $P_i(x)$ の項数も最大 n となる. したがって, (6)式の各項を求めるのに必要な x のべきの加算回数は, 最大 $n(n+1)(k+1)$ 回となり, これに各加算ごとに x のべきを大きい順に並べる操作時間が加えられるから, 結局, $n(n+1)(k+1)(d_2+2)$ 回の加算が必要となる. ここに d_2 は剰余多項式の項数の平均である. したがって, 演算時間は l の増大と共に, ほぼ l の対数に比例して増大する.

通常の割算を用いた場合の加算回数に対する (6)式を用いた方法の加算回数の比を C_r とすると, $l=2^kn+r$ ($0 \leq r < 2^kn$) として,

$$C_r=\frac{n(n+1)(k+1)(d_2+2)}{(2^kn+r-n+1)(n+1)(d_1+2)}<\frac{k+1}{2^k} \cdot \frac{d_2+2}{d_1+2}$$

となる. d_1 と d_2 は, 剰余の平均項数であるから, 割算の仕方では大きな差はないと考えると,

$$C_r<\frac{k+1}{2^k} \quad (10)$$

となる. 仮に, $k=10$ とすれば, $C_r < 1/100$ となる.

Table 1 は, 通常の割算による方法と, 本方法との演算時間の比較の例を示したものである. l は各 k に対して, 2^kn と $2^{k+1}n$ との中間付近になるように選んだ. この表から, (10)式の評価がオーダーとしておおよそ正しいことがわかる. (10)式を導くにあたって数えた計算の回数は最大の場合を考えたので, その比

Table 1 Comparison of calculation time by the fast method with that by the conventional division method

l	k	Computation time in ms		$T1/T2$	$\frac{k+1}{2^k}$
		Fast method T1	Division method T2		
3750	7	122	1989	0.0613	0.0625
7500	8	149	4066	0.0366	0.0352
15000	9	179	8136	0.0220	0.0195
30000	10	205	16414	0.0125	0.0107
60000	11	227	32831	0.0069	0.0059
120000	12	238	66011	0.0036	0.0032

として求められた (10) 式は、実際には必ずしも成り立つとは限らないが、およその評価として使用してよいことがわかる。

Fig. 1 は、Table 1 とは別の計算機で計算した高速演算法による演算時間を示す。演算時間は、被除多項式の次数 l の対数関数的にふえていることがわかる。

演算の手順を示すために、例を示そう。いま、 $g(x) = x^{30} + x^{20}$ 、 $f(x) = x^7 + x^4 + x^3 + x^2 + 1$ とする。 $f^{2^k}(x)$ で割るという操作は、 $f(x)$ の各項の x のベキの n の補数の 2^k 倍を順に剰余の最高次より引くことになるから、この場合、3, 4, 5, 7 の 2^k 倍を順に減ずることになる。このようにしてできた数のうち、すでにある剰余のベキと同じものがあれば打ち消す。この過程を図にすると、**Fig. 2** のようになる。簡単のため、 x のベキのみを示した。表の形にかくと **Table 2(a)** のようになる。下線をひいた数がそのときの剰余の最高次のベキで、それがつぎの行のかって内の数に「分解」されることを意味する。これを (b) のように、大きさの順に並べておくと見やす

Fig. 1 Calculation time vs. $\log_{10} l$ in case where x^l is to be obtained modulo $f(x) = x^{100} + x^{50} + x^2 + x + 1$

Fig. 2 Schematic diagram showing the procedure of the fast method

い。(b)の行の最初の数が n より小さくなった時点で計算を終了する。

被除多項式の項数がふえれば、一般には計算時間が増えるが、逆に減る場合もある。途中で相殺する項がふえるためである。その例を **Table 3** に示す。Table 2 より、2 ステップだけ早く計算を終了している。

計算手順をまとめると、つぎのようになる。

- (1) 被除多項式の最高次のベキを見出し、それを l とする。 $l \geq 2^k n$ となる最大の k を探す。
- (2) x^l を $f^{2^k}(x)$ で割る。
- (3) 剰余多項式の次数が n 以上なら、ステップ (1) → (3) を繰返す。 n より小さいなら終了する。

Table 2 Illustrative example of the fast method in case where $g(x) = x^{30} + x^{20}$ is to be obtained modulo $f(x) = x^7 + x^4 + x^3 + x^2 + 1$

Step	Remainder polynomial	
	(a)	(b)
1	<u>30</u> , 20	30, 20
2	(18, 14, 10, 2), <u>20</u>	20, 18, 14, 10, 2
3	<u>18</u> , 14 , 10 , 2, (14 , 12, 10 , 6)	18, 12, 6, 2
4	(18 , <u>10</u> , 8, 4), 2, 14 , 6	10, 8, 6, 4, 2
5	(7, 10 , 5, 3), <u>8</u> , 4, 2, 14	8, 7, 5, 4, 3, 2
6	7, 10 , 14 , (14 , 10 , 1), <u>1</u> , 14 , 2	7, 2, 1
7	(4, 3, 14 , 0), 1, 14	4, 3, 1, 0

Table 3 Illustrative example of the fast method in case where $g(x) = x^{20} + x^{10} + x^0$ is to be obtained modulo $f(x) = x^7 + x^4 + x^3 + x^2 + 1$

Step	Remainder polynomial	
	(a)	(b)
1	<u>30</u> , 20, 10	30, 20, 10
2	(18, 14, 10 , 2), <u>20</u> , 10	20, 18, 14, 2
3	<u>18</u> , 14 , 2, (14 , 12, 10, 6)	18, 12, 10, 6, 2
4	(18 , 14 , <u>8</u> , 4), 2, 14 , 10 , 6	8, 6, 4, 2
5	(5, 14 , 3, 1), <u>1</u> , 2, 6	6, 5, 3, 2, 1

なお、本方法は、剰余が n より大きなある次数以下でよい場合にも、途中で演算を止めればよいから、容易に適用することができる。たとえば、次節で述べる遅れたM系列を得る場合、M系列発生シフトレジスタにいくつかの段数をつなぎ合せて剰余多項式の次数をある程度大きくしておけば、タップの数が著しく減る場合などには、本方法を有効に使用することができる。

3. 遅れた M 系列発生への応用

あるM系列よりある位相だけ遅れたM系列を、シフトレジスタの段の適当な線形結合で発生することができる。その結合を知るには、位相のずれを l とするとき、 x^l を特性多項式 $f(x)$ で割った剰余多項式を求めればよい²⁾。この問題に対し、特に l が大きいときに本高速演算法が有用である。

例として、98 次のM系列に対して、半周期位相のずれたM系列を得たいとする。周期 $T=2^{98}-1$ であるから、半周期の位相のずれは、 $l=(T+1)/2$ である。特性多項式を $f(x)=x^{98}+x^{84}+x^{32}+x^3+1$ として、 $x^l \bmod f(x)$ を本高速演算法で求めると、

$$x^l = x^{97} + x^{83} + x^{48} + x^{41} + x^{31} + x^{15} + x^2 \quad \bmod f(x) \quad (11)$$

が容易に求められる。すなわち、シフトレジスタの i 段目の出力を $m(i)$ とすると、

$$m(2^{97}) = m(97) + m(83) + m(48) + m(41) + m(31) + m(15) + m(2) \quad (12)$$

が得られる。この場合、 $l=2^{97} \approx 10^{32}$ であって、通常の割算を行うことはほとんど不可能であり、またM系列発生器を作って、 l 回シフトレジスタをまわしたあ

と、シフトレジスタの内容をみる（これができれば初期値と連立させて線形結合の関係を求めることができる）ことも l が大きすぎて、ほとんど不可能である。

4. あとがき

GF(2)上の多項式を法とする演算について、高速演算法を提案した。本方法は、被除多項式や除多項式になんらの制限もなく、高速に法演算が行える特徴をもつ。本質的には、Millerなどの法多項式が3項よりなる場合の演算法の拡張になっている。通常の割算による方法との比較によって本方法の高速性を示した。また遅れたM系列発生への応用の例を挙げて、本方法の有用性を示した。なお、一般の $GF(p)$ の場合も(5)式と同様の式が成立つから、本方法と同様の方法で高速化をはかることが可能である。

終りに、本研究に関し御協力を得た熊本大学大学院 菅 晴彦氏（現在、東芝勤務）に感謝の意を表する。

参考文献

- 1) W.W. Peterson: Error-Correcting Codes, MIT Press (1961)
- 2) A.C. Davies: Delayed Versions of Maximal-Length Linear Binary Sequences, Electron. Lett., **1**-3, 61/62 (1965)
- 3) A.J. Miller, A.W. Brown and P. Mars: A Simple Technique for the Determination of Delayed Maximal Length Linear Binary Sequences, IEEE Trans. Comput., **C-26**-8, 808/810 (1977)
- 4) A.N. Van Luyn: Shift resistor Connections for Delayed Version of m-Sequences, Electron. Lett., **14**-22, 713/715 (1978)
- 5) 柏木, 森内, 菅: GF(2)上の多項式を法とする演算の高速化, 第22回自動制御連合講演会前刷, 147/148 (1979)